Defense in Depth: Protecting Against Zero-Day Attacks

Chris McNab

FIRST 16, Budapest 2004

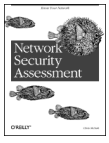
Agenda

- Exploits through the ages
- Discussion of stack and heap overflows
- Common attack behavior
- Defense in depth overview
- Kernel modifications
- PaX
- Q & A

Introduction

Who is Chris McNab?

- Technical Director of Matta Consulting, a London-based security services company providing training and assessment services to numerous large clients
- Author of *Network Security Assessment*



Exploits Through the Ages

Started out really simple

- PHF and other logic flaws back in 1994
- In 1996 the first buffer overflow was posted to BugTraq

Stack overflows were all the rage (1996 onwards)

- Return address (aka. saved EIP) overwritten
- Evil assembler placed on the stack (aka. shellcode)
- Easy to exploit reliably (most worms are stack overflows)
- In 2004 these are become increasingly rare

Heap overflows came next (1998 onwards)

- Since 1998 numerous heap overflows have been found
- Trickier to exploit reliably (no known heap worms)

Exploits Through the Ages

Then Stack Off-by-one bugs (2000 to present)

- The German group TESO pioneered this attack
- Numerous bugs identified, mainly in open source code

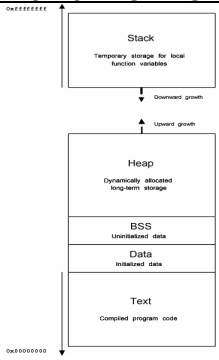
.. And don't forget Format String bugs (2001 to present)

- Easy to identify and exploit
- Abuses features of functions like printf () and syslog()

.. Or Integer overflows (2001 to present)

- Integer overflows are technically a delivery mechanism – resulting in a stack or heap overflow through code expecting a number of a certain size, and instead getting a larger number, or one that is negative. Integer overflows are complex to identify and exploit

Runtime Memory Layout



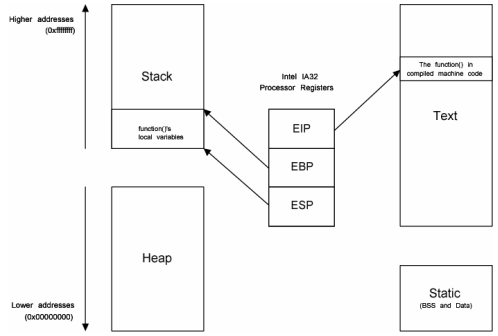
The Stack is used to store local function variables of a known size, such as answers to questions, flags and other manageable data. The stack grows downwards to lower memory addresses as more data is placed on it

Space is allocated on the heap dynamically when the size of the data to be stored isn't necessarily known. The heap represents the largest region of memory for most processes

BSS and Data segments are used to store global data that is set at runtime, such as authentication flags or permissions

The text segment contains the executable code and libraries for the program. The text is not written to during runtime

Processor Registers

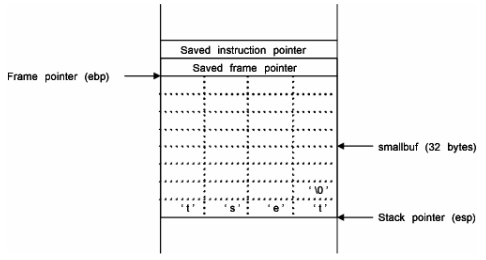


The Hacking Concept

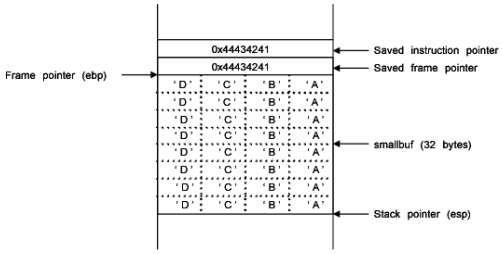
"Hacking is the art of manipulating a process so that it performs an action that is useful to you"

- By overwriting important values on the stack or heap, we can influence logical program flow, and often execute code
- I will now discuss stack and heap overflows..

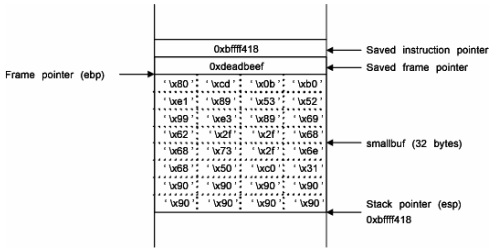
Using the Stack Legitimately



Overflowing the Stack



Doing Something Useful



What About the Heap?

The Heap is Very Different Depending on the Platform

- Linux (glibc)
- AIX
- BSD
- Windows
- Oracle and others use their own heap implementations

Heap Exploitation and Issues

- Heap exploitation requires a lot of research
- Many heap exploits use brute forcing of addresses
- Difficult to turn into worms

Examples of IIS Overflows

IDQ.DLL Overflow (aka. The Code Red bug)

CVE-2001-0500

```
GET /default.ida?[C x 240]=x HTTP/1.1
Host: www.example.org
Blah: [C x 10,000][shellcode]
```

Examples of IIS Overflows

MSW3PRT.DLL Overflow (aka. The .printer bug)

CVE-2001-0241

```
GET /NULL.printer HTTP/1.1
Blah: [shellcode]
Host: [ C x 420 ][Address of shellcode]
```

Examples of IIS Overflows

NTDLL.DLL Overflow (aka. The WebDAV bug)

CVE-2003-0109

```
SEARCH /[Address of shellcode x 300] HTTP/1.1
Host: www.example.org
Content-Type: text/xml
Content-Length: 480
<?xml version="1.0"?>
<g:searchrequest xmlns:g="DAV:">
<g:sql>
Select "DAV:displayname" from scope()
</g:sql>
</g:searchrequest>
0x01 [shellcode]
```

Common Attack Behavior

Network Activity

- A malformed request (HTTP, FTP, et al)
- Usually long arguments and variables are provided
- Request contains shellcode if an overflow is being abused

Host Activity

- A pointer is overwritten in memory, changing logical program flow (stack frame variables, heap control structures, structured exception handlers, et al)
- Shellcode is executed from the stack or heap, or..
- Suspicious system calls are made (system(), socket())

Defense in Depth Overview

We can use this behavior to pick up on zero-day attacks before they reach vulnerable components:

- Network monitoring and analysis – firewalls and IDS
- Server-based sanitizing of traffic – Microsoft URLScan

We also use local kernel modifications to protect logical program flow:

- Canary values in memory to prevent overflows
- Non-executable stack and heap
- Randomization of library system calls, GOT, PLT entries

Kernel Modifications

Stack & Heap Protection

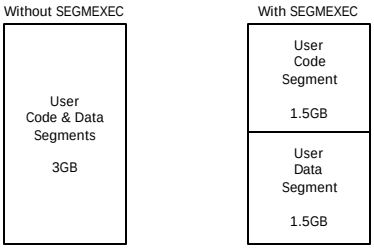
- Canary values to prevent clobbering of sensitive variables
- Non-executable stack and heap, preventing shellcode from being executed from the stack or heap
- Windows 2003 Server, XP SP2, OpenBSD, and Linux systems running kernel patches such as OpenWall..
- These mechanisms can be easily bypassed if an attacker has local access, by using overwriting function pointers and calling functions in common loaded libraries (such as system()) to compromise the system. Remote bypass is more tricky, but not impossible..

Kernel Modifications

Non-executable stack and heap are not enough!!

- Windows 2003 canary values and non-executable stack can be bypassed. See David Litchfield's paper at: **http://www.nextgenss.com/papers/defeating-w2k3-stack-protection.pdf**
- Need to prevent sensitive data on the stack from being overwritten and processed, as function pointers and exception handlers can be overwritten and result in code execution, even with canaries and non-exec stack/heap
- PaX (a Linux kernel patch) deals with this…

PaX



PaX uses a feature known as SEGMEXEC to mirror executable memory pages. Before the executable pages are fetched by the processor, they are matched to those in the other segment, to ensure validity.

PaX Continued..

Other features include:

- KERNEXEC, preventing kernel exploitation
- Randomization of everything:
  - Stack
  - Heap
  - Library images
  - Executable images
- It is impossible to execute arbitrary code in either user or kernel space if PaX is installed
- Linux only – part of www.grsecurity.net patches

The Future

Microsoft and others are behind

- Non-exec stack and heap do not end overflows
- Locally launched attacks are still successful
- Linux has had non-exec patches for 6 years, Windows 2003 Server and XP SP2 just got theirs..

Free trusted computing is here now..

- GRSec patches (www.grsecurity.net)
- SELinux (www.nsa.gov/selinux/)

Thank you all for coming!

Questions?

chris.mcnab@trustmatta.com

www.trustmatta.com