Domain Fronting Through Microsoft Azure and CloudFlare: How to Identify Viable Domain Fronting Proxies

Charles Miller
Department of Information Systems
University of Maryland Baltimore County
Baltimore, Maryland, United States
cmiller1111@gmail.com

Michael Pelosi
Department of Computer Science
Texas A&M Texarkana
Texarkana, Texas, United States
mpelosi@tamut.edu

Michael Scott Brown
Department of Information Systems
University of Maryland Baltimore County
Baltimore, Maryland, United States
michaelb@umbc.edu

***Abstract*— Domain fronting is a technique for internet connection obfuscation and also internet censorship circumvention that uses different domain names in different communication layers of an HTTPS connection to discreetly connect to a different target domain than is discernible to third parties monitoring the traffic. Domain fronting involves using different domain names in the DNS/SNI headers of the visible HTTPS packet and the Host header of the encrypted HTTP packet. If both domains are served from the same Content Delivery Network (CDN), then the CDN may proxy the request to the address specified in the HTTP header after unwrapping the TLS encrypted HTTPS payload. As a result, connection monitoring outside the CDN server network will not be able to ascertain where the connection packets are ultimately going to or coming from.**

This paper explores and expands upon methodologies for identifying viable domain fronting proxies within the CloudFlare and Microsoft Azure Content Delivery Networks (CDNs). Despite claims by Microsoft to block domain fronting behavior on all Azure products, our research successfully identified 14 Azure edge servers on 6 Microsoft domains that successfully proxied domain fronted traffic. Comparably, the CloudFlare CDN yielded over 2000 viable proxies among the 30 domains tested, with an average of 6.61 viable proxies per domain (excluding outliers).

Unlike similar research conducted in 2017-2018 by penetration testers Vincent Yiu and Raphael Mudge [14], [23], no consistent pattern was found between a domain's DNS record and its ability to proxy fronted traffic. As an example, the domain huffingtonpost.com contains a different CDN address in its DNS records but still exhibited three subdomains as proxy-willing CloudFlare edge servers. In response to these findings, this paper presents a methodology, subdomain enumeration using brute force scripting, as a more effective method of identifying domain fronting proxies within popular CDNs.

Additionally, the domainfuzzer.py application developed as part of this study plays a crucial role in the analysis of viable domain fronting proxies within a CDN. By providing a user-friendly tool, domainfuzzer.py enables non-technical users to identify CDN edge servers capable of proxying domain fronted traffic. For more technical users, this methodology can easily be adapted to any CDN, empowering users to build their own domainfuzzer.py for use on a CDN of their choosing, should they be so motivated.

Keywords—Domain Fronting, Cybersecurity.

I. Introduction

Domain fronting is a web traffic obfuscation technique popularized by the 2015 UC Berkeley paper titled "Blocking-resistant communication through domain fronting" [9]. In this paper Fifield describes a means of using Content Delivery Networks (CDNs) to forward the contents of HTTPS packets to a forbidden location hosted on the same CDN via customized HTTP packet headers. Since this discovery domain fronting has been widely used as a method of web traffic obfuscation for users wishing to retain their internet privacy, however many of the largest CDN providers have decided to block domain fronted traffic in order to prevent malicious use by would-be attackers. In 2018 Google and Amazon announced they would be blocking all domain fronted traffic [4], and Microsoft followed suit in 2022 [10]. Additionally, many of the techniques used by penetration testers in 2017-2018 for identifying viable CDN edge instances to use as domain fronting proxies have been rendered obsolete by changes to public DNS records and responses.

With these global improvements to operational security, it is more difficult than ever for users to identify viable domain fronting proxies. DNS records of all websites vary so greatly from domain to domain that the previously established methods of proxy identification often yield more false positives than true, viable proxies. While AWS documentation may have previously instructed web developers to expose their unique CDN edge instance URLs in the CNAME record, this is no longer the case. Now, even identifying domains to test for domain fronting is difficult. Many companies have chosen to remove CNAME records from their public DNS and others have gone as far as to remove any mention of their CDN provider from their DNS records at all; further disrupting one's ability to collect intelligence on a target network. Domain fronting requires a considerable amount of technical proficiency to understand and even more so to use manually. The technical knowledge needed combined with increasing efforts to hide CDN information from

the internet has made identifying viable domain fronting proxies quite formidable.

This paper presents a method for identifying domain fronting proxies using subdomain enumeration that ensures all returned addresses are true positives. Using a free static website created on the target CDN, this method presents a set of steps to collect open-source intelligence, build a simple brute forcing script in Python, and enumerate a list of web addresses that will automatically proxy domain fronted traffic. As a proof of concept this methodology is first tested on the CloudFlare CDN using well known CloudFlare hosted domains, and second on the Microsoft Azure CDN to examine the veracity of Microsoft's recent claim to block all domain fronted traffic.

For many, domain fronting means safety from persecution by a corrupt government; for others it is access to world news. However, freedom of privacy invariably means freedom of criminal activity. Anonymizing services that utilize domain fronting like Tor Browser, Telegram, and Signal, are also used by criminals to conceal incriminating communications and hide entire trafficking networks. This duality in how domain fronting has been used has created much ethical debate on the line between internet privacy and government oppression. Unfortunately (or fortunately for the citizens seeking a free internet), the task of identifying and blocking domain fronted traffic is often more complicated and expensive than anticipated. The feasibility of domain fronting relies on the unwillingness of government censors to block an entire Content Delivery Network just to prevent a small percentage of domain fronted traffic to that network. To avoid the "collateral damage" of blocking domain fronted traffic at the IP layer many world governments, including China and the United States, implemented an expensive solution called Deep Packet Inspection that effectively breaks and rebuilds TLS encrypted tunnels at the ISP firewall in order to examine packet contents. This tug of war between internet privacy advocates and the law authorities of both democratic and authoritarian governments has inspired much innovation into the detection, prevention, and improved effectiveness of domain fronting.

II. History of Domain Fronting

The need for Content Delivery Networks arose out of the global increase in popularity of content providers. Companies like Netflix, Facebook, YouTube and Google were developing their source content in the U.S. but delivering it to locations thousands of miles away. Delivering content over such long distances yields considerable latency and struggles with the inevitable packet loss of data connections that large. The solution was Edge Computing: dedicated servers located at the "edge" of a global network in high population regions across the globe that contain cached copies of content and serve it directly to geographically nearby requesters, thus removing the latency and unreliability of planet-sized connections. However, CDNs do not keep cached copies of all content on every edge server in every region, as that would be impractical, especially for content that is infrequently requested. For any content not cached locally on the edge server receiving the request, the edge server will pass the request to the content source so that it may complete the request; passing the response (content) back to the edge server, who then hands it off to the requester. This means that CDN edge servers will work as geographical proxies, passing communications between a nearby requester and any site on the CDN (see Figure 1).

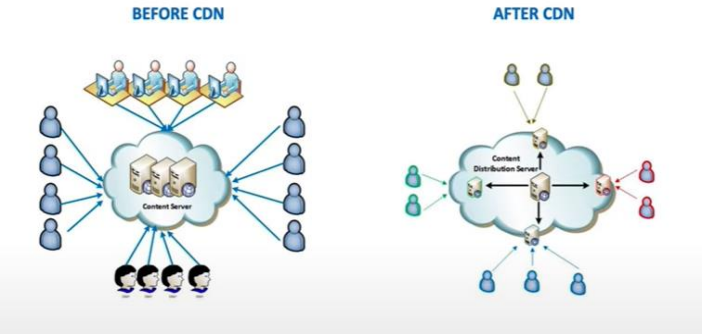

Fig. 1. Content Delivery Before and After CDN Architecture [8]

A. How Domain Fronting Works

A user wishes to view the forbidden website blocked-example.com but, as the name implies, censorship of some kind is blocking any traffic addressed to this site. We know blocked-example.com to use the CloudFlare CDN to host its web services and if we also know another website, allowed-example.com, to host its web services on the CloudFlare CDN , we may be able to "domain front" our HTTPS traffic through allowed-example.com using the externally visible (not encrypted) DNS and SNI headers of HTTPS packets. CloudFlare.com describes the SNI (Server Name Index) header as "somewhat like mailing a package to an apartment building instead of a house" [22], meaning that after an HTTPS packet reaches a CDN edge server addressed in the DNS/SNI headers, it will decrypt the packet and handle the rest of the routing internally, out of view of a would-be censor's firewall. Typically, these HTTPS headers would share the same destination as the HTTP Host header contained within the packet, and when a user browses the internet using a common web browser like Google Chrome, Mozilla Firefox, or Opera, they do. However, by using HTTPS DNS and SNI headers addressed to allowed-example.com and a customized HTTP Host header addressed to blocked-example.com concealed within the encrypted payload of the packet, we allow the CDN edge server to function as it was designed; a geolocational proxy that handles its own internal routing (see Figure 2).

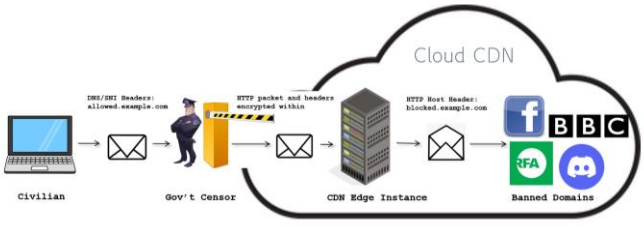

Fig. 2. How censorship is circumvented using a fronted domain

Although it was invented for censorship circumvention, domain fronting is frequently used to cloak Command & Control (C2) communications between criminal hackers and compromised devices on a botnet. These covert communication channels are pivotal in a hacker's ability to maintain a foothold

inside a corporate network that actively seeks to identify compromised devices (see Figure 3). It is this malicious use of domain fronting that has inspired so much effort to identify and prevent domain fronted traffic, although some claim the true motivation behind Google, Amazon and now Microsoft's bans are due to pressure from the pro-censorship governments in Russia and China [12].

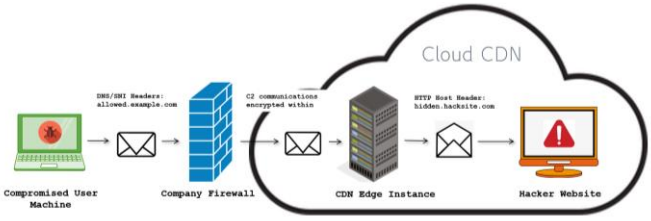

Fig. 3. How a firewall is bypassed to ex-fill sensitive data using domain fronting

Once the robustness of CDN internal routing was discovered, penetration testers like Raphael Mudge [14, 25] and Vincent Yiu [23-24] realized the next logical step was enumeration of domains available to front traffic through. This continued to prove fruitful as Mudge [14] discovers AWS's own edge servers (a0.awsstatic.com) can be used as proxies and Yiu [23] then discovers AWS CloudFront documentation instructs developers to expose their CDN instance's URL in their public DNS CNAME record [19]. Because it was commonplace at this time to include an edge server address in the CNAME record of any CDN hosted web services, Yiu was able to enumerate a large list of potentially viable domains including the Arizona State government's website cdn.az.gov. He later proves this site to be viable as a domain fronting proxy by fronting traffic through the domain to a separate site hosted within AWS CloudFront.

Since the discovery of these enumeration techniques, however, website owners have implemented OpSec updates to their public facing infrastructure that impede the ability to identify and use their web addresses as domain fronting proxies. Many companies have elected to completely remove all CNAME records from public DNS, with some even removing any mention of CDN provider name from their records altogether. Additionally, all public DNS servers now either drop or reject any query containing the "ALL" parameter, making large scale record enumeration difficult and time consuming. Because DNS records are no longer a reliable clue in identifying CDN edge instances of a web domain, enumeration techniques that focus on finding "likely" domain front proxies, like those used by Mudge and Yiu, have been rendered obsolete.

III. Literature Review

Along with the discovery of domain fronting [9], Fifield et al. released an open-source API called MEEK [16] for the popular anonymizing web browser, The Onion Router (Tor), that allows users to obfuscate their web traffic using domain fronted connections to existing MEEK servers provided by UC Berkeley in both the Amazon and Google CDNs. Since the release of this paper, MEEK has become widely used as a Tor "pluggable transport" and has inspired considerable research into the detection and prevention of domain fronted traffic through the MEEK pluggable [11], [13], [18], [20]. These innovations in detection/prevention are typically neural networks trained to identify the "domain fronted traffic through Tor pluggable MEEK" using only TCP packet sequencing and no Deep Packet Inspection [20].

However, alongside these innovations in censorship technology came additional innovations in the circumvention techniques around them. Sheffey and Aderholdt [17] propose adversarial techniques that allow "traffic shaping" TCP packets so that the timing and sequences (especially during initial TLS handshakes of HTTPS communications) are dynamically altered with random delays, making the TCP packet sequencing of Wang et al.'s [19] neural networks unusable. Despite years of research attempting to thwart domain fronting as an obfuscation technique, the fundamental concepts on which domain fronting is based upon have yet proved insurmountable: (1) using a popular CDN ensures an undesirable amount of "collateral damage" for any censor wishing to block this traffic with traditional ACL-based blocking methods, (2) the internal routing of CDN edge servers to any location within the CDN is a feature of most cloud networks. Unless the provider of a Content Delivery Network changes internal routing protocols at the fundamental level, domain fronting will continue to be possible.

The python application FindFrontableDomains.py [3] attempts to achieve the same goal of identifying domain fronting proxies through subdomain enumeration. The application makes a large number of queries to public DNS servers and scans the results for CNAME records that contain urls indicative of CDN edge instances. Namely, web addresses found in the same name record that contain the name of a popular CDN provider including CloudFlare, AWS CloudFront, Google Cloud, Microsoft Azure, etc. Unfortunately, FindFrontableDomains.py does not attempt to successfully front traffic through the identified subdomains and, upon testing with domain "nginx.com", it returns a set of subdomains that do not function as proxies, aka false positives. The solution designed in FindFrontableDomains.py likely worked in 2017, when organizations commonly left their CDN instance addresses in their public CNAME records. As discussed in Section 2, companies like Nginx have removed their instance addresses from their public CNAME records, rendering a solution of this nature obsolete.

In a 2020 DEFCON presentation, Erik Hunstad [12] presents a means of circumventing Google and Amazon's domain fronting blocks by leveraging the ESNI header of HTTP packets in communications encrypted with the more modern TLS v1.3. Called "domain hiding" this technique uses an almost identical solution as domain fronting, just with a different HTTP header. Unfortunately, TLS v1.3 has not seen the widespread use that was expected of it and Amazon/Google's CDN servers now refuse traffic containing the ESNI header at all (likely due to Hunstad's discovery [12]).

IV. Methodology

A. Assessment Parameters

This method adapts Yiu's methodology away from using DNS records to identify likely proxies, electing to enumerate

subdomains that successfully respond to a domain fronted request, confirming beyond doubt the viability of return web addresses. Using a static website hosted on the target CDN and any domain known to host web services using the target CDN, any requests that successfully return the contents of the static site are known-working domain fronting proxies within the target CDN. Because identifying domain fronting proxies using subdomain enumeration is based soundly in cloud networking concepts it should be easily adaptable on any other CDNs that do not explicitly block traffic with mismatching HTTPS and HTTP headers.

CloudFlare provides one of the largest CDNs in the world, serving 152 regions globally with over 7.5 million active websites [6]. With a free CloudFlare account one gains access to the Pages dashboard where users can create a web page and connect it to a GitHub repository. The static webpage, domainfronter.pages.dev, is connected to a repository containing a small text file, df.txt, with the contents "domain fronting works!" As in Yiu's work, this file will serve as the proof of concept when successfully requested from one website while existing on a completely separate site.

B. Identifying Domains to Test

In Domain Fronting via CloudFront Alternate Domains [24], Yiu uses DNS brute forcing to collect the top 1 million DNS CNAME records and then searches through the results for any records that contain the tell-tale "*.cloudfront.net" URL of a CDN edge instance on the CloudFront network. In this proof of concept, Yiu finds several domains with their CDN instance address publicly exposed and successfully domain fronts traffic through the Arizona State Government (cdn.az.gov) web domain as proof. Since then, however, companies have begun hiding their CDN instance addresses by removing these CNAME records from public DNS. State of Authority or SOA records can be used as a default DNS response when queried for a record that doesn't exist. Here we can see that of the 29 known CloudFront hosted domains queried, none of them have existing CNAME records and only 20 have a cloudflare.com address in the default SOA record (Table 1).

However, even though these records do not contain exact edge server addresses like they did for Yiu in 2017 [23], we can still see these records point to CloudFlare resources, confirming they are hosted by our target CDN.

These queries were conducted using a text file containing a list of websites known to host their web services using the CloudFlare CDN [1], [5], [21] and the following command:

dig -f domains.txt -t CNAME | grep ".cloudflare.com"

Using the DNS querying tool, dig [7], this command iteratively queries public DNS for the CNAME record of all domains in the list and then pipes the results out to a grep search for any records containing the typical attribute of a CDN instance address, ".cloudflare.com".

C. Testing Identified Domains

To determine whether or not these web addresses point to actual CDN edge proxies, we must make a successful HTTPS GET request for df.txt from the viable web address. Because df.txt is hosted on a completely different website, domainfronter.pages.dev, one would anticipate an error when requesting the file from a website like http://xeroshoes.com. However, by using a custom HTTP header hidden within the HTTPS payload addressed to domainfronter.pages.dev, the XeroShoes.com instance on the CDN edge server automatically proxies traffic between the requester and domainfronter.pages.dev, passing the request and response (df.txt) between the two. This is achieved using the wget command and the following syntax [14], [15], [23]:

wget -q -O - -U demo http://xeroshoes.com/df.txt --header "Host: domainfronter.pages.dev"

Here we can see the response (Figure 4):

```
$dig wget -q -O - -U demo
http://xeroshoes.com/df.txt –header "Host:
domainfronter.pages.dev"
domain fronting works!
```

Fig. 4. Domain fronted GET request and positive response

If one were to browse http://xeroshoes.com/df.txt using a normal web browser, you would receive an error from the Xero Shoes web server stating that the df.txt file was unable to be found, yet here we receive a response. We have successfully smuggled our traffic through an HTTPS (TLS) encrypted tunnel with XeroShoes.com to another location within the CloudFlare CDN. Any censors between our machine and the CloudFlare edge server would be unable to view the true domainfronter.pages.dev destination of our packets.

Using the same wget syntax all 30 known-CloudFlare sites were tested. Of the 20 websites with cloudflare.com addresses in their public records, only 15 successfully returned the df.txt file. Of the other 9 sites that did not contain a CloudFlare address in the SOA record, 4 still successfully proxied domain fronted traffic: SourceForge.net, Vimeo.com, Shopify.com, and Bloomberg.com. This shows that Yiu's method of identifying domain fronting proxies using clues in the public DNS record is not reliable and returns both false positives and false negatives. This methodology may have been a reliable means of identifying domain fronting proxies in 2017, but modern DNS configurations vary so widely from target to target that this is no longer the case; see responses in Table 2.

D. Subdomain Enumeration

As seen, domains such as SourceForge.net and Bloomberg.com exhibit none of the tell-tale signs of a publicly exposed CDN instance, however, surprisingly, the top level domains still successfully proxy domain fronted traffic. As such, if CNAME records are no longer a reliable indicator of CDN edge proxies, there may be more false negatives within the list of CloudFlare domains that remain yet unidentified. With this knowledge we can hypothesize that there are other subdomains within these domains that point to CDN edge instances (and would therefore proxy our traffic), we need only identify them. If we are able to find these subdomains and prove that they will proxy our traffic we can create a robust and 100% reliable list of viable proxies on any given domain within a CDN.

Fuzzing is a method of identifying all entities of a given type on a web server by resending (brute forcing) the same request

multiple times, and each time replacing (fuzzing) one or many parameters with those from a list of commonly seen entities. Penetration testers can then analyze the server responses for variations that tell them if a given entity exists. Using programs like ffuf or gobuster, we can fuzz web servers for entities like existing usernames, SQL directories, or even existing subdomains. However, this paper seeks to take this enumeration a step further and identify subdomains that also point to functional CDN edge proxies. Using Python programming to take user input as the domain and fuzzing subdomains from a list of top 1000 popular subdomains [2] into Yiu's wget parameters, we can test for responses that include the contents of our df.txt file ("domain fronting works!") and report back a list of successful subdomains.

The Python app and BitQuark subdomain list used in this paper can be found at https://github.com/cmiller-umbc/subdomain_fuzzer. As shown below (see Figure 5), domainfuzzer.py allows the user to input a desired domain and reports back a list of subdomains through which traffic was successfully domain fronted:

```
$python3 domainfuzzer.py
Domain: nginx.com
fuzzing top 1000 subdomains for nginx.com...
Successful subdomains:
pages.nginx.com
900 subdomains remaining...
800 subdomains remaining...
700 subdomains remaining...
600 subdomains remaining...
500 subdomains remaining...
400 subdomains remaining...
300 subdomains remaining...
200 subdomains remaining...
100 subdomains remaining...
cdn.nginx.com
email.nginx.com
www.nginx.com
0 subdomains remaining…
```

Fig. 5. Subdomain enumeration of nginx.com using domainfuzzer.py

As shown all 1000 subdomains are iteratively tested over nginx.com and 4 are reported back as positive matches. Using Yiu's proof of concept we confirm these 4 subdomains to work as domain fronting proxies (see Figure 6).

```
$wget -q -O - -U demo http://pages.nginx.com/df.txt --
header "Host: domainfronter.pages.dev"
domain fronting works!
 $wget -q -O - -U demo http://cdn.nginx.com/df.txt --
header "Host: domainfronter.pages.dev"
domain fronting works!
 $wget -q -O - -U demo http://email.nginx.com/df.txt --
header "Host: domainfronter.pages.dev"
domain fronting works!
 $wget -q -O - -U demo http://www.nginx.com/df.txt --
header "Host: domainfronter.pages.dev"
    domain fronting works!
```

Fig. 6. Manual domain fronted requests to nginx.com confirming proxy viability

E. *Python Scripting*

While domainfuzzer.py contains other elements, such as threading, to improve efficiency, the crux of the script's functionality lies within a single while loop (see Figure 7).

```
while len(subdomains) > 0:
            subDomain = subdomains.pop()
            cmd = "wget -q --connect-timeout=2 -O - -U
demo http://" + subDomain + "." + domain + "/df.txt --
header \"Host: domainfronter.pages.dev\""
            sp = subprocess.Popen(str(cmd), shell=True,
stdout=subprocess.PIPE, stderr=subprocess.PIPE,
                                  universal_newlines=True)
            rc = sp.wait()
            out, err = sp.communicate()

            if out == "domain fronting works!":
                termcolor.cprint("   " + subDomain + "."
+ domain, "blue")
                results.writelines(subDomain + "\n")
                counter = counter + 1
            elif len(subdomains) == 0:
                break
            elif len(subdomains) % 100 == 0:
                print(str(len(subdomains))+" subdomains
remaining...")
```

Fig. 7. Code snippet from domainfuzzer.py of subdomain enumeration loop

During each iteration of the while loop a variable "cmd" is set to Yiu's wget statement, fuzzing in subdomains from the BitQuark list. Using the Python subprocess library the cmd string variable is passed to the host operating system as a shell command, sending our custom HTTPS request to the desired address. The response is then examined for the contents of df.txt. If "domain fronting works!" is received back as a response the subdomain is printed to stdout and our total count of viable subdomains increases.

F. *Adapting the Model to Microsoft Azure and other CDNs*

As mentioned earlier in this paper, Microsoft recently announced a ban on domain fronting traffic within the Azure CDN starting Nov 8th, 2022. To test this ban we can adapt domainfuzzer.py to enumerate subdomains on known Azure hosted web domains using the same basic cloud networking concepts as before. First we build a free Azure static website hosting the df.txt file: delightful-sand-056d73f0f.2.azurestaticapps.net. Then update the Host header parameter of the wget statement contained within the while loop of domainfuzzer.py to this address (instead of domainfronter.pages.dev). Once these changes are made we run the new script, appropriately named azure_domainfuzzer.py, and pass it domain names of known-Azure hosted websites (see Figure 8).

```
└──╼ $python3 azure_domainfuzzer.py
Domain: microsoft.com
fuzzing top 1000 subdomains for microsoft.com...
Successful subdomains:
   security.microsoft.com
900 subdomains remaining...
   portfolio.microsoft.com
800 subdomains remaining...
   project.microsoft.com
700 subdomains remaining...
600 subdomains remaining...
500 subdomains remaining...
400 subdomains remaining...
300 subdomains remaining...
   ads.microsoft.com
200 subdomains remaining...
   es.microsoft.com
100 subdomains remaining...
0 subdomains remaining...
   mail.microsoft.com
```

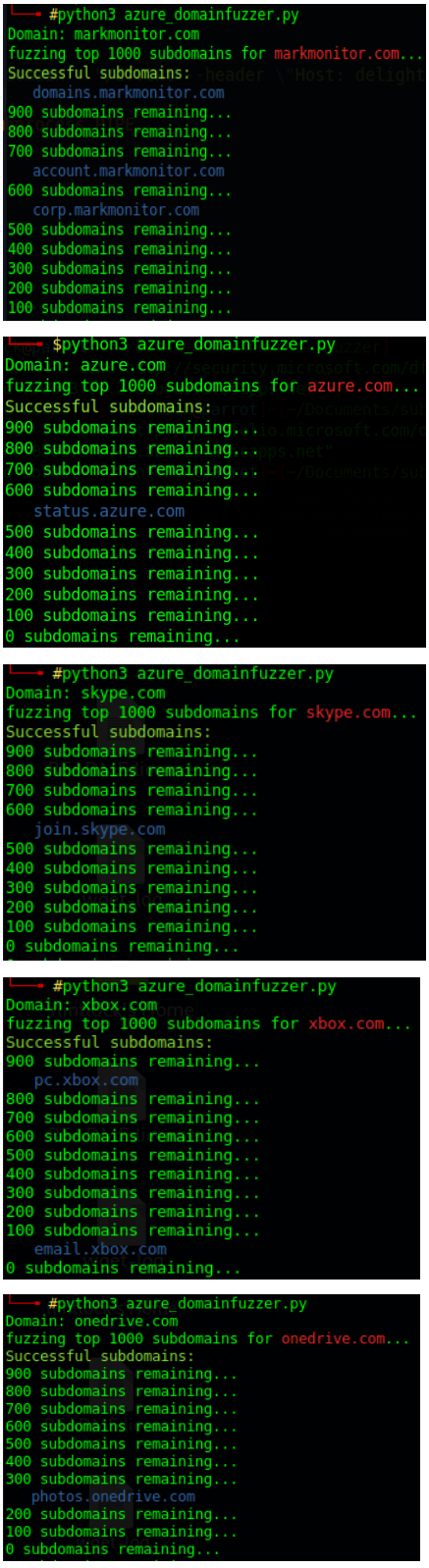

Fig. 8. Successful subdomain enumeration of six Azure CDN domains

As of February 26th, 2023 the following Azure resources were identified as fully functional domain fronting proxies: security.microsoft.com, portfolio.microsoft.com, project.microsoft.com, ads.microsoft.com, es.microsoft.com, mail.microsoft.com, domains.markmonitor.com, account.markmonitor.com, corp.markmonitor.com, status.azure.com, join.skype.com, pc.xbox.com, email.xbox.com, and photos.onedrive.com. To ensure accuracy all identified proxies were manually tested using Yiu's wget syntax and the contents of df.txt were returned.

V. Results

Viable domain fronting proxies were successfully identified in both the CloudFlare and Microsoft Azure CDNs using this method of brute force subdomain enumeration. Despite a November 8th, 2022 announcement from Microsoft claiming functionality for their Azure Front Door, Azure Front Door (classic), and Azure CDN Standard products that "block any HTTP request that exhibits domain fronting behavior" 14 viable domain fronting proxies on 6 of Microsoft's own domains were successfully proven to proxy domain fronted traffic. Unsurprisingly, viable subdomains were identified more frequently and at a greater average quantity on domains hosted within the CloudFlare CDN, with over 2000 viable proxies identified among the 30 domains tested. Nearly all of these domains yielded at least one viable proxy with only 3 CloudFlare sites refusing all domain fronted traffic. Two of the domains enumerated, medium.com and sourceforge.net, exist as extreme outliers in the number of viable subdomains discovered with 996 and 970 respectively. Ignoring these extreme outliers, CloudFlare domains return an average of 6.61 viable proxies using this methodology. See Tables 3 and 4.

While analyzing the results a pattern was identified among the subdomains most commonly returned as true positive. Over half of the CloudFlare domains tested have a www subdomain that proxies domain fronted traffic. Other frequently seen subdomains include api (10), cdn (8), mail (5), and ftp (5). See Table 5.

No connection could be drawn between the contents of a domain's DNS records and whether or not that domain would proxy fronted traffic. Over half of the domains tested showed a CloudFlare server address in their DNS records, but 5 of those still refused to proxy fronted traffic. 6 of the CloudFlare domains tested contained no mention of CloudFlare services in their DNS records, yet still successfully proxied the fronted traffic. One domain's records, huffingtonpost.com, contain the address of a completely different CDN (awsdns-hostmaster.amazon.com), yet still returned 3 viable fronting proxies.

VI. Conclusion

By building upon existing innovations in domain fronting this paper shows subdomain enumeration using Python brute force scripting as a viable methodology for identifying domain fronting proxies within a given Content Delivery Network. Despite formidable innovations in the identification and prevention of domain fronting traffic, penetration testers are still able to identify and use domain fronting proxies within many of today's popular CDNs. Per their announcement, Microsoft seeks to join the likes of Google and Amazon in blocking domain fronted traffic, but they are, as of yet, not entirely successful. The domainfuzzer.py application achieves an

additional goal of improving access to domain fronting capabilities for non-technical users wishing to conceal their web traffic from internet censorship. The cloud networking fundamentals on which this methodology is based allow it to be easily adapted for use on most CDNs.

At a 2020 DEF CON, Hunstad [12] presented "domain hiding" as a logical evolution to domain fronting by using TLS v1.3's ESNI header to conceal the forbidden address. Amazon and Google have since made efforts to thwart this functionality, but more research is needed to determine if it could be leveraged within domainfuzzer.py to identify viable domain hiding proxies in those CDNs. Alternatively, CloudFlare offers a bulk redirector function for Pages currently in Open Beta. More research is needed to determine if turning domainfronter.pages.dev into a redirector, could provide a DIY version of the MEEK pluggable API [9], using the hidden destination to proxy traffic outside of the CloudFlare CDN.

As history has confirmed, we can also expect more impressive innovations in censorship technology. These could include more finely tuned neural networks that utilize advanced "deep fingerprinting" to identify domain fronted traffic [18] or improved security postures [18] of companies as cyber security gains awareness globally. While subdomain enumeration is currently a solution for identifying domain fronting proxies, it will eventually become obsolete and need to be upgraded or replaced by further innovation.

Acknowledgment

We would like to thank Dr. Augusto Casas of the University of Maryland Baltimore County for some preliminary input and reviews of this paper.

References

[1] *32 Companies That Use Cloudflare [2022]*. (2022, May 21). Retrieved October 22, 2022, from https://www.starterstory.com/tools/cloudflare/companies-using

[2] BitQuark - *bitquark/dnspop: Analysis of DNS records to find popular trends*. (Mar 10, 2016). GitHub. Retrieved October 18, 2022, from https://github.com/bitquark/dnspop

[3] Borosh, S. (2018, April 12). *SSL Domain Fronting 101 - rvrsh3ll*. Medium. https://medium.com/rvrsh3ll/ssl-domain-fronting-101-4348d410c56f

[4] Brandom, R. (2018, April 18). *A Google update just created a big problem for anti-censorship tools*. The Verge. https://www.theverge.com/2018/4/18/17253784/google-domain-fronting-discontinued-signal-tor-vpn

[5] *CDN Finder - CDN Planet*. (n.d.). Retrieved December 13, 2022, from https://www.cdnplanet.com/tools/cdnfinder/

[6] Dean, B. (2021, July 2). *Cloudflare Stats for 2022: How Many Websites Use Cloudflare?* https://backlinko.com/cloudflare-users#cloudfare-users

[7] *Dig Command Manual*. (n.d.). DigGui.com. Retrieved October 7, 2022, from https://www.diggui.com/dig-command-manual.php

[8] Ergun, O. (2021, January 20). *What is CDN - Content Delivery Networks*. OrphanErgun.net. https://orhanergun.net/what-is-cdn-content-delivery-networks

[9] Fifield, D., Lan, C., Hynes, R., Wegmann, P., & Paxson, V. (2015, May 15). *Blocking-resistant communication through domain fronting*. https://www.petsymposium.org/2015/papers/03_Fifield.pdf

[10] *Generally available: Block domain fronting behavior on newly create. . .* (n.d.). Microsoft Azure. https://azure.microsoft.com/en-us/updates/generally-available-block-domain-fronting-behavior-on-newly-created-customer-resources/

[11] Guan, Z., Gou, G., Guan, Y., & Wang, B. (Nov. 12-14, 2019). *An Empirical Analysis of Plugin-based Tor Traffic over SSH Tunnel*. MILCOM 2019 - 2019 IEEE Military Communications Conference. Retrieved October 7, 2022, from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9020938

[12] Hunstad, E. [DEFCONConference]. (2020, August 5). *DEF CON Safe Mode - Domain Fronting is Dead, Long Live Domain Fronting Using TLS 1.3* [Video]. YouTube. https://www.youtube.com/watch?v=TDg092qe50g

[13] Li, Z., Wang, M., Wang, X., Shi, J., Zou, K., & Su, M. (2019). *Identification Domain Fronting Traffic for Revealing Obfuscated C2 Communications*. IEEE Explore, 2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9750455

[14] Mudge, R. [Raphael Mudge] (2017, February 3). *Domain Fronting and High-trust Redirectors with Cobalt Strike*. YouTube. Retrieved October 7, 2022, from https://www.youtube.com/watch?v=IKO1ovl7Ky4

[15] Muradova, G., & Heymatyar, M. (Oct 23-25, 2019). *Securing and Hiding the Destination of Confidential Medical Information with Domain Fronting*. 2019 IEEE 13th International Conference on Application of Information and Communication Technologies (AICT). Retrieved October 7, 2022, from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8981756

[16] *Pluggable Transports-Meek*. (n.d.). Tor Project. Retrieved October 7, 2022, from https://gitweb.torproject.org/pluggable-transports/meek.git/tree/

[17] Sheffey, S. R., & Aderholdt, F. (2019). *Improving Meek With Adversarial Techniques*. Usenix. Retrieved October 7, 2022, from https://www.usenix.org/system/files/foci19-paper_sheffey.pdf

[18] Sirinam, P., Imani, M., Juarez, M., & Wright, M. (2018, October 15). *Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning*. CCS '18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. https://dl.acm.org/doi/pdf/10.1145/3243734.3243768

[19] *Using custom URLs by adding alternate domain names (CNAMEs) - Amazon CloudFront*. (n.d.). Retrieved October 7, 2022, from https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/CNAMEs.html

[20] Wang X., Chen Z., Li Z., Huang W., Wang M., Pan S., Shi J. (2022) *Identification of MEEK-Based TOR Hidden Service Access Using the Key Packet Sequence* https://www.iccs-meeting.org/archive/iccs2022/papers/133500548.pdf

[21] *Websites using CloudFlare • Hunter TechLookup*. (n.d.). Hunter. Retrieved October 22, 2022, from https://hunter.io/technologies/cloudflare

[22] *What is SNI (Server Name Indication)?* (2022). CloudFlare.com. Retrieved October 22, 2022, from https://www.cloudflare.com/learning/ssl/what-is-sni/

[23] Yiu, V. [Vincent]. (2017, February 4). *Abusing Domain Fronting on Amazon CloudFront [4KHD]* [Video]. Retrieved October 7, 2022, from https://www.youtube.com/watch?v=zSBnM2HcRTw

[24] Yiu, V. (February 2017). *Domain Fronting via. CloudFront Alternate* Domains. Retrieved October 22, 2022, from https://www.vincentyiu.com/red-team/domain-fronting/domain-fronting-via.-cloudfront-alternate-domains

[25] Mudge, R. (2019, November 12). *Red Team Ops with Cobalt Strike (2 of 9): Infrastructure* [Video]. YouTube. Retrieved October 20, 2022, from https://www.youtube.com/watch?v=5gwEMocFkc0

TABLE I. 29 KNOWN CLOUDFRONT HOSTED DOMAINS QUERIED, NONE OF THEM HAVE EXISTING CNAME RECORDS

```
$dig -f cfdomains.txt -t CNAME | grep "CNAME"
;\239\187\191discord.gg.          IN      CNAME
;sourceforge.net.                 IN      CNAME
;nginx.com.                       IN      CNAME
;bloomberg.com.                   IN      CNAME
;shopify.com.                     IN      CNAME
;medium.com.                      IN      CNAME
;vimeo.com.                       IN      CNAME
;nodejs.org.                      IN      CNAME
;huffingtonpost.com.              IN      CNAME
;xeroshoes.com.                   IN      CNAME
;rightmessage.com.                IN      CNAME
;snappa.com.                      IN      CNAME
;wheelysales.com.                 IN      CNAME
;counterweightcreative.co.        IN      CNAME
;bonify.de.                       IN      CNAME
;populum.com.                     IN      CNAME
;episode.ninja.                   IN      CNAME
;leavemealone.app.                IN      CNAME
;huntakiller.com.                 IN      CNAME
;uplink.com.                      IN      CNAME
;marketbeat.com.                  IN      CNAME
;scribemedia.com.                 IN      CNAME
;jobscan.co.                      IN      CNAME
;katiegoesplatinum.com.           IN      CNAME
;tickettailor.com.                IN      CNAME
;upwork.com.                      IN      CNAME
;breezymobility.com.au.           IN      CNAME
;keyholesoftware.com.             IN      CNAME
;codekitapp.com.                  IN      CNAME

$dig -f cfdomains.txt -t CNAME | grep "SOA"
discord.gg.               30      IN      SOA     c.ci-servers.org. dnsmaster.channelisles.net. 2023022244…
sourceforge.net.          30      IN      SOA     ns11.constellix.com. dns.constellix.com. 2015010283 43200…
nginx.com.                30      IN      SOA     ns.nginx.com. hostmaster.nginx.com. 20220460 28800 7200…
bloomberg.com.            30      IN      SOA     pdns1.ultradns.net. dnsmaster.bloomberg.com. 2013076992…
shopify.com.              30      IN      SOA     ns1.dnsimple.com. admin.dnsimple.com. 1477237709 86400 7200…
medium.com.               30      IN      SOA     alina.ns.cloudflare.com. dns.cloudflare.com. 2302529701…
vimeo.com.                30      IN      SOA     ns-70.awsdns-08.com. awsdns-hostmaster.amazon.com. 20140201…
nodejs.org.               30      IN      SOA     meera.ns.cloudflare.com. dns.cloudflare.com. 2301190932…
huffingtonpost.com.       30      IN      SOA     huffingtonpost.com. awsdns-hostmaster.amazon.com. 1 28800…
xeroshoes.com.            30      IN      SOA     damon.ns.cloudflare.com. dns.cloudflare.com. 2300071886…
rightmessage.com.         30      IN      SOA     bayan.ns.cloudflare.com. dns.cloudflare.com. 2291500221 100…
snappa.com.               30      IN      SOA     dawn.ns.cloudflare.com. dns.cloudflare.com. 2278239029 1000…
wheelysales.com.          30      IN      SOA     amanda.ns.cloudflare.com. dns.cloudflare.com. 2299983011 10…
counterweightcreative.co. 30      IN      SOA     joaquin.ns.cloudflare.com. dns.cloudflare.com. 2300148293…
bonify.de.                30      IN      SOA     ned.ns.cloudflare.com. dns.cloudflare.com. 2302503105 10000…
populum.com.              30      IN      SOA     connie.ns.cloudflare.com. dns.cloudflare.com. 2280072249…
episode.ninja.            30      IN      SOA     donna.ns.cloudflare.com. dns.cloudflare.com. 2280032398…
leavemealone.app.         30      IN      SOA     ivan.ns.cloudflare.com. dns.cloudflare.com. 2300292581 1000…
huntakiller.com.          30      IN      SOA     ken.ns.cloudflare.com. dns.cloudflare.com. 2300765135 10000…
uplink.com.               30      IN      SOA     pdns03.domaincontrol.com. dns.jomax.net. 2023020702 28800…
marketbeat.com.           30      IN      SOA     isla.ns.cloudflare.com. dns.cloudflare.com. 2302413936 1000…
scribemedia.com.          30      IN      SOA     derek.ns.cloudflare.com. dns.cloudflare.com. 2300157391 100…
jobscan.co.               30      IN      SOA     candy.ns.cloudflare.com. dns.cloudflare.com. 2301881876…
katiegoesplatinum.com.    30      IN      SOA     amy.ns.cloudflare.com. dns.cloudflare.com. 2280541693 10000…
tickettailor.com.         30      IN      SOA     alex.ns.cloudflare.com. dns.cloudflare.com. 2300665933 1000…
upwork.com.               30      IN      SOA     fay.ns.cloudflare.com. dns.cloudflare.com. 2302549985 10000…
breezymobility.com.au.    30      IN      SOA     q.au. noc.afilias-nst.info. 1534290277 10800 3600 2764800…
keyholesoftware.com.      30      IN      SOA     fred.ns.cloudflare.com. dns.cloudflare.com. 2302506017 1000…
codekitapp.com.           30      IN      SOA     andy.ns.cloudflare.com. dns.cloudflare.com. 2280028566 1000…
```

TABLE II. MODERN DNS CONFIGURATIONS VARY SO WIDELY FROM TARGET TO TARGET THAT THIS IS NO LONGER THE CASE

```
#wget -q -O - -U demo http://discord.gg/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://sourceforge.net/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://nginx.com/df.txt --header "Host: domainfronter.pages.dev"
#wget -q -O - -U demo http://bloomberg.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://shopify.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://medium.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://vimeo.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://nodejs.org/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://huffingtonpost.com/df.txt --header "Host: domainfronter.pages.dev"
#wget -q -O - -U demo http://xeroshoes.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://rightmessage.com/df.txt --header "Host: domainfronter.pages.dev"
#wget -q -O - -U demo http://snappa.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://wheelysales.com/df.txt --header "Host: domainfronter.pages.dev"
#wget -q -O - -U demo http://counterweightcreative.co/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://bonify.de/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://populum.com/df.txt --header "Host: domainfronter.pages.dev"
#wget -q -O - -U demo http://episode.ninja/df.txt --header "Host: domainfronter.pages.dev"
#wget -q -O - -U demo http://leavemealone.app/df.txt --header "Host: domainfronter.pages.dev"
#wget -q -O - -U demo http://huntakiller.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://skyalliance-va.com/df.txt --header "Host: domainfronter.pages.dev"
#wget -q -O - -U demo http://marketbeat.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://scribemedia.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://jobscan.co/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://katiegoesplatinum.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://tickettailor.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://upwork.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://breezymobility.com.au/df.txt --header "Host: domainfronter.pages.dev"
#wget -q -O - -U demo http://keyholesoftware.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
#wget -q -O - -U demo http://codekitapp.com/df.txt --header "Host: domainfronter.pages.dev"
domain fronting works!
```

TABLE III. CLOUNDFLARE DOMAINS

CloudFlare Domain	CloudFlare address found in DNS records	Top level domain fronts traffic	# of Viable Subdomains
medium.com	Y	Y	996
sourceforge.net	N	Y	970
shopify.com	N	Y	43
upwork.com	Y	Y	21
bloomberg.com	N	Y	19
counterweightcreative.co	Y	Y	12
vimeo.com	N	Y	10
bonify.de	Y	Y	10
jobscan.co	Y	Y	9
xeroshoes.com	Y	Y	9
tickettailor.com	Y	Y	6
huntakiller.com	Y	Y	5
katiegoesplatinum.com	Y	Y	4
rightmessage.com	Y	N	4
snappa.com	Y	Y	4
scribemedia.com	Y	Y	4
marketbeat.com	Y	Y	3
leavemealone.app	Y	N	3
wheelysales.com	Y	N	3
huffingtonpost.com	N	N	3
nginx.com	N	N	3
nodejs.org	Y	Y	2
episode.ninja	Y	N	2
discord.gg	N	Y	2
uplink.com	N	Y	2
codekitapp.com	Y	Y	1
populum.com	Y	N	1
keyholesoftware.com	Y	Y	0
breezemobility.com.au	N	N	0
skyalliance-va.com	N	N	0

TABLE IV. TABLE TYPE STYLES

Microsoft Azure Domain	Azure Cloud address found in DNS records	Top level domain fronts traffic	# of Viable Subdomains
microsoft.com	Y	N	6
markmonitor.com	N	N	3
xbox.com	Y	N	2
azure.com	Y	N	1
onedrive.com	Y	N	1
skype.com	Y	N	1
outlook.com	Y	N	0
live.com	Y	N	0
hotmail.com	Y	N	0
bing.com	N	N	0
bingbar.com	Y	N	0
internetexplorer.com	Y	N	0
sqlserver.net	N	N	0
visualstudio.com	Y	N	0
xbox360.com	Y	N	0
xboxone.com	Y	N	0
zune.com	Y	N	0

TABLE V. TABLE TYPE STYLES

Commonly Viable Subdomains	Count
www	17
api	10
cdn	8
status	7
help	7
app	7
blog	7
dev	6
support	5
search	5
careers	5
mail	5
ftp	5
go	5
admin	5