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

Delegated Presenter : ***Shuai Hao***

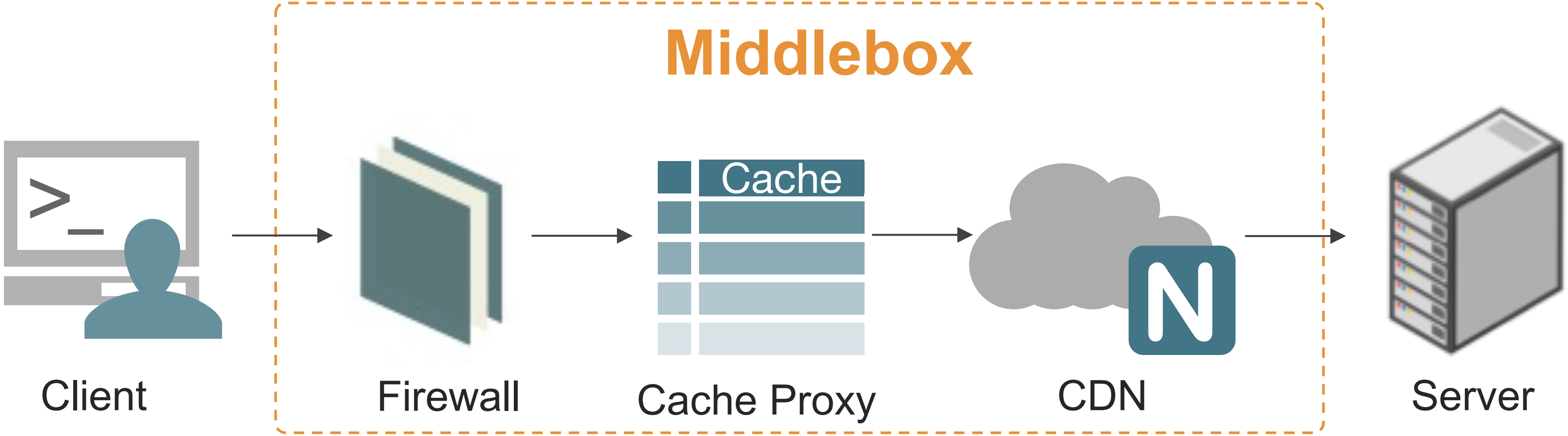
TSINGHUA UNIVERSITY
1911

QI-ANXIN
Leader in next-generation cybersecurity

DSN 2022 - June 28, 2022

Middleboxes are widely deployed with semantic gaps

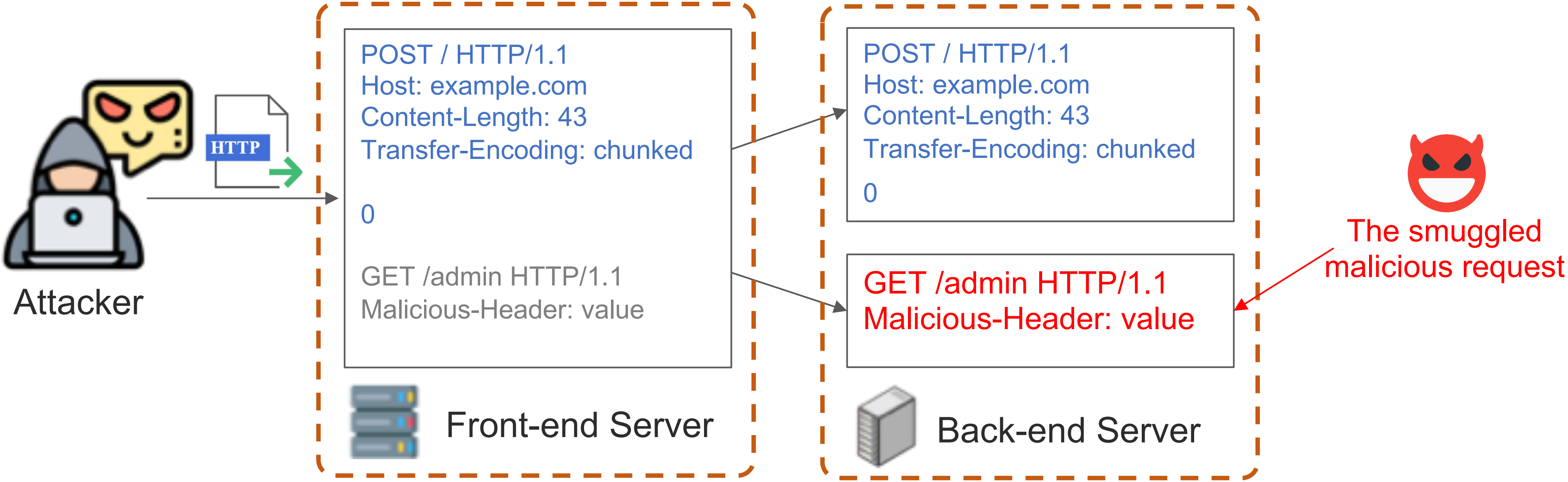
- Middleboxes: intermediate devices deployed for security or performance benefits (e.g., firewall, cache proxy, and CDN).
- Different middleboxes may interpret messages differently, causing semantic gaps.



An end-to-end HTTP request is processed by multiple middleboxes.

A Case Study for Semantic Gap Attack

HTTP Request Smuggling



```
POST / HTTP/1.1
Host: example.com
Content-Length: 43
Transfer-Encoding: chunked

0

GET /admin HTTP/1.1
Malicious-Header: value
```

```
POST / HTTP/1.1
Host: example.com
Content-Length: 43
Transfer-Encoding: chunked
0
```

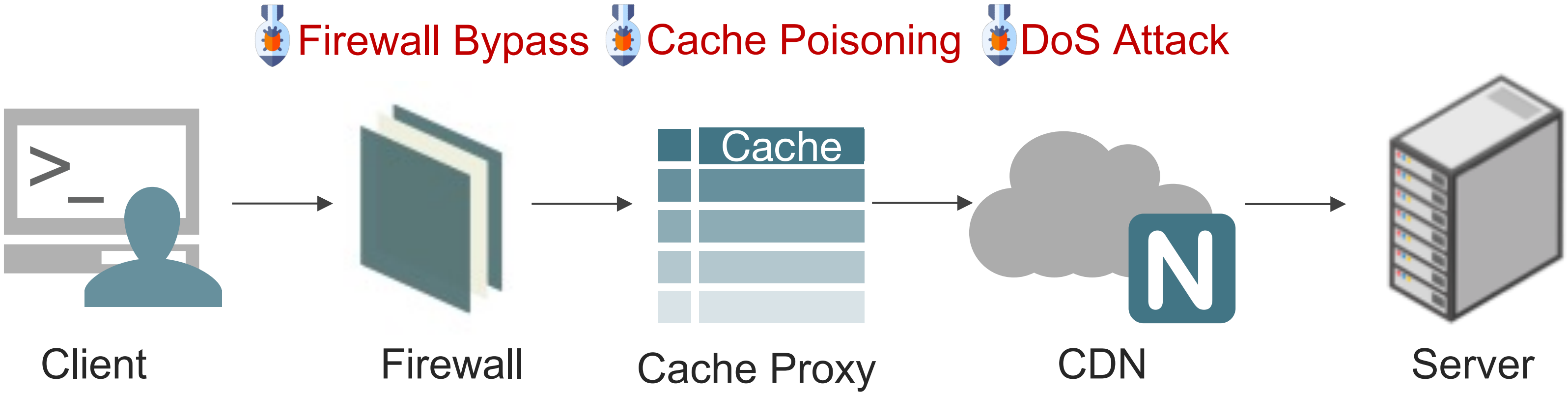
```
GET /admin HTTP/1.1
Malicious-Header: value
```

Semantic gap in parsing more than one Content-Length or Transfer-Encoding header fields to smuggle a hidden request

- Bypass Front-end Security Controls
- Exploit Reflected XSS
- Web Cache Poisoning

Semantic Gap Attack: a Serious Threat to the Internet

- Semantic Gap Attack: Inconsistent Interpretation of an Ambiguous HTTP Request
  - Host of Troubles [CCS’16]
  - HTTP Request Smuggling [BHUSA’19]
  - Cache-Poisoned Denial-of-Service Attack [CCS’19]



Firewall Bypass

Cache Poisoning

DoS Attack

Client

Firewall

Cache Proxy

CDN

Server

Most previous studies relied on fully manual analysis

How to automatically discover semantic gap attacks ?

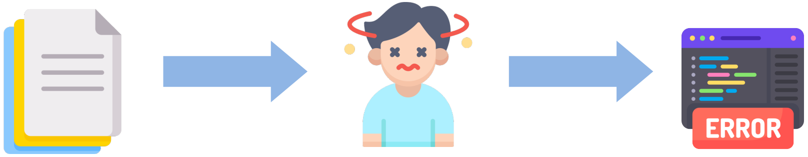
The Root Causes of Semantic Gap Attacks

❖ Implementations not following RFCs:

➢ Intended relaxation for robustness principle

> Be conservative in what you send, be liberal in what you accept.
>
> \- Robustness Principle

➢ Programming mistakes due to the misunderstanding of RFCs

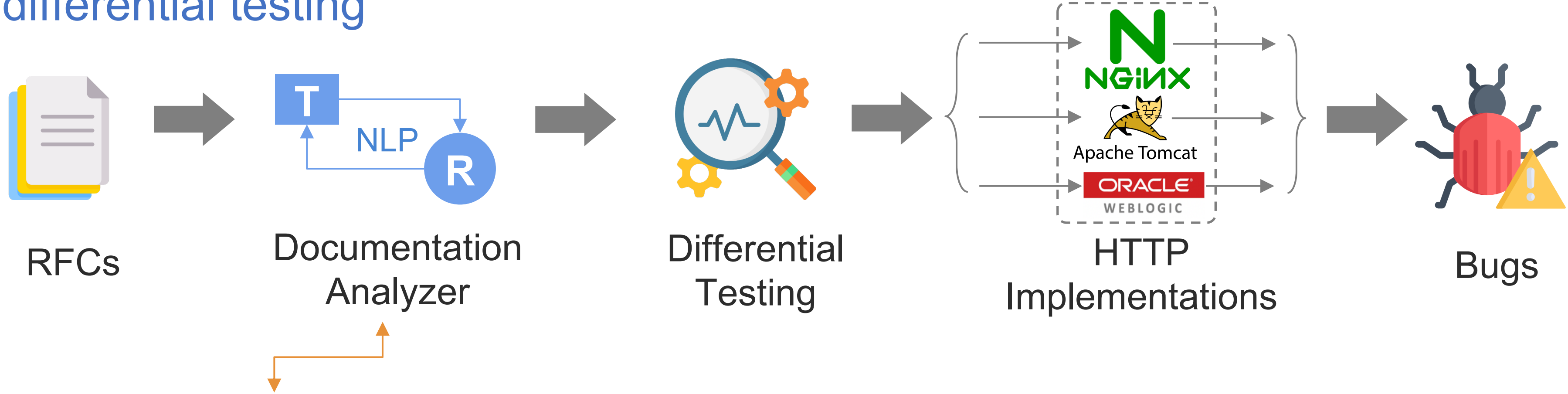


❖ Different implementations of optional requirements:

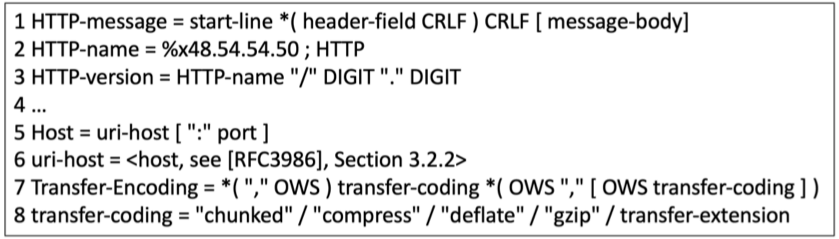
➢ RFC defines optional requirements allowing developers to use their discretion

HDiff: a Semi-automatic Testing Framework

New Detecting Framework: Discovering semantic gaps with RFC-directed differential testing



- **Syntax Rule: ABNF Grammar**

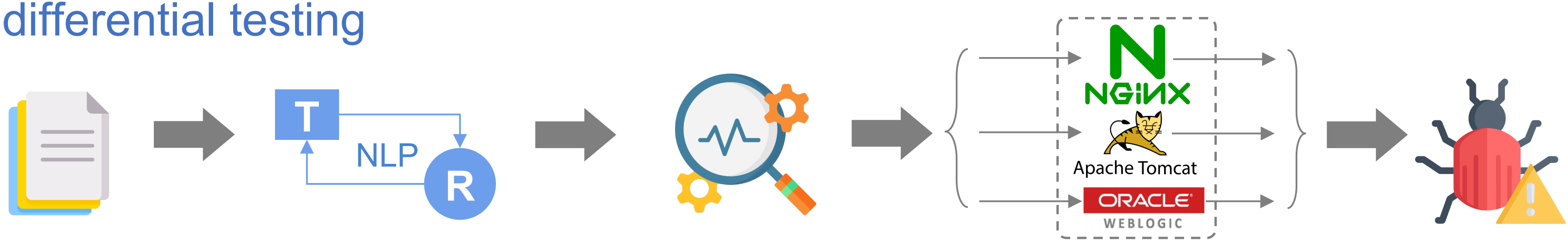


```
HTTP-message = start-line *( header-field CRLF ) CRLF [ message-body]
HTTP-name = %x48.54.54.50 ; HTTP
HTTP-version = HTTP-name "/" DIGIT "." DIGIT
...
Host = uri-host [ ":" port ]
uri-host = <host, see [RFC3986], Section 3.2.2>
Transfer-Encoding = *( "," OWS ) transfer-coding *( OWS "," [ OWS transfer-coding ] )
transfer-coding = "chunked" / "compress" / "deflate" / "gzip" / transfer-extension
```

ABNF rules defining HTTP grammar from RFC 7230.

HDiff: a Semi-automatic Testing Framework

New Detecting Framework: Discovering semantic gaps with RFC-directed differential testing



RFCs

Documentation Analyzer

Differential Testing

HTTP Implementations

Bugs

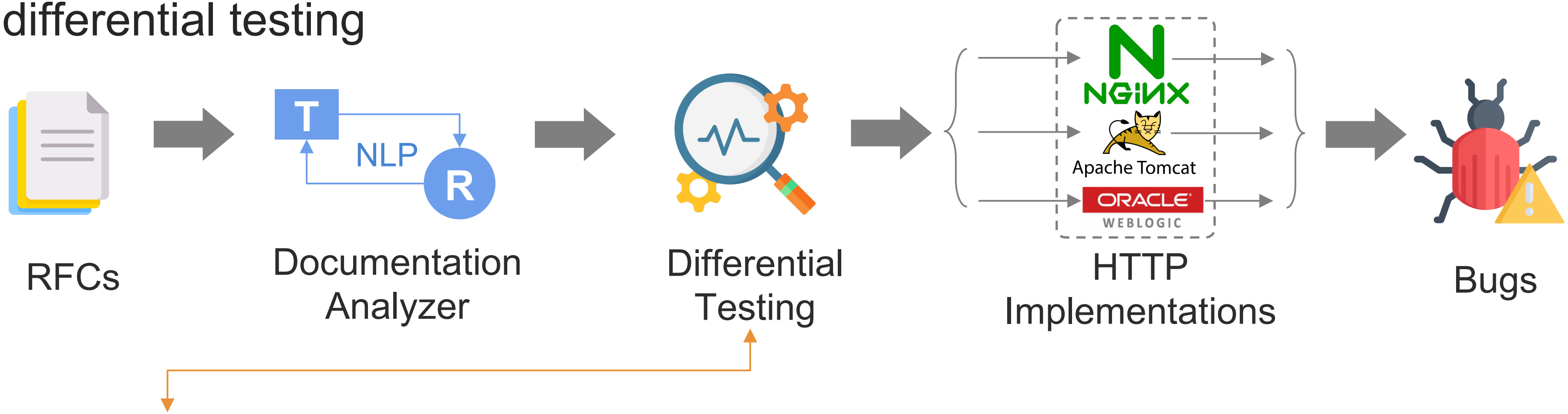
- ➢ Syntax Rule: ABNF Grammar
- ➢ **Semantic Rule: Specification Requirements**
- Informal descriptions to define HTTP semantic actions
- Guide developers to implement the protocol correctly and ensure security

> If a Transfer-Encoding header field is present in a request and the chunked transfer coding is not the final encoding, the server MUST respond with the 400 (Bad Request) status code and then close the connection.
>
> \- RFC 7230

An example of Specification Requirement (SR)

HDiff: a Semi-automatic Testing Framework

New Detecting Framework: Discovering semantic gaps with RFC-directed differential testing



RFCs

Documentation Analyzer

Differential Testing

HTTP Implementations

Bugs

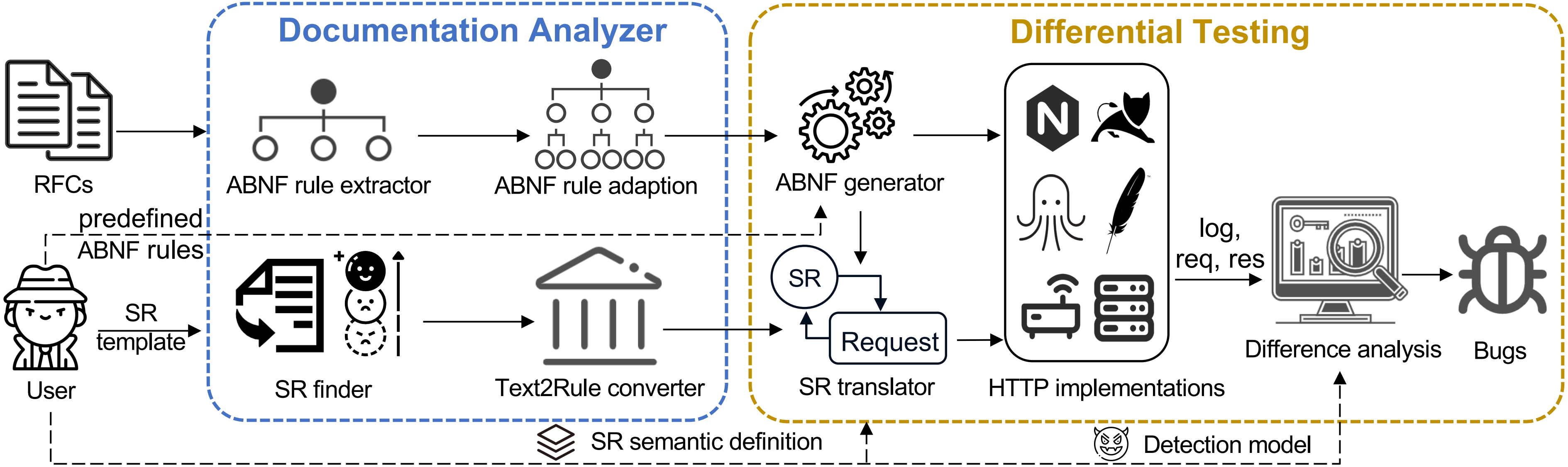
Differential Testing

- **Semantic Metrics:** $HMetrics = \langle uuid, status\_code, host, data, \ldots \rangle$
- **Detecting Bugs:** users can define different detection rules based on HMetrics to discover semantic gap attacks.

HDiff: Design and Implementation

- Documentation Analyzer :
  - Using NLP techniques to extract rules from RFCs

- Differential Testing :
  - Utilizing differential testing to discover semantic gap attacks



The Architecture of HDiff

An End-to-End Example for HTTP Request Smuggling Attack

Research Challenges for Documentation Analyzer

- Automatic extraction of Specification Requirements (SR) from RFC is not easy
  - **Manually extracting SRs needs significant human efforts and is error-prone:**
    - HTTP RFC specifications are lengthy (RFC 7230 includes 89 pages in total)

  - **Traditional regular templates or keyword-based approaches do not work well**
    - RFC documents are described in natural language rather than formal language, in which the sentences are complex and flexible in expression.
    - The same semantics can be expressed in multiple forms, including synonym substitution and grammatical variations (e.g., passive tense)

Step 1: Sentiment-based Specification Requirement Finder

❖ Key Observation:

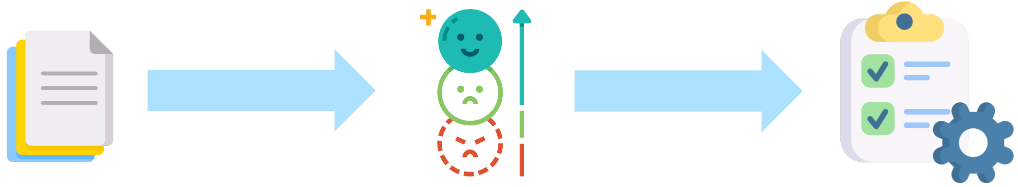
➢ All SRs are characterized by a strong sentiment to stress the constraints

> If a Transfer-Encoding header field is present in a request and the chunked transfer coding is not the final encoding, the server MUST respond with the 400 (Bad Request) status code and then close the connection.
>
> \- RFC 7230

An example of Specification Requirement (SR)

❖ Sentiment-based Specification Requirement Finder:

➢ Automatically identify strong sentiment sentences with potential SRs



RFC Documents

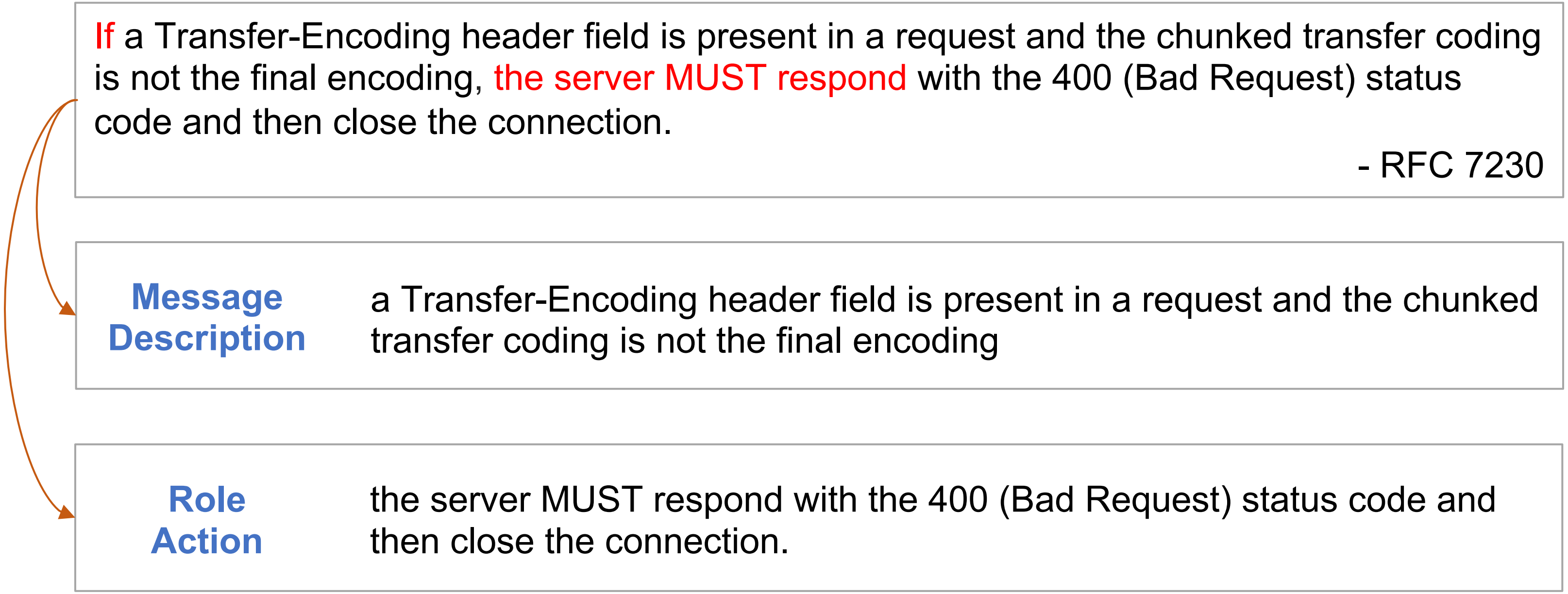
SR Finder

Specification Requirements

Step 2: Text2Rule Converter

- **Key Observation:** All specification requirements tend to follow a specific semantic structure
  - **A message description:** [field-name] header is [represent/valid/invalid/multiple]
  - **A role action:** [role] respond [200/302/400] status code

- **Dependency Tree Analysis:**



If a Transfer-Encoding header field is present in a request and the chunked transfer coding is not the final encoding, the server MUST respond with the 400 (Bad Request) status code and then close the connection.

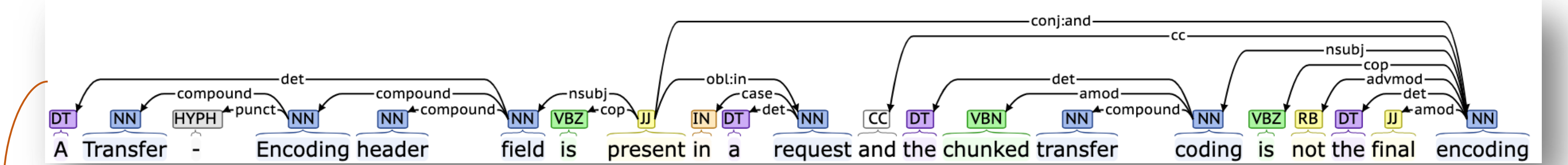
- RFC 7230

Message Description a Transfer-Encoding header field is present in a request and the chunked transfer coding is not the final encoding

Role Action the server MUST respond with the 400 (Bad Request) status code and then close the connection.

Step 2: Text2Rule Converter

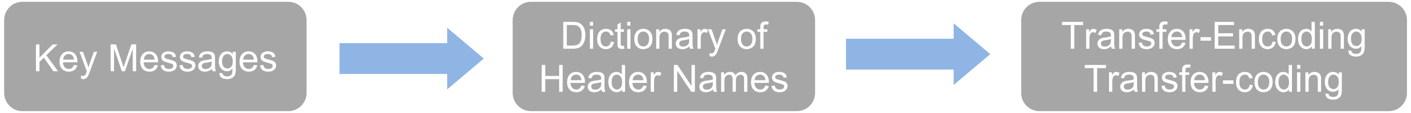
- **Key Observation:** All specification requirements tend to follow a specific semantic structure
  - **A message description:** [field-name] header is [represent/valid/invalid/multiple]
  - **A role action:** [role] respond [200/302/400] status code
- **Part-of-speech tagging:**


conj:and
cc
nsubj
det
compound
obl:in
case
cop
advmod
amod
punct
DT NN HYPH NN NN NN VBZ JJ IN DT NN CC DT VBN NN NN VBZ RB DT JJ NN
A Transfer - Encoding header field is present in a request and the chunked transfer coding is not the final encoding

NN

Message Description

a Transfer-Encoding header field is present in a request and the chunked transfer coding is not the final encoding



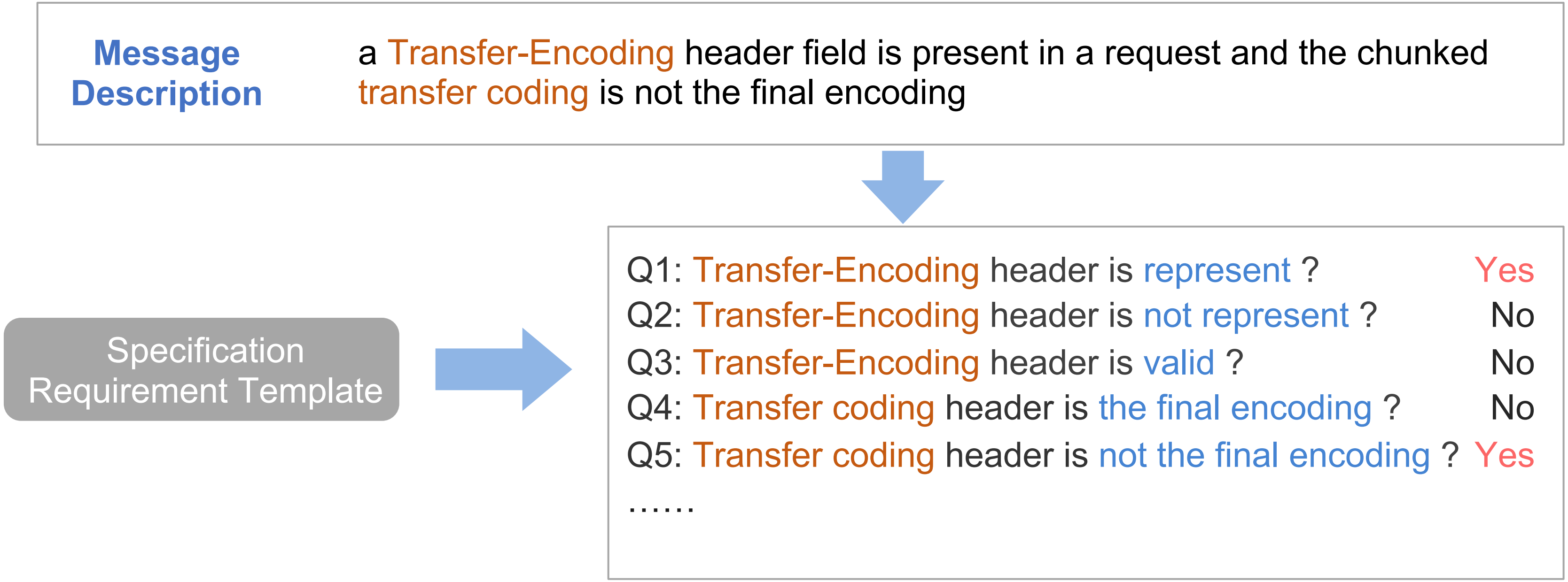
Key Messages → Dictionary of Header Names → Transfer-Encoding Transfer-coding

The header names defined in ABNF rules

The extracted field-name

Step 2: Text2Rule Converter

- ❖ **Key Observation:** All specification requirements tend to follow a specific semantic structure
  - ➢ **A message description:** [field-name] header is [represent/valid/invalid/multiple]
  - ➢ **A role action:** [role] respond [200/302/400] status code
- ❖ **Textual Entailment Analysis:**



Message Description a Transfer-Encoding header field is present in a request and the chunked transfer coding is not the final encoding

Specification Requirement Template

Q1: Transfer-Encoding header is represent ? Yes

Q2: Transfer-Encoding header is not represent ? No

Q3: Transfer-Encoding header is valid ? No

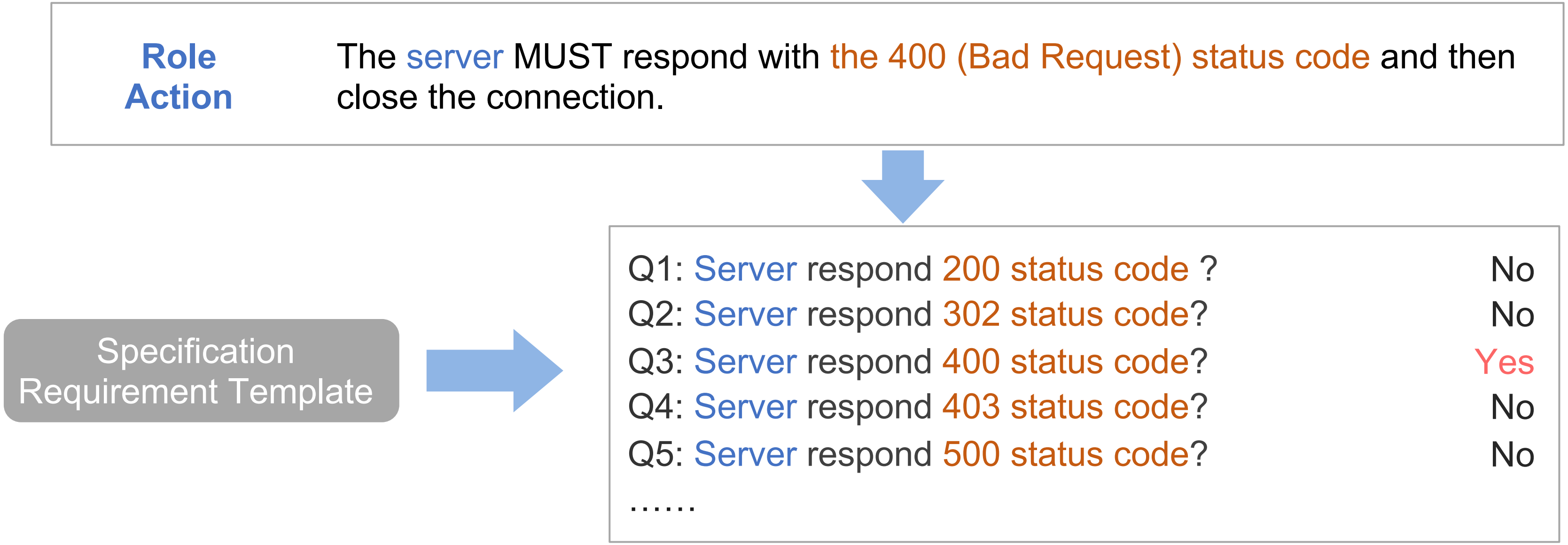
Q4: Transfer coding header is the final encoding ? No

Q5: Transfer coding header is not the final encoding ? Yes

……

Step 2: Text2Rule Converter

- **Key Observation:** All specification requirements tend to follow a specific semantic structure
  - **A message description:** [field-name] header is [represent/valid/invalid/multiple]
  - **A role action:** [role] respond [200/302/400] status code
- **Textual Entailment Analysis:**



Role Action

The server MUST respond with the 400 (Bad Request) status code and then close the connection.

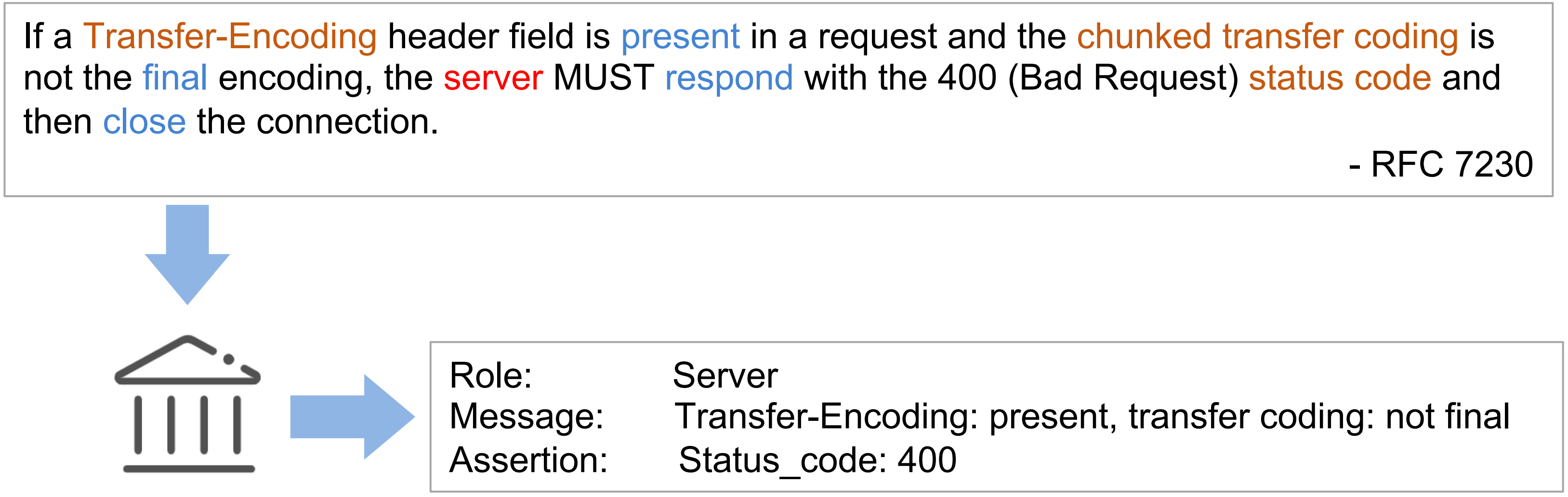
Specification Requirement Template

Q1: Server respond 200 status code ?	No
Q2: Server respond 302 status code?	No
Q3: Server respond 400 status code?	Yes
Q4: Server respond 403 status code?	No
Q5: Server respond 500 status code?	No
……	

Step 2: Text2Rule Converter

- **Key Observation:** All specification requirements tend to follow a specific semantic structure
  - **A message description:** [field-name] header is [represent/valid/invalid/multiple]
  - **A role action:** [role] respond [200/302/400] status code

- **Text2Rule Converter:**



If a Transfer-Encoding header field is present in a request and the chunked transfer coding is not the final encoding, the server MUST respond with the 400 (Bad Request) status code and then close the connection.

- RFC 7230

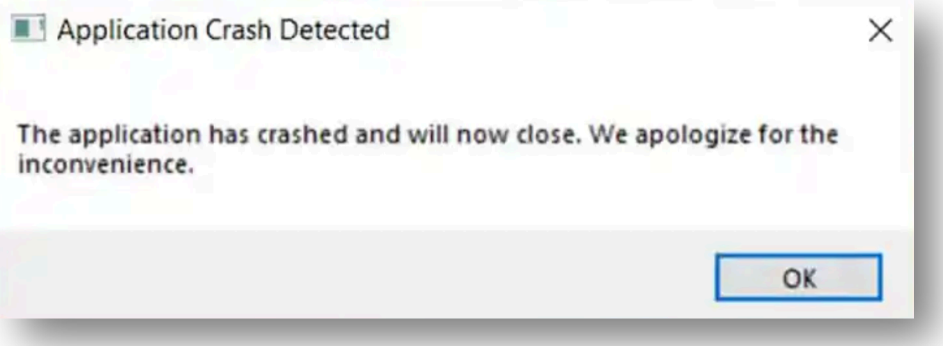
Role: Server
Message: Transfer-Encoding: present, transfer coding: not final
Assertion: Status_code: 400

Text2Rule Converter

The Converted Specification Requirement (SR)

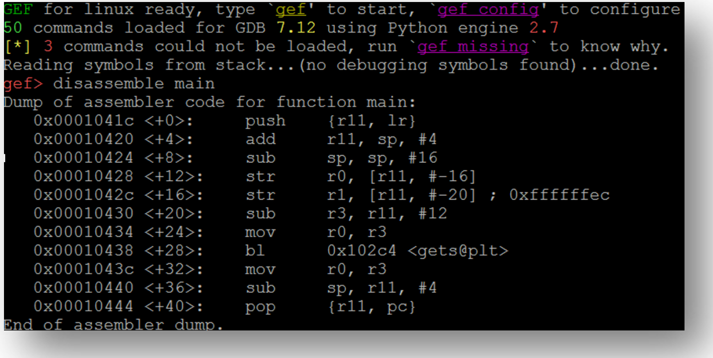
Research Challenges for Differential Testing

- Generating efficient test cases is not easy:
  - Too distorted test cases are easy to be rejected by the target server
  - Randomly generated test cases are not efficient
- Semantic gap bugs are hard to detect :
  - No explicitly erroneous behavior, like crashes or memory corruption



The application has crashed and will now close. We apologize for the inconvenience.

Application Crashes



```
GEF for linux ready, type `gef' to start, `gef config' to configure
50 commands loaded for GDB 7.12 using Python engine 2.7
[*] 3 commands could not be loaded, run `gef missing` to know why.
Reading symbols from stack...(no debugging symbols found)...done.
gef> disassemble main
Dump of assembler code for function main:
   0x0001041c <+0>:     push    {r11, lr}
   0x00010420 <+4>:     add     r11, sp, #4
   0x00010424 <+8>:     sub     sp, sp, #16
   0x00010428 <+12>:    str     r0, [r11, #-16]
   0x0001042c <+16>:    str     r1, [r11, #-20] ; 0xffffffec
   0x00010430 <+20>:    sub     r3, r11, #12
   0x00010434 <+24>:    mov     r0, r3
   0x00010438 <+28>:    bl      0x102c4 <gets@plt>
   0x0001043c <+32>:    mov     r0, r3
   0x00010440 <+36>:    sub     sp, r11, #4
   0x00010444 <+40>:    pop     {r11, pc}
End of assembler dump.
```

Memory Corruption

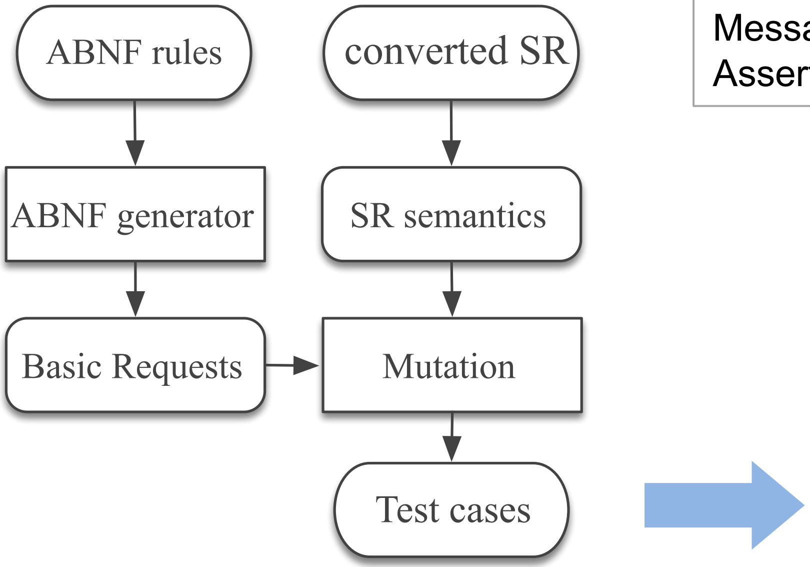
Step 3: Specification Requirement Translator

❖ SR Translator:

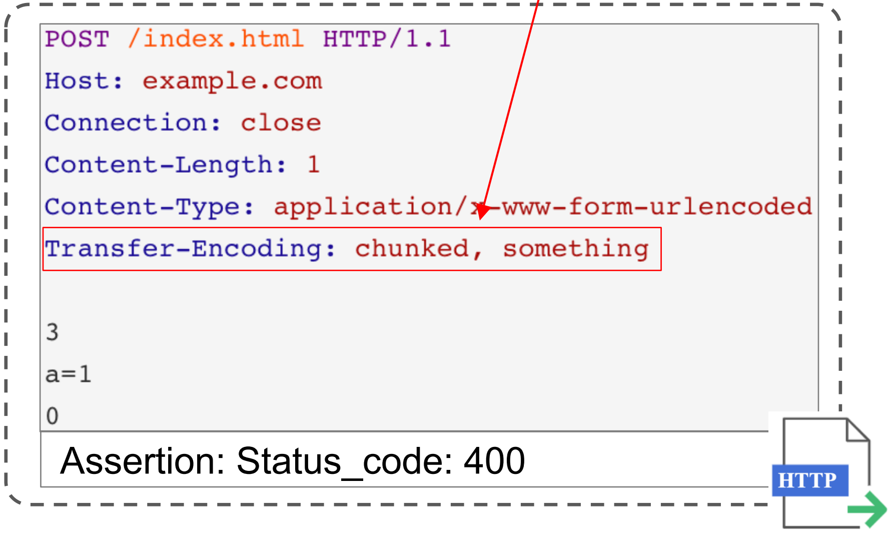
- ➢ Translate the converted specification requirement into test cases with assertions

Role:	Server
Message:	Transfer-Encoding: present, transfer coding: not final
Assertion:	Status_code: 400

The Converted SR



The Workflow of SR translator



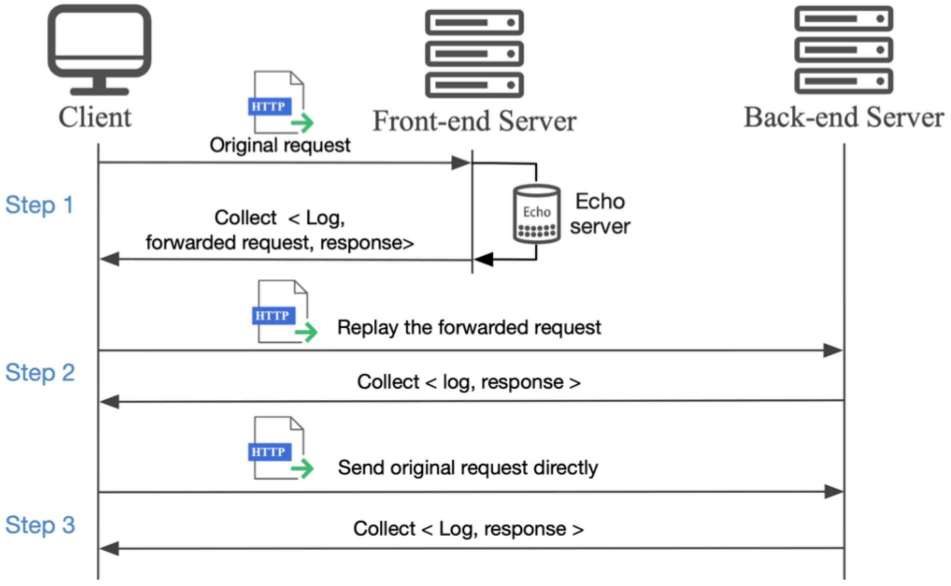
```
POST /index.html HTTP/1.1
Host: example.com
Connection: close
Content-Length: 1
Content-Type: application/x-www-form-urlencoded
Transfer-Encoding: chunked, something


3
a=1
0
```

Assertion: Status_code: 400

An example of Test Cases

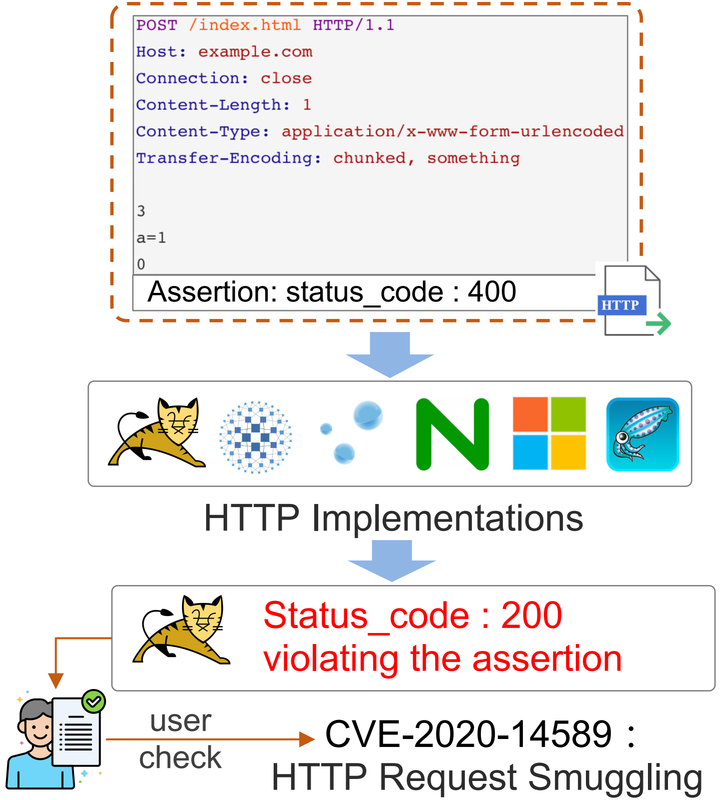
Step 4: Difference Analysis

- **Utilizing difference analysis to discover semantic gap attacks:**
  - Semantic Metrics: $HMetrics = \langle uuid, status\_code, host, data, ...\rangle$



Client
Front-end Server
Back-end Server
HTTP
Original request
Step 1
Echo
Echo server
Collect < Log, forwarded request, response>
HTTP
Replay the forwarded request
Step 2
Collect < log, response >
HTTP
Send original request directly
Step 3
Collect < Log, response >

The Test Workflow



```
POST /index.html HTTP/1.1
Host: example.com
Connection: close
Content-Length: 1
Content-Type: application/x-www-form-urlencoded
Transfer-Encoding: chunked, something


3
a=1
0
```

Assertion: status_code : 400

Findings & Summary

Experiments and Findings

- Extracting 117 specification requirements and 269 ABNF rules from the HTTP 1.1 core specifications (RFC 7230-7235)
- Evaluating the effectiveness of discovering three representative semantic gap attacks in 10 popular HTTP implementations
  - Host of Troubles [CCS'16]
  - HTTP Request Smuggling [BHUSA'19]
  - Cache-Poisoned Denial-of-Service Attack [CCS'19]

traffic server™

VARNISH CACHE

HAPROXY

Apache Tomcat

LIGHTTPD fly light.

NGINX

Squid Proxy

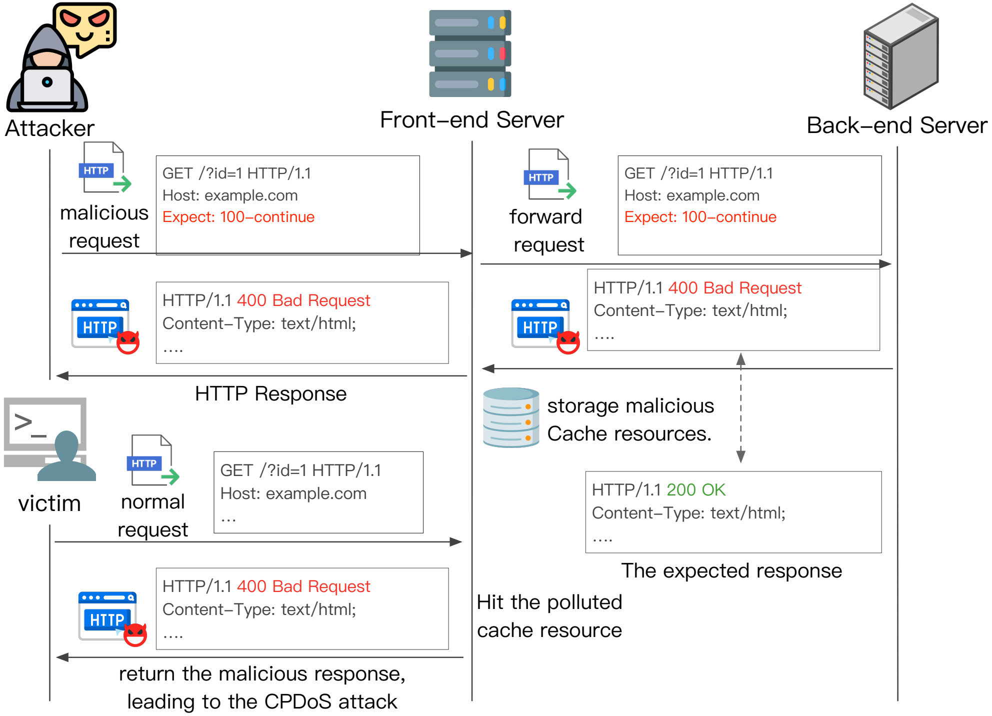
Microsoft IIS

ORACLE WEBLOGIC

Apache HTTP SERVER

Experiments and Findings

- Found 14 vulnerabilities, including three new types of attack payloads.



```
GET /?id=1 HTTP/1.1
Host: example.com
Expect: 100–continue
```

```
GET /?id=1 HTTP/1.1
Host: example.com
Expect: 100–continue
```

```
HTTP/1.1 400 Bad Request
Content–Type: text/html;
....
```

```
HTTP/1.1 400 Bad Request
Content–Type: text/html;
....
```

```
GET /?id=1 HTTP/1.1
Host: example.com
...
```

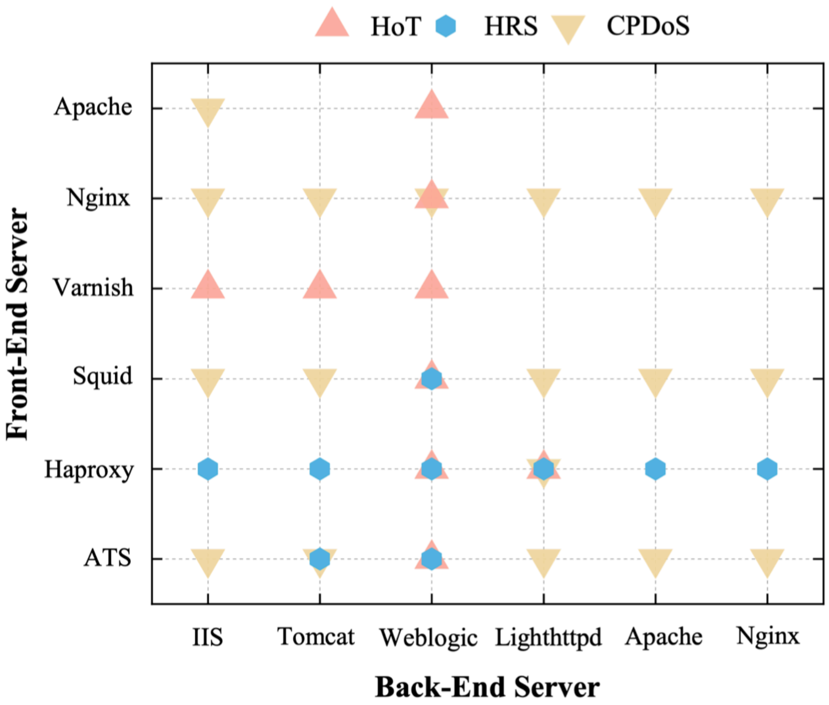
```
HTTP/1.1 200 OK
Content–Type: text/html;
....
```

```
HTTP/1.1 400 Bad Request
Content–Type: text/html;
....
```

Case Study: the inconsistent processing of Expect header leading to the CPDoS attack

Experiments and Findings

- Found 29 exploitable server pairs
- Obtained 7 new CVEs



Apache Tomcat	CVE-2019-17569, CVE-2020-1935
Microsoft IIS	CVE-2020-0645
	CVE-2020-14588
ORACLE WEBLOGIC	CVE-2020-2867, CVE-2020-14589
traffic server™	CVE-2020-1944

Summary

- **New Detecting Framework:**
  - HDiff, a novel detecting framework, exploring semantic gap attacks in HTTP implementations
- **New Findings:**
  - Finding 14 vulnerabilities and 29 vulnerable server pairs in 10 popular HTTP implementations
- **Responsible Disclosure:**
  - Receiving 7 new CVEs from IIS, Apache, Tomcat, and Weblogic

ORACLE WEBLOGIC

Microsoft IIS

traffic server™

Apache Tomcat

Thank you!   *Q & A*

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

Delegated Presenter: ***Shuai Hao***
(Old Dominion University)

DSN 2022 - June 28, 2022

Tsinghua University   Qi An Xin Group Corp