CSE 484 / CSE M 584
Computer Security: Buffer Overflows II

TA: Viktor Farkas
vfarkas@cs
Original slides by Franzi
Lab 1 Deadline Reminders

• Lab 1 Checkpoint (Sploits 1-3) **due April 18th at 8pm!**
  – Turn in text file of md5sums for sploits 1-3
    8a4d47b908dc53f760e8fa51b02bd440 sploit1.c
    545879cf5523e93be9a693111ee967e8 sploit2.c
    1cea0ba2bb9b5bb0fafe448a8a7bf0df sploit3.c

• Lab 1 Final due in two weeks (April 29th, 8pm).

• If you don’t have a group or access yet, talk to me today!

• Upcoming office hours:
  – Tomorrow (Friday) Thomas and Kevin - 2:00-3:00pm, CSE 021
Lab 1 Notes/Hints

• If you get stuck, move on!
• **Don’t procrastinate** on Sploits 4-7. Some of them are much harder.
• Sploit 3: No frame pointer, so you can only change last byte of saved EIP. Think about an existing instruction you could point to that would have desirable side effects.
• You have **more than one copy of your buffer**: (1) as argument to function, (2) where it gets copied.
• Sploit 4 is not necessarily harder than Sploit 3.
Sploit 5 Tips

• Buffer copied to the heap.
• Target 5 uses the implementation that’s found in ~/sources/tmalloc.c.
• Read “Once upon a free()”: http://www.phrack.org/issues.html?issue=57&id=9&mode=txt
Dynamic Memory Management in C

• Memory allocation: `malloc(size_t n)`
  – Allocates n bytes and returns a pointer to the allocated memory; memory not cleared.

• Memory deallocation: `free(void * p)`
  – Frees the memory space pointed to by p, which must have been returned by a previous call to `malloc()` (or similar).
    – If `free(p)` has been called before (“double free”), undefined behavior occurs.
  – If p is null, no operation is performed.

(Some memory management slides adapted from Vitaly Shmatikov)
Target5: What’s the problem?

```
char *p; char *q;

if ( (p = tmalloc(160)) == NULL)
{ exit(EXIT_FAILURE); }

if ( (q = tmalloc(160)) == NULL)
{exit(EXIT_FAILURE); }

tfree(p);
tfree(q);

if ( (p = tmalloc(320)) == NULL)
{exit(EXIT_FAILURE); }

obsd_strlcpy(p, arg, 320);

tfree(q);  // “Undefined” behavior on second free()
```
Free Chunks (as used in tmalloc.c)

- Chunks organized into doubly-linked list.
- Each chunk on list contains **forward/back pointers to next/previous chunks** in the list.
  - LSB of right pointer contains free bit.
  - Adjacent free chunks are consolidated.

<table>
<thead>
<tr>
<th>Previous pointer</th>
<th>Next pointer</th>
<th>User Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocated Chunk</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Previous pointer</th>
<th>Next pointer</th>
<th>Unused space (not cleared)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Chunk</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chunk Maintenance

One big free chunk:

Split to malloc:

Split to malloc (twice):

Free (twice):

Consolidate free chunks:
Chunks in tmalloc.c

• Lines 20-28 give chunk structure:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ptr to Left</strong></td>
<td><strong>Ptr to Right</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Data</strong></td>
</tr>
</tbody>
</table>

• Look at chunk consolidation in `tfree(p)`:

```c
q = p->s.l;
...
q->s.r = p->s.r;
p->s.r->s.l = q;
```

Hey look, if we control chunks p (and q), this code will write the value of q (address of buffer?) to a location we specify (location of saved EIP?).

• Goal: populate (fake) chunks appropriately.
Format string Vulnerability (6)

Print out the contents at the address 0x10014808 using format-string vulnerability

For %s: print out the contents pointed by this address

Credit: http://www.cis.syr.edu/~wedu/Teaching/cis643/LectureNotes_New/Format_String.pdf
General Questions?