Web Security

Week 11 - Local HTTP Server Security

Old Dominion University
Department of Computer Science
CS 495/595 Spring 2022
Michael L. Nelson <mln@cs.odu.edu>

2022-04-04

The most dangerous code you run every day

```
// Anyone can connect to the server at http://<your-ip>:4000
server.listen(4000)

// Only your device can connect to the server
server.listen(4000, '127.0.0.1')
```

Zoom Zero Day: 4+ Million Webcams & maybe an RCE? Just get them to visit your website!

A vulnerability in the Mac Zoom Client allows any malicious website to enable your camera without your permission. The flaw potentially exposes up to 750,000 companies around the world that use Zoom to conduct day-to-day business.

Jonathan Leitschuh Follow
Jul 8 · 16 min read

From 2019

Responsible disclosure: the researchers gave Zoom 90 days to fix the problem and they did nothing, so they published this blog post.

CVE-Numbers

- DOS Vulnerability — Fixed in Client version 4.4.2 — CVE-2019–13449
- Information Disclosure (Webcam) — Unpatched —CVE-2019–13450

UPDATE — July 9th (am)

As far as I can tell this vulnerability also impacts Ringcentral. Ringcentral for their web conference system is a white labeled Zoom system.

https://infosecwriteups.com/zoom-zero-day-4-million-webcams-maybe-an-rce-just-get-them-to-visit-your-website-ac75c83f4ef5?gi=f80c27935bec

Zoom zero day

"This vulnerability allows any website to forcibly join a user to a Zoom call, with their video camera activated, without the user's permission"

"On top of this, this vulnerability allowed any webpage to DOS (Denial of Service) a Mac by repeatedly joining a user to an invalid call"

"Additionally, if you've ever installed the Zoom client and then uninstalled it, you still have a localhost web server on your machine that will happily re-install the Zoom client for you, without requiring any user interaction on your behalf besides visiting a webpage. This re-install 'feature' continues to work to this day"

Zoom zero day

"Let me start off by saying having an installed app that is running a web server on my local machine with a totally undocumented API feels incredibly sketchy to me"

"Secondly, the fact that any website that I visit can interact with this web server running on my machine is a huge red flag for me as a Security Researcher"

"Having every Zoom user have a web server that accepts HTTP GET requests that trigger code outside of the browser sandbox is painting a huge target on the back of Zoom"

Demo: How does a site communicate with a local HTTP server?

- With the following local HTTP server:

```
const COMMAND = 'open /System/Applications/Dictionary.app'

app.get('/', (req, res) => {
  exec(COMMAND, err => {
    res.set('Access-Control-Allow-Origin', '*')
    if (err) res.status(500).send(err)
    else res.status(200).send('Success')
  })
})
```

- Any site can send a GET request to **http://localhost:4000** to launch the Dictionary application

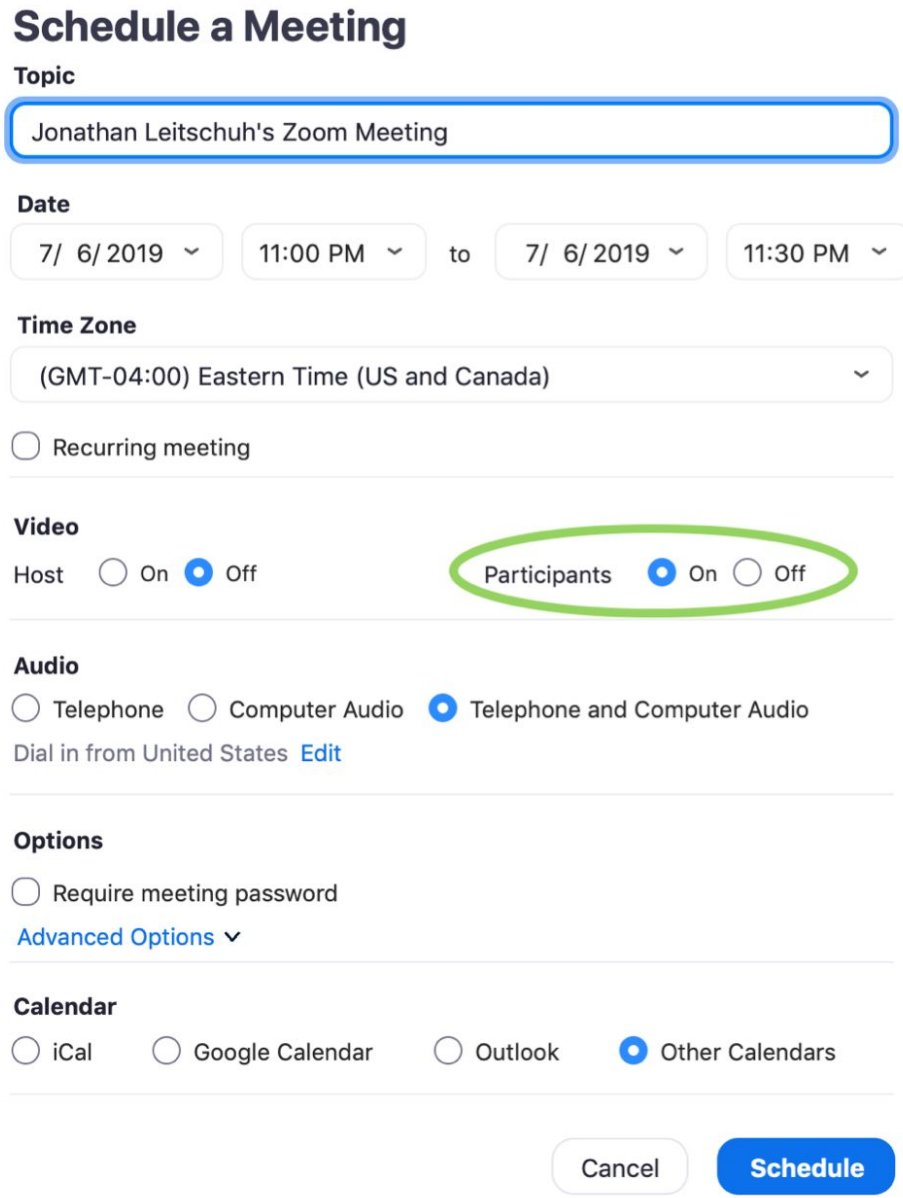
Demo: How many servers are running on your computer?

```
$ lsof -i -P | grep -i "listen"
BlueJeans   540  mln   15u  IPv4 0xc62afb1fdf7a3d27     0t0  TCP
localhost:18170 (LISTEN)
node     655  mln   23u  IPv6 0xc62afb1fd58657df    0t0  TCP *:5001 (LISTEN)
$
```

TrendMicro Antivirus on Windows local HTTP server Remote Code Execution (RCE)

- Local HTTP server was vulnerable to RCE from any site
  - See Google Project Zero issue: https://bugs.chromium.org/p/project-zero/issues/detail?id=693&redir=1

```
x = new XMLHttpRequest()
x.open("GET", "https://localhost:49155/api/openUrlInDefaultBrowser?url=c:/windows/system32/calc.exe", true);
try { x.send(); } catch (e) {};
```


Schedule a Meeting
Topic
Jonathan Leitschuh's Zoom Meeting
Date
7/ 6/ 2019
11:00 PM
to
7/ 6/ 2019
11:30 PM
Time Zone
(GMT-04:00) Eastern Time (US and Canada)
Recurring meeting
Video
Host On Off
Participants On Off
Audio
Telephone Computer Audio Telephone and Computer Audio
Dial in from United States Edit
Options
Require meeting password
Advanced Options
Calendar
iCal Google Calendar Outlook Other Calendars
Cancel
Schedule

Zoom UI denial-of-service

Zoom UI denial-of-service

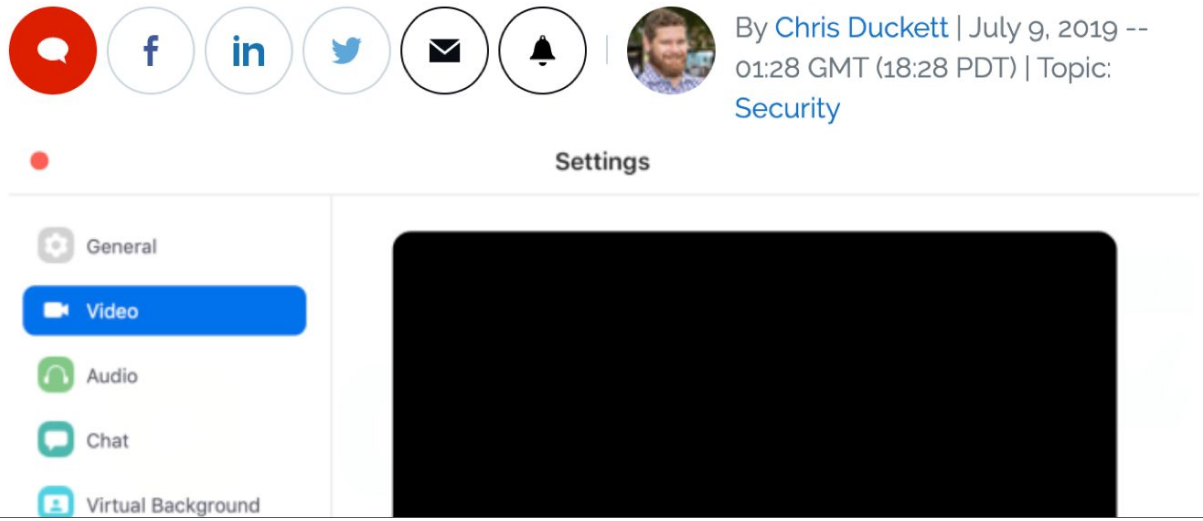
```
// It's actually better if this number isn't a valid zoom conference number
const confNum = '694138052'

setInterval(() => {
  new Image().src =
    'http://localhost:19421/launch?action=join&confno=' +
    confNum + '&' + Date.now()
}, 1)
```

Browser loses focus briefly on each request. Repeat the requests often enough and the user can't regain focus.

Zoom defends use of local web server on Macs after security report

Local web server will also reportedly reinstall Zoom if a user removes the application and joins a meeting.



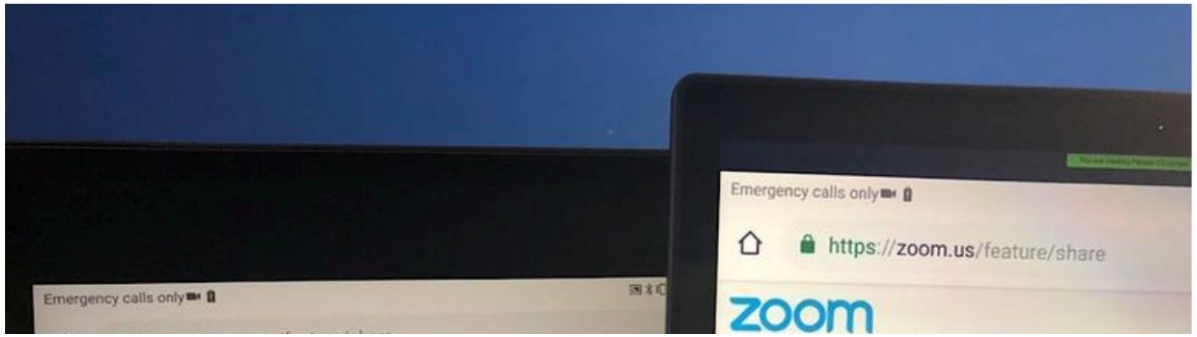
https://www.zdnet.com/article/zoom-defends-use-of-local-web-server-on-macs-after-security-report/

MUST READ: Google's new AI tool could help decode the mysterious algorithms that decide everything

Zoom reverses course to kill off Mac local web server

Less than a day after backing its approach to get around Safari restrictions on Mac, Zoom's local web server is no more.

By Chris Duckett | July 10, 2019 -- 00:50 GMT (17:50 PDT) | Topic: Security



https://www.zdnet.com/article/zoom-reverses-course-to-kill-off-mac-local-web-server/

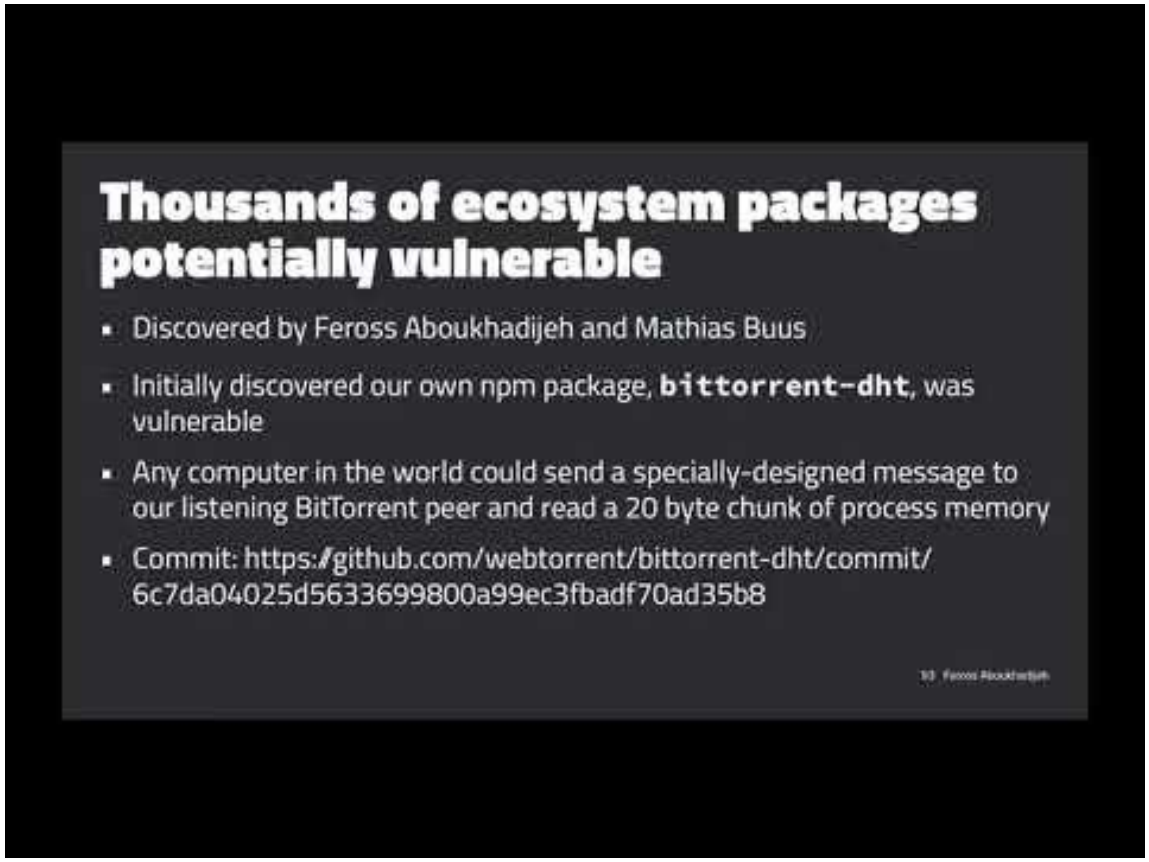
Cleaning up the mess

- Zoom issued an updated app which uninstalled the local HTTP server and added a new UI prompt to confirm that you want to join a meeting
- User who did not open the app for a while would be vulnerable until they installed the update
- Users who previously uninstalled Zoom would not get the update, so they'd be stuck with the vulnerable local server

Remote Code Execution (RCE)

- Around one week after the local server issue came to light, another research team discovered a RCE vulnerability
- The complete exploit allowed a zero-interaction RCE just by visiting a malicious site – yikes!
  - https://blog.assetnote.io/bug-bounty/2019/07/17/rce-on-zoom/

Zoom RCE



http://assetnotehackszoom.com no longer live; nor is it archived
http://web.archive.org/web/*/assetnotehackszoom.com

APPLE | TECH | CYBERSECURITY

Apple is silently removing Zoom's web server software from Macs

48

For users who haven't seen all the drama

By Dieter Bohn | @backlon | Jul 10, 2019, 7:12pm EDT

SHARE

https://www.theverge.com/2019/7/10/20689644/apple-zoom-web-server-automatic-removal-silent-update-webcam-vulnerability

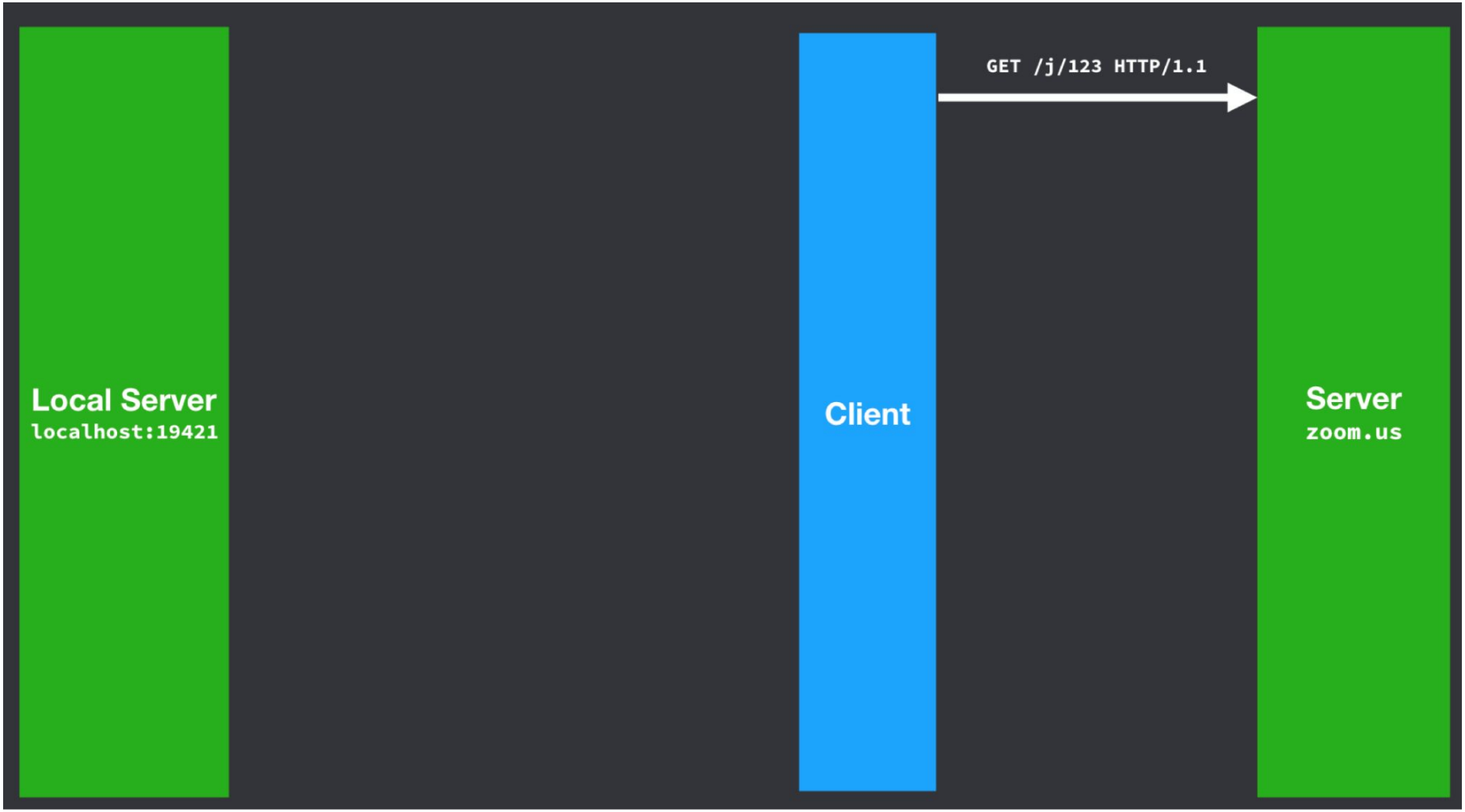
Apple takes steps

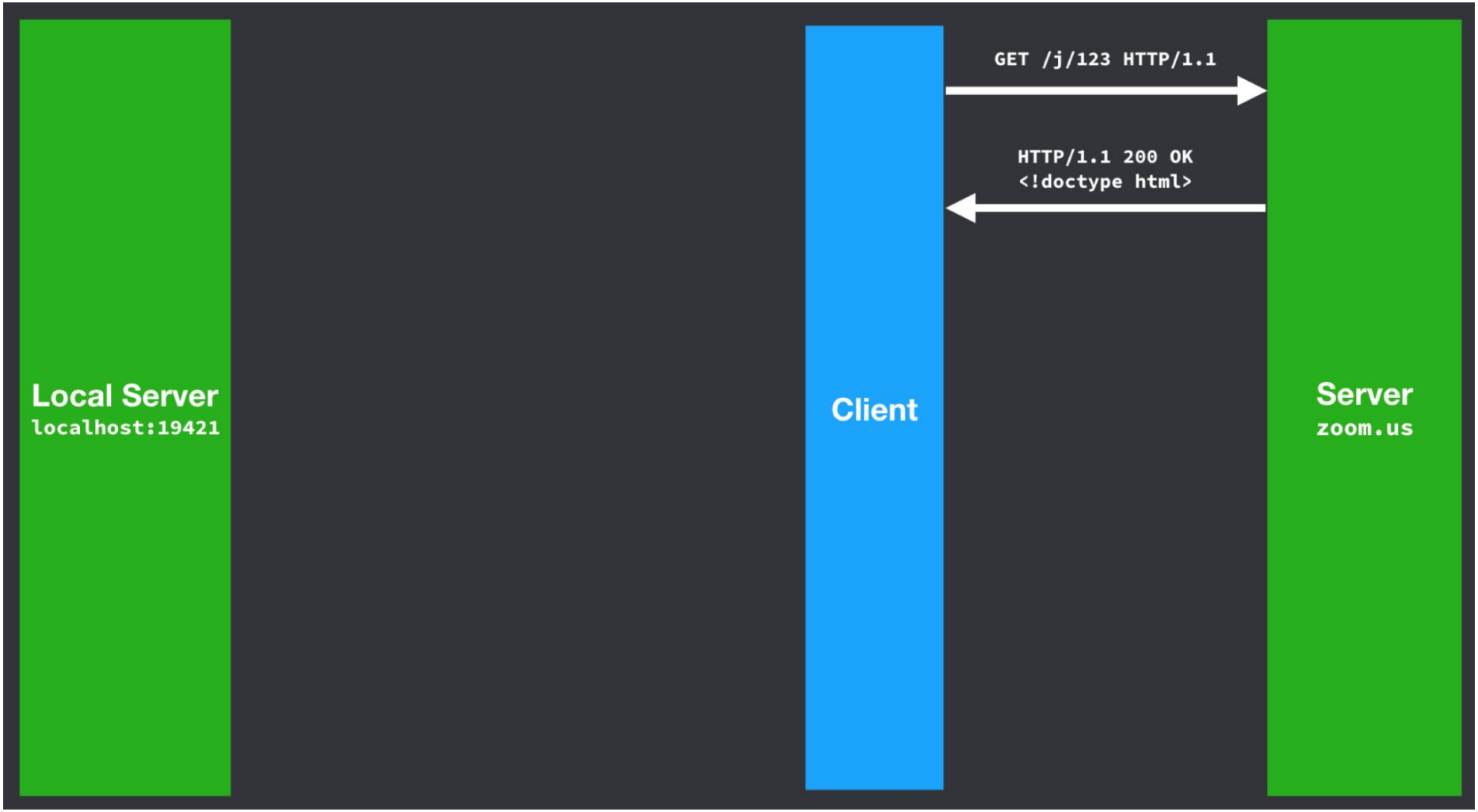
- macOS has a silent update mechanism for disabling specific executable files (Malware Removal Tool)
- No OS update required – checks for new banned executables in the background, regularly
- Useful for disabling fast-spreading malware or vulnerable software affecting lots of users

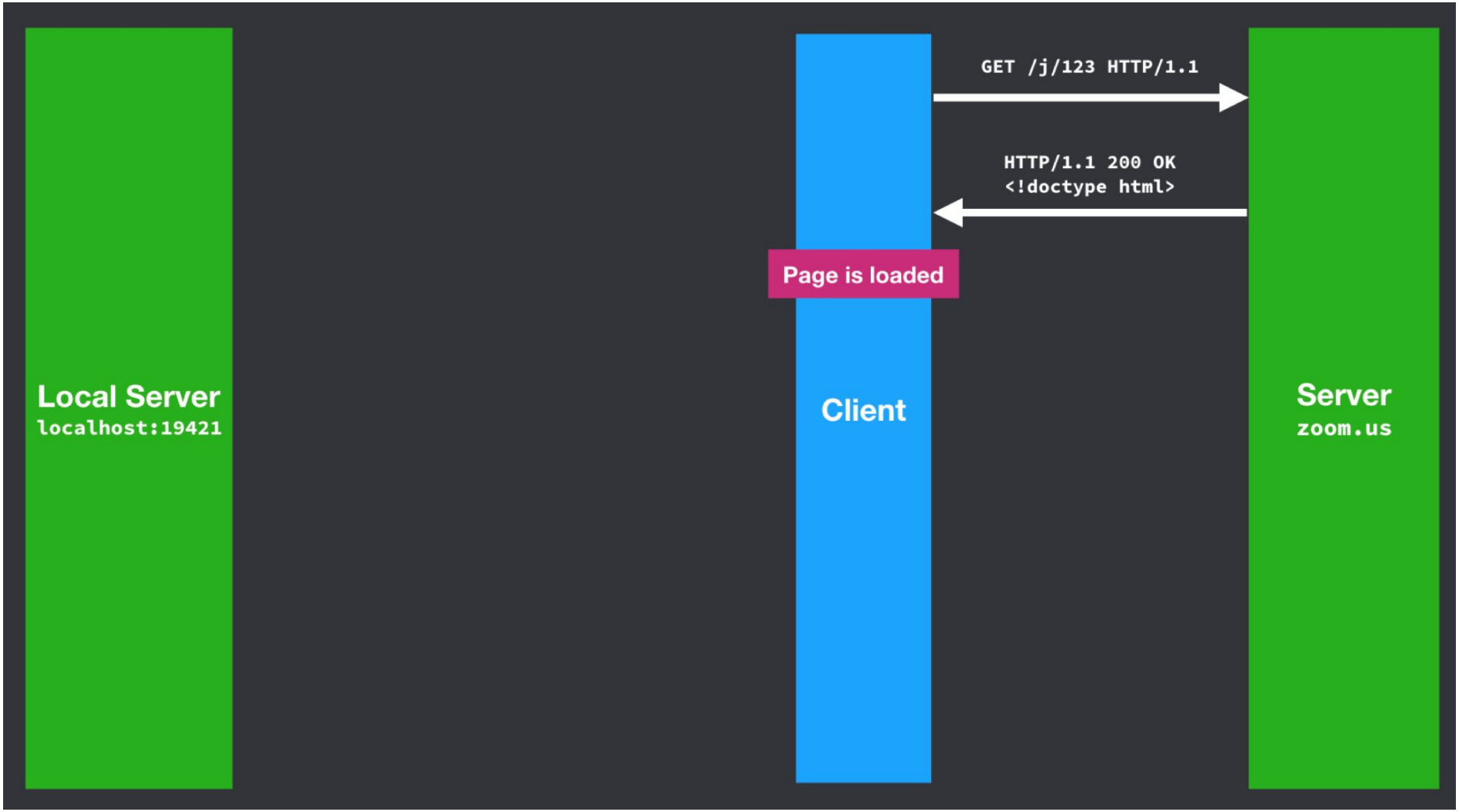
Zoom didn’t understand CORS

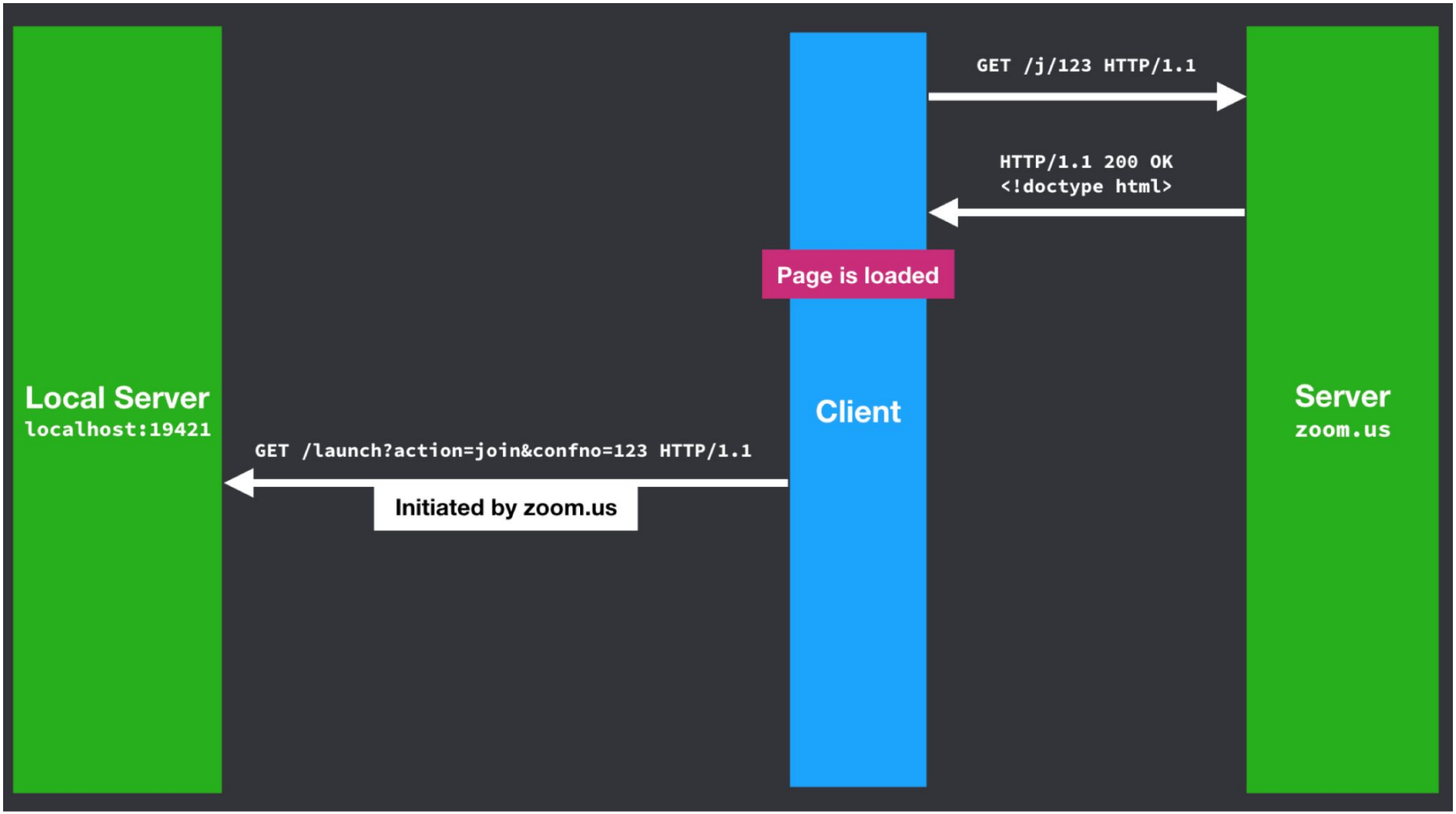
User joins a zoom call (vulnerable)

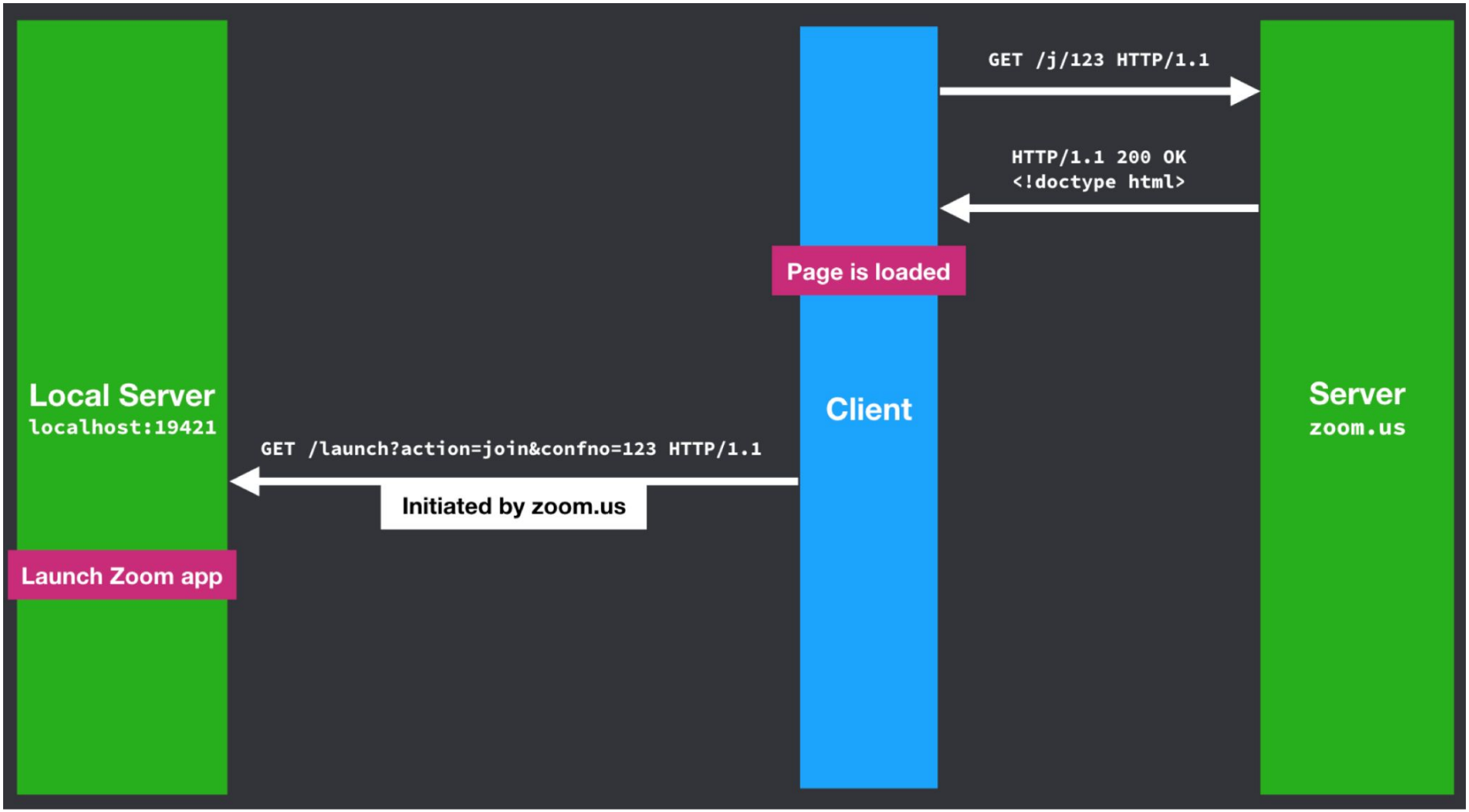

Local Server
localhost:19421
Client
Server
zoom.us

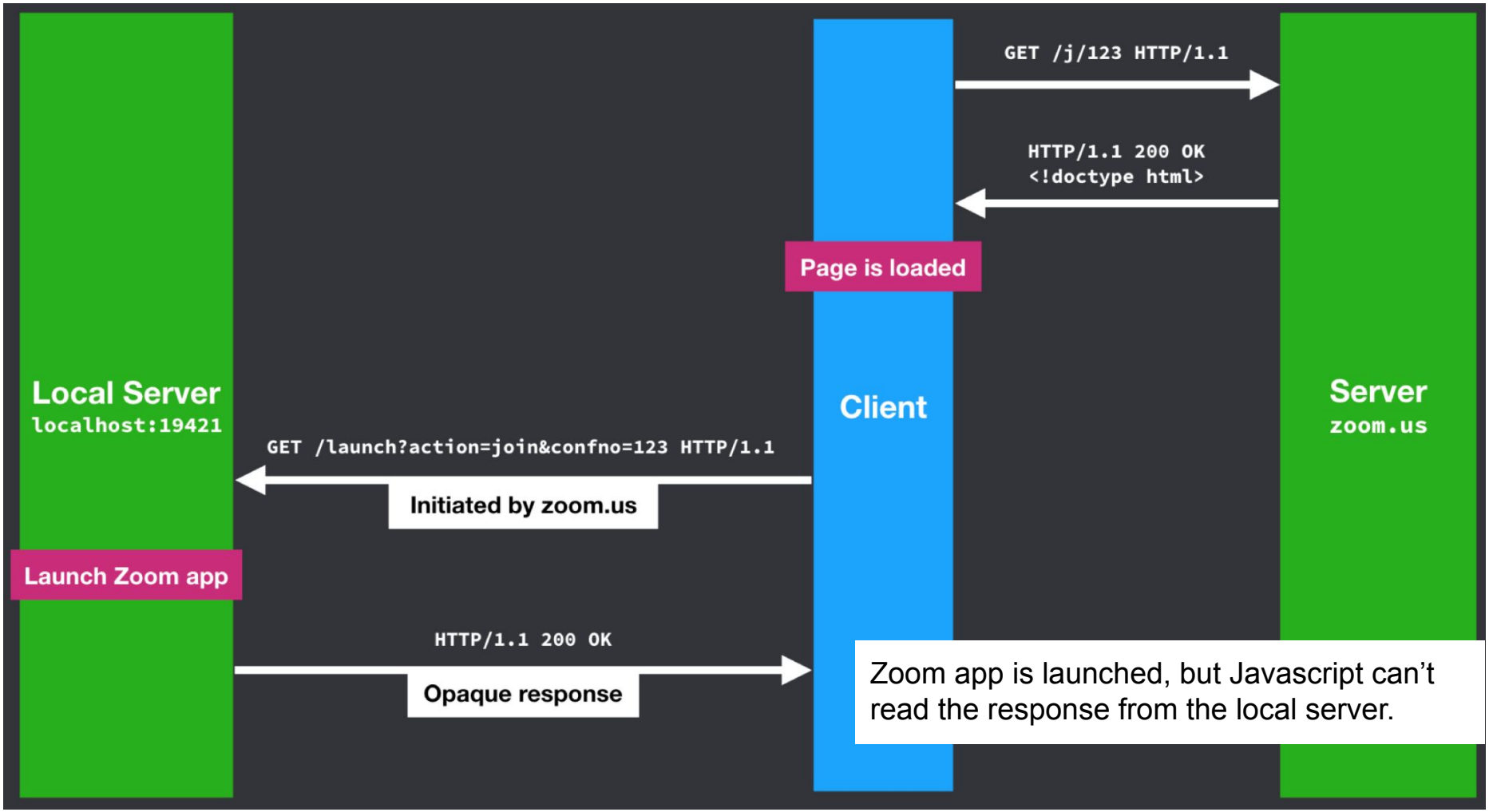

Local Server
localhost:19421
Client
GET /j/123 HTTP/1.1
Server
zoom.us

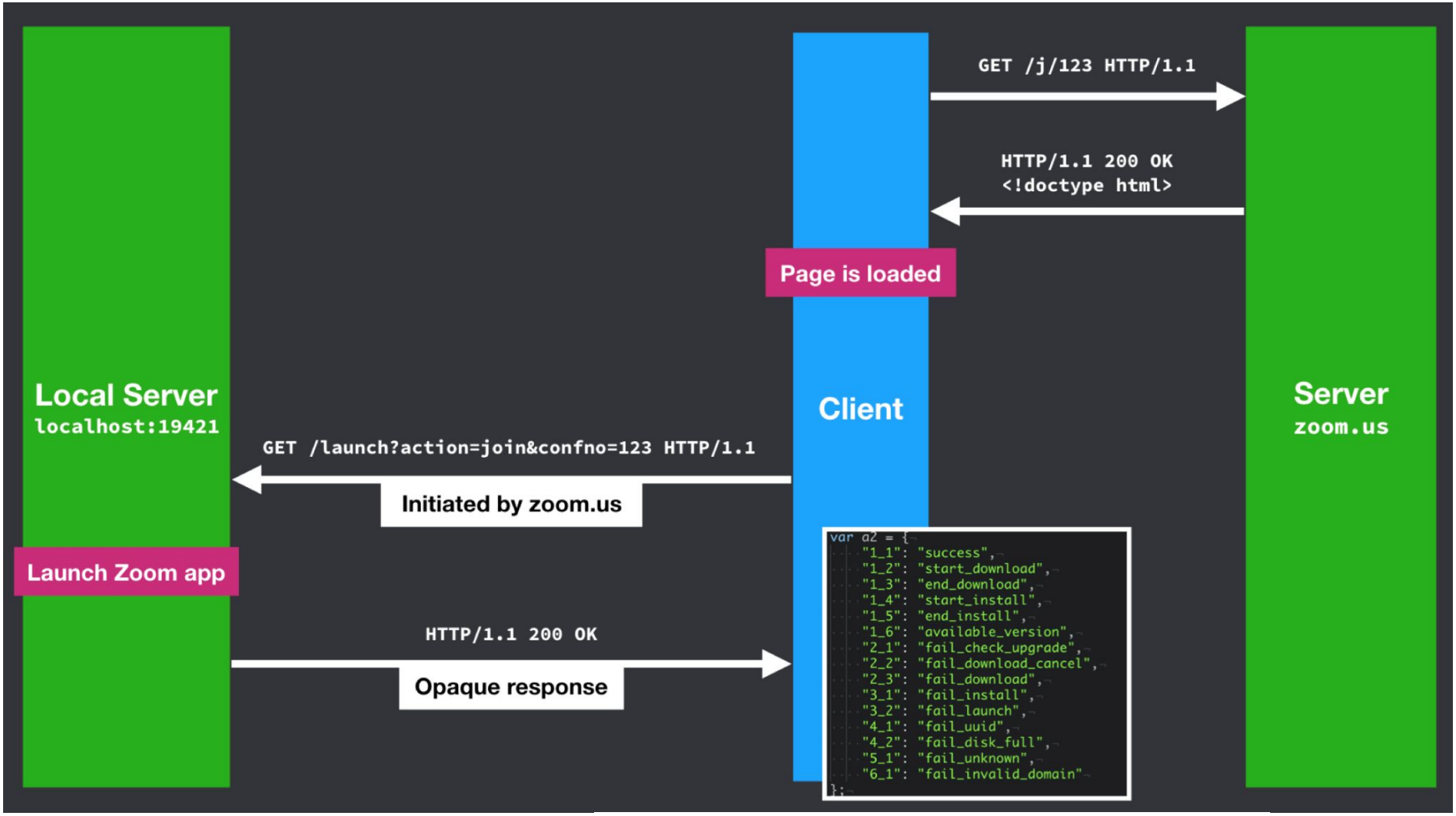

Local Server
localhost:19421
Client
GET /j/123 HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Server
zoom.us


Local Server
localhost:19421
Client
GET /j/123 HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Server
zoom.us


GET /j/123 HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
Client
Server
zoom.us
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by zoom.us


GET /j/123 HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
Client
Server
zoom.us
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by zoom.us
Launch Zoom app


GET /j/123 HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
Client
Server
zoom.us
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by zoom.us
Launch Zoom app
HTTP/1.1 200 OK
Opaque response
Zoom app is launched, but Javascript can't read the response from the local server.

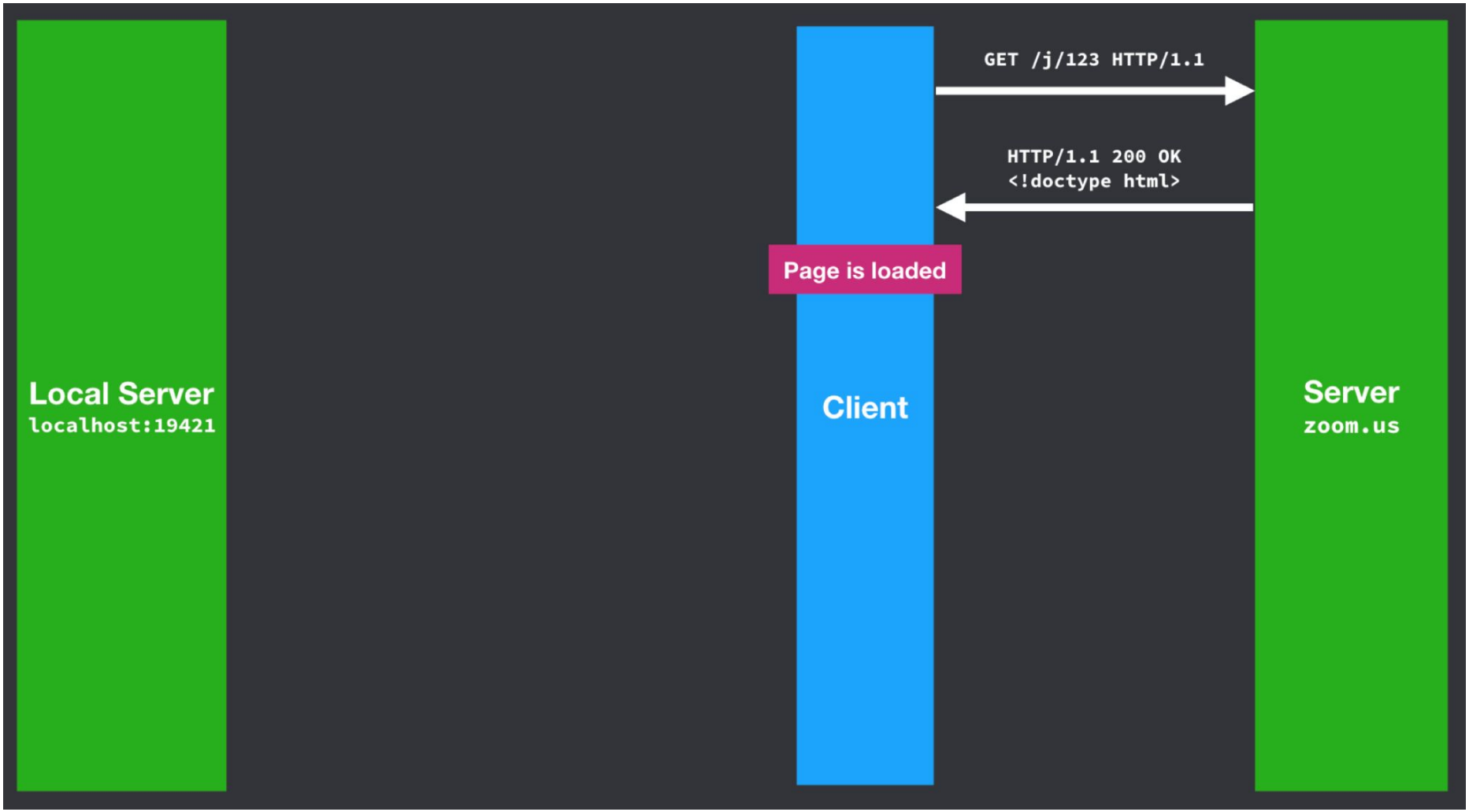


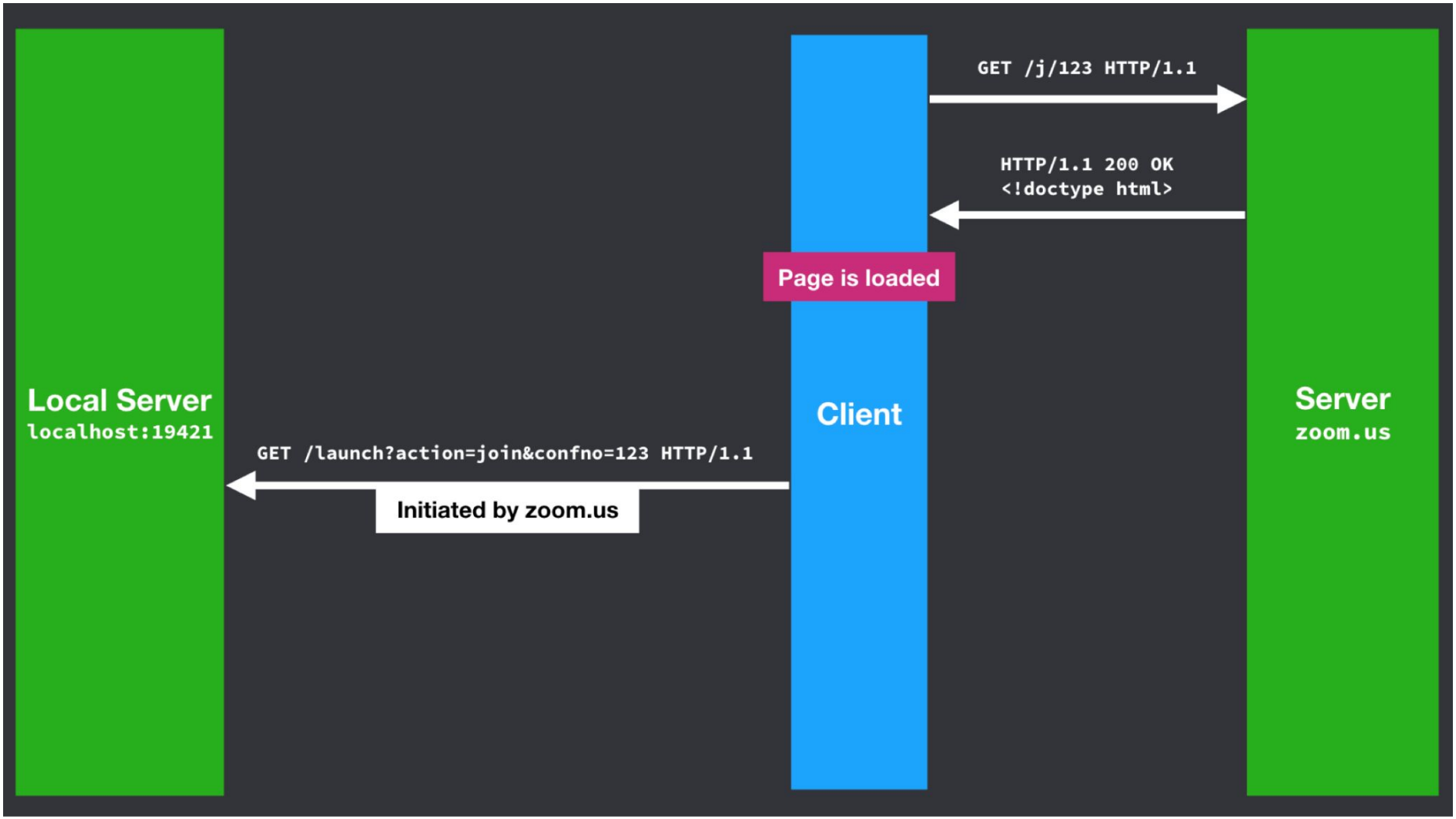
Depends on pixel width instead to convey status.

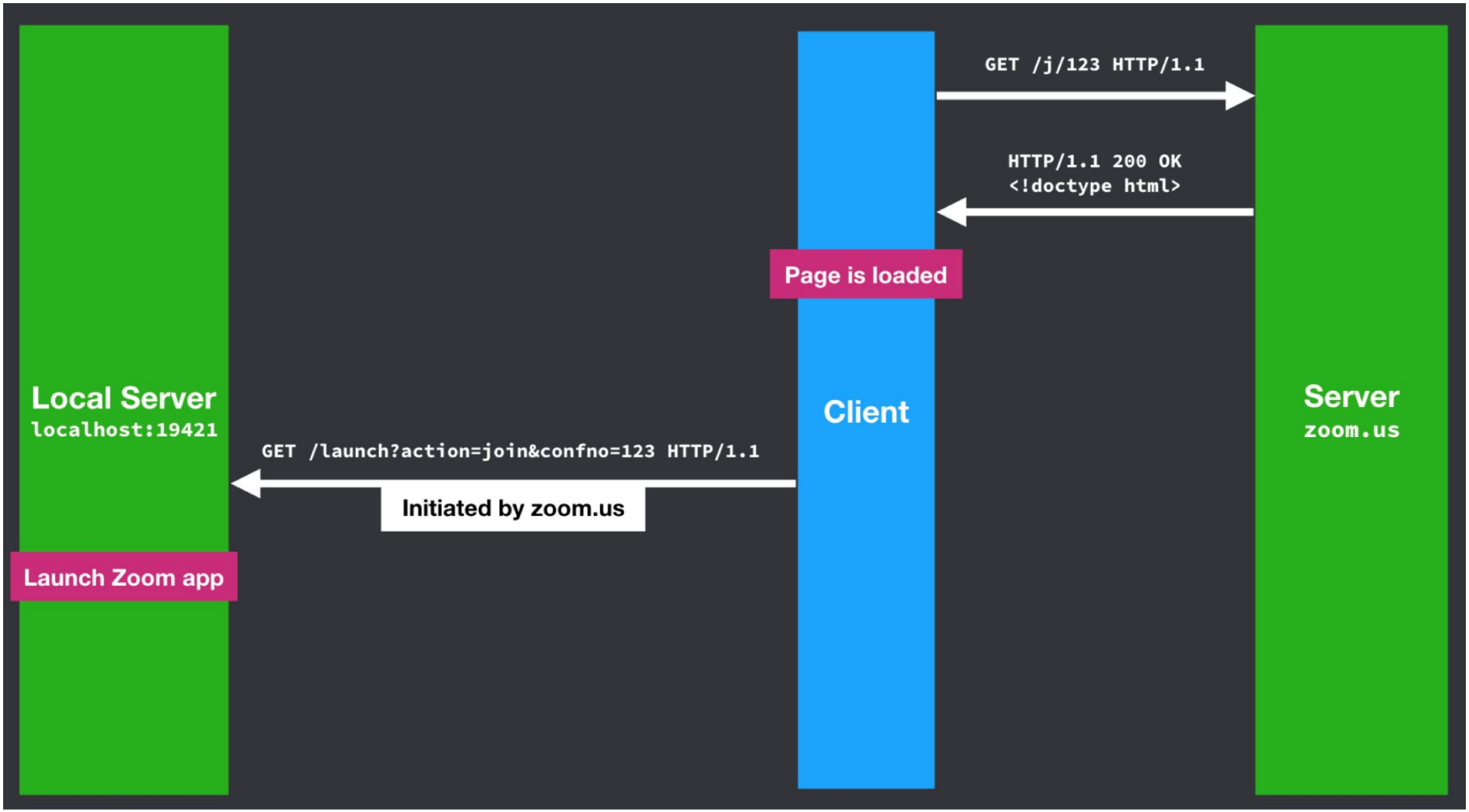
Zoom doesn't understand how CORS works?

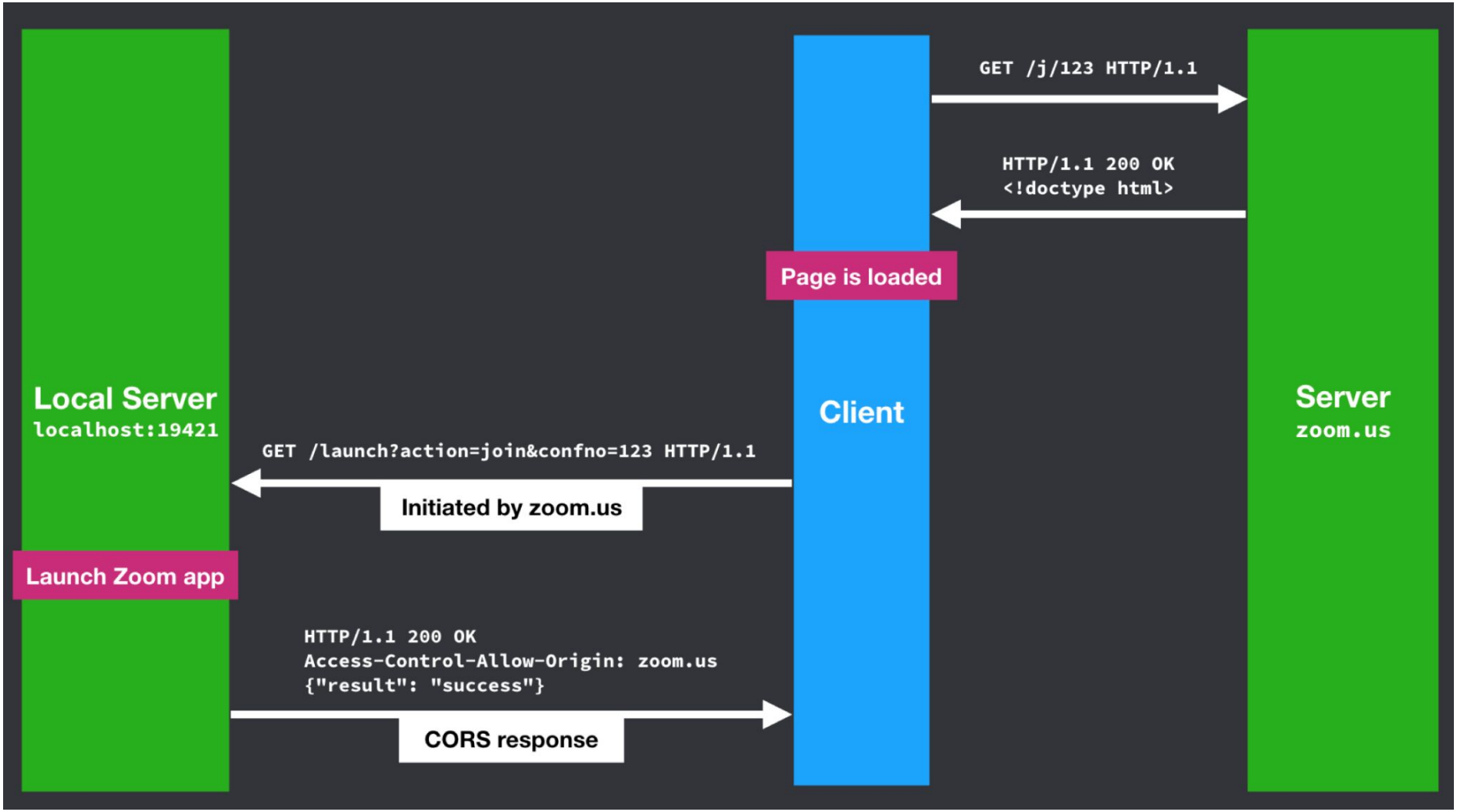
- The `http://localhost:19421/launch?action=join&confno=###` endpoint returns information about whether the request succeeded, but since it's triggered from `https://zoom.us` the same origin policy doesn't allow reading the response
- So, they returned an image with different widths/heights to "leak" information to the site that triggered the request
- They could have just used `Access-Control-Allow-Origin` to specify particular sites which would be allowed to read the response

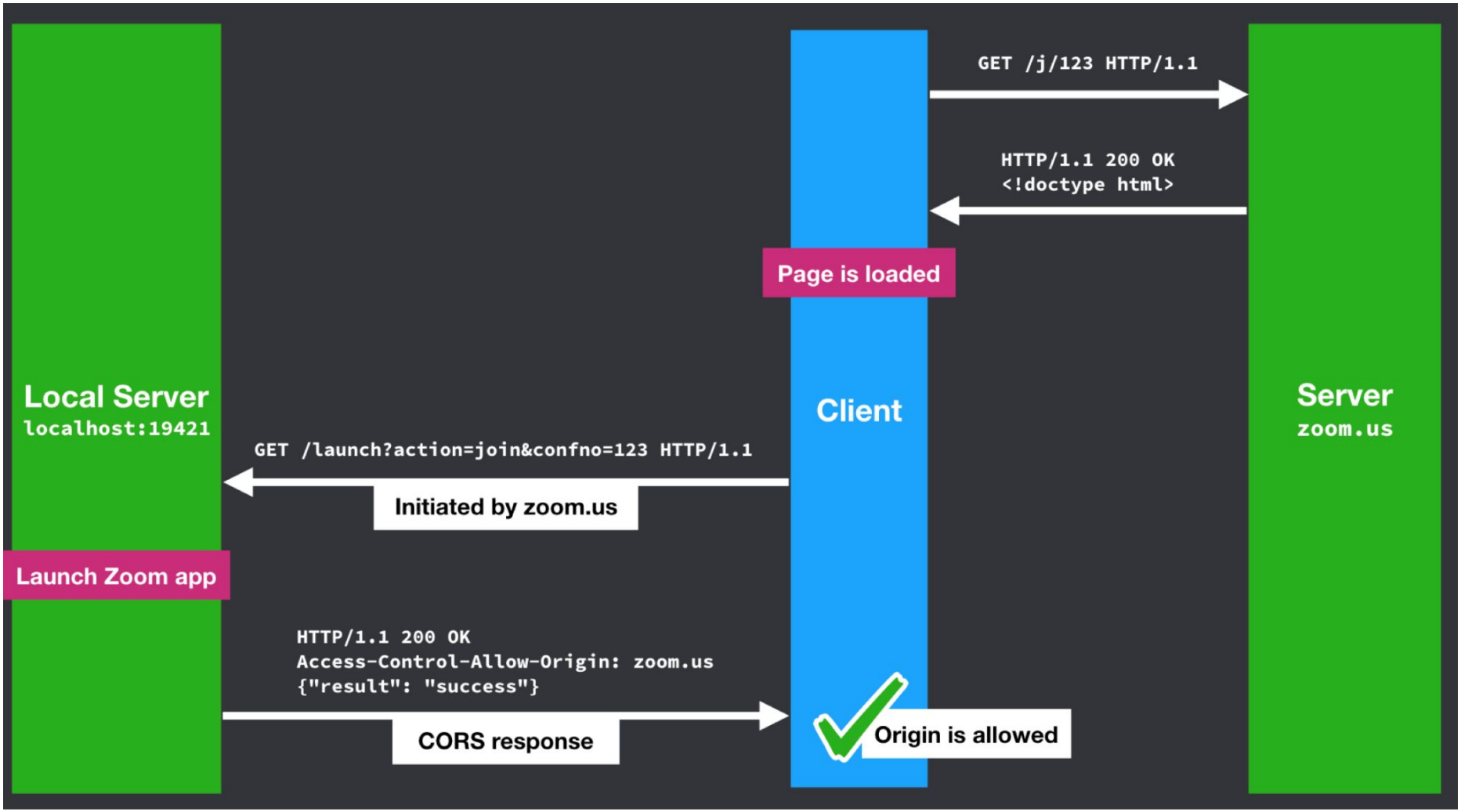
User joins a zoom call (with CORS endpoint) (vulnerable)


Local Server
localhost:19421
Client
Server
zoom.us
GET /j/123 HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded

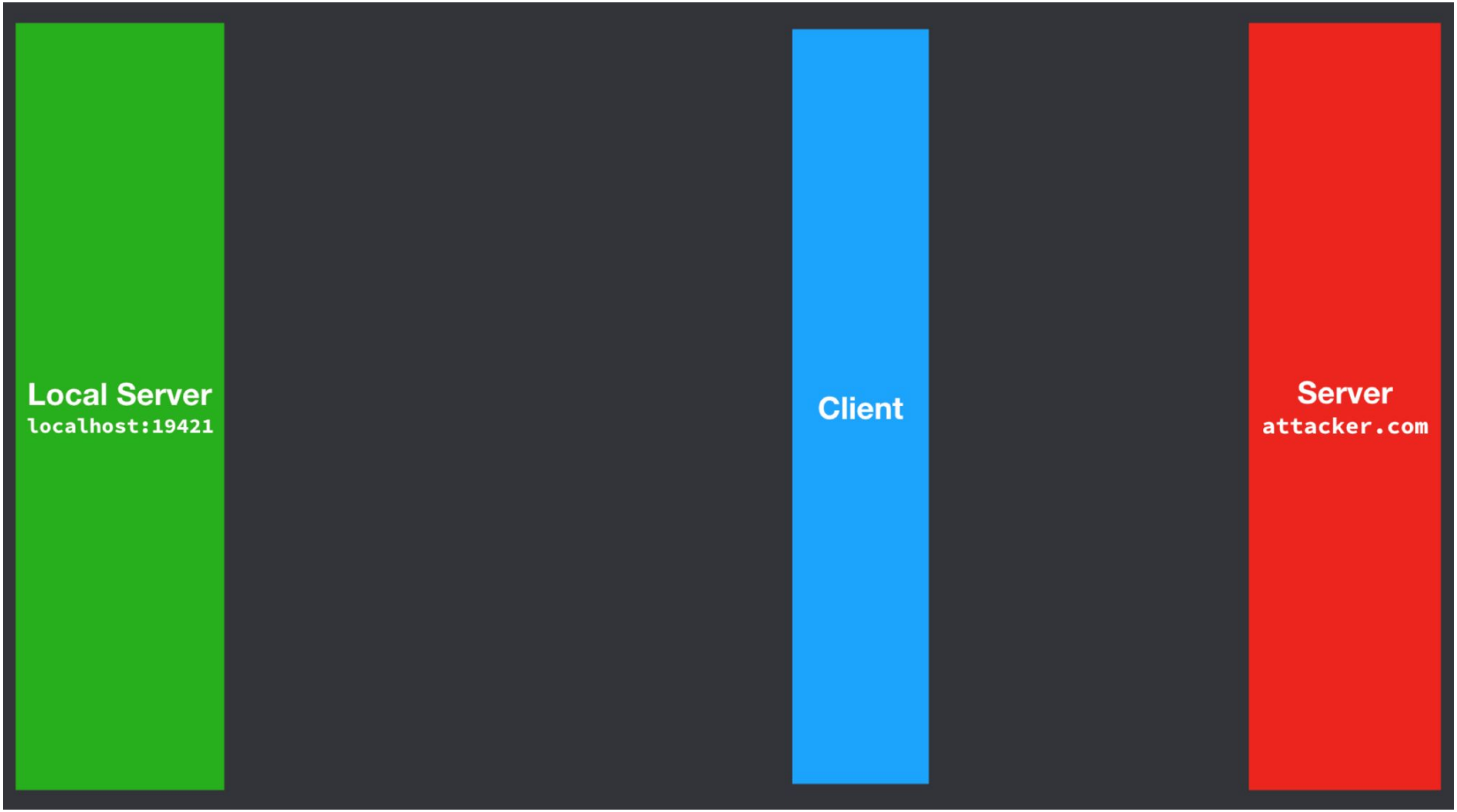

Local Server
localhost:19421
Client
Server
zoom.us
GET /j/123 HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by zoom.us


GET /j/123 HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
Client
Server
zoom.us
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by zoom.us
Launch Zoom app

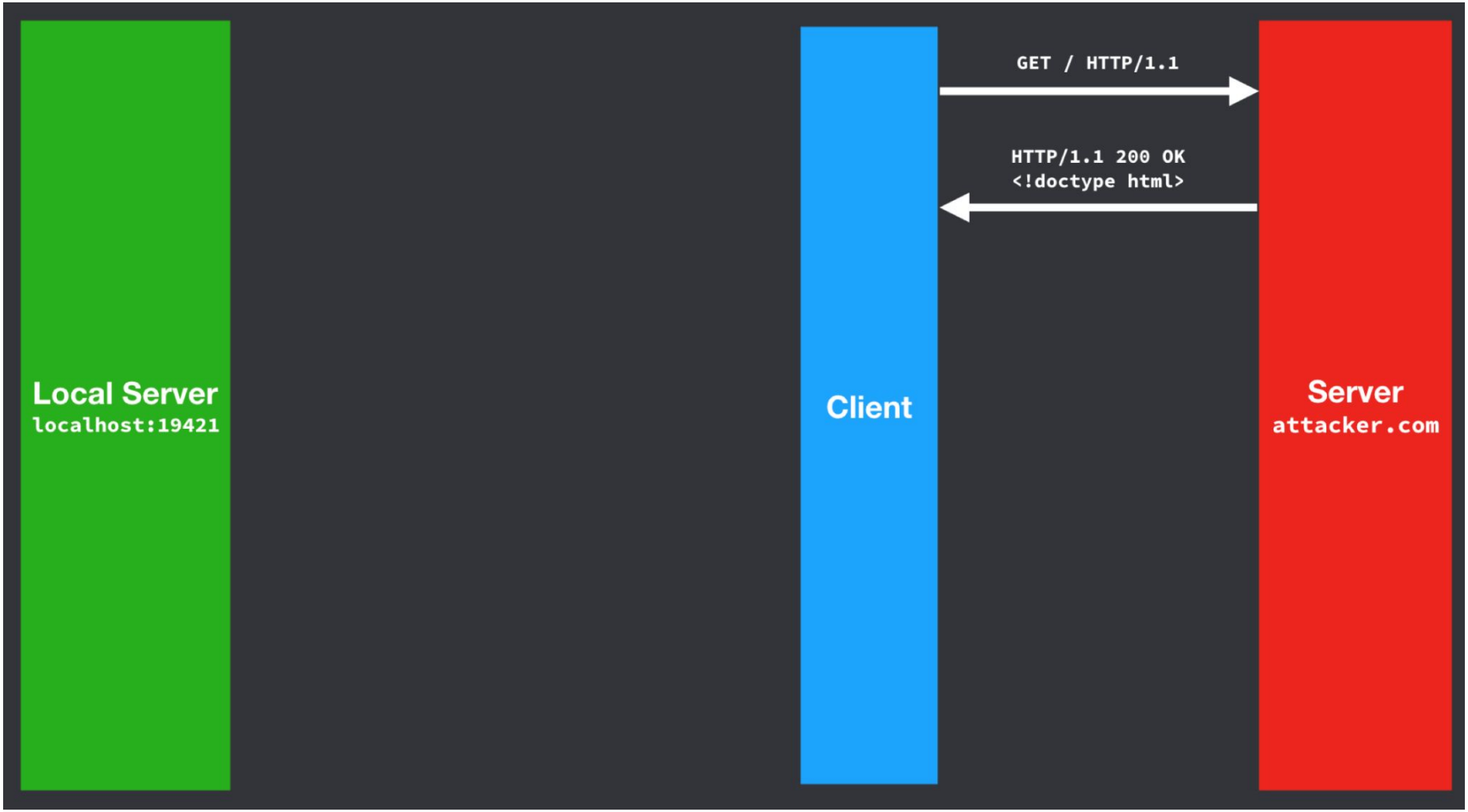

Local Server
localhost:19421
Client
Server
zoom.us
GET /j/123 HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by zoom.us
Launch Zoom app
HTTP/1.1 200 OK
Access-Control-Allow-Origin: zoom.us
{"result": "success"}
CORS response

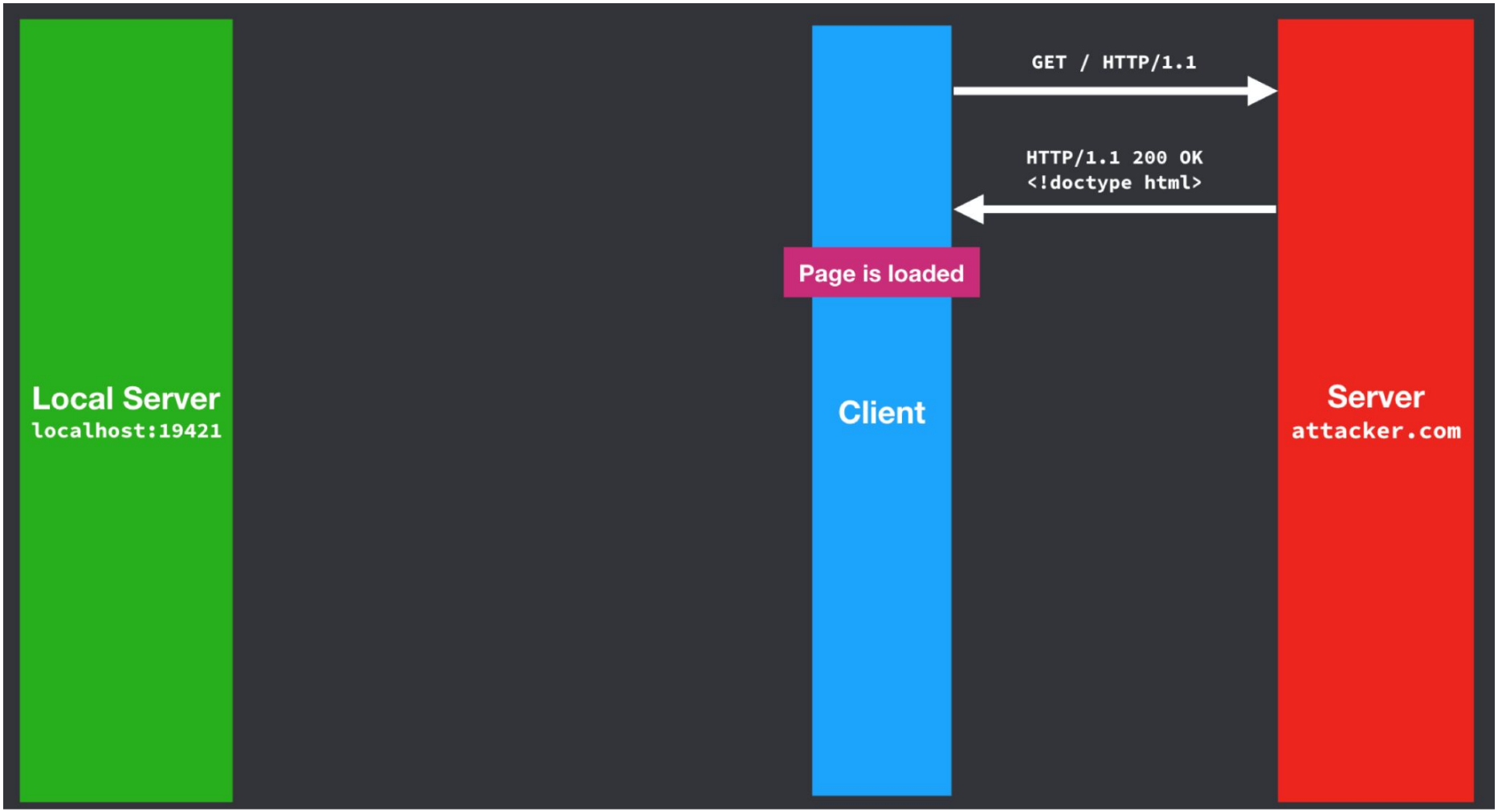

Local Server
localhost:19421
Client
Server
zoom.us
GET /j/123 HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by zoom.us
Launch Zoom app
HTTP/1.1 200 OK
Access-Control-Allow-Origin: zoom.us
{"result": "success"}
CORS response
Origin is allowed

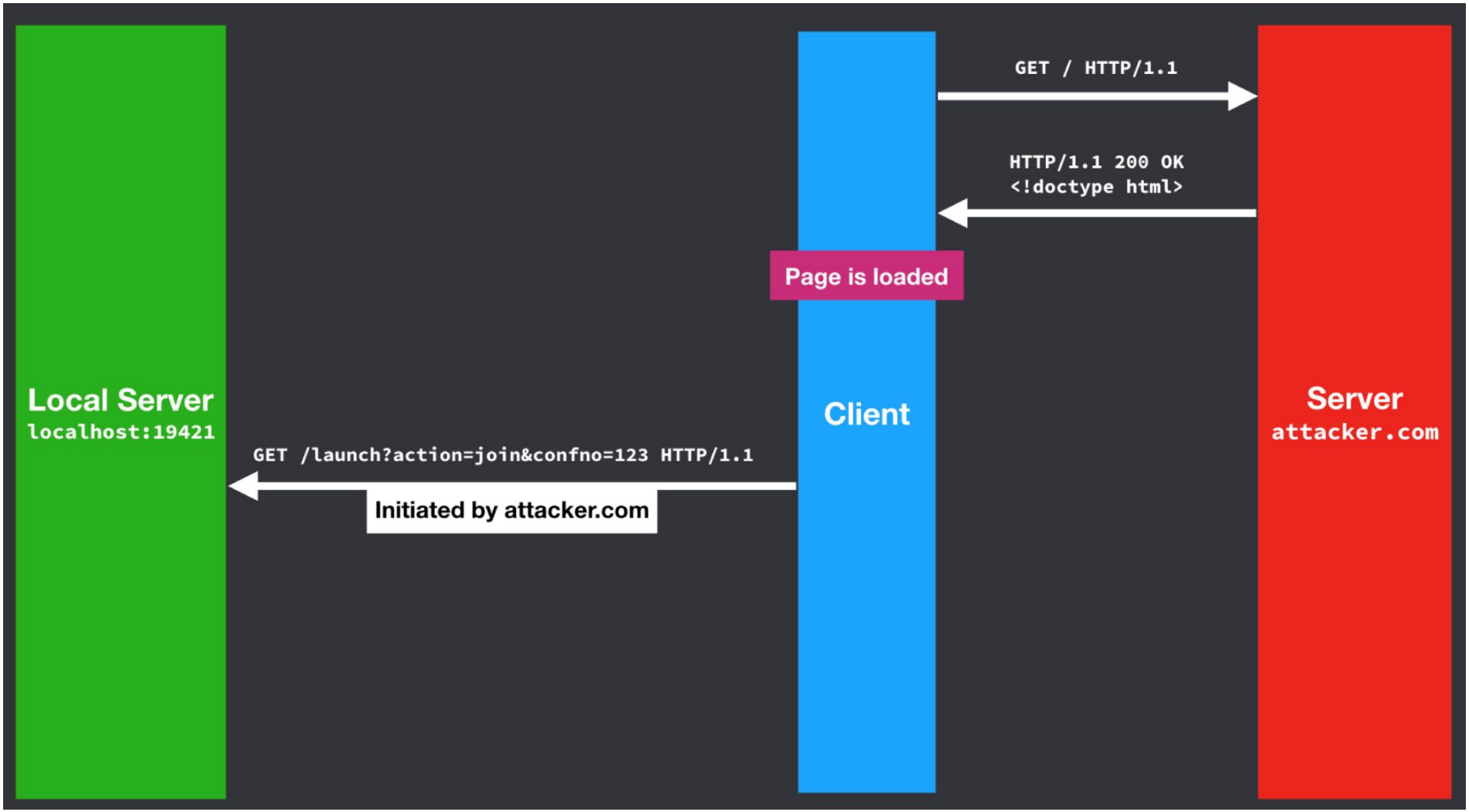
Attacker joins user into a zoom call

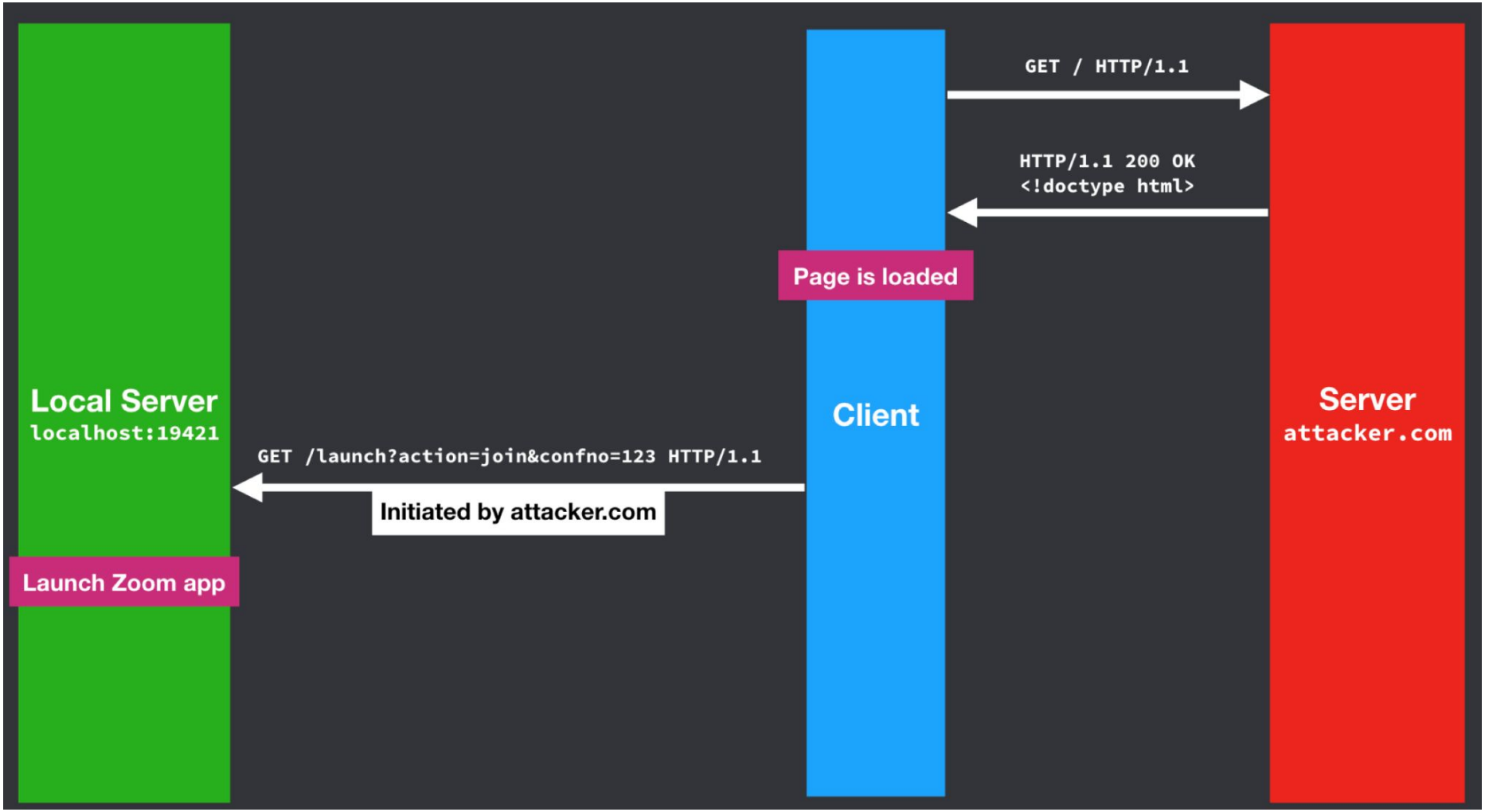

Local Server
localhost:19421
Client
Server
attacker.com

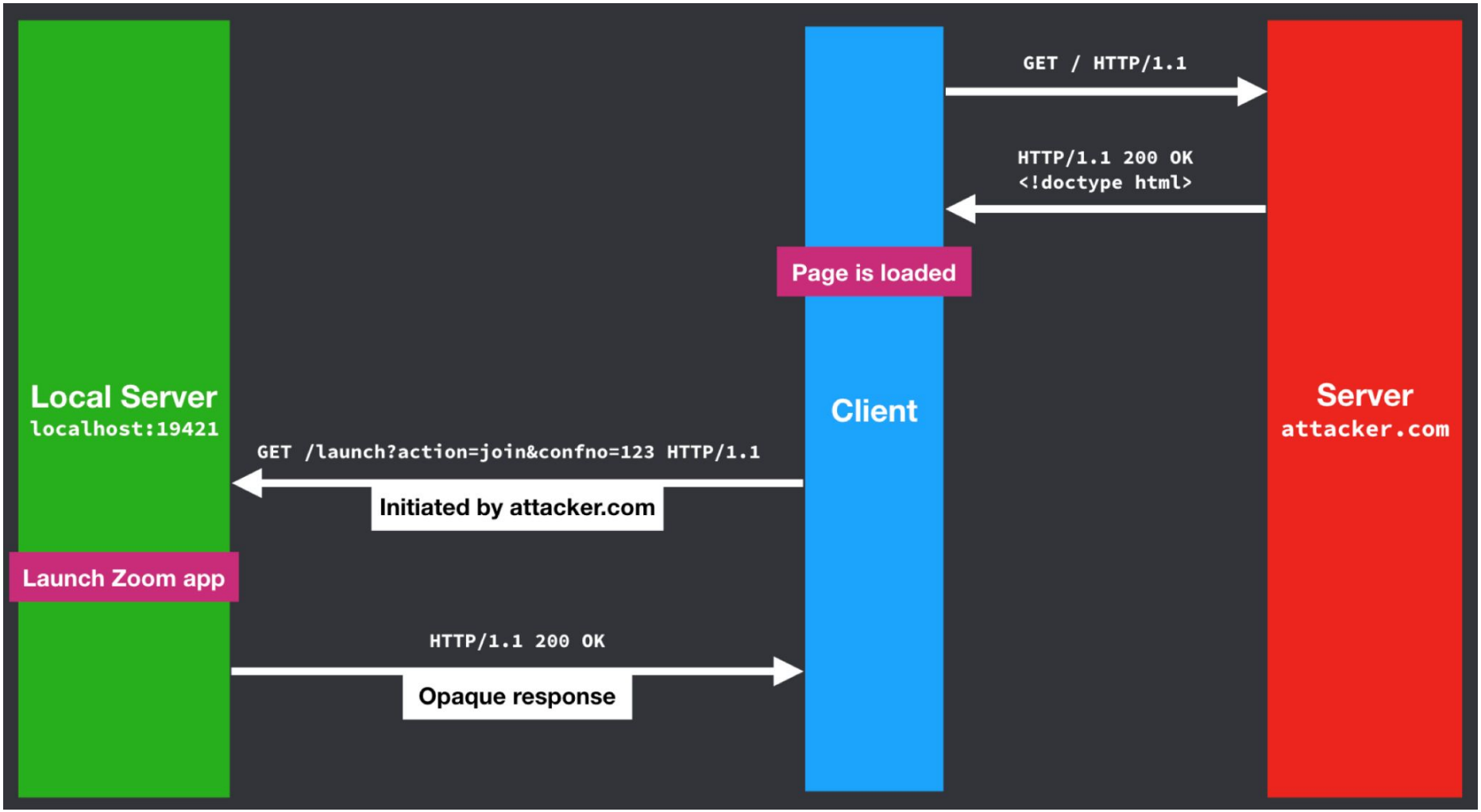

Local Server
localhost:19421
Client
GET / HTTP/1.1
Server
attacker.com


Local Server
localhost:19421
Client
GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Server
attacker.com

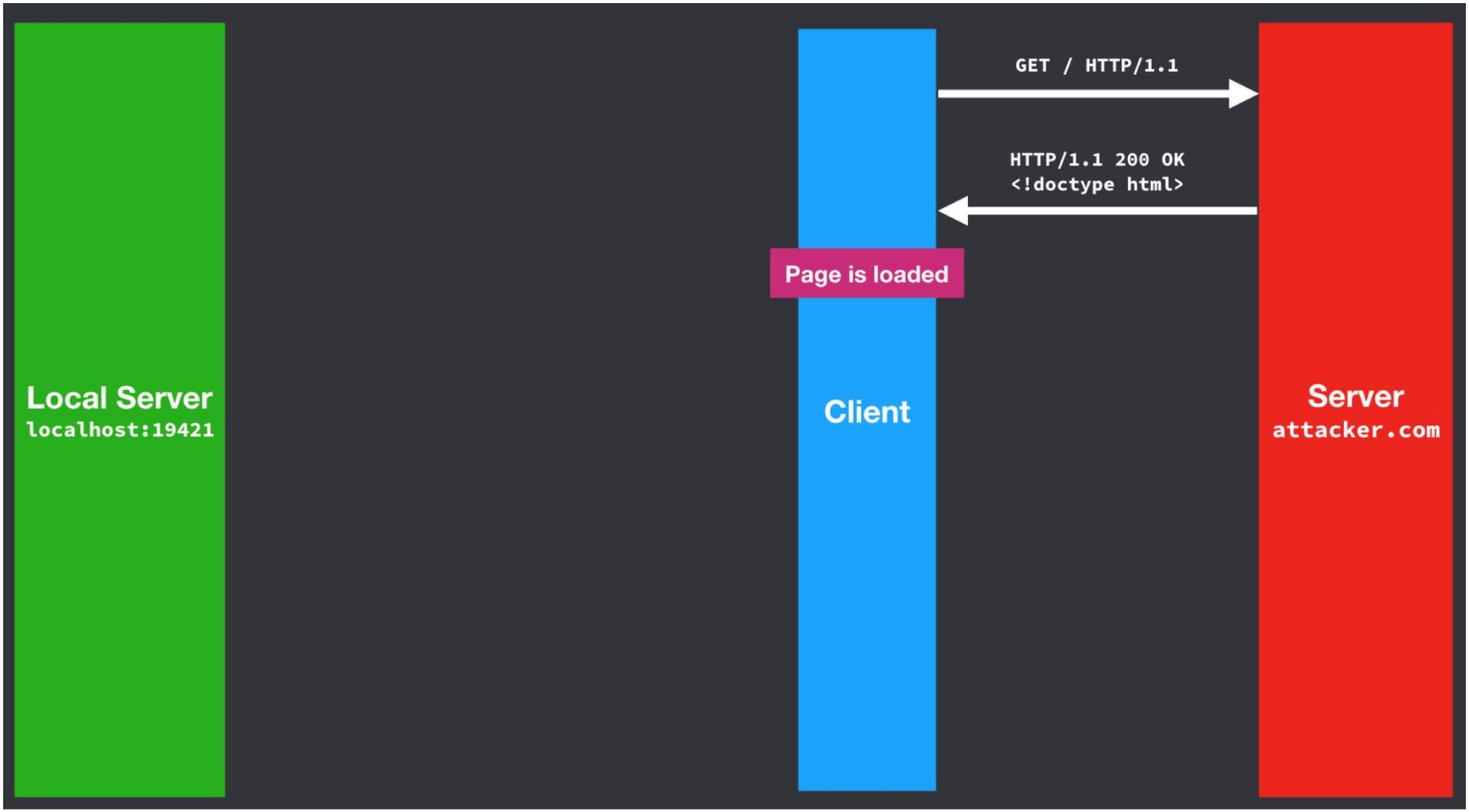

Local Server
localhost:19421
Client
Server
attacker.com
GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded

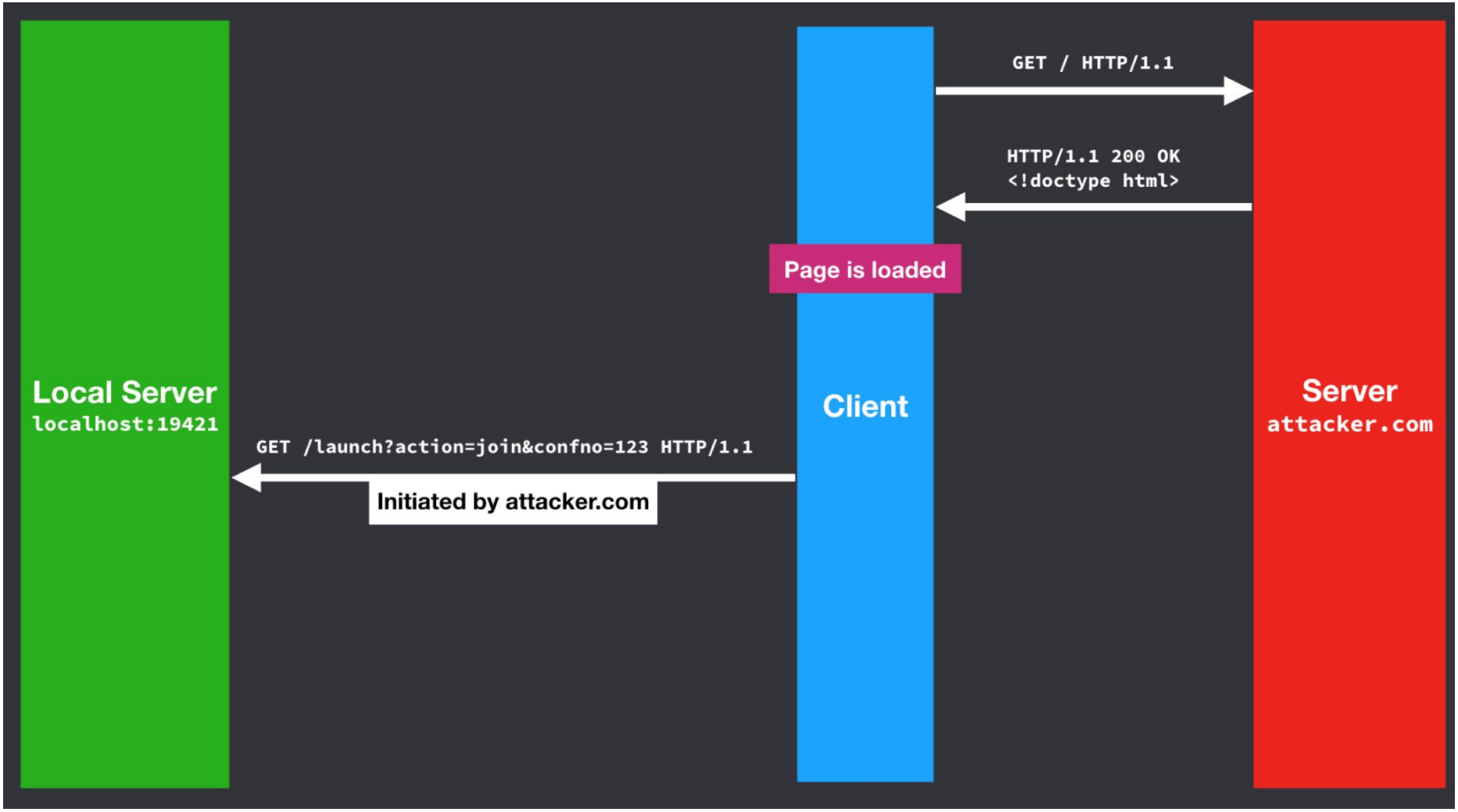

Local Server
localhost:19421
Client
Server
attacker.com
GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by attacker.com

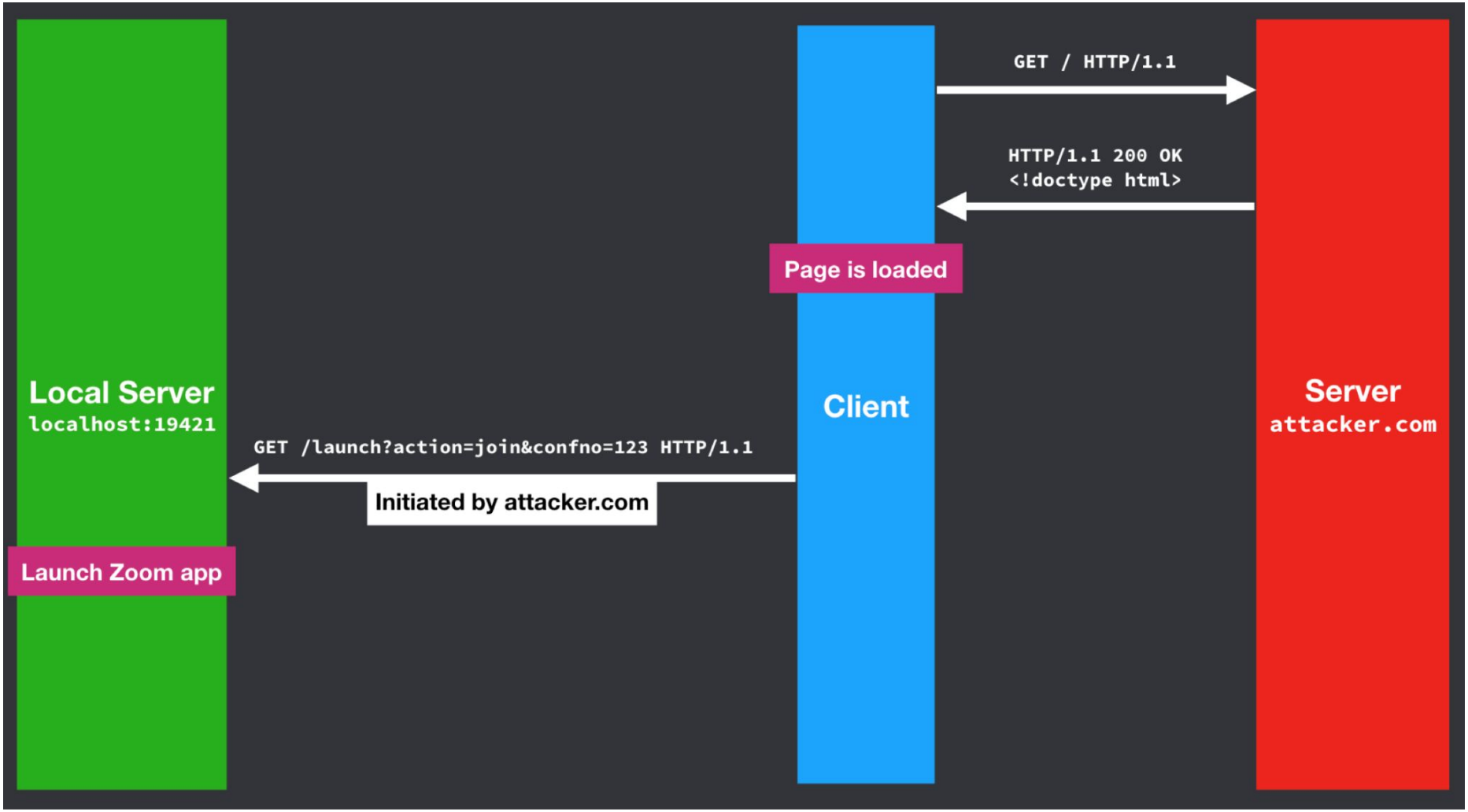

GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
Client
Server
attacker.com
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by attacker.com
Launch Zoom app

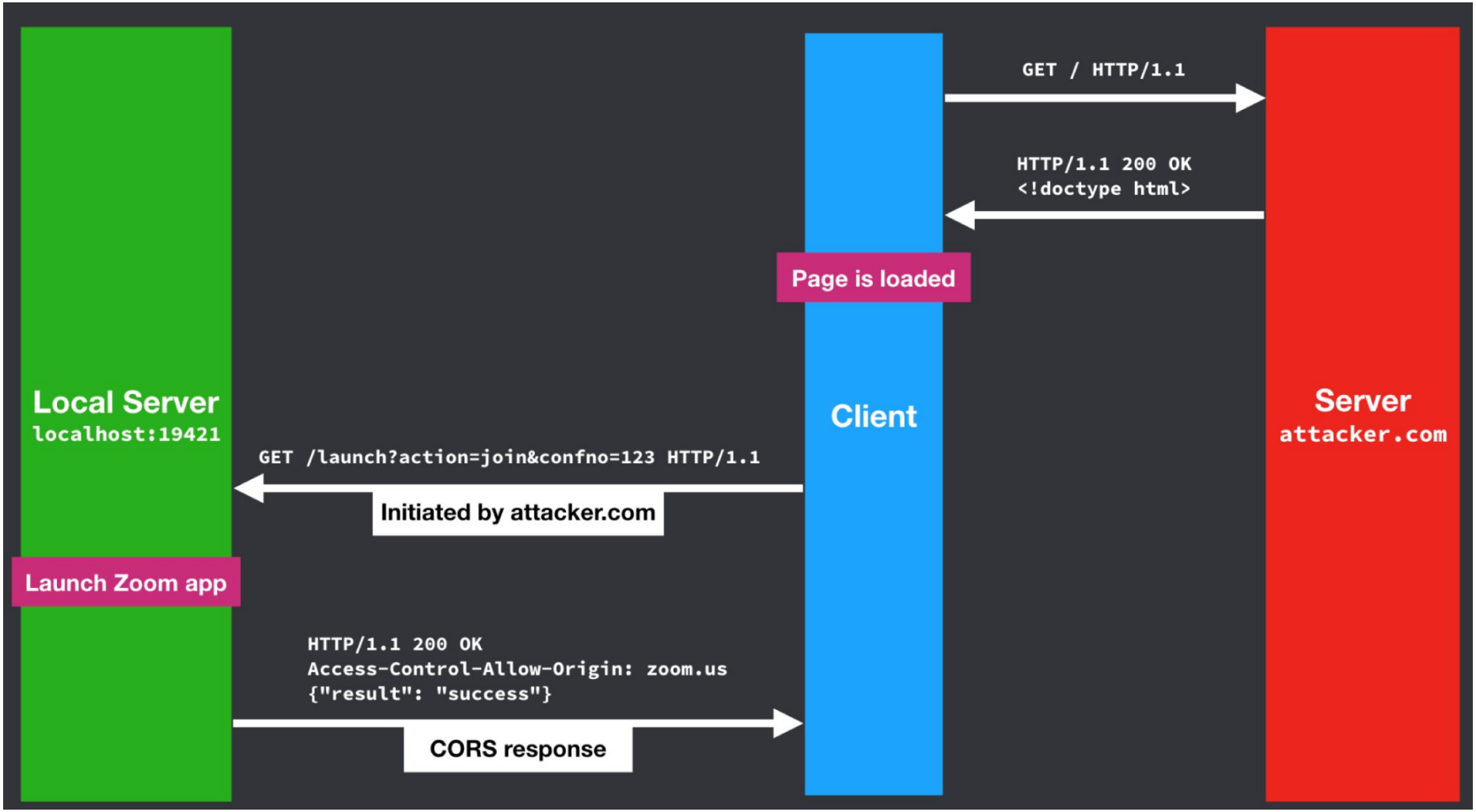

GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
Client
Server
attacker.com
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by attacker.com
Launch Zoom app
HTTP/1.1 200 OK
Opaque response

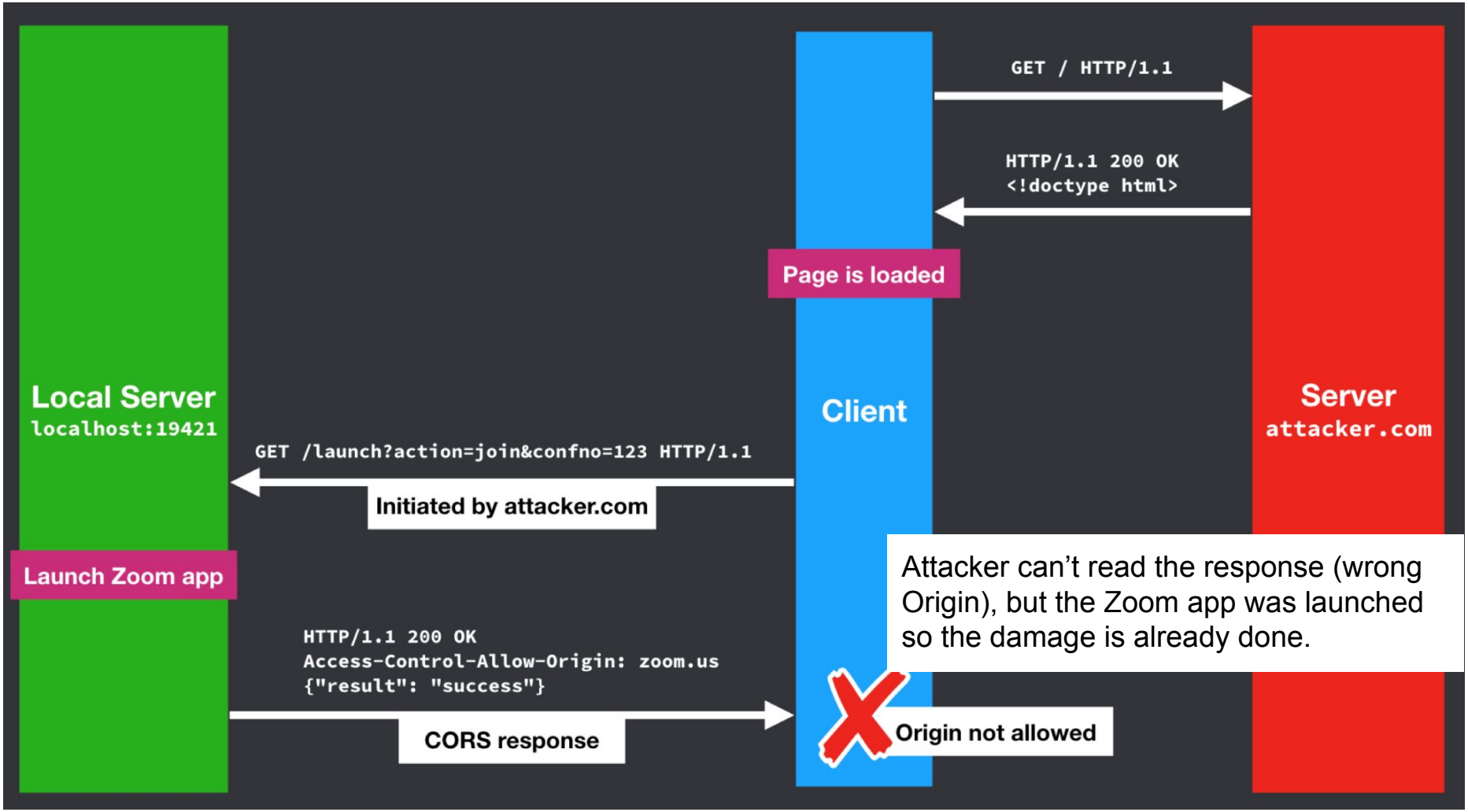
Attacker joins user into a zoom call (with CORS endpoint)


Local Server
localhost:19421
Client
Server
attacker.com
GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded


Local Server
localhost:19421
Client
Server
attacker.com
GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by attacker.com


Local Server
localhost:19421
Launch Zoom app
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by attacker.com
Client
Page is loaded
GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Server
attacker.com

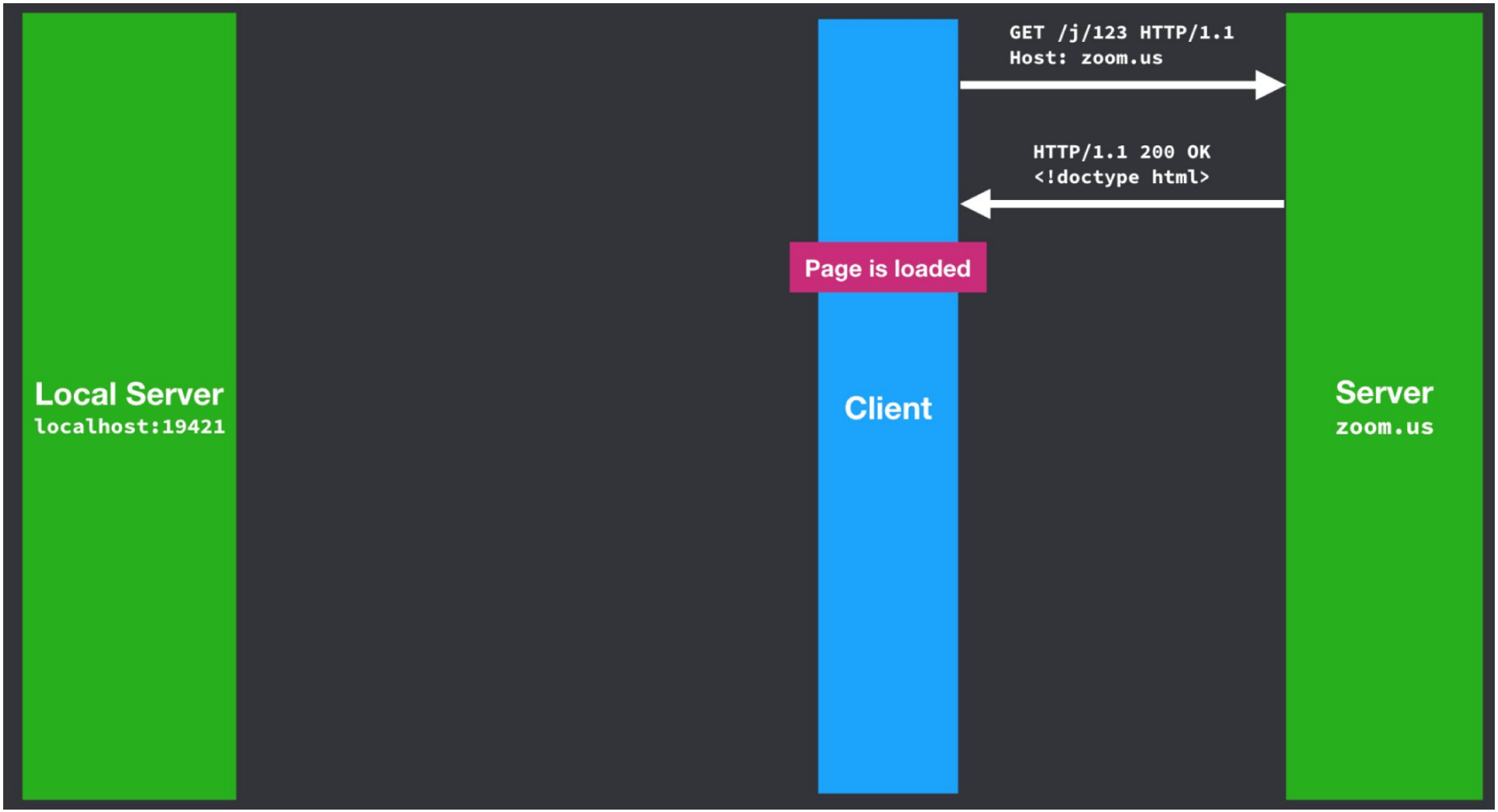

GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
Client
Server
attacker.com
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by attacker.com
Launch Zoom app
HTTP/1.1 200 OK
Access-Control-Allow-Origin: zoom.us
{"result": "success"}
CORS response

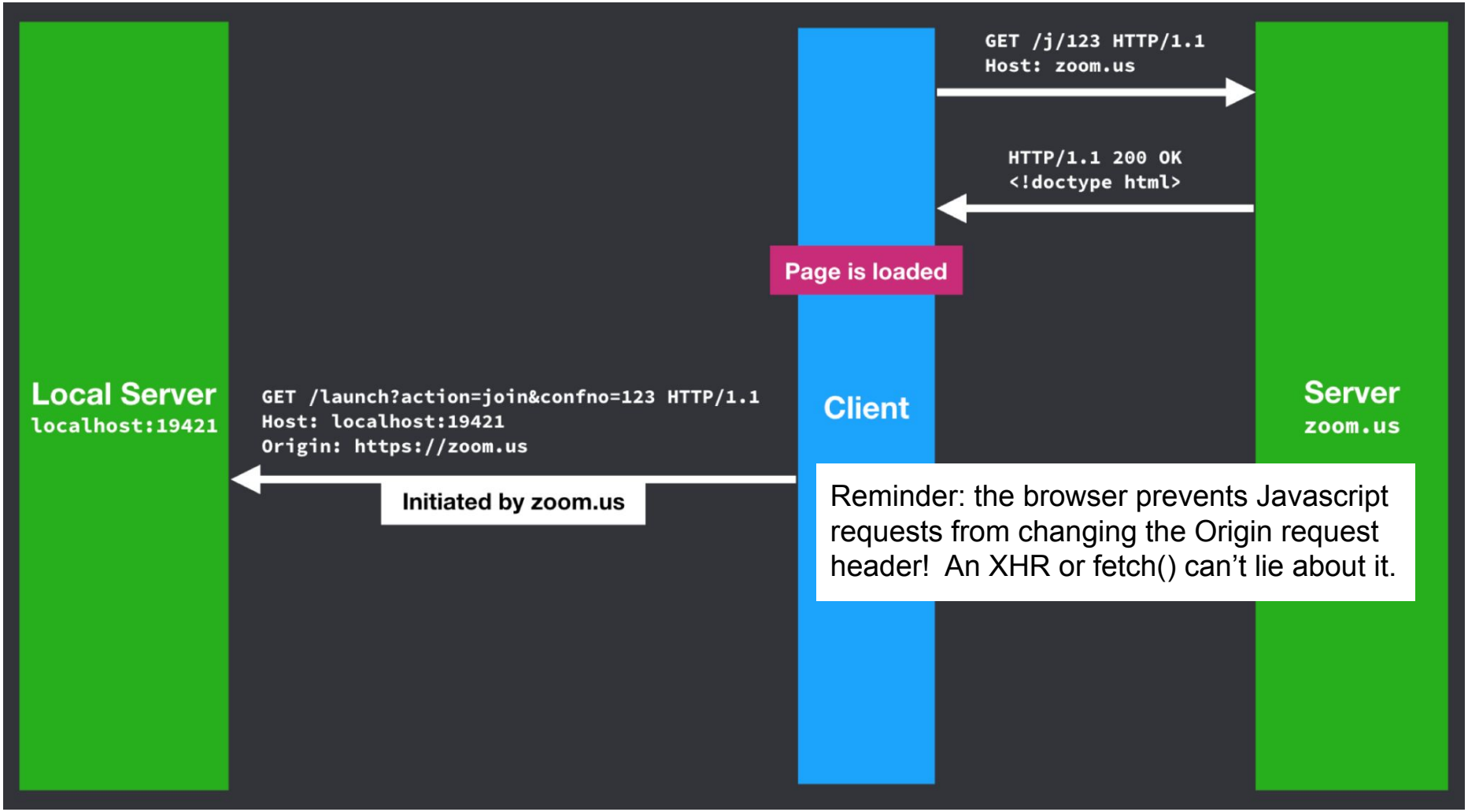

Local Server
localhost:19421
Launch Zoom app
Client
Server
attacker.com
GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
GET /launch?action=join&confno=123 HTTP/1.1
Initiated by attacker.com
HTTP/1.1 200 OK
Access-Control-Allow-Origin: zoom.us
{"result": "success"}
CORS response
Origin not allowed
Attacker can't read the response (wrong Origin), but the Zoom app was launched so the damage is already done.

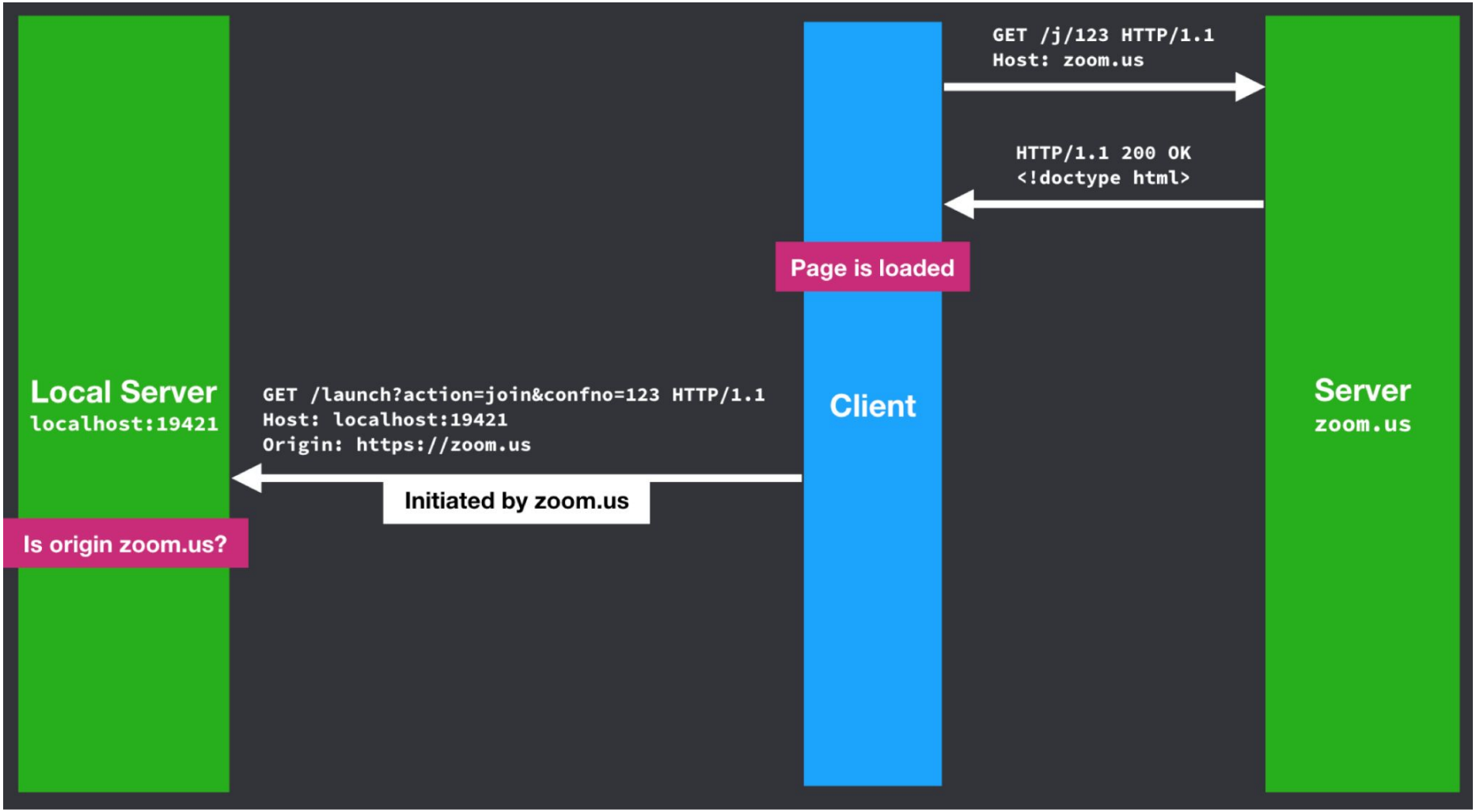
Let's fix the issue

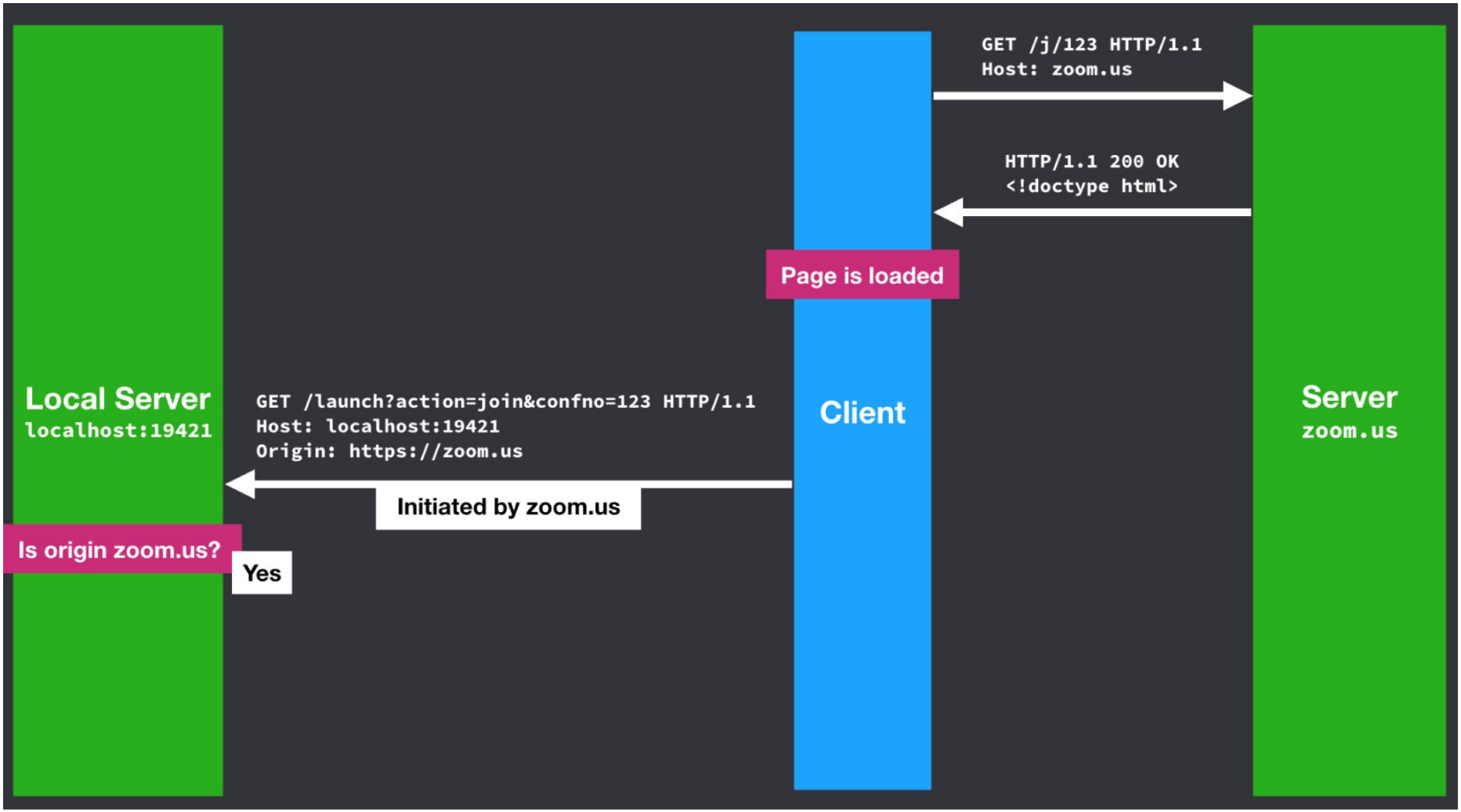
- Best solution: remove the local HTTP server and just register a `zoom://` scheme (protocol handler)
  - https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml
- However, let's assume we need to keep the local HTTP server (probably a bad idea)
- Ideas for securing it:
  - Require user interaction before joining, don't allow host to automatically enable video
  - Only allow zoom.us to communicate with the local server

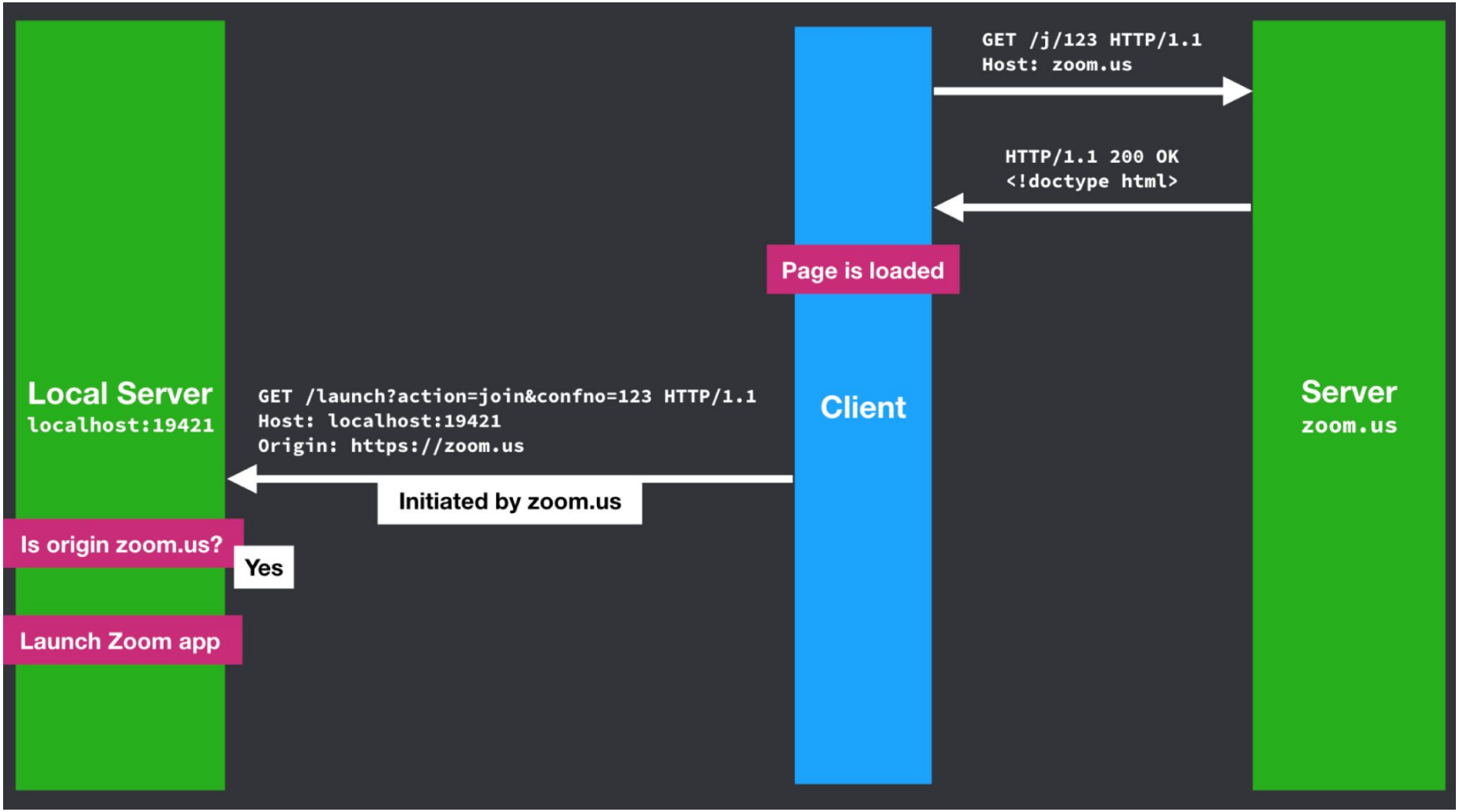
User joins a zoom call (local server inspects `Origin` header)

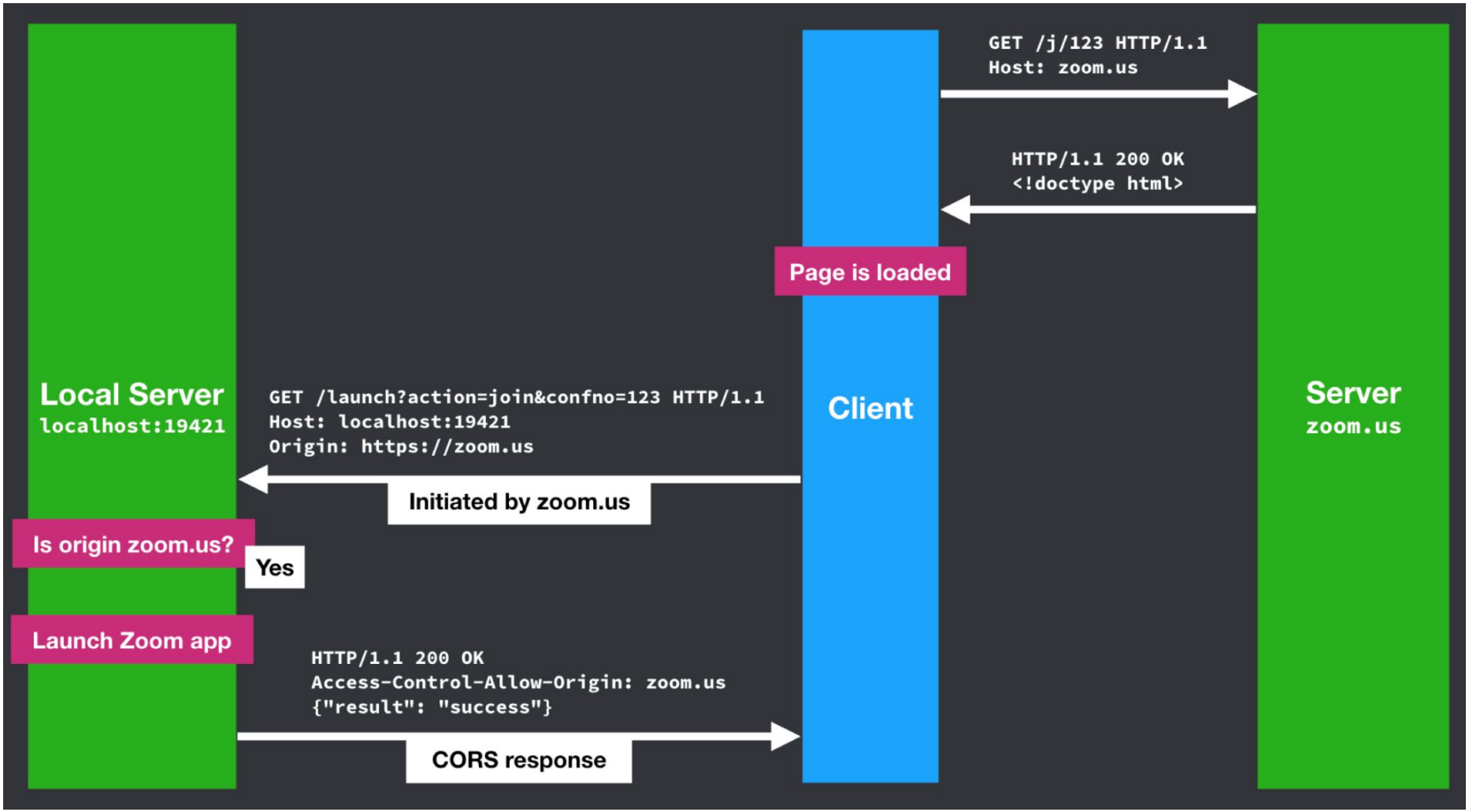

Local Server
localhost:19421
Client
Server
zoom.us
GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Page is loaded

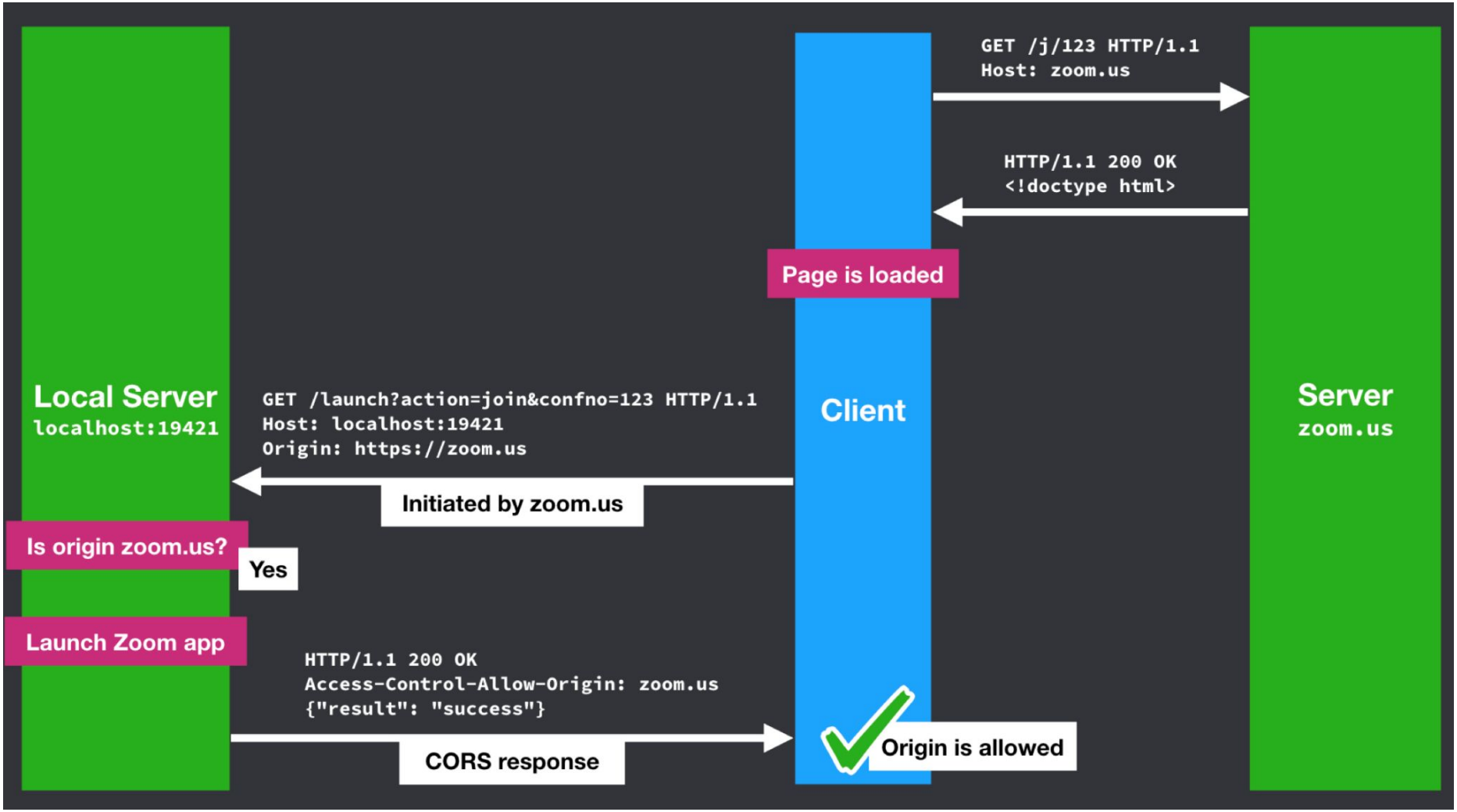

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
GET /launch?action=join&confno=123 HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Initiated by zoom.us
Client
Server
zoom.us
Reminder: the browser prevents Javascript requests from changing the Origin request header! An XHR or fetch() can't lie about it.


GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
GET /launch?action=join&confno=123 HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Client
Server
zoom.us
Initiated by zoom.us
Is origin zoom.us?

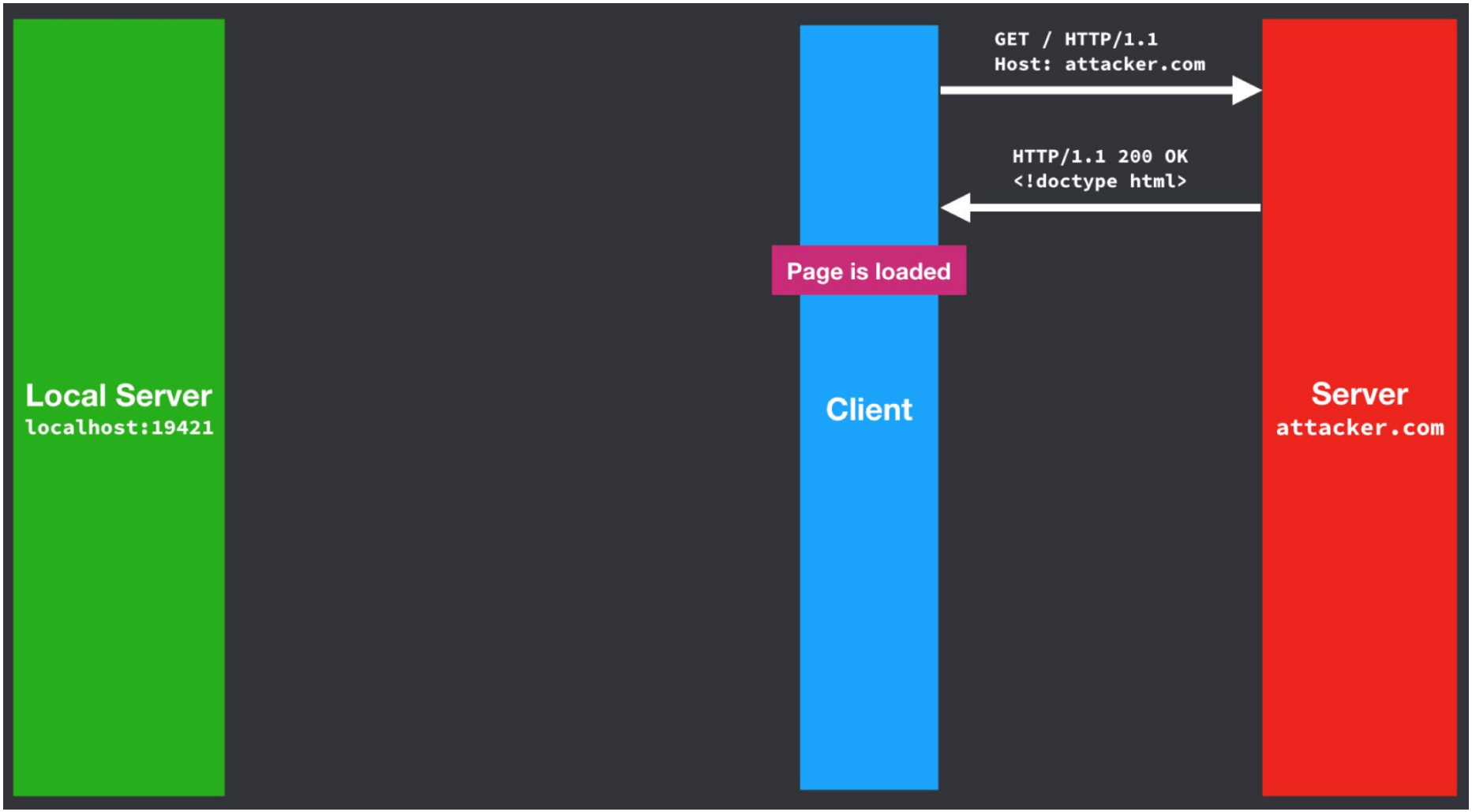

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
GET /launch?action=join&confno=123 HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Client
Server
zoom.us
Initiated by zoom.us
Is origin zoom.us?
Yes

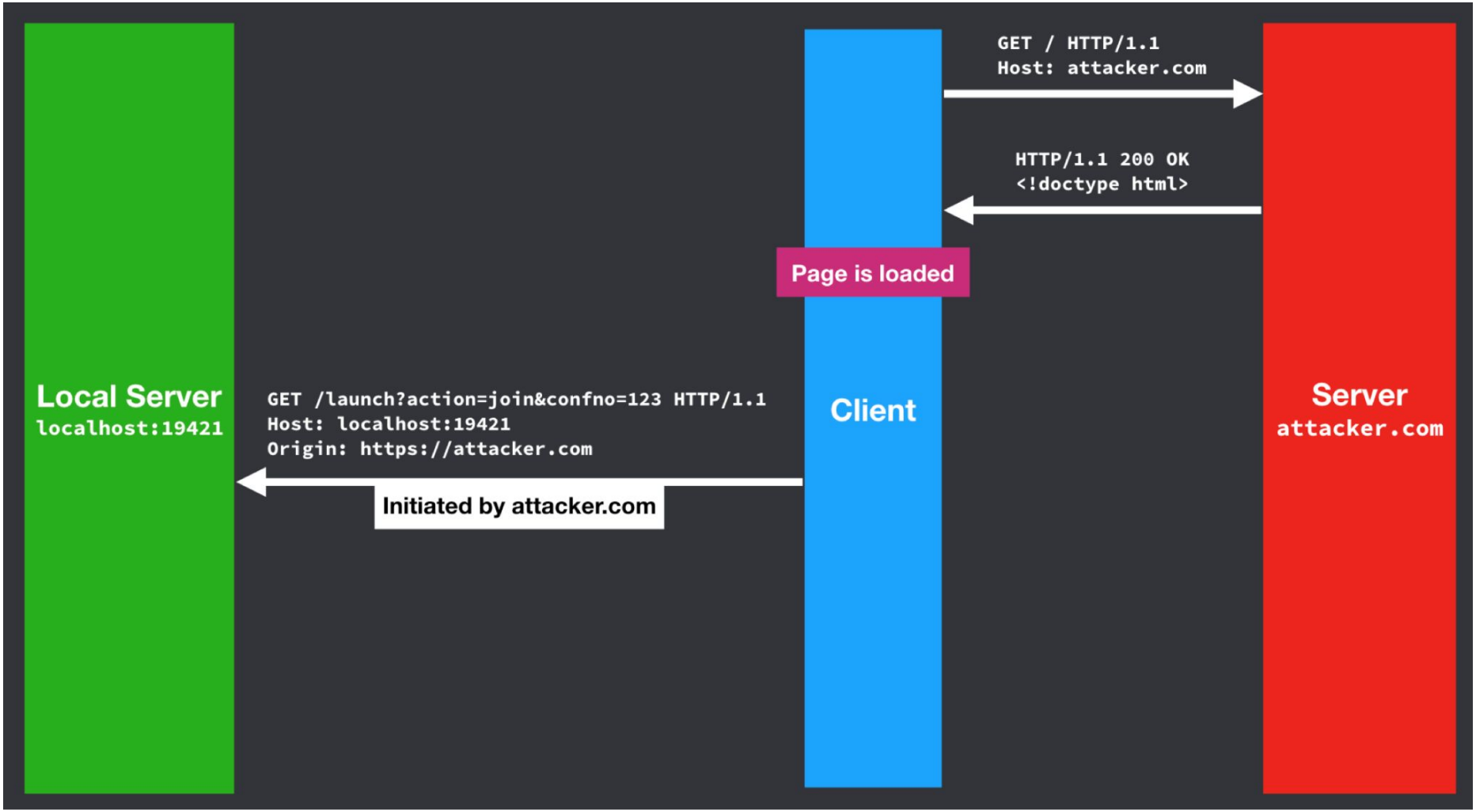

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
GET /launch?action=join&confno=123 HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Client
Server
zoom.us
Initiated by zoom.us
Is origin zoom.us?
Yes
Launch Zoom app

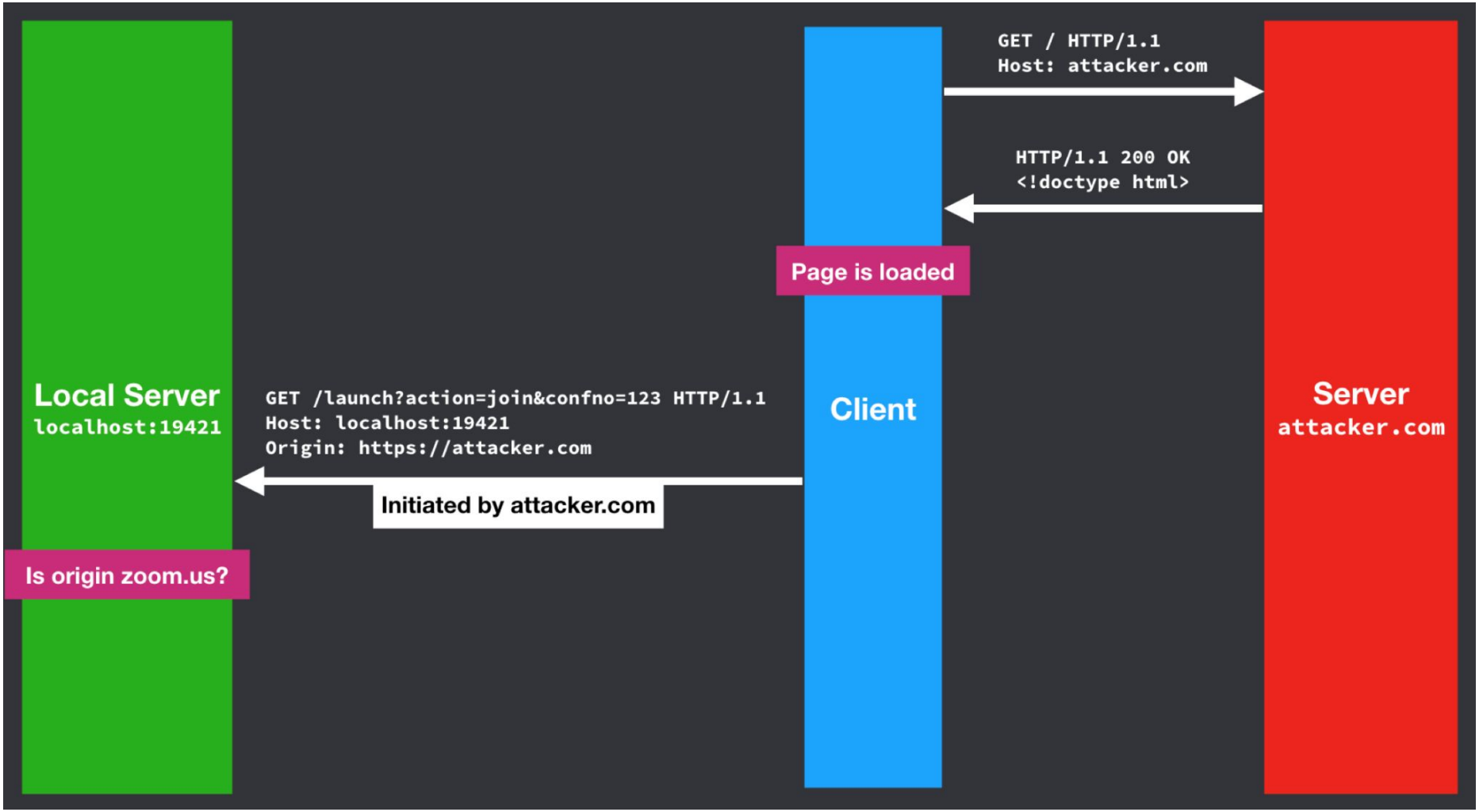

Local Server
localhost:19421
Client
Server
zoom.us
GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
GET /launch?action=join&confno=123 HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Initiated by zoom.us
Is origin zoom.us?
Yes
Launch Zoom app
HTTP/1.1 200 OK
Access-Control-Allow-Origin: zoom.us
{"result": "success"}
CORS response

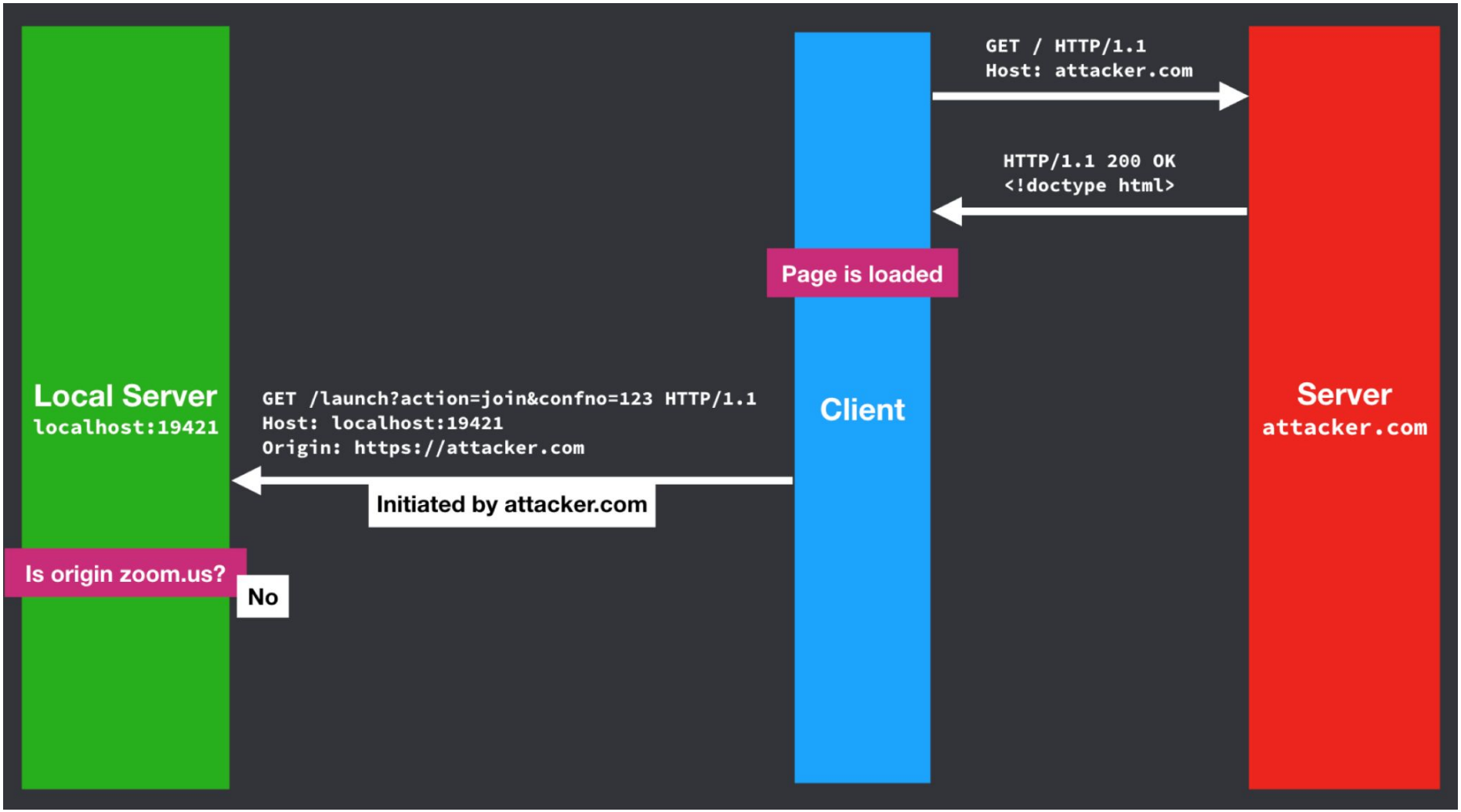

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
GET /launch?action=join&confno=123 HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Client
Server
zoom.us
Initiated by zoom.us
Is origin zoom.us?
Yes
Launch Zoom app
HTTP/1.1 200 OK
Access-Control-Allow-Origin: zoom.us
{"result": "success"}
CORS response
Origin is allowed

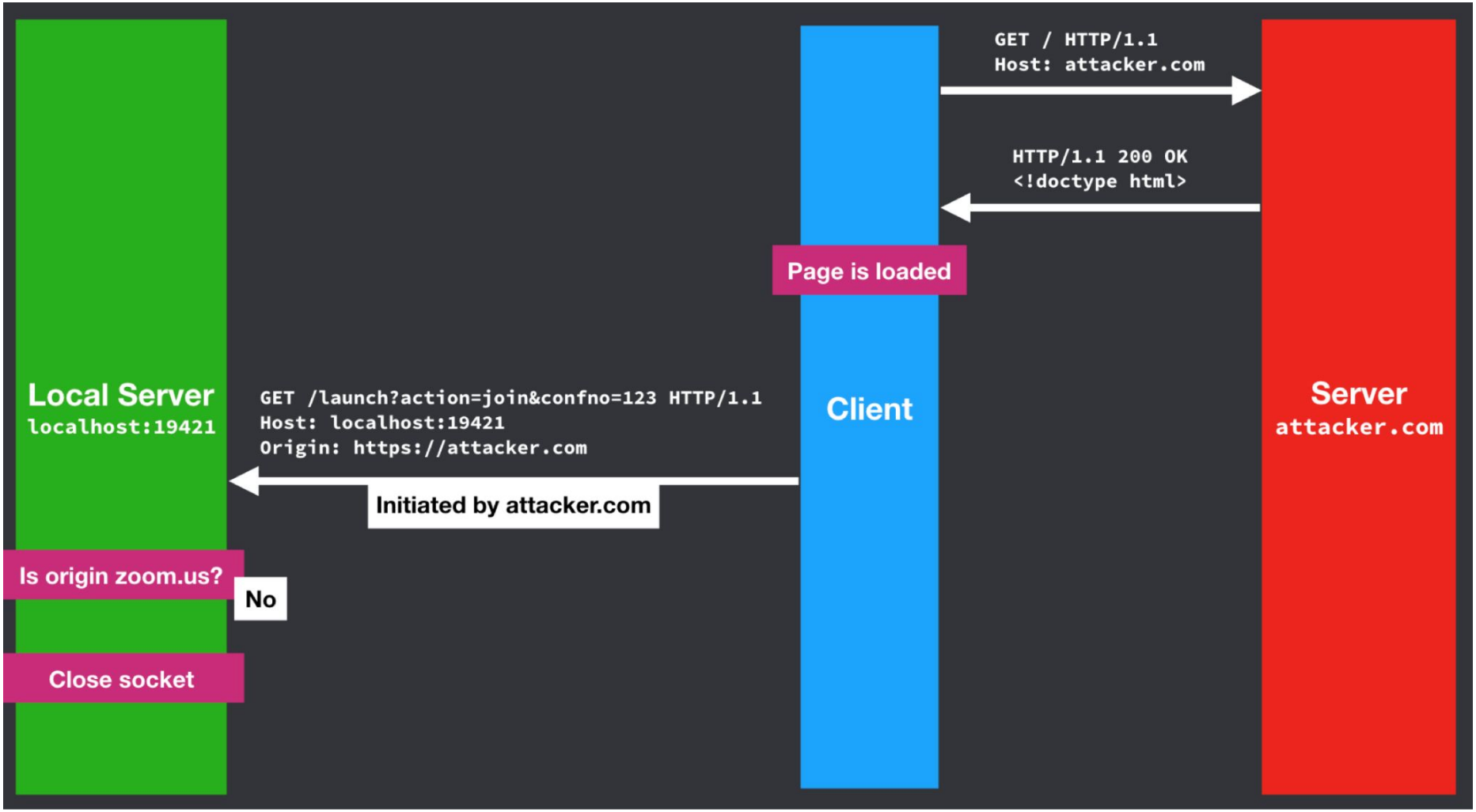
Attacker joins a zoom call (local server inspects `Origin` header)


Local Server
localhost:19421
Client
Server
attacker.com
GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
Page is loaded


Local Server
localhost:19421
GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Client
GET /launch?action=join&confno=123 HTTP/1.1
Host: localhost:19421
Origin: https://attacker.com
Initiated by attacker.com
Server
attacker.com


GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
GET /launch?action=join&confno=123 HTTP/1.1
Host: localhost:19421
Origin: https://attacker.com
Client
Server
attacker.com
Initiated by attacker.com
Is origin zoom.us?


Local Server
localhost:19421
GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
Server
attacker.com
Page is loaded
Client
GET /launch?action=join&confno=123 HTTP/1.1
Host: localhost:19421
Origin: https://attacker.com
Initiated by attacker.com
Is origin zoom.us?
No


GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
Page is loaded
Local Server
localhost:19421
GET /launch?action=join&confno=123 HTTP/1.1
Host: localhost:19421
Origin: https://attacker.com
Client
Server
attacker.com
Initiated by attacker.com
Is origin zoom.us?
No
Close socket

Problems with inspecting `Origin` header

- Problem: Browser doesn't always add `Origin` header
  - for "simple" requests (e.g., <img> or <iframe> tags)
  - for same origin requests (e.g., fetch() to same origin)
  - Very old browsers
- Solution: block requests where `Origin` header is omitted
- Solution: change the endpoint to require a "preflighted" request so that `Origin` header is always sent (e.g., change the HTTP method to PUT)

"Simple" HTTP requests

- An HTTP/1.1 GET, HEAD or a POST is the request method
- In the case of a POST, the Content-Type of the request body is one of `application/x-www-form-urlencoded`, `multipart/form-data`, or `text/plain`
- No custom HTTP headers are set (or, only CORS-safelisted headers are set)
- In other words, “simple” requests are requests that your browser can make *without* using Javascript

"Preflighted" HTTP requests

- Before a "preflighted" requests can be sent to the target server, the browser must check that it is safe to send
- So it first sends an HTTP request with the `OPTIONS` method to the same URL

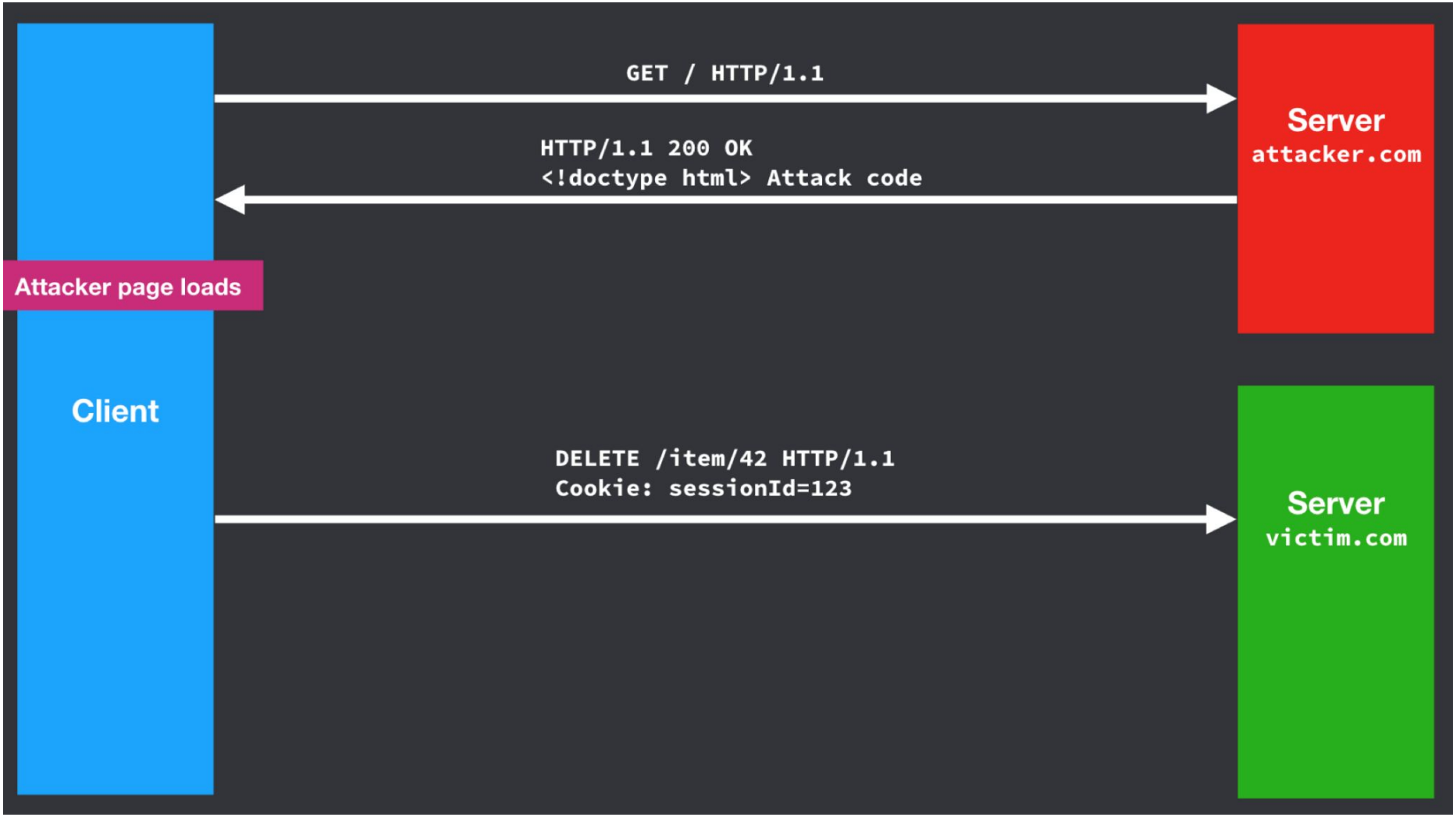
What happens if the browser does not preflight “non-simple” requests?

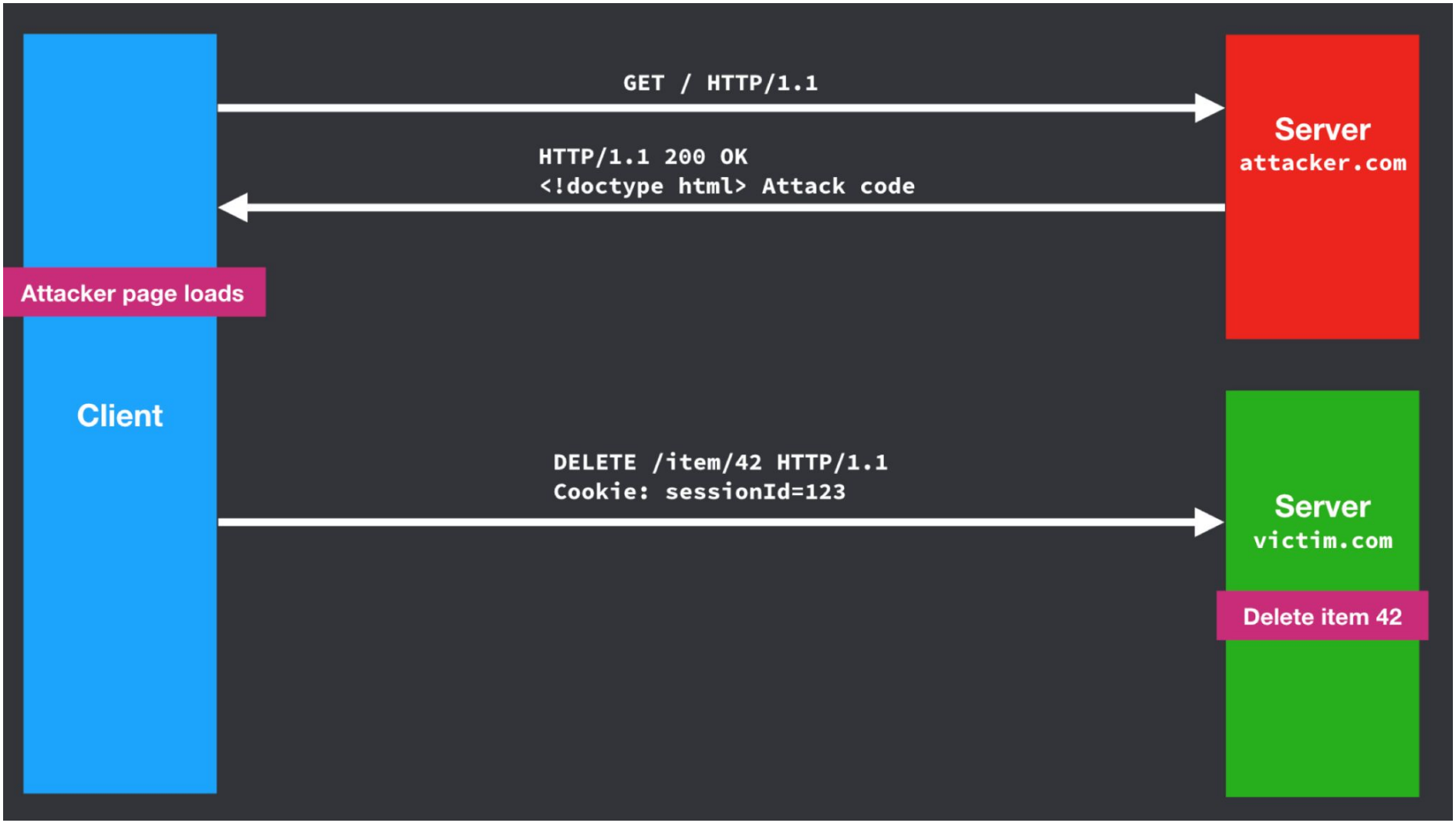

Client
Server
attacker.com
Server
victim.com


GET / HTTP/1.1
Client
Server
attacker.com
Server
victim.com


GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html> Attack code
Client
Server
attacker.com
Server
victim.com


GET / HTTP/1.1
HTTP/1.1 200 OK
<!doctype html> Attack code
Server
attacker.com
Attacker page loads
Client
Server
victim.com


GET / HTTP/1.1
Server
attacker.com
HTTP/1.1 200 OK
<!doctype html> Attack code
Attacker page loads
Client
DELETE /item/42 HTTP/1.1
Cookie: sessionId=123
Server
victim.com


GET / HTTP/1.1
Server
attacker.com
HTTP/1.1 200 OK
<!doctype html> Attack code
Attacker page loads
Client
DELETE /item/42 HTTP/1.1
Cookie: sessionId=123
Server
victim.com
Delete item 42


GET / HTTP/1.1
Server
attacker.com
HTTP/1.1 200 OK
<!doctype html> Attack code
Attacker page loads
Client
DELETE /item/42 HTTP/1.1
Cookie: sessionId=123
Server
victim.com
HTTP/1.1 200 OK
<!doctype html> Deleted!
Access-Control-Allow-Origin: victim.com
Delete item 42


GET / HTTP/1.1
Server
attacker.com
HTTP/1.1 200 OK
<!doctype html> Attack code
Attacker page loads
Client
DELETE /item/42 HTTP/1.1
Cookie: sessionId=123
Server
victim.com
HTTP/1.1 200 OK
<!doctype html> Deleted!
Access-Control-Allow-Origin: victim.com
Delete item 42
Read not allowed
Javascript from attacker.com can't read the response, but the DELETE still happened!

Remember the OPTIONS method?

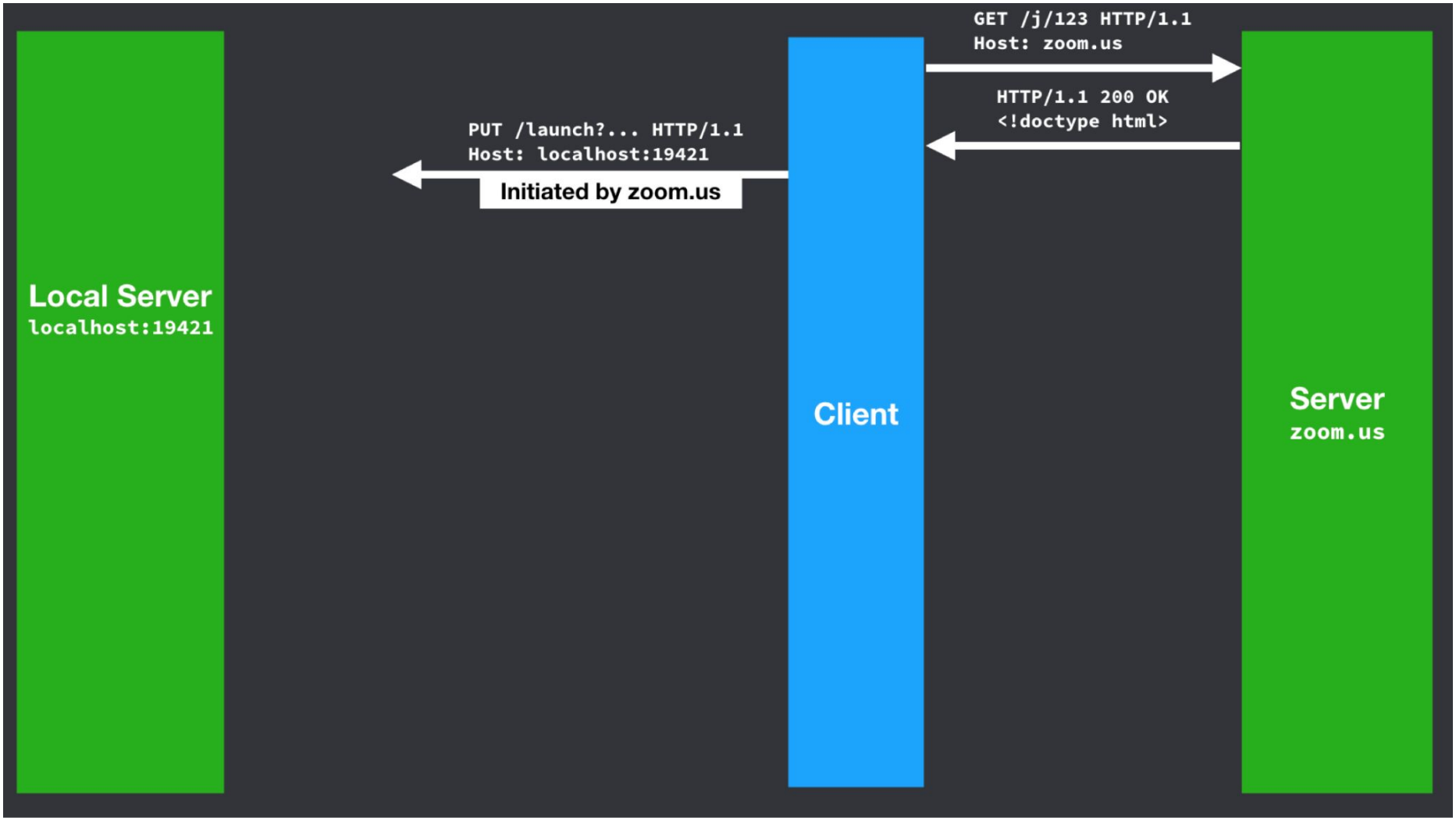
- Browser sends OPTIONS method first to ask the server if the request we want to send is okay
- If server doesn't support OPTIONS (either because it is old or because it doesn't want to support preflighted requests) then, preflighted requests are denied
- Idea: the browser intercepts the potentially dangerous requests and verifies if the server is willing to accept the request before the browser sends it
  - as opposed to: browser sends request without pre-checking and then checks to see if the server denied it (e.g., 403 response). If all the code running on the server was smart, that would be ok too, but we've seen lots of examples where the the code wasn't smart (e.g., 2019 Zoom exploits)

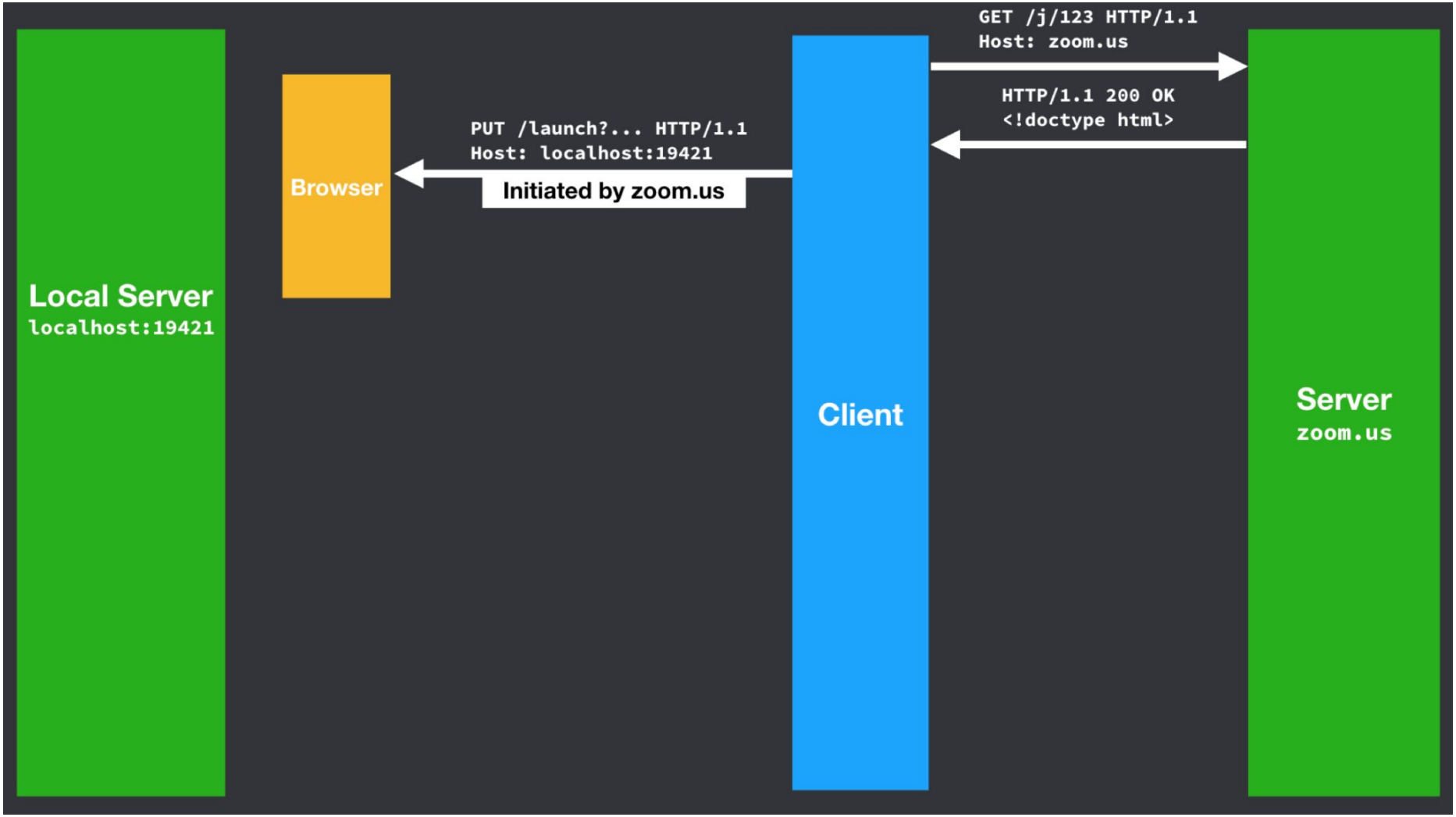
User joins a zoom call (local server requires “preflighted” request

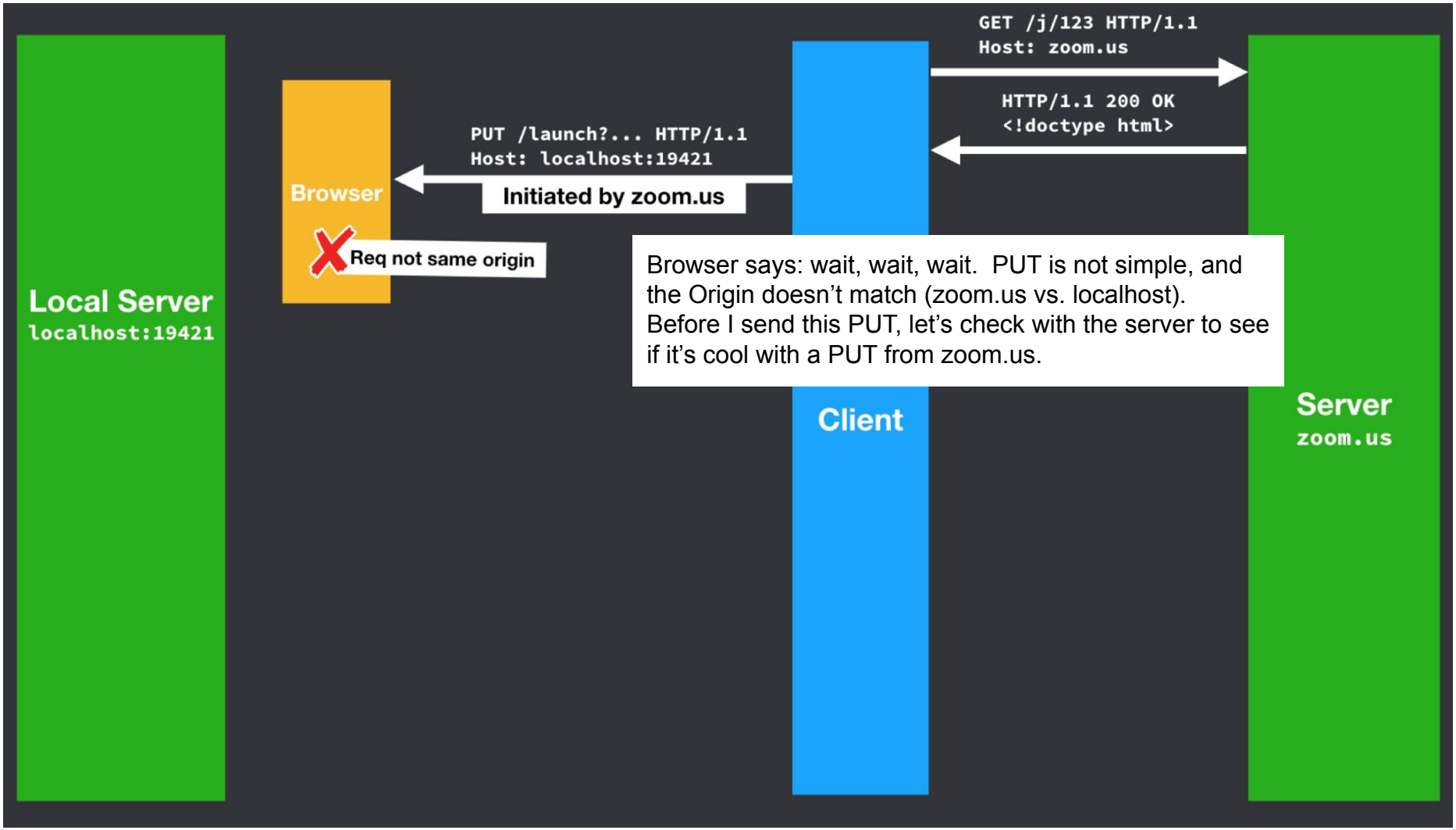

Local Server
localhost:19421
Client
Server
zoom.us

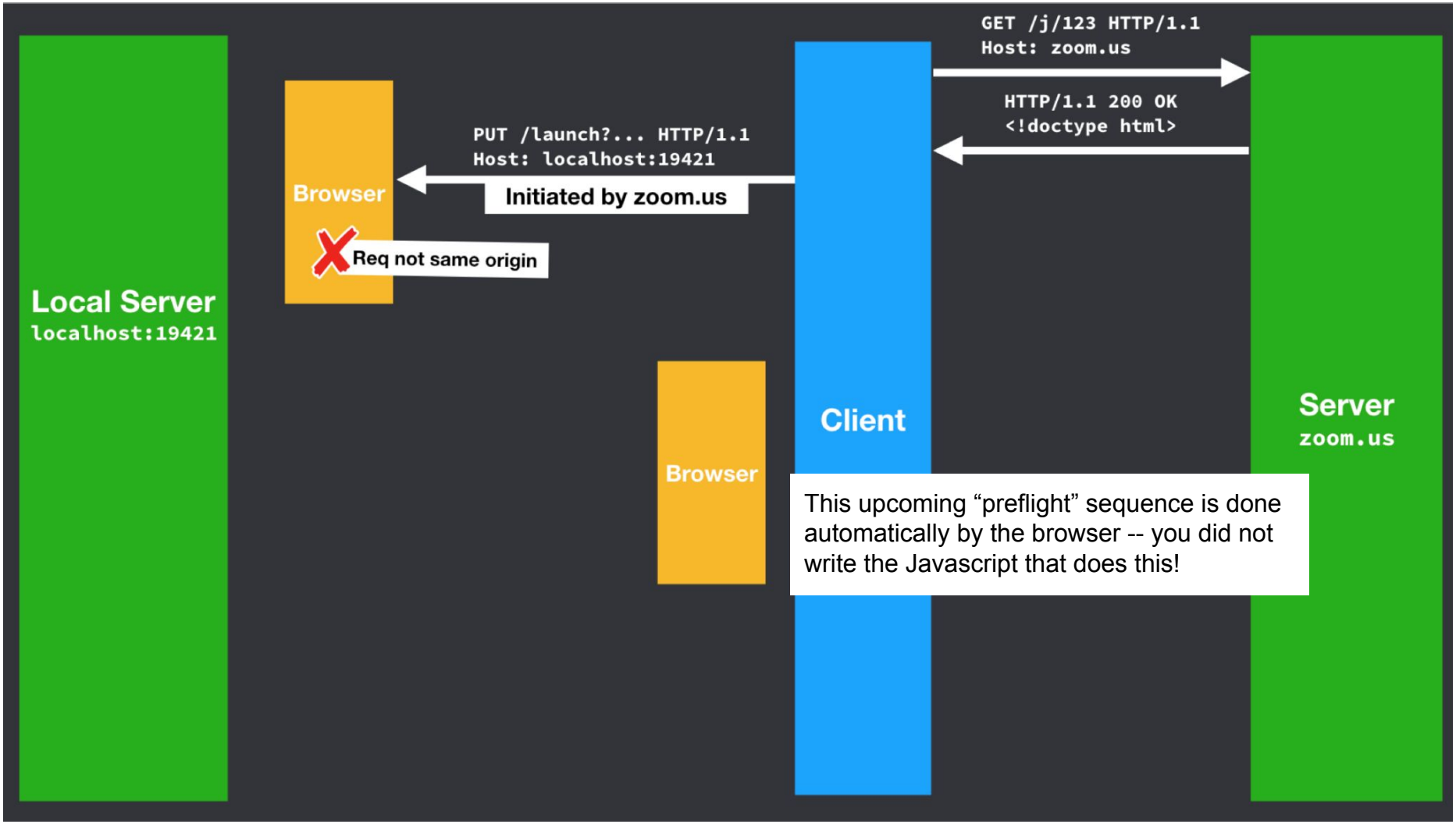

GET /j/123 HTTP/1.1
Host: zoom.us
Local Server
localhost:19421
Client
Server
zoom.us

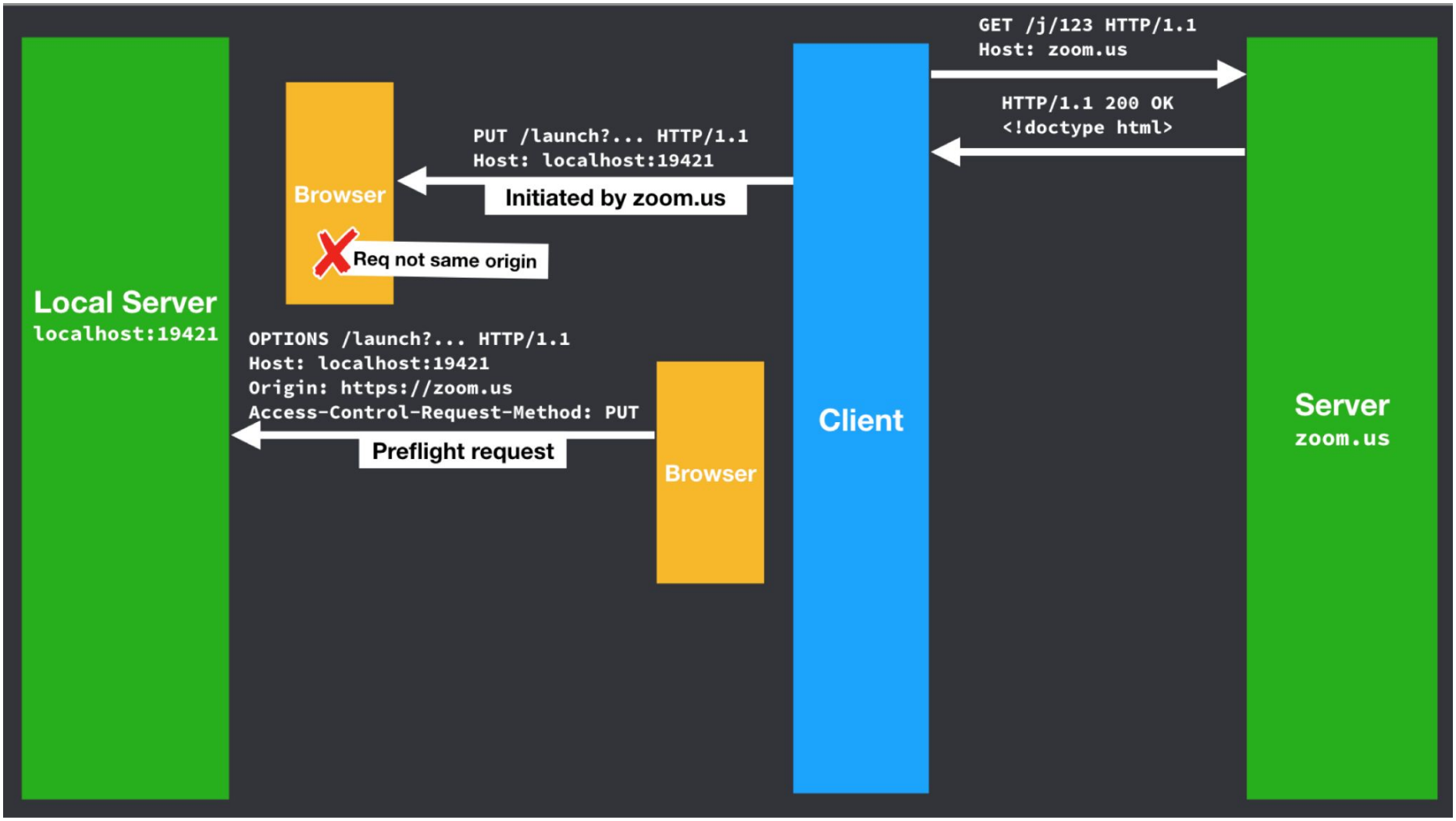

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Local Server
localhost:19421
Client
Server
zoom.us

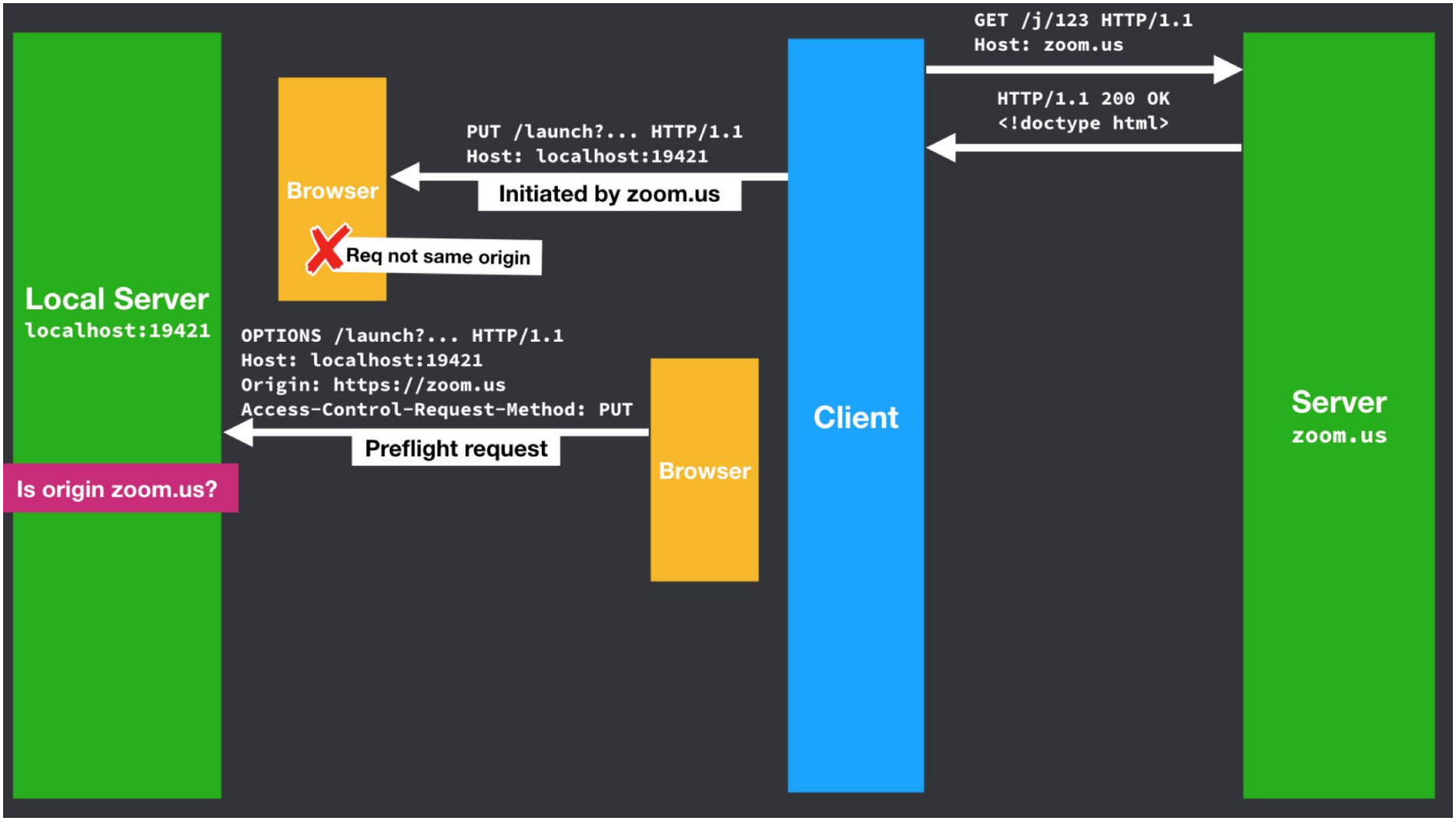

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Local Server
localhost:19421
Client
Server
zoom.us

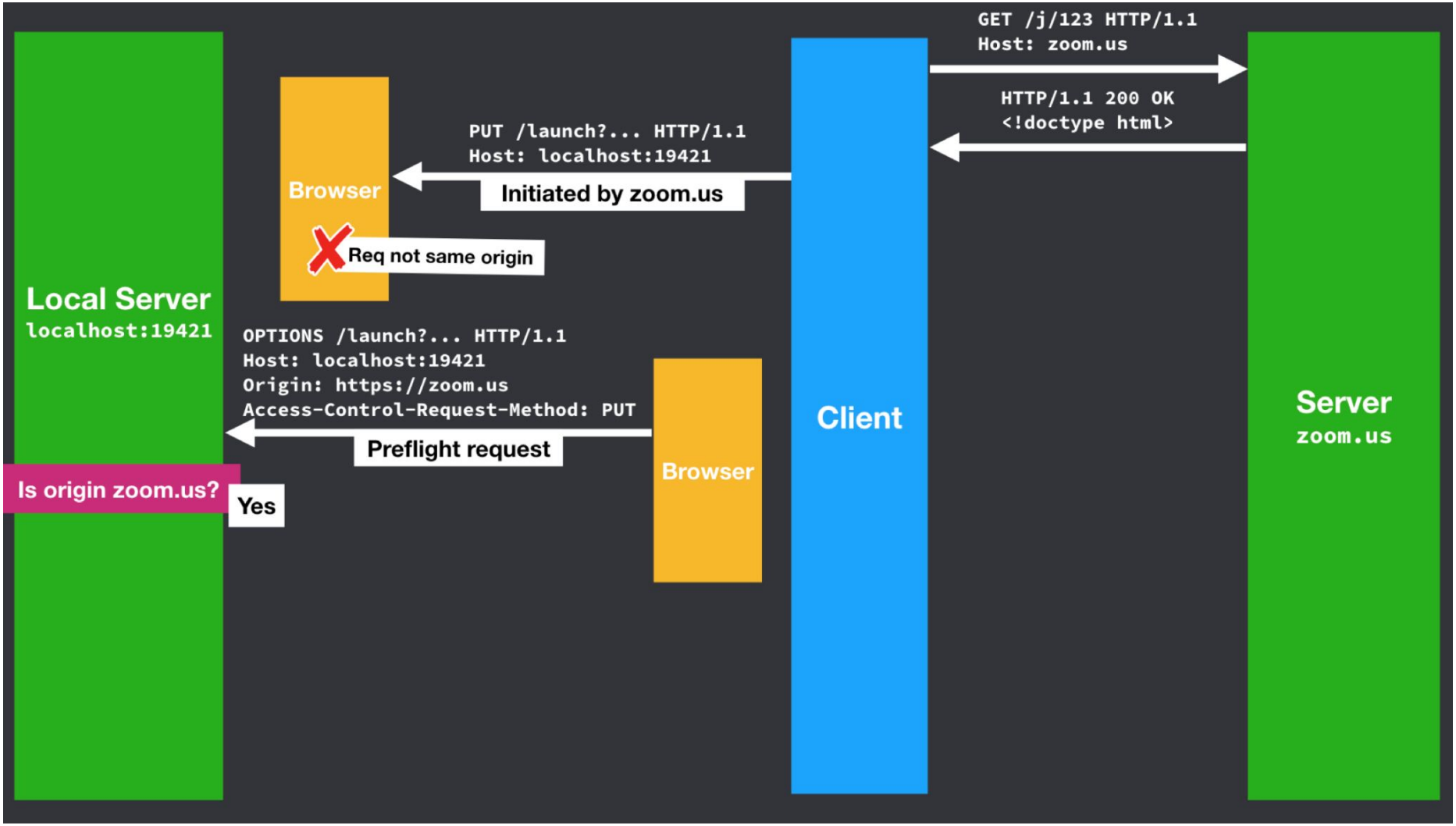

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Browser
Local Server
localhost:19421
Client
Server
zoom.us

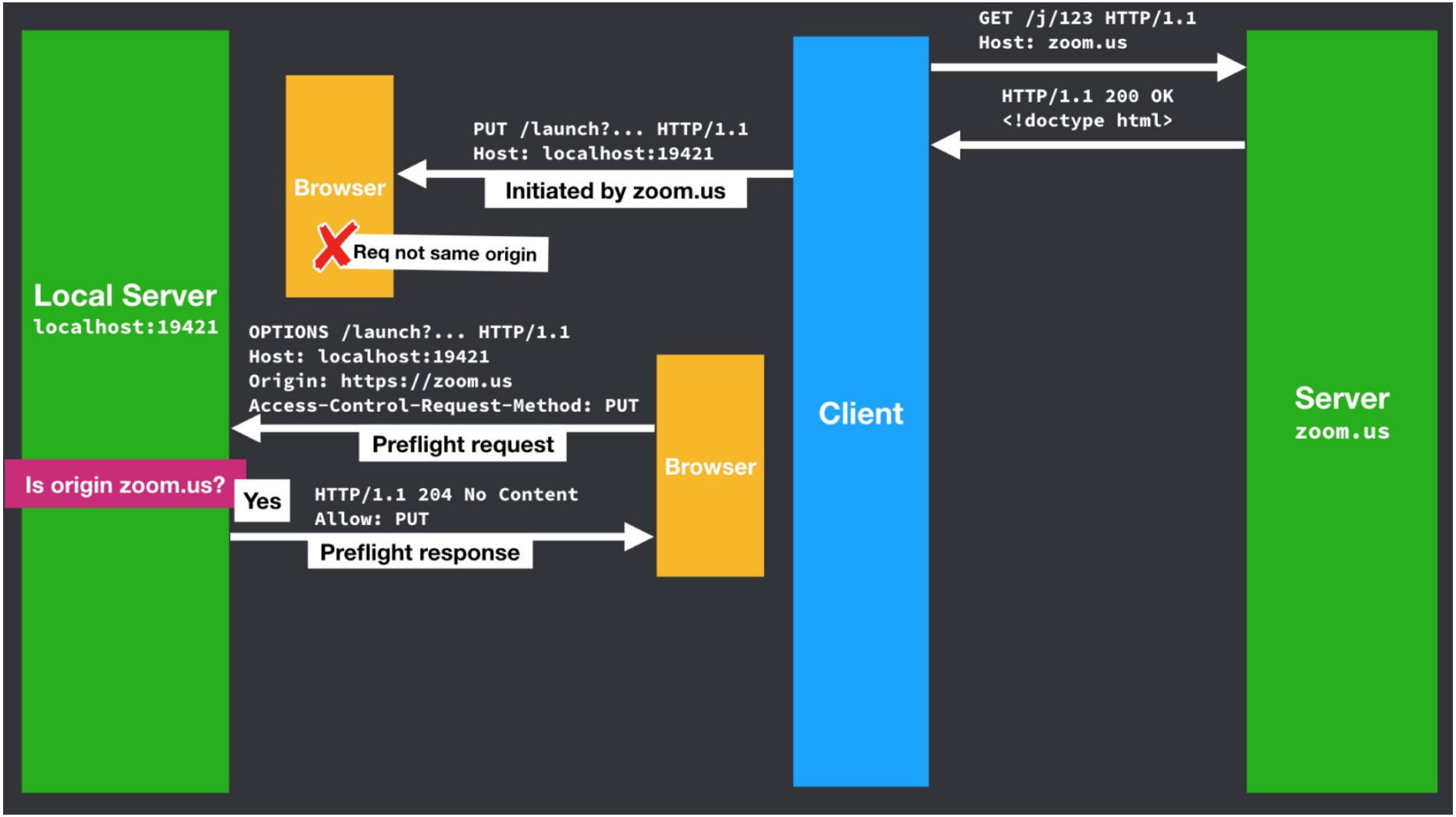

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Browser
Req not same origin
Local Server
localhost:19421
Client
Server
zoom.us
Browser says: wait, wait, wait. PUT is not simple, and the Origin doesn't match (zoom.us vs. localhost). Before I send this PUT, let's check with the server to see if it's cool with a PUT from zoom.us.

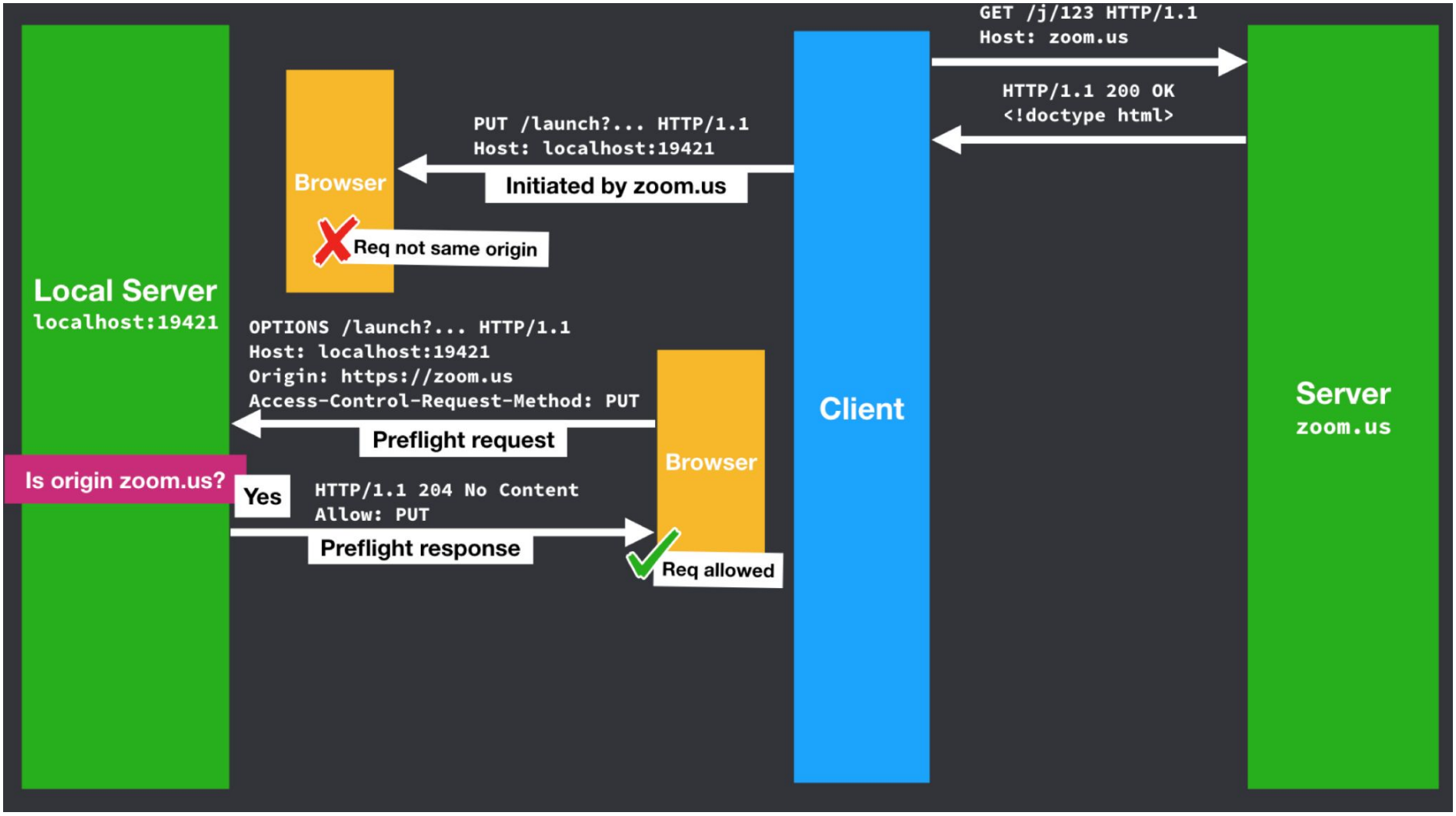

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Browser
Req not same origin
Local Server
localhost:19421
Browser
Client
Server
zoom.us
This upcoming “preflight” sequence is done automatically by the browser -- you did not write the Javascript that does this!

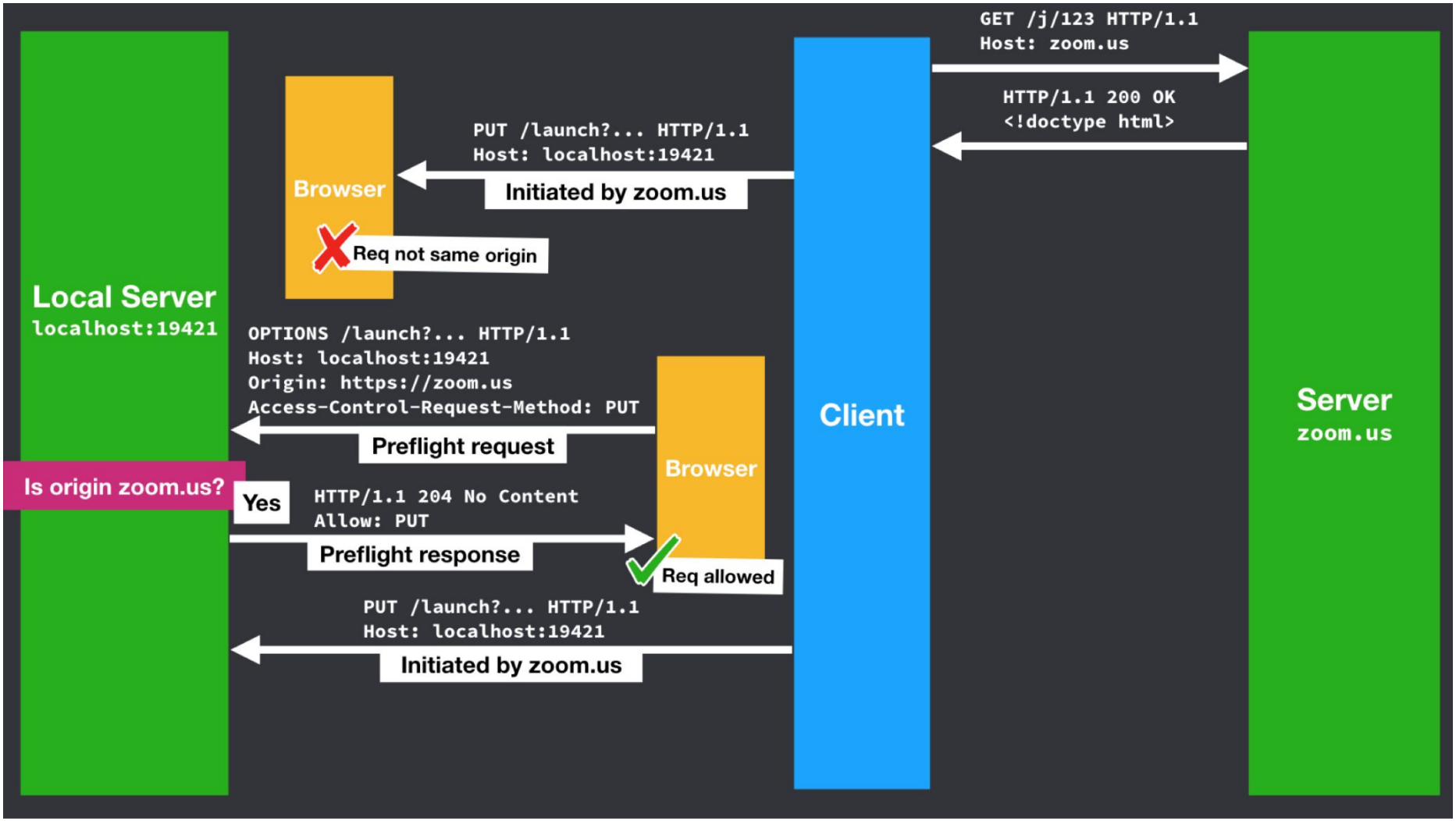

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Server
zoom.us
Client
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Browser
Req not same origin
Local Server
localhost:19421
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Access-Control-Request-Method: PUT
Preflight request
Browser

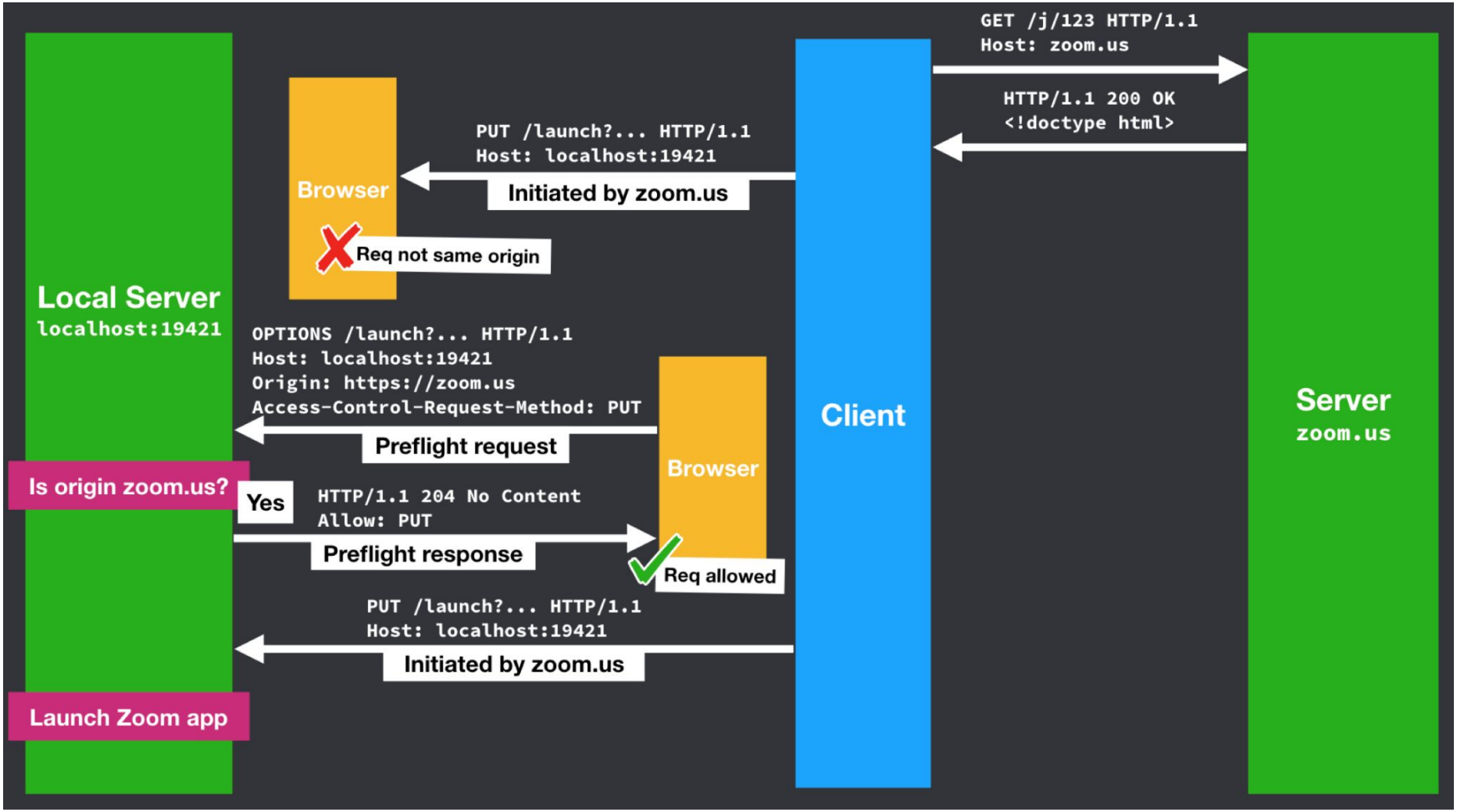

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Server
zoom.us
Client
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Browser
Req not same origin
Local Server
localhost:19421
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Access-Control-Request-Method: PUT
Preflight request
Browser
Is origin zoom.us?

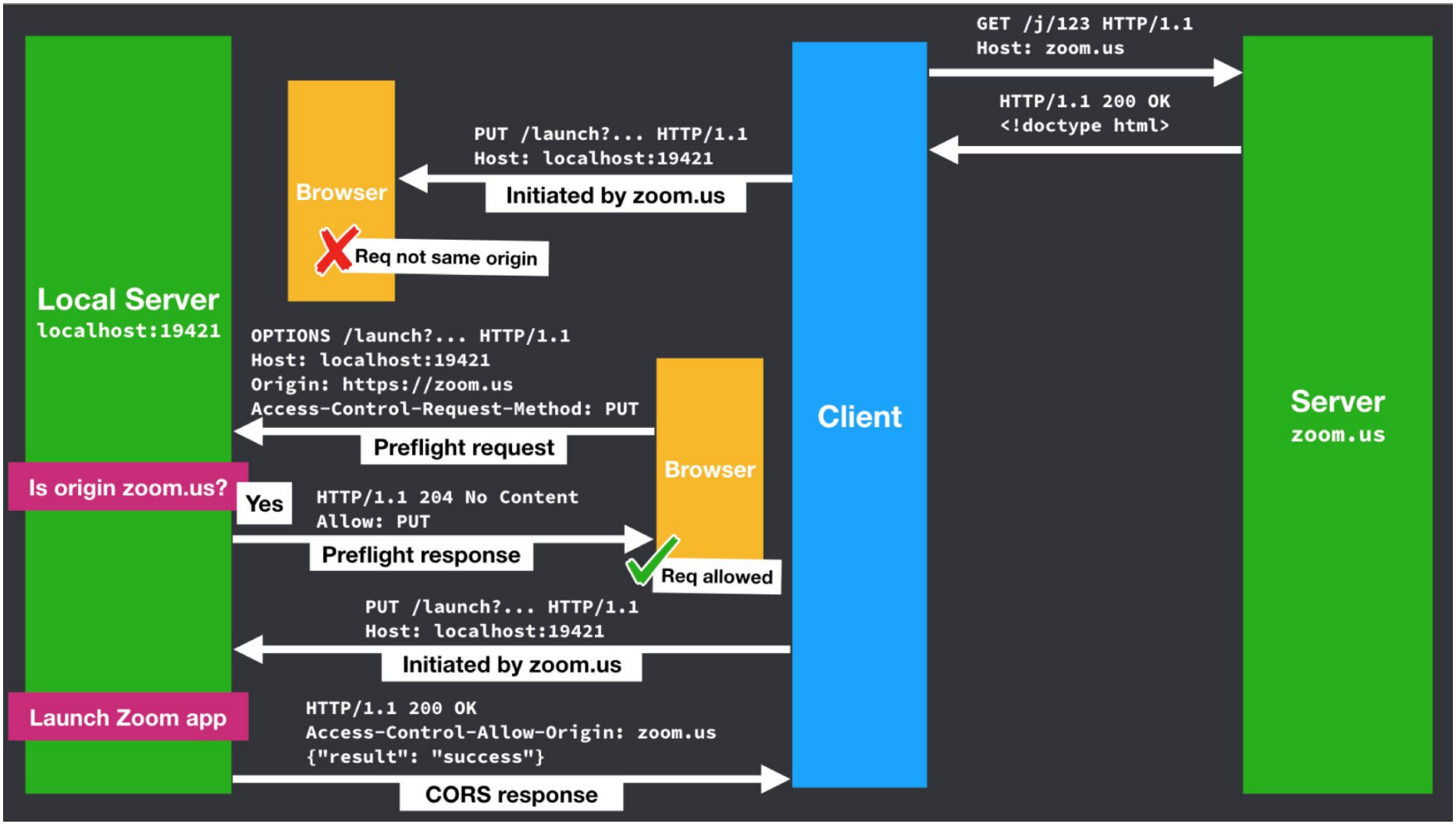

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Server
zoom.us
Client
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Browser
Req not same origin
Local Server
localhost:19421
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Access-Control-Request-Method: PUT
Preflight request
Browser
Is origin zoom.us?
Yes

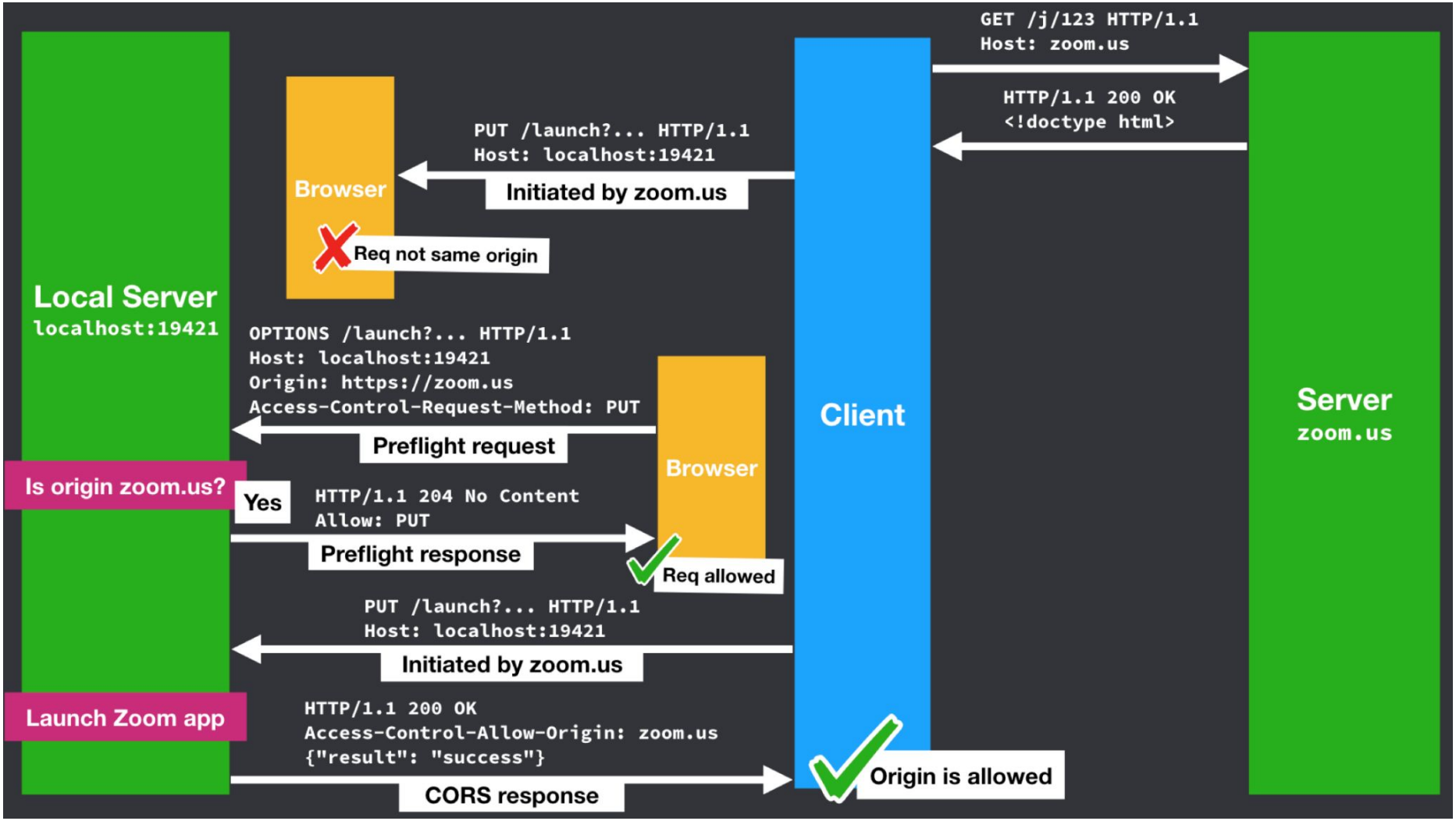

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Server
zoom.us
Client
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Browser
Req not same origin
Local Server
localhost:19421
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Access-Control-Request-Method: PUT
Preflight request
Browser
Is origin zoom.us?
Yes
HTTP/1.1 204 No Content
Allow: PUT
Preflight response


GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Server
zoom.us
Client
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Browser
Req not same origin
Local Server
localhost:19421
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Access-Control-Request-Method: PUT
Preflight request
Browser
Is origin zoom.us?
Yes
HTTP/1.1 204 No Content
Allow: PUT
Preflight response
Req allowed

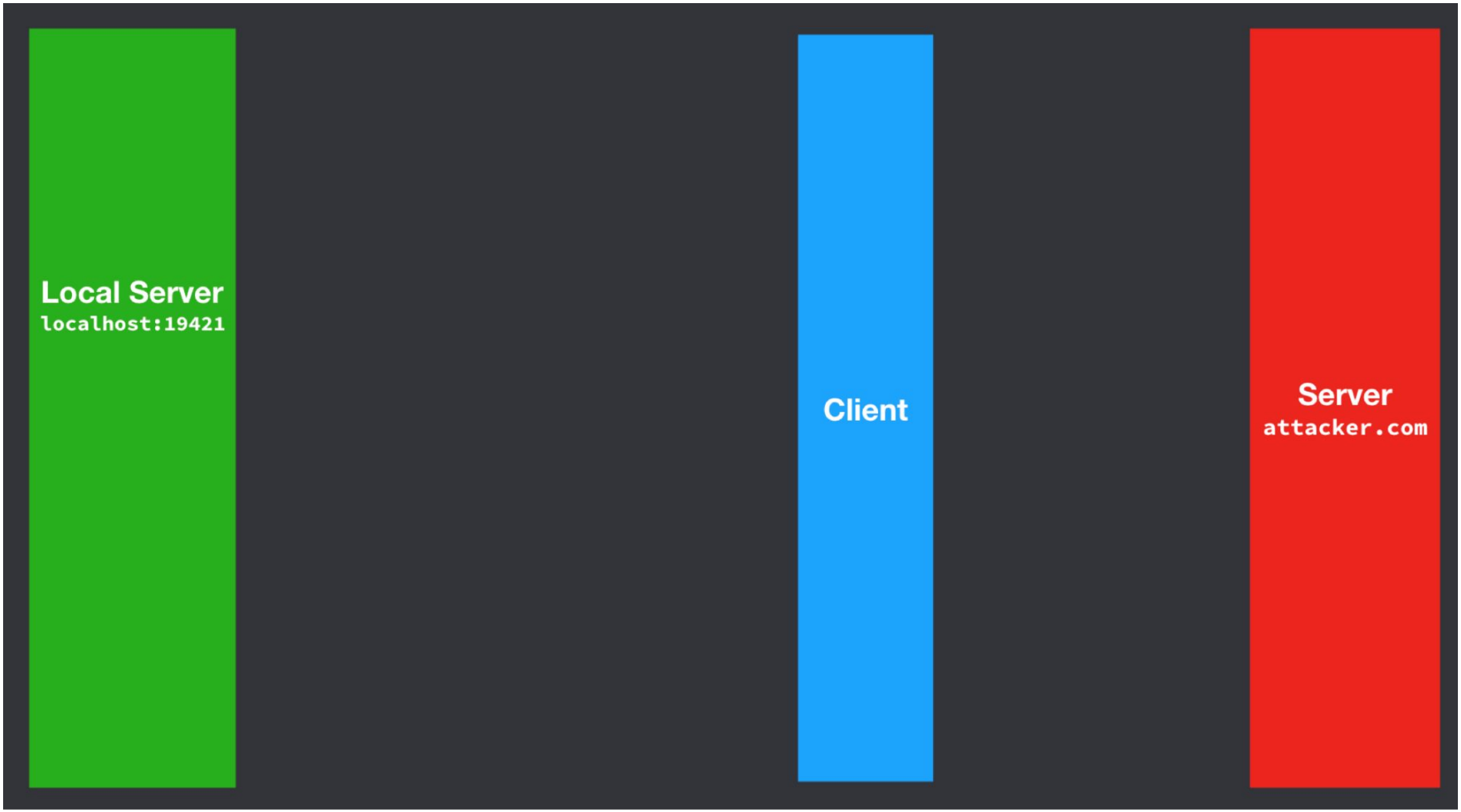

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Server
zoom.us
Client
Local Server
localhost:19421
Browser
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Req not same origin
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Access-Control-Request-Method: PUT
Preflight request
Browser
Is origin zoom.us?
Yes
HTTP/1.1 204 No Content
Allow: PUT
Preflight response
Req allowed
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us


GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Server
zoom.us
Client
Local Server
localhost:19421
Browser
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Req not same origin
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Access-Control-Request-Method: PUT
Preflight request
Browser
Is origin zoom.us?
Yes
HTTP/1.1 204 No Content
Allow: PUT
Preflight response
Req allowed
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Launch Zoom app


GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Server
zoom.us
Client
Browser
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Req not same origin
Local Server
localhost:19421
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Access-Control-Request-Method: PUT
Preflight request
Browser
Is origin zoom.us?
Yes
HTTP/1.1 204 No Content
Allow: PUT
Preflight response
Req allowed
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Launch Zoom app
HTTP/1.1 200 OK
Access-Control-Allow-Origin: zoom.us
{"result": "success"}
CORS response

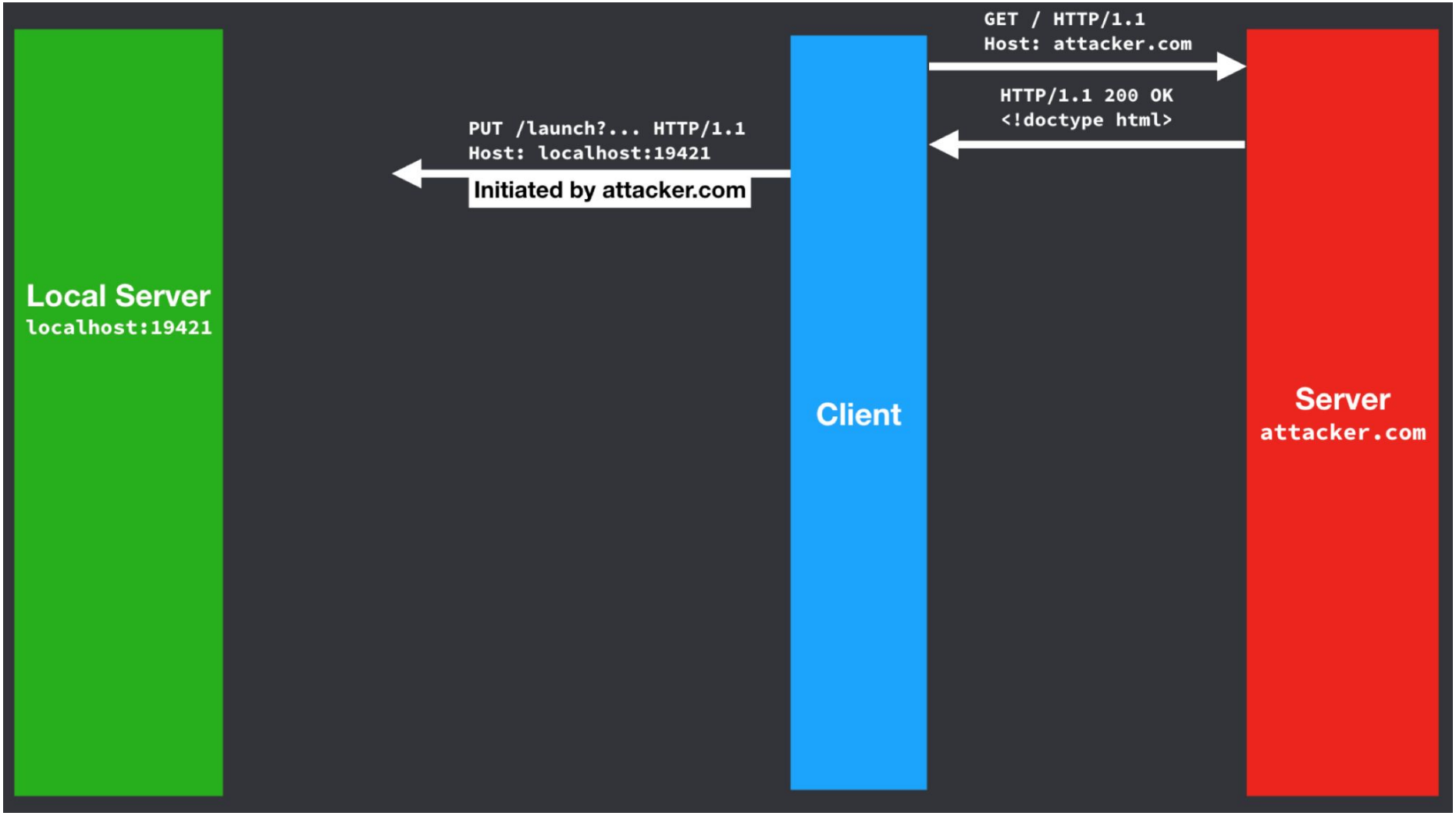

GET /j/123 HTTP/1.1
Host: zoom.us
HTTP/1.1 200 OK
<!doctype html>
Server
zoom.us
Client
Local Server
localhost:19421
Browser
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Req not same origin
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://zoom.us
Access-Control-Request-Method: PUT
Preflight request
Browser
Is origin zoom.us?
Yes
HTTP/1.1 204 No Content
Allow: PUT
Preflight response
Req allowed
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by zoom.us
Launch Zoom app
HTTP/1.1 200 OK
Access-Control-Allow-Origin: zoom.us
{"result": "success"}
CORS response
Origin is allowed

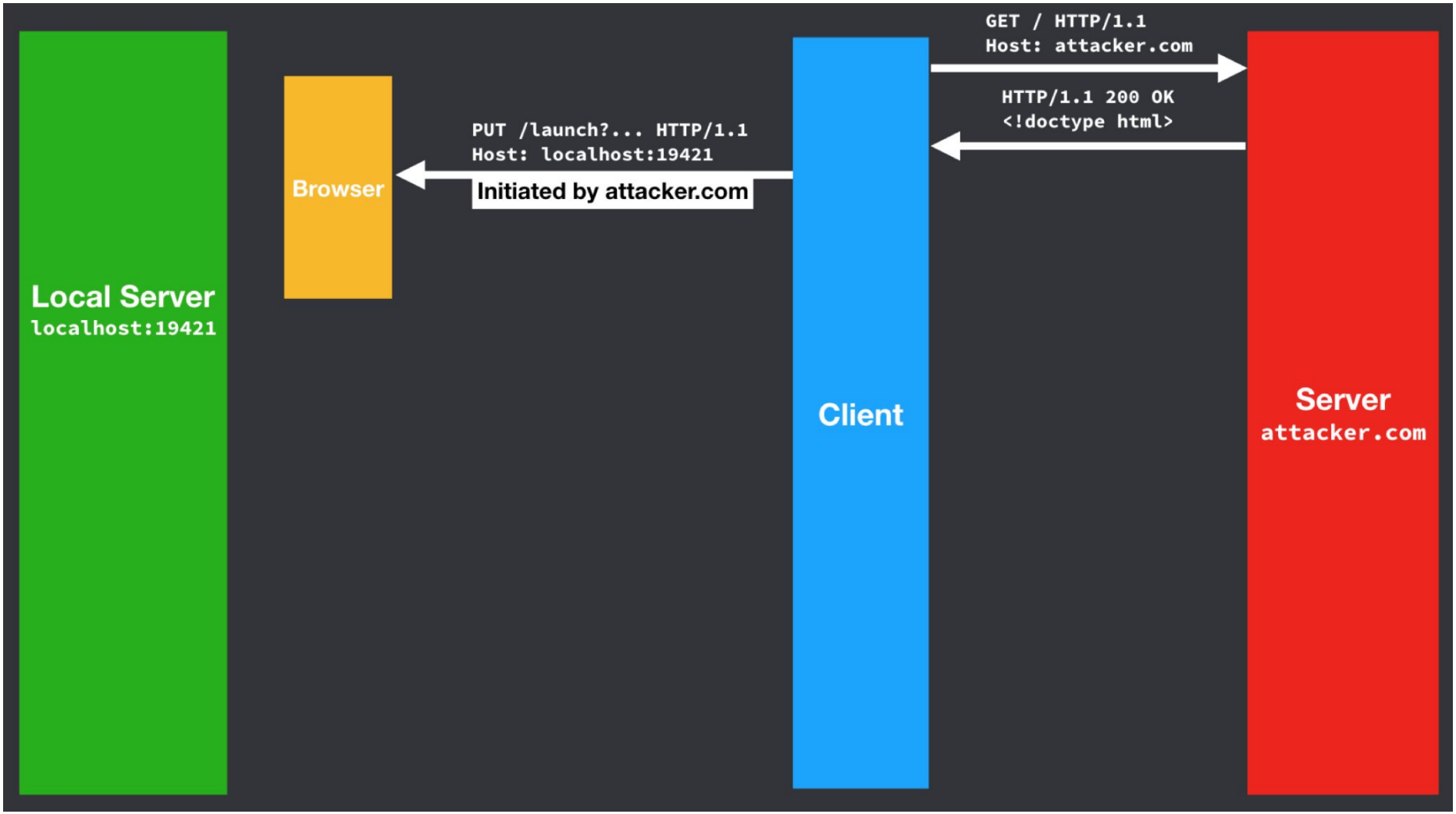
Attacker joins a zoom call (local server requires “preflighted” request

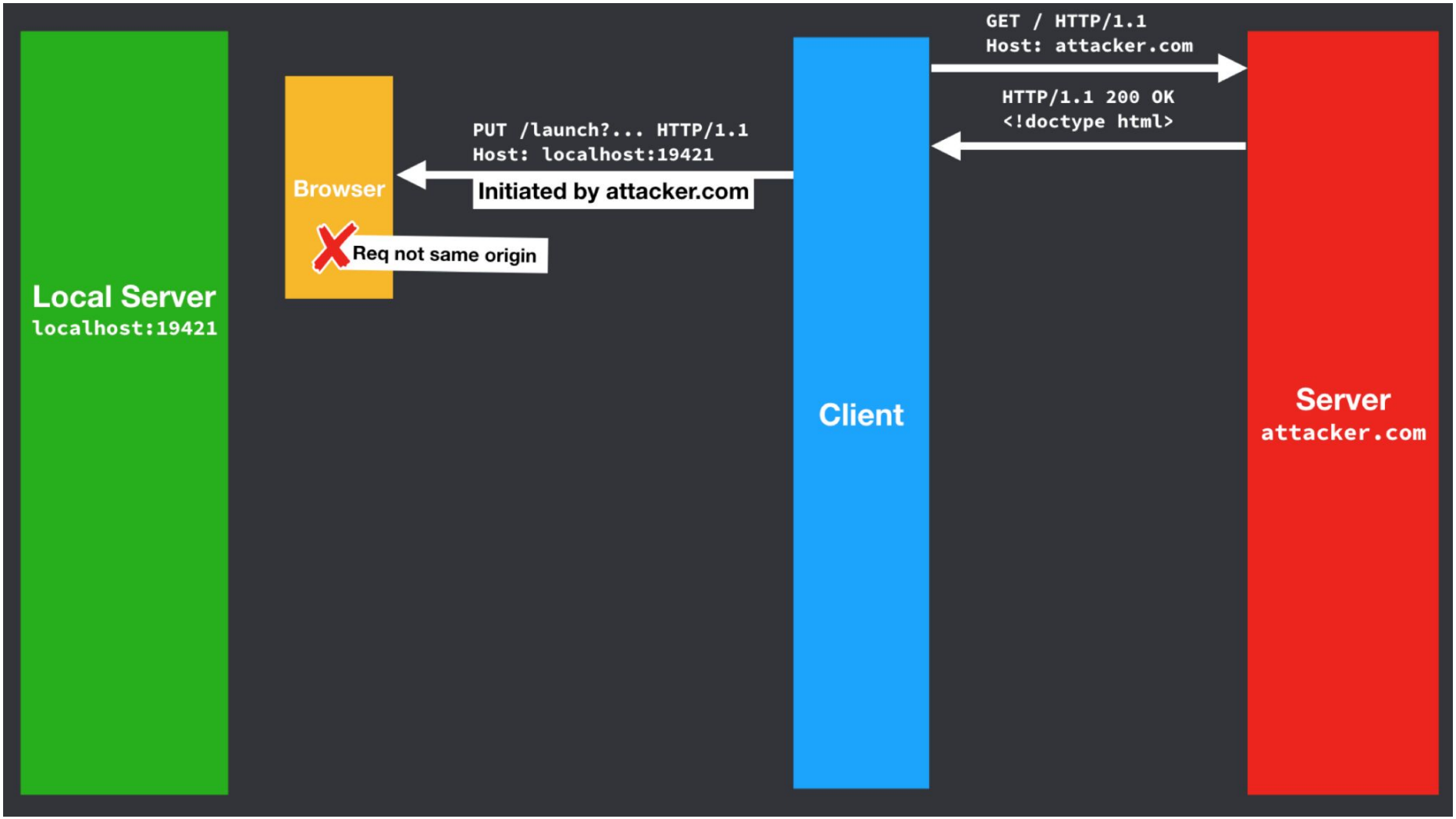

Local Server
localhost:19421
Client
Server
attacker.com

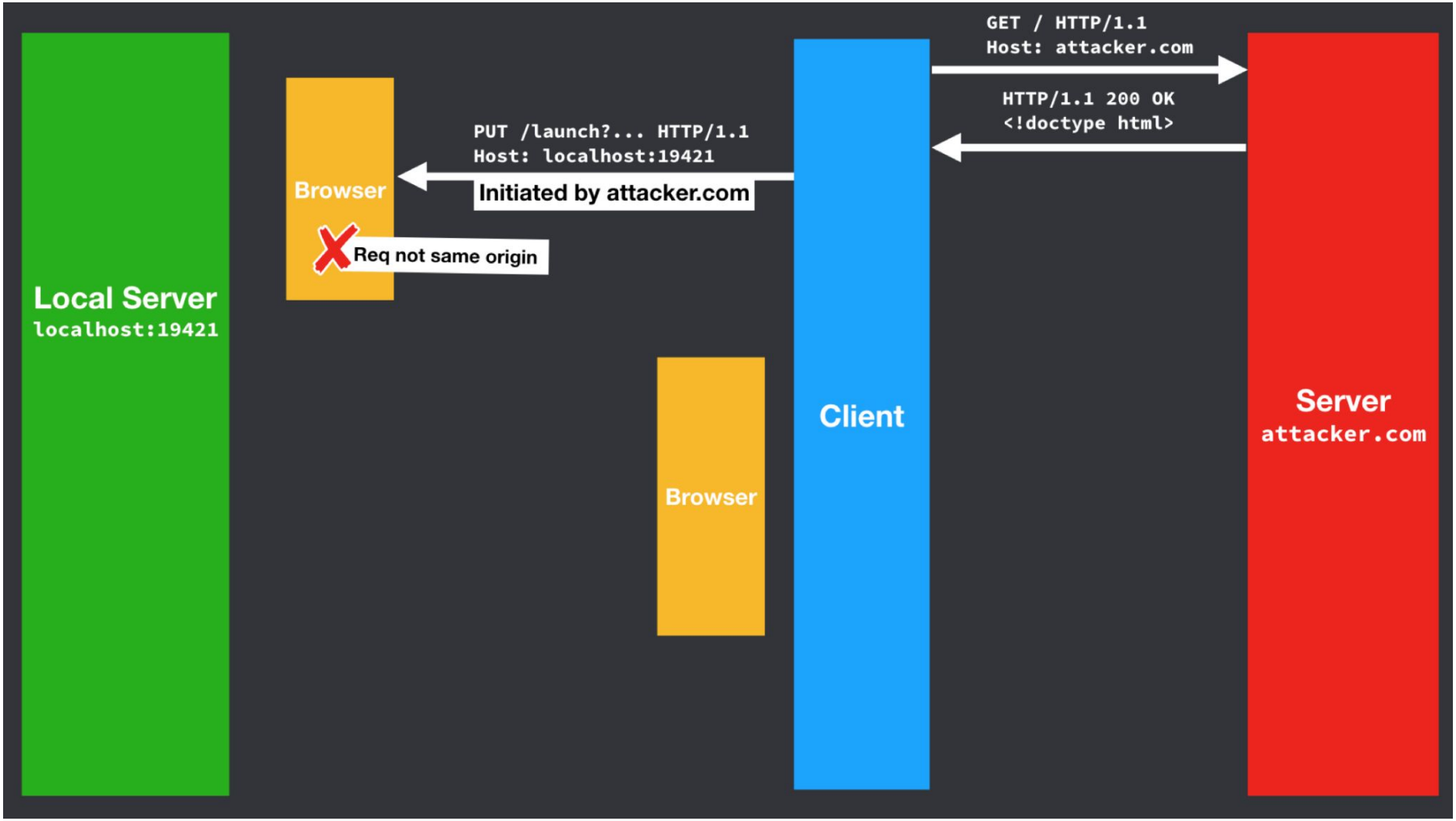

GET / HTTP/1.1
Host: attacker.com
Local Server
localhost:19421
Client
Server
attacker.com

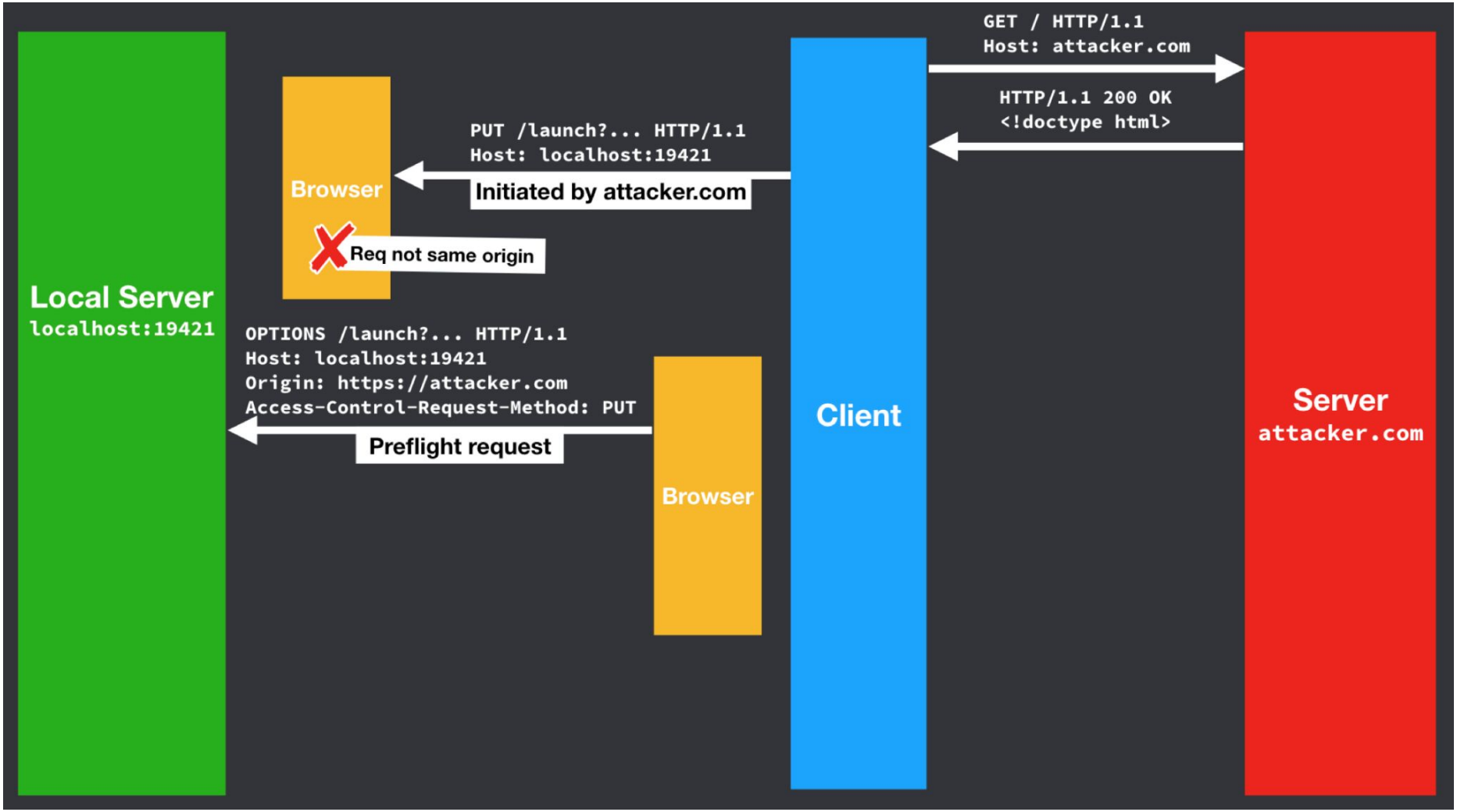

Local Server
localhost:19421
Client
GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
Server
attacker.com

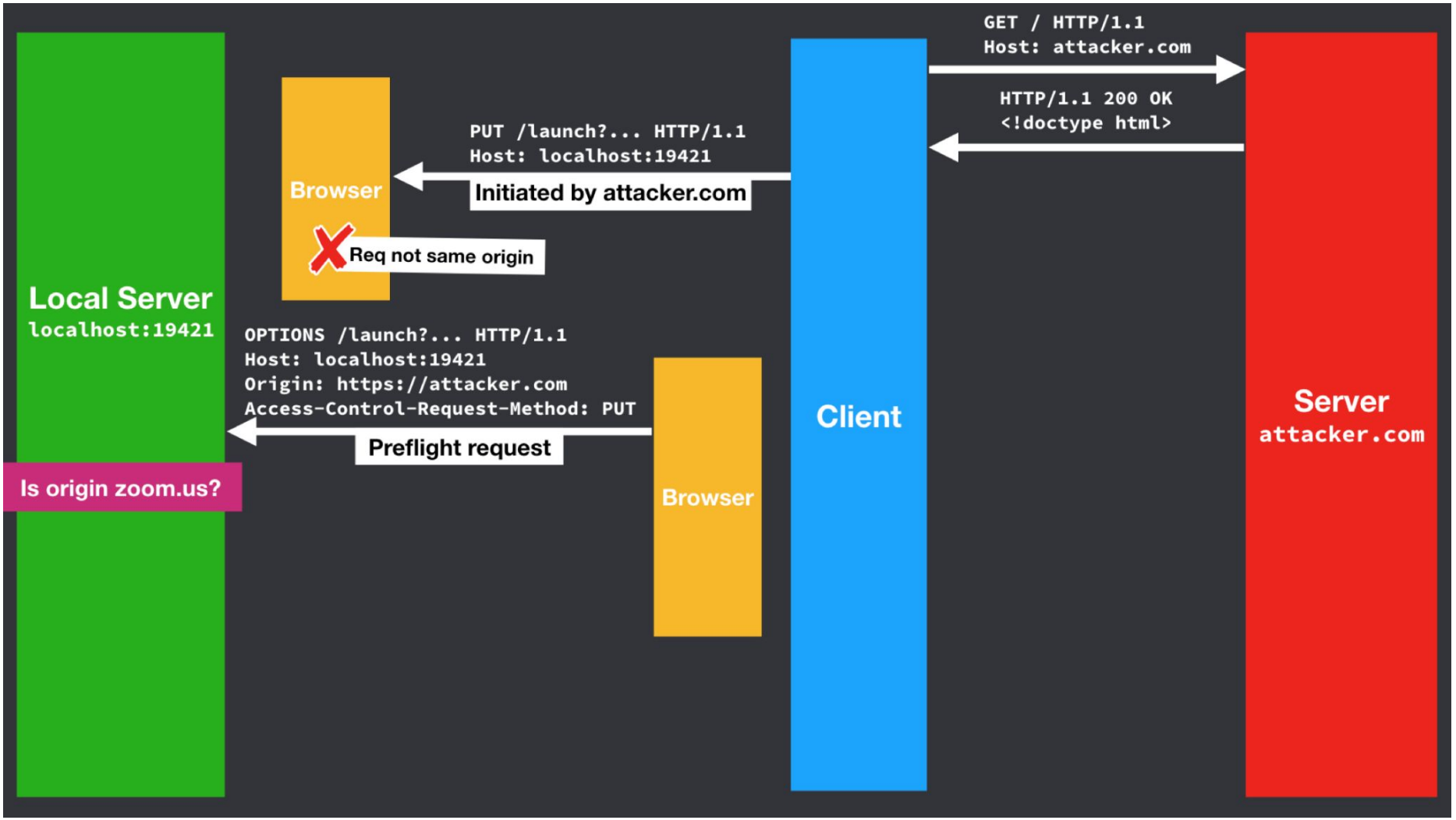

GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by attacker.com
Local Server
localhost:19421
Client
Server
attacker.com

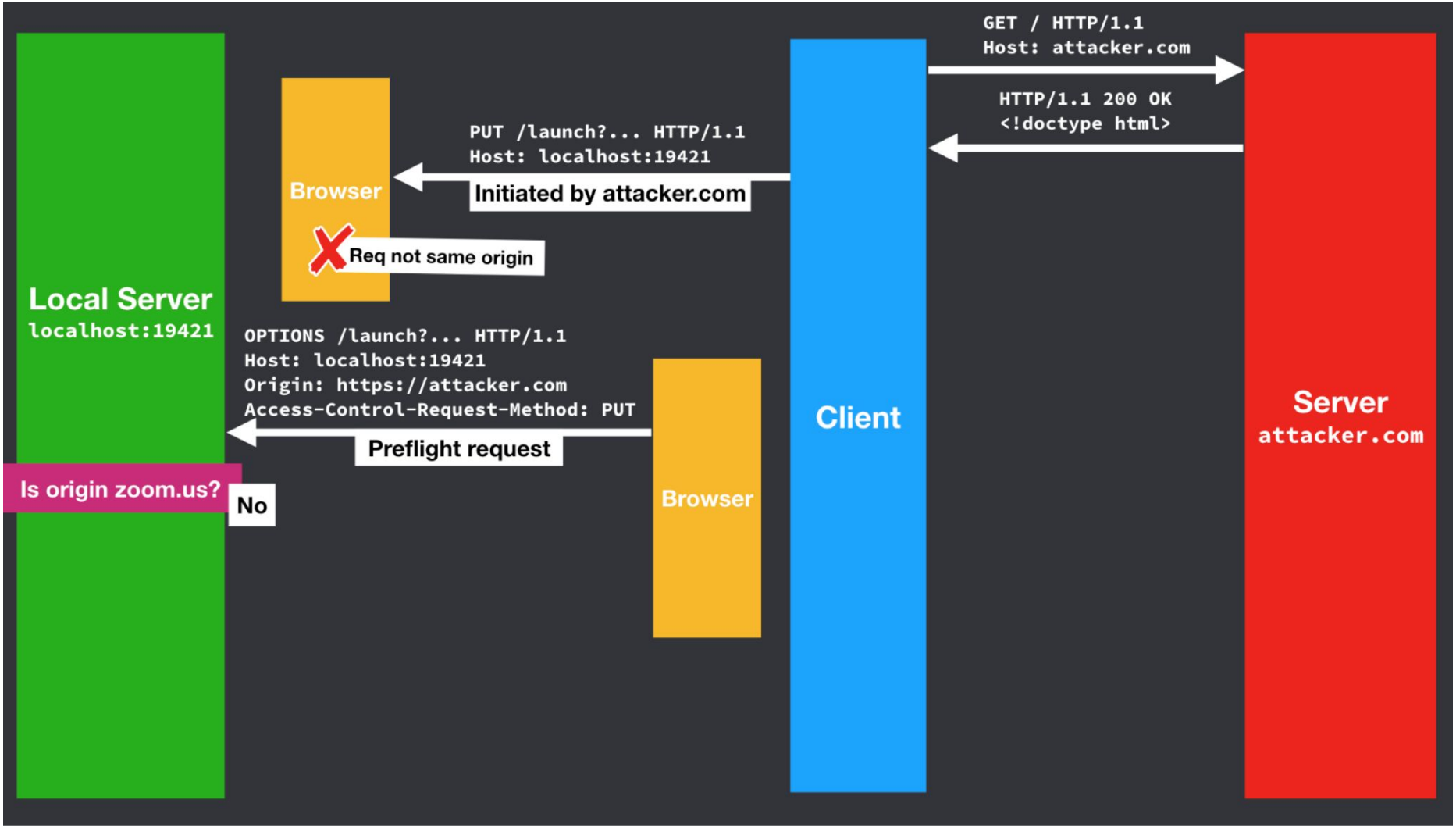

GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by attacker.com
Browser
Local Server
localhost:19421
Client
Server
attacker.com

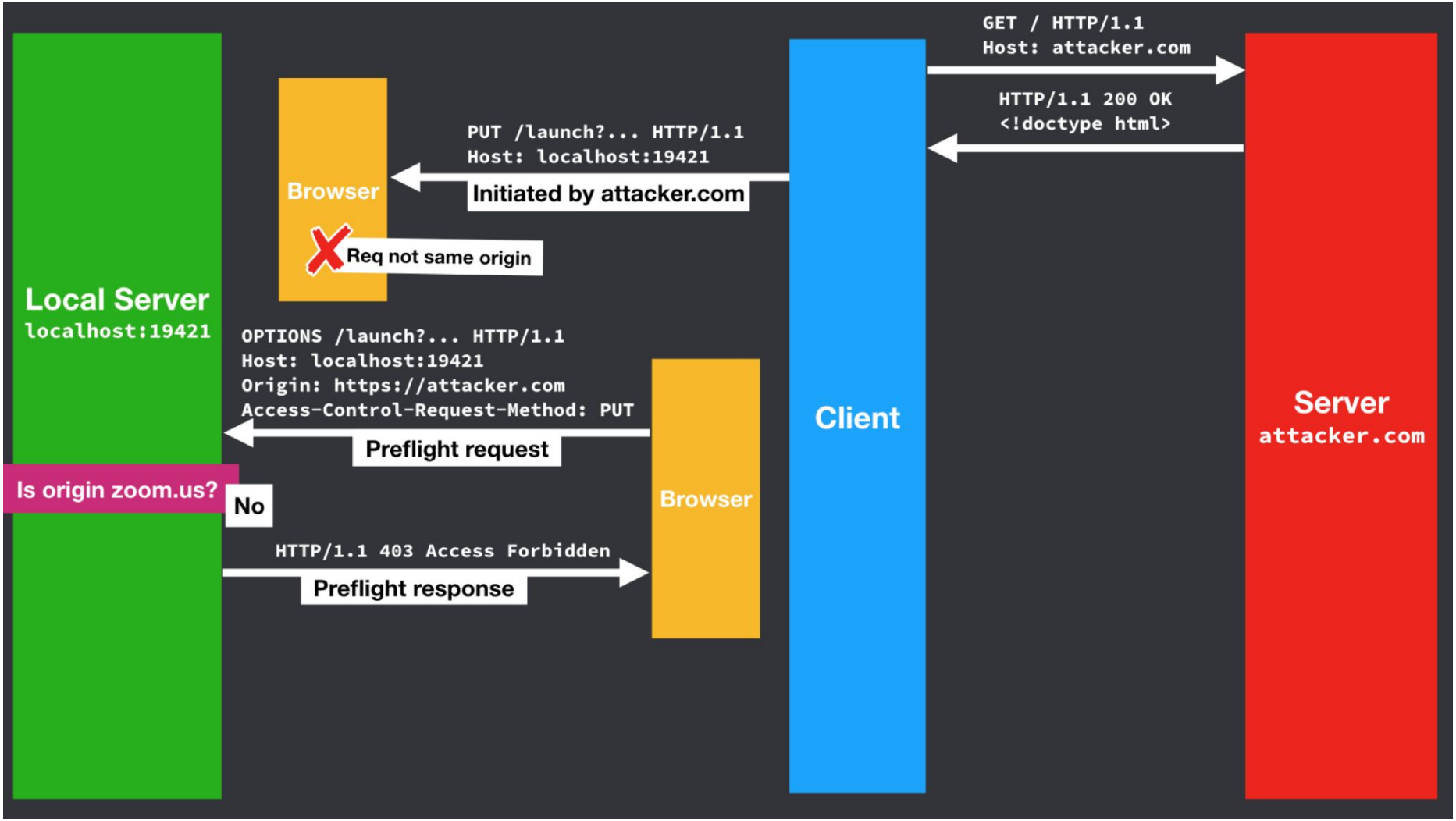

GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by attacker.com
Browser
Req not same origin
Local Server
localhost:19421
Client
Server
attacker.com

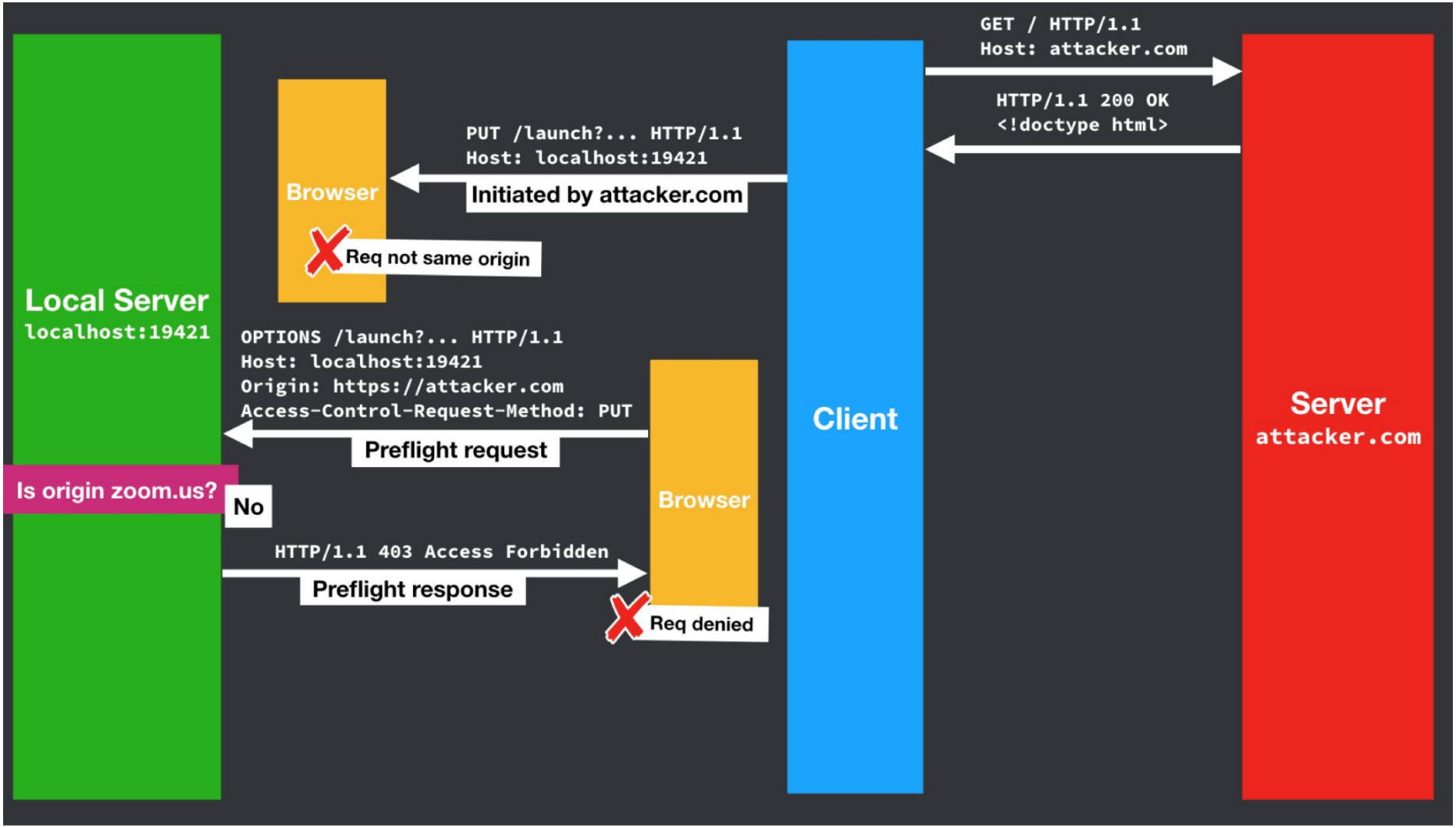

GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by attacker.com
Browser
Req not same origin
Local Server
localhost:19421
Browser
Client
Server
attacker.com


GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by attacker.com
Browser
Req not same origin
Local Server
localhost:19421
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://attacker.com
Access-Control-Request-Method: PUT
Preflight request
Browser
Client
Server
attacker.com


GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
Server
attacker.com
Client
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by attacker.com
Browser
Req not same origin
Local Server
localhost:19421
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://attacker.com
Access-Control-Request-Method: PUT
Preflight request
Browser
Is origin zoom.us?


GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
Server
attacker.com
Client
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by attacker.com
Browser
Req not same origin
Local Server
localhost:19421
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://attacker.com
Access-Control-Request-Method: PUT
Preflight request
Browser
Is origin zoom.us?
No


GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
Server
attacker.com
Client
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by attacker.com
Browser
Req not same origin
Local Server
localhost:19421
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://attacker.com
Access-Control-Request-Method: PUT
Preflight request
Is origin zoom.us?
No
Browser
HTTP/1.1 403 Access Forbidden
Preflight response


GET / HTTP/1.1
Host: attacker.com
HTTP/1.1 200 OK
<!doctype html>
Server
attacker.com
Client
PUT /launch?... HTTP/1.1
Host: localhost:19421
Initiated by attacker.com
Browser
Req not same origin
Local Server
localhost:19421
OPTIONS /launch?... HTTP/1.1
Host: localhost:19421
Origin: https://attacker.com
Access-Control-Request-Method: PUT
Preflight request
Is origin zoom.us?
No
Browser
HTTP/1.1 403 Access Forbidden
Preflight response
Req denied

Who can still launch the app from the local server?

- Preflight requests seems to allow the local server to distinguish requests from zoom.us and those from random sites
- However, other native apps running on the same device can still fool the local server
  - The browser enforces that sites can't tamper with the `Origin` header, but a native app (e.g., a Node.js or Python script) can make a request and set the `Origin` header to https://zoom.us
  - Note: if a script is already running on your local machine, you have a whole lot of other (potential) problems

One more thing...

- Every site on the web can send requests to our local HTTP server!
  - Works against the server that required "preflighted" requests as well as the server which just checked the `Origin` header
- “localhost” seems simple but is actually highly magical -- what if you could redefine what “localhost” meant?
  - Next time: DNS rebinding attacks