CSE 291

Building Secure Systems using Programming Languages and Analysis

Fall 2016
Tue/Thurs 5:00-6:20PM
Deian Stefan
UC San Diego
Who am I?

• New assistant professor
  ➢ PhD at Stanford (Mazieres & Mitchell)
• I like to build secure systems and think about them formally
  ➢ Security + Systems + PL
  ➢ Large focus: web servers and web browsers
• I have a startup: security runtime sys for node
• I sometimes participate in W3C spec work
Who are you?

(Please write your name on paper and put in in front of you.)
Today

• Details about the course
• Course topics
• Read and discuss paper
Administrivia

• Course website
  ➢ https://cseweb.ucsd.edu/~dsteфан/cse291-fall16
  ➢ https://cse291.programming.systems

• Contact
  ➢ Piazza: https://piazza.com/ucsd/fall2016/cse291
  ➢ Personal: deian+cse291@cs.ucsd.edu

• Office hours
  ➢ Wed 1:30-2:30PM
Course objectives

• Objectively read research papers

• Think critically (sometimes formally) about security and system designs

• Work on a research project that spans PL, OS, and security
  ➤ Leverage ideas from one domain to solve problems in another

• Present research results
Course objectives

- Objectively read research papers
- Think critically (sometimes formally) about security and system designs
- Work on a research project that spans PL, OS, and security
  - Leverage ideas from one domain to solve problems in another
- Present research results
Course objectives

- Objectively read research papers
- Think critically (sometimes formally) about security and system designs
- Work on a research project that spans PL, OS, and security
  - Leverage ideas from one domain to solve problems in another
- Present research results
Course objectives

• Objectively read research papers
• Think critically (sometimes formally) about security and system designs
• Work on a research project that spans PL, OS, and security
  ➤ Leverage ideas from one domain to solve problems in another
• Present research results
Course style

• Read and discuss 1 paper / class meeting
  ➢ Short writing assignments due before each class
  ➢ Most class time will be spent discussing papers

• Work on a relatively large project
  ➢ Short presentation at the end of quarter
  ➢ Short write-up (approx. 5pp) at the end of quarter
Course style

• Read and discuss 1 paper / class meeting
  ➤ Short writing assignments due before each class
  ➤ Most class time will be spent discussing papers

• Work on a relatively large project
  ➤ Short presentation at the end of quarter
  ➤ Short write-up (approx. 5pp) at the end of quarter
Writing assignments (30%)

• Summarize paper
  ➢ Main points, 1-2 paragraphs
  ➢ Exemplary summaries may be posted on course site

• Answer questions
  ➢ Goal: think deeply about the paper
  ➢ Non-goal: testing you
  ➢ Exemplary/interesting answers may be posted on site
Writing assignments (30%)

• Summarize paper
  ➤ Main points, 1-2 paragraphs
  ➤ Exemplary summaries may be posted on course site

• Answer questions
  ➤ Goal: think deeply about the paper
  ➤ Non-goal: testing you
  ➤ Exemplary/interesting answers may be posted on site
Class participation (25%)

- Lead the discussion one paper
  - Choose paper (will post howto on Piazza)
  - Write discussion notes to be posted on site
  - Keep the class engaged with questions/comments
  - Often helpful to read some of the related work to get more breadth/depth
  - Come talk to me about other resources
Class participation (25%)

• Come to class prepared to discuss paper
  ➤ No discussions = no fun
  ➤ Read paper 2-3 times, small details matter
  ➤ Come with feedback, thoughts, and questions
  ➤ Question the paper problem statement, question assumptions, question solution, question evaluation, question everything!
  ➤ Post comments, questions, etc. on Piazza
One rule

Be excellent to each other.
Be excellent to each other.
Final project (45%)

• Work on original research
  ➢ Build a new system or extend an existing one, formalize/prove something about a system, disprove the results of an existing paper, etc.

• Can use your research for the final project
  ➢ Please confirm this with me first

• I will provide a list of project ideas soon
Final project (45%)

• Work on original research
  ➤ Build a new system or extend an existing one, formalize/prove something about a system, disprove the results of an existing paper, etc.

• Can use your research for the final project
  ➤ Please confirm this with me first

• I will provide a list of project ideas soon
Final project (45%)

• Form teams of 2-3 people in next 2 weeks
  ➢ Outside this range: come talk to me

• Mid-quarter updates
  ➢ Come talk to me about status of project

• Final presentation and write-up
  ➢ Show off what you did
  ➢ Tell us what you learned + where/why/how things failed
  ➢ Write short conference-like paper describing your work
Final project (45%)

• Fallback: paper reading project
  ➢ Alternative to building
  ➢ Read handful of papers on common theme
  ➢ Come up with research direction from the papers

• Must get approval for this from me
  ➢ Expectation: understand the papers and area deeply
Final project (45%)

- Fallback: paper reading project
  - Alternative to building
  - Read handful of papers on common theme
  - Come up with research direction from the papers
- Must get approval for this from me
  - Expectation: understand the papers and area deeply
Grading summary

• Writing assignments (30%)
• Class participation (25%)
• Final project (45%)
• No exams!
You’ll also get 2 free passes

• 2 no-questions asked passes towards
  ➢ Writing assignments
  ➢ Class participation (not when leading discussion)

• What does this mean?
  ➢ You didn’t do the writing assignment (in time): use up a pass
  ➢ You can’t show up to class: use up a pass

• Exceptional cases: contact me
You’ll also get 2 free passes

• 2 no-questions asked passes towards
  ➢ Writing assignments
  ➢ Class participation (not when leading discussion)

• What does this mean?
  ➢ You didn’t do the writing assignment (in time): use up a pass
  ➢ You can’t show up to class: use up a pass

• Exceptional cases: contact me
Collaboration policy: collaborate!

• Talk with each other, talk on Piazza
  ➤ Good ideas come from talking to smart people

• Writing assignments
  ➤ Write your own, but if you discussed with others/used external resources: acknowledge them

• Project
  ➤ Talk to others about your project, acknowledge them in your write-up if it helped/led to something
Again,

*Be excellent to each other.*
Again,

Be excellent to each other.
Who should take this class?

- Those interested in learning how to:
  - build secure systems
  - use various (PL) techniques to address security
  - reason about security using PL semantics
Prerequisites

• Programming languages
  ➤ Type systems, structural operational semantics, parse trees, CFGs

• Operating systems
  ➤ Processes, virtual memory, concurrency, CPU modes

• Security
  ➤ Web security, buffer overflows, TLS, MPC
Prerequisites

• Some familiarity + willingness to learn

• If you’re not familiar with something: ask!
  ➤ I can post external resources (e.g., book chapters)
  ➤ Post on Piazza: others can help explain things
  ➤ Ask questions in class
  ➤ Come to office hours

• Not knowing something is okay
  ➤ Asking + providing help counts towards participation
Prerequisites

• Some familiarity + willingness to learn

• If you’re not familiar with something: ask!
  ➢ I can post external resources (e.g., book chapters)
  ➢ Post on Piazza: others can help explain things
  ➢ Ask questions in class
  ➢ Come to office hours

• Not knowing something is okay
  ➢ Asking + providing help counts towards participation
Today

- Details about the course
- Course topics
- Read and discuss paper
Topics

We’re going to learn how different PL techniques can be used to provide security in various systems domains.
PL techniques

• Language runtime security monitors
• Type systems for enforcing security
• Authenticated data structures
• Domain specific languages
• Symbolic execution and micro-grammars
• Refinement types and protocol verification
Security properties/mechanisms

- Mandatory access control and confinement
- Least privilege
- Privilege separation
- Software fault isolation
- Control flow integrity
System domains

• Language runtimes
• Server-side web frameworks
• Browser and extension architectures
• New and existing operating systems
• New hardware architectures
• Cryptography and network protocols
Example: server-side security

- Problem: web apps are leaking user data
- Why?
  - Apps are plagued with bugs
  - Bugs have security implications
    - Most code runs with privilege of process: grave
How can PL techniques help to address this problem?

- **Eliminate classes of bugs!**
  - Types can be used to eliminate code injection
  - DSLs (e.g., ORMs) can rid of SQLi
  - New programming models can prevent bugs due to programmer policy enforcement
  - Security monitors can enforce that
    - potentially buggy code doesn’t leak sensitive data
    - untrusted user input is always sanitized (XSS/SQLi)
How can PL techniques help to address this problem?

• Eliminate classes of bugs!
  ➤ Types can be used to eliminate code injection
  ➤ DSLs (e.g., ORMs) can rid of SQLi
  ➤ New programming models can prevent bugs due to programmer policy enforcement
  ➤ Security monitors can enforce that
    ➤ potentially buggy code doesn’t leak sensitive data
    ➤ untrusted user input is always sanitized (XSS/SQLi)
How can PL techniques help to address this problem?

• **Eliminate classes of bugs!**
  - Types can be used to eliminate code injection
  - DSLs (e.g., ORMs) can rid of SQLi
  - New programming models can prevent bugs due to programmer policy enforcement
  - Security monitors can enforce that
    - potentially buggy code doesn’t leak sensitive data
    - untrusted user input is always sanitized (XSS/SQLi)
How can PL techniques help to address this problem?

- **Eliminate classes of bugs!**
  - Types can be used to eliminate code injection
  - DSLs (e.g., ORMs) can rid of SQLi
  - New programming models can prevent bugs due to programmer policy enforcement
    - Security monitors can enforce that
      - potentially buggy code doesn’t leak sensitive data
      - untrusted user input is always sanitized (XSS/SQLi)
How can PL techniques help to address this problem?

- **Eliminate classes of bugs!**
  - Types can be used to eliminate code injection
  - DSLs (e.g., ORMs) can rid of SQLi
  - New programming models can prevent bugs due to programmer policy enforcement
  - Security monitors can enforce that
    - potentially buggy code doesn’t leak sensitive data
    - untrusted user input is always sanitized (XSS/SQLi)
Why take the PL approach?

• This is the interface to the programmer

• Knowing if SQL statement is good/bad is hard
  ➤ ORM can provide safe interface by construction

• Right abstraction layer for enforcing app security
  ➤ OS pages are a bit too coarse grained to be used to protect objects within app from 3rd party lib
  ➤ Apps typically have notion of users ≠ OS UIDs
  ➤ Have more information about what’s going on
Why take the PL approach?

• This is the interface to the programmer

• Knowing if SQL statement is good/bad is hard
  ➤ ORM can provide safe interface by construction

• Right abstraction layer for enforcing app security
  ➤ OS pages are a bit too coarse grained to be used to protect objects within app from 3rd party lib
  ➤ Apps typically have notion of users ≠ OS UIDs
  ➤ Have more information about what’s going on
Why take the PL approach?

- This is the interface to the programmer
- Knowing if SQL statement is good/bad is hard
  - ORM can provide safe interface by construction
- Right abstraction layer for enforcing app security
  - OS pages are a bit too coarse grained to be used to protect objects within app from 3rd party lib
  - Apps typically have notion of users ≠ OS UIDs
  - Have more information about what’s going on
Why take the PL approach?

- Techniques are finally mature and fast enough for us to use
- Amendable to formal analysis
  - E.g., can prove that if your write code with this DSL that code can’t leak or corrupt sensitive data
  - E.g., can prove that if your circuit type checks then it doesn’t have timing channels
Why take the PL approach?

- Techniques are finally mature and fast enough for us to use
- Amendable to formal analysis
  - E.g., can prove that if your write code with this DSL that code can’t leak or corrupt sensitive data
  - E.g., can prove that if your circuit type checks then it doesn’t have timing channels
This is all fluid

- We can consider:
  - alternative domains
  - specific techniques (e.g., faceted values)
  - alternative papers within domain (e.g., seL4)

- Class is meant to be fun for you!
Today

- Details about the course
- Course topics
- Read and discuss paper

http://thewritesisters.blogspot.com/2012/06/mentor-monday-common-mistakes.html