18732: Secure Software Systems

Course Overview

Lujo Bauer

Spring 2015
Plan for Today

• Course logistics
• Software security in the news
• Overview of course topics
Pittsburgh & Silicon Valley

• Silicon Valley microphones are **always** live
  – Don’t put pen, paper, bag near microphones

• Pittsburgh: Raise your hand before speaking!
  – Touch **and hold** base of microphone to ask questions
  – Only two microphones can be live at once(?)
Course Staff

• Instructor: Lujo Bauer
  – Office: CIC 2203
  – Email: lbauer@cmu.edu

• TAs: Jassim Aljuraidin, Yannis Mallios, Blase Ur

• Best way to get in touch: Piazza
Office Hours

• Tue 9-10am EST in CIC 2203 (Lujo)
• Tue 10-11:15am EST in CIC 2206 (Jassim)
• Wed 2-3pm in CIC 2214 (Blase)
• Wed 5:30-6:30pm EST / 2:30-3:30pm PST on BlueJeans (Blase)
• Fri 10:30-11:30am in CIC 2206 (Yannis)
Logistics

- Location: GCH 4307
- Days: Tue & Thu
- Time: 1:30-2:50pm

- Web page: [http://www.ece.cmu.edu/~ece732/](http://www.ece.cmu.edu/~ece732/)
  - Schedule, subject to change
  - Exam dates are fixed
- Blackboard
Logistics (2)

• Discussion board:
  – https://piazza.com/cmu/spring2015/18732
  – Don’t sign up manually; we’ll add you

• Use Piazza for any questions
  – General or specific, to class or instructors
  – Use appropriate visibility settings

• Answer questions, edit existing answers
  – Instructors will label correct answers

• Easy way to get class participation credit!
Grading

• Exams: 40% (20% + 20%)
  – Three in-class, closed book exams
  – Only two highest grades count!

• Class participation: 10%

• Course assignments: 50% (5x 10%)
  – Next slide

• Re-grading policy
  – Grade could go up or down
Scribes

• 3 students per lecture to take notes
• Focus on what’s **not** on the slides
  – Questions asked during lecture and answers
  – Deviations from slide content
  – Key insights needed to understand a slide
• Summarize questions and answers on Piazza
  – **One summary per lecture**
• Submit summary within 24 hours of lecture
• Counts towards class participation grade!
• *Online sign-up sheet will be posted soon*
Assignments

Assignments = projects = homeworks
- Teams of 2 (form team by end of week)
- All team members receive the same grade

1. Implementing buffer overflow attacks
2. Implementing web attacks
3. Analyzing software for security vulnerabilities using off-the-shelf commercial tools
4. Building software safe by construction using a program verifier
5. Building an interpreter that implements dynamic taint tracking
Assignment Logistics

• Grading using Autolab
  – https://autolab.cs.cmu.edu/18732-s15/

• Grading will be automated or semi-automated
  – Can turn in early and multiple times with no penalty

• Lateness policy:
  – 10 overall late days
  – 4 max per project
  – No additional extensions – use late days wisely!
Reading

• No textbook for the course
• Slides will be posted after lectures
• Research papers or book chapters on which lectures are based
  – Posted on web site or on Blackboard
Prerequisites

• Familiarity with C is required

• An introductory course in computer security such as 18-487 or 18-730 is recommended, but not required

• Undergraduate courses in OS, Compilers would be a bonus

• Quick class poll
Cheating

http://www.cmu.edu/policies/documents/Cheating.html

Some highlights:

• Cheating includes submitting solutions that are not your own, or helping another student to do so (e.g., by sharing your solutions with them)

• Examples of cheating
  – Looking up the answers on the web or elsewhere (even with citation)
  – Formulating solutions in a group, except for a class-sanctioned group
  – Allowing others to see your solutions (including “by accident”)
Questions about logistics?
Software Bugs in the News
Software Bugs in the News

Unmanned European rocket explodes on first flight

Europe's newest unmanned satellite-launching rocket, the Ariane 5, intentionally was blown up Tuesday just seconds after taking off on its maiden flight. …


… The internal SRI software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer. This resulted in an Operand Error. The data conversion instructions (in Ada code) were not protected from causing an Operand Error, although other conversions of comparable variables in the same place in the code were protected. …

Software Bugs in the News

... A previously-unknown **software flaw** in a widely-deployed General Electric energy management system contributed to the **devastating scope of the August 14th northeastern U.S. blackout** ...  

[Security Focus, Feb 11 2004]

The Northeast Blackout of August 2003, the largest in North American history, shut down 62,000 MW of generation capacity, and **cost businesses an estimated $13 billion** in productivity. ...

[IEEE-USA Today’s Engineer, Feb 2005]

... “There was a couple of **processes that were in contention for a common data structure**, and through a software coding error in one of the application processes, they were both able to get write access to a data structure at the same time ... And that corruption led to the alarm event application getting into an infinite loop and spinning.” ...

[Security Focus, Apr 7 2004]
Software Bugs in the News

1 bug = $13 billion in losses
= 260,000 years of CMU tuition
= MS degrees for 130,000 people
E-voting vendor: Programming errors caused dropped votes

... E-voting machines from Premier Election Solutions, formerly called Diebold Election Systems, **dropped hundreds of votes** in 11 Ohio counties during the primary election, as the machine's memory cards uploaded to vote-counting servers. ...

[Network World, Aug 22 2008]
... Software bugs in a Soviet early-warning monitoring system nearly brought on nuclear war in 1983, according to news reports in early 1999. The software was supposed to filter out false missile detections caused by Soviet satellites picking up sunlight reflections off cloud-tops, but failed to do so. Disaster was averted when a Soviet commander, based on a what he said was a ‘…funny feeling in my gut’, decided the apparent missile attack was a false alarm. The filtering software code was rewritten. ...

Software Bugs in the News

- Accidents
- Monetary loss
- Effect on political process?
- Military conflict?

- But is it a computer security problem?
Malware implicated in fatal Spanair plane crash

Computer monitoring system was infected with Trojan horse, authorities say

Authorities investigating the 2008 crash of Spanair flight 5022 have discovered a central computer system used to monitor technical problems in the aircraft was infected with malware. … infected computer failed to detect three technical problems with the aircraft … Flight 5022 crashed just after takeoff from Madrid-Barajas International Airport two years ago today, killing 154 and leaving only 18 survivors.

Software Bugs in the News

• Accidents
• Monetary loss
• Effect on political process?
• Military conflict?

• But is it a computer security problem?
• Computer security = protecting computers against misuse and interference
• Bugs can be (and are) purposefully exploited
Exploiting Bugs as a Nuisance

• MyDoom (2004) - $38.5 billion
• SoBig (2003) - $37.1 billion
• Love Bug (2000) - $15 billion
• Code Red (2001) - $2 billion
Exploiting Bugs for Profit

• Hacker convicted of breaking into a business’ computer system, stealing confidential information and threatening disclosure if $200,000 not paid

• 11 people indicted for stealing more than 40 million credit card and debit card numbers
  [ CNN, Aug 2008 ]
The Economics of Botnets

Money stream:

- Create a botnet
- Rent a botnet
- Sell a botnet

- DDoS
- Spammers
- Installing Adware and Malware
- Stealing confidential information
- Phishing
- Spamdexing

- Selling bank accounts and credit card information
- Selling Internet services and shops accounts
- Selling identity information

[ Y. Namestnikov. The economics of botnets. Kaspersky Lab, 2009. ]
Pricelists

- $100-180 per 1000 installs (2011)
- $1-1,500 stolen bank account details (2009)
- $20-100+ US credit card (2013)
- $5-8 US citizen personal data (2009)
- $7-15 user accounts for paid online services (2009)
- $1000-2000 per month for botnet spam services (2009)
- $50-$$ per day for botnet DDoS services (2009)
- $125,000 for zero-day browser exploit to private party (2012)
- $250,000 for zero-day iOS exploit to government (2012)

Sources:
ISC BIND CVE-2014-8500 Remote Denial of Service Vulnerability
2015-01-12
http://www.securityfocus.com/bid/71590

DenyHosts 'regex.py' Remote Denial of Service Vulnerability
2015-01-12
http://www.securityfocus.com/bid/64478

Linux Kernel CVE-2010-5313 Local Denial of Service Vulnerability
2015-01-12
http://www.securityfocus.com/bid/71363

Linux Kernel 'espfix64' Local Denial of Service Vulnerability
2015-01-12
http://www.securityfocus.com/bid/71250

Linux Kernel CVE-2014-7841 SCTP NULL Pointer Dereference Denial of Service Vulnerability
2015-01-12
http://www.securityfocus.com/bid/71081

Linux Kernel 'trace_syscalls.c' Multiple Local Denial of Service Vulnerabilities
2015-01-12
http://www.securityfocus.com/bid/70971

Linux Kernel KVM CVE-2014-7842 Local Denial of Service Vulnerability
2015-01-12
http://www.securityfocus.com/bid/71078

Linux Kernel 'trace_syscalls.c' Multiple Local Denial of Service Vulnerabilities
2015-01-12
http://www.securityfocus.com/bid/70972

174 vulnerabilities reported yesterday!
In software we use regularly:
Linux Kernel, Corel, PHPKIT, Tapatalk, WordPress plugins, libpng, Jboss, OpenSSL, Drupal, WebKit, NTP, Oracle Java,…
Why Software Security Is Important

• Impact of attacks is wide-ranging, from monetary loss to physical harm

• There are lots of software bugs that can be and are being exploited by attackers

• Huge numbers of bugs are continually discovered …

… in software that we rely on in almost every aspect of our lives
10,000-foot View

SO WE’LL CALL THIS API, THEN—WAIT WAIT WAIT. THIS IS TOO LOW-LEVEL.

I NEED CONTEXT. THE 10,000-FOOT VIEW.

OK.

SO WE HAVE TWO APPLICATIONS THAT—NO. NO. PULL WAY BACK. THE REALLY HIGH-LEVEL VIEW.

IT ALL BEGAN FOUR BILLION YEARS AGO ON A PLANET CALLED “EARTH.” OK, THAT’S HIGH ENOUGH.
Course Goals

1. Understand current software attacks and defenses
2. Understand general principles of secure software system design, implementation, and analysis
Topics

0. Attacks
1. Software Security Architectures
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement
Buffer Overflow Attacks

- One cause: Array abstraction not protected in C
  - Suppose array is 128 bytes
  - Possible to write 200 bytes into the array and overwrite return address of function

Program P: `exec( "\'/bin/sh" )`
Questions for the class

• What are some possible defenses?
  – Run-time Enforcement
  – Security Analysis of Software
  – Language-based Security

• What general security property is violated?

  Control Flow Integrity
Control Hijacking Attacks

• Attacker causes arbitrary attack code (e.g. root shell) to execute on target machine by hijacking the control flow of an application program

• Examples
  – Buffer overflow attacks
  – Integer overflow attacks
  – Format string vulnerabilities
A Reflected XSS Attack

1. visit web site
2. receive malicious page
3. click on link
echo user input
4. send valuable data

Questions for the class

• What are some possible defenses?
  – Run-time enforcement

• What general security property is violated?
  Information Flow
Web Attacks

Web-based attacks (Top 3 web site vulnerabilities)

1. Cross-site scripting (XSS)
2. Cross-site request forgery
3. SQL injection
Learning Outcome

• Understanding of the state-of-the-art in
  – control hijacking attacks and associated defenses
  – web-based attacks and associated defenses

• Note: These two classes of attacks currently top the charts of common security vulnerabilities databases
Topics

0. Attacks
1. Software Security Architectures
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement
Isolation (is good for security)

- Virtual machine-style architecture aims to provide *isolation* between multiple VM’s running on the same physical machine.

- Will cover a number of related approaches and their implications for security
  - Java VM, Web security, Android security
How do we trust remote servers?
Attestation to Code Integrity

I'll ask what programs they have run!

How do I know if I can trust that server?

Hey server!

What code have you run?

Since I last rebooted, I’ve run:

0: Operating System
1: Web Server
2: Shopping Cart Server

I trust those programs to protect my data!
Learning Outcome

• Systematic thinking about system security
  – What is the security mechanism?
  – What is the adversary model?
  – What is the desired security property?
  – Does system guarantee security property in the presence of adversaries?
  – What is the trusted computing base (TCB)? What are our assumptions about the TCB?

• Understanding of specific security architectures for isolation and attestation
Topics

0. Attacks
1. Software Security Architectures
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement
Security Analysis of Software

#include <stdio.h>

int main(int argc, char *argv[]) {
    /* the number of arguments in argv */
    /* of null-terminated strings also known */
    /* as an array of arrays of char. It */
    /* holds the contents of the command line */
    int loopcount;
    /* NOTE: The name with which the program */
    /* was invoked is held in argv[0]. */
    for (loopcount=1; loopcount < argc; loopcount++) {
        switch (argv[loopcount][0]) {
        case '-':
            printf("OPTION \n",
            argv[loopcount][1]);
            break;
        default:
            printf("FILENAME \n",
            argv[loopcount]);
            break;
        }
    }
}

Program

Vulnerability description

Analytical engine

priv

unpriv

setuid(getuid())

exec()
Learning Outcome

• Understanding of
  – general methods of software analysis (model checking, static analysis, dynamic analysis)
  – their application to identifying security vulnerabilities in legacy code (buffer overflows, race conditions, setuid, chroot, malware) and for verifying higher-level security properties (e.g., memory isolation)

• Understanding of strengths and limitations of various methods

• Hands-on experience with a state-of-the-art industrial static analysis tool – Coverity Code Advisor
  
• Hands-on experience with a state-of-the-art program verifier (and language) – Dafny


Assignment 3
Assignment 4
Topics

0. Attacks
1. Software Security Architectures
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement
Security Typed Languages

• Type safe programming languages
  – Think Java (as opposed to C)
  – Better protection for language abstractions such as arrays
    • No buffer overflow attacks on Java programs

• Security-typed programming languages
  – Language designed so that a program that compiles correctly does not incur security violation at run-time
What Is a Type System?

• Jif Example

```
int {alice\rightarrow bob, alice; bob\leftarrow alice} y;
int {bob\rightarrow bob} x;
int {alice\rightarrow bob; bob\rightarrow alice} z;
if (x == 0)
z = y;
```

"Now! That should clear up a few things around here!"

Secrecy violation: Information about x is leaked to Alice

• Well-typed program won’t have security violations
• Type checking done by compiler
Learning Outcome

• Understanding of basic concepts of type systems for programming languages: syntax, static and dynamic semantics, type safety

• Understanding the security property of non-interference (an information flow property) and how type soundness can imply non-interference

• Understanding of type systems to improve security of mobile (PCC) and assembly code (TAL)

• Hands-on experience with building a secure interpreter that dynamically tracks information flows
Topics

0. Attacks
1. Software Security Architectures
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement
Run-time Security Enforcement

- Enforce security using mechanisms that monitor systems as they execute (as opposed to static analysis of code)

- Some examples
  - *Stackguard*: Dynamic checks to prevent buffer overflows
  - *Taint Analysis*: Information flow control, malware detection
  - Stack inspection, firewalls, applet sandboxing, displaying security warnings, …
Learning Outcome

• Understanding of fundamentals of run-time enforcement of security properties
  – What class of security policies can be enforced using run-time monitoring?
  – How do we specify policies for run-time monitors to enforce?

• Understanding of a general run-time mechanism for guaranteeing control flow integrity (CFI)
  – CFI guarantees absence of control hijacking attacks
Summary: Course Topics & Goals

Course Topics:
0. Attacks
1. Software Security Architectures
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement

Course Goals:
1. Understand current software attacks and defenses
2. Understand general principles of secure software design, implementation and analysis

• Control Flow Integrity
• Information Flow
A Final Comment

Do not try attacks at home! Our goal is to educate so you can *defend*, not *attack*.