Computer Hard Drive Geolocation by HTTP Feature Extraction

Technical Report UCSC-SSRC-12-04
May 2012

Ziqian Wan
wan.ziqian@gmail.com

Alex J. Nelson
ajnelson@cs.ucsc.edu

Tao Li
litao@scu.edu.cn

Darrell D. E. Long
darrell@cs.ucsc.edu

Andy Hospodor
hospodor@soe.ucsc.edu

Storage Systems Research Center
Baskin School of Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064
http://www.ssrc.ucsc.edu/

Computer Hard Drive Geolocation by HTTP Feature Extraction

Ziqian Wan
Sichuan University

Alex J. Nelson
University of California, Santa Cruz

Tao Li
Sichuan University

Darrell D. E. Long
University of California, Santa Cruz

Andy Hospodor
University of California, Santa Cruz

Abstract

Geolocation data have high value to forensic investigators because computer activities may be associated with physical locations in the past. However, locating and extracting useful location information from an off-line disk image is a difficult problem. Most forensic investigations employ tools that focus on extracting content, such as emails, databases, and hidden or deleted data, and then manually investigate the results with practices like keyword searches. While this can work on a drive-by-drive basis, without a uniform approach to the location question, it is easy for an investigator to miss an answer that could be found from an evaluated technique known to other investigators.

To determine drive location, we develop a two-step approach that analyzes a drive image for geolocation purposes, finding substantial location information in HTTP headers from common and default sources. First, we extract HTTP headers from the memory page (swap) files that reside on the hard drive. Second, we apply a weight based algorithm that parses those headers to determine the past geographical locations of the drive. We apply our method to drive images from the publicly available M57 Patents corpus and identify the hard drives' location with low recall but high precision.

1 Introduction

The locations of a computer are valuable data in forensic investigations. However, investigators may only have a subject's powered-off computer and need evidence that pinpoints where it has been in the past. For example, one may need to corroborate a suspect's given travel story with evidence from their laptop. Today, this is often a difficult and manually intensive task. An automated method for extracting geographic information from the hard drive image would allow the investigators to spend less time finding forensic data and more time analyzing it.

We have devised a method to automatically recognize a computer's recent locations, solely by analyzing the HTTP header content from prominent websites. Our method exploits the web sites' practice of embedding the user's location in a recognizable metadata field. We found several major web sites, including the Windows default home page of Internet Explorer, provide a discoverable geolocation artifact in a predictable location.

Our method works around several significant challenges with hard drive geolocation, compared to network host location, including:

1. The only resource we have for hard drive location is the hard drive image itself. We cannot access real time memory data.

2. Real time feedback or data flow from the network is also not available to us. Thus, network based location methods, such as latency measurement [2], are not suitable for hard drive geolocation.

3. IP addresses, an old standard for locating a computer, suffer from a "Time shift" problem: The recorded address may not be valid by the time of an investigation.

While IP addresses are of classic utility for geolocation, we focus instead on evaluating the location value of HTTP headers in this work. We compare IP addresses' value for our data in a later section.

1.1 Our contribution

In this paper, we describe a tool we developed that extracts HTTP headers from the page file residing on the hard drive. We describe the tool's two step method in Section 3. We apply our approach to a realistic and available

*Ziqian Wan performed this work at the University of California, Santa Cruz.

data set described in Section 4. Experiments show that our method is nearly perfectly accurate with hard drives that contain geographic information within the HTTP headers, though those features were only available on 17% of our tested disk images. However, we found evidence that these features are highly likely to be available on computers that use default Windows settings.

2 Related work

Katz *et al.* proposed geolocating an Internet host by measuring the latency of packets crossing the Internet [12]. Arif *et al.* used an improved algorithm for network latency measurement [2]. Unfortunately, these methods require real time feedback on the Internet and are not suitable for offline hard drive geolocation.

IP address location has been the most common method for network host (server) geolocation. McCurley [15] and Buyukkokten *et al.* [5] used WHOIS lookup to build a database of IP address local information. They applied this database to web pages. The US Census Bureau provides a *Gazetteer*[1] constructed using the names of cities, counties and states extracted from US Census data [18]. Additional location data sets, such as Geolite [14], provide the place name, area code and longitude/latitude related to IP addresses.

Many web sites detect the geographic information of the client and provide related advertisements or services according to the user's current location. There has been much work done in the reverse as well, where the web server is located instead from its given resources. Muir and Oorschot provided a survey of Internet host geolocation technologies [16]. Wang *et al.* divided the web resources into three categories: provider location, content location and serving location [19]. Separately, Wang *et al.* proposed a method to detect geographic locations from these three types of web resources [20]. While this research could be applicable if we were analyzing users' web cache content, we focus instead on where the server believes the user is.

Extraction of geographic features from a web resource is another method employed in network geolocation. Many web resources, such as web pages and web sites, have associated geographic features [1, 5, 13, 15]. For example, Jones *et al.* found that local web pages are more likely to provide customized services according to the client's region, such as local weather report, local advertisements and tailored context services [11]. Many web sites first detect the geographic information of the client and then provide related advertisements and customized services according to the current user's location.

To focus on *client* geolocation, we extend the approach used by Garfinkel [8] and Beverly *et al.* [4] in applying regular expressions across regions of arbitrary data (a basic form of *carving*). Their original work focused on IP address, email addresses, bank account numbers, telephone numbers, zip codes and URLs. Like Garfinkel and Beverly, we extracted IP addresses by using a module of Bulk Extractor [9]. We performed ground truth experiments to map all the IPs to Google maps according to the Geolite database [14]. While we found that we could extract the IP address from an image's host, we also realized that it was difficult to differentiate the vast number of IP addresses in the corpus for geolocation purposes, which we discuss later.

In this paper, we introduce a new method for hard drive geolocation that focuses on extracting and analyzing the HTTP headers residing within memory that has been swapped to disk. We perform our geolocation technique without any network dependence, unlike all related work except Garfinkel [8] and Beverly *et al.* [4]. We propose an algorithm that extracts the geolocation information from these HTTP headers and correlates the hits to a likely physical location.

3 Analytical procedures

This section discusses the extraction procedure we conducted to determine a hard drive's geolocation. We describe the HTTP header nomenclature in Section 3.1 and salient memory features found in the Windows page file in Section 3.2. The HTTP header extraction method is proposed in Section 3.3. We then present the geolocating method in Section 3.4.

3.1 HTTP header nomenclature

We use these terms throughout the rest of the paper. They are drawn from the HTTP Protocol RFC [7].

HTTP header field: this component of the message header defines an operating parameter of an HTTP transaction request or response. A header field is divided into a *name* and *value* separated by a colon.

HTTP header: this is a list of header fields.

Cookie field: this field of the HTTP header identifies a user to a server, with a server-supplied token. The token content is entirely decided by the web server, with a *Set-Cookie* header field. This state information is retained on the user's computer and returned to the website during a subsequent HTTP request.

[1] A gazetteer is a geographical dictionary or directory, an important reference for information about places and place names.

Date field: describes the date and time that the HTTP message was sent.

HTTP headers are fairly easy to locate with text search: Each header begins with "`HTTP/`*version number*." In case we search through a page file and part of the text is reclaimed by arbitrary data, we note that characters falling outside a restricted set close to printable ASCII are not supposed to be found in headers, and particularly cookies [3, Section 4.1.1]. Such data are possibly valid header field values, as the cookie data are defined as octets which need not decode to valid UTF-8, and can be over 4KiB in length [3, Sections 5.4 and 6.1]. However, we have observed such data in neither the page files of the M57 corpus; nor the page files of a set of computers acquired in China, part of a larger research data set of real user data [10].

3.2 Important features in the page file

The page file residing on the hard drive contains uncommonly used RAM pages. Many recent network connections may be present. From that network data, we analyze IP addresses and then HTTP headers.

Ma and Tanaka [13] proposed that most web resources contain web pages, which have geographic features. Location based web applications can provide tailored information according to a user's location, such as local weather reports, location-based web search, and local advertisements. We found some web sites with significant amounts of traffic, such as CNN and The Financial Times, also record this location in the session cookie, illustrated in Figure 1. When the user visits the same web page or domain again, the user's web browser reads this cookie from disk and provides it to the web page or domain, placing another artifact in RAM.

This HTTP header information resides within the main memory of the client and will be lost if the system is powered off or the memory is overwritten. However, if the system needs more memory to deal with additional applications, these HTTP headers may be paged to the hard drive. Also, if the computer was suspended or hibernated this important data would end up in a file such as `hiberfil.sys`. For this paper, we focus on the Windows page file, as it was uniformly available in the M57 corpus.

3.3 Extracting HTTP headers

To extract headers from the page file, we perform string search. The regular expression "`HTTP/1.\d`" indicates a header, so we extract all the recognized header fields that immediately follow that regular expression match.

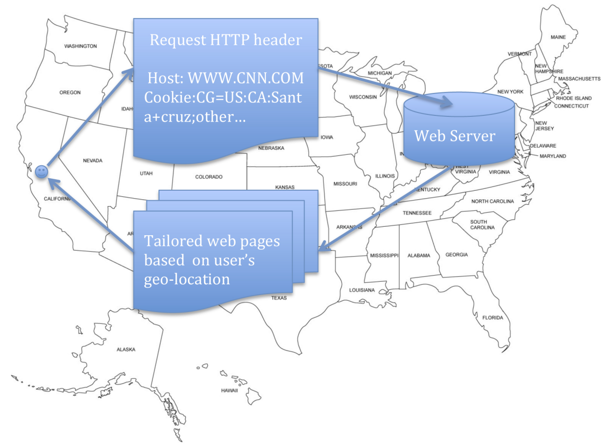

Figure 1: Cookie in the HTTP header, with the user's location used as part of the identifier.

A scan of the entire page file produces an HTTP header listing. Our locating method extracts the geographic information from this listing and calculates the image's location with the algorithm in the following section.

3.4 Locating algorithm

Our locating method used the 2010 U.S. Census Gazetteer [18] to ground the geographic references, extracted from HTTP headers, to a specific location.

We build two lists of US states, one for the state's full name and the other one for its two letter abbreviation. A city name list is built for each state, and each geographic item in the gazetteer has a weight value as shown in Figure 2. We extract each HTTP header cookie field to build a cookie list. Each state's full name and abbreviated name are searched through every cookie in the list. If we find one state's name in the cookie field, then we search that cookie for all the known cities within that state. If we match a city name then we increment that city's weight[2]. If we cannot find a city name in the same cookie, then, it's assumed to be a false positive. If the city's name is the same as its state's name, such as New York city is part of New York state, we do not increase its weight value unless it appears more than once. Finally, the city whose weight value is the highest is considered to be the domain location of the drive image. Our locating algorithm is described in algorithm 1.

[2]Oddly enough, in experiments on personal data not included in Section 4, we found we did not need to apply character decoding (*e. g.* `uudecode`) to find multi-word cities, such as "Santa Cruz."

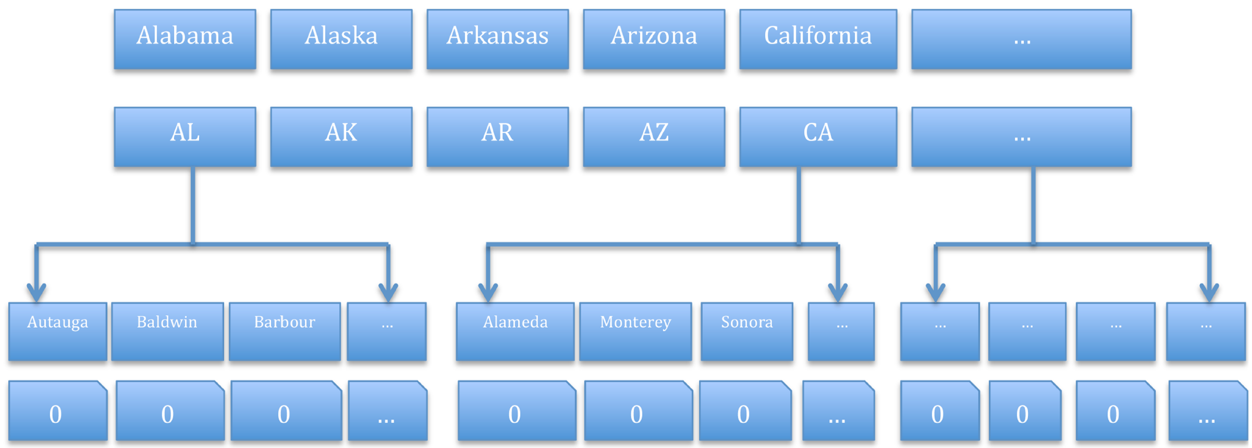

Figure 2: The gazetteer data structure supporting our city lookup algorithm. Locations are counted by identifying if a state string and city string are both identified in an HTTP header.

4 Experiments

We verify our method using drive images with known locations, from the M57 Patents research scenario [6, 21]. This collection of drives comes from a fictitious company called M57 Patents, which started operation on November 13th, 2009, and ceased operation on December 12, 2009. The scenario was developed in Monterey, California, USA. At least one disk image was taken for each of four employees every day, by rebooting their desktop Windows workstations into a Linux environment at day's end. The four employees in this company are: Charlie, Pat, Terry and Jo. The total data set has seventy-nine disk images, with an additional four non-computer storage images not considered.

4.1 Analysis of HTTP header

Figure 3 shows in hexedit view one HTTP header residing on Charlie's image when he visited the Financial Times web site on November 23rd. As can be seen in the cookie field, the user's location and then-IP address are embedded. Figure 4 shows that HTTP header resides on Pat's image when he visited the MSNBC web site on November 23rd. We see here that the cookie field contains his city and zip code.

We find the tracking information left by local.msn.com to be substantial, which empowers this geolocation method. local.msn.com stores the user's city, zip code and latitude/longitude in the cookie. msn.com loads local.msn.com for a local weather report, generating a location artifact in RAM. This is significant because msn.com is the default web page for Windows, as observed in US-localized copies of version XP and 7. Hence, if a Windows computer does not have its default home page changed, the default browser on-open behavior of viewing the home page leaves timestamped geolocation evidence in memory.

From the time information, one seems more likely to find recent locations from HTTP headers. We extracted the Date field of each header, parsing the value with the `dateutil` Python library [17], to determine the overall header age distribution. For simplicity of parsing, we only counted Date values that didn't end with "GMT." From Figures 5 and 6, we can see many HTTP headers are paged to the hard drive on the same day that the image was built. That means if a location is recorded, it will be a recent location.

These headers also clarified an inaccuracy in the file system timestamps of `pagefile.sys`. As we processed the M57 data, XP and Vista machines, we found the mtime was typically accurate to only within the last day or two of activity. For instance, some Monday images left the page file's mtime as late in the previous week. Making the assumption that the header times were not perturbed in memory after being received from the sending web servers, these embedded timestamps within the page file present a more accurate picture of the system's recent use time.

4.2 Drive location

We processed the M57 data set with our location-identifying method. Table 1 shows the results with the place name and weight value of each image. Although all images in this data set were from Monterey, California, US, we were only able to identify a location – any location – for fourteen of the images. The other images did

```
034519C0  50002100 5A010802 00000000 2F636F6E 74656E74 2F696D61 6765732F 37326161  P.!.Z......./content/images/72aa
034519E0  35316263 2D343438 642D3131 64652D38 3264362D 30303134 34666561 62646330  51bc-448d-11de-82d6-00144feabdc0
03451A00  2E696D67 20485454 502F312E 310D0A48 6F73743A 20696D2E 6D656469 612E6674  .img HTTP/1.1..Host: im.media.ft
03451A20  2E636F6D 0D0A5573 65722D41 67656E74 3A204D6F 7A696C6C 612F352E 30202857  .com..User-Agent: Mozilla/5.0 (W
03451A40  696E646F 77733B20 553B2057 696E646F 7773204E 5420352E 313B2065 6E2D5553  indows; U; Windows NT 5.1; en-US
03451A60  3B207276 3A312E39 2E312E35 29204765 636B6F2F 32303039 31313032 20466972  ; rv:1.9.1.5) Gecko/20091102 Fir
03451A80  65666F78 2F332E35 2E350D0A 41636365 70743A20 696D6167 652F706E 672C696D  efox/3.5.5..Accept: image/png,im
03451AA0  6167652F 2A3B713D 302E382C 2A2F2A3B 713D302E 350D0A41 63636570 742D4C61  age/*;q=0.8,*/*;q=0.5..Accept-La
03451AC0  6E677561 67653A20 656E2D75 732C656E 3B713D30 2E350D0A 41636365 70742D45  nguage: en-us,en;q=0.5..Accept-E
03451AE0  6E636F64 696E673A 20677A69 702C6465 666C6174 650D0A41 63636570 742D4368  ncoding: gzip,deflate..Accept-Ch
03451B00  61727365 743A2049 534F2D38 3835392D 312C7574 662D383B 713D302E 372C2A3B  arset: ISO-8859-1,utf-8;q=0.7,*;
03451B20  713D302E 370D0A4B 6565702D 416C6976 653A2033 30300D0A 436F6E6E 65637469  q=0.7..Keep-Alive: 300..Connecti
03451B40  6F6E3A20 6B656570 2D616C69 76650D0A 52656665 7265723A 20687474 703A2F2F  on: keep-alive..Referer: http://
03451B60  7777772E 66742E63 6F6D2F68 6F6D652F 75730D0A 436F6F6B 69653A20 41595343  www.ft.com/home/us..Cookie: AYSC
03451B80  3D5F3034 63615F31 33555341 5F313455 53415F31 37736F75 74687765 73745F31  =_04ca_13USA_14USA_17southwest_1
03451BA0  386D6F6E 74657265 795F3234 6E6F7274 68253235 3230616D 65726963 615F3235  8monterey_24north%2520america_25
03451BC0  68696768 5F323638 33315F32 37505654 5F3B2046 54557365 72547261 636B3D32  high_26831_27PVT_; FTUserTrack=2
03451BE0  30352E31 35352E36 352E3130 332E3431 34343132 35383430 37383931 3934373B  05.155.65.103.41441258407891947;
03451C00  20727369 5F736567 733D4A30 37373137 5F313032 39363B20 475A4950 3D313B20   rsi_segs=J07717_10296; GZIP=1;
03451C20  46544D44 3D71763B 20727369 5F63743D 32303039 5F31315F 32333A31 0D0A0D0A  FTMD=qv; rsi_ct=2009_11_23:1....
```

Figure 3: A nearly-intact HTTP request header residing on Charlie's image for Nov. 23rd, as displayed by `hexedit`. Byte addresses are within-file addresses of `pagefile.sys`. We observe the cookie field, highlighted, includes what appear to be an accurate city, IP address (`205.155.65.103`, whose location resolves to Monterey, CA) and an accurate date (`2009_11_23`).

```
12FB7C20  25324361 61353730 44002100 8C010A02 00000000 2F646566 61756C74 2E617368  %2Caa570D.!........./default.ash
12FB7C40  782F6964 2F323732 38373230 32204854 54502F31 2E310D0A 41636365 70743A20  x/id/27287202 HTTP/1.1..Accept:
12FB7C60  2A2F2A0D 0A526566 65726572 3A206874 74703A2F 2F777777 2E6D736E 62632E6D  */*..Referer: http://www.msnbc.m
12FB7C80  736E2E63 6F6D2F69 642F3333 38393130 37382F6E 732F7465 63686E6F 6C6F6779  sn.com/id/33891078/ns/technology
12FB7CA0  5F616E64 5F736369 656E6365 2D737061 63652F3F 4754313D 34333030 310D0A41  _and_science-space/?GT1=43001..A
12FB7CC0  63636570 742D4C61 6E677561 67653A20 656E2D75 730D0A55 7365722D 4167656E  ccept-Language: en-us..User-Agen
12FB7CE0  743A204D 6F7A696C 6C612F34 2E302028 636F6D70 61746962 6C653B20 4D534945  t: Mozilla/4.0 (compatible; MSIE
12FB7D00  20382E30 3B205769 6E646F77 73204E54 20352E31 3B205472 6964656E 742F342E   8.0; Windows NT 5.1; Trident/4.
12FB7D20  30290D0A 41636365 70742D45 6E636F64 696E673A 20677A69 702C2064 65666C61  0)..Accept-Encoding: gzip, defla
12FB7D40  74650D0A 486F7374 3A207777 772E6D73 6E62632E 6D736E2E 636F6D0D 0A436F6E  te..Host: www.msnbc.msn.com..Con
12FB7D60  6E656374 696F6E3A 204B6565 702D416C 6976650D 0A436F6F 6B69653A 204D4331  nection: Keep-Alive..Cookie: MC1
12FB7D80  3D563D33 26475549 443D6437 33616364 64656139 63313439 66623839 31343263  =V=3&GUID=d73acddea9c149fb89142c
12FB7DA0  62306132 35613330 62633B20 4D554944 3D443639 35434139 38413936 44343433  b0a25a30bc; MUID=D695CA98A96D443
12FB7DC0  45394636 42414346 45453533 39433133 353B206D 683D4D53 46543B20 43433D55  E9F6BACFEE539C135; mh=MSFT; CC=U
12FB7DE0  533B2043 554C5455 52453D45 4E2D5553 3B207A69 703D7A3A 39333934 307C6C61  S; CULTURE=EN-US; zip=z:93940|la
12FB7E00  3A33362E 367C6C6F 3A2D3132 312E3839 317C633A 55537C68 723A313B 2051313D  :36.6|lo:-121.891|c:US|hr:1; Q1=
12FB7E20  39333934 302C6D6F 6E746572 65792C63 612C756E 69746564 20737461 7465732C  93940,monterey,ca,united states,
12FB7E40  75730D0A 0D0A7469 1D004400 4B006502 A0C36E01 28F84A01 436F6F6B 69653A20  us....ti..D.K.e...n.(.J.Cookie:
```

Figure 4: One HTTP header residing in `pagefile.sys` of Pat's image for Nov. 16th. Similar to Figure 3, specific location information is reported within the cookie identifier.

not contain enough geographic information in their HTTP headers for accurate geolocation.

There are 29,513 cities in the U.S. according to the 2010 Census Gazetteer. Fourteen images in the M57 corpus had the geographic HTTP headers, and all these images were geolocated to Monterey, California.

Table 1: The highest non-zero location weights of M57. All weights were for Monterey, California.

Image Owner	*Date*	*RAM Size*	*Pagefile Size*	*Weight value*
Charlie	20091116	1 GiB	2046 MiB	174
Charlie	20091117	1 GiB	2046 MiB	42
Charlie	20091123	1 GiB	2046 MiB	87
Charlie	20091207	1 GiB	2046 MiB	142
Charlie	20091208	1 GiB	2046 MiB	89
Charlie	20091209	1 GiB	2046 MiB	58
Charlie	20091210	1 GiB	2046 MiB	58
Charlie	20091211	1 GiB	2046 MiB	58
Pat	20091116	256 MiB	768 MiB	9
Pat	20091117	256 MiB	768 MiB	1
Pat	20091118	256 MiB	768 MiB	1
Terry	20091116	2 GiB	1024 MiB	145
Terry	20091117	2 GiB	2346.3 MiB	120
Terry	20091118	2 GiB	2346.3 MiB	126

4.3 Comparison with IP-based location

We also performed ground truth experiments on the M57 corpus to analyze the feasibility of the IP geolocation method and compared it with our method. We used a module of Bulk Extractor [9] to extract IP addresses and the Geolite data [14] for February, 2011 to assign locations to them.[3] We classified the location accuracy as

[3]Because this data set is more recent, we verified that the publicly visible IP address within the M57 scenario would still be listed as Monterey in these data.

Table 2: The results of IP extracting compared with our method. The in-US counts include Monterey, CA.

		Non-distinct IPs			*Distinct IPs*			
Image owner	*Image date*	*Geolocatable to Monterey*	*Geolocatable in US*	*Geolocatable outside US*	*Geolocatable to Monterey*	*Geolocatable in US*	*Geolocatable outside US*	*Geolocatable HTTP headers*
Charlie	20091116	4	5329	10671	1	1005	1577	174
Charlie	20091117	6	4894	10052	1	958	1577	42
Charlie	20091123	10	4810	10779	1	1011	1610	87
Charlie	20091203	12	4648	10849	1	977	1617	0
Charlie	20091204	12	4977	11808	1	988	1623	0
Charlie	20091207	12	4716	11031	1	972	1632	142
Charlie	20091208	12	4495	10809	1	965	1629	89
Charlie	20091209	12	4748	10944	1	990	1633	58
Charlie	20091210	52	4315	11328	1	979	1602	58
Charlie	20091211	52	4322	11307	1	979	1602	58
Pat	20091116	0	8640	12964	0	1020	1598	9
Pat	20091117	2	7356	12722	1	1025	1575	1
Pat	20091118	4	9440	13879	1	1062	1587	1
Pat	20091124	18	10806	16288	1	1019	1622	0
Pat	20091130	16	12148	16504	1	1053	1651	0
Pat	20091201	10	12115	16512	1	1029	1645	0
Pat	20091202	10	12047	15925	1	1077	1648	0
Pat	20091203	12	10262	15212	1	1048	1652	0
Pat	20091204	12	10163	15134	1	1014	1607	0
Pat	20091207	12	10488	15760	1	1015	1616	0
Pat	20091208	14	13055	18622	1	1015	1611	0
Pat	20091209	14	12755	17796	1	1021	1613	0
Pat	20091210	14	13279	18479	1	1042	1643	0
Pat	20091211	14	13976	19922	1	1045	1649	0
Terry	20091116	10	721000	64309	1	1510	2177	145
Terry	20091117	10	531300	63684	1	1494	2179	120
Terry	20091118	12	422137	68223	1	1512	2163	126
Terry	20091123	16	647091	78019	1	1422	2250	0

correct-city, in-US (including Monterey), and non-US, and list the tallies in Table 2.

We found nineteen images had IP addresses that resolved in or geographically very close to Monterey. However, the quantity of in-city IP occurrences is orders of magnitude smaller than the quantity of IPs associated with other locations. Visual analysis from mapping the beginning and endpoints of supposed IP communications also laid the IP addresses literally all over the map, which leads us to believe Bulk Extractor produces significant amounts of false-positive IP addresses. We conclude that IP address geolocation can corroborate other location evidence, however there is a significant risk for false positive matches from extracted addresses that may simply be data misidentified as IPs.

5 Future considerations

We focused on extracting HTTP headers from the Windows page file to determine a drive's past location. Unfortunately, the recall rate from our data was fairly low. Also, our measurements of HTTP header dates (whether they had location information or not) in Figures 5 and 6 cast doubt on the longevity of the artifacts on which our location inference technique relies. While this work establishes that a location is discoverable through HTTP headers, we require additional, more diverse data to properly study how often and under what conditions headers yield location data. One data set we could consider is the Real Data Corpus [10], but unfortunately those data lack ground truth on locations of actual use.

We acknowledge, but did not consider the Windows hibernation file in this work. Our data came from desktop computers that were shut down at the end of their work days. One computer in the M57 corpus (Terry) was a Vista machine with some updates to its `hiberfil.sys`, noted in mtimes and checksums. However, the other three M57 computers were Windows XP machines without the hibernation file. We chose to use the uniformly available page files, which gave us sufficient network information to make claims about the host's location. We suspect that other data residing within the hibernation file, coupled with the growing popularity of laptop and handheld devices, will yield additional forensic information.

As operating systems and applications continue to grow in size they put an increasing burden on the paging system. As more information is paged onto the hard drive, even if it is later unallocated, our method becomes more effective.

6 Conclusion

In this paper, we proposed a method to extract geographic features from the hard drive images, focusing on page files containing HTTP artifacts. We developed an algorithm

Algorithm 1 Locating method.

```
Input: HTTP header file, state list file, city list file
  statenamelist= read from state list file
  fullnamelist =read from state list file
  place[][]= read from city list file
  weight[][]
  cookie[]
  for i in len(place) do
    for j in len(place[i]) do
      weight[i][j]=0
    end for
  end for
  for header in HTTP header file do
    temp = header.split('\n')
    for i in len(temp) do
      if temp[i].startswith('Cookie:') then
        cookie.append(temp[i])
      end if
    end for
  end for
  for i in len(cookie) do
    for stateindex in len(place) do
      stateabbrnamecount=cookie[i].upper().find(
      statenamelist[stateindex].upper())
      statefullnamecount=cookie[i].upper().find(
      fullnamelist[stateindex].upper())
      count=stateabbrnamecount+statefullnamecount
      if count >0 then
        for cityindex in len(place[stateindex]) do
          citycount=cookie[i].upper().find(
          state[stateindex][cityindex].upper())
          if citycount >0 then
            weight[stateindex][cityindex]++
          end if
        end for
      end if
    end for
  end for
```

to geolocate a hard drive image from HTTP headers, exploiting client-server session identifiers that contain the user's location. Experimental results on a publicly available disk image set showed that our method can identify a disk image's location when certain telling HTTP headers have been paged to disk. Our initial experiments focused on the United States, using location names from the US Census Bureau. In the future, a global gazetteer may extend our geolocation method to any named location on the planet.

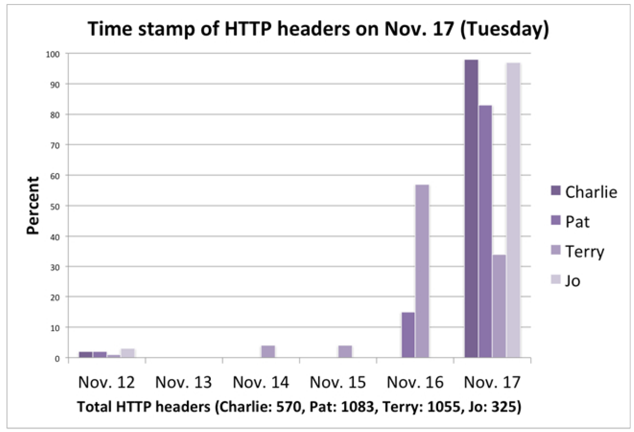

Figure 5: Time stamps of HTTP headers on Nov. 17.

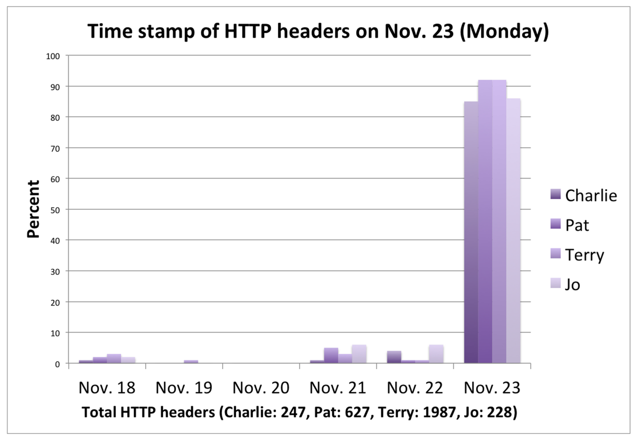

Figure 6: Time stamps of HTTP headers on Nov. 23.

Acknowledgments

The authors would like to thank DJ Capelis, Zhike Zhang, Xiaoyan Zhu, Ranjana Rajendran, and the anonymous reviewers of IEEE Transactions on Information Forensics and Security for their valuable comments on this paper.

This product includes GeoLite data created by MaxMind, available from `http://www.maxmind.com/`.

References

[1] E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-Where: Geotagging web content. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 273–280, New York, NY, USA, 2004. ACM.

[2] M. J. Arif, S. Karunasekera, and S. Kulkarni. GeoWeight: Internet host geolocation based on a probability model for latency measurements. In *Pro-*

ceedings of the Thirty-Third Australasian Conference on Computer Science, volume 102 of *ACSC '10*, pages 89–98, Darlinghurst, Australia, 2010. Australian Computer Society, Inc.

[3] A. Barth. RFC 6265: HTTP State Management Mechanism, 2011.

[4] R. Beverly, S. Garfinkel, and G. Cardwell. Forensic carving of network packets and associated data structures. *Digital Investigation*, 8:S78–S89, 2011.

[5] O. Buyukkokten, J. Cho, H. Garcia-Molina, L. Gravano, and N. Shivakumar. Exploiting geographical location information of web pages. In *ACM SIGMOD Workshop on The Web and Databases (WebDB'99)*, 1999.

[6] DigitalCorpora.org. M57 Patents Scenario Disk Images. `http://digitalcorpora.org/corpora/scenarios/m57-patents-scenario`, 2009. Accessed December 2010.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616: Hypertext Transfer Protocol–HTTP/1.1, 1999.

[8] S. Garfinkel. Forensic feature extraction and cross-drive analysis. *Digital Investigation*, 3:71–81, 2006.

[9] S. Garfinkel. Bulk Extractor 1.0.0. `http://afflib.org/software/bulk_extractor`, June 2011. Accessed June 2011.

[10] S. L. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt. Bringing science to digital forensics with standardized forensic corpora. In *Proceedings of the 9th Annual Digital Forensic Research Workshop (DFRWS)*, Quebec, Canada, August 2009.

[11] M. Jones, P. Jain, G. Buchanan, and G. Marsden. Using a mobile device to vary the pace of search. *Human-Computer Interaction with Mobile Devices and Services*, pages 390–394, 2003.

[12] E. Katz-Bassett, J. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe. Towards IP geolocation using delay and topology measurements. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, pages 71–84. ACM, 2006.

[13] Q. Ma and K. Tanaka. Retrieving regional information from web by contents localness and user location. In S. Myaeng, M. Zhou, K.-F. Wong, and H.-J. Zhang, editors, *Information Retrieval Technology*, volume 3411 of *Lecture Notes in Computer Science*, pages 301–312. Springer Berlin / Heidelberg, 2005.

[14] MaxMind, Inc. GeoLite City. `http://www.maxmind.com/app/geolitecity`, 2010. Accessed December 2010.

[15] K. S. McCurley. Geospatial mapping and navigation of the web. In *Proceedings of the 10th International Conference on World Wide Web*, pages 221–229. ACM, 2001.

[16] J. A. Muir and P. C. van Oorschot. Internet geolocation: Evasion and counterevasion. *ACM Computing Surveys*, 42:4:1–4:23, December 2009.

[17] G. Niemeyer. python-dateutil. `http://labix.org/python-dateutil`, 2011. Accessed January 2012.

[18] US Census Bureau. 2010 Gazetteer. `http://www.census.gov/geo/www/gazetteer/gazette.html`, 2010. Accessed July 2011.

[19] C. Wang, X. Xie, L. Wang, Y. Lu, and W. Ma. Web resource geographic location classification and detection. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, pages 1138–1139. ACM, 2005.

[20] C. Wang, X. Xie, L. Wang, Y. Lu, and W. Y. Ma. Detecting geographic locations from web resources. In *Proceedings of the 2005 Workshop on Geographic Information Retrieval*, pages 17–24. ACM, 2005.

[21] K. Woods, C. Lee, S. Garfinkel, D. Dittrich, A. Russel, and K. Kearton. Creating realistic corpora for forensic and security education. In *2011 ADFSL Conference on Digital Forensics, Security and Law*, Richmond, VA, 2011. Elsevier.