Project 2:

Malware Analysis and Machine Learning

Spring 2022

Introduction:

The goals of this project:

- Familiarize you with the types of behaviors that can be displayed by malware and how to safely analyze these behaviors.
    - Provide hands-on experience using a standard tool in the industry called Cuckoo. For more details about Cuckoo you can visit https://cuckoosandbox.org/ and read more about it.
    - Demonstrate what a safe environment to run malware looks like.
- Familiarize you with machine learning concepts applied to malware analysis.
    - Build on your experience using Cuckoo by machine learning strategies.
    - Get familiar with malware behavior clustering, and how to use it to classify malware.
    - Build a machine learning based malware classification tool using Malheur. For more details about Malheur you can visit http://www.mlsec.org/malheur/manual.html and read more about it.

Additional information:

- There is a Canvas quiz where you can submit answers for phase 1, and another quiz where you can submit answers for phase 2.
- Phase 3 will be submitted in Gradescope..
- Phase 4 is an essay portion which will be submitted via Gradescope **and** via Canvas.
- Make sure you have a computer with sufficient horsepower to run the Project's VM. Minimum RAM required for this project is 4 GB RAM for the VM, 8GB RAM on your host.
- IMPORTANT WARNING PLEASE READ: This assignment contains actual executable malware artifacts used to help you understand how malware behaves. Although we have taken steps to make the assignment as safe as possible, you must be extra careful not to execute and/or transfer malware code outside of the sandbox environment. Failure to do so might cause harm to you and others.
- We offer a Frequently Asked Questions (F.A.Q.) thread in Ed Discussions here.

- The F.A.Q thread will be constantly updated. Therefore, before asking anything, take a look at it, and, if your question is not covered already, feel free to post it as a reply in the same thread.

Phase 1 (50 points):

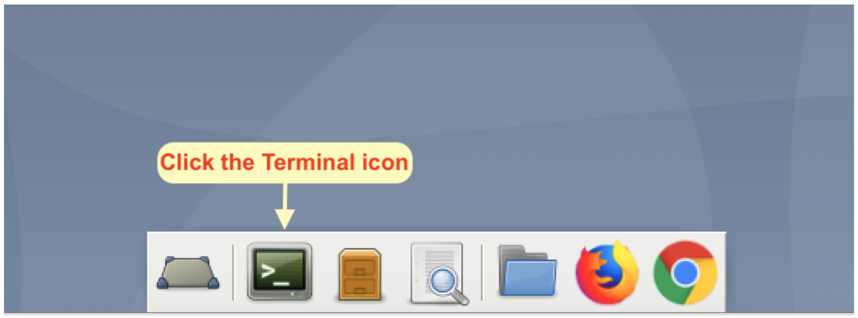
Setup (0 points)

1. Run the project virtual machine (please see the Ed discussion posts regarding VMWare and Mac M1 if you have any issues regarding your virtual machine).
2. Download the project VM here: https://cs6035.s3.amazonaws.com/CS6035_P2_P3_P4_Spring_2022.ova
   - Please note that the file is over 8GB, so it will take some time to download. Do not wait until the last minute to download it. Do it right away!
   - SHA-256 for this file: D9729FA95AE647607D83BFE4BBEFDEAF286E1AA4F02B8DBC69CF79A283E2AB91

3. Import the OVA into VirtualBox. Note that this may be easiest to do by double-clicking the OVA file and letting the file association open VirtualBox.

4. Log in to the newly imported VM using these credentials:
   - Username: debian
   - Password: debian

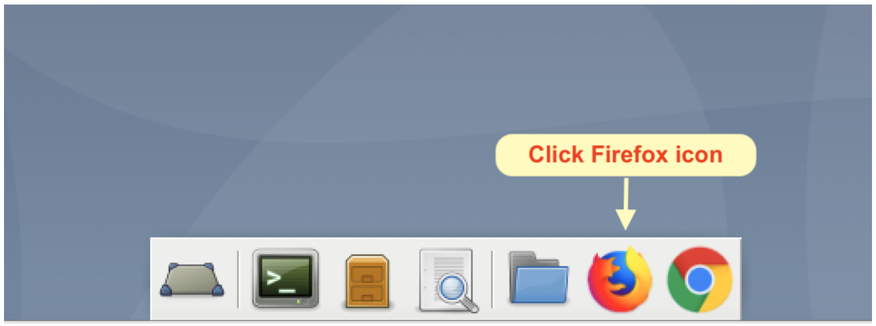
5. Once you are inside Project 2 VM, open a terminal window by clicking the proper icon in the VM GUI taskbar, as follows:



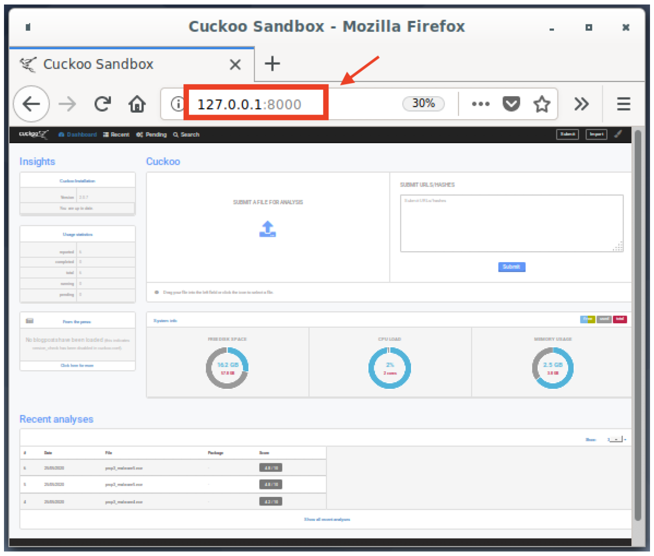
6. In the terminal window, execute the following commands to start up Cuckoo's core system:

```
$ . venv/bin/activate
$ cuckoo web runserver 0.0.0.0:8000
```

7. Open up Firefox web browser by clicking the proper icon in the VM GUI taskbar, as follows:



8. In the Firefox window, type http://127.0.0.1:8000/ in the address bar, and hit enter:



NOTE: If you happen to reboot your VM, then make sure you go through steps 4 through 8 again. If you don't, then you will not be able to use Cuckoo properly.

Getting used to Cuckoo's Web-based Analysis Interface (0 points)

Cuckoo provides a web-based analysis interface. It is very intuitive, and you should not have any trouble navigating through it. You can also look at Cuckoo's manual if you wish to get familiar with its advanced options: https://cuckoo.sh/docs/.

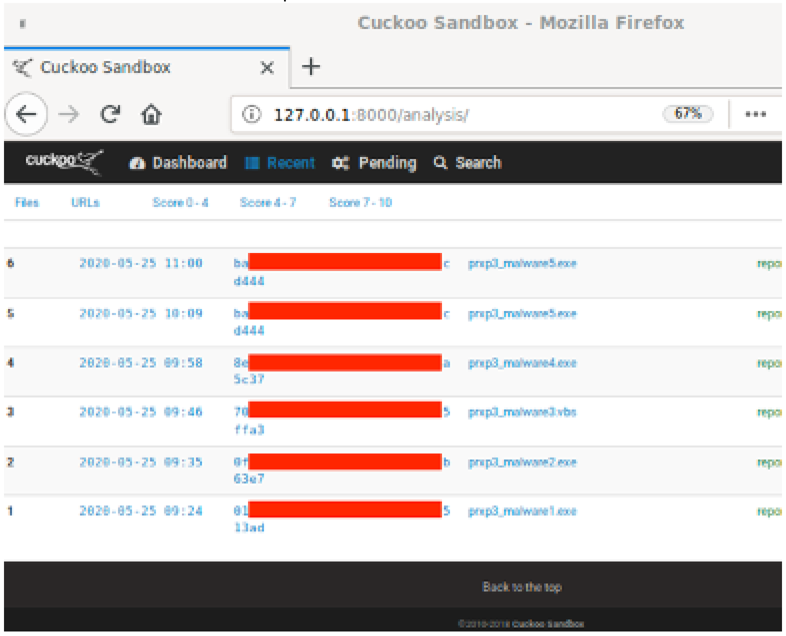
Normally, the process of running a malware sample through Cuckoo consists of three stages:

1. Submit the malware sample to Cuckoo.
2. Run the sample in Cuckoo's sandbox environment to collect data about the malware file and its execution.
3. Analyze the data and artifacts that Cuckoo has collected during malware execution (that includes a complete report in JSON format about the malware behavior among other things).

We have already completed steps #1 and #2 for you. You need to finish the Canvas assignment based on the data and reports generated by Cuckoo in this VM.

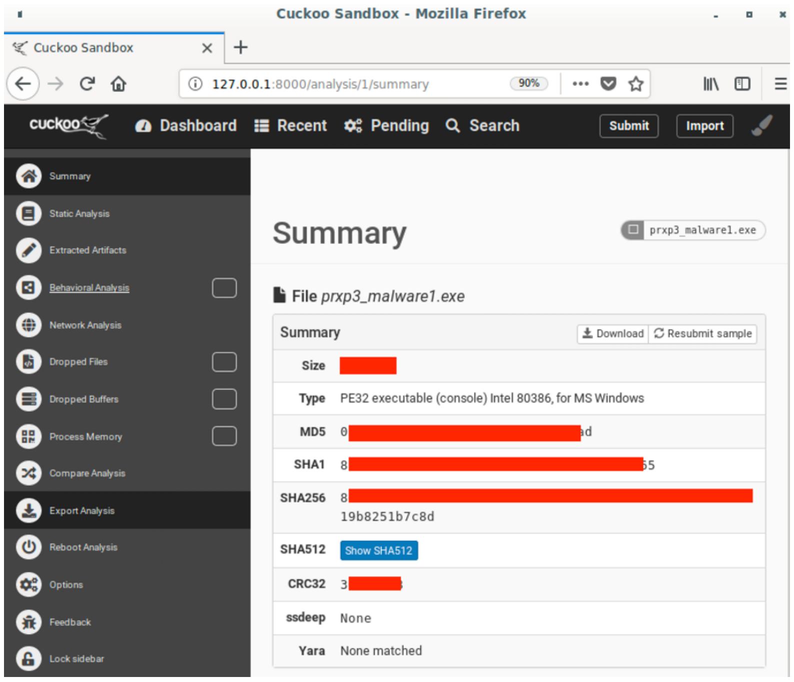
To complete this project, you will have to analyze **Cuckoo sandbox results** for each malware sample. To analyze those results via Cuckoo web interface, do the following:

1. First, click the "Recent" tab on Cuckoo's web interface top banner. There you will see the list of all malware samples you are required to analyze. You can click any of the malware samples to access its report. You can see that list in the picture below.



6	2020-05-25 11:00	ba[redacted]c d444	prxp3_malware5.exe	repo
5	2020-05-25 10:09	ba[redacted]c d444	prxp3_malware5.exe	repo
4	2020-05-25 09:58	8e[redacted]a 5c37	prxp3_malware4.exe	repo
3	2020-05-25 09:46	70[redacted]5 ffa3	prxp3_malware3.vbs	repo
2	2020-05-25 09:35	0f[redacted]b 63e7	prxp3_malware2.exe	repo
1	2020-05-25 09:24	01[redacted]5 13ad	prxp3_malware1.exe	repo

2. After you click the sample you want to assess, you will be taken to its report page as depicted in the image below:

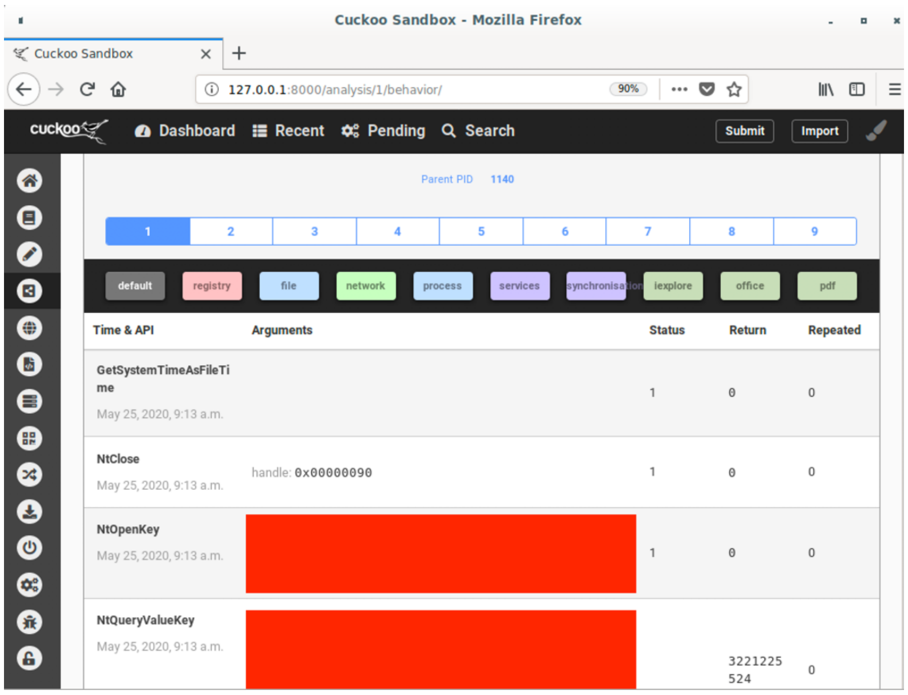


Summary

File *prxp3_malware1.exe*

Summary	
Size	[redacted]
Type	PE32 executable (console) Intel 80386, for MS Windows
MD5	0[redacted]d
SHA1	8[redacted]5
SHA256	8[redacted] 19b8251b7c8d
SHA512	Show SHA512
CRC32	3[redacted]
ssdeep	None
Yara	None matched

3. As you can see, Cuckoo offers many sections in the malware report, like "Summary", "Static Analysis", "Behavioral Analysis", "Dropped Files", among many others. You are advised to explore all options, as many of them will give you insights on how to answer the Project's questions. Nevertheless, you should pay special attention to the section called "Behavioral Analysis", as it provides the core of Cuckoo's analysis information. That section shows all API calls (usually thousands of them) that were executed by the malware while running in the sandbox environment. Those API calls are divided into many pages and they can be filtered by category to better facilitate your assessment as depicted in the picture below. Make sure to inspect all processes and their API calls before making a judgment about an answer.

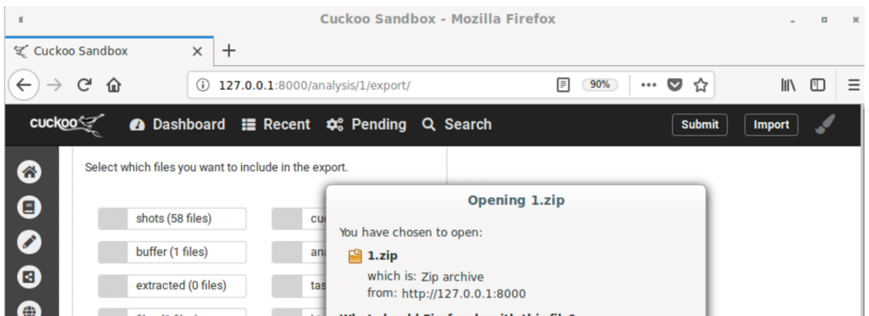


Parent PID 1140

Time & API	Arguments	Status	Return	Repeated
GetSystemTimeAsFileTime May 25, 2020, 9:13 a.m.		1	0	0
NtClose May 25, 2020, 9:13 a.m.	handle: 0x00000090	1	0	0
NtOpenKey May 25, 2020, 9:13 a.m.	[redacted]	1	0	0
NtQueryValueKey May 25, 2020, 9:13 a.m.	[redacted]		3221225524	0

4. Although Cuckoo's web interface should be the first tool you use for your analysis process, **we strongly recommend that you also download the full Cuckoo report in JSON format with accompanying artifacts and base your final decision for any question on it as well.** We instruct you to download the JSON file because Cuckoo is still an experimental software and any minor glitch in the web interface may prevent an inexperienced Malware Analyst from finding the correct answer. Also, that approach is necessary in case you want to automate your analysis or simply wish to use a lighter interface to read through your malware sample report.

5. You may download the full Cuckoo JSON report with accompanying artifacts by going to "Export Analysis" tab in your malware report page. There, you should first unselect everything, and select only "reports", so you do not download unnecessary (and potentially dangerous) files along with the reports. The following picture exemplifies this procedure:



Select which files you want to include in the export.

Opening 1.zip

You have chosen to open:

1.zip

which is: Zip archive
from: http://127.0.0.1:8000

6. Make sure you are only selecting “reports” before you click OK. You will then save a ZIP file containing the reports in your Downloads folder. You should navigate to that folder and unpack the contents of the ZIP file to a new folder that is automatically created by the ZIP application. Inside this newly created folder, you will find another folder named “reports”. You should enter it and open the file “report.json”. This file is a structured text file (it can be opened by any text editor) and it contains all the information about your Cuckoo analysis. A very effective strategy for finding answers for the assignment is simply searching for strings inside this JSON file that would correspond to the correct evidence that could back up your response. JSON files can also be opened (and rendered in a pretty searchable interface) in Firefox. More information about the JSON format online here.

Analyze your malware samples (50 points)

You will investigate and label some of the more sophisticated malware behaviors from the five malware samples we provided. Use the included Cuckoo reports to identify the malware's behavior. Note that malware samples can share behaviors. So initially you should assume that each malware we question you about below has every behavior listed. It's your job to determine if that assumption is actually true.

Hint: Look at the API/system call sequence under each process generated by the malware sample and determine what the malware is doing. Note that each Cuckoo report may contain multiple processes with many different system call sequences. If any of the behaviors are seen (or attempted, but not necessarily successful) in any process in the report, then that malware has attempted that behavior. This is, of course, not completely practical, as legitimate applications may perform the same actions in a benign fashion. We are not concerned with differentiating the two in this assignment, but it is some food for thought.

Clarification for attempted: We mean by “attempted” that a specific action was attempted but failed. By “specific” we mean that it is clear which action is attempted. If you have a registry key, for instance, that is unambiguous (like, say, it is used only to set a startup option), but it fails to change the key, that is an attempt for our purposes. But if you have a more generic registry key that governs multiple settings, we don’t know for sure which key or keys it is attacking and so the action would not count as an “attempt”.

When using hints you will encounter that the same API functions can end with either a W or an A. This is a standard practice in the Windows API, and this document explains the difference (either one could in theory be present in the wild):
https://docs.microsoft.com/en-us/windows/desktop/intl/unicode-in-the-windows-api

For each of the following questions in Canvas, mark which of the malware exhibit the identified behavior:

A. Malware sets itself to run whenever Windows starts up
   a. **Hint:** https://support.microsoft.com/en-us/kb/179365

B. Malware looks up the computer name (possibly doing some reconnaissance)
   a. **Hint:** https://www.technlg.net/windows/computername-registry-key/

C. Potentially looks through Microsoft Outlook address book contents
   a. **Hint**: Look for the malware opening the "Outlook.Application" registry key.

D. Creates and executes a Visual Basic Script (VBS) called "WinVBS.vbs"

E. Prevents users from accessing registry tools
   a. **Hint:** http://www.thewindowsclub.com/prevent-access-to-registry-editor-windows

F. Hides all drives on computer
   a. **Hint:** https://technet.microsoft.com/en-us/library/cc938267.aspx

G. Prevents users from changing remote administrator settings
   a. **Hint:** https://docs.microsoft.com/en-us/windows/access-protection/user-account-control/user-account-control-group-policy-and-registry-key-settings

H. Searches for all possible drives on computer
   a. **Hint:** Look for the malware attempting to open each drive.
   b. **Further hint:** If it is searching for possible drives, it probably isn't calling a specific function to do search behavior but is doing more of a brute force approach

I. Checks for its privileges (this isn't inherently malicious, but the malware possibly performs some different behaviors if it has the proper permissions to do so)
   a. **Hint:** https://msdn.microsoft.com/en-us/library/windows/desktop/aa379180(v=vs.85).aspx

J. Hooks the keyboard (potentially a keylogger)
   a. **Hint:** https://msdn.microsoft.com/en-us/library/windows/desktop/ms644990(v=vs.85).aspx

K. Hooks the mouse
   a. **Hint:** Similar to choice (J)

L. Potentially monitors messages before they appear in a window to the user (possible reconnaissance)
   a. **Hint:** Similar to choices (J) and (K)

M. Communicates with external hosts via IP addresses or domain names, possibly indicative of C2 activity.
   a. **Hint**: Search for the meaning of "Command and Control" (C2) with regards to malware behavior. Examine API calls tagged as "network". Look for a network connection that exchanges buffers with external hosts. Investigate these hosts.

N. Retrieves the current user's username
   a. **Hint:** https://msdn.microsoft.com/en-us/library/windows/desktop/ms724432(v=vs.85).aspx

O. Adds mutex for Eclipse DDoS malware

   a. **Hint:** https://securitynews.sonicwall.com/xmlpost/eclipse-bot-emerges-with-ddos-capabilities-may-8-2014/

P. Adds mutex for IPKillerClient malware
   a. **Hint**: some archived information on this can be found here: https://web.archive.org/web/20150804072501/https://asert.arbornetworks.com/mp-ddoser-a-rapidly-improving-ddos-threat/

Q. Adds mutex for DarkDDoSer malware
   a. **Hint:** Similar to choice (O) and (P)

R. Contacts various SMTP servers (possibly for spamming)
   a. **Hint:** It contacts multiple "smtp.*" domains

S. Drops potentially malicious files on the device.
   a. **Hint**: Take a look at the "dropped files" section of the malware report. A malware sometimes drops a copy of itself to alternate locations in the file system.

T. Adds a malicious cryptographic certificate to the system.

DELIVERABLE: There is a quiz in Canvas for you to match behaviors with each malware sample. Complete the quiz in Canvas and provide the answers for each of the malware samples in **Project 2 - Phase I: Malware Analysis Multiple Choice**.

Phase 2 (10 points):

Dissect some behaviors

Now you must dissect some of the behaviors you have found in Phase 1. You will have to answer open questions about some of the activities you have already analyzed. Be thorough, detail-driven, and patient. You will nail it!

A) C2 Server Identification.
   - For answering this question, you have to identify the C2 server that was used by each malware sample during analysis by providing the IP address of the server. It may be the case that C2 activity appears in the report under a DNS host name (instead of a numerical IP address). In that case, you should resolve the IP address for that host name (e.g. you can use "nslookup" for that). There is a "fill in the blank" question for each malware in Canvas where you are supposed to fill with your answer (the correct IP address). In case there is no C2 activity for a particular malware, you should use "none" (no quotes) as your answer in Canvas. IP addresses should be entered in dotted decimal notation. (**Please note: There will be no credit given for typos.**)

B) External SMTP Servers Identification.
   - To answer this question, you will have to list all DNS fully qualified domain names for the SMTP servers that malware samples might have tried to contact during the analysis (according to Phase 1 – Letter R). Your answer should include the exact strings found in the report in a comma-separated list, without spaces. (**Please note: There will be no credit given for typos.**)

DELIVERABLE: There is a quiz in Canvas with fill in the blank answers for each behavior. Please submit the answers in Canvas quiz **Project 2 - Phase II: Open Questions**.

Remember to submit your answers to Canvas before the deadline. If you start another attempt and leave it open, it will autosubmit at the until date and be late (and possibly receive 0 if you don't answer the questions). **We will not correct this**. It is your responsibility to submit.

Phase 3 (20 points):

Background

In this phase you will learn how to apply Machine Learning concepts to malware classification. You'll be given a dataset of malware samples (similar to those analyzed in Project 2). Using Malheur, the software used for clustering malware in this project, you'll run an unsupervised learning clustering algorithm in order to classify them by behavior.

Understanding malheur (0 points)

1. Malheur uses a hierarchical clustering method to cluster malware samples. However, due to the number of features the malware has (there are many dimensions to the malware data), just performing a hierarchical clustering will take too long. In order to complete this assignment, and become accustomed to how Malheur works, you should read the following sources about malheur:

    - malheur original paper: http://www.mlsec.org/malheur/docs/malheur-jcs.pdf
    - malheur manual: http://www.mlsec.org/malheur/manual.html

2. Pay special attention to malheur configuration parameters. You are given a malheur sample configuration file to serve as a starting point. You will be asked to work with these parameters (not necessarily all of them) during this assignment. Nevertheless, some of these parameters should remain unchanged, since changing them will be detrimental to the assignment's objectives. This way, unless you know exactly what you are doing, you should not change values for:

Parameter	Value
`generic.input_format`	`"text"`
`generic.event_delim`	`";"`
`generic.state_dir`	`"./malheur_state"`
`classify.max_dist`	`1.00`

All other values can be changed in the configuration file. Refer the malheur manual for specifics on each configuration parameter.

A special note about ngram_len

Each semester, many students are concerned about the value of ngram_len in the configuration file and how it relates to this project. The ngram_len parameter is one of the parameters that can be changed in the configuration file, and you may submit any value for this parameter. The malheur manual states the following about ngram_len:

"This parameter specified the length of n-grams. If the events in the reports are not sequential, this parameter should be set to 1. In all other cases, it determines the length of event sequences to be mapped to the vector space, so called n-grams."
http://www.mlsec.org/malheur/manual.html

While the malware behavior is encoded by listing all API calls in sequential order, (see **Understanding the dataset** below) if you receive better performance by selecting a different parameter value than the default provided to you, you may select another value. In Phase 4, discuss why you chose this different parameter, and what might be the cause.

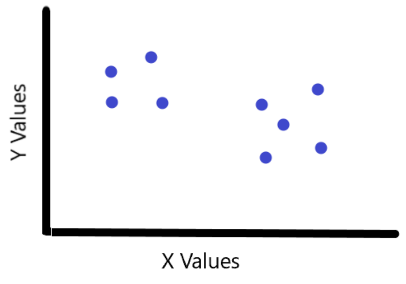
Understanding Unsupervised Learning and Clustering (0 points)

1. Unsupervised Learning is a type of Machine Learning used to extract information from a dataset. It is considered unsupervised, because no prior information on the structure of the dataset is provided to the algorithm. Instead, the unsupervised learning algorithm attempts to extract this information. One of the most common types of unsupervised learning is called clustering. Clustering looks for clusters of data-points in the dataset, and assigns those to their own group (called a cluster).

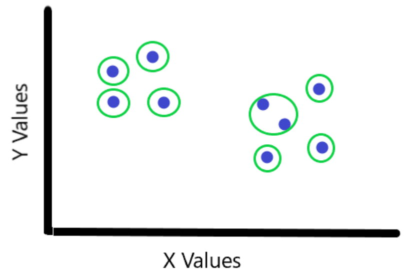
2. There are many kinds of clustering used in data analysis. Malheur, the software used for clustering malware in this project, uses a hierarchical clustering technique. Hierarchical clustering begins by setting each entry in the dataset to be its own cluster. Then, it combines the closest two clusters into one new cluster. It repeats this until either a target number of clusters is reached, or the distance between each cluster exceeds a target amount.

3. To further help you understand hierarchical clustering, we provide an example below. This example uses 2-d data to help you visualize what clustering is actually doing. The process of clustering malware samples is similar, but the data has many more features.

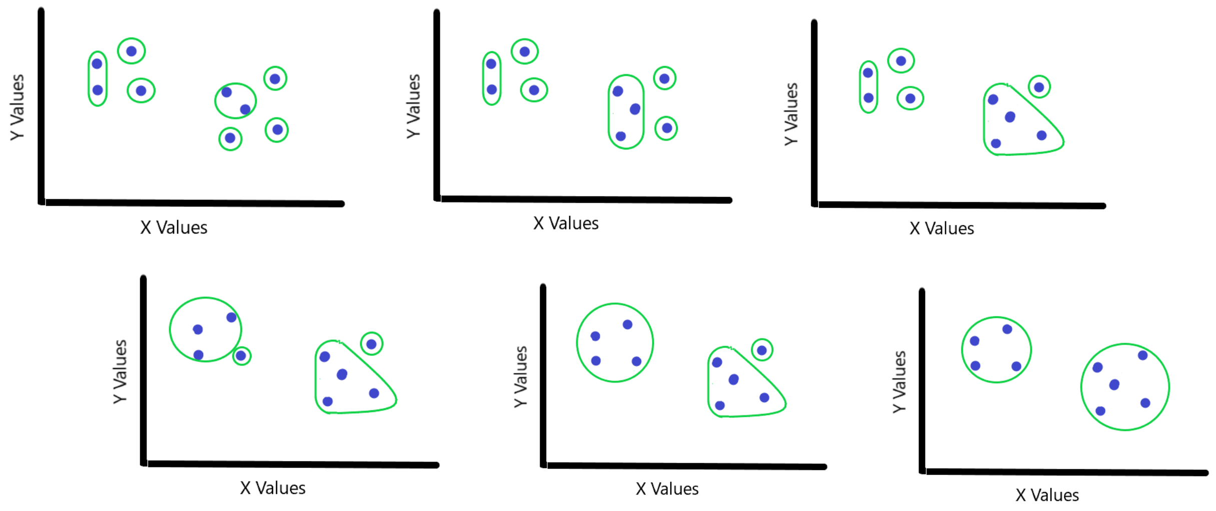
4. Let's say we're given the following data-set. It consists of x-y coordinate data that, when plotted, looks like:



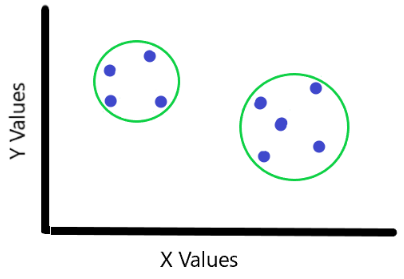
Then, we combine the closest two clusters.



And repeat the process....



We do this until each of the closest two clusters are fairly far away from one another. The exact minimum distance between the final clusters, can usually be set as a parameter to the clustering algorithm. This gives us the clusters we were expecting:



5. This is a general example of hierarchical clustering. In order to cluster malware, malheur extends this method, as you will see in the next section. In order to complete this assignment, you will have to develop a deeper understanding of what Malheur is and how it performs this clustering. We encourage you to further research Machine Learning and clustering methods, beyond this section, in order to better understand the next section.

Understanding the dataset (0 points)

1. In machine learning, a data set is a collection of data points for one of more features that represent the object of study (malware behavior, in our case).

2. Our dataset is based on information extracted from Cuckoo malware behavior reports.

3. Cuckoo malware behavior reports have a section called ‘behavior’. In this section of the report, you can find all API Calls that were part of the malware behavior during Cuckoo’s analysis.

4. The way we chose to encode malware behavior was by simply listing all API Calls, in the order they happened, as indicated by the report.

5. More specifically, we build a list of API call names by extracting the contents of Cuckoo JSON report fields such as ‘behavior->process->[X]->calls->[Y]->api’, where X is the process index number and Y is the process API call index number.

6. In the end, every feature file will be a semicolon separated list of strings, where each string is an API call name, and those API call names are displayed in the order they have actually been called by the malware during Cuckoo analysis.

7. Each sample in our dataset is labeled as to indicate which malware family it belongs to. To gather labels and assign them to our samples, we have used a malware labeling tool called AVClass (https://github.com/malicialab/avclass) along with data from Virus Total website (https://www.virustotal.com/). Because of the way malheur recognizes labels, we have placed them in each dataset sample’s file name as the file extension, while the main part of the file name is the SHA256 hash code for the original malware binary. For example, take the feature file named as:

*0bc19b9304d5c409b9f480a9121c8c8abcef2f3a595ed6b2758daeb2d679b74a.**dinwod***

(Since the file extension for this malware sample is ‘dinwod’, it belongs to malware family Dinwod.)

8. You can use the hash code that represents the file name to research extra information about each individual dataset sample. Several Malware research related websites provide complete malware reports indexed by Hash codes. Since we have used data from Virus Total for our labels, it’s probably a good place to start. You can use the URL https://www.virustotal.com/gui/file/<HASH_CODE> to directly open a malware detection report in Virus Total for the hash code <HASH_CODE>.

    - For example, to gather extra information about a dataset sample with the following file name:

***0bc19b9304d5c409b9f480a9121c8c8abcef2f3a595ed6b2758daeb2d679b74a**.dinwod*

    - All you have to do is to remove the label (dinwood) from the file name and use the hash to craft and visit the following URL:

      https://www.virustotal.com/gui/file/0bc19b9304d5c409b9f480a9121c8c8abcef2f3a595ed6b2758daeb2d679b74a

9. All dataset files are located in the “dataset” folder and they are distributed in two groups: “training” and “testing”.

    - The “training” folder contains files with features extracted from various malware samples based on the strategy we discussed previously. You will use those files to

  "train" your machine learning model using clustering techniques.
  - The "testing" folder is much smaller and contains the files you will use to test the model you will eventually create.
  - We will discuss the training and testing phases of your assignment in more details ahead.

Setup (0 points)

1. No setup required, mostly! We have already provided the malheur binaries, datasets, and example configuration files. Just enter the avml directory on the VM desktop:

```
$ cd /home/debian/Desktop/avml/
```

Using malheur (0 points)

Now, it's time to get familiar with *malheur*. As an initial warm up exercise, let's learn same sample commands:

Cluster samples (training phase):

```
$ malheur -c config.mlw -o training.txt -vv cluster dataset/training/; head training.txt
```

1. This command tells malheur to analyze the dataset contained in dataset/training/ (11,000+ samples) in order to cluster all its samples into groups of similar malwares. Since malheur is using the dataset to learn how to group malware families, we are using a special subset that is called training dataset.

2. In this phase, malheur will output quality assessment information. More specifically, it will show you three main measurements: precision, recall, and f-score. Take this opportunity to research a little bit about them. Those measurements are all related to how good our clustering has been. The way malheur does it is by looking at each sample's label. If you take a look at the contents of directory 'dataset/training/', you will see lots of files, each of which with a different file extension. Those files extensions are the labels for each sample, and malheur extracts them to see if each cluster actually groups similar samples (similar labels) and how many samples that should have been part of the group were left out. This analysis is eventually represented in numbers by precision, recall, and f-score.

Verify clustering (testing phase):

```
$ malheur -c config.mlw -o testing.txt -vv classify dataset/testing/; head testing.txt
```

3. This command tells malheur to use the clusters it has built in the training phase (using samples in dataset/training/) to classify all samples inside dataset/testing. Our objective is to decide whether

the model generated in the training phase is good or bad. That is one of the reasons all samples in the testing dataset used to be part of the training dataset but were randomly selected and put apart. They are selected this way, so testing data represents reality while minimizing bias towards the training data. Think about It as it were a student preparing for an exam: the end goal is succeeding in life, but the exam will give us a rough idea if that will actually happen. The same way, malheur "studies" (analysis) and "learns" (builds a model) from the training dataset (training phase) so it can succeed in real life (classify malware samples in the wild). Nevertheless, malheur will have to pass the exam (testing phase) to make sure it's ready. Would you put the exact same questions from study material in the exam? I hope not! Exam questions should be unique and, at the same time, representative of the real-life skills students are supposed to learn. Welcome to the testing phase!

4. In this phase, malheur will also output precision, recall, and f-score. You should think of those measurements as malheur's final grade. During this phase, malheur isn't being tested against the same exact data used to train it. It now uses a different dataset where all testing instances were statically chosen to adequately represent all training data, even though it is not the same data used in the training phase.

Cluster and Classify (20 points)

1. Work with malheur parameters in its configuration file so you can build a model to classify malware samples with at least 70% f-score during the testing phase. For that, you should run thorough, successive runs of the training phase command followed by the testing phase command. Before every run, you should adjust malheur's configuration file parameters the way you see fit, based on the knowledge you have acquired during assignment preparation (if you have skipped it, it is probably a good idea to take a step back now and read it). You can choose to do it blindly or to make judgment calls before each run. You just have to remember two things:

    - You have to achieve at least 70% f-score in the testing phase;
    - You will have to explain the strategy you used to achieve your results, and how the value for each parameter relates to those results.

2. After you have reached the 70% f-score goal, use the same model you have trained malheur with to classify each of your Project 2 original malware samples. For that, we have provided you feature files for all of Project 2 malware samples containing features extracted from their Cuckoo's JSON. Those feature files are located inside the directory "subjects" in the main assignment path (/home/debian/Desktop/avml/subjects).

3. In order to use your newly created malheur model to classify the given malware samples, make sure you are inside the assignment main directory, and run the following command:

Classify Project 2 samples (classification phase):

```
$ malheur -c config.mlw -o classify.txt -vv classify subjects/; head classify.txt
```

- Your goal is to have all Project 2 samples properly classified (none should be labeled 'rejected') using your model with classify.max_dist set to 1. (as mentioned above, this is required)
- If you can't achieve those classification results for Project 2 malware samples, try revisiting your training and testing phases with different parameters until you can

comply with everything.

- Also, you can take a look at Virus Total reports for the prototypes that malheur has assigned to your subjects. Seeing the full malware behavior might give you a better idea about the direction you should take.

4. By achieving all Task's goals, you will have built a tool that is able to classify unknown executable binary samples (no prior knowledge needed) into potential malware families. It is your very own AV software!

5. TASK GOALS: You have two main goals for this task:

   - GOAL 1: the first goal is to achieve 70% f-score during the testing phase only (you should NOT consider f-score results for other phases for the first goal, just the testing phase).

   - GOAL 2: the second goal is to classify all Project 2 malware samples with maximum distance of 1 (none should be labeled 'rejected') as indicated above.

Goal 1 is a prerequisite for Goal 2. Therefore, any results for Goal 2 are only considered valid if your configuration parameters also allow for Goal 1 to be achieved.

6. PARTIAL CREDIT: Partial credit of **10 points** will be given in case you deliver Goal 1 without Goal 2. Since Goal 1 is a prerequisite for Goal 2, you will be given 0 credits for the entire task in case you do not achieve Goal 1, no matter what you do with regards to Goal 2.

7. DELIVERABLE: Your deliverable for this part of the assignment will be your final malheur configuration file (config.mlw), the one you have used to achieve all goals (70% f-score during testing phase and Project 2 malware samples classification with maximum distance of 1). There should be only one configuration file for both objectives. Please submit the malheur configuration file in the Gradescope assignment **Project 2 - Phase III: Malware Clustering**.

Phase 4 (10 points)

Malheur Summary

1. Explain the importance of Precision with regards to the model you have just created using malheur. Provide the formula/equation and also why the precision calculation is important to evaluating the model's performance. **(1.5 points)**

2. Explain the importance of Recall with regards to the model you have just created using malheur. Provide the formula/equation and also why the Recall calculation is important to evaluating the model's performance. **(1.5 points)**

3. Explain the importance of F-score with regards to the model you have just created using malheur. Provide the formula/equation and also why the F-score calculation is important to evaluating the model's performance. **(1.5 points)**

4. A key aspect to malheur and performing unsupervised machine learning is the use of 'prototypes'. Please explain in detail what 'prototypes' are with regards to malheur? **(1.5 points)**

5. Why are prototypes important to performing clustering and classification in malheur? Explain how they are related to describing the clustering group. **(1.5 points)**

6. Please identify and explain the importance of the key mathematical distance calculation Malheur uses to calculate distances between samples? In particular, explain how this calculation is used in respect to q-grams. **(1.5 points)**

7. Why is it important to have API call names (our features) in the order calls were actually made in the malware execution? Also, what happens If the order is lost? **(1.5 points)**

8. Explain the difference between intra-cluster cohesion and inter-cluster separation. What is the commonly used measure for intra-cluster cohesion? **(1.5 points)**

9. Provide a detailed summary explaining how you have achieved your proposed values for malheur's parameters in task 1 (i.e. strategy, rationale, steps), and how those values relate to your results (i.e. precision, recall, f-scores). No credit will be given for this question in case you choose not to complete Phase 3 Goal 1. **(8 points)**

DELIVERABLE: Phase 4 answers will be submitted in Gradescope and Canvas in PDF format. Please enter your answers in the Gradescope and Canvas assignments **Project 2 - Phase IV: Malheur Summary**. You can access this in Canvas by clicking on the assignment of the same name.

a) You **MUST** provide citations in JDF format which uses APA style; weblinks alone do NOT count. Refer to sections "3.1 In-line citations", "3.2 Reference lists", and "4 References". (**5 points** will be deducted for not providing citations)
b) You **MUST** submit to both the Gradescope and Canvas assignments. If you do not submit to **both Canvas and Gradescope** your submission may not be graded and you have the potential to lose all points. See the class policy on submitting to both Canvas and Gradescope here.

Reflection:

Now that you have some experience analyzing malware, take a moment to read this brief reflection. For this project you used an analysis tool that does the analysis for you. In practice, entire teams of people are devoted to work on a single malware executable at a time to debug it, disassemble it and study its binary, perform static analysis techniques, dynamic analysis techniques, and other techniques not included in Cuckoo to thoroughly understand what the malware is doing. Luckily for you, it takes an enormous amount of time to perfect/improve the skills of malware analysis, so we don't require it for this project. However, to give you a scale of how much work this all takes, consider that antivirus companies receive somewhere on the order of 250,000 samples of (possible) malware every day. We had you analyze 5 binaries. Imagine the types of systems needed to handle this amount of malware and study them thoroughly enough for that day, because the next day they're going to receive 250,000 new samples. If a malware analysis engine is unable to analyze a piece of malware within a day, they've already lost to malware authors. Also consider that not all of the 250,000 samples will be malicious. According to "Prudent Practices for Designing Malware Experiments: Status Quo and Outlook"(Rossow et al., 2012), as many as 3-30% may be benign!

Another way to look at the size issue of malware analysis, consider this paper “Needles in a Haystack: Mining Information from Public Dynamic Analysis Sandboxes for Malware Intelligence.”(Graziano et al., 2015) where the authors discovered that notorious malware samples had actually been submitted months, even years before the malware was detected and classified as malicious in the wild.

Remember, analyzing malware is a delicate and potentially dangerous act. Please be cautious and use good practices when analyzing malware in the future. If you let malware run for too long, you may be contributing to the problem and may be contacted by the FBI (and/or other authorities) as a result of this unintentional malicious contribution. At Georgia Tech, researchers, professors, and graduate students are able to analyze malware in controlled environments and have been given permission by the research community to perform these analyses long-term. We make efforts to contact the general research community and Georgia Tech's OIT Department to inform them that we are running malware, so they won't raise red flags if they detect malicious activity coming out of our analysis servers.

For your curious mind:

There is disagreement in the malware research community as to what exactly classifies malicious activity. For example, some say that adware is a form of malware, while others do not. Can you think of arguments for either side? Let's take this kind of thinking one step further. As a thought experiment, ask yourself this: If a piece of software has malicious code contained within it, but the malicious code is never executed when it is run, is/should that software be considered malicious? What if the malware author intentionally put in a buffer overflow vulnerability that allows someone to execute that malicious code? So, the only way of knowing the code can be executed is to exploit the malware. This seems like it would be a much more advanced form of trigger malware doesn’t it? Think of other tricks malware authors may employ to prevent researchers from discovering a malware’s true intentions.

Also, note the mutexes you found in the malware (the DDoS’ing mutexes). Did you notice the malware displaying any DDoS behaviors? There’s a reason you didn’t. It’s because these mutexes do not belong to these samples of malware. Essentially the malware source is trying to trick the (novice) malware analyst into thinking it’s one sample of malware, when it’s really another.

Be careful if you ever get your hands-on malware source code. We always make sure we read and fully understand malware source code before we compile and run it. Remember, safety is the number one priority in malware analysis.

If you're interested in reading more information about researching malware, we recommend you read "The Art of Computer Virus Research and Defense" by Peter Szor. It's known in the research community as a must-read for those interested in studying malware.

References

Rossow, C., Dietrich, C. J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., . . . Steen, M. V. (2012). Prudent practices for designing malware experiments: Status quo and outlook. 2012 IEEE Symposium on Security and Privacy. doi:10.1109/sp.2012.14
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6234405

Graziano, M., Canali, D., Bilge, L., Lanzi, A. & Balzarotti, D. (2015). Needles in a Haystack: Mining Information from Public Dynamic Analysis Sandboxes for Malware Intelligence.. In J. Jung & T. Holz (eds.), USENIX Security Symposium (p./pp. 1057-1072), : USENIX Association.
https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-graziano.pdf