An Investigation on HTTP/2 Security

Meenakshi Suresh[1], P. P. Amritha[1], Ashok Kumar Mohan[1]
and V. Anil Kumar[2]

[1]*TIFAC-CORE in Cyber Security, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India*
[2]*CSIR Fourth Paradigm Institute (CSIR-4PI), Bangalore, India*
E-mail: cb.en.p2cys16017@cb.students.amrita.edu;
{pp_amritha, m_ashokkumar}@cb.amrita.edu; anil@csir4pi.in

Received 04 January 2018; Accepted 29 March 2018;
Publication 04 May 2018

Abstract

In the current world scenario where everyone is using the Internet, it is becoming a strenuous task to preserve security. Furthermore the world is becoming progressively digital by the passing of each minute. A large portion of the Internet is conducted using the Hyper Text Transfer Protocol (HTTP). But in 2015, it underwent a consequential enhancement and was released as HTTP/2.

HTTP/2 includes pipelining, response multiplexing, server push and header compression using HPACK besides the properties of HTTP/1.1. These properties make it difficult for the eavesdroppers to monitor or fingerprint a website running on HTTP/2. This paper deals with the research on how strong the HTTP/2 protocol keeps the user information hidden and secure. By monitoring a live network traffic, its properties with HTTP/2 is assessed. This study helps understand the different aspects of the protocol and its influence on the network and browsers.

Keywords: HTTP, HTTP/2, SPDY, Server Push, HPACK.

Journal of Cyber Security and Mobility, Vol. 7_1, 161–180. *River Publishers*
doi: 10.13052/jcsm2245-1439.7112

1 Introduction

As of 2016 it was calculated that 81% of people in the developed parts of the world use Internet. Whenever the people are accessing Internet, it is of considerable significance to guarantee that they are doing it in a secure way. Especially when it includes the sharing of delicate information, such as personal or financial data, the security goals such as Confidentiality, Integrity and Availability (CIA) has to be maintained. The Internet users will lose quite a lot of information when they are using online facilities, and it is difficult to ensure its complete secrecy and security, irrespective of how secure the website or the browser seems to be. Since no protocol or system is 100% secure, there will be vulnerabilities and defects always.

Hyper Text Transfer Protocol [16], an application layer protocol, is an avid and successful protocol that stands as the basis of the world wide web (WWW). It was developed to make the communication between client and server efficient. HTTP is liable when messages are crafted and presented, i.e; it deals with how the data has been requested from the servers are displayed to the users through browsers. It works on the major protocols that rule the Internet, TCP/IP. Currently in use is the updated version of HTTP that provides security is HTTP/2 and is gaining its popularity in the recent times [5, 17].

HTTP/2 has made tremendous changes to the current way of network traversals with its revised properties and security. It is a derivative of the Google's SPDY protocol which tries to solve the issues faced in the previous versions of HTTP. SPDY will manage the HTTP traffic with the help of appropriately predefined goals to reduce web page load latency and also enhance the web security. HTTP/2 was designed to enhance the features of SPDY and thereby create a revamped version of HTTP which tries to enable an efficient and faster use of the network, in a secure way. It does not revise the sheer definition of the previous version 1.1, but modifies it to enable an efficient use of the network resources. It also includes a diminished approach of latency by adding compression to the header fields using HPACK [6] and allows numerous concurrent transactions on the same connection.

2 Literature Survey

HTTP has undergone many changes rapidly that the first standardized form of HTTP, HTTP/1.1 was distributed just a few months after the initial versions HTTP/0.9 and HTTP/1.0. The old version has changed by keeping its veracity as such and remaking the structure. It has advanced from an early convention

to exchange data in a semi-credible research facility condition to the cutting edge complication of the Internet, now conveying pictures, recordings in high determination and 3D. This paper proceeds by understanding the growth of HTTP and describes its journey till the current version, HTTP/2.

2.1 Evolution - HTTP

The transition of HTTP protocol from HTTP/0.9 to the current version of HTTP/2 has been a long journey. HTTP/1.1 was the most used version of the protocol, until now. The need for a better and effective security while using the Internet led to this evolution. The security shortcomings made an overall negative impact for the protocol.

HTTP/0.9

It all started with the introduction of the WWW, when HTTP, a new system was used to keep track of the information used or created. The first version that was devised was the HTTP/0.9 which was a very simple protocol with no headers and a lot of limitations. It just had a single method, GET, which was designed to fetch the HTML data only and no the images. These shortcomings were addressed by the next generation of HTTP, i.e. HTTP/1.0.

HTTP/1.0

Version 1.0 [1] led to an enormous supply of modification to the little protocol which had started the revolution to the world of networks. It had many new properties like Headers, Errors, Conditional requests, Content encoding, Response codes etc. This scenario made HTTP much bigger than what it was. Even though these changes were tremendous, HTTP/1.0 suffered from some vulnerabilities. It came across the problem of not being able to keep a connection open till that request is satisfied. It also lacked the mandatory Host header and also the base option for caching.

HTTP/1.1

Soon after the release of version 1.0, the new modifications were added to the current version and that was introduced as HTTP/1.1 [2]. This new version of the protocol has addressed most of the above mentioned problems of 1.0. One of the major changes was that it made the Host headers compulsory and also extended cacheability. Earlier HTTP/1.0 allowed only one active request per TCP connection and this was considered as a time consuming procedure. Adding pipelining to the connections were considered to be a fix to

the issue at hand. It makes multiple requests and opens multiple connections to achieve concurrency and thus reduce latency. This protocol gave advantage over the performance since it is not necessary for the browser to renew the TCP connection for each and every request. Additional changes like Upgrade header, OPTIONS method, range request and compression with transfer-encoding were also included in this protocol. But this version still suffers from few problems like head-of-the-line blocking [10].

SPDY

HTTP/1.1 stayed on without any new improvements for a solid 10 years until SPDY came along. All throughout these years only the web that we were using has changed. But SPDY came as new change from the monotony of HTTP which everyone thought was irreplaceable. SPDY was proposed as an alternative to HTTP. This new protocol, soon after its proposal, proved that betterment of HTTP was possible. It was integrated quickly into Chrome and Firefox. Similarly, the necessary support in servers and proxies came along at about the same pace. Even though SPDY was introduced, it never replaced HTTP/1.1, it was just added along to the current protocols as another updated version. Because of this HTTP/1.1 was used by mostly all the browsers and websites, as a reliable protocol for more than 15 years until HTTP/2 came in the year 2015.

HTTP/2

SPDY laid the foundation for HTTP/2 [5, 14] and was used for proving out some of its key features.The underlying principle of HTTP/2 is to improve the performance of the transport layer and also to enable higher threshold with lower latency. HTTP/2 along with its new and improved properties like binary property, multiplexing, server push, header compression, flow control, priorities and stream dependencies etc., promises to provide a better web experience.

2.2 Evolution - HTTPS

Hyper Text Transfer Protocol Secure [3] is the encrypted and secure version of HTTP. Mostly classified operations like banking and financial transactions are done using HTTPS as they require more security and encryption. Browsers include a padlock symbol in the address bar to indicate the presence of HTTPS. It generally uses one of the 2 asymmetric Public Key Infrastructure System

protocols like SSL (Secure Socket Layer) or TLS (Transport Layer Security) to incorporate the encryption part. It uses a public key to encrypt the information and is decrypted with the private key. The benefits of the HTTPS connection is that it completely protects the user credentials, and the domains that are found as a secure HTTPS connection can be relied upon too.

HSTS

HTTP Strict Transport Security (HSTS) [4] is used to implement a secure connection for browsers which are communicating each others request and response. This protocol enforces the web servers to communicate only over HTTPS and not over HTTP. This is done to ensure the security of the information transmitted and received. This feature of HSTS is possible only over the server which are already supporting HTTPS and it has been ignored in the HTTP version. This is implemented by the server by supplying a header over an HTTPS connection. HSTS headers over HTTP are ignored. This will give an effect of HTTPS everywhere and is added as an additional security feature of the web. HSTS prevents certain attacks like downgrade attack, where the protocol is forced to work in a low quality mode rather than the high quality encryption that it usually uses, cookie/ session hijacking, which leads to the unauthorized access to information or services by jeopardizing the active session and cookie data, SSL stripping, which is a type of Man-In-The-Middle (MITM) attack technique through which a website secured with HTTPS can be downgraded to HTTP.

3 HTTP/2 Protocol Overview

Internet services and connection speeds have significantly improved since the 1990s, however the HTTP protocol remained comparatively unchanged. The protocol only allowed a single HTTP request to be active at a time, with each request requiring it's own individual packet. This limitation resulted in slow page load times. Header fields sizes have also increased. Due to these issues, Google in 2012 proposed SPDY which acted as an alternative to the legacy of HTTP/1.1 protocol. SPDY aimed to solve these issues by introducing multiplexed requests and by introducing a new header compression technique. SPDY along with a few changes was formally adopted as the new HTTP protocol, HTTP/2 in 2015 and released as a RFC [5]. The following section will contain some of the additional features that the protocol has.

- Binary: The performance of HTTP/2 is optimized by making it a binary protocol. This makes it smaller in size but powerful enough to parse. It also makes the protocol less error susceptible and complex.
- Header Compression: HTTP/2 uses header compression to reduce the bandwidth usage. This property also decreases the redundant header fields. HPACK algorithm is used for the compression procedure, which is based on Huffman Coding [13]. This feature of HTTP/2 provides it with another layer of encryption because of the compression done, while the mandatory usage of TLS in the protocol acts as the first layer of encryption.
- Multiplexing: Under this, several requests can be sent in succession on the same TCP connection, and responses can be received out of order. This technique eliminates the need for multiple connections between the client and the server. The request can be send without waiting for the responses from the previous requests. Multiplexing also reduces the connection overheads, as the TCP connection remains open for the transmission of multiple requests.
- Server Push and Prioritization: Server Push is used to additionally push the objects along with the responses of the requests made. This feature predicts that the user might need additional information regarding the request that was made and proactively pushes it along with the response. Prioritization enables the client to specify which resource is more important than the other, so that it can be handled in the specified order.
- Flow Control: It allows equal utilization of streams and thereby increase the performance. This control intends to make it conceivable to use resources in a better way by not enabling a specific stream to starve, and by managing the changing upstream and downstream associations satisfactorily.

Figure 1 [9] show the transfer of frames which are usually done during the normal working of HTTP/2 protocol.

- 1^{st} Payload from Client to Server: Once the TCP connection is established after the 3-way handshake, client will send Connection Preface, SETTINGS frame and WINDOW_UPDATE frame on a stream with identification number 0 while the HEADERS frame is sent on another stream with identification number 1. Both these streams are a part of the 1^{st} HTTP/2 payload [9].

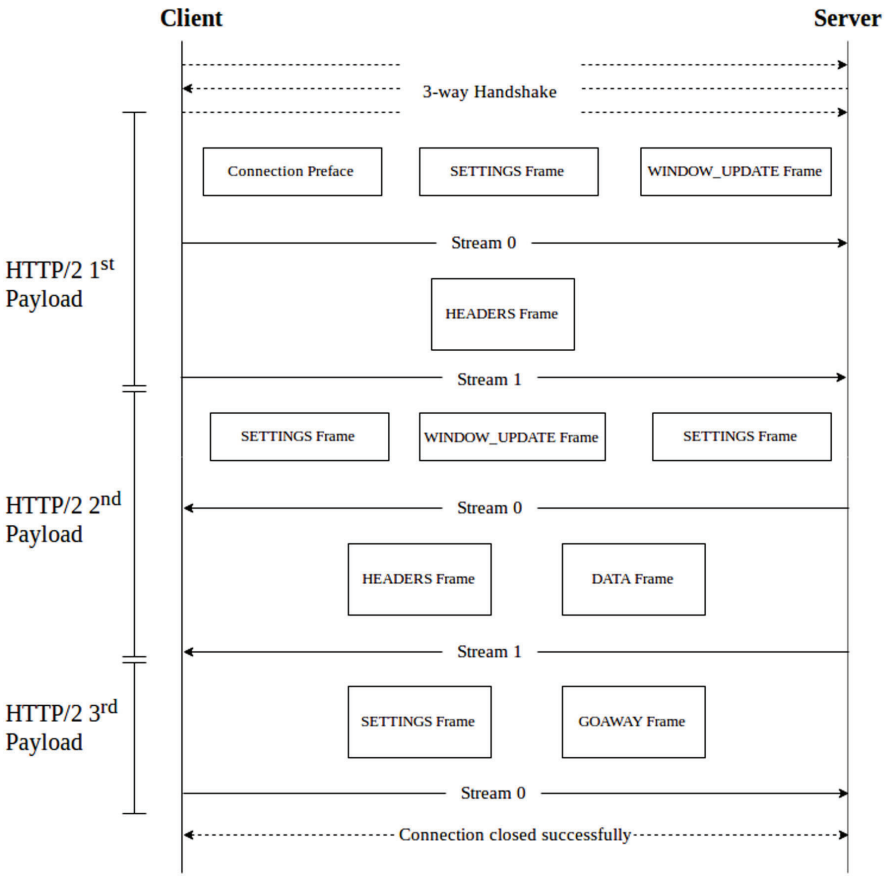

Figure 1 HTTP/2 Working

- 2nd Payload from Server to Client: As soon as the server receives the 1st HTTP/2 payload, it confirms the SETTINGS frame by sending an empty SETTINGS frame on the stream 0 as an acknowledgement. Along with this frame, a WINDOW_UPDATE frame and a non-empty SETTINGS frame is also send by the server. As a reply to the GET request from the HEADER frame, server also sends HEADERS and DATA frames on another stream back to the client [9].
- 3rd Payload from Client to Server: When the client receives the 2nd HTTP/2 payload, it confirms the SETTINGS frame sent by the server and closes the connection by sending a GOAWAY frame [9].

4 Related Work

SANS Institute, an information and cyber security training center, has released a paper which contains a study of the HTTP/2 protocol [12]. It has also shown methods to capture the traffic from different servers browsers etc.

E Adi et al [7], has put up a research thesis, which analyzed HTTP/2 and its risk from constant disruptions due to various factors. Their research focused on exploiting the flow mechanisms and setting frames. They performed analysis on a testbed to illustrate the potential attacks and their attack strengths against a HTTP/2 web server.

Imperva, a cyber security vendor, released a white paper in 2016 with four HTTP/2 flaws [8]. The flaws were a mix of software implementation and protocol bugs. The protocol flaws included a slow read attack, an attack against flow control mechanism which could create a dependency loop, and lastly a flaw against HPACK were they managed to create a header compression bomb, which was able to consume the memory of the web server memory.

5 Rationale for Proposed Work

The new version of HTTP came as an effective change to the world of networks, as it is a much awaited development which is drawing great interest from the research as well as web industry. HTTP/2 unlike its predecessor has an extra layer of encryption because of which a wide distribution of this version of the protocol can be expected.

According to [11] the share of HTTP/2 traffic has increased from 51% to 68% by over a time span of 6 months from October, 2015 to April, 2016 as shown in Figure 2 and Figure 3.

Also Figure 4 shows the result of a survey conducted between the time period of May 2015 to March 2018 has stated that the percentage of websites using HTTP/2 traffic has also increased to 24.4% of all websites. Most of the extensively used websites like Google, Twitter, Gmail, Youtube, Facebook etc. have already ported to HTTP/2 [12].

Figure 5 shows the comparative trends with the percentages of websites using the default HTTP/1.1 protocol over the HTTP/2 protocol based on ranking. So from Figure 4 and Figure 5 we can see that the usage of HTTP/2 has been on the increase since the time it was created.

HTTP/2 has got two versions to itself, h2c and h2. The h2c version is the HTTP/2 version that can be implemented over clear text. This will lead to a connection establishment over an insecure network. A client which makes

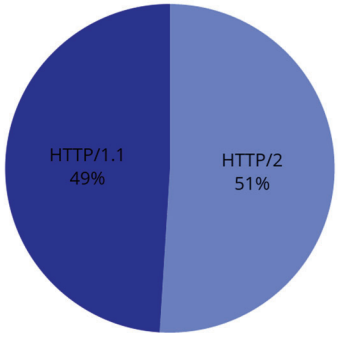

Figure 2 HTTP/2 traffic - Oct, 2015.

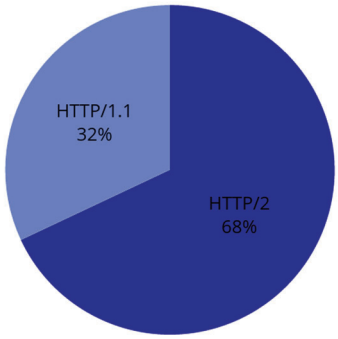

Figure 3 HTTP/2 traffic - April, 2016.

a request for an "http" URI without previous information about support for HTTP/2 on the next hop uses the HTTP Upgrade mechanism. The client does so by making an HTTP/1.1 request that includes an Upgrade header field with the "h2c" token (h2c stands for HTTP/2 cleartext). Such an HTTP/1.1 request MUST include exactly one HTTP2-Settings header field like in the example given below.

```
GET/HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: <base64url encoding of HTTP/2 SETTINGS
                payload>
```

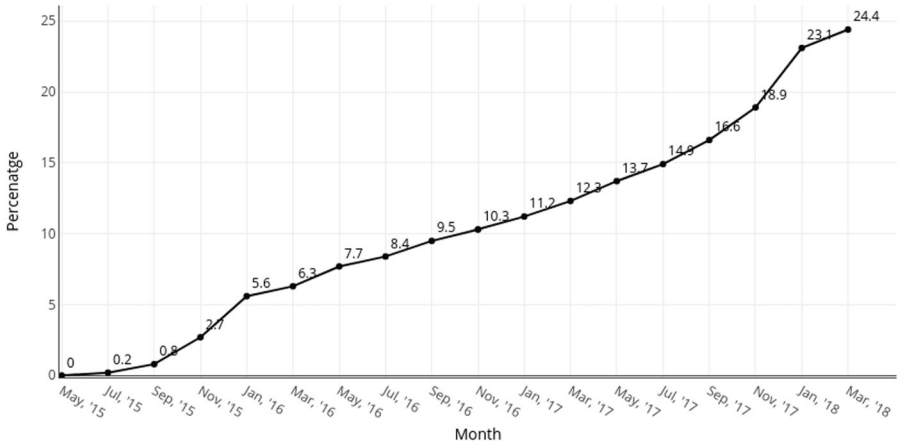

Figure 4 Historical Trends.

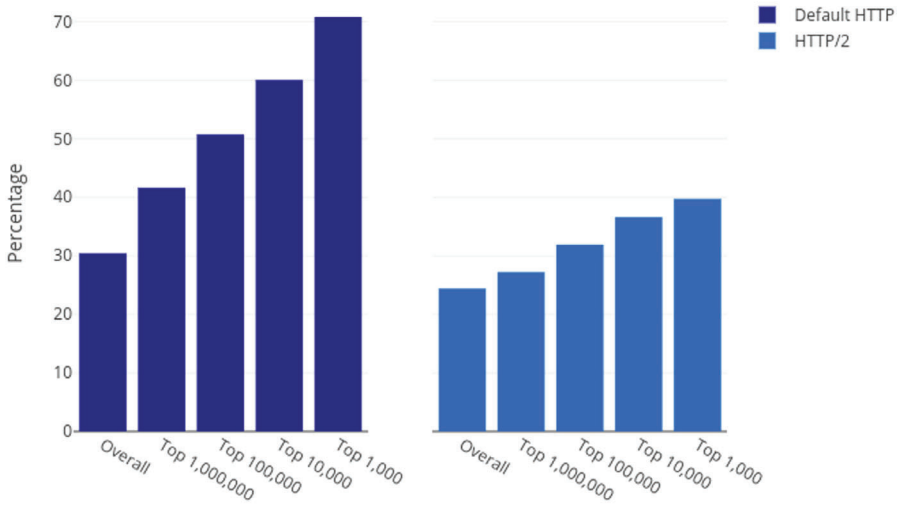

Figure 5 HTTP/2 traffic - April, 2016.

If the server does NOT support HTTP/2 then there is no "Upgrade" header present in the server response otherwise it will respond with a 101 status code for Switching Protocols response.

The h2 version is the most important one that pertains to the security factor that is required. H2 is used to implement connection over a secure channel. A client that makes a request to an "https" URI uses TLS with the application-layer protocol negotiation (ALPN) extension. The ALPN extension seeks to

evade an additional network round-trip by bundling the protocol negotiation with the exchange of hello messages. As a component of the initial request (ClientHello), the client provides a list of protocols which it fortifies. The server then selects a protocol from that list and sends that selection back to the client along with its part of the TLS handshake (ServerHello). For antecedently established connections, the protocol selection can be negotiated over an abbreviated hand shake. But the issue here is that even if HTTP/2 is secure, it leaves traces behind for a malicious attacker to sniff about the information. This entire paper is all about the above addressed issue and the analysis of it.

6 Contribution

The behavior of HTTP/2 is studied by collecting the network traffic and analyzing it. This study helps to understand the security aspects of the protocol. In this paper, we have explained and demonstrated the feasibility of decrypting an encrypted HTTP/2 traffic with SSLKEYLOGGING feature. Analysis of our own network traffic is done for this purpose.

6.1 Test Environment

In-order to test the HTTP/2 protocol and to figure out the implications of the results, a specific environment is created with the required operating system and softwares needed with it.

Kali Linux has been used as the operating system for the monitoring. Version 2017.2 of the same is used. The Kali image is updated and upgraded with the default libraries and packages after installation. For network traffic analysis Wireshark [15], a free and open-source packet analyzer that captures and dissects network traffic is used and a version of 2.4.2 is installed. Mozilla Firefox Developer edition, 58.0b5 version, is also installed for the monitoring the traffic as it provides a better flexibility to the management of data.

6.2 Decrypting the Sessions

There a variety of encryption algorithms available like DES, AES, IDEA and RSA which helps to encrypt the data transfered between the client and Server [21]. The servers private key is required for decrypting an encrypted traffic. Browsers also have the ability to log these keys into a file called SSLKEYLOGFILE [12] with which the decryption procedure becomes easy.

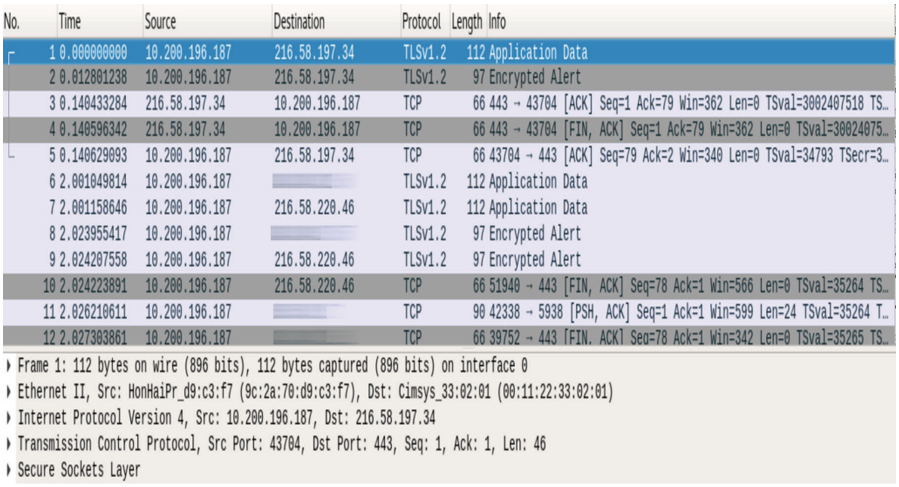

Figure 6 Hidden HTTP/2 packets.

Here the SSLKEYLOGFILE is used for the session decryption. HTTP/2 packets are by default hidden in Wireshark, when the network trace is analyzed. It is because of the encryption and the compression feature of he protocol. Figure 6 shows the trace of the network where the HTTP/2 packets are hidden, i.e: before assigning the SSLKEYLOGFILE for session decryption. These log files are generated, by default, each time a browser is used, so the Firefox browser is used to browse through HTTP/2 utilizing websites. Once the log files are created, they are exported to another locally accessible log file using the subsequent command.

```
$ export SSLKEYLOGFILE=~/path/to/sslkeylog.log
```

This command generates an auto updating log file which contains the symmetric session keys written in NSS Key Log Format[1]. This format is supported by Wireshark and can also be used in them. These log files should be used as the Pre-Master Secret log in Wireshark by setting the protocol preference to SSL. Once this is done then the traffic has to be captured which will have the earlier invisible HTTP/2 traffic. Figure 7 shows the HTTP/2 packets that were captured from the network when the browser was used. Dissecting the HTTP/2 packet headers will show the headers and the protocol features used by HTTP/2 Figure 8 provides a deeper an in-depth insight into the packet

[1]Network Security Services.

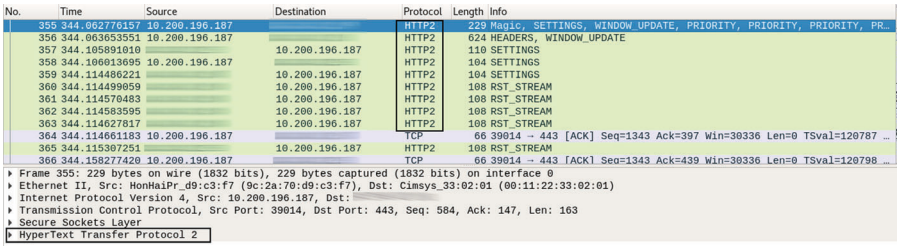

Figure 7 HTTP/2 Traced.

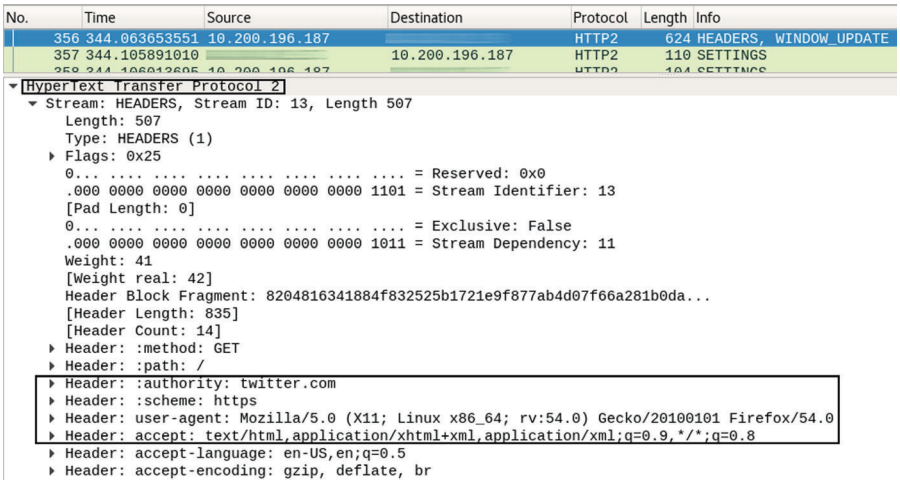

Figure 8 Header, Handshakes and Packets.

and frame structure of HTTP/2 protocol for one random HTTP/2 packet. This figure also shows the websites visited, OS used along with the version of the browser too from which the network traffic was generated. So this explains the fact that, the details which we assume to be hidden (like the sites visited) due to the compression and encryption techniques, can be found out with the help of this method.

7 Pitfalls in HTTP/2

The quest for all the loopholes in HTTP/2 led to other vulnerabilities or the attacks that can be possible [8].

- HTTP/2 Stream Multiplexing: A single TCP connection is used to manage multiple streams. There is risk in this feature because of the fact that the connection partitions are completely logical and can be manipulated to send out-of-context frames [7, 8, 18]. This can also lead to Denial-Of-Service issues too.
- Slow Read Attacks: It utilizes all the open threads by opening as many connections as possible in the victims server, thereby leading to a Denial of Service (DoS) situation [7, 8, 19].
- Dependency and Priority: These new features when assessed from a security point-of-view may not fulfill all the necessary requirements of it. The size of the dependency tree that can stored in the memory is not limited, therefore unnecessary wastage of memory can also happen [7, 8].
- Head-of-The-Line Blocking: Even though HOL Blocking persistent in the application level in HTTP/1.1 was solved in the version 2, another type of HOL is present in HTTP/2. This one is over the TCP level, where one lost packet in the TCP stream can make all the streams to wait until that package is re-transmitted and received.

8 Result and Observations

Even though HTTP/2 is known to be a very secure version, this procedure of key-logging will enable a malicious attacker to find out basic informations about the user. The attacker can find out details like the website visited, the OS of the machine and even the browser being used. An HTTP/2 packet which is capture in Wireshark contains 3 types of data fields the basic frame, the Decompressed SSL Data and the Decompressed Header. The frame section shows the encrypted data which corresponds to the selected packet from the capture. The Decompressed SSL Data shows the decrypted traffic, with one layer of encryption removed. Here the TLS encryption is removed. The final field is the Decompressed Header and it gives the completely decrypted data of the packet. Figure 9 shows the decompressed header of an HTTP/2 packet which contains data like website visited, the OS of the machine and the browser being used. When an attacker receives this kind of information, it becomes easy for them to monitor the user information and his actions.

So deeper analysis of HTTP/2, will lead to more vulnerabilities being found. These cracks are not found fatal yet, but can be used to take the whole security aspect sideways.

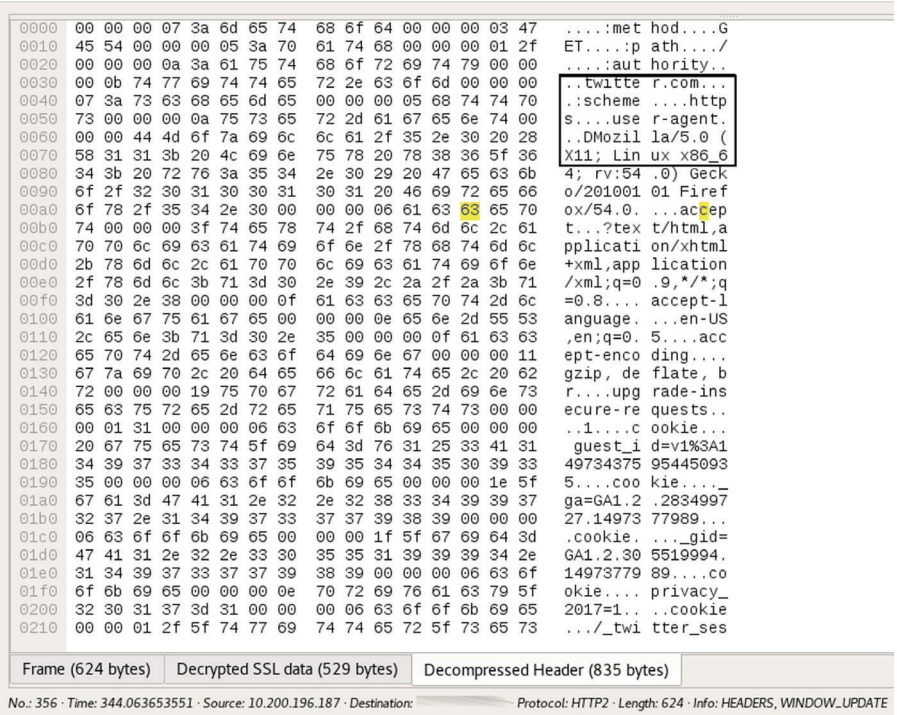

Figure 9 Decompressed Header of an HTTP/2 packet.

9 Conclusion

In the future, the network traffic will be seen as a mixture of both HTTP/1.1 as well as HTTP/2, as all web browsers and servers have not ported their sites to support the new protocol. This will lead to an increase in the susceptibility towards vulnerability. HTTP/2 being a protocol which is backward compatible, its 2 versions also will cause additional tendency to jeopardize the security. Since the traffic would be a mixture of both the protocols, vulnerabilities from both the sides would be seen possible on one another. So, sooner the better, the network traffic can be ported completely to the h2 as it is going to be the future of Internet world till something new is coming up. Even though TLS is not made compulsory in HTTP/2, the client implementations, as mentioned earlier, has decided to support h2 over TLS only which will lead to the encryption being a mandatory factor in network traversals. This will, furthermore, make any sort of network forensics a complex and tedious job.

10 Future Works

What next after HTTP/2, is something that always goes through the minds of researchers or how can this protocol be used to enhance the security of the networks. The vulnerabilities provided here regarding this protocol is the basic analysis. When worked in detail on each of the pitfalls mentioned, it all leads to a common factor of DoS. More research is required on this aspect of the protocol as it can deeply affect the quality of service provided.

The TCP HOL blocking issues can be addressed by using another protocol called QUIC [20, 22]. This is a TCP-like protocol that is implemented over a UDP connection where each stream is independent so that a lost packet only stops the particular stream to which that lost packet belongs, while the other streams can go on trouble-free. Trying out the same attacks that were performed on HTTP/2 protocol over QUIC to understand how differently they react to the same situations can provide a better understanding on deciding which protocol will prevail in future and also to know which can provide the best security available [22].

References

[1] Berners-Lee, T., Fielding, R., and Frystyk, H. (1996). Hypertext transfer protocol–HTTP/1.0 (No. RFC 1945).

[2] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext transfer protocol–HTTP/1.1 (No. RFC 2616).

[3] Rescorla, E. (2000). RFC 2818, HTTP Over TLS. Internet Engineering Task Force. May 2000.

[4] Hodges, J., Jackson, C., and Barth, A. (2012). Http strict transport security (hsts) (No. RFC 6797).

[5] Belshe, M., Thomson, M., and Peon, R. (2015). Hypertext transfer protocol version 2 (http/2).

[6] Peon, R., and Ruellan. H. (2015). RFC 7541, HPACK: Header Compression for HTTP/2. Internet Engineering Task Force. May 2015.

[7] Adi, E. (2017). Denial-of-service attack modelling and detection for HTTP/2 services. Doctorates and Masters Theses, Edith Cowan University, 2017.

[8] Hacker Intelligent Initiative HTTP/2:In-depth analysis of the top four flaws of the next generation web protocol. Red Hat conference Publication By Imperva Defense Center, 2016.

[9] Tripathi, N., and Hubballi, N. (2018). Slow rate denial of service attacks against HTTP/2 and detection. Computers & Security, 72, 255–272.

[10] Corbel, R., Stephan, E., and Omnes, N. (2016, July). HTTP/1.1 pipelining vs HTTP2 in-the-clear: Performance comparison. In *2016 13th International Conference on New Technologies for Distributed Systems (NOTERE)*, (pp. 1–6).

[11] Jackson, B. (2017). HTTP/2 Statistics KeyCDN Report on HTTP/2 Distribution. Available at: https://www.keycdn.com/blog/http2-statistics/

[12] Winkel, S. (2015). Network Forensics and HTTP/2 SANS Institute InfoSec Reading Room, December 2015.

[13] Binu, P. K., Sreekutty, H. L., and Sreekutty, V. S. (2016). Security plugin for Mozilla which integrates cryptography and steganography features. In *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, (pp. 1–6).

[14] de Saxcé, H., Oprescu, I., and Chen, Y. (2015). Is HTTP/2 really faster than HTTP/1.1. In *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, (pp. 293–299).

[15] Wang, S., Xu, D., and Yan, S. (2010). Analysis and application of Wireshark in TCP/IP protocol teaching. In *2010 International Conference on E-Health Networking, Digital Ecosystems and Technologies (EDT)*, (Vol. 2, pp. 269–272).

[16] Newmarch, J. (2017). *Network Programming with Go: Essential Skills for Using and Securing Networks*. Apress. (8)137–160, 2017.

[17] Varvello, M., Schomp, K., Naylor, D., Blackburn, J., Finamore, A., and Papagiannaki, K. (2016). Is the web http/2 yet?. In *International Conference on Passive and Active Network Measurement* (pp. 218–232). Springer, Cham.

[18] CVE-2016-0150. In. Vulnerability Information HTTP.sys Denial of Service Vulnerability, 2016

[19] CVE-2016-1546 In. low: mod http2: denial of service by thread starvation, 2016.

[20] Cui, Y., Li, T., Liu, C., Wang, X., and K'hlewind, M. (2017). Innovating transport with QUIC: Design approaches and research challenges. *IEEE Internet Computing*, 21(2), 72–76.

[21] Sreedhanya, A. V., and Soman, K. P. (2012). Secrecy of cryptography with compressed sensing. In *2012 International Conference on Advances in Computing and Communications (ICACC)*, (pp. 207–210).

[22] Bakri, H., Allison, C., Miller, A., and Oliver, I. (2015). HTTP/2 and QUIC for Virtual Worlds and the 3D Web. *Procedia Computer Science*, 56, 242–251.

Biographies

Meenakshi Suresh is pursuing her M.Tech in Cyber Security from Amrita University and will graduate in 2018. She attended TocH Institute Of Science and Technology from where she received her B.Tech in Computer Science. Her current area of research is Networks and Security based on the protocols.

P. P. Amritha received her M.Tech. in Cyber Security from Amrita University. She is now a PhD scholar at Amrita University. Her current research interests include: Steganography and Code Obfuscation.

Ashok Kumar Mohan, M. Tech specialized in Cyber Security, is a Research Associate at TIFAC-CORE in Cyber Security, Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu, India. He is currently a PhD scholar doing his research in the area of Cyber Forensics funded by Ministry of Electronics & Information Technology (Government of India) under Visvesvaraya PhD scheme for Electronics and IT. He is currently pursuing his research over the

cyber security core vicinity in Metadata Forensics, Wireless Security Auditing, Rumor Prediction in Social Media Networks and Slack Space Analysis of NTFS File Systems. He is also the Certified EC-Council Instructor (CEI) for ethical hacking and penetration testing certification courses at the research centre.

V. Anil Kumar is a Principal Scientist at CSIR Fourth Paradigm Institute (CSIR-4PI), Bangalore, India. His research interests are Cyber Security, High Performance Computing and Protocol Engineering. He has about 25 research papers in international journals and conference proceedings. He has filed one International and two Indian patents on security aspects of transport protocols. He received DAAD Fellowship from German Academic Exchange Service and subsequently worked at Fraunhofer Institute for Open Communication system, Germany during 2002–2004. From 2009 to 2010, he was with French National Research Institute in Computer Science and Control (INRIA), France as Senior Expert Engineer. He has worked on a large European Union Project called OneLab2 to establish a geographically distributed and federated network testbed. He was one of the project leaders for establishing a supercomputing facility, which was ranked no. 1 in India and 58th fastest in the world, as per the June 2012 list of top500 supercomputers. He also received Internet Society (ISOC) to participate in the 95th Internet Engineering Task Force (IETF) meeting.