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.

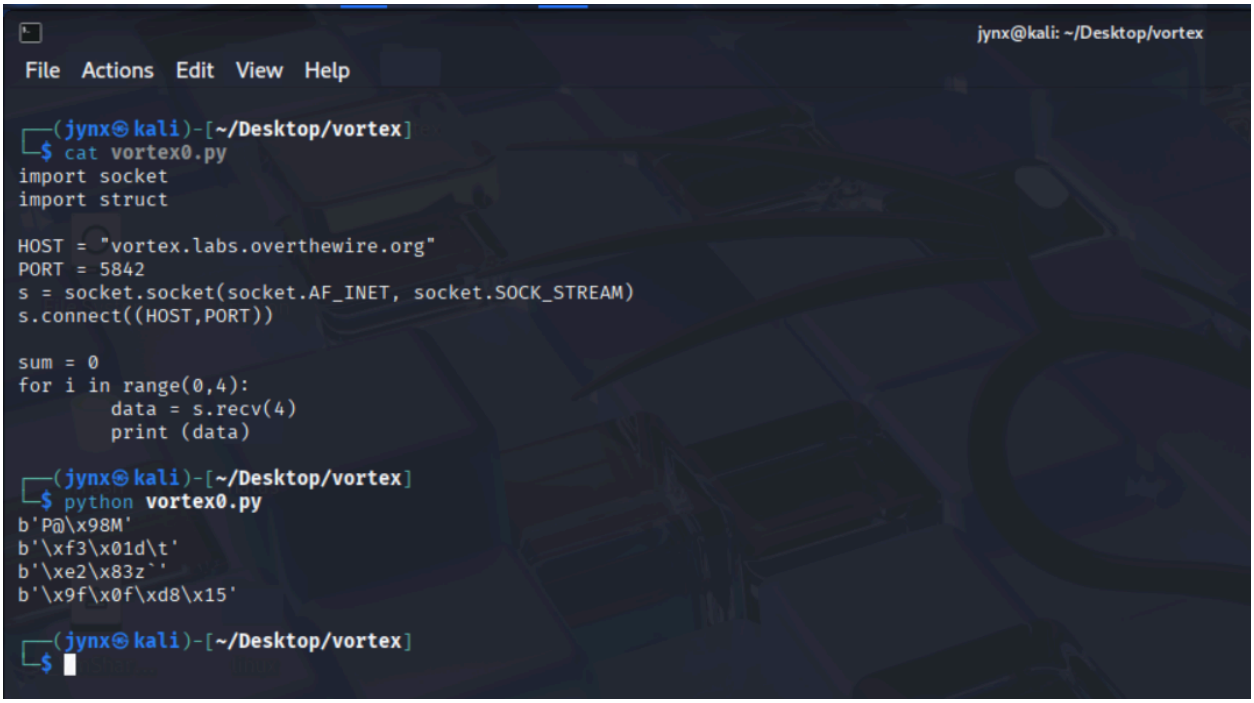
`HOST`

- 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:



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**).

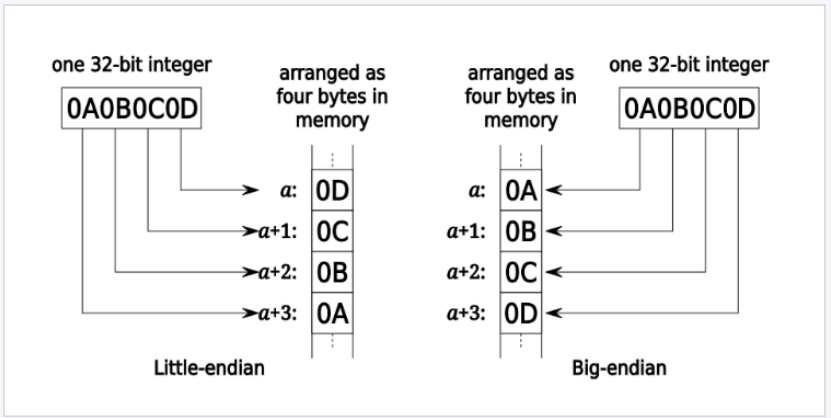


Diagram demonstrating big- versus little-endianness

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.[1][2][3] 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.

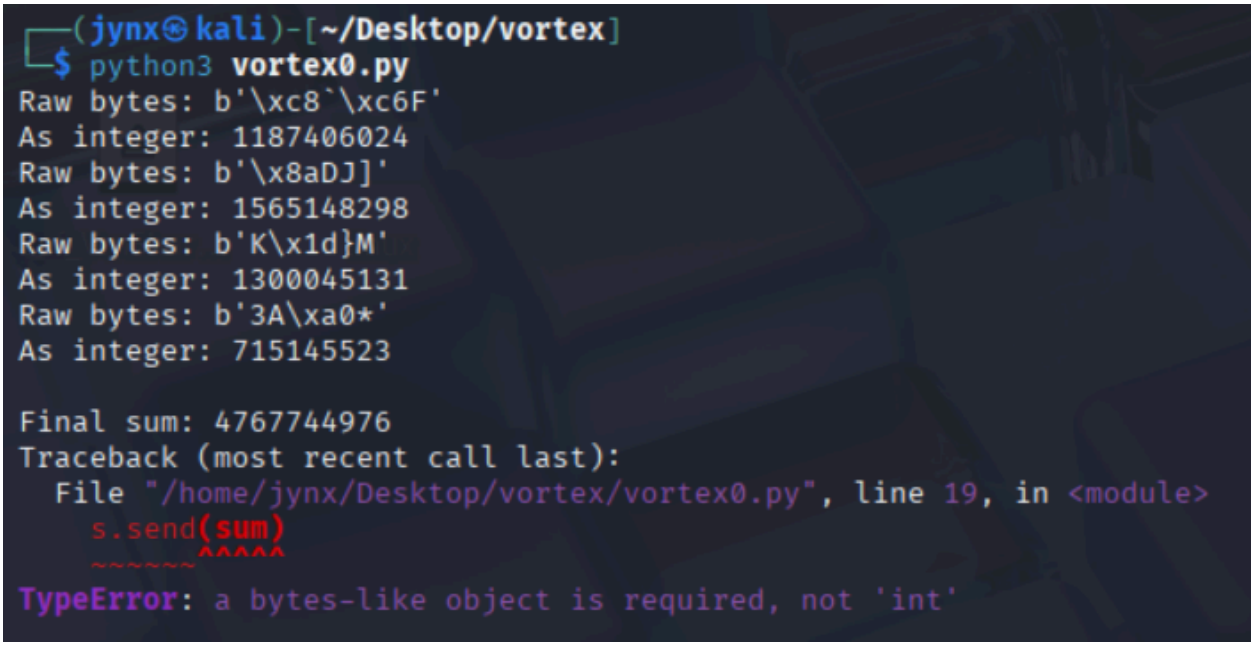
That does remind me of something I read in of the classes during undergrad, nonetheless, I think I will look up some write-up that could help me make sense of this gibberish numbers to me.

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:

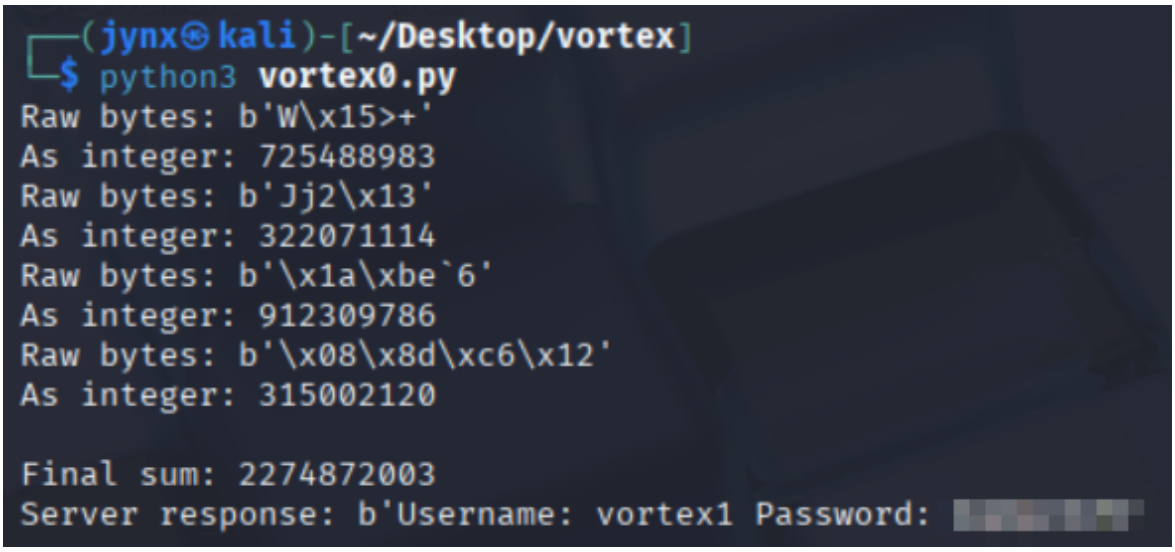
```
import socket
import struct
```

```
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}")
```



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}")
```



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 :)

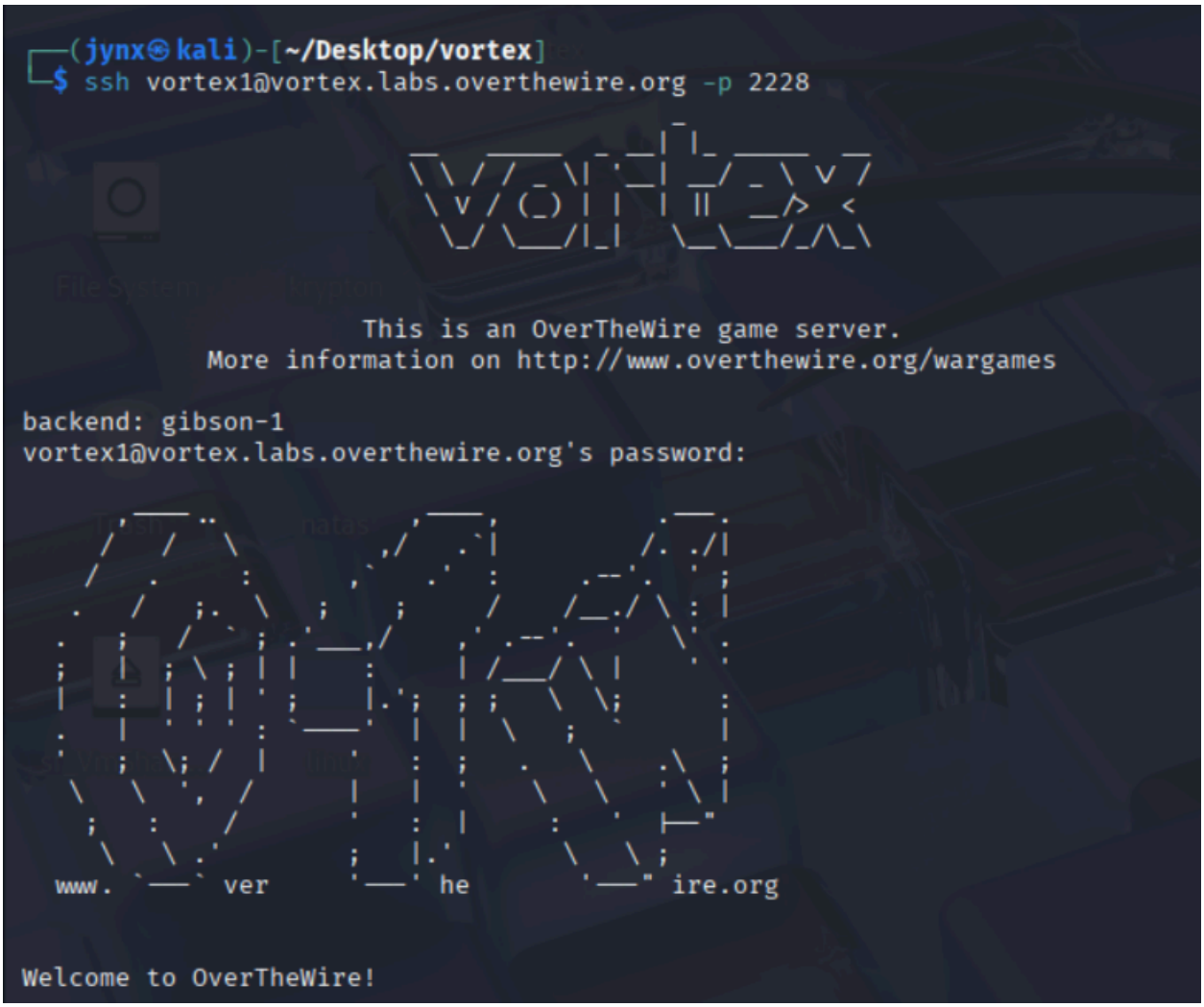
Let me breakdown the rest of the code:

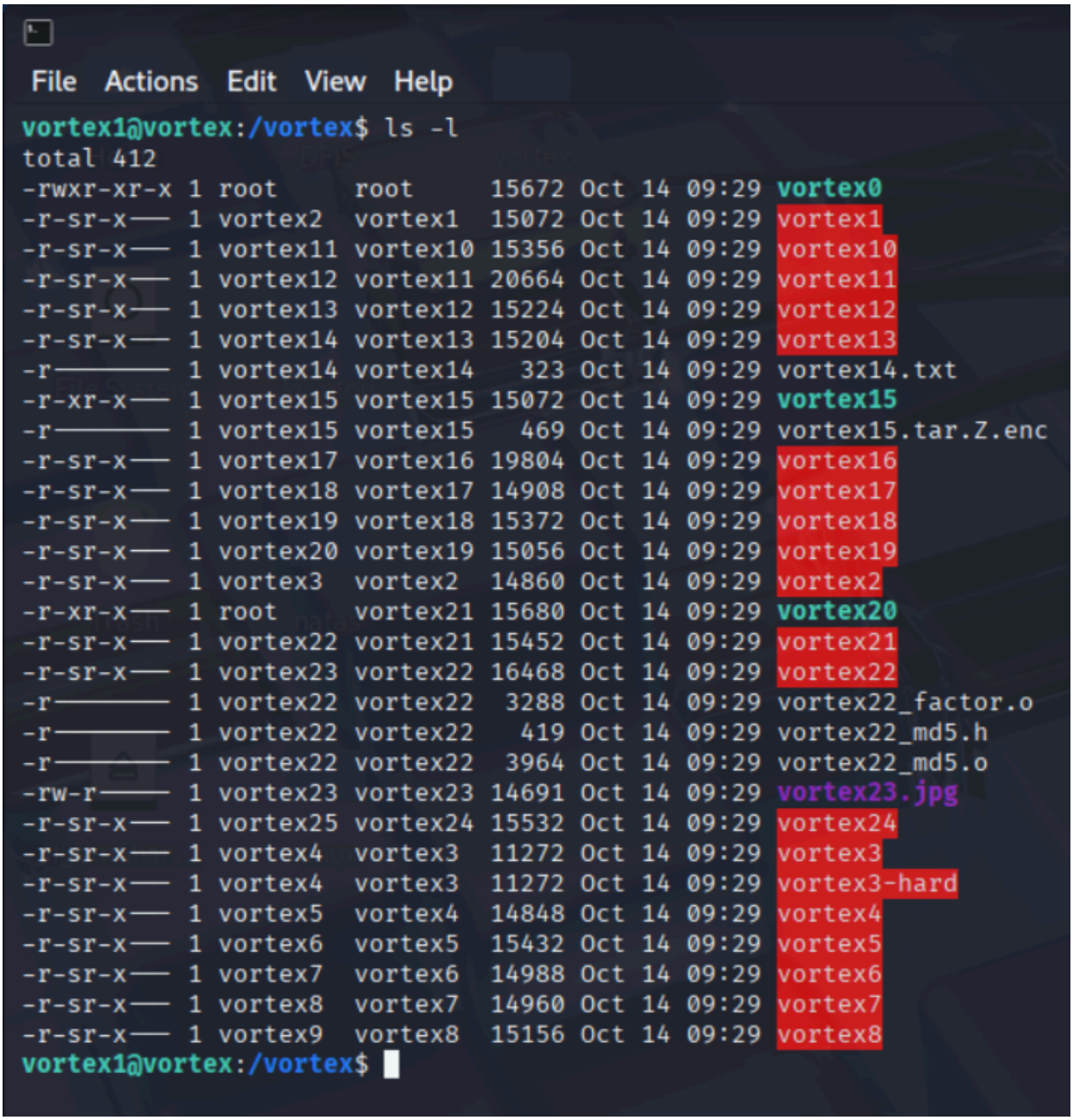
And there we go, we can finally login to our next vortex level:

```
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 forma
t)

    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
erver
# 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 messag
e)
# 1024 is the buffer size - large enough to hold text responses of unknown len
gth

print(f"Server response: {response}")
# Display the server's response (typically contains the password for the next le
vel)
# 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
This is an OverTheWire game server.
More information on http://www.overthewire.org/wargames
backend: gibson-1
vortex1@vortex.labs.overthewire.org's password:
Welcome to OverTheWire!



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/