Command-Line Log Analysis

JOEMCCRAY PRO FEB 20TH, 2016 (EDITED) 6,732 NEVER

text 25.10 KB raw download report diff

```
#####################################
# Pentester Academy Log Analysis #
#####################################

I'm doing this set of videos for my good friend Vivek Ramachandran at SecurityTube.net/PentesterAcademy.com





##########
# VMWare #
##########
- For this workshop you'll need the latest version of VMWare Workstation (Windows), Fusion (Mac), or Player.

- Although you can get the VM to run in VirtualBox, I will not be supporting this configuration for this class.

VM for these labs
-----------------
https://s3.amazonaws.com/infosecaddictsvirtualmachines/Win7x64.zip
        username: workshop
        password: password




##############################################
# Log Analysis with Linux command-line tools #
##############################################
The following command line executables are found in the Mac as well as most Linux Distributions.

cat –  prints the content of a file in the terminal window
grep – searches and filters based on patterns
awk –  can sort each row into fields and display only what is needed
sed –  performs find and replace functions
sort – arranges output in an order
uniq – compares adjacent lines and can report, filter or provide a count of duplicates



###############
# Apache Logs #
###############
```

```

Reference:
http://www.the-art-of-web.com/system/logs/

wget https://s3.amazonaws.com/SecureNinja/Python/access_log


You want to list all user agents ordered by the number of times they appear (descending order):

awk -F\" '{print $6}' access_log | sort | uniq -c | sort -fr



Using the default separator which is any white-space (spaces or tabs) we get the following:

awk '{print $1}' access_log         # ip address (%h)
awk '{print $2}' access_log         # RFC 1413 identity (%l)
awk '{print $3}' access_log         # userid (%u)
awk '{print $4,5}' access_log       # date/time (%t)
awk '{print $9}' access_log         # status code (%>s)
awk '{print $10}' access_log        # size (%b)

You might notice that we've missed out some items. To get to them we need to set the delimiter to the " character which changes the way the lines are 'exploded' and allows the following:

awk -F\" '{print $2}' access_log    # request line (%r)
awk -F\" '{print $4}' access_log    # referer
awk -F\" '{print $6}' access_log    # user agent


awk -F\" '{print $6}' access_log \
  | sed 's/(\([^;]\+; [^;]\+\)[^)]*)/(\1)/' \
  | sort | uniq -c | sort -fr


The next step is to start filtering the output so you can narrow down on a certain page or referer. Would you like to know which pages Google has been requesting from your site?

awk -F\" '($6 ~ /Googlebot/){print $2}' access_log | awk '{print $2}'
Or who's been looking at your guestbook?

awk -F\" '($2 ~ /guestbook\.html/){print $6}' access_log


Reference:
https://blog.nexcess.net/2011/01/21/one-liners-for-apache-log-files/

# top 20 URLs from the last 5000 hits
tail -5000 ./access_log | awk '{print $7}' | sort | uniq -c | sort -rn | head -20
tail -5000 ./access_log | awk '{freq[$7]++} END {for (x in freq) {print freq[x], x}}' | sort -rn | head -20

# top 20 URLS excluding POST data from the last 5000 hits
tail -5000 ./access_log | awk -F"[ ?]" '{print $7}' | sort | uniq -c | sort -rn | head -20
tail -5000 ./access_log | awk -F"[ ?]" '{freq[$7]++} END {for (x in freq) {print freq[x], x}}' | sort -rn | head -20

# top 20 IPs from the last 5000 hits
```

```
tail -5000 ./access_log | awk '{print $1}' | sort | uniq -c | sort -rn | head -20
tail -5000 ./access_log | awk '{freq[$1]++} END {for (x in freq) {print freq[x], x}}' | sort -rn | head -20

# top 20 URLs requested from a certain ip from the last 5000 hits
IP=1.2.3.4; tail -5000 ./access_log | grep $IP | awk '{print $7}' | sort | uniq -c | sort -rn | head -20
IP=1.2.3.4; tail -5000 ./access_log | awk -v ip=$IP ' $1 ~ ip {freq[$7]++} END {for (x in freq) {print freq[x], x}}' | sort -rn | head -20

# top 20 URLS requested from a certain ip excluding, excluding POST data, from the last 5000 hits
IP=1.2.3.4; tail -5000 ./access_log | fgrep $IP | awk -F "[ ?]" '{print $7}' | sort | uniq -c | sort -rn | head -20
IP=1.2.3.4; tail -5000 ./access_log | awk -F"[ ?]" -v ip=$IP ' $1 ~ ip {freq[$7]++} END {for (x in freq) {print freq[x], x}}' | sort -rn | head -20

# top 20 referrers from the last 5000 hits
tail -5000 ./access_log | awk '{print $11}' | tr -d '"' | sort | uniq -c | sort -rn | head -20
tail -5000 ./access_log | awk '{freq[$11]++} END {for (x in freq) {print freq[x], x}}' | tr -d '"' | sort -rn | head -20

# top 20 user agents from the last 5000 hits
tail -5000 ./access_log | cut -d\  -f12- | sort | uniq -c | sort -rn | head -20

# sum of data (in MB) transferred in the last 5000 hits
tail -5000 ./access_log | awk '{sum+=$10} END {print sum/1048576}'


##############
# Cisco Logs #
##############

```

```
wget https://s3.amazonaws.com/StrategicSec-Files/LogAnalysis/cisco.log


AWK Basics
----------
To quickly demonstrate the print feature in awk, we can instruct it to show only the 5th word of each line. Here we will print $5. Only the last 4 lines are being shown for brevity.

cat cisco.log | awk '{print $5}' | tail -n 4




Looking at a large file would still produce a large amount of output. A more useful thing to do might be to output every entry found in “$5”, group them together, count them, then sort them from the greatest to least number of occurrences. This can be done by piping the output through “sort“, using “uniq -c” to count the like entries, then using “sort -rn” to sort it in reverse order.

cat cisco.log | awk '{print $5}'| sort | uniq -c | sort -rn




While that’s sort of cool, it is obvious that we have some garbage in our output. Evidently we have a few lines that aren’t conforming to the output we expect to see in $5. We can insert grep to filter the file prior to feeding it to awk. This insures that we are at least looking at lines of text that contain “facility-level-mnemonic”.

cat cisco.log | grep %[a-zA-Z]*-[0-9]-[a-zA-Z]* | awk '{print $5}' | sort | uniq -c | sort -rn

```

```
Now that the output is cleaned up a bit, it is a good time to investigate some of the entries that appear most often. One way to see all occurrences is to use grep.

cat cisco.log | grep %LINEPROTO-5-UPDOWN:

cat cisco.log | grep %LINEPROTO-5-UPDOWN:| awk '{print $10}' | sort | uniq -c | sort -rn

cat cisco.log | grep %LINEPROTO-5-UPDOWN:| sed 's/,//g' | awk '{print $10}' | sort | uniq -c | sort -rn

cat cisco.log | grep %LINEPROTO-5-UPDOWN:| sed 's/,//g' | awk '{print $10 " changed to " $14}' | sort | uniq -c | sort -rn




##################################
# Using Python for log analysis #
##################################




############################################
# Python Basics Lesson 1: Simple Printing #
############################################
```

```

>>> print 1

>>> print hello

>>> print "hello"

>>> print "Today we are learning Python."



####################################################
# Python Basics Lesson 2: Simple Numbers and Math #
####################################################

>>> 2+2

>>> 6-3

>>> 18/7

>>> 18.0/7

>>> 18.0/7.0

>>> 18/7

>>> 9%4

>>> 8%4

>>> 8.75%.5

>>> 6.*7

>>> 6*6*6

>>> 6**3

>>> 5**12

>>> -5**4






#####################################
# Python Basics Lesson 3: Variables #
#####################################

>>> x=18

>>> x+15

>>> x**3

>>> y=54

>>> x+y

>>> age=input("Enter number here: ")
        43

>>> age+32

>>> age**3

>>> fname = raw_input("Enter your first name: ")

>>> lname = raw_input("Enter your first name: ")

>>> fname = raw_input("Enter your name: ")
Enter your name: Joe

>>> lname = raw_input("Enter your name: ")
Enter your name: McCray

>>> print fname
Joe

>>> print lname
McCray
```

```


>>> print fname lname

>>> print fname+lname
JoeMcCray



NOTE:
Use "input() for integers and expressions, and use raw_input() when you are dealing with strings.





#################################################
# Python Basics Lesson 4: Modules and Functions #
#################################################

>>> 5**4

>>> pow(5,4)

>>> abs(-18)

>>> abs(5)

>>> floor(18.7)

>>> import math

>>> math.floor(18.7)

>>> math.sqrt(81)

>>> joe = math.sqrt

>>> joe(9)

>>> joe=math.floor

>>> joe(19.8)









###################################
# Python Basics Lesson 5: Strings #
###################################

>>> "XSS"
>>> 'SQLi'

>>> "Joe's a python lover"

>>> 'Joe\'s a python lover'

>>> "Joe said \"InfoSec is fun\" to me"

>>> a = "Joe"

>>> b = "McCray"

>>> a, b

>>> a+b








########################################
# Python Basics Lesson 6: More Strings #
########################################

```

```
>>> num = 10

>>> num + 2

>>> "The number of open ports found on this system is " + num

>>> num = str(18)

>>> "There are " + num + " vulnerabilities found in this environment."

>>> num2 = 46

>>> "As of 08/20/2012, the number of states that enacted the Security Breach Notification Law is " + `num2`



NOTE:
Use "input() for integers and expressions, and use raw_input() when you are dealing with strings.







###############################################
# Python Basics Lesson 7: Sequences and Lists #
###############################################
```

```
>>> attacks = ['Stack Overflow', 'Heap Overflow', 'Integer Overflow', 'SQL Injection', 'Cross-Site Scripting', 'Remote File Include']

>>> attacks
['Stack Overflow', 'Heap Overflow', 'Integer Overflow', 'SQL Injection', 'Cross-Site Scripting', 'Remote File Include']

>>> attacks[3]
'SQL Injection'

>>> attacks[-2]
'Cross-Site Scripting'






########################################
# Python Basics Level 8: If Statement #
########################################
>>> attack="SQLI"
>>> if attack=="SQLI":
        print 'The attacker is using SQLI'

>>> attack="XSS"
>>> if attack=="SQLI":
        print 'The attacker is using SQLI'

```

```

##############################
# Reference Videos To Watch #
##############################
Here is your first set of youtube videos that I'd like for you to watch:
https://www.youtube.com/playlist?list=PLEA1FEF17E1E5C0DA (watch videos 1-10)





#####################################
# Lesson 9: Intro to Log Analysis #
#####################################

Login to your StrategicSec Ubuntu machine. You can download the VM from the following link:

https://s3.amazonaws.com/StrategicSec-VMs/Strategicsec-Ubuntu-VPN-163.zip
        username: strategicsec
        password: strategicsec

Then execute the following commands:
---------------------------------------------------------------------------------------------------------


wget https://s3.amazonaws.com/SecureNinja/Python/access_log


```

```
cat access_log | grep 141.101.80.188

cat access_log | grep 141.101.80.187

cat access_log | grep 108.162.216.204

cat access_log | grep 173.245.53.160

---------------------------------------------------------

Google the following terms:
        - Python read file
        - Python read line
        - Python read from file




########################################################
# Lesson 10: Use Python to read in a file line by line #
########################################################


Reference:
http://cmdlinetips.com/2011/08/three-ways-to-read-a-text-file-line-by-line-in-python/



```

```



Let's have some fun.....



>>> f = open('access_log', "r")

>>> lines = f.readlines()

>>> print lines

>>> lines[0]

>>> lines[10]

>>> lines[50]

>>> lines[1000]

>>> lines[5000]

>>> lines[10000]

>>> print len(lines)



```

```
---------------------------------------------------------
vi logread1.py


## Open the file with read only permit
f = open('access_log', "r")

## use readlines to read all lines in the file
## The variable "lines" is a list containing all lines
lines = f.readlines()

print lines


## close the file after reading the lines.
f.close()

---------------------------------------------------------


Google the following:
        - python difference between readlines and readline
        - python readlines and readline





#################################
# Lesson 11: A quick challenge #
#################################

Can you write an if/then statement that looks for this IP and print "Found it"?


141.101.81.187






---------------------------------------------------------
Hint 1: Use Python to look for a value in a list

Reference:
http://www.wellho.net/mouth/1789_Looking-for-a-value-in-a-list-Python.html



```

```

---------------------------------------------------------
Hint 2: Use Python to prompt for user input

Reference:
http://www.cyberciti.biz/faq/python-raw_input-examples/




---------------------------------------------------------
Hint 3: Use Python to search for a string in a list

Reference:
http://stackoverflow.com/questions/4843158/check-if-a-python-list-item-contains-a-string-inside-another-string





Here is my solution:
-------------------
$ python
>>> f = open('access_log', "r")
>>> lines = f.readlines()
>>> ip = '141.101.81.187'
>>> for string in lines:
...     if ip in string:
```

```
...             print(string)




Here is one student's solution - can you please explain each line of this code to me?
-------------------------------------------------------------------------------------
#!/usr/bin/python

f = open('access_log')

strUsrinput = raw_input("Enter IP Address: ")

for line in iter(f):
    ip = line.split(" - ")[0]
    if ip == strUsrinput:
        print line

f.close()




-------------------------------

Working with another student after class we came up with another solution:

#!/usr/bin/env python
```

```


# This line opens the log file
f=open('access_log',"r")

# This line takes each line in the log file and stores it as an element in the list
lines = f.readlines()


# This lines stores the IP that the user types as a var called userinput
userinput = raw_input("Enter the IP you want to search for: ")



# This combination for loop and nested if statement looks for the IP in the list called lines and prints the entire line if found.
for ip in lines:
    if ip.find(userinput) != -1:
        print ip



##################################################
# Lesson 12: Look for web attacks in a log file #
##################################################

In this lab we will be looking at the scan_log.py script and it will scan the server log to find out common hack attempts within your web server log.
Supported attacks:
```

```
1.          SQL Injection
2.          Local File Inclusion
3.          Remote File Inclusion
4.          Cross-Site Scripting



wget https://s3.amazonaws.com/SecureNinja/Python/scan_log.py

The usage for scan_log.py is simple.  You feed it an apache log file.

cat scan_log.py | less                  (use your up/down arrow keys to look through the file)





################################
# Log Analysis with Powershell #
################################

VM for these labs
-----------------
https://s3.amazonaws.com/infosecaddictsvirtualmachines/Win7x64.zip
        username: workshop
        password: password


```

```
You can do the updates in the Win7 VM (yes, it is a lot of updates).

You'll need to create directory in the Win7 VM called "c:\ps"

#####################
# Powershell Basics #
#####################

PowerShell is Microsoft’s new scripting language that has been built in since the release Vista.

PowerShell file extension end in .ps1 .

An important note is that you cannot double click on a PowerShell script to execute it.

To open a PowerShell command prompt either hit Windows Key + R and type in PowerShell or Start -> All Programs -> Accessories -> Windows PowerShell -> Windows PowerShell.

dir
cd
ls
cd c:\


To obtain a list of cmdlets, use the Get-Command cmdlet

Get-Command


```

```

You can use the Get-Alias cmdlet to see a full list of aliased commands.

Get-Alias




Don't worry you won't blow up your machine with Powershell
Get-Process | stop-process                              What will this command do?
Get-Process | stop-process -whatif


To get help with a cmdlet, use the Get-Help cmdlet along with the cmdlet you want information about.

Get-Help Get-Command

Get-Help Get-Service –online

Get-Service -Name TermService, Spooler

Get-Service –N BITS

Start-Transcript

PowerShell variables begin with the $ symbol. First lets create a variable

$serv = Get-Service –N Spooler

```

```
To see the value of a variable you can just call it in the terminal.

$serv

$serv.gettype().fullname


Get-Member is another extremely useful cmdlet that will enumerate the available methods and properties of an object. You can pipe the object to Get-Member or pass it in

$serv | Get-Member

Get-Member -InputObject $serv





Let’s use a method and a property with our object.

$serv.Status
$serv.Stop()
$serv.Refresh()
$serv.Status
$serv.Start()
$serv.Refresh()
$serv.Status

```

```



Methods can return properties and properties can have sub properties. You can chain them together by appending them to the first call.



##############################
# Simple Event Log Analysis #
##############################

Step 1: Dump the event logs
---------------------------
The first thing to do is to dump them into a format that facilitates later processing with Windows PowerShell.

To dump the event log, you can use the Get-EventLog and the Exportto-Clixml cmdlets if you are working with a traditional event log such as the Security, Application, or System event logs.
If you need to work with one of the trace logs, use the Get-WinEvent and the ExportTo-Clixml cmdlets.

Get-EventLog -LogName application | Export-Clixml Applog.xml

type .\Applog.xml

$logs = "system","application","security"

The % symbol is an alias for the Foreach-Object cmdlet. It is often used when working interactively from the Windows PowerShell console
```

```

$logs | % { get-eventlog -LogName $_ | Export-Clixml "$_.xml" }



Step 2: Import the event log of interest
----------------------------------------
To parse the event logs, use the Import-Clixml cmdlet to read the stored XML files.
Store the results in a variable.
Let's take a look at the commandlets Where-Object, Group-Object, and Select-Object.

The following two commands first read the exported security log contents into a variable named $seclog, and then the five oldest entries are obtained.

$seclog = Import-Clixml security.xml

$seclog | select -Last 5


Cool trick from one of our students named Adam. This command allows you to look at the logs for the last 24 hours:

Get-EventLog Application -After (Get-Date).AddDays(-1)

You can use '-after' and '-before' to filter date ranges

One thing you must keep in mind is that once you export the security log to XML, it is no longer protected by anything more than the NFTS and share permissions that are assigned to the location where you store everything.
By default, an ordinary user does not have permission to read the security log.
```

```
Step 3: Drill into a specific entry
-----------------------------------
To view the entire contents of a specific event log entry, choose that entry, send the results to the Format-List cmdlet, and choose all of the properties.


$seclog | select -first 1 | fl *

The message property contains the SID, account name, user domain, and privileges that are assigned for the new login.


($seclog | select -first 1).message

(($seclog | select -first 1).message).gettype()



In the *nix world you often want a count of something (wc -l).
How often is the SeSecurityPrivilege privilege mentioned in the message property?
To obtain this information, pipe the contents of the security log to a Where-Object to filter the events, and then send the results to the Measure-Object cmdlet to determine the number of events:

$seclog | ? { $_.message -match 'SeSecurityPrivilege'} | measure

If you want to ensure that only event log entries return that contain SeSecurityPrivilege in their text, use Group-Object to gather the matches by the EventID property.
```

```
$seclog | ? { $_.message -match 'SeSecurityPrivilege'} | group eventid

Because importing the event log into a variable from the stored XML results in a collection of event log entries, it means that the count property is also present.
Use the count property to determine the total number of entries in the event log.

$seclog.Count






#############################
# Simple Log File Analysis #
#############################


You'll need to create the directory c:\ps and download sample iss log http://pastebin.com/raw.php?i=LBn64cyA


mkdir c:\ps
cd c:\ps
(new-object System.Net.WebClient).DownloadFile("http://pastebin.com/raw.php?i=LBn64cyA", "c:\ps\u_ex1104.log")


```

```
##############################################
# Intrusion Analysis Using Windows PowerShell #
##############################################

Download sample file http://pastebin.com/raw.php?i=ysnhXxTV into the c:\ps directory





(new-object System.Net.WebClient).DownloadFile("http://pastebin.com/raw.php?i=ysnhXxTV", "c:\ps\CiscoLogFileExamples.txt")

Select-String 192.168.208.63 .\CiscoLogFileExamples.txt




The Select-String cmdlet searches for text and text patterns in input strings and files. You can use it like Grep in UNIX and Findstr in Windows.

Select-String 192.168.208.63 .\CiscoLogFileExamples.txt | select line

```

To see how many connections are made when analyzing a single host, the output from that can be piped to another command: Measure-Object.

```
Select-String 192.168.208.63 .\CiscoLogFileExamples.txt | select line | Measure-Object
```

To select all IP addresses in the file expand the matches property, select the value, get unique values and measure the output.

```
Select-String “\b(?:\d{1,3}\.){3}\d{1,3}\b” .\CiscoLogFileExamples.txt | select -ExpandProperty matches | select -ExpandProperty value | Sort-Object -Unique | Measure-Object
```

Removing Measure-Object shows all the individual IPs instead of just the count of the IP addresses. The Measure-Object command counts the IP addresses.

```
Select-String “\b(?:\d{1,3}\.){3}\d{1,3}\b” .\CiscoLogFileExamples.txt | select -ExpandProperty matches | select -ExpandProperty value | Sort-Object -Unique
```

In order to determine which IP addresses have the most communication the last commands are removed to determine the value of the matches. Then the group command is issued on the piped output to group all the IP addresses (value), and then sort the objects by using the alias for Sort-Object: sort count –des.
This sorts the IP addresses in a descending pattern as well as count and deliver the output to the shell.

```

Select-String “\b(?:\d{1,3}\.){3}\d{1,3}\b” .\CiscoLogFileExamples.txt | select -ExpandProperty matches | select value | group value | sort count -des




This will get the setting for logs in the windows firewall which should be enabled in GPO policy for analysis.
The command shows that the Firewall log is at:
%systemroot%\system32\LogFiles\Firewall\pfirewall.log, in order to open the file PowerShell will need to be run with administrative privileges.


First step is to get the above command into a variable using script logic.
Thankfully PowerShell has a built-in integrated scripting environment, PowerShell.ise.

netsh advfirewall show allprofiles | Select-String FileName | select -ExpandProperty line | Select-String “%systemroot%.+\.log" | select -ExpandProperty matches | select -ExpandProperty value | sort –uniq


##############################################
# Parsing Log files using windows PowerShell #
##############################################

Download the sample IIS log http://pastebin.com/LBn64cyA


(new-object System.Net.WebClient).DownloadFile("http://pastebin.com/raw.php?i=LBn64cyA", "c:\ps\u_ex1104.log")
```

```
Get-Content ".\*log" | ? { ($_ | Select-String "WebDAV")}



The above command would give us all the WebDAV requests.

To filter this to a particular user name, use the below command:

Get-Content ".\*log" | ? { ($_ | Select-String "WebDAV") -and ($_ | Select-String "OPTIONS")}



Some more options that will be more commonly required :

For Outlook Web Access : Replace WebDAV with OWA

For EAS : Replace WebDAV with Microsoft-server-activesync

For ECP : Replace WebDAV with ECP



To find out the count of the EWS request we can go ahead and run the below command

(Get-Content ".\*log" | ? { ($_ | Select-String "WebDAV") -and ($_ | Select-String "Useralias")}).count


```

```
Explain to me how this script works.
```

RAW Paste Data

```
####################################
# Pentester Academy Log Analysis #
####################################

I'm doing this set of videos for my good friend Vivek Ramachandran at SecurityTube.net/PentesterAcademy.com
```