ECE 471 – Embedded Systems
Lecture 25

Vince Weaver

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

vincent.weaver@maine.edu

13 November 2019

Announcements

- Still behind on grading homeworks
- And midterms
- Did reply with project topics
- Demosplash results
- HW#9 will be posted, due a week from Friday

HW#9 Info

- Read temperature probe, print temp on i2c display
- Re-use code from past assignments
- Follow spec on functions to put code in, how to print results
- Testing: 4 cases described, and set up to allow unit tests

Buffer Overflows

- User (accidentally or on purpose) copies too much data into a fixed sized buffer.

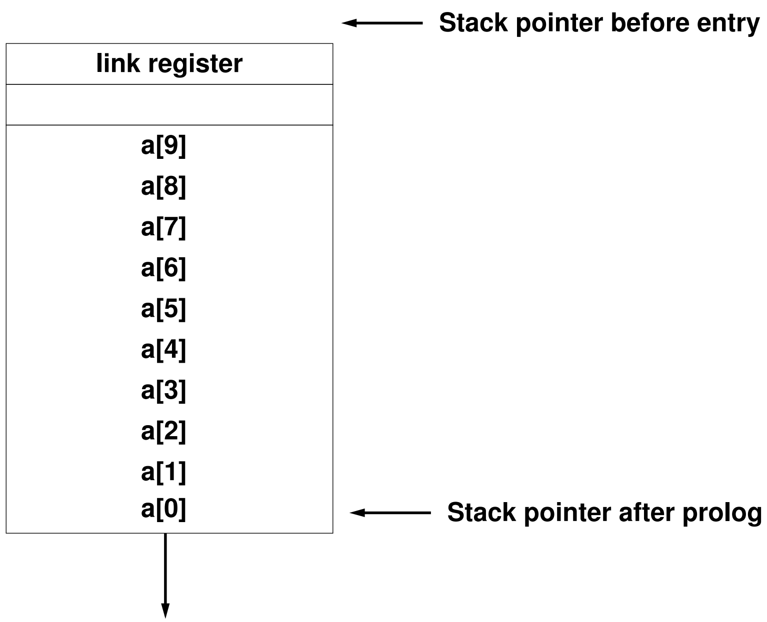
- Data outside expected area gets over-written. This can cause a crash (best case) or if user carefully constructs code, can lead to user taking over program.

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.

- 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).

Information Leakage

- Can leak info through side-channels
- Detect encryption key by how long other processes take? Power supply fluctuations? RF noise?
- Timing attacks
- Meltdown and Spectre

Finding Bugs

- Source code inspection
- Watching mailing lists
- Static checkers (coverity, sparse)
- Dynamic checkers (Valgrind). Can be slow.
- Fuzzing

Computer Security

Social Engineering

- Often easier than actual hacking
- Talking your way into a system
- Looking like you know what you are doing
- "The Art of Deception"

Worrisome embedded systems

- Backdoors in routers.
- Voting Machines, ATMs
- pacemakers
- Rooting phones
- Rooting video games
- Others?

Voting Machines

- Maine has paper ballot — not too bad
- Often are old and not tested well (Windows XP, only used once a year)
- How do researchers get them to test? e-bay?
- USB ports and such exposed, private physical access
- Can you trust the software? What if notices it is Election Day and only then flips 1/10th the vote from Party A to Party B. Would anyone notice? What if you have source code?

- What if the OS does it. What if Windows had code that on Election Day looked for a radio button for Party A and silently changed it to Party B when pressed?
- OK you have and audit the source code. What about the compiler? (Reflections on Trusting Trust). What about the compiler that compiled the compiler?
- And of course the hardware, but that's slightly harder to implement but a lot harder to audit.

Examples – CANbus

- 2010 IEEE Symposium on Security and Privacy. *Experimental Security Analysis of a Modern Automobile* U of Washington and UCSD.
- Fuzzing/ARM/CANbus
- can control brakes (on / off suddenly)
- heating, cooling, lights, instrument panel
- windows/locks Why? fewer wires if on a bus then direct-wired
- electronic stability control, antilock, need info from each

wheel

- roll stability control (affect braking, turning to avoid rollover)
- cruise control
- pre-crash detection (tighten seatbelts, charge brakes)
- while it might be nice to have separate busses for important and unimportant, in practice they are bridged
- Locks– monitor buttons, also remote keyfob... but also disengage if airbag deploys
- OnStar – remotely monitor car, even remotely stop it (in case of theft) over wireless modem

- Access? OBD-II port, also wireless
- 2009 car
- cars after 2008 required to have canbus?
- Problems with CAN
  - Broadcast... any device can send packets to any other
  - Priority.. devices set own priority, can monopolize bus
  - No authentication... any device can control any other
  - Challenge-response. Cars are supposed to block attempts to re-flash or enter debug mode without auth. But, mostly 16-bits, and required to allow a try every 10s, so can brute force in a week.

  - If you can re-flash firmware you can control even w/o ongoing access
- Not supposed to disable CAN or reflash firmware while car moving, but on the cars tested they could.
- Probing – packet sniffing, fuzzing (easier as packet sizes small)
- experiments – on jackstands or closed course
- controlled radio – display, sounds, chimes
- Instrument panel – set arbitrary speed, rpm, fuel, odometer, etc
- Body control – could lock/unlock (jam by holding down

lock), pop trunk, blow horn, wipers on, lights off

- Engine... mess with timing. forge "airbag deployed" to stop engine
- Brakes.. managed to lock brakes so bad even reboot and battery removal not fix, had to fuzz to find antidote
- can over-ride started switch. wired-or
- test on airport. cord to yank laptop out of ODB-II
- fancy attacks. Have speedometer read too high. Disable lights. "self-destruct" w countdown on dash, horn beeping as got closer, then engine disable.

Stuxnet

- SCADA – supervisory control and data acquisition
- industrial control system
- STUXNET.. targets windows machines, but only activates if Siemens SCADA software installed. four zero-day vulnerabilities
  USB flash drives
  signed with stolen certificates

- Interesting as this was a professional job. Possibly by US/Israel targeting very specific range of centrifuges reportedly used by Iran nuclear program. While reporting "everything OK" the software then spun fast then slow enough to ruin equipment.

Examples – JTag/hard-disk

- JTAG/Hard-disk takeover
- `http://spritesmods.com/?art=hddhack&page=8`
- Find JTAG
- 3 cores on hard-disk board, all ARM. One unused.
- Install custom Linux on third core. Then have it do things like intercept reads and change data that is read.

Places for More Info

- Embedded projects: `http://hackaday.com`
  They had a recent series on CAN-bus

- Computer Risks and Security Issues: The RISKS digest from comp.risks
  `http://www.risks.org`

Software Bugs

- Not all bugs are security issues
- Coding bugs can have disastrous effects

Automotive

- Until recently no standard
- Bugs, Toyota firmware
- `http://www.edn.com/design/automotive/4423428/2/Toyota-s-killer-firmware--Bad-design-and-its-conse`

Airplanes

- DO-178B / DO-178C
- Software Considerations in Airborne Systems and Equipment Certification
  - Catastrophic: fatalities, loss of plane
  - Hazardous: negative safety, serious/fatal injuries
  - Major: reduce safety, inconvenience or minor injuries
  - Minor: slightly reduce safety, mild inconvenience
  - No Effect: no safety or workload impact

- AA Flight 965. Autopilot to waypoint R. Re-entered it, two starting with R, so it helpfully picked one with highest frequency, did a semi-circle turn to east right into a mountain.
- Air France Flight 447, reliance on autopilot

Military

- Patriot missile – clock drift slightly, but when on for hundreds of hours enough to affect missile tracking
- Yorktown smart ship – 1997 – Running Windows NT. Someone entered 0 in a field, divide by 0 error, crashed the ship. Database crash, crashed propulsion system. Rumors that it needed to be towed in, but no, only down for 2.75 hours.
- F-22s computers crashed when crossing 180 degrees longitude? Lost navigation and communication, had to

follow tankers back to Hawaii.