Joseph Guzman

12/22/2019

Leviathan01

This is the login info for this lab:

```
ssh server: leviathan.labs.overthewire.org
port: 2223
username: leviathan1
password: rioGegei8m
```

Once you login you'll notice a directory named `check`. It is a 32-bit lsb executable. I know this because I ran the `file` command on it. See *figure 1* for details.

Figure 1: file details

When you execute the file it prompts you for a password. We can assume that it gives us the password of the Leviathan2 if we guess the right password. I tried two things which turned out to be overkill and not really work.

The first thing I did was use retdec to decompile the lsb assembly code. I've included the result code in the github repo as leviathan1.c. This was a logical approach. I did notice in the C code that the user input was being compared to this int: `int32_t v1 = 0x786573; // bp-24` via the strcmp function. When I converted this to ASCII I got the string "xes". I attempted to use "xes" as the password and later found out that the password was "sex". This could have been a bug in retdec or just an inconsistency. It doesn't make sense anyway to use strcmp on between an int and a string.

The second thing I tried was using gdb to edit and read the assembly live. I tracked down exactly where the password check was taking place. It was check with jne instruction. So, I changed the eax register directly, before the check was taking place (see *Figure 2*). This did allow me to bypass the password entry, and get the sh shell prompt. However, for some reason I only got the shell as user leviathan1 instead of leviathan2 as shown in *Figure 3*. This stackoverflow link claims that this is because you cannot debug a setuid binary properly using gdb, unless you use sudo. I did not have access to sudo on the server. A setuid binary is a binary that when executed will execute with the permissions of the owner of the file.

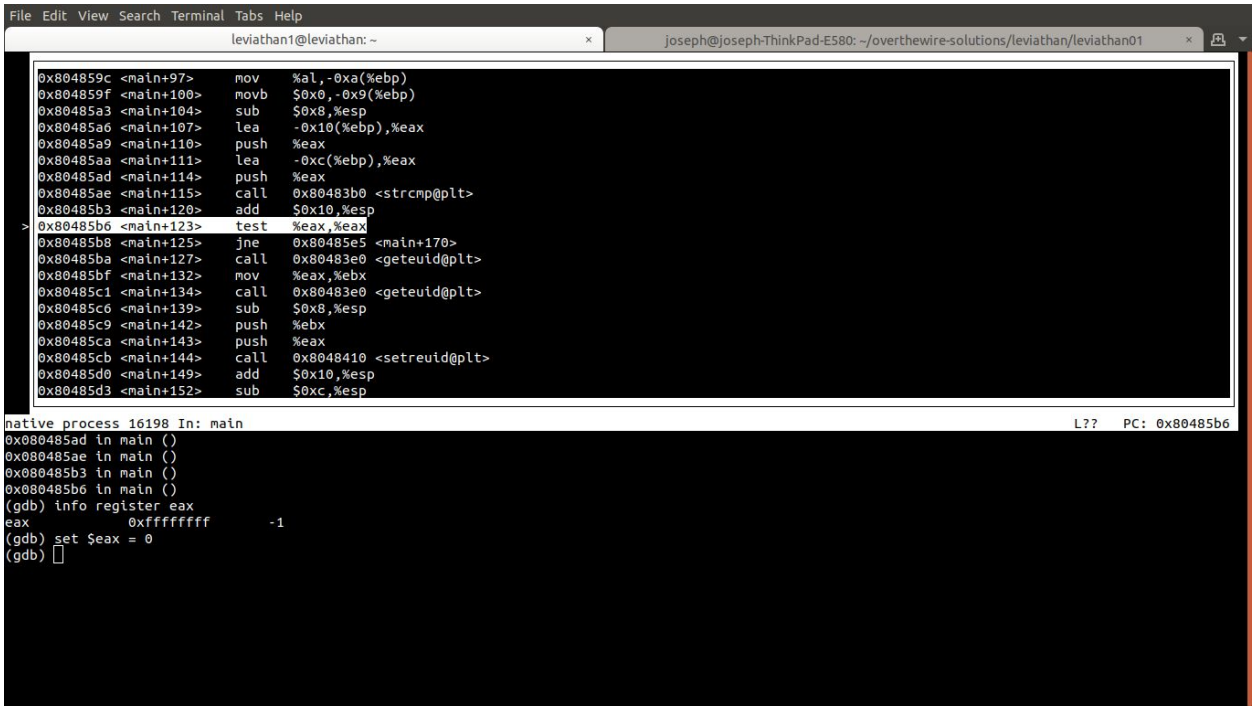

Figure 2: Manually Bypass Password Check

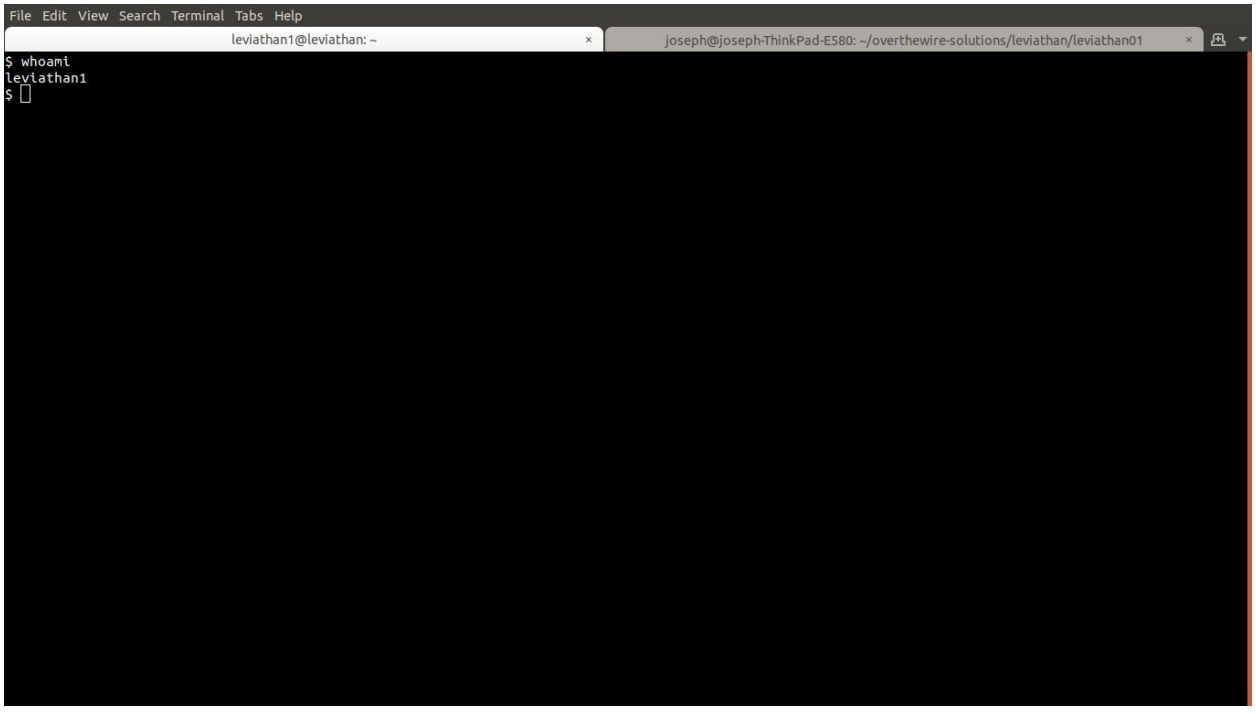

Figure 3: Continue Program, Enter Shell

After making these attempts I googled a solution online. This link helped me. Basically all that was required was checking the password once with ltrace which shows you the strcmp comparison. We can clearly see that our password attempt is being compared to the string “sex” (see *Figure 4*). So we know that this was the password. It’s insecure that the password is stored raw within the program. It would be smarter to hash the password as is standard in web applications.

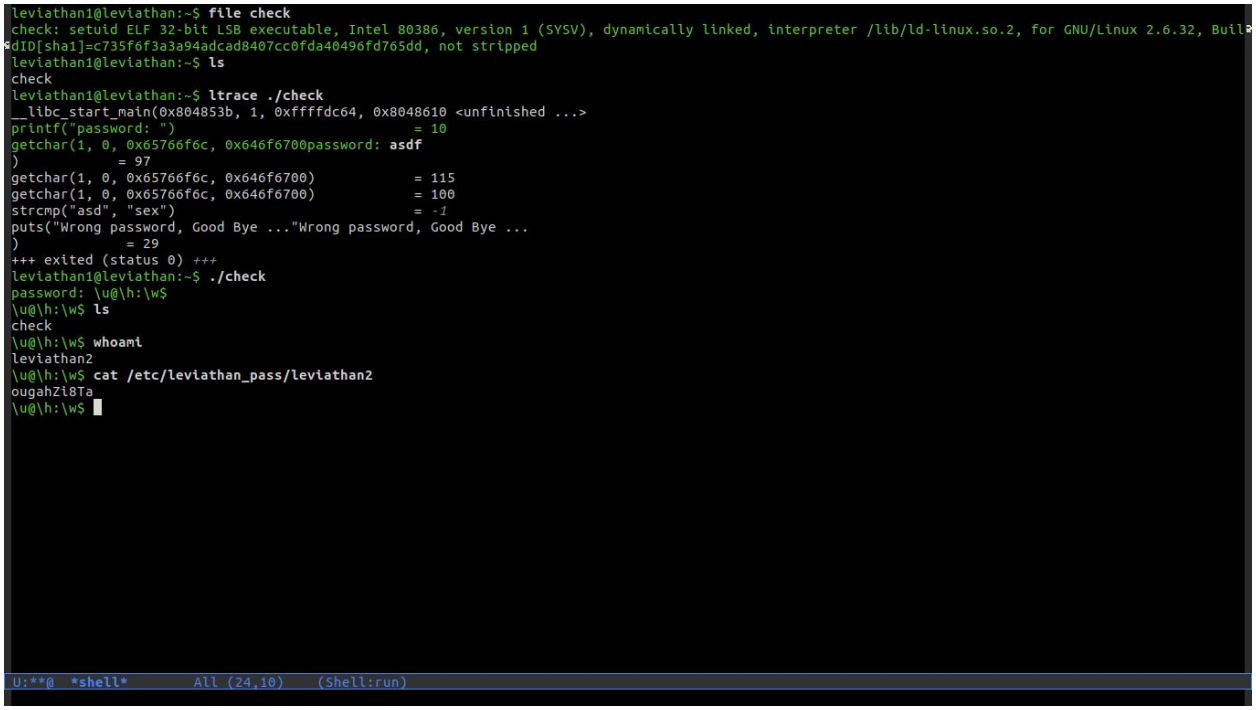

Figure 4: Ltrace Solution

This is the login info for the next lab:

```
ssh server: leviathan.labs.overthewire.org
port: 2223
username: leviathan2
password: ougahZi8Ta
```