Secure Programming and Scripting

OVERFLOWS

Luke Fox -
20076173

BUFFER OVERFLOW

Program Compilation: > gcc -o buffer buffer.c -w -fno-stack-protector

Usage: The program will first generate a random 5-character password for the user. The user is then asked to log-in to this account using this unique password. If the entered password does not match the generated password, the user will be denied access to the account and the program ends. However, entering a password more than 15 characters will bypass the password and grant access to the admin account. This is because a 15+ characters exceeds the buffer size, leading to other data types being affected – in this case, and int changing value from 0 to 1.

Example:

> gcc -o buffer buffer.c -w -fno-stack-protector

> ./buffer

Welcome to Bank Of Ireland! We will now generate your 5 digit password!

Your generated password is IENTU

Please enter your password:

testpasswordmorethan15characters

WRONG

Welcome Back User

Prevention: Buffer overflow attacks can best be avoided by using fgets(). This function, unlike gets(), will stop when either n-1 characters are read (n being a char array size). This prevents users entering an amount exceeding size n thus preventing the overflow.

INTEGER OVERFLOW

Program Compilation: > gcc -o integer integer.c -w -fno-stack-protector

Usage: This program simply takes the input from the user, where the input is not 0 or 1, and wraps it in a while loop multiplying it by 2 infinite times – printing each iteration. Eventually, the outputted number will exceed the max possible int value of 2,147,483,647, and the next value on will print −2,147,483,648. Triggering this overflow could severely impact stability and usability of programs.

Example:

> gcc -o integer integer.c -w -fno-stack-protector

> ./integer

Please enter a number:

2

4

6

…

1073741824

−2,147,483,648

Prevention: A simple way of avoiding this is to avoid wrapping values in a loop that performs a mathematical function infinite times. Another simple way to avoid integer overflows is to create a form of validation, restricting values from going over the max int value.

FORMAT STRING OVERFLOW

Program Compilation: > gcc -o format format.c -w -fno-stack-protector

Usage: This program presents users with options to activate a product, either as a free user or premium user. Both selections will require the user to enter an email. In the context of this program, a regular user will not have access to the premium key stored in the program. But when selecting to proceed as a regular user and entering an email, the user may exploit the program into possibly dumping hexadecimal addresses for the stack- using %x. This potentially gives users an address to chunks of memory inside the program which can be used to a malicious extent. Users can also initiate a denial of service attack by crashing the program – using %s.

The additional emails stored at the top of the program exist as potential data to be dumped, demonstrating the risk of leaking sensitive information in format string overflows.

Example:

> gcc -o format format.c -w -fno-stack-protector

> ./format

1. Welcome to Photoshop, please enter which account you'd like:

1). Basic Account

2). Premium Account

1

Selected Basic Account

Please enter your email: %x%x%x%x%x%x

1823057700823038c000 // hexadecimal representation of data from the stack

2. Welcome to Photoshop, please enter which account you'd like:

1). Basic Account

2). Premium Account

1

Selected Basic Account

Please enter your email: %s%s%s%s

Segmentation Fault // program crashes

Prevention: Some steps to be taken to prevent this overflow include always specifying a format string as part of the program and not relying on user input to specify it.