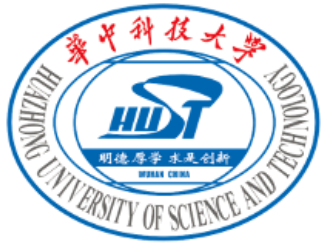

华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

网络空间安全学院

FORTIFY_SOURCE

网络空间安全学院 慕冬亮

Email : dzm91@hust.edu.cn

2. FORTIFY_SOURCE

FORTIFY_SOURCE 为一些不安全的操作函数（如 strcpy, sprintf）提供安全变种来工作,同时编译需要开启优化O1以上

```
gcc -D_FORTIFY_SOURCE=1 -o test test.c -O1        // 较弱的检查
gcc -D_FORTIFY_SOURCE=2 -o test test.c -O1        // 较强的检查
```

级别	说明
0	关闭FORTIFY_SOURCE
1	仅仅只会在编译时进行检查 (特别像某些头文件 <string.h>)
2	程序运行时也会有检查 (如果检查到缓冲区溢出，就终止程序)

FORTIFY_SOURCE

这是在glibc 2.3.4中引入的功能，通过检测某些C库函数中的缓冲区溢出来提高程序的安全性。当设置了_FORTIFY_SOURCE，像strcpy这样的函数会被替换成加强版的函数，这些函数使用__builtin_object_size函数来确定缓冲区的大小，并包含对缓冲区溢出的检查。

```
0000000000001189 <main>:
    1189:       f3 0f 1e fa             endbr64
    118d:       55                      push   %rbp
    118e:       53                      push   %rbx
    118f:       48 83 ec 18             sub    $0x18,%rsp
    1193:       bb 28 00 00 00          mov    $0x28,%ebx
    1198:       64 48 8b 03             mov    %fs:(%rbx),%rax
    119c:       48 89 44 24 08          mov    %rax,0x8(%rsp)
    11a1:       31 c0                   xor    %eax,%eax
    11a3:       48 8b 76 08             mov    0x8(%rsi),%rsi
    11a7:       48 8d 6c 24 03          lea    0x3(%rsp),%rbp
    11ac:       ba 05 00 00 00          mov    $0x5,%edx
    11b1:       48 89 ef                mov    %rbp,%rdi
    11b4:       e8 d7 fe ff ff          callq  1090 <__strcpy_chk@plt>
    11b9:       48 89 ef                mov    %rbp,%rdi
    11bc:       e8 af fe ff ff          callq  1070 <puts@plt>
    11c1:       48 8b 44 24 08          mov    0x8(%rsp),%rax
    11c6:       64 48 33 03             xor    %fs:(%rbx),%rax
    11ca:       75 0c                   jne    11d8 <main+0x4f>
    11cc:       b8 00 00 00 00          mov    $0x0,%eax
    11d1:       48 83 c4 18             add    $0x18,%rsp
    11d5:       5b                      pop    %rbx
    11d6:       5d                      pop    %rbp
    11d7:       c3                      retq
    11d8:       e8 a3 fe ff ff          callq  1080 <__stack_chk_fail@plt>
    11dd:       0f 1f 00                nopl   (%rax)
```

```
0000000000001189 <main>:
    1189:       f3 0f 1e fa             endbr64
    118d:       55                      push   %rbp
    118e:       48 89 e5                mov    %rsp,%rbp
    1191:       48 83 ec 20             sub    $0x20,%rsp
    1195:       89 7d ec                mov    %edi,-0x14(%rbp)
    1198:       48 89 75 e0             mov    %rsi,-0x20(%rbp)
    119c:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
    11a3:       00 00
    11a5:       48 89 45 f8             mov    %rax,-0x8(%rbp)
    11a9:       31 c0                   xor    %eax,%eax
    11ab:       48 8b 45 e0             mov    -0x20(%rbp),%rax
    11af:       48 83 c0 08             add    $0x8,%rax
    11b3:       48 8b 10                mov    (%rax),%rdx
    11b6:       48 8d 45 f3             lea    -0xd(%rbp),%rax
    11ba:       48 89 d6                mov    %rdx,%rsi
    11bd:       48 89 c7                mov    %rax,%rdi
    11c0:       e8 ab fe ff ff          callq  1070 <strcpy@plt>
    11c5:       48 8d 45 f3             lea    -0xd(%rbp),%rax
    11c9:       48 89 c7                mov    %rax,%rdi
    11cc:       e8 af fe ff ff          callq  1080 <puts@plt>
    11d1:       b8 00 00 00 00          mov    $0x0,%eax
    11d6:       48 8b 4d f8             mov    -0x8(%rbp),%rcx
    11da:       64 48 33 0c 25 28 00    xor    %fs:0x28,%rcx
    11e1:       00 00
    11e3:       74 05                   je     11ea <main+0x61>
    11e5:       e8 a6 fe ff ff          callq  1090 <__stack_chk_fail@plt>
    11ea:       c9                      leaveq
    11eb:       c3                      retq
    11ec:       0f 1f 40 00             nopl   0x0(%rax)
```

FORTIFY_SOURCE

__strcpy_chk : __dest, __src, __bos (__dest)

__builtin___strcpy_chk : __dest, __src, __bos (__dest)

- __strcpy_chk 和 __builtin___strcpy_chk 是强化版的字符串复制函数，它们被设计用于在编译时防止缓冲区溢出错误。这些函数的前两个参数继承自 strcpy，分别是目的字符串和源字符串。第三个参数是目的字符串的长度，这个长度是在编译时确定的。
- __builtin___strcpy_chk 是 GCC 内建的检查函数，它在运行时会检查源字符串是否会溢出目的缓冲区。如果溢出，程序将会调用 __chk_fail 并终止执行，从而防止潜在的溢出攻击。
- 在 __strcpy_chk 的实现中，如果指定的大小是 -1，GCC 会优化调用，直接调用 strcpy。如果源字符串的长度超出了目的缓冲区的大小，那么会调用 __chk_fail 来触发错误。

FORTIFY_SOURCE

当编译时定义 _FORTIFY_SOURCE=1 宏时，GCC 会发出警告：

```
#include <string.h>
int main()
{
    char string[5];
    strcpy(string, “hello world");
    return 0;
}
```

```
root@f953676d993d:/mnt/software-security-dojo/integer-overflow/level-1-1# gcc fority_test.c -O1  -o fority_2 -D_FORTIFY_SOURCE=1
<command-line>: warning: "_FORTIFY_SOURCE" redefined
<built-in>: note: this is the location of the previous definition
In file included from /usr/include/string.h:495,
                 from fority_test.c:1:
In function ‘strcpy’,
    inlined from ‘main’ at fority_test.c:5:3:
/usr/include/x86_64-linux-gnu/bits/string_fortified.h:90:10: warning: ‘__builtin___strcpy_chk’ writing 12 bytes into a region of size 5 overflows the destination
-Wstringop-overflow=]
   90 |   return __builtin___strcpy_chk (__dest, __src, __bos (__dest));
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

FORTIFY_SOURCE

当代码发生改变时，动态情况下也可以检测

```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[])
{
    char string[5];
    strcpy(string, argv[1]);
    printf("%s\n", string);
    return 0;
}
```

```
root@f953676d993d:/mnt/software-security-dojo/integer-overflow/level-1-1# gcc fority_test2.c -O1  -o fority_2 -D_FORTIFY_SOURCE=2
root@f953676d993d:/mnt/software-security-dojo/integer-overflow/level-1-1# ./fority_2 11111111111111111111
*** buffer overflow detected ***: terminated
Aborted (core dumped)
root@f953676d993d:/mnt/software-security-dojo/integer-overflow/level-1-1#
```