This research is supported by Award #1542465:
RET Site: Cyber Security Initiative for Nevada Teachers (CSINT)

Hands-on Activity 1

Hashing:

A hash value is a numeric value of a fixed length that uniquely identifies data. Hash values represent large amounts of data as much smaller numeric values, so they are used with digital signatures. You can sign a hash value more efficiently than signing the larger value.

Here are the most common Hash Algorithms (though there are more)

- MD-5 (Message Digest 5 - 32 characters)
- SHA-1 (Secure Hash Algorithm - 40 characters)
- SHA-256 (Secure Hash Algorithm - 64 characters)

1) Artifacts can be compared using their HASH values.
2) If two (or more) documents, graphics, (artifacts) look similar when viewed , but when Hashed, their HASH values are different, then are they identical?
3) They are not the same file (a single bit difference could cause this).
4) But will the courts “still say” they are “ A reasonable representation of each other?”

Activity:

Open notepad and create the following text files.

File Name	Contents
text1.txt	“the quick brown fox jumped over the lazy dogs back”
text2.txt	“THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK” (typed in all caps)
text3.txt	“the quick brown fox jumped over the lazy dogs back ” (spaces added at end of sentence).
text4.txt	“The quick brown fox jumped over the lazy dogs back” (first letter of the first word in upper case).
Text4 renamed.txt	“The quick brown fox jumped over the lazy dogs back” File renamed and saved text4.txt to test4 renamed.txt

Tools:

There are numerous applications that can calculate hash values. We used HashMyFiles which can be downloaded at https://www.nirsoft.net/utils/hashmyfiles-x64.zip

Hands-on Activity 2

Event Logs:
For introduction and activty see Event Log Slides(eventlogs.pdf)

Tools:
Windows Event Viewer which can be found by clicking the start button and search for event viewer.

Hand-on Activity 3(referenced from https://hshrzd.wordpress.com/2016/03/19/introduction-to-ads-alternate-data-streams/)

Alternate Data Streams

Introduction

In FAT file system – used by old versions of windows – file consisted of 2 elements: attributes and data.

In NTFS it is different – file consists of attributes, security settings, main stream and alternate streams. By default, only the main stream is visible.

Activity

Let's see how it works by creating a sample file: **test.txt**. At this moment it's main stream will be empty. However, we will create an alternte data stream. We can write into it using echo command and simple stream redirection.
Naming convention:
[filename.extension]:[alternate_stream_name]

optionally we can use ::$DATA at the end, i.e:
[filename.extension]:[alternate_stream_name]::$DATA

```
C:\Users\tester>echo This message is saved in the ADS > test.txt:hidden_stream
```

echo This message is saved in the ADS > test.txt:hidden_stream
Let's list the directory and see the newly created file (**test.txt**)

```
C:\Users\tester>dir
 Volume in drive C has no label.
 Volume Serial Number is 448D-3B2B

 Directory of C:\Users\tester

2016-03-18  19:11    <DIR>          .
2016-03-18  19:11    <DIR>          ..
2015-06-05  17:20           225 280 baretail.exe
2015-06-18  21:24    <DIR>          Contacts
2016-03-14  16:50    <DIR>          Desktop
2015-07-22  12:38    <DIR>          Documents
2015-07-22  10:52    <DIR>          Downloads
2015-06-18  21:27    <DIR>          Favorites
2015-06-18  21:24    <DIR>          Links
2015-06-18  21:24    <DIR>          Music
2015-06-18  21:24    <DIR>          Pictures
2015-06-18  21:24    <DIR>          Saved Games
2015-06-18  21:24    <DIR>          Searches
2016-03-18  19:11                 0 test.txt
2015-06-18  21:24    <DIR>          Videos
               2 File(s)        225 280 bytes
              13 Dir(s)  13 883 052 032 bytes free
```

dir

As we can notice, the file length is displayed as 0 bytes. If we try to open this file by some text editor (i.e notepad) we can see that it is empty. Does it really have something inside? Let's confirm:

```
C:\Users\tester>more < test.txt:hidden_stream
This message is saved in the ADS
```

more < test.txt:hidden_stream

Now, finally, our text showed up.

So, how we will find out what are the alternate data streams available in particular files? There are several tools dedicated to reading and editing ADS, but if we don't want to bother about it, we can just use a command **dir**, with an appropriate parameter:

```
C:\Users\tester>dir /?
Displays a list of files and subdirectories in a directory.

DIR [drive:][path][filename] [/A[[:]attributes]] [/B] [/C] [/D] [/L] [/N]
  [/O[[:]sortorder]] [/P] [/Q] [/R] [/S] [/T[[:]timefield]] [/W] [/X] [/4]

  [drive:][path][filename]
              Specifies drive, directory, and/or files to list.

  /A          Displays files with specified attributes.
  attributes   D  Directories                R  Read-only files
               H  Hidden files               A  Files ready for archiving
               S  System files               I  Not content indexed files
               L  Reparse Points             -  Prefix meaning not
  /B          Uses bare format (no heading information or summary).
  /C          Display the thousand separator in file sizes.  This is the
              default.  Use /-C to disable display of separator.
  /D          Same as wide but files are list sorted by column.
  /L          Uses lowercase.
  /N          New long list format where filenames are on the far right.
  /O          List by files in sorted order.
  sortorder    N  By name (alphabetic)       S  By size (smallest first)
               E  By extension (alphabetic)  D  By date/time (oldest first)
               G  Group directories first    -  Prefix to reverse order
  /P          Pauses after each screenful of information.
  /Q          Display the owner of the file.
  /R          Display alternate data streams of the file.
```

dir /R – display alternate data streams of the file

```
C:\Users\tester>dir /r
 Volume in drive C has no label.
 Volume Serial Number is 448D-3B2B

 Directory of C:\Users\tester

2016-03-18  19:11    <DIR>          .
2016-03-18  19:11    <DIR>          ..
2015-06-05  17:20           225 280 baretail.exe
2015-06-18  21:24    <DIR>          Contacts
2016-03-14  16:50    <DIR>          Desktop
2015-07-22  12:38    <DIR>          Documents
2015-07-22  10:52    <DIR>          Downloads
2015-06-18  21:27    <DIR>          Favorites
2015-06-18  21:24    <DIR>          Links
2015-06-18  21:24    <DIR>          Music
2015-06-18  21:24    <DIR>          Pictures
2015-06-18  21:24    <DIR>          Saved Games
2015-06-18  21:24    <DIR>          Searches
2016-03-18  19:29                 0 test.txt
                                 35 test.txt:hidden_stream:$DATA
2015-06-18  21:24    <DIR>          Videos
               2 File(s)        225 280 bytes
              13 Dir(s)  13 883 064 320 bytes free
```

Now we can see the same file, **test.txt**, listed twice: once with a size 0, and then again – with the size 35, with the ADS name added.

We can edit the file in a normal way, and the alternative stream will stay untouched. By the same way we can create several streams.

```
C:\Users\tester>more < test.txt:hidden_stream
This message is saved in the ADS

C:\Users\tester>more < test.txt
This is a main stream
```

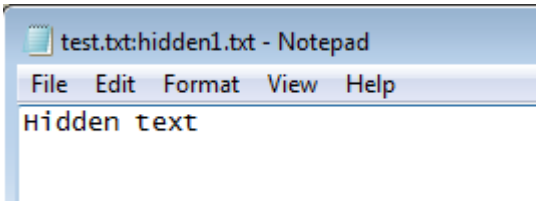
File in file using ADS

Example 1

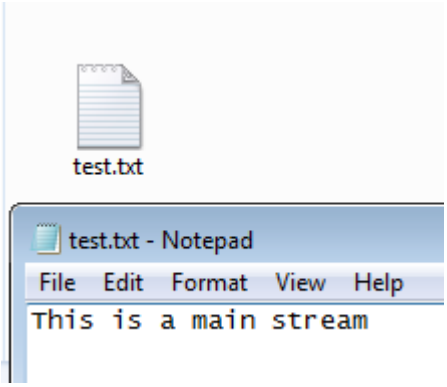
We can also hide another file on the alternate data stream. On the below example – we create a new txt file on another. We can then edit it with typical tools:

```
C:\Users\tester>echo Hidden text > test.txt:hidden1.txt

C:\Users\tester>notepad test.txt:hidden1.txt
```



Yet, opening the file by default way, we can only see it’s main stream:



Example 2

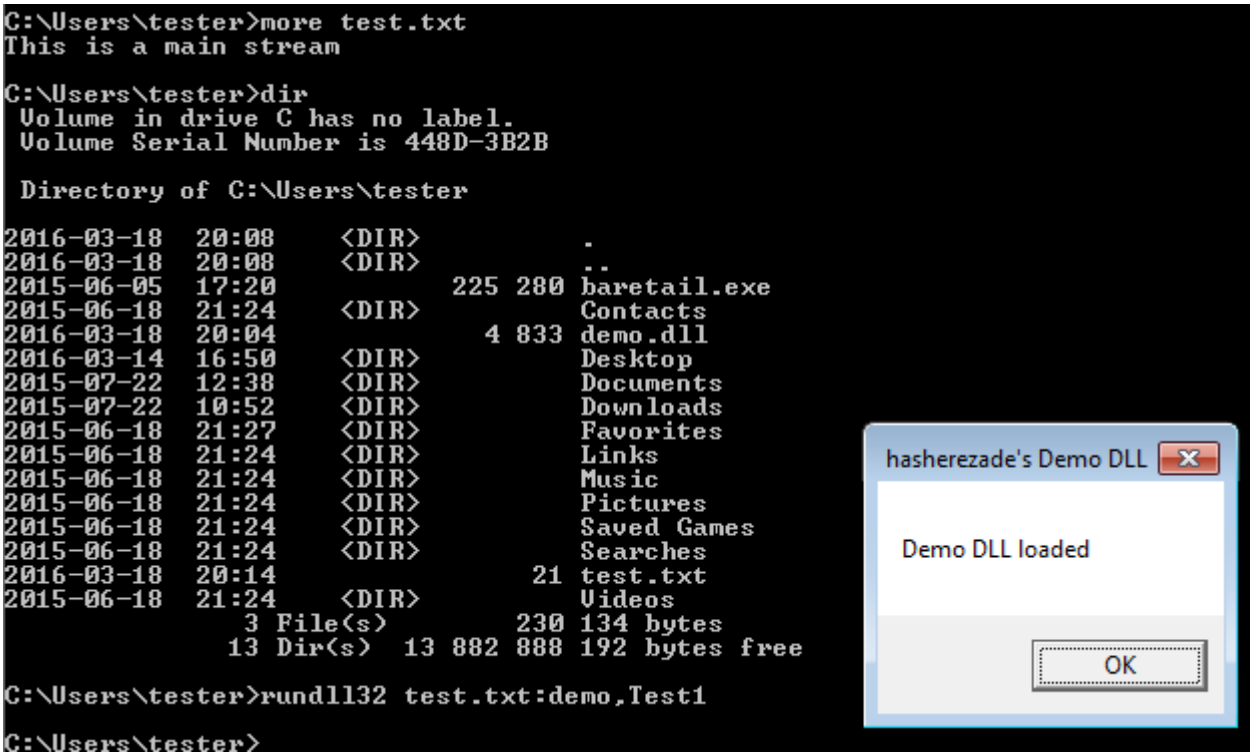
We can also paste an existing file on an alternate data stream, by using a command ***type***

Let's take as an example a demo.dll – it is a 32bit Portable Executable, exporting one function: *Test1*. We will place it in the alternate stream of **test.txt**

```
C:\Users\tester>type demo.dll > test.txt:demo
```

type demo.dll > test.txt:demo

Maybe the alternate stream it is hard to notice – but running it is still very easy:



rundll32 test.txt:demo,Test1

Example 3

Exactly the same can be done with (malicious) macros:

type malware.vbs > readme.txt:malware.vbs

Wscript readme.txt:malware.vbs

Zone.Identifier

One of the legitimate usages of alternate data streams is Zone.Identifier. It is a feature used to identify the file origin. In case if the file comes from some untrusted source, i.e. have been downloaded from the internet, Windows displays a security warning before it can be run.

There are several variants of Zone.Identifier value:

0 My Computer
1 Local Intranet Zone
2 Trusted sites Zone
3 Internet Zone
4 Restricted Sites Zone

file.exe:Zone.Identifier

Sample content of Zone.Identifier of the file downloaded from the internet:

[ZoneTransfer]
ZoneId=3

Malware downloaders may edit Zone.Identifier of the downloaded file, in order to make it run without displaying alert.

ADS and PowerShell

PowerShell comes with a built-in feature to read ADS. There are several commands that can be used to read and edit them:

- Get-Item
- Set-Item
- Remove-Item
- Add-Content
- Get-Content
- Set-Content

Examples

Listing all the streams of a file:

Get-Item -Path [filename] -Stream *

Adding hidden message into ADS:

Add-Content -Path [filename] -Value [my hidden message] -Stream [new_stream]

Cheatsheet

Creating ADS from commandline:

echo This is a hidden message > testfile.txt:hidden_stream

Displaying files with their alternative data streams:

dir /r

Displaying stream of a file:

more < testfile.txt:hidden_stream::$DATA

Tools:
Windows command prompt, notepad, and powershell.

Hands-on Activity 4

Log Parsing
For log parsing I just have the students work through the tutorial located at https://www.linkedin.com/pulse/7-log-analysis-techniques-digital-forensics-incident-sverdlov/

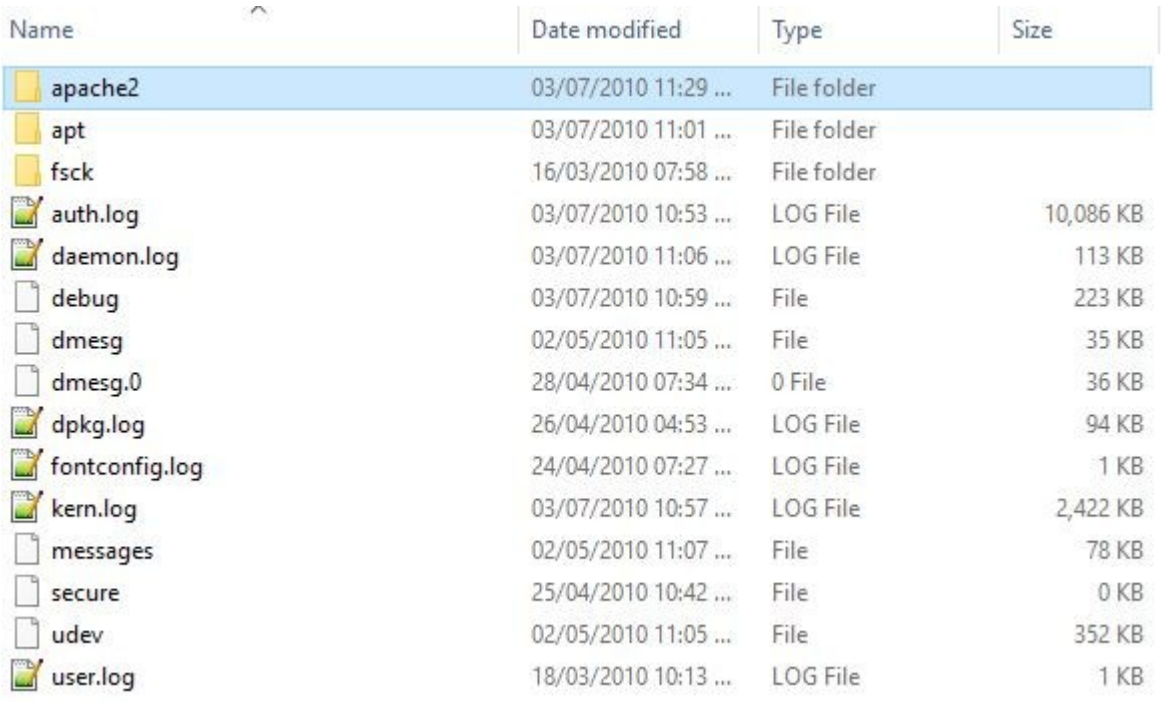
I figure that it explains the process fairly well. Below I have included the method utilizing both Mandiant Highlighter and Notepad++(Note Mandiant does require registration to download, but I do believe it is still free).

Logs can be huge – and analyzing a 500MB or even a 1-2 GB log file can quickly get daunting and tiring. Going through such a file to find a single instance of the right command which got your server compromised could be an all-nighter. Let's make it 15-45 min., shall we?

Let's first set up our task:

Head over to https://honeynet.org/challenges/2010_5_log_mysteries and download some logs. Specifically, the file located at https://honeynet.org/sites/default/files/files/sanitized_log.zip . Following the challenge, we will complete it in several programs utilizing a few useful filtering techniques. If you disagree with me or have suggestions, feel free to post them in the contact form on my website – I will be happy to amend my article or reply with my thoughts.

Extract the archive and look around.



Name	Date modified	Type	Size
apache2	03/07/2010 11:29 ...	File folder	
apt	03/07/2010 11:01 ...	File folder	
fsck	16/03/2010 07:58 ...	File folder	
auth.log	03/07/2010 10:53 ...	LOG File	10,086 KB
daemon.log	03/07/2010 11:06 ...	LOG File	113 KB
debug	03/07/2010 10:59 ...	File	223 KB
dmesg	02/05/2010 11:05 ...	File	35 KB
dmesg.0	28/04/2010 07:34 ...	0 File	36 KB
dpkg.log	26/04/2010 04:53 ...	LOG File	94 KB
fontconfig.log	24/04/2010 07:27 ...	LOG File	1 KB
kern.log	03/07/2010 10:57 ...	LOG File	2,422 KB
messages	02/05/2010 11:07 ...	File	78 KB
secure	25/04/2010 10:42 ...	File	0 KB
udev	02/05/2010 11:05 ...	File	352 KB
user.log	18/03/2010 10:13 ...	LOG File	1 KB

From the list we could see immediately the log which could interest us – auth.log

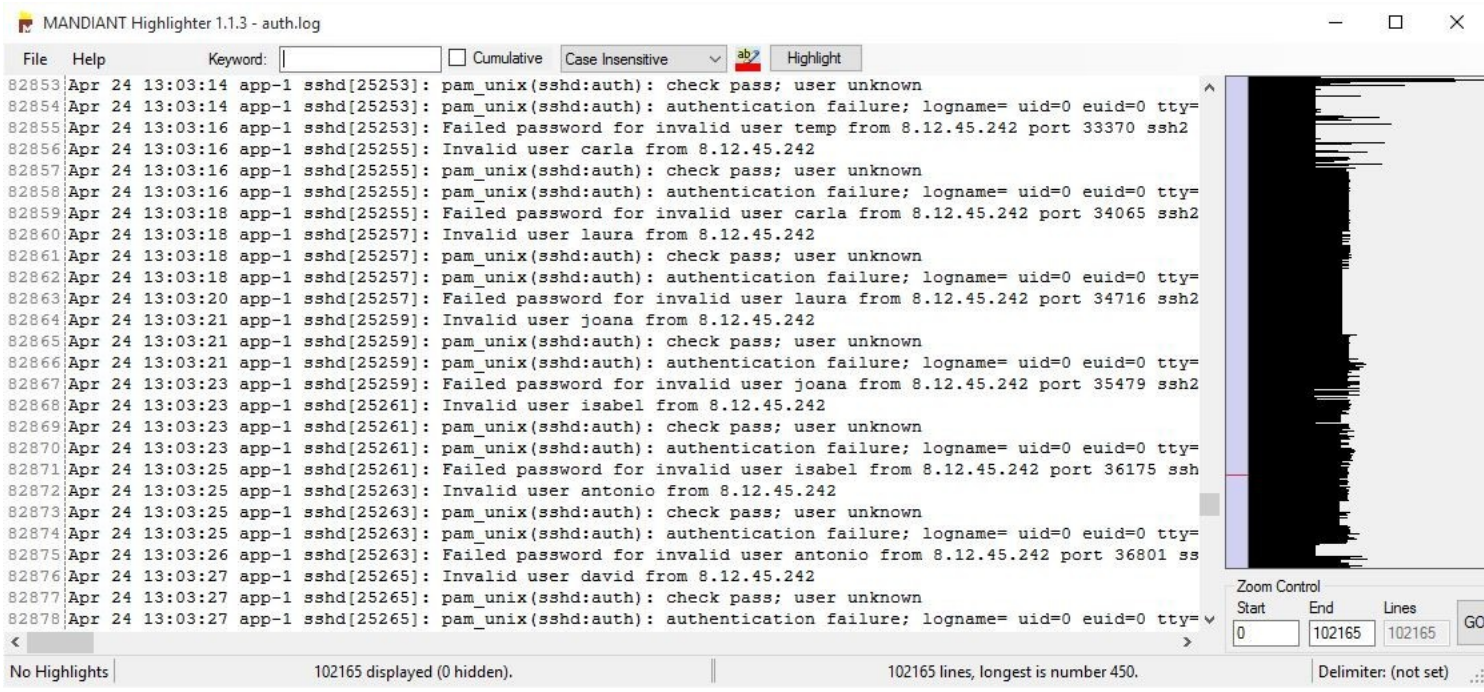
Our objective will be to answer the challenge's questions with the programs below.

For our exercise, we will need:

1. Notepad++
2. LogExpert

1. Quick Filtering with Mandiant Highlighter

I hope Mandiant (FireEye) keeps this tool, if not updated (gosh, they have not updated it in ages!), at least online for long enough for a sane developer to develop something modern and at least as useful and much more stable. From the frequent crashes I've experienced with it I would only recommend it for files less than 100MB in size. Anything bigger and some complex tasks simply kill the program. Yes, it can open huge files – but opening is one thing, complex filtering is another.



On opening the auth.log file with Highlighter we see it contains **102165 lines**. Not realistic for reading the whole thing, so let's get rid of the lines we don't want to see.

We accomplish this by glancing over the file, scrolling from top to bottom and noting any lines which are frequent and useless at the same time. For example, we would be interested statistically in what usernames were attempted to login to ssh, but if they were invalid they pose no interest to us. So we could search for "**Invalid user**", select the 2 words, right-click and select "Remove" which would remove all lines containing them. This removes roughly 13 000 lines, or more than 10%. We can do the same for "**Failed password for invalid user**", " authentication failure", (so far 50% of the log file has been filtered out), "user unknown", "check pass; user unknow", "**Failed password for root from**", " Failed password for", " session closed for user" (because we might not be interested in logouts as much as in logins, right?). Even so, we see a line containing "session opened for user root" – and we might be more interested in "Accepted password", instead – so we remove even the session opened lines.

One more string to remove is "POSSIBLE BREAK-IN ATTEMPT!" – this alert sounds scary but is not very helpful in identifying actual breaches, unless we see a successful login attempt from the same IP later on (which is a part of a deeper statistical analysis).

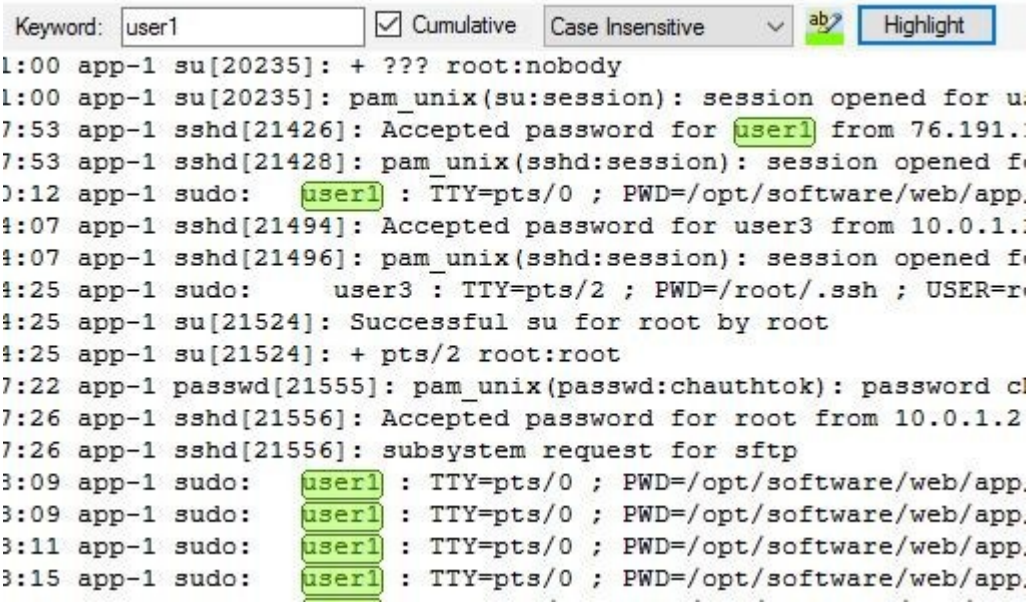
2. We are left with a whopping 1747 lines!

All that in just a few seconds of filtering. Neat, especially knowing that we can reclaim any lines removed from the GUI (right click, Line operations – reclaim lines previously removed).

The remaining line allow us to build a timeline of events and the commands used to compromise the server and answer all questions in the challenge above.

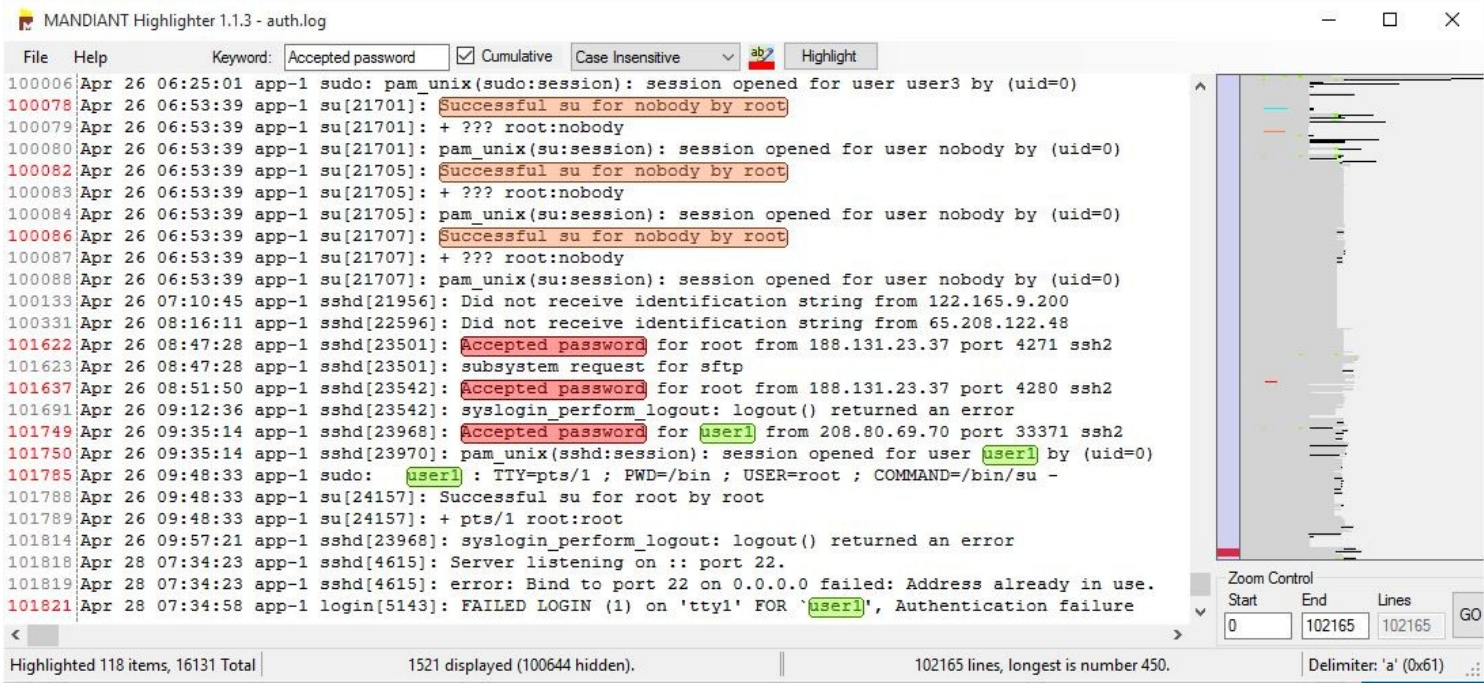
That is with just one function of Highlighter – "Remove"! Let's not forget we can highlight different things with different colors to make our analysis easier:

For example, we see a lot of instances of commands and actions from “user1”. We go to Keyword, enter user 1, select “cumulative”, “Case insensitive”, change the color to a distinctive one, press “Highlight”, voila!



We can do the same for “Successful su for nobody by root”, “Successful su for www-data by root”, “Successful su for www-data by root” (3 hits), “Accepted password” (with a RED color and we found 118 hits), rinse and repeat for all strings which pose interest and would help us solve the puzzle.

The final window should look something like:



Which can be used in presentations to management and reports and is much better than simple text excerpts.

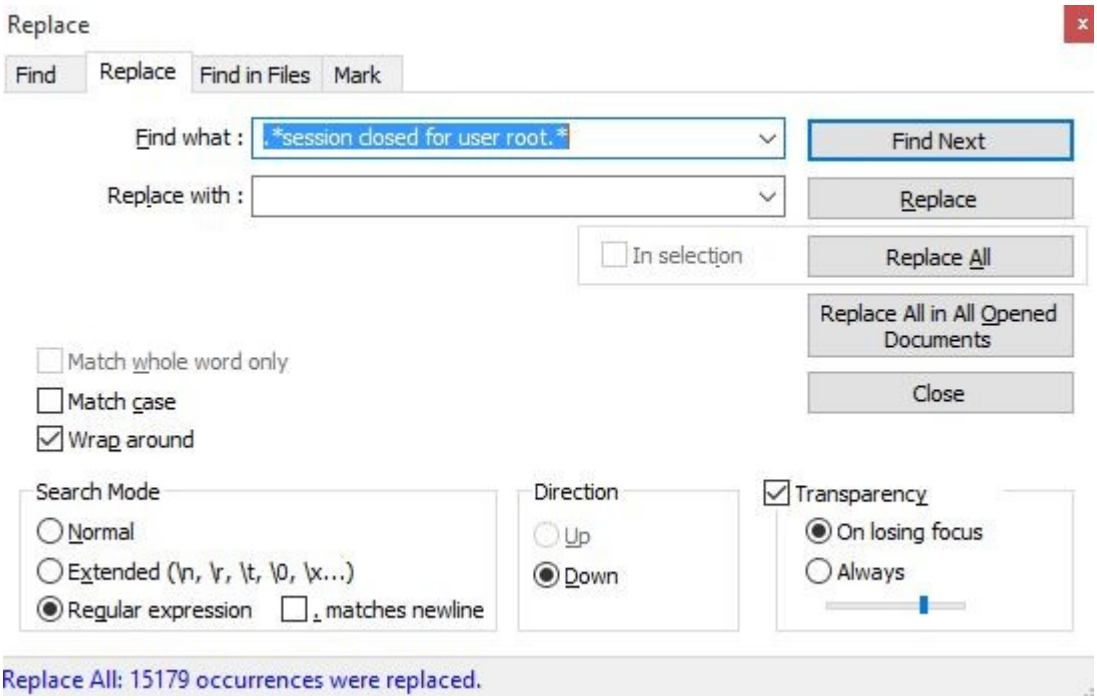
You can also experiment with selecting a string (for example, a user or an IP address) and selecting “Show only” – which filters out everything except the selected string. If you have one suspect this allows to quickly narrow your view to just their actions, temporarily.

3. Now let’s repeat the filtering that with Notepad++

Why notepad++? Because analyzing logs with highlighter is easy, but it often breaks with exceptions and errors and with very large files tends to die completely. Some things are better done in a more stable program and there are a few very useful plugins for Notepad++ for log analysis.

Let’s open the same file in this program and see if we could repeat the same filtering with it.

Select a string you wish to filter out (altogether with the whole line it is on), press **Ctrl+H:**



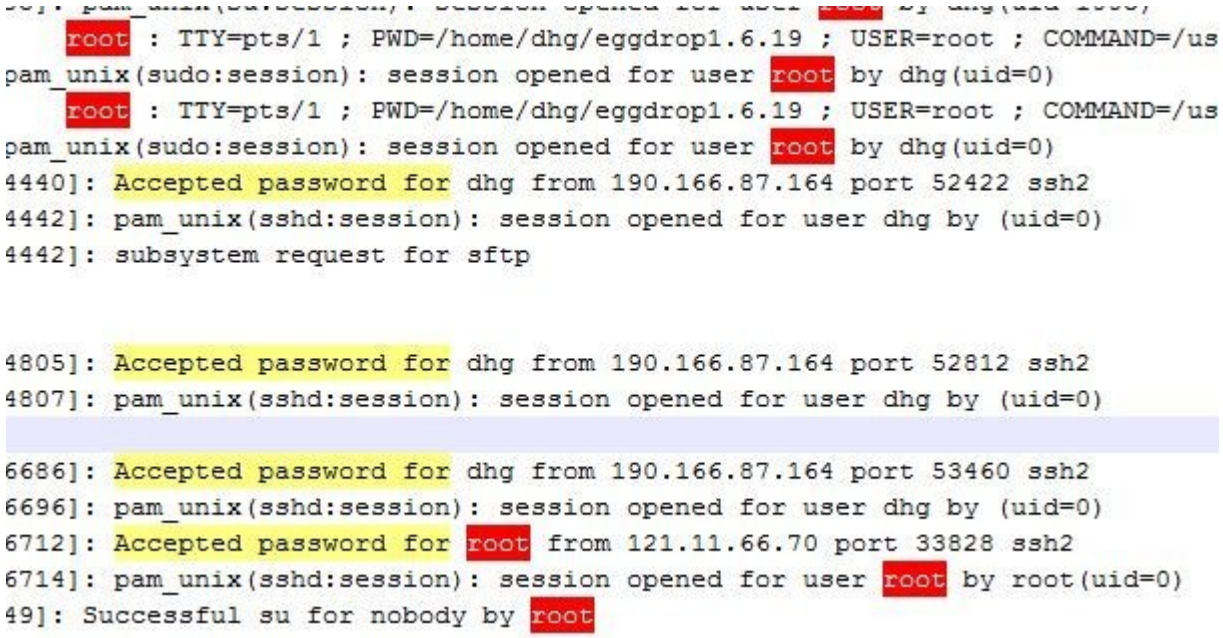
add .* before the string and .* after the string, so it would look like **.*string.***, then select “Regular expression” and click Replace All. This will remove all lines containing your string, leaving an empty line instead.

These blank lines can easily be removed using a Notepad++ plugin (if missing, install it with Plugin manager): TextFX -> TextFX Edit -> **Delete Blank Lines** (select all text first).

4. Notepad++ color highlights

Next, we will use the guideline from https://dzone.com/articles/tip-using-notepad-read-log to create the same colorful highlights as in Highlighter, but Better! Because we can have the highlights automatically done for us, depending on keywords, every time we open the same type of log file, without having to re-define them and re-highlight again.

The result:



The technique is especially useful for more complex logs (for example, when analyzing an MFT table from a Windows operating system) and searching for multiple IOCs (indicators of compromise) – highlighting key values on file opening and selecting your custom language (you can define separate languages per log / file type) saves a ton of time.

Tools:
Mandiant Highlighter and Notepadd++

This research is supported by Award #1542465:
RET Site: Cyber Security Initiative for Nevada Teachers (CSINT)