**Carl Bunt**
Email: cbunt@psu.edu
Course: CS 491 - Intro to Security
Instructor: Dr. Karen L. Karavanic

**Homework - 3**
ID: 915910838
Term: Fall 2020
Due Date: 09 NOV 20

**Question A:**
Written Exercises

1. Define: Race condition
2. Define: Atomic operation
3. Define: Impact surface
4. Dectribe the attack surface for the Linux Lab machies in CS.

*Response:*

1. **Race condition: When 2 threads are attempting to access data at the same time and there is no way to guarantee which will get to it first.**
2. **Atomic operation: An operation that is compleated in a single process cycle.**
3. **Impact surface: The collection all events that happen as a result of a successful exploit.**
4. **Linux Lab attack surface: The three most obvious attack vectors I can think of are:**

| | |
|---|---|
| **Physical security** | **A nefarious actor may obtian physical access to the system to steal physical system to access private data.** |
| **Network access** | **A nefarious actor may either on local network or on across the internet send commands to invoke maliscous code.** |
| **User ignorance** | **A User may fall victim of a phishing scheme or introduce an infected file to the system.** |

**Question B:**
Hands On: Linux/C/C++

1. Submit an 8-by-8 table of the conversion effects of in C or C++ of data types.
2. Provide one example of exploitable code for each of integer overflow and integer underflow.
3. write a C/C++ program with a buffer overflow vulnerability.

*Response:*

1. C/C++ conversions

| | s-char | u-char | s-short | u-short | s-int | u-int | s-long | u-long |
|---|---|---|---|---|---|---|---|---|
| s-char | SWID BP VP | SWID BP VC | ZEXT BC VP | ZEXT BC VC | ZEXT BC VP | ZEXT BC VC | ZEXT BC VP | ZEXT BC VC |
| u-char | SWID BP VC | SWID BP VP | SEXT BC VP | SEXT BC VP | SEXT BC VP | SEXT BC VP | SEXT BC VP | SEXT BC VP |
| s-short | TRNC BP VC | TRNC BP VC | SWID BP VP | SWID BP VC | ZEXT BC VP | ZEXT BC VC | ZEXT BC VP | ZEXT BC VC |
| u-short | TRNC BP VC | TRNC BP VC | SWID BP VC | SWID BP VP | SEXT BC VP | SEXT BC VP | SEXT BC VP | SEXT BC VP |
| s-int | TRNC BP VC | TRNC BP VC | TRNC BP VC | TRNC BP VC | SWID BP VP | SWID BP VC | ZEXT BC VP | ZEXT BC VC |
| u-int | TRNC BP VC | TRNC BP VC | TRNC BP VC | TRNC BP VC | SWID BP VC | SWID BP VP | SEXT BC VP | SEXT BC VP |
| s-long | TRNC BP VC | TRNC BP VC | TRNC BP VC | TRNC BP VC | TRNC BP VC | TRNC BP VC | SWID BP VP | SWID BP VC |
| u-long | TRNC BP VC | TRNC BP VC | TRNC BP VC | TRNC BP VC | TRNC BP VC | TRNC BP VC | SWID BP VC | SWID BP VP |

Table 1: Data Type Conversions

**Key:**
sign-extended(SEXT)
zero-extended(ZEXT)
same width(SWID)
truncated(TRNC)
value changed(VC)
value preserved(VP)
bit-pattern changed(BC)
bit-pattern preserved(BP)

2. Integer OverFlow

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

uint16_t SHIRT_VAL = 20;

int main(int argc, char *argv[]) {
  if(argc < 2) {
    printf("usage: OverFlow <Shirt count>\n");
    exit(EXIT_FAILURE);
  }
  uint16_t count = atoi(argv[1]);
  uint16_t Val = SHIRT_VAL * count;


  printf("Total: %d\n", Val);


  return 0;
}
```

Say you buy a bulk order of shirts for $20 each and the total is stored in a 16-bit uint. All is fine unless you order more then 3277. The overflow will reslut in a order total of $4.

Integer UnderFlow

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

int16_t SHIRT_VAL = 20;

int main(int argc, char *argv[]) {
  if(argc < 2) {
    printf("usage: UnderFlow <Shirt count>\n");
    exit(EXIT_FAILURE);
  }
  int16_t count = atoi(argv[1]);
  int16_t Val = SHIRT_VAL * (count * (-1));


  printf("Total: %d\n", Val);


  return 0;
}
```

Now you are returning the exess shirts. This time it is stored in a 16-bit int. Account debt will show in a negative value until 1639 shirts are returned then suddenly you will owe them $32756 to return those shirts.

3. Buffer overFlow

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

uint16_t SHIRT_VAL = 20;

int main(int argc, char *argv[]) {
  if(argc < 3) {
    printf("usage: BufferOverFlow <Item count> <Order name>\n");
    exit(EXIT_FAILURE);
  }
  char Customer[8];
  uint16_t count = atoi(argv[1]);
  uint16_t Val = SHIRT_VAL * count;


  strcpy(Customer, (argv[2]));


  printf("Total: %d\n", Val);


  return 0;
}
```

If running the code with char string of 8 characters then the total will be $0. Anything longer then that the total is over writen with the ASCII input.

| | |
|---|---|
| ./BufferOverFlow 10 Carl | $200 |
| ./BufferOverFlow 10 Carl0000 | $0 |
| ./BufferOverFlow 10 Carl00000 | $48 |
| ./BufferOverFlow 10 Carl000000 | $12336 |