Module 10 HTTP Analysis

Download and install Splunk from http://www.splunk.com/download. Use the following link to help install & configure; http://docs.splunk.com/Documentation/Splunk/7.2.0/Installation/Chooseyourplatform for Windows, Linux and MacOS users.

Tip: When importing data into Splunk, it appears that Splunk allows automatic recognition of timestamps only if data is not older than 2000 days. The workaround is to change one of the config files (props.conf). After changing the MAX_DAYS_AGO from 2000 to 10000, the data is imported properly. Use the below link for additional guidance. http://docs.splunk.com/Documentation/Splunk/latest/Data/Configuretimestamprecognition Also spinning up a Linux VM to update the .conf file to edit the date and time for the Splunk implementation, will be easier than working with Windows installation.

First Step: Load into Splunk the provided web log file found within Module 10 Content Folder, called "HTTP Log Supplemental Data for Assignment" and perform the following tasks. Submit a write-up outlining how you performed each task and the results of the task via CANVAS. Submission must be in either Word or PDF format.

Data schema's are the attributes of a database structure and/or excel file; like tables, columns and properties.

Tasks:

1. Define a schema representing the contents of the log data, in this schema, you should include the following:

   a. A list of fields in the file *(e.g., ip, dtg, request, code, size, user id, user agent, time date, status code, etc.)*

   b. For each field, provide a definition of the type of the field *(e.g., numeric, enumeration, date/time, string, integer, etc*. Explain your decision for each type.

2. Create a script in the language of your choice for parsing the data in the file and converting it into your schema. The script should parse the web log file provided and should have the following properties: Tip: for assistance with writing python scripts for implementing one variable, like IP, this will help: *using the python's pandas package and ingesting everything as a csv with the "\" delimiter which opens up a bunch of functionality for returning row numbers and occurrence information.*

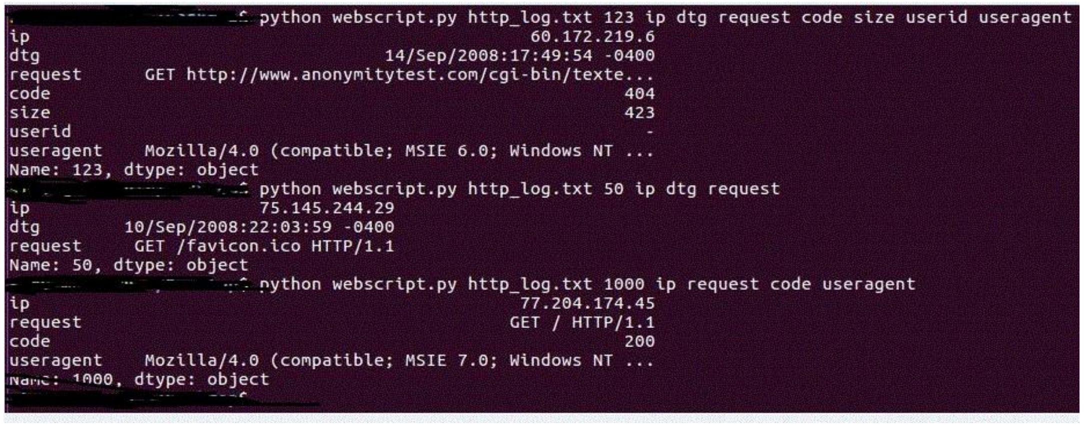
   a. It will take the original provided web log file as input

   b. It will take a number (the index of the record) and one or more optional fields and output the results as specified. For example, if your schema defines "sip", "dip" and "uri" as fields, you should be able to call "script 200 dip sip" and receive both addresses back. In other words; *parse the provided file looking for the specific hard coded field names from the schema you created*

c. The output should be human readable, but the internal representations should be converted to your internal types

d. Since the python or power shell scripts can be large in size, *submit a copy of your script used and provide screen shots of the script running.* You don't need to submit a stand-alone executable.

Example parser script/source code:



```
[redacted] python webscript.py http_log.txt 123 ip dtg request code size userid useragent
ip                                                   60.172.219.6
dtg                                    14/Sep/2008:17:49:54 -0400
request      GET http://www.anonymitytest.com/cgi-bin/texte...
code                                                          404
size                                                          423
userid                                                          -
useragent    Mozilla/4.0 (compatible; MSIE 6.0; Windows NT ...
Name: 123, dtype: object
[redacted] python webscript.py http_log.txt 50 ip dtg request
ip                         75.145.244.29
dtg        10/Sep/2008:22:03:59 -0400
request     GET /favicon.ico HTTP/1.1
Name: 50, dtype: object
[redacted] python webscript.py http_log.txt 1000 ip request code useragent
ip                                                  77.204.174.45
request                                            GET / HTTP/1.1
code                                                          200
useragent    Mozilla/4.0 (compatible; MSIE 7.0; Windows NT ...
Name: 1000, dtype: object
[redacted]
```

3. There are several records in the file that are corrupt, and some that are problematic. Provide a count of corrupt and problematic records, and explain the problems. Modify your script if necessary to automatically eliminate corrupt and problematic records. *Provide a screen shot of your results*. By problematic I mean *entries which are either; security concerns, entries that don't make sense, and/or are malicious/abnormal behavior that is questionable or unusual. Hint: can modify script to skip over lines with extra fields.*

Analysis with Splunk:

4. There are number of spiders in the file, provide a list of the 10 busiest spiders by 1.) user--agent string, 2.) the number of unique addresses they appear from and 3.) describe their associated search engine. *Provide a screenshot of your results*.

5. Attackers will masquerade as spiders, identify at least 3 attackers by IP address that are masquerading as a spider. *Provide a screenshot of your results*. Tip: possible search could be; useragent="<?system*" AND useragent="*bot*" | stats values(ip).

6. Plot a CLEAN timeline of Googlebot activity (*by clean, I mean that you are actually plotting googlebot and not attackers masquerading as googlebot*). How do you determine that you're looking at googlebot? Use the plotting tool of your choice, and plot the total number of requests on month--by--month basis.

7. Explain 3 advantageous for using Splunk at an Enterprise level, other than to view and create data schemas. (e.g., filter specific events a SOC would be interested in- user login attempts, IP's, domains, using Security Orchestration, Automation, and Response (SOAR) solutions, etc.)