Web Cache Deception Escalates!

Seyed Ali Mirheidari
University of Trento, Italy & Splunk Inc.
mirheidari.ali@gmail.com

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

Kaan Onarlioglu
Akamai Technologies
kaan@iseclab.org

Engin Kirda
Northeastern University
ek@ccs.neu.edu

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

Background: Web Caches

62%

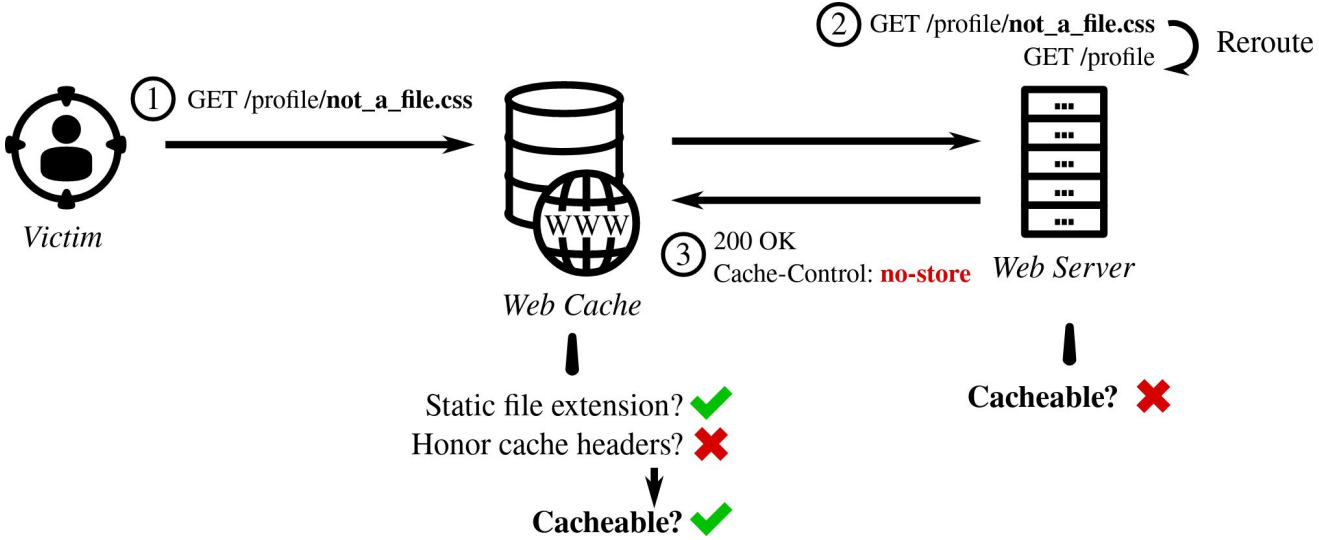
of the Alexa Top 10k is behind a **CDN**

- Cache **public content**
- Frequently implemented by Content Delivery Networks (**CDNs**)
  - Increase **scalability**, **availability**, and **performance**
  - Offer options to **disregard cache-control headers** to increase performance

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

Background: Web Cache Deception

- Exploits **request processing discrepancies** between the origin server and a public web cache
- The cache is tricked into erroneously storing **sensitive content**



The Issue: State Of The Art Limitations

The **state of the art** detection methodology is **marker-based** [1]

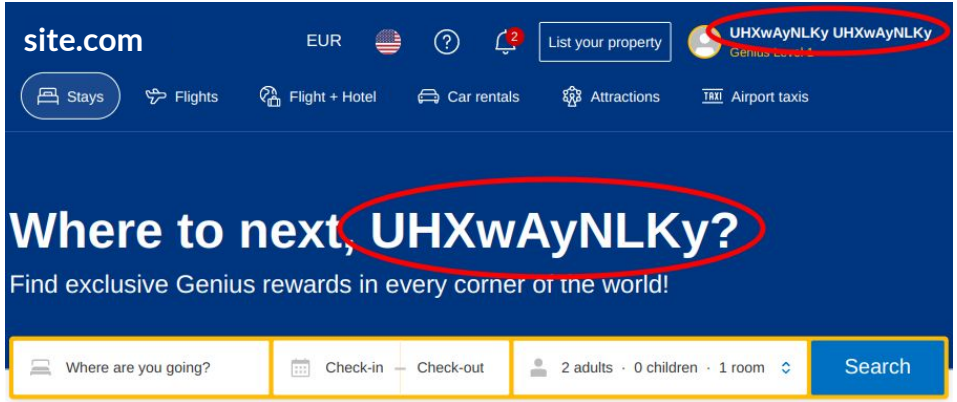
Issues:

- Assumes that **the victim is authenticated**
  - **More personal data** that can be leaked

[1] S. A. Mirheidari, S. Arshad, K. Onarlioglu, B. Crispo, E. Kirda, and W. Robertson. *Cached and Confused: Web Cache Deception in the Wild*. In USENIX Security Symposium, 2020

The Issue: State Of The Art Limitations

- **Scalability**: costly **manual** setup process
  - Account creation
  - Populate the target site with a unique marker string
- **Coverage**: cannot test
  - Pages that do not reflect the marker
  - Sites that do not need authentication



Motivation & Goals: Web Cache Deception Escalates!

New methodology that:

- **Overcomes** the state of the art **limitations**
- Targets **unauthenticated victims** and **pages**

Uses:

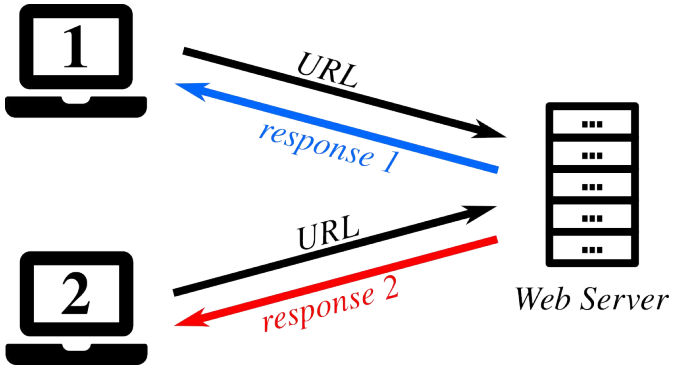
- **Content identicality** checks
- **Cache headers** heuristics

Methodology: Content Identicality Checks

Content identicality checks algorithm

```
response1 <- HTTP.get(URL)
response2 <- HTTP.get(URL)

if response1 ≠ response2 then
    return "Dynamic Content"
else:
    return "Static Content"
```



Methodology: Cache Headers Heuristics

Cache status headers used by popular web caches

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`
Google Cloud	`cdn_cache_status`	`hit`	`miss`
KeyCDN	`X-Cache`	`HIT`	`MISS`
Azure	`X-cache`	`TCP_HIT, TCP_REMOTE_HIT`	`TCP_MISS`
Apache, ATS	`X-Cache`	`HIT`	`MISS`
NGINX	`X-Proxy-Cache`	`HIT`	`MISS`
Rack Cache	`X-Rack-Cache`	`hit`	`miss`
Squid	`X-Cache`	`HIT from *`	`MISS from *`
Varnish	`X-Cache`	`HIT`	`MISS`
Unknown	`x-cache-info`	`cached`	`caching`

Methodology: Web Cache Deception Escalates!

Step 1. Does the tested URL return **dynamic content**?

Step 2. Does the **crafted attack URL** still respond with dynamic content?

Step 3. Is the **origin response** to the attack URL **cached**?

Experiment: Comparative Evaluation

Analysis of **404 websites** using:

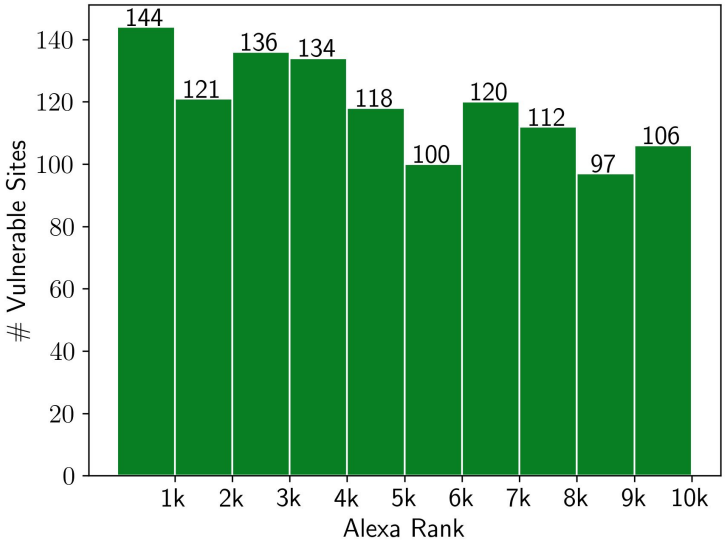
- **CC** state of the art methodology **with authentication**
- **DE** $_{auth}$ our new methodology **with authentication**
- **DE** our new methodology **without authentication**

	CC		DE_{auth}		DE	
Total Detections	21	(5.20%)	134	(33.17%)	129	(31.93%)
True Positives	18	(4.46%)	115	(28.47%)	104	(25.74%)
False Positives	3	(0.74%)	19	(4.70%)	25	(6.19%)
Unique True Positives	2	(0.50%)	13	(3.22%)	5	(1.24%)

Experiment: Large-Scale Analysis

Analysis of the **Alexa Top 10k** using DE

Vulnerable Sites	1188	(11,88%)
CSRF Token	436	(36.70%)
CSP Nonce	13	(1.09%)
OAuth State	34	(2.86%)
Session ID	63	(5.30%)



Results: Security Impact

- **Unauthenticated** and **authenticated victims** are impacted

- Stealing of **security tokens**
  - Enables **bypassing security protections**
- Leakage of **personal information**

Results: Case Studies

4200$

in **bug bounties**

Attack scenario chains:

- **Login CSRF** by stealing the **OAuth state**
- **CSRF** by stealing **CSRF tokens**
- **Session Hijacking** by stealing **Session IDs**
- **Cache Poisoning** leading to **stored XSS**

- **Supply chain issue**: one vulnerable cloud service provider impacted **456 sites**

Conclusions

- **Unauthenticated victims** are impacted as well
  - **Publicly accessible pages** can contain **security-critical secrets**

- **WCD** is still an **open problem** and we don't have a clear and reliable solution

Conclusions

- WCD is a **system problem** and no single component is necessarily faulty in isolation
  - Website operators must **consider the full integration**
    - Caches are not plug & play technologies
  - Simple caching rules can have **far-reaching effects**

Web Cache Deception Escalates!

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

- **DE**: **new methodology** for **WCD vulnerabilities detection**
  - **Overcomes** the **limitations** of the state of the art
  - **Outperforms** the state of the art

- Different **target**: **unauthenticated victims** and **pages**

- **Prevalence** of WCD vulnerabilities
- **Successful exploitation** of vulnerabilities