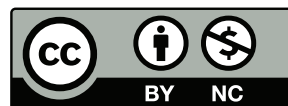


---

# Security Analysis I



---

# Attacking Things

- Why learn to attack?
- Is it true that “it takes a thief”? Do we have to learn to think like hackers to protect our systems?
- The answer, of course, is both “yes” and “no”

---

## Construction versus Destruction

- An architect and a demolition expert need to know about, say, the strength of steel beams
- The architect has to choose, balancing cost, esthetics, the customer's needs, and the local building codes
- The demolition engineer has to work with what he's got. There are choices, but they're very different ones

---

## Why Study Demolition in an Architecture Class?

- Unlike architects, someone is trying to knock our structures down *before* we're done with them
- We need some assurance that we've done a good job
- Part of the solution — repeat, *part* — is a “tiger team”

---

## Our Role

If we perceive our role aright, we then see more clearly the proper criterion for success: a toolmaker succeeds as, and only as, the users of his tools succeed with his aid. However shining the blade, however jeweled the hilt, however perfect the heft, a sword is tested only by cutting. The blacksmith is successful whose clients die of old age.

*The Mythical Man-Month*

Frederick P Brooks, Jr.

---

# Building Tiger Teams

- What are you trying to test?
- What are the weak spots you want probed?
- What are the constraints?

---

## Rules of Engagement

- Are there things you agree the tiger team can learn, and you'll just provide?
- Are physical or social engineering attacks allowed?
- What goals must the tiger team achieve?
- What protections do you offer the tiger team?

---

## Selecting a Tiger Team

- Whom should you hire?
- Is this a good spot for a former hacker who's gone into the security business?
- Let's put it like this: you're inviting that person to probe your defenses, with a guarantee that you won't prosecute
- Are you *certain* that they've gone straight?
- For some people, the answer is very clearly yes. In other cases, it's rather more doubtful



---

## When is Hacking Ethical?

- Under exactly one circumstance: when you have the permission of the person responsible for the system you're attacking
- This is not a fuzzy line; it is a very bright, sharp barrier
- Even then, there are unethical techniques, such as anything that risks the health or well-being of the hacker or the victim

---

## Different Goals

- Penetrate a running system
- Assess the strength of a particular program
- Let's look at the latter

---

## What's the mount Command?

- Attaches some device as a subtree at some point in the file system
- Example:

```
root@mikado: # ls /mnt
root@mikado: # mount /dev/sdb1 /mnt
root@mikado: # ls /mnt
BROTHER from-deirdre-lawful-hacking.pdf
mets.pdf Old
```

- That is a flash drive (/dev/sdb1) in a Linux box of mine

---

## Looking on Linux...

- `$ ls -l /bin/mount`  
`-rwsr-xr-x 1 root root 84232 May 24 2006 /bin/mount`
- Hmm — **mount** is setuid root. I wonder why...
- (The **mount** command is *not* setuid on Solaris or MacOS.)
- The man page reveals that sometimes, ordinary users can mount file systems
- Let's try it, with the real version and a non-setuid version

---

## Trying Mount

```
$ /bin/mount -t iso9660 /dev/cdrom /mnt/cdrom
```

```
mount: only root can do that
```

```
$ cp /bin/mount .
```

```
$ ./mount -t iso9660 /dev/cdrom /mnt
```

```
mount: must be superuser to use mount
```

---

## Note the Difference

- With the setuid version, there is a message from the application itself
- With the unprivileged version, the mount system call fails
- How can we find out what's happening?

---

# System Call Tracing

- Linux has a command **strace** that monitors a process' system calls
- (Solaris calls it **truss**; BSDs call it **ktrace**)

---

## Tracing `/bin/mount`

Leaving out the goo:

```
getuid32 () = 7994
geteuid32 () = 7994
lstat64 ("/etc/mtab", st_mode=S_IFREG|0644, st_size=1634, ...)
stat64 ("/sbin/mount.iso9660", 0xbfffa9a0) = -1 ENOENT (No such file or directory)
rt_sigprocmask (SIG_BLOCK, ~ [TRAP SEGV RTMIN], NULL, 8) = 0
mount ("/dev/cdrom", "/mnt/cdrom", "iso9660",
      MS_POSIXACL|MS_ACTIVE|MS_NOUSER|0xec0000, 0) = -1 EPERM (Operation not permitted)
rt_sigprocmask (SIG_UNBLOCK, ~ [TRAP SEGV RTMIN], NULL, 8) = 0
...
write (2, "mount: must be superuser to use ...")
```

Note: the message is the *unprivileged* version



---

## Why the Difference?

- Tracing a privileged program is a security breach
- The kernel silently disabled the setuid-ness — note the return values for `geteuid32 ()`
- But we can watch its complete behavior up to the point it needs privilege
- By the way — was the access control error (EPERM) logged?

---

## Tracing Library Calls

- The `ltrace` command traces library calls:

```
__strdup(0xbff2abb3, 4095, 0xb7d0a000, 373, 0) = 0x9c3cd28  
__strdup(0xbff2aba0, 4095, 0xb7d0a000, 373, 0) = 0x9c3cd38  
sprintf("/sbin/mount.iso9660", "/sbin/mount.%s", "iso9660") = 19
```

- Sprintf? Hmm – can we exploit it?
- Probably not — the source shows a `strlen()`
- Sure enough — for a very long file type, it skips the `sprintf`

---

## What Else Can We Learn?

- Run **strings** on the command

```
$ strings mount | more
/lib/ld-linux.so.2
libc.so.6
putchar
strcpy
waitpid
ioctl
chown
...
```

- That looks like a symbol table

---

## The Option Table

loop=  
vfs=  
offset=  
encryption=  
keybits=  
nohashpass  
speed=  
comment=  
uhelper=

What does the “nohashpass” option do? It’s not documented...

---

## Other Interesting Stuff

- Look for file names
- Anything in /tmp? Anything *predictable* in /tmp (Why is that interesting?)

---

## More Avenues to Explore

Does the program manipulate uids?

```
$ strings mount | grep uid
```

```
geteuid
```

```
getuid
```

```
getpwuid
```

```
setuid
```

```
nosuid
```

```
uid=useruid
```

```
uid=%d
```

```
uid=
```

```
or by label, using -L label or by uuid, using -U uuid .
```

```
broken_suid
```

---

## What Other Ways Do We Know to Break In?

- What files does it read or write? (`string | grep /`)
- Does it exec anything it shouldn't?
- What UID/GID does it run under? (`ps` can tell you that)

---

## Finding Holes isn't Following a Recipe

- A script can let you exploit a known hole
- It won't help you find a new one
- Look for edge cases, push boundaries
- It's often trial and error



---

## Further Steps

- We could use a disassembler, but that takes a lot of skill
- That said, it is possible, and people do it
- Run the program under `gdb` and trace it. Again, it takes time and skill

---

# Fuzzing

- A way to find input-parsing bugs by randomly or systematically modifying input streams
- Can be random (no knowledge of input formats), smart (handles input formats, checksums, etc.), black box (smart, but with no validation of code coverage), or white box (systematically test different code paths)
- Input data can be generated automatically or by mutating valid inputs
- Extremely powerful technique, used by testers and attackers

---

## Why Does Fuzzing Work?

- It exercises seldom-tested code paths
- It pushes boundary conditions
- Note: tester must use other tools to look for memory leaks, deadlocks, code coverage, etc.

---

## Conclusions

- Programs leak a lot of data while executing
- I can still run `strace` — but not `ltrace` — even if the executable is read-protected
- Such monitoring gives valuable clues as to its behavior
- (Other operating systems have similar facilities)

---

## Higher-Level Tiger Teams

- Network attacks
- Social attacks
- Physical attacks
- Infiltration

---

## Infiltration

- Get a job at the company of your choice
- Get a job at a supplier to the company of your choice
- Make it a nice, low-level job
- Do employers really check references and credentials? Many don't

---

## Physical Attacks

- Physically break in to the building
- Or break into the computer room
- Have you checked your false ceilings lately?

---

**From the NY Times, 11/30/2005**

## **Men Posing as FedEx Workers Rob Jewelry Wholesaler**

Two men posing as Federal Express employees breezed through security at a tightly guarded building in the city's diamond district yesterday, rode the elevator to the sixth floor and robbed a jewelry wholesaler of goods valued at \$4 million, the police said.

...

According to Commissioner Raymond W. Kelly, one of the robbers carried a package that was too big to slide under the door at Doppelt & Greenwald. The robbers said it had to be signed for, the commissioner said, and when the door was opened, they showed the gun.



---

## Social Attacks

- Talk people into giving you what you want
- If you sound convincing, you can be remarkably successful
- If you look the part, you can be remarkably successful
- (Do I classify that robbery as a physical attack or a social one?)

---

# Network Attacks

- Find vulnerable host
- Find a vulnerable application
- Walk right in. . .
- (More a subject for 4180)

---

# Reconnaissance

- All of these attack techniques depend on *reconnaissance*
- The attacker has to know the weak points
- Blocking, or at least spotting, reconnaissance is a major defense

---

## Blocking Reconnaissance

- Intrusion detection systems can spot probes
- Sophisticated variant: low, slow, distributed reconnaissance — probe from multiple points
- Must do the same in the physical world — it's a virtual certainty that those robbers had watched how FedEx workers were treated at the entrance

---

## Hiding Version Numbers Doesn't Help Much

```
# nmap -sV -O -p 1-1024 bigboy
```

```
Starting Nmap 4.20 ( http://insecure.org ) at 2007-04-16 22:02 EDT
Interesting ports on bigboy.machshav.com (192.168.2.79):
Not shown: 1020 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 4.4 (NetBSD 20061114; protocol 2.0)
111/tcp   open  rpcbind  2-4 (rpc #100000)
631/tcp   open  ipp      CUPS 1.2
1022/tcp  open  rpc
MAC Address: 00:11:11:5B:7A:CD (Intel)
Device type: general purpose
Running: NetBSD
OS details: NetBSD 4.99.4 (x86)
Network Distance: 1 hop
Service Info: OS: NetBSD
```

---

## Scanning a CLIC Machine

```
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.9 - 2.6.12 (x86)
Uptime: 8.469 days (since Sun Apr  8 11:16:07 2007)
Network Distance: 1 hop
Service Info: OS: Unix
```

**matches the local reality:**

```
$ uptime
 22:33:00 up 8 days, 11:21,  7 users,  load average: 0.42, 0.
$ uname -a
Linux kathmandu.clic.cs.columbia.edu 2.6.9-42.0.8.EL #1 Tue J
```

---

## Process Helps

- Make sure people *a/ways* follow process
- Always check badges, credentials, etc.
- *Never* give out extra information, no matter the circumstance

---

## Establish Technical Processes

- How are system changes made?
- Are they logged?
- Who can change your code base?
- Version control systems — CVS, subversion, others — are a major help; they let you know who made what changes, when, and why



---

## Don't Just Go Through the Motions

- It just annoys people, and doesn't protect you
- (Example: demanding a picture ID when nothing is done with the information.)

---

## A Good Example

99 Park Ave.

VISITOR

11/26/07

12:14:51



STEVEN BELLOVIN

estimation /16

They took my picture, and typed in my name as on my driver's license. This is much better than asking for an ID and doing nothing, or simply logging the name and not comparing my face to the picture on the license. (But note the typographical error in my name...) Also — do they *enforce* anything limiting me to the 16th floor?

But what about privacy?

---

## Practice, Practice, Practice

- The only way to avoid complacency is to practice
- People won't listen to lectures and training
- Make sure the culture and the incentives are set up properly
- Reward people for following policy, even if it inconveniences senior management