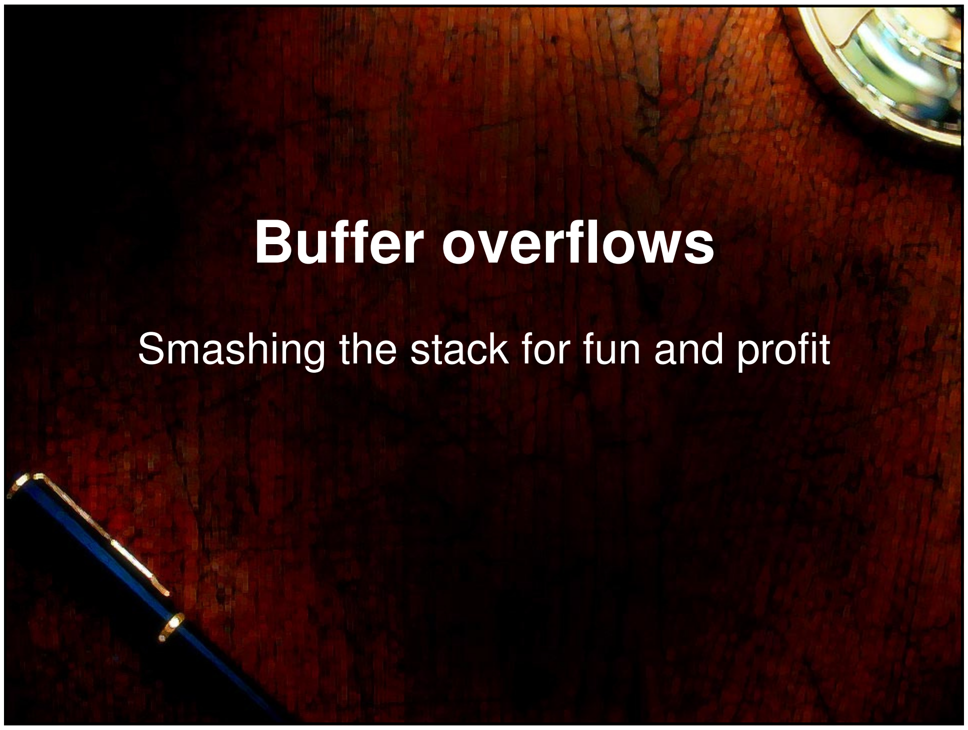
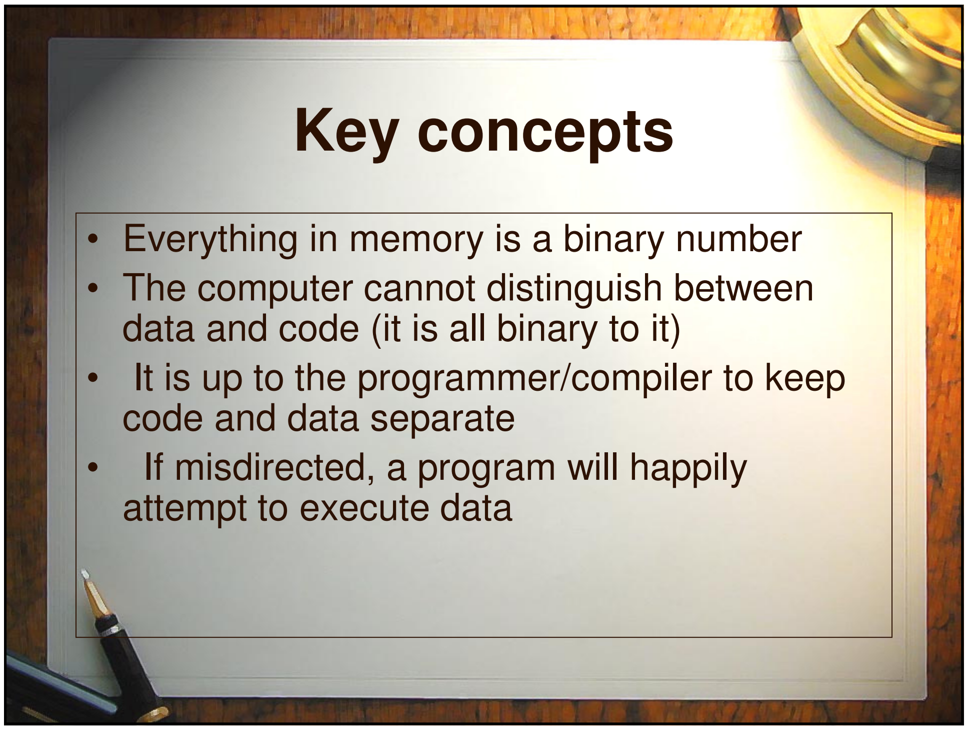


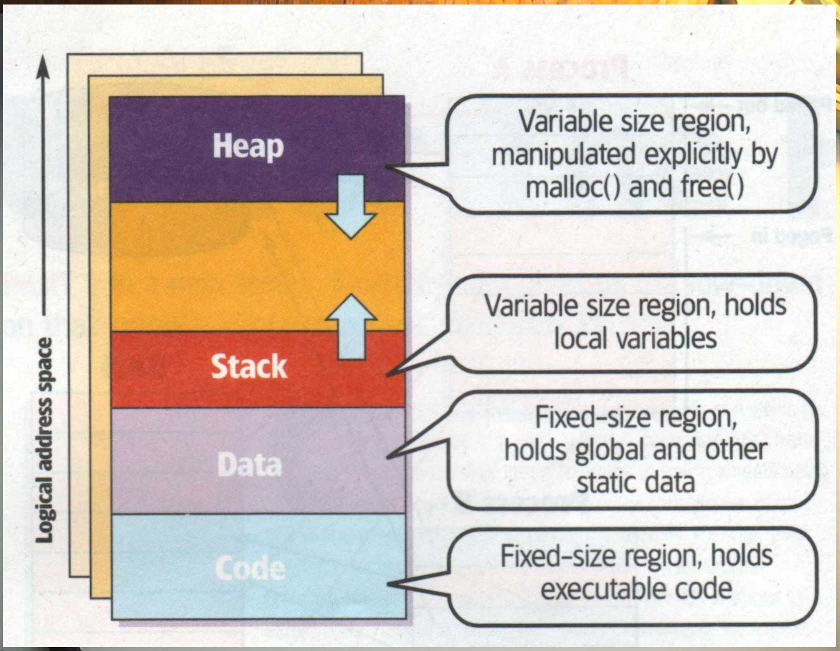
Buffer overflows

Smashing the stack for fun and profit



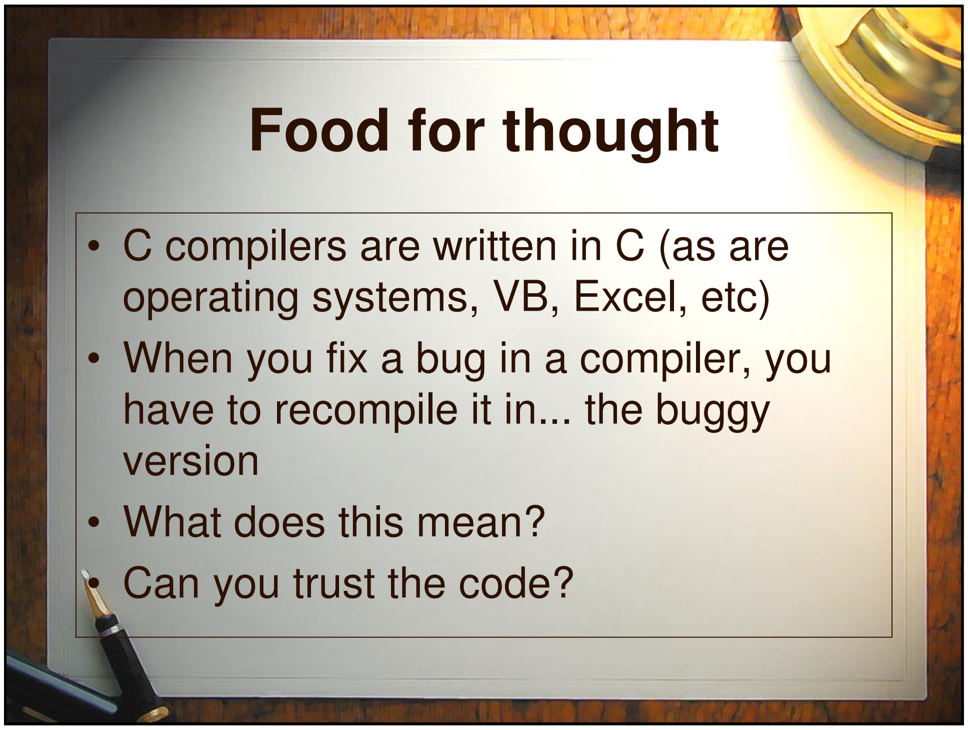
Key concepts

- Everything in memory is a binary number
- The computer cannot distinguish between data and code (it is all binary to it)
- It is up to the programmer/compiler to keep code and data separate
- If misdirected, a program will happily attempt to execute data


Logical address space
Heap
Stack
Data
Code
Variable size region, manipulated explicitly by malloc() and free()
Variable size region, holds local variables
Fixed-size region, holds global and other static data
Fixed-size region, holds executable code

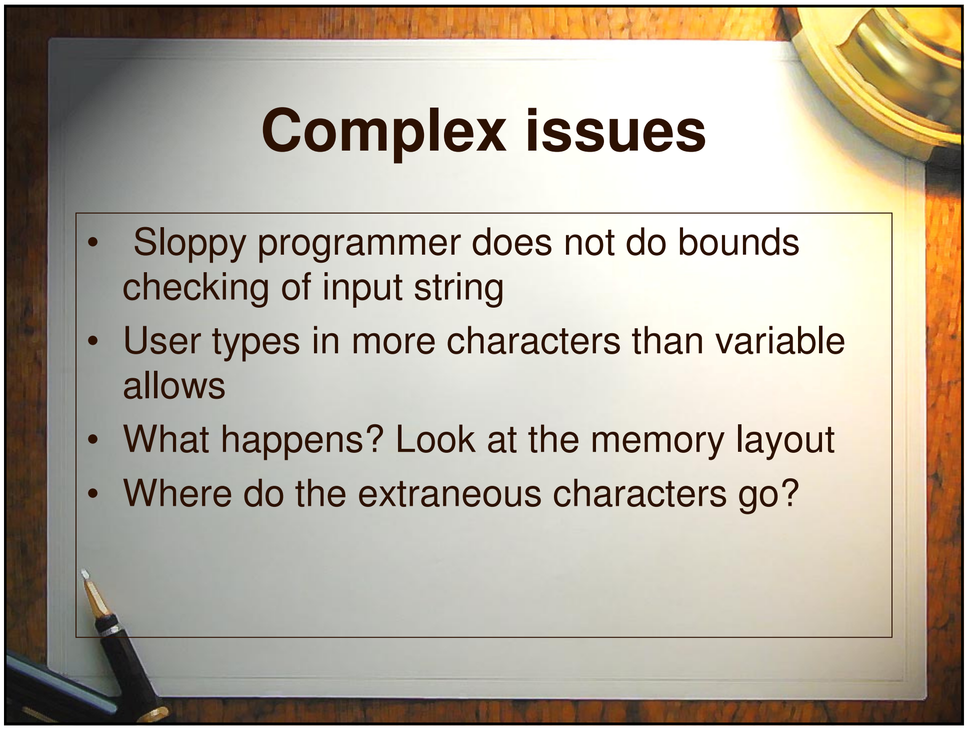
Key concepts #2

- Bounds checking is critical (variables, arrays, structures)
- C by default does not really check bounds (variable or array sizes)
- Most software is still written in C



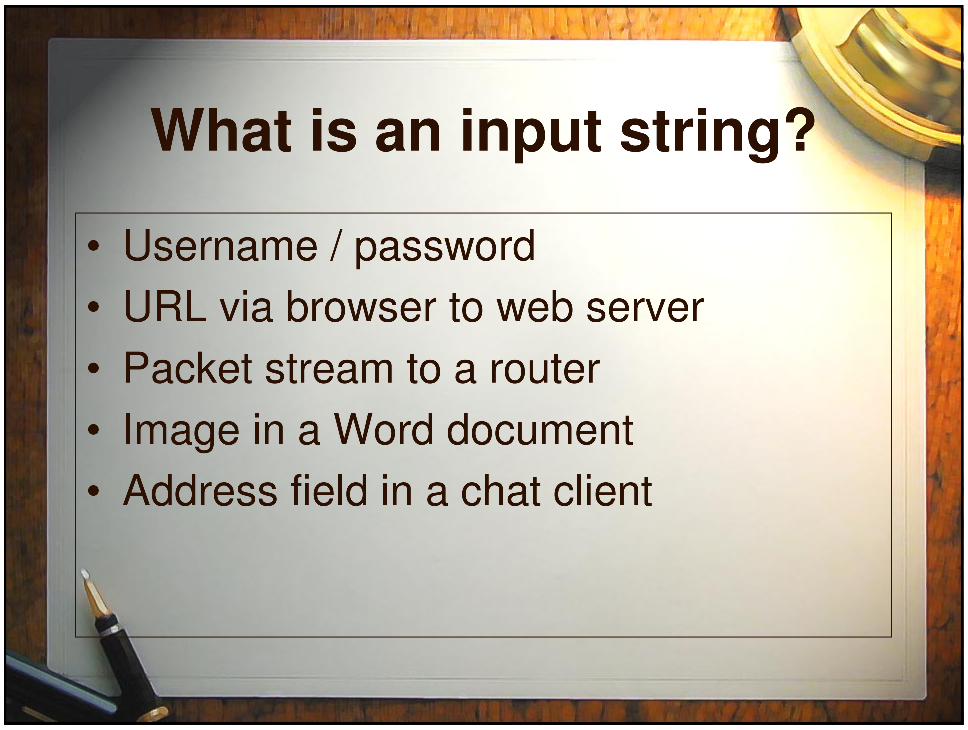
Food for thought

- C compilers are written in C (as are operating systems, VB, Excel, etc)
- When you fix a bug in a compiler, you have to recompile it in... the buggy version
- What does this mean?
- Can you trust the code?



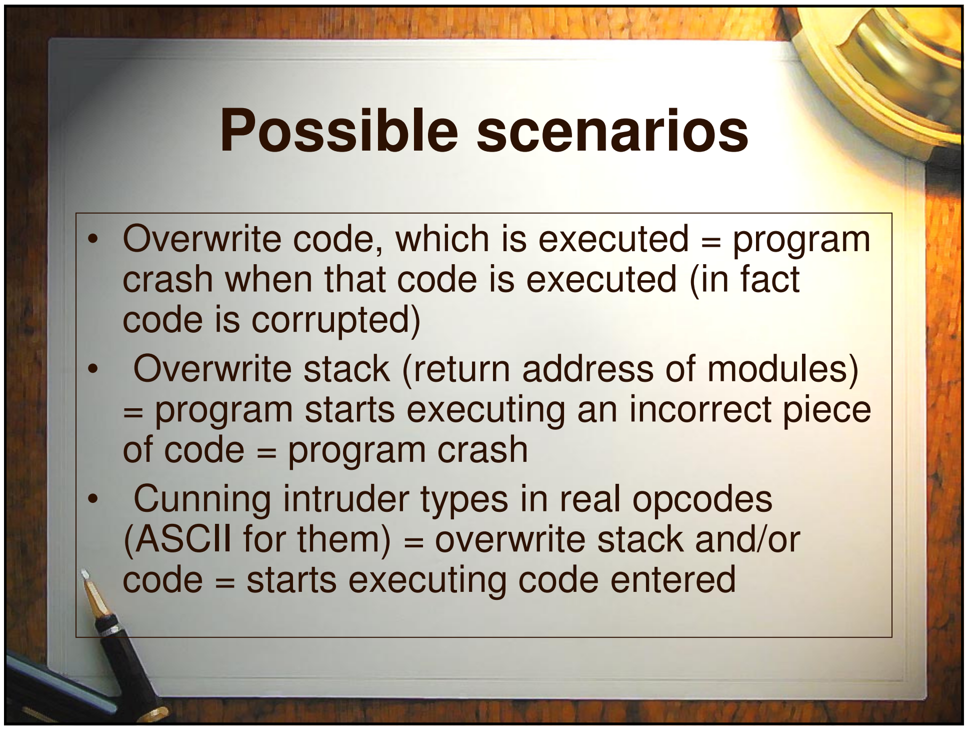
Complex issues

- Sloppy programmer does not do bounds checking of input string
- User types in more characters than variable allows
- What happens? Look at the memory layout
- Where do the extraneous characters go?



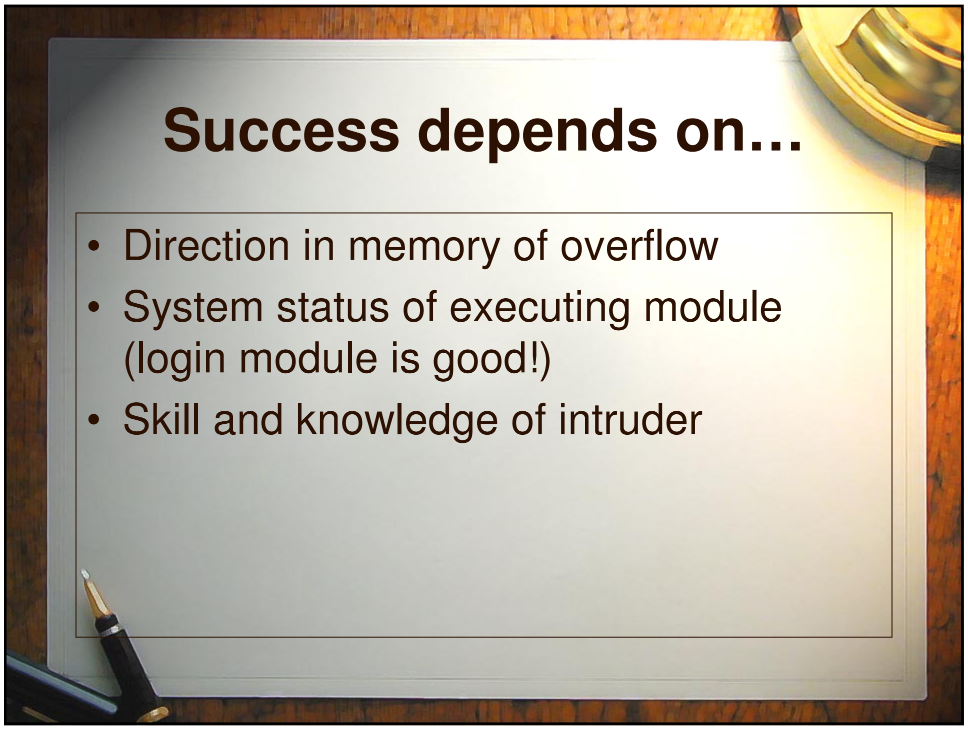
What is an input string?

- Username / password
- URL via browser to web server
- Packet stream to a router
- Image in a Word document
- Address field in a chat client



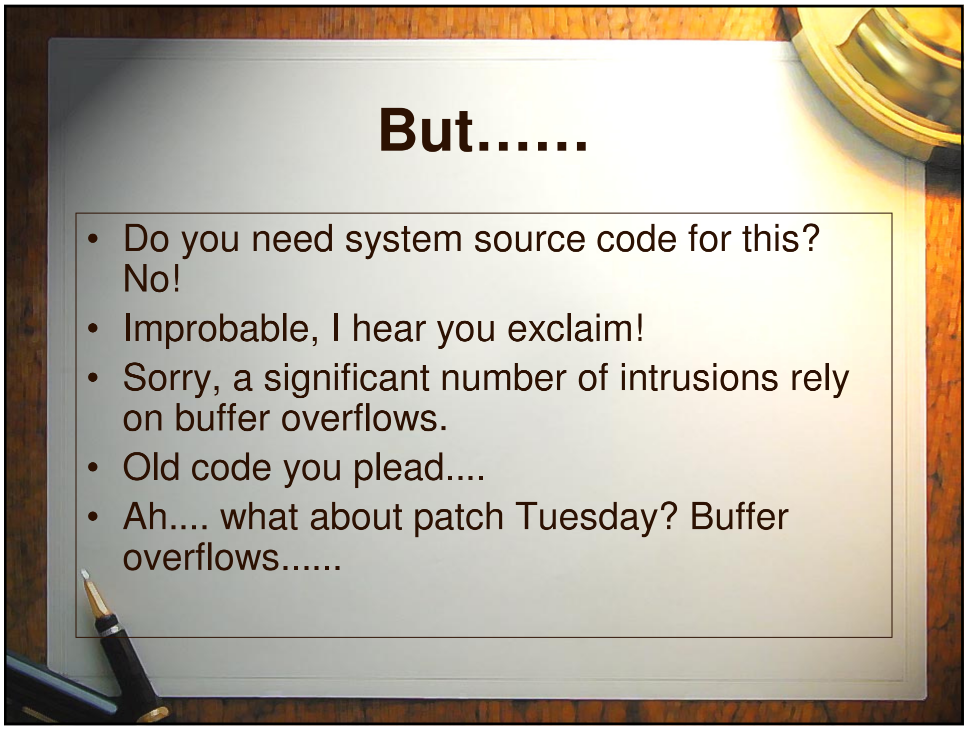
Possible scenarios

- Overwrite code, which is executed = program crash when that code is executed (in fact code is corrupted)
- Overwrite stack (return address of modules) = program starts executing an incorrect piece of code = program crash
- Cunning intruder types in real opcodes (ASCII for them) = overwrite stack and/or code = starts executing code entered



Success depends on…

- Direction in memory of overflow
- System status of executing module (login module is good!)
- Skill and knowledge of intruder



But……

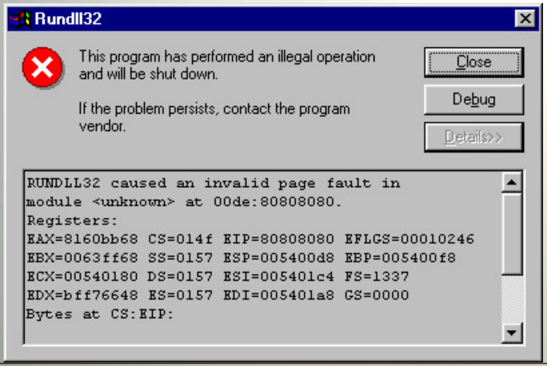
- Do you need system source code for this? No!
- Improbable, I hear you exclaim!
- Sorry, a significant number of intrusions rely on buffer overflows.
- Old code you plead....
- Ah.... what about patch Tuesday? Buffer overflows......

Have we learnt anything?

- Lots of spending on “security”
- Exploit - Patch - Exploit cycle - crazy?
- Is there a better way?
- What about hardware protection?
- What about software redesign?

What is this?

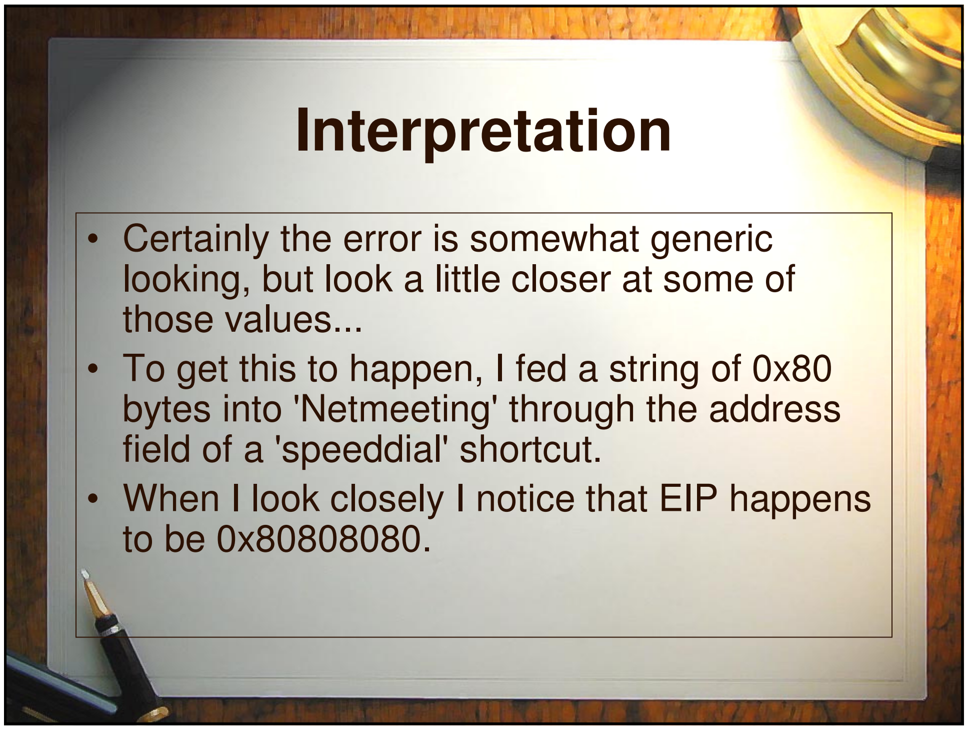
Interpreting crash windows



This program has performed an illegal operation and will be shut down.

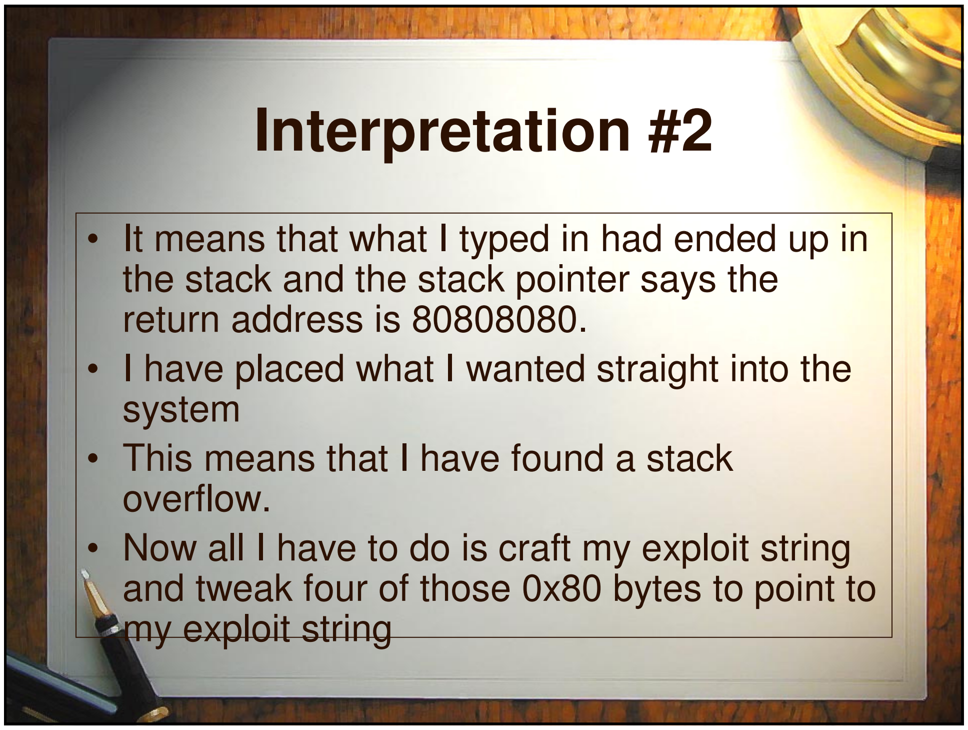
If the problem persists, contact the program vendor.

```
RUNDLL32 caused an invalid page fault in
module <unknown> at 00de:80808080.
Registers:
EAX=8160bb68 CS=014f EIP=80808080 EFLGS=00010246
EBX=0063ff68 SS=0157 ESP=005400d8 EBP=005400f8
ECX=00540180 DS=0157 ESI=005401c4 FS=1337
EDX=bff76648 ES=0157 EDI=005401a8 GS=0000
Bytes at CS:EIP:
```



Interpretation

- Certainly the error is somewhat generic looking, but look a little closer at some of those values...
- To get this to happen, I fed a string of 0x80 bytes into 'Netmeeting' through the address field of a 'speeddial' shortcut.
- When I look closely I notice that EIP happens to be 0x80808080.



Interpretation #2

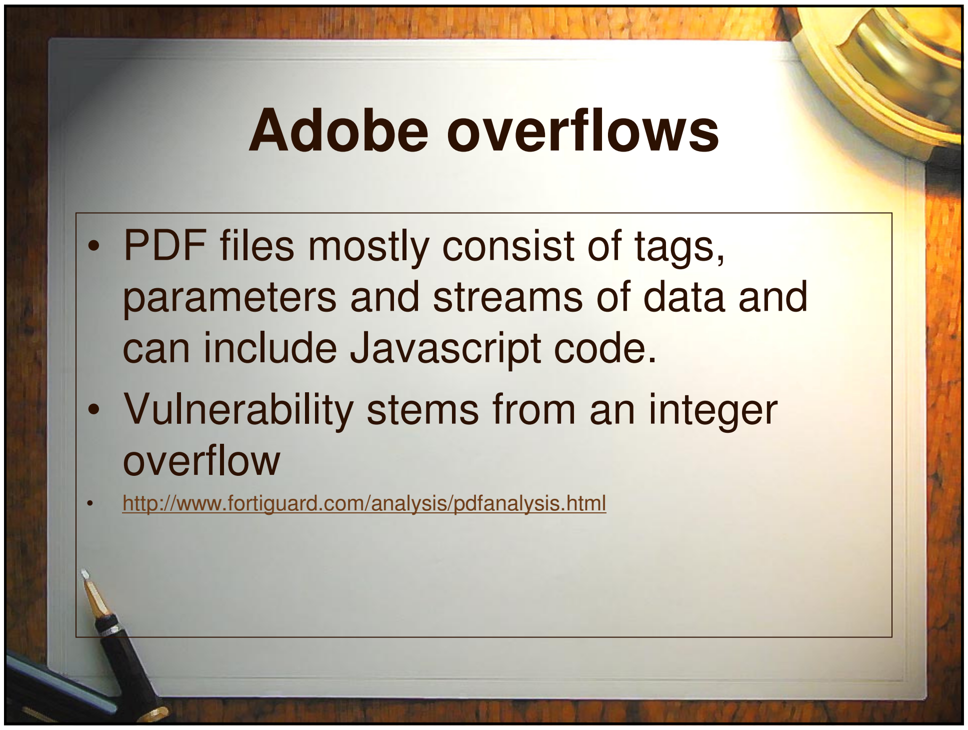
- It means that what I typed in had ended up in the stack and the stack pointer says the return address is 80808080.
- I have placed what I wanted straight into the system
- This means that I have found a stack overflow.
- Now all I have to do is craft my exploit string and tweak four of those 0x80 bytes to point to my exploit string

Current overflows

- Vista SP2 and prior and Server 2008 contain a vulnerability that could allow an unauthenticated, remote attacker to cause a denial of service (DoS) condition or execute arbitrary code.
- The vulnerability is due to errors when processing protocol headers in Server Message Block version 2 (SMB2) Negotiate Protocol Request messages. An unauthenticated, remote attacker could exploit this vulnerability by sending a malicious network request to the vulnerable system. Successful exploitation could allow the attacker to cause the vulnerable system to restart, resulting in a DoS condition. Alternatively, the attacker may be able to execute arbitrary code.
- Exploit code that can achieve code execution is publicly available
- Microsoft has released a security bulletin: MS09-050 (October 2009)

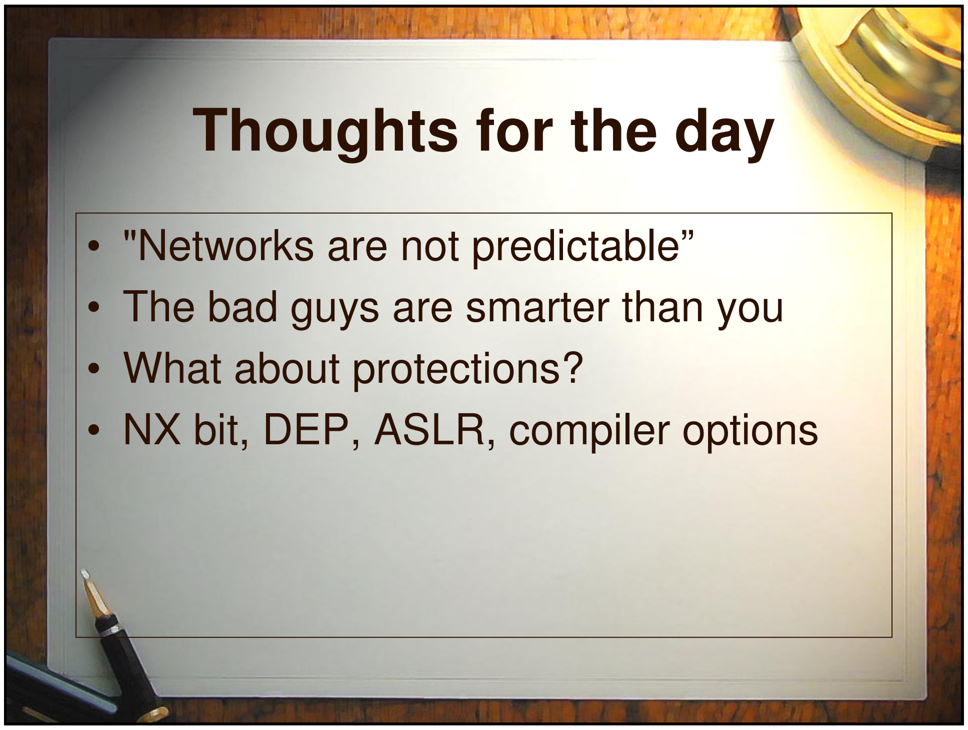
Word overflows

- Microsoft Office Word contains a vulnerability that could allow an unauthenticated, remote attacker to execute arbitrary code.
- This vulnerability exists due to insufficient boundary restrictions on parameters within Word documents. An unauthenticated, remote attacker could exploit this vulnerability by convincing a user to view a malicious document. A successful exploit could allow the attacker to execute arbitrary code with the privileges of the user
- Microsoft has released a security bulletin : MS09-027 (June 2009)



Adobe overflows

- PDF files mostly consist of tags, parameters and streams of data and can include Javascript code.
- Vulnerability stems from an integer overflow
- http://www.fortiguard.com/analysis/pdfanalysis.html



Thoughts for the day

- "Networks are not predictable”
- The bad guys are smarter than you
- What about protections?
- NX bit, DEP, ASLR, compiler options