# Project 2 : Malware Analysis

*Fall 2019*

## The goals of this project :

- This project is to get you familiar with the types of behaviors that can be displayed by malware and with how to safely analyze these behaviors.
- You will run malware through a secure environment to see what actions the malware will perform. In the real world, you would be looking at an unknown file to see what actions it takes.
- You will get hands on experience using a standard tool used in the industry called Cuckoo. For more details about Cuckoo you can visit http://www.cuckoosandbox.org and read more about it.
- You will learn what a safe environment to run malware looks like. In this case, you will have an outer VM where you will be doing all of your work, and an inner VM that will be used by Cuckoo to do the analysis. The outer VM is Ubuntu Linux, and the inner VM will be Windows XP.
- There will be NO report turned in and nothing submitted to Canvas for this assignment. Below you will be given a website where you will submit your answers; this website is the location from which your grade will be taken.
- While you are not required to do this, some students have submitted screen caps to Canvas of their answers from the website. That way they feel like they have some external confirmation of their responses. These would not be graded but could be used to verify submission results if you so desire. Again, this is completely optional.

## A Helpful Suggestion:

Make sure you have a main computer with sufficient horsepower to run the outer VM. Since you will be running a VM inside of a VM at the same time, you will find that 4 GB of RAM is probably insufficient to power the machines with any speed; you probably want 8 GB at minimum.

# Project Tasks (100 points):

## Setup (0 points)

1. Install VirtualBox on your own workstation. Please use at least version 5.0.26
2. Download the “CS6035 Project 2019_Summer_Main” OVA from https://tinyurl.com/y2vvyg6e. Please note that it is about 5 GB so it will take some time.
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.
    - Username: ubuntu
    - Password: 123456 (yes, we know, it is not secure and Spaceballs ridicules such a password. However, here we just want you to get in without any hangups)
4. Once you are into the VM, browse to /home/ubuntu/Desktop/malware/Phase1 and look at the contents (remember if you have a Windows background that Linux is case sensitive). Notice that there are four malware files. In this folder you will rename all four of these files, just for Phase 1 (subsequent files will NOT be renamed), as follows:
    - Add your userID (the one you use to log into Canvas) to the beginning of the malware sample’s name.
    - Example:

        - Canvas user ID is “edowning3”
        - Before renaming, the malware is called “malware1.exe”
        - After renaming, the malware is called “edowning3_malware1.exe”
    - Failure to follow this set of steps will cause you not to pass the Phase 1 portion of the assignment.
5. From an open terminal window, execute the following steps:

```
$ cd /home/ubuntu/Desktop/setup-folder
$ ./setup-cuckoo.sh
```

   This will install Cuckoo.

6. Import the Windows XP VM into the VirtualBox that is already installed inside the Ubuntu VM.
    - Start the VirtualBox GUI inside the VM.
    - Using the GUI, import the OVA in the VM
        - The OVA is located at /home/ubuntu/Desktop/setup-folder
        - If it asks you to upgrade VirtualBox inside the VM, say NO. Do NOT upgrade VirtualBox.
        - In the GUI go to the File menu and select “Import Appliances”
        - Follow the GUI instructions to import the OVA listed above.
        - Exit the VirtualBox GUI.
7. The terminal program in Ubuntu is tabbed. You will need three tabs for the following instructions, or else open three different Terminal windows. The important thing is for there to be three simultaneous terminals open. In tab one, start the Windows XP VM as follows:

```
$ VBoxHeadless --startvm "WindowsXPSP3"
```

8. In tab two, connect to the desktop of the Windows XP VM:

```
$ rdesktop 192.168.56.1:3389
```

9. This will open a VM window that you can work in. You want to verify that the VM starts cleanly and that there are no windows on the desktop
    - You will be in good shape on the verify if a command prompt appears stating “Starting agent on 0.0.0.0:8000…”
    - You MUST minimize this window (leave it open but not visible) in order for you to see the contents of the desktop for your Cuckoo runs.
        - Be warned, this is a tricky operation due to how the pointer appears
        - You may find it easier to put the pointer on the command prompt button in the taskbar, right clicking it, and choosing “Minimize”.
10. In tab three of the terminal window, execute the following commands:

```
$ cd /home/ubuntu/Desktop/setup-folder
$ VBoxManage snapshot "WindowsXPSP3" take "XP1" --pause
$ VBoxManage controlvm "WindowsXPSP3" poweroff
$ VBoxManage snapshot "WindowsXPSP3" restore "XP1"
$ ./setup-firewall.sh
$ mv ./cuckoo ../
```

    These commands will take a snapshot of the Windows XP VM that Cuckoo can use over and over to start from the same location. This process will cause the rdesktop window to disappear and close; this is normal and expected. The last two commands will set up the external firewall to our specs for Phase 1 and move the Cuckoo folder to its main location we will use.

11. IMPORTANT POST SETUP INSTRUCTIONS:

- If you have to turn off the Ubuntu VM and come back later that is OK. Just be sure to run these steps every time you boot up the VM:

```
$ cd /home/ubuntu/Desktop/setup-folder
$ ./config.sh
```

- DO NOT UPDATE ANY SOFTWARE IN THE VM. Doing so can cause it to break.
- Do not let any Cuckoo task run longer than the suggested 10 minutes
- There are two key folders to track for this project, “cuckoo” and “malware”. They are both on the Desktop after installation. The “cuckoo” folder contains the application and will do the work for you for the project. The “malware” folder contains the malware for the various phases of the project.

## Run the project website while working (0 points)

12. In order to submit your answers, you will need to log onto our homework submission website with account information that will be sent to you: http://35.161.124.173:4000/
13. You will be given a specific user name and password in your email. This is what you must use to log in.
14. In order for your answers for Phase 1 to be submitted, you MUST be running the web page and logged in as yourself WHILE Phase 1 is running (this will allow the system to connect as you to our back end and register your work). If you run Phase 1 without this website up and logged in, you will NOT get credit for it.
15. The main page on the website will update once you complete Phase 1 and have it registered successfully.
16. IMPORTANT: This process assumes that there will be no firewall blocking our activity. In our experience, this is not the case when people try to use their work laptops because their IT will lock down what can be sent. If this is the case, our team CANNOT make individual adjustments to make this work – it is the student’s responsibility to have a flexible system. Therefore we strongly urge you NOT to use a work machine to do this project.

## Phase One: Run the first set of malware correctly - (10 points)

**Note:** This task is to get your feet wet running Cuckoo. All you will be doing at this point is learning how to submit Cuckoo tasks and verifying that they succeed.

1. Open the Terminal program and make two tabs.
2. In tab one, start the cuckoo process and let it run in the background using these commands:

```
$ cd Desktop
$ ./cuckoo/cuckoo.py
```

3. In tab two, submit your malware for phase 1.
   - Use the following commands to submit your tasks:

```
$ cd Desktop
$ python ./cuckoo/utils/submit.py --timeout 600 ./malware/Phase1
```

   - This particular command tells Cuckoo to run all malware contained in the folder "./malware/Phase1" on a virtual machine (we only have one, so we don’t care about specifying this) for maximum time of 600 seconds (10 minutes).
   - Note that it may take Cuckoo a long time to analyze the data generated by the malware sample, so running all of the malware may take somewhere between 1 and 2 hours total.

4. To check up on its progress, look at the output generated in the first tab of Terminal (where Cuckoo is running). You will see some warnings every now and again. Don't worry, these warnings are by design. However, you should not see any errors.
5. Wait for Cuckoo to finish analyzing all 4 pieces of malware. The terminal will say "Task #4: analysis report completed".
6. In the second tab start the Cuckoo report web server.
    - Run the following command:

      ```
      $ python ./cuckoo/utils/web.py
      ```

    - This will start the web service for Cuckoo which will allow you to view the results of the Cuckoo analysis.
7. Open Firefox and navigate to the URL "localhost:8080"
8. On this webpage click "Browse" at the top of the page and you will see the results of Cuckoo's analysis organized nicely for you.
    - Expand the Ubuntu VM's screen size and Firefox's window size within the Ubuntu VM so that you can see the "Browse" link
9. Read and become familiar with these reports. You will use their contents later in PhaseII.
10. Not all malware will run for 10 minutes. There are various reasons for this but you only need to know that 10 minutes is a maximum duration, not a fixed duration.

## Phase Two: Analyze the first set of malware (40 points)

Now that you have a bit more experience with running malware, now it is your job to investigate and label some of the more sophisticated malware's behaviors from Phase I. Use the Cuckoo reports from Phase I label the malware's behavior. Note that malware can share the same 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 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, not necessarily succeeded) 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 currently, 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”.

BEHAVIORS: on the website for this project, please for each of the four malwares whether it demonstrates the following behaviors. Please note that there is no guarantee every behavior will be checked this particular term. You will be graded incorrect if you do not check a demonstrated behavior; you will be graded incorrect if you check a behavior that was not checked. Each malware is worth a total of 10 points for 40 points for all four.

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

Just check the box for the letter below if a particular malware shows the following behaviors:

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. Lowers Windows alert level for risky files that are downloaded or run (e.g., .exe, .bat, .vbs, etc.)
    a. Hint: https://support.microsoft.com/en-us/kb/883260 (look for the "Inclusion list for low, moderate, and high risk file types" section)

D. Displays message on Desktop to taunt user that they have been infected
    a. HINT – look at the screenshots
    b. Taunting = an actual message to the user; things like blank message boxes are NOT taunting

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

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

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

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

I. 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

J. Disables Command Prompt
    a. Hint: http://www.thewindowsclub.com/enable-disable-command-prompt-windows
    b. Hint: in earlier versions of Windows, the Command Prompt was also called the MSDOS Prompt

K. 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

L. 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

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

N. Hooks the mouse
    a. Hint: Similar to choice (M)

O. Potentially monitors messages before they appear in a window to the user (possible

reconnaissance)
   a. Hint: Similar to choices (M) and (N)

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

Q. Adds mutex for Eclipse DDoS malware
   a. Hint: https://www.mysonicwall.com/sonicalert/searchresults.aspx?ev=article&id=677

R. 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/

S. Adds mutex for DarkDDoSer malware
   a. Hint: Similar to choice (Q) and (R)

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

U. Copies potentially malicious files to the device

## Phase Three: Triggering dormant malware behavior (15 points)

You have been given 3 more pieces of malware in the folder "/home/ubuntu/Desktop/malware/Phase3". These malware will only activate if a certain condition is true. Your job is to find that particular condition via brute-force techniques. Use the same type of command for running malware using Cuckoo from PhaseI for this phase, substituting the correct location for the malware.

- Malware Eight
  - This malware's activity is triggered on some day in the month of March in the year 2004. Your job is to find what day it shows the most activity on. Submit your answer as a digit (e.g., 1, 2, 3, etc.).
  - Hint: Write a script that submits the malware at every day in the month at 2PM (because of some offset timings with the virtual machine). Letting the malware run for 180 seconds should be enough time to show its intended behavior. Brute-force the solution. To find out how to run the malware at a specific time, read Cuckoo's documentation (http://docs.cuckoosandbox.org/en/latest/).

- Malware Nine
  - This malware's activity is triggered after a certain amount of time has passed since it was executed. In essence, it delays its activities in order to evade analysis by malware analysis environments that employ fixed-time executions on its malware (which is a majority of malware analysis engines today).
  - Your job is to find out how many milliseconds the malware delays its **initial** activities. Submit your answer as a number (e.g., 1234, 234532, 352373, etc.)
  - Hint: Look at the system call sequence. What's the **first** attempt to delay execution at the start of the program?

- Malware Ten and Eleven

- Some malware won't show any behavior if they don't have certain filename(s). This is also an effort to evade detection, as malware researchers usually rename the malware for various reasons like keeping track of unique samples by the hashed value of their contents. Malware10 and malware11 are identical samples of the same malware, but with different filenames. Run malware10 and malware11 for 180 seconds each. Compare the Cuckoo reports:
  - Which malware has the proper trigger name (malware10.exe or malware11.exe)? (i.e., which executable shows more behavior?)
  - What extra file is dropped by the properly named malware (besides the malware itself)?
  - Looking at the screenshots generated by the two malwares. One of the malware let's you know the correct name is correct in a message, but the other one only states the malware is not correct. What is the message given by the incorrect malware? (**Please note: There will be no credit given for typos**)

## Phase Four: Analyzing steps taken to run malware safely (35 Points)

There are various ways to run malware in a safe environment. Running the malware inside of a virtual machine is a good start. That pretty much covers the system-side of things, but what about the network-side? We can use firewall rules in order to prohibit the malware from spreading throughout our network, sending spam, etc. We can even rate-limit the network connection (http://askubuntu.com/questions/20872/how-do-i-limit-internet-bandwidth) so that if a DDoS attack is used by our malware, we won't cause too much harm to the rest of the Internet. Your job is to read and interpret the firewall rules we've employed on our malware analysis system (the Ubuntu VM).

- Execute the following in a Ubuntu terminal:
  a. $ sudo iptables-save
- Read these rules outputted to the screen, read iptables documentation referenced below, and answer the questions below (Hint: Here is some good iptables material to read, as iptables can be difficult to read/understand and even more difficult to write properly.
  - https://en.wikipedia.org/wiki/Iptables
  - https://www.centos.org/docs/5/html/Deployment_Guide-en-US/ch-iptables.html
  - https://wiki.debian.org/iptables
  - http://www.howtogeek.com/177621/the-beginners-guide-to-iptables-the-linux-firewall/)

Questions:

1. What IP address CIDRs are not allowed to be communicated with by our malware?
   a. Hint: Cuckoo uses the IP addresses 192.168.56.1 and 192.168.56.101 to connect the malware to the Internet.
2. What IP address is all email traffic forwarded to?
3. Do the rules accept SSH connections? (yes or no)
4. Do the rules allow the analysis machine to be ping'd on the eth0 interface? (yes or no)
5. Why do the rules drop outbound connections to ports 135, 139, and 445? (Pick your letter answer from the choices below)
   A. They are primarily used by malware to send spam.
   B. They are primarily used by malware to propagate.
   C. They are primarily used by malware to launch DoS attacks.
   D. They are primarily used by malware to detect themselves being analyzed.
   - Hint: Google these port numbers. They are well-known to be used by Windows malware.
   - Hint2: http://www.berghel.net/col-edit/digital_village/dec-05/dv_12-05.php

## The final deliverables:

- As stated above, the only deliverable for this project are the submissions on your website responses.

## Reflection:

Well cool. Now you've got some experience under your belt with analyzing malware. 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 didn'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. We had you analyze 7 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 [1], as many as 3-30% may be benign!

Another way to look at the size issue of malware analysis, consider this paper [2] 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 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.

References:

[1] Rossow, Christian, Christian J. Dietrich, Chris Grier, Christian Kreibich, Vern Paxson, Norbert Pohlmann, Herbert Bos, and Maarten Van Steen. "Prudent Practices for Designing Malware Experiments: Status Quo and Outlook." In Security and Privacy (SP), 2012 IEEE Symposium on, 65–79. IEEE, 2012. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6234405.

[2] Graziano, Mariano, Davide Canali, Leyla Bilge, Andrea Lanzi, and Davide Balzarotti. “Needles in a Haystack: Mining Information from Public Dynamic Analysis Sandboxes for Malware Intelligence.” In 24th USENIX Security Symposium (USENIX Security 15). USENIX Association. Accessed September 23, 2015. https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-graziano.pdf.

## For your curious mind:

In PhaseI, all of these malware are slightly modified real-world samples. We modified them to prevent them from spamming and DDoS’ing and performing other nefarious network activities. We did this for safety concerns. Some of them, you'll notice, aren't even detected/classified as malicious by all or a majority of antivirus companies. This is because either (1) the samples are still too new and are still being studied and classified by antivirus companies and malware research facilities or (2) the samples are classified differently between antivirus companies. Truly 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.

In PhaseIII, we modified real-world malware source code to create real triggers seen in other real-world malware. We designed it this way because it's nicer if we can control and determine the malware's behavior by modifying its source.

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.