CySER Seminar

WASHINGTON STATE UNIVERSITY

Information Flow Control

From Classical Security to Agentic AI

Basavesh Ammanaghatta Shivakumar

School of Electrical Engineering and Computer Science
Washington State University

Before we start

A little about me



Basavesh Ammanaghatta Shivakumar

Assistant Professor, School of EECS, Washington State University

basavesh.github.io

MY PATH SO FAR

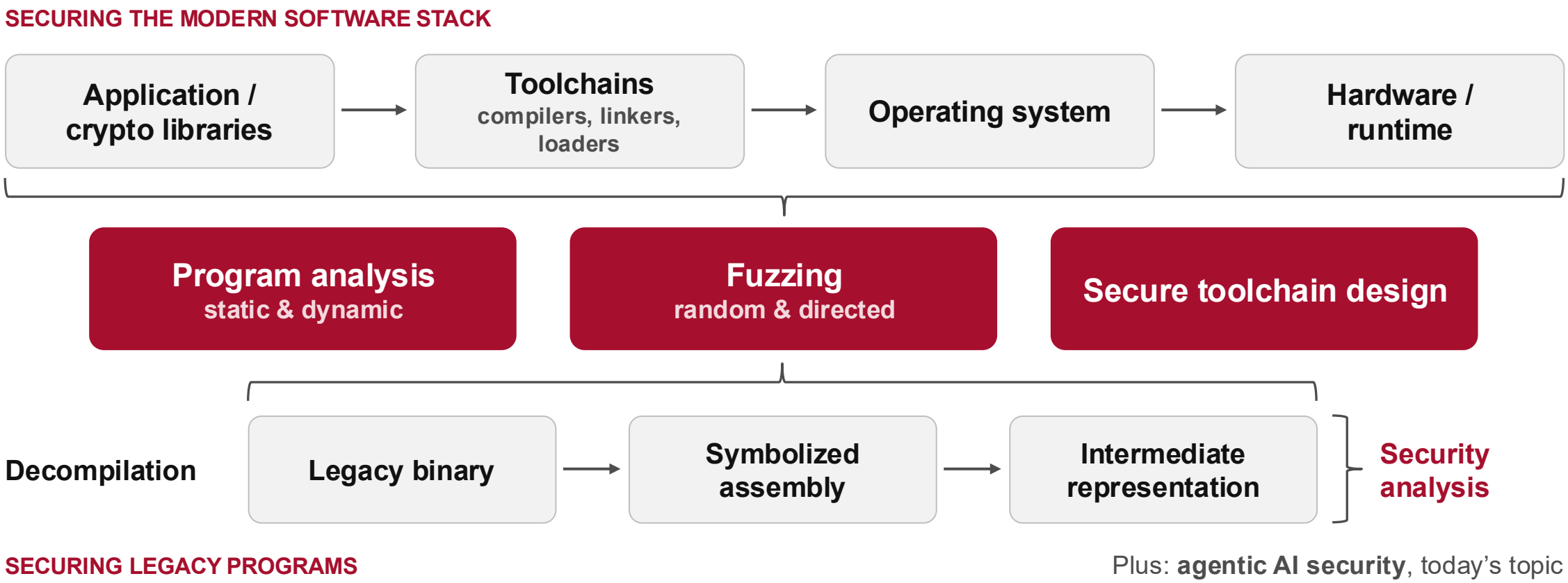
NITK Surathkal	Purdue University	Radboud University	Virginia Tech	Washington State University
B.Tech., Computer Engineering	M.S.	Ph.D., research at MPI-SP, Germany	Postdoctoral Associate	Assistant Professor

WHAT I WORK ON

- Systems & software security
- Program analysis
- Fuzzing
- Agentic AI security
- Formal methods for security

Published at **IEEE S&P, ACM CCS** and **PLDI**. Today's talk joins two of these threads: leaks in cryptographic code, and keeping AI agents safe.

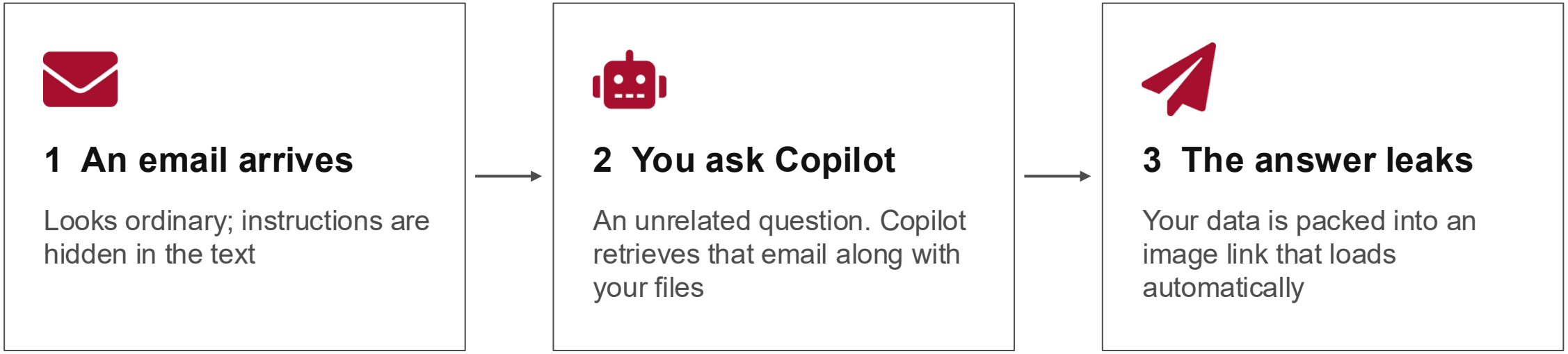
My research: securing the software stack



Plus: **agentic AI security**, today's topic

A real vulnerability · Microsoft 365 Copilot · 2025

An email nobody opened could leak your data



No password stolen. No permission violated. No click needed.

The data simply flowed somewhere it should not. Today: how do we control where information goes, not just who may read it?

EchoLeak, CVE-2025-32711 (CVSS 9.3, Microsoft's rating) / Aim Labs (2025) / Fixed server-side by Microsoft; no known exploitation

Who may read it vs. where it may go

Access control

May this app read your profile?

Checked once, at the door

Information flow control (IFC)

Where may your data end up?

Checked wherever the data flows next



REAL WORLD · 2014–2018

Cambridge Analytica: about 270,000 people took a quiz app. Facebook's API **allowed** it to collect their friends' data too. Passing that data on broke Facebook's rules: up to 87 million people affected.

Sabelfeld & Myers (2003) / Facebook (2018) / FTC (2019)

The two-worlds test (noninterference)

World 1	World 2
public = 7	public = 7
secret = 0	secret = 1
public output = 14	public output = 14

Program: output = 2 × public. Change only the secret: if nothing an outsider can see changes, nothing leaked.



A FAMOUS ANECDOTE · 1990

“Pentagon pizza”: a Domino’s franchisee said late-night CIA orders hit a record the night before Iraq invaded Kuwait. Nobody revealed the secret, yet the outside world looked different when a crisis was on.

Goguen & Meseguer (1982) / Deseret News (16 Jan 1991)

Labels: every piece of data carries a sticker

Public ≤ Internal ≤ Secret



Information may flow up the chain, never down

Combine two pieces of data, and the result takes the higher label (the “join”)

REAL WORLD · U.S. CLASSIFIED DOCUMENTS

Every paragraph carries its own marking, and the whole document is classified at the **highest level of any paragraph** inside it. That is the join, enforced on paper.

Denning (1976) / 32 CFR §2001.21

Secrecy and trust are different questions

Confidentiality

Who could learn this?

Secrets must not reach a public page

Integrity

Who could steer this?

Untrusted text must not trigger an action



REAL WORLD · DECEMBER 2023

A user told a car dealer's ChatGPT-based chatbot to agree with anything, then got it to "agree" to sell a 2024 Chevy Tahoe for $1. **Nothing secret leaked; untrusted text steered the system.** Public is not the same as trustworthy.

Sabelfeld & Myers (2003) / GM Authority (Dec 2023)

Quick poll

Does this program leak the secret?

```
out = 0
if secret:
    out = 1
print(out)
```

Vote: safe or unsafe?

Hint: no line copies secret into out.



YOU HAVE SEEN THIS BUG

“No account with that email” vs. “Wrong password” reveals who is registered, **through which branch ran, not through the data.** OWASP recommends one generic message (and equal timing) for both.

Sabelfeld & Myers (2003)

Poll answer: unsafe. Here is why

Labels must follow data and decisions

Explicit flow: data is copied

total = salary + bonus → Secret

A photocopy of a secret memo is still secret

Implicit flow: data steers a choice

The offer you get depends on salary

So the offer is Secret too: the “pc” label is the label of the branch you are in



REAL WORLD · 2012

A retailer’s pregnancy-prediction score decided which coupons customers were mailed. **The coupons themselves revealed the prediction** to anyone who saw the mail.

Sabelfeld & Myers (2003) / Duhigg, NYT Magazine (2012)

Releasing and trusting, on purpose

Declassification

Deliberately release part of a secret

e.g., publish an average, not every record



REAL WORLD · 2018

Strava's global heatmap published "only aggregate" workouts. **It still outlined military bases** and patrol routes.

Endorsement

Deliberately trust part of untrusted input

e.g., trust a bank account once verified



REAL WORLD · 2025

Business email compromise: a well-formed account number in an email is not proof the vendor sent it. **$3.05B** in losses reported to the FBI in 2025.

Sabelfeld & Myers (2003) / Strava heatmap (2018) / FBI IC3 (2025)

Enforce in the compiler, or in the OS

Jif: in the language

Labels in Java types; leaky code won't compile

Built Civitas, a research voting system (2008)

Flume: in the operating system

Labels on processes; every file, pipe, socket checked

Ran the MoinMoin wiki with per-user privacy

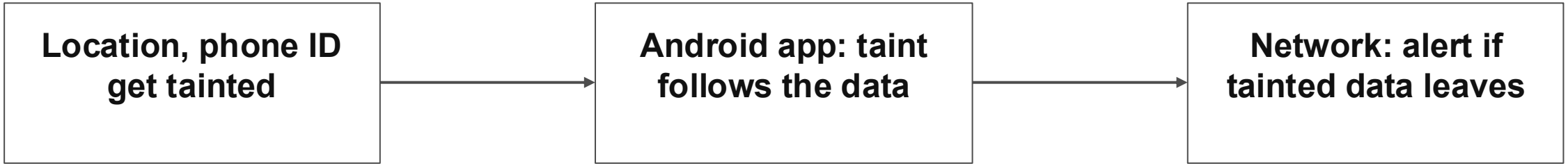


ANALOGY

A cleanroom: once a process has touched secret data, it may only write to places at least as secret, unless it holds the privilege to declassify.

Myers (1999), Jif / Krohn et al. (2007), Flume / Clarkson, Chong & Myers (2008)

TaintDroid: catching apps that leak your location



15 of 30 popular apps sent your location to advertising servers (2010 study)

REAL WORLD · 2013

The FTC settled with the maker of “Brightest Flashlight Free”, **a flashlight app that sent users’ precise location and device ID** to third parties, including ad networks.

Enck et al. (OSDI 2010) / FTC (2013)

SQL injection: when data becomes code

```
# Keep query structure separate from values
db.execute(
  "SELECT title FROM papers WHERE author = ?",
  (author_name,))
```

Integrity view: untrusted input may fill in a value, never change the query's shape.

Parameter binding enforces exactly that.

Hold that thought: prompt injection (instructions hidden in text an AI reads) is the same bug, but English has no parameter binding.



REAL WORLD · 2023

MOVEit Transfer: one SQL injection flaw (CVE-2023-34362), mass-exploited by the Cl0p gang, hit more than **2,700 organizations and 95 million people**.

Chong et al. (2007) / OWASP / CISA AA23-158A / Emsisoft (2024)

Constant time: secrets must not steer timing

Secret may influence

The values computed

The final, intended output

Secret must not influence

Which branch runs

Which memory is touched (cache timing)



REAL WORLD · 1996

Paul Kocher showed that **timing** RSA and Diffie-Hellman operations can reveal the private key, like a safecracker listening for clicks.

Kocher (CRYPTO 1996) / Kolosick et al. (PLDI 2025)

Early exit turns a password check into Wordle

```
def check(guess, secret):
    for g, s in zip(guess, secret):
        if g != s:
            return False  # stops early
    return True
```

Brute force: up to **26^4 = 456,976** guesses.
Watching the steps: at most **4 × 26 = 104**.

Secret = ABCD. Steps before the check returns:

Guess	Early exit	Full scan
ZXXX	1	4
AXXX	2	4
ABXX	3	4
ABCX	4	4



REAL WORLD · 2007

Xbox 360: the boot loader compared a hash byte by byte and stopped early. Hackers guessed it one byte at a time, enough to load an older, exploitable kernel.

Free60 Project, Xbox 360 timing attack (2007)

Robust constant-time cryptography



A library's guarantee must survive the application it is linked into

REAL WORLD · 2014

Heartbleed: OpenSSL's cryptography was fine, but a missing bounds check let anyone read up to 64 KB of server memory per request. **Researchers recovered private TLS keys that way.**

A perfect vault door doesn't help if the wall has a window.

Kolosick et al. (PLDI 2025) / CVE-2014-0160 / Cloudflare (2014)

Three ways the surrounding app can leak a key

What the app can do	What it means	Example
Read gadget	A memory-safety bug lets app code read memory it shouldn't	Heartbleed (2014)
Speculative read gadget	The CPU runs ahead, touches a secret, and leaves traces in the cache	Spectre (2018)
Concurrent observation	Another app thread watches the library's memory while it runs	Threads sharing memory

RoboCop, our toolchain, adds only the protections the surrounding application needs: overhead generally under 2% for read gadgets and under 4% for speculative ones. Write gadgets (control-flow hijacking) are out of scope.

Kolosick et al. (PLDI 2025) / Kocher et al., Spectre (IEEE S&P 2019)

Agents bring IFC back: the lethal trifecta

Private data

your email, files, repositories

IFC: a confidential source

+

Untrusted content

web pages, emails, issues

IFC: a low-integrity source

+

A way out

links, images, emails, PRs

IFC: a public sink

Agent = an AI model (LLM) that reads content and calls tools for you. **Give one all three,** and an attacker's text can move your data out.

SLACK AI · 2024

Instructions in a public channel → a link that leaks a private-channel secret when clicked

GITHUB MCP · 2025

A malicious public issue → an agent using GitHub's MCP tool connector copied private-repo data into a public pull request

M365 COPILOT · 2025

One incoming email → zero-click exfiltration of internal data (EchoLeak)

PERPLEXITY COMET · 2025

A hidden Reddit comment → the browser agent posted the user's email and one-time code

Willison (2025) / PromptArmor (2024) / Invariant Labs (2025) / Aim Labs (2025) / Brave (2025)

Indirect prompt injection



Content the agent reads can steer actions it takes with your authority

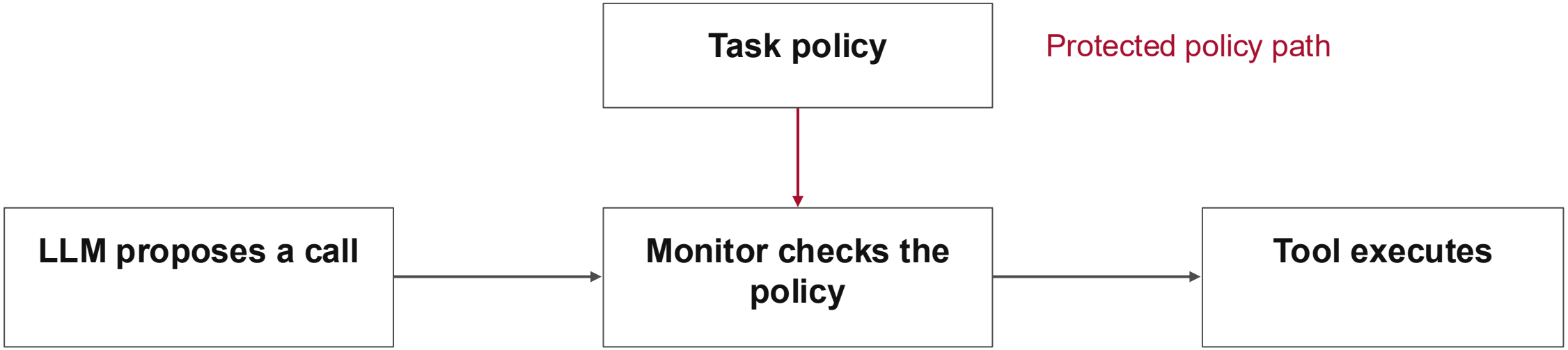
Same bug as SQL injection: instructions and data share one channel.

DIRECT VS. INDIRECT

The $1 Tahoe was **direct** injection: the attacker typed into the chat. **Indirect** is worse: the attacker only needs the agent to read something. Greshake et al. showed it against Bing Chat in 2023; it is the root of every 2024–25 incident on the previous slide.

Greshake et al. (AISec 2023) / GM Authority (Dec 2023)

Check the action before it happens



Policy = the developer's rules for allowed actions, kept where the model can't edit them

ANALOGY

A pharmacist checks the prescription before handing anything over, however convincing the story. The model can be fooled; the monitor still has a clear job.

FIDES: handles instead of raw text



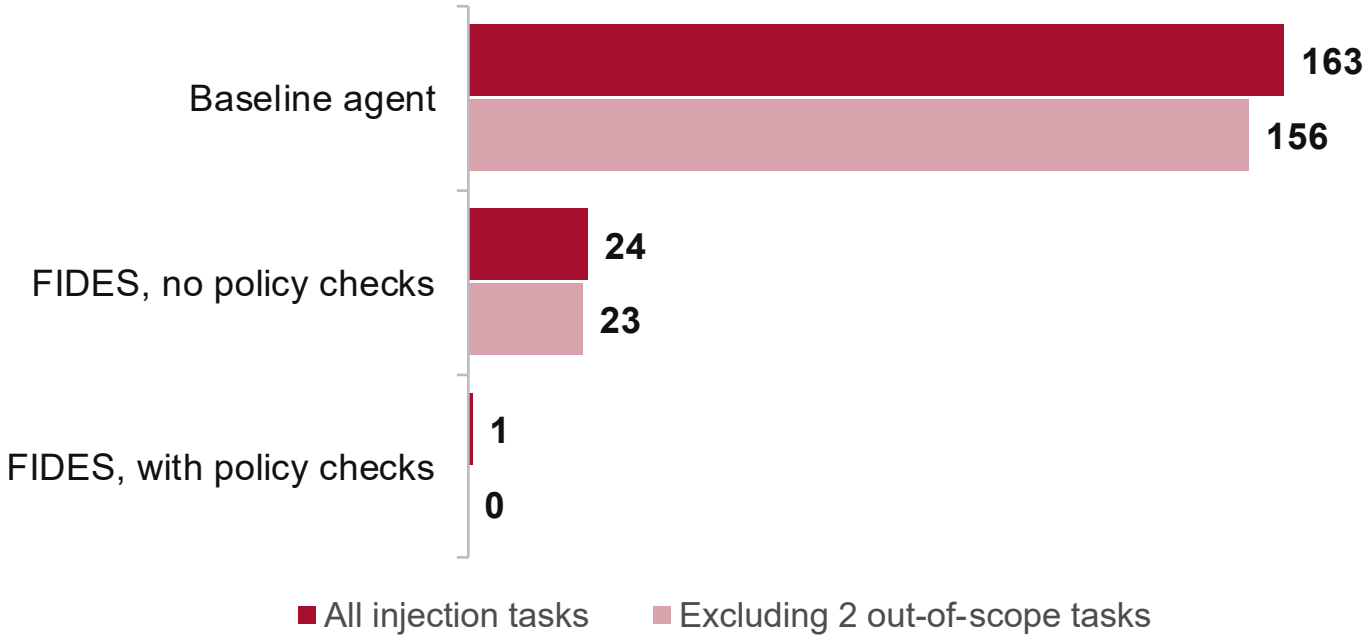
The planner (the LLM choosing tool calls) works with handles like “email #3”, so untrusted text can’t decide which tools run or where data goes

BACK TO THE OPENING STORY

EchoLeak through this lens: the email is untrusted, your files are confidential, and the auto-loaded image link is a public sink. **A flow check on that sink refuses the leak**, provided the reply renderer is a checked sink (FIDES as evaluated checks tool calls only). Related designs: Dual LLM pattern (2023), CaMeL (2025).

Costa et al. (NeurIPS 2026; arXiv v2, 2025) / Debenedetti et al. (SaTML 2026), CaMeL

FIDES in numbers



163 → 1

successful prompt injections, GPT-4o on AgentDojo (agent tasks with planted attacks)

The study also checks that legitimate tasks still get done: blocking everything is not a win.

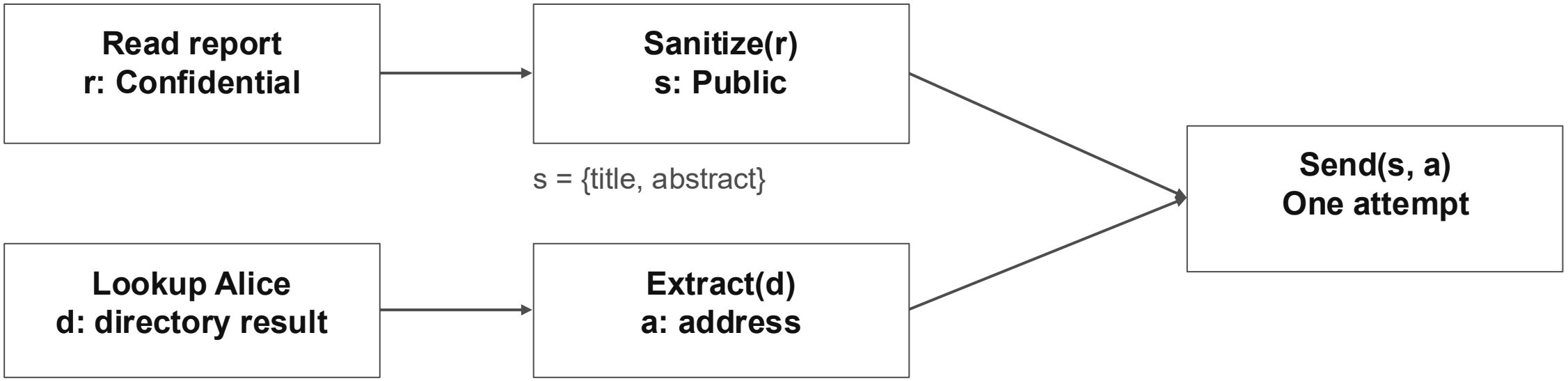
GPT-4o, AgentDojo “tool knowledge” attack, 949 attack pairings (Table 1, v2). Lighter bars leave out two injection tasks the authors do not count as attacks on FIDES: one only alters the final text, one leaks no data.

Costa et al. (NeurIPS 2026; arXiv v2, 2025), Table 1 / Debenedetti et al. (NeurIPS 2024), AgentDojo

Beyond labels: from where, to whom, how often

Example task: send only the report's public fields to Alice, exactly once

r = {title, abstract, private_notes}



Shivakumar et al. (2026), v2 · Sections 3–9

Four checks before every Send

Check	In this task	Same check in accounts payable
What is sent? classification	The body is labeled Public	Only the approved invoice goes out
Where did it come from? origin	The body was made by Sanitize	It came through the official system
Who receives it? destination	Alice, from the directory lookup	Pay the account on file, not one from an email
Any budget left? budget	User and app both have a send left	Never pay the same invoice twice

A human “OK” can add a one-time permit, but it never overrides a failed check

Shivakumar et al. (2026), v2 · Sections 3–9

Spot the problem: which sends are rejected?

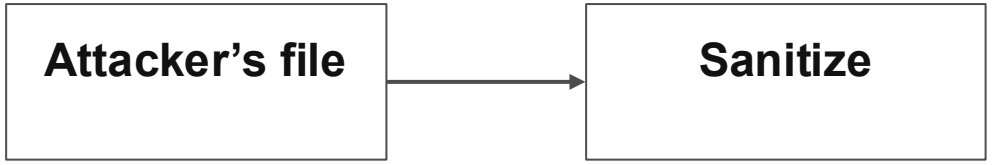
Proposed send	Decision	Reason
Public Sanitize result to Alice	Allow	All four checks pass
Raw confidential report to Alice	Reject	What is sent
Public body made by another tool	Reject	Where it came from
Approved body to another address	Reject	Who receives it
Retry after an attempted send failed	Reject	Budget already spent

An approval does not override a failed policy check

Shivakumar et al. (2026), v2 · Sections 3–9

A missing rule can approve the wrong document

What the rule checks



Output is Public and came from Sanitize

Passes the rule

What it should also check

Sanitize’s input must come from Read(report)

Rejects the attacker’s file

Lesson: check what goes into a tool, not just what comes out

ANALOGY

A notary’s stamp proves who signed, not which pages were stapled behind the signature page. The policy must say which report Sanitize may process.

Shivakumar et al. (2026), v2 · Sections 3–9

Checking policy updates before deployment

Results on 51 policy edits we wrote

21 / 21 compatible updates certified

0 / 30 breaking updates wrongly certified

Certified = Lean-checked proof that old workflows still work

What the guarantee assumes

One call at a time; services answer the same under old and new policy

Trusted: Python translation, dependencies, tool behavior

Safe = forbidden actions stay forbidden (a separate question)

Compatible ≠ safe: raising the send budget keeps every old workflow, and also allows a second send

Shivakumar et al. (2026), v2 · Sections 3–9

Same idea, four different boundaries

Setting	What is tracked	Where it is enforced	Failure it targets
Jif / Flume / TaintDroid	Values, processes, runtime taint	Compiler, OS, Android runtime	Programs leaking secrets, e.g. location to ad servers
Constant-time crypto	Which branches and addresses a secret touches	Code and its application context	Timing leaks, Heartbleed-style reads
FIDES	Labeled content, what the planner sees	Planner and tool policies	Injected text steering tool calls
Agenra	Origins, recipients, spent budget	Every call, before it runs	Wrong body, wrong recipient, double send

Myers (1999), Jif / Krohn et al. (2007) / Enck et al. (2010) / Kolosick et al. (PLDI 2025) / Costa et al. (2025), v2 / Shivakumar et al. (2026), v2

Open problems, and good first projects

Program analysis

Track implicit flows automatically

Compare what explicit vs. implicit tracking catches

Agent policy

Express recipient and budget rules

Re-check workflows after every policy edit

Evaluation

Count attacks blocked and useful tasks completed

Cover leaks via final replies, not just tool calls



THE BIG OPEN QUESTION

Who writes the policy? Today developers hand-write rules, per tool or per task. Deriving them safely from the user’s request is still open; early attempts (Progent, Conseca) use an LLM to draft them.

Four habits that carry over to agents

01 Name what you protect, and what an outsider can observe
Strava heatmap, login messages, pizza deliveries

02 Track data and the decisions it steers
Coupons in the mail, early-exit password checks

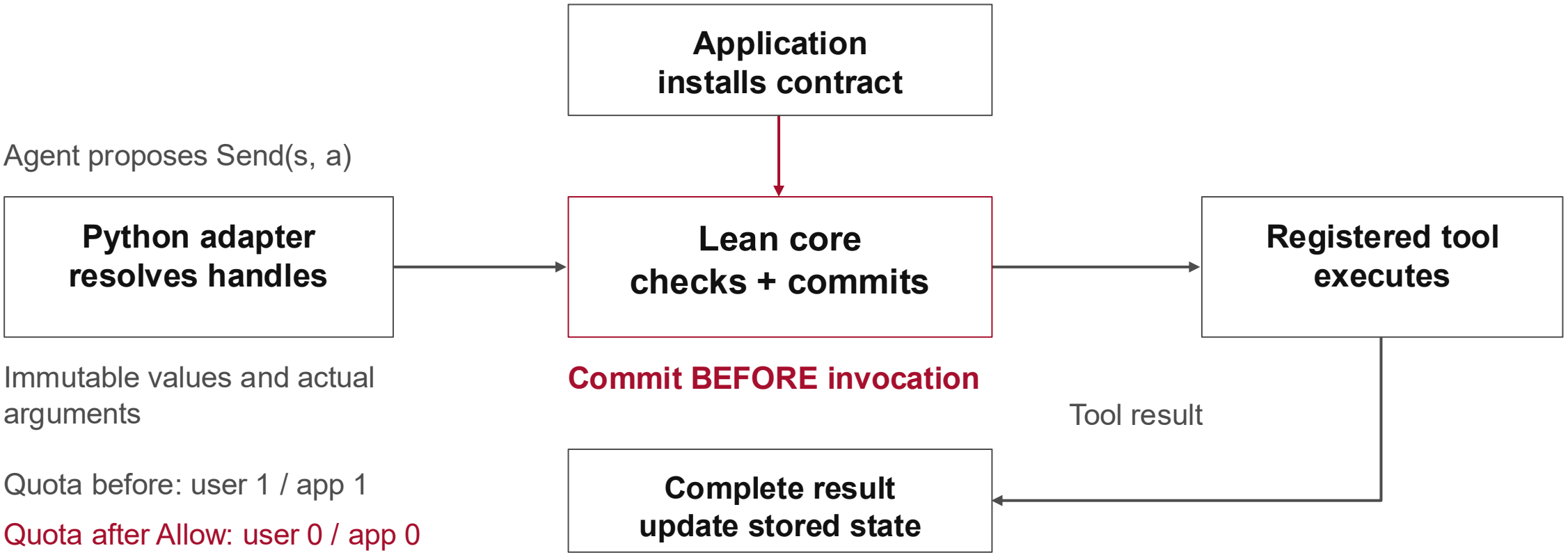
03 Write down every release and trust assumption
Heartbleed, EchoLeak

04 Measure safety and usefulness separately
FIDES, Agenra

Questions? basavesh.shivakumar@wsu.edu · basavesh.github.io

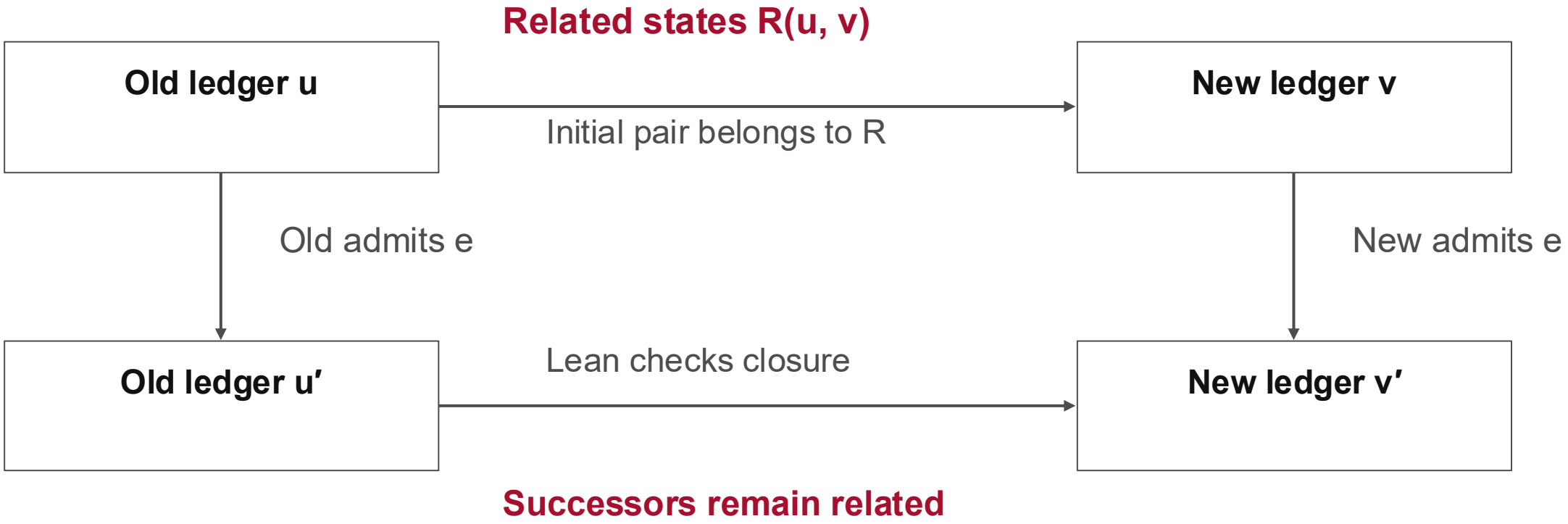
WASHINGTON STATE UNIVERSITY

Admission, dispatch and completion



Shivakumar et al. (2026), v2 · Sections 3–9

What a preservation certificate checks



Shivakumar et al. (2026), v2 · Sections 3–9

What real enforcement must guarantee

01 Complete mediation of every observable effect

02 Unforgeable labels and protected policy state

03 Automatic dependencies across branches and exceptions

04 Tool behavior consistent with declared effects

05 Explicit timing, termination and concurrency assumptions

Primary references: classical IFC

Dorothy E. Denning. A Lattice Model of Secure Information Flow. Communications of the ACM 19(5), 236–243, 1976.
https://courses.cs.duke.edu/compsci510/spring15/readings/ifc-denning76.pdf

J. A. Goguen and J. Meseguer. Security Policies and Security Models. IEEE S&P, 1982.
https://www.cs.purdue.edu/homes/ninghui/readings/AccessControl/goguen_meseguer_82.pdf

Andrei Sabelfeld and Andrew C. Myers. Language-Based Information-Flow Security. IEEE JSAC 21(1), 5–19, 2003.
https://www.cs.cornell.edu/andru/papers/jsac/sm-jsac03.pdf

Andrew C. Myers. JFlow: Practical Mostly-Static Information Flow Control. POPL, 1999. Jif language and compiler:
https://www.cs.cornell.edu/jif/ and https://www.cs.cornell.edu/jif/doc/jif-3.3.0/overview.html

Stephen Chong, K. Vikram and Andrew C. Myers. SIF: Enforcing Confidentiality and Integrity in Web Applications. USENIX Security, 2007.
https://www.cs.cornell.edu/andru/papers/usenix07-html/paper.html

Primary references: systems and cryptography

Maxwell Krohn et al. Information Flow Control for Standard OS Abstractions. SOSP, 2007. https://pdos.csail.mit.edu/~yipal/papers/flume-sosp07.pdf

William Enck et al. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. OSDI, 2010. https://www.usenix.org/conference/osdi10/taintdroid-information-flow-tracking-system-realtime-privacy-monitoring

Matthew Kolosick, Basavesh Ammanaghatta Shivakumar, Sunjay Cauligi, Marco Patrignani, Marco Vassena, Ranjit Jhala and Deian Stefan. Robust Constant-Time Cryptography. PLDI 2025 (PACMPL 9(PLDI), Article 207), doi:10.1145/3729310. Published-version author manuscript: https://squera.github.io/pdf/rct.pdf . Conference record: https://pldi25.sigplan.org/details/pldi-2025-papers/61/Robust-Constant-Time-Cryptography

OWASP. Injection Prevention Cheat Sheet, accessed 20 October 2026. https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html

Primary references: agent security and design

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz and Mario Fritz. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. AISec '23 (ACM CCS workshop), 79–90, 2023. https://arxiv.org/abs/2302.12173

Manuel Costa et al. Securing AI Agents with Information-Flow Control. NeurIPS 2026; arXiv:2505.23643v2, 3 September 2025. Sections 5 and 8, Table 1. https://arxiv.org/abs/2505.23643

Basavesh Ammanaghatta Shivakumar, Swarn Priya and Peng Gao. Agenra: A Language and Verified Core for Stateful LLM Agent Authorization. arXiv:2608.22868v2, 15 October 2026. https://arxiv.org/html/2608.22868v2

Edoardo Debenedetti et al. Defeating Prompt Injections by Design (CaMeL). IEEE SaTML 2026; arXiv:2503.18813. https://arxiv.org/abs/2503.18813

Real-world examples used in this talk

EchoLeak: Microsoft, CVE-2025-32711 advisory (June 2025); Aim Labs write-up. https://msrc.microsoft.com/update-guide/vulnerability/CVE-2025-32711

Cambridge Analytica: Facebook newsroom, 4 Apr 2018; FTC press releases, 24 Jul 2019. https://www.ftc.gov/news-events/news/press-releases/2019/07/ftc-imposes-5-billion-penalty-sweeping-new-privacy-restrictions-facebook

Pentagon pizza: “Gimme 55 large pepperonis and hold the crisis”, Deseret News, 16 Jan 1991. https://www.deseret.com/1991/1/16/18901123/gimme-55-large-pepperonis-and-hold-the-crisis/

Classification marking: 32 CFR §2001.21. https://www.ecfr.gov/current/title-32/subtitle-B/chapter-XX/part-2001/subpart-C/section-2001.21

Chevrolet dealer chatbot: GM Authority, Dec 2023. https://www.gmauthority.com/blog/2023/12/gm-dealer-chat-bot-agrees-to-sell-2024-chevy-tahoe-for-1/

OWASP Authentication Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

C. Duhigg, “How Companies Learn Your Secrets”, New York Times Magazine (online 16 Feb 2012; print 19 Feb 2012).

Strava heatmap: CNN Money, 29 Jan 2018. https://money.cnn.com/2018/01/29/technology/strava-nathan-ruser/index.html

FBI IC3, 2025 Internet Crime Report (Apr 2026). https://www.ic3.gov/AnnualReport/Reports/2025_IC3Report.pdf

Clarkson, Chong & Myers, Civitas: Toward a Secure Voting System, IEEE S&P 2008.

FTC, Brightest Flashlight settlement, 5 Dec 2013. https://www.ftc.gov/news-events/news/press-releases/2013/12/android-flashlight-app-developer-settles-ftc-charges-it-deceived-consumers

MOVEit: CISA advisory AA23-158A; Emsisoft statistics. https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-158a

P. Kocher, Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems, CRYPTO 1996.

Xbox 360 timing attack: Free60 Project. https://free60.org/Timing_Attack/

Heartbleed: https://heartbleed.com/ ; Cloudflare challenge results. https://blog.cloudflare.com/the-results-of-the-cloudflare-challenge/

Kocher et al., Spectre Attacks: Exploiting Speculative Execution, IEEE S&P 2019. https://arxiv.org/abs/1801.01203

S. Willison, The lethal trifecta for AI agents, 16 Jun 2025. https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/

PromptArmor, Slack AI data exfiltration (Aug 2024); Invariant Labs, GitHub MCP exploit (May 2025); Brave, Comet prompt injection (Aug 2025).

Debenedetti et al., Defeating Prompt Injections by Design (CaMeL), arXiv:2503.18813 (2025); S. Willison, The Dual LLM pattern (Apr 2023).