FORCES
FOUNDATIONS OF RESILIENT
CYBER-PHYSICAL SYSTEMS

# CPS Software Security Analysis and Enhancement: A Case Study

*Chao Zhang*
*Prof. Dawn Song*
*UC Berkeley*

Berkeley UNIVERSITY OF CALIFORNIA

Massachusetts Institute of Technology

NSF

MICHIGAN

VANDERBILT UNIVERSITY

# DARPA Hacks GM's OnStar To Remote Control A Chevrolet Impala (Feb 2015)

Charlie Miller

Chris Valasek

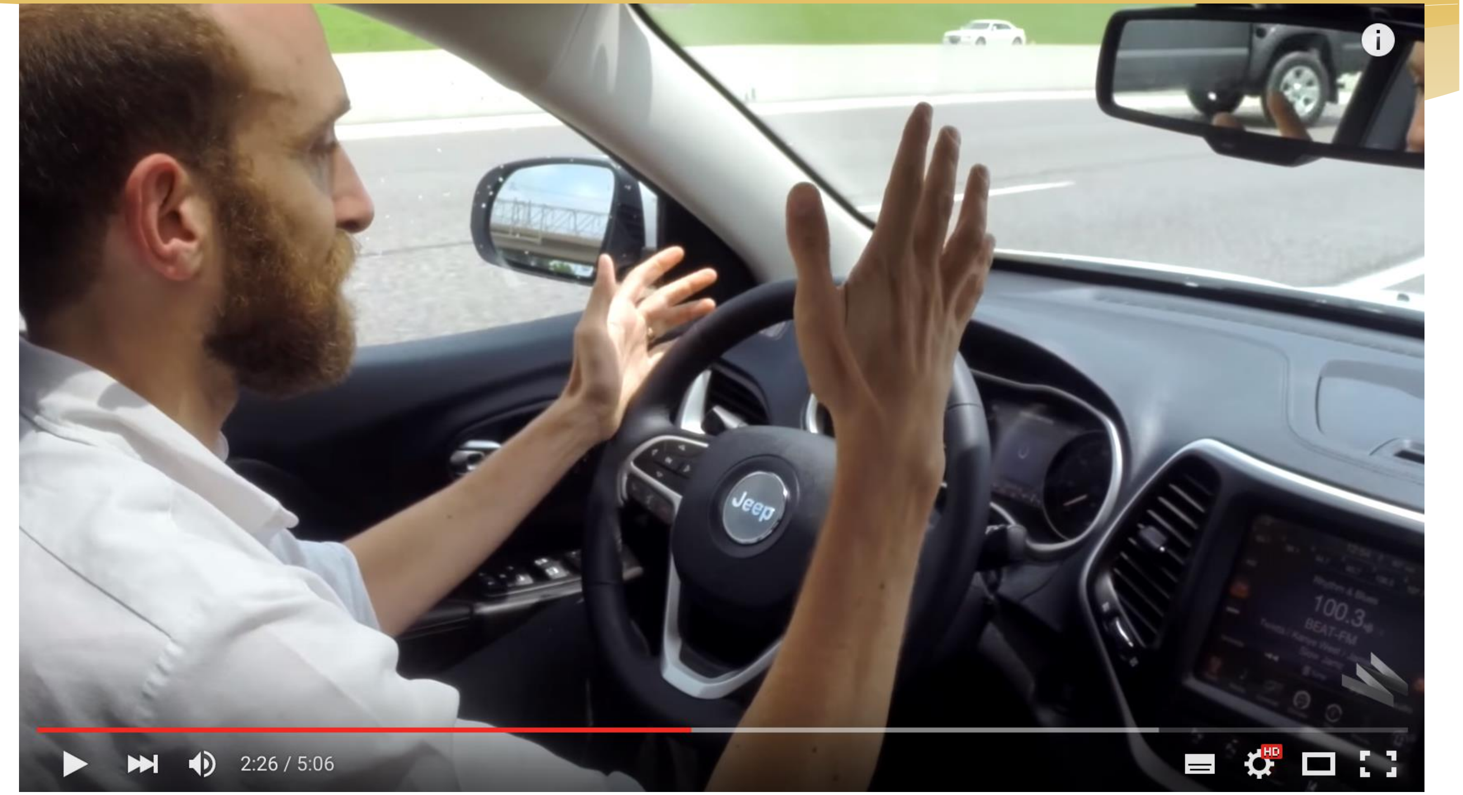
- Dial into the OnStar system (locally), and feed it with malicious packets (containing code), and take control of the car
- http://www.cbsnews.com/news/darpa-dan-kaufman-internet-security-60-minutes/

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

# Hackers Remotely "Kill" a Jeep on the Highway (July 2015)

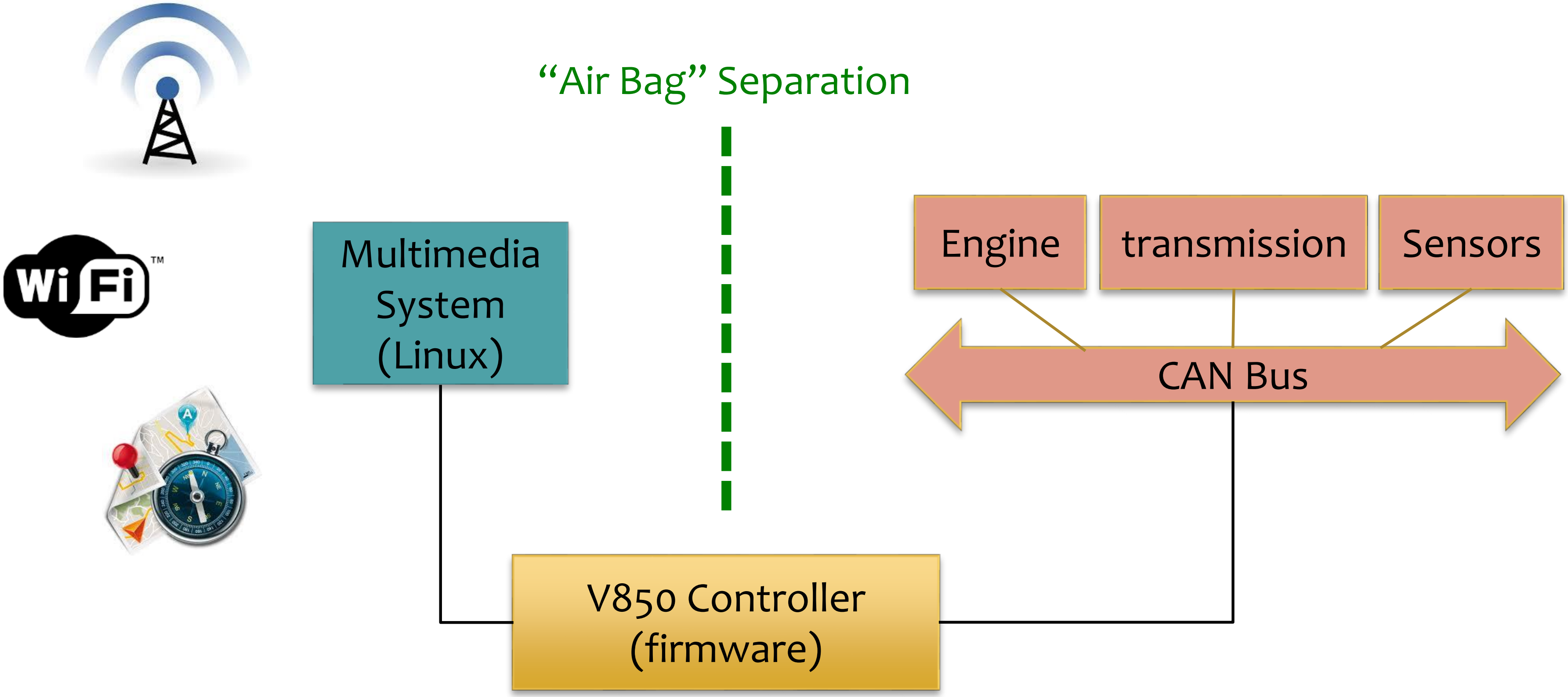
Charlie Miller

Chris Valasek



- Remotely control a jeep on the highway, at a speed of 70mph
  - radio, music player, display
  - horn, windshield wipers, brakes, seat belt, wheel steering
- [https://www.youtube.com/watch?v=MK0SrxBC1xs](https://www.youtube.com/watch?v=MK0SrxBC1xs)

# Under the hood: Chrysler Jeep



“Air Bag” Separation

Multimedia System (Linux)

Engine

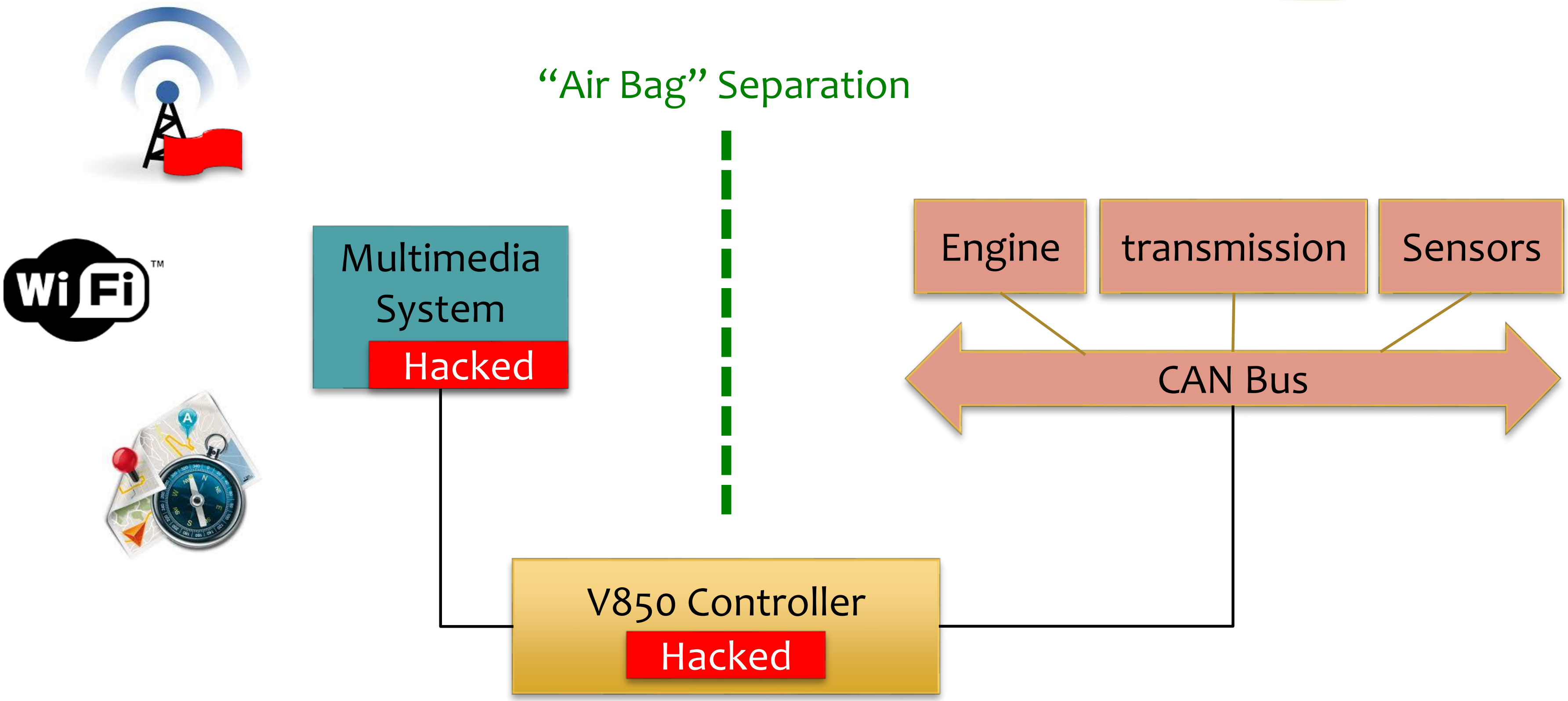
transmission

Sensors

CAN Bus

V850 Controller (firmware)

# Under the hood: Chrysler Jeep



“Air Bag” Separation

Multimedia System

Hacked

Engine

transmission

Sensors

CAN Bus

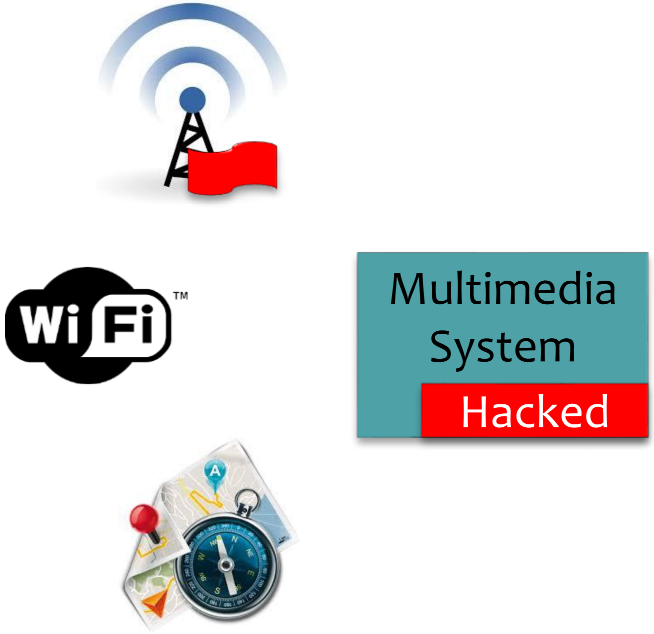
V850 Controller

Hacked

# Outline

- Motivation
- Root Cause Analysis
- Case Study
- Program Hardening

# The Key



Multimedia System

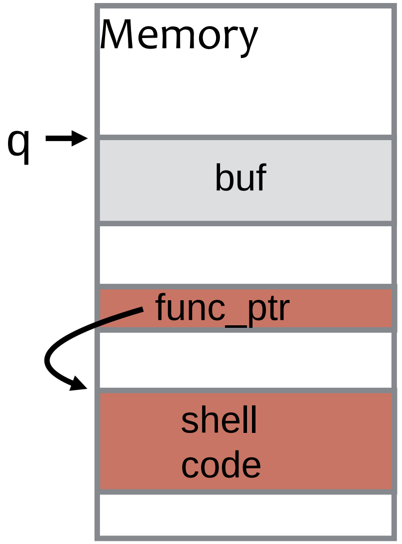
Hacked

- Malicious input
- Software vulnerability
- Exploit and take control

# How to take control? Control-Flow Hijack Attack

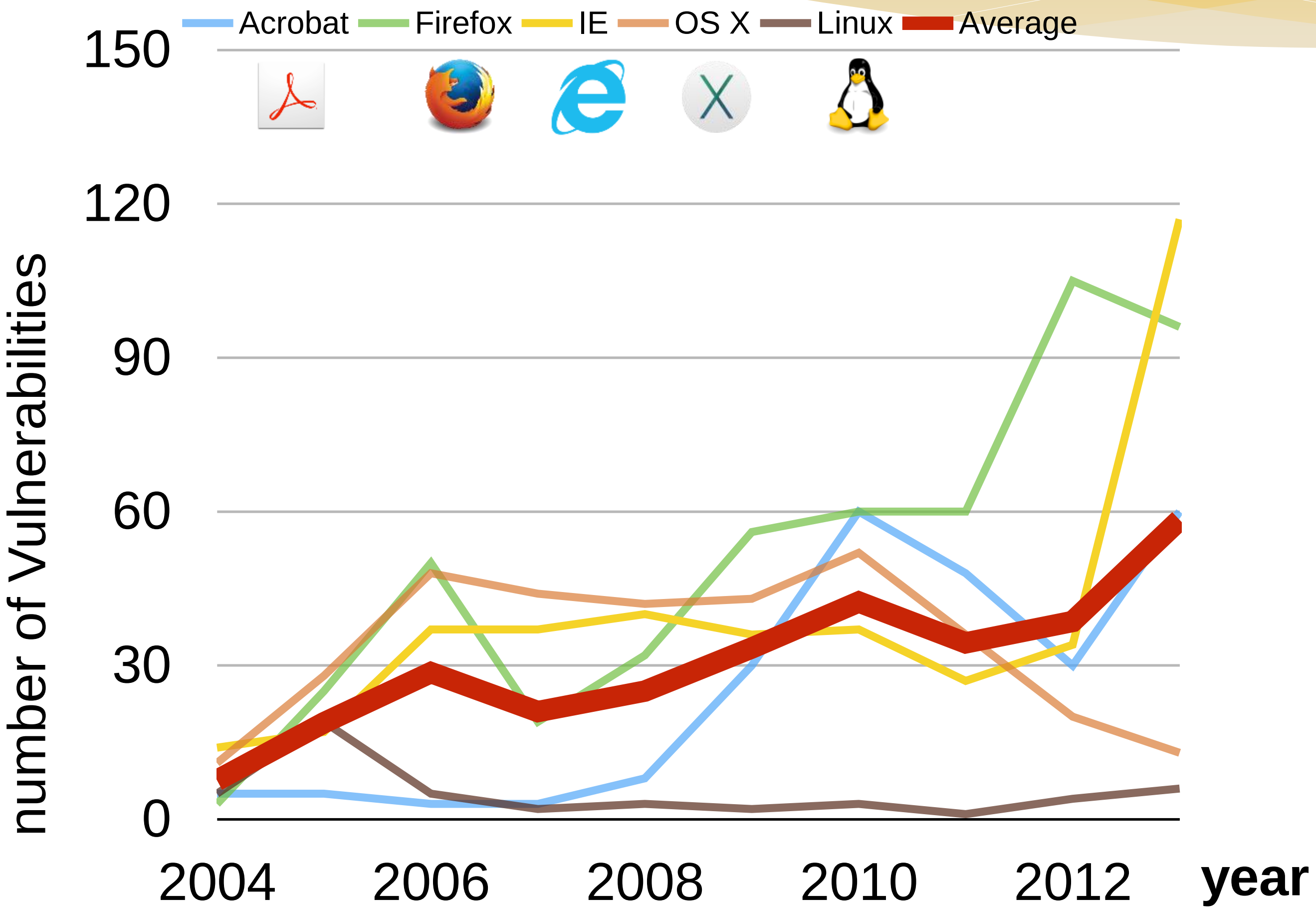
```
int buf[100];
int *q = buf + input;
*q = input2;
…
(*func_ptr)();
```

***execute arbitrary code!***



It started 50 years ago…

# Top Vulnerabilities in CVE Database (Control-Flow Hijack)



FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

# Can we eliminate vulnerabilities?

- Many attack vectors
  - Attackers can feed inputs to software in many ways

- Vulnerabilities are inevitable
  - program complexity and programmer errors
  - vulnerability detection is undecidable

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

# Outline

- Motivation
- Root Cause Analysis
- Case Study: OpenDavinci
- Program Hardening

# OpenDavinci

- What is it?
  - A realtime-capable software development and runtime environment for CPS.

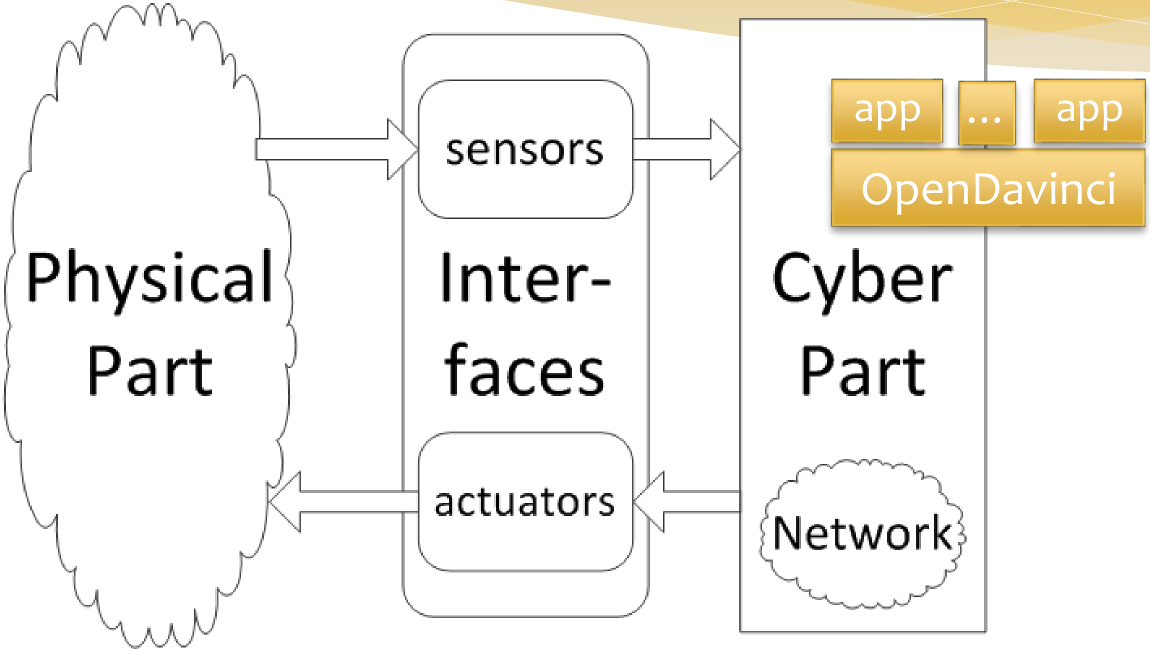
- Use cases

UC Berkeley’s AGV

CaroloCup miniature competition 2013 & 2014

Univ. of Arizona’s AGV

# Attack Vector Analysis



- Sensor input
  - fake sensor, replaced sensor, man-in-the-middle
- Network input
  - fake CPS nodes, replaced nodes, man-in-the-middle

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

# Vulnerability Analysis: Methods

- Static analysis
  - *syntactic* analysis: pattern matching
  - *semantic* analysis: data-flow & control-flow analysis etc.

- Dynamic analysis
  - smart fuzzing: feed programs with **crafted** inputs

- Symbolic execution
  - mark program inputs as symbol, execute the program on symbol values, and check for candidate vulnerabilities

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

# Syntactic Static Analysis (on OpenDavinci)

- FlawFinder

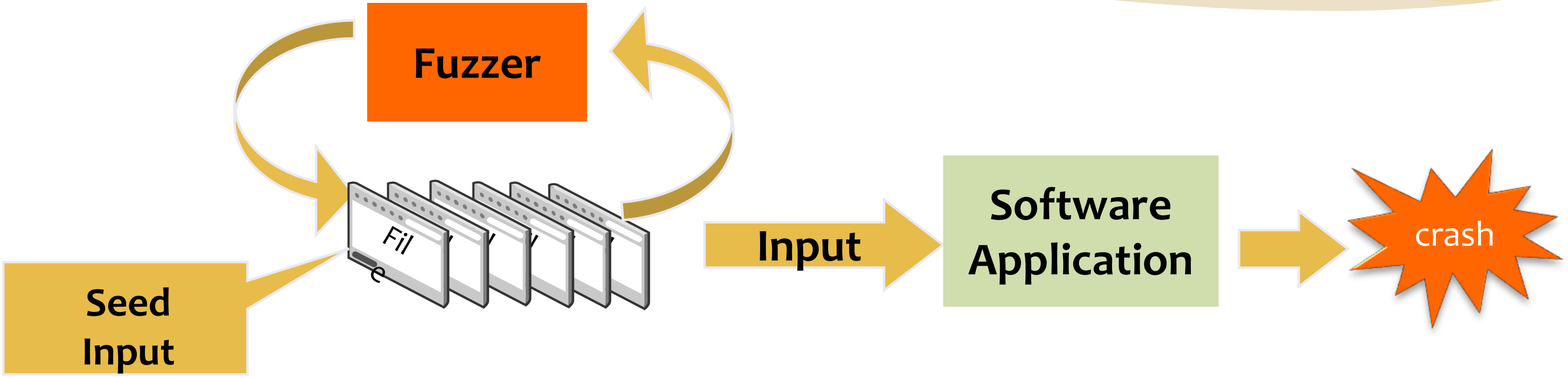
| Risk Level | # Warnings |
|---|---|
| 5 | 5 |
| 4 | 65 |
| 3 | 42 |
| 2 | 384 |
| 1 | 2255 |

- RATS

| Risk Level | # Warnings |
|---|---|
| High | 162 |
| Medium | 697 |

- All high risk warnings are false positives, confirmed manually.
- **Syntactic** static analysis is not sufficient to find real vulnerabilities.

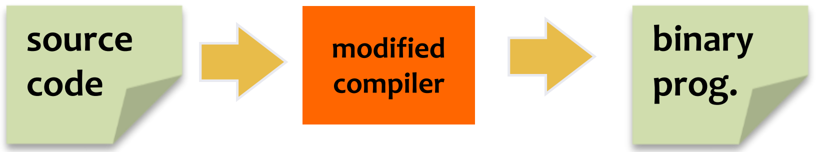
# Smart Fuzzing: Method (on OpenDavinci)



- Basic fuzzing strategy
  - random mutate some bytes of the seed inputs
  - special values (e.g., max, min, 0, 1, etc.)
- Smart Fuzzing
  - we extend the popular fuzzer AFL
  - monitor the execution of inputs, record the traversed code block information
  - filter inputs that trigger new blocks, and mutate them, to explore as many program paths as possible

# Smart Fuzzing: Test-flow (on OpenDavinci)

- Compile OpenDavinci
  - instrument runtime monitoring code

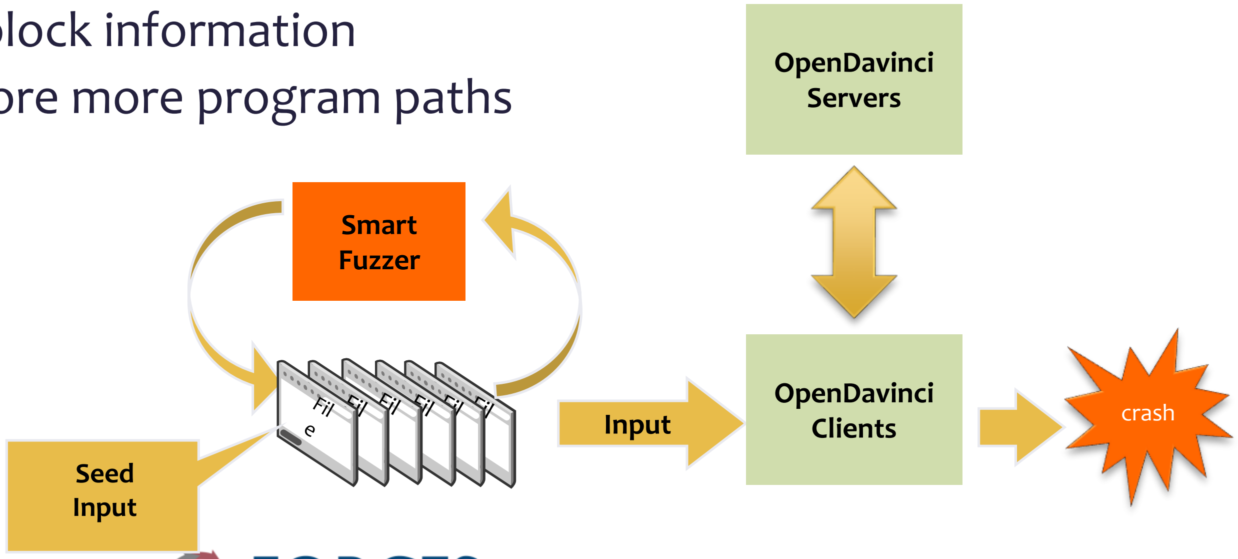


source code

modified compiler

binary prog.

- Test OpenDavinci (distributed)
  - collect runtime code block information
  - mutate inputs to explore more program paths



OpenDavinci Servers

Smart Fuzzer

Seed Input

Input

OpenDavinci Clients

crash

# Smart Fuzzing: Results (on OpenDavinci)

- Target app: ***odrecintegrity***

| Metrics | Value |
|---|---|
| run time | 25 hours |
| total execs | 11.5M times |
| total crashes | 238K |
| unique crashes | **31** |

- All the crash samples can trigger the program to crash
  - i.e., vulnerabilities exist
- Work-in-progress:
  - verify whether these vulnerabilities are exploitable

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

# Smart Fuzzing: Results (on OpenDavinci)

- Target app: ***odsplit***

| Metrics | Value |
|---|---|
| run time | 25 days |
| total execs | 2.21M times |
| total crashes | 2.16M |
| unique crashes | **5000+** |

- Work-in-progress:
  - filter out crashes that are not real bugs

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

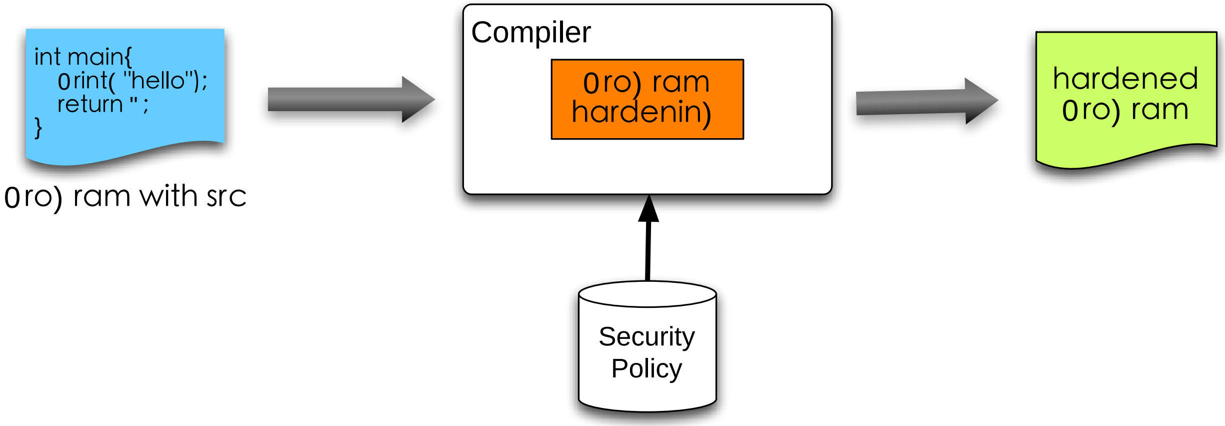
# Outline

- Motivation
- Root Cause Analysis
- Case Study: OpenDavinci
- Program Hardening

# Question & Solution

- The question: *how to protect vulnerable programs?*
  - too many attack vectors to stop
  - vulnerability detection is undecidable

- The solution: *proactive program hardening*



```
int main{
   0rint( "hello");
   return " ;
}
```

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

# Our Security Policy

## Control-flow hijack

```
int *q = buf + input;
*q = input2;
…
(*func_ptr)();
```

## Code Pointer integrity

Enforce the control-flow targets to be intact.

# Code Pointer Integrity

- Separate sensitive pointers and regular data

  *Sensitive pointers =*
  *code pointers +* ***indirect pointers to sensitive pointers***

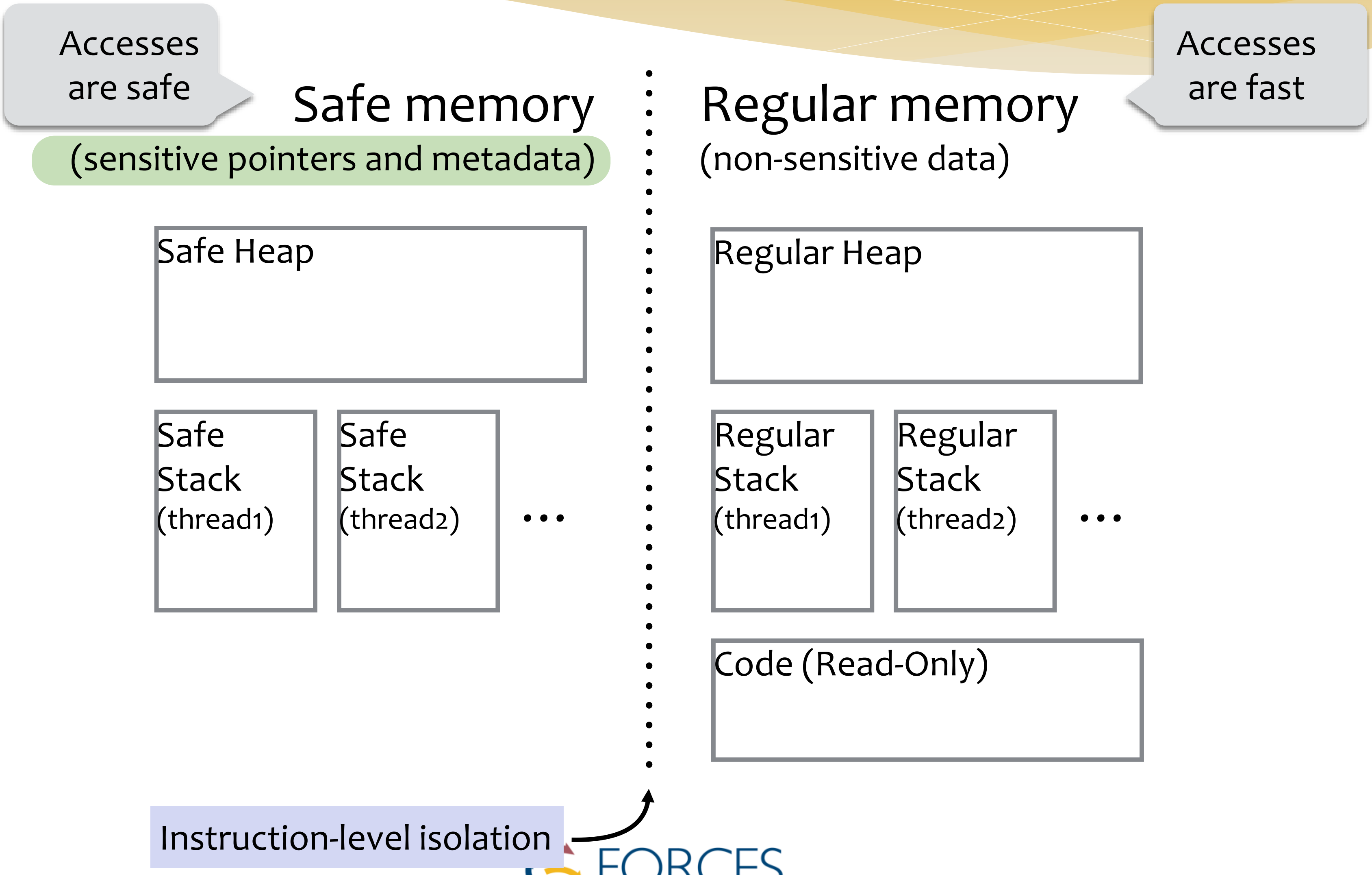
- Enforce sensitive pointers accesses to be safe

  *Separation +* ***runtime checks***

- Keep regular data accesses intact (fast)

  *Instruction-level safe region isolation*

# Guaranteed Protection (CPI): Memory Layout



FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

# Full OS Distribution

- Hardened the entire FreeBSD distribution…
- … and more than 100 packages

FreeBSD hardened

python™

SQLite

lame

OpenSSL

PostgreSQL

GraphicsMagick

Apache

# Harden OpenDavinci with CPI

- Compilation time evaluation
  - the extra program hardening process takes a negligible time.

| Time | Original compilation | CPI compilation |
|---|---|---|
| real | 18m 45.762s | 18m 50.381s |
| user | 10m 1.032s | 10m 2.336s |
| sys | 0m 56.844s | 0m 55.536s |

- File size evaluation
  - all 30 hardened programs have the same size as non-hardened ones
- Work-in-progress
  - performance evaluation (no sufficient benchmarks)
  - security evaluation (no usable exploits)

# Conclusion

- Vulnerabilities are inevitable in software, including CPS software, making them vulnerable to attacks.
- We analyzed a CPS software OpenDavinci, and found more than 30 crashes (i.e., vulnerabilities) in it.
- We proposed a lightweight program hardening solution CPI, able to protect vulnerable programs from being attacked.
- We hardened OpenDavinci with CPI, and evaluated its overhead.

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

# Thanks!

Q&A