Browser Forensics

Peter C. Hewitt
GCFA, GCIH, GAWN, GCWN, GWAS, CISSP, CISA, CPA

Manuel Humberto Santander Peláez
GCFA, GCIH, GCIA, GCFW, GNET, SSP-MPA

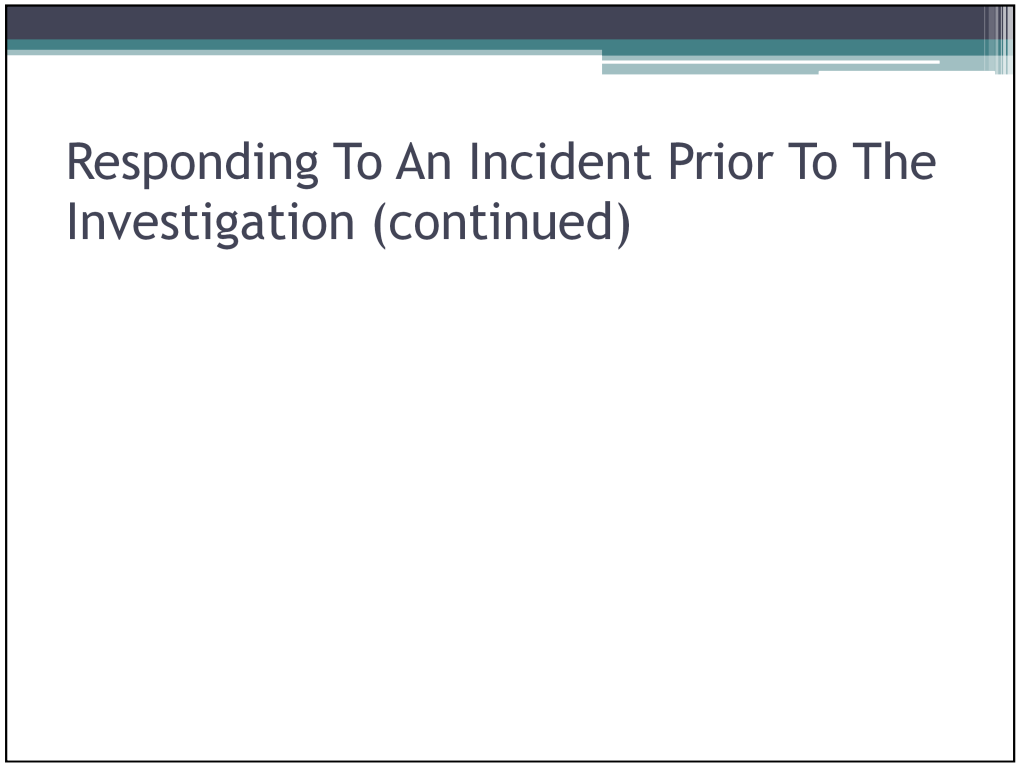
This page intentionally left blank.

Responding To An Incident Prior To The Investigation

- Web browser forensics are usually performed in response to an incident
- Either you
  - Were able to lock down the incident and want to find out what the heck went wrong
  - Were unable lock down the incident and NEED to find out what the heck went wrong!

Why would you want to perform a forensic analysis of a computer system? Most of the time, it's in response to an incident – e.g. something's gone wrong, and that something may have been actively perpetrated by a person (rather than a simple hardware failure or software error.) Oftentimes (and especially in web browser forensics) the evidence you gather may not even be directly related to the incident – if a person is terminated for over-socializing during work hours, and management suspects that the now-ex-employee may file a wrongful termination suit, you may be called in to see if s/he made use of any of the common social-networking sites (Facebook, MySpace, Friendster) to bolster management's assertions. Other reasons to perform an investigation include internal policy violations (conducting personal business at work, harassment), crime (child pornography, fraud, theft of trade secrets), and downloading of forbidden software (especially hacking tools.)

Alternatively, your forensic review may be directly related to an incident. If your company's main office network was nearly taken down by a virus infection or "0-day" exploit that you barely managed to beat back via disabling the external routers and manually updating / cleaning each workstation, you might want to find where the problem originated so you could prevent it from happening again (especially if the problem was an employee surfing to a non-work site and downloading unauthorized code.) In the real nail-biting scenario, all your protective controls have failed, and you need to know which employee uploaded the latest design specs to your biggest competitor's website, if for no other reason than to prove the competitor's malfeasance in a lawsuit.



Prior to discussing a forensic review, let's review the essential steps in the Incident Response Process. To minimize the potential for losing evidence, experienced incident handling professionals divide the process into six phases:

•*Preparation* comprises the controls put in place in anticipation of an incident occurring, such as the basic step of having a written incident response program you can follow.

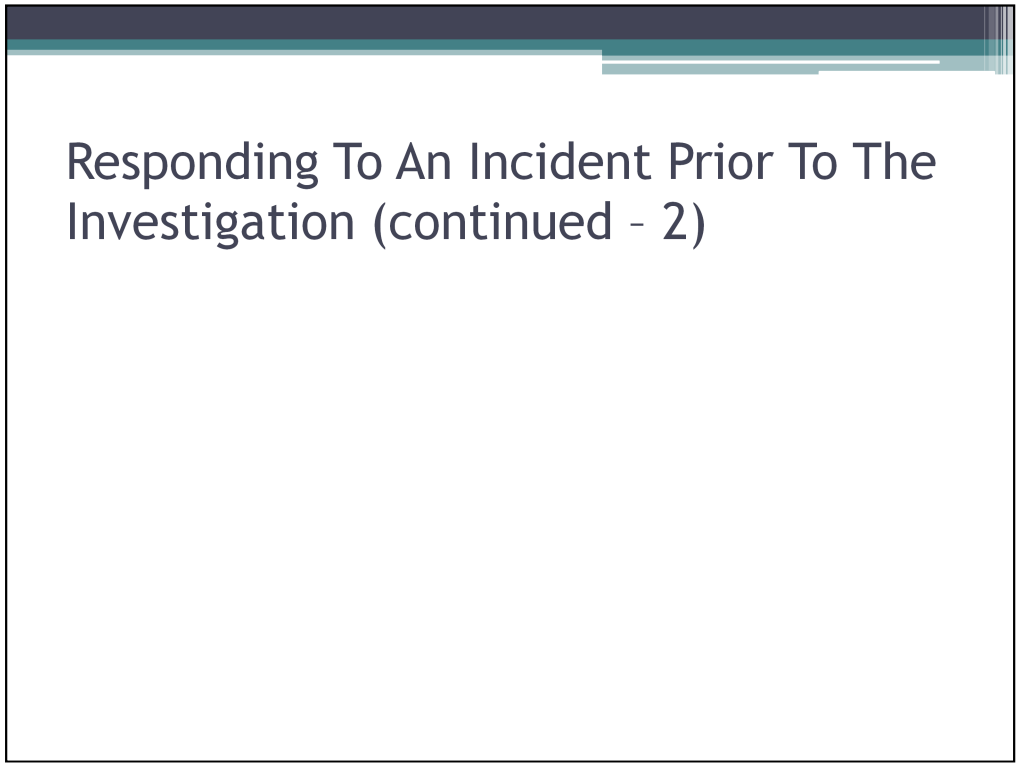
•*Identification* means determining that an incident actually took place, and if so, the nature of the incident and the extent of the damage.

•*Containment* is the process whereby the incident (and the effects of it) are prevented from spreading or getting any worse.

•*Eradication* entails removing the cause of the incident. In the case of a fast-spreading malicious code incident, eradication may involve not only removing the cause from the source computer, but "stamping it out" on every last machine that was subsequently infected. In the case of a network intrusion, eradication will require the "locking down" of the intrusion point (vulnerable port, source IP address); unfortunately, the intruder may simply choose to attack via a different avenue.

•*Recovery* is the set of actions taken to restore all affected systems to operational status. This can range from re-enabling network connectivity to a full system restoration from backups (which requires you to verify the integrity of the backups prior to restoration.)

•*Lessons Learned* is the most-often-neglected step in Incident Reponse, because most people just want to get back to business and forget the incident ever occurred. It is vital, however, to review the incident and the steps taken to respond to it, in order to improve the response process for the next time – and there *will* be a next time. Additionally, Lessons Learned allows you to apply the cardinal rule of both Incident Response and Forensics – *document everything*.



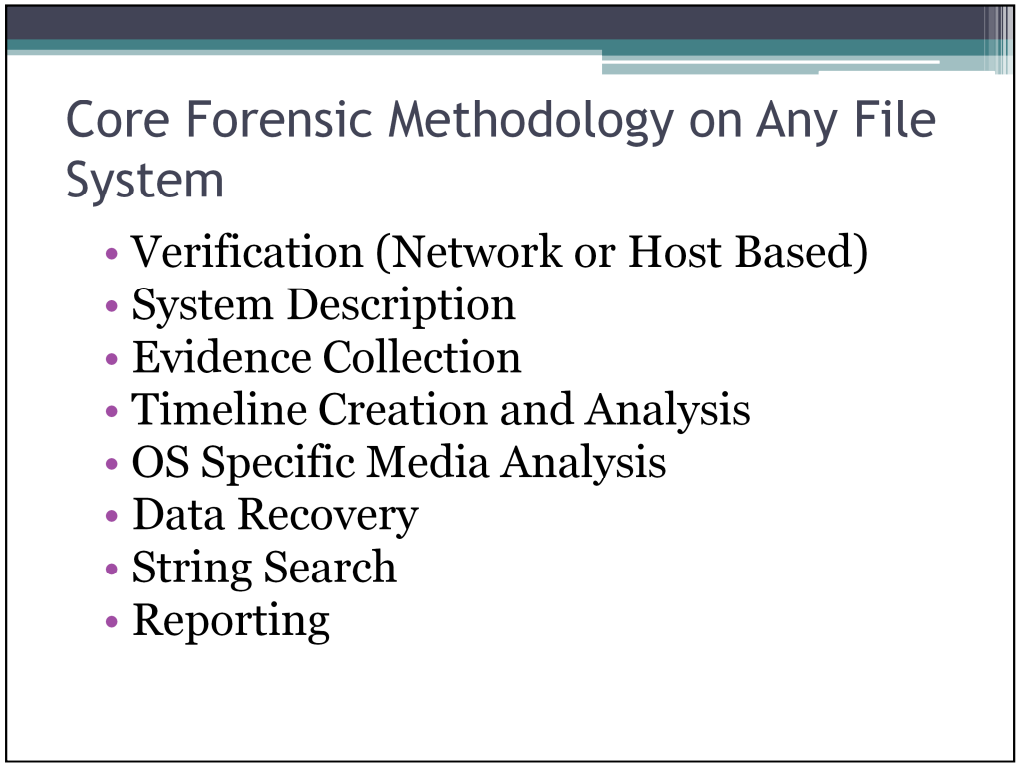
Your forensic investigation will likely begin at the Eradication step and proceed onward with the Incident Response. Note that you need to find a balance between responding to the incident (ensuring business can get back to normal) with the need to preserve forensic evidence. Eradication and Recovery can easily erase all of the intruder's tracks!

Order of Volatility of Evidence

Type of Data	Lifespan
• Registers, peripheral memory, caches, etc.	• Nanoseconds or less
• Main memory	• Ten nanoseconds
• Network state	• Milliseconds
• Running processes	• Seconds
• Disk	• Minutes
• Floppies, backup media, etc.	• Years
• CD-ROMs, printouts, etc.	• Tens of years

Not all information-based evidence is the same! Evidence can be organized into an "order of volatility" meaning how long it will stick around for you to collect until it automatically is lost. Dan Farmer & Wietse Venema created the above table of evidence volatility, which is commonly referenced by forensic professionals. For example, information stored on a CD-ROM or similar optical media can last for 10-100 years, depending on the brand used. Information stored in a computer's main memory, by contrast, will last for only tens of nanoseconds before it is wiped out by the computer's normal processing.

Going back to the balance of responding to an incident versus preserving information-based evidence, most people will want to turn a computer off (or at the very least unplug it from the network) when they realize an incident has occurred. However, as noted in the chart above, you will lose evidence in main memory and "network state" information (which other systems the computer is connected with and what information they are exchanging) with such an approach. Even shutting down a computer the "normal" way (Start / Turn Off Computer / Turn Off in Windows XP) can delete evidence, as Windows performs a number of housekeeping tasks in the shutdown process, such as closing opened files and clearing out the temporary disk cache. A forensic examiner needs to keep in mind evidence volatility when determining how to proceed with an incident response and forensic investigation.



Just like with incident handling, it doesn't matter what type of computer or operating system you're examining; in a forensic investigation, there are specific steps that need to be followed.

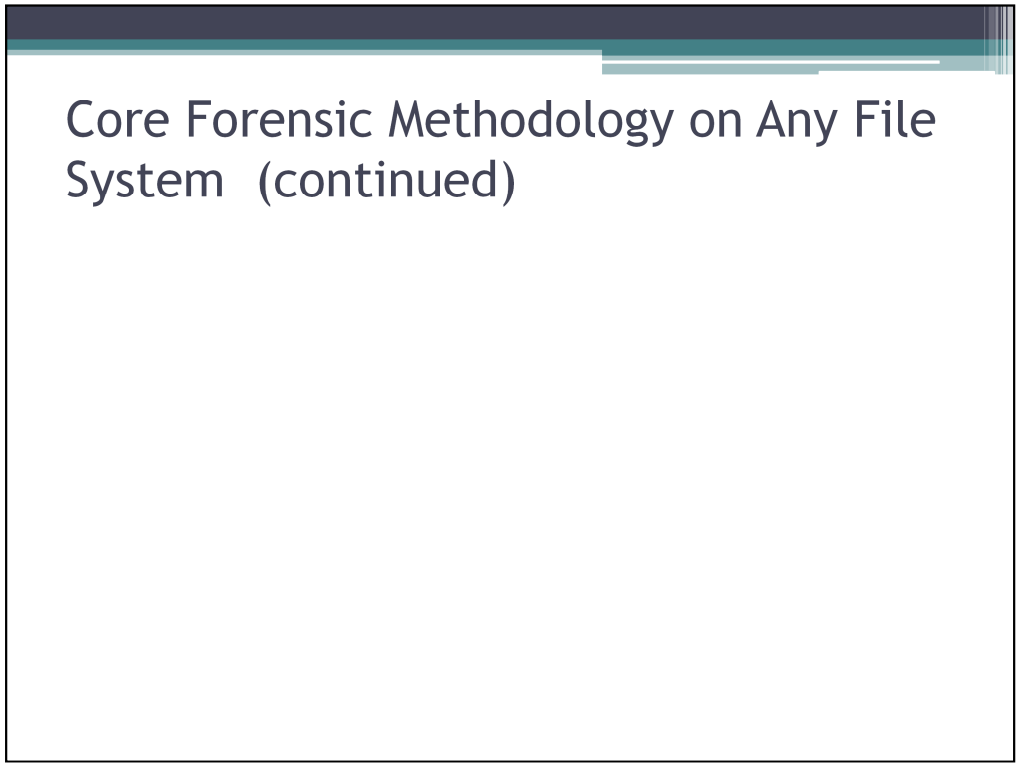
Verification – Similar to the Identification step in Incident Response. What are the symptoms of the incident? What information was involved (this can be critical to determining if a forensic investigation is necessary)? How many systems were affected?

System Description – Continuing in our "document everything" frame of mind, create a detailed description of the system under review. "Dell Optiplex Mini-ATX Tower P4/3.2gHz, serial number 6342734897" is the level of detail you want, as well as a tag number so you can organize the evidence you've collected. Separately describe the hard drive and peripherals (optical drives, network cards) and give them tag numbers as well.

Evidence Collection – Bring the evidence into a place where you can examine it. Most of the time, you will want to copy the evidence to another computer rather than analyze it on the original machine.

Timeline Creation and Analysis – Based on the evidence you've reviewed, create a timeline of what happened when and by whom. Some programs can do this for you automatically, based on the file Modification / Access / Change ("MAC" metadata) times kept by the operating system.

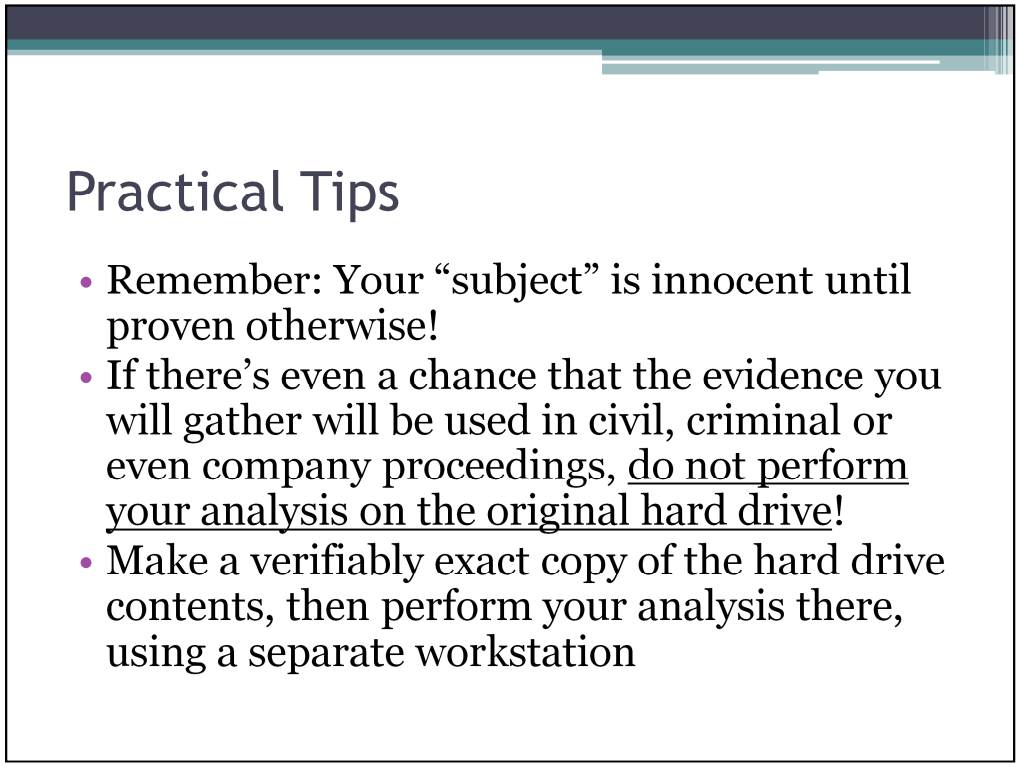
OS Specific Media Analysis – The "nitty gritty" of your analysis, where you pore through the various files on the system to determine what took place. Common areas to search are the My Documents folder, the Windows / WINNT folder (for evidence of system compromise such as rootkits) and, as we will see, the folders containing the user's web browser history and saved files.



Data Recovery – Constructing readable information from files that may have been deleted, obfuscated, or otherwise protected from casual viewing. Windows-based browsers by design generally save their relevant information for anyone to see, so the real challenge becomes organizing the reams of readable data into something about which conclusions can be drawn.

String Search – Searching for specific "strings" (phrases) of words, such as "drugs", "deal", "stash" or "pusher" when performing a narcotics investigation.

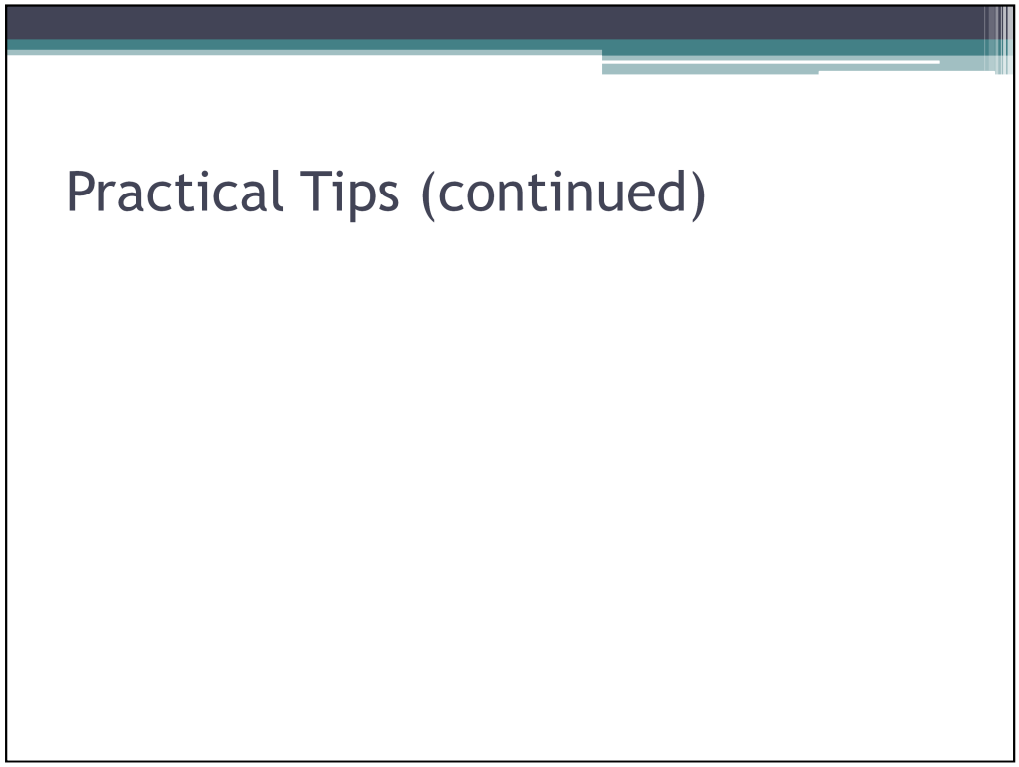
Reporting – Drawing conclusions from your investigation and presenting them to another party.



Something to keep in mind before we dive into the investigation: The person running the browser (we’ll call him or her our “subject” going forward) is innocent of any wrongdoing until you obtain evidence to the contrary. Management may ask you to conduct an investigation with a “gettum” attitude, hoping anything you can find will lead to their “subject” being fired, sued or locked away. Your investigation should always be about what the evidence proves, not what someone *hopes* it will prove, and mere presence of evidence doesn’t necessarily prove anything at all (except that some*one* was doing some*thing*.)

Another factor to consider before the investigation starts: you (along with your organization’s Management, Legal and Human Resources departments) need to determine how far you want to pursue the investigation. Is this just something for internal management purposes (rare) or may it turn into an eventual civil or even criminal proceeding? Or will the evidence be used in a disciplinary action or a termination? Either way, get input from your higher-ups, *especially Legal.*

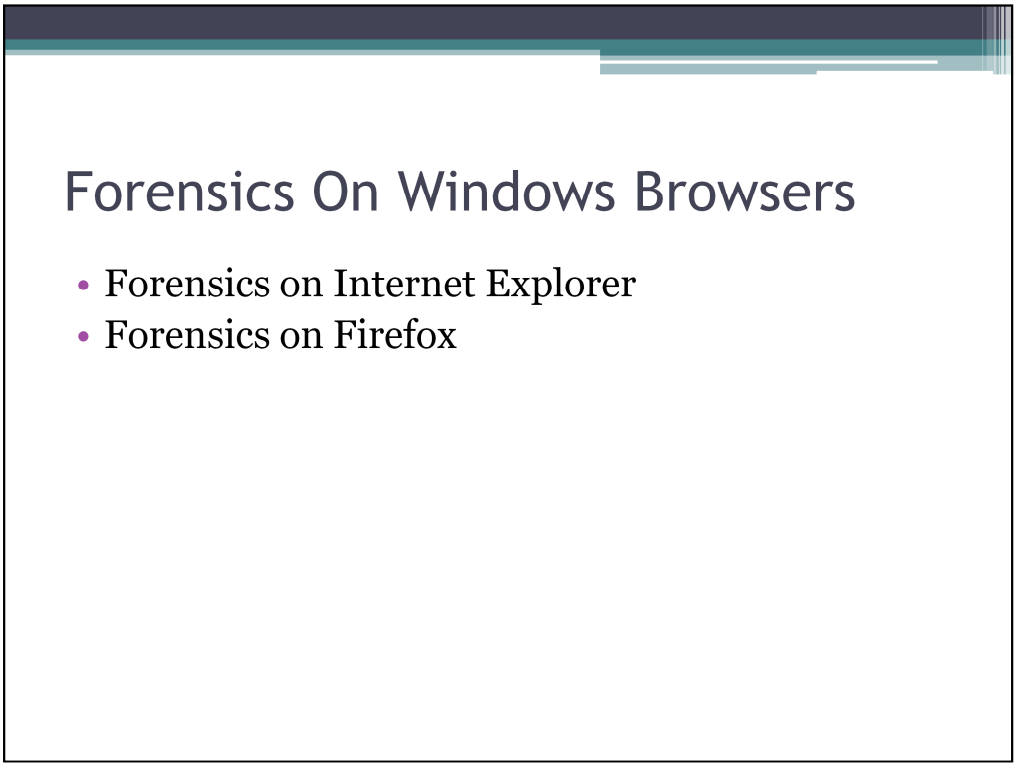
The most dangerous thing about forensics on a computer is that evidence on a running Windows computer follows the Schrödinger (Copenhagen interpretation) theory of quantum mechanics: by merely observing the evidence, you are changing it (or at least changing the MAC metadata.) Your "subject" (or your "subject’s” lawyer) can say that your evidence is invalid, because the changed MAC times indicate you altered the evidence after your "subject" used the computer. Instead, you should perform all of your analysis on an exact *copy* of the hard drive, and through mathematical processing you can prove that nothing was altered by your investigation.



So how do we get the information off the hard drive if we can't use Windows to view or copy it? We use Helix! Helix (http://www.e-fense.com/helix/) is a "live CD" (meaning you can drop it in the CD drive, boot the computer, and get a complete and functional operating system) that is specifically designed to never alter information on the host computer's hard drive. You can therefore use it to copy the entire contents of a "subject"'s PC to another hard drive, safe in the knowledge that the information will not be changed.

There's even a way to prove it – use a cryptographic hash to compare the two copies. The Message Digest 5 (MD5) "hashing" algorithm produces a 32-character fingerprint that is unique to a specific set of data; changing even one character of the data would result in an observably different fingerprint output, so it is infeasible with today's technology (although theoretically possible) to produce two different files that would create the same fingerprint. The MD5 fingerprint is widely accepted as valid in court cases. Another algorithm, Secure Hash Algorithm 1 (SHA-1) is more sophisticated, and it is even more improbable to produce two files with the same output. If you run an MD5 or SHA-1 analysis on your "subject"'s hard drive (both programs are built into Helix) before the copy process, then run one after your forensic examination is complete, you can show (by comparing the before-and-after numbers) that nothing was changed.

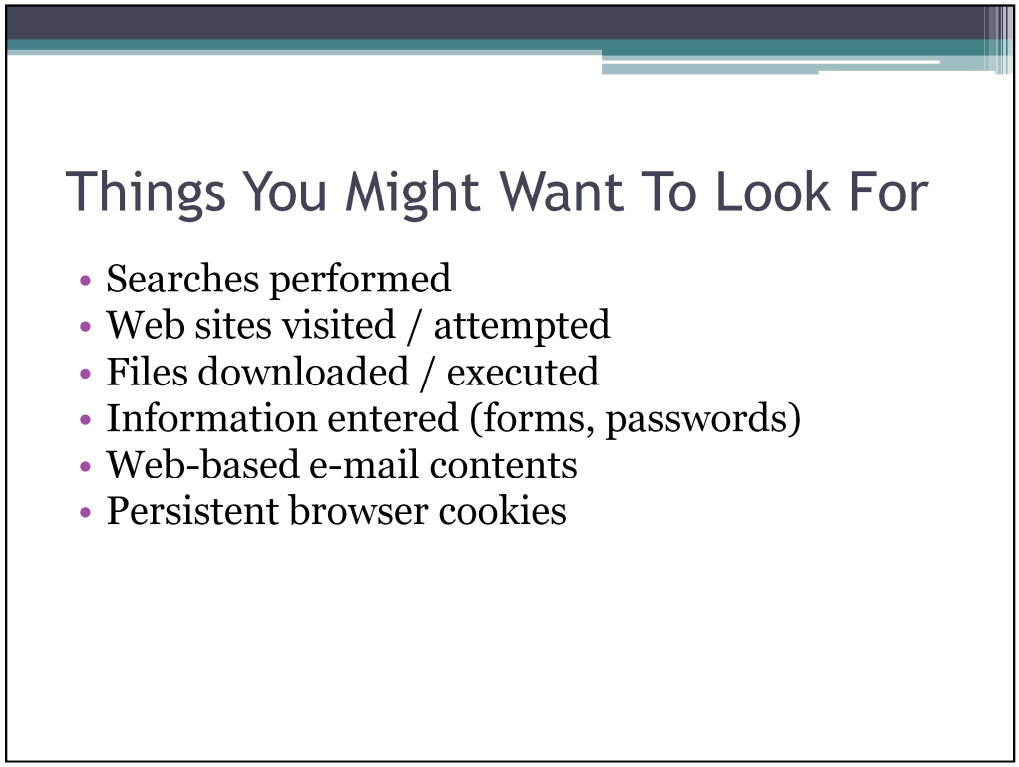
To further guard against accusations of tampering, use a separate workstation that is dedicated to forensics work, and has the proper tools (such as the ones included in Helix, and the ones we will review in this course) already installed. Use Helix to copy the hard drive contents to this workstation, and perform your full analysis there. MD5 / SHA-1 will prove that what you are analyzing is equal to the original information set.



This presentation takes you through retrieving evidence of web browsing activity on the most-utilized Windows-based browsers, Internet Explorer and Mozilla Firefox. We will discuss each of the browsers in turn, and how they each save and access files that can be retrieved for forensic evidence. We also include an Appendix to briefly discuss forensics of the Firefox browser on Unix-derivative systems.

While this course strives to not overwhelm the student with advanced technical concepts, it is expected that the student will be familiar with basic incident response and evidence gathering and preservation techniques. We will be utilizing publicly available freeware and shareware utilities to extract information from the computer of the person running the browser (we'll call him or her our "subject" going forward) and analyze it to determine what browser-related activity took place. For a complete grounding in forensics, there are multiple courses and learning opportunities on the Internet; future versions of this presentation will include a links page to some suggested sites.

One additional assumption: We assume that your \Windows directory (and related directories such as \Documents and Settings and \Program Files) is assigned to a drive labeled C:\, as is most common with Windows machines. If you have installed Windows to a different partition, please substitute its assigned drive letter for C:\ in our examples.



One more thing to consider before we actually start looking for information: What kind of information will allow us to determine what the subject was doing with his or her web browser? This will be useful in guiding our search because we are going to be sifting through a *lot* of information, and we'd like to narrow our focus as early as possible.

Searches performed will tell you where the subject was *thinking* of going. It also conveniently provides "keywords" for you to use later on when reviewing files kept by the system.

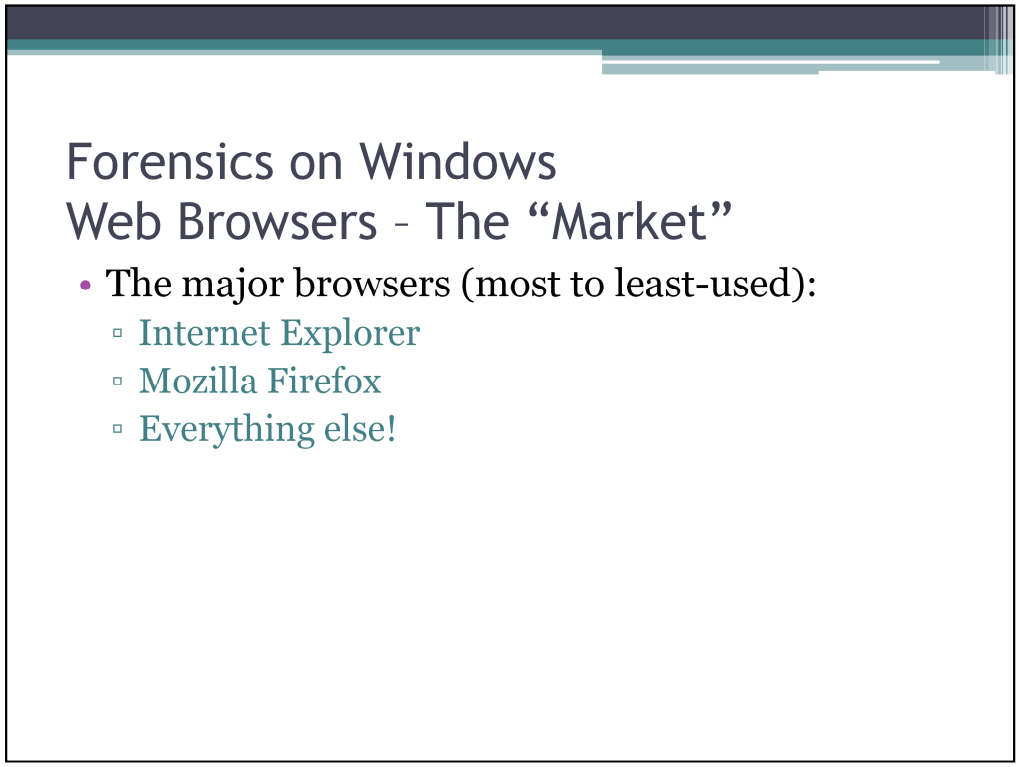
Web sites visited / attempted will comprise the bulk of your search. Remember that even seemingly innocuous sites can give you clues as to the subject's activities (such as a treasury clerk checking the weather in Barbados when you're investigating a financial fraud situation.)

Files downloaded / executed is the second-most-important piece of information, because if your subject is downloading unauthorized code onto your company's computers, s/he can cause a lot of damage, accidentally or intentionally.

Information entered (forms, passwords) can give you clues to unlock other locations your subject has visited, such as web-based e-mail sites, order forms, blog comment boxes, or even bank accounts. Note that you should definitely obtain the permission of your organization's management, Human Resources and Legal departments before utilizing password information, as now you're moving outside of the subject's web browser and onto a remote server.

Web-based e-mail contents don't necessarily need to be retrieved from the e-mail website; our browsers are nice enough to store the contents of read or composed emails on the hard drive whether we ask for it or not.

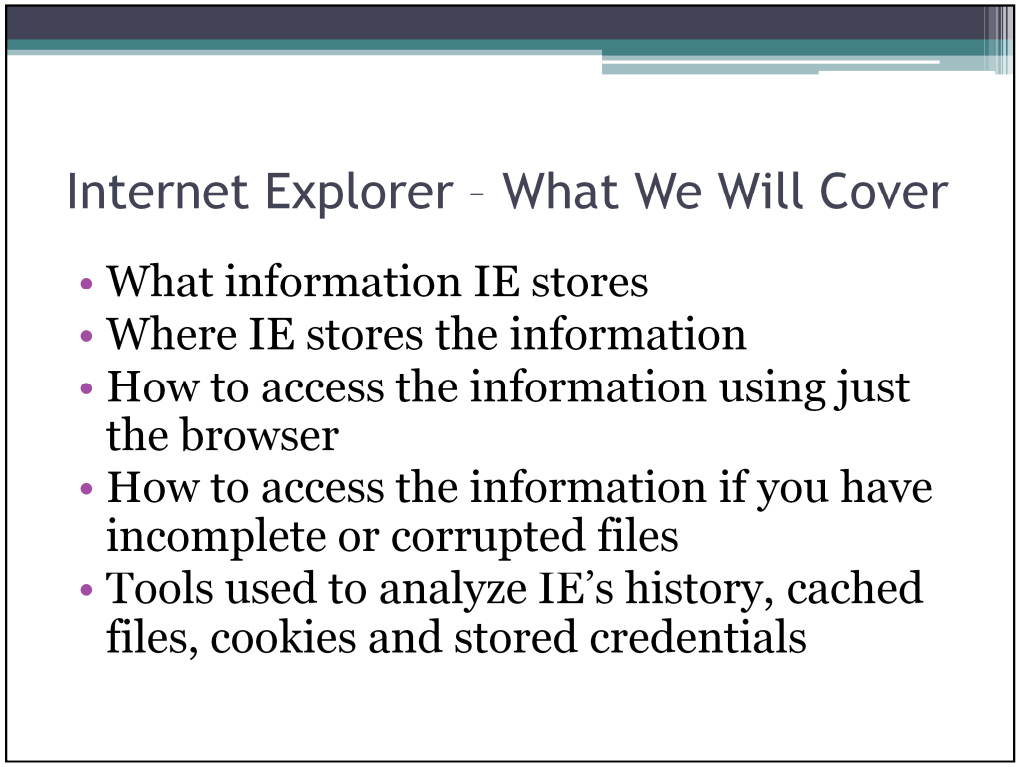
Finally, *persistent browser cookies* can reveal where a subject has gone, even if s/he has been careful enough to erase the browsing history.



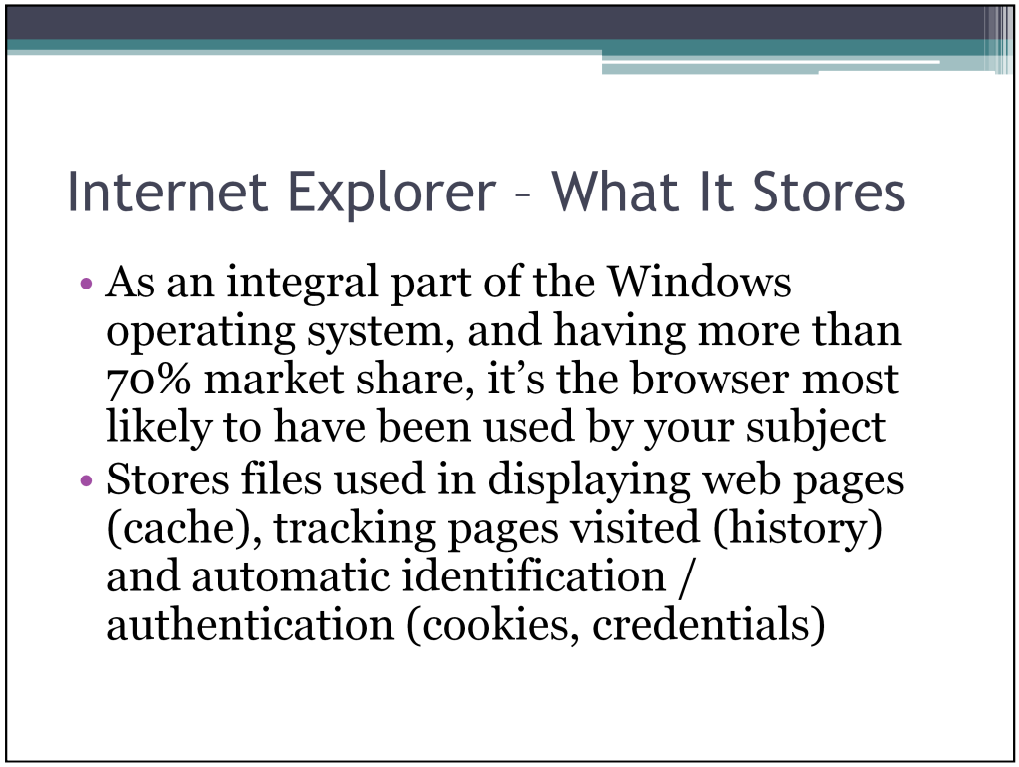
On to the tools of your subject's "trade." Not surprisingly, as it comes pre-installed with every copy of Windows, Microsoft's Internet Explorer still holds a commanding lead in browser "market share" (kind of a strange concept since most of them are free for the taking), followed very distantly by the much-ballyhooed Mozilla Foundation's Firefox.

According to the website Hitslink.com, the percent breakdown of Windows browser share at the time of this presentation's update (February 2010) is:

Internet Explorer	61.58
Mozilla Firefox	24.23
All other browsers	14.19



This page intentionally left blank.



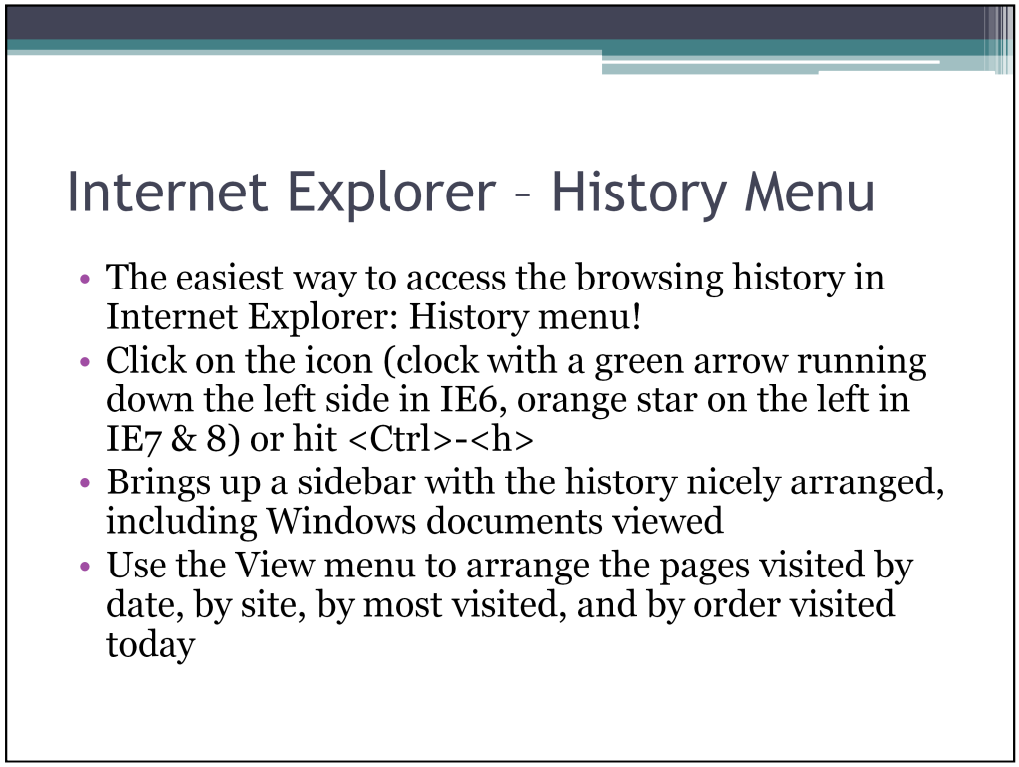
Famous turn-of-the-century thief Willie Sutton was once asked why he robbed banks; Mr. Sutton replied "Because that's where the money is." Following this same logic, it is statistically more likely that your subject will use Microsoft's Internet Explorer for his/her nefarious tasks than any other browser. It comes with the Windows operating system (ever since Windows 95), and most websites are tuned to work with it.

A browser will store certain files as you explore the web, in order to make your next web experience more enjoyable. This means the browser will save certain files to the hard drive as you view web pages, so if you ever decide to view those same web pages again:

• The page you view will retrieve its page code and embedded files (such as graphics) from the hard drive rather than the server, so the page loads faster (cache)

• You will be able to see a record of pages you've recently visited, rather than having to memorize a long web address in order to get back to a website (history)

• You won't have to sign in again at sites that require it, or specify your preferences again (cookies and credentials.) Of course, also note that cookies are used by the site you visit and other sites to track your web browsing, which is a privacy discussion all its own.



Clicking on the History icon (looks like a clock with a green arrow running down the left side counter-clockwise in IE6; click on the orange star on the left side of the screen in IE7 & 8 to access history, favorites, and RSS feeds) or hitting <Ctrl>-<h> brings up Internet Explorer's History menu, on the left side of the browsing window. You will be presented with multiple "views" you can utilize to see the sites you have visited (and Windows files you have opened.) The views include By Date, By Site, By Most Visited, and By Order Visited Today. In the By Site view, files are listed under My Computer.)

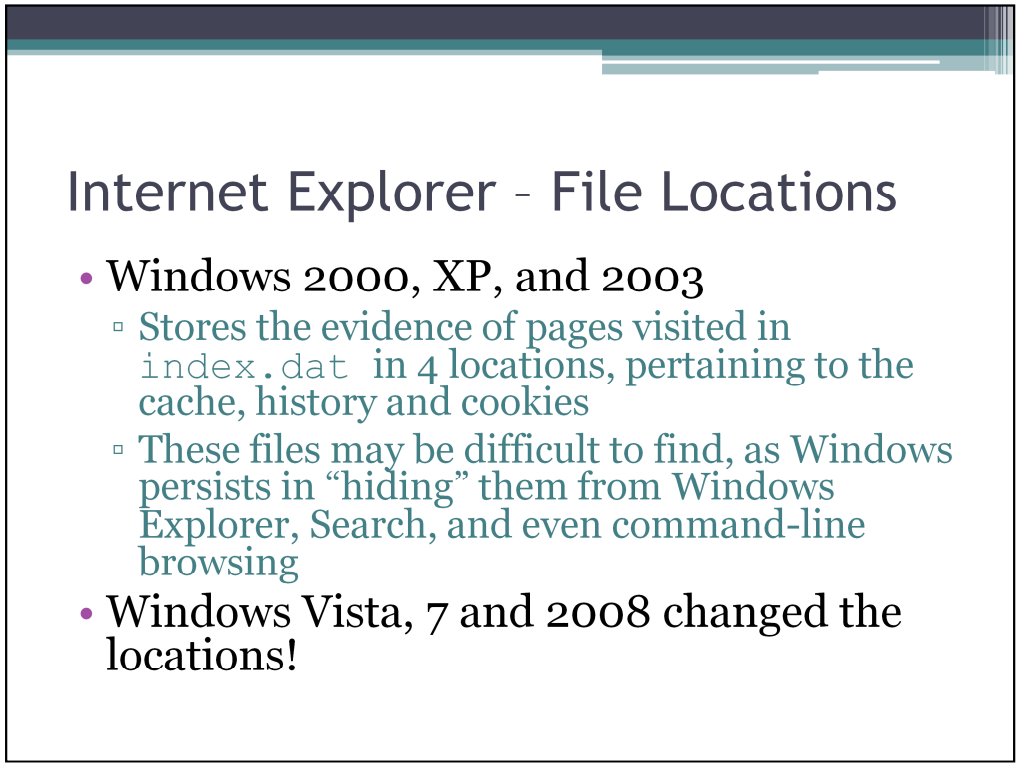
This method is being shown to you as a feature of Internet Explorer, but you should only use it as an absolute last resort to examine the files on a subject's computer. By viewing sites listed in the history, you are changing the browsing history, and thus the evidence of your subject's activities. *The three methods described below are far superior to viewing files with <Ctrl>-<h>, and you should use them whenever possible.*

There are three ways to properly view evidential information without altering it:

•On your forensics workstation, overwrite the history files with copies of your subject's system's history files, fire up Internet Explorer, and do the <Ctrl>-<h> key combination. This works in a pinch, but does not arrange the information in an easily exportable format. This method also has a dark side: you *could* pin the blame on someone else by copying history files (showing a history of browsing to all sorts of unsavory sites) to his / her computer. A more in-depth forensic examination will show if this has occurred.

•Another method is to create a VMWare image out of the extracted drive image and boot into that on your forensics workstation. You can then look around all you want using the standard operating system interface, confident in the knowledge that any changes you make will disappear the next time the image is restarted in VMWare. A freeware tool for converting the extracted image into VMWare format can be found in Bradley Schatz's dd2vmdk, located at http://www.bschatz.org/servlet/dd2vmdk/dd2vmdk.html. Note that this is an online tool and only shows you how to calculate the disk geometry necessary to convert the image, and then gives a list of Unix commands you can paste in to perform the actual process.

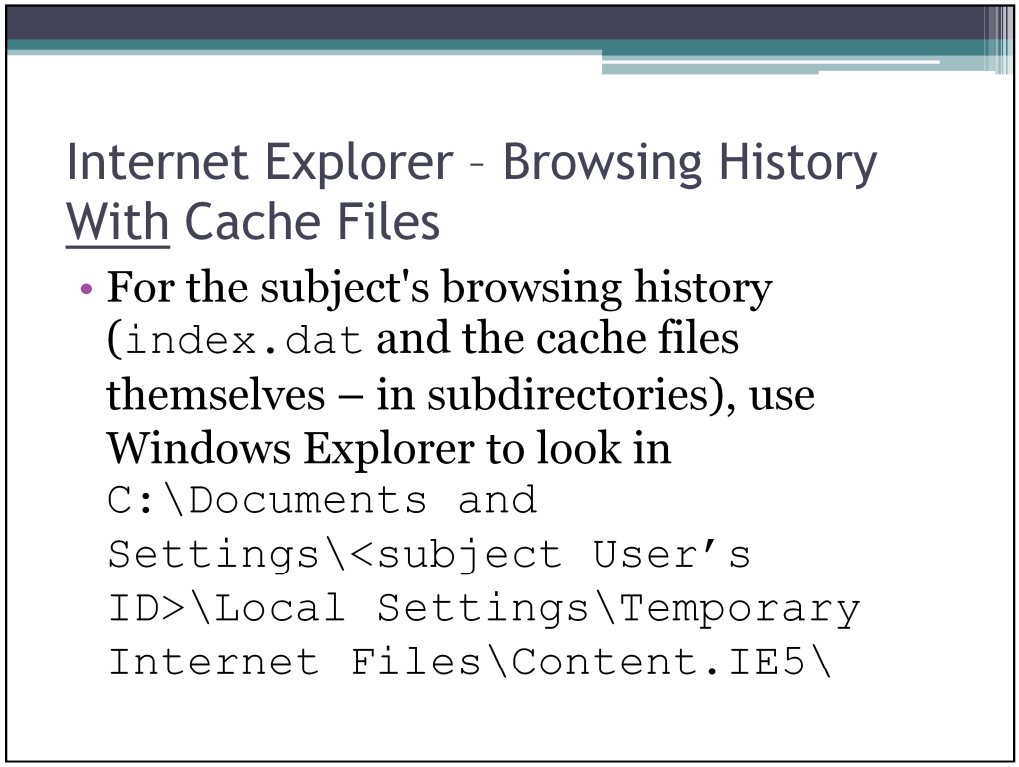
•The recommended method is to use another utility to view and parse the history separately from the browser (again, on a forensics workstation.)



The most recent versions of Windows (2000, XP, and 2003) store information about the pages viewed by the browser in a file called `index.dat`. One of the `index.dats`, in turn, contains information pointing to other files used in the browsing session. Windows has 3 types of `index.dat` files, for the cache, history and cookie files, respectively. Obviously, viewing all 3 types (and any companion files) will give us the best understanding of what browsing took place. Of course, Microsoft had to go and change the file and folder locations (but not the files and folders themselves) in Windows Vista, 7 and 2008. We'll get to the actual file locations / paths in the next few slides (so no looking ahead!)

Getting to these files may not be easy – Windows, trying to be "friendly" (and protect users from accidentally destroying operating system files), hides some of these files from Windows Explorer, the Search function, and even straight directory browsing from the command line. Some have even accused Microsoft of purposefully hiding these locations so most users would be blissfully unaware of the information being accumulated about them. The plausibility of this claim is left for the reader to ponder.

You can see *some* of the `index.dat` files in Windows Explorer (and see some of the other files in a browser or command prompt), but there's no way of getting them into a nice, neat output for analysis. For that task, we'll turn to our tools; but first, a look at "live" analysis of Internet Explorer records on a Windows machine.



In this case, we're accessing the `index.dat` file that contains the browsing history and links to the cache files, and then seeing where the cache files themselves are stored. *Sometimes*, you can simply navigate to `C:\Documents and Settings\<subject User's ID>\Local Settings\Temporary Internet Files\Content.IE5` in Windows Explorer. Other times, you can only get as far as the `Temporary Internet Files` folder and you'll see a bunch of cache files and no `index.dat` – it seems to vary from computer to computer. The most reliable ways to locate these files are as follows:

. Using your browser (Internet Explorer or Firefox) type `file:///C:/Documents and Settings/<subject User's ID>/Local Settings/Temporary Internet Files/Content.IE5` (yes, the slashes go forward here, and that's *three* slashes in front of the C), or

. Go to the command prompt, and use the `cd` command to navigate to `c:\Documents and Settings\<subject User's ID>\Local Settings\Temporary Internet Files\Content.IE5` (yes, you should use backslashes here.) Don't be confused if you do a `DIR` command at `Temporary Internet Files` and don't see the `Content.IE5` directory – it has the "System" attribute set which means it is hidden from the standard DIR command. You can use the DIR command with the /AS (view files with the "System" attribute) option to see it, or just use another `cd` command to get to it.

The `Content.IE5` directory, in addition to `index.dat`, also contains a number of subfolders (with apparently random names like 7Y5WH3RZ) that hold the actual cache files. The `index.dat` file maps the cache files (in the randomly-named directories) to the web pages it lists. So you can see (almost) exactly the pages your subject has visited.

Further complicating matters is that there exists another `index.dat` file of the same type in a different directory: `C:\Documents and Settings\<subject User's ID>\UserData`. What makes this one unique? It pertains to "housekeeping" processes performed periodically by the Windows operating system, such as Windows Update. As above, the `UserData` directory also contains subdirectories with random names, which are tied back to websites inside of the `index.dat` file. While it would be rare for evidence of your subject's browsing to be contained in here, it is a mark of thoroughness to analyze this file as well.

On Windows Vista, 7 and 2008, you must substitute `C:\Users\<subject User's ID>\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5` or `C:\Users\<subject User's ID>\AppData\Local\Microsoft\Windows\Temporary Internet Files\Low\Content.IE5` for the `Content.IE5` directory discussed above. The "Low" directory represents directories created by Vista, 7 and 2008 when IE version 7 & 8 is run in "protected" mode (which it does by default.) These "Low" directories are where Vista, 7 and 2008 permits IE7 & 8 to write files, so that potentially malicious code can be contained.

Similarly, for the `UserData` directory discussed above, in Vista, 7 and 2008 navigate to `C:\Users\<subject User's ID>\AppData\Local\Microsoft\Internet Explorer\UserData` or `C:\ Users\<subject User's ID>\AppData\Local\Microsoft\Internet Explorer\UserData\Low (`if your user has a domain roaming profile, substitute `Roaming` for `Local` in the path above.)

We will use tools to analyze these `index.dat` files and match their contents to the cache files.

Internet Explorer – Browsing History Without Cache Files

- For the subject's browsing history (`index.dat` without the the cache files), use a browser (NOT Windows Explorer) or command prompt to look in `C:\Documents and Settings\<subject User's ID>\Local Settings\History\History.IE5\`

As before, you can navigate to `C:\Documents and Settings\<subject User's ID>\Local Settings\History` using just Windows Explorer (this seems to work all the time). From here, you can see a number of subfolders, arranged by history date range (two weeks ago / last week / today), which are further subdivided into folders by site visited, and finally the shortcut files to the viewed pages – all nicely organized! This is useful for a preliminary overview of your subject's browsing, when you need to get an answer to management as quickly as possible, and is the reason why we would view this directory rather than the "Temporary Internet Files" directory on the previous slide. However, it doesn't give us as much information (and in as exportable a format) as `index.dat`, so we need to look even further.

Looking even further entails navigating to the "hidden" (from Windows Explorer) subdirectories under `History.`IE5. To do so, you need to use one of two methods:

. Using your browser (Internet Explorer or Firefox) type `file:///C:/Documents and Settings/<subject User's ID>/Local Settings/History/History.IE5` (yes, the slashes go forward here, and that's *three* slashes in front of the C), or

. Go to the command prompt, and use the `cd` command to navigate to `C:\Documents and Settings\<subject User's ID>\Local Settings\History\History.IE5` (yes, you should use backslashes here.) Also as before, the `History.IE5` directory is hidden from the `DIR` command, but visible with the DIR /AS command. The `index.dat` file in the `History.IE5` directory is similar to the `index.dat` file in the `Temporary Internet Files/Content.IE5` directory, but this time *without* links to the cache files. The "hidden" subdirectories are named as follows:

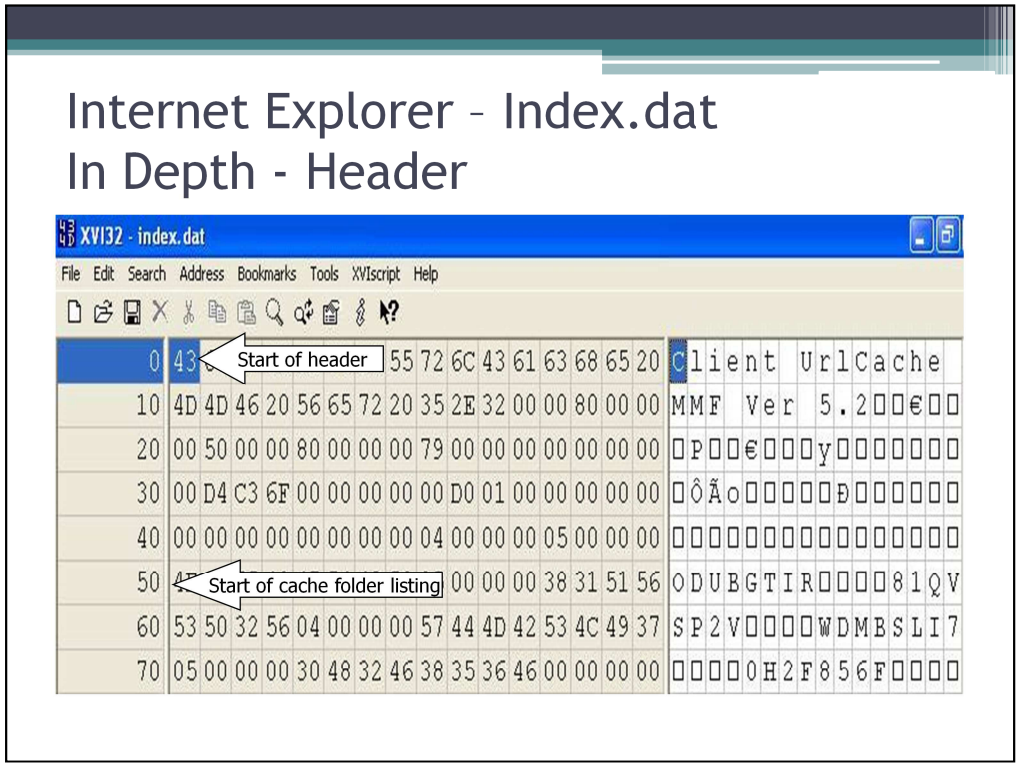
. Daily history: MSHist01(start)YYYYMMDD(end)YYYYMMDD

. Weekly history: MSHist01(start)YYYYMMDD(end)YYYYMMDD

For example, a directory with the browsing history from August 19, 2007 to August 26, 2007 would be MSHist012007081920070826. Each subdirectory contains its own `index.dat` file, which we can analyze using our tools. Note that the "parent" `index.dat` file (directly under `History.IE5`) is a "superset" of the `index.dats` in the subfolders (i.e. contains all of their information in one spot.)

On Windows Vista, 7 and 2008, you must substitute `C:\Users\<subject User's ID>\AppData\Local\Microsoft\Windows\History\History.IE5` or `C:\Users\<subject User's ID>\AppData\Local\Microsoft\Windows\History\History.IE5\Low` for the `History.IE5` directory discussed above.

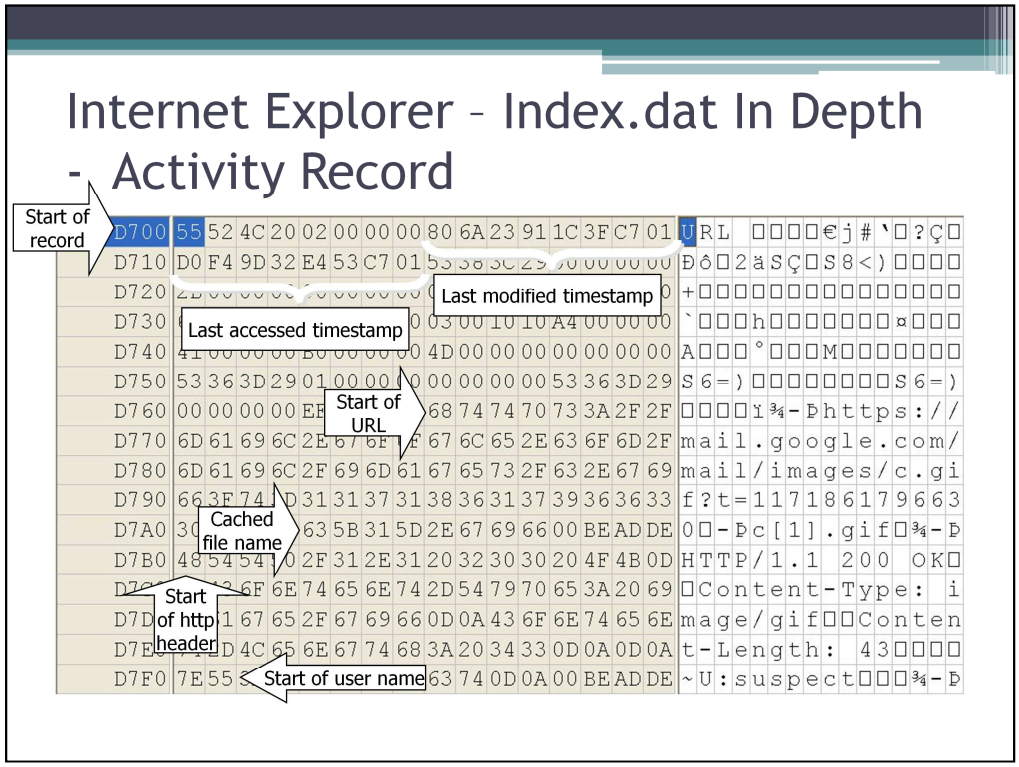
Now that we know more than we ever wanted to about the location of the files, let's take a look inside the `index.dat` file itself.



Why look at the details of `index.dat` if we already have programs that will parse it out for us? Unfortunately, you won't always be so lucky as to have an intact `index.dat` file to review: the file may have been purposely deleted, partially overwritten, or otherwise corrupted. When reviewing a forensic image, you may have to do a string search to locate the boundaries of the `index.dat` file, then "carve it out" of the data, and perhaps place it into another intact `index.dat` file, in order to be able to view it in one of the parsing programs. Finally, it's a good idea to know the basics of the file structure – you might wind up having to explain it to opposing counsel one day. :S

Since `index.dat` is a binary file, you can't just open it in Windows Notepad – you need a hexadecimal editor. Fortunately, one such editor, XVI32, is available for free from http://www.chmaas.handshake.de/delphi/freeware/xvi32/xvi32.htm. We'll use XVI32 to open an `index.dat` file from the `Temporary Internet Files` folder (so it will show both the history and point to local copies of cache files.)

When we open the file, we notice right away the header information at offset 0x00: "Client UrlCache MMF" and then a version number pertaining to the version of Internet Explorer in use. The next important set of information, always located at offset 0x50, is the listing of randomly-named folders in which the cache files are stored; there's a minimum of four, but there may be more depending on the size of the cache.



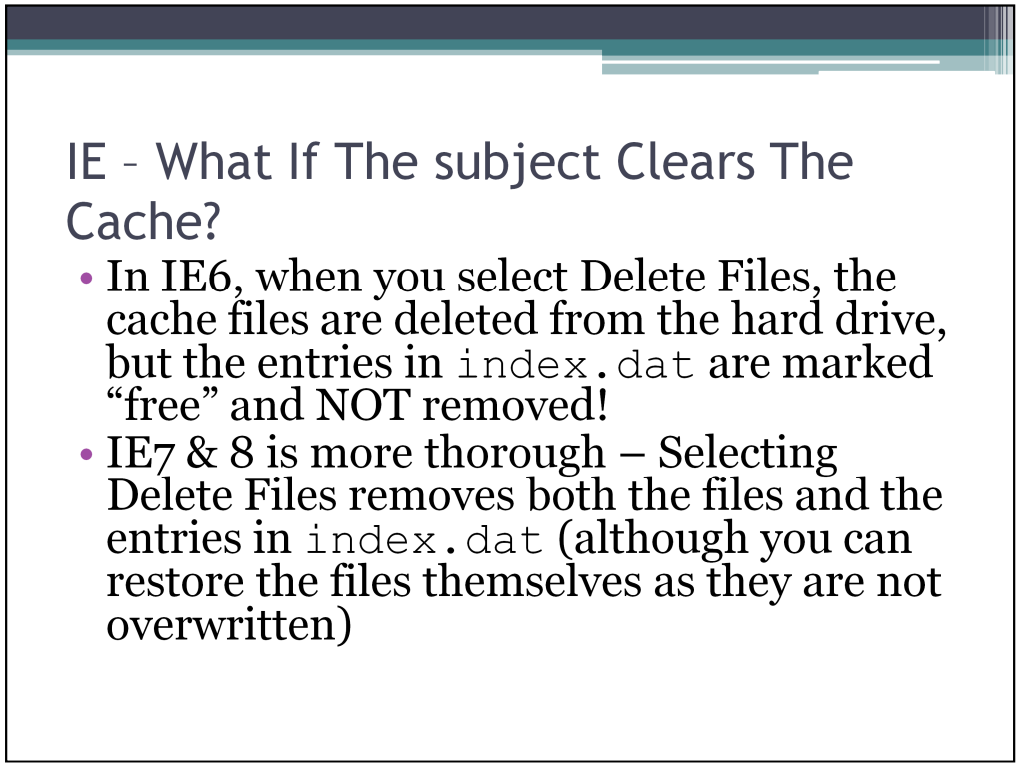
We now turn to the activity record itself, and in doing so will measure various fields as an offset of the beginning of the record rather than the beginning of the `index.dat` file. There are two "types" of web browsing activity in `index.dat`; when you click on a link to go to a website directly (usually labeled "URL", but sometimes "LEAK" in the record if the activity was deleted in `index.dat` but the related cache file could not be removed), or when code in the website automatically redirects you to another website (labeled "REDR".) In the example above, we see a URL record beginning at offset 0700. 0x08 bytes away from the record start we find the first of two timestamps in the record, the Last Modified timestamp. This is the time the file referenced by the record was last modified on the webserver (and, therefore, the last time the file was downloaded to the browser's cache.) In contrast, the next 0x08 bytes show the Last Accessed timestamp, or when the web page referenced by the record was last accessed by the browser (irrespective of whether it downloaded a new file or pulled it from the cache.) We'll get to the format of these timestamps and how to make sense of them in a moment.

Next on our information list, 0x68 bytes from the record start, is the URL that was visited by the browser. Since URLs vary in length, the rest of the information in the record will not be at a consistent offset from the beginning of the record, but once you have the location of the header and timestamps, you should be able to piece together the rest of the record. Following the URL is the filename of the locally cached file referred to by the record; a quick way to see where it starts is to back up into the URL and look for the same filename (most of the time, they will be identical.)

Another good piece of information stored by `index.dat` is the entire http header of the web page the record refers to. Finally, we see beginning with a U: the name of the user operating the browser at the time the web page was retrieved – VERY good information indeed!

Back to the timestamps: To get them into a format that you can use, you must convert the timestamp information no less than three times. First, the timestamp must be converted from hexadecimal to binary notation. Second, these binary numbers are in a format called "FILETIME" , which represents the number of 100-millisecond blocks of time since January 1, 1601 (no, we don't know how they chose that date either.) Once we convert FILETIME into a standard time and date format, we find that the time is in Coordinated Universal Time (UTC), and so a final conversion is required to translate the time into the time zone in which the subject's computer resides.

No, we are not going to manually take you through the 3-step conversion process (please feel free to take a moment and breathe a collective sigh of relief.) Instead, there exists a Windows-based utility called Hex Time Decoder, or DCode for short, available at http://www.digital-detective.co.uk/freetools/decode.asp. Run the utility, put in the hex code from the record, and you should have the timestamps in UTC format; add or subtract hours to convert it to the subject computer's local time.

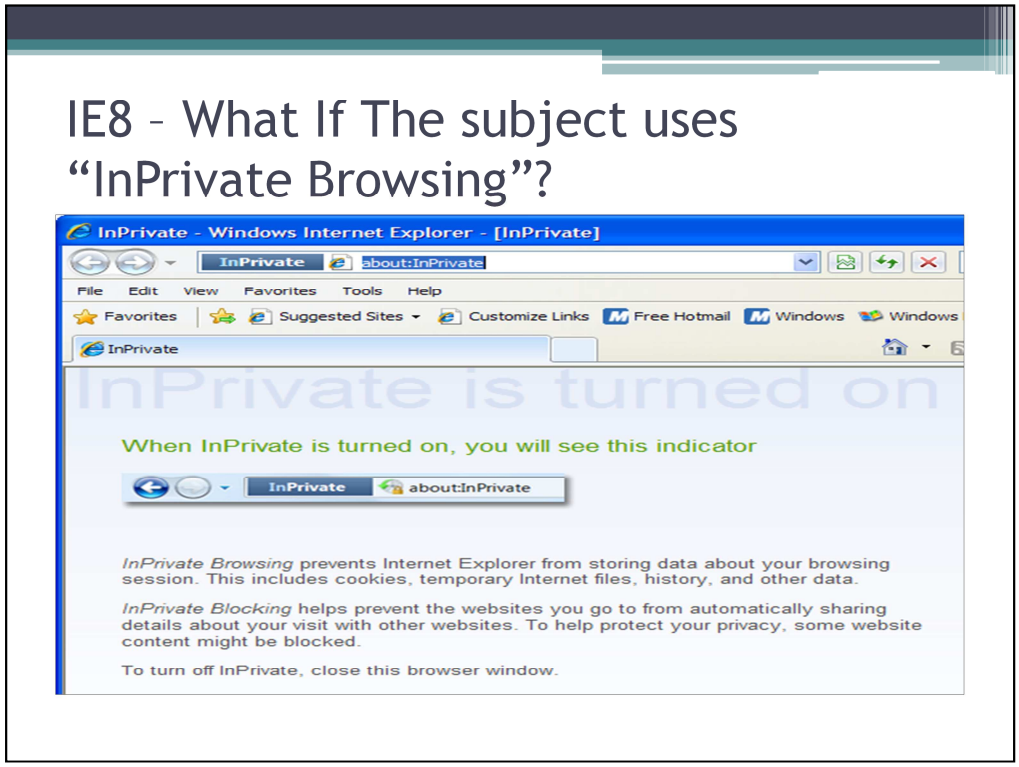


Sometimes, a user, desiring to conceal his or her browsing activity, will use Internet Explorer's standard built-in tool to clear the cache files. This works… some of the time.

In Internet Explorer 6, if you go to Tools / Internet Options / Temporary Internet Files and select Delete Files, you may think you have cleanly wiped up evidence of web pages you have visited. However, this "deletion" is incomplete. The Temporary Internet Files are deleted as you might expect. However, within the `index.dat` file, the entries for the deleted files are marked "free" but are not removed from the file, so a forensic investigator would still see the URLs for the sites they came from. Eventually, the entries are overwritten as Internet Explorer needs the space or does periodic housekeeping (compares files in the cache with the `index.dat` list) but the evidence does stay around for at least a while after you hit "Delete."

One security improvement Microsoft made in Internet Explorer 7 & 8 is that entries in the `index.dat` file is now overwritten when a user goes to Tools / Options / Delete Browsing History / Temporary Internet Files and selects Delete Files.

Whether the `index.dat` information remains intact or is overwritten, the files themselves are, as always, marked "free" and do not disappear until they are overwritten. While a cache file itself is not as valuable as a cache file with a browsing record to tie it to, sometimes the file's purpose can be self-evident (such as a file named "HackingTools.exe".)



Internet Explorer 8 (at the time of this writing, in Beta version 2) features a setting called "InPrivate Browsing" which humorous types in the media have redubbed "Porn Mode" due to the potential for users to access prurient content without evidence thereof being recorded. You can start InPrivate Browsing by opening a new tab and selecting Browse with InPrivate or selecting it from the Safety button on the top right corner of the browser window, or using the key combination <Ctrl>-<Shift>-<P>. As seen on the slide graphic, a cute little "InPrivate" logo appears to the left of the address bar while in this mode. According Microsoft's web site, "InPrivate Browsing in Internet Explorer 8 helps prevent your browsing history, temporary Internet files, form data, cookies, and usernames and passwords from being retained by the browser, leaving no evidence of your browsing or search history."

To paraphrase Daniel Faraday on "Lost": That's…… not entirely true.

InPrivate does make the forensic examiner's job more difficult by not recording items such as typed addresses, visited links, and forms, queries and passwords entered, including not recording the "host records" (URLS) in `index.dat`. It also deletes the contents of Temporary Internet Files when the "subject"exits the browsing session. *However*, items (such as the cached filename and page header information) are still dutifully written to `index.dat`, making it still possible for an investigator to infer where the "subject" has been surfing. More importantly, as with IE7 & 8's clearing of the cache, the files themselves are, once again, merely marked "free" and do not actually disappear until they are overwritten. An enterprising investigator can then retrieve the files using one of the many publicly available forensic tools.

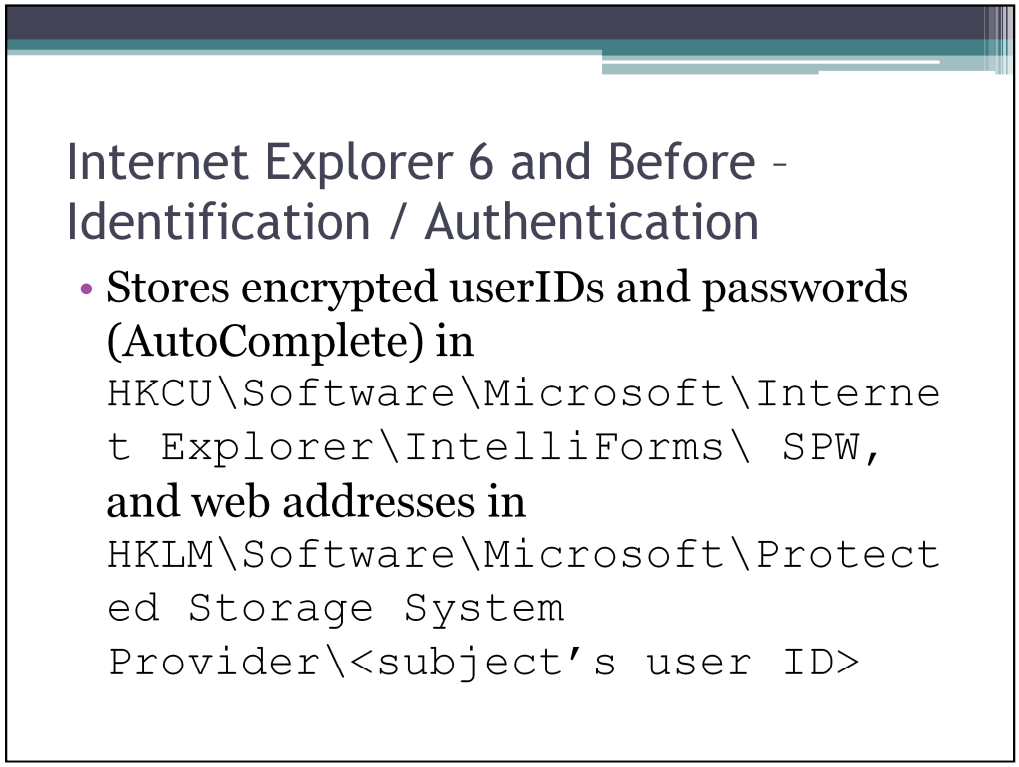
Internet Explorer – Cookies

- For cookies saved on the subject's hard drive (individual cookie text files), use Windows Explorer to look in `C:\Documents and Settings\<subject User's ID>\Cookies\`

Internet Explorer stores its cookies in individual text files in the format of `<subject User's ID>@domain-name-without-top-level-domain.txt`. For example, a cookie for visiting Yahoo.com would be in the form of `subject@yahoo.txt`, without the .com top level domain. We will explore (pardon the pun) these files using tools. You can also directly view the cookie files themselves, but the tools make the information much easier to read and organize. The Cookies folder has its own resident `index.dat` file, linking the cookies to the sites the subject visited, but as each cookie text file also contains a link to its visited site, we will conduct our analysis there.

On Windows Vista, you must substitute `C:\Users\<subject User's ID>\Roaming\Microsoft\Windows\Cookies` or `C:\Users\<subject User's ID>\Roaming\Microsoft\Windows\Cookies\Low` for the `Cookies` directory discussed above.

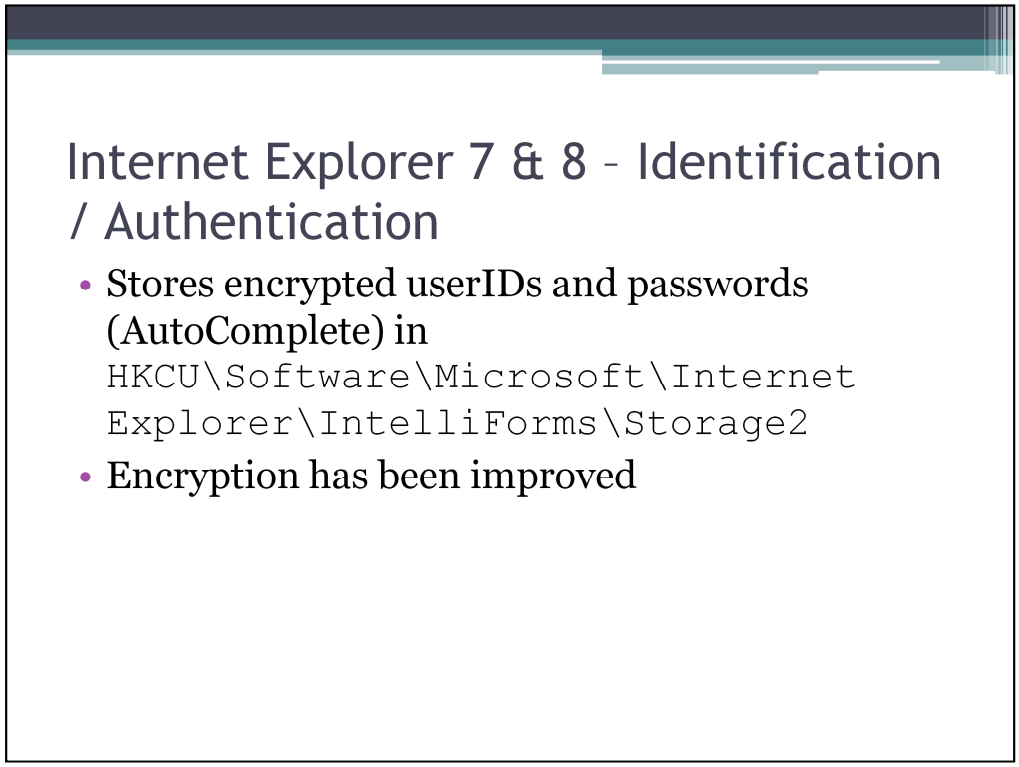
It does seem odd that the file index is directly accessible through Windows Explorer for cookies and not the two other types of files. Windows is like that. ☺



One more location where Internet Explorer stores information is the AutoComplete (Identification / Authentication.) As stated previously, one of the things you might want to look for is entered information, such as web forms, blog comment boxes, or even website userIDs and passwords. Internet Explorer's AutoComplete feature comes as a double-edged sword, permitting the user to not have to remember multitudes of identifying and authenticating information, but at the same time keeping that information for a forensic examiner to discover.

For Internet Explorer versions 4 through 6, the encrypted AutoComplete user names and passwords are stored in the user's Windows Registry under `HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\IntelliForms\SPW`. Web addresses, form fields, and other entered information is stored in `HKEY_LOCAL_MACHINE\Software\Microsoft\Protected Storage System Provider\<subject's user ID>`. This encryption (Triple application of the Data Encryption Standard, or Triple DES) isn't all that good anymore, and can be broken with the right tools.

You can also find a listing of the most recently typed web addresses (Uniform Resource Locators, or URLs) under HKEY_USERS/Default/Software/Microsoft/Internet Explorer/TypedURLs/ or HKEY_CURRENT_USER/Software/Microsoft/Internet Explorer/TypedURLs. This listing is not encrypted, so you can directly view the visited URLs. Also included in this listing are non-web addresses such file:/// etc.



Of course, Microsoft had to go and change things for version 7 & 8 of Internet Explorer. Encrypted userIDs, passwords and other form field information is *now* stored in `HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\IntelliForms\Storage2`. This presents a problem for recovery utilities, as they are still hardcoded to look in the Registry locations for IE 4-6; additionally, the encryption has been improved, so it is harder to break. IE7 & 8's encryption uses the Secure Hashing Algorithm (SHA) to store hashes of the encryption key, which is the URL of the web page that contains the specific form field.

One thing that has not changed is that you can still find a listing of the most recently typed web addresses (Uniform Resource Locators, or URLs) under `HKEY_USERS/Default/Software/Microsoft/Internet Explorer/TypedURLs/ or HKEY_CURRENT_USER/Software/Microsoft/Internet Explorer/TypedURLs.` As before, this listing is not encrypted.

As stated previously, you can view the files saved by Internet Explorer using the Windows Explorer interface. However, while the files are nicely organized, that very organization makes them difficult to see all at once, or to sort them in any other order than date. Additionally, the output is difficult to export to another document (other than taking a Print Screen) for reporting or further analysis. Reading the `index.dat` directly is also complicated, as we have seen – it can be done, but it's not an effort you want to regularly undertake.

Enter Mandiant's Web Historian. This free program parses the otherwise-difficult-to-read `index.dat` file and outputs it to a friendly format, such as comma-separated-value (CSV) or even Microsoft Excel. It also shows additional information not available in the Windows Explorer view. You can get Web Historian from the Mandiant website at http://www.mandiant.com/webhistorian.htm.

Another benefit of Web Historian is that it will also analyze the browsing histories of Mozilla, Firefox, (the now-abandoned) Netscape Navigator, Opera and Safari. One disappointing limitation: Web Historian, as of this writing, will not run on Windows Vista.

Mandiant Web Historian – Running

- When you run the program, you are presented with two ways of obtaining an `index.dat` file
- Note that only certain approaches work for certain files, and using the wrong approach may lock the Web Historian program!

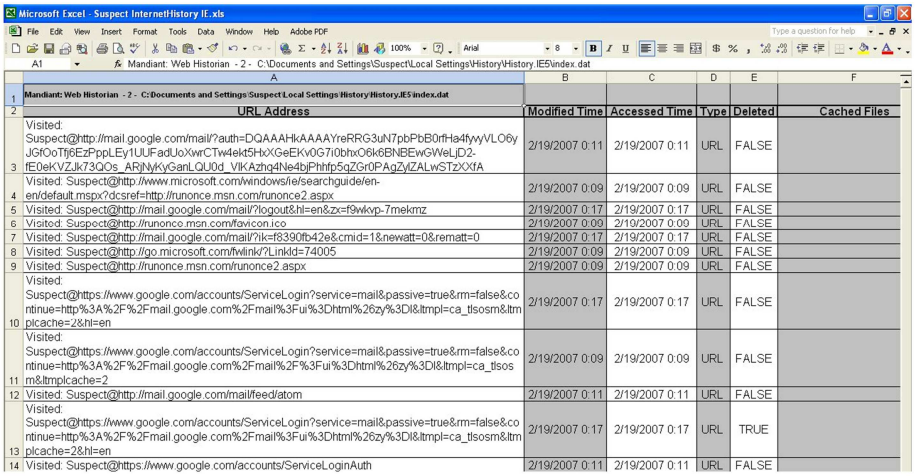
Web Historian has two approaches to obtaining an index.dat file. Be careful when choosing an approach, because not all approaches work for each type of index.dat file. You may even lock the Web Historian program, which certainly doesn't help your investigation.

The first method allows you to browse for an `index.dat` file use the File Open interface. This method only works for `index.dat` files that aren't in their default locations, such as a file you've copied from another computer. As noted before, you can't Explorer-browse to the `index.dat` file in the default `Temporary Internet Files` and `History` directories, and Web Historian doesn't work on Cookie files.

The second method allows you to specify a directory to search for an `index.dat` file. The search will include subdirectories of the folder you specify. This method may lock the program searching for multiple `index.dat` files, so try to specify only one directory at a time.

Select your file or directory, then go get a soda – it's going to take a few minutes to process the file, especially if your subject has done a lot of browsing.

Mandiant Web Historian – History Report



Microsoft Excel - Suspect InternetHistory IE.xls

Mandiant: Web Historian - 2 - C:\Documents and Settings\Suspect\Local Settings\History\History.IE5\index.dat

	URL Address	Modified Time	Accessed Time	Type	Deleted	Cached Files
3	Visited: Suspect@http://mail.google.com/mail/?auth=DQAAAHkAAAAYreRRG3uN7pbPbB0rfHa4fywVLO6yJGfOoTfj6EzPppLEy1UUFadUoXwrCTw4ekt5HxXGeEKv0G7i0bhxO6k6BNBEwGWeLjD2-fE0eKVZJk73QOs_ARjNyKyGanLQU0d_VlKAzhq4Ne4bjPhhfp5qZGr0PAgZylZALwSTzXXfA	2/19/2007 0:11	2/19/2007 0:11	URL	FALSE	
4	Visited: Suspect@http://www.microsoft.com/windows/ie/searchguide/en-en/default.mspx?dcsref=http://runonce.msn.com/runonce2.aspx	2/19/2007 0:09	2/19/2007 0:09	URL	FALSE	
5	Visited: Suspect@http://mail.google.com/mail/?logout&hl=en&zx=f9wkvp-7mekmz	2/19/2007 0:17	2/19/2007 0:17	URL	FALSE	
6	Visited: Suspect@http://runonce.msn.com/favicon.ico	2/19/2007 0:09	2/19/2007 0:09	URL	FALSE	
7	Visited: Suspect@http://mail.google.com/mail/?ik=f8390fb42e&cmid=1&newatt=0&rematt=0	2/19/2007 0:17	2/19/2007 0:17	URL	FALSE	
8	Visited: Suspect@http://go.microsoft.com/fwlink/?LinkId=74005	2/19/2007 0:09	2/19/2007 0:09	URL	FALSE	
9	Visited: Suspect@http://runonce.msn.com/runonce2.aspx	2/19/2007 0:09	2/19/2007 0:09	URL	FALSE	
10	Visited: Suspect@https://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&continue=http%3A%2F%2Fmail.google.com%2Fmail%3Fui%3Dhtml%26zy%3Dl&ltmpl=ca_tlsosm&ltmplcache=2&hl=en	2/19/2007 0:17	2/19/2007 0:17	URL	FALSE	
11	Visited: Suspect@https://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&continue=http%3A%2F%2Fmail.google.com%2Fmail%2F%3Fui%3Dhtml%26zy%3Dl&ltmpl=ca_tlsosm&ltmplcache=2	2/19/2007 0:09	2/19/2007 0:09	URL	FALSE	
12	Visited: Suspect@http://mail.google.com/mail/feed/atom	2/19/2007 0:11	2/19/2007 0:11	URL	FALSE	
13	Visited: Suspect@https://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&continue=http%3A%2F%2Fmail.google.com%2Fmail%3Fui%3Dhtml%26zy%3Dl&ltmpl=ca_tlsosm&ltmplcache=2&hl=en	2/19/2007 0:17	2/19/2007 0:17	URL	TRUE	
14	Visited: Suspect@https://www.google.com/accounts/ServiceLoginAuth	2/19/2007 0:11	2/19/2007 0:11	URL	FALSE	

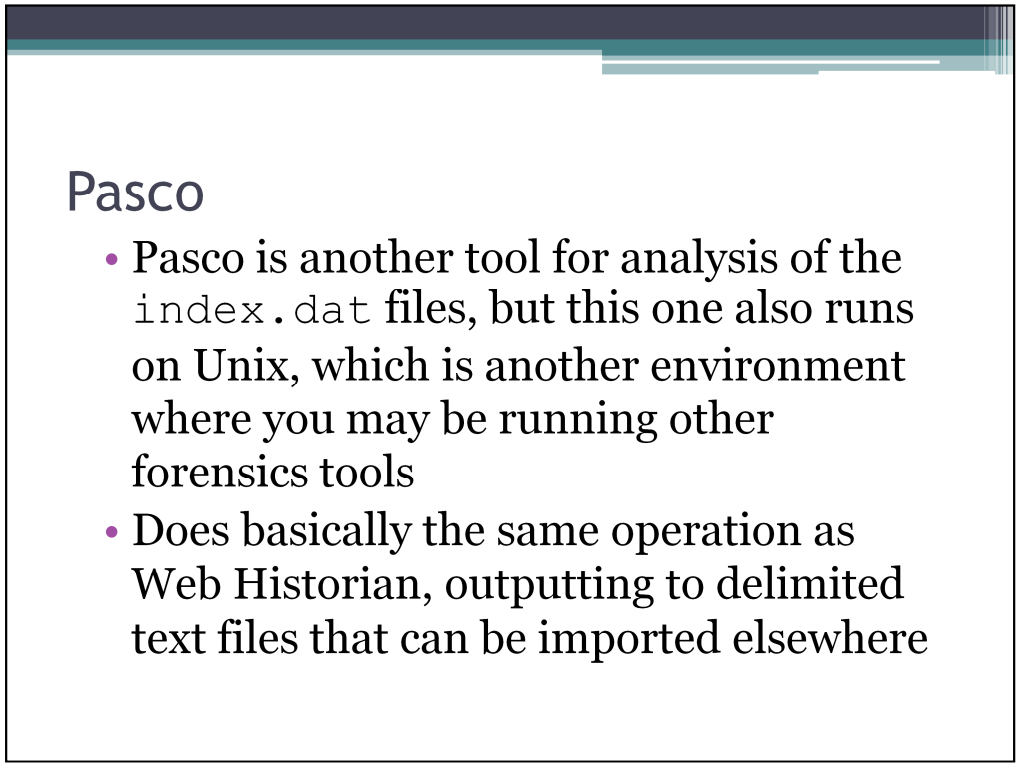
Web Historian has several output options, including Microsoft Excel and Excel-compatible (comma-or-tab-delimited), and .htm (web page.) Web Historian outputs the Excel-compatible formats using Extensible Markup Language (XML.) You should therefore choose the Excel-compatible option only if you have Office XP or newer, or OpenOffice 2, as only they can cleanly import the information.

The column headers include the Uniform Resource Location (URL) address, (last) modified time(stamp, in UTC), (last) accessed time(stamp, in UTC), type (URL or REDIR), deleted status (yes or no), the name of the cached file itself (complete with path info), and an output of the HyperText Transport Protocol (HTTP) header returned by the website called by the URL.

The most useful columns to sort by are URL and HTTP header, because that will sort the rest of the information by website visited. You can also utilize the search features of your favorite database program to query for records by file type, so you only see the specific web file types you are looking for.

To see what your subject saw when browsing, your best bet is to look under the Cached Files column (which only has content, naturally, when you are examining the `index.dat` from the `\Temporary Internet Files\Content.IE5` folder), then use Explorer to open and view the referenced files. Remember, you should be viewing a copy of the files on a dedicated forensics workstation!

The Modified and Accessed Time columns can give you a timeline of the web page accesses, which you can compare to other records (building access, network login) to establish who was doing the browsing.



Previous slides have told you to perform your analysis from files copied to a forensics workstation. Most of the time, that workstation will be Windows-based, and it's required to run Mandiant's Web Historian. However, some of you may be more comfortable with the *nix environment (Linux, BSD, Solaris, etc.) Pasco is a free command-line tool that largely performs the same functions as Web Historian, with the added benefit that it runs both on Windows and *nix environments.

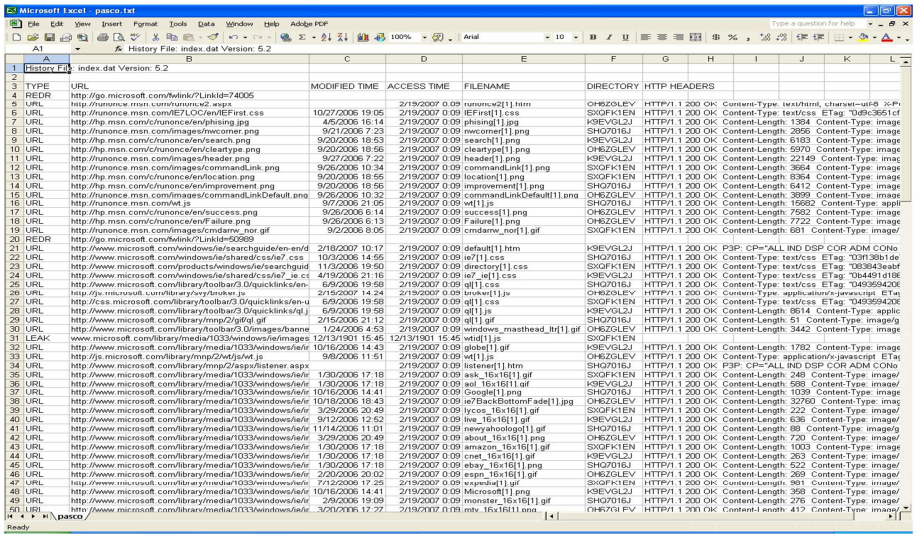
To use Pasco, type on the command line:

```
pasco <options> (filename)
```

The possible options are -d (undelete activity records) and -t (column delimiter – defaults to tab.) Use the > symbol after the filename to "redirect" output into a file (such as a CSV.) It's always a good idea to run Pasco twice, once with the –d option and once without, and compare the results so you can see which records were deleted.

Speaking of forensics tools, Pasco and other tools (Galleta for cookie analysis, and Rifiuti for looking through the Windows recycle bin) can be found at the ODESSA (Open Digital Evidence Search and Seizure Architecture) website: http://odessa.sourceforge.net. You can also download them from the author's company, FoundStone, but the ODESSA files will have the most recent updates.

Pasco - History with Cache

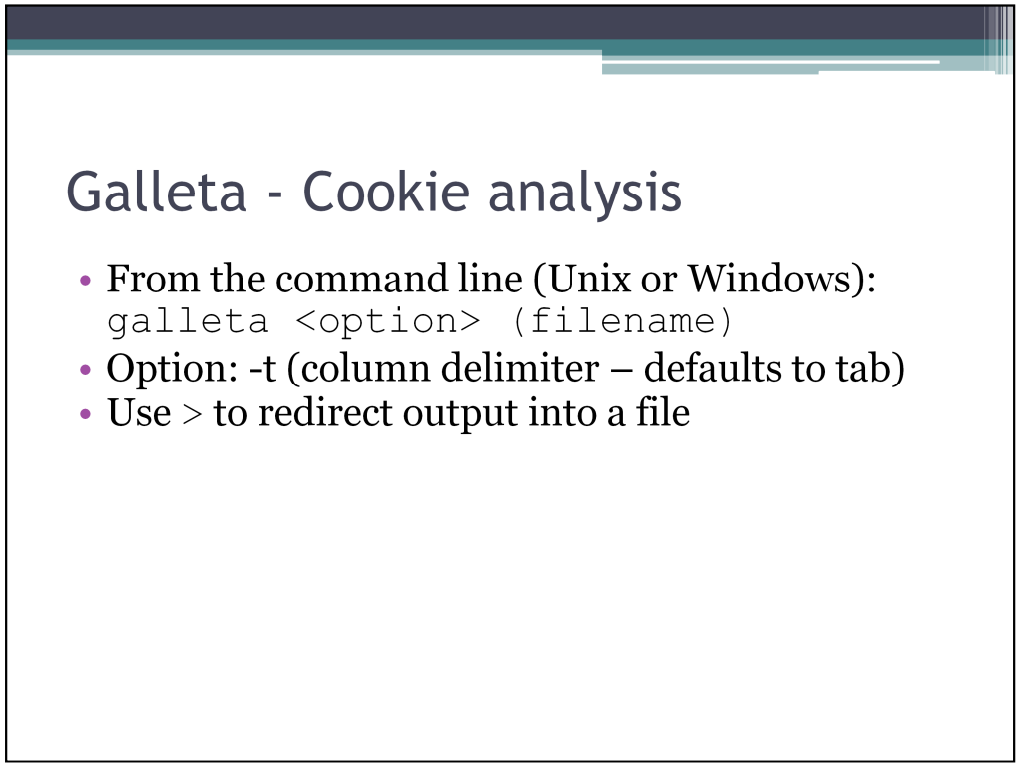


The column headers output by Pasco are similar to Mandiant's Web Historian: Type (URL or REDR), URL address, (last) modified time(stamp, in UTC), (last) accessed time(stamp, in UTC), the name of the cached file itself, the directory in which it is contained, and an output of the HTTP header returned by the website called by the URL.

Look under the Filename column to get a list of the files loaded by your subject's browser. Since the 3-digit Windows filetype extension (.doc, .xls etc.) is attached to the filename, you should import this information into a database so you can search for specific kinds of files. Remember, you should be viewing a copy of the files on a dedicated forensics workstation!

To sort the information by websites visited, use the URL and HTTP column headers. Compare the Modified and Accessed Time columns to other records (building access, network login), which can provide further evidence to tie the subject to the browsing.

To ensure successful execution, it's a good idea to copy the `index.dat` you wish to parse to the same directory as the Pasco executable file.



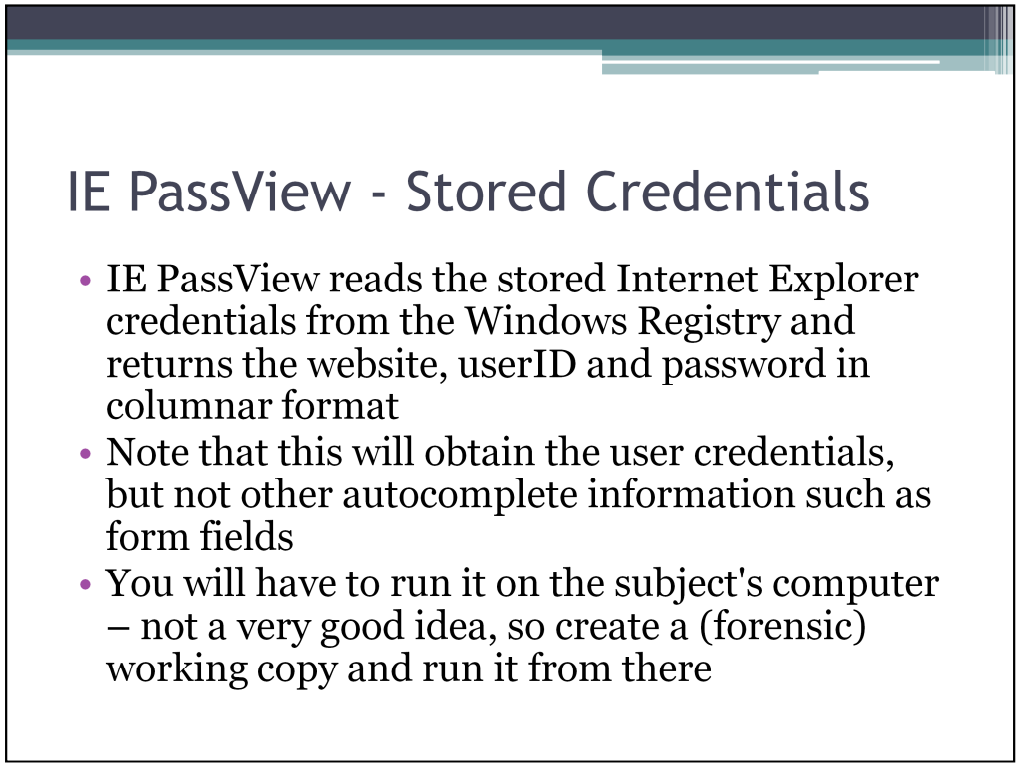
Galleta is a free program from the ODESSA project that can be used to analyze Internet Explorer cookies. Like Pasco, it runs from the command line on both Windows and *nix environments.

To use Galleta, type on the command line (Unix or Windows):
`galleta <option> (filename)` – but *this* time we ignore the index.dat file and instead concentrate on the individual .txt files. Kind of a pain, but at least it formats the output into columns for easier viewing.

There's only one possible option: -t, for the column delimiter of the output (defaults to tab.) As with Pasco, you can use the > symbol after the filename to "redirect" output into a file (such as a tab-delimited file that can then be read into Microsoft Excel.) To speed up and simplify things, you can use a script to call the Galleta executable, read each .txt file in turn, and use the double chevron (>>) to append the results to one centralized file.

Why are we ignoring the `index.dat` file this time? The cookie `index.dat` file has a similar structure to the other types (including the URL / REDIR record heading and the header information at 0x68 after the record start) but the Galleta authors choose to look directly at the text files (which contain the visited-site information), perhaps to save themselves a headache or two. ☺

To ensure successful execution, it's a good idea to copy the cookie text files you wish to parse to the same directory as the Galleta executable file.



NirSoft's IE PassView (IEPV) is an application that will dig those credentials that your subject has been using to access websites out of Internet Explorer. The website being accessed, the userID and password used for the access are displayed in a nice columnar format that can be exported to a text file or HTML report. Although these items are encrypted, IEPV can decrypt the information. However, the requirement is that you MUST be logged in as your subject user in order to reveal the passwords. This creates a two-faceted problem, as you really shouldn't be conducting a forensic investigation directly on the subject's computer, and you'll need the subject's userID and password to login. Your best bet is to make a "throwaway" (albeit forensically sound) copy of the hard drive, put it in another machine, use a Windows password cracker to get the subject's login password (or use a tool to clear the password), and then run IEPV on that machine. Note that such a solution may not be feasible if the subject utilizes strong (multi-factor) authentication, or chooses such a good password that it is not crackable in a reasonable amount of time (at that point, the clear password option is the best way to go.) You can find IEPV at http://www.nirsoft.net/utils/internet_explorer_password.html. Your virus scanner may identify the executable installer as matching the pattern of "DR/PSW.NetPass.E.3"; no, there's no virus in there.

Another limitation of IEPV is that it only obtains user credentials, not other auto complete information (such as user searches, information filled in for subscription forms, etc.)

A final reminder: You should only utilize your subject's discovered userIDs and passwords to access machines on your organization's network. The minute you go to a website outside of your organization, you need a properly executed search warrant, *no exceptions.*

Internet Explorer – Review

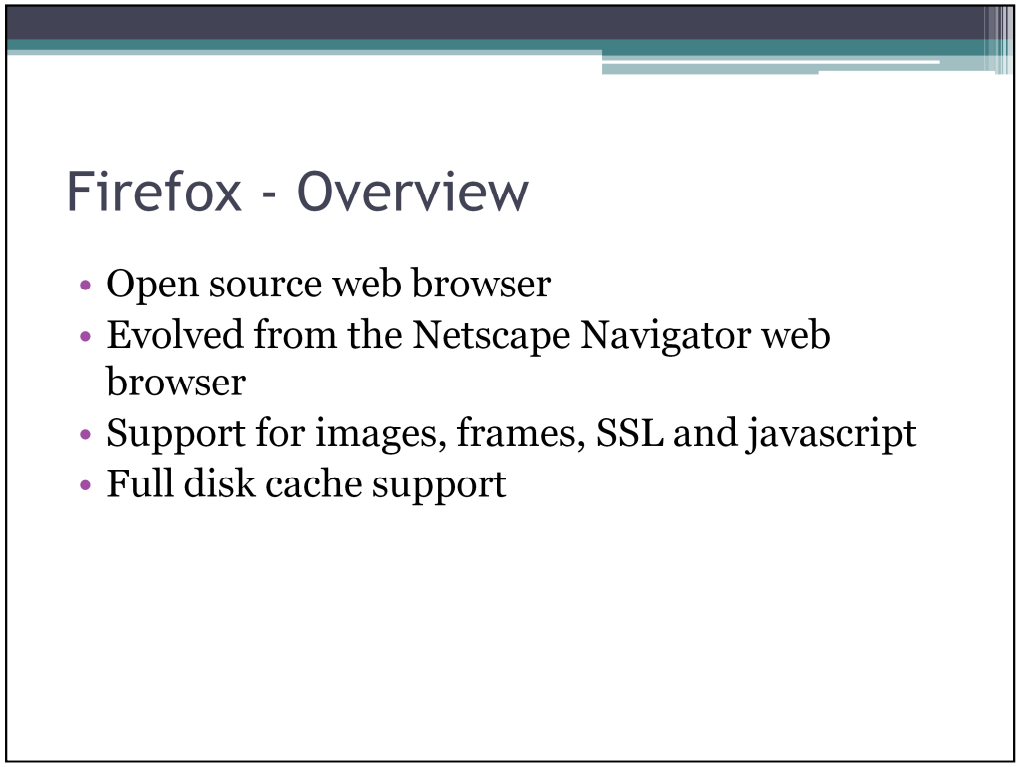
- IE stores history, cache, cookie and authentication information in multiple locations
- The consistent size of hex offsets allows you to analyze an incomplete or corrupted history file
- There are multiple tools to analyze IE's stored information, even if the information is "deleted" or protected by a password

This page intentionally left blank.

Firefox – What We Will Cover

- Where Firefox stores files used in displaying web pages (cache), tracking pages visited (history) and automatic identification / authentication (cookies, credentials)
- How to access the information using just the browser
- Tools used to analyze Firefox's history, cached files, cookies and stored credentials
- Tools used to override protection of the stored credentials

This page intentionally left blank.



Firefox is a web browser based on the Gecko layout engine. This programs (and the Gecko engine) evolved from the Netscape Navigator web browser (which in turn got its underpinnings from the original web browser, Mosaic, at the National Center for Supercomputing Applications.) You can download Firefox from http://www.mozilla.com/en-US/firefox/firefox.html.

Full disk caching of viewed web pages is supported, just like with Internet Explorer. Firefox's supporting files contain a subset of the data fields previously described:

• URL: The Uniform Resource Location (URL) of the visited web page. The user can invoke a URL either by following a link or by typing a URL in the browser's location bar.

• Time and date of access: Time and date when a specific URL was accessed. In Mozilla and Firefox, this includes both the time and date the page was first accessed, and the time and date it was last accessed.

• Times accessed: The number of times a given URL was invoked (includes the act of downloading files from the site.)

• Cache objects: The files that make up a web page, which are downloaded and saved according to Firefox's caching algorithm (which we'll get to later.)

• Downloads: Information about every file downloaded that is not a cache object (i.e. file attachments that the user has intentionally downloaded.)

Firefox – File Locations

- Firefox stores its history, downloads, form fields, cookies, and Identification / Authentication files in the same location:
- `C:\Documents and Settings\<subject User's ID>\Application Data\Mozilla\Firefox\Profiles\<seemingly random characters>.default\` (Windows XP) or
- `C:\Users\<subject User's ID>\AppData\Local\Mozilla\Firefox\Profiles\<seemingly random characters>.default\` (Windows Vista, 7 and 2008)

In Firefox , six types of files are of interest to the forensic examiner: `places.sqlite` for browsing history, `CACHE_MAP` for locating the actual cache files, `downloads.sqlite` for downloaded files, `formhistory. sqlite` for filled-out form fields, `cookies.sqlite` for cookie information, and finally `signons3.txt` for encrypted userIDs and passwords. Five of these six files (`places.sqlite, downloads.sqlite, formhistory.sqlite, cookies.sqlite` and `signons3.txt`) are stored in the same location:

`C:\Documents and Settings\<subject User's ID>\Application Data\Mozilla\Firefox\Profiles\<seemingly random characters>.default\` (Windows XP)

or

`C:\Users\<subject User's ID>\AppData\Local\Mozilla\Firefox\Profiles\<seemingly random characters>.default\` (Windows Vista, 7 and 2008 – if your user has a domain roaming profile, substitute `Roaming` for `Local` in the path above.)

Firefox – File Locations (2)

- Firefox stores its cache files in a different location:
- `C:\Documents and Settings\<subject User's ID>\Local Settings\Application Data\Mozilla\Firefox\Profiles\<seemingly random characters>.default\Cache\` (Windows XP) or
- `C:\Users\<subject User's ID>\AppData\Local\Mozilla\Firefox\Profiles\<seemingly random characters>.default\Cache\` (Windows Vista)

Just to be cute, Firefox had to go and store its cache files (including `CACHE_MAP`) in a completely different location:

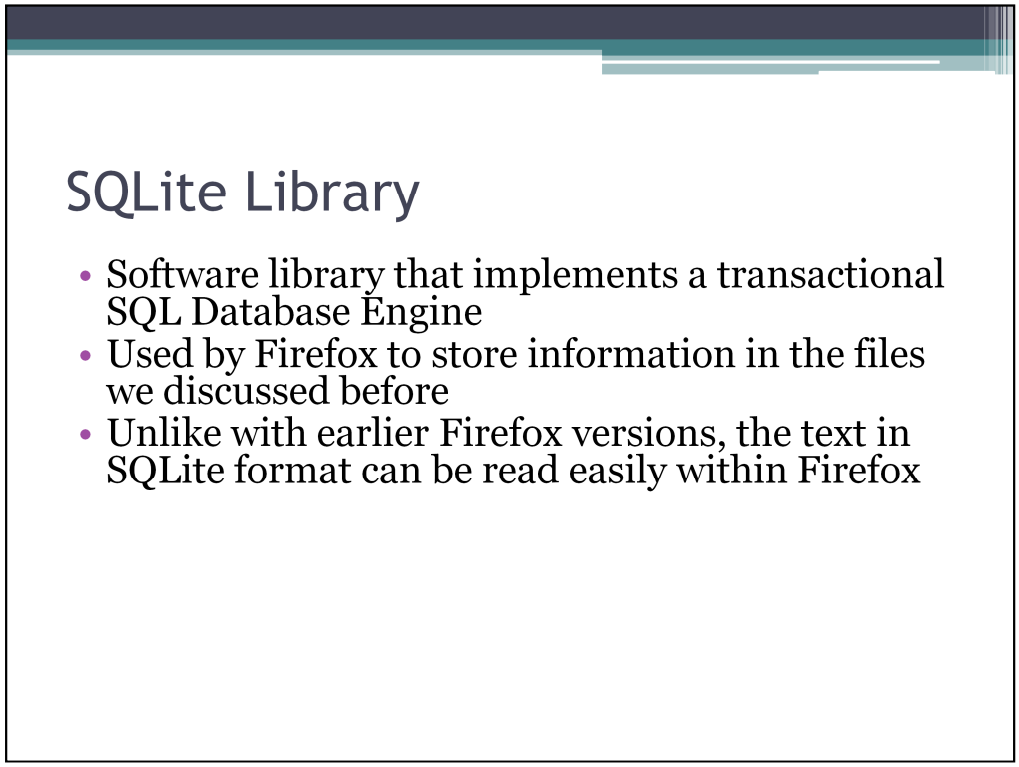
`C:\Documents and Settings\<subject User's ID>\Local Settings\Application Data\Mozilla\Firefox\Profiles\<seemingly random characters>.default\Cache\` (Windows XP)

or

`C:\Users\<subject User's ID>\AppData\Local\Mozilla\Firefox\Profiles\ <seemingly random characters>.default\Cache\` (Windows Vista, 7 and 2008 – if your user has a domain roaming profile, substitute `Roaming` for `Local` in the path above.)

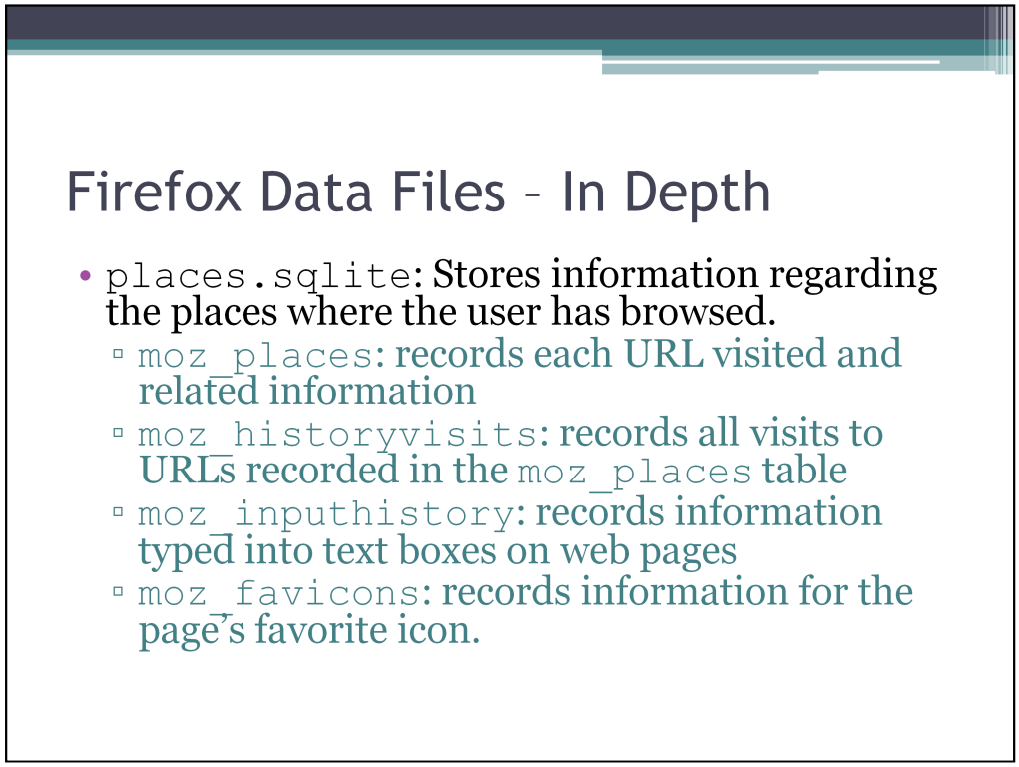
Even cuter, the cache files aren't immediately usable like those used by Internet Explorer; you can't go into the `Cache` directory and double-click on, say, an image file to bring it up in Paint. The cache files are stored across multiple other files; how to "pry them out" for your viewing will be covered a few slides from now.

None of the files are human-readable with a text editor; that'd be too easy! Luckily our tools once again will provide us with a method to parse the information.



SQLite is a software library that implements a single-file-contained, zero configuration, transactional Structured Query Language (SQL) database engine. Firefox makes use of this library to read from and/or write to some of the files we discussed before (`places.sqlite`, `downloads.sqlite`, `formhistory.sqlite`, and `cookies.sqlite`), both on Windows and UNIX operating systems.

If we want to read what information is present in these files, we can use a separate program to invoke the library and then perform SQL queries on the database. The library is self-contained in Firefox.



Let's take some time to delve further into these files. Again, places.sqlite is a database file (analogous to History.dat in IE) that contains tables with information about where the subject has surfed. The tables in places.sqlite are as follows:

- moz_places
- moz_historyvisits
- moz_inputstory
- moz_favicons

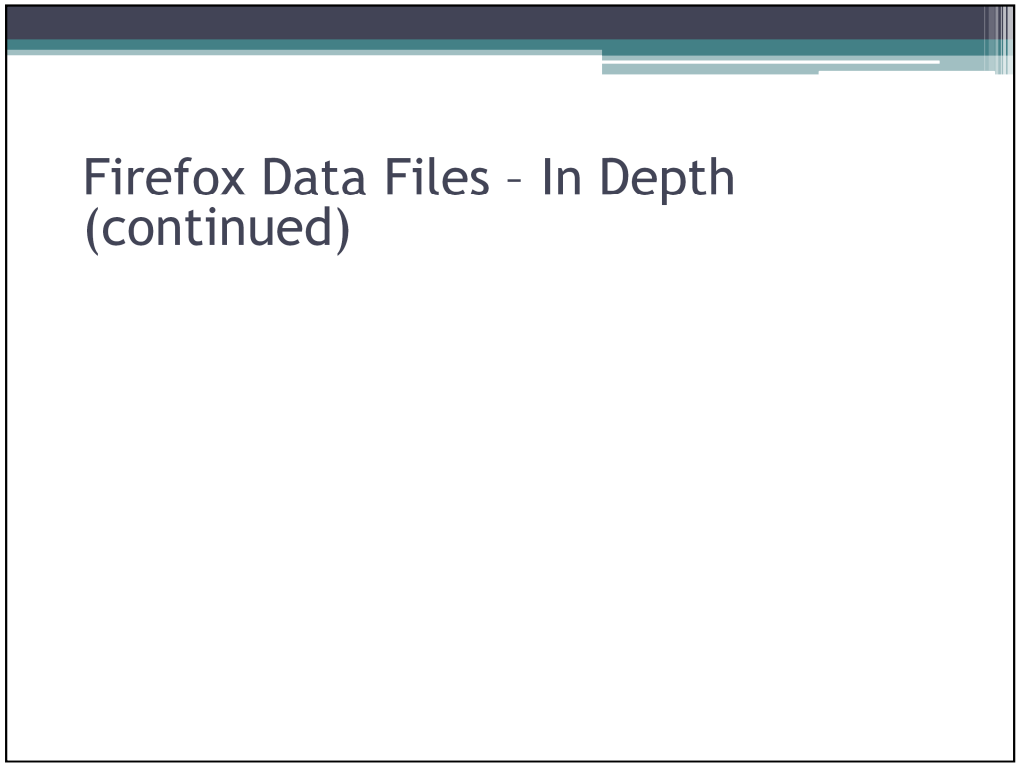
moz_places records each URL the subject browsed to and some additional information associated with it. This table consists of the following fields:

• ID: Unique identified of the page visited. Corresponds to the place_id field in the moz_historyvisits table and serves to connect records in the two tables to one another.

• URL: URL of the page visited and the protocol used.

• Title: Page title read from the HTML <TITLE> tag of the visited page. If the URL is not an HTML page (like an ASP page), this value is instead set to the one shown in the title bar of the web browser.

• Rev_url: This is the hostname of the URL, typed backwards with a trailing period (no, we don't know why they decided to store it that way either.)



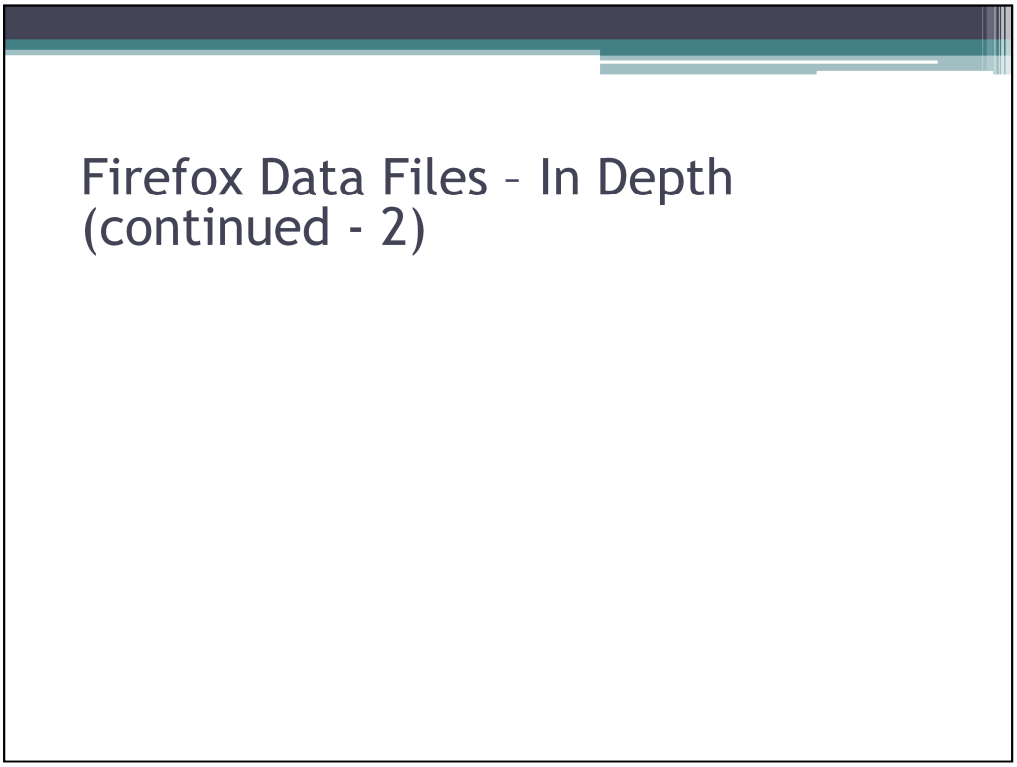
• Visit_count: This is the number of times the page has been visited. For a given place_id in moz_historyvisits (remember, that corresponds to ID in this table), there is a visit_type associated in moz_historyvisits. If this visit_type is not 0, 4 (embedded) or 7 (download), the visit_count field is not incremented. The field is also not incremented when the subject hits the reload button in Firefox within a specific URL.

• Hidden: This field allows values of 0 or 1, where 0 means not hidden and 1 means hidden. A hidden URL means the user did not specifically navigate to it and instead the URL was called from another page. Some examples of this are embedded pages, i-frames, RSS bookmarks and javascript calls.

• Typed: This field also allows values 0 or 1, where 0 means the user did not invoke it directly by typing it from the location/URL bar.

• Favicon_id: This field is the favorite icon id for the URL and corresponds to the same field in the moz_favicons table.

• Frecency: Yes, you read correctly, frecency - a combination of frequency and recency. This field is used to order the suggested URLs that appear in the autocomplete function of Firefox's location/URL bar. The more a frequently a URL is visited, and the more recently is the last time it was visited, the more the value of this field is increased. A -1 frecency value means the frecency for the URL has not been calculated. If there is a 0 value, the page will be excluded from the autocomplete bar. All other URL values will be ordered based on the frecency.



moz_historyvisits records all visits to URLs recorded in the moz_places table. This table consists of the following fields:

• ID: A Firefox-generated value for indexing the records in the table; this value increments with each new record.

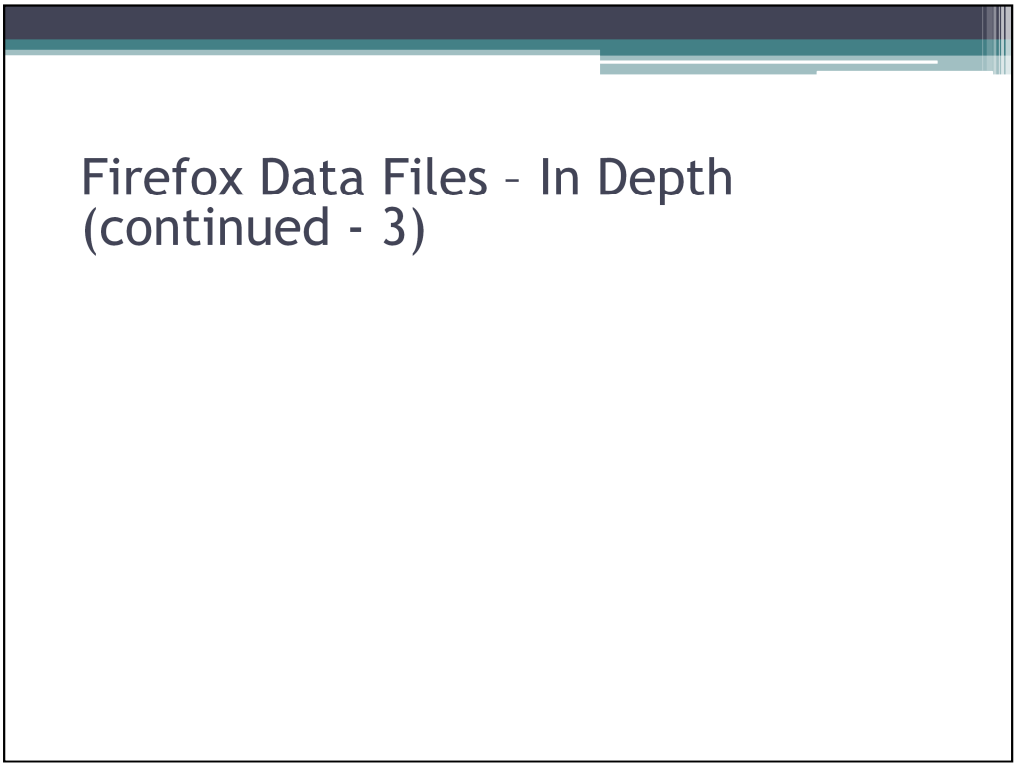
• place_id: A Firefox-generated value to uniquely identify the URL visited. Recall that this field corresponds to the ID field in moz_places.

• from_visit: If the URL visited is invoked from a link contained on a previously visited URL, this field will have the place_id value from that visit.

• visit_date: Date and time that the browsing occurred, formatted as PRTime. We'll get to PRTime below.

• visit_type: The value of this field records the means by which the user navigated to the web page.

- 1: The user followed a link
- 2: The user typed the URL into the location/URL bar or selected it using the autocomplete feature, clicked on it from the history sidebar, history menu or
- 3: The user followed a bookmark
- 4: The URL is embedded in another URL. That means the user accessed one URL, which as part of its page load invoked a different one. Examples of this include an embedded page, i-frame, or a javascript call, among others.
- 5: The user arrived to the URL via a permanent redirect (HTTP 301 http://en.wikipedia.org/wiki/HTTP_301).
- 6: The user arrived to the URL via a temporary redirect (HTTP 302/307 http://en.wikipedia.org/wiki/HTTP_307#3xx_Redirection)
- 7: This URL was used to download a file.



moz_inputhistory is part of Firefox's autocomplete feature and records information typed into text boxes on web pages. This table consists of the following fields:

• place_id: ID of the page where the text was input. This field corresponds to place_id in the moz_historyvisits table.

• input: The actual text that the user typed.

• use_count: This counter shows how many times the saved information has been used again by the user.

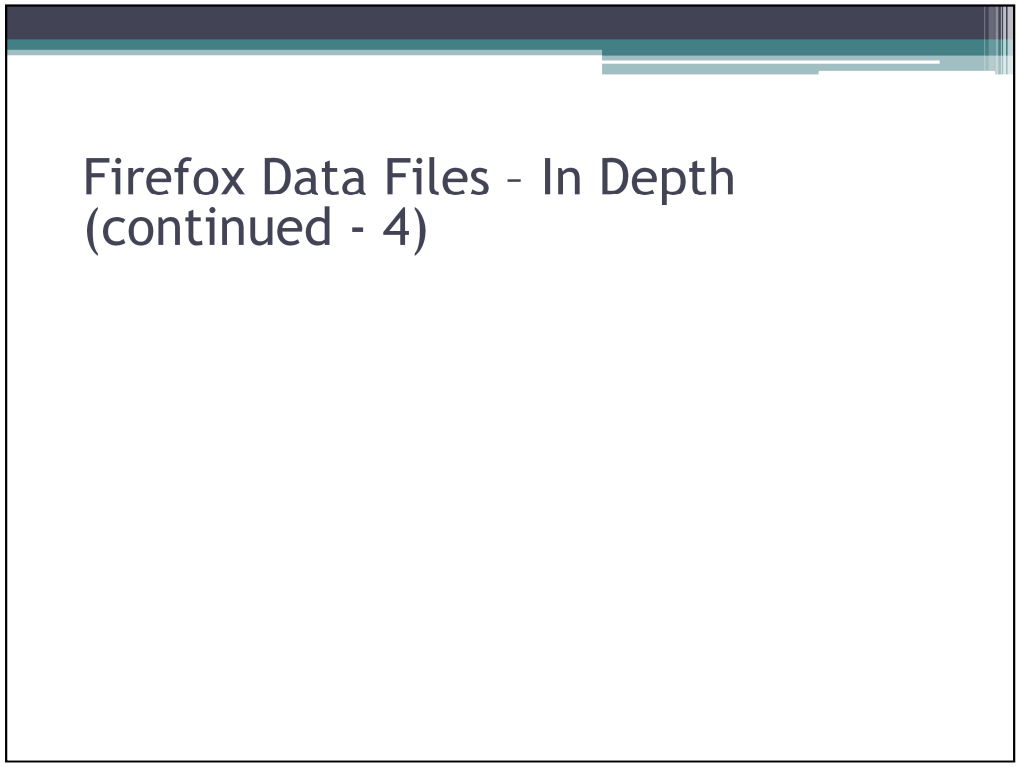
moz_favicons records each favorite icon retrieved from a webpage (the little icon that is visible just to the left of the URL in the location / address bar.) This table consists of the following fields:

• ID: ID of the page corresponding to the icon. This field corresponds to place_id in the moz_historyvisits table.

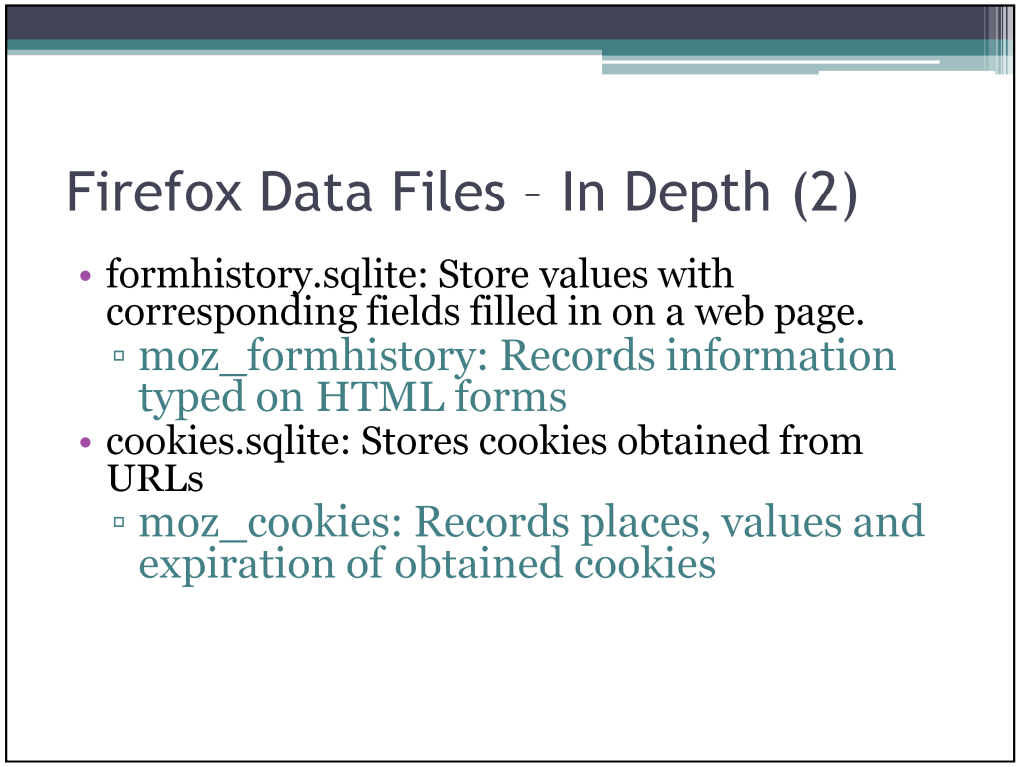
• URL: URL from where the icon was downloaded.

• data: The icon itself.

• mime/type: The icon file format.



Did we miss anything? Oh yes, PRTime. PRTime is a time format unique to Firefox 3. Fortunately, it's also very close to a more common time format: Unix time. The only difference is that PRTime counts in microseconds since January 1, 1970 (again, don't ask us why), whereas Unix time counts in full seconds from that same moment in history. There are multiple resources on the web for converting Unix time into "regular" time; all you have to do in order to make use of PRTime is divide it by 1,000,000 and then put it into one of the conversion websites, like this one: http://www.onlineconversion.com/unix_time.htm (the output of this page is in good old GMT, so be sure to convert it to the local time zone where the subject computer was located.)



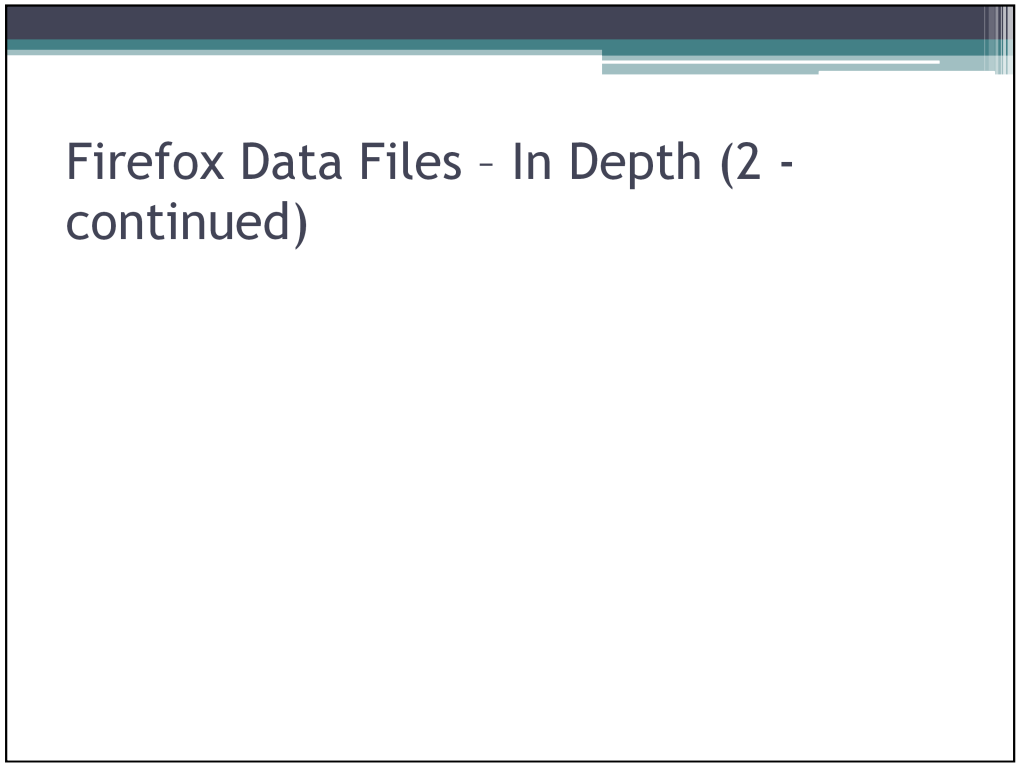
The formhistory.sqlite database file stores information typed in by the subject in fields on a web page form. This information is stored in the moz_formhistory table, which has the following fields:

• ID: A Firefox-generated value for indexing the records in the table; this value increments with each new record.

• fieldname: Name of the web page's form field where the subject typed the text.

• value: The actual text typed by the subject in the fieldname, above.

"Wait!" you may be saying. "I thought moz_inputhistory saved those values!" It does, but only for pages with one field to fill in (like, say, Google.) formhistory.sqlite/moz_formhistory saves data for the more complex multi-field forms.



The cookies.sqlite database file stores cookies saved by Firefox as the subject browses. This information is stored in the moz_cookies table, which has the following fields:

• ID: A Firefox-generated value for indexing the records in the table; this value increments with each new record.

• name: The name of the cookie as sent by the server hosting the web page.

• value: This is the cookie itself.

• host: The domain of the hosting server that sent the cookie, beginning with a period.

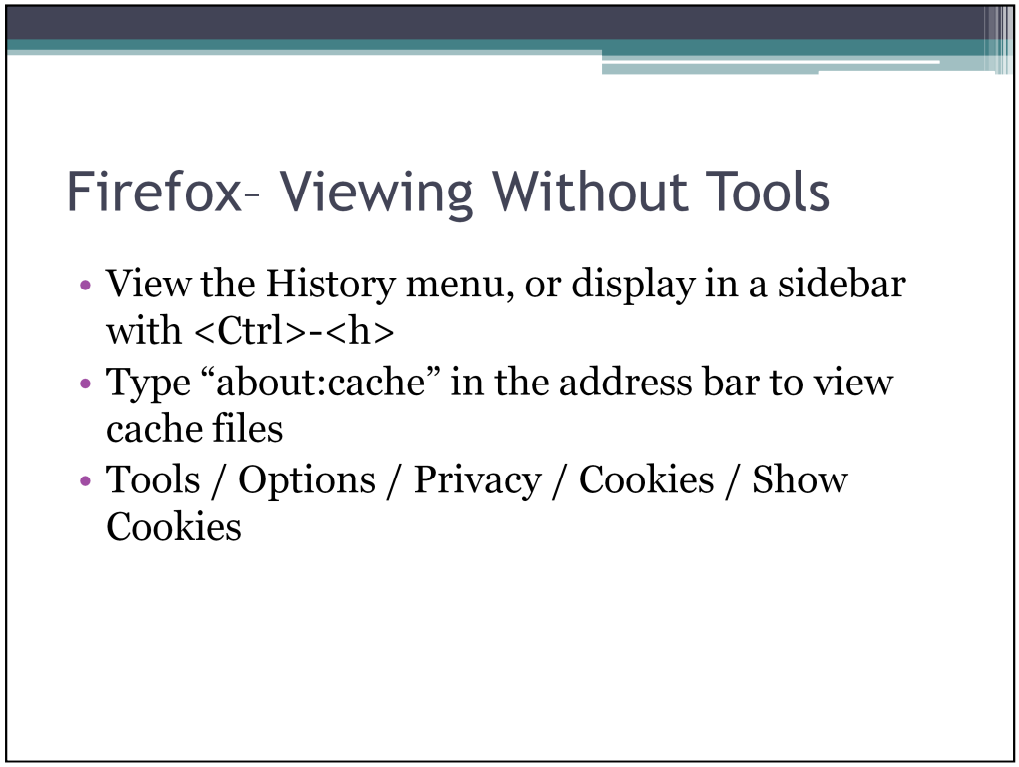
• path: The file path to the cookie on the hosting server.

• expiry: Date and time of the cookie's expiration, formatted in PRTime.

• last_accessed: Last time the cookie was accessed by the browser, formatted in PRTime.

• IsSecure: The value for this field is 1 if the cookie was sent using https (SSL/TLS) protocol. Otherwise, it is set to 0.

• IsHttpOnly: The value for this field is 1 if the cookie cannot be accessed using a client-side script. Otherwise, it is set to 0.



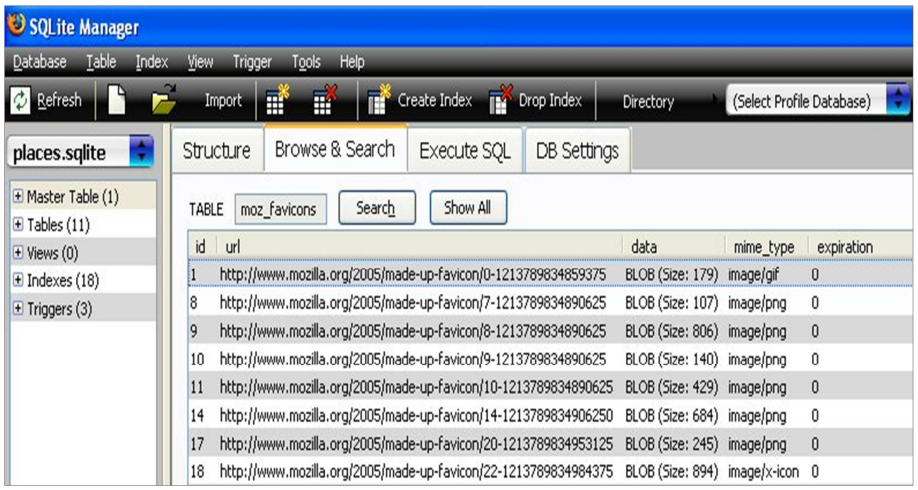
Firefox features a History choice right in the top menus. Unfortunately, this History is far from complete – only the last 10 sites visited show in the menu. Our old friend <Ctrl>-<h> comes into play here, bringing up a sidebar menu of visited sites that is nicely organized. You can use the View drop-down menu to arrange the pages visited by date and site, by site, by date, by most visited, and by last visited. You can also use the Search box to find a page. Is any of this sounding familiar? ☺

You can also view cached files directly in Firefox by typing “about:cache” in the browser’s address bar. This will give you a choice (you click on a web link to choose) of viewing the memory cache or the disk cache, organized by the last site visited. You can click once on a given file to get more detailed information about it, including its related HTTP headers; click again to see the file re-loaded in the browser.

Finally, Firefox allows you to view cookies directly by going to the Tools / Options / Privacy / Cookies / Show Cookies menu, and then clicking on any cookie for detailed information.

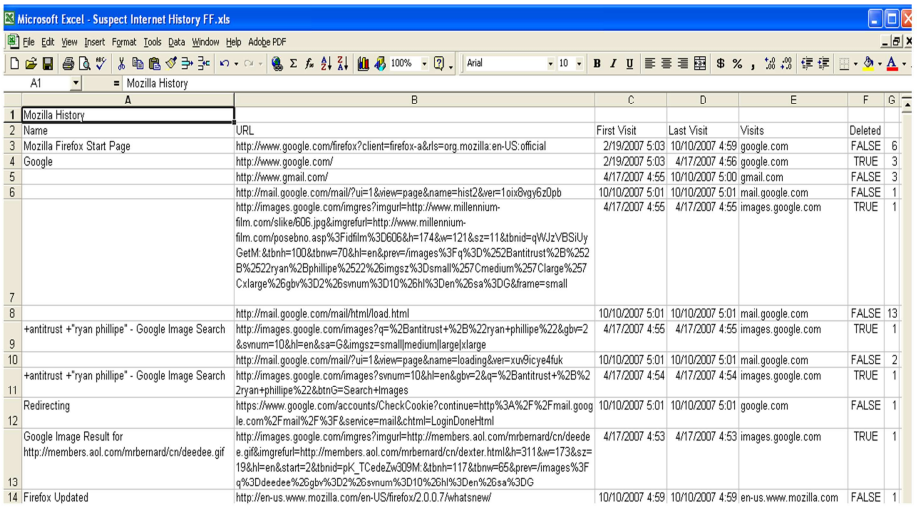
Pop quiz: These are nice features, but what are you doing by viewing this information directly in the browser? Wait for it….. yes, you’re changing the very information you want to analyze! Additionally, by using the History menu, you can’t view or export the information all at once for further analysis.

Firefox– Viewing (Almost) Without Tools



A program called SQLite Manager plugs right into Firefox and allows you to directly browse the various SQLite files contained within Firefox's profiles. SM can be downloaded from https://addons.mozilla.org/en-US/firefox/addon/5817. Refer to the previous slides as to the location and purpose of the various databases, tables and fields. Remember that, once again, you're directly viewing information and therefore changing it as you view it. To avoid this contamination of evidence, we turn to tool-based analysis of the historical files.

Mandiant Web Historian – Firefox



Microsoft Excel - Suspect Internet History FF.xls

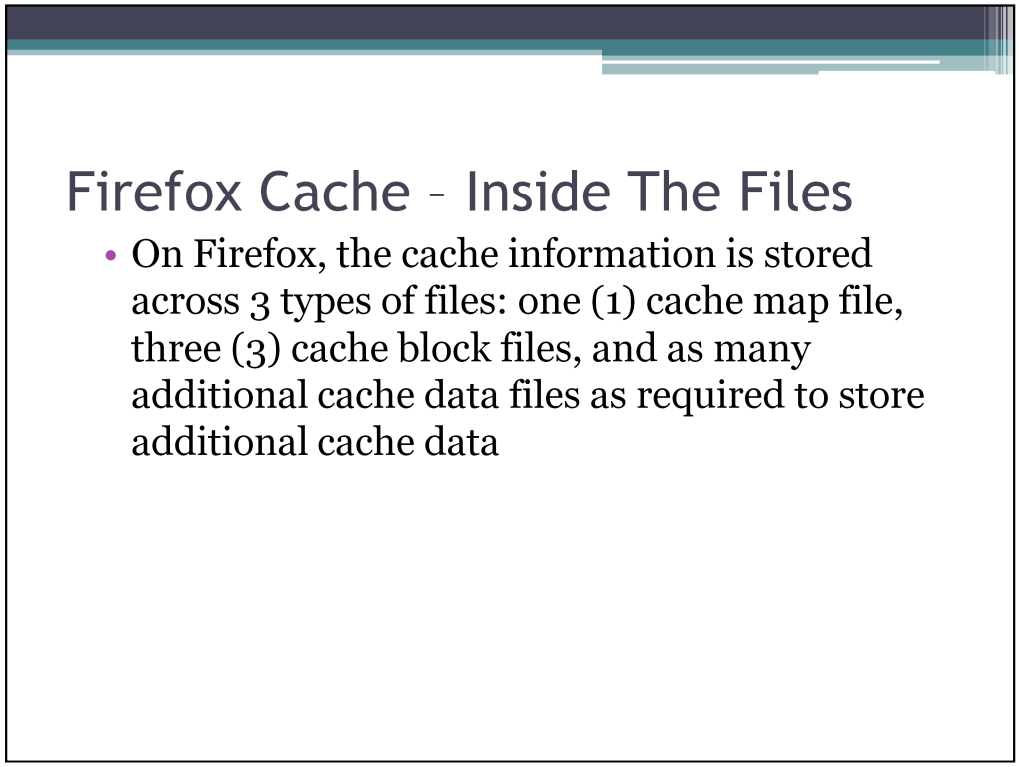
	A	B	C	D	E	F	G
1	Mozilla History						
2	Name	URL	First Visit	Last Visit	Visits	Deleted	
3	Mozilla Firefox Start Page	http://www.google.com/firefox?client=firefox-a&rls=org.mozilla:en-US:official	2/19/2007 5:03	10/10/2007 4:59	google.com	FALSE	6
4	Google	http://www.google.com/	2/19/2007 5:03	4/17/2007 4:56	google.com	TRUE	3
5		http://www.gmail.com/	4/17/2007 4:55	10/10/2007 5:00	gmail.com	FALSE	3
6		http://mail.google.com/mail/?ui=1&view=page&name=hist2&ver=1oix8vgy6z0pb	10/10/2007 5:01	10/10/2007 5:01	mail.google.com	FALSE	1
7		http://images.google.com/imgres?imgurl=http://www.millennium-film.com/slike/606.jpg&imgrefurl=http://www.millennium-film.com/posebno.asp%3Fidfilm%3D606&h=174&w=121&sz=11&tbnid=qWJzVBSiUyGetM:&tbnh=100&tbnw=70&hl=en&prev=/images%3Fq%3D%252Bantitrust%2B%252B%2522ryan%2Bphillipe%2522%26imgsz%3Dsmall%257Cmedium%257Clarge%257Cxlarge%26gbv%3D2%26svnum%3D10%26hl%3Den%26sa%3DG&frame=small	4/17/2007 4:55	4/17/2007 4:55	images.google.com	TRUE	1
8		http://mail.google.com/mail/html/load.html	10/10/2007 5:01	10/10/2007 5:01	mail.google.com	FALSE	13
9	+antitrust +"ryan phillipe" - Google Image Search	http://images.google.com/images?q=%2Bantitrust+%2B%22ryan+phillipe%22&gbv=2&svnum=10&hl=en&sa=G&imgsz=small|medium|large|xlarge	4/17/2007 4:55	4/17/2007 4:55	images.google.com	TRUE	1
10		http://mail.google.com/mail/?ui=1&view=page&name=loading&ver=xux9icye4fuk	10/10/2007 5:01	10/10/2007 5:01	mail.google.com	FALSE	2
11	+antitrust +"ryan phillipe" - Google Image Search	http://images.google.com/images?svnum=10&hl=en&gbv=2&q=%2Bantitrust+%2B%22ryan+phillipe%22&btnG=Search+Images	4/17/2007 4:54	4/17/2007 4:54	images.google.com	TRUE	1
12	Redirecting	https://www.google.com/accounts/CheckCookie?continue=http%3A%2F%2Fmail.google.com%2Fmail%2F%3F&service=mail&chtml=LoginDoneHtml	10/10/2007 5:01	10/10/2007 5:01	google.com	FALSE	1
13	Google Image Result for http://members.aol.com/mrbernard/cn/deedee.gif	http://images.google.com/imgres?imgurl=http://members.aol.com/mrbernard/cn/deedee.gif&imgrefurl=http://members.aol.com/mrbernard/cn/dexter.html&h=311&w=173&sz=19&hl=en&start=2&tbnid=pK_TCedeZw309M:&tbnh=117&tbnw=65&prev=/images%3Fq%3Ddeedee%26gbv%3D2%26svnum%3D10%26hl%3Den%26sa%3DG	4/17/2007 4:53	4/17/2007 4:53	images.google.com	TRUE	1
14	Firefox Updated	http://en-us.www.mozilla.com/en-US/firefox/2.0.0.7/whatsnew/	10/10/2007 4:59	10/10/2007 4:59	en-us.www.mozilla.com	FALSE	1

As mentioned before, Mandiant Web Historian is capable of analyzing the history information from multiple browsers, including Firefox. It's a simple matter of going to File / Options and putting a check next to Mozilla / Firefox / Netscape Navigator under Browser activity to search for. Then, point the program at the `places.sqlite` file (either in the default location or wherever you have extracted it to) and run the analysis. The file output options are the same as before.

The column headers include the Name (web page title), Uniform Resource Location (URL) address, First Visit (timestamp, in UTC), Last Visit (timestamp, in UTC), Visits (the Fully Qualified Domain Name of the website hosting the page, without all the subdirectories or script names), Deleted (status true or false), and the number of times the page was visited by the browser.

The most useful columns to sort by are Name, URL and Visits, because that will sort the rest of the information by website visited.

As before, the First Visit and Last Visit columns can give you a timeline of the web page accesses, which you can compare to other records (building access, network login) to establish who was doing the browsing.



Further complicating the cache-viewing situation in Firefox browsers is that the actual cache information (and cache files) may be stored across four or more files, as needed, to store additional data. The cache map ("_CACHE_MAP_") file contains records of the location of the cache file's metadata (last modified / expiry date etc.) as well as the location of the cache data / cache block files themselves.

A very involved algorithm determines whether the metadata and data are stored in the cache block files ("_CACHE_001_ through 003") or in the cache data files. This algorithm also determines the name of the cache data file, if that's where the data is to be stored. And *no*, we aren't going to slog through that algorithm in this course, or we'd be here a week. We'll simply let a program do the math for us.

For more information about the algorithm and how to use it to manually extract cache files (if you are a math expert), see the seminal article "Browser Forensics" at :

www.securityfocus.com/print/infocus/1832

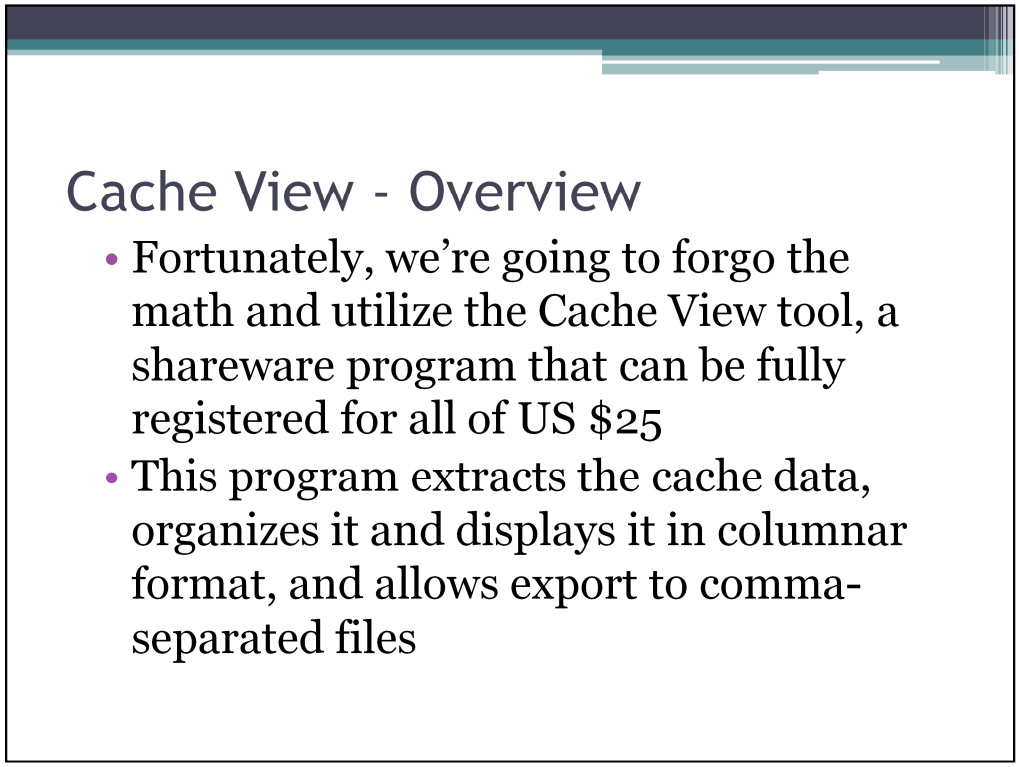
www.securityfocus.com/print/infocus/1827

Firefox – What If The subject Clears The Cache?

- In Firefox, the situation is skewed much more in favor of the subject. Going to Tools and selecting Clear Private Data deletes not only the cache files, but handily removes the cache map and cache block files, so tying the files (assuming you could recover them) to the cache map and blocks becomes quite a bit more difficult

One of the reasons people perceive the Firefox browser to be more secure than Internet Explorer (irrespective of whether it is actually true) is that it has more of a "secure by default" posture, meaning that the default configuration of the browser has more of the security options set to "yes" (like prevent popup windows, for example) and it allows people to take security-improving actions more easily. This is certainly true in the case of cache files. Instead of making the user first delete the cache files, then find and manually delete the index to those cache files, Firefox simply blows away anything having to do with your cache in one fell swoop. The cache map and block files are deleted right along with the cache files themselves using the Tools / Clear Private Data command. The map and block files are regenerated the next time Firefox is directed to a website, or when the browser is restarted.

What does this mean for the investigator? A big headache, that's what, when your subject is clever enough to take this action. You are now stuck with the task of locating and extracting the cache map, cache block, and cache files, and then (and only then) can you run the convenient analysis tools we'll look at in the next few slides.

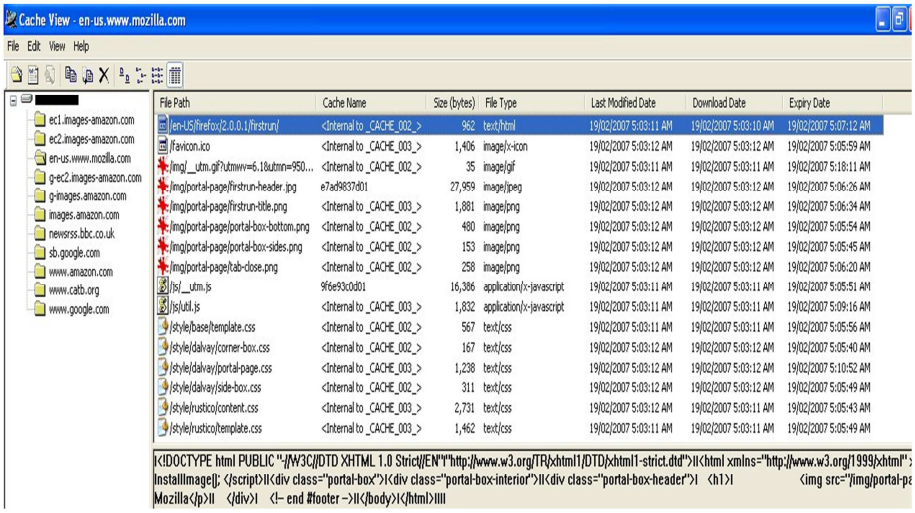


Cache View is a terrific little program that will display the extracted cache data in a columnar format for easy review, analysis, and export (into a comma-separated file.) The program is shareware, meaning that you can use all of its features, but the author requests you register it for the very small sum of $25. By default, the unregistered version of Cache View requires you to specify the location of the cache files you want to view. Registering Cache View means it will automatically scan your hard drive and display the caches from the various browsers you may have installed on your system. Of course, registering also encourages the author to make future improvements to the application (all of which are free to the registered user) and is The Right Thing To Do as well.

To have Cache View analyze other cache files (say, those imported to your forensic workstation – hint, hint) go to View / Options / Displayed Cache Directories, pick Show browser cache in this folder, and click browse to find it.

You can download the latest version of Cache View (2.8.05, at the time of this writing) at http://www.progsoc.uts.edu.au/~timj/cv/.

Cache View - Firefox

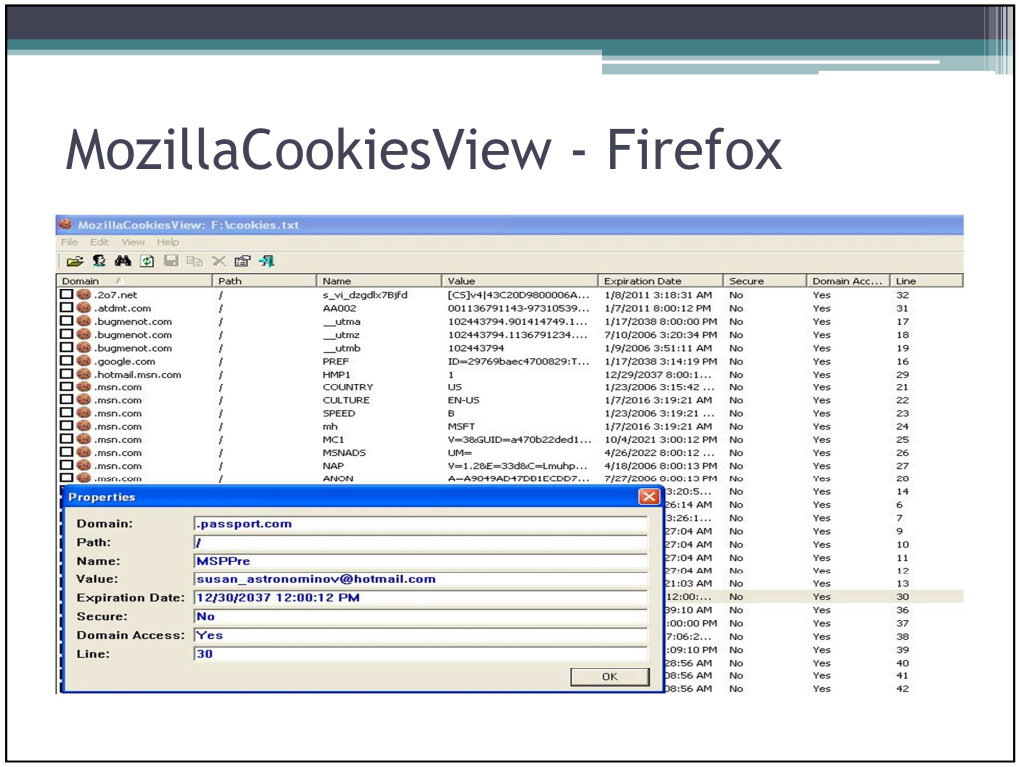


The column headers include the original path to and filename of the cached file, where the cache data was located (cache block or separate file), the file size, the file type, last modified date / time, download (first access) date / time, and the cache file expiration date / time. You can click on any file to show the HTTP header information that called it (below the file listing.)

There are multiple things you can do with each file by right-clicking on it:

- •Open the file locally
- •Open the URL (web address) on the Internet associated with the file
- •See detailed properties information about the file
- •Copy / move a collection of files to another directory so you will have a local copy of the website that was visited

This last point is the most useful, as you can sort the columns by the file path, copy the ones relevant to a particular website, paste them elsewhere in your computer, and you will have a working local copy of the visited website.



MozillaCookiesView (MZCV) is a free program from Nirsoft that accomplishes for Firefox the same goal that Galleta accomplishes for Internet Explorer: getting all the cookie information into one nice, readable file. When started, MZCV will access the local computer's default Firefox cookies file and display it. To analyze a different cookies file (or profile folder, if you want to analyze multiple cookies files at once) go to File / Select Cookies File/Folder. You can download MZCV from http://www.nirsoft.net/utils/mzcv.html.

In addition to the GUI, you can also run MozillaCookiesView from the command line:
`mzcv –cookiesfile (filename) <option>` or `mzcv –profilesfolder (foldername) <option>`

The `–profilesfolder` option accomplishes the same thing as selecting a profile folder in the GUI.

The command option switches are numerous here, all having to do with the output file format: save as a text file, a tab-delimited file, a tabular text file, an HTML file (horizontal or vertical) or an XML file: `/stext, /stab, /stabular, /shtml, /sverhtml,` and `sxml`.

Stop us if you've heard this before: Follow the advice on the Web Historian slides for column sorts and file retrieval. To ensure successful execution, it's a good idea to copy the cookie text files you wish to parse to the same directory as the `mzcv` executable file.

FireMaster – Stored Credentials

- Firefox gives you the option to save your often-used userIDs and passwords that you utilize to access websites
- Unfortunately for the forensic investigator, the subject may specify a Master password, which prevents access to all the other passwords
- FireMaster cracks this master password, allowing you to access the password list in the browser or via FirePassword

Similar to Internet Explorer, Firefox will ask you if you want to save often-used userID and password information (say, for logging into your favorite webmail site) so that you don't have to type it in again and again. This can be a gold mine of information to the investigator, as it allows him or her to access information stored outside of the browser, or even on the subject's computer. Do we have to remind you to get appropriate permissions before pursuing this line of research? ☺

Unfortunately, sometimes the subject will be appropriately paranoid and assign a Master Password to his or her password list. Therefore, you can't see all the other passwords without specifying the Master Password.

N.Y. Talekar has written the excellent utilities FireMaster and FirePassword (which we'll review in the next slide), both of which can be downloaded at http://www.securityxploded.com/download.php, to solve this problem. From the command line, type

```
Firemaster [type of crack you want – -d (dictionary), -b (brute-force) or –h (hybrid)] [-f name of wordlist] [location of signons.txt]
```

There are 3 types of crack attempts:

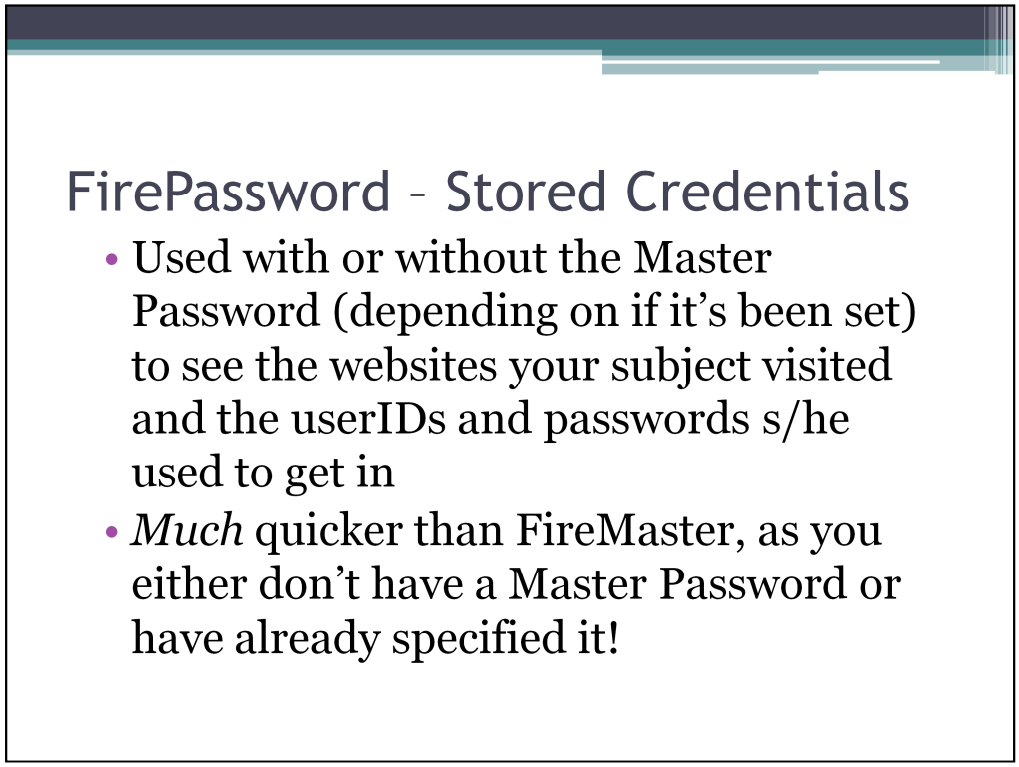
•Dictionary – Try every word listed in a specified wordlist file

•Brute-force – Try every letter, number or special character specified in every combination, up to a maximum password length specified

•Hybrid – A dictionary attack that also uses specified characters (like password12345.)

It's preferable to run this program on a dedicated, fast system – even a minor dictionary crack will pin your processor at 100% for hours, so don't try to do anything else on that computer while the crack takes place.

Once you have the Master Password, you can enter it into the subject's browser, but then you'd be changing evidence, correct? With the Master Password in hand, however, we can use FirePassword to get at all the other passwords.



Now that we have the Master Password (or our subject didn't bother to use one in his or her copy of Firefox) we can use FirePassword to complete the revealing of the website access credentials. The output includes the name of the website, the userID (it may not always be called "ID" – for example, in Amazon your ID is your e-mail address) and the password.

From the command line, type:

```
FirePassword [-m "master password" ] [location of
signons.txt]
```

Remember, you can also use the > symbol after the filename to "redirect" output into a file for printing.

Dump AutoComplete – Stored Form Fields

- Information in `formhistory.sqlite` can be viewed with any sqlite-compatible program, but must be extracted to be the most useful to the investigator
- Use the Dump Auto Complete program to bring out the form field name and what was entered
- Even though these aren't credentials, a lot of information about the subject can be gleaned

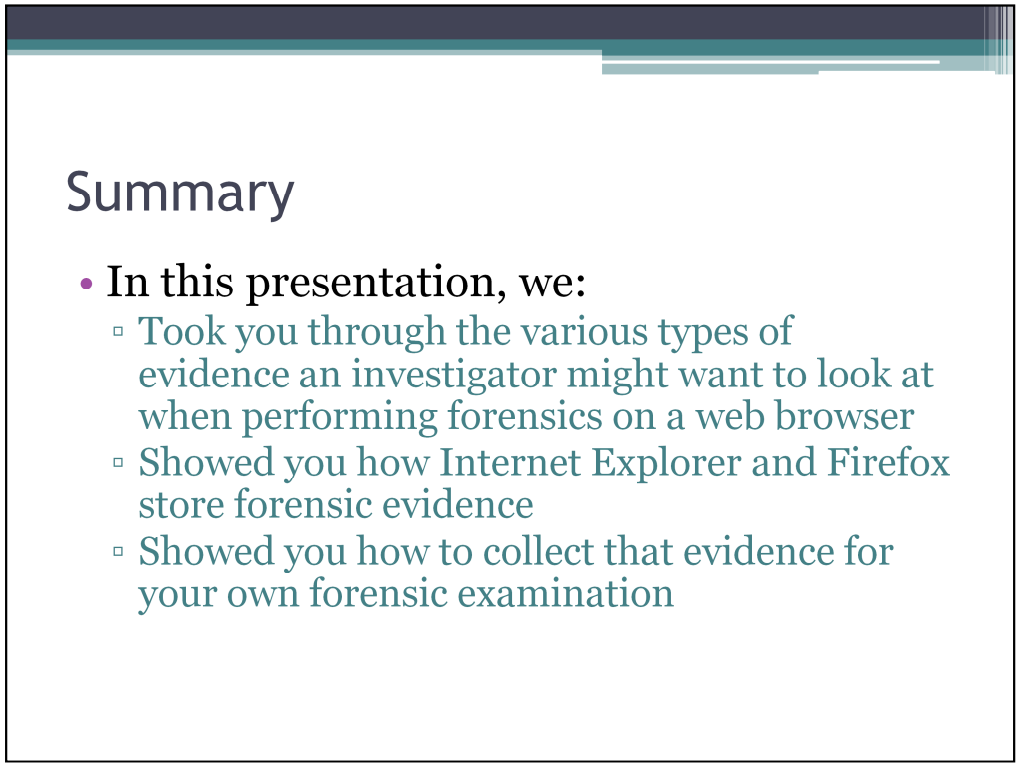
In Firefox, completed form-fields stored unencrypted in `formhistory.sqlite` (as opposed to user credentials, stored encrypted in `signons3.txt`) can be brought out and viewed by a program called Dump AutoComplete. You may ask, "if this information is unencrypted, can't I just read it with Sqlite manager like you showed me before?" Unfortunately, while that program permits the browsing of sqlite information directly inside Firefox, in order to be of maximum usefulness to the investigator you need to be able to extract the information. Since we (and you, trust us) don't want to have to run through creating SQL queries by hand, we rely on DumpAutoComplete instead.

To use Dump AutoComplete, go to a command line and type "dumpautocomplete." The only options are for you to specify a formhistory.dat sqlite location (if none is specified it will automatically grab the one in the current user's default Firefox location.) Of course, as with all our command line programs, you can use the > symbol after the filename to "redirect" output into a file (such as a tab-delimited file that can then be read into Microsoft Excel.)

Firefox – Review

- Firefox stores history, cache, cookie and authentication information in multiple locations, which differ in placement and format from Internet Explorer
- There are multiple tools to analyze Firefox's stored information, even if the information is "deleted" or protected by a password – however, in the first instance it's more difficult than IE, and in the second it's easier ☺

This page intentionally left blank.

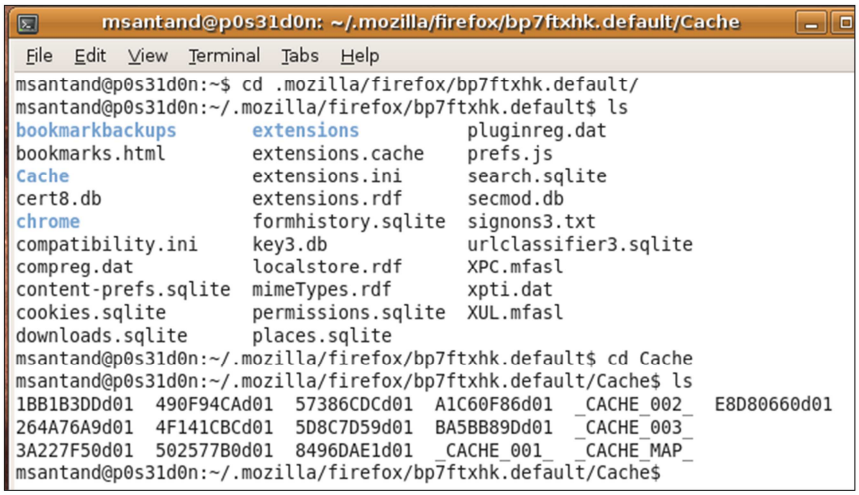


This concludes our Browser Forensics presentation. You have been given a basic overview of the types of evidence recorded by web browsers that would be of interest to a forensic investigator. You were then treated to an in-depth coverage of where and how each of the two major Windows-based browsers (Internet Explorer and Mozilla Firefox) store these pieces of evidence about the person using them. We then covered the methodologies and tools used to access this information so that it can be used in a forensic review. Next you will find an Appendix that discusses forensics for the Mozilla Firefox browser on Unix-derivative systems.

Appendix – Firefox on Unix

This page intentionally left blank.

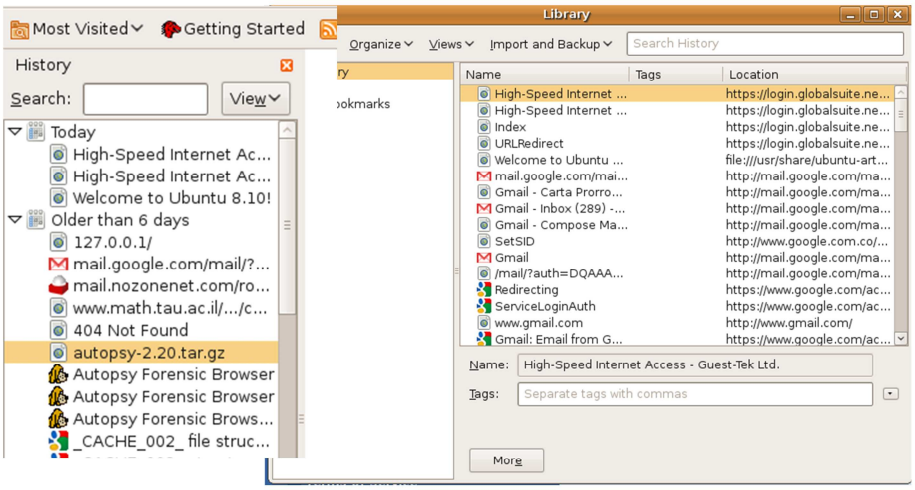
Firefox File Locations - Unix



Firefox uses the .mozilla/firefox directory, both under the ./home/<Suspect User's ID>/ folder. The following directories are of interest to a forensic investigator:

- `/home/<Suspect User's ID>/.mozilla/firefox/<seemingly random characters>.default`: Where Firefox keeps the `places.sqlite`, `formhistory.sqlite`, `cookies.sqlite` and `signons3.txt` files.
- `/home/<Suspect User's ID>/.mozilla /firefox/<seemingly random characters>.default/Cache`: Where Firefox keeps the Cache files.

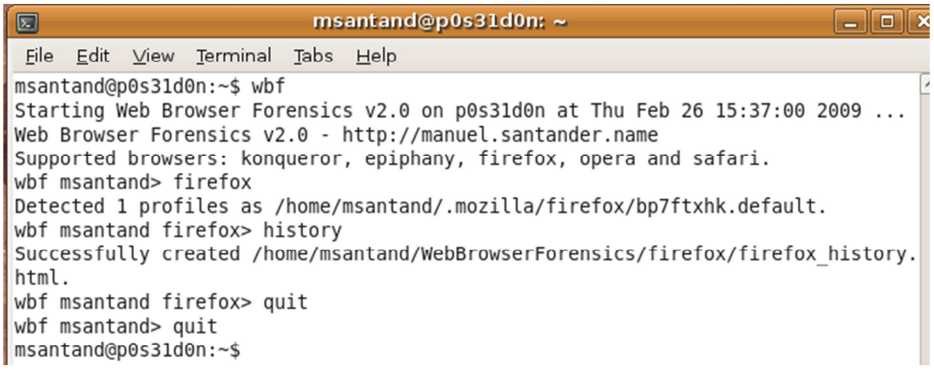
Firefox – Viewing Without Tools



The same principles for viewing forensic information without tools apply to Firefox for Linux / Unix (nice to know there are some constants in the world.) You can view the last 10 sites visited in the History menu or examine the whole navigation history by clicking the History menu and then library, or use <Ctrl>-<h> to get a much more complete listing, or even type “about:cache” in the Address bar if you’re inclined to look at the memory or disk cache records. Just to change things a bit, you need to go to Edit / Preferences / Privacy / Cookies / View Cookies menu to view the cookies directly.

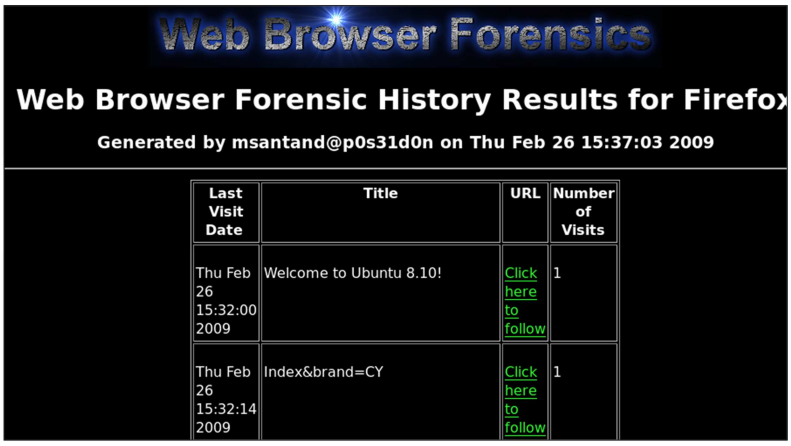
Firefox permits looking inside the history file by date, site, most visited and least visited URL. The history file for firefox browser is written in a sqlite database and needs to be readed using a program that invokes the database API.

WBF Analysis of Firefox History



Wbf is a program written by Manuel Humberto Santander Peláez to extract history information from Firefox. Wbf generates an HTML file in the same directory that the script was run. The file is named `firefox_history.html`, while the page it creates is entitled “Web Browser Forensic Results for Firefox”.

WBF Analysis of Firefox History



Web Browser Forensics

Web Browser Forensic History Results for Firefox

Generated by msantand@p0s31d0n on Thu Feb 26 15:37:03 2009

Last Visit Date	Title	URL	Number of Visits
Thu Feb 26 15:32:00 2009	Welcome to Ubuntu 8.10!	Click here to follow	1
Thu Feb 26 15:32:14 2009	Index&brand=CY	Click here to follow	1

For each URL in the `places.sqlite` file, Wbf provides the last date and time it was accessed, the number of times the URL was accessed, the page title and the URL itself (well, actually, a hyperlink, but you get the URL if you hover the mouse over it and look at the bottom of the browser window) in columnar format. Wbf is executed by issuing one of the following commands:

- Type *wbf*
- Type *firefox* to select the browser type. All the input for the program will be done from ~/.mozilla/firefox.
- Select history
- Exit the program typing *quit* twice.

Hey, remember the advice we gave you on importing HTML files into a spreadsheet program so you can sort them more easily? Yup, it applies here too. ☺

License Notice