

The war on adblocking

CNAME Cloaking Exposed



April 29, 2021

Web application security

Nathanial Mendes (nmm6145)

# Introduction

## What is Ad blocking?

A red sign with a white hand in the middle

Description automatically generated with low confidenceAdvertisement blocking is a means to blocking or removing the visual content from loading. Ad blocking is typically done by browser extensions but can also be done with software and DNS query modifications. I use Ad blocking for a multitude of reasons including protecting privacy, limiting exposure to malicious advertisements, and better web usability. The advertisements blocked typically result in lost revenue from advertising providers, creating a systemic resistance incentivizing the generation of anti-ad blocking techniques. Techniques to block advertisements and block ad blockers have developed over the years are parallel to a cat-and-mouse game, each side constantly on-upping the other to gain an advantage.

## Technologies used

There are three primary methods in which we can block advertisements on modern computers, each has its nuance, including DNS modifications (1); browser extensions (2); Software (3). Regarding DNS modification there are two primary means, host-based and network-based. Host-based DNS blocking requests editing the host file to redirect advertisement hostnames to a block hole – typically 0.0.0.0. Host-based ad blocking is typically inefficient in catching most tracking providers’ content and is inefficient with extensive lists, however network-based solutions such as an Ad blocking DNS provider negate these shortcomings. Network-based ad blocking can be done by using a VPN, proxy, or changing the DNS provider of a machine. For users who cannot use advanced techniques like changing their DNS may use ad blocking software, which can use internal proxies, while others act as networking software. AdMuncher is an example of a host-based ad-blocking software that has extended functionality of DNS-based solutions. It can block elements such as JavaScript pop-ups and ad containers along with the advertisement.

The most used technology is a browser extension. Browser extensions add functionality to a browser, for example, blocking advertisements, however, there are many other functional extensions. There is a unique distinction between an advertisement blocker and a content blocker. Ad blockers are limited to blocking the display of advertisements while content blockers block a range of teaching elements such as advertisements, fingerprinting, and CNAME cloaking. Raymond Hill created one of the most used browser extensions, uBlock Origin, an open-source solution to blocking browser tracking mechanisms.

Figure 1 - uBlock Origin Logo (Origin)

## Adblock methods

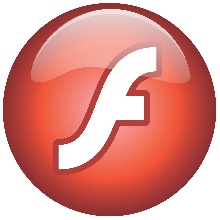
The history of blocking advertisements on the web is short but robust. Initially blocking flash elements on browsers developed into blocking scripts and pop-ups. Then advertising conglomerates flooded the internet with visual advertisements, taking up significant real estate on pages. This led to the rise of a popular ad blocking technique known as filter lists. Filter lists are lists filled with rules regarding what types of elements, hostnames, and other media are filtered when a page is loaded. ad blocking extensions like uBlock Origin use these filter lists and parse a page each time it is loaded to permit or deny specific elements.

Figure 2 - Retro Adobe Flash Icon (Adobe)

## New Issues

In late 2019, Aeris, a user on GitHub posted an issue on uBlock Origin’s repository regarding first-party tracking. The site, https[.]liberation[.]fr embedded a seemingly *first-party* tracker within their domain (Aeris). This subdomain, however, pointed to a third-party tracking provider: Eulerian. In 2019 this was a new form of tracking internet users and was uncommon, however, usage has grown over the past 3 years. Nicknamed CNAME cloaking due to how the true destination is hidden, this method is superior to many other bypass methods. For example, tracking providers can use randomized subdomains (1); CNAME resolution is done with DNS resolver (i.e., Cloudflare, Google, local ISP) (2); filtering IP address is inefficient and slow browsing (Abrams). This provides Adblock maintainers a challenge to overcome, the primary being how DNS queries a host for a CNAME record.

# Hypothesis

There has been an ongoing technical war between tracking protection technologies and anti-tracking technologies. CNAME tracking is one of the many methods used to track users of the internet including, bounce trackers, fingerprinting, and AdBlock detection.

## What is a CNAME

A canonical name (CNAME) is a DNS record that points to a domain name. Although, it can point to another CNAME (Mockapertric). CNAME’s are used to ensure that DNS records are updated in a hierarchal stretch, rather than manual modification; this ensures that there is a limited area to make mistakes. It is often the case that websites with use CNAMES within their subdomains that point that the root domain, this limited the reconfiguration costs when an IP address is changed; this is what tracking providers exploit to bypass adblocking techniques.

## CNAME Cloaking

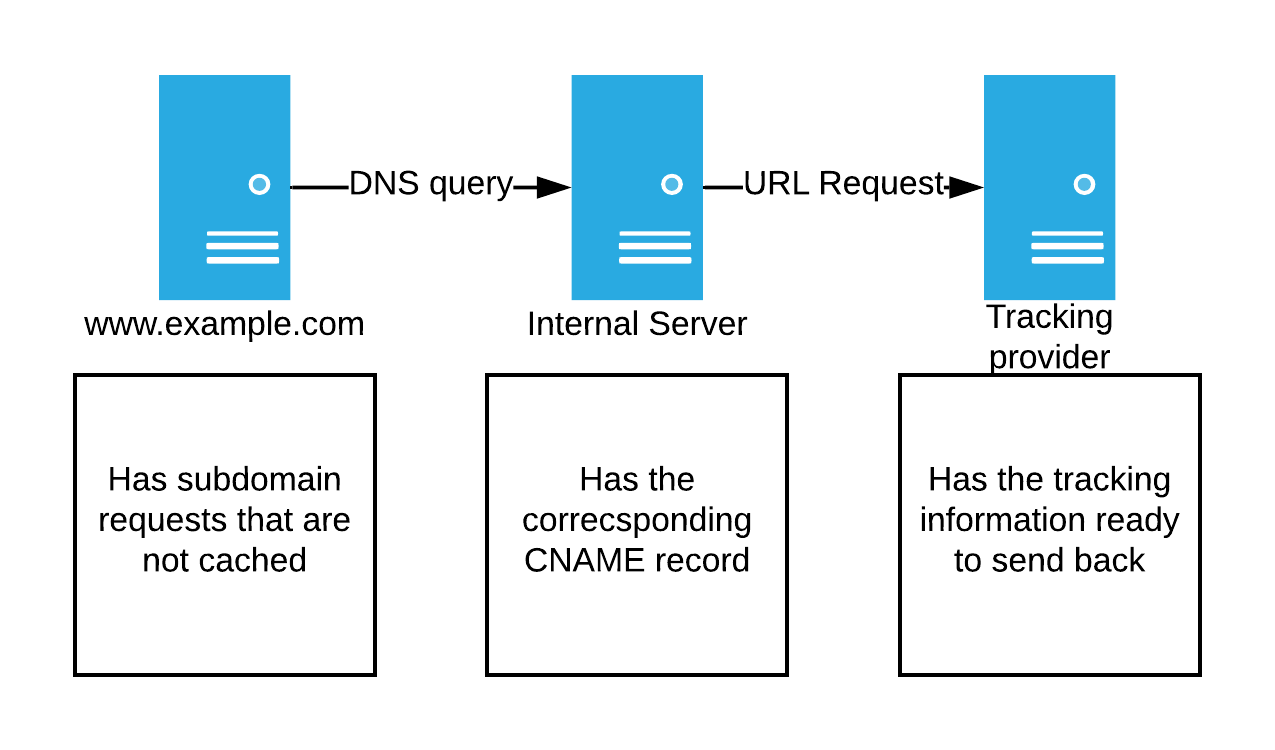
CNAME cloaking is a technique used to bypass browser-level methods of tracking protection (Adobe Inc.). By using CNAMES’s websites can trick a browser into believing it is contacting a first-party domain – which they initially are – however, the subdomain contacted resolved into a third-party domain (Aeris). This clock method works because browser extensions are not permitted to access DNS requests, including CNAME queries – Firefox has asses the ability to access DNS queries, though Chromium and WebKit browsers will likely never see this feature. (Gorhill) To combat this Adblock extensions would have to maintain massive lists containing all subdomains that use CNAME cloaking, which is inefficient and impotent on some devices. Third-parties domains look like a first-party domain and therefore have access to all first-party resources including cookies, browser parameters, and other metadata within the browser’s session (Figure 1). 

Figure 3 - CNAME Cloaking Route

# Research Tasks

I tasked myself with finding subdomains a common site that may be using CNAME cloaking. To do this I first needed to capture traffic some a popular website, then need to parse the DNS requests and develop logic to find CNAME cloaking.

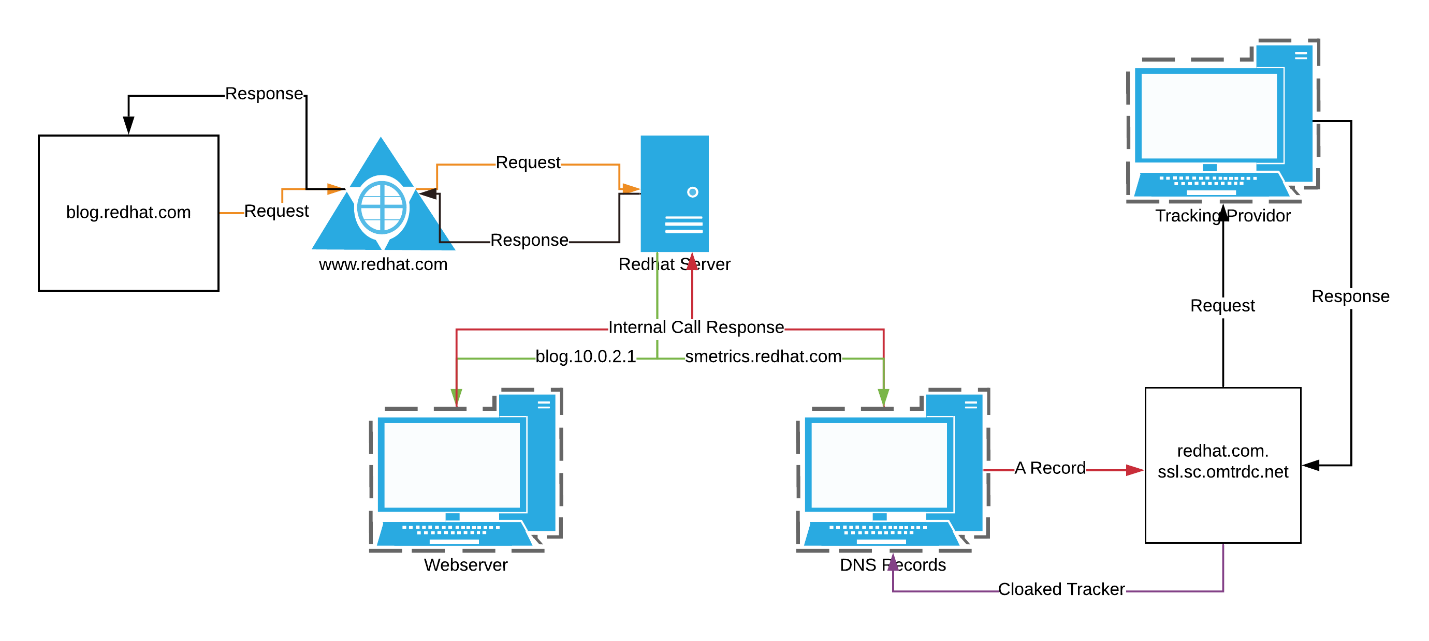


Figure 4 - Flow of HTTPS request

For the first step, I created a fresh VM with Windows 10, with Wireshark monitoring all packets on my host machine. I then browsed the site RedHat, I stayed on that page for around 15 seconds before stopping my Wireshark scan. After filtering for only DNS requests in Wireshark, I could find quite a few CNAME queries. A few looked promising, a sub-domain of RedHat: smetrics[.]redhat[.]com that contained the word metrics, a word commonly used to track users (Figure 4).

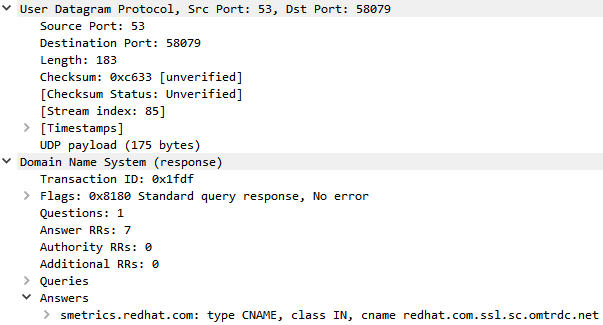


Figure 5 - Wireshark CNAME query

After examining some more packets I began to build a script that would parse each packet, check if it was using CNAME cloaking, then print the results. I used Scapy to iterate through the packets and create objects that could be parsed. From each packet, I extracted the fields from AN record because that would contain the necessary DNS request information, the *rrname,* and *rdata* cells were the most important because they contained the source and destination hostnames (at points IP addresses).



Figure 6 - CNAME Cloaking script init method

Second, I would iterate through each packet and ensure that it was a CNAME packet (DNS query type 5).



Figure 7 - Method to Parse Packets

I then compared both the source and destination IP addresses and the source and destination domains. If neither the IP addresses nor hostnames matched, I consider it possible CNAME cloaking. From the resulting packets, I would remove trackers that did not use CNAME Cloaking to hide their requests.



Figure 8 - Method for DNS Cloaking check

The remaining packets were legitimate CNAME requests or CNAME Cloaking requests. To discern between the two, I used a modified Adguard tracking filter list to determine if the resolved domain is associated with tracking.



Figure 9 - Method to filter only tracking cloaking

Finally, I printed out my result for this website that contained 2 entries of the same RedHat sub-domain: smetrics[.]redhat[.]com which resolved to smetrics[.]redhat[.]ssl[.]sc[.]omtrdc[.]net which is under the domain name omtrdc[.] a tracking provider (Figure 10).



Figure 10 - Cloaking subdomains on Redhat

## Drawbacks

There are limitations to the script I have created, I limited it to filter lists (1); hardcoded to read *pcap* files (2); post browsing analysis (3); will not block CNAME Cloaking hosts (4). When searching for domains that are using CNAME cloaking if the domain is not already associated with a maintainer list such as EasyList, Adguard, or Dan Pollock’s the program cannot tell if the domain is used for tracking. With the use of cloud services such as AWS, MS Azure, and VPS it’s hard to find out if a domain is tracking. The script cannot tell if the website is currently tracking a user it is used to examine one browser session DNS queries. Similarly, the program is not based on real-time searching parsing the information can only be done after it is capture with Wireshark or TCPDump, however could be changed to act as a proxy and monitor browsing sessions. Because of this, it will only be able to tell if a sub-domain is using CNAME cloaking and cannot block or intercept the request. Additional functionality would have to be added to make this a more robust tool.

## Results

There are two methods to finding CNAME trackers, dynamic or static DNS resolution. Dynamic resolutions monitor only DNS requests that are not known to be a tracking domain. The software will then have to resolve all queued DNS requests and check the resolved queries against a filter list. While this will decrease the overall bandwidth usage of the software, it may cause slower loading speeds and is a potential privacy flaw. Alternatively, the software could resolve all DNS requests, this software would find CNAME cloaking by default–should the resolved domain match one in a filter list, because the default DNS server filters by default. Current solutions exist such as Adguard, NextDNS, and Pi-Hole. It would be preferable to have a solution such that locally parses and filters resolved DNS requests to ensure that personally identifiable information is not leaked and can run on a multitude of machines.

Regulations may also be another means to limit advertising tracking, including CNAME tracking. In countries where the GDPR is in effect, websites are required to disclose information about their tracking habits and are held financially liable for consumer data. Implementing similar laws in the United States could cause a diminished market for advertising providers to prey on consumer data.

CNAME Cloaking to track internet users is an effective method that is hard to negate. The method uses technologies developed in the 1980s to bypass current browser-level protections. Due to the effective nature of this technique’s usage will continue to grow; it is hopeful to believe that open-source projects like uBlock Origin will continue to combat tracking methods developed by advertising providers.

# Appendix

* 1. [CNAME Cloaker finder](https://github.com/Demon-tk/Web-Application-Security-Blog): The repository for my tool to find CNAME trackers.
  2. [uBlock Origin](https://github.com/gorhill/uBlock): The repository for the open-course content blocker.
  3. [AdMuncher](https://www.admuncher.com): The homepage for the adblocking software AdMuncher.
  4. [Redhat](https://www.redhat.com/en): The homepage for the “enterprise open-source leader”
  5. [Adguard Tracking Filter List](https://github.com/AdguardTeam/AdguardFilters/blob/4ed4c9a0e45131b42b4796eef984a1eb3427ae54/SpywareFilter/sections/tracking_servers.txt): The repository for AdGuard filter lists
  6. [Wireshark](https://www.wireshark.org): The homepage for the packet capture and analysis tool Wireshark.
  7. [TCPDump](https://www.tcpdump.org): The home page for TCPDump, a command-line packet analyzer.
  8. [EasyList](https://easylist.to): The homepage of EastList where there is a collection of filter lists.
  9. [Dan Pollock list](https://someonewhocares.org/hosts/): The homepage of a large, curated filter list.

# References

Abrams, Lawrence. *Bleeping Computer*. 20 November 2019. HTML. 26 April 2021. <https://www.bleepingcomputer.com/news/security/ublock-origin-now-blocks-sneaky-first-party-trackers-in-firefox/>.

Adobe. *Adobe Flash Logo*. Image.

Adobe Inc. "Data Collection CNAMEs and Cross-Domain Tracking." 25 September 2020. *Web Archive.* Document. 27 April 2021. <https://web.archive.org/web/20200925065025/docs.adobe.com/content/help/en/id-service/using/reference/analytics-reference/cname.html>.

Aeris. *Github*. 10 November 2019. HTML. 26 Arpil 2021. <https://github.com/uBlockOrigin/uBlock-issues/issues/780>.

Gorhill. "Address 1st-party tracker blocking." 19 November 2019. *Github.* Document. 27 April 2021. <https://github.com/uBlockOrigin/uBlock-issues/issues/780#issuecomment-555643036>.

Mendes, Nate. *Web-Application-Security-Blog*. Vers. 1.0. 3 May 2021. Repository. <https://github.com/Demon-tk/Web-Application-Security-Blog>.

Mockapertric, P. *Internet Standard*. November 1987. HTML. 27 Arpril 2021. <https://tools.ietf.org/html/rfc1035>.

Origin, uBlock. *uBlock Origin Logo*.