IBM AIX Heap Exploitation Methods

Presented by
Tim Shelton

HAWK NETWORK DEFENSE

About HAWK

HAWK Network Defense, Inc.

- *Application and Network Security company incorporated in Texas focused on Security Information & Event Management and other security services.*

About Tim Shelton

Tim Shelton

- *One of the first people to break out of virtualization in 2005 by breaking through VMWare's vmnat daemon, thereby compromising the host operating system.*
- *Involved with countless Microsoft vulnerabilities and exploits and was the first to publish a universal exploit for the 2009 Adobe JBIG2 vulnerability, along with Russell Sanford (xort).*
- *Also known as redsand, is an active security contributor, and is associated with blacksecurity.org*

Overview

HAWK Network Defense, Inc.

– *Importance of IBM AIX*
  - *Origins of IBM AIX*
  - *Mission Critical Applications*
– *Methods*
  - *Conquering leftmost*
  - *Conquering rightmost **new***
    – *Live demo*

IBM’s View of AIX

Businesses today need to maximize the return on investment in information technology. Their IT infrastructure should have the flexibility to quickly adjust to changing business computing requirements and scale to handle ever expanding workloads—without adding complexity. But just providing flexibility and performance isn’t enough; the IT infrastructure also needs to provide rock solid security and near-continuous availability and while managing energy and cooling costs.

These are just some of the reasons why more and more businesses are choosing the AIX operating system (OS) running on IBM systems designed with Power Architecture® technology.

AIX Version 6.1 – ibm.com

Importance of IBM AIX

- Enterprise Resource Planning
- Customer Relationship Management
- Mission Critical Databases
  - Oracle for AIX
  - IBM DB2 for AIX

Initial AIX Heap Exploitation Research

An Introduction to Heap overflows on AIX 5.3L

– *by David Litchfield, in August of 2005*

– heap's free()/rightmost() functions

So What Specifically Does This Cover?

– *Exploiting heap corruption when followed by a free() request.*

Initial AIX Heap Exploitation Research

An Introduction to Heap overflows on AIX 5.3L

- Litchfield's research pioneered heap exploitation analysis on IBM AIX 5.3
- Method from 5.3 also affects 6.1
- Results in a bi-directional double pointer overwrite

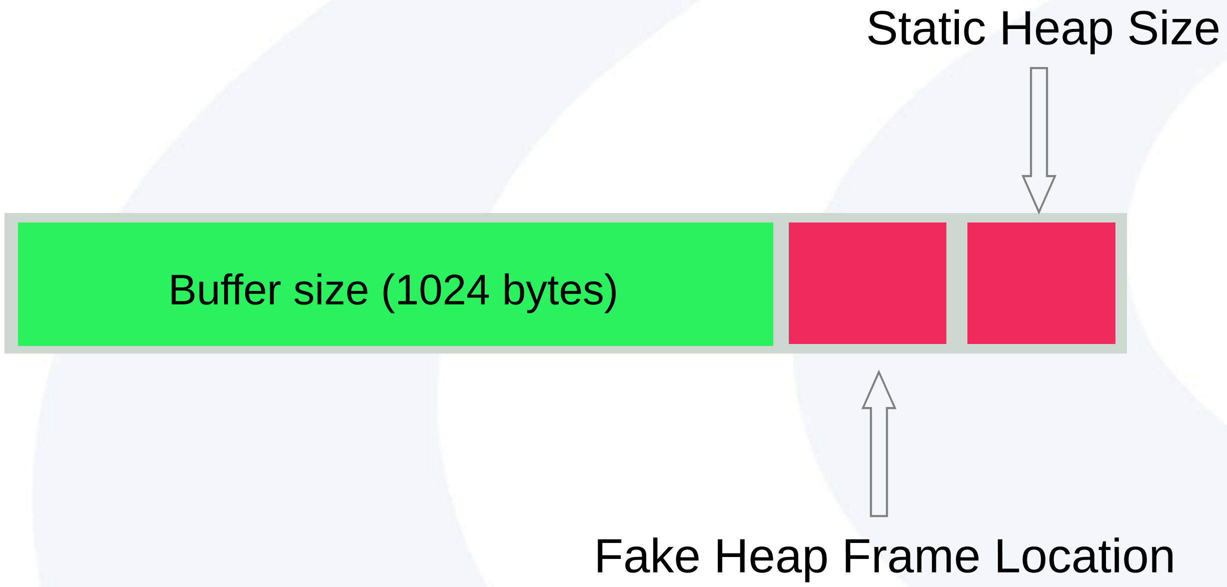
Initial AIX Heap Exploitation Research

Overview of Exploiting free()/rightmost()

- Overwrite our heap header during memory corruption
  - First 4 bytes – location of our fake heap frame in memory
  - Last 4 bytes – heap size, will match our heap size in the fake frame

Initial AIX Heap Exploitation Research

Overview of Exploiting free()/rightmost()



Initial AIX Heap Exploitation Research

Overview of Exploiting free()/rightmost()

- Create our fake heap frame in memory
    - Heap Frame is responsible for allowing Heap Structure to function and control as required
    - Heap Frame is an element of a linked list
    - Requires a controllable location of memory
    - Write 16 bytes containing our unique structure

Initial AIX Heap Exploitation Research

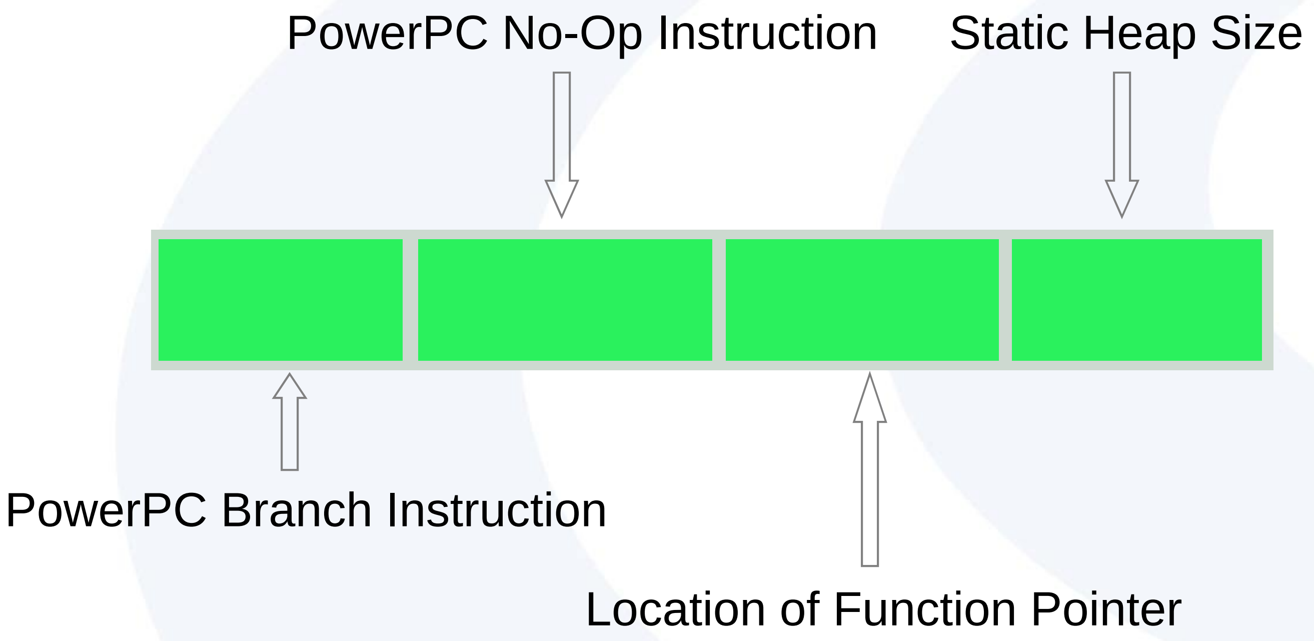
Overview of Exploiting free()/rightmost()

– Our Fake Frame – total of 16 bytes

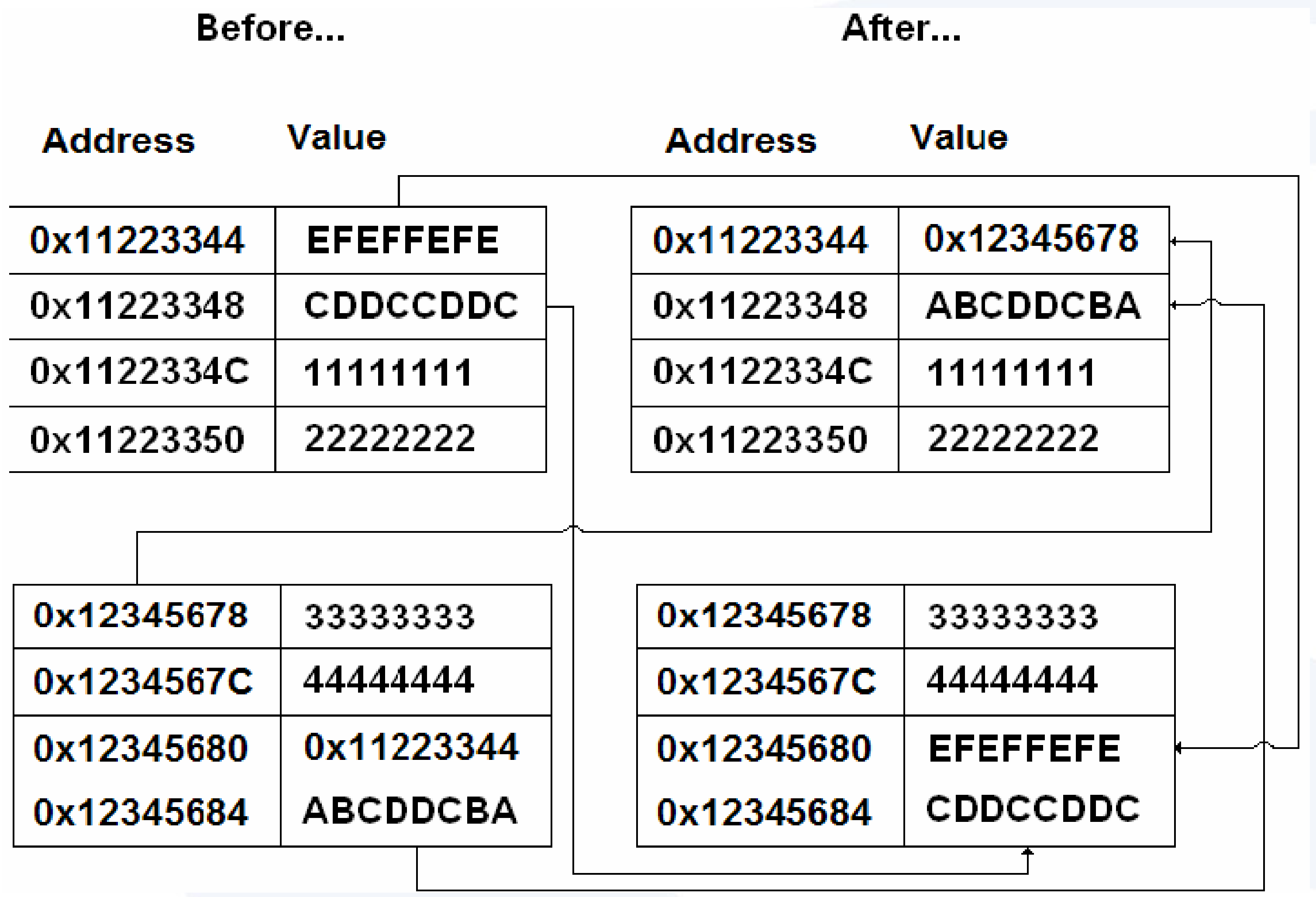
- First 4 bytes – PowerPC branch instruction
- Second 4 bytes – PowerPC no-op instruction
- Third 4 bytes – pointer to the value we want to overwrite
- Last 4 bytes – heap size, will match our heap size in the overflow

Initial AIX Heap Exploitation Research

Fake Frame – free()/rightmost()


PowerPC No-Op Instruction
Static Heap Size
PowerPC Branch Instruction
Location of Function Pointer

Initial AIX Heap Exploitation Research



Before...

Address	Value
0x11223344	EFEFFEFE
0x11223348	CDDCCDDC
0x1122334C	11111111
0x11223350	22222222

0x12345678	33333333
0x1234567C	44444444
0x12345680	0x11223344
0x12345684	ABCDDCBA

After...

Address	Value
0x11223344	0x12345678
0x11223348	ABCDDCBA
0x1122334C	11111111
0x11223350	22222222

0x12345678	33333333
0x1234567C	44444444
0x12345680	EFEFFEFE
0x12345684	CDDCCDDC

Note: image came from Litchfield's "An Introduction to Heap overflows on AIX 5.3L"

Initial AIX Heap Exploitation Research

```
int foo(char *arg) {
    char *ptr1 = NULL;
    ptr1 = (char *) malloc(20);
    strcpy(ptr1,arg);
    free(ptr1);
    return 0;
}
```

Initial AIX Heap Exploitation Research

Overview of Exploiting free()/rightmost()

- Pointers to Hijack
  - Hijack a saved link register on the stack
  - Hijack a callable function within the application’s export list
  - Hijack function pointers found within Heap or Stack

New AIX Heap Exploitation Research

Exploiting Heap's malloc()/leftmost functions

– heap's malloc()/leftmost() functions

So What Specifically Does This Cover?

– *Exploiting heap corruption when followed by a malloc() request.*

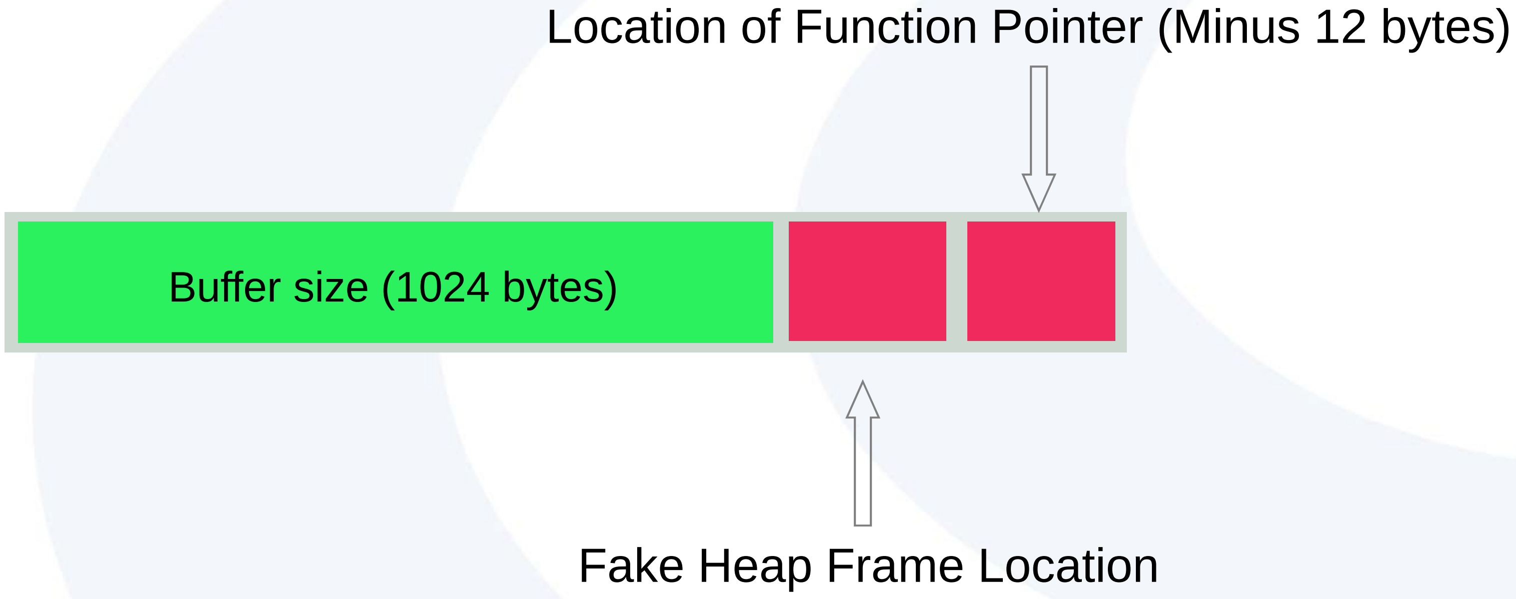
New AIX Heap Exploitation Research

Overview of Exploiting malloc()/leftmost()

- Create our fake heap frame in memory
- Overwrite our initial 8 bytes during memory corruption
  - First 4 bytes – location of our fake heap frame in memory
  - Last 4 bytes – heap size, should be the location of our shellcode

New AIX Heap Exploitation Research

Overview of Exploiting malloc()/leftmost()



New AIX Heap Exploitation Research

Overview of Exploiting malloc()/leftmost()

- Create our fake heap frame in memory
  - Heap Frame is an element of a linked list
  - Requires a controllable location of memory
  - Write 8 important bytes containing our unique structure
  - Still the same 16 byte structure for leftmost and rightmost

New AIX Heap Exploitation Research

Conquering leftmost/malloc Heap Corruption

– With minimal heap corruption

– Complete Code Execution

– Takes minutes to write

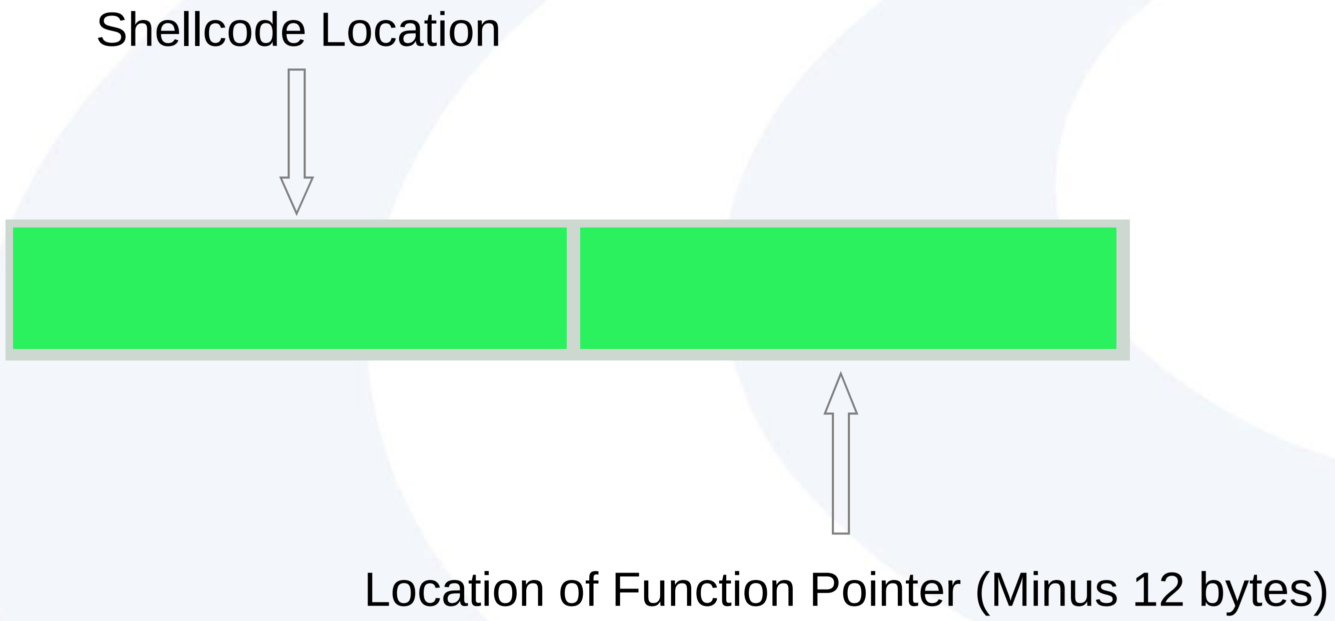
New AIX Heap Exploitation Research

Overview of Exploiting malloc()/leftmost()

- Our Fake Frame – total of 8 bytes
  - First 4 bytes – Location of our shellcode in memory
  - Second 4 bytes – Pointer to our value to Overwrite (minus 12 bytes)

New AIX Heap Exploitation Research

Fake Frame – malloc()/leftmost()



New AIX Heap Exploitation Research

Overview of Exploiting malloc()/leftmost()

- Pointers to Hijack
  - Attack the immediate saved link register on the stack
  - Corruption will occur later if not
    - Would require repairing heap headers

New AIX Heap Exploitation Research

Overview of Exploiting malloc()/leftmost()

```
(gdb) bt
#0  0xXXXXXXXX in leftmost () from /usr/lib/libc.a(shr.o)
#1  0xXXXXXXXX in malloc_y () from /usr/lib/libc.a(shr.o)
#2  0xXXXXXXXX in malloc_common@AF80_63 () from
   /usr/lib/libc.a(shr.o)
#3  0xXXXXXXXX in malloc () from /usr/lib/libc.a(shr.o)
```

New AIX Heap Exploitation Research

```
int foo(char *arg) {
    char *ptr1 = NULL, *ptr2=NULL;
    ptr1 = (char *) malloc(20);
    strcpy(ptr1,arg);
    ptr2 = (char *) malloc(0x1020);
    return 0;
}
```

Initial AIX Heap Exploitation Research

Live (pre-recorded) Demo

AIX Heap Exploitation Research

Additional Tips

- Help from Malloc Optoins
  - MALLOCDEBUG
  - MALLOCTYPE
  - MALLOCOPTIONS

AIX Heap Exploitation Research

Additional Tips

– Output debugging information to file

```
$ MALLOCDEBUG=output:/tmp/foo
```

AIX Heap Exploitation Research

Additional Tips

- Hunting for heap corruption
- Reverses Heap Growth
  - Grow Heap towards 0x00000000
- Corruption easier to detect/identify.

AIX Heap Exploitation Research

Additional Tips

- PowerPC instruction caching (icache)
  - This will affect your shellcode
  - Must use “immediate” instructions
  - Otherwise, unpredictable results

AIX Heap Exploitation Research

Additional Tips

- PowerPC instruction caching (icache)
    - ICU – Instruction Caching Unit
    - DCU – Data Caching Unit

AIX Heap Exploitation Research

The ICU supplies up to two instructions every cycle to the Fetch and Decode unit. The ICU can also forward instructions to the Fetch and Decode unit during a cacheline fill, minimizing execution stalls caused by instruction-cache misses. When the ICU is accessed, four instructions are read from the appropriate cacheline and placed temporarily in a line buffer. Subsequent ICU accesses check this line buffer for the requested instruction prior to accessing the cache array. This allows the ICU cache array to be accessed as little as once every four instructions, significantly reducing ICU power consumption.

- “PowerPC Architecture - Instruction Cache (I-Cache)“

AIX Heap Exploitation Research

Additional Tips

- PowerPC instruction caching (icache)
  - ICU – Instruction Caching Unit
    - Cached Instructions will affect shellcode
    - What you see is not what you get

Questions & Answers

Any Questions?

Tim Shelton
Sr. Vice President, Research & Development

HAWK Network Defense, Inc.

tshelton@hawkdefense.com