UNIVERSITY OF TRENTO

Hidden Web Caches Discovery

Matteo Golinelli
matteo.golinelli@unitn.it
University of Trento
Trento, Italy

Bruno Crispo
bruno.crispo@unitn.it
University of Trento
Trento, Italy

Published in the "*Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*" (RAID '24)
https://doi.org/10.1145/3678890.3678931

Background: Web Caches

- Cache **public** and **static** content
- Increase **scalability**, **availability**, and **performance**
- Frequently implemented by Content Delivery Networks (**CDNs**)
  - **Geographically distributed**, physically closer to the end client

62%

of Top 10k is behind a **CDN**[1]

+ many other **stand-alone caches technologies** (e.g., Squid, Varnish, NGINX)

[1] BuiltWith. BuiltWith Technology Lookup. https://trends.builtwith.com/CDN/Content-Delivery-Network

Background: Cache Status Headers (CSH)

Used by caches to communicate whether a response is coming from the cache or from the origin server

- Are **not standardized**

CDN / Cache	Header Name(s)	*Hit* value(s)	*Miss* value(s)
Akamai	`server-timing, X-Cache, X-Cache-Remote`	`desc=HIT, TCP_HIT`	`desc=MISS, TCP_MISS`
CDN77	`X-Cache`	`HIT`	`MISS`
Cloudflare	`cf-cache-status`	`HIT`	`MISS`
CloudFront	`x-cache`	`Hit from cloudfront`	`Miss from cloudfront`
Fastly	`X-Cache`	`HIT`	`MISS`

Examples of cache status headers of popular cache technologies[2]

[2] Mirheidari, S. A., Golinelli, M., Onarlioglu, K., Kirda, E., & Crispo, B. (2022). **Web Cache Deception Escalates!**. In *31st USENIX Security Symposium (USENIX Security 22)*

Background: Cache Key

Cache key: **unique identifier** for an object in a cache

- Based on some request components (called **keyed**)
  - Domain name and path: `example.com/path/to/index.html`
  - Query string: `?id=1&order=reverse`
  - Headers: `Accept-Language: en-US,en`

↪ Cache **HIT**: the request has the **same cache key** of a previously cached request

↪ Cache **MISS**: the cache key is **different** from all the objects already in the cache

[3] Amazon AWS. **Understanding the cache key**. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/understanding-the-cache-key.html

Background: Cache Busting

Modify keyed elements of HTTP requests to change the cache key

- Forcefully receive a **fresh copy** of the response, instead of a cached one

Example: add random parameter to the **query string**

`https://site.com/` ➪ `https://site.com/?ran=dom`

Cache Busting Techniques

We tested different cache busting techniques on the Tranco Top 10k to identify the **most effective** ones

Cache-busting technique	Cache busted
Query string	2112 (60.4%)
`Origin` header	817 (23.4%)
`User-Agent` header	78 (2.2%)
`X-Forwarded-Host` header	327 (9.4%)
`X-Forwarded-Scheme` header	329 (9.4%)
`X-Method-Override` header	338 (9.7%)
Headers in `Vary` header	616 (17.6%)
All techniques combined	2946 (84.3%)

We can **cache-bust** requests on **84.3%** of sites

The Issue: Detect Cached Responses

Detecting cached responses is crucial to create detection methodologies for web cache vulnerabilities (Web Cache Deception, Cache Poisoning, …)

State-of-the-art methodologies to detect cached responses are based on **lookups of cache status headers**:

- **Not effective** when cache status headers are missing, wrong, or custom

Goal: Develop a methodology that detects caching **without relying** on cache status headers

Methodology: Timing Measurements

We send:

- **n** pairs of requests where both requests have random cache busters (**Randomized** group)
- **n** pairs of requests where only one request has a random cache buster (**Fixed** group)

↪ We measure the relative time difference between receiving the two responses

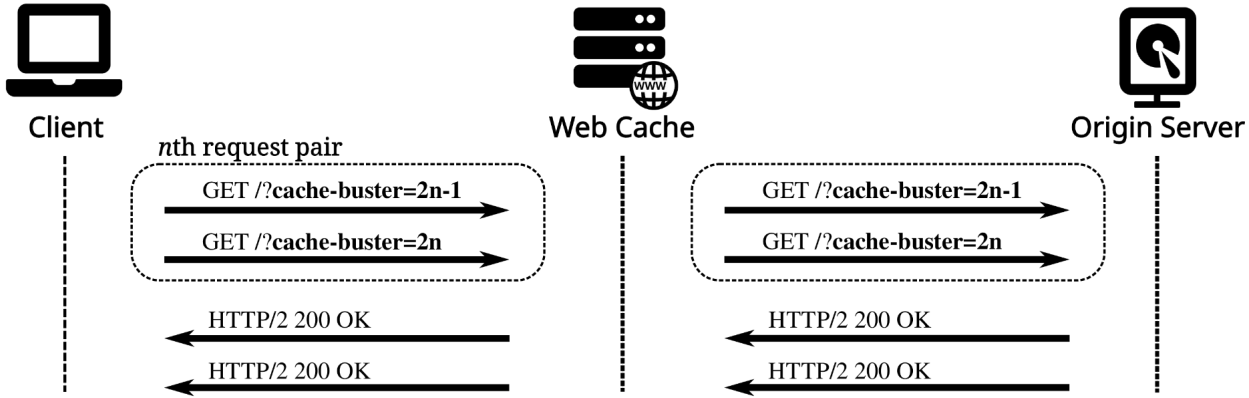
Request pairs are sent in a single packet using **HTTP/2 multiplexing**

- Based on Van Goethem et al. "Timeless Timing Attacks"[4]

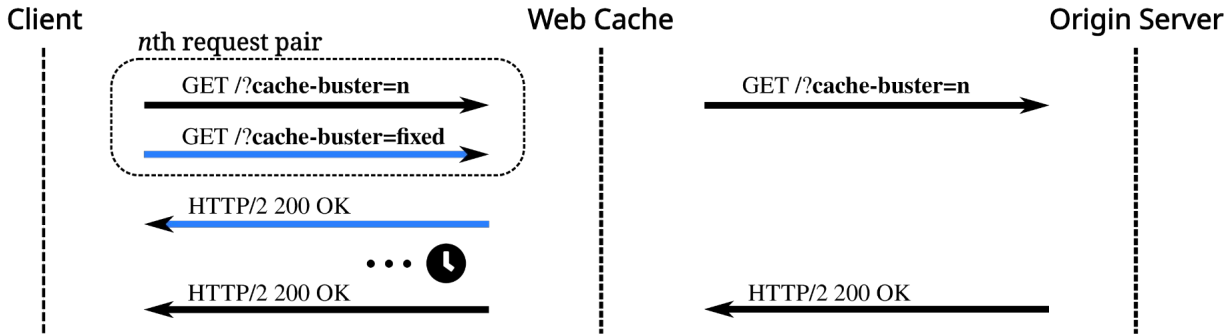
[4] Goethem, T. V., Pöpper, C., Joosen, W., & Vanhoef, M. (2020). **Timeless Timing Attacks: Exploiting Concurrency to Leak Secrets over Remote Connections**. In *29th USENIX Security Symposium*

Methodology: Overview

① Randomized group



② Fixed group



Methodology: Timing Analysis

We analyze the collected **timing differences** using a statistical test to determine whether their difference is **statistically significant** or not

- If it is: the non-cache-busted request is cached

Time measurements with cached responses

Group	Time diff. (ms)	Cache Status 1	Cache Status 2
Randomized	-60.09	MISS	MISS
	62.42	MISS	MISS
	-58.35	MISS	MISS
	67.32	MISS	MISS
	-77.45	MISS	MISS
Fixed	-600.95	MISS	HIT
	-504.63	MISS	HIT
	-591.15	MISS	HIT
	-516.49	MISS	HIT
	-536.35	MISS	HIT

Time measurements with no cached responses

Group	Time diff. (ms)	Cache Status 1	Cache Status 2
Randomized	34.37	MISS	MISS
	97.29	MISS	MISS
	-486.03	MISS	MISS
	132.2	MISS	MISS
	-325.18	MISS	MISS
Fixed	-169.52	MISS	MISS
	12.2	MISS	MISS
	-409.99	MISS	MISS
	-31.29	MISS	MISS
	217.21	MISS	MISS

Preliminary Experiment

Goal: test our methodology on sites that report caches status headers and **compare** it **to the SOTA methodology** in the Tranco Top 10k

	Number of sites	Percentage*
Analysed	1.946	19.5%
Correct classification	1.743	89.6%
Wrong classification	203	10.4%

↪ Our methodology has an **accuracy** of **89.6%**

Methodology: Accuracy

We selected **100** sites where our methodology had a **different classification** compared to the state-of-the-art methodology and **manually verified** them

- **82**/100 were due to **wrong cache status headers** (our methodology was right)

↪ The **accuracy** of our methodology is likely far higher than **89.6%**

Large-Scale Experiment

Goal: discover **hidden web caches** (caches that do not use cache status headers) in the Tranco Top 50k

	Number of sites	Percentage*
Reachable	39.159	78.3%
Tested	28.243	56.5%
No cache status headers	17.700	62.7%
Cache	1.627	5.8%
No cache	16.073	56.9%

% over tested

Vulnerabilities Detection

We use our methodology to detect how many sites with hidden caches **cache dynamic content**

- Caching dynamic content is not always an indication of a vulnerability, but in certain cases might lead to the **leakage of sensitive information**

  1.020/1.627 sites with hidden caches **cache dynamic content**

We manually investigated **35** cases:

↪ We identified **5** sites vulnerable to Web Cache Deception

Conclusions

We developed an **accurate methodology** that **detects caching** without relying on cache status headers

- Useful to create detection methodologies for web cache vulnerabilities
- Open source on https://github.com/golim/hidden-web-caches-discovery

We detected **hidden web caches on 5.8% of sites** in the Tranco Top 50k that support HTTP/2

- The same methodology can be implemented using HTTP/3

We used our methodology to find **well-hidden vulnerabilities** that would otherwise be impossible to spot and detect

"We acknowledge the support of the MUR PNRR project PE SERICS – SecCO (PE00000014) CUP D33C22001300002 funded by the European Union under NextGenerationEU. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the granting authority can be held responsible for them."