DIGITAL INVESTIGATION 8 (2011) S78–S89

available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/diin

Digital Investigation

Forensic carving of network packets and associated data structures

Robert Beverly, Simson Garfinkel, Greg Cardwell*

Naval Postgraduate School, Monterey, California, United States

Keywords:
Carving
Network Carving
bulk_extractor
Network analysis
Cross drive analysis

ABSTRACT

Using validated carving techniques, we show that popular operating systems (*e.g.* Windows, Linux, and OSX) frequently have residual IP packets, Ethernet frames, and associated data structures present in system memory from long-terminated network traffic. Such information is useful for many forensic purposes including establishment of prior connection activity and services used; identification of other systems present on the system's LAN or WLAN; geolocation of the host computer system; and cross-drive analysis. We show that network structures can also be recovered from memory that is persisted onto a mass storage medium during the course of system swapping or hibernation. We present our network carving techniques, algorithms and tools, and validate these against both purpose-built memory images and a readily available forensic corpora. These techniques are valuable to both forensics tasks, particularly in analyzing mobile devices, and to cyber-security objectives such as malware analysis.

Published by Elsevier Ltd.

1. Introduction

The value of many devices, whether personal computers, laptops, electronic book readers, or tablets, lies in their ability to attach to networks and communicate with other devices and services via the Internet. Information about the networks to which devices have connected and the communication patterns of those devices similarly provides forensic investigators a wealth of information.

For example, the set of source Internet Protocol (IP) addresses a device has acquired over time can provide clues to its physical network attachment points and mobility patterns. The destination addresses of IP packets can provide a picture of commonly visited web sites, parties transferring files, and the exchange of email. As a result, it is common forensic practice to scavenge a subject's device for network information and present that information to analysts and investigators.

Ethernet Media Access Control (MAC) addresses can also be of significant forensic value. The first three octets of an Ethernet MAC are assigned to specific equipment vendors and can be used to infer information about the equipment using a particular MAC address. But the MAC addresses of other machines on the subnet of the subject's device can also be revelatory. For example, MAC addresses of wireless routers can be correlated with war-driving databases and used to determine the physical location of a device. The mere fact that a computer has been associated with a physical network (either wired or wireless) can also be used to infer membership in an organization or access to a physical location.

Existing forensic tools examine media, *e.g.* a captured hard drive or memory stick, for email addresses, web site addresses, and domain names. Typically these tools work by extracting printable strings using regular expressions from web browser caches, email messages, and the like. In contrast,

* *Corresponding author.*
E-mail addresses: rbeverly@nps.edu (R. Beverly), slgarfin@nps.edu (S. Garfinkel).
1742-2876/$ – see front matter Published by Elsevier Ltd.
doi:10.1016/j.diin.2011.05.010

our work relies on lower-layer binary network signatures such as packet and socket data structures. These structures are created in memory and maintained by the operating system during the normal course of network activity. However, network structures are not confined to memory images as they are invariably written to fixed storage when a computer swaps or enters system hibernation. Additionally, network programs (*e.g.* domain name clients and servers, DHCP clients and servers, *etc.*), may use binary network data structures when storing configurations or caching results to the file system, making such files useful sources of data.

To validate the potential utility of carving for binary network data structures, we add network carving support to the *bulk_extractor* toolkit (Garfinkel, in submission May 2011) and verify our results against both disk images with known provenance and a large research forensic corpus. On ground truth images using checksum validation, we achieve perfect recall (all IP addresses found), and perfect precision (no spurious IP addresses identified).

Additionally, we find a surprising amount of discoverable residual network information in a public corpus of approximately 1800 images. For these, where ground truth is unknown, we show that there is a correlation between discovered network structures and their ASCII representation elsewhere. This dual-modality analysis adds confidence that our approach is broadly applicable and produces correct results.

This paper starts with a brief review of prior work in Section 2. Next we present our validated carving technique for IP packets and network data structures (Section 3). Section 4 presents experimental results of applying these techniques to a variety of forensic datasets. We conclude with a discussion of applications and opportunities for future work in Section 5.

Our work is particularly valuable to forensic analysts examining mobile devices such as laptops. Not only do laptops frequently hibernate during their normal course of operation, their natural mobility makes establishing connection activity and network attachment point location important. "Smart" mobile phone handsets may similarly be amenable to network carving; this is a subject of future research. Finally, network carving may be useful in other domains such as malware analysis.

2. Prior work

In this work, we broaden the scope of network forensics to include the discovery of long-terminated network data in memory and on fixed media. Whereas prior work and existing tools are largely limited to the discovery of so-called "fully-qualified domain names", (*e.g.* www.cnn.com) and "dotted-quad" IP addresses, (*e.g.* 157.166.224.26), we find significant additional information present in the form of intact packets and binary network data structures.

The Volatility memory analysis framework (Walters & Petroni, February 2007) most closely parallels our effort. Volatility can analyze the memory of live systems, memory dumps, and entact Windows hibernation files. The framework finds, traces, and prints a variety of memory structures from various operating systems. Most relevant to our work is Volatility's connscan2 function, which performs limited carving of Windows memory structures associated with TCP connections. In addition to these Windows-specific connections, our network carver extracts and validates socket, packet, and Ethernet frame data structures. As a result, we achieve a significantly higher recall against ground truth data than does Volatility.

Beyond Volatility, we find just one other reference to "network carving" in the literature—the syllabus of a SANS course (SANS, 2010) that discusses carving of web pages out of the caches of web proxies, and carving files out of network packet streams. We found this use of the word "carving" odd, since tools such as tcpflow (Elson and Garfinkel, 2011) break individual streams into their own files, performing much the same function.

Several groups have explored searching through kernel memory for code or data signatures. Dolan-Gavitt et al. (2009) describe an approach for identifying kernel-mode rootkits by searching for code signatures by are automatically generated. Lin et al. (February 2011) present an approach finding malware by finding inconsistencies in signature graphs generated for kernel data structures. Yet a third technique has been developed by Liang et al. (March 2011). However, we have found no prior work carving IP packets or Ethernet frames out of kernel memory or memory dumps. Originally we thought that such carving might not be possible, as we assumed that portions of the Ethernet and/or IP protocol would be handled in hardware for performance reasons and not exposed to the operating system. Instead, as we shall show, we find significant residual binary network structure data.

Because hibernation files are an invaluable source of network structures, it is important that we handle them correctly. Unlike MacOS and Linux, Microsoft Windows hibernation files are partially compressed using the Microsoft XPress Block Memory Compression algorithm (Suiche, 2008). Volatility implements hibernation file decompression but only for intact hibernation files. Unfortunately, Windows corrupts the beginning of a hibernation file when the system reboots so that it cannot be used again. We extend Suiche's work by implementing an improved scanner that can detect signatures associated with the start of intact Windows hibernation pages (which may be present in a corrupt hibernation file or located elsewhere on the hard drive as residual data from previous hibernation files) and optimistically decompresses them until an error occurs. As a result, our scanner is able to process many more hibernation files than Volatility currently can.

Garfinkel performed multi-drive correlation ("cross-drive analysis") using "pseudo-unique information" (actually persistent identifiers) such as social security numbers, credit card numbers and email addresses (Garfinkel, August 2006). Here we extend that work by performing multi-drive correlation using Ethernet MAC addresses extracted from validated IP packets. Garfinkel (August 2007) also proposed improving file carving by validating internal document binary structures. We extend that work by validating structures not just for internal consistency, but for consistency with external databases (in this case, the Internet's schema for assigning IPv4 addresses).

Finally, our work in geolocation was assisted by geolocation databases created by companies such as Google (Google Mobile, 2011) and Skyhook (Skyhook, 2011).

Our hope in this work is to demonstrate the power by network carving and to document our techniques so that they may be incorporated into current media forensics tools.

3. Network carving

In this section we discuss the design and construction of our network carver; the methodology to detect network signatures, the network structures we extract; and the ground truth dataset construction for verifying our carver.

3.1. Creating ground truth data

We commenced this project by creating a ground truth dataset on which to develop our carving strategy. We started by securely erasing all data on a machine's hard drive and installing a virgin copy of the operating system. Each machine was then used to connect to multiple servers on our local network and on the Internet at large. We performed file transfers of four files of different sizes from three distinct systems as shown in Table 1. During this period, all of the packets entering and leaving the machine were captured using a promiscuous recorder. We then induced the machine to hibernate and imaged the disk. For the ground truth data, we experimented with a variety of operating systems including Microsoft Windows XP (service pack 3), Windows 7 professional, and Macintosh OSX 10.6.5.

3.2. Developing carving signatures

In order to develop our network carver, we require two items: the aforementioned dataset[1] and a technique to generalize discriminatory patterns. We note that a binary IPv4 address is nothing more than an unsigned 32-bit integer, which appears everywhere, therefore we must rely on surrounding contextual data for validation.

We started with a set of manually-derived heuristics, for instance: "a four-byte IP address is preceded by a variable fragment field and a protocol field equal to six (TCP)". Mirroring prior research in named entity extraction (Nadeau, 2007), we found that such heuristics are brittle, difficult to define, and often inaccurate.

To develop better heuristics we wrote an exploratory program that searched for the byte patterns of our known IP addresses and computed the frequency of the n-gram byte patterns preceding and following the addresses at various offsets within a specified window. We then crafted new carvers using these signatures. Finally we re-processed our hibernation files with these signatures, tabulating all of the IP addresses that were discovered.

For performance measures, we term a true positive (TP) as a discovered IP address from Table 1, a false positive (FP) as an address that was not from that table or the host computer, and a false negative (FN) as an address in the table that is not discovered. We use standard metrics of precision ($TP/(TP + FP)$) and recall ($TP/(TP + FP)$), and manually tuned our algorithms using the selection of n-grams.

[1] A product of this research is the creation of disk images with known ground truth that will be contributed to an existing forensic corpus (Garfinkel et al., August 2009).

Table 1 – Ground truth flows. Each flow was downloaded from the specified address using either HTTP or SCP.

File	Size	Host
Small	10 kB	18.26.0.230
Medium	100 kB	18.9.22.69
Large	1 MB	128.30.2.134
Huge	10 MB	18.26.1.76

As an illustrative example, we performed numerous network connections from Windows, Linux, and OSX workstations to the destination address 172.20.104.199, suspended the machine and captured its hibernation file. We then scanned memory for the hex sequence corresponding to this address (0xAC1468C7). Recall that the IP source and destination addresses occur 12 and 16 bytes into an IP packet (Postel, September 1981). Next we computed the frequency distribution of the two-byte grams within a fixed 20 byte window surrounding the address. Table 2 shows the most frequent 2-grams from this analysis.

We observe that the most frequent 2-gram preceding an IP address is 0x4000 which, upon manual inspection, appears in the IP flags field and indicates that the "don't fragment bit" is set. Next are 0x0800 and 0xF202, the trailing bytes of the Ethernet (Hornig, April 1984) link-layer encapsulation of the IP packet (0x0800 is the Ethernet type field indicating an IP packet while 0xF202 is the last two bytes of the source Ethernet address). As we set out to discover IP packets, finding Ethernet structures is one demonstration of the power of the frequency analysis approach. We use this discovery to carve Ethernet addresses in §3.3.

0x4508 and 0x4500 indicate the start of an IP version 4 packet with the type of service bits set or cleared (the *ssh* secure shell file transfer sessions have this bit set). The 0x4006 represents an IP TTL of 64 (0x40) and protocol TCP (0x06). The fact that 0xF202, 0x4006, and 0x4508 all have the same frequency is relevant and represent 368 Ethernet encapsulated TCP/IP packets with TTL of 64 and type of service bits set.

Similar analysis with different search addresses and different n-gram sizes reveal additional clues for choosing

Table 2 – IP 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
…	…	…	…

discriminating features. For example, the most frequent 8 byte successor gram that is common across different target addresses is all zeros. This detail supported our decision to scan for socket network address structures (struct sockaddr_in), as precisely eight sequential zero bytes is somewhat unusual in memory dumps.

Although it is possible to take the results of our n-gram study and use this to train a machine learning algorithm, such an approach would train the system on both the structure of our data (the IP packets) as well as the content (the actual data being sent over the network connection). We believe that instead our manual approach to feature selection results in a scanner that is more accurate, since it relies on expert data that is unavailable to the naïve machine learning algorithm.

3.3. *Methodology*

On the basis of the preceding signature analysis, we created a new module (scan_net) for bulk_extractor, an open source forensic tool. This scanner carves the following data structures:

- *IP Packets*: As depicted in Fig. 1, we identify a potential IP packet as having a first byte of 0x45 (IP version 4 and a standard header length), a flags and offset field of 0x0000 or 0x4000 and a protocol of either 0x06 or 0x11 (TCP or UDP). When possible, we verify the IP-level packet checksum.
- *Socket structures*: We find socket structures in the form shown in Fig. 2 and specified in netinet/in.h. An address family of 0x02 (Internet) and port precedes the remote address. Eight bytes of zero follow. Note that because long strings of zero bytes are common on disk, socket structures generate the most false positives. An analysis of the theoretical false positive rate is provided in §4.1.
- *Windows*: TCPT Microsoft Windows platforms maintain TCP-specific connection structures. We adopt the functionality in Volatility's connscan by scanning for the four-byte signature 0x54455054 ("TCPT") with the correct pool size, and then interpreting the following 28 bytes as a _TCPT_OBJECT.
- *Ethernet*: The most common link-layer technologies are 802.3 Ethernet and 802.11 wireless, both of which include globally unique six-byte hardware MAC addresses. Many of the IP packets we discover while carving are encapsulated in an Ethernet frame[2]. The best signature is 0x0800, Ethernet IP type (Hornig, April 1984), followed by the beginning of an IP packet (Fig. 3). Recovering Ethernet MAC addresses enables addition inferences, as there now exist databases of wireless router MAC addresses Such databases, (*e.g.* wigle.net; Skyhook) and other war-driving collections, can be used to accurately geolocate where the computer was previously used.

In practice, carving for all of these various binary network structures proves valuable. For instance, the previously available technique of discovering Windows TCPT structures accounts for only 1% of the discovered IP address instances on our selected ground truth data.

[2] We did not, however, find any instances of ARP packets.

Second, many hosts, particularly those found in residential installations, are given a private IP address (Rekhter et al., February 1996) and use some form of network address translation (NAT). The NAT maps the private local address to a globally routeable Internet address. Unfortunately, a private IP address effectively hides the network attachment point of the host. In these instances, the discovery of the source Ethernet MAC addresses can be vital—for example, by connecting a piece of seized media to a specific home router.

3.4. *Validating IP addresses*

As shown in §4.1, our signatures have a low, but non-negligible analytic false positive rate. In practice the false positive rate can be higher as the prior probability of any given byte is not uniform, and the probabilities of individual bytes are not independent. To cope with false positives, we implement several techniques for validating IP address hits.

- *Checksums*: IP packets provide a means of self-validation by including a checksum over the IP header (Braden et al., September 1988). Where possible, we validate that a correct checksum exists for all discovered IP packets. Unfortunately, we cannot exclude occurrences of invalid checksums as modern hardware performs checksum offloading (Chase et al., April 2001) as a performance optimization. In many cases, garbage or zero bytes are passed by the operating system to the physical network card which then computes the checksum in hardware for transmission on the wire. On our ground truth data, we find approximately 18% of discovered IP address instances have an invalid checksum while the remaining 82% validate. The addresses with invalid checksums are all false positives, suggesting that on our particular hardware, checksum bytes are passed from the hardware to the operating system.
- *Filtering*: A second means of validation is to identify bogus addresses. IP addresses are allocated in a structured, hierarchical manner (Hubbard et al., November 1996). While an IP address is nothing more than an unsigned 32-bit integer, many IP addresses are reserved (*e.g.* 127.0.0.0/8), invalid (*e.g.* 240.0.0.0/4), or simply unlikely (*e.g.* 224.0.0.0/4). Even more importantly, many addresses allocated to providers are not announced in the global BGP routing tables (Rekhter et al., January 2006). As such, we take the global routing table as aggregated from many vantage points in the Routeviews project (Meyer, 2010) and filter addresses which do not appear in the table.
- *Frequency analysis*: We compute a histogram of the IP addresses that are found in the carved data structures. IP addresses that appear more frequently are more likely to be the result of actual operating system activity and less likely to be the result of random matching. To this end, we configure *bulk_extractor* to create a histogram of recovered IP addresses, TCP connections, and Ethernet MAC addresses.
- *Correlation between modalities*: Lastly, correlation using different methods builds confidence in an inference. IP addresses that are found in both IP packets and socket structures are more likely to be IP addresses from prior traffic than IP addresses recovered via just one modality.

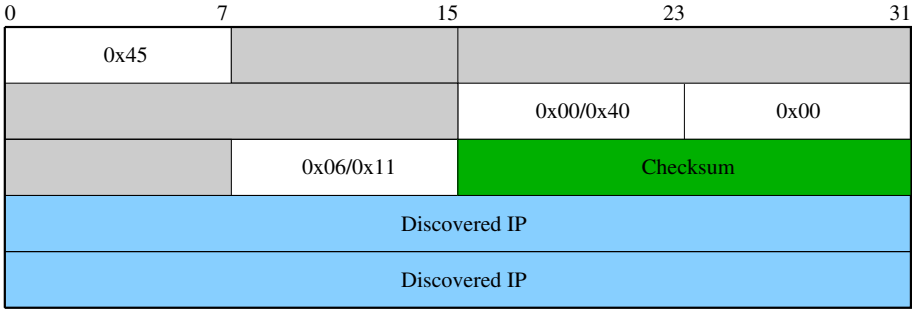

Fig. 1 – Carving IP packets: The white bytes provide a signature for a likely IP packet (p(false positive) $\simeq 2^{-30}$). Some hardware and operating systems preserve the IP checksum (in green), providing an improved means to validate the packet on some hardware configurations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

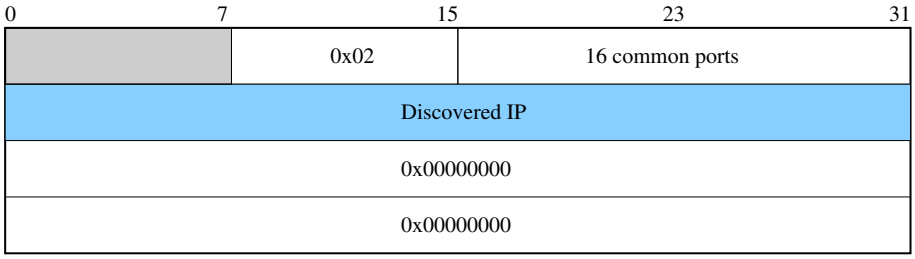

Fig. 2 – Carving sockets: The white bytes provide a signature for the struct sockaddr network socket structure. While the signature is large, long runs of zero bytes a common in memory dumps, forcing additional validation for IP addresses found using this method.

As indicated above, all of these techniques can be used to increase confidence in a specific IP or Ethernet address. When conducting triage operations, the analyst may find that it is useful to use confidence to prioritize the effort applied to each address. On the other hand, when performing an in-depth analysis, it may be more productive to investigate every confirmed and potential address that is recovered from the media.

3.5. *Local versus remote addresses*

The histogram of IP addresses from IP packets provides valuable insight into the previous connection patterns of the device in question, but does not differentiate between address(es) used by the subject device, *i.e.* the local source address, and those of remote systems.

As one step to identify the device's source address, we observe that IP packets can leak valuable information (Beverly, April 2004). We utilize the IP time-to-live (TTL) parameter of packets to determine if they were originated by the system, or received from a remote system. In particular, IP routers decrement the TTL field of each packet while operating systems set the initial TTL of outgoing IP packets in powers of two (the most typical values being 64, 128, and 255). Because packets originating from the local host will not have encountered any routers, and hence not had the TTL field decremented, we can expect their TTL to be a power of two.

We use this power-of-two trick to label IP addresses discovered via carving as local or remote. We can also use this trick to determine the distance, measured in router hops, of the packet source. To do so we use the well-known heuristic of selecting the next-highest power of two as the originating

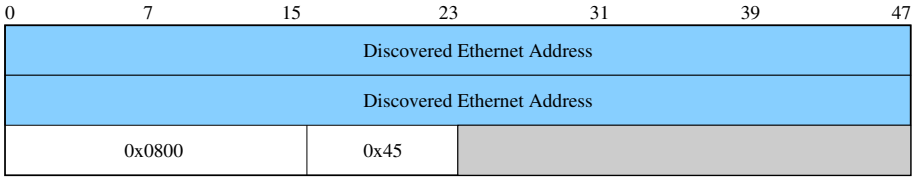

Fig. 3 – Carving Ethernet: The white bytes provide a signature for a likely Ethernet structure, including an Ethernet IP type (Hornig, April 1984) and first byte of IP. Ethernet addresses are especially valuable for forensics as they are unique and attributable.

TTL. If the observed TTL is greater than 128 we infer an original TTL of 255 and if less than 32 we infer 32.

Local and remote addresses can also be directly distinguished by the presence of addresses in struct sockaddr structures, if they are present.

3.6. Hibernation decompression

Because network data structures are typically created in system memory, we naturally find significant amounts of network information contained in hibernation files. Microsoft Windows hibernation files compress some of the hibernated memory pages, implying that decompression can yield better carving recall and performance. Unfortunately, Windows overwrites the beginning of hibernation files when it resumes from hibernation. As the beginning of the file contains memory map information critical for reassembly, overwriting makes existing hibernation systems (*e.g.* Walters and Petroni, February 2007; Suiche, 2008) fail. Furthermore, there may exist fragments of previously-used hibernation files present in the unallocated areas of a hard drive if the user erased the HIBERFIL.SYS and the file was later recreated by the operating system; existing systems cannot decompress such fragments.

Fortunately, the beginning of each page of the Windows memory map that is compressed using the XPress compression algorithm is preceded with a 32-byte header containing an 8-byte signature of 0x81 0x81 0x78 0x70 0x72 0x65 0x73 0x73 and three bytes (a, b, c) from which the length of the compressed buffer can be decoded using equation (1):

$$\text{length} = \frac{a \ll 8 + b \ll 16 + c \ll 24}{1024} + 1 \quad (1)$$

We implement a Windows hibernation scanner and decompression pipeline in the *bulk_extractor* framework using a modified version of Suiche's source code.

Decompression improves the IP address hit recall by an order of magnitude. Table 3 shows the frequency of IP addresses discovered with and without hibernation decompression enabled for one of our ground truth data systems. All of the addresses are known to be valid, as they represent known flows from our exploratory ground truth data image. Whereas in compressed form the most frequently carved IP address is 18.26.0.230, the systems' true source IP address (172.20.105.74) becomes the most commonly found IP address following decompression.

Table 3 – Windows hibernation file decompression: by implementing a per-fragment decompression algorithm, we significantly improve IP address carving recall.

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
…	…	…

4. Experimental results

In this section, we apply the aforementioned techniques to both our ground-truth data corpus and a large, public corpus in order to better understand the potential of our approach. First, however, we derive the expected false positive rate on random data for baseline comparison.

4.1. Theoretical false positive rate

If we define a hit in our carving as a match of a specified m fixed bytes from a window of N bytes, the probability of a false positive against a uniformly distributed random N bytes is 2^{-8m}. For example, if we hit on IP packets by finding a leading 0x45 (IP version 4 and header length 5), an offset of 0x0000 or 0x4000 (none or "don't fragment"), and a protocol of 0x06 or 0x11 (TCP or UDP), we find: $P(\text{IP false positive}) = 2^{-30}$. Thus, on random data, we will generate approximately one false positive per GB of data.

Because we match on eight bytes of zeros and a byte of family in struct sockaddr, we expect a 2^{-72} chance of generating a false hit with this second approach. In reality, disks frequently contain long strings of zeros, effectively reducing our rate to: $p(\text{sockaddr false positive}) \simeq 2^{-8} \times p(\text{trans})$, where $p(\text{trans})$ is the probability of a transition between a run of 8 zero bytes and a region of non-zero bytes.

Finally, the probability of generating a false positive using only IP checksums (Braden et al., September 1988) against randomly generated IP packets is relatively high: 2^{-16}. To see this, note that the IP checksum uses the 1's compliment of two-byte words in the IP header. Therefore, a random ensemble of nine two-byte words generates a randomly drawn two-byte word which must match the checksum field. Thus, there is a one in 2^{16} chance of a match at random (Stone et al., October 1998).

4.2. Comparison with volatility

Our first performance experiment concerns the ability to discover additional network information beyond that inferred by Volatility. We created a fresh install of Windows XP in our controlled environment. We then performed the large file transfer while using an external packet capture device. Once the transfer completed, we manually hibernated the machine and retrieved the hibernation file from the disk. Running Volatility against this image using "connscan2" found no addresses or connections, while our binary network carving technique discovered both the host's source IP address and the remote destination.

For additional external validation, we compared our tool against Volatility on the NIST CFReDS (NIST) memory images. These images are labeled with the destinations to which the browser had been pointed. Whereas Volatility does not find a network connection to the web site w3.org, our technique discovers a residual IP packet.

4.3. Against ground truth data

Next, we examine the performance of our scanner against the ground truth images we previously created. In all instances,

across all operating systems, we are always able to retrieve the machine's local IP address. Table 4 shows the precision of network carving using only high-confidence and all addresses. We label an IP address as a true positive when it is discovered and also appears in the captured packet trace. Similarly, a false positive is one that we discover, but does not appear in the packet trace. The precision of the validated, high-confidence addresses is quite high. Not only are the host's source IP address and the HTTP transfer destination addresses discovered, we are typically able to discover many of the other machines in the subnet that are sending broadcasts, for instance Windows file sharing.

4.4. *Against real data corpus*

Having validated our methodology, we apply our network carving to 1817 images in the Real Data Corpus (Garfinkel et al., August 2009). Note that this corpus contains images from not only general purpose computers, but also from digital cameras, music players, *etc.* Specifically, using operating system inference techniques, 335 images are a version of Windows. Of the 1817 images, we discover IP addresses on 723 (≃40%). Those images in which no IP addresses are found have been excluded from this analysis.

Unlike the controlled dataset, we have only limited ground truth over the images in the larger corpus. To ascertain how representative are these discovered IP addresses, we correlate them against discovered dotted-quads as represented in ASCII. We expect some degree of correlation as a host's address will often appear in configuration files, logs, caches, *etc.* that are found in the natural course of carving by *bulk_extractor*.

Fig. 4 shows a scatter plot for the drives. On the typical drive there are significantly more IP addresses that can be found from the ASCII than from network carving, and the number of addresses that correlate varies considerably per drive.

Notice that there are between one and two orders of magnitude more IP addresses from ASCII carving than by network carving. This is not just because network carving finds relatively few IP addresses—it is because state-of-the-art ASCII IP address carving techniques search for IP addresses using regular expressions, and thus generate a significant number of false positives from software version numbers, numbers in spreadsheets, and other sources. In contrast network carving, especially from IP packets with valid checksums, generates high-quality IP addresses with a negligible false-positive rate.

We find an average of 2258 IP addresses in ASCII and 21 IP addresses via binary network structure carving per drive (respectively a standard deviation of 19,635 and 52). On 66 of the drives, we find validated IP addresses via network carving that do not appear elsewhere in ASCII form.

Table 4 – Windows 7 ground truth performance.

File	Precision (High confidence)	Precision (All)
Small	0.84	0.55
Medium	1.00	0.25
Large	0.50	0.22
Huge	1.00	1.00

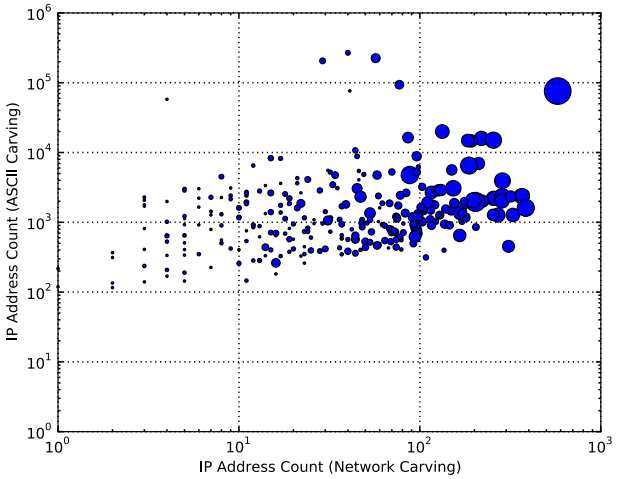

Fig. 4 – Correlation between scanning modalities across 723 images in the corpus. Each circle corresponds to a single hard drive, where the X axis indicates the number of addresses found through binary carving, the Y axis indicates the number of addresses found by ASCII carving, and the size of the circle indicates the number of addresses that are the same.

For a given image i, let the correlation factor c_i be the fraction of IP addresses found in network structures that are also found in ASCII form elsewhere in the image. Fig. 5 shows the cumulative distribution of the 723 images as a function of c. We analyze two cases, first the correlation factor for all discovered IPs and second the correlation for IPs with a frequency greater than one. From the Figure, we see that for approximately 50% of the images, none of the IP addresses discovered in network structures are found in ASCII.

However, around 20% of the images have a $c_i \geq 0.2$, indicating that for these images of only partially known provenance, there is correlation between our method and a completely different analysis.

Looking a bit more deeply, we determine the difference in correlation between IP addresses discovered within IP versus socket structures. Fig. 6 shows that ≃53% of images contain none of the socket addresses elsewhere. However, the correlation is much stronger for addresses discovered within IP structures where there is some correlation for 74% of the images. In fact, for IP structures, 50% of the images have $c_i \geq 0.5$, indicating a strong correlation.

Thus, there are high-confidence IP addresses (those we find through correlation between modalities) and lower-confidence IP addresses (those which are found using a single modality). Depending on the importance of a disk and a target, an analyst might go after only the high-confidence IP addresses or might go after all of them.

Finally, we examine the file source of IP addresses discovered in the corpus. For 10% of the 723 images where we discovered addresses, one of the IPs was found in the hiberfil.sys Windows hibernation file while 2% were in the

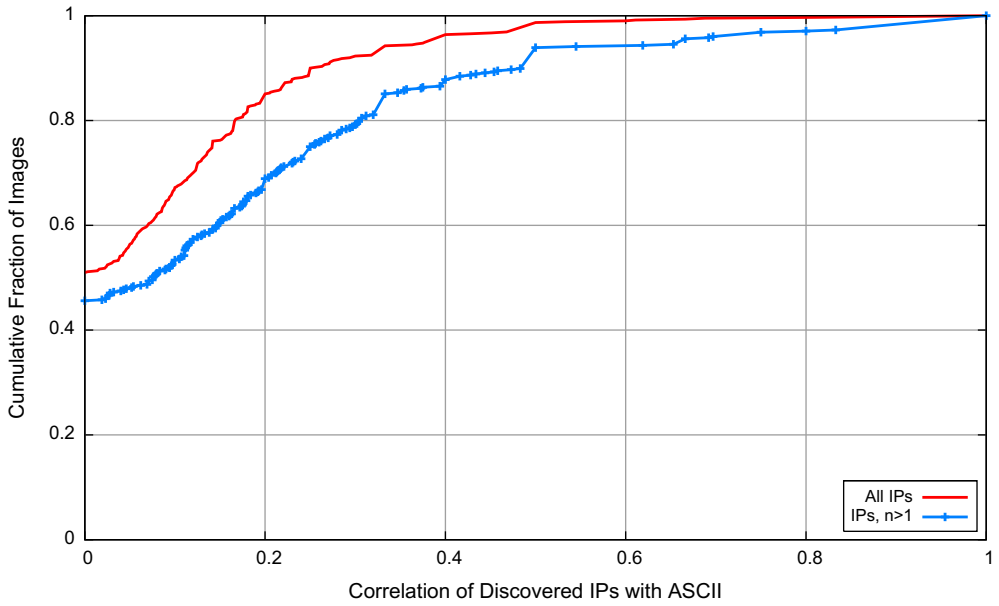

Fig. 5 – Correlation between scanning modalities across 723 images in the corpus: cumulative distribution of the fraction of IP addresses discovered in network structures also found in ASCII form.

WIN386.SWP Windows swap file. However, fully 58% of the images had discovered IP addresses that could not be attributed to any file as inferred by the file system data.

There are several mechanisms that could result in memory from hibernation and swap files being deposited into unallocated regions of the hard drive for which there is no corresponding file:

- After the WIN386.SWP is expanded by Windows, the file may be fragmented. When the file is later defragmented, some sectors may be mapped out of the file without being overwritten.
- Users may intentionally delete the hibernation file in an effort to recover disk space. Once deleted, the information in the sectors will remain until they are overwritten. If the system creates a new hibernation file, it is unlikely to occupy the same sectors.
- Other processes may result in memory being written to disk in user files that are then later deleted, leaving the memory behind as residual data.

Our experience in finding network artifacts in unallocated regions of the hard drive that are not associated with swapping and hibernation strongly suggests that significant

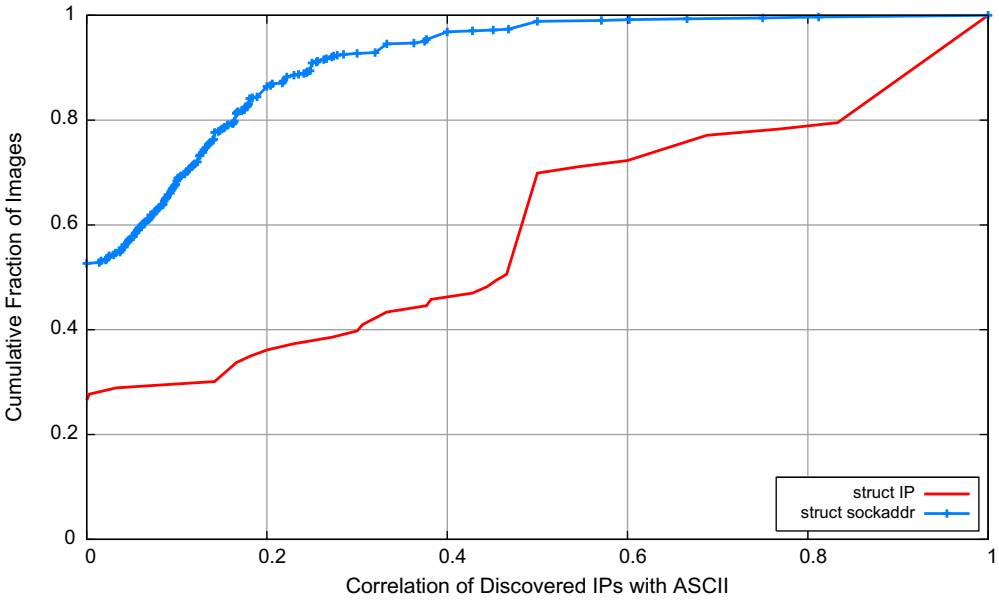

Fig. 6 – Difference in IP correlation between addresses discovered in IP versus socket structures across 723 images in the corpus.

amounts of valuable information from ephemeral stores (*e.g.* swap) is available only by examining the entire drive, and not restricting examination to the hiberfil.sys and WIN386.SWP files.

4.4.1. *Geolocation*

We scanned the NPS Real Data Corpus and found a total of 11,823 unique Ethernet MAC addresses on 74 drives. Of these addresses, the majority of them (11,355) were multicast addresses (as indicated by the 01:00:5E prefix Deering, August 1989), leaving 468 addresses associated with actual physical interfaces. We extracted these physical MAC addresses and attempted to run them through the Wigle, Skyhook and Google geolocation databases. We were not able to find any of the MAC addresses in the Wigle database. Skyhook told us that using their database in this manner was a violation of their terms-of-service. We were unable to find any of the addresses using Google's API, but we were never sure that it was operating properly due to the lack of error codes.

Our lack of success with this database should not be taken as evidence that the technique does not work, as the Wigle database is very sparse and the drives in the Real Data Corpus were primarily located in areas with little coverage by the Skyhook or Google databases. Nevertheless, it is evidence that a significant effort needs to be undertaken in building a comprehensive MAC address database before it can be successfully used to geolocate disk images in this fashion.

We had more luck performing Geolocation on extracted IP addresses. Partially this is because IP addresses are allocated in blocks and follow a typology. Using this information geolocation services are able to provide a physical location for virtually every publicly routable IP address—although certainly some of the reported locations are less accurate than others.

Images in the Real Data Corpus are generally tagged with the country in which they were acquired. Using this information as ground truth, we explored our ability to infer the origin of images on the basis of discovered IP addresses. In particular, we discover IP addresses on 30 images from India, 5 images from Mexico, 28 images from Israel, and 6 images from China. For each image, we use MaxMind to geolocate each discovered IP address and then infer the origin to be the most frequent country.

Using this approach, we achieve accuracies of 64%, 50%, 40%, and 23% for Israel, China, Mexico, and India respectively. However, we note that the most frequent mis-classification is the United States which has a large concentration of servers and infrastructure to which hosts from all over the world access, particularly those in less developed countries. This analysis therefore provides further validation against the larger corpus by confirming that the carved IP addresses represent real residual network traffic.

4.4.2. *Cross-drive analysis with MAC addresses*

Using the method outlined by Garfinkel (August 2006), we performed a multi-drive correlation of all 11,823 unique Ethernet MAC.

Our multi-drive correlation was able to identify several cases of computers in the NPS corpus having the same Ethernet MAC address present on their hard drive. We were able to identify 16 Ethernet MAC addresses that were present on more than one disk image. Treating both the MAC addresses and disk images as nodes in a graph and the presence of an address on an image as a link, we were able to partition the collection of disk images into eight distinct clusters (Table 5 and Fig. 7). Clusters #1, #2, and #3 all correspond to sets of drives that were removed from service at three different organizations in India and then resold on the secondary market, giving evidence that multi-drive correlation can be used to infer network membership.

Cluster #4 is somewhat confusing, as it contains three computers from India, one from Israel and one from the Palestinian Authority. However, the Ethernet prefix 01-00-5E is the Ethernet IPv4 Multicast prefix, and not an Ethernet address associated with a physical device. Indeed, a Google search for the Ethernet Mac address "01-00-5E-7F-FF-FA" we conducted in January 2011 resulted more than 52,000 hits, demonstrating that the address is widely used and thus not suitable for multi-drive correlation.

We have no explanation for Cluster #5, which appears to correlate a disk acquired in Israel with one acquired in Mexico. The Ethernet prefix 00-E0-D0 is allocated to the NetSpeed corporation, a company that was acquired by Cisco in 1999; it is possible that more than one machine is using that Ethernet MAC address.

Cluster #6 groups a computer from the Palestinian Authority with two computers from Israel. This may be a false match; then again, it may not be.

Table 5 – Using Ethernet MAC addresses for multi-drive correlation resulted in eight partitions. The social network graph of these MAC addresses and disk images is presented in Fig. 7.

Cluster	MAC addresses	Disk images in cluster
1	00-0E-90-D5-E6-5E 00-16-76-A2-60-6E 00-1B-B9-9B-D3-61 00-1B-B9-9B-D5-BB 00-1B-B9-9C-47-67 00-1E-90-D5-EE-5E 00-1E-90-DE-F1-07 00-1E-A6-01-9E-3A	IN10-0009 IN10-0010 IN10-0047 IN10-0048 IN10-0049 IN10-0050 IN10-0051 IN10-52
2	00-16-76-A2-60-6E	IN10-0413 IN10-0414
3	00-04-ED-66-C7-19 00-26-18-BD-D9-E9	IN10-0561 IN10-0562
4	01-00-5E-7F-FF-FA	IL38 IN10-0014 IN10-0095 IN10-0118 PS01-021
5	00-E0-D0-13-14-94	il42 mx5-30
6	00-50-04-EE-6C-F9	ps01-036 il02 il04
7	00-05-5F-EF-14-01 00-15-F2-4B-E5-1E	th01-01 cn20-01
8	00-D0-B7-69-0A-41	cn4-06 DE-1039

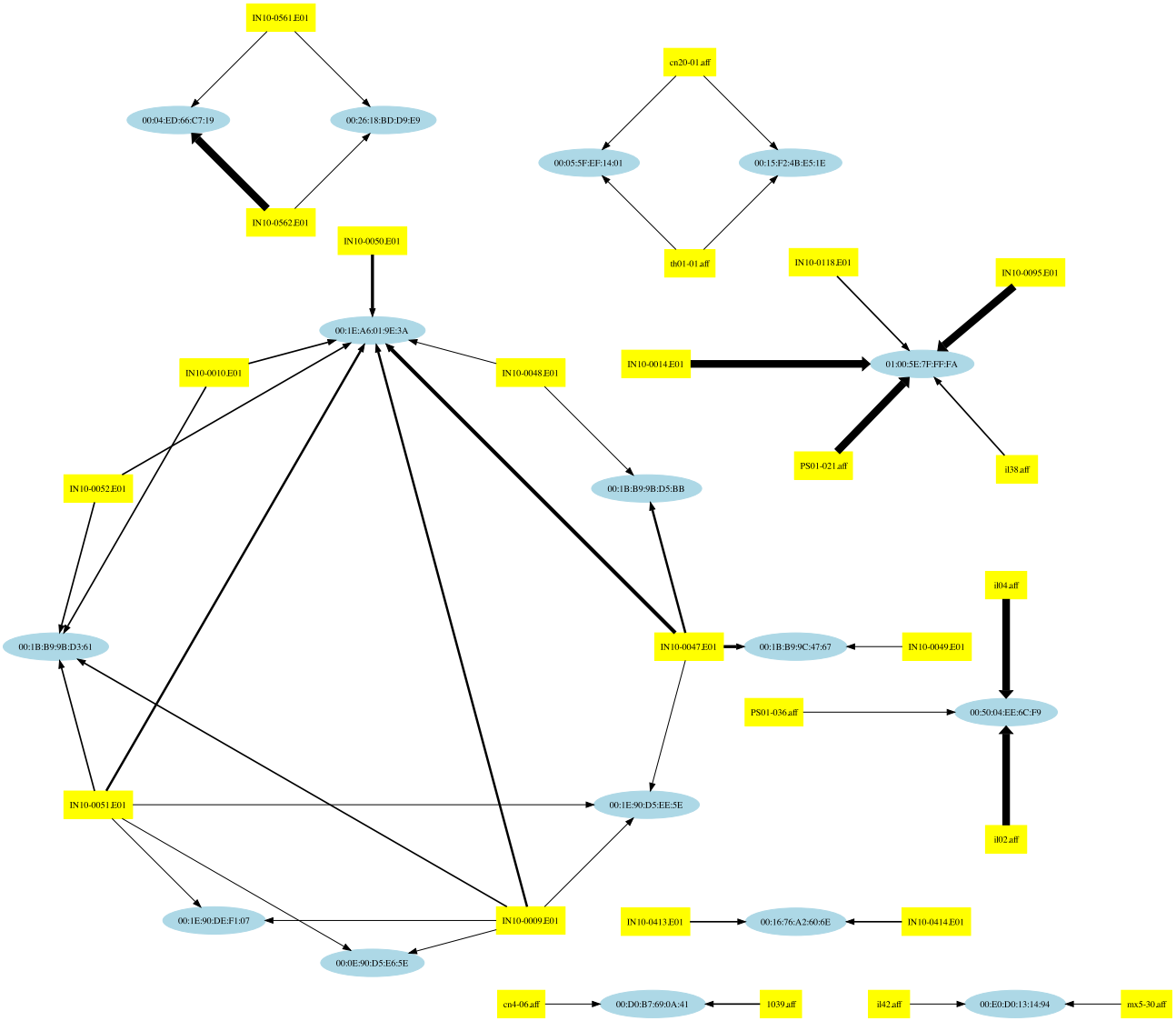

Fig. 7 – The social network graph of the table presented in Table 5. Squares indicate nodes, ovals indicate MAC addresses. Line width is proportional to the number of times that each MAC address was found on each node. This graph was generated automatically by running *bulk_extractor* on a set of 1800+ disk images, processing the resulting carved Ethernet addresses with a multi-disk correlation program, and sending the results to the GraphViz (Gansner and North, 2000) circo program for drawing. No human intervention was required for the selection of the graph nodes or the drawing of the links.

Cluster #7 matches a computer from Thailand with a computer from China with two MAC addresses, one assigned to Cisco Systems, one assigned to ASUSTek.

Cluster #8 matches a computer from China with a computer from Germany.

We believe that this very preliminary evaluation demonstrates the power of using MAC addresses for cross-drive analysis.

5. Conclusions

We have shown that IP packets and network data structures from memory are frequently persisted onto mass storage devices as a result of system hibernation and virtual memory. We have developed a tool that can be used to automatically extract this information.

5.1. *Code availability*

We have incorporated our code into the current release of *bulk_extractor*. The network carving code is enabled by default in version 1.0.

5.2. *Future work*

This initial work suggests many avenues for future research. We wish to investigate the extent to which residual network information can be better categorized and filtered to reconstruct network activities of interest. For instance, consider the