Does https encrypt the whole URL?

Asked 9 years, 8 months ago Modified 9 years, 8 months ago

Viewed 8k times

9

I googled a lot and many answers are **Yes**. For example: Is GET data also encrypted in HTTPS? But the senior security engineer in our company told me the URL would not be encrypted.

> Image that, if the URL was encrypted, how does the DNS server find the host and connect?

I think is this is very strong point although it's against most of the answers. So I'm really confused and my questions are:

1. Does https encrypt the everything in the request? (including the URL, host, path, parameters, headers)
2. If yes, how the DNS server decrypt the request and send it to the host server?

I tried to access https://www.amazon.com/gp/css/homepage.html/ref=ya_surl_youracct and my IE sent two requests to the server:

First:

```
CONNECT www.amazon.com:443 HTTP/1.0
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64;
Trident/7.0; rv:11.0) like Gecko
Host: www.amazon.com
Content-Length: 0
DNT: 1
Connection: Keep-Alive
Pragma: no-cache
```

Second:

```
GET /gp/css/homepage.html/ref=ya_surl_youracct
HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: en-US,zh-CN;q=0.5
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64;
Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: www.amazon.com
DNT: 1
Connection: Keep-Alive
```

It seems my browser has requested twice: the first time is to establish the connection with host (without encryption) and the second time send an encrypted request over https? Am I right? If I am understanding this correctly, when a client call the RESTFUL API using https, it sends the requests (connection and get/post) twice every time?

security **url** **encryption** **https** **dns**

Share

Improve this question

edited May 23, 2017 at 11:46

Community Bot

1 ● 1

Follow

asked Apr 17, 2015 at 9:27

53iScott

857 ● 2 ● 13 ● 18

In terms of security you should assume that the URL is public. This isn't really the case (see JohnWu's responses) but you, as @T.Rob states, you should assume that they can be viewed and put nothing sensitive in them. – Neil Smithline Apr 17, 2015 at 18:11

3 Answers

Sorted by: Highest score (default)

14

The URL **IS** encrypted from the time it leaves the browser until it hits the destination server.

What happens is that the browser extracts the domain name and the port from the URL and uses that to resolve DNS itself. Then it starts an encrypted channel to the destination server IP:port. Then it sends a HTTP request through that encrypted channel.

The important part is anyone but you and the destination server can only see that you're connecting to a specific IP address and port. They can't tell anything else (like specific URLs, GET parameters, etc).

Attackers can't even see the domain in most cases (though they can infer it if there is actually a DNS lookup - if it wasn't cached).

The big thing to understand is that DNS (Domain Name Service) is a completely different service with a different protocol from HTTP. The browser makes DNS lookup requests to convert a domain name into an IP address. Then it uses that IP address to issue a HTTP request.

But at no time does the DNS server receive a HTTP request, and at no time does it actually do anything other than provide a domain-name - IP mapping for users.

Share Improve this answer

Follow

answered Apr 17, 2015 at 13:21

ircmaxell

165k ● 35 ● 268 ● 315

How about the RESTFUL APIs? When the client tried to call them, does the client also send two requests every time? Like the browsers do? – 53iScott Apr 17, 2015 at 14:36

REST has nothing to do with it. And the result of the DNS request can be cached up until the length of time specified by the DNS response (typically 1 hour). – ircmaxell Apr 17, 2015 at 14:39

8

While the other responses are correct so far as they go, there are many other considerations than just the encryption between the browser and the server. Here are some things to think about...

- The IP address of the server is resolved.
- The browser makes a TCP socket connection to the server's IP address using TLS. This is the `CONNECT`

you see in your example.

- The request is sent to the server over the encrypted session.

If this was all there is to it, you are done. No problem.

But wait, there's more!

Having the fields in a `GET` instead of a `POST` reveals sensitive data when...

- Someone looks in the server logs. This might be a snoopy employee, but it can also be the NSA or other three-letter government agency, or the logs might become public record if subpoenaed in a trial.
- An attacker causes the web site encryption to fall back to cleartext or a broken cipher. Have a look at the SSL checker from Qualsys labs to see if a site is vulnerable to this.
- *Any* link on the page to an external site will show the URI of the page as the referrer. User ID and passwords are unintentionally yet commonly given away in this fashion to advertising networks. I sometimes spot these in my own blog.
- The URL is available in the browser history and therefore accessible to scripts. If the computer is public (someone checks your web site from the guest PC in the hotel or airport lounge) the `GET` request leaks data to anyone else using that device.

As I mentioned, I sometimes find IDs, passwords and other sensitive info in the referrer logs of my blogs. In my case, I contact the owner of the referring site and tell them they are exposing their users to hacking. A less scrupulous person would add comments or updates to the site with links to their own web site, with the intention of harvesting the sensitive data in their referrer logs.

So your company's senior security engineer is correct that the URL is not encrypted in many places where it is extremely important to do so. You and the other respondents are also correct that it *is* encrypted in the very narrow use case of the browser talking to the server in context of a TLS session. Perhaps the confusion you mention has to do with the difference in the scope of these two use cases.

Please see also:

- Testing for Exposed Session Variables (OTG-SESS-004)
- Session Management - How to protect yourself (Note that "always use `POST`" is repeated over and over on this page.)
- Client account hijacking through abusing session fixation on the provider

Share Improve this answer

Follow

answered Apr 17, 2015 at 16:45

T.Rob

31.8k ● 9 ● 63 ● 105

The URL (also known as "Uniform Resource Locator") contains four parts:

1. Protocol (e.g. https)
2. Host name (e.g. stackoverflow.com)
3. Port (not always included, typically 80 for http and 443 for https)
4. Path and file name or query

Some examples:

```
ftp://www.ftp.org/docs/test.txt
mailto:user@test101.com
news:soc.culture.Singapore
telnet://www.test101.com/
```

The URL as an entire unit is not actually encrypted because it is not passed in its entirety. The URL is actually pulled apart into bits and each part is used in different ways. E.g. the protocol portion will tell your browser how to use the rest of the URL, the host name will tell it how to look up the IP address of the intended recipient, and the port will tell it, well, which port to use. **The only portion of the URL that is passed in the payload itself is the path and query, and that portion is encrypted.**

If you take a look at an HTTP request in the raw, it looks something like this:

```
GET /docs/index.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0;
Windows NT 5.1)
(blank line)
--Body goes here--
```

What you see in the example above is passed. Notice the full URL appears nowhere. The host header can actually be omitted completely (it is not used for routing). The only portion of the URL that appears here is to the right of the GET verb, and only includes the rightmost portion of the original URL. The protocol and the port number appear nowhere in the message itself.

Short answer: Everything to the right of the port number in the URL is included in the payload of the https request and is in fact encrypted.

edited Apr 17, 2015 at 21:53

answered Apr 17, 2015 at 9:40

John Wu
52.2k ● 8 ● 47 ● 89