01net - Website analysis

Nicolas van de Walle
Dec 16, 2018

***Abstract*—This paper contains an analysis of the website 01net.com. It was conducted as part of the course of computer networks given by Pr. Olivier Bonaventure at *Universite Catholique de Louvain*.**

I. Introduction

01net.com is a french media website specialized in high-tech and new technologies. It is owned by the media group *NextRadioTV* [1] and is, according to Alexa.com [2], the 107th most visited website in France.

This paper will analyze this site regarding four different aspects:

- the Domain Name System (DNS) and the Nameservers
- the HyperText Transfer Protocol (HTTP)
- the Transport Layer Security (TLS)
- the Transmission Control Protocol (TCP)

In order to perform this analysis, I first decided to go on https://01net.com. The first thing I discovered is the fact that we are automatically redirected to https://www.01net.com. That's why, in this paper, I will focus on this www subdomain. *Note : The reasons of that redirection will be explained in this report*

II. DNS

When performing the `dig www.01net.com` command from the UCLouvain computer on Dec. 4, 2018, we get the following results (Fig. 1) which we will interpret in the subsections below.

```
;www.01net.com.                IN      A

;; ANSWER SECTION:
www.01net.com.          60      IN      CNAME   d5w2uqtmtwurf.cloudfront.net.
d5w2uqtmtwurf.cloudfront.net. 60 IN     A       54.230.129.77
d5w2uqtmtwurf.cloudfront.net. 60 IN     A       54.230.129.248
d5w2uqtmtwurf.cloudfront.net. 60 IN     A       54.230.129.181
d5w2uqtmtwurf.cloudfront.net. 60 IN     A       54.230.129.237

;; AUTHORITY SECTION:
d5w2uqtmtwurf.cloudfront.net. 1831 IN   NS      ns-568.awsdns-07.net.
d5w2uqtmtwurf.cloudfront.net. 1831 IN   NS      ns-1655.awsdns-14.co.uk.
d5w2uqtmtwurf.cloudfront.net. 1831 IN   NS      ns-6.awsdns-00.com.
d5w2uqtmtwurf.cloudfront.net. 1831 IN   NS      ns-1160.awsdns-17.org.

;; ADDITIONAL SECTION:
ns-1160.awsdns-17.org.  67415   IN      A       205.251.196.136
ns-1160.awsdns-17.org.  67415   IN      AAAA    2600:9000:5304:8800::1
ns-1655.awsdns-14.co.uk. 99339  IN      A       205.251.198.119
ns-1655.awsdns-14.co.uk. 99339  IN      AAAA    2600:9000:5306:7700::1
ns-568.awsdns-07.net.   62107   IN      A       205.251.194.56
ns-568.awsdns-07.net.   62107   IN      AAAA    2600:9000:5302:3800::1
ns-6.awsdns-00.com.     59701   IN      A       205.251.192.6
ns-6.awsdns-00.com.     59701   IN      AAAA    2600:9000:5300:600::1
```

Fig. 1. `dig www.01net.com` command result (UCLouvain - Dec. 4, 2018)

A. Aliases and Records

The first line of the answer shows us that when we request `www.01net.com`, an alias is used (`CNAME = d5w2uqtmtwurf.cloudfront.net.`) and this one points to several IPv4 adresses which all have the same `/24` IPv4 prefix (`54.230.129.[...]`).

When we try to get some more information about those IP's and prefix with the website otx.alienvault.com [3], we can see that the `54.230.0.0/16` prefix is owned by *AS16509 Amazon.com, Inc*. This is our first clue that `www.01net.com` is hosted by Amazon Web Services (AWS) [4]. The cloudfront.net domain used as CNAME also gives us that information as cloudfront, as said on the AWS official website [5], is a Content Delivery Network owned by Amazon.

This CDN also explains why we don't get the same IP addresses depending on where the request is coming from. At the same time, using the Google DNS resolver (`8.8.8.8`) [6], we get different results from Arlon and Louvain-la-Neuve (`54.192.13.120` $\neq$ `54.230.95.250` for example).

B. Nameservers

The nameservers used by `www.01net.com` are also a clue that 01net uses Amazon Web Services. In fact, the cloudfront CNAME they use has the following nameservers :

- `ns-568.awsdns-07.net.`
- `ns-1655.awsdns-14.co.uk.`
- `ns-6.awsdns-00.com.`
- `ns-1160.awsdns-17.org.`

Those addresses all point to an AWS nameserver's domain + subdomain (`ns-XX.awsdns-YY.EXT.`) where XX corresponds to the nameserver's number on the YY domain with an EXT $\in$ {com, co.uk, net, org}.

As we can see on the figure 1 in the "ADDITIONAL SECTION", all those nameservers can be reached via both IPv4 and IPv6 addresses which respectively belong to the prefixes `205.251.0.0/16` and `2600:9000::/32`

C. Time to live

The first Time To Live we will talk about is the one of the CNAME (d5w2uqtmtwurf.cloudfront.net) and the A records that are related to it. Its value is 60 seconds. The second one, the one of the NS records corresponding to the CNAME has a value of 21.600 seconds (6 hours). Finally, the IP addresses corresponding to those nameservers also have a TTL of 6 hours.

Let's now explain why those TTL are so different. The biggest ones, 6 hours, are because the nameserver's addresses

change very rarely as the resolver (AWS in this case) configures the servers and their addresses are set once. The CNAME can also use those nameservers for a long time as they should not change very often. The A records of the CNAME have a very small TTL (1 minute) because requests are coming from many users and the best server to reach, for a single user, might easily change over time (congestion, load...).

D. Extra records

We can also find some other records such as TXT or SOA.

- TXT ones are used to store some text data. According to whois.com [7], 01net.com has 2 TXT records which contains some domain verification for Facebook and Google. They both have a TTL of 300 seconds.
- SOA records are used to control zone transfers and contains some information about the DNS administrator and authority (TTL = 60 seconds).

III. HTTP AND HTTPS

In this section, unless it is specified, HTTP will refer to HTTP and HTTPS. *Note: HTTP(S) stands for HyperText Transfer Protocol (Secure).*

A. Request and Response Headers

As said in the introduction, when we are trying to reach http://01net.com, we are automatically redirected to https://www.01net.com. The way it works can be explained with the HTTP headers.

First, the client (Google Chrome BETA) sends an HTTP/1.1 request to the server with, among others, the `upgrade-insecure-request` header which is set to 1. It tells to the server that, as explained on the W3C official website [9], the server should redirect the client to the HTTPS version of the website if available.

When the server intercepts the request, it gets the headers and generates an HTTP/1.1 response which contains the header `Location` set to `https://www.01net.com` and a `Status code` with the value `301`. The `www` subdomain has been added by the programmers in the website's code. The `HTTPS` can as well have been added by the programmer or in the server's NGINX configuration.

The client gets the HTTP/1.1 response, is redirected to `https://www.01net.com` and then tries to reach this URL by sending an HTTP/2 request which is secured with TLS (explained in section IV).

Finally, the server responds with a 200 Status code and the source code of the page stored in the body.

Every HTTPS request has been established over the port 443 of the server while the non-secured initial request was made using the port 80.

A request/response can also contain many other standard headers such as: `user-agent`, `accept`, `referer`, `accept-encoding`, `accept-language`... in the request and `status`, `content-type`, `date`, `server`, `last-modified`, `e-type`, `vary`, `via`... in the response.

As 01net.com uses the HTTP/2 version of the protocol, they also use the "pseudo-header" field which, in this case, contains the following ones: `:method`, `:authority`, `:scheme` and `:path`

Non-standards headers like `x-cache` and `x-amz-cf-id` are also used in the responses.

- `x-cache` is used by many CDN to specify whether the request was served by the origin or a proxy of the CDN.
- `x-amz-cf-id` is a non-standard header added by Amazon Web Services to "identify the request" [10] when it is sent from the cloudfront to the origin server. It is then set into the response the client gets.

B. Resources types

When we get a website source code, it often contains many references to scripts, stylesheets, images, fonts etc. Those are requested from many other websites such as: `static.bfmtv.com`, `img.bfmtv.com`, `www.google-analytics.com`, `dmp.theadex.com`.... The following figure (2) shows the distribution of those data in term of size.

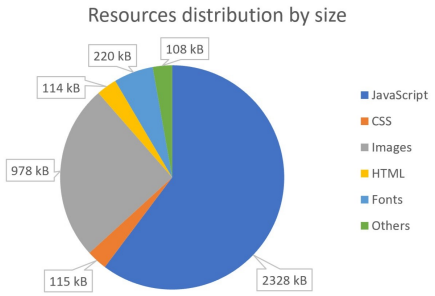

Fig. 2. Resources distribution by size (Arlon Dec. 9, 2018)

As we can see, the Javascript, CSS, fonts and images resources represent more than 3.5MB of data (compared to the 114.1 kB of the page itself). That is why browsers keep them in cache so they don't need to reload them again.

When we look deeper into the source code, we can observe that some very used libraries like JQuery [11] or CSS frameworks like Twitter Bootstrap [12] are loaded from the 01net CDN (Amazon) and not from the Jquery or Bootstrap CDN. That means that, for example, if an user already has the JQuery code (requested from the JQuery CDN) in cache, he will have to download it again from 01net. That's of course the same for the Twitter bootstrap or any other widely used library. Loading them from the origin CDN could improve the loading speed of the website.

C. Cookies and Session variables

Navigating on the 01net.com website implies getting a lot of cookies of many different types stored on our computer. Those cookies can as well be set by 01net itself but they can also be set by other websites which are included (via IFrames or JS scripts) on the 01net website.

In order to analyze the cookies which are saved when loading the page, I chose to access the website following 2 different scenarios:

1) Regular: In this scenario, I have already visited the website with the same browser several times with an advertising blocker (AdBlock [13]). The cookies and javascript are authorized in the browser's settings.
2) Incognito: In this scenario, I am visiting the website using the "Incognito mode" of Chrome BETA. This mode disables all the extensions (AdBlock is then also disabled) and there are no cookies before the first request. The cookies and javascript are authorized in the browser's settings.

Before requesting the website, there are already a lot of cookies in the "regular" scenario. Here are some examples we can find: `Locale`, `ADSENSE`, `CONSENT`, `TCPID`, `TestIfCookieP`, `bm_last_load_status`, `pubconsent`, `euconsent`, `gdprconsent`, `publica_session_id`, `_ga` etc...

When the request is done, in the "regular" scenario, there is no significant difference in term of amount and names of the cookies. On the other hand, in the "incognito" scenario, most of the cookies that existed in the "regular" mode have been created and a lot of new ones such as `_fbp`, `uid`, `ad-id`, `ad-privacy`, `adtrc`, `sasd...` have also been set.

Let's now talk about the role of those cookies. As their name has been set by the developers, they do not always mean something and are hard to understand. Hopefully, with some experience and deduction, we can try to explain their usage by looking at their names, values or domain.

In the incognito mode, most of the "new" cookies are related to advertising. They are used to track the user. When we navigate on the web, as other websites also include IFrames or JS scripts of advertising services, we send those cookies back and the advertisers know we went on that specific website.

Many cookies are also related to the user's consent. In fact, since the General Data Protection Regulation (GDPR) [14] came into force, websites are required to ask for people consent in order to set tracking, marketing etc. cookies. Those agreements are the stored as cookies...

Some other cookies just store the user's locale or the session id (required in order to store session variables).

Session variables are not really cookies. They consist in variables that store a value and that are deleted when the user closes his browser. Cookies, them, are deleted when the expire.

IV. Transport Layer Security

In order to securely communicate with its clients, 01net exchanges information using the version 1.2 of the Transport Layer Security (TLSv1.2) protocol. This protocol consists in a Certificate delivered by a third party which allows the client to be sure that the server he is contacting is effectively the one he is trying to reach. It also permits to encrypt and decrypt the exchanged data.

As said before, when we try to access http://www.01net.com, we are redirected by the server to https://www.01net.com. In fact, even if 01net.com has disabled the `HTTP Strict Transport Security` which only authorizes a client to reach the website via `HTTPS`, they redirect the client to the secure version of the website.

A. Certificate

The certificate I was talking about earlier has been issued by `GlobalSign Organization NV-SA` to `NEXT RADIO TV`. It can be used to communicate with the domain and every subdomain of `01net.com`.

On December 13, 2018, the certificate was valid until February 12, 2019 and the signature algorithm that was used is `PKCS #1 SHA-256 With RSA Encryption`. Let's now decrypt what it means.

- `PKCS #1` corresponds to Public-Key Cryptography Standards [15]. It provides specifications to implement the RSA algorithm.
- `SHA-256` is an hashing algorithm (irreversible) which is deterministic and gives a 256 bit output which corresponds to the hash of the input.
- `RSA` is a system which relies on public-private key pair. The public key is shared and used to encrypt messages by anyone. The private key is kept secret and is the only way to decode the encrypted message.

In order to be able to communicate, the certificate must be trusted by both parties (server and client). According to my tests using different browsers (Microsoft EDGE 44, Google Chrome BETA 65, Mozilla Firefox Developer 65 and Internet Explorer 11) and the results returned by SSLLABS.com [16], the issuer (Globalsign) is trusted by all browsers.

It doesn't mean the handshake will work with all browsers. As said on SSLLabs, "*This site works only in browsers with SNI support*". `SNI` corresponds to Server Name Indication. It is an extension to the TLS protocol which enables the client client to send the domain name as part of the TLS negotiation. The connection does not only rely on the IP address but also on the virtual domain. As said before, some (old) browsers do not support that extension and thus are not able achieve the TLS handshake.

B. TLSv1.2 protocol details

Let's now suppose that an attacker has intercepted the encrypted data and, afterwards, has also been able to find the secret key to decode those data. As we can once again see on SSLLabs, `forward secrecy` is enabled on 01net. That means that the attacker will be able to decode the data that will be sent in the future but he won't be able to decode the one that was sent before he got the key.

Another interesting security feature is the `Downgrade attack prevention`. As mentioned on the Internet Engineering Task Force website [17], the aim is to prevent malicious users from using a lower version of the security protocol to take advantage of vulnerabilities. On 01net.com, the protocol takes care of preventing this attacks

as `TLS_FALLBACK_SCSV` (downgrade attack prevention) is supported.

In order to be 100% safe to navigate on, all the domain names that are accessed by www.01net.com are accessed using HTTPS so that everything is encrypted end-to-end.

V. Transmission Control Protocol

In order to analyze the Transmission Control Protocol (TCP) for the 01net.com website, I decided to analyze the TCP segments sent and received by a Windows 10 computer and an Android smartphone. Both were using Google Chrome BETA to reach the website. The packets were collected by the tPacketCapture App [18] on Android and then analyzed using Wireshark [19]. On windows, both steps have been done using Wireshark.

A. Parallel connections

Before requesting the 01net website, I looked up the addresses answered by the DNS in order to know which one the browser could request.

- Computer : I found 2 of the 4 addresses that were used in parallel to request the site (`54.230.129.215` and `54.230.129.5`).
- Smartphone : The 4 addresses returned by the DNS were used (`52.85.224.{83, 87, 92, 170}`).

This difference might come from the fact that phones are known to have a slower internet connection and thus, the browser establishes more TCP connections to reduce congestion and have more path to get the information.

B. Starting the connection

During the three-way handshake, the client and the server exchange specific segments (SYN, SYN-ACK, ACK) in order to define how they are going to use the protocol.

First, they each send their window size. It says how much data can be transmitted before receiving an acknowledgment. On my smartphone, the window has a size of 65.535 just like the server's one. On the other hand, on my computer, the window size is 17.520 and the server communicates a window size of 29.200.

The following options have the same negotiated value using the computer or the phone:

- Maximum segment size (MSS): 1452 Bytes. It represents the maximum size of the carried payload. Actually, the client sends a MSS of 1460 and the server replies with 1452.
- Window Scale: 8. That means that the calculated window size equals $2^8 = 256$ times the window size.
- Selective acknowledgment: Permitted. When a packet is lost, we acknowledge the next ones with its sequence number but tell the sender which packets were received using the left edge and right edge of the interval.

Regarding the timestamp, there is a negotiation of that option when using chrome BETA for Android but not on Windows 10. It communicates the client's initial timestamp value.

C. Round Trip Time and packets' path

In order to get the response from the server, the packets need to travel to that server. That operation may take some time. My measurements made using the `ping`, `traceroute` and Wireshark showed an RTT between 25 and 38 ms. This value has been computed from Arlon, the path followed by the packets is shown on figure 3. This small amount of time comes from the fact that we are requesting the optimal server to handle our request thanks to the CDN.

Fig. 3. Path followed by the packets from Arlon (Dec. 15, 2018)

D. Connection release

Since the time when the browser started the connection, this TCP connection remains open for 120 seconds before the browser sends a `FIN` packet to close the connection. The connection is then ended gracefully. The server does not close the connection itself. If I manually close the brower, a `RST` packet is sent and that is the only moment the connection is closed ungracefully.

VI. Conclusion

During this analysis of 01net.com, we have discovered they were using Amazon Web Services to host their website. This assumption has been confirmed by different elements: the amazon nameservers, the usage of cloudfront, the specific HTTP headers, the IPs etc.

On the HTTP side, I found out that some resources were hosted on their CDN as it would be more efficient to use the origin CDN.

I also discovered that reaching the website using my computer or my phone using the same browser (Google Chrome BETA) does not have the same effect in term of TCP connection. In fact, different amount of TCP connections were initialized on the different devices.

It has been impossible for me to analyze everything and to say everything that could be said using 4 pages. I could have discussed the expiring date of the cache, detailed the impact of the advertisers on the website, explained how a CDN works etc. Hopefully, we already have a global view with some details of the analysis of the 01net.com website.

REFERENCES

[1] Wikipedia.com. 2018. “01net (site web)“. [ONLINE] Available at: https://fr.wikipedia.org/wiki/01net_(site_web). [Accessed on December 4, 2018].

[2] Alexa.com. 2018. “01net.com Traffic, Demographics and Competitors“ - Alexa. [ONLINE] Available at: https://alexa.com/siteinfo/01net.com. [Accessed on December 4, 2018].

[3] otx.alienvault.com. 2018. “AlienVault - Open Threat Exchange“. [ONLINE] Available at: https://otx.alienvault.com/indicator/ip/54.230.129.59. [Accessed on December 9, 2018].

[4] Aws.amazon.com. 2018. “Amazon Web Services (AWS) - Cloud Computing Services“. [ONLINE] Available at: https://aws.amazon.com/. [Accessed on December 5, 2018].

[5] Aws.amazon.com/cloudfront/. 2018. “Content Delivery Network (CDN) — Low Latency, High Transfer Speeds, Video Streaming — Amazon CloudFront“. [ONLINE] Available at: https://aws.amazon.com/cloudfront/. [Accessed on December 9, 2018].

[6] Developers.google.com. 2018. “Public DNS — Google Developers“. [ONLINE] Available at: https://developers.google.com/speed/public-dns/. [Accessed on December 9, 2018].

[7] Whois.com. 2018. “Whois.com - Domain Names & Identity for Everyone“. [ONLINE] Available at: https://dig.whois.com.au/dig/www.01net.com. [Accessed on December 9, 2018].

[8] Site24x7.com. 2018. “Analyze webpages, improve web site performance: Site24x7 Tools“. [ONLINE] Available at: https://www.site24x7.com/web-page-analyzer.html. [Accessed on December 9, 2018].

[9] W3.org. 2018. “Upgrade Insecure Requests“. [ONLINE] Available at: https://www.w3.org/TR/2015/CR-upgrade-insecure-requests-20151008/. [Accessed on December 10, 2018].

[10] Docs.aws.amazon.com. 2018. “Request and Response Behavior for Custom Origins - Amazon CloudFront“. [ONLINE] Available at: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/RequestAndResponseBehaviorCustomOrigin.html. [Accessed on December 10, 2018].

[11] Jquery.com. 2018. “Bootstrap The most popular HTML, CSS, and JS library in the world.“. [ONLINE] Available at: https://getbootstrap.com/. [Accessed on December 13, 2018].

[12] Getbootstrap.com. 2018. “JQuery“. [ONLINE] Available at: https://jquery.com/. [Accessed on December 10, 2018].

[13] Getadblock.com. 2018. “Surf the web without annoying pop ups and ads¡‘. [ONLINE] Available at: https://getadblock.com/. [Accessed on December 10, 2018].

[14] Eugdpr.org. 2018. “EUGDPR Information Portal“. [ONLINE] Available at: https://eugdpr.org/. [Accessed on December 10, 2018].

[15] Wikipedia.org. 2018. “PKCS 1 - Wikipedia“. [ONLINE] Available at: https://en.wikipedia.org/wiki/PKCS_1. [Accessed on December 13, 2018].

[16] SSLLabs.com. 2018. “SSL Server Test: www.01net.com (Powered by Qualys SSL Labs)“. [ONLINE] Available at: https://www.ssllabs.com/ssltest/analyze.html?d=www.01net.com&s=13.35.125.11. [Accessed on December 13, 2018].

[17] IETF.org. 2018. “RFC 7507 - TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks“. [ONLINE] Available at: https://datatracker.ietf.org/doc/rfc7507/. [Accessed on December 13, 2018].

[18] Taosoftware Co.,Ltd. 2018. “tPacketCapture version 2.0.1“ [Android Application] Available at: https://play.google.com/store/apps/details?id=jp.co.taosoftware.android.packetcapture. [Downloaded on December 15, 2018].

[19] Wireshark.org. 2018. “Wireshark Go Deep.“. [ONLINE] Available at: https://www.wireshark.org/. [Accessed on December 15, 2018].