Stack Buffer Overflow

Integrantes:

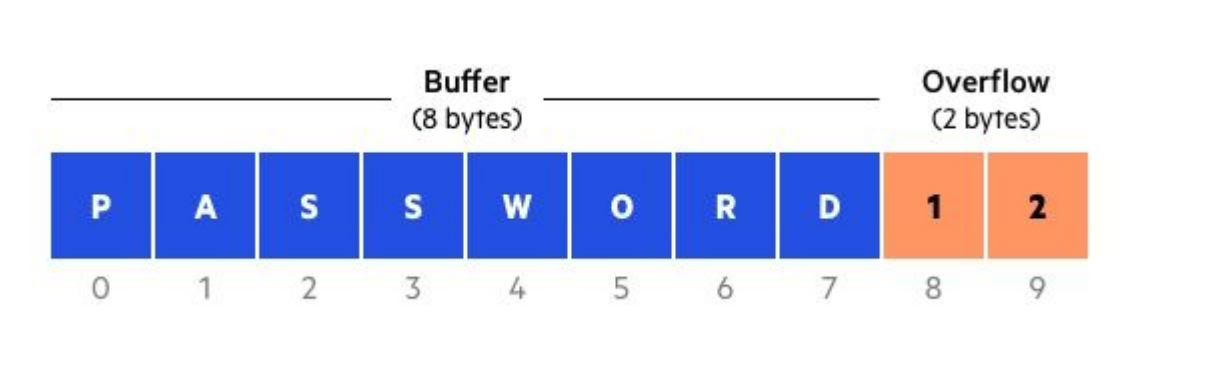
Mauritania Pineda Salgado
Sandra Paola Bautista García
Carlos Emilio Zavala Lopez
Farid Alejandro Mejía Melchor

Orlando Ochoa Magallan
Yamir Alejandro García Padilla
Elí Vladimir Loza Prado

Definición /tipos /Cómo funciona

Buffer overflow es una de las vulnerabilidades persistentes a pesar de la evolución y complejidad de los mecanismos de seguridad. Se encuentra presente en diversas aplicaciones por lo que aparece constantemente en las listas de vulnerabilidades críticas publicadas por instituciones enfocadas a la notificación de nuevas amenazas de seguridad.

El Buffer overflow es una vulnerabilidad causada por la inserción de datos con tamaño superior al esperado por una aplicación, lo que provoca la sobrescritura de espacios adyacentes en la memoria.



Definición /tipos /Cómo funciona

ESTRUCTURA DE LA MEMORIA

El **stack** almacena los argumentos de las funciones, las variables locales y las direcciones de retorno de las llamadas a funciones.

El **heap** se encarga de gestionar la memoria dinámica, es decir la memoria solicitada durante el tiempo de ejecución.

PRINCIPALES TIPOS DE BUFFER OVERFLOW

Existen dos tipos principales de buffer overflow, cuyo nombre se deriva del espacio en memoria sobre el cual es localizada la vulnerabilidad:

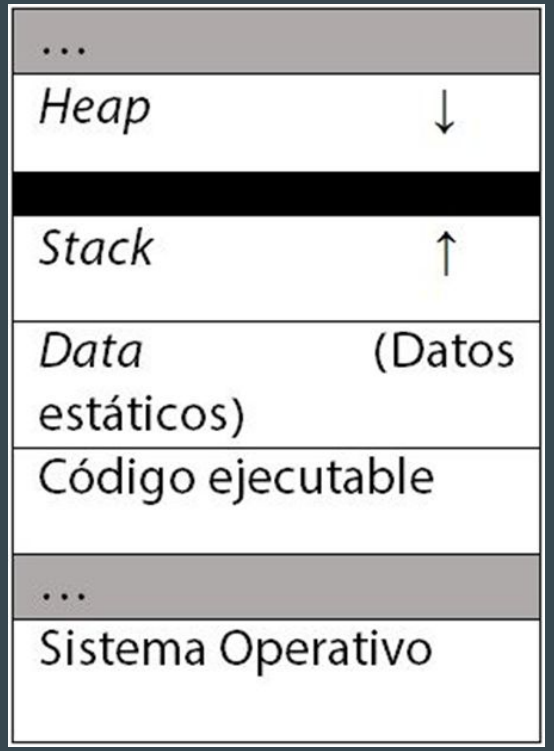
•**Stack overflow** o desbordamientos basados en pilas, son los más comunes y aprovechan la memoria de pila que sólo existe durante el tiempo de ejecución de una función.

•**Heap overflow** o ataques basados en montón, son más difíciles de llevar a cabo ya que implican inundar el espacio de memoria asignado para un programa más allá de la memoria utilizada para las operaciones de tiempo de ejecución actuales.

Definición /tipos /Cómo funciona

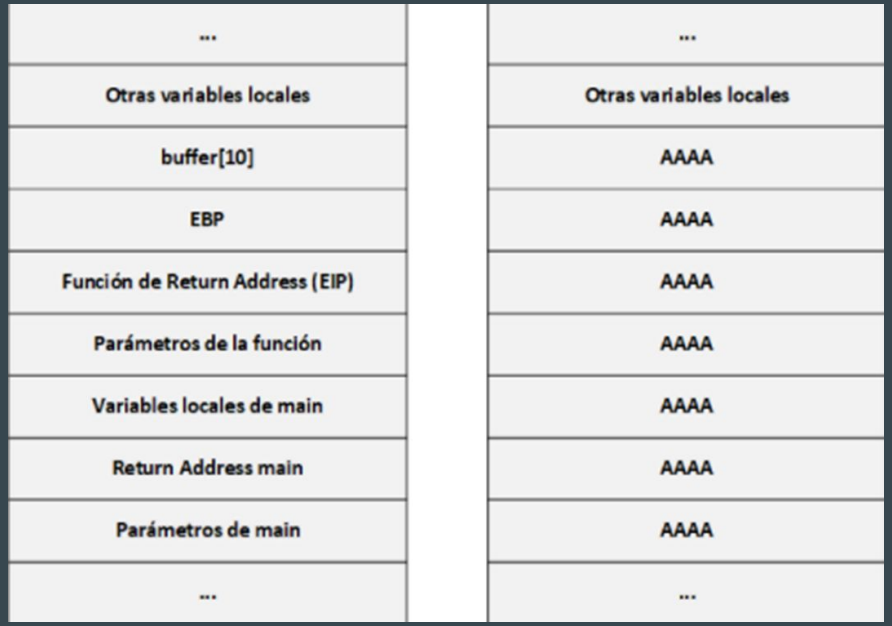
Existen regiones reservadas para datos de entrada del programa, pueden ser localizadas tanto en el stack como en el heap dependiendo del tipo de datos que van a almacenar. Estas regiones son llamadas buffer, por lo que se puede definir el buffer como un espacio en memoria que sirve como almacenamiento temporal de datos de entrada en un programa.

Para el buen funcionamiento de los programas, las instrucciones y los datos en ejecución se almacenan temporalmente en la memoria. Los datos ubicados después del búfer contienen una dirección de retorno que le permite al programa continuar su tiempo de ejecución.



Definición /tipos /Cómo funciona

El problema de que sobrescriba los datos en memoria es que el atacante puede asegurarse de que lo que se sobrescriba sean direcciones de memorias que si existan, y por lo tanto se pueden ejecutar otras aplicaciones como lo puede ser abrir una terminal y ejecutar comandos permitiendo que el atacante pueda tomar el control del sistema



Esquemas de Protección

Apilar canarios: Los canarios de pila, se utilizan para detectar un desbordamiento de búfer de pila antes de que pueda ocurrir la ejecución de código malicioso.

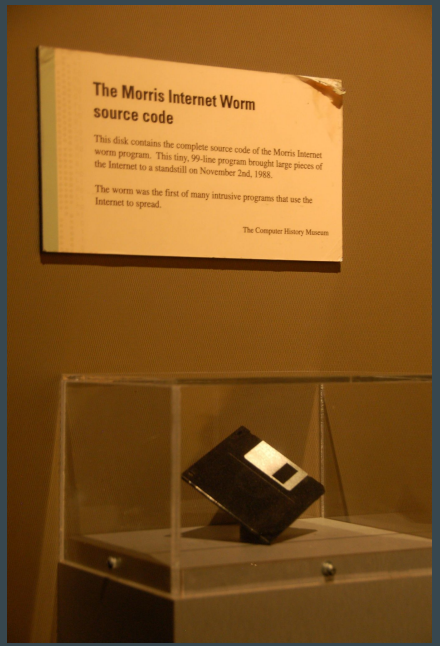
Pila no ejecutable: Aplicar una política de memoria en la región de memoria de la pila que no permita la ejecución desde la pila.

Aleatorización: Introducir la aleatorización en el espacio de memoria del programa en ejecución. Dado que el atacante necesita determinar dónde reside el código ejecutable que se puede usar para poder atacar.

Ataques que se han registrado

1988

El gusano Morris



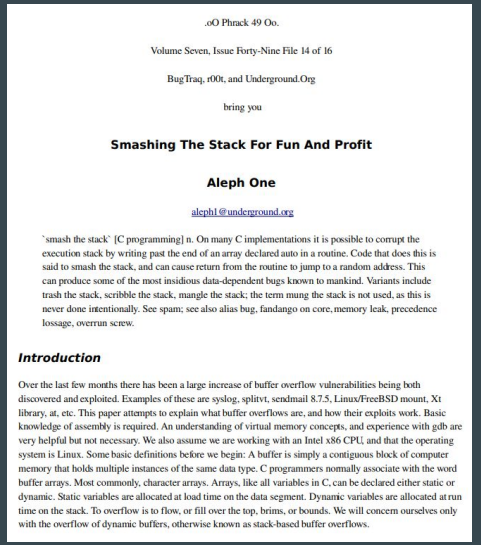
1995

Thomas Lopatic

1996

Elias Levy

"Smashing the Stack for Fun and Profit



.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine File 14 of 16

BugTraq, r00t, and Underground.Org

bring you

Smashing The Stack For Fun And Profit

Aleph One

aleph1@underground.org

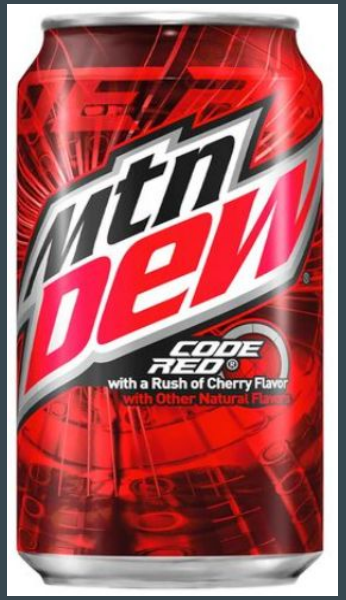
`smash the stack` [C programming] n. On many C implementations it is possible to corrupt the execution stack by writing past the end of an array declared auto in a routine. Code that does this is said to smash the stack, and can cause return from the routine to jump to a random address. This can produce some of the most insidious data-dependent bugs known to mankind. Variants include trash the stack, scribble the stack, mangle the stack; the term mung the stack is not used, as this is never done intentionally. See spam; see also alias bug, fandango on core, memory leak, precedence lossage, overrun screw.

Introduction

Over the last few months there has been a large increase of buffer overflow vulnerabilities being both discovered and exploited. Examples of these are syslog, splitvt, sendmail 8.7.5, Linux/FreeBSD mount, Xt library, at, etc. This paper attempts to explain what buffer overflows are, and how their exploits work. Basic knowledge of assembly is required. An understanding of virtual memory concepts, and experience with gdb are very helpful but not necessary. We also assume we are working with an Intel x86 CPU, and that the operating system is Linux. Some basic definitions before we begin: A buffer is simply a contiguous block of computer memory that holds multiple instances of the same data type. C programmers normally associate with the word buffer arrays. Most commonly, character arrays. Arrays, like all variables in C, can be declared either static or dynamic. Static variables are allocated at load time on the data segment. Dynamic variables are allocated at run time on the stack. To overflow is to flow, or fill over the top, brims, or bounds. We will concern ourselves only with the overflow of dynamic buffers, otherwise known as stack-based buffer overflows.

2001
Code Red Worm

Microsoft
IIS




```
HELLO! Welcome to http://www.worm.com! Hacked By Chinese! :)
```

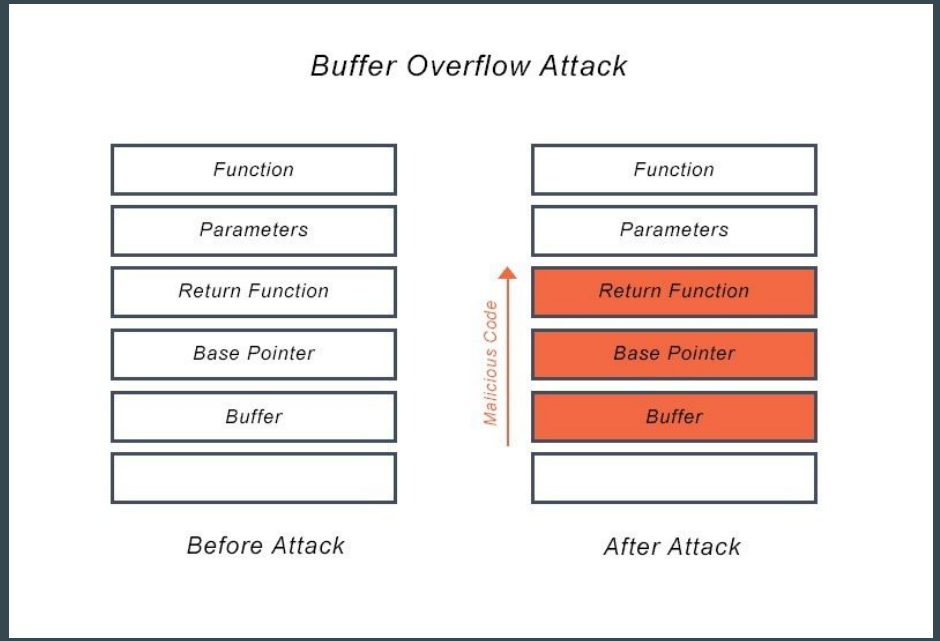
2003
SQL Slammer

Microsoft
SQL Server 2000

Ejemplo de cómo realizarlo

- ➔ El atacante envía información cuidadosamente elaborada a un programa.
- ➔ El programa intenta almacenar la entrada en un búfer que no es lo suficientemente grande para la entrada.
- ➔ El atacante establece nuevos valores para apuntar a una dirección de su elección.
- ➔ El programa se bloquea parcialmente porque la pila se desbordó .
- ➔ El programa intenta recuperarse yendo a la dirección de retorno, pero la dirección de retorno ha sido cambiada para apuntar al comando que especificó el atacante.

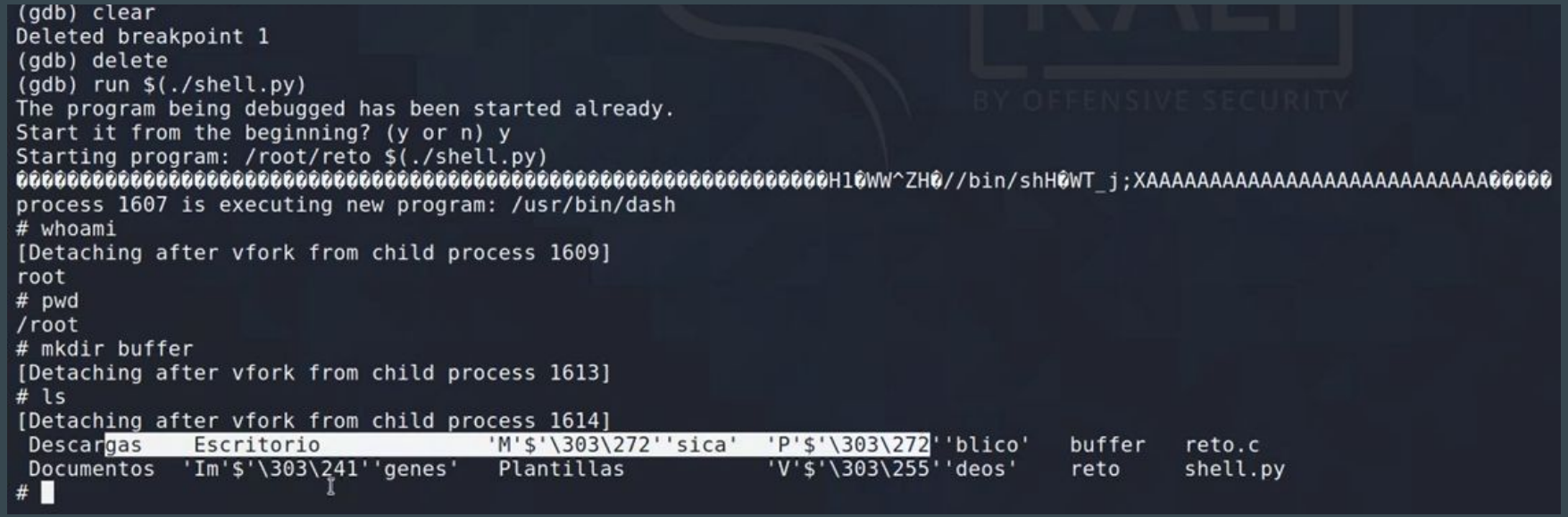
Una manera de introducir código malicioso es mediante una función shellcode, que prácticamente es código que se inyecta en la memoria de un programa vulnerable bajo la forma de un string de bytes.



¿Cómo se vería en práctica?

1. Se prepara la función maliciosa (shellcode), ya sea en lenguaje python o en algún otro para así introducirlo o concatenarlo en la memoria en código ensamblador

2. Se ejecuta la función por medio de la terminal de comandos, una vez hecha la ruta a la máquina o programa a atacar



Cómo defenderse

- Seguir estándares de desarrollo de código seguro.
- Utilizar lenguajes de programación que, además de ser eficientes en cuanto al uso de memoria, sean seguros.
- Evitar que ocurran condiciones de desbordamiento del buffer en el código. Por ejemplo, cuando se espera un máximo de 8 bytes como datos de entrada, la cantidad de datos que se puede escribir en el buffer se limitará a 8 bytes en cualquier momento.
- Que las aplicaciones contengan , por ejemplo. ejecutables de tipo position-independent.
- Otra forma de proteger los desbordamientos del buffer es detectarlos a medida que ocurren y mitigar la situación.
- Usar sistemas de detección de intrusos(IDS). Un IDS es capaz de detectar firmas en el tráfico de red que se sabe que explotan las vulnerabilidades de desbordamiento del buffer.