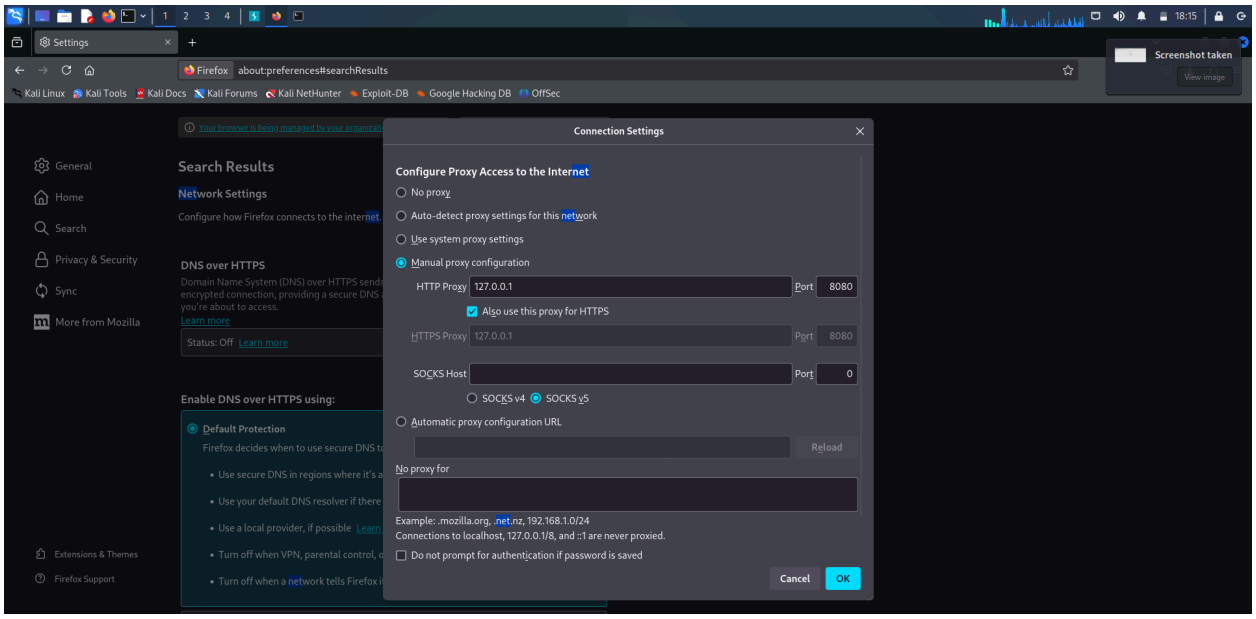
Ritesh Jha | SY-IT(B3) | 16010423076 | Cyber Honors

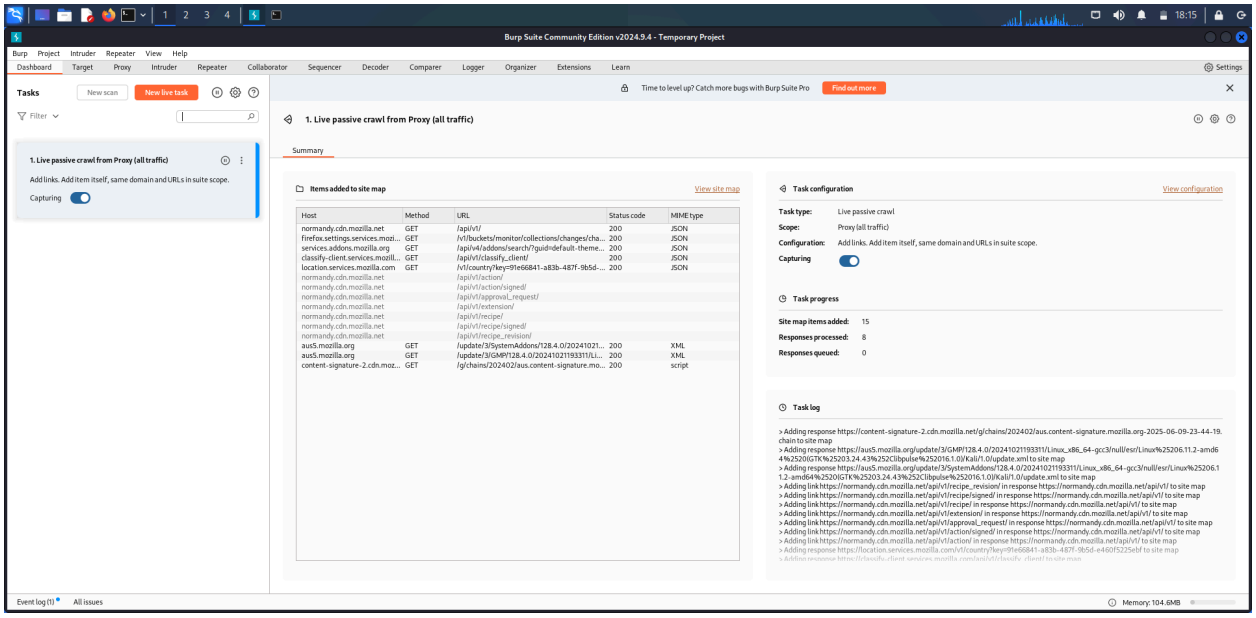
Compensation-1 | Date : 24 - April - 2025

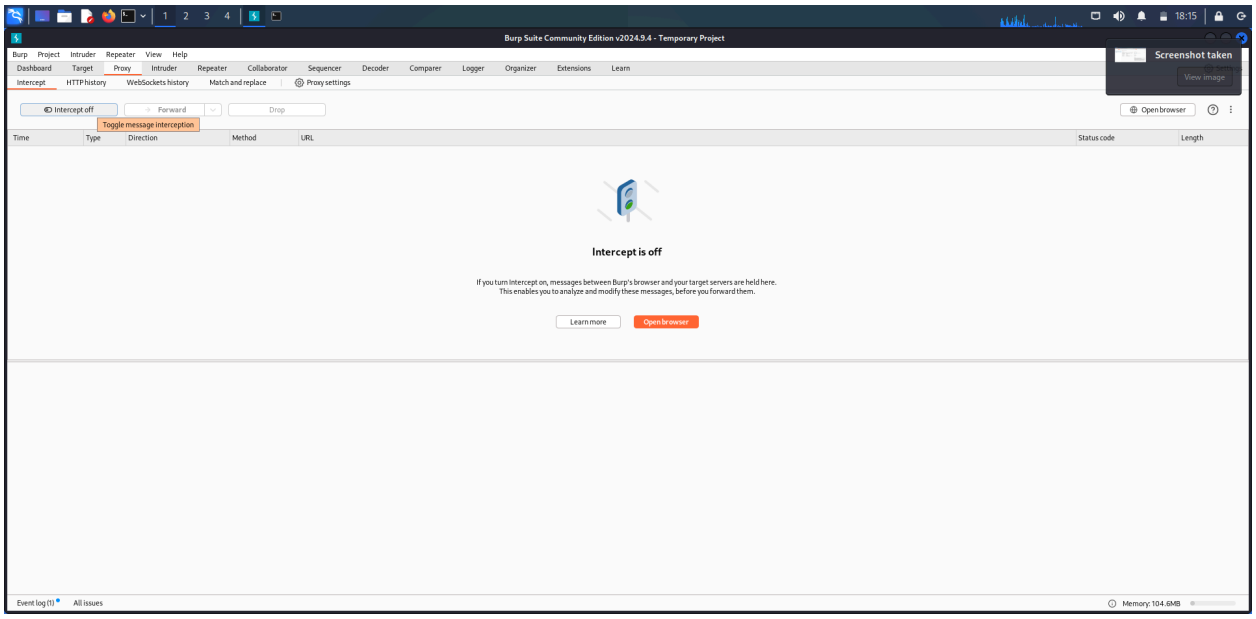
Web Cache Detection

1. Initial Setup

I started by opening Burp Suite and selecting a temporary project to keep things simple. No need for saving configs here, just wanted to test stuff quickly. Then I went to my Chrome browser settings and manually set the proxy to 127.0.0.1:8080, which is Burp's default. This allowed all browser traffic to pass through Burp so I could intercept and inspect what was going on between my browser and the server.


Settings
Firefox about:preferences#searchResults
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
Screenshot taken
General
Home
Search
Privacy & Security
Sync
More from Mozilla
Search Results
Network Settings
Configure how Firefox connects to the internet.
DNS over HTTPS
Enable DNS over HTTPS using:
Default Protection
Extensions & Themes
Firefox Support
Connection Settings
Configure Proxy Access to the Internet
No proxy
Auto-detect proxy settings for this network
Use system proxy settings
Manual proxy configuration
HTTP Proxy 127.0.0.1 Port 8080
Also use this proxy for HTTPS
HTTPS Proxy 127.0.0.1 Port 8080
SOCKS Host Port 0
SOCKS v4 SOCKS v5
Automatic proxy configuration URL
Reload
No proxy for
Example: .mozilla.org, .net.nz, 192.168.1.0/24
Connections to localhost, 127.0.0.1/8, and ::1 are never proxied.
Do not prompt for authentication if password is saved
Cancel
OK

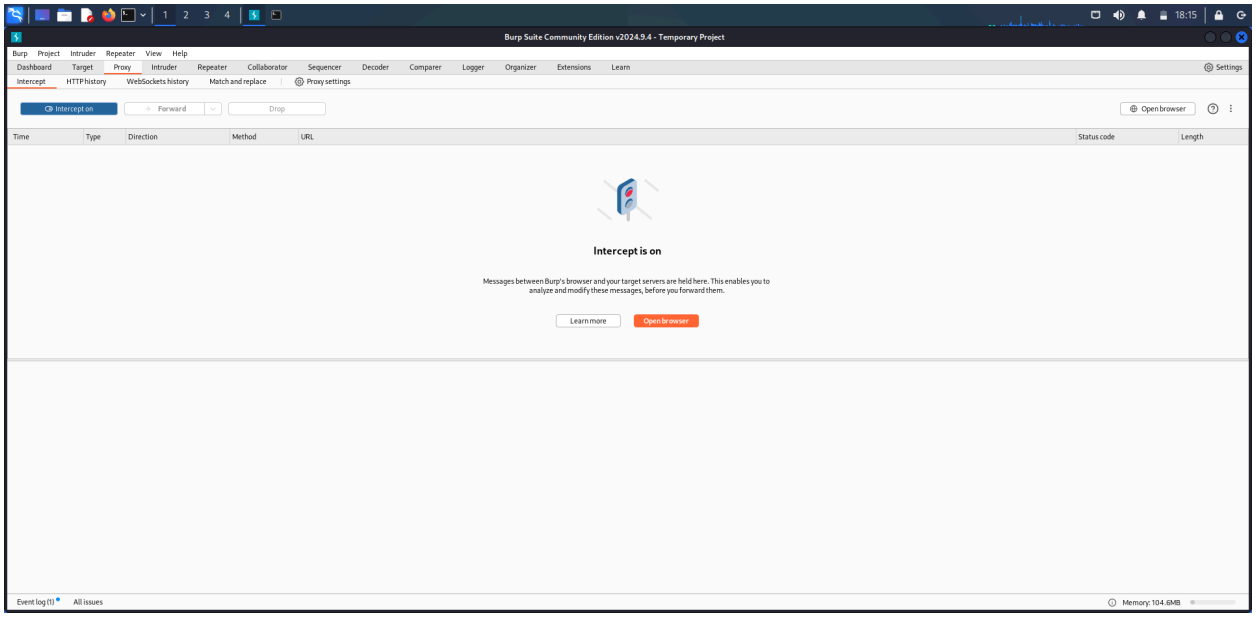

Burp Suite Community Edition v2024.9.4 - Temporary Project
Tasks
New scan
New live task
1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, same domain and URLs in suite scope.
Capturing
Time to level up? Catch more bugs with Burp Suite Pro
Find out more
1. Live passive crawl from Proxy (all traffic)
Summary
Items added to site map
View site map
Task configuration
View configuration
Task type: Live passive crawl
Scope: Proxy (all traffic)
Configuration: Add links. Add item itself, same domain and URLs in suite scope.
Capturing
Task progress
Site map items added: 15
Responses processed: 8
Responses queued: 0
Task log
Event log (1)
All issues
Memory: 104.6MB

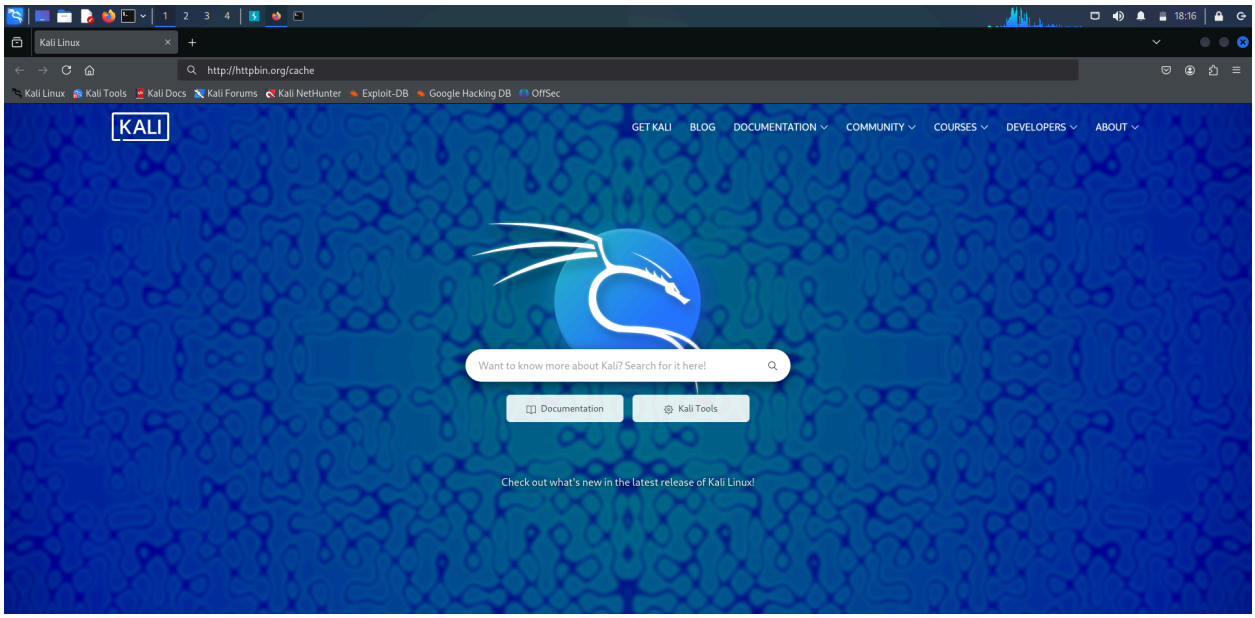


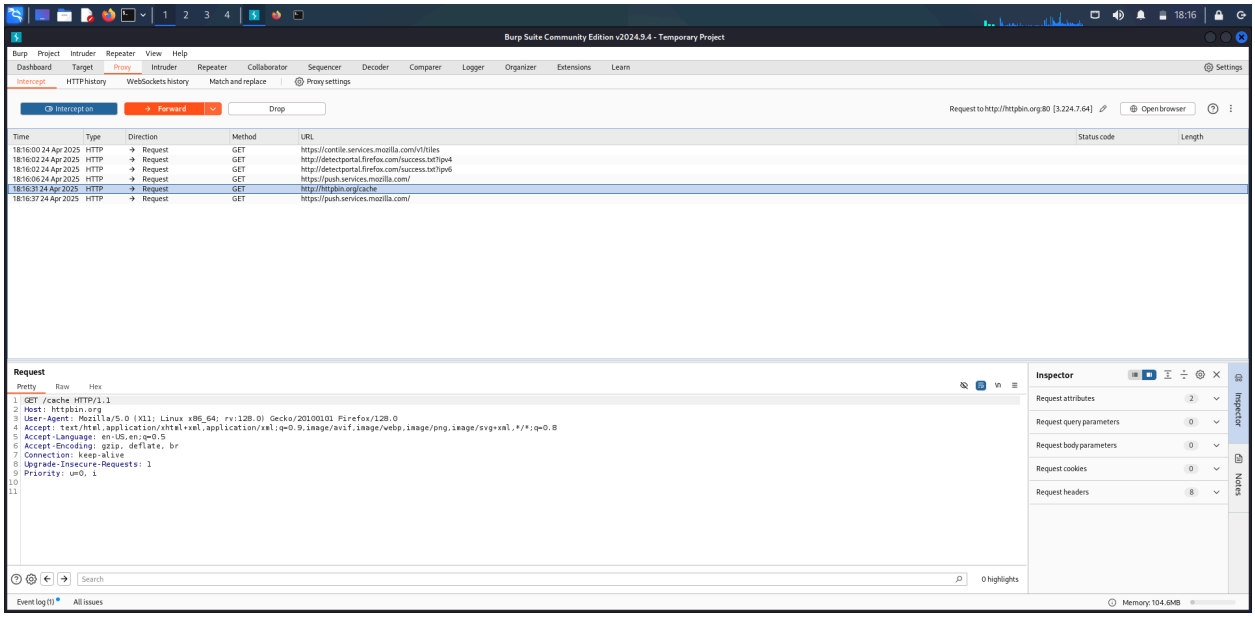
---

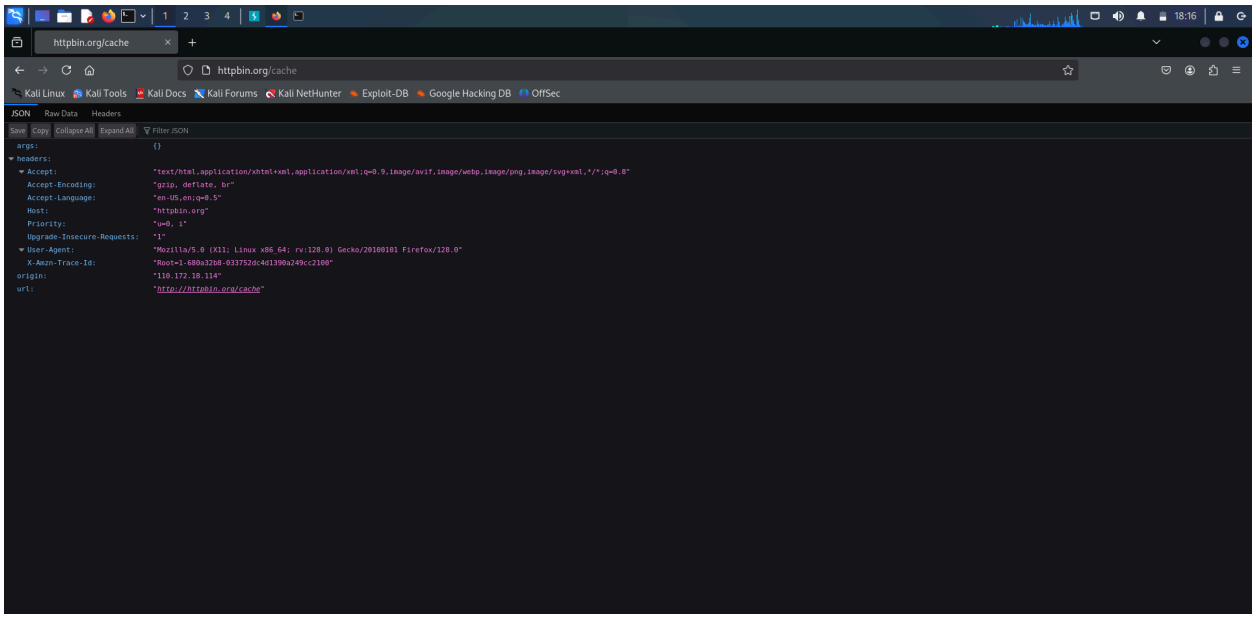
2. Intercepting the First Request

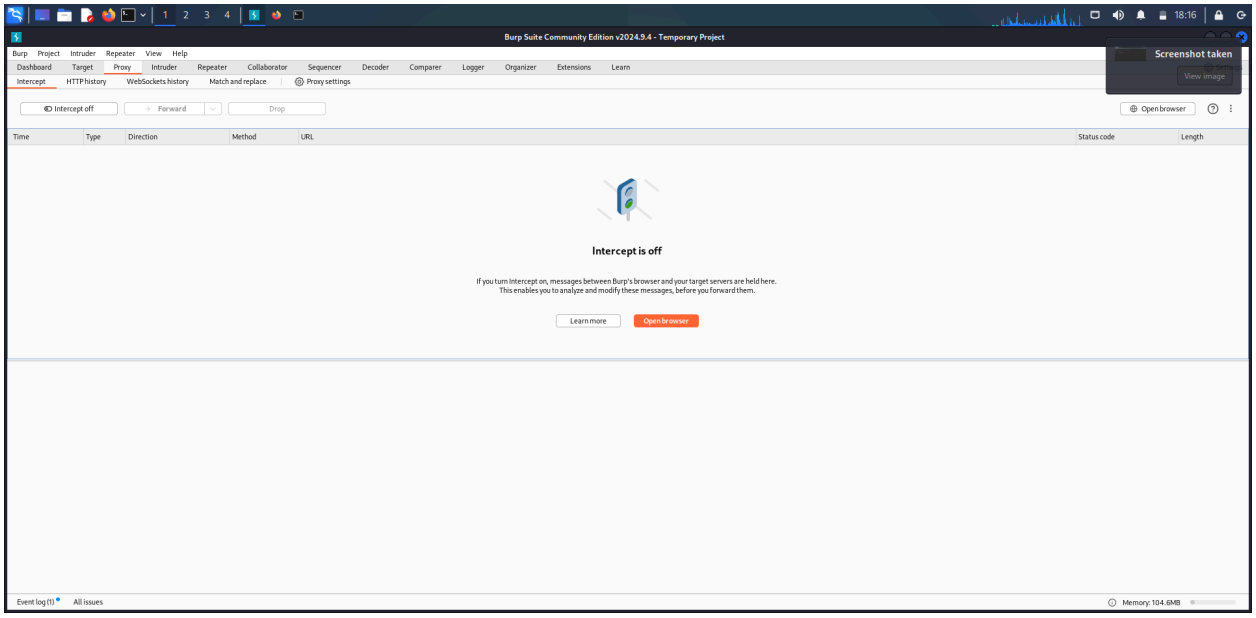
With Burp ready, I turned *Intercept ON* and visited http://httpbin.org/cache in the browser. Burp immediately caught the GET request, showing that traffic was being routed correctly. I forwarded the request so it could reach the server, then turned intercept OFF, from here on, I just wanted to observe how the browser and server were talking, especially when it came to caching behavior.




Kali Linux
http://httpbin.org/cache
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
KALI
GET KALI BLOG DOCUMENTATION COMMUNITY COURSES DEVELOPERS ABOUT
Want to know more about Kali? Search for it here!
Documentation
Kali Tools
Check out what's new in the latest release of Kali Linux!


Burp Suite Community Edition v2024.9.4 - Temporary Project
Request
Inspector
Request attributes
Request query parameters
Request body parameters
Request cookies
Request headers

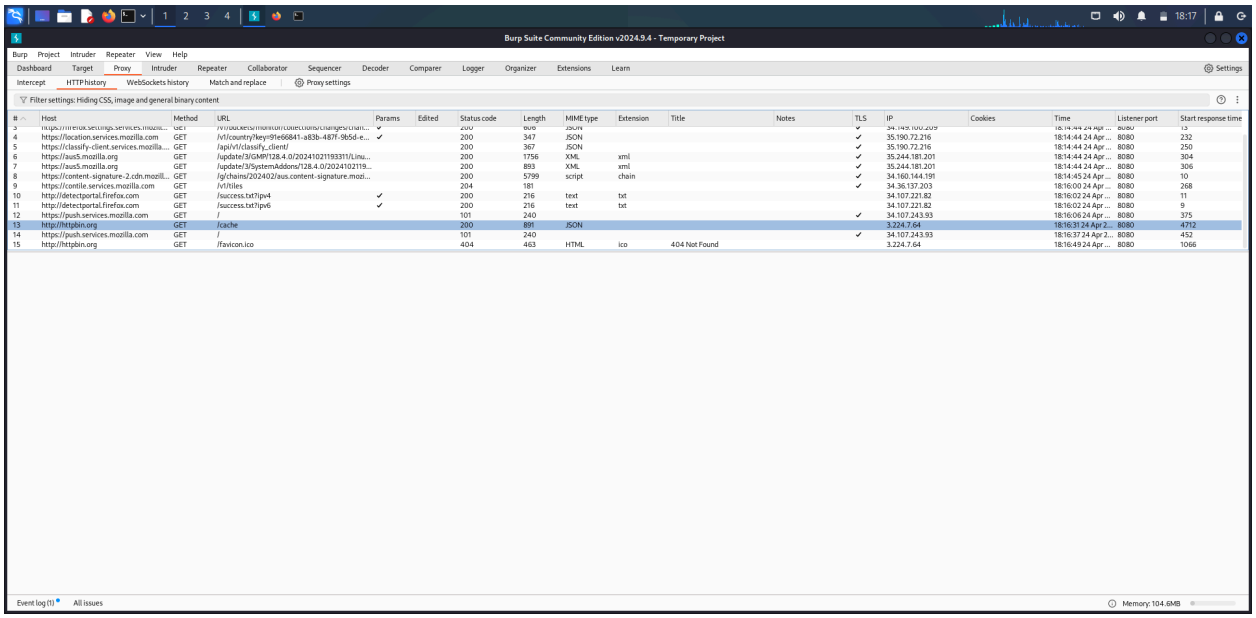


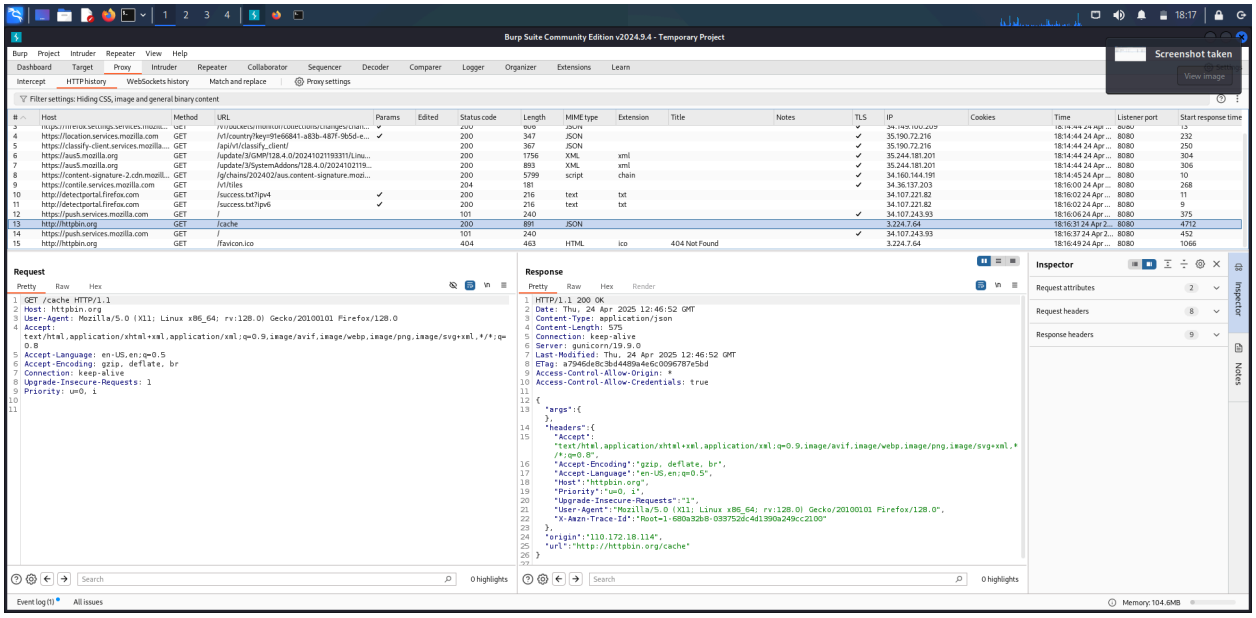


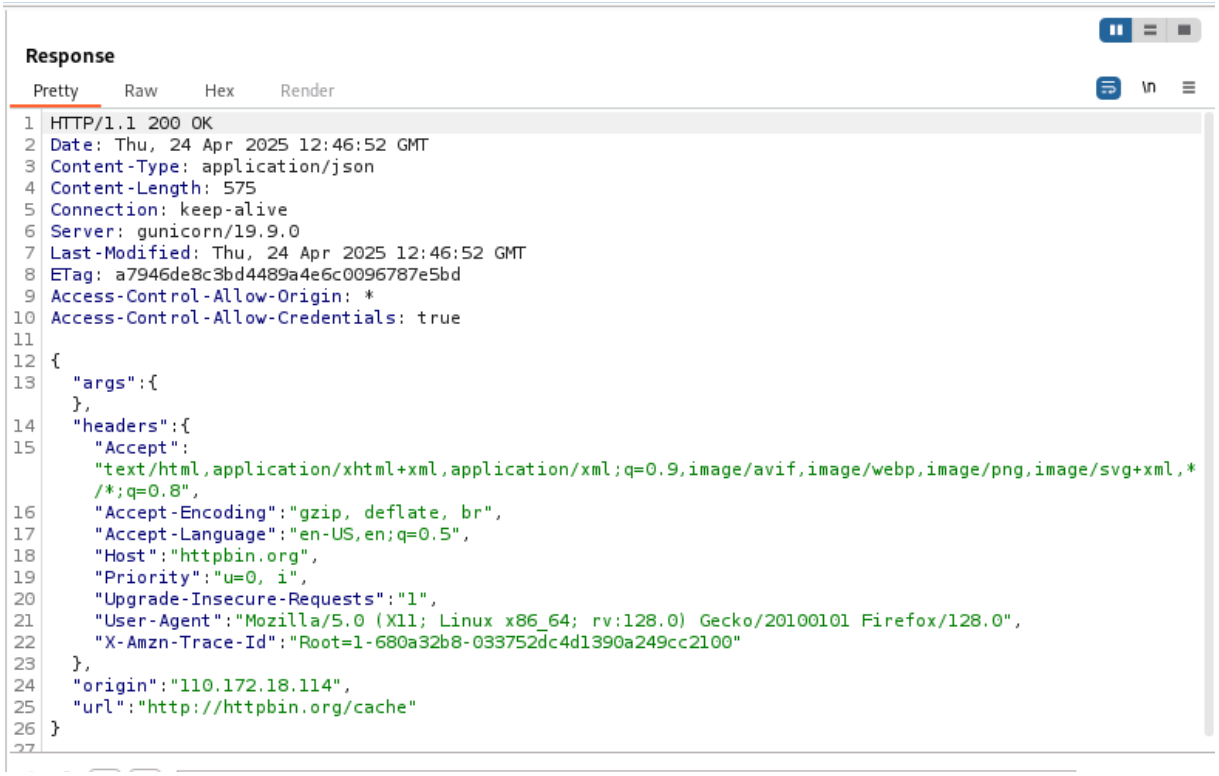
---

3. Checking Response Headers

Next, I went into the HTTP history tab inside Burp to check the response we got back from the server. I looked for caching headers like Cache-Control, which had public, max-age, meaning the server was okay with the content being cached publicly for an hour. Also saw an ETag, which is used to track content changes, and a Last-Modified timestamp. These were good indicators that the server supported caching and was set up to optimize repeated access.


Burp Suite Community Edition v2024.9.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Filter settings: Hiding CSS, image and general binary content
Event log (1) All issues
Memory: 104.6MB

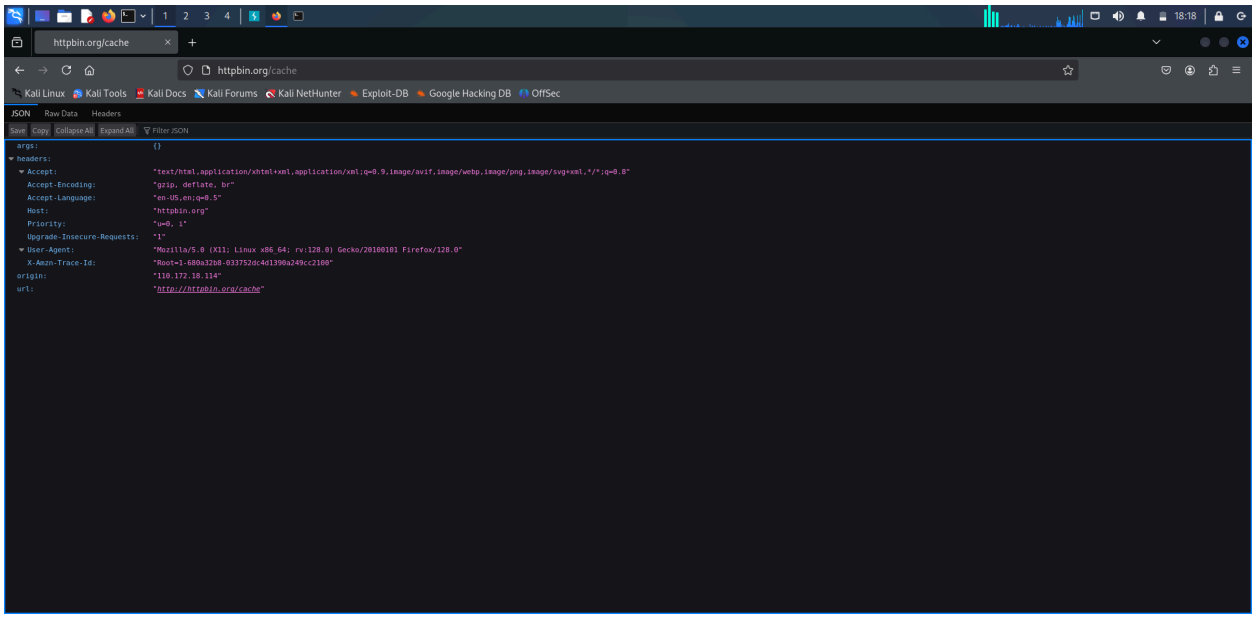

Burp Suite Community Edition v2024.9.4 - Temporary Project
Screenshot taken
View image
Request
Response
Inspector
Request attributes
Request headers
Response headers
Event log (1) All issues
Memory: 104.6MB

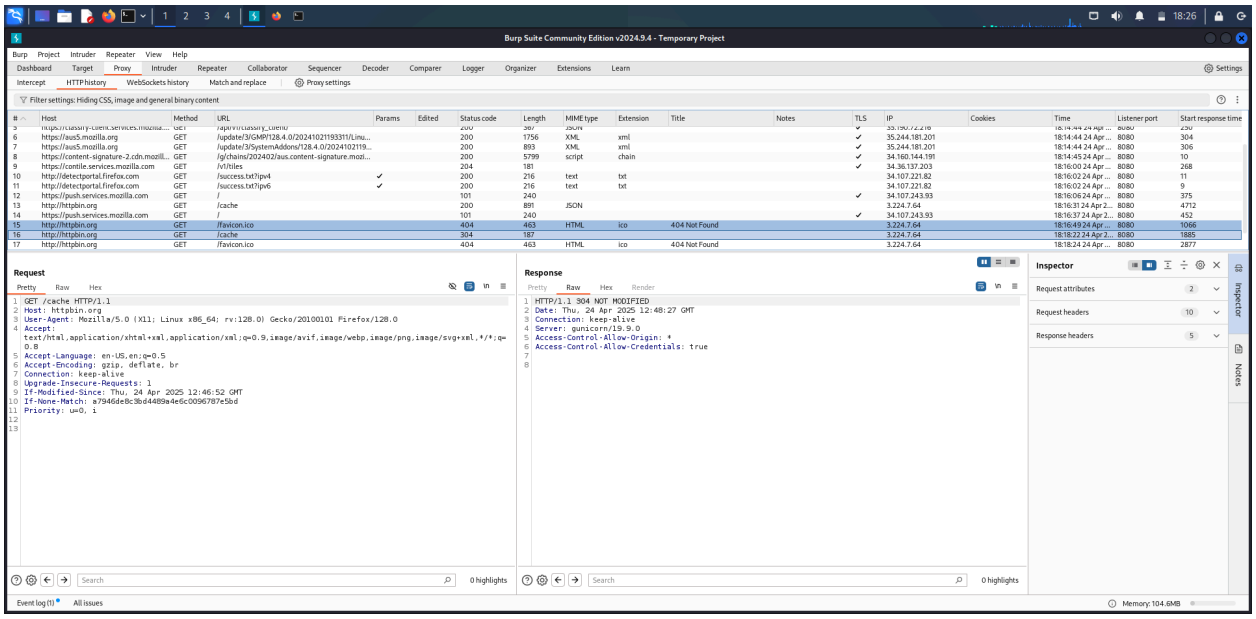


---

4. Testing the Caching

To see if caching actually worked, I refreshed the page in my browser and watched what Burp showed this time. The second request popped up, and I noticed the response came quicker. More importantly, the server returned a 304 Not Modified status, that's a classic sign that the browser asked if the resource had changed, and the server said nope, use your local cache. So this confirmed the caching behavior was not just in headers but actually functioning.


httpbin.org/cache
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
JSON
Raw Data
Headers
args: {}
headers:
Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8"
Accept-Encoding: "gzip, deflate, br"
Accept-Language: "en-US,en;q=0.5"
Host: "httpbin.org"
Priority: "u=0, i"
Upgrade-Insecure-Requests: "1"
User-Agent: "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0"
X-Amzn-Trace-Id: "Root=1-680a32b8-033752dc4d1390a249cc2100"
origin: "110.172.18.114"
url: "http://httpbin.org/cache"


Burp Suite Community Edition v2024.9.4 - Temporary Project
Request
GET /cache HTTP/1.1
Host: httpbin.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Thu, 24 Apr 2025 12:46:52 GMT
If-None-Match: a7946de8c3bd44859a4e6c0096787e5bd
Priority: u=0, i
Response
HTTP/1.1 304 NOT MODIFIED
Date: Thu, 24 Apr 2025 12:48:27 GMT
Connection: keep-alive
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Inspector


Response
Pretty
Raw
Hex
Render
HTTP/1.1 304 NOT MODIFIED
Date: Thu, 24 Apr 2025 12:48:27 GMT
Connection: keep-alive
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

Conclusion

Through this demo, I successfully detected and confirmed the presence of **server-side web caching mechanisms** using Burp Suite. By analyzing HTTP response headers like Cache-Control, ETag, and Last-Modified, I was able to determine that the server explicitly instructs clients to cache content for a specified period. The most definitive indicator came during the second request, where the server responded with **304 Not Modified**, confirming that the browser relied on its local cache rather than downloading the same content again. This behavior demonstrates how caching improves efficiency by reducing load times, saving bandwidth, and optimizing overall network usage. This kind of detection is not just academic, it's crucial for understanding performance optimization, security implications (like cache poisoning), and behavior under real-world usage. Even a basic setup like this reveals how servers control caching logic and how clients respond to it.