

# OverTheWire Vortex [ Level - 0 ]

This is an elaborate each level oriented write-up for the Vortex wargame from OverTheWire.org. These challenges provide invaluable hands-on learning experiences in cybersecurity and exploitation techniques. If you find these resources helpful, please consider supporting the OverTheWire team who create and maintain these educational platforms—they're doing important work making security education accessible to everyone.

**Donate at:** <https://overthewire.org/information/donate.html>

**Author:** Jinay Shah

**Tools Used:**

- Kali Linux
- Python

## Vortex Level 0

### Level info:

#### Level Goal:

Your goal is to connect to port 5842 on `vortex.labs.overthewire.org` and read in 4 unsigned integers in host byte order. Add these integers together and send back the results to get a username and password for vortex1. This information can be used to log in using SSH.

**Note:** vortex is on an 32bit x86 machine (meaning, a little endian architecture)

Helpful Reading Material:

- [1. C Programming Introduction](#)
- [2. Network Programming Tutorial](#)

### Solution:

Even before beginning with this whole challenge, I would like to mention I have my hands on scripting, C programming and understanding Assembly- but Python networking is truly an uncharted territory for me, this isn't easy for me, I will be using the help of writeups and resources as and when I require, although I will be transparent about it. No then let's get into it.

Level 0 seems pretty straight forward I think, we have 4 integers in byte order- we need to send the summation of them back to port 5842 of the remote server and in return we will get the password for the next level.

Let's try building a python script, beginning with setting up a connection first:

```
import socket
import struct

HOST = "vortex.labs.overthewire.org"
PORT = 5842
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST,PORT))
```

#### `import socket`

- Loads Python's socket library.
- Sockets are endpoints for network communication.
- This allows the script to create TCP/UDP clients/servers.

#### `import struct`

- Handles packing/unpacking binary data.
- Useful when dealing with C-style binary formats.
- Vortex levels often send integers in binary form — `struct` converts them into Python ints.

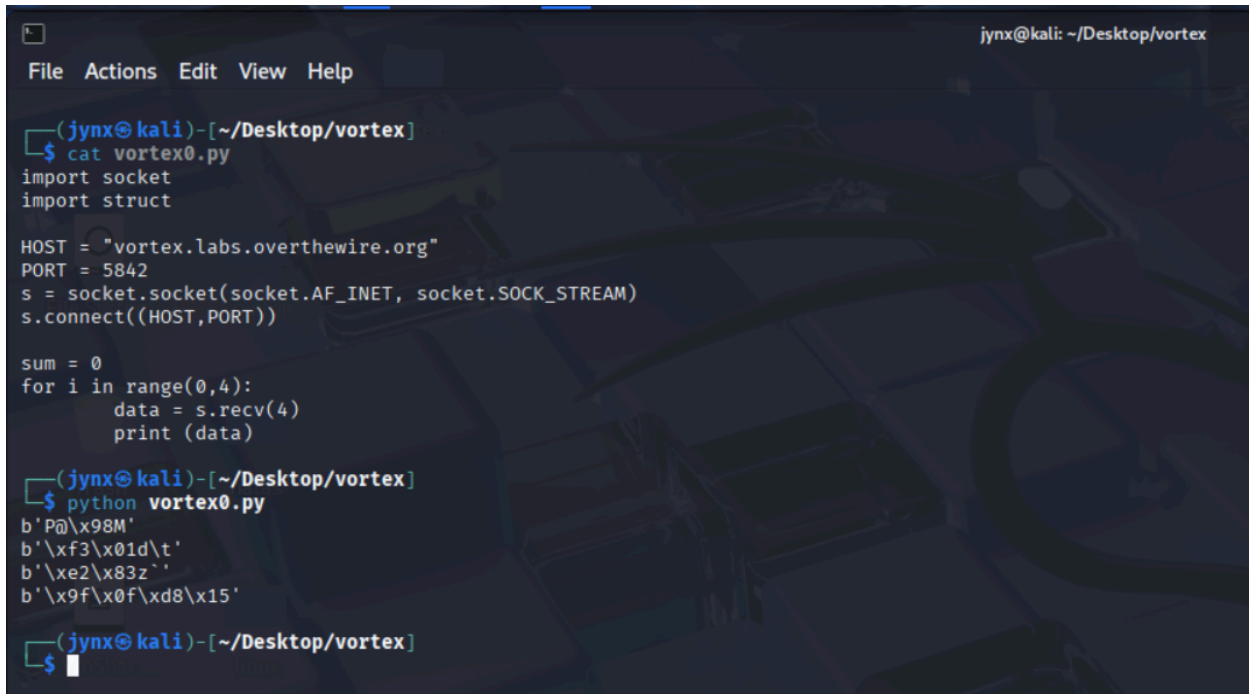
#### `HOST`

- The domain name of the OverTheWire Vortex wargame server.

#### `PORT`

- Port number for the specific level.

Let's now try building the logic for retrieving the numbers and summing them up, since the level does mention 4 integers, an integers size typically being 4 Bytes or 32 bits, let's see what it fetches:



```
(jynx@kali) - [~/Desktop/vortex]
$ cat vortex0.py
import socket
import struct

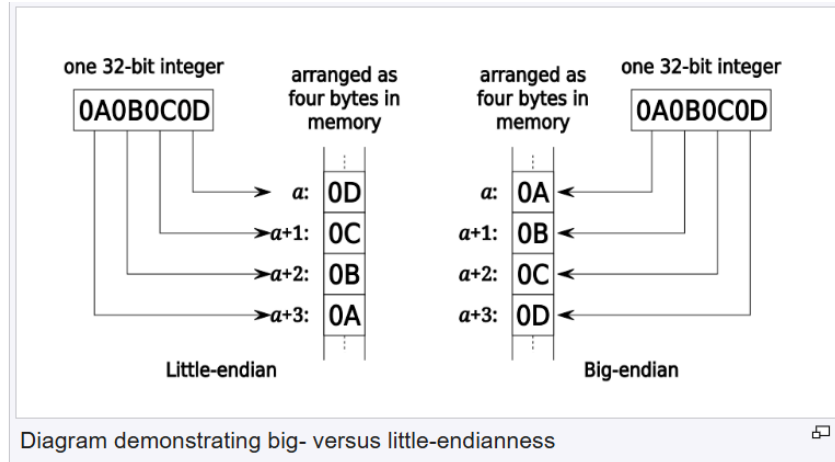
HOST = "vortex.labs.overthewire.org"
PORT = 5842
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))

sum = 0
for i in range(0,4):
    data = s.recv(4)
    print (data)

(jynx@kali) - [~/Desktop/vortex]
$ python vortex0.py
b'P@\x98M'
b'\xf3\x01d\t'
b'\xe2\x83z`'
b'\x9f\x0f\xd8\x15'
```

It is definitely NOT something I understand lol, but I think I know what to look up: little endian architecture, lets look that up to get some idea:

In **computing**, **endianness** is the order in which **bytes** within a **word data type** are transmitted over a **data communication** medium or addressed in **computer memory**, counting only byte **significance** compared to earliness. Endianness is primarily expressed as **big-endian (BE)** or **little-endian (LE)**.



Computers store information in various-sized groups of binary bits. Each group is assigned a number, called its *address*, that the computer uses to access that data. On most modern computers, the smallest data group with an address is eight bits long and is called a byte. Larger groups comprise two or more bytes, for example, a **32-bit** word contains four bytes.

There are two principal ways a computer could number the individual bytes in a larger group, starting at either end. A big-endian system stores the **most significant byte** of a word at the smallest **memory address** and the **least significant byte** at the largest. A little-endian system, in contrast, stores the least-significant byte at the smallest address.<sup>[1][2][3]</sup> Of the two, big-endian is thus closer to the way the digits of numbers are written left-to-right in English, comparing digits to bytes and assuming addresses increase from left to right.

That does remind me of something I read in of the classes during undergrad, nonetheless, I think I will look up some write-up that could help me make sense of this gibberish numbers to me.

Okay I got my next lead hereon forth, two in fact, we need to unpack the raw bytes from little endian format to decimal and then add or sum it up and send it back to server and we should ideally get the password for the next level, let's try my script and see if it works- I'll explain the workings below and give a code breakdown as well:

Here's how we proceed:

```
import socket
import struct
```

```

HOST = "vortex.labs.overthewire.org"
PORT = 5842
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST,PORT))

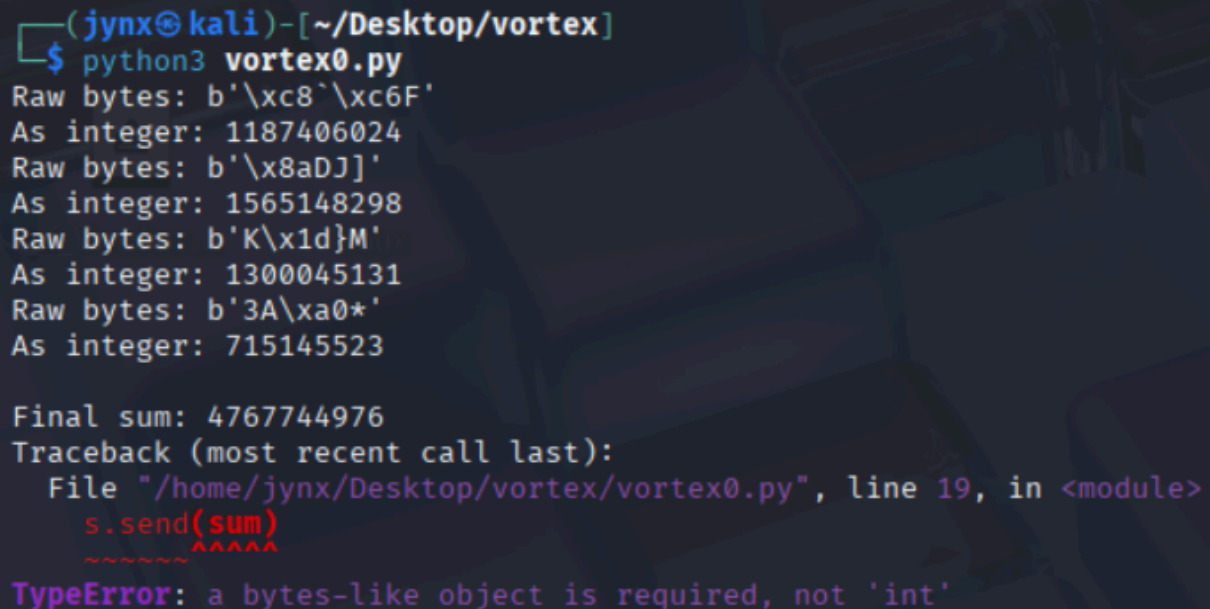
sum = 0
for i in range(0,4):
    data = s.recv(4)
    print(f"Raw bytes: {data}")

    number = struct.unpack("<I", data)[0]
    print(f"As integer: {number}")
    sum += number

print(f"\nFinal sum: {sum}")
s.send(sum) # Requires to send it to the server in little endian format [petty mi
stake]

response = s.recv(1024)
print(f"Server response: {response}")

```



```

(jynx@kali)-[~/Desktop/vortex]
$ python3 vortex0.py
Raw bytes: b'\xc8`\xc6F'
As integer: 1187406024
Raw bytes: b'\x8aDJ]'
As integer: 1565148298
Raw bytes: b'K\x1d}M'
As integer: 1300045131
Raw bytes: b'3A\xa0*'
As integer: 715145523

Final sum: 4767744976
Traceback (most recent call last):
  File "/home/jynx/Desktop/vortex/vortex0.py", line 19, in <module>
    s.send(sum)
    ~~~~~^~~~~
TypeError: a bytes-like object is required, not 'int'

```

Let's try again:

```
import socket
import struct

HOST = "vortex.labs.overthewire.org"
PORT = 5842
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST,PORT))

sum = 0
for i in range(0,4):
    data = s.recv(4)
    print(f"Raw bytes: {data}")

    number = struct.unpack("<I", data)[0]
    print(f"As integer: {number}")
    sum += number

print(f"\nFinal sum: {sum}")
s.send(struct.pack("<I",sum))          #corrected to little endian formatting

response = s.recv(1024)
print(f"Server response: {response}")
```

```
(jynx@kali)-[~/Desktop/vortex]
$ python3 vortex0.py
Raw bytes: b'W\x15>+'
As integer: 725488983
Raw bytes: b'Jj2\x13'
As integer: 322071114
Raw bytes: b'\x1a\xbe`6'
As integer: 912309786
Raw bytes: b'\x08\x8d\xc6\x12'
As integer: 315002120

Final sum: 2274872003
Server response: b'Username: vortex1 Password: ██████████'
```

Unlike previous stages this time around I will not be revealing the password itself, but well I will illustrate how I achieved it if I do that is :)

Let me breakdown the rest of the code:

And there we go, we can finally login to our next vortex level:

```
sum = 0
# Initialize accumulator variable to store the total sum of all received numbers

for i in range(0,4):
# Loop 4 times (i = 0, 1, 2, 3) because the server sends exactly 4 numbers

    data = s.recv(4)
    # Receive 4 bytes from the server socket (one 32-bit integer in binary format)

    print(f"Raw bytes: {data}")
    # Display the raw binary data (e.g., b'\xc8\xc6F') for debugging

    number = struct.unpack("<I", data)[0]
    # Convert 4 bytes to unsigned integer using little-endian byte order
    # "<I" format: < = little-endian, I = unsigned 32-bit int
    # [0] extracts the first element from the returned tuple
```

```

print(f"As integer: {number}")
# Display the converted decimal number for verification

sum += number
# Add the current number to the running total

print(f"\nFinal sum: {sum}")
# Display the total sum of all 4 numbers received

s.send(struct.pack("<I", sum))
# Convert the integer sum back to 4 bytes in little-endian format and send to s
# server
# struct.pack() is the reverse of struct.unpack() - converts int → bytes

response = s.recv(1024)
# Receive up to 1024 bytes from the server
# This captures the server's response (likely the password or success message)
# 1024 is the buffer size - large enough to hold text responses of unknown length

print(f"Server response: {response}")
# Display the server's response (typically contains the password for the next level)
# Output will be in bytes format (e.g., b'Correct! Password: xyz123\n')

```

And finally this level comes to an end, below is login session from vortex level 1;



```
(jynx@kali)-[~/Desktop/vortex]  
$ ssh vortex1@vortex.labs.overthewire.org -p 2228
```



File System: /usr/bin/ls

Vortex

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

backend: gibson-1  
vortex1@vortex.labs.overthewire.org's password:

OverTheWire

Welcome to OverTheWire!

```
File Actions Edit View Help
vortex1@vortex:/vortex$ ls -l
total 412
-rwxr-xr-x 1 root      root      15672 Oct 14 09:29 vortex0
-r-sr-x--- 1 vortex2   vortex1   15072 Oct 14 09:29 vortex1
-r-sr-x--- 1 vortex11  vortex10 15356 Oct 14 09:29 vortex10
-r-sr-x--- 1 vortex12  vortex11 20664 Oct 14 09:29 vortex11
-r-sr-x--- 1 vortex13  vortex12 15224 Oct 14 09:29 vortex12
-r-sr-x--- 1 vortex14  vortex13 15204 Oct 14 09:29 vortex13
-r----- 1 vortex14  vortex14   323 Oct 14 09:29 vortex14.txt
-r-xr-x--- 1 vortex15  vortex15 15072 Oct 14 09:29 vortex15
-r----- 1 vortex15  vortex15   469 Oct 14 09:29 vortex15.tar.Z.enc
-r-sr-x--- 1 vortex17  vortex16 19804 Oct 14 09:29 vortex16
-r-sr-x--- 1 vortex18  vortex17 14908 Oct 14 09:29 vortex17
-r-sr-x--- 1 vortex19  vortex18 15372 Oct 14 09:29 vortex18
-r-sr-x--- 1 vortex20  vortex19 15056 Oct 14 09:29 vortex19
-r-sr-x--- 1 vortex3   vortex2   14860 Oct 14 09:29 vortex2
-r-xr-x--- 1 root      vortex21 15680 Oct 14 09:29 vortex20
-r-sr-x--- 1 vortex22  vortex21 15452 Oct 14 09:29 vortex21
-r-sr-x--- 1 vortex23  vortex22 16468 Oct 14 09:29 vortex22
-r----- 1 vortex22  vortex22   328 Oct 14 09:29 vortex22_factor.o
-r----- 1 vortex22  vortex22    419 Oct 14 09:29 vortex22_md5.h
-r----- 1 vortex22  vortex22   3964 Oct 14 09:29 vortex22_md5.o
-rw-r----- 1 vortex23  vortex23 14691 Oct 14 09:29 vortex23.jpg
-r-sr-x--- 1 vortex25  vortex24 15532 Oct 14 09:29 vortex24
-r-sr-x--- 1 vortex4   vortex3   11272 Oct 14 09:29 vortex3
-r-sr-x--- 1 vortex4   vortex3   11272 Oct 14 09:29 vortex3-hard
-r-sr-x--- 1 vortex5   vortex4   14848 Oct 14 09:29 vortex4
-r-sr-x--- 1 vortex6   vortex5   15432 Oct 14 09:29 vortex5
-r-sr-x--- 1 vortex7   vortex6   14988 Oct 14 09:29 vortex6
-r-sr-x--- 1 vortex8   vortex7   14960 Oct 14 09:29 vortex7
-r-sr-x--- 1 vortex9   vortex8   15156 Oct 14 09:29 vortex8
vortex1@vortex:/vortex$
```

I'll attempt the next stage i.e. vortex level 2 tomorrow.

## Resources:

1. Wikipedia  
<https://en.wikipedia.org/wiki/Endianness>

2. Hammer Of Thor

<https://mcpa.github.io/vortex/wargame/web/overthewire/2015/09/30/vortex00/>

3. Axtaxt's Blog

<https://axtaxt.wordpress.com/2010/11/14/overthewire-vortex-level0/>