SANS

www.sans.org

**FORENSICS 408**

COMPUTER FORENSIC ESSENTIALS

# 408.3-408.6

# Core Windows Forensics & Digital Forensic Challenge and Mock Trial

*The right security training for your staff, at the right time, in the right location.*

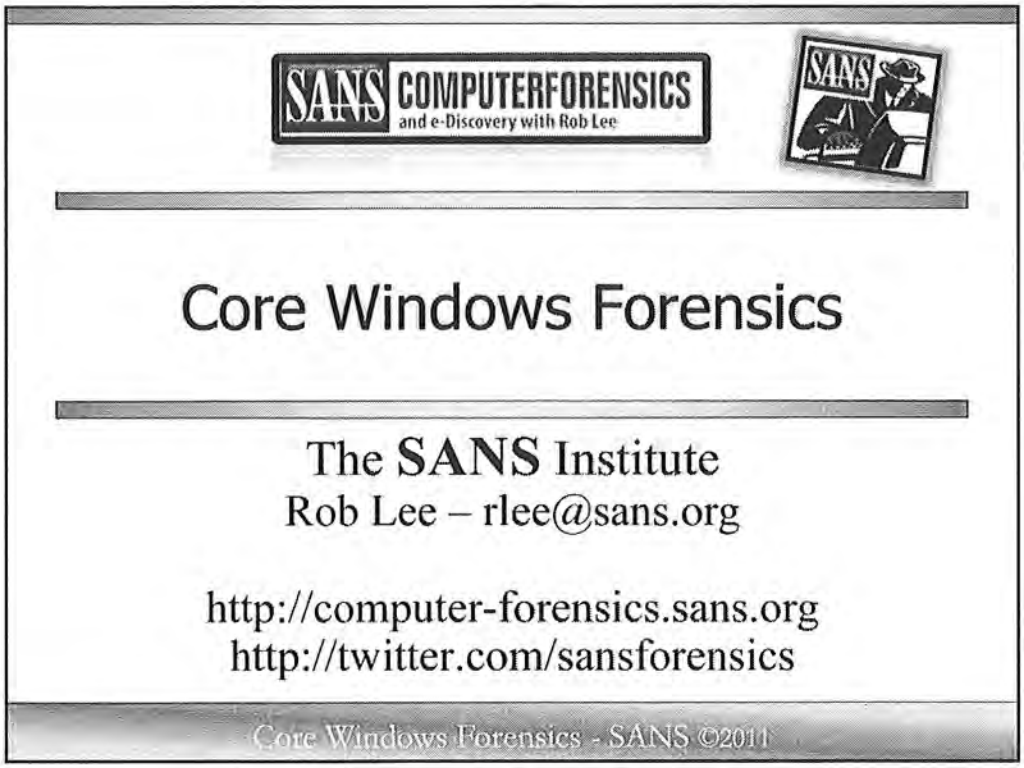
Copyright © 2011, The SANS Institute. All rights reserved. The entire contents of this publication are the property of the SANS Institute.

IMPORTANT-READ CAREFULLY:

This Courseware License Agreement ("CLA") is a legal agreement between you (either an individual or a single entity; henceforth User) and the SANS Institute for the personal, non-transferable use of this courseware. User agrees that the CLA is the complete and exclusive statement of agreement between The SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA. If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this courseware. BY ACCEPTING THIS COURSEWARE YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. IF YOU DO NOT AGREE YOU MAY RETURN IT TO THE SANS INSTITUTE FOR A FULL REFUND, IF APPLICABLE. The SANS Institute hereby grants User a non-exclusive license to use the material contained in this courseware subject to the terms of this agreement. User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of this publication in any medium whether printed, electronic or otherwise, for any purpose without the express written consent of the SANS Institute. Additionally, user may not sell, rent, lease, trade, or otherwise transfer the courseware in any way, shape, or form without the express written consent of the SANS Institute.

The SANS Institute reserves the right to terminate the above lease at any time. Upon termination of the lease, user is obligated to return all materials covered by the lease within a reasonable amount of time.

V2011_0228




Welcome to Core Windows Forensics.

Rob Lee
rlee@sans.org
http://twitter.com/robtlee
http://twitter.com/sansforensics

Special thanks to Chad Tilbury for his work on this day in helping create SEC408 Computer Forensics and E-Discovery Essentials in tech review and helping with slides for Browser Forensics and E-mail Forensics.

[redacted]

## Cell Phones and Pagers...

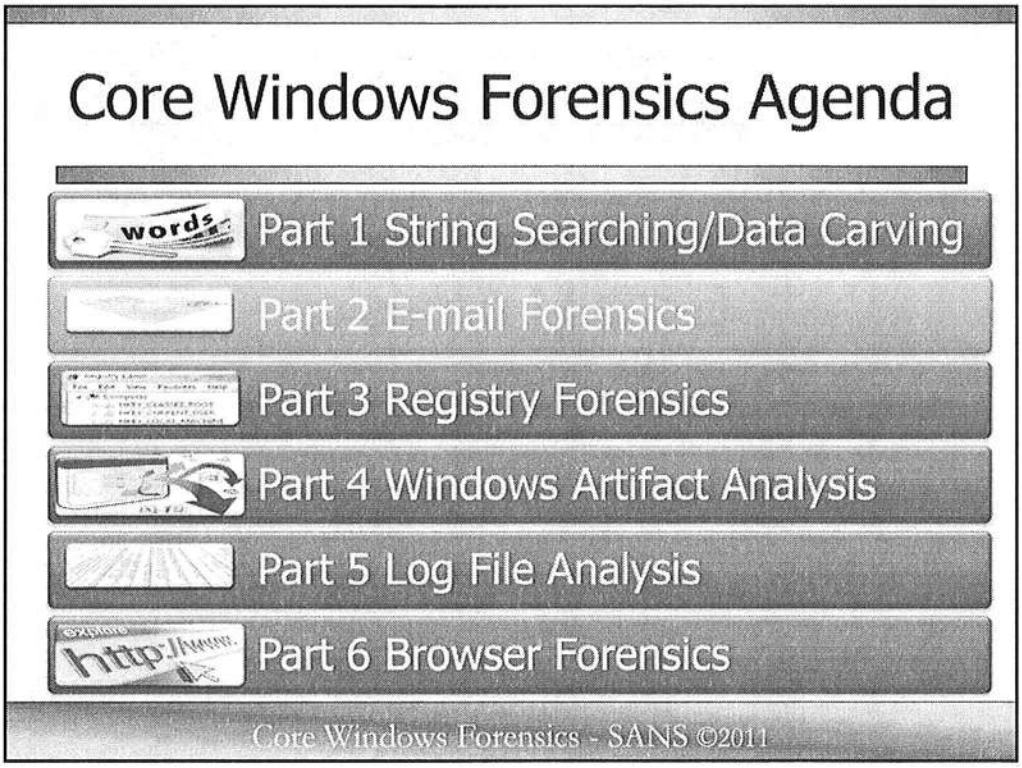
The high rate of slide delivery means that distractions will cause your fellow students to miss material. If you are a "high interrupt" person, please consider moving to the back of the room or disabling your pagers and phones.

Thank you.

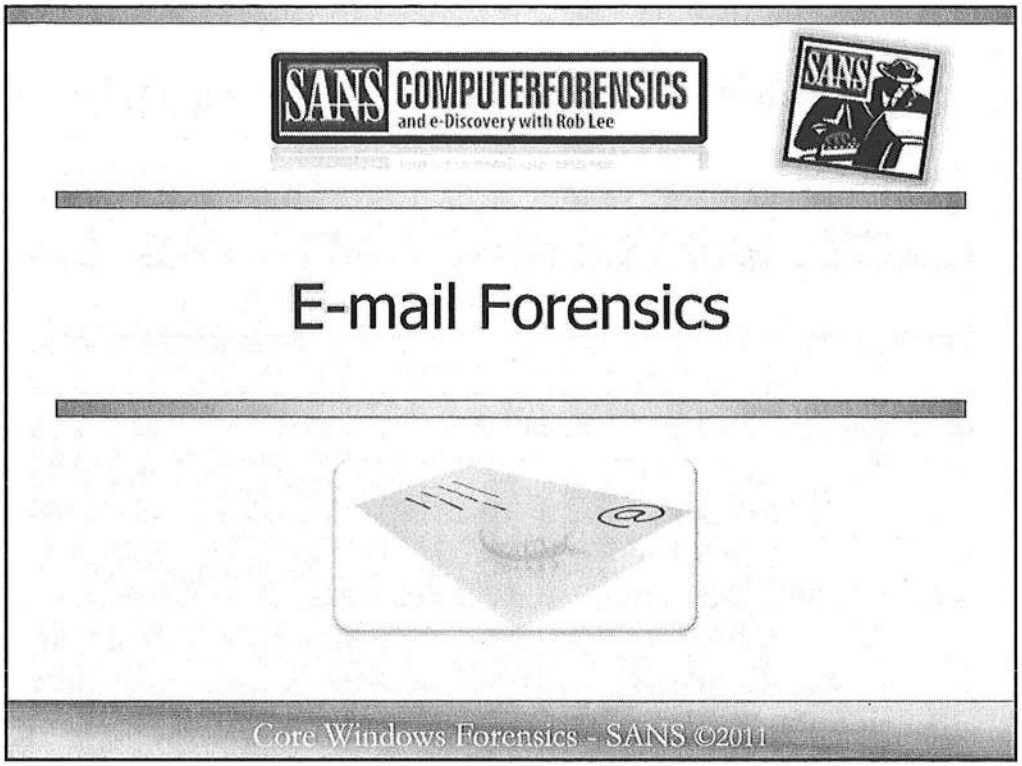


Please know that I understand the need to stay "connected", especially in the positions that we hold. However, we will be covering quite a bit of material today, and cell phone/pager interruptions do cause your neighbor to miss material. Most pagers and cell phones have a "vibrate" setting. Please use this feature during your class time at SANS. If you must answer an important call, the best way to do it is to quietly go out into the hallway and conduct business there.

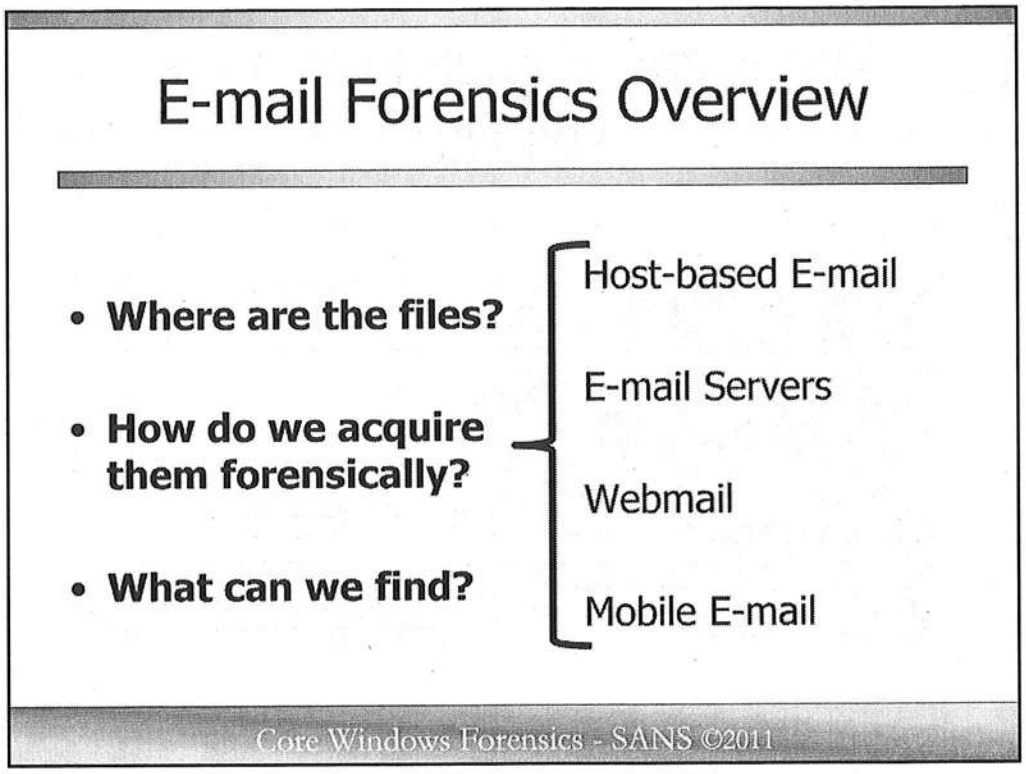
I thank you, and the person sitting next to you does too! ☺




This page intentionally left blank.

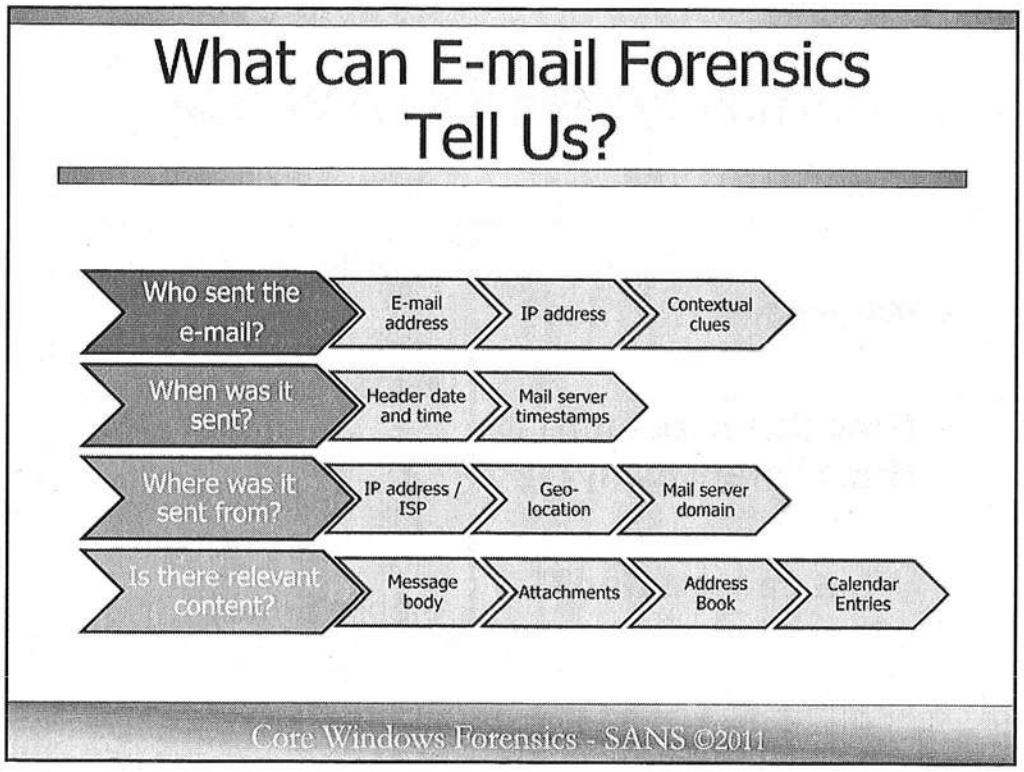



This page intentionally left blank.




Depending on the type of investigation and authorization, a wealth of evidence can be unearthed through the analysis of e-mail files. Recovered e-mail can bring excellent corroborating information to an investigation, and its informality often provides very incriminating evidence. When conducting forensic reviews, and particularly in the E-Discovery world, an investigator must be savvy about the various locations where e-mail evidence can exist. It is common for a user to have e-mail that exists locally on their workstation, on their company e-mail server, and in multiple webmail accounts.

In this section we will be discussing what types of information can be relevant to an investigation, where to find e-mail files, and how to use forensic tools to facilitate the analysis process. We will find that the analysis process is similar across different types of e-mail stores, but the real work takes place in the preparation - finding and extracting the e-mail files from a variety of different sources.




For such a simple protocol, e-mail can provide the examiner a wealth of information to aid an investigation. When conducting e-mail examinations we focus on answering four questions:

**Who sent the e-mail?**

Computers don't generate e-mail autonomously and when reviewing e-mails we will spend a considerable amount of time attempting to identify the originator of a message. This isn't always as easy as it sounds, particularly taking into account the ease with which e-mail communications can be faked. However, we do often have some good leads to follow, including the originating e-mail address, the IP address of the originating computer, and whatever contextual clues we can pick up from the message itself (such as signature block, textual anomalies, references, etc.).

**When was it sent?**

We can usually determine when a message was sent with a high degree of accuracy. The message itself will have a timestamp associated with it, but more importantly we will be focusing on the mail server timestamps which have been added to the message. These are more difficult to fake and we often have logs and some control over at least one of those servers in an enterprise environment.

**Where was it sent from?**

Determining where a message was sent from can be difficult, but we do have some good places to look. If it exists, the originating IP address can be very helpful, particularly when paired with geo-location databases which can pinpoint the physical location of that IP address. Tracking the domains and IP addresses of the various ISPs and mail servers referenced in the message header can also allow us to narrow down on a location, and at a minimum give us a starting location to issue subpoenas for ISP records.

**Is there relevant content?**

This is usually where an investigator will start. After all, if the message isn't useful to the investigation, what does it matter where it was sent from? We will see in this section that e-mail evidence can provide an amazing amount of information, even beyond the messages and attachments we will find in e-mail stores. The contact information, calendar appointments and task lists we find during e-mail examinations can be extremely helpful in tying the case together.

# Creating an E-mail

***HELO mycompany.com***
250 mail.mycompany.com Hello mycompany.com [127.0.0.1], pleased to meet you

***MAIL FROM: sender@mycompany.com***
250 sender@mycompany.com... Sender ok

***RCPT TO: ima@sobusted.com***
250 ima@sobusted.com... Recipient ok (will queue)

***DATA***
354 Enter mail, end with "." on a line by itself
***Subject: Smoking Gun***
***Here is everything we discussed.***
.
250 GDA17049 Message accepted for delivery

Core Windows Forensics - SANS ©2011

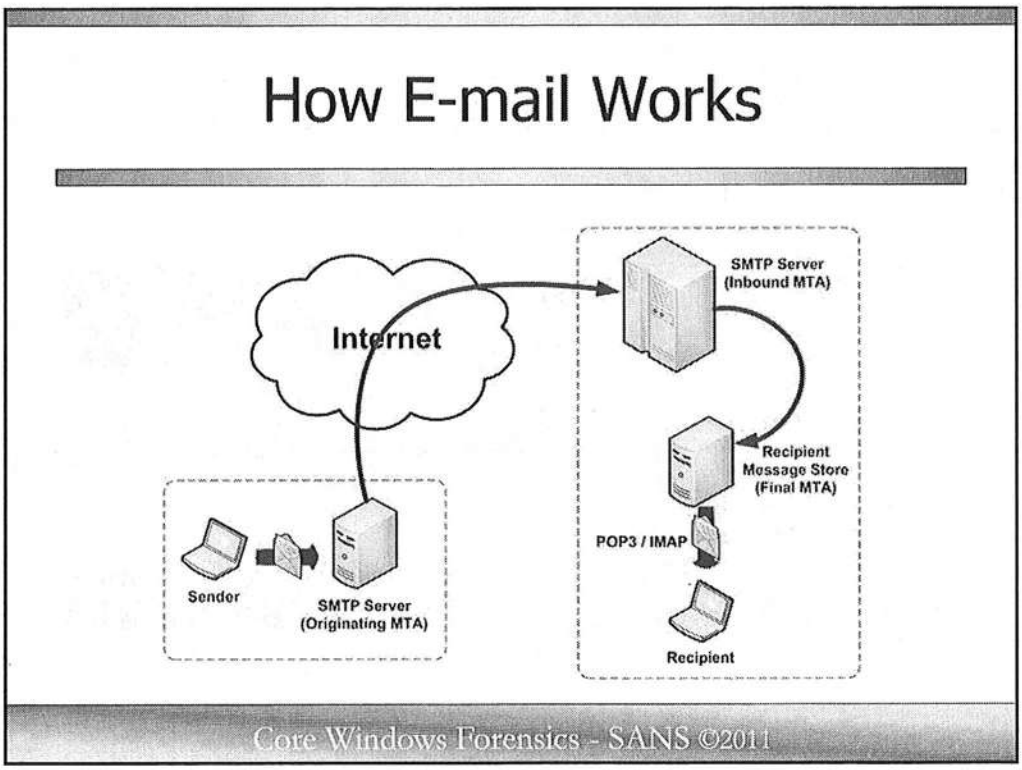
Creating and sending an e-mail is a simple process. As shown here, a series of four commands were used to access the mail server (`HELO`), indicate the sender (`MAIL FROM`), identify the recipient (`RCPT TO`), and create the body of the message (`DATA`). Information typed by the user is shown on the slide in bold italics. These commands are defined in the Simple Mail Transport Protocol (SMTP) Internet standard. [1] Most users are unfamiliar with them because modern e-mail clients initiate this process behind the scenes.

The simplicity of the e-mail specification comes at a cost. There is no authentication mechanism to verify information provided by the sender, and thus messages can easily be forged. This is called e-mail spoofing, and in most cases only information added by the mail servers involved in the transaction (such as IP address) can be trusted. Taking into account that the originating mail server could be owned or commandeered by the forger, the only information that can be fully trusted is data coming from a mail server that you control.

The owners of mail servers have taken steps to reduce the likelihood of spoofed e-mail. One of the key steps has been the vast reduction in the number of "open relays", or servers that allowed anyone to access and send mail through them. However, the steady growth of SPAM messages proves that existing mechanisms are only a hindrance to a persistent adversary. Several e-mail authentication schemes are in development, but we are likely years away from a mass adopted solution.

**References**

[1] http://tools.ietf.org/html/rfc5321

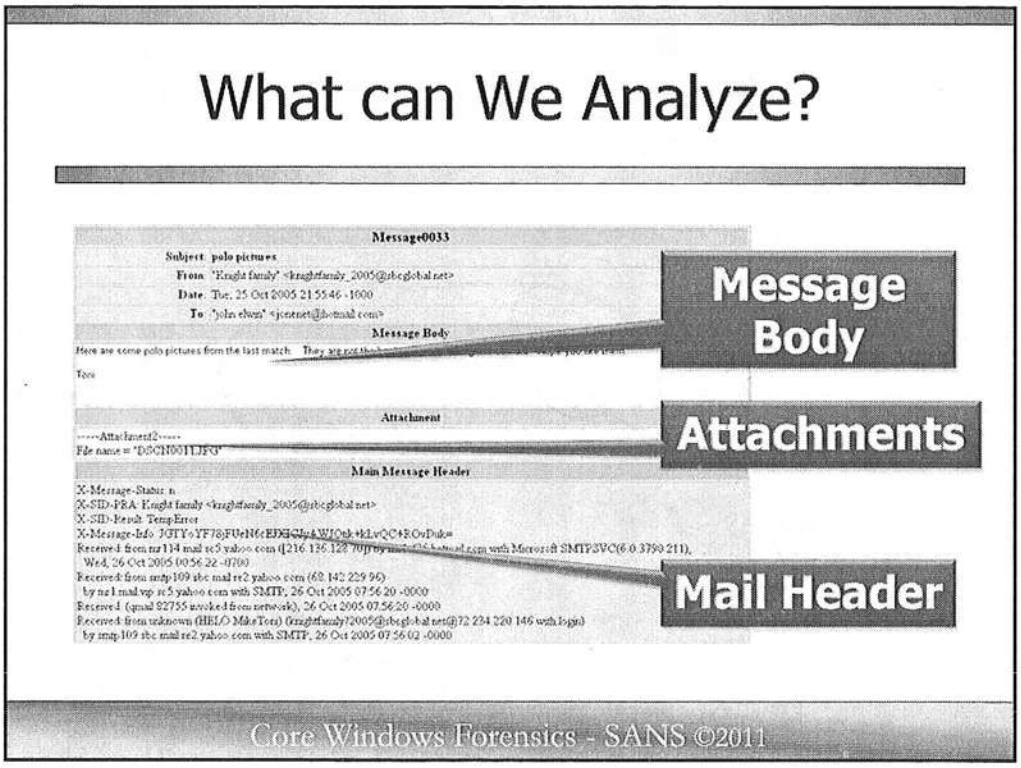



By studying how an e-mail message is sent over the Internet, we can gain insight into the type of meta-data information we should expect to find surrounding an e-mail of interest.

An e-mail is composed and sent using a mail client, which is usually either local to the host computer (such as Outlook) or a web-based application like Yahoo! Mail. The client sends the message to a Mail Transfer Agent (MTA), which is a mail server running the Simple Mail Transfer Protocol (SMTP). The MTA then locates the advertised mail server for the recipient and passes the message along. Given the complexity of modern enterprise environments, a message can be passed through several MTAs before it reaches the mail server belonging to the recipient. Each MTA a message passes through adds a timestamp to the message. When we analyze e-mail headers, this timestamp will be one of our key sources of information. An example timestamp follows:

```
Received: from smtp109.sbc.mail.re2.yahoo.com (68.142.229.96) by
mir1.mail.vip.sc5.yahoo.com with SMTP, 26 Oct 2005 07:56:20 -0000
```

In the final step of the process, the recipient accesses their mail server using a protocol such as POP3 or IMAP and downloads the message to their e-mail client.




E-mail has a relatively simple file structure, giving us three basic components to analyze. This slide shows the way an e-mail message is displayed using Access Data's Forensic Toolkit (FTK). The standard e-mail client configuration hides the majority of the mail header information from the user. While most clients have the option to display header information, a dedicated e-mail analysis tool simplifies the process.

The message body is the text included by the sender. Attachments are optional, and include any files appended to the e-mail message. The mail header is the "envelope" used by mail clients and servers to send and receive the e-mail. Each of these components will be covered in their own slide.

## Message Body

- Typed by sender
- How do we analyze?
  - Manual review
  - String searches
- Can be stored as binary data
  - Forensic tool must understand format



Analysis of the e-mail message body is relatively straightforward. Unlike e-mail headers where the mail client and server add information, the message body is 100% created by the sender.

The easiest way to analyze e-mail messages is via manual review. This entails using an appropriate e-mail client (or forensic tool) to manually read each message. When working with foreign languages, care should be taken to use a client that can parse unicode characters. If you are working with a large number of messages, string searches and de-duplication can be used as data reduction methods. A keyword list is created and a forensic tool is used to search each message (and sometimes the entire image file) for instances of those keywords. Any duplicate e-mails are removed from the review pool. Once data reduction is complete, a manual review is conducted on any keyword matches.

A key point to remember is that while e-mail messages are always text based, e-mail clients/servers can store them as binary data. Thus it is critical to use a forensic tool that is capable of native searching within the e-mail database – a simple text based search of the entire drive will fail to find matches within binary e-mail files.

## Attachments

- Up to 80% of all e-mail data
  - Difficult to manage during analysis
- E-mail standard only allows text
  - Attachments must be encoded
  - MIME / base64 format
- Frequent vector for viruses!
  - Scan all attachments before reviewing



Unsurprisingly, e-mail attachments are a key source of information during forensic analysis. The e-mail industry estimates that 80% of e-mail data is stored via attachments. Thus, you should expect to spend 80% of your analysis time reviewing them. Attachments can be cumbersome to catalog and review, particularly individually. In some cases, specialized viewers or applications will be necessary to open files. Some forensic tools allow you to export all attachments for later review, but it can be difficult to match an attachment of interest with its original e-mail message.

When reviewing raw e-mail messages (i.e. looking at a message as it was transmitted over a network), it is important to remember that the e-mail standard only allows ASCII text to be transmitted. Thus, attachments must be encoded (typically in MIME/base64 format). In this format, attachments will look like a long line of garbled characters. They can often be unencoded manually (using a base64 conversion tool), or an e-mail client can be used.

Attachments are a leading cause of virus infections, so it is important to treat them with care. Scanning attachments for viruses is critical. If the attachments have been exported, then a simple scan of the directory is sufficient. Otherwise, you can install virus software on your analysis machine with the correct e-mail client plug-in.

# E-mail Headers

**Main Message Header**

```
X-Message-Status: n
X-SID-PRA: john elvin <jonenet@hotmail.com>
X-SID-Result: Pass
X-Message-Info: JGTYoYF70jHY9Amuhd9gkO1odYrG0jca75CVz15j+4c=
Received: from mc12-f24.hotmail.com ([65.54.167.160]) by imc1-s37.hotmail.com with Microsoft SMTPSVC(6.0.3790.211);
        Thu, 27 Oct 2005 16:10:32 -0700
Received: from hotmail.com ([65.54.174.47]) by mc13-f24.hotmail.com with Microsoft SMTPSVC(6.0.3790.211);
        Thu, 27 Oct 2005 16:10:31 -0700
Received: from mail pickup service by hotmail.com with Microsoft SMTPSVC;
        Thu, 27 Oct 2005 16:10:31 -0700
Message-ID: <BAY103-F372925368BADB3DFF5D770BE680@phx.gbl>
Received: from 65.54.174.200 by by103fd.bay103.hotmail.msn.com with HTTP;
        Thu, 27 Oct 2005 23:10:31 GMT
X-Originating-Email: [jonenet@hotmail.com]
X-Sender: jonenet@hotmail.com
X-Originating-IP: [72.234.28.253]
From: "john elvin" <jonenet@hotmail.com>
To: jonenet@hotmail.com
Bcc:
Subject: Chinese Astrology:Compatibility chart
Date: Thu, 27 Oct 2005 13:10:31 -1000
Mime-Version: 1.0
Content-Type: text/html; format=flowed
X-Mailer: Microsoft Office Outlook, Build 11.0.5510
X-OriginalArrivalTime: 27 Oct 2005 23:10:31.0469 (UTC) FILETIME=[BEC7CDD0:01C5DB4C]
Return-Path: jonenet@hotmail.com
```

Core Windows Forensics - SANS ©2011

The e-mail header is a required component of every e-mail and provides the "envelope" that a message relies upon for getting it to the proper destination. The header can be thought of as a transit ticket of sorts. It can tell you who made the booking (sender), the destination (recipient), the departure date and time (when the message was sent), what route the message took (the servers it travelled through), and the arrival date and time (last server that received the message).[1]

There is a wealth of information provided by the header, but as investigators we will focus on data that can tie the message to a computer system. We do need to be careful, because much of the header information can be spoofed (faked). The only completely reliable information comes from Mail Transfer Agents (MTAs) that you own or trust (via the "`Received`" value). While it is important to be aware of spoofing, it is rare and most often seen in SPAM investigations. An investigator can often find valuable clues even within a forged message. Besides the client issued values, "From:" "To:" "CC:" "BCC:" "Subject", and "Date", four additional header values an investigator should focus on are:

**Message-ID:** Provided by the originating mail server and consists of a unique identifier appended to the server name with an '@' symbol. It is similar to a tracking number for the message and is logged by receiving mail servers. A search of mail server logs for the Message-ID will provide evidence of the e-mail passing through.

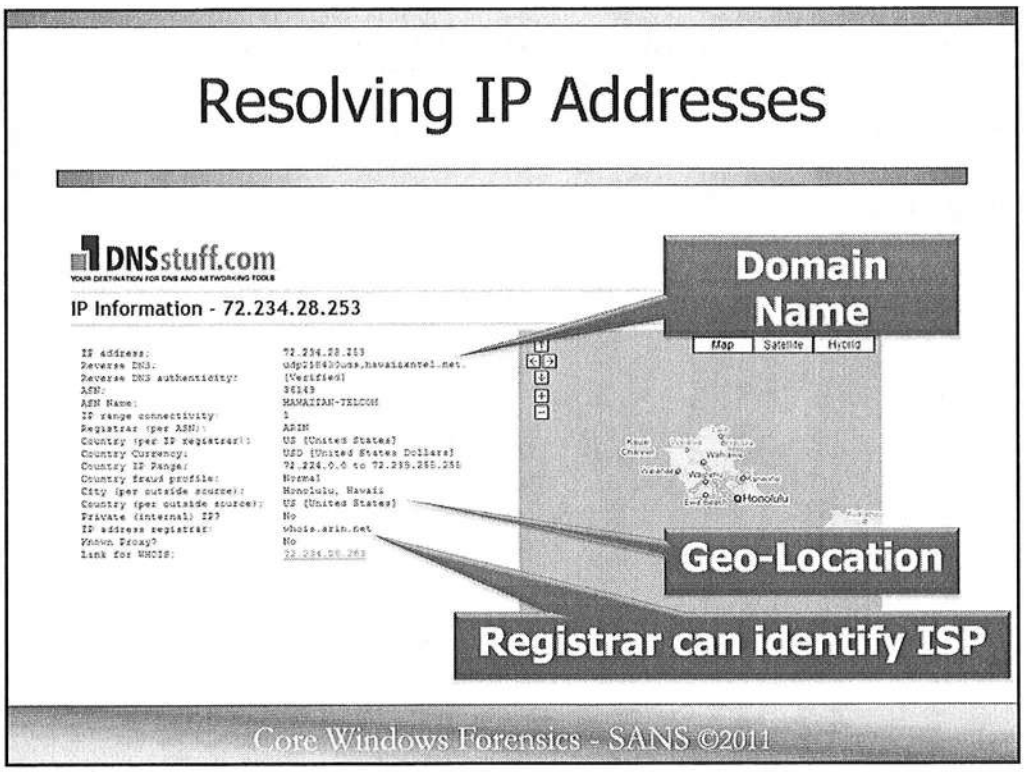
**Received:** Most reliable of the header values. You can trace the path a message took by reading the "Received" entries starting from the bottom-most entry (the originating mail server). Includes the server IP address, server name, date, time, and time zone.

**X-Originating-IP (also X-IP):** An optional tag that identifies the IP address of the computer used to send the original message. Can be forged, but requires control of the originating e-mail MTA.

**X-Mailer:** Identifies the e-mail client used to create the e-mail message. It's inclusion is optional and it is a good indicator of whether the e-mail was created locally or via web-based client.

**References**

[1] http://www.ietf.org/rfc/rfc5322.txt




Once we have an originating IP address, what can we do with it? All IP addresses are unique and they all belong to a registered owner. There is a wealth of information that can be found by querying the publicly available databases which hold this registration information.

During e-mail investigations, resolving an IP address to its domain name can be helpful to double check the information you find within the mail header. Many mail servers will do this for you automatically by adding reverse DNS lookup information into the header. In some cases, resolving an originating IP address (from the `x-originating IP` tag) can provide additional information about who sent the message.

Geo-Location of IP addresses is made possible by databases which track who owns the IP address blocks and where they are located. This is not a perfect science, and various database owners claim accuracy rates of between 80-97%. While they aren't perfect, they can often identify the general area where an e-mail originated from. The author is aware of at least one case where IP address Geo-Location was used to force a settlement – without needing to go the extra step of subpoenaing the ISP to resolve the IP to a specific customer (who would have almost certainly been the defendant).

The Whois Database is like the white pages for the Internet. It contains ownership and contact information for every assigned IP address and domain name. This is particularly important for an investigator who is interested in serving a preservation letter or subpoena for e-mail records. IP addresses are assigned in blocks, so a query accomplished on one IP will return information for the Internet Service Provider (ISP) controlling that specific IP block. Regional registries, such as ARIN, APNIC, and RIPE, collect this information, and most web-based tools identify the proper registry to query behind the scenes. It is important to note that information provided through the Whois service is not always correct, but legitimate ISPs usually keep their information up to date.

A variety of free web-based tools can make collecting this information very easy. The information on this slide was collected using a tool called DNSstuff.com. [1] Similar tools are SamSpade.org and DomainTools, and there are many more to choose from. [2,3]

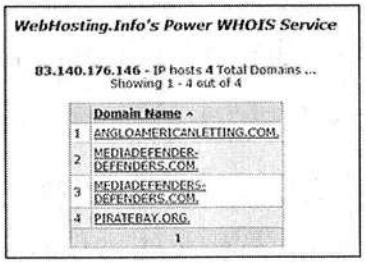
[1] http://www.dnsstuff.com

[2] http://samspade.org

[3] http://www.domaintools.com

# Using Reverse IP

- Use reverse IP lookups to find other domains hosted at same IP address
  - Identify other sites related to your investigation
  - Phishing and scam sites are often hosted together
  - Assume that most domain information is fake!




**http://whois.webhosting.info**



Reverse IP is a little used tool that can be quite helpful in identifying additional entities, which could be relevant to an investigation. It essentially queries the Whois database, and returns all domain names which are listed as being hosted on the IP address provided. Domain names are cheap, and individuals engaged in phishing or other types of scams will often register a large number of domains and host them on the same server. Thus, by identifying one suspect IP address, you can often find several others which would be useful to search for in your forensic investigation.

Be aware that shared hosting of domains is popular, and in some cases, thousands of domain names are associated with a single IP address. In most cases, it is very obvious as to whether the sites are related. In less clear cases, domain name lookups can be performed on the results to determine if the contact information matches. In the example shown in this slide, two of the other domain names redirect to the main site, piratebay.org. [1]

There are few checks on the validity of domain information entered when registering them, but a careful examination can often pick up good clues to tie things back to an individual. This is particularly true when you can identify multiple domains owned by the same individual. There is often an e-mail address or billing contact which goes back to a real person.

**References**

[1] http://whois. webhosting.info

## Host-based E-mail

- E-mail stored on the local machine
- Identify all e-mail storage locations
  - Uses Index and message files
  - Find via filetype searches
  - Using e-mail client configuration info
- Potential for password protection
- Search for deleted e-mail archives!



Host-based e-mail stores can be defined as any e-mail archive stored locally on a computer, independent of an e-mail server. Host-based e-mail typically employs an index file that acts as a table of contents and stores metadata such as read status, flags, replied/forwarded information, and whether the message has been marked for deletion. A separate message store houses the e-mail messages themselves. Both files are necessary to accurately parse and review e-mail the e-mail archive.

Local e-mail archives are almost always associated with an installed e-mail client, giving us a clue of what to look for by reviewing a system's installed applications. They can also be found by file extension searching and reviewing e-mail file configurations. Forensic suites often discover well-known e-mail archives automatically.

Regardless of whether server-based e-mail is employed in the enterprise, workstations can be a great source to find e-mail. IT departments often set mailbox size limits, requiring users to locally archive messages when the quota is reached. It is also possible to find backup mail or contact lists imported from other systems. Deleted e-mail stores are frequently found and recovered during investigations.

# Microsoft Outlook

## File extension: .PST

- Archive stored by default in:
  - `%USERPROFILE%\Local Settings\Application Data\Microsoft\Outlook` (WinXP and prior)
  - `%USERPROFILE%\AppData\Local\Microsoft\Outlook` (Vista)
- Registry key tells what archives are being used: `HKEY_CURRENT_USER\Software\Microsoft\WindowsNT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook`
- Encryption/obfuscation enabled by default
- Archive size can be up to 20GB



The Outlook .PST file is a single repository for e-mail, folders, attachments, contacts, and calendar items. If in use on a system, it is one-stop shopping for an investigator. With newer versions, the PST size can be up to 20GB (prior to Outlook 2003 there was a 2GB limit), providing a wealth of data to review. [1] The default archive is named "`Outlook.pst`", but the name is user configurable and multiple .PST files often exist on systems due to backups or auto-archiving.

Deleted messages may be present in the .pst file, but specialized software (like the FTK forensic suite) is necessary to scan and recover messages from the archive.

The HKEY_CURRENT_USER registry configuration indicates what local archives are in use on the system as well as which MS Exchange server is connected to by the mail client.

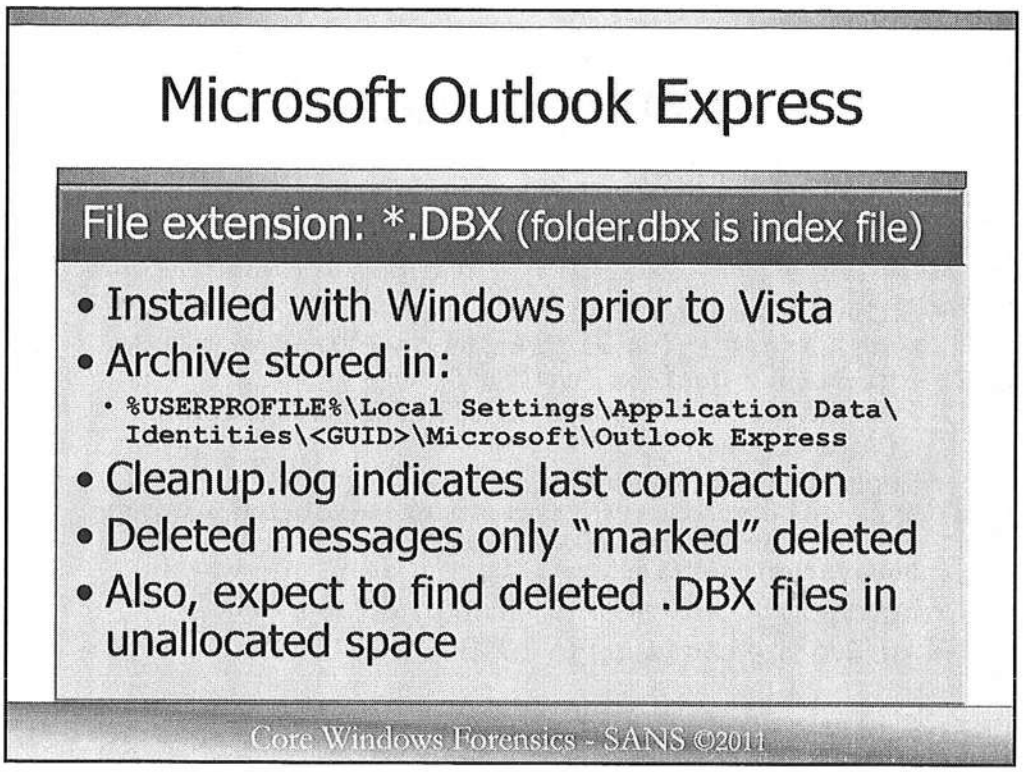
When created, the encryption level of the archive is selected. There are three settings. [2] "No encryption" stores the archive in such a way that it can be read by a text/hex editor. "Compressible encryption" is the default and obfuscates the contents so no clear text is available. It uses a well known, simple cipher and is weak protection at best. However, given that it is the default setting, it should be noted that it will defeat keyword matching efforts by bit-level string search tools. "High encryption" is the strongest mode and employs an encryption algorithm (though the algorithm can be easily defeated). Finally, users have the option of password protecting their archive. This password can be cracked using programs like PstPassword. [3]

**References**

[1] http://support.microsoft.com/kb/832925

[2] http://support.microsoft.com/kb/829971

[3] http://www.nirsoft.net/utils/pst_password.html




Unlike Outlook, which is oriented towards business users, Outlook Express was built for standalone configurations and is installed on the vast majority of machines running Windows (prior to MS Vista). Messages are stored in .DBX files, with a separate file created for each folder used within the client. The index file is named "folder.dbx" and Outlook Express will not operate without it. The .DBX format stores messages in plain text, meaning bit-level string search tools can find keyword matches within the files. Outlook express v4.x (bundled with Windows 98) used the .IDX format for index files and .MBX for messages. [1]

Deleted messages are kept in the .DBX archive and simply marked as deleted. [2] Deleted messages are removed when the archive is compacted (cleaned up and compressed). A file named cleanup.log is located in the same directory as the outlook express files and tells us when the last time the archive was compacted. The longer it has been since compaction, the better your chances will be of finding deleted e-mails in the archive. More recent Outlook Express versions create a copy of the .DBX files in the Recycle Bin before running the compaction process (to insure against archive corruption). Thus it is very common to find (and recover) entire .DBX archives in unallocated space.

Outlook express was replaced in Microsoft Vista operating systems by the new Windows Mail application.

**References**

[1] http://www.microsoft.com/technet/archive/ie/reskit/ie4/Part5/part5a.mspx?mfr=true

[2] http://oedbx.aroh.de/index.html

# Windows Mail

**File extension: *.EML (.FOL is index file)**

- Currently installs with Windows Vista
- Archive stored in:
  - `%USERPROFILE%\AppData\Local\Microsoft\Windows Mail`
- Deleted mail stored in the "Deleted Items" folder in the local store
  - Stored as individual .EML files and not cleared by default



Windows Mail currently appears exclusively in Windows Vista as a replacement for Outlook Express. The functions are similar, but there are major differences in the location of messages and how they are stored.

Instead of using an e-mail database archive, messages are stored individually using the .EML extension. This solves much of the database corruption issues that plagued the Outlook Express format. Folders and index information is stored in .FOL files, with one created for each folder being used in the client. E-mail messages in the .EML format are completely text based, making searching for keywords simple. All critical information – the header, body and attachments – are stored within the individual .EML file.

Since messages are stored individually, accessing deleted messages is also easier . Deleted mail is simply moved to the `Deleted Items` folder in the local store (under the Windows Mail directory). While Windows Mail does perform compaction (the user is prompted to do so every 100 times the application is opened by default), only the index file is actually compacted , which only removes references to deleted files, not the files themselves. Users can manually delete files out of the Deleted Items folder, or set an option within Windows Mail to clear the directory periodically. Knowing this, we should expect to find deleted .EML files in unallocated space on systems using Windows Mail.

Windows Live Mail looks to be the eventual replacement for Windows Mail (and Outlook Express). It also uses the .EML format for storing messages locally. [1]

References

[1] http://www.windowslive.com/connect/post/LiveUnplugged.spaces.live.com-Blog-cns!F92775FC46A390CA!319.entry#

## Mozilla Thunderbird

**File extension: <none> (.MSF for index)**

- Archive stored in:
  - `%USERPROFILE%\Application Data\Thunderbird\Profiles\<random>.default\Mail\`
- Compaction performed manually
- Folders must be smaller than 4GB
- Archives can be difficult to find
- Auto-compaction turned off by default

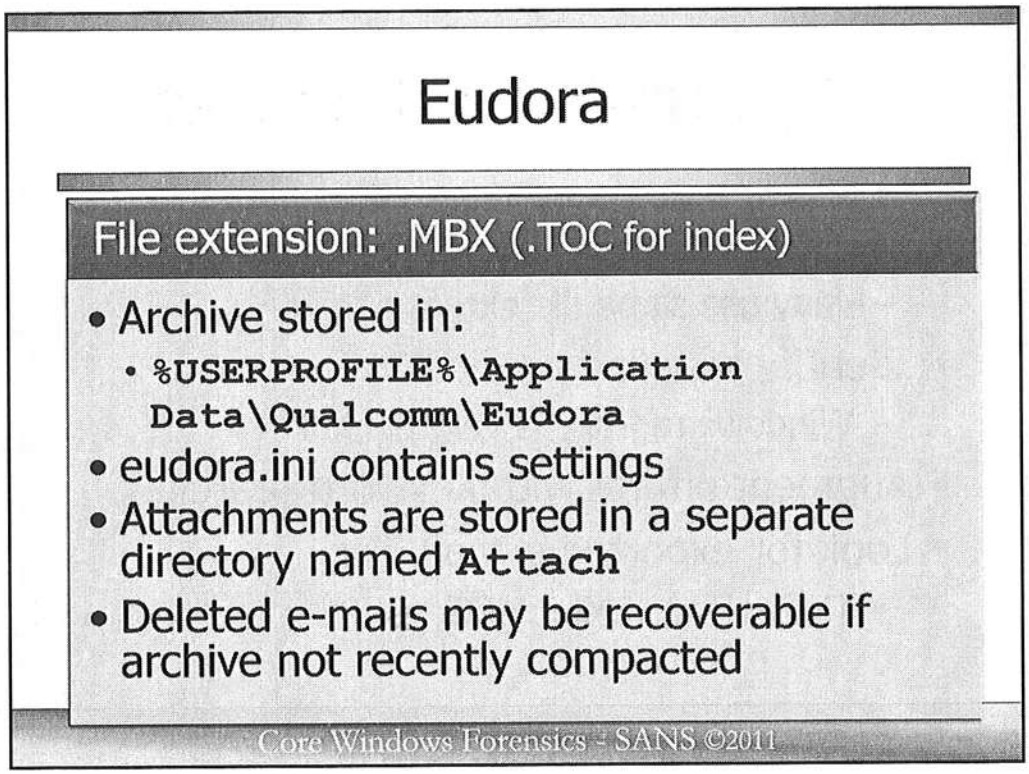
Core Windows Forensics - SANS ©2011

Thunderbird is an open source mail client that works on a variety of computer platforms, including Windows, Apple OS X, and Linux. Thunderbird message archives do not have an extension. This can make searching for archives based on filename difficult, particularly when multiple e-mail accounts are setup within the client. Since we do not have registry information, local stores can be determined by reviewing the Local Directory value in Tools -> Account Settings -> Server Settings in the client. Alternatively, the location of archives can be found in the *prefs.js* file in the local store directory. The archive itself is a flat text file, making it susceptible to keyword searching tools. Index files are given the .MSF extension and correspond to each folder in the e-mail store. The presence of .MSF files should lead the investigator to the files containing the message archive.

Similar to other e-mail clients utilizing message archives, deleted messages are not removed from the database. Instead, their status is simply changed by being marked as deleted. Compaction is required to fully remove deleted e-mails and it is turned off by default. [1] After compaction, e-mail fragments are often still present in unallocated clusters of the file system.

**References**

[1] http://kb.mozillazine.org/Thunderbird_:_Tips_:_Compacting_Folders




Eudora is a classic e-mail client with a long history. It still has a strong following and was recently released back into the open-source community, building upon the Mozilla Thunderbird architecture.

The .MBX file is text based, allowing keyword matches to be found using keyword search tools. The .TOC files are the index files for the various folders. [1] The `eudora.ini` file contains settings such as if multiple e-mail accounts are being accessed by the mail client (called "Personalities" in Eudora) and whether automatic compaction is enabled. The .INI file can be renamed and is required to be present in each e-mail store (for instance if multiple e-mail accounts are managed through the same client).

A unique feature of Eudora is that message attachments are automatically decoded and stored in a separate folder – named "`Attach`" by default (under the primary Eudora directory). This default folder can be moved by the user. The attachment folder for IMAP personalities is located under the respective IMAP folder. This feature makes it easy for an investigator to quickly peruse all attachments. To map attachments to the message the belong to, the messages themselves must be reviewed (Eudora marks the message with where it saved the attachment file).

Deleted e-mails may be present in the .MBX file if it has not been recently compacted. Otherwise, we would expect to find e-mail fragments in the file system unallocated space.

**References**

[1] http://www.eudora.com/techsupport/kb/1602hq.html

## Other Host-Based Formats

- There are a multitude of e-mail clients
  - Many use same file extensions
- Identify installed applications
  - Windows registry is very helpful
- Augment efforts with keyword searches
- Look for exported e-mail files
  - .TXT, .HTM, .EML, .MSG

Core Windows Forensics - SANS ©2011

It would be impossible for us to cover every e-mail client in existence, but some other common e-mail clients are: The Bat!, Poco, Pegasus, FoxMail, Incredimail, and AOL. [1] While the task of identifying every possible e-mail type sounds daunting, there are several steps we can take to make the task easier.

First, we can review all programs that have been installed on the system looking for possible e-mail clients. The Windows registry can be a quick source for this information and can even include references to programs that were uninstalled. If we find an unfamiliar application, an Internet search will often provide information on file types and archive structure.

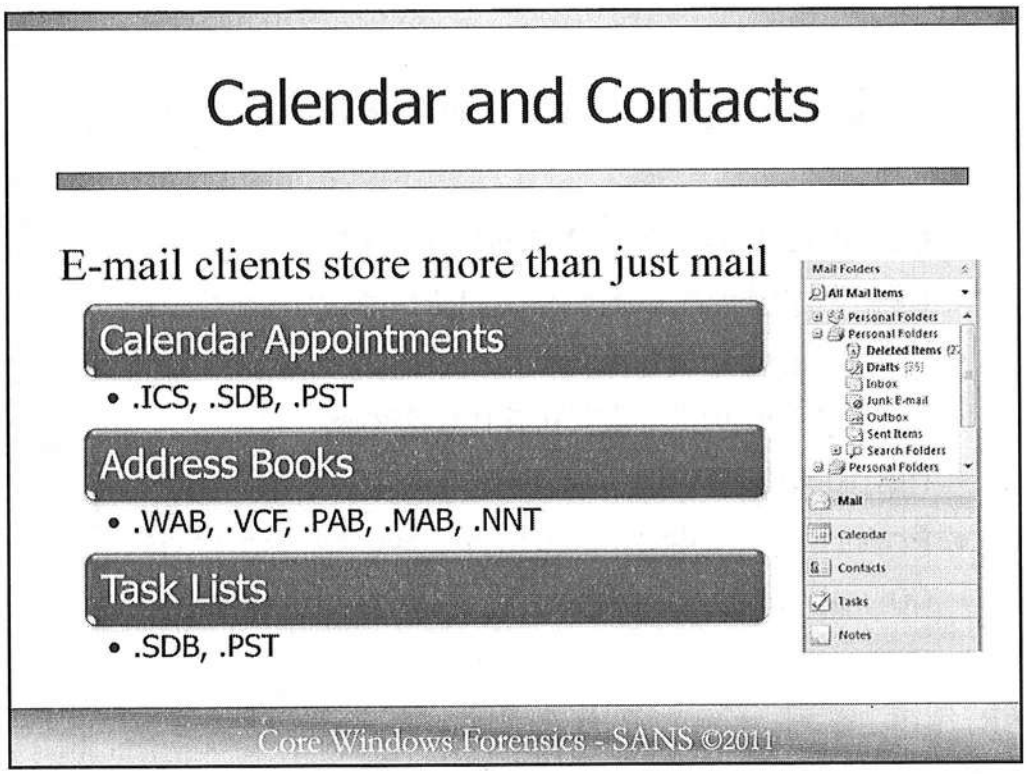
As we saw with most of the host-based clients covered in this course, the majority are utilizing flat text archives to store messages. This means that by performing our bit-level keyword searches, we will often find matches within e-mail files in both allocated and unallocated locations. These matches can then inform our efforts at finding other types of e-mail on the system.

We also need to keep in mind that data from e-mail clients can be exported into other file types. If a user right-clicks on an e-mail in Thunderbird and clicks "save as", that e-mail will be stored on the local system as a .EML file. Thus there could be relevant e-mails on the system outside of the e-mail client archive.

Finally, most of the major forensic suites do a good job of automatically recognizing and parsing the most common e-mail formats, making the investigator's job much easier.

**References**

[1] http://en.wikipedia.org/wiki/Comparison_of_e-mail_clients




Increasingly, e-mail clients are being used for more than just sending and receiving mail. They have become fully functional productivity suites, with appointment scheduling, address books, and task lists (see the slide graphic taken from Microsoft Outlook). Since so much critical information can be stored in these clients, it is recommended that the investigator identify all e-mail clients which have been used on the system and familiarize themselves with the features of each (for instance, Eudora and Outlook Express do not have a calendar feature).

Calendar entries can give a good view into a person's activities, though they can sometimes be challenging to review. Many e-mail clients support importing and exporting of calendar entries in the iCal standard. These files are typically stored in the .ICS format and are often attached to e-mail messages as meeting reminders that can be auto-imported into the client's calendar. Thus it is common to find orphan .ICS files stored in temp directories or in unallocated space. Mozilla Thunderbird/Lightning utilizes the SQLite format for their calendar entries, with a .SDB extension. Finally, outlook .PST files contain e-mail, appointments, contacts, and tasks all in one file.

There are a variety of different address book formats. While some are stored as binary data (like .WAB files in Outlook Express and .PAB files in Outlook), others are largely stored in text format (.VCF "vCard" format in Outlook, .MAB in Thunderbird, and .NNT in Eudora) making them easily searchable. Along those lines, Eudora typically saves contact lists in text format (.TXT), but has a simple virus-protection feature which will rename contact lists to use the .NNT extension when enabled.

Task lists are less commonly found in e-mail clients, but are a rich source of data to be mined. Mozilla Thunderbird/Lightning stores tasks within its calendar files, using the SQLite format and the .SDB extension. The best way to analyze this data is to install the client on your forensic station and import the files for review.

## Corrupted E-mail Archives

- Common in host-based clients
- One solution is to rebuild message store
  - Vendor / third party tools available
  - May eliminate deleted messages
- Manually review files using text editor
- Forensic tools are often very tolerant of corrupted archives

Core Windows Forensics - SANS ©2011

Frequent users of host-based e-mail clients are likely very familiar with corrupted e-mail archives. A simple Internet search reveals hundreds of third party tools available for repairing and recovering e-mail from archives. While virus infections may be the cause of some corruptions, the majority are caused by issues within the client itself: very large or dirty archives, and the index and message files getting out of sync. Outlook Express is famous for users losing mail due to file corruption.

When we encounter a corrupted archive we have a few options. Vendor cleanup tools, like *scanpst.exe*, can be used to reverse corruption, but they come at the price of usually removing any deleted data that may still be resident in the archive through compaction. [1,2] Third party tools do the same, and it can be difficult to trust the application. When using either of these options, make sure to run them only on copies of the evidence and to document which tools were used.

In the worst case scenario, text-based e-mail archives can be reviewed manually using a text or hex editor. The header, body, and attachments should all be available (the latter will need to be decoded), but it is difficult to piece together any e-mail client metadata (like deleted status, flags, replies, etc.) since that information is distributed amongst the index and folder files.

In many cases the best option it to try to access the archive using a compatible forensic suite. Forensic tools are used to accessing the raw data in an archive and thus tend to do a good job of recovering corrupted data.

**References**

[1] http://office.microsoft.com/en-us/outlook/HA010563001033.aspx

[2] http://www.microsoft.com/windows/IE/community/columns/filecorruption.mspx

## Host-Based E-mail Review

- Mail archives composed of multiple files
- Message files often stored as clear text
- Built-in security / encryption is rare
- Deleted messages can be recovered

Core Windows Forensics - SANS ©2011

Nearly all of the e-mail clients covered in this section shared common traits. They used index, message, and folder files to organize the e-mail store. Knowing this, investigators are advised to copy the all of the mail directories when exporting e-mail archives (as opposed to just the message files).

Most host-based e-mail clients store messages in text form, making it easy to use search tools to find archives in unusual spots, and giving another way to review the messages (using a text editor) if the archive is corrupted.

While most e-mail clients utilize passwords, they typically only limit access to the identities located on the client itself. Thus any investigator who knows where to find the archive files can freely analyze them without needing to crack passwords. Mail PassView from NirSoft is often successful at breaking passwords used by most common e-mail clients. [1]

From an investigator's standpoint, the e-mail archive format is a benefit because it provides the opportunity to recover deleted e-mails if compaction has not taken place. It should be kept in mind that e-mail clients are designed to hide messages marked as deleted, and thus an alternate viewer is required to review them. There is also the possibility of recovering entire deleted archives using traditional forensic file recovery techniques.

Microsoft Outlook is the only e-mail client covered that deviates from these commonalities. The Outlook archive is typically composed of only one file (.pst), it includes binary obfuscation by default and offers the choice of encryption. Similar to the others, deleted messages are kept in the archive until some for of compaction is completed.

**References**

[1] http://www.nirsoft.net/utils/mailpv.html

# E-mail Encryption

- Individual message encryption is most commonly found
  - S/MIME and PGP/MIME
- Enterprise clients typically have the most robust mail archive encryption
  - Key servers make recovery feasible
- Network based encryption usually does not hamper forensic efforts



E-mail privacy solutions and standards have come a long way in recent years, greatly increasing their adoption rate. When performing e-mail recovery, the investigator needs to be cognizant of the potential for message encryption. While it is beyond the scope of this course to cover this subject in detail, there are a few basic concepts which describe most of the situations we are likely to encounter in the field.

Individual message encryption using a public-key protocol like Secure MIME (S/MIME) or PGP/MIME is the most common type of encryption encountered. It is an end-to-end protocol (meaning it is not decrypted enroute) and enables users to digitally sign and encrypt messages. The message typically is stored in the e-mail archive as encrypted, and decrypted manually by the user. An encrypted message looks like the following:


```
-----BEGIN PGP MESSAGE-----
Charset: ISO-8859-1
Version: GnuPG v1.4.9
(MingW32hQIOA72S+OZoP/UMEAf+JAeDzUagsWjzh233ewOX0EAq2B5Ok/vRfT8UU7MqAQH
B
...
h/XDlmUM7+kbF0Jws2YvxQX0Rvj3OWDegpQ8etw88YTfPevhPR3/ViQ7gIMk44fe
=rL9S
-----END PGP MESSAGE-----
```

These protocols are very secure and unless a mistake was made by the user (i.e. not password protecting their private key), it is unlikely that they can be cracked by conventional means. Keep in mind that you need participation from both parties to encrypt, so the amount of encrypted e-mail still tends to be small. Investigators should keep an eye out for files with .PGP (PGP) or .P7M (S/MIME) extensions which are good indicators that encrypted information may be present. [1]

While it is rare for server-based e-mail databases to be encrypted, the client software used to access these stores is often feature rich. Thus it is common to find encrypted e-mail within those databases. Most clients have built in support for S/MIME. Both Outlook and Lotus Notes allow users to encrypt any local archives, and Notes even has a built in feature to encrypt all incoming e-mail. [2] Some enterprise environments require S/MIME encryption for internal communications, but they often have a centralized key server, making recovery feasible.

Network based mail encryption using Transport Layer Security (TLS) or Secure Sockets Layer (SSL) only encrypts the message in transit and typically does not affect forensic efforts unless e-mail is being captured off of the wire.

Finally, we should not forget that when doing a workstation analysis, even if encrypted messages are found, they were likely to have been viewed on that system. Thus there is the potential for uncovering remnant data using forensic techniques.

**References**

[1] http://www.file-extensions.org/filetype/extension/name/encoded-and-encrypted-files

[2] http://www.ibm.com/developerworks/lotus/library/ls-Notes_Encryption/

## E-mail Servers

- Most corporate environments employ dedicated mail servers
  - Could be hosted offsite
- Business considerations make getting forensic copies difficult
  - Expect massive amounts of data
- Deleted mail exists, but is less likely to be found

Core Windows Forensics - SANS ©2011

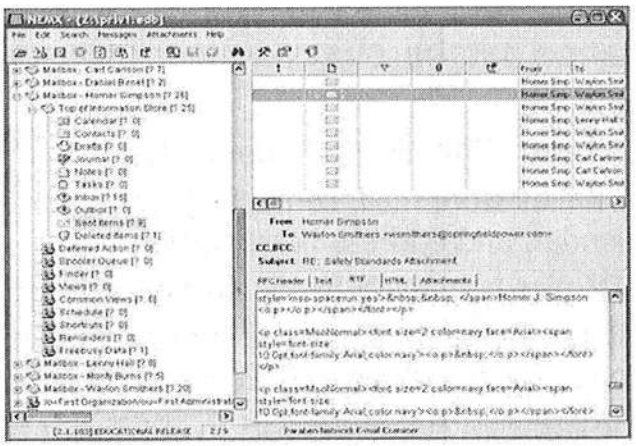
All e-mail, at one time or another, has been located on an e-mail server. The big distinction is whether the e-mail resides on the server, or is instead stored locally on a workstation. In most business environments we see a combination of both – the most recent e-mail traffic is accessed through the company e-mail server, and offline storage on workstations is used to archive old messages or synchronize mailboxes for use outside of the office. Determining where all e-mail resides is the first challenge for the investigator, and it is usually prudent to not give too much credence to polices or system administrator testimony. Very few IT departments have strong controls over workstation data and thus you will frequently find e-mail archives exactly where they were not supposed to have existed.

Getting e-mail evidence from mail servers can be challenging. In the worst case scenario, the server itself could be offsite or operated by a shared hosting firm. Numerous companies exist to host/manage e-mail servers, and small to medium sized companies often find it more cost efficient to outsource their mail infrastructure. Another complicating factor is that mail servers are critical to business operations, and thus, traditional forensic techniques like shutting down the system to get a complete image are often impossible. A forensic examiner usually has four options:

1) Full disk image (difficult to obtain for highly utilized, critical servers except via a live image)
2) Logical or volume level image (performed live)
3) Export e-mail database files (making sure to obtain entire directories to ensure collection of all necessary files)
4) Work from e-mail backups (typically on tape)

Similar to host-based e-mail archives, deleted messages can be found in server-based archives. That being said, they are frequently overwritten due to high reuse and frequent compaction / compression by the server. Thus investigators should expect to find less deleted messages in server-based e-mail stores.

# Forensic Tools for Server E-mail



**Network E-mail Examiner from Paraben Forensics**

Core Windows Forensics - SANS ©2011

Numerous tools exist to make the process of acquiring and analyzing server-based e-mail easier. Many of these tools were spawned by the explosive growth of the E-Discovery industry and hence provide a wealth of features for reviewing massive amounts of e-mail messages.

Pictured here is the Network E-mail Examiner tool from Paraben Forensics. This is one of the lower cost options in the field and supports MS Exchange, Lotus Notes, and Novell GroupWise e-mail stores. Encase v6 can examine MS Exchange .EDB files and Lotus Notes databases.

Server-based e-mail forensic tools provide benefits similar to those we discussed with the standard forensic suites: deleted message recovery, ability to export relevant e-mails, metadata review, searching, and tolerance for corrupted archives. The key difference is the ability for these tools to access the entire message store at once, allowing a simultaneous review/search across multiple user accounts. When coupled with de-duplication technologies, this greatly reduces review time.

# Microsoft Exchange

- Exchange database is container for user mailboxes
  - .EDB and .STM files form database
  - .EDB files store mail, contacts, journal, notes, tasks, calendar, and address book entries
  - .STM files hold multi-media files in native format
  - Database may be broken up into multiple storage groups, each with own .EDB and .STM files
- Mailboxes can be exported in .PST file format



Microsoft Exchange has emerged as the mail server market leader It is very common in corporate enterprises and is almost always run on a standalone server (or virtualized server). The Exchange server can be thought of as a container that holds a large number of individual user mailboxes. Thus it often takes some digging in order to find what you are looking for.

A database container in Exchange is composed of two primary files [1]. The .EDB and .STM files are synchronized and matched, which can cause database corruption if they are improperly acquired. An example database would be: `priv1.edb` *and* `priv1.stm`. Exchange tools run against .edb files typically also include the .stm files transparently.

Files with the .EDB extension form the core of the database and contain the e-mail messages along with their associated headers and some attachments. The associated .STM file contains audio, video and other media attachments formatted as streams of Multipurpose Internet Mail Extensions (MIME) data. While a forensic analysis could be conducted on just the .EDB file, it would be missing many of the associated multi-media attachments.

In order to simplify administration and make backup/recovery operations simpler in large enterprises, newer Exchange databases can be broken up into multiple storage groups, each able to contain multiple database files. When acquiring the Exchange mail store, it is important to work with the server administrator to ensure that you are getting all relevant data. Alternatively, individual mailboxes can be exported in the .PST file format.

[1] http://technet.microsoft.com/en-us/library/aa996118%28EXCHG.65%29.aspx

## Deleted Objects in Exchange

- "Soft" deletes put items in the Deleted Items folder
  - Stay resident until removed from folder
- "Hard" deletes go directly to Exchange Dumpster
  - Using `<shift>+<delete>`
  - Deleted mail from POP3 or IMAP accounts
- Exchange Dumpster
  - By default, retains e-mail for 7 days and accounts for 30 days
  - Exchange 2010 includes indexing and retention of *all* deleted objects
- Online recovery of Dumpster objects done via Outlook client
  - In Outlook, use Tools → Recover Deleted Items
  - `DumpsterAlwaysOn` registry key must be added and set to "1"

Core Windows Forensics - SANS ©2011

Two types of deleted files exist in Exchange databases. Soft deletes occur when a message is deleted by a user and it is placed in the `Deleted Items` folder. Soft deleted files will remain in a users `Deleted Items` folder until it is emptied by the user or via a space-recovery rule set by the administrator. A hard delete occurs by pressing `shift+delete` to delete a message, when a message is removed from the `Deleted Items` folder, or when a message is deleted from an account using POP3 or IMAP. Hard deleted e-mail files are kept in the Exchange Dumpster for seven days by default or whenever the last database backup occurred (whatever is longer) [1]. Thus it may be necessary to review old backups of the server to recover messages hard deleted more than a week prior. Deleted user accounts are retained in the database for thirty days by default.

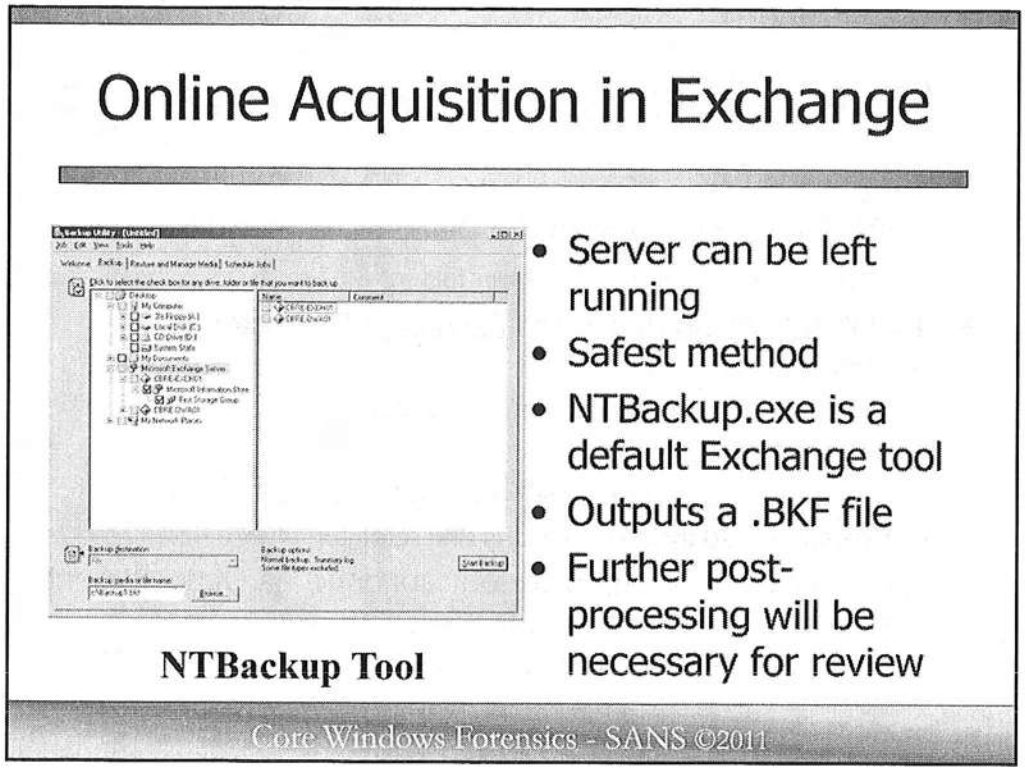
The Dumpster in Exchange 2010 has been re-architected to reflect the need for more granular corporate document retention polices and to facilitate things like litigation holds common in e-Discovery requests. As such, it now retains all deleted objects (i.e. calendar and task entries) instead of just e-mail and the default retention time has been increased to 14 days [2]. It also now indexes all Dumpster content, to allow searches common in e-Discovery. It also removes the ability that users had to purge e-mail from the Dumpster.

Dumpster objects can be recovered via the Outlook client. In order to recover Dumpster objects outside of the Deleted Items folder, the registry key HKLM\SOFTWARE\Microsoft\Exchange\Client\Options\DumpsterAlwaysOn must be added to the client system and the value set to "1" [3]. This is set by default in Outlook 2007, but must be done manually in all prior versions. Once this action is complete, the admin or user can use the `Tools -> Recover Deleted Items` option to pull back objects from the Dumpster. Alternatively, some commercial forensic tools have the ability to parse Dumpster files from within a .EDB file.

[1] http://technet.microsoft.com/en-us/library/aa996226%28EXCHG.65%29.aspx

[2] http://msexchangeteam.com/archive/2009/09/25/452632.aspx

[3] http://technet.microsoft.com/en-us/library/aa997155%28EXCHG.80%29.aspx




Online acquisitions of Exchange data can be the safest means to ensure a working database copy, but the restore process can be time consuming for the forensic investigator. While it is often easiest to simply shut down Exchange and copy the database files off the system, there is a downside. The first problem is that e-mail access (along with contacts and scheduling stored on the server) for the organization will be down for as long as it takes to make the copies (and Exchange archives can be very large). Acquiring the data in this fashion also circumvents the consistency checks that are built into the Exchange backup application programming interface (API). On the other hand, using the built-in online backup feature provides several benefits. Critically, it allows a backup to be made while the database is still mounted and the server is operational. Microsoft emphatically recommends online backups due to the file-level consistency checks that are performed by the API. [1] They state, "online backups are the single most valuable tool for detecting corruption and performing data recovery".

The NTBackup tool (shown on this slide and installed by default in Exchange) is the simplest way to perform an online backup. There are also numerous third party backup software packages leveraging the online backup API that you may encounter. A few of the most popular are: Symantec Backup Exec, Computer Associates ARCserve Backup, and EMC NetWorker. Exchange backups can be extremely large and NTBackup was designed to output to large capacity backup devices, like tape drives. Backups are output in one file, using a .BKF file extension. This output format is not natively recognized by many forensic tools and thus the additional step of using NTBackup to restore the .EDB and .STM files from the backup file is often required. If tape backups exist, they will likely be in the .BKF format. Depending on the case, tape backups can provide a wealth of deleted information that no longer exists in the current e-mail database. Due to the time consuming nature of the restore process, many forensic professionals still opt to perform offline backups, directly copying the .EDB and .STM files, when possible.

Domain Backup Operator permissions are required for the user account that is performing the online backup.

**References**

[1] http://support.microsoft.com/kb/909644

## Offline Acquisition in Exchange

1. Unmount Exchange database
2. Verify databases are consistent with **`eseutil.exe`**
3. Copy the .EDB and .STM files
4. Check-mount database copies
5. Verify checksums with **`esefile.exe`**



There are cases where it is preferable to acquire Exchange database files while the server is offline. Since you will not be using the backup API with its built in consistency checks, your top priority is to ensure that consistency. Part of the Exchange database is kept in memory and thus it is important that the .EDB and .STM files match one another and that they were cleanly shutdown. Unmounting the database within Exchange is our first step, and *should* provide that consistency. The `eseutil.exe` tool (installed by default on Exchange server) checks to see if the last database shutdown was clean or dirty. [1] Sample command lines are: `eseutil /mh priv.edb` and `eseutil /mh priv.stm`

Once we are satisfied with the consistency of the database files, we can copy them to external media or a network share. The newly copied database files can then be mounted to ensure that the data is accessible. This can be done on the current Exchange server or a server in your lab. From a strict forensic perspective, we don't like to open evidence files on the system they were recovered from, but in many cases it may be preferable to do so in order to assure that the database backup was successful while you are still onsite. Like the other steps in this process, make sure you document your actions in your investigator notes!

The final check that can be made on the database files is to run the esefile.exe tool to check the database page checksums. This tool will only work on .EDB files. This is an optional step, but provides a final confirmation that you have a working copy of the database. You will find esefile.exe on the Exchange CD-ROM and a sample command line is: `esefile /s priv.edb`

These instructions are for a "point in time" backup, meaning log files are not captured, and the ability to roll the database forward (integrate it back into a running instance of the server and add back any data since the last backup) is not possible. This is consistent with the needs of most forensic analysis / E-Discovery scenarios. [2]

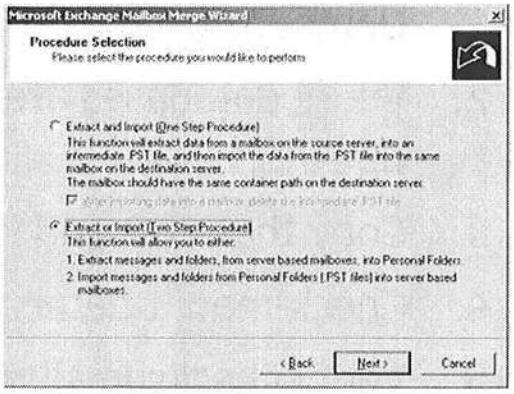
**References**

[1] http://support.microsoft.com/kb/271987

[2] http://support.microsoft.com/kb/296788

# Acquiring Single Mailboxes - ExMerge Tool

- On Exchange Server 2003, 2000, and 5.5
  - Recover .PST using ExMerge (MS Exchange Mailbox Merger)
  - Download Location: In notes
  - HOWTO GUIDE: http://www.msexchange.org/tutorials/ExMerge-Recover-Mailbox.html




Core Windows Forensics - SANS ©2011

In addition to acquiring the entire database, tools like ExMerge exist that can be used to export individual user mailboxes from the Exchange database files. Sometimes a complete acquisition of the server database is not required, and having access to just a few mailboxes will suffice. Mailboxes are typically exported as a .PST file, which is a well supported format for reviewing and conducting forensic analysis.

The servers should have backups (as all good e-mail servers do right?) Seize the backups as well as the account information and collect log files from the server itself. It will contain information vital to the examination of the incident. You should have the exchange administrator export the mailboxes of the subject as *.pst files using a tool such as ExMerge.

DOWNLOAD: http://www.microsoft.com/downloads/details.aspx?FamilyID=429163ec-dcdf-47dc-96da-1c12d67327d5&displaylang=en

HOWTO GUIDE: http://www.msexchange.org/tutorials/ExMerge-Recover-Mailbox.html

**References**

[1] http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q174197&ID=KB;EN-US;Q174197&

# Export-Mailbox Tool

- Command line tool in Exchange 2007 and 2010
- Can export single mailboxes or an entire database
  - `Export-Mailbox -Identity rob@sans.org -PSTFolderPath C:\rob.pst`
  - `Get-Mailbox -Database 'Corporate' | Export-Mailbox -PSTFolderPath C:\PST`
- Improved filtering capabilities
  - Search by content, subject, sender, recipient, date range, attachment name, and folders
- Automatically recovers Dumpster files (Exchange 2007)
  - Exchange 2010 removed this capability
- No size limit on PST files (ExMerge has 2GB limit)

Core Windows Forensics - SANS ©2011

Exchange 2007 and 2010 now come with an ExMerge replacement called Export-Mailbox (implemented as a PowerShell cmdlet). It extends the ExMerge functionality, allowing more flexibility when exporting mailboxes from an Exchange Server. Similar to ExMerge, it allows data to be preserved without shutting down Exchange.

Export-Mailbox is a command line only tool which leverages the inherent advantages of PowerShell, including chained actions and the ability to script commands. In Exchange 2007, the command is performed from the client and requires the Exchange Management Tools to be installed, installing a snap-in to the PowerShell [1]. Exchange 2010 changed this functionality by building a "remote shell" that leverages PowerShell to run commands directly on the remote Exchange server [2].

By default, the following items are exported by the tool:

| | |
|---|---|
| Inbox | Deleted Items |
| Drafts | Junk E-Mail |
| Outbox | Sent Items |
| Journal | Calendar |
| Contacts | Notes |
| Tasks | |

A single mailbox can be exported, or the command can be chained with others (like Get-Mailbox) to export an entire database into individual PST files by mailbox. Similarly, built-in filtering commands can now be used to filter results by content, subject, sender, recipient, date range, attachment name, and folders [3]. This is commonly used by administrators to search for and remove e-mail with malicious attachments, but could also be used for limited e-Discovery request production.

As an added bonus, Exchange 2007 automatically restores all files from the Dumpster for the given mailbox and returns them to their original locations [4]. This feature was removed in Exchange 2010.

Finally, Export-Mail does not have the same 2GB .PST size limitation that frustrated so many attempts at using ExMerge on very large Exchange mailboxes.

[1] http://technet.microsoft.com/en-us/library/bb123778%28EXCHG.80%29.aspx
[2] http://technet.microsoft.com/en-us/library/dd297939.aspx
[3] http://technet.microsoft.com/en-us/library/aa998579.aspx
[4] http://technet.microsoft.com/en-us/library/bb266964%28EXCHG.80%29.aspx

## Offline Folder Files

- Exchange has feature to make mail available while user is offline
- Stored on local system as .OST file
- Mailbox is synchronized with server
  - De-duplication is often necessary
- Orphan .OST files are common
- Cannot be imported into Outlook

Core Windows Forensics - SANS ©2011

Unlike webmail, local e-mail clients have the added benefit of allowing users to interact with their e-mail even when they are not connected to a network. Microsoft Exchange implements this feature through the use of offline folder files, or .OST files. [1] Upon login to the Exchange server, the offline folder file is opened, unread mail is downloaded, and any changes (drafts, sent mail, etc.) made to mail while offline are then synchronized with the server.

Offline mail files are important to a forensic investigator because they can contain messages which no longer exist on the mail server. Their default directory is: `C:\Documents and Settings\<username>\Local Settings\Application Data\ Microsoft\Outlook`.

Orphan .OST files are quite common – if a mailbox is damaged or an Exchange server is rebuilt, the old .OST file will likely not sync with the newly created user account, and a new file is created behind the scenes. For this reason, offline folder files are often found in unallocated space or in deprecated directories in the file system.

Unlike .PST files, recovering data from an .OST file is not trivial. They cannot be imported into Outlook, and they generally utilize compressible encryption, rendering plaintext viewing impossible. A common fix is to convert the .OST file to a .PST file which can then be imported into Outlook. Many third party tools exist to perform this task, and the freeware tool ost2pst.exe has a loyal following on the net. [1] Additionally, some forensic suites (including EnCase) will natively parse OST files, facilitating search and review. If you are recovering e-mail from both the host and the server, keep in mind that you are likely to see a fair amount of duplicated data since the primary purpose of .OST files is to synchronize these data stores.

Outlook installations include a tool, `scanost.exe`, that can be used to repair a corrupt .OST file.

**References**

[1] http://office.microsoft.com/en-us/outlook/HP052422611033.aspx

[2] http://www.outlookexchange.com/articles/drewnicholson/utilities.asp

## Lotus Notes / Domino (1)

- Everything in Notes is a database
  - Databases saved as .NSF files
  - E-mail, calendars, tasks, journals, and contacts all typically found in .NSF
  - Local replicas allow offline use
  - notes.ini file gives locations of .NSF files
- .NSF files can be corrupted
  - IBM Fixup utility installed on server



Lotus Notes is a popular enterprise e-mail client, providing access to databases located on Lotus Domino server(s). It is important to understand that everything displayed in Notes is part of a database, and thus the client typically has access to much more data than other e-mail clients. E-mail is just one of the databases accessed by Notes, others typically include: calendars and scheduling, to-do lists, journals, documents, and business contacts. Databases are saved as .NSF files and can be located both on the Domino server as well as on the local machine. On local Windows machines, .NSF files are typically located in the following directories:

```
C:\Documents and Settings\user\Local Settings\Application Data\Lotus\Notes\Data
C:\Users\user\AppData\Local\Lotus\Notes\Data    (Vista)
```

The `notes.ini` file will be located in the "Notes" directory and will identify any alternate locations where .NSF files may be stored.

Like other large database files, corruption can be an issue. Especially if the Notes database was not cleanly shutdown. Lotus Domino installs with a tool named "fixup" that can be run through the Domino server or from the command line. [1] This tool will attempt to clean the affected database so it can be re-mounted for analysis.

**References**

[1] http://www-01.ibm.com/support/docview.wss?rs=0&uid=swg27006573

## Lotus Notes / Domino (2)

- Accessing encrypted databases
  - Default setting as of Notes version 6 (2002)
  - Password + Notes .ID file required
  - .ID file can be cracked
- Recovering deleted messages
  - Soft deletes in Trash folder
  - Hard deletes are difficult to recover
  - "Removed" files still located in All Messages folder

Core Windows Forensics - SANS ©2011

.NSF files can be reviewed using a local installation of Lotus Notes, but dedicated forensic tools typically make analysis of the database files easier. Products by Paraben (P2), Guidance Software (EnCase), and AccessData (FTK) all have the capability to mount and search .NSF files. IBM also provides a free tool, NotesPeek, for reviewing Notes database files. [1]

Since Lotus Notes 6 was released (in 2002), encrypted database files are much more common since it is the default setting. When exporting Notes archives for review, it is critical that the .ID file associated with the database is included. This file contains the password hash necessary to decrypt the archive. If the password is unknown, several third-party password cracking utilities support Notes .ID files.

Messages deleted by the user are placed in the Trash folder within the Notes database. They can be recovered until the point where they are purged from the folder – manually or via automated cleanup. Once a message has been purged, it becomes much less likely that it will be recovered. Unless another user has a link to that message, the Domino server immediately deletes the message from the database, and on production servers, physical overwrites occur very frequently. [2]

Lotus notes also has a "remove" feature. This feature allows users to remove an obsolete message from their folder or view, while still keeping it archived in the database. Removed messages will only be located in the "All Messages" folder, underscoring the need to review this folder in addition to any others in the database.

**References**

[1] http://www-10.lotus.com/ldd/sandbox.nsf/ByDateNJ/2791869f4e1d3fa385256f2c00432973?OpenDocument

[2] http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp?topic=/com.ibm.help.domino.admin.doc/DOC/H_SPECIFYING_A_SHARED_MAIL_DATABASE_FOR_NEW_MAIL_DELIVERIES_MIDTOPIC_24682880029225326.html

## Novell Groupwise (1)

- Includes e-mail, contacts, calendar, tasks, and document management
- Databases stored with .DB extension
  - Most e-mail, documents, etc. stored here
  - Can encounter up to 25 .DB files
- Very complicated file structure
  - Acquire entire `Dom` and `Post` dirs
  - Place in same location on forensic system



Groupwise is by far the most difficult application to retrieve evidence from out of all of the popular mail servers and clients. From acquisition to review, Groupwise is a challenge, and if you are tasked with performing forensics on a Groupwise archive, it is recommended that you get assistance from a subject-matter expert. Luckily, Groupwise is far less common than either MS Exchange or Lotus Notes.

There are two main types of databases in the GroupWise application. Administrative databases are stored in the Domain directory (named "Dom" by default on the server) and they contain users and their permissions. [1] The Post Office databases are located by default in the "Post" directory, and contain a series of files and directories forming each user's message store. [2] While most of the interesting data is kept within a large number of database files (up to 25 files using the .db extension), Groupwise is very particular about the location of files within its directory structure. The best practice is to acquire both the dom and post directories in their entirety and place them in their same locations on your forensic machine.

[1]
http://www.novell.com/documentation/gw65/index.html?page=/documentation/gw65/gw65_tsh3/data/h6rf9mws.html

[2]
http://www.novell.com/documentation/gw65/index.html?page=/documentation/gw65/gw65_tsh3/data/h6rf9mws.html

## Novell Groupwise (2)

- Attachments stored separately
- Exported e-mail stored locally as .MLM files
  - Saved as WordPerfect documents!
- Messages can be restored from trash folder if cleanup has not occurred
- Consider a specialized tool for analysis
  - Paraben, Nexic Forensics, or UniAccess



Similar to the way Eudora handles attachments, Groupwise separates user attachments outside of the message databases to increase efficiency. They are located in the `post\offiles` directory and provide an easy means to quickly review all attachments related to the GroupWise message store.

GroupWise does utilize a Trash Folder and messages can be restored from it. When GroupWise is known to be used, the investigator should also search the local system for .MLM files, which is the format used when a user exports them. Interestingly, .MLM files are saved in WordPerfect format, which will require converting to be viewed a plain text.

Few forensic tools currently support GroupWise archives. Paraben's Network E-mail Examiner includes support as do some targeted tools like Nexic Forensics and UniAccess. The latter is often used to convert GroupWise message stores to other formats to support application migration. [1]

[1]
http://www.inse.com/INSE/INSEWeb4.nsf/0/65F54F008B4CEEDCC12570C200539F3E!OpenDocument

## Webmail Forensics

- E-mail typically stored on ISP servers
  - Possible exception for POP or IMAP
- User IP address and subscriber info may be available from ISP
- Look for webmail addresses
- Cached copies can be recovered
  - Web 2.0 technology reduces chances

Core Windows Forensics - SANS ©2011

Webmail complicates our forensic efforts because there is often no offline archive. The lion's share of all webmail exists on the servers run by the service or ISP. An exception exists for users who have setup offline e-mail storage using the POP or IMAP protocols allowed by some services. In this case, we would expect to find mail in the user's e-mail client and the host-based e-mail techniques previously discussed could be used to recover and analyze it. Barring that, we are often left with recovering webmail fragments using traditional computer forensic techniques (such as keyword searching and carving), or requesting the preservation and release of e-mail data from the ISP using a court order.

It should be noted that in addition to the e-mail data, the ISP may have other information which could aid an investigation. Subscriber information could help tie the account to an individual. Most webmail services now record the IP address of the computer that created the webmail account. And in some cases, IP address information may be available for who has accessed the webmail account. Being able to tie the use of an account to an individual and/or a computer can be very useful information, but to get it, a court order is necessary.

One of the biggest problems with investigating the use of webmail is determining *if* it is being used and, even more critically when requesting data from the ISP, what e-mail accounts the user owns. Web browser forensics often holds the key to this information along with using regular expression searching to identify e-mail addresses on the target media. Searching the protected storage and auto-complete functions within the web browser can also provide both account names and passwords. [1] However, it should be noted that using recovered passwords to log into a users e-mail account will almost always be outside of the authorization for the forensic exam and could incur civil or even criminal penalties.

First generation webmail services often provide a wealth cached information on the local system (Yahoo! Classic e-mail is a good example). With these older versions, investigators can expect to find HTML pages containing messages displayed from those services using the computer's web browser (sent messages are more

rare since they aren't usually cached). As webmail services move to a Web 2.0 infrastructure, technologies like AJAX tend to leave much less residual data. In some cases (like Gmail), the "no-cache" option within AJAX is utilized to explicitly direct the browser not to cache, thus increasing user security on shared systems. We should keep in mind that even without information in the cache, our traditional forensic methods still often provide evidence from webmail. Fragments can be found in unallocated space, and even more exciting is the new research into memory forensics that is pulling out a wealth of webmail information from memory dumps and/or hibernation files. [2,3]

**References**

[1] http://www.nirsoft.net/utils/pspv.html

[2] http://sansforensics.wordpress.com/2008/10/20/pdgmail-new-tool-for-gmail-memory-forensics/

[3] http://sansforensics.wordpress.com/2009/01/12/pdymail-yahoo-mail-in-memory/

# Cryptome Legal Guides

**http://cryptome.org/**

- "Leaked" guides for contacting and requesting information from webmail providers
- Guides available for most major providers
  - Contact information
  - What data is preserved and available for request
  - Request procedures and paperwork required
- Intended for law enforcement / legal departments



Cryptome is a website established in 1996 to collect an preserve documents related to "freedom of expression, privacy, cryptology, dual-use technologies, national security, intelligence, and secret governance" [1]. Legal guides sourced from ISPs and other Internet providers have been circulating on Newsgroups and P2P intermittently and have been since archived on Cryptome. Guides are available for most major Internet service providers, and given the site's recent prominence, any remaining guides will likely make their way there in the near future. These guides are typically intended for law enforcement agencies or legal groups which plan to request information via a court order (or equivalent). In February 2010, Microsoft served Cryptome with a DMCA takedown notice for their legal guide. Cryptome was unreachable for three days until Microsoft withdrew their complaint [2]. An excellent article on the SANS Computer Forensics Blog links to many of the most popular guides [3].

[1] http://cryptome.org/

[2] http://en.wikipedia.org/wiki/Cryptome

[3] http://blogs.sans.org/computer-forensics/2010/03/02/cryptome-spying-guides-as-a-digital-forensic-resource/

# Cryptome Legal Guide – Yahoo!

**COMPLIANCE GUIDE AT A GLANCE**

- **How do I contact Yahoo! Legal?**

  Questions: Compliance Team
  Yahoo! Inc.
  701 First Avenue
  Sunnyvale, California 94089
  [redacted] (tel.)

  Subpoenas/Other Service of Process: Fax requests for documents to Custodian of Records at [redacted]
  Subpoenas for in-person testimony must be personally served.

  After-hours emergencies: Yahoo! Security at [redacted]

- **General Tips:**
  - Include a Yahoo! ID or Yahoo! email address in your request.
  - Before making a request, check to see if the information sought is publicly available. See http://help.yahoo.com to find publicly available information.
  - Make requests as specific and narrowly tailored as possible.

- **What Information Can Yahoo! Provide?**
  - **Subscriber Information**
    - Subscriber information supplied by the user at the time of registration, including name, location, date account created, and services used.
    - IP addresses associated with log-ins to a user account are available for up to one year.
    - Registration IP address data available for IDs registered since 1999.
  - **Yahoo! Mail (including email associated with specific properties such as Personals, Small Business, Domains, and Flickr)**
    - Any email available in the user's mail account, including IP address of computer used to send email.
    - Yahoo! is not able to search for or produce deleted emails.
    - Note that Yahoo! now hosts two new email domains: ymail.com and rocketmail.com.
  - **Yahoo! Chat Messenger**
    - Friends List for Yahoo! Messenger
    - Time, date, and IP address logs for Chat and Messenger use within the past 45-60 days.

Data Available:

- Subscriber info
- Profile data
- Yahoo! Mail
- Yahoo! Messenger
- Yahoo! Groups
- GeoCities
- Flickr

Core Windows Forensics - SANS ©2011

Legal guides provided by Internet service providers can be extremely detailed. As such, they can be very valuable when brainstorming for the types of data which may be useful to a specific investigation.

The Yahoo! guide is 17 pages long and provides detailed contact information, including an after-hours contact number [1]. It covers information requests for subscriber information, profile data, Yahoo! Mail, Yahoo! Messenger, Yahoo! Groups, Yahoo! GeoCities, and Flickr. For each service, the data availability is covered including how long the data is accessible and how often (if ever) data is purged. Multiple sample forms are provided to facilitate data requests. To illustrate how comprehensive the document is, a consent form is even provided which could be used by law enforcement or a corporation to seek an individual's consent as a means of compelling Yahoo! to release the data.

[1] http://cryptome.org/isp-spy/yahoo-spy.pdf

## Compressed Webmail Remnants

- Webmail is often transferred in a compressed format
- Internet cache will contain compressed files
  - Must be "unzipped" to view HTML data
- File signature analysis may be required to identify compressed files

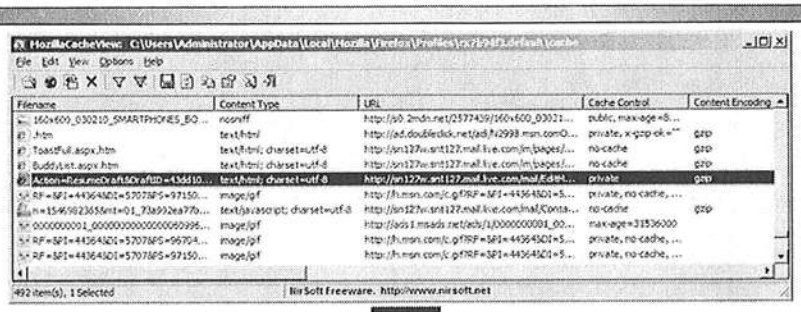


A recent trend with webmail is the sending of data in compressed files. Starting with HTTP version 1.1, browsers support the receiving (and inflation) of compressed content. While this reduces content transfer times, it also creates a layer of obfuscation when conducting computer forensic analysis. Simply put, it is very easy to overlook webmail evidence within the Internet cache files since it may not be stored in HTML format.

Forensic tools, and specifically file signature analysis, make identifying compressed content much easier. In browsers such as Mozilla Firefox, cache files are not stored with extensions and thus identifying relevant items can be a laborious process. After a file signature analysis is complete, it is necessary to inflate (or decompress) the found files, which can then be reviewed using a standard web browser.

It is important to point out that string searches will not find content which is in a compressed format. Tools like EnCase require the investigator to first mount each compressed file, allowing string searches to be conducted within the uncompressed files.

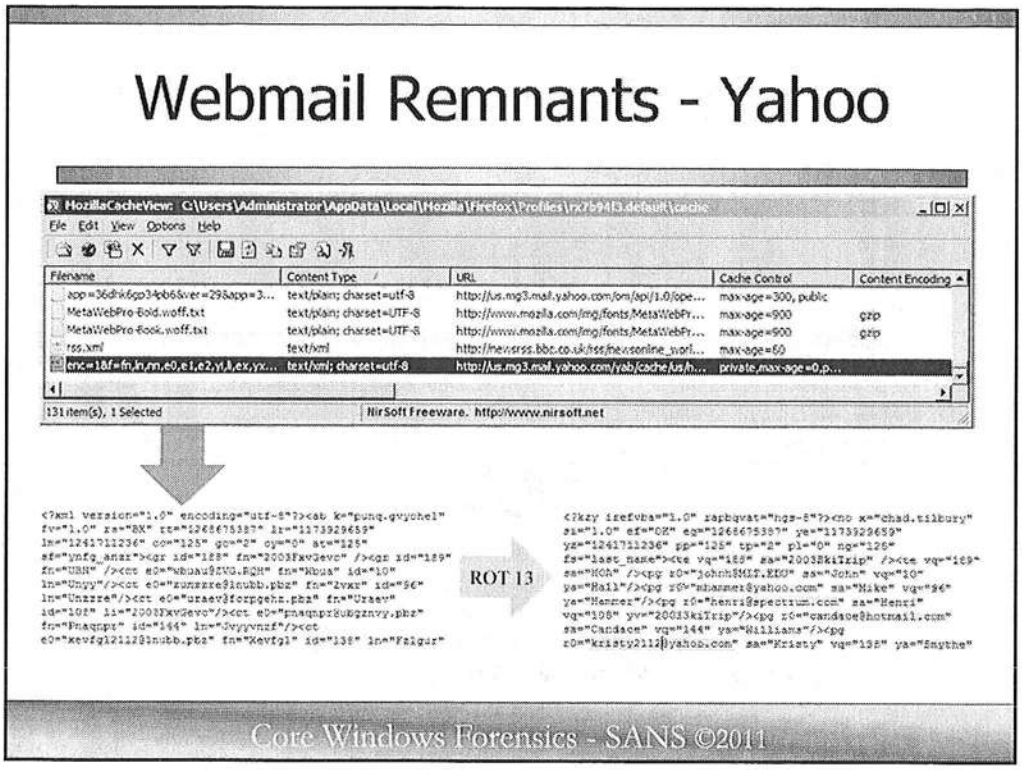
# Webmail Remnants - Hotmail



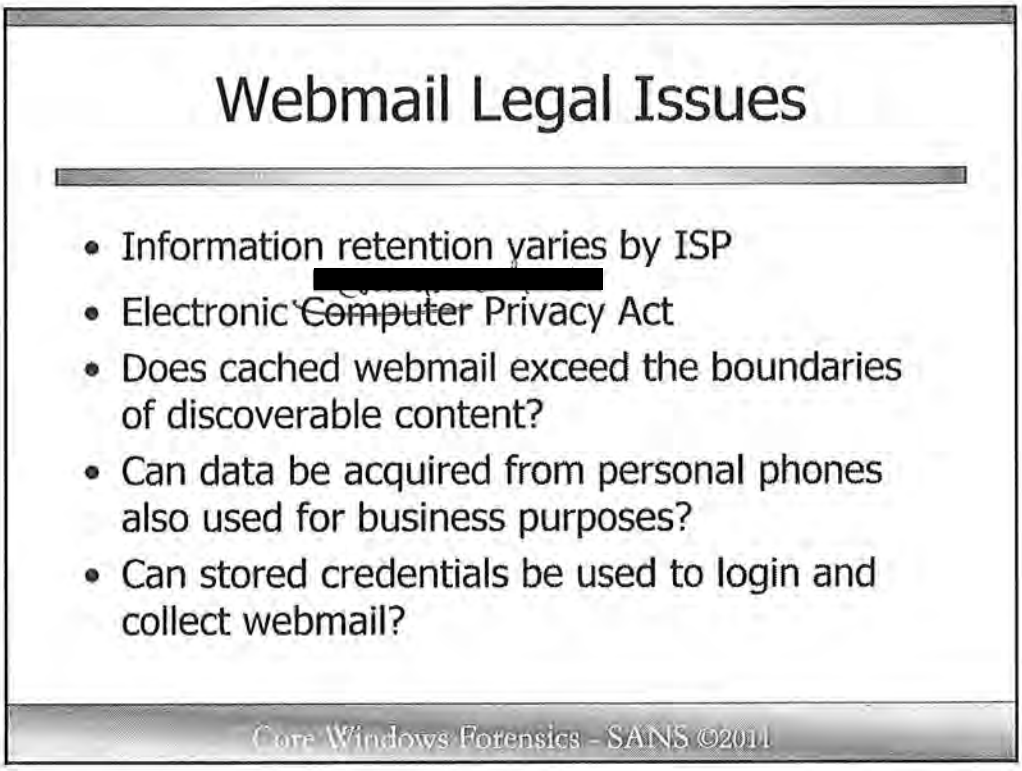
Verdana;}

</style>
</head>
<body class= hmmessage >We need an emergency meeting now.  Things went badly.  Same place / time as before.  <br></body>
</html></textarea>

Core Windows Forensics - SANS ©2011

With the advent of Web 2.0 technologies, webmail remnants are becoming increasingly sparse. That being said, a thorough review and good string search terms often find relevant information. This slide shows remnants from a Microsoft Hotmail session (http://live.mail.com). Amongst all of the session artifacts, a file named "Action=ResumeDraft..." is found. Most artifacts are not as nicely named as this one, but it is in fact an e-mail saved to the Drafts folder within Hotmail. The investigator should be very careful when reviewing these artifacts because it is easy to miss text amongst all of the various HTML, XML, headers, and encoding artifacts. On a heavily used system, your best may be to rely upon string searching. However it should be pointed out that this artifact was found in a gzip file, meaning it was compressed and thus not inherently searchable. This underscores the reason why forensic tools, such as FTK, make our lives much easier by "mounting" compound or compressed files and indexing text strings within them. One additional item to point out is that several of the cache files displayed have been tagged with the "no-cache" option. These items will not be stored in the cache, and only metadata should be expected to be recovered.




This slide is a great example of how challenging webmail artifacts can be to find. The file name gives no indication of the file contents, though the URL does seem to indicate it is somehow related to a "cache". The file itself is in XML format, and is not compressed. While that combination would typically be excellent for string searching, in this case, the data is encoded with at ROT13 cipher, making string matching more difficult. Once the data is unobfuscated, it is clear that the file is a copy of the user's address book, complete with e-mail addresses, first and last names, and groups (i.e. 2003SkiTrip). After analysis, we can surmise that the "yab" directory within the URL is short for Yahoo! Address Book.




Webmail can add many difficulties to the job of a forensic investigator. Unlike local and server based e-mail, webmail is almost always located on shared servers which are not part of the enterprise. Thus recovery of web based e-mail is largely dependent on the practices of the webmail provider. In most countries, there is no legislation outlining the type of information to be kept by ISPs nor how long information should be retained (the European Union is close to enacting laws related to this). Some webmail providers maintain account holder IP address information in addition to messages. Some ISPs maintain data in unused accounts for several months, while others eliminate accounts monthly. In short, it is very difficult for an investigator to predict what type of information may be available from a webmail provider.

To make things more difficult, some ISPs have refused to furnish data requested by civil subpoenas due to the fact that the Electronic Computer Privacy Act (ECPA) does not have a provision for disclosure to non-law enforcement entities. [1]

Given these difficulties, the best location to recover webmail is likely to be local copies. In the best case, these will be copies stored in an e-mail client using IMAP or POP3. However in many cases, the only local data will be what is located in the Internet cache. For E-discovery purposes, it is important to identify whether compressed, cached, copies located in hidden directories fall under the guidelines of content that is "reasonably accessible" in the Federal Rules of Civil Procedure Rule 26(b). [2] Similarly, it must be weighed whether content on employee smartphones has the potential to contain relevant enough information to outweigh the cost and burden of producing it. Finally, a temptation sometimes exists to log into webmail accounts using recovered or stored credentials. Without a specific court order to do so, this could open the investigator up to a host of potential civil / criminal penalties.

**References**

[1] http://www.law.com/jsp/legaltechnology/pubArticleLT.jsp?id=1202423171198

[2] http://www.law.cornell.edu/rules/frcp/Rule26.htm

## Mobile E-mail

- Mobile devices receive e-mail from a corporate or webmail server
  - Mail often stored in both locations
- SMS messages difficult to retain
  - BlackBerry PINcushion servers often employed to assist with compliance
- Look for local backup copies
  - .PST files, BlackBerry .IPD files, iPhone .mddata and .mdbackup files



Many investigators fail to take into account mobile e-mail when acquiring evidence for a case or E-discovery effort. In some cases, this is justified, since many smartphones are synched to a corporate mail server and only maintain copies of e-mail. In an investigation, it is important to verify how e-mail-capable phones interact with these mail servers and determine whether webmail acquisition is in scope. In the case of webmail, the device could have messages on it that would be difficult to get elsewhere, and thus could be a target for acquisition and review.

Text, or SMS, messaging is notoriously difficult to archive and collect, and in most cases must be acquired from the mobile phone itself (or potentially from the ISP if it has a SMS retention policy). Many organizations subject to compliance legislation have begun to turn to solutions like the BlackBerry PINcushion server, which operates as a repository for SMS, instant messaging, webmail, and BlackBerry Pin-to-Pin communications. [1] If it exists, data from such a server could be vital to an investigation.

Smartphone backups to the local machine are often found. Microsoft ActiveSync often dumps data to Outlook .PST files and BlackBerry systems backup phone date to .IPD files. [2] A recent backup file may obviate the need to acquire the actual mobile device, or provide historical data which has since been deleted. Depending on the firmware version, the iPhone uses .mdbackup or .mddata files to store backups on the local system – the default folders are `Users/Library/Application Support/MobileSync/Backup` on Mac OS X, and `%user profile%\AppData\Roaming\Apple Computer\MobileSync\Backup` on Vista/Windows 7.

[1] http://www.commondesk.com/products/compliance/products-compliance-engine.html

[2] http://www.forensicswiki.org/wiki/Blackberry_Forensics#BlackBerry_Simulator

## Filtering E-mail Duplicates

- Exact Duplicates
  - Utilizes forensic hash to find matches
  - Not as effective for e-mail
  - Forwarded e-mail does not match original
- "Near" Duplicates
  - Compare specific e-mail fields
  - Piecewise or "fuzzy hashing"

Core Windows Forensics - SANS ©2011

The identification and collection of data is fairly trivial when compared with the difficulty of processing and reviewing the data once it is preserved. This is particularly true with respect to e-mail, which tends to be high volume and very duplicative. Imagine an e-mail you receive regarding a new company policy. In a large organization, hundreds or thousands of copies of that e-mail could be collected from server stores, local archives, and offline folders. When successfully using de-duplication, only one copy would need to be reviewed. Thus, data de-duplication is a critical part of the E-discovery process because it can greatly reduce the amount of data that needs to be reviewed.

There are currently two methods for identifying duplicates. The first is relatively straightforward and follows from computer forensic techniques – create a hash for every item collected and use it to eliminate items with duplicate hashes. This is a precise method, and tends to work better for documents than more dynamic e-mail files. Unfortunately, the amount of e-mail collected in a typical E-discovery effort is an order of magnitude greater than the number of documents.

Going back to our company policy e-mail scenario, how do we filter out copies of that e-mail that have been forwarded or slightly changed in transit? One answer is instead of trying the match the entire e-mail, we can identify specific pieces of the e-mail, such as time, date, and message body, to focus our matching efforts. Piecewise hashing can be used to fingerprint a document by breaking it into tiny pieces and hashing each piece. Each piece can be compared between the documents and a percent likelihood of matching can be generated. This approach suffers from two problems: it is much more resource intensive to calculate ad compare piecewise hashes, and it is up to the parties to negotiate what level of confidence they will require when eliminating duplicates.

Given the billion dollar industry that E-discovery has become, a plethora of companies exist with de-duplication technologies and research is ongoing in improving the de-duplication process.

## Commercial E-Discovery Suites

- Searching
  - Conceptual searching
- Organization
  - Communication Relationships
  - Timeline creation
  - Statistical analysis
- Production
  - Native (w/ metadata), .TIFF, .PDF outputs

Core Windows Forensics - SANS ©2011

Performing E-Discovery on e-mail messages is difficult work, and a vast array of companies have product offerings to try to simplify the process. In addition to standard de-duplication and searching techniques like indexing or regular expression usage, one of the newer technologies to emerge is called conceptual searching or "latent semantic indexing". [1] In simple terms, this technology groups messages into related clusters utilizing the concepts they refer to, instead of specific keywords. When it works, it can provide an improvement over standard de-duplication and be very helpful for reviewing large quantities of similar messages.

E-mail in particular lends itself to several different means of organization. Many products allow e-mail archives to be organized according to sender and recipients and when messages were sent. Statistical analysis of e-mail frequency and volume is a feature of some E-Discovery software and can also be especially helpful if the investigation is centered upon specific events.

Production, or output of data deemed relevant to the legal discovery request, is important and most suites give the option of output in native (with metadata), TIFF, or PDF formats. Bates numbering schemes are also standard.

Many computer forensic companies have adapted their core products to facilitate E-discovery. Examples include AccessData eDiscovery and EnCase Enterprise. Additionally, an entire cottage industry has sprung up to support this field, with products from Clearwell, Attenx, MetaLINCS, Ring Tail, Concordance, and ImageMaker commonly found in use.

**References**

[1] http://en.wikipedia.org/wiki/Latent_semantic_analysis

# Review: Forensic E-mail Analysis

1. Review installed applications
2. Locate and acquire e-mail archives
3. Process and review e-mail using forensic tools
4. Export relevant files from archive
   a. Maintain hash values for exports



E-mail analysis is a critical part of almost any forensic examination. This is particularly true in the E-discovery world. There are several popular e-mail servers and many more e-mail clients used in the wild. While it is impossible to be an expert in every combination, there is a simple methodology that should be followed.

The first step is to identify what e-mail clients exist (or have existed) on the system, and what e-mail servers they connecting to. This usually requires reviewing the folder structure of the system as well as reviewing the Windows registry for installed applications. Webmail use can be identified through the Internet History, cookies, and cached files.

All mail archives should be forensically acquired within the scope of authority to search. This includes both server archives as well as anything stored locally on the system. Since it is common to have many different types of e-mail collected during an investigation, investigators often convert archives to a consistent format (.PST is the current favorite) to aid in review and with the de-duplication process. In cases where mail cannot be converted or reviewed by your forensic suite, it may be necessary to install the specific server software on a system and import the archive for review.

Once relevant files are identified, they should be exported out into a portable, easy to review format. Hash values should be collected for these new files as they should be considered evidence. In an E-discovery case, this collection of files will be those that are "produced" to the opposing counsel and will need to be rendered in the format requested (TIFF, PDF, raw).

# Spam Case Hands-On (1)

## Scenario:

Your office has been investigating a major spammer working for unlicensed online drug companies. The local prosecutor has finally taken an interest in the case and has asked several pertinent questions that must be answered.

Sadly, the previous examiner was hit by a bus and left no notes related to their investigation. You have been asked to quickly analyze the raw e-mail archives harvested from a SPAM trap and provide your results to the prosecutor.



We will pause the case on Donald Blake for a minute. You receive a call to investigate a spammer. You receive the evidence in E01 Format and you need to examine it.

Unfortunately this scenario is not all that uncommon! As an investigator it is quite likely that you will have to redo an analysis due to the previous examiner not documenting their work. You'll only need to do this once to realize why excellent note taking and report writing is critical to this business.

In this hands-on exercise, you will be provided with an image file containing e-mail recovered from SPAM traps in .PST format. To make the image file more compact, we have removed any other files that would normally be part of a disk image. You will be using the Forensic Toolkit to load the image file and analyze the e-mail. Remember that these e-mails were harvested randomly from the Internet and only some will be relevant to your case! The next few slides will walk you through doing your analysis.

# Spam Case Hands-On (2)

Find Spam.E01 and case overview on DVD, directory **email-case** :

1. Create new FTK Case
2. Add Spam.E01 as evidence
   a. How many e-mail archives are found? _____
   b. How many total messages? _____
3. Perform a **live search** for "pharmacy"
   a. How many files were found? _____

Core Windows Forensics - SANS ©2011

Find the location of the file Spam.E01 and open up your copy of the Forensic Toolkit. We will be performing a bit of pre-processing of the image, specifically loading our evidence file and indexing it to make searching faster.

From the toolbar, click **File -> New Case.** A series of dialogs will be opened to gather input:

**New Case Dialog:** Insert case number, case name, and an optional description

**Forensic Examiner Dialog:** Add any relevant information

**Case Log Options:** Leave all defaults values

**Processes to Perform:** Make sure **Full Text Index** is selected (all other options should be unchecked)

**Refine Case:** Leave all default values

**Refine Index:** Leave all default values

**Add Evidence to Case:** Click **Add Evidence-> Acquired Image of Drive ->** select **Spam.E01 file ->** select **Mountain Time with Daylight Saving as the time zone**

**Case Summary:** Click **Finish**

To perform the search in FTK:

Go to the **Search** tab

Select the **Live Search** tab

Enter search term: pharmacy

Click **Add**

Ensure ASCII and Unicode is checked

Click **Search -> Choose All Files**

Click **View Results**

# Spam Case Hands-On (3)

4. Do a **live search** for "viagra"
   a. Compare your results with an **index search** for "viagra"
      i. Why do the results differ? _____
5. Review findings from your searches
   a. What evidence is there that these messages are related?
      i. _____
      ii. _____



We will continue our search for relevant content. This time performing both a "live search" and an "index search" and comparing the results.

To perform the live search in FTK:

Go to the **Search** tab

Select the **Live Search** tab

Enter search term: viagra

Click **Add**

Ensure ASCII and Unicode is checked

Click **Search -> Choose All Files**

Click **View Results**

To perform the index search in FTK:

Go to the **Search** tab

Select the **Indexed Search** tab

Enter search term: viagra

Click **Add**

Click **View Item Results**

Now review the results of these three searches, looking for any commonalities in the message body, headers, and any attachments.

## Spam Case Hands-On (4)

6. What pharmacy site(s) was this person working for? _____
7. Do we have any clues to his/her identity? _____
8. Where might this person be located? _____
9. **Extra-credit:** What other illegal activity is this individual involved in?



You should have identified some messages which may be related to your investigation on the online pharmacy spammer.

What websites were the messages advertising?

Do we have any potential clues to who may have been the originator of these messages? It's okay if we can't definitively prove who the person is – right now we are looking for clues that we can build upon in other investigative steps.

Did you find any clues that might give any information related to where the spammer may live?

Using the clues you have to this person's identity, how could you try to identify other spam messages that they may be involved in?

**Don't turn the page until you are ready to look at some additional hints and the exercise answers.**

## Spam Case Hints

- Review e-mail message headers
- Review any IP address information
- www.dnsstuff.com can provide information on IP addresses
- Run additional searches to try find any other related messages



E-mail messages contain relatively few places for evidentiary data. Make sure that you analyze the information provided in the message bodies and particularly the header information. Spammers take great pains to remain anonymous, but like most criminals, they can let their guard down and make mistakes.

IP address information is very useful for identifying individuals behind spamming information. Remember that almost every piece of the e-mail headers can be faked – including the IP addresses of most of the mail servers. Sometimes we can get lucky here, but another excellent item to look for is X-Originating-IP tags which can be the IP address of the computer the message was sent from.

You can use sites like dnsstuff.com, samspade.com, and domaintools.org to help you with IP address lookups.

Finally, take the clues you find and run additional searches across the e-mail archives. Remember that forensic investigations are very iterative!

# Spam Case Answers (1)

Find Spam.E01 and case overview on DVD, directory **`email-case`** :

1. Create new FTK Case
2. Add Spam.E01 as evidence
   a. How many e-mail archives are found? **3**
   b. How many total messages? **2172**
3. Perform a **live search** for "pharmacy"
   a. How many files were found? **8 Hits in 3 Files**



The answers to the exercise are on the slides and discussed below.

First, we will create our case file and add our image file in for pre-processing.

Answer to question 2: This is found on the **`Overview`** tab under **`File Category`**. The **`Archives`** line indicates 3 archives. Further analysis shows all archives are .PST files. The **`E-mail Messages`** line reports a total of 2172 messages found in the image.

The first search we will conduct so for the keyword, pharmacy.

Answer to question 3: A live search in FTK conducts a bit-level keyword search on the entire forensic image. A search for the keyword "pharmacy" will return 8 hits found in 3 total files. The hits can be reviewed in the right-most pane.

## Spam Case Answers (2)

4. Do a live search for "viagra"
   a. Compare your results with an indexed search for "viagra"
      i. Why do the results differ? **Need to use wildcards for index searches**
5. Review findings from your searches
   a. What evidence is there that these messages are related?
      **i. Similar e-mail addresses**
      **ii. Same X-mailer header entry**



Perform a second **`live search`**, this time for "viagra", followed by an **`indexed search`** for comparison purposes. It is important to know the benefits and limitations to various options in your forensic tools.

Answer to question 4: The **`live search`** identifies 16 hits in 3 files. An **`indexed search`** initially finds no hits. Notice in the list of indexed words list that viagra1, viagracializ, viagraprofessional, viagrasofttabs, and viagrasuperactive are all present, but did not give hits. The index is being searched literally and thus wildcards will be needed to get the results we are looking for. An indexed search for **`*viagra*`** gives the same 16 hits.

Now we will review the e-mail messages found as a result of our three searches. Notice that there is some overlap between the searches, meaning that hits from different terms were found within the same message. If we were looking to narrow our results list, we could have searched for both terms at the same time.

Answer to question 5: Three related messages are found in our results list: messages 235, 318, and 434. All three messages share a similar e-mail address and associated name of Bettyann Cathy. Two of the messages also share the same X-mailer entry of Pocomail 2.63 (1077).

## Spam Case Answers (3)

6. What pharmacy site(s) was this person working for? **prosperitythriving.com, rxlist.net, lixjeseg.cn**
7. Do we have any clues to his/her identity?
   **Name could be a derivative of BettyAnn Cathy?**
   **X-Originating IP address of 74.12.232.136**
8. Where might this person be located? **Mississauga, Ontario (from DNSStuff.com)**
9. Extra-credit: What other illegal activity is this individual involved in? **Phishing**



Reviewing our related messages, we can now identify the companies this individual may be working for.

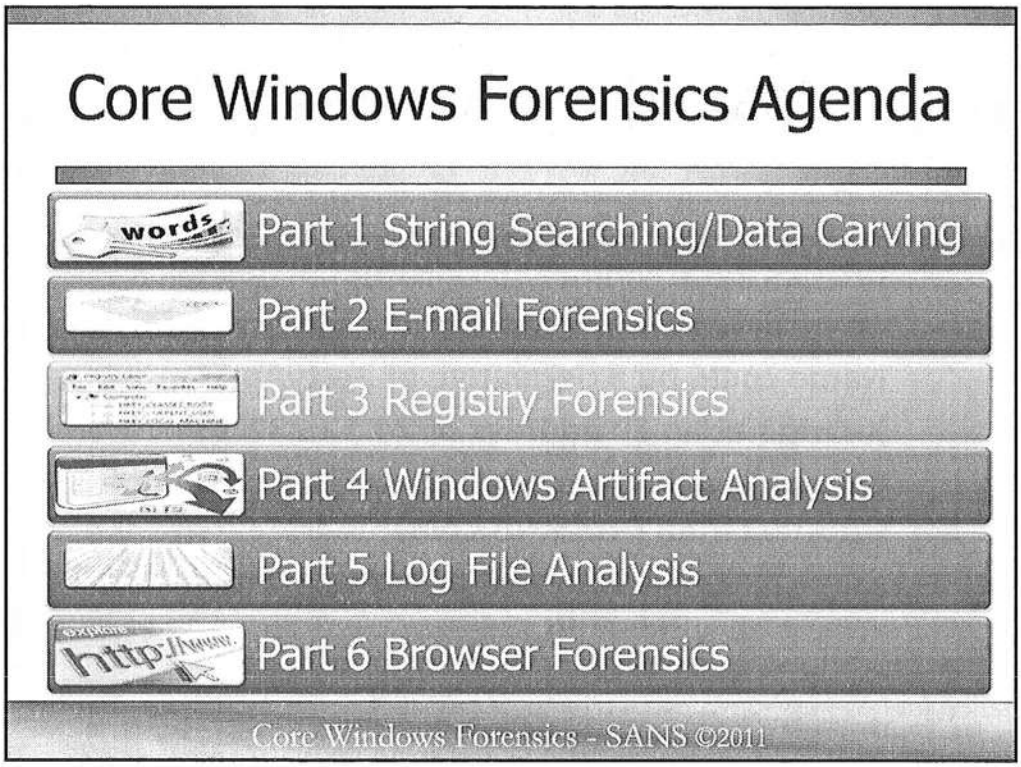
Answer to question 6: prosperitythriving.com, rxlist.net, lixjeseg.cn

Answer to question 7: The mail server IP information in the message headers appears to be inconsistent and in some cases, fake. However, one of the messages includes a valuable clue: an X-Originating IP stamp. If this hasn't also been faked, it could lead directly back to the individual sending the SPAM! In this case, the IP address is 74.12.232.136

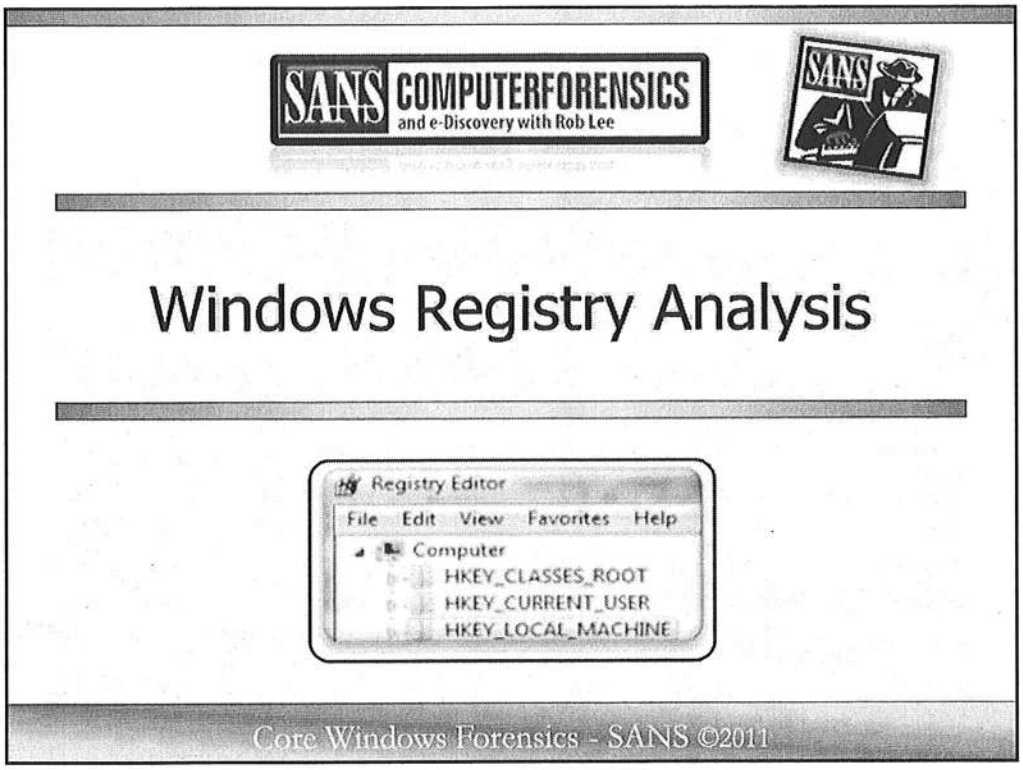
Answer to question 8: Taking the IP address from question 7, we can run it through geo-location software. In this case we will use the free tools on www.DNSStuff.com, which gives us a result of Mississauga, Ontario.

We only found three e-mails out of over 2000 captured by our SPAM traps. Remember that our search was quite narrow, based on only two search terms. A broader search of the e-mail archives using our new found information could lead to additional evidence.

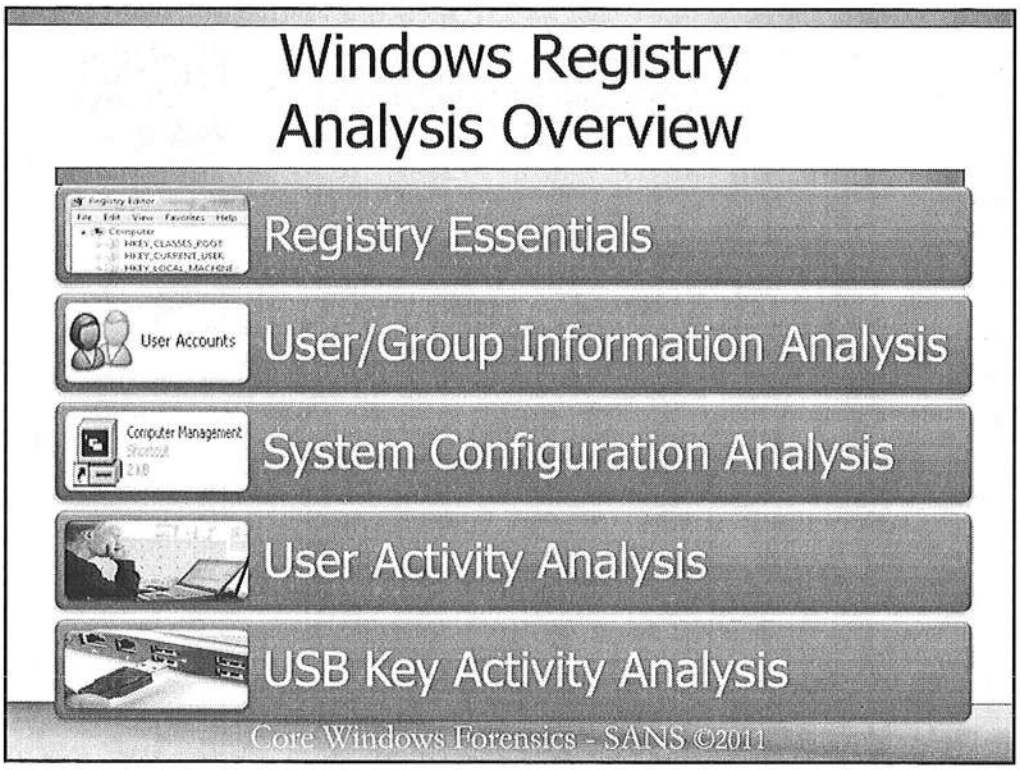
Answer to question 9: Searching the e-mail archives for the IP address identified in question 7 will find one additional hit. In this case, it is not online drug SPAM. Instead, it is a phishing e-mail looking for personally identifiable information (PII).




The first section we are going to examine is registry forensics.




This page intentionally left blank.



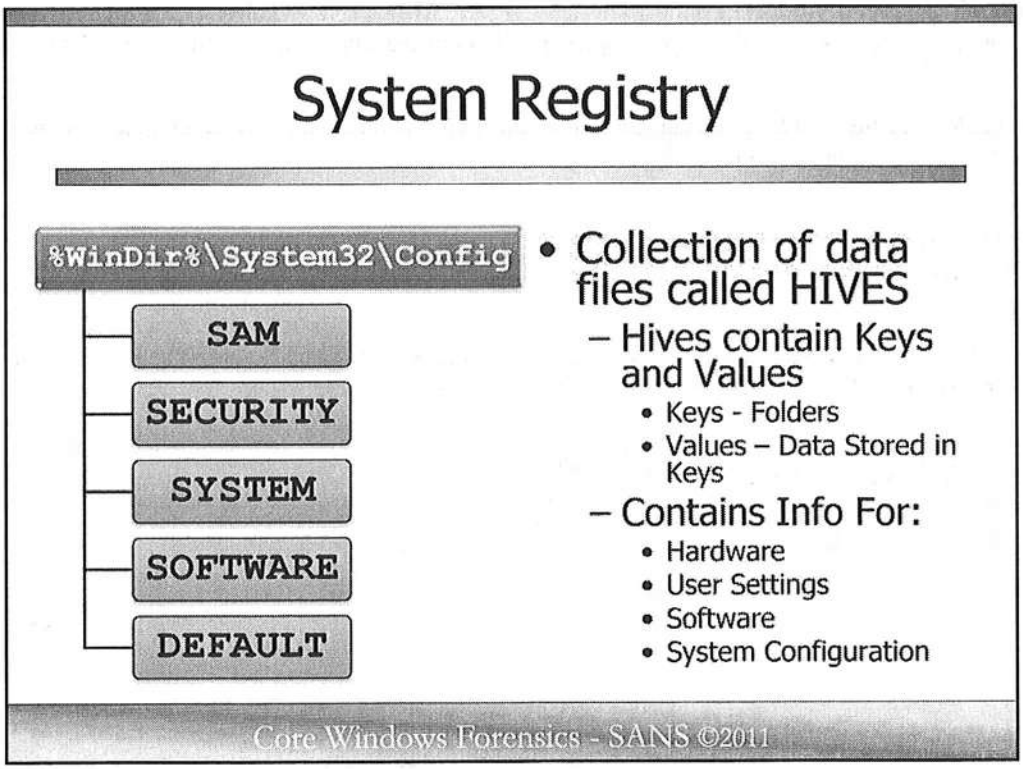

There are five sections that were going to go through for Windows registry analysis. We will start with going through Windows registry essentials. This will give you the background that is necessary to understand how to read the registry information that might be useful to your investigation.

The second section will examine user and group information that is found inside the registry. This will allow us to examine last login times as well as specific users that might have had access to this machine.

Third, we will examine the way the system is configured. Through registry analysis, we will discover the last IP address of this machine or even be able to identify the last wireless access points that it joined.

The fourth section is probably the most useful for forensic analysis. We can track user activity via Windows registry analysis. You can examine documents that have been recently opened, applications that have been recently run, or even the files that were recently searched for by the user.

The final section is probably the most famous, registry analysis section. We will guide you through a step-by-step methodology to examine USB key registry artifacts that are left on the machine.




The registry is a collection of database files that store vital configuration data for the system. It stores software, hardware, and system component information. The registry can detail the software that has been installed, system configuration, recently used files, and startup programs. The registry can be viewed and manipulated using "regedt32". You can also load each hive from a compromised system that was imaged.

The registry has four root keys:

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS

The registry can be accessed either on a live system or off line. If off line, you will need to know where the registry files are stored. The majority of the files are stored under the \%WINDIR%\system32\config directory. These registry hives are DEFAULT, SAM, SECURITY, SOFTWARE, and SYSTEM. The files correspond to their meaning in the registry. All of these system files would reside under the HKEY_LOCAL_MACHINE hive under their sub-meaning SAM, SECURITY, SOFTWARE, and SYSTEM. The HKEY_LOCAL_MACHINE contains the system setup, startup files, machine configuration, and other default files.

The SYSTEM hive stores all the data that you see in the HKEY_LOCAL_MACHINE\SYSTEM including hardware and service configuration. It will also list the majority of the raw device names for volumes and drives on the system including USB keys.

The SOFTWARE hive stores data from the HKEY_LOCAL_MACHINE\SOFTWARE which is where all the settings for applications are found. Windows programs/products also have their settings stored here.

The NTUSER.DAT hive contains the configuration and environment settings which includes a slew of identifiable data pertaining to user activity.

The SAM hive contains all the local user accounts and groups. It is found on your system at HKEY_LOCAL_MACHINE\SAM.

The SECURITY hive contains all the security information that is utilized by the SAM and the operating system including password policies, membership of group information, and more.

Reference: http://support.microsoft.com/kb/256986

# User Registry

%UserProfile%

NTUSER.DAT

- Each individual user has a registry hive
- Can show specific details as to user activity on a machine
- One of the most critical aspects of computer forensics that should be examined

Location:

```
C:\Documents and Settings\username\NTUSER.dat (XP)
C:\Users\username\NTUSER.dat (Vista and Win7)
```

Can show critical forensic details related to the System and Individual Users

Core Windows Forensics - SANS ©2011

Each user that has used the machine will have his section. The registry can be used to enumerate the most recently used files. It can also show you the last files that he searched for on the hard drive. It can also show the last typed URLs that he inputted into his browser windows. It can also show the last commands executed on the system as well as the files that were opened. You can also witness which files were last saved to the Windows system.

The NTUSER.dat hive contains all the keys related to the specific user. It is found on your system at HKEY_CURRENT_USER.

# Registry Hives

- Collection of data files called HIVES
- When viewed in a Registry Viewer - Hive Names are used

- **HKEY_LOCAL_MACHINE (HKLM)**
  - Contain Hives
    - SAM
    - SECURITY
    - SYSTEM
    - SOFTWARE
- **HKEY_CURRENT_USER (HKCU)**
  - Contain Hive
    - NTUSER.DAT

| Registry Hive | Nickname | File |
|---|---|---|
| HKEY_LOCAL_MACHINE\SAM | HKLM\SAM | SAM |
| HKEY_LOCAL_MACHINE\Security | HKLM\Security | SECURITY |
| HKEY_LOCAL_MACHINE\System | HKLM\System | SYSTEM |
| HKEY_LOCAL_MACHINE\Software | HKLM\Software | SOFTWARE |
| HKEY_USER or HKEY_CURRENT_USER | HKCU | NTUSER.DAT |

Core Windows Forensics - SANS ©2011

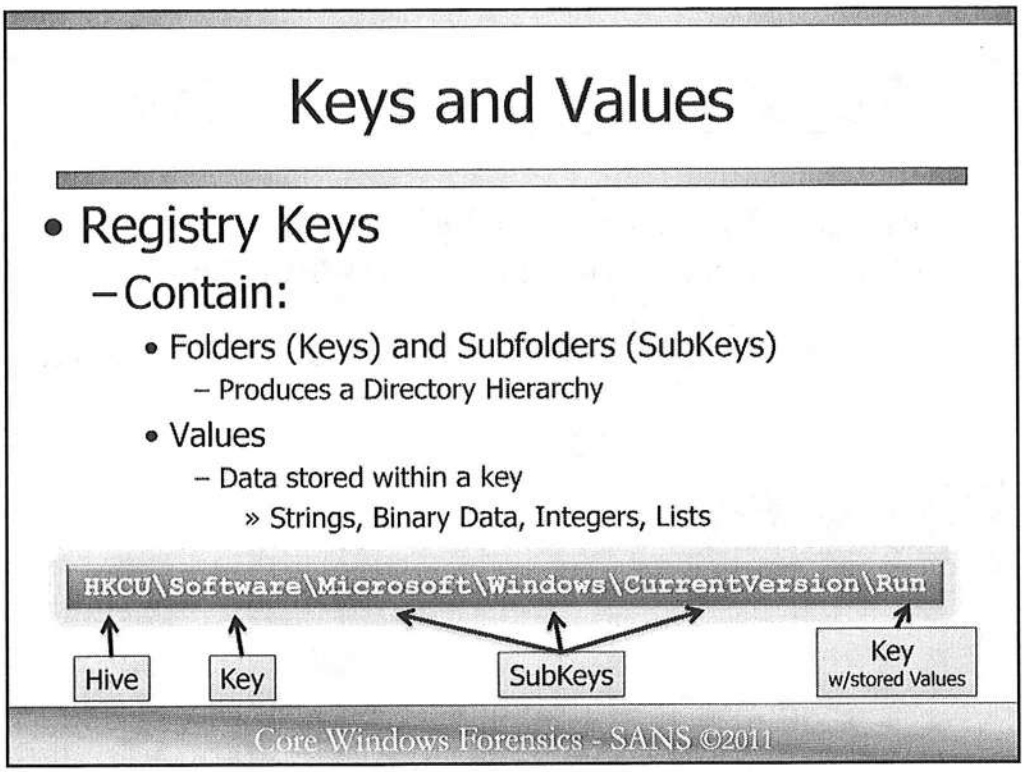
Registry hives on a live, powered on, machine look much different than they would when you analyze it offline. Most individuals, when reading up on the registry hives online, will encounter their formal names.

**HKEY_LOCAL_MACHINE\SAM**
**HKEY_LOCAL_MACHINE\Security**
**HKEY_LOCAL_MACHINE\System**
**HKEY_LOCAL_MACHINE\Software**
**HKEY_USER or HKEY_CURRENT_USER**

These are commonly found when you run regedit and you can easily open up the HKEY_LOCAL_MACHINE key and see the sub-keys which include our 4 main hive files (SAM, Security, System, Software) from the C:\Windows\System32\config directory. For short, their nicknames reduce the HKEY_LOCAL_MACHINE to HKLM when written.

When you are examining a user's hive, you are usually browsing to HKEY_USER (if more than one user is logged in) or HKEY_CURRENT_USER (the currently logged in user).

Reference: http://msdn.microsoft.com/en-us/library/ms724877.aspx



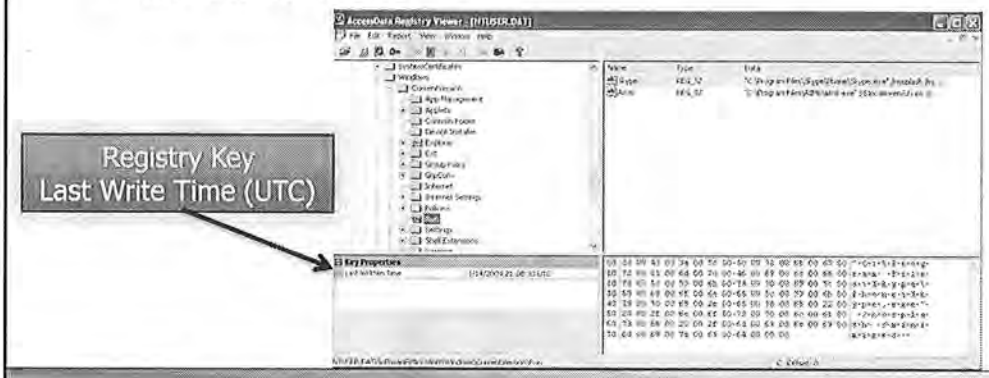

Registry keys are very similar to file folders in a file system. They can contain sub-keys (sub-folders) and values (files). Usually when you write the location of a registry key you would write the full key so that an individual can easily find it. One of the things that will help out the investigator greatly will be the fact that each registry key will have a last write time or last time of modification. This is useful to know when a value changes within a key and it contains something that is pertinent to your case you can map that time to other activity that was occurring on your machine.

What kind of data can be stored inside a value? Well, pretty much anything, but you usually find a combination of strings, binary (hex) data, integers, and lists when you are performing computer forensics.

[redacted]

# Registry Key Last Write Time

- Timestamps in Registry:
  - Every **KEY** in each HIVE has a **Last Write Time**
  - Time stored in UTC



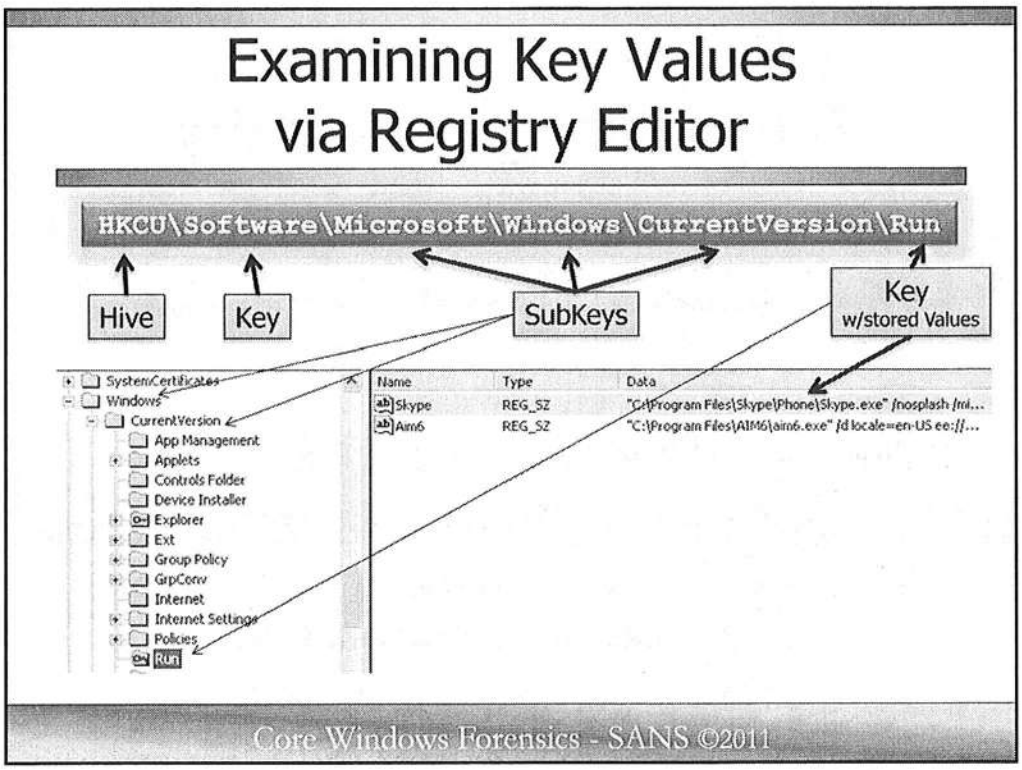

Core Windows Forensics - SANS ©2011

The last write time is tracked for every key on the machine. Embedded into the registry itself, a timestamp of the last update of any of the key values will be stored. This value, which is usually displayed in UTC, has been crucial to forensic investigations of the registry.

The time, while by itself might not be important, it gives us the time the activity being investigated took place. For example, you could see the last time the subject ran a search for files that contain certain words related to a SECRET project that he had given a competitor. Knowing the time that this occurred and pairing that data with additional system facts such as when a user was logged on and if the files were copied over to a USB key begins to tell a compelling story.

Remember, the last write time is updated when a value is updated or added to the key. Different keys are updated at different points depending on how the program is written. If it is a key that is not well known in the forensic community, I always recommend testing to verify how the last write time is updated for that particular action on the system.

Forensic Note: Many novice investigators get tripped up by time zone conversions in many Windows artifacts that they are looking at. Make sure you are positive whether the value you are examining is UTC or set at the time zone of the local machine.

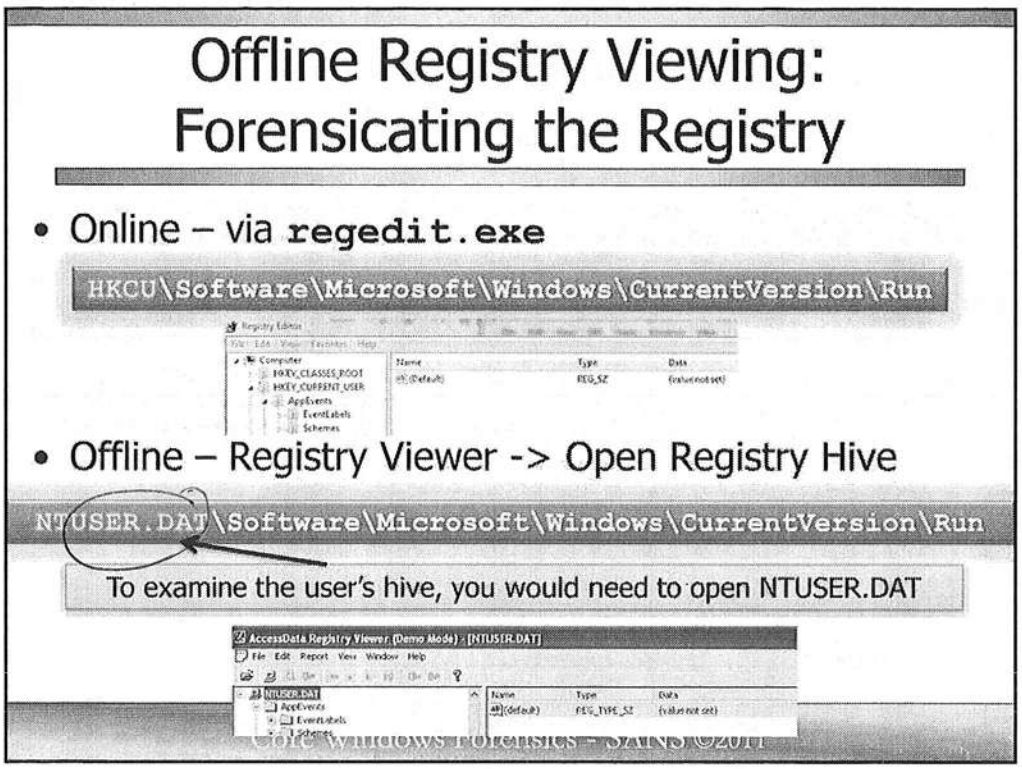



This is a graphical representation of how Hive, Keys, and SubKeys relate to each other when viewed in a registry editor or forensic tool such as Registry Examiner. This is a key that we will be examining later. The key "Run" has several values that are contained within it (Skype and AIM).

The key "Run" will have a last write time. However, that time will be set when the last value was added or modified inside the key. In some cases, a value list is provided that keeps track of the order that values were added making it easy to identify when the last value was added/modified thus changing the parent key's last write time. An investigator can utilize this information to see when the last value was added which can mean many things. In this case, it would show the last program added that would automatically launch when the machine is booted. More on specific keys and their meanings for the forensic investigator shortly.

Browsing to a specific key is easy once you understand that it is another name for folder and value another name for a file. While not exactly the same, it is close enough for this course.

For detailed information on the makeup of the Registry, reference the extremely technical and detailed article by Mark Russinovich (author of Windows Internals) http://technet.microsoft.com/en-us/library/cc750583.aspx




There is a difference when an investigator looks at a registry via the registry editor on a live machine versus parsing a hive file that was recovered on the forensic analysis workstation. In fact, it looks different only at the beginning of the key.

Generally, when you execute regedit you will see registry hives presented properly by their formal names (HKEY_CURRENT_USER). However, once you are working offline the registry files will need to be manually loaded into a registry viewer and examined. Typically you accomplish this by pointing the registry viewer to the location on your image where the files will be located. Usually the user's home directory or C:\Windows\System32\config.

When a file is opened offline, everything looks the same except the formal name is now missing and the parent directory is now simply showing the file you opened. In the case above, you can see that NTUSER.DAT was opened via the viewer. But if you viewed that same location in the live registry, NTUSER.DAT is translated to its formal name (HKEY_CURRENT_USER).

Do not make the novice mistake and forget the mappings of the formal name of a hive to the file name. The list for reference is only a few slides back if you need to remind yourself how the formal name maps to a specific file.

From here forward in this course, we will reference the location as if you were viewing the file offline via a viewer rather than on a live machine. But it is always important to know how the registry values are named especially when writing a report.

# Answer the Following Questions Below in Your Notes:

1. What are the four core hive files and where are they located?
2. What is the name of the user hive file?
3. Where is the user hive file located on Windows XP?
4. Where is the user hive file located on Windows Vista/ Win 7?
5. How many user hive files can exist on a Windows machine?
6. What is the difference between a key and a value?
7. Describe why the last write time of a key is important.

8. Extra Credit: Where will the user hive files reside in a Windows Active Directory Environment with roaming profiles enabled? [redacted]



1. What are the four core hive files and where are they located? [redacted]

2. What is the name of the user hive file? [redacted]

3. Where is the user hive file located on Windows XP? [redacted]

4. Where is the user hive file located on Windows Vista/Win7? [redacted]

5. How many user hive files can exist on a Windows machine?

6. What is the difference between a key and a value?

7. Describe why the last write time of a key is important.

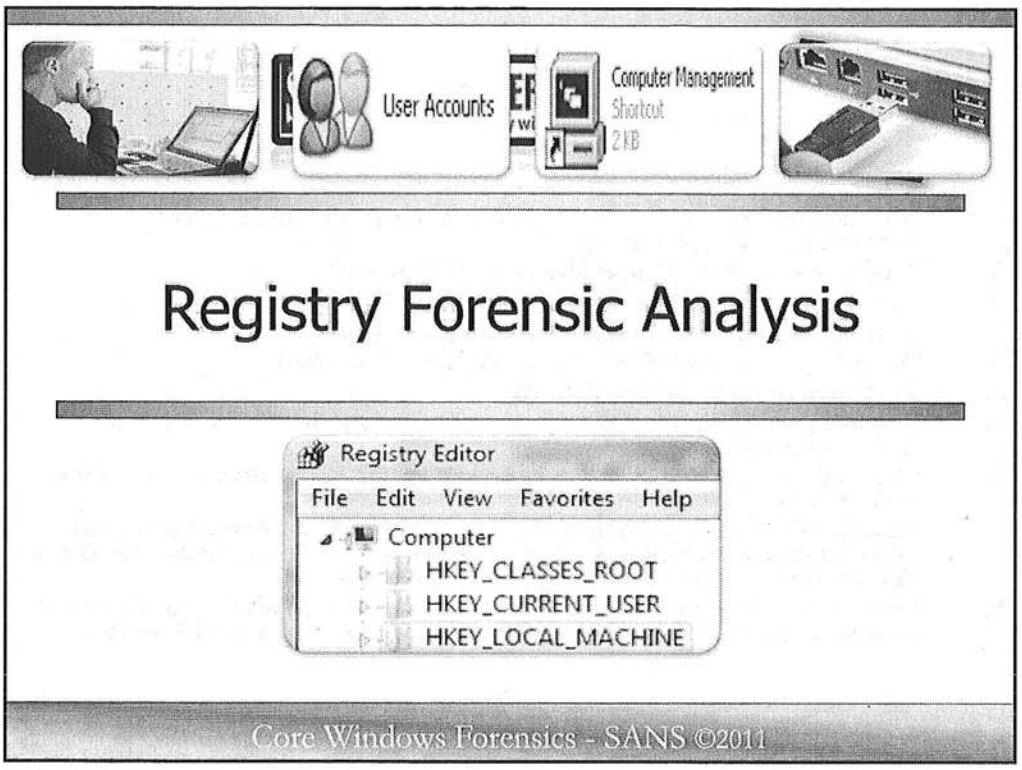
8. Extra Credit: Where will the user hive files reside in a Windows Active Directory Environment with roaming profiles enabled?

# Answers

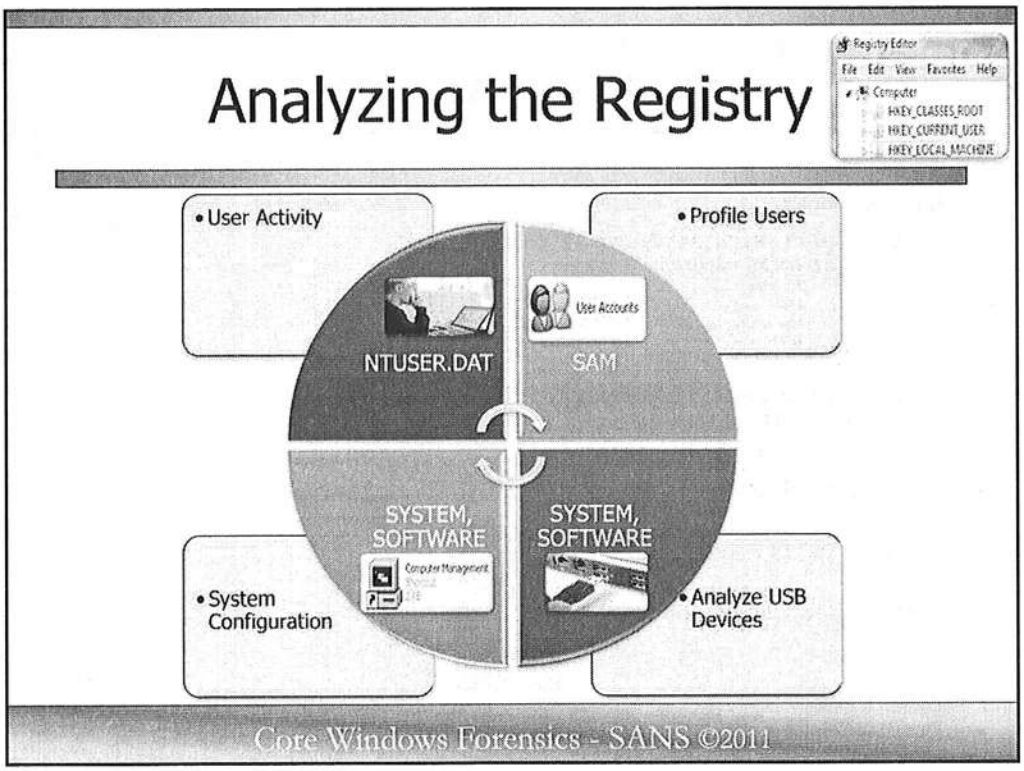
1. What are the four core hive files and where are they located? ***System, Software, SAM, Security***
2. What is the name of the user hive file? ***NTUSER.DAT***
3. Where is the user hive file located on Windows XP?
   ***C:\Documents and Settings\Username\NTUSER.DAT***
4. Where is the user hive file located on Windows Vista/Win7?
   ***C:\Users\Username\NTUSER.DAT***
5. How many user hive files can exist on a Windows machine? ***One NTUSER.DAT for every user***
6. What is the difference between a key and a value? ***Key is similar to a folder. Values contain data.***
7. Describe why the last write time of a key is important. ***An investigator can determine when a key was last written to which in many circumstances link an action of a user to the time of an event.***
8. Extra Credit: Where will the user hive files reside in a Windows Active Directory Environment with roaming ? ***The Domain Controller and Local System***

Core Windows Forensics - SANS ©2011

1. What are the four core hive files and where are they located? ***System, Software, SAM, Security***
2. What is the name of the user hive file? ***NTUSER.DAT***
3. Where is the user hive file located on Windows XP?
   *C:\Documents and Settings\Username\NTUSER.DAT*
4. Where is the user hive file located on Windows Vista or Win7?
   ***C:\Users\Username\NTUSER.DAT***
5. How many user hive files can exist on a Windows machine? *One* ***NTUSER.DAT*** *for every user*
6. What is the difference between a key and a value? *Key is similar to a folder. Values contain data.*
7. Describe why the last write time of a key is important. *An investigator can determine when a key was last written to which in many circumstances link an action of a user to the time of an event.*
8. Extra Credit: Where will the user hive files reside in a Windows Active Directory Environment? ***The Domain Controller and Local System***




This page intentionally left blank.




When we analyze the registry it is best to categorize the types of items we will want to find into categories. For this class, we created for categories of evidence we would like to parse.

The first is User/Group Profiling, which is accomplished through viewing the SAM hive. This information will tell us when a user last logged in, number of times logged in, and their SID for example. We will begin our analysis here usually.

The second is System Configuration which is accomplished through viewing the SYSTEM or SOFTWARE hives. This information will tell us the way a computer system is set up, the networks it has connected to, and the time zone in addition to other information.

The third is User Activity which is accomplished through viewing the NTUSER.DAT file in the subject's home directory. This information will tell us specific things the user was doing on the system. For example, we can tell the recent documents, last programs executed, and their commands typed into the machine.

The last and fourth is Analyzing USB devices which is accomplished through viewing the SYSTEM and SOFTWARE hives. Analyzing USB devices has become increasingly important for investigators to accomplish as tracking these devices is difficult and could make or break your case. Specific time is spent here today for such a critical skill for investigators.

# Registry Analysis

- System Hives
  - Located in %WINDIR%\System32\Config
    - HKEY_LOCAL_MACHINE
      - System
      - SAM
      - Security
      - Software
- User Hives
  - Located in %UserDir%\NTUSER.DAT
    - HKEY_CURRENT_USER

      C:\Documents and Settings*username*\NTUSER.dat (XP)
      C:\Users*username*\NTUSER.dat (only Vista and Win7)
- Registry Analysis Tools on your XP SIFT Workstation

  –Access Data's Registry Viewer

  –Registry Ripper (Regripper)

Core Windows Forensics - SANS ©2011

In order to analyze the forensic residue that is found in the registry it is useful to know what tools we might use and where the evidence resides in the disk image.

System Hives

Located in **%WINDIR%\System32\Config**

**HKEY_LOCAL_MACHINE**

**System**

**SAM**

**Security**

**Software**

User Hives

Located in **%UserDir%\NTUSER.DAT**

**HKEY_CURRENT_USER**

**C:\Documents and Settings*username*\NTUSER.dat (XP)**

**C:\Users*username*\NTUSER.dat (only Vista and Win7)**

System hives are where we will be working from for operating system wide forensic artifacts including USB devices, user information, and system configuration. The name of these locations is usually prepended with HKEY_LOCAL_MACHINE. But when we examine the specific registry hive located in the C:\Windows\System32\config directory via a viewer or an automated tool it will utilize only the file hierarchy.

For the user activity forensic artifacts, we will find most of that information sitting in the NTUSER.DAT file.

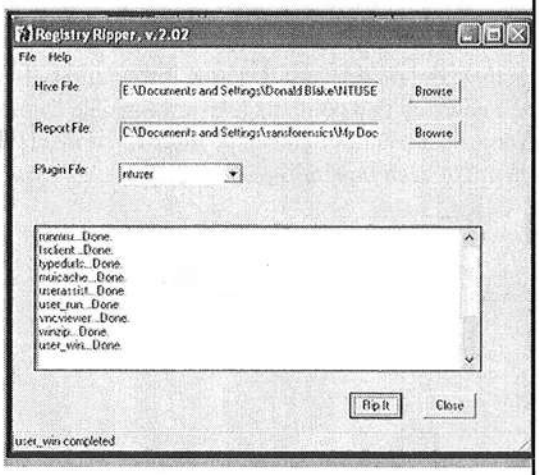
The two tools we will utilize for Registry Work are Access Data's Registry Viewer and Harlan Carvey's RegRipper.

Access Data Registry Viewer works just like a regular registry editor (minus the editing part). The registry viewer will automatically decode and mine certain keys for data without requiring the user to deobfuscate a key's value (such as ROT13 encoding).

Harlan Carvey's RegRipper is not a registry viewer. Regripper will automatically parse the contents of keys/values that contain relevant Registry Information. Regripper is plugin based and can be extended fairly easily once a new key that contains relevant forensic data is discovered. Currently, RegRipper can be utilized on the major forensic attributes that can be discovered inside the different hive files. RegRipper can specifically pull information out of each one individually.

# RegRipper

- Automated Registry Ripping
  - By Harlan Carvey
  - `http://www.regripper.net/`
- Windows GUI interface
  - Hivefile
  - Report File
  - Plugfile (Hivetype)




Core Windows Forensics - SANS ©2011

RegRipper is an automated HIVE parser that can parse the forensic contents of the SAM, SECURITY, SYSTEM, SOFTWARE, and the NTUSER.DAT HIVES that it is pointed at. You can even use this to forensically mine the contents of restore point registry files. RegRipper, written by Harlan Carvey, can be found at www.regripper.net and is based off a "plugin" system. Each plugin in the plugins directory will parse a separate forensic artifact in a specific registry hive. You can add additional plugins to the directory as a result and it will parse the new data the next time the tool runs.

There is a GUI wrapper located on your desktop on your XP SIFT Workstation. To easily run RegRipper, click on rr.exe and it will launch the GUI so you can manually load and set your output file as needed. Just like in the command line version, you have to set your HIVETYPE via the plugin file dropdown. You have ntuser, sam, security, software, and system.

Regripper.exe can be invoked by pointing the HIVE FILE at the hive you would like to mine forensically. You also need to tell regripper the type of hive file it is (sam, security, software, system, ntuser) using the option with the PLUGINFILE option.

This is an example of regripper.

You will be utilizing regripper in your analysis shortly.

# RegRipper Output

ntuser.txt - Notepad

File Edit Format View Help

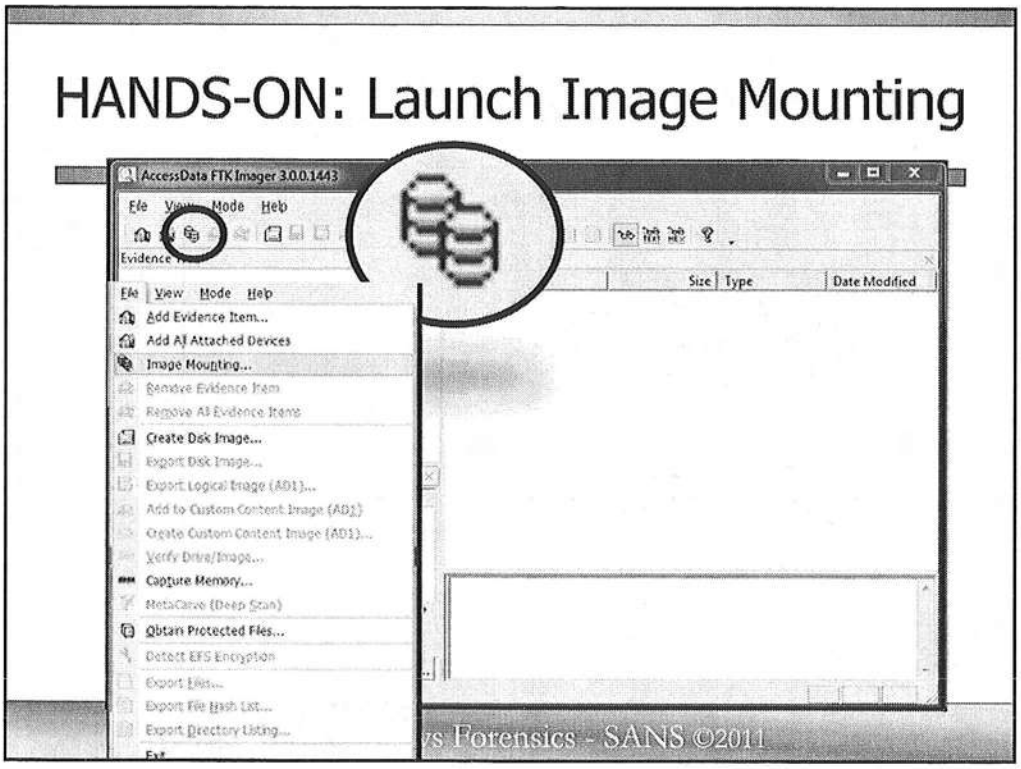
```
Logon User Name
Software\Microsoft\Windows\Currentversion\Explorer
Lastwrite Time [Fri Jan 16 23:19:34 2009 (UTC)]
Logon User Name = Donald Blake
----------------------------------------
ACMru - Search Assistant
Software\Microsoft\Search Assistant\ACMru
Lastwrite Time Wed Jan 14 21:03:45 2009 (UTC)
5603 [Wed Jan 14 21:02:34 2009 (UTC)]
        000 -> santorini

5604 [Wed Jan 14 21:02:47 2009 (UTC)]
        000 -> 055

5647 [Wed Jan 14 21:21:05 2009 (UTC)]
        000 -> \\falconii
        001 -> falconii

----------------------------------------
Adoberdr v.20080324
Adobe Acrobat Reader version not found.
----------------------------------------
Software\America Online\AOL Instant Messenger (TM)\Currentversion\Users not found.
----------------------------------------
Applets
Software\Microsoft\Windows\Currentversion\Applets
Lastwrite Time Fri Jan 16 23:13:45 2009 (UTC)

Software\Microsoft\Windows\Currentversion\Applets\Paint\Recent File List not found.
----------------------------------------
fileexts
Software\Microsoft\Windows\Currentversion\Explorer\FileExts

File Extension: .doc
Lastwrite: Fri Jan 16 23:13:45 2009
MRUList: a
  a => WORDPAD.EXE
```

Core Windows Forensics - SANS ©2011

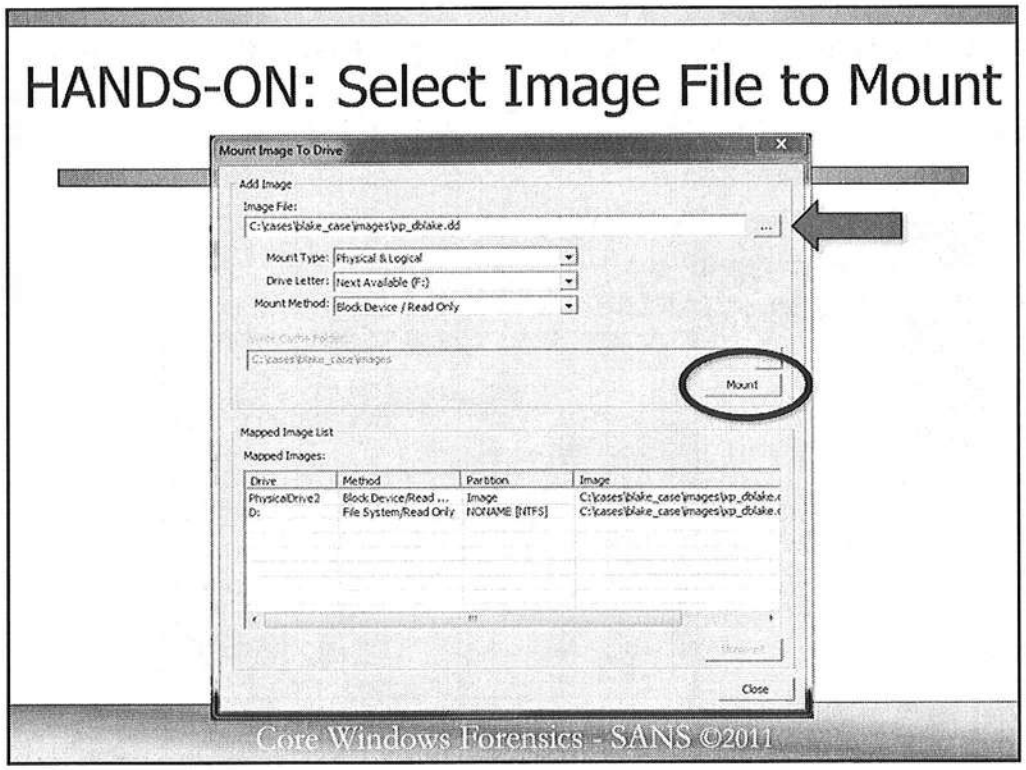
This is an example of a RegRipper Output file. In this file each new key that contains data is separated by a line of dashes. ----------

For each key, the key location is named and usually the last write time in addition to the data. In some cases, a short description is written, but in most cases it is missing. However, we will be going through this section today discussing the various keys and why their data is useful to examine.

Automated registry parsing is an incredible time saver when working on cases such as these. It will save you the need to examine each key individually via the Registry Viewers.



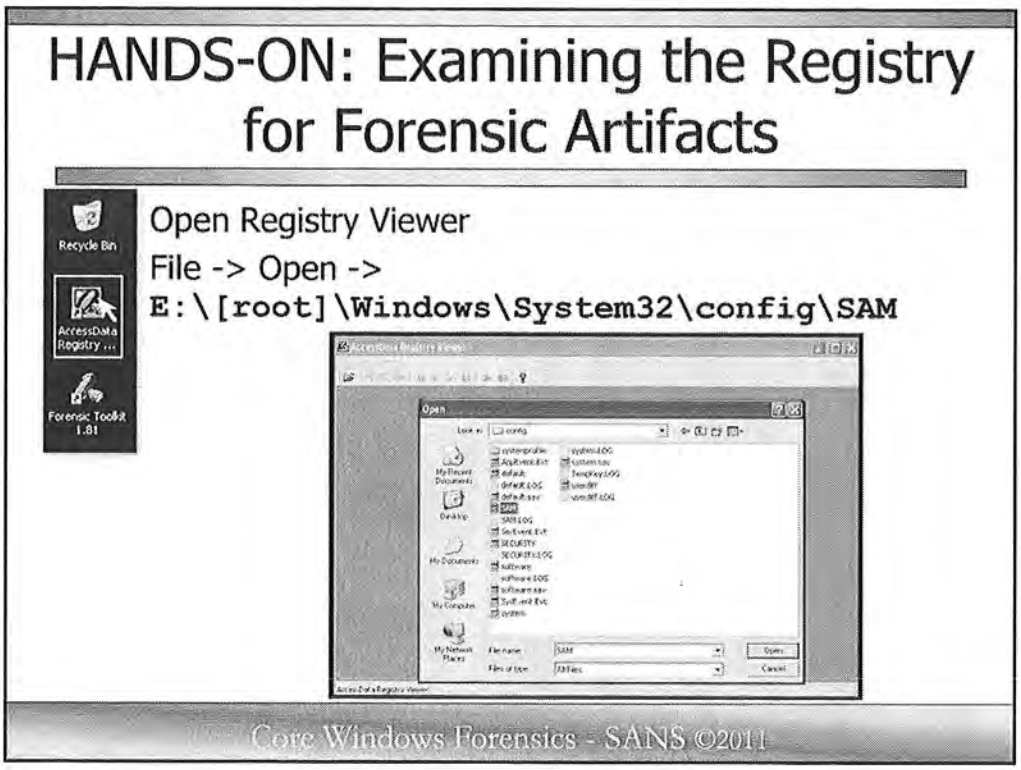

With FTK Imager open, either select the third icon from the left on the Tool Bar, or from the Menu Bar, select "**File**" then "**Image Mounting…**". If you are already have a forensic image added as evidence, you can simply right click on the image in the Evidence Tree window and select "**Image Mounting…**".




Click the "…" adjacent the "Image File" field. Brows to the location you image file is located. Let browse to the **C:\cases\blake_case\images** directory and select the "**xp_dblake.dd**" image file. Click "**Open**".

Now that you have selected the image file, the "Map Type" will default to the supported mapping based on the type of image that is selected. There are three different map types, Physical & Logical, Physical Only, and Logical Only. If the map type includes one of the "Logical" choices, the "Drive Letter" automatically defaults to the next available drive letter. You can optionally select the drive letter to assign the mount point manually.

Click "**Mount**" and the selected image file will be mounted. You can see in the bottom half of the "Mount Image to Drive" area the details of the mounted images in the "**Mounted Image List**".




Once you have your xp_dblake.dd image mounted, open up the Access Data registry viewer from your desktop.

For the first part of the examination we will be analyzing the SAM Hive.

Click File->Open **E:\[root]\Windows\System32\config\SAM.**

[redacted]

SANS COMPUTERFORENSICS and e-Discovery with Rob Lee

# Collecting User Information

## Analyzing the SAM

This page intentionally left blank.

## Profiling Users/Groups

- SAM Contains User/Group Information
  - Username
  - Security Identifier (SID)
  - User Login Information
    - Last Login
    - Last Failed Login
    - Logon Count
    - Password Policy
  - Group Information
    - Administrators
    - Users
    - Remote Desktop Users



Profiling a user is critically important to a case. It is useful to know how often a user has logged onto a specific machine, the last login time, the last failed login, and the password policy. In addition, many programs, like the Recycle Bin, use the user's SID (Security Identified) instead of the username. It is important to examine how to go about looking for this residue on a machine during an investigation.

On a multi-user system, how can we map a SID back to the original user if we only have a disk image to examine? Answer: We examine the SAM registry hive where the accounts are managed to see which user is mapped to which SID and answer all the above questions. All you need to do is import the SAM hive into your registry viewer and examine the contents.

# Profiling Local Users

**Purpose**

- Lists the local accounts of the system and their equivalent security identifiers

**Location**

- `SAM\Domains\Account\Users\Names`

**Why Key Is Useful?**

- Discover Usernames and the SID mapped to them
- Able to see a variety of information related to user activity
  - Last Login
  - Last Failed Login
  - Logon Count
  - Password Policy

Core Windows Forensics - SANS ©2011

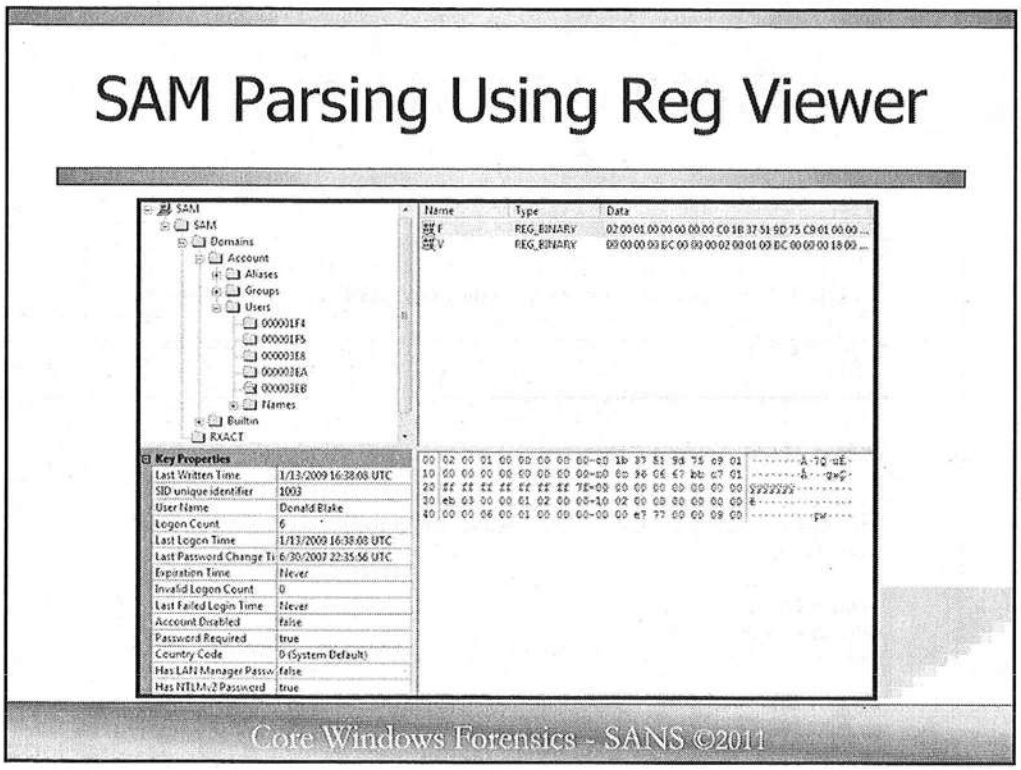
The first place we are going to look for information is the SAM. The SAM hive will help us enumerate all the users that have a profile on the machine. If your machine is a part of a domain, the SAM file containing the user profiles will be located on the Domain Controller.

For a forensic investigator there are several things we are looking for here.

1) Tying a username to a SID (Security Identifier). There are many artifacts on a Windows machine that point back to a user's SID and not their username. It is important to document the username and the corresponding SID for later purposes. This is a very important step.

2) Profiling the user. It is incredibly important to profile a user of a machine based on their login habits and mistakes. You can easily see the last login of each user and the total number of logins that this user has made. Both are important as it might tell you how active a specific user might be on a machine.

Notice all the times stored in the registry will be in UTC, so any time zone conversions will need to be accomplished.



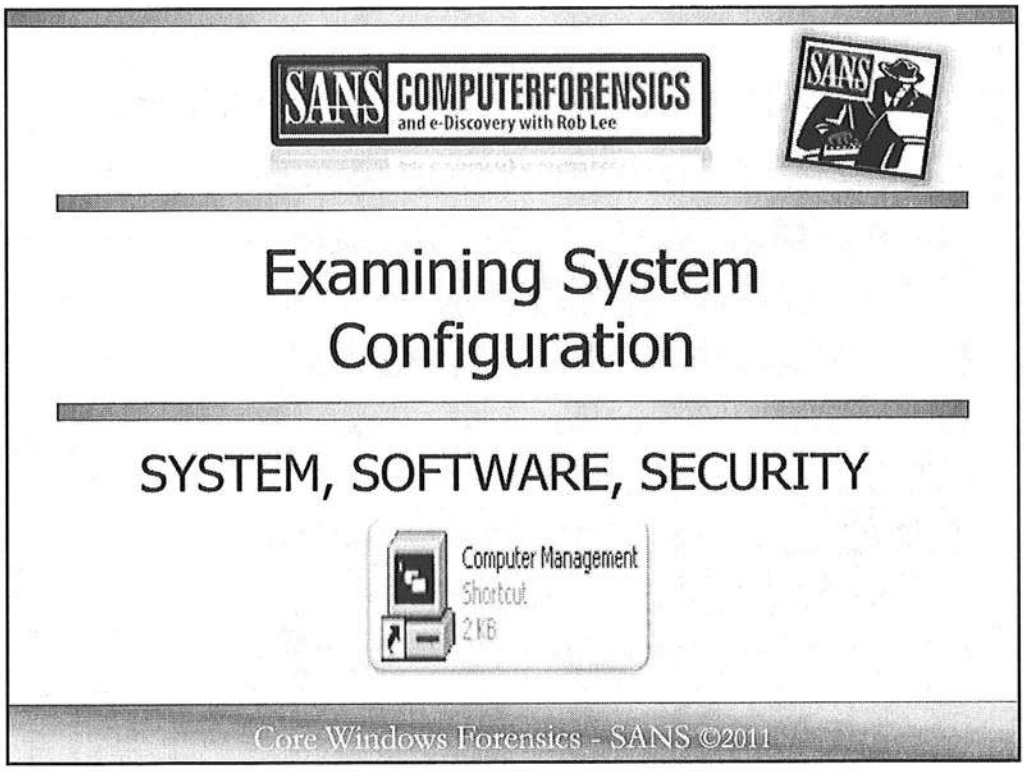

The information we are looking for is stored in the SAM hive. Above is an example using Access Data's registry viewer to examine the SAM in order to see the user's SID and other pertinent information. In this case, Donald Blakes's SID was 1003, which is useful later when we are examining data related to Donald's activities on the system.

You should also notice the last time Donald Blake logged in, the total number of times you've logged in, as well as the last time the password was changed.

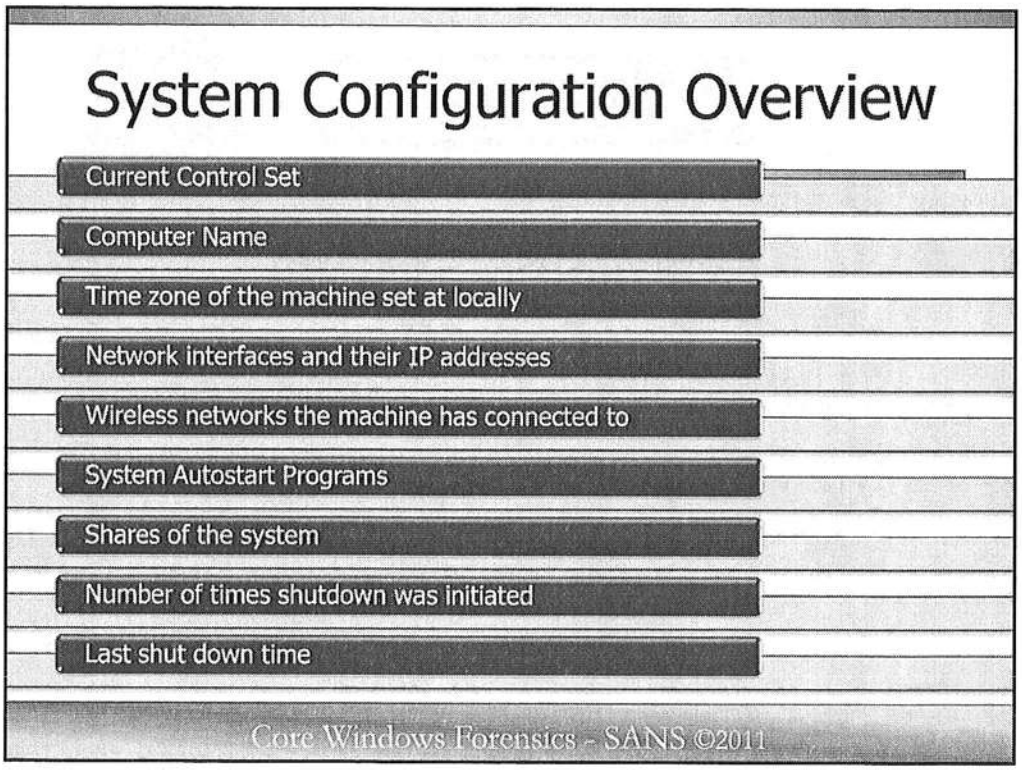
What does the Password Required field mean? Analysts have taken the "false" value to mean that the account does not have a password. This is not true. If the value is "false", it means that password policies (length, complexity, etc.) applied to the user accounts on the system do not apply to this account.

This is supported by MS KB article 305144 [1], which indicates that enabling the flag means that a password is not required.

[1] http://support.microsoft.com/kb/305144"




This page intentionally left blank.




There are many keys we will examine today that will address the various configuration of the system you are analyzing. In each instance, it is important to remember that the information could be utilized not only for system examination, but for network forensics as well.

I have spent extensive time comparing the keys between XP, Vista, and Windows 7. The keys that are listed here might contradict other information posted online. When that is the cases, I always double check my results and compare. In most cases, information found online or even in some books is incomplete. As this course evolves, my aim is to keep this section completely up to date with accurate and relevant information.

In any situation where you might find yourself analyzing data that could make or break your case, you should always double check the data location across multiple systems to ensure the results are repeatable and reproducible in every situation.

# Identify the Current Control Set (1)

**Purpose**

- Identifies which control set (ControlSet00x) is considered the *CurrentControlSet*
- Contains information about the system's configuration settings

**Location**

- `SYSTEM\Select`
- `SYSTEM\Select\Current`

**Why Key Is Useful?**

- Determines which ControlSet00x to use as the "*CurrentControlSet*"
- Once you identify which ControlSet00x is the "Current Control Set" you should focus your examination there

Core Windows Forensics - SANS ©2011

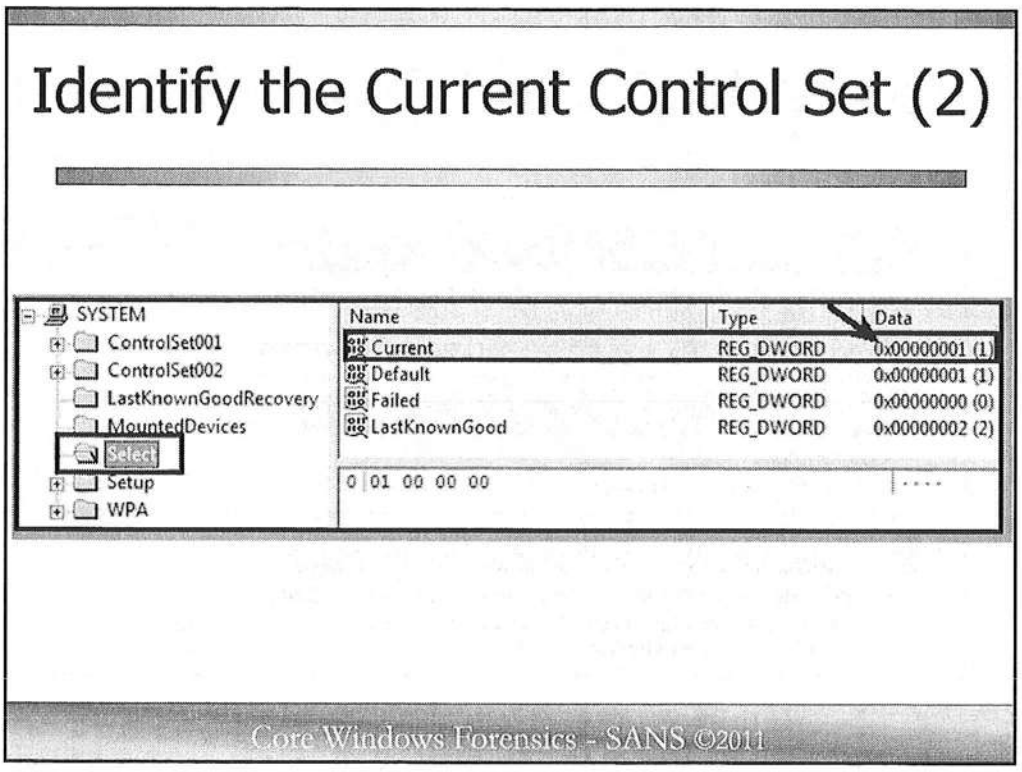
Why are there two ControlSets usually (ControlSet001 and ControlSet002).

ControlSet001 is typically the ControlSet that you just booted into the computer with. This is usually the most up-to-date version of the ControlSet.

ControlSet002 is the "Last Known Good" version. This version is the one that is considered good when the previous boot occurred in case something drastic happened during the current boot cycle.

Considering we are examining the ControlSet not in an active machine another way to look at this would be ControlSet001 would be the last successful boot of the machine and ControlSet002 would be the previous "successful boot" before that.

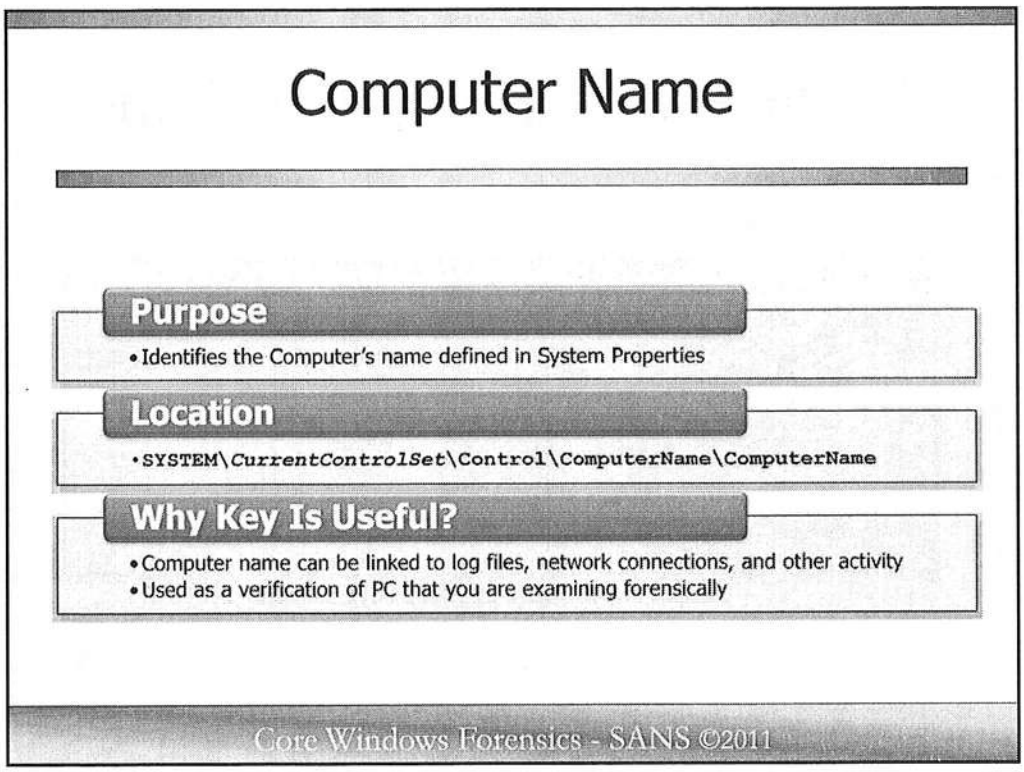
Reference: http://support.microsoft.com/kb/100010




When we open up an example SYSTEM hive we immediately notice that there are two registry paths for ControlSets. ControlSet001 and ControlSet002. In order for us to examine critical system information for this machine we need to know which of those two registry paths is the representation of the Volatile Hive "*CurrentControlSet*." To accomplish this we will examine the Select key.

In this example, we used the registry viewer to examine the SYSTEM hive locating the Select Key. The REG_DWORD, Current, for the Select key contains the number for the ControlSet00x that is "current." In this example the current value is set to 0x1 or "1". Therefore, the ControlSet001 is the registry path that is set to the "*CurrentControlSet*" and the one that should be examined in-depth.

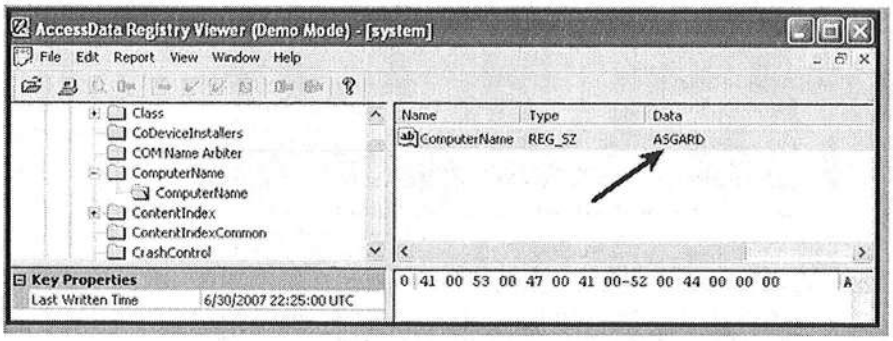
The second thing you could notice is that the "LastKnownGood" key is currently set to 0x2 or "2" meaning that ControlSet002 is the LastKnownGood snapshot of this registry path during the last successful boot that occurred on this machine.




The computer name is useful mainly for logging purposes and verification, but it should not go unnoticed. The machine name will also appear in other locations throughout the system. But what I generally use this for is to ensure I am on the correct PC that I was told in addition to noting the information for later use and correlation.

The computer name's location is in the SYSTEM hive.
**SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName**

# Document the Computer Name




Core Windows Forensics - SANS ©2011

This is an example of looking at the computer name via the registry.

# Time Zone Information (1)

**Purpose**

- Indentifies the current system time zone

**Locations**

- `SYSTEM\CurrentControlSet\Control\TimeZoneInformation`

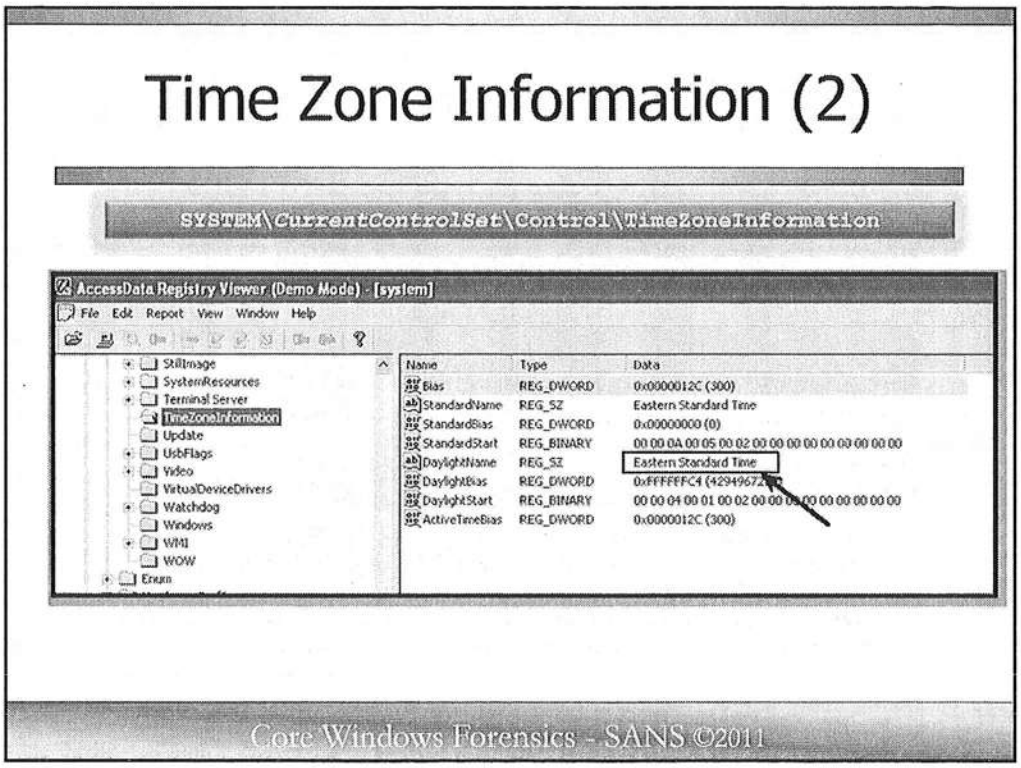
**Why Key Is Useful?**

- Time activity is incredibly useful for correlation of activity
- Internal log files and date/timestamps will be based off of the system time zone information
- You might have other network devices and you will need to correlate information to the Time Zone information collected here.



While most of the registry times and last write times will be recorded in UTC, the overall system time will have its log files and other time based information tied to the specific time zone. A good example of this is the file system timestamps on FAT file systems. The timestamps of the FAT file system will be updated and marked associated with the local time zone set in the control panel applet that controls that time zone.

A user could easily change the time zone. The last write time to this key will show the last time the time zone was changed for this machine.




This is an example of the location of the Time Zone information within the SYSTEM hive. In this case the Time Zone is set to Eastern standard Time. The Time Zone information is important to remember for temporal based information. Some data on the computer system is set in local time. Other data, including registry keys, is set in GMT or Universal Time Coordinated (UTC).

# Last Access Time On/Off

## Purpose

- Turns last access timestamp ON or OFF. If disabled, the last access timestamp recording in the filesystem will not occur

## Location

- **SYSTEM*CurrentControlSet*\Control\FileSystem**
- Located Locate NtfsDisableLastAccessUpdate -> If set to 0x1 then Access time stamps are turned on

## Why Key Is Useful?

- If the last access timestamp is turned off we will not be able to see when file data was last accessed by the system.
- This is a default setting in Windows Vista and Win7 but not other versions of Windows
- Without the last access time it might be more difficult to tell when programs were last opened by the user
  - Pictures being viewed
  - MP3s being played
  - Office Documents and PDFs being examined



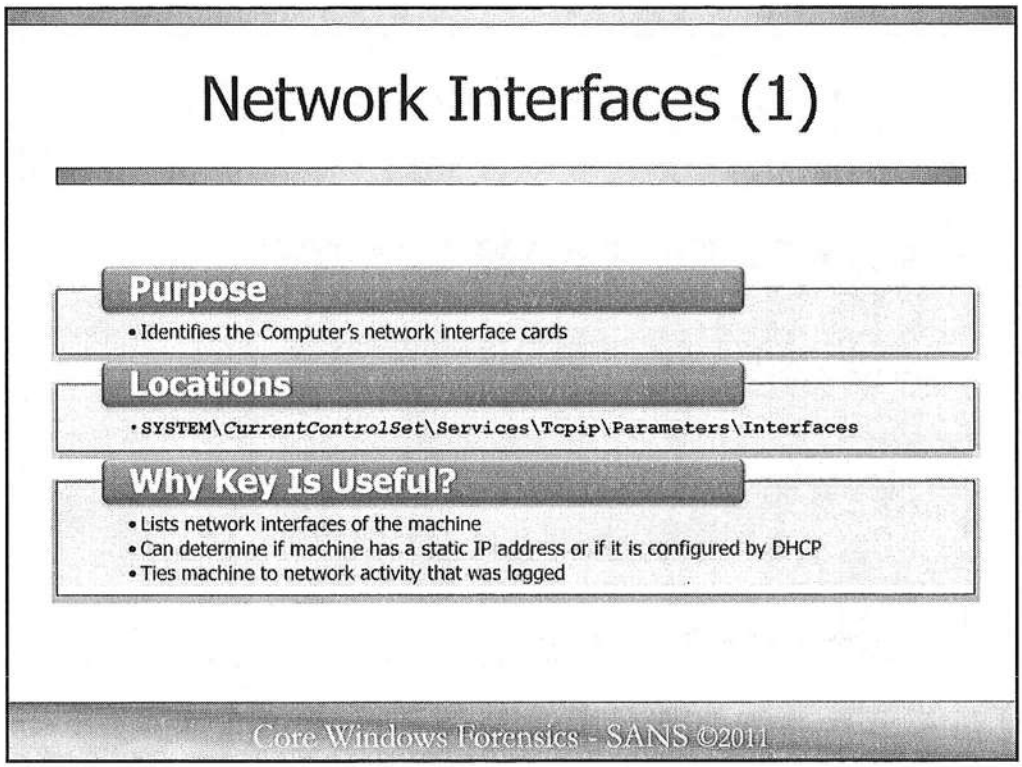
The last access timestamp is a critical timestamp for analysis of any file. It will show when the data of a file was last opened or when an executable was last executed. On most Windows operating systems, the access timestamp is turned on and will time/date stamp appropriately. However, with the release of Windows Vista and Win7, the last access timestamp is turned off across all versions. This means that the last access times of files will not update.

Where is the key located?

**SYSTEM\CurrentControlSet\Control\FileSystem**

Locate NtfsDisableLastAccessUpdate -> If set to 0x1 then Access time stamps are turned on

If the last access timestamp is turned off, we will not be able to see when file data was last accessed by the system. Without the last access time it might be more difficult to tell when programs were last opened by the user such as pictures being viewed, MP3s being played, or Office Documents and PDFs being examined.




Being able to list the network interfaces is very important. Not only is the network interface the key to many cases it could save you a lot of trouble in the long run.

The key's location is found here in the SYSTEM hive:
**SYSTEM*CurrentControlSet*\Services\Tcpip\Parameters\Interfaces**

The key that will contain much information that is good to know and understand. Specifically it will allow you to see the TCP/IP information configured, the IP address, the gateway, and other potentially useful information. If the machine was configured for DHCP, it will contain the DCHP IP address that was assigned, the subnet mask, and the DHCP server's IP address.

This information could be incredibly useful to a case that has any network based information collected. Again, this is something that is consider essential and basic information to be displayed about the machine when a report is compiled.

# Network Interfaces (2)

Core Windows Forensics - SANS ©2011

This is an example of the key for the Network Interfaces.

# Vista/Win7 Network List (1)

## Purpose

- Identify networks that the computer has been connected to that are considered part of an Intranet.
- Networks could be wireless or wired.

## Locations

- SOFTWARE\Microsoft\WindowsNT\CurrentVersion\NetworkList\Nla\Cache\Intranet

## Why Key Is Useful?

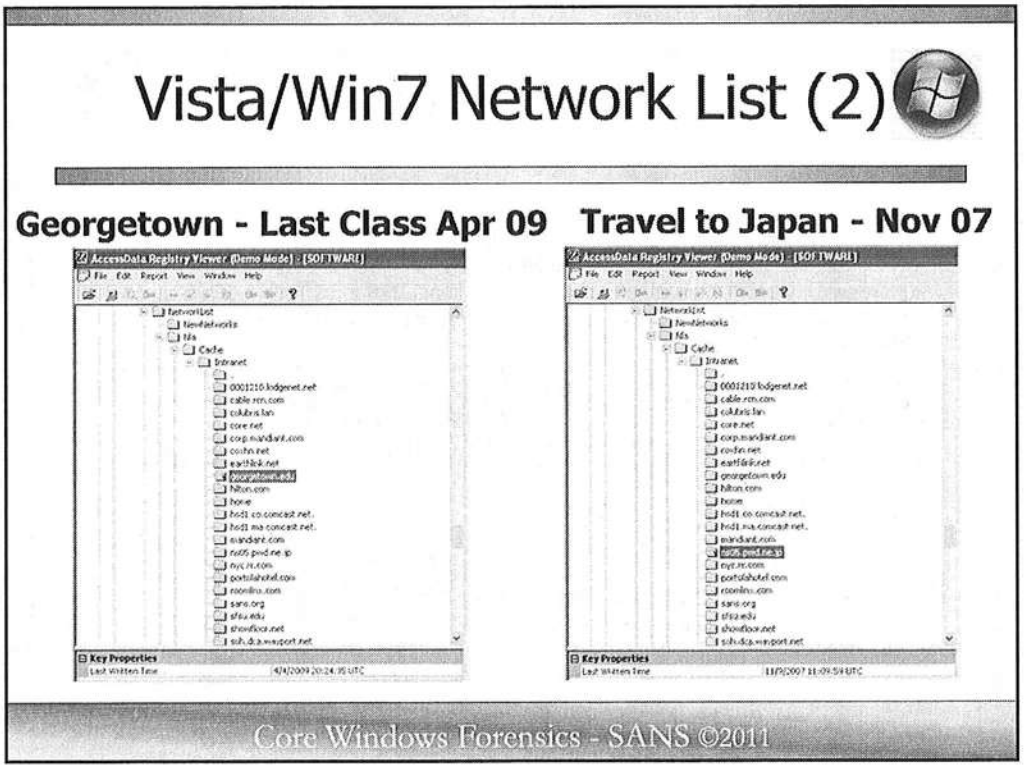
- Identifying intranets and networks that a computer has connected to is incredibly important
- Not only can you tell the intranet name, you can tell the last time the network was connected to based on the last write time of the key
- This will also list any networks that have been connected to via a VPN



A new key in Vista and Win7 is incredibly important to investigators. It will show a list of all the networks the machine has ever connected to via their DNS suffix. (e.g. sans.org) You can discover the last time that network was connected to by examining the last write time of the key that maps to the network you are interested in investigating.

Identifying intranets and networks that a computer has connected to is incredibly important. Not only can you tell the intranet name, you can tell the last time the network was connected to based on the last write time of the key. This will also list any networks that have been connected to via a VPN on the Vista/Win7 operating system.

Investigators will find this key useful due to the fact that in some cases, that just by examining this key, it might be able to give the you the geo-location of where this laptop might have been based on identifying the networks that it attached to and when.

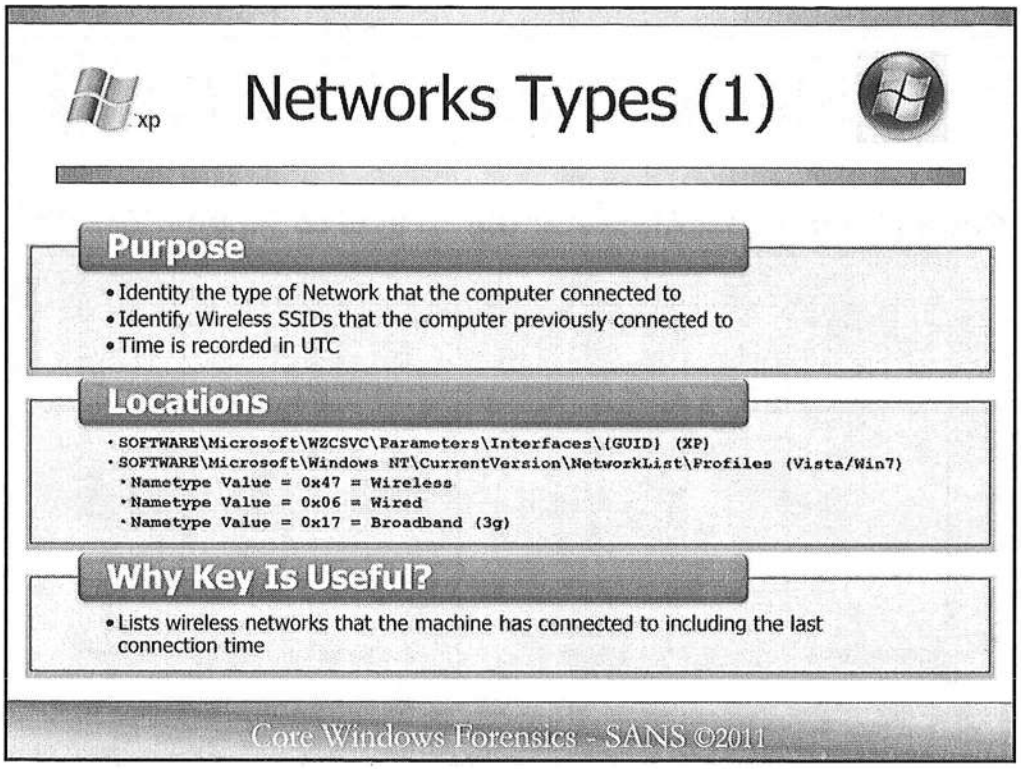



Not having a readily available Vista or Windows 7 machine to use for an example, I decided to examine my own registry hives on the laptop I was using for school and for examples when I teach forensics. What I found was quite enlightening.

In the first example, I could easily tell the last day I was in class at Georgetown University - April 04, 2009. It appears that the key is updated every time a person connects to the network. As a result, you can easily tell when the last time a user connected to a network.

The second example comes from the second week I owned this laptop. I purchased it in mid-October 2007. The laptop shows my first trip overseas to Japan. The date shows that I was connected 11/9/2007. This is also a useful way to show why the times displayed in UTC are so important. It doesn't matter which Time Zone you are in, you can always figure out the time of day that it actually occurred without fail.

In the end, being able to tell the last time a specific network a user connected to their computer could be very important, especially in cases where a user could claim the opposite.




For Windows XP: The final network information that we will examine is the wireless network settings found in the Windows wireless zero configuration settings. These settings are used for a machine to automatically remember previous access points that it has connected to and their respective SSIDs. The SSIDs are a unique security identifier that the machine uses to know that it has seen this network before and, if allowed, connect to it again.

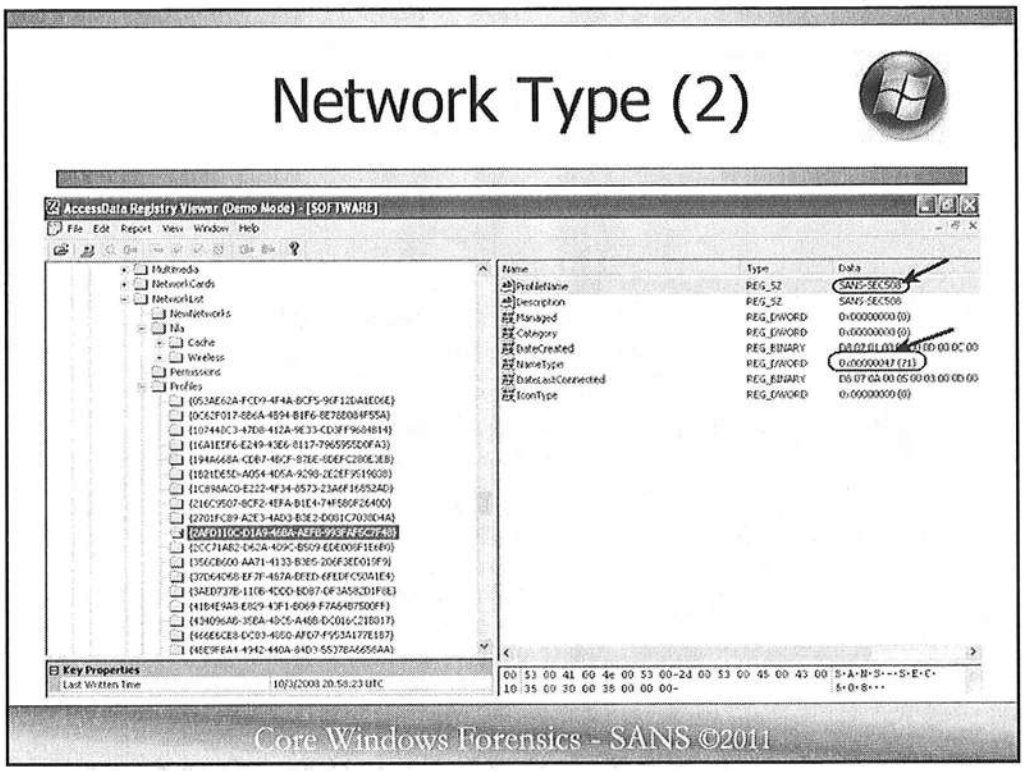
There are many cases where knowing that a machine was in proximity to a location and connected to that specific network could make or break a case.

The key's location is **`SOFTWARE\Microsoft\WZCSVC\Parameters\Interfaces\{GUID}`**

Underneath the GUID identifiers you will see the specific wireless networks that the machine was connected to. The last write time is the last time this machine was connected to that network in UTC. Again, very useful information for an investigator.

If the machine is a Vista or Windows 7 Machine, the key is in a much different location. The key is now located in **`SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles`** . Each sub-key located in this key will contain a GUID that will contain the network name and the network type. The network type will be listed as either 0x47 (hex) for wireless, 0x06 (hex) for wired, or 0x17 (hex) for broadband (3g) networks.

Being able to identify the network type will be useful in determining how a user connected to a network name that was defined earlier in the course.




In this screen capture, you can see that this Vista PC connected to an SSID called SANS-FOR508. The date the SSID was last connected to was 10/3/2008 at 20:58 UTC.

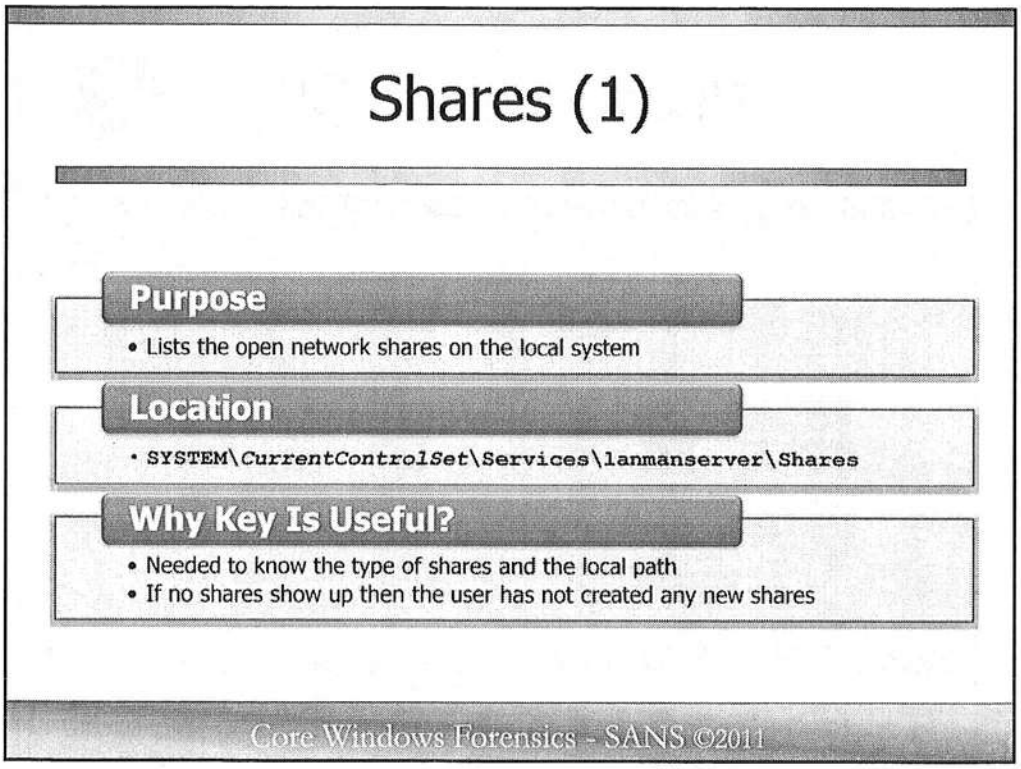
You can clearly see the network name in this key. You can also see the NameType which in this case is 0x47 (hex) which means that this network was a wireless network with the SSID of SANS-FOR508.

The other types of Nametype include

**Nametype Value = 0x47 = Wireless**

**Nametype Value = 0x06 = Wired**

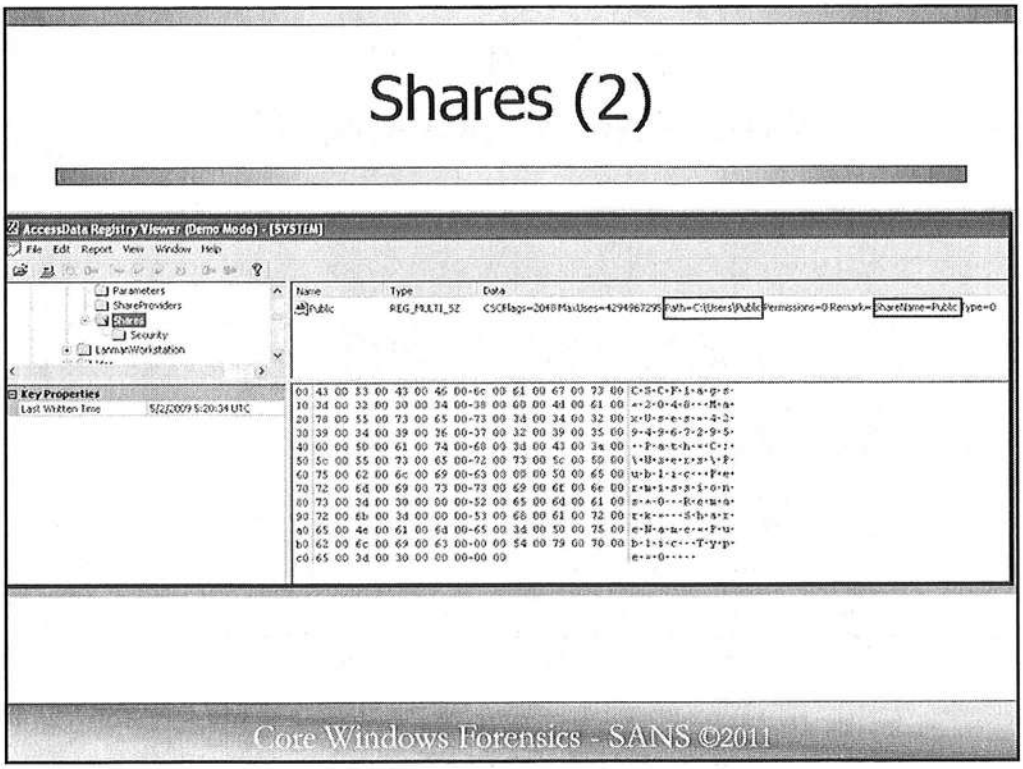
**Nametype Value = 0x17 = Broadband (3g)**



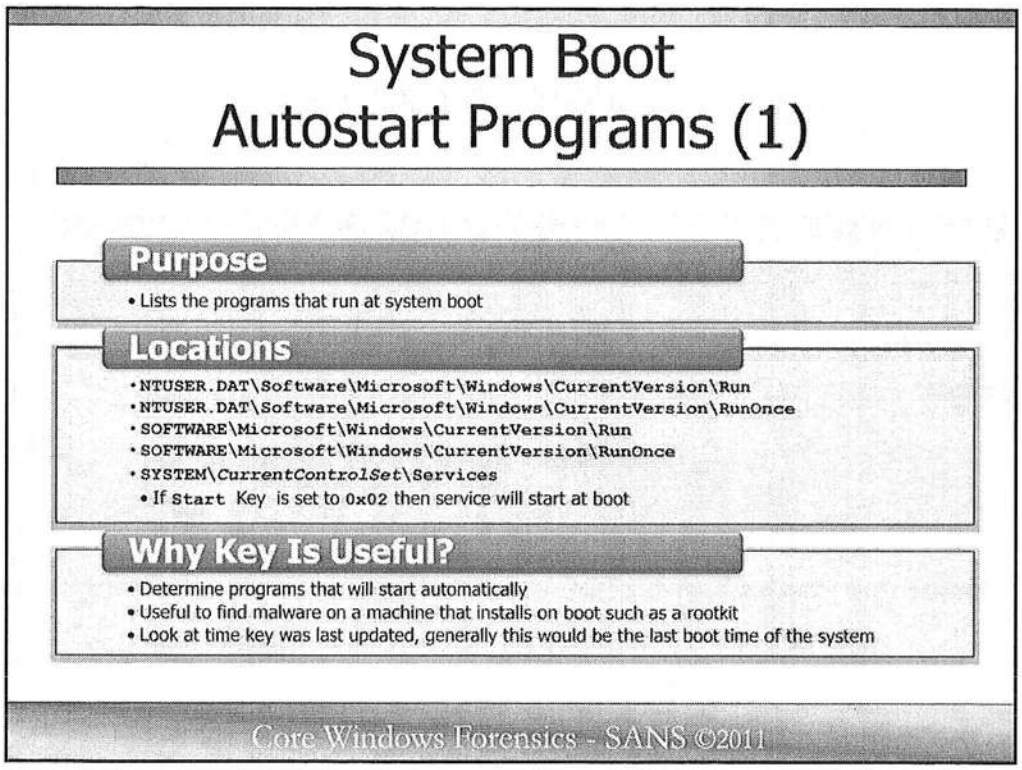

The file shares on a machine need to be examined. In many cases, the user does not know their machine has open shares where remote users could have the capability read/write/upload/download files. This is a serious possibility and could open up many arguments as to how files showed up on a specific machine.

If a user has shared out their entire hard drive, it might be hard to track how specific files ended up on their workstation. However, knowing that the shares were turned off could be also useful in preventing an argument that remote upload/download has occurred.

Finally, I have seen attackers remotely upload their malware on a remote machine and then schedule the malware to be run via the Windows system scheduler. This easy attack is most effective inside an organization, but also is quite prolific in the hacker community.




This is an example of an open share from a Vista machine. In this C:\Users\Public, which is one of the default shares on Microsoft Vista and Win7. How do we know for sure it is a Vista or Win7 operating system? The use of the C:\Users instead of C:\Documents and Settings directory specifically.




**NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Run**

**NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\RunOnce**

**SOFTWARE\Microsoft\Windows\CurrentVersion\Run**

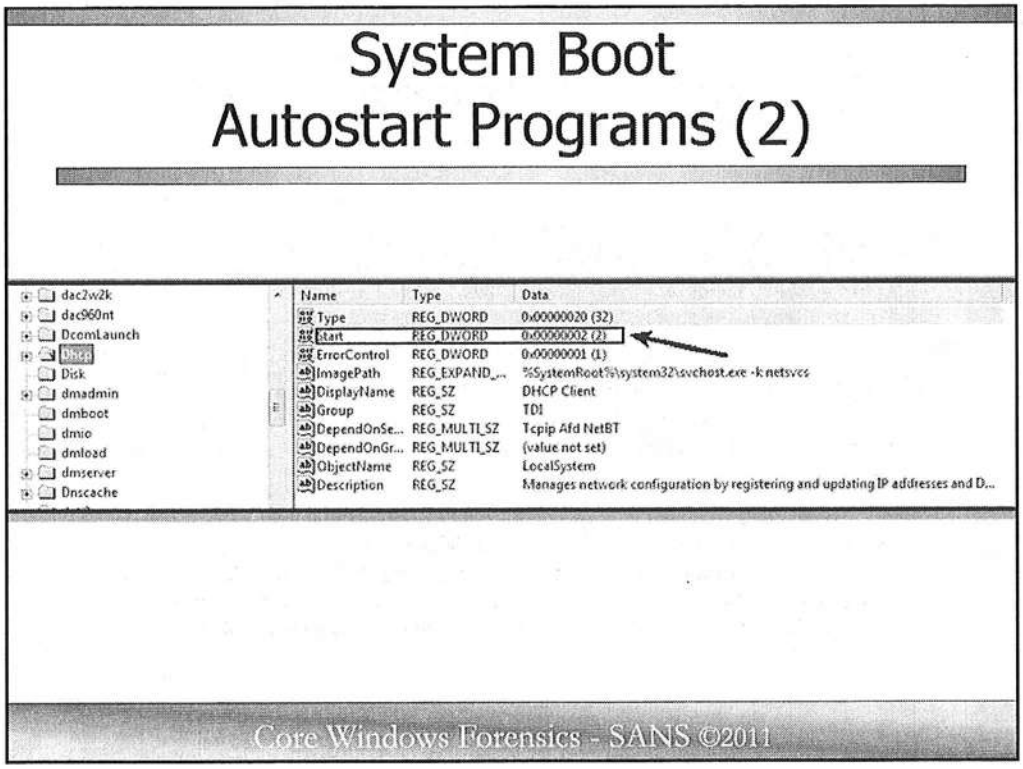
**SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce**

The above lists commands to be run each time a user logs on. These commands are not executed at boot. There is no specific order to the startup and these keys exists in both the NTUSER.DAT and the SOFTWARE hives.

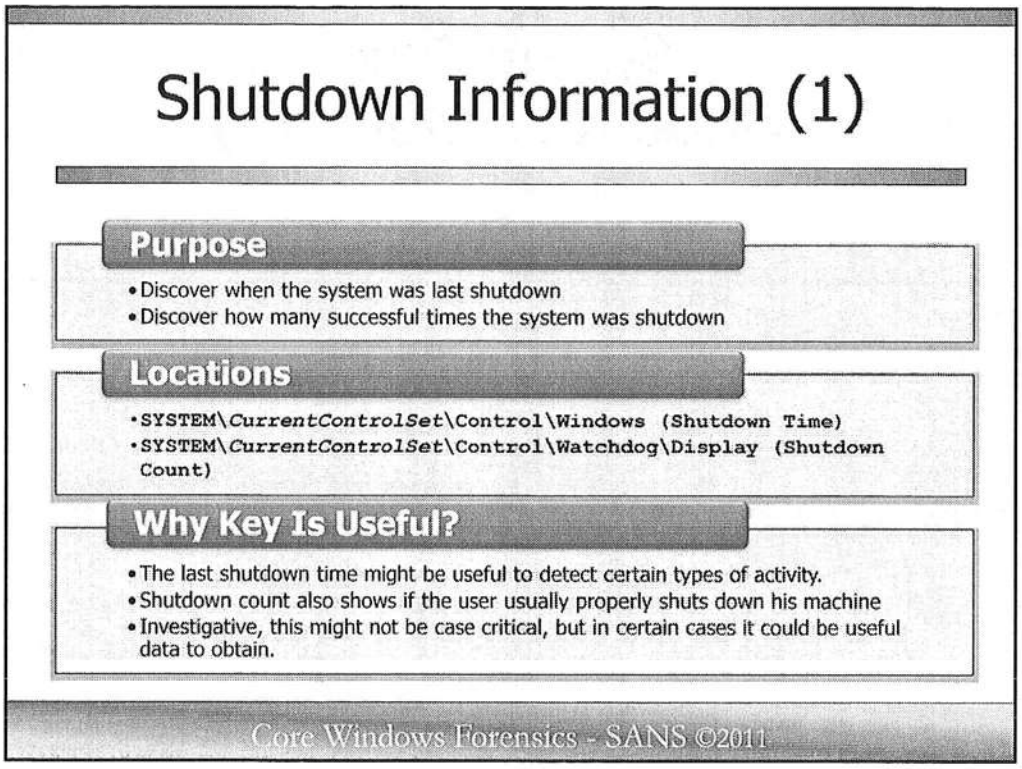
**SYSTEM*CurrentControlSet*\Services**

If **Start** Key is set to **0x02** then service will start at boot

For services started at boot, check the above key in the SYSTEM hive. If the Start key is set to the hex value 0x02, then the service will start at boot.




This is an example of looking at the files contained in **`SYSTEM\CurrentControlSet\Services`**. If **`Start`** Key is set to **`0x02`** then service will start at boot.

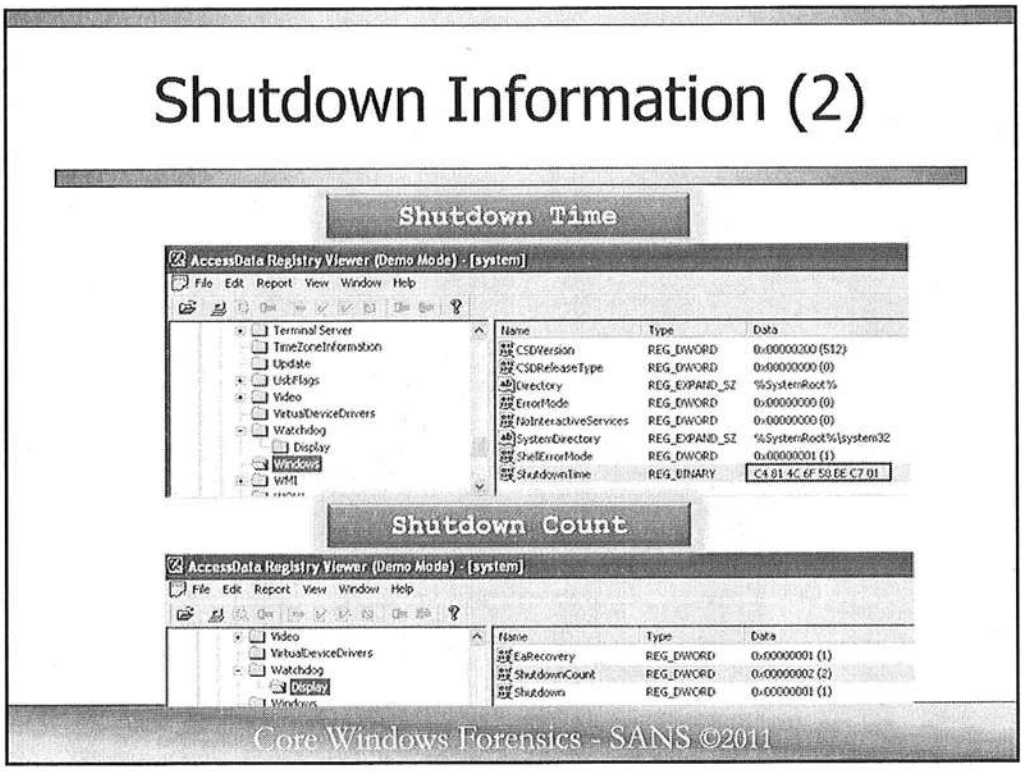



Sometimes you need to know when the system was last shutdown in addition to how many successful times the system was shutdown. The following keys will be useful.

**SYSTEM*CurrentControlSet*\Control\Windows (Shutdown Time)**

**SYSTEM*CurrentControlSet*\Control\Watchdog\Display (Shutdown Count)**

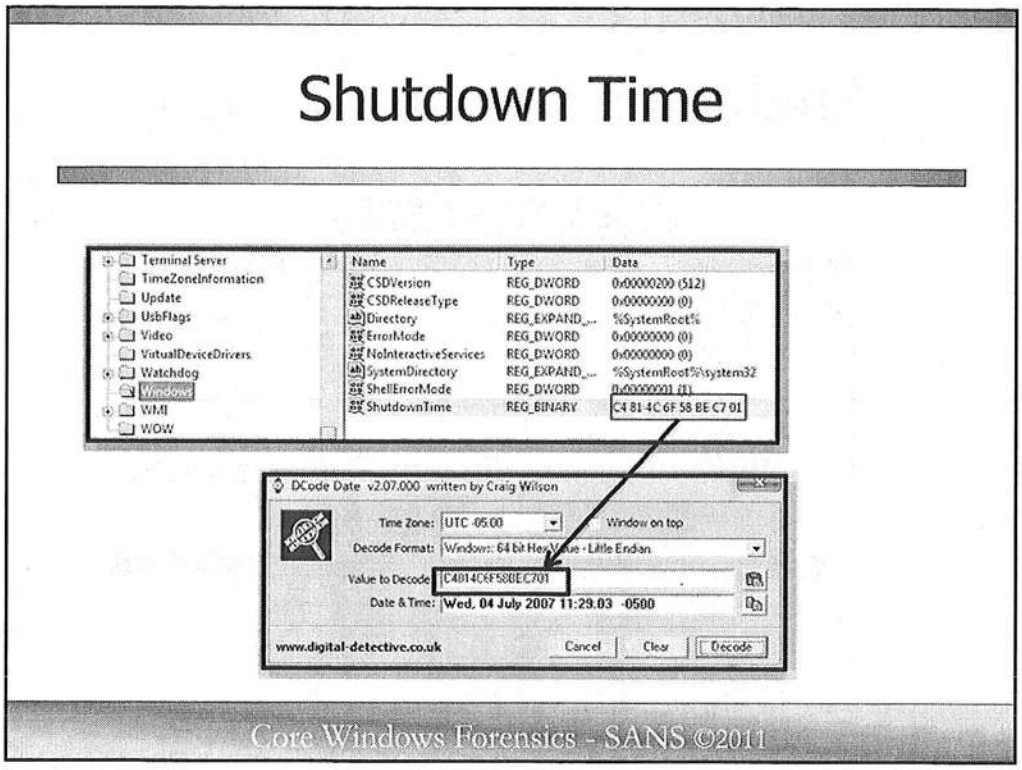
The last shutdown time might be useful to detect certain types of activity. Shutdown count also shows if the user usually properly shuts down his machine. Investigatively, this might not be case-critical, but in certain cases it could be useful data to obtain.




This is an example of looking at the Shutdown time and the Shutdown keys.

**SYSTEM\CurrentControlSet\Control\Windows (Shutdown Time)**
**SYSTEM\CurrentControlSet\Control\Watchdog\Display (Shutdown Count)**

Notice the shutdown time is in hex. This time is in Windows 64bit time. Luckily we have a converter for us that will show us what date is stored there.




By utilizing Decode Date on your desktop, we can write the values and press decode. It will tell us the date that is stored at that location.

# System Configuration Review

- Current Control Set
- Computer Name
- Time zone of the machine set at locally
- Network interfaces and their IP addresses
- Network MAC addresses
- Wireless networks the machine has connected to
- System Autostart Programs
- Shares of the system
- Number of times shutdown was initiated
- Last shut down time

Core Windows Forensics - SANS ©2011

This page intentionally left blank.

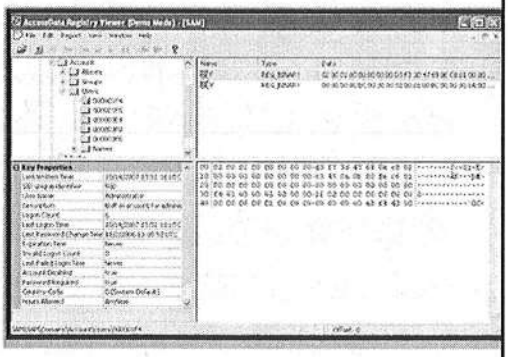
# HANDS-ON Registry Viewer

- Execute Access Data Registry Viewer

1. Open SAM Hive File located in:

   **E:\[root]\Windows\System32\config\SAM**

2. Examine USER information in **\SAM\Domains\Account\Users**



Core Windows Forensics - SANS ©2011

Execute Access Data Registry Viewer located on your desktop in your XP SIFT Workstation:

1. Open SAM Hive Files located in: **E:\[root]\Windows\System32\config\SAM**
2. Examine USER information in **\SAM\Domains\Account\Users**

# HANDS-ON:
# Detail User Information

**USERS**

- Username:_____
  - SID: _____
  - Last Login:_____
  - Login Count:_____
  - Last PW Change:_____
  - Invalid Logon Count:_____
  - Last Failed Login:_____
  - Has LM Pass? _____
  - Has NTLMv2 Pass?_____

**Answer these questions:**

1. What will the SID tell you?
2. What can you infer by the last login time?
3. What time zone are the times set in? [redacted]
4. If the user does not have either a LM password or a NTLMv2 Password, what will that mean?



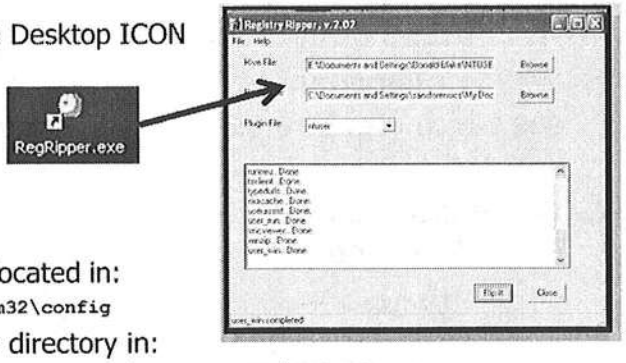
Username:_[redacted]
SID: _[redacted]
Last Login:_[redacted]
Login Count:_[redacted]
Last PW Change:_[redacted]
Invalid Logon Count:_[redacted]
Last Failed Login:_[redacted]
Has LM Pass? _[redacted]
Has NTLMv2 Pass?_[redacted]

Username:_[redacted]
SID: _[redacted]
Last Login:_[redacted]
Login Count:_[redacted]
Last PW Change:_[redacted]
Invalid Logon Count:_[redacted]
Last Failed Login:_[redacted]
Has LM Pass? _[redacted]
Has NTLMv2 Pass?_[redacted]

1. What will the SID tell you? *The SID is used to map a security identifier back to a user. This will be useful for recycle bin analysis in addition to mapping user activity back to a name instead of a number.*
2. What can you infer by the last login time? *This is the last time an individual logged into the machine using that specific user account.*
3. What time zone are the times set in? *The time is set at UTC or Universal Coordinate Time.*
4. If the user does not have either a LM password or a NTLMv2 Password, what will that mean? *The user account did not have a password.*

# HANDS-ON RegRipper
# Rip the SAM Hive

- Execute Regripper from Desktop ICON




1. Select SAM Hive Files located in:
   **E:\[root]\Windows\System32\config**
2. Select OUT file in case directory in:
   **C:\Users\sansforensics\My Documents\caseYYYYMMDD\SAM-RIP.txt**
3. Choose SAM plugin file
4. Click "Rip It"
5. Open **SAM-RIP.txt**

Core Windows Forensics - SANS ©2011

1. Select SAM Hive Files located in:
   **E:\[root]\Windows\System32\config**
2. Select OUT file in case directory in:
   **C:\Users\sansforensics\My Documents\caseYYYYMMDD\SAM-RIP.txt**
3. Choose SAM plugin file
4. Click "Rip It"
5. Open **SAM-RIP.txt**

# HANDS-ON:
# Detail Group Information

**Answer these questions:**

1. Why is it important to see which groups a user is a part of?
2. Which group should be of particular interest?

[redacted]

**GROUPS**

- Group:_____
  - Users in Group
    - _____
    - _____
    - _____
- Group:_____
  - Users in Group
    - _____
    - _____
    - _____



Group:_____

Users in Group

_____

_____

_____

Group:_____

Users in Group

_____

_____

_____

Group:_____

Users in Group

_____

_____

_____

Group:_____

Users in Group

_____

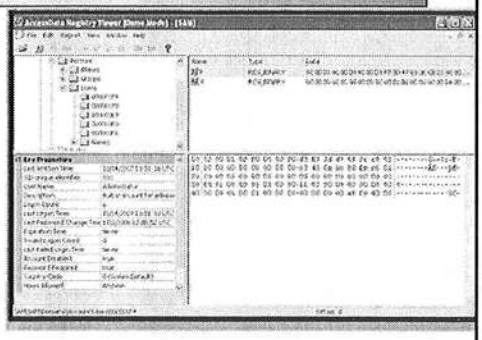
_____

_____

1. Why is it important to see which groups a user are a part of? *It is important to see which users are privileged users with administrator credentials. In addition, it is usually useful to determine which users could possible remotely control the machine via remote desktop protocol.*
2. Which group would be of particular interest if a user were a part of that group? *The administrators group and the remote desktop users.*

# HANDS-ON Registry Viewer
# Examine the SYSTEM Hive

- Execute Access Data Registry Viewer



1. Open SYSTEM Hive File located in:

**E:\[root]\Windows\System32\config\SYSTEM**

2. Fill out System Configuration Profile Guide in Appendix A

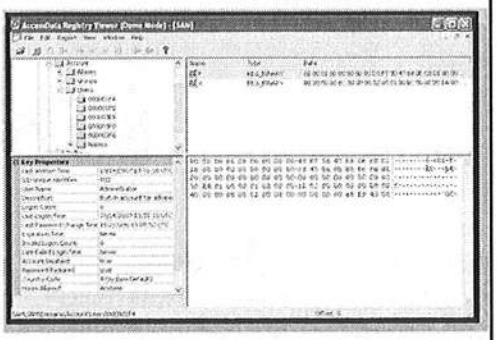
Core Windows Forensics - SANS ©2011

Execute Access Data Registry Viewer located on your desktop in your XP SIFT Workstation:

1. Open SYSTEM Hive File located in: **E:\[root]\Windows\System32\config\SYSTEM**
2. Fill out System Configuration Profile Guide in Appendix A.

# HANDS-ON Registry Viewer
# Examine the SOFTWARE Hive

- Execute Access Data Registry Viewer



1. Open `SOFTWARE` Hive File located in:

**`E:\[root]\Windows\System32\config\SOFTWARE`**

2. Continue to Fill out System Profiling Form in Appendix A

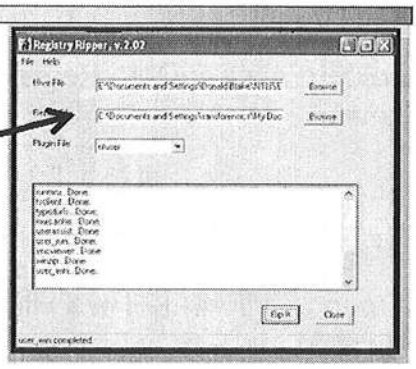


Execute Access Data Registry Viewer:

1. Open `SOFTWARE` Hive File located in:
   **`E:\[root]\Windows\System32\config\SOFTWARE`**
2. Continue to Fill out System Profiling Form in Appendix A.

# HANDS-ON RegRipper
# Rip the SYSTEM Hive

- Execute Regripper from Desktop ICON



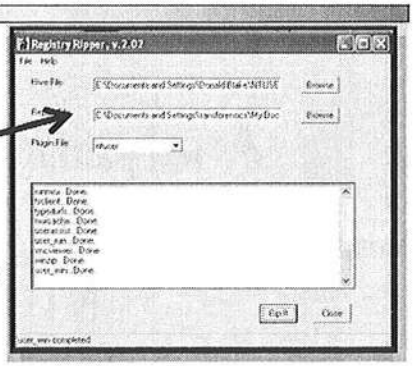

1. Select SYSTEM Hive Files located in:
   **E:\[root]\Windows\System32\config**
2. Select OUT file in case directory in:
   **C:\Users\sansforensics\My Documents\caseYYYYMMDD\SYSTEM-RIP.txt**
3. Choose SYSTEM plugin file
4. Click "Rip It"
5. Open **SYSTEM-RIP.txt** and compare against Appendix A

Core Windows Forensics - SANS ©2011

1. Select SYSTEM Hive Files located in:
   **E:\[root]\Windows\System32\config**
2. Select OUT file in case directory in:
   **C:\Users\sansforensics\MyDocuments\caseYYYYMMDD\SYSTEM-RIP.txt**
3. Choose SYSTEM plugin file
4. Click "Rip It"
5. Open **SYSTEM-RIP.txt** and compare against Appendix A

# HANDS-ON RegRipper
# Rip the SOFTWARE Hive

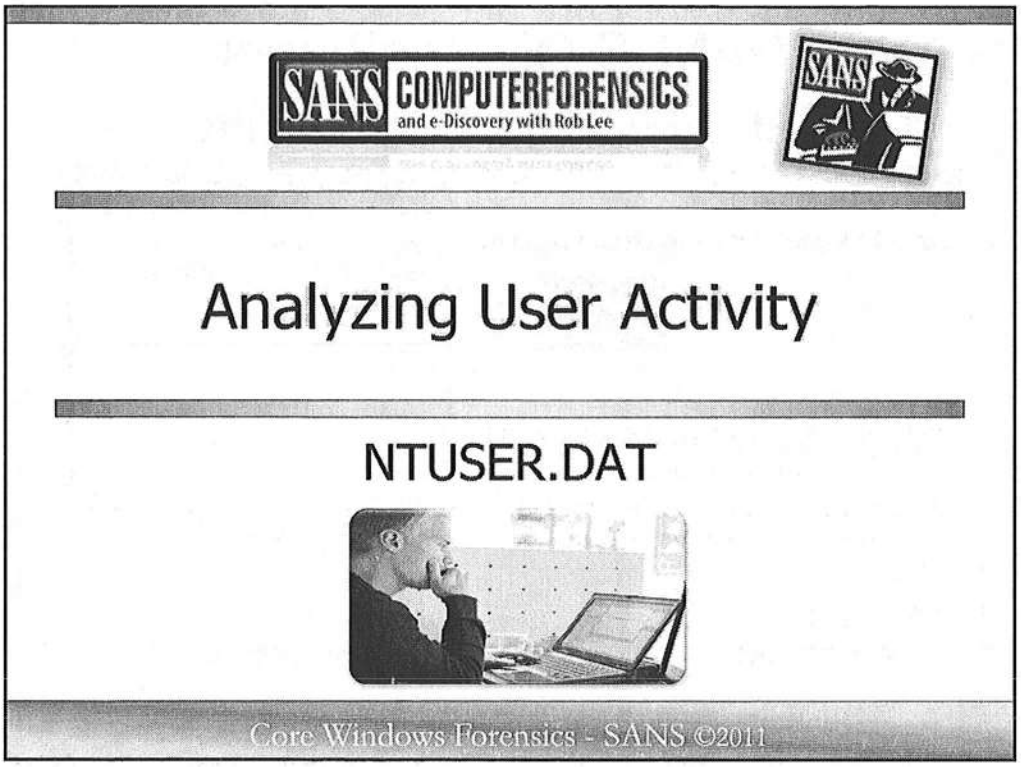
- Execute Regripper from Desktop ICON




1. Select SOFTWARE Hive Files located in:
   **E:\[root]\Windows\System32\config**
2. Select OUT file in case directory in:
   **C:\Users\sansforensics\My Documents\caseYYYYMMDD\SOFTWARE-RIP.txt**
3. Choose SOFTWARE plugin file
4. Click "Rip It"
5. Open **SOFTWARE-RIP.txt** and compare against Appendix A

Core Windows Forensics - SANS ©2011

1. Select SOFTWARE Hive Files located in:
   **E:\[root]\Windows\System32\config**
2. Select OUT file in case directory in:
   **C:\Users\sansforensics\MyDocuments\caseYYYYMMDD\SOFTWARE-RIP.txt**
3. Choose SOFTWARE plugin file
4. Click "Rip It"
5. Open **SOFTWARE-RIP.txt** and compare against Appendix A




This page intentionally left blank.

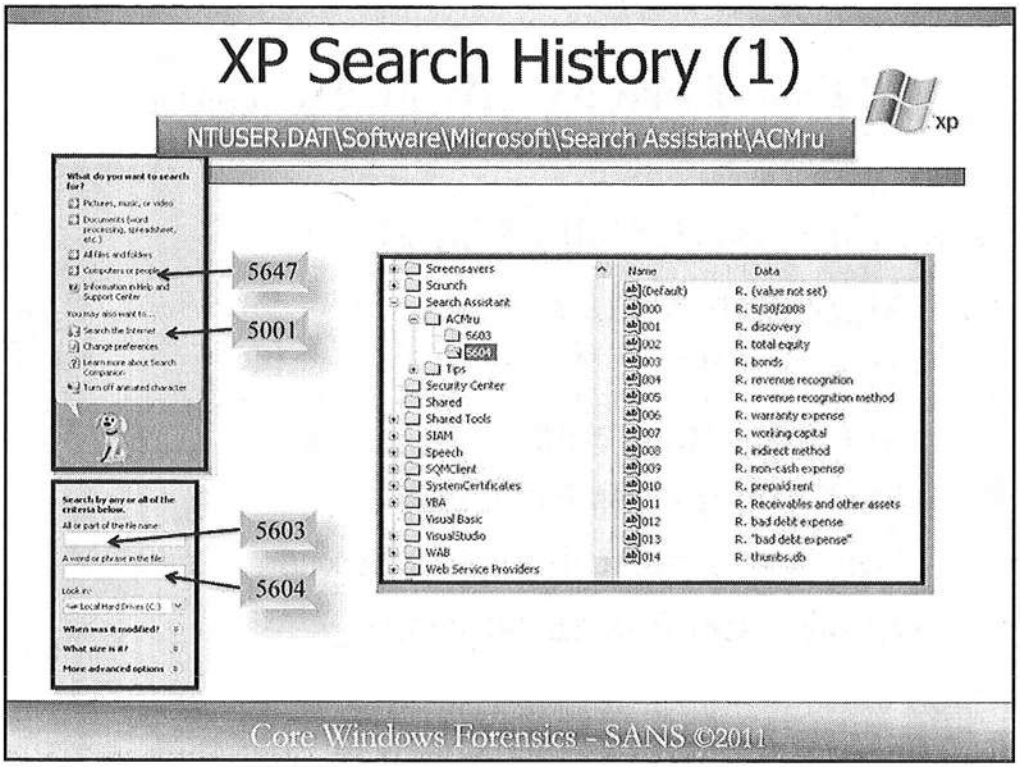
## User Activity Registry Keys

- Major User Activity Found
  - XP Search History
  - Typed URLS
  - Recent Documents
  - Last Commands Executed
  - Open-> Save/Run Dialog Boxes
  - Application Execution History

Core Windows Forensics - SANS ©2011

This page intentionally left blank.

[redacted]




You can search for multiple things through the search assistant on a Windows XP machine. The search assistant will remember a user's search terms for filenames, computers, or words that are inside a file. This is an example of where you can find the "Search History" on the Windows system.

**NTUSER.DAT\Software\Microsoft\Search Assistant\ACMru**

Maintains a list of items searched for via Start->Search; the subkeys (5001, 5603, 5604, etc.) correspond to the textfields where the user enters search. There are four keys in this:

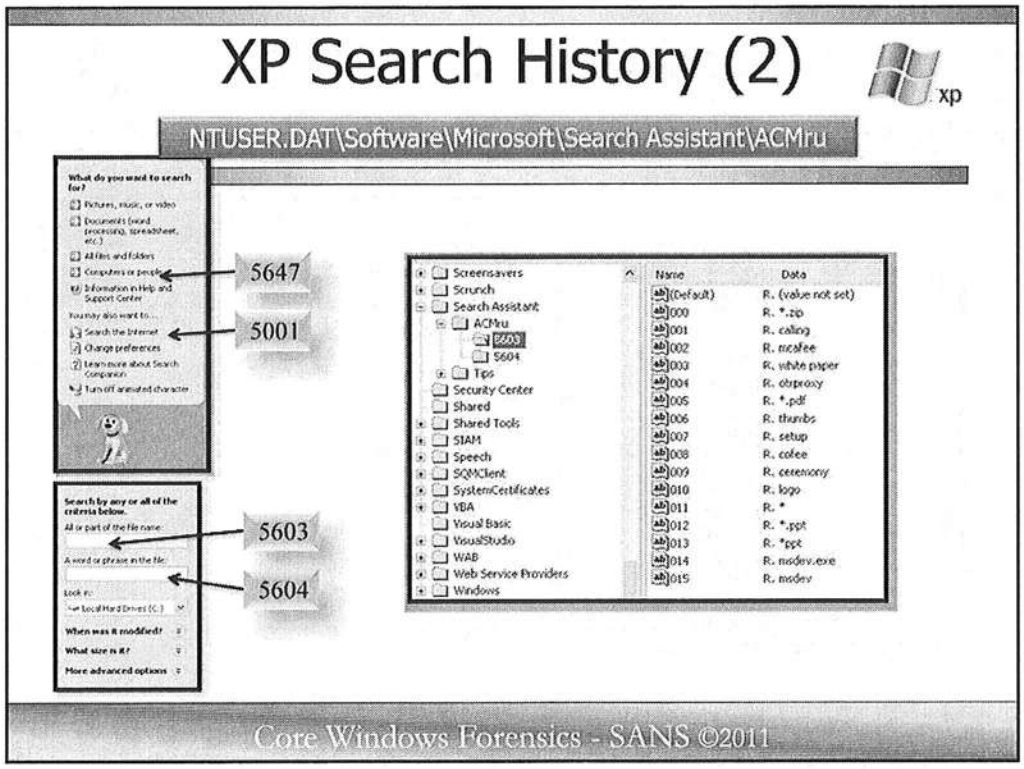
Search the Internet **Software\Microsoft\Search Assistant\ACMru\5001**

All or part of a document name **Software\Microsoft\Search Assistant\ACMru\5603**

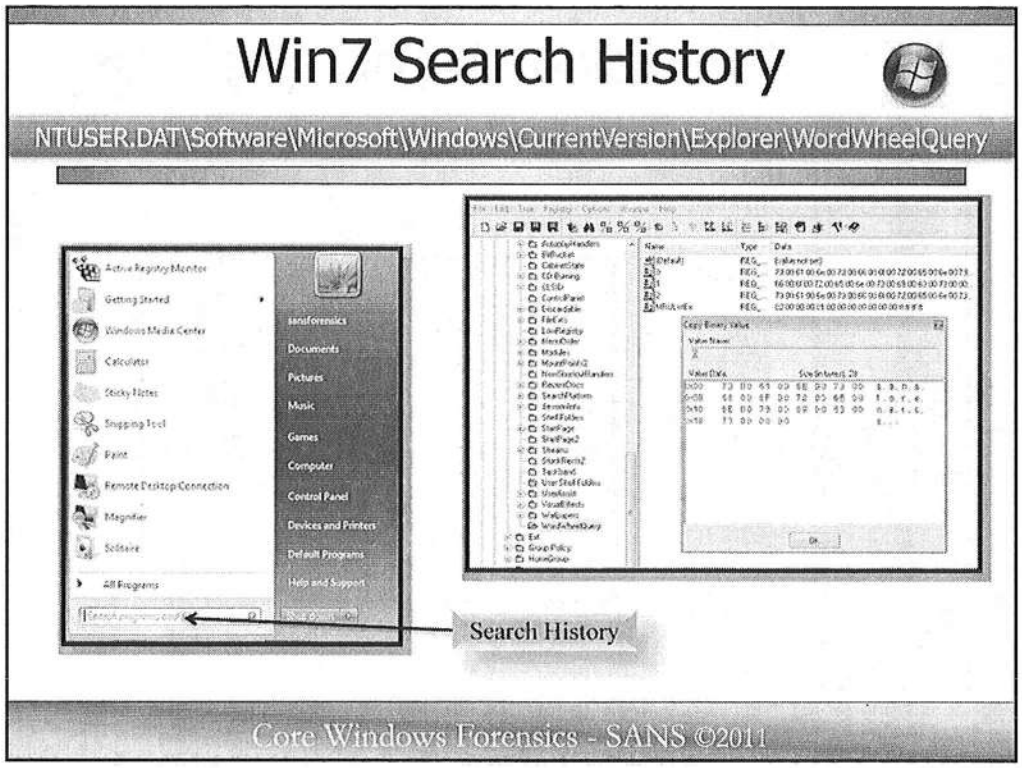
A word or phrase in a file **Software\Microsoft\Search Assistant\ACMru\5604**

Printers, Computers and People **Software\Microsoft\Search Assistant\ACMru\5647**

This key only exists on Windows XP through Windows 2003 operating systems. It does not exist on Windows Vista and Windows 7.



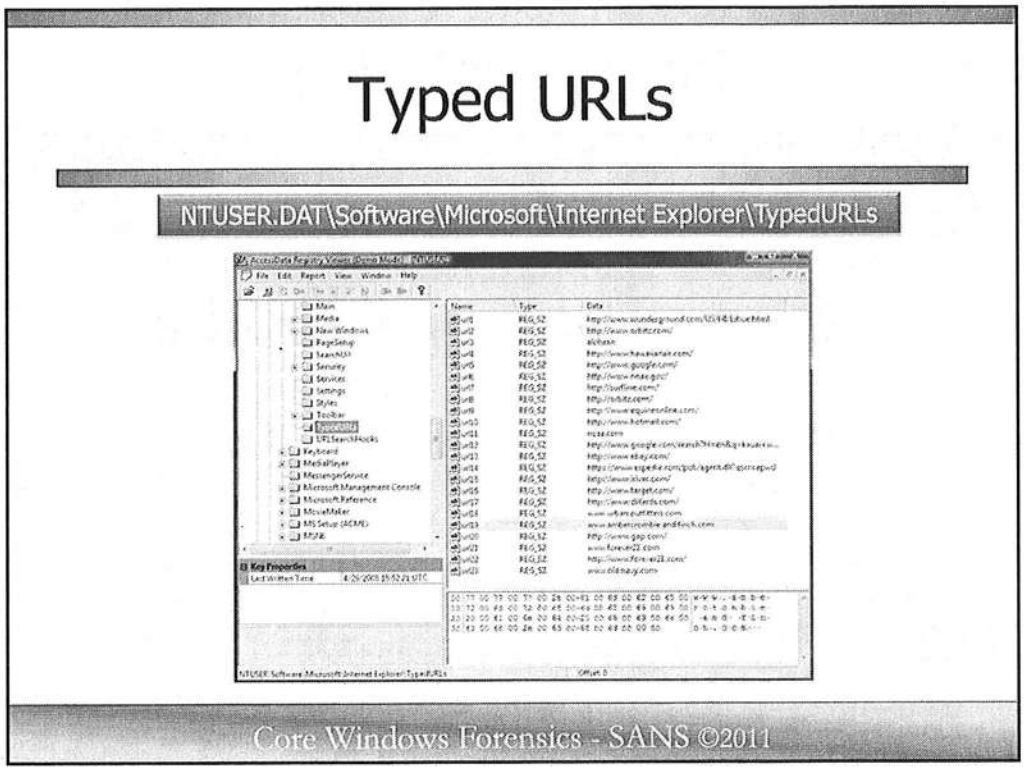

This is an example of an individual searching for parts of filenames. Notice how the wildcard values are still present in this key. The searches are in numeric order with the most recent being last in the list.




While Vista does not have any values recorded in the registry, while we were writing this course we discovered that Windows 7 does, in fact, have a registry key that does. The registry key called "WordWheelQuery" is new and only exists on Windows 7.

While the majority of Windows Vista configuration settings exist for both Vista and Windows 7, this is an example where there is clearly a difference. It will record the historical searches for programs and files on the machine. Any files or program names will be listed into the query itself.

Again, being able to know exactly when a specific user looked for an artifact is extremely useful to an investigation. With Windows 7 launching in Oct 2009, watch for additional keys being discovered.



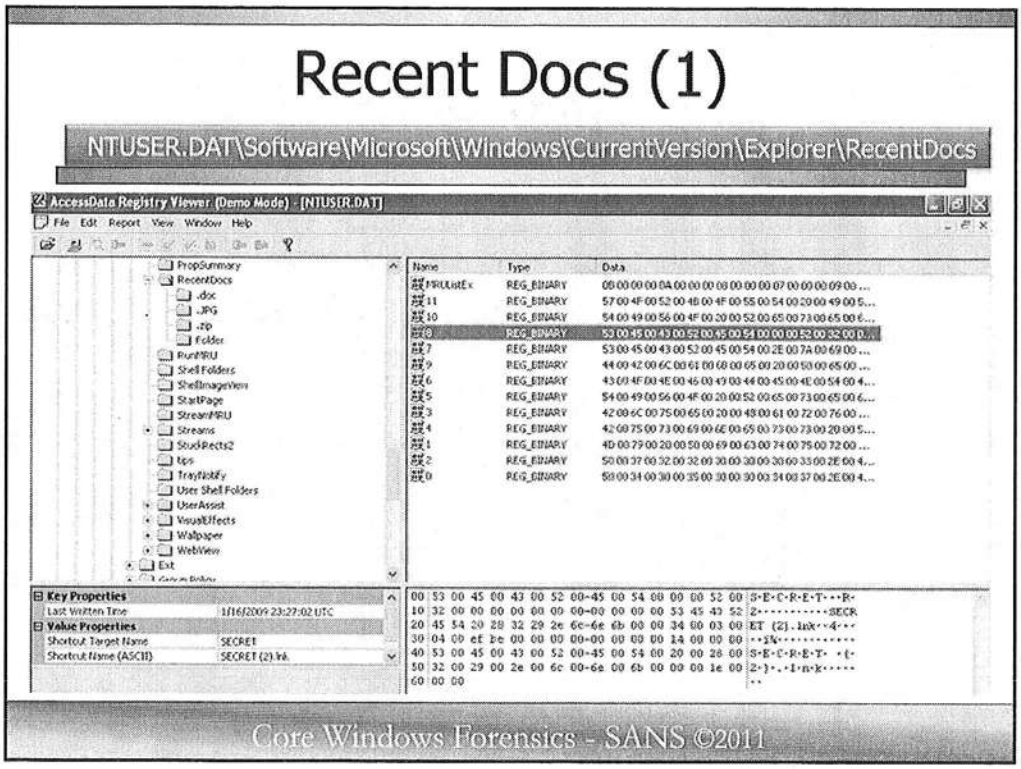

This is an example of where you can find the "Typed URLs" on the Windows system. (Windows 2000 -> Windows 7).

**`NTUSER.DAT\Software\Microsoft\Internet Explorer\TypedURLs`**

This is probably my favorite registry key. In many cases, people who search the Internet or go to a site claim that the site "popped up" (i.e., that the site just "magically appeared" in their browser). Now, this key is added and edited when a user types the website directly into the URL bar, or if they do a copy and a paste into the URL bar for the specific website.

This key will not be updated by clicking on the link. This is means the person has cognitively put the website address into the URL bar.

Inside this key will list every Typed website that a user typed in from most recent to furthest back in history.




The most recently used documents on a Windows XP, Vista, or Win7 machine can easily be examined by looking at the RecentDocs key. This key and the subkeys beneath it are extremely valuable when looking for activity surrounding a specific file.

When you examine the raw values of this key, the values are hard to read. However, Access Data's Registry Viewer can parse the contents of these keys so they can be easily viewed.

The values when displayed, via a viewer, will be displayed in MRUlist order. The MRU list shows the most recent file opened first. The file that was opened the furthest back in time will be listed last.

**NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs**

# Recent Docs (2)

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs

```
----------------------------------------
RecentDocs - recentdocs
Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
Lastwrite Time Fri Jan 16 23:27:02 2009 (UTC)
  11 = WORKOUT IPO (F:)
  10 = TIVO Research - CONFIDENTIAL - BACKUP2.doc
  8 = SECRET
  7 = SECRET.zip
  9 = DBlake Personal (E:)
  6 = CONFIDENTIAL_SPREADSHEETS.zip
  5 = TIVO Research - CONFIDENTIAL.doc
  3 = Blue Harvest Business Plan v1.doc
  4 = Business Plans
  1 = My Pictures
  2 = P7220003.JPG
  0 = P4050047.JPG
  4294967295 =
```

```
Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.docx
Lastwrite Time Sun May  3 05:04:01 2009 (UTC)
MRUListEx = 5,7,2,9,0,8,4,1,6,3,4294967295
  5 = MemoryzeStormWorm.docx
  7 = Forensic Handout 1.5.docx
  2 = Your mission...docx
  9 = Work For Hire Contract revised 11 19 07-1.docx
  0 = Webcast Request.docx
  8 = TOP 7 THINGS.docx
  4 = SIFTKIT_INFO.docx
  1 = SERVICE BASED DVR INDUSTRY FIVE FORCES MODEL.docx
  6 = SANS TOP 7 NEW IR FORENSIC TRENDS IN 2008.docx
  3 = SANS SEC 508 Course Descriptions.docx
  4294967295 =
```

Core Windows Forensics - SANS ©2011

This is the example of the same registry information from RecentDocs but viewed via the RegRipper output file which is much easier to interpret for a variety of reasons. 1. The MRUlist is used to create the actual order in which the files are opened. 2. You can see all the files in a single list instead of having to manually parse through each individual entry.

As I am sure you are beginning to realize, tools such as RegRipper will make life much easier to parse data such as this.

# Microsoft Office RecentDocs

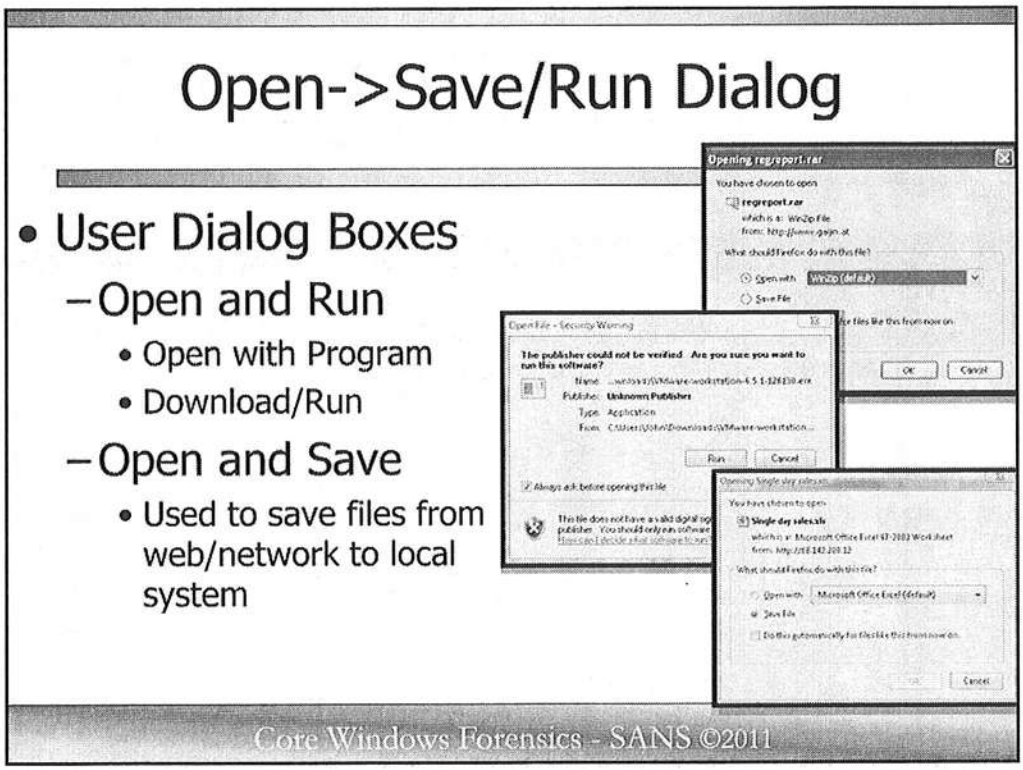
- Microsoft Office Stores "RecentDocs"
  - Each Version of office product under

    NTUSER.DAT\Software\Microsoft\Office\VERSION
  - The version can be varied.
    - 12.0 = Office 2007
    - 11.0 = Office 2003
    - 10.0 = Office XP

| Office Program | Location in User's NTUSER.DAT Hive |
|---|---|
| Word 2007 | Software\Microsoft\Office\12.0\Word\FileMRU |
| Excel 2007 | Software\Microsoft\Office\12.0\Excel\FileMRU |
| Powerpoint 2007 | Software\Microsoft\Office\12.0\Powerpoint\FileMRU |

Core Windows Forensics - SANS ©2011

Not only will the operating system keep track of the user's last documents saved, the Microsoft Office product will as well. Each version of Office will keep a list of all the files opened in a specific program. The location is always in a key called FileMRU. Inside this key will list many of the recent the documents, spreadsheets, and PowerPoint presentations that the user has opened. This list could be quite extensive and go back much further in time than the system RecentDocs key/values,>due to having more space and not needing to overwrite the data as fast.




When you are surfing the web, in many cases you might come upon a situation where you will need to download and save a file to your local system. In other cases, you might be downloading and executing the program directly, or viewing the file automatically via a viewer.

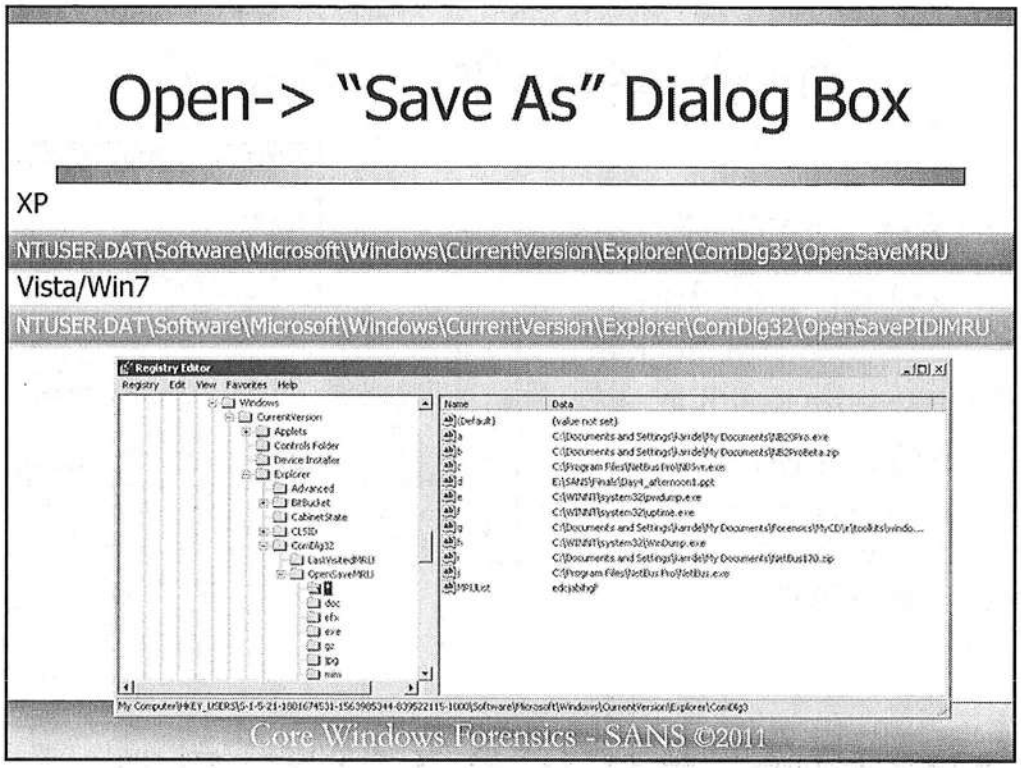
There are two specific situations tracked within Microsoft Windows.

The first is when you Open and Run a file via a specific program on the system. Either the program itself will execute or an assigned program will execute in order to read the data from the file. Both instances will create the entry for this key which show which program was executed to read the data from the file. This will create specific registry artifacts in the Windows system.

Second, when you Open and Save a file to the local disk, this operation is also logged and stored within the registry.

This information is not easily deleted as I have found out. It is persistent and was not easily removed until I utilized a privacy cleaner.

This activity would be very useful in cases where you think the user opened a file that contained a piece of malware. Alternatively, the user could have downloaded and executed a program that they thought was a safe program sent to them from a friend, but turned out to be spear phishing. This area ends up being one of the first places I check to see if the files listed here could possibly contain code that subverts the operating system.




This is an example of where you can find the "Last Files Saved" on the Windows system.

Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU*

If you take a look at the output above you can see the last save locations for many files via Windows. Also notice that one of the files, a PowerPoint file was saved to a drive at E:. This could be a USB key. We will talk about the registry and USB keys over the next few days, but this could have been a critical file or intellectual property and you now know you need to track what was drive letter "E".

[redacted]

# Open->Run Dialog Box

XP

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU

Vista/Win7

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedPIDlMRU

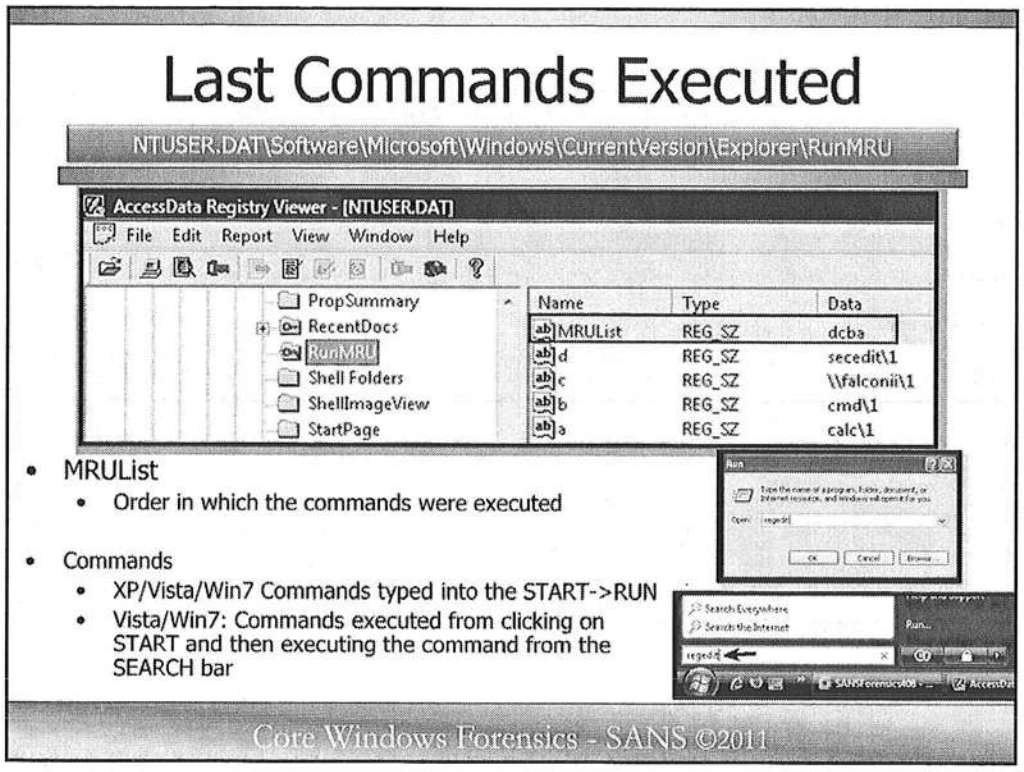
```
comdlg32 v.20080324
ComDlg32\LastvisitedMRU
**All values printed in MRUList order.
Software\Microsoft\Windows\Currentversion\Explorer\ComDlg32\LastvisitedMRU
LastWrite Time Wed Oct 19 16:50:31 2005 (UTC)
        MRUList = aehgfdcb
        a -> C:\Documents and Settings\Tech\Desktop\iexplore.exe
        e -> C:\Documents and Settings\Tech\My Documents\NOTEPAD.EXE
        h -> C:\Documents and Settings\Tech\My Documents\My Pictures\kirs pix of europe cd\ZoomBrowser.exe
        g -> C:\Documents and Settings\Tech\My Documents\My Pictures\Launcher.exe
        f -> C:\Documents and Settings\kirsten\My Documents\My Pictures\Kentucky Derby etc\Stitch.exe
        d -> C:\Documents and Settings\Tech\My Documents\WORDPAD.EXE
        c -> C:\Documents and Settings\Tech\My Documents\dellix.exe
        b -> C:\Documents and Settings\Tech\Desktop\winzip32.exe
```

Core Windows Forensics - SANS ©2011

As noted here, you can see the programs that were executed via the Open->Run dialog box, which executes the appropriate program if a file, or the program itself if an executable.

Either the program itself will execute or an assigned program will execute in order to read the data from the file. Both instances will create the entry for this key, which shows which program was executed to read the data from the file. This will create specific registry artifacts in the Windows system.

This information is also viewed via the output from RegRipper, showing that the MRUlist of the file is followed and the files are listed in order with the most recent file opened at the top of the list. The timestamp of the key will match the last file that was executed in this list.




This is an example of where you can find the "Last Command Executed" on the Windows system.

\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU

Whenever someone does a Start -> Run command, it will log the entry for the command that they executed. This key will list the commands executed from the Start -> Run from the Windows bar on their desktop.

The order in which the commands are executed are listed in the RunMRU list value. The letters represent the order in which the commands were executed.

## Application Artifacts in the Registry

- Do programs leave traces in the registry?
- If our suspect executed a program, can we tell when it was last executed?
- Can we tell how many times a program was executed on the system?
- The USERASSIST keys in the suspect's NTUSER.dat registry hive will aid you here.



Applications also leave footprints specific to a user's activities in the NTUSER.DAT hive. The keys in general that we will examine are the USERASSIST keys which can show the last time of execution as well as the number of times a program was executed.

# When Was the Program Last Run? How Many Times?

- UserAssist Keys in NTUSER.DAT

NTUSER.DAT\Software\Microsoft\Windows\Currentversion\Explorer\UserAssist\{GUID}\Count

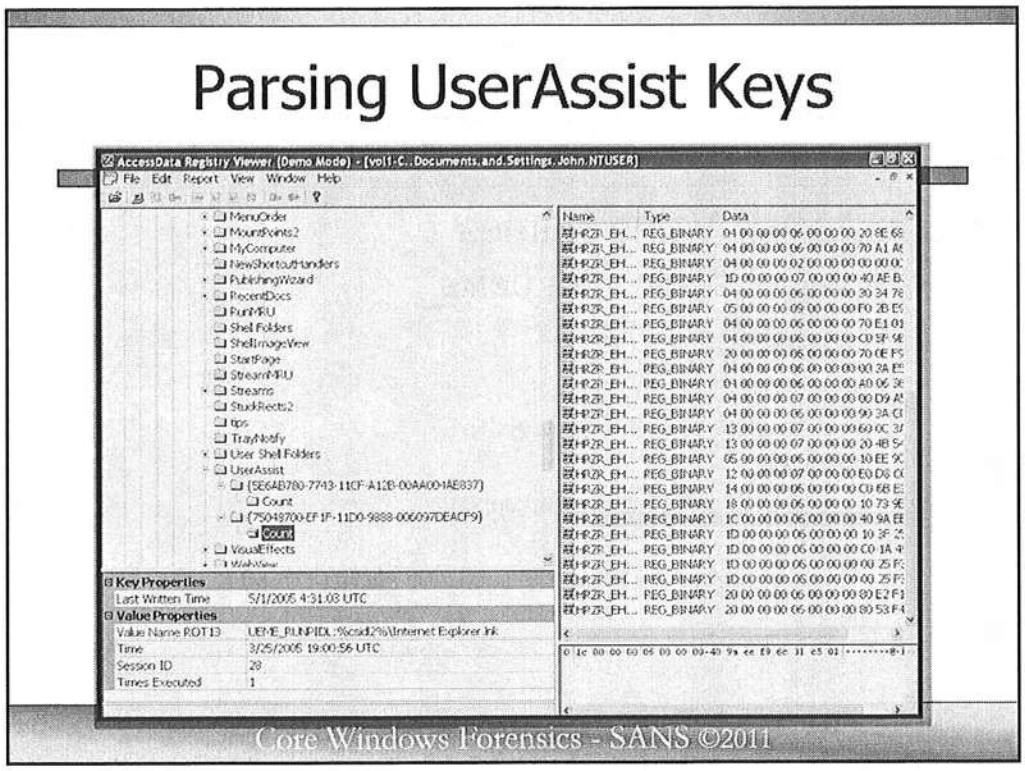
  - GUID
    - 5e6ab780… -> Internet Toolbar
    - 75048700… -> Active Desktop
- Logs
  - Last Run Time
  - Run Count (starts count at 6)
  - Name of Application
- Values are ROT-13 encoded

Core Windows Forensics - SANS ©2011

One of the more interesting things about a Windows system is the sheer amount of data that is collected. One of these is found in the user's registry. If you happen to be logged on as the user that is under investigation, you can run a script that will examine that user's registry file under Software\Microsoft\Windows\Currentversion\Explorer\UserAssist\{GUID}\Count. The GUID (Globally Unique Identifier) points to the type of application that is being utilized.

The great thing about the UserAssist key is that it will list all the applications launched from the Windows GUI interface for each user (it will not list command line applications). Futhermore, it will list the last run time of each application and the total amount of times that it was executed.

The values of these keys that you find are not readable<,> as they are ROT-13 encoded. This is a simple cipher and easily decrypted using AccessData's Registry Viewer. The important part of it is that you can see the last run time, run count, and the name of the application executed. The run count starts counting at 6<,> so if you want an accurate run count<,> then you would have to subtract 5 from it to see the number of times the application was really executed.




This is an example of using the Registry Viewer from AccessData to examine the UserAssist keys. Notice it parsed the ROT-13 encoding to pull out the name as well as the run count. It automatically subtracted five to get the actual run count from the registry value.

# UserAssist Name Values

- UserAssist Value Names
  - All values begin with **UEME_**
  - Followed by:
    - **RUNPATH**
      - Absolute path of an executable
    - **RUNCPL**
      - Launch of a control panel applet
    - **RUNPIDL**
      - A pointer to the actual file such as a shortcut or a LNK file



The value of the name has meaning as well in the User Assist keys. The key's name value will always be preceded by the UEME_ and then followed by either RUNPATH, RUNCPL, or RUNPIDL. Each one of these names carry a significance to how the file was executed.

The RUNPATH is the absolute path of an executable. A user most likely double clicked on the executable through the Windows Explorer interface.

The RUNCPL is the launching of a Control Panel applet. This is interesting if you see any user changing values to the security or the user settings on the system.

The RUNPIDL is a pointer to the actual file<,> such a shortcut from a LNK file.

## Review User Activity Registry Keys

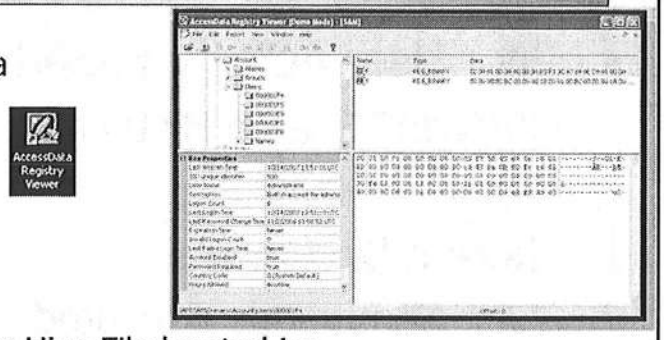
- Major User Activity Found
  - XP/Win7 Search History
  - Typed URLS
  - Recent Documents
  - Last Commands Executed
  - Open-> Save/Run Dialog Boxes
  - Application Execution History



This page intentionally left blank.

# HANDS-ON Registry Viewer
# Examine the NTUSER.DAT Hive

- Execute AccessData Registry Viewer



1. Open **NTUSER.DAT** Hive File located in:
   **E:\Documents and Settings\Donald Blake\NTUSER.DAT**
2. Fill out User Profile Guide in Appendix B

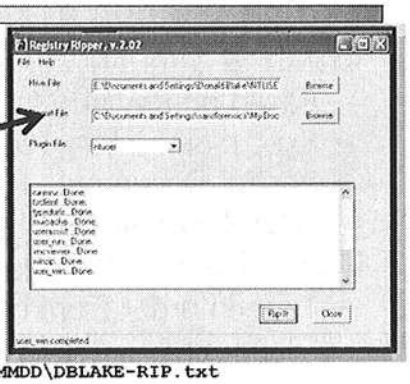
Core Windows Forensics - SANS ©2011

Execute AccessData Registry Viewer:

1. Open NTUSER.DAT Hive File located in:
   **E:\Documents and Settings\Donald Blake\NTUSER.DAT**
2. Fill out System Configuration Profile Guide in Appendix B

# HANDS-ON RegRipper
# Rip the NTUSER.DAT Hive

- Execute RegRipper from Desktop ICON




1. Select NTUSER.DAT Hive File located in:
   **E:\Documents and Settings\Donald Blake**
2. Select OUT file in case directory in:
   **C:\Users\sansforensics\My Documents\caseYYYYMMDD\DBLAKE-RIP.txt**
3. Choose NTUSER plugin file
4. Click "Rip It"
5. Open **DBLAKE-RIP.txt** and compare against Appendix B

Core Windows Forensics - SANS ©2011

Execute RegRipper from Desktop ICON:

1. Select NTUSER.DAT Hive File located in:
   **E:\Documents and Settings\Donald Blake**
2. Select OUT file in case directory in:
   **C:\Users\sansforensics\My Documents\caseYYYYMMDD\DBLAKE-RIP.txt**
3. Choose NTUSER plugin file
4. Click "Rip It"
5. Open **DBLAKE-RIP.txt** and compare against Appendix B

# Hands-On: Create Timeline

- Create a Timeline of Known Events in EST/EDT
  - Based on Registry Last Write Times
  - Execution History (UserAssist)
  - Recent Documents History (RecentDocs)
  - Open Save MRU
- Use Table on Next Page:
  - Time in order from Most Recent to Earliest
  - Filename
  - Event (What Happened)
  - Factual Source of the Event
    - UserAssist
    - RecentDocs

Core Windows Forensics - SANS ©2011

This page intentionally left blank.

| Time | Filename | What Happened? | Source? |
|---|---|---|---|
| Jan 16 6:27:02 PM | WORKOUT IPO (F:) | | |
| Jan 16 6:27:02 PM | | | |
| Jan 16 6:27:02 PM | | | |
| | | | |
| Jan 16 6:25:30 PM | | | |
| Jan 16 6:25:28 PM | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| Jan 14 4:04:21 PM | P7220003.JPG | | |

Core Windows Forensics - SANS ©2011

This page intentionally left blank.

| ANSWERS: Create Timeline | | | |
|---|---|---|---|
| Time | Filename | What Happened? | Source? |
| Jan 16 6:27:02 PM | WORKOUT IPO (F:) | Folder Opened | Recent Docs |
| Jan 16 6:27:02 PM | TIVO Research - CONFIDENTIAL - BACKUP2.doc | Document Opened | Recent Docs |
| Jan 16 6:27:02 PM | Wordpad.exe | Wordpad Executed | UserAssist |
| | SECRET | | Recent Docs |
| Jan 16 6:25:30 PM | SECRET.zip | ZipFile Opened | Recent Docs |
| Jan 16 6:25:28 PM | Winzip.exe | Winzip Execute | UserAssist |
| | DBlake Personal (E:) | Folder Opened | Recent Docs |
| | CONFIDENTIAL_SPREADSHEETS.zip | Zipfile Opened | Recent Docs |
| | TIVO Research - CONFIDENTIAL.doc | Document Opened | Recent Docs |
| | Blue Harvest Business Plan v1.doc | Document Opened | Recent Docs |
| | Business Plans | Folder Opened | Recent Docs |
| | My Pictures | Folder Opened | Recent Docs |
| Jan 14 4:04:21 PM | P7220003.JPG | Picture Opened | Recent Docs |

How do we know the order of the Files that do not have a specific time?

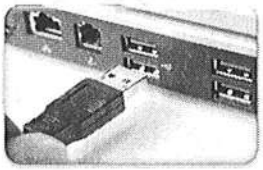
1. The Recent Docs has a MRUList which keeps track of the exact order the values were opened.
2. The Recent Docs *.jpg had a last write time of Jan 14 4:04 PM.
3. The Recent Docs *.zip had a last write time of Jan 16 6:25 PM.
4. The files were opened in order between those two times.

RecentDocs – recentdocs (FROM RegRipper)
*****All values printed in MRUList\MRUListEx order.***
Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
LastWrite Time Fri Jan 16 23:27:02 2009 (UTC)
11 = WORKOUT IPO (F:)
10 = TIVO Research - CONFIDENTIAL - BACKUP2.doc
8 = SECRET
**7 = SECRET.zip Jan 16 6:25**
9 = DBlake Personal (E:)
6 = CONFIDENTIAL_SPREADSHEETS.zip
5 = TIVO Research - CONFIDENTIAL.doc
3 = Blue Harvest Business Plan v1.doc
4 = Business Plans
1 = My Pictures
**2 = P7220003.JPG Jan 14 4:04**
0 = P4050047.JPG

SANS COMPUTERFORENSICS
and e-Discovery with Rob Lee

# Analyzing USB Devices

## SYSTEM, SOFTWARE, NTUSER.DAT

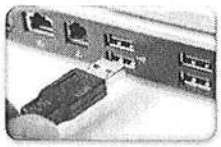




This page intentionally left blank.

# Purpose of USB Device Forensics

- USB Information
  - –Vendor/Make/Version
  - –Unique Serial Number



- User Information and Activity With USB Device
  - –Last Drive Letter
  - –Volume Name
  - –The username that used the USB Device
  - –Time of First Use of USB Device
  - –Time of First Use of USB Device Since Last Reboot
  - –Time of Last Use of USB Device

Core Windows Forensics - SANS ©2011

The purpose of USB Key forensic analysis is to determine if a specific USB Device (Key, Printer, Camera, iPod) was plugged into a system.

An investigator should be able to be familiar with pulling data from both XP, Vista, and Windows 7 machines. While similar, some of the steps vary slightly and it is important to know exactly how to work with both considering the large amount of cases now involving peripheral devices.

There is quite a bit of information you can map back to a device. Each device will contain a specific serial number even if it is reformatted. That information can be utilized to see if a device was plugged into a specific machine or to find a specific device that was utilized in a crime.

## Where is USB Information Stored?

- Windows Registry Hives
  - System.dat
  - Software.dat (Vista/Win7 only)
  - Ntuser.dat
- Setupapi.log (Plug and Play Log)
- Event Logs



This page intentionally left blank.

# USBSTOR

**Purpose**

- Track USB devices plugged into a machine

**Location**

- **SYSTEM*CurrentControlSet*\Enum\USBSTOR**

**Why Key Is Useful?**

- Identify Vendor, Product, and Version of a USB device plugged into a machine
- Identify a unique USB device plugged into the machine
- Determine the time a device was plugged into the machine

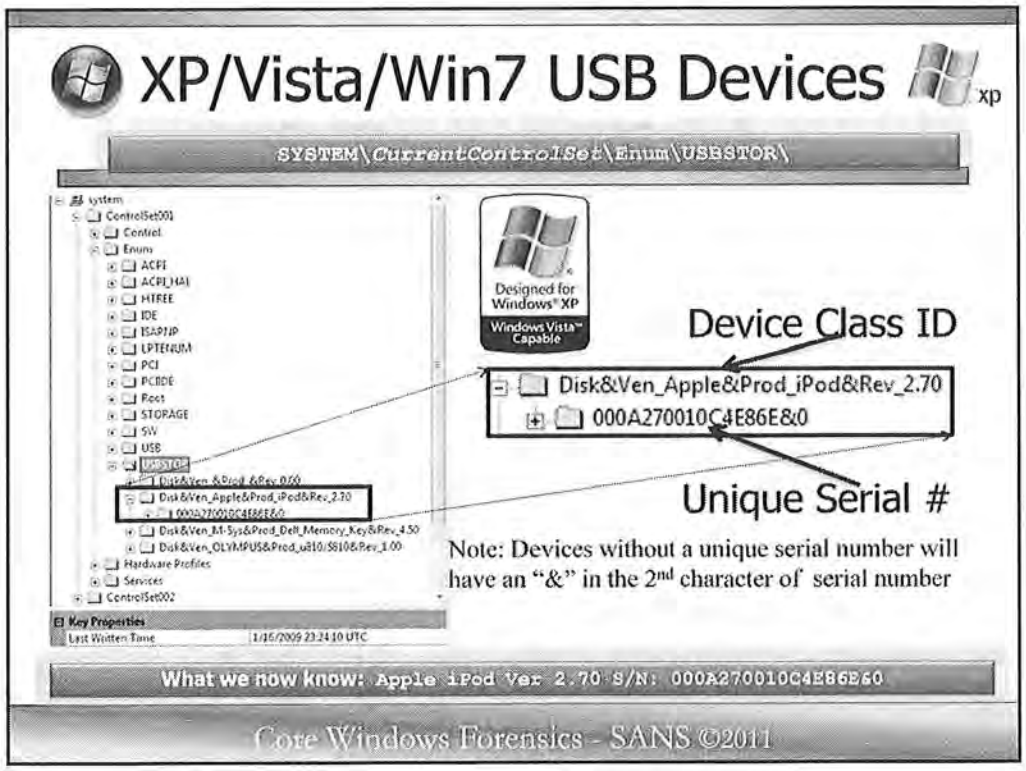


USB Key Analysis is one of the most talked about areas in computer forensics. The forensic analysis is useful because you can track USB Devices that have been plugged into a machine. With USB Key analysis you can: 1. Identify Vendor, Product, and Version of a USB device plugged into a machine. 2. Identify a unique USB device plugged into the machine 3. Determine the time a device was plugged into the machine.

The USB Key information will be stored at the **SYSTEM*CurrentControlSet*\Enum\USBSTOR** key.

From the USB FAQ: Intermediate (www.microsoft.com/whdc/connect/usb/usbfaq_intermed.mspx):

"If the device has a serial number, Microsoft requires that the serial number uniquely identify each instance of the same device. For example, if two device descriptors have identical values for the *idVendor*, *idProduct*, and *bcdDevice* fields, the *iSerialNumber* field will distinguish one from the other."




In order to qualify for the Windows Logo Program "Designed for Windows", the USB device must contain a unique serial number. Essentially, while the vendor ID, product ID, and the revision number might be the same for similar devices, each unique device will contain a unique serial number to differentiate it from the other devices.

(REF: http://www.microsoft.com/whdc/archive/usbfaq.mspx#EPCAC)

However, not all devices conform to this standard and thus do not include a unique serial number.

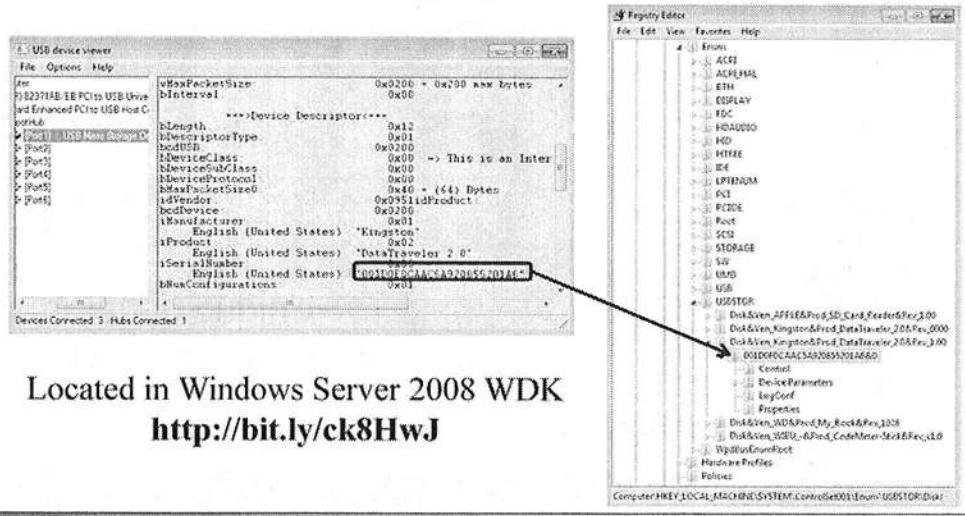
Devices that do not have a unique serial number, will have an "&" in the second character of the serial number. For example if a serial number was listed with 4&762219b0&0 that would mean that this device does not have a unique serial number and possibly all similar devices would have the same unique ID. This device does not conform to the Microsoft standards.

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270010C4E86E&0**

# Linking to a Physical Device: Microsoft UVCView

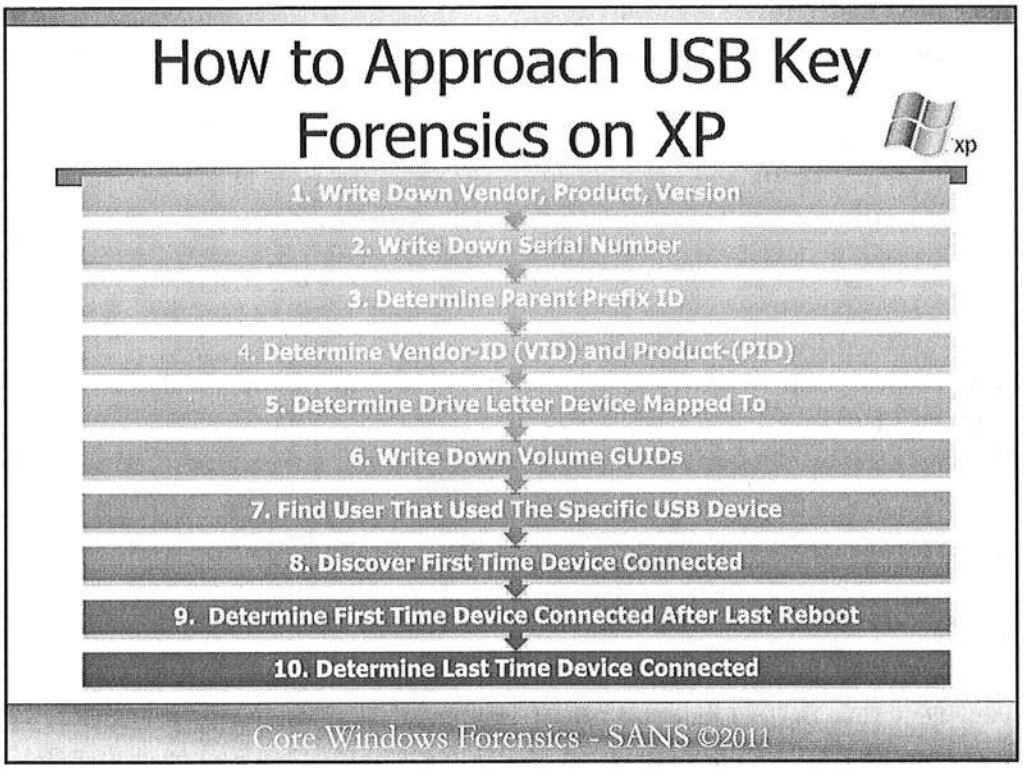



Located in Windows Server 2008 WDK

**http://bit.ly/ck8HwJ**



In many cases, you might find a USB Device that you need to pull the serial number off of first before finding the system that it was connected to. In this situation<,> you can download a tool called UVCView that will detail the unique information that is associated with the currently plugged in USB device.




**<u>XP USB KEY/Thumbdrive</u>**

**1. Write Down Vendor, Product, Version**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Vendor =

Product =

Version =

**2. Write Down Serial Numbers**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Serial Number =

**3. Determine Parent Prefix ID**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Parent Prefix ID=

**4. Determine Vendor-ID (VID) and Product-(PID)**

**SYSTEM\CurrentControlSet\Enum\USB ->** Perform search for S/N

**VID_XXXX =**

**PID_YYYY =**

**5. Determine Drive Letter Device Mapped To**

**SYSTEM\MountedDevices->** Perform search for ParentIdPrefix in the Drive Letter

Drive =

**6. Write Down Volume GUIDs**

**SYSTEM\MountedDevices->** Perform Search for ParentIdPrefix in the GUIDs

{GUID} =

**7. Find User That Used The Specific USB Device**

**NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2->**

Search for Device GUID

User =

**8. Discover First Time Device Connected**

**C:\Windows\setupapi.log** -> Perform search for Serial Number

Time/Timezone =

**9. Determine First Time Device Connected After Last Reboot**

**SYSTEM\CurrentControlSet\Control\DeviceClasses\{53f56307-b6bf-11d0-94f2-00a0c91efb8b}->** Perform search for S/N

<u>or</u>

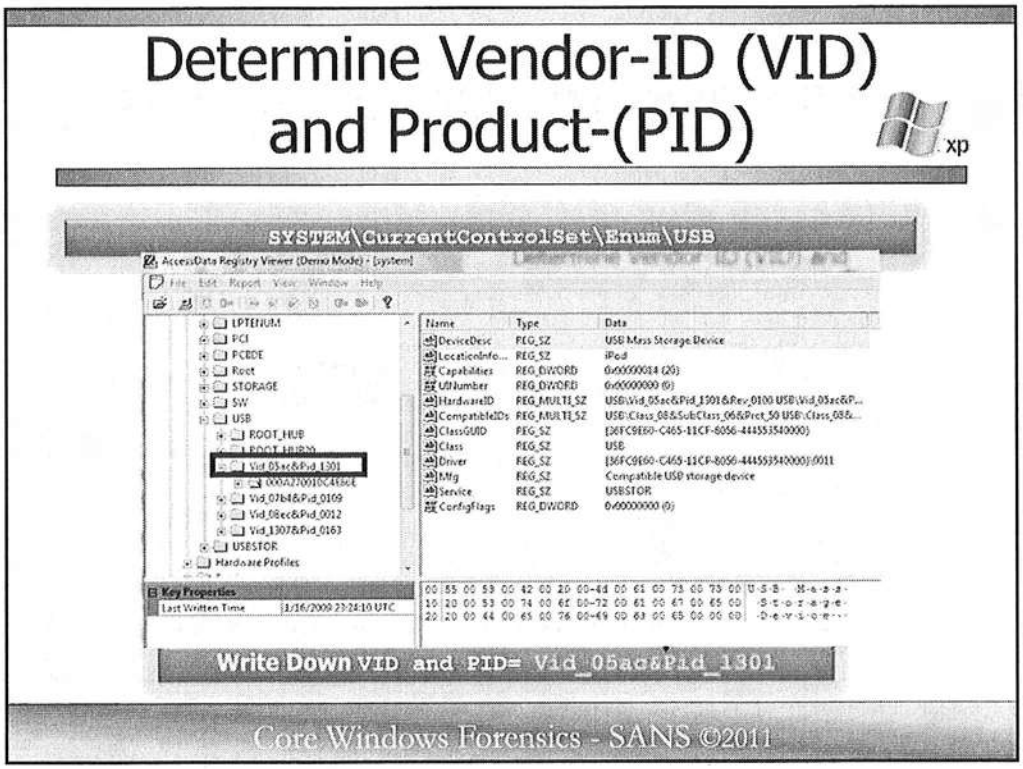
**SYSTEM\CurrentControlSet\Enum\USB\ VID_XXXX&PID_YYYY** -> Perform search for Serial Number (Last Written Time of Serial Number Key)

Time/Timezone =

**10. Determine Last Time Device Connected**

**NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID}** -> Perform search for Device {GUID}

Time/Timezone =




Find the VID and PID for later use to determine the last time the device was connected to the machine.

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270010C4E86E&0**

VID and PID= **Vid_05ac&Pid_1301**

# Discover the Drive Letter

## Purpose

- Discover the drive letter of the USB Device when it was plugged in the machine

## Locations

- Find Parent Prefix ID
  - `SYSTEM\CurrentControlSet\Enum\USBSTOR`
- Using ParentIdPrefix Discover Last Mount Point
  - `SYSTEM\MountedDevices`

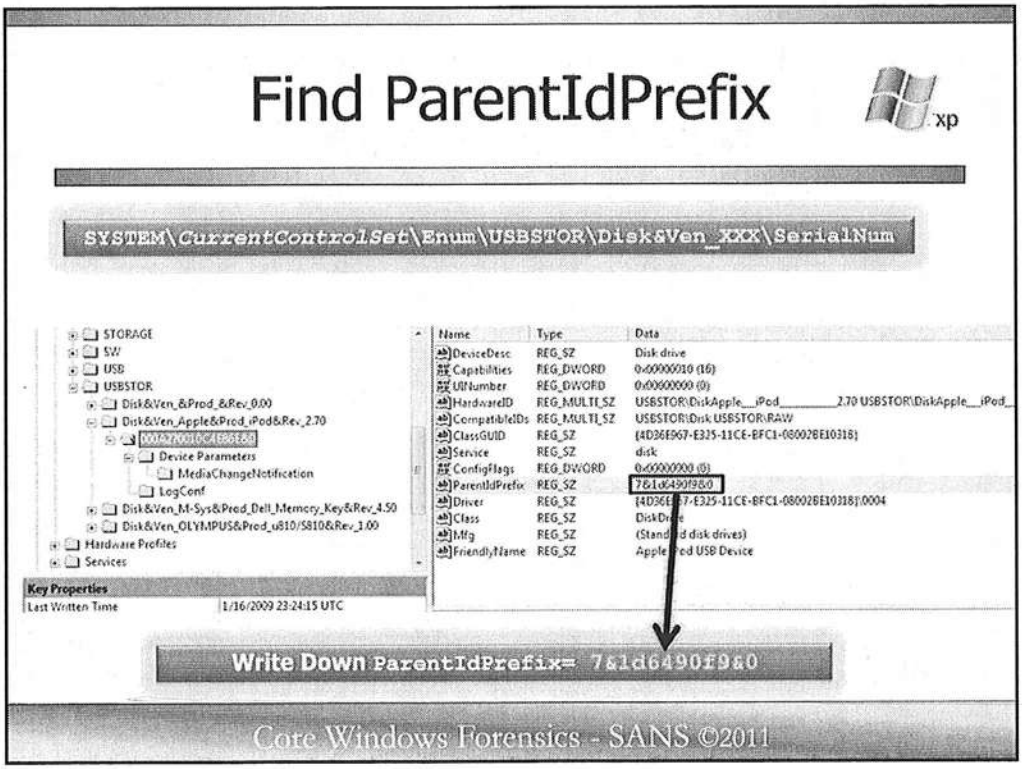
## Why Key Is Useful?

- Identify the USB device that was last mapped to a specific drive letter on a Windows XP Machine

Core Windows Forensics - SANS ©2011

Discovering the drive letter of a USB device when it was plugged into a machine is a two step process. First you need to have written down the ParentIdPrefix from the USBSTOR earlier. Then you open up the SYSTEM hive and examine the MountedDevices key.

The last device that was mounted at a drive letter will have its parent prefix ID listed in a value. Search for the drive with that parent prefix ID and then you will know the drive letter of that device. If another device was plugged in and assumed that drive letter, then you will not know the drive letter for a specific device. This only works for the last device that was mapped to a specific drive letter. This is not historical like you will find on Windows Vista or Windows 7 with the Windows Portable Devices key.



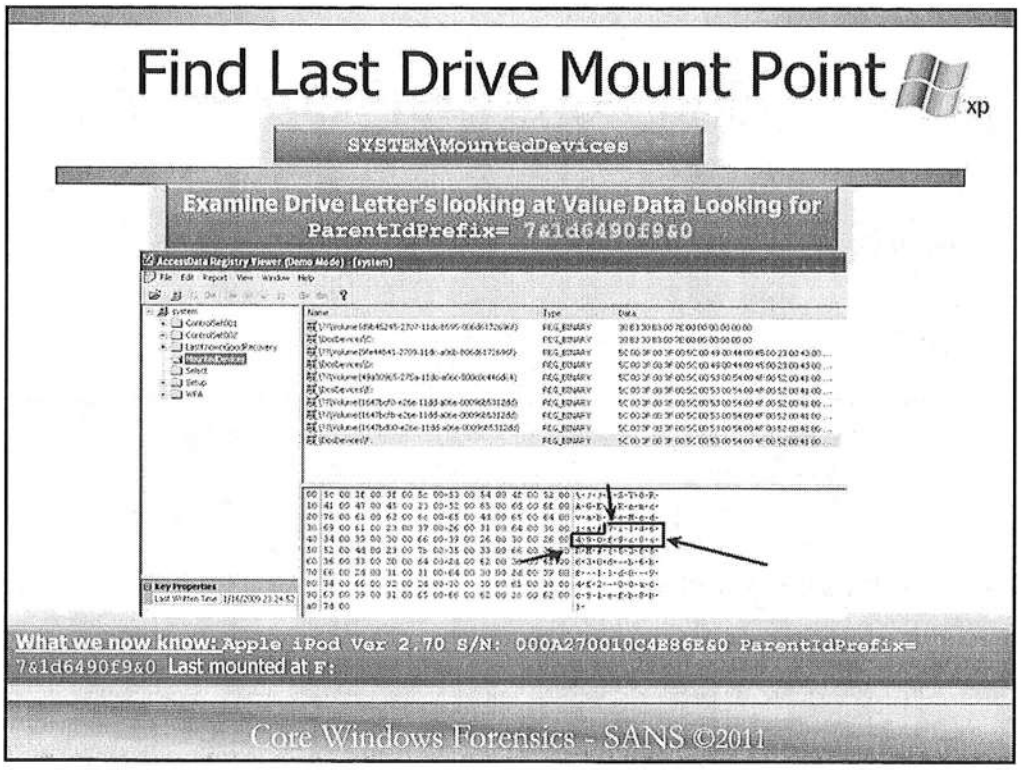

Finding the ParentIdPrefix is important for being able to correlate some additional data later on for forensic purposes. When you click on the serial number, you will see the parentID Prefix listed. In this case it is 7&1d6490f9&0. This ID will be useful in being able to track potential data found on this key in both .lnk files as well as correlating the information to the mounted devices key which we will talk about shortly.

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270010C4E86E&0**

ParentIdPrefix= **7&1d6490f9&0**



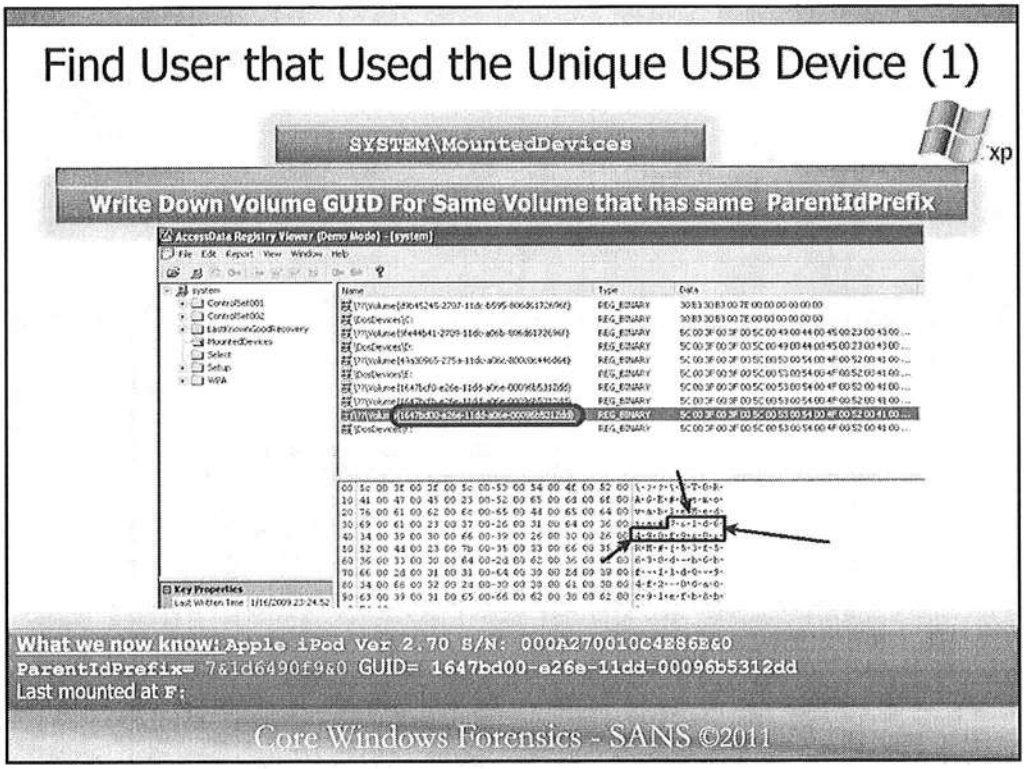

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270010C4E86E&0**

ParentIdPrefix= **7&1d6490f9&0**

Last mounted at **F:**




Globally Unique Identifier (GUID) - A 16-byte number generated by Microsoft programs that uniquely identifies a network, user, computer or document. This GUID will be used next to identify the user that plugged in the device. The number will be referenced in the user's personal mountpoint's key in the NTUSER.DAT Hive.

**What we now know:**

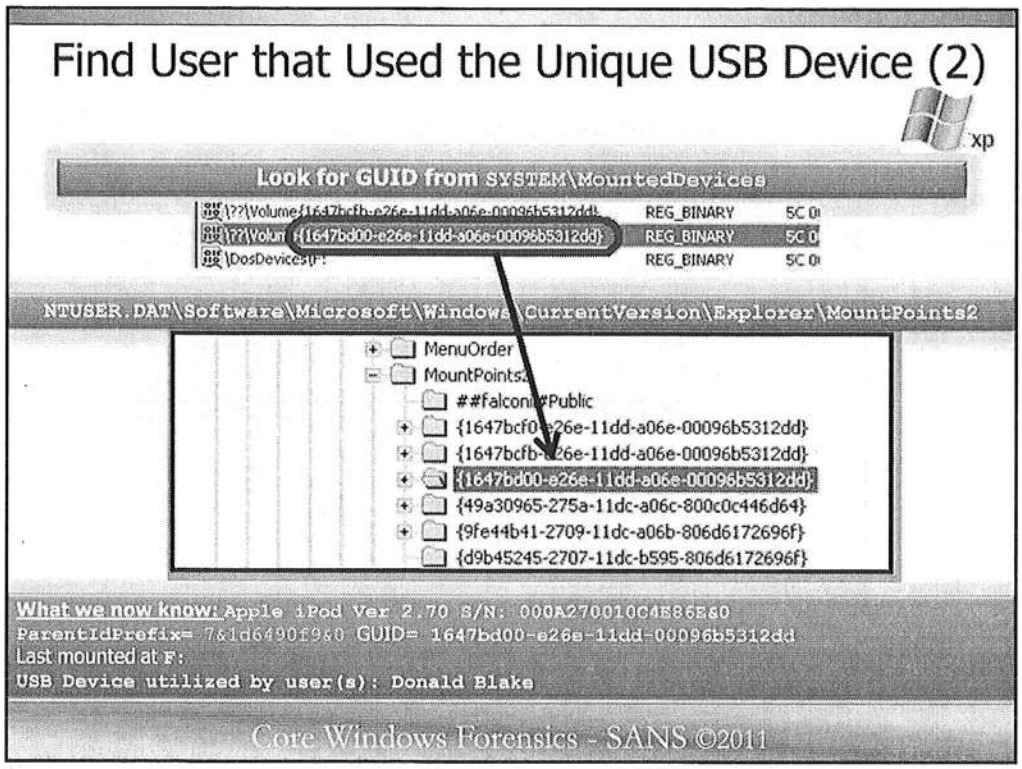
Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270010C4E86E&0**

ParentIdPrefix= **7&1d6490f9&0**

Last mounted at **F:**

GUID= **1647bd00-e26e-11dd-00096b5312dd**




It is one of the elements of information that can be passed when you connect to an Internet site, and it may be stored in cookies.

This GUID will be used next to identify the user that plugged in the device. The last write time of this key also corresponds to the last time the device was plugged into the machine by that user. The number will be referenced in the user's personal mountpoint's key in the NTUSER.DAT Hive.

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

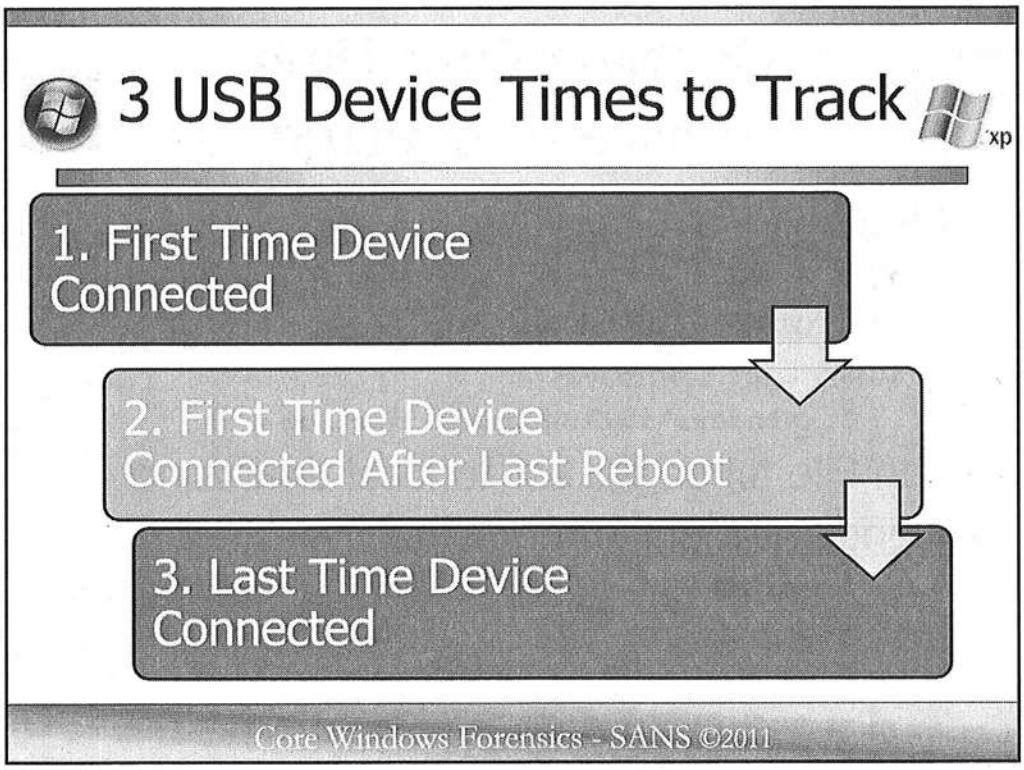
S/N: **000A270010C4E86E&0**

ParentIdPrefix= **7&1d6490f9&0**

GUID= **1647bd00-e26e-11dd-00096b5312dd**

Last mounted at **F:**

USB Device utilized by user(s): **Donald Blake**




There are 3 USB times we are able to track when examining USB Keys. One of the most difficult things to examine personally over the past 6 months is that each operating system changes its practice of where these dates might be stored. Remember, it is not Microsoft's intention that these keys would be used in this intention. A simple service pack could change the entire precision of using this data effectively in a forensic case. However, given that, the following information is known as the most up to date information regarding temporal data regarding USB keys on a Windows operating system.

There are three times we are able to track:

1. First Time Device Connected
2. First Time Device Connected After Last Reboot
3. Last Time Device Connected

The benefits of these times could be quite useful to an investigator for a variety of situations. Some of these times could be obtained when viewing previously discussed keys, but in order to present it to you in a conceptual fashion, we will go through them in order one by one.

# Discover First Time Device Connected

- Plug and Play Log Files
  - XP
    - `C:\Windows\setupapi.log`
  - Vista and Windows 7
    - `C:\Windows\inf\setupapi.dev.log`
- Log File times are set to local time zone
- Mandiant Highlighter Helps
  - Search for Device Serial Number and look for first entry



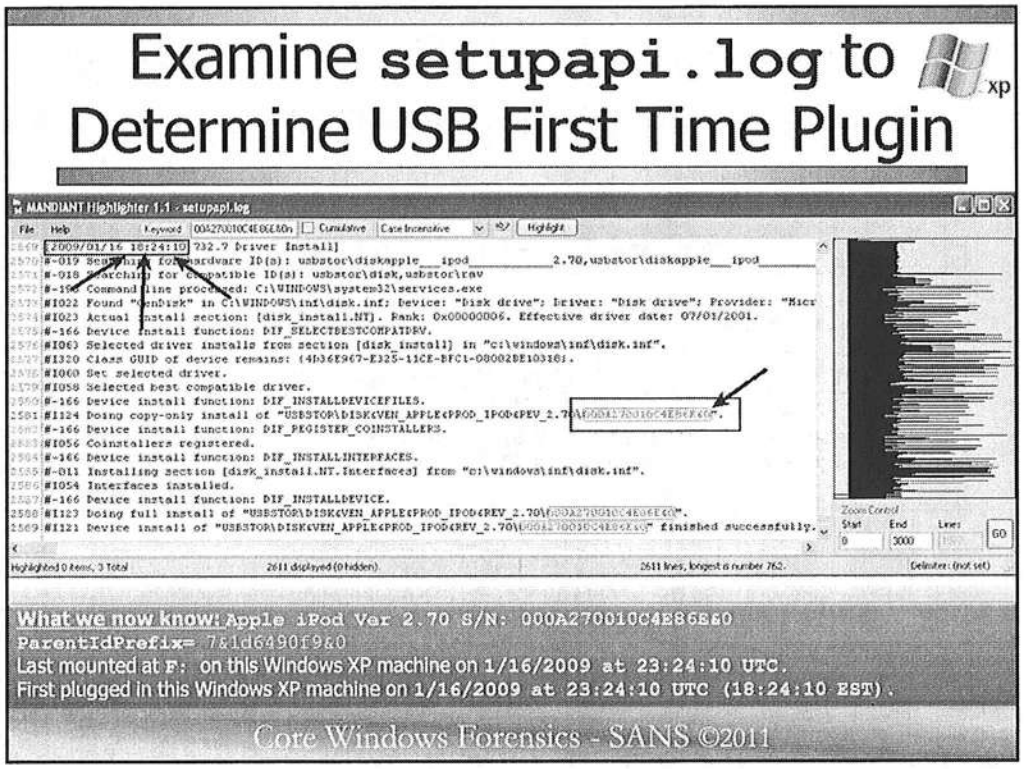
To determine the first time a device is connected is very similar on both Vista, Win7, and XP, but the files are in slightly different locations.

The setupapi.log file is modified during the install of a USB device. The setupapi log keeps track of hotfix installs, device driver installs, and service pack installs for troubleshooting purposes. Microsoft has a decent website that details the information in the log file.

http://www.microsoft.com/whdc/driver/install/setupapilog.mspx

This shows us the device that we previously discussed and the first time it was installed into the system. Correlating this information along with user activity with regripper might give the investigator more delineation of how USB devices are utilized on a system.

On Vista/Win7, it looks as though the SETUPAPI.LOG file has been slightly renamed to SETUPAPI.DEV.log and moved, but the concept is exactly the same This is good to know. It also looks as though the logfile also has been rewritten. It actually is a bit easier to read now. You can clearly see when the device was first plugged into the machine and the time that it occurred.




This is an example of the XP SETUPAPI.LOG file viewed through Mandiant's Highlight to help aid us in searching for the serial number of the USB Device. Once you find the serial number, locating the initial time of install is not a problem.

We add that information to the information we know about the USB device.

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270010C4E86E&0**

ParentIdPrefix= **7&1d6490f9&0**

Last mounted at **F:**

GUID= **1647bd00-e26e-11dd-00096b5312dd**

USB Device utilized by user(s): **Donald Blake**First Time Mounted at **1/16/2009 at 23:24:10 UTC (18:24:10 EST)**

# Determine First Time Device Connected After Last Reboot (1)

xp

## Purpose

- Discover When a USB Device was first connected to a machined after the last reboot

## Location

- **SYSTEM*CurrentControlSet*\\Enum\\USB\\VID_XXXX&PID_YYYY**
  - Perform search for Serial Number (Last Written Time of Serial Number Key)

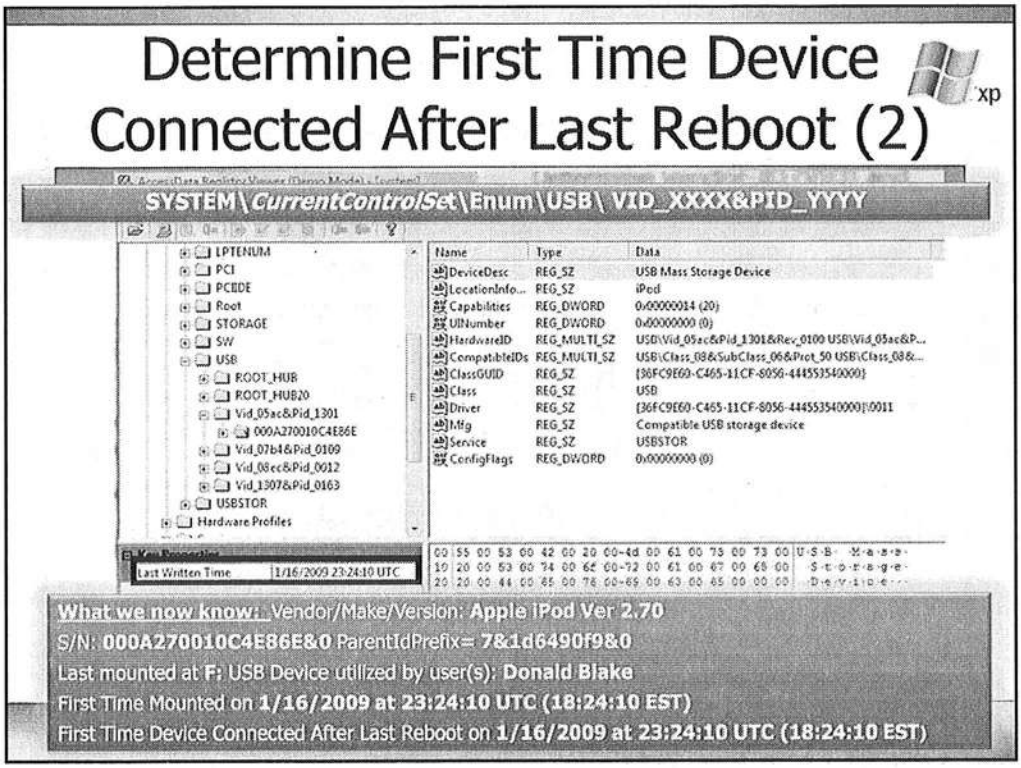
## Why Key Is Useful?

- USBKEY -> Using the Serial Number as the marker, you can determine the first time a specific USB device was connected to the local machine after the last reboot

Core Windows Forensics - SANS ©2011

The *LastWrite* time of this key corresponds to the last time the disk device was connected to the system.

There are two places where this will be recorded. You can always check both locations. The first is the easiest to check as it is located using only the VID and PID which we found earlier in the USB registry key not the USBSTOR key.




**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270010C4E86E&0**

ParentIdPrefix= **7&1d6490f9&0**

VID and PID= **Vid_05ac&Pid_1301**

Last mounted at **F:**

GUID= **1647bd00-e26e-11dd-00096b5312dd**

USB Device utilized by user(s): **Donald Blake**

First Time Mounted on **1/16/2009 at 23:24:10 UTC (18:24:10 EST)**

First Time Device Connected After Last Reboot on **1/16/2009 at 23:24:10 UTC (18:24:10 EST)**

# Determine Last Time Device Connected (1)

## Purpose

- Discover When a USB Device was last connected to a machine

## Location

- NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID}

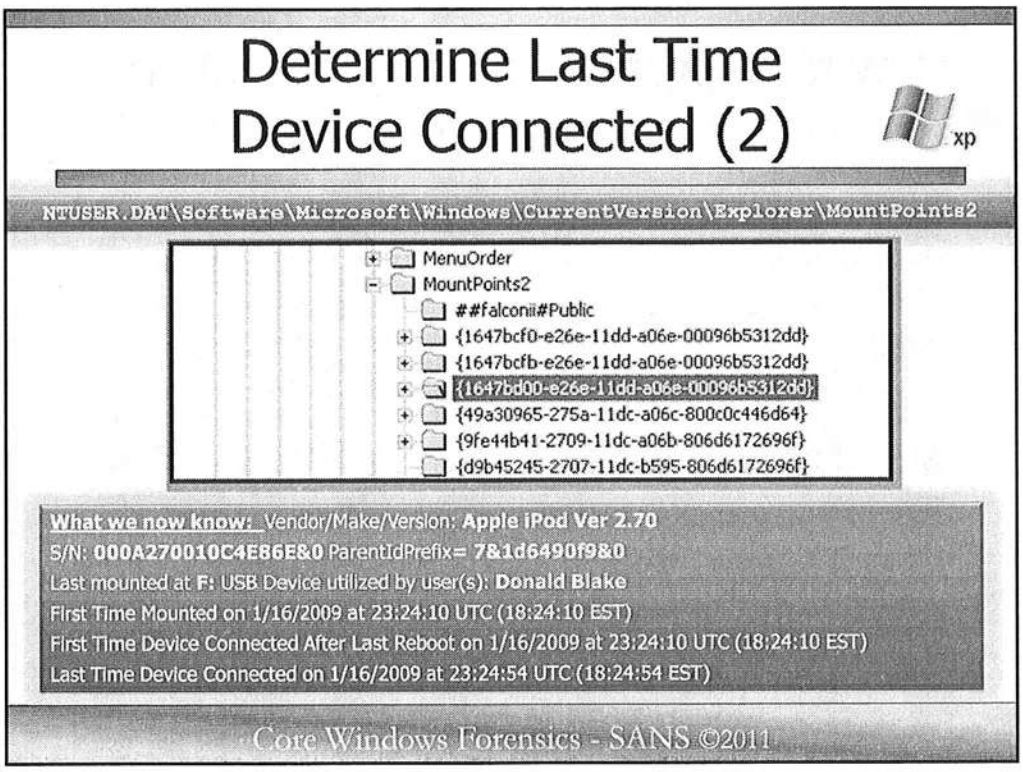
## Why Key Is Useful?

- USBKEY -> Using the Serial Number as the marker, you can determine the last time a specific USB device was last connected to the local machine

Core Windows Forensics - SANS ©2011

The *LastWrite* time of this key corresponds to the last time the disk device was connected to the system.

**NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID}**




**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270010C4E86E&0**

ParentIdPrefix= **7&1d6490f9&0**

VID and PID= **Vid_05ac&Pid_1301**

Last mounted at **F:**

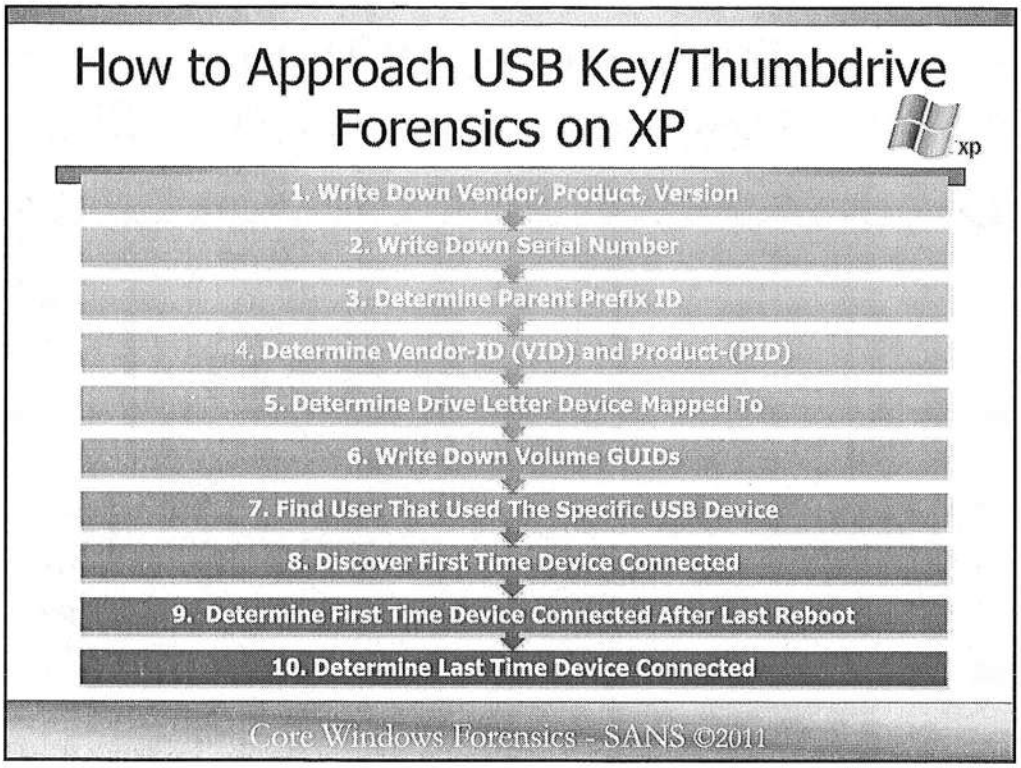
GUID= **1647bd00-e26e-11dd-00096b5312dd**

USB Device utilized by user(s): **Donald Blake**

First Time Mounted on **1/16/2009 at 23:24:10 UTC (18:24:10 EST)**

First Time Device Connected After Last Reboot on **1/16/2009 at 23:24:10 UTC (18:24:10 EST)**

Last Time Device Connected on **1/16/2009 at 23:24:54 UTC (18:24:54 EST)**




**1. Write Down Vendor, Product, Version**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Vendor =

Product =

Version =

**2. Write Down Serial Numbers**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Serial Number =

**3. Determine Parent Prefix ID**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Parent Prefix ID=

**4. Determine Vendor-ID (VID) and Product-(PID)**

**SYSTEM\CurrentControlSet\Enum\USB** -> Perform search for S/N

**VID_XXXX =**

**PID_YYYY =**

**5. Determine Drive Letter Device Mapped To**

**SYSTEM\MountedDevices->** Perform search for ParentIdPrefix in the Drive Letter

Drive =

**6. Write Down Volume GUIDs**

**SYSTEM\MountedDevices->** Perform Search for ParentIdPrefix in the GUIDs

{GUID} =

**7. Find User That Used The Specific USB Device**

**NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2->**

Search for Device GUID

User =

**8. Discover First Time Device Connected**

**C:\Windows\setupapi.log** -> Perform search for Serial Number

Time/Timezone =

**9. Determine First Time Device Connected After Last Reboot**

**SYSTEM\CurrentControlSet\Control\DeviceClasses\{53f56307-b6bf-11d0-94f2-00a0c91efb8b}->** Perform search for S/N

<u>or</u>

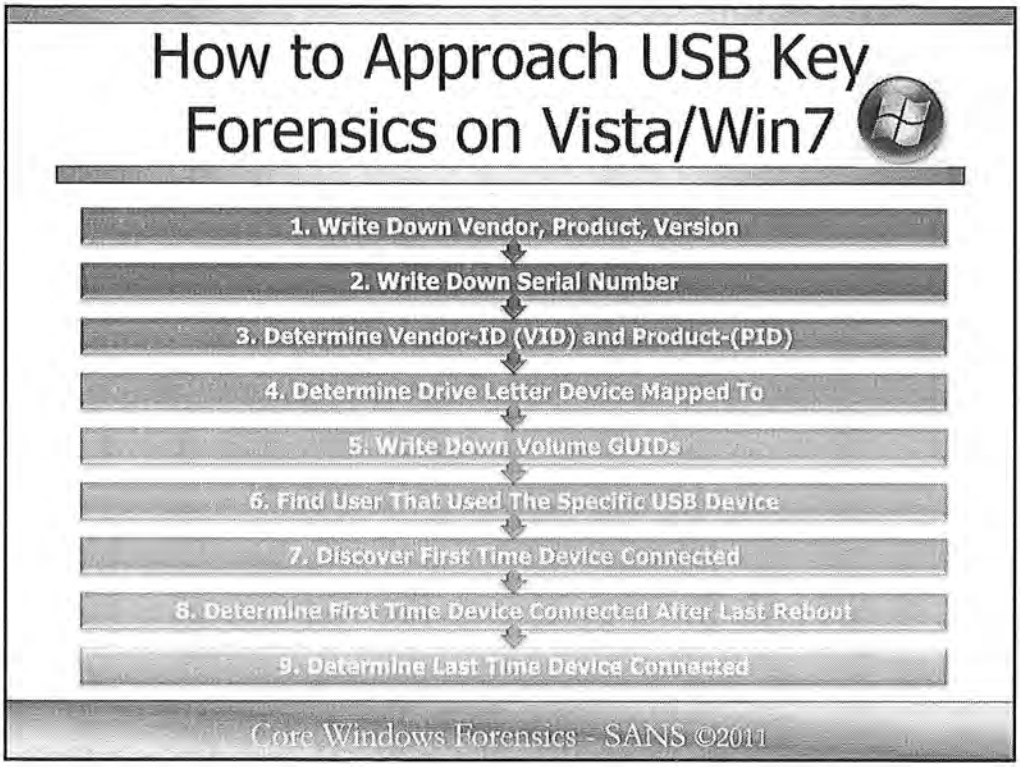
**SYSTEM\CurrentControlSet\Enum\USB\ VID_XXXX&PID_YYYY ->** Perform search for Serial Number (Last Written Time of Serial Number Key)

Time/Timezone =

**10. Determine Last Time Device Connected**

**NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID} ->** Perform search for Device {GUID}

Time/Timezone =




**USB KEY**

**1. Write Down Vendor, Product, Version**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Vendor = [redacted]

Product =

Version =

**2. Write Down Serial Numbers**

**SYSTEM\CurrentControlSet\Enum\USBSTOR** [redacted]

Serial Number = [redacted]

**3. Determine Vendor-ID (VID) and Product-(PID)**

**SYSTEM\CurrentControlSet\Enum\USB ->** Perform search for S/N

**VID_XXXX =** [redacted]

**PID_YYYY =**

**4. Write Down Volume GUIDs**

**SYSTEM\MountedDevices->** Perform Search for Serial Number

GUID =

**5. Determine Drive Letter and Volume Name Device Mapped To**

**SOFTWARE\Microsoft\Windows Portable Devices\Devices->** Perform Search for Serial Number and Match with Volume Name

Drive Letter =

Volume Name=

**6. Find User That Used The Specific USB Device**

**NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2->**

Search for Device GUID

User =

**7. Discover First Time Device Connected**

**C:\Windows\inf\setupapi.dev.log** -> Perform search for Serial Number

Time/Timezone =

**8. Determine First Time Device Connected After Last Reboot**

**SYSTEM\CurrentControlSet\Enum\USBSTOR\ VID_XXXX&PID_YYYY ->** Perform search for Serial Number (Last Written Time of Serial Number Key)

**or**

**SYSTEM\CurrentControlSet\Control\DeviceClasses\{53f56307-b6bf-11d0-94f2-00a0c91efb8b}->** Perform search for S/N (Last Written Time of Key that has Serial Number and Vendor/Product/Revision)

Time/Timezone =

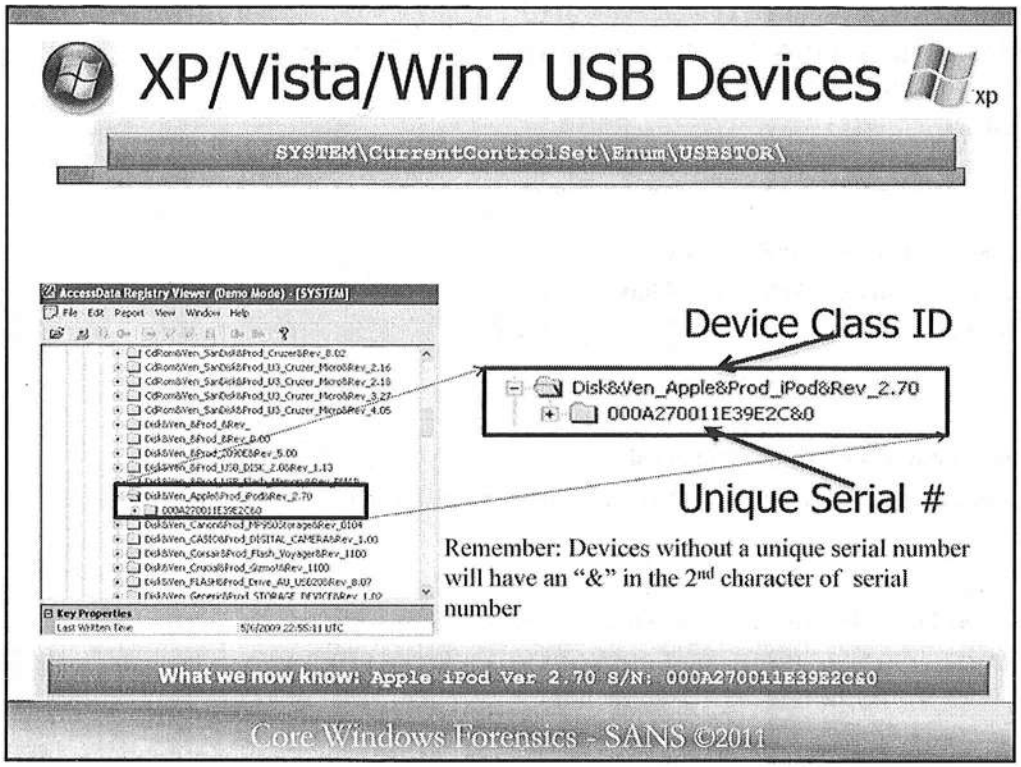
**9. Determine Last Time Device Connected**

**SYSTEM\CurrentControlSet\Enum\USB\ VID_XXXX&PID_YYYY ->** Perform search for Serial Number (Last Written Time of Serial Number Key)

**or**

**NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID} ->** Perform search for Device {GUID}

Time/Timezone =




Performing examination of a Vista USB Device starts exactly the same as one for XP.

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270011E39E2C&0**

# Win7/Vista: Discover the Drive Letter

**Purpose**

- Discover the drive letter of the USB Device when it was plugged in the machine

**Locations**

- `SOFTWARE\Microsoft\Windows Portable Devices\Devices`

**Why Key Is Useful?**

- Identify the USB device that was last mapped to a specific drive letter on a Vista or Windows 7 using the S/N of the USB Device
- Find Serial Number via USBSTOR
  - `SYSTEM\CurrentControlSet\Enum\USBSTOR\`
- Able to identify the volume name of the device that was mapped to a specific USB device and Serial Number

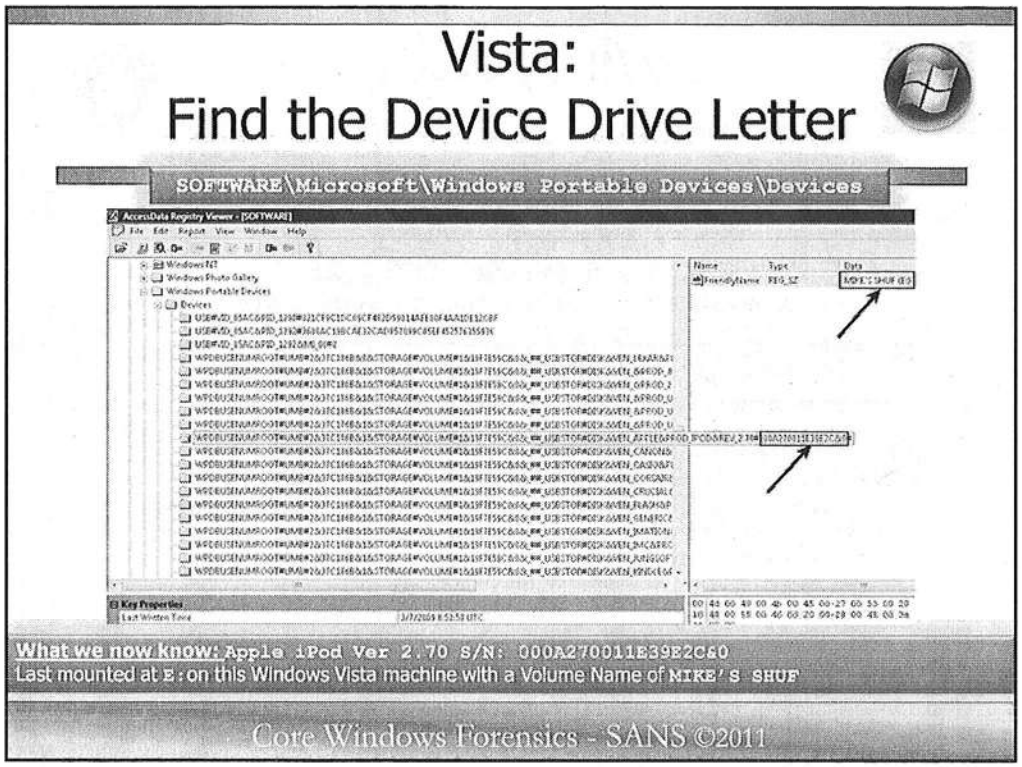
Core Windows Forensics - SANS ©2011

On Win7 and Vista, we no longer utilize the ParentIdPrefix to help us determine the drive letter. In fact, the value for ParentIdPrefix does not exist on Windows Vista USBSTOR entries.

However, we can still map information back to the drive letter by examining a brand new key created on Vista that logs the last drive letter and volume name for each device plugged into the key. You can easily find the drive letter for the device by performing a search for the device's unique serial number.

**`SOFTWARE\Microsoft\Windows Portable Devices\Devices`**

With this key, you are able to identify the USB device that was last mapped to a specific drive letter on a Vista system using the S/N of the USB device. You are also able to identify the volume name of the device that was mapped to a specific USB device and serial number. Windows 7 seems to change this a bit as not every device will have the previous drive letter recorded. With Win7 you will need to check here, as well as use the MountedDevices key with the device GUID, to determine the last mount point of the device.




This is an example of finding the drive letter and the volume name for the USB device we profiled earlier on this Vista machine.

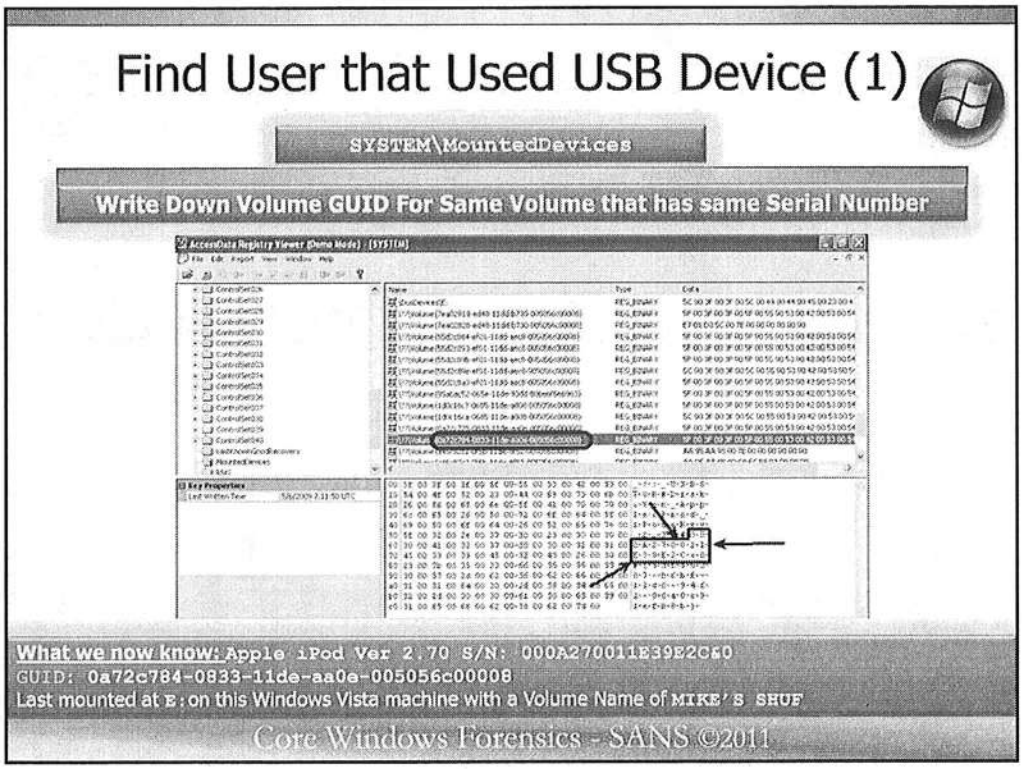
**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270011E39E2C&0**

Last mounted at **E:**

Volume Name is **MIKE'S SHUF**




Just like we did for Windows XP, we will use the GUID of the device to map it back to a specific user that utilized the USB device. This GUID will be used next to identify the user that plugged in the device. The number will be referenced in the user's personal mountpoint's key in the NTUSER.DAT Hive.

**What we now know:**

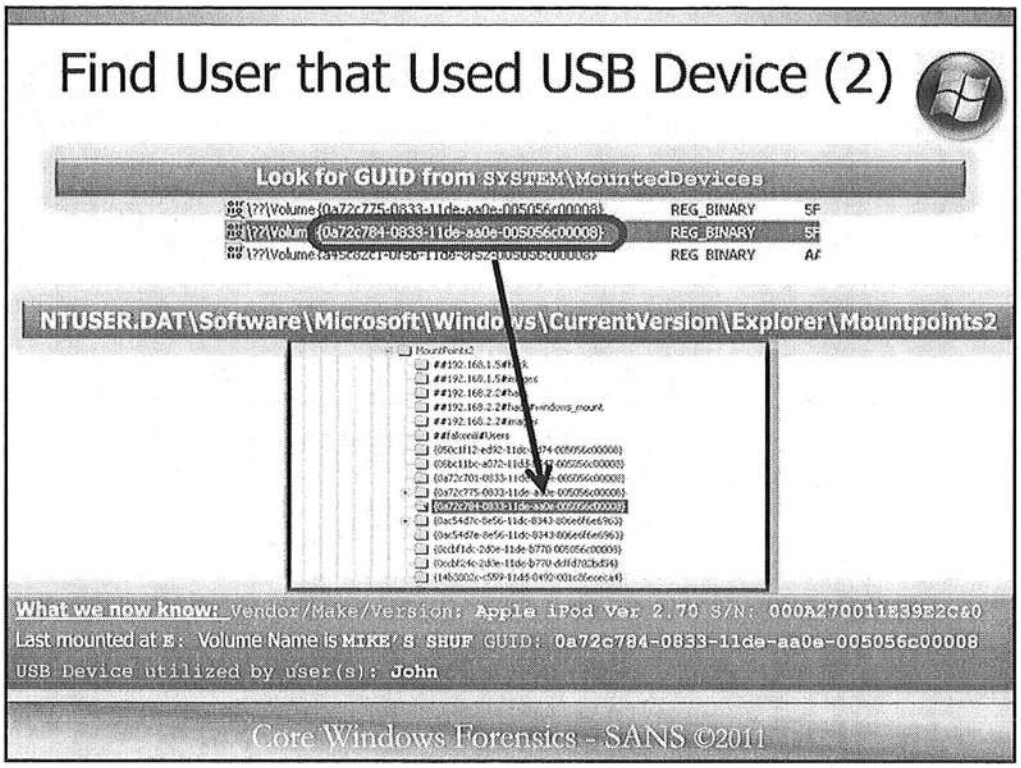
Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270011E39E2C&0**

Last mounted at **E:**

Volume Name is **MIKE'S SHUF**

GUID: **0a72c784-0833-11de-aa0e-005056c00008**




Once we know the GUID we can now utilize that information by searching through the MountPoints2 key found here: **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\Mountpoints2**

If we have multiple users, that could slow the process down, but the GUID will be mapped to the user that inserted the key and appropriately recorded.

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

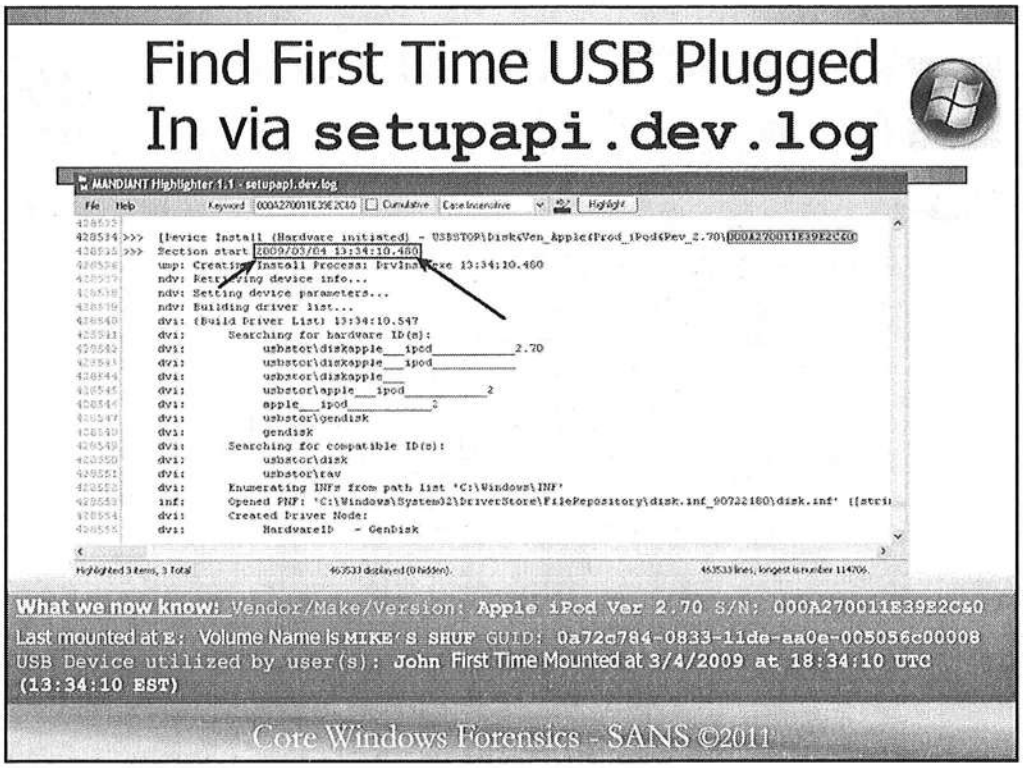
S/N: **000A270011E39E2C&0**

Last mounted at **E:**

Volume Name is **MIKE'S SHUF**

GUID: **0a72c784-0833-11de-aa0e-005056c00008**

USB Device utilized by user(s): **John**




Utilizing the setupapi.dev.log file in C:\Windows\inf directory via Mandiant's Highlighter we can easily view the values that it contains. We again perform a search for the unique serial number and identify the first time the device was plugged into the Windows Vista operating system.

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270011E39E2C&0**

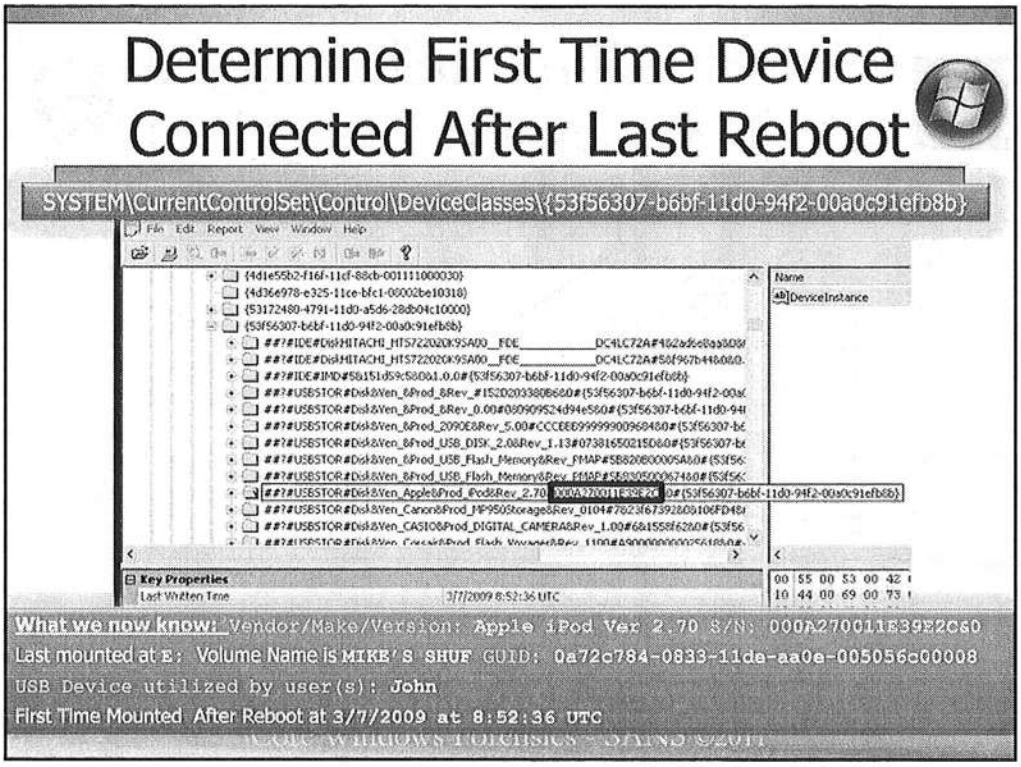
Last mounted at **E:**

Volume Name is **MIKE'S SHUF**

GUID: **0a72c784-0833-11de-aa0e-005056c00008**

USB Device utilized by user(s): **John**

First Time Mounted at **3/4/2009 at 18:34:10 UTC (13:34:10 EST)**




This is exactly the same exercise we accomplished with Windows XP. This time we will use the control key to tell us the information of the last time the device was plugged in after last reboot. Notice we utilized the unique serial number again to track the device to determine the last time it was connected to the machine.

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270011E39E2C&0**

Last mounted at **E:**

Volume Name is **MIKE'S SHUF**

GUID: **0a72c784-0833-11de-aa0e-005056c00008**

USB Device utilized by user(s): **John**

First Time Mounted at **1/16/2009 at 23:24:10 UTC (18:24:10 EST)**

First Time Mounted After Reboot at **3/7/2009 at 8:52:36 UTC (3:53:36 EST)**

# Vista/Win7: Last Time Device Connected

## Purpose

- Discover the last time the device was connected to the computer

## Locations

- **SYSTEM/CurrentControlSet/Enum/USB/VID_XXXX&PID_YYYY**
  - Perform search for Serial Number (Last Written Time of Serial Number Key)
- **NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID}**
  - Perform search for Device {GUID}

## Why Key Is Useful?

- The key is useful because we can determine the last time a device is connected to the machine

Core Windows Forensics - SANS ©2011

On XP, we used the VID_XXXX&PID_YYYY to determine the first time the device was connected after last reboot. This changed on Windows Vista and Windows 7. These keys now directly relate to the last time a device is connected to a machine. Not sure why the change occurred, just know after extensive testing that it did.

**SYSTEM/CurrentControlSet/Enum/USB/ VID_XXXX&PID_YYYY**

Perform search for Serial Number (Last Written Time of Serial Number Key)

**NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID}**

Perform search for Device {GUID}

It should be noted that there are a variety of texts out there that might have different information. Usually that is due to the fact their testing was limited to only Windows XP. Please note that this is changed on Windows VISTA and Windows 7.

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270011E39E2C&0**

Last mounted at **E:**

Volume Name is **MIKE'S SHUF**

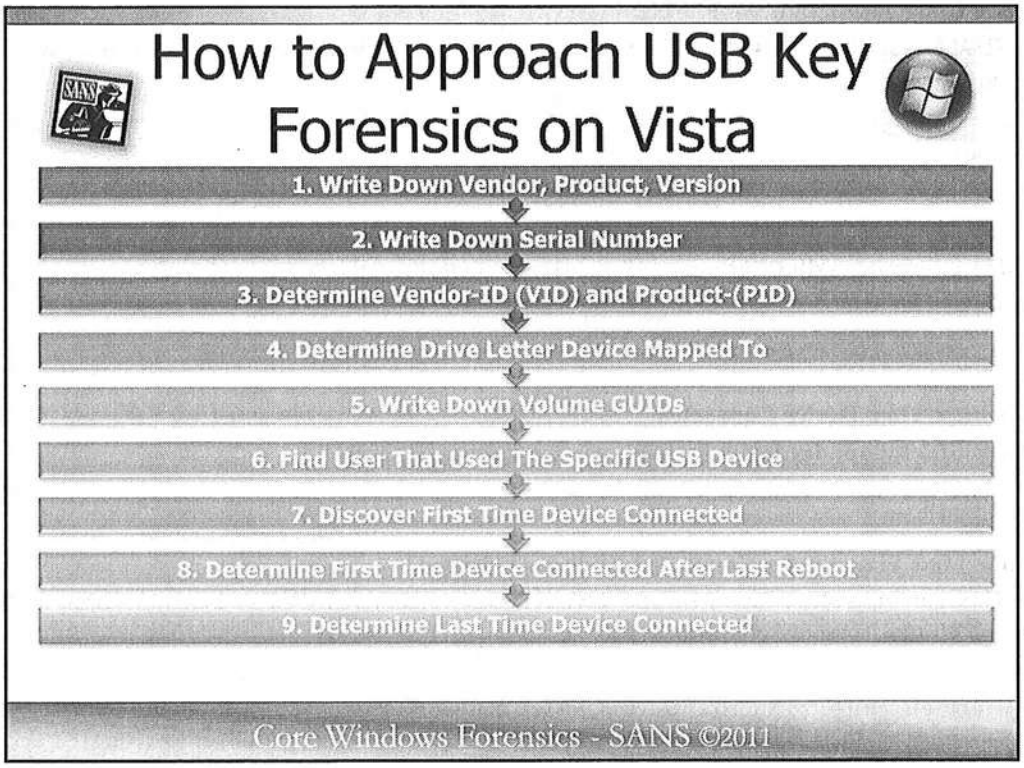
GUID: **0a72c784-0833-11de-aa0e-005056c00008**

USB Device utilized by user(s): **John**

First Time Mounted at **1/16/2009 at 23:24:10 UTC (18:24:10 EST)**

First Time Mounted After Reboot at **3/7/2009 at 8:52:36 UTC (3:53:36 EST)**

Last Time Mounted **3/7/2009 at 8:52:36 UTC (3:53:36 EST)**




**USB KEY**

**1. Write Down Vendor, Product, Version**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Vendor =

Product =

Version =

**2. Write Down Serial Numbers**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Serial Number =

**3. Determine Vendor-ID (VID) and Product-(PID)**

**SYSTEM\CurrentControlSet\Enum\USB ->** Perform search for S/N

**VID_XXXX =**

**PID_YYYY =**

**4. Write Down Volume GUIDs**

**SYSTEM\MountedDevices->** Perform Search for Serial Number

GUID =

**5. Determine Drive Letter and Volume Name Device Mapped To**

**SOFTWARE\Microsoft\Windows Portable Devices\Devices->** Perform Search for Serial Number and Match with Volume Name

Drive Letter =

Volume Name=

**6. Find User That Used The Specific USB Device**

**NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2->**

Search for Device GUID

User =

**7. Discover First Time Device Connected**

**C:\Windows\inf\setupapi.dev.log** -> Perform search for Serial Number

Time/Timezone =

**8. Determine First Time Device Connected After Last Reboot**

**SYSTEM\CurrentControlSet\Enum\USBSTOR\ VID_XXXX&PID_YYYY ->** Perform search for Serial Number (Last Written Time of Serial Number Key)

**or**

**SYSTEM\CurrentControlSet\Control\DeviceClasses\{53f56307-b6bf-11d0-94f2-00a0c91efb8b}->** Perform search for S/N (Last Written Time of Key that has Serial Number and Vendor/Product/Revision)

Time/Timezone =

**9. Determine Last Time Device Connected**

**SYSTEM\CurrentControlSet\Enum\USB\ VID_XXXX&PID_YYYY ->** Perform search for Serial Number (Last Written Time of Serial Number Key)

**or**

**NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID} ->** Perform search for Device {GUID}

Time/Timezone =

**What we now know:**

Vendor/Make/Version: **Apple iPod Ver 2.70**

S/N: **000A270011E39E2C&0**

Last mounted at **E:**

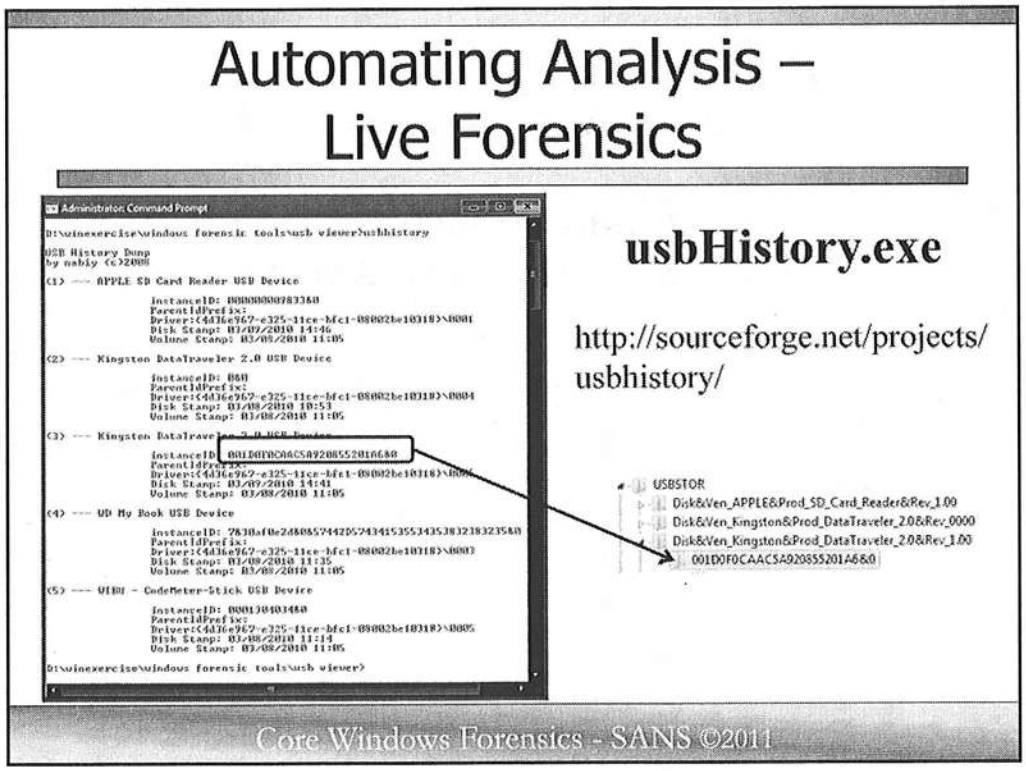
Volume Name is **MIKE'S SHUF**

GUID: **0a72c784-0833-11de-aa0e-005056c00008**

USB Device utilized by user(s): **John**

Last Time Mounted at **3/7/2009 at 8:52:36 UTC (3:53:36 EST)**

First Time Mounted at **3/4/2009 at 18:34:10 UTC (13:34:10 EST)**

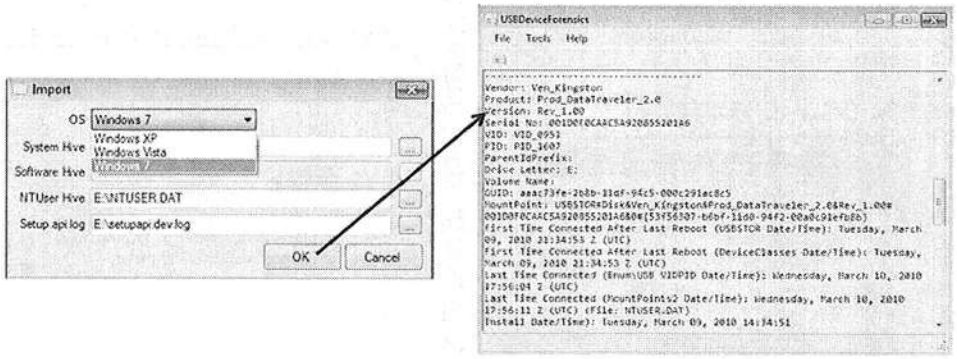



If you are on a live Windows machine you can use the usbHistory.exe program to help you determine the Serial Number and ParentIDPrefix of every USB Device that has been plugged into the registry.

# Automating Analysis – Dead Forensics

**USBDeviceForensics**

http://www.woany.co.uk/usbdeviceforensics/



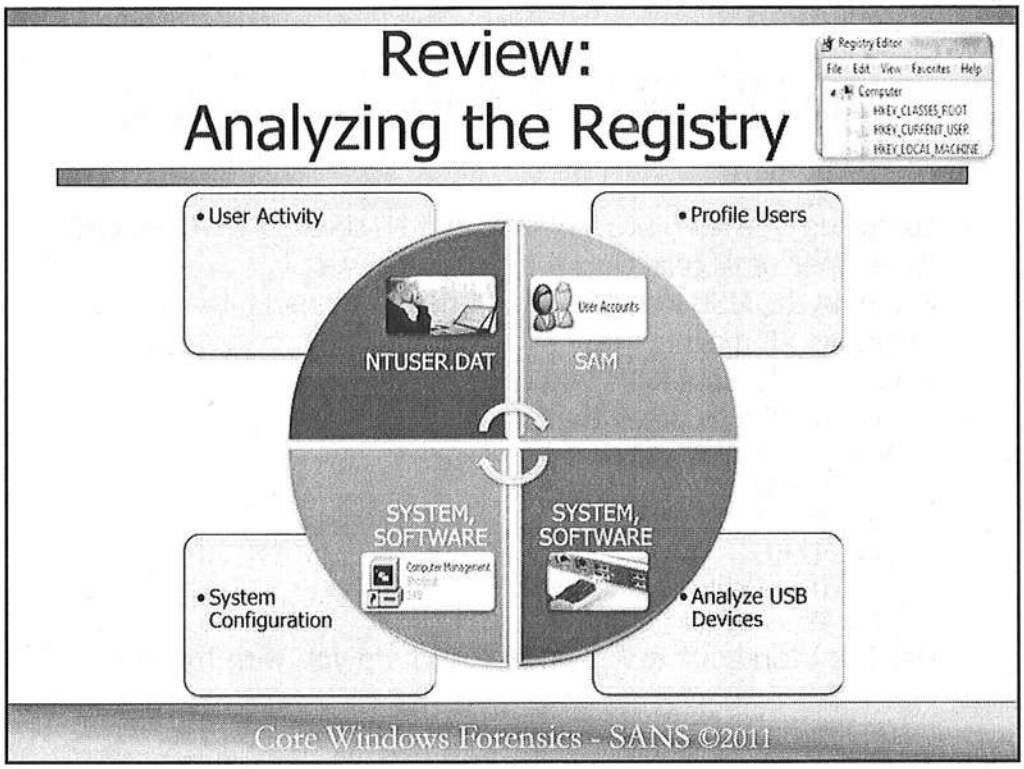

Core Windows Forensics - SANS ©2011

USBDeviceForensics is an application to extract numerous bits of information regarding USB devices.

Per the website:

"It should be noted that whilst the information in the blog posting is accurate, there is a caveat to be aware of. During my testing I have found that an unknown process can update the Date/Time values across all keys, in particular the USBSTOR keys. Therefore, you could see the same Last Written Date/Time value on each device key. If you see this occurring, then you obviously cannot rely on the values retrieved.

All of the dates should be UTC. It is possible to set a time zone offset by using the Time Zone window e.g. Tools->Time Zone menu item. The Install date time zone is currently unknown, it is either UTC or the local time zone of the system where the files were extracted from."




This page intentionally left blank.

# HANDS-ON USB Key Profiling

- Using the SYSTEM, SOFTWARE, and NTUSER.DAT Hives and RegRipper or Access Data's Registry Viewer
- Examine the USB Keys that have been plugged into the Windows XP machine
- There should be 4 keys you need to examine.
- We would like to detail the:
  - Vendor, Product, Version
  - Serial Numbers
  - Parent Prefix ID
  - Drive Letter
  - Last Time Device Connected
  - First Time Device Connected
- Use the Workbook in Appendix C to help you with this exercise

Core Windows Forensics - SANS ©2011

Using the SYSTEM Hive and RegRipper or Access Data's Registry Viewer.

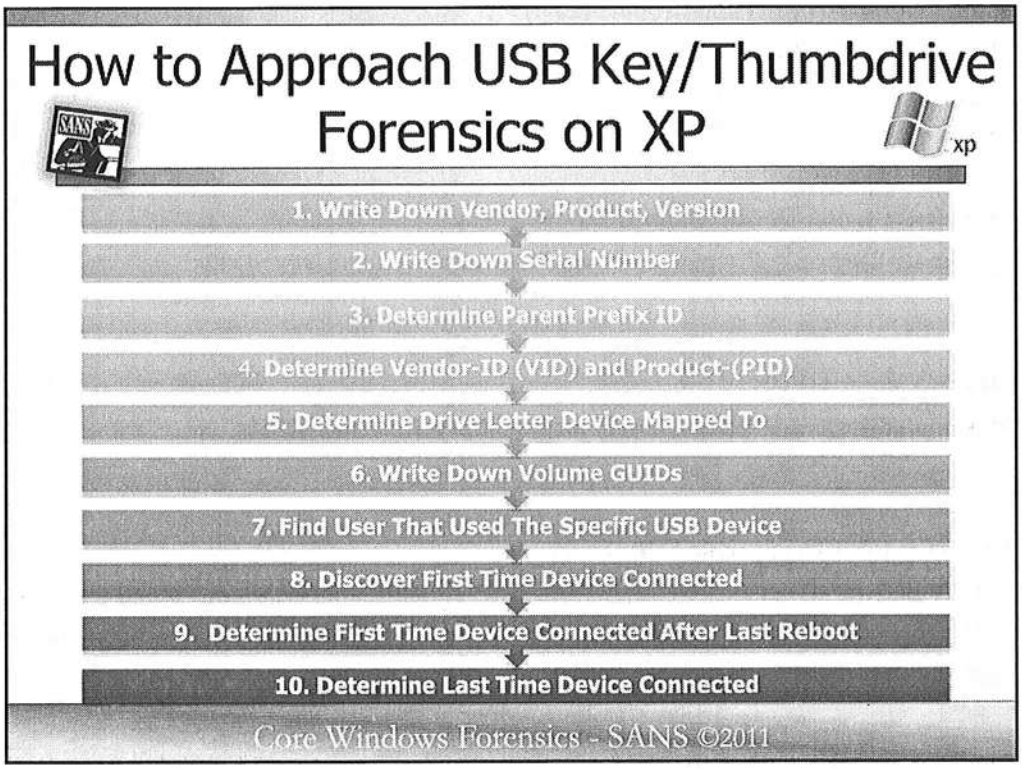
Examine the USB Keys that have been plugged into the Windows XP machine.

There should be 4 keys you need to examine.

We would like to detail the:

Vendor, Product, Version
Serial Numbers
Parent Prefix ID
Drive Letter
Last Time Device Connected
First Time Device Connected

Use the Workbook in Appendix C to help you with this exercise.




This is a reminder of the steps you will step through in Appendix B to help you answer the questions for this exercise.

**1. Write Down Vendor, Product, Version**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Vendor =

Product =

Version =

**2. Write Down Serial Numbers**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Serial Number =

**3. Determine Parent Prefix ID**

**SYSTEM\CurrentControlSet\Enum\USBSTOR**

Parent Prefix ID=

**4. Determine Vendor-ID (VID) and Product-(PID)**

**SYSTEM\CurrentControlSet\Enum\USB ->** Perform search for S/N

**VID_XXXX =**

**PID_YYYY =**

**5. Determine Drive Letter Device Mapped To**

**SYSTEM\MountedDevices->** Perform search for ParentIdPrefix in the Drive Letter

Drive =

**6. Write Down Volume GUIDs**

**SYSTEM\MountedDevices->** Perform Search for ParentIdPrefix in the GUIDs

{GUID} =

**7. Find User That Used The Specific USB Device**

**NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2->**

Search for Device GUID

User =

**8. Discover First Time Device Connected**

**C:\Windows\setupapi.log** -> Perform search for Serial Number

Time/Timezone =

**9. Determine First Time Device Connected After Last Reboot**

**SYSTEM\CurrentControlSet\Control\DeviceClasses\{53f56307-b6bf-11d0-94f2-00a0c91efb8b}->** Perform search for S/N

<u>or</u>

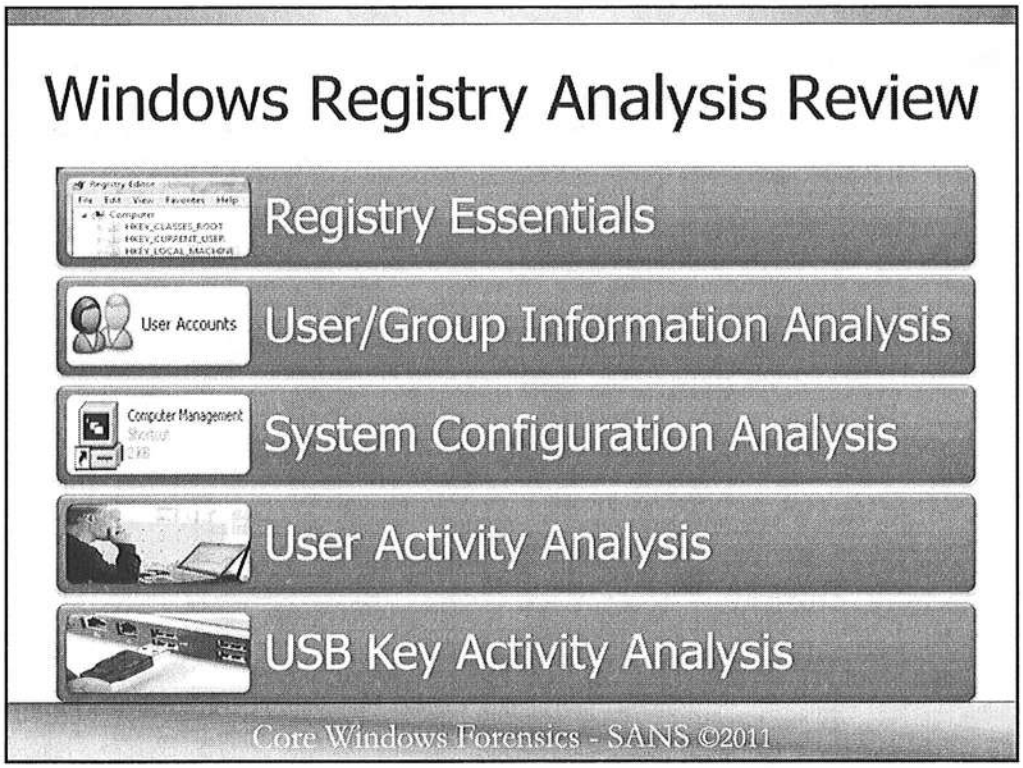
**SYSTEM\CurrentControlSet\Enum\USB\ VID_XXXX&PID_YYYY ->** Perform search for Serial Number (Last Written Time of Serial Number Key)

Time/Timezone =

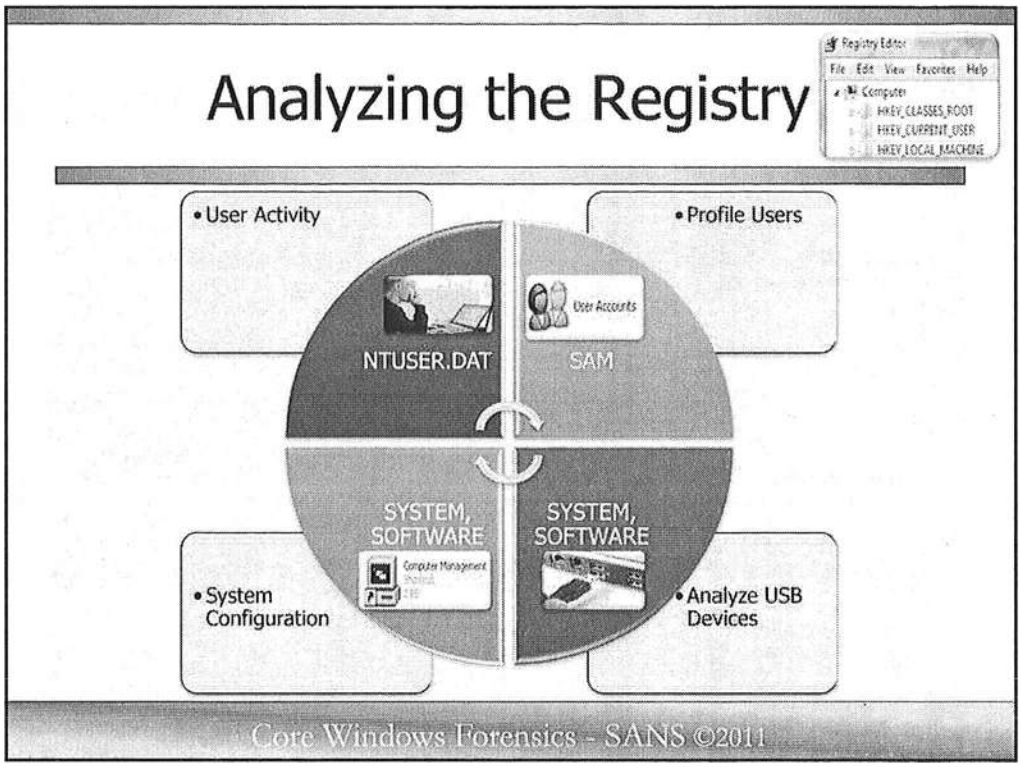
**10. Determine Last Time Device Connected**

**NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID} ->** Perform search for Device {GUID}

Time/Timezone =



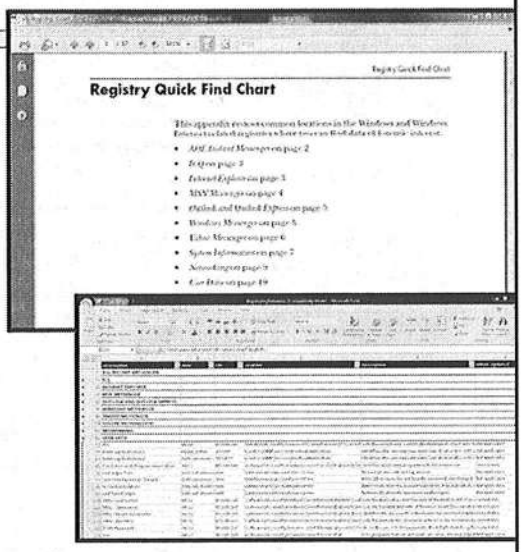

This page intentionally left blank.




This page intentionally left blank.

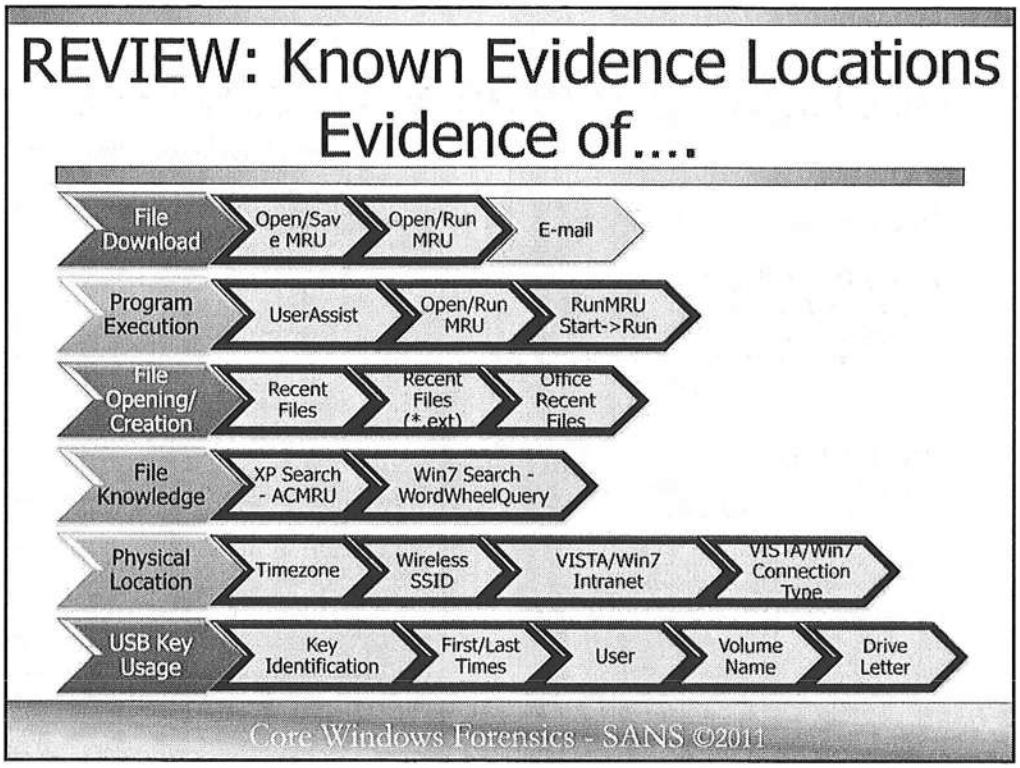
# Additional Reference Documents

- DVD has:
- Access Data's Registry Keys Reference Paper
  `\forensic reference documents\wp.Registry_Quick_Find_Chart.en_us.pdf`
- Registry Reference Excel SpreadSheet.
  - `\forensic reference documents\registry_reference.xls`
  - `\forensic reference documents\RegistryKeyReference.xls`

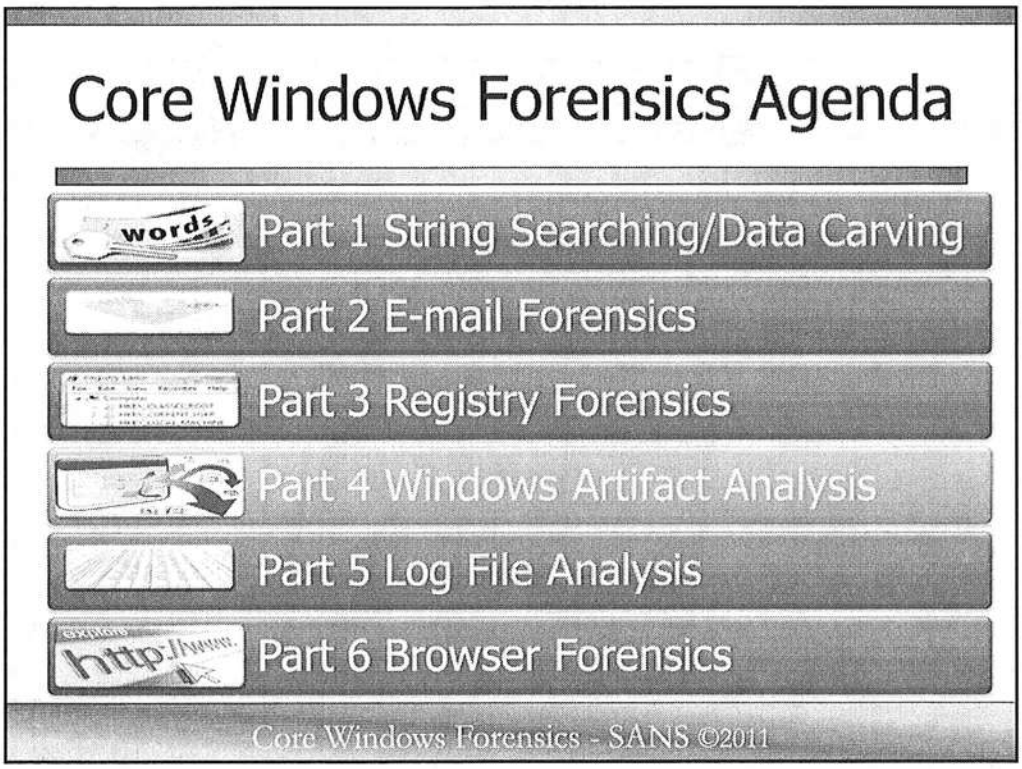




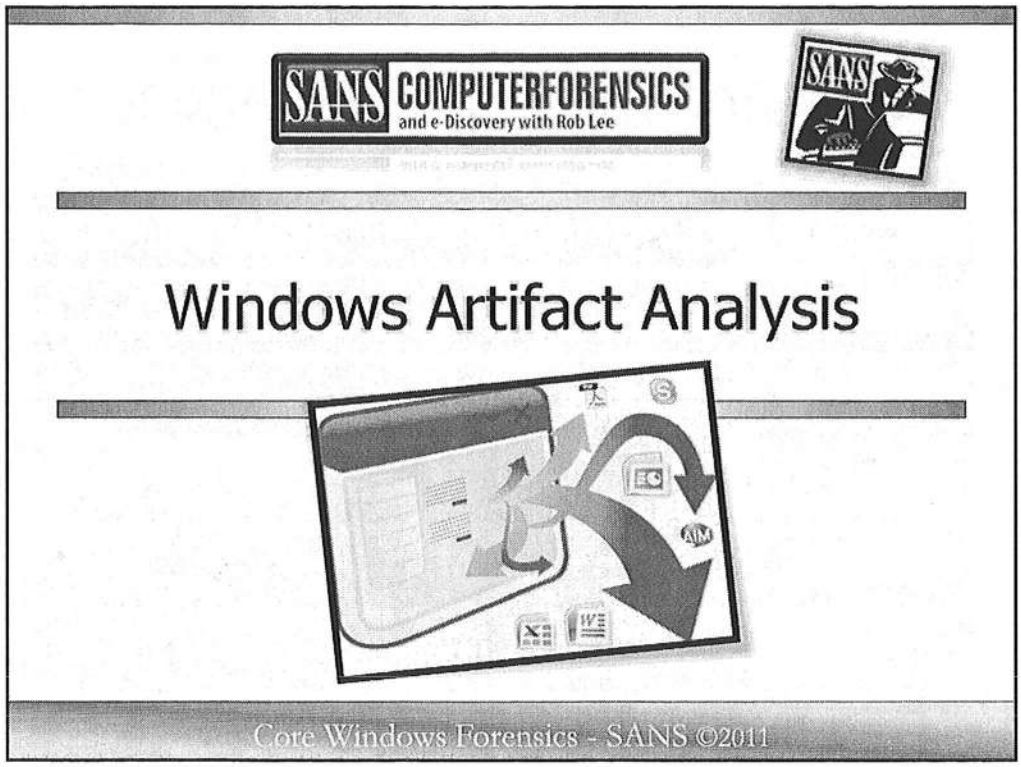

This page intentionally left blank.




This list is meant to help review and identify some of the major items that will aid you in solving cases you are working on. The list does list every artifact, but just some of the major ones we just covered in order to help you, the investigator, know where to look to answer questions that need to be answered during a case.




This page intentionally left blank.




This page intentionally left blank.

# Windows Artifact Analysis

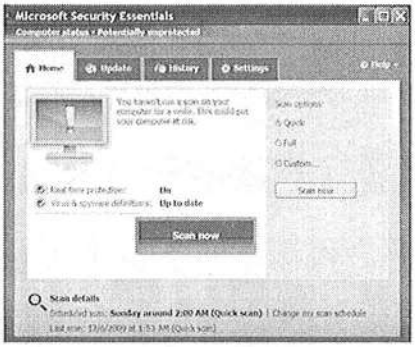
- Link Files
- Microsoft Office Metadata
  - Documents
  - Spreadsheets
  - Powerpoint
  - Adobe PDF Files
- Media File Metadata (EXIF)
- Thumbnail Analysis
  - XP Thumbs.db
  - Vista/Win7 Thumbscache
- Internet Chat and Web E-mail Extraction
  - Internet Evidence Finder
  - Skype
- Recycle Bin
  - XP Recycle Bin
  - Vista/Win7 Recycle Bin
- Windows Prefetch



This page intentionally left blank.

# Anti Virus Scan

- Run the remote mounted drive through an Anti Virus Scanner with the latest updates

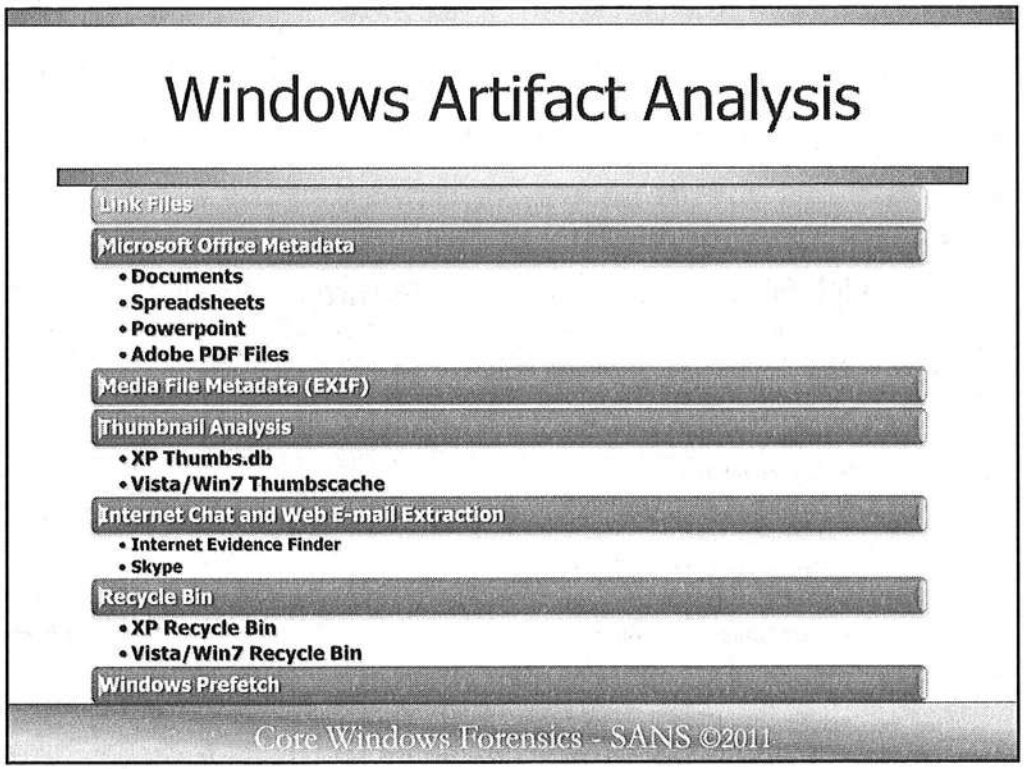


Core Windows Forensics - SANS ©2011

Anti Virus scanners are tools you should quickly run on your system when you mount the drive for analysis. Anti-virus scanners are also useful in incident response verification as it can quickly parse through your file system looking for malicious files. However, ensure that your anti-virus software is configured correctly or it may automatically quarantine the evidence you are searching for.

Usually once I have the system mounted as a read-only drive, I will scan the system for any malware or suspicious items. I would point the scanner to the E:\ drive the file is located at.

Another option is to use an online virus scanner to scan the E:\ you have mounted, such as EsetOnline with NOD32 or Kaspersky.




We will begin this section parsing metadata files. Over the next part of this course we will begin to understand that many files typically, but not always have embedded data inside the file itself. This data is called metadata. Which mean data about data. The existence of metadata has caused much controversy in the legal world on whether metadata should be provided during any type of litigation.

In many situations, the legal teams fail to request files with embedded metadata. If that occurs, the opposing party sometimes has requested the metadata to be provided later. Judges have ruled, in certain situations, that unless specifically asked for at the beginning of the litigation that people have to work with what had been given to them. This is called "playing it as it lies" in the e-discovery world.

# Document/File Metadata

- Files/Documents Contain Internal File Metadata
  - Link Files
    - Location
    - Timestamps
    - Type of Location
  - Office Documents
    - Document Title
    - Version of MS Office Word
    - Last Author
    - Document Creation Date/Time
    - Last Saved Date/Time
    - Last Printed Date/Time
  - Picture/Media Files
    - EXIF Data
    - Camera Make/Model
    - Creation Time
    - Modification Time
    - Photo Information
    - GPS Coordinates (iPhone and other newer cameras)

Core Windows Forensics - SANS ©2011

Office internal metadata is a key point of discussion for e-discovery related cases. This metadata is embedded into the file itself and includes information that might be quite useful for a variety of litigation reasons. In many court cases, the judges have ruled that metadata specifically needs to be requested, otherwise the party is under no obligation to provide it.

What kind of metadata might be included within Microsoft Office Documents, Picture Files, and Link Files? In most cases, it is data about the file itself. Similar to an MP3 metadata (artist, album, rating), metadata would include items that have nothing to do with the actual file except it helps describe how the data was created, the location, and author.

This type of information might be key to a case where one side might want to show a picture was taken at a particular location from a specific camera. Office documents that are stolen and show up inside a different company could be examined to prove that the document originally came from another company.

It is no wonder that many judges are wrestling with metadata requests surrounding e-discovery especially. It could make or break a case.

# Shortcut Files (.lnk)

C:\Documents and Settings\<username>\Recent\

C:\Users\<user>\AppData\Roaming\Microsoft\Windows\Recent\

C:\Users\<user>\AppData\Roaming\Microsoft\Office\Recent\

- Shortcut Files automatically created by Windows
  - Recent Items
  - Open local and remote data files and documents will generate a shortcut file (`lnk`)
  - Shortcut Files will contain:
    - MACtimes of the original file
    - Volume Information (Name, Type)
    - Network Share information
    - Original Location
    - Name of System

Core Windows Forensics - SANS ©2011

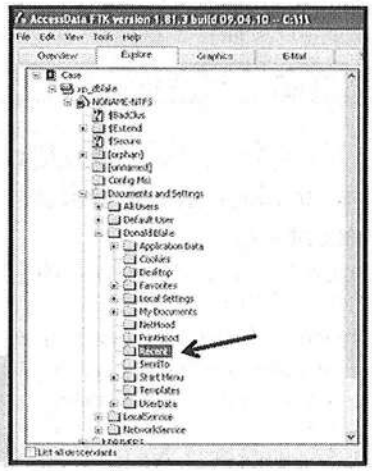
While an individual user might not necessarily create link or shortcut files on a system, the Windows operating system definitely does. Windows creates link files when a user is opening and utilizing media files that show up in Recent Documents/Files folder. If a user opened up and edited a file from a USB key, but never copied the file to the system, that link file will be created in the Recent Items folder under that user account.

Inside the link file will contain the original MACtimes of the file, the path to the location of the file, and the volume information including the drive letter or the network share where the file is found.

I am not sure how many files the Recent folder will store. I noticed on my Win7 machine that it had over 160 .lnk files that were created dating back to almost six months ago.

# Finding the Recent Folder (1)

- To open link files:
- Click on Explore Tab
- Browse to appropriate Recent document folder



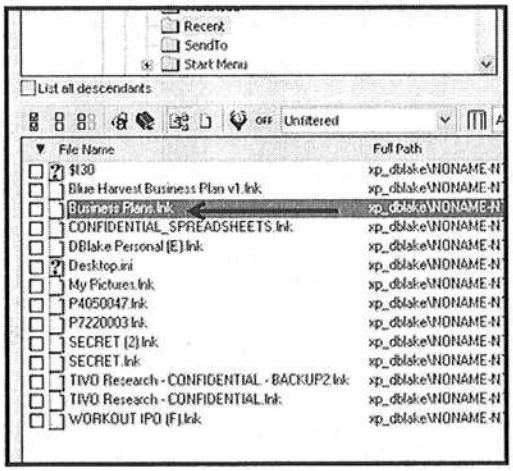

Core Windows Forensics - SANS ©2011

Where will most of the link files you might be interested in reside? Usually the Recent folder under a user directory would be an excellent location to start searching from.

Start by using the information on the previous slide to locate the RECENT folder for your particular operating system. Then utilize FTK or EXPLORER to open that directory and you will see the .lnk files that have been generated by the operating system itself.

# Finding the Recent Folder (2)

- To open link files:
- Click on Explore Tab
- Browse to appropriate Recent document folder




Core Windows Forensics - SANS ©2011

Once you have located the Recent folder, highlight the file that you would like to examine. This will open up the link summary information in the viewer above. The summary information will include data such as the timestamps, the original location, and the location type. This information could prove valuable during USB examination where you can clearly see a file that was opened via a USB device, but was never saved locally on the system itself.

Doing USB key analysis would show that the device was last plugged in and had a specific drive letter. Looking at the time the device was last plugged in, the .lnk file creation date, and when the file was last edited you can easily come to a conclusion that the file is located on a specific USB device.

## Shortcut (.lnk)- Time of First/Last Open

**Date/Time File by that name was first opened**

- Creation Date of Shortcut File

**Date/Time File by that name was last opened**

- Last Modification Date of Shortcut File

E:\Documents and Settings\Donald Blake\Recent

First Opened | Last Opened

| Name | Type | Date Created | Date Modified |
|---|---|---|---|
| WORKOUT IPO (F) | Shortcut | 1/16/2009 6:27 PM | 1/16/2009 6:27 PM |
| TIVO Research - C... | Shortcut | 1/16/2009 6:27 PM | 1/16/2009 6:27 PM |
| SECRET (2) | Shortcut | 1/16/2009 6:14 PM | 1/16/2009 6:25 PM |
| SECRET | Shortcut | 1/16/2009 6:14 PM | 1/16/2009 6:25 PM |
| DBlake Personal (E) | Shortcut | 1/16/2009 6:18 PM | 1/16/2009 6:21 PM |
| CONFIDENTIAL_SP... | Shortcut | 1/16/2009 6:14 PM | 1/16/2009 6:21 PM |
| TIVO Research - C... | Shortcut | 1/16/2009 6:13 PM | 1/16/2009 6:18 PM |
| Blue Harvest Busine... | Shortcut | 1/16/2009 6:13 PM | 1/16/2009 6:18 PM |
| Business Plans | Shortcut | 1/16/2009 6:13 PM | 1/16/2009 6:14 PM |
| P7220003 | Shortcut | 1/14/2009 4:04 PM | 1/14/2009 4:04 PM |
| My Pictures | Shortcut | 1/14/2009 4:01 PM | 1/14/2009 4:04 PM |
| P4050047 | Shortcut | 1/14/2009 4:01 PM | 1/14/2009 4:04 PM |
| Desktop.ini | Configuratio... | 6/30/2007 5:36 PM | 6/30/2007 5:36 PM |

The file system timestamps of the LNK file will show you some important details. The creation date of the file will be the first time a file that has that exact name has been opened regardless of location. The last modification date of the file will be the last time a file of that exact name has been opened regardless of location.

Link files are often useful in identifying files that no longer exist on a local machine. They might have been wiped or deleted, but often the link file will remain and provide an important clue that a file used to exist at that point.

# Fixed Drive Link

**Shortcut File**

| | |
|---|---|
| **Link target information** | |
| Local Path | C:\Documents and Settings\Donald Blake\My Documents\Business Plans |
| Volume Type | Fixed Disk |
| Volume Serial Number | A04E-0001 |
| File size | 0 |
| Creation time (UTC) | 1/16/2009 11:14:26 PM |
| Last write time (UTC) | 1/16/2009 11:14:27 PM |
| Last access time (UTC) | 1/16/2009 11:14:43 PM |
| **File attributes** | |
| Directory | |
| **Optional fields** | |
| Relative Path | \My Documents\Business Plans |
| **Target system information** | |
| NetBIOS name | asgard |
| MAC address | 00-09-6b-53-12-dd |

Core Windows Forensics - SANS ©2011

This is an example of lslnk parsing a link file where the original file was found on a local volume. At which will help us identify the computer system where the Link file was written from. At the bottom of this graphic you'll notice two additional items in the link file that are particular interesting. First you see the NetBIOS name which in this case is Asgard second you see the Mac address that is also listed at the bottom which will help us identify the computer system that generated this link file.

[redacted]

# Removable Drive Shortcut

**Shortcut File**

| **Link target information** | |
|---|---|
| Local Path | E:\CONFIDENTIAL_SPREADSHEETS.zip |
| Volume Type | Removeable Disk |
| Volume Label | [illegible] Personal |
| Volume Serial Number | F434-F590 |
| File size | 70725 |
| Creation time (UTC) | 1/16/2009 11:18:15 PM |
| Last write time (UTC) | 1/16/2009 10:49:52 PM |
| Last access time (UTC) | 1/16/2009 11:20:59 PM |
| **File attributes** | |
| Archive | |
| **Target system information** | |
| NetBIOS name | asgard |
| MAC address | 00-09-6b-53-12-dd |

Core Windows Forensics - SANS ©2011

This is an example of FTK parsing a link file where the original file was found on a removable volume. Notice the Volume Name, that is the volume name of the USB key when it was plugged in the machine. If this were a Vista/Win7 machine, that name would clearly be seen in the registry artifacts for the Windows Portable Devices registry key.

In this example, the link file points to the removable desk that has of a file on it. The file is E: confidential_spreadsheets.zip. Earlier when we perform registry analysis we noticed several USB keys that have been plugged in the machine. One of the keys was mapped to a drive letter named E: Can you go back in your notes and remember the specific serial number of that device and when the last time that device was plugged in?

# Network Based Shortcut Link

**Shortcut File**

| | |
|---|---|
| **Link target information** | |
| Network Path | \\FALCONII\USERS\Rob\Documents\Incident Response.docx |
| File size | 51446 |
| Creation time (UTC) | 2/29/2008 10:32:35 PM |
| Last write time (UTC) | 3/2/2008 11:54:51 PM |
| Last access time (UTC) | 3/2/2008 11:54:51 PM |
| **File attributes** | |
| Archive | |
| **Optional fields** | |
| Working directory | \\falcon\Users\Rob\Documents |
| **Target system information** | |
| NetBIOS name | falcon |
| MAC address | 00-50-56-c0-00-08 |

Core Windows Forensics - SANS ©2011

This is an example of FTK parsing a link file where the original file was found on a network share. Notice in this example<,> a file is located on a network share of the computer system named FalconII. Notice that the MAC address at the bottom of this link file has a different MAC address than the two previous link files. Since this is a network-based link file, it reports the MAC address of the remote system hosting the file share.

# HANDS-ON: Link File Analysis

- Examine The Link Files

  C:\Documents and Settings\<username>\Recent\

- Fill out Table on Next Page
  - File Name
  - First Time Opened
  - Last Time Opened
  - File Creation Date
  - File Modification Date
  - Current Location

Core Windows Forensics - SANS ©2011

This page intentionally left blank.

# HANDS-ON: Link File Analysis

| Name | First Opened | Last Opened | Creation Date | Modification Date | Current Location | Type of Volume/ VolName |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |



This page intentionally left blank.

# Windows Artifact Analysis

**Link Files**

**Microsoft Office Metadata**

- Documents
- Spreadsheets
- Powerpoint
- Adobe PDF Files

**Media File Metadata (EXIF)**

**Thumbnail Analysis**

- XP Thumbs.db
- Vista/Win7 Thumbscache

**Internet Chat and Web E-mail Extraction**

- Internet Evidence Finder
- Skype

**Recycle Bin**

- XP Recycle Bin
- Vista/Win7 Recycle Bin

**Windows Prefetch**

Core Windows Forensics - SANS ©2011

This page intentionally left blank.

# Metadata in Media Files
## Exif Parsing

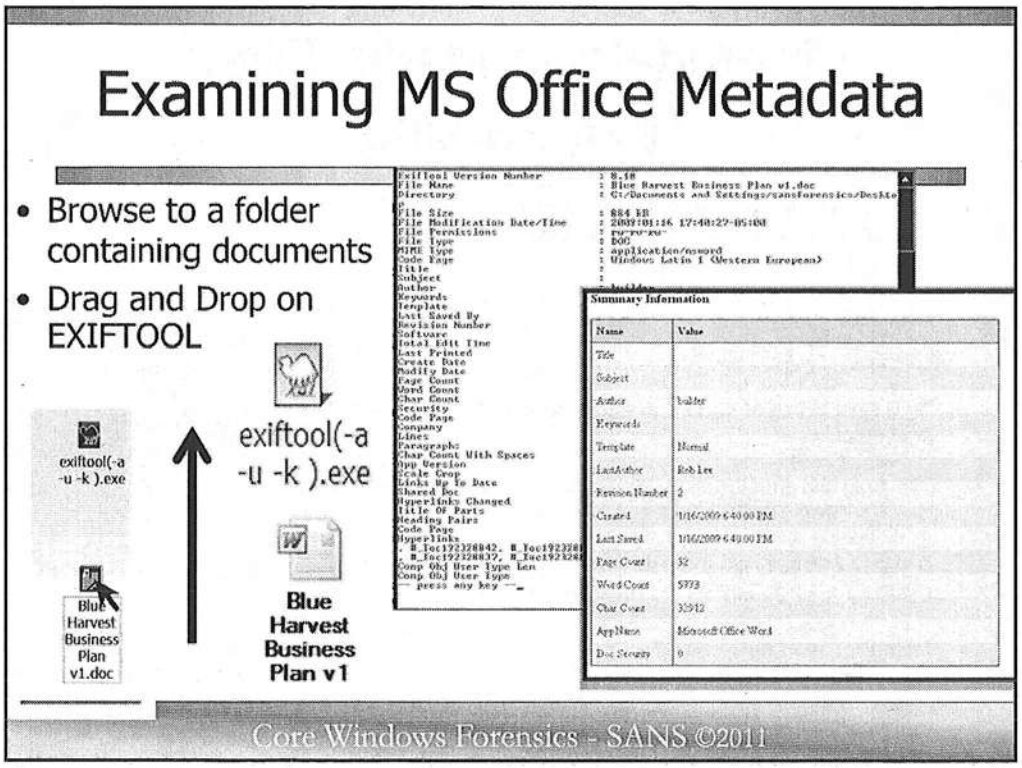
- The metadata tags defined in the internal metadata standard cover a broad spectrum including:
  - Pictures
  - Office Documents
  - Audio Fields
  - Video Fields
  - Executable Files
- Tool Needed to Metadata in all these data formats easily and quickly
  - `Exiftool`
  - Drag and drop file on `exiftool` or execute from command line against picture.
- `http://owl.phy.queensu.ca/~phil/exiftool/`
- `Updated Regularly - Update Often!`
- Location: Icon on Desktop

Core Windows Forensics - SANS ©2011

Every picture will have information embedded inside the picture about how the picture was taken. This useful information will include the original picture creation date and the type of camera that was used to take the picture. This information could be critical to an investigation especially if you are trying to link a user's camera to pictures that have shown up somewhere on the Internet. This fingerprint can be pulled out using a tool to parse the Exif data embedded in the picture, audio, or video file.

One of these tools is the exiftool which is a universal parser for these types of files. The program can be downloaded from http://owl.phy.queensu.ca/~phil/exiftool/.

You can simply drag and drop a picture file on top of the exiftool icon to get it to parse the picture's Exif data.




Examining the metadata of the Microsoft Office documents in the SIFT Workstation is fairly trivial. Simply browse to the folder containing Microsoft Office documents and click on a document of interest and drag and drop it on top of the EXIFTOOL on your desktop. When you do<,> in the viewing pane summary information will show the relevant metadata for that Microsoft Office document.

The metadata of the Microsoft Office document will include: author information, creation time, last print time, and in some cases the Microsoft Office version that created it.

# Examining PowerPoint Presentations

- Send To: Exiftool
- FTK: Browse to a folder containing documents -> **SummaryInformation**
- Examine:
  - Title
  - Author Information
  - Last Author
  - Edit Time
  - Created Time
  - Last Saved Time
  - Last Printed
  - And More

```
Exiftool Version Number          : 8.18
File Name                        : USAFA_Preso.ppt
Directory                        : C:/Documents and Settings/sansforensics/Deskto
p
File Size                        : 11 MB
File Modification Date/Time      : 2010:04:17 13:51:03-05:00
File Permissions                 : rw-rw-rw-
File Type                        : PPT
MIME Type                        : application/vnd.ms-powerpoint
Current User                     : rob
Code Page                        : Windows Latin 1 (Western European)
Title                            : Title of presentation
Author                           : Rob
Template                         : 2009 Mandiant presentation template
Last Saved By                    : rob
Revision Number                  : 403
Software                         : Microsoft Office PowerPoint
Total Edit Time                  : 4.1 days
Create Date                      : 2007:07:19 07:18:06
Modify Date                      : 2010:04:17 18:51:02
Word Count                       : 16752
Thumbnail Clip                   : (Binary data 57736 bytes, use -b option to ext
ract)
Code Page                        : Windows Latin 1 (Western European)
Presentation Targ
Bytes
Paragraphs
Slides
Notes
Hidden Slides
MM Clips
App Version
Scale Crop
Links Up To Date
Shared Doc
Hyperlinks Changn
Title Of Parts
Courier New, Tine
tation template,
```

Summary Information

| Name | Value |
|---|---|
| Title | Slide 1 |
| Author | Rob |
| Template | SE_template |
| LastAuthor | Rob Lee |
| Revision Number | 429 |
| Edit Time | 1:10:15 AM |
| Last printed | 4/21/2004 10:58:01 PM |
| Created | 11/09/2001 5:11:17 PM |
| Last Saved | 5/21/2008 7:30:30 AM |
| Word Count | 5619 |
| AppName | Microsoft PowerPoint |

Core Windows Forensics - SANS ©2011

Starting in the upper right-hand corner is an example of using EXIFTOOL to examine a PowerPoint presentation. In this case, you can see similar information in the Microsoft Office PowerPoint presentation that you see in a Microsoft Office Word document.

One of the metadata properties that I see in this PowerPoint presentation that I did not notice in other metadata for Microsoft Office products is something called the edit time. This metadata property fascinates me. If someone creates a new PowerPoint presentation and copies slides into that presentation, then the edit time would be extremely small. However, if someone actually wrote the PowerPoint presentation, then the added time would be quite large.

# Examining Excel Spreadsheets

- Send To: EXIFTOOL
- FTK: Browse to a folder containing documents -> **`SummaryInformation`**
  Examine:
  - Date Information
  - Author Information
  - Last Printed Time
  - Created Time
  - Last Saved Time
  - And More

```
File Name                       : Copy of 072707_CS_Pensacola_Sec508_Delucia_Sec
res_Day4-5_adjust_RL.xls
Directory                       : C:/Documents and Settings/sansforensics/Deskto
p
File Size                       : 148 kB
File Modification Date/Time     : 2007:08:19 07:06:24-05:00
File Permissions                : rw-rw-rw-
File Type                       : XLS
MIME Type                       : application/vnd.ms-excel
Code Page                       : Windows Latin 1 (Western European)
Title                           : Eval Master Track 1
Author                          : Zoe Dias
Last Saved By                   : Rob
Software                        : Microsoft Excel
Last Printed                    : 2007:04:22 19:00:05
Create Date                     : 2000:06:21 22:32:49
Modify Date                     : 2007:08:19 12:06:24
Security                        : 0
Code Page                       : Windows Latin 1 (Western European)
Manager                         : Stephen Northcutt
Company                         : SANS Institute
App Version                     : 12.0000
Scale Crop                      : No
Links Up To Date                : No
Shared Doc                      : No
Hyperlinks Changed              : No
Title Of Parts                  : Summary, 508.1, 508.2, 508.3, 508.4, 508.5, 5
8.6
Heading Pairs                   : Worksheets, 7
Comp Obj User Type Len          : 38
Comp Obj User Type              : Microsoft Office Excel 2003 Worksheet
```

**Summary Information**

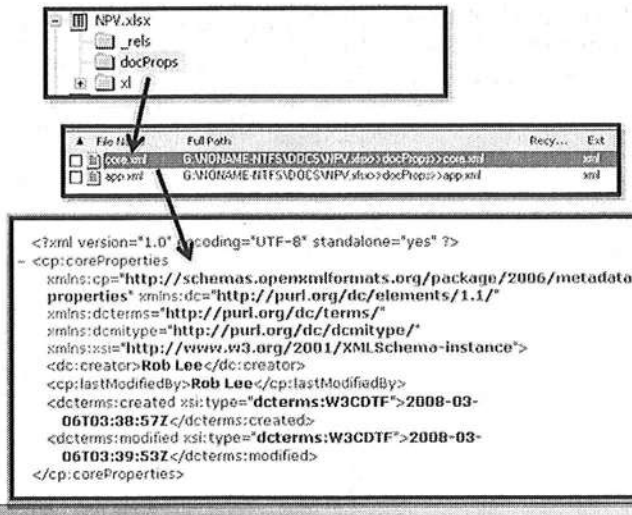
| Name | Value |
|---|---|
| Author | Nico |
| LastAuthor | Rob Lee |
| Created | 1/24/2008 6:41:47 PM |
| Last Saved | 2/7/2008 11:49:06 PM |
| AppName | Microsoft Excel |
| Doc Security | 0 |

Core Windows Forensics - SANS ©2011

This page intentionally left blank.

# Examining Office 2007 Files (1)
# (.docx, .pptx, .xlsx)

- Office 2007 Files are compressed archives
- XML Format
- View in FTK
- Summary Data is stored in a .xml file
  - **docProps/core.xml**




Core Windows Forensics - SANS ©2011

Starting with Microsoft Office 2007, a new format for Microsoft Office files was released. This format had an "x" extension on the back of the traditional three letter extension.(.DOC,.PPT,.XLS)->(.DOCX, .PPTX,.XLSX). The new files are merely compressed archives with embedded XML files.

Inside one of these XML compressed files you will see three directories. One of the directories is called docProps. It is inside this directory you're going to find a file called core.XML. Inside the file core.XML, you will see your XML formatted summary information and document properties for this particular file.

Notice in the lower right-hand corner, I have a new Excel 2007 spreadsheet. After examining the core.XML file inside docProps directory, you can see that I created the file on March 6, 2008 at 3:38 UTC. You can also notice the last modification time is about a minute later. Notice that the creator is Rob Lee and the last author is also Rob Lee. If someone else had modified the file other than Rob Lee, their name would show up instead.

# Examining Office 2007 Files (2)
# (.docx, .pptx, .xlsx)

- Office 2007 Files are compressed archives
- Parsing via EXIFTOOL

```
File Name                       : Forensic Sunnit EU_v_1.0.xlsx
Directory                       : C:/Documents and Settings/sansforensics/Deskto
p
File Size                       : 22 kB
File Modification Date/Time     : 2010:01:27 00:26:00-05:00
File Permissions                : rw-rw-rw-
File Type                       : XLSX
MIME Type                       : application/vnd.openxmlformats-officedocument.
spreadsheetml.sheet

Creator                         : Rob
Last Modified By                : Rob
Create Date                     : 2010:01:19 03:08:52Z
Modify Date                     : 2010:01:27 05:26:00Z
```

```
Title                           : Slide 1
Creator                         : RobLee
Last Modified By                : rob
Revision                        : 579
Last Printed                    : 2004-04-22T02:58:01Z
Create Date                     : 2001:11:20 21:11:13Z
Modify Date                     : 2009:06:13 20:42:49Z
```

```
Template                        : UrbanReport
Total Time                      : 38 minutes
Pages                           : 10
Words                           : 2801
Characters                      : 15962
Application                     : Microsoft Office Word
Doc Security                    : 0
Lines                           : 133
Paragraphs                      : 37
Scale Crop                      : No
Heading Pairs                   : Title, 1
Titles Of Parts                 :
Company                         :
Links Up To Date                : No
Characters With Spaces          : 18731
Shared Doc                      : No
Hyperlinks Changed              : No
App Version                     : 12.0000
```

Core Windows Forensics - SANS ©2011

In this example, we are utilizing the EXIFTOOL to parse the metadata of an Office 2007 Excel document. You can see that Rob Lee is the author and when the file was last printed, created, and last modified. You also see the file path embedded in the document which could help verify the location of the current file was when it was last saved. This could be an important piece of information if trying to identify drive letters and directories.

Tools like EXIFTOOL make parsing more advanced and new formats easily without trouble.

# PDF Metadata

- **Exiftool**
  - Extracts Basic Information
    - Author
    - Creation Date
    - Creator
    - Modification Date
    - Title

```
ExifTool Version Number          : 8.18
File Name                        : USBKEY-Guide.pdf
Directory                        : C:/Documents and Settings/sansforensics/Deskto
p
File Size                        : 235 kB
File Modification Date/Time      : 2009:12:13 17:16:58-05:00
File Permissions                 : rw-rw-rw-
File Type                        : PDF
MIME Type                        : application/pdf
PDF Version                      : 1.6
Encryption                       : Standard v4.4
Warning                          : Encryption algorithm 4 currently not supported
```

Core Windows Forensics - SANS ©2011

While Microsoft's Office products have a lot of metadata to examine<,> Adobe PDF files also have metadata…just not as much. There is not the graphical viewer inside of FTK to be able to examine the metadata that's embedded inside Adobe PDF files. EXIFTOOL also easily parses PDF metadata.

# Windows Artifact Analysis

- Link Files
- Microsoft Office Metadata
  - Documents
  - Spreadsheets
  - Powerpoint
  - Adobe PDF Files
- Media File Metadata (EXIF)
- Thumbnail Analysis
  - XP Thumbs.db
  - Vista/Win7 Thumbscache
- Internet Chat and Web E-mail Extraction
  - Internet Evidence Finder
  - Skype
- Recycle Bin
  - XP Recycle Bin
  - Vista/Win7 Recycle Bin
- Windows Prefetch

Core Windows Forensics - SANS ©2011

Now that we've looked at both Adobe PDFs and Microsoft Office products for the metadata that is stored inside those files. We also need to consider what other files contain internal data that might be useful during a forensic investigation. In fact most files actually have internal metadata stored inside of them. It is just a matter of finding a tool that will display that data for us.

Take, for example, MP3 files: inside an MP3 file you'll see metadata such as artist, album, track number, and genre. While this information might not be useful forensically, it is nevertheless, still stored inside the file's metadata, not inside the data of the actual song itself.

# Exiftool iPhone Picture Analysis

- iPhone Pictures
  - Internal GPS
  - Time and Date
  - Questions:
    - When was this picture taken?
    - Where was this picture taken?

In order to show the true capability of the EXIFTOOL, we will take a look at one of the most popular cell phones on the market, the Apple iPhone. With version 2 of the Apple iPhone, which has an embedded GPS device, every picture that is taken will be embedded with the GPS coordinates of the location where that picture was shot.

So the two key questions one would have with iPhone pictures would be: 1. When was this picture taken? 2. Where was this picture taken?

# iPhone Picture Exiftool Output

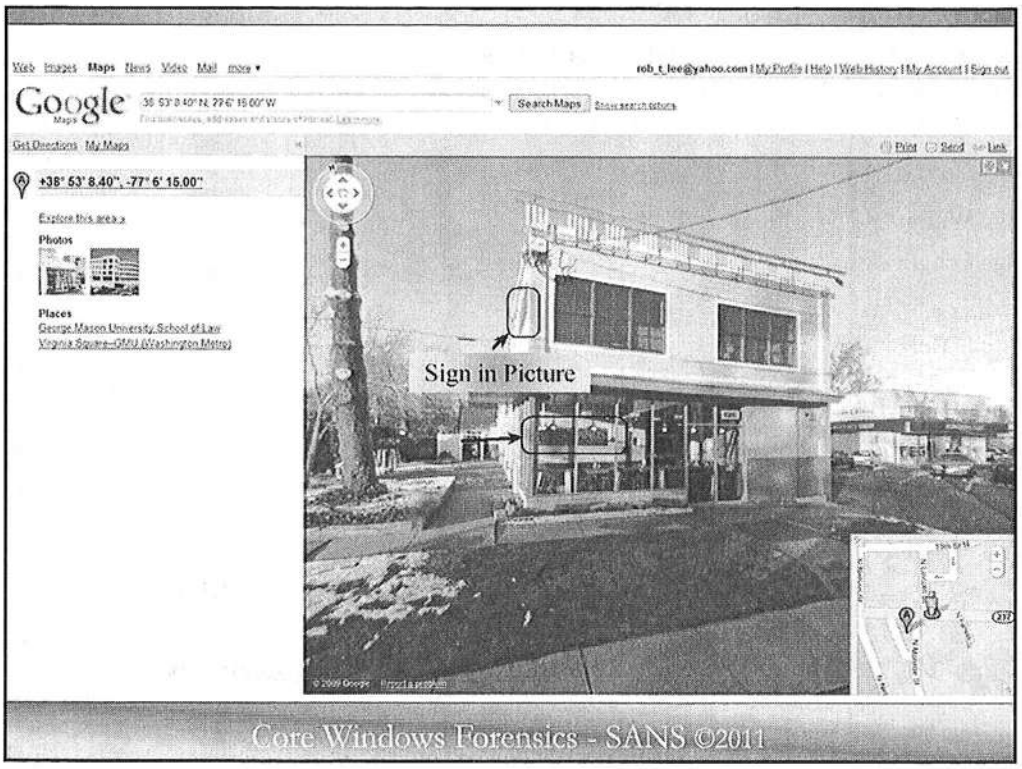
```
C:\Documents and Settings\sansforensics\Desktop>exiftool.exe IMG_0284.JPG
ExifTool Version Number         : 6.94
File Name                       : IMG_0284.JPG
Directory                       : .
File Size                       : 331 kB
File Modification Date/Time     : 2009:05:15 00:43:02
File Type                       : JPEG
MIME Type                       : image/jpeg
JFIF Version                    : 1.1
Make                            : Apple
Camera Model Name               : iPhone
X Resolution                    : 72
Y Resolution                    : 72
Resolution Unit                 : inches
Modify Date                     : 2009:05:15 00:43:02
F Number                        : 2.8
Date/Time Original              : 2009:05:14 19:45:42
Create Date                     : 2009:05:14 19:45:42
Color Space                     : sRGB
Exif Image Width                : 1600
Exif Image Length               : 1200
Gamma                           : 2.2
GPS Latitude Ref                : North
GPS Longitude Ref               : West
GPS Time Stamp                  : 19:45:38.21
Compression                     : JPEG (old-style)
Thumbnail Offset                : 640
Thumbnail Length                : 3559
Creator Tool                    : Microsoft Windows Photo Gallery 6.0.6001.18000

Creatortool                     : Microsoft Windows Photo Gallery 6.0.6001.18000

Software                        : Microsoft Windows Photo Gallery 6.0.6001.18000

Orientation                     : Horizontal (normal)
Image Width                     : 1600
Image Height                    : 1200
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:2 (2 1)
Aperture                        : 2.8
GPS Latitude                    : 38 deg 53' 8.40" N
GPS Longitude                   : 77 deg 6' 15.00" W
GPS Position                    : 38 deg 53' 8.40" N, 77 deg 6' 15.00" W
Image Size                      : 1600x1200
Thumbnail Image                 : (Binary data 3559 bytes, use -b option to extr
act)
```

Core Windows Forensics - SANS ©2011

This is the example output from an iPhone picture that was taken recently on my iPhone. Notice that you have the device make and manufacturer in addition to the creation date and time as well as the GPS coordinates embedded inside this picture. Let's take a look at where those GPS coordinates point to using and Google maps.

| | |
|---|---|
| ExifTool Version Number | : 6.94 |
| File Name | : IMG_0284.JPG |
| Directory | : . |
| File Size | : 331 kB |
| File Modification Date/Time | : 2009:05:15 00:43:02 |
| File Type | : JPEG |
| MIME Type | : image/jpeg |
| JFIF Version | : 1.1 |
| Make | : Apple |
| Camera Model Name | : iPhone |
| X Resolution | : 72 |
| Y Resolution | : 72 |
| Resolution Unit | : inches |
| Modify Date | : 2009:05:15 00:43:02 |

| | |
|---|---|
| F Number | : 2.8 |
| Date/Time Original | : 2009:05:14 19:45:42 |
| Create Date | : 2009:05:14 19:45:42 |
| Color Space | : sRGB |
| Exif Image Width | : 1600 |
| Exif Image Length | : 1200 |
| Gamma | : 2.2 |
| GPS Latitude Ref | : North |
| GPS Longitude Ref | : West |
| GPS Time Stamp | : 19:45:38.21 |
| Compression | : JPEG (old-style) |
| Thumbnail Offset | : 640 |
| Thumbnail Length | : 3559 |
| Creator Tool | : Microsoft Windows Photo Gallery 6.0.6001.18000 |
| Creatortool | : Microsoft Windows Photo Gallery 6.0.6001.18000 |
| Software | : Microsoft Windows Photo Gallery 6.0.6001.18000 |
| Orientation | : Horizontal (normal) |
| Image Width | : 1600 |
| Image Height | : 1200 |
| Encoding Process | : Baseline DCT, Huffman coding |
| Bits Per Sample | : 8 |
| Color Components | : 3 |
| Y Cb Cr Sub Sampling | : YCbCr4:2:2 (2 1) |
| Aperture | : 2.8 |
| GPS Latitude | : 38 deg 53' 8.40" N |
| GPS Longitude | : 77 deg 6' 15.00" W |
| GPS Position | : 38 deg 53' 8.40" N, 77 deg 6' 15.00" W |
| Image Size | : 1600x1200 |
| Thumbnail Image | : (Binary data 3559 bytes, use -b option to extract) |




Plugging the GPS coordinates into Google maps is a fairly simple process. I switched from map view to Google Street view, so I could tell what I'm looking at. I rotated the viewing camera to the left and I can see the location where the picture was taken. On the picture we saw a sign called Rocklands barbecue. Notice on the side of the building you actually see the same sign. The picture was actually taken next to the tree that you see in the lower left-hand side of the picture.

As you can tell, being able to use the exiftool to be able to mine the metadata, including the GPS coordinates out of a picture, could prove fairly useful during the computer investigation.

# Windows Artifact Analysis

- Link Files
- Microsoft Office Metadata
  - Documents
  - Spreadsheets
  - Powerpoint
  - Adobe PDF Files
- Media File Metadata (EXIF)
- Thumbnail Analysis
  - XP Thumbs.db
  - Vista/Win7 Thumbscache
- Internet Chat and Web E-mail Extraction
  - Internet Evidence Finder
  - Skype
- Recycle Bin
  - XP Recycle Bin
  - Vista/Win7 Recycle Bin
- Windows Prefetch



Most people don't know this but Microsoft Windows operating systems actually record the thumbnails of pictures when they are viewed inside the operating system. Windows XP and Windows Vista are similar in that they both store thumbnail pictures, but their implementations are completely different.

In Windows XP they use a file called thumbs.db, which will be located in any directory that contains pictures that were viewed in Windows Explorer.exe. In Windows Vista/Win7 they are stored at a single location that contains all the thumbnails are viewed on entire system for that specific user.

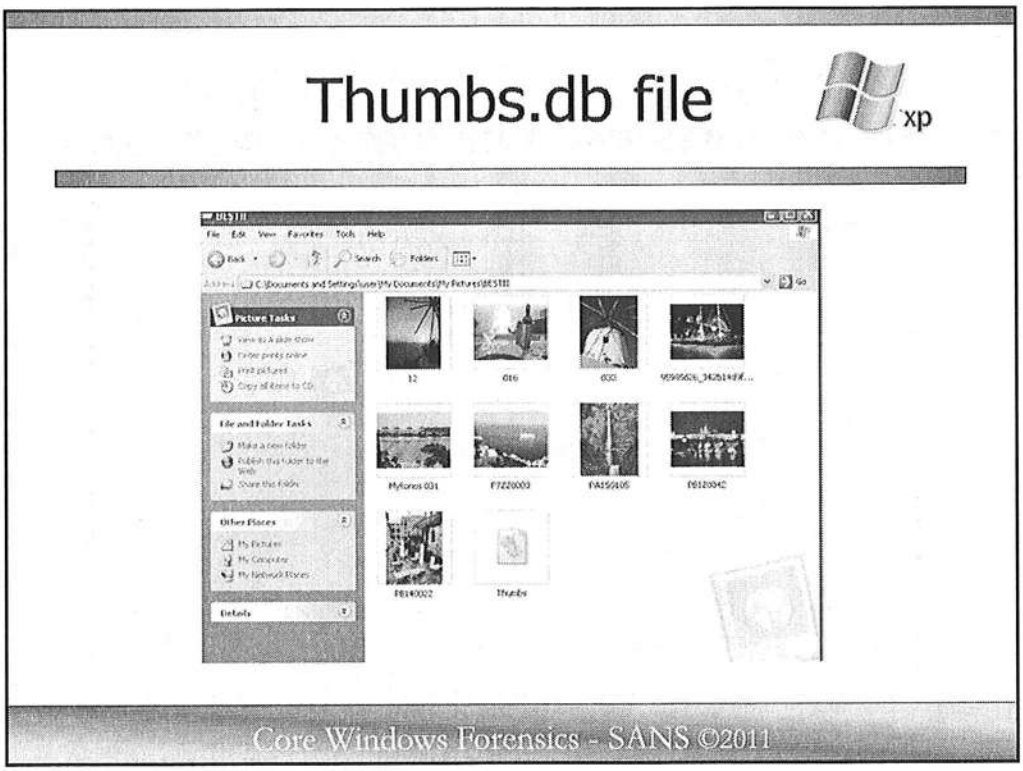
# Thumbnail Forensics
## thumbs.db

- **thumbs.db**
  - Hidden file in directory where pictures on Windows XP machine exist
- Catalogs all the pictures and stores a copy of the thumbnail even if the pictures were deleted
  - Include:
    - Thumbnail Picture of Original
    - Last Modification Time
    - Original Filename

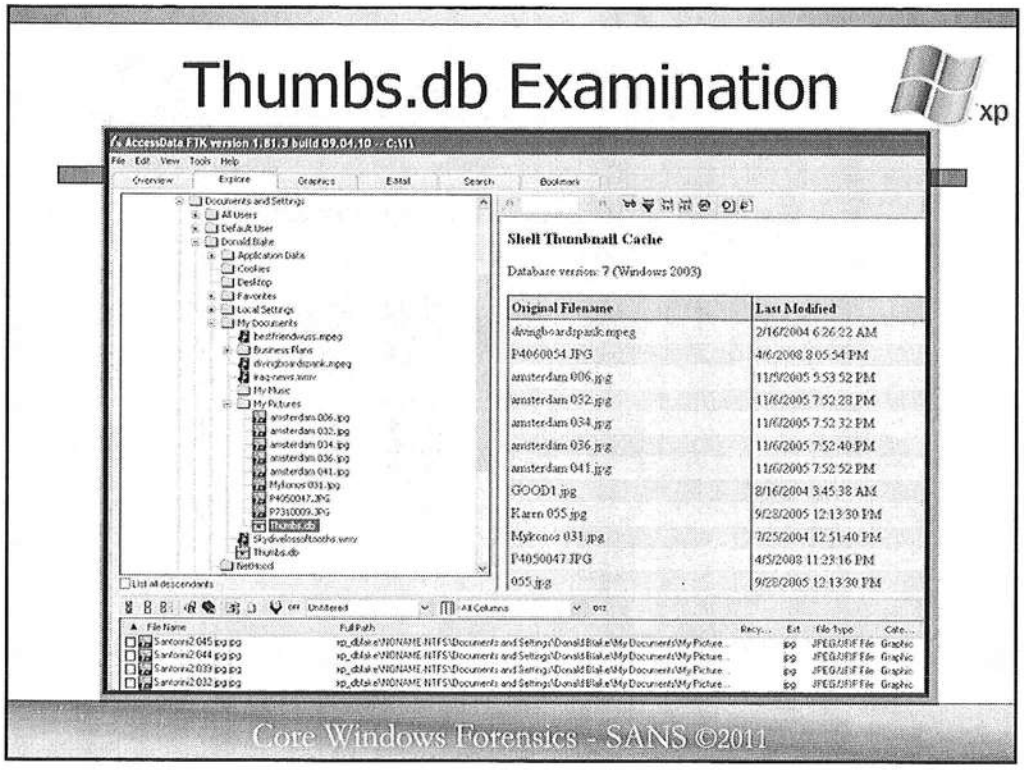


Thumbs.db is a hidden file in each directory where pictures on Windows XP machine exist. It catalogs all the pictures and stores a copy of the thumbnail even if the pictures were deleted. It's useful to see if original pictures once existed in a directory. It is only created if the files are viewed in thumbnail or filmstrip mode.

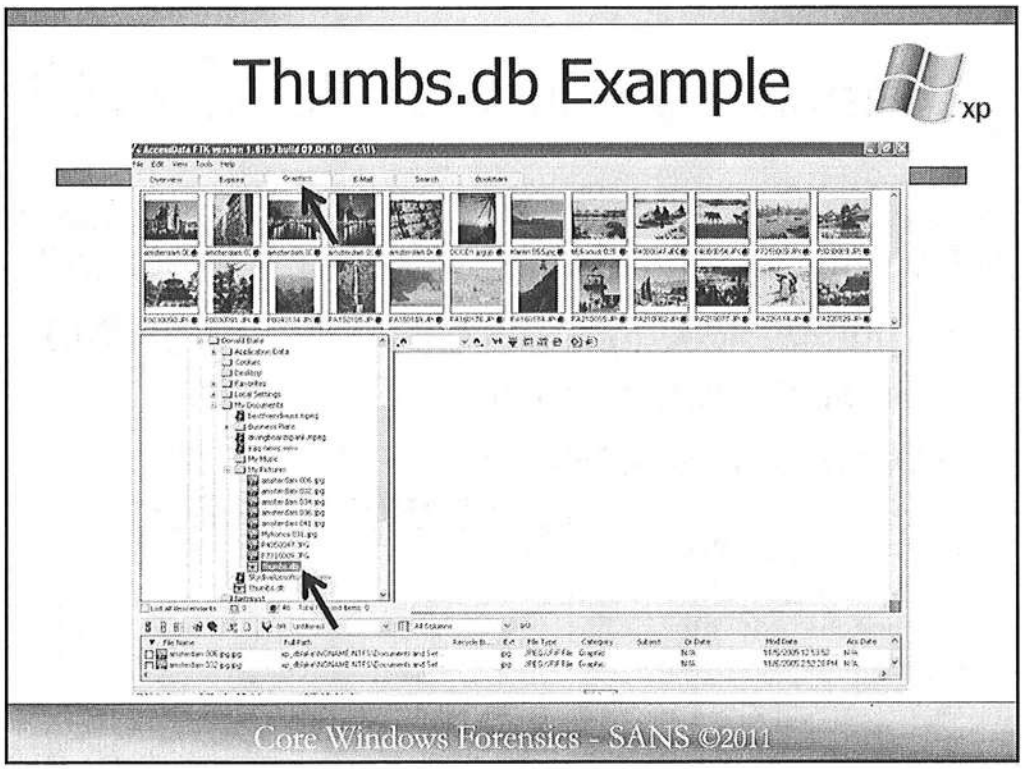
Essentially, what you tell with the thumbs.DB file is that it will list all the files, both deleted and existing, that had resided inside this directory. While you might only see a few pictures in a directory, the thumbs.DB file will show us all the pictures that had once existed inside this directory. The thumbs.DB file has proved invaluable for child exploitation cases.




This is the location of the Thumbs.db file on an XP machine. Notice the pictures that can be seen as we are in thumbnail view. The pictures that we are viewing are actually stored in the file called Thumbs.db, also shown in the same directory when you view all the files in that directory (including hidden files). Thumbs.db collects the original filename, a thumbnail view of the picture, and relevant time and date stamps as a part of the data of the file. If you delete the pictures from the directory, but fail to remove the thumbs.db, the contents can still be recovered by forensically analyzing the previously stored data in the thumbs.db file.




Using FTK, parsing the contents of a thumbs.DB file is easy work. Simply, browse to a directory that contains a thumbs.DB file, and select it. On the right-hand side in the viewer, you will see any a summary of all the picture files that existed inside this directory. Inside the file viewing pane on the bottom of the screen, you will see the actual JPEG files that are contained inside the thumbs.DB file.




If you select "graphics mode" and still select the thumbs.DB file, across the top you will see all the JPEG's. This makes it much easier to quickly ascertain what the pictures were by scanning the actual thumbnails that are contained inside the thumbs.DB file.

As you could see, examining the contents of the thumbs.DB file is relatively painless. Finding where the thumbs.DB file is on the filesystem could be a challenge. FTK makes it easy for you to do this through its interface.

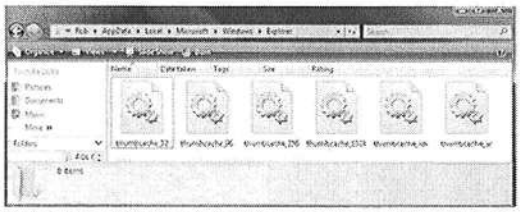
Mark's tool and FTK 1.81.6 will not parse Win7 thumbcaches (neither will Encase 6.15.0.82). FTK 3.1.2 will parse Win7 Thumbcaches.

## Vista/Win7 Thumbnail Forensics

C:\Users\<username>\AppData\Local\Microsoft\Windows\Explorer\

- Thumbs.db no longer exists in Vista/Win7.
  - Thumbnails only (Small, Medium, Large, XL)
  - Original location "might" be stored
  - Date/Time is not stored

- Thumbcache_32.db
- Thumbcache_96.db
- Thumbcache_256.db
- Thumbcache_1024.db



Core Windows Forensics - SANS ©2011

On Vista/Win7 versions of Windows, Thumbs.db does not exist. The data now sits under a single directory for each user of the machine located in their application data directory under their home directory. It is almost if Microsoft was trying to make our lives easier when we performing Vista/Win7 based forensics. No more searching for thumbs.db files!

There is now one directory stores all the thumbnails in files called a thumbcache.

The location of the thumbcache files is:

**C:\Users\<*username*>\AppData\Local\Microsoft\Windows\Explorer\.**

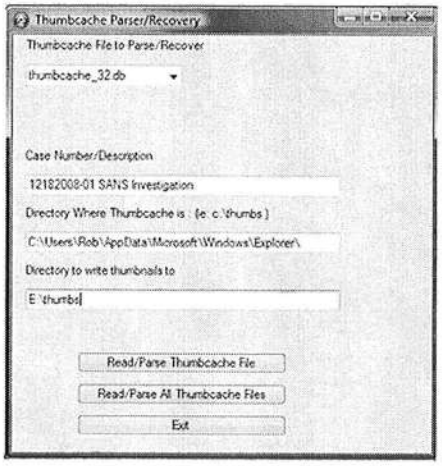
These are created when a user switches a folder to thumbnail mode or views pictures via a slide show. As it were, our thumbs are now stored in separate database files. Vista/Win7 has 4 sizes for thumbnails and the files in the cache folder reflect this.

| | |
|---|---|
| 32 | -> small |
| 96 | -> medium |
| 256 | -> large |
| 1024 | ->extra large |

The Thumbcache will store the thumbnail copy of the picture based on the thumbnail size in the content of the equivalent database file.

Parsing these files is relatively painless using the **Thumbcache Parser** written by Mark McKinnon. The Parser is able to load up the database cache through the GUI interface rather easily. You can choose to examine a specific size cache (**32,96,256,1024**) via the options or you can analyze all the cache databases at once. The fact that the cache for thumbnails is in one folder makes examining a user's thumbnail viewing activities much easier than on an XP machine.

# Thumbcache Parser/Recovery



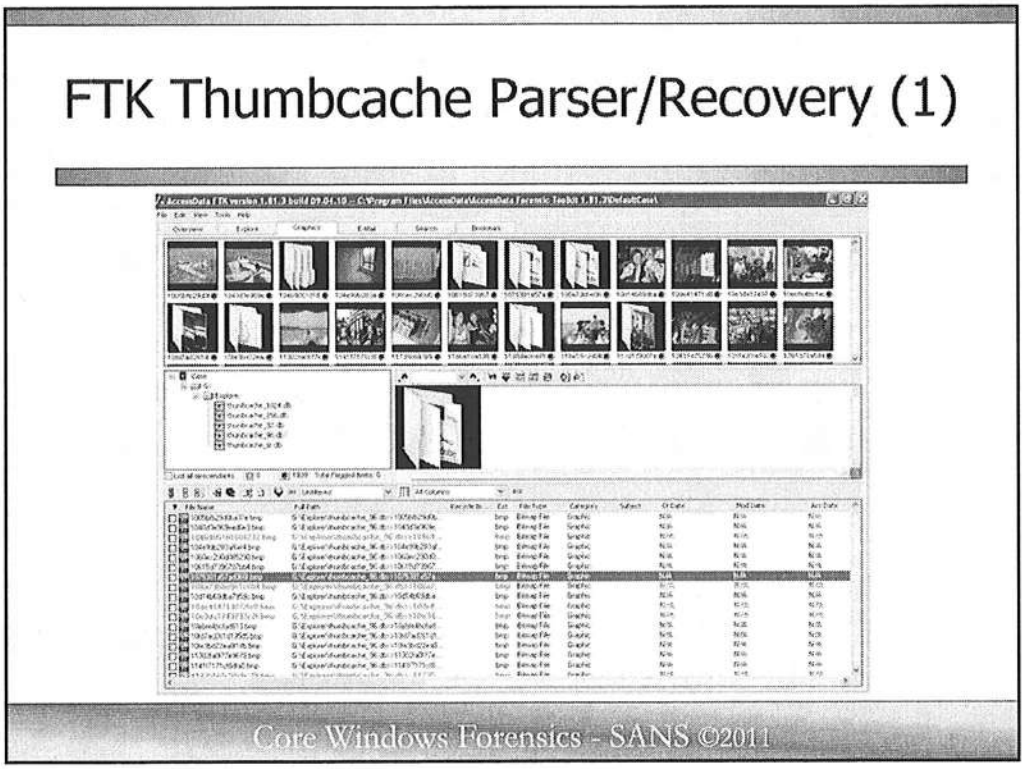

- FTK Will Parse Explorer Directory and all four Cache Files
- Thumbcache Parser (on Desktop)
  - By Mark Mckinnon
  - Will create a report utilizing
  - Case Number
  - Thumbcache Directory
  - Output Directory
- Even able to put your own logo on the output report



The Thumbcache Parser located on your desktop of your XP SIFT Workstation allows you to carve the contents of the Thumbcache directory on a Vista machine easily. The tool is configurable to allow you to create output reports with evidence tags, case name, and even your organization logo.

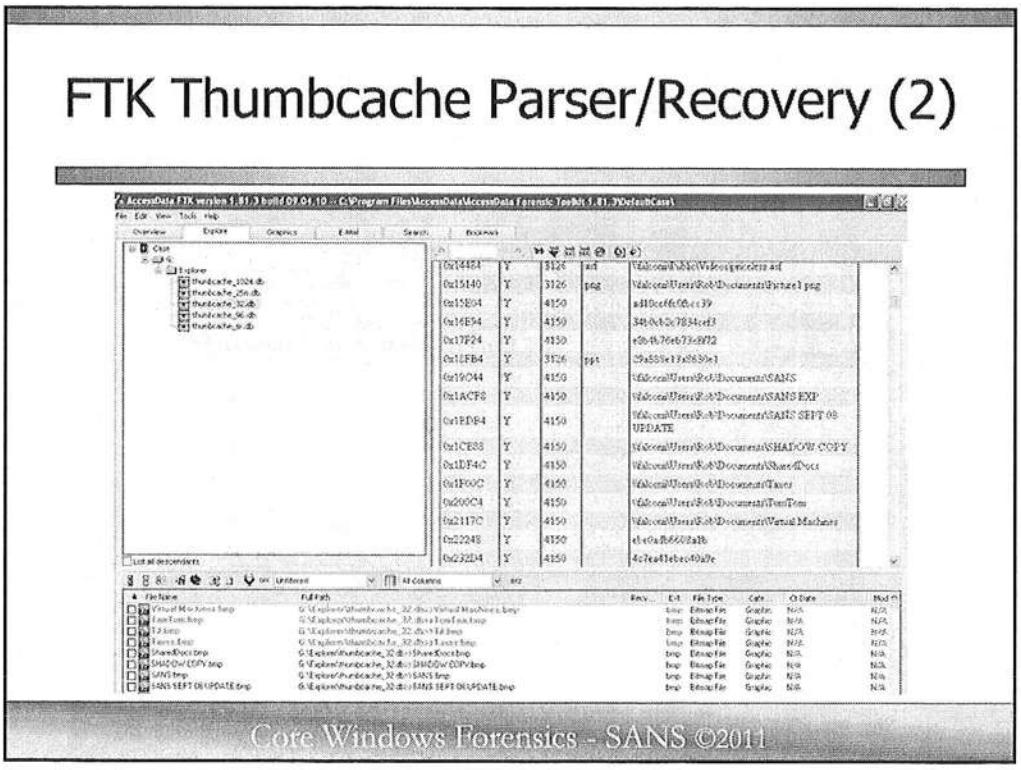
You can have Thumbcache Parser parse one of the specific Thumbcache_*.db files or them all and write it to an output directory.

This will enable you in your case to quickly parse the thumbnails database files on a Vista machine.




FTK will also parse Vista/Win7 thumbscache thumbcache files. Again<,> simply browse to the directory in containing the Vista/Win7 files and select them in graphics mode. This makes easy work of examining all the thumbnail pictures that existed for this specific user on this Windows Vista/Win7 operating system.

Notice that the original modification time and date stamps are not included with the thumbcache entries.



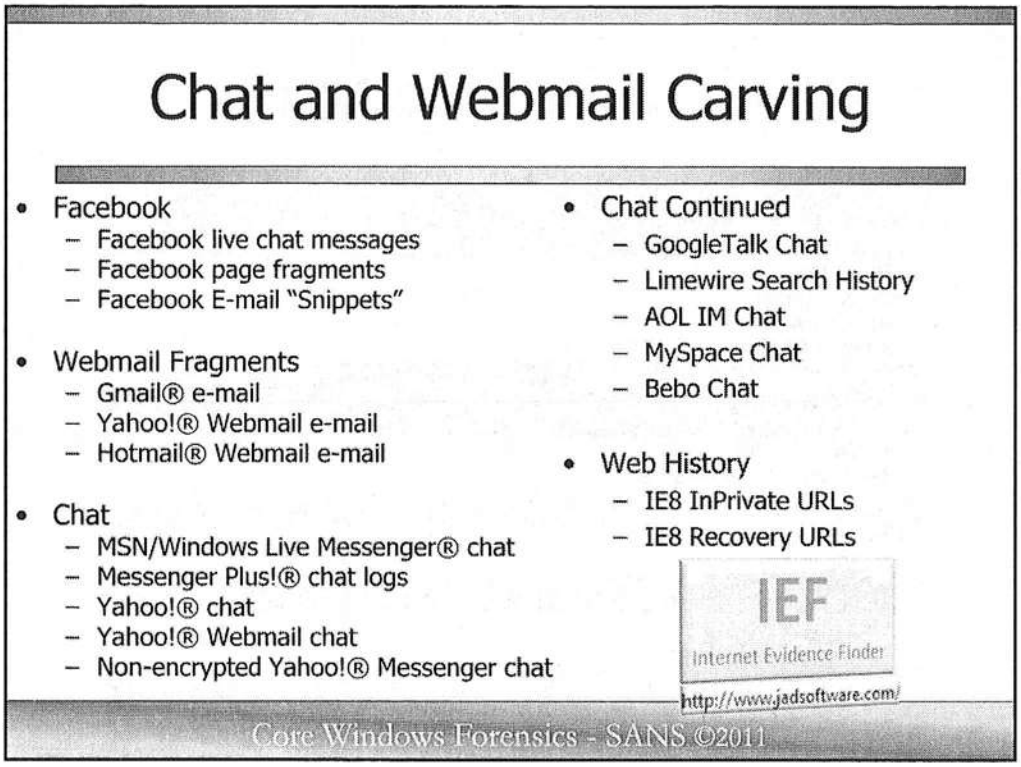

In Explorer mode, you can easily identify the summary information by clicking on one of the files that contains the thumbnails. The drawback to this is that with a single summary file there will be many entries. While making it easier for investigators to find thumbnails on a Windows Vista/Win7 operating system, it adds to the frustration by having so many files to examine at a single location. However, I found this more useful than trying to find thumbs.DB files all over the system.

# Windows Artifact Analysis

- Link Files
- Microsoft Office Metadata
  - Documents
  - Spreadsheets
  - Powerpoint
  - Adobe PDF Files
- Media File Metadata (EXIF)
- Thumbnail Analysis
  - XP Thumbs.db
  - Vista/Win7 Thumbscache
- Internet Chat Programs
  - Internet Evidence Finder
  - Skype
- Recycle Bin
  - XP Recycle Bin
  - Vista/Win7 Recycle Bin
- Windows Prefetch



Let us now take a look at one of the dominant Internet Chat Programs, Skype.




FROM THE AUTHOR'S WEBSITE: http://www.jadsoftware.com/

---

The Internet Evidence Finder (IEF) searches the selected drive, folder (and sub-folders, optionally), or file (memory dumps, pagefile.sys, hiberfil.sys, etc) for Internet artifacts. IEF can be downloaded from http://www.jadsoftware.com/home/ief.htm

Depending on the items selected, IEF creates a report containing the search results or creates individual files containing the data found.

Web History
IE8 InPrivate/Recovery URLs

*Facebook® Chat Messages*
If this item is checked, IEF will search for messages sent and received using the Facebook® live chat feature. Facebook® Live Chat can be found in live memory, pagefile.sys/hiberfil.sys files, and on the hard drive in allocated and unallocated space. Information found with the message can include the Facebook® profile ID used to send/receive the message, the from/to names and ID's, and the date/time (in UTC) that the message was sent. However, not all messages found include all this data. An HTML file is also saved to the 'Facebook Live Chat' folder to assist in looking up Facebook ID's. Located messages are exported into a CSV or TSV file format in the 'Facebook Live Chat Report' folder.

*Facebook® Page Fragments*
If this item is checked, IEF will search for any Facebook® related web pages, including but not limited to the Inbox page, e-mails, photo galleries, groups, and so on. Most recovered items will be fragments and not the complete page, but attempts are made to recover the entire page and filter out false positives. A header is added to the fragment to aid in viewing the page. Facebook® page fragments can be found in live memory, pagefile.sys/hiberfil.sys files, and on the

hard drive. Items found in this category are exported to files in the 'Facebook pages' folder with an .htm extension. An 'index.htm' file is also created in that folder; it lists source / output files along with hyperlinks to the output files.

*MSN®/Windows Live Messenger Chat Messages*
If this item is checked, IEF will search for chat messages sent/received using Windows Live Messenger®. Windows Live Messenger/MSN chat log files and chat fragments are found in live memory dumps, the pagefile.sys/hiberfil.sys files, and on the hard drive in allocated and unallocated space. Information found with the messages varies. If a Windows Live Messenger chat log file is found, a MessageLog.xsl file is created to aid in viewing the log file(s). Located messages are exported into text files (MSN protocol fragments), HTML files (incomplete logs), or XML files (complete logs) in the 'Windows Live Messenger chat' folder. An 'index.htm' file is also created in that folder; it lists source / output files along with hyperlinks to the output files. An option is available that will save all located chat log messages into one CSV or TSV report file. (Note: The Windows Live Messenger® search option is backwards compatible with MSN Messenger®, and these two program names are used interchangeably in IEF.)

*Yahoo!® Chat Messages*
If this item is checked, IEF will search for chat messages sent and received using Yahoo!® Messenger. This search option will recover chat logs found in live memory dumps, the pagefile.sys/hiberfil.sys files and allocated/unallocated space on a hard drive. The Yahoo!® Messenger local username must be provided to decrypt the messages (e.g. if the login e-mail address used is chippy@yahoo.com, the username would usually be 'chippy'). The remote username is not stored in the logs and is not recoverable. If multiple accounts are used on the same computer, you can provide IEF with all the usernames and a report file will be created for each one. A number of false positives are unavoidable, especially when multiple account logs are on the system as there is no way to determine if a log was decrypted successfully or not. Located chat messages are exported into a CSV or TSV file in the 'Yahoo chat' folder.

*GoogleTalk® Chat Messages*
If this item is checked, IEF will search for messages sent or received using GoogleTalk® live chat. These messages are left behind in live memory, pagefile.sys/hiberfil.sys files, and possibly on the hard drive. Information found with the message can include the message ID, the Sender/Recipient e-mail addresses, and the sender/recipient's ID. Dates and times are not available to recover at this time. This search option may also recover chat left behind from other chat programs that utilize the 'Jabber' chat protocol (the sender/recipient ID will be your clue, containing an abbreviated name of the client used by that person). Located messages are exported into a CSV or TSV file format in the 'GoogleTalk' folder.

*Yahoo!® Webmail Chat Messages*
If this item is checked, IEF will search for messages sent or received using the live webmail chat found in Yahoo!® Webmail. These messages are left behind in live memory, pagefile.sys/hiberfil.sys files, and possibly on the hard drive. Information found with the message can include the Status number, the version number and vendor ID, the session ID, and the Sender/Recipient usernames. Dates and times are not available to recover at this time. Located messages are exported into a CSV or TSV file format in the 'Yahoo Webmail Chat' folder.

*Gmail® E-mail*
If this item is checked, IEF will search for Gmail® e-mail fragments left behind in live memory. The fragments may be also found in the pagefile.sys/hiberfil.sys files and possibly the hard drive. Information found will vary and no proper format has been determined at this time. IEF will do its best to clean up the located fragment and convert encodings into a more readable format. Some fragments will be of the folder view with the sender name/address, subject, and first segment of the body of the e-mail. Located Gmail® fragments are exported into a text file format (.txt) in the 'Gmail' folder. An 'index.htm' file is also created in that folder; it lists source / output files along with hyperlinks to the output files.

*Limewire® Search History*
If this item is checked, IEF will look for search keywords left behind in live memory by Limewire® (tested with Limewire® v5.2.8 – 5.5.8). They are also found in the pagefile.sys/hiberfil.sys files and possibly unallocated space on the hard drive. Search keywords/terms that are recovered have an associated number indicating how many search

results were returned for that search term at the time the keyword was left in memory. The recovered search terms are search keywords that were entered by the local user. Other search keywords that were passed through the client ("Incoming Searches") from other clients on the P2P network are not recovered. Located search keywords are exported into a CSV or TSV file format in the 'Limewire Search History' folder.

*Limewire.props files*
If this item is checked, IEF will look for fragments of Limewire.props files. Fragments of these files are found in live memory, the pagefile.sys/hiberfil.sys files and on the hard drive in allocated and unallocated space. These files contain configuration data for the Limewire® peer to peer file sharing client and can include geolocations, recent downloads, and many other useful items. Located file fragments are exported into .txt (text) files in the 'Limewire.props files' folder. An 'index.htm' file is also created in that folder; it lists source / output files along with hyperlinks to the output files.

*IE8 InPrivate/Recovery URLs*
If this item is checked, IEF will look for URLs visited during InPrivate browsing in IE8 and URLs saved in Internet Explorer recovery files. These URLs are left behind in live memory, the pagefile.sys/hiberfil.sys files and on the hard drive in unallocated space. At this time, there is no known method of distinguishing between these two types of URL artifacts. Also found with the URLs is a page title or description, but this is not always present. Located URLs and titles/descriptions are exported into a CSV or TSV file format in the 'IE8 InPrivate and Recovery URLs' folder.

*Yahoo!® Messenger Group Chat*
If this item is checked, IEF will look for messages sent or received in Yahoo!® Messenger Group chat rooms. These chat messages are left behind in live memory, the pagefile.sys/hiberfil.sys files and possibly on the hard drive. Information found within these fragments can include the date/time, the username that sent the message, and the message itself. The name of the Yahoo! Messenger group that the message is sent within is not available to recover. Located messages are exported into a CSV or TSV file format in the 'Yahoo Messenger Group Chat' folder.

*Yahoo!® Webmail e-mail*
If this item is checked, IEF will search for Yahoo!® webmail left behind in live memory, pagefile.sys/hiberfil.sys files, and on the hard drive in allocated and unallocated space. Multiple types of Yahoo!® webmail interfaces are supported, including 'Classic view' and the New Yahoo! Webmail view. Recovered e-mail messages, e-mail compose pages, and folder views are located and saved to .htm (HTML) files in the 'Yahool webmail' folder. An 'index.htm' file is also created in that folder; it lists source / output files along with hyperlinks to the output files.

*Hotmail® Webmail e-mail*
If this item is checked, IEF will search for Hotmail® webmail left behind in live memory, pagefile.sys/hiberfil.sys files, and on the hard drive in allocated and unallocated space. Recovered e-mail messages, contact listings, and folder views are located and saved to .htm (HTML) and .txt (text) files in the 'Hotmail webmail' folder. An 'index.htm' file is also created in that folder; it lists source / output files along with hyperlinks to the output files.

*AOL® Instant Messenger chat logs*
If this item is checked, IEF will search for AOL® Instant Messenger (AIM) chat logs. These logs are left behind in live memory, pagefile.sys/hiberfil.sys files, and on the hard drive in allocated and unallocated space. Each log recovered is saved to an individual HTML (.htm) file. A 'styles.css' style sheet file is created to aid in viewing these logs. Located logs are saved in the 'AIM chat' folder. An 'index.htm' file is also created in that folder; it lists source / output files along with hyperlinks to the output files.

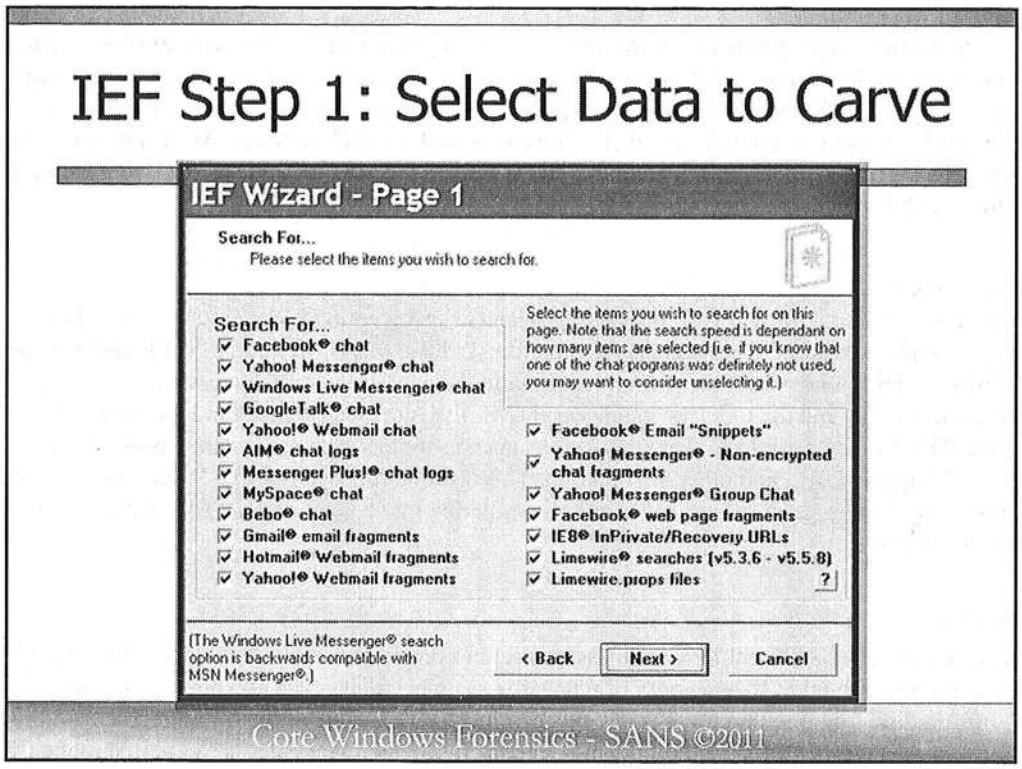
*Messenger Plus!® chat logs*
If this item is checked, IEF will search for Messenger Plus!® chat logs. Messenger Plus! is an add-on for Windows Live Messenger/MSN Messenger that adds a number of features to the chat program. These chat logs are left behind in live memory, pagefile.sys/hiberfil.sys files, and on the hard drive in allocated and unallocated space. Located chat logs are saved into individual HTML (.htm) files in the 'Messenger Plus! chat' folder. An 'index.htm' file is also created in that folder; it lists source / output files along with hyperlinks to the output files.

*MySpace® chat*
If this item is checked, IEF will look for messages sent or received in MySpace® live chat. These chat messages are left behind in live memory, the pagefile.sys/hiberfil.sys files and possibly on the hard drive. Information found within these fragments can include the status of the message, the date/time, the sender ID, target ID, and the message itself. An HTML file is created to assist in looking up the MySpace ID's. Some user info is also recoverable, such as the real name/username associated to a MySpace ID, image URL, and other information. This information is saved to a 'User Info' file in the 'MySpace chat' folder. Located messages are also exported into a CSV or TSV file format in the 'MySpace chat' folder.

*Bebo® chat*
If this item is checked, IEF will look for messages sent or received in Bebo® live chat. These chat messages are left behind in live memory, the pagefile.sys/hiberfil.sys files and possibly on the hard drive. Information found within these fragments can include the status of the message, the date/time, the sender username, target username, and the message itself. Located messages are exported into a CSV or TSV file format in the 'Bebo chat' folder.

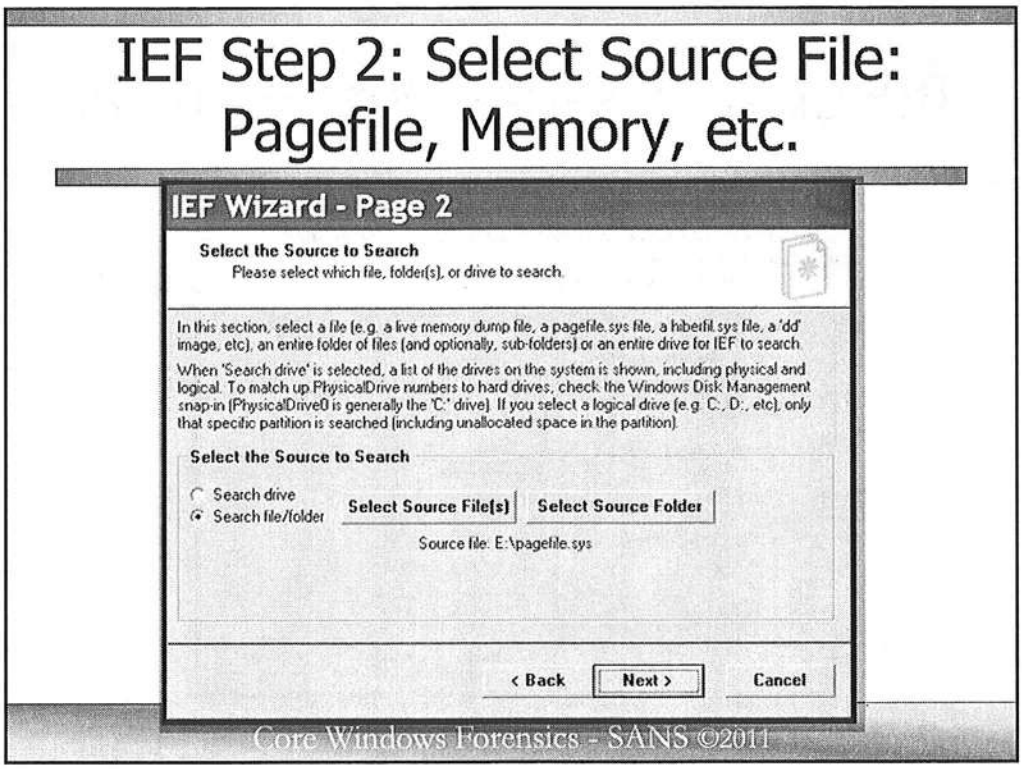
*Non-encrypted Yahoo!® Messenger chat*
If this item is checked, IEF will search for non-encrypted chat messages left behind by Yahoo!® Messenger. These messages are artifacts from the actual Yahoo!® Messenger chat window and are found in memory dumps and the pagefile/hiberfil.sys files. No username(s) are required to recover these messages. Messages of this type include the sending user name, the date/time (local time, not UTC), and the message itself. The recipient is not found in these fragments but can usually be ascertained by viewing the chat conversation. Located messages are saved to a CSV or TSV file format in the 'Yahoo chat – Non-encrypted' folder.

*Facebook® E-mail "Snippets"*
If this item is checked, IEF will search for Facebook® e-mail "snippets" (previews of a full e-mail message) left behind in live memory, pagefile.sys/hiberfil.sys files, and on the hard drive in unallocated space. This artifact is left behind when a user is viewing their Inbox or Sent Messages folder in their Facebook® account. It can include the Subject line, Original Author user ID, Recent Authors user IDs (the participants of the e-mail conversation), Time Last Updated (the last time a message was posted in the thread), thread ID (ID# of the message in the user's mailbox), and the "snippet" itself. Located "snippets" are exported into a CSV or TSV file in the 'Facebook Email Snippets' folder.



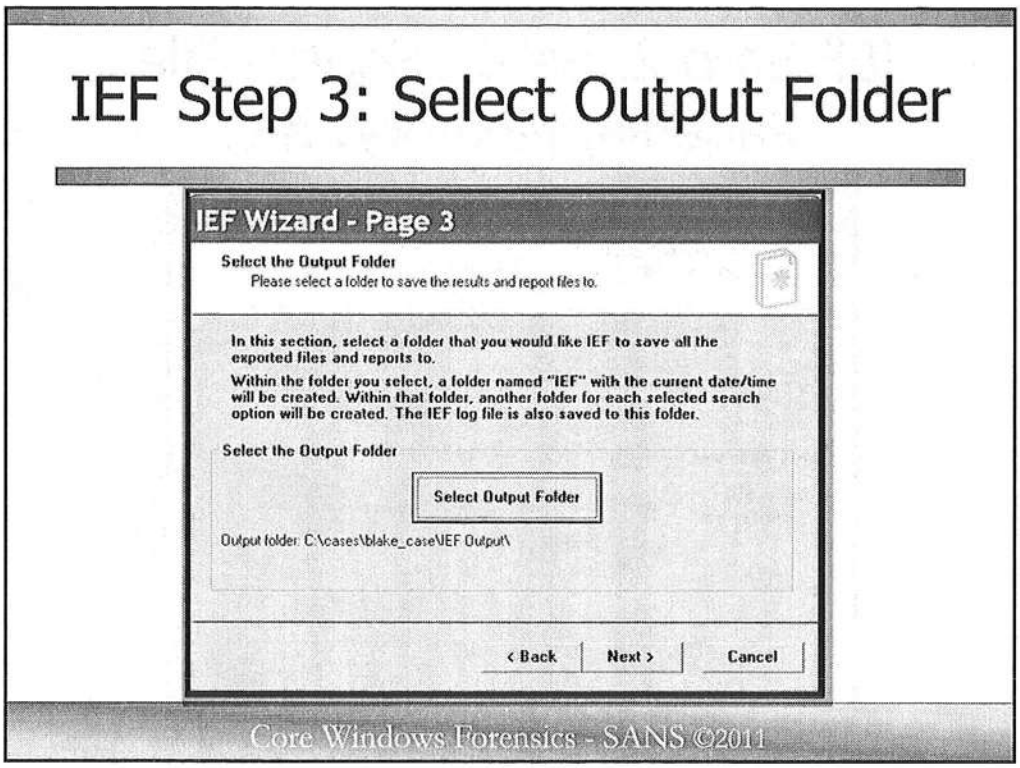

Obtain Key from instructor for IEF full version that will be good for 21 days. Once activated, start the wizard and select all fragments.

Usually you should start by selecting all of the fragments that you can. This will allow you to target as many data pieces as possible that you can recover. Remember this will not place context surrounding the items, only that the items exist. It may requrie you to find these items using a string search later on to determine which files or processes they were originally in.

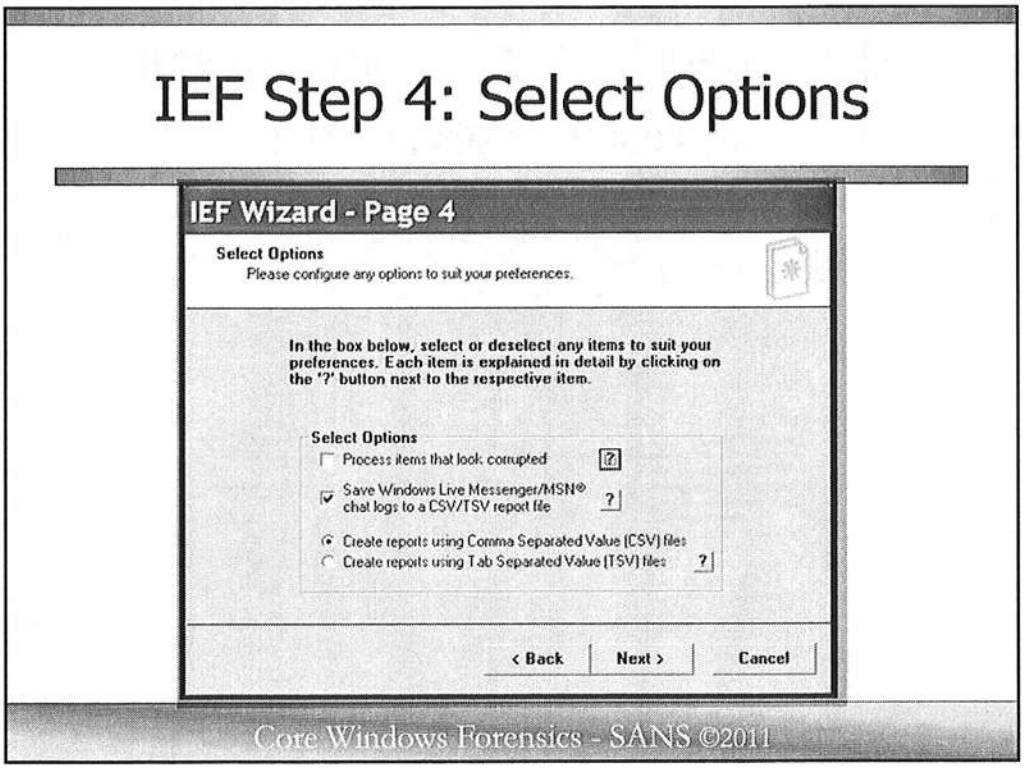



Select the source file. Typically, users like using the IEF tool on memory or the pagefile, as that could carve out some pretty interesting fragments using this tool. The tool can also be used against unallocated space or slack (recover and export via FTK Imager) or against an entire file system image file.

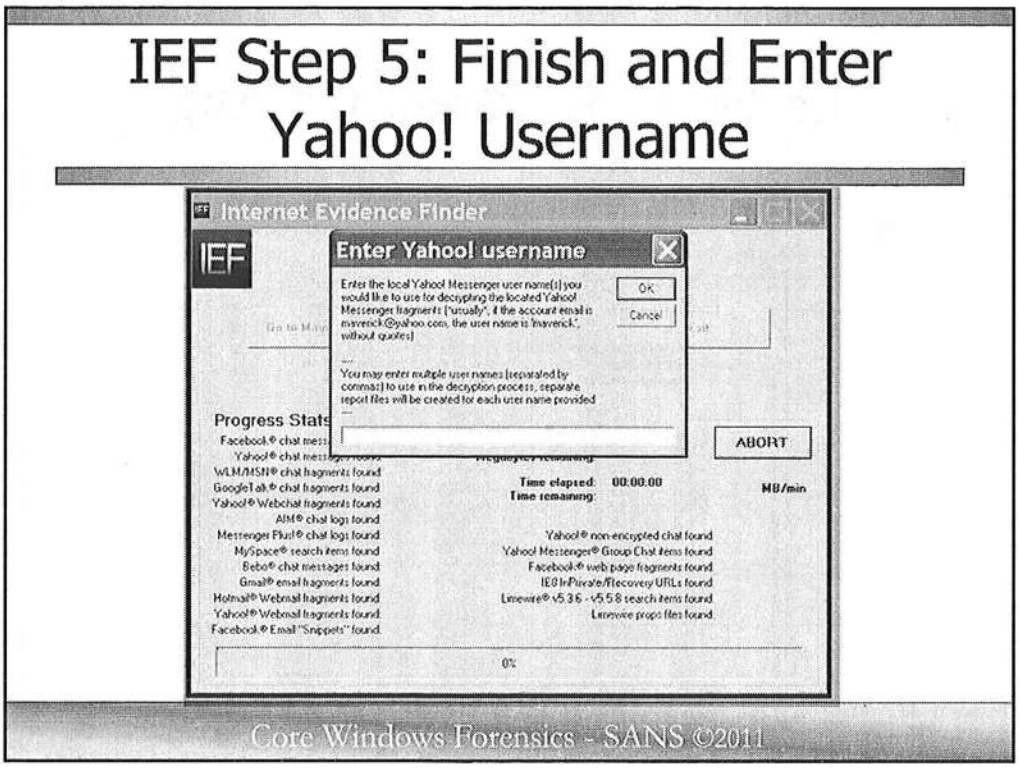
Memory and the pagefile will have many items that are not ever saved to the hard drive such as IE8 InPrivate browsing artifacts or chat sessions. These locations can be extremely useful to your case since many criminals think that this information is not saved anywhere and they feel more comfortable "speaking freely" than they would if they felt they were being monitored. Lucky for the digital forensic investigator that these items are not cleared when the program exits and can be discovered using capabilities such as Internet Evidence Finder.




Select your output folder where your carved items will reside once the tool is completed.

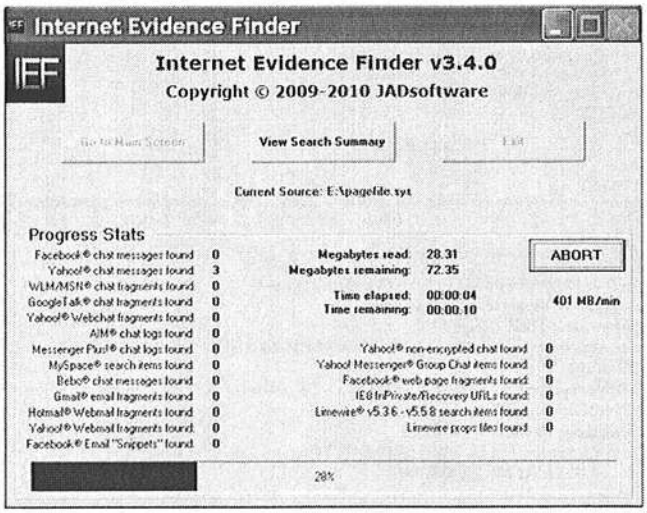



Finally, select the options you would like to use with IEF. Generally, the default options work the best.

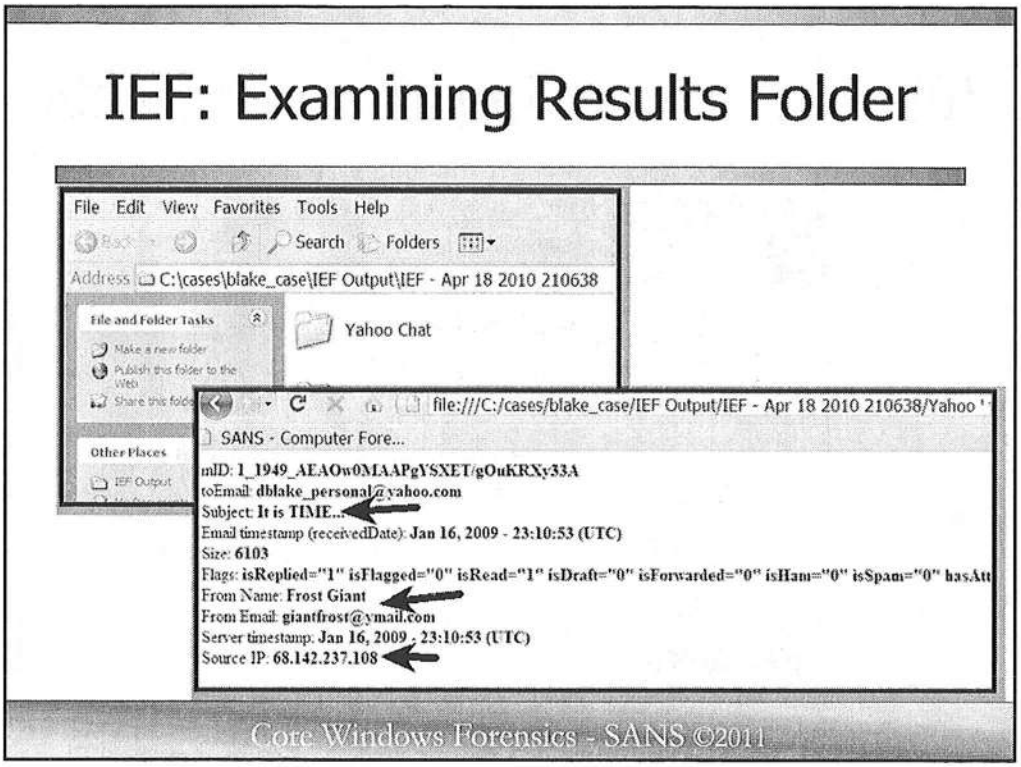



Once you complete the options and IEF is about to run, in order to examine the Yahoo! Messenger Chat fragments, you will need to input the user's e-mail address you want to decrypt. Note that you can use more than one Yahoo e-mail address, seperated by a comma.

# IEF Step 6: Carving the Fragments




IEF might take awhile to run depending on the size of your image. It will give you a status of every fragment that it was able to find and recover from the system.




The results folder will show you the artifacts that IEF discovered in this example, IEF found that there were several e-mails from Frost Giant on Jan 16, 2009 23:10:53 UTC (6:10:53 PM EST). This corresponds to an e-mail that alerts Donald Blake to the fact that he was going to be fired on Jan 19, 2009.

If you perform a keyword search for the giantfrost@ymail.com e-mail address, you should be able to find the XML file and import that into an XML viewer to examine the contents.

One trick to use: When you find the e-mail contents inside the XML file, notice that it is written in HTML. You can cut away the script so it leaves just the HTML so that it looks like this:

<html><head><style type="text/css"><!--DIV {margin:0px;}--></style></head><body> Donald,   I hate to bring bad news to you.  But I learned at corporate today that they found out about the "you-know-what" activity in the accounts.  They saw your name all over it.  I know you were directed to do this by "he-who-shall-not-be-named."  Well it looks as though your marching orders are being written and handed down on Monday.  We had a good run at it, but it looks as though they figured out we were planning to start the company here soon.  This just means that you have at least a day to pull the files off the server that we need to bring over the clients to the CLEARLY better team.  US.   They are trying to surprise you by not giving you any time on your system prior to your notification.  But now you know, so you can get the files we need quickly.     I will have to wait a week and then resign in protest over your firing.     If I were you, I think it is time we started that company we always talked about now.  Our clients like us more anyway.     Be sure and grab all the files marked CONFIDENTIAL... and if possible SECRET.     See you soon.  Sorry I had to let you know this way, but who knows if they are reading corporate e-mail.   Fred Smith (LOKI) <br> </body></html>

**When you save that into a notepad document and save it as HTML. It will now look like this:**

---

Donald,

I hate to bring bad news to you. But I learned at corporate today that they found out about the "you-know-what" activity in the accounts. They saw your name all over it. I know you were directed to do this by "he-who-shall-not-be-named." Well it looks as though your marching orders are being written and handed down on Monday. We had a good run at it, but it looks as though they figured out we were planning to start the company here soon. This just means that you have at least a day to pull the files off the server that we need to bring over the clients to the CLEARLY better team. US.

They are trying to surprise you by not giving you any time on your system prior to your notification. But now you know, so you can get the files we need quickly.

I will have to wait a week and then resign in protest over your firing.

If I were you, I think it is time we started that company we always talked about now. Our clients like us more anyway.

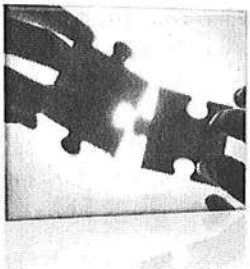
Be sure and grab all the files marked CONFIDENTIAL... and if possible SECRET.

See you soon. Sorry I had to let you know this way, but who knows if they are reading corporate e-mail.

Fred Smith (LOKI)

# Chat and Webmail Carving Summary

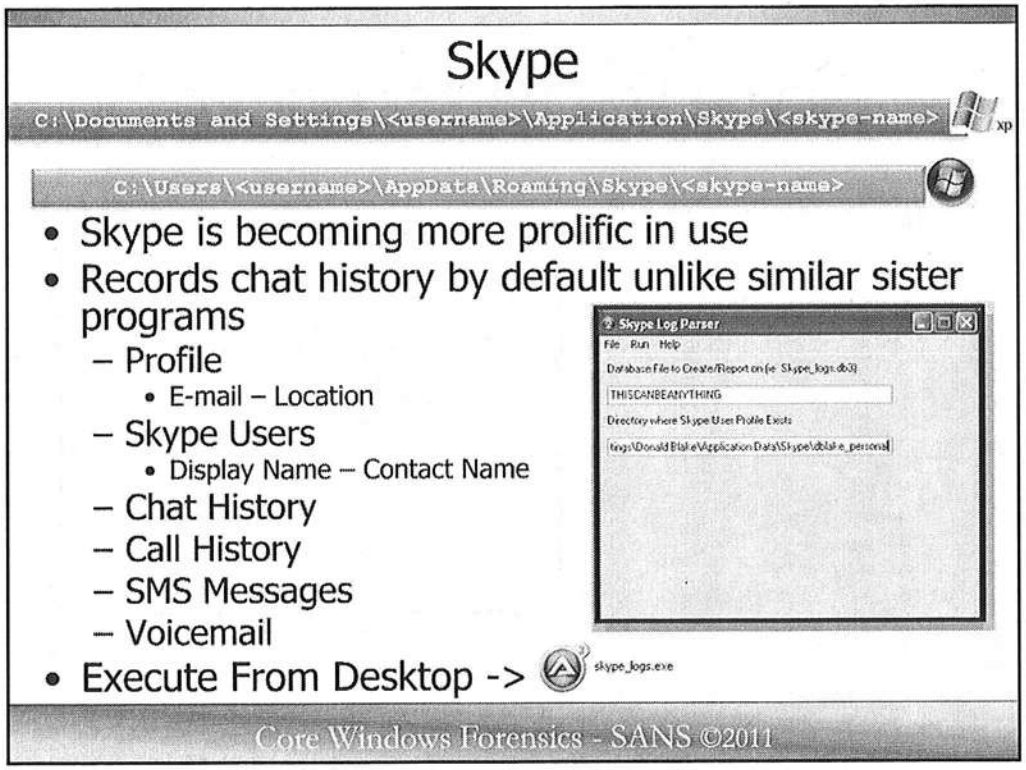
- Select Source File For Carving
  - Memory
  - Pagefile
  - Unallocated Space
  - Entire Drive



- Carve out looking for critical data items related to case
- In the end ->
  - You recovered the pieces of the puzzle
  - Analyze fragments and put it back together
  - Finding the data does not place enough context behind it -> Will require further analysis



This page intentionally left blank.




Skype is becoming more prolific in use. Most people do not know this, but Skype actually records every call, every chat session, and even SMS messages that originated from a specific machine. They are recorded in proprietary files that are stored within the user's home directory. Futhermore, Skype records chat history by default, unlike similar sister programs.

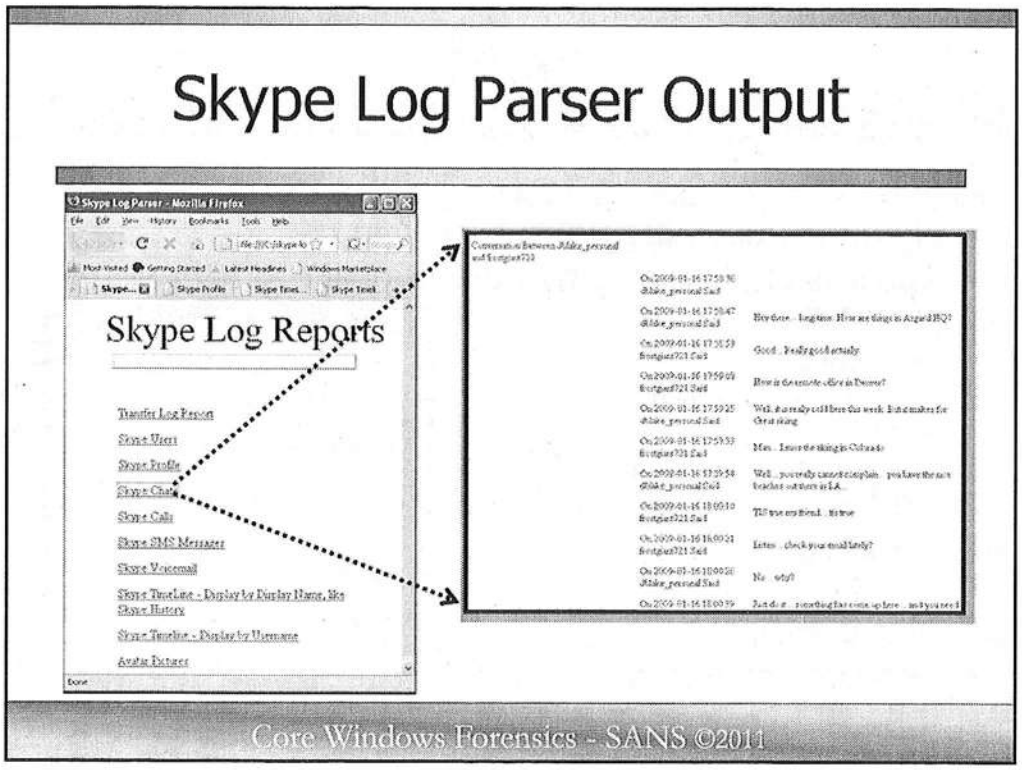
Some of the information that will record includes chat history, call history, SMS messages, and voicemail. In addition, by examining the profile of a Skype user, you could see their location, as well as, their e-mail address.

In order to examine the contents of the Skype files for specific user, there's a program called Skype Log Parser that will be found on your desktop. When you execute Skype Log Parser from your desktop, there will be two boxes asking you questions. The first box is simply the case name or whatever you would like to call it. It is not actually used by the program. The second box asks you for the location of the Skype log records, which are usually found in the users home directory.

XP-> C:\Documents and Settings\<username>\Application data\Skype\<skype-name>

Vista/Win7-> C:\Users\<username>\AppData\Roaming\Skype\<skype-name>

After running the program, a report file will be created in the Skype Log Parser's home directory. The default index.html file will automatically open when the program has completed processing the Skype history files. On the next page is the output from the chat history between Donald Blake and an individual named Frost Giant.




Conversation Between dblake_personal and frostgiant721

On 2009-01-16 17:58:36 dblake_personal Said

On 2009-01-16 17:58:47 dblake_personal Said Hey there... long time. How are things in Asgard HQ?

On 2009-01-16 17:58:59 frostgiant721 Said Good... Really good actually.

On 2009-01-16 17:59:08 frostgiant721 Said How is the remote office in Denver?

On 2009-01-16 17:59:25 dblake_personal Said Well, it is really cold here this week. But it makes for Great skiing.

On 2009-01-16 17:59:33 frostgiant721 Said Man... I miss the skiing in Colorado

On 2009-01-16 17:59:54 dblake_personal Said Well... you really cannot complain... you have the nice beaches out there in LA...

On 2009-01-16 18:00:10 frostgiant721 Said TIS true my friend... tis true

On 2009-01-16 18:00:21 frostgiant721 Said Listen... check your e-mail lately?

On 2009-01-16 18:00:26 dblake_personal Said No... why?

On 2009-01-16 18:00:39 frostgiant721 Said Just do it... something has come up here... and you need to know something

On 2009-01-16 18:01:19 dblake_personal Said OK... hold one

On 2009-01-16 18:11:45 dblake_personal Said No way

On 2009-01-16 18:12:45 frostgiant721 Said Yep... sorry, but the Cubs lost last night *wink wink* would hate to have Jerry listen in one our conversation since he is a BIG white sox fan... *wink wink*

On 2009-01-16 18:12:52 dblake_personal Said OH..... yeah... OK....

On 2009-01-16 18:12:59 dblake_personal Said sorry... a little startled still

On 2009-01-16 18:13:13 frostgiant721 Said well, you only have a few minutes before you have to go... might as well make the MOST of it

On 2009-01-16 18:13:18 dblake_personal Said yeah....

On 2009-01-16 18:13:21 dblake_personal Said Yes.

On 2009-01-16 18:13:22 dblake_personal Said ok

On 2009-01-16 18:13:25 dblake_personal Said Will chat soon

On 2009-01-16 18:29:35 dblake_personal Said DONE

# Windows Artifact Analysis

- Link Files
- Microsoft Office Metadata
  - Documents
  - Spreadsheets
  - Powerpoint
  - Adobe PDF Files
- Media File Metadata (EXIF)
- Thumbnail Analysis
  - XP Thumbs.db
  - Vista/Win7 Thumbscache
- Internet Chat Programs
  - Skype
- Recycle Bin
  - XP Recycle Bin
  - Vista/Win7 Recycle Bin
- Windows Prefetch

Core Windows Forensics - SANS ©2011

The next section we're going to look at is the Windows XP and Windows Vista/Win7 recycle bins.

# Recycle Bin Forensics

- Hidden System Folder
  - Subfolder is created with user's SID
  - Contains Recovery Files
    - "RECYCLED" 95/98
    - "RECYCLER" 2000/NT/XP/2003
  - Hidden File in directory called "INFO2"
  - INFO2 Contains Deleted Time and Original Filename
    - » Filename in both ASCII and UNICODE
    - "$Recycle.bin" Vista/Win7
  - Deleted Time and Original Filename contained in separate files for each deleted recovery file



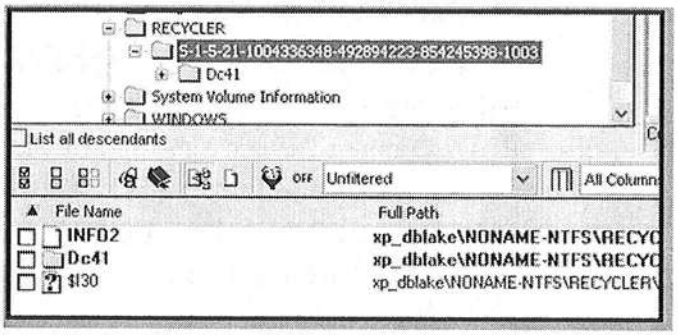
The recycle bin is a very important location on a Windows file system to understand. It can help you when accomplishing a forensic investigation as every file that is deleted from a Windows recycle bin aware program is generally first put in the recycle bin Generally, files that use command line or third party software are not placed into the recycle bin. The recycle bin is a hidden folder on your system and can be accessed from the root directory.

On Windows NT/2000/XP/2003, when a user deletes a file, his unique SID (security identifier) will be used to create a subfolder in the "Recycler" directory the first time he deletes a file. Also, internal to the directory is another hidden binary file called "INFO2" that maps the recycle bin filename to the time and date that the file was deleted as well as the true filename of the file.

On Vista/Windows 7 it changes from the Recycler directory to the $Recycle.Bin directory. Instead of a single INFO2 File, there will be a single $I (INFO) file for every single recycled file. This is more useful for forensic investigators as it will be easier to recover fragments if the recycle bin is cleared and you need to perform a data carve to search for additional $I files.

# Recycle Bin (2)

- SID can be mapped to user via Registry Analysis
- Hidden File in Recycle Bin called INFO2
- Maps filename to the actual name and path it was deleted from




Core Windows Forensics - SANS ©2011

The files in the Recycle Bin are not named by their normal names that they were deleted by. The names of the files and the paths are kept in another file. The time and date stamps on the files though will be the same.

There is a hidden file in the Recycle Bin directory under each specific SID directory that was created during a file deletion. This hidden file, named INFO or INFO2 contains the full path name, time and date, of the files that are in the directory.

When Explorer sends a file or folder to the Recycle Bin, the item's data clusters don't move. Instead, Explorer simply moves the item's directory entry into the appropriate physical bin folder on the same drive, and then renames the directory entry. Windows stores the original names and locations in a hidden index file named INFO2, which is stored in each physical bin folder. The names of items within folders sent to the Recycle Bin are not changed and are not stored in the index file.

Notice by using FTK we can easily browse to the Recycle Bin directory and open up Donald Blake's Recycle Bin folder. The folder it is using Donald Blake's security identifier, which we had previously recovered during the registry forensics section. Underneath the structuring you could clearly see the INFO2 file which contains the original folder location and filename of the deleted files that are contained in the Recycle Bin. In addition to the original folder and file name, it will contain the recycle time.

# "Recycle Bin" Examination

- You can use **FTK or rifiuti from command line** to examine an INFO2 file from your XP SIFT Workstation machine

```
E:\RECYCLER\S-1-5-21-1004336348-492894223-854245398-1003>rifiuti INFO2
Recycle bin file: 'INFO2'
Version: 5

Index   Deleted Time      Gone?   Size    Path
41      2009-01-16 18:27:24     No      34816   C:\Documents and Settings\Donald
 Blake\My Documents\Business Plans\SECRET
```

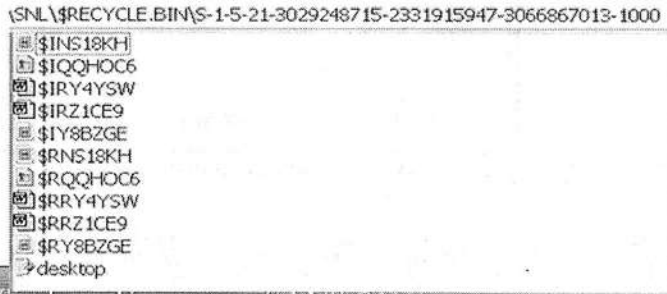
Core Windows Forensics - SANS ©2011

Let's examine the INFO2 file via the interface provided with forensic toolkit with FTK. We can see that there was originally a directory called "secret" located in the My Documents directory of Donald Blake. Using the FTK interface, you can also examine the contents of that directory by browsing inside that directory file. Are these files considered confidential or secret? When were these files recycled? These files are considered and intellectual property by Asgard Inc. They also were recycled during the last time Donald Blake was able to access this machine before he was fired.

We also demonstrate a second tool called Rifiuti that will parse the INFO2 file via the command line. Rifiuti allows you to automatically carve the INFO2 file to show the deletion time, original location, and the file size. This is an example of rifiuti parsing an INFO2 file from a user's SID directory.

# Vista/Win7 Recycle Bin

- Under **$Recycle.Bin** and **SID**
- Files Preceded by **$I######** files contain
  - Original PATH and name
  - Deletion Date/Time
- Files Preceded **$R######** files contain
  - Recovery Data



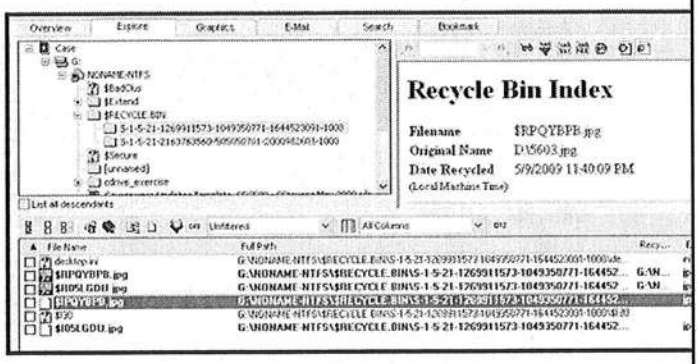

Core Windows Forensics - SANS ©2011

The Vista Recycle Bin is a little different from the previous Windows versions such as XP and 2003. The Recycle Bin in Vista/WIN7 no longer has the INFO2 file in it. Instead, it has been replaced by two types of files. The first is the $I file, which will include the deletion date/time and the original name and path. The second is the $R file, which includes the actual file recovery data.

# Parsing Vista/Win7 Recycle Bin

Vista/Win7 Parsers

- FTK – Select $I file for
  - Original Filename
  - Date Recycled
- rifuitti-vista.exe
  - Command line executable
  - Will output in text display




Core Windows Forensics - SANS ©2011

Forensic Toolkit (FTK) also parses Windows Vista/Win7 Recycle Bin files. Here is an example of selecting a $I file via the viewer. You can clearly see the original name of the file and the date recycled in the summary location. Alternatively, you use another command line executable called rifuitti-vista.exe. Like its cousin, and it will output in text-only mode via the command prompt.

# Windows Artifact Analysis

- Link Files
- Microsoft Office Metadata
  - Documents
  - Spreadsheets
  - Powerpoint
  - Adobe PDF Files
- Media File Metadata (EXIF)
- Thumbnail Analysis
  - XP Thumbs.db
  - Vista/Win7 Thumbscache
- Internet Chat Programs
  - Skype
  - AIM
- Recycle Bin
  - XP Recycle Bin
  - Vista/Win7 Recycle Bin
- Windows Prefetch

Core Windows Forensics - SANS ©2011

The final section will examine the Microsoft Windows prefetch directory found on both Windows XP and Windows Vista/Win7.

# Windows Prefetch/Superfetch

Windows XP - Vista/Win7
- Increases performance of system by pre-loading code pages of commonly used applications
- Cache Manger monitors all files and directories referenced for each application or process and maps them into a `.pf` file.
- Can be utilized to know an application was executed on a system.
- Even if someone deletes the application, a prefetch file might have been created proving existence
- Correlation with registry, logs, and timeline could tie running of application back to a specific user

- `c:\Windows\Prefetch`

  Limited to 128 files on XP and Vista/Win7
  - `(exename)-(hash).pf`

- `c:\Windows\Prefetch\Layout.ini`
  - Windows regularly examines the contents of the .pf files and writes the files and directories used by the process to the `layout.ini` file
  - Contains original path names of the files located in the prefetch
  - Disk Defragmenter will use information in layout.ini to relocate all directories and files to a contiguous area of the disk



Prefetching is a process in which the operating system loads key pieces of data and code from disk into memory before it's actually needed. The prefetch directory is populated once an application is executed, thus why it might be a good idea to grab the contents of the prefetch directory manually before performing any incident response of a machine as the contents of this directory could be considered fairly volatile. The Cache Manger monitors all files and directories referenced for each application or process and maps them into a .pf file.

Windows XP/2003/Vista/Win7 are set up by default for boot prefetching to decrease the time to boot the system.

A prefetch file will be named with the executable followed by a hex based representation of the hash of the path the file resides in, and a pf extension. Located within the file will detail the number of times the application has been executed, the original path of execution, and the last time of execution.

Windows regularly examines the contents of the .pf files and writes the files and directories used by the process to the layout.ini file, which contains original path names of the files located in the prefetch. The Disk Defragmenter will use information in layout.ini to relocate all directories and files to a contiguous area of the disk.

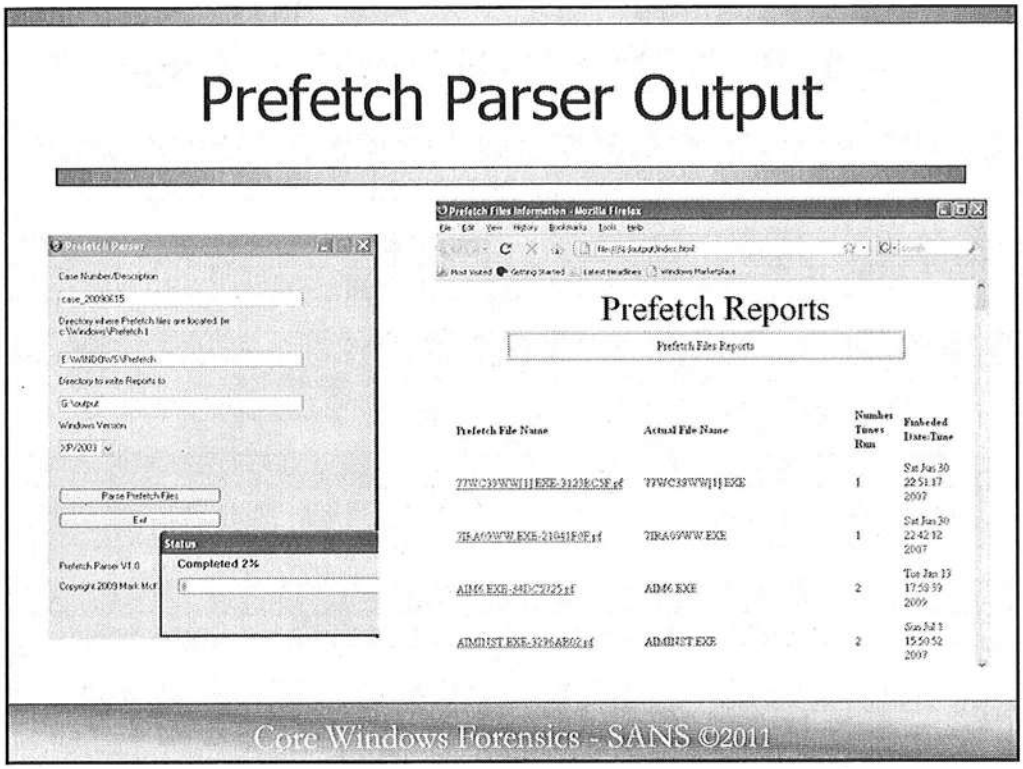
# Windows Prefetch

- Vista/Win7 SuperFetch
  - System Memory will populate super-fetch as user programs will be swapped in and out depending on if the user is actively using the programs.
- **`Prefetch Parser`** Program
  - Written by Mark Mckinnon
  - Will analyze the prefetch directory and retrieve the MACtimes and the last run times of the files in the prefetch

On Windows Vista/Win7, prefetch, now called Superfetch, was improved by loading the majority of the files and data needed by a process into memory and when it isn't needed, the prefetched information is written out to the pagefile. Superfetch also analyzes applications a user utilizes to include information about how often a program is used and the time it is most likely to be used. Research is currently taking place on these artifacts to see how this information can be forensically mined.

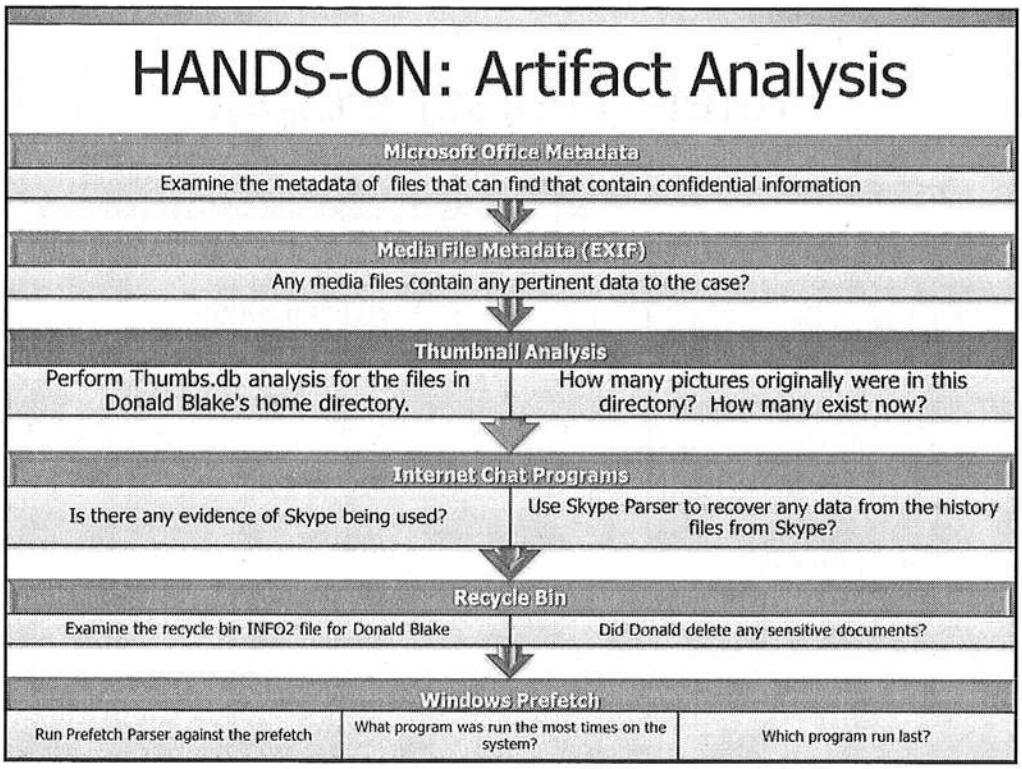
Prefetch

http://www.microsoft.com/windows/products/windowsvista/features/details/performance.mspx




The Prefetch Parser is relatively easy to run. When you execute it you type in the case name, the prefetch directory, and the outputs directory where you want to report to be written to. It will also allow you to select between Windows XP and Windows Vista/Win7 prefetch files.

What I like about this tool is it will show you the last date the program was executed in addition to the total number of times that specific program has been executed.




**Link Files**

Examine link files in Donald Blake's Home Directory

Detail any files/locations that seem suspicious.

**Microsoft Office Metadata**

Examine the metadata of files that might contain critical confidential information

**Media File Metadata (EXIF)**

Any media files contain any pertinent data to the case?

**Thumbnail Analysis**

Perform Thumbs.db analysis for the files in Donald Blake's home directory.

How many pictures originally were in this directory? How many exist now?

**Internet Chat Programs**

Is there any evidence of Skype being used?

Use Skype Parser to recover any data from the history files from Skype?

**Recycle Bin**

Examine the recycle bin INFO2 file for Donald Blake

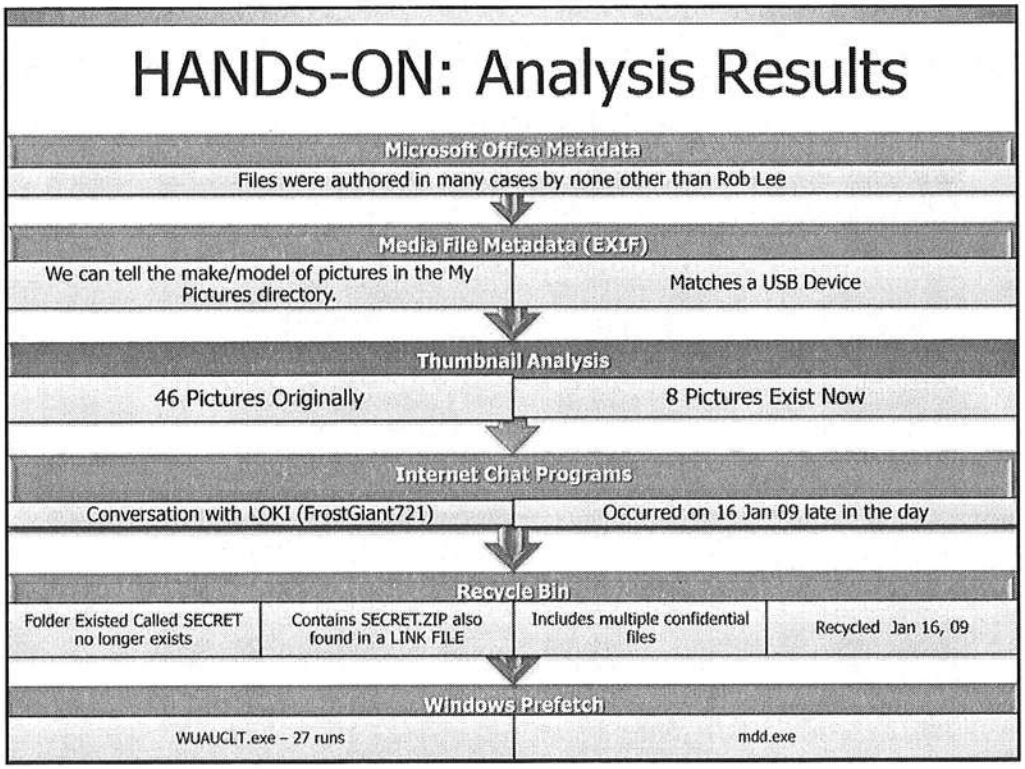
Did Donald delete any sensitive documents?

**Windows Prefetch**

Run Prefetch Parser against the prefetch

What program was run the most times on the system?

Which program run last?




**Link Files**

Link Files show files on two removable drives E: and F:

Matches USB Device From Earlier

**Microsoft Office Metadata**

Files were authored in many cases by none other than Rob Lee

**Media File Metadata (EXIF)**

We can tell the make/model of pictures in the My Pictures directory.

Matches a USB Device

**Thumbnail Analysis**

46 Pictures Originally

8 Pictures Exist Now

**Internet Chat Programs**

Conversation with LOKI (FrostGiant721)

Occurred on 16 Jan 09 late in the day

**Recycle Bin**

Folder Existed Called SECRET no longer exists

Contains SECRET.ZIP also found in a LINK FILE

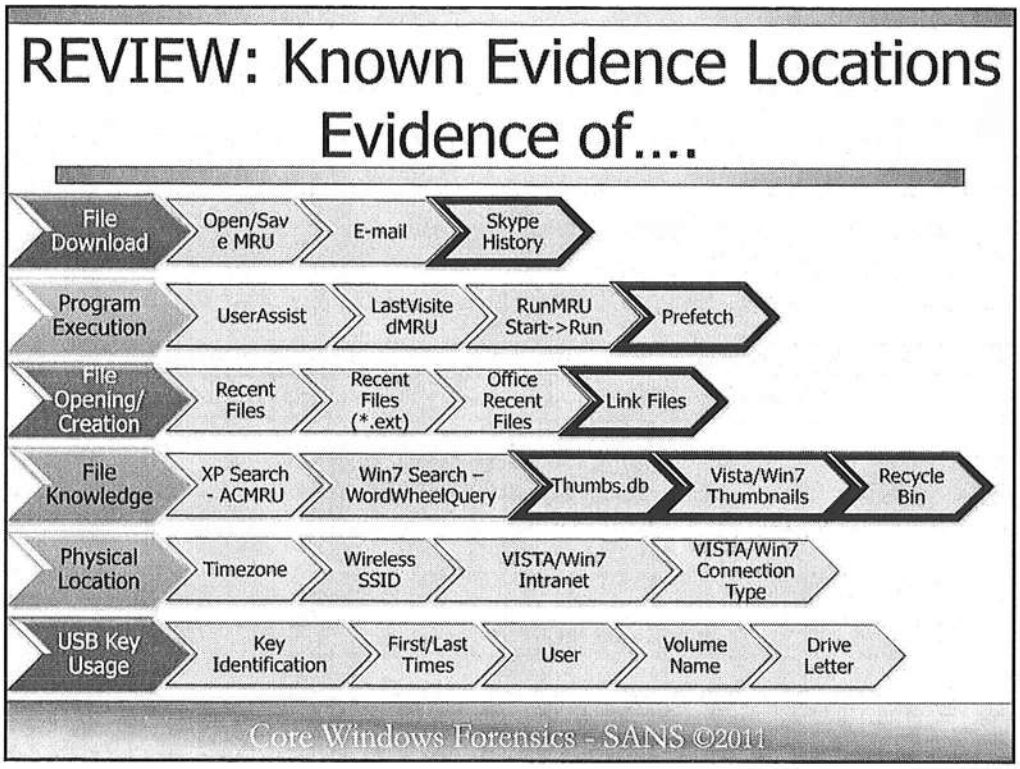
Includes multiple confidential files

Recycled Jan 16, 09

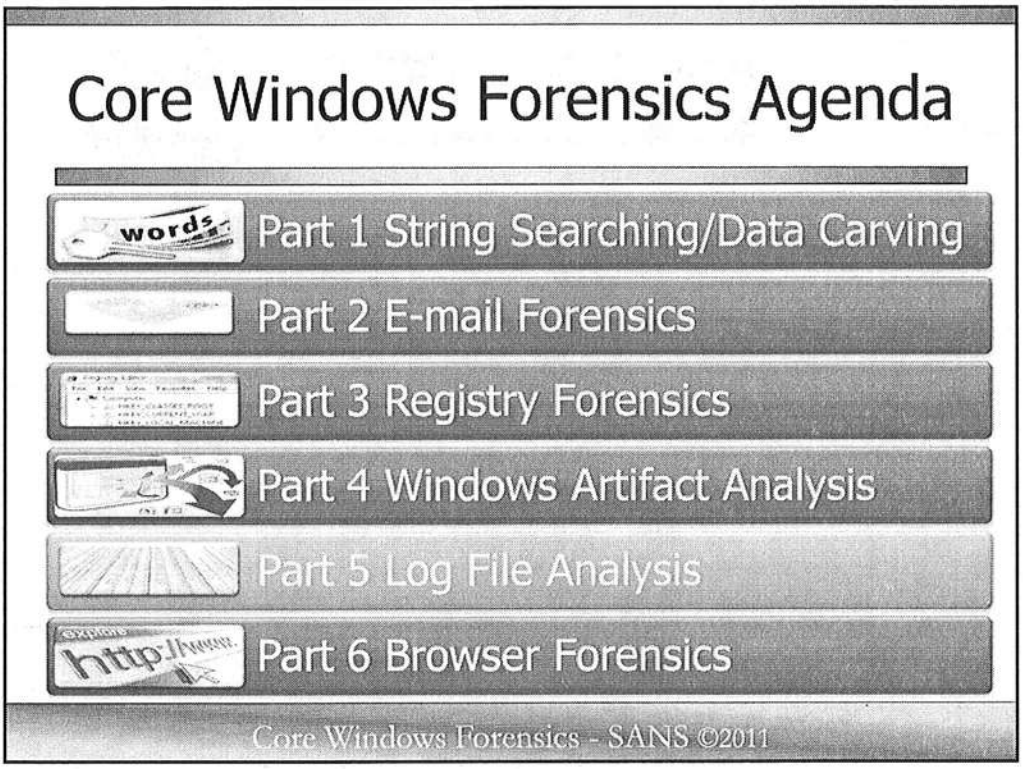
**Windows Prefetch**

WUAUCLT.exe – 27 runs

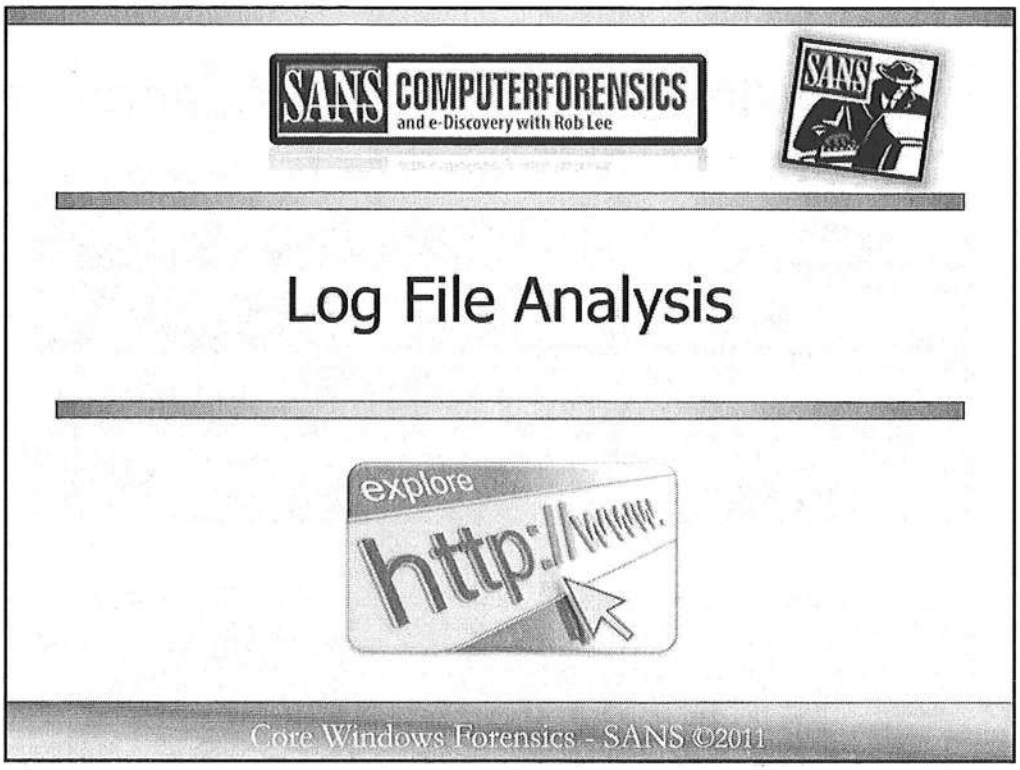
mdd.exe was run last




This page intentionally left blank.




This page intentionally left blank.




This page intentionally left blank.

# Log File Analysis

- Event Logs
- Windows Firewall Logs
- IIS Logs



This page intentionally left blank.

SANS COMPUTERFORENSICS
and e-Discovery with Rob Lee

# Event Log Analysis



This page intentionally left blank.

# Event Log Overview

- Event Log Fundamentals
- Event Log Analysis
- Vista / Win7 / Server 2008 Logs
- Advanced Techniques



This page intentionally left blank.

[redacted]

[redacted]

# Windows Events

**"Any significant occurrence in the system or in a program that requires users to be notified"**

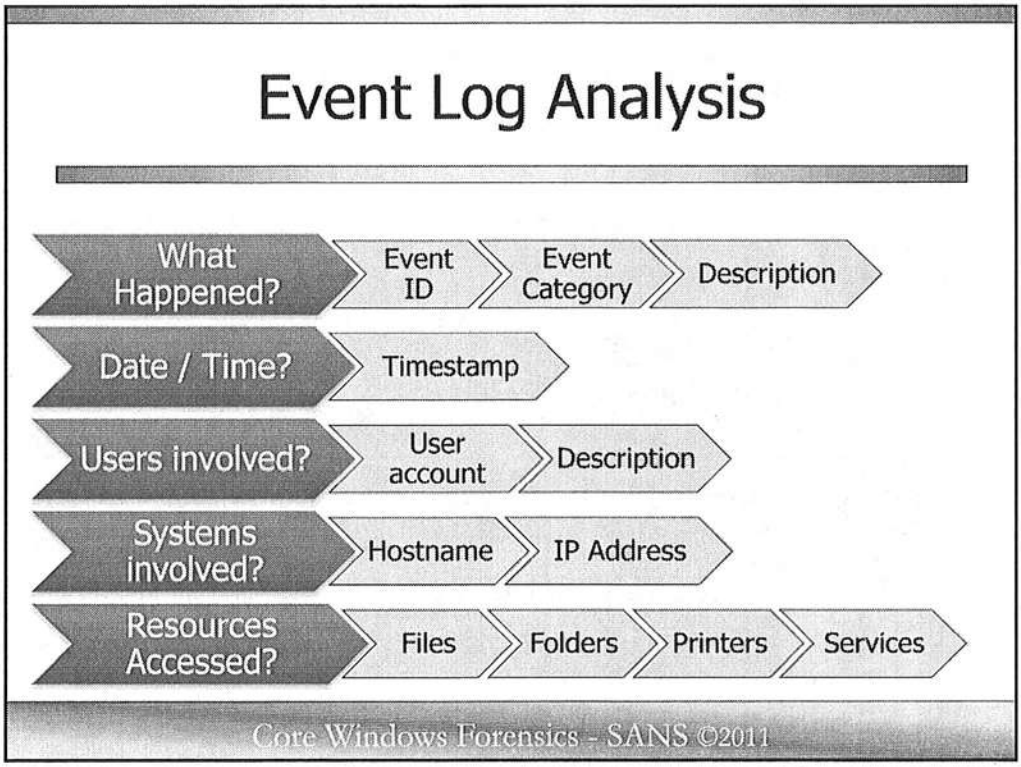
- Centralized recording of information about:
  - Software
  - Hardware
  - Operating system functions
  - Security
- Multiple events comprise an *event log*



Event logging provides a standard, centralized way for the operating system and associated applications to record important software and hardware information. Microsoft describes an event as "any significant occurrence in the system or in a program that requires users to be notified, or an entry added to a log. " [1]

Events are collected and stored by the Event Logging Service. It stores events from various sources in a single collection called an *event log*. Event logs provide historical information that can help illuminate system and security problems as well as tracking user actions and system resource usage. However, what is actually recorded in the event logs is highly dependent on the application involved and the system settings. As an example, security event logging is disabled by default on most freshly installed Windows systems. If they exist, event logs can be an incredible boon to a forensic investigation, providing both local and network context that is difficult to replicate with other artifacts.

[1] http://support.microsoft.com/kb/308427




Event logs provide a wealth of information that can help an investigator piece together relevant actions that occurred on the system. Similar to other forensic artifacts, it is helpful to go through the mental exercise of determining what questions event log data can help answer. Some of the most common are:

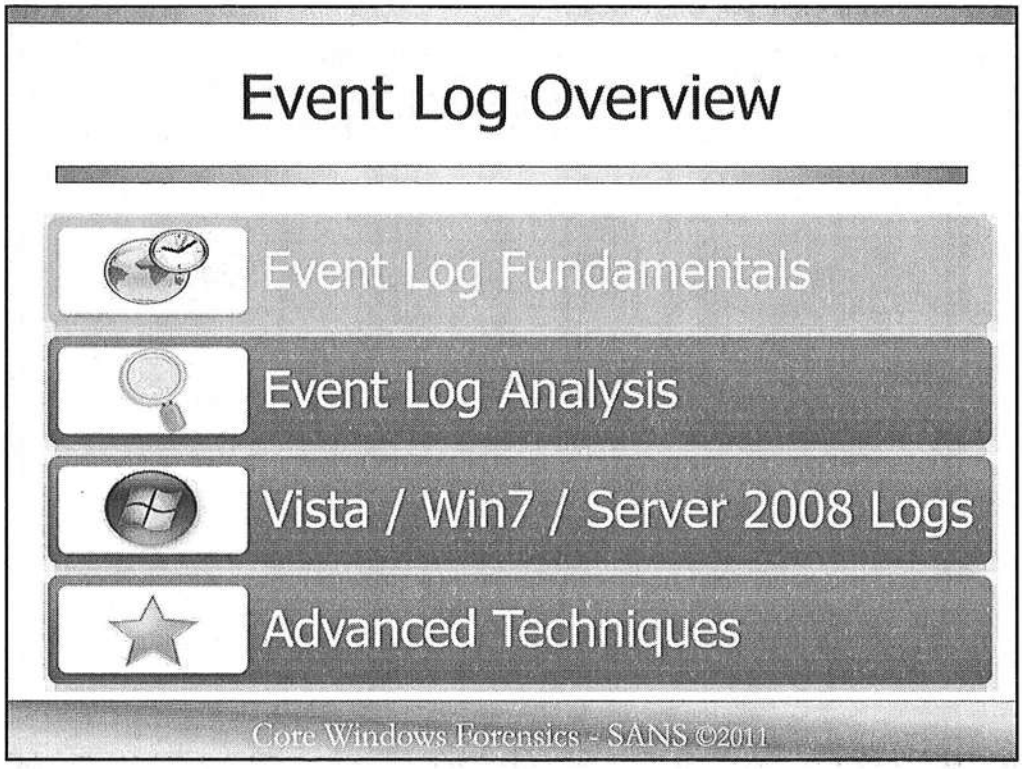
What Happened? Event logs can be cryptic to the lay user, but they are designed to provide very specific information about activities that occurred on the system. Items like Event IDs and Event Categories help us quickly find relevant events and the Event Description can provide further information as to its nature.

Date/Time? Timestamps are a key part of event logs, providing a temporal context for the events. With systems recording thousands of events, timestamps can also help the investigator narrow their focus.

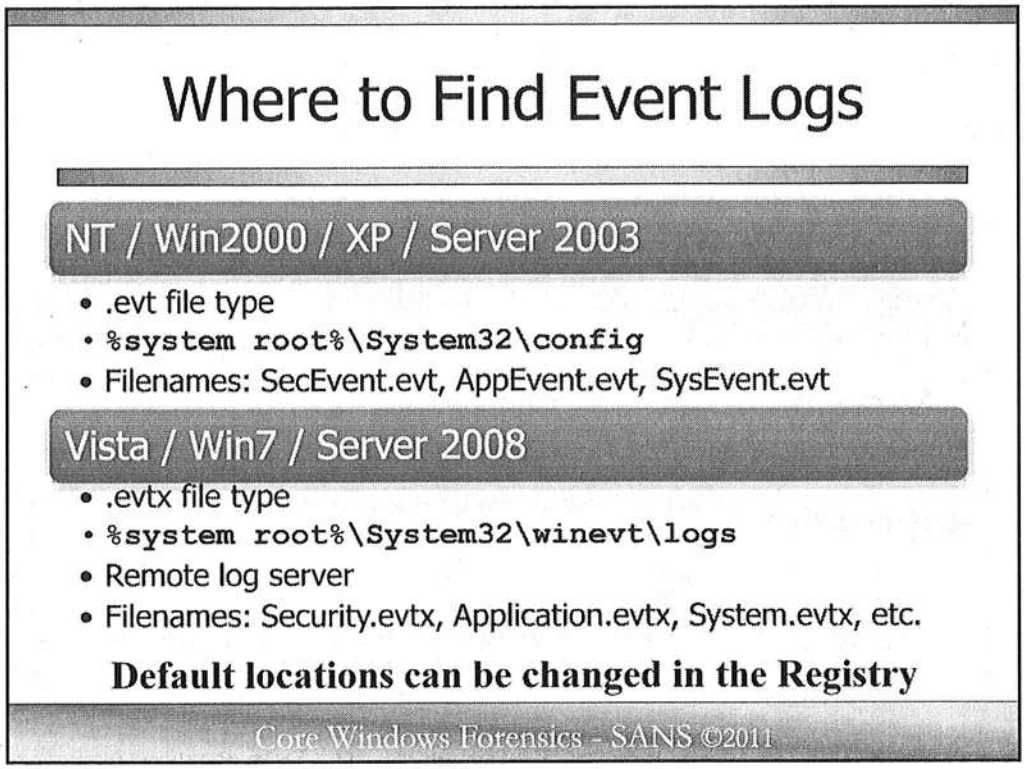
Users Involved? Everything accomplished within Windows is done within the context of an account. We can identify references to specific users as well as information about Windows operating system activities performed via special accounts like System and NetworkService.

Systems Involved? In a networked environment, we will very commonly find references to systems other than the host as resources are accessed remotely. Originally, only the Netbios name was recorded, making tracking and attribution much more difficult. In systems post-Windows 2000, IP addresses are now recorded within the event logs (when applicable).

Resources Accessed? The Event Logging Service can be configured to store very granular information regarding the use of various system objects. With nearly every resource considered an object, this provides very powerful auditing. As an example, this can help identify attempted access to unauthorized files on a system.




This page intentionally left blank.




Event logs as we know them today originated with the NT 3.1 operating system in 1993. [1] Small upgrades were seen throughout the Windows NT evolution, but the names and locations of logs largely remained unchanged until Windows 2003. This original log format used the .evt extension. Logs are stored in binary format, complicating byte-level string searching, and are implemented using a circular buffer. The circular buffer loops around to (eventually) overwrite the oldest entries with the most recent. Event logs prior to Vista can be found in:

**%system root%\System32\config**

Starting with the Vista and Server 2008 product line, significant changes to the event log structures, log types, and log location were made. Event logs have historically exacted a huge performance drain on systems and hence the new format, using the .evtx extension, was created to fix this and many other problems. The good news is that with the new optimizations, we are more likely to find Event Logs being used on the newest operating systems. In addition to radical changes to the event log structures, Vista and above systems now employ a much, much larger number of logs, and hence a new folder was created to house these 60+ logs. These logs can now be found in:

**%system root%\System32\winevt\logs**

Additionally, the new log format (finally) allows logs to be sent to remote log collector, so it is important to remember that additional logs may be available on external servers.

It is important to note that the folders listed here are only the default locations. The administrator can designate locations for individual logs within the following registry keys (Vista employs these and several others to describe newly added logs):

**HKLM\SYSTEM\CurrentControlSet\Services\EventLog\Application**
**HKLM\SYSTEM\CurrentControlSet\Services\EventLog\System**
**HKLM\SYSTEM\CurrentControlSet\Services\EventLog\Security**

[1] http://en.wikipedia.org/wiki/Event_Viewer

# Types of Event Logs

| | |
|---|---|
| **Security** | • Records access control and security settings information<br>• Events based on audit and group policies<br>• Example: Failed Logon; Folder access |
| **System** | • Contains events related to Windows services, system components, drivers, resources, etc.<br>• Example: Service stopped; System rebooted |
| **Application** | • Software events unrelated to Operating System<br>• Example: SQL Server fails to access a database |
| **Custom** | • Custom application logs<br>• Examples: Server logs including *Directory Service, DNS Server*, and *File Replication Service* |



Event logging began with three primary logs, Security, System, and Application, and these logs have maintained their importance throughout all Windows NT platforms. Along the way, additional logs have been added for specialized logging, which we have grouped here under "Custom" logs. Vista, Win7 and Server 2008 have greatly expanded the number of "custom" logs, enforcing log segmentation and providing specialized logs for processes like the Powershell, Task Scheduler, and the Windows Firewall.

**Security Log:** Records events based on auditing criteria provided by local or global group policies.

**System Log:** Records events logged by the operating system or its components, such as the failure of a service to start during the boot cycle.

**Application Log:** Records events logged by applications, such as the failure of MS SQL to access a database or an anti-virus alert.

**Directory Service:** Standard on Domain Controllers. Records events logged by Active Directory and its related services.

**File Replication Service:** Standard on Domain Controllers. Records updates between the domain controller infrastructure.

**DNS Server:** Standard on servers running the DNS Service. Records DNS queries, responses, and other DNS activities.

# Security Log

- Most commonly reviewed log in forensics
  - User authentication and logon
  - User behavior and actions
  - File / Folder/ Share access
  - Security settings modifications
- Failure and Success can be audited
  - Detailed logging can be enabled on specific user accounts
- Only updated by the LSASS process
  - Third party applications cannot insert events



While almost all event logging has the potential to be useful during an investigation, most of the questions we are looking to answer during a forensic investigation tend towards answers found in the Security Log. The System and Application logs store information more useful for troubleshooting by system administrators. The Security log records an audit event whenever a given system or user action meets the criteria set forth by the audit policy in use. They can provide details on a variety of actions, including user authentication (logons, runas command, remote access, etc.) and what a particular user did on a system after authentication. As an example, privilege use and object auditing can trigger events showing that a protected file or folder was accessed, which user account accessed it, and the date and time it occurred. Auditing is also allowed on the security settings themselves, providing a good record of any modifications to the existing security policies on the system.

An important concept is that the audit policy can be set to trigger events for both successful and failed attempts. This allows for a finer granularity of auditing and provides tailored data reduction – only recording the exact actions that a security administrator finds useful in their environment. As forensic analysts, we would love to have everything logged, but that entails a performance and storage hit that is not possible in many environments. That being said, audit policy should be vetted, because it is not always obvious to a non-security professional why something like Successful AND Failed logon attempts should be logged (the former could allow us to track a compromised account being used throughout he environment while the latter could indicate password guessing attacks). Keep in mind that it is possible to tailor auditing for a specific user account using the **`auditusr.exe`** command-line tool in Windows. Thus if you suspect a specific account has been compromised by an intruder, or you would like more detailed auditing on critical accounts like Domain Administrators, this tool can provide that capability on a per account basis.

Due to its nature, the Security Log has more protections in place than the System and Application logs. With XP SP2, the API was deprecated for applications other than the Windows Security Service to trigger events in the Security Log. [1] This ability is now only held by the Local Security Authority Subsystem Service (LSASS) since it is responsible for enforcing the security policy on the system. Additionally, only user accounts with Administrator permissions can review, export, or clear the log.

[1] http://support.microsoft.com/kb/891749

## What is Recorded?
## Security Event Categories

| | |
|---|---|
| **Account Logon** | Events stored on system who *authorized* logon (i.e. Domain Controller or local system for non-Domain accounts) |
| **Account Mgmt** | Account maintenance and modifications |
| **Directory Service** | Attempted access of Active Directory objects |
| **Logon Events** | Each instance of logon / logoff on local system |
| **Object Access** | Access to objects identified in system access control list |
| **Policy Change** | Change of user rights, audit policies or trust policies |
| **Privilege Use** | Each case of an account exercising a user right |
| **Process Tracking** | Process start, exit, handles, object access, etc. |
| **System Events** | System start and shutdown; actions affecting security log |

Core Windows Forensics - SANS ©2011

Security Event Categories give us a quick means to identify event log entries that may be of interest to our investigation. They follow directly from the enabled audit policies on a Windows system. For each of these categories, the audit policy can be set to No Auditing, Success, Failure, or both Success and Failure.

When events are triggered and recorded in the logs, they will be marked with the specific category that they belong to. When auditing is disabled for any given category, we should expect to not see any recorded events of that type.

From Microsoft Technet [1]:

**Audit account logon events:** audit each instance of a user logging on to or logging off from another computer in which this computer is used to validate the account.

**Audit account management:** audit each event of account management on a computer. Examples of account maintenance include password changes, user account and group modifications.

**Audit directory service access:** audit the event of a user accessing an Active Directory object that has its own system access control list (SACL) specified.

**Audit logon events:** audit each instance of a user logging on to or logging off from a computer. Note that this is different than the 'Audit account login events' category. This tracks the logon event to a specific server. The former tracks which domain controller authenticated the user.

**Audit object access:** audit the event of a user accessing an object that has its own system access control list (SACL) specified. Examples of objects are files, folders, registry keys, printers, etc.

**Audit policy change:** audit every incident of a change to user rights assignment policies, audit policies, or trust policies. Audit privilege use - audit each instance of a user exercising a user right.

**Audit process tracking:** audit detailed tracking information for events such as program activation, process exit, handle duplication, and indirect object access.

**Audit system events:** audit when a user restarts or shuts down the computer or when an event occurs that affects either the system security or the security log.

[1] http://technet.microsoft.com/en-us/library/cc766468%28WS.10%29.aspx

## What are We Likely to Find? Default Security Logging

| | Workstation (NT-Win7) | Server (NT-2008) | Domain Controller (2003-2008*) | Recommended Baseline |
|---|---|---|---|---|
| **Account Logon** | | 2003-Success | Success | Success / Failure |
| **Account Mgmt** | | | Success | Success / Failure |
| **Directory Service** | | | Success | Success / Failure |
| **Logon Events** | | 2003-Success | Success | Success / Failure |
| **Object Access** | | | | Success / Failure |
| **Policy Change** | | | Success | Success / Failure |
| **Privilege Use** | | | | Failure |
| **Process Tracking** | | | | |
| **System Events** | | | Success | Success / Failure |

* Windows 2008 includes more detailed policy options

Core Windows Forensics - SANS ©2011

There is nothing more disappointing during an investigation than hitting a dead end. A frequent dead end encountered is the lack of event logs. In certain types of investigations, such as intrusion cases, event logs are crucial for tracking activities throughout the enterprise. In others, such as employee misuse, they provide additional artifacts to strengthen the case such as after-hour logons, program installation or restricted folder access. The sad fact is that most environments do not have strong enough logging. Traditional Windows logging can generate an overwhelming amount of logs if left unchecked. In more regulated environments, such as PCI-compliant shops, event logs will often be available and well managed. This chart shows the default audit policies for Workstations, Servers, Domain Controllers, and finally what the recommended auditing baseline should be on a secure system. As shown here, this is a massive discrepancy between the default and recommended configurations. [1]

In most Windows installations, event logging is largely disabled out of the box. Enabling it requires an administrator to modify the security audit policies after installation. Workstations are where we typically find the least amount of logging. This is particularly true in a standalone installation, since there is no logging enabled by default. Thus in the "vanilla" law enforcement search warrant served on a residence, they will more often than not find no event logs on the seized systems. In a Domain or Active Directory environment, the local audit policy will be overridden by the group policy which may enable local logging.

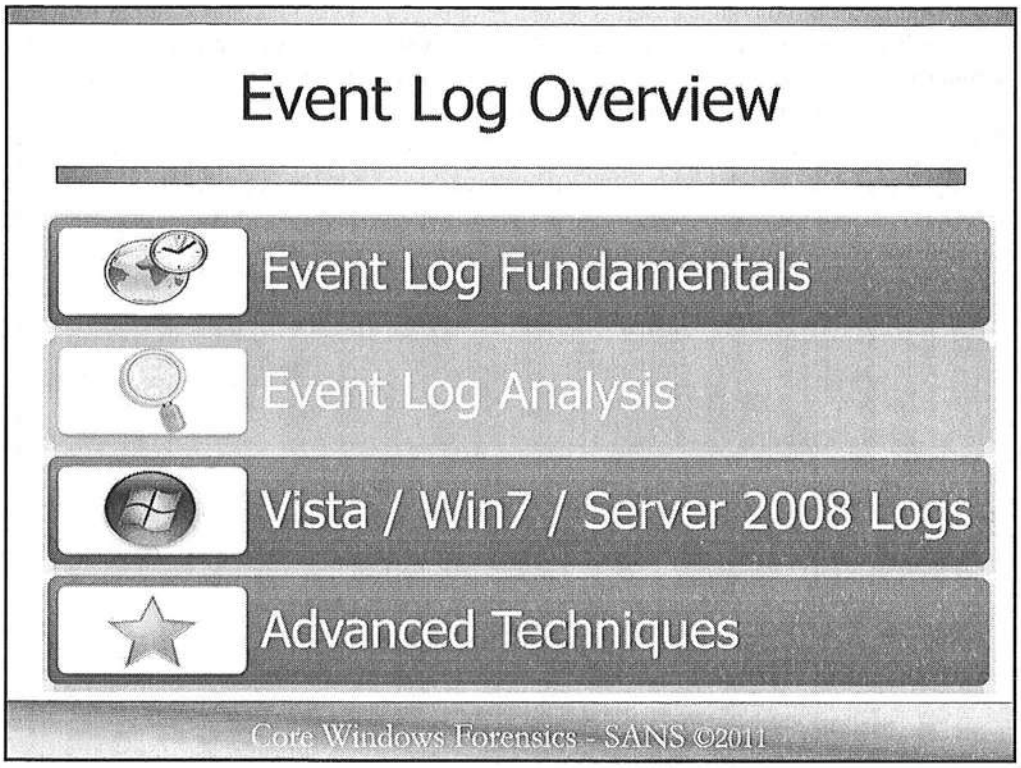
Many investigators wrongly assume that Microsoft server products will have strong logging enabled. Except for Windows 2003, which monitors successful logons, the standard server configuration has no logging enabled by default, from NT 3.1 all the way to Windows Server 2008 R2.

The default Domain Controller install for Windows 2003 and above does have some logging enabled by default. Windows 2008 servers have slightly different logging than shown here because that operating system now provides more detailed policy options within many of the audit categories listed. [1] These detailed options were included to allow better control over just the events that the administrator cares to audit/review.

The Recommend Baseline column is a well accepted audit policy for secure systems. [2] It limits logging for privilege use and process tracking because those two categories can provide an overwhelming amount of data for every action performed by every user/process. They are also the least used events when performing security investigations.

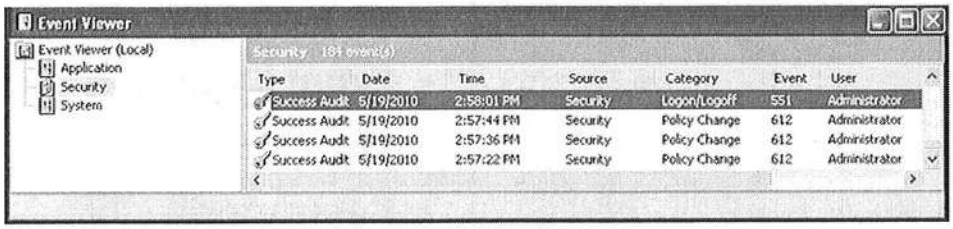
[1] http://blogs.technet.com/askds/archive/2007/10/19/introducing-auditing-changes-in-windows-2008.aspx

[2] Smith, Ben: Microsoft Windows Security Resource Kit




This page intentionally left blank.

# Microsoft Event Viewer



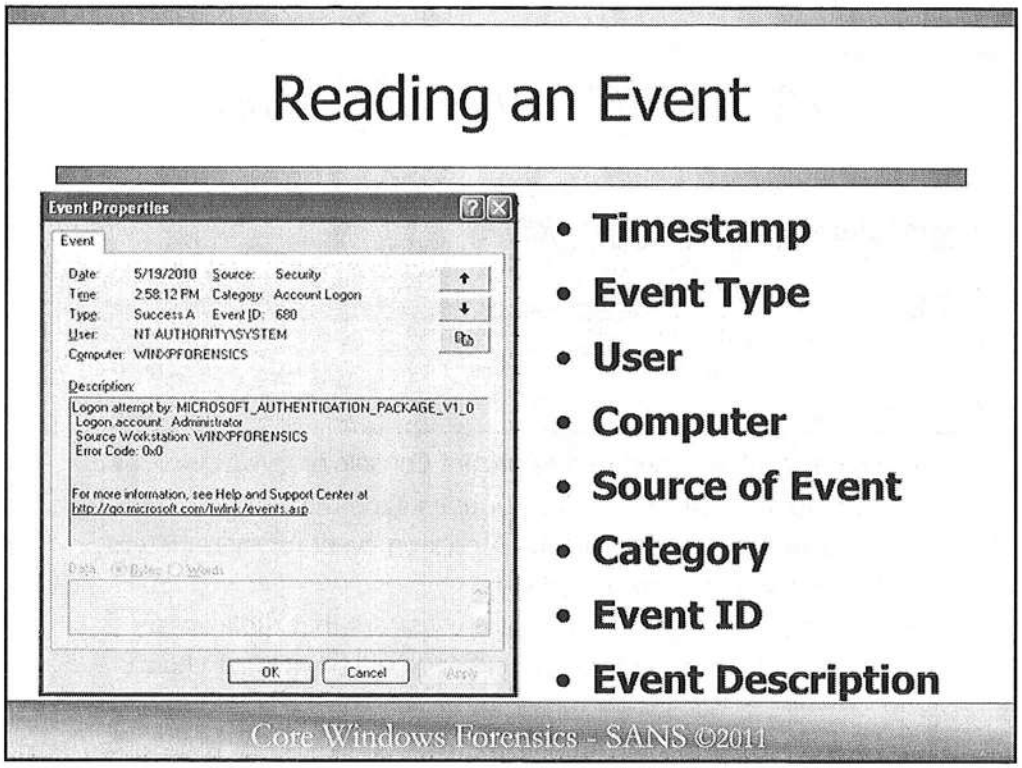

- Built-in tool accessed through Mgmt Console or **`eventvwr.exe`**
- Used to view live system logs or open exported logs
- Gives table view of all events → click on event for more details
- Very intolerant of corrupted logs
- Filtering and search capabilities are present, but rudimentary
- Vista version is superior, but neither would be my first choice



System administrators should feel right at home with the Microsoft Event Viewer. It has stayed largely static until MS Vista, and is the built-in tool for reviewing event logs recorded on a system. The viewer can be accessed via the Computer Management Microsoft Management Console (MMC) -- <right-click> on My Computer in the Start Menu and select "Manage". Alternatively it can be opened via the Run command bar or command line using its file name, **`eventvwr.exe`**.

While most often used to view live event logs (a feature not used very frequently during forensics), it also has the capability to open and read logs in the .evt format. Thus logs exported from other machines can be safely reviewed on a Windows forensic workstation using this tool. Once a log file is opened, the viewer will display a table view of all available events. The table can be sorted by clicking on the column headers, and filtered using the View → Filter command. To see more detail, double-click on the event and an "Event Properties" GUI will emerge showing all information stored for that event.

While the Event Viewer gets the job done, it has some drawbacks. It can be very cumbersome to review a large number of events within the interface. There is no capability to load multiple security logs (i.e. from different systems) and filter or search in parallel. Its biggest negative is that it is very intolerant of corrupted logs. Due to the nature of forensic data collection, we often encounter corrupted event logs, and I have seen less experienced forensic analysts give up when those logs could not be opened in Event Viewer! Never fear, this course will teach you alternative methods.




Each recorded event consists of nine elements which define the event and provide valuable information regarding why that event was triggered. [1] In this slide we are showing these elements using the detailed Event Properties function of the Event Viewer. However, it is important to note that the elements are all stored in a standard binary log format and can be parsed and displayed using a variety of tools. The information available is as follows:

**Date / Time:** The timestamp of when the event was triggered. Displayed in **Local System Time**

**Event Type:** Provides the reason the event triggered and in some cases its severity

**User:** The account that triggered the event

**Computer:** The name of the computer where the event occurred (useful when reviewing logs from different systems simultaneously)

**Source:** The application, service, or Microsoft component that logged the event

**Category:** The category of the event; in Security logs this will be the Security Event Category managed via the audit policy.

**Event ID:** A unique code assigned to various system functions. Can often be the most useful indicator of what triggered the event (if you have a good Event ID reference)

**Description:** A canned text description of the event, sometimes with additional information. Descriptions excel at "saying nothing", but can sometimes provide valuable information including remote IP addresses, hostnames, etc.

**Data:** An optional field that can contain additional raw data (or error codes) generated by the event

[1] http://technet.microsoft.com/en-us/library/bb726966.aspx

# Event Types

| | |
|---|---|
| **Error** | • Significant problem; Loss of data or functionality<br>• Example: Service fails to load |
| **Warning** | • Not significant, but could indicate a future problem<br>• Example: Disk space is low |
| **Information** | • Successful operation of application, driver or service<br>• Example: Event Log Service was started |
| **Success Audit** | • Audited security event completed successfully<br>• Example: Successful user logon |
| **Failure Audit** | • Audited security event did not complete successfully<br>• Example: Failed access to a network drive |

Core Windows Forensics - SANS ©2011

Event types give us our first indicator of why an event was recorded in the first place. They can also give important information as to the severity of the event, allowing system administrators to focus limited time on those event which could pose critical risks to the health of the system. The following describes these event types in greater detail [1]:

**Error**: Indicates a significant problem such as loss of data or loss of functionality. For example, a Service fails to load

**Warning**: Not necessarily significant, but could indicate a possible future problem. For example, low disk space

**Information**: Describes the successful operation of an application, driver, or service. For example, the Event Log Service was started successfully

Security logs are typically populated with Success and Failure audit event types (Information events are also sometimes seen). These event types are driven directly from the audit policy set up by the administrator. Hence for each Security Event Category, it is possible to see either Success or Failure event types.

**Success Audit**: An audited security access attempt that was successful. For example, a successful user logon

**Failure Audit**: Records an audited security access attempt that fails. For example, failed access to a network drive

[1] http://support.microsoft.com/kb/308427

# Analysis Scenarios

- **Tracking Account Usage**
  - Brute Force Attack
  - Remote Desktop Protocol
  - Finding a Rogue Local Account
- **Analyzing File and Folder Access**
- **Application Installation**
- **Malware Execution**
- **Suspicious Services**
- **Event Log Clearing**
- **Unauthorized Hardware Devices**



This page intentionally left blank.

# Tracking Account Usage

**Scenario**

- Determine which accounts have been used for attempted logons.
- Track account usage for known compromised accounts

**Relevant Event IDs**

- 528 – Successful Logon
- 529 – Failed Logon
- 538 – Successful Logoff
- 540 – Successful Network Logon (example: file shares)

**Investigative Notes**

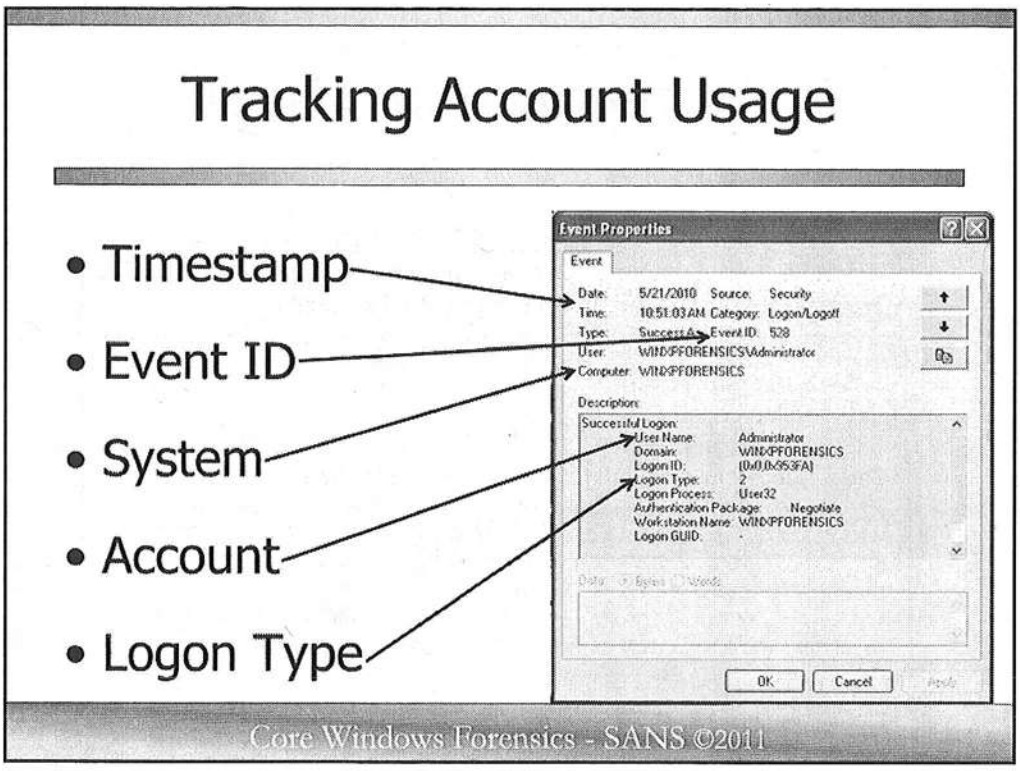
- Event IDs 528-552 cover an exhaustive list of logon activities; codes listed here are most common.
- Logon events not recorded when backdoors, exploited services, or similar malicious means are used to access a system

Core Windows Forensics - SANS ©2011

Tracking account usage is one of the more common uses for reviewing event logs. Knowing when a user account logged on to a system and subsequently logged off can provide helpful corroborating evidence along with other forensic artifacts found. If account credentials are suspected to be compromised, reviewing successful logons throughout your network can help track where the hacker has been. We will also see that remote logons are recorded and can again provide very good profiling information on how an authorized or unauthorized user is traversing the network and attempting to authenticate with resources.

Account usage Event ID Codes can be confusing because there are so many codes used by Windows (prior to Vista). There are 24 Event IDs for logon actions, numbered 528-552. The most common are the 528 / 538 pair, which show a successful logon / logoff as well as the time period of the complete user session. 529 IDs logon failures are often reviewed for evidence of password guessing attacks, and 540 IDs provide information on remote logons, complete with the IP address and hostname of the machine attempting to authenticate. The IDs covered on this slide are triggered by a mix of Success and Failure audits. They can be used for justifying the inclusion of both Success and Failure into your Logon Events Audit Policy.

When a hacker gains access to a system through some sort of an exploit (remote code execution, privilege escalation, service exploitation, client-side attacks resulting in backdoors, etc.) there is typically no record of "logon" within the event logs. This is intuitive because a backchannel is being used and the standard APIs for access are being circumvented.




When auditing account usage events, we will focus on five fields of the event record. Our preliminary information comes from the header. There we find the timestamp information, providing the date and time that the account logon occurred. The System reference tells us the hostname of the system that the event was recorded on. This can be very helpful when reviewing logs from multiple systems simultaneously to find logon patterns. Finally, the Event ID tells us what type of Logon/Logoff event this is, and in this case it is Event ID 528, which is reserved for Successful Logons. An excellent reference for a wide range of Event IDs is the following website:

http://www.ultimatewindowssecurity.com/securitylog/encyclopedia/

We can gather further details within the Event Description. Most notably, the User Name field tells us which account was successfully logged into. In this case it was an account named "Administrator". The Logon Type field is easy to miss, but plays a very important role in tracking account usage. There are multiple different ways that a logon can occur in a modern day Windows system. It isn't enough to just tell us that the Administrator account successfully logged into this system at 10:51:03 AM on 5/21/2010. We would also like to know if they were using the console, or logging in via some network protocol like SMB, or using Remote Desktop. The Logon Type can give us this further information. In this case, Logon Type 2 indicates that the logon was accomplished via a user sitting at the physical machine console.

It is important to note that we usually don't just rely upon one event. After gathering information from this 528 Event, we would then review other events surrounding it in addition to looking for a matching 538 Event, indicating that the user logged off from this session.

# Logon Type Codes

| Logon Type Code | Explanation |
|---|---|
| 2 | Logon via console (at physical machine) |
| 3 | Network logon |
| 4 | Batch Logon – Often used by Scheduled Tasks |
| 5 | Windows Service Logon |
| 7 | Credentials used to unlock screen |
| 8 | Network logon sending credentials in cleartext |
| 9 | Different credentials used than logged on user – runas command |
| 10 | Remote interactive logon (Remote Desktop Protocol) |
| 11 | Cached credentials used to logon – system likely offline from DC |



Logon Events can give us very specific information regarding the nature of account authorizations on a system if we know where to look and how to decipher the data that we find. In addition to telling us the date, time, username, hostname, and success/failure status of a logon, we can also determine by exactly what means a logon was attempted. Logon Type Events are provided within the "Description" information of Logon Events and can give us this further information about how a logon was attempted. This information can be exceedingly helpful to an investigation, allowing us to determine whether actual "hands on keyboard" were used to logon or if the session was creating using remote means via a Server Messaging Block (SMB) protocol connection or by using something like Remote Desktop. Further, we can use this information to differentiate between system-based logons such as those performed by Scheduled Tasks, and user-based interactive logons.

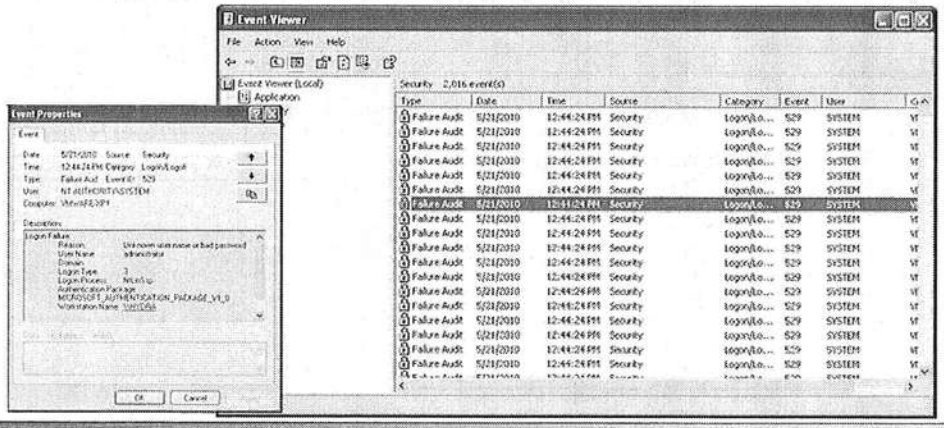
The following Logon Type Codes can be used [1]:
2: Logon via the physical console (using the keyboard)
3: Network logon (often using something like SMB for drive mapping)
4: Batch logon (Scheduled Tasks) – non-interactive
5: Windows Service Logon – non-interactive
7: Unlock of running screen saver application
8: Network logon sending credentials in cleartext (potentially indicative of a downgrade attack or older admin tool)
9: Different credentials used to authenticate than those currently logged on with (Runas command or similar)
10: Remote interactive logon (Terminal Services / Remote Desktop Protocol)
11: Cached credentials used to logon due to system not in communication with the Domain Controller

In addition to helping our investigations, knowing these Logon Types can also be an excellent way to audit security within your enterprise. For instance, if you see Logon Type 11 in your event logs, this indicates that cached credentials are being used within your environment. That may be perfectly acceptable for a laptop system, but on something like a server, this could be a dangerous security misstep that could allow an attacker to very quickly harvest additional accounts.

[1] http://www.windowsecurity.com/articles/Logon-Types.html

# Tracking Account Usage
# Brute Force Password Attack

**Evidence of a network based (Logon Type 3) password guessing attack on the Adminstrator account**




Core Windows Forensics - SANS ©2011

In this slide, we show another application for tracking account usage. In this case, our review of the event logs shows a large number of 529 Events, indicating a failed logon attempt. Looking at the Event Properties tells us that the failed attempts are for the Administrator account, and that they are being accomplished with Logon Type 3, or over the network. Given the Logon Type, this could be an attacker attempting to access network resources on the machine via SMB. The workstation name could give us a clue if the hostname were part of our enterprise (say another hacked machine), but in this case it only gives us "Hydra", which is likely provided by the brute force cracking tool of the same name. Event Logs starting with Windows 2003 now show both the hostname AND the IP address, providing a critical piece of information that was previously missing. A final piece of information we can discover from these logs is that by reviewing the timestamps we see 10s of failed logons occurring within the same second, indicating that this is almost certainly an automated attack.

# Built-in Service Accounts

In addition to explicit user account references, you will often see Windows System Accounts in event logs

| | |
|---|---|
| **SYSTEM** | Most powerful local account; unlimited access to system |
| **LOCAL SERVICE** | Limited privileges similar to authenticated user account; can only access network resources via null session |
| **NETWORK SERVICE** | Slightly higher privileges than LOCAL SERVICE; can access network resources similar to authenticated user account |
| **ANONYMOUS LOGON** | Null session w/o credentials used to authenticate with resource |

Core Windows Forensics - SANS ©2011

When reviewing Event Logs, it can be confusing or disconcerting to see users listed other than standard user account names. By definition, every process and service within Windows must run using an account with some associated security privileges. Since many Windows processes run before a user is even logged on (i.e. boot time), or without explicit user knowledge (system maintenance processes), Windows has a set of service accounts that can be used to allow processes to run within different security contexts. [1]

**SYSTEM:** Prior to Windows 2003, this was the primary account used for non-user related actions. The problem was that this is the most powerful account on the system and hence any process running under its authority that was subverted by an attacker would give absolute access to that attacker. Hence as part of the Microsoft Trustworthy Computing Initiative, more limited accounts were added which were more in line with the security principal of least privilege.

**LOCAL SERVICE:** Designed to have limited privileges (similar to the standard USER account) and used for services that do not require network access. Hence, it has no ability to authenticate with network resources and is relegated to using null sessions for network communications.

**NETWORK SERVICE**: Similar to LOCAL SERVICE, but with slightly higher privileges which allow it to impersonate standard computer accounts and authenticate over the network. It is assigned for processes or services which require network access.

**ANONYMOUS LOGON**: This account is the most often misunderstood. It was originally created as a means for Windows systems to communicate without requiring explicit credentials. In older or insecure systems anonymous logons, or Null Sessions, can be used to enumerate account information, security policy, registry data, and network shares. These abilities have slowly been removed (by default) in the newer operating systems such as XP, 2003, and now Win 7 / 2008. That being said, the account is still used by Windows networks do facilitate things like file and print sharing and maintaining the browse list. If these services are present in your

environment, there is a good chance that you will see Anonymous Logon usage. The key takeaway is to not automatically assume the system has been under some sort of enumeration attack. Further contextual information is usually needed to understand what an Anonymous Logon may mean.

The example shown in the slide shows several of these accounts being used during a system boot cycle. We can see LOCAL SERVICE and NETWORK SERVICE accounts successfully logging in to execute specific services as well as an ANONYMOUS LOGON connection, which in this case relates to the browse list (aka My Network Places). The ID 515 event run under the SYSTEM account is the initial Logon service starting to allow subsequent logons.

[1] http://technet.microsoft.com/en-us/library/cc875826.aspx

# Tracking Account Usage
# Remote Desktop Protocol (1)

**Scenario**

- Track Remote Desktop Protocol logons to target machines

**Relevant Event IDs**

- 682 – Session Connected / Reconnected
- 683 – Session Disconnected

**Investigative Notes**

- Event log provides hostname and IP address of remote machine making the connection
- On workstations you will often see current console session disconnected (683) followed by RDP connection (682)

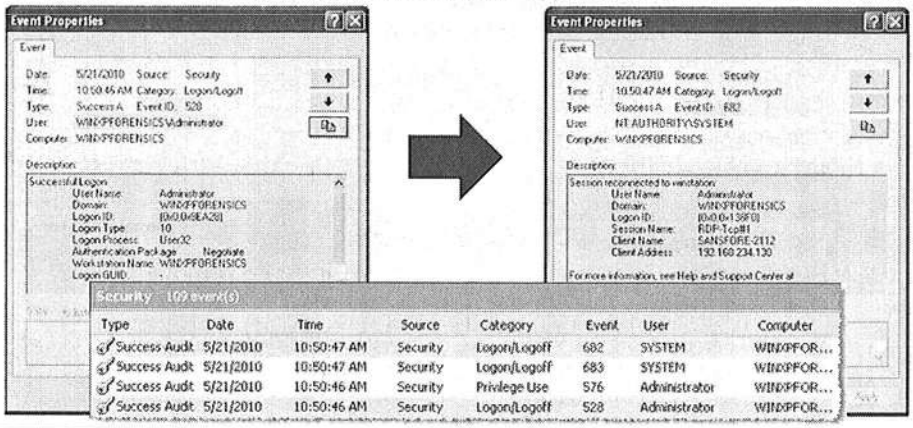


In addition to the standard logon Event ID codes, there are some specialized Event IDs that are specifically tied to the Remote Desktop Protocol (RDP). RDP is used extensively within many enterprises and as such is often also used by those will mal-intent. Why install sophisticated backdoors when a Domain Administrator account can allow you remote access to any system in the Active Directory Forest? When specifically looking for RDP connections, two Event IDs can make our lives much easier: ID 682 indicates that a RDP session was initiated and ID 683 indicates that a remote session was terminated. These two Event IDs in tandem help us bookend a complete RDP session.

One of the big advantages to IDs 682 and 683 is that they include the IP address and hostname of the system which established the connection. We should also expect to see a near-simultaneous ID 528 event (successful logon) because ID 682 only indicates a successful remote session was established, and ID 528 indicates that the credentials provided were accepted. The same goes for ID 683 and ID 538 (successful logout) events. To even further reinforce this idea of nearby events providing more context, we will also often see on workstations a ID 683 (session disconnected) event immediately before a ID 682 (session connected) event. This is due to Windows workstations only allowing one interactive logon at a time. When a RDP connection is made to a system that currently has a user logged into the console, the console session must be first disconnected.

# Tracking Account Usage
# Remote Desktop Protocol (2)

**Evidence of a Type 10 Logon (RDP) to WINXPFORENSICS using the local Administrator account from SANSFORE-2112 (192.168.234.130)**




Core Windows Forensics - SANS ©2011

This is an example of a successful Remote Desktop Protocol (RDP) session being established. Looking at the table view of the Security Event Log we see a successful logon (ID 528) following by a privilege assignation (ID 576) to the Administrator account on the machine WINXPFORENSICS. We can take a closer look at this event by looking at the properties, where we see that the event has a Logon Type 10 associated with it, which indicates a RDP session.

Subsequent to the Administrator logon, we see a session logoff (ID 683) which in this case happens to be the current session logged on at the console (physical screen/keyboard). This is immediately followed by a newly established session (ID 682) which is the new RDP connection. Looking at the properties for the latter event shows us that the RDP connection was initiated by a machine named SANSFORE-2112 at IP address 192.168.234.130.

This is an excellent example of using multiple events to paint the complete picture of an action taken on a machine. It is important to note that this RDP connection will only be logged on the receiving system. The system that initiated the RDP connection will not have these events logged (the only event log evidence would be in Process Tracking for the terminal services client being executed).

## Account Logon Events

- Different than ***Logon Event*** category
- Recorded on system that authenticated credentials
  - Local Account / Workgroup = on Workstation
  - Domain / Active Directory = on Domain Controller
- Event ID Codes (NTLM protocol)
  - 680: Successful / Failed account authentication
  - 681: Failed account authentication (NT / Win2000)
- Event ID Codes (Kerberos protocol)
  - 672: Ticket Granting Ticket was issued (successful logon)
  - 673: Service Ticket Issued (successful access to server resource)
  - 675: Pre-authentication failed (failed logon)

Core Windows Forensics - SANS ©2011

A big hurdle to understanding event log categories and audit policy is understanding the difference between *Logon Events* and *Account Logon Events*. Most of the confusion stems from the poor name choices. Logon Events refer to login / logoff activity that happens on the actual system being logged into. Thus they are stored locally on that end system. Account Logon Events refer to third-party authorization of credentials provided during that logon session. In a Windows Domain environment, the vast majority of user accounts are actually Domain Accounts, with their credentials stored on the Domain Controller NOT the local system. This is invisible to most users who use the same computer day in and day out. However, behind the scenes, before that user can log onto a workstation in a Domain environment, their username and password must be validated by the Domain Controller using either the NTLM or Kerberos authentication protocol. The Account Logon Events record this process and in this case would be stored on the Domain Controller that verified the credentials. Thus a single user logon can lead to several different events being spread across the workstation (Logon Events 528, 538, etc.) and across the Domain Controller (Account Logon Events 680, or 672, 673).

There is one crucial exception to all of this. If the user is logging in using a *local account*, an account created only on the workstation itself and not part of any Domain, then the workstation itself will be doing the final authentication (using the local SAM database) and hence we will see both Logon Events and Account Logon Events in the event logs of the workstation. Since this is quite rare in an enterprise, it is often an interesting artifact to look for since it could indicate rogue accounts that were created on the local system.

Since there are two possible authentication protocols, the Event ID Codes have been broken up into those used by NTLM and those used by Kerberos. For NTLM, both successful and failed events are recorded using ID 680, with the exception of Windows 2000 or before systems which had a separate ID (681), to designate failures.

Kerberos uses several unique Event ID codes, with the most commonly seen being 672 (successful logon), 675 (failed logon), and 673 (successful authentication to a server resource such as a file share).

# Account Logon Error Codes (NTLM)

Found in the Event Description for Event IDs 680/681

- Provide reason for failed NTLM protocol authentication
- Decimal used for NT / Win2000 and hex used for XP and above

| Hex | Decimal | Reason |
|---|---|---|
| 0xC0000064 | 3221225572 | Non-existent account username |
| 0xC000006A | 3221225578 | Incorrect password (username correct) |
| 0xC000006F | 3221225583 | Account not allowed due to day/time restrictions |
| 0xC0000070 | 3221225584 | Account not allowed to log on from this computer |
| 0xC0000071 | 3221225585 | Expired password |
| 0xC0000072 | 3221225586 | Disabled Account |
| 0xC0000193 | 3221225875 | Expired account |
| 0xC0000234 | 3221226036 | Account locked |

Core Windows Forensics - SANS ©2011

Whenever a failure event is recorded using Event ID 680 (XP and above), or Event ID 681 (NT / Win2000), an error code is generated and stored in the Event Description. The purpose of this code is to provide additional information to the administration as to why the credential authentication was denied. Error codes can provide a lot of information to the investigator regarding possible user actions. For instance, the existence of a locked account could be indicative of a password guessing attack (or just very strict password policies). Authentication attempt from restricted workstations (error code 0xC0000070), could indicate intent to access resources off-limits to that user, or large scale network enumeration. The error codes are as follows [1]:

**0xC0000064:** Non-existent account username

**0xC000006A:** Incorrect password (username correct)

**0xC000006F:** Account not allowed to log on at this time

**0xC0000070:** Account not allowed to log on from this computer

**0xC0000071:** Expired password

**0xC0000072:** Disabled Account

**0xC0000193:** Expired account

**0xC0000234:** Account locked

[1] http://technet.microsoft.com/en-us/library/bb742435.aspx

# Account Logon Error Codes (Kerberos)

Found in the Event Description for Event ID 675

- Provide reason for failed Kerberos authentication
- Over 40 codes – most common shown here
- Decimal used for Win2000 and hex used for XP and above

| Hex | Decimal | Reason |
|---|---|---|
| 0x6 | 6 | Invalid username |
| 0x7 | 7 | Requested server not found |
| 0xC | 12 | Policy restriction prohibited logon |
| 0x12 | 18 | Account locked, disabled, or expired |
| 0x17 | 23 | Password expired |
| 0x18 | 24 | Password invalid |
| 0x25 | 37 | Clock skew between machines is too great |



Beginning with Windows 2003 Server, the default authentication protocol in a Domain is Kerberos. Kerberos involves more steps than the NTLM protocol and creates more logs. In a nutshell, Kerberos works by the user first supplying credentials to the authentication server (often a Domain Controller). [1] The credentials are verified and if they are correct, a Ticket Granting Ticket (TGT) is issued to the user for a period of time. The TGT can then be used as a sort of passport allowing authentication via other domain controllers. To access resources on another system, such as a server, a separate ticket is requested which is called a Service Ticket.

If the Kerberos "pre-authentication" fails, an Event ID 675 will be written to the authentication server's log. In addition to providing information on date/time, hostname, client IP address, and supplied username, an Error Code will be included specifying the reason for the authentication failure. There are over 40 possible codes that can be issued [2]. We have included on this slide the most common error codes seen in Kerberos failure events:

**0x6**: Invalid / non-existent user account. This can also be caused by replication issues between Active Directory servers.

**0x7**: Requested server not found. This can also be caused by replication issues between Active Directory servers.

**0xC**: Policy restriction prohibited logon; client system restricted from accessing resource or restricted based on time / date

**0x12**: Account locked, disabled or expired

**0x17**: Expired password

**0x18**: Invalid password

**0x25**: Clock values between server and client are skewed too greatly; Kerberos relies on a timing system to invalidate old TGTs

[1] http://technet.microsoft.com/en-us/library/bb742516.aspx

[2] http://technet.microsoft.com/en-us/library/bb463166.aspx

# Finding a Rogue Local Account

Scenario: A local account named *root* was used to map a network share

- Event ID **680** indicates that the *root* account successfully authenticated from workstation SANSFORE2112
- Event ID **540** shows a successful network logon immediately following

Core Windows Forensics - SANS ©2011

Taking what we know about Account Logon Events, this scenario demonstrates how easy it is to find a potential rogue account via the Event Logs. This scenario assumes that local accounts (particularly privileged ones) are not routinely used to logon within the environment, which is a reasonable assumption in many enterprises.

Doing a quick filter of the event logs for Event ID 680 (Account Logon Success/Failure) identifies an entry on 5/26/2010 at 8:23:50PM on computer WINXPFORENSICS. Opening that event shows that the event was a Success Audit Event and that the logon account used was named *root*. Since the 680 Event was not located on the Domain Controller, this indicates that root is an account local to the system WINXPFORENSICS. Looking further at the 680 Event Properties, we see that the source workstation that successfully authenticated was named SANSFORE-2112. Now we have a successful remote access of our host system using a local account. Strange.... The next step is to review the events surrounding this anomalous 680 event for further information. An Event ID 540 is seen using the *root* account within the same second, indicating that the logon authentication was for a network resource such as a mapping a share on the WINXPFORENSICS system.

# Evidence of Malware Execution (1)

**Scenario**

- Evil doer is thought to have utilized malware to elevate privileges. Identify potential malware and determine if it was executed.

**Relevant Event IDs**

- 592 (SecLog) - New process created (includes executable path)
- System Event Log (look for Warning and Error events)
- Application Event Log (look for Warning and Error events)

**Investigative Notes**

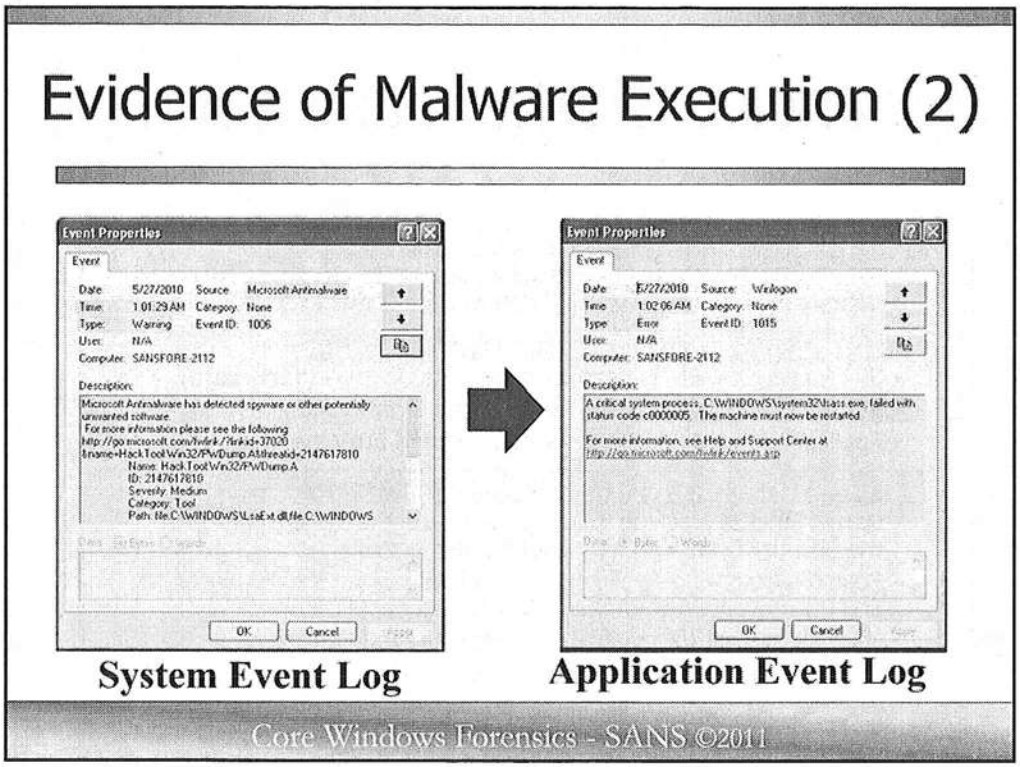
- Unless Process Tracking is being audited on the system (rare), the Security Event Log may not be much help
- Search System and Application event logs looking for Warning and Error events from A/V or other security applications
- Keep an eye out for crashed processes and reboots



In this scenario we will be looking at Event Logs to try to identify any malware that was executed on the system. In a situation like this, we will often rely more upon the System and Application Logs than the Security log. While it is true that the Security Event Log does have a capability to record every process that is executed via Process Tracking, in practice this is usually not audited because of the massive amount of logging that it creates. If you reference security baselines for auditing, you will see that Process Tracking is left disabled in all but the most critical of systems. Hence, we will be looking to the System and Application logs to help us.

Event IDs are well documented and can be very helpful when working within the Security Event Log. However, they are unfortunately next to useless when doing many System and Application log reviews. Events IDs for these logs are poorly documented and a single ID can have a myriad of different meanings. In light of this, our recommendation is to look for most critical events in these logs, namely the Error and Warning events. While there can be a large number of these events present in the log, most are duplicates that get logged every hour, every boot cycle, etc. Instead, we will be looking for the outliers, and in this scenario, we will be especially looking for anti-virus or anti-malware product warnings that may have identified suspicious activity on the system. We would also be looking for unexplained reboots or crashed system processes, since malware is a dirty business and crashes occur at least as often as successful compromises.

Once we find evidence of malware, we can then mine the information from the events (such as the alert information, process name, or executable path) to quickly search event logs on other systems.



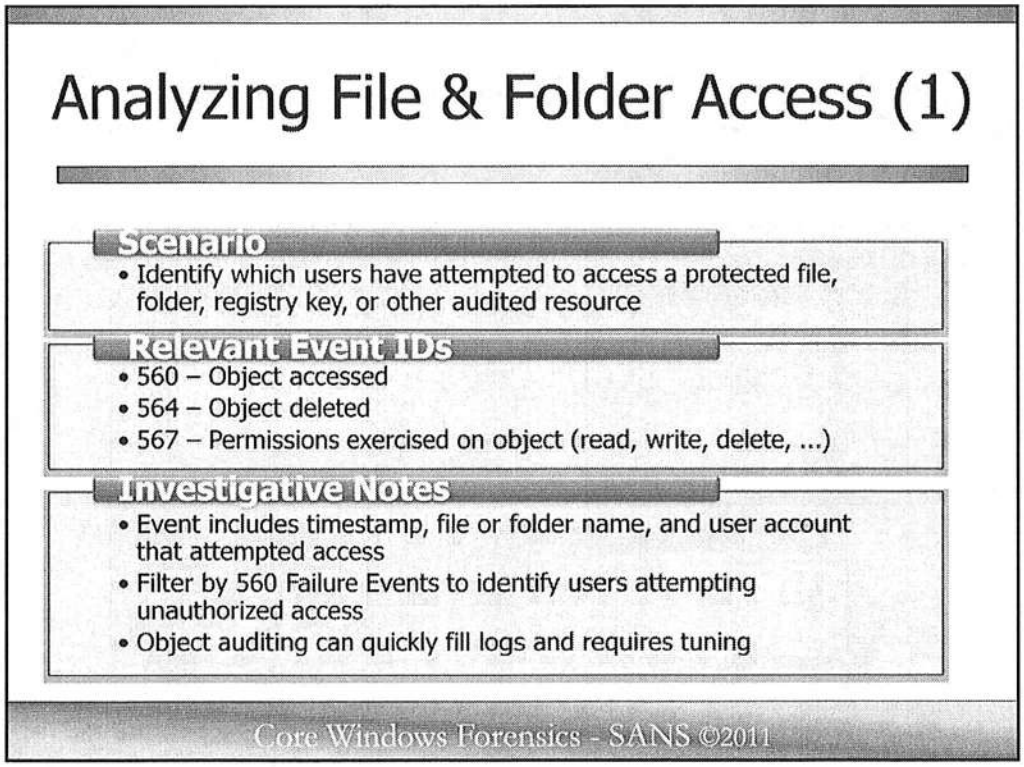

This is a great example of finding evidence in logs other than the Security Event Log. While most of our security related events are located in there, keep in mind that the operating system and applications running on the system may have security mechanisms in place that only log to the System or Application logs.

In this scenario, an attacker attempted to run the PWDump hacking tool remotely against a Windows XP system. The PWDump tool works by mounting a share, copying malware to the system, executing the malware to perform DLL process injection into the Local Security Authority Subsystem Service (LSASS) process and using the now co-opted LSASS process to dump the password hashes for the system.

A very strong clue that something bad happened on the system comes from the System Event Log. We see that the Microsoft Antimalware service (Security Essentials Antivirus) alerted on a tool named PWDump at 1:01:29 AM. The antivirus real-time protection likely matched a signature on the LsaExt.dll file that was copied over to the local system for execution. Approximately thirty seconds later, we find an entry in the Application Event Log indicating that the LSASS process crashed and that the system was preparing for an immediate reboot. In both of these cases, the Event IDs are irrelevant. We found these entries due to them being Warning and Error events, respectively. That being said, Microsoft does have a resource for looking up Event IDs, though it rarely provides much help. [1]

Since we know that PWDump mounts a share to copy its malware, a final piece of this log review would be to look in the Security Event Log for 540 Network Logon events around this time, which could give us the compromised account used and the hostname of the system that launched the attack.

[1] http://www.microsoft.com/technet/support/ee/ee_advanced.aspx




Utilizing event logging to monitor key files, folders and network shares is a powerful feature that few administrators take the time to implement. Object Access Events can quickly fill a security log if they are implemented hastily. When done correctly, they can give the analyst very detailed information about when critical resources are accessed, by what user accounts, and what actions were performed on them. With companies focusing more on intellectual property theft, you may find yourself lucky enough to have these events available during your investigations.

Object Access Events are triggered based on resources which have been previously set up to audit. This can be accomplished through the Auditing options under the Advanced Security Options of a resource. For instance, <right-click> on a file → Select Properties → Security Tab → Advanced Button → Auditing Tab. Options include auditing access, read, write, deletes, etc. Children objects can be set to inherit these options from their parent folder. [1]

For auditing access, we will focus on three Event IDs within the Security Event Log. ID 560 indicates that a resource was accessed and can be either a Success or Failure Event. [2] Along with this information, we are also provided with the timestamp, user account responsible, and the name or path of the resource accessed. In XP systems and later, this will be paired with ID 567 event which gives specific information about what specific actions were taken on the object. A variety of actions are available, but those most pertinent to us are Read, Write, and Delete. These actions will be listed in the Description section of the 567 Event, under the property *Access Mask*. Finally, when auditing deleted resources, we can use Event ID 564 to indicate when audited resources have been deleted. When found, this event will be paired along with an associated 560 and 567 event as well.

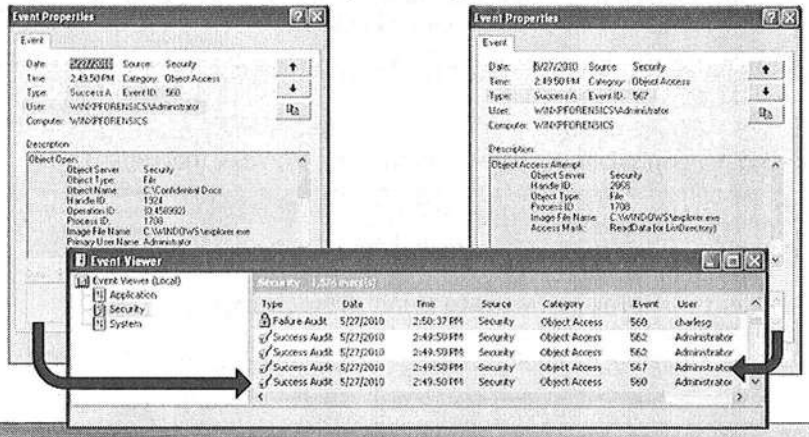
An interesting implementation of this could be to set "write" auditing on system files in the Windows and System32 directories. These files should legitimately only change with Service Pack updates, hot fixes, etc. If an intruder has attempted to modify these files, it would be relatively easy to identify the activity.

[1] http://technet.microsoft.com/en-us/library/bb727008.aspx

[2] http://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=560

# Analyzing File & Folder Access (2)

**Evidence of Administrator account accessing and listing the contents of the C:\Confidential Docs folder; similarly user charlesg fails to access**




Core Windows Forensics - SANS ©2011

In this example, we are reviewing the Security Event Logs, checking access to a folder named "Confidential Docs". The permissions for this folder are that only users within the Administrator group should have access. The Event Viewer shows several successful events and one failed event. The events begin with an ID 560 event showing that the user *Administrator* successfully accessed the folder C:\Confidential Docs at 2:49:50Pm on 5/27/2010. Additionally we can see by reviewing the Image File Name property within the Event Description that the folder was accessed via Windows Explorer (explorer.exe).

The ID 560 event is followed by an ID 567 event. This event should tell us exactly what permissions were exercised by the user account *Administrator*. In this case, the *Access Mask* shows that the "List Directory" permission was used.

Finally, at the top of the Event Viewer table, we see an ID 560 failure event. This tells us that on 5/27/2010 at 2:50:37PM, the user *charlesg* attempted to access a resource that it did not have permissions for. If we were to open this event, we would see that the resource was C:\Confidential Docs. Failed object access does not lead to a corresponding ID 567 event which requires actual access. Further research into the *charlesg* account would indicate that it was not part of the Administrators group, explaining why access was denied to that resource.

[redacted]

# Suspicious Services (1)

**Scenario**

- Analyze logs for suspicious services running at boot time
- Review services started or stopped around the time of a suspected compromise

**Relevant Event IDs**

- 7034 – Service crashed unexpectedly
- 7035 – Service sent a Start / Stop control
- 7036 – Service started or stopped
- 7040 – Start type changed (Boot | On Request | Disabled)

**Investigative Notes**

- All Event IDs reference the **System Log**
- A large amount of malware and worms in the wild utilize Services
- Services started on boot illustrate persistence (desirable in malware)
- Services can crash due to attacks like process injection



Windows Services control almost every element of the operating system. A service is a process that runs without user interaction and regardless of whether a user is logged in. [1] They are a common target for malware and can provide excellent clues as to malicious behavior on the system you are investigating. Events related to services are located in the System Log, and hence all Event IDs discussed in this section will be for System Log Events.

Services are governed by the Service Control Manager (SCM). It transmits control requests to running services and drivers and maintains status information about those services. The SCM is responsible for updating the System Log with service-related events. When looking for suspicious services, four System Log events are the most useful [2]:

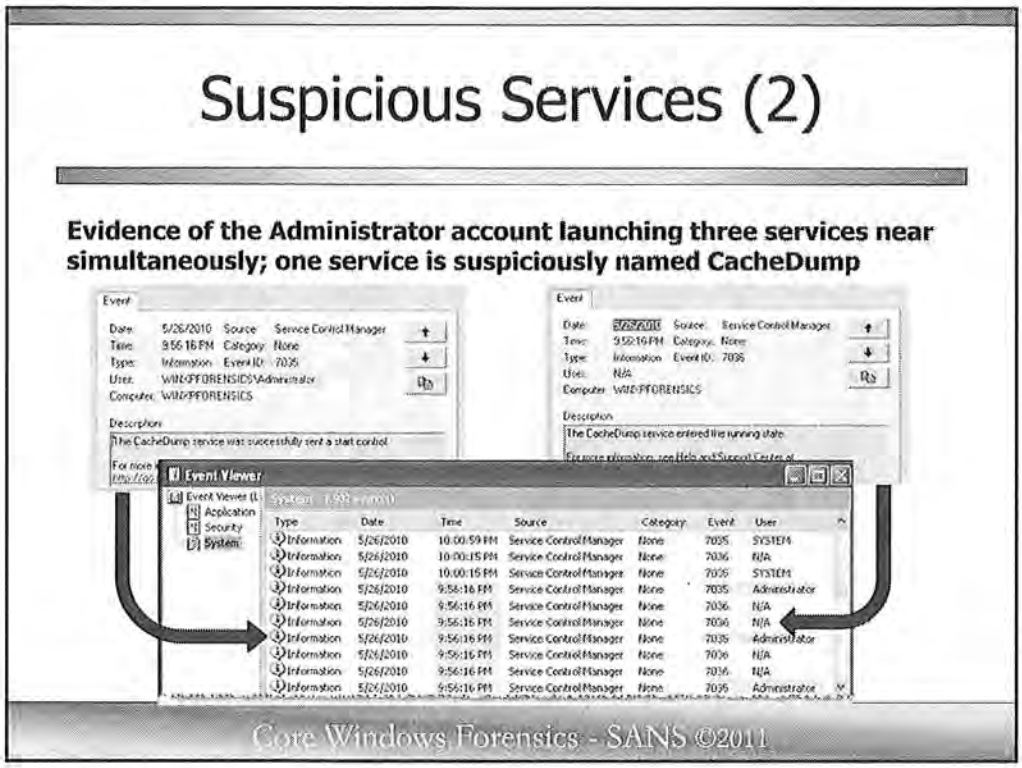
**7034**: Service crashed unexpectedly – Services should only crash on rare occasions, thus this is an interesting anomaly that may be worth investigating. The event will also indicate how many time the service has crashed previously. Sophisticated attacks like process / dll injection have been known to crash services

**7035**: Service sent a Start / Stop control – The SCM has ordered the service to either launch or shutdown. A 7036 Event will indicate when this actually happens.

**7036**; Service started or stopped – Indicates that the service is either fully operational, or completely shutdown

**7040**: Start type changed (Boot | On Request | Disabled) – Reports on any changes to the Start Type of a service. Services that are marked as Boot will start at boot time. Boot time starts are of particular interest because they can be a vector for malware to survive a system reboot. On Request indicates that a process must request the SCM to start the service and Disabled indicates that a service will not start even when sent a control by the SCM.

[1] http://technet.microsoft.com/en-us/library/aa998749%28EXCHG.65%29.aspx
[2] http://technet.microsoft.com/en-us/library/cc756379%28WS.10%29.aspx




This event log example shows the execution of the fgdump password hash dumping tool on a Windows XP system. [1] Assuming that we didn't have a good starting point, like a time of attack from an IDS sensor, one of the things that should stick out immediately in this System Log, is that we have several services that are being sent controls by a non-SYSTEM account (in this case it is the Administrator account). This is an irregular occurrence, made even more so by the fact that they were all sent within the same second.

To find out what the Administrator account was up to, we would need to review the Event Properties. In this slide we are showing two of the suspicious events. On the left, we have an event with ID 7035 indicating that this is a command to the Service Control Manager to, in this case, start the CacheDump service. The subsequent event with Event ID 7036 (on the right), informs us that CacheDump service successfully started. A simple Google search for "CachedDump Service" quickly yields multiple articles about password hash dumping tools. [redacted]

One note here is that we may not always encounter an obvious name like CacheDump. Malware will often choose a name that appears to be related to legitimate system functions. This makes it more difficult to immediately recognize as a malicious process, but there are usually other clues like we saw in the example. Most notably the services being started by the Administrator account and so many controls being sent together. As a counter-example, the Conficker worm created and started a randomly named service that was relatively easy to identify. [2]

[1] http://www.foofus.net/fizzgig/fgdump/

[2] http://support.microsoft.com/kb/962007

# Application Installation (1)

**Scenario**

- Review logs to identify evil or unauthorized software installations
- Track uninstalled software for dangerous apps previously present
- Identify failed installation attempts

**Relevant Event IDs**

- 1033 – Installation completed (with success / failure status)
- 1034 – Application removal completed (with success / failure status)
- 11707 – Installation completed successfully
- 11708 – Installation operation failed
- 11724 – Application removal completed successfully

**Investigative Notes**

- All Event IDs reference the **Application Log**
- Events only logged when applications use Windows Installer API



Knowing which applications were installed on a system in a given period of time can be an important piece of evidence. The Application Log stores this information. Installation events are triggered by the Windows Installer application. Some of the more useful items we can mine from these logs are what applications were installed successfully, what application installations failed (perhaps a key component of the application was missing on the system), and finally what applications were uninstalled. For each of these scenarios, the event logs will record timestamp information and the user account that attempted the install/uninstall. A key point is that event logs will only be present for applications which utilized the Windows Installer application.

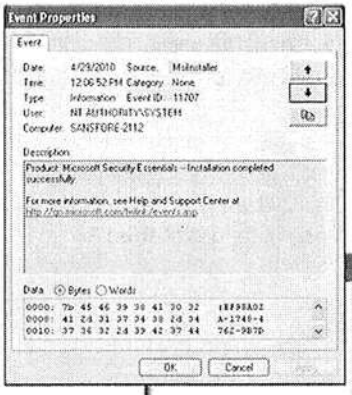
Event IDs related to applications installation and removal can be tricky. The reason is that there are two different mechanisms that can potentially log to the Application Log. The Windows Installer itself has Event IDs 1001-1038 allocated to log its status. [1] Additionally, the application itself can return an "Error Code" at the completion of an install or removal. There are hundreds of potential error codes and hence to keep them unique within the log, their final Event ID is <Error Code + 10000>. [2] Thus all installer events logged in this fashion will have IDs > 11101. In some cases both entities will trigger events, resulting in both a 1033 and a 11707 event logged for the same successful installation. Having a good reference for the Event IDs and error codes is essential, and two excellent ones are referenced on this slide.

One essential difference with the standard Windows Installer events listed here (IDs 1033 and 1034) is that unlike their counterparts (IDs 11707, 11709, and 11724) they do not imply success or failure. Instead they just imply that the operation was "complete". The success or failure of the operation is communicated via a Status field within the Event Description.

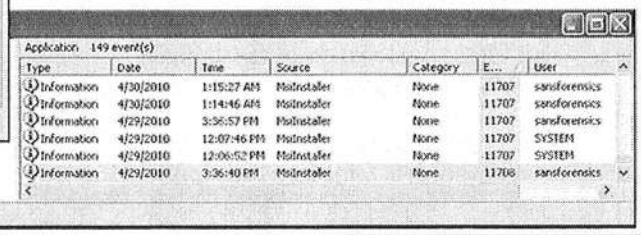
Unfortunately no logs are created for users who fail to install an application due to insufficient account privileges (i.e. not having administrator rights). Windows does have a feature called Software Restriction Policy that can be enacted via Group Policy that will log this type of failure, but it is rarely implemented.

[1] http://msdn.microsoft.com/en-us/library/aa368560%28VS.85%29.aspx
[2] http://msdn.microsoft.com/en-us/library/aa372835%28v=VS.85%29.aspx

# Application Installation (2)



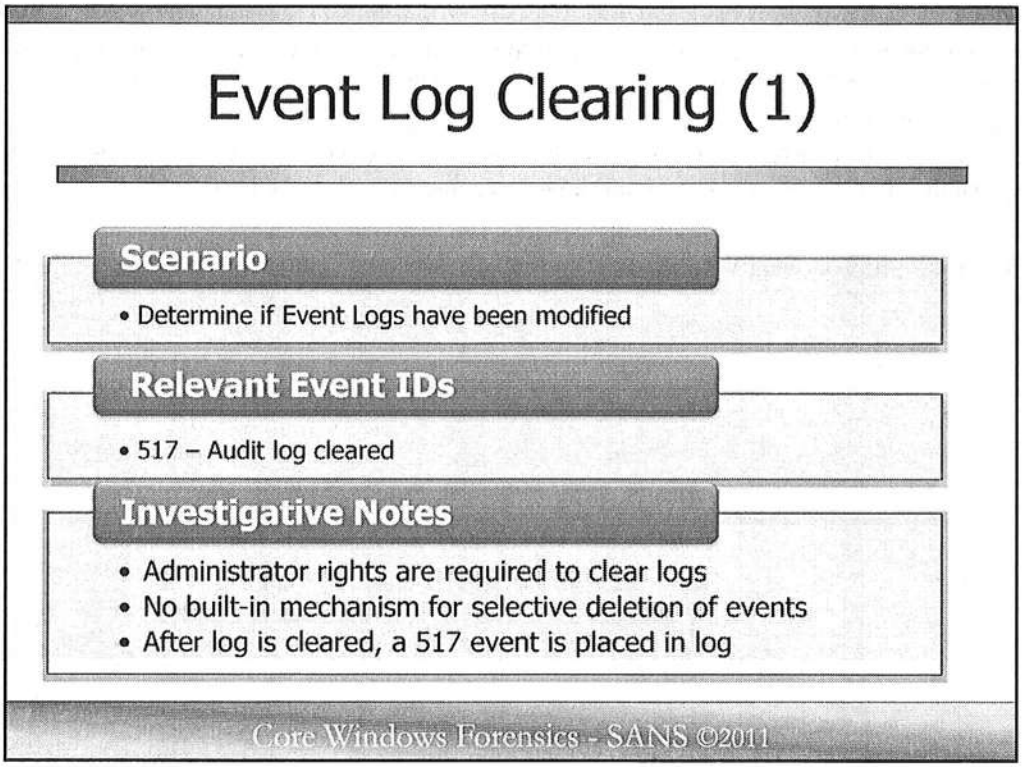

- Evidence of the Security Essentials Application installed by the SYSTEM account
- A large amount of activity occurred on 4/29-4/30
- The sansforensics account also installed apps
- One failed installation (11708) is also present




Core Windows Forensics - SANS ©2011

One of the easiest methods to quickly scan application installations is to sort the Event Viewer by Event ID and review the 1000+ and 11000+ events, looking for anomalies. You should be looking for installations and un-installations that were accomplished at strange times and dates (or are within a specific window that you are investigating) and that were performed by user accounts which are relevant to the case.

In this example we show evidence of several applications installed (Event ID 11707) over the period of 4/29-4/30. Additionally, there is one failed installation (Event ID 11708) that was initiated by the sansforensics account. To look deeper at one of the events, we open the Event Properties dialog, where we can see that the SYSTEM account successfully installed Microsoft Security Essentials on 4/29/2010 at 12:06:52PM. Potentially more interesting than this would be if the Security Essentials was un-installed, potentially showing a user trying to degrade the security posture of the system.




One topic that invariably comes up when discussing Event Log review is that of log clearing. Local Administrators, Domain Administrators and the local SYSTEM account all have the privileges to clear event logs. In many cases we investigating a user that has administrator rights (either legitimately or illegitimately) and as such may have the capability of covering their tracks. Surprisingly, this does not happen very often, which may be a consequence of the fact that it is very difficult to clear event logs without a trace. Whenever an administrator clears a log, an ID 517 event is placed in the log to indicate that a clear took place. Thus even if we do not know what the logs were, we at a minimum know that a specific user account cleared the logs which can be evidentiary in itself, particularly when working with cases where data spoliation is of interest. By reviewing 517 entries, we can see when the logs were last cleared an ensure they match with known legitimate administrator activities and are in line with data retention a security policies.

Log clearing is built to be all-or-nothing. Individual records should be able to be selectively removed. However, there has been proof of concept code released of a hacking tool capable of selectively deleted evens within a live log. The tool is named WinZapper and was for NT and Win2000 systems. [1] It never worked that well and was just as likely to crash the Event Log service or render the logs corrupted as it was to be successful. It turns out that the Event Logs are well protected by the operating system, with one reason being that they are always in use and are highly intolerant of log corruption. That being said, you should assume that a knowledgeable user with full access to a machine *could* have the capability to modify logs – one option is to do so directly in memory before they are written. While there are no documented tools or cases of this happening in modern Windows systems, anything is possible (though not necessarily probable) when someone fully owns a system.

An additional item to keep in mind is that logs may be missing due to the Event Log Service itself overwriting older events. There are three options when the maximum size of a log is reached:

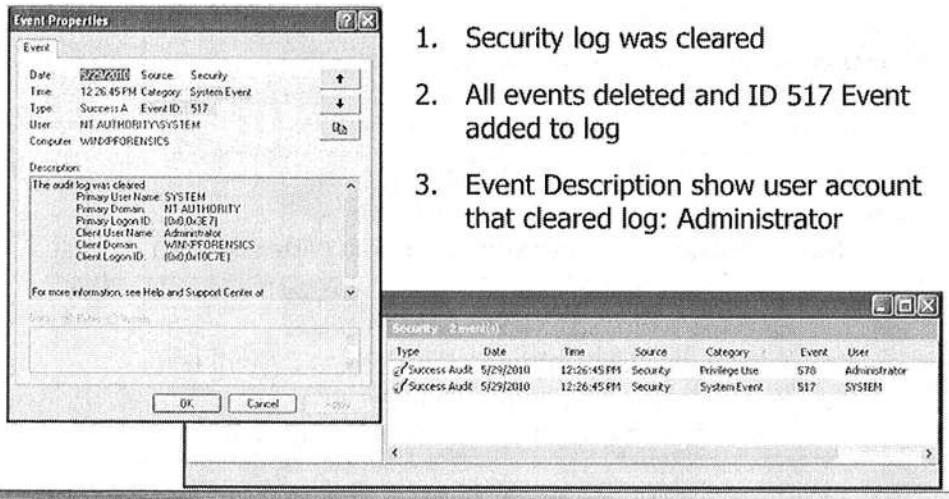
**Overwrite Events As Needed:** Events in the log are overwritten when the maximum file size is reached.

**Overwrite Events Older Than X Days:** When the log is full, events are overwritten only if they are older than the setting selected. If the maximum size is reached and the events can't be overwritten, error messages are generated.

**Do Not Overwrite Events (Default):** When the max log size is reached, error messages are generated to indicate that the event log is full. The administrator must manually clear the logs.

[1] http://en.wikipedia.org/wiki/Winzapper

# Event Log Clearing (2)




1. Security log was cleared
2. All events deleted and ID 517 Event added to log
3. Event Description show user account that cleared log: Administrator

Core Windows Forensics - SANS ©2011

When an event log has been cleared, the first event in the log chronologically will be an ID 517 System Event indicating that the log was cleared. This of course implies that the system has the correct auditing policy in place to audit successful System Events. Taking a look at the ID 517 Event Properties, we see that we have confirmation that the audit log was cleared and we have a date and time of when it occurred. Within the Description section, the Client User Name will be displayed, indicating the user account that exercised its privileges to clear the log. In this example, it was the Administrator account.

Looking at the event table in this example, we also see a success audit event ID 578. This system was also auditing Privilege Use, and thus when the event logs were cleared, the Administrator account had to exercise its SeSecurityPrivilege, which gives the account the rights to change the audit policy AND clear the event logs.

# Unauthorized Hardware Devices (1)

**Scenario**

- Determine if and when hardware devices have been installed on the system

**Relevant Event IDs**

- 20001 – Plug and Play driver install attempted (Vista and later)

**Investigative Notes**

- New logging capability introduced with Vista System Log
- Will identify device type and serial number
- Only shows *first* time a device has been plugged in



With the new update to Event Logging within Vista, a large number of new entities started triggering events for the various logs. One of these additions is that the Universal Plug and Play manager is now logging events and it creates events for each device driver that is installed. Thus we can use these events to determine when a Plug and Play device was plugged in for the first time, since by definition the Plug and Play Manager will attempt to install a device driver when a new device is detected.

When a Plug and Play driver install is attempted, the service will log an ID 20001 event and provide a Status within the event. If the status is anything other than "0", an error was reported during the installation. The installation error codes have not changed since XP and can be found on the Microsoft website. [1]

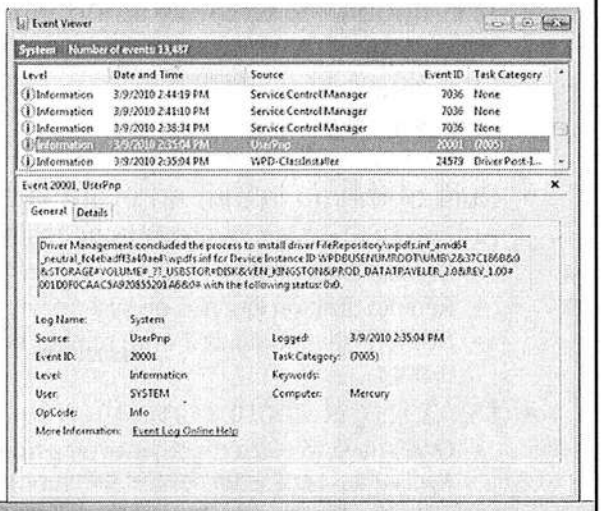
The ID 20001 event gives the particulars of the device installed, including the device name and vendor name and the serial number of the device (if it exists). The user account name that was logged in when the device was introduced is not stored in the event log. This information would need to be correlated with logon events or discovered via the Windows Registry.

It is important to note that this event will trigger for any Plug and Play capable device, including but not limited to USB, Firewire, and PCMICIA devices.

[1] http://support.microsoft.com/kb/310123

# Unauthorized Hardware Devices (2)

- Vista System Log
- Event ID 20001
- Timestamp
- Device information
- Device serial num
- Status (0 = no errors)






This example demonstrates the new Event Viewer re-designed for Vista (and included in every OS distribution since). The event table and properties section are now integrated, making review easier. In this case, we are reviewing the System Log and have selected an ID 20001 event triggered by the UserPnp service (Plug and Play). This event tells us the following:

Timestamp: The time and date that the device driver installation was attempted (3/9/2010 2:35:04PM)

Device Information: Embedded device information captured by the Plug and Play Manager (Kingston Datatraveler 2.0 Revision 1.0)

Device Serial number: The unique serial number embedded in the device (001D0F0CAAC5A920855201A6)

Status: The error code associated with the device installation (0x0 = no errors)

Thus doing a simple search for ID 20001 events allows us to quickly view any devices that were plugged into the system and recognized by the Plug and Play Manager. Using the timestamp, it would be trivial to correlate this event with logon events to determine which user account was logged on when the device was plugged in.

# Locating Log Evidence

- It is important to understand where logs are recorded
  - Domain Controllers
  - System logged into
  - Remote system or server
- Rule of thumb is that an event is triggered wherever authentication or system processing took place
  - Domain credential authentication = on Domain Controller
  - Remote desktop logon = on system being logged into
  - Network share mounted = on remote system where shared folder resides
- Don't forget about correlating log events!
  - Often have to piece together events from Security, System, and Application Logs from several different systems

Core Windows Forensics - SANS ©2011

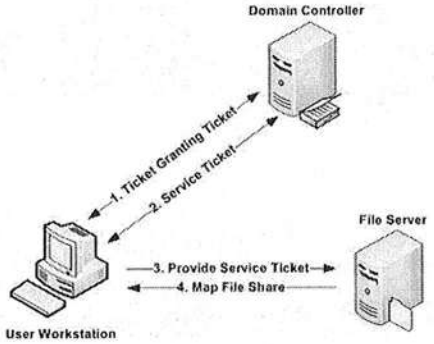
Reviewing event logs can become overwhelming at times, but with practice they can provide a massive amount of information about how systems were used and interacted with one another. Keep in mind that it is just as important to understand where you can possibly retrieve logs as it is to understand what individual events indicate. Just because your investigation centers on a user's workstation does not mean that there isn't evidence external to that computer image that can assist your investigation. In cases of employee misuse, intellectual property theft, and computer intrusions, we will often need to look at network data to help piece the puzzle back together. Event logs will often exist on the workstation, the Domain Controller(s), and remote systems that were accessed by the workstation. This is especially important when we consider that logs may not be recovered from every location. Different systems may have different audit policies and hence what we find in one set of event logs may be our only indication that a particular action occurred.

When trying to determine what events you should expect to find logged on a system, a good thing to remember is that events are triggered by processes. Thus we would expect to find relevant events wherever a given activity was actually processed. When local user accounts are utilized, authentication only takes place on the workstation, and hence the event logs would be located there. Domain accounts require an external entity – the Domain Controller – to perform authentication, and thus we find event logs on both the workstation AND on the Domain Controller. If a system is accessed via Remote Desktop, we would expect to find logon events in the logs on the remote system.

Finally, don't forget that even on a single system we are rarely just reviewing one log file! Correlation of events found via multiple logs is challenging, but it is absolutely necessary. You will often need to piece together information using Application, System, and Security Logs from many different systems.

# Example: Locating Log Evidence

**Scenario: User logs in with a domain account (via Kerberos) and maps a network file share**




Events Recorded:

- Workstation
  - 528
- Domain Controller
  - 672
  - 673 (Client)
  - 673 (Domain Controller)
  - 540
  - 538
  - 673 (File Server)
- File Server
  - 540
  - 538

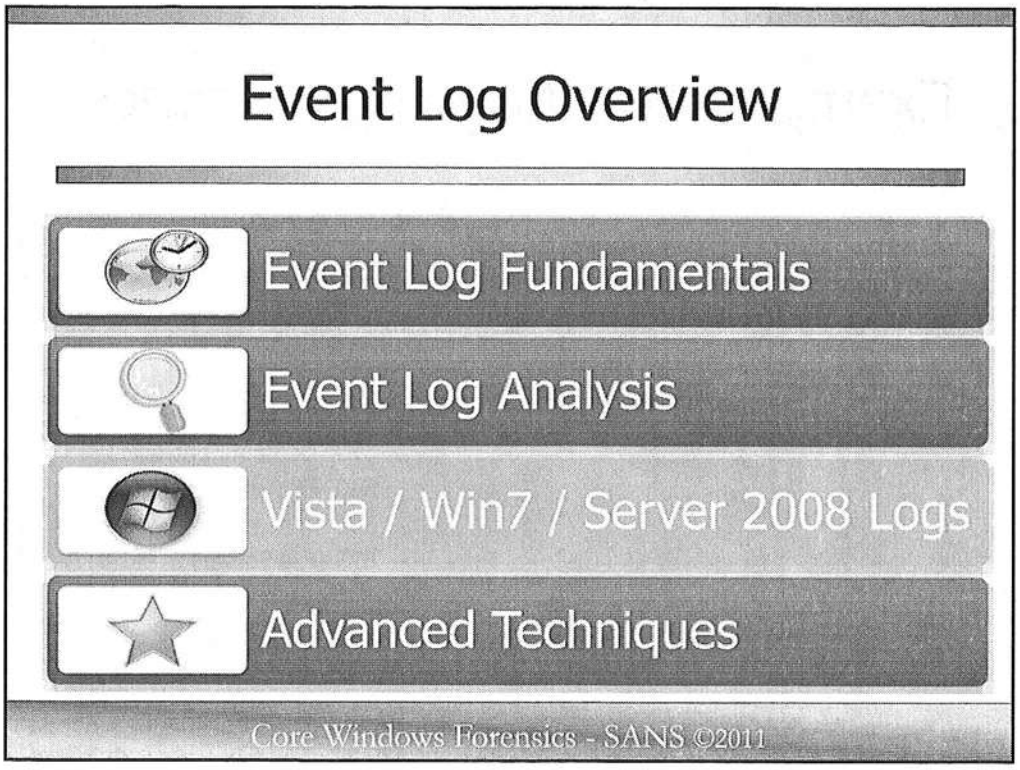


This is a great example of the "embarrassment of riches" that we sometimes have when good auditing policies are in place. Very commonly a simple action by a user can cause 10 or more event log entries to by recorded throughout the network. In this case we have a Windows 2003 enterprise environment utilizing Kerberos for authentication. The following steps occurred:

1. A user logged into the workstation using their domain account
2. Upon logon, a folder was mapped on a network file server

This simple process results in at least nine events being recorded throughout the three systems involved.

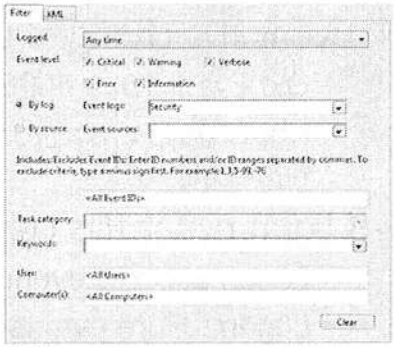
When reviewing Event Logs in a networked environment it is very important to understand all the location where logs may reside. In this example very few of the actual events were recorded on the workstation itself. To get the full picture of activities (and when they occurred) it is necessary to recover logs from multiple systems.




This page intentionally left blank.

# .evtx Log Format

- Memory efficiencies
  - Less costly to log
- XML and filtering
- Improved messaging
  - IP addresses
  - EventIDs changed
- Expanded number of event logs
  - Increased granularity of audit controls




Core Windows Forensics - SANS ©2011

With Windows Vista came a replacement to the "Windows Event Log", named "Event Logging". The new logging service is vastly better and solves some of the very real pain points complained about since the early days of Windows NT. One of the biggest pain points was memory management. In previous versions, the entire log file had to be mapped into memory, meaning up to 300MB (the maximum event log size) of memory might have to be allocated just for event logs. [1] This had a very real impact on performance and hence was an excellent justification for administrators to turn off logging. Log files in Vista and later now consist of a small header followed by self-contained 64KB chunks. Now only the current chunk is required to be located in memory, with obvious performance benefits. This turns out to be a big boon for forensic investigators since it increases the likelihood of logging being enabled.

The format of event logs also changed. Logs now carry the .EVTX file extension and are stored in XML format, making log deconstruction much more intuitive and facilitating the use of industry standards like X-Path filtering to create much more powerful searches and filters. That being said, logs are still stored in binary form and raw string searches will miss much of the information in these files.

Another pain point for analysts was the cryptic messaging provided in the old Event Logs. Microsoft increased the readability of the log messaging. They now include IP addresses in addition to hostnames when applicable (this was also added to later versions of Win2003), and they massaged the Event IDs, collapsing some series and adding new event types. It remains to be seen whether the new Event IDs will be better documented.

The number of logs has greatly increased, allowing for specialized logs to be used instead of filling the standard Application, System, and Security logs. With new logs comes new events, giving us a better view than ever before into many of the running processes on the system like Task Scheduler and Plug and Play. Finally, Vista includes more audit options, making the previous options less "all or nothing" and allowing audit policies to be intelligently crafted.

[1] http://technet.microsoft.com/en-us/library/cc722385%28WS.10%29.aspx

## .evtx Event ID Changes

- Event IDs changed with .evtx logs
  - Old Event ID + 4096 = New Event ID
  - Not always a one to one mapping – more of a heuristic
- Logon event IDs collapsed
  - Logon success events 528 & 540 → ID 4624
  - Logon failure events 529-537, 539 → ID 4625
- Format is not backward compatible
  - Always use same OS or newer to review logs



For old school event log enthusiasts, the new .evtx logs include one serious problem: all of the Event IDs have changed. Because the format changed so radically, Microsoft developers felt that there needed to be a very clear differentiation between pre-Vista and post-Vista events. Luckily, they have kept a relationship between the "old" version of event IDs and the new events. In simplest terms, for many .evtx events, you can subtract 4096 to get the Event ID in the pre-Vista world. Why 4096? Event IDs are stored in hex and they chose to distinguish the new events by adding 0x1000 (4096 in decimal). [1]

There is a point where this simple formula breaks down. Some of the new events are not exactly the same and give subtly different information. In instances like this, the developers have stated that they tried to keep the spirit of the new event the same. Also, many of the old events no longer exist in the new Event Logging. One of the more radical changes was the 10 different event IDs used to describe various Logon failure events (529-537, 539) being collapsed into one event (ID 4625). With the new event, the specific failure type is listed in the Event Description.

A final note is that the format is not backward compatible. In addition to the new structure of the data, there are many new elements that are now part of an event log record (IP addresses are one of these). Thus, evtx files cannot be read using pre-Vista Event Viewers. Forward compatibility has also been documented to cause some problems. Microsoft recommends that when reviewing log files from multiple different Windows versions, to use the newest version of the OS Event Viewer to convert the .evt format to a .evtx format. [2] While it is better than no solution, it does introduce a level of complexity when reviewing logs from disparate machines involved in the same incident. Not to mention the mental gymnastics of keeping two different sets of Event IDs in your head!!

[1] http://blogs.msdn.com/ericfitz/archive/2007/04/18/vista-security-events-get-noticed.aspx

[2] http://blogs.technet.com/askperf/archive/2007/10/12/windows-vista-and-exported-event-log-files.aspx

# New Event Logs

- .evtx logs are stored in a new folder: `%systemroot%\winevt\logs`
- In addition to Application, System and Security, we now have many more logs to potentially review:

| | |
|---|---|
| **Setup** | • Records installation and update information on all applications |
| **Forwarded Events** | • Repository for events retrieved from other systems |
| **Applications and Services** | • Contains over 60 logs in a standard Windows 7 installation<br>• Useful logs include Task Scheduler, Windows Firewall, and Windows Defender |

Name
Microsoft-Windows-HomeGroup Provi
Microsoft-Windows-PrintService%4Adm
Microsoft-Windows-Diagnosis-DPS%4O
Microsoft-Windows-Resource-Exhaustio
Microsoft-Windows-NetworkProfile%4O
Microsoft-Windows-Windows Firewall
Microsoft-Windows-DriverFrameworks-
Microsoft-Windows-TerminalServices-L
Microsoft-Windows-BranchCacheSMB
Microsoft-Windows-User Profile Service
Microsoft-Windows-GroupPolicy%4Op
Microsoft-Windows-Kernel-EventTracin
Application
Microsoft-Windows-Kernel-WHEA%4O
Microsoft-Windows-OfflineFiles%4Ope
Security
System
OSession
Microsoft-Windows-UAC-FileVirtualiz
Microsoft-Windows-Audio%4Operation
Microsoft-Windows-Resource-Exhaustio

Core Windows Forensics - SANS ©2011

As if the new .evtx log formation didn't change enough, now we have a new default location where event logs are stored. You will find event logs in the `%systemroot%\winevt\log` folder on Vista and later systems. The primary reason for providing event logs with their own directory, is that instead of just three logs, you will find over 60 logs in a standard Windows 7 install! While having that many logs to review is a daunting thought, it turns out to be a good thing for the forensic analyst. For one, more logs mean a greater likelihood that more information will be stored. We already saw one example of that in regard to Plug and Play installation logging. It also means that logs will be increasingly dedicated to a specific purpose, giving one-stop shopping for things like scheduled tasks, the Windows Firewall, and the Windows Defender anti-malware solution. And of course keep in mind that many of the 60+ logs are unused on a typical system. You can tell this immediately by sorting the folder by file size.

The new logs can be broken up into a few categories

**Setup:** This is a new log that is intended to be a close companion to the Application, System, and Security log. It attempts to preserve installation information for every application on the system. This includes updates to applications, and is an excellent location to review to determine what Windows security updates, patches, and hotfixes have been added to the system.

**Forwarded Events:** This log is part of a new, much needed feature: the ability to consolidate logs from multiple machines on a "collector" system. If you are reviewing logs on such a system, you will see those logs sent from other systems present in the Forwarded Events log.

**Applications and Services:** This comprises all of the new "custom" logs introduced in Vista. A majority of th logs are found in the Microsoft folder.

# Event Log Subscriptions

- Allows logs from multiple systems to be aggregated by a collector machine
  - Local system may not have most complete logs
- Filters can be applied before retrieving events
- Two entities "Forwarder" and "Subscriber"
- Logs sent via Windows Remote Management
  - To determine if a system may be forwarding logs look to see if Remote Management service is started: `SYSTEM\CurrentControlSet\Services\WinRM\Start`
- Retrieved events stored in Forwarded Events Log



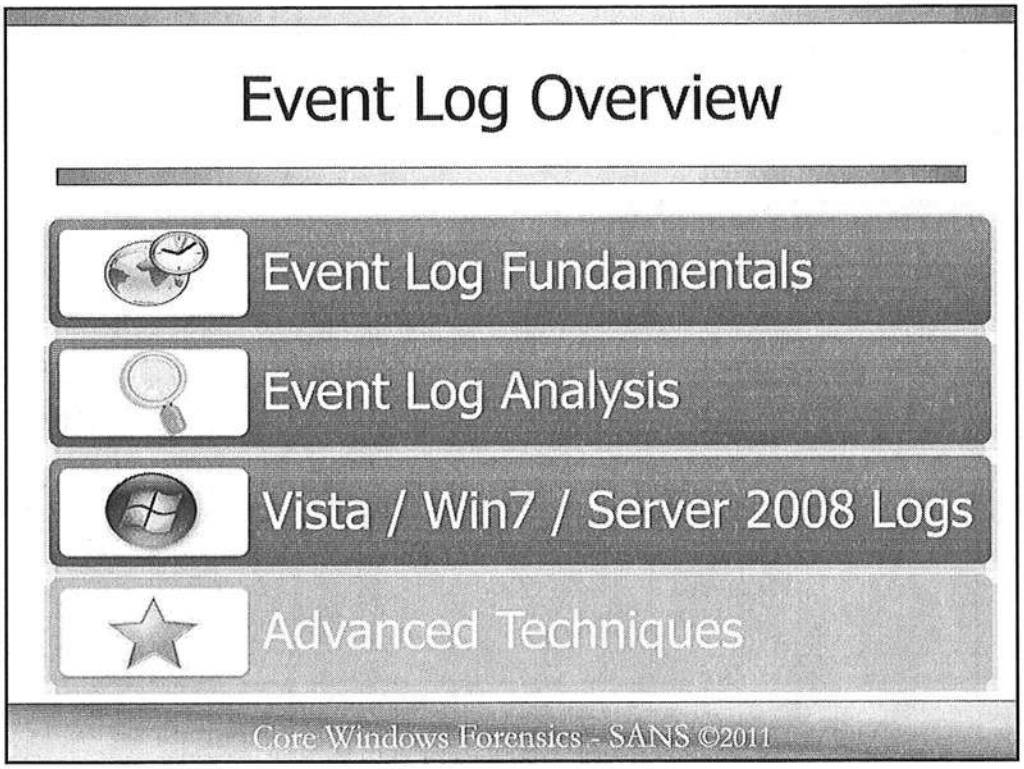
Event log subscriptions fill a large gap in log management prior to Windows Vista (note: Windows Server 2003 R2 now includes this functionality as well). Previously the only default method for reviewing event logs was on the local machine. In a large scale incident, this can become very cumbersome and thus third party tools like SNARE were built to allow event logs to be sent to remote servers. Event log subscriptions now allow 2003 / Vista / Win7 and 2008 machines (Forwarders) to provide event log access to a machine that is a central collection repository (Subscriber). This functionality is implemented using the Windows Remote Management Service, and hence one way to determine if a system is set up to provide logs is to look for this service being started. Looking in the registry, `SYSTEM\CurrentControlSet\Services\WinRM\Start` will be set to the value of "2" if it is enabled.

If this functionality has been configured, it can be utilized by a forensic investigator to quickly collect logs of interest from many machines. A filtering capability is built into the service, allowing fine grained control over what records are retrieved from each machine.

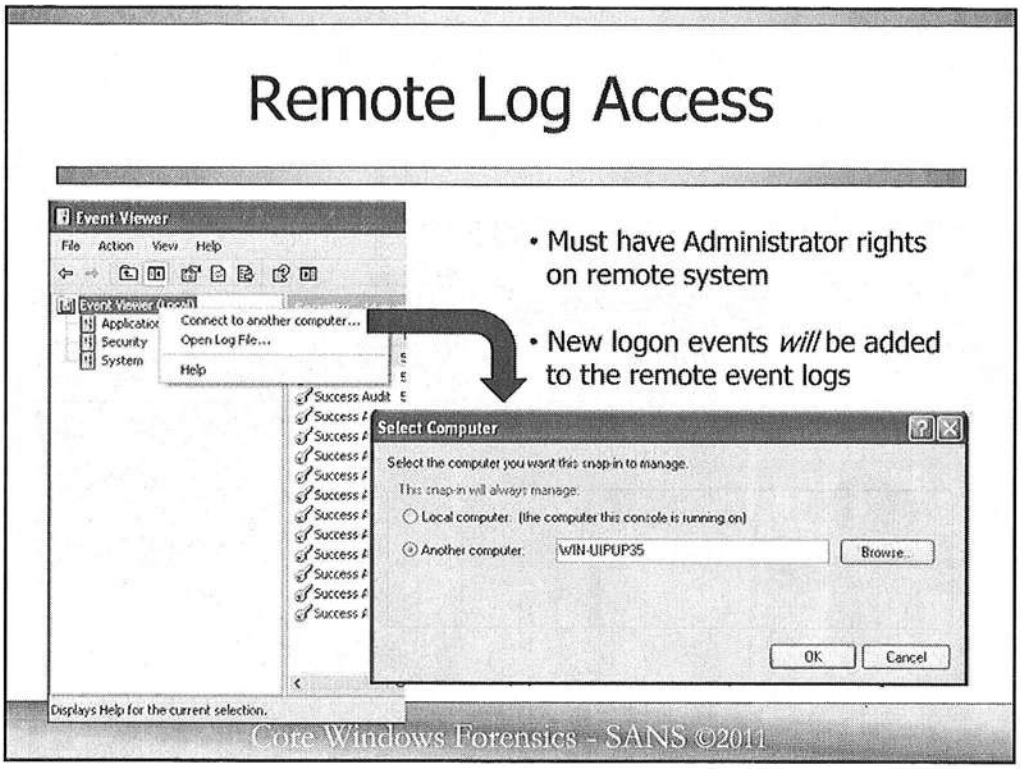
While event logs can be retrieved either through HTTP or HTTPS, both methods employ encryption in their default settings (use of HTTP requires Kerberos encryption) [1]. However, there are options to downgrade the protection level during installation, effectively allowing event logs to be sent in clear text, opening up the possibility of modification in-transit [2].

[1] Charter, Brandon. EVTX and Windows Event Logging, GCFA Gold Paper, 2008.

[2] http://blogs.dirteam.com/blogs/sanderberkouwer/archive/2008/02/23/remotely-managing-your-server-core-using-winrm-and-winrs.aspx




This page intentionally left blank.




One of the more interesting features of the Event Viewer is the capability to view events on other machines within the network. This provides a faster alternative to using Remote Desktop and can be much more efficient than sitting at the console or exporting the Event Logs individually. While it would not necessarily be recommended for a complete log review, it can be very useful in live response applications where something very specific needs to be checked on a system with great urgency. As an example, you could be interested in whether a specific known compromised account logged on to the remote system, or whether a piece of malware was identified or a specific Service was started. Having a means to quickly answer critical questions makes you even more flexible as an incident responder and forensic professional.

To view logs on another system, you need to <right-click> on the "Event Viewer (Local)" item in tree view and select "Connect to Another Computer". Select "Another Computer" either via IP Address or Hostname (or click "Browse" to view Network Neighborhood machines) and click "OK".

Your account must be a member of the Administrators group on the remote system, or alternatively, part of the Domain Admins group. The Vista Event Viewer now allows a different account to be specified (similar to "runas"), but this functionality is not available on pre-Vista systems. A simple workaround is to mount the Admin$ share of the remote system before accessing the remote logs. This can be done via the Net Use command:

```
Net use * \\<IP address>\Admin$ /u:<administrator account>
```

Keep in mind that you will be viewing the logs on a live system, and as such all of your activities will be recorded as new events on that system. This of course includes the remote logon required to view the logs in the first place. So you may see new 540, 680 / 673, and 578 events in the logs (depending upon network authentication type and auditing policy).

While this capability works for all systems, Windows NT through Server 2008, it is a best practice to view different event log types on like systems (i.e. remote .evtx logs should be viewed using a Vista or later system, and vice versa). It is also worth noting that this feature is only a subset of the "Remote Management" capability natives to Windows systems. It is possible to accomplish the same thing by using the Computer Management Console to connect to a remote system.

# Extracting Event Logs

## Exporting from a live system

- Exporting from Event Viewer
  - Multiple formats: .evt, .evtx, .csv, .xml, .txt
- Use FTK Imager to copy locked files
  - High potential to corrupt .evt logs
- PsLogList tool (Sysinternals)

## Offline extraction

- Offline log parsers can read binary logs and output text
  - Lsevt.exe for .evt logs (Carvey)
  - Evtx_parser (Schuster)



When performing Event Log analysis, a very common technique used is to export the logs to be reviewed on your forensic workstation. This is done for a couple of different reasons. First, you are often working with multiple logs (i.e. System and Security) from multiple systems (i.e. Workstation and Domain Controller) and it is usually easier to have the logs all in one place when doing your review. Additionally, logs are often exported because most forensic tools do a poor job of parsing Event Logs. As an example FTK 1.81 will not parse log files and FTK 3.1 will only parse .evt files. To be honest, this is not much of a limitation, because logs can be extensive and we would almost always prefer to review them in a dedicated event viewing tool, whether that be the Microsoft Event Viewer, or in a third-party solution.

Exporting Online Logs

Luckily we have several options for exporting logs out of both live and offline systems. When working with a live system it is important to keep in mind that Event Logs are always in use and hence locked by the Operating System. This presents a little bit of challenge. One of the best options for live exports is using the Event Viewer itself. Right-clicking on the event log of interest will give you the option to "Save log". Logs can be saved in a variety of formats including native (.evt or .evtx), .csv, .xml and .txt. This is the safest option when you are concerned with possible log corruption.

Another alternative for live extraction is to use FTK imager to mount the Physical Disk and export the event logs. Given that the Event Log format is very finicky, and the files are always in use, this very often results in a corrupted log. While this usually does not prevent us from reviewing the events, it could preclude that log from being opened using the Microsoft Event Viewer which is very strict about the clean/dirty status of the event database. With the new .evtx format, log corruption happens much more rarely since much less of the log file is kept resident in memory.

A final excellent alternative for extracting logs on a live system is the free PsLogList tool from Sysinternals. This is a command line tool with many features. It can dump live logs to a text or .csv file, read and output exported event logs in their native .evt/.evtx format, pre-filter output, and even dump event logs from remote systems.

Exporting Offline Logs

Event logs are databases stored in binary form, so if you have a .evt or .evtx file that you need to review, a tool that understands that format is necessary. A couple of excellent free tools have been released which take Event Logs in native form and convert them to text so they can be analyzed with virtually any tool. Since these tools parse the individual data structures, they have the very important ability to read and output data from corrupted Event Logs. Harlan Carvey wrote the popular lsevt tool for .evt logs (two other versions exist with additional features: lsevt2 and lsevt3). Andreas Schuster is a well known researcher in .evtx logs and he wrote a freely available parser called evtx_parser. [2]

[1] http://windowsir.blogspot.com/2006_02_01_archive.html

[2] http://computer.forensikblog.de/en/2010/02/evtx_parser_1_0_3.html

# Event Viewer Export vs. Lsevt.exe

**Security Event Log Exported using Event Viewer**

securitylog - WordPad

```
5/31/2010   1:19:41 AM  Security    Failure Audit    Logon/Logoff      529
      NT AUTHORITY\SYSTEM     WINXPFORENSICS     "Logon Failure:
      Reason:            Unknown user name or bad password
      User Name:  Administrator
      Domain:
      Logon Type: 2
      Logon Process:     Advapi
      Authentication Package: MICROSOFT_AUTHENTICATION_PACKAGE_V1_0
      Workstation Name: WINXPFORENSICS"
```

**Lsevt.exe output for Security Event Log**

Command Prompt

```
C:\>lsevt e:\Export\SecEvent.evt

Record Number : 1229
Source        : Security
Computer Name : WINXPFORENSICS
Event ID      : 529
Event Type    : EVENTLOG_AUDIT_FAILURE
Time Generated: Mon May 31 07:19:41 2010
Time Written  : Mon May 31 07:19:41 2010
SID           : 01 01 00 00 00 00 00 05 12 00 00 00
Message Str   : Administrator

2
Advapi
MICROSOFT_AUTHENTICATION_PACKAGE_V1_0
WINXPFORENSICS
```

**Lsevt.exe** parses event data in raw format:

- Timestamp = UTC
- Message String includes
  - User Name
  - Logon Type
  - Workstation Name
- Reason not included
- SID only in lsevt output
- Record Number only in lsevt output



In this slide we are contrasting the output from two sources: a text based export from the Microsoft Event Viewer and the output from the Lsevt.exe tool. These tools differ in a couple of ways, which necessarily impact the types of output we should expect to see. The output from the Event Viewer was pulled from a live system and its output is almost identical to what you would see if using the actual Event Viewer (as we would expect). The lsevt.exe output was performed on an extracted and corrupted version of the same log in an offline manner. As such, the lsevt output does not have the luxury of using the Event Logging Service and has parsed the data in its raw form. Thus we see several differences:

**Timestamp:** lsevt shows timestamps in UTC time, as they are stored within the binary file. The Event Viewer output is shown in Local System Time.

**Message String:** lsevt provides all the same data as the live export, but is not able to provide the pretty labels to go along with them. Hence, you must really know what you are looking for to identify the User Name (Administrator), Logon Type (2), and the Workstation Name (WINXPFORENSICS).

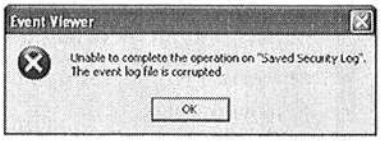
**Reason:** lsevt cannot display the Reason field, because this is not stored in the Event Log. Instead, it is provided at runtime by a .dll file, allowing easy language internationalization.

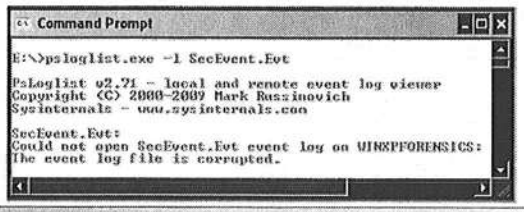
**SID:** lsevt does an excellent job of displaying all of the information it can find, including the SID embedded into the event record. While this is not a useful SID in this exanoke, in some events it can be very helpful. Unfortunately post-processing needs to be performed on the SID since it is stored in Little Endian hexadecimal.

**Record Number:** The is the identification number of that event within the Event Log Database. Interestingly, record numbers are not always sequential with respect to the timestamps because the Event Log is implemented as a circular buffer.

# Corrupted Event Logs

- Examiners often encounter corrupted logs
  - Particularly with the .evt format
- Standard Microsoft tools fail on corrupted logs
  - Third party tools tend to be much more tolerant
- Log corruption can be manually repaired
- Worst case: use an offline parser to convert to .csv file




Core Windows Forensics - SANS ©2011

The .evt Event Log format is very brittle and hence encountering corrupted Event Logs is more common than you might think. Log files can be extremely large (up to 300MB), are always open, always mapped into memory, are very frequently accessed by the system, and are very fault intolerant. This is a recipe for failure, and as such we have been living with corrupted logs since the early days of NT. Forensic techniques like "pulling the plug" and exporting live, locked files using tools like FTK Imager also greatly increase the potential that you will encounter a corrupted log file. The .evtx format has addressed many of these issues and as such is much more robust, but it not immune.

Microsoft's guidance on how to deal with corrupted logs leaves a little to be desired: it says to just clear the log and start over. [1] We can do better than that, and while it isn't ideal, finding corrupted logs turns out to not be that much of an inconvenience. The biggest impact is that the built-in Microsoft tools like Event Viewer and in some cases PsLogList (see slide images) will not work. Since many forensic analysts don't use those tools anyway, the impact is minimal. In fact, many administrators and forensic investigators have turned to third-party tools for just this reason. Tools like Event Log Explorer usually have very little problems parsing corrupted logs, and provide better features than the Event Viewer.

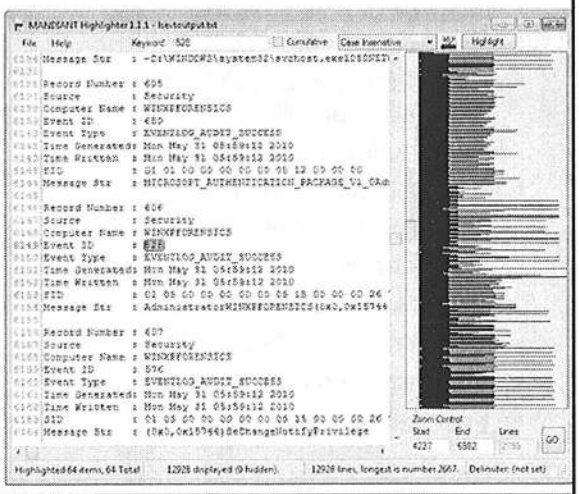
It doesn't take much to corrupt a log file, and thus the fix is usually just as easy. The most common problems are syncing issues or problems with the header or footer. It is beyond the scope of this class, but there are several resources online that can walk you through an Event Log repair. Steve Bunting has some excellent information on his site. [2] However, it is rarely necessary to do this because we almost always have tools that will "ignore" the corruption and give us the data we need.

The good news is that several offline Event Log parsers exist which are capable of mining data out of even the most damaged log. The reason for their success is that they are looking through the log file for very specific data structures, and when they find one, they output its associated data. By performing a raw export, these tools effectively ignore most of the peculiarities of the format that cause corruption. Lslnk.exe and evtx_parser are two of the best offline parsing tools available.

[1] http://technet.microsoft.com/en-us/library/bb457024.aspx
[2] http://www.stevebunting.org/udpd4n6/forensics/repaireventlogfile.htm

# Log Review with Mandiant Highlighter

- Excellent text review tool
- Reads very large input files
- Histogram shows text distribution within file
- Graphical highlighting
  - Helps distinguish patterns
- Strong filtering capability
  - Can remove unwanted lines and duplicates from viewer
  - Alternatively, show only lines with specific keywords
- Plot events by date/time
  - Quickly identify spikes and lulls in activity



Core Windows Forensics - SANS ©2011

Mandiant Highlighter provides an amazing array of features not found in any other tool. [1] It was specifically written to aid in the review of very large log files. One of its most noticeable features is the histogram located on the right side. This shows a graphical representation of text distribution throughout the file. It is particularly helpful when a very large or very small entry may indicate something of significance (for instance, very long input strings on a SQL Server experiencing SQL injection attacks). You can use the histogram to zoom into any area of the file and any keyword highlights are also displayed within the histogram. This allows visual patterns to be distinguished based on keyword frequency.

As the name suggests, Highlighter does a very good job of finding keywords, either individually or by using an input file. You can choose different colors to be used for each keyword. It also gives a hit count, telling you how many times it found a given keyword. Highlighter allows the analyst to perform data reduction by selecting a piece of text and having the tool remove all lines that include that text. Alternatively, you can only show lines with that text, allowing very quick comparison between like entries.

Of particular interest with logs is the ability to plot entries on a timeline. You must first "teach" highlighter what timestamps you want it to graphically represent, and then it will plot a timeline showing the data density according to date/time. This allows patterns to be viewed and helps identify spikes or lulls in activity.

Highlighter is particularly well suited for review of event log output from tools like the evtx_parser. Often analysts will use output in .csv format, forcing all event entries to be on one line to better take advantage of the "remove" and "show only" capabilities built into Highlighter. Best of all, it is a free tool!

[1] http://www.mandiant.com/products/free_software/highlighter/

# Event Log Explorer

- Reads both .evt and .evtx formats
- Can open up multiple logs at once for simultaneous searching and correlation activities
  - Including remote Event Logs!
- Very tolerant of log corruption
  - Standard and direct mode
- Excellent filtering
  - Including by Event Description
- Color coding by Event IDs
- Supports logs from NT – 2008
- Free for personal use

Core Windows Forensics - SANS ©2011

Event Log Explorer is a third-party event log management software package that runs circles around the built-in Microsoft tools. [1] It provides just about every feature that a forensic analyst could want when doing a log review and greatly speeds up the process. It supports logs from every current Windows NT operating system (Windows NT to Windows 2008) and as such can read both .evt and .evtx log formats. One of its biggest benefits is that it is very capable of working with corrupted log files. You can open up a log file in one of two ways: Standard Mode attempts to open up the log using API-like methods and Direct Mode parses the log in raw form. Thus we get the benefits of an offline parser tool combined with a GUI interface and review platform.

Another area where Event Log Explorer shines is in its features that assist log review. It allows many log files to be opened simultaneously, greatly aiding with event correlation and reducing the amount of time needed to search. This ability extends to being able to open logs on multiple ***remote*** systems simultaneously for live reviews. It has a very robust filtering capability, including access to the text-based Event Description field where so many of our forensic artifacts are located (such as Logon Type in ID 528 Security Events). Another nice feature is the ability to color code different Event IDs so you can quickly zoom into the places within the log where you want to focus you analysis.

Event Log Explorer is free for personal, non-commercial use, and is approximately $99 for a full license.

[1] http://www.eventlogxp.com/

This page intentionally left blank.

## Windows Firewall Log

- Connection information per machine
- Monitors blocked packets
  - TCP
  - UDP
- Outbound successes are not logged
- Outbound traffic that is not blocked is not logged

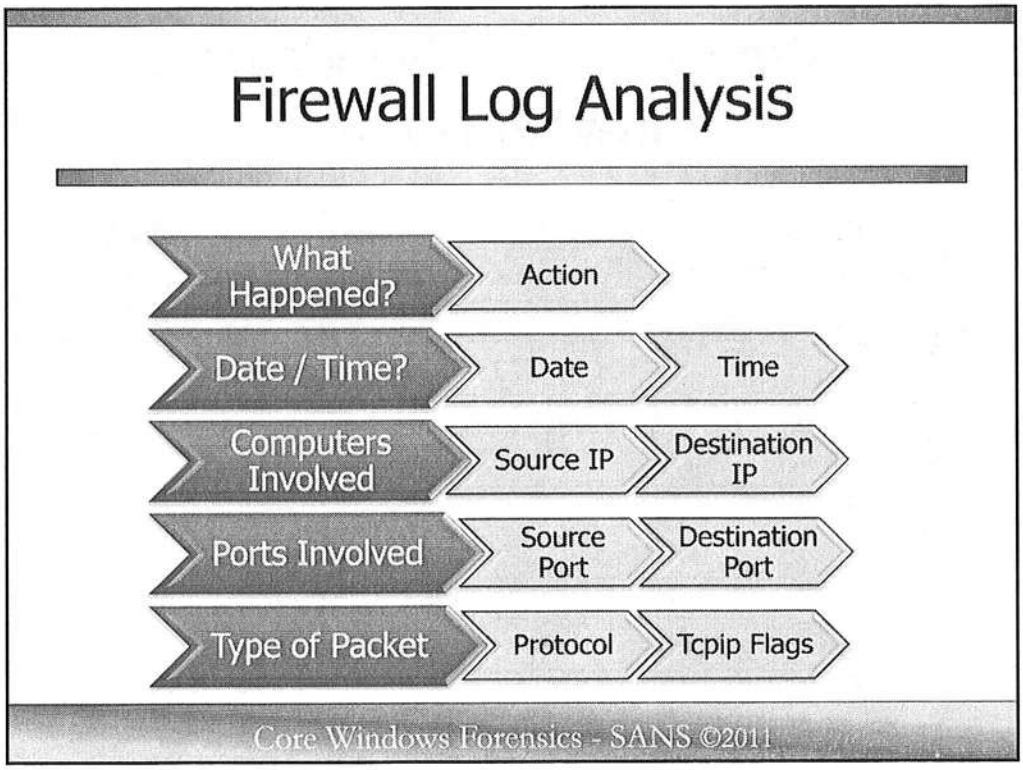
Core Windows Forensics - SANS ©2011

Windows Firewall is a built-in, host-based, stateful firewall that is included in Windows Vista, Windows Server 2008, Windows XP with Service Pack 2 and later, and Windows Server 2003 with Service Pack 1 and later. Windows Firewall drops incoming traffic that does not correspond to either traffic sent in response to a request of the computer (solicited traffic) or unsolicited traffic that has been specified as allowed (excepted traffic). Windows Firewall helps provide protection from malicious users and programs that rely on unsolicited incoming traffic to attack computers. In Windows Vista and Windows Server 2008, Windows Firewall can also drop outgoing traffic and is configured using the Windows Firewall with Advanced Security snap-in, which integrates rules for both firewall behavior and traffic protection with Internet Protocol security (IPsec).[1]

A great source of information that might exist on each Windows machine is the monitor TCP and UDP connections and packets that are blocked by Windows Firewall. The log file provides source and destination IP addresses, port numbers, and protocols.

By default, the log file is disabled, which means that no information is written to the log file. To use the log file, you must enable the logging of dropped packets, successful connections, or both. You can also configure other log file settings, including the name, location, and maximum size of the log file.

[1] http://technet.microsoft.com/en-us/network/bb545423.aspx




Firewall logs, if they exist on your machine, could provide you some interesting data as to when network connections were attempted on your machine. One of the major gripes of the firewall log is that it is challenging to map the packet back to the owning application that initiated the connection. The log file is not set up by default. Any investigator reading this should consider setting a group policy to set up global firewall log settings so it will exist on the majority of machines on your network. This will help produce a forensic rich environment. Despite the drawbacks and some of the most common things you can examine from the Windows Firewall are:

What Happened? Indicates the operation that was observed by the firewall. The options available to the firewall are OPEN, CLOSE, DROP, and INFO-EVENTS-LOST. An INFO-EVENTS-LOST action indicates the number of events that occurred but were not recorded in the log.

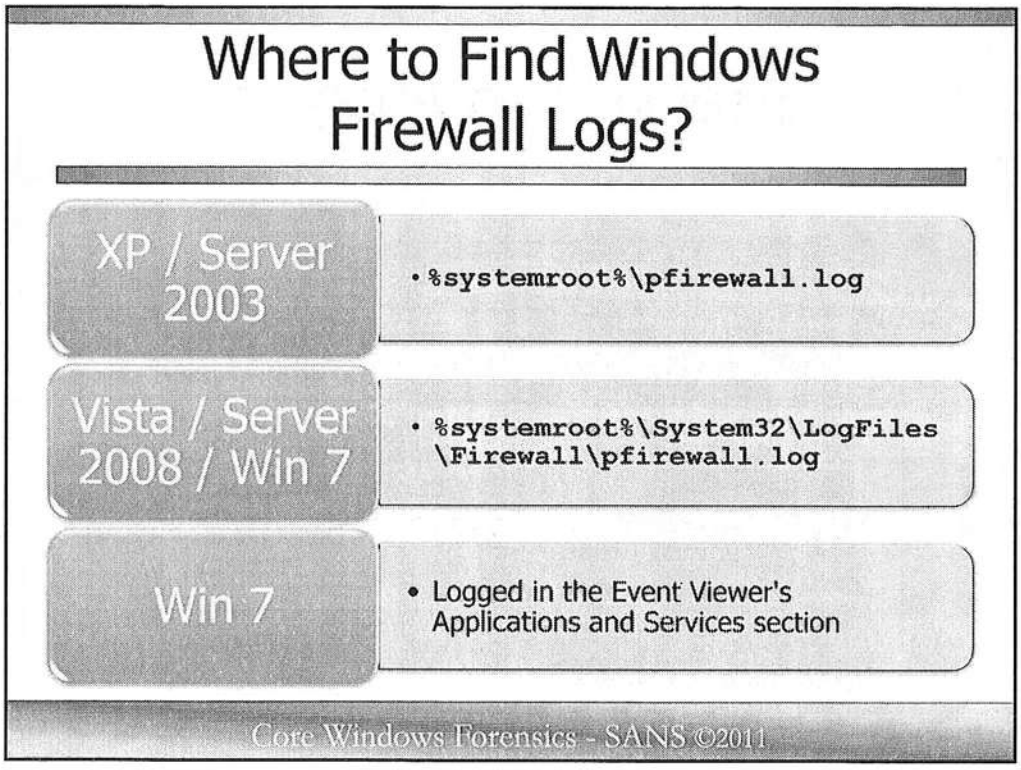
Date/Time? Timestamps are the key to examining firewall logs.

Computers Involved? Being able to examine both the source and the destination IP addresses it is useful to determine where connection attempts occurred. In many scenarios you might be looking for multiple connections to a single IP address or for repeated in-bound connections from a single source IP address.

Ports Involved? Knowing the TCP/UDP port is useful to determine which application might have generated the packet. In addition, you could use this information to see what type of connection context might have taken place. For example, a connection to a specific port will indicate a potential type of activity such as web surfing, file transfer, or file sharing.

Type of Packet? The packet type will tell you the TCP or UDP protocol used and the flags of the packet. The TCPIP flags will help you determine if it was the beginning of a connection such as a SYN flag.

Overall, the Windows Firewall Log is useful if it is turned on. In many environments, administrators have turned this on. As a result, it is worthwhile to check the file out to correlate this information with the Windows Event logs.




**To configure Windows Firewall logging for Windows XP or Windows Server 2003**

Good How-To Found Here: http://support.microsoft.com/default.aspx?kbid=875357

1. Open the Group Policy Management Console to Windows Firewall.
2. In the navigation pane, click either Domain Profile or Standard Profile.
3. In the details pane, double-click Windows Firewall: Allow logging.
4. Click Enabled.
5. No logging actually occurs until you set one of following two options:
   - To create a log entry when Windows Firewall drops an incoming network packet, select Log dropped packets.
   - To create a log entry when Windows Firewall allows an inbound connection, select Log successful connections.
6. The default path for the log is %windir%\pfirewall.log. If you want to change this, type the path to the new location, or click Save As to select a file location.
7. The default maximum file size for the log is 4096 KB. If you want to change this, type in the new size in KB, or use the up and down arrows to select a size. The file will not grow beyond this size; when the limit is reached, old log entries are deleted to make room for the newly created ones.
8. Click OK twice.

**To configure Windows Firewall logging for Windows 7, Windows Vista, Windows Server 2008, or Windows Server 2008 R2**

1. Open the Group Policy Management Console to Windows Firewall with Advanced Security.
2. In the details pane, in the **Overview** section, click **Windows Firewall Properties**.
3. For each network location type (Domain, Private, Public), perform the following steps.
    1. Click the tab that corresponds to the network location type.
    2. Under **Logging**, click **Customize**.
    3. The default path for the log is **%windir%\system32\logfiles\firewall\pfirewall.log**. If you want to change this, clear the **Not configured** check box and type the path to the new location, or click **Browse** to select a file location.
    4. The default maximum file size for the log is 4,096 kilobytes (KB). If you want to change this, clear the **Not configured** check box, and type in the new size in KB, or use the up and down arrows to select a size. The file will not grow beyond this size; when the limit is reached, old log entries are deleted to make room for the newly created ones.
    5. No logging occurs until you set one of following two options:
        - To create a log entry when Windows Firewall drops an incoming network packet, change **Log dropped packets** to **Yes**.
        - To create a log entry when Windows Firewall allows an inbound connection, change **Log successful connections** to **Yes**.
    6. Click **OK** twice.

Source: http://technet.microsoft.com/en-us/library/cc947815%28WS.10%29.aspx#bkmk_2

A, C, D

# pfirewall.log

pfirewall.log - Notepad

File Edit Format View Help

```
#Version: 1.5
#Software: Microsoft Windows Firewall
#Time Format: Local
#Fields: date time action protocol src-ip dst-ip src-port dst-port size tcpflags tcpsyn tcpack tcpwin icmptype icmpcode info path

2004-10-27 11:56:18 DROP TCP 192.168.1.100 192.168.1.101 2270 445 48 S 1584384258 0 65535 - - - RECEIVE
2004-10-27 11:56:18 DROP TCP 192.168.1.100 192.168.1.101 2271 139 48 S 2322815226 0 65535 - - - RECEIVE
2004-10-27 11:56:21 DROP TCP 192.168.1.100 192.168.1.101 2270 445 48 S 1584384258 0 65535 - - - RECEIVE
2004-10-27 11:56:21 DROP TCP 192.168.1.100 192.168.1.101 2271 139 48 S 2322815226 0 65535 - - - RECEIVE
2004-10-27 11:56:27 DROP TCP 192.168.1.100 192.168.1.101 2270 445 48 S 1584384258 0 65535 - - - RECEIVE
2004-10-27 11:56:27 DROP TCP 192.168.1.100 192.168.1.101 2271 139 48 S 2322815226 0 65535 - - - RECEIVE
2004-10-27 12:04:05 OPEN-INBOUND TCP 192.168.1.100 192.168.1.101 2276 445 - - - - - - - - -
2004-10-27 12:04:05 OPEN-INBOUND TCP 192.168.1.100 192.168.1.101 2277 139 - - - - - - - - -
2004-10-27 12:04:05 CLOSE TCP 192.168.1.101 192.168.1.100 139 2277 - - - - - - - - -
2004-10-27 12:04:17 CLOSE TCP 192.168.1.101 192.168.1.100 445 2276 - - - - - - - - -
```

Core Windows Forensics - SANS ©2011

The security log contains two sections. The *Header* provides information about the version of the log and the fields available. The *body* of the log is the compiled data that is entered as a result of traffic that tries to cross the firewall.

These fields are written from left to right across the page. The (-) is used when there is no entry available for the field. The data below corresponds to each field directly below each header field across the top, although the fields do not line up. Your log file may be slightly different. The important thing about the log is that this will give you clues about what is trying to tamper with your network or your PC.

In this log file, both successful connections as well as dropped packets are being logged. The first six lines consist of three repeated attempts that connect on port 445. The connection for these attempts was dropped. DROP indicates a packet blocked. The next few lines show that TCP on port 445 and 139 were open, followed by CLOSE in the last two lines. CLOSE indicates a normal closure of a TCP connection that was opened in the firewall. This is only logged when "Log successful connections" is checked.

# What is Recorded? (1)

| Fields | Description | Example |
|---|---|---|
| Date | Displays the year, month, and day that the recorded transaction occurred. Dates are recorded in the format YYYY-MM-DD, where YYYY is the year, MM is the month, and DD is the day. | 2001-01-27 |
| Time | Displays the hour, minute, and seconds when the recorded transaction occurred. Times are recorded in the format: HH:MM:SS, where HH is the hour in 24-hour format, MM is the number of minutes, and SS is the number of seconds. | 21:36:59 |
| Action | Indicates the operation that was observed by the firewall. The options available to the firewall are OPEN, CLOSE, DROP, and INFO-EVENTS-LOST. An INFO-EVENTS-LOST action indicates the number of events that occurred but were not recorded in the log. | OPEN |
| Protocol | Displays the protocol that was used for the communication. A protocol entry can also be a number for packets that are not using TCP, UDP, or ICMP. | TCP |
| src-ip | Displays the source IP address, or the IP address of the computer, that is trying to establish communications. | 192.168.0.1 |
| dst-ip | Displays the destination IP address of a communication try. | 192.168.0.1 |

Core Windows Forensics - SANS ©2011

Date
Displays the year, month, and day that the recorded transaction occurred. Dates are recorded in the format YYYY-MM-DD, where YYYY is the year, MM is the month, and DD is the day.

Time
Displays the hour, minute, and seconds when the recorded transaction occurred. Times are recorded in the format: HH:MM:SS, where HH is the hour in 24-hour format, MM is the number of minutes, and SS is the number of seconds.

Action
Indicates the operation that was observed by the firewall. The options available to the firewall are OPEN, CLOSE, DROP, and INFO-EVENTS-LOST. An INFO-EVENTS-LOST action indicates the number of events that occurred but were not recorded in the log.

Protocol
Displays the protocol that was used for the communication. A protocol entry can also be a number for packets that are not using TCP, UDP, or ICMP.

src-ip
Displays the source IP address, or the IP address of the computer, that is trying to establish communications.

dst-ip
Displays the destination IP address of a communication try.

# What is Recorded? (2)

| Fields | Description | Example |
|---|---|---|
| size | Displays the packet size in bytes. | 60 |
| tcpflags | Displays the TCP control flags that are found in the TCP header of an IP packet:<br>•Ack acknowledgement field significant<br>•Fin No more data from sender<br>•Psh Push function<br>•Rst Reset the connection<br>•Syn Synchronize sequence numbers<br>•Urg Urgent Pointer field significant<br>Flags are written as uppercase letters. | AFP |
| tcpsyn | Displays the TCP sequence number in the packet. | 1315819770 |
| tcpack | Displays the TCP acknowledgement number in the packet. | 0 |
| tcpwin | Displays the TCP window size in bytes in the packet. | 64240 |
| icmptype | Displays a number that represents the Type field of the ICMP message. | 8 |

Core Windows Forensics - SANS ©2011

Size

Displays the packet size in bytes.

Tcpflags

Displays the TCP control flags that are found in the TCP header of an IP packet:

- Ack acknowledgement field significant
- Fin No more data from sender
- Psh Push function
- Rst Reset the connection
- Syn Synchronize sequence numbers
- Urg Urgent Pointer field significant
- Flags are written as uppercase letters.

Tcpsyn

Displays the TCP sequence number in the packet.

Tcpack

Displays the TCP acknowledgement number in the packet.

Tcpwin

Displays the TCP window size in bytes in the packet.

Icmptype

Displays a number that represents the Type field of the ICMP message.

# Identifying Malicious Activity

[redacted]

Open Logfile in Mandiant ~~Web Historian~~

Identify DROP entries in the action field

Note of the destination IP addresses of the packets.

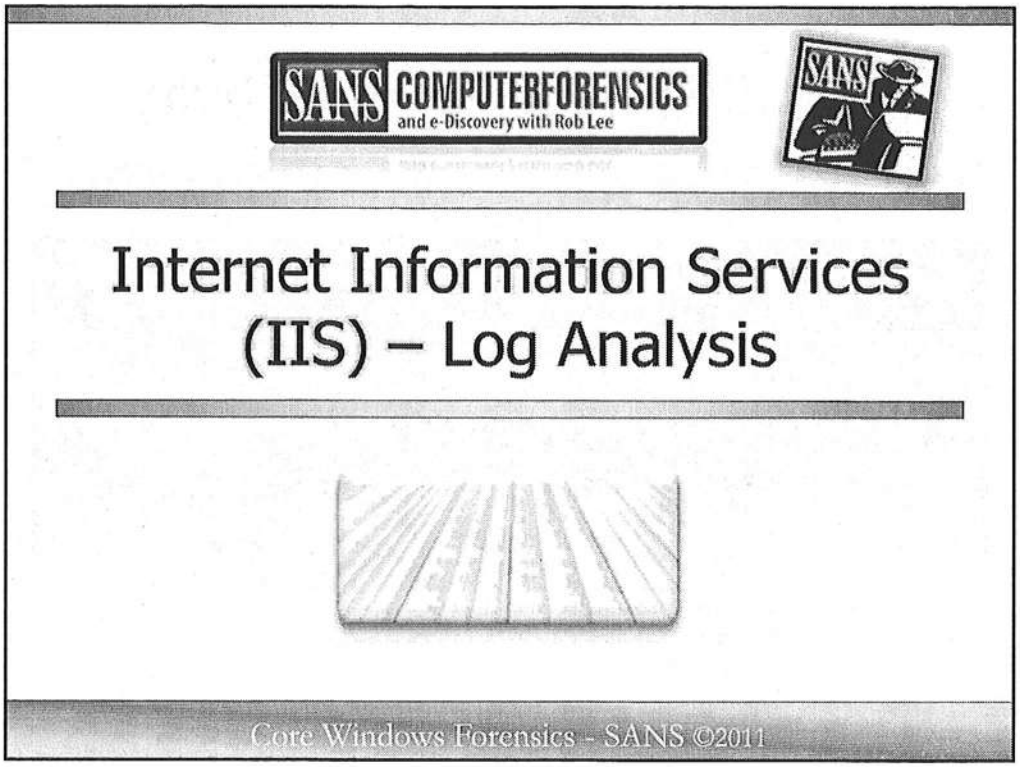
If those destination IP addresses are all the same, similar, or systematic, write down the source IP addresses (srcip) and the destination IP addresses.

- Systematic dropped packets can be considered suspicious.
- Suspicious dropped packets often have systematic port hit entries as well

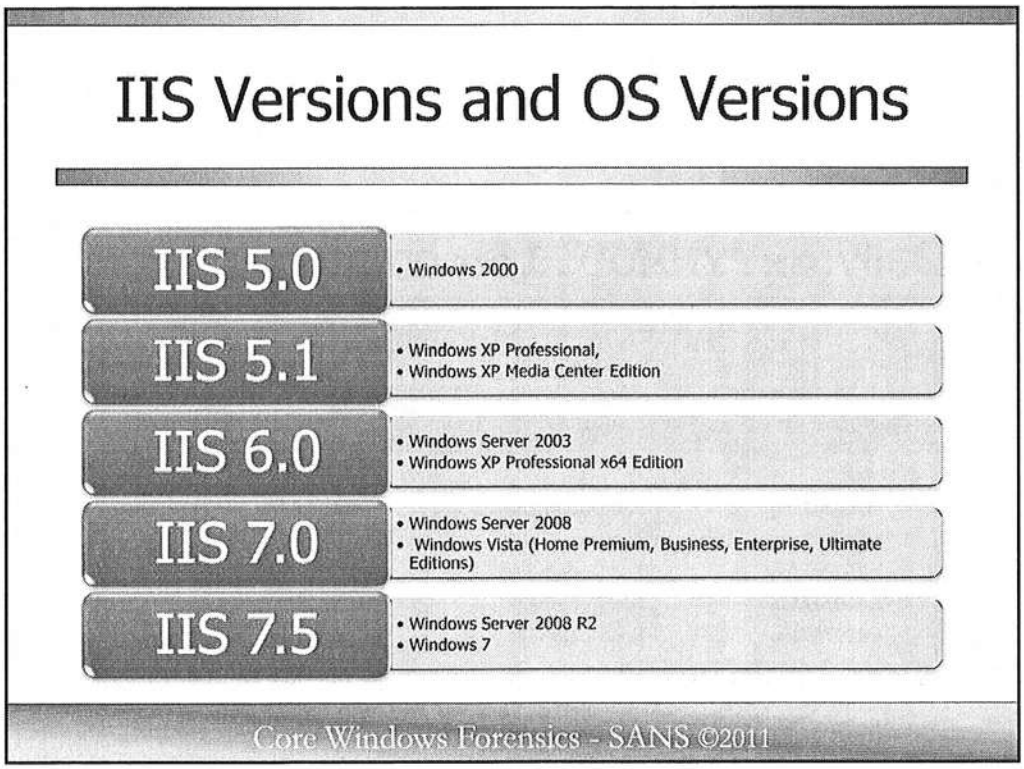
Core Windows Forensics - SANS ©2011

1. To identify malicious activity With the Windows Firewall log file open in Mandiant Web Historian, scroll through the file from beginning to end.
2. Look at each log entry with DROP in the action field and note whether the destination IP address (dstip) ends with a number other than 255.
3. If you find many such entries, take note of the destination IP addresses of the packets.
4. If those destination IP addresses are all the same, similar, or systematic, write down the source IP addresses (srcip) and the destination IP addresses. These dropped packets can be considered suspicious. Suspicious dropped packets often have systematic port hit entries as well.

http://technet.microsoft.com/en-us/library/cc782192%28WS.10%29.aspx




This page intentionally left blank.




Internet Information Services (IIS) – formerly called Internet Information Server – is a web server application and set of feature extension modules created by Microsoft for use with Microsoft Windows. The protocols supported in the latest version, IIS 7, include: FTP, FTPS, SMTP, NNTP, and HTTP/HTTPS.

- IIS 1.0, Windows NT 3.51 available as a free add-on
- IIS 2.0, Windows NT 4.0
- IIS 3.0, Windows NT 4.0 Service Pack 3
- IIS 4.0, Windows NT 4.0 Option Pack
- IIS 5.0, Windows 2000
- IIS 5.1, Windows XP Professional, Windows XP Media Center Edition
- IIS 6.0, Windows Server 2003 and Windows XP Professional x64 Edition
- IIS 7.0, Windows Server 2008 and Windows Vista (Home Premium, Business, Enterprise, Ultimate Editions)
- IIS 7.5, Windows Server 2008 R2 and Windows 7

IIS is not turned on by default when Windows is installed, but it can be selected from the list of optional features. It is available in all editions of Windows Vista and Windows 7, including Home Basic, but some features are not supported on client versions of Windows.

# IIS Log Introduction

- Daily `exYYMMDD.log` created

C:\Windows\System32\LogFiles\W3SVC1

nize ▾ Views ▾ Burn

| Name | Type | Size |
|---|---|---|
| ex100704.log | Text Document | 3 KB |
| ex100705.log | Text Document | 30 KB |
| ex100706.log | Text Document | 99 KB |
| ex100707.log | Text Document | 8 KB |
| ex100708.log | Text Document | 61 KB |



This is a snapshot of a Windows Server 2003 system (IIS 6.0) and the ANSI based logfiles.

# Where to Find IIS Logs

**IIS5 - IIS6**

- %windir%\System32\LogFiles
  - exYYMMDD.log

**IIS 7 – IIS 7.5**

- %SystemDrive%\inetpub\logs\LogFiles
  - u_exYYMMDD.log (utf-8)
  - exYYMMDD.log (ANSI)

Core Windows Forensics - SANS ©2011

IIS5 - IIS6

```
%windir%\System32\LogFiles
        exYYMMDD.log
```

IIS 7 – IIS 7.5

```
%SystemDrive%\inetpub\logs\LogFiles
        u_exYYMMDD.log (utf-8)
        exYYMMDD.log (ANSI)
```

Default locations can be changed in the Registry or the IIS management console.

By default, IIS 7.0 stores log files by using UTF-8 encoding. This changes the default file naming convention so that the files start with u_ (for example, u_exYYMMDD.log). UTF-8 encoding allows for single-byte and multi-byte characters in one string. This encoding enables you to read text-based logs (for example, logs that use W3C Extended, IIS, and NCSA Common formats) in a language other than English. IIS does not support the UTF-8 format for the built-in FTP Publishing Service log files. UTF-8 encoding is available in IIS 6.0, but it is not enabled by default. If you do not want to have your logs use UTF-8 encoding, you can use ANSI as the format.[1]

[1] http://iis70.blogspot.com/2009/02/ii70-troubleshooting-and-logging.html

# IIS Log Formats

- Log File Formats
  - IIS
  - W3C Extended Log File Format (Default)
  - NCSA

```
#Software: Microsoft Internet Information Services 6.0
#Version: 1.0
#Date: 2008-07-19 15:14:08
#Fields: date time s-sitename s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs(User-Agent) sc-status sc-substatus sc-win32-status
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
2008-07-19 15:14:08 W3SVC1 192.168.252.129 GET /bank/login.aspx uid=JohnSmith&passw=';declare%2F%2A%2A%2F%40a%2F%2A%2A%2Fvarchar%28800%29%2F%2A%2A%2
```

Core Windows Forensics - SANS ©2011

What is the W3C Extended Log File Format?

An extended log file contains a sequence of lines containing ASCII characters terminated by either the sequence LF or CRLF. Log file generators should follow the line termination convention for the platform on which they are executed. Analyzers should accept either form. Each line may contain either a directive or an entry.

Entries consist of a sequence of fields relating to a single HTTP transaction. Fields are separated by whitespace, the use of tab characters for this purpose is encouraged. If a field is unused in a particular entry dash "-" marks the omitted field. Directives record information about the logging process itself.

Lines beginning with the # character contain directives.

The following directives are defined:

- Version: <integer>.<integer> The version of the extended log file format used. This draft defines version 1.0.
- Fields: [<specifier>...] Specifies the fields recorded in the log. Software: string Identifies the software which generated the log.
- Start-Date: <date> <time> The date and time at which the log was started.
- End-Date:<date> <time> The date and time at which the log was finished.
- Date:<date> <time> The date and time at which the entry was added.
- Remark: <text> Comment information. Data recorded in this field should be ignored by analysis tools.

The directives Version and Fields are required and should precede all entries in the log. The Fields directive specifies the data recorded in the fields of each entry. [1]

[1] http://www.w3.org/TR/WD-logfile

# W3C Extended Log Format (IIS)

| Field | Appears As | Description | Default |
|---|---|---|---|
| Date | date | The date on which the activity occurred. | Y |
| Time | time | The time, in coordinated universal time (UTC), at which the activity occurred. | Y |
| Client IP Address | c-ip | The IP address of the client that made the request. | Y |
| User Name | cs-username | The name of the authenticated user who accessed your server. Anonymous users are indicated by a hyphen. | Y |
| Service Name and Instance Number | s-sitename | The Internet service name and instance number that was running on the client. | N |
| Server Name | s-computername | The name of the server on which the log file entry was generated. | N |
| Server IP Address | s-ip | The IP address of the server on which the log file entry was generated. | Y |
| Server Port | s-port | The server port number that is configured for the service. | Y |
| Method | cs-method | The requested action, for example, a GET method. | Y |
| URI Stem | cs-uri-stem | The target of the action, for example, Default.htm. | Y |
| Field | Appears As | Description | Default |
| Date | date | The date on which the activity occurred. | Y |

Core Windows Forensics - SANS ©2011

This log file format is used by used by **Microsoft Internet Information Server** (**IIS**) 4.0, 5.0, 6.0, 7.0 and 7.5. A log file in the extended format contains a sequence of lines containing ASCII characters. Each line may contain either a directive or an entry. Entries consist of a sequence of fields relating to a single HTTP transaction. Fields are separated by white space. If a field is unused in a particular entry dash "-" marks the omitted field. Directives record information about the logging process itself. [1]

[1] http://www.loganalyzer.net/log-analyzer/w3c-extended.html

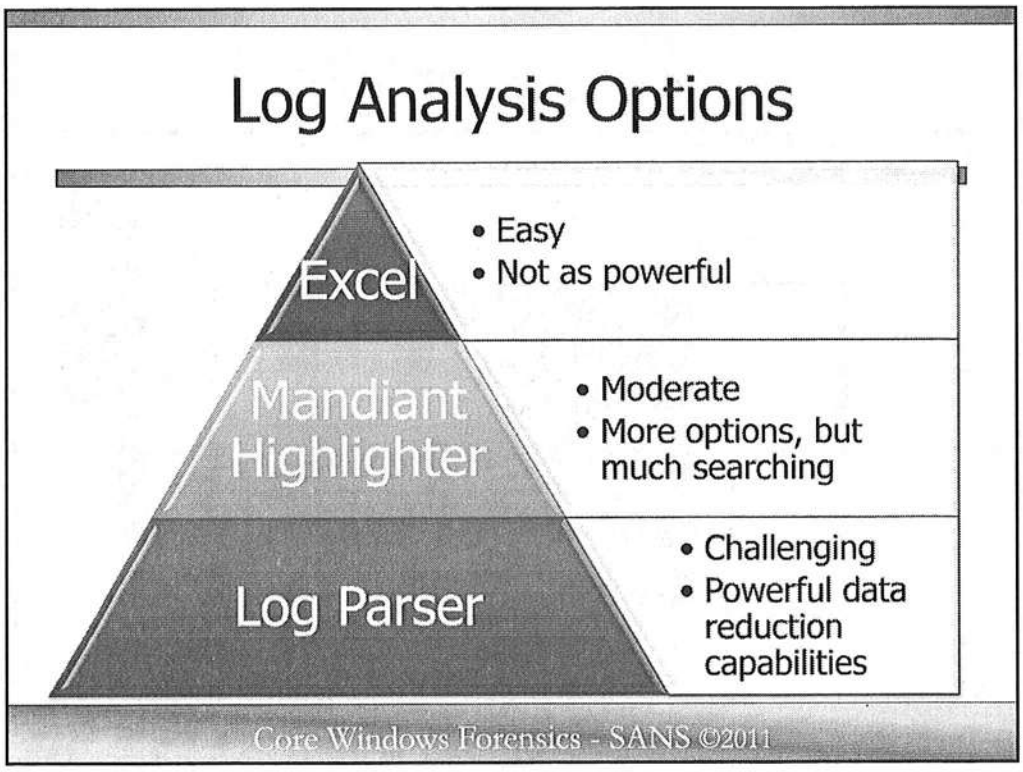
[2] http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/bea506fd-38bc-4850-a4fb-e3a0379d321f.mspx?mfr=true

# W3C Extended Log Format (2)

| Field | Appears As | Description | Default |
|---|---|---|---|
| URI Query | cs-uri-query | The query, if any, that the client was trying to perform. A Universal Resource Identifier (URI) query is necessary only for dynamic pages. | Y |
| HTTP Status | sc-status | The HTTP status code. | Y |
| Win32 Status | sc-win32-status | The Windows status code. | N |
| Bytes Sent | sc-bytes | The number of bytes that the server sent. | N |
| Bytes Received | cs-bytes | The number of bytes that the server received. | N |
| Time Taken | time-taken | The length of time that the action took, in milliseconds. | N |
| Protocol Version | cs-version | The protocol version —HTTP or FTP —that the client used. | N |
| Host | cs-host | The host header name, if any. | N |
| User Agent | cs(User-Agent) | The browser type that the client used. | Y |
| Cookie | cs(Cookie) | The content of the cookie sent or received, if any. | N |
| Referrer | cs(Referrer) | The site that the user last visited. This site provided a link to the current site. | N |
| Protocol Substatus | sc-substatus | The substatus error code. | Y |

Core Windows Forensics - SANS ©2011

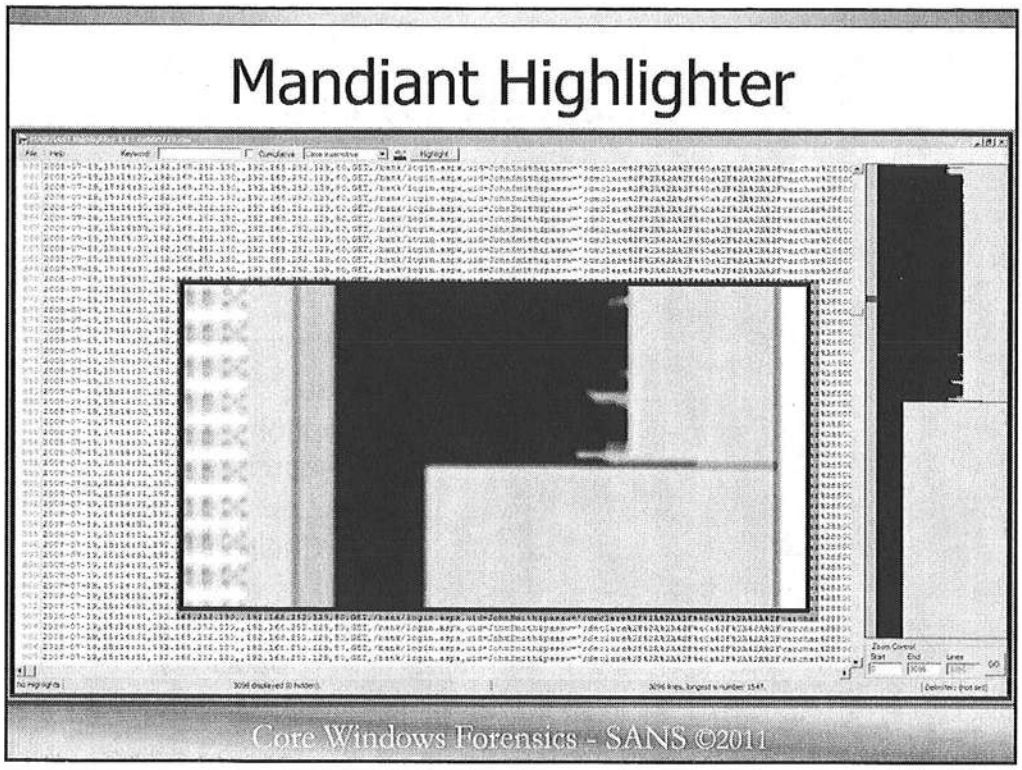
Continuation from the previous slide.



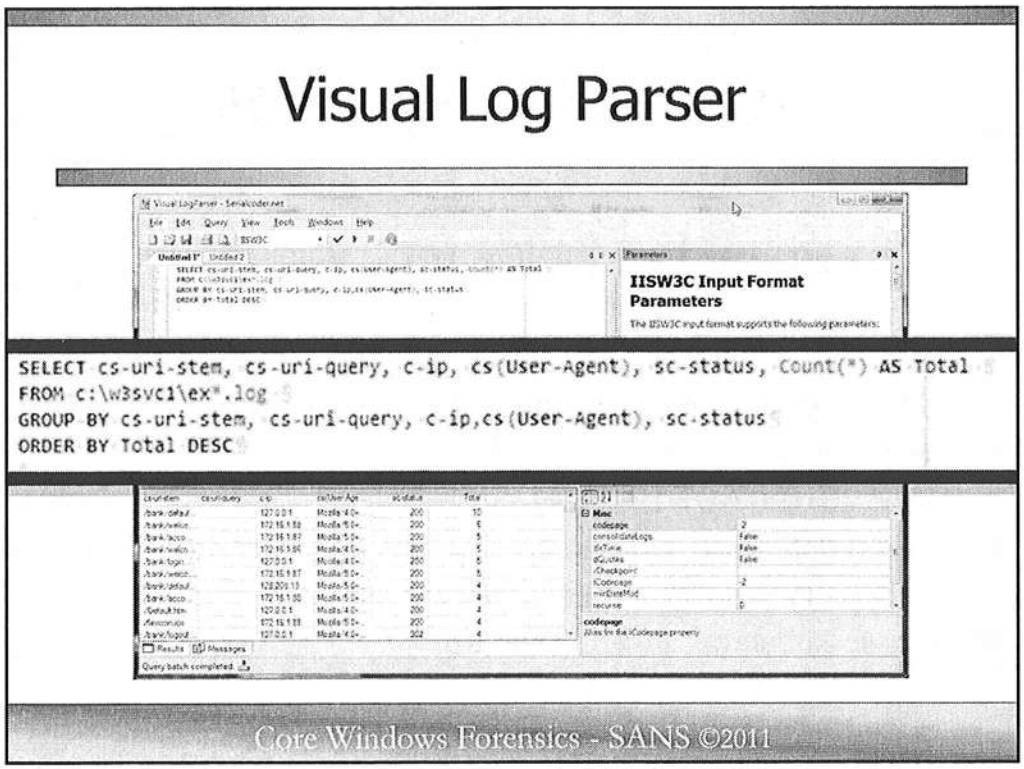

There are many ways to analyze W3C IIS Log Files. The most generic way is to use Microsoft Office Excel or a Text Editor such as Notepad. However, while in Excel it is powerful, it does not help you as much as using one of the other options that are available to the trained examiner.

Mandiant Highlighter, covered earlier, allows an investigator to parse/view a log file from a 30,000 foot view to potentially see lines that are out of place much easier.

The most difficult analysis, but also the most powerful for searching through many log files created by IIS, is the Log Parser tool. While challenging to learn the query language, it does allow for the most options available to the investigator.




Mandiant Highlighter is very useful to search for specific strings inside the IIS logfiles. Here is an example where you can easily identify where there might be some trouble. On the right side, you can see the entire log file from above. You can identify where a very long line occurs. This is usually indicative of potential bad activity.




Log Parser and Visual Log Parser offer an advanced investigator the ability to begin examining the intricacies of the log files. An investigator could examine the total number of errors that occurred on a specific day, week, or month. The examiner could look for all files of a specific type. The log parser capabilities are very powerful, however, they do tend to be more advanced than a standard examination of the log file via excel or Highlighter.

In Appendix E, there is an article describing the more advanced features of the logparser tool and how it can be used in advanced scenarios. While this leads more toward 508 material, we do feel that some exposure to this in the core/essentials course is useful to providing an investigator with thoughts on how it can be used in more advanced intrusion detection scenarios.

# Review Types in Logs

- Examine the Fields for Types

#Software: Microsoft Internet Information Services 6.0
#Version: 1.0
#Date: 2010-07-07 19:43:41
#Fields: date time s-sitename s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs(User-Agent) sc-status sc-substatus sc-win32-status



What does log parser look for? Log Parser will examine the fields area and make determinations based off of that data. Log parser can be used to examine and of the fields located in the W3C header. Which fields are needed?

# Needed Types?

| Log File Heading | Description | Needed? |
|---|---|---|
| Date | The date on which the request was served (Universal Time Coordinates (UTC) time) | Y |
| Time | The time at which the request was served (Universal Time Coordinates (UTC) time) | N |
| s-sitename | The IIS service name and site instance number that served the request | N |
| s-ip | The IP address of the server that served the request | N |
| Cs-method | The HTTP request verb or FTP operation | Y |
| cs-uri-stem | The HTTP request uri-stem or FTP operation target | Y |
| cs-uri-query | The HTTP request uri-query, or NULL if the requested URI did not include a uri-query | Y |
| s-port | The server port number that received the request | N |
| cs-username | The name of the authenticated user that made the request, or NULL if the request was from an anonymous user | Maybe |
| c-ip | The IP address of the client that made the request | Y |
| cs(User-Agent) | The client request User-Agent header | Y |
| sc-status | The response HTTP or FTP status code | Y |
| sc-substatus | The response HTTP sub-status code *(this field is logged by IIS version 6.0 and later only)* | N |
| sc-win32-status | The Windows status code associated with the response HTTP or FTP status code | N |



The following fields with a "Y" are labeled as types that you might want to perform searches off of. If you examine Appendix E, you will see many of the queries that are built off of these locations.

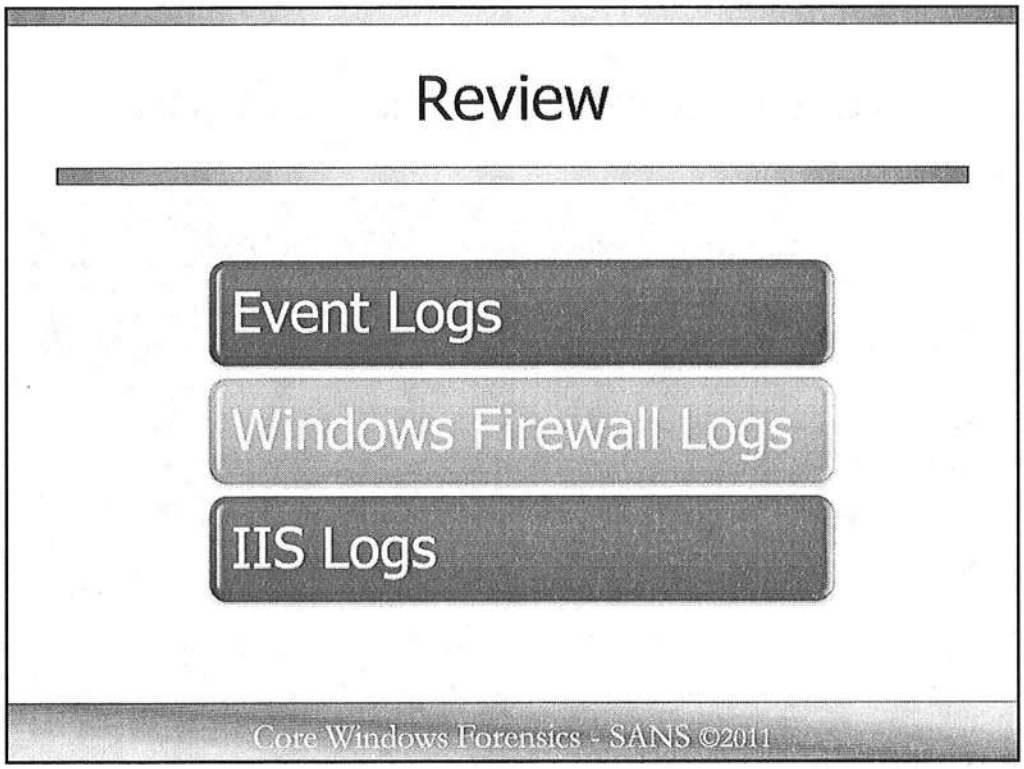
# Indicators of Compromise Via IIS Logfile Analysis

**Search for the following strings that indicate potential SQL injection or questionable activity**

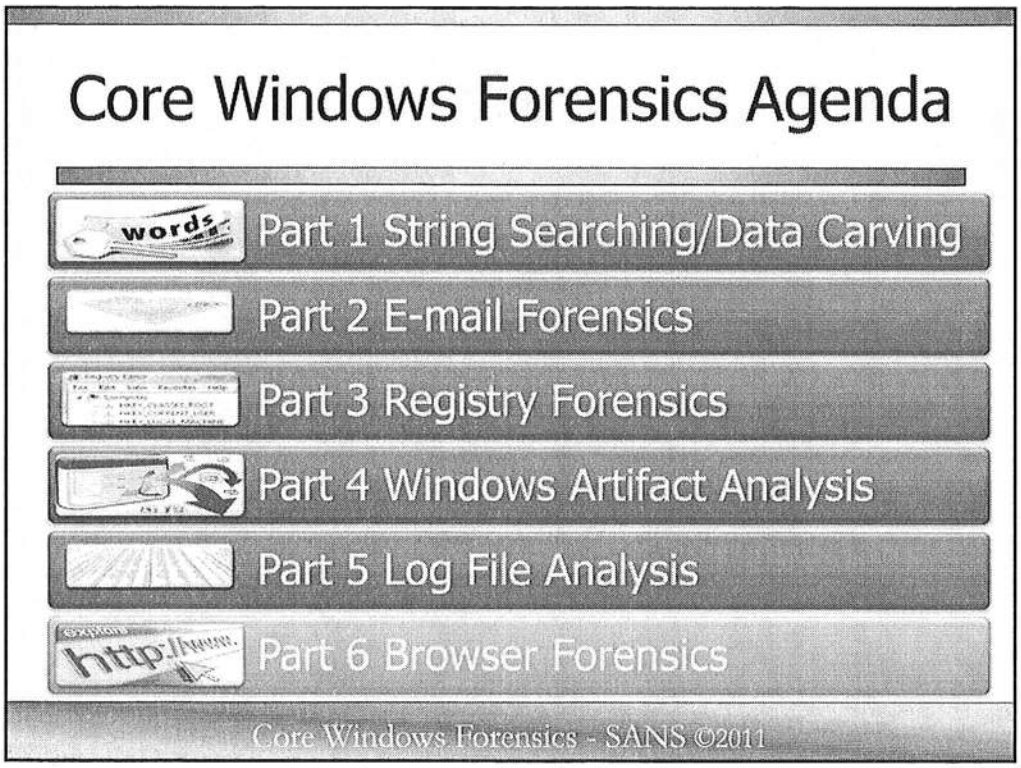
- xp_cmdshell
- cast
- union
- update
- opendatasource
- convert
- Select
- openrowset
- queryout



The xp_cmdshell, OPENROWSET, and OPENDATASOURCE stored procedures should be disabled on all databases unless they are explicitly serving a business need within the network. The xp_cmdshell procedure allows someone to execute commands on a local system from the database, with the permissions of the service account used for the database. The OPENROWSET and OPENDATASOURCE procedures allow one to cause the database to transfer data from the local database to a remote database and vice versa.



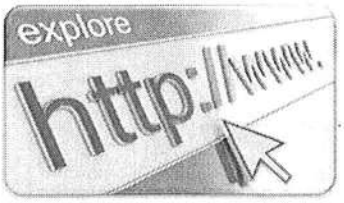

This page intentionally left blank.




This page intentionally left blank.

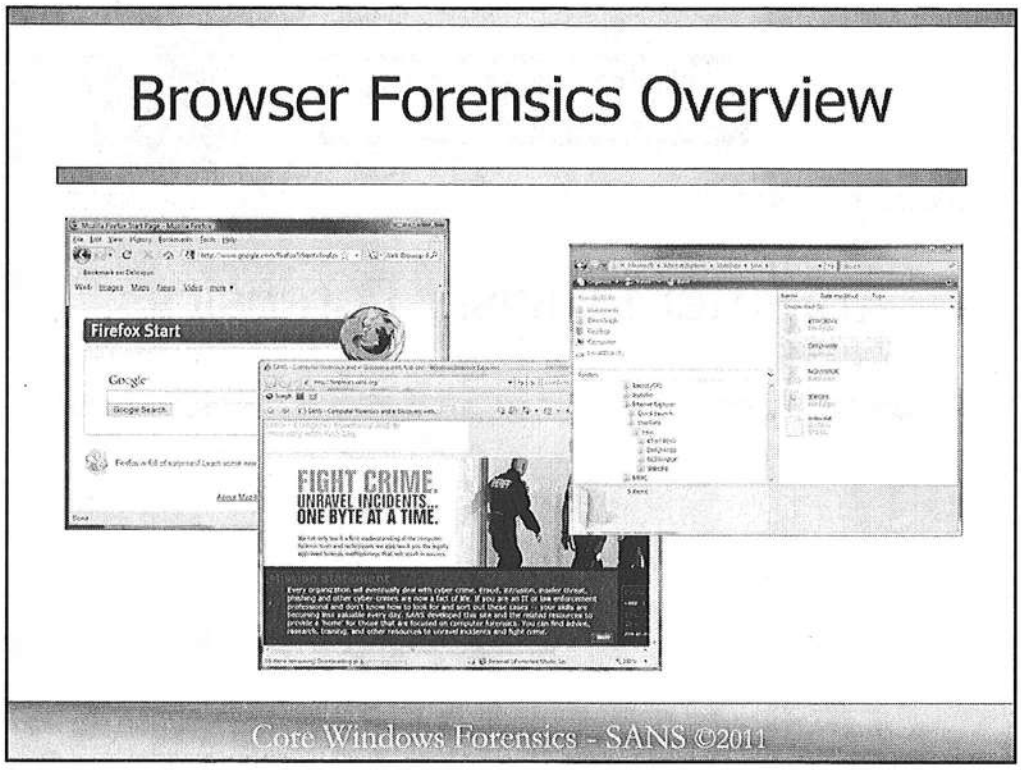
SANS COMPUTERFORENSICS
and e-Discovery with Rob Lee

# Internet Browser Forensics



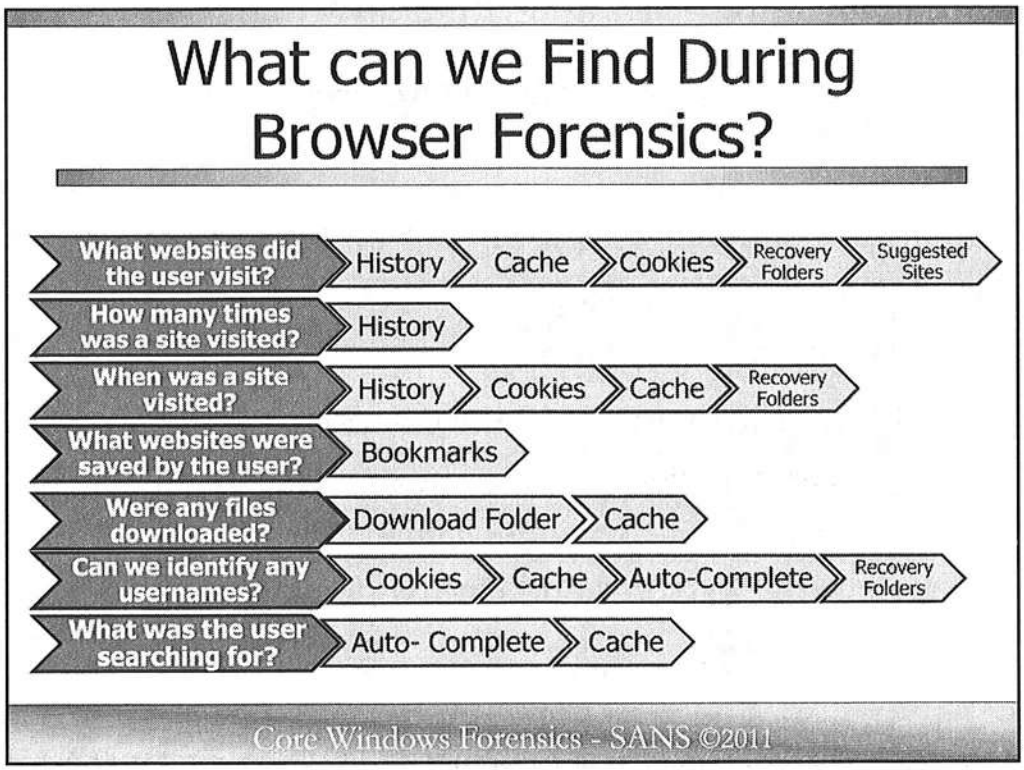



This page intentionally left blank.




Browser forensics is a critical skill because it can provide an overwhelming amount of information to the investigator. Accessing the Internet is one of the most frequent user activities and browsers are the key portal used to facilitate that access. In some cases, such as employee misuse, Internet activity alone may provide the key pieces of evidence. In other cases Internet activity may not be the focus, but can provide valuable corroborating evidence. A frequent example is reviewing the history for local file access or browsing of network shares during a computer intrusion investigation.

In this section we cover the range of information we can expect to find when conducting browser forensics. We will focus on the two dominant browsers on the market: Internet Explorer and Mozilla Firefox. First we will discuss the concepts using Internet Explorer as a model and then extend those ideas to examine browser artifacts left by Mozilla Firefox.




During any forensic analysis, the investigator should always strive to identify what a piece of trace evidence is and how it relates to the fundamental questions that he or she is trying to answer. Like other applications we will investigate, Internet browsers store a large amount of user data, that we will call artifacts. While there are many different browser artifacts, three in particular provide the foundation of most browser evidence:

- History Files
- Browser Cache
- Cookies

Evidence from these locations go a long way toward helping the investigator profile a user: determining what sites were visited, how often, at what times, and what kinds of activities were performed on that site. While these three artifacts are invaluable, it is important to be aware that there are ancillary locations which can greatly help us corroborate evidence and fill in the picture of what happened on a computer system. Items like bookmarks can show knowledge and intent of the user who chose to save them. The download folder and temporary directories often house long forgotten downloads that could be relevant. And auto-complete information, while not always easy to get to, can provide excellent information on form data entered, search terms, and usernames utilized. History and cache files are also the items most often cleared by users, so in some cases the "ancillary" artifact locations will be the only ones available to aid your investigation.

# Browser Basics

- Used to view a variety of web content

| | | |
|---|---|---|
| • Text | • Video | • File servers |
| • Images | • Code | • Data feeds |

- Many different browsers exist

| | |
|---|---|
| • Internet Explorer | • Firefox |
| • Safari | • Chrome |

- Internet Explorer and Firefox dominate

Core Windows Forensics - SANS ©2011

An Internet browser is a piece of software which allows users to interact with web pages or web content. Originally this was text pages coded using Hyper-Text Markup Language (HTML), but the ubiquity of the web has added a vast array of items that web browsers are now used to view. Video and images are commonplace, and programming code like javascript and Adobe Flash are adding increasing functionality to web sites. Browsers are now used as web-based e-mail clients, RSS feed readers, to transfer files, and to view offline content. Given all of these activities, it should come as no surprise that browser artifacts give an investigator an excellent window into how a computer system is being used.

While the first two "browser wars" are over, the world is growing increasingly dependent on the web and many different organizations are competing for a piece of the browser market. [1] Currently Internet Explorer and Mozilla Firefox control over 90% of the market, but the rapid adoption of Apple iPhones and Mac systems has allowed Safari to gain some market share. [2] The Chrome browser was released by Google in 2008, but has yet to make much of an impact in the marketplace.

**References**

[1] http://en.wikipedia.org/wiki/Browser_wars

[2] http://gs.statcounter.com

# Internet Explorer (IE)

- **IE6**
  - Still encountered on Windows 2000 and XP
- **IE7**
  - Most popular browser by a wide margin
  - Standard on Vista and up-to-date XP machines
- **IE8**
  - Released in 2009; default for Windows 7

All Versions Share the Same Artifacts and Locations



Internet Explorer is the 800 pound gorilla of web browsers due to its tight coupling with the Windows operating system. It once controlled over 95% of the browser market, but is now closer to 70%. It has gone through many iterations, but where and how it stores its browser artifacts has largely remained unchanged since Internet Explorer 5. This is a great boon to investigators since the forensic methodology to analyze these browsers remains virtually unchanged. The one caveat to this is that Microsoft radically changed the folder structure and thus the artifact locations in their Vista/Win7 release. These new locations are addressed in a slide later in the course. Investigators are most likely to see the following IE versions installed on modern computer systems:

IE6 – Released in 2001 and found on Windows 2000 and Windows XP systems. In 2007, IE6 still had a 35% market share, but this has dropped dramatically when the IE7 upgrade was released by Microsoft as part of an auto-update for XP, Vista, and Server 2003 in early 2008 (IE7 will not run on Windows 2000). [1]

IE7 – Released in 2006 and now the most commonly found browser on Windows machines. If an XP machine is up-to-date with its patches, it will likely be running IE7. IE7 was released as the default browser in Vista.

IE8 – Released in early 2009 and set to be the default browser for Windows 7. Adoption has been slow, but improved security features may encourage users to upgrade quickly.

References

[1] http://www.infoworld.com/archives/emailPrint.jsp?R=printThis&A=/article/08/01/17/Microsoft-warns-businesses-of-autoupdate-to-IE7_1.html

# Internet Explorer Data Locations*

**History Files**

- `%userprofile%\Local Settings\History\History.IE5`

**Cache**

- `%userprofile%\Local Settings\Temporary Internet Files\Content.IE5`

**Cookies**

- `%userprofile%\Cookies`

**Bookmarks**

- `%userprofile%\Favorites`

***Not including Vista/Win7**



With the exception of Microsoft Vista/Win7 (which will be covered in a later slide), the primary locations for browser artifacts have remained largely unchanged for the last ten years. This is evidenced even in the folder names, with their references to "IE5".

The **`Local Settings`**, **`History`**, and **`Temporary Internet Files`** directories are hidden in the Windows Explorer graphical user interface (GUI). To see them, a user must do the following:

- Open **`Folder Options`** from the control panel
- Select **`Show hidden files and folders`**
- Uncheck **`Hide protected operating system files`**

Further, the **`History.IE5`** and **`Content.IE5`** folders are replaced within the Windows Explorer GUI with virtual directories showing a representation of the actual files. On a live system, the best way to view these directories as they exist on disk, is to use the command line shell, making sure to use the **`dir /a`** (show all) option.

Cookie and Favorite files can be easily viewed within a user's profile.

In most cases, investigators will be reviewing these folders using disk images and forensic tools, which show all available directories and files by default.

# Index.dat Files

- In IE, most browser residue is stored in index.dat files
  - Used to store history, cookies, and cache information
  - Great for investigators since deleted records may be present
- Binary database format same since Internet Explorer 4
  - Index.dat structure the same, but data varies according to type
  - Possible to find 10 or more index.dat files on a target system
- Located in hidden, system directories
  - Viewed from the command line using the `dir /a` command
- Very difficult for a user to delete since they are always in use and locked by the operating system



To understand browser residue within the Internet Explorer application, it is critical to understand the **`index.dat`** database format. Similar to the folder locations, this format has been in place for over ten years and is used exclusively by the three most important browser artifacts: history, cache, and cookies.

Index.dat files are stored in binary form, meaning that special tools are required to decode their contents. This also means that basic text searches run on a forensic image may not be able to identify relevant hits within these files. While the name index.dat is used by history, cache, and cookie databases, the files themselves store very different information, reflecting the different artifacts they are responsible for. Most forensic suites have viewers which will decode the files, and many open-source tools exist as well. It is common to have to review a large number of index.dat files on a system, particularly if there are multiple user accounts and there has been a lot of Internet usage. Most will be related to user activity, but some will be located in the **`default`** and **`userdata`** directories, reflecting system usage. It is also possible to find old **`index.dat`** entries in unallocated space.

The directories used to store **`index.dat`** files may be hidden, system-level directories, such as **`History.IE5`** and **`Content.IE5`** which cannot be viewed from the Windows graphical user interface. Forensic suites or the command line can be used to export / review these files.

A great advantage to the investigator is that it is quite difficult for a user to remove **`index.dat`** files. The web abounds with questions from users asking how to delete these files. Since they are used by the Windows operating system as well as by the browser, they are "locked" on boot and thus cannot be removed using traditional means. However, tools within the IE browser itself can clear entries within the files.

# Internet Explorer History: Investigating Sites Visited

- Excellent for profiling Internet usage!
- Records websites visited by date & time
  - Details stored for each local user account
  - Records number of times visited (frequency)
  - Also tracks access of local system files
- Used by browser for auto-complete
- Trivial for user to clear / manipulate
  - Clear using the browser or delete entries in `History` folder
- Registry determines time span of records kept:

  `HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\URL History`

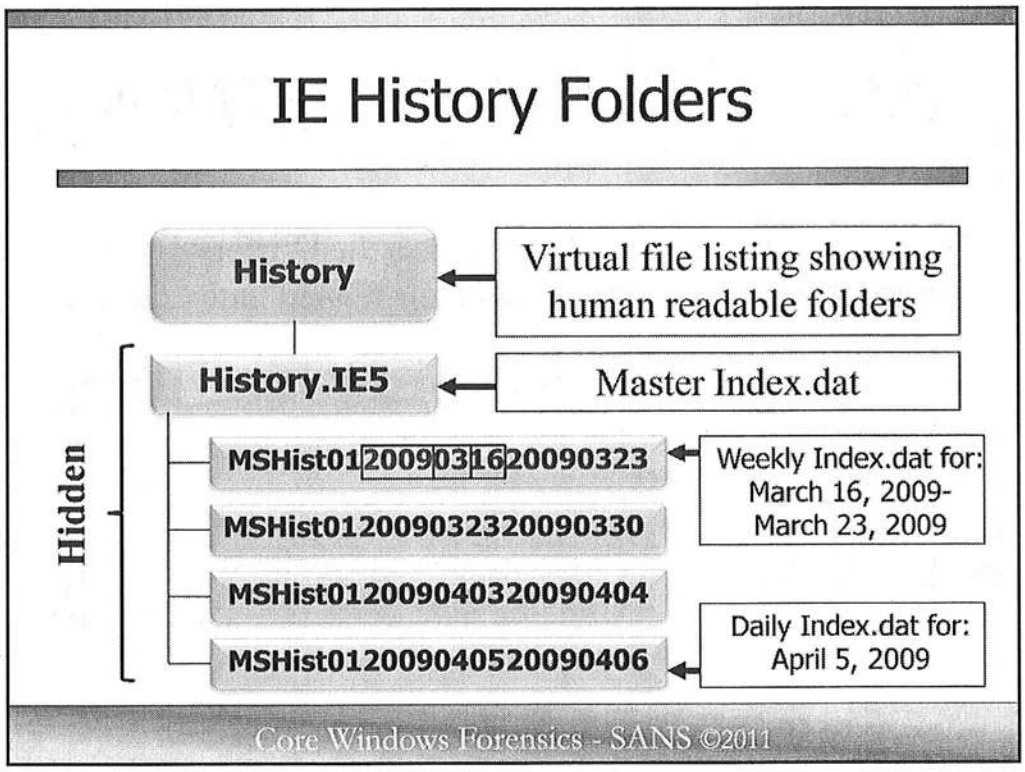


The Internet Explorer history artifact is a spectacular benefit to the forensic investigator. Essentially, Internet Explorer neatly tracks all of the websites visited during the entire time span it has been requested to do so. The default number of days to record the history is 20 days, and this value can be changed within the browser (Tools -> Internet Options) or via the Windows Registry. Should a user choose the "0 days" option (essentially turning the history feature off), a history is still kept while the user is logged in, but is deleted upon system shutdown or upon carry over to the next day. If no activity is accomplished on a given day, no daily history file is created.

History information can tell an investigator:

- What sites have been visited in the last X days (X being the number of days the history is set to record)
- What files were accessed on the system in the last X days
- How many times each site was visited (frequency information)
- The user account that was used to visit the site (history information is stored within a user's profile)
- The specific time the site was last accessed

Like most trace evidence in Windows, this feature was not created just to benefit the investigator. Its primary purpose is to query the website history to suggest websites on the URL bar of the browser while the user is typing (to save the user from having to type the entire address of frequently visited sites). Users can clear their history through the browser **or** by deleting individual history files in the hidden "History" directory.




The **History** folder viewed using the Windows Explorer graphical user interface is actually a virtual directory. Windows parses the information stored in a series of binary files (called `index.dat` files) to create an easy to review, virtual set of directories encompassing the days, weeks, and sometimes months of history records. The directories "Today", "Friday", "Last Week", etc., do not exist on the disk and are populated from actual directories with names like **MSHist012009040520090406**.

The directories holding history information are named according to the following convention:

**MSHIST01<StartYear><StartMonth><StartDay><EndYear><EndMonth><EndDay**

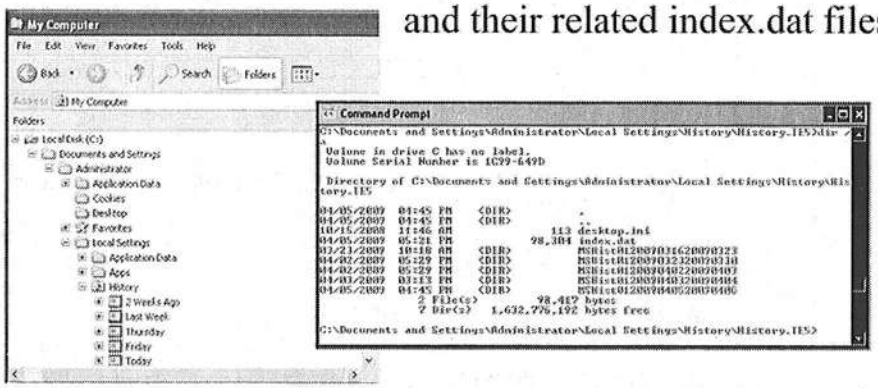
Thus **MSHist012009040520090406** equates to the history folder for records from April 5th, 2009 to April 6th, 2009. This slide shows two "daily" history folders and two "weekly" history folders. Daily history content is rolled up to the weekly folders after the sixth day. When this rollup occurs, the **index.dat** files in those directories are updated accordingly.

Each history folder has its own **index.dat** file which is a binary database containing all of the history records for that period. A "master" **index.dat** file is present in the parent **History.IE5** directory and contains most of the history records present in each directory. However, for a comprehensive review, it is important to review all of the **index.dat** files present in the various directories because sometimes daily or weekly entries do not get propagated to the master **index.dat** file.

Viewing the **History.IE5** directory and its child directories cannot be accomplished via Windows Explorer (on a live machine). Instead the investigator must use the **dir /a** command from the command line to force all of the files and directories to be displayed. Since a majority of our analysis will be conducted on forensic image files, this is rarely much of a problem.

# Virtual vs. Actual History Folders

- Windows Explorer will not show actual history folders and their related index.dat files

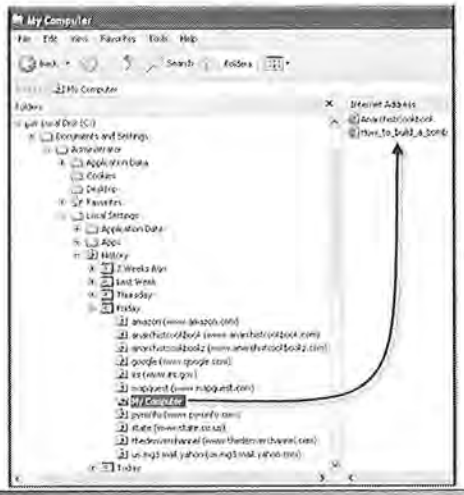





This slide visually shows how the history folders are displayed using the Windows Explorer GUI contrasted with their view from the command line. In the case of the former (image on the left), the history directories and index.dat files within **`History.IE5`** are parsed by the operating system and displayed in a series of user friendly virtual directories, starting with Today, and continuing on to previous days and weeks of activity. No **`index.dat`** files are shown within the GUI interface. Interestingly, a knowledgeable user can go into this virtual directory structure and selectively delete history entries. These deletions are subsequently cleared from the corresponding **`index.dat`** file.

The Command Prompt image shows the history information as it exists on disk. These are the folders you should expect to see when doing an analysis of a forensic image. Each directory contains an **`index.dat`** file which corresponds to that time period. The **`desktop.ini`** file maintains the folder options, namely hiding these directories from the user.

# Local File Access in IE History



- IE History also records local file access
- Under "My Computer" in Windows Explorer
- Stored in index.dat as: **`file:///C:/Anarchist Cookbook.doc`**
- Does **not** mean file was opened in browser



A little known fact about the IE History is that the information stored in the history files is not just related to Internet browsing. The history also records local and remote (via network shares) file access, giving us an excellent means for determining which files and applications were accessed on the system, day by day! The remote file access can be particularly interesting since it can identify other volumes used for file storage such as USB thumb-drives and other removable media. [redacted]

In this example, the files AnarchistCookbook and How_to_build_a_bomb were accessed on April 3rd, 2009. Notice that within the Windows Explorer GUI, local file access is recorded under the virtual folder, **`My Computer`**. Right-clicking on the entry stored there will give the file path. Local file access is recorded in the **`index.dat`** file in an identical fashion as Internet URLs are. The following is from the **`index.dat`** file:

```
Administrator@file:///C:/Documents%20and%20Settings/Administrator/My%20Do
cuments/Anarchy/AnarchistCookbook.doc

Administrator@file:///C:/Documents%20and%20Settings/Administrator/My%20Do
cuments/Anarchy/How_to_build_a_bomb.doc
```

It is important to note that seeing these entries in the **index.dat** file does not necessarily mean that the file was opened within the IE browser. It simply indicates that the file was opened without giving any information as to what file viewer was used. [redacted]

# IE History File Timestamps

- The Index.dat file stores two timestamps for each entry in the history database
  - Unfortunately, timestamp information varies depending on which index.dat file you are reviewing
  - Windows Explorer shows last access in local time

| Index.dat Location | Timestamp 1 | Timestamp 2 |
|---|---|---|
| History.IE5 (Master) | Last Access (UTC) | Last Access (UTC) |
| Daily History Folder | Last Access (Local) | Last Access (UTC) |
| Weekly History Folder | Last Access (Local) | Index.dat creation time (UTC) |

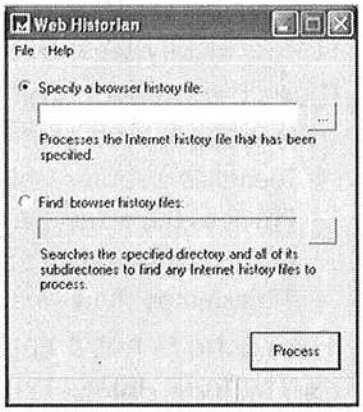
Core Windows Forensics - SANS ©2011

Timestamps used by the IE history **index.dat** files can be confusing, and could potentially lead to false assumptions being made. The **index.dat** file structure maintains two timestamps for each entry. If you recall, a given collection of history files will consist of multiple **index.dat** files. We have the master **index.dat** located in the **History.IE5** directory, and an **index.dat** file for each of the daily and weekly directories. For whatever reason, Microsoft has allowed history timestamps to vary according to which **index.dat** file they are stored in. The table above shows how each type of **index.dat** file displays time slightly differently. This is particularly important when trying to match up entries in a daily or weekly history (where time is kept in local machine time) with the master **index.dat**, which stores time exclusively in UTC time. The moral of the story here is to always know where your **index.dat** file came from before doing your analysis.

Additionally, keep in mind that if you are viewing history entries using the Windows Explorer GUI (on a live system), the time shown there will be the last access time in local machine time.

# Mandiant Web Historian

- Finds and parses History Index.dat files
  - Works on IE cache files too
- Provides XLS, CSV or HTML output
- Identifies deleted entries
- Also parses histories for Firefox, Opera and Safari
- Will not run on Vista




Located on your DVD and SIFT Workstation



Mandiant Web Historian is a freeware tool that specializes in parsing browser artifacts. [1] It collects data from IE, Firefox, Mozilla, Netscape, Safari, Opera, and exports it in a very user friendly output file in either Microsoft Excel or HTML format.

One of the nice features of Web Historian is that you can either specifically point it to an **`index.dat`** file, or it will scan the entire directory structure looking for files that it can parse and provide output for. This can be a godsend for **`index.dat`** files since there can sometimes be twenty or more scattered throughout the user directories. The output of each browser file found will be placed in its own tab, with the path of the file displayed at the top.

The latest release, version 1.3, does have a couple of limitations. It currently will not run from a Vista operating system and it does not show the number of times a website has been visited (hits) from the IE History **`index.dat`**. Additionally, the time column labels used in its reporting are confusing – make sure to check them with the IE History File Timestamps slide earlier in this section.

Its limitations notwithstanding, Web Historian has saved many an investigator a lot of time when working with browser artifacts. It is an excellent complement to whatever forensic suite you may be using.

**References**

[1] http://www.mandiant.com/software/webhistorian.htm

# Internet Explorer Cache: Viewing the Browser Stockpile

- The cache is place where web page components can be stored locally to speed up subsequent visits
- Gives the investigator a "snapshot in time" of what a user was looking at online
  - Identifies websites which were visited
  - Provides the actual files the user viewed on a given website
  - Cached files are tied to a specific local user account
  - Timestamps show when the site was first saved and last viewed
- The cache is not a complete record of every page visited
- Can include deleted (cleared) entries
  - IE7 and IE8 do a much better job of wiping cleared entries



The Internet cache works to speed up the web surfing experience. It takes advantage of the fact that items stored locally (i.e. on the hard drive) load much faster than items which need to be retrieved from a remote location such as a website. Thus behind the scenes, the web cache is constantly stockpiling content viewed in the web browser and archiving it for future use. Whenever a web request is made, the cache is first checked to see if a current version of that page or image exists and if it does, it returns the locally cached version. It is particularly effective when the same sites and resources are requested frequently. As an example, when you hit the "back" button on your browser, you are almost always accessing the cached version of that previous page (instead of having to re-download the entire thing). Other applications may also utilize the cache for storing files. One prominent example is Outlook Express, which stores opened attachments in the cache directory.

The cache is an excellent tool for profiling user activity, particularly when used in conjunction with browser history information. The cache is tied to a user account, storing its **`index.dat`** files within the user's profile folders. Since the cache stores the actual pages, the investigator can reconstruct what the user saw on the site and gather information on what activities were taking place. For example, in the case of a web-based file backup solution, the history files would tell us that the user accessed the site, and the cache files may provide us with the actual HTML pages listing the filenames of items stored.

By default, IE6 allocates 10% of the system drive to store cached files. Thus the larger the system volume, the more cached files we should expect to find. IE7 defaults to 50MB of cache space that can be increased up to 250MB. The cache is very dynamic and files are often swapped out via expiration or to make room for newer content. Thus it cannot be treated as a complete record of every page visited or resource requested. Additionally, part of the HTML standard allows sites to specify that their content should not be cached (**`no_cache_write`** option). Web applications like Google Mail do so to provide an extra measure of security for users. [1]

Similar to the IE history, the cache utilizes an **`index.dat`** file. These files have been known to include entries even after they have been manually cleared by a user. This is particularly true in IE6, where a clear cache request by the user deletes the actual cached files, but only marks the entries in the **`index.dat`** as "unused". Starting with IE7, a better effort is made at erasing both the files and the entries within the **`index.dat`** file.

**References**

[1] http://msdn.microsoft.com/en-us/library/aa383928(VS.85).aspx

# IE Cache: Index.dat

The index.dat is critical for reconstructing the cache

- Stores the original URL address
- Identifies the path of cached file and maps it to the original URL
- Populates a Type field that tells why the record exists (this field is also used in other index.dat files):

| | |
|---|---|
| **URL** | Indicates the URL was visited as part of a normal request |
| **LEAK** | Undocumented – appears to be caused by errors |
| **REDR** | Site visited as a result of a browser redirect (not saved in cache) |
| **HASH** | A hash index of the contents of the index.dat |

- Keeps the return HTTP header from the original website request
- Maintains timestamps for each entry

Core Windows Forensics - SANS ©2011

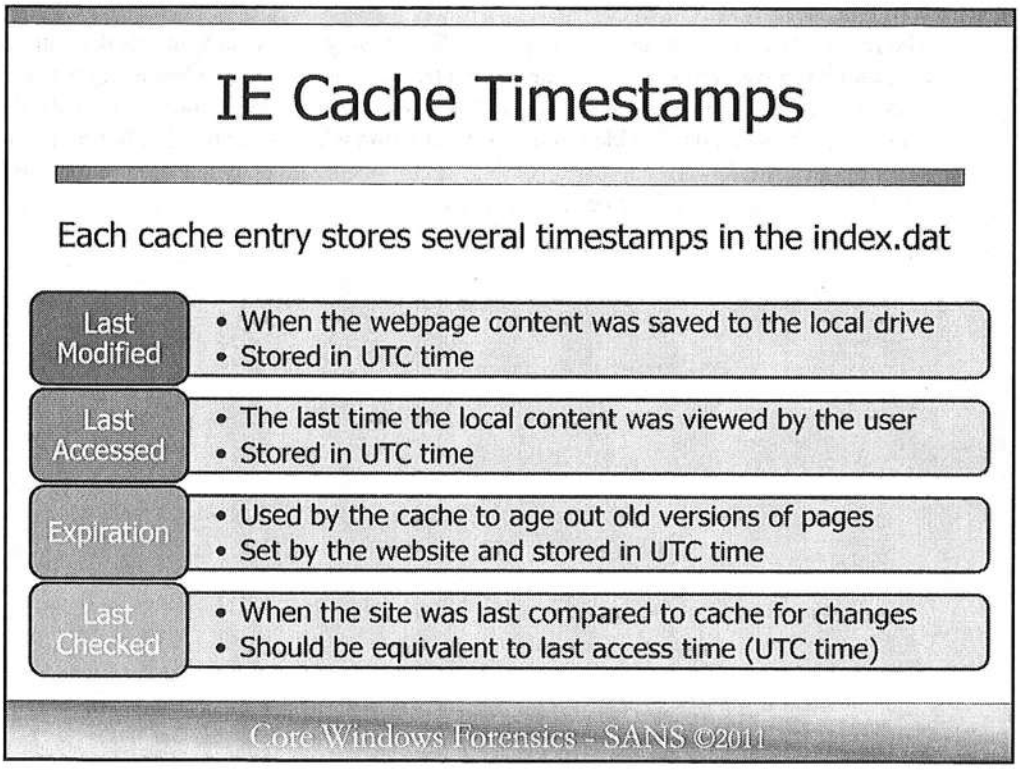
The **`index.dat`** file is the brains behind the IE cache. It allows us to make sense out of the seemingly random files stored by the browser and provides valuable metadata for each cache entry. The first priority for the **`index.dat`** file is to track the caching structure so that it knows which directories are allocated and holding cached files. Every time a new directory is added to the cache, the **`index.dat`** file is updated. Similarly, the **`index.dat`** file is responsible for mapping the files stored in the cache with their original website locations. So if a browser request is made for the index page of www.cnn.com, the index.dat allows the cache to be quickly searched for any locally stored files pertaining to that page.

The final role that the cache **`index.dat`** file plays, is to store metadata for each entry. The following are stored for each cached file:

- Original URL address
- Filename and path of the cached file (i.e. the cache directory is it located in)
- Type of cache record (URL, LEAK, REDR, HASH) – also found in other browser artifact **`index.dat`** files
    - URL is the most common type and indicates a cached entry triggered by a website visit. Most of the information relevant to an investigation will be gathered from this type of entry.
    - LEAK is not documented by Microsoft, but it appears to be caused by problems with the cache process - REDR means an entry was created due to the website redirecting the request to a different page or webserver. This tends to be for informational purposes only – files are typically not cached as a result of a REDR entry.
    - HASH is an entry used to store a part of the hash index for the index.dat, allowing faster searching of the file. The bigger the index.dat is, the more HASH entries it will maintain.

- HTTP header
  The return HTTP headers are stored as part of each entry's metadata, unless the entry did not originate via a web request. These are stored by the system so that when a request is fulfilled from the cache (meaning a local copy is used instead of the remote copy requested), the original headers can be provided to the browser along with the cached file to provide a seamless substitution. Headers can sometimes provide an investigator more information about what was returned from the webserver request.

- Timestamps will be covered on their own slide.




There are four timestamps which are kept for each cache entry. Timestamps are stored in the index.dat file and stored in binary, requiring a forensic tool like FTK to parse the values for the investigator. The timestamps are:

Last Modified – Indicates when the latest version of the page or file was saved to the cache. This can only tell us the last time a copy was stored – any earlier versions of this content will have been overwritten in the cache by the latest version.

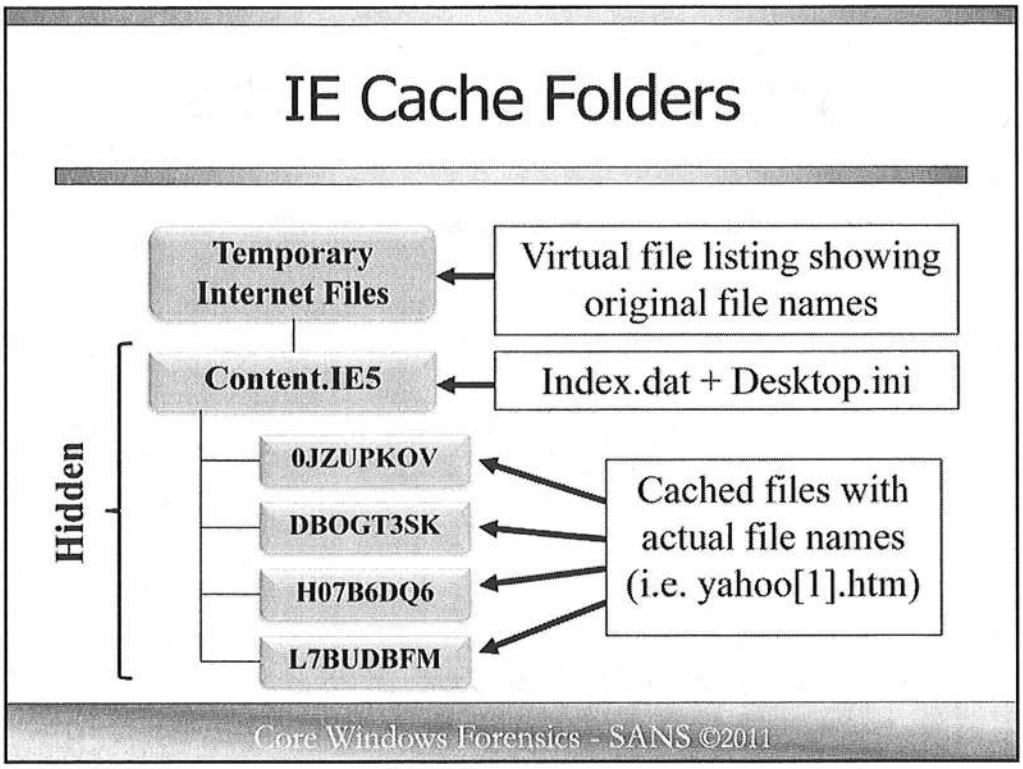
Last Accessed – Gives us the date and time that the cached content was last requested and viewed by the user.

Expiration – This is set by the website providing the content, allowing it to set up an expected lifespan for the page. Scavenger processes scan the cache frequently and eliminate any cached content which has expired.

Last Checked – Whenever a file request is made, the cache compares its version of the content with the latest version available on the website. This is typically done using the E-tag provided by the site in the HTTP headers. If the stored E-tag is the same as the one currently on the website, no changes have been made and it is okay to use the cached content. This timestamp records the last time this check was made. It usually is equivalent to the last access time, but could be different in the case of the user opting to not check for changes (to speed up browsing). [1]

**References**

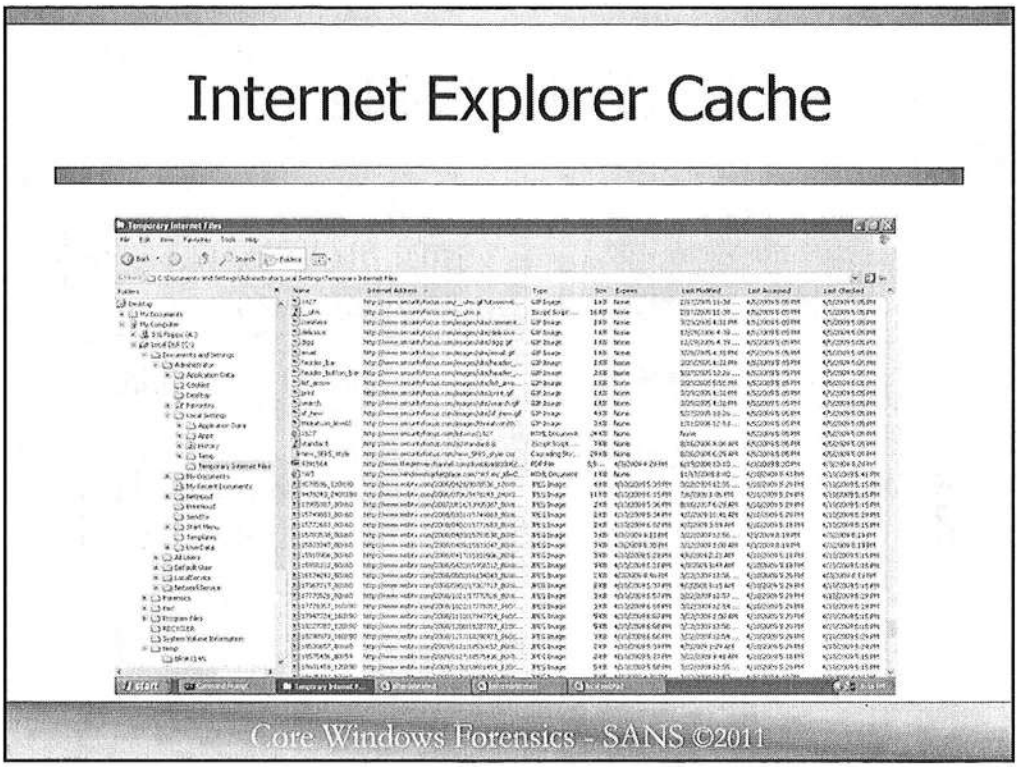
[1] http://support.microsoft.com/kb/263070




The cache uses a series of folders to store its data on the local system. The **`Temporary Internet Files`** folder is similar to the virtual folder structure created by Windows for the Internet history files. Within the Windows Explorer GUI, users can only see the **`Temporary Internet Files`** (TIF) directory, and even that is hidden unless they have chosen to unhide system files and show protected operating system files. Within the GUI, the TIF folder displays a neatly manicured listing of all of the files and information contained in the **`index.dat`** file and associated cache directories.

Behind the scenes, things are very different. The **`Content.IE5`** directory contains an **`index.dat`** file, which provides the metadata for each cached file. A series of at least four directories sits below the **`Content.IE5`** directory. In systems with a lot of browser usage and/or large caches, the number of directories can be much more than four. These directories are named in a psuedo-random fashion and kept track of by the **`index.dat`**. Each directory contains a part of the overall collection of cached files. Cached files are renamed by appending a bracketed number to their original name in order to ensure name collisions do not occur (i.e. **`facebook_icon[1].gif`** for the first saved copy of an image named facebook_icon). Any time a change is made to the cache files, the **`index.dat`** is updated accordingly.

The **`desktop.ini`** located in the **`Content.IE5`** directory is used to store the folder options, namely to hide the cache directories from the user.




The **`Temporary Internet Files`** directory is shown here as seen from the Windows Explorer GUI. The GUI parses the information within the cache **`index.dat`** file and displays it in this "virtual directory". This folder is hidden by default by the operating system.

A closer look at the directory gives us a nice view of the kinds of data stored within the cache. We see an HTML page for the Security Focus site along with a large number of related image files. Each entry has a series of time stamps telling us when it is set to expire, when it was last modified (downloaded from the web), when it was last accessed (viewed by the user), and when the site was last checked to determine if changes were made since it was cached. This information can be very useful for an investigator looking to corroborate Internet usage with other trace evidence like history files.

We also see in this slide that a **`PDF`** file is present within the cache. This could be a result of an e-mail attachment that was opened, or a file link on a website that was opened instead of saved.

Unfortunately we will rarely be seeing this view since the majority of our forensic analysis will not be conducted on live systems. Instead, we will need to rely on applications which manually parse the cache **`index.dat`** file to give us this information.

# Internet Explorer Cookies: Going Deep into Website Activity

- Cookies give insight into what websites have been visited and what activities may have taken place there
- They are small text files saved to the local drive to customize the website experience. An example is:
  `%userprofile%\Cookies\jdoe@expedia[1].txt`
- Cookies give an investigator the following information:
  - Issuing website (i.e. expedia.com)
  - Local user account (i.e. jdoe)
  - Modified, Accessed, and Created times for the cookie file
  - Any data the website wishes to include in the text file
- Remember that only persistent cookies are saved!



Cookies are critical to the web as we know it and provide an interesting repository of information to the forensic investigator. The HTTP (web) protocol is stateless, meaning that there is no collective memory to keep track of user activity on a website, or even within a specific web page. Cookies allow our authentication details to be remembered so they don't have to be entered on every page, and give us the ability to have things like a shopping cart, remembering all of the items we have selected for purchase. They are small text files (<4 KB) that are stored locally on the computer and passed to the proper website whenever a request is made. [1] In this sense, they can continue to be updated as a user navigates through a site. As anyone who has tried will tell you, not allowing cookies to be stored on your computer significantly degrades the web experience. Additionally, users often will preserve their cookies while regularly deleting their browser history and cache since the latter does not impact their web experience. In this case, cookies would be our best chance for identifying Internet activity on a system.

What can we derive through cookie analysis? From the cookie name itself we can usually determine the website and the local user account that visited site. A sample cookie filename follows for the site Trialpay.com in the Administrator profile:

```
administrator@trialpay[2].txt
```

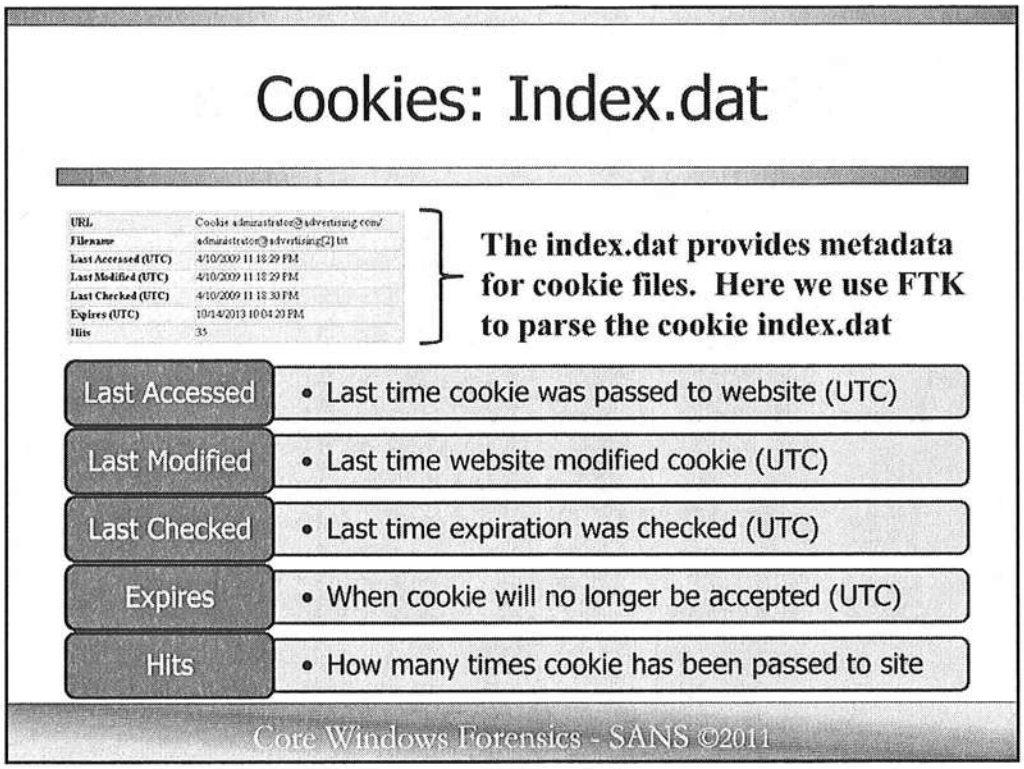
The cookie file will have standard operating system MAC times associated with it, giving us information about when it was created (first known visit to that page/site), when it was last accessed (last known visit), and when the cookie was last modified by the target website. The final piece of information that can be found in a cookie file is the data added by the website itself. This data is freeform and completely reliant on how the originating website chooses to store its information.

It is important to note that cookie files do not provide a complete record of Internet activity. Some sites do not require them, and others utilize "session" cookies instead of "persistent" cookies. Session cookies are only stored in memory and are only kept for the duration of the web session. [2] Persistent cookies have expirations, but tend to be more permanent and are stored locally on the hard drive.

**References**

[1] http://www.ietf.org/rfc/rfc2965.txt

[2] http://support.microsoft.com/kb/260971

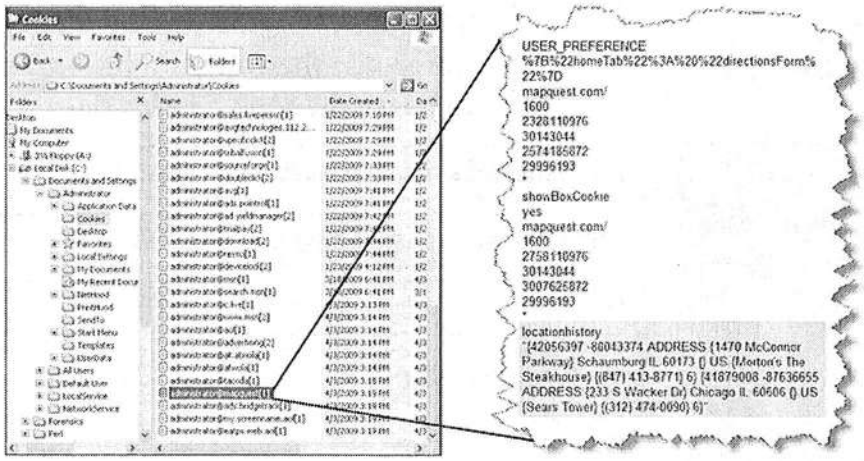



Internet Explorer uses the ubiquitous **`index.dat`** file to manage cookies as well. Cookies are simple text files that are saved in the **`%userprofile%\Cookies`** directory. An **`index.dat`** file exists within that directory and stores metadata for each cookie file. The metadata stored by the cookie **`index.dat`** is as follows:

- The URL that generated the cookie file.
- The local user account that accessed the URL (stored as part of the URL: **administrator**@advertising.com).
- The filename for the cookie text file.
- Last Accessed Time: if a cookie exists for a particular URL, it will be passed to that site on every request. The last access time tells us the last time the cookie was passed to the website. Stored in UTC time.
- Last Modified Time: Either the time the cookie was created, or the last time the site made a modification to it. Stored in UTC time.
- Last Checked: Indicates the last time that the expiration was checked for that cookie. Stored in UTC time.
- Expires: The expiration time for the cookie. It is set by the issuing website and indicates to the browser when that cookie will no longer be accepted by the issuing website. Stored in UTC time.
- Hits is the number of times that the cookie has been passed to the issuing website.

Similar to other types of **`index.dat`** files, references to deleted cookies can sometimes be found in **`index.dat`** entries. In most cases, the cookie file will no longer be present, but the metadata can still provide a wealth of information.

# Internet Explorer Cookies




Core Windows Forensics - SANS ©2011

Cookies can be difficult to decipher because the content is almost completely dependent on what the originating website wants to put in them. In most cases, such as session authentication cookies, the website does its best to make the cookie indecipherable and unrepeatable. At other times, an astute investigator can get lucky and pull out a wealth of information from the cookie contents.

The example in this slide shows a cookie generated by MapQuest.com. Knowing that MapQuest provides mapping information allows us to make some good guesses as to what action spawned the creation of this cookie. We see near the top that this cookie is associated with a "directions form". At the bottom, we see two addresses (one for Morton's Steakhouse and one for the Sears Tower). The other data in the cookie (such as the series of numbers) may or may not be relevant to our investigation, but would likely require analysis on a bigger set of Mapquest cookies to try to discern their meaning.

# Internet Explorer Bookmarks: Looking at Saved Locations

- Bookmarks tell us what websites a user has saved, giving insight into their interests and web activities
- IE stores bookmarks as `.url` files in the `Favorites` folder

  `%userprofile%\Favorites\Twitter.url`
- Favorites give an investigator the following information:
  - Saved website, page on site, and any URL parameters
  - What local user profile saved the favorite
  - Modified, Accessed, and Created times for the `.url` file
- A bookmark does not prove the site was visited
  - Bookmarks could be installed with OS or imported
  - Malware has been known to add favorites



Internet Explorer calls browser bookmarks, "Favorites" and stores them in similarly named folder. Favorites are created by the user and are shortcuts to websites that are frequently visited, or a user wants to save for a particular reason. Favorites have the file extension, "`.url`" and are simple text files. The following is the contents of a simple `.url` text file:

```
[InternetShortcut]
URL="http://support.dell.com/support/index.aspx?c=us&l=en&s=gen&~ck=mn"
```

In most cases if we find a bookmark that is relevant to our investigation, we can assume that the user both visited the site and actively made a choice to earmark it for future reference.

Favorites give us several pieces of information:

- The website of interest *and* the specific page (including any inline parameters) that was bookmarked
- They are stored within a user's profile in Windows and thus identify what account saved the Favorite
- The timestamps from the operating system give information on when the favorite was created, when it was last accessed, and when it was last modified.

A number of bookmarks are automatically created when Internet Explorer is installed. These largely point to Microsoft sites (or any sites added by the computer vendor) and should all have the same creation timestamp (typically when the last version of Internet Explorer was installed). Additionally, it is possible for a user to "import" favorites from outside sources by copying favorite files into their `%user profile%\Favorites` directory. In the latter case, we would need additional evidence to show that the user actually visited the site (an Access timestamp more recent than the Created timestamp, for example).

Some malware has been known to create Favorite files within user profiles without the knowledge of the user. Thus it is very important to take in the totality of the evidence. For instance, if several Favorite files were created within seconds of one another, this could be the result of malicious code. If the system has history files or cache entries from the bookmarked site, it is a good indication that the user could be the one who created the Favorite. Running anti-virus checks against the target system and checking anti-virus logs from the system are also helpful to identify resident malware.

A note of caution to investigators is to be careful when trying to view `.url` files (Favorites) on a live system. If you view them using a Windows Notepad or equivalent, they will display the HTML source of the website they reference instead of the information stored within the `.url` file. In almost all cases, browser artifacts should not be viewed through the live operating system.

# IE Download History: Examining What Was Downloaded

- How can we determine what a user downloaded / opened?
- Files downloaded within IE are often saved to default locations
  - The default download folder is located in the Registry (WinXP): **HKCU\Software\Microsoft\Internet Explorer**
  - The "Download Directory" value defaults to the user's desktop
  - Vista/Win7 uses **%userprofile%\Downloads** as the default
- Opened files (using "Open" option) are stored in IE Cache
- IE History will also show evidence of downloads



Internet browsers can be used to do much more than just view web pages. One of their key secondary functions is to facilitate the downloading of files from web or FTP servers. It is very common for an unsuspecting user to click on a hyper-link and be provided with the option to "Save" or "Open" a file. In many cases, the user pays little attention to where a file will be saved to and often has no idea that when opening a file that a temporary copy is saved to the local disk. Hence reviewing the default locations for these actions can provide excellent evidence during an investigation.

In Windows 2000 and XP, a Registry key controls the location of the default download directory. The key **HKCU\Software\Microsoft\Internet Explorer** has a value named **Download Directory** which is populated with **%userprofile%\Desktop** by default. When working with a disk image or "dead" system, HKCU registry keys will be located in a user's **ntuser.dat** hive file, located within their **%userprofile%** folder.

Windows Vista/Win7 significantly changed how default downloads are handled. The **Download Directory** value in the Registry is not present by default. Instead, each user profile has a **Downloads** folder, which is the default download location. This folder can reside anywhere on the disk and is changed by right clicking on the **%userprofile%\Downloads** folder and updating the information in the **Location** tab. In most cases the user does not go through the trouble of changing the defaults.

How do we know if the user downloads a file and then removes it from the default directory? In the case of a file being deleted, we could look for unallocated files still pointed to by that folder. Alternatively, the Internet History files could have a record of the file and the path it was downloaded to.

When a file is opened (instead of saved), a temporary copy is first saved to the local drive, and then opened with the proper file viewer from the operating system. These temporary files are stored in the IE Cache and are present even after the file viewer has been closed, providing an excellent location for file recovery that many users are ignorant of.

## IE Auto-Complete Function: What Was the User Typing?

- Address bar history - Typed URLs Registry Key:
  **HKCU\Software\Microsoft\InternetExplorer\TypedURLs**
  - Records last 25 web addresses typed by user
- History files are also used for Auto-Complete
- Internet Explorer Protected Storage
  - Auto-Complete form data
  - Website usernames and passwords
  - Network credentials and FTP Passwords
  - If it isn't acquired live, requires cracking
    - Nirsoft IEPassView v1.2 is a free tool for live acquisitions
  - More secure mechanism put in place for IE7, IE8, and Vista/Win7



The majority of browser artifacts that we look at in forensic investigations are used by the system in order to make the browsing experience better for users. History files, the cache, cookies, and bookmarks all make the browser experience more user friendly and efficient for the user. The auto-complete functionality is yet another example of this.

Internet Explorer saves the last 25 URLS that the user has typed in the IE address bar in the following registry key:

**HKCU\Software\Microsoft\InternetExplorer\TypedURLs**

TypedURLs values are used in conjunction with the IE History files to suggest recently visited sites to the user as they are typing in subsequent URLs. For an investigator, these values provide a valuable artifact showing intent for a specific user – we know that they typed in the web address (as opposed to just blindly clicking on a link). When working with a disk image or "dead" system, HKCU registry keys will be located in a user's **ntuser.dat** hive file, located within their **%userprofile%** folder.

The auto-complete function stores a variety of additional information within the registry. Values entered on web forms, usernames and passwords for websites, and credentials used to log into network resources are all stored (the latter two are only stored if the user asks IE to remember their passwords). Since this information is sensitive, IE stores it in encrypted form in the registry. This storage area is labeled by Microsoft as "Protected Storage" and has been in use since IE4. The primary registry key associated with it is:

**HKCU\Software\Microsoft\Protected Storage System Provider\<SID>**

Protected Storage has an inherent weakness which allows access to the unencrypted contents on a live system. A free-ware tool by NirSoft, PassView, is available to export this data out for the investigator. This is by far the simplest solution to accessing this data. If you are not able to get the information while it is live, it will need to be cracked using a tool like the FTK Password Recovery Toolkit.

Vista, IE7, and IE8 no longer support the use of protected storage due to its inherent weakness. Instead, they use a stronger encryption algorithm and store Auto-Complete passwords in a new location in the registry:

**HKCU\Software\Microsoft\Internet Explorer\IntelliForms\Storage2**

Website passwords in IE7 and above are now encrypted with the hash of the site they belong to. Thus, in order to (easily) crack the encryption the website must be known, or present in the History files. NirSoft has released a new tool, IEPassView v1.15, to reveal stored usernames and passwords when the website they belong to is known. [1]

Tread lightly when working with discovered passwords. You will rarely have the authority to log into a user's account, even if you are able to identify the username and password. In many cases the username is the most important item to recover because it can be used to identify the account used on the site and provide enough information to potentially subpoena the web service to for records belonging to that account.

**References**

[1] http://www.nirsoft.net/utils/internet_explorer_password.html

[redacted]

# IE6 vs. IE7

- Forensic methodology identical
- IE7 added new security features
  - Stronger encryption algorithm for stored passwords (no longer uses Protected Storage)
  - "Delete all" button added to clear browser artifacts
  - Improved wiping of index.dat files upon clear
- Deleted and orphan entries can still be found in cleared index.dat files



Internet Explorer 7 was released in 2006 and Microsoft has made a steady push to have it replace version 6, both because supporting multiple codebases is costly, as well as the fact that IE7 provides more user security. As far as our browser artifacts are concerned, the only change was in the auto-complete functionality of the browser, namely the shift from the old Protected Storage model to an updated encryption algorithm. All other artifacts such as the history, cache files, cookies, and favorites stayed the same. Additionally, IE7 utilizes the same directory structure as IE6 (and IE5), keeping backwards compatibility. Thus for forensic purposes, our methodology will be the same, regardless of whether we are analyzing artifacts from IE6 or IE7.

While our methodology stays consistent, IE7 does provide several new features which could impact the amount of browser residue that will be found. With its improved encryption, it is now more difficult to retrieve auto-complete data, particularly if the history files on the system have been cleared (the encryption algorithm relies upon knowing what website the data was saved for). IE7 simplified its `Tools->Internet Options` interface for clearing browser artifacts, providing a "Delete all" button. [1] In comparison, a user would need to find (and click) at least four different buttons in IE6 to accomplish the same tasks as the IE7 Delete All button performs.

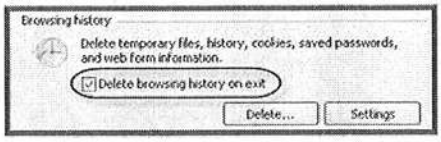
Clearing browser residue in IE6 was known to be somewhat ineffective because while it deleted files pertaining to the cache and cookies, it did not do a good job of wiping the entries in the index.dat files. IE7 addressed this problem by doing a much more complete job of wiping entries within the index.dat files. That being said, deleted entries can still sometimes be found and orphans, entries that for whatever reason do not have matching files, often are not cleared. Experience also indicates that entries of types LEAK and REDR are commonly found in the IE7 index.dat files after clearing.

**References**

[1] http://blogs.msdn.com/ie/archive/2006/01/12/512232.aspx

# Internet Explorer 8 (IE8)

- No file structure change from IE6/IE7
- New features provide additional forensic artifacts
  - Recovery Folders
  - Suggested Sites
  - DOM Storage




- Less evidence may be found due to new user options:
  - InPrivate Browsing mode eliminates some browser artifacts
  - Option to delete cache when browser is closed
  - Option to delete browsing history on exit

Core Windows Forensics - SANS ©2011

Internet Explorer 8 made its debut in 2009 and followed the trend of IE6/7 by not making any changes to the files it uses (i.e. index.dat database files) nor to the directories where browser artifacts are stored. Hence, similar to IE7, our forensic methodology remains unchanged.

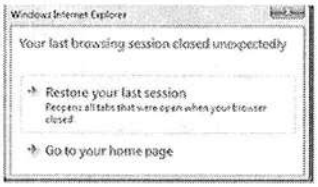
Several new features were added to IE8 that could result in less data being recovered during forensic analysis. InPrivate browsing allows users to switch to a new browser session that reportedly does not record any browser artifacts. We will find that this is not necessarily true. IE8 adds a few additional "clearing" features to the browser above and beyond what was added in IE7. [1] Now there are options that can be persistently set to clear the IE cache and history every time the user closes the browser. These options are not set by default, so a user must explicitly activate them. Similar to the features in IE7, cleared data is deleted, but not immediately overwritten, so it is quite common to find remnants of this data in unallocated space (through key word searches, etc.)

Recovery folders provide a wealth of information about sites visited and the tabs that were open in the user's last browsing session. Suggested Sites is yet another area which records sites visited by the user. And DOM Storage opens up a vast area for web applications to store data on the local system.

[1] http://msdn.microsoft.com/en-us/library/cc288472.aspx#_replace

# IE8 Session Recovery

- IE8 saves the current and last browser session
  - Provides crash recovery and "rollback" feature
  - Enabled by default and deleted when History is cleared
- Allow us to identify:
  - Tabs open in last session
  - Historical websites viewed in *each* tab
  - Referring websites
  - Time session ended
    - Modified time of .dat files in LastActive folder
  - Time each tab opened (only when crash occurred)
    - Creation time of .dat files in Active folder
  - HTML, javascript, and XML from the page
  - Other artifacts such as form data



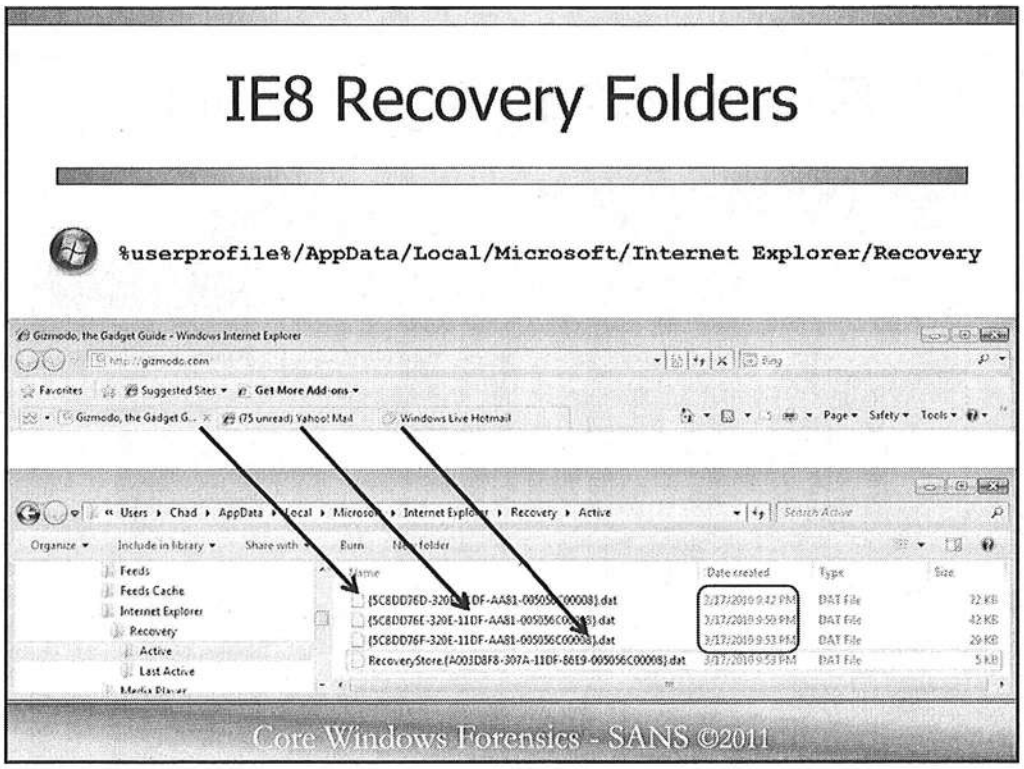

Core Windows Forensics - SANS ©2011

As part of their Trustworthy Computing initiative, Microsoft made a big push to make IE8 much more reliable [1]. A key piece is the new Automatic Crash Recovery features built into the browser. Amazingly, IE8 now continuously records all activity, including the actions taken in each individual tab, for the current and last browser session. As you might imagine, this is a goldmine of information that forensic investigators can use to reconstruct a user's actions and behavior within the IE browser. By analyzing the IE8 Recovery folder we can find:

- All of the tabs that were open in the last session (and the current session if a crash occurred)
- A historical record of websites viewed in *each* tab
- The referring websites for each of the above sites
- When the browser was closed
  - Modified times of files in the LastActive folder indicate when the last session ended
- Time each tab opened (only if a crash occurred and current session artifacts are available)
  - If files still present in Active folder (indicating a crash), Creation times show when each Tab was opened
- HTML, javascript, and XML from the page
- Other artifacts such as form data saved so IE can rebuild a crashed tab

Session Recovery is enabled by default. It can be disabled either through editing Group Policy or the local Registry. The GPO setting is "`Turn off Reopen Last Browsing Session`" (set to Enabled). In the Registry, a subkey must be added to the `HKCU\Software\Policies\Microsoft\Internet Explorer\Recovery` key named `NoReopenLastSession`. The value for the new subkey should be "1" to disable to feature. Session Recovery data is deleted when IE History is cleared, but the files are still recoverable using forensic methods.

[1] http://blogs.msdn.com/ie/archive/2008/07/28/ie8-and-reliability.aspx




Session recovery data is stored in the following folders:

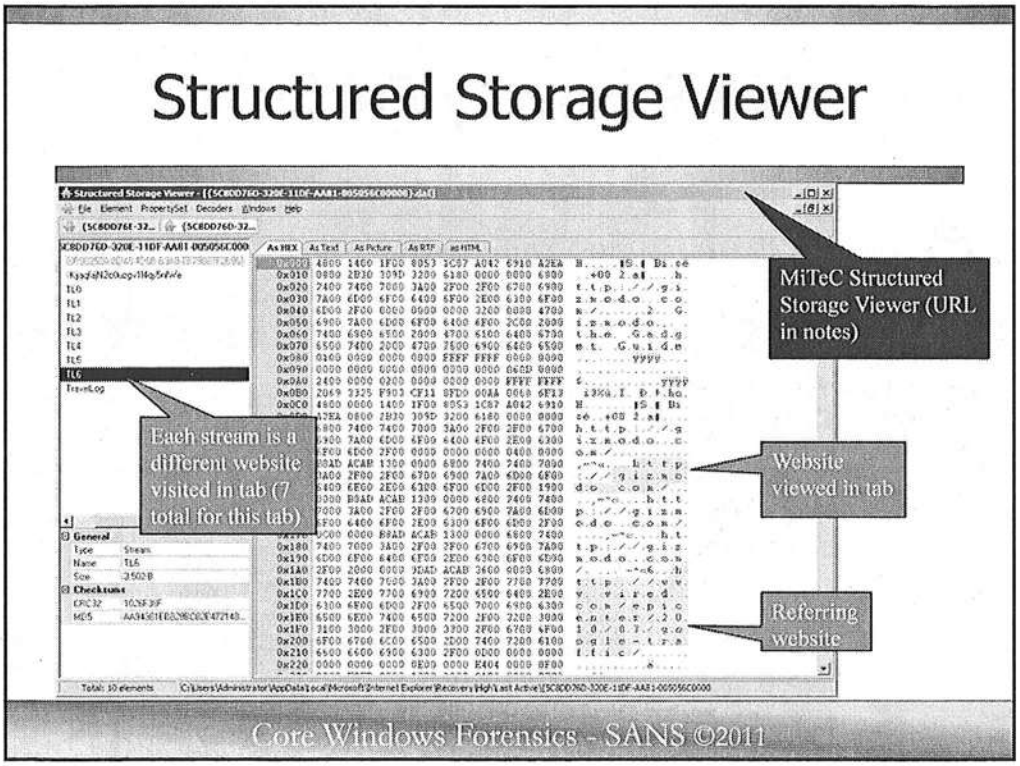
Vista/Windows 7
**%userprofile%/AppData/Local/Microsoft/Internet Explorer/Recovery/Active (Current Session)**
**%userprofile%/AppData/Local/Microsoft/Internet Explorer/Recovery/Last Active (Last Session)**

XP
**%userprofile%/Application Data/Local Settings/Microsoft/Internet Explorer/Recovery/Active**
**%userprofile%/Application Data/Local Settings/Microsoft/Internet Explorer/Recovery/LastActive**

Each Tab will have its own **{GUID}.dat** file, named using a unique GUID. This slide shows how three tabs within IE8 are mapped to their respective **.dat** files. The **RecoveryStore{GUID}.dat** file is the master file for the session, but has not been found to contain any usable artifacts.

This slide shows a live session in progress and hence files are present in the **Active** folder. Upon a stable IE shutdown, these files will be moved to the **LastActive** folder. If IE crashes, the original files will remain in the **Active** folder. Old, previously deleted **{GUID}.dat** files can be recovered from these folders using forensic methods.

The Creation timestamps in the **Active** folder show when each tab was opened (in this case the Hotmail tab was opened at 9:53 PM on 3/17/2010). When looking at the **LastActive** folder, the Modified timestamps indicate when the browser session was ended.




New Windows artifacts are employing the Structure Storage format more and more. Structured Storage provides an efficient means for incorporating multiple pieces of data within a single file [1]. It is essentially a compound file, storing files (or streams) within another file.

There are several free tools available for parsing files in this format. This slide shows the MiTeC Structured Storage Viewer looking at the tab we previously saw being used to view Gizmodo.com [2]. On the left are a series of streams that we can look at individually. What you are looking at is a historical record of sites viewed within this tab (during this session). Seven different sites were visited through this tab. For each stream (site), the viewer will show the stored data, including:

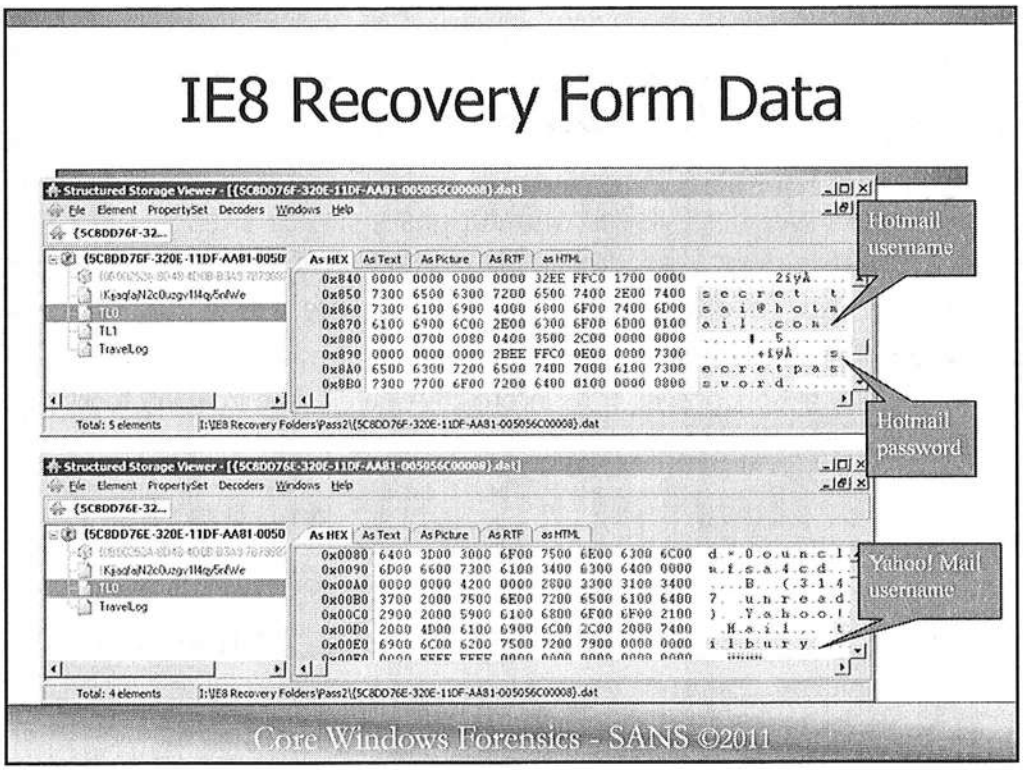
- Full website path
- Referring path
- HTML, javascript, and XML remnants necessary to reconstruct the tab
- Form data and other artifacts (research still pending)

Within the TravelLog stream is a history of forward and backward button use within each tab [3]. However, this artifact has yet to be decoded reliably. The stream with the seemingly random name (KjjaqfajN2c0uzgv....) tracks the most recently viewed stream, similar to a Registry MRU key.

[1] http://msdn.microsoft.com/en-us/library/aa380369%28VS.85%29.aspx

[2] http://www.mitec.cz/ssv.html

[3] http://blogs.msdn.com/ie/archive/2008/07/28/ie8-and-reliability.aspx




Since the specification for the `{GUID}.dat` files used in Session Recovery is not published, there is still much to be learned from these artifacts. One potentially useful artifact is the storage of data entered into forms. This allows IE8 to rebuild a tab exactly as it was in the case of a crash. While there is currently no easy means to extract this data, searching through the streams in each tab is recommended.

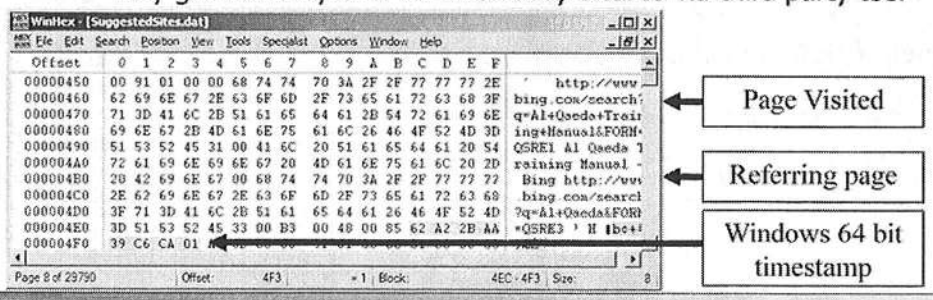
Two examples of form data findings are shown on this slide. The top `{GUID}.dat` file represents a tab being used for a Hotmail (web-based e-mail) session. Amazingly, we not only find the Hotmail username (secret.tsai@hotmail.com), but the account password (secretpassword) is also stored along with it! This is likely the result of a developer not following proper web standards for storing form data as password fields should be excluded.

The second example shows a similar session for Yahoo! Mail. While we don't find the account password, we do find the username of the account (tilbury – likely tilbury@yahoo.com).

These artifacts are all stored in Unicode. Thus we should expect that a good keyword list and string searching methodology would find relevant hits within these Structured Storage formatted files.

# IE8 Suggested Sites

- Tracks websites visited and suggests similar sites
  - **`SuggestedSites.dat`** file starts with 5MB of storage
- Does not record local, HTTPS or InPrivate usage
- *Should* be wiped along with History
  - May get out of sync or be incorrectly cleared via third party tool




Core Windows Forensics - SANS ©2011

Suggested Sites is a feature newly introduced in IE8. It aims to broaden the web surfing experience for users by suggesting sites related to those that they have previously visited. Users have the option to opt-out of the feature at installation time, but since most are uninformed, it is a good chance many will opt-in even if they never use the service [1].

When turned on, Suggested Sites records the majority of sites visited in IE8 and ships a sanitized version to a Microsoft aggregator which returns "suggestions". The visited sites are recorded in a file named **`SuggestedSites.dat`** which is created in the **`%userprofile%AppData\Local\Microsoft\Windows\Temporary Internet Files\Low`** **folder**. The file starts out at 5MB in size, giving an indication of the vast amount of information it is designed to hold. The **`SuggestedSites.dat`** is of a different format than the **`Index.dat`** History file, but records similar information. In particular it records the complete URL of the page visited, the complete URL of the referring page, and a standard Windows 64 bit timestamp of when the page was visited. There are no known tools currently available to parse this new file format, so analysis will need to be done within a hex editor.

The feature was designed to mitigate privacy concerns by not saving/sending references to local sites (such as internal webservers) and sites utilizing SSL [2]. Additionally, InPrivate browsing will not be recorded. When IE History is cleared, references are supposed to also be cleared from **`SuggestedSites.dat`** file, but the two archives can get out of sync. Additionally, many third party "cleaner" tools will clear history and leave the **`SuggestedSites.dat`** file untouched.


[1] http://blogs.msdn.com/ie/archive/2009/05/09/ie-setup-experience-just-the-facts-and-the-screenshots.aspx

[2] http://blogs.msdn.com/ie/archive/2009/02/05/suggested-sites-privacy.aspx

# IE8 InPrivate Browsing Mode

InPrivate http://www.cnn.com/

- InPrivate browsing mode affects all browser artifacts
  - Enabled via the Tools menu or by <Ctrl-Shift-P>
  - Opens a new browser session with tighter artifact restrictions
- History information for the session is not saved
- Cookie files are not created while in InPrivate Mode
  - All cookies are treated as session cookies
- TypedURLs and Form data are not saved
- Cache files are created but deleted at end of session
  - Cache Index.dat file is often not completely cleared
- Can be disabled using Group Policy or via the registry

Core Windows Forensics - SANS ©2011

InPrivate Browsing mode is one of the most significant additions to IE8. Dubbed "Porn Mode" by the press, it allows an IE8 user to shift into a private mode via the Tools menu or using the shortcut **<CTRL-Shift-P>**.[1] Fueling this designation is the fact that InPrivate mode cannot be entered when parental controls are activated on the browser. When engaged, InPrivate creates a new user session and takes extra steps to try not to save browser artifacts for the duration of the session.

InPrivate mode does an okay job of reducing the number of artifacts stored during a browsing session. Browser history information is kept in memory and not saved to the corresponding index.dat file. Instead of storing cookies to disk, all cookies are converted to session cookies, meaning they are only stored in memory. Form data is not saved to the Registry.

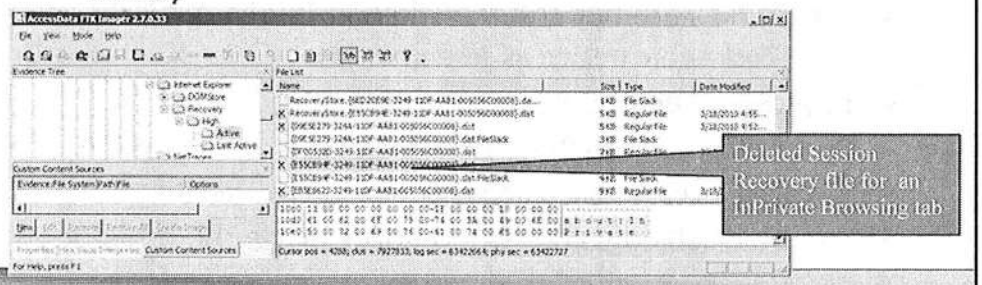
Cache files are not protected as well as many of the other artifacts. The InPrivate implementation still creates cache files as usual during the browsing session. Upon ending the session, all related cache files are deleted. Additionally, the cache index.dat file is not cleared properly and often retains some references to these deleted files, including URL and LEAK entries. Even without recovering the actual cache files, it may be possible to recreate many of the activities of the user by analyzing the names of previously cached files and the URLs they are associated within the **index.dat** file. Thus this turns out to be a significant data leak by the IE8 application.

Upon installation of IE8, InPrivate mode is activated for all users. System administrators can disable this option using either Group Policy, or by directory modifying the system registry. The GPO policy is found in: **Administrative Templates/Windows Components/Internet Explorer/InPrivate/Turn off InPrivate Browsing**. The registry key controlling InPrivate access is **HKLM/Software/Policies/Microsoft/Internet Explorer/Privacy/EnableInPrivateBrowsing**.

[1] http://www.microsoft.com/windows/internet-explorer/features/browse-privately.aspx

# Recovering InPrivate Artifacts

- Some artifacts can be recovered via file undeletion
  - Cache files
  - Session Recovery files
- Residue from all other artifacts can still be found
  - Unallocated space / Pagefile.sys
  - Memory




Core Windows Forensics - SANS ©2011

Microsoft advises that InPrivate mode was never designed to prevent artifact recovery using computer forensic tools. Forensic techniques are very powerful and given the current Windows architecture, any attempt at making a forensic-proof application would almost certainly be doomed to failure. As such, we should expect to be relatively successful at recovering information from InPrivate sessions.

Luckily, some of our most important artifacts are the least protected by InPrivate mode. Both cache and Session Recovery files are written to disk and removed via simple deletion at the end of a session. InPrivate browsing does not wipe any artifacts. **If the browser happens to crash or the system power fails, these artifacts are often not deleted at all.** The Session Recovery files in particular are extremely valuable because not only can they provide an excellent summary of sites visited during the InPrivate session, they can also be used to reconstruct what the tabs looked like and provide information on when the session started and stopped.

This slide shows an example of an InPrivate session that was started during an existing browser session. When InPrivate kicked off, a second set of Session Recovery Files were created and maintained. When the InPrivate session was closed, those recovery files were deleted as seen using FTK Imager. In this case, it appears there were three tabs opened in the InPrivate session. The highlighted file shows the original "about:InPrivate" tab that opens when a new InPrivate session has begun.

So what about IE History, Cookies, and Form Data? All of these artifacts are tracked in memory, and thus can be (at least partially) recovered from a memory image. Additionally, we would expect to see residue in the Pagefile and ultimately in unallocated space when artifacts are paged out of RAM and placed into virtual memory. In this case, InPrivate browsing does make our analysis more difficult since we often lose some measure of artifact attribution and meta-data when doing low-level recovery in unallocated areas and memory. Regardless, it would be very difficult indeed to completely hide web browsing activity, even using something like InPrivate mode.

# Browser Artifacts in Vista/Win7

- Vista made big changes to user folders
- Roaming versus Local folders
  - Cookies located in roaming profile
  - Cache & history found in local folder
- "Protected Mode" conducts web browsing as an unprivileged user
  - A new set of locations were added: **low** folders
  - Most browser evidence will be in **low** folders



With the release of Vista/Win7, Microsoft significantly changed the folder structure and mechanisms used by the operating system for user profiles. One of these changes was to make roaming profiles more explicit. Roaming profiles allow users to log onto other systems in the domain and have their profile information follow them. They have been around for many years, and in Vista/Win7, Microsoft decided to make what follows a user, and what doesn't follow a user much more explicit. Hence within a user profile in Vista/Win7, there are now two different set of folders: **Roaming** and **Local**. For our purposes, we want to be able to determine where our browser artifacts will be located in this new file structure. Traditionally Microsoft has included cookies in a roaming profile and excluded cache and history files by default. Thus, cookies are now found under the **Roaming** folder and history and cache can be found within the **Local** folder.

Similar to the steps required in XP, to see the **Local** and **Roaming** folders, users need to enable the **Show hidden files and folders** option and disable the **Hide protected operating system files** option within Folder and Search Options in the control panel.

The second major change within Vista/Win7 that affects us when performing browser forensics is the newly implemented "Protected Mode". [1] This was built as part of Microsoft's recent commitments to improving the security of the operating system and in a nutshell walls off most web browsing activities as if they were conducted by an unprivileged user. The idea is that if malicious code is run in the browser, it will not have the necessary privileges to cause harm to the operating system. Since not all activities using the browser will be unprivileged, a duplicate set of directories were necessary to store files from unprivileged use, called **Low** folders. An example of what this looks like in the file system is:

**%userprofile%\AppData\Local\Microsoft\Windows\History\Low\History.IE5**
(for the IE history files)

This means that when conducting our investigation within Vista/Win7, it will be necessary to look in two distinct areas: the standard locations (medium and above permissions) as well as the `low` (low permissions) folders. IE7 and IE8 have been updated to support Protected Mode when run on Vista machines. Most of our user Internet browser artifacts will be found in the `low` folders, but as we see with many things within Vista/Win7, residue files are sometimes inexplicably stored in the standard folders, so both should be analyzed for completeness. There are also a couple of notable exceptions:

- Local file usage is stored in the standard history folder (because it is not performed with restricted permissions)
- If Protected Mode is turned off, low folders will not be utilized
- If User Access Control (UAC) is turned off, low folders will not be utilized (it is required for Protected Mode to operate)
- If the instance of IE is run with Administrator permissions, the low folders are also not used

**References**

[1] http://msdn.microsoft.com/en-us/library/bb250462.aspx

# IE Data Locations - Vista/Win7

| | |
|---|---|
| **History Files** | • %userprofile%\AppData\Local\Microsoft\Windows\History\History.IE5<br>• %userprofile%\AppData\Local\Microsoft\Windows\History\Low\History.IE5 |
| **Cache** | • %userprofile%\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5<br>• %userprofile%\AppData\Local\Microsoft\Windows\Temporary Internet Files\Low\Content.IE5 |
| **Cookies** | • %userprofile%\AppData\Roaming\Microsoft\Windows\Cookies<br>• %userprofile%\AppData\Roaming\Microsoft\Windows\Cookies\Low |
| **Bookmarks** | • %userprofile%\Favorites |

Core Windows Forensics - SANS ©2011

This slide is a cheat sheet for browser artifact locations within Microsoft Vista/Win7 operating systems. As shown, for history, cache, and cookie files there are two different locations that must be searched and analyzed due to the Protected Mode function. Even the Temp directory (not shown on the slide) has two locations (an unprivileged, or **low** folder, and the standard folder). User favorites are the only primary IE browser artifact which utilize a single directory. The Vista/Win7 locations are as follows:

History= 1) **%userprofile%\AppData\Local\Microsoft\Windows\History\Low\History.IE5**

2) **%userprofile%\AppData\Local\Microsoft\Windows\History\History.IE5**

Cache= 1) **%userprofile%\AppData\Local\Microsoft\Windows\Temporary Internet Files\Low\ Content.IE5**

2) **%userprofile%\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5**

Cookies= 1) **%userprofile%\AppData\Roaming\Microsoft\Windows\Cookies\Low**

2) **%userprofile%\AppData\Roaming\Microsoft\Windows\Cookies**

Temp= 1) **%userprofile%\AppData\Local\Temp\Low**

2) **%userprofile%\AppData\Local\Temp**

Bookmarks= 1) **%userprofile%\Favorites**

# IE Forensic Methodology

1. Do not review browser files on a live system
   a. Consider running tool to retrieve protected storage data
2. Begin Profiling Internet Activity Using Forensic Image
   a. Analyze history index.dat files
   b. Review contents of cache
3. Fill in any gaps by reviewing additional artifacts
   a. Bookmarks
   b. Cookies
4. Export files that will be reviewed with 3rd party tools
   a. Ntuser.dat registry hives to review relevant artifacts
5. Save relevant findings to include in report

Core Windows Forensics - SANS ©2011

Once we know what is available to us when doing Internet Explorer examinations, our process becomes relatively simple. This methodology assumes that we will be performing at least part of our analysis using a forensic image of computer system and a forensic tool like the Forensic Toolkit (FTK). This is a reasonable assumption because most forensic suites on the market do a decent job of identifying and parsing IE data.

There is always a temptation when encountering a live system to poke around to see if any smoking guns are readily visible. Since many of our investigations involve illicit web activities, this temptation is even greater when preparing for browser forensics. This is particularly the case in IE investigations because of the pretty virtual file directories that Windows creates for its history and cache files. This is a bad idea for several reasons. Most importantly is that you can affect evidence on the system. A great example of this is when reviewing cache files. Often cached pages do not have every piece present in the page, and clicking on them will cause the browser to go out to the web to retrieve them, adding items and overwriting evidence on the live system. Windows takes great pains to hide relevant data on a live system. As an example, favorites do not show actual link data when opened via Windows Explorer GUI. You'll also find it impossible to see many of the index.dat files from the GUI. All of that being said, if you do have access to the live system, you may consider running a tool like NirSoft IEPassView to recover auto-complete registry information which may be difficult to decode from an image file.

The first place most browser investigations start is with reviewing the history and cache files. Pound for pound, these files give you the greatest breadth of information about what Internet activities have taken place on the computer system. FTK does an excellent job of parsing these **index.dat** files and facilitating the review process.

As we have seen in this section, there are plenty of other artifacts which can provide data above and beyond what is in the history and cache files. These artifacts are particularly useful if the history and cache files have been cleared! Different artifacts have different longevities on the system, and many times you will find website references within bookmarks or cookies that have long since been expired out of the history file. Checking the default download and temp directories is another example of finding long lost files which may or may not be references within other artifacts.

It is hard for any one forensic tool to be able to do it all, and many times you will want to export out specific artifact files from the image to run in other forensic tools. Previous students have spoken about doing this to use Mandiant Web Historian to create nicely formatted spreadsheet files that they can quickly provide to HR or their chain of command. Finding IE information like TypedURLs and the default download directory will require reviewing the registry hives. These can be exported and reviewed in your favorite registry viewer.

Finally, you will find that in these types of cases you can have an overwhelming amount of information. Recording and cataloging as you go will great assist with putting it all together in your final report. Tools like FTK can help you "bookmark" key findings as you go.

# Internet Explorer Review

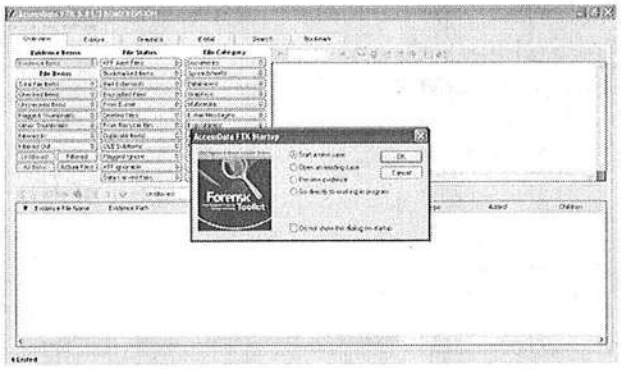
| Browser Artifacts | Where to find them : IE 6, 7, & 8 |
|---|---|
| **Internet History** | History index.dat<br>SuggestedSites.dat (IE8 only)<br>Session Recovery {GUID}.dat (IE8 only) |
| **Cache Files** | Cache index.dat; content.ie5 folder |
| **Cookies** | Cookie .txt files; Cookie index.dat |
| **Bookmarks** | .url files |
| **Download History** | History index.dat |
| **TypedURLs** | Registry |
| **Auto-Complete / Form History** | History index.dat<br>Protected Storage / Registry |
| **Web Passwords** | Protected Storage / Registry |

Core Windows Forensics - SANS ©2011

Internet Explorer has a reputation for being relatively easy to examine, and that is largely due to its familiarity. When looking at the components of a successful review, a different picture emerges. IE offers a wide range of browser artifacts that can be examined, but we often need to look in multiple places to get all of the data and metadata for a specific artifact. The IE cache is a good example of this with its index.dat containing metadata and the content.ie5 folder containing the (renamed) cached files. Timestamp information is inconsistent (like in the history index.dat files) or reliant on the filesystem MAC timestamps outside of the artifact metadata (like cookie and favorites files). Several of the key artifacts utilize the same filename, index.dat. This can simplify things when searching a filesystem or unallocated space, but it is important to keep track of where you found index.dat files so you have some idea of what to expect see as results from your forensic tools. Finally, our artifacts are split between well hidden files in the filesystem and information stored within the registry (such as auto-complete data). Luckily the market share and consistency of Internet Explorer artifacts over the different IE versions has afforded most forensic tool vendors the ability to maintain good solutions for finding and parsing these artifacts. When studying Firefox we will see that this is not the case with that browser.

# HANDS-ON: Examining Browser Artifacts (1)

1. Open Forensic Toolkit (FTK)



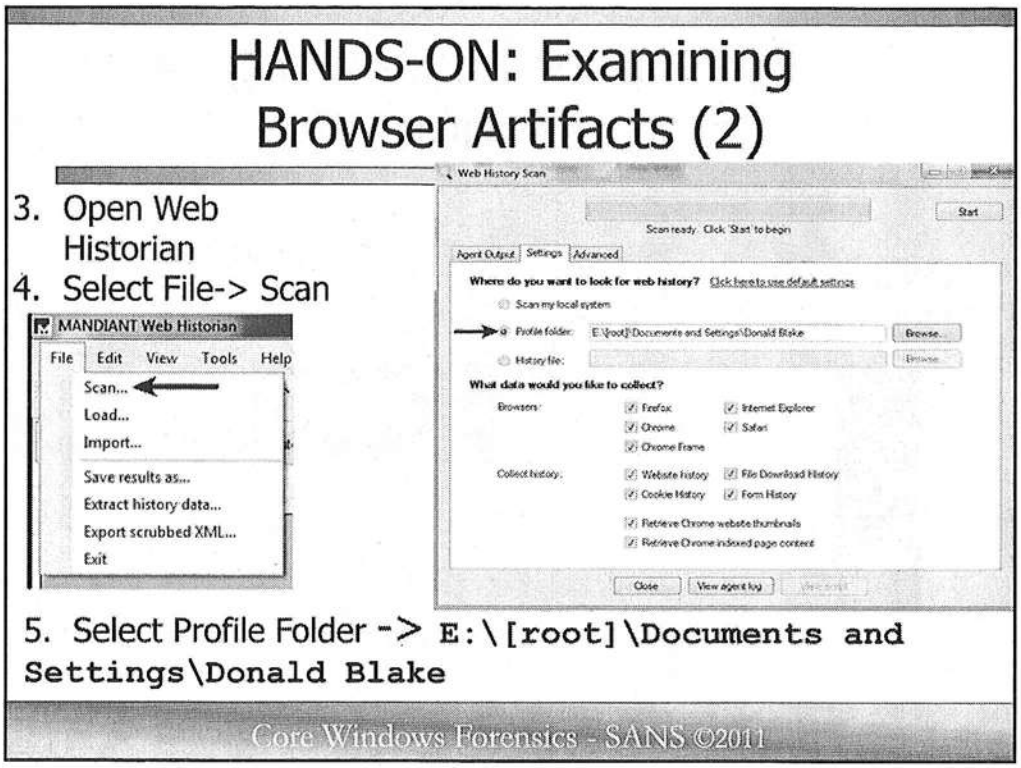
2. Open your Donald Blake case within FTK

Core Windows Forensics - SANS ©2011

We will be conducting this hands-on exercise using the Forensic Toolkit (FTK). FTK will allow us to quickly find and parse the relevant Internet Explorer data within our image file.

Find the FTK shortcut on your SIFT desktop and open FTK. You will need to make sure that you are not using the Demo Version because the image you will be examining has much more than the 5000 file limit.

Once FTK starts, open the previous case that you created using the xp_dblake.dd image. If you can't find your case, go ahead and create a new one using the FTK Startup dialog presented when you load FTK.




We will be using the image you mounted using FTK Imager to practice supplementing our FTK analysis with a third-party tool. Specifically, we will be using Mandiant Web Historian to scan our mounted image and parse history files into human-readable format (in this case HTML format that we can view in our browser).

Web Historian is a program that allows an investigator to collect, display and analyze web history data using Mandiant Intelligent Response (MIR) technology. Beyond this simple definition, Web Historian seeks to provide a **customizable yet simplistic** interface to view and navigate voluminous amounts of web history data. Perhaps the most powerful feature is the ability to **correlate** and provide multiple views of the data (including graphical and timeline) through the Analyzer and Web Profiler tool, in the hopes that investigators can come to well-informed conclusions about the data quickly.

Supported browsers:

- Firefox 2, Firefox 3+
- Chrome 3+
- Internet Explorer 5-8
- Safari 3+

**Features**

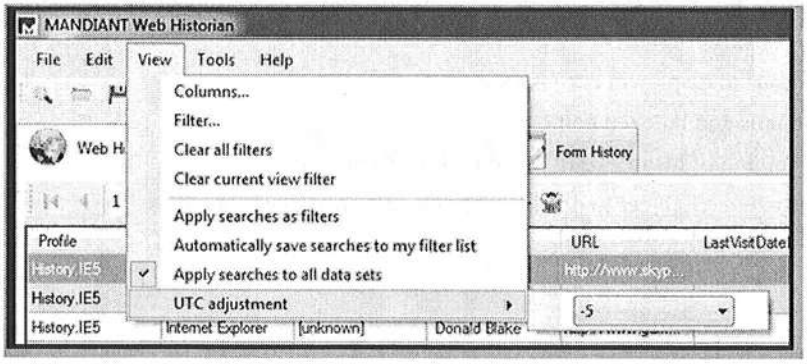
- Collects web history, cookie history, file download history, and form history into data sets
- Simple/powerful UI based on tabbed organization of datasets
- Perform a scan of one or more user profiles on a local or remote system

- Perform a scan of one or more arbitrary history files from all supported browsers
- Import results from existing XML scan documents
- Data displayed in gridview style with full search, sort, and filter capabilities
- Custom filters can be created and applied to one or more data sets
- Export data sets to XML, HTML or CSV
- Extract and export history data from any folder
- Quick copy/paste selected gridview rows to clipboard
- Customizable scan settings can tweak the scan to target specific browsers and data sets
- Right-click context menu for narrowing gridview data instantly
- Select which columns to display in each dataset
- View page thumbnails and indexed content
- Export sanitized version of history results to distribute to others
- Visualize datasets using Analyzer to draw bar graphs, pie charts and timelines
- Website Profiler shows a quick "report card" of artifacts for various websites
- Specify UTC offset for consistent representation of date fields

Start Mandiant Web Historian and select "**File->Scan**". Then select the Profile folder of drive letter of your local mount point for the image (selected through FTK Imager). Select the "Donald Blake" profile under E:\[root]\Documents and Settings\Donald Blake. Finally, click the **Start** button to parse the file. One of the nice features of Web Historian is that it will scan the entire drive looking for any relevant history files from several different type of browsers. Remember that every user on the system will have their own history files, so pay attention to the path of the history file you are examining so you can tie those results back to the appropriate user account.

# HANDS-ON: Examining Browser Artifacts (3)

7. Select View-> UTC Adjustment to UTC-5 (EST)






However, with the exception of one special case, in the end, all browsers report dates in Universal Time Coordinated (UTC). Internet Explorer stores the last visit date (termed last accessed date) for weekly index.dat files (stored in the MsHist* folders) in local time. For "live" scans – that is, scans that do not specify a specific path to scan via the "Profile folder" or "History file" scan settings - this is not a problem. In that case, Web Historian knows the history files it is examining came from the local system, so it can safely convert this local time to UTC, adjusting for time zone changes as well as daylight savings time (if observed on that system). The problem arises when the "Profile folder" or "History file" setting is used, Web Historian has no way of guaranteeing that the history files in that path or file came from the local system. Therefore, it can't use the local system's timezone information to adjust the stored local time to UTC. This is where the user must intervene, and hence the option to set the UTC adjustment value.

In this case the local time that the PC was used in was EST5EDT. Once you set this value all the dates will be displayed normally.

So how do you know which UTC value to pick? The answer is based on two factors:

1. What time zone is the system that contained the target history files configured to use
2. Whether or not the system was configured to observe Daylight Savings Time (DST)

For example, if the history files came from a system configured to report time in Eastern Standard Time (EST), the UTC/GMT offset would be UTC-5. However, if the system is configured to use Eastern Daylight Time (EDT), the offset would be UTC-4 during the summer and UTC-5 in the winter.

# HANDS-ON: Examining Browser Artifacts (4)

**Answer the following questions below in your notes:**

1. What directory are the History and Temporary Internet Files in?
   a. What operating system does this indicate?
2. What Favorites did the user Donald Blake create?
3. Reviewing Donald Blake's cookies, where would you surmise he lives or travels to?
4. Export user Donald Blake's history index.dat files from the image and use Mandiant Web Historian to review the data
   a. Is any local file access recorded in the history files?
5. Review the IE cache files and index.dat for user Donald Blake
   a. What webmail provider is he using?

   **Bonus:** What is his e-mail account for the webmail provider?

   **Bonus:** What restaurant did he make reservations at?

Core Windows Forensics - SANS ©2011

1. What directory are the History and Temporary Internet Files in? [redacted]

   - What operating system does this indicate?

2. What Favorites did the user Donald Blake create?

3. Reviewing Donald Blake's cookies, where would you surmise he lives or travels to?

4. Export user Donald Blake's history index.dat files from the image and use Mandiant Web Historian to review the data.

   - Is any local file access recorded in the history files?

5. Review the IE cache files for user Donald Blake

   - What webmail provider is he using?

**Bonus:** What is his e-mail account for the webmail provider?

**Bonus:** What restaurant did he make reservations at?

# Answers

1. What directory are the History and Temporary Internet Files in?
   **C:\Documents and Settings\Donald Blake\Local Settings**
   a. What operating system does this indicate? **Windows XP (or before)**
2. What Favorites did the user Donald Blake create?
   **CNN.com, ESPN.Com, opentable.com**
3. Reviewing Donald Blake's cookies, where would you surmise he lives or travels to? **Colorado (Denver)**
4. Export user Donald Blake's history index.dat files from the image and use Mandiant Web Historian to review the data
   a. Is any local file access recorded in the history files? **Yes - a large amount**
   b. What restaurant did he make reservations at? **Blue Star Restaurant**
5. Review the IE cache files and index.dat for user Donald Blake
   a. What webmail provider is he using? **Yahoo! Mail**

   **Bonus:** What is his e-mail account for the webmail provider? **dblake_personal**



Exercise Answer Notes:

1. If this were a Vista system you were analyzing, the History and Temporary Internet files would have been located in the directory: **`%userprofile%\AppData\Local\Microsoft\Windows\`**. Remember that you would also have to look in the low folders for both of the artifacts as well!

2. There are several .URL files in the favorites directory. Most are default entries included upon installation. A closer analysis of the time stamps indicates that the files for CNN.com, ESPN.com, and opentable.com were created after the file system was installed (created on 1/13/09 and likely install date was 6/30/2007).

3. Two of the cookies have data referencing Denver, including the cookies for weather.com, craiglist.org. There are also cookies present for Vail resorts (in Colorado).

4. There are many references to local files in the history files. You should be looking for references like: **`file:///F:SECRET/SECRET.zip`**

1. The cache index.dat file includes a reference to mail.yahoo.com as its first entry. A review of the XML files pointed to by the mail.yahoo.com entries will show a username of dblake_personal@yahoo.com.

   Further review of the history index.dat files will show several visits to opentable.com. In one entry, the title of the page visited is Blue Star Restaurant.

# Mozilla Firefox

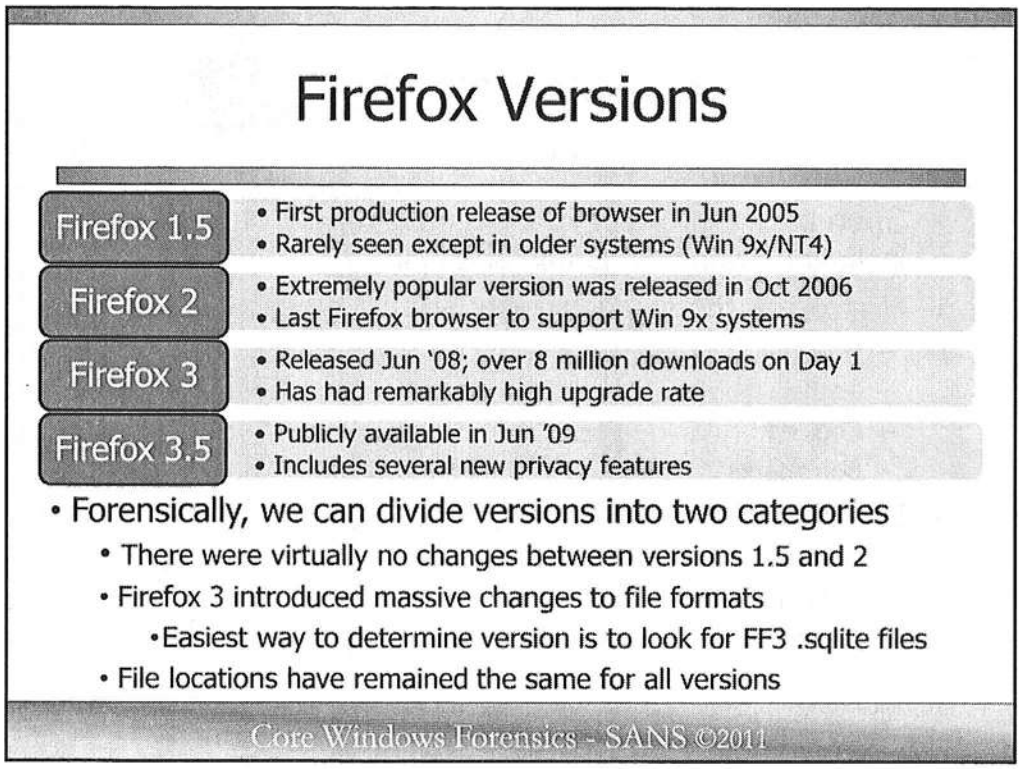
- Open-source browser running on a variety of platforms including Windows, Linux, and MacOS
- Firefox has the same browser artifact types as IE
  - History
  - Cache
  - Cookies
  - Bookmarks
- Firefox makes some of our job easier, and some harder
  - No registry entries are used
  - Files are centrally located
  - Database files are complicated and not human-readable



Firefox is the second most popular browser in the world, after Internet Explorer. It is an open-source project sponsored by the non-profit Mozilla foundation. Its popularity has been fueled by innovation (such as tabbed browsing) and the wide range of third-party plug-ins for the browser platform. Unlike Internet Explorer, Firefox runs on a wide array of operating systems, including Windows, Linux, and the MacOS. We will be focusing on Firefox installed within Microsoft Windows, and thus any specific file locations, application defaults, and examples discussed will be valid for Windows machines. For Firefox versions installed under other operating systems you should expect minor differences.

Since we are working primarily with Windows in this course, it is a common occurrence to find both IE and Firefox installed on the same machine. This is primarily because IE is installed by default, but you will often find that both browsers are being used for different purposes (for example, IE for accessing the internal company intranet and Firefox for general browsing).

While their implementations differ, Firefox has the same combination of browser artifacts that IE does. History, cache, cookies, bookmarks, and auto-complete data are all present in the Firefox browser. In some instances, Firefox can give us even more evidence than IE, particularly in Firefox version 3. However, that information comes at a cost and we will see in this section that Firefox utilizes database structures that are not trivial to parse. We will discuss several tools available to assist investigators with retrieving relevant data from these database structures. On the positive side, Firefox makes our lives easier by keeping browser files within a profile directory and by not utilizing registry entries. This makes it very easy for us to export Firefox data to be processed on our forensic workstations.




Firefox has seen steady development since its inception. The first production release of Firefox was version 1.5, which is rarely encountered. Firefox version 2.0 brought the browser into the mainstream and adoption of Firefox has increased steadily since its release. Firefox 3.0 broke the Internet record for most downloads in a day, with over 8 million recorded. Mozilla has done an excellent job of encouraging users to upgrade their browsers, hence the number of users still using Firefox 2 is quite low. The versions were released as follows:

Firefox 1.5 – June 2005

Firefox 2.0 – October 2006

Firefox 3.0 – June 2008

Firefox 3.5 – June 2009

For the purposes of forensic analysis, Firefox versions can be divided into two categories: Firefox 3 and pre-Firefox 3. All versions have kept the same file locations . While there were few changes to the file structures themselves in versions 1.5 and 2.0, Firefox 3.0 introduced massive changes to all of the key browser files we will be analyzing in our forensic reviews. In short, version 3.0 adopted a new database structure for storing browser data called SQLite, that we will review in this section. With the exception of a couple small privacy features, all FF 3.x versions have the same artifact components and locations.

When performing our analysis, we typically will not have the luxury of loading up a subject's computer to view the application versions being used. An easy way to tell if the Firefox being used on a system is version 3.0 (or higher) is to search for the existence of **`.sqlite`** files in the Firefox **`profile`** directory. This format was not introduced until version 3.0. Another method to find the exact version is to review the **`prefs.js`** file in the **`profile`** directory. Within this file will be a value under "**`extensions.lastAppVersion`**". This value corresponds to the current version of the software installed. For example:

```
user_pref("extensions.lastAppVersion", "3.0.9")
```

# Where to Start: Firefox File Locations

**2000 / XP**

**History – Cookies – Bookmarks – Auto-Complete**

- `%userprofile%\Application Data\Mozilla\Firefox\Profiles\<random text>.default`

**Cache**

- `%userprofile%\Local Settings\Application Data\Mozilla\Firefox\Profiles\<random text>.default\Cache`

**Vista/Win7**

**History – Cookies – Bookmarks – Auto-Complete**

- `%userprofile%\AppData\Roaming\Mozilla\Firefox\Profiles\<random text>.default`

**Cache**

- `%userprofile%\AppData\Local\Mozilla\Firefox\Profiles\<random text>.default\Cache`



Knowing the location of the key browser files should be your starting point when conducting browser forensics. The first thing they tell us is if Firefox is installed on the system. Remember that we will most often be reviewing a dead file system copy and thus won't have all of the handy indicators present on a live system, such as shortcut files or entries on the start bar. Searching for specific files and folders used by an application can be the easiest way to determine if artifacts exist. (the Windows registry is another useful place to search). Applications often leave behind files and folders even after they have been uninstalled on the system. A secondary reason for knowing where the application folders are is that you will need to know where to point your forensic tools. Many of the tools we will be using for Firefox forensics will require the investigator to identify where the key browser files are located.

Remember that in Windows, the **`Application Data`** (**`AppData`** folder in Vista/Win7) is hidden by default. To see the contents of this folder, you will need to go into the Control Panel, select Folder Options, and click "Show hidden files and folders" in the View menu.

File locations changed for Windows Vista/Win7 installations. Similar to what is done using the **`Local Settings`** directory in XP/2000, Vista/Win7 splits the Cache directory out and places it within the **`Local`** directory, ensuring that it is used only on the local system not saved with any roaming profiles used within a domain environment. This same distinction is made in IE installations and one of the reasons it is done is that **`Cache`** folders can be quite large in size and thus should not be passed around the network. Firefox does not utilize the "Protected Mode" feature of Vista/Win7 and thus does not utilize **`low`** folders as seen in IE7 and IE8.

The **`<random text>.default`** folder within the Firefox file path is a Firefox profile folder. A user can maintain multiple Firefox profiles and if that is the case, you will see multiple **`<random text>.default`** folders within the **`Profiles`** directory. In this case, files in each folder will need to be examined individually.

# Firefox 1.5 / 2 Profile Files

- Many key browser artifact files are stored in Mork format
  - Mork is an obscure database format that is largely text based
- Bookmarks, downloads and cookies are stored in text format
- Cache files are stored using multiple database files
  - We will need tools to easily parse Firefox data!

| | |
|---|---|
| `history.dat` | History |
| `bookmarks.html` | Bookmarks |
| `downloads.rdf` | Download History |
| `formhistory.dat` | Saved Form Data |
| `cookies.txt` | Cookies |
| `_CACHE_<XXX>` | Cache files; XXX = MAP or a 3 digit number |



Files in Firefox are notorious for being difficult to parse and analyze. The Firefox 2 history and formhistory files are stored in Mork format, which is a complicated text file with database like organization. The complaints about the Mork format are legion, and likely were an impetus for the new file format we see in Firefox 3. [1] In short, a forensic tool is essential to parse the Mork format and review the **`history.dat`** and **`formhistory.dat`** files.

**`Bookmarks.html`** is stored in HTML format, making it very easy to review this file. However, if you are reviewing this file without a forensic tool, make sure to do so using a text editor (as opposed to a browser), because some of the key bookmark attributes are only displayed in the HTML tags which are not displayed by the browser.

Similarly, **`cookies.txt`** and **`downloads.rdf`** are stored as text files, but their attributes are best understood when using a dedicated tool to parse the data.

Finally, files in the **`Cache`** directory are stored using a very complicated database structure. Cache data is stored in binary format and thus can only be read using a tool created for that purpose. [redacted]

**References**

[1] https://bugzilla.mozilla.org/show_bug.cgi?id=241438

# Firefox 3 Profile Files: SQLite

- Small, self-contained database format based on SQL
- A majority of Firefox data is stored in SQLite Databases
- SQLite is a binary format with a complicated table structure -- we will be relying upon tools to parse
- The most important Firefox 3 database files are:

| | |
|---|---|
| `places.sqlite` | History – Bookmarks – Auto-Complete |
| `downloads.sqlite` | Download History |
| `formhistory.sqlite` | Auto-Complete Form Data |
| `cookies.sqlite` | Cookies |

- The Firefox cache was not updated to SQLite from FF2



Firefox 3 chose a completely different format to store a majority of its key browser information. This caused a majority of forensic software capable of parsing Firefox 2 files to need to be rewritten. The Firefox developers chose the SQLite format, which is a standard format used in applications where lightweight database qualities are desirable, but the added overhead of installing and administrating a database are not. SQLite is a binary format and requires a tool to accurately identify information within it.

The key benefits for Firefox are that the SQLite format affords much faster searching than the previous Mork format, allowing more data to be stored and accessed efficiently. This turns out to be a boon to the forensic investigator, because more data means more browser artifacts that can potentially aid an investigation. Firefox 3 developers used the new format to include more attributes for every browser artifact.

Nearly every browser file that we will be working with was switched to the SQLite format in version 3. In the case of the **`places.sqlite`** file, multiple files were combined into one (history, bookmarks, and some form history data is all kept in this one file). The only critical structure not changed to the SQLite format was the cache. The cache was left alone and has the same structure as was used in previous Firefox versions. Thus tools that will read the Firefox 2 cache, will also read the cache files in Firefox 3.

For both Firefox 2 and 3 file types, the key takeaway is that it is useful to know what format your browser artifacts are stored in, but 99% of the time you will be using a forensic tool to assist you with parsing that information and conducting your analysis.

# History Artifacts in Firefox: Investigating Sites Visited

## Firefox maintains more history info than IE

| Investigative Questions | Firefox 2 history.dat | Firefox 3 places.sqlite |
|---|---|---|
| What was the complete URL that was visited? | URL | url |
| What was the title of the page visited? | Name | title |
| When was the site first visited? | FirstVisitDate | visit_date* |
| When was the site last visited? | LastVisitDate | visit_date* |
| How many visits were made to the site? | VisitCount | visit_count |
| Was the URL typed by the user? | Typed | typed |
| Was the page retrieved w/o any user actions? | Hidden | hidden |
| What page led the user to this one? | Referrer | from_visit |
| How did the user request the page? | N/A | visit_type |

Core Windows Forensics - SANS ©2011

History information is one of the most important browser artifacts for the investigator. Firefox maintains a wealth of data which can be an invaluable resource for profiling a user's Internet activities. It actually records more data than Internet Explorer, as evidenced by the "hidden", "from_visit / referrer", and "visit_type" fields. As an extra bonus, the default number of days within Firefox to store history data is 90 days.

In Firefox 2, history information is stored in the history.dat file, and the fields within that file are displayed in the second column of this slide. [1] Firefox 3 uses the file **`places.sqlite`**, shown in column three, and adds one extra attribute called "type". [2] The attribute names within the table indicate what the fields are called within these history files (in the event that manual parsing is required).

It is easy to get overwhelmed by the amount of data available, so the best way to review history is to first focus on the URLs. The URL and title give us a quick view of the types of sites that are being visited. The URL stored will be the complete URL, including any sub-pages and parameters that were passed to the website. This often allows us to pinpoint exactly where in the site a user was visiting. As sites are identified as possibly relevant, you can then delve deeper into all of the data surrounding that URL. As an example, sorting by the "visit_count" gives a good indication of the user's favorite sites. A visit count of greater than one should be a good indication that the site was not visited accidently.

If you are interested in a particular time range of activity (for example you have WebSense logs showing inappropriate activity happening on a certain day), then it may be worthwhile to sort by the time. Firefox 3 keeps a record of every visit to a site, allowing us to create a timeline of activity. It only records one date, "visit_date", which can then be used to identify the first and last visit since each visit is recorded. Firefox 2 only records the first and last date visited, making it critical to seize a copy of the system as soon as possible to prevent the "last" dates from being overwritten with more recent activity. When looking at the raw data (i.e. without using a tool), keep in mind that the dates stored are in PRTIME, which is a 16 digit number representing the number of microseconds since midnight, 1 January 1970 (UTC). [3] As an example, the value 1240021077618504 equates to 8:17:57 PM, 4/17/2009.

The final attributes give further information about how a site came to be stored in the history file. The "typed" field tells us whether the user physically typed the URL into the browser (equivalent to the TypedURLs registry key in IE). The "hidden" flag indicates whether the page was retrieved automatically – without a specific request by the user. The "from_visit" attribute tells us what page (if any) the currently stored page was request from. Finally, the "visit_type" field provides valuable information about how a page was requested (from a link, a bookmark, etc.).

**References**

[1] http://kb.mozillazine.org/History.dat

[2] http://kb.mozillazine.org/Places.sqlite

[3]http://developer.mozilla.org/en/PRTime

## How Was the Web Page Requested? Visit Types in Firefox 3

The type field gives us specific information about how the web page in the history file was requested

| Type | Description |
|---|---|
| 1 | User followed a link and the page was loaded |
| 2 | User typed the URL to get to the page (with or without auto-complete) |
| 3 | User followed a bookmark to get to the page |
| 4 | Indicates some inner content was loaded such as images and iframes |
| 5 | Paged accessed due to a permanent redirect (HTTP 301 status code) |
| 6 | Paged accessed due to a temporary redirect (HTTP 302 status code) |
| 7 | File indicated by history was downloaded (non-HTML content) |

Core Windows Forensics - SANS ©2011

The "type" field is new to the history file (`places.sqlite`) as of Firefox 3. [1] It is important to understand because it gives very granular information about why a specific URL was recorded in the history. This can prove very valuable when attempting to prove that a user had the intent to visit a specific web page, and provides insightful information as to how they got there.

There are seven different values for the type field, representing the universe of possibilities for a URL being recorded in the history file. We will expand on each of them below.

Type 1: The user clicked on a link and a new top-level window was populated with the new web page

Type 2: The user physically typed the URL – if the URL was previously typed and stored within auto-complete, they could have typed part of the URL and selected the correct URL for the choices provided by auto-complete

Type 3: A bookmark was used to retrieve the page. Keep in mind that the bookmark may have been since deleted.

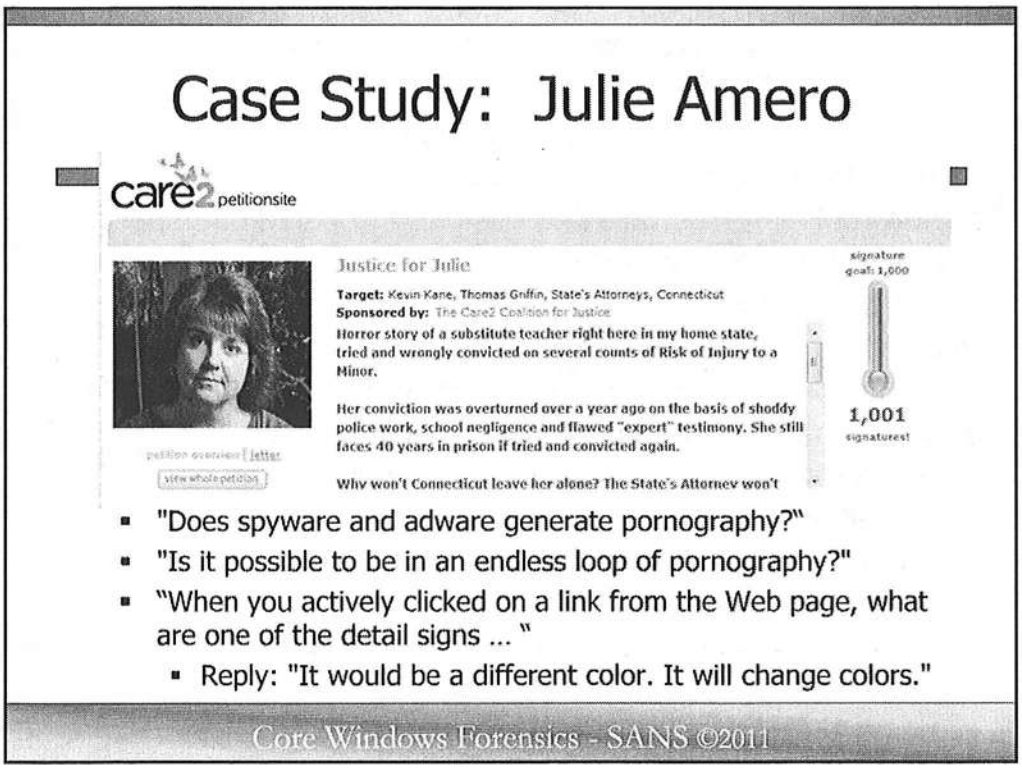
Type 4: Dynamic content was loaded as a result of a page being visited. This is likely not content specifically requested by the user (the original page visited should be a separate entry preceding it in the history file)

Type 5/6: The page requested by the user was not displayed, and instead the webserver redirected to a different page (used for load balancing, invalid page requests, re-organization of the website, etc.) Redirects are tricky because it is possible for a malicious site to redirect a user to a completely different domain that they did not intend to visit. The "from_visit" attribute or other history file entries may provide more information, or it may require more research to determine how the redirect occurred.

Type 7: This indicates an entry that was recorded due to a download taking place. In many cases, the name of the file will be recorded in the URL field, providing a great means to identify what files have been downloaded.

**References**

[1] https://developer.mozilla.org/en/NsINavHistoryService




The State of Connecticut vs. Julie Amero case is a great case to take a moment to discuss. Much of the key evidence hinged on browser forensics, and hence it is very relevant to this class. While it is dangerous to armchair quarterback a case without all of the facts, this case has received enough expert scrutiny that it appears to be a classic case of an incomplete forensic review having a very lasting impact on someone's life. Here is a quick summary of the case:

Julie Amero was a substitute teacher for a seventh grade classroom. The classroom had a computer and while she was in control of the classroom, began displaying pornographic popups in full view of the students. The computer was seized by the local police and analyzed using forensic preview software (the ComputerCop Pro application). The investigating officer had limited forensic training and during testimony made statements indicating a very limited understanding of forensics and malware. Amero originally faced up to 40 years in prison for endangering minors. A jury convicted Amero, but the judge set aside the verdict and granted a new trial. After a total four years and facing a second trial, Amero agreed to a plea bargain of a misdemeanor count and lost her teaching license [1].

The analysis of the case is fascinating and well worth a read [2]. One of the key mistakes made in the prosecution's case was misunderstanding browser artifacts (another was not checking for malware, but that is another story). Neither side denies that a large number of pornographic sites were visited. Thus the History and Cache files were populated with the sites and content accessed. The point of contention is whether Julie Amero clicked on those links or they were instead populated by malware. *Finding a history entry, cached file, cookie, or even a bookmark does not automatically indicate that the account owner knowingly accessed a site.* Malware has been known to add bookmarks, spawn popups, and even use javascript or open hidden iframes that access sites without a user's knowledge. Firefox's Visit Types get us much closer to being able to answer some of these questions, but they are not perfect. In testing, I have seen Firefox incorrectly labeling a visit as Type 1 (user followed a link) when it was actually a hidden javascript applet that did so. It is imperative that all the evidence be gathered and analyzed before making any definitive conclusions.

[1] http://www.usatoday.com/tech/columnist/andrewkantor/2007-03-16-julie-amero-update_N.htm

[2] http://www.sunbelt-software.com/ihs/alex/julieamerosummary.pdf

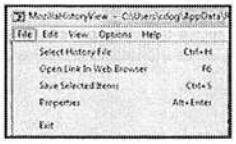
# Using NirSoft Tools – Step by Step

NirSoft tools are freeware and designed to parse one specific browser artifact (i.e. browser history)

- Default to displaying data from live system

**Process:**

1. Export browser files to forensic system
2. Open the exported file within the viewer
   - Example: **`File -> Select History File`**




3. Highlight relevant entries using **`<ctrl><left-click>`**
4. Export selected items for inclusion in report
   - **`File -> Save Selected Items`**
   - **`View -> HTML Report - Selected items`**

Core Windows Forensics - SANS ©2011

We use a variety of NirSoft tools in this course for parsing and reviewing Firefox browser artifacts because many forensic suites do not do a good job of parsing Firefox data. NirSoft tools are freeware and regularly updated. The tools tend to be dedicated for a single purpose, such as parsing Firefox history files. They use a common interface, making it easy to use any of the tools once you learn one of them. We do need to keep in mind that the tools default to parsing and displaying from the live system they are run from. In a forensic investigation, we will rarely be using these tools on a live system. Thus, we will need to point the tools to whatever files we have exported onto our forensic workstation (and keep in mind that before we do this, any tool output will be displaying data from our live forensic computer).

The NirSoft tools used in this section work on all versions of Firefox. They do not require any installation, making them very portable. Our process for using these tools is as follows:

1. Export browser files to forensic system

In most cases, we will be exporting the profile folder from our evidentiary image file onto our forensic system for review.

2. Open the exported file within the viewer

In this step, we point our viewer to where we have saved the exported profile files. This overrides the default setting to view data from the live system. We do this using the File menu and telling it to select whatever type of file that viewer is expecting. As an example: File -> Select History File for the MozillaHistoryViewer application.

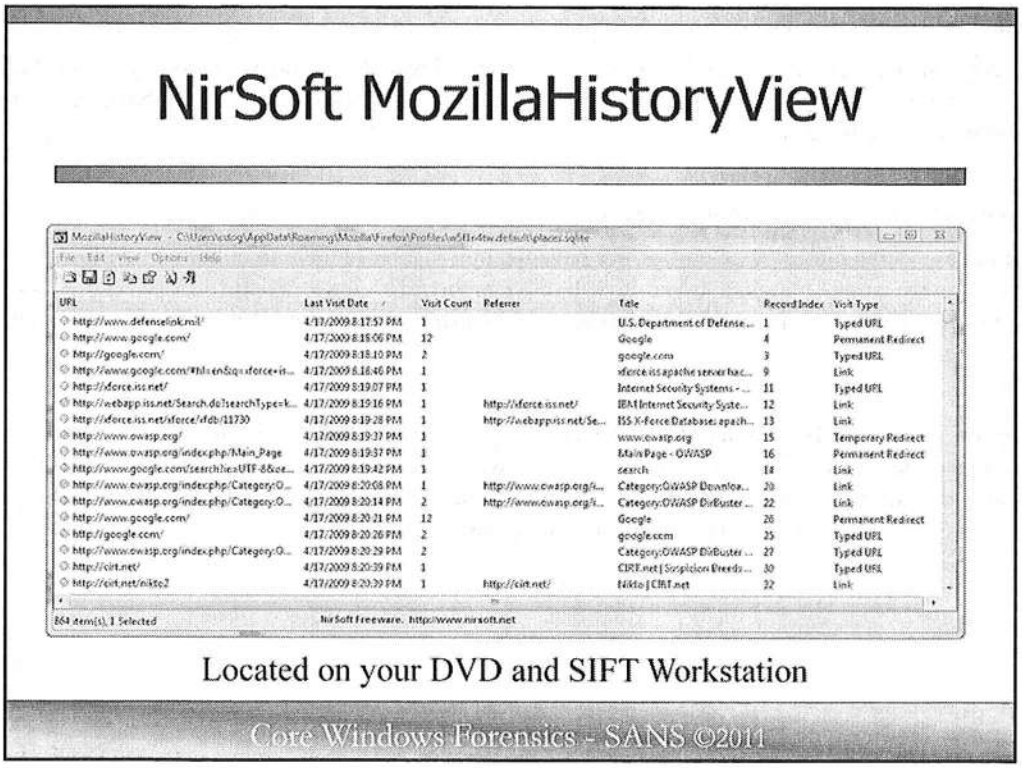
3. Highlight relevant entries using <ctrl><left-click>

While conducting our review, we will highlight any entries relevant to our investigation by left clicking on them. To select multiple entries, use the <ctrl> key. Some of the tools will have check boxes next to each entry that can be used instead.

4. Export selected items for inclusion in report

Once we have selected our relevant entries, we have the option to save them to either a text file or to a HTML file that we can then include in our investigative report. To save as a text file, use File -> Save Selected Items and to view/save in HTML format, use the menu option View -> HTML Report – Selected items.

Some NirSoft tools have the option (when relevant) to open a selected URL within a web browser. Keep in mind that this option retrieves the page from the actual webserver. In some cases this could be undesirable because we may not want to make a connection to the webserver (imagine a case where the subject is the owner of the site, or if the webserver is suspected of hosting malware).




MozillaHistoryView parses Firefox history database files and presents the data in a user-friendly format. [1] We will be directing the tool to open the **`history.dat`** and/or **`places.sqlite`** files that we have exported to our forensics system.

The columns displayed by the tool correspond to the data fields stored in the history file:

URL – The URL that was visited and stored in the history file
First Visit Date (not shown) – The first occurrence of the URL in the history file (this is N/A for Firefox 3). By default, time is shown in the local time of the machine it is being viewed on. **`Select Options -> Show Date/Time in GMT`** to see UTC time as stored in the history file.
Last Visit Date – The last occurrence of the URL in the history file (equivalent to visit_date in Firefox 3)
Visit Count – Number of times the site was accessed during the time the history was being recorded
Referrer – The page the user was at before the recorded URL was accessed
Host Name (not shown) – The fully qualified domain name of the website that was accessed
Title – A short description of the page visited (provided by the page itself)
Record Index – Shows the order of entries within the history file
Visit Type – Indicates how the page was accessed (via link, bookmark, typed URL, redirect, etc.) – not present in Firefox 2

One important thing to keep in mind is that we may see local file references in the history file. As an example:

```
file:///C:/Program%20Files/VMware/VMware%20Workstation/help/index.html
```

Unlike IE, a local reference stored in the Firefox history file means that the local file was viewed using the Firefox browser.

**References**

[1] http://www.nirsoft.net/utils/mozilla_history_view.html

# Firefox Cache: Viewing the Browser Stockpile

- The cache is a repository for web data a user has viewed
- Firefox maintains more cache metadata than IE
- Cache format and location is consistent in FF1.5-FF3
  - Located in `local` folder in Windows Vista/Win7
- Default cache size is 50MB

**What can we find in the Firefox cache?**

| URL | Num Times Fetched | Is the File Present? |
|---|---|---|
| Filename | File Type | File Size |
| Last Modified Time | Last Fetched Time | Request Header |



The Firefox cache is another important artifact for an examiner to use when investigating a user's Internet activities. Inside the cache, there are often a large amount of website pages, files, and images which have been stored locally to speed up future requests to that same website. This gives the investigator a window into not only what sites have been visited by the user, but also the actual content files which were downloaded from those sites. The default cache size for Firefox is 50 megabytes, providing a large amount of evidentiary data that can be analyzed for each profile.

The cache format and location is the same for all versions of Firefox,

%userprofile%\Local Settings\Application Data\Mozilla\ Firefox\Profiles\<random text>.default\Cache

with the exception of Firefox installations on Windows Vista/Win7, where the cache has been relocated to the local folder:

%userprofile%\AppData\Local\Mozilla\ Firefox\Profiles\<random text>.default\Cache

The database format used by the cache is very complicated, and is best parsed using a tool created for that purpose.

For each item cached by the browser, metadata is stored to provide context. The most interesting metadata to an investigator are:

URL – What web page the given content was saved from

Number of times fetched – How often the cached content has been used

Missing (Is the file present?) – Tells us if the items the metadata refers to are actually stored in the cache. When data is missing, it is usually a result of cache control parameters set by the webserver requesting that no data be stored (i.e. no-cache)

Filename – The name of the content that was retrieved from the website

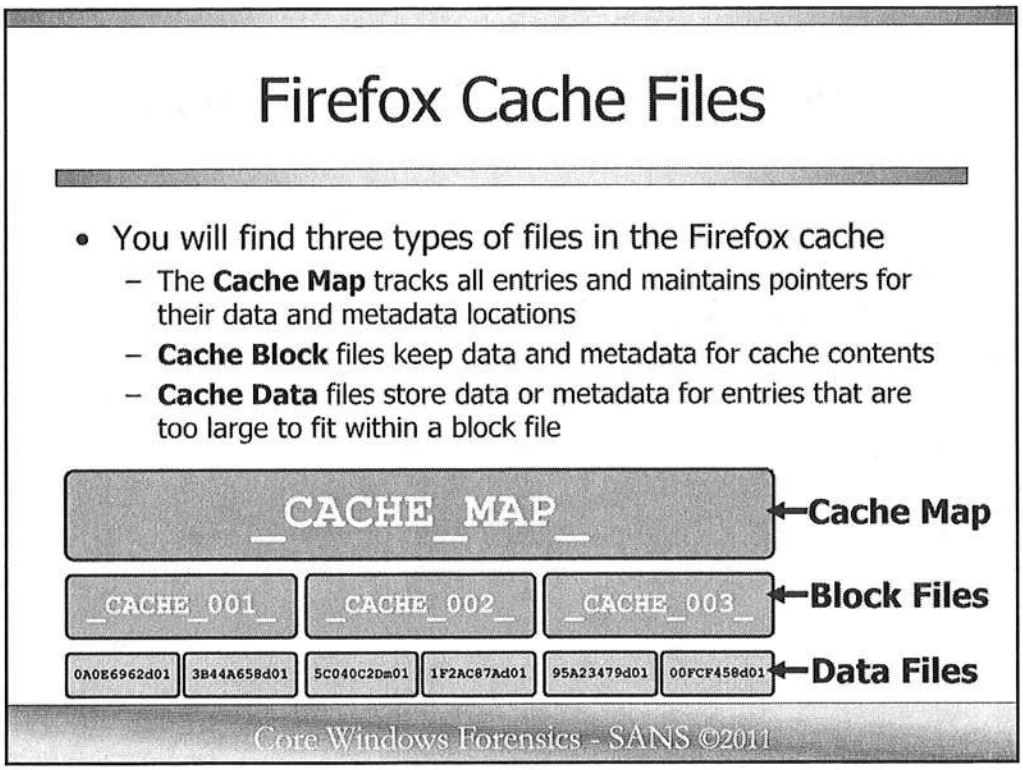
File Type – The type of file stored in the cache (i.e. HTML, jpg, gif, javascript, etc.)

File Size – How large the downloaded file is

Last Modified Time – The time and date that the content was last stored to the cache.

Last Fetched Time – The time and date that the content was retrieved (used) from the cache. This gives us an indication of when the page the cached content is from was last visited.

Request Header – The HTTP header from the requesting website. This is used by the cache to "replay" the request when cached content is used instead of actually going out to the site to do another download. Firefox stores the full header, which provides much more information than what is stored in the Internet Explorer index.dat file. Stored within the header is the content encoding information, cache parameters, the webserver name, the response code generated by the webserver, and the webserver time when the request was made.



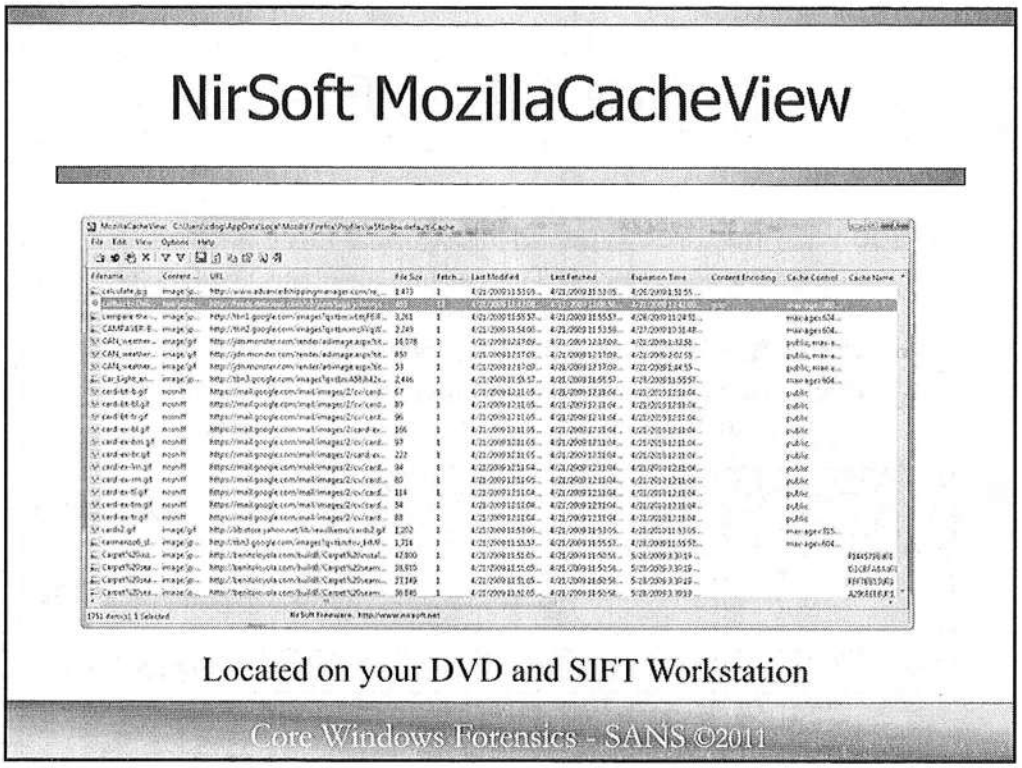

When looking in the cache directory, you will find three different types of files. [1] These files together make up the cache database. The Cache Map is the most important file because it keeps track of everything that is currently stored in the cache along with where the actual data lies (i.e. what cache file is holding it) and what file is maintaining the metadata for each entry. The map file is necessary because there are two different places where the cache data and metadata can be stored. The preferred location is in a Cache Block file, which can be thought of as a container holding a large number of cached files and metadata. If the data or metadata for an entry is too large to fit within the space allocated in the Cache Block file, a separate Cache Data file will be created to hold that data.

While the complicated structure of the cache makes analysis difficult (without tools), it has the benefit of being sufficiently obscure for most users to be unaware that their files are being stored there. Unlike the Temporary Internet Files directory used by IE, there is virtually no way for a Firefox user to look at the cache directory and determine what files are kept in the cache. We will see that there is an easy way to do so within the browser itself, but users generally ignore the cache, leaving a wealth of information for the investigator.

Knowing what a Firefox cache looks like is important when we are exporting files for offline analysis, but you will almost never need to parse the files manually. For that we will use tools build specifically for this purpose.

**References**

[1] http://www.securityfocus.com/infocus/1832




Parsing the various cache files is a very difficult process, so we will need to turn to a tool to do it for us. MozillaCacheView takes the various binary cache files and allows the contents of the cache to be reviewed in a user-friendly fashion. [1] Recall that the cache format remains consistent through all versions of Firefox, and thus this tool works for any version.

The columns displayed by the tool correspond to the metadata store for each file in the cache:

Filename – The name of the file that was downloaded from the webserver

Content-Type – The file type of the saved content

URL – The specific URL the file was downloaded from

File Size – The size of the downloaded file

Fetch Count – How many times the file has been retrieved from the cache (how often it has been requested)

Last Modified – When the file was saved to the cache

Last Fetched – When the file was last requested / retrieved from the cache

Expiration Time – Time set by webserver indicating when cached content should be automatically deleted

Content Enconding – Tells the browser what type of encoding (if any) the downloaded file has been subject to. Gzip encoding is very popular as a means of reducing the amount of data that must be transferred.

Cache Control – From the request header; indicates any cache parameters set by the webserver for the downloaded content. Identifies expiration times, and whether the content should be saved at all (i.e. no-cache parameter)

Cache Name – If the entry is stored in a Cache Data file, the name of the file is displayed

Server Name (not shown) – From the request header; indicates the name of the webserver providing the content

Server Last Modified (not shown) - From the request header; tells when the content was last changed on the webserver

Missing (not shown) – Indicates whether the data is currently stored in the cache, or if the cache entry is just for reference. Entries are typically marked missing when the webserver has requested that they not be stored (Gmail often does this for privacy purposes)

E-Tag (not shown) - From the request header; provides a means for the cache to determine if content has been updated on the webserver since the data was cached

Server Time (not shown) - From the request header; the time on the server when the file was downloaded

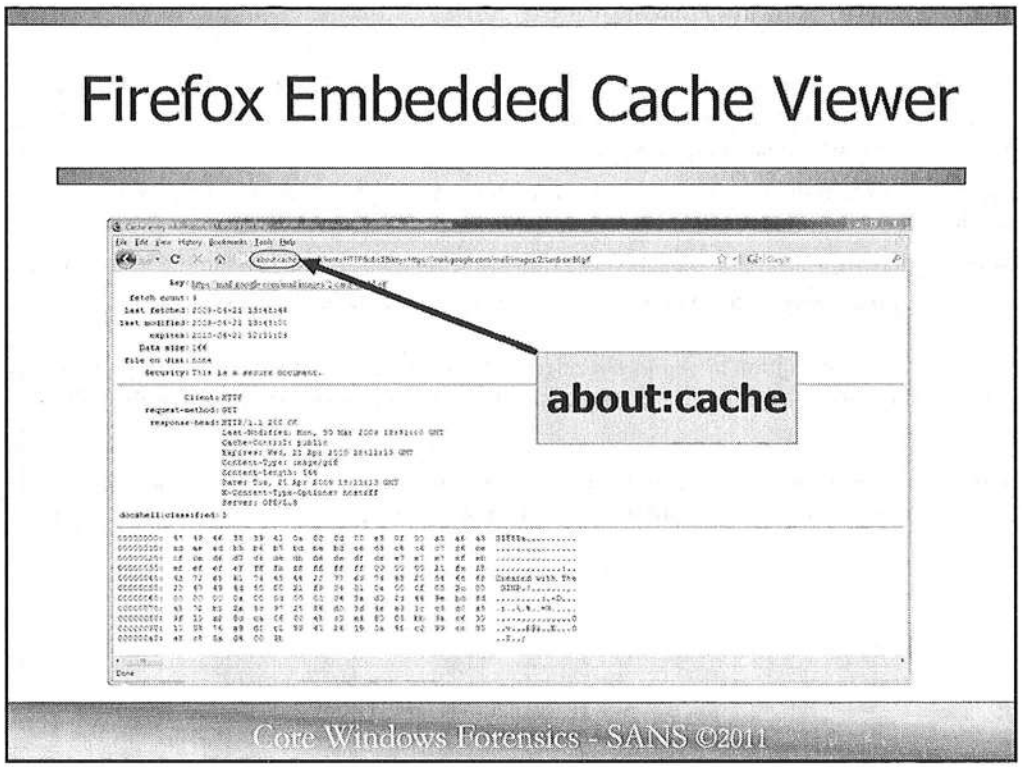
Server Response (not shown) – The HTTP response code issued by the webserver

MozillaCacheView has the option to export out selected files from the cache (as opposed to just saving out a report). The option is accessed from the File menu: **`File -> Copy Selected Cache Files to`** ...

One item to note is that when using this tool to review cache data on a live system, Firefox cannot be running. The cache files are locked by the application while it is running and thus MozillaCacheView cannot open the cache files.

References

[1] http://www.nirsoft.net/utils/mozilla_cache_viewer.html




While it is nearly impossible to review the Firefox cache directory and determine what is currently held in the cache, the developers at Firefox built a very useful function within the browser itself. If a user types **`about:cache`** into the URL bar, they are presented with a set of HTML pages allowing the cache to be perused. Information such as the number of cache entries, the maximum allocated cache size, the amount of the cache currently used and the cache directory are displayed to the user. Following this information, each cache entry is listed using the original URL of the request, and giving timestamp and fetch count information. Shown on this slide is the results of clicking on specific cache entry on this list. All of the relevant information for this cache entry is displayed, including the full response header from the webserver and a hex / ascii representation of the file. Finally, the file name can be clicked and the browser will display the page or file if possible.

This feature is extremely useful when reviewing cache entries on a live system, but it cannot be used to review exported files on a forensic workstation. It does show the level of granularity of data that a user can access about his/her cache files. Luckily for forensic investigators, one feature that is lacking is the ability to delete individual files from the cache (this function does not allow any modifications to the cache).

# Firefox Cookies: Going Deep into Website Activity

- Provide an additional means to profile Internet activity
- Firefox stores all of a user's cookies in a single file
  - Stored in the profile folder, linking them to a user account

| Investigative Questions | Firefox 2 cookies.txt | Firefox 3 cookies.sqlite |
|---|---|---|
| What website domain issued the cookie? | Column 1 | host |
| What is the cookie name? | Column 6 | name |
| Was the cookie issued in a secure connection? | Column 4 | isSecure |
| What values / preferences were stored? | Column 7 | value |
| When was the cookie created? | N/A | id |
| When was the cookie / site last accessed? | N/A | lastAccessed |



Each browser artifact provides slightly different information. There isn't just one artifact which can provide all of the information that an investigator might need. Cookies are a good example of this layered effect. While the history file might be the first stop of an investigator when looking for what sites have been accessed, cookies may provide additional sites and information which has been deleted or overwritten in the history file. Cookies are often saved longer than history information and are not as frequently deleted by security-conscious users.

Unlike IE, which stores cookies in individual files, Firefox collects all cookies into a single file. For Firefox 2, this file is called **`cookies.txt`** and is a simple text file consisting of a row for each cookie and tabbed columns for each of the cookie's values. An example of what a cookie looks like in the **`cookies.txt`** file is as follows [1]:

| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 |
|---|---|---|---|---|---|---|
| **.msn.com** | **TRUE** | **/** | **FALSE** | **1555802906** | **CULTURE** | **EN-US** |

Column 1: Name of the website that stored the cookie

Column 2: Can the cookie be read by other parts of the website (using the same domain name)?

Column 3: Webserver directory path that the cookie is relevant for

Column 4: Does the cookie require a secure (HTTPS) connection?

Column 5: Expiration time for cookie (in Unix epoch time – number of seconds since 1 Jan 1970)

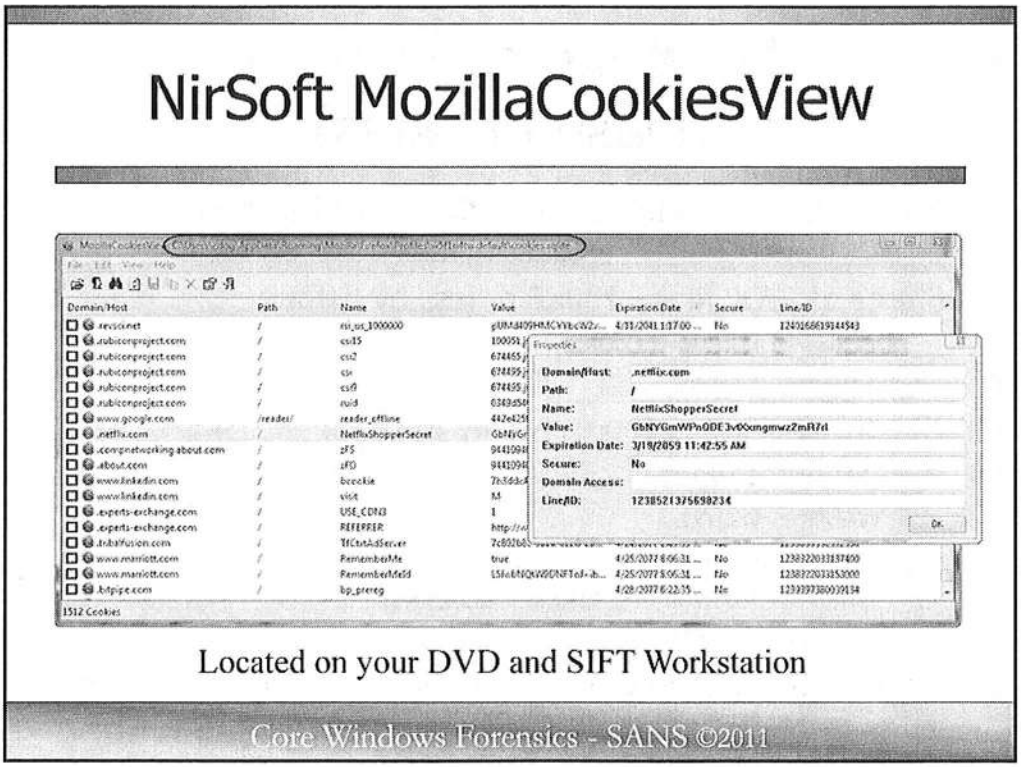
Column 6: How the cookie is referenced by the webserver

Column 7: Stored information within the cookie (user preferences, site state information, etc.)

A couple of critical pieces of information are absent from Firefox 2 cookies file. Notably, we do not have the created date nor the last accessed date that we have in IE cookies (from the MAC times of the individual files). This was rectified in the SQLite database used by Firefox 3.

**References**

[1] http://kb.mozillazine.org/Cookies.txt




MozillaCookiesView provides a simple interface for viewing cookies from both Firefox 1.5/2 and Firefox 3. [1] You simply point it to the file you are interested in parsing (using **`File -> Select Cookies File/Folder`**) and analyze the results. You can click on the column labels to sort by the value, which can be handy since sites often use more than one cookie. Right clicking on any entry will allow you to view the properties box, which displays data about that cookie in a different format.

The columns displayed by the tool correspond to the data fields stored in the cookie file:

Domain/Host - Name of the website that stored the cookie

Path - Webserver directory path that the cookie is relevant for

Name – How the cookie is referenced by the webserver

Value - Stored information within the cookie (user preferences, site state information, etc.)

Expiration Date -Expiration time for cookie (can be changed to GMT time using a menu option)

Secure - Does the cookie require a secure (HTTPS) connection?

Domain Accessible (Not shown) - Can the cookie be read by other parts of the website (using the same domain name)?

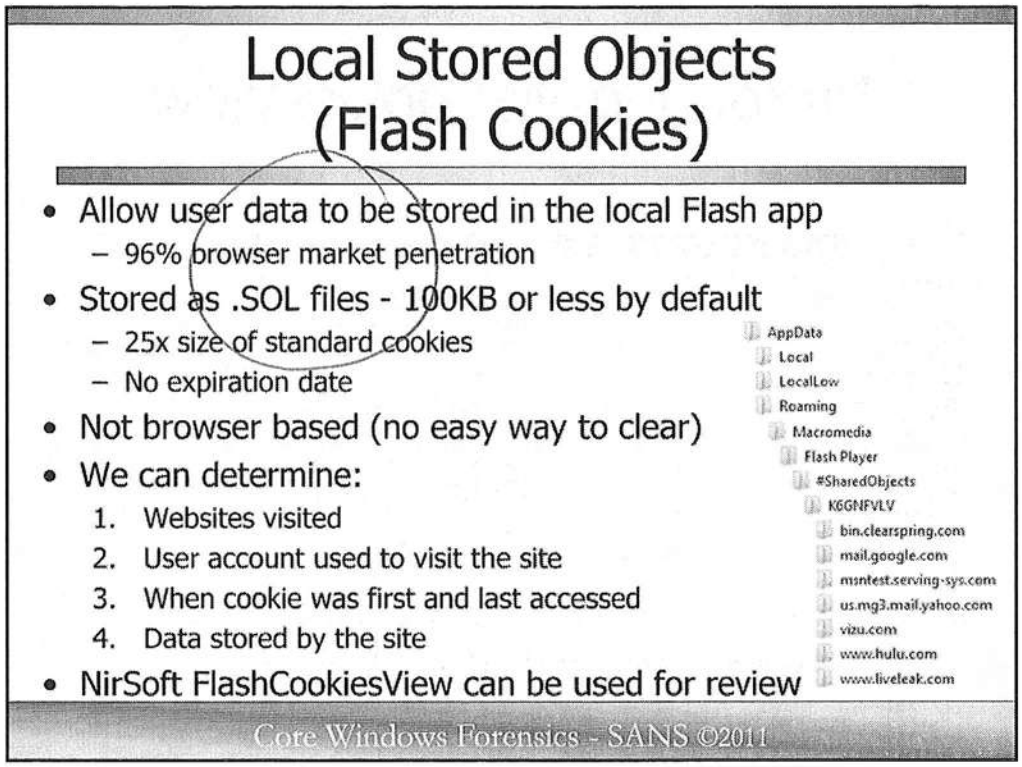
Line/ID – The location of the cookie within the cookie file (in Firefox 3 this value is also the creation time in PRTIME format)

Unfortunately one limitation of this tool is that it does not display the created and last accessed dates from the Firefox 3 SQLite database. This is likely because it was retrofitted from the Firefox 2 version which does not have these data elements. We will show you another tool later in this section which does provide this information (FoxAnalysis). [2]

**References**

[1] http://www.nirsoft.net/utils/mzcv.html

[2] http://forensic-software.co.uk/foxanalysis.aspx




Local Stored Objects (LSOs), or Flash Cookies, have become ubiquitous on most systems due to the extremely high penetration of Flash applications across the Internet. LSOs allow a web application to store information that can later be accessed by that same application (or domain). They are larger (up to 25 times the size of a 4KB cookie), and tend to be much more persistent since they do not expire and there is no built in mechanism within the browser to remove them (you must us the Adobe Flash Player Setting Manager). In fact, many sites have begun using LSOs for their tracking mechanisms since they rarely get cleared like traditional cookies [1].

Anything typically stored in a cookie (and more) can be stored in a LSO. Thus we should expect to find preferences, search terms, location information, visit frequency, etc. In addition to this, we can also use the existence of the cookie itself to give us the following information [2]:

- Websites visited
    - LSOs are stored hierarchical by domain. Thus in the example on the slide we can surmise that the user account has both Gmail and Yahoo! Mail accounts. Flash based advertisements also have the ability to save LSOs. This is important because in some cases we can't necessarily conclude that it was the user's intent to access the domain. The origin of the LSO is often obvious (i.e. ads1.msn.com), but testing or additional artifacts may be necessary to make definitive conclusions.
- User account used to visit the site
    - LSOs are stored under a user profile, indicating that that specific account was used to visit the site.
- When cookie was created and last accessed
    - Since cookies are stored as individual files, we can utilize the MACB timestamps of Windows to determine when that LSO was created and when it was last accessed. This at a minimum gives us a date range of known activity on that site.

LSOs use the .SOL file extension and are stored as individual files. They can be found in many locations, but are most commonly in the following folders (%APPDATA% is equivalent to the %userprofile%\Application Data folder):

Windows XP

%APPDATA%\Macromedia\Flash Player\

%APPDATA%\Macromedia\Flash Player\#SharedObjects\<random profile id>

%APPDATA%\Macromedia\Flash Player\macromedia.com\support\flashplayer\sys

Vista / Windows 7

%APPDATA%\Roaming\Macromedia\Flash Player\

%APPDATA%\Roaming\Macromedia\Flash Player\#SharedObjects\<random profile id>

%APPDATA%\Roaming\Macromedia\Flash Player\macromedia.com\support\flashplayer\sys

NirSoft has released a tool for facilitating the viewing of LSOs. It is called FlashCookiesView and similar to other NirSoft products, can be used on a live system, or pointed to an exported (or mounted) directory during a dead analysis [3].

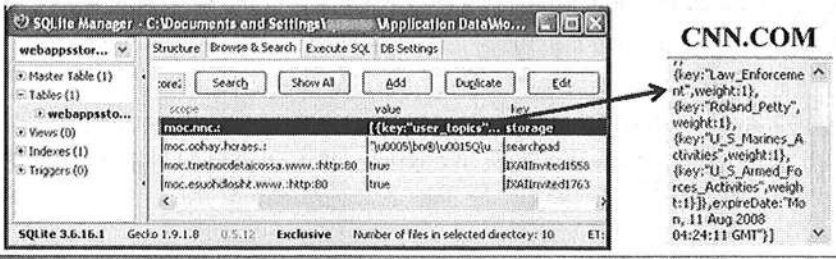
[1] http://www.schneier.com/blog/archives/2009/08/flash_cookies.html

[2] http://blogs.sans.org/computer-forensics/2009/08/28/flash-cookie-forensics/

[3] http://www.nirsoft.net/utils/flash_cookies_view.html

# DOM Storage

- Up to 10MB of local storage space per domain, per user
  - Preferences, keywords, visit tracking, usernames, offline files
  - No expiration and cleared along with cookies (in IE8 and FF)
  - IE8 includes timestamps for DOM Storage files
- Firefox – located in `webappstore2.sqlite` database
- IE8 – .XML files and index.dat in the `DOMStore` folder




Core Windows Forensics - SANS ©2011

Document Object Model (DOM) Storage is considered to be a "Super Cookie" in that it provides similar benefits to traditional cookies, but greatly expands their capabilities. It is part of the HTML5 specification and with the release of IE8, now most browsers support it (Safari and Firefox were early adopters) [1, 2]. As web applications have become more feature rich and client-side scripting interacts more with data on the local file system, a means to store large amounts of data that can persistent between individual sessions is needed. DOM Storage fills that need, but is not well known outside of the web development community.

From a forensic perspective, there is a great potential to recover useful artifacts. Each domain can use up to 10MB of disk space to store data on the local system. No specification or requirements exist for data formats, obfuscation, encryption, etc. Data within DOM Storage does not expire (though individual applications can choose to prune data).

In IE, DOM Storage is stored in .XML files under the following directories. While data can be read directly from the .XML files themselves, it is not always possible to ascertain context (i.e. what site they are from) for the files. DOMStore is set up in a similar fashion to the IE Cache. An index.dat file manages the DOMStore filenames and the owning sites. While not intended for that use, Mandiant Web Historian will parse the DomStorage index.dat file, giving the original website, the cached files, and the associated timestamp (which is likely modification time).

Windows XP

%userprofile%\Local Settings\Application Data\Local\Microsoft\Internet Explorer\DOMStore

Vista / Windows 7

%userprofile% \AppData\Local\Microsoft\Internet Explorer\DOMStore

- Firefox saves its DOM Storage files in a SQLite database named `webappstore2.sqlite`. Table columns include the scope (associated domain), the key name, the originating site, and the DOM Storage value.

This slide shows an example from DOM Storage in Firefox. The SQLite Manager tool was used to open the `webappstore2.sqlite` database. The storage key for CNN.Com (domains are stored in reverse order) contains data related to recent stories viewed as well as the number of visits to the site (not shown).

DOM Storage can be cleared by the user from within the browser. It is cleared in conjunction with clearing of traditional cookies. In Firefox, the user must select the time range of "Everything" in order to clear DOM Storage [3].

[1] http://en.wikipedia.org/wiki/DOM_storage
[2] http://msdn.microsoft.com/en-us/library/cc197062%28VS.85%29.aspx
[3] https://developer.mozilla.org/en/DOM/Storage

# Firefox Bookmarks: Looking at Saved Locations

- Bookmarks indicate sites of interest saved by the user
  - Found in **moz_bookmarks** table in **places.sqlite** (Firefox 3)
  - Firefox 2 uses text file, bookmarks.html
  - Bookmarks can be added without actually visiting the website
  - Identify bookmark usage in FF3 through History **type** value

| Investigative Questions | Firefox 2 bookmarks.html | Firefox 3 places.SQLite |
|---|---|---|
| What is the complete URL of the bookmark? | *in HTML <A> Tag | url |
| What is the title of the bookmarked page? | *in HTML <A> Tag | title |
| When was the bookmark added? | ADD_DATE | dateAdded |
| When was the bookmark last modified? | N/A | lastModified |

Core Windows Forensics - SANS ©2011

Bookmarks are of interest to an investigator because they give information about what sites interested a user and what sites they planned to return to. Bookmarks can be added or imported by a user without them visiting the site, but it is still more common for a site to be bookmarked while the browser is viewing the specific page. Unfortunately, there is no value which identifies what method was used in bookmarking a site. While bookmarks cannot definitively prove that a user visited a given website, they provide a very helpful data point when used in conjunction with other browser artifacts.

Unlike Internet Explorer, where Favorites were stored individually in files, Firefox collects bookmarks in a single file. In Firefox 2 and below, the bookmarks file is named **bookmarks.html** and stored in a human readable format (HTML). Some of the key information we will be looking for is stored in the HTML tags and is not displayed when viewing the HTML file in a browser. Thus we recommend using a forensic tool to view the file, or at a minimum reviewing the file using a text editor which will show the contents of those tags.

Firefox 3 deprecated the **bookmarks.html** file and now stores bookmarks within the **places.sqlite** file (the same file used to store Internet history information). [1] The bookmarks table within the places file is called **moz_bookmarks** and the specific fields are listed in this slide. Bookmarks in FF3 are no longer stored in a human-readable format (a tool is required to parse the SQLite database). On the positive side, the developers added a new field, **lastModified**, providing information about the last time the bookmark was changed by the user.

Interestingly, Firefox 3 does give us a means to identify sites that were visited using a bookmark. In the history table in **places.sqlite (moz_historyvisits)**, there is a **visit_type** field, which tells us how the page was accessed. This field is covered in depth in the Firefox history section of this course.

**References**

[1] https://developer.mozilla.org/en/The_Places_database

# Firefox Backup Bookmarks

- Firefox maintains backup copies of the bookmark file
  - Located in: `%userprofile%\Application Data\Mozilla\Firefox\Profiles\<random text>.default\bookmarkbackups`
  - Up to five daily backup copies will be stored in this directory
- Firefox 2 backups are named: `bookmarks-<date>.html`
  - Same format as `bookmarks.html` file
- Firefox 3 backups are named: `bookmarks-<date>.json`
  - Stored in Javascript Object Notation (JSON) format
  - Text based and refer to same values as in `places.sqlite`
- Backups are temporal – overwritten every 5 days



Firefox provides us with another unique browser artifact: the backup bookmarks directory. This folder is located under the user profile and contains up to five backup copies of the bookmark files. Backups are created daily and replaced in a first-in, first-out rotation, meaning the last five days of bookmark backup files should be present in this directory. This is admittedly a very short time window, and thus if bookmarks are a critical part of your investigation, it is important to take an image of the computer system as soon as possible.

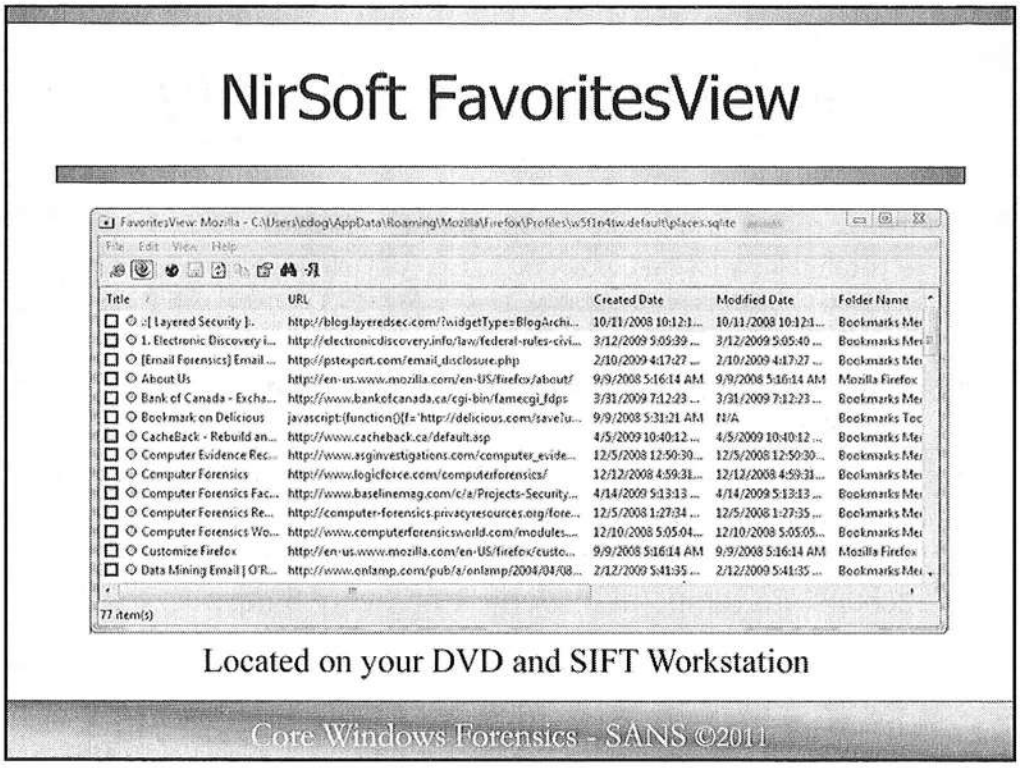
In Firefox 2 and below, backup bookmark files are named **`bookmarks-<date>.html`** and are stored in HTML format. [1] They can be reviewed in the same manner as the primary bookmark file (**`bookmarks.html`**).

Firefox 3 bookmarks are now stored in a SQLite database, and instead of creating multiple copies of that database, the developers chose to export the bookmark data into JSON format. JSON is Javascript Object Notation, a very simple text based format for data interchange. Backup files are named **`bookmarks-<date>.json`**. JSON is relatively easy to parse, and luckily the fields used are named the same as we will find in the **`places.sqlite`** database where the primary bookmark information is stored. An example bookmark in JSON format follows:


```
{"index":48,"title":"1. Electronic Discovery in the Federal Rules of Civil
Procedure | ELECTRONIC DISCOVERY - E-DISCOVERY
LAW GUIDES, BLOG & FORUM","id":102,"parent":2,"dateAdded":1236834339772600,
"lastModified":1236834340257600,
"annos":[{"name":"bookmarkProperties/description","flags":0,
"expires":4,"mimeType":null,"type":3,"value":"The amendments to the Federal
Rules of Civil Procedure that relate to electronic discovery of
electronically stored information went into effect n"}],"type":"text/x-moz-
place","uri":"http://electronicdiscovery.info/law/federal-rules-civil-
procedure/","charset":"UTF-8"}
```


**References**

[1] http://kb.mozillazine.org/Lost_bookmarks




NirSoft FavoritesView parses both **`bookmarks.html`** (Firefox 2) and **`places.sqlite`** (Firefox 3) and provides bookmark data in an easy to analyze format. Columns can be sorted to identify bookmarks by date, URL, etc.

The columns displayed by the tool correspond to the data fields stored in the relevant bookmarks file:

Title – Page title of the bookmarked page, or user-created label
URL – The specific URL pointed to by the bookmark
Created Date – When the bookmark file was created
Modified Date – When the bookmark was last changed
Folder Name – The name of the bookmarks folder which holds the bookmark (used to organize bookmark files)
Full Path (not shown) – Not applicable for Firefox; for IE it identifies what folder on the local drive holds the **`.url`** file

The screenshot shown in this slide immediately gives us some interesting information about what this user is interested in. We see several references to computer forensics and computer security. We also see a reference to Canada (are they planning a trip there?) and a bookmark to Delicious, which is an offline bookmarking service. Seeing the Delicious link indicates that a portion of the user's bookmarks may not be located on the local system.

FavoritesView will also parse IE Favorite files, displaying them in a similar format to Firefox bookmarks. Firefox 3 favorites on a live system cannot be viewed by the tool while Firefox is running (the **`places.sqlite`** file is locked).

**References**

[1] http://www.nirsoft.net/utils/faview.html

# Firefox Download History: Examining what was Downloaded

Firefox records a wealth of information about downloads

| Investigative Questions | Firefox 2 downloads.rdf | Firefox 3 downloads.sqlite |
|---|---|---|
| What was the file name? | `Name` | `name` |
| What was the file type? | `N/A` | `mimeType` |
| Where was the file downloaded from? | `URL` | `source` |
| What was the referring page? | `N/A` | `referrer` |
| Where was the file saved? | `File` | `target` |
| When did the download start? | `DateStarted` | `startTime` |
| When did the download end? | `DateEnded` | `endTime` |
| Was the download successful? | `DownloadState` | `state` |

- The default download directory is the user's desktop folder
  - If the user changes the default, it is set in **`prefs.js`** file

Core Windows Forensics - SANS ©2011

Firefox has a built-in download manager application which keeps a history of every file download by the user. This browser artifact can provide excellent information about what sites a user has been visiting and what kinds of files they have been downloading from them. It is not unheard of to find download histories of 600 files or more. Downloads occur frequently while browsing and many users do not pay attention to the history being kept or in some cases, where the files are actually being downloaded to. Imagine having an investigation into a system administrator suspected of running hacking tools on the corporate intranet. If a review of his Firefox download history indicates hacking tools were downloaded, you have taken a giant step towards solving your case.

The download history is kept in a text/XML file named **`downloads.rdf`** in Firefox 2 and within a SQLite database named **`downloads.sqlite`** in Firefox 3. [1] The download history tells us the name of the file downloaded, the URL it was downloaded from, where the file was saved, when the file was downloaded, and importantly, whether the download was successful. As we have seen with other browser artifacts, the shift in FF3 to SQLite databases has increased the amount of metadata kept for each browser artifact. Download history is no different. Firefox 3 also reports on the file type that was downloaded as well as what the referring site was that provided the link for the download (particularly useful if the file is located on a different domain than where its link is).

There have not been many tools written to parse Firefox download history. Luckily, the Firefox 1.5/2 download history file is in XML format, making it relatively easy to review within a text editor. For the FF3 SQLite database we will be using the Firefox SQLite Manager plug-in covered in a separate slide. Keep in mind that downloads.rdf keeps time information in local machine time, and downloads.sqlite maintains time information in PRTIME format.

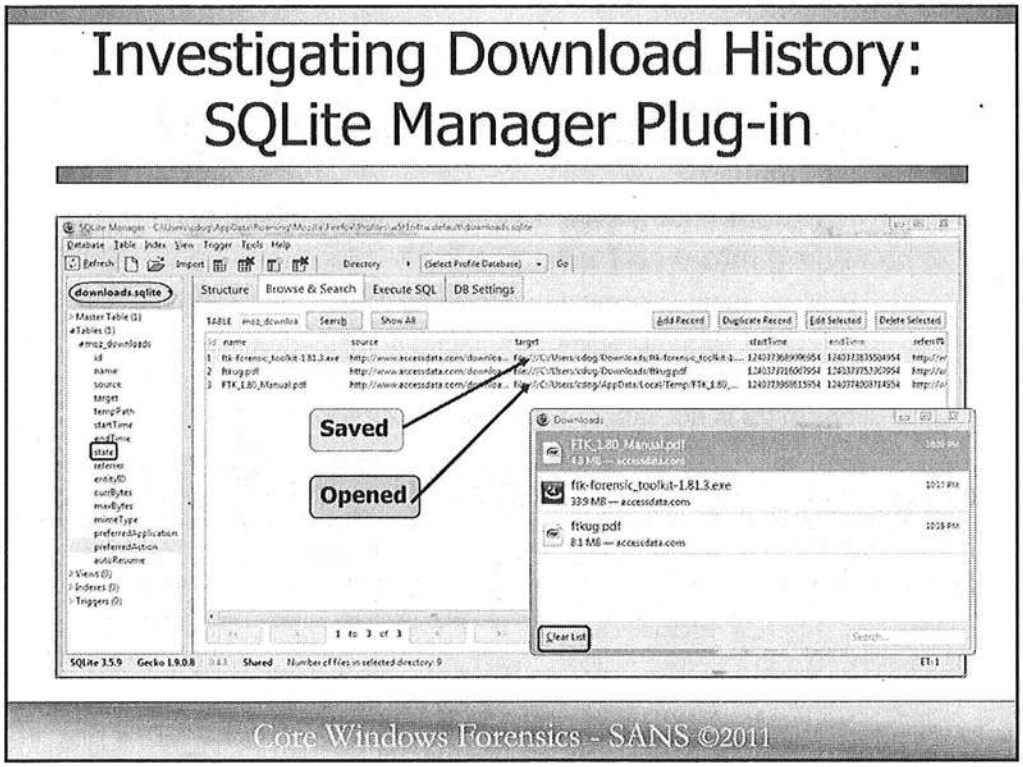
Users often download files and either forget that they downloaded them, or can't find where their default download directory is located. Identifying where the default download directory is located and reviewing its content can be helpful to an investigation. The default download directory in Firefox is the user's desktop folder: **`%userprofile%\Desktop\`**

This is often updated by the user, and is stored in the **prefs.js** file within the Firefox profile folder. Specifically, within this text file the field **browser.download.dir** holds the user specified directory (if it doesn't exist, then the default was not changed). The value will look like this:

```
user_pref("browser.download.dir", "C:\\Documents and
        Settings\\Administrator\\Downloads");
```

**References**

[1] http://kb.mozillazine.org/Downloads.rdf




Investigating the Firefox download history gives us a good opportunity to present a very useful tool in understanding the Firefox 3 SQLite database format. SQLite Manger is a plug-in that can be downloaded and installed within the Firefox browser. [1] Once installed, it is accessed via the Tools menu: **`Tools -> SQLite Manager`** opens up a new window similar to the one displayed on this slide. SQLite manager allows us to interact with any SQLite database, including the ones utilized by Firefox 3. We can see on the slide that we have opened the **`downloads.sqlite`** file within the application and have access to all the tables and fields within that database. This view is contrasted with what the user sees within the Firefox download manager, and we see that the data matches in both, with the **`downloads.sqlite`** providing much more information about the download history.

Looking at the target field on this slide, we see that files were saved in two different locations. A closer look (and possibly checking the prefs.js file) tells us that the **`%userprofile%\Downloads`** directory appears to be the default directory for saved files. The second directory identified in the download history is the **`%userprofile%\AppData\Local\Temp\`** folder, which is the temporary files directory within Microsoft Vista. This indicates that the file was opened by the user instead of saved. Reviewing data within temporary folders can be very useful because many users have no idea that a local copy of the file is saved when they open a file from the web. In this case, the download history allows us to tie a file within a temporary folder to a specific URL and a specific date and time.

One field in the **`moz_downloads`** table worth expounding on is the **`state`** field. This field tells us whether the download was successful or not. The following key translates **`state`** values into download results:

1=Successful download
2=Error occurred during download; aborted
3=Download cancelled
4=Download paused

**References**

[1] https://addons.mozilla.org/en-US/firefox/addon/5817

# Firefox Auto-complete: What Was the User Typing?

- Firefox maintains a vast amount of data which a user has typed into web forms
  - Helps users fill in common web forms without re-typing the data
  - Perfect for usage profiling: providing items and people being searched for, usernames and aliases, e-mail addresses, etc.
  - Form data is not tied to a specific website (only a form name)

| Investigative Questions | Firefox 2 formhistory.dat | Firefox 3 formhistory.SQLite |
|---|---|---|
| What type of form was the data entered into? | `Name` | `fieldname` |
| What was the data typed by the user? | `Value` | `value` |
| When was the data first typed in? | `N/A` | `firstUsed` |
| When is the last time the data was used? | `N/A` | `lastUsed` |

Core Windows Forensics - SANS ©2011

As browsers continue to add functionality to make user's experiences better, they inevitably leave even more browser artifacts for the forensic investigator to find. Auto-complete data has been around for some time now in both Internet Explorer and Firefox, and Firefox 3 adds to the amount of metadata we can expect to find in auto-complete data. In general form history, or auto-complete information, gives the investigator an exciting chance to see a catalog of items that a user has entered into web forms while surfing the Internet. These forms could be usernames, addresses, search terms, contact e-mails, dates, airports, frequent flyer numbers, etc. An investigator should expect to find hundreds of pieces of form data when investigating a Firefox browser that has not regularly deleted the form data. Auto-complete information is helpful to the user too – they do not have to re-type commonly used information on a variety of different websites. Thus, there is an incentive for users to not regularly clear this data.

An important thing to understand about auto-complete data is that it is not tied to a specific website. Thus we do not have the means to prove that a user entered a specific piece of data on a specific site. Instead, the data is tied to the form name (i.e. first name, last name, address, e-mail address). This way it works on any site which names their forms in a similar fashion.

Firefox version 1.5 and 2 use a file named **`formhistory.dat`** to store auto-complete data. This file is in Mork format and is extremely difficult to parse by hand. Instead, we will rely upon a tool called DumpAutoComplete (covered in a separate slide). The key information an investigator will find within this file is simply the name of the form and the data that has been stored for it.

Firefox 3 transitioned the form history to an SQLite database, named **`formhistory.sqlite`**. In doing so, two additional fields were added which could be of interest to an investigator: the date that data was first entered and when it was last used. Remember that time within the FF3 SQLite databases is stored as PRTIME and must be converted accordingly.

# Foundstone DumpAutoComplete

- Command-line only tool
  - Outputs in XML format
- Works with all Firefox versions
- Does not provide time info from SQLite files
- Only extracts data from **formhistory** files
  - **places.sqlite** file also stores auto-complete data

Located on your DVD and SIFT Workstation

Core Windows Forensics - SANS ©2011

While we try to stay away from command-line tools within this course, sometimes the best tool for the job happens to not have a pretty user interface. DumpAutoComplete has been around for a few years now and is the standard for dumping the form history data in Firefox. It works for all versions of Firefox (1.5-3.0), extracting data from both **formhistory.dat** and **formhistory.sqlite**. The output of the tool is in XML format, which can be easily imported into Microsoft Excel for analysis and filtering purposes.

The tool usage is as follows:

Step 1 Open a command shell: **Start -> Run -> cmd.exe**
Step 2 Change directory to find exported browser files: **cd <export directory>**
Step 3 Run tool: **dumpAutoComplete [formhistory.dat | formhistory.sqlite] > output.txt**

The example usage above redirects the tool output from the screen into a file which can then be imported into your viewer of choice.

This slide shows a **formhistory.dat** file parsed from a Firefox 2 installation. For Firefox 3 SQLite files, DumpAutoComplete will not display the **firstUsed** and **lastUsed** time fields. If this information is important, the SQLite Manager plugin or the FoxAnalysis tool can be used (both covered on separate slides).

A final note is that while the formhistory files in Firefox contain the vast majority of auto-complete data, there may be some relevant auto-complete information located the **places.sqlite** file (Firefox 3). Specifically, the **moz_inputhistory** table within **places.sqlite** maintains auto-complete data for the URL location bar, assisting users with frequently typed URLs. A comprehensive review of auto-complete data within Firefox 3 would require looking in both locations.

**References**

[1] http://www.foundstone.com/us/resources/proddesc/DumpAutoComplete.htm

# Firefox 3 FoxAnalysis Tool: Form History Data

- Only parses FF3 files
  - Internet History
  - Bookmarks
  - Cookies
  - Downloads
  - Form History
- Does not parse cache files
- Includes robust filtering
- Exports to XML, CSV, HTML

Located on your DVD and SIFT Workstation

Core Windows Forensics - SANS ©2011

FoxAnalysis is a relatively new tool that was created to parse the SQLite databases used in Firefox 3 and display them in a very user-friendly format. [1] It has an easy to use executable installer and requires the Microsoft .Net 3.5 (or greater) library to be installed on your forensic system. FoxAnalysis extracts and presents data from:

- Internet History (**`places.sqlite`**)
- Bookmarks (**`places.sqlite`**)
- Cookies (**`cookies.sqlite`**)
- Downloads (**`downloads.sqlite`**)
- Form History (**`formhistory.sqlite`**)

Cache files cannot be extracted using FoxAnalysis.

FoxAnalysis has several useful features:

- The ability to convert timestamps to any time zone (all FF3 timestamps are stored in UTC time)
- Robust filtering by multiple keywords, date, web visit type, and download status
- Exporting of results to XML, CSV, and HTML formats

On this slide we are showing an example of the form history being parsed by FoxAnalysis. We can see the form names and the data saved for each form type. Notice that there are multiple entries for the name "searchbarhistory". The form history will store every unique value that has been entered into a form by that name to provide as many auto-complete options as possible to the user. We see a physical

and e-mail address that was entered as well as a "messageSubject". The latter is likely from a web-based e-mail account form. The output here is a marked contrast to the command line output offered by DumpAutoComplete. Unfortunately FoxAnalysis currently only works for Firefox version 3.0 (and above). As Firefox 2 continues to get phased out around the world, this will be less and less of a problem.

**References**

[1] http://forensic-software.co.uk/foxanalysis.aspx

# Firefox Session Restore

- Records the following information for open windows / tabs:
  - Tab history
  - Cookies
  - Form Data
  - Failed file download information

| File List | | |
|---|---|---|
| Name | Size | Date Modified |
| sessionstore.js | 7 KB | 3/19/2010 5:12:33 AM |
| sessionstore.js | 3 KB | 3/18/2010 5:07:27 AM |
| sessionstore.js | 1 KB | 3/18/2010 4:54:16 AM |

- Saved in **`sessionstore.js`** file
  - Javascript (text) format -- much easier to analyze than IE8 artifacts
  - File deleted when browser is closed
  - Possible to find multiple deleted **`sessionstore.js`** files
  - User can opt to save tab information once or always

Core Windows Forensics - SANS ©2011

Long before Internet Explorer had Session Recovery, Firefox 2 implemented Session Restore [1]. The two features have the same goal: to restore a user's browsing session to a previous state. With the advent of tabbed browsing, the amount of data necessary to do this becomes much greater, ultimately benefiting us as forensic investigators.

Firefox uses the **`sessionstore.js`** file to automatically restore a session after an automatic update, or when using the restart button from the Add-ons Manager [2]. When the browser crashes, or a user shuts down the browser with multiple tabs open, Firefox gives the option to restore the session. Session Restore is also used to allow a user to undo a tab closure (**`History`** → **`Recently Closed Tabs`**).

Sessionstore.js records the following information [2]:

- All open windows and tabs
- Width, height, and position of each window
- Scroll position within each scrollable area in each window
- Tab histories
- Cookies
- Text typed in forms
- Failed downloads available to restart

The **sessionstore.js** file is stored in javascript format within the Firefox **profile** folder, and is easily string searchable. Mozilla developers have discussed encrypting the contents, but it has not yet been accomplished. For privacy reasons, the file is deleted upon a normal exit from the Firefox application [2]. Since the file is not wiped, it is possible to recover multiple historical copies, as shown by the graphic in this slide. There are two primary instances when the file will not be deleted: in the event of a crash and when the user has opted to save the tab information. Tab information can be saved as a result of the user opting in upon browser close, or it can be done persistently via **Tools→Options→General→When Firefox starts: Show my windows and tabs from last time**. In either case, the same **sessionstore.js** file will be recycled for the next session.

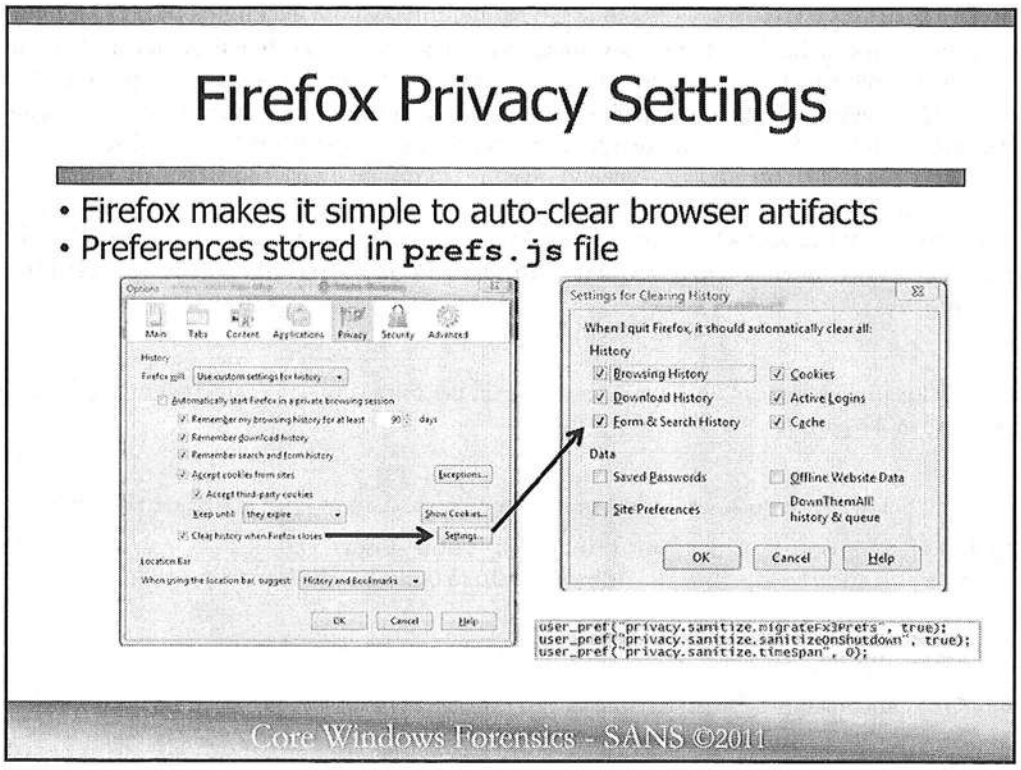
The Session Restore option can be disabled by a user in the following ways, and investigators can check these settings in the **prefs.js** file:

Firefox 3 and below: Set **browser.sessionstore.enabled** to false in **about:config**.

Firefox 3.5+: Set **browser.sessionstore.max_tabs_undo** and **browser.sessionstore.max_windows_undo** to 0 in **about:config**.

[1] http://kb.mozillazine.org/Session_Restore

[2] https://wiki.mozilla.org/Session_Restore



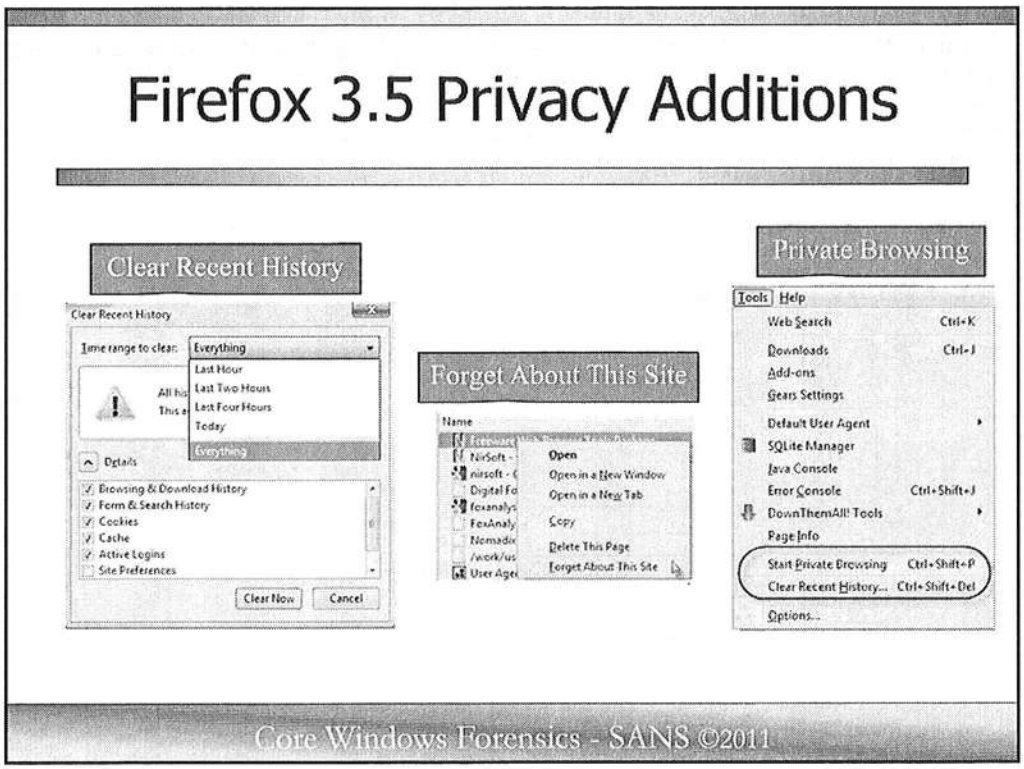

The **Privacy** tab located within the **Tools -> Options** menu item in Firefox provides several options for users to customize how their data is stored within Firefox. We see in the slide that history data is currently set for the default, 90 days. Setting this to zero or unchecking the box would effectively prevent history information from being recorded. The same goes for auto-complete information and download history located in that same section.

Within the Private Data section in the Privacy tab is the option to clear private data whenever the browser is closed. It goes without saying that this option can seriously impact the amount of data located within browser artifact files. When this option is selected, users can select exactly what will be cleared and what will remain. The options and what Firefox 2 and Firefox 3 files they affect are as follows:

Browsing History: Clears the history information and location bar auto-complete information (history.dat / places.sqlite)
Download History: Clears the history data in the download manager (downloads.rdf / downloads.sqlite)
Saved Form and Search History: Clears form data autocomplete and the search bar autocomplete (formhistory.dat / formhistory.sqlite)
Cache: Deletes all cache files in the cache directory
Cookies: Removes all saved cookies (cookies.txt / cookies.sqlite)
Offline Website Data: Deletes all cache files within the offline cache directory
Saved Passwords: Clears all data saved in the Password Manager application (signons2.txt / signons3.txt)
Authenticated Sessions: Deletes all session cookies from memory

Any selections made within this tab will be stored in the user's **prefs.js** file within their profile folder. [1]

[1] http://kb.mozillazine.org/Category:Security_and_privacy-related_preferences




Firefox can provide the investigator with a large amount of data from many helpful browser artifacts. The flip side of the coin is that Firefox also has a robust set of preferences available to users to delete much of the data located in these artifacts. While these options are not set as default, they are exceedingly easy for a knowledgeable user to implement. Firefox 3.5 improved upon existing privacy features by providing finer grain control over their settings and increasing the number of proactive and reactive privacy options [1].

The selective ability of the new privacy controls in FF3.5+ makes our job as forensic analysts a bit tougher. In previous versions of FF, artifact clearing was often all or nothing. If we were confident that FF was being used (say it was the default browser) and there were little or no historical artifacts, we could at least start looking for indications that evidence had been deleted (and possibly be well on our way to a data spoliation claim). The problem with selective removal is that it is much, much harder to determine if potential evidence has been removed.

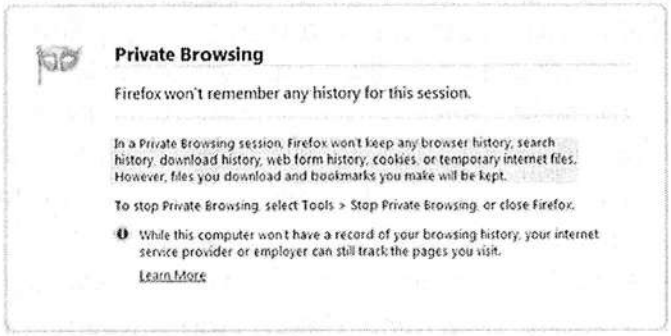
**Clear Recent History**: Allow users to not only choose specifically what they wish to clear (i.e. clear History, but leave cookies), it now gives the options of several timeframes. Notably, "Last Hour", "Last Two Hours", "Last Four Hours", "Today", and "Everything". We mentioned in the DOM Storage section that in order for those "Super Cookies" to be cleared, a user would need to select both "Cookies" and "Everything".

**Forget About This Site**: When viewing the History Library (History→Show All History) there is now a right-click option to "forget" or remove the site from the Places.sqlite database. In previous versions this could be accomplished by using the (<ctrl><shift><H>) hotkey, and individual history entries could be removed with the <delete> key (in Firefox 1.5 users are required to use <shift><delete>). A similar method can be used to selectively delete within the download manager, the bookmark dialog box, etc. The key difference is the visibility of the new option to the average user.

**Private Browsing:** This option is quite similar to IE8 InPrivate Browsing, in that it attempts to create an artifact-free browsing session for the user. Thus a user could turn on private browsing to visit one particular site, and then turn it off to return back to their original Firefox session.

[1] http://blog.mozilla.com/faaborg/2009/06/30/firefox-35-and-privacy/

# Private Browsing Mode




- The Firefox implementation of Private Browsing is robust
- Artifacts from these sessions are difficult to recover
  - Unallocated or memory analysis is often required

Core Windows Forensics - SANS ©2011

It is hard to believe that IE and Chrome both beat Firefox to implementing a private browsing feature, but it is true. There have been FF extensions for a long while to do so, but Firefox 3.5 was the first release to have the capability built in.

Private Browsing Mode in Firefox protects:

- History
- Search History
- Download History
- Form Data
- Cookies
- Cache Data

Bookmarks and files downloaded (both opened or saved) are not protected and are recoverable.

The implementation of this feature is strong, not in small part due to the file formats that Firefox employs. Since items can be easily removed from SQLite databases, we don't see any of the index.dat data leakage problems that Internet Explorer has had over the years. The same goes for the FF cache – it uses an obtuse database structure as opposed to individually save files in IE. Thus when files are deleted from the cache, they are much harder to recover. In testing, the only reliable way to recover artifacts from a Private Browsing session was to carve and search unallocated space and memory images.

## Recovering Deleted Artifacts in Firefox

1. What if a user clears all data using privacy settings?
   - Browser artifact files are overwritten with new files
   - SQLite is notorious for spraying remnants throughout free space
   - Cache Map and Block files are overwritten making recovery hard
2. What if the browser artifact files are manually deleted?
   - Also occurs when Firefox has been uninstalled
   - Recover deleted files using forensic tools and perform analysis
3. What if a user selectively deletes individual records?
   - Using Private Browsing, Clean Recent History, etc.
   - Finding deleted records within Firefox database files is rare
   - SQLite database files perform frequent cleansing operations
   - Remnants will likely exist in unallocated space and memory

Core Windows Forensics - SANS ©2011

With the user-friendly data clearing features present in Firefox, it is inevitable that we will run into browser artifacts which have been cleared prior to our investigation. The manner of how the artifacts were cleared determines how difficult it will be for an investigator to recover them.

The most common way users clear Firefox artifacts is by using the privacy settings in the **Tools -> Options -> Privacy** menu. Users now have fine grained control over what is deleted, so finding one cleared artifact does not mean that every file has been cleared. When a user clears an artifact using this method, Firefox removes all user data within the file, essentially overwriting the old file with a much smaller new file. This process makes full forensic recovery of the original file very difficult. On the positive side, the vast majority of the data from the file will now be located as fragments in unallocated space, where it can be discovered by keyword searches. However, analysis of these file fragments will be much harder since we will not have the luxury of being able to load them into our tools to automatically parse them.

By design, SQLite databases require a large number of temporary files. This is due to implementing an entire database within a single file. To perform database functions, temp files are created very frequently. This is such a problem that SQLite databases can often not be used for applications where system resources discourage creating temp files [1]. From a forensic perspective this is great news. It means that even though we may have issues recovering the original artifacts, we have an excellent chance of finding (multiple) copies of those same artifacts in unallocated space. Thus we should expect to find more Firefox artifact fragments in free space than in non-SQLite based browsers like Internet Explorer.

When the user clears the cache, the Cache Map and Cache Block files are overwritten in a similar way. Any Cache Data files are simply deleted, making full file recovery possible. However, the binary format of the cache and the loss of the Cache Map file makes recovering cache information very difficult in this circumstance.

If you recall, another method of clearing Firefox artifacts is to simply delete the key files and allow Firefox to automatically regenerate them. When this occurs, forensic recovery can be much easier, because the entire file can be recovered (as opposed to file fragments as in the previous case). Unfortunately this is the least common of the deletion options.

The final method that a user can use to clear artifacts is to utilize the new privacy features in Firefox 3.5+ to selectively delete records from within the browser. Unlike in Internet Explorer where the index.dat files can contain old records, finding deleted records in Firefox database files occurs very rarely. Firefox performs frequent cleansing operations on its database files which eliminate old and deleted records. While it is uncommon to find records deleted this way within the original file, it is still possible to recover any deleted data by searching unallocated space.

[1] http://www.sqlite.org/tempfiles.html

# Firefox Extensions

- Extensions are applications that can be downloaded by a user to extend the functionality of Firefox
  - Can give information about how browser was used or where to look for additional browser artifacts
- Each application will be listed in `extensions.rdf`
  - Located in the Firefox profile directory; user-specific
  - File is in XML format, making it human-readable
- The `extensions.rdf` file provides the following information helpful to an investigator:

| Extension Name | Installed Version |
|---|---|
| Description | Homepage |

Core Windows Forensics - SANS ©2011

Extensions, or plug-ins, are one of the key items which has fueled the adoption of Firefox. There are thousands of third-party applications that can be loaded into Firefox to build upon its functionality. We have already covered one such plug-in, SQLite Manager. Knowing what extensions have been loaded by a user may be of use to an investigator. One example would be if an application like "Stealther" is present, which allows users to surf the Internet without adding data to the browser artifact files. An application like "Cookie Watcher" or "Firebug" could be an indication that a user is interested in web application hacking (or maybe just a web developer). Cataloging a user's extensions is an additional way that an investigator can profile the usage of the person they are investigating.

The `extensions.rdf` file is a global list of all extensions installed for that Firefox profile. [1] Additionally, each installed extension is required to have a file named `install.rdf`, typically located in the `extensions` directory in the user's Firefox profile. [2] The information contained in both files is identical and is kept in XML format, making the files easy to review within a text editor. When an extension is removed (uninstalled) the `install.rdf` file is deleted and the entries within the `extensions.rdf` are removed.

The `extensions.rdf` file can provide several pieces of information which can make analysis of installed plug-ins easier:

Name – The registered name for the extension (can be used to perform Internet searches to learn more)

Description – A short summary of what the extension is supposed to do

Version – The installed version number of the extension

HomepageURL – The home page for the extension (if it exists)

**References**

[1] http://kb.mozillazine.org/Uninstalling_extensions

[2] http://kb.mozillazine.org/Install.rdf

## Firefox Forensic Methodology

1. Identify Firefox version and profile locations
   a) Recall that both local and roaming folders are used in Vista
2. Export relevant files to forensic workstation
   a) It is often easiest to export both `Profiles` folders
   b) Identify how many Firefox profiles are present for the user
   c) Check FF privacy preferences if files are small or missing
3. Review exported browser files, starting with history
   a) Use specialized parsing tools to identify key browser artifacts
   b) Each artifact can answer questions in a slightly different way
4. Record / export relevant information and files

Core Windows Forensics - SANS ©2011

There are a lot of moving pieces that make up the Firefox browser, but our analysis strategy is straightforward. Our first step is to confirm that Firefox is installed on the system we are reviewing and determine where in the directory structure its key artifact files are located. Once found, a quick review of the file extensions (.dat versus .sqlite) will tell the investigator what major version of the browser they will be analyzing.

While many forensic tools can perform analysis using a disk image file, most of the tools available for Firefox analysis require that the artifact files be exported out of the image. Luckily, since Firefox keeps all of its files in one place, exporting files to our forensic workstation is simple. To ensure that you get all files which you may need to look at, simply export all relevant `Profiles` folders. Recall that Firefox has **Profiles** folders in two locations for each user: one in the local and one in the roaming folder. Make sure to get the **Profiles** folders in the local directories as well, since that is where cache information will be stored. Doing this will ensure that you get all relevant subdirectories like `extensions`, `backupbookmarks`, `cache`, and `offlinecache`. If a user is utilizing multiple Firefox profiles, these will also all be located under the `Profiles` folder. Once the files are exported, take a quick look to ensure you have all files that you expect. If files are missing or appear small in size, the user could be deleting browser data using privacy settings, manually deleting files, or simply not using Firefox.

When conducting browser artifact analysis, a good rule of thumb is to start with the browser history file. This should give the investigator a good overview of Internet activity on the system. Each additional artifact reviewed should build upon this foundation and continue to flesh out the user activity profile. Remember that some artifacts may keep data longer than others, and some may be regularly cleared by the user. The breadth of information provided by the available artifacts frequently gives more than one way to prove Internet activity.

Finally, keep good notes and save or export any relevant findings from the analysis. On frequently used systems, the amount of data found within browser artifacts can be enormous. It is important to take detailed notes and export data as you go, because you will likely be using an iterative process and returning to browser data files as you discover key findings. Most forensic tools, including the NirSoft tools introduced here, have a means to select relevant data and save it to an external file.

# Firefox Hands-on (1)

- Using FoxAnalysis
  - Close All Firefox Windows on your XP SIFT Workstation
  - Create a New Case
  - Select the Firefox 3 Directory
    - `%userprofile%\Application Data\Mozilla\Firefox\Profiles\<random text>.default`
    - Select Check Files -> All files should be "OK"
  - Insert Case Properties
    - Case Number, Your Name, Profile Name, and Notes
  - Select Time Zone Settings
    - Select current time zone
    - Configure DST
      - Default to US -> Select OK

Core Windows Forensics - SANS ©2011

This page intentionally left blank.

# Firefox Review

| Browser Artifacts | Firefox 1.5 / 2.0 | Firefox 3.0 |
|---|---|---|
| **Internet History File** | history.dat | places.sqlite |
| **Cache Files** | _CACHE_<XXX> | _CACHE_<XXX> |
| **Cookies** | cookies.txt | cookies.sqlite |
| **Bookmarks** | bookmarks.html | places.sqlite |
| **Download History** | downloads.rdf | downloads.sqlite |
| **Auto-Complete / Form History** | formhistory.dat | formhistory.sqlite<br>places.sqlite |
| **Installed Extensions** | extensions.rdf | extensions.rdf |



From the outside, Firefox appears to be much more complicated to conduct a forensic analysis on than Internet Explorer. It's file formats are difficult to parse and it does not have the nicely formatted **History** and **Temporary Internet Files** folders to display artifact data on a live system. Additionally, forensic suites like FTK have not taken the time to introduce built-in parsing features like we see for Internet Explorer.

One advantage that Firefox does have is that its browser files are generally all in one place, allowing us to just export the **Profiles** folders to our forensic workstation. Once have the files on our workstation, our analysis techniques between IE and Firefox are very similar. We have a nice suite of freeware tools which make easy work of parsing, filtering, reviewing, and exporting Firefox artifacts. The introduction of the SQLite database format in Firefox 3.0 has only added to the amount of metadata available to investigators. Firefox in general maintains more user data than Internet Explorer, which can be of great benefit to us as forensic investigators. The only catch is you have to know where to find it. This slide shows a table of the various browser artifacts covered in this section with their corresponding files in the various Firefox versions.

# Firefox Hands-on (2)

- Using FoxAnalysis
  - Close All Firefox Windows on your XP SIFT Workstation
  - Create a New Case
  - Select the Firefox 3 Directory
    - `%userprofile%\Application Data\Mozilla\Firefox\Profiles\<random text>.default`
    - Select Check Files -> All files should be "OK"
  - Insert Case Properties
    - Case Number, Your Name, Profile Name, and Notes
  - Select Time Zone Settings
    - Select current time zone
    - Configure DST
      - Default to US -> Select OK



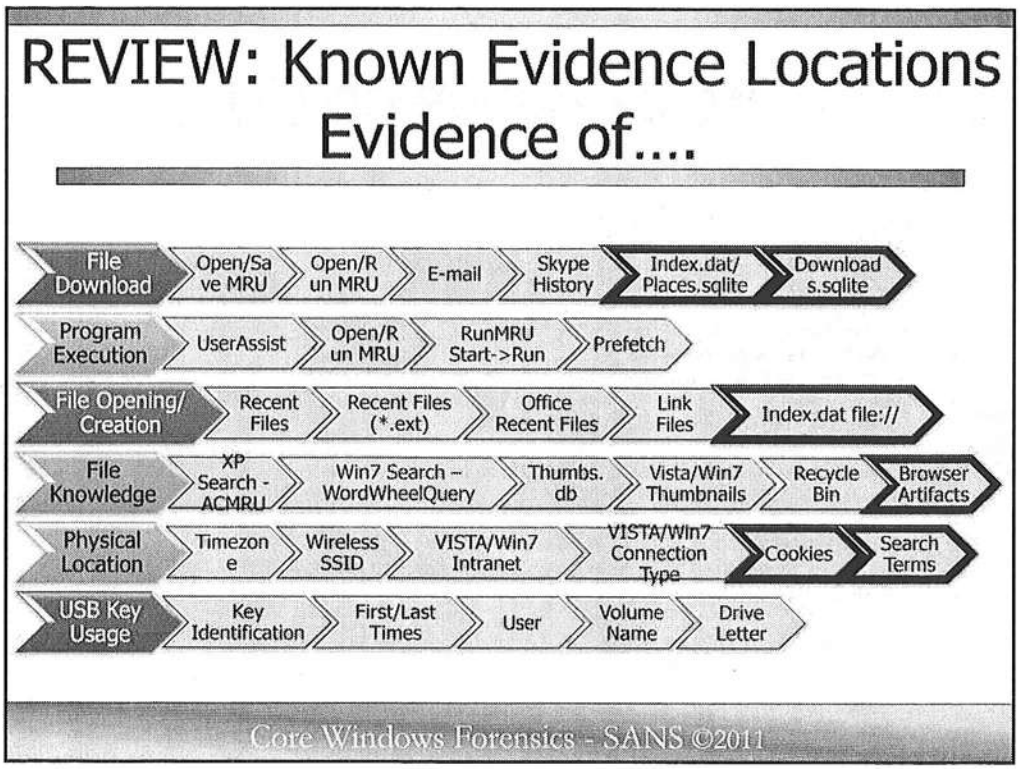
This page intentionally left blank.

# Firefox Hands-on (3)

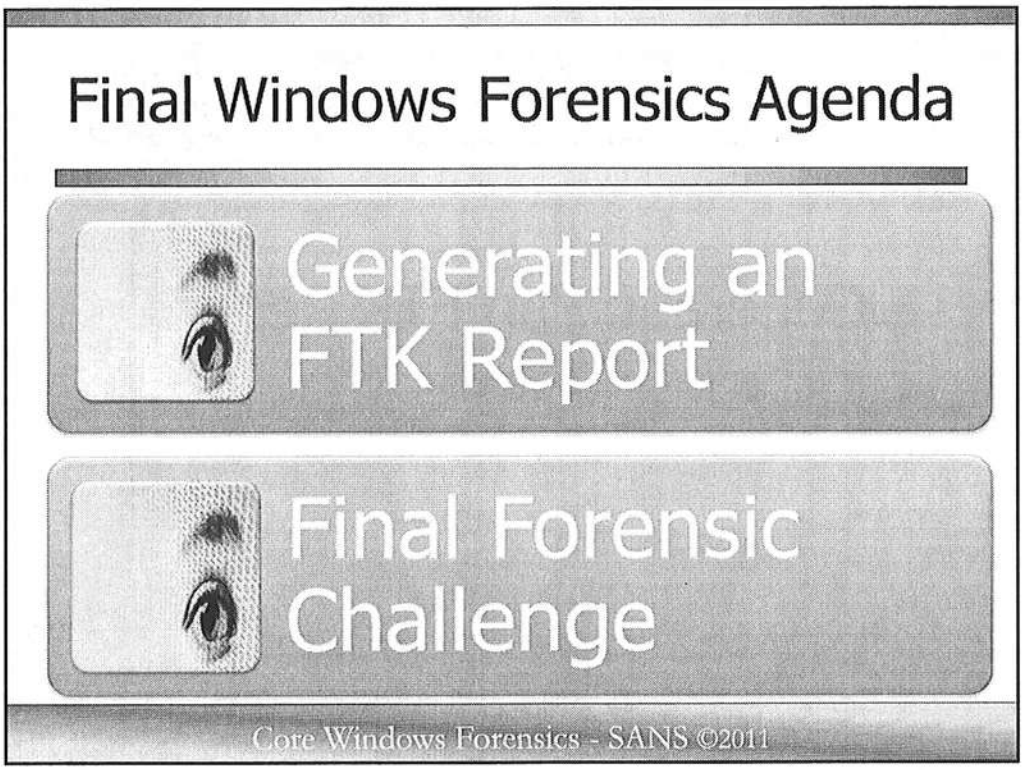
- View
  - History
  - Bookmarks
  - Cookies
  - Downloads
  - Form History
- List the last three sites you visited
  - What is your favorite website according to the FoxAnalysis tool? Why?
- Find a bookmark
  - When was the bookmark added?
- Examine cookies
  - Find the first cookie. When was it created? When was it last accessed?
- Have you downloaded any files?
  - Which ones?
- Any Form History saved by Firefox? What does it tell you?
- Select File -> Export To HTML -> Select Everything -> Save to Desktop as Firefox.html
- Open Desktop/Firefox.html and examine contents



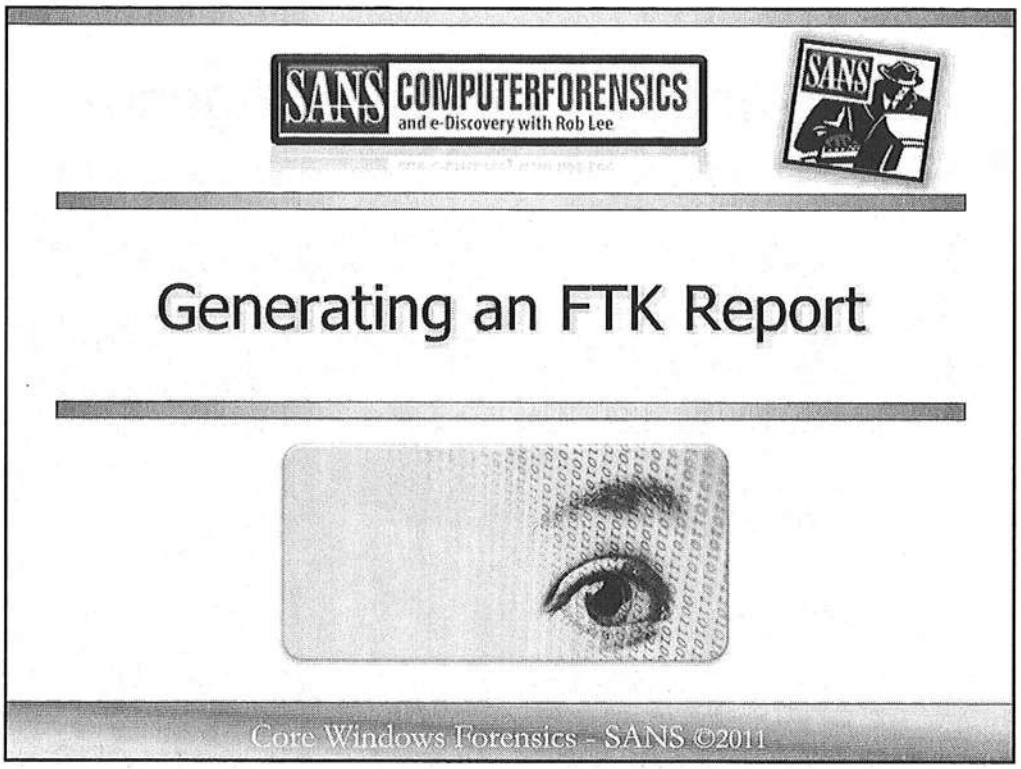
This page intentionally left blank.



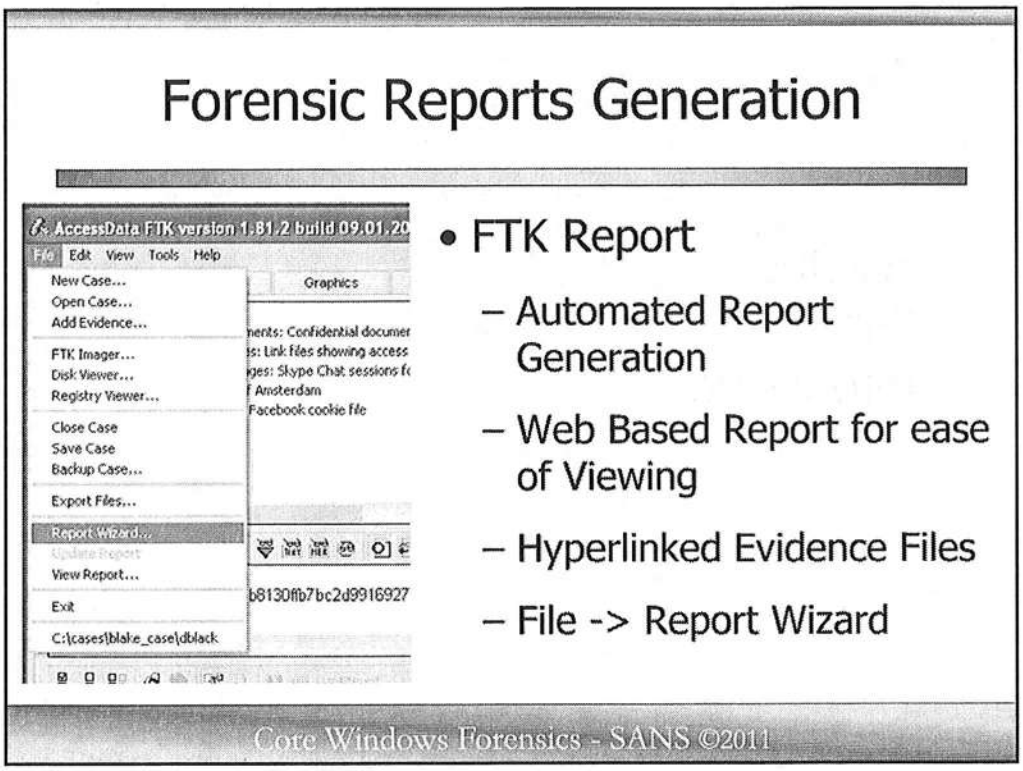

This page intentionally left blank.




This page intentionally left blank.




This page intentionally left blank.




One of the great features of FTK is that it has the capability to generate very nice web based reports that are easy for the customer to review from any computer on any operating system. These reports, since they are based in HTML can be completely edited by the publishing examiner to fit your agencies style, color scheme, etc.

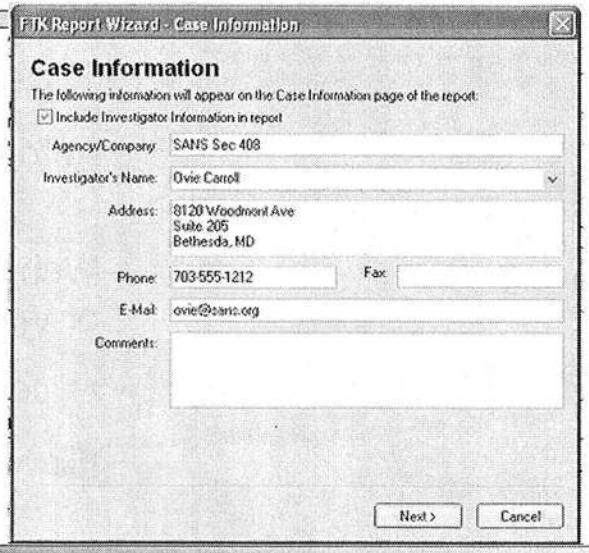
Another really nice feature is that all the evidence will be hyperlinked within the report. This way the report can be cleaner, and if they want to see the actual evidence they can click on the hyperlink.

The most important thing you should remember is that FTK report is generated based on the bookmarks you select and if you choose to export those items.

To start the FTK Report Generation Wizard, select "**Report Wizard**" from the "**File**" menu.

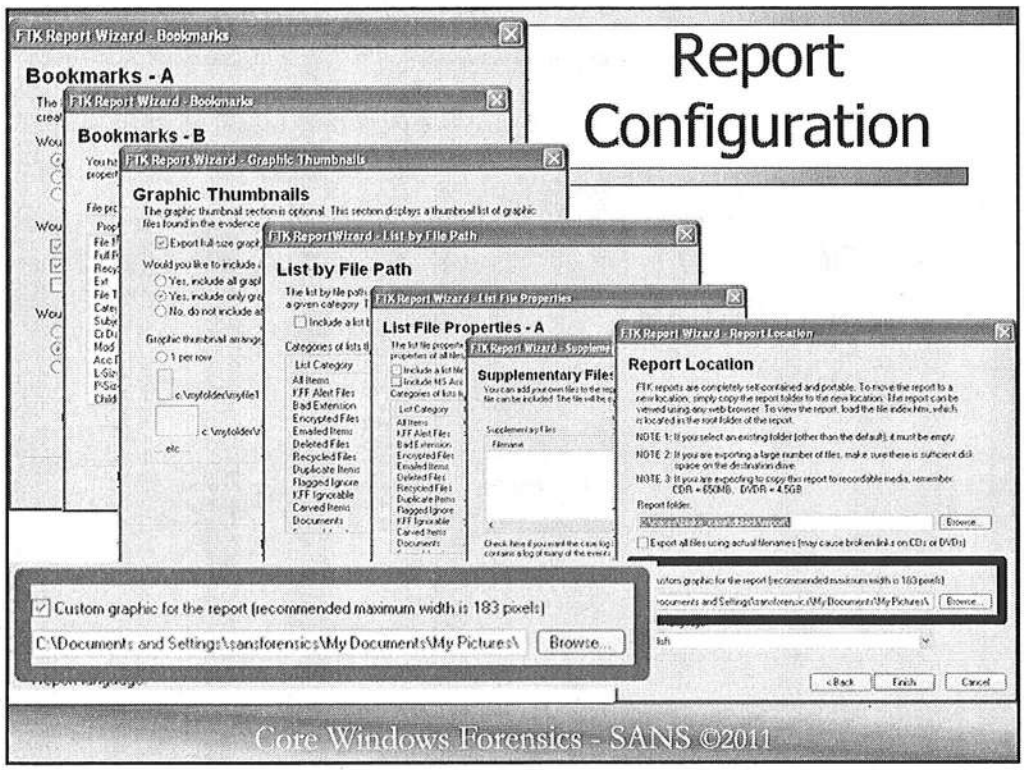
# Case Information

- Fill Out Case Information




Core Windows Forensics - SANS ©2011

The very next screen you will see is the Case information Screen. The information you fill out here will be placed on the case information page of your FTK report when generated. This information may be different than when you originally started the case. For example, your agency may choose to list the forensic examiners name here in the Investigator Name field.

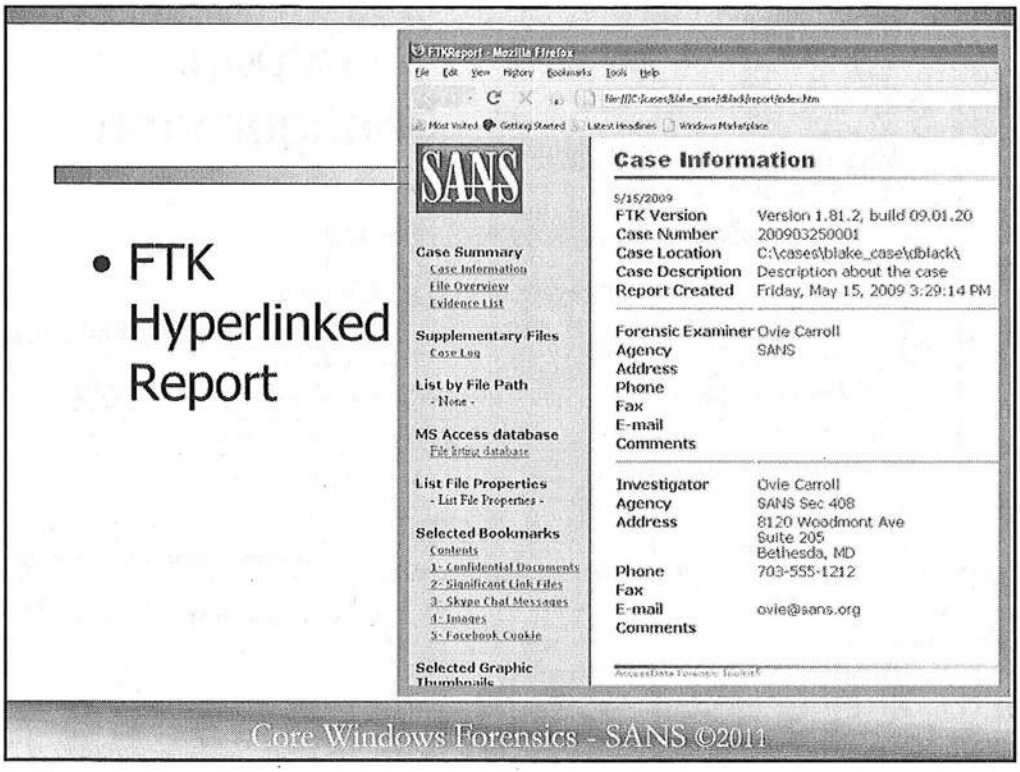



FTK will walk you through a series of options as to how you want to generate your report. FTK has so many options to allow you to tailor your report that you should first determine if your agency has a standard report option and if they do not then you should do some experimentation yourself to find they style, information and layout you like best.

I think, for the most part, the default options put way too much technical detail in the report, so I like to minimize all the file attributes. But as I said, rather than us trying to tell you how and what to put in your report, it is really best to talk to your agency to see what their policy is first.

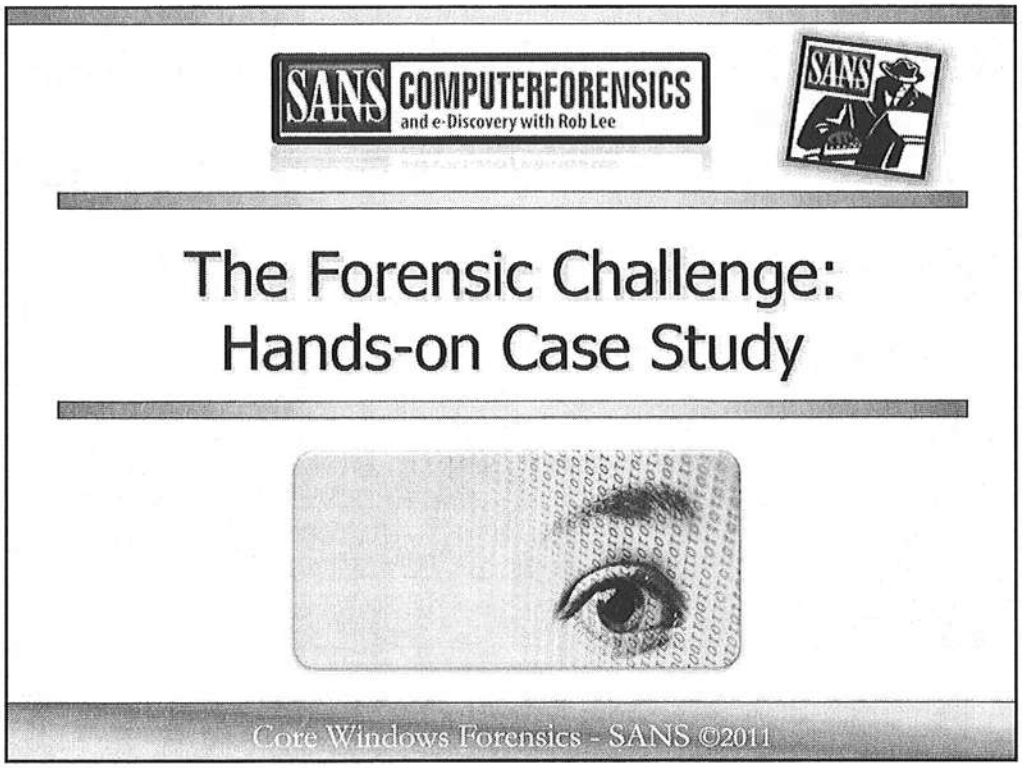
So for our purposes, we will just select all the default options and only make one change, that is on the dialog box that is titled "Report Location" at the bottom, you will see the option to add a **"Custom graphic for report"** Select the box next to that option then you can add your agency logo. Here, I have selected the SANS logo from my "**My Pictures**" folder.

Now all that is left is to click FINISHED.

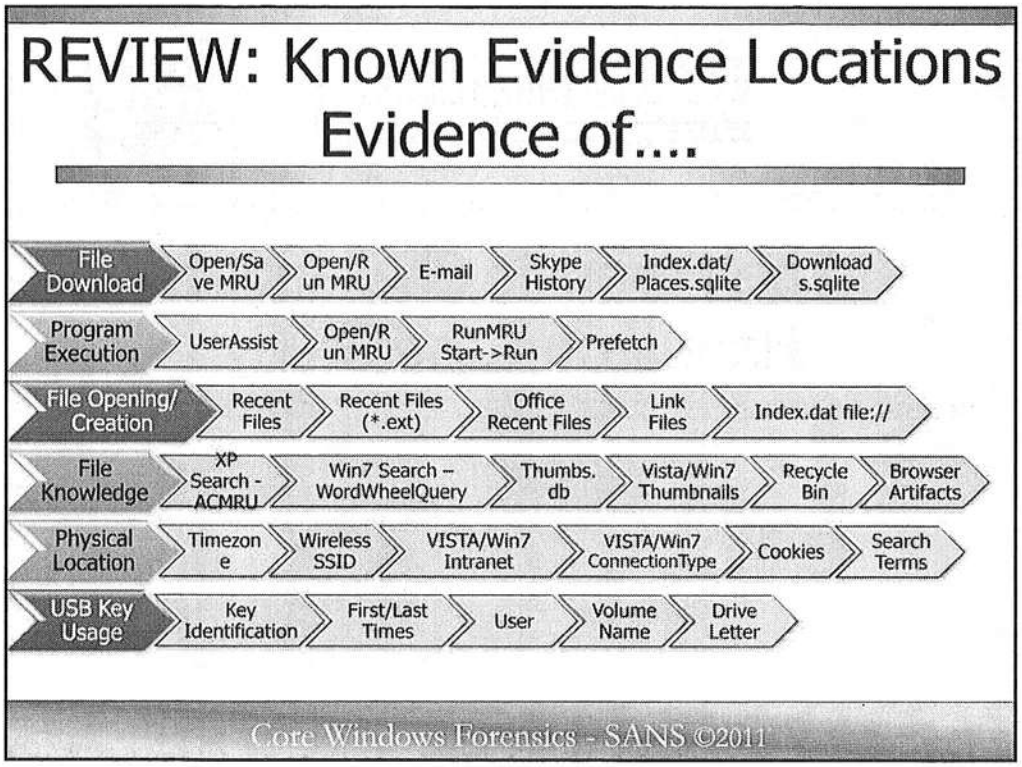



Once the report is finished being generated you will see a dialog box asking you if you want to view the report now. Select yes and the report will open with your default web browser. From here you can navigate the report like a web page. To distribute the report to your customer, depending on the amount of exported exhibits, you can burn the report folder to a CD or DVD. Include an autorun file so the report will automatically open when someone inserts the CD or DVD into the drive and you are done.

One last thought about reports. I have found, particularly if you are dealing with sensitive, pornographic or other type of files or images that might disturb the customer, if you create another HTML/web page as a splash screen with RED stripes and a big warning banner alerting the viewer that the information contained in this report are disturbing, sensitive, for official use, property of your agency, etc., it will provide you an extra layer of protection so no one complains they were psychologically damaged or exposed to information they were not suppose to be exposed to by viewing the contents of your report.




This page intentionally left blank.




This page intentionally left blank.

# Forensic Challenge and Mock-Trial (1)

- Analysis
  - You will be grouped into teams of 3-4 people each
  - Each team will analyze the case together and share findings/results
  - Over half the day will be spent analyzing the case image
- Prepare Summary
  - 1 Page Executive Summary using Day 1 Book to guide you
  - 1 5-10 slide Powerpoint Presentation showing the 5-10 key facts you uncover and what they mean to the case.
  - Use graphics as much as possible to convince the jury/judge of your findings.



Analysis

You will be grouped into teams of 3 people each

Each team will analyze the case together and share findings/results

Over half the day will be spent analyzing the case image

Prepare Summary

1 Page Executive Summary using Day 1 Book to guide you

1 5-10 slide Powerpoint Presentation showing the 5-10 key facts you uncover and what they mean to the case. Use graphics as much as possible to convince the jury/judge of your findings.

The summary statements from the two selected teams will be handed out to the class for examination.

# Forensic Challenge and Mock-Trial (2)

- Presentation Summary
  - Two Selected Teams will be given an additional 15-25 minutes to prepare to present
- Cross-Examination
  - Those not on the two teams will "cross-examine" the two teams based on their knowledge of the case
- Summary Presentation
  - Two teams will stand up in the class and read or orally present their summary
  - To make it fair, the team who didn't go first will be in another room.
  - Team that wins will be the team who presents the best summary as well as has thoughtful answers to cross examination questions posed from the class.
  - The instructor will be the moderator/judge of the winner.



Presentation Summary

Two Selected Teams will be given an additional 15 minutes to prepare to present.

Cross-Examination

Those not on the two teams will "cross-examine" the two teams based on their knowledge of the case.

Summary Presentation

Two teams will stand up in the class and read or orally present their summary.

To make it fair, the team who didn't go first will be in another room.

Team that wins will be the team who presents the best summary as well as answers thoughtfully to cross examination questions posed from the class.

The instructor will be the moderator/judge of the winner.

# Track Agenda

**Day 1 Forensic and E-Discovery Fundamentals**

- What you need to know to get started

**Day 2 Evidence Acquisition and Analysis**

- Proper Essential Evidence Collection and Analysis Techniques

**Day 3/4/5 Windows Forensics Analysis**

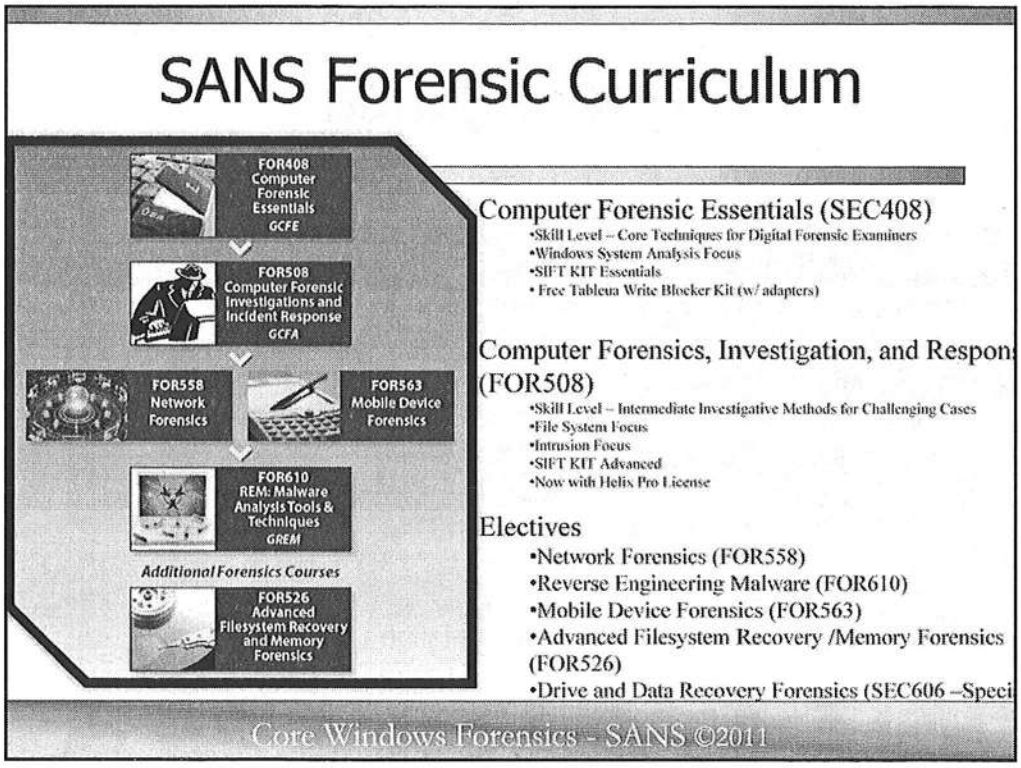
- Core Windows Forensic Knowledge

**Day 6 Windows Forensic Challenge**

- Putting Together A Real Case And Presentation

Core Windows Forensics - SANS ©2011

This page intentionally left blank.




Computer Forensic Essentials (SEC408)

- Skill Level – Core Techniques for Digital Forensic Examiners
- Windows System Analysis Focus
- SIFT KIT Essentials
- Free Tableua Write Blocker Kit (w/ adapters)

Computer Forensics, Investigation, and Response (FOR508)

- Skill Level – Intermediate Investigative Methods for Challenging Cases
- File System Focus
- Intrusion Focus
- SIFT KIT Advanced
- Now with Helix Pro License

Electives

- Network Forensics (FOR558)
- Reverse Engineering Malware (FOR610)
- Mobile Device Forensics (FOR563)
- Advanced Filesystem Recovery /Memory Forensics (FOR526)
- Drive and Data Recovery Forensics (SEC606)

# Forensic Links/References

**SANS**

- http://sansforensics.wordpress.com/
- http://www.sans.org/reading_room/white papers/forensics/
- http://www.sans.org/newsletters/

**Computer Forensic BLOGS**

- http://windowsir.blogspot.com
- http://www.forensickb.com/
- http://cfed-ttf.blogspot.com/
- http://seccure.blogspot.com/
- http://jessekornblum.livejournal.com
- http://computer.forensikblog.de/en/
- http://www.forensicblog.org
- http://geschonneck.com/index.php
- http://www.forensicblog.org
- http://www.forensicfocus.com
- http://forensicir.blogspot.com
- http://www.msuiche.net
- http://volatilesystems.blogspot.com/
- http://blog.mandiant.com
- http://computer.forensikblog.de/en/
- http://digfor.blogspot.com/

**e-Discovery Blogs**

- http://ralphlosey.wordpress.com
- http://www.granick.com/blog
- http://ridethelightning.senseient.com
- http://commonscold.typepad.com/eddupdate/
- http://www.ediscoverylaw.com
- http://www.ediscoverylaw.com
- http://legal-beagle.typepad.com

**Computer Forensics Podcasts**

- http://cyberspeak.libsyn.com
- http://forensic4cast.com

Computer Forensics Wiki
http://www.forensicswiki.org



This page intentionally left blank.

# THANK YOU!!

- Please thank your SANS behind-the-scenes staff and volunteers for putting together a wonderful training for you.
- Any additional questions:

  rlee@sans.org

  http://twitter.com/robtlee

  http://twitter.com/sansforensics



This page intentionally left blank.

SANS COMPUTERFORENSICS
and e-Discovery with Rob Lee

Here is my lens. You know my methods. -Sherlock Holmes

Any additional questions:
rlee@sans.org
http://twitter.com/robtlee
http://twitter.com/sansforensics



This page intentionally left blank.

This page intentionally left blank.

# System Configuration Profile Guide

| SYSTEM HIVE | |
|---|---|
| Identify the CurrentControlSet. | |
| **SYSTEM\Select**<br>**SYSTEM\Select\Current**<br>**Why is it important to know what the CurrentControlSet is?** | [redacted] |
| Identify the Time Zone of the machine. | |
| **SYSTEM\CurrentControlSet\Control\Time ZoneInformation**<br>Why is it important to know the time zone that the machine is set in? | [redacted] |
| Is the last access timestamp for files on or off? | |
| **SYSTEM\CurrentControlSet\Control\File System**<br>Located NtfsDisableLastAccessUpdate -> If set to 0x1or does not exist then Access time stamps are turned on and will update normally | |
| Identify the Computer Name of the machine. | |
| **SYSTEM\CurrentControlSet\Control\Comp uterName\ComputerName** | [redacted] |
| Identify the Network interfaces and their IP addresses. | |
| **SYSTEM\CurrentControlSet\Services\Tcp ip\Parameters\Interfaces** | [redacted] |
| Identify the Shares of the system. | |
| **SYSTEM\CurrentControlSet\Services\lan manserver\Shares** | [redacted] |
| Identify the Last shut down time. | |
| **SYSTEM\CurrentControlSet\Control\Wind ows (Shutdown Time)** | [redacted] |
| Identify Number of times shutdown was initiated. | |
| **SYSTEM\CurrentControlSet\Control\Watc hdog\Display (Shutdown Time)** | [redacted] |
| **SOFTWARE HIVE** | |
| Identify the networks the machine has connected to (if any). | |
| **XP ONLY - SOFTWARE\Microsoft\WZCSVC\Parameters\ Interfaces\{GUID}** | |
| **VISTA/WIN7 - SOFTWARE\Microsoft\WindowsNT\CurrentV ersion\NetworkList\Nla\Cache\Intranet** | |
| **VISTA/WIN7 - SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profile s (Vista/Win7)**<br>• **Nametype Value = 0x47 = Wireless**<br>• **Nametype Value = 0x06 =** | |

| | |
|---|---|
| Wired<br>• Nametype Value = 0x17 = Broadband (3g) | |
| Identify some of the System Autostart Programs. | |
| SOFTWARE\Microsoft\Windows\CurrentVersion\Run | |
| SOFTWARE\Microsoft\Windows\Current Version\RunOnce | |

# System Configuration Profile Guide

| SYSTEM HIVE | |
|---|---|
| Identify the CurrentControlSet. | |
| **SYSTEM\Select** | |
| **SYSTEM\Select\Current** | |
| **Why is it important to know what the CurrentControlSet is?** | |
| Identify the Time Zone of the machine. | |
| **SYSTEM\CurrentControlSet\Control\Time ZoneInformation** | |
| Why is it important to know the time zone that the machine is set in? | |
| Is the last access timestamp for files on or off? | |
| **SYSTEM\CurrentControlSet\Control\File System**<br>Located NtfsDisableLastAccessUpdate -> If set to 0x1or does not exist then Access time stamps are turned on and will update normally | |
| Identify the Computer Name of the machine. | |
| **SYSTEM\CurrentControlSet\Control\Comp uterName\ComputerName** | |
| Identify the Network interfaces and their IP addresses. | |
| **SYSTEM\CurrentControlSet\Services\Tcp ip\Parameters\Interfaces** | |
| Identify the Shares of the system. | |
| **SYSTEM\CurrentControlSet\Services\lan manserver\Shares** | |
| Identify the Last shut down time. | |
| **SYSTEM\CurrentControlSet\Control\Wind ows (Shutdown Time)** | |
| Identify Number of times shutdown was initiated. | |
| **SYSTEM\CurrentControlSet\Control\Watc hdog\Display (Shutdown Time)** | |
| **SOFTWARE HIVE** | |
| Identify the networks the machine has connected to (if any). | |
| **XP ONLY - SOFTWARE\Microsoft\WZCSVC\Parameters\ Interfaces\{GUID}** | |
| **VISTA/WIN7 - SOFTWARE\Microsoft\WindowsNT\CurrentV ersion\NetworkList\Nla\Cache\Intranet** | |
| **VISTA/WIN7 - SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profile s (Vista/Win7)**<br>• **Nametype Value = 0x47 = Wireless**<br>• **Nametype Value = 0x06 =** | |

| | |
|---|---|
| Wired<br>• Nametype Value = 0x17 = Broadband (3g) | |
| Identify some of the System Autostart Programs. | |
| SOFTWARE\Microsoft\Windows\CurrentVersion\Run | |
| SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce | |

# System Configuration Profile Guide

| SYSTEM HIVE | |
|---|---|
| Identify the CurrentControlSet. | |
| **SYSTEM\Select** | |
| **SYSTEM\Select\Current** | |
| **Why is it important to know what the CurrentControlSet is?** | |
| Identify the Time Zone of the machine. | |
| **SYSTEM\CurrentControlSet\Control\TimeZoneInformation** | |
| Why is it important to know the time zone that the machine is set in? | |
| Is the last access timestamp for files on or off? | |
| **SYSTEM\CurrentControlSet\Control\FileSystem**<br>Located NtfsDisableLastAccessUpdate -> If set to 0x1or does not exist then Access time stamps are turned on and will update normally | |
| Identify the Computer Name of the machine. | |
| **SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName** | |
| Identify the Network interfaces and their IP addresses. | |
| **SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces** | |
| Identify the Shares of the system. | |
| **SYSTEM\CurrentControlSet\Services\lanmanserver\Shares** | |
| Identify the Last shut down time. | |
| **SYSTEM\CurrentControlSet\Control\Windows (Shutdown Time)** | |
| Identify Number of times shutdown was initiated. | |
| **SYSTEM\CurrentControlSet\Control\Watchdog\Display (Shutdown Time)** | |
| **SOFTWARE HIVE** | |
| Identify the networks the machine has connected to (if any). | |
| **XP ONLY - SOFTWARE\Microsoft\WZCSVC\Parameters\Interfaces\{GUID}** | |
| **VISTA/WIN7 - SOFTWARE\Microsoft\WindowsNT\CurrentVersion\NetworkList\Nla\Cache\Intranet** | |
| **VISTA/WIN7 - SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles (Vista/Win7)**<br>• **Nametype Value = 0x47 = Wireless**<br>• **Nametype Value = 0x06 =** | |

| | |
|---|---|
| Wired<br>• Nametype Value = 0x17 = Broadband (3g) | |
| Identify some of the System Autostart Programs. | |
| SOFTWARE\Microsoft\Windows\CurrentVersion\Run | |
| SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce | |

# User Profile Exercise

| NTUSER.DAT HIVE | |
|---|---|
| XP Search History | |
| **NTUSER.DAT\Software\Microsoft\Search Assistant\ACMru**<br>5647->Computers/People<br>5001->Internet Search<br>5603->Filename Search<br>5604->Text Search | |
| Typed URLS | |
| **NTUSER.DAT\Software\Microsoft\Internet Explorer\TypedURLs** | |
| Recent Documents | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs**<br>Follow MRUList Order | |
| Open-> "Save As" Dialog Box | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU**<br>**Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePIDlMRU (VISTA)** | |
| Open-> "Run As" Dialog Box | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU**<br>**\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedPIDlMRU (VISTA)** | |
| Last Commands Executed | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU** | |
| Last Programs Executed via Explorer, Times Run, and Last Time Executed | |
| NTUSER.DAT\Software\Microsoft\Windows\Currentversion\Explorer\UserAssist\{GUID}\Count | |
| Identify some of the User Autostart Programs. | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Run** | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\RunOnce** | |

This page intentionally left blank.

| NTUSER.DAT HIVE | |
|---|---|
| XP Search History | |
| **NTUSER.DAT\Software\Microsoft\Search Assistant\ACMru**<br>5647->Computers/People<br>5001->Internet Search<br>5603->Filename Search<br>5604->Text Search | |
| Typed URLS | |
| **NTUSER.DAT\Software\Microsoft\Internet Explorer\TypedURLs** | |
| Recent Documents | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs**<br>Follow MRUList Order | |
| Open-> "Save As" Dialog Box | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU**<br>**Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePIDlMRU (VISTA)** | |
| Open-> "Run As" Dialog Box | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU**<br>**\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedPIDlMRU (VISTA)** | |
| Last Commands Executed | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU** | |
| Last Programs Executed via Explorer, Times Run, and Last Time Executed | |
| NTUSER.DAT\Software\Microsoft\Windows\Currentversion\Explorer\UserAssist\{GUID}\Count | |
| Identify some of the User Autostart Programs. | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Run** | |
| **NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\RunOnce** | |

This page intentionally left blank.

# Profile Windows XP USB Keys/Thumbdrives

| XP USB KEY/Thumbdrive | |
|---|---|
| **1. Write Down Vendor, Product, Version** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | Vendor =<br>Product =<br>Version = |
| **2. Write Down Serial Numbers** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | Serial Number = |
| **3. Determine Parent Prefix ID** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | Parent Prefix ID= |
| **4. Determine Vendor-ID (VID) and Product-(PID)** | |
| `SYSTEM\CurrentControlSet\Enum\USB ->` Perform search for S/N | `VID_XXXX =`<br>`PID_YYYY =` |
| **5. Determine Drive Letter Device Mapped To** | |
| `SYSTEM\MountedDevices->` Perform search for ParentIdPrefix in the Drive Letter | Drive = |
| **6. Write Down Volume GUIDs** | |
| `SYSTEM\MountedDevices->` Perform Search for ParentIdPrefix in the GUIDs | {GUID} = |
| **7. Find User That Used The Specific USB Device** | |
| `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2->` Search for Device GUID | User = |
| **8. Discover First Time Device Connected** | |
| `C:\Windows\setupapi.log` -> Perform search for Serial Number | Time/Timezone = |
| **9. Determine First Time Device Connected After Last Reboot** | |
| `SYSTEM\CurrentControlSet\Enum\USB\VID_XXXX&PID_YYYY ->` Perform search for Serial Number (Last Written Time of Serial Number Key) | Time/Timezone = |
| **10. Determine Last Time Device Connected** | |
| `NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID} ->` Perform search for Device {GUID} | Time/Timezone = |

# Profile VISTA USB Key/Thumbdrives

| VISTA USB KEY/Thumbdrive | |
|---|---|
| **1. Write Down Vendor, Product, Version** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | Vendor =<br>Product =<br>Version = |
| **2. Write Down Serial Numbers** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | Serial Number = |
| **3. Determine Vendor-ID (VID) and Product-(PID)** | |
| `SYSTEM\CurrentControlSet\Enum\USB ->` Perform search for S/N | `VID_XXXX =`<br>`PID_YYYY =` |
| **4. Write Down Volume GUIDs** | |
| `SYSTEM\MountedDevices->` Perform Search for Serial Number | GUID = |
| **5. Determine Drive Letter and Volume Name Device Mapped To** | |
| `SOFTWARE\Microsoft\Windows Portable Devices\Devices->` Perform Search for Serial Number and Match with Volume Name<br>**Or** | Drive Letter =<br>Volume Name= |
| **6. Find User That Used The Specific USB Device** | |
| `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2->` Search for Device GUID | User = |
| **7. Discover First Time Device Connected** | |
| `C:\Windows\inf\setupapi.dev.log` -> Perform search for Serial Number | Time/Timezone = |
| **8. Determine First Time Device Connected After Last Reboot** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR\Vendor_Product_Version ->` Perform search for Serial Number (Last Written Time of Serial Number Key) | Time/Timezone = |
| **9. Determine Last Time Device Connected** | |
| `SYSTEM\CurrentControlSet\Enum\USB\VID_XXXX&PID_YYYY ->` Perform search for Serial Number (Last Written Time of Serial Number Key)<br>**or**<br>`NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID} ->` Perform search for Device {GUID} | Time/Timezone = |

# Profile Windows 7 USB Keys/Thumbdrives

Windows 7

| Win7 USB Key/Thumbdrive | |
|---|---|
| **1. Write Down Vendor, Product, Version** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | Vendor =<br>Product =<br>Version = |
| **2. Write Down Serial Numbers** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | Serial Number = |
| **3. Determine Vendor-ID (VID) and Product-(PID)** | |
| `SYSTEM\CurrentControlSet\Enum\USB ->` Perform search for S/N | `VID_XXXX =`<br>`PID_YYYY =` |
| **4. Determine Volume Name Device Mapped To** | |
| `SYSTEM\MountedDevices->` Perform search for Serial Number in the Drive Letters | Volume Name = |
| **5. Determine Drive Letter Device Mapped To** | |
| `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs ->` Perform Search for Volume Name | Drive Letter = |
| **6. Write Down Volume GUIDs** | |
| `SYSTEM\MountedDevices->` Perform Search for Serial Number in the GUIDs | GUID = |
| **7. Find User That Used The Specific USB Device** | |
| `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2->` Search for Device GUID | User = |
| **8. Discover First Time Device Connected** | |
| `C:\Windows\inf\setupapi.dev.log` -> Perform search for Serial Number | Time/Timezone = |
| **9. Determine First Time Device Connected After Last Reboot** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR\Vendor_Product_Version ->` Perform search for Serial Number (Last Written Time of Serial Number Key) | Time/Timezone = |
| **10. Determine Last Time Device Connected** | |
| `SYSTEM\CurrentControlSet\Enum\USB\VID_XXXX&PID_YYYY ->` Perform search for Serial Number (Last Written Time of Serial Number Key)<br>**or**<br>`NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID} ->` Perform search for Device {GUID} | Time/Timezone = |

This page intentionally left blank.

# Profile Windows XP USB Drive Enclosures

| XP USB Drive Enclosures | |
|---|---|
| **1. Write Down Vendor, Product, Version** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | |
| **2. Write Down Serial Numbers** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | Serial Number = |
| **3. Determine Vendor-ID (VID) and Product-(PID)** | |
| `SYSTEM\CurrentControlSet\Enum\USB ->` Perform search for S/N | `VID_XXXX =`<br>`PID_YYYY =` |
| **4. Determine Disk Identity** | |
| `Examine MBR of hard drive (Sector 0)->` 4 byte hex value starting at offset 440 | Disk ID = |
| **5. Determine Drive Letter Device Mapped To** | |
| `SYSTEM\MountedDevices->` Perform search for Disk ID in Drive Letters | Drive = |
| **6. Write Down Volume GUIDs** | |
| `SYSTEM\MountedDevices->` Perform Search for Disk ID in the GUIDs | {GUID} = |
| **7. Find User That Used The Specific USB Device** | |
| `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2->` Search for Device GUID | User = |
| **8. Discover First Time Device Connected** | |
| `C:\Windows\setupapi.log` -> Perform search for Serial Number | Time/Timezone = |
| **9. Determine First Time Device Connected After Last Reboot** | |
| `SYSTEM\CurrentControlSet\Enum\USB\VID_XXXX&PID_YYYY ->` Perform search for Serial Number (Last Written Time of Serial Number Key) | Time/Timezone = |
| **10. Determine Last Time Device Connected** | |
| `NTUSER//Software/Microsoft/Windows/CurrentVersion/Explorer/MountPoints2/{GUID} ->` Perform search for Device {GUID} | Time/Timezone = |

# Profile VISTA USB Drive Enclosures

| VISTA USB Drive Enclosures | |
|---|---|
| **1. Write Down Vendor, Product, Version** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | |
| **2. Write Down Serial Numbers** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | Serial Number = |
| **3. Determine Vendor-ID (VID) and Product-(PID)** | |
| `SYSTEM\CurrentControlSet\Enum\USB ->` Perform search for S/N | `VID_XXXX =`<br>`PID_YYYY =` |
| **4. Determine Disk Identity** | |
| `Examine MBR of hard drive (Sector 0)->` 4 byte hex value starting at offset 440 | Disk ID = |
| **5. Write Down Volume GUIDs** | |
| `SYSTEM\MountedDevices->` Perform Search for Disk ID | GUID = |
| **6. Determine Drive Letter Device Mapped To** | |
| `SYSTEM\MountedDevices->` Perform search for Disk ID in the Drive Letters | Drive = |
| **7. Find User That Used The Specific USB Device** | |
| `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2->` Search for Device GUID | User = |
| **8. Discover First Time Device Connected** | |
| `C:\Windows\inf\setupapi.dev.log` -> Perform search for Serial Number | Time/Timezone = |
| **9. Determine First Time Device Connected After Last Reboot** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR\Vendor_Product_Version ->` Perform search for Serial Number (Last Written Time of Serial Number Key) | Time/Timezone = |
| **10. Determine Last Time Device Connected** | |
| `SYSTEM\CurrentControlSet\Enum\USB\VID_XXXX&PID_YYYY ->` Perform search for Serial Number (Last Written Time of Serial Number Key) | Time/Timezone = |

# Profile Windows 7 USB Drive Enclosures

Windows 7

| Win7 USB Drive Enclosures | |
|---|---|
| **1. Write Down Vendor, Product, Version** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | |
| **2. Write Down Serial Numbers** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR` | Serial Number = |
| **3. Determine Vendor-ID (VID) and Product-(PID)** | |
| `SYSTEM\CurrentControlSet\Enum\USB ->` Perform search for S/N | `VID_XXXX =`<br>`PID_YYYY =` |
| **4. Determine Disk Identity** | |
| `Examine MBR of hard drive (Sector 0)->` 4 byte value starting at offset 440 | Disk ID = |
| **5. Determine Drive Letter Device Mapped To** | |
| `SYSTEM\MountedDevices->` Perform search for Disk ID in the Drive Letters | Drive = |
| **6. Write Down Volume GUIDs** | |
| `SYSTEM\MountedDevices->` Perform Search for Disk ID in the GUIDs | GUID = |
| **7. Find User That Used The Specific USB Device** | |
| `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2->` Search for Device GUID | User = |
| **8. Discover First Time Device Connected** | |
| `C:\Windows\inf\setupapi.dev.log` -> Perform search for Serial Number | Time/Timezone = |
| **9. Determine First Time Device Connected After Last Reboot** | |
| `SYSTEM\CurrentControlSet\Enum\USBSTOR\Vendor_Product_Version ->` Perform search for Serial Number (Last Written Time of Serial Number Key) | Time/Timezone = |
| **10. Determine Last Time Device Connected** | |
| `SYSTEM\CurrentControlSet\Enum\USB\VID_XXXX&PID_YYYY -> Perform search for Serial Number (Last Written Time of Serial Number Key)` | Time/Timezone = |

This page intentionally left blank.

# APPENDIX E

# Forensic Log Parsing with Microsoft's LogParser

July 17th, 2003

By Mark Burnett

Investigating a web-based intrusion can be a daunting task, especially when you have no information other than knowing it was web-based. It is easy to waste precious time digging through megabytes, perhaps even gigabytes, of log files trying to locate suspicious activity. Often this search turns up little useful evidence.

Consider this scenario: an e-commerce site receives several reports from customers about unauthorized orders on their accounts. They suspect that someone has compromised their web-based ordering system so they gather the log files from several different IIS web servers. They have the dates and times of the orders, but the corresponding IP addresses in the log files turn out to be anonymous proxies used by the suspect. Searching for activity from those IP addresses in the log files turns up nothing. Browsing through the raw log files for those dates also turns up nothing. Somehow, someone found a flaw in the ordering system but he or she could have discovered the flaw months before exploiting it. Tracking down the flaw and IP addresses used by the suspect seems impossible. But there are techniques that can facilitate log file forensics. The purpose of this article is to demonstrate log file forensics of IIS logs using SQL queries with Microsoft's LogParser tool.

**IIS Log Fields**

The first step is to prepare for security incidents by logging as much information as possible. IIS can log a significant amount of information about each web request, but many of the available log fields are not enabled by default. To enable full logging, open the Internet Services Manager and edit the Extended Logging Properties to include all available log fields. Much of this information has some forensics value as shown in Table 1.

**Table 1: IIS Log Fields**

| Field Name | Description | Uses |
|---|---|---|
| Date (date) | The date of the request. | Event correlation. |
| Time (time) | The UTC time of the request. | Event correlation, determine time zone, |

| | | |
|---|---|---|
| | | identify scanning scripts. |
| Client IP Address (c-ip) | The IP address of the client or proxy that sent the request. | Identify user or proxy server. |
| User Name (cs-username) | The user name used to authenticate to the resource. | Identify compromised user passwords. |
| Service Name (s-sitename) | The W3SVC instance number of the site accessed. | Can verify the site accessed if the log files are later moved from the system. |
| Server Name (s-computername) | The Windows host name assigned to the system that generated the log entry. | Can verify the server accessed if the log files are later moved from the system. |
| Server IP Address (s-ip) | The IP address that received the request. | Can verify the IP address accessed if the log files are later moved from the system or if the server is moved to a new location. |
| Server Port (s-port) | The TCP port that received the request. | To verify the port when correlating with other types of log files. |
| Method (cs-method) | The HTTP method used by the client. | Can help track down abuse of scripts or executables. |
| URI Stem (cs-uri-stem) | The resource accessed on the server. | Can identify attack vectors. |
| URI Query (cs-uri-query) | The contents of the query string portion of the URI. | Can identify injection of malicious data. |
| Protocol Status (sc-status) | The result code sent to the client. | Can identify CGI scans, SQL injection and other intrusions. |
| Win32 Status (sc-win32-status) | The Win32 error code produced by the request. | Can help identify script abuse. |
| Bytes Sent (sc-bytes) | The number of bytes sent to the client. | Can help identify unusual traffic from a single script. |
| Bytes Received (cs-bytes) | The number of bytes received from the client. | Can help identify unusual traffic to a single script. |
| Time Taken (time-taken) | The amount of server time, in milliseconds, taken to process the request. | Can identify unusual activity from a single script. |
| Protocol Version (cs-version) | The HTTP protocol version supplied by the client. | Can help identify older scripts or browsers. |

| | | |
|---|---|---|
| Host (cs-host) | The contents of the HTTP Host header sent by the client. | Can determine if the user browsed to the site by IP address or host name. |
| User Agent (cs(User-Agent)) | The contents of the HTTP User-Agent header sent by the client. | Can help uniquely identify users or attack scripts. |
| Cookie (cs(Cookie)) | The contents of the HTTP Cookie header sent by the client. | Can help uniquely identify users. |
| Referer (cs(Referer)) | The contents of the HTTP Referer header sent by the client. | Can help identify the source of an attack or see if an attacker is using search engines to find vulnerable sites. |

While I normally recommend logging all fields, the actual fields you choose to log should be based on a balance between forensics capabilities and disk space.

**Custom Logging**

IIS does provide many log fields, but there may be other fields you wish to record. For example, if the request comes from a proxy server, you may want to see if the proxy server sends the client's real IP address through other HTTP headers. For example, some proxy servers add the "X-Forwarded-For" header containing the client's real IP address.

IIS has a limited capability to log custom fields through the Response.AppendToLog method. The limitation, however, is that a new field is not created in the log files, but this data is appended to the URI Query field. To distinguish the two values, you can separate them with a character such as the pipe ("|"). Below is example ASP code to log additional proxy headers:

```
<%
sHeader= Request.ServerVariables("X-Forwarded-For")
If Len(sHeader) Then Response.AppendToLog "|" & sHeader
%>
```

Note that other common proxy headers are Forwarded, Client_IP, Remote_Addr, Remote_Host, Forwarded, VIA, HTTP_From, Remote_Host_Wp, Xonnection, Xroxy_Connection, and X_Locking.

**Microsoft's LogParser Tool**

Digging through logs requires that you have some common interface to perform queries across hundreds of individual log files. One method is to dump all the logs into an SQL database. Another solution is Microsoft's LogParser tool. This robust tool provides an SQL interface to a variety of log file formats and is fast enough for log file analysis of most web sites. I won't go into detail here about how to use LogParser, but the document included with the package is very

helpful to get started. Because LogParser is a command-line tool, I have found it useful to either to copy the file to the C:\WINNT directory or to add the LogParser directory to your PATH variable.

You can download Microsoft's LogParser 2.0 here, but the IIS 6 Resource Kit includes LogParser 2.1, which has some new features. Although LogParser 2.1 runs fine on a Win2k system, you cannot install the IIS 6 resource kit on Win2k. However, you can manually extract the resource kit files using the command: iis60rkt.exe /V/a.

It is important to note that when doing any log file processing, be sure to work on copies of the logs to help preserve the integrity of the original files (see Maintaining Credible Logfiles). I also find it helpful to only copy those logs for the time period I want to analyze to reduce the size of the query results.

This article will demonstrate many of the forensic capabilities of LogParser. Keep in mind that I wrote each of these example queries for a typical configuration, therefore you may need to adjust them for your particular site. Not all queries listed here will be effective for you, depending on your site configuration and traffic level.

**Finding the Intrusion**

If you do not know anything about the intruder or the nature of the intrusion, you must first do some high-level queries to know where to start your hunt. Most attacks leave some kind of trail or have some side-effect on your server. The trick is finding them.

**Trojan Files**

Before we dig in to the actual log files, it may be useful to do a quick check of the newest files on the web site. If the intruder was able to create or modify files within the web content directories, he or she may have uploaded Trojan ASP scripts or executables. You might just get lucky and find these files. The following query lists the 20 newest files on the web site:

```
C:\>logparser -i:FS "SELECT TOP 20 Path, CreationTime from
c:\inetpub\wwwroot\*.* ORDER BY CreationTime DESC" -rtp:-1

 Path                                                      CreationTime ---
------------------------------------------------------- ------------------
c:\inetpub\wwwroot\Default.asp                            6/22/2003 6:00:01
c:\inetpub\wwwroot\About.asp                              6/22/2003 6:00:00
c:\inetpub\wwwroot\global.asa                             6/22/2003 6:00:00
c:\inetpub\wwwroot\Products.asp                           6/22/2003 6:00:00
...
```

And this query lists the 20 most recently modified files:

```
C:\>logparser -i:FS "SELECT TOP 20 Path, LastWriteTime from
c:\inetpub\wwwroot\*.* ORDER BY LastWriteTime DESC" -rtp:-1
```

```
 Path                                                          LastWriteTime --
------------------------------------------------------------ ------------------
c:\inetpub\wwwroot\Default.asp                                 6/22/2003
14:00:01 c:\inetpub\wwwroot\About.asp
6/22/2003 14:00:00 c:\inetpub\wwwroot\global.asa
6/22/2003 6:00:00 c:\inetpub\wwwroot\Products.asp
6/22/2003 6:00:00 ...
```

But suppose the attacker was careful and deleted all Trojan files when finished. In that case, the files will not exist, but there will be log entries showing successful requests for those files. To identify these log entries you must make a list of all files on your site that have resulted in 200 HTTP status codes. From your log files directory, execute the following query:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT DISTINCT TO_LOWERCASE(cs-
uri-stem) AS URL, Count(*) AS Hits FROM ex*.log WHERE sc-status=200 GROUP BY
URL ORDER BY URL" -rtp:-1

 URL                                          Hits ------------------------------
---------- ----- /About.asp                                   122 /Default.asp
9823 /downloads/setup.exe                          701 /files.zip
1 /Products.asp                                   8341 /robots.txt
2830 ...
```

Carefully review this list and make sure that each item listed is part of your web application. In particular, watch for files such as nc.exe, tini.exe, root.exe, cmd.exe, upload.asp, aspexec.asp, etc.

**Script Abuse**

If searching for new or modified files turns up nothing, it is time to check out your scripts and executables. Any script or executable that accepts user input is a potential attack vector. Before starting, you should identify which executable file extensions you use in your web content areas. The following query will give you a report of all file extensions that exist within your web content (adjust the path names as necessary):

```
C:\>logparser -i:fs "SELECT TO_LOWERCASE(SUBSTR(Name,
LAST_INDEX_OF(Name,'.'), STRLEN(Name))) AS Extension, Count(*) as Files from
c:\inetpub\wwwroot\*.*, c:\inetpub\scripts\*.* WHERE Attributes NOT LIKE 'D%'
GROUP BY Extension ORDER BY Files DESC" -rtp:-1

 Extension Files --------- ----- .gif      704 .asp      180 .jpg      44
.css      43 .htm      28 .txt      21 .html     6 .dll      5 .zip      4
```

According to this list, the site contains several file extensions that may be of concern to us: .asp and .dll. Therefore, all the example queries from this point on will specifically look for ASP and DLL files. You will likely need to adjust this depending on which executable extensions you use on your web site.

One way to detect script abuse is to see if any one script has an unusually high number of hits. Since web-based attacks often require some trial and error, you should expect to see noticeable

statistical variances, unless of course your web site gets millions of hits a day. Nevertheless, it is sometimes useful to see if any single day produced unusually high traffic.

The following query will show the number of hits for each day for each ASP and DLL file. From your log files directory, type the following:

```
C:\WINNT\System32\LogFiles\W3SVC1>LogParser "SELECT
TO_STRING(TO_TIMESTAMP(date, time), 'yyyy-MM-dd') AS Day, cs-uri-stem,
COUNT(*) AS Total FROM ex*.log WHERE (sc-status<400 or sc-status>=500) AND
(TO_LOWERCASE(cs-uri-stem) LIKE '%.asp%' OR TO_LOWERCASE(cs-uri-stem) LIKE
'%.exe%') GROUP BY Day, cs-uri-stem ORDER BY cs-uri-stem, Day" -rtp:-1

 Day        cs-uri-stem         Total ---------- ------------------- -----
2003-04-01 /Default.asp        127 2003-04-02 /Default.asp        121 2003-
04-03 /Default.asp        132 2003-04-04 /Default.asp        116 2003-04-05
/Default.asp        107 2003-04-06 /Default.asp        144 2003-04-07
/Default.asp        466 2003-04-08 /Default.asp        174 2003-04-09
/Default.asp        118 ...
```

In the sample results above the number of hits on 2004-04-07 is suspiciously high and should be investigated further.

Another good attack indicator is the number of errors per hour. The following script returns the dates and hours that had more than 25 error codes returned. This value will likely need adjusting depending on how much traffic your site receives:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT date, QUANTIZE(time,
3600) AS hour, sc-status, Count(*) AS Errors FROM ex03*.log WHERE sc-
status>=400 GROUP BY date, hour, sc-status HAVING Errors>25 ORDER BY Errors
DESC" -rtp:-1

 date       hour     sc-status Errors ---------- -------- --------- ------
2003-06-22 22:00:00 404       110 2003-04-21 13:00:00 404       36 2003-04-19
23:00:00 404       36 2003-04-19 13:00:00 404       27 ...
```

Further investigation of the dates listed above may show that the high number of 404 errors are CGI scans looking for vulnerable scripts on your site. The 404 errors themselves are not as much as a concern as are the 200 results during that same time that may indicate a successful attack. This query will return all valid requests from any IP address that also had a 404 error on 2003-06-22:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT c-ip, cs-uri-stem,
Count(*) as Hits FROM ex*.log WHERE TO_LOWERCASE(cs-uri-stem) NOT LIKE
'%.gif' AND TO_LOWERCASE(cs-uri-stem) NOT LIKE '%.jpg%' AND c-ip IN (SELECT
c-ip FROM ex030622.log WHERE sc-status=404) AND sc-status=200 GROUP BY c-ip,
cs-uri-stem" -rtp:-1

 c-ip            cs-uri-stem         Hits --------------- -------------------
---------------- 199.154.189.199 /Default.asp        3 199.154.189.199
/main.css           3 199.154.189.199 /Products.asp        7 199.154.189.199
/About.asp          1 63.54.202.2     /Products.asp        18 63.54.202.2
```

```
/main.css            1 81.112.9.62     /Default.asp        1 ...
```

Looking at these results, you can see two IP addresses that had an unusual number of hits on Products.asp. It could be that these were two different attackers or the same attacker who used two different proxies to conceal his or her IP address. One way to find out if they are likely the same person is to check the User-Agent header for the two different IP addresses:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT DISTINCT c-ip, cs(User-
Agent) FROM ex030622.log WHERE c-ip='198.54.202.2' or c-ip='62.135.71.223'" -
rtp:-1

 c-ip              cs(User-Agent) ---------------- --------------------------
---------------------------------- 63.54.202.2
Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+Q312461;+.NET+CLR+1.0.3705
199.154.189.199
Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+Q312461;+.NET+CLR+1.0.3705
)
```

This proves that those two IP addresses are either the same user or two different users with the exact same OS, browser, service pack level, installed hotfixes, and .NET installation. It is not a perfect indicator but it is significant. To support this evidence, you could go through your logs and discover when each of the IP addresses first hit your web site. When a user visits a web site for the first time, the browser downloads the page and any graphics and stores it all in the browser's temporary cache. This is so that subsequent visits to the page will not require downloading all the graphics again. However, the browser does check to see if the graphics have been modified before using the cached versions. If the graphic has not been modified, the server will return a 304 HTTP status code. Therefore, if you create a query for a specific IP address with a status code of 200 for any particular graphic, that log entry will be the user's first visit, providing they have not cleared their cache. So if a user switches to a different proxy server, the file will still be cached and therefore there will never be a first visit from one of the IP addresses. If one of the two IP addresses mentioned above turns up not having a first visit, chances are that they first visited the site from the other IP address. If neither IP address shows a 200 result, then there are more IP addresses left to discover.

**SQL Injection**

If you read this paper (PDF) from NGSSoftware you will see that attacks such as SQL injection are based on sending faulty requests to a server and interpreting the error messages. Some of the indicators of this type of attack are:

- Numerous sequential hits from the same IP address to the same URL;
- High numbers of 500 HTTP status codes or other errors;
- GET requests to ASP pages that normally only receive POST requests; and
- Other clusters of anomalous web site activity.

It may also be useful to see an unusually high number of hits on a single page from a single IP address. The following query shows any IP address that hit the same page more than 50 times in a single day:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT DISTINCT date, cs-uri-
stem, c-ip, Count(*) AS Hits FROM ex*.log GROUP BY date, c-ip, cs-uri-stem
HAVING Hits>50 ORDER BY Hits Desc" -rtp:-1

 date       cs-uri-stem                          c-ip            Hits --------
-- ---------------------------------- --------------- ---- 2003-05-19
/Products.asp                        203.195.18.24   281 2003-06-22
/Products.asp                        210.230.200.54  98 2003-06-05
/Products.asp                        203.195.18.24   91 2003-05-07
/Default.asp                         198.132.116.174 74 ...
```

Looking at these results, it is immediately obvious that one IP address hit the same page 281 times one day and 91 times another day, which is obviously suspicious.

Another useful technique is to view exactly what ASP errors IIS encountered while serving requests. Most attempts at breaking into a web site will inevitably result in some kind of error. The following query will return a list of every ASP error recorded in the log files:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-query, Count(*) AS
Total FROM ex*.log WHERE sc-status>=500 GROUP BY cs-uri-query ORDER BY Total
DESC " -rtp:-1

 cs-uri-query
Total ---------------------------------------------------------------------
-- ---------- Out-of-process+ISAPI+extension+request+failed.
18 |55|8000ffff|Catastrophic_failure__
8 |49|8000ffff|Catastrophic_failure__                                      6
|74|800a01c2|Wrong_number_of_arguments_or_invalid_property_assignment     1
...
```

If you find any errors that are interesting, you could write another query to drill down to the specific error. In particular, you want to watch for ODBC and ADO errors, indicating a possible attempt at SQL injection.

Another way to identify errors is to look at the status codes returned by the server. If you want to see a detail of what status codes IIS returned for each page, try the following query:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem, sc-status,
Count(*) AS Total FROM ex*.log WHERE TO_LOWERCASE(cs-uri-stem) LIKE '%.asp%'
or TO_LOWERCASE(cs-uri-stem) LIKE '%.exe%' GROUP BY cs-uri-stem, sc-status
ORDER BY cs-uri-stem, sc-status" -rtp:-1

 cs-uri-stem                                  sc-status Total --------------
------------------------------ --------- ----- /Default.asp
200       9258 /Default.asp                                  500       3
/MSOffice/cltreq.asp                          404       12 /MailResult.asp
404       1 /asp/aspmail.asp                                302       86
/asp/aspmail.asp                              500       28 /autocomplete.asp
404       2 /awards.asp                                     404       4 ...
```

Also of interest are the Win32 Status codes, which may be attack indicators:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem,
WIN32_ERROR_DESCRIPTION(sc-win32-status) as Error, Count(*) AS Total FROM
ex*.log WHERE sc-win32-status>0 and (TO_LOWERCASE(cs-uri-stem) LIKE '%.asp%'
or TO_LOWERCASE(cs-uri-stem) LIKE '%.exe%') GROUP BY cs-uri-stem, Error ORDER
BY cs-uri-stem, Error" -rtp:-1

 cs-uri-stem              Error                                  Total --------
---------------- ------------------------------------ ----- /Default.asp
The RPC server is unavailable.       2 /Default.asp              The remote
procedure call failed     1 /asp/aspmail.asp          The RPC server is
unavailable.       12 /download/Default.asp    The RPC server is unavailable.
3 ...
```

Some ASP pages should only accept form input from previous pages. If, for example, you may have a page such as checkout1.asp that sends a POST request to checkout2.asp, then anything other than a POST request to checkout2.asp may be suspicious. This query will show what HTTP methods were sent to each page:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem, cs-method,
Count(*) AS Total FROM ex*.log WHERE (sc-status<400 or sc-status>=500) AND
(TO_LOWERCASE(cs-uri-stem) LIKE '%.asp%' or TO_LOWERCASE(cs-uri-stem) LIKE
'%.exe%') GROUP BY cs-uri-stem, cs-method ORDER BY cs-uri-stem, cs-method" -
rtp:-1

 cs-uri-stem                          cs-method Total ----------------------
------------- --------- ----- /Default.asp                          GET
9136 /Default.asp                          HEAD      125 /asp/aspmail.asp
GET       3 /asp/aspmail.asp                       POST      111
/awards/Default.asp                   GET       269 /compare/Default.asp
GET       437 /compare/Default.asp                 HEAD      3
/download/Default.asp                 GET       5018 /download/Default.asp
HEAD      436 /download/default.asp                GET       727
/download/default.asp                 HEAD      1 /orders/Default.asp
GET       1420 /orders/Default.asp                  POST      3 ...
```

You may also want to write a query that checks the HTTP referer header to make sure the traffic is coming from where you expect it to be coming from.

**Digging Deeper**

At this point, you should begin to see patterns emerge. You should be able to narrow down the attack to specific dates and URL's. If you still have not found any apparent patterns, you may need to dig deeper. Sometimes an attack will involve sending a large amount of information back to the attacker. The following query will report some statistics for the number of bytes sent to the client

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem, Count(*) as
Hits, AVG(sc-bytes) AS Avg, Max(sc-bytes) AS Max, Min(sc-bytes) AS Min,
Sum(sc-bytes) AS Total FROM ex*.log WHERE TO_LOWERCASE(cs-uri-stem) LIKE
'%.asp%' or TO_LOWERCASE(cs-uri-stem) LIKE '%.exe%' GROUP BY cs-uri-stem
ORDER BY cs-uri-stem" -rtp:-1
```

```
 cs-uri-stem                Hits  Avg    Max     Min  Total -------------------
----- ----- ------ ------- ---- -------- /Default.asp              9261  18321
19920   145  16967359 /MSOffice/cltreq.asp     12    227    269     221  2724
/MailResult.asp          1     221    221     221  221 /asp/aspmail.asp
114   545    704     218  62232 /complete.asp            2     230    240
221  461 /orders/Default.asp      269   6998   7625    6692 1882463 ...
```

And this one will report on bytes sent from the client:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem, Count(*) as
Hits, AVG(cs-bytes) AS Avg, Max(cs-bytes) AS Max, Min(cs-bytes) AS Min,
Sum(cs-bytes) AS Total FROM ex*.log WHERE TO_LOWERCASE(cs-uri-stem) LIKE
'%.asp%' or TO_LOWERCASE(cs-uri-stem) LIKE '%.exe%' GROUP BY cs-uri-stem
ORDER BY cs-uri-stem" -rtp:-1

 cs-uri-stem                 Hits  Avg  Max  Min Total ----------------------
----- ----- ---- ---- --- ------- /Default.asp                9261  435  1544
49  4037788 /MSOffice/cltreq.asp        12    369  482  276 4430
/MailResult.asp             1     313  313  313 313 /asp/aspmail.asp
114   1418 2383 153 161685 /complete.asp                2     172  191  154
345 /orders/Default.asp         269   441  1062 118 118766 ...
```

Another indicator may be how much time the server spent processing the request. It is not uncommon for exploits to take an unusually large amount of time or even timeout completely. The following query reports on time taken:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-uri-stem, Count(*) as
Hits, AVG(time-taken) AS Avg, Max(time-taken) AS Max, Min(time-taken) AS Min,
Sum(time-taken) AS Total FROM ex*.log WHERE TO_LOWERCASE(cs-uri-stem) LIKE
'%.asp%' or TO_LOWERCASE(cs-uri-stem) LIKE '%.exe%' GROUP BY cs-uri-stem
ORDER BY cs-uri-stem" -rtp:-1

 cs-uri-stem                Hits  Avg    Max     Min Total -------------------
------ ----- ------ ------- --- --------- /Default.asp              9261  8
312     0   75228 /MSOffice/cltreq.asp      12    4      16      0   48
/MailResult.asp           1     0      0       0   0 /asp/aspmail.asp
114   699    31719   0   79765 /complete.asp             2     7      15
0   15 /orders/Default.asp       269   4      32      0   1206 ...
```

### User Logins

If your site is mostly unauthenticated anonymous access, then any user login may be suspicious. To see what users have authenticated to the site, try the following query:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT cs-username, Count(*) AS
Hits from ex*.log WHERE cs-username IS NOT NULL GROUP BY cs-username ORDER BY
Hits Desc" -rtp:-1
```

### User-Agents

Sometimes it is possible to identify an attack script by looking at the HTTP User-Agent header sent by the client. You can get a list of non-standard User-Agent strings with this query:

```
C:\WINNT\System32\LogFiles\W3SVC1>logparser "SELECT DISTINCT cs(User-Agent)
FROM ex*.log WHERE TO_LOWERCASE(cs(User-Agent)) NOT LIKE '%mozilla%' AND
TO_LOWERCASE(cs(User-Agent)) NOT LIKE '%opera%' ORDER BY cs(User-Agent)" -
rtp:-1
```

**Closing In**

Following these same patterns, you will eventually close in on the source of the intrusion or identify unknown intrusions. With each query, try to add more criteria and more detail to identify the specific log evidence to identify the attacker or type of attack. LogParser is a very powerful tool, but the real power comes when you learn how to use these and other queries to quickly bring information to your fingertips.

---