Holistic Software Security

Introduction

Aravind Machiry

What this class is not about!

Writing exploits - Although, you will have better idea to do after the class.

Binary analysis - Although, the principles are similar.

Software Security

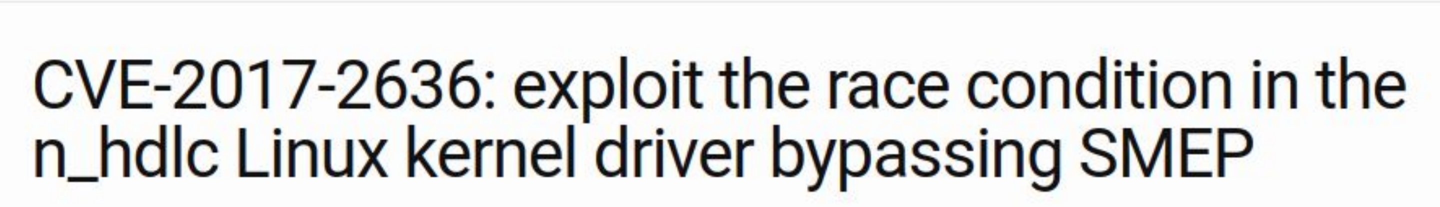
- What do we mean by this?
- Why do we need this?
- How to achieve this?

What?

- Ensuring that the given software (e.g., a program, OS) does not have security flaws.
- Security flaws:
    - Arbitrary code execution.
    - Arbitrary read/write.
    - Denial-of-Service.
    - Race condition.

What?

- Depending on the software, flaws might be more serious.
    - Race condition on a **local program `ls`** v/s in **Linux Kernel**.



CVE-2017-2636: exploit the race condition in the n_hdlc Linux kernel driver bypassing SMEP

Bug v/s Vulnerability

- Bug: Program misbehaves and/or does not produce desired outcome.

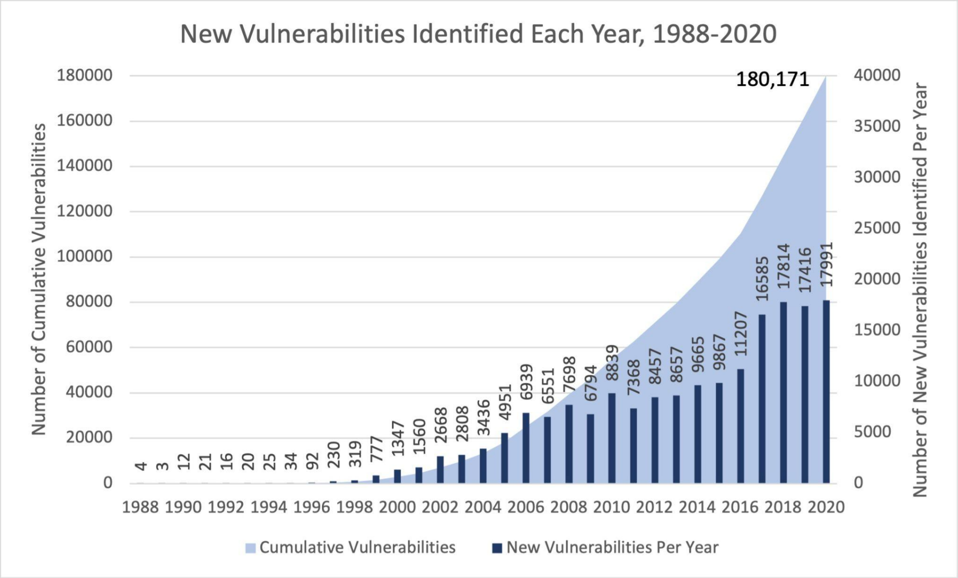
```
scanf(“%d”, &i);
j = i + 2;
```

- **Vulnerability**: A bug which could be exploited to cause a security flaw.

```
p = malloc(j);
p[i] = ...
```

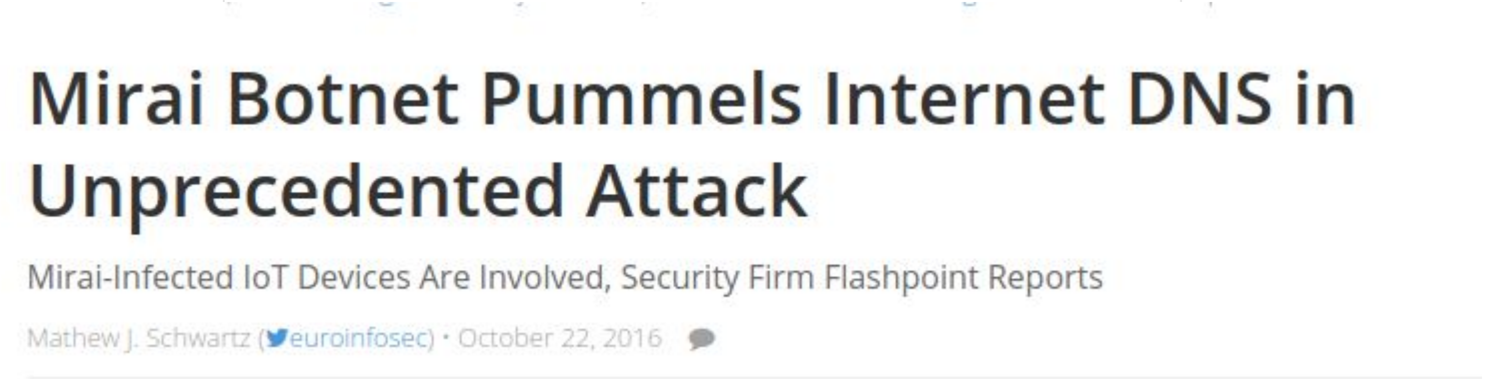
Why we need Software Security?

Why we need Software Security?



Why we need Software Security?

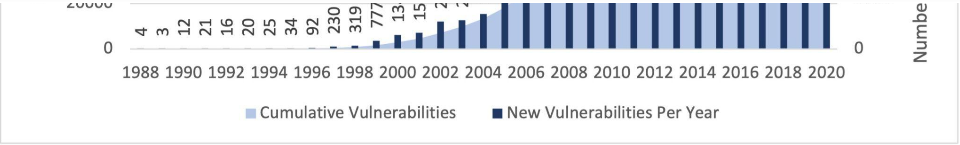
New Vulnerabilities Identified Each Year, 1988-2020



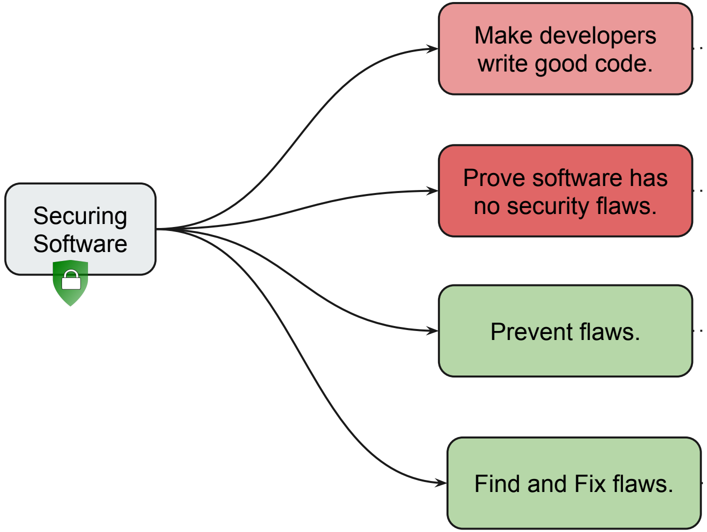
Mirai Botnet Pummels Internet DNS in Unprecedented Attack

Mirai-Infected IoT Devices Are Involved, Security Firm Flashpoint Reports

Mathew J. Schwartz (euroinfosec) · October 22, 2016



How can we achieve this?



Securing Software

Make developers write good code.

Prove software has no security flaws.

Prevent flaws.

Find and Fix flaws.

Learn how to code securely

Innovative & Gamified skills development for JAVA and C#

Training Ground

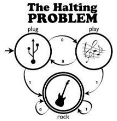
Hinting System

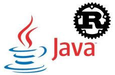
Challenge Feedback

LinkedIn Integration

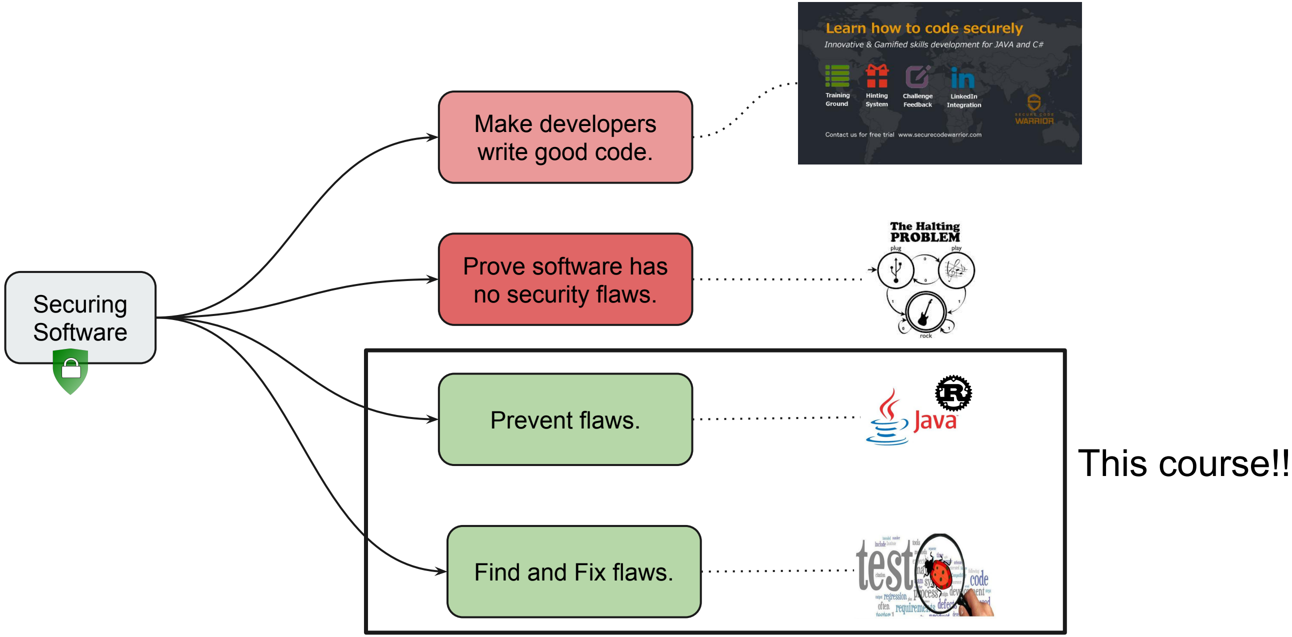
SECURE CODE WARRIOR

Contact us for free trial www.securecodewarrior.com





How can we achieve this?



Securing Software

Make developers write good code.

Learn how to code securely

Innovative & Gamified skills development for JAVA and C#

Training Ground · Hinting System · Challenge Feedback · LinkedIn Integration

Contact us for free trial www.securecodewarrior.com

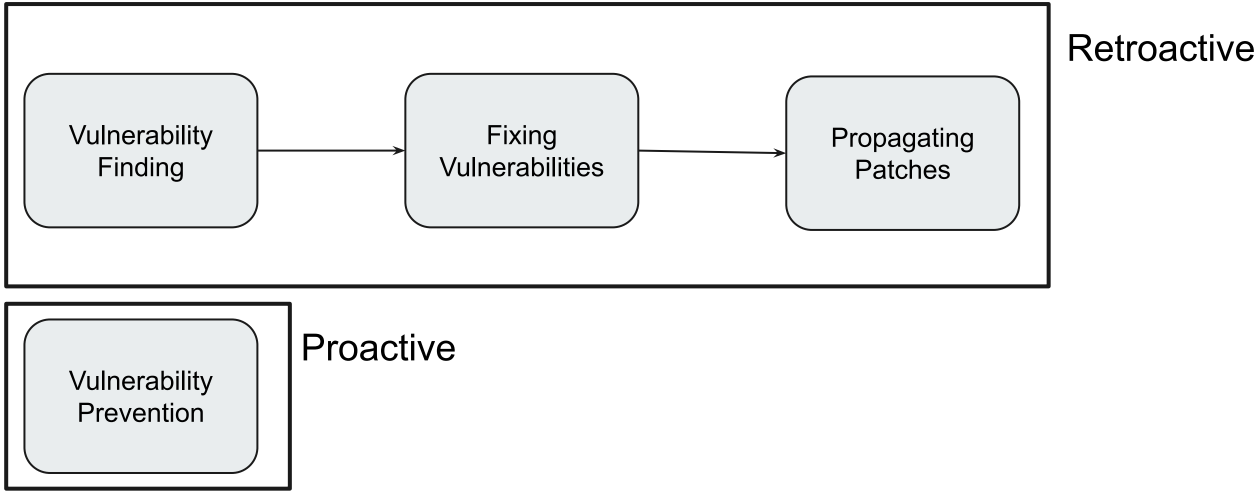
Prove software has no security flaws.

Prevent flaws.

Find and Fix flaws.

This course!!

Course: Organization



Course: Details

- We focus on software written in C/C++.
- Assume source code is available.
- Main focus on memory safety (but will be covering other flaws):
  - Arbitrary read/write.
- Lectures/Research Papers.

Course: Expectations

- Proficiency in C/C++: Ability to work with large code bases.
- OS concepts: Process isolation, User space/kernel space, virtual memory.
- Ability to read scientific papers:
  - https://web.stanford.edu/class/ee384m/Handouts/HowtoReadPaper.pdf
- Lectures/Research Papers.

Course: Expectations (Hopeful)

- Real world impact:
  - You may find zero days in open-source software.
- Get a scientific publication.

Course: Grading

- Four Assignments (10% each = 40%).
- Midterm 1 and 2 (10% each = 20%).
- Paper presentation (10%):
    - You need to pick a paper and present to the class.
- Project (30%)

Project (30%)

- Semester long project:
  - Related to software security (Fairly open ended).
  - Research project.
  - Report, Implementation and Presentation.

- Group of 2 - 3 students (define the project accordingly).
  - Will share the potential list in email.
  - Can pick your own, but should get approval from the professor.

Projects

- Solve halting problem.
- Develop IoT cloud: use idle IoT devices as compute resources.
- Implement stack canaries.
- Automatically fuzz a given program.
- Use Active Learning to find vulnerable functions.
- Runtime shuffling of stack variables.

Thank you!

- ➔ Course Webpage: https://purs3lab.github.io/hss/
- ➔ Join slack using your @purdue email (Link in webpage).
- ➔ Think about your projects.