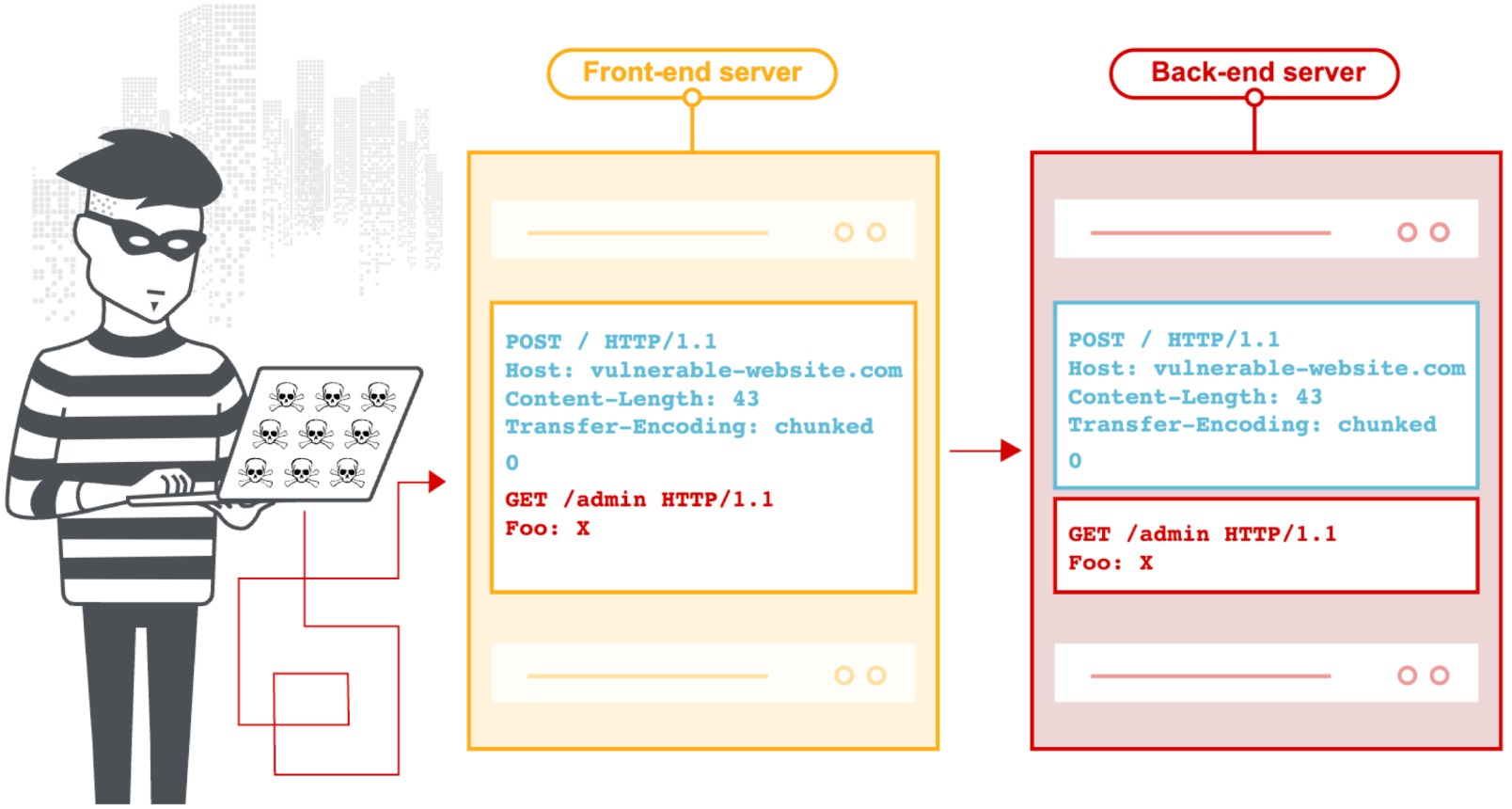
HDiff: A Semi-automatic Framework for Discovering Semantic Gap Attack in HTTP Implementations

Kaiwen Shen*, Jianyu Lu‡, Yaru Yang*, Jianjun Chen*✉,
Mingming Zhang*, Haixin Duan*‡, Jia Zhang*†✉, Xiaofeng Zheng*‡,
*Tsinghua University, ‡QI-ANXIN Technology Research Institute
†Beijing National Research Center for Information Science and Technology
{skw17, zmm18, zxf19}@mails.tsinghua.edu.cn, zhangjia@cernet.edu.cn,
{jianjun, duanhx}@tsinghua.edu.cn, {zeddyu.lu, yangyr17}@gmail.com,

DSN 2022

Background

- **HTTP Request Smuggling (HRS)**
  - An attack that exploits the HTTP parsing discrepancies between two servers(front-end and back-end).
  - Content-Length
  - Transfer-Encoding
  - HTTP/1.1



HDiff

- **Observations**
  - Some HTTP implementations do not follow RFC requirements.
  - RFC defines optional requirements allowing developers to use their discretion.
  - Previous studies are based on ad-hoc manual analysis or only analyze one type of semantic gap attack.

- **Challenges**
  - It is difficult to extract such informal descriptions from RFC documents and convert them to formal invariants.
  - Most semantic gap bugs do not display any explicitly erroneous behavior like crashes or memory corruption bugs and thus are hard to detect.

HDiff

➢ Overview

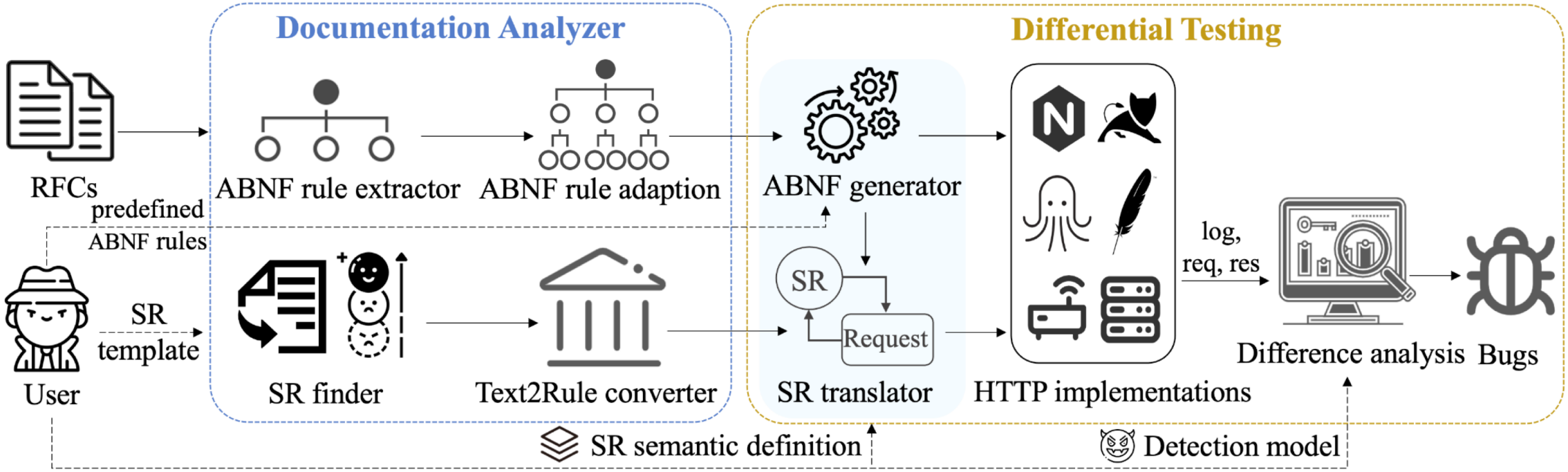


Fig. 3: The Architecture of HDiff.

HDiff

➢ Document Analyzer

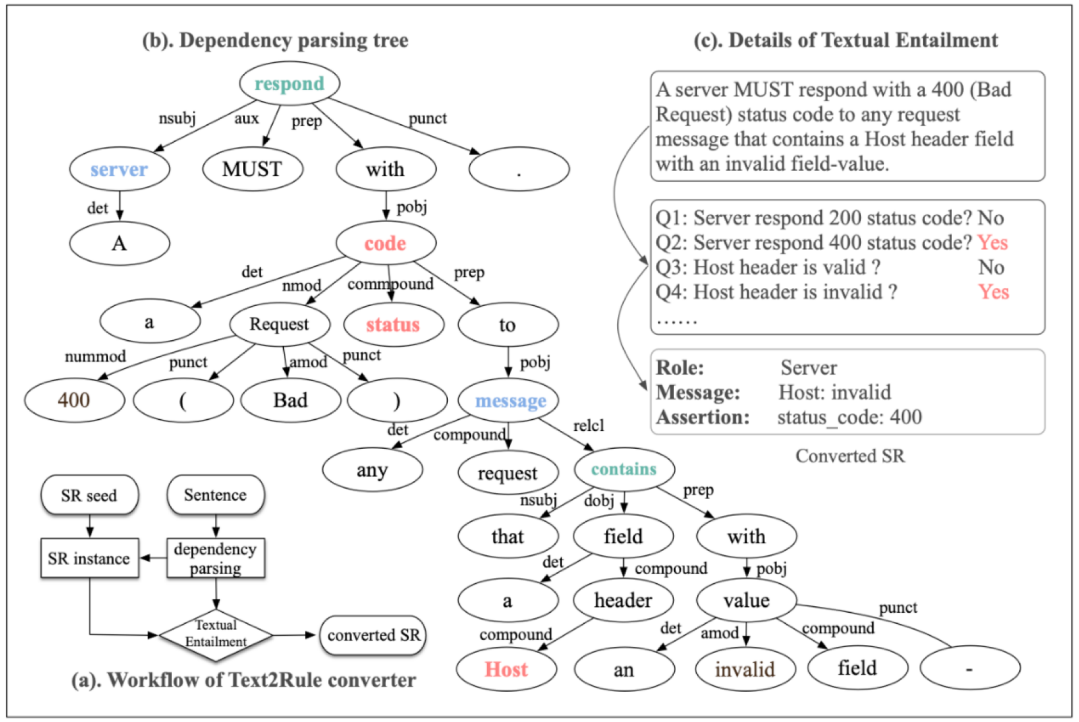


Fig. 4: An example showing details of Text2Rule Converter.

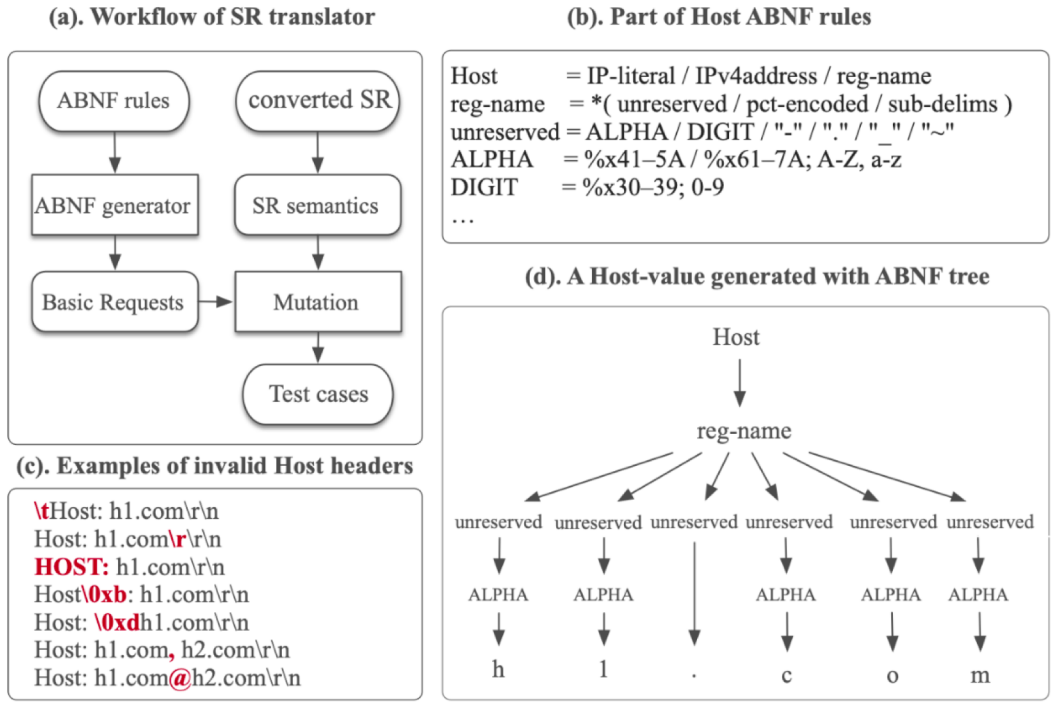


Fig. 5: An example showing details of SR Translator.

HDiff

➢ Evaluation

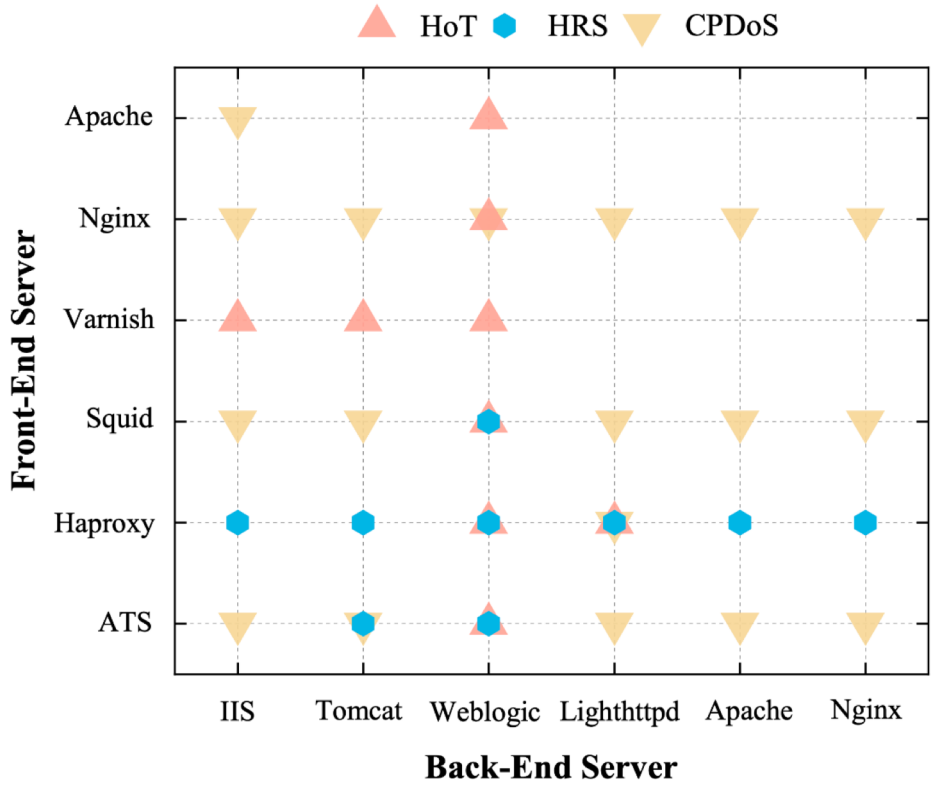


Fig. 7: Server pairs affected by three types of attacks.

TABLE II: Examples of semantic gap attacks found by HDiff.

HTTP Field	Description	Example	Vulnerability
Request-Line	Invalid HTTP-version	**1.1/HTTP**; HTTP/**3-1**; **h**TTP/1.1;	CPDoS
	lower/higher HTTP-version	HTTP/**0.9**;**1.0 with chunked**; HTTP/**2.0**	HRS, CPDoS
	Bad absolute-URI vs Host	**test://**h2.com/?a=1; **h1@h2.com**;	HoT
	Fat HEAD/GET request	HEAD/GET with **message-body**	HRS, CPDoS
Header-field	Invalid CL/TE header	Content-Length: **+6**\r\n Content-Length: **6,9**\r\n Content-Length: **[sc]**9\r\n **[sc]**Transfer-Encoding: chunked\r\n Transfer-Encoding: chunked**\r**\r\n Transfer-Encoding**[sc]**: chunked\r\r\n	HRS
	Multiple CL/TE headers	Content-Length: 10\r\n Content-Length: **0xff**\r\n; Content-Length: 10\r\n Transfer-Encoding**[sc]**: chunked\r\n;	HRS
	Invalid Host header	Host: h1.com**@**h2.com\r\n Host: h1.com**,** h2.com\r\n Host: h1.com**/../../test?**\r\n Host:**[sc]** h1.com\r\n	HoT, CPDoS
	Multiple Host headers	**[sc]**Host: h1.com\r\n Host: h2.com\r\n	HoT
	Hop-by-Hop headers	Connection: close,**Host**\r\n Connection: **Cookie**\r\n	CPDoS
	Expect header	**Expect:** 100-continuce\r\n	HRS, CPDoS
	Obs-fold header	Host: h1.com**\t\n**h2.com\r\n	HoT
	Obsoleted header or value	Transfer-encoding: chunked,**identity**\r\n	HRS, CPDoS
Message-body	Bad chunk-size value	**[bignumber]**\r\nabc\r\n0\r\n; **0xfgh**\r\nabc\r\n9\r\n;	HRS
	NULL in chunk-data	3\r\n**\x00**abc\r\b0\r\n	HRS

1. The symbol **[sc]** represents special characters, including common spaces(e.g., ,\t,\0xb,\0xd), grammatical characters(e.g., {,},<,>,@,',"$) and unicode characters (e.g., \u0000,\uffff,\u202e).

T-Reqs: HTTP Request Smuggling with Differential Fuzzing

Bahruz Jabiyev
Northeastern University
Boston, MA, USA

Steven Sprecher
Northeastern University
Boston, MA, USA

Kaan Onarlioglu
Akamai Technologies
Cambridge, MA, USA

Engin Kirda
Northeastern University
Boston, MA, USA

CCS 2021

Background

- **Related work**
  - Specific attacks : such as cache poisoning, session hijacking.
  - HTTP/2 Request smugging
  - Narrow scope : focus on two HTTP headers (CT, TE)
  - HRS is a system interaction problem. The key aspect of HRS (the HTTP parsing discrepancies) has not been explored.

T-Reqs

➢ Workflow

- Construct the context-free grammar (CFG).
- Generating requests.
- Mutating requests: string mutations and tree mutations.
- Collecting feedback.

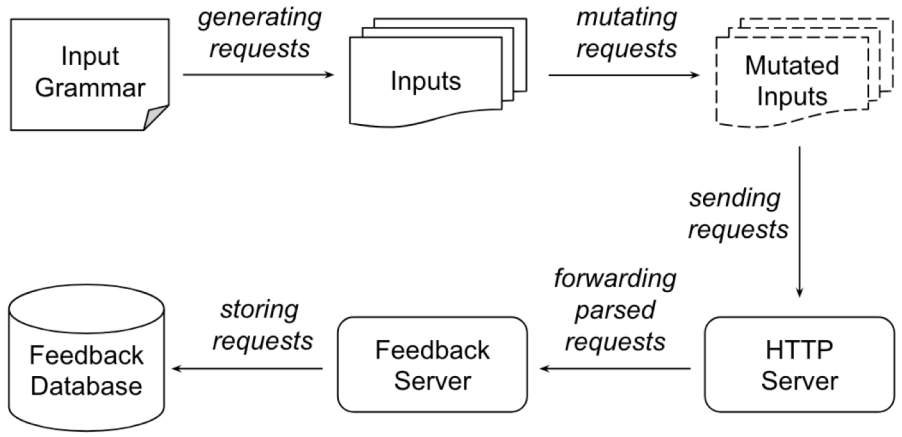


Figure 1: Inputs are generated from a grammar, mutated, and sent to the tested server. The feedback server collects feedback from the requests forwarded by the tested server and stores it for analysis.

```
1 POST //search HTTP/1.1
2 Host: example.com
3 Content-Length: 13
4
5 query=bananas
```

Listing 7: String mutations.

```
1 HTTP /search /search HTTP/1.1
2 Host: example.com
3 Content-Length: 13
4
5 query=bananas
```

Listing 8: Tree mutations.

```
<start> ::= <request>
<request> ::= <line><headers><newline><body>
<line> ::= "POST /search HTTP/1.1\r\n" | "PUT / HTTP/1.1\r\n"
<headers> ::= <host><content-length> | <host>
<host> ::= "Host: example.com\r\n"
<content-length> ::= "Content-Length: 16\r\n"
<newline> ::= "\r\n"
<body> ::= "query=funny+cats" | "query=carrots"
```

Listing 6: Example CFG for a simple HTTP request.

Evaluation

- **Finding Discrepancies**
  - Exposing discrepancies in message body parsing behavior.
  - Three separate experiments on each part of the HTTP request : request line, request headers, request body.

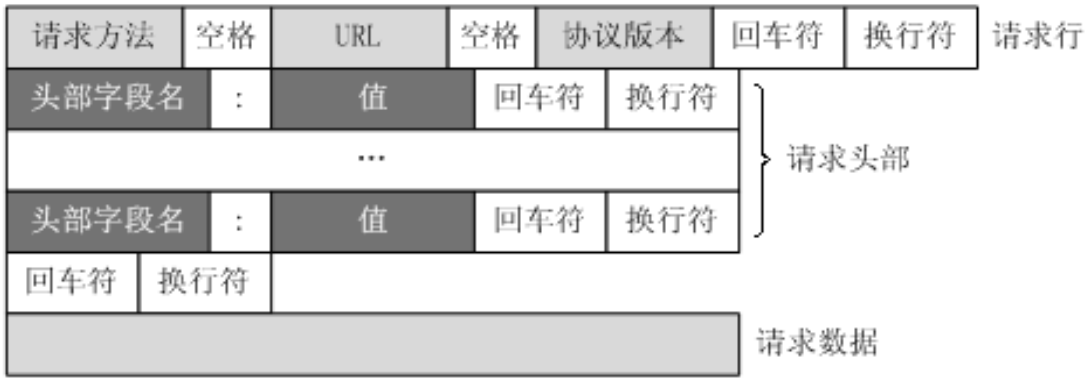


Table 2: General information about experiments.

Name	Duration	# Inputs
Request line	70 hours	8,857K
Request headers	94 hours	3,096K
Request body	72 hours	2,051K

Evaluation

- **Discrepancy Reduction and Classification**
  - Successful Mutation Sets.
  - Classification of Mutation Sets.
- **Request line mutation categories**

Table 5: Experiment success values.

Experiment Name	# Inputs	# Successful
Request line	8,857K	5K
Request headers	3,096K	1K
Request body	2,051K	595K

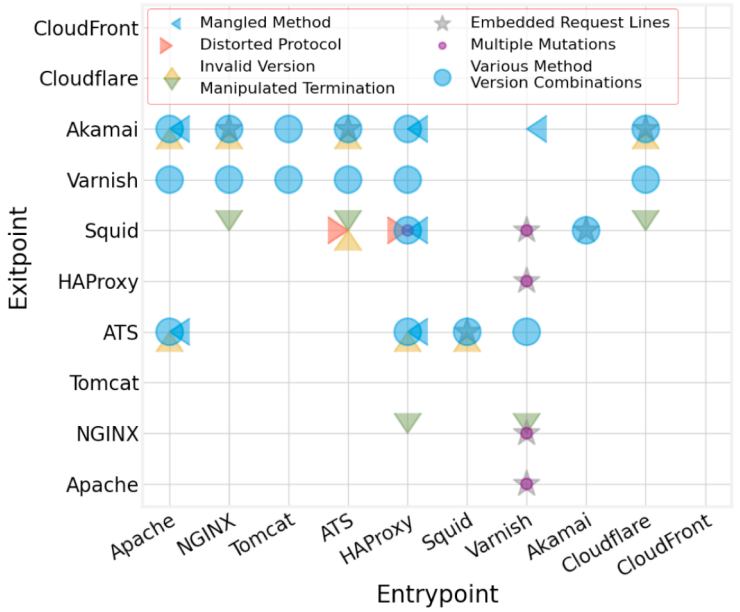


Figure 2: Request line mutation categories affecting server pairs.

Table 6: Examples for each request line mutation category.

Category	Request Line	Entrypoint-Exitpoint
mangled method	`hEAD / HTTP/1.1\r\n`	Apache-Akamai Apache-ATS HAProxy-Akamai HAProxy-ATS
distorted protocol	`GET / HhTTP/1.1\r\n`	ATS-Squid
invalid version	`GET / HTTP/+.19\r\n`	ATS-Akamai ATS-Squid
manipulated termination	`CONNECT / HTTP/1.0 \r\n\r\n`	Varnish-NGINX HAProxy-NGINX
embedded request lines	`OPTIONS / HTTP/OPTIONS / HTTP/0.9\r\n1.1\r\n`	Akamai-Squid
multiple mutations	`GET / HT+P//1.1\r\n`	HAProxy-Squid
various method version combinations	`TRACE / HTTP/1.0\r\n`	Apache-ATS HAProxy-ATS Squid-ATS Varnish-ATS

Evaluation

➢ Request header mutation categories

Figure 3: Request header mutation categories affecting server pairs.

Table 7: Examples for each request header mutation category.

Category	Method ; Request Header	Entrypoint-Exitpoint
distorted header value	`GET;Transfer-Encoding: chunked,\r\n`	Tomcat-Akamai ATS-Akamai HAProxy-Akamai
manipulated termination	`GET;Transfer-Encoding: chunked\r\n␣{Header}:{Value}\r\n`	NGINX-ATS Cloudflare-ATS CloudFront-ATS
expect header	`POST;Expect: 100-continue\r\n`	NGINX-Apache (truncated)
identity encoding	`POST;Transfer-Encoding: identity\r\n`	Cloudflare-Squid CloudFront-Squid (truncated)
v1.0 chunked encoding	`POST;Transfer-Encoding: chunked\r\n`	Apache-Tomcat (truncated)
double transfer-encoding	`POST;Transfer-Encoding: identity\r\nTransfer-Encoding: chunked\r\n`	Cloudflare-Tomcat CloudFront-Tomcat Cloudflare-Akamai (truncated)
various method version combinations	`OPTIONS-0.9;Content-Length:5\r\n`	HAProxy-Squid Akamai-Squid

Evaluation

- **Request body mutation categories**

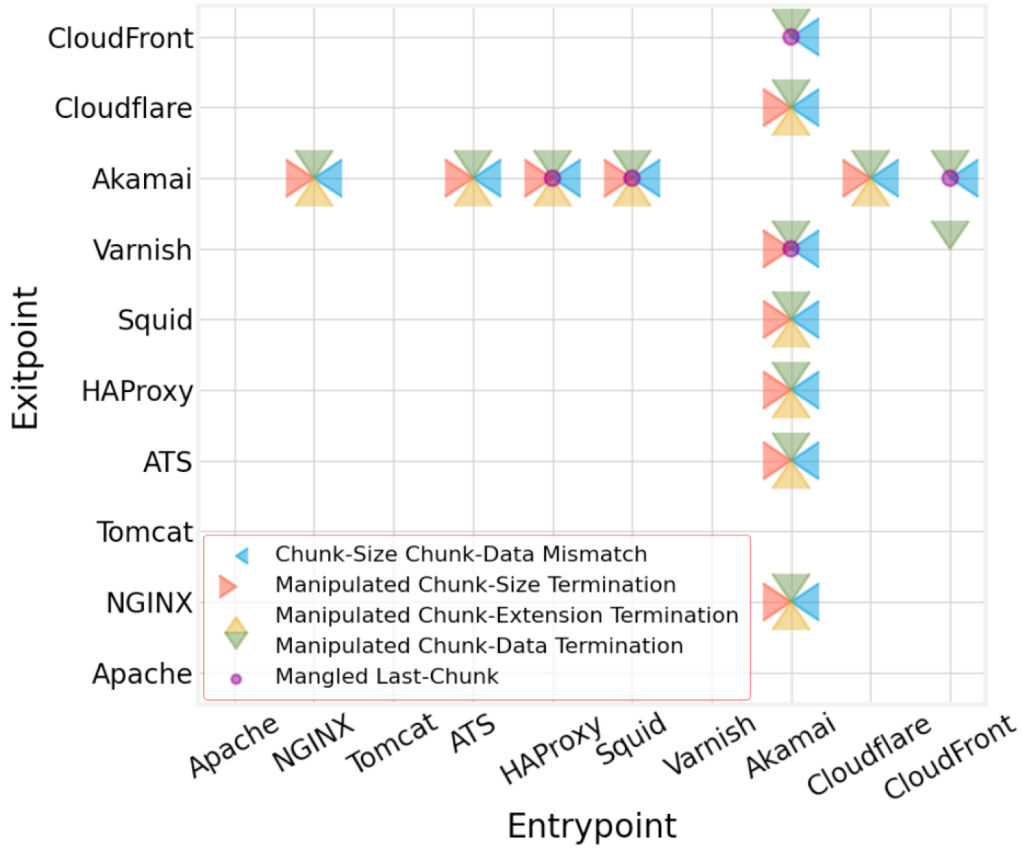


Figure 4: Request body mutation categories affecting server pairs.

Table 8: Examples for each request body mutation category.

Category	Request Body	Entrypoint-Exitpoint
chunk-size chunk-data mismatch	`4\r\nBBBB \r\n0\r\n\r\n`	Akamai-NGINX Akamai-Varnish (truncated)
manipulated chunk-size termination	`4\t\nBBBB \r\n0\r\n\r\n`	Cloudflare-Akamai Squid-Akamai (truncated)
manipulated chunk-extension termination	`4;foo=bar\r\nBBBB \r\n0\r\n\r\n`	Akamai-Cloudflare Akamai-ATS (truncated)
manipulated chunk-data termination	`4\r\nBBBB\r\n4 \r\nBBBB \r\n0\r\n\r\n`	CloudFront-Varnish Akamai-Varnish (truncated)
mangled last-chunk	`4\r\nBBBB \r\n20\r\n\r\n`	HAProxy-Akamai Squid-Akamai (truncated)

Evaluation

➢ Determining Discrepancy HRS Potential

- Set up lab to position every unique server pair as entrypoint and exitpoint on path.
- Craft a smuggler request for every mutation.

```
1 hEAD / HTTP/1.1
2 Host: example.com
3 Content-Length: 2
4
5 A␣
```

Listing 12: Smuggler request.

```
1 POST / HTTP/1.1
2 Host: example.com
3 Content-Length:5
4
5 AAAAA
```

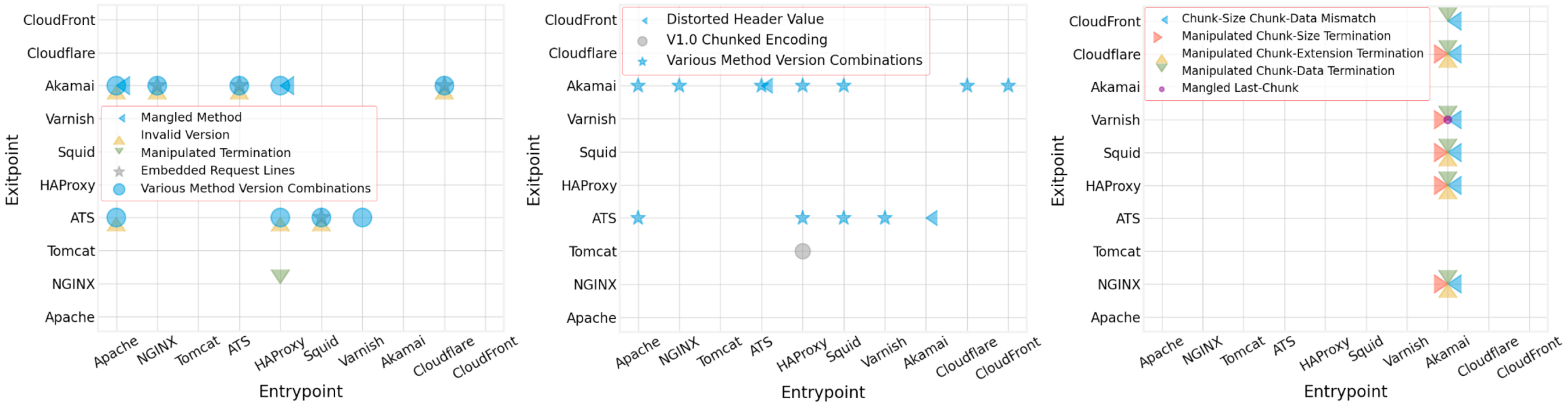
Listing 13: Benign request.

```
1 A␣POST / HTTP/1.1
2 Host: example.com
3 Content-Length:5
4
5 AAAAA
```

Listing 14: Poisoned request.

Evaluation

➢ Determining Discrepancy HRS Potential

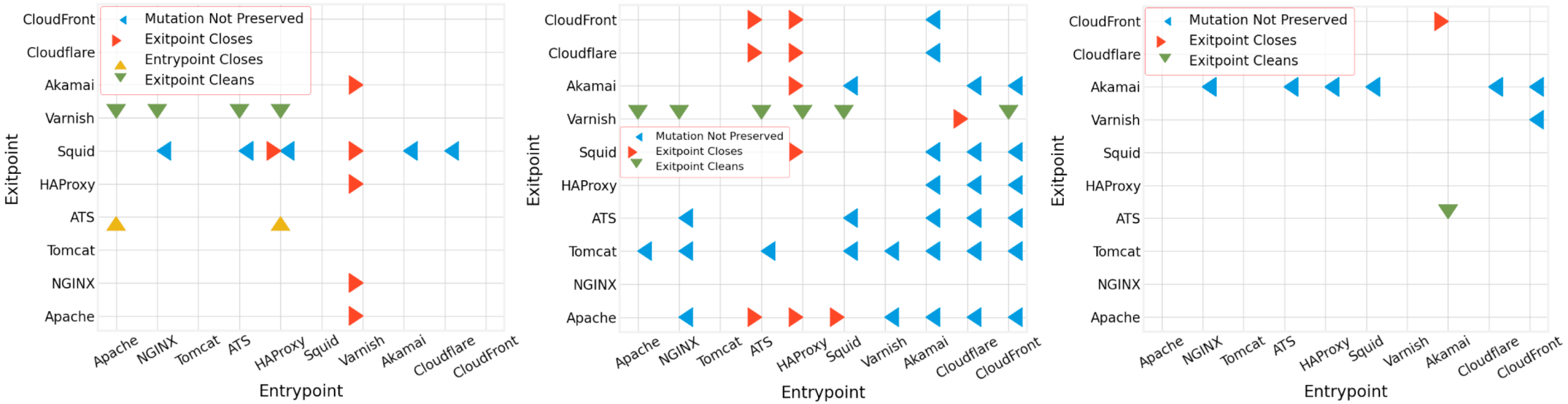


(a) Pairs affected by line mutations. (b) Pairs affected by header mutations. (c) Pairs affected by body mutations.

Figure 5: Server pairs affected by request smuggling.

Evaluation

➢ Determining Discrepancy HRS Potential

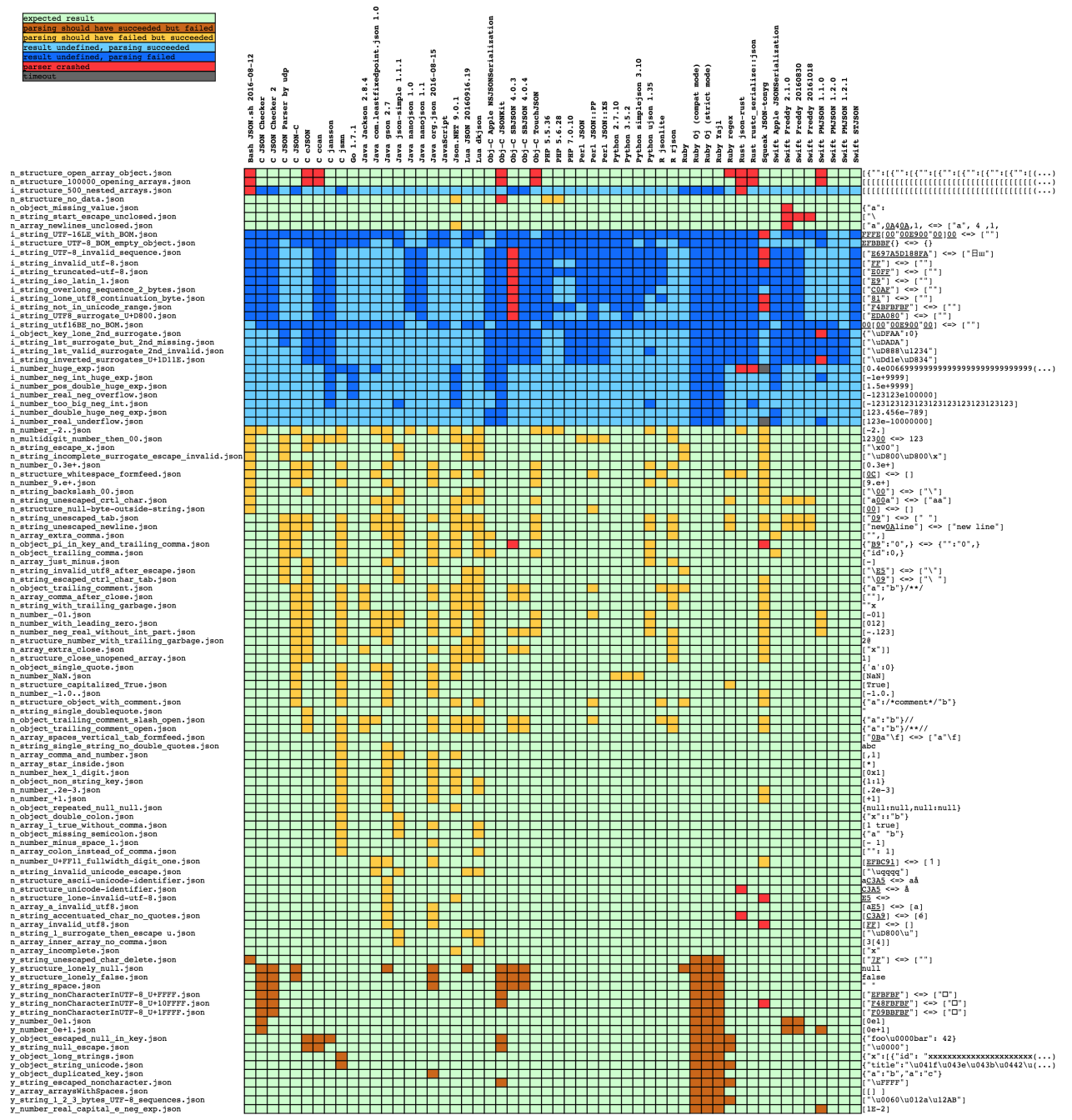


(a) Pairs affected by line mutations. (b) Pairs affected by header mutations. (c) Pairs affected by body mutations.

Figure 6: Failed request smuggling reasons for each server pair.

Recent work

- Discrepancies between parsers
  - JSON
  - XML
  - ZIP
  - HTTP
  - CAN
  - Linux kernel
- Hybridfuzz



Q&A