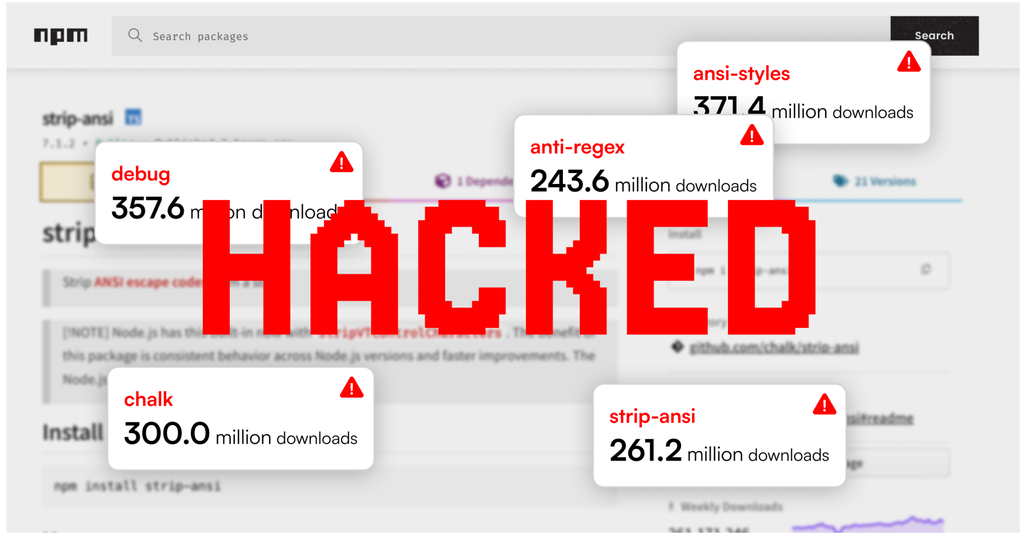
Student Research Competition

Verification and Classification of Exploits for Node.js Vulnerabilities

Sungmin Park

KOREA UNIVERSITY

Javascript Library Vulnerabilities on Web



20 popular **npm packages** with **2 billion weekly downloads**

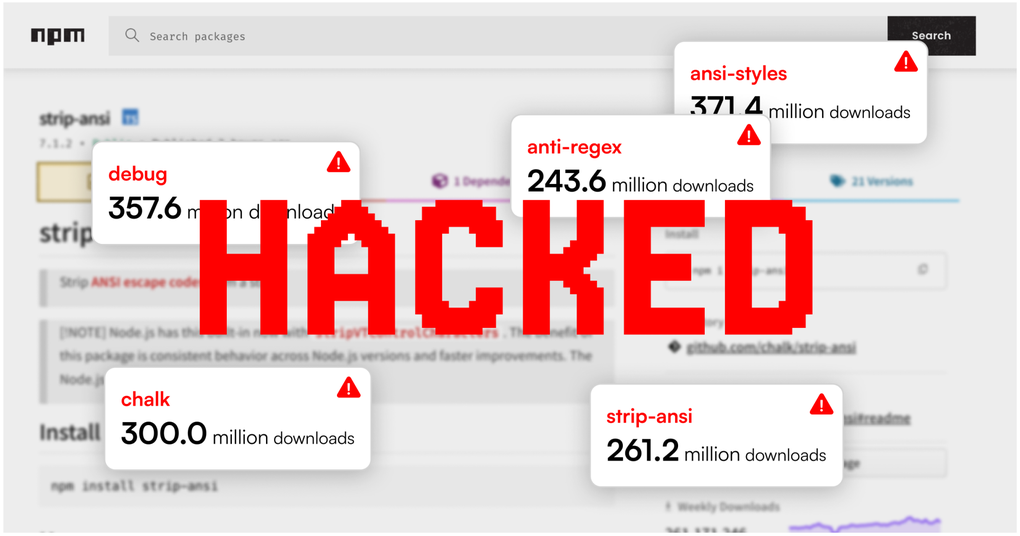
compromised in **supply chain attack**.

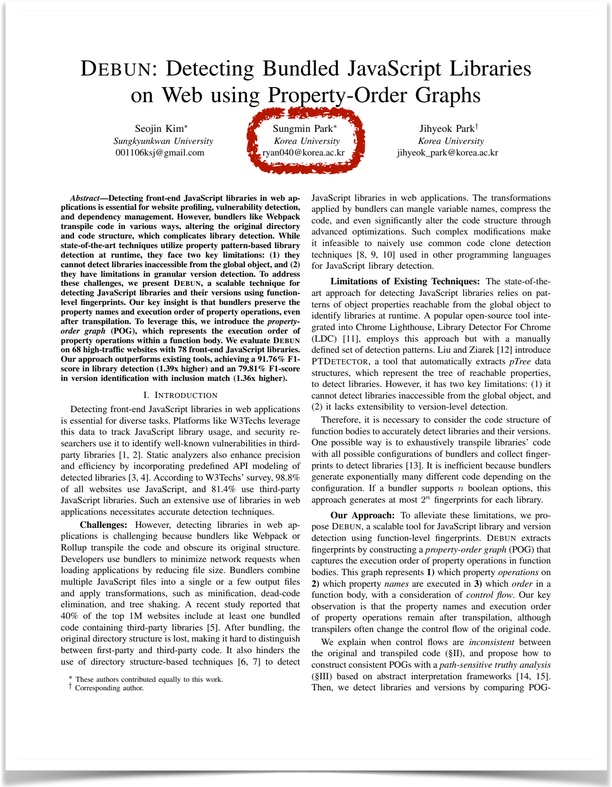
The Hacker News. (**2025, September**)

PLRG

Javascript Library Vulnerabilities on Web

[ASE'25] DEBUN: **Detecting** Bundled **JavaScript Libraries** on **Web** using Property-Order Graphs





DEBUN: Detecting Bundled JavaScript Libraries on Web using Property-Order Graphs

Seojin Kim*
Sungkyunkwan University
001106ksj@gmail.com

Sungmin Park*
Korea University
ryan040@korea.ac.kr

Jihyeok Park†
Korea University
jihyeok_park@korea.ac.kr

***Abstract*—Detecting front-end JavaScript libraries in web applications is essential for website profiling, vulnerability detection, and dependency management. However, bundlers like Webpack transpile code in various ways, altering the original directory and code structure, which complicates library detection. While state-of-the-art techniques utilize property pattern-based library detection at runtime, they face two key limitations: (1) they cannot detect libraries inaccessible from the global object, and (2) they have limitations in granular version detection. To address these challenges, we present DEBUN, a scalable technique for detecting JavaScript libraries and their versions using function-level fingerprints. Our key insight is that bundlers preserve the property names and execution order of property operations, even after transpilation. To leverage this, we introduce the *property-order graph* (POG), which represents the execution order of property operations within a function body. We evaluate DEBUN on 68 high-traffic websites with 78 front-end JavaScript libraries. Our approach outperforms existing tools, achieving a 91.76% F1-score in library detection (1.39x higher) and an 79.81% F1-score in version identification with inclusion match (1.36x higher).**

I. INTRODUCTION

Detecting front-end JavaScript libraries in web applications is essential for diverse tasks. Platforms like W3Techs leverage this data to track JavaScript library usage, and security researchers use it to identify well-known vulnerabilities in third-party libraries [1, 2]. Static analyzers also enhance precision and efficiency by incorporating predefined API modeling of detected libraries [3, 4]. According to W3Techs' survey, 98.8% of all websites use JavaScript, and 81.4% use third-party JavaScript libraries. Such an extensive use of libraries in web applications necessitates accurate detection techniques.

Challenges: However, detecting libraries in web applications is challenging because bundlers like Webpack or Rollup transpile the code and obscure its original structure. Developers use bundlers to minimize network requests when loading applications by reducing file size. Bundlers combine multiple JavaScript files into a single or a few output files and apply transformations, such as minification, dead-code elimination, and tree shaking. A recent study reported that 40% of the top 1M websites include at least one bundled code containing third-party libraries [5]. After bundling, the original directory structure is lost, making it hard to distinguish between first-party and third-party code. It also hinders the use of directory structure-based techniques [6, 7] to detect JavaScript libraries in web applications. The transformations applied by bundlers can mangle variable names, compress the code, and even significantly alter the code structure through advanced optimizations. Such complex modifications make it infeasible to naively use common code clone detection techniques [8, 9, 10] used in other programming languages for JavaScript library detection.

Limitations of Existing Techniques: The state-of-the-art approach for detecting JavaScript libraries relies on patterns of object properties reachable from the global object to identify libraries at runtime. A popular open-source tool integrated into Chrome Lighthouse, Library Detector For Chrome (LDC) [11], employs this approach but with a manually defined set of detection patterns. Liu and Ziarek [12] introduce PTDETECTOR, a tool that automatically extracts *pTree* data structures, which represent the tree of reachable properties, to detect libraries. However, it has two key limitations: (1) it cannot detect libraries inaccessible from the global object, and (2) it lacks extensibility to version-level detection.

Therefore, it is necessary to consider the code structure of function bodies to accurately detect libraries and their versions. One possible way is to exhaustively transpile libraries' code with all possible configurations of bundlers and collect fingerprints to detect libraries [13]. It is inefficient because bundlers generate exponentially many different code depending on the configuration. If a bundler supports n boolean options, this approach generates at most 2^n fingerprints for each library.

Our Approach: To alleviate these limitations, we propose DEBUN, a scalable tool for JavaScript library and version detection using function-level fingerprints. DEBUN extracts fingerprints by constructing a *property-order graph* (POG) that captures the execution order of property operations in function bodies. This graph represents **1)** which property *operations* on **2)** which property *names* are executed in **3)** which *order* in a function body, with a consideration of *control flow*. Our key observation is that the property names and execution order of property operations remain after transpilation, although transpilers often change the control flow of the original code.

We explain when control flows are *inconsistent* between the original and transpiled code (§II), and propose how to construct consistent POGs with a *path-sensitive truthy analysis* (§III) based on abstract interpretation frameworks [14, 15]. Then, we detect libraries and versions by comparing POG-

* These authors contributed equally to this work.
† Corresponding author.

20 popular **npm packages** with **2 billion weekly downloads**

compromised in **supply chain attack**.

The Hacker News. (**2025, September**)

PLRG

Next Step - Exploitability

Attacker input

PLRG

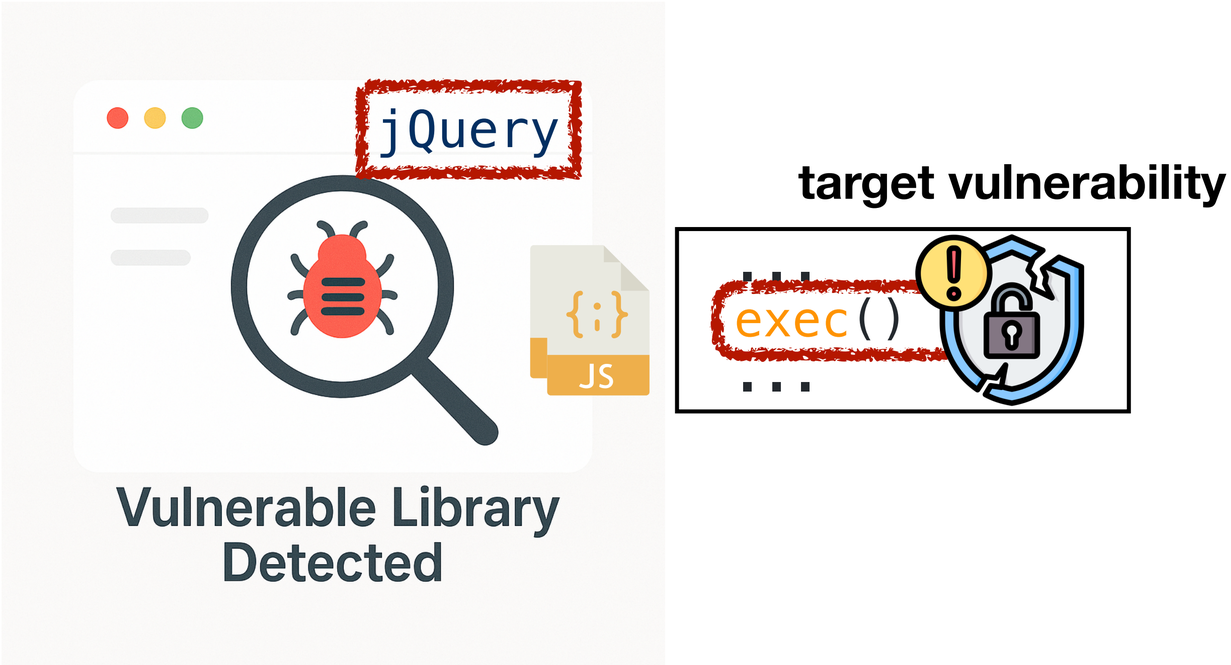
Next Step - Exploitability

Attacker input


```
...
exec()
...
```

Next Step - Exploitability

Attacker input



Next Step - Exploitability


```
const jq = require("jQuery");
jq('touch a;');
```

Next Step - Exploitability

Attacker input

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

target vulnerability

PLRG

Prior Work - PoC Generation

- **Manual Dataset**
  - SecBench.js (ICSE'23)
- **Static Analysis** (abstract interpretation, symbolic execution)
  - FAST (SP'23), Explode.js (PLDI'25)
- **Dynamic Analysis** (fuzzing)
  - NodeMedic (EuroS&P'23), NodeMedic-Fine (NDSS'25)
- **LLM**
  - PoCGen (arXiv)

CWE-78	Command-Injection	Attacker input **executed as OS commands**
CWE-1321	Prototype-Pollution	Attacker input **manipulates object prototype**
CWE-94	Code-Injection	Attacker input **executed as code**
CWE-1333	ReDos	Attacker input **cause excessive regex matching**

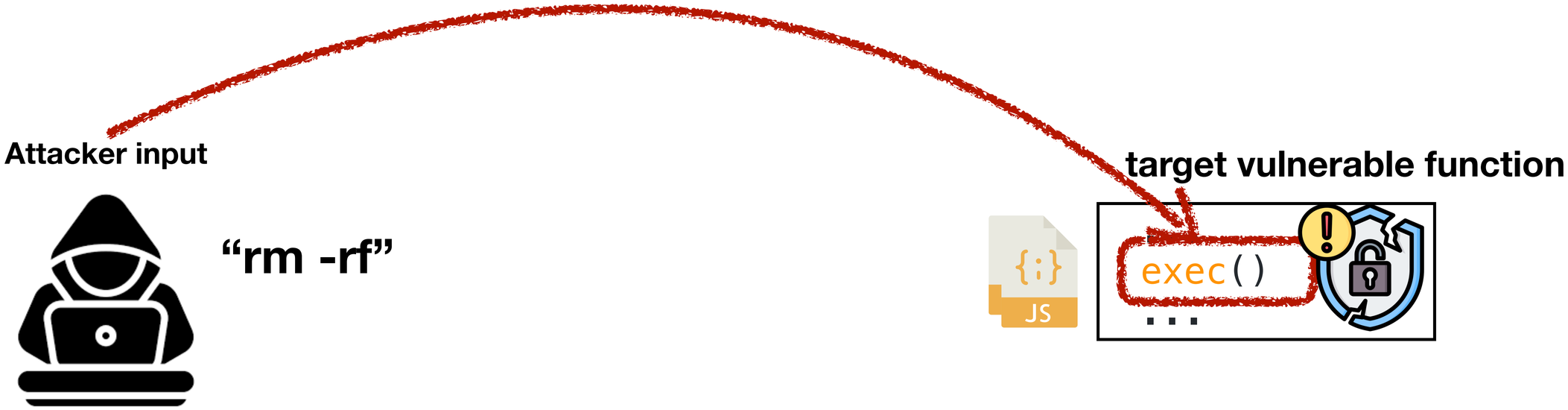
PLRG

Prior Work - PoC Generation

- **Manual Dataset**
  - SecBench.js (ICSE'23)
- **Static Analysis** (abstract interpretation, symbolic execution)
  - FAST (SP'23), Explode.js (PLDI'25)
- **Dynamic Analysis** (fuzzing)
  - NodeMedic (EuroS&P'23), NodeMedic-Fine (NDSS'25)
- **LLM**
  - PoCGen (arXiv)

CWE-78	Command-Injection	Attacker input **executed as OS commands**

Command-Injection (CWE-78)



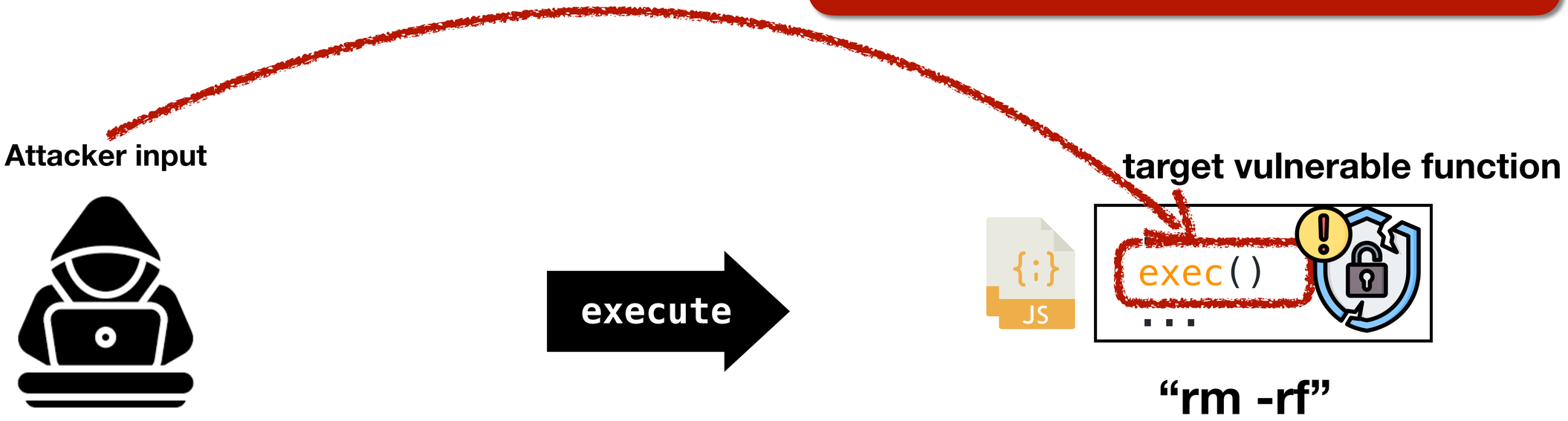
Command-Injection (CWE-78)

Can execute arbitrary OS command

PLRG

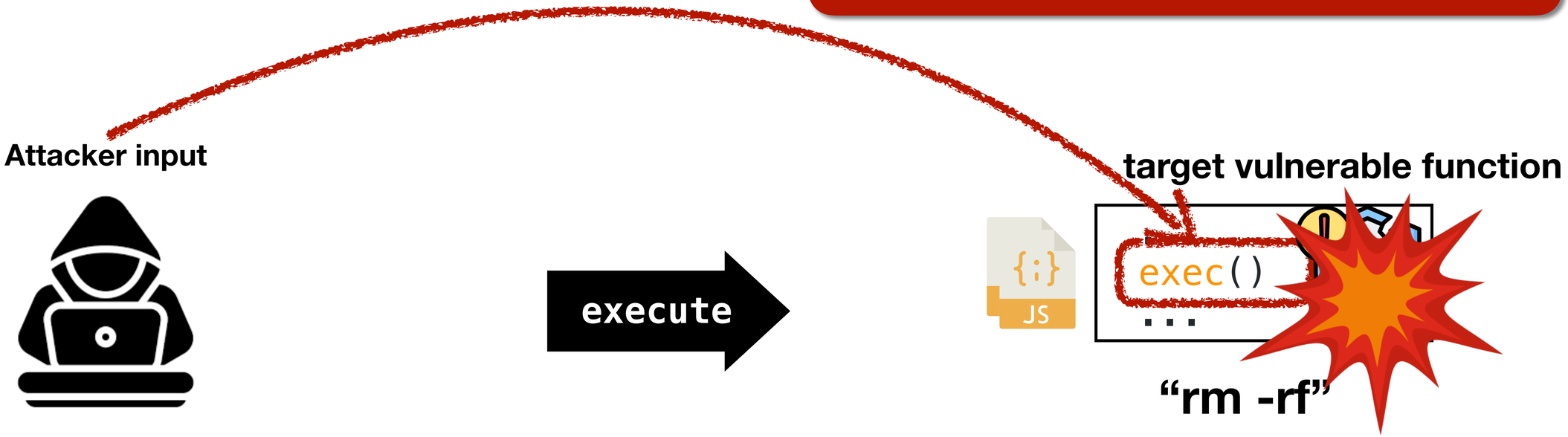
Command-Injection (CWE-78)

Can execute arbitrary OS command

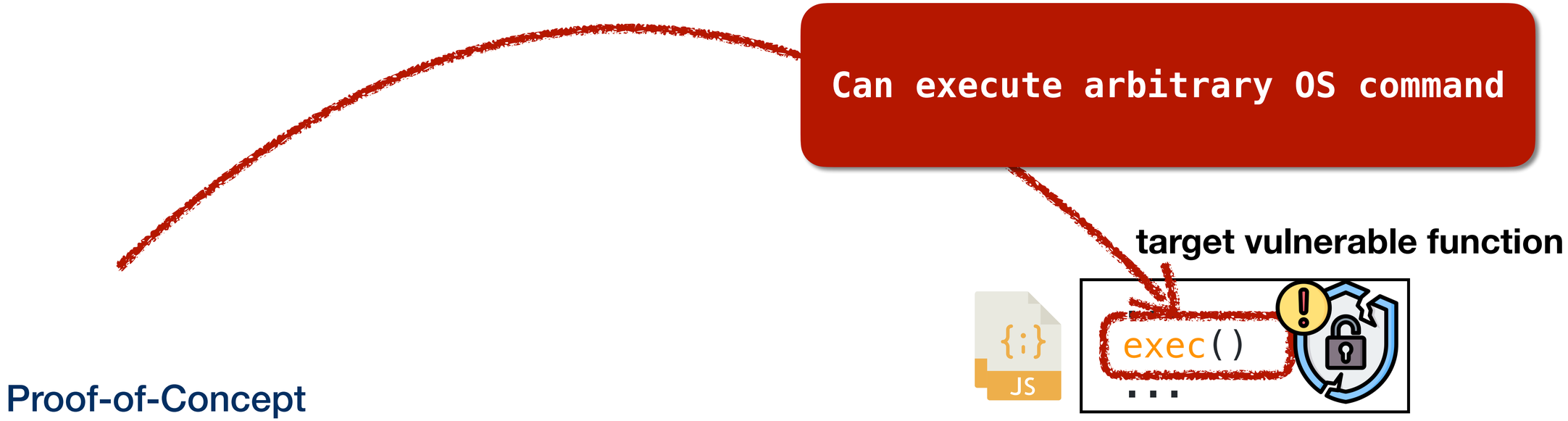


Command-Injection (CWE-78)

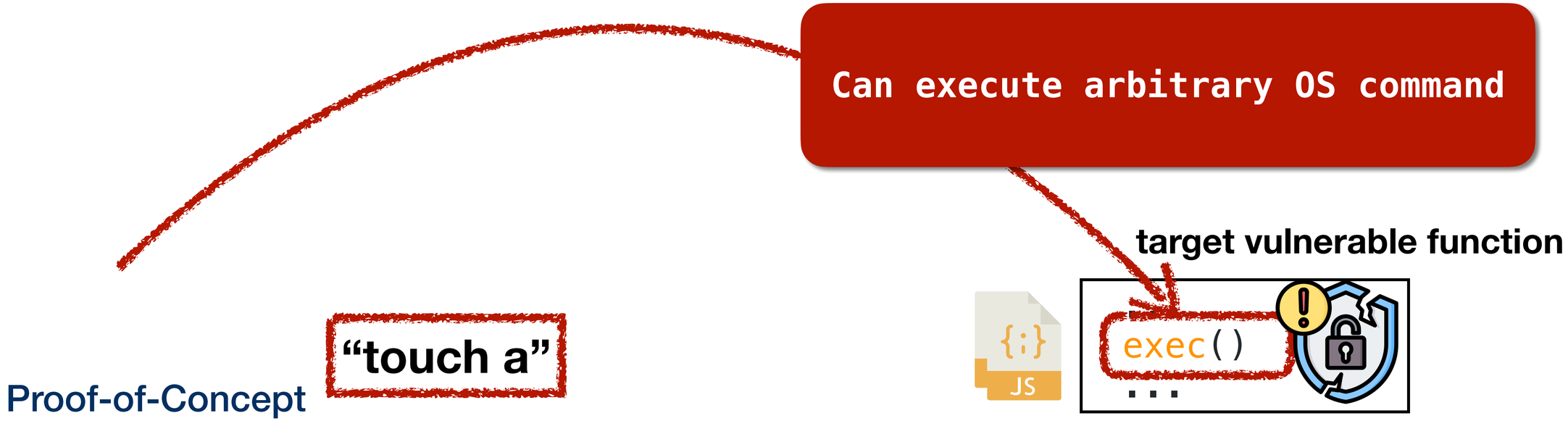
Can execute arbitrary OS command



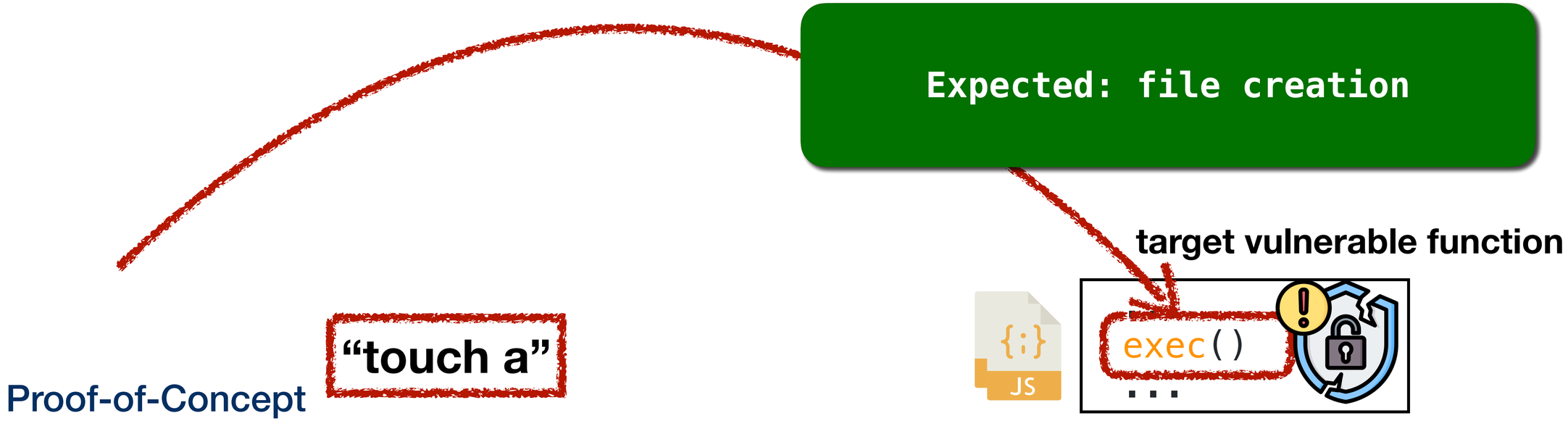
Command-Injection PoC



Command-Injection PoC



Command-Injection PoC



Command-Injection PoC

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

“touch a”

Expected: file creation

target vulnerable function

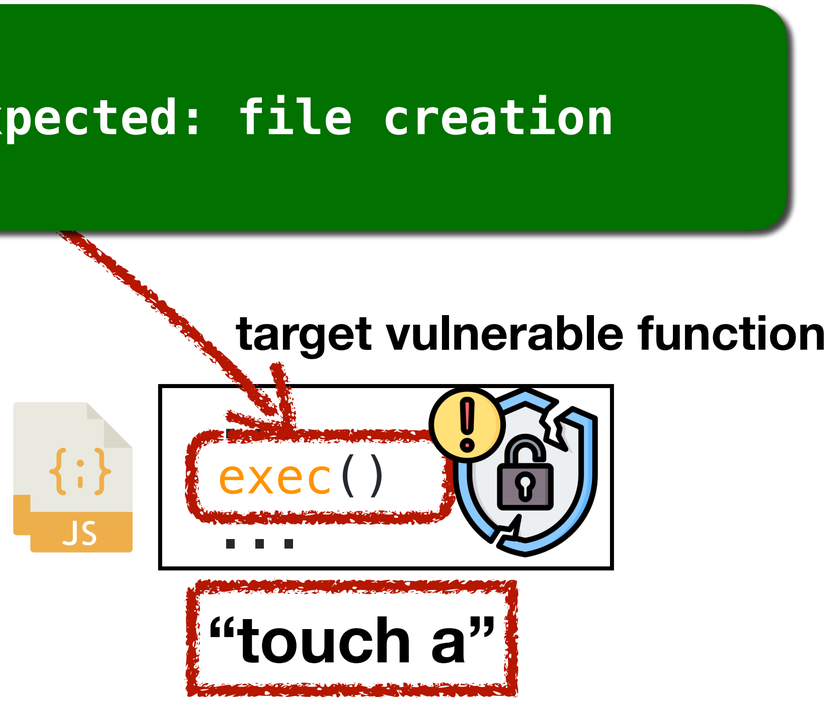
PLRG

Command-Injection PoC

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

execute



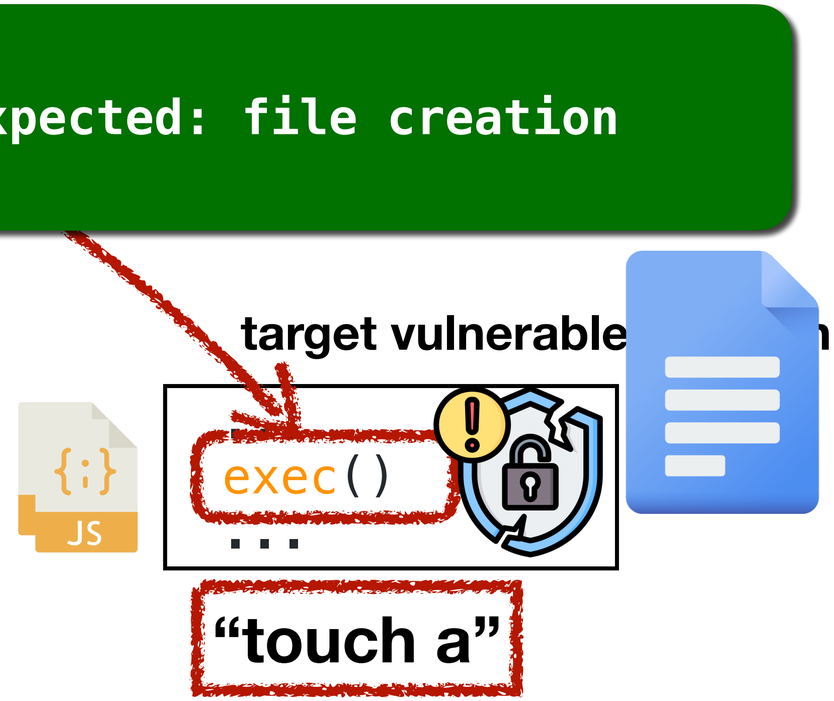
PLRG

Command-Injection PoC

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

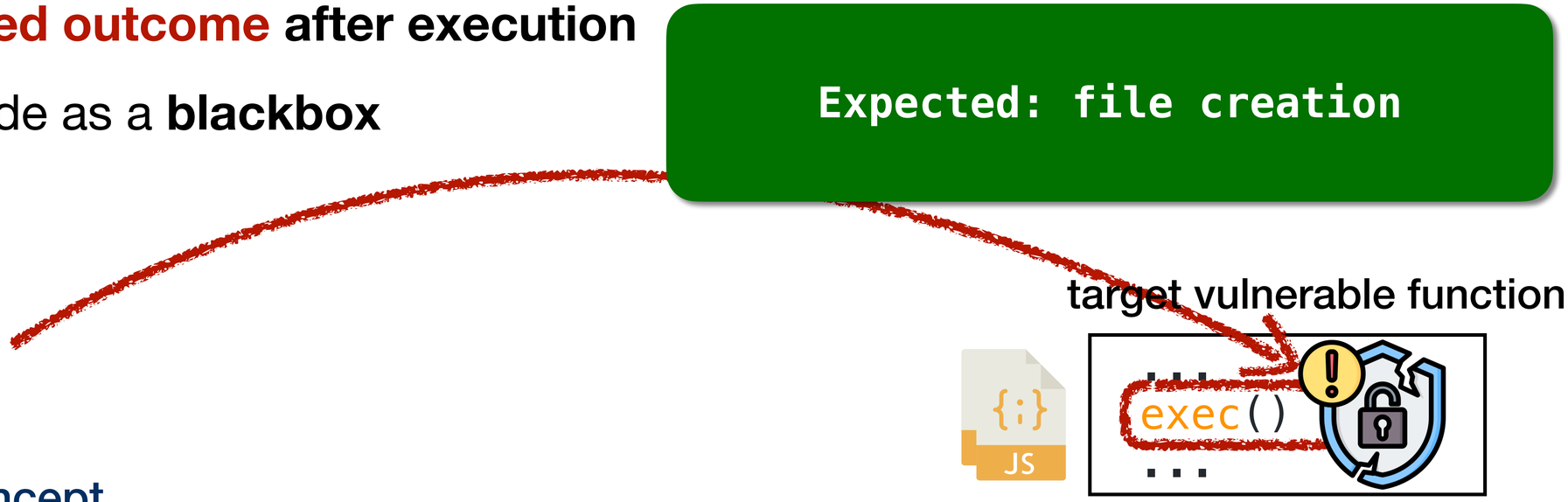
execute



PLRG

Prior Work - PoC Generation

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**



Proof-of-Concept

Prior Work - PoC Generation

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected: file creation

target vulnerable function

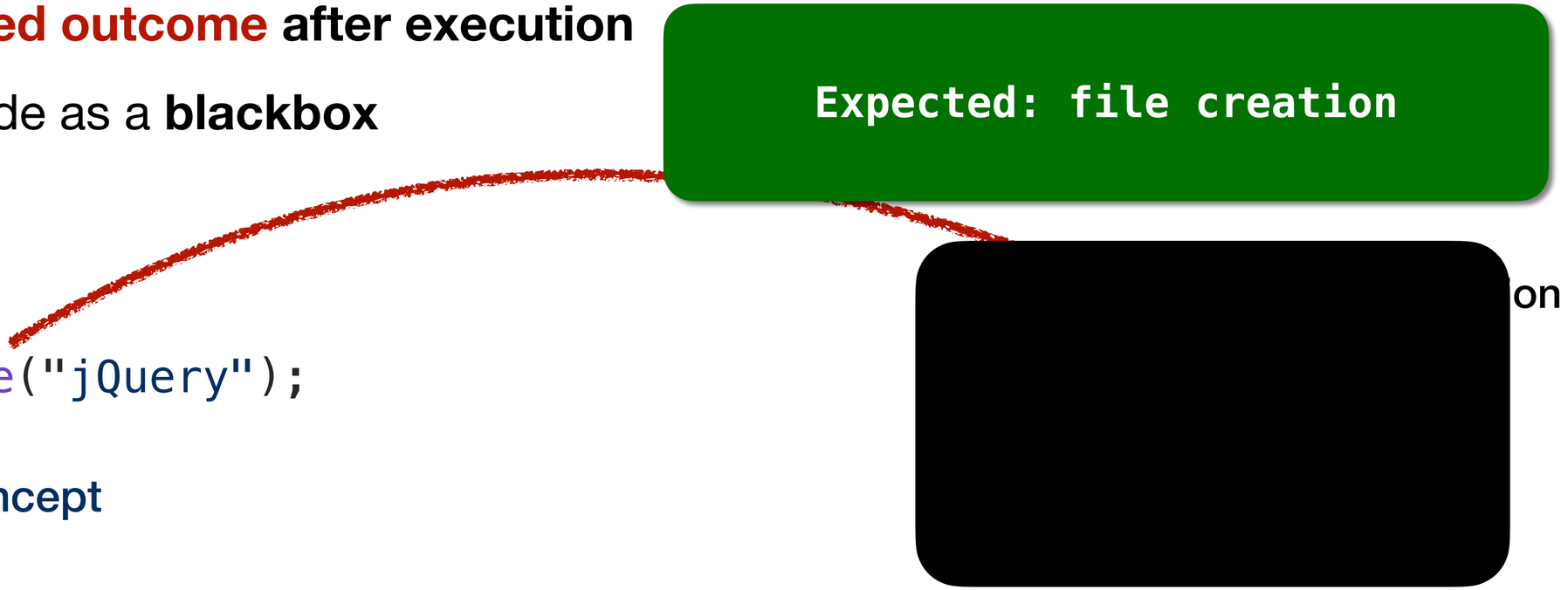

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

PLRG

Prior Work - PoC Generation

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

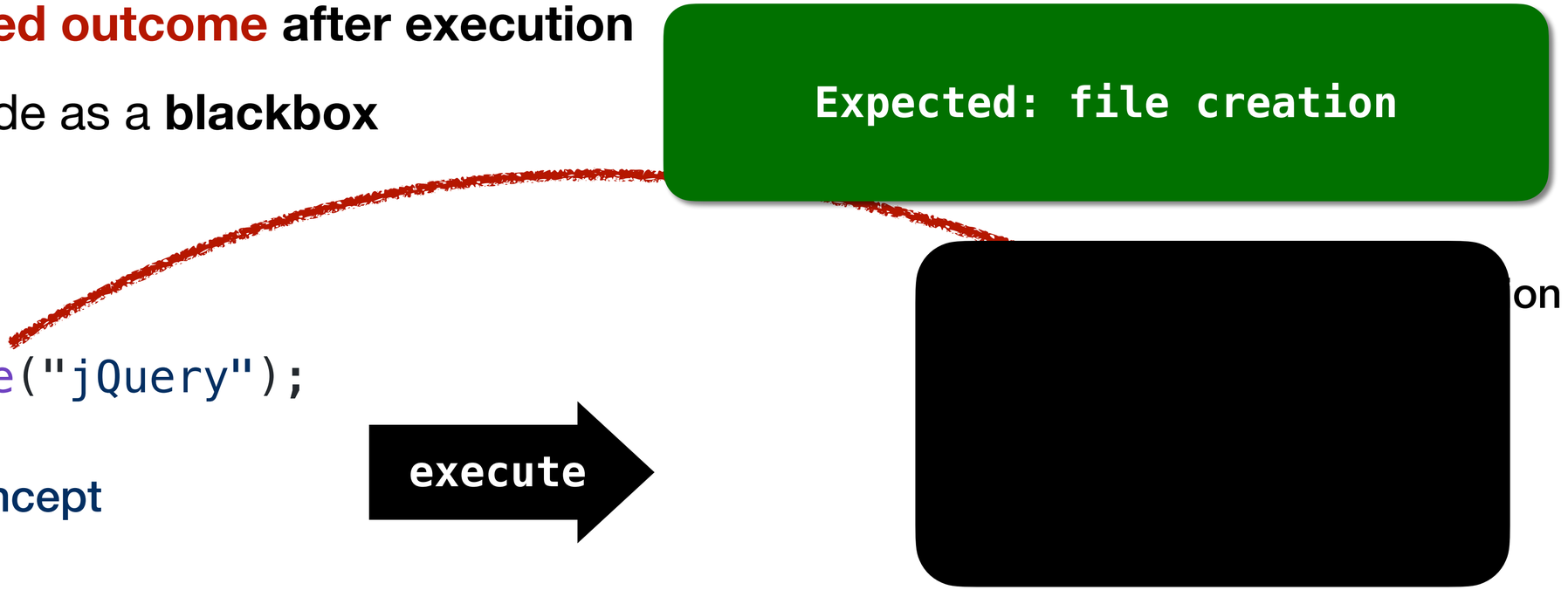



```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

Prior Work - PoC Generation

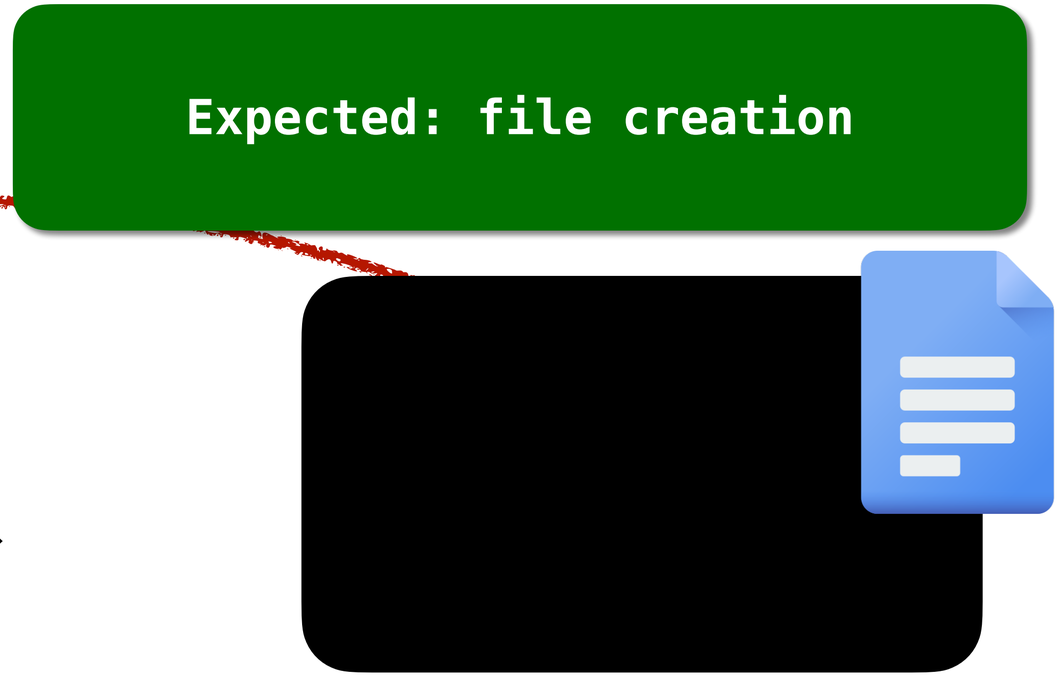
- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**



PLRG

Prior Work - PoC Generation

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**




```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

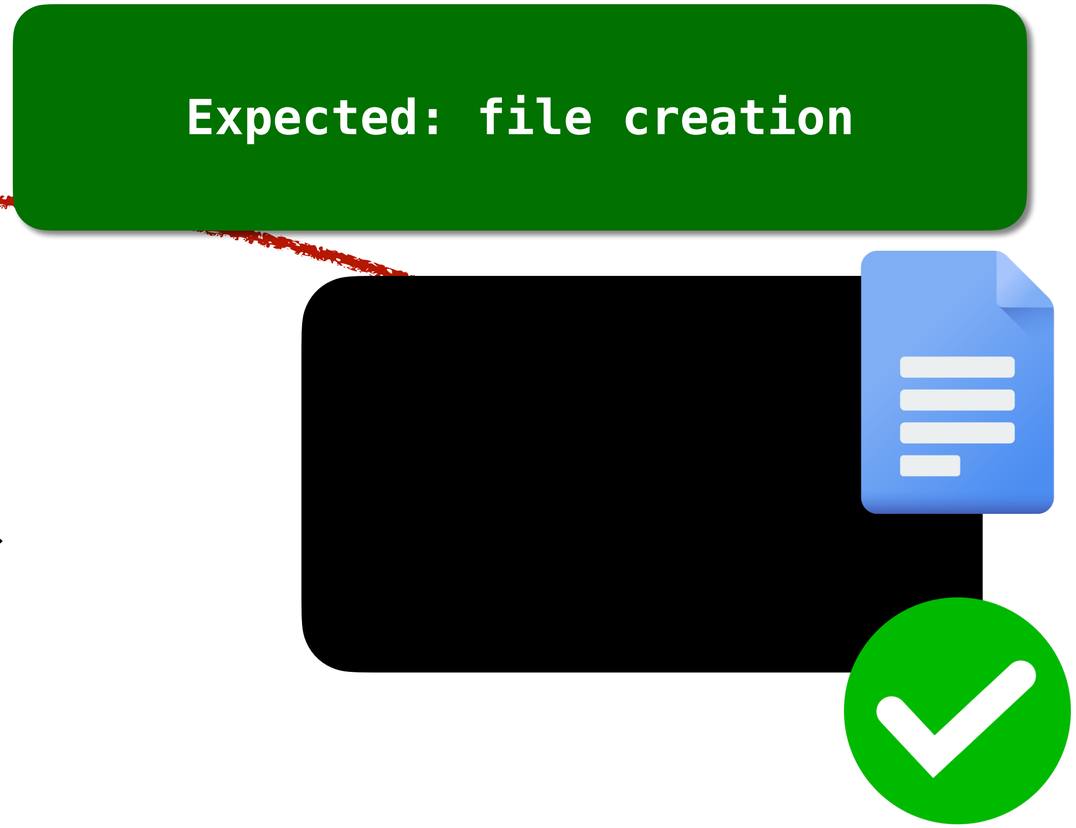
Prior Work - PoC Generation

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

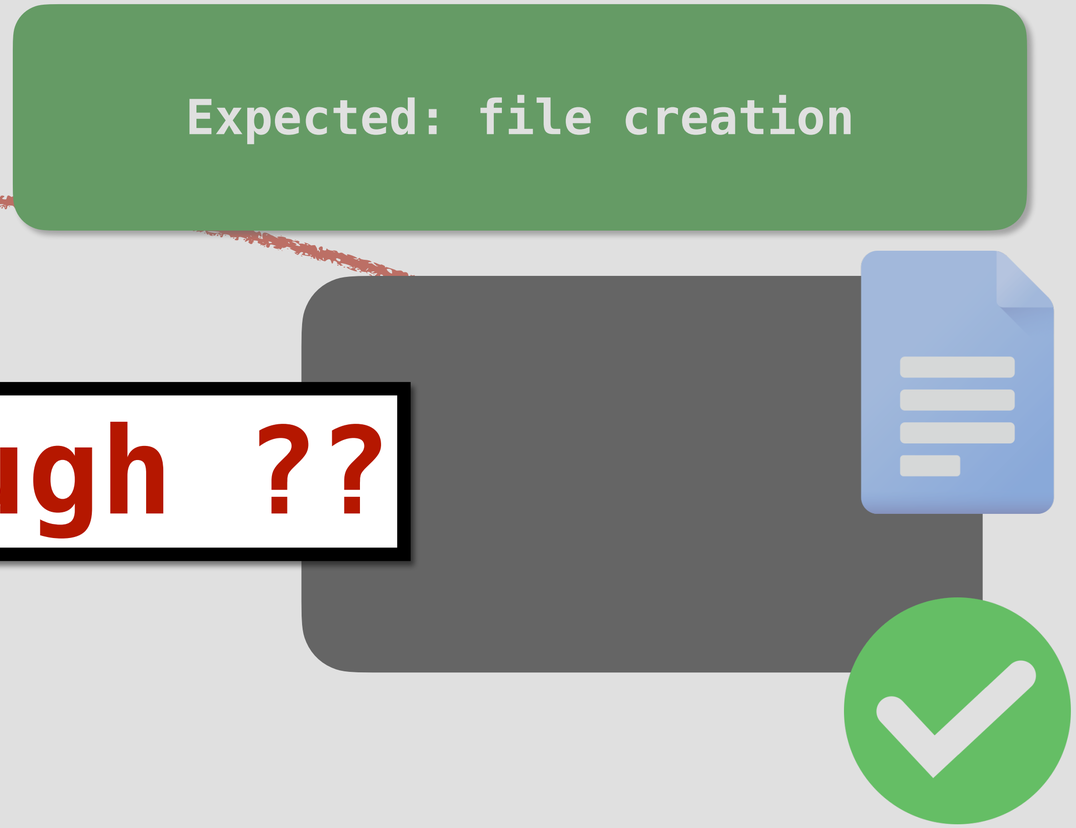
execute



PLRG

Prior Work - PoC Generation

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**



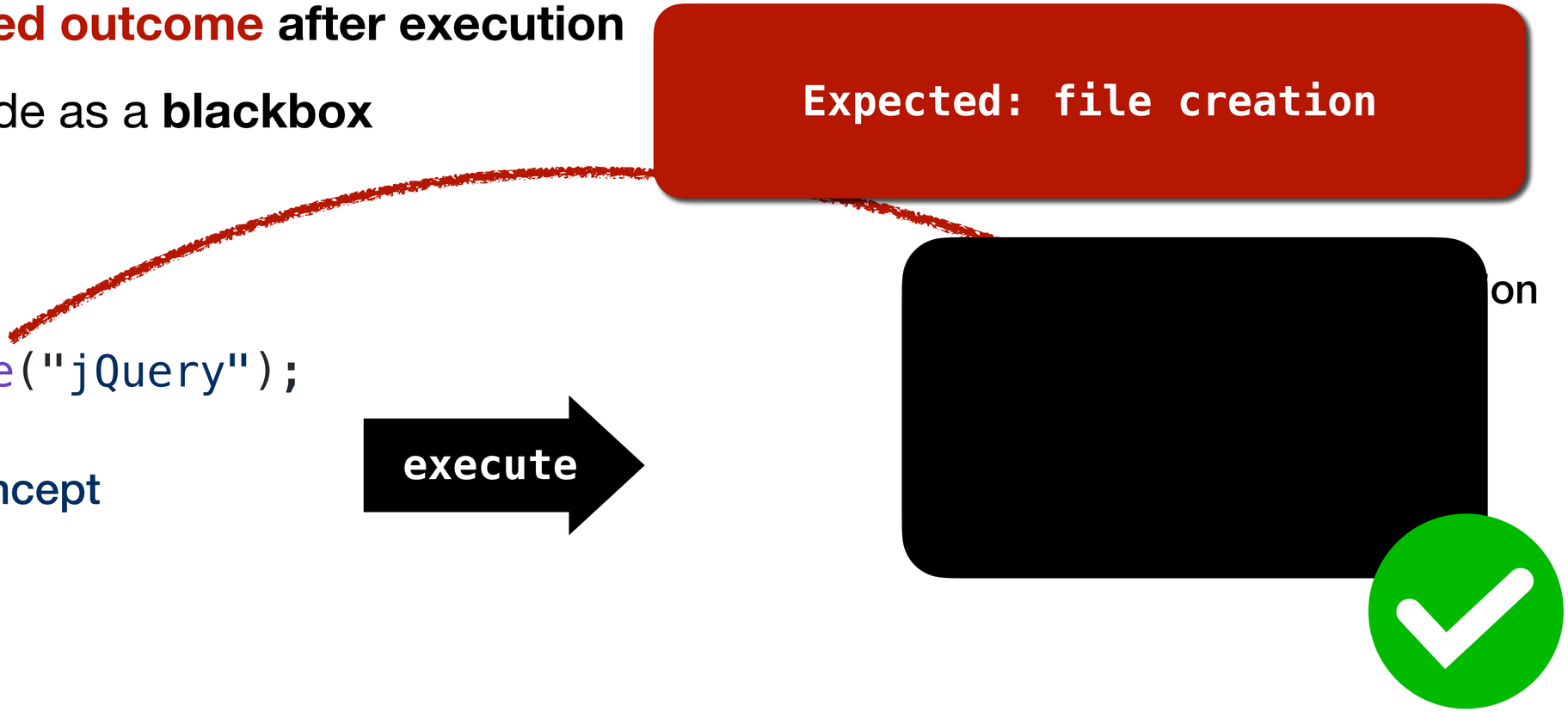

```
const jq = require("j
jq('touch a;');
```

Proof-of-Concept

PLRG

Prior Work - PoC Generation

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**



PLRG

Prior Work - PoC Generation

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

execute

Expected triggering the targeted vulnerability

target vulnerable function

PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected
triggering the targeted vulnerability

Command-Injection in function ‘set’

Proof-of-Concept

```
...
function set(callback){
    exec(getCommand);
}
function remove(callback){
    exec(addCommand);
}
...
```

PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected
triggering the targeted vulnerability

Command-Injection in function ‘set’

Proof-of-Concept

target vulnerable function

```
...
function set(callback){
    exec(getCommand);
}
function remove(callback){
    exec(addCommand);
}
...
```

PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected
triggering the targeted vulnerability

Command-Injection in function ‘set’

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

target vulnerable function

```
...
function set(callback){
    exec(getCommand);
}
function remove(callback){
    exec(addCommand);
}
...
```

PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected triggering the targeted vulnerability

Command-Injection in function ‘set’

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected
triggering the targeted vulnerability

Command-Injection in function ‘set’

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute

PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected triggering the targeted vulnerability

Command-Injection in function ‘set’

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute

PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected triggering the targeted vulnerability

Command-Injection in function ‘set’

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute

PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected
triggering the targeted vulnerability

Command-Injection in function 'set'

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute

```
...
function set(callback){
    exec(getCommand);
}
function remove(callback){
    exec(addCommand);
}
...
```

target vulnerable functi

PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected
triggering the targeted vulnerability

Command-Injection in function ‘set’

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

target vulnerable functi

```
...
function set(callback){
    exec(getCommand);
}
function remove(callback){
    exec(addCommand);
}
...
```

PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected
triggering the targeted vulnerability

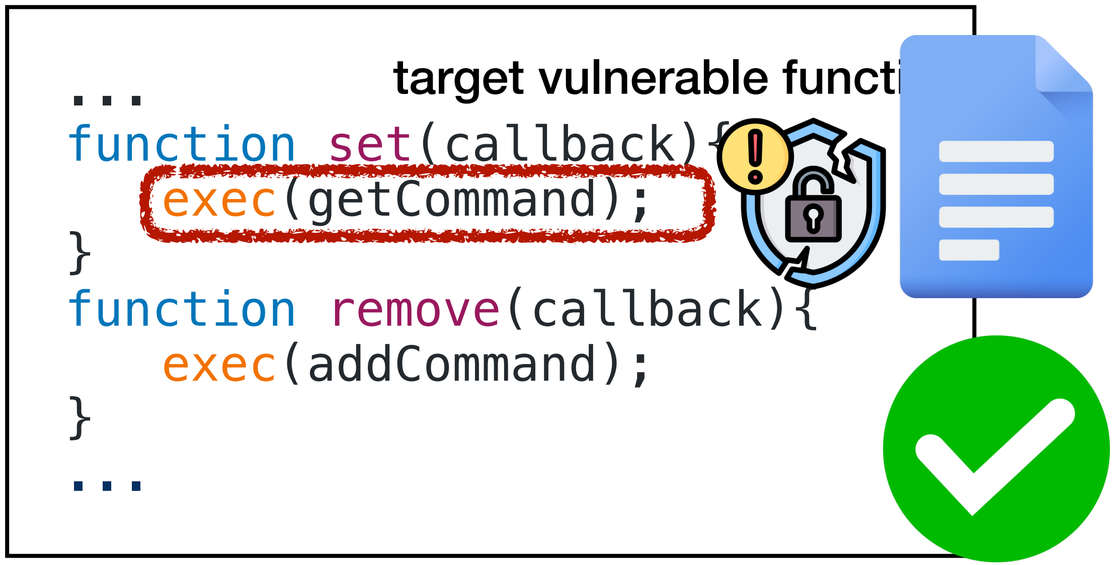
CVE®

Command-Injection in function ‘set’

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute



PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Expected
triggering the targeted vulnerability

CVE®

Command-Injection in function ‘set’

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute

```
...
function set(callback){
    exec(getCommand);
}
function remove(callback){
    exec(addCommand);
}
...
```

target vulnerable functi

PLRG

Problem of Prior Work

- **Check** the **expected outcome** **after execution**
- Treat the library code as a **blackbox**

Mismatched PoC

CVE

Command-Injection in function 'set'

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute

target vulnerable functi

```
...
function set(callback){
    exec(getCommand);
}
function remove(callback){
    exec(addCommand);
}
...
```

PLRG

Approach

Expected
alfred-workflow/index.js line 270

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.set(‘ "; touch a #');
```

Proof-of-Concept

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

Expected
alfred-workflow/index.js line 270

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.set(‘ "; touch a #');
```

Proof-of-Concept

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

Expected
alfred-workflow/index.js line 270

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.set(‘ "; touch a #');
```

Proof-of-Concept

target vulnerable function

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

Expected
alfred-workflow/index.js line 270

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.set(‘ "; touch a #');
```

Proof-of-Concept

execute

target vulnerable function

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

Expected
alfred-workflow/index.js line 270

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.set(‘ "; touch a #');
```

Proof-of-Concept

execute

target vulnerable function

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

Expected
alfred-workflow/index.js line 270

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

trigger the **vulnerability** in
alfred-workflow/index.js line 270

```
const af = require("alfred-workflow");
af.set(‘ "; touch a #');
```

Proof-of-Concept

execute

target vulnerable function

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.set(‘ "; touch a #');
```

Proof-of-Concept

execute

Expected
alfred-workflow/index.js line 270

trigger the **vulnerability** in
alfred-workflow/index.js line 270

target vulnerable function

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

Expected
alfred-workflow/index.js line 270

trigger the vulnerability in
alfred-workflow/index.js line 270

```
const af = require("alfred-workflow");
af.set(‘ "; touch a #');
```

Proof-of-Concept

execute

target vulnerable function

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

Expected
alfred-workflow/index.js line 270

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

Expected
alfred-workflow/index.js line 270

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

target vulnerable function

PLRG

Approach

Expected
alfred-workflow/index.js line 270

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute

target vulnerable function

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

Expected
alfred-workflow/index.js line 270

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute

target vulnerable function

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

Expected
alfred-workflow/index.js line 270

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

trigger the **vulnerability** in
alfred-workflow/index.js line 273

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute

target vulnerable function

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

PLRG

Approach

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute

Expected
alfred-workflow/index.js line 270

≠

trigger the **vulnerability** in
alfred-workflow/index.js line 273

target vulnerable function

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

Approach

- Instrument **vulnerable functions** to **detect triggering**
- Monitor triggering at **runtime**

```
const af = require("alfred-workflow");
af.remove(' "; touch a #');
```

Proof-of-Concept

execute

Expected
alfred-workflow/index.js line 270

≠

trigger the **vulnerability** in
alfred-workflow/index.js line 273

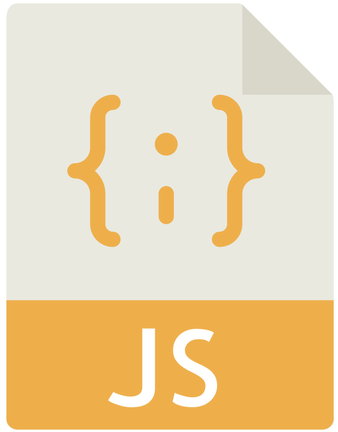
target vulnerable function

```
268 ...
269 function set(callback){
270   exec(getCommand);
271 }
272 function remove(callback){
273   exec(addCommand);
274 }
  ...
```

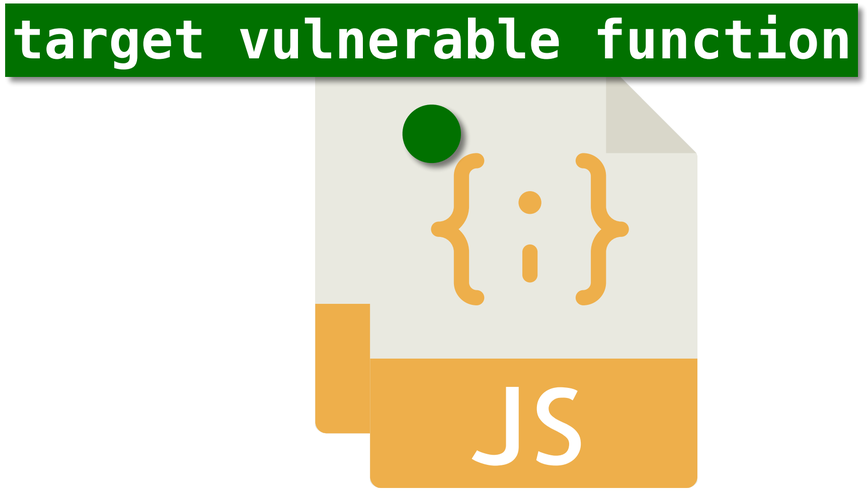
PLRG

Verification and Classification

Proof-of-Concept

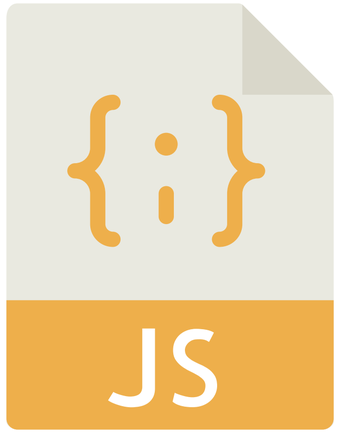


target library



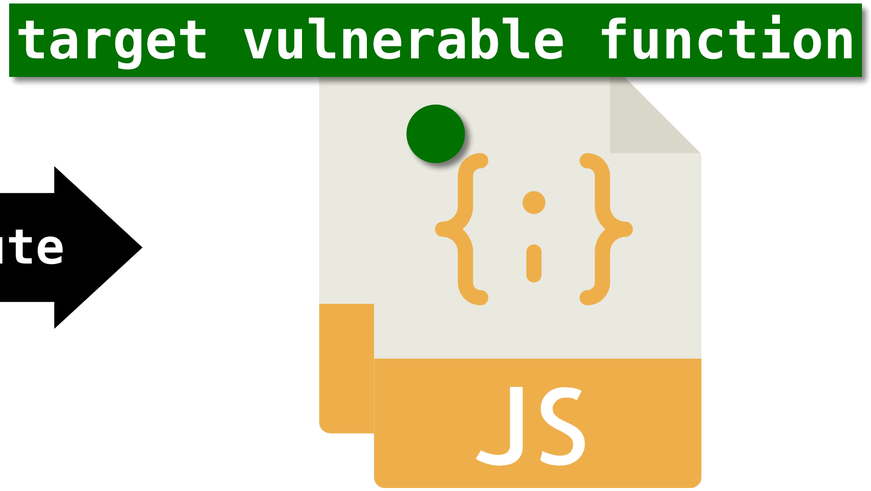
Verification and Classification

Proof-of-Concept

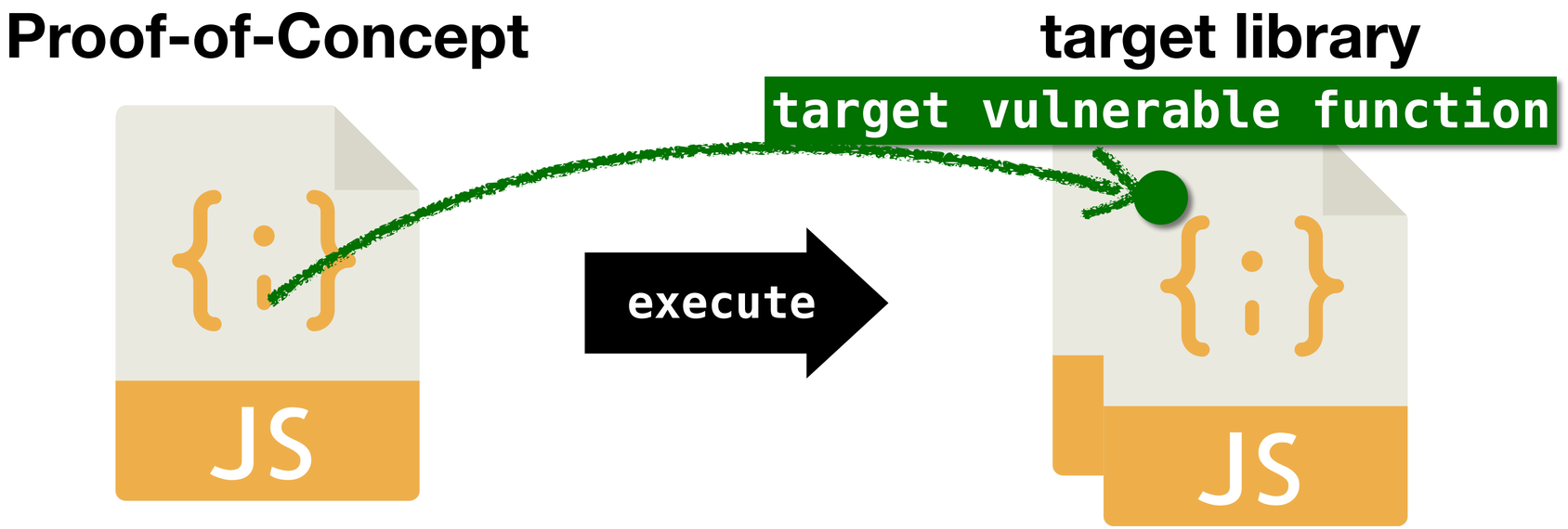


execute

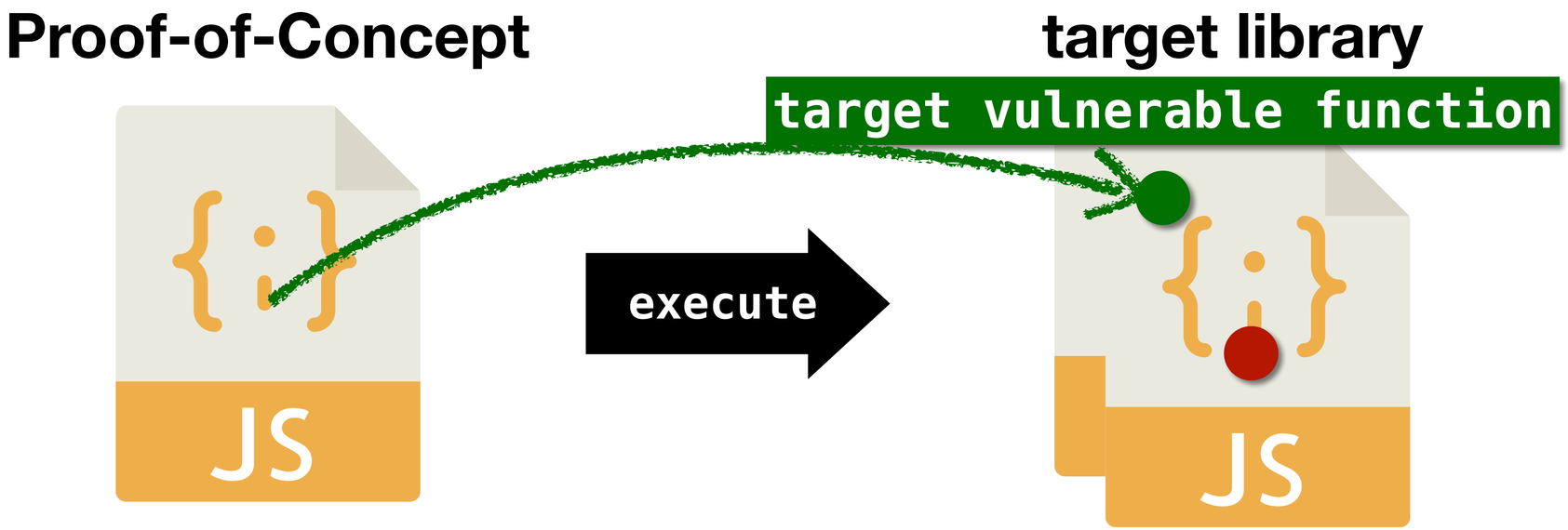
target library



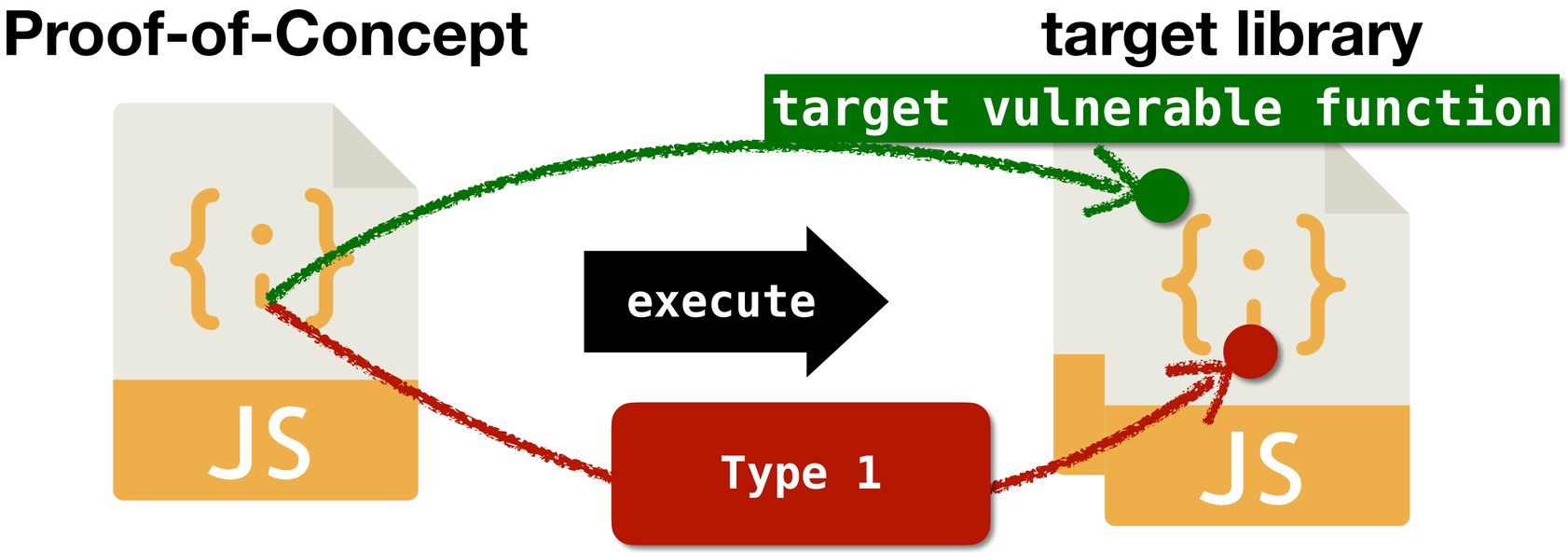
Verification and Classification



Verification and Classification

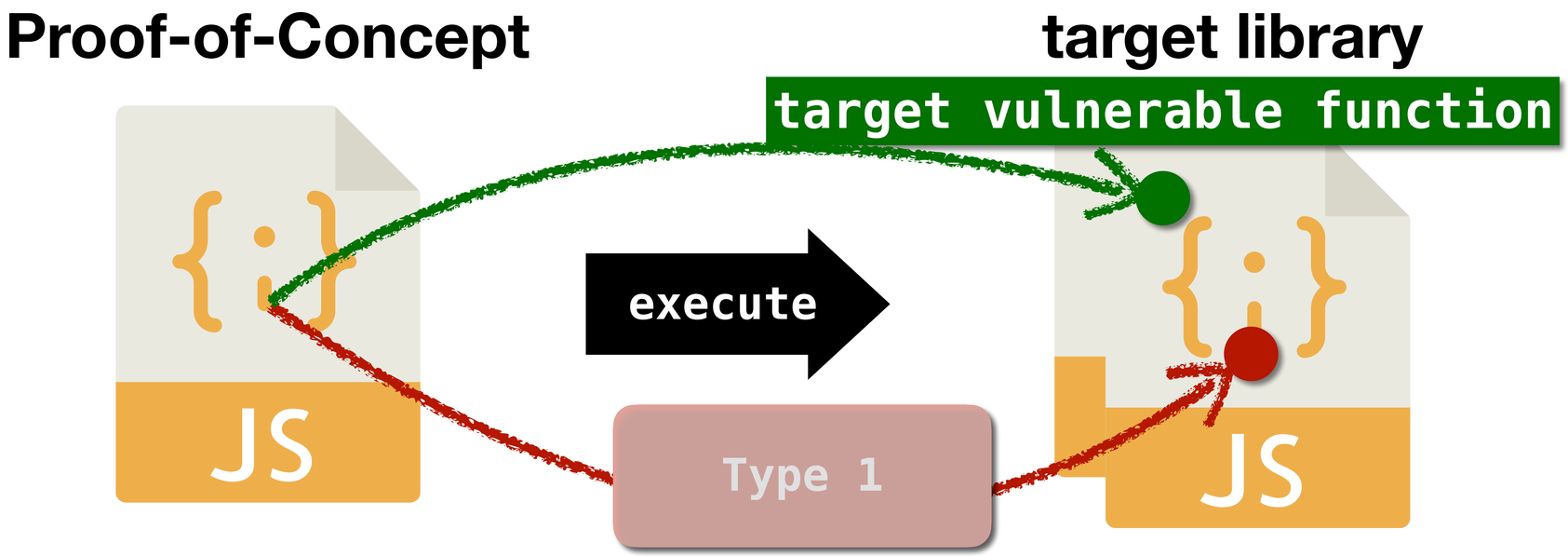


Verification and Classification



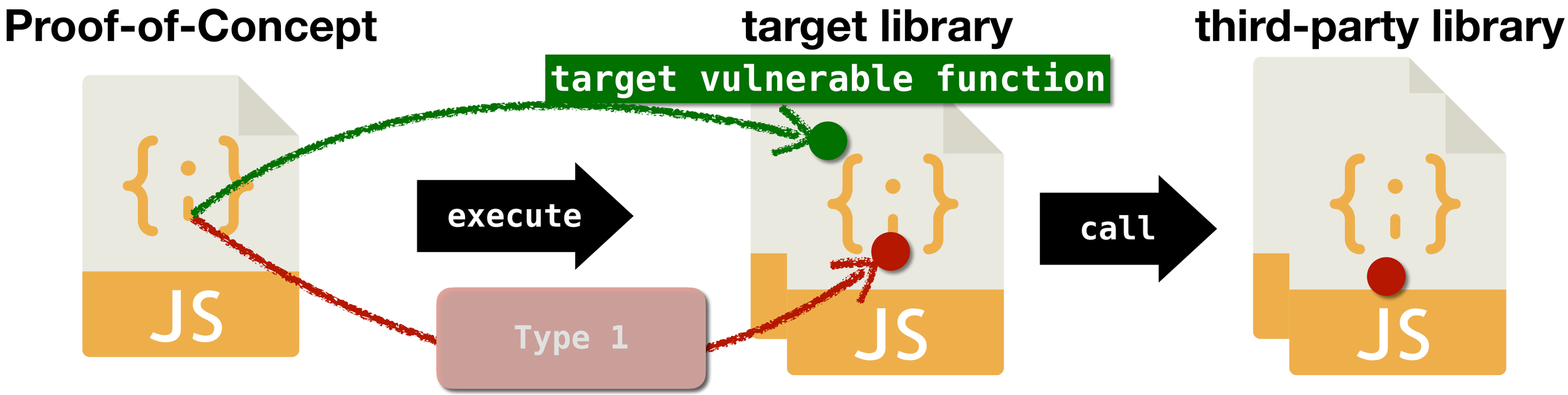
PLRG

Verification and Classification

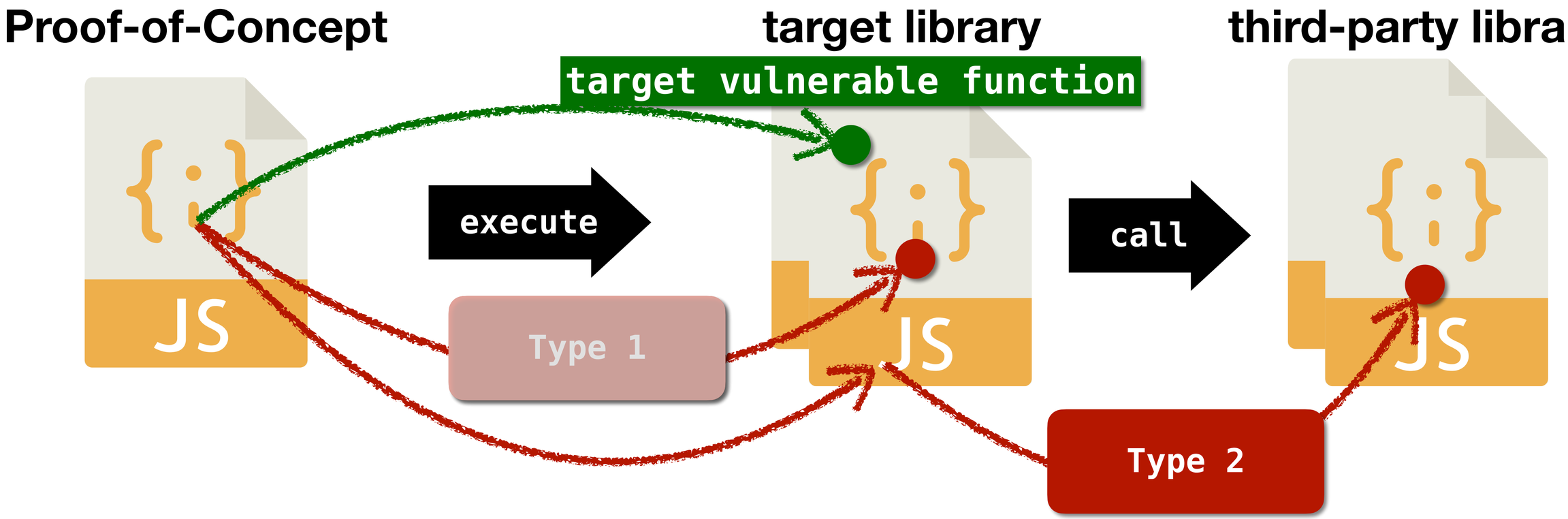


PLRG

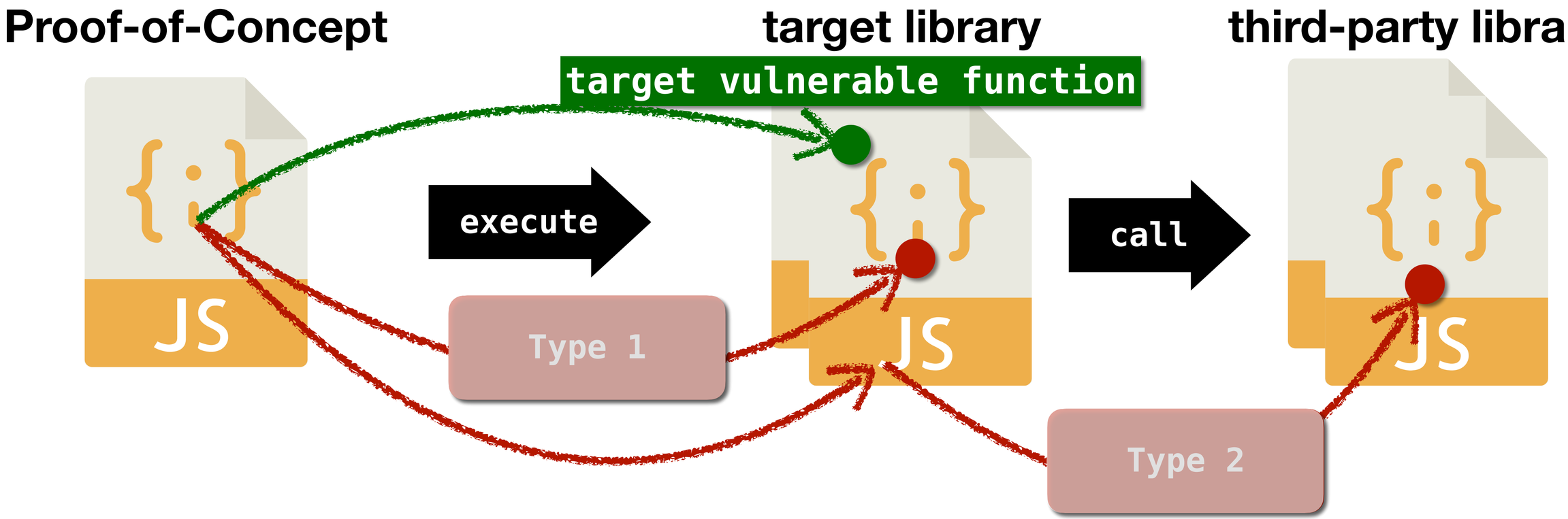
Verification and Classification



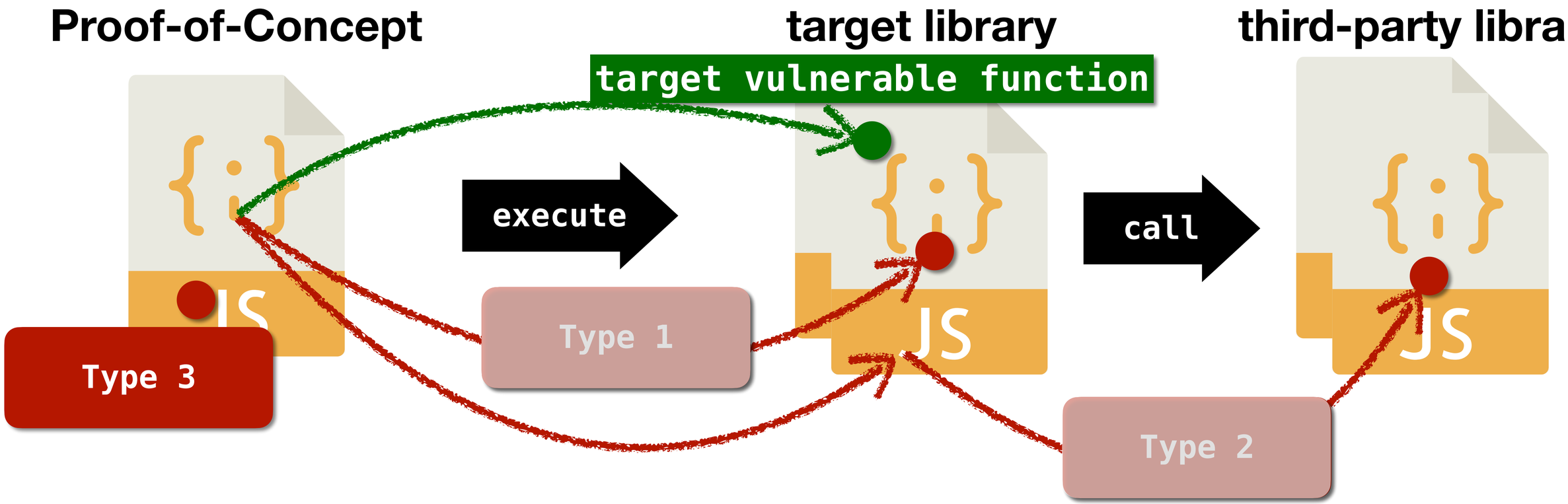
Verification and Classification



Verification and Classification



Verification and Classification



Verification and Classification

- **Type 1)** A **different sink** within the **same library**

Expected
jQuery/index.js line 306

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

jQuery


```
203 ...
204 exec()
...
305 ...
306 exec()
```

PLRG

Verification and Classification

- **Type 1)** A **different sink** within the **same library**

Expected
jQuery/index.js line 306

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

PLRG

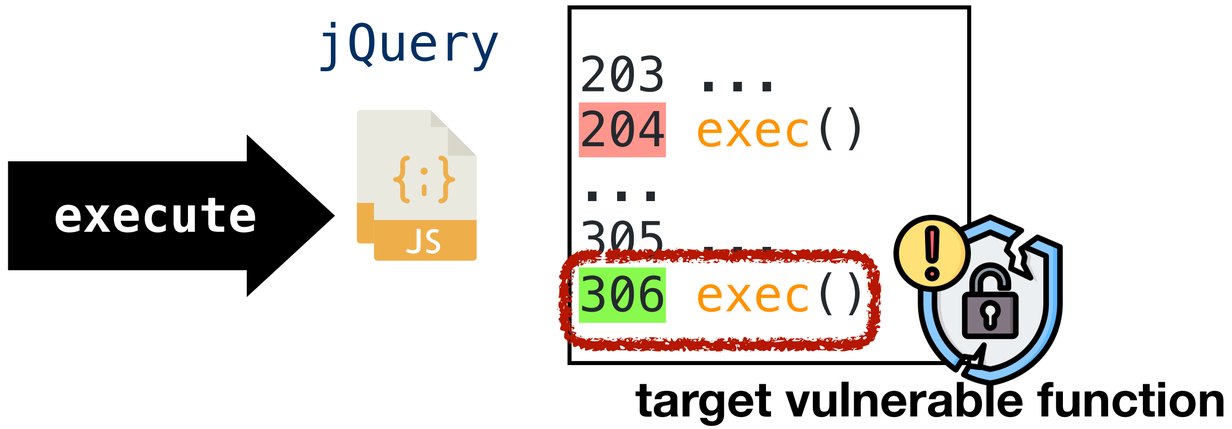
Verification and Classification

- **Type 1)** A **different sink** within the **same library**

Expected
jQuery/index.js line 306

```
const jq = require("jQuery");
jq('touch a;');
```

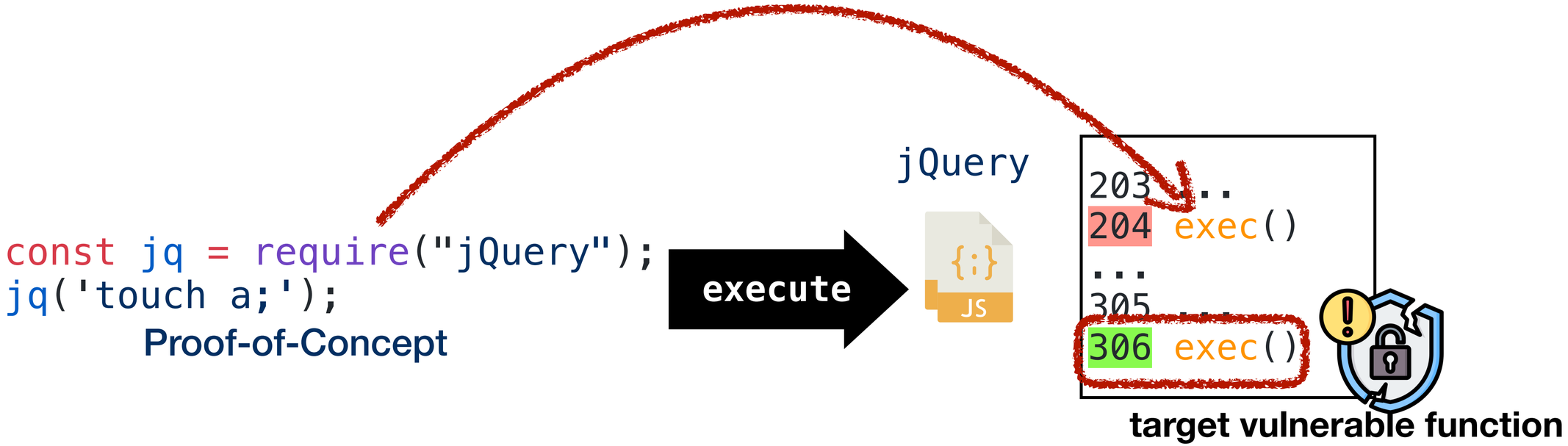
Proof-of-Concept



Verification and Classification

- **Type 1)** A **different sink** within the **same library**

Expected
jQuery/index.js line 306



Verification and Classification

- **Type 1)** A **different sink** within the **same library**

Expected
jQuery/index.js line 306

trigger the **vulnerability** in jQuery/index.js line 204

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

execute

jQuery


```
203 ...
204 exec()
...
305 ...
306 exec()
```

target vulnerable function

Verification and Classification

- **Type 2)** A **different sink** in a **third-party library**

Expected
jQuery/index.js line 306

```
const jq = require("jQuery");
jq('touch a;');
```

“touch a;”

Proof-of-Concept

jQuery


```
273 ...
274 Lodash()
...
305 ...
306 exec()
```

Lodash


```
...
55 exec()
```

PLRG

Verification and Classification

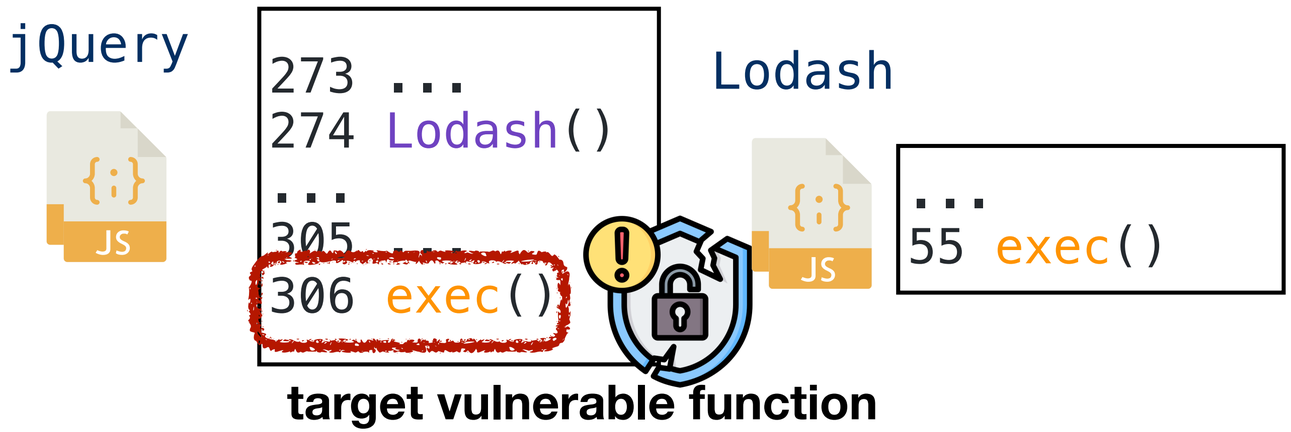
- **Type 2)** A **different sink** in a **third-party library**

Expected
jQuery/index.js line 306

```
const jq = require("jQuery");
jq('touch a;');
```

“touch a;”

Proof-of-Concept



PLRG

Verification and Classification

- **Type 2)** A **different sink** in a **third-party library**

Expected
jQuery/index.js line 306

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

“touch a;”

execute

jQuery

```
273 ...
274 Lodash()
...
305 ...
306 exec()
```

Lodash


```
...
55 exec()
```

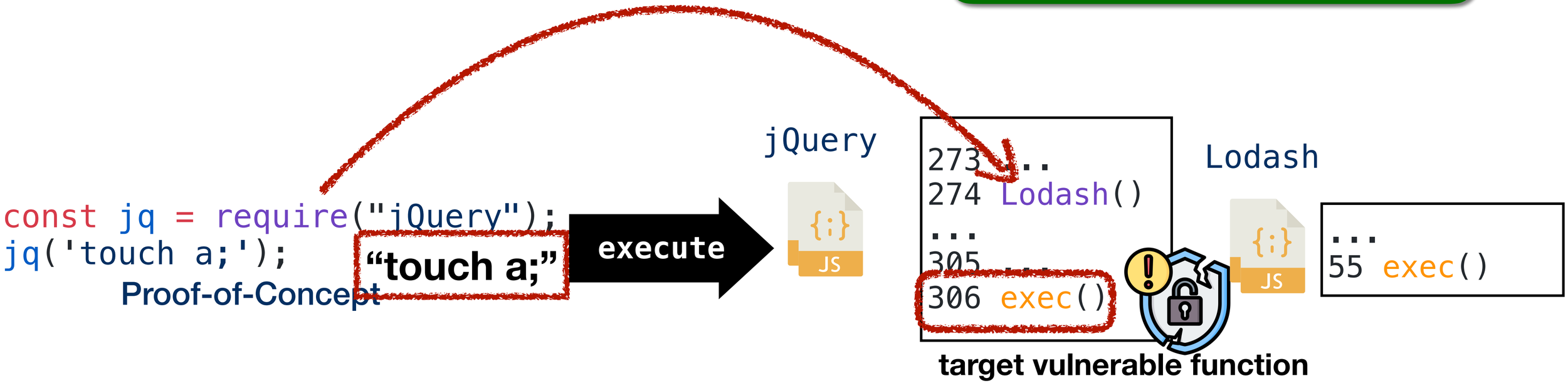
target vulnerable function

PLRG

Verification and Classification

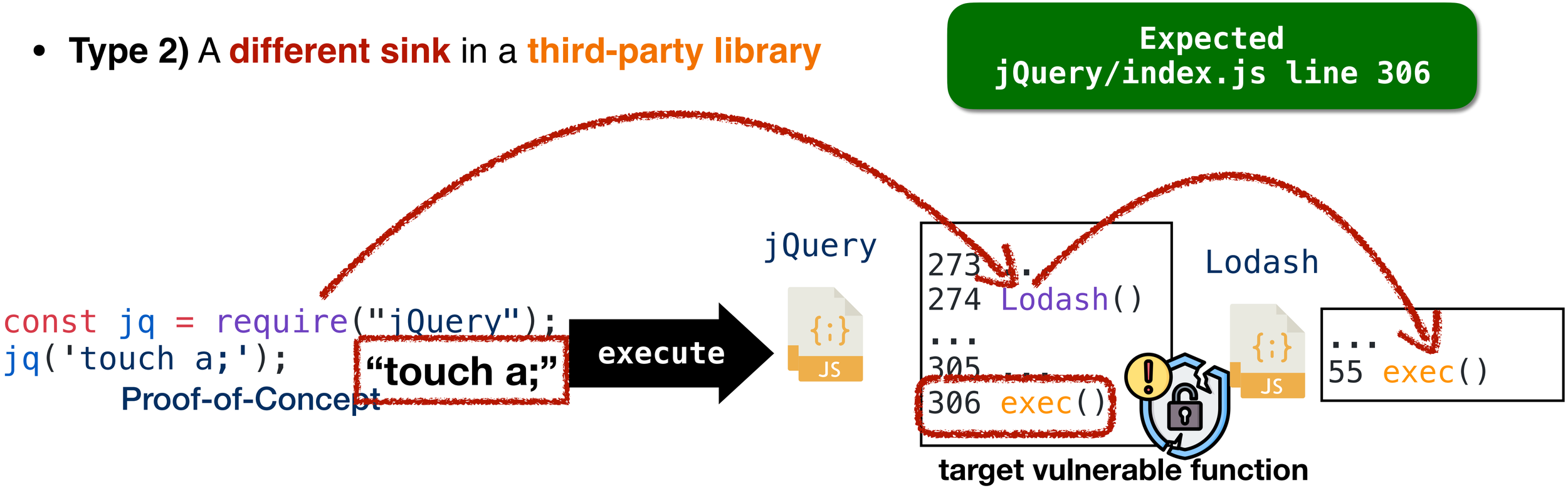
- **Type 2)** A **different sink** in a **third-party library**

Expected
jQuery/index.js line 306



Verification and Classification

- **Type 2)** A **different sink** in a **third-party library**



Verification and Classification

- **Type 2)** A **different sink** in a **third-party library**

Expected
jQuery/index.js line 306

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

execute

jQuery

JS

```
273 ...
274 Lodash()
...
305 ...
306 exec()
```

"touch a;"

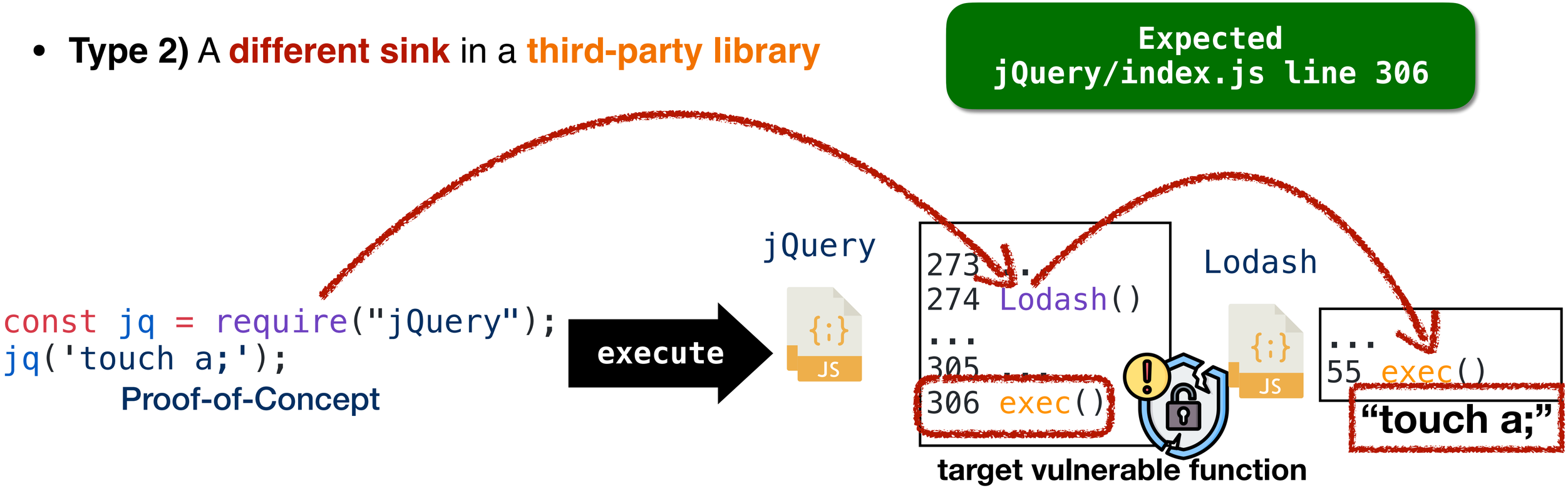
Lodash

target vulnerable function

```
...
55 exec()
```

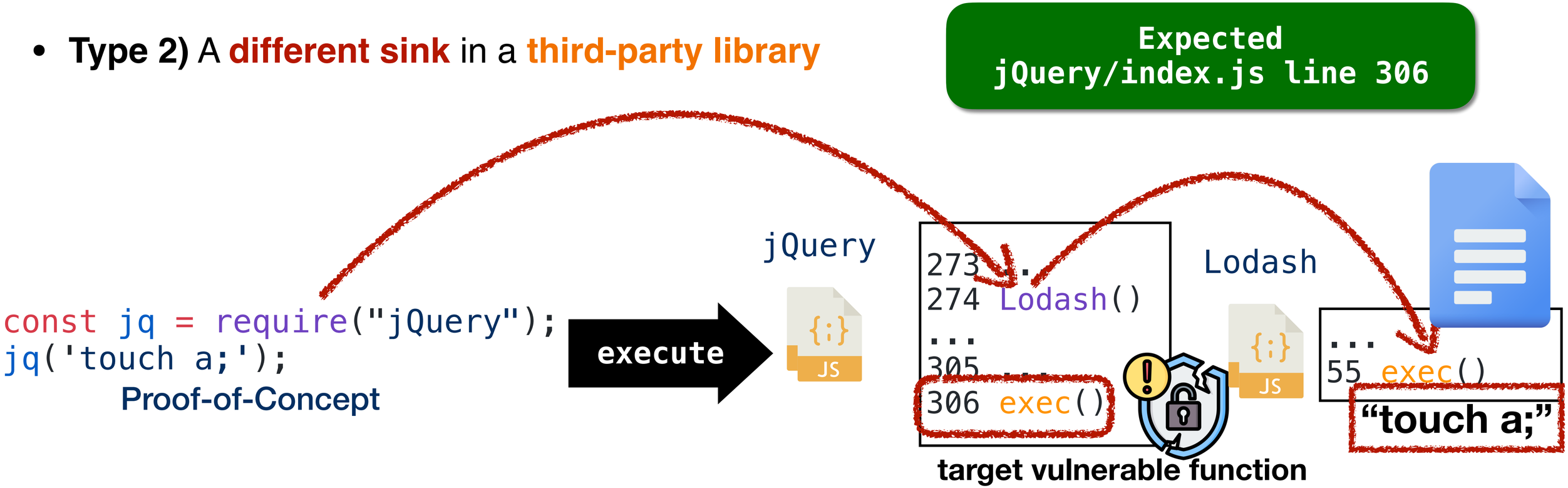
Verification and Classification

- **Type 2)** A **different sink** in a **third-party library**



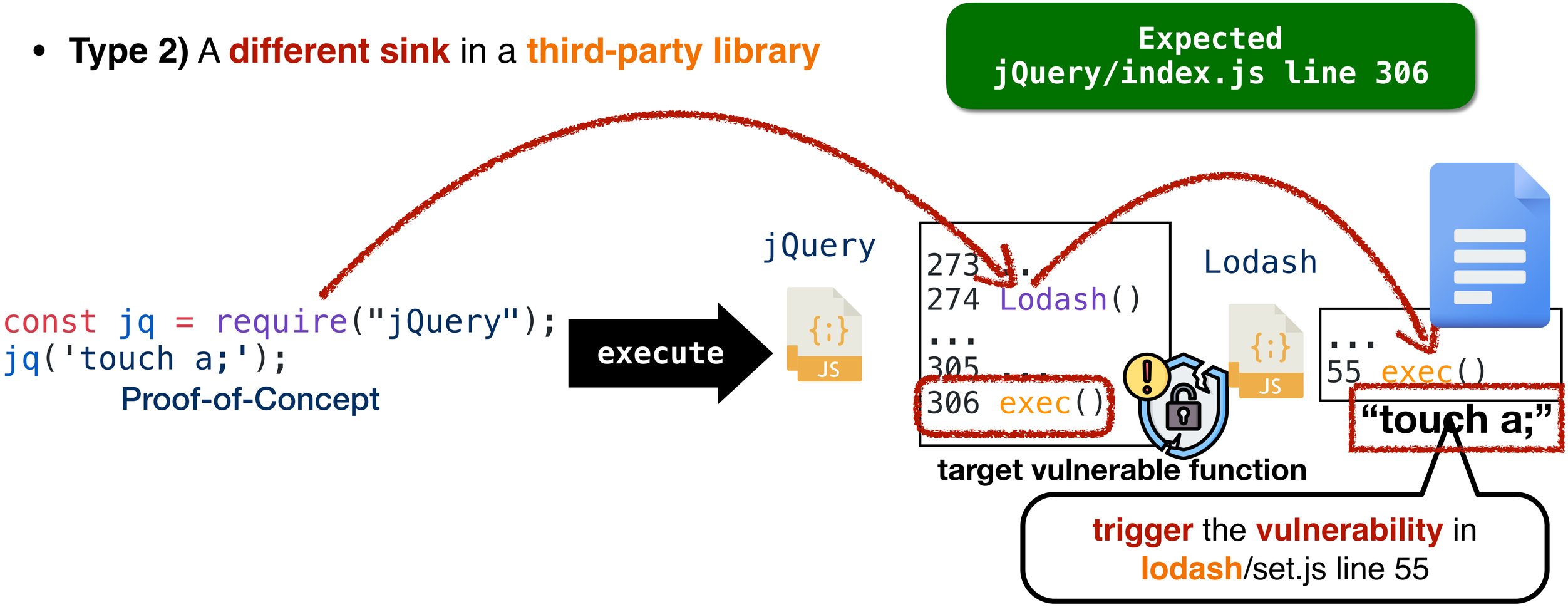
Verification and Classification

- **Type 2)** A **different sink** in a **third-party library**



Verification and Classification

- **Type 2)** A **different sink** in a **third-party library**



Verification and Classification

- **Type 3)** PoC **self-generates** expected outcome **without libraries**

Expected
jQuery/index.js line 306

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

jQuery


```
...
305 ...
306 exec()
```

PLRG

Verification and Classification

- **Type 3)** PoC **self-generates** expected outcome **without libraries**

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept

Expected
jQuery/index.js line 306

PLRG

Verification and Classification

- **Type 3)** PoC **self-generates** expected outcome **without libraries**

```
const jq = require("jQuery");
jq('touch a;');
exec('touch a;');
```

Proof-of-Concept

Expected
jQuery/index.js line 306

Verification and Classification

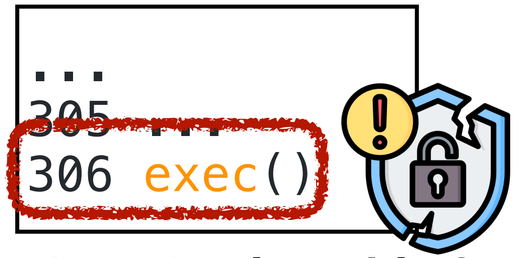
- **Type 3)** PoC **self-generates** expected outcome **without libraries**

```
const jq = require          ry");
jq('touch a;');
exec('touch a;');
```

Proof-of-Concept

Expected
jQuery/index.js line 306

jQuery



target vulnerable function

Verification and Classification

- **Type 3)** PoC **self-generates** expected outcome **without libraries**

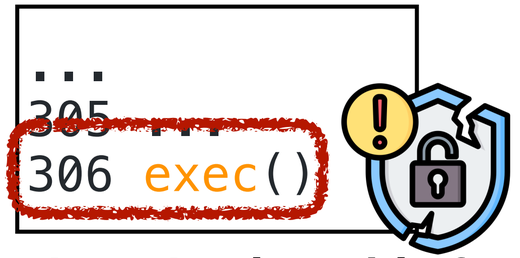
Expected
jQuery/index.js line 306

```
const jq = require          ry");
jq('touch a;');
exec('touch a;');
```

Proof-of-Concept

trigger the **vulnerability** in **PoC**

jQuery



target vulnerable function

Summary

target vulnerability
jQuery/index.js line 274

Type 1) A **different sink** within the **same library**

- ex) jQuery/index.js line 112, jQuery/util.js line 78

Type 2) A **different sink** in a **third-party library**

- ex) **lodash**/set.js line 55

Type 3) PoC **self-generates** expected outcome **without libraries**

Type 4) No expected outcome, simple mistake

PLRG

Evaluation

- Prototype-Pollution Exploits

	Type 1	Type 2	Type 3	Type 4	Mismatched	Total
SecBench.js	0	4	4	3	11	194
Explode.js	18	0	0	0	18	83
PoCGen	14	1	2	2	19	148

- Assume SecBench.js, a manual dataset, as **ground truth**
- **11.29%** of exploits fail to match with their **target sink**

Evaluation

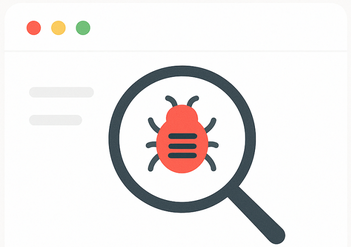
- Prototype-Pollution Exploits

	Type 1	Type 2	Type 3	Type 4	Mismatched	Total
SecBench.js	0	4	4	3	11	194
Explode.js	18	0	0	0	18	83
PoCGen	14	1	2	2	19	148

- Assume SecBench.js, a manual dataset, as **ground truth**
- **11.29%** of exploits fail to match with their **target sink**

PLRG

Future Work



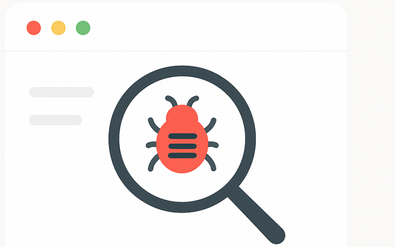
Vulnerable Library Detected

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept (POC) Generation

PLRG

Future Work



Vulnerable Library Detected

```
const jq = require("jQuery");
jq('touch a;');
```

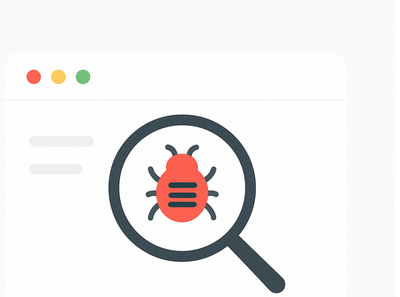
Proof-of-Concept (POC) Generation

This work

Verification and Classification of Exploits

PLRG

Future Work



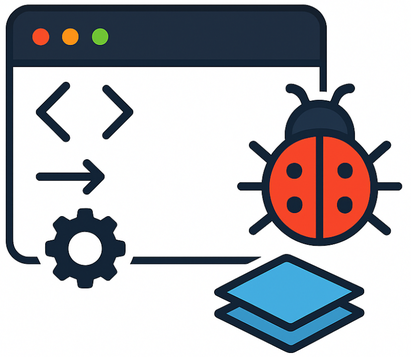
Vulnerable Library Detected

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept (POC) Generation

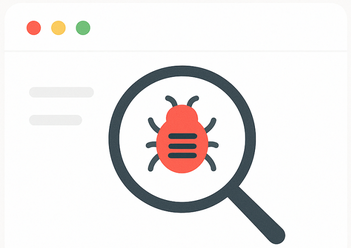
Verification and Classification of Exploits

This work



Real-world Web Vulnerabilitity Detection

PLRG



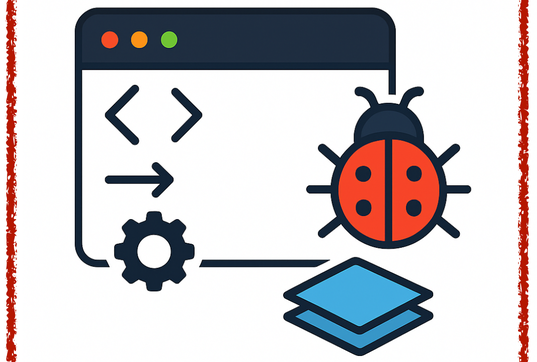
Vulnerable Library Detected

```
const jq = require("jQuery");
jq('touch a;');
```

Proof-of-Concept (POC) Generation

This work

Verification and Classification of Exploits



Real-world Web Vulnerabilitity Detection

KOREA UNIVERSITY

PLRG

ASE 2025