Carving network traces with bulk_extractor and tcpflow

Simson L. Garfinkel
Associate Professor, Naval Postgraduate School

February 3, 2012

http://afflib.org/

NPS is the Navy's Research University.

Location: Monterey, CA

Students: 1500

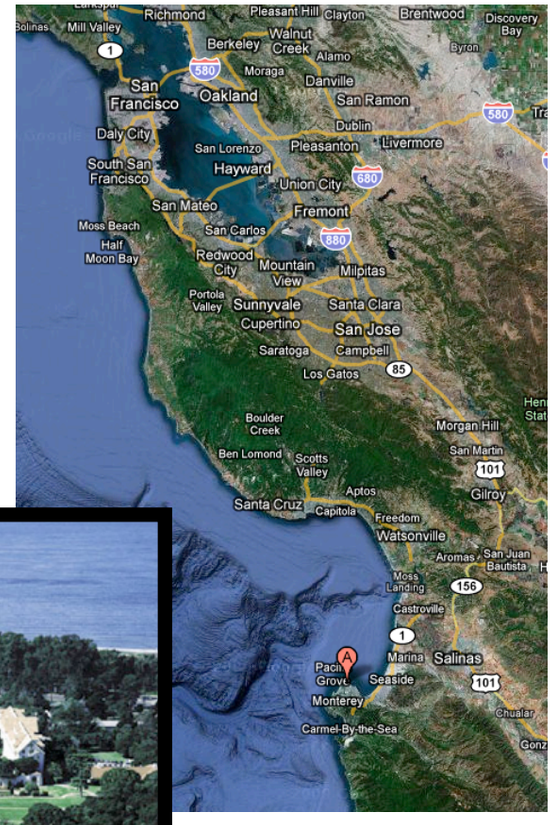
- US Military (All 5 services)
- US Civilian (Scholarship for Service & SMART)
- Foreign Military (30 countries)
- *All students are fully funded*

Schools:

- Business & Public Policy
- Engineering & Applied Sciences
- Operational & Information Sciences
- International Graduate Studies

NCR Initiative:

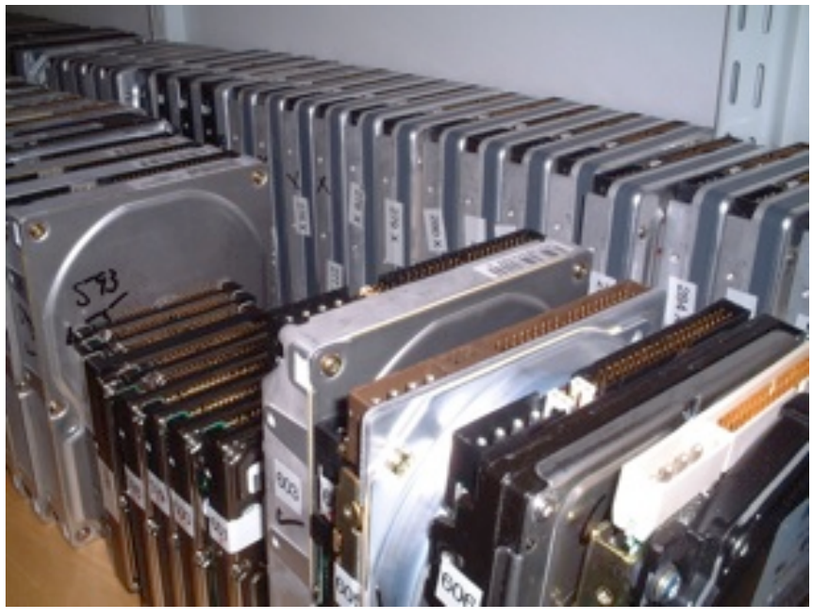
- 8 offices on 5th floor, 900N Glebe Road, Arlington
- 4 professors, 2 staff, 4 contractors



Current NPS research thrusts in computer forensics

Area #1: End-to-end automation of forensic processing

- Digital Forensics XML Toolkit
- Disk Image -> Power Point



Area #2: Bulk Data Analysis

- Statistical techniques (sub-linear algorithms)
- Similarity Metrics

Area #3: Data mining for digital forensics

- Automated social network analysis (cross-drive analysis)

Area #4: Creating Standardized Forensic Corpora

- Freely redistributable disk and memory images, packet dumps, file collections.

Network forensics using data at rest

Network information has significant forensic value

Devices are (invariably) connected to network(s)

Users, applications, and operating systems interconnect (both explicitly and in the background)

Network activity is invaluable forensic information:

- Commonly visited web sites
- Network attachment point(s)
- File transfer

etc.

Disks contain network forensic data

Traditional disk forensics (MEDEX) makes extensive use of residual network information.

- Logfiles

```
Jul 27 00:00:00 mncrnpsedu postfix/pickup[2767]: 6CBC66D2EC5: uid=501
from=<simsong>
Jul 27 00:00:00 mncrnpsedu postfix/cleanup[2770]: 6CBC66D2EC5: message-
id=<20110727040000.6CBC66D2EC5@mncrnpsedu.local>
Jul 27 00:00:00 mncrnpsedu postfix/qmgr[2768]: 6CBC66D2EC5:
from=<simsong@mncrnpsedu.local>, size=556, nrcpt=1 (queue active)
```

- Email Messages

```
Received: from [172.20.48.199] (172.20.48.199) by smtp.nps.edu
(172.20.24.111)
 with Microsoft SMTP Server (TLS) id 14.1.355.2; Thu, 2 Feb 2012 13:44:32
 -0800
Message-ID: <4F2B03C0.6060601@nps.edu>
Date: Thu, 2 Feb 2012 13:44:32 -0800
To: Simson Garfinkel <slgarfin@nps.edu>
```

Hard drives have IP packets as well.
This is an opportunity for traditional network forensics!

Sources of packets:

Attacker ran a packet sniffer; pcap file found on compromised machine.

- —*File might be allocated or deleted*
- —*Fragments of the file might be present in unallocated space*

RAM ... swapped to disk (PAGEFILE.SYS)

RAM ... hibernated (WIN386.SWP)

Most forensic tools will ignore these packets

Mainstream carvers don't have signatures for packets.

EnCase and FTK don't have viewers for pcap

But packets have great stuff!

Data that might not have been written to disk.

IP addresses — Information about connections in process

MAC addresses — Information about hosts that were physically connected.

Beverly, Garfinkel and Cardwell, "Forensic Carving of Network Packets and Associated Data Structures", 2011

Network Carving Prior Work:

- Network data in ASCII form, e.g. web cache, cookies, etc.
- Fully-qualified Domain Names, e.g. www.cnn.com
- E-Mail Domain Names, e.g. rob@nps.edu
- “Dotted Quads,” e.g. 157.166.224.26

Volatility [Walters]

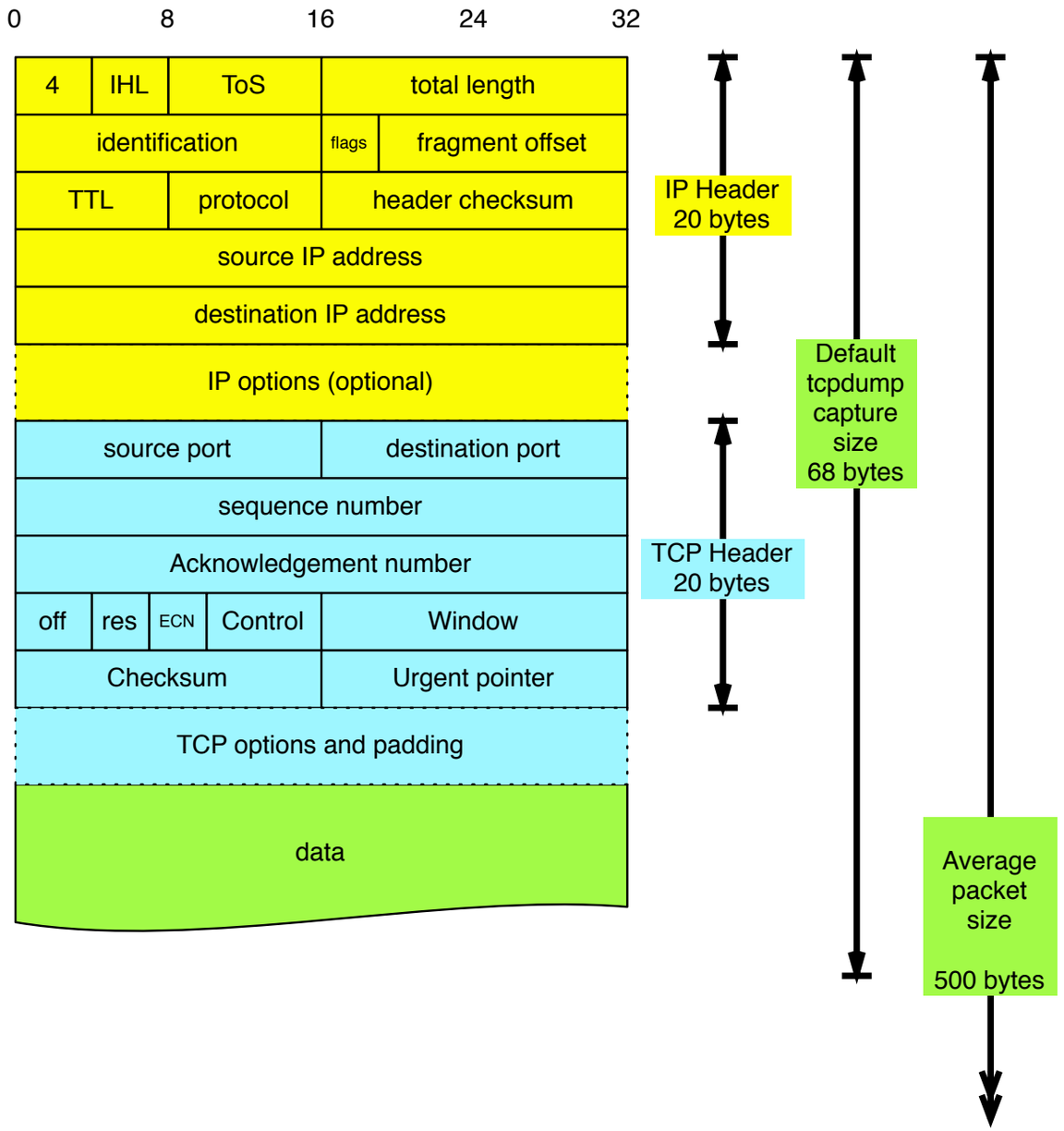
- Volatility memory analysis framework “connscan2” closest in spirit to our effort
- Carves memory dumps and intact Windows hibernation files for Windows TCP connection structures

Our approach for finding packets:

- Don’t look at network traffic on the wire
- Don’t look at logs
- Instead — Look at disk images.

No prior evidence that packets would be on disks.

Many thought kernel would separate packet headers & payload.



But we were pretty sure that packets would be found, if we looked!

Our Contributions

Evidence-based discovery

Created ground-truth corpus with known TCP connections.

Using ground-truth corpus, develop methodology for carving binary network data:

- *Windows _TCPT_OBJECT IP Packets*
- *Ethernet Frames*
- *Socket Structures*

Technical improvements:

Opportunistic hibernation decompression, including fragments

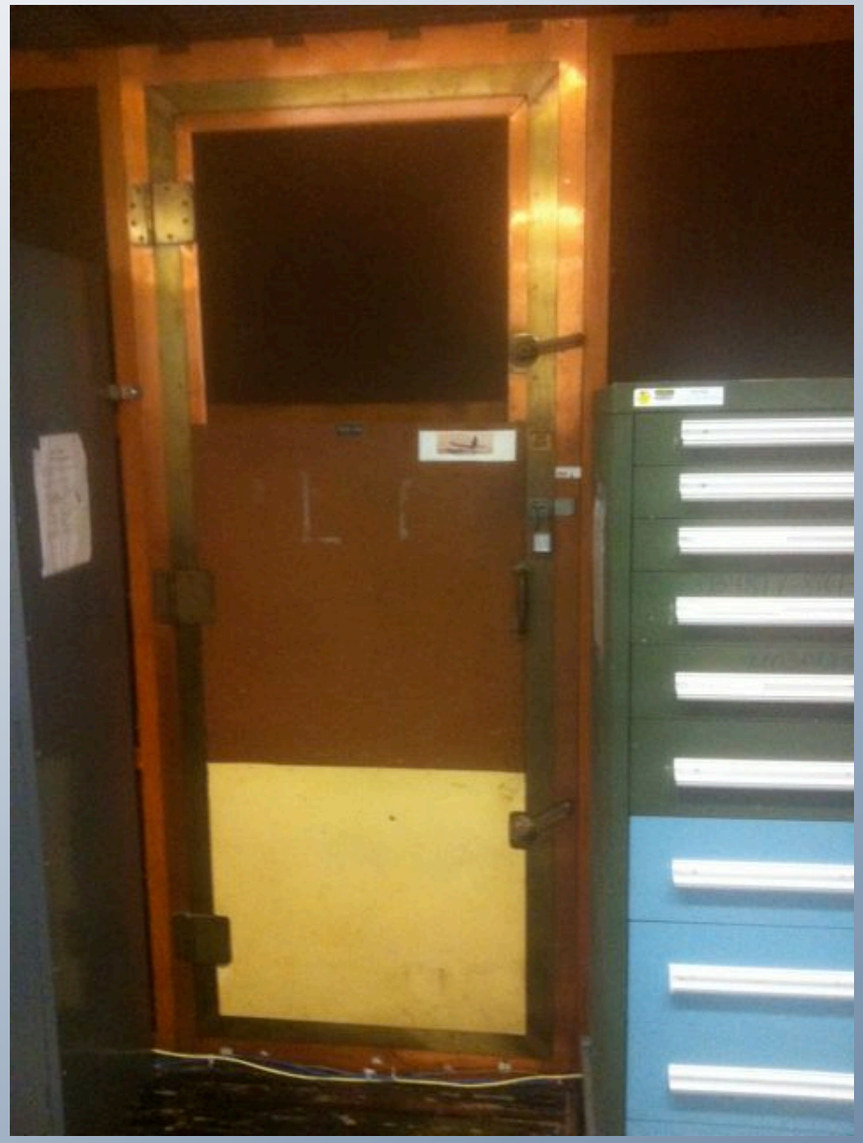
Filtering and Validation techniques

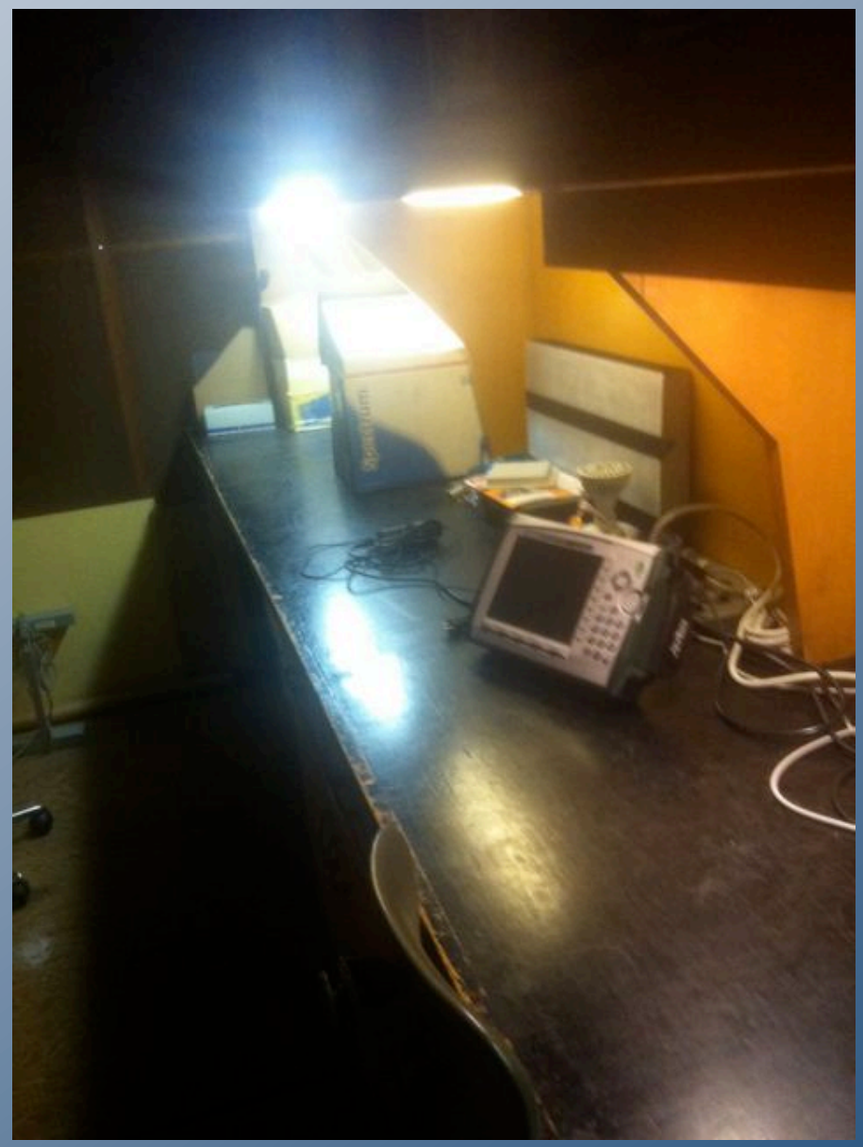
Integration into bulk_extractor

Expansion to handle IPv6

Evaluation on real data:

Search of 1800-drive corpus for live packets.





NPS Faraday Cage

Evidence-based Discovery

We created a series of disk images likely to have packets.

VM / Desktop / Laptop Computers

- Experimented with: Windows, OSX, Linux
- Wipe drive with DBAN to ensure no residual data
- Clean OS install
- Establish several HTTP and SCP connections to known destination IPs
- Image the host's disk after each connection

Android Phones (Cardwell '2011)

- Factory-fresh Android phones.
- Connected to 802.11, BlueTooth, and GSM base station in Faraday Cage

Searching for Signatures

We *know* the signature of the IP, Mac, BTADDR, and GSM stations

- IPv4 address is simply a 32-bit integer
- Chances of a false positive — $1 : 2^{32}$

Find the addresses in the memory

- Look at local context.
- Determine if there are common predecessor/successor patterns.

It is tempting to write the heuristics first...

- e.g. “a four byte IP address is preceded by a variable fragment field and a protocol field equal to six”
- But heuristics brittle, difficult to define, and (possibly) inaccurate...

2-Gram Frequency Analysis produced surprising results.

IPv4 2-Gram Analysis

Predecessor Freq		**Successor Freq**	
Count	2-gram	Count	2-gram
434	0x4000	428	0x0016
421	0x0800	426	0x0447
368	0xF202	412	0x0A79
368	0x4006	374	0xAC14
368	0x4508	374	0x694A
368	0x0017	41	0x0000
66	0x4500	12	0x2000
...	...	...	...

0x4000 — IP Flags = Don't Fragment

0x0800 — Ethernet "type" = IP

0x4508/0x4500 — IPv4 w/ & w/o TOS

0x4006 — IP TTL=64, Prot=TCP (TTL=64 doesn't generalize)

This gave us a carving signature for IPv4 data

Signatures: Manual Inspection + *N*-Gram Analysis

Key

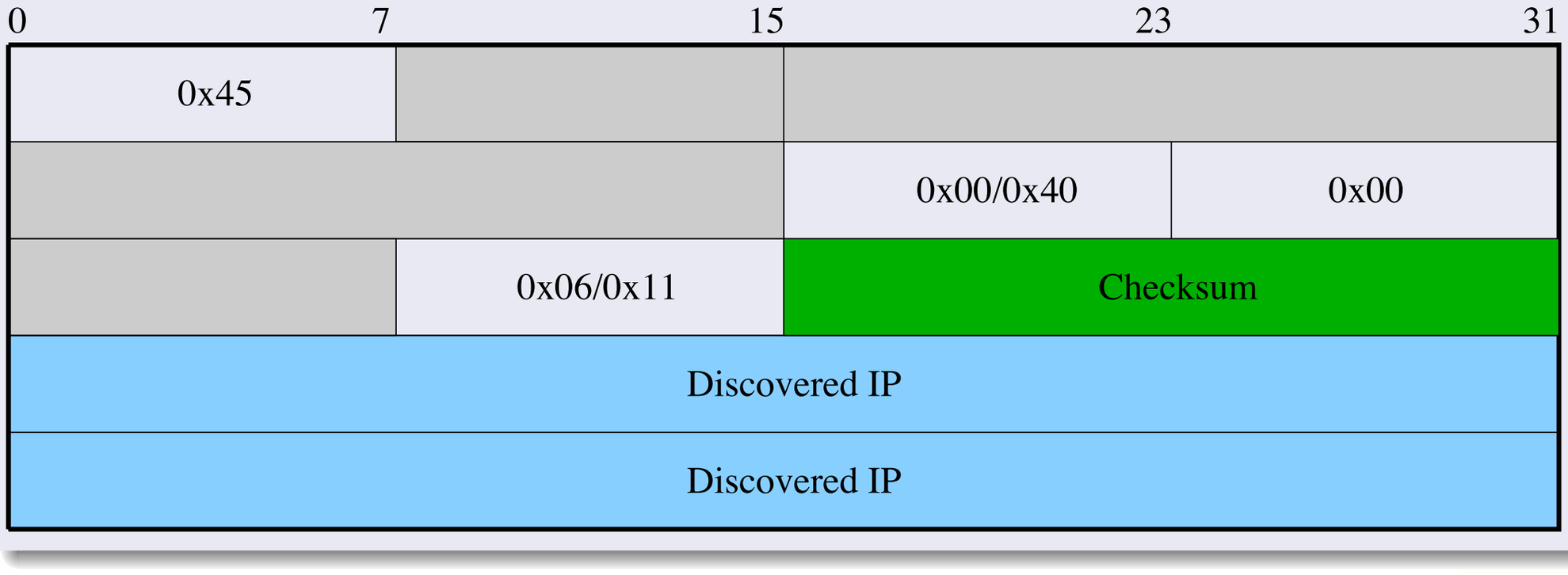
= Required

= Carved

= Wildcard

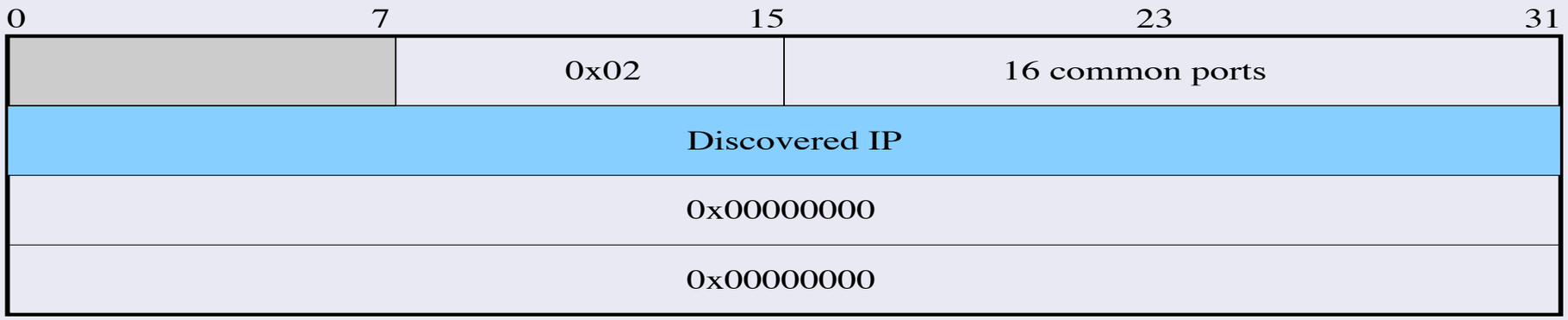
= Validation

IP Carving

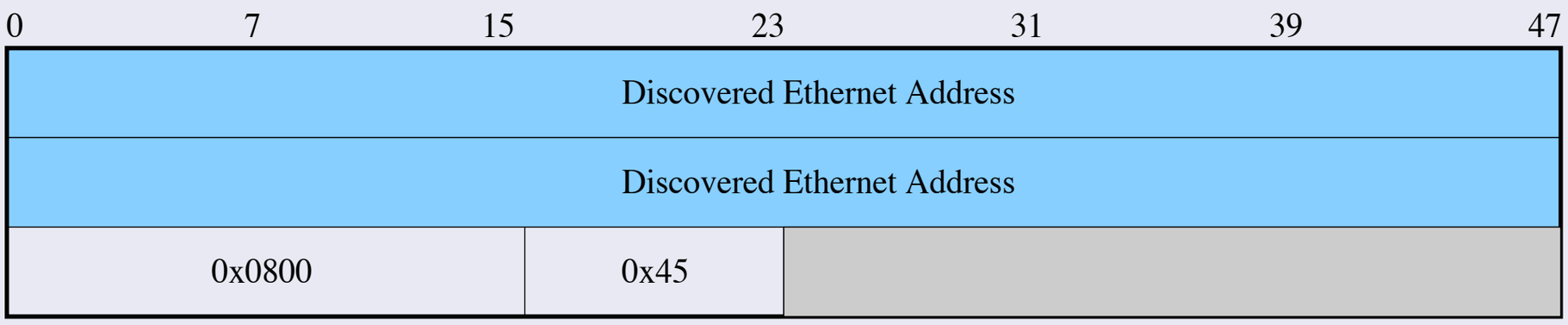


Signatures for Sockets and Ethernet are less discriminating.

Socket Carving



Ethernet Carving



bulk_extractor is a tool for bulk data analysis

Stream-Based Disk Forensics:
Scan the disk from beginning to end; do your best.

0 → 1TB

3 hours, 20 min to *read* the data

1. Read all of the blocks in order.
2. Look for information that might be useful.
3. Identify & extract what's possible in a single pass.

Primary Advantage: Speed

No disk seeking.

Potential to read and process at disk's maximum transfer rate.

Potential for intermediate answers.

Reads all the data — allocated files, deleted files, file fragments.

Separate metadata extraction required to get the file names.

bulk_extractor is a high-performance stream-based forensics tool.

Written in C, C++ and GNU flex

- Command-line tool.
- Linux, MacOS, Windows (compiled with mingw)

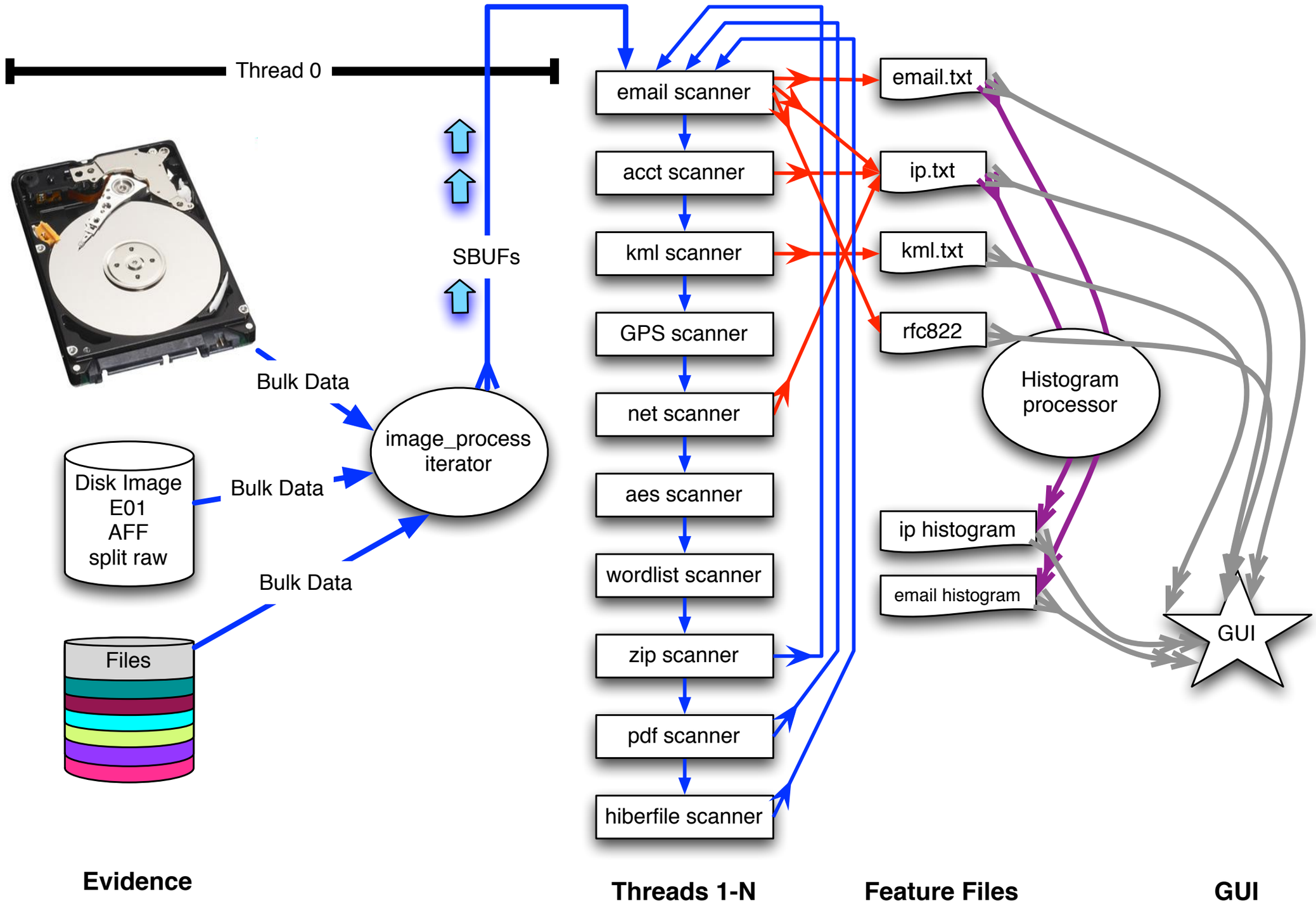
Key Features:

- “Scanners” look for information of interest in typical investigations.
- Recursively re-analyzes compressed data.
- Results stored in “feature files”
- Multi-threaded

Java GUI

- Runs command-line tool and views results

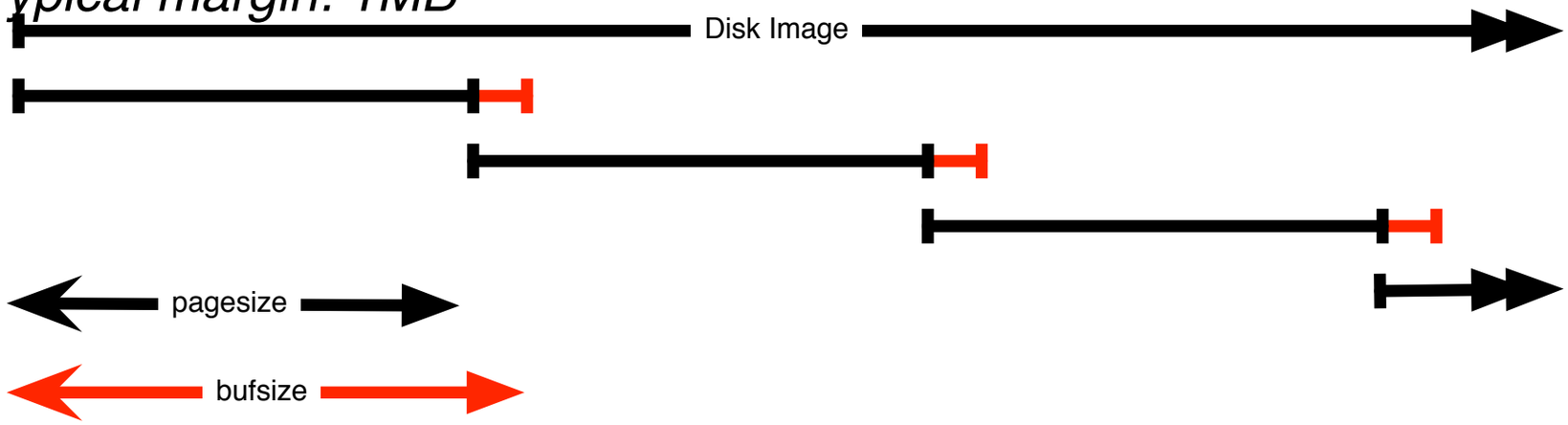
bulk_extractor: system diagram



The “pages” overlap to avoid dropping features that cross buffer boundaries.

The overlap area is called the *margin*.

- Each sbuf can be processed in parallel — they don’t depend on each other.
- Features start in the page but end in the margin are *reported*.
- Features that start in the margin are *ignored* (we get them later)
  - — *Assumes that the feature size is smaller than the margin size.*
  - — *Typical margin: 1MB*



Entire system is automatic:

- Image_process iterator makes **sbuf_t** buffers.
- Each buffer is processed by every scanner

Scanners process each page and extract features

scan_email is the email scanner.

inputs: **sbuf** objects

outputs:

email.txt

— *Email addresses*

rfc822.txt

— *Message-ID*

— *Date:*

— *Subject:*

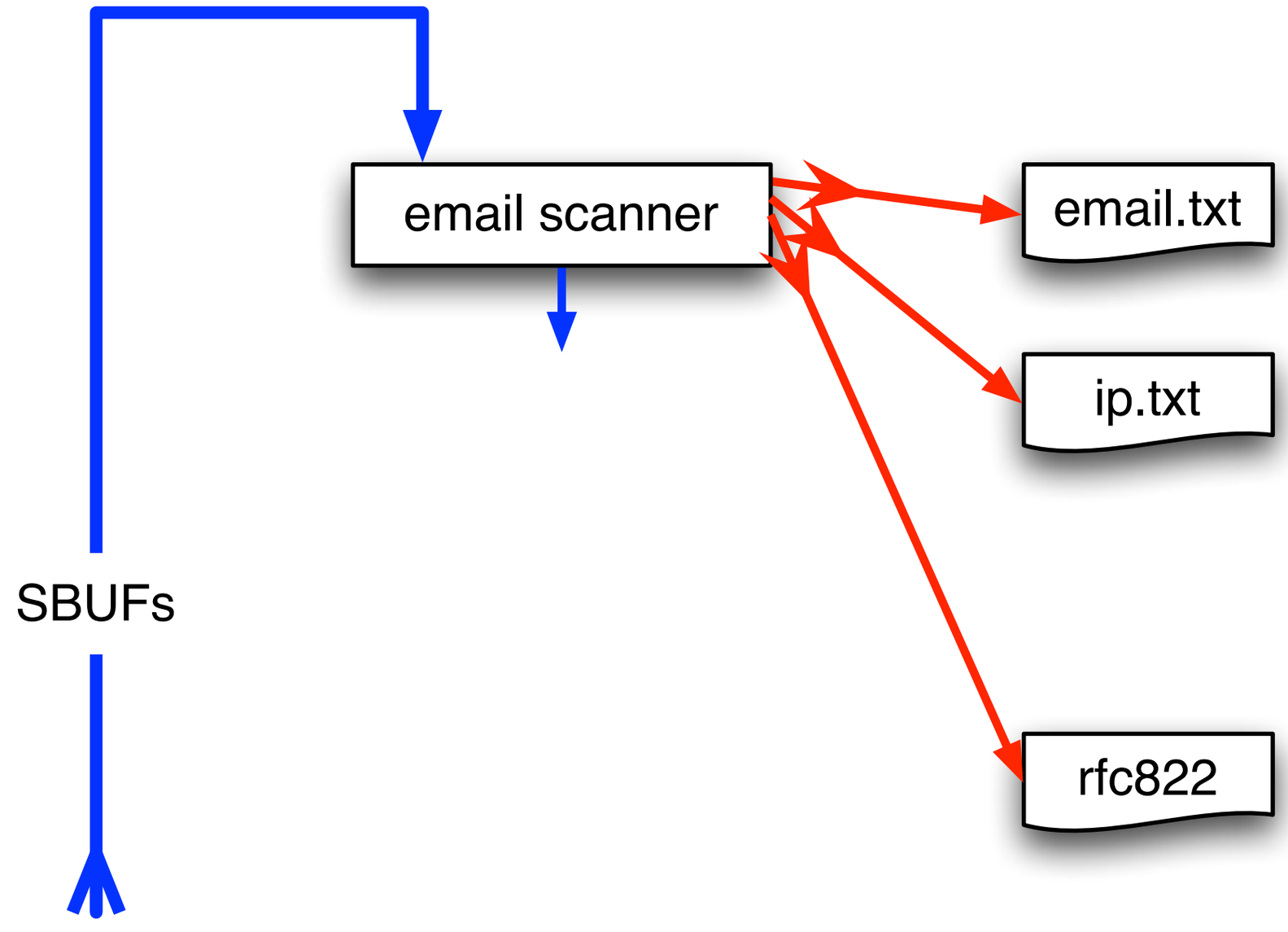
— *Cookie:*

— *Host:*

domain.txt

— *IP addresses*

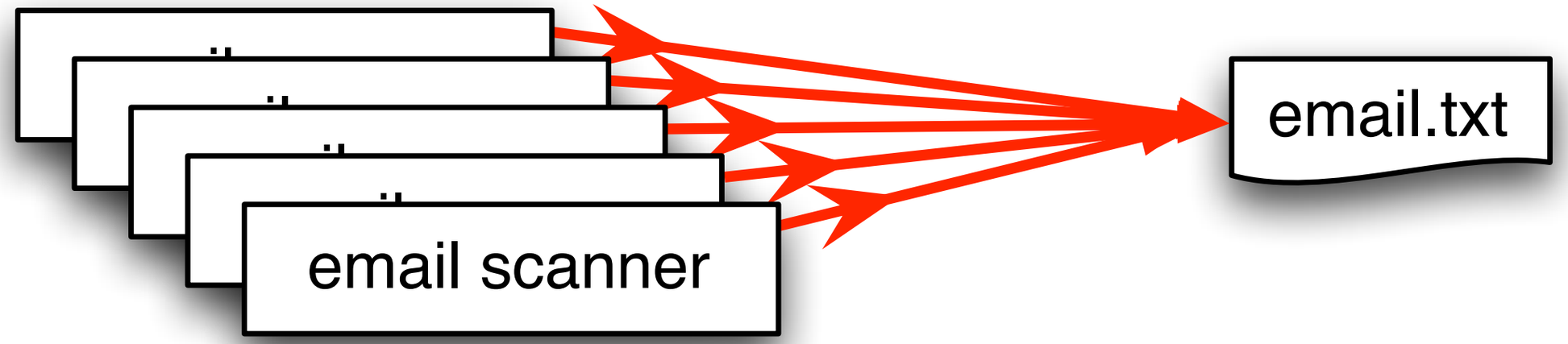
— *host names*



The *feature recording system* saves features to disk.

Feature Recorder objects store the features.

- Scanners are given a (feature_recorder *) pointer
- Feature recorders are *thread safe*.



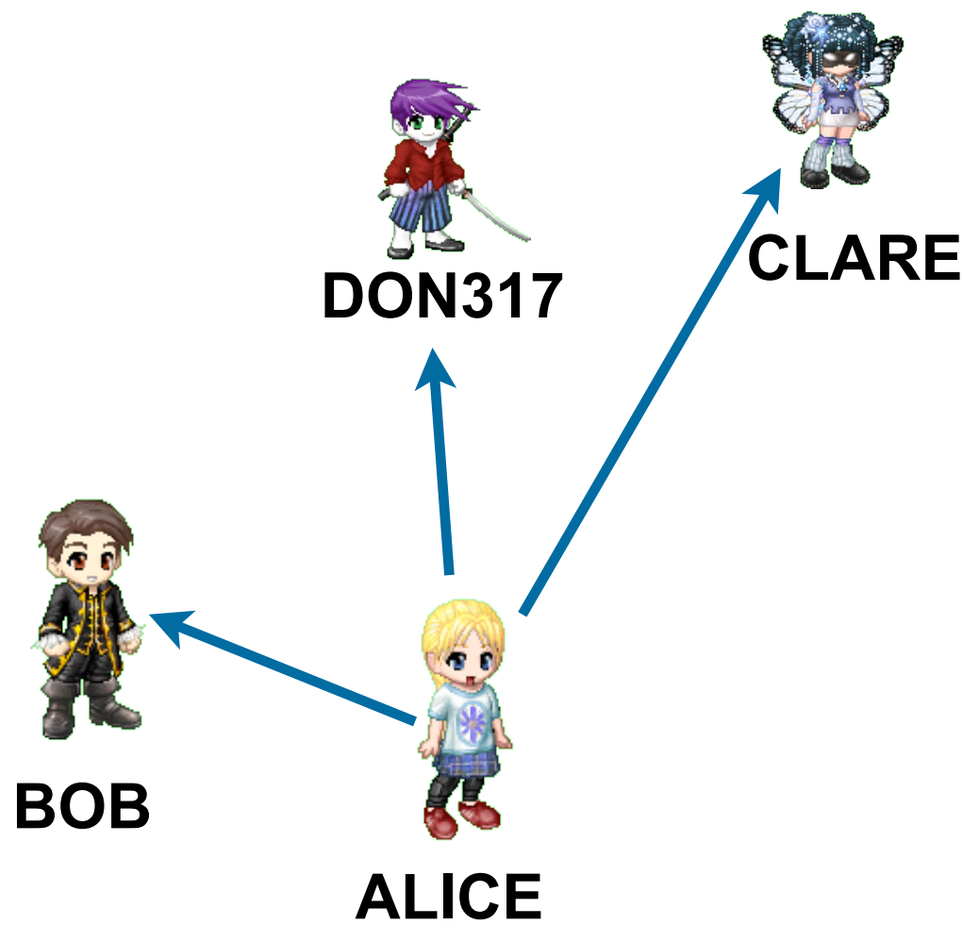
Features are stored in a *feature file:*

offset	feature	feature in evidence context
48198832	domexuser2@gmail.com	tocol>____<name>domexuser2@gmail.com/Home</name>____
48200361	domexuser2@live.com	tocol>____<name>domexuser2@live.com</name>____<pass
48413829	siege@preoccupied.net	siege) O'Brien <siege@preoccupied.net>_hp://meanwhi
48481542	danilo@gnome.org	Danilo __egan <danilo@gnome.org>_Language-Team:
48481589	gnom@prevod.org	: Serbian (sr) <gnom@prevod.org>_MIME-Version:
49421069	domexuser1@gmail.com	server2.name", "domexuser1@gmail.com");__user_pref("
49421279	domexuser1@gmail.com	er2.userName", "domexuser1@gmail.com");__user_pref("
49421608	domexuser1@gmail.com	tp1.username", "domexuser1@gmail.com");__user_pref("

Histograms are a powerful tool for understanding evidence.

Email histogram allows us to rapidly determine:

- Drive's primary user
- User's organization
- Primary correspondents
- Other email addresses



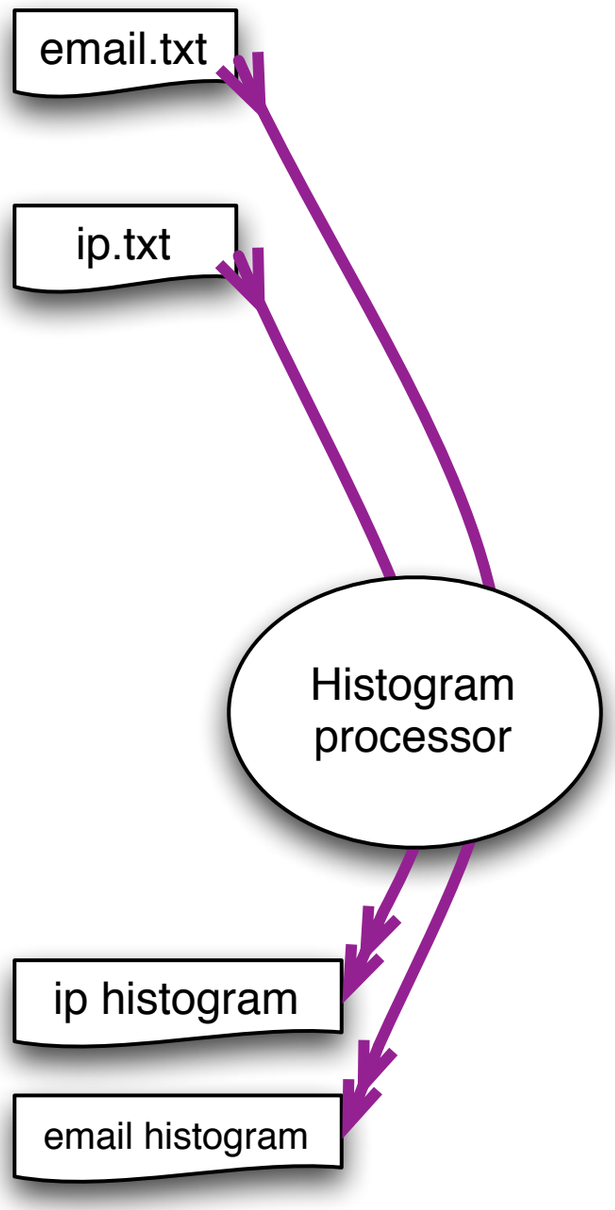
Drive #51 (Anonymized)

ALICE@DOMAIN1.com	8133
BOB@DOMAIN1.com	3504
ALICE@mail.adhost.com	2956
JobInfo@alumni-gsb.stanford.edu	2108
CLARE@aol.com	1579
DON317@earthlink.net	1206
ERIC@DOMAIN1.com	1118
GABBY10@aol.com	1030
HAROLD@HAROLD.com	989
ISHMAEL@JACK.wolfe.net	960
KIM@prodigy.net	947
ISHMAEL-list@rcia.com	845
JACK@nwlink.com	802
LEN@wolfenet.com	790
natcom-list@rcia.com	763

The feature recording system *automatically* makes historgrams.

Simple histogram based on feature:

```
n=579    domexuser1@gmail.com
n=432    domexuser2@gmail.com
n=340    domexuser3@gmail.com
n=268    ips@mail.ips.es
n=252    premium-server@thawte.com
n=244    CPS-requests@verisign.com
n=242    someone@example.com
```



Based on regular expression extraction:

For example, extract search terms with **.*search.*q=(.*)**

```
n=18     pidgin
n=10     hotmail+thunderbird
n=3      Grey+Gardens+cousins
n=3      dvd
n=2      %TERMS%
n=2      cache:
n=2      p
n=2      pi
n=2      pid
n=1      Abolish+income+tax
n=1      Brad+and+Angelina+nanny+help
n=1      Build+Windmill
n=1      Carol+Alt
```

bulk_extractor has *multiple* feature extractors. Each scanner runs in order. (Order doesn't matter.)

Scanners can be turned on or off

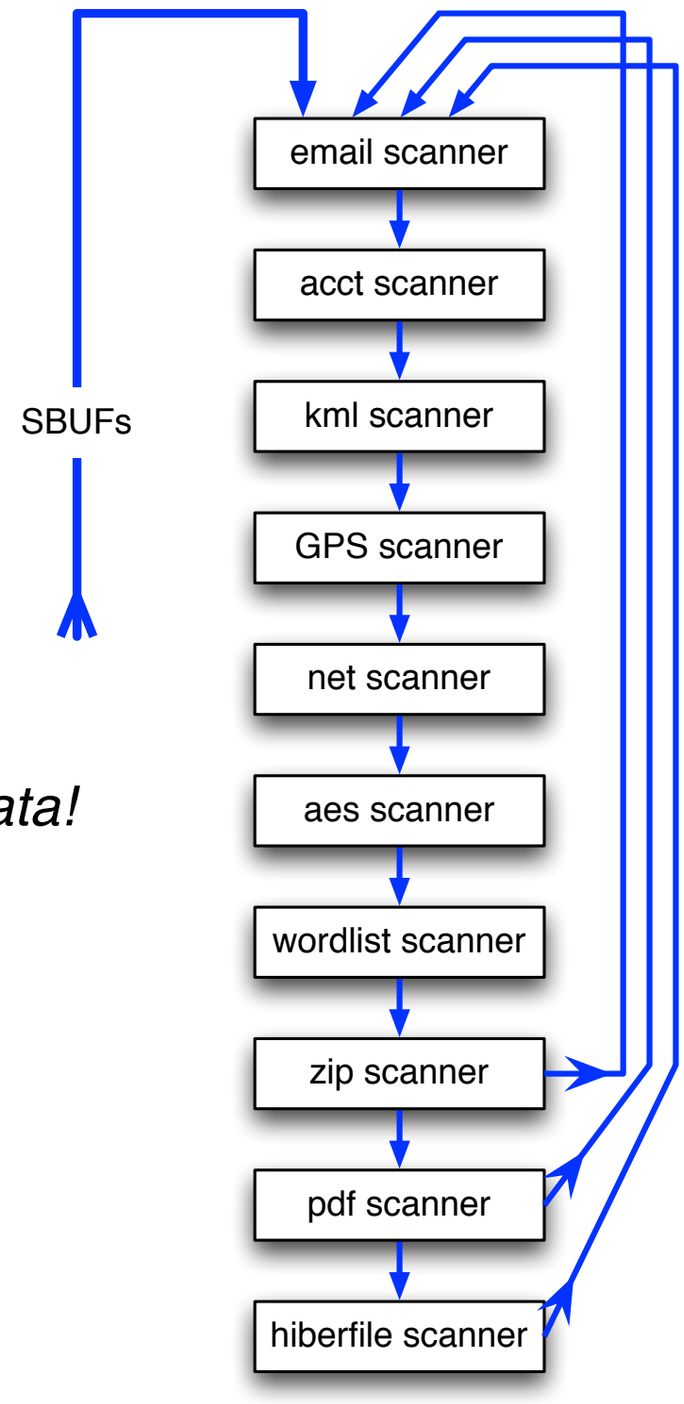
- Useful for debugging.
- AES key scanner is *very slow* (off by default)

Some scanners are *recursive.*

- *e.g.* scan_zip will find zlib-compressed regions
- An **sbuf** is made for the decompressed data
- The data is re-analyzed by the other scanners
  - — *This finds email addresses in compressed data!*

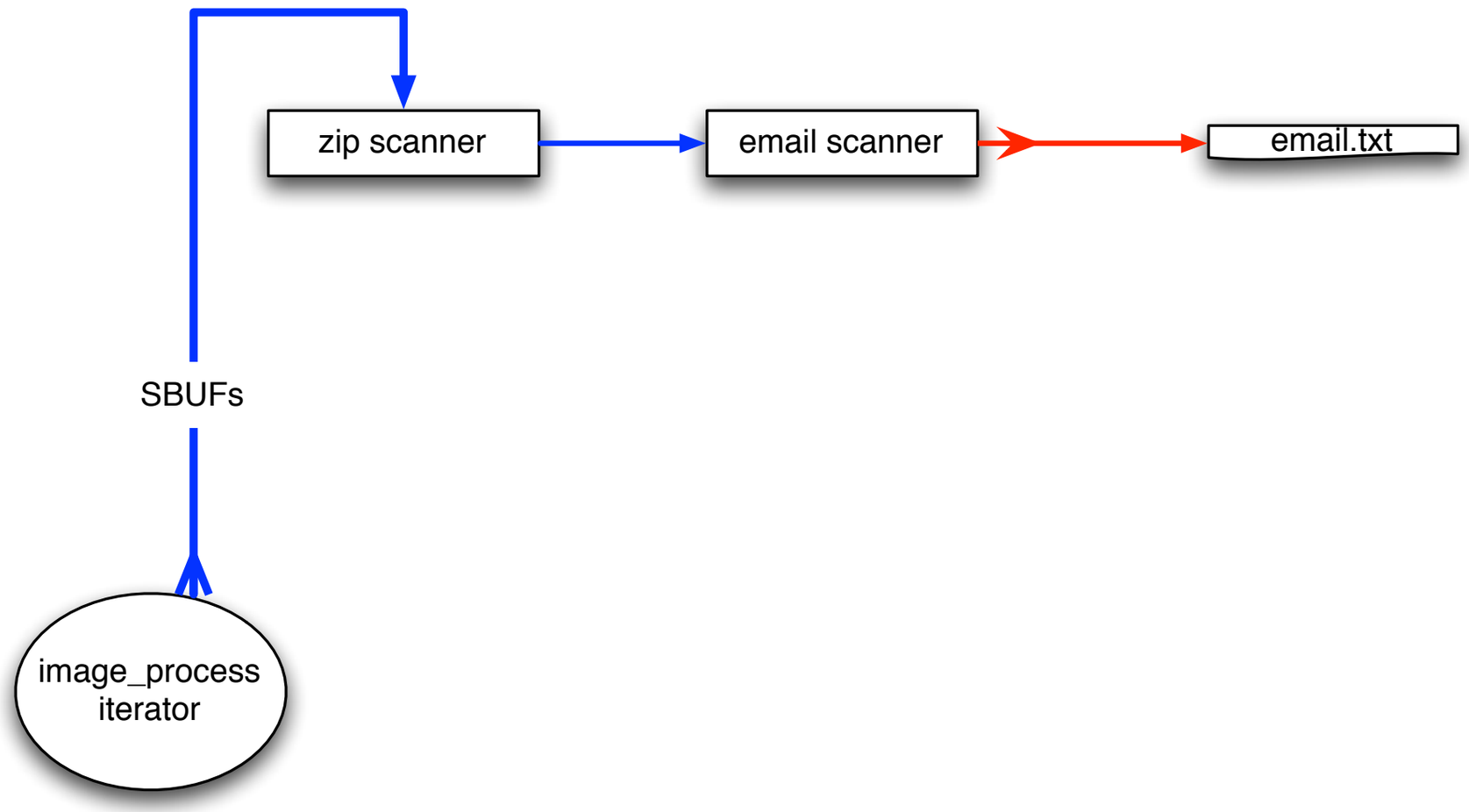
Recursion used for:

- Decompressing ZLIB, Windows HIBERFILE,
- Extracting text from PDFs
- Handling compressed browser cache data



Recursion requires a *new way* to describe offsets. bulk_extractor introduces the "forensic path."

Consider an HTTP stream that contains a GZIP-compressed email:



We can represent this as:

```
11052168704-GZIP-3437   live.com    eMn='domexuser1@live.com';var srf_sDispM
11052168704-GZIP-3475   live.com    pMn='domexuser1@live.com';var srf_sPreCk
11052168704-GZIP-3512   live.com    eCk='domexuser1@live.com';var srf_sFT='<
```

For this project, we made two changes to bulk_extractor

scan_hiberfile — Hibernation File Scanner

Previous hibernation file tools required *intact* hibernation files.

Our version can decompress

scan_net — Network Scanner

Implemented the algorithms in test, then final.

Key advantages of using bulk_extractor for research:

multithreading code — much faster to test

histogram system — makes analysis faster

production-ready release — once the code works, it's ready to be distributed.

bulk_extractor's plug-in architecture minimizes interactions between

Why focus on Hibernation — because it's a copy of RAM!

Hibernation files have *all memory*

- Network data structures in system memory
- Persist to hibernation
- Windows overwrites beginning of hibernation files when resuming
- Prevents existing systems from analyzing hibernation
- We find an 8-byte XPress compression signature within compressed memory page header.

Swap files have swapp

- Not pages wired down.

Opportunistically decompress XPress pages

Address	Count	Decompressed Count
172.20.105.74	25	600
172.20.104.199	41	434
18.26.0.230	43	162
172.20.20.11	0	4
. . .	. . .	. . .

- Improves recall by an order of magnitude on our test image!

Minimizing false positives

Minimizing false-positives

Checksum: Self-validate using IP checksum. Not always feasible due to checksum offloading. 82% of IPs in ground-truth have valid checksums.

Filtering: Eliminate bogus IP addresses not appearing in the BGP routing table, e.g. 127.0.0.0/8 and 240.0.0.0/4.

Frequency: Compute histograms of discovered IPs to determine most likely addresses.

Correlation: We examine if discovered binary IPs correspond to e.g. ASCII addresses

Determining client vs. server:

When TTL=2^n, we assume packet was captured *before* it was sent on the wire.

Validation

In comparisons with State-of-the-Art, we find dramatically more packets.

Fresh Windows XP install Large transfer, then hibernation

- We find the true source and destination IPs with high confidence as most frequent
- Volatility connscan2 finds nothing

NIST CFReDS memory images, labeled with ground-truth

- We discover IP of connection to w3.org
- Volatility connscan2 finds nothing

Analysis of Real Data Corpus is the real excitement. (1,817 images, including cameras, MP3 players, etc.)

We found IP packets and data structures on 40% of the images

Binary carving permits checksum validation == High Confidence!

We don't know how many are real...

- So we correlated against ASCII IP addresses.
  - *—20% of the images had correlation.*
  - *—On 66 drives, we found validated IPs that were NOT in the ASCII*

IP addresses were found throughout the drive

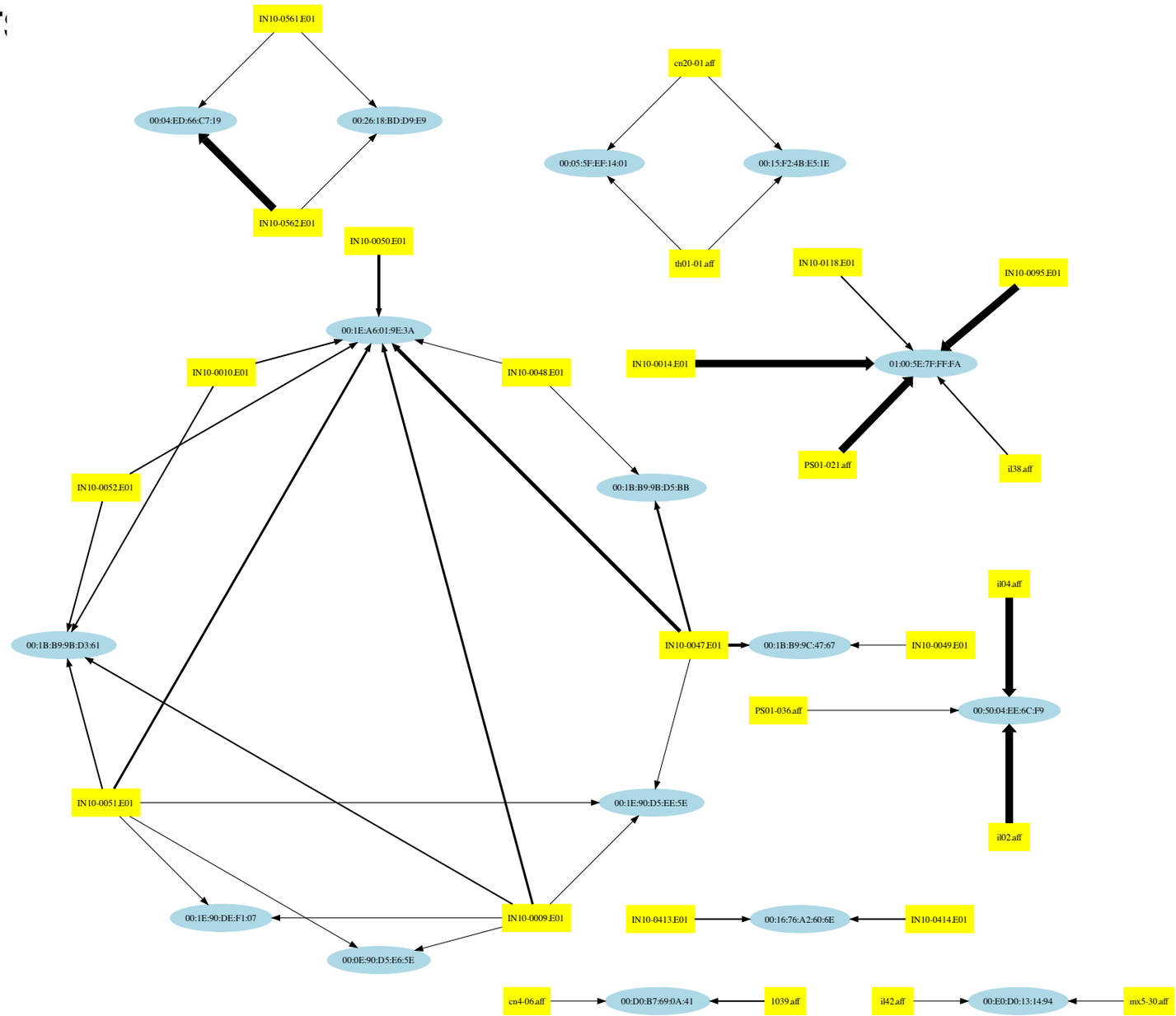
- 10% in **hiberfile.sys**
- 2% in **WIN386.SWP**
- 58% in unallocated regions of the disk!!!
  - ***—hiberfile.sys** and **WIN386.SWP** move on the disk with defragmentation.*

Cross-Drive MAC Analysis

Many RDC images were bought in batches.

We found 16 Ethernet addresses common between images.

8 distinct cluster



PCAP Carving — Give the customer what they need.

Version 1.0 bulk_extractor — Ethernet detection and carving

We thought this was super-cool.

Most users were not able to make use of the data.

Version 1.1 bulk_extractor — PCAP carving.

New scan_net detects PCAP packet header (time, capture size, etc)

```
struct pcap_pkthdr {
        struct timeval ts;      /* time stamp */
        bpf_u_int32 caplen;     /* length of portion present */
        bpf_u_int32 len;        /* length this packet (off wire) */
};
```

Now we create PCAP files!

— *PCAP files can be analyzed with existing tools.*

— *We also capture timestamps when carving PCAP files*

PCAP files can be analyzed with existing tools

On Hamming, we have a lot of files online:

```
$ !ls
ls -l */*pcap
-rw------- 1 slgarfin slgarfin  392 Feb  2 16:41 AE1001-1007/packets.pcap
-rw------- 1 slgarfin slgarfin 3084 Feb  2 16:57 AE1001-1009/packets.pcap
-rw------- 1 slgarfin slgarfin    0 Feb  2 15:07 AE10-1002/packets.pcap
-rw------- 1 slgarfin slgarfin    0 Feb  2 15:41 AE10-1011/packets.pcap
-rw------- 1 slgarfin slgarfin 2808 Feb  2 16:54 nps-2009-domexusers/
packets.pcap
$
```

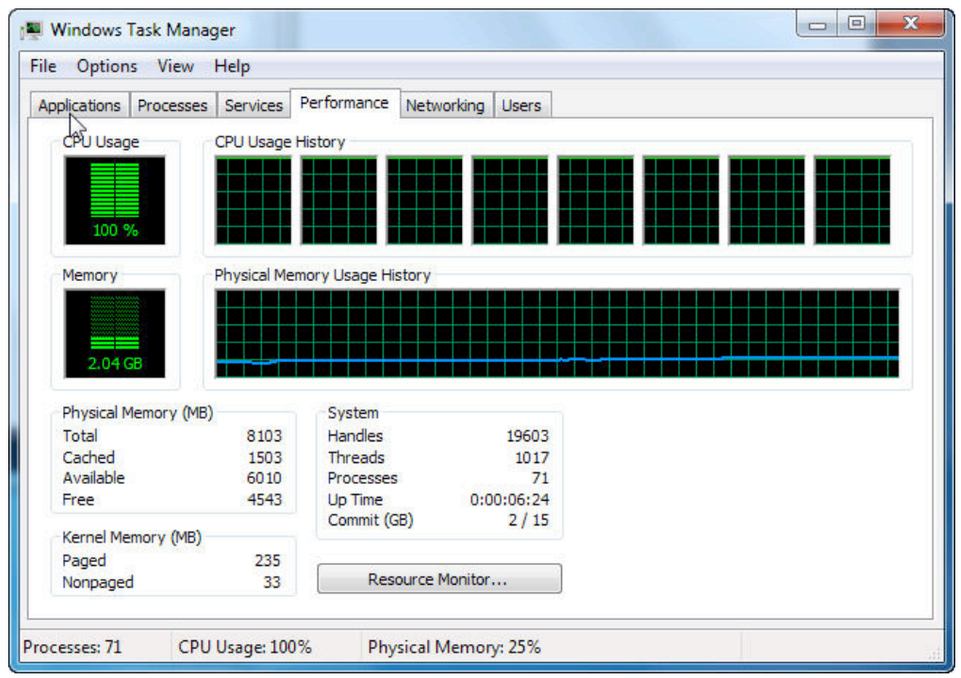
```
$ /usr/sbin/tcpdump -r nps-2009-domexusers/packets.pcap
reading from file nps-2009-domexusers/packets.pcap, link-type EN10MB
(Ethernet)
-7:-59:-59.000000 IP crl.verisign.net.http > 192.168.2.129.activesync: P
3875286422:3875287854(1432) ack 2739939955 win 64240
-7:-59:-59.000000 IP crl.verisign.net.http > 192.168.2.129.activesync: P
1432:2644(1212) ack 1 win 64240
$
```

Multiple packets can be combined into flows with tcpflow

In conclusion, bulk_extractor gives a new way to do network forensics.

Scan every sector of the disk for packets.

- Ignore the file system
- Read data out of hibernation and swap files.
- Find PCAP files that were fragmented or deleted.



Big advantage:

- Lots more hidden data!
- multi-threading makes program fast.
- Use tcpflow to reassemble TCP streams

Disadvantages:

- Packets outside of PCAP files have no date/time information
- Packets can be from distant past.

Questions?