

main

Go to file

> 1337UP_2022

> UTCTF_2022

> VishwaCTF_2022

> Cryptography/John_the_Rocker

> files

idrsa_id_rsa.docx

idrsa_id_rsa.txt

> img

README.md

> Forensic

> So_Forgetful

> files

Ghost.pcap

> img

README.md

LambdaMamba

Added writeup for VishwaCTF So Forgetful

8e4fe15 · 2 years ago

History

Name	Last commit message	Last commit date
..		
files	Added writeup for VishwaCTF So Forgetful	2 years ago
img	Added writeup for VishwaCTF So Forgetful	2 years ago
README.md	Added writeup for VishwaCTF So Forgetful	2 years ago

README.md

So Forgetful! (Category: Forensic)

The challenge is the following,

Challenge

390 Solves

×

Challenge

390 Solves

×

So Forgetful!

250

Once my friend was connected to my network, he did some office work and left. Next day he called me that he forgot his password, and wanted me to rescue him <3

Ghost.pcap

Flag

Submit

And we are given the file [Ghost.pcap](#). I used `tcp.stream` in the filters, and looked at the packets. Packet 112, which uses `HTTP` caught my attention because it uses `POST`. I looked into that packet and saw the following,

Time	Source	Destination	Protocol	Length	Info
100	135.651372	10.0.0.5	10.0.0.1	TCP	66 59162 → 8080 [ACK] Seq=128 Ack=1466 Win=32128 Len=0 TSva
101	135.651818	10.0.0.5	10.0.0.1	TCP	66 59162 → 8080 [ACK] Seq=128 Ack=1868 Win=35008 Len=0 TSva
102	135.651965	10.0.0.1	10.0.0.5	TCP	66 8080 → 59162 [FIN, ACK] Seq=1868 Ack=128 Win=28992 Len=0
103	135.657646	10.0.0.5	10.0.0.1	TCP	66 59162 → 8080 [FIN, ACK] Seq=128 Ack=1869 Win=35008 Len=0
104	135.657693	10.0.0.1	10.0.0.5	TCP	66 8080 → 59162 [ACK] Seq=1869 Ack=129 Win=28992 Len=0 TSva
107	150.097382	10.0.0.5	10.0.0.1	TCP	74 59163 → 8080 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_P
108	150.097451	10.0.0.1	10.0.0.5	TCP	74 8080 → 59163 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=
109	150.098106	10.0.0.5	10.0.0.1	TCP	66 59163 → 8080 [ACK] Seq=1 Ack=1 Win=29248 Len=0 TSval=418
110	150.098956	10.0.0.5	10.0.0.1	TCP	453 59163 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=29248 Len=387 TS
111	150.099075	10.0.0.1	10.0.0.5	TCP	66 8080 → 59163 [ACK] Seq=1 Ack=388 Win=30080 Len=0 TSval=4
112	150.099682	10.0.0.5	10.0.0.1	HTTP	107 POST /pages/main.html HTTP/1.1 (application/x-www-form-
113	150.099789	10.0.0.1	10.0.0.5	TCP	66 8080 → 59163 [ACK] Seq=1 Ack=429 Win=30080 Len=0 TSval=4
114	150.100188	10.0.0.1	10.0.0.5	TCP	83 8080 → 59163 [PSH, ACK] Seq=1 Ack=429 Win=30080 Len=17 T
115	150.101006	10.0.0.5	10.0.0.1	TCP	66 59163 → 8080 [ACK] Seq=429 Ack=18 Win=29248 Len=0 TSval=
116	150.101034	10.0.0.1	10.0.0.5	TCP	165 8080 → 59163 [PSH, ACK] Seq=18 Ack=429 Win=30080 Len=99
117	150.101464	10.0.0.5	10.0.0.1	TCP	66 59163 → 8080 [ACK] Seq=429 Ack=117 Win=29248 Len=0 TSval
118	150.102588	10.0.0.1	10.0.0.5	HTTP	66 HTTP/1.0 200 OK
119	150.107394	10.0.0.5	10.0.0.1	TCP	66 59163 → 8080 [FIN, ACK] Seq=429 Ack=118 Win=29248 Len=0
120	150.107437	10.0.0.1	10.0.0.5	TCP	66 8080 → 59163 [ACK] Seq=118 Ack=430 Win=30080 Len=0 TSval
121	163.781628	10.0.0.5	10.0.0.1	TCP	74 59164 → 8080 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_P
122	163.781696	10.0.0.1	10.0.0.5	TCP	74 8080 → 59164 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=
123	163.782512	10.0.0.5	10.0.0.1	TCP	66 59164 → 8080 [ACK] Seq=1 Ack=1 Win=29248 Len=0 TSval=421
124	163.783864	10.0.0.5	10.0.0.1	HTTP	193 GET /pages/tools.html HTTP/1.1

0000	08 00 27 3d 47 5d 08 00 27 38 2c 5c 08 00 45 00	..'=G].. '8,\..E.
0010	00 5d ec 95 40 00 40 06 3a 00 0a 00 00 05 0a 00]..@.@. :.....
0020	00 01 e7 1b 1f 90 e9 d7 8e c2 65 ad ce d8 80 18e.....
0030	01 c9 d2 3f 00 00 01 01 08 0a 00 06 61 ad 00 06	...?.a....
0040	7f d8 75 73 65 72 69 64 3d 73 70 69 76 65 79 70	..userid =spiveyp
0050	26 70 73 77 72 64 3d 53 30 34 78 57 6a 5a 51 57	&pswrd=S 04xWjZQW
0060	46 5a 35 4f 51 25 33 44 25 33 44	FZ50Q%3D %3D

I went to `Export Objects > HTTP` and selected packet 112,

Wireshark · Export · HTTP object list

Text Filter:

Content Type: All Content-Types

Packet	Hostname	Content Type	Size	Filename
9	10.0.0.1:8080		408 bytes	help.html
43	10.0.0.1:8080		408 bytes	help.html
65	10.0.0.1:8080		402 bytes	about.html
82	10.0.0.1:8080		1280 bytes	about.html
84	10.0.0.1:8080		303 bytes	about.html
99	10.0.0.1:8080		402 bytes	about.html
112	10.0.0.1:8080	application/x-www-form-urlencoded	41 bytes	main.html
130	10.0.0.1:8080		1254 bytes	tools.html
189	10.0.0.1:8080		73 bytes	tools.html
205	10.0.0.1:8080		99 bytes	status.html

Help

Preview

Save All

Close

Save

And I saved it as a .txt text file, which contained,

userid=spiveyp&pswrd=S04xWjZQWfZ50Q%3D%3D

I couldn't find anything information related to the website where these credentials were used, and the challenge didn't specify any server to log in to. Therefore, I assumed that the password `S04xWjZQWfZ50Q%3D%3D` itself would be the flag (after processing it more). The `%3D%3D` at the end of the flag looked like [Base64 in URL](#).

The URL applications [\[edit \]](#)

Base64 encoding can be helpful when fairly lengthy identifying information is used in an HTTP environment. For example, a database persistence framework for [Java](#) objects might use Base64 encoding to encode a relatively large unique id (generally 128-bit [UUIDs](#)) into a string for use as an HTTP parameter in HTTP forms or HTTP GET [URLs](#). Also, many applications need to encode binary data in a way that is convenient for inclusion in URLs, including in hidden web form fields, and Base64 is a convenient encoding to render them in a compact way.

Using standard Base64 in [URL](#) requires encoding of `' + ' / ' and '='` characters into special [percent-encoded](#) hexadecimal sequences (`' + '` becomes `'%2B'`, `' / '` becomes `'%2F'` and `' = '` becomes `'%3D'`), which makes the string unnecessarily longer.

For this reason, [modified Base64 for URL](#) variants exist (such as [base64url](#) in [RFC 4648](#)), where the `' + '` and `' / '` characters of standard Base64 are respectively replaced by `' - '` and `' _ '`, so that using [URL encoders/decoders](#) is no longer necessary and has no effect on the length of the encoded value, leaving the same encoded form intact for use in relational databases, web forms, and object identifiers in general. A popular site to make use of such is [YouTube](#).^[1] Some variants allow or require omitting the padding `' = '` signs to avoid them being confused with field separators, or require that any such padding be percent-encoded. Some libraries^[which?] will encode `' = '` to `' %3D '`, potentially exposing applications to relative path attacks when a folder name is encoded from user data.

So I converted it to the standard Base64 format by replacing `%3d` with `=`, to give,

S04xWjZQWfZ50Q==

Finally, I went to [CyberChef](#) to convert it from Base64, which gave,

KN1Z6PXVy9

Therefore, I assumed that this would be the flag,

vishaCTF{KN1Z6PXVy9}

And submitting it confirmed that this was the flag.