Enhancing Security in Third-Party Library Reuse：Comprehensive Detection of 1-day Vulnerability through Code Patch Analysis

Shangzhi Xu, Jialiang Dong, Weiting Cai, Juanru Li, Arash Shaghaghi, Nan Sun, Siqi Ma*

February 27,2025

NDSS SYMPOSIUM

UNSW CANBERRA

CSIRO

UNSW SYDNEY

TU Delft
Delft
University of
Technology

Contents

1. Background
   - Motivation
   - Research Questions
   - Previous Works

2. VULTURE Overview
   - Database Construction
   - TPL Reuse Detection
   - 1-day Vulnerability Detection

3. Evaluation and Results
   - Database Quality
   - Benchmark Vulnerability Detection
   - Vulnerability Detection In the Wild

Contents

1. Background
   - Motivation
   - Research Questions
   - Previous Works

2. VULTURE Overview
   - Database Construction
   - TPL Reuse Detection
   - 1-day Vulnerability Detection

3. Evaluation and Results
   - Database Quality
   - Benchmark Vulnerability Detection
   - Vulnerability Detection In the Wild

UNSW CANBERRA

Motivation

Current approach to code reusing

Motivation

Current approach to code reusing

Library developer

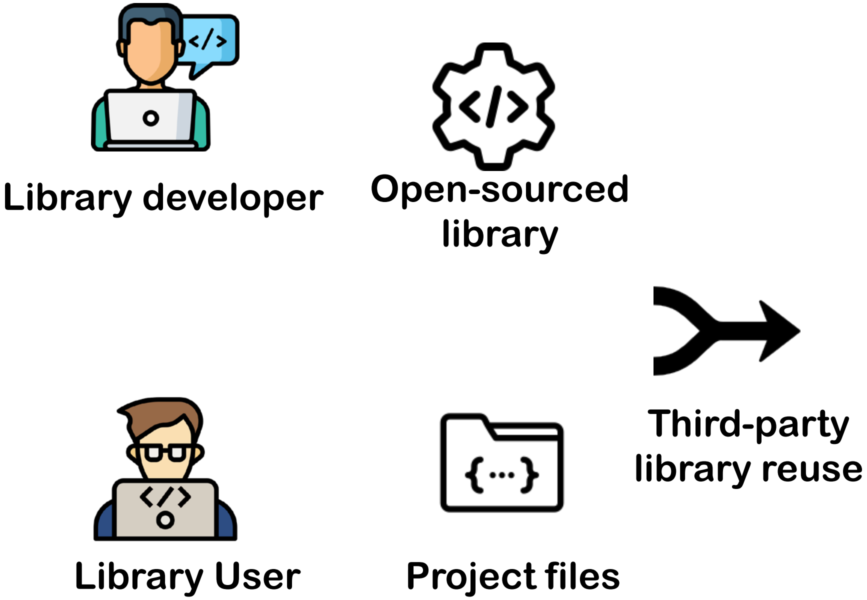
Open-sourced library

Library User

Project files

Motivation

Current approach to code reusing



Library developer

Open-sourced library

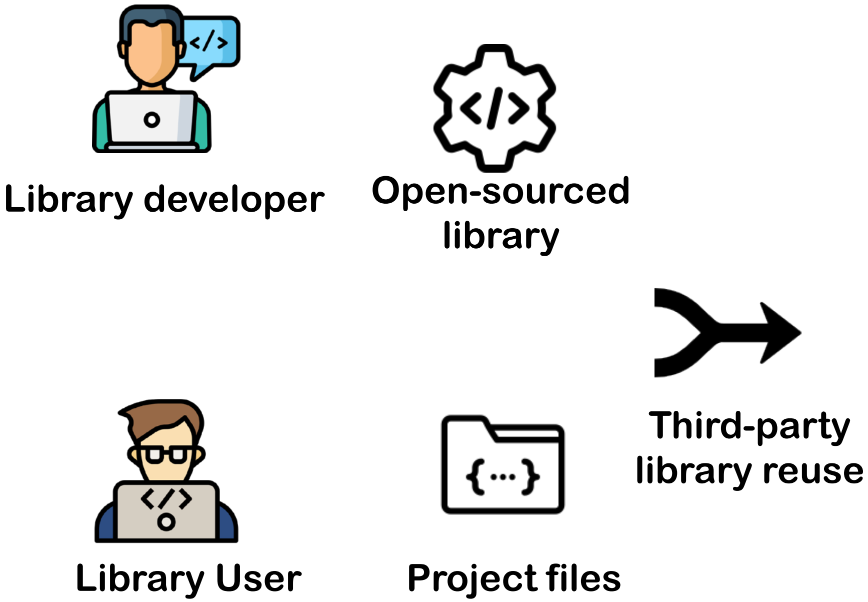
Third-party library reuse

Library User

Project files

Motivation

Current approach to code reusing



Library developer

Open-sourced library

Third-party library reuse

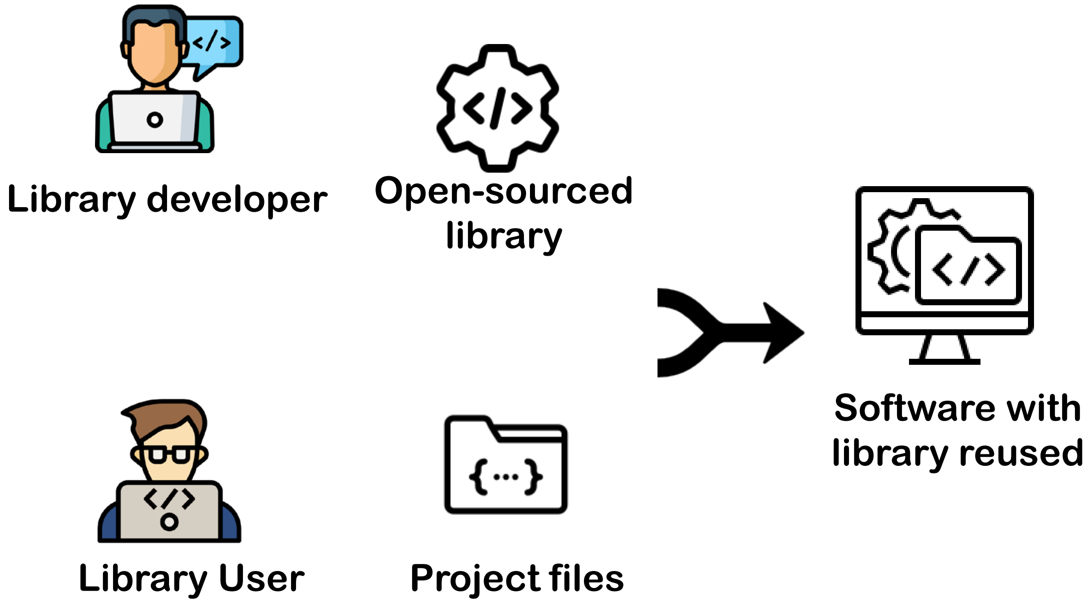
Library User

Project files

For brevity, Third-party libraries will be referred to as TPLs in the following

Motivation

Current approach to code reusing



Library developer

Open-sourced library

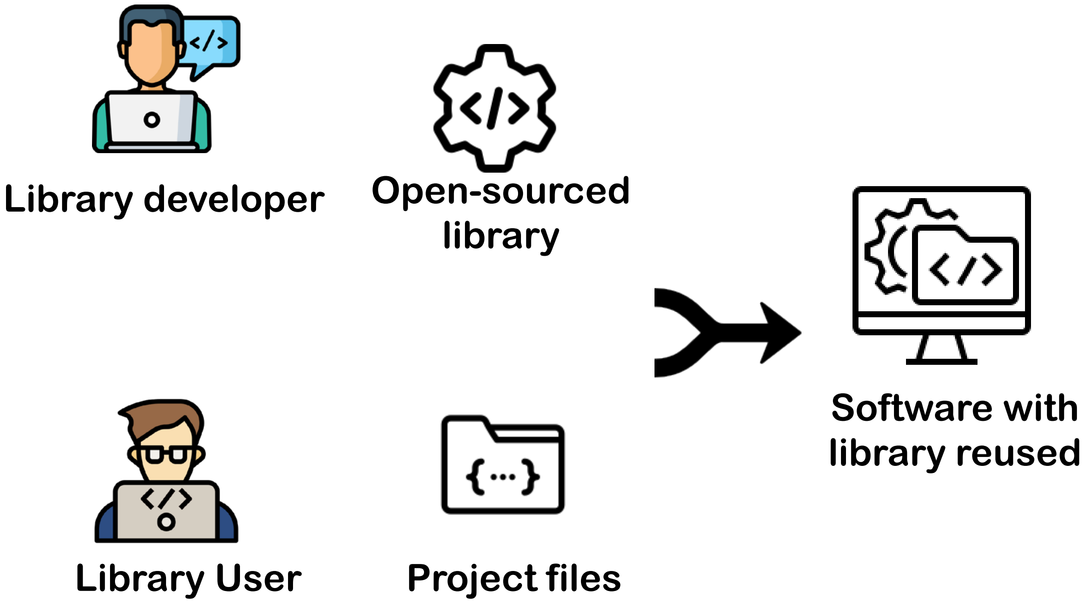
Library User

Project files

Software with library reused

Motivation

Current approach to code reusing



Library developer

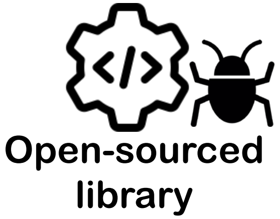
Open-sourced library

Software with library reused

Library User

Project files

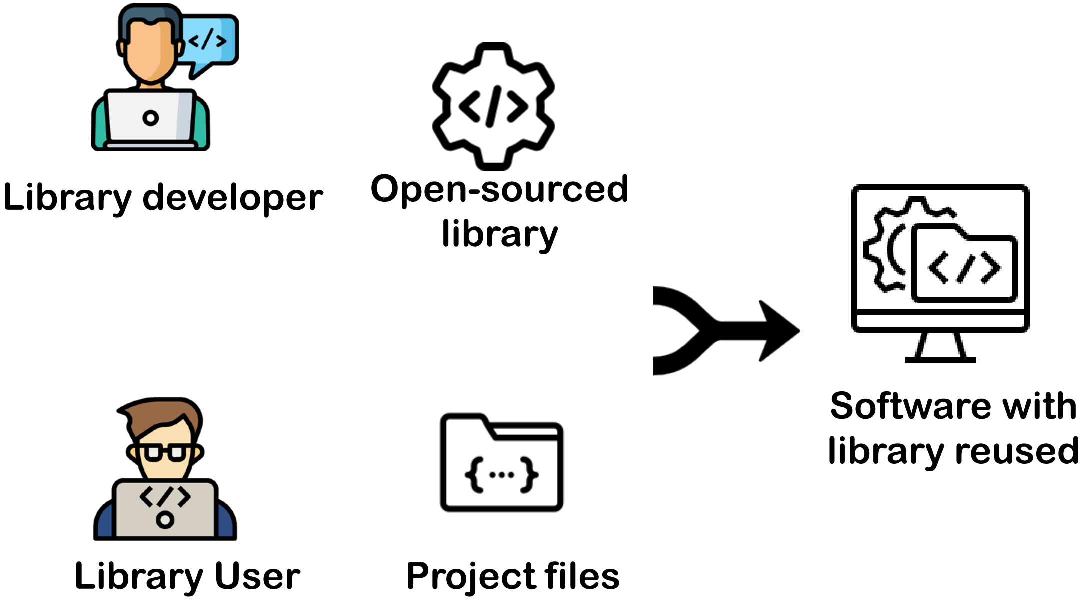
Patching vulnerabilities



Open-sourced library

Motivation

Current approach to code reusing



Library developer

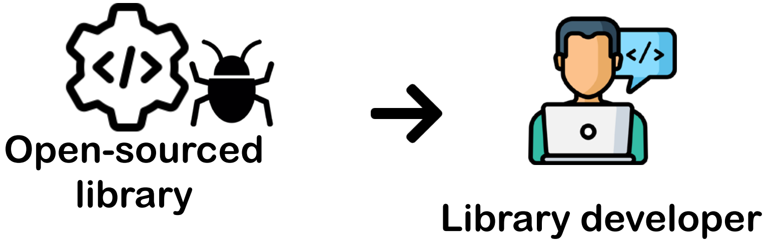
Open-sourced library

Library User

Project files

Software with library reused

Patching vulnerabilities

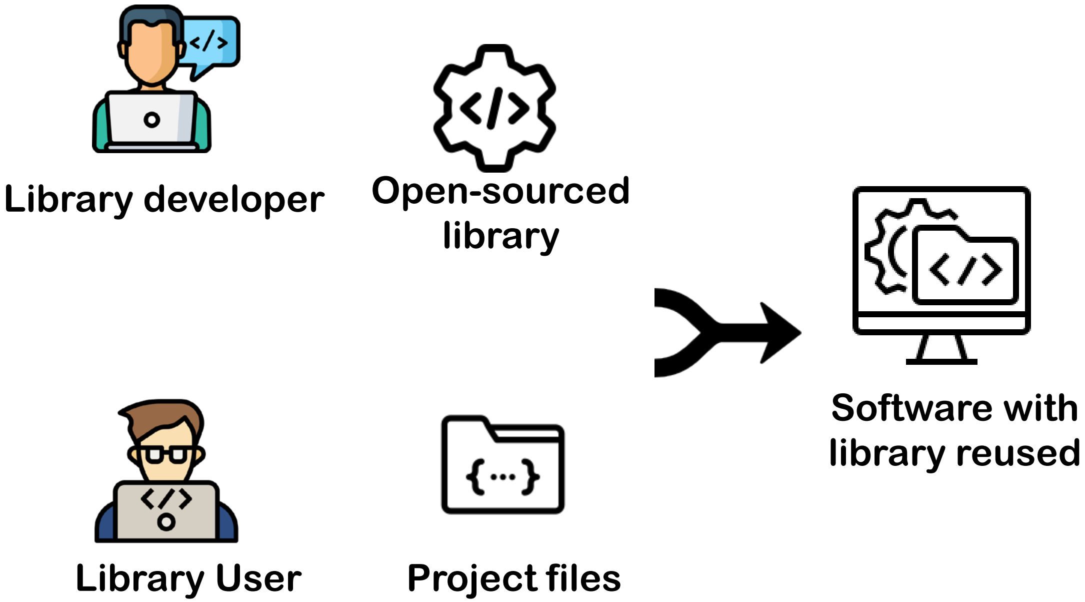


Open-sourced library

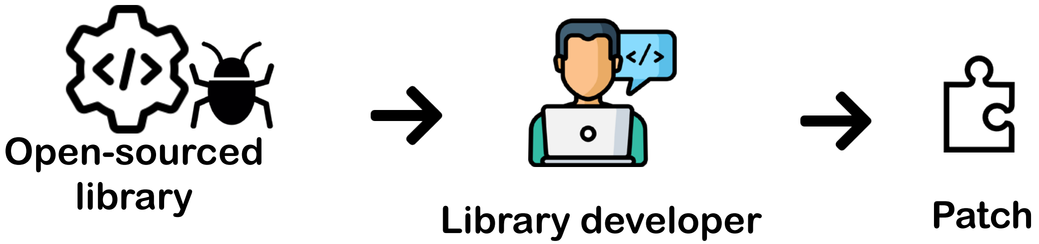
Library developer

Motivation

Current approach to code reusing

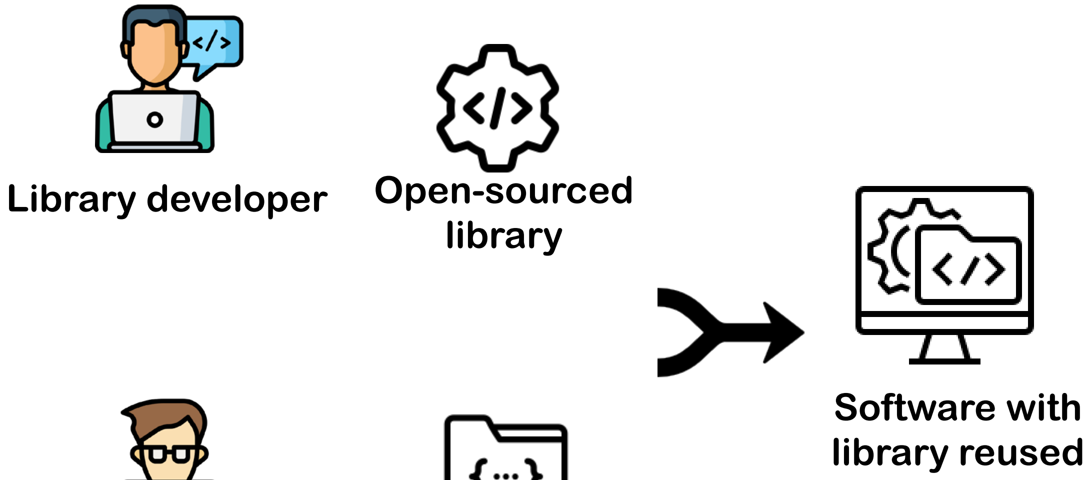


Patching vulnerabilities



Motivation

Current approach to code reusing



Library developer

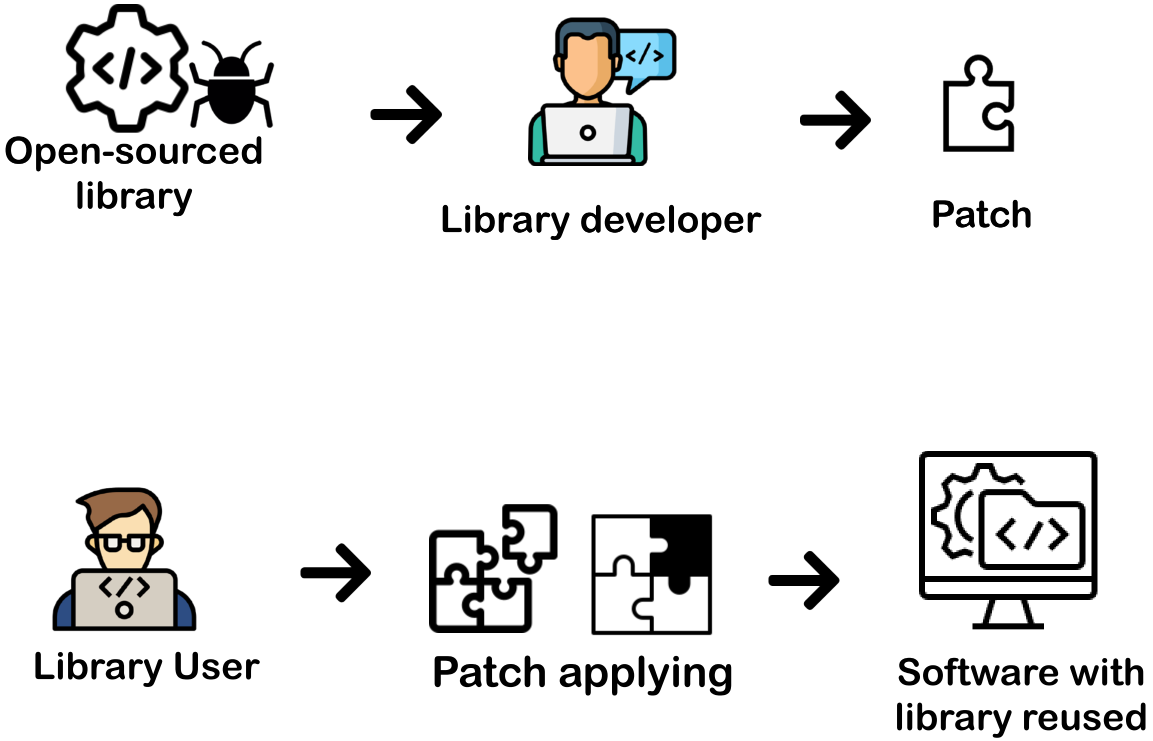
Open-sourced library

Library User

Project files

Software with library reused

Patching vulnerabilities



Open-sourced library

Library developer

Patch

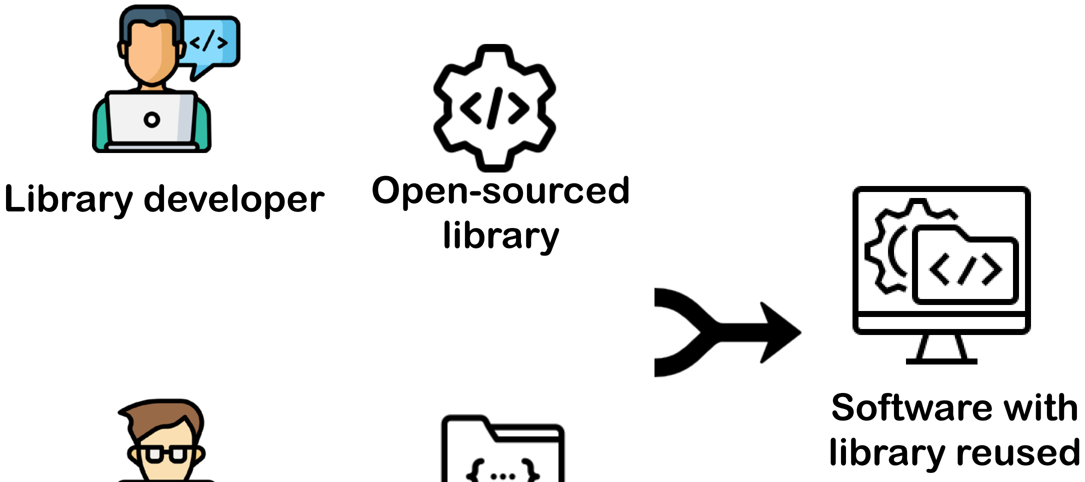
Library User

Patch applying

Software with library reused

Motivation

Current approach to code reusing



Library developer

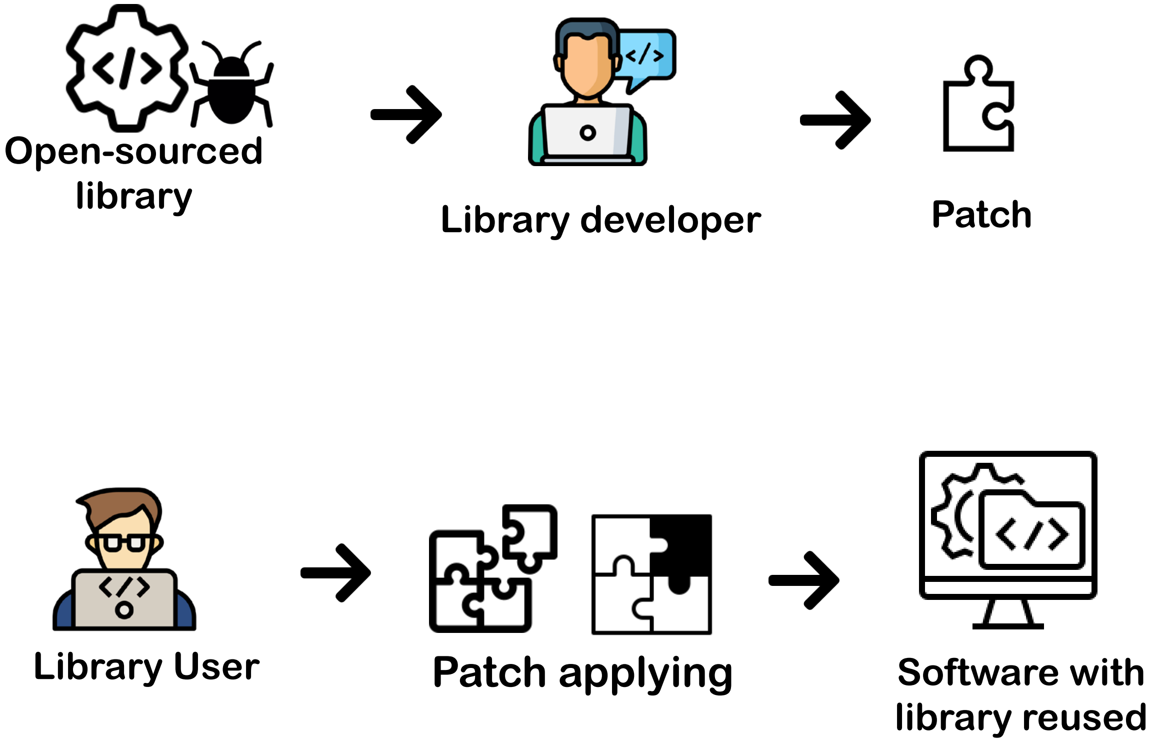
Open-sourced library

Library User

Project files

Software with library reused

Patching vulnerabilities



Open-sourced library

Library developer

Patch

Library User

Patch applying

Software with library reused

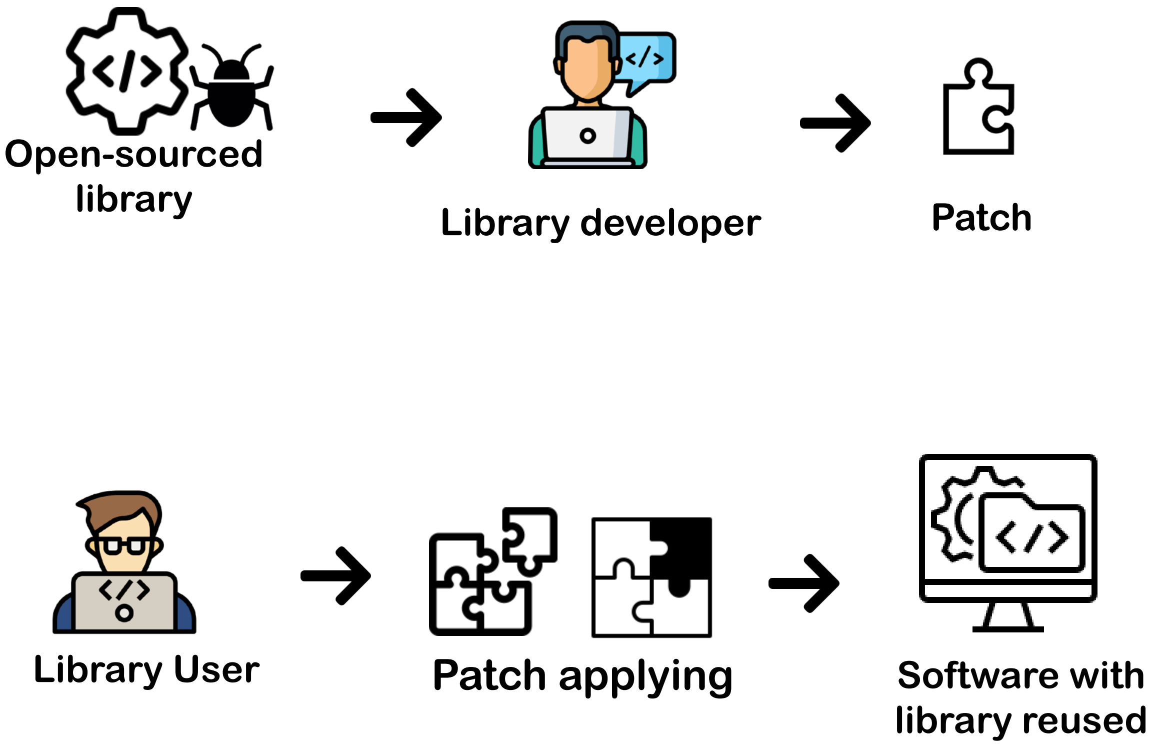
A 1-day vulnerability might propagate to downstream software if not properly patched.

Motivation

Current approach to code reusing

ReactOS

Patching vulnerabilities



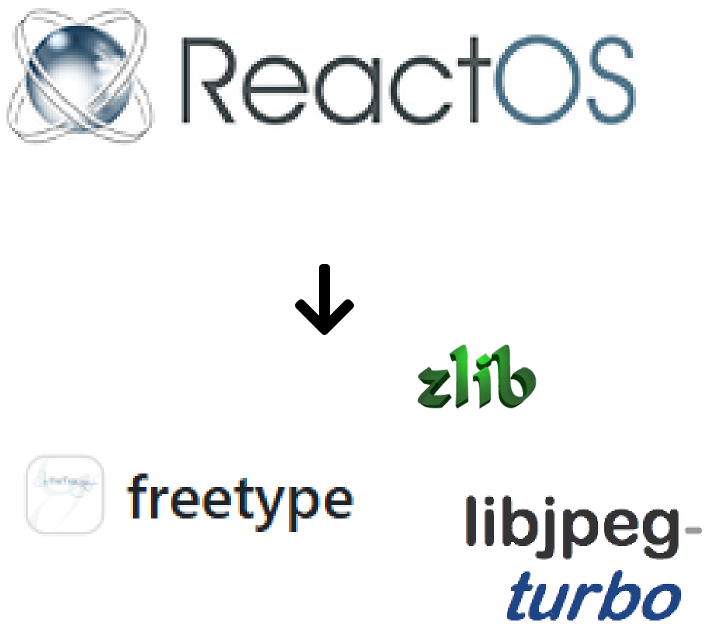
Open-sourced library → Library developer → Patch

Library User → Patch applying → Software with library reused

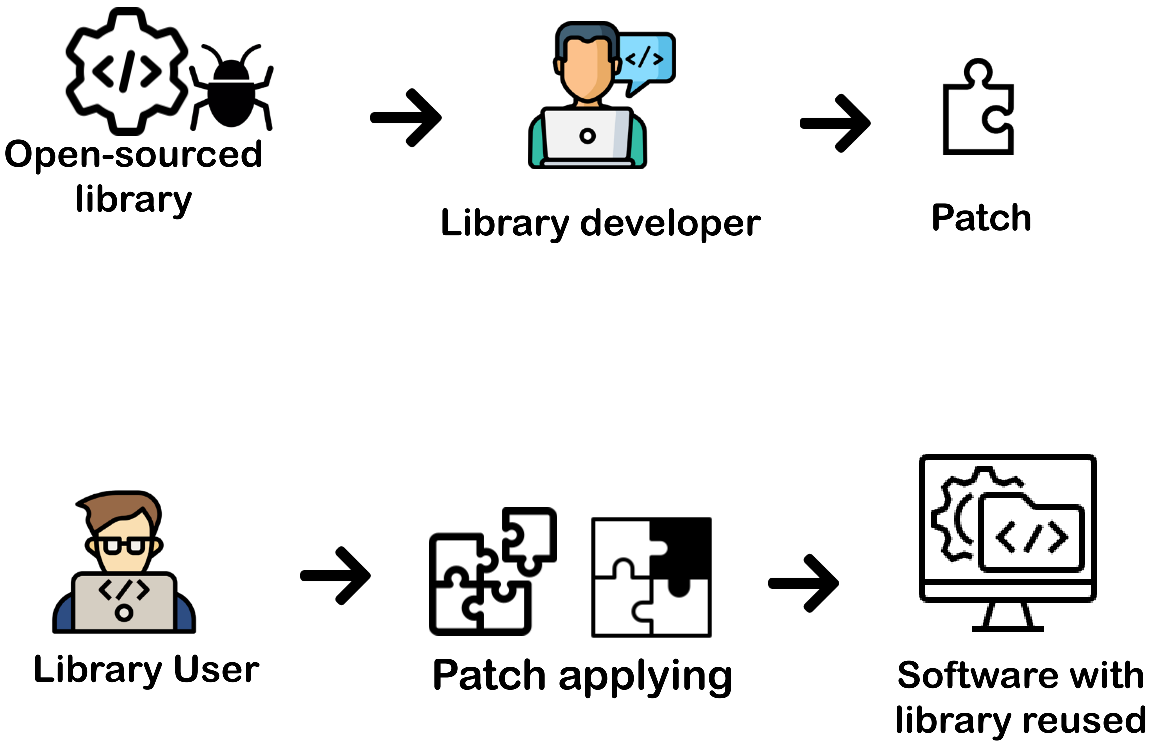
A 1-day vulnerability might propagate to downstream software if not properly patched.

Motivation

Current approach to code reusing



Patching vulnerabilities



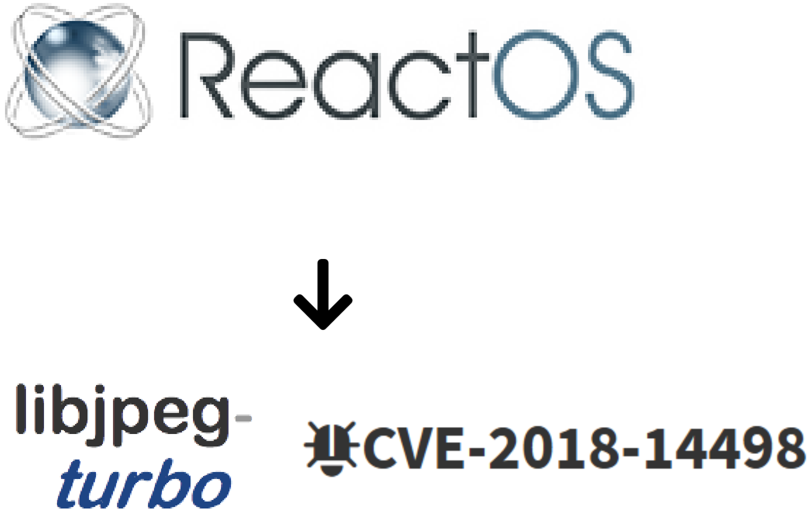
Open-sourced library → Library developer → Patch

Library User → Patch applying → Software with library reused

A 1-day vulnerability might propagate to downstream software if not properly patched.

Motivation

Current approach to code reusing



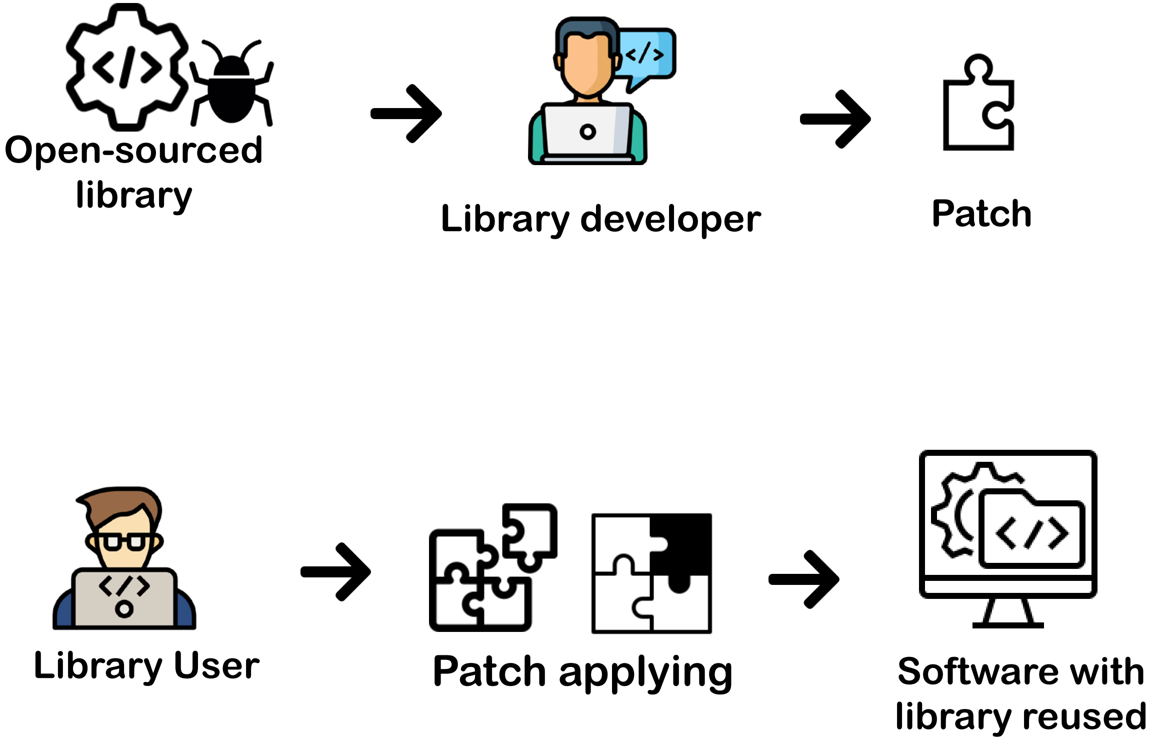
ReactOS

↓

libjpeg-turbo

CVE-2018-14498

Patching vulnerabilities



Open-sourced library → Library developer → Patch

Library User → Patch applying → Software with library reused

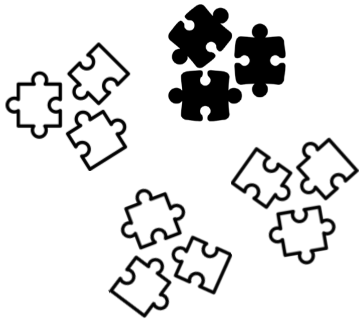
A 1-day vulnerability might propagate to downstream software if not properly patched.

Research Question

As the code base grows...

Research Question

As the code base grows...

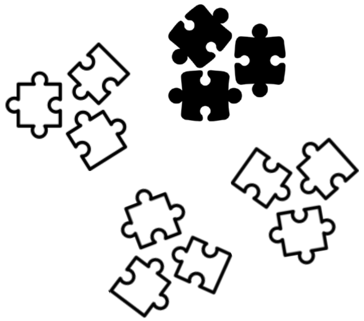


Large-scale patch collection requires significant human effort.

①

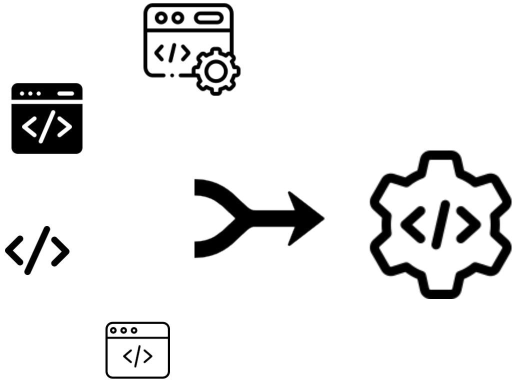
Research Question

As the code base grows...



Large-scale patch collection requires significant human effort.

①

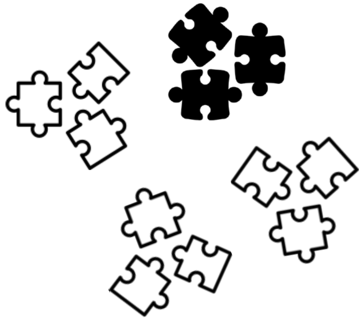


Libraries may also be nested and reuse other libraries.

②

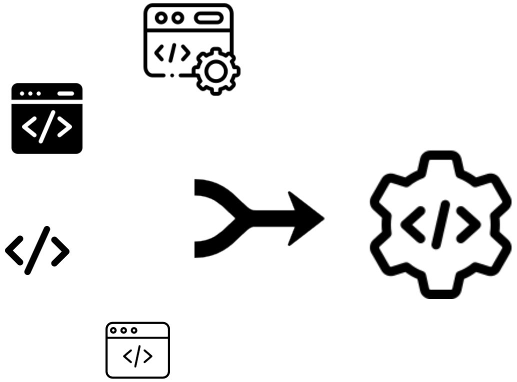
Research Question

As the code base grows...



Large-scale patch collection requires significant human effort.

①



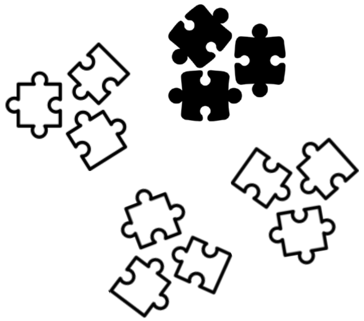
Libraries may also be nested and reuse other libraries.

②

ReactOS

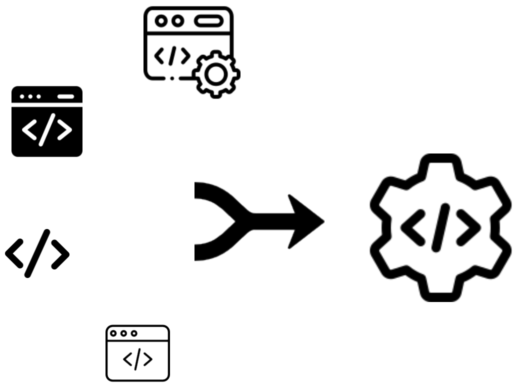
Research Question

As the code base grows...



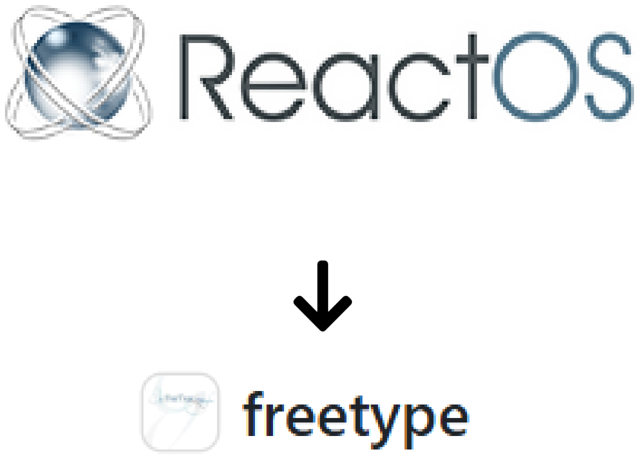
Large-scale patch collection requires significant human effort.

①



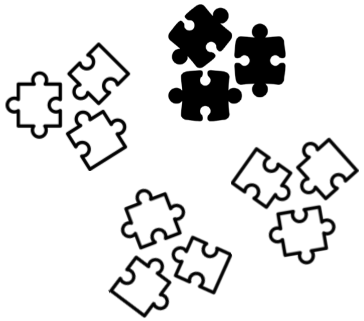
Libraries may also be nested and reuse other libraries.

②



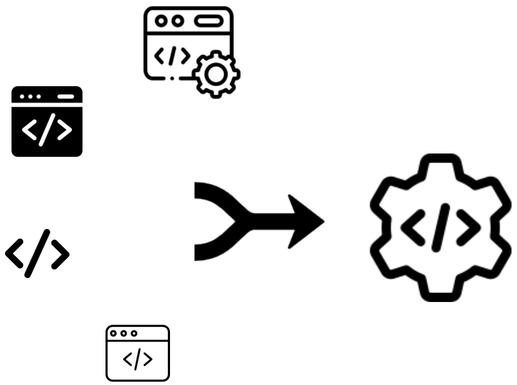
Research Question

As the code base grows...



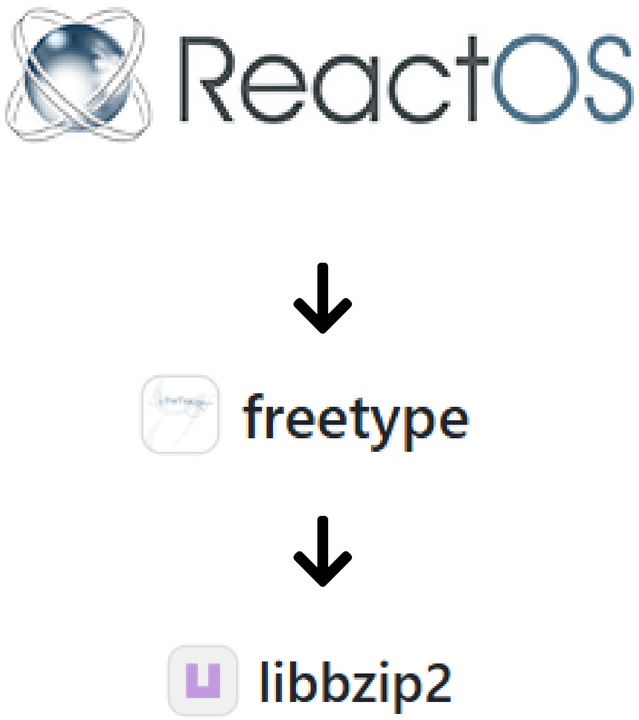
Large-scale patch collection requires significant human effort.

①



Libraries may also be nested and reuse other libraries.

②



ReactOS

↓

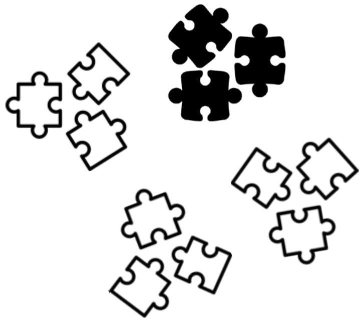
freetype

↓

libbzip2

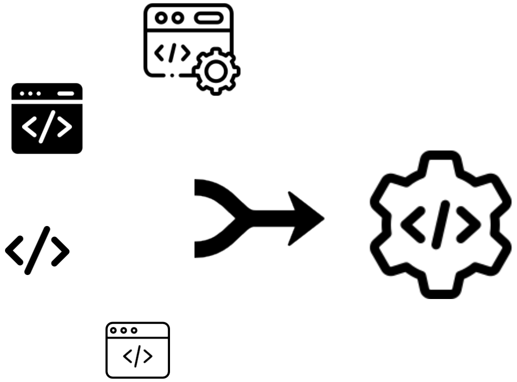
Research Question

As the code base grows...



Large-scale patch collection requires significant human effort.

①



Libraries may also be nested and reuse other libraries.

②

Hard to determine if a vulnerability will affect the software, as libraries may contain custom modifications.

③

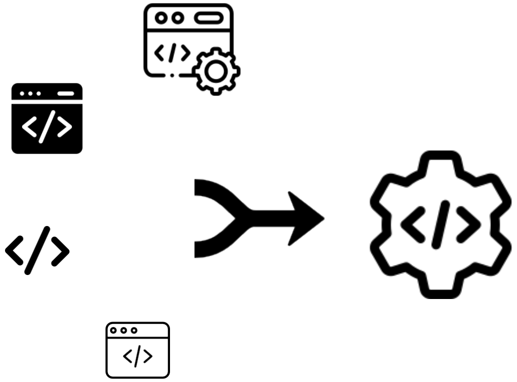
Research Question

As the code base grows...

RQ1 How to efficiently construct a database to support 1-day vulnerability detection

Large-scale patch collection requires significant human effort.

①



Libraries may also be nested and reuse other libraries.

②

Hard to determine if a vulnerability will affect the software, as libraries may contain custom modifications.

③

Research Question

As the code base grows...

RQ1 How to efficiently construct a database to support 1-day vulnerability detection

Large-scale patch collection requires significant human effort.

①

RQ2 How to identify reused TPLs in target program

Libraries may also be nested and reuse other libraries.

②

Hard to determine if a vulnerability will affect the software, as libraries may contain custom modifications.

③

Research Question

As the code base grows...

RQ1 How to efficiently construct a database to support 1-day vulnerability detection

Large-scale patch collection requires significant human effort.

①

RQ2 How to identify reused TPLs in target program

Libraries may also be nested and reuse other libraries.

②

RQ3 How to accurately identify 1-day vulnerabilities efficiently even with custom modifications

Hard to determine if a vulnerability will affect the software, as libraries may contain custom modifications.

③

Research Question

RQ1 How to efficiently construct a database to support 1-day vulnerability detection

Observation 1: A suitable database for 1-day vulnerability detection must be comprehensive, specific, and maintainable. And LLMs can minimize manual efforts of data collection.

Research Question

RQ1 How to efficiently construct a database to support 1-day vulnerability detection

Observation 1: A suitable database for 1-day vulnerability detection must be comprehensive, specific, and maintainable. And LLMs can minimize manual efforts of data collection.

RQ2 How to identify reused Third-party libraries in target program

Observation 2: TPL reuses can be identified through code similarity comparison. The comparison process should be optimized by incorporating sufficient and accurate TPL information for effective reuse detection.

Research Question

RQ3 How to accurately identify 1-day vulnerabilities efficiently even with custom modifications

Observation 3: A 1-day vulnerability can be identified through patch analysis. Selecting key vulnerable features, such as semantic information, enhances 1-day vulnerability detection.

Previous Work

Previous Work

License-based detection

Previous Work

License-based detection

Code-based detection

Previous Work

License-based detection

Limitations:

- High false negative rate due to poorly maintained licenses by software developers.

Code-based detection

Limitations:

- High false positive rate caused by custom modification.

Previous Work

License-based detection

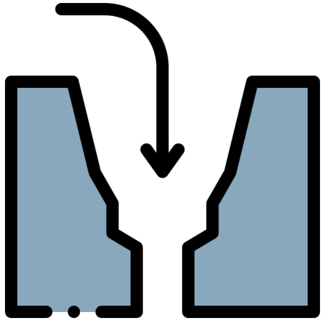
Limitations:

- High false negative rate due to poorly maintained licenses by software developers.

Code-based detection

Limitations:

- High false positive rate caused by custom modification.



Previous Work

License-based detection

Limitations:

- High false negative rate due to poorly maintained licenses by software developers.

Code-based detection

Limitations:

- High false positive rate caused by custom modification.

VULTURE: An accurate 1-day vulnerability detection tool

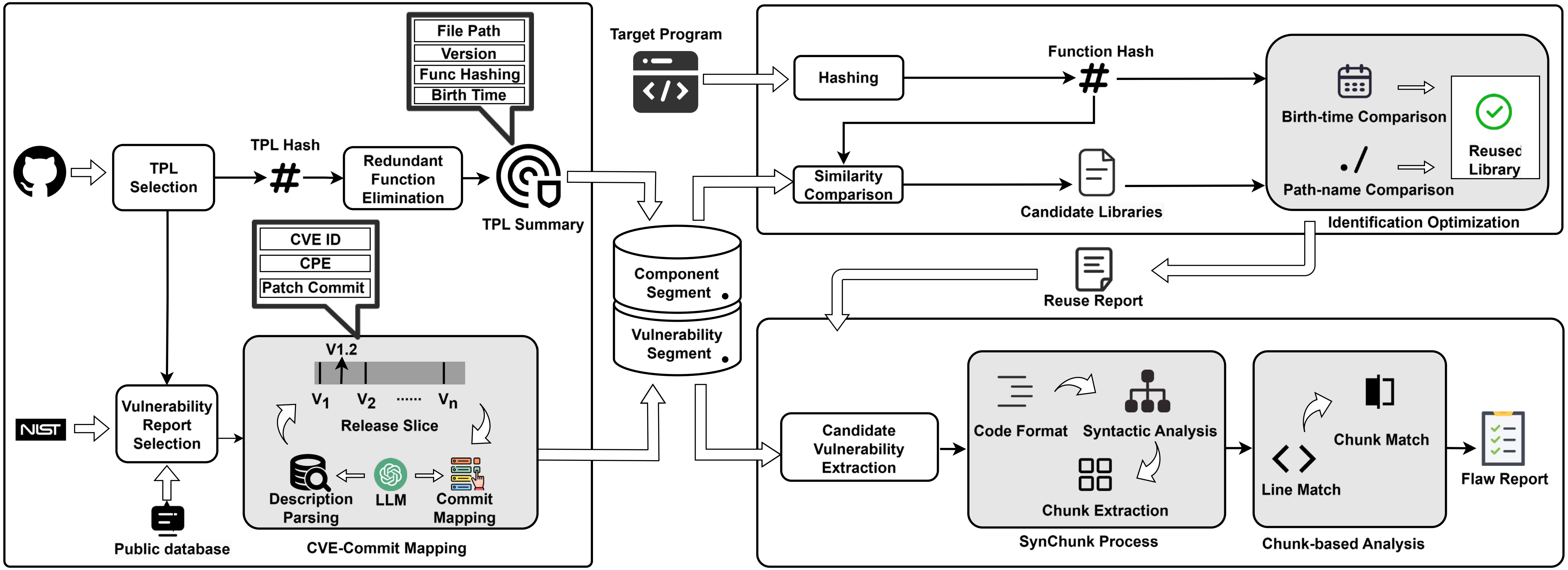
Contents

1. Background
   - Motivation
   - Research Questions
   - Previous Works

2. VULTURE Overview
   - Database Construction
   - TPL Reuse Detection
   - 1-day Vulnerability Detection

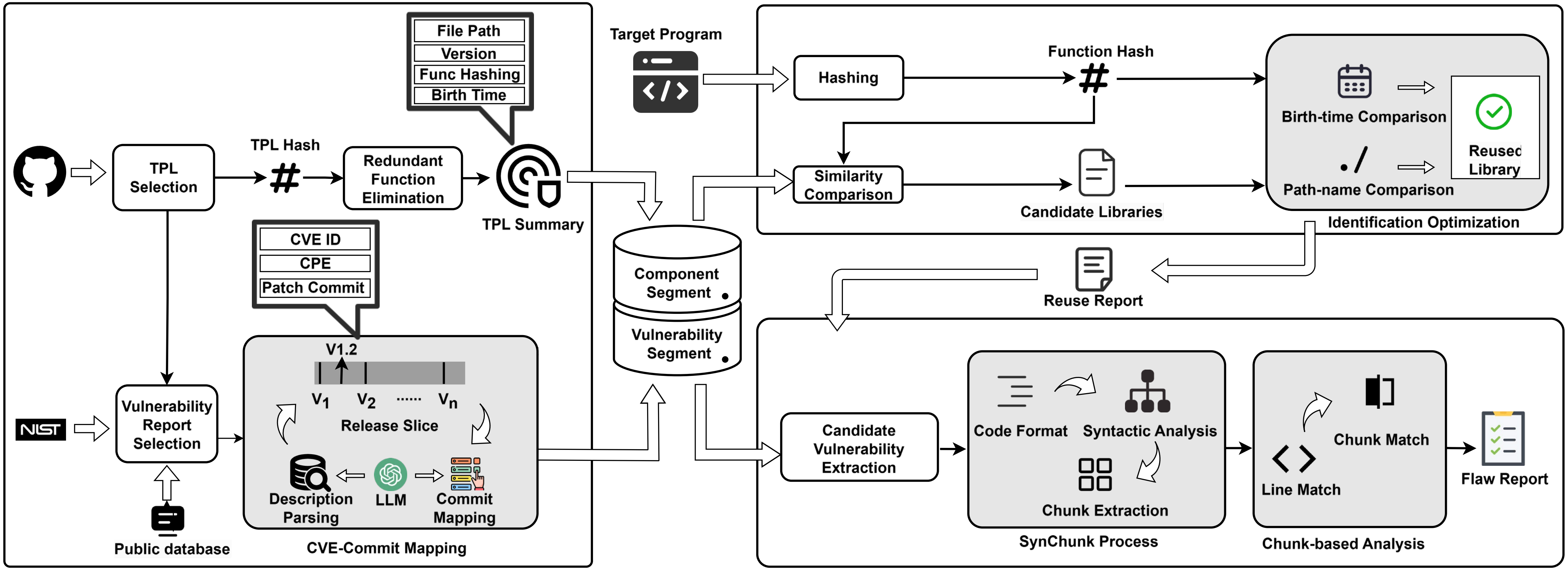
3. Evaluation and Results
   - Database Quality
   - Benchmark Vulnerability Detection
   - Vulnerability Detection In the Wild

Overview of VULTURE



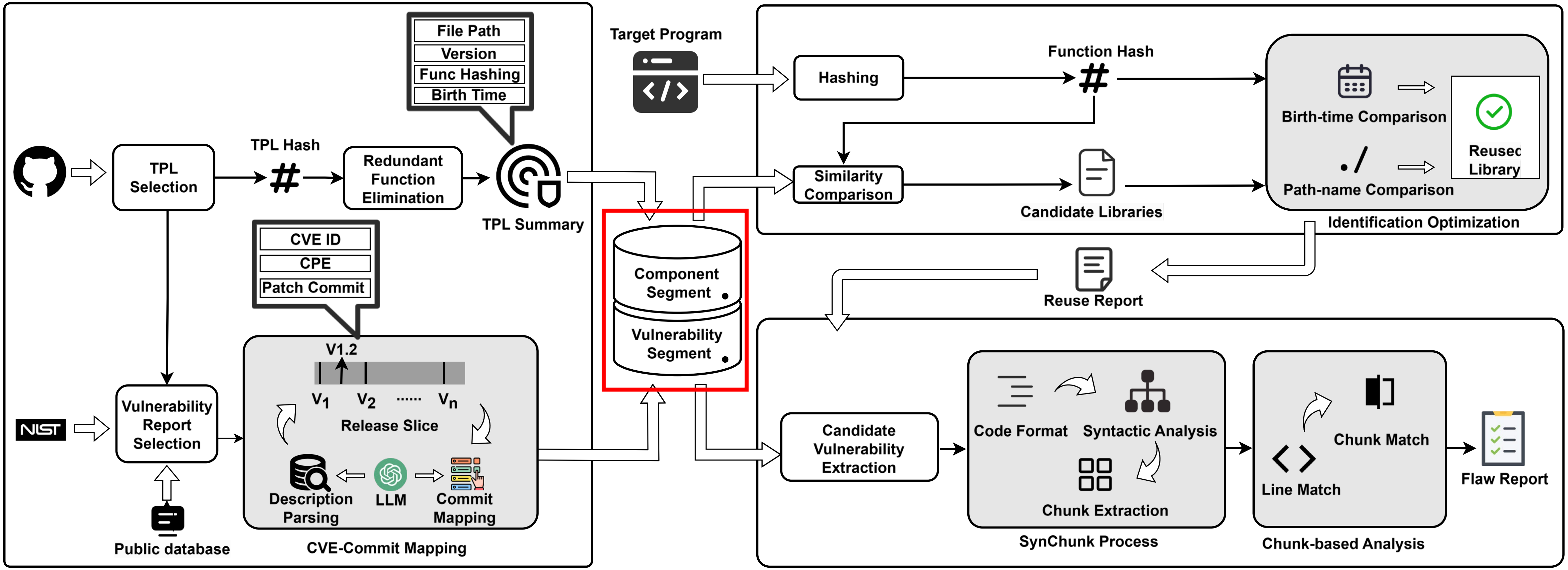
Overview of VULTURE

TPLFILTER Construction(RQ1)

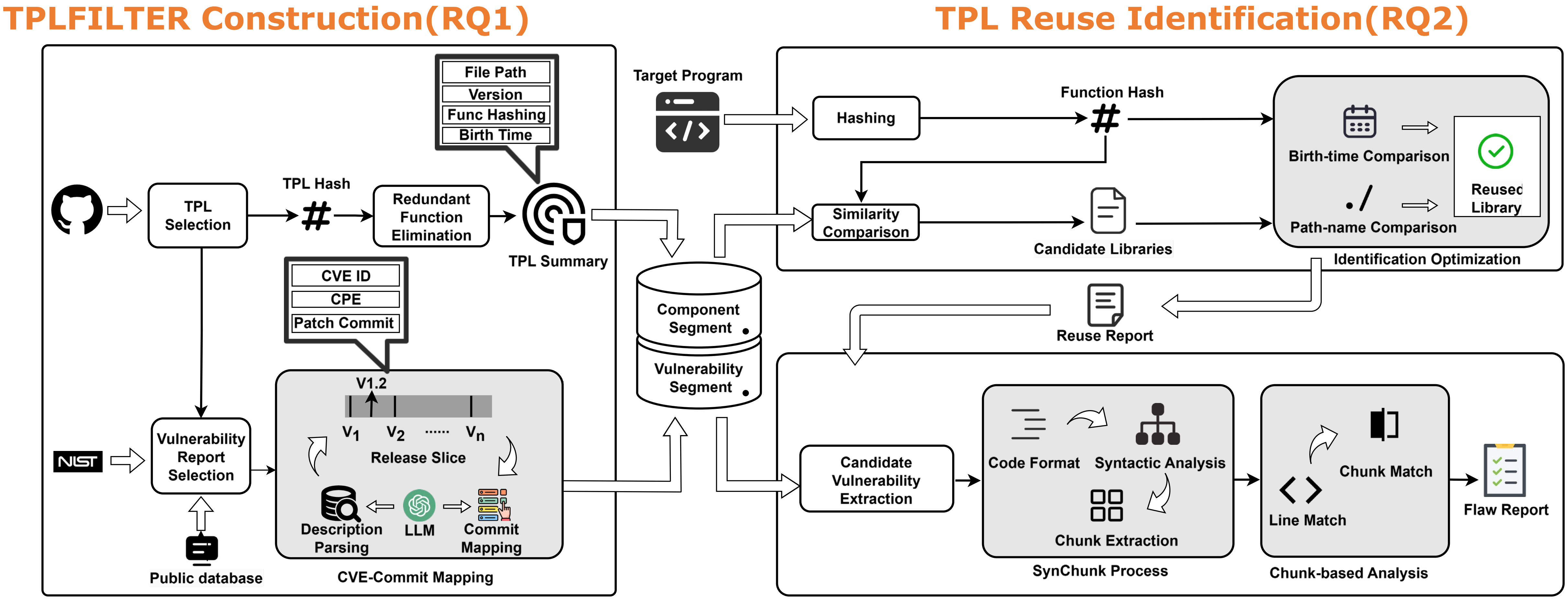


Overview of VULTURE

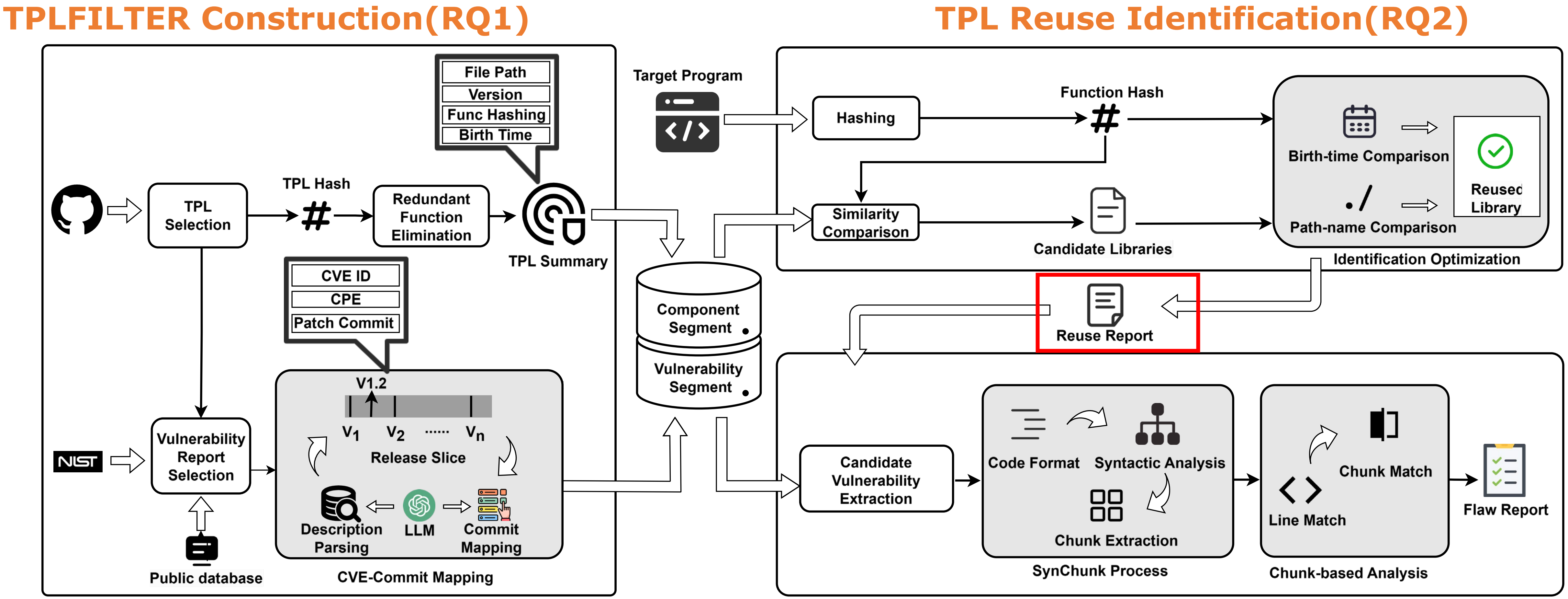
TPLFILTER Construction(RQ1)



Overview of VULTURE

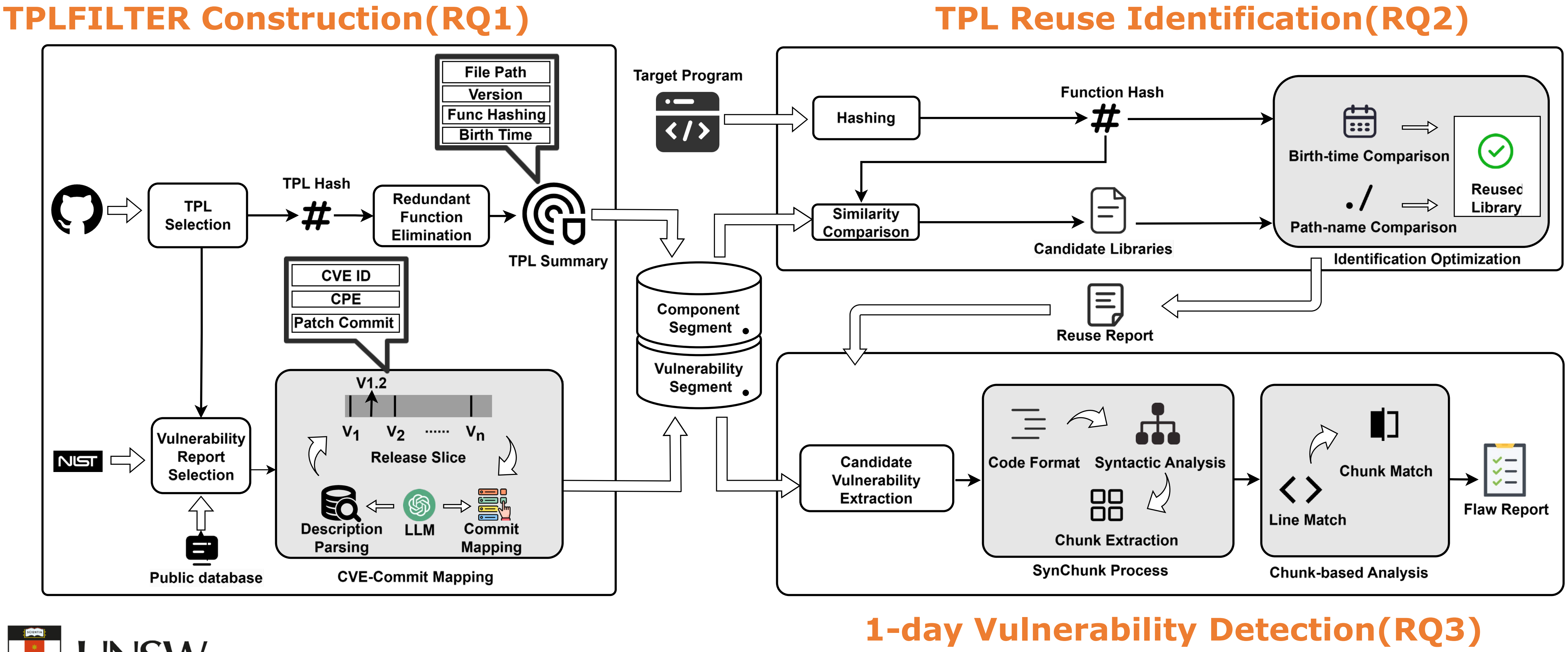


Overview of VULTURE

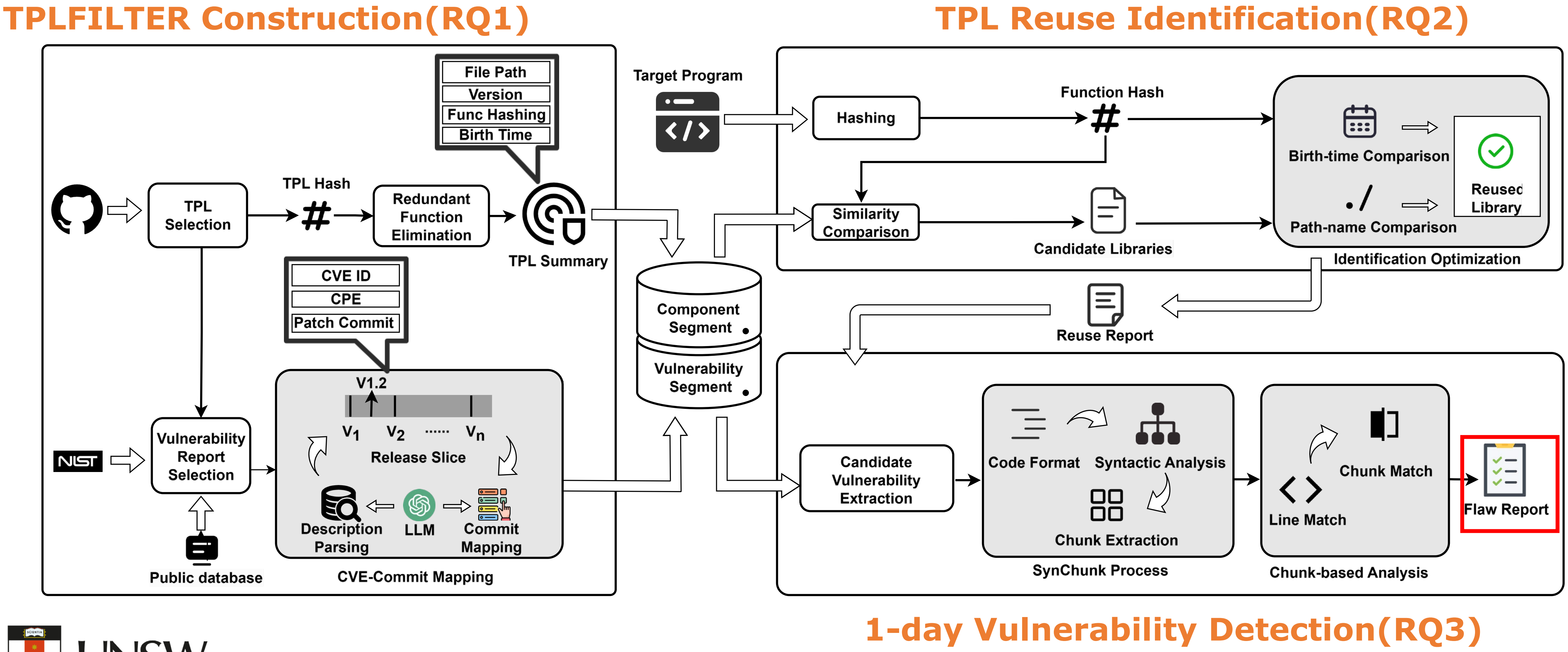


UNSW CANBERRA

Overview of VULTURE



Overview of VULTURE



Overview of VULTURE

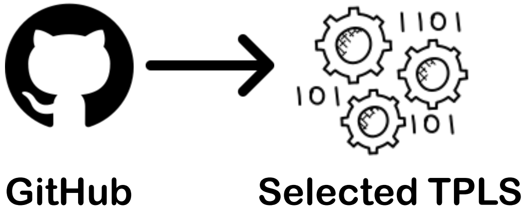
TPLFILTER Construction

Func Summary

GitHub

Overview of VULTURE

TPLFILTER Construction

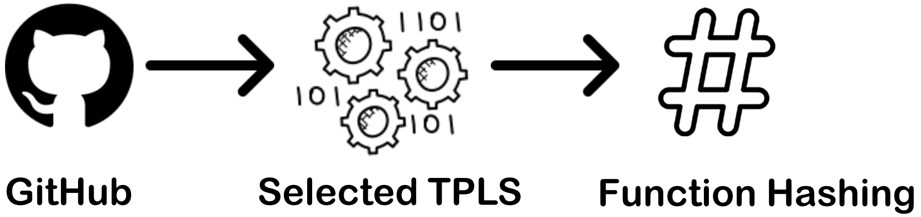


Step 1: TPL Selection

- Select repositories with 100+ stars
- Scan the documents with keywords match to exclude non-library repositories

Overview of VULTURE

TPLFILTER Construction



Step 1: TPL Selection

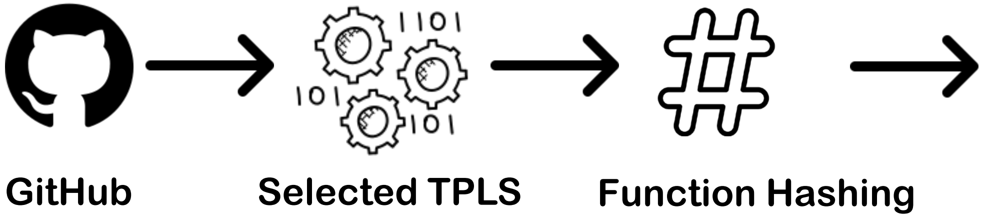
- Select repositories with 100+ stars
- Scan the documents with keywords match to exclude non-library repositories

Step 2: Function Hashing

- Extract each function in selected TPLs to calculate hashing value.

Overview of VULTURE

TPLFILTER Construction

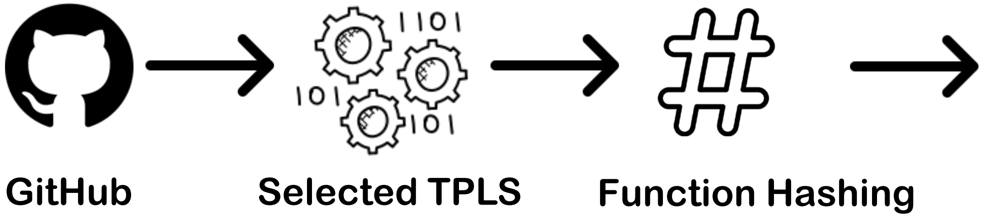


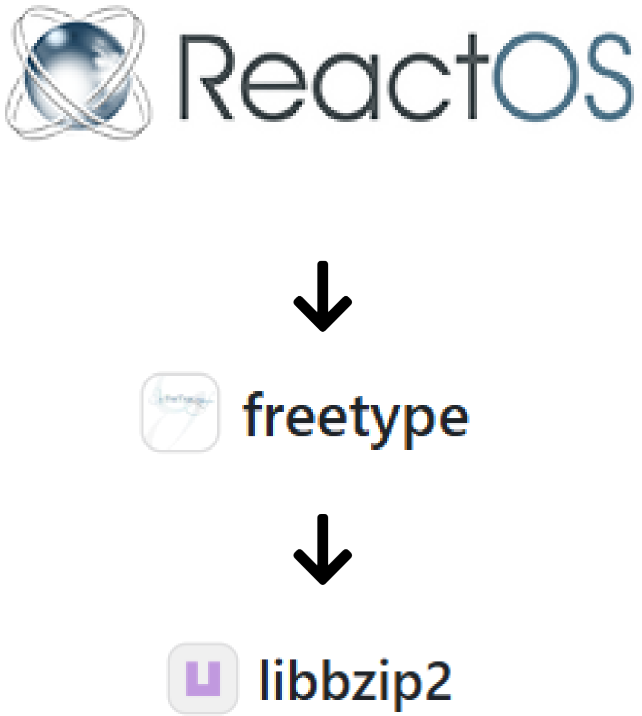
Step 3: Redundant Function Elimination

- As TPLs may nest other TPLs, redundant functions will exist in the database.

Overview of VULTURE

TPLFILTER Construction



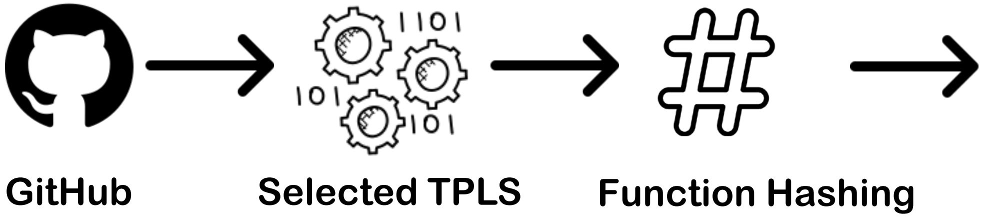


Step 3: Redundant Function Elimination

- As TPLs may nest other TPLs, redundant functions will exist in the database.

Overview of VULTURE

TPLFILTER Construction



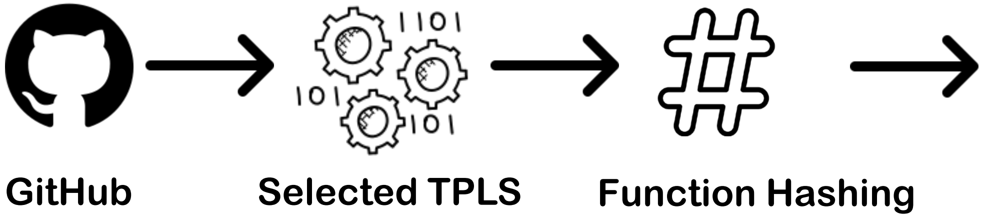
Step 3: Redundant Function Elimination

- As TPLs may nest other TPLs, redundant functions will exist in the database.

- Eliminate redundant functions according to

Overview of VULTURE

TPLFILTER Construction

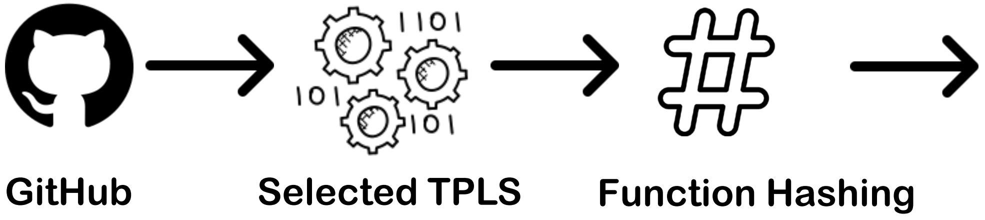


Step 3: Redundant Function Elimination

- As TPLs may nest other TPLs, redundant functions will exist in the database.
- Eliminate redundant functions according to
  - Function creation time

Overview of VULTURE

TPLFILTER Construction



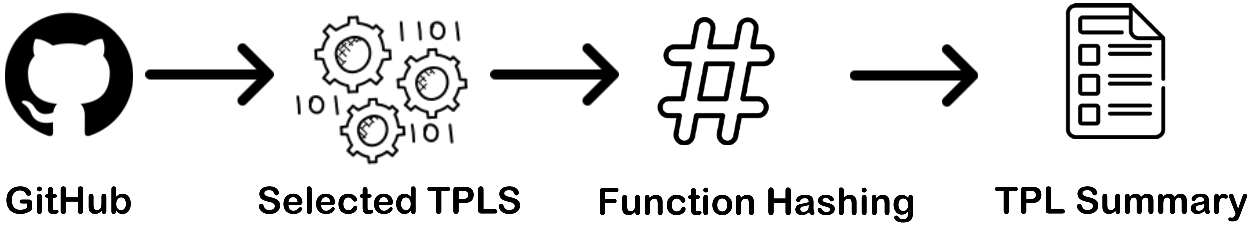
Step 3: Redundant Function Elimination

- As TPLs may nest other TPLs, redundant functions will exist in the database.

- Eliminate redundant functions according to
  - Function creation time
  - Similarity between function name & TPL name

Overview of VULTURE

TPLFILTER Construction

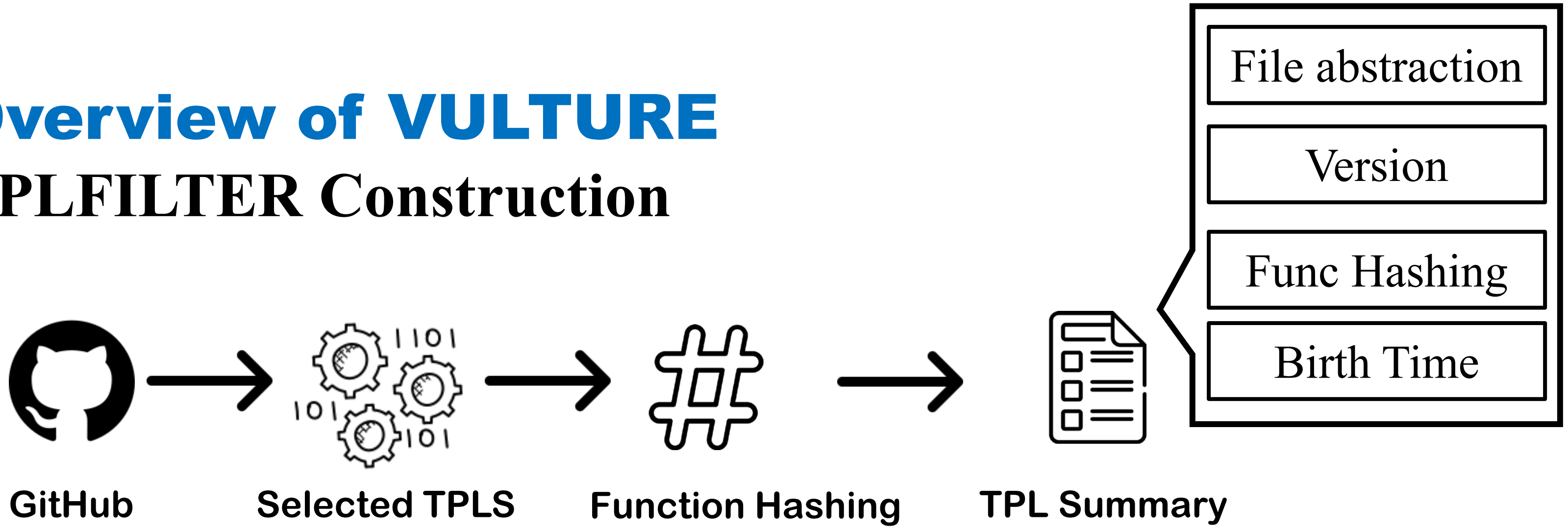


Step 3: Redundant Function Elimination

- As TPLs may nest other TPLs, redundant functions will exist in the database.

Overview of VULTURE

TPLFILTER Construction



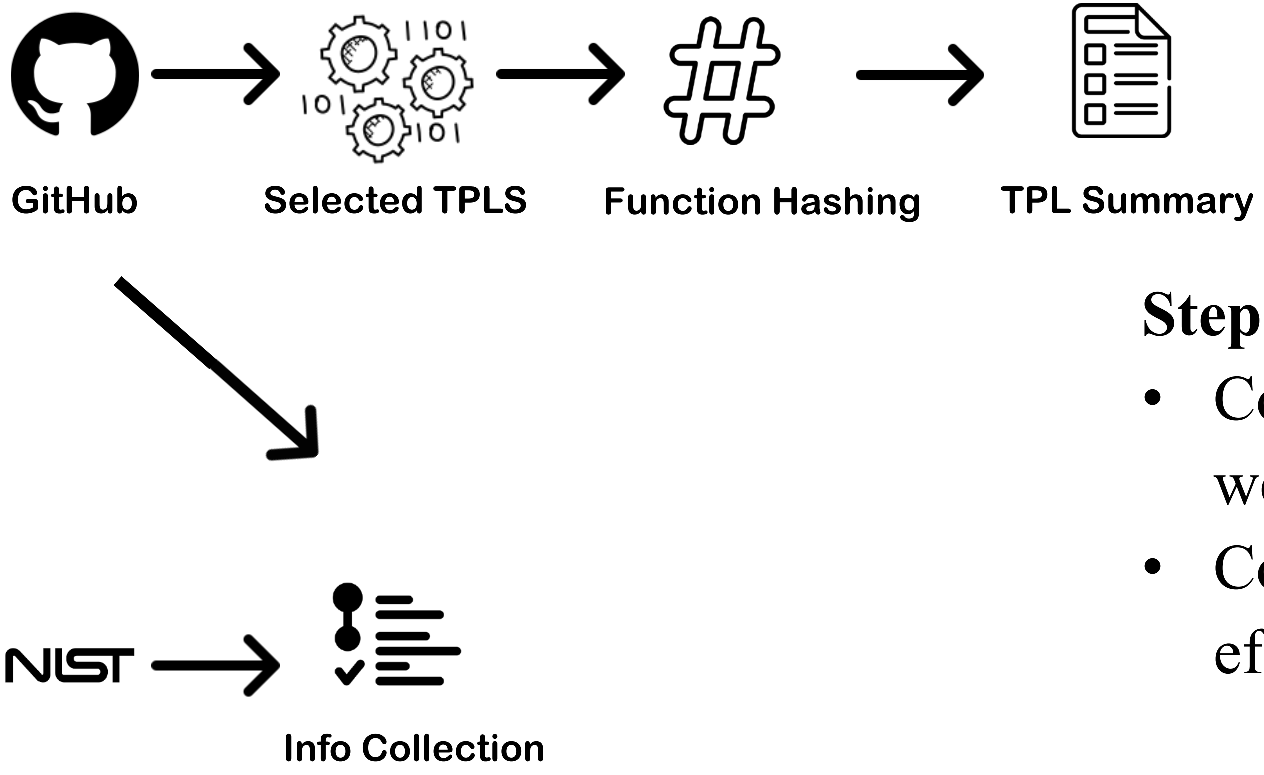
Step 3: Redundant Function Elimination

- As TPLs may nest other TPLs, redundant functions will exist in the database.

Overview of VULTURE

TPLFILTER Construction

Patch Collection

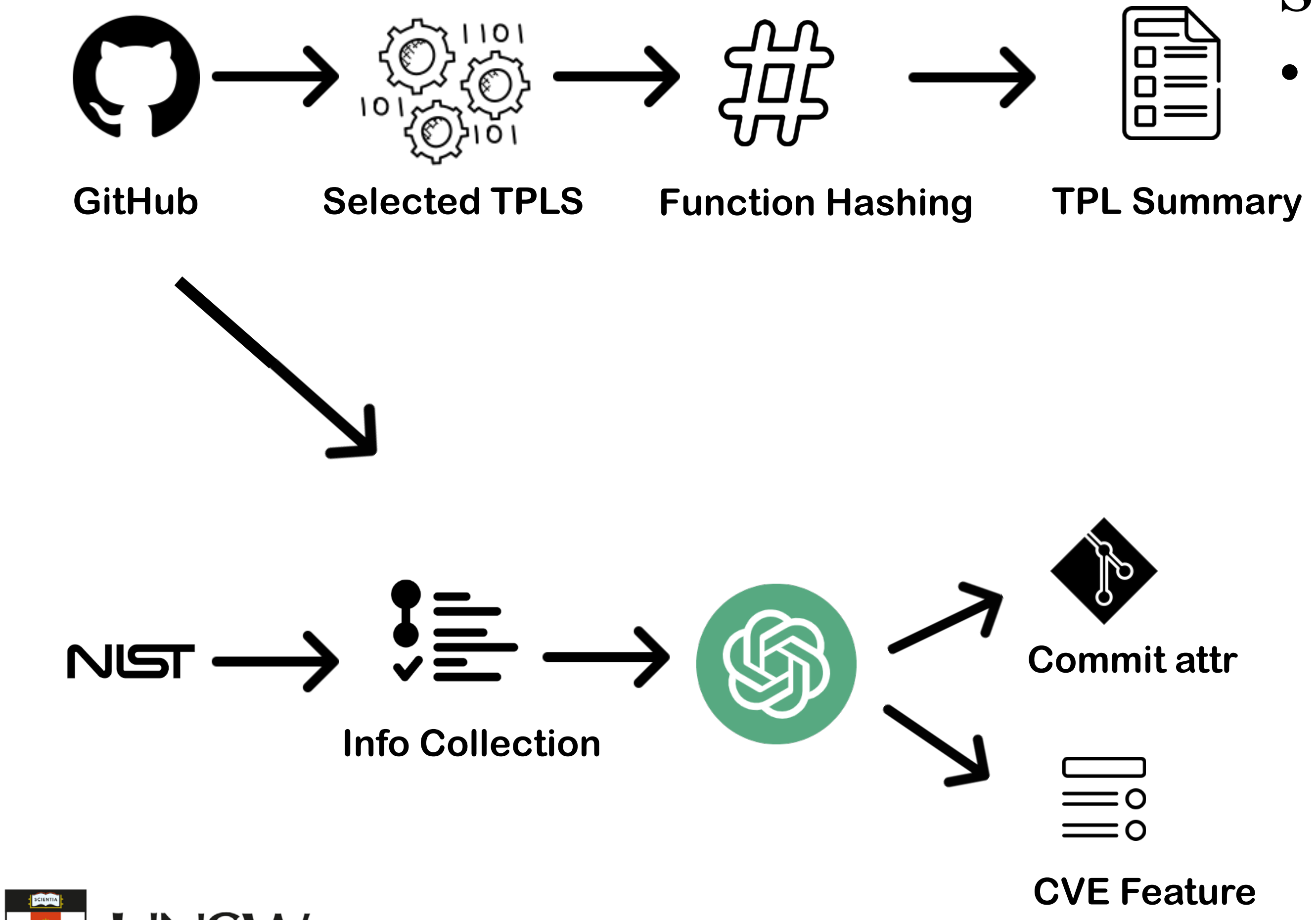


Step 1: Info Collection

- Collect CVE & description from NIST website
- Collect all of the commits of CVE effected repo from GitHub

Overview of VULTURE

TPLFILTER Construction



Step 2: LLM-based mapping

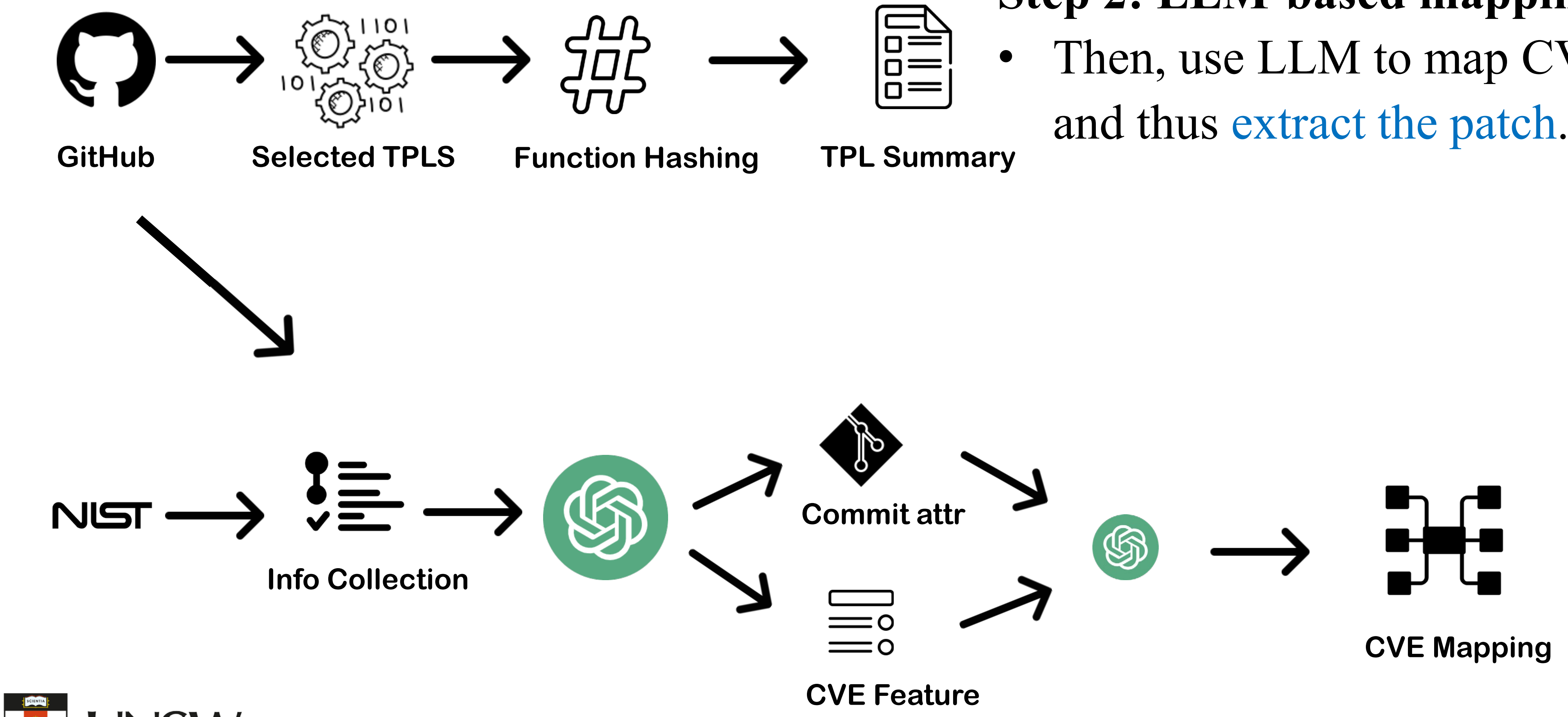
- Utilize LLM to identify CVE features and commit attributes (e.g., affected files, function names)

Overview of VULTURE

TPLFILTER Construction

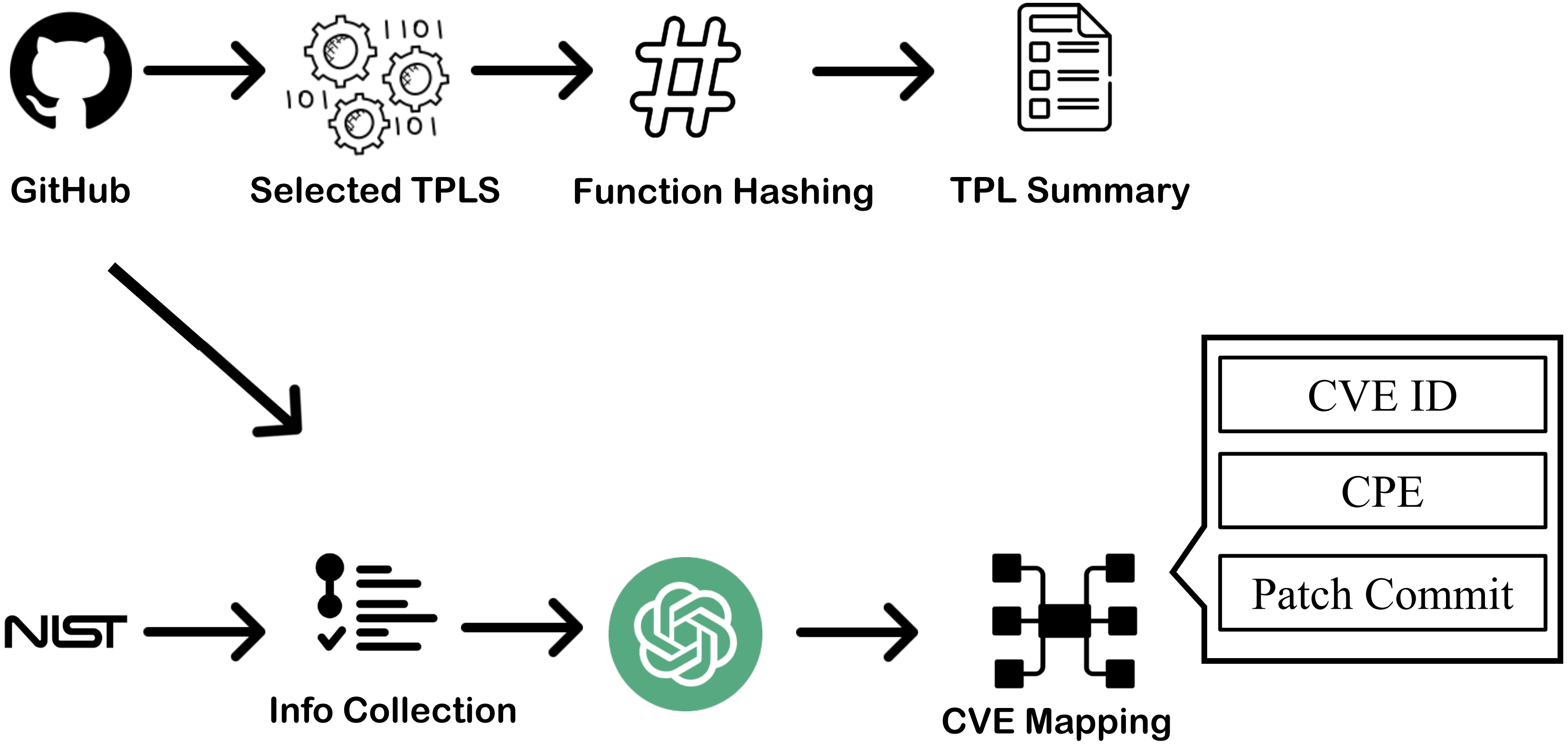
Step 2: LLM-based mapping

- Then, use LLM to map CVEs to commits and thus extract the patch.



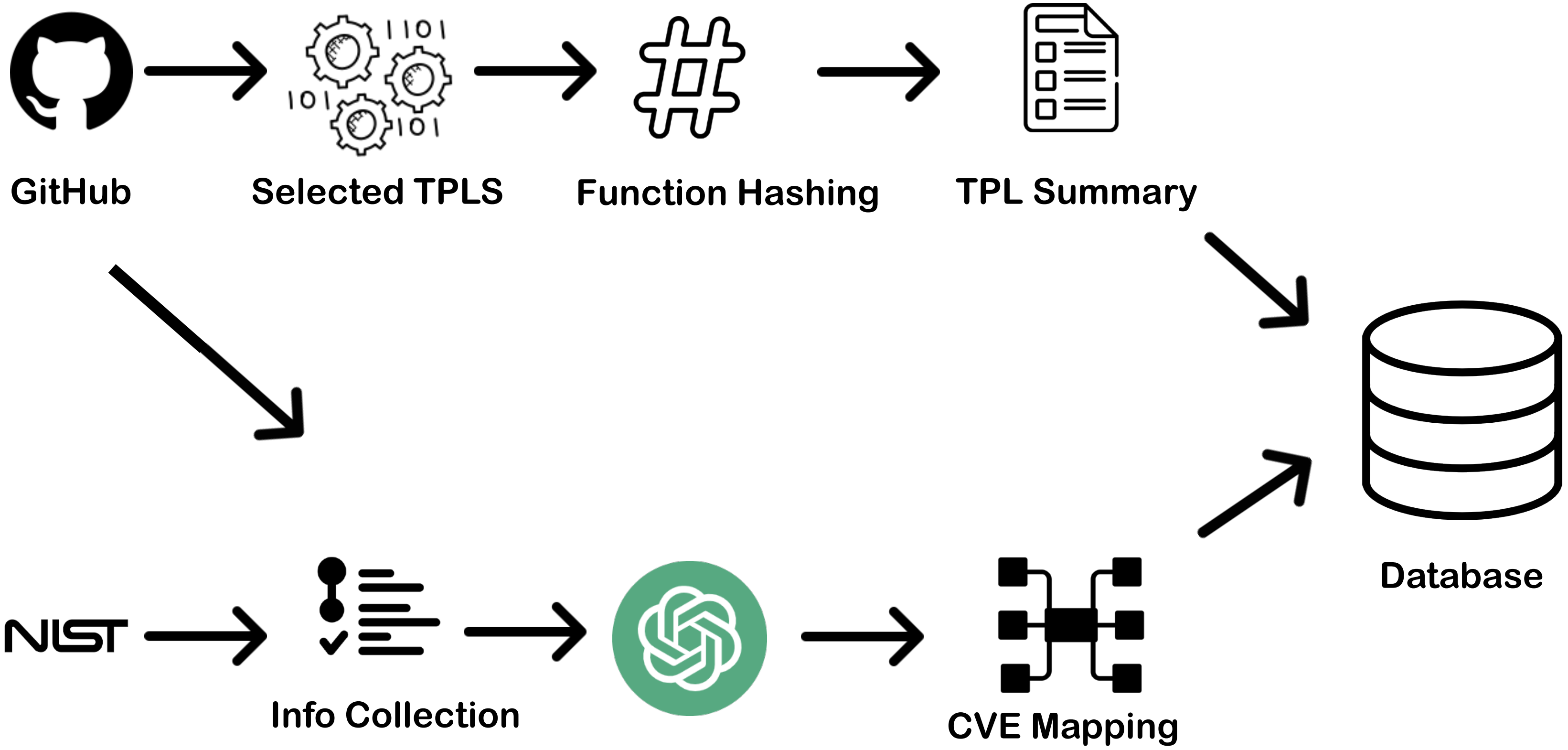
Overview of VULTURE

TPLFILTER Construction



Overview of VULTURE

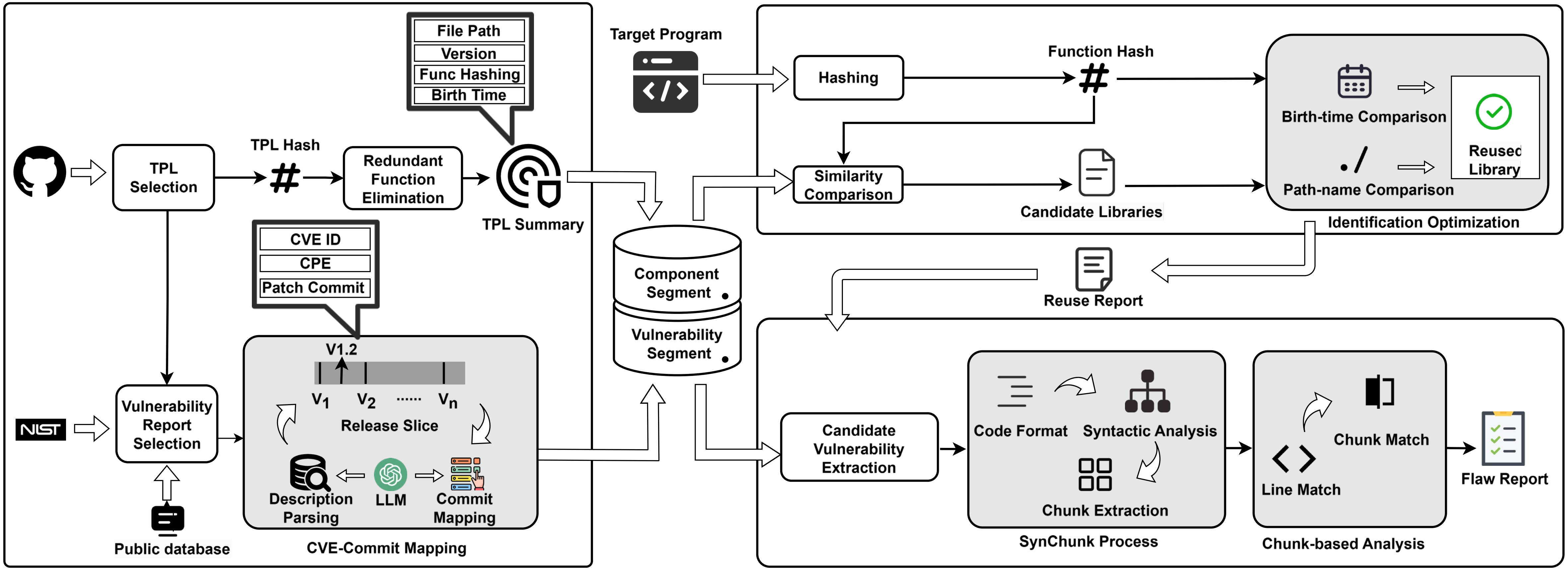
TPLFILTER Construction



RQ1 How to efficiently construct a database to support 1-day vulnerability detection

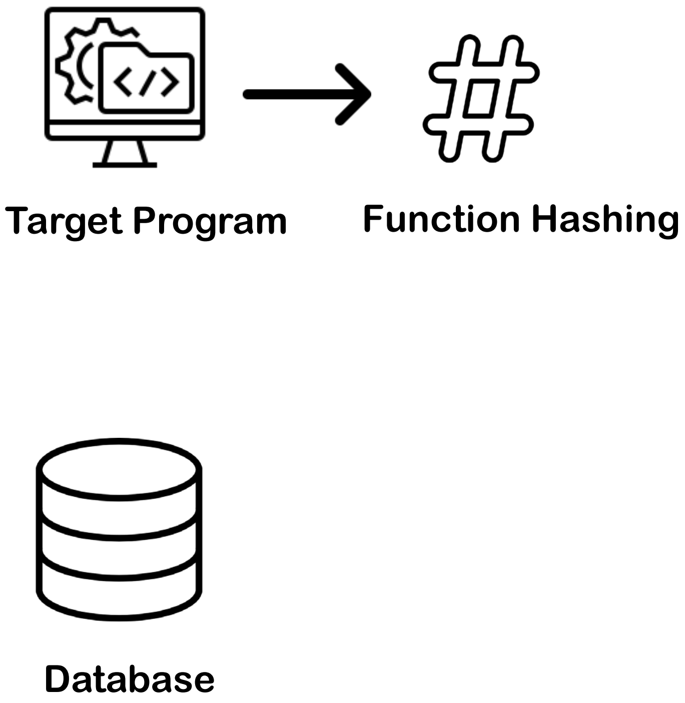
Overview of VULTURE

TPL Reuse Identification(RQ2)



Overview of VULTURE

TPL Reuse Identification

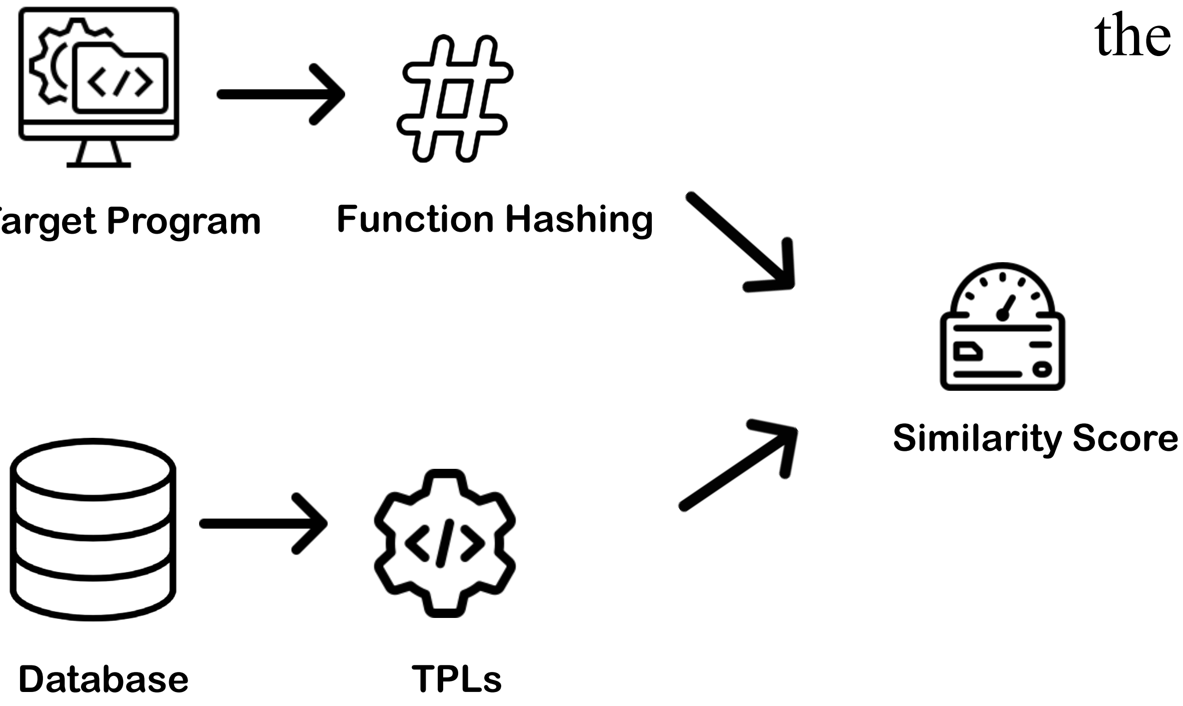


Step 1: Candidate library identification

- Generate a function hash for each function in the target program.

Overview of VULTURE

TPL Reuse Identification

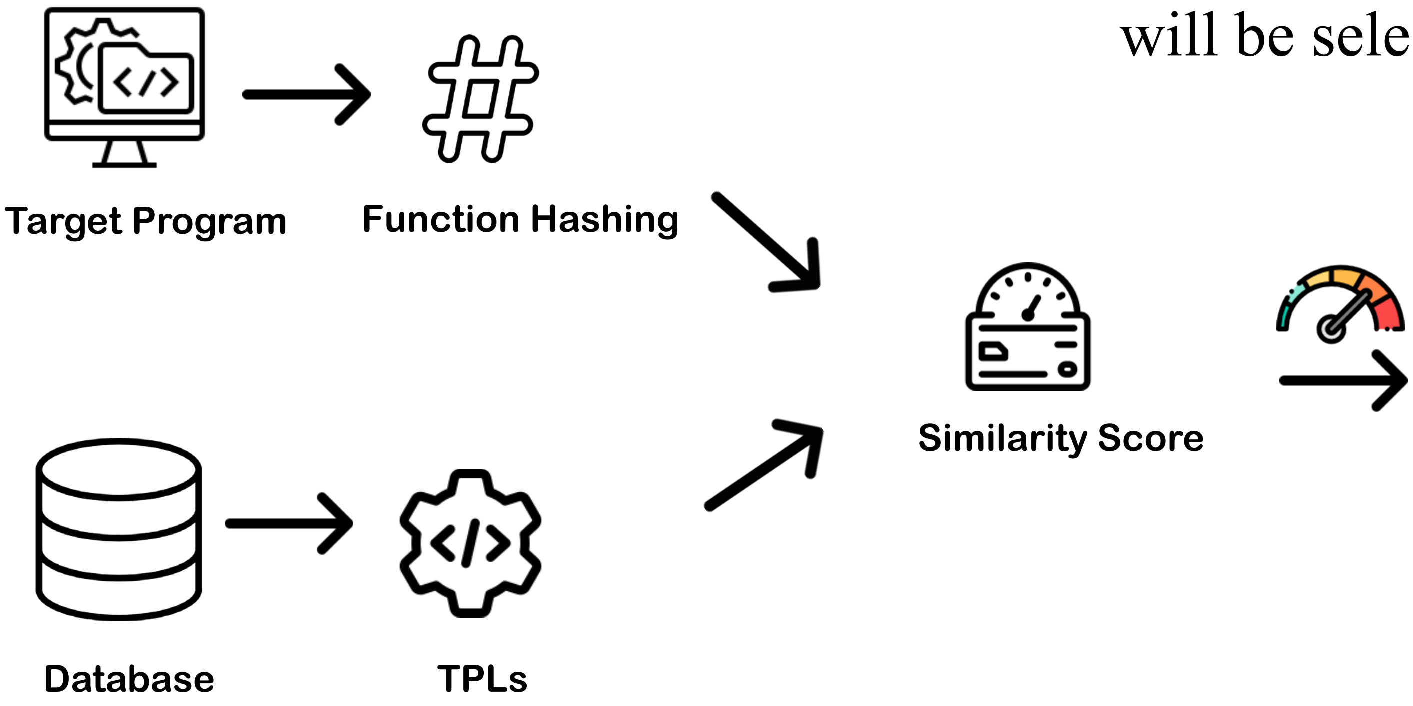


Step 1: Candidate library identification

- Compute similarity between each TPL and the target program.

Overview of VULTURE

TPL Reuse Identification



Step 1: Candidate library identification

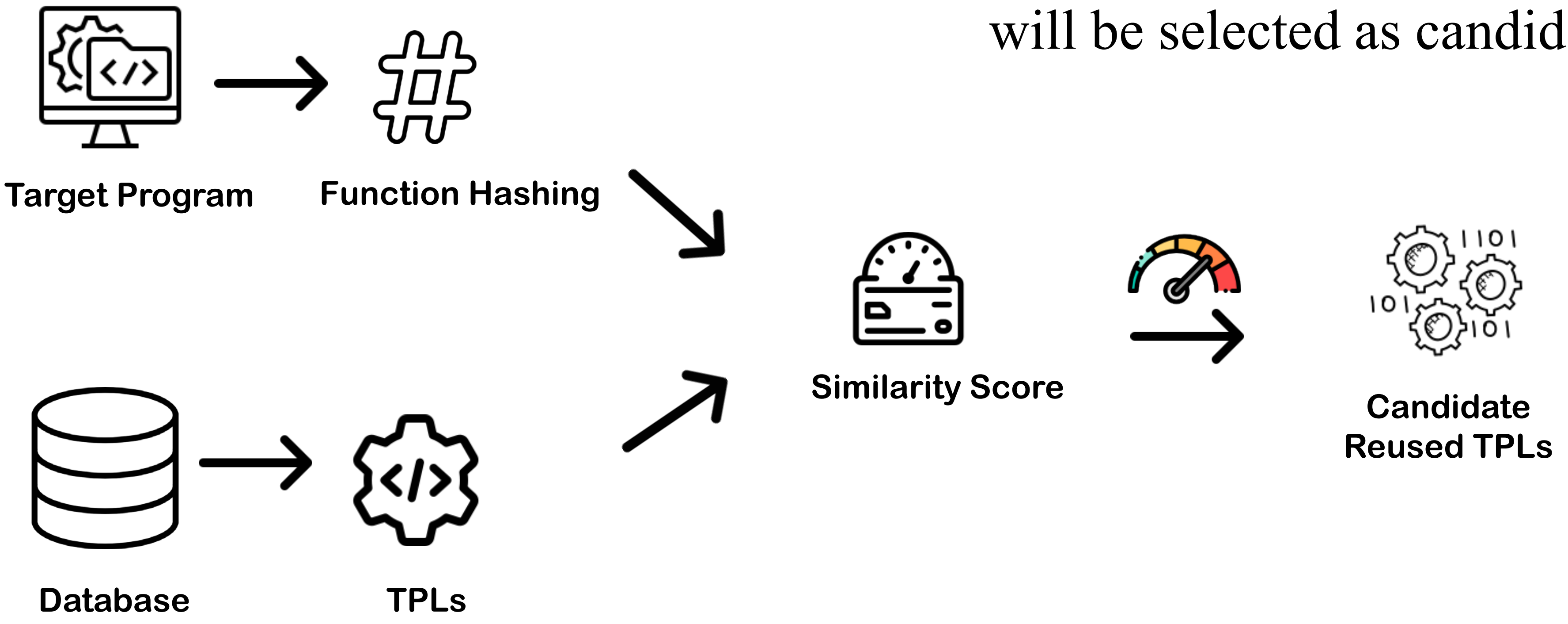
- TPLs exceeding the similarity threshold will be selected as candidate reused TPLs.

Overview of VULTURE

TPL Reuse Identification

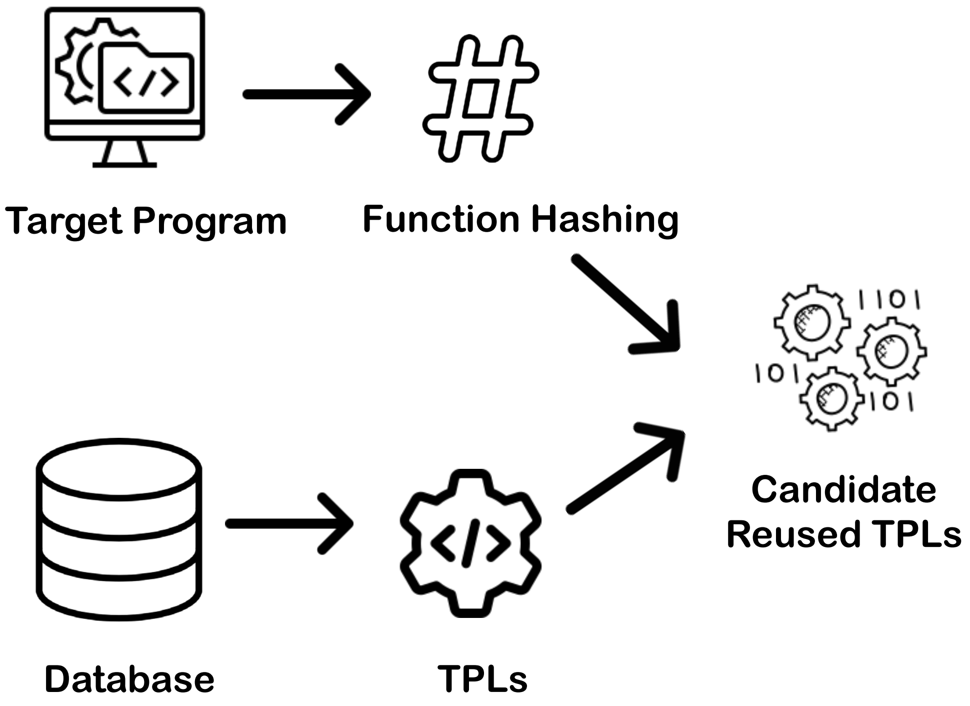
Step 1: Candidate library identification

- TPLs exceeding the similarity threshold will be selected as candidate reused TPLs.



Overview of VULTURE

TPL Reuse Identification

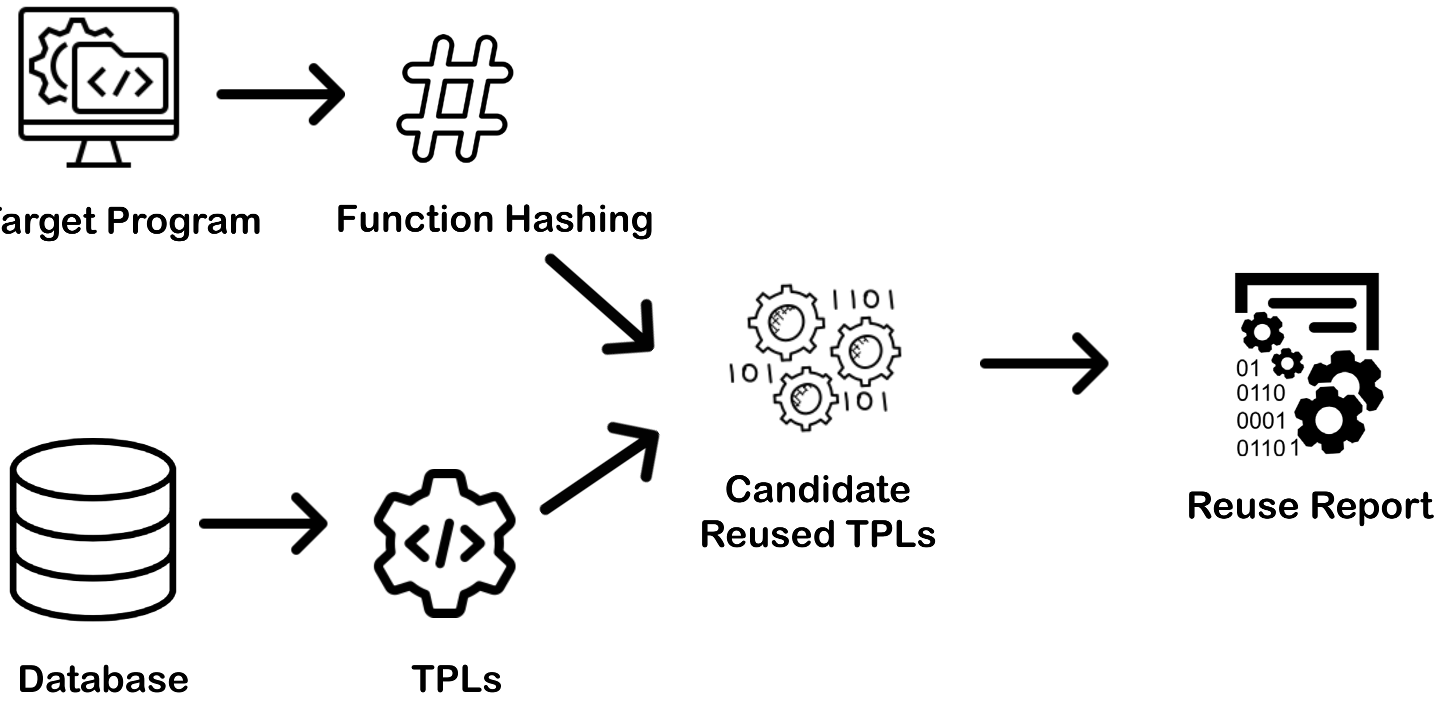


Step 2: Identification Optimization

- Compare each candidate TPL with its reuse path in the target program.
- Keep the earliest created TPL with the most similar name to the reuse path, removing others.

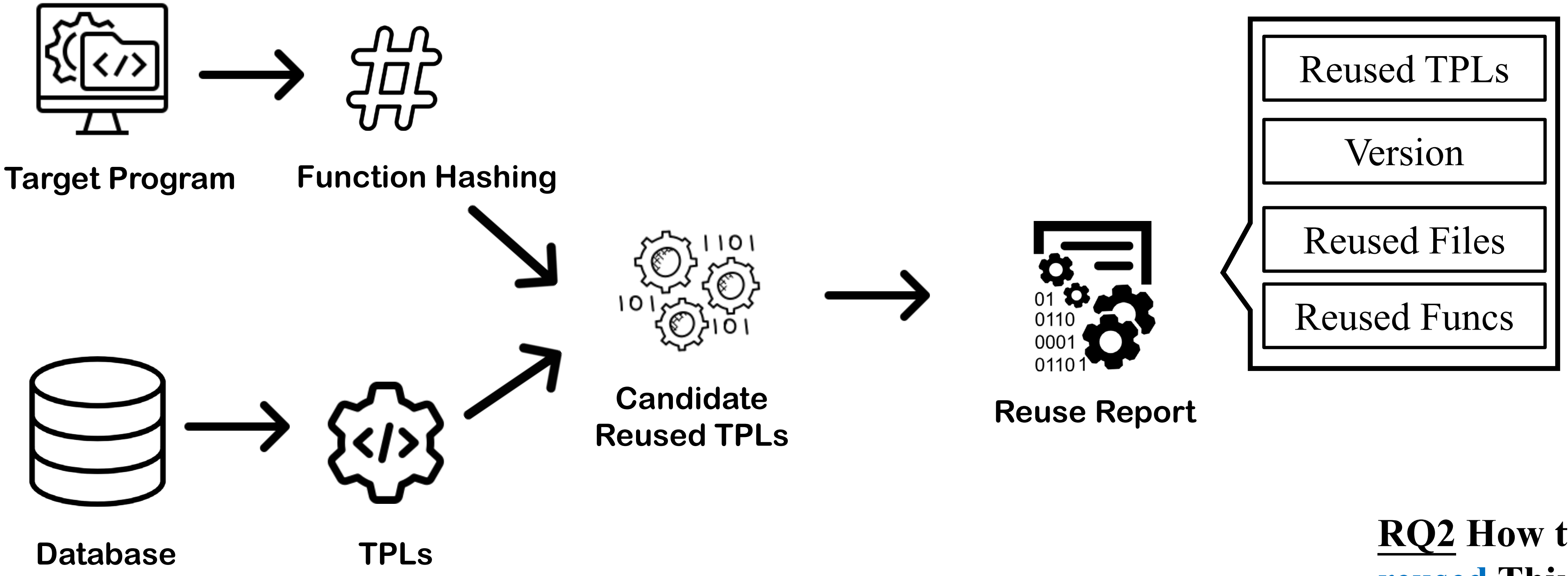
Overview of VULTURE

TPL Reuse Identification



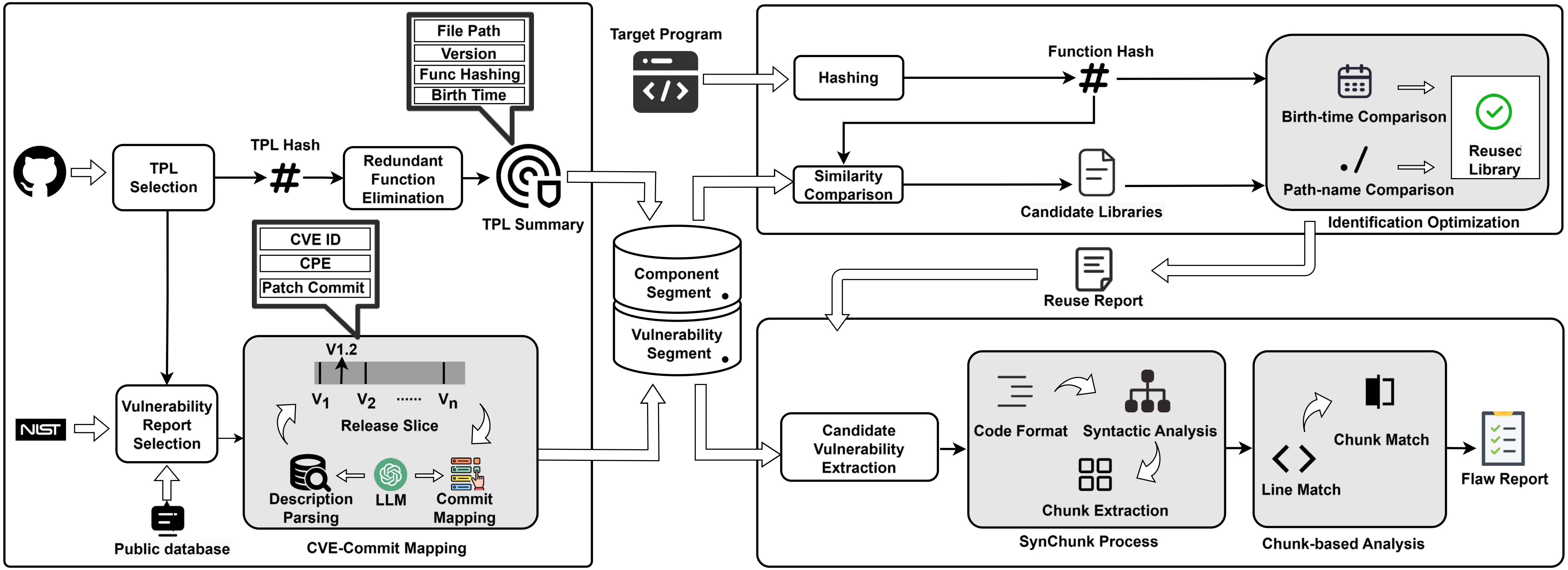
Overview of VULTURE

TPL Reuse Identification



RQ2 How to identify reused Third-party libraries in target program

Overview of VULTURE

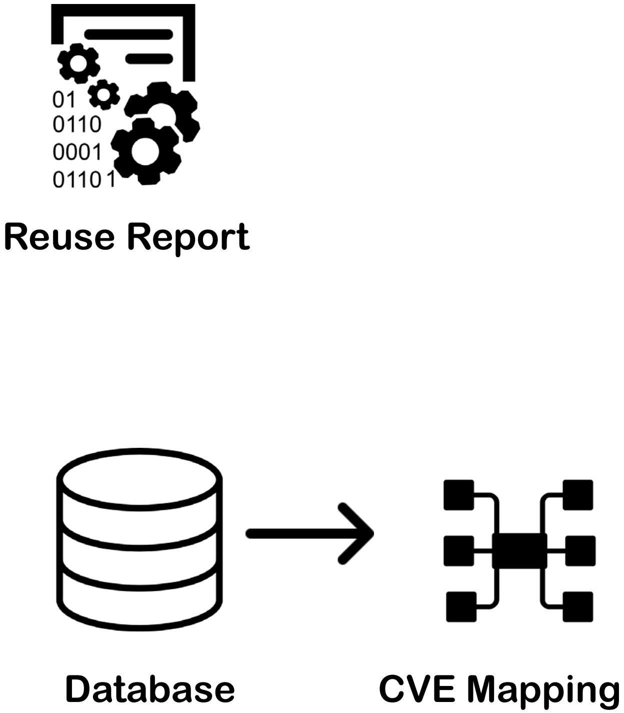


1-day Vulnerability Detection(RQ3)

UNSW CANBERRA

Overview of VULTURE

1-day Vulnerability Detection

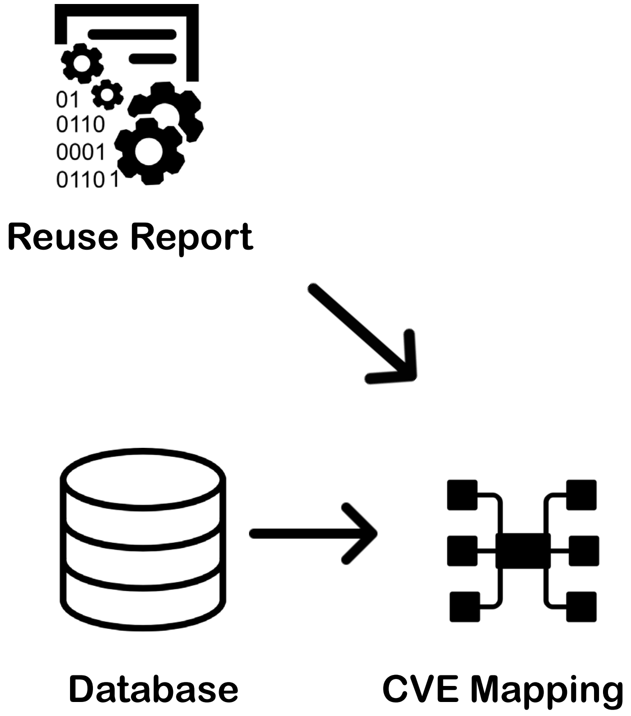


Step 1: Candidate library identification

- According to the report, identify candidate 1-day vulnerabilities based on reused TPLs and their versions.

Overview of VULTURE

1-day Vulnerability Detection

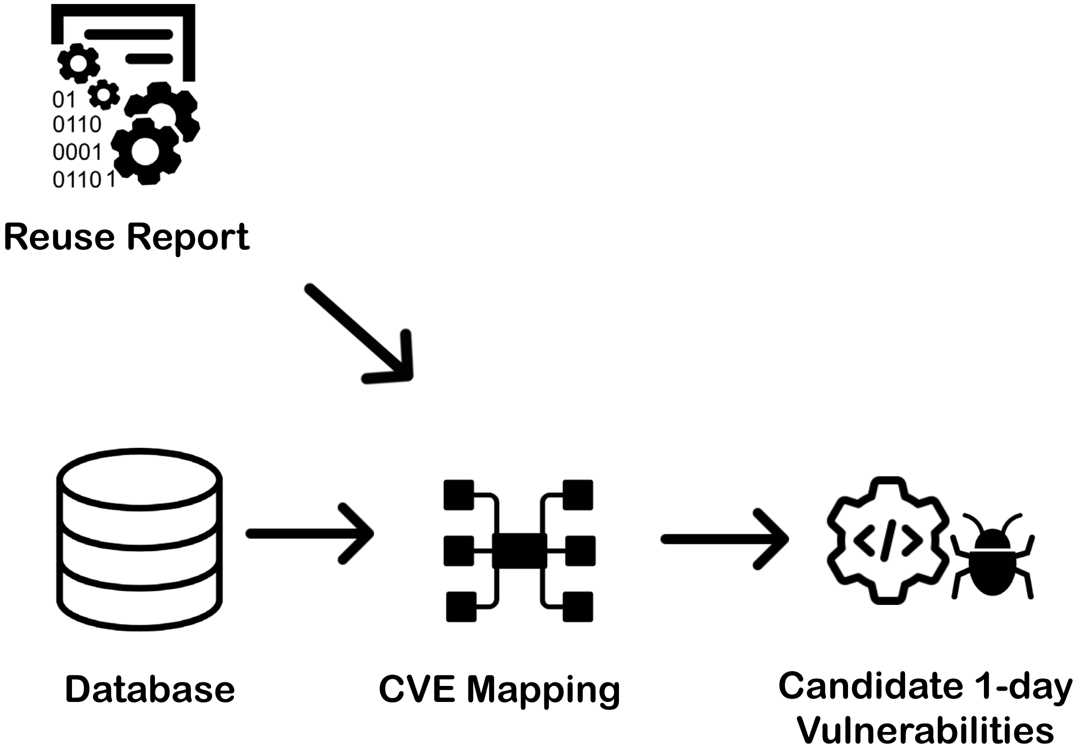


Step 1: Candidate library identification

- According to the report, identify candidate 1-day vulnerabilities based on reused TPLs and their versions.

Overview of VULTURE

1-day Vulnerability Detection

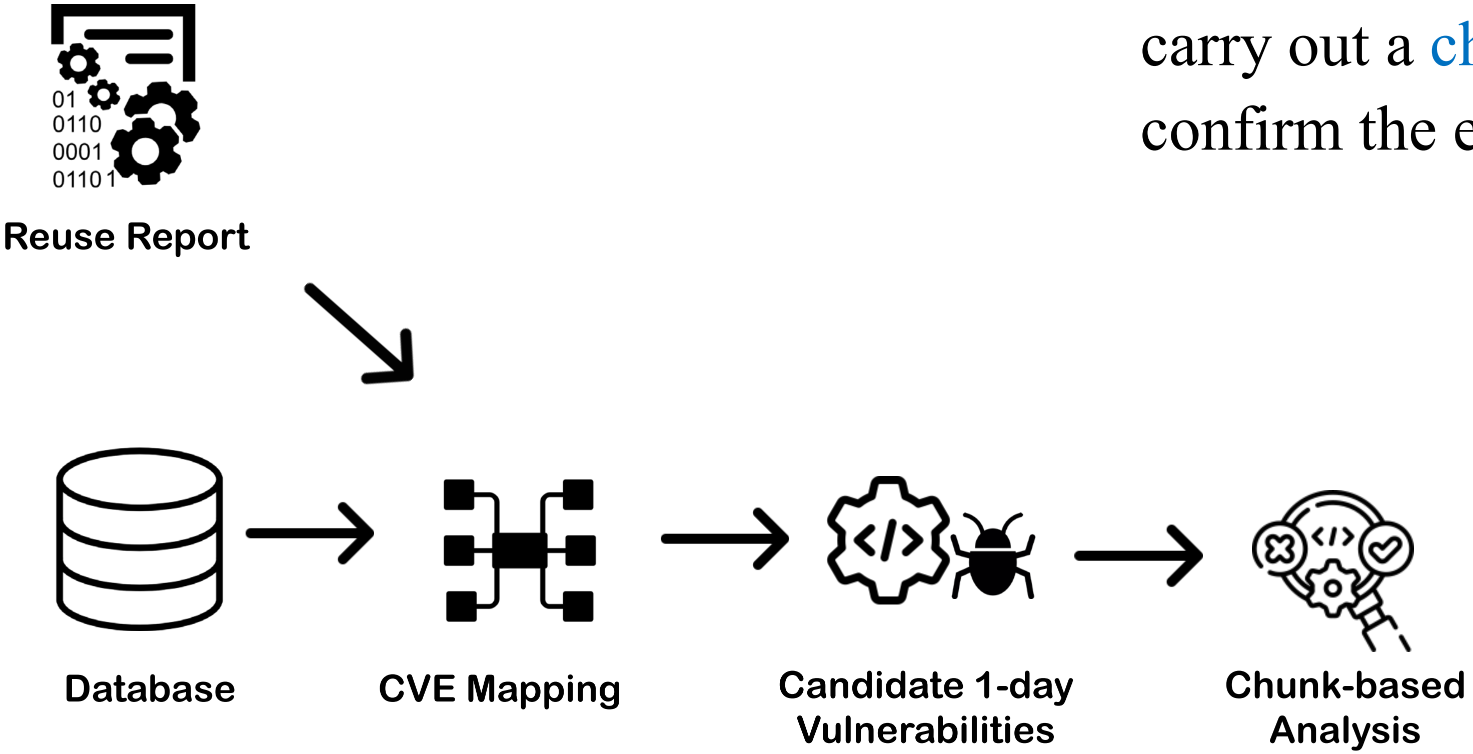


Step 1: Candidate library identification

- According to the report, identify candidate 1-day vulnerabilities based on reused TPLs and their versions.

Overview of VULTURE

1-day Vulnerability Detection



Reuse Report

Database

CVE Mapping

Candidate 1-day Vulnerabilities

Chunk-based Analysis

Step 2: Candidate library identification

- For each candidate 1-day vulnerability, carry out a chunk-based analysis to confirm the existence of vulnerability.

1-day Vulnerability Detection

Chunk-based Analysis

- Designed to handle custom modifications in TPL reuses.
- Extracts semantic information through static patch analysis to identify meaningful changes.
- Groups modified lines into chunks based on control and variable dependencies.
- Compares target, vulnerable, and patched code chunks to verify security patches.

1-day Vulnerability Detection

Chunk-based Analysis

- Designed to handle custom modifications in TPL reuses.
- Extracts semantic information through static patch analysis to identify meaningful changes.
- Groups modified lines into chunks based on control and variable dependencies.
- Compares target, vulnerable, and patched code chunks to verify security patches.

1-day Vulnerability Detection

Chunk-based Analysis

- Designed to handle custom modifications in TPL reuses.
- Extracts semantic information through static patch analysis to identify meaningful changes.
- Groups modified lines into chunks based on control and variable dependencies.
- Compares target, vulnerable, and patched code chunks to verify security patches.

Overview of VULTURE Chunk-based Analysis

1-day Vulnerability Detection

Chunk-based Analysis

- Designed to handle custom modifications in TPL reuses.
- Extracts semantic information through static patch analysis to identify meaningful changes.
- Groups modified lines into chunks based on control and variable dependencies.
- Compares target, vulnerable, and patched code chunks to verify security patches.

Overview of VULTURE

Chunk-based Analysis

1-day Vulnerability Detection

```
METHODDEF(JDIMENSION) get_8bit_row(...){

  +   int cmaplen = source->cmap_length;
    ...
  + if (t >= cmaplen)
  +    return 1;
    output++ = colormap[0][t];
    ...
  + if (getUsed(bitClrUsed))
  +    return 1;
    ...
}
```

Patch for CVE-2018-14498

Overview of VULTURE Chunk-based Analysis

1-day Vulnerability Detection

```
METHODDEF(JDIMENSION) get_8bit_row(...){

  +   int cmaplen = source->cmap_length;
    ...
  + if (t >= cmaplen)
  +    return 1;
    output++ = colormap[0][t];
    ...
  + if (getUsed(bitClrUsed))
  +    return 1;
    ...
}
```

Patch for CVE-2018-14498

```
METHODDEF(JDIMENSION) get_8bit_row(...){

  +   int cmaplen;
    ...
  +  cmaplen = source->cmap_length;
    ...
  + if (t >= cmaplen)
  +   return 1;
    output++ = colormap[0][t];
    ...
  + if (getUsed(bitClrUsed))
  +     return 1;
    ...
}
```

Patch applied by ReactOS

Overview of VULTURE Chunk-based Analysis

1-day Vulnerability Detection

```
METHODDEF(JDIMENSION) get_8bit_row(...){
  +   int cmaplen = source->cmap_length;
    ...
  + if (t >= cmaplen)
  +    return 1;
    output++ = colormap[0][t];
    ...
  + if (getUsed(bitClrUsed))
  +    return 1;
    ...
}
```

Patch for CVE-2018-14498

```
METHODDEF(JDIMENSION) get_8bit_row(...){

  +   int cmaplen;
    ...
  +  cmaplen = source->cmap_length;
    ...
  + if (t >= cmaplen)
  +   return 1;
    output++ = colormap[0][t];
    ...
  + if (getUsed(bitClrUsed))
  +     return 1;
    ...
}
```

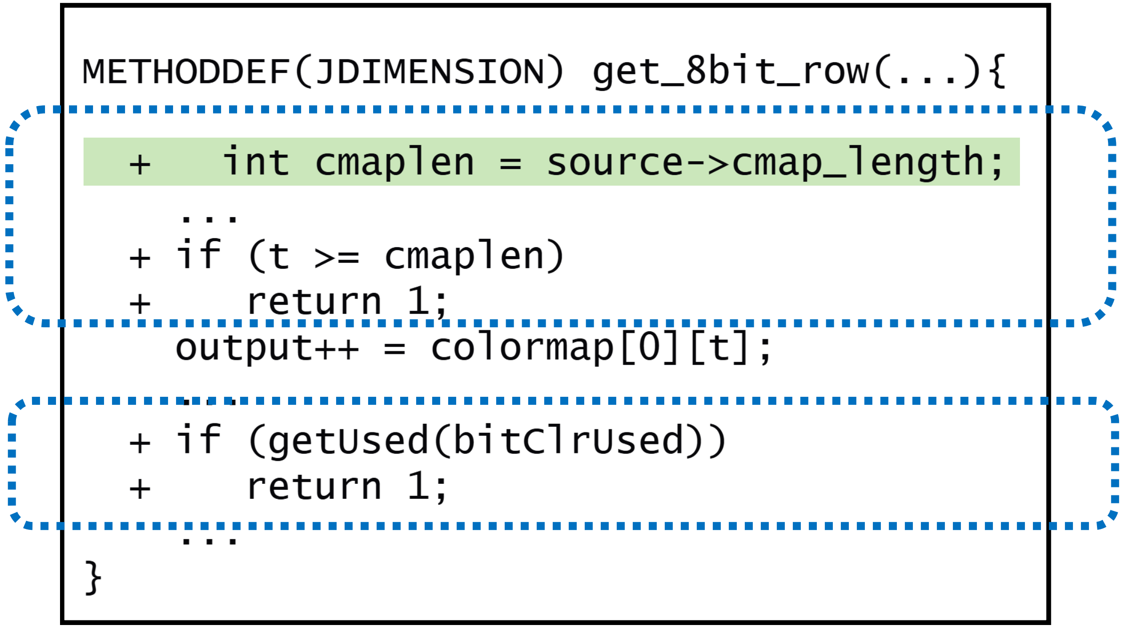
Patch applied by ReactOS

UNSW CANBERRA

Overview of VULTURE

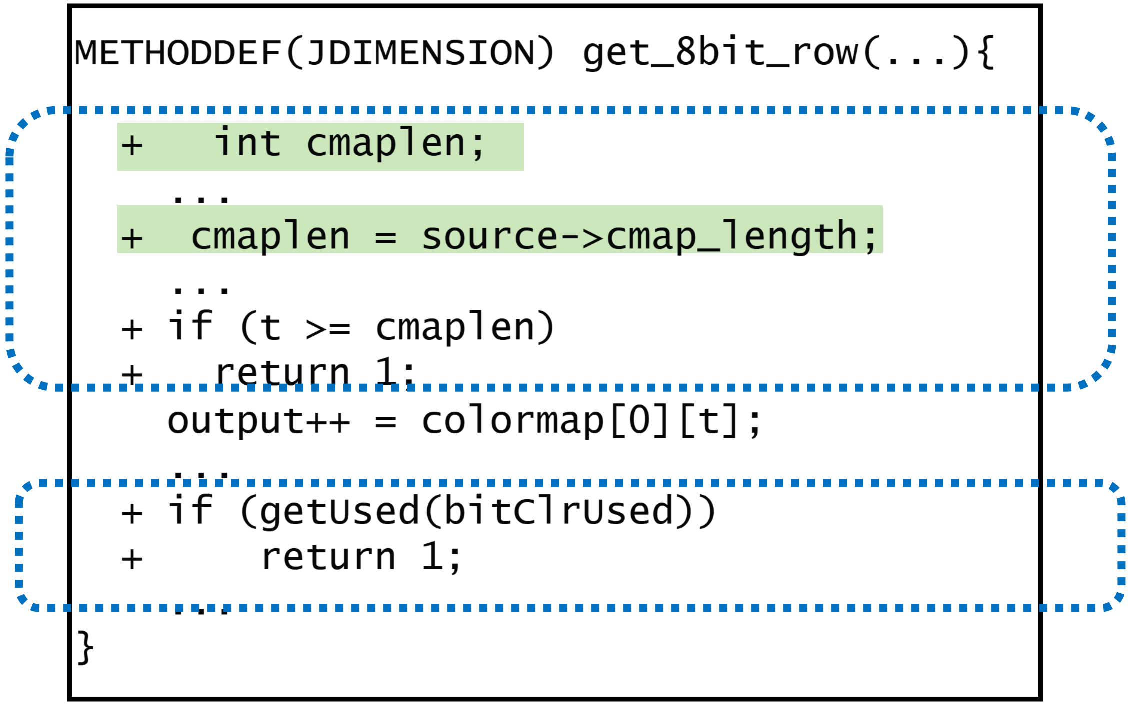
Chunk-based Analysis

1-day Vulnerability Detection



```
METHODDEF(JDIMENSION) get_8bit_row(...){
  +   int cmaplen = source->cmap_length;
      ...
  + if (t >= cmaplen)
  +     return 1;
    output++ = colormap[0][t];
      ...
  + if (getUsed(bitClrUsed))
  +     return 1;
      ...
}
```

Patch for CVE-2018-14498



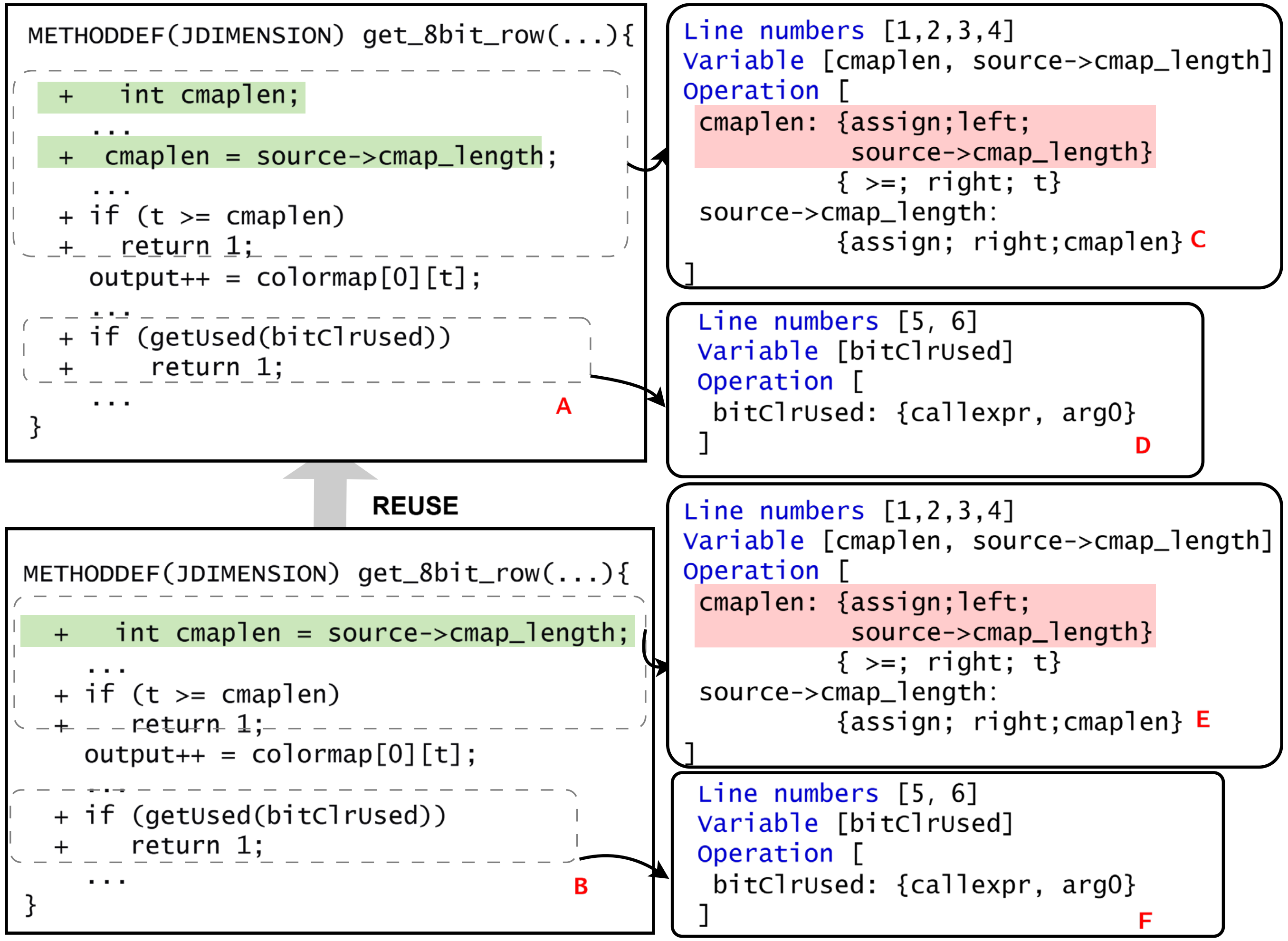
```
METHODDEF(JDIMENSION) get_8bit_row(...){
  +   int cmaplen;
      ...
  +  cmaplen = source->cmap_length;
      ...
  + if (t >= cmaplen)
  +    return 1;
    output++ = colormap[0][t];
      ...
  + if (getUsed(bitClrUsed))
  +     return 1;
      ...
}
```

Patch applied by ReactOS

Overview of VULTURE Chunk-based Analysis

1-day Vulnerability Detection

Extract variables, operations, and how operations are applied to variables in each chunk.



```
METHODDEF(JDIMENSION) get_8bit_row(...){
 +   int cmaplen;
   ...
 +  cmaplen = source->cmap_length;
   ...
 + if (t >= cmaplen)
 +     return 1;
   output++ = colormap[0][t];
   ...
 + if (getUsed(bitClrUsed))
 +     return 1;
   ...
}
```

```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
           source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

```
Line numbers [5, 6]
Variable [bitClrUsed]
Operation [
 bitClrUsed: {callexpr, arg0}
]
```

```
METHODDEF(JDIMENSION) get_8bit_row(...){
 +   int cmaplen = source->cmap_length;
   ...
 + if (t >= cmaplen)
 +     return 1;
   output++ = colormap[0][t];
   ...
 + if (getUsed(bitClrUsed))
 +     return 1;
   ...
}
```

```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
           source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

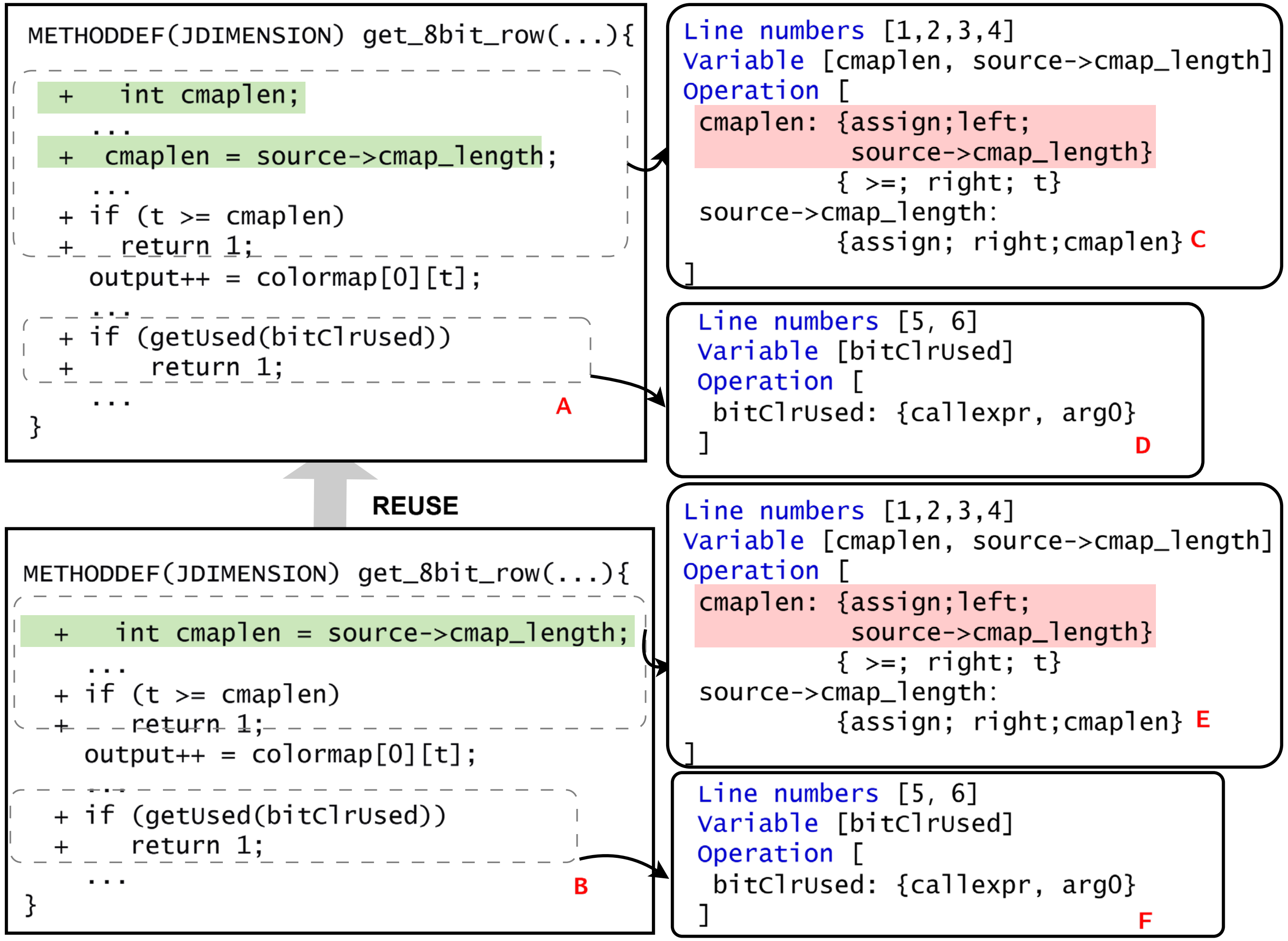
```
Line numbers [5, 6]
Variable [bitClrUsed]
Operation [
 bitClrUsed: {callexpr, arg0}
]
```

Overview of VULTURE

Chunk-based Analysis

1-day Vulnerability Detection

Extract variables, operations, and how operations are applied to variables in each chunk.



```
METHODDEF(JDIMENSION) get_8bit_row(...){
  +   int cmaplen;
    ...
  +  cmaplen = source->cmap_length;
    ...
  + if (t >= cmaplen)
  +   return 1;
    output++ = colormap[0][t];
    ...
  + if (getUsed(bitClrUsed))
  +     return 1;
    ...
}
```

```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
           source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

```
Line numbers [5, 6]
Variable [bitClrUsed]
Operation [
 bitClrUsed: {callexpr, arg0}
]
```

```
METHODDEF(JDIMENSION) get_8bit_row(...){
  +   int cmaplen = source->cmap_length;
    ...
  + if (t >= cmaplen)
  +   return 1;
    output++ = colormap[0][t];
    ...
  + if (getUsed(bitClrUsed))
  +   return 1;
    ...
}
```

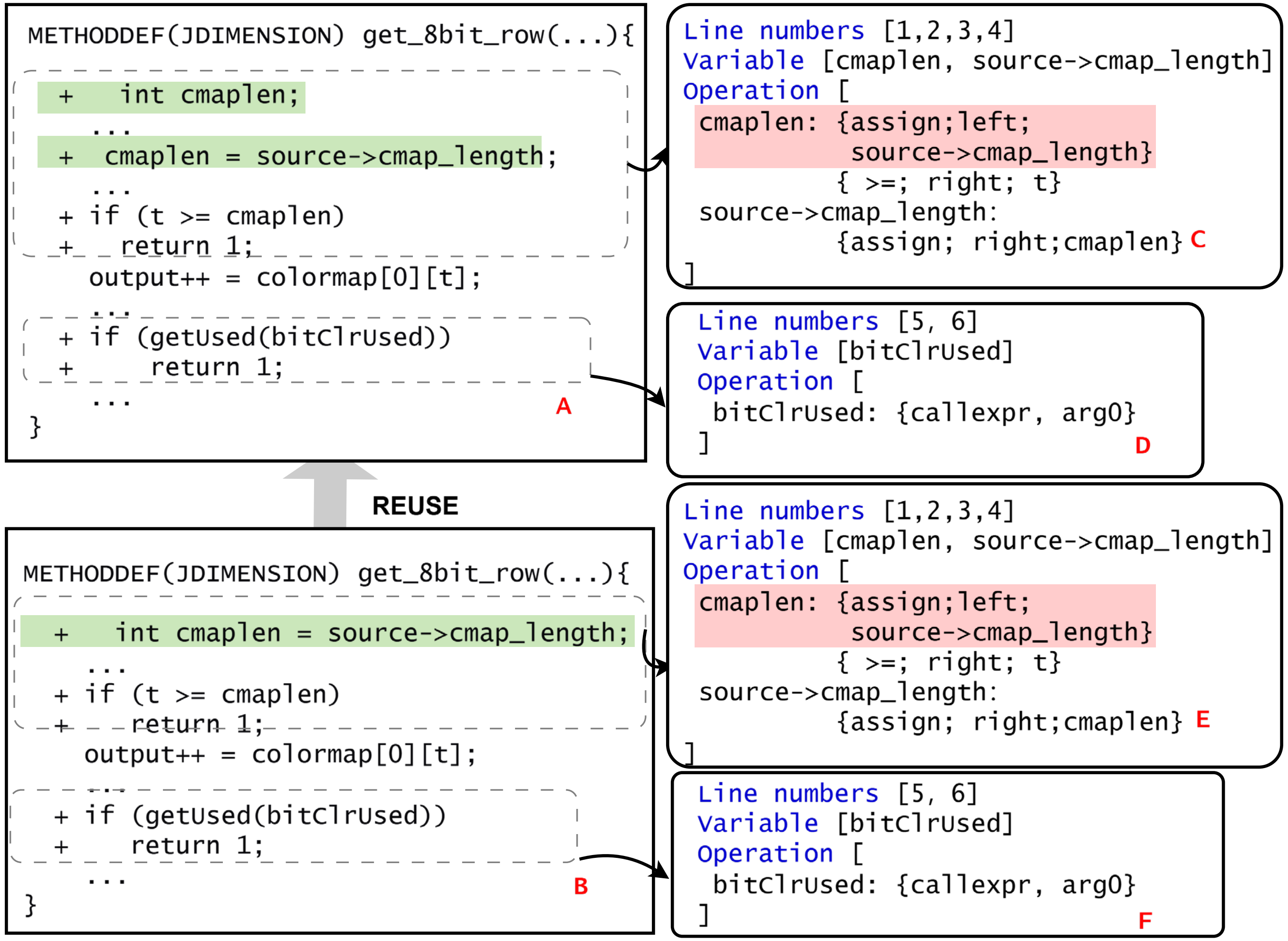
```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
           source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

```
Line numbers [5, 6]
Variable [bitClrUsed]
Operation [
 bitClrUsed: {callexpr, arg0}
]
```

Overview of VULTURE Chunk-based Analysis

1-day Vulnerability Detection

Extract variables, operations, and how operations are applied to variables in each chunk.



```
METHODDEF(JDIMENSION) get_8bit_row(...){
  +    int cmaplen;
       ...
  +   cmaplen = source->cmap_length;
       ...
  + if (t >= cmaplen)
  +     return 1;
    output++ = colormap[0][t];
       ...
  + if (getUsed(bitClrUsed))
  +      return 1;
       ...
}
```

```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
           source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

```
Line numbers [5, 6]
Variable [bitClrUsed]
Operation [
 bitClrUsed: {callexpr, arg0}
]
```

```
METHODDEF(JDIMENSION) get_8bit_row(...){
  +    int cmaplen = source->cmap_length;
       ...
  + if (t >= cmaplen)
  +     return 1;
    output++ = colormap[0][t];
       ...
  + if (getUsed(bitClrUsed))
  +     return 1;
       ...
}
```

```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
           source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

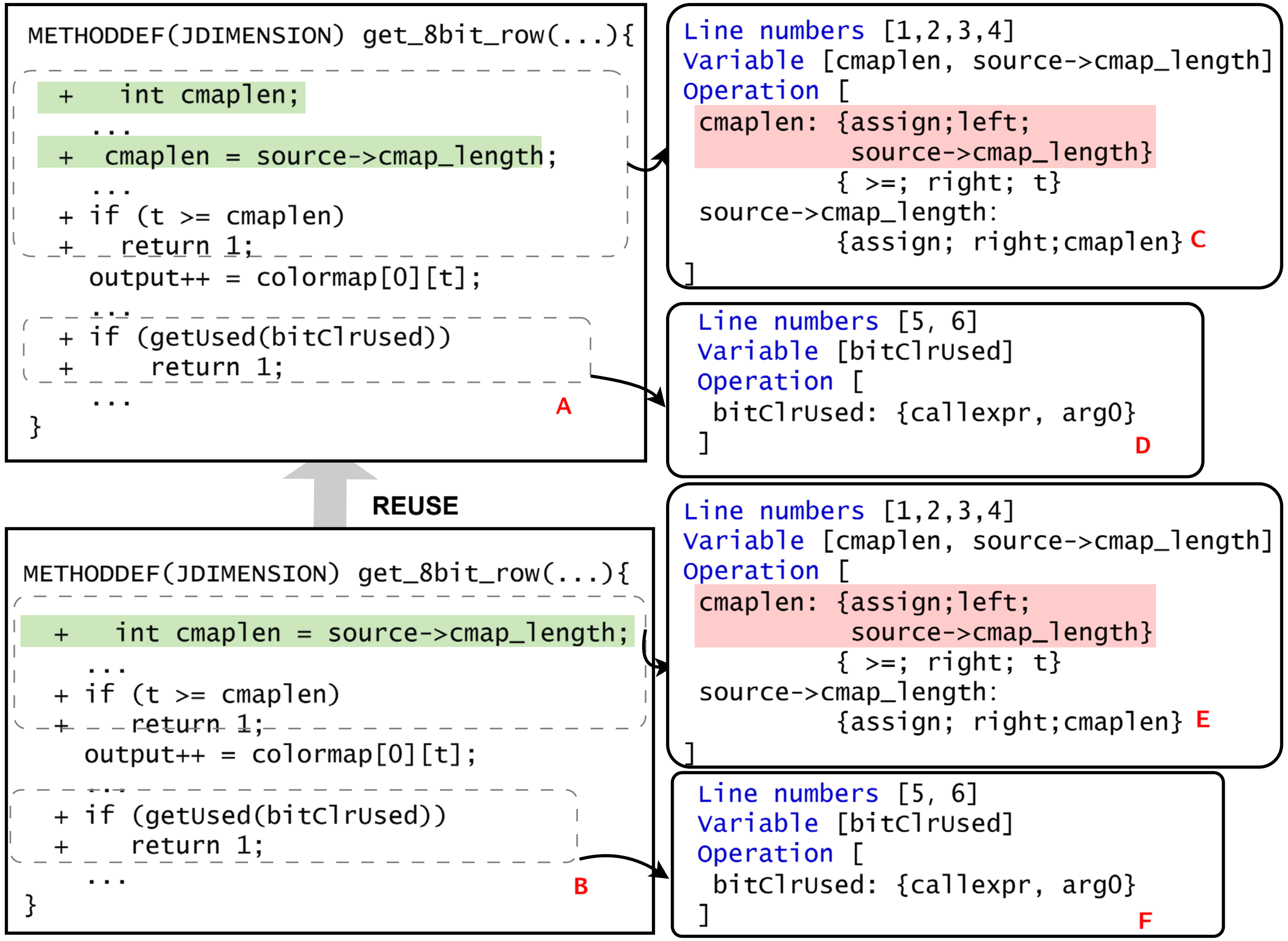
```
Line numbers [5, 6]
Variable [bitClrUsed]
Operation [
 bitClrUsed: {callexpr, arg0}
]
```

Overview of VULTURE

Chunk-based Analysis

1-day Vulnerability Detection

Extract variables, operations, and how operations are applied to variables in each chunk.



```
METHODDEF(JDIMENSION) get_8bit_row(...){
 +   int cmaplen;
     ...
 +  cmaplen = source->cmap_length;
     ...
  + if (t >= cmaplen)
  +     return 1;
    output++ = colormap[0][t];
    ...
  + if (getUsed(bitClrUsed))
  +     return 1;
    ...
}
```

```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
             source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

```
Line numbers [5, 6]
Variable [bitClrUsed]
Operation [
 bitClrUsed: {callexpr, arg0}
]
```

```
METHODDEF(JDIMENSION) get_8bit_row(...){
 +   int cmaplen = source->cmap_length;
    ...
  + if (t >= cmaplen)
  +     return 1;
    output++ = colormap[0][t];
    ...
  + if (getUsed(bitClrUsed))
  +     return 1;
    ...
}
```

```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
             source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

```
Line numbers [5, 6]
Variable [bitClrUsed]
Operation [
 bitClrUsed: {callexpr, arg0}
]
```

Overview of VULTURE Chunk-based Analysis

1-day Vulnerability Detection

```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
           source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

Patch for CVE-2018-14498

```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
           source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

Patch applied by ReactOS

Overview of VULTURE Chunk-based Analysis

1-day Vulnerability Detection

```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
           source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

Patch for CVE-2018-14498

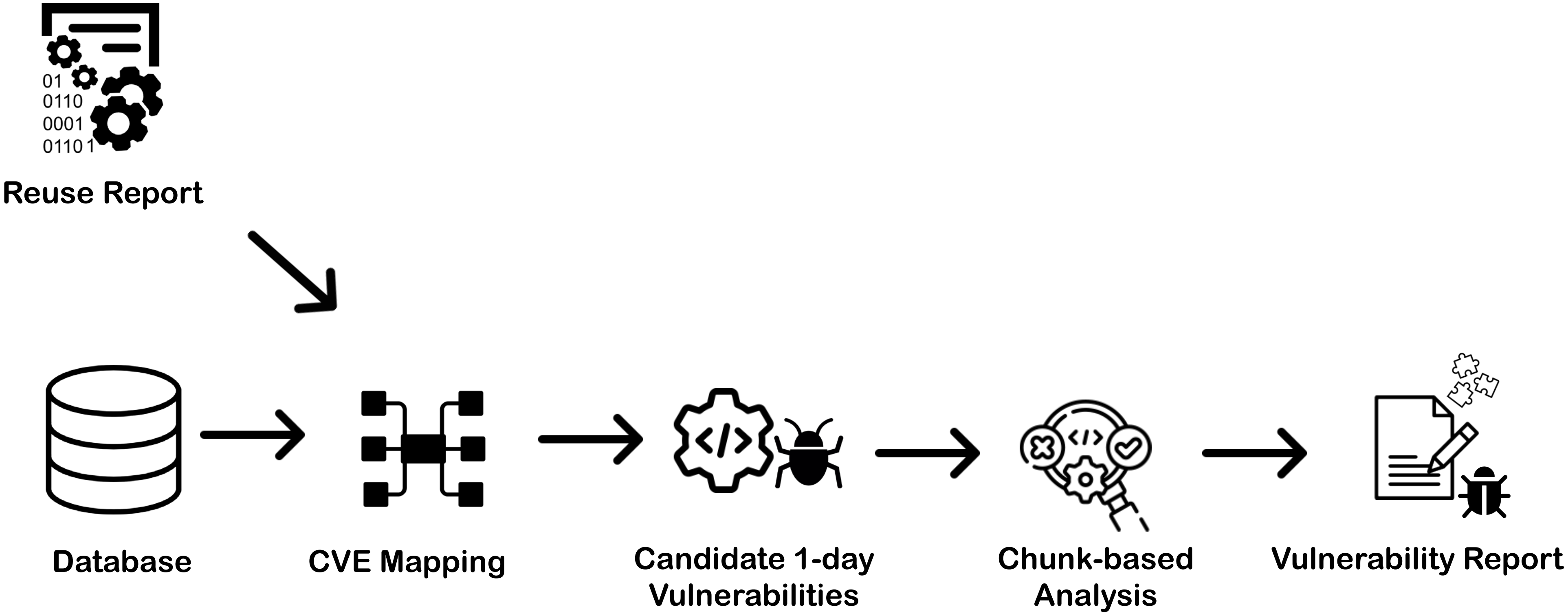
```
Line numbers [1,2,3,4]
Variable [cmaplen, source->cmap_length]
Operation [
 cmaplen: {assign;left;
           source->cmap_length}
          { >=; right; t}
 source->cmap_length:
          {assign; right;cmaplen}
]
```

Patch applied by ReactOS

Identical, patch applied,
no 1-day vulnerability

Overview of VULTURE

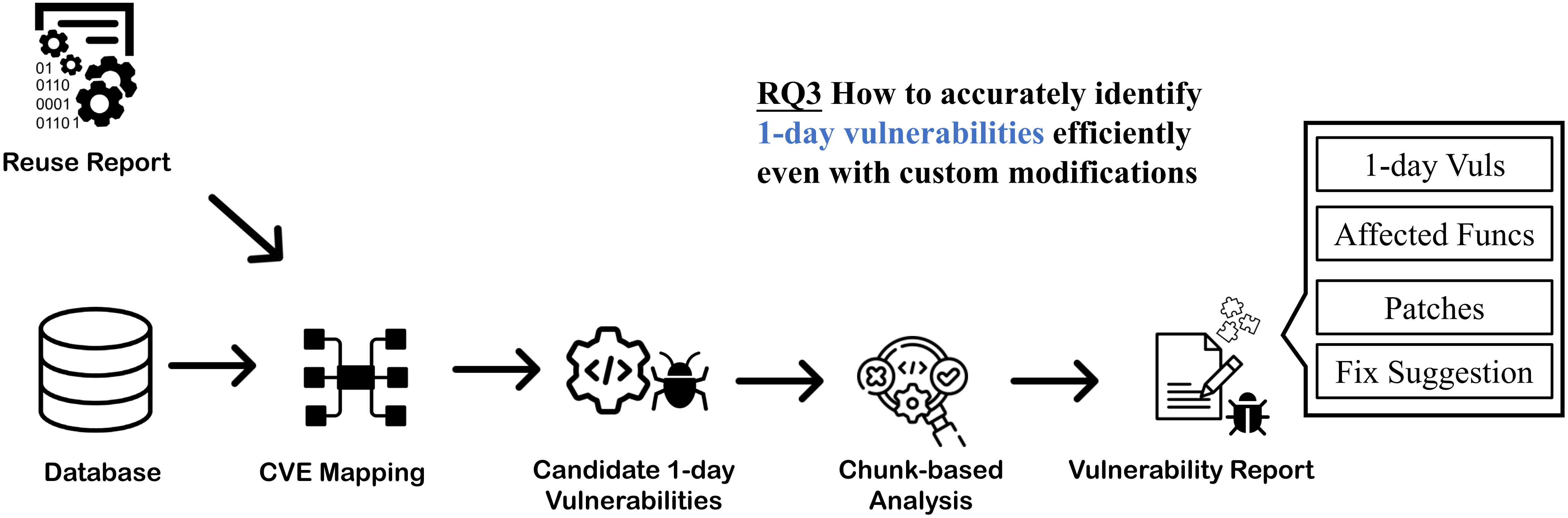
1-day Vulnerability Detection



Overview of VULTURE

1-day Vulnerability Detection

RQ3 How to accurately identify 1-day vulnerabilities efficiently even with custom modifications



Reuse Report

Database

CVE Mapping

Candidate 1-day Vulnerabilities

Chunk-based Analysis

Vulnerability Report

- 1-day Vuls
- Affected Funcs
- Patches
- Fix Suggestion

Contents

1. Background
   - Motivation
   - Research Questions
   - Previous Works

2. VULTURE Overview
   - Database Construction
   - TPL Reuse Detection
   - 1-day Vulnerability Detection

3. Evaluation and Results
   - Database Quality
   - Benchmark Vulnerability Detection
   - Vulnerability Detection In the Wild

Experiment Setup

Database Quality

- Storage efficiency.
- Time cost.
- Detection accuracy.

Experiment Setup

Database Quality

- Storage efficiency.
- Time cost.
- Detection accuracy.

Baseline:

- Centris: TPL reuse detector
- VFCFinder: Patch collector

Experiment Setup

Scalability Evaluation

- Benchmark Vulnerability Detection
  - Custom benchmark 200 reuse cases
- Vulnerability Detection In the Wild
  - 10 real-world popular programs.
  - TPL reuse detection.
  - 1-day vulnerability detection.
  - Time cost.

Experiment Setup

Scalability Evaluation

- Benchmark Vulnerability Detection
  - Custom benchmark 200 reuse cases
- Vulnerability Detection In the Wild
  - 10 real-world popular programs.
  - TPL reuse detection.
  - 1-day vulnerability detection.
  - Time cost.

Baseline:

- V1SCAN: 1-day vulnerability detector
- SNYK: Commercial 1-day vulnerability detector

UNSW CANBERRA

Database Quality

Storage efficiency & Time cost

Tool	TPL Number	Storage (GB)
VULTURE	1,872	3.5
Centris	10,288	20.0

Database Quality

Storage efficiency & Time cost

Tool	TPL Number	Storage (GB)
VULTURE	1,872	3.5
Centris	10,288	20.0

- **With TPLSelection, VULTURE excluded non-library repositories and retained only widely used TPLs, thereby reducing storage consumption.**

Database Quality

Storage efficiency & Time cost

Tool	TPL Number	Storage (GB)
VULTURE	1,872	3.5
Centris	10,288	20.0

Tool	Update database (s)	Frequency of Comparisons
VULTURE	0.1	9,207.1
Centris	115.1	1,508,924.7

Database Quality

Storage efficiency & Time cost

Tool	TPL Number	Storage (GB)
VULTURE	1,872	3.5
Centris	10,288	20.0

Tool	Update database (s)	Frequency of Comparisons
VULTURE	0.1	9,207.1
Centris	115.1	1,508,924.7

- **When adding new TPLs, VULTURE's database is more compact, making expansion easier and faster.**

UNSW CANBERRA

Database Quality

Storage efficiency & Time cost

Tool	Time Cost (s)	Memory Cost (MB)	F1(%)
VULTURE	84.68	3.5	87.94
VFCFinder	285.92	20.0	61.69

Database Quality

Storage efficiency & Time cost

Tool	Time Cost (s)	Memory Cost (MB)	F1(%)
VULTURE	84.68	3.5	87.94
VFCFinder	285.92	20.0	61.69

- **When collecting patches, VULTURE is more efficient and accurate, requiring less time and memory compared to VFCFinder.**

Benchmark Vulnerability Detection

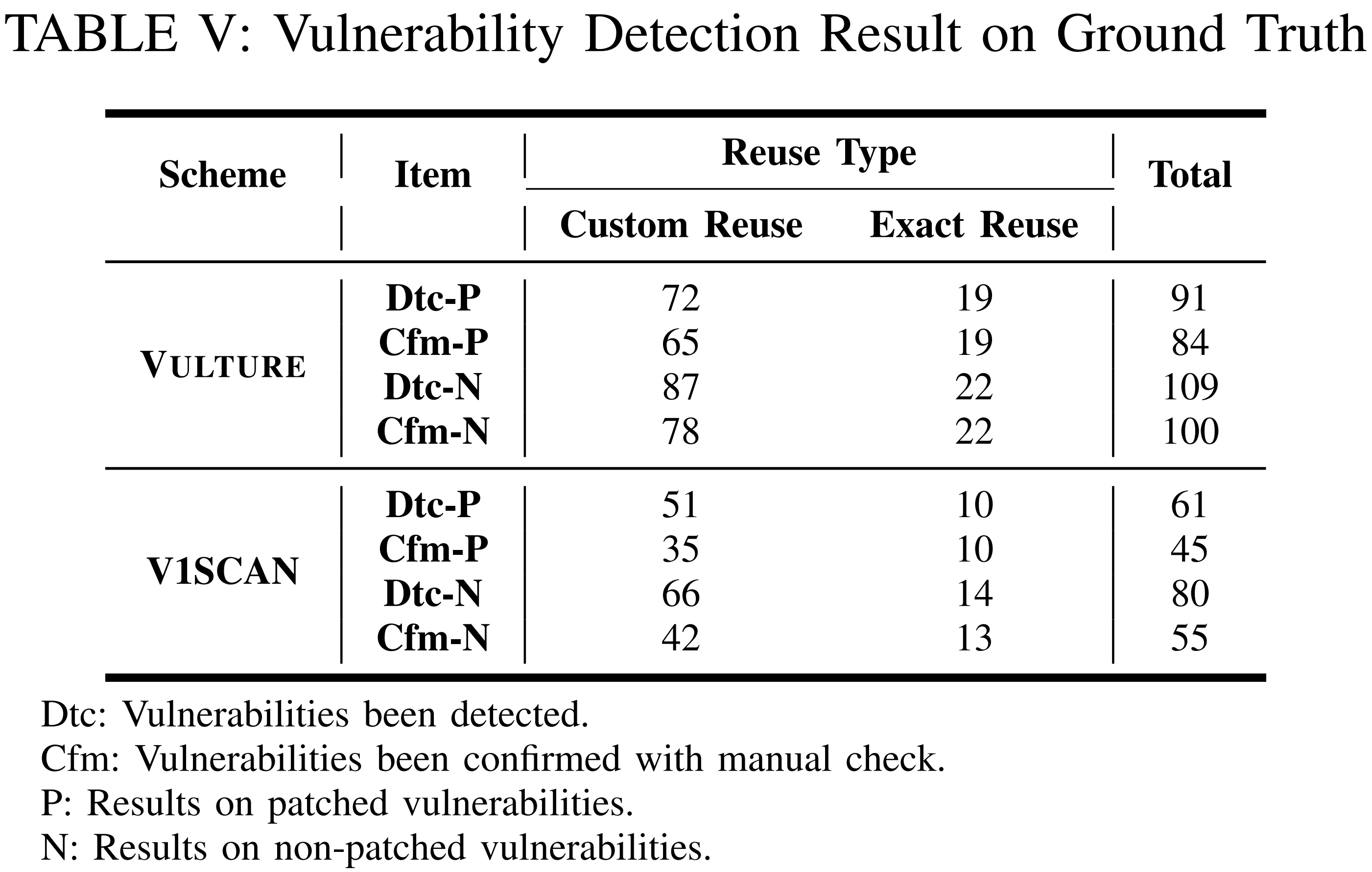


TABLE V: Vulnerability Detection Result on Ground Truth

Scheme	Item	Reuse Type		Total
		Custom Reuse	Exact Reuse	
Vulture	**Dtc-P**	72	19	91
	Cfm-P	65	19	84
	Dtc-N	87	22	109
	Cfm-N	78	22	100
V1SCAN	**Dtc-P**	51	10	61
	Cfm-P	35	10	45
	Dtc-N	66	14	80
	Cfm-N	42	13	55

Dtc: Vulnerabilities been detected.
Cfm: Vulnerabilities been confirmed with manual check.
P: Results on patched vulnerabilities.
N: Results on non-patched vulnerabilities.

Benchmark Vulnerability Detection

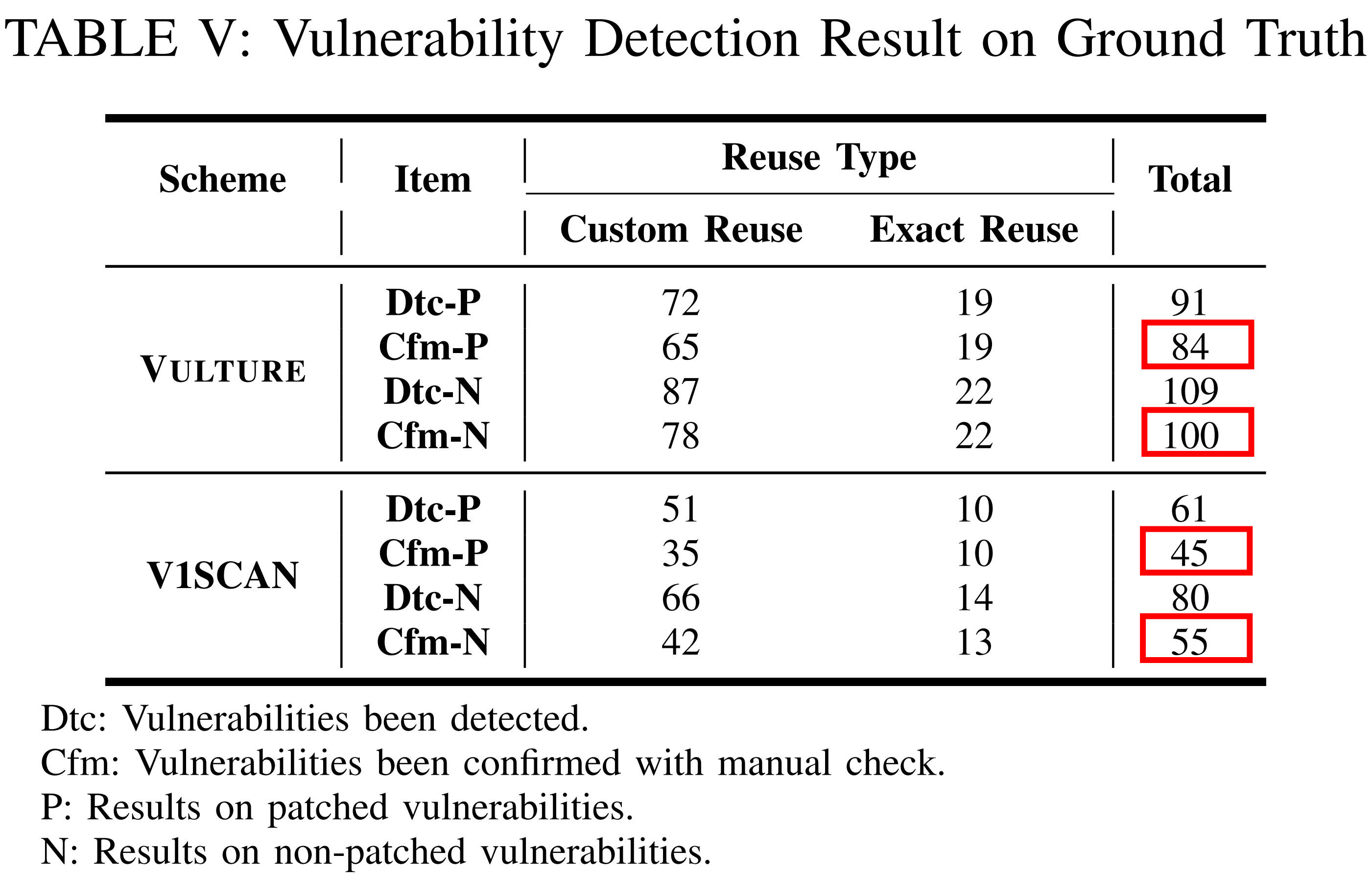


TABLE V: Vulnerability Detection Result on Ground Truth

Scheme	Item	Reuse Type		Total
		Custom Reuse	Exact Reuse	
VULTURE	Dtc-P	72	19	91
	Cfm-P	65	19	84
	Dtc-N	87	22	109
	Cfm-N	78	22	100
V1SCAN	Dtc-P	51	10	61
	Cfm-P	35	10	45
	Dtc-N	66	14	80
	Cfm-N	42	13	55

Dtc: Vulnerabilities been detected.
Cfm: Vulnerabilities been confirmed with manual check.
P: Results on patched vulnerabilities.
N: Results on non-patched vulnerabilities.

- **In 200 test cases, VULTURE successfully identified 184 cases.**

- **V1SCAN successfully identified 100 cases.**

Vulnerability Detection In the Wild

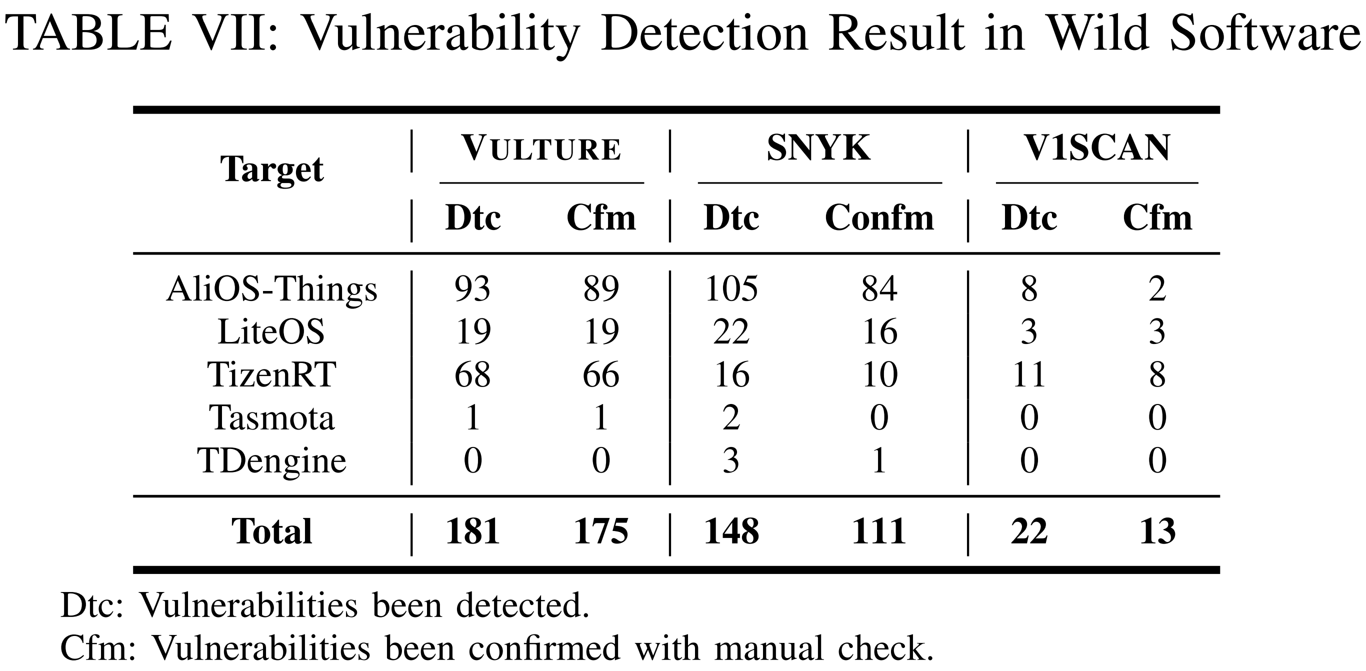


TABLE VII: Vulnerability Detection Result in Wild Software

Target	VULTURE		SNYK		V1SCAN	
	Dtc	**Cfm**	**Dtc**	**Confm**	**Dtc**	**Cfm**
AliOS-Things	93	89	105	84	8	2
LiteOS	19	19	22	16	3	3
TizenRT	68	66	16	10	11	8
Tasmota	1	1	2	0	0	0
TDengine	0	0	3	1	0	0
Total	**181**	**175**	**148**	**111**	**22**	**13**

Dtc: Vulnerabilities been detected.
Cfm: Vulnerabilities been confirmed with manual check.

Vulnerability Detection In the Wild

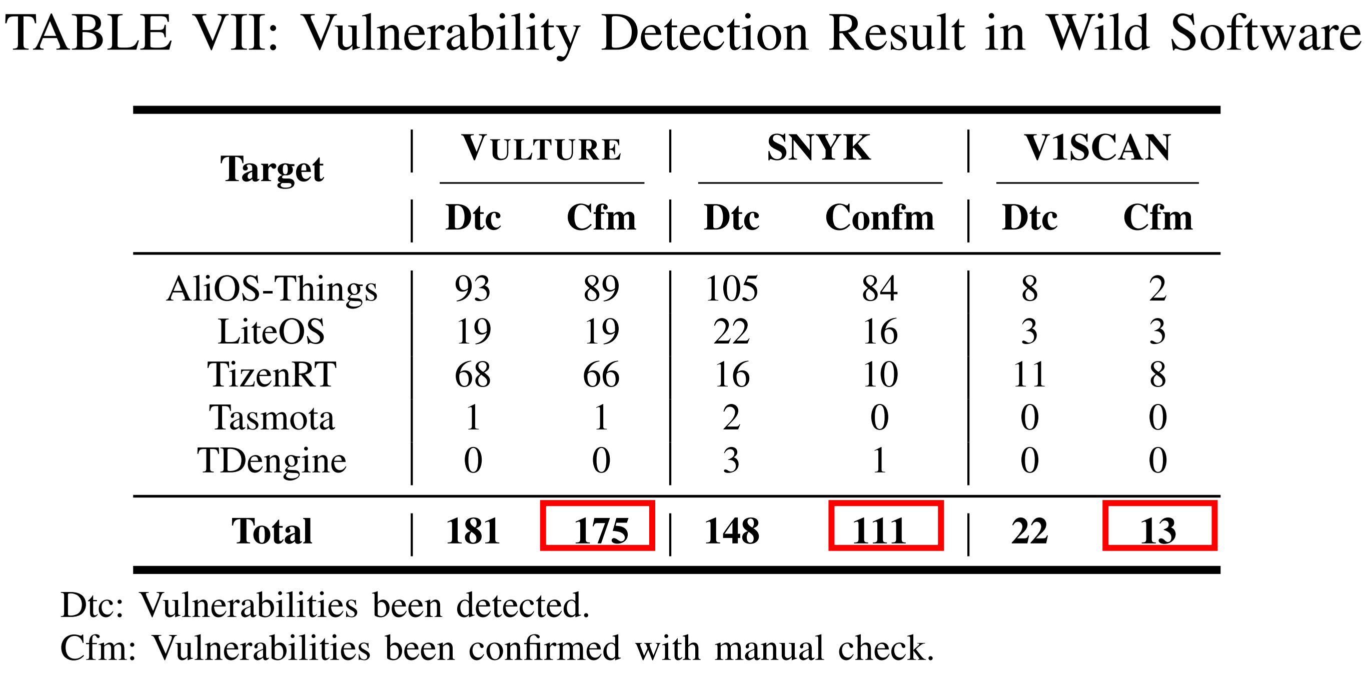


TABLE VII: Vulnerability Detection Result in Wild Software

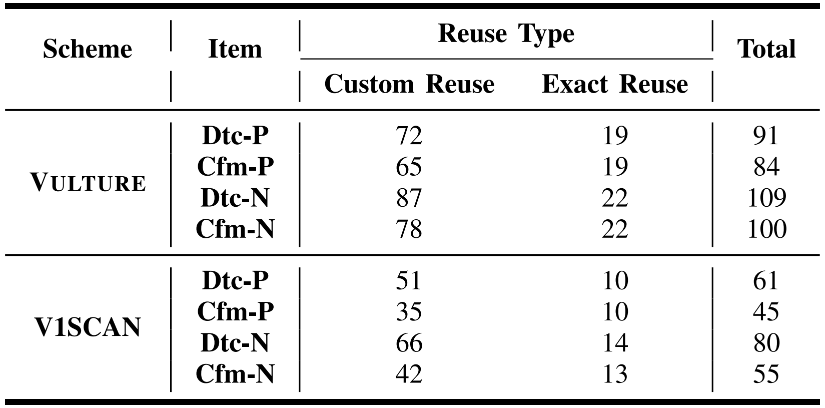
Target	VULTURE		SNYK		V1SCAN	
	Dtc	**Cfm**	**Dtc**	**Confm**	**Dtc**	**Cfm**
AliOS-Things	93	89	105	84	8	2
LiteOS	19	19	22	16	3	3
TizenRT	68	66	16	10	11	8
Tasmota	1	1	2	0	0	0
TDengine	0	0	3	1	0	0
Total	**181**	**175**	**148**	**111**	**22**	**13**

Dtc: Vulnerabilities been detected.
Cfm: Vulnerabilities been confirmed with manual check.

- **In 5 target programs, VULTURE successfully identified 175 1-day vulnerabilities.**
- **Commercial tool SNYK identified 111 vulnerabilities.**
- **V1SCAN successfully identified 13 vulnerabilities.**

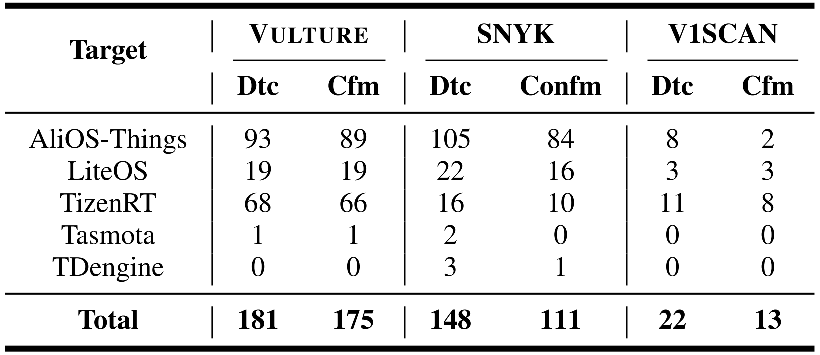
Scrutinize of Results

TABLE V: Vulnerability Detection Result on Ground Truth



Scheme	Item	Reuse Type		Total
		Custom Reuse	Exact Reuse	
VULTURE	Dtc-P	72	19	91
	Cfm-P	65	19	84
	Dtc-N	87	22	109
	Cfm-N	78	22	100
V1SCAN	Dtc-P	51	10	61
	Cfm-P	35	10	45
	Dtc-N	66	14	80
	Cfm-N	42	13	55

TABLE VII: Vulnerability Detection Result in Wild Software



Target	VULTURE		SNYK		V1SCAN	
	Dtc	Cfm	Dtc	Confm	Dtc	Cfm
AliOS-Things	93	89	105	84	8	2
LiteOS	19	19	22	16	3	3
TizenRT	68	66	16	10	11	8
Tasmota	1	1	2	0	0	0
TDengine	0	0	3	1	0	0
Total	**181**	**175**	**148**	**111**	**22**	**13**

- **VULTURE performs best with its well-designed database and accurate semantic analysis.**
- **The comprehensiveness and specificity of the database eliminates false negatives.**
- **False positives are reduced with TPL reuse identification optimization and chunk-based analysis.**

Vulnerability Detection In the Wild

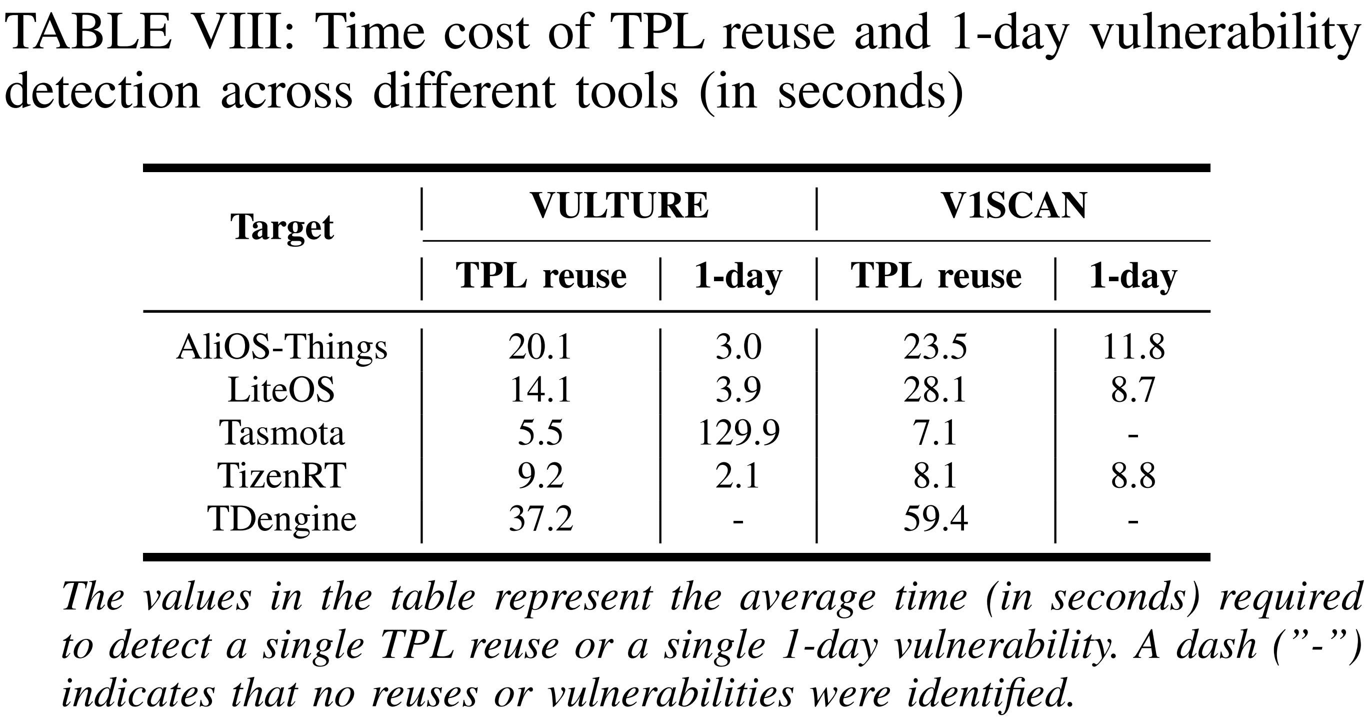


TABLE VIII: Time cost of TPL reuse and 1-day vulnerability detection across different tools (in seconds)

Target	**VULTURE**		**V1SCAN**	
	TPL reuse	**1-day**	**TPL reuse**	**1-day**
AliOS-Things	20.1	3.0	23.5	11.8
LiteOS	14.1	3.9	28.1	8.7
Tasmota	5.5	129.9	7.1	-
TizenRT	9.2	2.1	8.1	8.8
TDengine	37.2	-	59.4	-

The values in the table represent the average time (in seconds) required to detect a single TPL reuse or a single 1-day vulnerability. A dash ("-") indicates that no reuses or vulnerabilities were identified.

Vulnerability Detection In the Wild

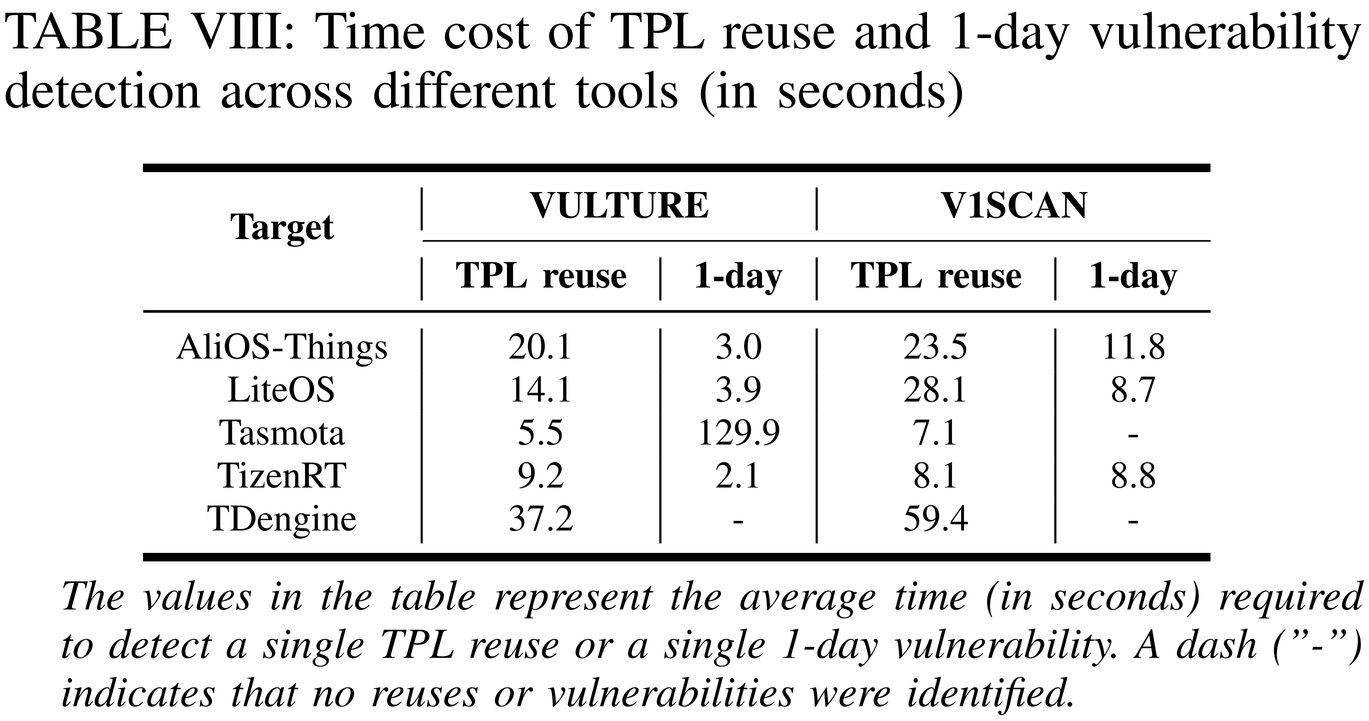


TABLE VIII: Time cost of TPL reuse and 1-day vulnerability detection across different tools (in seconds)

Target	VULTURE		V1SCAN	
	TPL reuse	1-day	TPL reuse	1-day
AliOS-Things	20.1	3.0	23.5	11.8
LiteOS	14.1	3.9	28.1	8.7
Tasmota	5.5	129.9	7.1	-
TizenRT	9.2	2.1	8.1	8.8
TDengine	37.2	-	59.4	-

The values in the table represent the average time (in seconds) required to detect a single TPL reuse or a single 1-day vulnerability. A dash (”-”) indicates that no reuses or vulnerabilities were identified.

- **VULTURE detects one TPL reuse in 25 seconds and one 1-day vulnerabilities in 5 seconds.**
- **V1SCAN takes longer for extensive TPL reuse**

Conclusion

- We introduced VULTURE, leveraging static analysis and LLMs to support the whole process of 1-day vulnerability detection.
- VULTURE reduces false alarms and improved detection accuracy, surpasses existing academic and commercial tools.

Thanks

Q & A