What should I do with the following?

0101111010101000010101000011110111111001001100101010110001000001100011110111111011011110000110100111011001100101010110001000001100001110111011001000100000110010111110001100101110001111101011110100110010101000001110100110100011001010100000110011011011111101100110110011011111110111110100111011101100111010000011000111101111110010011001010100000110111111011100100000110000101000001110110110000111100101101001110010111101001111001010000011011111100110010000011011111110000110010111100101100001111010011010011101110110011101000001110011111100111100111110100110010111011011110011010000001010001100001111010001000001110100110100011001010100000111011011001011110010111100101000001101100110010111000011110011111010001000001110100111001011110010100000110000101000001110110110010111100101110011110100111011111101110010000011011111100110010000010011001101001110111011101011111000010000011000011101110110010001000001100001010000011101101100101111001011100111101001110111111011100100000110111111001100100000101011111010011101110110010011011111110111111001101010010101110010000001000001000011110111111011011101101110010111011101110100010000011011111101110010000011110011101111111010111100100100000110111111000101110011110010111100101110110110000111101001101001110111111011101110011010111001000000100000101010011010001100101110111001000001100011110111111011011101101110010111011101110100010000011011111101110010000011101111101000110000111101000100000111100111011111110101010000011101001101000110100111011101101011010000011101001101000110010101000001100011110111111001001100101010000011010011110011010110001000001110111110100011000011110100010000011010011110100010000011001001101111110010111100110101100010000011010001101111111011101000001100001110111001000001100001111010011101001100001110001111010111100101111001001000001101101110100111001111101000111010001000001110101111001111001010100000110100111101000101100010000011000011101110110010001000001110111110100011000011110100010000011110011101111111010101000001101101110100111001111101000111010001000001100100110111101000001110100110111101000001100100110010111000011101100010000011101111101001111010011010000100000111001111101011100011110100001000001100001110111001000001100001111010011101001100001110001111010110101110000101001010100101111000101000010101101001110111011101000100000110110111000011101001110111001010001101001110111011101000100000110000111100101100111110001101011000100000110001111010001100001111001001010100101010010000011000011110010110011111101100101001000101011110110001010000100111001101101111111001001000000101000011101101110110101001000101000010010001001111001111110011110011111010011001011101101010100011000011110010110011111101101011011011000010111010101001011101100010101111101

I played around with this and overthought it plenty. Eventually I arrived at the conclusion that this is a binary representation of an ASCII string because running the following in python 2 (with the above stored as a string called bstring)

print len(bstring) % 7

outputs 0.

So I completed the script to split the bit string into segments 7 digits long, interpret the integer in base 2, then get the ASCII character associated with that value and append it to a new string.

This is the script:

bstring = "0101111...111101"

newstr = ""

for li in xrange(0, len(bstring), 7):

byte = bstring[li:li+7]

value = int(byte, base=2)

char = chr(value)

newstr += char

print newstr

The result is this:

/\*

Code, compile, and execute the following code on a variety of operating systems (at the very least try a version of Linux and a version of Windows). Comment on your observations. Then comment on what you think the code is, what it does, how an attacker might use it, and what you might do to deal with such an attack.

\*/

int main(int argc, char\*\* argv)

{

for (;;)

system(argv[0]);

}

Looking at this I immediately recognize a fork bomb because I Googled fork bomb implementations in a variety of languages a few months ago (thanks Wikipedia). I wrote one in Windows 7 batch because I wanted to see how many instances of CMD it would take to crash the VDIs they give us at Southwest Airlines.

Basically the first argument passed into argv will always be the name of the executable, so calling system with that string will just spin up another instance of the same process.

for (;;) is the same as while (true).

Anyway:

Because I am a good boy I compiled and executed this on Ubuntu Linux (WSL) and Windows 10 Home edition (with no protection because I want a reason to install arch, unfortunately everything was fine) with GCC (MinGW on Windows).

Ubuntu Linux:

Because I used WSL for this, there was no GUI so it was pretty uneventful but still a learning experience.

The command I used (sol.py is the python file name)

python sol.py | gcc -xc - ; ./a.out

After pressing enter, I got a compiler warning because there are no #include’s for system(), but otherwise the command line went dead. Sending SIGINT did nothing. I gave it a minute to see how Windows would handle a fork bomb in WSL, but nothing happened so I closed the process and moved on. How boring!

Windows 10 Home Edition

I ran this directly on the hardware half expecting corruption to occur as a result of this irresponsible behavior (Unity Engine did it to me so why wouldn’t a fork bomb).

The command is pretty much the exact same (using the exact same file)

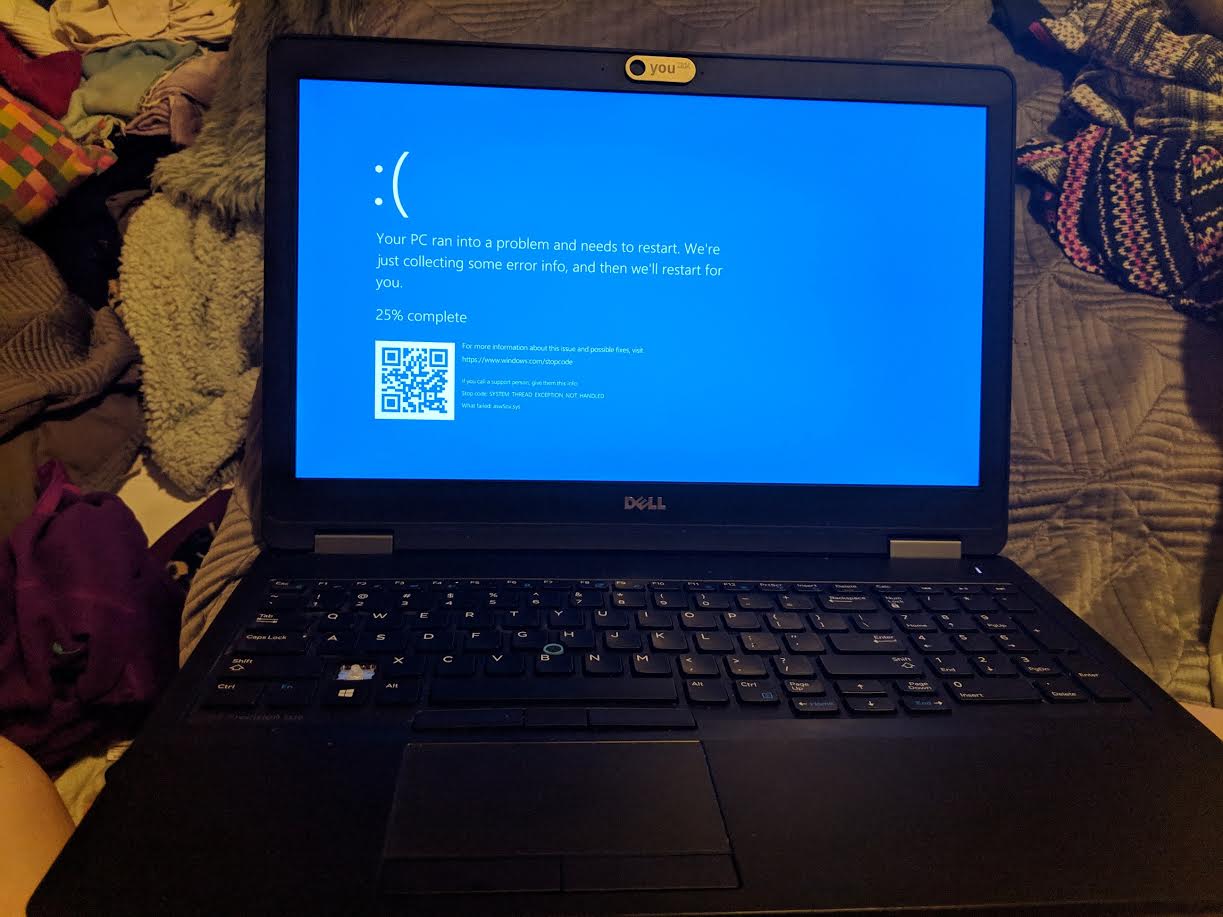
py2 sol.py | gcc -xc - && .\a.exe

py2 is an alias for my python 2 path because I use python3 by default.

For some reason MinGW gcc didn’t yell at me for not having the #include for system().

Anyway, this one was more fun. My computer got REALLY hot in like 10 seconds. My antivirus struggled to “scan” all of the processes being started, how could it compete? I’m not sure why I keep it around anyway. Then I got a 5 second freeze and a blue screen because Windows ran out of thread space :D.

Here is a picture to use in your future anti-microsoft rants.



Continuing to follow the instructions in the C comment:

The advantage of this kind of attack would be that the system effected loses all of its compute resources to trying to process the attack, so the user you are trying to troll can end up with an unresponsive computer until their system crashes. In terms of cyberstorm, this means they are probably not earning points. In terms of real world applications, if you’re trying to down somebody’s website (etc) in a non DDoS way this might work assuming you can get access to whatever box(es) it runs on. It may be worth tweaking the fork bomb such that you keep the system at processing capacity but have it self-limit so that it doesn’t crash the system.

Google says a good way to prevent a fork bomb (on Linux) is to set a process count limit on a user or group in /etc/security/limits.conf.

<https://www.cyberciti.biz/tips/linux-limiting-user-process.html>

They didn’t like my adblock.

My own idea to prevent this specific fork bomb implementation would be to scan new processes for duplicate names, because each process instance will have the same name. Say if 30 instances of ‘nonsuspiciousprocessignoreme’ spin up in like 50 ms then obviously I will probably want to stop that.