CCN-CERT

BP/10

Security recommendations for CDNs

GOOD PRACTICES REPORT

FEBRUARY 2022

ccn-cert centro criptológico nacional

CCN centro criptológico nacional

Edited by:

GOBIERNO DE ESPAÑA MINISTERIO DE DEFENSA

CCN centro criptológico nacional

Pº de la Castellana 109, 28046 Madrid

Release date: February 2022

Index

1. About CCN-CERT

The CCN-CERT (www.ccn-cert.cni.es) is the Computer Emergency Response Team of the National Cryptologic Centre, CCN (www.ccn.cni.es). This service was created in 2006 as the Spanish **Governmental/ National CERT** and its functions are set out in Law 11/2002 regulating the National Intelligence Centre, RD 421/2004 regulating the CCN and RD 3/2010, of 8 January, regulating the National Security Scheme (ENS), modified by RD 951/2015, of 23 October.

According to all of them, the CCN-CERT is responsible for managing cyber-incidents that affect **public sector systems, companies and organisations of strategic interest** to the country and any classified system. Its mission, therefore, is to contribute to the improvement of Spanish cybersecurity, being the national alert and response centre that cooperates and helps to respond quickly and efficiently to cyber-attacks and to actively face cyber-threats.

CCN-CERT is the Information Security Incident Response Capability of the National Cryptologic Centre, CCN.

2. Introduction to CDNs

Content delivery networks (CDNs) are highly valuable elements, responsible for the delivery of various content with which the user interacts on a daily basis. This type of architecture arose mainly to solve the problem of latency, understood as the delay that occurs between the moment a website is requested and the moment the content is delivered and displayed on the screen.

This process is influenced by a number of factors, many of them specific to the type of content, server and website requested, the most important of which is the physical distance between the user and the server hosting the content.

Such architectures were developed to solve the latency problem.

2. Introduction to CDNs

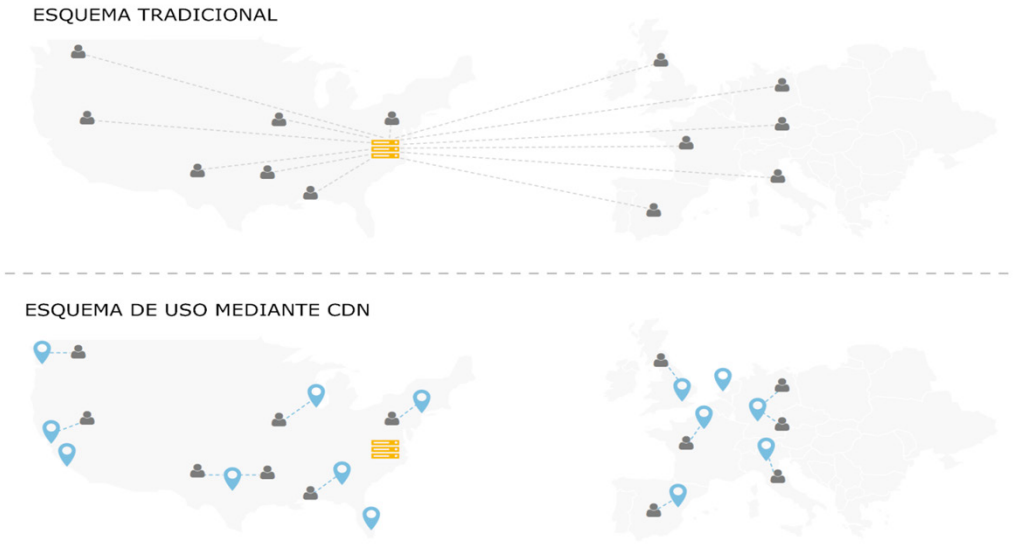


Figure 1 - Diagram of how a CDN network works.

The main mission of a CDN is to virtually reduce this physical distance, with the aim of improving speed and performance. To do this, a CDN caches a version of the content to be served in multiple geographic locations, known as *Points of Presence* (PoP). Each of these PoPs contains a series of storage servers responsible for delivering content to visitors who are geographically close.

In addition to this optimisation in terms of latency, a CDN provides a number of other advantages:

- Increases the loading speed of a website.
- Blocks bots, *spammers* and other harmful tools.
- Reduces bandwidth consumption.
- Allows load balancing between different servers.
- Protects websites from distributed denial of service (DDoS) attacks.
- It increases the level of security through different rules and protection mechanisms.

2. Introduction to CDNs

For these mechanisms to be applicable, as a general rule, it will be necessary to modify the DNS root server of the domain (e.g. mydomain.com). In essence, the primary A record of the DNS will need to be modified to point to a specific IP address in the CDN range, although there are also providers that allow implementation at the CNAME level without the need to modify the root DNS server. In both cases, the end user will be redirected to the CDN network instead of the origin server.

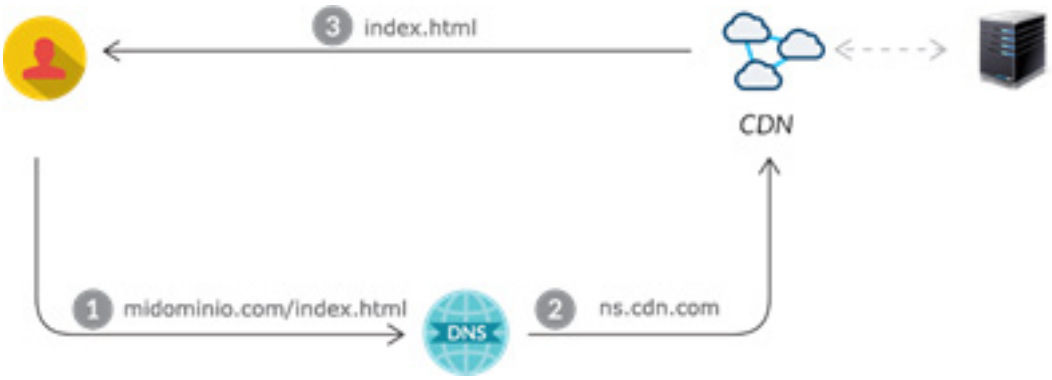


Figure 2 - User interaction with the CDN network

There are a large number of CDN service providers, with both free and paid versions. This guide will provide generic recommendations for any of them.

3. Overall comparison of the main CDN providers

	CLOUDFLARE	Incapsula	Akamai
POPS (points of presence)	120	42	2200
Minimum contract time (months)	1	1	12
Online control panel	✓	✓	✓
Online registration	✓	✓	✗
Cname support	✓	✓	✓
Spydy support	✗	✗	✓
HTTP/2 support	✓	✓	✓
IPV6 support	✓	✓	✓
GZIP compression	✓	✓	✓
DDOS protection	✓	✓	✓
WAF	✓	✓	✓
IP access control	✓	✓	✓

3. Overall comparison of the main CDN providers

	CLOUDFLARE	Incapsula	Akamai
Own SSL certificate	✓	✓	✓
API use	✓	✓	✓
Access to logs in RAW format	✓	✗	✓
Video on demand (VOD)	✗	✗	✓
Storage options	✗	✗	✓
Real-time statistics	✓	✓	✓

4. Techniques for protecting CDNS against DDOS attacks

A distributed denial of service (DDoS) attack is a type of attack that aims to make an online service unavailable, usually by temporarily disrupting or suspending the services provided by the server. Attacks originate from compromised devices, whether personal computers, routers or IoT devices, often distributed globally in what is known as a botnet.

These attacks differ from a conventional denial-of-service (DoS) attack, as DoS attacks would only use a single device connected to the Internet (a network connection) to flood a target with malicious traffic.

A DDoS attack is a type of attack aimed at making an online service unavailable, usually by interrupting or suspending the services provided by the server.

4. Techniques for protecting CDNS against DDOS attacks

Once the attacker has control of a *botnet*, he can control the machines by sending updated instructions to each *bot via* a remote control panel. When the *botnet targets* a victim's IP address, each *bot* will respond by sending requests to the target, which could cause the target server or network to exceed its capacity to respond, leading to a degradation of service, which could even result in an indefinite outage.

Because each *bot* is a legitimate Internet device, separating harmful traffic from legitimate traffic is not an easy task.

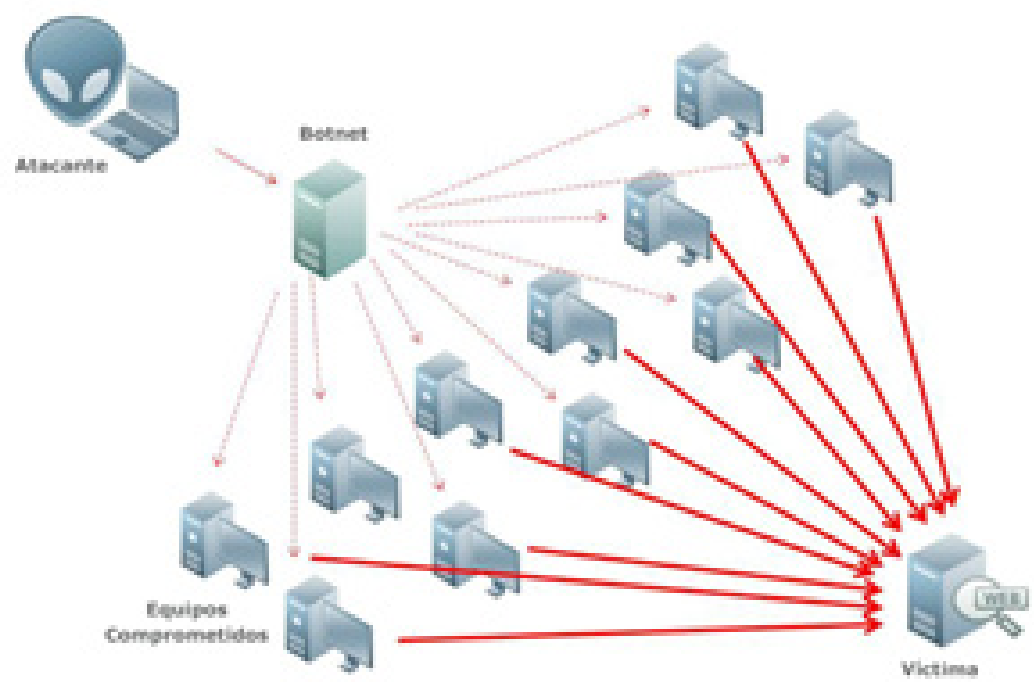


Figure 3 - Diagram of how a botnet works.

4. Techniques for protecting CDNS against DDOS attacks

Broadly speaking, DoS and DDoS attacks can be divided into three (3) types:

- **Volumetric attacks:** includes UDP floods, ICMP floods and other floods caused by artificially generated packets. The objective of the attack is to saturate the bandwidth of the attacked site, and is measured in bits per second (bps).
- **Protocol-level attacks:** includes SYN floods, fragmented packet attacks, etc. This type of attack consumes real server resources or intermediate communication equipment such as firewalls and load balancers, and is measured in packets per second (pps).
- **Layer 7 attacks (at the application layer):** includes attacks such as slowloris, GET/POST floods and attacks targeting Apache server vulnerabilities among others. Generally, seemingly legitimate requests are used and the aim of these attacks is to crash the web server. They are measured in requests per second (rps).

Attackers are mainly motivated by:

- **Ideology:** So-called "hacktivists" use DDoS attacks as a means of attacking websites with which they disagree ideologically.
- **Business conflicts:** Companies can use DDoS attacks to strategically take down competitors' websites, e.g. to prevent them from participating in a significant event (e.g. Black Friday).
- **Vandalism:** script-kiddies use publicly available tools on the Internet to launch DDoS attacks without a clear motivation or real intention to make a profit in return.
- **Extortion:** Attackers use DDoS attacks or the threat of DDoS attacks as a means to extort money from their victims.
- **Digital warfare:** government-sanctioned DDoS attacks can be used against the infrastructure of an enemy country.

Mitigating a DDoS attack requires a strategy depending on the type of attack, and whether it is one or several types at the same time. Generally speaking, the more complex the attack, the more difficult it is to identify harmful traffic, as the attacker's goal is to make legitimate traffic appear legitimate by making mitigation as inefficient as possible.

For this reason, **practically all CDN services offer services to mitigate this type of attack.** Depending on the security level that the user has selected, usually **Low, Medium** or **High Attack,** the CDN will analyse and filter the connections, implementing a series of additional security measures to filter traffic.

4.1 Javascript challenges

It is a type of challenge that is used to filter out attackers, who use automated tools, from legitimate clients. The challenge is based on sending each client, attacker or legitimate user, a *JavaScript* code that includes some kind of challenge. Virtually any browser today has a *JavaScript* engine and will easily understand and resolve the challenge transparently (without user interaction), while automated DDoS tools are usually not equipped with such *JavaScript* functionalities and therefore will not be able to overcome the challenge and establish the connection to the end server.

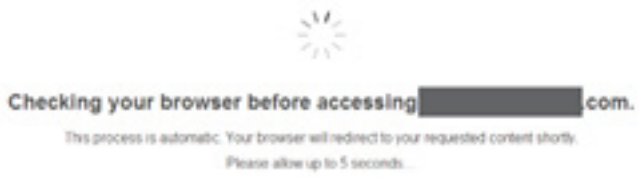


Figura 4 - Desafío JavaScript proporcionado por CloudFlare

4.2 Testing on TCP SYN packets

With this method, the aim is to check that the client's TCP stack is valid and correctly implemented, looking for a correct response to certain packets constructed under unusual conditions, allowing to detect packets from spoofed source IP addresses and *socket-generated* packets using a DDoS tool.

Common tactics range from returning an RST packet on the first SYN received (expecting the client to resend it) to deliberately sending a SYN-ACK with an incorrect sequence number expecting the client to return an RST and then try again.

The simplest way to respond to this type of test is to allow the operating system to respond to these packets, making it effective when it is a legitimate connection.

There are mainly two (2) techniques:

a. **TCP Reset:** The CDN will send a packet with the RESET (RST) *flag* active to reset established TCP connections (those that have successfully completed the *handshake*). This is the most common method of verification, as the DDoS tools and bots designed do not have this logic implemented, unlike a connection through a real browser.

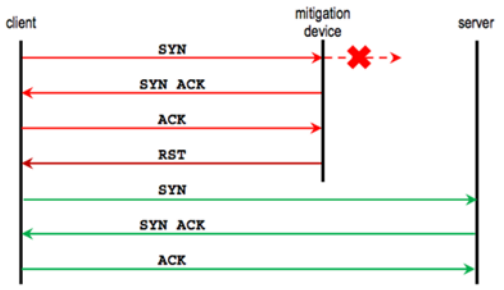


Figure 5 - Functioning of TCP Reset mitigation

b. **TCP out-of-sequence:** unlike the previous method, the CDN can deliberately challenge the client by sending SYN-ACK responses with an out-of-sequence number as seen in the figure below. Since the sequence number is incorrect, the client is supposed to re-establish the TCP connection and establish the connection again. Again, a *bot* or a DDoS tool would not resolve these cases.

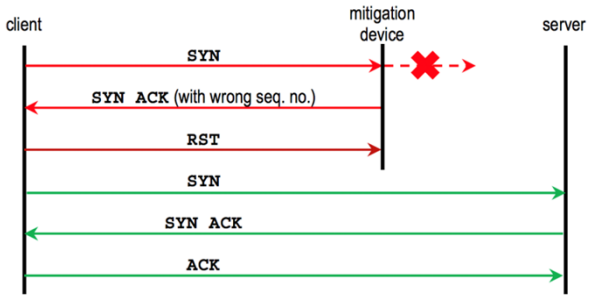


Figure 6 - Operation of TCP out-of-sequence mitigation

4.3 SSL Connection filtering

Today, due to increased security measures against conventional DDoS techniques, we are starting to see an increase in damaging floods of SSL connections.

These SSL floods bypass a large number of security devices, such as firewalls or intrusion protection systems (IPS). In addition, there are current variants such as SSL renegotiation attacks, where unlike traditional attacks, the attacker needs only one tenth of the computational power of the unprotected server to take over its resources and disrupt legitimate traffic to them.

A CDN provider simply removes empty or bad SSL connections, protecting the resources behind them.

4.4 302 HTTP Redirect

The basic idea is that a legitimate browser will respect HTTP 302 redirects. For this reason, dynamically generated redirects are tested to ensure that the visitor is a legitimate user, who can interpret these actions through the browser.

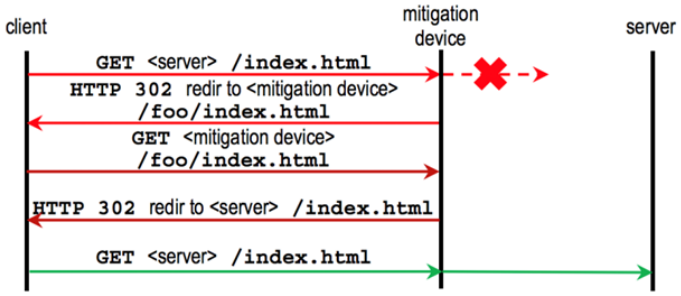


Figure 7 - Operation of HTTP redirection mitigation

4.5 HTTP Cookies

This technique is generally linked to the previous one and allows malicious traffic to be identified by dynamically injecting a cookie into the connection between the supposedly legitimate client and the CDN. Traffic that is not able to interpret this new situation, because it is likely to be malicious traffic, will be discarded.

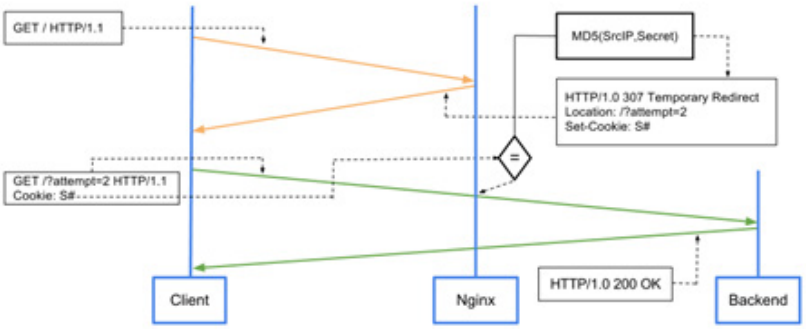


Figure 8 - Functioning of HTTP cookie mitigation in an Nginx module

4.6 CAPTCHA

A CAPTCHA *(Completely Automated Public Turing test to tell Computers and Humans Apart)* is a challenge-response test used to determine whether or not the user is human.

This technique is probably the most widespread, simplest and safest to use as it involves direct human intervention. If the client succeeds in resolving it, it will be kept on the whitelist for a certain time or for a certain amount of traffic, after which it will have to be re-authenticated.

Figure 9 - Example of how a CAPTCHA-based challenge-response system works.

This method is, in itself, quite intrusive and in practice is generally used when configuring the CDN in *Under Attack* mode.

5. Safety recommendations for the use of CDN

5.1 Configure SSL/TLS on the connection between the user and CDN

SSL/TLS communication between the CDN and the origin server should be used as much as possible. *Transport Layer Security* (TLS) is a protocol for encrypting data sent over the Internet, which evolved from *Secure Sockets Layer* (SSL), in order to correct most of the security flaws of this protocol (the industry still uses these terms interchangeably for historical reasons).

In this way TLS is designed to provide:

Authentication

Encryption

Integrity

In order to enable TLS over the configuration, the website will need an SSL certificate and its corresponding private key. There are a number of tools, as well as a simple evaluation methodology, that allow administrators to evaluate the SSL server configuration where factors such as certificate validity and trust, protocol support, key exchange and encryption support are taken into account.

Combining the scores from these tests, with a score between 0 and 100, gives an overall score which is converted into a letter, from A (highest score) to F (lowest score).

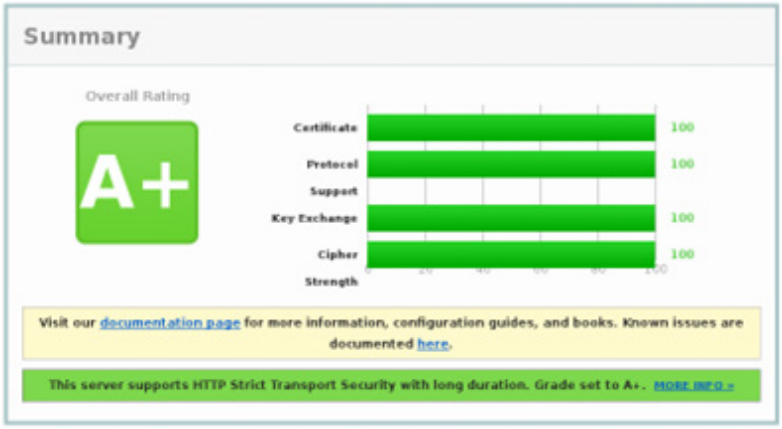


Figure 10 - Example of the Qualys SSL Server Test scoring tool

Enabling SSL/TLS on the CDN configuration has the benefit of providing security to visitors using a self-generated certificate.

Because visitors connect only to the CDN, a less secure certificate (e.g., rated C) that is in use between the origin server and the CDN will not affect this experience, as the one offered by the CDN will probably score higher, without the need to modify the configuration on the origin server.

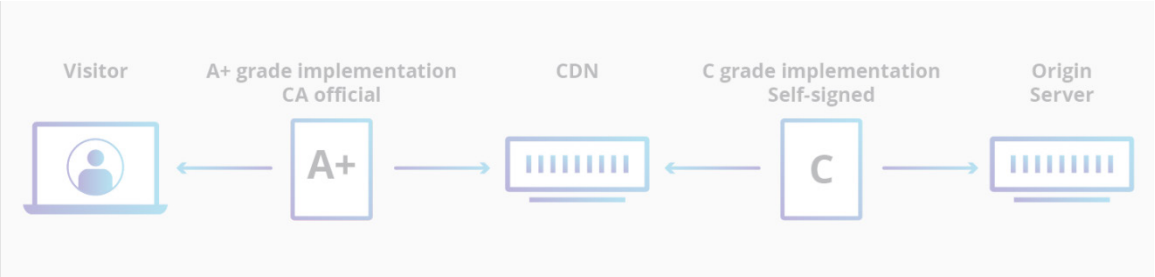


Figure 11 - Implementation and securitisation of CDN certificate

5.2 Configuration for services under HTTP connection

There are some cases where the origin server is not configured to offer secure connections under SSL, either because of incorrect administration or because it is a service that cannot be securely configured.

In these cases it will be necessary to configure the CDN connection over SSL for the traffic between the CDN and the visitors, while the connection between the CDN and the website is maintained over *http* without the need for any additional configuration on the origin web server.

In addition, we must take into account that the lack of SSL implementation can have negative results in other aspects, such as in the positioning of some search engines like Google.

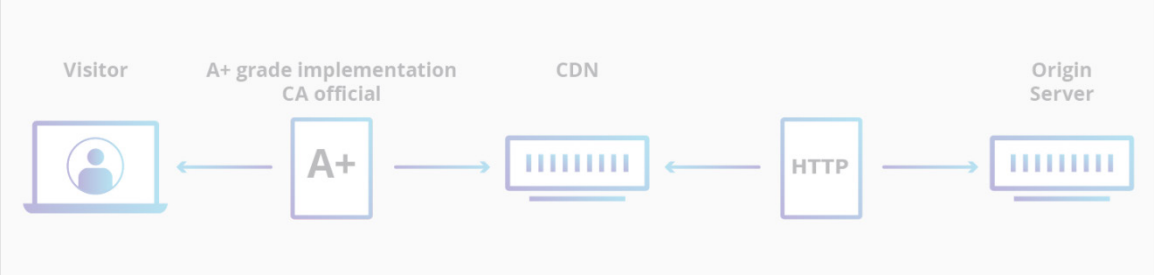


Figure 12 - SSL implementation on http servers

In addition, the CDN itself will choose the most secure encryption algorithms, regardless of the configuration of the website, as well as implement other options such as *Forward Secrecy.*

5.3 Activate the use of HSTS

HSTS *(HTTP Strict Transport Security)* is a web security policy technology, designed to help protect HTTPS servers against degradation attacks.

This type of degradation attacks (known as *SSL stripping attacks*) allow for MiTM (*Man-in-the-middle*) attacks where a would-be attacker could redirect the browser from a correctly configured HTTPS web server to his own server via an HTTP channel. Once this redirection has been achieved, the exchanged data, such as cookies, can be compromised.

Browsers that support HSTS include:

Google Chrome as of version 4.0.211.0.

Google Chrome for Android since version 18.

Firefox and Firefox Mobile since version 4.

Opera since version 12.

Safari since version 7.

Android Browser since Android version 4.4.

Internet Explorer plans to implement it in version 12 of its browser.

Most CDNs allow you to configure and activate this option by adding headers where you can define the duration of this policy, include different subdomains, etc.

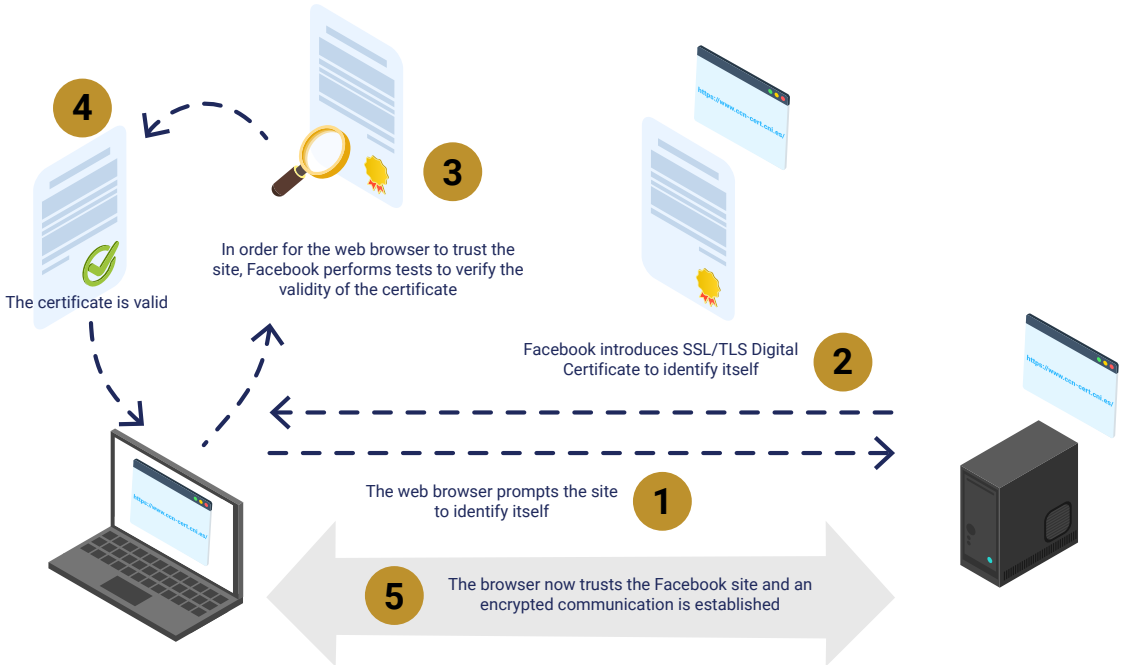
5.4 Use of customer certificate

TLS (the modern version of SSL) allows a client to verify the identity of the server to which it is making a connection. Normally, a TLS *handshake* is one-way, i.e. the client can verify the identity of the server, but the server cannot verify the identity of the client. In a client-authenticated TLS *handshake*, both sides provide a certificate to verify their identity.

If the origin server is configured to accept requests using a valid client certificate from the CDN, traffic that does not pass through the CDN's network will be discarded because it does not have the correct certificate.

This means that attackers cannot bypass CDN features such as the WAF, connection limit rules or protection against DDoS attacks. Even if a would-be attacker were to spoof the source IP address by spoofing any IP address in the CDN pool, they would not be able to establish a connection to the end server.

Therefore, implementing and configuring the CDN provider's client certificate will add a new level of security to the web service, as well as prevent potential data leaks and eliminate much of the risk of mass scanning tools.



5.5 Change the original IP address associated with the server

If the website was not initially configured with the CDN, there may have been a period where the DNS pointed directly to the source IP address. With the use of tools available online it is possible to easily query the history of DNS records, which can reveal IP addresses in use before the CDN was activated and which may still be in use for the same purpose.

This risk cannot be mitigated directly, as the information available is historical. However, a new IP address can always be requested from a different range than the one that would have been maintained during the implementation of the CDN.

Below is an example of one such tool showing the historical information it maintains.

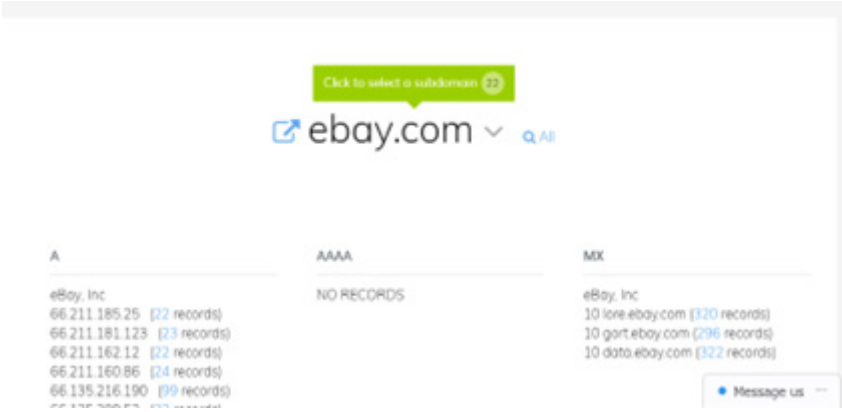


Figure 13 - Obtaining historical information on a domain

5.6 Allow only access to the CDN IP address pool

One of the main tasks to be performed when implementing a CDN is to whitelist all IP addresses specified by the provider to allow connections, blocking any other external requests.

This must be done at both IPv4 and IPv6 level, if used, and in the case of Linux, *iptables* can be used to do this in a simple way. For example, to allow the connection from IP address 1.1.1.1 you can execute the following command:

```
$ iptables -I INPUT -p tcp -m multiport --dports http,https -s “1.1.1.1”
-j ACCEPT
```

A final rule will then be added to block any connection attempts from IP addresses that have not been specified with the previous command:

```
$ iptables -I INPUT -p tcp -m multiport --dports http,https -j DROP
```

It should be noted that, depending on the operating system used, it will also be necessary to save this table in order to retrieve it in case of a system reboot. In the case of Linux systems it is possible to save the state of the *iptables* rules with the following commands:

- **Debian/Ubuntu:** iptables-save > /etc/iptables/rules.v4
- **RHEL/CentOS:** iptables-save > /etc/sysconfig/iptables
- **Debian/Ubuntu:** ip6tables-save > /etc/iptables/rules.v6
- **RHEL/CentOS:** ip6tables-save > /etc/sysconfig/ip6tables

5.7 Protect against brute force attacks and connection limits

Another useful feature of CDNs is that they allow the administrator to set connection limits to protect the website against denial-of-service attacks, brute-force login attempts against an access or administration panel, and other abusive behaviour directed at the application layer.

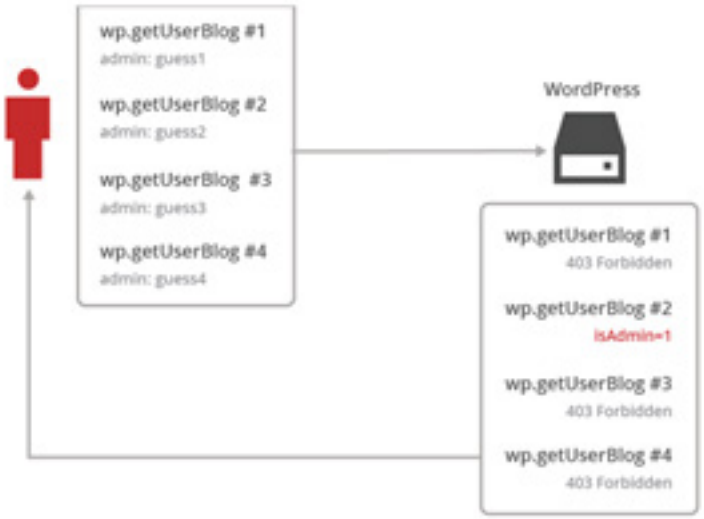


Figure 14 - Functioning of a brute force attack

Generally, login modules are performed through POST requests where a username and password are sent to the web server. During the preparation of a brute-force attack, an attacker captures these login requests and uses them to generate an automated attack, launching requests consecutively, at the highest possible rate, in order to obtain a valid username and password combination that will allow access.

For example, the system can be configured to prevent a possible attacker from carrying out a brute force attack against the administration panel of a Wordpress site, generating a block when a certain number of failed attempts is reached or generically when a 401 (unauthorised access) or 403 (forbidden access) error is detected, detecting a harmful *crawler* or *bot* and generating an action when, for example, it accesses pages that are not found (404 codes), etc.

This limitation, more generally, allows setting time thresholds, defining the type of response on specific URLs of the website, even reducing the bandwidth used and protecting the end server.

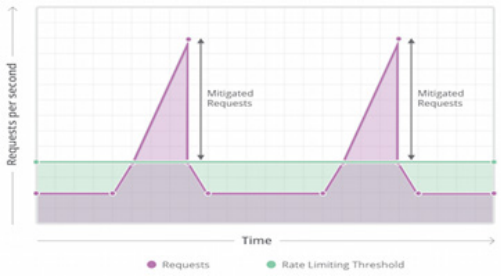


Figure 15 - Operation of Rate Limiting

5.8 Check the configuration of the DNS records

None of the DNS records should contain any mention of the source IP address. The following DNS records should be checked thoroughly to ensure that they do not contain any information about the source:

SPF: SPF (the acronym for Sender Policy Framework) is a protection against address spoofing when sending emails. It identifies authorised SMTP mail servers for the transport of messages via DNS records.

TXT: A TXT record is a type of DNS record that contains text information from sources external to a domain and is added to the domain configuration.

In addition, special attention should be paid to the configuration of the rest of the hosted subdomains, as some of them could still be configured directly to the IP address of the origin server, exposing it to attackers and negating the CDN's mitigation against different attacks.

5.9 Host mail on a different server

If the mail server is hosted on the same IP address as the originating web server, an attacker could find the IP address in an outgoing email.

For example, an attacker could send an email to a non-existent address to generate a bounce, where the source IP address of the server could be exposed in the headers (the bounce may contain the IP address of your server in one of its fields). There are even cases where it is not necessary to send an email if the platform, for example, has a user

registration system or to obtain the password in case the user has forgotten it.

In addition, MX records always show the actual mail server. Therefore, if the mail of a domain goes through the same web server, you can easily find the IP address through the MX records.

The following image shows an example, where the real IP address of the server (52.x.x.x) has been exposed through the use of e-mail:

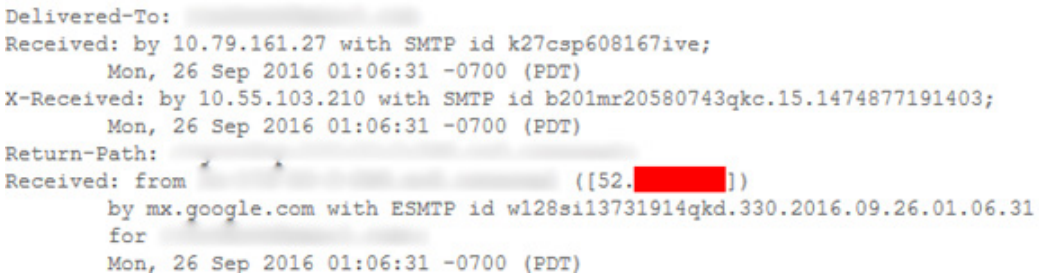


Figure 16 - Source IP discovery through mail service

Another option to avoid this type of information leakage would be to use the service through a third party to send the mails on behalf of one's own domain.

5.10 Disable dynamic file inclusion

A large number of web services offer the user the option to set, for example, a personalised image as the users' avatar. In most cases, they offer the possibility not only to directly upload an image to the server, but also to provide a remote URL from which to retrieve the image.

If the server downloads this file, this opens the option for a new attack vector, allowing a potential attacker to obtain the real IP address easily.

With a few lines of PHP and a correctly chosen *.htaccess* file, it is possible to create a link to a resource (e.g. an image) that logs the IP address of anyone trying to retrieve it:

```
# Content of .htaccess
AddHandler application/x-httpd-php5 .jpg

# Example file file.php
    <?php

    $fh =@fopen("log.txt", "a+");
    $timestamp = date('l jS \of F Y h:i:s A');
    $hostname = @gethostbyaddr($_SERVER['REMOTE_ADDR']);

    @fwrite($fh, "\r\n$timestamp\r\n");
    @fwrite($fh, 'REMOTE_ADDR: '.$_SERVER['REMOTE_ADDR']."\r\n");
    @fwrite($fh, 'Host Name: '."$hostname\r\n");
    @fwrite($fh, 'HTTP_CLIENT_IP: '.$_SERVER['HTTP_CLIENT_IP']."\r\n");
    @fwrite($fh, 'HTTP_USER_AGENT: '.$_SERVER['HTTP_USER_AGENT']."\r\n");

    fclose($fh);
    // bait.png is the image to show when grabbing the ip
    // give 755 permissions to bait.png
    $im = imagecreatefrompng("bait.png");
    header('Content-Type: image/jpeg');
    imagepng($im);
    imagedestroy($im);
    ?>
```

A potential attacker could use a number of other online services to achieve the same purpose, without the need for their own web server, with the same results and with an even greater level of anonymity.

5.11 Configure WAF and application-level protection

In some cases, attackers use DDoS attacks to weaken perimeter defences or block security devices. For this reason, many CDNs provide the possibility to use a WAF *(Web Application Firewall)* system, which will protect the client from a large number of attacks.

A web application firewall is a firewall that monitors web security and filters or blocks traffic to and from a web application. In this way, it can filter web application content, while a network firewall protects traffic only between servers.

As a general rule, a WAF will contain the rules that apply to the top 10 OWASP vulnerabilities:

- **Injection.**
- **Authentication and session management.**
- **Cross-Site Scripting (XSS).**
- **Insecure object references.**
- **Weak security settings.**
- **Leaking of confidential information.**
- **Lack of access control.**
- **Cross-site request forgery (CSRF).**

- **Use components with known vulnerabilities.**
- **Redirects and unvalidated submissions.**

Además, deberá tenerse especial atención en la configuración del resto de subdominios alojados, ya que alguno de estos podría encontrarse aún configurado de forma directa a la dirección IP del servidor de origen, exponiendo ésta a posibles atacantes y anulando la mitigación del CDN frente a diferentes ataques.

Figure 17 - CDN-based WAF protection system

5.12 Avoiding service search engines

There are a number of service and server lookup tools available online that can provide access to the source IP address information of the newly configured service.

One of the best known is **Shodan**, a search engine that allows the user through a variety of filters to find specific equipment (routers, servers, etc.) connected to the Internet. It can also be understood as a search engine for service banners, which are metadata that the server sends back to the client.

This information may contain versions of the server software, which options the service supports, a welcome message, or anything else that the client may know before interacting with the server and that can be used to locate the origin server behind a CDN service.

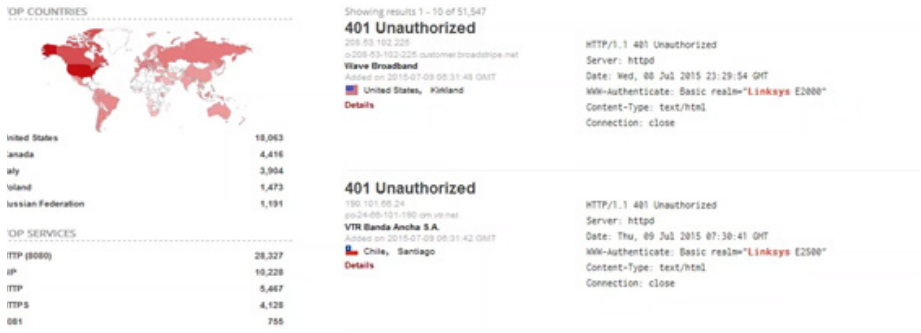


Figure 18 - Example of a search in Shodan

Another widely used service is **Censys**, which collects data on servers, services and websites on a daily basis, scanning the entire spectrum of IPv4 addresses. For example, knowing that the domain moz.com is protected by a CDN, you can run a search on Censys to try to get more information and get the following:

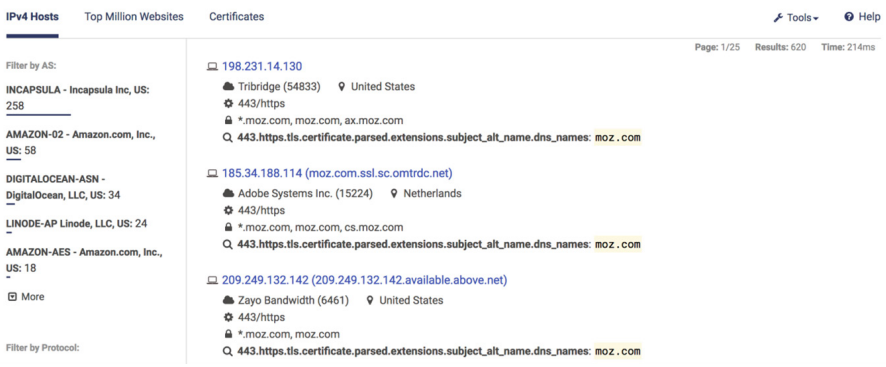


Figure 19 - Censys search of the domain moz.com

As can be seen in the screenshot, there are a large number of fields that can be used to perform these searches, such as CN (Common Name), SAN (Subject Alternative Name), the SHA256 computation of the certificate, the HTTP response code (in this case 200), etc. up to finding out the source IP address of the domain, which in this case belongs to the range 209.249.132.0/24.

6. Basic security decalogue

This Decalogue of Best Practices aims to lay the groundwork for security measures to be taken into account when migrating a service to a CDN provider.

Basic security decalogue

1. SSL/TLS configuration, including HSTS, on the connection between the end user and the CDN.
2. Use client certificates between the CDN and the origin server.
3. Modify the IP address of the origin server if it has been exposed before.
4. Allow only access to the origin server by an IP address from the CDN pool.
5. Protect the web service against brute force attacks and implement limiting the number of client connections.
6. Check the configuration of DNS records.
7. Migrate the e-mail service if necessary.
8. Disable, if any, dynamic file inclusion in the corresponding web service.
9. Have a WAF in place and implement security measures at application level.
10. Check that the origin server data is not found in Internet search engines such as Shodan or Censys.

References

- http://hopandfork.org/2016/11/16/cloudflare-ip-unveil.html
- https://www.defcon.org/images/defcon-21/dc-21-presentations/Mui-Lee/DEFCON-21-Miu-Lee-Kill-em-All-DDoS-Protection-Total-Annihilation-WP-Updated.pdf

centro criptológico nacional

centro criptológico nacional

www.ccn.cni.es

www.ccn-cert.cni.es

oc.ccn.cni.es