ECE 598 – Advanced Operating Systems
Lecture 22

Vince Weaver

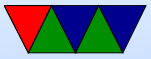
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

19 April 2018

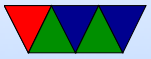
Announcements

- HW#10 Still Delayed
- Still need project volunteers for Thursday
- Midterm on May 1st



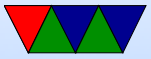
HW#8 Review

- Was hoping for interesting timestamps, just 1/2/3 a little boring. Could change the date as well
- Bug in df? Was not multiplying by 100 so always returning 0?



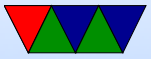
HW#9 Review

- Text printing, went OK. Fore color / back color not black/white. If made black background transparent the gradient would shine through.
- Gradient. Hoping for continuous across. Tricky. Need to clarify difference between horizontal and vertical
- Two groups did broken heart something cool, but different
- Note on VGA fonts, hard to find, made my own.
- Pain of adding new syscall, update at all locations



In reality, just as bad on real OS like Linux.
Especially big flamewars about including kernel headers in user programs (copy or direct link in /usr/include)
Also on Linux, how do you had new syscall number? What if two people try to add a new one at same time?

- Alignment, mailbox specification. Mostly so can cheat and use bottom bits for code
  True often when you mess with alignment it's for performance, but not in this case.
- Why slow?
  Framebuffer memory uncached



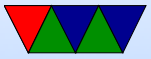
Local copy is cached (before screen) that's why it speeds up once caches are enabled

We update full screen on each write() even if only 1 byte, no buffering. Things like "ls" which print a byte at a time really noticeable

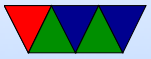
It is updating screen in interrupt handler, so no interrupts happen (look to see the LED slow down)

- What could we do?

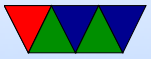
  One good one is to have a "screen dirty" flag and only update maybe once every 64Hz or so, not in IRQ handler but a kernel thread. Loop unrolling, writing in asm? Yes,



but Amdahl's law.

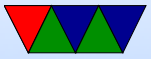


OS Security



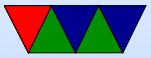
Computer Security

- Most effective security is being unconnected from the world and locked away in a box
- Modern systems are increasingly connected to networks, etc.
- Internet of Things



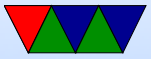
The Problem

- Untrusted inputs from user can be hostile.
- Users with physical access can bypass most software security.



What can an attacker gain?

- Fun / Mischief
- Profit
- A network of servers that can be used for illicit purposes (SPAM, Warez, DDOS)
- Spying on others (companies, governments, etc)

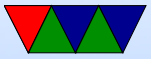


Sources of Attack

- Untrusted user input
  Web page forms
  Keyboard Input

- USB Keys (CD-ROMs)
  Autorun/Autostart on Windows
  Scatter usb keys around parking lot, helpful people plug into machine.

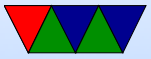
- Network



cellphone modems
Ethernet/internet
wireless/bluetooth

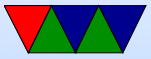
- Backdoors
  Debugging or Malicious, left in place

- Brute Force – trying all possible usernames/passwords

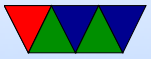


Types of Compromise

- DoS (Denial of Service)
  Fork bombs, etc.
- Crash (extreme form of DoS)
  “ping of death”
- User account compromise
- Root account compromise
- Privilege Escalation

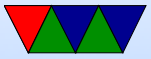


- Rootkit
- Re-write firmware? VM? Above OS?

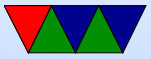


Trojan Horse and others

- Create program in home directory called "ls" that gives you root permission, then runs actual ls. Convince someone with root to cd to your dir and run ls. This is why "." is not in the search path by default.
- Fake login screen, get password (sort of like phishing)
- Backdoors
- Virus that can infect executables. How can you detect this? Heuristics? Read-only-fs? offline list of checksums?

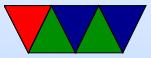


- Worm – famously the Morris worm in the 1980s



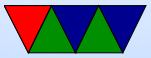
Unsanitized Inputs

- Using values from users directly can be a problem if passed directly to another process
- SQL injection attacks; escape characters can turn a command into two, letting user execute arbitrary SQL commands; xkcd `Robert '); DROP TABLE Students;--`
- If data (say from a web-form) directly passed to a UNIX shell script, then by including characters like ; can issue arbitrary commands



Buffer Overflows – Big Drawback of C

- User (accidentally or on purpose) copies too much data into a fixed sized buffer.
- Data outside expected area gets over-written. This can cause a crash (best case) or if user carefully constructs code, can lead to user taking over program.

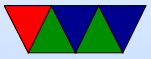


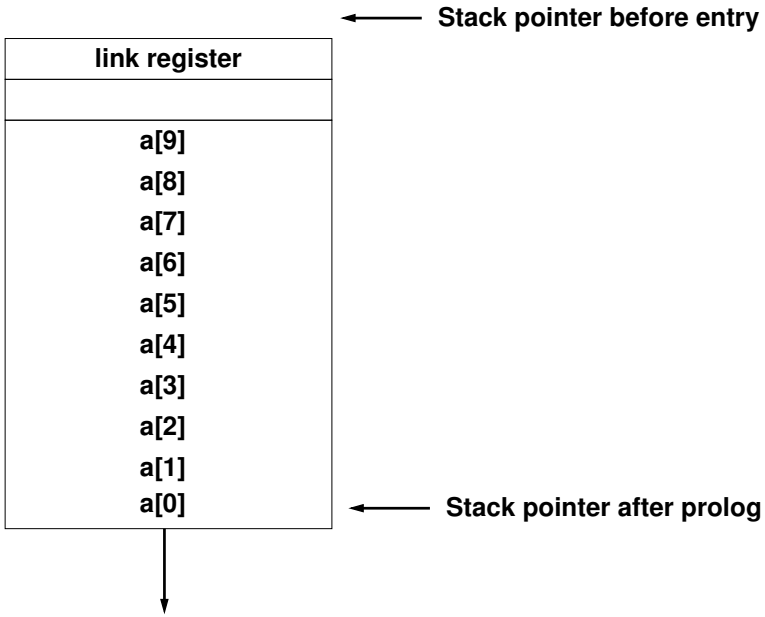
Buffer Overflow Example

```
void function(int *values, int size) {
    int a[10];

    memcpy(a,values,size);

    return;
}
```

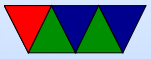
Maps to

```
    push    {lr}
    sub sp,#44

    memcpy

    add sp,#44
    pop {pc}
```



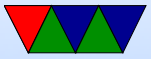


A value written to a[11] overwrites the saved link register. If you can put a pointer to a function of your choice there you can hijack the code execution, as it will be jumped to at function exit.



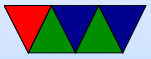
Mitigating Buffer Overflows

- Extra Bounds Checking / High-level Language (not C)
- Address Space Layout Randomization
- Putting lots of 0s in code (if strcpy is causing the problem)
- Running in a "sandbox"



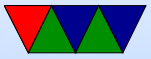
Dangling Pointer / Null Pointer Dereference

- Typically a NULL pointer access generates a segfault
- If an un-initialized function pointer points there, and gets called, it will crash. But until recently Linux allowed users to `mmap()` code there, allowing exploits.
- Other dangling pointers (pointers to invalid addresses) can also cause problems. Both writes and executions can cause problems if the address pointed to can be mapped.



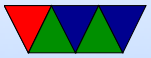
Privilege Escalation

- If you can get kernel or super-user (root) code to jump to your code, then you can raise privileges and have a "root exploit"
- If a kernel has a buffer-overrun or other type of error and branches to code you control, all bets are off. You can have what is called "shell code" generate a root shell.



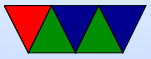
Setuid Programs

- Some binaries are setuid.
- They run with root privilege but drop them.
- If you can make them run your code before dropping privilege you can also have a root exploit.
- Tools such as ping (requires root to open raw socket), X11 (needs root to access graphics cards), web-server (needs root to open port 80).
- Ways to solve?
  - Careful coding



- Capabilities (fine grained permissions)
- Can you give an exec permission *only* to open port80?

  Problem is clever hackers are good at taking one capability and escalating to full.



File Creation Races

- Many programs open temporary files in /tmp and write to them
- If you can guess the name of a /tmp file a program will open, you can create a link or otherwise redirect it.
- Manage to create a link to /etc/shadow or similar