Malware Analysis Project

Disclosure:

We are always looking to improve our homework assignments. If you see any errors, whether they are grammatical or technical, please email the TAs to report them. If anything is unclearly stated, please contact the TAs.

Purpose:

The purpose of this assignment is to have you gain experience with running malware through an analysis engine and perform investigations on a malware's behaviors. You will be running malware through an analysis engine called Cuckoo (http://www.cuckoosandbox.org/). You will learn how to use Cuckoo and how to run malware ***safely***. There is no sense in studying malicious behavior if you're going to contribute to the problem.

Grading:

Your score for this project will be out of 100 points. The points are allocated as follows: Phases I (10 points), Phase II (40 points), Phase III (15), Phase IV (35).

Setup:

1. Install VirtualBox (at least version 5.0.26)
2. Download Project2.ova from evan.gtisc.gatech.edu/cs6035/project2/Project2.ova. It is about 4GB in size, so it may take some time.
3. Import the Project2.ova file to VirtualBox
    a. Username: ubuntu
    b. Password: 123456
4. Install Guest additions
    a. In VirtualBox menu: Devices->Insert Guest Additions CD image
    b. Follow the instructions that appear on the Ubuntu screen
    c. You will need to be able to resize your window to properly read the malware reports generated by Cuckoo.
5. Shutdown the Ubuntu VM and start it again. If you simply restart the VM, the changes of the Guest Additions may not take complete effect.
6. In /home/ubuntu/Desktop/malware/Phase1, rename each malware sample as such:
    a. Add your user ID (as seen by T-Square) to the beginning of the malware sample's name.
    b. Example:
        i. Georgia Tech user ID: "edowning3"
        ii. Before renaming: "malware1.exe"
        iii. After renaming: "edowning3_malware1.exe"
7. Follow the remaining instructions in the README file in /home/ubuntu/Desktop/setup-folder to setup Cuckoo and configure the analysis environment.

Important:

1. If you shutdown or restart the VM, execute this script after the VM has finished booting:

    a. $ cd /home/ubuntu/Desktop/setup-folder
    b. $ ./config.sh
2. There are two key folders you will be using for this entire project: "cuckoo" and "malware." Both folders are located on the Desktop of your user. The folder "cuckoo" contains the Cuckoo software and will be responsible for submitting and analyzing our malware. The folder "malware" contains malware for each Phase of this project.
3. Refrain from updating **any** software inside the virtual machine. We have configured it with precise versions and if one is updated/upgraded the system **will** break.
4. Do not run the malware for more than the suggested 10 minutes.

Submitting answers:

In order to submit your answers, you will need to log onto our homework submission website: http://malaise.gtisc.gatech.edu:3000

You **must** login before performing the Phase I operations or your execution of Phase I will not be recorded.

Log on using your Georgia Tech email and the password that was emailed to you.

Use the submission forms to (1) check that Phase I has recorded that you have successfully run the malware and (2) answer questions from the rest of the phases.

Phase I [10 points]: Learn how to run malware using Cuckoo and get familiar with reading its reports

1. Open Terminal
2. Open two tabs in this Terminal
3. In one tab, start Cuckoo:
    a. $ cd Desktop
    b. $ ./cuckoo/cuckoo.py
4. In the second tab, submit your pieces of malware
    a. $ cd Desktop
    b. $ python ./cuckoo/utils/submit.py --timeout 600 ./malware/Phase1
5. This particular command tells Cuckoo to run all malware contained in the folder "./malware/PhaseI" 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.**
6. 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.
7. Wait for Cuckoo to finish analyzing all 4 pieces of malware. The terminal will say "Task #4: analysis report completed".
8. In the second tab in Terminal, execute:

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

9. Open Firefox and navigate to the URL "localhost:8080"
10. 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.
11. Read and become familiar with these reports. You will use their contents later in PhaseII.

You will note that not all malware may actually run for the full 10 minutes. This can be for various reasons: the Command & Control (C&C) server decides that the malware is being analyzed and does not want it to be run anymore, the malware decides to exit because it has completed its nefarious task and does not need to run any more, etc.

Phase II [40 points]: Learn how identify different behaviors in malware

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 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 in **any process** in the report, then that malware has 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.

Questions:

List behaviors (just the letters representing the behaviors) for:

Malware1, Malware2, Malware3, Malware4

Choose your answer(s) from one or more of the following choices:

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

B. Checks the computer's name (possibly doing reconnaissance)
   - Hint: http://www.windowstipspage.com/computername-registry-key/

C. Lowers Windows alert level for risky files that are downloaded or run (e.g., .exe, .bat, .vbs, etc.)
   - 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

- Hint: Look at the screenshots.

E. Potentially looks through Microsoft Outlook address book contents
   - 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
   - Hint: http://www.thewindowsclub.com/prevent-access-to-registry-editor-windows

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

I. Prevents users from changing remote administrator settings
   - Hint: http://www.pctools.com/guides/registry/detail/183/

J. Disables Command Prompt
   - Hint: http://www.pctools.com/guides/registry/detail/118/

K. Searches for all possible drives on computer
   - Hint: Look for the malware attempting to open each drive.

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)
   - Hint: https://msdn.microsoft.com/en-us/library/windows/desktop/aa379180(v=vs.85).aspx

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

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

O. Potentially monitors messages before they appear in a window to the user (possible reconnaissance)
   - Hint: Similar to choices (M) and (N)

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

Q. Adds mutex for Eclipse DDoS malware

- Hint: https://www.mysonicwall.com/sonicalert/searchresults.aspx?ev=article&id=677

R. Adds mutex for IPKillerClient malware
   - Hint: Look under “Created Mutexes” section of “Behavioral Information” : https://www.virustotal.com/en/file/c96015a878d476197770f656d8efae559f78e0cfb9a5facecad8c3ffa2a4ec30/analysis/

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

T. Contacts various SMTP servers (possibly for spamming)
   - Hint: It contacts multiple “smtp.*” domains

U. Shows antisandbox behaviors
   - Hint: Particularly pay attention to lines 85 and the function IsRunningInSandbox() https://github.com/3val/Athena/blob/master/Source%20-%20Bot/Source/Protection/AntiDebugEmulate.cpp

Phase III [15 points]: Learn how to trigger dormant malware behavior

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 workflow for running malware using Cuckoo from PhaseI for this phase.

Questions:

Malware8:

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/).

Malware9:

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?

Malware10 and Malware11:

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. Run malware10 and malware11 for 180 seconds each. Compare the Cuckoo reports:

1. Which malware has the proper trigger name (malware10.exe or malware11.exe)? (i.e., which executable shows more behavior?)
2. What extra file is **dropped** by the properly named malware (besides the malware itself)?
3. Looking at the screenshot generated by the **improperly** named malware. What is the message that is displayed on the screen?

Phase IV [35 points]: Learn how malware is run safely

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 command in a Ubuntu Terminal:

- $ 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

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.