# Lab 9: Network Forensics 2

In this lab we will investigate the usage of regular expressions, and using Wireshark filters.

## A Detecting content

For Table 1, and using a Wireshark filter, and Table 2, determine the required evidence. An example is:

http contains "\x89\x50\x4E\x47"

|  |  |  |
| --- | --- | --- |
| **No** | **PCap file** | **Evidence** |
| **1** | http://asecuritysite.com/log/with\_png.zip | Names of PNG files: |
| **2** | http://asecuritysite.com/log/with\_pdf.pdf | Names of PDF files: |
| **3** | <http://asecuritysite.com/log/with_gif.zip> | Names of GIF files: |
| **4** | http://asecuritysite.com/log/with\_jpg.zip | Names of JPG files: |
| **5** | http://asecuritysite.com/log/with\_mp3.zip | Names of MP3 files: |
| **6** | http://asecuritysite.com/log/with\_rar.zip | Names of RAR files: |
| **7** | http://asecuritysite.com/log/with\_avi.zip | Names of AVI files: |
| **8** | http://asecuritysite.com/log/with\_gz.zip | Names of GZ files: |
| **9** | http://asecuritysite.com/log/email\_cc2.zip | Email addresses: |
| **10** | http://asecuritysite.com/log/email\_cc2.zip | Credit card details: |
| **11** | http://asecuritysite.com/log/webpage.zip | IP address details: |
| **12** | http://asecuritysite.com/log/webpage.zip | Domain name details: |

**Table 2:** Examples of signatures

|  |  |
| --- | --- |
| PNG file | "\x89\x50\x4E\x47" |
| PDF file | "%PDF" |
| GIF file | "GIF89a" |
| ZIP file | "\x50\x4B\x03\x04" |
| JPEG file | "\xff\xd8" |
| MP3 file | "\x49\x44\x33" |
| RAR file | "\x52\x61\x72\x21\x1A\x07\x00" |
| AVI file | "\x52\x49\x46\x46" |
| SWF file | "\x46\x57\x53" |
| GZip file | "\x1F\x8B\x08" |
| Email addresses | "[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.\_%+-]" |
| IP address | "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}.[0-9]{1,3}" |
| Credit card details (Mastercard) | "5\d{3}(\s|-)?\d{4}(\s|-)?\d{4}(\s|-)?\d{4}" |
| Credit card details (Visa): | "4\d{3}(\s|-)?\d{4}(\s|-)?\d{4}(\s|-)?\d{4}" |
| Credit card details (Am Ex). | "3\d{3}(\s|-)?\d{6}(\s|-)?\d{5}" |
| Domain name: | "[a-zA-Z0-9\-\.]+\.(com|org|net|mil|edu|COM|ORG|NET|MIL|EDU|UK)" |

## B Tshark

We can also process the network traces using Tshark, which is a command line version of Wireshark. For example we can search for a ZIP file with:

tshark -Y "http matches \"\x50\x4B\x03\x04\"" -r with\_zip.pcap -x -V > list

and then view the **list** file.

Now repeat some of the example from the first part, and determine some of the details:

|  |  |  |
| --- | --- | --- |
| **No** | **PCap file** | **Evidence** |
| 1 | http://asecuritysite.com/log/with\_png.zip | Frame numbers with content:  IP addresses involved in exchange: |
| 2 | http://asecuritysite.com/log/with\_pdf.pdf | Frame numbers with content:  IP addresses involved in exchange: |
| 3 | <http://asecuritysite.com/log/with_gif.zip> | Frame numbers with content:  IP addresses involved in exchange: |
| 4 | http://asecuritysite.com/log/with\_jpg.zip | Frame numbers with content:  IP addresses involved in exchange: |

## C Content identification

There are 30 files contained in this evidence bag:

[**http://asecuritysite.com/evidence.zip**](http://asecuritysite.com/evidence.zip)

Using a Hex Editor, see if you can match the magic number, and then change the file extension, and see if you can view them.

|  |  |  |
| --- | --- | --- |
| **File** | **Type** | **What it contains …** |
| **file01** |  |  |
| **file02** |  |  |
| **file03** |  |  |
| **file04** |  |  |
| **file05** |  |  |
| **file06** |  |  |
| **file07** |  |  |
| **file08** |  |  |
| **file09** |  |  |
| **file10** |  |  |
| **file11** |  |  |
| **file12** |  |  |
| **file13** |  |  |
| **file14** |  |  |
| **file15** |  |  |
| **file16** |  |  |
| **file17** |  |  |
| **file18** |  |  |
| **file19** |  |  |
| **file20** |  |  |
| **file21** |  |  |
| **file22** |  |  |
| **file23** |  |  |
| **file24** |  |  |
| **file25** |  |  |
| **file26** |  |  |
| **file27** |  |  |
| **file28** |  |  |
| **file29** |  |  |
| **file30** |  |  |
| **file32** |  |  |
| **file33** |  |  |
| **file34** |  |  |
| **file35** |  |  |
| **file36** |  |  |
| **file37** |  |  |
| **file38** |  |  |
| **file39** |  |  |
| **file40** |  |  |

There is a list of magic numbers here: <http://asecuritysite.com/forensics/magic>

# Additional

## D RegEx

Using regex101.com, enter the following code:

There is not much we can do apart from contacting There is not much we can do apart from contacting f.smith@home.net to see if he would like to reboot the server at 192.168.0.1. If he can do this then I will call him on 444.3212.5431. My credit card details are 4321-4444-5412-2310 and 5430-5411-4333-5123 and my name on the card is Fred Smith. I really like the name domain fred@home. Overall our target areas are SW1 7AF and EH105DT. I tested the server last night, and I think the IP address is 10.0.0.1 and there are two MAC addresses which are 01:23:45:67:89:ab or it might be 00.11.22.33.44.55.

The book we will use is “At Home” and it can be bought on amazon.com or google.com, if you search for 978-1-4302-1998-9. My password is:

a1b2c3

Best regards,

Bert.

EH14 1DJ

+44 (960) 000 00 00

1/1/2009

Now, using the Python code generator, create Python code to detect the following:

1. Email address.
2. MAC address.
3. Credit card details.

Some hints are at: https://asecuritysite.com/dlp/day1

## E Splunk

Using Splunk at **http://** **asecuritysite.com:8000** determine the following. You will be allocated a login. We can use regular expressions to find information. For example, to find the number of accesses from an IP address which starts with “182.”, we can use:

get | regex \_raw="182\.\d{1,3}\.\d{1,3}\.\d{1,3}"

Determine the number of accesses for GET from any address which begins with 182:

The security team search for an address that is ending with .22, and do a search with:

get | regex \_raw="\d{1,3}.\d{1,3}.\d{1,3}.22"

But it picks up logs which do not include addresses with .22 at the end. What is the problem with the request, and how would you modify the request:

You are told that there’s accesses to a file which ends in “a.html”. Using a regular expression, such as:

get | regex \_raw="[a]+\.html"

Outline three HTML files which end with the characters ‘a’, or an ‘e’, and have ‘.html’ as an extension:

A simple domain name check is:

get | regex \_raw="[a-zA-Z\.]+\.(com|net|uk)"

If we now try:

get | regex \_raw="[a-zA-Z0-9\-\.]+\.(com|org|net|mil|edu|COM|ORG|NET|MIL|EDU|UK)"

we will return events with domain names:

Outline which ones have been added:

We can search for email addresses with:

get | regex \_raw="(?<email>[\w\d\.\-]+\@[\w\d\.]+)"

Which email addresses are present:

We can search for times using regular expressions, such as:

get | regex \_raw="[0-9]{2}\:22\:[0-9]{2}"

How many GET requests where there at 22 minutes past the hour:

How many GET requests were made at 14 seconds past the minute: