

Roll No: 2103163

Batch: C32

Name: Om Shete

## Experiment No 10

**Aim:** Simulate Buffer overflow attack using Splint

**Description:**

Buffer overflow is a mistake that exist in some C implementations. These classes of bugs are dangerous as they write past the end of a buffer or array and hence corrupt the process stack. They often change the return address of a process after a function call to a secret memory location where a malicious code is planted.

There are main two types

- Stack based attacks
- Heap based attacks

Heap-based attacks flood the memory space reserved for a program, but the difficulty involved with performing such an attack makes them rare. Stack-based buffer overflows are by far the most common.

**Splint** is a tool for statically checking C programs for security vulnerabilities and programming mistakes. Splint does many of the traditional lint checks including unused declarations, type inconsistencies, use before definition, unreachable code, ignored return values, execution paths with no return, likely infinite loops, and fall through cases. More powerful checks are made possible by additional information given in source code annotations. Annotations are stylized comments that document assumptions about functions, variables, parameters and types. In addition to the checks specifically enabled by annotations, many of the traditional lint checks are improved by exploiting this additional information. Splint is designed to be flexible and allow programmers to select appropriate points on the effort benefit curve for particular projects. As different checks are turned on and more information is given in code annotations the number of bugs that can be detected increases dramatically.

Problems detected by Splint include:

- Dereferencing a possibly null pointer
- Using possibly undefined storage or returning storage that is not properly defined
- Type mismatches, with greater precision and flexibility than provided by C compilers
- Violations of information hiding
- Memory management errors including uses of dangling references and memory leaks

Roll No: 2103163

Batch: C32

Name: Om Shete

- Dangerous aliasing
- Modifications and global variable uses that are inconsistent with specified interfaces
- Problematic control flow such as likely infinite loops, fall through cases or incomplete switches and suspicious statements
- Buffer overflow vulnerabilities
- Dangerous macro implementations or invocations
- Violations of customized naming conventions

### **Steps :**

#### **1. Installation**

```
$ sudo apt-get install splint
```

#### **2. Checking Vulnerability**

```
$ splint program1.c
```

### **Implementation:**

#### **Program 1:**

```
#include <stdio.h>
```

```
#include <string.h>
```

```
void vulnerableFunction(char *str) {
```

```
    char buffer[10];
```

Roll No: 2103163

Batch: C32

Name: Om Shete

```
strcpy(buffer, str); // Copies the input str to buffer without checking its length
```

```
}
```

```
int main() {
```

```
    char largeString[] = "This string is way too long for the buffer";
```

```
    vulnerableFunction(largeString);
```

```
    printf("Returned normally from vulnerableFunction\n");
```

```
    return 0;
```

```
}
```

## Output:

```
root@admini-OptiPlex-3050:/home/admini# splint a.c +bounds
Splint 3.1.2 --- 21 Feb 2021

a.c: (in function vulnerableFunction)
a.c:6:5: Possible out-of-bounds store: strcpy(buffer, str)
  Unable to resolve constraint:
    requires maxRead(str @ a.c:6:20) <= 9
    needed to satisfy precondition:
    requires maxSet(buffer @ a.c:6:12) >= maxRead(str @ a.c:6:20)
    derived from strcpy precondition: requires maxSet(<parameter 1>) >=
    maxRead(<parameter 2>)
  A memory write may write to an address beyond the allocated buffer. (Use
  -boundswrite to inhibit warning)

Finished checking --- 1 code warning
```

## Program 2:

```
#include <stdio.h>
```

```
#include <string.h>
```

```
char password[] = "password";
```

```
int get_password()
```

Roll No: 2103163

Batch: C32

Name: Om Shete

```
{  
  
    int auth_ok = 0;  
  
    char buff[16];  
  
    printf("Enter password: ");  
  
    scanf("%s", buff);  
  
    if (strcmp(buff, password) == 0)  
    {  
        auth_ok = 1;  
        return auth_ok;  
    }  
  
void success()  
  
{  
  
    printf("Success!\n");  
  
}  
  
int main(int argc, char **argv)  
  
{  
  
    int res = get_password();  
  
    if (res == 0)  
    {  
        printf("Failure \n");  
        return 0;  
    }  
}
```

Roll No: 2103163

Batch: C32

Name: Om Shete

```
        success();

        return 0;

}
```

## Output:

```
root@admini-OptiPlex-3050:/home/admini# splint a.c
Splint 3.1.2 --- 21 Feb 2021

a.c: (in function get_password)
a.c:8:2: Return value (type int) ignored: scanf("%s", buff)
      Result returned by function call is not used. If this is intended, can cast
      result to (void) to eliminate message. (Use -retvalint to inhibit warning)
a.c: (in function main)
a.c:16:14: Parameter argc not used
      A function parameter is not used in the body of the function. If the argument
      is needed for type compatibility or future plans, use /*@unused@*/ in the
      argument declaration. (Use -paramuse to inhibit warning)
a.c:16:27: Parameter argv not used
a.c:3:6: Variable exported but not used outside a: password
      A declaration is exported, but not used outside this module. Declaration can
      use static qualifier. (Use -exportlocal to inhibit warning)
a.c:4:5: Function exported but not used outside a: get_password
      a.c:11:18: Definition of get_password
a.c:12:6: Function exported but not used outside a: success
      a.c:15:1: Definition of success

Finished checking --- 6 code warnings
```

## Program 3:

```
#include<stdio.h>

main()

{

char buff[5];

printf("My stack looks

like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n");

buff[5]='abcdefghijklmnopshgkgfks';

printf("%c\n",buff[5]);

printf("My new stack looks
```

Roll No: 2103163

Batch: C32

Name: Om Shete

```
like:\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n");
```

```
}
```

### Output:

```
Cannot continue.  
abhi@MegaBlaziken:/mnt/c/Users/User/Downloads$ splint file.c  
Splint 3.1.2 --- 21 Feb 2021  
  
file.c:1:9: Parse Error: Non-function declaration: include :  
        int. (For help on parse errors, see splint -help parseerrors.)  
*** Cannot continue.
```