Cybersecurity Challenge Report Team Contributors: 1. Sneha – Krypton Challenges

1. Krypton Challenges (Windows Command Prompt Execution)

Level 0 → Level 1

Tools Used:

- `type` (Windows equivalent of `cat`)
- PowerShell ROT13 decoding
- `echo`
- SSH via Windows OpenSSH or PuTTY

Objective:
Decode a ROT13-encrypted message stored in a file to extract the password.

Steps Followed:

Opened Command Prompt and connected via SSH:

```
ssh krypton0@krypton.labs.overthewire.org -p 2222
```

1. Navigated to the `/krypton` directory and viewed the file:

   ```
   krypton0
   ```
2. Output:

3. ```
   YRIRY GJB CNFFIBEQ EBGGRA
   ```
   ```

4. Recognized it as ROT13 and decoded in PowerShell:

```
   "YRIRY GJB CNFFIBEQ EBGGRA" -split '' | ForEach-Object {
```

```
    if ($_ -match '[A-Z]') {
        [char](( [byte][char]$_ - 65 + 13) % 26) + 65)
    } else { $_ }
} -join ''
```

5. Output:

```
   LEVEL TWO PASSWORD ROTTEN
```

6. Used `ROTTEN` to log into Level 1:

```
   ssh krypton1@krypton.labs.overthewire.org -p 2222
```

Conclusion:
Applied ROT13 decoding using PowerShell instead of Linux `tr`.

Level 1 → Level 2

Tools Used:

- `type`
- PowerShell ROT13 function
- SSH client

Objective:
Decrypt another ROT13 message to obtain the password.

Steps Followed:

Viewed the file:

```
type krypton1
```

1. Created a reusable PowerShell function:

```
   function ROT13($text) {
```

```
    return ($text -split '' | ForEach-Object {
        if ($_ -match '[A-Z]') {
            [char](((  [byte][char]$_ - 65 + 13) % 26) + 65)
        } else { $_ }
    }) -join ''
}
```

2. Ran:

```
ROT13 (Get-Content krypton1)
```

3. Extracted the password and logged into Level 2.

Conclusion:
Replaced Linux `tr` with a PowerShell ROT13 function.

Level 2 → Level 3

Tools Used:

- `type`
- PowerShell `-replace` method

Objective:
Perform ROT13 decryption on a longer string.

Steps Followed:

Viewed file:

```
krypton2
```

1. Decoded in PowerShell without splitting:

```
(Get-Content krypton2) -replace '([A-Z])', {
    [char](((  [byte][char]$args[0].Value - 65 + 13) % 26) + 65)
}
```

2. Retrieved password and logged into Level 3.

Conclusion:
Practiced quick ROT13 decoding without breaking text into characters.

Level 3 → Level 4

Tools Used:

- `type`
- PowerShell ROT13
- `findstr` (CMD equivalent of `grep`)

Objective:
Decode ROT13 message and filter for password.

Steps Followed:

Decoded and filtered in one command:

```
(Get-Content krypton3) -replace '([A-Z])', {
    [char](( [byte][char]$args[0].Value - 65 + 13) % 26) + 65)
} | findstr password
```

1. Found password and logged into Level 4.

Conclusion:
Combined decoding and searching like Linux `grep`.

Level 4 → Level 5

Tools Used:

- Sysinternals `strings.exe`

- `findstr`
- Running `.exe` in CMD

Objective:
Extract password from a compiled binary.

Steps Followed:

Ran strings search:

```
strings.exe krypton4.exe | findstr /I pass
```

1. Executed the binary:

   ```
   krypton4.exe
   ```
2. Entered found string to reveal password.

Conclusion:
Used Windows tools to replace Linux `strings` and `objdump`.

Level 5 → Level 6

Tools Used:

- `strings.exe`
- HxD Hex Editor
- CMD execution

Objective:
Analyze binary for hidden logic.

Steps Followed:

Ran:

```
strings.exe krypton5.exe
```

1. Opened binary in **HxD** to inspect raw data.
2. Ran the program and tried suspected passwords until success.

Conclusion:
Applied hex inspection and dynamic testing on Windows.

Level 6 → Level 7

Tools Used:

- `strings.exe`
- PowerShell brute-force loop

Objective:
Brute-force obfuscated binary to find password.

Steps Followed:

Found hints using:

```
strings.exe krypton6.exe
```

1. Wrote PowerShell brute-force script:

```
   for ($i = 0; $i -lt 1000; $i++) {

  $pin = "{0:D3}" -f $i
  $output = & .\krypton6.exe $pin
  if ($output -match "success") {
      Write-Host "Password found: $pin"
      break
  }
}
```

2. Script found correct PIN and displayed password.

Conclusion:

Simulated Linux bash brute-force in PowerShell.