# Name-Chirantan Sawant

# Intern ID -286

# 🌐 Natas Wargame – Complete Case Study

---

## 🔹 Level 0 → Level 1

**Objective:** Find password in page source.
**Tools:** Browser (View Source).
**Steps:** Open DevTools → see password in HTML comment.
**Conclusion:** Sensitive data can leak in source code.

---

## 🔹 Level 1 → Level 2

**Objective:** Hidden password inside HTML.
**Tools:** DevTools (Elements tab).
**Steps:** Inspect hidden element → password found.
**Conclusion:** Hidden fields are not secure.

---

## 🔹 Level 2 → Level 3

**Objective:** Password in `/files/` directory.
**Tools:** `curl`.
**Steps:**

```
curl http://natas2.natas.labs.overthewire.org/files/users.txt
```

**Conclusion:** Exposed directories = data leakage.

---

## 🔹 Level 3 → Level 4

**Objective:** Access hidden `/s3cr3t/` folder.
**Tools:** URL manipulation, `wget`.
**Steps:** Visit `/s3cr3t/users.txt`.
**Conclusion:** Security through obscurity fails.

---

## 🔹 Level 4 → Level 5

**Objective:** Manipulate cookies for authentication.
**Tools:** DevTools → Application → Cookies.
**Steps:** Change `loggedin=0` → `1`. Refresh → password revealed.
**Conclusion:** Weak cookie-based access control.

---

## 🔹 Level 5 → Level 6

**Objective:** Bypass Referer check.
**Tools:** `curl`.
**Steps:**

```
curl -H "Referer: http://natas5.natas.labs.overthewire.org/" ...
```

**Conclusion:** Headers can be forged easily.

---

## 🔹 Level 6 → Level 7

**Objective:** Read included secret file.
**Tools:** Browser, wget.
**Steps:** Access /includes/secret.inc.
**Conclusion:** Sensitive includes must be protected.

---

## 🔹 Level 7 → Level 8

**Objective:** Exploit weak input validation.
**Tools:** Burp Suite.
**Steps:** Submit username=admin&password=admin.
**Conclusion:** Input validation is critical.

---

## 🔹 Level 8 → Level 9

**Objective:** Base64 decode.
**Tools:** base64, Python.
**Steps:**

```
echo "YWRtaW4=" | base64 -d
```

**Conclusion:** Encoding ≠ encryption.

---

## 🔹 Level 9 → Level 10

**Objective:** Command injection.
**Tools:** curl.
**Steps:** Inject:

```
needle=admin; cat /etc/natas_webpass/natas11
```

**Conclusion:** Sanitization prevents injections.

---

## 🔹 Level 10 → Level 11

**Objective:** Modify XOR-encrypted cookies.
**Tools:** Python.
**Steps:** Write XOR decode script, edit cookie, re-encrypt.
**Conclusion:** Weak encryption = easy bypass.

---

## 🔹 Level 11 → Level 12

**Objective:** File upload bypass.
**Tools:** Burp Suite.
**Steps:** Upload PHP disguised as image. Access → run `cat` command.
**Conclusion:** File upload filters are bypassable.

---

## 🔹 Level 12 → Level 13

**Objective:** Bypass image MIME checks.
**Tools:** ExifTool.
**Steps:** Add PHP code to JPG metadata.
**Conclusion:** Metadata manipulation can bypass validation.

---

## 🔹 Level 13 → Level 14

**Objective:** SQL Injection.
**Tools:** `sqlmap`.
**Steps:** Inject:

```
username=admin" --
```

**Conclusion:** Always sanitize SQL input.

---

## 🔹 Level 14 → Level 15

**Objective:** Blind SQL Injection (time-based).
**Tools:** Python script with requests + sleep().
**Steps:** Extract password char by char.
**Conclusion:** Timing attacks reveal hidden data.

---

## 🔹 Level 15 → Level 16

**Objective:** Same as above, refine script.
**Conclusion:** Practice in blind SQL injection.

---

## 🔹 Level 16 → Level 17

**Objective:** Time-based SQL injection again.
**Tools:** `curl`, Python.
**Conclusion:** Reinforcement of blind SQLi skills.

---

## 🔹 Level 17 → Level 18

**Objective:** Predict session IDs.
**Tools:** Python + cookies.
**Steps:** Iterate session IDs → find admin.
**Conclusion:** Session IDs must be random.

---

## 🔹 Level 18 → Level 19

**Objective:** Exploit session fixation.
**Tools:** URL manipulation.
**Steps:** Use `?PHPSESSID=admin`.
**Conclusion:** Session fixation = privilege escalation.

---

## 🔹 Level 19 → Level 20

**Objective:** Abuse experimenter page.
**Tools:** ZAP proxy.
**Steps:** Modify parameters → admin access.
**Conclusion:** Auxiliary pages can be dangerous.

---

## 🔹 Level 20 → Level 21

**Objective:** Bypass redirect.
**Tools:** curl.
**Steps:**

```
curl --location-trusted ...
```

**Conclusion:** Server-side redirects must be secure.

---

## 🔹 Level 21 → Level 22

**Objective:** PHP type juggling (0e...).
**Tools:** curl.
**Steps:** Submit password=0e12345.
**Conclusion:** Loose comparisons are insecure.

---

## 🔹 Level 22 → Level 23

**Objective:** More PHP type juggling.
**Steps:** Submit password[]=1.
**Conclusion:** Arrays break strcmp() logic.

---

## 🔹 Level 23 → Level 24

**Objective:** Log poisoning + file inclusion.
**Tools:** curl, Burp Suite.

**Steps:** Inject PHP in User-Agent, include log.
**Conclusion:** Logs can be exploited.

---

## 🔹 Level 24 → Level 25

**Objective:** PHP object injection.
**Tools:** Python (pickle).
**Steps:** Craft malicious serialized object.
**Conclusion:** Never unserialize user input.

---

## 🔹 Level 25 → Level 26

**Objective:** SQL Injection again.
**Steps:** `username=admin' OR 1=1--`.
**Conclusion:** Classic SQL injection flaw.

---

## 🔹 Level 26 → Level 27

**Objective:** Command injection in Perl script.
**Tools:** Shell injection.
**Steps:** `; cat /etc/natas_webpass/natas29`.
**Conclusion:** Perl backticks are unsafe.

---

## 🔹 Level 27 → Level 28

**Objective:** Regex authentication bypass.
**Steps:** Use `(.*)` to bypass regex.
**Conclusion:** Bad regex = insecure.

---

## 🔹 Level 28 → Level 29

**Objective:** Environment variable manipulation.
**Steps:** `export USER=admin`.
**Conclusion:** Don't trust env variables for auth.

---

## 🔹 Level 29 → Level 30

**Objective:** File descriptor manipulation.
**Steps:** Redirect to password file.
**Conclusion:** FD mismanagement is risky.

---

## 🔹 Level 30 → Level 31

**Objective:** Reverse engineer Perl serialization.
**Tools:** Burp Suite, Perl.
**Steps:** Craft payload:

```
serialize({cmd => 'cat /etc/natas_webpass/natas33'})
```

**Conclusion:** Advanced code review + serialization exploitation.

---

## 🔹 Level 31 → Level 32

**Objective:** Login as root.
**Steps:** Analyze traffic in Burp → exploit serialized input.
**Conclusion:** Final step proves mastery of serialization + web exploitation.

---

## 🔹 Level 33 → Level 34

**Objective:** Confirm game completion.
**Steps:** Login shows congratulatory message.
**Conclusion:** Level 33 is the last Natas level.

---

# ✅ Final Conclusion

- **Early Levels (0–8):** HTML, cookies, headers, encoding.
- **Mid Levels (9–20):** Injections, file upload, SQLi, session hijacking.
- **Advanced Levels (21–33):** Type juggling, serialization, log poisoning, environment exploitation.

👉 **Natas builds real penetration testing skills step by step** – from viewing source code to advanced code injection attacks.