Malware Analysis

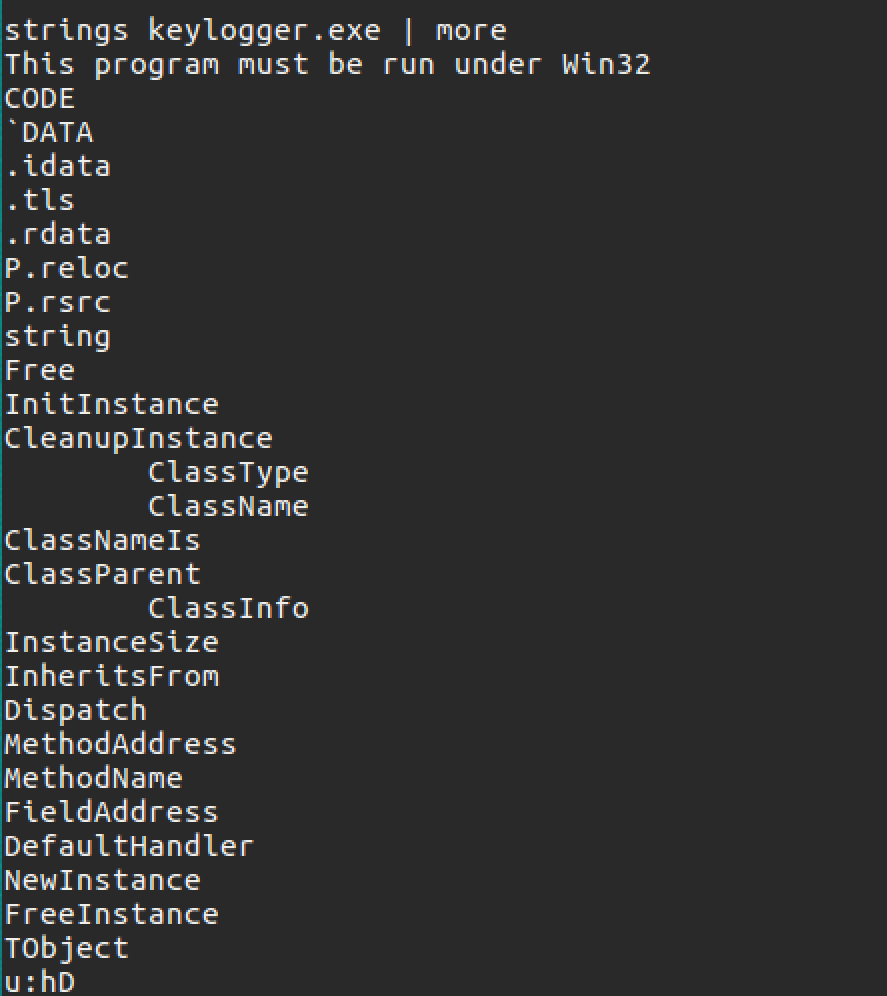
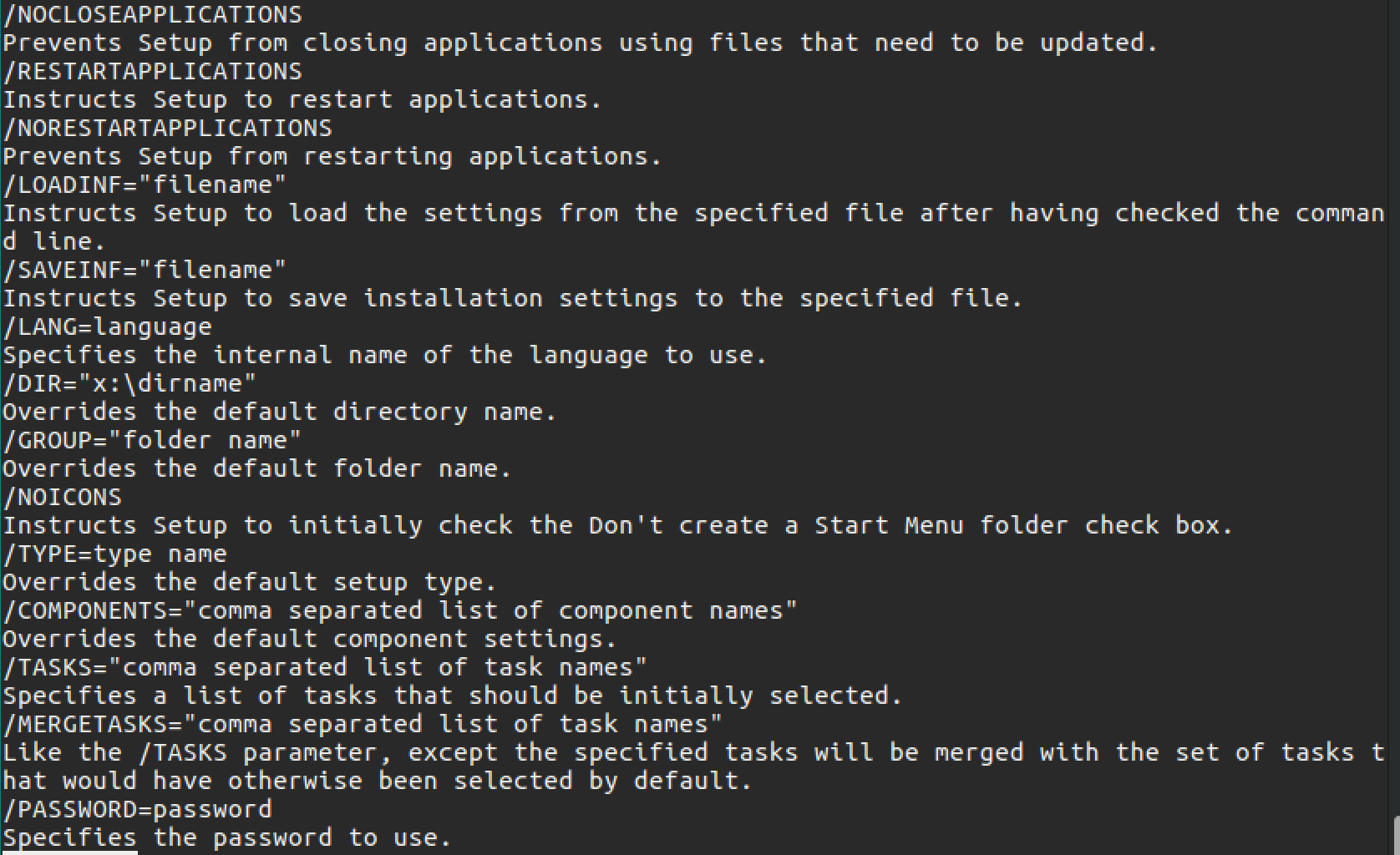
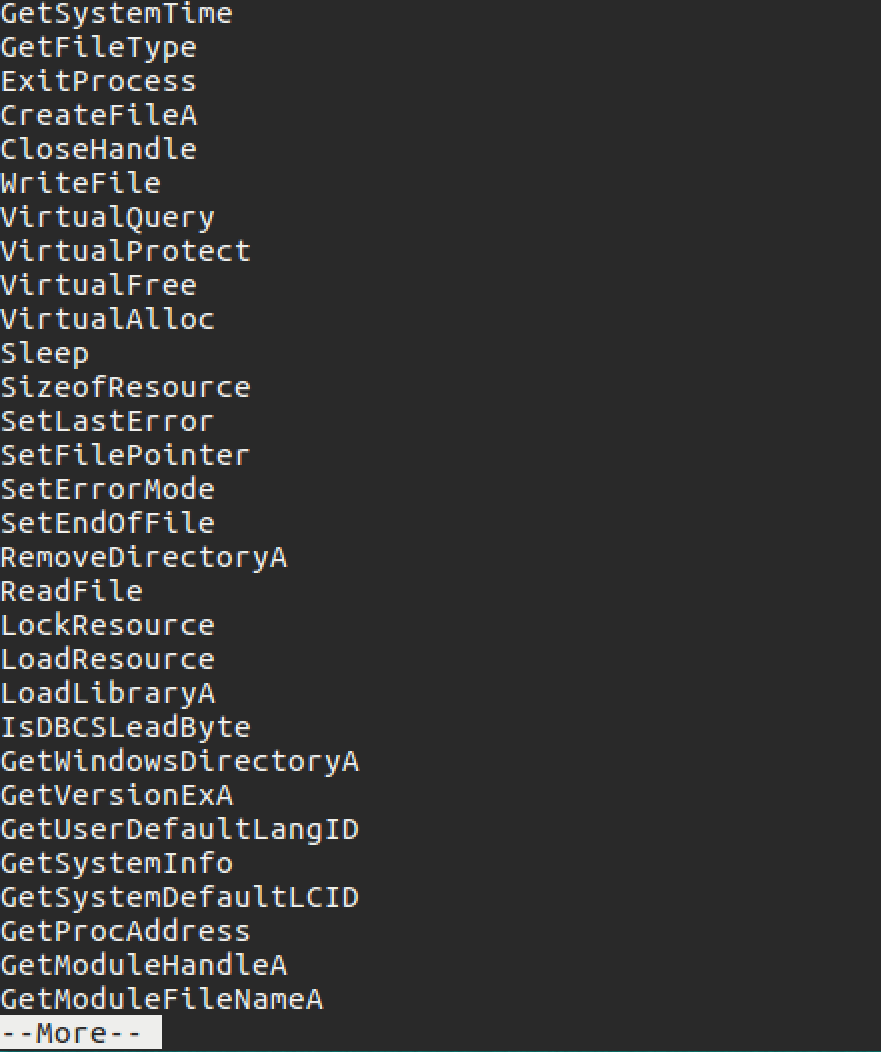
**Module 02 - Basic Static Analysis**

# Finding Hashes & Strings: Lab Solutions

Total point values of this lab: 10 points. Point value is assigned by each task below.

### **Exercise 1 – Step 1:**

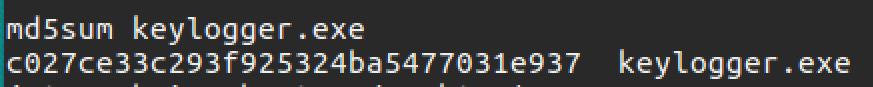
*Learning Outcome 2 – 2 Points*

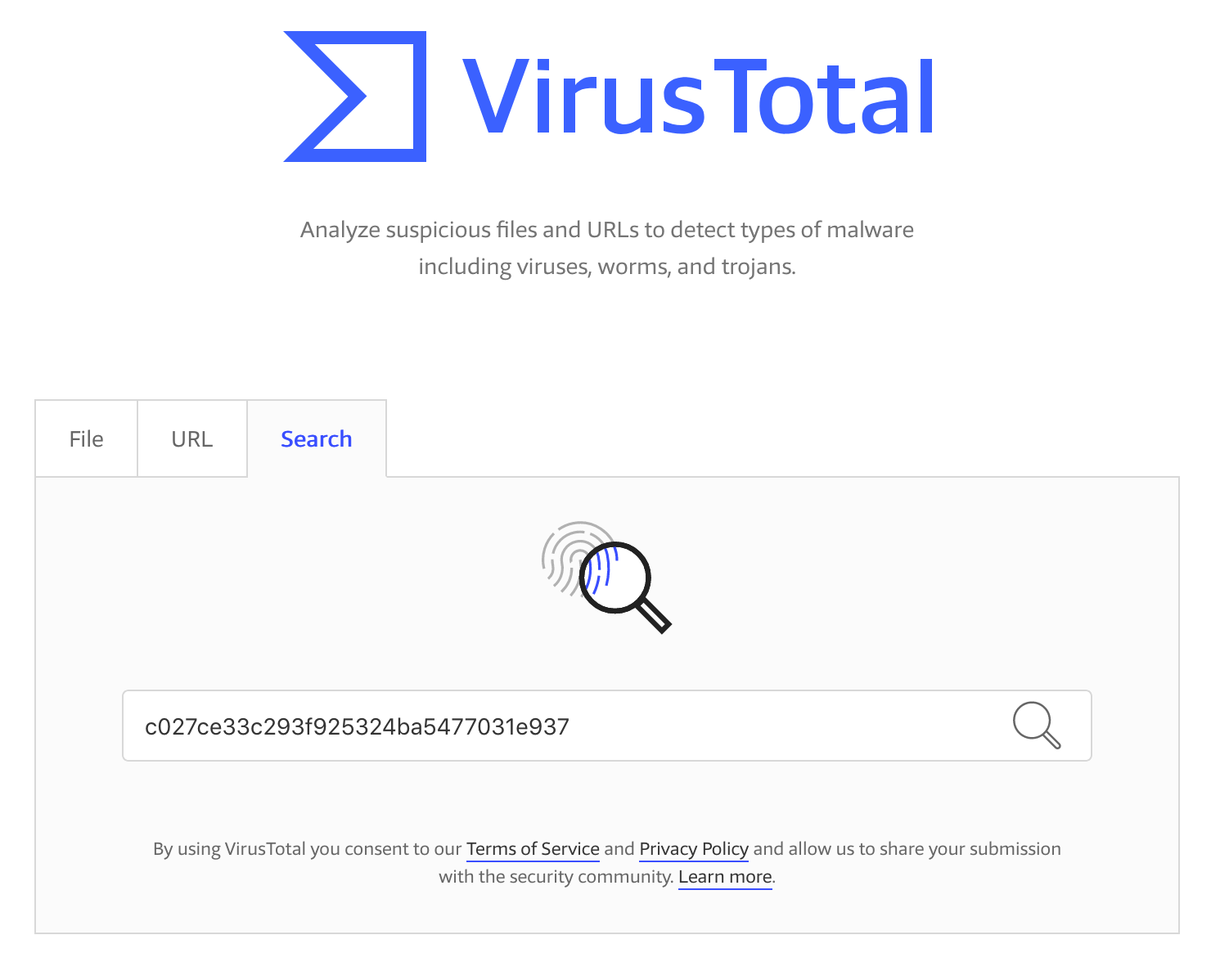
1. Using the strings utility in a Linux terminal is one of the easier methods available for the student. Generally, this will create a lot of output and it is helpful to redirect through additional utilities such as MORE. In addition, the student can use GREP to help filter the output for such things as URLs and message strings.   
     
     
     
   Searching through the output, information about possible commands and insight into program functionality can be found:  
     
     
     
   Strings will also display information about imported libraries and APIs. This can often confuse the student and it thoroughly covered in the Windows API module. While these strings provide valuable information, it is often easier to view in a PE parsing utility such as PE Studio.  
     
   

### **Exercise 1 – Step 2:**

*Learning Outcome 1 – 2 Points*

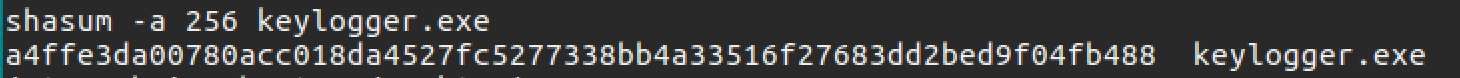
There are many utilities available to generate hashes based on a file. Similar to the last exercise, it is often easiest to generate hashes using a Linux terminal and MD5SUM.

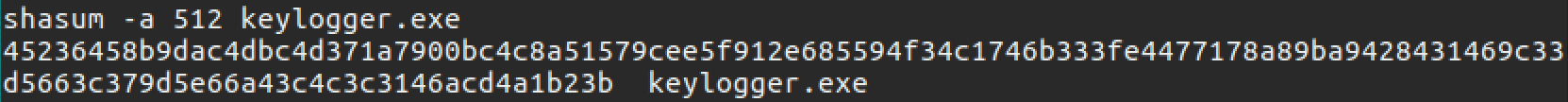


This hash value can be used to perform additional research, such as using VirusTotal to help determine if the file is malicious:  
  


### **Exercise 1 – Step 3:**

*Learning Outcome 1 – 1 Point*

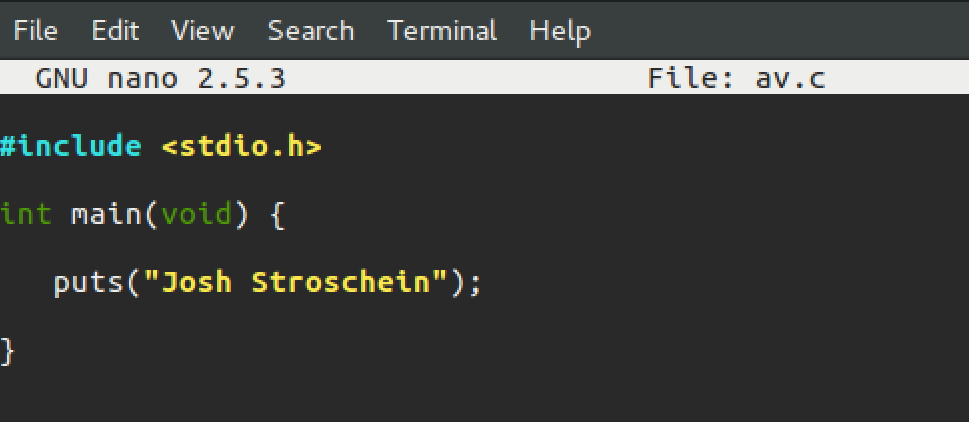




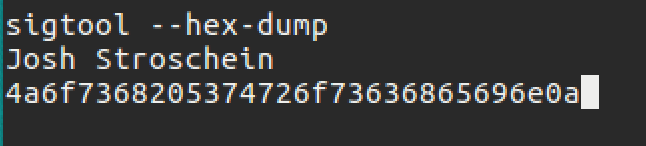
### **Exercise 2 – Step 1:**

*Learning Outcome 3 – 5 Points*

This exercise will introduce the basics of anti-virus through the exploration of signature development. Instruct the student to generate a custom program with an easily identifiable ASCII string, such as their name. This exercise works best if using a language such as C or C++ in which there is no interpreter to deal with. For example, the following C program would suffice:

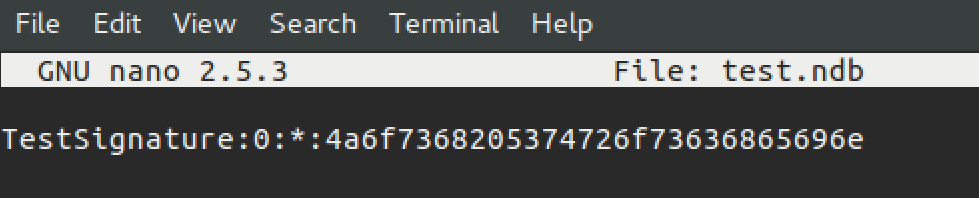


The string that they will write the signature for would be the argument for puts. Next, use Clam-AV’s sigtool to generate the corresponding byte sequence for the ASCII string – note the ‘0a’ at the end of the byte sequence, this is the newline from the user pressing the enter key and is NOT part of the signature:



Finally, create a custom signature file (text file with an extension of ndb) and add the signature in the following format:

SignatureName:0:\*:Signature



Run clamscan and get results – don’t forget to compile the test program before scanning!

