# Objective: Recover the Tolkien Ring—Answers Only

## Wireshark Practice

### Answers to Wireshark Practice

#### Question 1

Files can be extracted from captures, and Wireshark can grab some types of objects for you automatically. Only one type exists in this pcap (you can try them all to prove this.) The answer is HTTP  
Graphical user interface, application

Description automatically generated

#### Question 2

Export-Objects -> HTTP gives you this. The answer is app.php  
Graphical user interface, text, application

Description automatically generated

#### Question 3

The last, largest, app.php file starts at packet 687, and that is the one they want.

#### Question 4

The server is the one that responds to the requests sent by the client. If you are not sure, look at the [three-way handshake](https://www.techopedia.com/definition/10339/three-way-handshake). The client sends the SYN, and the server sends the SYN/ACK. 192.185.57.242

#### Question 5

This requires more digging. If you use, Follow -> HTTP Stream on the GET /app.php packet you will see the contents of app.php. (Note: click on a packet with HTTP in the protocol column so that HTTP stream is available on the Follow -> menu. If you Follow -> TCP stream, Wireshark will not expand the GZIP in the response, and you will see gibberish.) Or you can save app.php as a file from the HTTP object list, above. At any rate, after the beginning shown here you see a giant block of what appears to be base64 encoded data. This is the malware file. After the base64 you can see the code where the file is decoded and saved. The answer for the objective is Ref\_Sept24-2020.zip

Top of streamText

Description automatically generated

Scrolled way down Graphical user interface, text

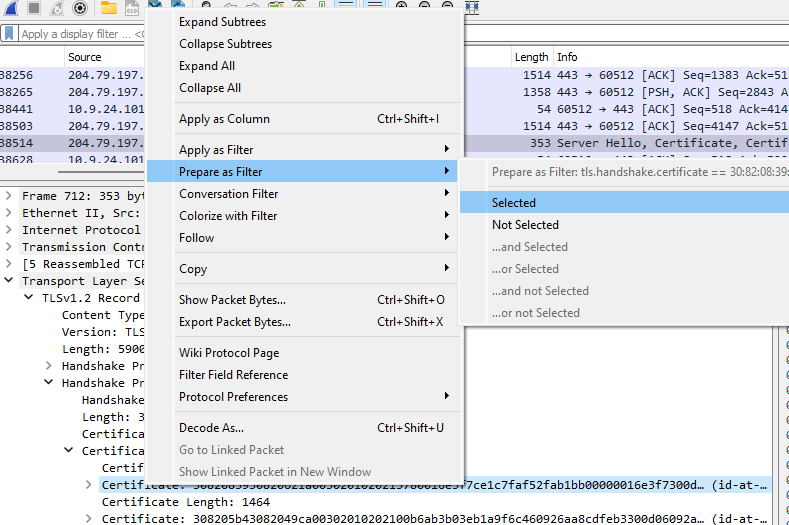
Description automatically generated

#### Question 6

I can seldom find the display filter I need by downloading big lists of filters. Instead, I find what I want in a packet.

Graphical user interface, text, application, email

Description automatically generated

Right click on the certificate line in the packet body. Select Prepare as filter -> selected.  


A display filter magically appears.   
Graphical user interface, text, application

Description automatically generated

Get rid of all the hex stuff because we want to find all certificates, not just a specific one. Now we have nothing but certificates.  
Graphical user interface, table

Description automatically generated

Here is what it looks like when you click through the certificate tree to find the country.  
Table

Description automatically generated with low confidence

With Wireshark, you must click through every certificate to find the country code. You can export the certificates If you wish to validate whether a certificate is valid or not, using [this method](https://community.fortinet.com/t5/FortiGate/Technical-Tip-How-to-extract-a-certificate-from-a-Wireshark/ta-p/189764). However, the challenge will accept countries with good certs as well as bad if all the bad ones are included. Note that Wireshark is giving country codes and the challenge wants full names. You can [look up the names here](https://www.ssl.com/country-codes/).

The bad ones were Israel, South Sudan. You could also have Ireland and United States in the mix and get credit.

If you like tshark, you can get all the countries very quickly with this from PowerShell,  
&'C:\Program Files\Wireshark\tshark.exe' -r .\suspicious.pcap -Y 'tls.handshake.certificate' -T fields -e 'x509sat.CountryName'

Or this from Linux:  
Text

Description automatically generated

We’ll work more with tshark in Web Ring later.

## Windows Event Logs

### Answers

The key part is to extract all the commands. Once you have that, you can answer the questions in the terminal.

#### PowerShell

First, put the text version of the event file into a variable and reverse it as Eric did in the talk. Here’s a [link to the reverse method.](https://devblogs.microsoft.com/scripting/powertip-reverse-array-with-powershell/)



Searching for 4014, the event ID Eric used, gives many hits. Note that the date is 12/24/2022.  
Table

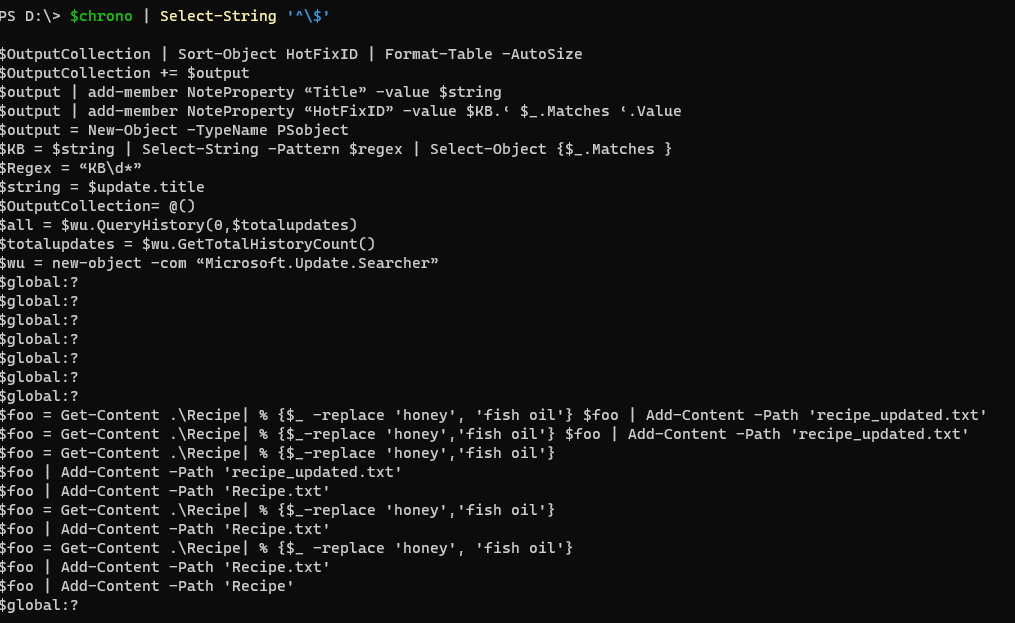
Description automatically generated with medium confidence

The actual command that was entered is in that line after each match. You can get it to output using   
$chrono | Select-String ‘4014’ -context 0,1  
if you want to.

We will move on to the searches Eric did.

Now look for variables, lines that start with $ like Eric did. The regex, ‘^\$’, says to look for lines where the beginning (^) is $. The $ must be escaped (\$) because $ finds the end of a line in a [regular expression.](https://towardsdatascience.com/regular-expressions-clearly-explained-with-examples-822d76b037b4)

$chrono | Select-String '^\$'



It appears everything is related to foo or recipe. Let’s search for that.  
Text

Description automatically generated

Darn ParameterBindings! Let’s get rid of them as Eric did.  
Text

Description automatically generated

The elf’s diary is fun to read, but we do not need it. Here is the list of commands we need, from the last part of the output.

cat .\Recipe

$foo = Get-Content .\Recipe| % {$\_ -replace 'honey', 'fish oil'} $foo | Add-Content -Path 'recipe\_updated.txt'

cat .\recipe\_updated.txt

$foo = Get-Content .\Recipe| % {$\_-replace 'honey','fish oil'} $foo | Add-Content -Path 'recipe\_updated.txt'

cat .\recipe\_updated.txt

$foo = Get-Content .\Recipe| % {$\_-replace 'honey','fish oil'}

$foo | Add-Content -Path 'recipe\_updated.txt'

cat .\recipe\_updated.txt

$foo | Add-Content -Path 'Recipe.txt'

cat .\Recipe

$foo = Get-Content .\Recipe| % {$\_-replace 'honey','fish oil'}

$foo | Add-Content -Path 'Recipe.txt'

cat .\Recipe

$foo = Get-Content .\Recipe| % {$\_ -replace 'honey', 'fish oil'}

$foo | Add-Content -Path 'Recipe.txt'

$foo | Add-Content -Path 'Recipe'

cat .\Recipe

del .\Recipe.txt

del .\recipe\_updated.txt

#### Linux

You can solve the challenge directly in the terminal if you like Linux.

When you grep 4104, you get hits, and they all have 12/24/2022 as the date. Remember that.  
Text

Description automatically generated

There are no commands though. I examined the fille in less and found that the line after the hit had the command in it. So, let’s also output the line after the hit.  
grep -A 1 4104 powershell.evtx.log | head -n 15  
A screenshot of a computer screen

Description automatically generated with medium confidence  
Ah, commands. Get-WMIObject and Get-Process are PowerShell commands.

Let’s look for lines that begin with a $, as Eric did.  
Text

Description automatically generated

There are many instances of foo and recipe; maybe a search for them will give more information. The or feature (|) of grep is only available in the extended version so we use -E. Since recipe and Recipe both appear, make the search case-insensitive with -i.  
grep -A 1 4104 powershell.evtx.log | grep -Ei '\$foo|recipe'  
Text

Description automatically generated

I forgot to reverse the order as Eric did. We can do that with tac (cat spelled backwards.)  
grep -A 1 4104 powershell.evtx.log | grep -Ei '\$foo|recipe'| tac  
Text

Description automatically generated

Now we have the same list of commands that PowerShell gave. I think it was easier in Linux, but I am biased.

#### Answer List

Our intruder, Grinchem most likely, made a lot of errors before he performed the attack correctly. Only the last seven lines really matter. Here are the answers.

Ready to begin? yes

1. What month/day/year did the attack take place? For example, 09/05/2021.

: 12/24/2022

2. An attacker got a secret from a file. What was the original file's name?

: Recipe

3. The contents of the previous file were retrieved, changed, and stored to a variable by the attacker. This was done multiple times. Submit the last full PowerShell line that performed only these actions.

: $foo = Get-Content .\Recipe| % {$\_ -replace 'honey', 'fish oil'}

4. After storing the altered file contents into the variable, the attacker used the variable to run a separate command that wrote the modified data to a file. This was done multiple times. Submit the last full PowerShell line that performed only this action.

: $foo | Add-Content -Path 'Recipe'

5. The attacker ran the previous command against a file many times. What is the name of this file?

: Recipe.txt

6. Were any files deleted? (Yes/No)

: Yes

7. Was the original file (from question 2) deleted? (Yes/No)

: No

8. What is the Event ID of the log that shows the actual command line used to delete the file?

: 4104

9. Is the secret ingredient compromised (Yes/No)?

: Yes

10. What is the secret ingredient?

: honey

## Suricata Regatta

### Answer 1

The parts of the answer are:

alert Our action will always be alert in this challenge. Others are block, pass, etc.

dns This calls the parser for the DNS protocol. It will detect DNS by content rather than port number, so the port number can be left at any

$HOME\_NET any -> any any This looks for traffic from the addresses listed in $HOME\_NET to any IP address and port. Fitzy asked for lookups, or outgoing traffic.

msg Paste in the message Fitzy asked for.

dns.query This [DNS Keyword](https://suricata.readthedocs.io/en/suricata-6.0.0/rules/dns-keywords.html) will decode the [strange compression method](https://cabulous.medium.com/dns-message-how-to-read-query-and-response-message-cfebcb4fe817) that DNS uses to save space, so you can search without considering it.

content Finally, the URI you are looking for.

sid some number that is unique in surricata.rules.

alert dns $HOME\_NET any -> any any (msg:"Known bad DNS lookup, possible Dridex infection"; dns.query; content:"adv.epostoday.uk"; nocase; sid:1000000;)

### Answer 2

The symbol for bidirectional traffic is <>. There is nothing after msg except for the sid.

alert http 192.185.57.242 any <> $HOME\_NET any (msg:"Investigate suspicious connections, possible Dridex infection"; sid:1000001;)

### Answer 3

The keyword you need to use is tls.cert\_subject. I added a nocase; to make it case insensitive since I was worried about the capital I in the name.

alert tls any any -> any any (msg:"Investigate bad certificates, possible Dridex infection"; tls.cert\_subject; content: "heardbellith.Icanwepeh.nagoya"; nocase; sid:1000002;)

### Answer 4

This rule also works with the file\_data keyword instead of http.response\_body.

alert http any any -> any any (msg:"Suspicious JavaScript function, possible Dridex infection"; http.response\_body; content:"let byteCharacters = atob"; nocase; sid:1000003;)

A picture containing text, device, meter, gauge

Description automatically generated