Freezing the Web: A Study of ReDoS Vulnerabilities in JavaScript-based Web Servers

Cristian-Alexandru Staicu Michael Pradel

TU Darmstadt

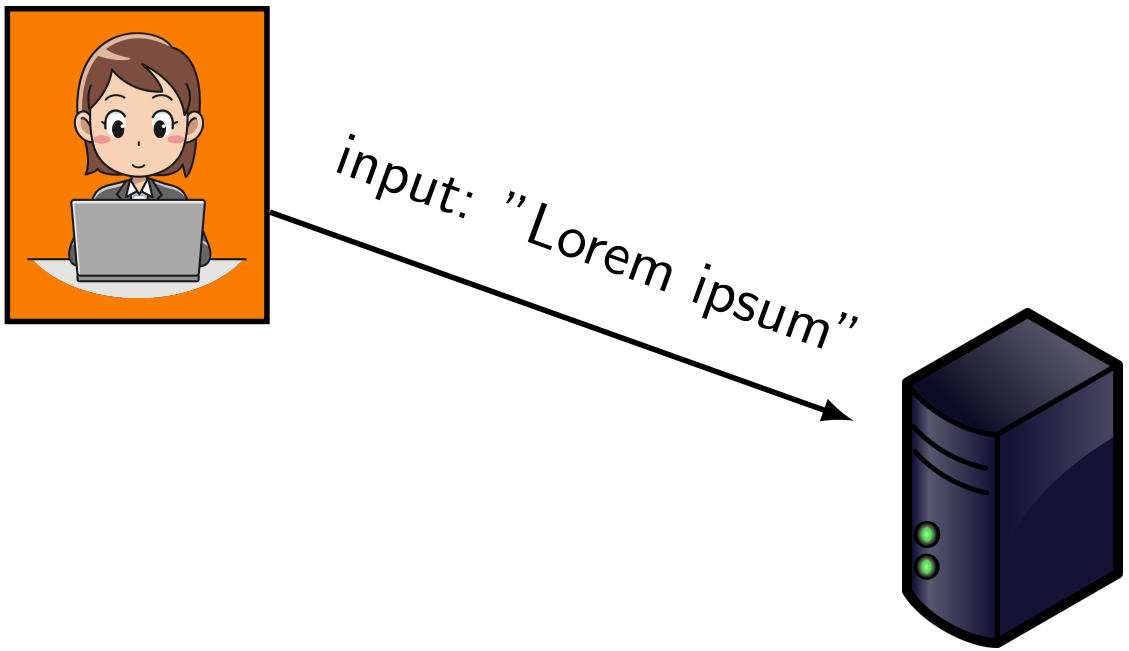
www.software-lab.org

15th August 2018

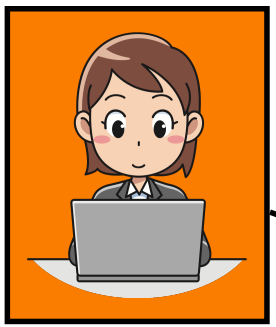
TECHNISCHE UNIVERSITÄT DARMSTADT

SOLA SoftwareLab

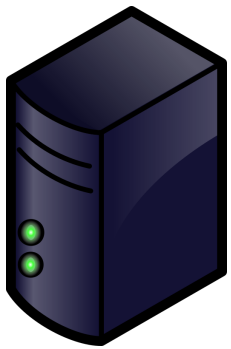
Regular Expression Denial of Service (ReDoS)



Regular Expression Denial of Service (ReDoS)

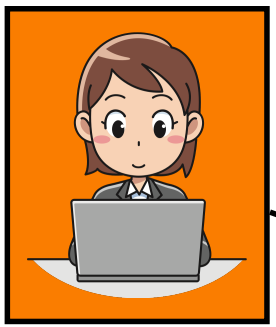


input: "Lorem ipsum"



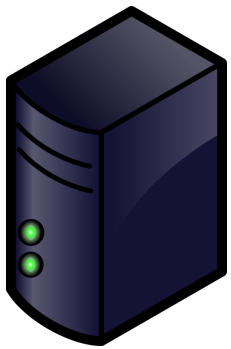
input.match(regexp);

Regular Expression Denial of Service (ReDoS)



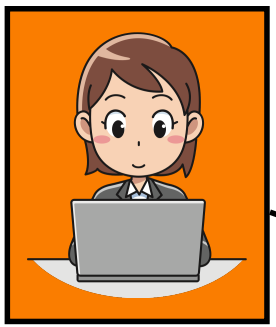
input: "Lorem ipsum"

processing time: $\mathcal{O}(1)$



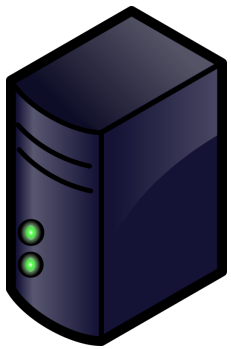
input.match(regexp);

Regular Expression Denial of Service (ReDoS)



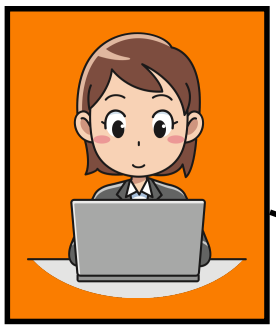
input: "Lorem ipsum"

processing time: $\mathcal{O}(1)$



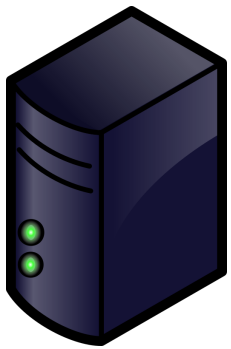
input.match(regexp);

Regular Expression Denial of Service (ReDoS)



input: "Lorem ipsum"

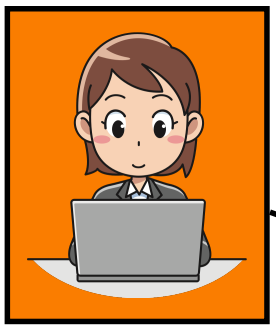
processing time: $\mathcal{O}(1)$



input.match(regexp);

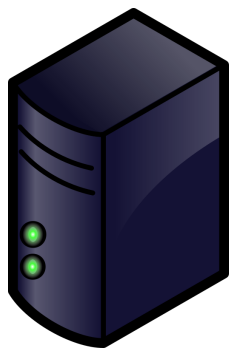
input: "ǎóÁs̩" × 1000

Regular Expression Denial of Service (ReDoS)



input: "Lorem ipsum"

processing time: $\mathcal{O}(1)$

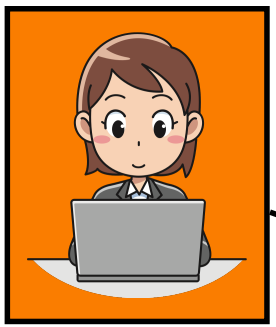


input.match(regexp);

input: "ăóÁș" x 1000

processing time: $\mathcal{O}(n^x), x > 1$

Regular Expression Denial of Service (ReDoS)



input: "Lorem ipsum"

processing time: $\mathcal{O}(1)$

input.match(regexp);

input: "ăóÁș" x 1000

processing time: $\mathcal{O}(n^x), x > 1$

ReDoS affects libraries

we identify 25 vulnerabilities in popular npm modules

This Talk

ReDoS affects libraries

we identify 25 vulnerabilities in popular npm modules

ReDoS affects websites

hundreds of live websites are vulnerable

This Talk

ReDoS affects libraries

we identify 25 vulnerabilities in popular npm modules

ReDoS affects websites

hundreds of live websites are vulnerable

Novel methodology

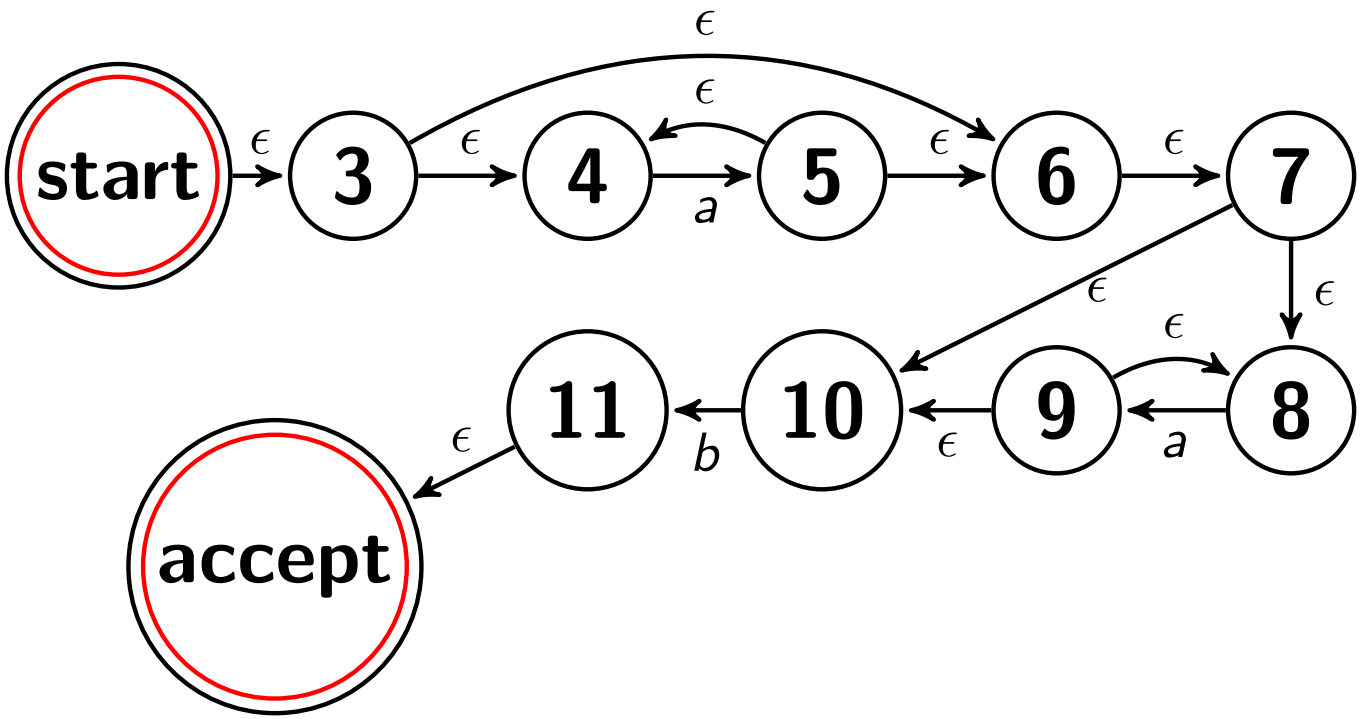
library vulnerability $\rightarrow$ website vulnerability

Backtracking-based Matching

```
var regEx = /^a*a*b$/;
```

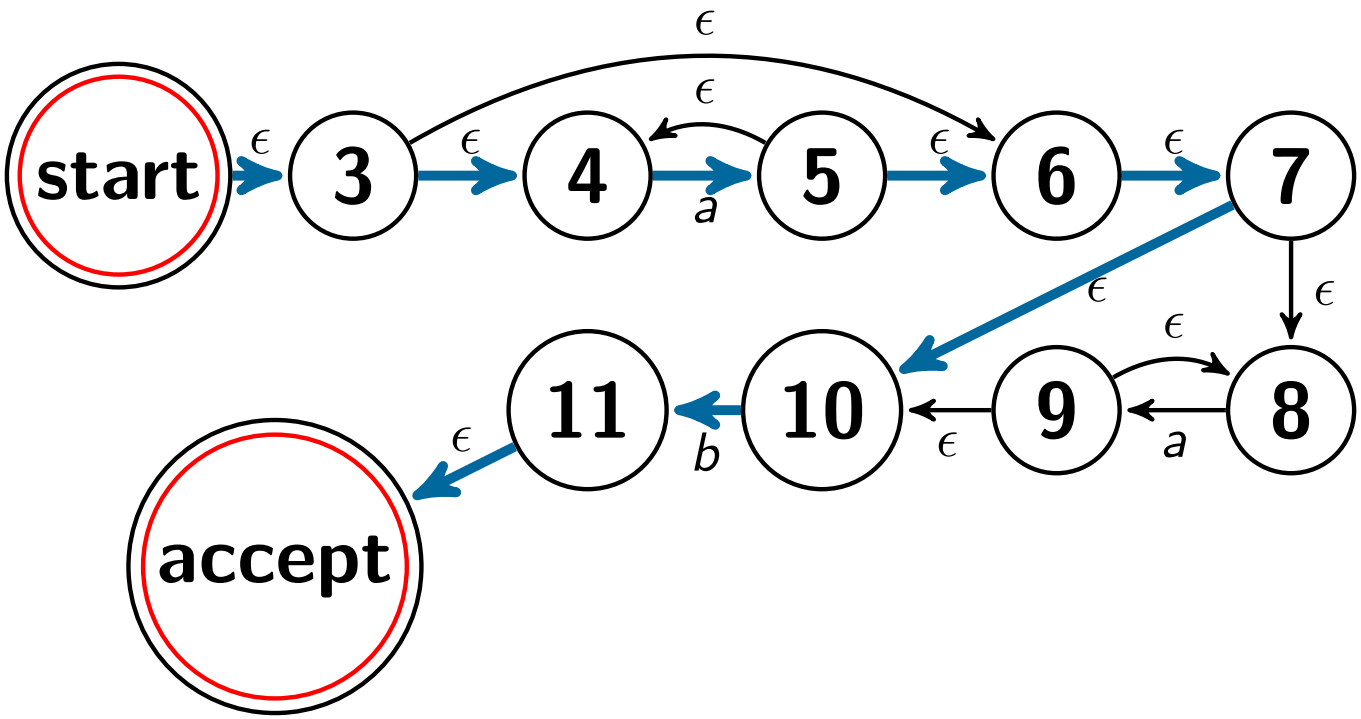
Backtracking-based Matching

```
var regEx = /^a*a*b$/;
```



Backtracking-based Matching

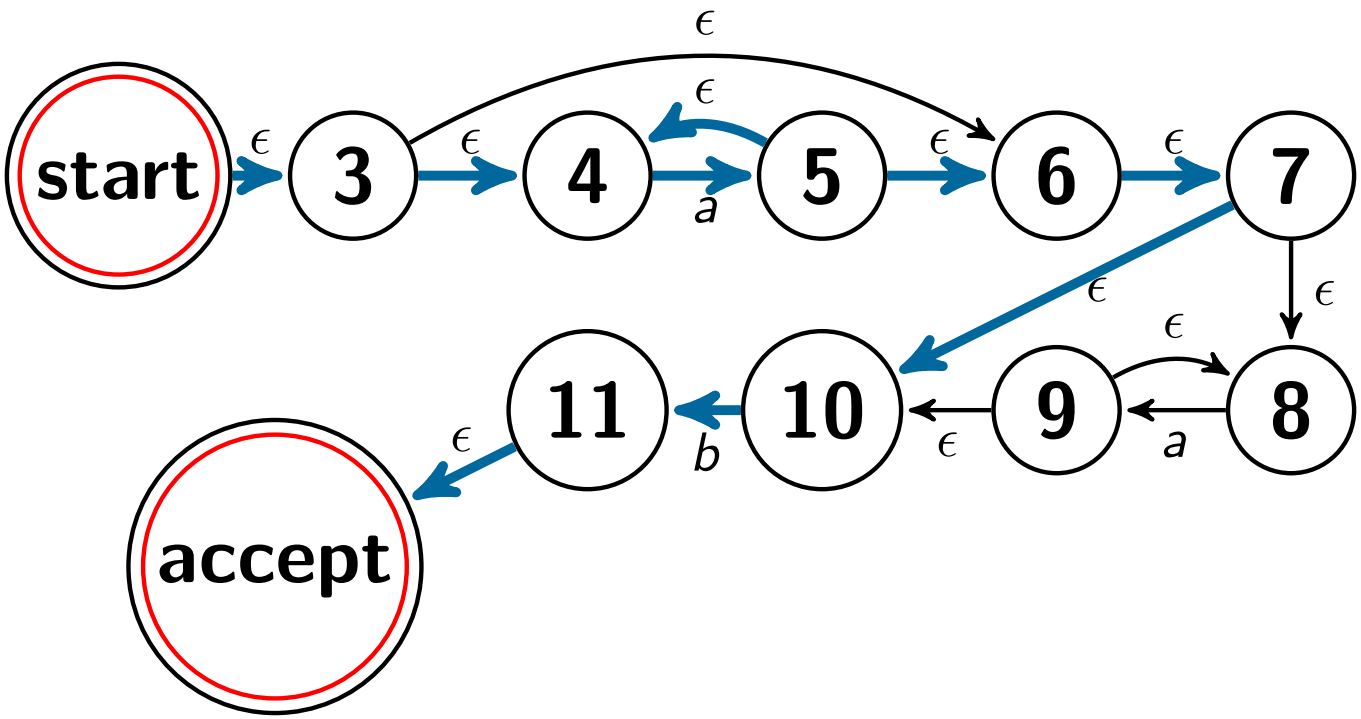
```
var regEx = /^a*a*b$/;
```



input: "ab"

Backtracking-based Matching

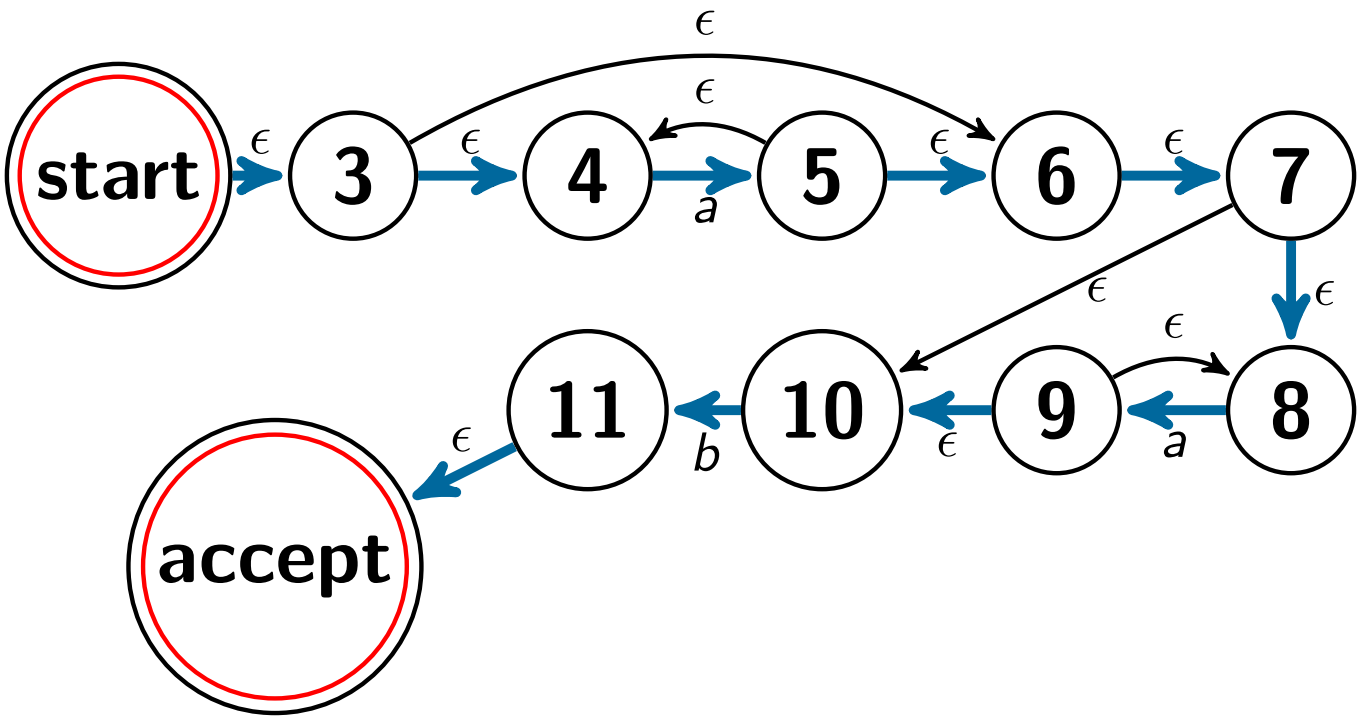
```
var regEx = /^a*a*b$/;
```



input: "aab"

Backtracking-based Matching

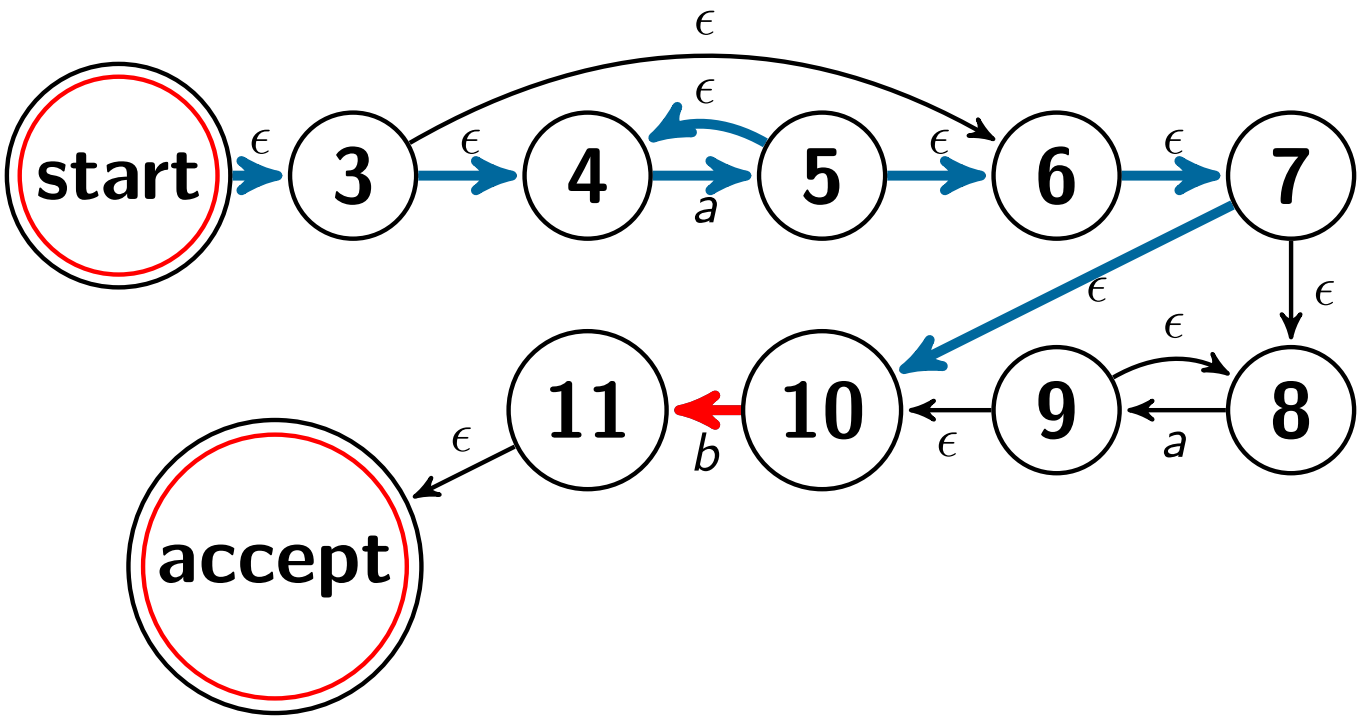
```
var regEx = /^a*a*b$/;
```



input: "aab"

Backtracking-based Matching

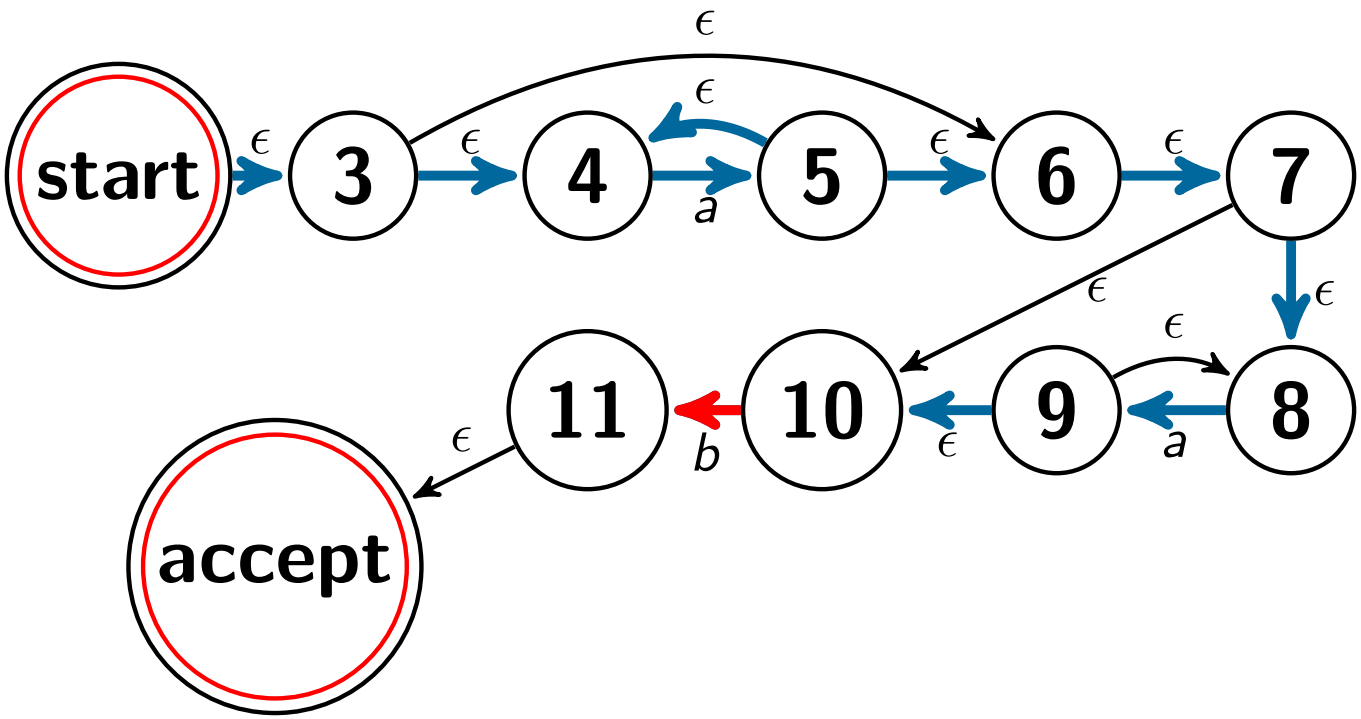
```
var regEx = /^a*a*b$/;
```



input: "aaaaaaaaaaaaaaaaaaaa"

Backtracking-based Matching

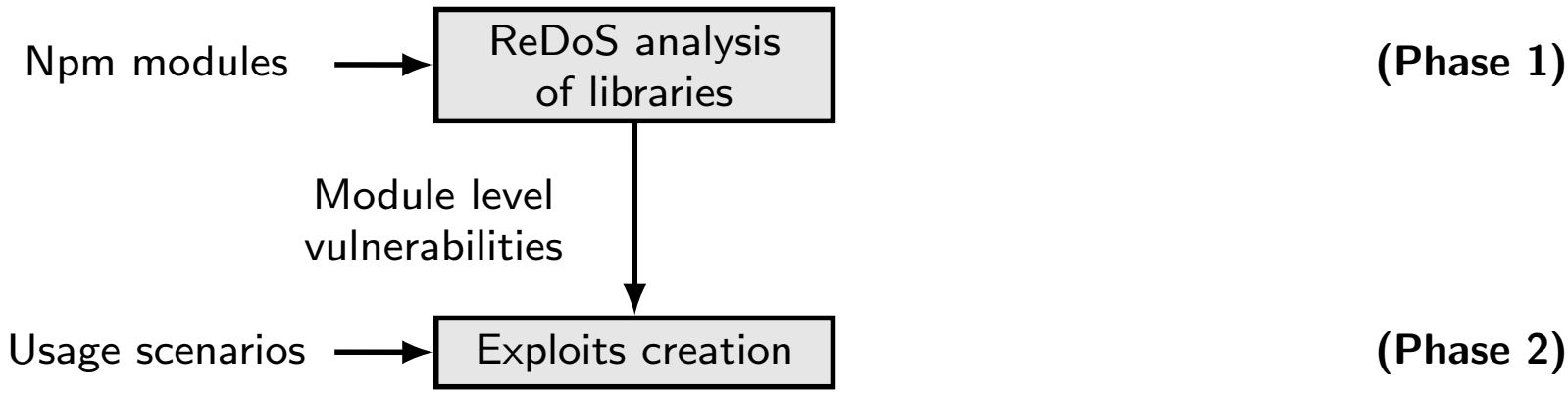
```
var regEx = /^a*a*b$/;
```



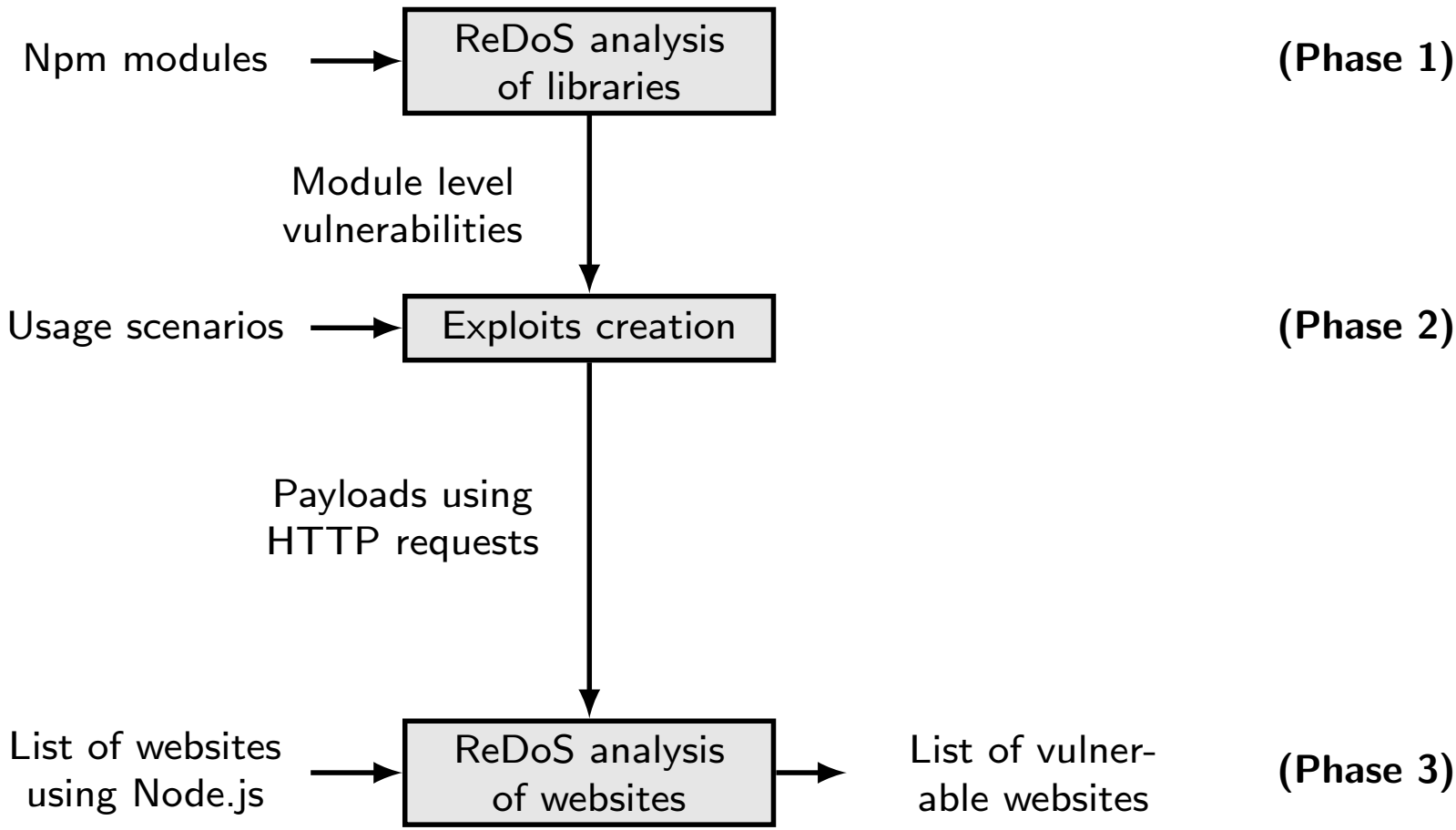
input: "aaaaaaaaaaaaaaaaaaaa"

Overview

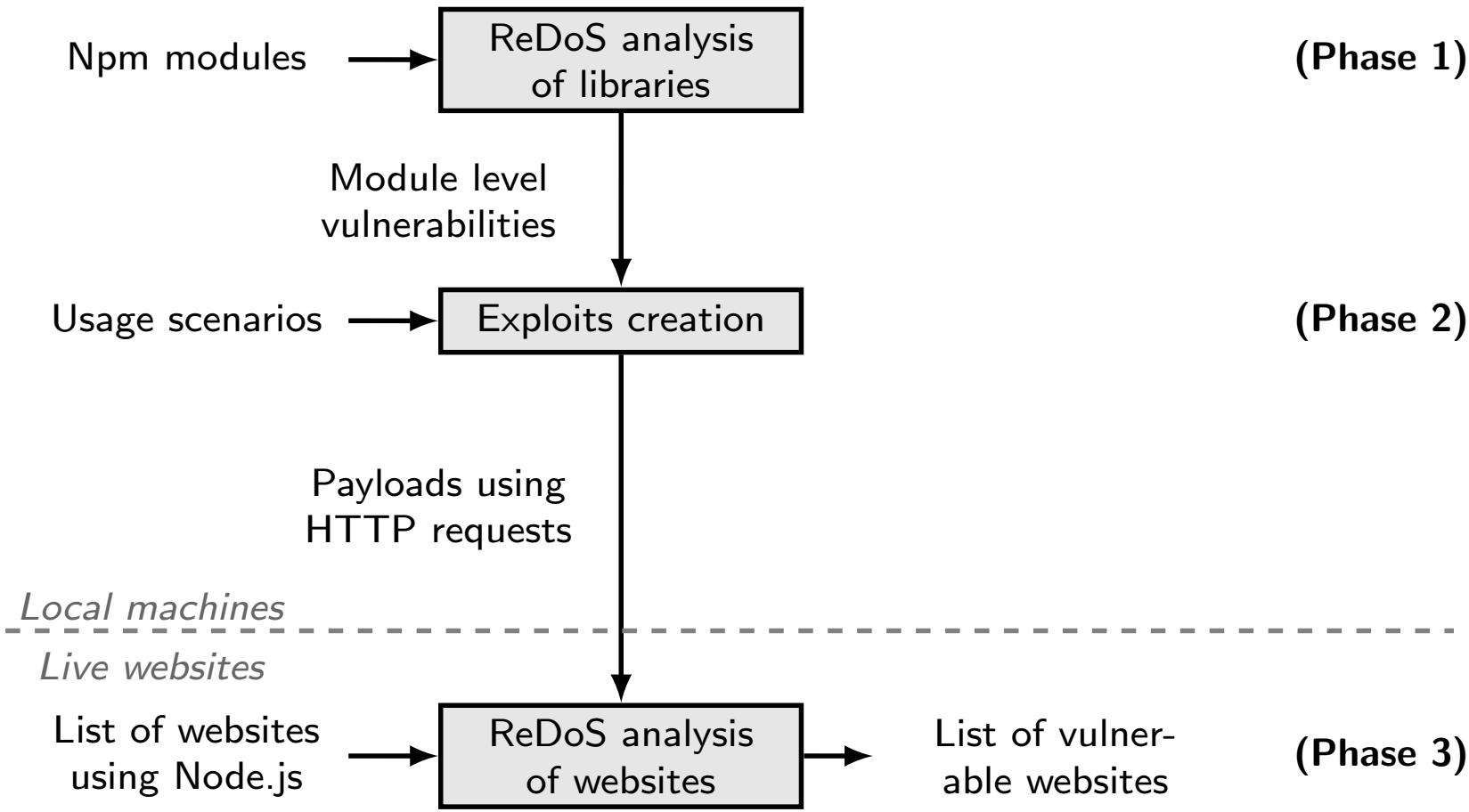
Overview


Npm modules
ReDoS analysis of libraries
(Phase 1)
Module level vulnerabilities
Usage scenarios
Exploits creation
(Phase 2)

Overview


Npm modules
ReDoS analysis of libraries
(Phase 1)
Module level vulnerabilities
Usage scenarios
Exploits creation
(Phase 2)
Payloads using HTTP requests
List of websites using Node.js
ReDoS analysis of websites
List of vulnerable websites
(Phase 3)

Overview


Npm modules
ReDoS analysis of libraries
(Phase 1)
Module level vulnerabilities
Usage scenarios
Exploits creation
(Phase 2)
Payloads using HTTP requests
Local machines
Live websites
List of websites using Node.js
ReDoS analysis of websites
List of vulner-able websites
(Phase 3)

Setup

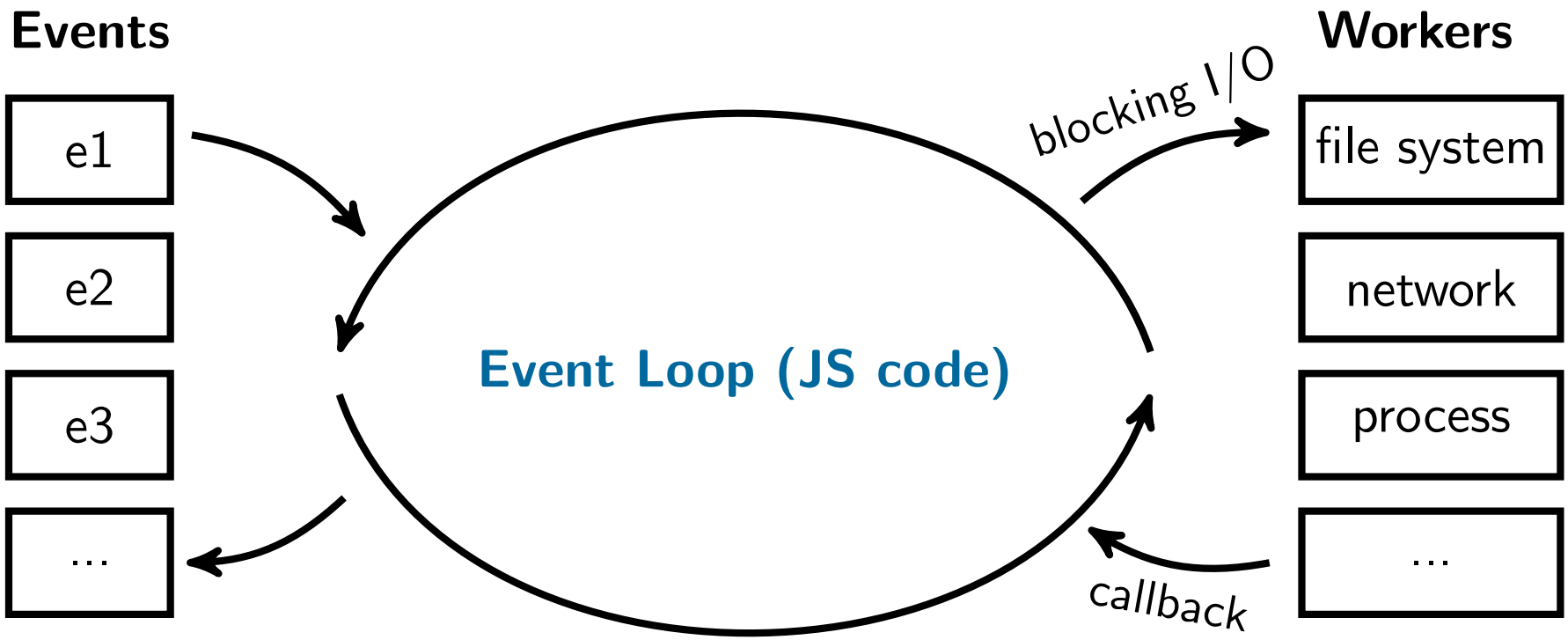
measure in single instance setup

manually analyze popular packages

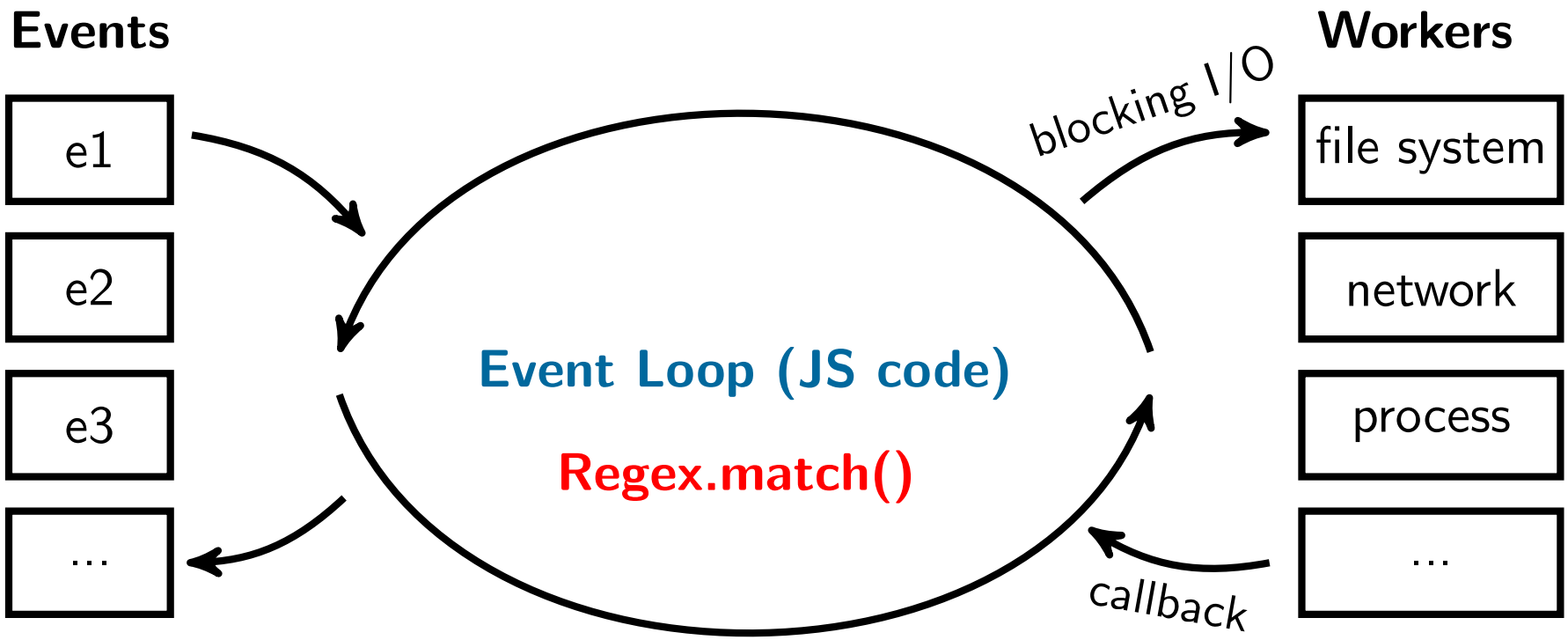
analyze 2,800 websites from Top 1 million

fifth most-dependent upon npm package

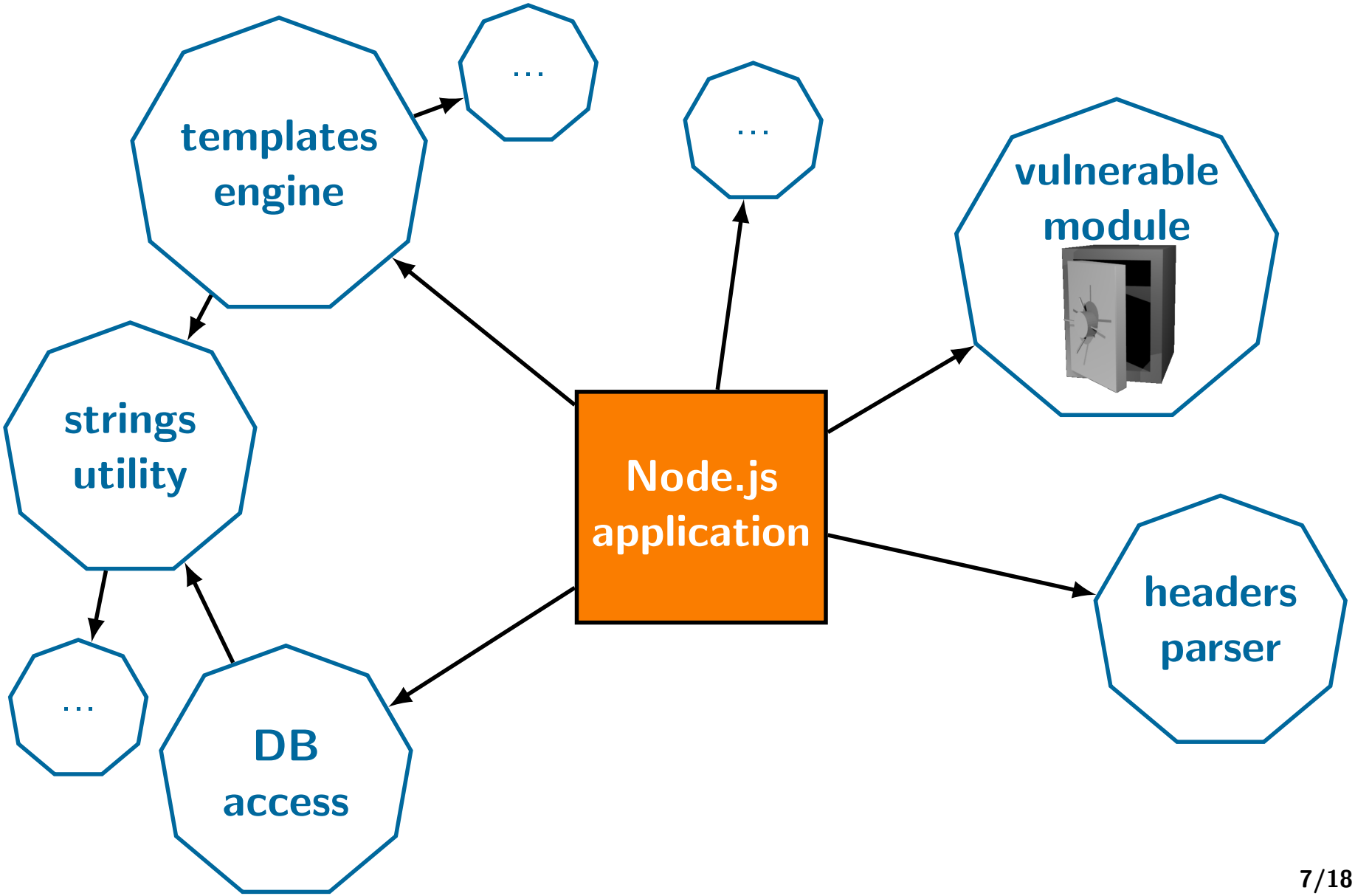
Node.js Particularities



Node.js Particularities



Node.js Particularities (2)



Ethical Considerations

Few payloads

80 requests in total

Ethical Considerations

Few payloads

80 requests in total

Iterative probing

most websites use redundancy

Ethical Considerations

Few payloads

80 requests in total

Iterative probing

most websites use redundancy

Safety mechanism

stop after timeout or error

Ethical Considerations

Few payloads

80 requests in total

Iterative probing

most websites use redundancy

Safety mechanism

stop after timeout or error

Vulnerabilities disclosure

the majority of them have been fixed

Phase 1: Npm Analysis

Criterion for vulnerable libraries

We consider a module to be vulnerable iff we find an input that

- is at most 80,000 characters long,
- whose **matching time takes more than 5 seconds.**

Phase 1: Npm Analysis

Criterion for vulnerable libraries

We consider a module to be vulnerable iff we find an input that

- is at most 80,000 characters long,
- whose **matching time takes more than 5 seconds.**

- Manual analysis of regular expressions and information flow

Phase 1: Npm Analysis

Criterion for vulnerable libraries

We consider a module to be vulnerable iff we find an input that

- is at most 80,000 characters long,
- whose **matching time takes more than 5 seconds.**

- Manual analysis of regular expressions and information flow
- Manually written exploits

Phase 1: Vulnerable Regular Expressions

- 25 ReDoS vulnerabilities

Phase 1: Vulnerable Regular Expressions

- 25 ReDoS vulnerabilities
- 13 advisories

Phase 1: Vulnerable Regular Expressions

- 25 ReDoS vulnerabilities
- 13 advisories
- One bug bounty

Phase 1: Vulnerable Regular Expressions

- 25 ReDoS vulnerabilities
- 13 advisories
- One bug bounty
- **Example 1**: content

```
/^([^\/]+\/[^\s;]+)(?:(?:\s*;\s*boundary=(?:"([^"
    ]+)"|([^;"]+)))|(?:\s*;\s*[^=]+=(?:(?:"(?:[^"
    ]+)")|(?:[^;"]+))))*$/i
```

Phase 1: Vulnerable Regular Expressions

- 25 ReDoS vulnerabilities
- 13 advisories
- One bug bounty
- **Example 1**: content

```
/^([^\/]+\/[^\s;]+)(?:(?:\s*;\s*boundary=(?:"([^"
    ]+)"|([^;"]+)))|(?:\s*;\s*[^=]+=(?:(?:"(?:[^"
    ]+)")|(?:[^;"]+))))*$/i
```

- **Example 2**: ua-parser-js

```
/ip[honead]+(.*os\s([\w]+)*\slike\smac|;\sopera)/
```

Phase 2: HTTP-level Payload Creation

- Local Node.js installation

Phase 2: HTTP-level Payload Creation

- Local Node.js installation
- For each payload, create a **usage scenario**

Phase 2: HTTP-level Payload Creation

- Local Node.js installation
- For each payload, create a **usage scenario**

```
var MobileDetect = require("mobile-detect");
var headers = req.headers["user-agent"];
var md = new MobileDetect(headers);
md.phone();
```

Phase 2: HTTP-level Payload Creation

- Local Node.js installation
- For each payload, create a **usage scenario**

```
var MobileDetect = require("mobile-detect");
var headers = req.headers["user-agent"];
var md = new MobileDetect(headers);
md.phone();
```

- For each scenario, create HTTP level payloads

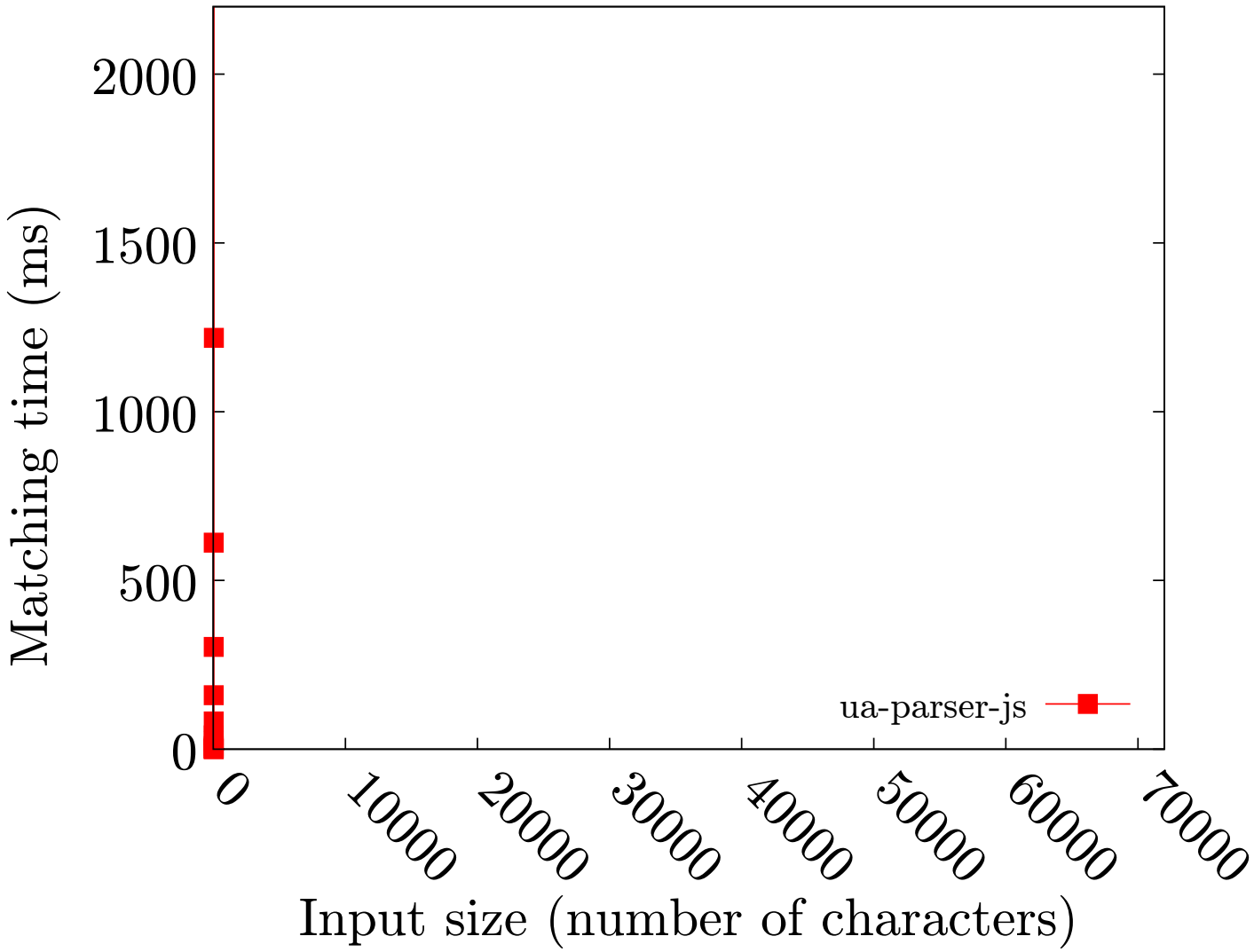
Phase 2: HTTP-level Payload Creation

- Local Node.js installation
- For each payload, create a **usage scenario**

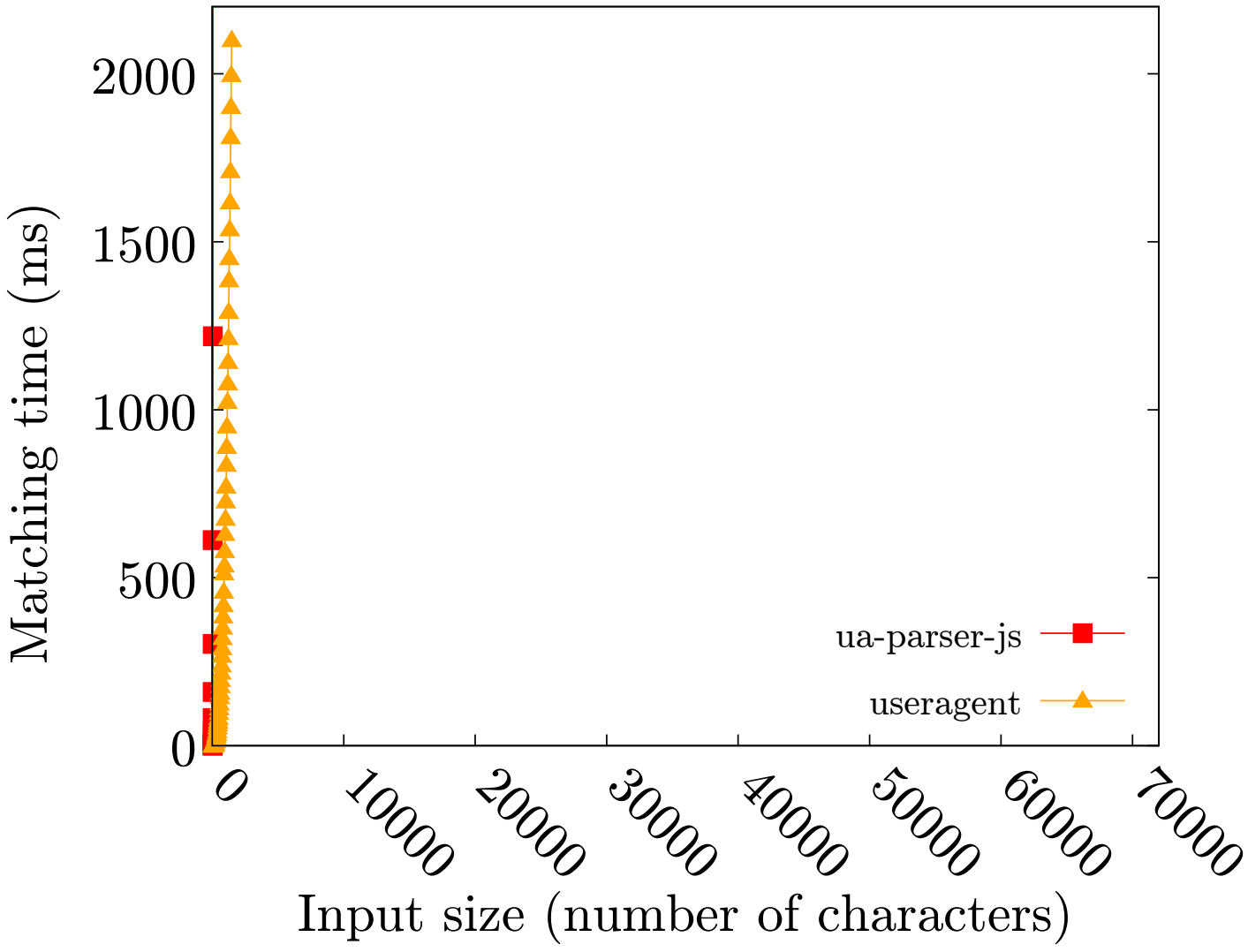
```
var MobileDetect = require("mobile-detect");
var headers = req.headers["user-agent"];
var md = new MobileDetect(headers);
md.phone();
```

- For each scenario, create HTTP level payloads
- In total **8 payloads** corresponding to 8 popular modules

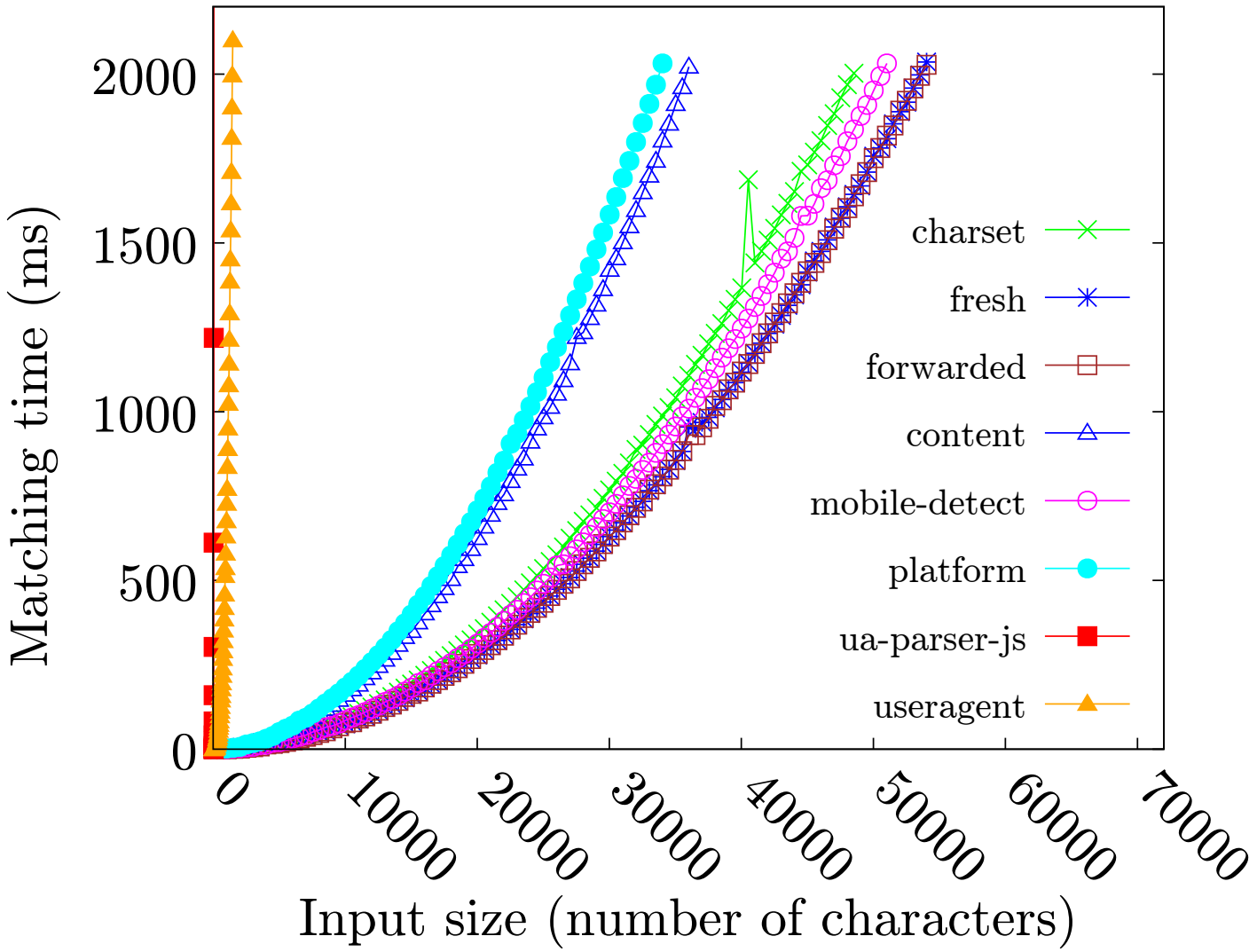
Phase 2: Input Dependency



Phase 2: Input Dependency



Phase 2: Input Dependency



Phase 3: Websites Analysis

P1

100ms

3x 5x

3x 5x

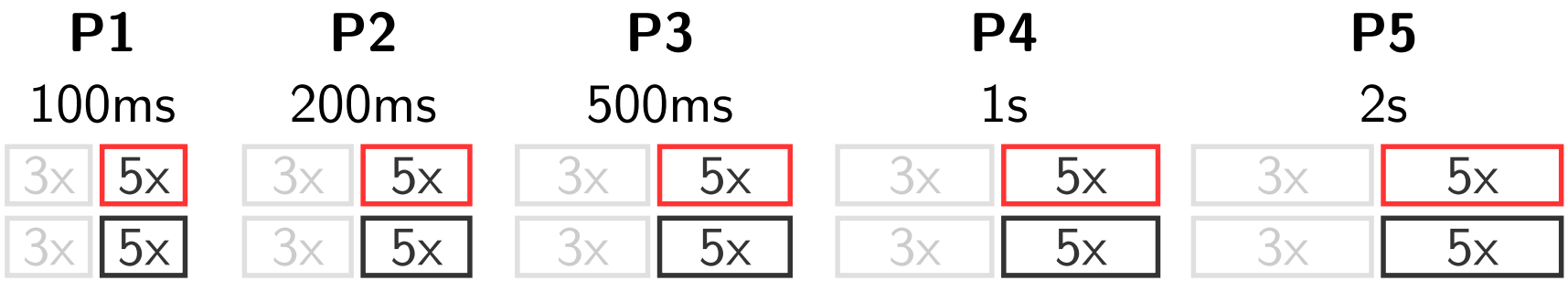
Phase 3: Websites Analysis

P1		**P2**	
100ms		200ms	
3x	5x	3x	5x
3x	5x	3x	5x

Phase 3: Websites Analysis

P1	**P2**	**P3**	**P4**	**P5**
100ms	200ms	500ms	1s	2s
3x 5x	3x 5x	3x 5x	3x 5x	3x 5x
3x 5x	3x 5x	3x 5x	3x 5x	3x 5x

Phase 3: Websites Analysis

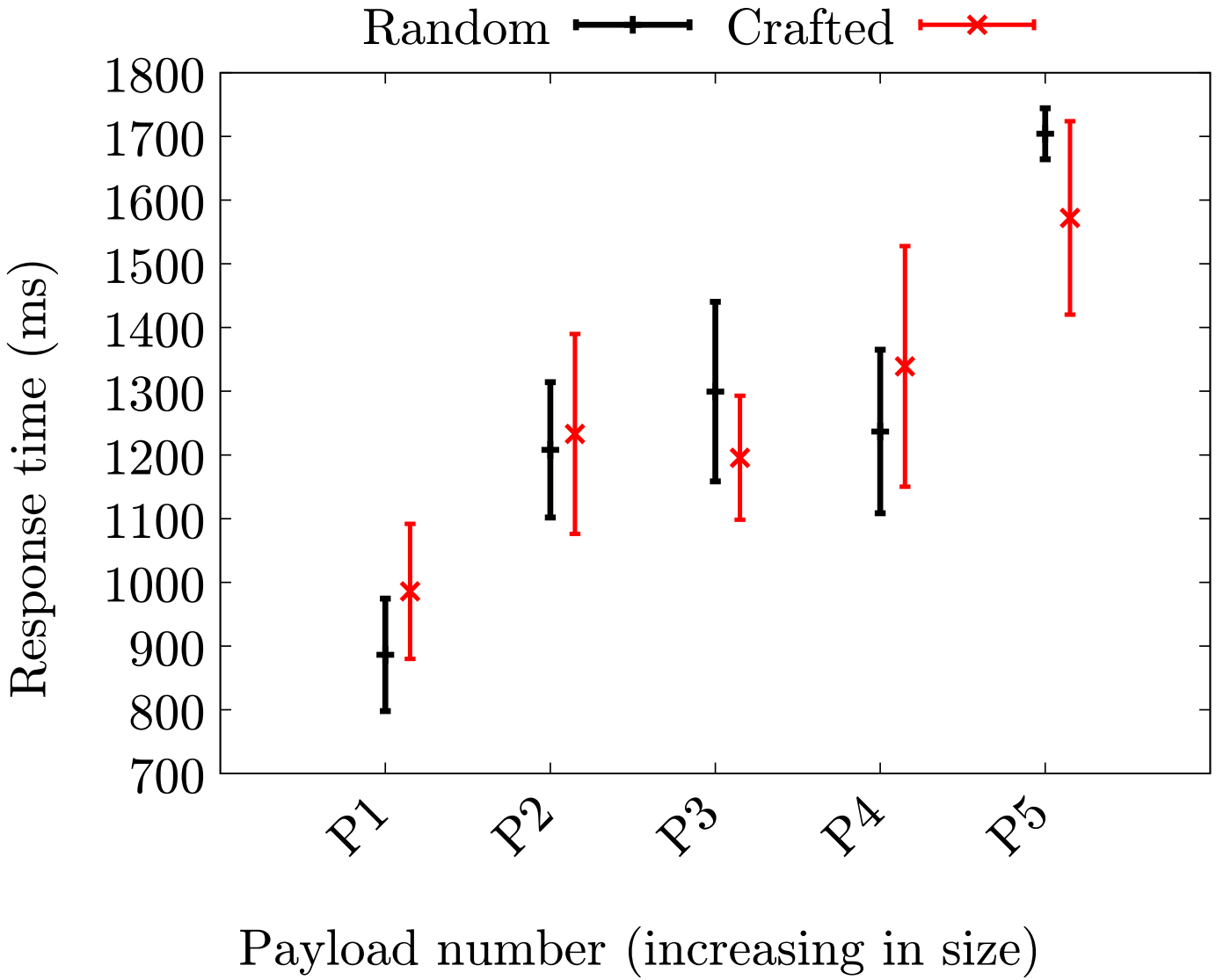


Criterion for vulnerable websites

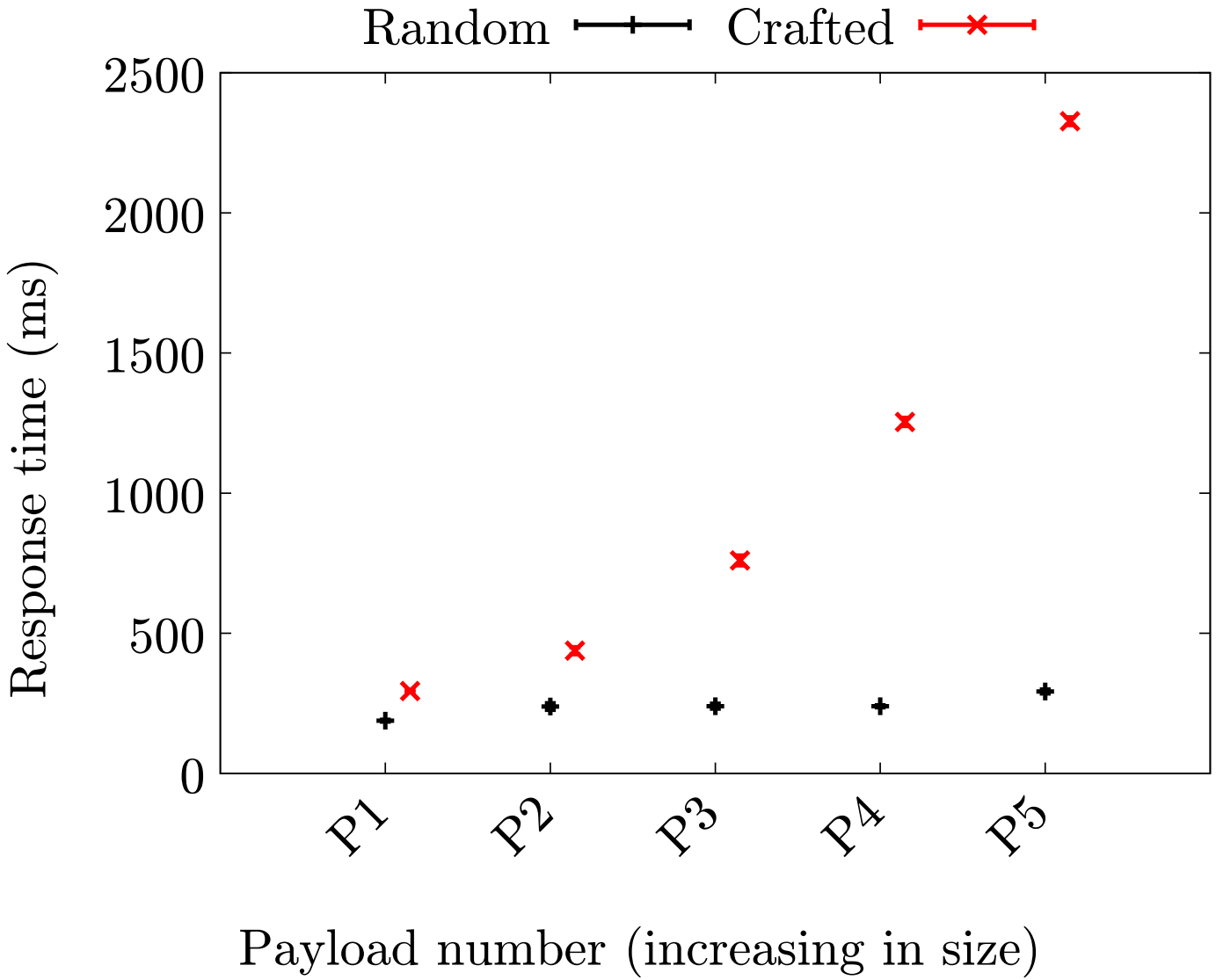
We consider a website to be vulnerable iff:

- statistically significant difference between the response times to **random** and **crafted** inputs,
- this difference increases when the input size increases.

Phase 3: Response Time of a Non-Vulnerable Website



Phase 3: Response Time of a Vulnerable Website

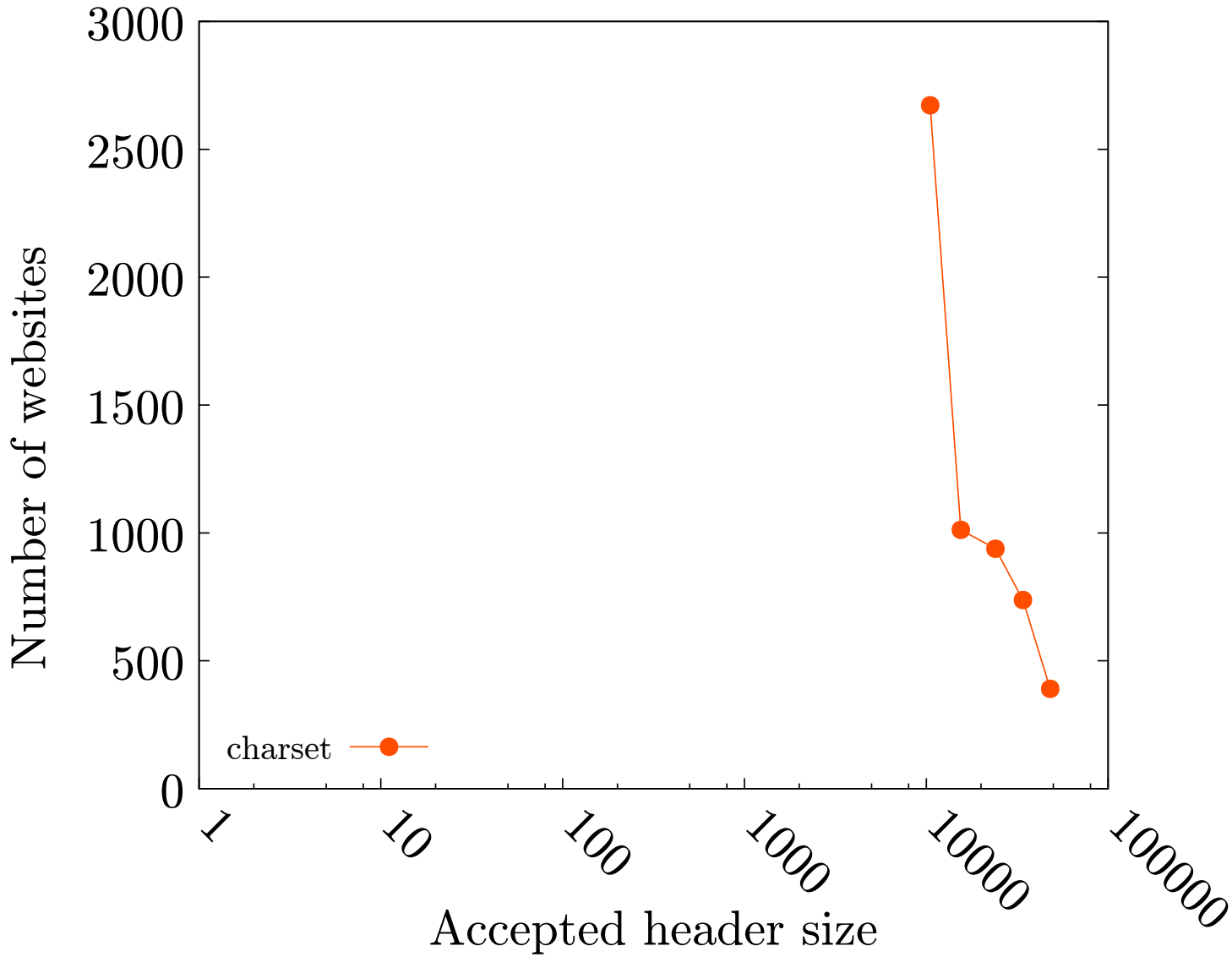


Phase 3: Number of Vulnerable Websites

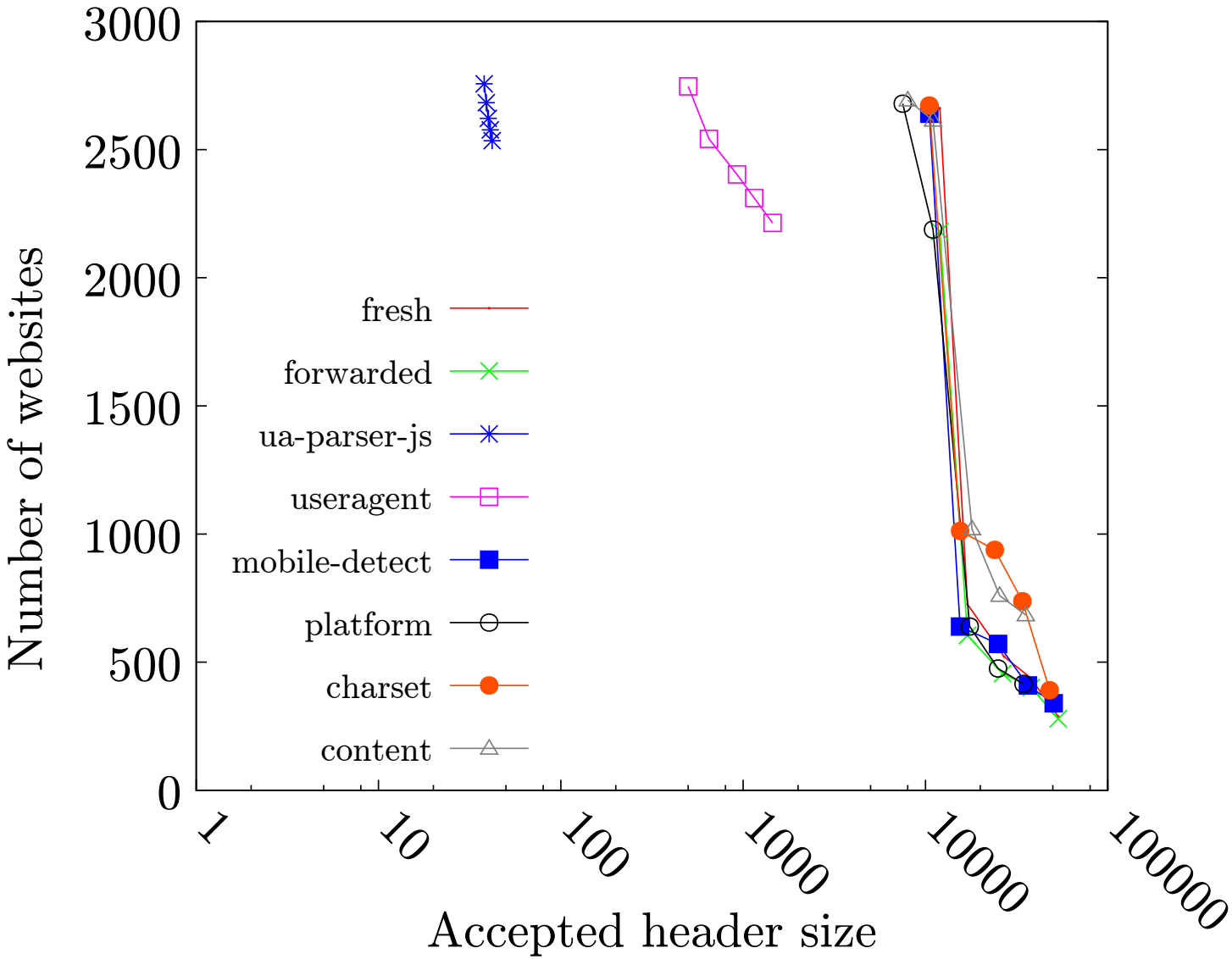
Exploit	Number of sites affected
fresh	241
forwarded	99
ua-parser-js	41
useragent	16
mobile-detect	9
platform	8
charset	3
content	0

In total: 339 (11%) websites are vulnerable

Defenses


3000
2500
2000
1500
1000
500
0
Number of websites
charset
1
10
100
1000
10000
100000
Accepted header size

Defenses


3000
2500
2000
1500
1000
500
0
Number of websites
1
10
100
1000
10000
100000
Accepted header size
fresh
forwarded
ua-parser-js
useragent
mobile-detect
platform
charset
content

Defenses (2)

- **Linear time matching algorithms / hybrid**
  Rust programming language

Defenses (2)

- **Linear time matching algorithms / hybrid**
  Rust programming language

- **Timeout on matching regular expressions**
  [Davis et al., USENIX Security, 2018], .NET framework

Defenses (2)

- **Linear time matching algorithms / hybrid**
  Rust programming language
- **Timeout on matching regular expressions**
  [Davis et al., USENIX Security, 2018], .NET framework
- **Tooling support for identifying ReDoS**
  Java programming language [Wüstholz et al., TACAS, 2017]

Conclusions

- ReDoS is a widespread problem in npm modules,

Conclusions

- ReDoS is a widespread problem in npm modules,
- Npm modules vulnerabilities **can be exploited** in live websites

Conclusions

- ReDoS is a widespread problem in npm modules,
- Npm modules vulnerabilities **can be exploited** in live websites
- **11% of websites** using Express are vulnerable to ReDoS

Conclusions

- ReDoS is a widespread problem in npm modules,
- Npm modules vulnerabilities **can be exploited** in live websites
- **11% of websites** using Express are vulnerable to ReDoS
- ReDoS vulnerabilities can be used to **fingerprint web servers**

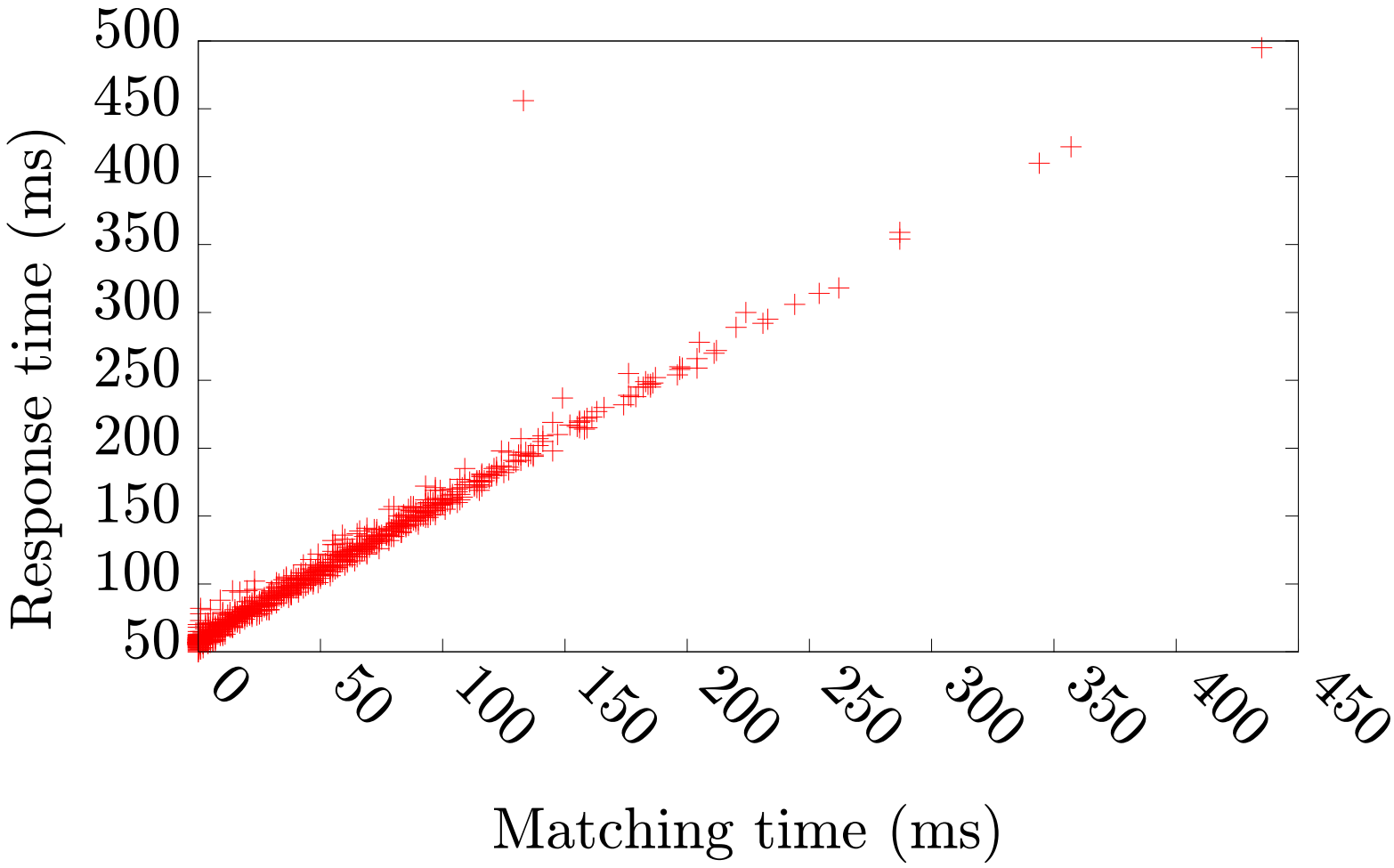
Conclusions

- ReDoS is a widespread problem in npm modules,
- Npm modules vulnerabilities **can be exploited** in live websites
- **11% of websites** using Express are vulnerable to ReDoS
- ReDoS vulnerabilities can be used to **fingerprint web servers**
- More tools are needed to mitigate the ReDoS risk

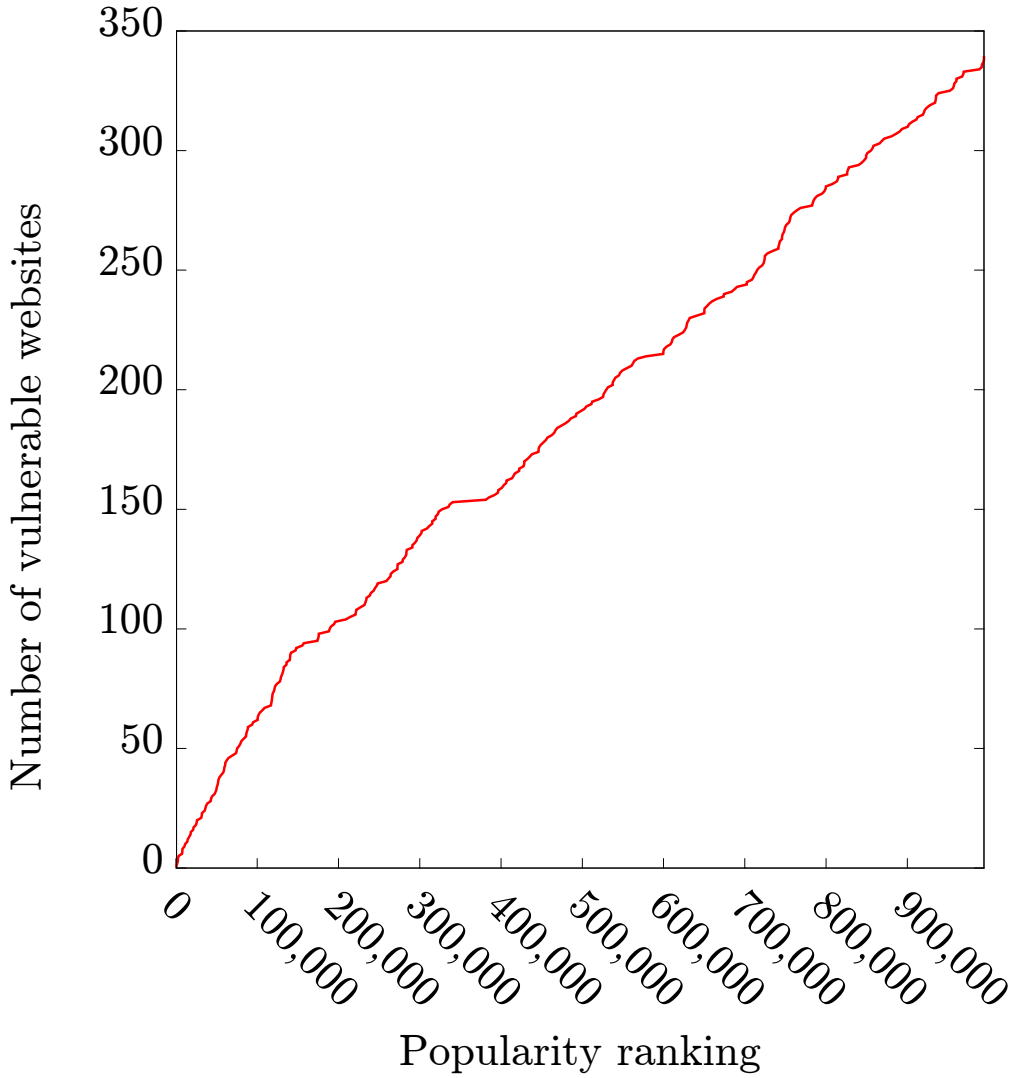
Conclusions

- ReDoS is a widespread problem in npm modules,
- Npm modules vulnerabilities **can be exploited** in live websites
- **11% of websites** using Express are vulnerable to ReDoS
- ReDoS vulnerabilities can be used to **fingerprint web servers**
- More tools are needed to mitigate the ReDoS risk

Is Response Time a Good Estimator?



Popularity of Vulnerable Websites



Dimensioning Payloads

Module	P1 100ms	P2 200ms	P3 500ms	P4 1s	P5 2s
fresh	12,000	17,000	27,000	37,500	**53,500**
forwarded	12,000	17,000	26,500	38,000	53,500
useragent	500	650	925	1,150	1,450
ua-parser-js	**38**	**39**	**40**	**41**	**42**
mobile-detect	10,500	15,500	25,000	36,500	50,500
platform	7,500	11,000	17,500	25,000	34,500
charset	10,500	15,500	24,000	34,000	48,000
content	8,000	11,000	18,000	25,500	35,500