Week 1: Lecture B
Research 101: Ideas

Wednesday, January 10, 2024
Recap: Course Website

cs.utah.edu/~snagy/courses/cs5963

CS 5963/6963: Applied Software Security Testing

This special topics course will dive into today’s state-of-the-art techniques for uncovering hidden security vulnerabilities in software. Introductory fuzzing exercises will provide hands-on experience with industry-popular security tools such as AFL+ and AddressSanitizer, culminating in a final project where you’ll work to hunt down, analyze, and report security bugs in a real-world application or system of your choice.

This class is open to graduate students and upper-level undergraduates. It is recommended you have a solid grasp over topics like software security, systems programming, and C/C++.

Learning Outcomes: At the end of the course, students will be able to:

- Design, implement, and deploy automated testing techniques to improve vulnerability on large and complex software systems.
- Assess the effectiveness of automated testing techniques and identify why they are well- or ill-suited to specific codebases.
- Distill testing outcomes into actionable remediation information for developers.
- Identify opportunities to adapt automated testing to emerging and/or unconventional classes of software or systems.
- Pinpoint testing obstacles and synthesize strategies to overcome them.
- Appreciate that testing underpins modern software quality assurance by discussing the advantages of proactive and post-deployment software testing efforts.
Recap: Course Resources

Course website .................. assignments, schedule, slides, paper signup

Piazza ........................................... questions, discussion, announcements

Canvas ........................................ homework submission, course gradebook

Instructor email (snagy@cs.utah.edu) ...................... administrative issues
Recap: Lateness Policy

- Assignments will be posted on course website
  - See cs.utah.edu/~snagy/courses/cs5963/assignments

- Due by **11:59 PM** on the specified deadline date
  - Late assignments will **not** be accepted

- If you are sick / traveling / abducted by aliens...
  - Try to keep me posted and we will figure something out
Recap: Course Materials

- No textbook is required for this course

- Some excellent resources on fuzzing are:
  - *The Fuzzing Book* by Zeller, Gopinath, Böhme, Fraser, and Holler
  - *Fuzzing Against the Machine* by Antonio Nappa and Blazquez

- Other general computer security textbooks:
  - *Introduction to Computer Security* by Goodrich and Tamassia
  - *Security Engineering* by Ross Anderson

- These are linked on the course syllabus
  - [cs.utah.edu/~snagy/courses/cs5963/](http://cs.utah.edu/~snagy/courses/cs5963/)
Recap: No Exams
Recap: Paper Presentations

- **Signup sheet** available on course website (must use **UofU gcloud** account)
- **38 fuzzing papers** from top venues in security, software engineering, and some workshops
- Choose one paper by **Monday, January 22**

**CS 5963/6963: Applied Software Security Testing**

This special topics course will dive into today’s state-of-the-art techniques for uncovering hidden security vulnerabilities in software. Introductory fuzzing exercises will provide hands-on experience with industry-popular security tools such as AFL+ and AddressSanitizer, culminating in a final project where you’ll work to hunt down, analyze, and report security bugs in a real-world application or system of your choice.

This class is open to graduate students and upper-level undergraduates. It is recommended you have a solid grasp over topics like software security, systems programming, and C/C++.

**Learning Outcomes:** At the end of the course, students will be able to:

- Design, implement, and deploy automated testing techniques to improve vulnerability on large and complex software systems.
- Assess the effectiveness of automated testing techniques and identify why they are well- or ill-suited to specific codebases.
- Distill testing outcomes into actionable remediation information for developers.
- Identify opportunities to adapt automated testing to emerging and/or unconventional classes of software or systems.
- Pinpoint testing obstacles and synthesize strategies to overcome them.
- Appreciate that testing underpins modern software quality assurance by discussing the advantages of proactive and post-deployment software testing efforts.
Recap: Key Dates

- **Jan. 15**  No class (MLK Jr. Day)
- **Jan. 22**  Select one paper to present
- **Feb. 07**  Lab 1 due
- **Feb. 14**  Lab 2 due
- **Feb. 19**  No class (President’s Day)
- **Feb. 28**  Lab 3 due
- **Feb. 28**  5-minute project proposals
- **Mar. 04 & 06**  No class (Spring Break)
- **Apr. 17 & 22**  Final project presentations

---

Part 1: Course Intro and Research 101

<table>
<thead>
<tr>
<th>Monday Meeting</th>
<th>Wednesday Meeting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan. 08</td>
<td>Jan. 10</td>
</tr>
<tr>
<td>Course Introduction</td>
<td>Research 101: Ideas</td>
</tr>
<tr>
<td>Jan. 15</td>
<td>Jan. 17</td>
</tr>
<tr>
<td>No Class (Martin Luther King Jr. Day)</td>
<td>Research 101: Writing</td>
</tr>
<tr>
<td>Jan. 22</td>
<td>Jan. 24</td>
</tr>
<tr>
<td>Research 101: Reviewing and Presenting</td>
<td>Introduction to Fuzzing</td>
</tr>
<tr>
<td>Sign up for paper presentations by 11:59pm</td>
<td>Beginner Fuzzing Lab released</td>
</tr>
</tbody>
</table>

Part 2: Fuzzing Fundamentals

<table>
<thead>
<tr>
<th>Monday Meeting</th>
<th>Wednesday Meeting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan. 29</td>
<td>Jan. 31</td>
</tr>
<tr>
<td>Input Generation</td>
<td>Runtime Feedback</td>
</tr>
<tr>
<td>&gt; Readings:</td>
<td>&gt; Readings:</td>
</tr>
<tr>
<td>Feb. 05</td>
<td>Feb. 07</td>
</tr>
<tr>
<td>Bugs &amp; Triage I</td>
<td>Bugs &amp; Triage II</td>
</tr>
<tr>
<td>&gt; Readings:</td>
<td>&gt; Readings:</td>
</tr>
<tr>
<td>Triage Lab released</td>
<td>Beginner Fuzzing Lab due by 11:59pm</td>
</tr>
<tr>
<td>Feb. 12</td>
<td>Feb. 14</td>
</tr>
<tr>
<td>Harnessing I</td>
<td>Harnessing II</td>
</tr>
<tr>
<td>&gt; Readings:</td>
<td>&gt; Readings:</td>
</tr>
<tr>
<td>Harnessing Lab released</td>
<td>Triage Lab due by 11:59pm</td>
</tr>
</tbody>
</table>

---

[cs.utah.edu/~snagy/courses/cs5963/schedule]
Questions?
This time on CS 5963...

Research 101: Ideas
What is “Research”?

This →
Generating an Idea

Also this ←
Pursuing an Idea
Course Goals

- Help you become better researchers
-Expose you to different perspectives
-Experience with state-of-the-art tools
-Get course credit so you can graduate?
-All while learning about software testing
Course Goals

- Help you become **better researchers**
- Expose you to **different perspectives**
- Experience with **state-of-the-art tools**
- Get course credit so you can graduate?
- All while learning about **software testing**

**Reading / evaluating published research**

**Conducting / presenting your own research**

**Research 101:** Ideas, Writing, Reviewing, and Presenting Research
Ideas:
The Foundation of Research
What are “Ideas”?
Ideas underpin research...

- Great ideas can be **ruined by bad execution**
  - Think of every neat Shark Tank product... with poor sales
Ideas underpin research...

- Great ideas can be **ruined by bad execution**
  - Think of every neat Shark Tank product... with poor sales

- Great execution **cannot promote a poor idea**
  - Think of every dumb Shark Tank product... with good sales
**Ideas underpin research...**

- Great ideas can be **ruined by bad execution**
  - Think of every neat Shark Tank product... with poor sales

- Great execution **cannot promote a poor idea**
  - Think of every dumb Shark Tank product... with good sales

- Great ideas are **context and time sensitive**
  - There is a “right time” for specific ideas
  - If you move too slow, you’ll get scooped!

---

*Great ideas underpin great research*
Where do great ideas come from?

How to come up with research ideas?

As a very new researcher who is exploring the best way to generate ideas, some guidance on this question would be very helpful. I have found that this is NOT easy. Ideas seem to pop out of my Professor every day and I wonder how he does it. This question is broad;

- How do you tend to come up with initial/seed ideas? What is your search method (if you have one)?
- What proportion of your ideas come from; (i) colleagues, (ii) intentionally browsing the literature for inspiration, (iv) conferences, (v) other?
- How do you prioritize research ideas?
- Is there any special, general criteria that you discovered to sift out those ideas that are likely to be unrealistic or not ready for a generation?

https://academia.stackexchange.com/questions/5853/how-to-come-up-with-research-ideas
### Where do great ideas come from?

How to come up with ideas?

*As a very interested audience member, I want to know more about this question. I'm not sure if my Professor has had the time to plop out of a box with the guidance on how to pop out of the box (if you didn't brood them). What is the guidance on how to pop out of the box (if you didn't brood them)? What are those ideas that

<table>
<thead>
<tr>
<th>120</th>
<th>128</th>
<th>9 years, 8 months ago</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Stefan Nagy</td>
</tr>
</tbody>
</table>

[Stephan Nagy](https://academia.stackexchange.com/questions/5853/how-to-come-up-with-research-ideas)
Generating Ideas
Problems are not ideas...
... but ideas *emerge* from problems!

- What is the **next big problem** that society will face?
- What are the **unsolved challenges** in a specific area?
- Are there any common yet unsupported **assumptions**?
- Does existing approaches have **unnecessary hurdles**?
**Big vs. Small Ideas**

- **Big ideas:** research directions or arcs
  - What grant proposals are centered around
  - Usually generated solo or with 1–2 others
  - Cannot be too complex—must be realistic
  - Often leads to—or requires—many small ideas
Big vs. Small Ideas

- **Big ideas**: research directions or arcs
  - What grant proposals are centered around
  - Usually generated solo or with 1–2 others
  - Cannot be too complex—must be realistic
  - Often leads to—or requires—many small ideas

- **Small ideas**: one small step forward
  - Concrete, “paper-sized” research projects
  - Usually generated with your collaborators
  - Often the projects assigned to grad students
  - Spin-offs of—or the inspiration behind—big ideas
Themes of Ideas

Observation: That’s odd... ?

Curiosity: What happens if... ?

Challenge: How do I do... ?

Transference: That works there... will it work here?
Before you brainstorm...

The foundation of good ideas is understanding

- How something works
- What assumptions it makes
- How your approach would work
- Why a problem is worth solving
Seeking Inspiration

- Read technical papers
- Read blogs and news sites
- Attend technical presentations
- **Talk with your colleagues**
- Volunteer to review papers
- Present already-published work
- Work with those in industry
- **Do something other than research**
Seeking Inspiration

- **Categorize previous work:** look for patterns or missed opportunities

<table>
<thead>
<tr>
<th>Execution Mechanism</th>
<th>Fuzzing Implementations</th>
<th>Level of Efficiency</th>
<th>Execution Correctness</th>
<th>Windows Kernel Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Target</td>
<td>Kernel</td>
<td></td>
</tr>
<tr>
<td>Process Creation</td>
<td>WinAFL, [18], Masnul [35], KillerBees [36],</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Forkserver-based Cloning</td>
<td>Winnie [19]</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>In-memory Looping</td>
<td>WinAFL, [18], TinyAFL, [37], Jackalope [38],</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Kernel-based Snapshotting</td>
<td>AFL++, LKM [17], Xu et al. [13], Zhao et al. [16],</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fuzzer</th>
<th>AFL</th>
<th>WinAFL</th>
<th>HonggFuzz</th>
<th>Peach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedback</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Forkserver</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Open-source</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Windows</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technique</th>
<th>Fast</th>
<th>SB</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>C8</th>
<th>C9</th>
<th>C10</th>
<th>Archs</th>
</tr>
</thead>
<tbody>
<tr>
<td>RetroWrite [19]</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>x64</td>
</tr>
<tr>
<td>Repica [23]</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>aarch64</td>
</tr>
<tr>
<td>Egalito [47]</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>aarch64, x64</td>
</tr>
<tr>
<td>DDisasm [22]</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>aarch64, x64</td>
</tr>
<tr>
<td>ICFG [32]</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>aarch64, x64, ppc</td>
</tr>
<tr>
<td>StochFuzz [50]</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>x64</td>
</tr>
<tr>
<td>E9Patch [20]</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>x64</td>
</tr>
<tr>
<td>Multiverse [12]</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>x64</td>
</tr>
</tbody>
</table>
Categorize previous work: look for patterns or missed opportunities

<table>
<thead>
<tr>
<th>Execution Mechanism</th>
<th>Fuzzing Implementations</th>
<th>Level of Efficiency</th>
<th>Execution Correctness</th>
<th>Windows Kernel Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Creation</td>
<td>WinAFL, Manul, KillerBeep</td>
<td>✓ ✓</td>
<td>✓</td>
<td>full</td>
</tr>
<tr>
<td>Forkserver-based Cloning</td>
<td>Winnie [19]</td>
<td>✓ ✓</td>
<td>✓</td>
<td>partial</td>
</tr>
<tr>
<td>In-memory Looping</td>
<td>WinAFL, TinyAFL, Jactalope [38]</td>
<td>✓ ✓</td>
<td>✓</td>
<td>full</td>
</tr>
<tr>
<td>Kernel-based Snapshotting</td>
<td>AFL++, LKM [17], Xu et al. [13], Zhao et al. [16]</td>
<td>✓ ✓ ✓</td>
<td>✓</td>
<td>none</td>
</tr>
<tr>
<td>Target-embedded Snapshotting</td>
<td>WinFuzz</td>
<td>✓ ✓ ✓</td>
<td>✓</td>
<td>full</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fuzzer</th>
<th>AFL</th>
<th>WinAFL</th>
<th>HonggFuzz</th>
<th>Peach</th>
<th>WINNIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedback</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Forkserver</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Open-source</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Windows</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technique</th>
<th>Fast</th>
<th>SB</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>C8</th>
<th>C9</th>
<th>C10</th>
<th>Archs</th>
</tr>
</thead>
<tbody>
<tr>
<td>RetroWrite [19]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x64</td>
</tr>
<tr>
<td>Repica [23]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>aarch64</td>
</tr>
<tr>
<td>Egalito [47]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>aarch64</td>
</tr>
<tr>
<td>DDIsasm [22]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>aarch64</td>
</tr>
<tr>
<td>ICFGFP [32]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>aarch64</td>
</tr>
<tr>
<td>StochFuzz [50]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x64, x64, ppc</td>
</tr>
<tr>
<td>E9Patch [20]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x64</td>
</tr>
<tr>
<td>Multiverse [12]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x64</td>
</tr>
<tr>
<td>ARMore</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>aarch64</td>
</tr>
</tbody>
</table>

1. 4
2. 2
3. N/A
4. 4
Finding great ideas is like mining gold.
Cultivating Ideas
You won’t think of an idea twice!

- **You will forget ideas**
  - You will also forget the *nuance* behind ideas

- **So... write every idea down!**
  - Technology makes this easy nowadays
  - Desktop text file, Google Doc, iOS Notes app, etc.
You won’t think of an idea twice!

- **You will forget ideas**
  - You will also forget the *nuance* behind ideas

- **So... write every idea down!**
  - Technology makes this easy nowadays
  - Desktop text file, Google Doc, iOS Notes app, etc.

- **Go through your idea book periodically**
  - Change in skills, interests, resources, news
  - Enhance, delete, and re-rank
  - **Junior researchers**: start documenting your ideas
  - **Senior researchers**: start organizing & grouping your ideas
Pushing Back on Ideas

Questions to ask yourself:

- What is the **fundamental problem** you are trying to solve?
- What **key observations or insights** lead to your approach?
- What exactly is **your approach** (in less than a paragraph)?
- What must be built to implement the idea? **Can you do it?**
- What are key **evaluation questions** to determine success?
Pushing Back on Ideas

Questions to ask yourself:
- What is the fundamental problem you are trying to solve?
- What key observations or insights lead to your approach?
- What exactly is your approach (in less than a paragraph)?
- What must be built to implement the idea? Can you do it?
- What are key evaluation questions to determine success?

Design a quick exploration to “feel” for success
- Did it work?
- If not, why?
Pursuing Ideas
You have to make a hard choice...

What idea should I work on?
You have to make a hard choice...

What idea should I work on?
The Excitement / Doability Trade-off

- The best ideas are often **the most uncertain**

- Questions to ask yourself
  - Can I learn enough to do it?
  - Will this even work?
  - Will others find this interesting?

- Uncertain ideas often spawn future (more certain) work
Other Considerations

- Advisor (interests, funding)
- Department/lab resources
- Your skills and experience
- Available experts
- Potential collaborators
- How easy to publish/potential impact
- Publication venues

and...
The “First Mover” Advantage

- If you are not the first to an idea, you likely will face an uphill battle.
- How far behind are you in knowledge, resources, and infrastructure?
Detouring vs. Committing

- Write distracting ideas down
- Talk with your advisor or labmates
- **Good ideas** can lead to **great ideas**
- Be prepared to let an idea go
We don’t publish an idea...

Hey Mark! I’m seeking one trillion dollars so that I can build the world’s first flying car!
We don’t publish an idea... we publish its proof!

Hey Mark! I’m seeking one trillion dollars so that I can build the world’s first flying car!

Hey Mark! I’ve built a flying car that costs far less than a conventional car, and I’d like one trillion dollars to mass-produce it...
Research is a process...
Research is a random process...
Research is a random process...
Research is a **random** process...
Research is a random and iterative process...
Resources

- **How to Look for Ideas in Computer Science Research** by Zhiyun Qian
- **Finding a Topic and Beginning Research** by Dianne Prost O'Leary
- **How to come up with research ideas?** on StackExchange
- **How to Get Startup Ideas** by Paul Graham
- **Creativity Techniques** on Wikipedia
Questions?
Next time on CS 5963/6963...

Research 101: Writing