

CMSC 491/691 Malware Analysis HW 7

Name: Daniel Roh

Assigned: 4/29/2020

Due: 5/11/2020 by 5:00pm

Download hw7_dataset.7z and extract it. The password is "infected". You may also consider downloading the dataset to a Linux box if you want to use the bash command for finding shared strings.

Instructions for installing YARA on your Flare VM:

- Download YARA 3.11.0 from <https://github.com/VirusTotal/yara/releases/download/v3.11.0/yara-v3.11.0-994-win64.zip>
- Add the folder containing yara64.exe to your \$PATH environment variable.

YARA documentation:

- <https://yara.readthedocs.io/en/v3.11.0/gettingstarted.html>
- <https://yara.readthedocs.io/en/v3.11.0/writingrules.html>
- <https://yara.readthedocs.io/en/v3.11.0/modules/pe.html>

Helpful bash command:

```
for filepath in ramnit_*; do strings $filepath | sort | uniq; done | sort |  
uniq -c | sort -nr | less
```

Example YARA rule for the ramnit family:

```
import "pe"
```

```
rule Ramnit {  
  strings:  
    $s1 = "KyUffThOkYwRRtgPP" fullword ascii  
    $s2 = "Srv.exe" fullword ascii  
    $b1 = { 60 E8 00 00 00 00 5D 8B C5 81 ED ?? ?? ?? ?? 2B }  
  
  condition:  
    uint16(0) == 0x5a4d and  
    all of ($s*) and  
    $b1 at pe.entry_point and  
    pe.sections[pe.number_of_sections - 1].name == ".rmnet"  
}
```

Part 1: Writing a YARA rule for the XtremeRAT family (27 pts)

Write a YARA rule for the XtremeRAT family that:

- Matches all ten XtremeRAT malware samples in hw7_infected
- Does not match any other files in hw7_infected or hw7_benign
- Checks that the file is a PE file
- Contains at least five non-generic strings
- Uses at least three of the following string modifiers: nocase, wide, ascii, fullword

1) Provide the YARA rule below. (10 pts)

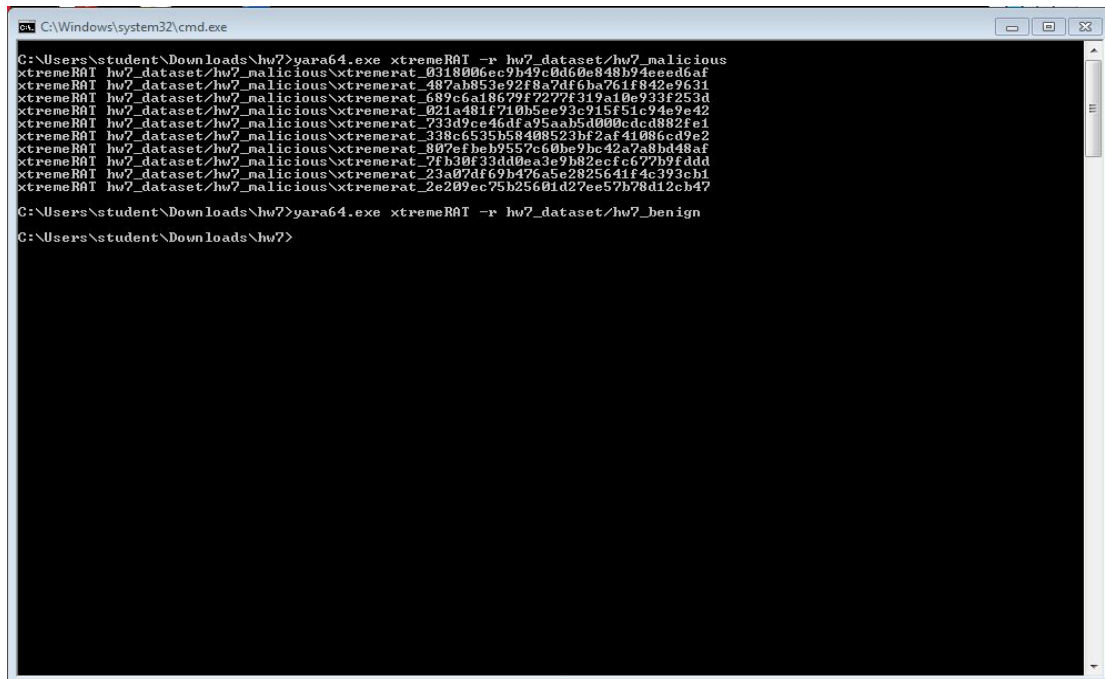
```
rule xtremeRAT {
  meta:
    description = "HW7 part 1 XtremeRat"

  strings:
    $s1 = "XTREME" wide
    $s2 = "XtremeKeylogger" wide
    $s3 = "XTREMEBINDER" wide
    $s4 = "Portions Copyright (c) 1999,2003 Avenger by NhT" fullword ascii
    $s5 = "ShellExecute" ascii

  condition:
    //Check if PE file
    uint16(0) == 0x5a4d and
    uint32(uint32(0x3C)) == 0x00004550 and

    //Check Strings
    $s1 or $s2 or $s3 and
    $s4 and $s5
}
```

2) Provide screenshots showing the output of scanning hw7_infected and hw7_benign using the XtremeRat YARA rule (10 pts)



```
C:\Windows\system32\cmd.exe
C:\Users\student\Downloads\hw7>yara64.exe xtremeRAT -r hw7_dataset\hw7_malicious
xtremeRAT hw7_dataset\hw7_malicious\xtremeRat_0318006ec9b49c0d60e848b94eed6af
xtremeRAT hw7_dataset\hw7_malicious\xtremeRat_487ab853e92f8a7df5ba761f842e7631
xtremeRAT hw7_dataset\hw7_malicious\xtremeRat_689c6a18679f7277f319a10e933f253d
xtremeRAT hw7_dataset\hw7_malicious\xtremeRat_021a481f710b5ee93c915f51c94e9e42
xtremeRAT hw7_dataset\hw7_malicious\xtremeRat_733d9ce46dfa95aab5d000cdcd882fe1
xtremeRAT hw7_dataset\hw7_malicious\xtremeRat_338c6535b58408523bf2af41086cd9e2
xtremeRAT hw7_dataset\hw7_malicious\xtremeRat_807efbcb9557c60be7bc42a7a8bd48af
xtremeRAT hw7_dataset\hw7_malicious\xtremeRat_7eb30f33dd0ea3e9b2ecf6c77b9fddd
xtremeRAT hw7_dataset\hw7_malicious\xtremeRat_23a07df67b476a5e2825641f4c393cb1
xtremeRAT hw7_dataset\hw7_malicious\xtremeRat_2e209ec75b25601d27ee57b78d12cb47
C:\Users\student\Downloads\hw7>yara64.exe xtremeRAT -r hw7_dataset\hw7_benign
C:\Users\student\Downloads\hw7>
```

3) **In a few sentences, describe your design choices for the YARA rule. (7 pts)**

When designing the YARA rule, I started by checking if the file is a PE file. As it makes sense that you would want to avoid detecting files which are not even an executable. From there, after taking a look at some of the samples. After looking at some of the strings contained in the samples of the xtremeRAT family, I found what looked to be a copyright info which also seemed to appear in some of the other samples. So I added this as a rule and was able to get a few matches. Looking around some more several unicode strings which seemed to be labels with "xtreme" in front. So I found a few of these and added them. After some minor changes to how the conditions are being ordered, I was able to detect all 10 samples of XtremeRat.

Part 2: Writing a YARA rule for the Poison Ivy family (27 pts)

Write a YARA rule for the Poison Ivy family that:

- Matches all **ten** Poison Ivy malware samples in hw7_infected
- Does not match any other files in hw7_infected or hw7_benign
- Checks that the file is a PE file
- Contains at least **two** non-generic strings
- Contains at least **one** byte sequence
- Uses at least **two** of the following modifiers: nocase, wide, ascii, fullword
- Uses the "at" and "pe.entry_point" keywords

4) Provide the YARA rule below. (10 pts)

```
import "pe"

rule PoisonIvy {
    meta:
        description = "HW7 part 2 Poison Ivy"

    strings:
        $s1 = "StubPath" fullword ascii
        $s2 = "advapi32" fullword ascii
        $s3 = "advpack" fullword ascii
        $s4 = "CONNECT %s:%i HTTP/1.0" fullword ascii

        $b1 = {A2 CA 81 7C 00 00 00 00 B8 00 04 40 00 FF D0 6A}
        $b2 = {53 4F 46 54 57 41 52 45 5C 43 6C 61 73 73 65 73 5C 68 74 74 70 5C 73 68 65 6C 6C 5C 6F 70
65 6E 5C 63 6F 6D 6D 61 6E 64 56 04 35 00 53 6F 66 74 77 61 72 65 5C 4D 69 63 72 6F 73 6F 66 74 5C 41 63
74 69 76 65 20 53 65 74 75 70 5C 49 6E 73 74 61 6C 6C 65 64 20 43 6F 6D 70 6F 6E 65 6E 74 73 5C}

        $h1 = {B8 00 04 40}

    condition:
        //Check if PE file
        uint16(0) == 0x5a4d and
        uint32(uint32(0x3C)) == 0x00004550 and

        //Strings to check
        all of ($s*) and
        ($b1 or $b2) and
        $h1 at pe.entry_point
}
```

5) Provide screenshots showing the output of scanning hw7_infected and hw7_benign using the Poison Ivy YARA rule (10 pts)

```
C:\Windows\system32\cmd.exe

C:\Users\student\Downloads\hw7>yara64.exe poisonIvy -r hw7_dataset\hw7_malicious
PoisonIvy hw7_dataset\hw7_malicious\poisonIvy_1e2ccc637c61c7f98d8b60721a42ef73
PoisonIvy hw7_dataset\hw7_malicious\poisonIvy_41f75a2fae5753171c75a3e6fd4df8ad
PoisonIvy hw7_dataset\hw7_malicious\poisonIvy_5ecf3a577e94d4514cb300e2ad055Feb
PoisonIvy hw7_dataset\hw7_malicious\poisonIvy_740d3fdce665ed6667587eb2b357a949
PoisonIvy hw7_dataset\hw7_malicious\poisonIvy_c341c2fa49f86bcfe498b8184af581dc
PoisonIvy hw7_dataset\hw7_malicious\poisonIvy_f6de36b1797bac4bd645b449de743ca9
PoisonIvy hw7_dataset\hw7_malicious\poisonIvy_98b4e5f127a62b31bd4fc3e92288ae53
PoisonIvy hw7_dataset\hw7_malicious\poisonIvy_030a0e03af51c53eb7c97395ad27a994
PoisonIvy hw7_dataset\hw7_malicious\poisonIvy_545e3ed00a18dabbe3bd18ad45e61c0
PoisonIvy hw7_dataset\hw7_malicious\poisonIvy_52eca5ea5609d78604ae6b149438c28c

C:\Users\student\Downloads\hw7>yara64.exe poisonIvy -r hw7_dataset\hw7_benign
C:\Users\student\Downloads\hw7>
```

6) In a few sentences, describe your design choices for the YARA rule. (7 pts)

Looking at the samples with the string command, I was able to find a string "StubPath", "advpack" and "advapi32" which was in each of the poison ivy samples. After adding this rule, I was able to get all of the poison ivy samples detected, but also resulted in getting a few other virus families detected as well. After looking more at the strings, I noted that all the samples seem to make a change to the registry to allow auto run at start up. When looking for the hex dump to find the hex address for the registry, a string in the hex dump that did not get caught in the string dump command I made caught my eye. The string "CONNECT %s:%i HTTP/1.0" which I was also able to find in the other ivy files. After reaching no result for a few hours, I contacted Ana in which she told me I was using an outdated database file (T.T) and gave me resources to figure out how to use the pe.entry_point. Thanks Ana :P

After downloading the proper database, I re-ran the rules and was able to get all 10 ivy samples with no other families. Given that we were also required to add a byte string into the yara rules. I found a common registry key that poison ivy uses and added that to the rules. Also, using the entry point information from DiE, took a look in the hexdump of some of the poison ivy files and found that the files all had a common starting hex of "B8 00 04 40". Which I also added into the yara rules. Interestingly, after finishing this, I noticed that even with just using the strings or byte sequence, or pe.entry_point alone, I can detect all 10 samples.

Part 3: Writing a YARA rule for heuristic malware detection (46 pts)

As described in the paper “Attributes of Malicious Files”

(<https://www.sans.org/reading-room/whitepapers/malicious/attributes-malicious-files-33979>), unusual PE metadata is a very useful indicator that a file may be malicious. In Part 3, you will write a YARA rule for detecting malware based on the indicators listed in this paper. You will evaluate the precision, recall, and F1 measure of your YARA rule using hw7_infected and hw7_benign. The student whose YARA rule has the highest F1 Measure in the class will receive 10 extra credit points. Students in the top ten will receive 5 extra credit points. To prevent you from gaming the system (and to make sure your YARA rule can generalize), F1 measure for the extra credit will be computed using a different dataset than the one provided.

Write a YARA rule that:

- Matches as many files in hw7_infected as possible
- Matches as few files in hw7_benign as possible
- Checks that the file is a PE file
- Has at least five conditions based on indicators from the “Attributes of Malicious Files” paper
- You may include anything else in your YARA rule that you wish
- The name of your YARA rule must include your last name to be eligible for extra credit

7) Provide the YARA rule below as text (12 pts)

```
import "pe"
import "math"

rule malwareFinder_Roh {
  meta:
    description = "HW7 part 3 Find all the Malware"

  condition:
    //Check if PE file
    uint16(0) == 0x5a4d and
    uint32(uint32(0x3C)) == 0x00004550 and

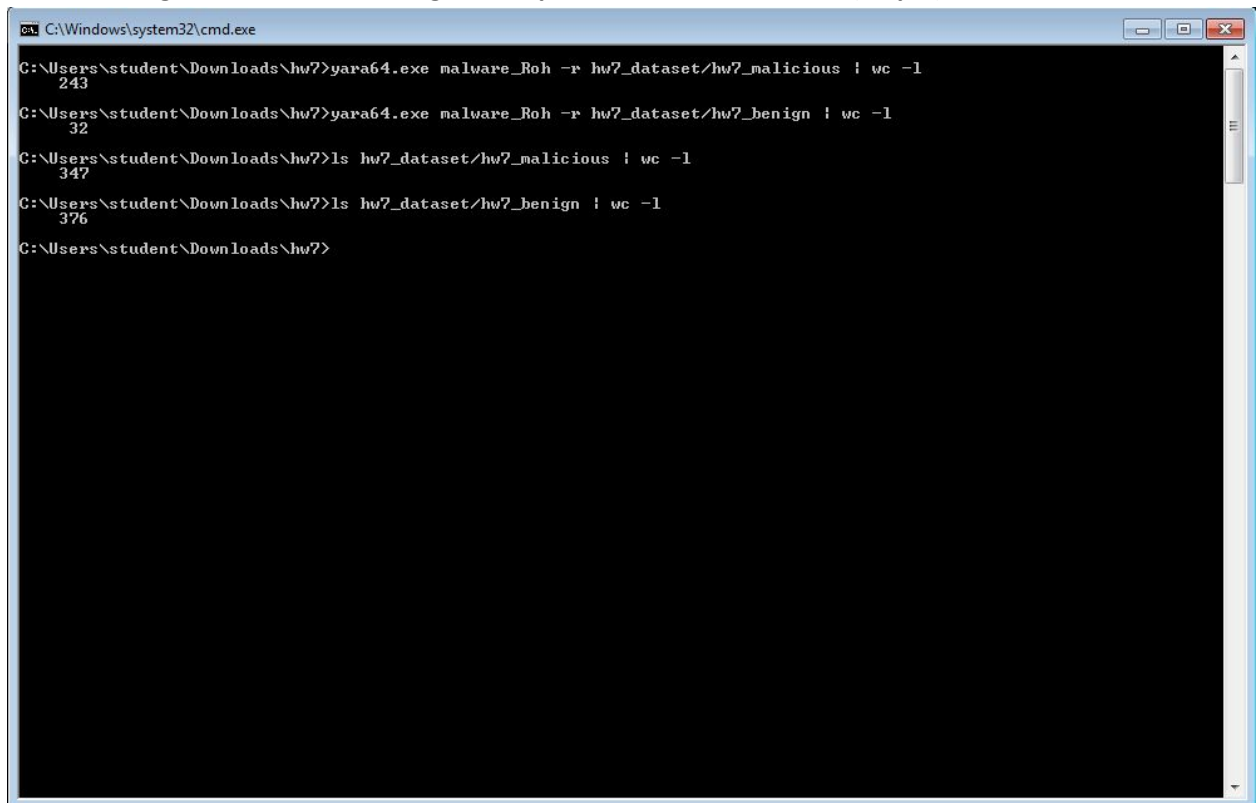
    (pe.number_of_sections < 1 and pe.number_of_sections > 9 ) or //Check for number of sections
    math.entropy(0, pe.size_of_image) >= 6.9 or //Find entropy of whole file

    for any i in (0..pe.number_of_sections -1): //Check entropy/size of each section
      math.entropy(pe.sections[i].raw_data_offset, pe.sections[i].raw_data_size) >= 7 or
      pe.sections[i].raw_data_size == 0
    ) or
    pe.number_of_rva_and_sizes != 16 or //Look optional header entries that are set
    pe.loader_flags != 0 //Check for flags set in the Loader
}
```

Run the following commands on your Flare VM:

- `yara64 -r [YARA rule] [path to hw7_malicious] | wc -l`
- `yara64 -r [YARA rule] [path to hw7_benign] | wc -l`
- `ls [path to hw7_malicious] | wc -l`
- `ls [path to hw7_benign] | wc -l`

- 8) Provide a single screenshot showing the output of all four commands (10 pts)



```
C:\Windows\system32\cmd.exe

C:\Users\student\Downloads\hw7>yara64.exe malware_Roh -r hw7_dataset\hw7_malicious ! wc -l
243

C:\Users\student\Downloads\hw7>yara64.exe malware_Roh -r hw7_dataset\hw7_benign ! wc -l
32

C:\Users\student\Downloads\hw7>ls hw7_dataset\hw7_malicious ! wc -l
347

C:\Users\student\Downloads\hw7>ls hw7_dataset\hw7_benign ! wc -l
376

C:\Users\student\Downloads\hw7>
```

- 9) In a few sentences, describe your design choices for the YARA rule. (7 pts)

To start, I added the rule that checks if the file is an PE file as it wouldn't make sense to detect non-executables. From there, for fun, I added a string "malware" and "virus" to see if any malware would actually reference its ability. To my surprise, one actually appeared. After taking a look at the file and finding out why it got detected. I decided that these strings would not be an ideal choice for detecting malware. As a string of say anti-malware services mentioning how good of a job it's doing would detect it.

Looking at the linked paper, There are various methods of detecting malware. So I started by going after methods that matched with what was discussed in class. I started by looking for individual entropies larger than 7, which was a common method of telling a potential malware in past HW's. Then I added a check to look for the amount of sections found in a file, as the paper notes this as giving way to increase detection without adding false positives.

Then, I looked at each section entropy of the entire file to see if it was larger than 6.9. As if the case that one section happens to have high entropy, there's a chance it may not get caught by checking the entropy of the whole file alone. While this is happening I decided to check the raw data size of each section as a packed file can result in a section reporting a data size of 0.

Next I took a look at the RVA as it was one of the things in the paper which resulted in a 0% false positive rate. I added this rule in an attempt to avoid getting more false positives, while gaining more true positives. Also, the paper noted to also check loader flags as the flags are not commonly used in regular programs and maybe set in mal-programs.

10) Answer the following: (8 pts)

- a. How many true positives (TP) did your YARA rule have? 243
- b. How many true negatives (TN) did your YARA rule have? 344
- c. How many false positives (FP) did your YARA rule have? 32
- d. How many false negatives (FN) did your YARA rule have? 104

$$Precision = \frac{TP}{TP+FP} \quad Recall = \frac{TP}{TP+FN}$$

TP = True Positive (Malware that is malware)

TN = True Negative (Not detected non malware)

FN = False Negative (Malware that was not caught)

FP = False Positive (Detected Malware that is not malware)

$$F1 \text{ measure} = \frac{2*Precision*Recall}{Precision+Recall}$$

11) Use the formulas above to answer the following: (9 pts)

- a. What is the precision of your YARA rule? 0.883636
- b. What is the recall of your YARA rule? 0.700288
- c. What is the F1 measure of your YARA rule? 0.781350