DOM-XSS Detection via Webpage Interaction Fuzzing and URL Component Synthesis

Nuno Sabino, Darion Cassel, Rui Abreu, Pedro Adão, Lujo Bauer and Limin Jia

Carnegie Mellon University

TÉCNICO LISBOA

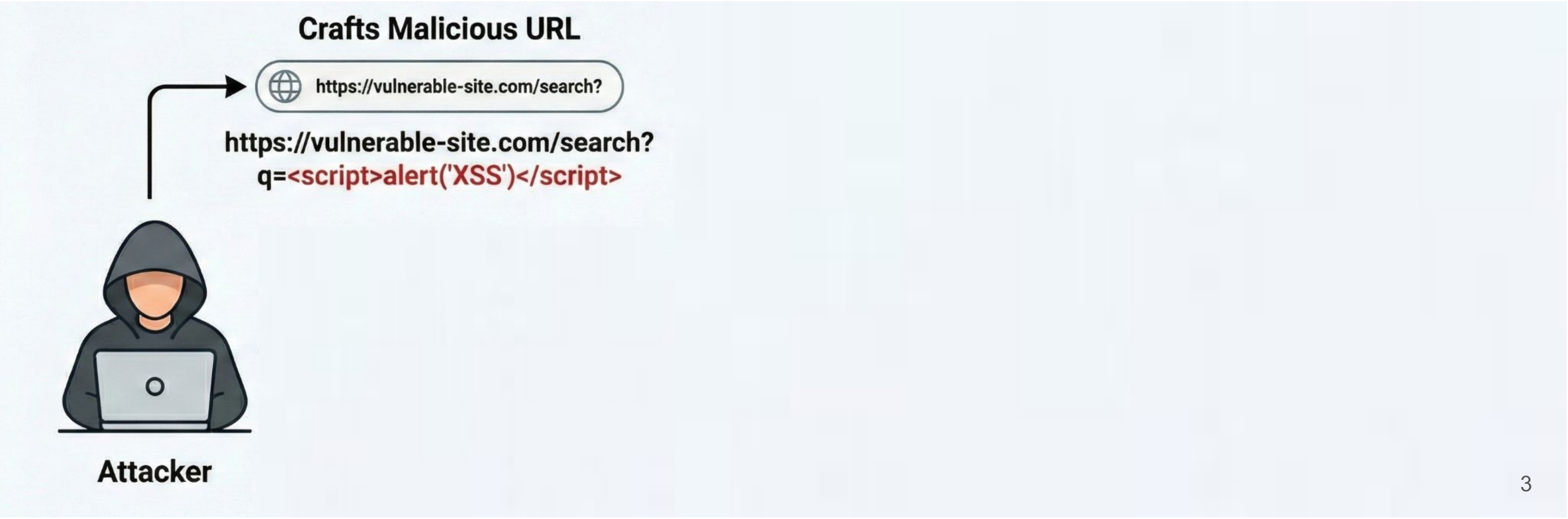
U. PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

DOM-based Cross-Site Scripting (DOM-XSS)

Nearly one in five web vulnerabilities are XSS (18.4% in 2025)

DOM-based Cross-Site Scripting (DOM-XSS)

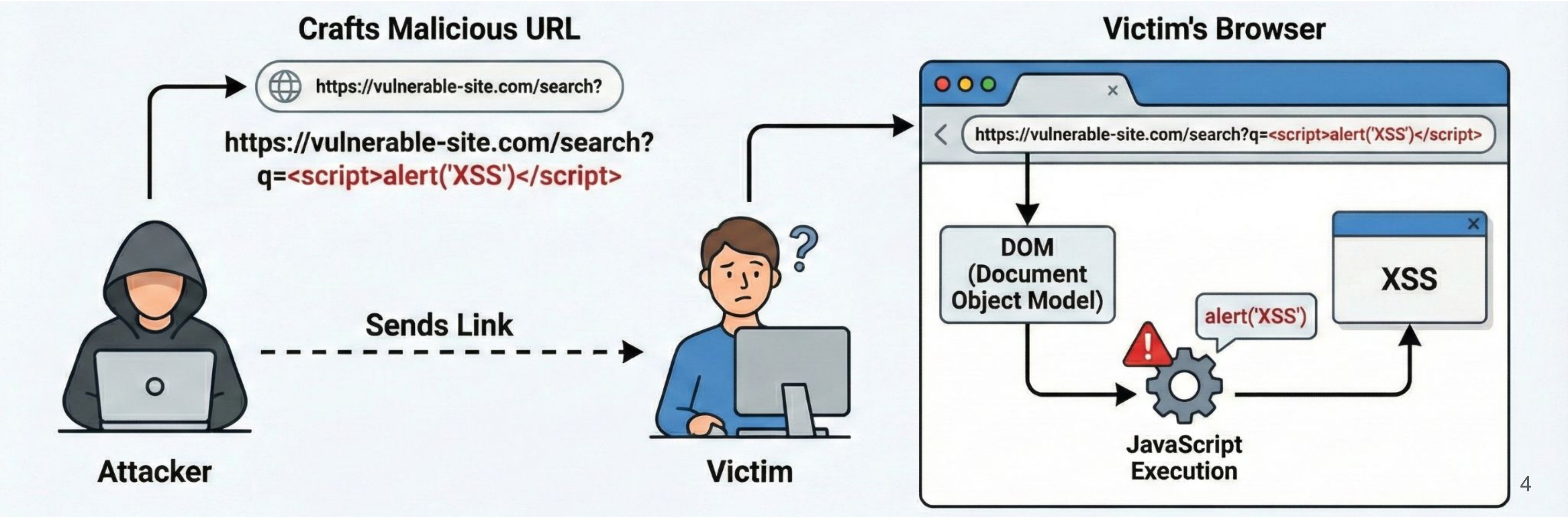
Nearly one in five web vulnerabilities are XSS (18.4% in 2025)



```
https://vulnerable-site.com/search?
q=<script>alert('XSS')</script>
```

DOM-based Cross-Site Scripting (DOM-XSS)

Nearly one in five web vulnerabilities are XSS (18.4% in 2025)


Crafts Malicious URL
https://vulnerable-site.com/search?
https://vulnerable-site.com/search?
q=<script>alert('XSS')</script>
Sends Link
Attacker
Victim
Victim's Browser
https://vulnerable-site.com/search?q=<script>alert('XSS')</script>
DOM (Document Object Model)
alert('XSS')
XSS
JavaScript Execution

Current client side web defenses are not effective

- CSP is often not configured properly

Current client side web defenses are not effective

- CSP is often not configured properly

- Trusted Types sanitizers are hard to implement correctly

Current client side web defenses are not effective

- CSP is often not configured properly
- Trusted Types sanitizers are hard to implement correctly
- Web application firewalls are easily bypassable

DOM-XSS studied over the years

Dynamic taint approaches can automatically and scalably analyse pages

- **25mflows (2013)**[1]→ 8,163 DOM-XSS flows after analyzing 504,275 pages

[1] Lekies et al. (2013) “25 million flows later: large-scale detection of DOM-based XSS”.

DOM-XSS studied over the years

Dynamic taint approaches can automatically and scalably analyse pages

- **25mflows (2013)**[1] → 8,163 DOM-XSS flows after analyzing 504,275 pages
- **DOMsday (2017)**[2] → 3,219 DOM-XSS flows after analysing 44,722 pages

[1] Lekies et al. (2013) "25 million flows later: large-scale detection of DOM-based XSS".

[2] Melicher et al. (2018) "Riding out domsday: Towards detecting and preventing DOM cross-site scripting".

DOM-XSS studied over the years

Dynamic taint approaches can automatically and scalably analyse pages

- **25mflows (2013)**[1] → 8,163 DOM-XSS flows after analyzing 504,275 pages
- **DOMsday (2017)**[2] → 3,219 DOM-XSS flows after analysing 44,722 pages
- **TalkGen (2020)**[3] → 7,199 DOM-XSS flows after analysing 390,092 pages

[1] Lekies et al. (2013) “25 million flows later: large-scale detection of DOM-based XSS”.

[2] Melicher et al. (2018) “Riding out domsday: Towards detecting and preventing DOM cross-site scripting”.

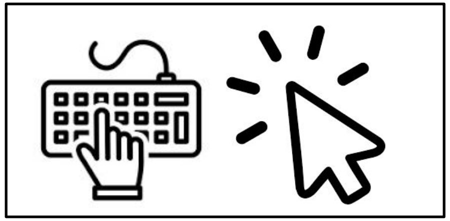
[3] Bensalim et al. (2021) “Talking about my generation: Targeted DOM-based XSS exploit generation using dynamic data flow analysis”.

Passive navigation misses vulnerabilities

- Page behavior may depend on:

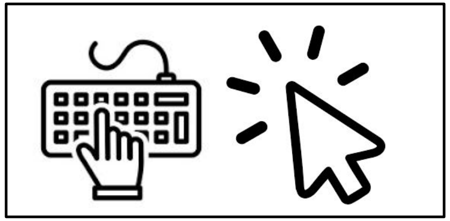
Passive navigation misses vulnerabilities

- Page behavior may depend on:
  - ↪ **Event handlers**



Passive navigation misses vulnerabilities

- Page behavior may depend on:
  - ↪ **Event handlers**



  - ↪ Parts of the URL, e.g., **GET parameters** and **fragment values**

https://www.example.com/calc?x=2&y=3#blue

Presentation overview

- ➢ Analysis and vulnerability confirmation methodology (shared with prior work)

Presentation overview

- ➢ Analysis and vulnerability confirmation methodology (shared with prior work)
- ➢ Novel components to improve DOM-XSS detection

Presentation overview

- ➢ Analysis and vulnerability confirmation methodology (shared with prior work)

- ➢ Novel components to improve DOM-XSS detection
  - ↪ Fuzzing to emulate user interactions

Presentation overview

- ➢ Analysis and vulnerability confirmation methodology (shared with prior work)

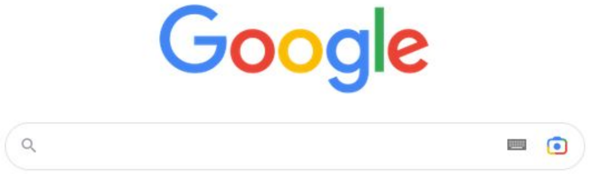
- ➢ Novel components to improve DOM-XSS detection
  - ↪ Fuzzing to emulate user interactions
  - ↪ Symbolic execution to synthesize GET parameters and hash values

Presentation overview

- ➢ Analysis and vulnerability confirmation methodology (shared with prior work)
- ➢ Novel components to improve DOM-XSS detection
  - ↪ Fuzzing to emulate user interactions
  - ↪ Symbolic execution to synthesize GET parameters and hash values
- ➢ Large scale evaluation results

Pipeline to detect DOM-XSS shared by prior work[1,2,3]

URL of target

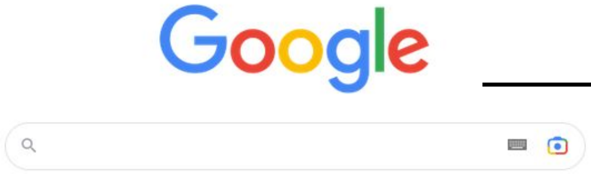


Most relevant prior work: TalkGen[1], DOMsday[2], 25mflows[3]

[1] Bensalim et al. (2021) “Talking about my generation: Targeted DOM-based XSS exploit generation using dynamic data flow analysis”.
[2] Melicher et al. (2018) “Riding out domsday: Towards detecting and preventing DOM cross-site scripting”.
[3] Lekies et al. (2013) “25 million flows later: large-scale detection of DOM-based XSS”.

Pipeline to detect DOM-XSS shared by prior work[1,2,3]

URL of target



Browser implementing Dynamic Taint Analysis

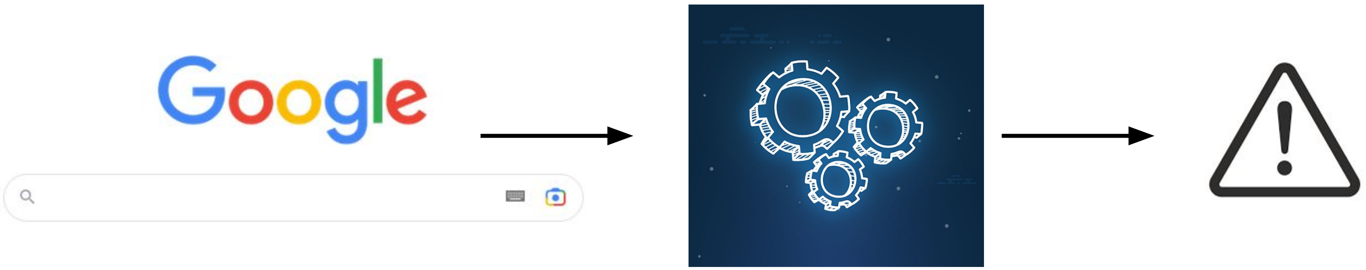
document.write(location.search)

Most relevant prior work: TalkGen[1], DOMsday[2], 25mflows[3]

[1] Bensalim et al. (2021) “Talking about my generation: Targeted DOM-based XSS exploit generation using dynamic data flow analysis”.
[2] Melicher et al. (2018) “Riding out domsday: Towards detecting and preventing DOM cross-site scripting”.
[3] Lekies et al. (2013) “25 million flows later: large-scale detection of DOM-based XSS”.

Pipeline to detect DOM-XSS shared by prior work[1,2,3]

URL of target **Browser implementing Dynamic Taint Analysis** **Potential flows**

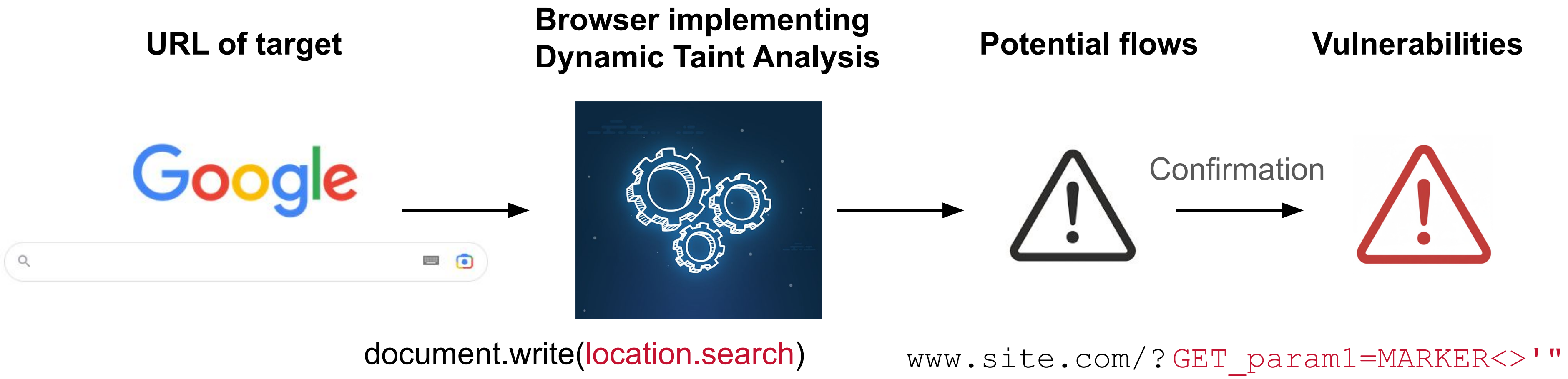


document.write(location.search)

Most relevant prior work: TalkGen[1], DOMsday[2], 25mflows[3]

[1] Bensalim et al. (2021) “Talking about my generation: Targeted DOM-based XSS exploit generation using dynamic data flow analysis”.
[2] Melicher et al. (2018) “Riding out domsday: Towards detecting and preventing DOM cross-site scripting”.
[3] Lekies et al. (2013) “25 million flows later: large-scale detection of DOM-based XSS”.

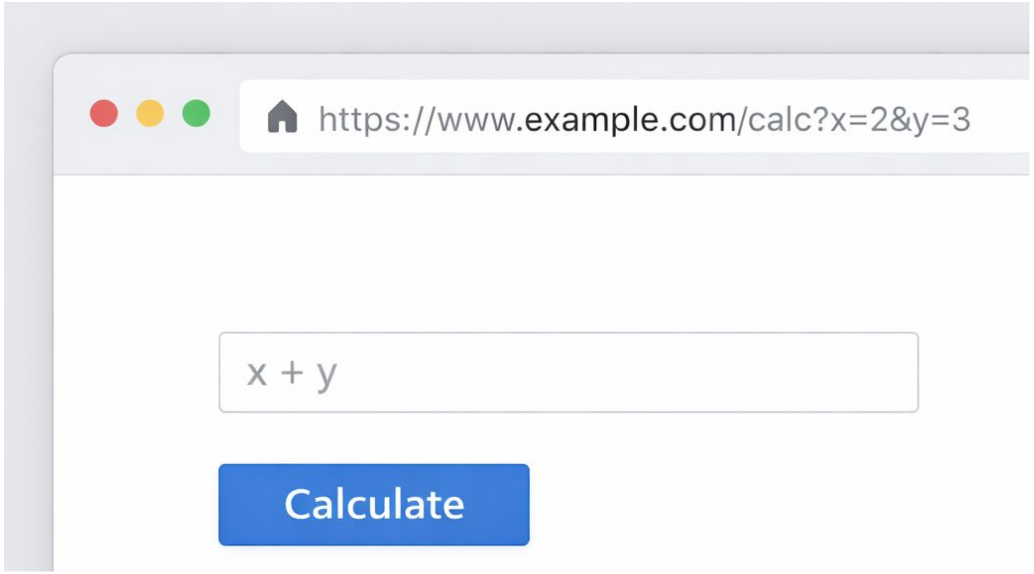
Pipeline to detect DOM-XSS shared by prior work[1,2,3]



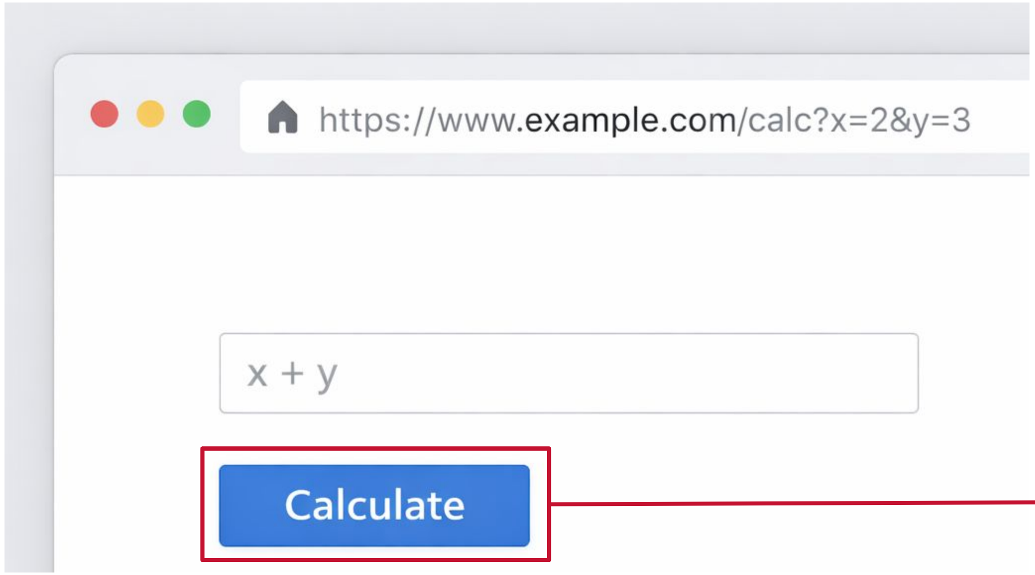
Most relevant prior work: TalkGen[1], DOMsday[2], 25mflows[3]

[1] Bensalim et al. (2021) “Talking about my generation: Targeted DOM-based XSS exploit generation using dynamic data flow analysis”.
[2] Melicher et al. (2018) “Riding out domsday: Towards detecting and preventing DOM cross-site scripting”.
[3] Lekies et al. (2013) “25 million flows later: large-scale detection of DOM-based XSS”.

Prior works miss flows that require program exploration

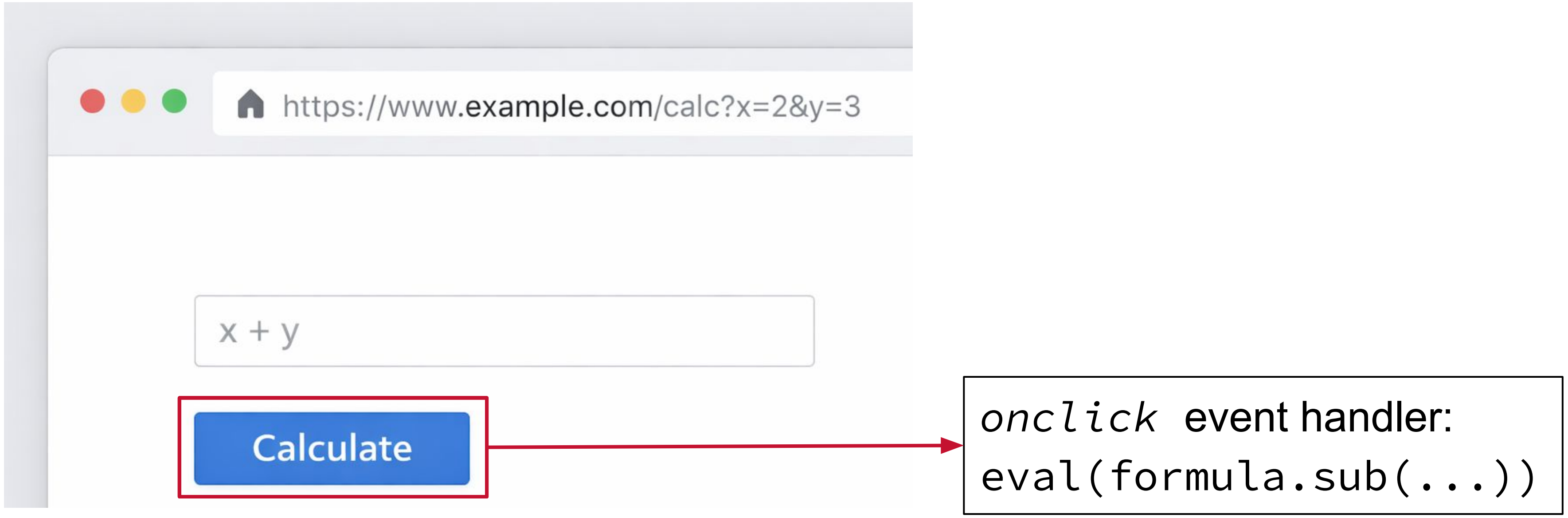


Prior works miss flows that require program exploration



`onclick` event handler:
`eval(formula.sub(...))`

Prior works miss flows that require program exploration



onclick event handler:

```
eval(formula.sub(...))
```

Our proposal: Fuzzing user interactions to trigger event handlers

Automatic methods to simulate user actions exist, but…

- Only support mouse, keyboard events and forms[1,2,4,5,6]

[1] Mesbah et al. (2012) “Crawling Ajax-based web applications through dynamic analysis of user interface state changes”.
[2] Pellegrino et al. (2015) “jÄk: Using dynamic analysis to crawl and test modern web applications”.
[3] Eriksson et al. (2021) “Black Widow: Blackbox Data-driven Web Scanning”.
[4] Drakonakis et al. (2023) “ReScan: A middleware framework for realistic and robust black-box web application scanning”.
[5] Weidmann et al. (2023) “Load-and-act: Increasing page coverage of web applications”.
[6] Stafeev et al. (2025) “YURASCANNER: Leveraging LLMs for Task-driven Web App Scanning”.

Automatic methods to simulate user actions exist, but…

- Only support mouse, keyboard events and forms[1,2,4,5,6]

- Can produce false positives via unrealistic event handler execution[1,2,3,4]

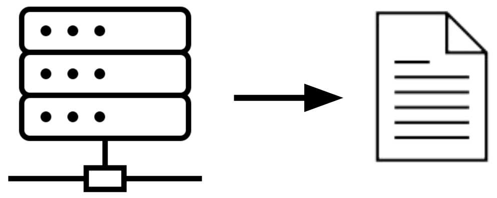
[1] Mesbah et al. (2012) “Crawling Ajax-based web applications through dynamic analysis of user interface state changes”.
[2] Pellegrino et al. (2015) “jÄk: Using dynamic analysis to crawl and test modern web applications”.
[3] Eriksson et al. (2021) “Black Widow: Blackbox Data-driven Web Scanning”.
[4] Drakonakis et al. (2023) “ReScan: A middleware framework for realistic and robust black-box web application scanning”.
[5] Weidmann et al. (2023) “Load-and-act: Increasing page coverage of web applications”.
[6] Stafeev et al. (2025) “YURASCANNER: Leveraging LLMs for Task-driven Web App Scanning”.

Automatic methods to simulate user actions exist, but…

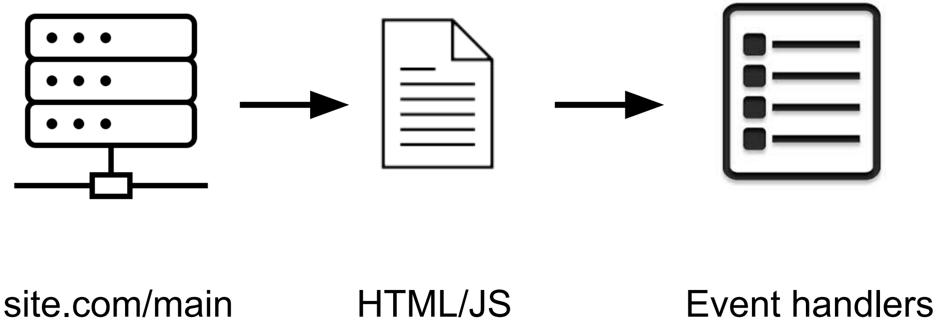
- Only support mouse, keyboard events and forms[1,2,4,5,6]
- Can produce false positives via unrealistic event handler execution[1,2,3,4]
- Computationally heavy [4,6]

[1] Mesbah et al. (2012) “Crawling Ajax-based web applications through dynamic analysis of user interface state changes”.
[2] Pellegrino et al. (2015) “jÄk: Using dynamic analysis to crawl and test modern web applications”.
[3] Eriksson et al. (2021) “Black Widow: Blackbox Data-driven Web Scanning”.
[4] Drakonakis et al. (2023) “ReScan: A middleware framework for realistic and robust black-box web application scanning”.
[5] Weidmann et al. (2023) “Load-and-act: Increasing page coverage of web applications”.
[6] Stafeev et al. (2025) “YURASCANNER: Leveraging LLMs for Task-driven Web App Scanning”.

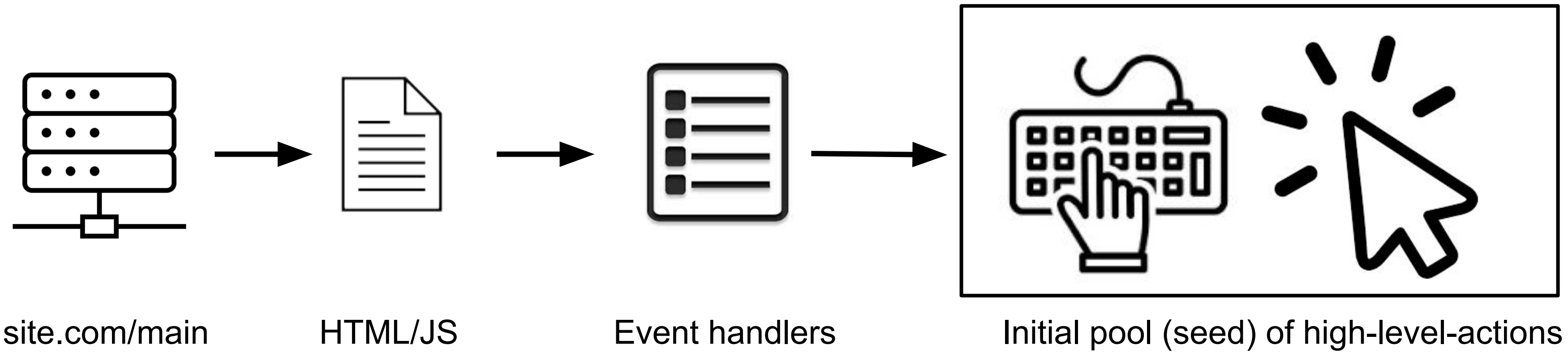
Fuzzing user interactions to execute event handlers



Fuzzing user interactions to execute event handlers



Fuzzing user interactions to execute event handlers

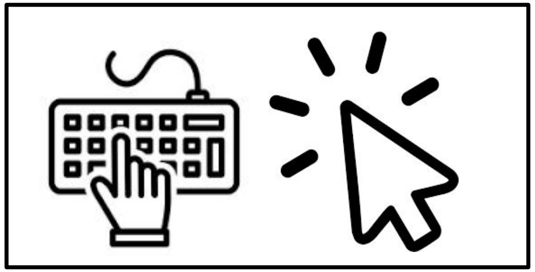


High level action: A specific interaction with the page

onkeydown → { `focus(element); generateKeyPress(key=random)` }

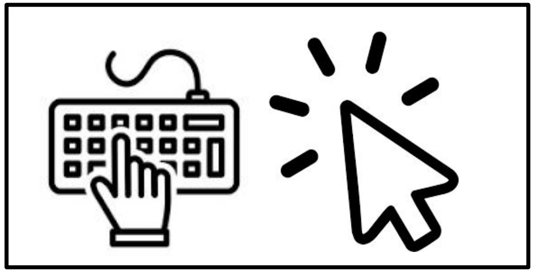
Interactions are combined to improve event handler execution

```
{ReleaseMouse(), Drag(position), Click(element)}
```



Interactions are combined to improve event handler execution

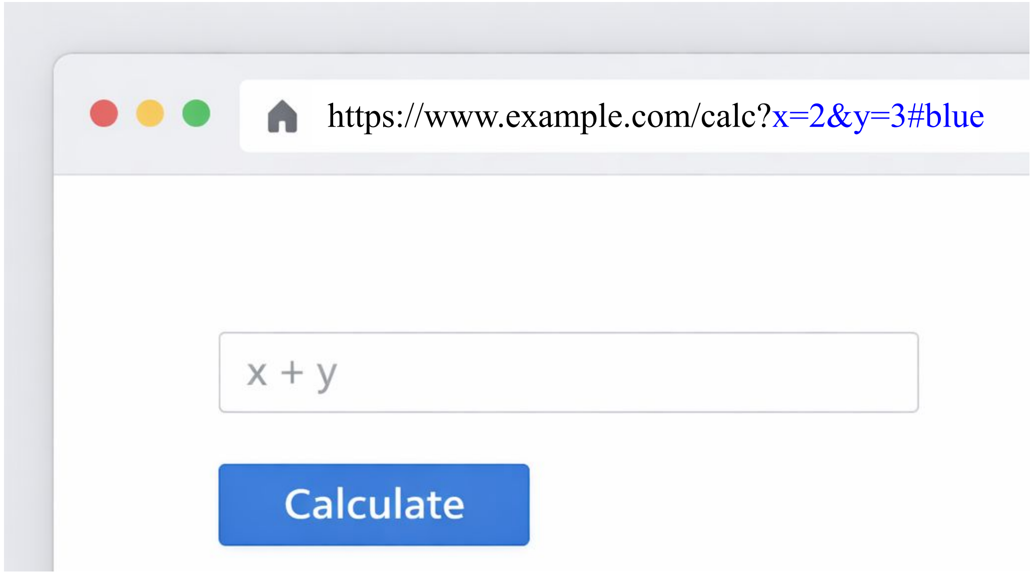
```
{ReleaseMouse(), Drag(position), Click(element)}
```



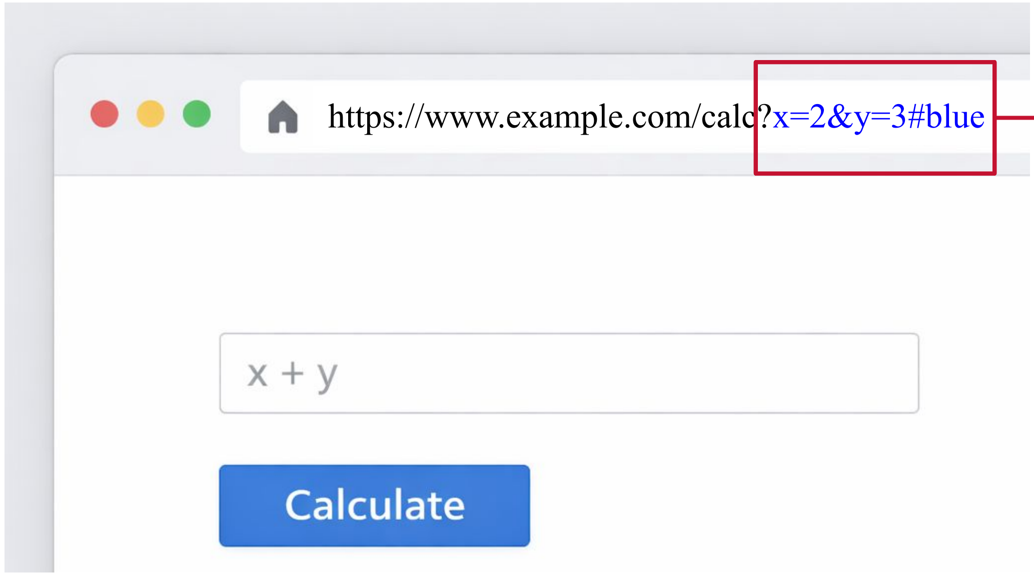
Mutate / combine actions

```
{Click(element);Drag(position);ReleaseMouse(), ... }
```

GET parameters can influence page behavior

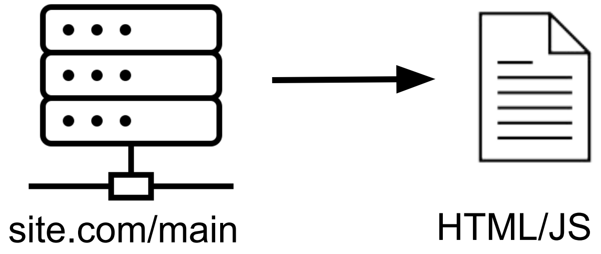

https://www.example.com/calc?x=2&y=3#blue

GET parameters can influence page behavior

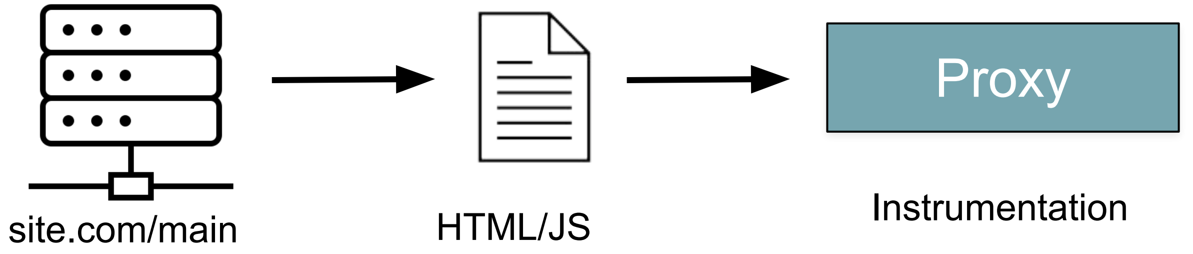

https://www.example.com/calc?x=2&y=3#blue

What happens with different values?

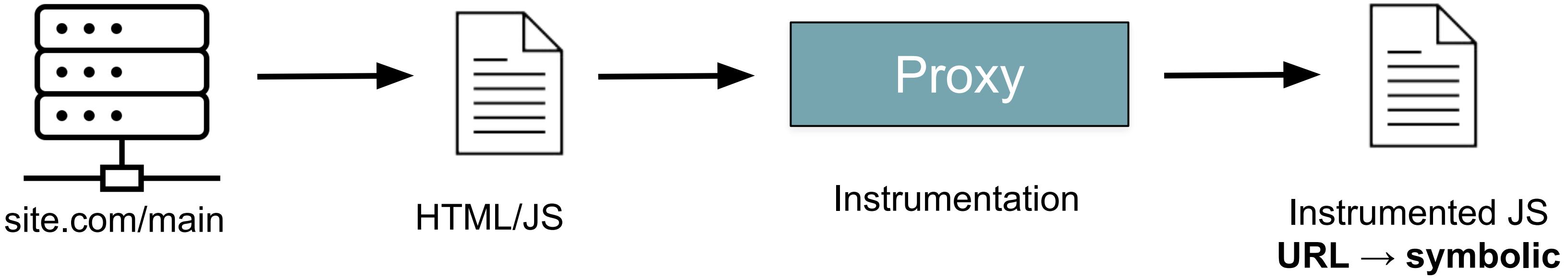
Instrumenting pages to track symbolic constraints



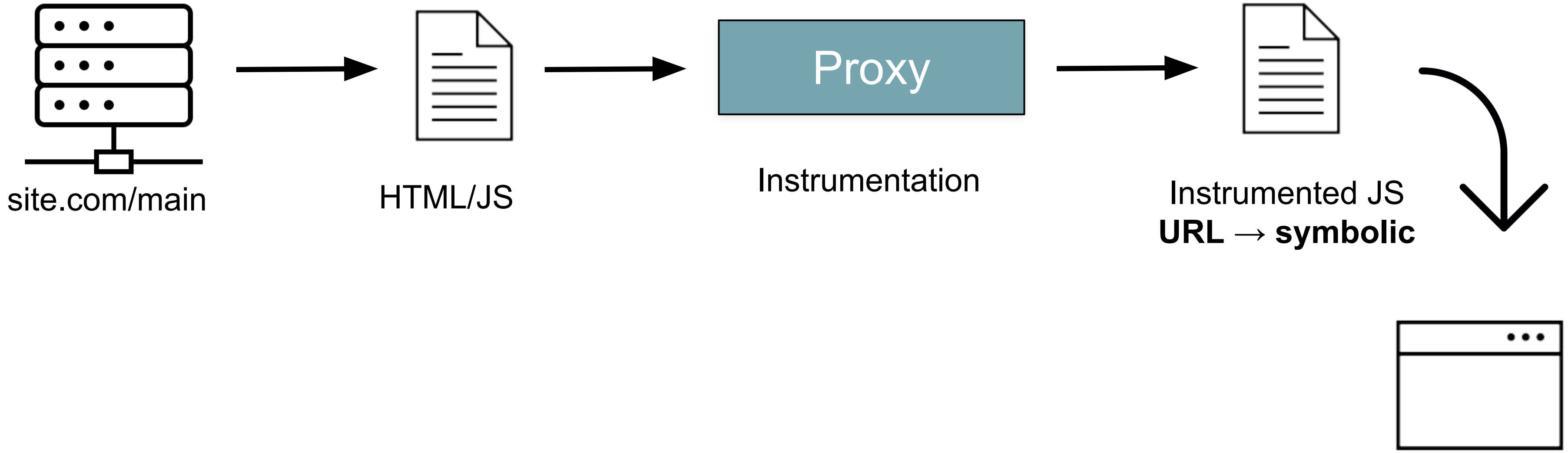
Instrumenting pages to track symbolic constraints



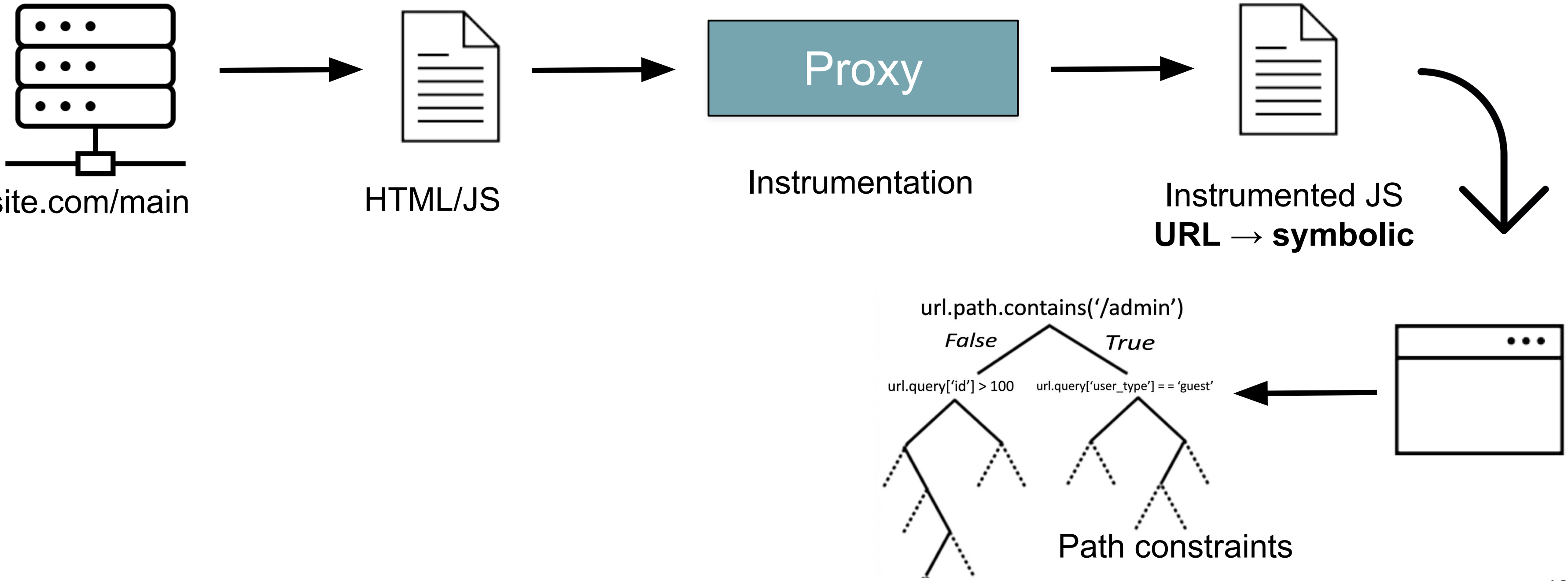
Instrumenting pages to track symbolic constraints



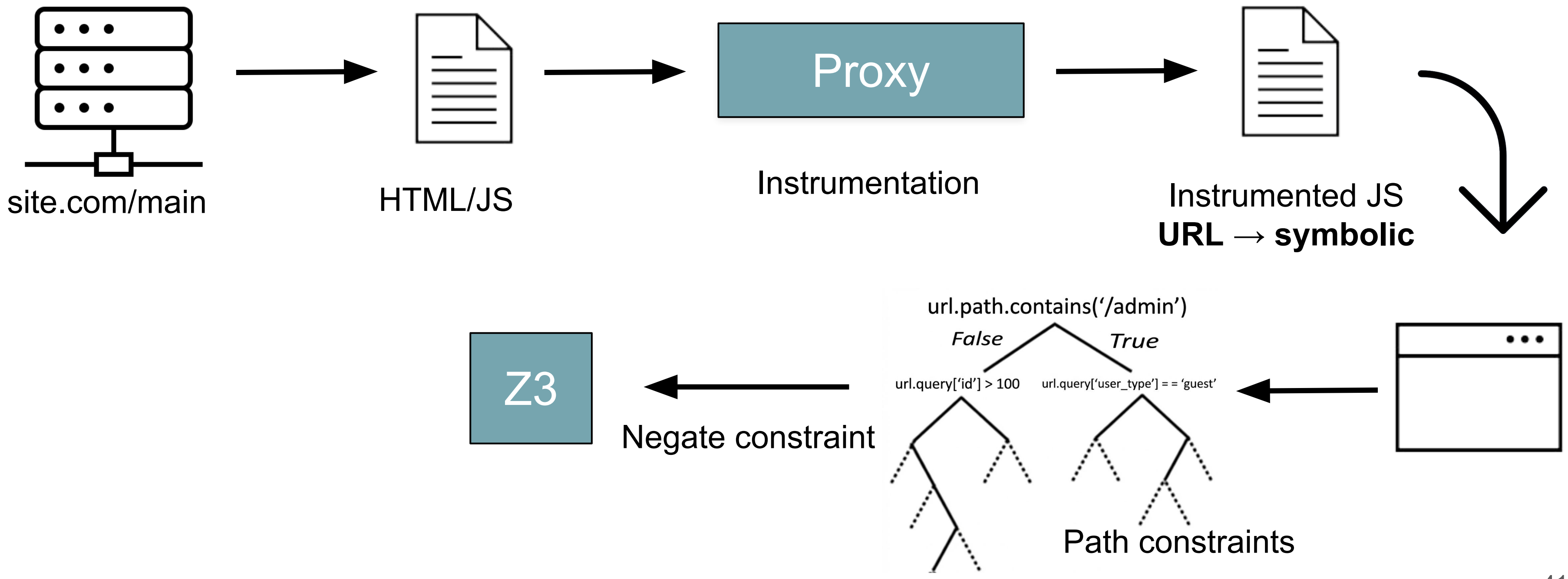
Instrumenting pages to track symbolic constraints



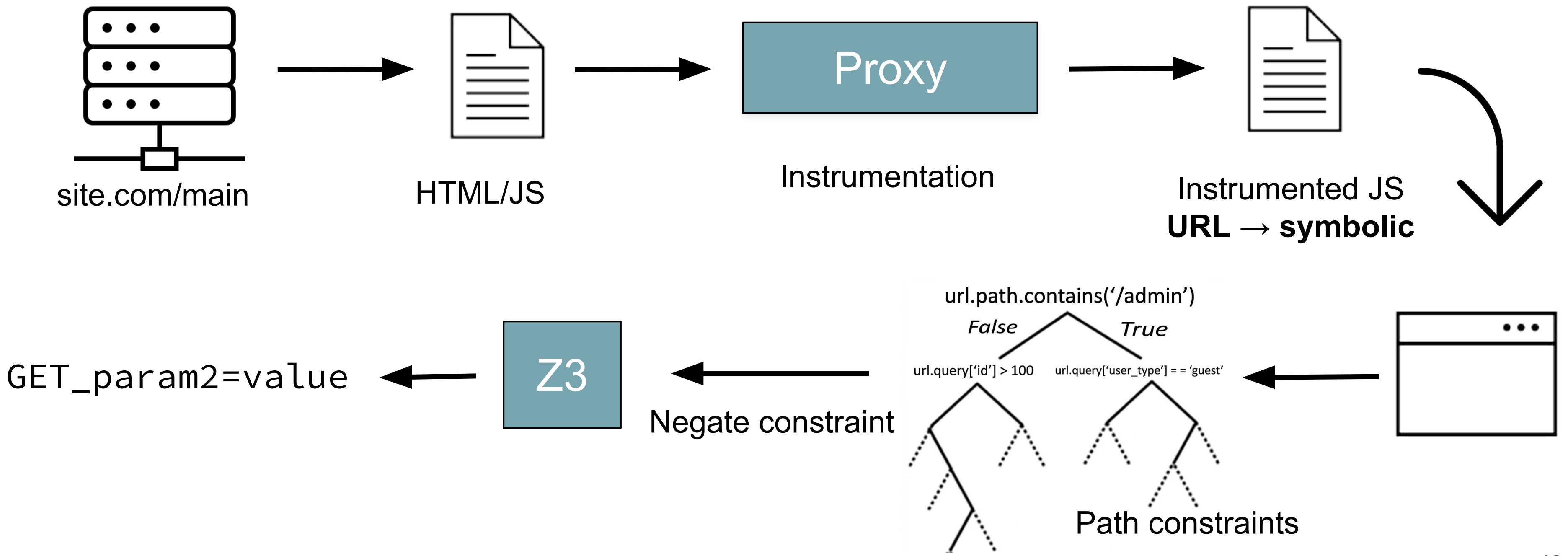
Instrumenting pages to track symbolic constraints



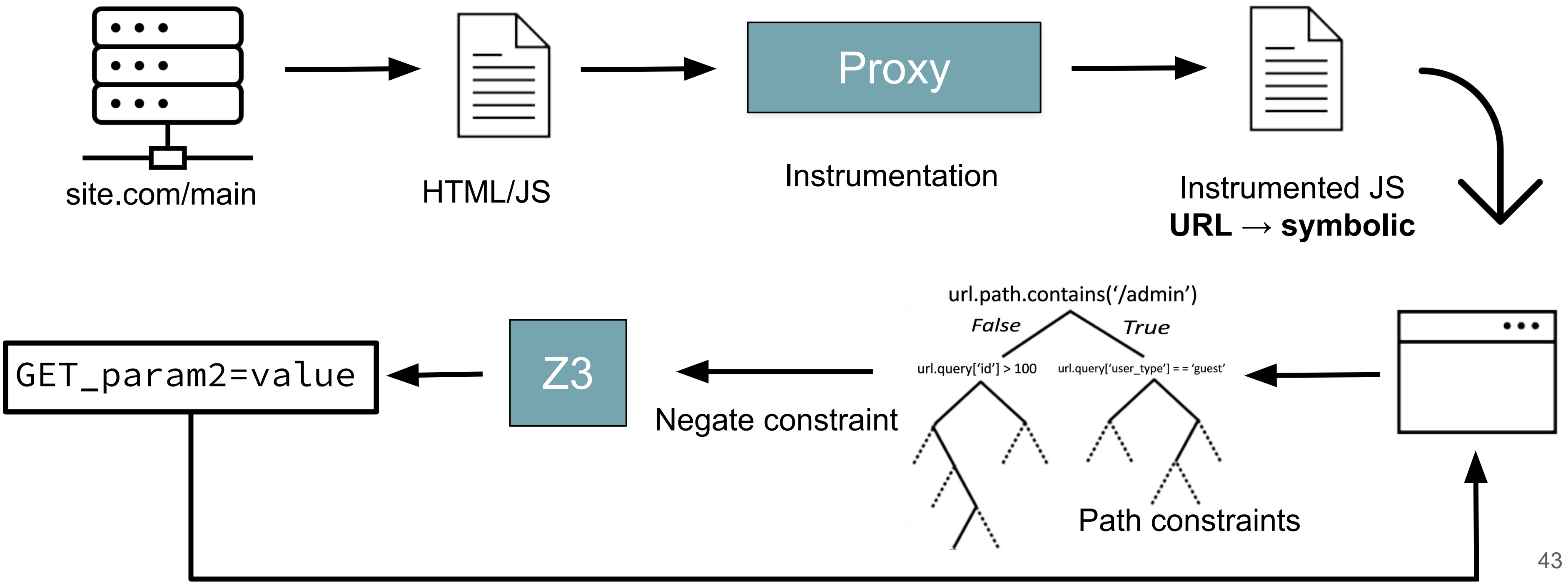
Instrumenting pages to track symbolic constraints



Instrumenting pages to track symbolic constraints



Instrumenting pages to track symbolic constraints



Evaluating user action fuzzer on a large-scale crawl

RQ1: How does fuzzing compare to passive navigation for DOM-XSS discovery?

RQ2: How effective is DSE at synthesizing relevant GET parameters?

RQ3: How does current DOM-XSS prevalence compare to what prior studies reported?

Evaluating user action fuzzer on a large-scale crawl

RQ1: How does fuzzing compare to passive navigation for DOM-XSS discovery?

RQ2: How effective is DSE at synthesizing relevant GET parameters?

RQ3: How does current DOM-XSS prevalence compare to what prior studies reported?

Dataset: 44,480 URLs from popular domains (Tranco list)

Evaluating user action fuzzer on a large-scale crawl

RQ1: How does fuzzing compare to passive navigation for DOM-XSS discovery?

RQ2: How effective is DSE at synthesizing relevant GET parameters?

RQ3: How does current DOM-XSS prevalence compare to what prior studies reported?

Dataset: 44,480 URLs from popular domains (Tranco list)

Experiments (3 minute / page timeout):

- **Passive** (RQ1,RQ3)**:** Replication of *DOMsday* on an upgraded Chrome

Evaluating user action fuzzer on a large-scale crawl

RQ1: How does fuzzing compare to passive navigation for DOM-XSS discovery?

RQ2: How effective is DSE at synthesizing relevant GET parameters?

RQ3: How does current DOM-XSS prevalence compare to what prior studies reported?

Dataset: 44,480 URLs from popular domains (Tranco list)

Experiments (3 minute / page timeout):

- **Passive** (RQ1,RQ3)**:** Replication of *DOMsday* on an upgraded Chrome
- **Fuzzer** (RQ1,RQ2)**:** Our fuzzing condition

Evaluating user action fuzzer on a large-scale crawl

RQ1: How does fuzzing compare to passive navigation for DOM-XSS discovery?

RQ2: How effective is DSE at synthesizing relevant GET parameters?

RQ3: How does current DOM-XSS prevalence compare to what prior studies reported?

Dataset: 44,480 URLs from popular domains (Tranco list)

Experiments (3 minute / page timeout):

- **Passive** (RQ1,RQ3)**:** Replication of *DOMsday* on an upgraded Chrome
- **Fuzzer** (RQ1,RQ2)**:** Our fuzzing condition
- **Fuzzer-DSE24** (RQ2): DSE augmentation (24 hour timeout) followed by fuzzing

Evaluating user action fuzzer on a large-scale crawl

RQ1: How does fuzzing compare to passive navigation for DOM-XSS discovery?

RQ2: How effective is DSE at synthesizing relevant GET parameters?

RQ3: How does current DOM-XSS prevalence compare to what prior studies reported?

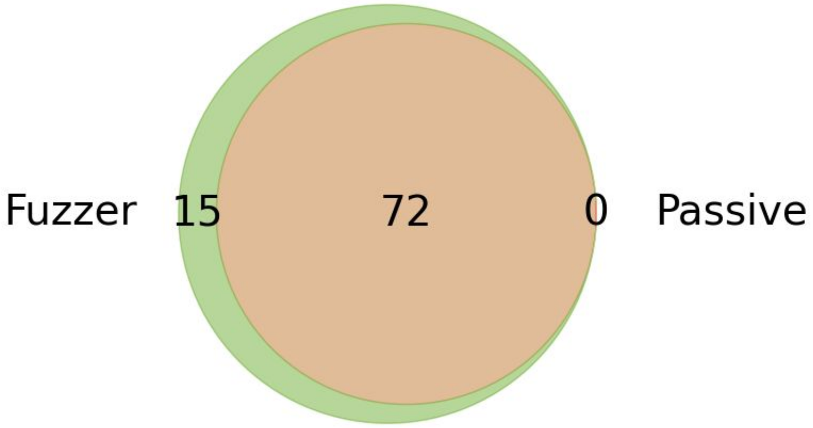
Dataset: 44,480 URLs from popular domains (Tranco list)

Experiments (3 minute / page timeout):

- **Passive** (RQ1,RQ3)**:** Replication of *DOMsday* on an upgraded Chrome
- **Fuzzer** (RQ1,RQ2)**:** Our fuzzing condition
- **Fuzzer-DSE24** (RQ2): DSE augmentation (24 hour timeout) followed by fuzzing
- **TalkGen** (RQ3)**:** Replication of *TalkGen* on an upgraded Firefox

RQ1: Fuzzing user interaction improves DOM-XSS detection

Fuzzing improves DOM-XSS confirmed flows discovery by 21% (compared to Passive)

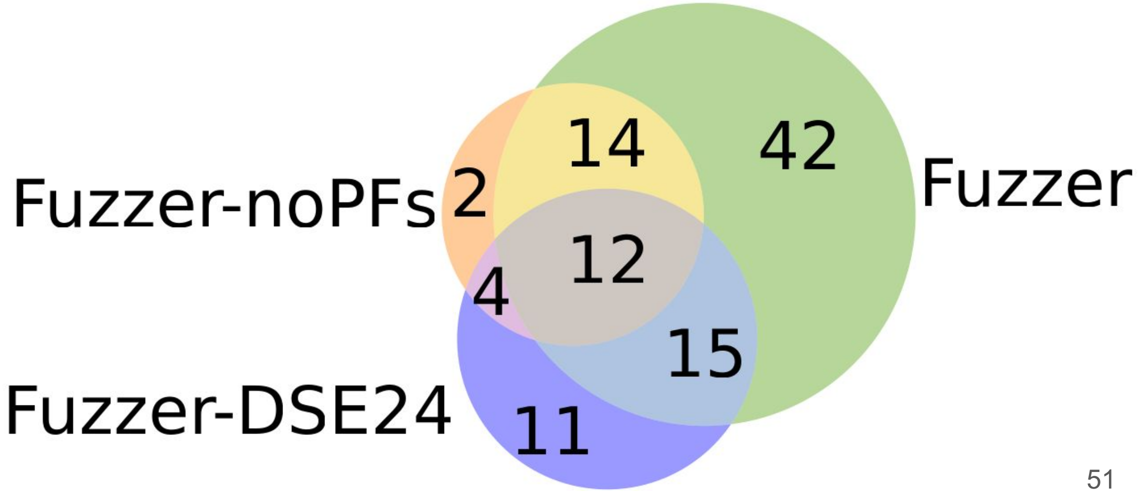


Confirmed flows discovered after five repetitions and after deduplicating flows

RQ2: DSE synthesizes PFs that trigger new vulnerabilities

Fuzzer-noPFs strips parameters and fragments from each target URL before Fuzzing

Symbolic execution generates new parameter combinations:

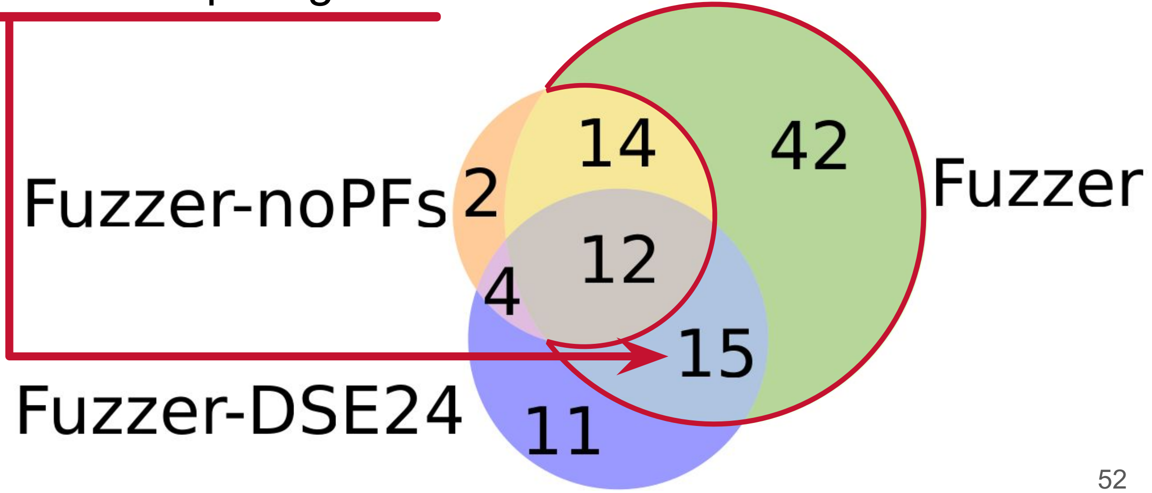


RQ2: DSE synthesizes PFs that trigger new vulnerabilities

Fuzzer-noPFs strips parameters and fragments from each target URL before Fuzzing

Symbolic execution generates new parameter combinations:

- rediscovers 26% of confirmed flows requiring PFs

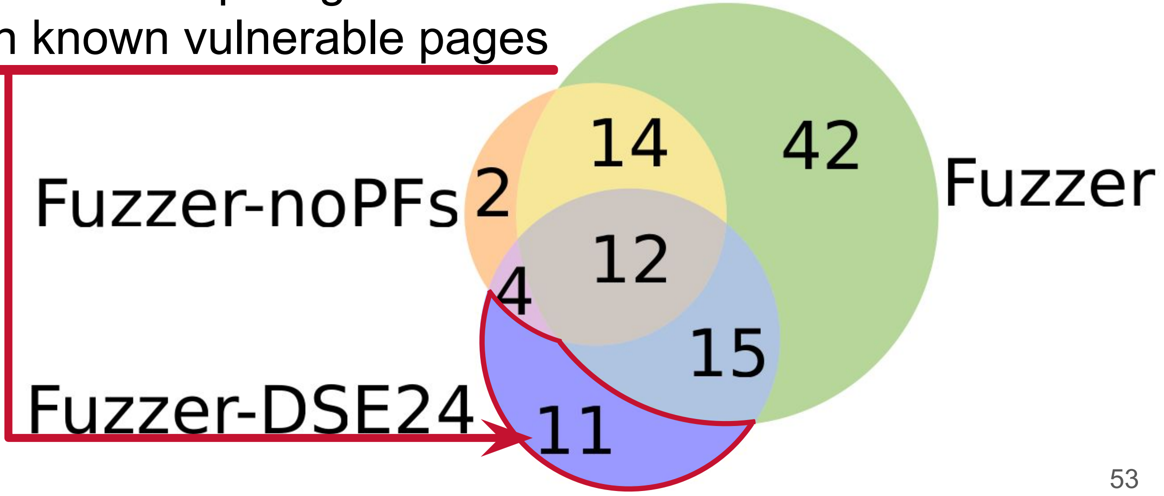


RQ2: DSE synthesizes PFs that trigger new vulnerabilities

Fuzzer-noPFs strips parameters and fragments from each target URL before Fuzzing

Symbolic execution generates new parameter combinations:

- rediscovers 26% of confirmed flows requiring PFs
- reveals new vulnerabilities on known vulnerable pages

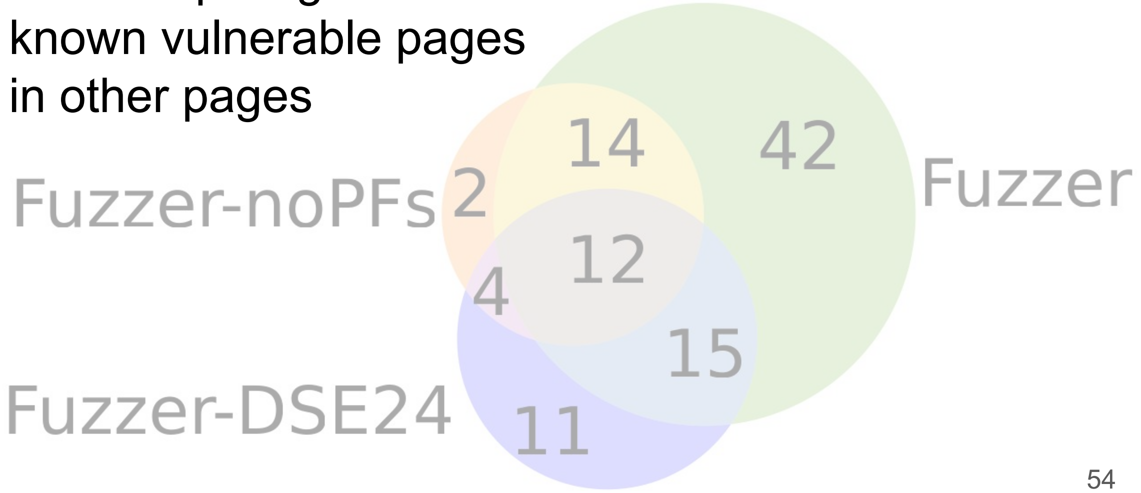


RQ2: DSE synthesizes PFs that trigger new vulnerabilities

Fuzzer-noPFs strips parameters and fragments from each target URL before Fuzzing

Symbolic execution generates new parameter combinations:

- rediscovers 26% of confirmed flows requiring PFs
- reveals new vulnerabilities on known vulnerable pages
- reveals 10 new vulnerabilities in other pages



RQ3: Comparison with prior work

Metric	DOMsday (replication) aka Passive	TalkGen (replication)	TalkGen	DOMsday	25mFlows
Date	04/2025	04/2025	09/2020	08/2017	11/2013
Pot. flows (per 1k pages)	45.5	76.6	40.3	116.7	?
Conf. flows (per 1k pages)	1.6	1.5	18.5	72.0	16.2

RQ3: Comparison with prior work

Metric	DOMsday (replication) aka Passive	TalkGen (replication)	TalkGen	DOMsday	25mFlows
Date	04/2025	04/2025	09/2020	08/2017	11/2013
Pot. flows (per 1k pages)	45.5	76.6	40.3	116.7	?
Conf. flows (per 1k pages)	**1.6**	1.5	18.5	**72.0**	16.2

- Significant decrease of confirmed flows frequency!

RQ3: Comparison with prior work

Metric	DOMsday (replication) aka Passive	TalkGen (replication)	TalkGen	DOMsday	25mFlows
Date	04/2025	04/2025	09/2020	08/2017	11/2013
Pot. flows (per 1k pages)	45.5	76.6	40.3	116.7	?
Conf. flows (per 1k pages)	1.6	**1.5**	**18.5**	72.0	16.2

- Significant decrease of confirmed flows frequency!

RQ3: Comparison with prior work

Metric	DOMsday (replication) aka Passive	TalkGen (replication)	TalkGen	DOMsday	25mFlows
Date	04/2025	04/2025	09/2020	08/2017	11/2013
Pot. flows (per 1k pages)	45.5	76.6	40.3	116.7	?
Conf. flows (per 1k pages)	1.6	**1.5**	**18.5**	72.0	16.2

Most impactful factor introducing a difference in confirmed flows:

- URL encoding mechanism evolution: inflates confirmed flows by 5x

RQ3: Comparison with prior work

Metric	DOMsday (replication) aka Passive	TalkGen (replication)	TalkGen	DOMsday	25mFlows
Date	04/2025	04/2025	**09/2020**	08/2017	11/2013
Pot. flows (per 1k pages)	45.5	76.6	40.3	116.7	?
Conf. flows (per 1k pages)	1.6	1.5	18.5	72.0	16.2
URL encoding	**Yes**	**Yes**	**No**		

RQ3: Comparison with prior work

Metric	DOMsday (replication) aka Passive	TalkGen (replication)	TalkGen	DOMsday	25mFlows
Date	04/2025	04/2025	09/2020	**08/2017**	11/2013
Pot. flows (per 1k pages)	45.5	76.6	40.3	116.7	?
Conf. flows (per 1k pages)	1.6	1.5	18.5	72.0	16.2
URL encoding	**Yes**	**Yes**	**No**	**No**	

RQ3: Comparison with prior work

Metric	DOMsday (replication) aka Passive	TalkGen (replication)	TalkGen	DOMsday	25mFlows
Date	04/2025	04/2025	09/2020	08/2017	**11/2013**
Pot. flows (per 1k pages)	45.5	76.6	40.3	116.7	?
Conf. flows (per 1k pages)	1.6	1.5	18.5	72.0	16.2
URL encoding	**Yes**	**Yes**	**No**	**No**	**No**

Conclusion

- **194** / 44,480 (0.44%) of analyzed popular web pages still suffer from DOM-XSS
  - ↪ Enables code execution in the browser of a victim
  - ↪ Reduction compared to prior studies largely due to **URL encoding mechanism**

- Our fuzzer improves DOM-XSS confirmed vulnerability detection by 15%
- Symbolic execution generates variations of URLs that trigger new page behavior
  - ↪ rediscovered 26% of confirmed flows that require URL parameters found in the wild
  - ↪ confirmed 20 unique confirmed flows that were not discovered via Fuzzing

More in the paper

- Web archiving method to improve determinism in web analysis
- DSE outperforms other PF-generating approaches like ZAP and Wapiti
- How we overcome several challenges in analyzing the web at scale

DOM-XSS Detection via Webpage Interaction Fuzzing and URL Component Synthesis

- **194** / 44,480 (0.44%) of analyzed popular web pages still suffer from DOM-XSS
  - ↪ Reduction compared to prior studies largely due to **URL encoding mechanism**
- Our fuzzer improves DOM-XSS confirmed vulnerability detection by 15%
- Symbolic execution can find new vulnerabilities by generating variations of URLs

Nuno Sabino, Darion Cassel, Rui Abreu, Pedro Adão, Lujo Bauer and Limin Jia