

Lab 5: Basic PE Static Analysis

ITSC 303: Malware Analysis

Table of Contents

[Lab Outcomes 2](#_Toc493062953)

[Background Reading 2](#_Toc493062954)

[Introduction 2](#_Toc493062955)

[1.0 Clam AntiVirus (ClamAV) 3](#_Toc493062956)

[1.1 Getting Started 3](#_Toc493062957)

[1.2 Full File Hash Signatures 4](#_Toc493062958)

[1.3 PE Section Signatures 6](#_Toc493062959)

[1.4 Body-Based Signatures 10](#_Toc493062960)

[1.5 Opcode Signatures 14](#_Toc493062961)

[1.6 Logical Signatures 17](#_Toc493062962)

[2.0 Writing Detections 19](#_Toc493062963)

[2.1 Full File Hash Detection 19](#_Toc493062964)

[2.2 PE Section Hash Detection 19](#_Toc493062965)

[2.3 Body-Based Detection 19](#_Toc493062966)

[References 22](#_Toc493062967)

Malware Analysis

Lab 5: Basic PE Static Analysis

Lab Outcomes

This lab will focus on the following outcomes:

* Use skills from previous labs.
* Become familiar with ClamAV.
* Create ClamAV file hash signatures.
* Create ClamAV PE section hash signatures.
* Create ClamAV body-based signatures.

Background Reading

* ClamAV documentation:
  + [Signatures](https://github.com/vrtadmin/clamav-devel/blob/master/docs/signatures.pdf) (https://github.com/vrtadmin/clamav-devel/blob/master/docs/  
    signatures.pdf)
  + [General](https://github.com/vrtadmin/clamav-devel/blob/master/docs/clamdoc.pdf) (https://github.com/vrtadmin/clamav-devel/blob/master/docs/  
    clamdoc.pdf)

Introduction

In previous labs, you learned how to identify and gather information about PE files, as well as reverse engineer PE files to verify their functionality. In this lab, you will scan for and identify malicious PE files. Software that scans for and identifies malicious software are typically referred to as *antivirus scanners* or *antivirus software* and can scan a computer’s hard drive looking for malicious files. The antivirus scanner relies on a constantly updated malware signature database in order to identify new or modified pieces of malicious software.

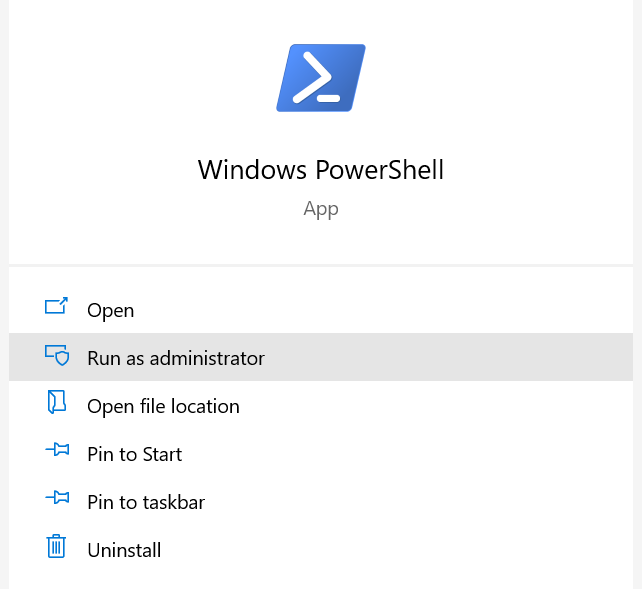
Many commercial antivirus scanners exist and although they are effective they don’t always allow users to easily add detections to their database of malicious file signatures. They may have methods to submit the signatures to their central database, which would ultimately find its way to your environment. ClamAV, on the other hand, is a free, open-source antivirus scanner that allows end users to create their own malware signatures. Many malware authors will also test their binaries against known AV tools, especially if they’re aware of the tools configured within your corporate environment.

This lab will use ClamAV to create detections for malicious PE files you have previously worked with. The lab will introduce file hash based signatures, PE section hash signatures and body‑based signatures. At the end of the lab you will know how to create all three types of signatures.

1. Clam AntiVirus (ClamAV)

ClamAV is an open-source, multi-platform antivirus scanning engine that provides antivirus signatures that are written and distributed by paid employees of the ClamAV team. In addition, ClamAV allows users to create their own custom malware detection signatures. Your VM environments have ClamAV available in the C:\tools\static-analysis\AV location however it is an older version of the toolset, so consider downloading the latest version at http://www.clamav.net/downloads/production/ClamAV-0.101.3.exe

* Download and install this executable onto your analysis machines, be sure to install as administrator
* Once the install is complete, open an administrator level Powershell window
  + Windows Key then type Powershell, but DO NOT press enter
  + Select run as administrator



* Inside your Powershell window, type the following commands:
  + cd "c:\program files\clamav"
  + copy .\conf\_examples\freshclam.conf.sample .\freshclam.conf
  + write.exe .\freshclam.conf
* This will open a Wordpad document, delete the line that says example, then save and close.

**\*A good tip, this lab works better if you run the clam tools from their install location (C:\Program Files\ClamAV”) and point the Clam tools to the complete paths for your signature files and samples to scan.\***

## 1.1 Getting Started

1. Using a command shell, navigate to **c:\temp\lab5**. Create the directory if required.
2. Execute the following command (your path may be **c:\Program Files\ClamAV**).

> dir "c:\Program Files\ClamAV"

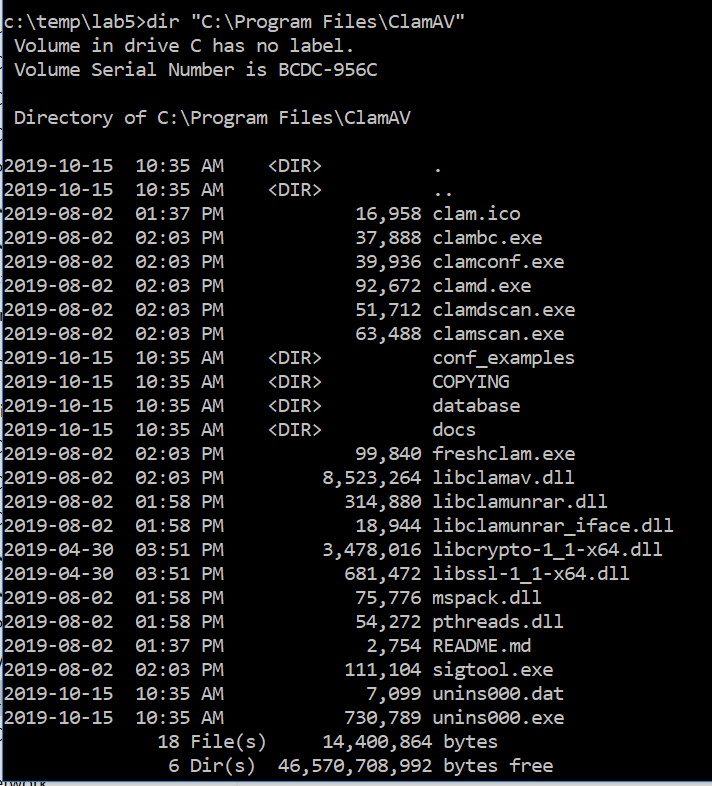


Figure 1: ClamAV Directory Listing

Used with permission from Microsoft.

Although, there are a number of executable files provided in the ClamAV home directory, you will use clamscan.exe to scan suspected malware files. Clamscan.exe can use signature databases downloaded from ClamAV or custom databases created by the user. In this lab you will create a custom database.

1. Inside your C:\temp\lab5 directory, create a folder named **to\_scan** (you will place files you want to scan in this folder).
2. In the lab5 folder, create an empty file named **lab5\_signatures.hdb**. To create this file, open notepad as administrator on your machine, and save the file as **lab5\_signatures.hdb** in the location.

Your folder should look like Figure 2.

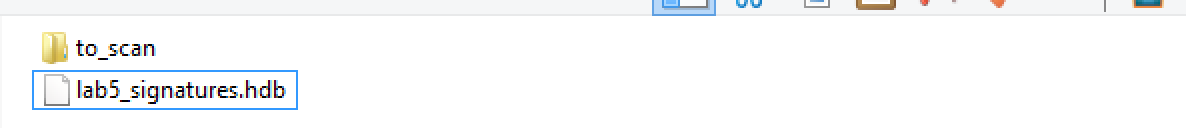


Figure 2: Prepared Folder

Used with permission from Microsoft.

1. Copy **WinDiagService.exe** to the *to\_scan* folder (you reverse engineered this malware sample in previous weeks).

## 1.2 Full File Hash Signatures

The first signature you will create is a hash-based signature. Start by calculating the SHA-256 of the sample you want to detect. In an earlier lab you calculated the SHA-256 hash of a file using a program called sha256sum.exe, and you created a Python file called sha256sum.py to calculate the SHA-256 hash of any file.

1. Use any method to calculate the SHA-256 hash of the file.

The example in Figure 3 demonstrates calculating the SHA-256 hash using the Sysinternals tool sigcheck.exe with the -h: option. Sysinternals is available at C:\tools\static-analysis\digital-signatures



Figure 3: Using sigcheck.exe to Calculate the SHA-256 Hash

Used with permission from Microsoft.

A ClamAV hash signature is defined as a single line in a text file with the following format:

HashString:FileSize:MalwareName

The signature has three values delimited by a colon (**:**).

* The first value is the hash string, which you calculated above.
* The second is the file size in bytes. File size is included so ClamAV can avoid calculating the SHA-256 hash for any file that doesn’t match the size, making ClamAV more efficient.
* The third value is the malware name, which is selected by the user.

1. To define the first signature, open **lab5\_signatures.hdb** in Notepad and enter the following line:

3094A3DB6068BE414FD060C2CFD4B10A81961D349393FE6E07A65423D160729C:77824:Hash.Downloader.Arud

1. Find the file size by right-clicking **WinDiagService.exe** and selecting **Properties**. The name is up to you.

The example uses the name Hash.Downloader.Arud.

**Notes:**

* It is traditional to use a period (**.**) as a separator in malware names.
* The string “Hash” indicates that the signature is based on a hash.
* The string “Downloader” indicates the sample’s category.
* The string “Arud” was gathered from the malware’s hardcoded command and control domain, hxxp://aruddheksl.ru/news.php. Names like Arud are up to the malware analyst and are often chosen from strings in the malware sample.

1. Save the file **lab5\_signatures.hdb**.
2. Open a command window at the install location for your Clam instance (default C:\Program Files\ClamAV)
3. To scan a single file, execute the following command:

> clamscan.exe -d <full path to>\lab5\_signatures.hdb <full path to>\to\_scan\WinDiagService.exe

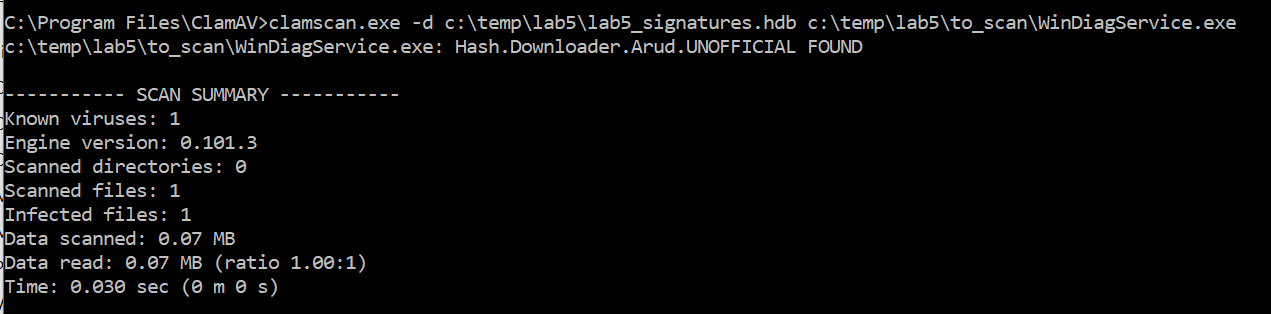


Figure 4: Using Clamscan.exe with custom database

Used with permission from Microsoft.

Clamscan.exe displays that the file was detected, and the string UNOFFICIAL is appended to the malware name to indicate that the signature is created by the end user.

## 1.3 PE Section Signatures

To demonstrate the limitations of hash-based signatures you will make an inconsequential change to WinDiagService.exe.

1. Launch a hex editor (FlexHex, Hexplorer, HxD etc) from your desktop or the Start menu.
2. Select **File > Open**, navigate to **C:\temp\lab5\to\_scan\** and open **WinDiagService.exe**.
3. Make a small, one byte change.

Figure 5 shows changing **p** at offset 0x53 to **P**.

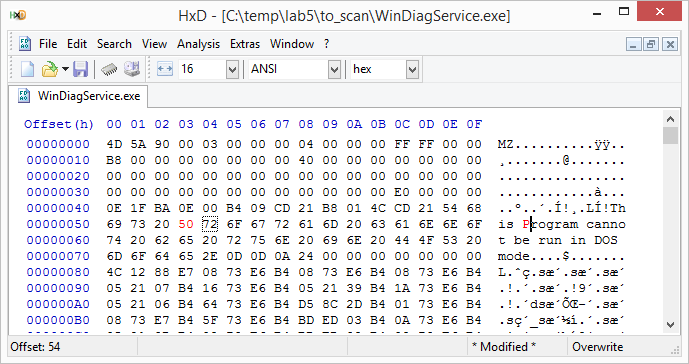


Figure 5: HxD with Single Byte Change

Source: HxD, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

1. When complete, select **File > Save As** and save the file as **WinDiagService\_2.exe**.
   * You may need to specify the extension as .exe when saving depending on the tool used
2. Using sigcheck.exe, sha256sum.exe or any other method, ensure that the SHA-256 of WinDiagService.exe and WinDiagService\_2.exe do not match.
3. To scan the entire to\_scan directory, use the following command:

> clamscan -d <full path to>\lab5\_signatures.hdb <full path to>\to\_scan\

Figure 6 shows that an inconsequential change to a malware sample can defeat hash-based signatures.

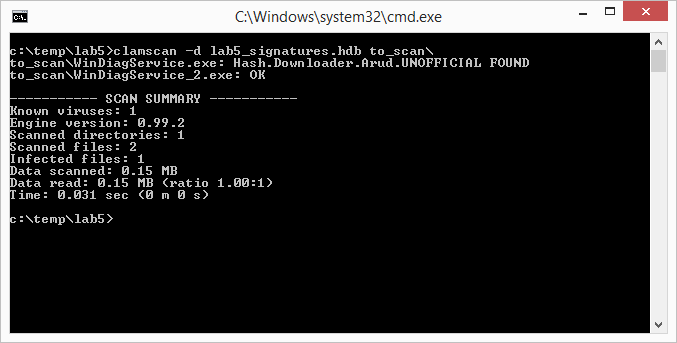


Figure 6: Output of clamscan.exe

Used with permission from Microsoft.

To improve detection, write a hash-based signature on the .text section of the WinDiagService.exe file. To do this, you need the SHA-256 hash of the .text section. Because ClamAV doesn’t include a tool to calculate the SHA-256 hash of PE file sections, you need to create a simple Python script.

1. Using the following skeleton, complete the Python script to calculate and display the SHA-256 hash of all PE file sections of a given input file.

import sys

import pefile

def main(pe\_path):

pe = pefile.PE(pe\_path)

#TODO: complete this part

# Note you can use the pe.sections list to get the PE

# file’s sections. The pefile module provides functions

# that remove the need to use hashlib, although, you # are free to use hashlib

if \_\_name\_\_ == '\_\_main\_\_':

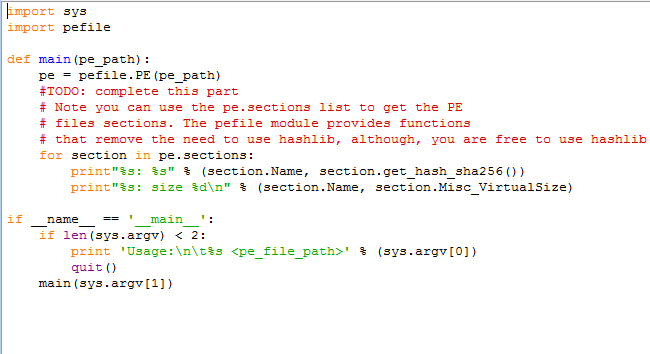
if len(sys.argv) < 2:

print 'Usage:\n\t%s <pe\_file\_path>' % (sys.argv[0])

quit()

main(sys.argv[1])

**\*You will need working code to progress through the lab, if you do not have a desire to write this code for the bonus marks associated, inform your instructor.\***

****

The output of your program should appear as in Figure 7 (with the correct SHA-256 hash values displayed).

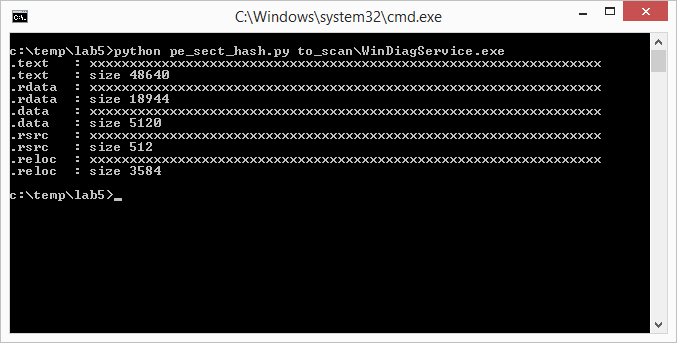


Figure 7: pe\_sect\_hash.py Output

Used with permission from Microsoft.

1. Using pe\_sec\_hash.py, calculate the PE section hashes for WinDiagService.exe and WinDiagService\_2.exe.

You should notice (and indeed expect) that the PE section hashes for the two files match.

1. Using the matching values for the SHA-256 of the PE .text section, write a better signature.
2. Put PE section signatures in a file named **lab5\_signatures.mdb** (notice the .mdb extension). The format for lines in an mdb file is as follows:

PESectionSize:PESectionHash:MalwareName

The size and hash value are reversed in order from the full file based-hash you used before.

1. For the MalwareName, use **SecHash.Downloader.Arud**.
2. When complete, run **clamscan.exe** and specify both signature files with multiple -d options, as shown in Figure 8.

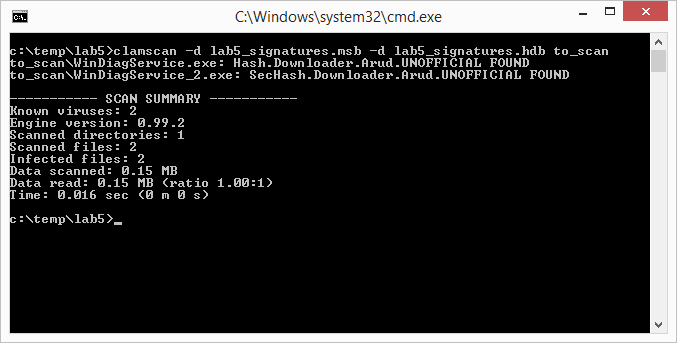


Figure 8: clamscan.exe Scan 1 Output

Used with permission from Microsoft.

Notice that ClamAV detects each file only once, even though WinDiagService.exe should match both Hash.Downloader.Arud and SecHash.Downloader.Arud.

1. To demonstrate this, scan the directory again, but specify only  
   **-d lab5\_signatures.mdb**.

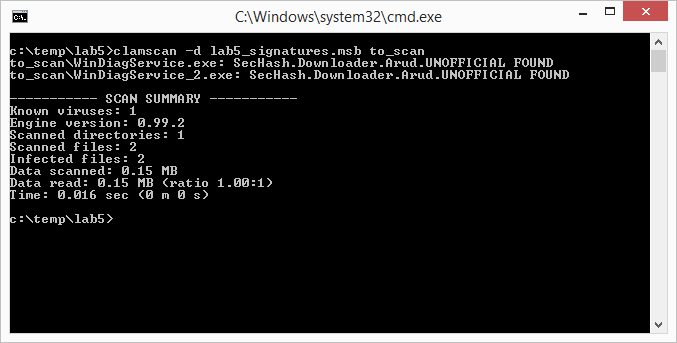


Figure 9: clamscan.exe Scan 2 Output

Used with permission from Microsoft.

## 1.4 Body-Based Signatures

Hash-based signatures over portions of a PE file offer some improvement over the hash of an entire file, but they are often not enough. ClamAV also offers body-based signatures, which are the equivalent of matching a sequence of bytes in a file.

1. Create an empty text file named **lab5\_signatures.ndb** to hold body-based signatures.

A body-based signature has the following format:

MalwareName:TargetType:Offset:HexSignature[:MinFL:[MaxFL]]

1. Refer to section 3.2.6 in [Creating signatures for ClamAV](https://github.com/vrtadmin/clamav-devel/blob/master/docs/signatures.pdf) (https://github.com/  
   vrtadmin/clamav-devel/blob/master/docs/signatures.pdf) for an explanation of all values. When writing a signature for PE files, set the TargetType to **1**.

Creating a good body-based signature takes skill, practice and experience. The following guidelines can be helpful:

* + Choose elements that are known not to change. If you have multiple samples of the same malware family that all share a common string, that string may be a good body‑based signature.
  + Choose body-based signatures that are likely to be unique to the malware sample or family, but unlikely to be used elsewhere.
  + Choose code sequences related to external data, such as code that deals with command and control communications, or choose a configuration file encryption/  
    decryption loop. The goal is to target code sequences that affect two things: the malware itself and something external to the malware. Code sequences like this are less likely to change.
  + Ideally, your signatures should detect only samples from a specific malware family and not detect any clean samples.

You will use two strings found in the .rdata section of WinDiagService.exe for the first body‑based signature. These strings are ?ver=%d&ser=%d and C:\Users\qa\Documents\projects\downloader\Release\downloader.pdb.

* Because the format string ?ver=%d&ser=%d is used for command and control communications, it is less likely to change frequently.
* The string with the path to the debugging symbols file is produced by the compiler and is less easily changed by the programmer. You will use only the last portion of the string: \downloader.pdb.

The two strings can be seen in the Ghidra output of WinDiagService.exe (see Figure 10).

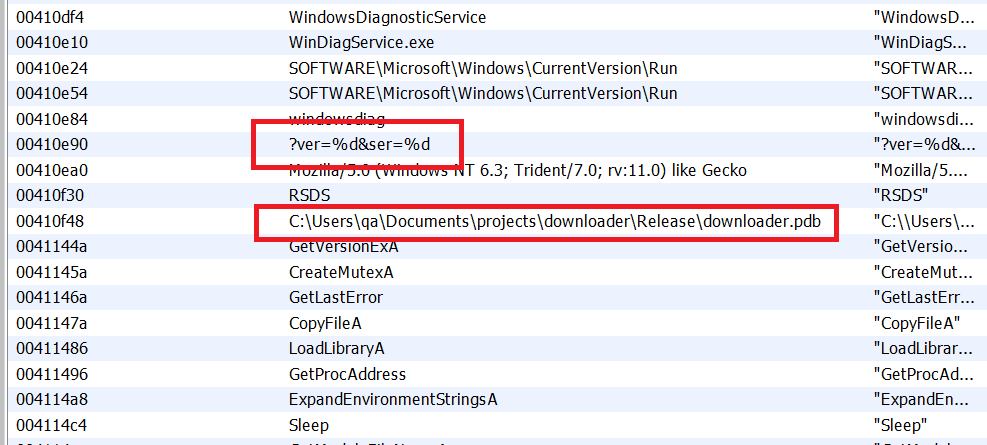


Figure 10: Ghidra Output Showing Strings

Source: Ghidra, 2019. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

For the HexSignature portion of the signature, use ?ver=%d&ser=%d\*\downloader.pdb

The above signature matches the format string, and the **\*** matches anything until the \downloader.pdb string is found. This does require that the format string is found *before* \downloader.pdb, but since the compiler is likely to place the \downloader.pdb string near the end of the .rdata section, this is an acceptable assumption.

ClamAV body-based signatures use hex values rather than ASCII, so convert the ASCII above into hex using ClamAV’s sigtool.exe.

1. The easiest method is to place the string to convert in a text file named **tohex.txt**.

The contents of tohex.txt are ?ver=%d&ser=%d

1. To convert the contents of tohex.txt using sigtool.exe, execute the following command:

> sigtool.exe --hex-dump < tohex.txt



Figure 11: sigtool.exe Output for the Format String

Used with permission from Microsoft.

The output for the second string is as follows:



Figure 12: sigtool.exe Output for the PDB String

Used with permission from Microsoft.

1. Because you are matching to the end of the strings, add a null byte (00) manually to the end of both strings.

The final version of HexSignature is:

3f7665723d2564267365723d256400\*5c646f776e6c6f616465722e70646200

The final value for a complete signature is Offset. You could use **\*** to search an entire file, but that is inefficient. Since you only need to search in the .rdata section, use the information in section 3.6.2 of the *Creating signatures for ClamAV* document to restrict your signature to the entire .rdata section.

You can use Hiew to determine that the .rdata section is the second section specified, so the index of the second section is 1.

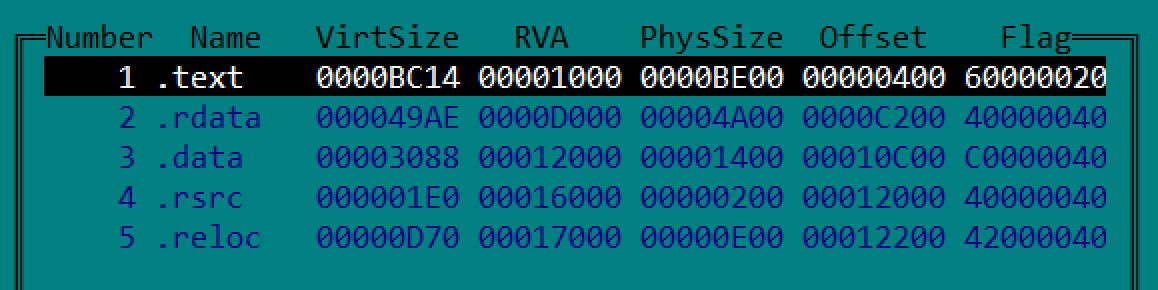


Figure 13: Section Listing using Hiew

Source: Hiew software, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

The value to use for Offset is SE1

1. Use PE.Downloader.Arud.A as the name for this signature. The entire signature is thus:

PE.Downloader.Arud.A:1:SE1:3f7665723d2564267365723d256400\*5c646f776e6c6f616465722e70646200

1. Copy the above signature as one line to **lab5\_signatures.ndb**.
2. Test the signature by running the following command:

> clamscan -d <full path to>\lab5\_signatures.ndb <full path to>\to\_scan

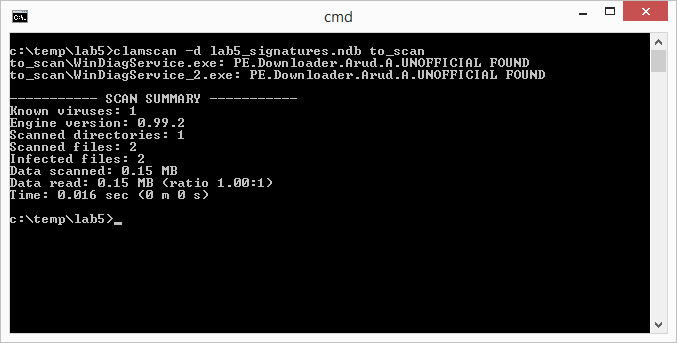


Figure 14: Detection using body based signature

Used with permission from Microsoft.

## 1.5 Opcode Signatures

Recall the function fun\_00401120 from WinDiagService.exe. When you reverse engineered the sample, you may have identified this function as decrypting string names.

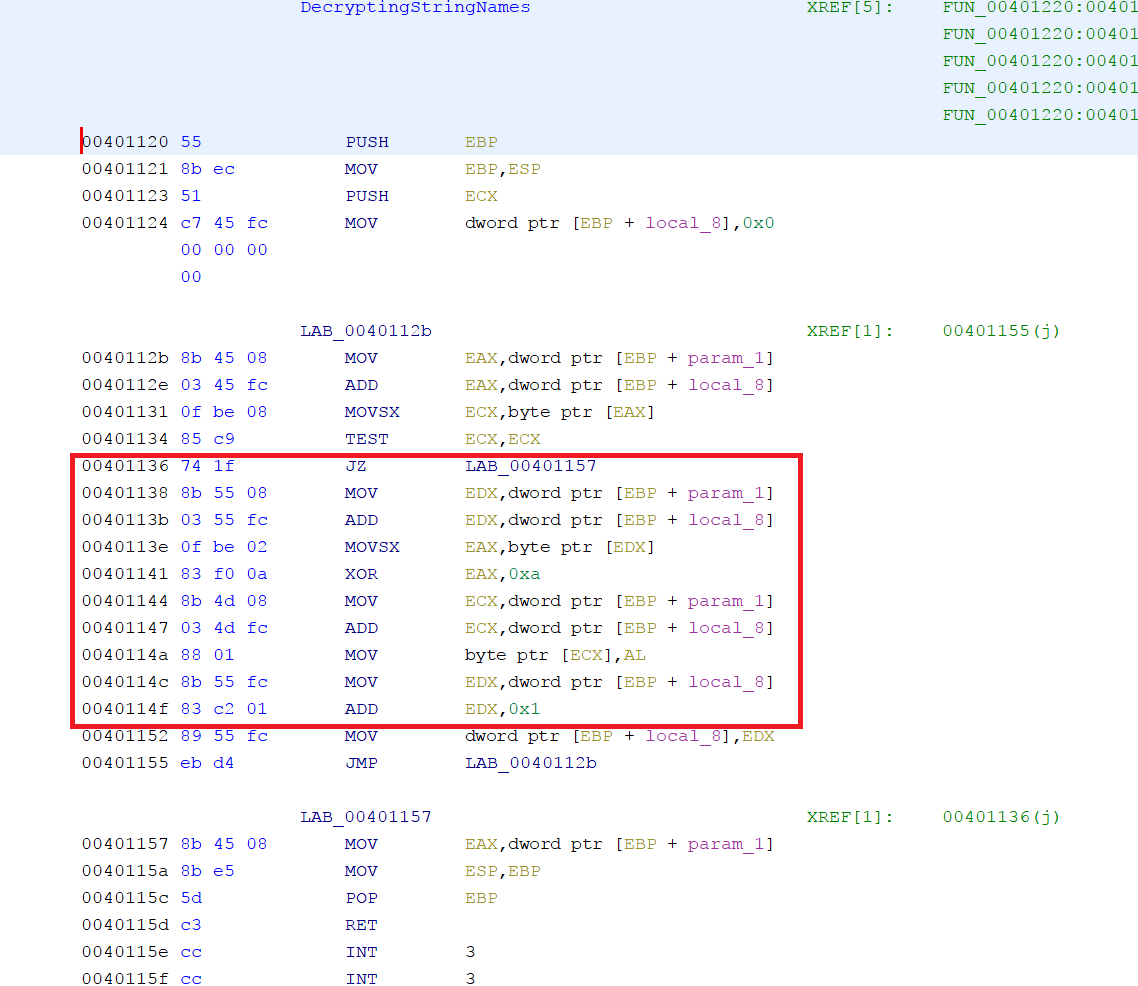


Figure 15: Ghidra output in WinDiagService.exe

The code blocks highlighted in red are the main parts of the decryption function. These code blocks could be good candidates for a body-based signature.

The opcodes for these instructions are listed beside the mnemonic instructions in the main view.

At this point the opcodes could simply be copied into a ClamAV body-based signature, but you will make one small improvement first.

Assume you want to account for any single byte xor key, rather than the hard-coded value of 0xa that is in the assembly listing. Specifically this instruction:

83 F0 0A xor eax, 0xA

1. To do this, replace the value of 0A with “??” to represent any byte. For example:

83 F0 ?? xor eax, ??

1. Now create a slightly more generic signature (note the “??” in the middle):

8B45080345FC0FBE0885C9741F8B55080355FC0FBE0283F0??8B4D08034DFC88018B55FC83C2018955FCEBD4

1. Using the above hex values, construct a second body-based signature to detect WinDiagService.exe. This time, use SE0 for the offset, because the function sub\_401120 is located in the .text section, which is the first section in WinDiagService.exe.

PE.Downloader.Arud.B:1:SE0:8B45080345FC0FBE0885C9741F8B55080355FC0FBE0283F0??8B4D08034DFC88018B55FC83C2018955FCEBD4

1. Create a file named **lab5\_signatures2.ndb** and copy the above signature to the file.
2. Test the signature by executing the following command:

> clamscan -d lab5\_signatures2.ndb to\_scan

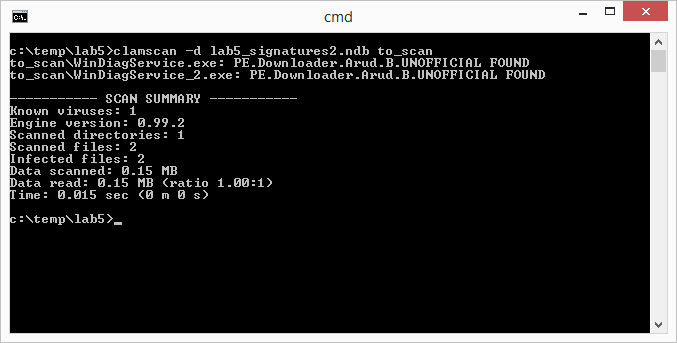


Figure 18: clamscan.exe Scan 3 Output

Used with permission from Microsoft.

## 1.6 Logical Signatures

Logical signatures allow for more flexible signature writing and can use Boolean logic.

Recall that your first body-based signature for WinDiagService.exe was:

PE.Downloader.Arud.A:1:SE1:3f7665723d2564267365723d256400\*5c646f776e6c6f616465722e70646200

One shortcoming of this signature is that it requires the first matching hex chunk (?ver=%d&ser=%d) to come before the second (\downloader.pdb). Logical signatures can be used to eliminate this restriction.

Section 3.2.7 of the *Creating signatures for ClamAV* document describes the details of logical signatures. The format of logical signatures is as follows:

SignatureName;TargetDescriptionBlock;LogicalExpression;Subsig0;

Subsig1;Subsig2;...

Now you’ll convert your body-based signature to a logical signature.

1. For SignatureName, use **PE.Downloader.Arud.C**.
2. The TargetDescription allows a number of options, which are all explained in the *Creating signatures for ClamAV* document. Use **Target:1**, because you are targeting PE files. Don’t use any additional values in TargetDescription.
3. For the LogicalExpression, use **0&1** where 0 refers to Subsig0 and 1 refers to Subsig1.
4. For Subsig0, use **3f7665723d2564267365723d256400** and for Subsig1, use **5c646f776e6c6f616465722e70646200**. As well, put them in the opposite order they appear in the malicious file to demonstrate that the order doesn’t matter for logical signatures.

The final signature is:

PE.Downloader.Arud.C;Target:1;0&1;5c646f776e6c6f616465722e70646200;3f7665723d2564267365723d256400

1. Create an empty text file named **lab5\_signatures.ldb** and copy the logical signature to it.
2. Use the clamscan.exe command line shown in Figure 19 to test the signature.

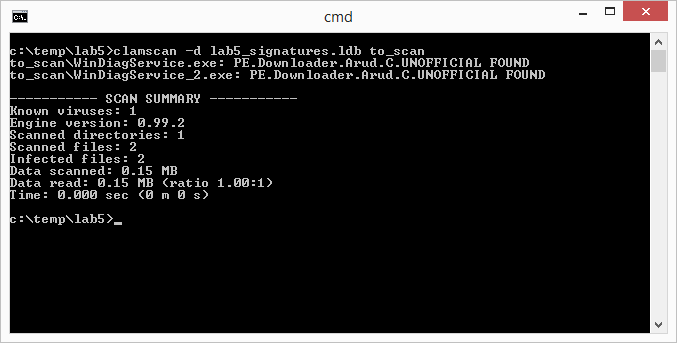


Figure 19: clamscan.exe Output for Logical Signature

Used with permission from Microsoft.

1. Writing Detections

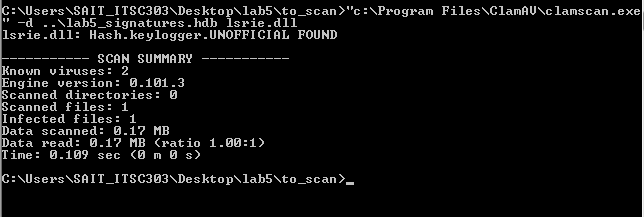
Complete the following detection problems to give yourself more practice.

## 2.1 Full File Hash Detection

1. Write a full file SHA-256 hash detection for the file **to\_scan\lsrie.dll**.
2. Fill out the following table. To develop a name, find either a string to use inside the file or find some other file data.

|  |  |  |
| --- | --- | --- |
| **HashString** | **FileSize** | **MalwareName** |
| B80C1BB48E94AE37B98EAA09A9EF20DCA7049C3DA12ADB94924B0D8B85EF4E29 | 178688 | Hash.keylogger |

1. Add the signature to the **file lab5\_signatures.hdb** and use clamscan.exe to test the signature.

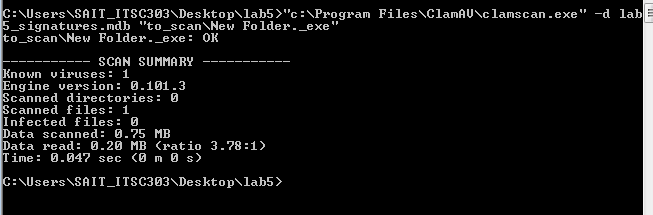


## 2.2 PE Section Hash Detection

1. Write a PE section hash detection for the file **to\_scan\New Folder.\_exe**.
2. Fill out the following table. To develop a name, find either a string to use inside the file or find some other file data.

|  |  |  |
| --- | --- | --- |
| **PESectionSize** | **PESectionHash** | **MalwareName** |
| 8192 | 3fafa78ef481a192df45b999883ea276dee74ecdf4e8cfbd4f9f5d739ed6ec69 | SecHash.escalation.xml |

1. Add the signature to the **file lab5\_signatures.mdb** and use clamscan.exe to test the signature.



## 2.3 Body-Based Detection

1. Write body-based detections for the following files in the **to\_scan** directory:

9222bca9e7b00c8918c4ac6fb415c77239e88dc296269273056372d5034b0daf.bin

943a7838f3eccc0984219642f533deaffb7b99e8c1d51157115bc87cf72aa80f.bin

1. Create body-based detections using either strings found in the binary-based or opcode-based detections. Provide a brief explanation of why you chose one or the other.
2. Refer to the *Creating signatures for ClamAV* documentation for help on writing body‑based detections.
3. Fill out the following table. To develop a name, find either a string to use inside the file or find some other file data.

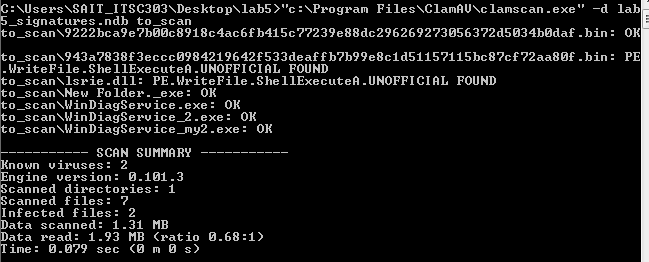
|  |  |  |  |
| --- | --- | --- | --- |
| **MalwareName** | **TargetType** | **Offset** | **HexSignature** |
| PE.GETPASSWORD1.REPLACEFILEDLG | 1 | se1 | 474554504053574f52443100\*5245504c41434546494c45444c4700 |
| PE.WriteFile.ShellExecuteA | 1 | se1 | 577269746546696c6500\*5368656c6c457865637574654100 |

1. Alternatively, use logical signatures for one or both of the signatures you write. If using logical signatures, fill out this table.

**As you can see, I used body based signatures not logical signatures**

|  |  |  |  |
| --- | --- | --- | --- |
| **SignatureName** | **TargetDesriptionBlock** | **LogicalExpression** | **Subsig0;Subsig1;…** |
|  | Target:1 |  |  |
|  | Target:1 |  |  |

1. Add the body-based signatures to the file **lab5\_signatures.ndb**. If you are using any logical signatures, add them to **lab5\_signatures.ldb**.
2. Use clamscan.exe to test the signatures.



**Questions**

1. For file 9222bca9e7b00c8918c4ac6fb415c77239e88dc296269273056372d5034b0daf.bin, describe the signature and why you chose what you did. Attempt to justify why it will uniquely discover this malware without creating false positives on clean files.

**I used the two strings get password and place file, as those are unique and suspicious.**

1. For file 943a7838f3eccc0984219642f533deaffb7b99e8c1d51157115bc87cf72aa80f.bin, describe the signature and why you chose what you did. Attempt to justify why it will uniquely discover this malware without creating false positives on clean files.

**The two strings I chose here was write file and shell execute, so this will scan for any files that will write and execute in a shell, (such as a trojan)**

# 

# References

*Fun with ClamAV*. (2013). Retrieved April 24, 2017, from http://www.pwnage.io/2013/06/fun-with-clamav.html

Hex-Rays. (2017). IDA Pro. [Computer software]. Retrieved from https://www.hex-rays.com/products/ida/support/download.shtml

Hörz, M. (2017). HxD (Version 1.7.7.0). [Computer software]. Retrieved from https://mh-nexus.de/en/hxd/

Kojm, T. (2017). *Clam AntiVirus 0.99.3 user manual*. Cisco Systems, Inc. Retrieved April 24, 2017, from https://github.com/vrtadmin/clamav-devel/blob/master/docs/clamdoc.pdf.

Kojm, T. (2017). *Creating signatures for ClamAV*. Retrieved April 24, 2017, from https://github.com/vrtadmin/clamav-devel/blob/master/docs/signatures.pdf

Suslikov, E. (2017). Hiew (Version 8.53) [Computer software]. Retrieved from http://www.hiew.ru/

*Writing signatures for Clam AV 0.99: A tutorial*. (2016). Retrieved April 24, 2017, from http://malwarefor.me/writing-signatures-for-clam-av-0-99-a-tutorial/