Smashing the Stack

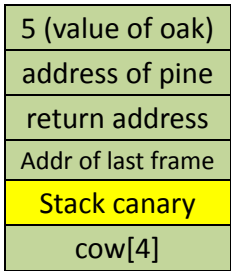
COMP620

Preventing Stack Overflow Exploits

- Better software engineering
- Avoid dangerous functions
- Language choice
- Compiler tools (Stack Guard)
- Analysis tools
- Execution Prevention

Stack Canaries

- A stack canary is a random number placed on the stack between the user data and the return address.
- Overflowing the local variable and changing the return address will also change the stack canary
- Before returning, the program checks the canary value.



Data Execution Prevention

- Most newer processors have a bit in the page table that inhibits instruction fetches from that page.
- Newer operating systems can set data execution prevention for stacks.
- This prevents the program from executing machine language loaded on the stack by an exploit.
- This does **not** prevent programs from overwriting the return address.

Stack Overflow Threats

- Most students:
  - Can't recognize a buffer overflow vulnerability when they see it, so they don't even think of it when coding
  - Are not attuned into the dangers of buffer overflows
  - Do not inspect or test their code as well as you would like
  - Are often not made aware of buffer overflows by instructors or textbooks

Online Examples

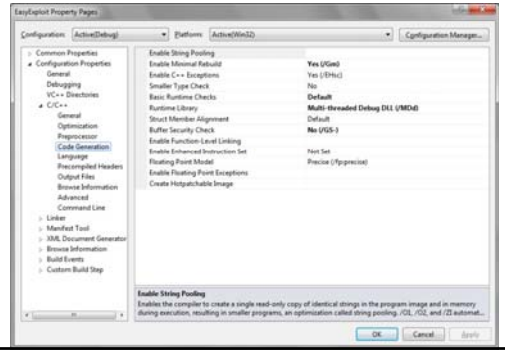
http://williams.comp.ncat.edu/overflow/Labs.html

Microsoft Defenses

- The Microsoft Visual C++ Express compiler takes several steps to avoid stack overflow
  - It uses stack canaries to detect overflow
  - The stack starts at a random address that is different for each execution of the same program.

Turning Off the Defenses

- To more easily attack the stack, we can turn off the Visual C++ security features.
- Select Project → Properties → Configuration Properties → C/C++ → Code Generation



Turning Off the Defenses

- To more easily attack the stack, we can turn off the Visual C++ security features.
- Set **`Basic Runtime Checks`** to **`default`**
- Set **`Buffer Security Check`** to **`No`**
- When using Visual C++ Express version 2010, select Tools → Settings → Expert Settings

Using the Debugger

- If you hover the mouse over a method name, it will give you the start address of the method
- You can also find the address of variables
- Once you know the start address of a method, the return address of a function called in that method will be a little bit higher.

Memory View

- Once you know the address of a variable on the stack, you can look at that variable and the memory around it with the debugger.
- Look for a likely return address or other data values.

Creating Attack Text

- It is helpful to have a text editor that allows you to input hexadecimal values.
- A free hex editor is Frhed, available from http://frhed.sourceforge.net