URL Filtering using SNI for HTTPS websites

Kevin Jones
Security Engineer, US Central
December 12, 2018

Contents
Check Point URL Filtering .................................................................................................................. 1
URL Filtering for HTTP ......................................................................................................................... 2
URL Filtering for HTTPS ..................................................................................................................... 4
Categorize HTTPS Websites via Certificate Checking ................................................................. 5
Problems with HTTPS Categorization via Certificate Checking ...................................................... 6
Categorize HTTPS Websites via SNI ............................................................................................. 12

Check Point URL Filtering

Check Point has included web filtering with its security gateways for many years. URL Filtering is a standard blade included with all Next Generation Threat Prevention (NGTP) and Next Generation Threat Extraction (NGTX) packages. As is common with most major web filtering vendors, Check Point tracks over 200 million websites and categorizes them into 70+ categories for easy identification and policy configuration. These categories are typically used in the Access Control policy to Block and/or Allow websites by their category tags, with specific sites being identified by the creation of Custom Site to allow for policy exceptions to the general category, or for categorization override. An example URLF policy for Check Point R80.10 is shown here:

The process of maintaining these website categorizations, querying the cloud database for current categories and locally enforcing web filtering policies based on these categories is necessarily a complex procedure with many options. For our purposes, however, we can assume that this is all magic and works perfectly in all circumstances. Still, in order for URL filtering to work accurately in real time, the security gateway must have a way to discover the website that is being accessed in the first place. As it turns out, this is not always easy.
URL Filtering for HTTP

For clear-text web connections using HTTP on standard port 80, the gateway only needs to read and categorize the site requested by the client via HTTP GET command. For example, I used Firefox to http://www.d155.org/ for our local school district website.

Via tcpdump/wireshark, we can see that request as it traverses the gateway with the original GET command:

As it happens, website request is successfully redirected to another related site which the browser then loads:
Since this is also a clear-text HTTP website, we can see that client / server communication as well via tcpdump / wireshark:

This recursively prompts another client GET for the new location, which is redirected one more time before the client requests the current site and the page is loaded:

The point here is that the requested website and all the client/server communication is clear-text and thus readable by the security gateway. It can parse the requested website
and lookup the associated category of the site, and thus enforce the configured web filtering policy.

**URL Filtering for HTTPS**

But, this method of URL capture and categorize is not possible on the large and growing number of websites that use HTTPS to encrypt the HTTP client/server communication. There are a number of sites that may respond to an HTTP client request simply to re-direct the client to an HTTPS-enabled website, and a number of others that will only communicate using HTTPS. Because the HTTP communication is encrypted, the gateway (or any other device in the middle) is unable to read the GET/POST, etc. of the HTTP messages. So how does the gateway determine the URL of the requested website in order to block or allow the connection based on the URL’s categorization?

Look at a simple request for Facebook which loads https://www.facebook.com:

![Facebook request](image)

The associated tcpdump/wireshark analysis of packet flow does not show any HTTP requests:

![TCPdump analysis](image)

Note that we only see the DNS request for www.facebook.com, then TCP handshake, and then TLSv1.3 using SSL. The DNS request, although clear-text, is not a reliable guide to the subsequent HTTPS connection.

Many organizations have opted to use full HTTPS inspection, inserting their edge security gateways into the middle of this transaction in order to decrypt and then re-encrypt the connection. This route involves practical, performance and security obstacles, and is not a workable solution in all environments. So what other option is there to reliably filter HTTPS web traffic?
Categorize HTTPS Websites via Certificate Checking

Check Point offers a feature called “Categorize HTTPS websites” in the advanced settings for Application Control & URL Filtering found under the “Manage & Settings” tab in R80.x:

How is this accomplished? Obviously, the gateway must be able to read the URL in the part of the client/server communication that is clear-text. So let’s take another look at the Facebook connection. Although the HTTP communication is SSL encrypted, the associated certificate is transferred in clear-text and can be read by the gateway. And the cert presumably will identify the website via Common Name (CN). In the case of Facebook, the CN is listed as *.facebook.com

This is how a Check Point gateway can determine the website being accessed, look up the category and enforce the filtering policy. The log for this traffic reflects the correct category:
Problems with HTTPS Categorization via Certificate Checking

That is a relatively simple solution, but it doesn’t work in all cases. Looking at the details in the site certificate:

Looking deeper, the certificate includes a field for Subject Alternative Name. For this certificate, there are a number of other DNS names listed including one for *.messenger.com.
What happens if we try to access `https://www.messenger.com`? We will get the exact same certificate:

And given the HTTPS categorization according to CN of the certificate, the site will be blocked or allowed as though this is a connection to Facebook.
And the associated URLF log shows this Accept based on Facebook category of Social Networking:

That might seem OK, because these are related sites as Messenger was developed by Facebook. But by itself, Messenger is not properly categorized as Social Networking, but as Instant Chat or Instant Messaging:

URL Categorization

For http://www.messenger.com
Categories: Instant Chat, Instant Messaging

If the URL policy accepts Social Networking, but blocks Instant Messaging, www.messenger.com will be incorrectly categorized and allowed. To correctly categorize
the site, a custom site specifically created in order to block Messenger would be required. And this is just one case of many similar websites that share a certificate with sites that belong in different categories.

For a more extreme case, consider the following scenario. A corporate URL filtering policy very commonly blocks sites categorized as Sex or Pornography. One such site is

![RedTube Porn Videos](https://www.redtube.xxx)

But with a Block on Pornography, why is that website allowed? It’s HTTPS, so we can look at the certificate, and we find that the CN is “sni136328.cloudflaressl.com”.

![Certificate Viewer](https://example.com/certificate.png)

And that URL is categorized as “Computers / Internet”, so we see that the traffic is allowed based on that information when we look at the associated log.
Why is “redtube.xxx” using a certificate from “sni136328.cloudflaressl.com”? And should the browser at least give a warning to the user that this site is not the same as listed in the certificate? If we look deeper at the certificate details, we find multiple DNS names listed as they all share the same certificate from cloudflare.
These sites are very different, but all using the same certificate as the basis for categorization. But each of these DNS name should be categorized differently according to Check Point’s URL lookup site [https://urlcat.checkpoint.com/urlcat/main.htm](https://urlcat.checkpoint.com/urlcat/main.htm)

- casino-zilla.com – Gambling
- aquariusbeachclub.com – Travel
- dariopereyra.com.ar – General
- vsseafood.com – Restaurants / Dining / Food

Moreover, [http://www.redtube.xxx](http://www.redtube.xxx) itself is categorized as a Sex site when using HTTP rather than HTTPS.
So the use of the CN listed in the certificate is not a reliable guide when the Subject Alternative Name (SAN) field is used to cover multiple different sites. And trying to use the SAN entries to make a category choice will lead to contradictory results. We need to discover which of those sites the client is trying to access. Also, this information must be found in clear-text portion of the client/server communication.

**Categorize HTTPS Websites via SNI**

Looking again at the tcpdump / wireshark analysis of the traffic, the very first Client Hello using SSL shows the server that the client is specifically requesting, under the Server Name Indication (SNI) extension.

Using this field, rather than relying on the CN of the certificate, gives more specific and accurate information about the requested site. The recently released hotfix for Check Point R80.10 gateways does change this behavior and utilize the SNI extension for categorization. Once installed, the feature can be enabled with the following command:

```bash
fw ctl set int urlf_use_sni_for_categorization 1
```

This hotfix also allows a further check to make sure that the SNI requested by the client matches one of the SAN entries on the certificate. To enable this feature, use this command:

```bash
fw ctl set int urlf_block_unauthorized_sni 1
```

With it, we can now see that this site, despite the shared certificate and HTTPS encryption, is properly categorized and blocked in the gateway logs.
This feature is available now for R80.10 gateways with Take 154 via Solution Center, and planned to be rolled into Jumbo Hotfix release for both R80.10 and R80.20 in the near future.