Forensic Carving of Network Packets and Associated Data Structures

Robert Beverly, Simson Garfinkel, Greg Cardwell

Naval Postgraduate School
`{rbeverly,slgarfin,gscardwe}@nps.edu`

August 2, 2011

DFRWS Conference 2011

Outline

1 Overview

2 Background

3 Methodology

4 Results

5 Conclusions

Networks and Forensics

Forensic Value of Network Information:

- 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.

Networks and Forensics

Our Approach:

- Not looking at network traffic on the wire
- Not looking at logs (IDS/Firewall/Anomaly detector, etc)
- Instead – a storage-centric view

Post-facto residual network data

Are *low-level binary* network data structures persisted to non-volatile storage?

Networks and Forensics

Our Approach:

- Not looking at network traffic on the wire
- Not looking at logs (IDS/Firewall/Anomaly detector, etc)
- Instead – a storage-centric view

Post-facto residual network data

Are *low-level binary* network data structures persisted to non-volatile storage?

Network Carving

In this work, we ask:

Are *low-level binary* network data structures persisted to non-volatile storage?

Network Carving

In this work, we ask:

Are *low-level binary* network data structures persisted to non-volatile storage?

e.g.:

```
struct ip {
  u_int   ip_v:4,                    /* version */
          ip_hl:4;                   /* header length */
  u_char  ip_tos;                    /* type of service */
  u_short ip_len;                    /* total length */
  u_short ip_id;                     /* identification */
  u_short ip_off;                    /* fragment offset field */
  u_char  ip_ttl;                    /* time to live */
  u_char  ip_p;                      /* protocol */
  u_short ip_sum;                    /* checksum */
  struct  in_addr ip_src,ip_dst;     /* source and dest address */
}
```

Network Carving

In this work, we ask:

Are *low-level binary* network data structures persisted to non-volatile storage?

e.g.:

```
struct ip {
  u_int   ip_v:4,                 /* version */
          ip_hl:4;                /* header length */
  u_char  ip_tos;                 /* type of service */
  u_short ip_len;                 /* total length */
  u_short ip_id;                  /
  u_short ip_off;                 /
  u_char  ip_ttl;                 /*
  u_char  ip_p;                   /* protocol */
  u_short ip_sum;                 /* checksum */
  struct  in_addr ip_src,ip_dst;  /* source and dest address */
}
```

Surprisingly, yes!

Outline

1 Overview

2 Background

3 Methodology

4 Results

5 Conclusions

Prior Work

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

NPS Research

Our Contributions

- Using ground-truth corpus, develop methodology for carving binary network data:
  - Windows `_TCPT_OBJECT`
  - IP Packets
  - Ethernet Frames
  - Socket Structures
- Opportunistic hibernation decompression, including fragments
- Filtering and Validation techniques
- Working implementation in the `bulk_extractor` (`http://afflib.org/`) tool
- Evaluation on ground-truth and large (1800 drive) corpus

Outline

1. Overview
2. Background
3. Methodology
4. Results
5. Conclusions

Ground Truth

Ground-Truth Corpus:

- In order to find binary network carving structure signatures, we carefully create a ground-truth corpus
- Experimented with: Windows, OSX, Linux
- Wipe drive with DBAN to ensure no residual data
- From a virgin OS install, we establish several HTTP and SCP connections to *known* destination IPs
- Image the host’s disk after each connection

Finding Signatures

Finding Signatures:

- A binary IPv4 address is simply an unsigned 32-bit integer
- To find network addresses, we find discriminatory surrounding context
- Determine if there exist common predecessor/successor patterns surrounding instances of the known IP

Frequency Analysis

Finding Signatures

- Tempting to use intuitive heuristics:
  - “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 inaccurate

Instead:

- Search for IP address
- Collect (within 20 Bytes offset) preceding and surrounding *N*-grams
- Where a “gram” is simply a byte

Frequency Analysis

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
...	...	...	...

Frequency Analysis

IPv4 2-Gram Analysis

Predecessor Freq		Successor Freq	
Count	2-gram	Count	2-gram
434	0x4000	4	
421	0x0800	4	
368	0xF202	4	
368	0x4006	3	
368	0x4508	3	
368	0x0017		
66	0x4500	1	
...	...	...	...

Decoding:

- `0x4000`: IP Flags=Don’t Fragment
- To our surprise, discovered Ethernet frame data!
- `0x0800`: Ethernet “type”=IP
- ...

Frequency Analysis

IPv4 2-Gram Analysis

Predecessor Freq		Successor Freq
Count	2-gram	C
434	0x4000	4
421	0x0800	4
368	0xF202	4
368	0x4006	3
368	0x4508	3
368	0x0017	
66	0x4500	
. . .	. . .	

Decoding:

- Manual inspection on *N*-Gram frequency leads to robust signatures
- `0x4508/0x4500:` IPv4, w/ & w/o ToS
- `0x4006:` IP TTL=64, Proto=TCP
- While TTL=64 is common here, doesn't generalize
- . . .

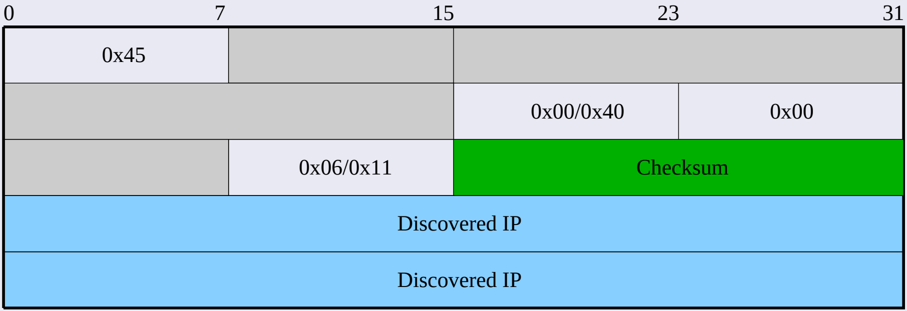
Carving Signatures

Signatures: Manual Inspection + *N*-Gram Analysis

Key

□	= Required	🟦	= Carved
▨	= Wildcard	🟩	= Validation

IP Carving



<table>
<tr><td>0</td><td>7</td><td>15</td><td>23</td><td>31</td></tr>
<tr><td colspan="1">0x45</td><td>▨</td><td colspan="2">▨</td></tr>
<tr><td colspan="2">▨</td><td>0x00/0x40</td><td>0x00</td></tr>
<tr><td>▨</td><td>0x06/0x11</td><td colspan="2">Checksum</td></tr>
<tr><td colspan="4">Discovered IP</td></tr>
<tr><td colspan="4">Discovered IP</td></tr>
</table>

Carving Signatures

Socket Carving

0 7 15 23 31

	0x02	16 common ports
Discovered IP		
0x00000000		
0x00000000		

Ethernet Carving

0 7 15 23 31 39 47

Discovered Ethernet Address		
Discovered Ethernet Address		
0x0800	0x45	

- Note: False positives possible, particularly with long strings of zeros; see paper for theoretical false positive analysis

Hibernation Decompression

Why Focus on Hibernation

- 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

Hibernation Decompression

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!

Validation

To Mitigate 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

Outline

1 Overview

2 Background

3 Methodology

4 Results

5 Conclusions

Comparisons to State-of-the-Art

Given our carving signatures and methodology:

- Compare to Volatility
- Analyze $\sim$ 1,800 images in Real Data Corpus

Comparisons to State-of-the-Art

Comparison to Volatility

- 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

Against Real Data Corpus

Real Data Corpus

- RDC: 1,817 images (including cameras, computers, mp3 players, etc)
- Discover IP addresses on 40% of images
- Note, binary carving permits checksum validation == high-confidence IPs!

How many addresses are “real?”

- We don’t have ground-truth
- Perform ASCII-based IP carving, correlate
- Good correlation between carving modalities for $\sim$ 20% of the images
- On 66 drives, we find validated IPs *not* found in ASCII form
- See paper for full analysis

RDC IP addresses

In RDC, where are IP addresses found?

- 10% in `hiberfil.sys`
- 2% in `WIN386.SWP`
- 58% in unallocated regions of disk!
- Suggests that valuable information in ephemeral stores needs to be carved by examining physical disk

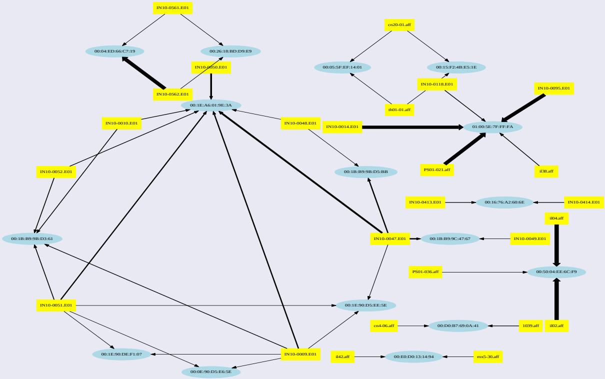
Geolocation

- Lots of private (RFC1918) addresses
- Limited success; see paper

Cross-Drive MAC Analysis

Cross-Drive MAC Analysis

- Many RDC images bought in batches
- We find 16 Ethernet common between images!
- Graph shows 8 distinct clusters:



Outline

1. Overview
2. Background
3. Methodology
4. Results
5. Conclusions

Future Work

Future Work:

- Examine other network structs: IPv6, 802.11, 802.15, 802.16, etc.
- Examine available application layer information
- Currently applying techniques to mobile smartphone images

Summary

- Demonstrated forensic value of *binary* network structures via controlled and real-world experiments
- Demonstrated importance of physical device scanning, including opportunistic hibernation decompression

Thanks!

Questions?