# Project 2: DNS activity monitor

Due 11:59pm Sunday, October 13th. (almost four weeks)

For this project you will be parsing DNS logs and creating a profile for a DNS intrusion detection system. You will be given pre-configured VMware Workstation virtual machine with dns proxy installed.

NOTE 1: This is an analysis-based project. Try to collect the required data and start working with a potential solution. There is no perfect solution. Your code will be graded based on the accuracy obtained.

NOTE 2: Get the VM running on your system as soon as possible. If you don't have access to any computer to run this VM, then I can provide you with some sample logs to get started.

## Setup

## Virtual Machine

1. Before starting you will need to install VMware Workstation. It can be downloaded from with the license

https://watson.onthehub.com/

It may ask you to reboot the computer after installation is complete.

2. Download the pre-configured VM from **https://goo.gl/gcJWQe**

If the file download is too slow for you, stop by my office with a flash drive of at least 6gb of free space.

3. Unzip the VM. You should be left with a directory named Debian_cns. This directory contains the virtual machine disk files (*.vmdk), virtual machine configuration file (*Debian_cns.vmx*) and some other important files

4. Start VMware Workstation, click File->Open and select the virtual machine configuration file in the Open dialog window.

5. Click VM->Settings and make sure your network adapter is set into bridged or NAT mode.

6. Start the VM.

VMware Workstation may ask if you moved or copied it. Answer that you have copied it.

7. At this point you should have VM up and running.

8. Login with the same credentials you use in the lab.

9. Launch Iceweasel browser. Make sure you can browse to some web sites with it.

10. Update /etc/resolv.conf file to look like:

*nameserver 127.0.0.1*

11. This setting will make all of the program inside Debian_cns use your own server. In other words any DNS requests made within this VM will be resolved and recorded in the logs by the DNS server located on the same VM.

12. If you run dhclient again it may update resolv.conf file. Make sure you restore the resolv.conf file after you run dhclient.

You may also protect the resolv.conf from being changed by setting immutable flag with

*chattr +i /etc/resolv.conf*

The +i option (attribute) write protects /etc/resolv.conf file so that no one can modify it including root user. If you want to remove immutable flag run *chattr -i /etc/resolv.conf*

Navigate to ~/Development/mana/sslstrip-hsts/dns2proxy. You will find a python program **dns2proxy.py**.

Run **python dns2proxy.py** in a terminal. This will start your own DNS server. It will listen to every dns

request your system will make including dns requests from your browser.

**Note: Once you change your nameserver to 127.0.0.1, your Internet access will work only when your DNS proxy server is running. So, make sure you are running the code (dns2proxy.py) in a terminal and then access the Internet.**

## Traces

The dns requests log file (*dnslog.txt*) can be found in dns2proxy.py current working directory. Generally, it is the same place where dns2proxy.py is located.

You will analyze *dnslog.txt* file. This file contains all the queries made to your server from various client programs (e.g. iceweasel). As a result, when you browse to some website all the dns request Iceweasel makes to render the web page will be logged in *dnslog.txt* file. This will be the file that you will use as an input to your program.

## Undergraduate Students

1. With the client VM visit the random web pages, wait at least 60-90 seconds before visiting the next page. The next page can be opened in new tab or can be run within the same tab. We will test both cases. It is very important that you do NOT clean the *dnslog.txt* file between visits. All of the clients DNS requests for the web pages should be in ONE file.
2. Write a program in python to process the dnslog file and extract useful information from it:

   a) For visited page count the number of unique DNS requests.

   b) For visited page, print out the unique DNS names observed.

   c) For visited page, print out the time the first requested webpage was visited.

The hard part is to correctly identify the main DNS request for the webpage visit. Present the output in a human readable format in a file report.txt.

HINT 1: **The dns2proxy.py code doesn't output time in milliseconds by default. Modify the code to produce time in milliseconds as shown in example below.**

For example, when we visit the webpage "google.com" later followed by "cnn.com" we get the following results in the DNS log: (This is only a sample)

```
2016-04-15 17:31:04.031 Client IP: 127.0.0.1    request is    www.google.com. IN A
2016-04-15 17:31:04.136 Client IP: 127.0.0.1    request is    36cc248b.mpstat.us. IN A
2016-04-15 17:31:04.234 Client IP: 127.0.0.1    request is    36cc248b.mpstat.us. IN AAAA
2016-04-15 17:31:04.654 Client IP: 127.0.0.1    request is    rgaudit.1rx.io. IN A
2016-04-15 17:33:04.012 Client IP: 127.0.0.1    request is    www.cnn.com. IN A
2016-04-15 17:33:04.098 Client IP: 127.0.0.1    request is    www.cnn.com. IN AAAA
2016-04-15 17:33:04.123 Client IP: 127.0.0.1    request is    aspen.turner.com. IN A
2016-04-15 17:33:04.138 Client IP: 127.0.0.1    request is    aspen.turner.com. IN AAAA
2016-04-15 17:33:04.280 Client IP: 127.0.0.1    request is    www.i.cdn.cnn.com. IN A
2016-04-15 17:33:04.339 Client IP: 127.0.0.1    request is    www.i.cdn.cnn.com. IN AAAA
2016-04-15 17:33:04.501 Client IP: 127.0.0.1    request is    z.cdn.turner.com. IN A
```

```
2016-04-15 17:33:04.654 Client IP: 127.0.0.1    request is    z.cdn.turner.com. IN AAAA
2016-04-15 17:33:04.998 Client IP: 127.0.0.1    request is    cdn.adsnetwork.com. IN A
2016-04-15 17:33:05.231 Client IP: 127.0.0.1    request is    cdn.adsnetwork.com. IN AAAA
2016-04-15 17:33:05.467 Client IP: 127.0.0.1    request is    a.tracking.skynet.com. IN A
2016-04-15 17:33:05.889 Client IP: 127.0.0.1    request is    a.tracking.skynet.com. IN AAAA
```

For this example, your program should produce the following output

```
www.google.com: 3 Time: 2016-04-15 17:31:04.031
1.  www.google.com
2.  36cc248b.mpstat.us
3.  rgaudit.1rx.io

www.cnn.com: 6 Time: 2016-04-15 17:33:04.012
1.  www.cnn.com
2.  aspen.turner.com.
3.  www.i.cdn.cnn.com
4.  z.cdn.turner.com
5.  cdn.adsnetwork.com
6.  a.tracking.skynet.com
```

**Some Caveats**: If you first visit "google.com", then "cnn.com", then "google.com" again.  The output should be as follows:

```
www.google.com: 3 Time: 2016-04-15 17:31:04.031
1.  www.google.com
2.  36cc248b.mpstat.us
3.  rgaudit.1rx.io

www.cnn.com: 6 Time: 2016-04-15 17:33:04.012
1.  www.cnn.com
2.  aspen.turner.com.
3.  www.i.cdn.cnn.com
4.  z.cdn.turner.com
5.  cdn.adsnetwork.com
6.  a.tracking.skynet.com

www.google.com: 4 Time: 2016-04-15 17:39:21.421
1.  www.google.com
2.  36cc248b.mpstat.us
3.  rgaudit.1rx.io
4.  imap.gmail.com
```

Each visit should produce its own entry in the report.

## Program specifications:

1. Source code with comments has to be named **dnsproject.py** (NOT myproject.py, prog1.py or anything else)

2. The program should not ask any questions. It should read **dnslog.txt** file in the current directory and produce the report file within the same directory.

3. Your program should run with no errors on **python 3**. (if it doesn't run you lose most of the grade)

4. It should not take more than 30 seconds minute to produce the result.

**Submission**

1. Your code should be uploaded in your GitHub repository by the deadline. Please try to use GitHub for your day to day progress. Your regular code progress should be pushed to cloud with commits. (It is a very good practice for you, so please make good use of this.)

2. Make sure your dnsproject.py code is placed in the **DNSproject** folder. Also include your dnslog.txt and the report.txt in the same directory.

3. Every student should comment their logic and use the given README.md file in repository to document your approach towards the problem. Documentation will be allotted some grades.

## Graduate Students

Graduate students start with the same project as the Undergrads and then do some additional work. Graduate students will design a DNS blocking filter.
**Task 1**. Implement undergraduate project.

***For submission***: put it into DNSproject folder along with your dnslog.txt and report.txt.

**Task 2**. If you go through the output of your report.txt, you might see some DNS requests are irrelevant to the main functionality of the web site (Ads, Identity trackers, etc.).
Let's assume visiting "cnn.com" results in the following dns requests:

www.cnn.com: 6 Time: 2016-04-15 17:33:04.012

1. www.cnn.com
2. aspen.turner.com.
3. www.i.cdn.cnn.com
4. z.cdn.turner.com
5. cdn.adsnetwork.com
6. a.tracking.skynet.com

In this list cdn.adsnetwork.com and a.tracking.skynet.com can be blocked because they host some things you don't want in your computer. To do so you will need to figure out how to use dns2proxy software to block/divert dns requests.
After blocking the dns request your browser should visit only unblocked web-sites and load resources only from unblocked web-sites.

www.cnn.com: 6 Time: 2016-04-15 17:33:04.012

1. www.cnn.com
2. aspen.turner.com.
3. www.i.cdn.cnn.com
4. z.cdn.turner.com

Find out how to use dns2proxy to prevent communication with websites you don't want. Modify dns2proxy code/configuration if necessary.
**HINT**: To block communication with servers with particular dns name, you can return some fake IP for this name (e.g.  some unused local network address like 192.168.127.127).  Let's say cdn.adsnetwork.com and a.tracking.skynet.com translate into 192.168.127.127.  Your browser will attempt to load resources from 192.168.127.127 since there is no such host the browser will fail to load them. Thus, the blocking behavior is achieved.
**For submission**: put the whole dns2proxy software and configuration files you used into "Graduate/Task2/" folder.

**Task 3.** By default, the dns2proxy code writes every DNS requests made to the dnslog.txt file. Modify the code such that it does not write the blocked dns requests to the file. The dnslog.txt file should only contain the dns requests which are not blocked. Test if your dnsproject.py program can run on filtered dnslog.txt file.

Submit this task in a separate folder "Graduate/Task3" folder

**Task 4**. Do some research and identify 6 large web sites that contain tons of ads, trackers etc. The good starting point are financial news web sites, news web sites, reviews web sites, insurance related sites etc. Build blocking profile for the ads and other junk they carry. Put it into blacklist and test it.

For submission: use the folder "Graduate/Task4"

Organize the information in separate folders such that:

a) Folder name is equal to website name

b) Each website folder should contain original dns2proxy dnslog.txt file.

c) The output of your dnsproject.py dnslog.txt for particular web site (report.txt) with comments identifying "bad" dns names.

d) The dnslog.txt file after the blocking was enabled (name it dnslog_with_blocking.txt). You will need to go to this web site again with blocking enabled.

e) The output of the second run of the dnsproject against new dnslog_with_blocking.txt (name it report_with_blocking.txt)

d) So, each folder has dnslog.txt, report.txt, dnslog_with_blocking.txt, report_with_blocking.txt, and all supporting config you use to block the "bad" dns.

**Task 5 (extra points)**. Make dns2proxy print the IP address where DNS name was resolved into. For example:

```
2016-04-15 17:33:04.280 Client IP: 127.0.0.1    request is    www.i.cdn.cnn.com. IN A      12.13.14.15
2016-04-15 17:33:04.339 Client IP: 127.0.0.1    request is    www.i.cdn.cnn.com. IN AAAA   12.13.14.16
2016-04-15 17:33:04.501 Client IP: 127.0.0.1    request is    z.cdn.turner.com. IN A       124.146.15.
2016-04-15 17:33:04.654 Client IP: 127.0.0.1    request is    z.cdn.turner.com. IN AAAA    191.55.40.5
2016-04-15 17:33:04.998 Client IP: 127.0.0.1    request is    cdn.adsnetwork.com. IN A    192.168.127.127
2016-04-15 17:33:05.231 Client IP: 127.0.0.1    request is    cdn.adsnetwork.com. IN AAAA   192.168.127.127
2016-04-15 17:33:05.467 Client IP: 127.0.0.1    request is    a.tracking.skynet.com. IN A   192.168.127.127
2016-04-15 17:33:05.889 Client IP: 127.0.0.1    request is    a.tracking.skynet.com. IN AAAA 192.168.127.127
```

Notice how it prints real IPs and spoofed IPs for different names.
For Reference of original Github repository: **https://github.com/LeonardoNve/dns2proxy**
**Note**: Even with Task 5 extra points, the maximum you can score is 100.

## Submission

1. Your code should be uploaded in your Github repository by the deadline. Please try to use GitHub for your day to day progress. Your regular code progress should be pushed to cloud with commits. (It is a very good practice for you, so please make good use of this.)

2. Use the following folders for your submission:

a. Anything required for Undergraduate part dnsproject put into "DNSproject" directory (see undergraduate submission section)

b. The whole dns2proxy directory which contains your custom modified dns2proxy and any relevant files in "Graduate/Task2".

c. Task 3 includes change of code. So, keep task 3 in a separate folder “Graduate/Task3” which includes the whole code from dns2proxy.py.

c. Your DNS research folders for task 4 put in folder “Graduate/Task4”

d. README.md file with any relevant comments and your approach for solving this problem.

e. Comment your dnsproject.py code, and your modifications to the dns2proxy.