The Pennsylvania State University
The Graduate School

HTTP REQUEST SMUGGLING – A SURVEY FROM HTTP/1.1 TO HTTP/2.0

A Project Proposal in
IST 554 Network Management and Security

by
Vaibhav Singh

Under the guidance of
Dr. Peng Liu
Professor, Dept of IST

Submitted in Partial Fulfillment
of the Requirements
for the Degree of
Master of Science

TABLE OF CONTENTS

Chapter 1 - BACKGROUND

Request Smuggling (*also known as Desync attack*)

HTTP request smuggling is a technique for interfering with the way a web site processes sequences of HTTP requests that are received from one or more users. Request smuggling vulnerabilities are often critical in nature, allowing an attacker to bypass security controls, gain unauthorized access to sensitive data, and directly compromise other application users.

What happens in a Request Smuggling attack?

Today's web applications frequently employ chains of HTTP servers between users and the ultimate application logic. Several requests are sent one after the other on the same backend network connection. For example, if you look at Amazon Dynamo's Service-oriented architecture then you will notice that client requests are first split based on various page rendering components and are then passed on to a content switching servers which further route the request to load balancers and eventually the request reaches various backend applications. As a result, there's an opportunity for an attacker to change the interpretation of the request during its journey to the backend.

In this situation, it is crucial that the front-end and back-end systems agree about the boundaries between requests. Otherwise, an attacker might be able to send an ambiguous request that gets interpreted differently by the front-end and back-end systems.

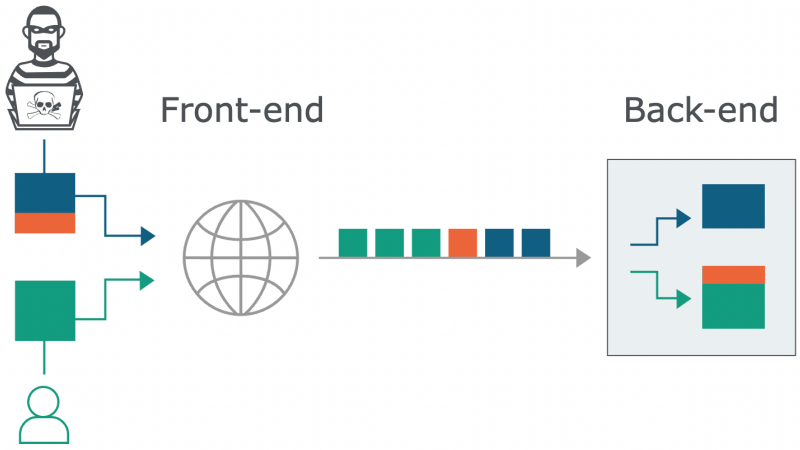

Figure 1.1

Here, the attacker causes part of their front-end request to be interpreted by the back-end server as the start of the next request. It is effectively prepended to the next request, and so can interfere with the way the application processes that request. This is a *request smuggling attack*, and it can have devastating results.

Chapter 2 - SCOPE

"HTTP request smuggling was first documented in 2005, and recently re-popularized by PortSwigger's research on the topic."

Initially, I had planned to try for a simulation driven project. Later, I realized that there are lots of references online which details Request Smuggling attack variants and to learn the nitty-gritties of this attack, I need to first organize this information such that I can eventually try for various kinds of simulations. Hence, in this survey, I am trying to consolidate information about Request Smuggling starting from the point where it was first documented to the most recent version of the attack. This survey will try to mention details about the attack and its types and then provide a list of ways in which attackers have approached the idea of this attack to come up with newer versions.

2.1 What are in the scope of this survey?

In this survey, I am mainly trying to cover Request Smuggling in the following aspects –

- What is Request Smuggling?
- Standard types of Request Smuggling?
  - CL.TE, TE.CL, TE.TE
- Abusing HTTP/1.1 Request Smuggling – Use cases
- Request Smuggling reborn
  - https://portswigger.net/research/http-desync-attacks-request-smuggling-reborn
- Newly discovered H2C request smuggling attack (HTTP/2.0)
  - https://bishopfox.com/blog/h2c-smuggling-request
- Industry stories
  - F5, Cisco, Citrix, Red Hat blogs
- Tools to try Desync attack – An overview

2.2 What are out of the scope of this survey?

- Hands–on simulations
  - Given the variety of ways in which Desync attacks can be performed, I have kept this as future task.
- Study on Defense mechanisms for HTTP/1.1
- Explore custom mitigations from top vendors
  - Various companies have incorporated their own custom version of defense against this attack.
- Explore the possibilities in HTTP/3.0
  - Recent work briefly mentions this but since it's an ongoing area of research, so I haven't included it for this course project.

Chapter 3 – CLASSIFICATION CRITERIA

3.1 System Model

We know that whenever HTTP requests originating from a client pass through more than one entity that parses them, there is a good chance that these entities are vulnerable to HTTP Request Smuggling (HRS). So, at the most abstract level, in a HRS attack we usually have –

- An attacker, and,
- Two attacked HTTP devices
    - A Client or a Front-end forwarding server
    - A Web Server
- Intermediate devices

HRS sends multiple, specially crafted HTTP requests that cause the two attacked devices to see different sets of requests, allowing the hacker to smuggle a request to one device without the other device being aware of it.

To make this attack work, the intermediate devices play a role for setting up the playground for various variants of this attack. Some common device setups are –

- a web cache server deployed between the client and the web server
- a firewall protecting the web server
- a web proxy server (not necessarily caching) deployed between the client and the web server.

The *protocol* being exploited for the attack is *HTTP*.

In a deeper sense, HRS tries to exploit small discrepancies in the way HTTP devices deal with *illegitimate or borderline* requests. Most HTTP request smuggling vulnerabilities arise because the HTTP specification provides two different ways to specify where a request ends: the *Content-Length header and the Transfer-Encoding* header. Attackers exploit the subtle details in the specification related to these fields to launch this attack.

This attack has been highlighted from HTTP/1.1 to HTTP/2.0 and research on HTTP/3.0 is in progress.

An abstract design has been highlighted in *Figure 1.1* in the Background section of the proposal.

3.2 Classification Criteria

HRS falls under the broad category of *Bypass Mitigation* attacks.

- Mitigation bypass is a process of fighting against and breaking mitigation measures in an environment where mitigations are enabled for the ultimate end of arbitrary code execution.

As per recent attack trends, bypass attacks can be further classified into following categories –

1. Intermittent, Short-Duration, High Frequency Attacks
2. Small-Sized, Multiple Destination IP Attacks
3. Attack Traffic Masquerading as Legitimate Traffic

HRS falls under the third category where the attacker tries to masquerade as a legitimate traffic.

- Within this category, attackers create custom traffic patterns for various layers to resemble that of normal traffic.

HRS falls under Application layer (HTTP) custom attacks within this category and are considered critical because they allow threat actors to bypass security controls, gain unauthorized access to sensitive data, and directly compromise other application users.

Chapter 4 – REFERENCES

https://brightsec.com/blog/http-request-smuggling-hrs/
https://portswigger.net/web-security/request-smuggling
https://www.imperva.com/learn/application-security/http-request-smuggling/
https://snyk.io/blog/demystifying-http-request-smuggling/
https://book.hacktricks.xyz/pentesting-web/http-request-smuggling

Request smuggling reborn
https://portswigger.net/research/http-desync-attacks-request-smuggling-reborn
https://portswigger.net/research/http-desync-attacks-what-happened-next

h2c request smuggling
https://bishopfox.com/blog/h2c-smuggling-request
https://blog.desdelinux.net/en/encontraron-una-nueva-version-del-ataque-http-request-smuggling/
https://portswigger.net/research/http2

Mitigation bypass
https://nsfocusglobal.com/what-you-should-know-about-mitigation-bypass/
https://blog.nexusguard.com/could-bypass-attacks-become-the-new-normal

Attack use cases
https://portswigger.net/web-security/request-smuggling/exploiting

CVEs
https://cwe.mitre.org/data/definitions/444.html

Videos
Practical HTTP Header Smuggling: Sneaking Past Reverse Proxies to Attack AWS and Beyond
HTTP Request Smuggling - NetGear Routers 0-Day, The Most Brute Forced Passwords, GoDaddy Breach
albinowax - HTTP Desync Attacks: Smashing into the Cell Next Door - DEF CON 27 Conference
HTTP Desync Attacks: Request Smuggling Reborn