CS 194-1 (CS 161) Computer Security

Lecture 13

Software security; Common implementation flaws; Principles

October 16, 2006 Prof. Anthony D. Joseph http://cs161.org/

Goals for Today

- Next 3 lectures are about software security
 Can have perfect design, specification, algos, but still have implementation vulnerabilities!
- Examine common implementation flaws - Many security-critical apps use C, and C has peculiar pitfalls
- Implementation flaws can occur with improper use of language, libraries, OS, or app logic

Lec 13.2

- · Principles for building secure systems
 - Trusted computing base (TCB)
 - Three Cryptographic principles
- 13 other security principles
- 10/16/06 Joseph CS161 ©UCB Fall 2006

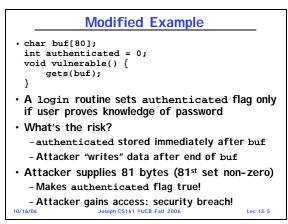
Buffer Overrun Vulnerabilities • Most common class of implementation flaw • C is basically a portable assembler • Programmer exposed to bare machine • No bounds-checking for array or pointer accesses • Buffer overrun (or buffer overflow) vulnerabilities • Out-of-bounds memory accesses used to corrupt program's intended behavior

10/16/06

Joseph CS161 ©UCB Fall 2006

Lec 13.3

Simple Example char buf[80]; void vulnerable() { gets(buf); 3 gets() reads all input bytes available on stdin, and stores them into buf[] · What if input has more than 80 bytes? - gets() writes past end of buf, overwriting some other part of memory - This is a bug! Results? - Program crash/core-dump? - Much worse consequences possible... 10/16/06 Lec 13.4



More Serious Exploit Example
<pre>• char buf[80]; int (*fnptr)();</pre>
•••
 Function pointer fnptr invoked elsewhere
 What can attacker do?
 Can overwrite fnptr with any address, redirecting program execution!
Crafty attacker:
 Input contains malicious machine instructions, followed by pointer to overwrite fnptr
 When fnptr is next invoked, flow of control re-directed to malicious code
• This is a malicious code injection attack 10/16/06 Jeseph CS161 @UCB Fall 2006 Lec 13.6



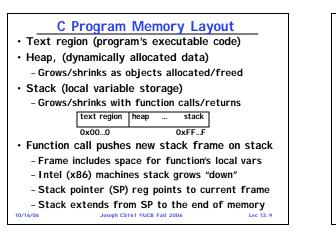
- Demonstrate how adversaries might be able to use a buffer overrun bug to seize control - This is very bad!
- Consider: web server receives requests from clients and processes them
 - With a buffer overrun in the code, malicious client could seize control of server process
 - If server is running as root, attacker gains root access and can leave a backdoor
 - » System has been "Owned"
- Buffer overrun vulnerabilities and malicious code injection attacks are primary/favorite method used by worm writers
 Joseph CS161 EUCB Fail 2006
 Lec 13.7

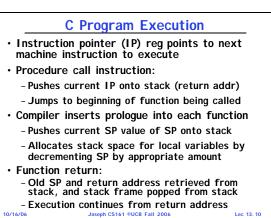
Buffer Exploit History • How likely are the conditions required to exploit buffer overruns? • Actually fairly rare... • But, first Internet worm (Morris worm) spread using several attacks • One used buffer overrun to overwrite authenticated flag in in.fingerd (network finger daemon) • Attackers have discovered much more effective methods of malicious code injection...

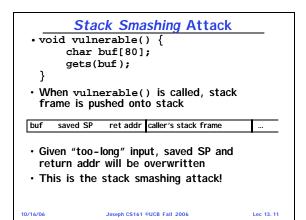
Joseph CS161 ©UCB Fall 2006

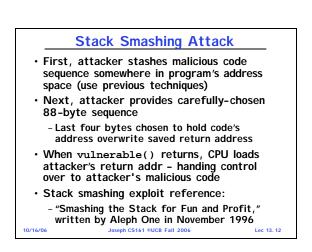
Lec 13.8

10/16/06









Buffer Overrun Summary

- · Techniques for when:
 - Malicious code gets stored at unknown location
 - Buffer stored on the heap instead of on stack
 - Can only overflow buffer by one byte
 - Characters written to buffer are limited (e.g., only uppercase characters)
- Exploiting buffer overruns appears mysterious, complex, or incredibly hard to exploit
 - Reality it is none of the above!
- · Worms exploit these bugs all the time - Code Red II compromised 250K machines by exploiting IIS buffer overrun Joseph CS161 ©UCB Fall 2006 10/16/06 Lec 13, 13

Buffer Overrun Summary · Historically, many security researchers have underestimated opportunities for obscure and sophisticated attacks - Very easy mistake to make ... · Lesson learned: - If your program has a buffer overrun bug, assume that the bug is exploitable and an attacker can take control of program · Buffer overruns are bad stuff - you don't want them in your programs! 10/16/06 Joseph CS161 ©UCB Fall 2006 Lec 13.14

• void vulne	erable() {	
char buf		
return	-	ULL)
printf(bu }	11);	
• Do you se	e the bug?	
• Last line s	<pre>should be printf("%s", buf)</pre>	
for non-	contains "%" chars, printf() will existent args, and may crash or ying to chase missing pointers	
• Reality is	worse	

10/16/06

Joseph CS161 ©UCB Fall 2006

Lec 13.15

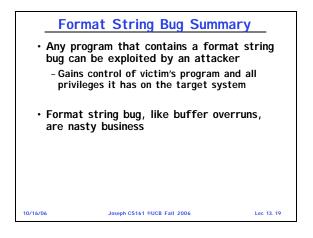
Attack Examples

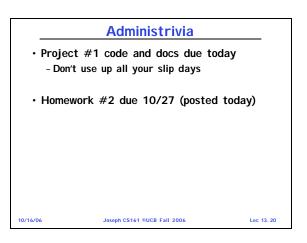
- Attacker can learn about function's stack frame contents if they can see what's printed - Use string "%x:%x" to see the first two words of
- stack memory
- What does this string ("%x:%x:%s") do?
- Prints first two words of stack memory
- Treats next stack memory word as memory addr and prints everything until first ' $\0'$
- Where does that last word of stack memory come from?
- Somewhere in printf()'s stack frame or, given enough %x specifiers to walk past end of printf()'s stack frame, comes from somewhere in vulnerable()'s stack frame 10/16/06 Joseph CS161 ©UCB Fall 2006 Lec 13, 16

A Further Refinement

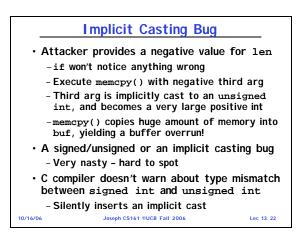
- buf is stored in vulnerable()'s stack frame
 - Attacker controls buf's contents and, thus, part of vulnerable()'s stack frame
 - Where %s specifier gets its memory addr!
- Attacker stores addr in buf, then when %s reads a word from stack to get an addr, it receives the addr they put there for it...
 - Exploit: "\x04\x03\x02\x01:%x:%x:%x:%s"
 - Attacker arranges right number of %x's, so addr is read from first word of buf (contains 0×01020304)
 - Attacker can read any memory in victim's address space - crypto keys, passwords... Joseph CS161 ©UCB Fall 2006 Lec 13, 17

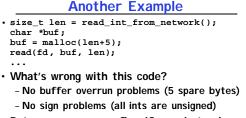
_	Yet More Troubles	
۰E	Even worse attacks possible!	
	- If the victim has a format string bug	
v	Jse obscure format specifier (%n) to write any value to any address in the <i>v</i> ictim's memory	
_	Enables attackers to mount malicious con njection attacks	ode
	 Introduce code anywhere into victim's memory 	
	- Use format string bug to overwrite retu address on stack (or a function pointer) with pointer to malicious code	
/16/06	Joseph CS161 ©UCB Fall 2006	Lec 13. 18

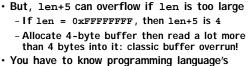




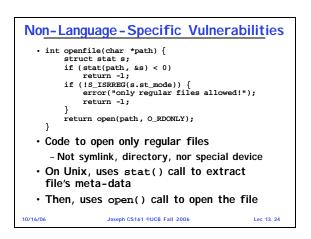
Another Vulnerability
<pre>• char buf[80];</pre>
<pre>void vulnerable() {</pre>
<pre>int len = read_int_from_network();</pre>
<pre>char *p = read_string_from_network();</pre>
if (len > sizeof buf) {
<pre>error("length too large, nice try!");</pre>
return;
}
<pre>memcpy(buf, p, len);</pre>
}
 What's wrong with this code?
 Hint - memcpy() prototype:
<pre>- void *memcpy(void *dest, const void *src, size_t n);</pre>
 Definition of size_t: typedef unsigned int size_t;
• Do you see it now? 10/16/06 Joseph CS161 @UCB Fail 2006 Lec 13.21







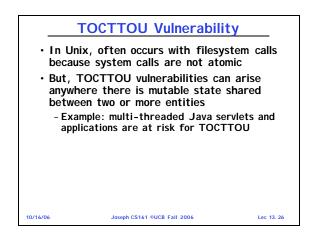




The Flaw?

- Code assumes FS is unchanged between stat() and open() calls - Never assume anything ...
- An attacker could change file referred to by path in between stat() and open()
 - From regular file to another kind
 - Bypasses the check in the code!
 - If check was a security check, attacker can subvert system security
- Time-Of-Check To Time-Of-Use (TOCTTOU) vulnerability
 - Meaning of path changed from time it is checked (stat()) and time it is used (open())

Joseph CS161 ©UCB Fall 2006



Many More Vulnerabilities... We've only scratched the surface! - These are the most prevalent examples

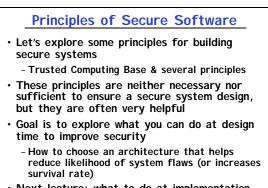
- · If it makes you just a bit more cautious about how you write code, good!
- In future lectures, we'll discuss how to prevent (or reduce the likelihood) of these kinds of flaws, and to improve the odds of surviving any flaws that do creep in

10/16/06

Joseph CS161 ©UCB Fall 2006

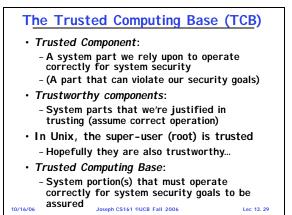
Lec 13.27

Lec 13.25



 Next lecture: what to do at implementation time 10/16/0 Lec 13.28

Joseph CS161 ©UCB Fall 2006



	TCB Definition
•	We rely on every component in TCB working correctly
•	Anything outside isn't relied upon - Can't defeat system's security goals even if it misbehaves or is malicious
•	TCB definition:
	 Must be large enough so that nothing outside the TCB can violate security

Lec 13. 30

Joseph CS161 ©UCB Fall 2006

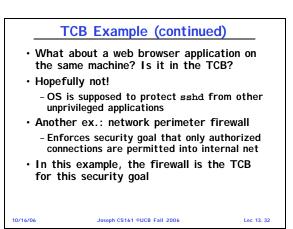
Page 5

10/16/06

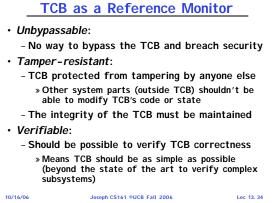
	TCB Example	
	rity goal: only authorized users ed to log into my system using S	SH
 What 	t is the TCB?	
	B includes SSH daemon (it makes thentication and authorization decision	ions)
ma car	sshd has a bug (buf overrun) or wa liciously reprogrammed (backdoor), n violate security goal by allowing authorized access	
	B also includes OS (can tamper wit hd's operation and address space)	h
	B also includes CPU (rely on it to ecute sshd correctly)	
10/16/06	Joseph CS161 ©UCB Fall 2006	Lec 13. 31

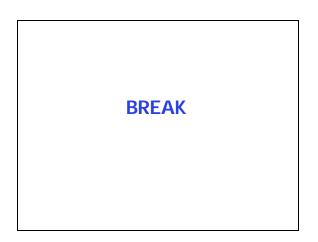


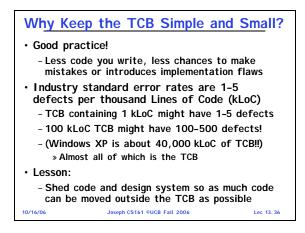
10/1



T	CB as Reference Monit	or	ТСВ	as a	
for er - Reca	e's always a mechanism responsible nforcing an access control policy call firewall lecture: this mechanism is a ference Monitor		- No way • Tamper-r	 Unbypassable: No way to byp Tamper-resista 	
goal o	ence monitor is the TCB for if ensuring access control poli			r systen to modif	
	eference monitor is just a TCB cialized for access control		- The inte	0 5	
refere - Unb	: three guiding principles for ence monitor ypassable, Tamper-resistant, a ifiable	ind	(beyo	be poss s TCB s nd the	
16/06	Joseph CS161 ©UCB Fall 2006	Lec 13. 33	SUDSY	stems)	







TCBs: What are They Good for?

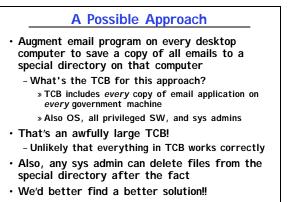
- Is the TCB concept just an esoteric idea?
 No, it is a very powerful and pragmatic idea
- TCB allows primitive, yet effective modularity • Separates system into two parts: securitycritical (TCB) and everything else
- Building secure and correct systems is hard!
 - More pieces makes security assurance harder
 - Only parts in TCB must be correct for system security -> focus efforts where they matter
 - Making TCB small gives us better odds of ending up with a secure system

10/16/06

Joseph CS161 ©UCB Fall 2006

Lec 13. 37

Lec 13.39



10/16/06

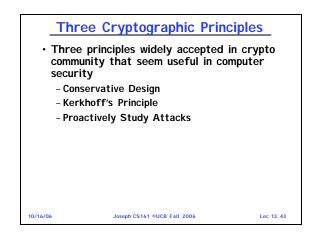
Joseph CS161 ©UCB Fall 2006

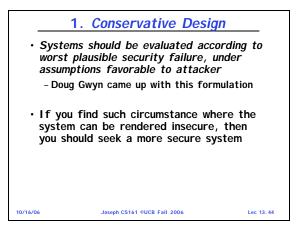
Another Approach Set up a high-speed networked printer - An email is "collected" when it is printed - Printer room is locked to prevent tampering - What's the TCB in this system? » TCB includes room's physical security » Also includes the printer Suppose we add a ratchet to paper spool so that it can only rotate forward - Don't need to trust the rest of the printer Wow! - TCB is only this ratchet, and room's physical security, nothing else! But, our approach uses a lot of paper! Joseph CS161 ©UCB Fall 2006 Lec 13.40 10/16/06

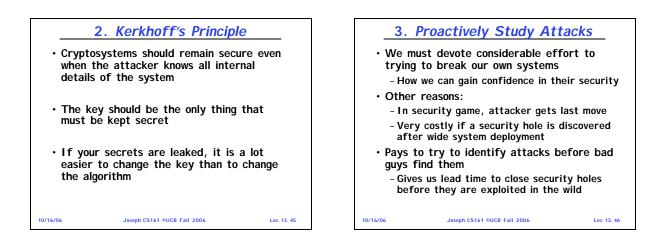
An All-Electronic Approach

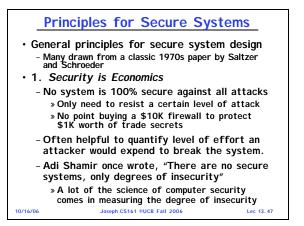
- Networked PC running special server SW
 Accepts email msgs and adds them its local FS
 - FS carefully implemented to provide write-once semantics: once a file is created, it can never be overwritten or deleted
 - Packet filter blocks all non-email connections
- What's in the TCB now?
 - Server PC/app/OS/FS, privileged apps on PC, packet FW, PC's sys admins, room's physical security, ...
- TCB is bigger than with a printer, but smaller than all machines approach's TCB
- I think you've earned your consulting fee

TCB Principles Summary
 Know what is in the TCB
 Design your system so that the TCB is clearly identifiable
 Try to make the TCB as unbypassable, tamper-resistant, and verifiable as possible
• Keep It Simple, Stupid (KISS)
 The simpler the TCB, the greater the chances you can get it right
 Decompose for security Choose a system decomposition/modularization based on simple/clear TCB
» Not just functionality or performance grounds
/16/06 Joseph CS161 ©UCB Fall 2006 Lec 13.42













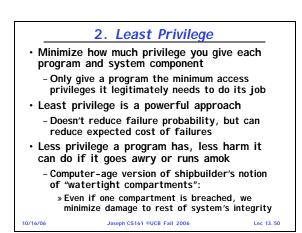
- Focus your energy on securing weakest links
 - Security is like a chain: it is only as secure as the weakest link
 - Attackers follow the path of least resistance, and will attack system at its weakest point
- No point in putting an expensive high-end deadbolt on a screen door
 - Attacker isn't going to bother trying to pick the lock when he can just rip out the screen and step through!

Lec 13.49

Lec 13.51

10/16/06

Joseph CS161 ©UCB Fall 2006



Principle of Least Privilege Examples
 Can help reduce damage caused by buffer overruns or other program vulnerabilities
 Intruder gains all the program's privileges
 Fewer privileges a program has, less harm done if it is compromised
 How is Unix in terms of least privilege?
- Answer: Pretty lousy!
- Programs gets all privileges of invoking users
 I edit a file and editor receives all my user account's privileges (read, modify, delete)
 Strictly speaking editor only needs access to file being edited to get job done

10/16/06

Joseph CS161 ©UCB Fall 2006