Software Exploitation Techniques

Gianni Tedesco <gxt@cs.nott.ac.uk>

"I can tell you I wish those people just would be quiet. It would be best for the world. That's not going to happen, so we have to work in the right fashion with these security researchers."

-- Steve Balmer CTO/CEO Microsoft Corporation, 2003

Introduction

You might learn:

- What sort of software errors are security relevant.
- The theory of stack based buffer overruns.
- The animated-cursor error in USER32.DLL

But I will not explain:

- How to write shellcodes
- Methods for ”getting away with it”

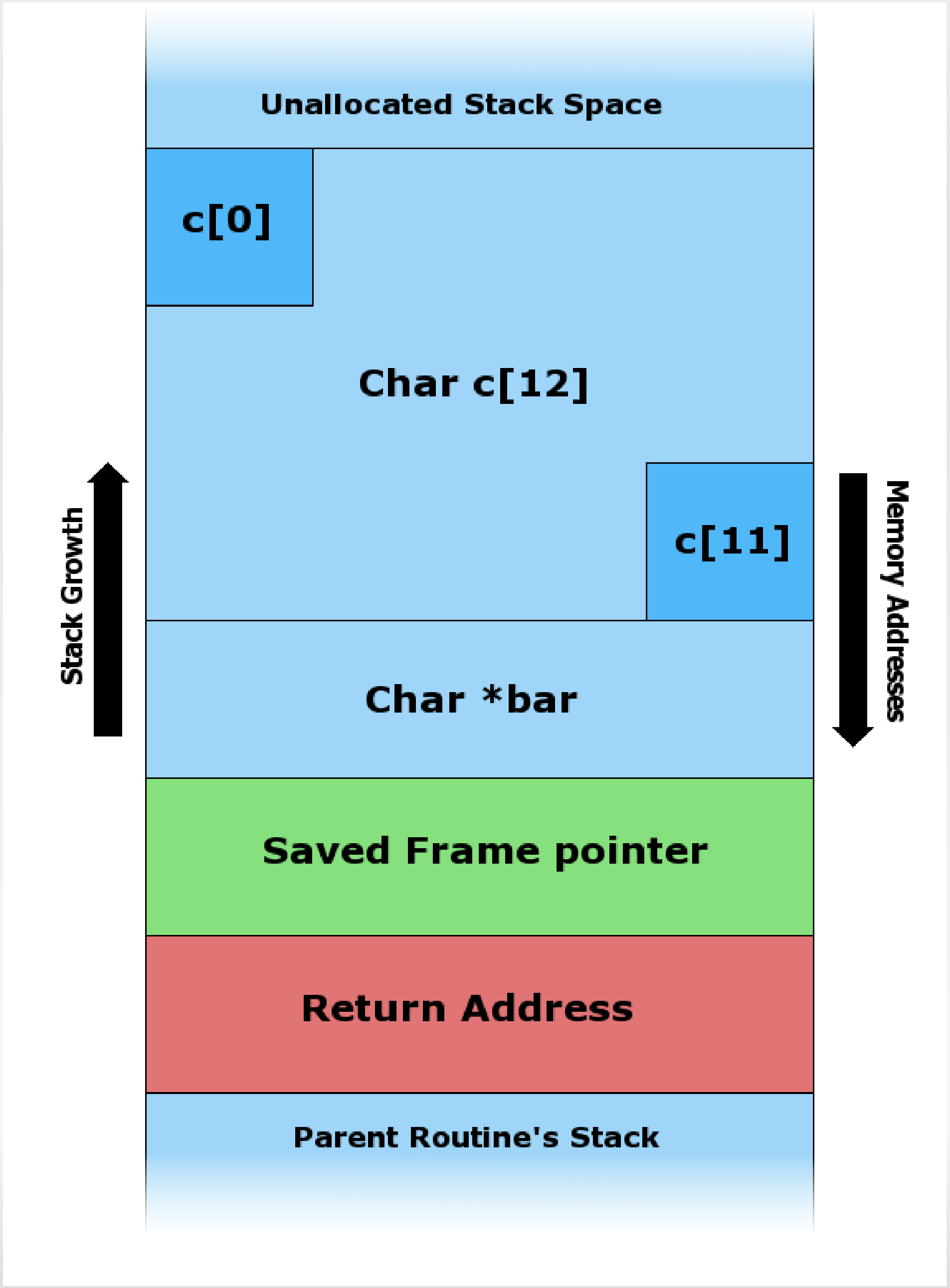
Errors and ”Bugs”

- Practically all software has errors.
- Errors lead to program behaviour unanticipated by the developers.
- Hard to say when errors can be used.
- Depends on all the little details.
- H4x0r d00ds have a little tool-bag of common techniques though.

Smashing the Stack

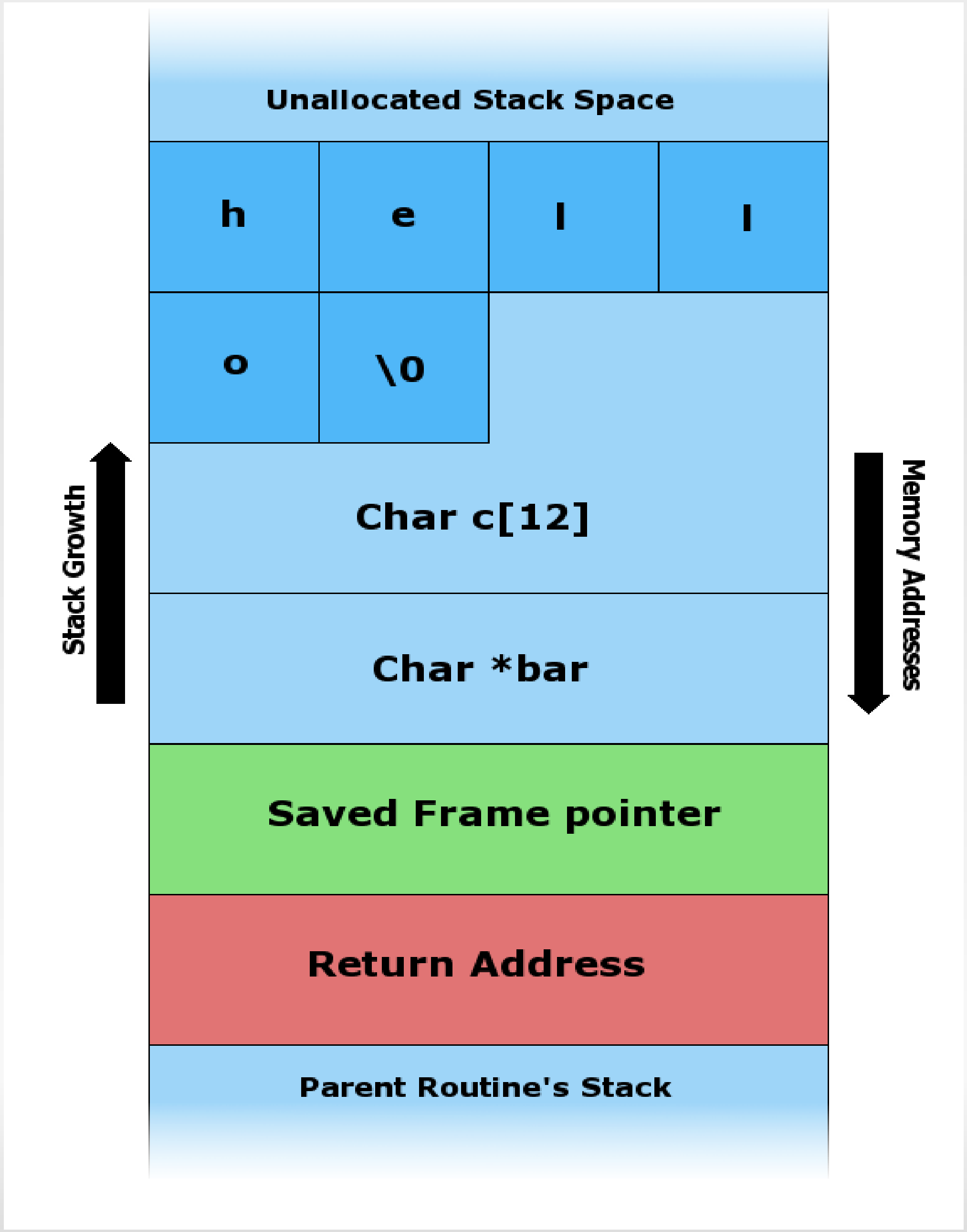
- Discovered in 1972. Computer Security Planning Study.
- Exploited in 1986. Morris worm.
- Published in phrack in 1994. Aleph one. ”Smashing the Stack for Fun and Profit.”
- Childs play these days!
- More complicated and obscure attacks exist now.

How it works #1



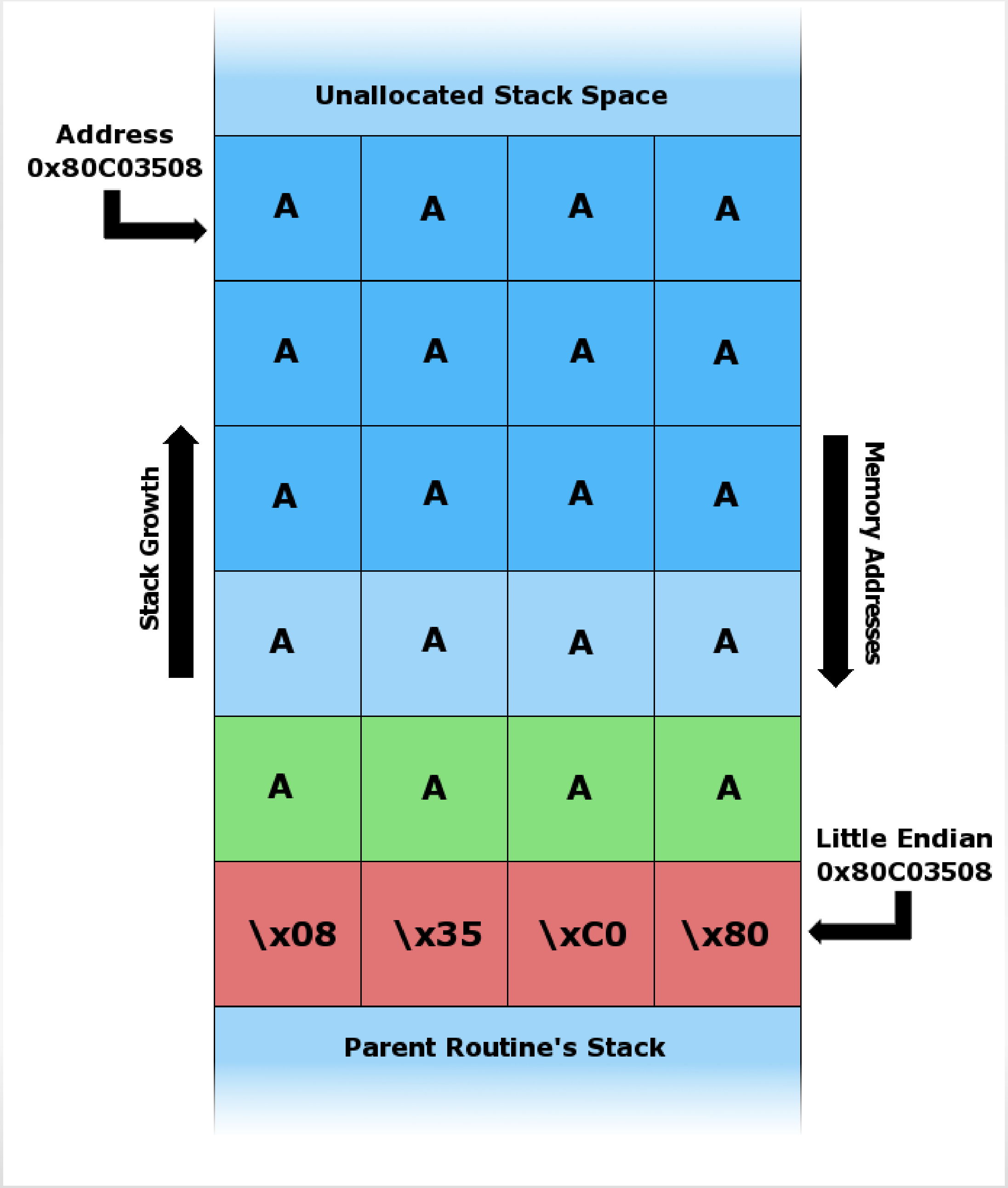
- Parameters in parent stack frame
- Locals in current stack frame
- To return, a function pops the return address in to the instruction pointer

How it Works #2



- ”Hello” fits in there no problems!

How it Works #3



- Oh dear
- Whoever wrote this string can overwrite the return address
- They can manipulate the control flow
- What value should they overwrite RET with?

LoadCursorFromFile Vulnerability

- Vulnerability exists in windows XP and Vista.
- Was patched, but the patch didn't fix it.
- Eventually after bad ANI files were spreading like the clap, a working patch was released.
- ANI is a type of riff file, like a WAV, AVI and other windows formats.
- It's a pretty complicated format.
- LoadCursorFromFile in USER32.DLL doen't validate it properly

Spot the Weakness

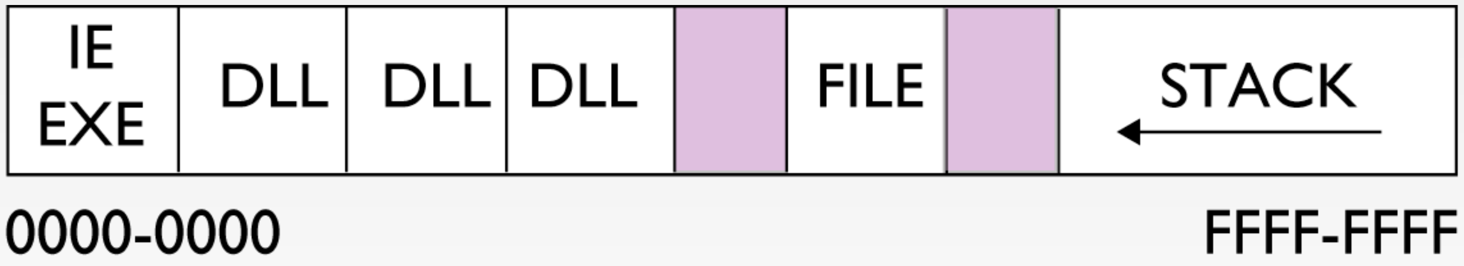
```c
void ReadAniFile(char *file, size_t len)
{
      struct riff_hdr *tmp;
      struct ani_hdr hdr;

      tmp = (struct riff_hdr *)file;
      memcpy(&hdr, tmp, tmp->chunk_size);

      /* Do some stuff */
}
```

Exploiting the Error

- If you can get someone to use an animated cursor. You can take over the program that loads it.
- I think the easiest way to do this is via Internet Explorer.
- It's easy to get someone to click on a URL.
- We can design a web page that tells IE to load our crafted cursor file whenever the mouse is over our web page.

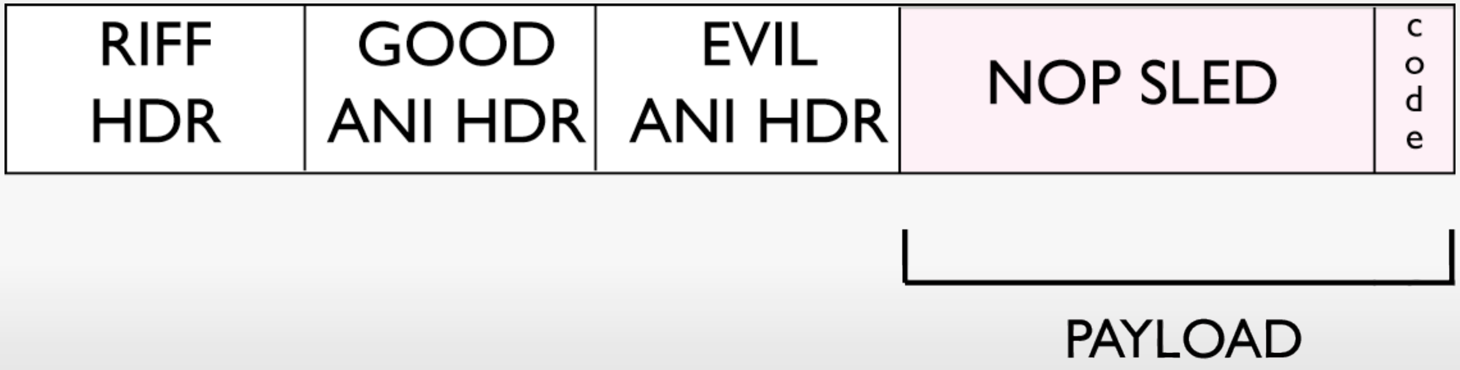
IE Memory Layout

- So we return to the problem of, what to overwrite RET with...
- Let's look at IE memory layout after it has loaded a cursor file.

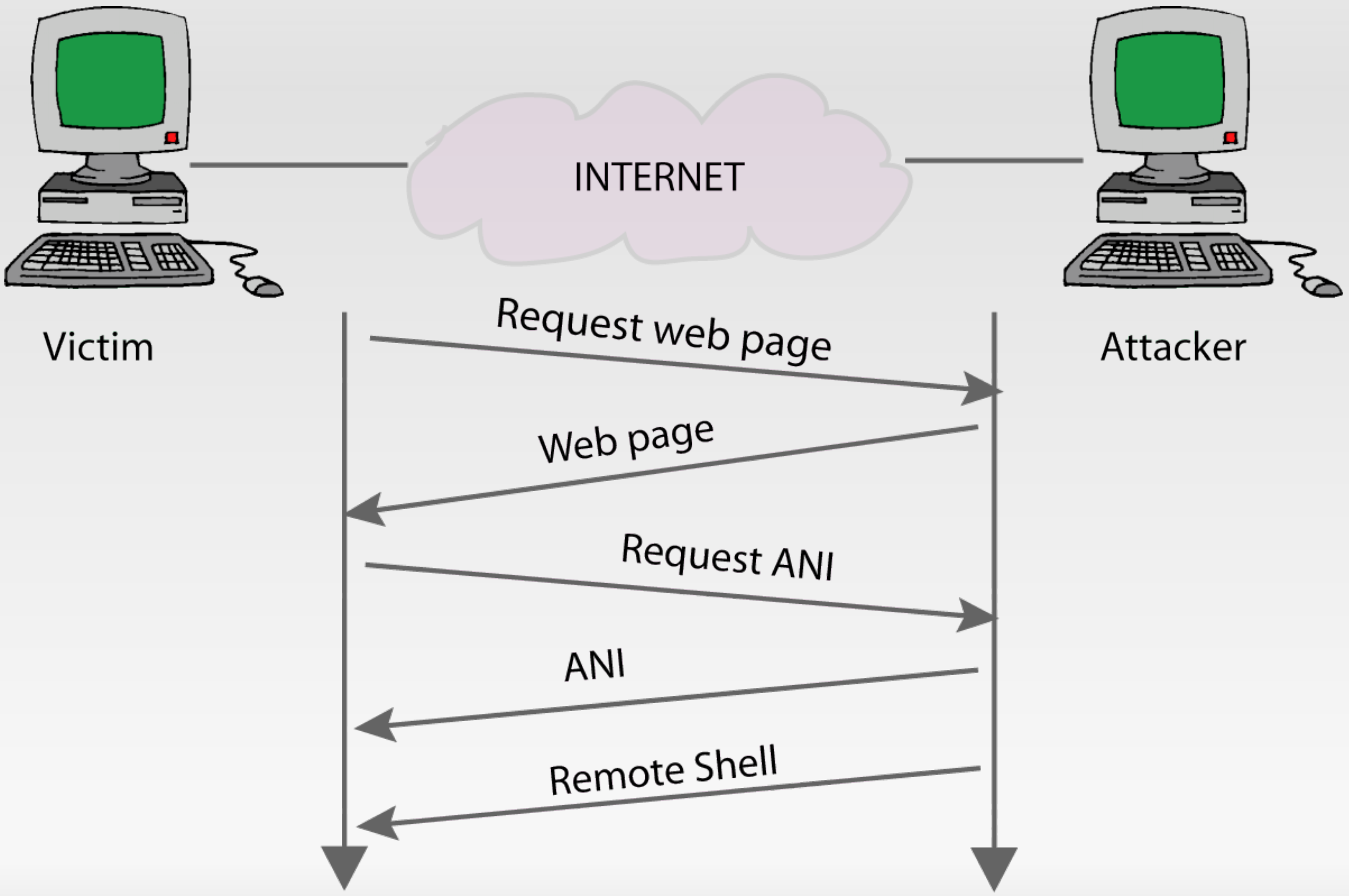


Making an Evil Animated Cursor

- So, when the vulnerable function returns, the whole ANI file is in memory.
- Let's just jump in to the ANI file and put some machine code in there.
- Decided to put the evil machine code in to the Copyright string section of the ANI.



Putting it All Together



Conclusions

- Errors like this are everywhere.
- There may be many kinds of exploitable errors which are not even known about.
- Computers are far too complicated to be able to say with certainty that they work as expected.
- Programmers first need to verify their design.
- Then they need to pay attention to every small detail when implementing.
- Oh, almost forgot... Don't be naughty now! :)

Thanks For Listening

"If you do publish then, worst case, all that will be accomplished is that you may cause a business somewhere to be compromised, and they may turn to you or your company for compensation for their financial losses. In the best case, due to September 11, 2001, you may end up on various government agencies' watch lists, and your potential career in the computer business may be altered in ways you did not intend."

-- Dan Grove. Hewlett Packard Software Security.