

DNSExfiltrator Challenge Writeup

Writeup for DNSExfiltrator challenges from Serbian Cybersecurity Challenge 2021.

Part 1

DNS Exfiltration - Part 1

[Challenge](#) [Forensics](#) [Network Security](#) [Wireshark](#)

Length: 2 mins
Difficulty: Medium

Description and resources

Mission brief

You've captured a communication between the infected machine and a hacker. Machine sends password file to hacker using DNSExfiltrator. You know that hacker uses simple password with three same letters to protect the exfiltration. You have to find the name of the file that is exchanged.

Instructions

Explore the pcap, explore the exfiltrator and decode the communication.

[Start exercise](#)

Solution:

We got *challengeDNS_anon.pcapng* file, and if we open it with Wireshark and apply **dns** filter we can see that first packet has name `init.OBLWI5KNOBDDC3DFFZTGYNDHPQZQ.base32.dnsresearch.ml`

No.	Time	Source	Destination	Protocol	Length	Info
54	6.328997645	10.3.27.86	192.168.72.171	DNS	115	Standard query 0xc00d TXT init.OBLWI5KN0BDDC3DFFZTG
55	6.331412228	10.3.27.86	192.168.72.171	DNS	115	Standard query 0xc00d TXT init.OBLWI5KN0BDDC3DFFZTG
62	6.390143153	192.168.72.171	10.3.27.86	DNS	130	Standard query response 0xc00d TXT init.OBLWI5KN0BDDC3DFFZTG
63	6.390285145	192.168.72.171	10.3.27.86	DNS	130	Standard query response 0xc00d TXT init.OBLWI5KN0BDDC3DFFZTG
64	6.495385063	10.3.27.86	192.168.72.171	DNS	305	Standard query 0xea4d TXT 0.AHC36UMJWLNYTF7ENQ2EFB
65	6.485392979	10.3.27.86	192.168.72.171	DNS	305	Standard query 0xea4d TXT 0.AHC36UMJWLNYTF7ENQ2EFB
70	6.469999733	192.168.72.171	10.3.27.86	DNS	319	Standard query response 0xea4d TXT 0.AHC36UMJWLNYTF7ENQ2EFB
71	6.469164988	192.168.72.171	10.3.27.86	DNS	319	Standard query response 0xea4d TXT 0.AHC36UMJWLNYTF7ENQ2EFB
72	6.472485449	10.3.27.86	192.168.72.171	DNS	305	Standard query 0xd9ea TXT 1.76LND25CZBPDGVZRONTHUMR1
73	6.472517131	10.3.27.86	192.168.72.171	DNS	305	Standard query 0xd9ea TXT 1.76LND25CZBPDGVZRONTHUMR1
78	6.530618214	192.168.72.171	10.3.27.86	DNS	319	Standard query response 0xd9ea TXT 1.76LND25CZBPDGVZRONTHUMR1
79	6.530683782	192.168.72.171	10.3.27.86	DNS	319	Standard query response 0xd9ea TXT 1.76LND25CZBPDGVZRONTHUMR1
80	6.532246383	10.3.27.86	192.168.72.171	DNS	80	Standard query 0x6517 TXT 2.CM0.dnsresearch.ml
81	6.532273511	10.3.27.86	192.168.72.171	DNS	80	Standard query 0x6517 TXT 2.CM0.dnsresearch.ml
86	6.584843058	192.168.72.171	10.3.27.86	DNS	94	Standard query response 0x6517 TXT 2.CM0.dnsresearch.ml
87	6.584934240	192.168.72.171	10.3.27.86	DNS	94	Standard query response 0x6517 TXT 2.CM0.dnsresearch.ml

Internet Protocol Version 4, Src: 10.3.27.86, Dst: 192.168.72.171	
User Datagram Protocol, Src Port: 64441, Dst Port: 53	
Domain Name System (query)	
Transaction ID: 0xc00d	
Flags: 0x0100 Standard query	
Questions: 1	
Answer RRs: 0	
Authority RRs: 0	
Additional RRs: 0	
Queries	
init.OBLWI5KN0BDDC3DFFZTGYNHPQZQ.base32.dnsresearch.ml: type TXT, class IN [Name Length: 55] [Label Count: 5] Type: TXT (Text strings) (16) Class: IN (0x0001) [Response in: 62]	
0000	74 da 88 4d 36 79 10 c9 4c d0 8b bf 00 00 45 00 t..M6y..N....E..
0010	00 65 01 4d 00 00 80 11 9a 8f 8a 03 1b 56 c0 a8 .e.M.....V..
0020	48 ab fb b9 00 35 00 51 94 97 c0 0d 01 00 00 01 H....5.Q.....
0030	00 00 00 00 00 00 04 69 6e 69 74 1c 4f 42 4c 571nit.OBLW
0040	49 35 4b 4e 4f 42 44 44 43 33 44 46 46 5a 54 47 I5KN0BDD C3DFFZTG
0050	59 4e 44 48 50 51 5a 51 06 62 61 73 65 33 32 0b YNDHPQZQ base32
0060	64 6e 73 72 65 73 65 61 72 63 68 02 6d 6c 00 00 dnsresea rch.ml..
0070	10 00 01

We can run next command to decode given string:

```
echo "OBLWI5KN0BDDC3DFFZTGYNHPQZQ====" | base32 -d
```

And we got the flag for the first part, name of the file that is exchanged is **pWduMpF1le.fl4g**

Alternatively, we can use CyberChef or any other online tool that can decode base32:

Operations	Recipe	Input
Search...	From Base32	OBLWI5KN0BDDC3DFFZTGYNHPQZQ
Favourites ★	Alphabet A-Z2-7=	
Data format	<input checked="" type="checkbox"/> Remove non-alphabet chars	
Encryption / Encoding		
Public Key		
Arithmetic / Logic		
Networking		
Language		
Utils		
Date / Time		
Extractors		
Compression		
Hashing		

Output
<p>time: 2ms length: 17 lines: 1</p> <p>pwduMpFile.fl4g 3</p>

If someone would like to see how this part can be solved using Python, you can take look at file `solve_part_1.py`

Part 2

DNS Exfiltration - Part 2

Challenge

Sensitive Data Exposure

Password Cracking

Forensics

Length: 19 mins

Difficulty: Medium

Description and resources

Mission brief

In the file that was captured in challenge DNS exfiltration - Part 1, find user John Doe and discover his password.

Instructions

Read the file, find what is needed. Use some well known password dictionaries.

Start exercise

Solution:

To solve the second part, we need to go back to description of part 1, and find out how DNS exfiltration works. We can find source code of DNSExfiltrator tool on github:

```
#---- Have we received all chunks of data ?
if chunkIndex == nbChunks:
    print '\n'
    try:
        # Create and initialize the RC4 decryptor object
        rc4Decryptor = RC4(args.password)

        # Save data to a file
        outputFileName = fileName + ".zip"
        print color("[+] Decrypting using password [{}]" .format(args.password,outputFileName))
        with open(outputFileName, 'wb+') as fileHandle:
            if useBase32:
                fileHandle.write(rc4Decryptor.binaryDecrypt(bytearray(fromBase32(fileData))))
            else:
                fileHandle.write(rc4Decryptor.binaryDecrypt(bytearray(fromBase64URL(fileData))))
            fileHandle.close()
            print color("[+] Output file [{}] saved successfully".format(outputFileName))
    except IOError:
        print color("[!] Could not write file [{}]" .format(outputFileName))
```

Here we can see, that on receiving side the data is firstly decoded from base32, then decrypted with RC4, and at the end we write data into file with .zip extension.

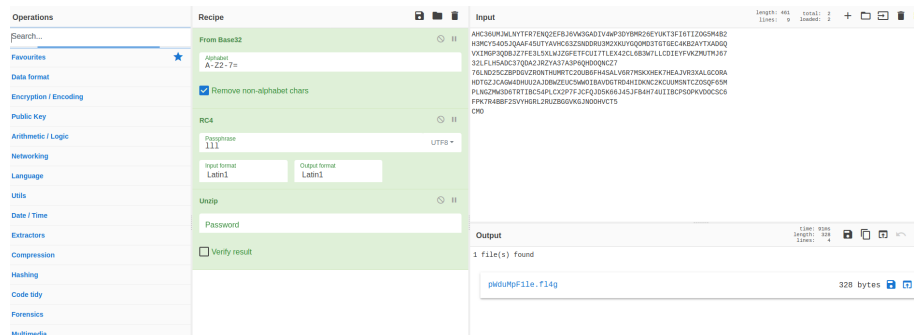
So to solve the challange we need to extract base32 encoded data from packets. Apply filter: **dns && ip.dst == 10.3.27.86**, and go to File -> Export Packet Dissections -> As Plain Text, from there we can extract base32 encoded strings, and join them into one:

```
AHC36UMJWLNYTFR7ENQ2EFBJ6VW3GADIV4WP3DYBMR26EYUKT3FI6TIZOG5M4B2
H3MCY5405JQA4F45UTYA VHC63ZSND DRU3M2XKUYGQOMD3TGTGEC4KB2AYTXADGQ
VXIMGP3QDBJZ7FE3L5XLWJZGFETF CUI7TLEX42CL6B3W7LLCDIEYFVKZMUTMJ67
32LFLH5ADC37QDA2JRZYA37A3P6QHDOQNCZ7
76LND25CZBPDGVZRONTHUMRTC2OUB6FH4SALV6R7MSKXHEK7HEAJVR3XALGCORA
HDTGZJCAGW4DHUU2AJDBWZEUC5WWOIBAVDGTDRD4HIDKNC2KCUUMSNTCZOSQF65M
PLNGZMW3D6TRTIBC54PLCX2P7FJCFQJD5K66J45JFB4H74UIIBCPSOPKVD0CSC6
FPK7R4BBF2SVYHGRL2RUZBGGVKGJN00HVCT5
CMO
```

Tshark is a terminal oriented version of Wireshark, so we also can use this tool to extract data from file:

```
tshark -r challengeDNS_anon.pcapng -Tfields -e dns.qry.name ip.addr==10.3.27.86 \
| sed -n '0~4p'
```

From the first part description, we know that the hacker uses simple password with three same letters to protect the exfiltration. So when we setup our cyberchef receip we can manually change letters (aaa, bbb, ccc...) for RC4 password, until we come to **lll** which will give us a valid zip file.



Inside the downloaded file we have this:

```
Administrator:500:NO PASSWORD*****:31D4CFE1D1DAE531B73C59D7E0C089C0:::
Guest:501:NO PASSWORD*****:NO PASSWORD*****:::
admin:1000:NO PASSWORD*****:0857632E42732DCC64201A80A05DD8D9:::
John Doe:1001:NO PASSWORD*****:515ABCC3B18D1C37B0EBF0238ED4261B:::
```

On the crackstation.net we can supply our hash for John Doe password:

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

S15ABCC3B1801C37B0EBF0238ED4261B

I'm not a robot

reCAPTCHA
Privacy · Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
S15ABCC3B1801C37B0EBF0238ED4261B	NTLM	mydreamsXOXO

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Our flag is: **mydreamsXOXO**

If someone would like to see how this part can be solved using Python, you can take look at file solve_part_2.py