

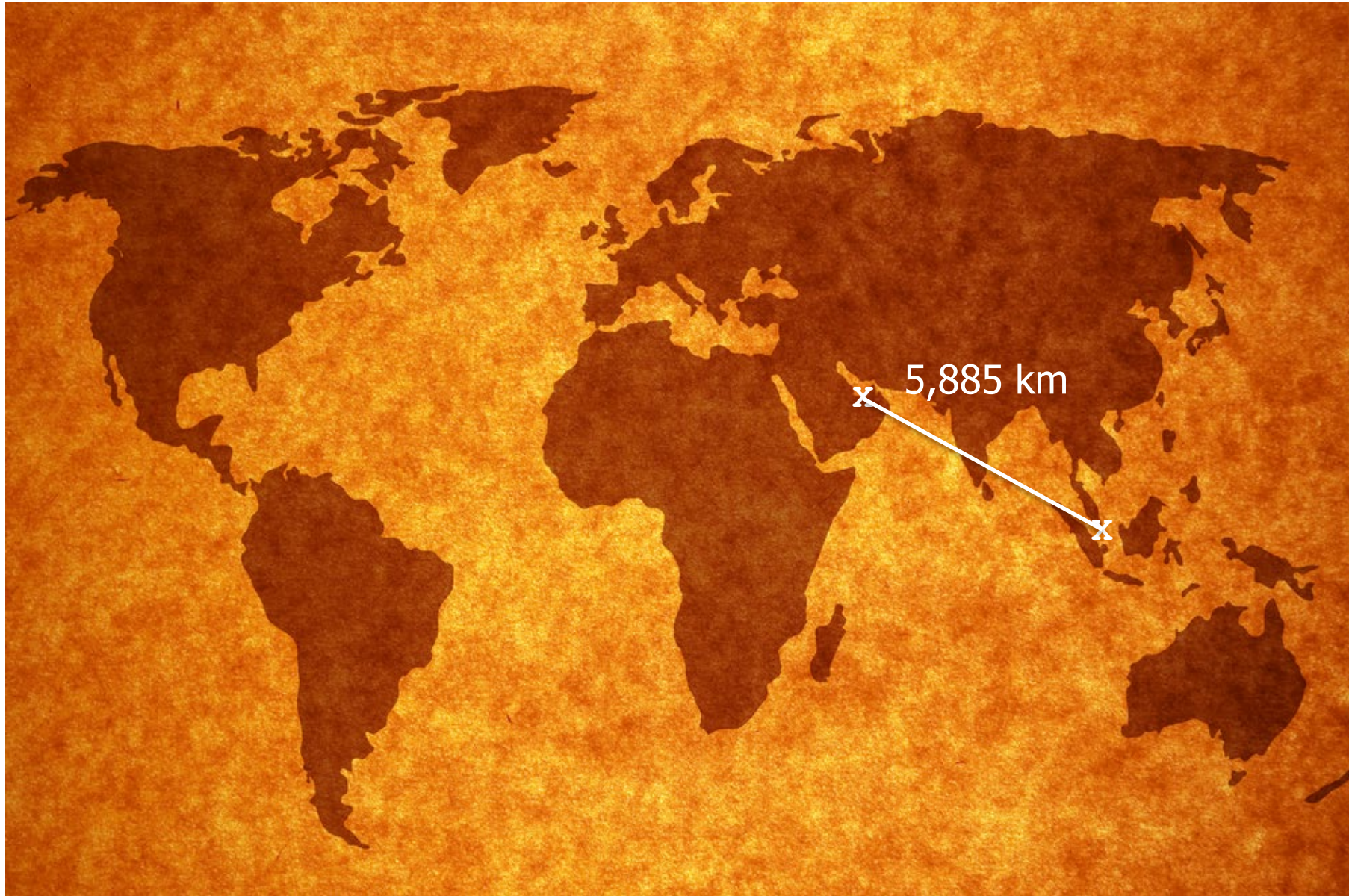


Engineering Vulnerability Discovery with Large Code Models: The TitanCA Experience

David Lo, FACM, FIEEE

*MBZUAI Symposium on Security in the Age of AI,
Abu Dhabi, UAE, Feb 2026*

Self-Introduction



Self-Introduction



Self-Introduction



Self-Introduction



Singapore Management University



- Third university in Singapore
- Number of students:
 - 8000+ (UG)
 - 1800+ (PG)
- Schools:
 - Business
 - Computing
 - Economics
 - Accountancy
 - Law
 - Social Science



Center for Research on Intelligent Software Engineering (RISE)



Elsevier JSS'21, Bibliometric Study

Table 3

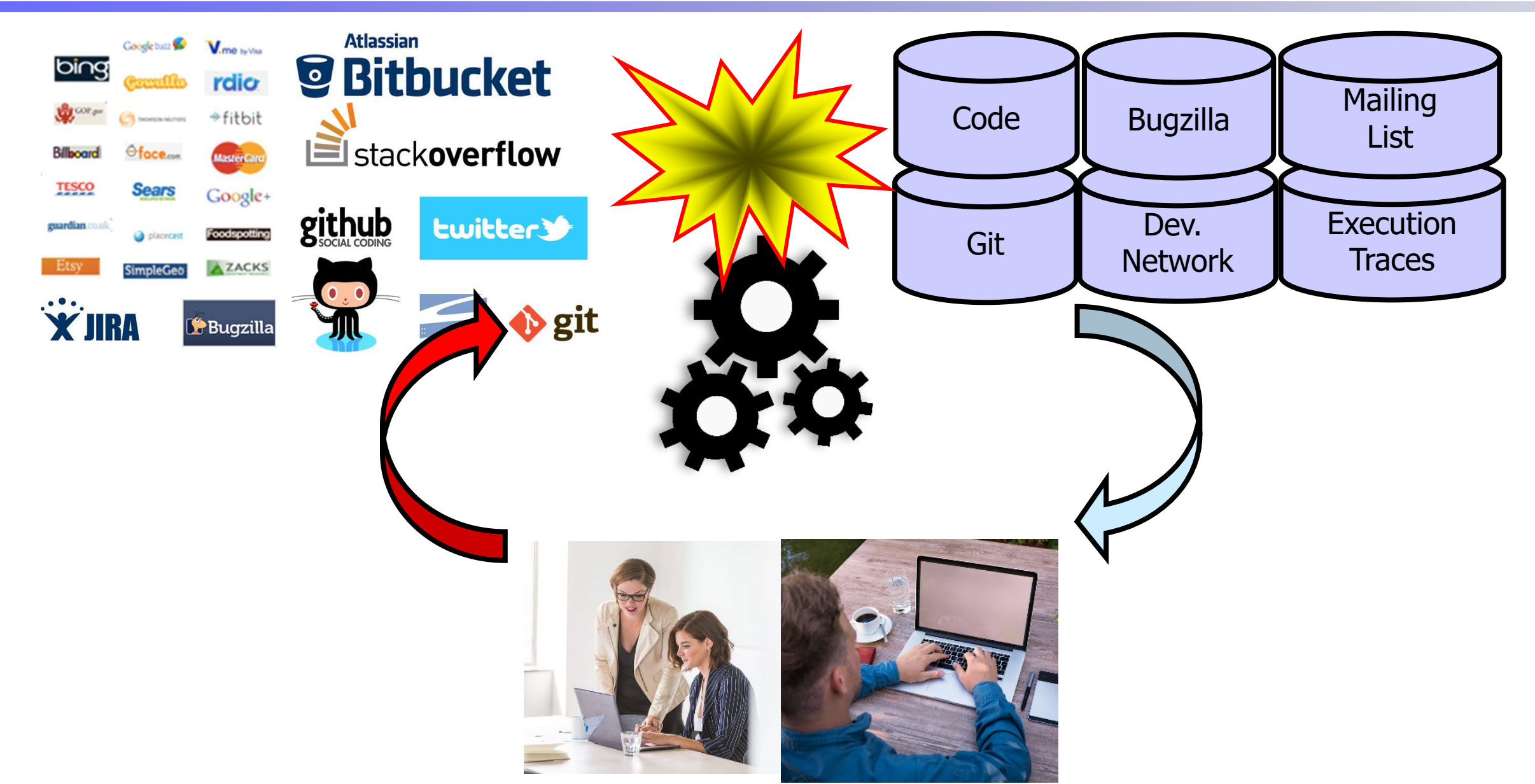
Most active institutions in software engineering

Rank	Name
1	University of California
2	Carnegie Mellon University
3	Nanjing University
4	Microsoft Research
5	Singapore Management University

CSRankings, SE, Jan 2026

1	▶ Nanjing University 🇨🇳 📊	53.5	50
2	▶ Carnegie Mellon University 🇺🇸 📊	30.3	16
3	▶ Peking University 🇨🇳 📊	28.9	21
4	▶ Zhejiang University 🇨🇳 📊	28.5	28
5	▶ Singapore Management University 🇸🇬 📊	26.5	9

AI for Software Engineering



Experience with AI4SE

SMArTIC: Towards Building an Accurate, Robust and Scalable Specification Miner

FSE'06

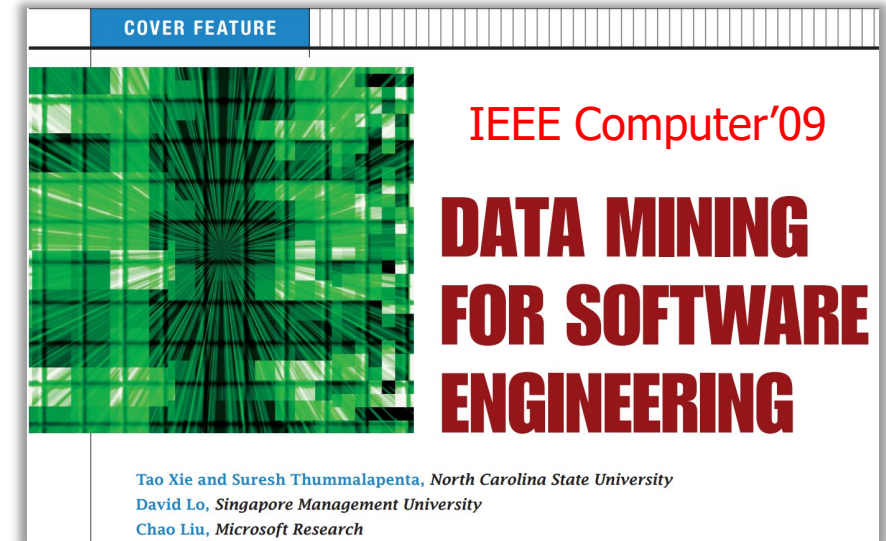
David Lo and Siau-Cheng Khoo
Department of Computer Science, National University of Singapore
{dlo,khoosc}@comp.nus.edu.sg

Efficient Mining of Iterative Patterns for Software Specification Discovery

KDD'07

David Lo and Siau-Cheng Khoo
Department of Computer Science
National University of Singapore
{dlo,khoosc}@comp.nus.edu.sg

Chao Liu
Department of Computer Science
University of Illinois-UC
chaoliu@cs.uiuc.edu



Experience with AI4SE

Classification of Software Behaviors for Failure Detection: A Discriminative Pattern Mining Approach

KDD'09

David Lo
Singapore Management University
davidlo@smu.edu.sg

Hong Cheng^{*}
Chinese University of Hong Kong
hcheng@se.cuhk.edu.hk

Jiawei Han[†]
University of Illinois at Urbana-Champaign
hanj@cs.uiuc.edu

Siau-Cheng Khoo and Chengnian Sun
National University of Singapore
{khoosc,sunch}@comp.nus.edu.sg

Test oracle generation

A Discriminative Model Approach for Accurate Duplicate Bug Report Retrieval

ICSE'10

Chengnian Sun¹, David Lo², Xiaoyin Wang³, Jing Jiang², Siau-Cheng Khoo¹

¹School of Computing, National University of Singapore

²School of Information Systems, Singapore Management University

³Key laboratory of High Confidence Software Technologies (Peking University), Ministry of Education

sunch@comp.nus.edu.sg, davidlo@smu.edu.sg, wangxy06@sei.pku.edu.cn,

jingjiang@smu.edu.sg, khoosc@comp.nus.edu.sg

Intelligent issue trackers

Tag Recommendation in Software Information Sites

MSR'13

Xin Xia^{*†}, David Lo[†], Xinyu Wang^{*}, and Bo Zhou^{*§}

^{*}College of Computer Science and Technology, Zhejiang University

[†]School of Information Systems, Singapore Management University

Intelligent crowdsourced SE

History Driven Program Repair

SANER'16

Xuan-Bach D. Le, David Lo
School of Information Systems
Singapore Management University
{dxb.le.2013,davidlo}@smu.edu.sg

Claire Le Goues
School of Computer Science
Carnegie Mellon University
clegoues@cs.cmu.edu

Intelligent program repair

*"History-driven
program repair
influence*

*our work, the overall
pipeline is similar"*

- Facebook
Engineers

Research Agenda in AI4SE

Trustworthy and Synergistic Artificial Intelligence for Software Engineering: Vision and Roadmaps

David Lo
School of Computing and Information Systems,
Singapore Management University,
Singapore
Email: davidlo@smu.edu.sg



ICSE'23 – Future of Software Engineering



Trustworthy and Synergistic AI4SE:
Vision and the Road Ahead

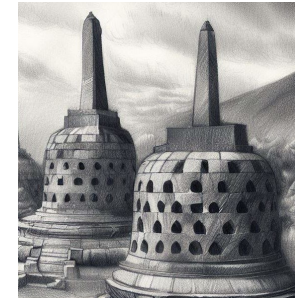


David Lo

School of
Computing and
Information Systems

AI for Software Engineering

History



Challenges



Vision



Trust Synergy



Roadmap I

Roadmap II

Call4Action

Towards Software Engineering 2.0

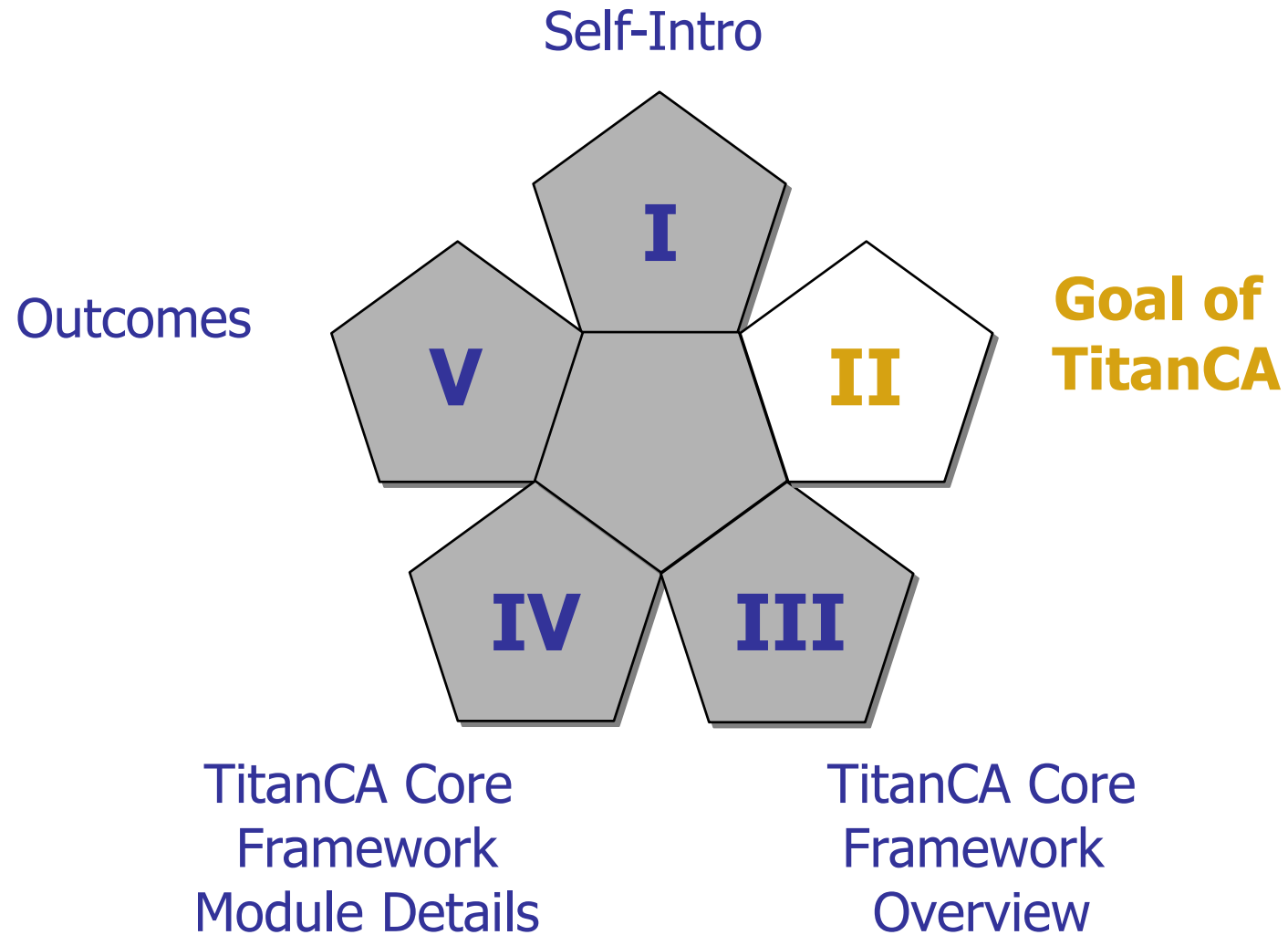
“If you want to go far, go together” – African Proverb





Engineering Vulnerability Discovery with Large Code Models: The TitanCA Experience

Talk Structure



Goal of TitanCA

- Building a localized and specialized large code model (LCM) solution for just-in-time (JIT) vulnerability discovery
 - The resulting solution would:



Generalizable and work across different programming languages (especially Java, C#, Python) and domains (especially web applications)



Have a high signal-to-noise ratio, for example, by avoiding decisions on cases with low confidence as measured against existing solutions



Work without sending data across the internet



Continually tuned with new vulnerability data from open-source channels

Goal of TitanCA

- Building a localized and specialized large code model (LCM) solution for just-in-time (JIT) vulnerability discovery
 - The resulting solution would:



Be able to **ingest vulnerability data of varying levels of quality**, while simultaneously minimizing the negative effects caused by lower quality data



Enable **a human operator** with domain knowledge to provide additional vulnerability data on which the model should be tuned on



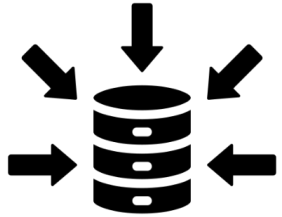
Be **refined** using characteristics of vulnerabilities in **SHIP-HATS 2.0 GitLab project repositories**



Be **integrable** with SHIP-HATS 2.0 GitLab and **security vulnerability dashboard**

Data

T1



Vulnerability
Collection

T2

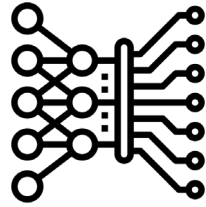


Data Quality
Audit &
Enrichment

TitanCA

Model

T3

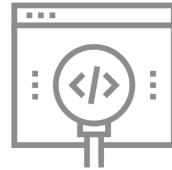


Vulnerability
Detection Models
from LCMs

T4



Model Precision &
Robustness Boost



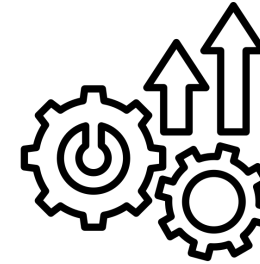
Static Analysis for
Better LCMs



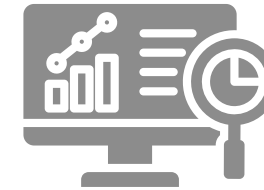
Explanation
Generation

Refinement

T5



Project
Specific
Refinement



Monitoring &
Adaptation



Adv. UI/UX &
User Studies

Within GovTech

Operation

T6



Data
Engineering



Model Efficiency
Optimization



Tooling & GitLab
Integration

T7

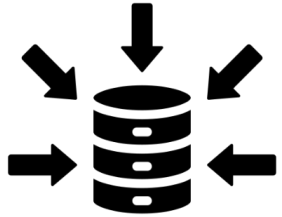


Periodic
Model
Maintenance

T8

Data

T1



Vulnerability
Collection

T2

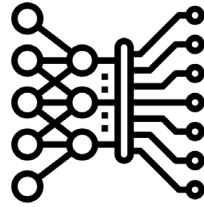


Data Quality
Audit &
Enrichment

TitanCA

Model

T3

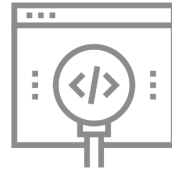


Vulnerability
Detection Models
from LCMs

T4



Model Precision &
Robustness Boost



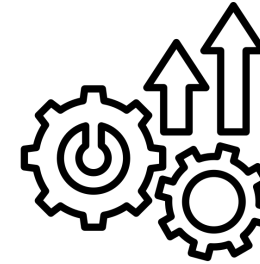
Static Analysis for
Better LCMs



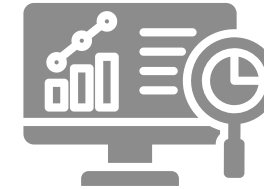
Explanation
Generation

Refinement

T5



Project
Specific
Refinement



Monitoring &
Adaptation



Adv. UI/UX &
User Studies

Within GovTech

Operation

T6



Data
Engineering



Model Efficiency
Optimization



Tooling & GitLab
Integration

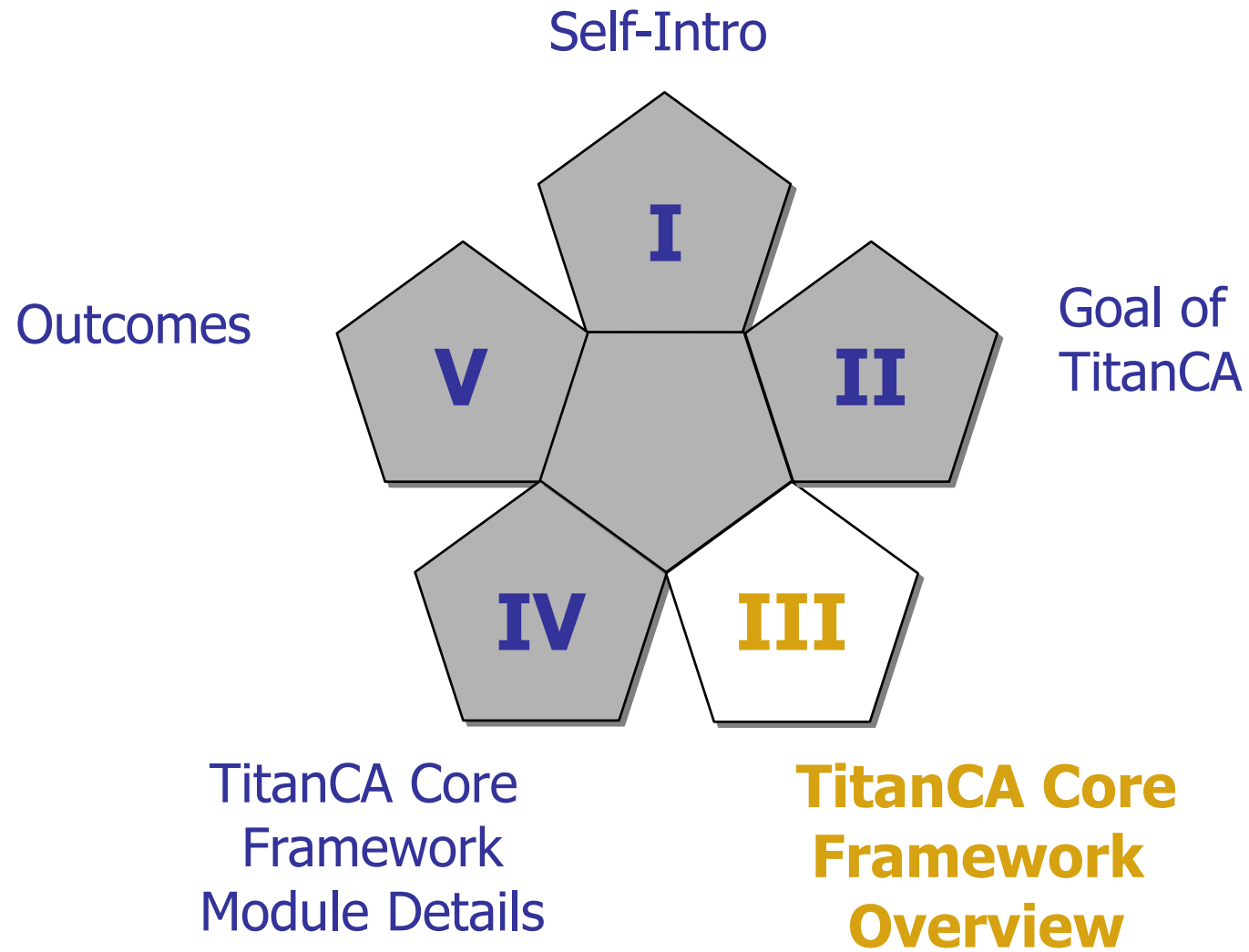
T7



Periodic
Model
Maintenance

T8

Talk Structure



TitanCA Core Operational Framework Overview

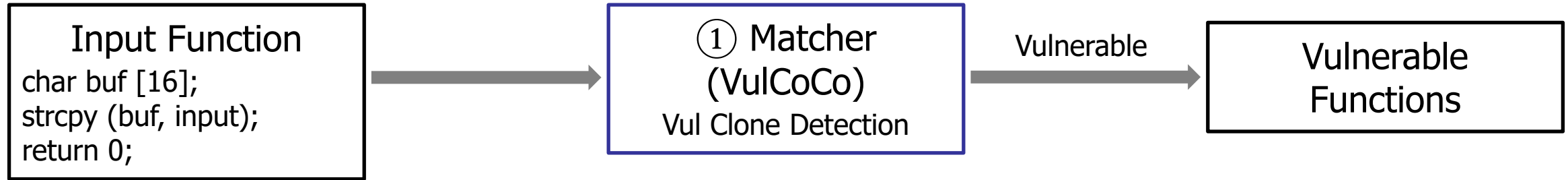
Input Function

```
char buf [16];  
strcpy (buf, input);  
return 0;
```

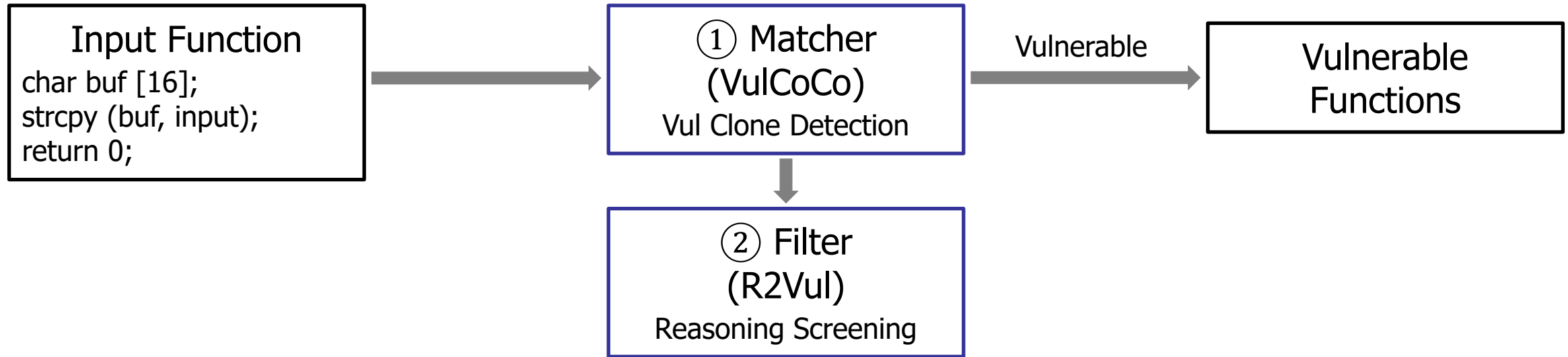
TitanCA Core Operational Framework Overview



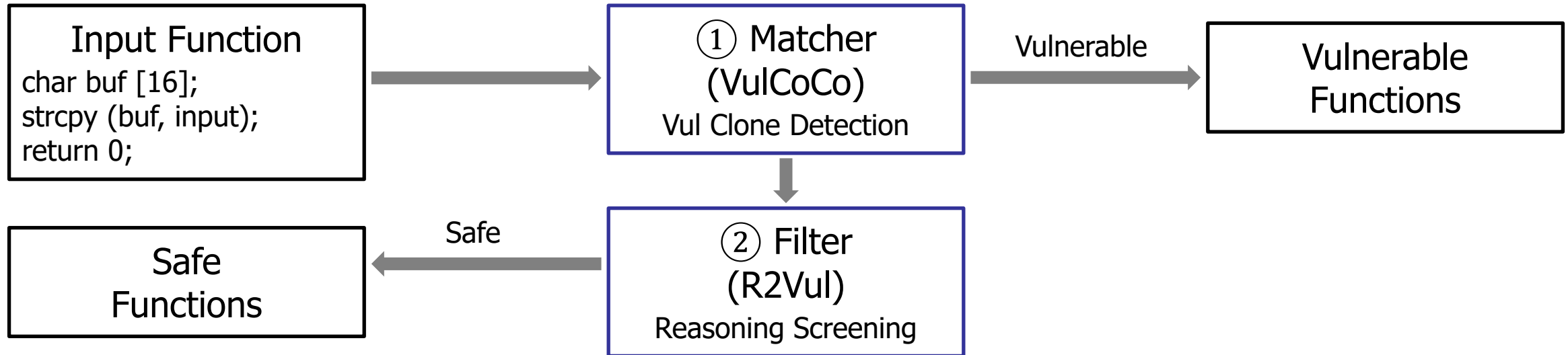
TitanCA Core Operational Framework Overview



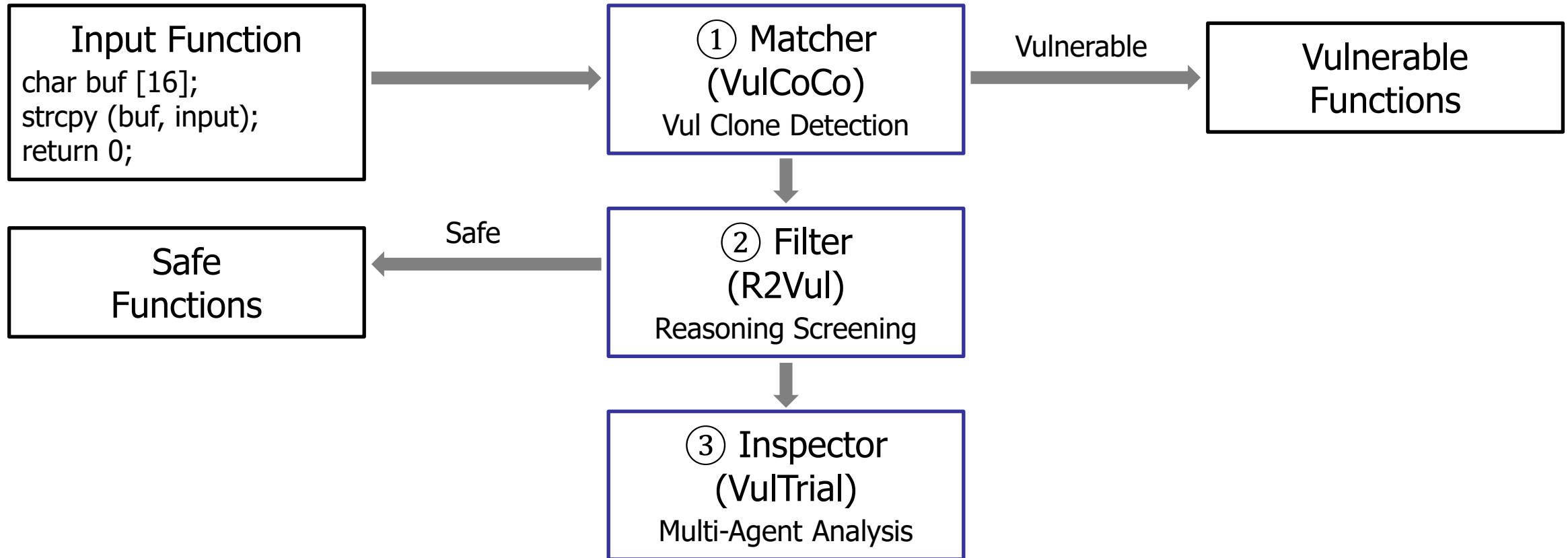
TitanCA Core Operational Framework Overview



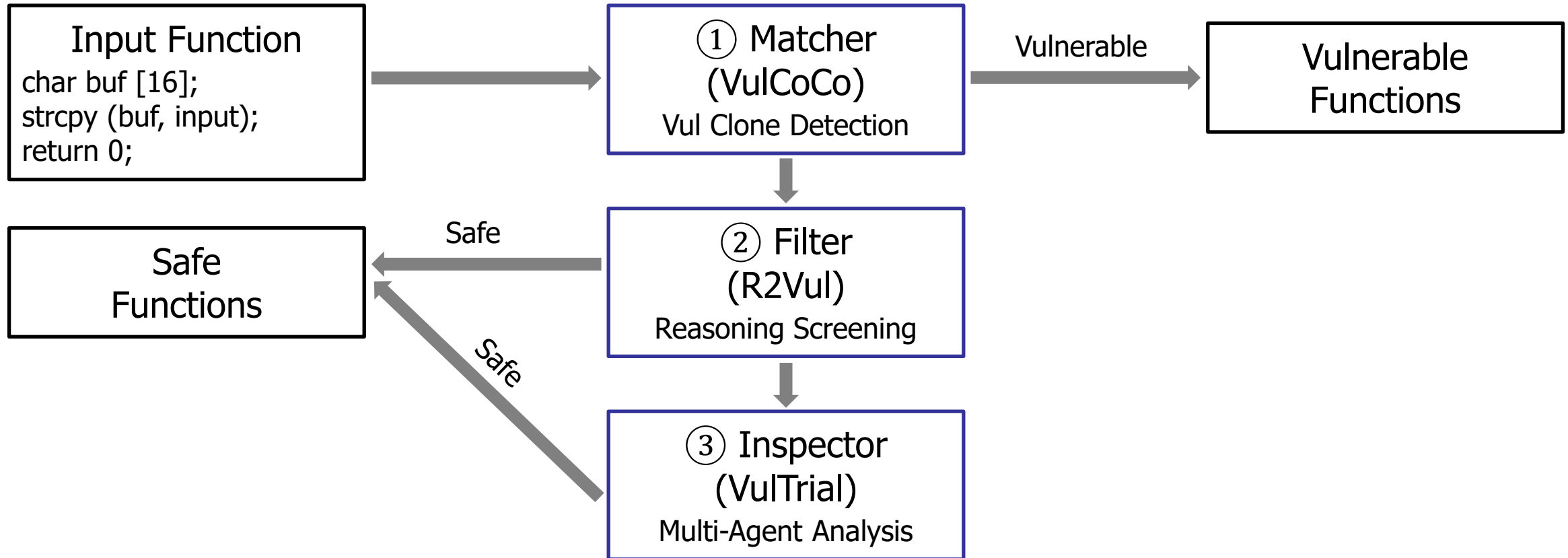
TitanCA Core Operational Framework Overview



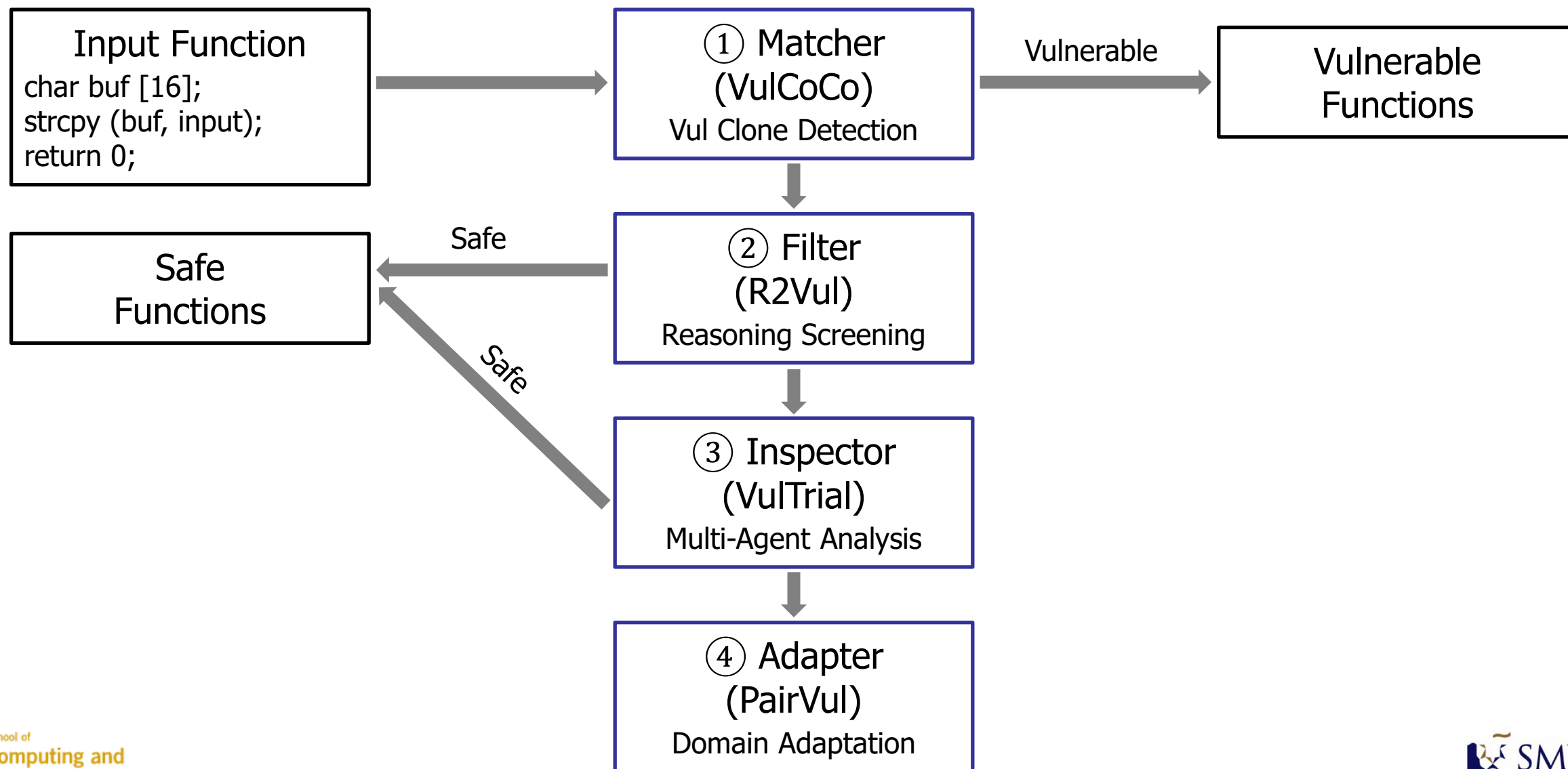
TitanCA Core Operational Framework Overview



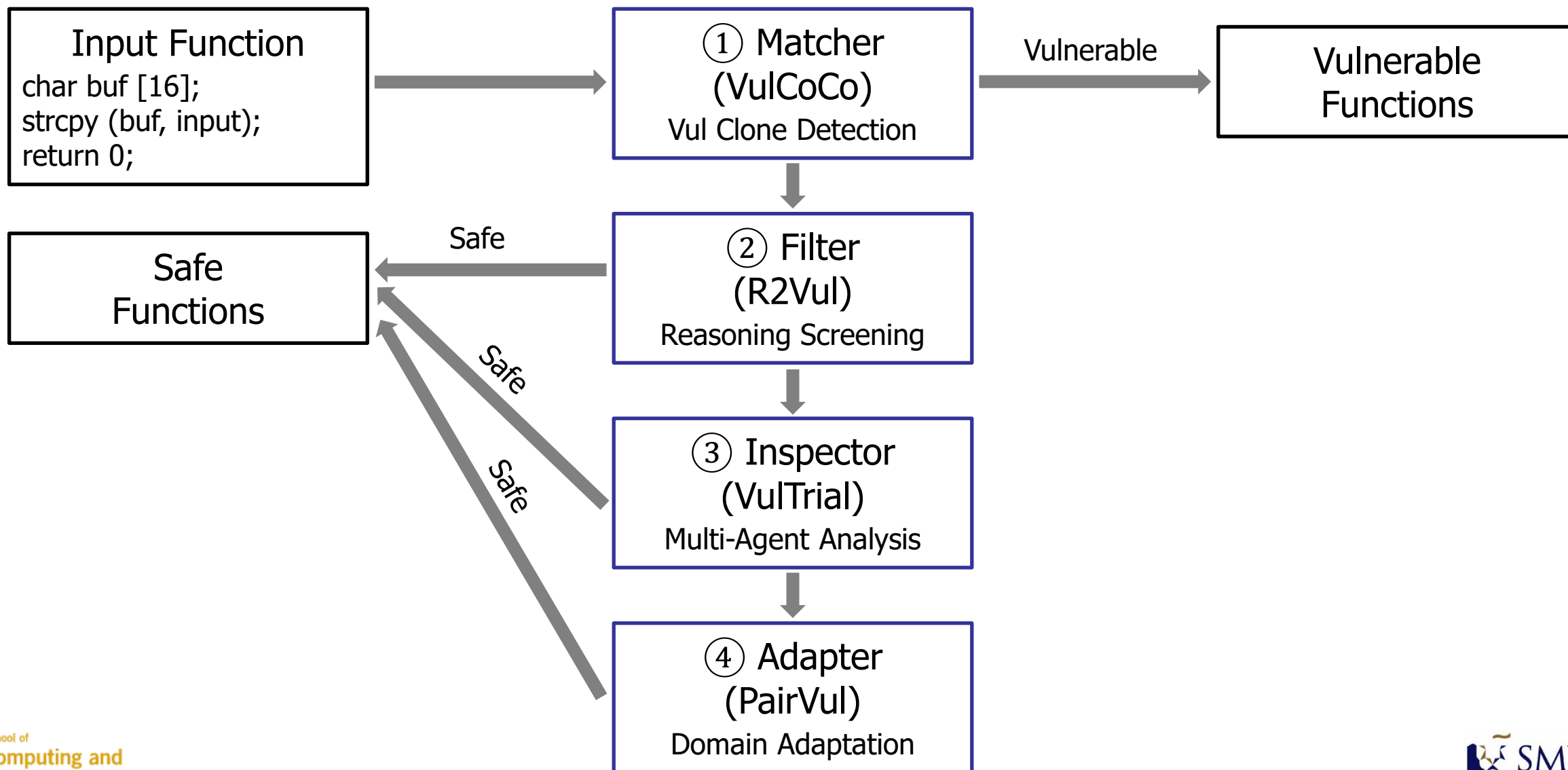
TitanCA Core Operational Framework Overview



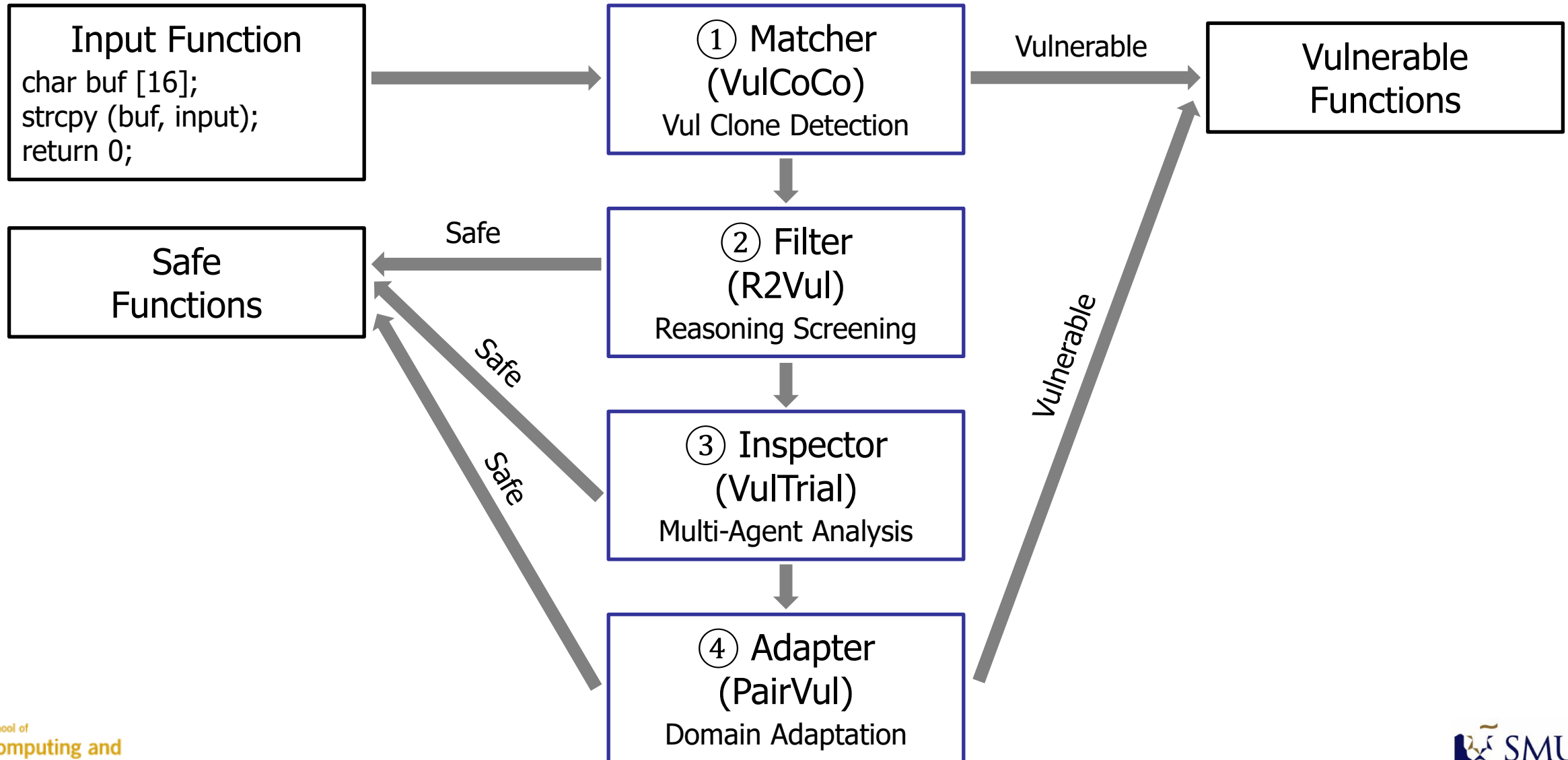
TitanCA Core Operational Framework Overview



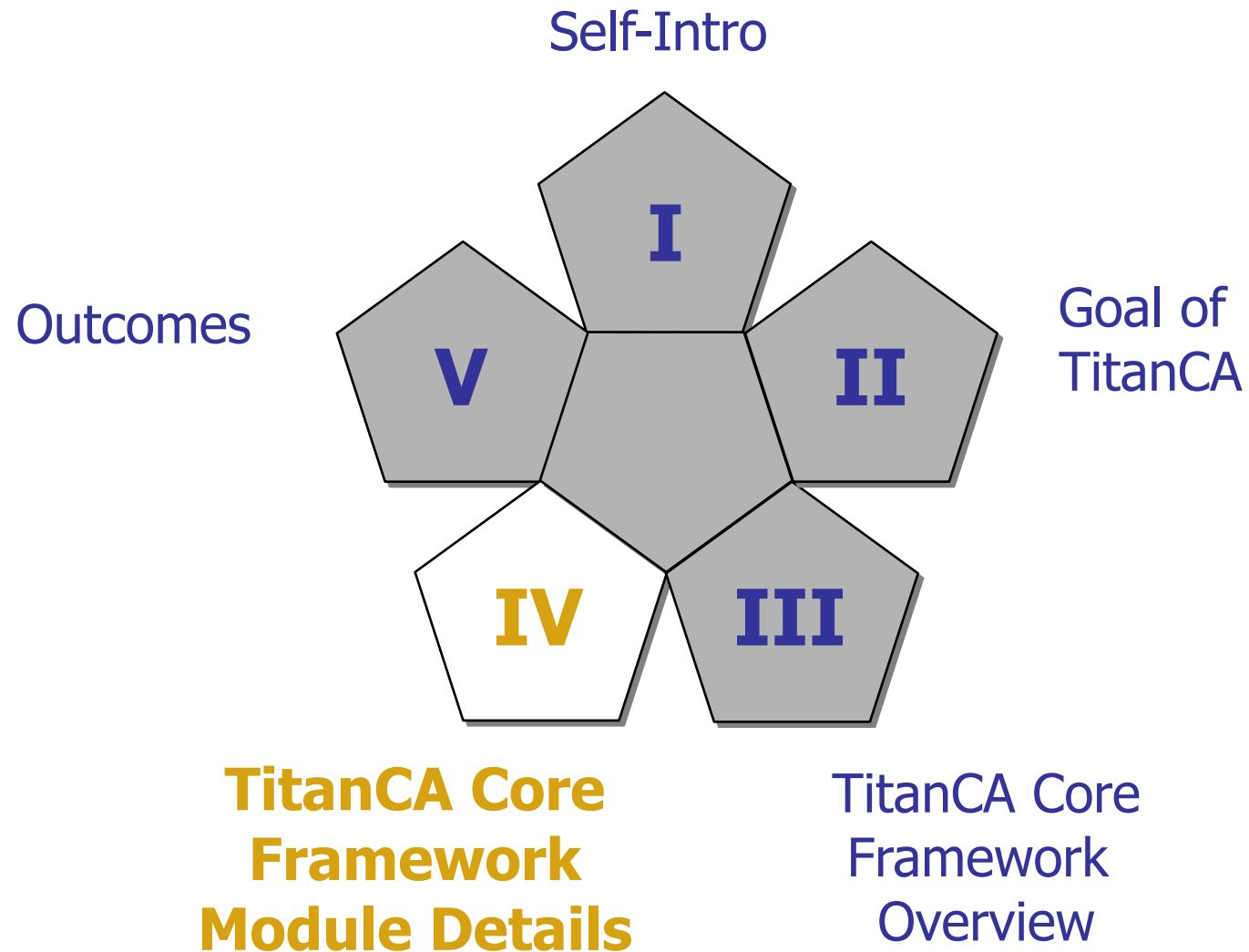
TitanCA Core Operational Framework Overview



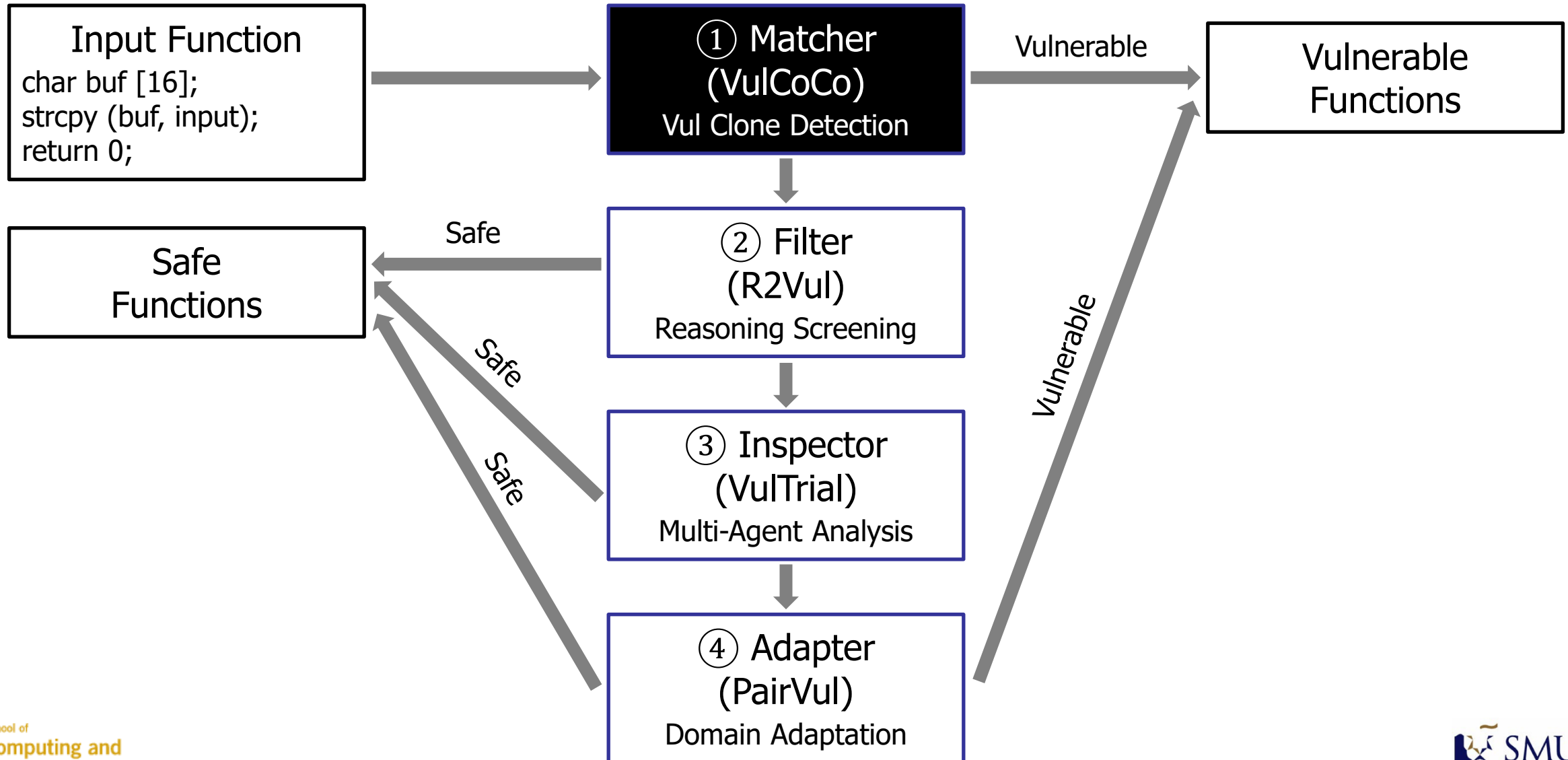
TitanCA Core Operational Framework Overview



Talk Structure



TitanCA Core Operational Framework Overview



Module Details - ① Matcher (VulCoCo)

- ① **Matcher**

- LLM-Powered Vulnerable Clone Detector

- **Goal**

- Detects whether an input function *retains the vulnerable logic* of a known vulnerable function to which it is *closely similar*

Module Details - ① Matcher (VulCoCo)

- Step 1: Embedding Generation
 - Converts existing vulnerable functions into vector representations

Existing Vul Func 1

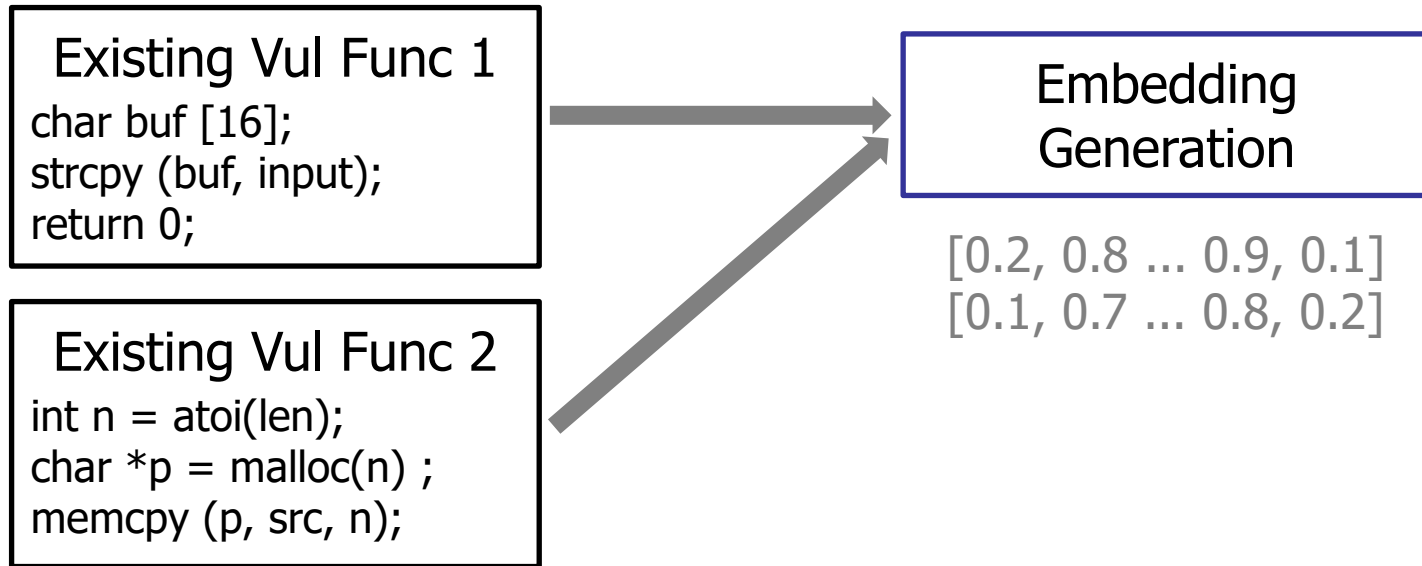
```
char buf [16];  
strcpy (buf, input);  
return 0;
```

Existing Vul Func 2

```
int n = atoi(len);  
char *p = malloc(n) ;  
memcpy (p, src, n);
```

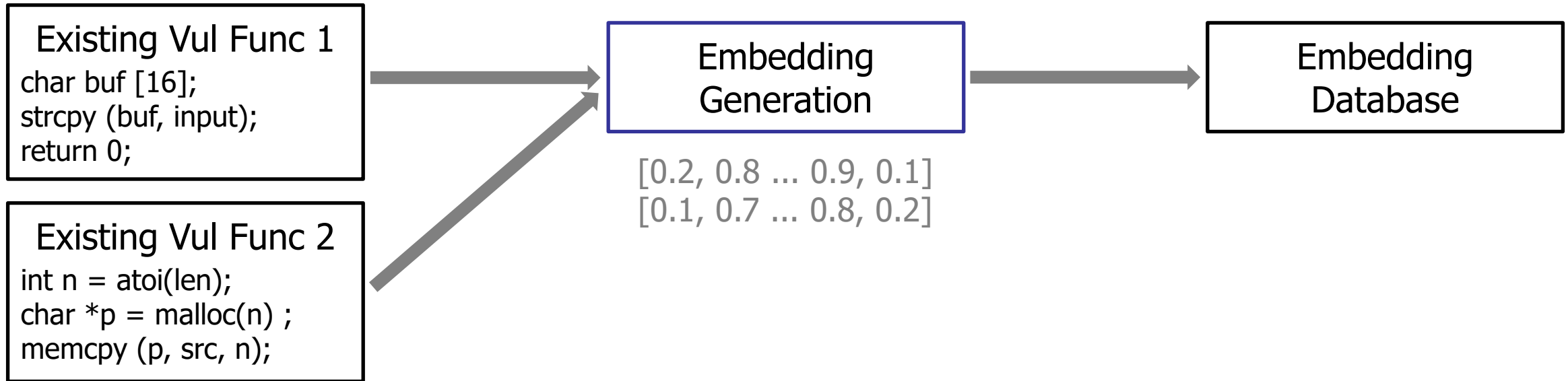
Module Details - ① Matcher (VulCoCo)

- Step 1: Embedding Generation
 - Converts existing vulnerable functions into vector representations



Module Details - ① Matcher (VulCoCo)

- Step 1: Embedding Generation
 - Converts existing vulnerable functions into vector representations



Module Details - ① Matcher (VulCoCo)

- Step 2: Screening Input Functions
 - Compares input function embeddings against known vulnerable functions

Input Function

```
char dst [10];  
strcpy (dst, src);  
return dst;
```

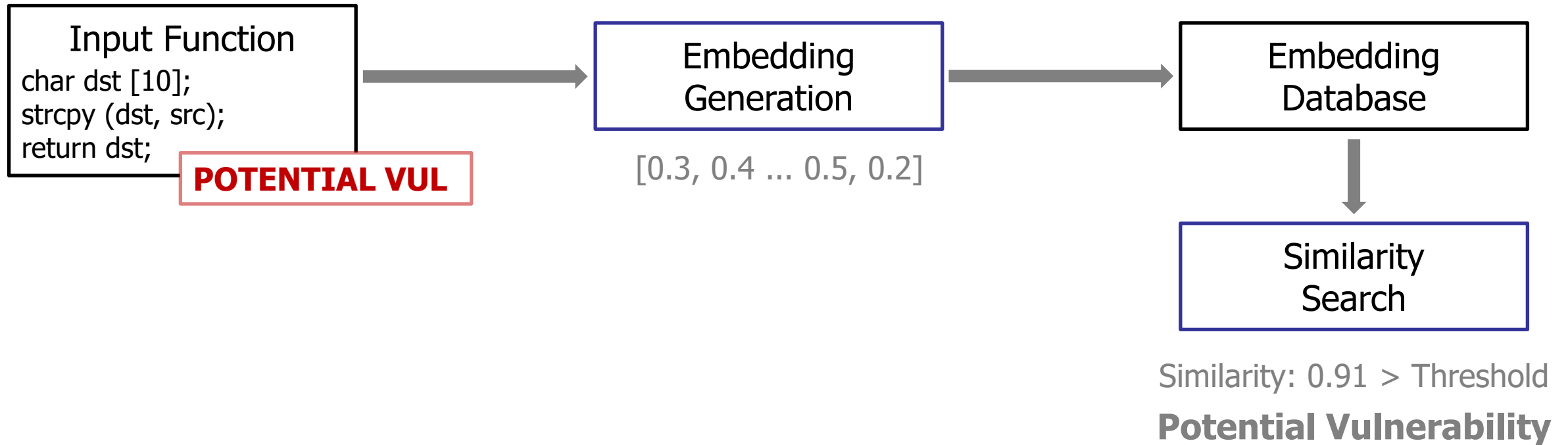
Module Details - ① Matcher (VulCoCo)

- Step 2: Screening Input Functions
 - Compares input function embeddings against known vulnerable functions



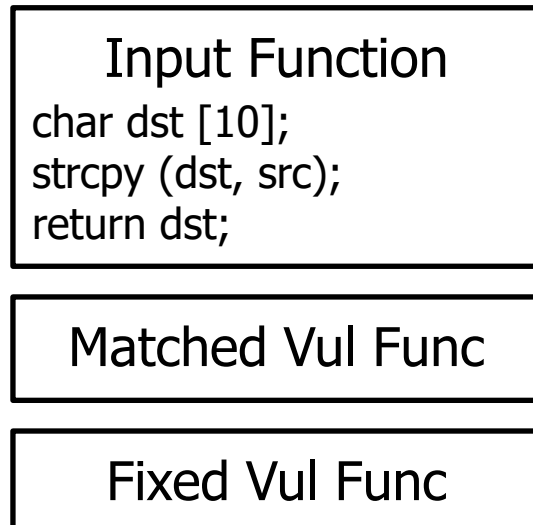
Module Details - ① Matcher (VulCoCo)

- Step 2: Screening Input Functions
 - Compares input function embeddings against known vulnerable functions



Module Details - ① Matcher (VulCoCo)

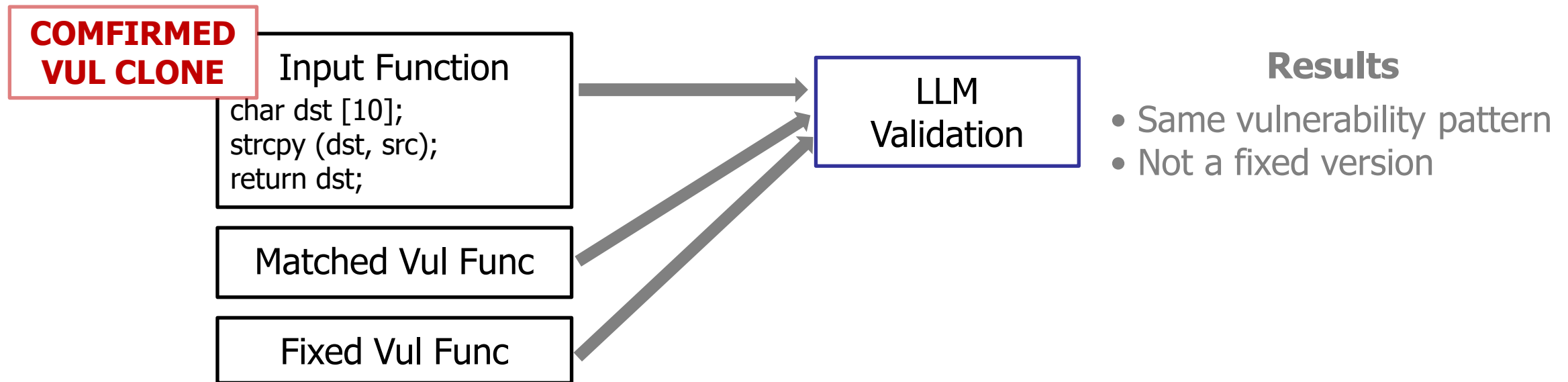
- Step 3: LLM Validation
 - Uses an LLM to validate candidate vulnerable clone matches
 - Filters out fixed or non-vulnerable clones



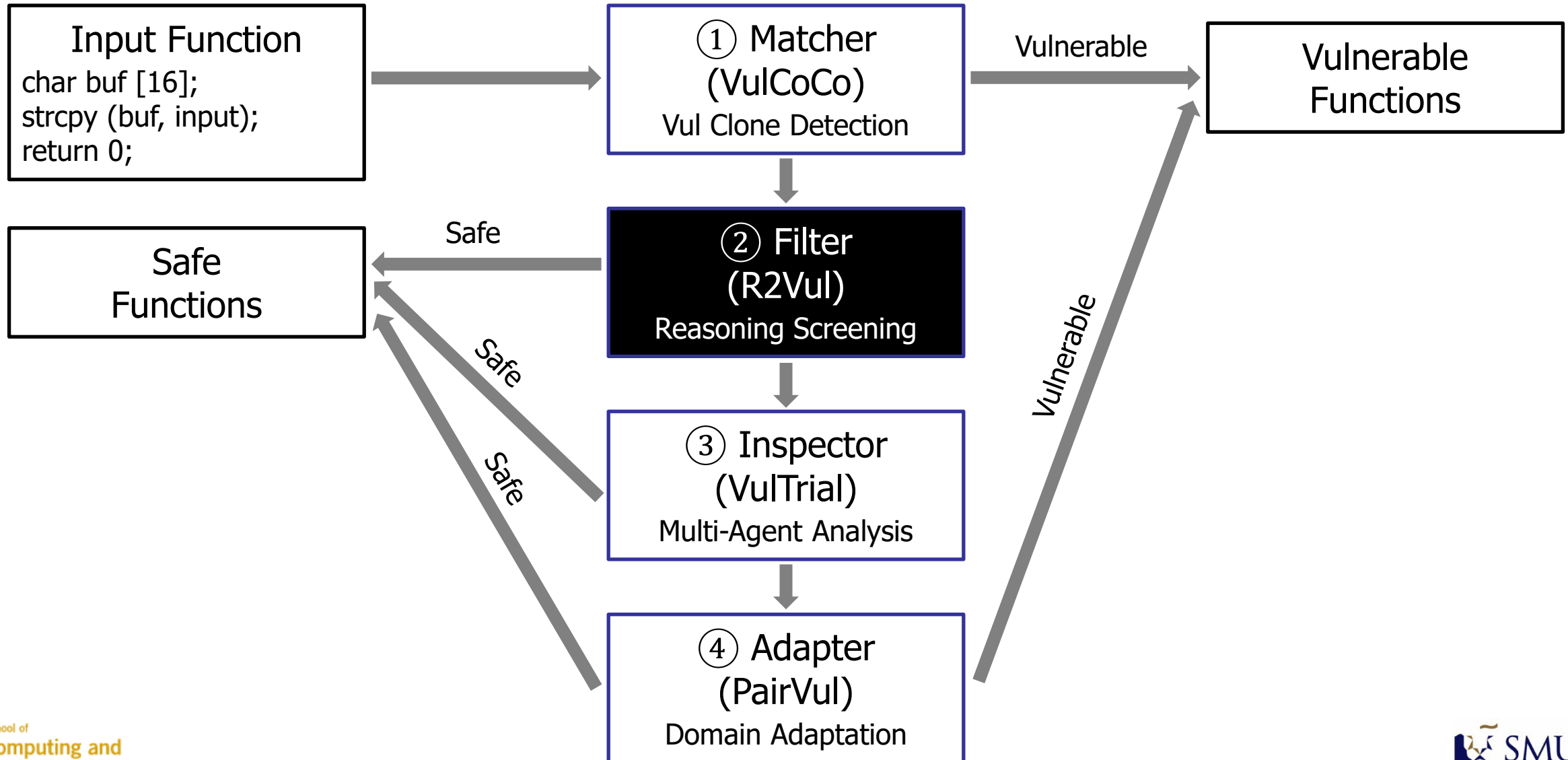
Module Details - ① Matcher (VulCoCo)

■ Step 3: LLM Validation

- Uses an LLM to validate candidate vulnerable clone matches
- Filters out fixed or non-vulnerable clones



TitanCA Core Operational Framework Overview



Module Details - ② Filter (R2Vul)

- ② **Filter (R2Vul)**

- Reasoning-Based Vulnerability Screening

- **Goal**

- Uses reinforcement learning and structured reasoning to identify functions that may contain vulnerabilities

Module Details - ② Filter (R2Vul)

- Step 1: Generate *Grounded* Reasoning from Labeled Data
 - Given CWE and CVE descriptions, generate structured reasoning for labeled functions (vulnerable / safe)

Vul Function 1

```
char buf [16];  
strcpy (buf, input);  
return 0;
```

CWE-119

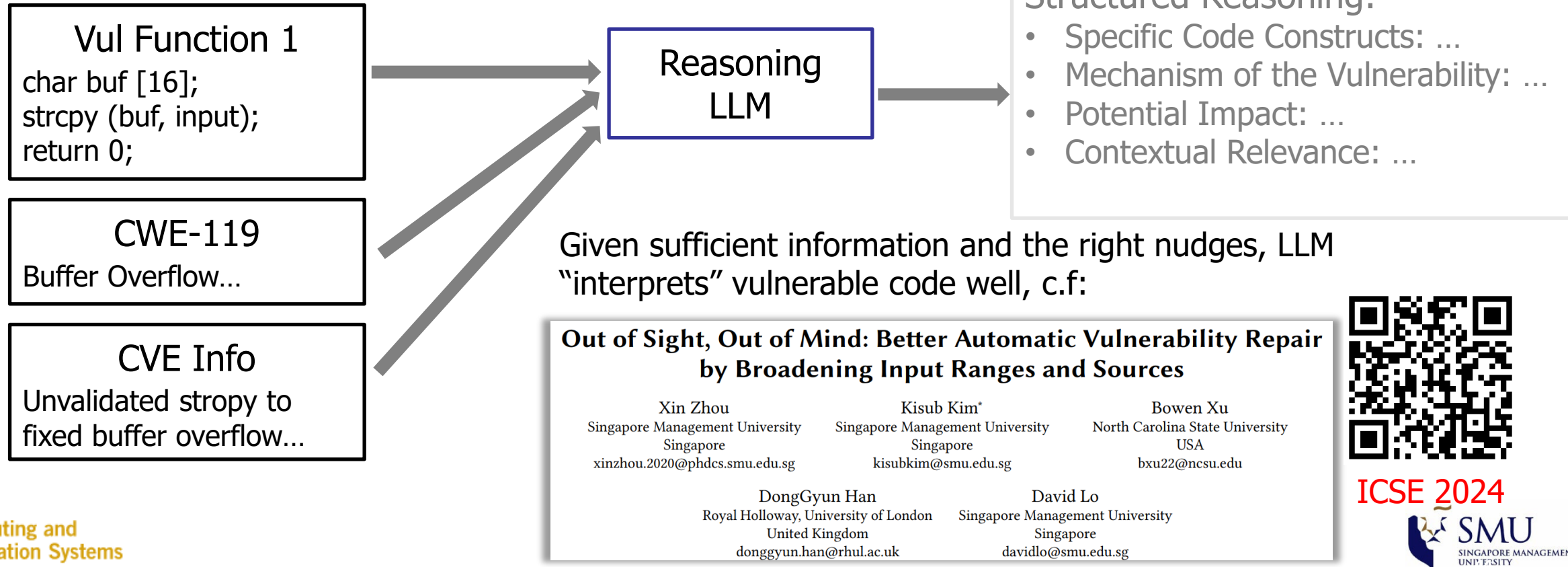
Buffer Overflow...

CVE Info

Unvalidated strcpy to
fixed buffer overflow...

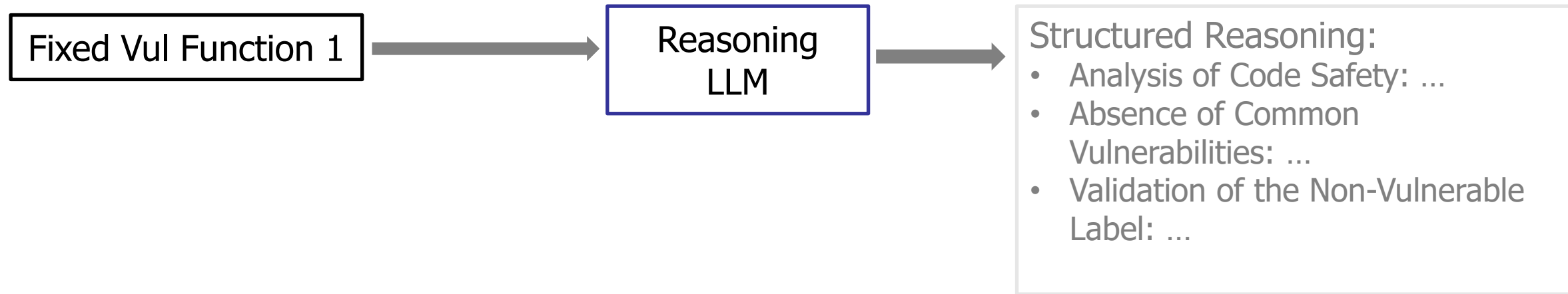
Module Details - ② Filter (R2Vul)

- Step 1: Generate *Grounded* Reasoning from Labeled Data
 - Given CWE and CVE descriptions, generate structured reasoning for labeled functions (vulnerable / safe)



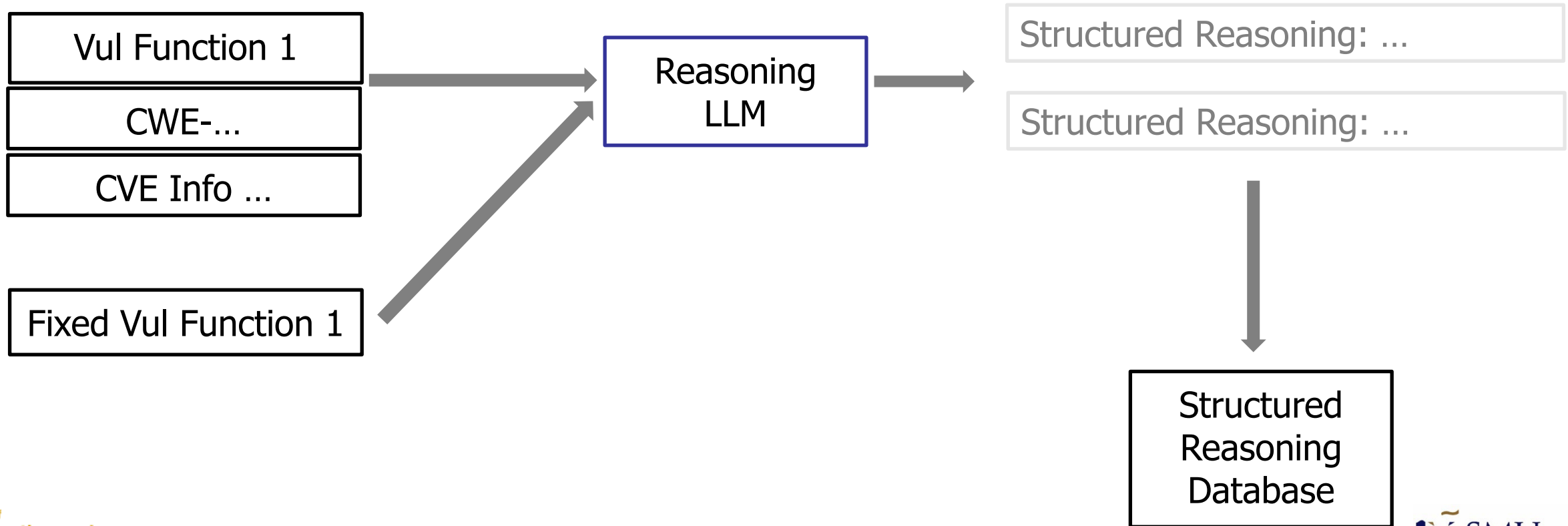
Module Details - ② Filter (R2Vul)

- Step 1: Generate *Grounded* Reasoning from Labeled Data
 - Given CWE and CVE descriptions, generate structured reasoning for labeled functions (vulnerable / safe)



Module Details - ② Filter (R2Vul)

- Step 1: Generate *Grounded* Reasoning from Labeled Data
 - Given CWE and CVE descriptions, generate structured reasoning for labeled functions (vulnerable / safe)

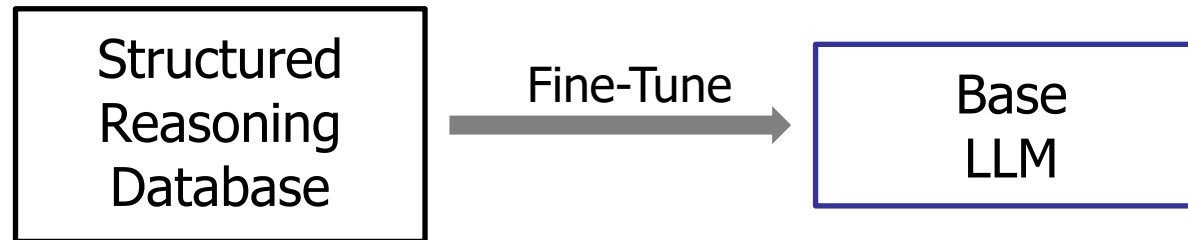


Module Details - ② Filter (R2Vul)

- Step 2: Generate *Misleading* Reasoning from Labeled Data
 - Flip the label of the data
 - Ask the LLM to generate explanations with the flipped label,
 - which will be *misleading*
 - Use the resulting pairs to provide *contrastive* supervision

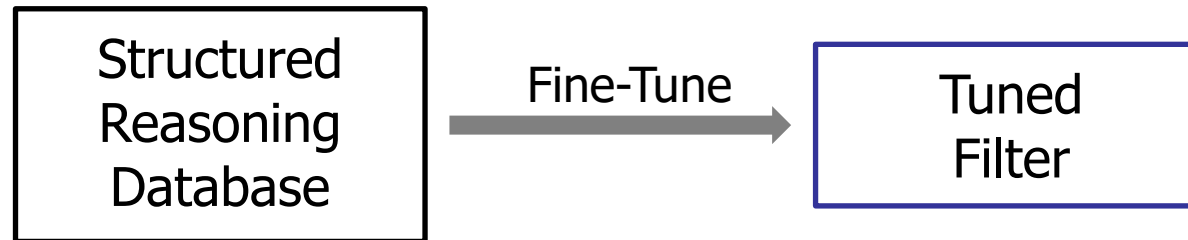
Module Details - ② Filter (R2Vul)

- Step 3: Fine-Tuning LLM with Reinforcement Learning
 - Use grounded and misleading reasoning in the database as *preference* signals
 - Fine-tune the model to prefer generating *grounded* reasoning and avoid generating *misleading* reasoning



Module Details - ② Filter (R2Vul)

- Step 3: Fine-Tuning LLM with Reinforcement Learning
 - Use grounded and misleading reasoning in the database as *preference* signals
 - Fine-tune the model to prefer generating *grounded* reasoning and avoid generating *misleading* reasoning



Module Details - ② Filter (R2Vul)

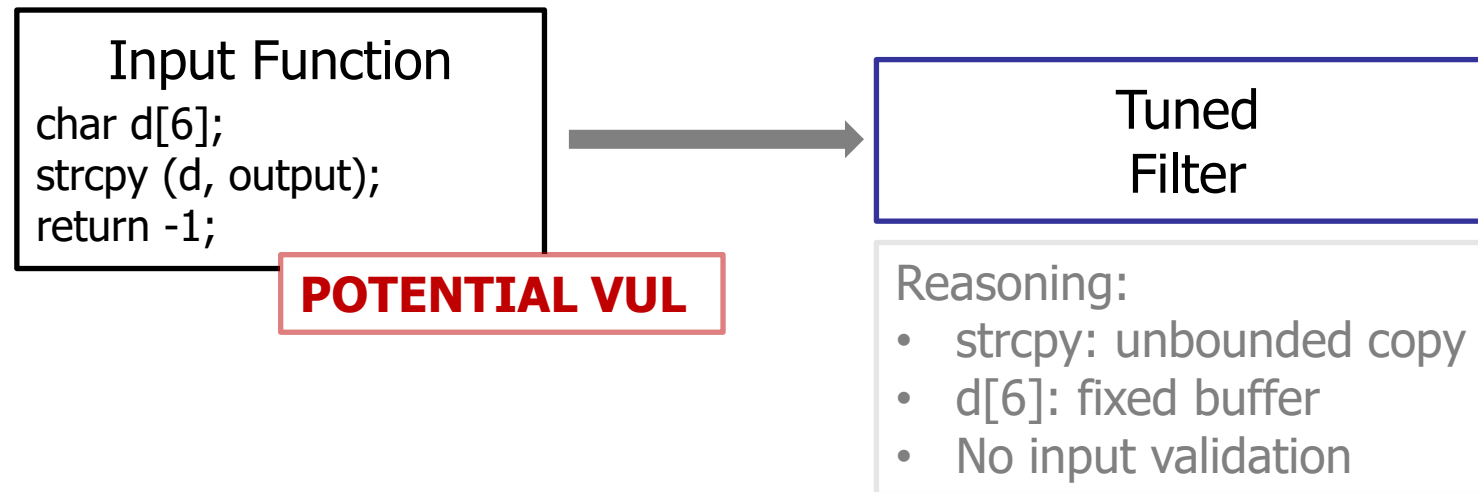
- Step 3: Screening Input Functions
 - Evaluate input functions via structured vulnerability reasoning

Input Function

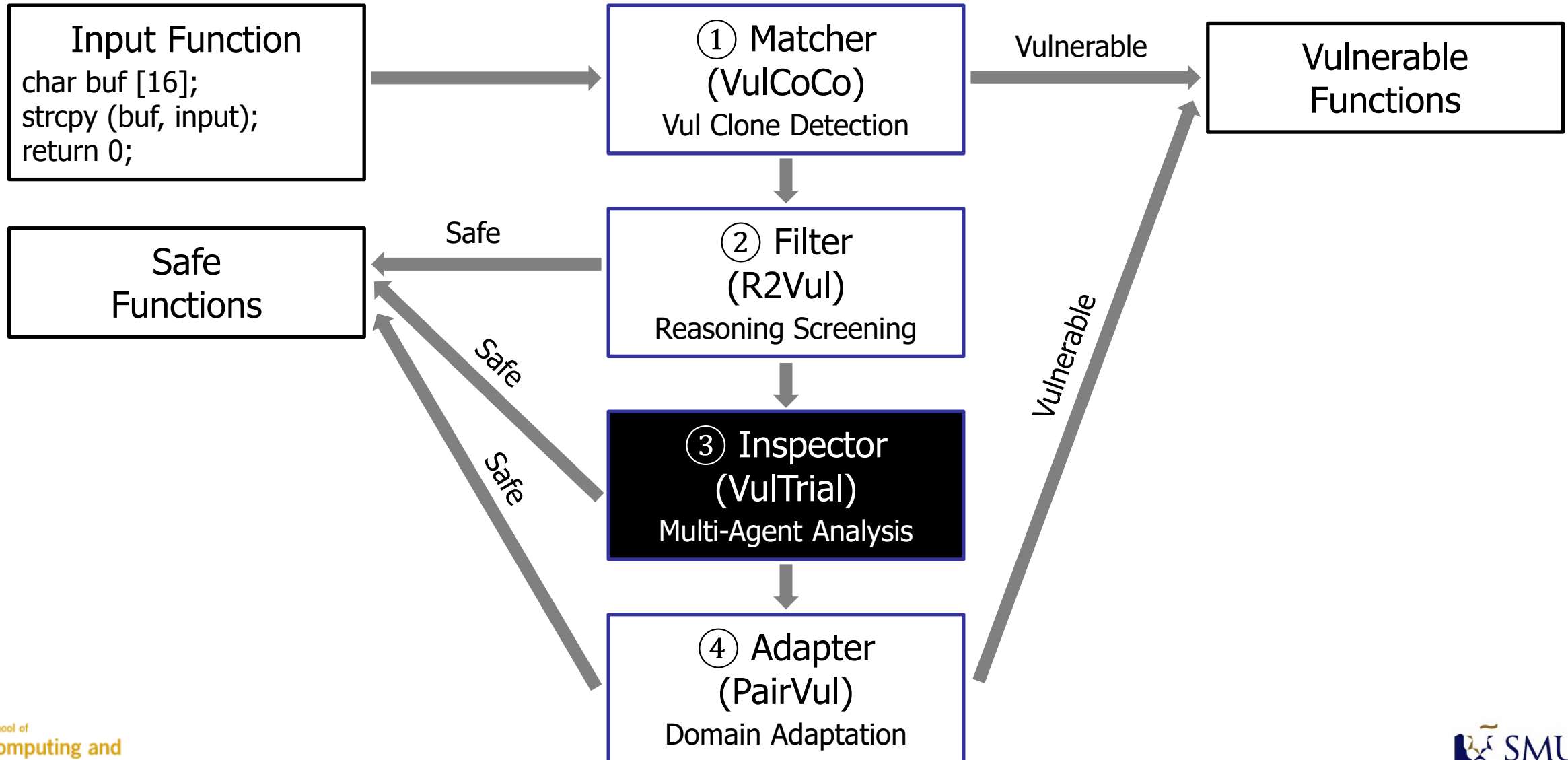
```
char d[6];  
strcpy (d, output);  
return -1;
```

Module Details - ② Filter (R2Vul)

- Step 3: Screening Input Functions
 - Evaluate input functions via structured vulnerability reasoning



TitanCA Core Operational Framework Overview



Module Details - ③ Inspector (VulTrial)

- **③ Inspector (VulTrial)**
 - Multi-Agent Vulnerability Examiner
- **Goal**
 - Applies multi-agent LLM analysis to examine each candidate in depth

Module Details - ③ Inspector (VulTrial)

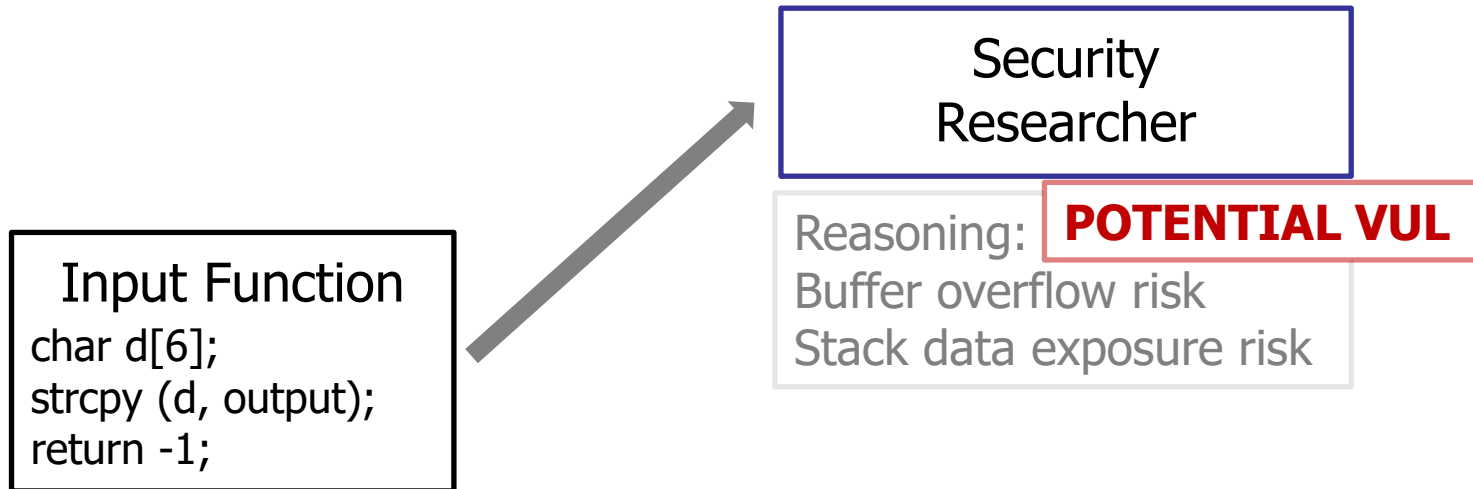
- Step 1: Screening Input Functions
 - Evaluate input functions via a review panel using multiple LLM agents

Input Function

```
char d[6];  
strcpy (d, output);  
return -1;
```

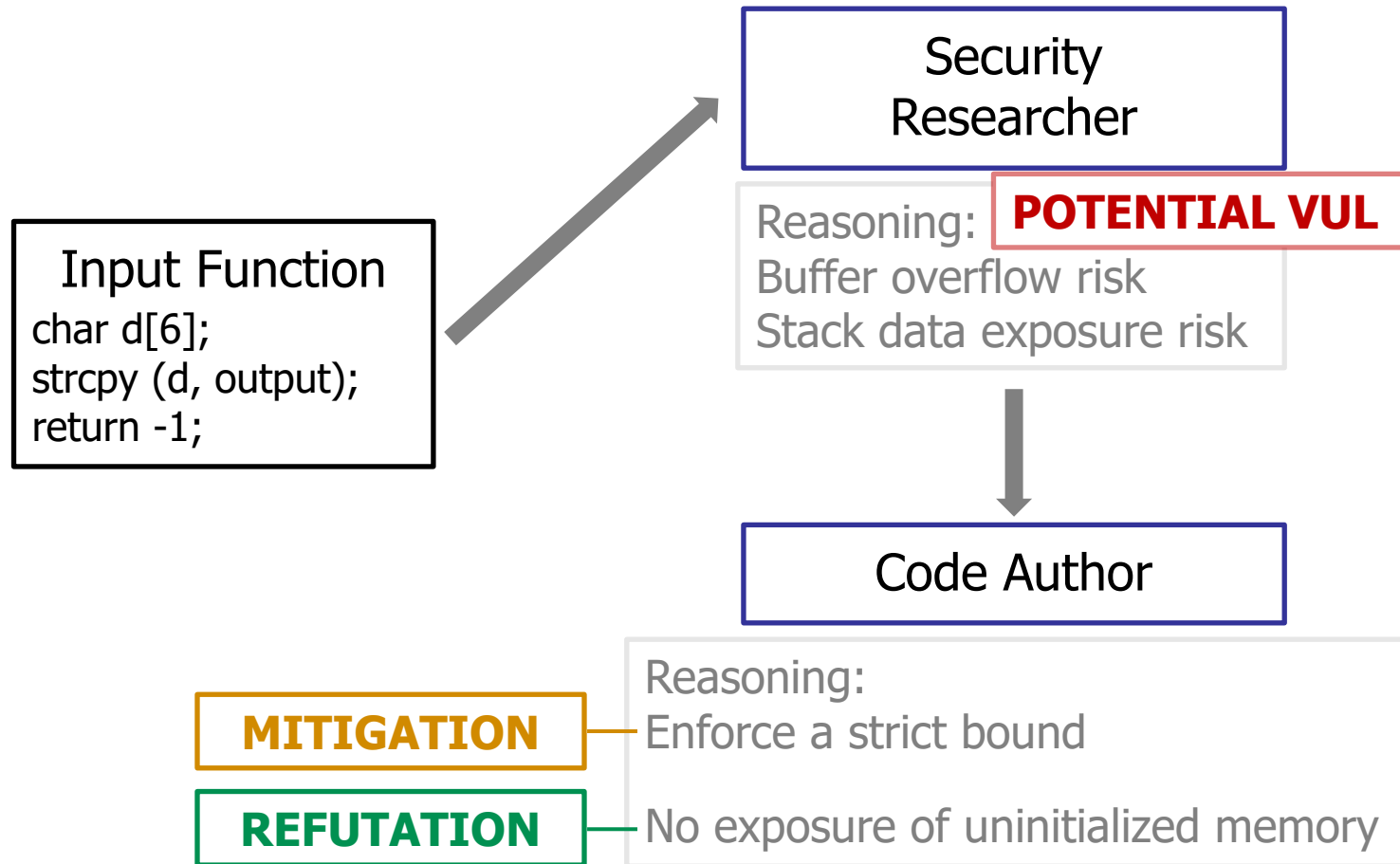
Module Details - ③ Inspector (VulTrial)

- Step 1: Screening Input Functions
 - Evaluate input functions via a review panel using multiple LLM agents



Module Details - ③ Inspector (VulTrial)

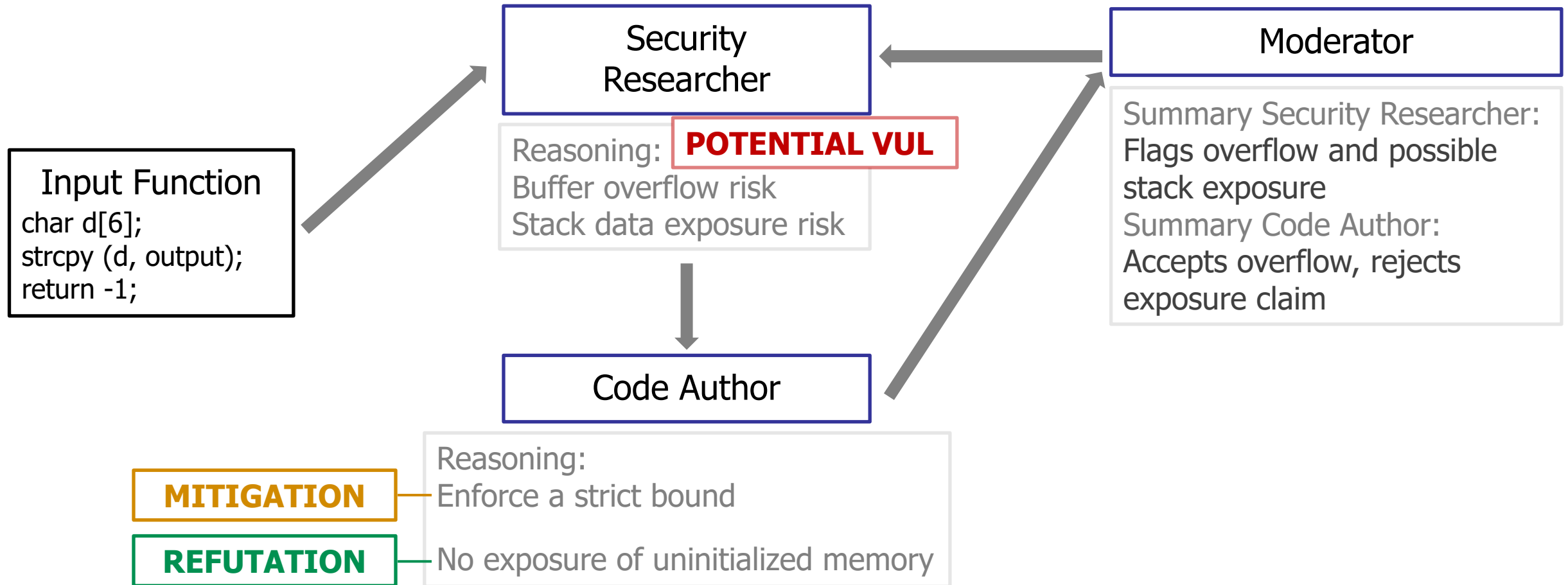
- Step 1: Screening Input Functions
 - Evaluate input functions via a review panel using multiple LLM agents



Module Details - ③ Inspector (VulTrial)

■ Step 1: Screening Input Functions

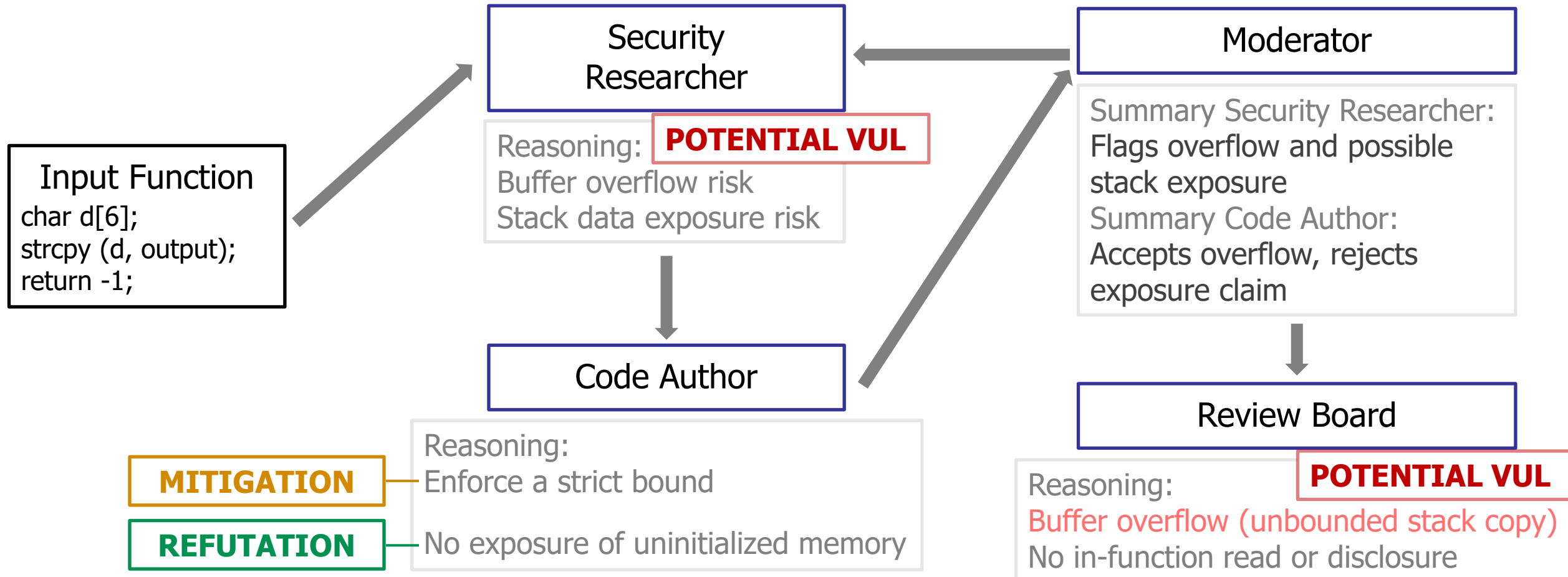
- Evaluate input functions via a review panel using multiple LLM agents



Module Details - ③ Inspector (VulTrial)

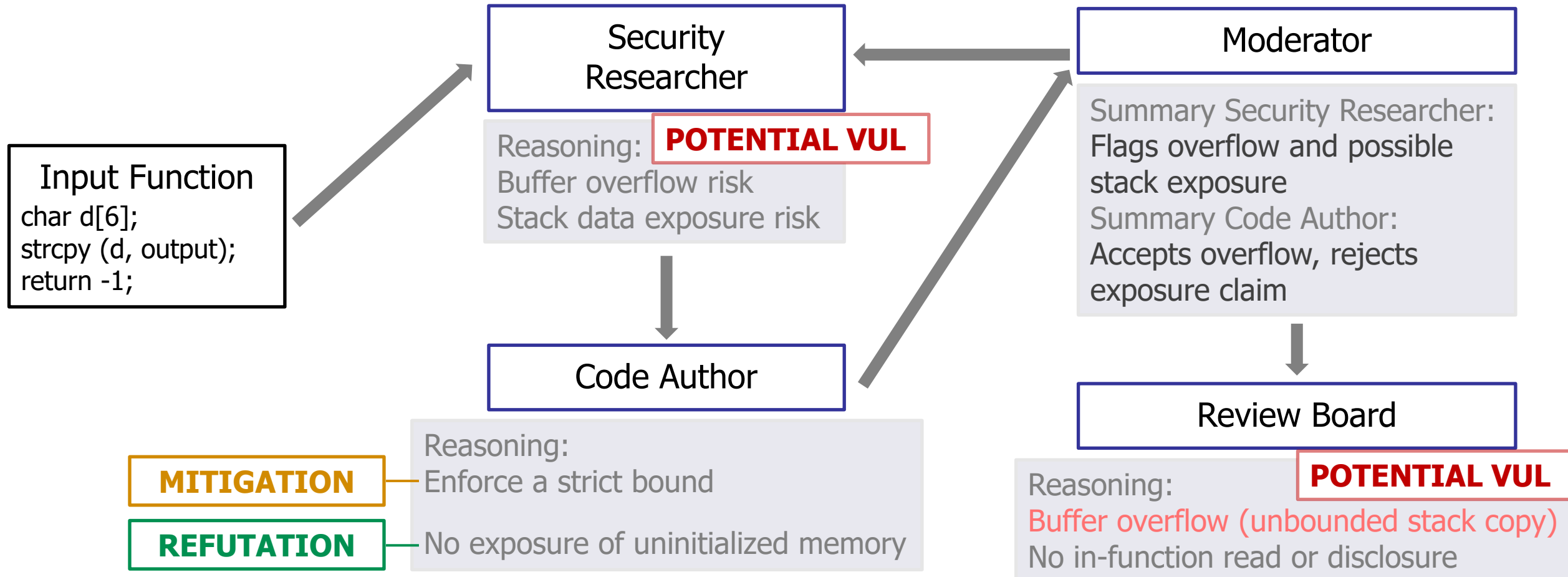
■ Step 1: Screening Input Functions

- Evaluate input functions via a review panel using multiple LLM agents



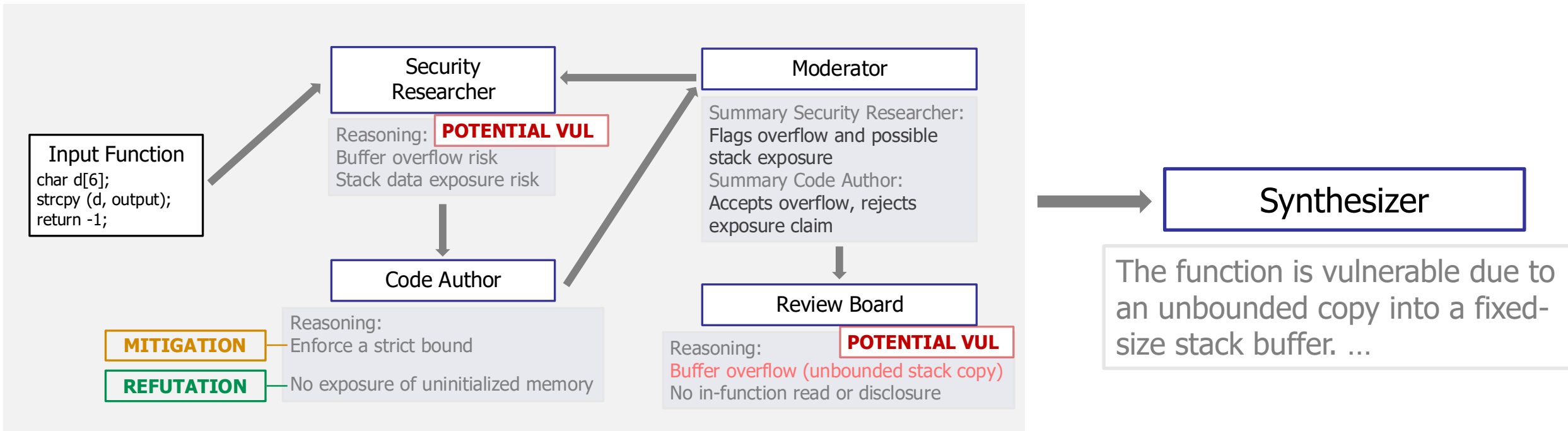
Module Details - ③ Inspector (VulTrial)

- Step 2: Improve Reasoning Readability
 - Synthesize all agent reasoning into a single coherent explanation

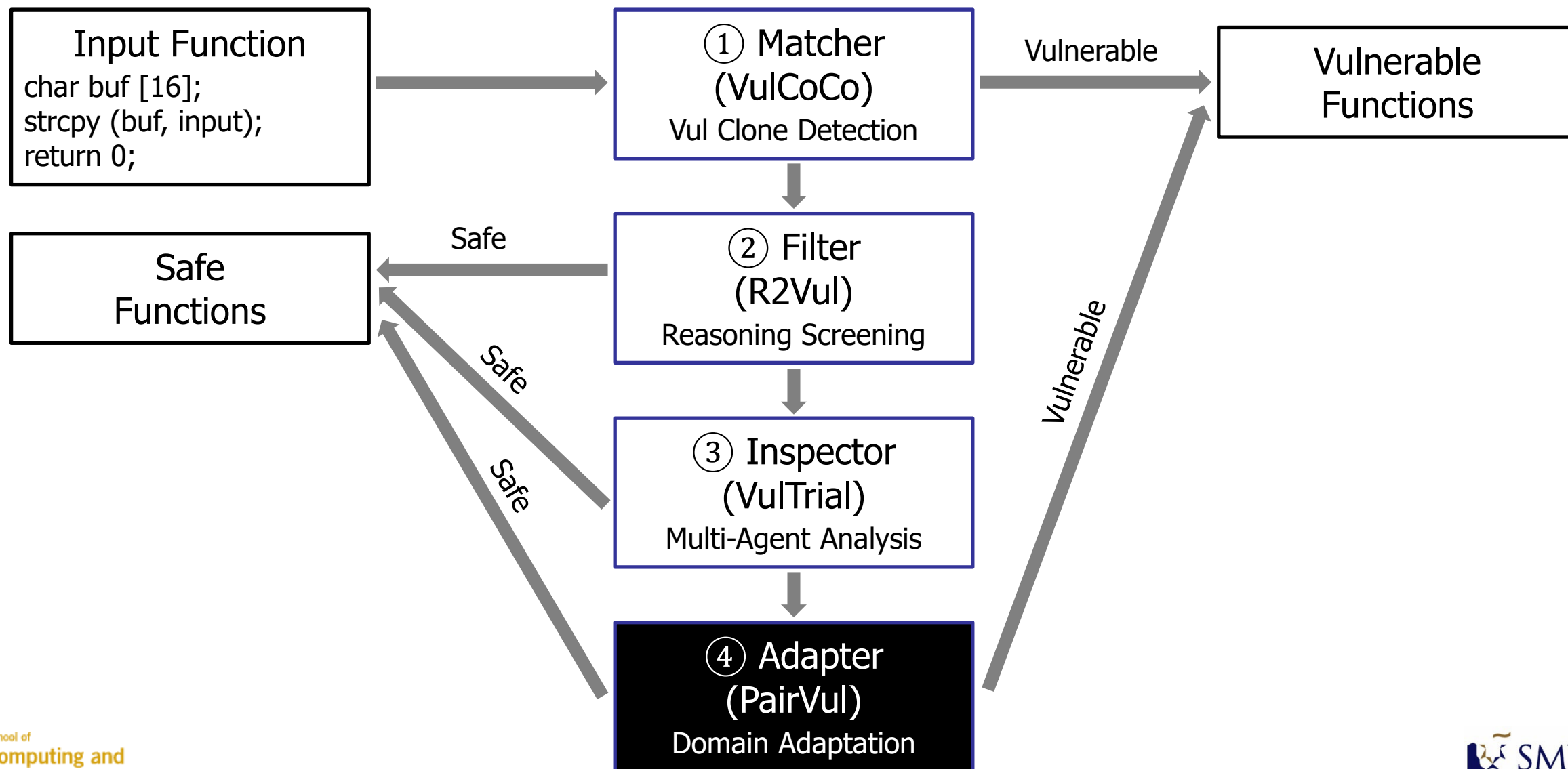


Module Details - ③ Inspector (VulTrial)

- Step 2: Improve Reasoning Readability
 - Synthesize all agent reasoning into a single coherent explanation



TitanCA Core Operational Framework Overview



Module Details - ④ Adapter (PairVul)

- ④ **Adapter (PairVul)**

- GovTech-Specific Model Adapter

- **Goal**

- Reduce GovTech-specific false positives by fine-tuning models to **desensitize** the TitanCA approach to the recurring false-alarm pattern

Module Details - ④ Adapter (PairVul)

- Step 1: False-Positive Analysis
 - Collect false positives from GovTech evaluations
 - Identify recurring error patterns

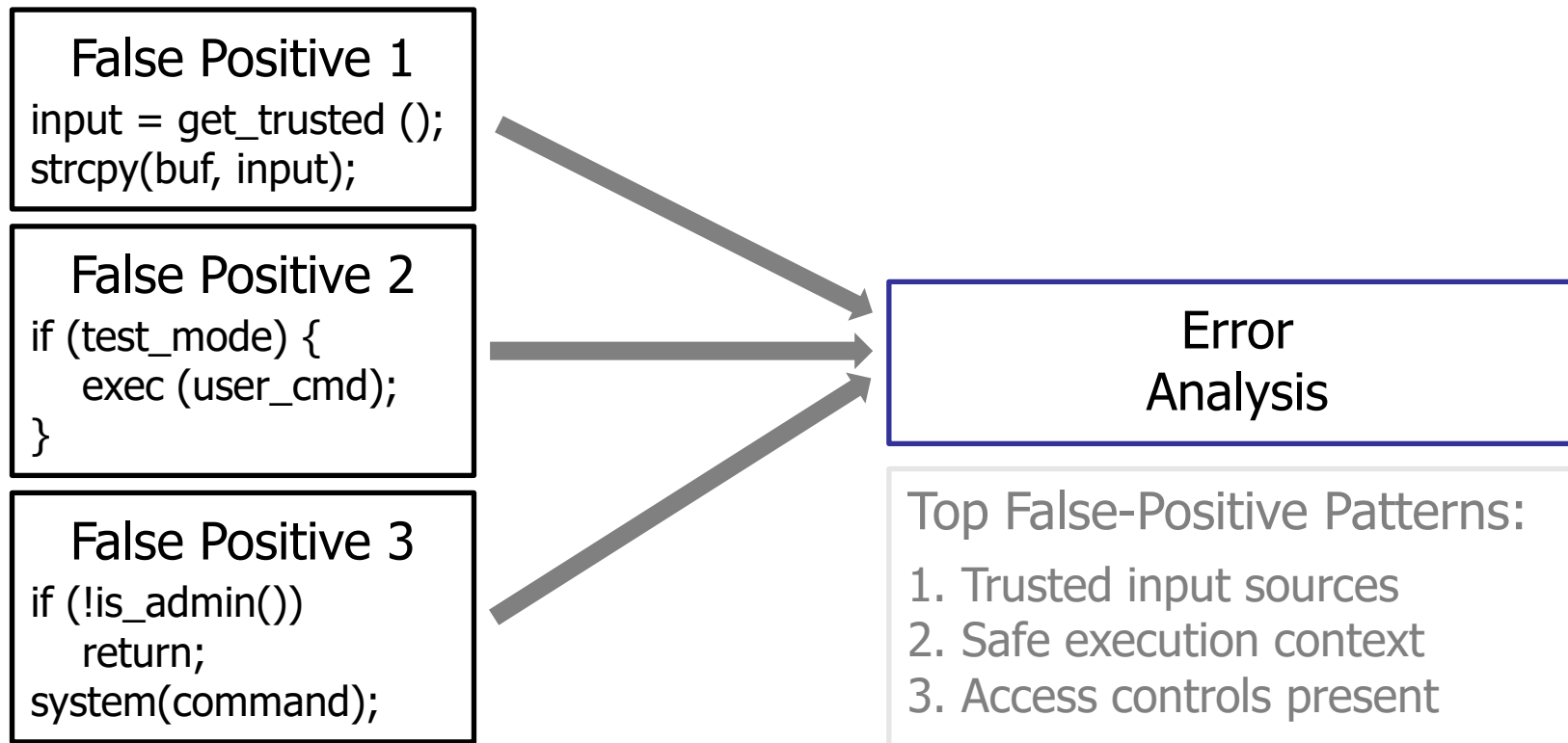
False Positive 1
`input = get_trusted ();`
`strcpy(buf, input);`

False Positive 2
`if (test_mode) {`
`exec (user_cmd);`
`}`

False Positive 3
`if (!is_admin())`
`return;`
`system(command);`

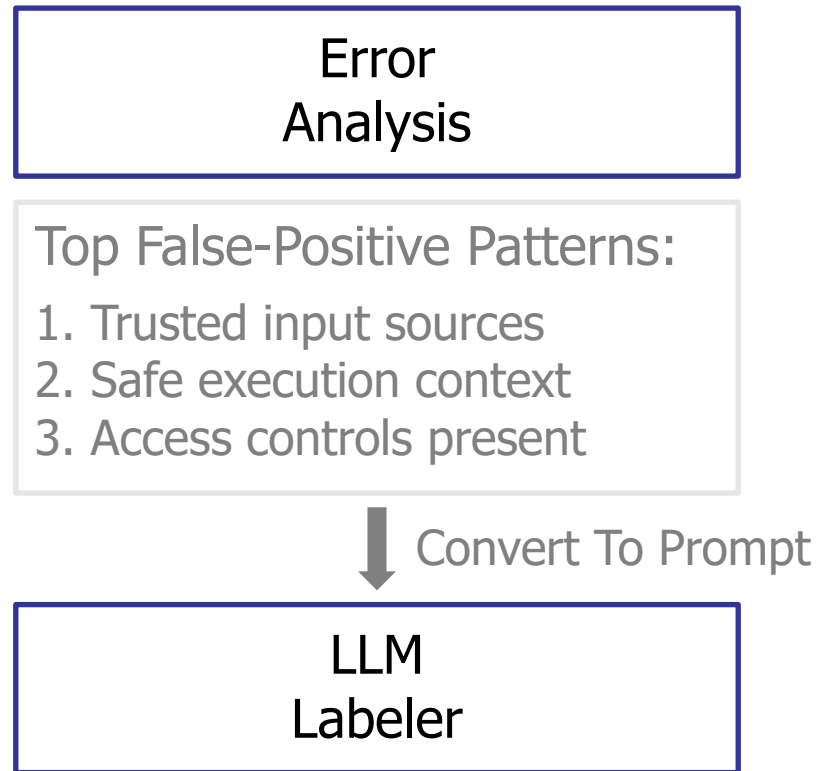
Module Details - ④ Adapter (PairVul)

- Step 1: False-Positive Analysis
 - Collect false positives from GovTech evaluations
 - Identify recurring error patterns



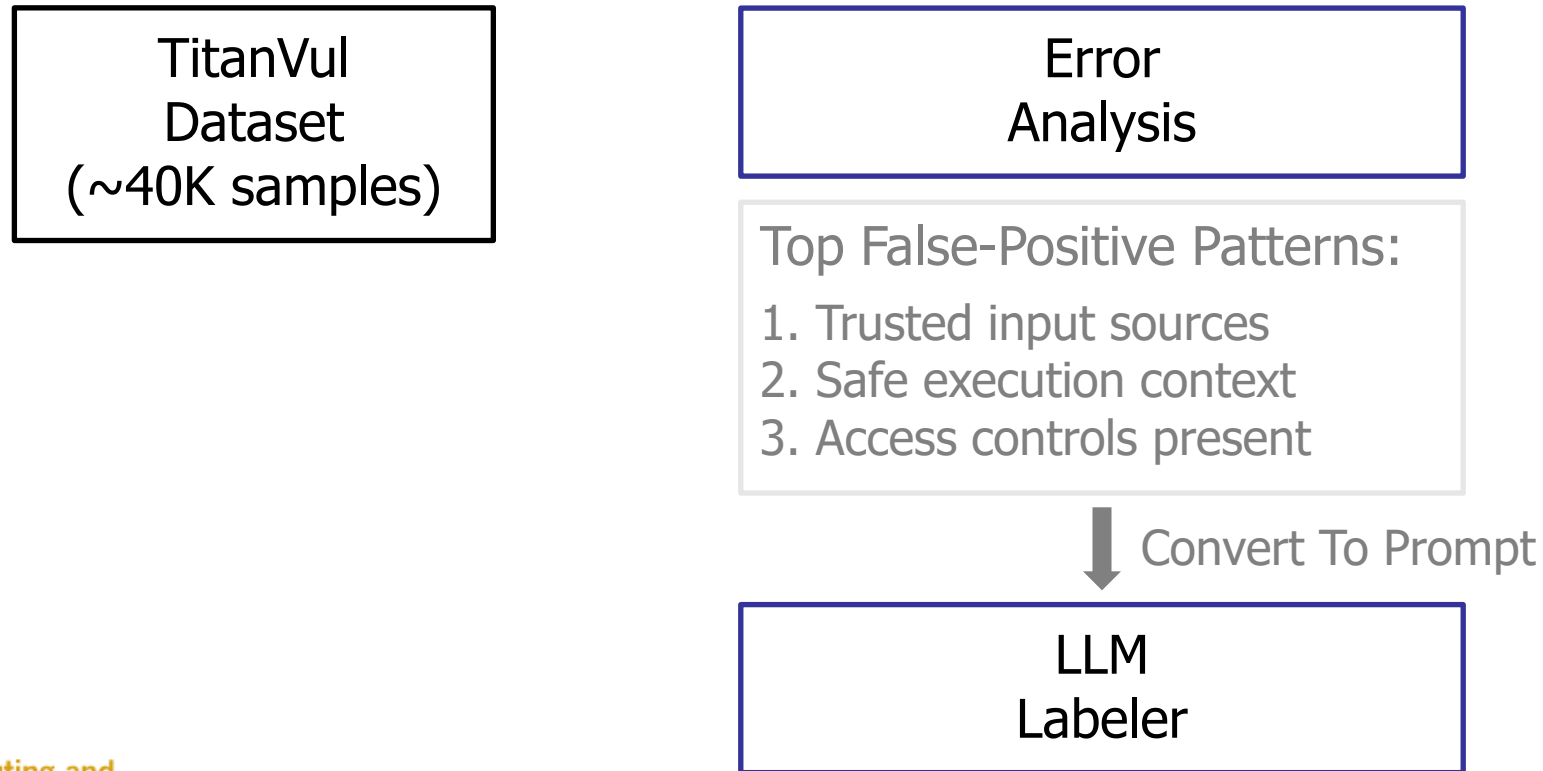
Module Details - ④ Adapter (PairVul)

- Step 2: LLM-Based Relabeling with Error Patterns
 - Use the prompt to relabel TitanVul, one of our released vulnerability datasets
 - Produce a GovTech-aligned relabeled TitanVul for fine-tuning



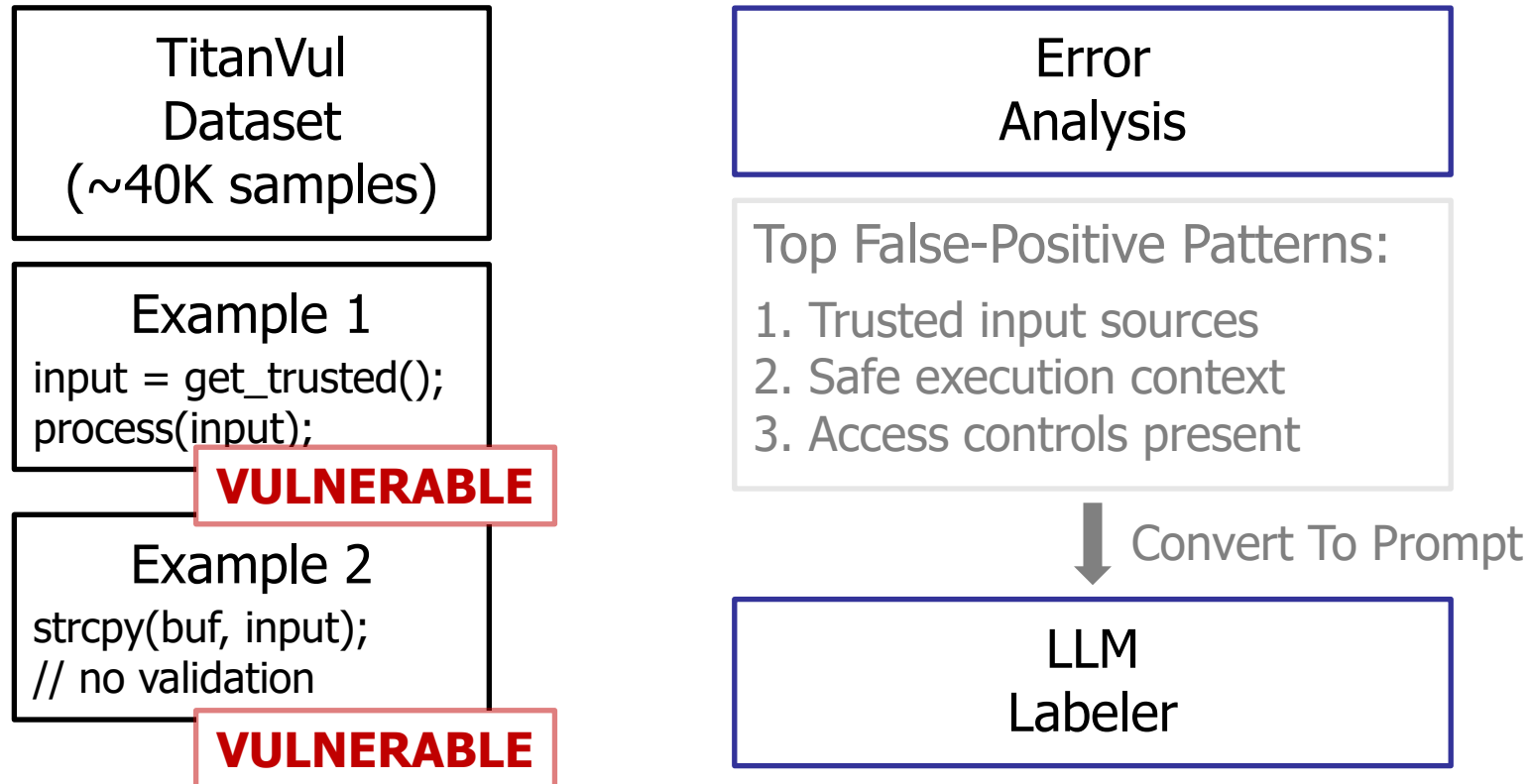
Module Details - ④ Adapter (PairVul)

- Step 2: LLM-Based Relabeling with Error Patterns
 - Use the prompt to relabel TitanVul, one of our released vulnerability datasets
 - Produce a GovTech-aligned relabeled TitanVul for fine-tuning



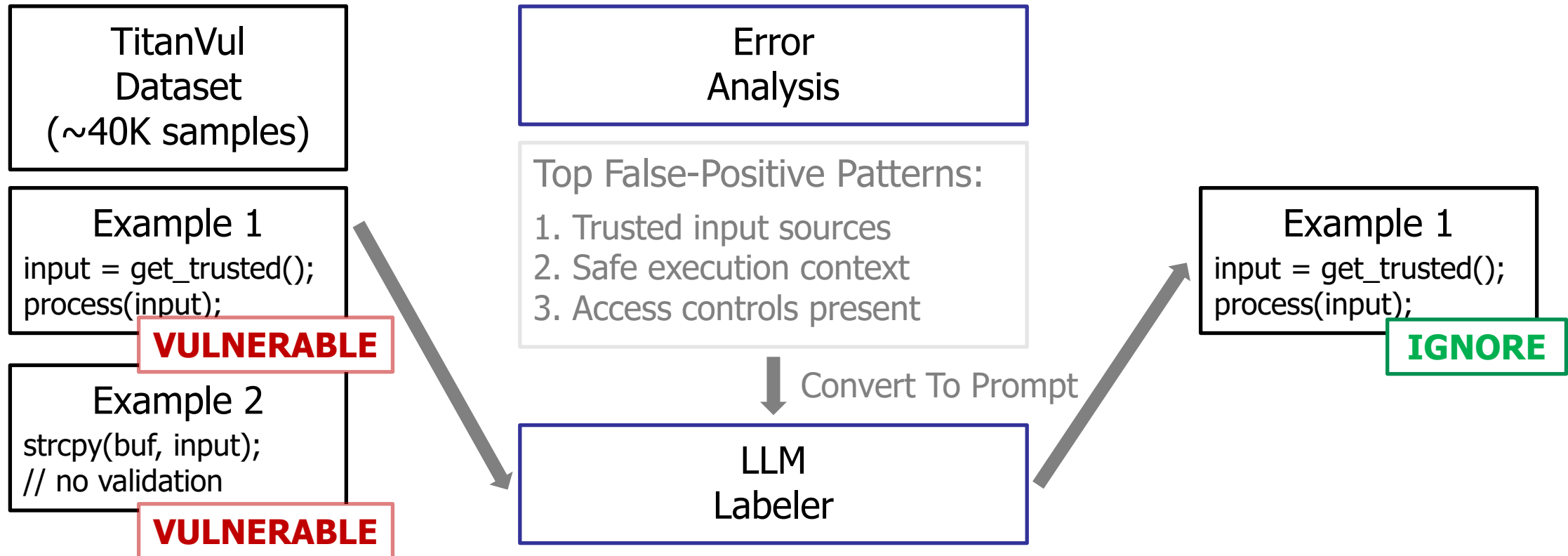
Module Details - ④ Adapter (PairVul)

- Step 2: LLM-Based Relabeling with Error Patterns
 - Use the prompt to relabel TitanVul, one of our released vulnerability datasets
 - Produce a GovTech-aligned relabeled TitanVul for fine-tuning



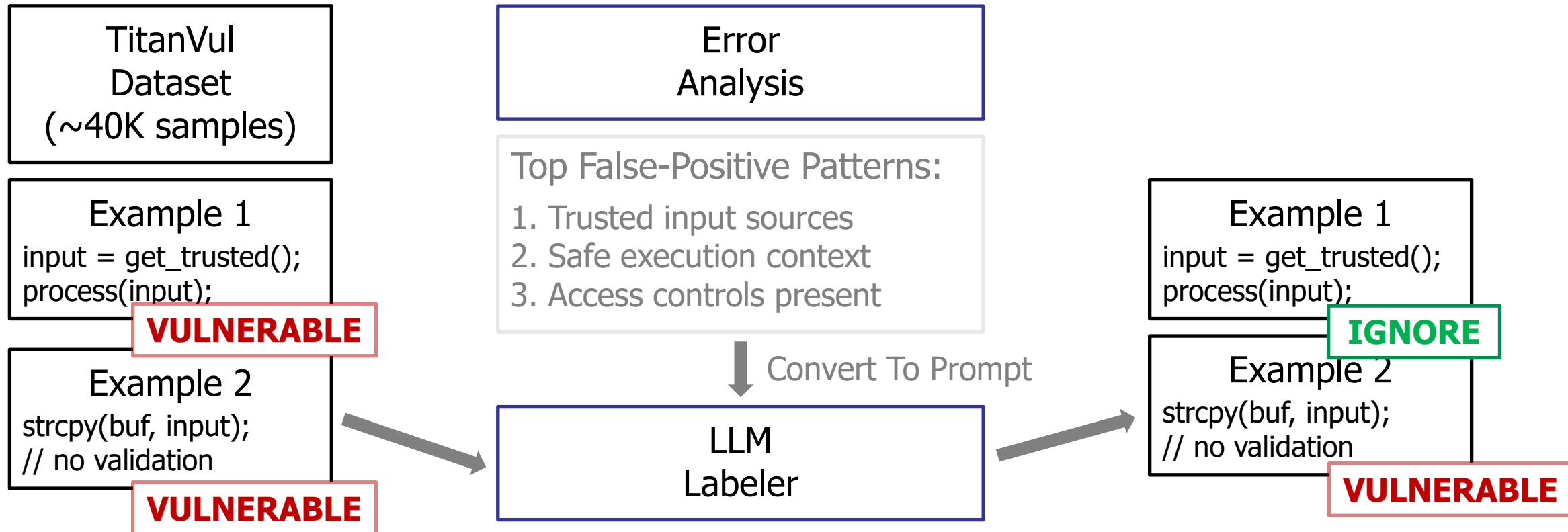
Module Details - ④ Adapter (PairVul)

- Step 2: LLM-Based Relabeling with Error Patterns
 - Use the prompt to relabel TitanVul, one of our released vulnerability datasets
 - Produce a GovTech-aligned relabeled TitanVul for fine-tuning



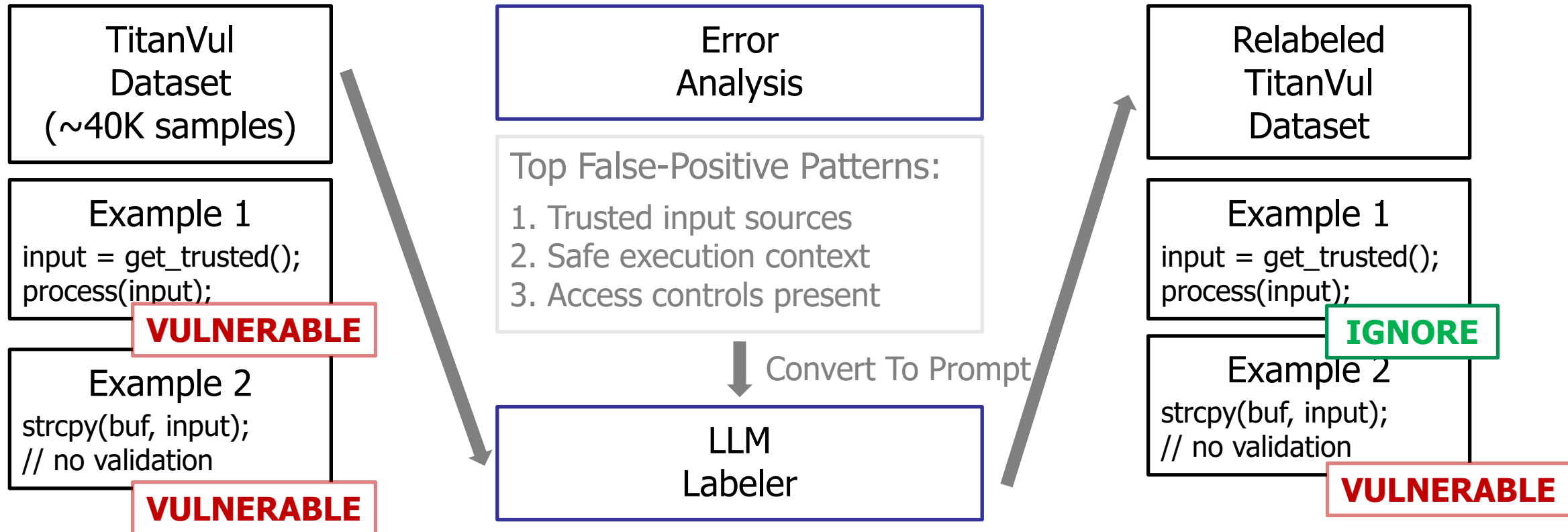
Module Details - ④ Adapter (PairVul)

- Step 2: LLM-Based Relabeling with Error Patterns
 - Use the prompt to relabel TitanVul, one of our released vulnerability datasets
 - Produce a GovTech-aligned relabeled TitanVul for fine-tuning



Module Details - ④ Adapter (PairVul)

- Step 2: LLM-Based Relabeling with Error Patterns
 - Use the prompt to relabel TitanVul, one of our released vulnerability datasets
 - Produce a GovTech-aligned relabeled TitanVul for fine-tuning



Module Details - ④ Adapter (PairVul)

- Step 3: Fine-Tuning Adapter
 - Fine-tune the adapter on the relabeled dataset to align with GovTech's data



Module Details - ④ Adapter (PairVul)

- Step 3: Fine-Tuning Adapter
 - Fine-tune the adapter on the relabeled dataset to align with GovTech's data



Module Details - ④ Adapter (PairVul)

- Step 3: Fine-Tuning Adapter
 - Fine-tune the adapter on the relabeled dataset to align with GovTech's data



Module Details - ④ Adapter (PairVul)

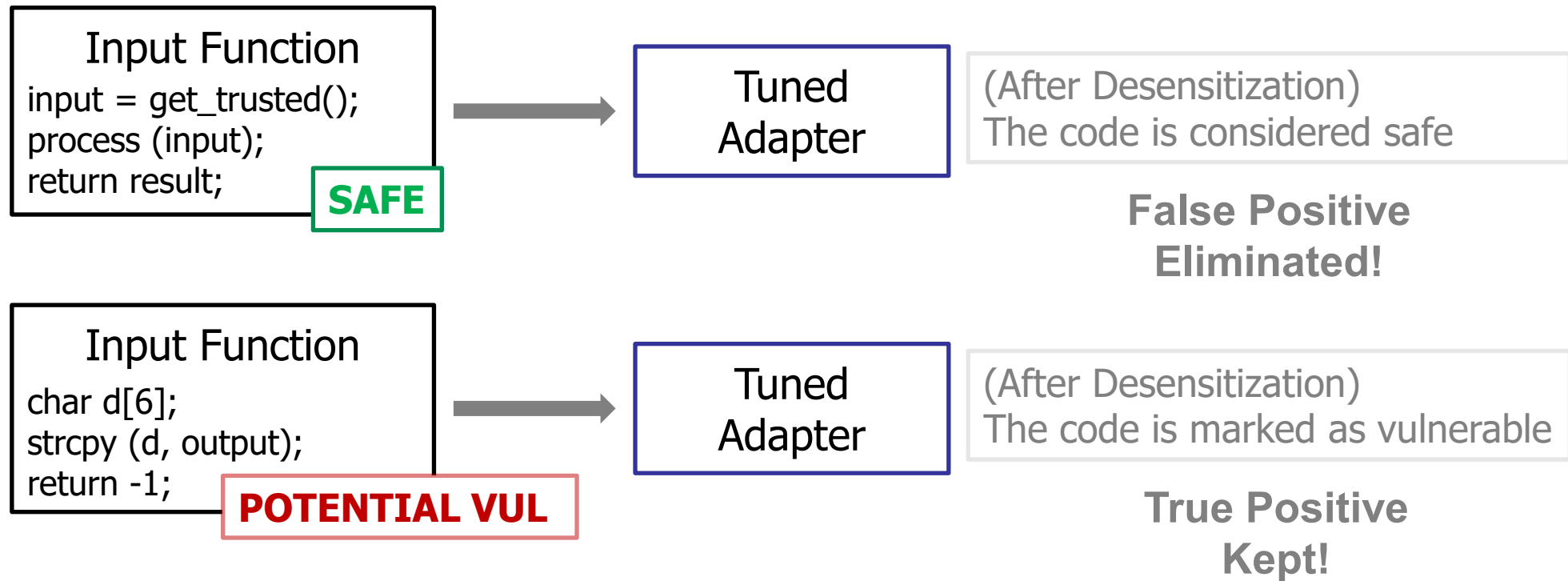
- Step 4: Screening Input Functions
 - Evaluate input functions with the fine-tuned adapter

Input Function

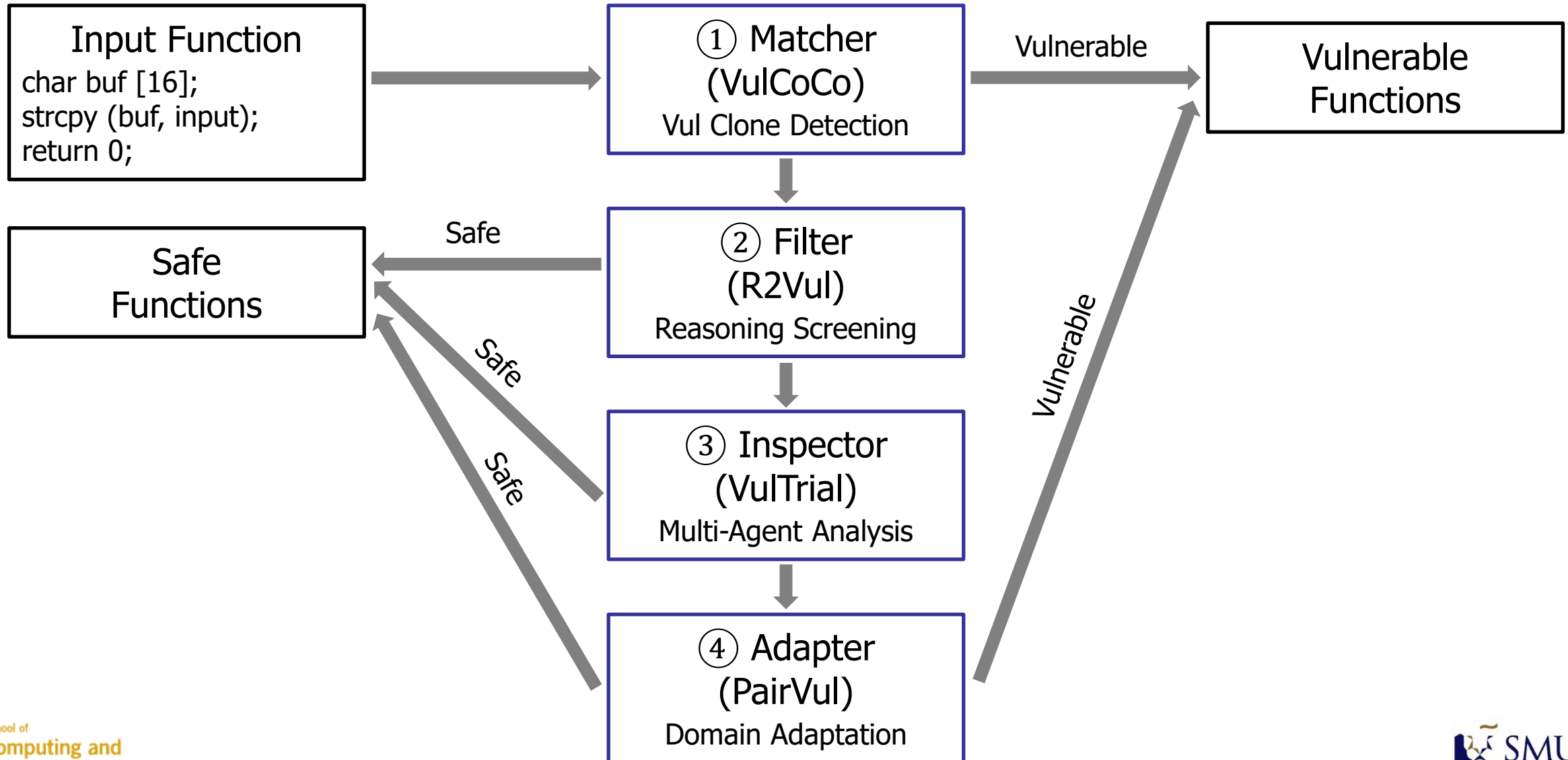
```
input = get_trusted();  
process (input);  
return result;
```

Module Details - ④ Adapter (PairVul)

- Step 4: Screening Input Functions
 - Evaluate input functions with the fine-tuned adapter



TitanCA Core Operational Framework Overview



More Information

VulCoCo: A Simple Yet Effective Method for Detecting Vulnerable Code Clones

TAN BUI, Singapore Management University, Singapore
YAN NAING TUN, Singapore Management University, Singapore
NGUYEN PHUC THANH, Singapore Management University, Singapore
YINDU SU, Singapore Management University, Singapore
FERDIAN THUNG, Singapore Management University, Singapore
YIKUN LI, Singapore Management University, Singapore
HAN WEI ANG, GovTech, Singapore
YIDE YIN, GovTech, Singapore
FRANK LIAUW, GovTech, Singapore
SHAR LWIN KHIN, Singapore Management University, Singapore
OUH ENG LIEH, Singapore Management University, Singapore
TING ZHANG*, Singapore Management University, Singapore
DAVID LO, Singapore Management University, Singapore



R2VUL: Learning to Reason about Software Vulnerabilities with Reinforcement Learning and Structured Reasoning Distillation

Martin Weyssow^{1*}, Chengran Yang¹, Junkai Chen¹, Ratnadira Widyasari¹, Ting Zhang¹, Huihui Huang¹, Huu Hung Nguyen¹, Yan Naing Tun¹, Tan Bui¹, Yikun Li¹, Ang Han Wei², Frank Liauw², Eng Lieh Ouh¹, Lwin Khin Shar¹, David Lo¹

¹Singapore Management University

²GovTech Singapore

mweyssow@smu.edu.sg



Let the Trial Begin: A Mock-Court Approach to Vulnerability Detection using LLM-Based Agents

Ratnadira Widyasari[◇], Martin Weyssow[◇], Ivana Clairine Irsan[◇], Han Wei Ang[▲], Frank Liauw[▲]
Eng Lieh Ouh[◇], Lwin Khin Shar[◇], Hong Jin Kang[▲], and David Lo[◇]

[◇]School of Computing and Information Systems, Singapore Management University, Singapore

[▲]GovTech, Singapore

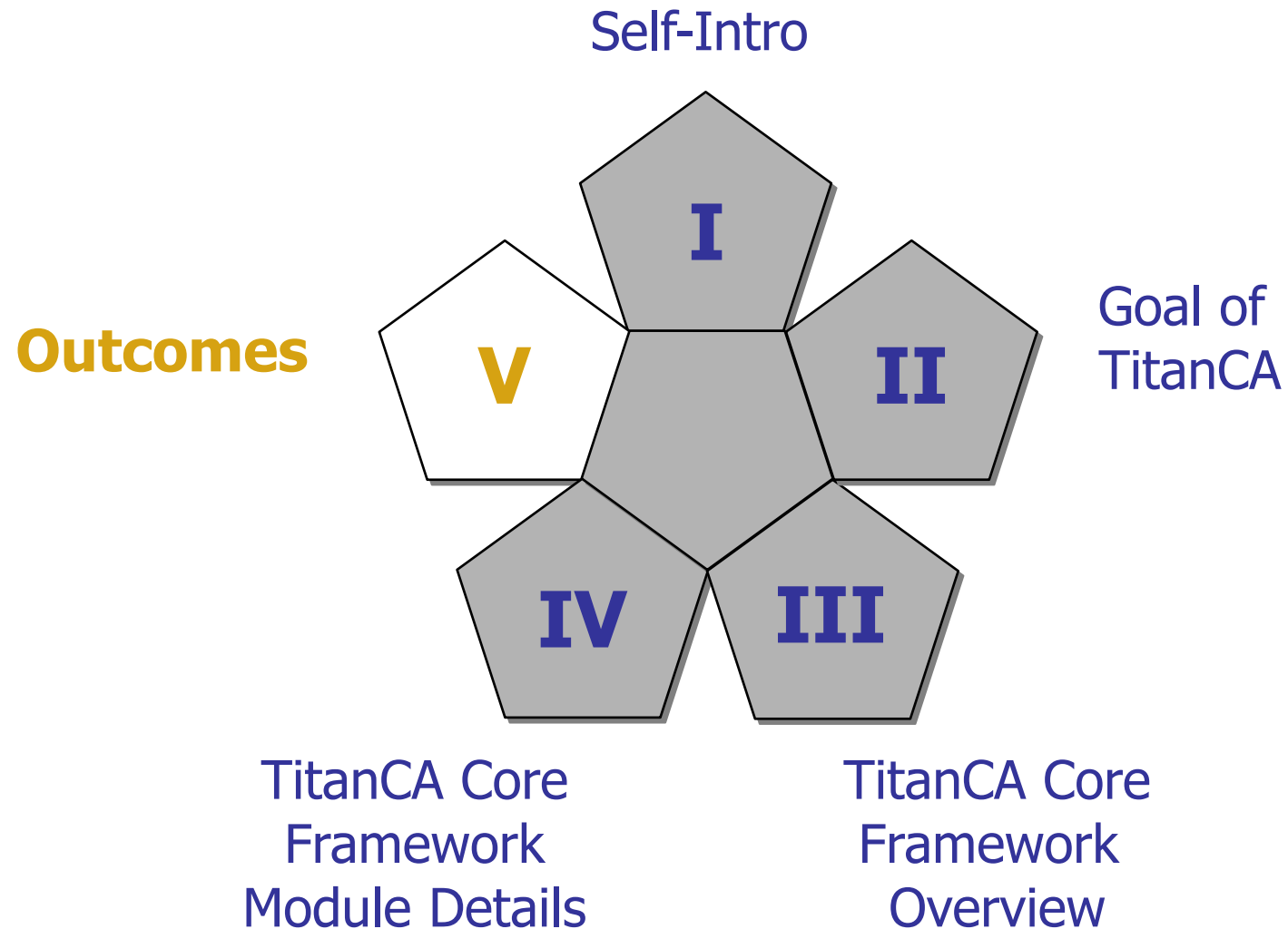
^{*}School of Computer Science, University of Sydney, Australia

{ratnadiraw.2020, mweyssow, ivanairsan, elouh, lkshar, davidlo}@smu.edu.sg,

{ang_han_wei, frank_liauw}@tech.gov.sg, hongjin.kang@sydney.edu.au

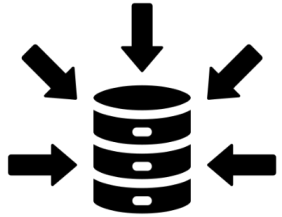


Talk Structure



Data

T1



Vulnerability
Collection

T2

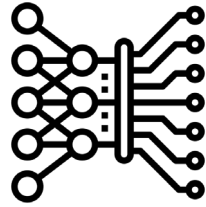


Data Quality
Audit &
Enrichment

TitanCA

Model

T3

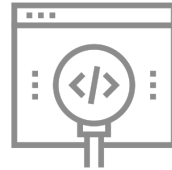


Vulnerability
Detection Models
from LCMs

T4



Model Precision &
Robustness Boost



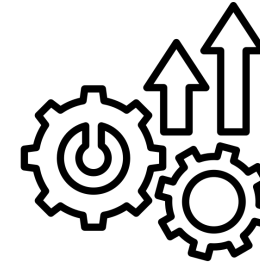
Static Analysis for
Better LCMs



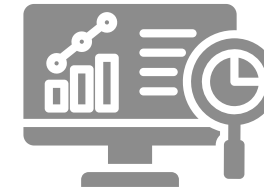
Explanation
Generation

Refinement

T5



Project
Specific
Refinement



Monitoring &
Adaptation



Adv. UI/UX &
User Studies

Within GovTech

Operation

T6



Data
Engineering



Model Efficiency
Optimization



Tooling & GitLab
Integration

T7



Periodic
Model
Maintenance

T8

Outcomes: Data Pillar



Listening scale



127,063

GitHub repositories monitored

(Target: 10,000)



12.7x target achieved

Vulnerability collection



342,843

Real and synthetic vulnerability samples

(Target: 100,000)



3.4x target achieved

Outcomes: Model Pillar



F0.3 score improvement



+46.3%

vs. popular SAST tools
(Target: +30%)



1.5x target achieved

Recall–Precision trade-off



**+10.7% precision
@18.3% recall loss**

(Target: $\geq 50\%$ trade-off ratio,
 $\leq 30\%$ recall loss)



Target met (58% trade-off ratio)

Robustness



80%

Reduction in attack success
(Target: -50%)



1.6x target achieved

Our solutions also *substantially outperform many other LLM-powered solutions* – see individual papers, e.g., on slide 78

Outcomes: Refinement Pillar



Operation performance

+109%

F0.3 score improvement vs. popular SAST tools

(*Target: +30%*)



3.6x target achieved

Outcomes: Operation Pillar



Data engineering



Infrastructure capable of managing **500 TB** of versioned data



Target achieved

GitLab integration

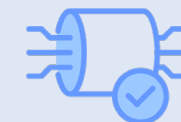


Process up to **10,000 commits** per day (~30k functions)



Target achieved

Model pipeline



Automated pipeline for periodic model retraining and updates



Target achieved

Broader Impact: Vulnerabilities in OSS

- Verified Real-World Vulnerabilities Identified Using TitanCA
 - Website link: <https://titancaproject.github.io/cves.html>

151

Vulnerabilities Found & Fixed

108

CVEs in Total

35

CVEs Published

73

CVEs in Pipeline

As of Jan 7, 2026

Discovered Vulnerabilities

The TITAN project has successfully identified and disclosed the following Common Vulnerabilities and Exposures (CVEs) through our intelligent vulnerability discovery system.

37 Published

75 In Progress

CVE-2025-14306

CRITICAL

CWE-22

A directory traversal vulnerability exists in the CacheCleaner component of Robocode version 1.9.3.6. The recursivelyDelete method fails to properly sanitize file paths, allowing attackers to traverse directories and delete arbitrary files on the system. This vulnerability can be exploited by sub...

Published: 2025-12-09

CVE-2025-14307

CRITICAL

CWE-377

An insecure temporary file creation vulnerability exists in the AutoExtract component of Robocode version 1.9.3.6. The createTempFile method fails to securely create temporary files, allowing attackers to exploit race conditions and potentially execute arbitrary code or overwrite critical files. ...

Published: 2025-12-09

CVE-2025-14308

CRITICAL

CWE-190

An integer overflow vulnerability exists in the write method

CVE-2025-14310

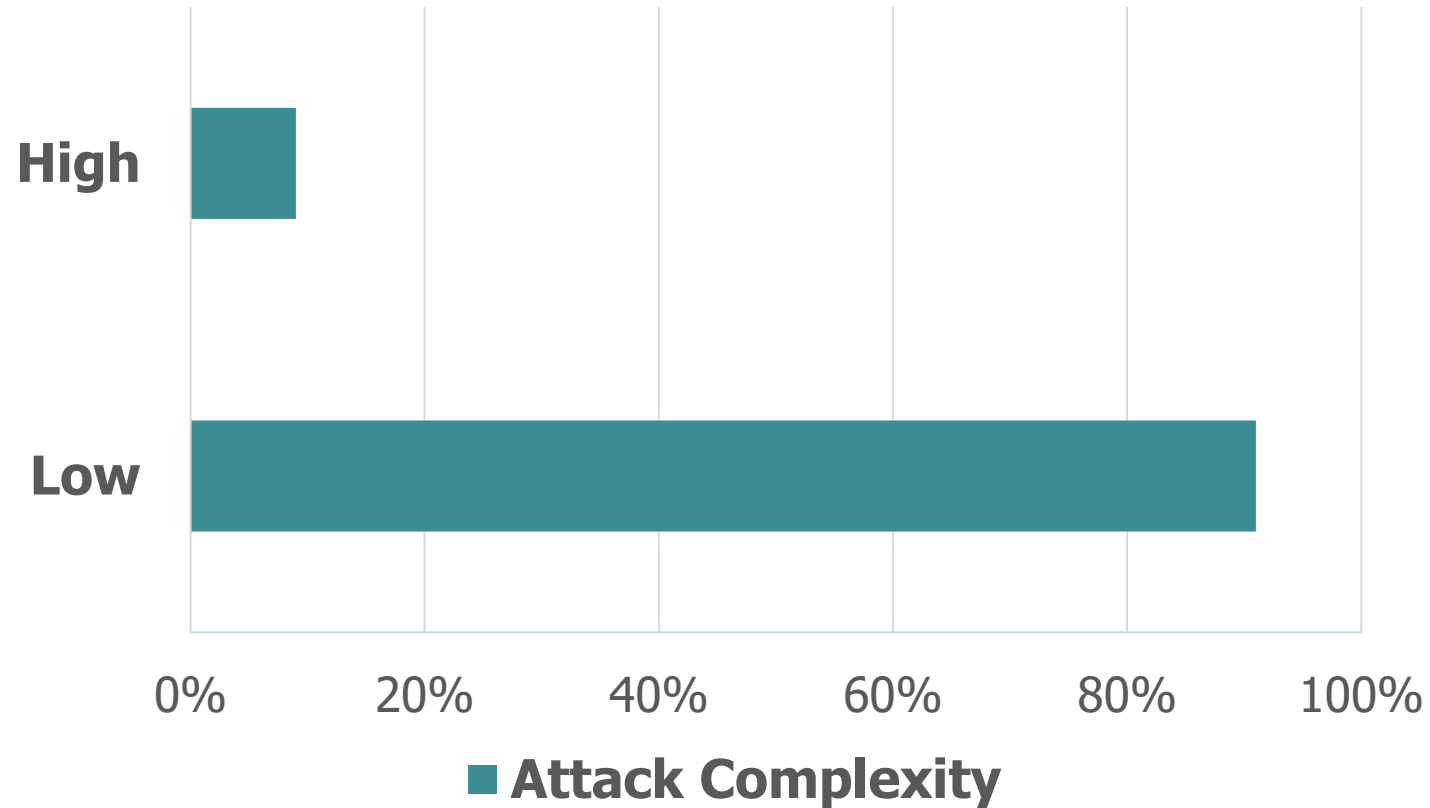
CRITICAL

CWE-120

Buffer Copy without Checking Size of Input ('Classic Buffer

Broader Impact: Vulnerabilities in OSS

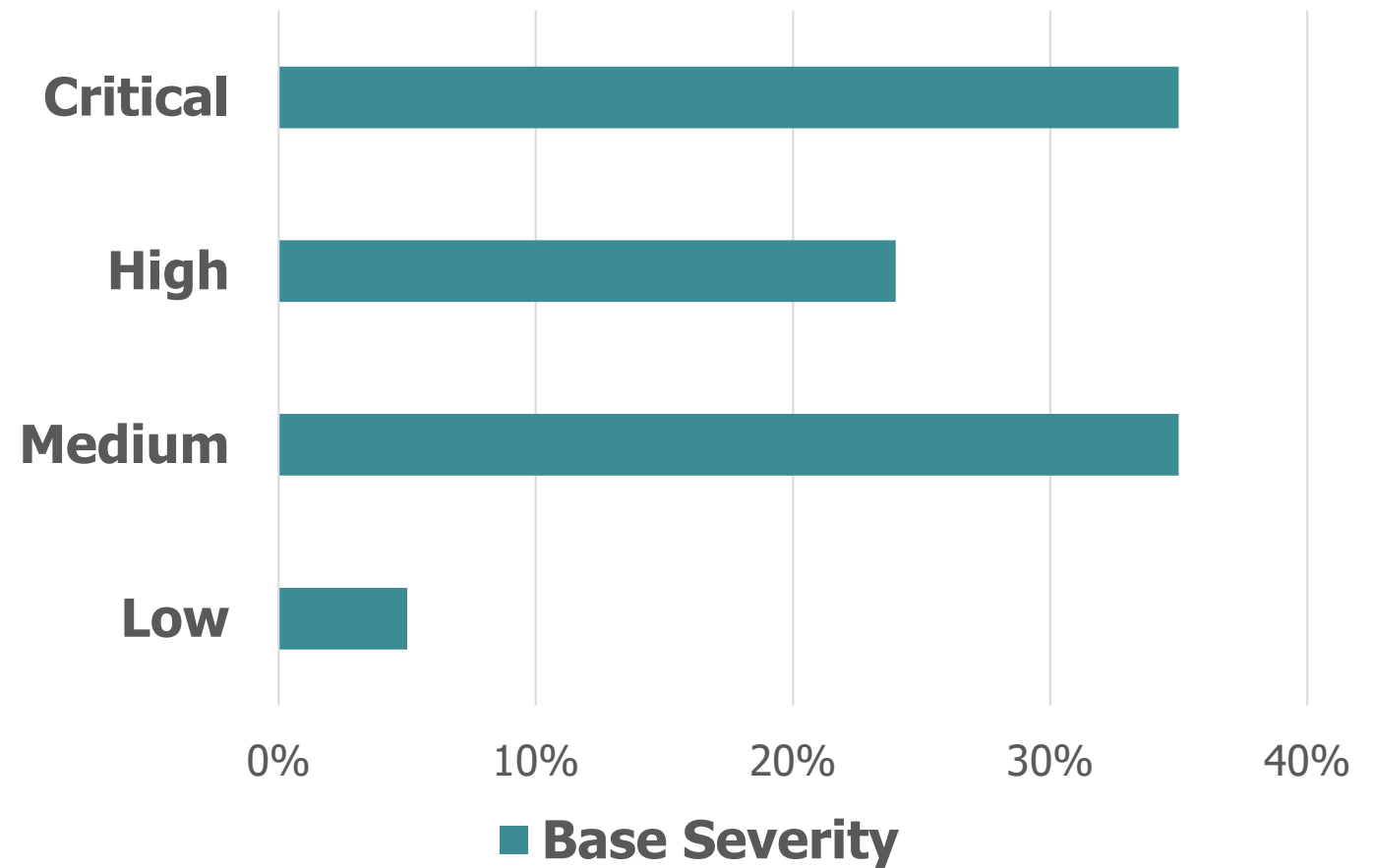
- Most (91%) of the vulnerabilities found are **easy to exploit**



This metric describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability.

Broader Impact: Vulnerabilities in OSS

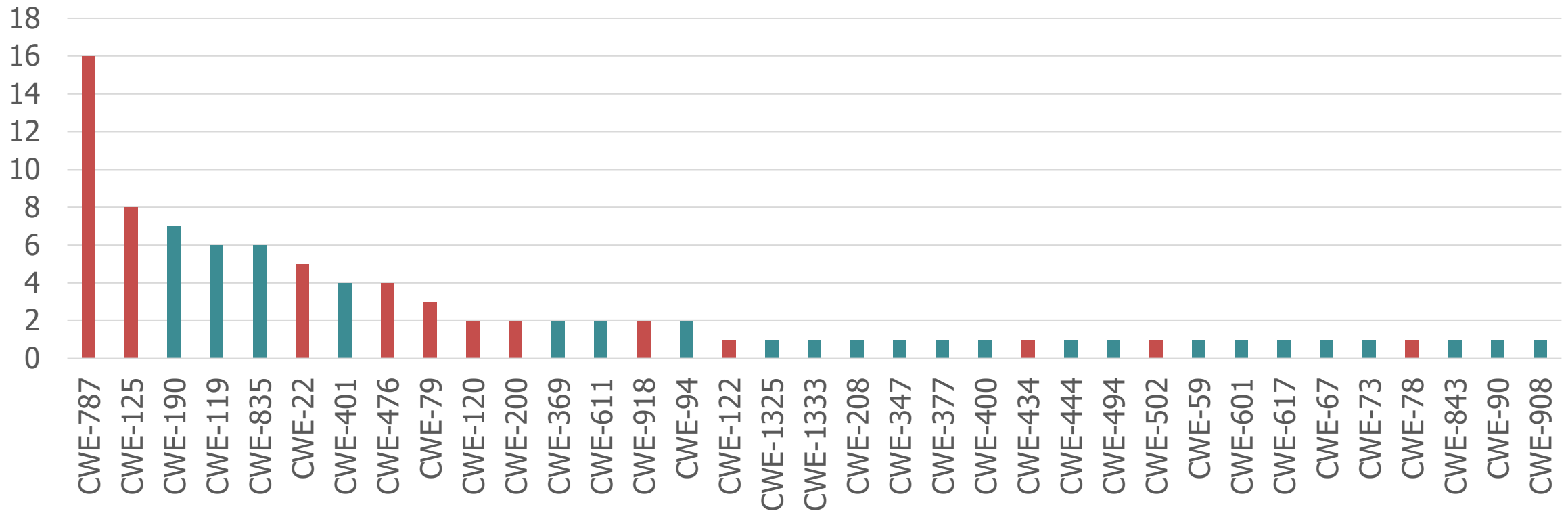
- 35% of the vulnerabilities identified have **critical** (highest) severity
- Most (95%) of the vulnerabilities identified have **at least medium** severity



*This metric reflects a vulnerability's fundamental, unchanging characteristics, calculated from two sub-groups: **Exploitability metrics** (How easy is it to exploit?) and **Impact metrics** (What happens if exploited?)*

Broader Impact: Vulnerabilities in OSS

- Many of the vulnerabilities (~50%) fall under the **Top-25 Most Dangerous Software Weaknesses**



Broader Impact: Data Cleaning Tools, Empirical Studies

Back to the Basics: Rethinking Issue-Commit Linking with LLM-Assisted Retrieval

Huihui Huang[◇], Ratnadira Widyasari[◇], Ting Zhang[▽], Ivana Clairine Irsan[◇], Jieke Shi[◇], Han Wei Ang[★], Frank Liauw[★], Eng Lieh Ouh[◇], Lwin Khin Shar[◇], Hong Jin Kang[★], and David Lo[◇]

[◇]School of Computing and Information Systems, Singapore Management University, Singapore

[▽]Faculty of Information Technology, Monash University, Australia

[★]GovTech, Singapore

[★]School of Computer Science, University of Sydney, Australia

{hhhuang, ratnadiraw.2020, ivanairsan, jiekeshi, elouh, lkshar, davidlo}@smu.edu.sg

ting.zhang@monash.edu, {ang_han_wei, frank_liauw}@tech.gov.sg, hongjin.kang@sydney.edu.au

ICSE 2026



CLEANVUL: Toward High-Quality Function-Level Vulnerability Datasets via LLM-Based Noise Reduction

YIKUN LI, Singapore Management University, Singapore

TING ZHANG, Singapore Management University, Singapore

RATNADIRA WIDYASARI, Singapore Management University, Singapore

YAN NAING TUN, Singapore Management University, Singapore

HUU HUNG NGUYEN, Singapore Management University, Singapore

TAN BUI, Singapore Management University, Singapore

IVANA CLAIRINE IRSAN, Singapore Management University, Singapore

YIRAN CHENG, Singapore Management University, Singapore

XIANG LAN, North Carolina State University, United States

HAN WEI ANG, GovTech, Singapore

FRANK LIAUW, GovTech, Singapore

MARTIN WEYSSOW, Singapore Management University, Singapore

HONG JIN KANG, Singapore Management University, Singapore

ENG LIEH OUH, Singapore Management University, Singapore

LWIN KHIN SHAR, Singapore Management University, Singapore

DAVID LO, Singapore Management University, Singapore



Broader Impact: New Benchmarks

Out of Distribution, Out of Luck: How Well Can LLMs Trained on Vulnerability Datasets Detect Top 25 CWE Weaknesses?

Yikun Li*, Ngoc Tan Bui*, Ting Zhang[♡], Chengran Yang*, Xin Zhou*, Martin Weyssow*, Jinfeng Jiang*, Junkai Chen*, Huihui Huang*, Huu Hung Nguyen*, Chiok Yew Ho[§], Jie Tan[‡], Ruiyin Li[¶], Yide Yin[†], Han Wei Ang[†], Frank Liauw[†], Eng Lieh Ouh*, Lwin Khin Shar*, David Lo*

*Singapore Management University, Singapore

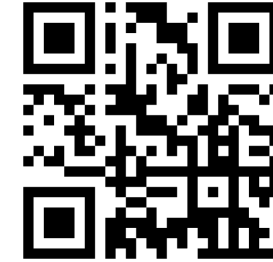
♡ Monash University, Australia

§ Chinese University of Hong Kong, China

‡ University of Groningen, The Netherlands

¶ Wuhan University, China

† GovTech, Singapore



ICSE 2026

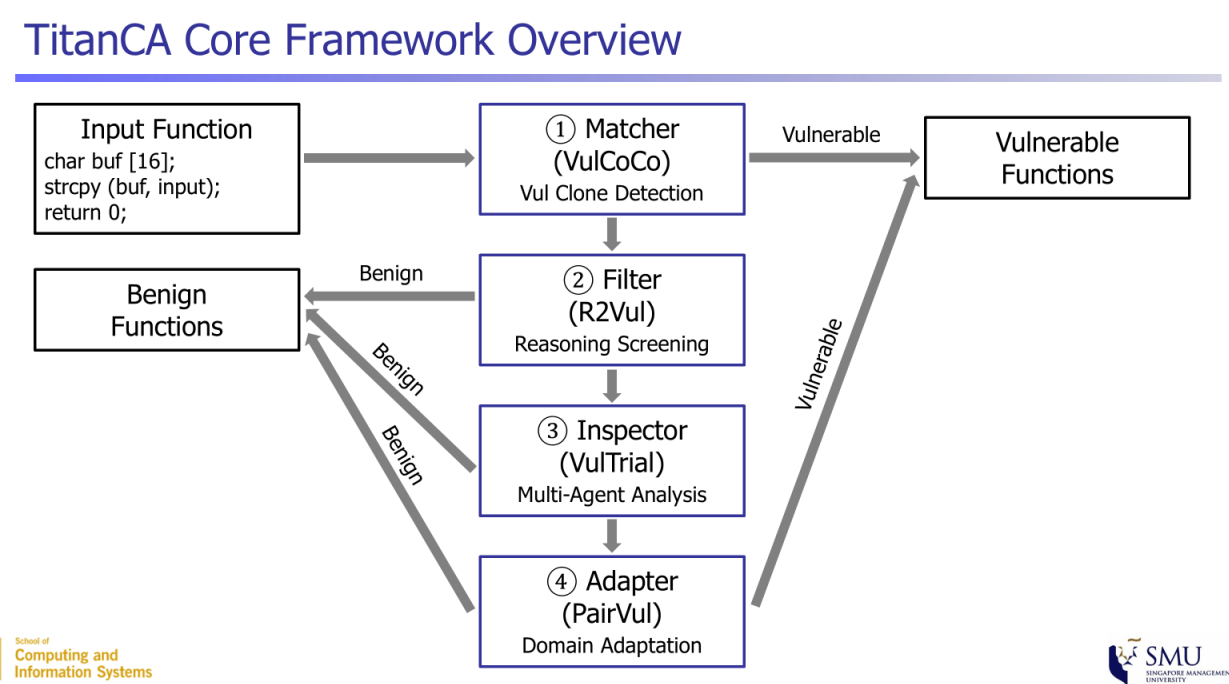
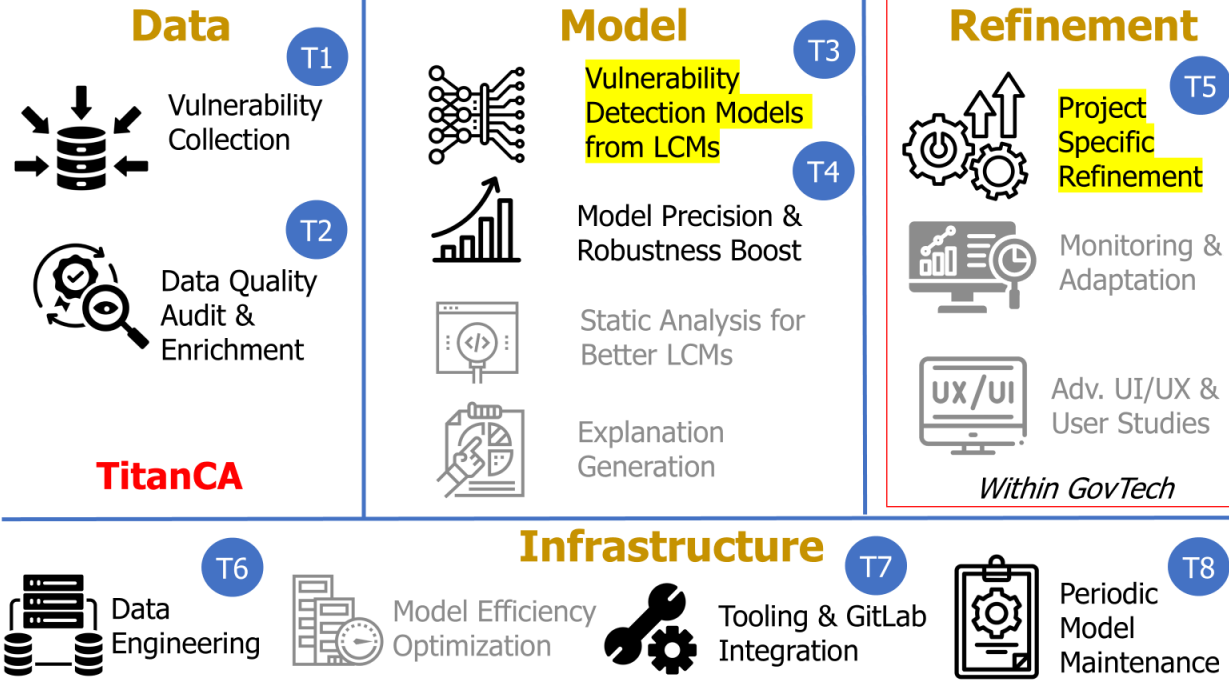
SECUREAGENTBENCH: Benchmarking Secure Code Generation under Realistic Vulnerability Scenarios

Junkai Chen^{1*}, Huihui Huang^{1*}, Yunbo Lyu¹, Junwen An², Jieke Shi¹, Chengran Yang¹, Ting Zhang³, Haoye Tian⁴, Yikun Li¹, Zhenhao Li⁵, Xin Zhou¹, Xing Hu⁶, David Lo¹

¹Singapore Management University, ²National University of Singapore, ³Monash University, ⁴Aalto University, ⁵York University, ⁶Zhejiang University







Outcomes: Refinement Pillar

Operation performance

+109%

F0.3 score improvement vs. popular SAST tools
(Target: +30%)

3.6x target achieved

Broader Impact: Vulnerabilities in OSS

- Verified Real-World Vulnerabilities Identified Using TitanCA
 - Website link: <https://titancaproject.github.io/cves.html>

As of Jan 7, 2026

151
Vulnerabilities Found & Fixed

108
CVEs in Total

35
CVEs Published

73
CVEs in Pipeline

Discovered Vulnerabilities

The TITAN project has successfully identified and disclosed the following Common Vulnerabilities and Exposures (CVEs) through our intelligent vulnerability discovery system.

37 Published 75 In Progress

CVE-2025-14306

CRITICAL CWE-22

A directory traversal vulnerability exists in the CacheCleaner component of Robocode version 1.9.3.6. The recursivelyDelete method fails to properly sanitize file paths, allowing attackers to traverse directories and delete arbitrary files on the system. This vulnerability can be exploited by sub...

Published: 2025-12-09

CVE-2025-14307

CRITICAL CWE-377

An insecure temporary file creation vulnerability exists in the AutoExtract component of Robocode version 1.9.3.6. The createTempFile method fails to securely create temporary files, allowing attackers to exploit race conditions and potentially execute arbitrary code or overwrite critical files. ...

Published: 2025-12-09

CVE-2025-14308

CRITICAL CWE-190

An integer overflow vulnerability exists in the write method

CVE-2025-14310

CRITICAL CWE-120

Buffer Copy without Checking Size of Input ("Classic Buffer

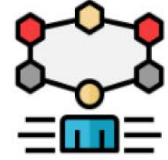
Acknowledgements



Acknowledgements



Advertisement: TitanCA 2.0 (3.5M; 2026-2028)

Analysis	Detection	Remediation	Operations
 WP1: LLM-Augmented Static Analysis for CPG Construction	 WP3: Agentic LLM for Interprocedural Vulnerability Detection	 WP5: Fine-Grained CWE Classification	 WP8: Workflow Integration
 WP2: Semantic Enrichment and Interprocedural Context Extraction	 WP4: Rapid Triage with GNNs	 WP6: Patch Generation	 WP9: MLOps
		 WP7: Vulnerability Report Generation	

Looking for Research Fellows, PhD Students, Visiting Research Students, and Collaborators



Thank you!



`unsafe_copy(input, d)`



Buffer overflow risk



`strcpy(d, output)`



CWE-119: Improper Restriction of Operations

Questions? Comments? Advice?
davidlo@smu.edu.sg