OULU UNIVERSITY OF
APPLIED SCIENCES

Ali Jawad

THE FUNDAMENTALS OF HTTP/2

THE FUNDAMENTALS OF HTTP/2

Ali Jawad
Bachelor's Thesis
June 2016
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme, Option of Internet Services

Author: Ali Jawad
Title of the bachelor's thesis: Fundamentals Of HTTP/2
Supervisor: Teemu Korpela
Term and year of completion: June 2016 Number of pages: 31

The purpose of this Bachelor's thesis was to research and study the new version of HTTP "HTTP2.0", which is considered to be the future of the web. Http/2 is drawing a great attention from the web industry. Most of the Http/2 features are inherited from SPDY.

This thesis shows how HTTP/2 enables a more efficient use of network resources and a reduced perception of latency by introducing a header field compression and allowing multiple concurrent exchanges on the same connection "multiplexing" and more other features.

Also, it discusses the security of Http/2 and the new risks and dangerous attacks that resurfaces with the arrival of this new protocol version.

The simulation results show how HTTP/2 influences the page load time comparing to the other previous versions of HTTP.

Keywords: HTTP1, HTTP/2, SPDY, SNI, DOS, CRIME, Downgrade-attack.

PREFACE

This thesis was written for Oulu University of Applied Sciences and done during 1 February – 23 May 2016. The role of the instructor was guiding the thesis from the requirements and bases of writing a thesis document through meetings. The role of the supervisor was instructing the thesis plan and its requirements which were done by the author. I would like to thank my thesis instructor Teemu Korpela for instructing and guiding my thesis and his great influence and support on my thesis which has resulted to a success.

Oulu, 23.5.2016
Ali Jawad

CONTENTS

VOCABULARY

- **HTTP**: Hypertext Transfer Protocol.
- **SPDY**: Pronounced speedy.
- **SSL**: Secure Sockets Layer
- **TLS**: Transport Layer Security.
- **TCP**: Transmission Control Protocol.

- **SNI**: Server Name Indication.
- **CRIME**: Compression Ratio Info-leak Made Easy.
- **SSH**: Secure Socket Shell.
- **DOS**: Denial-of-service.

1 INTRODUCTION

1.1 What is HTTP?

HTTP stands for a Hyper Text Transfer Protocol. It is a protocol that is used by the World Wide Web. HTTP is responsible for how messages are formatted and transmitted, in other words HTTP requests information from a server and displays web pages on the screen. HTTP runs on the top of the TCP/IP suite of protocols i.e. the main protocols for the Internet. For example, when the user enters a URL in the web browser, it sends an HTTP command to the Web server that is directing it and it fetches and transmits the target or requested Web page. This, as the user opens the web browser, the user uses HTTP indirectly. *(8)*

1.2 What is HTTP/2?

HTTP/2 was developed by the IETF's HTTP Working Group, it is made up of a number of HTTP implementers and HTTP experts. HTTP/2 is the upgraded version of the HyperText Protocol, that handles the connections between a web server and the browser. That means that web pages will load quickly, connections will last longer and servers will respond to the requests with a more content.

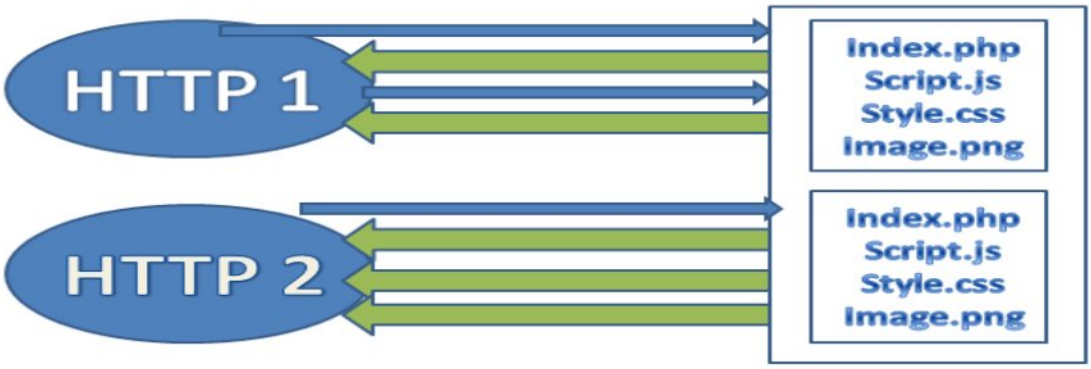

FIGURE 1. Loading pages of HTTP/1 & HTTP/2.

With time, web pages have steadily increased in size and content, which means more requests are sent out. Here HTTP/2 comes to offer a solution to improve page load speed and a decrease latency by enabling a full request and response multiplexing. HTTP/2 is supported by Firefox and Chrome. Other browsers that are based upon Blink (e.g Opera and Yandex Browser) will support HTTP/2, too. *(15)*

2 FEATURES OF HTTP/2

2.1 Key Features

HTTP/2 has the following key features compared to HTTP/1.1:

-Binary Protocol

HTTP/2 is a binary protocol. Binary protocols are more efficient to parse, this means that it is much more efficient on the wire. HTTP/2 defines just one code path to parse message, while HTTP/1.x defines different ways. This new binary framing layer is responsible for the message encapsulation and transferring between a server and a client, which means that both the server and the client should have this new binary mechanism to understand each other.

An ID called as Stream ID is given to every HTTP request and response which are divided into frames. Frames are binary data portions.

A client makes an HTTP request by dividing the request into binary frames and assigning a Stream ID to each frame. Then the client sends the frames to the server after initiating a TCP connection. When the server has the response ready, it divides it into frames and gives as the response frames the same response Stream ID. The server sends the response in frames.

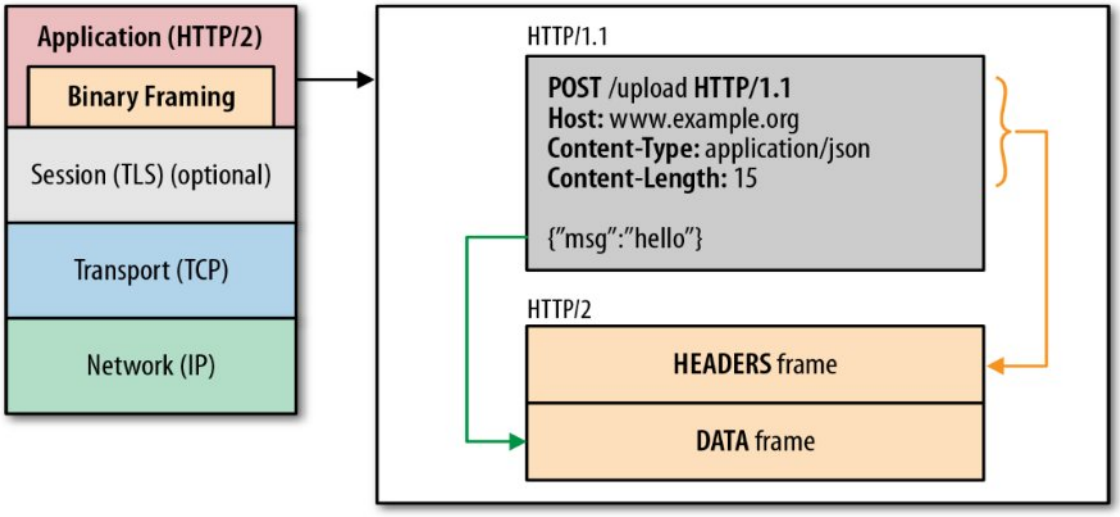

FIGURE 2. HTTP/2 binary framing layer. (16)

- Multiplexing:

HTTP/2 supports multiplexing several streams over a single connection. This means that a client can send multiple requests on the same connection, and the server can respond, starting from available responses. In Figure 2, we can see how the browser receives the response headers for file #3, and then it receives the response body for file #1. Next it starts getting the response body for file #3, before continuing on to file #2.

As mentioned above in the binary protocol, a Stream ID is necessary in multiplexing. Because multiple requests to an origin are made using a single TCP connection, so a Stream ID makes it possible to identify to which request or response a frame belongs to.

In multiplexing, Multiple HTTP/2 requests are divided into frames and assigned with Stream IDs. All the frames from multiple streams are sent non-synchronously. And the server also sends responses nonsynchronously. In case some responses take a long time to finish, then other responses do not have to wait. The client receives the frames and arranges them according to their Stream ID. *(11)*

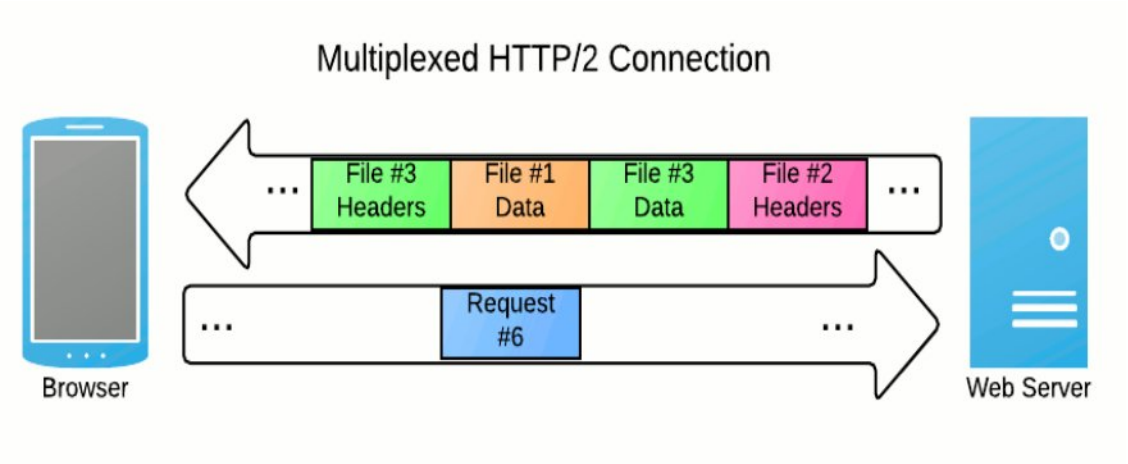

FIGURE 3. HTTP/2 request and response multiplexing within a shared connection. (2)

-Header Compression:

HTTP/2 uses an HPACK header compression in order to compress a header. HTTP headers can sometimes be bigger than the body of the request. Also, HTTP requests and responses have a large number of redundant headers. In other words, HTTP data is compressed before it is sent from the server. Compressing headers both reduces the overhead of additional requests and of introducing new headers.

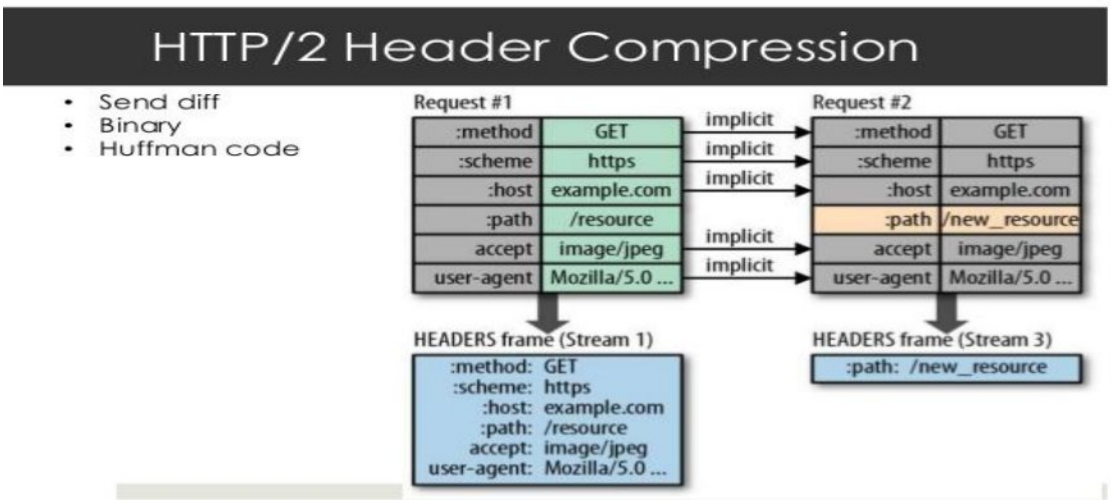

FIGURE 4. HTTP/2 Header Compression. (13)

- Server Push:

A Server push allows a server to send additional cacheable resources to the client so that the client is not explicitly asked for to do it. The Server Push allows the server to send a 'request promise' and an accompanying response to the client. This allows the server to anticipate the resources the client will request next, which saves a round trip. In other words, the Server Push feature allows the server to send the page resources immediately to the client without the need to parse and find out the resources of the page, which speeds up the page loading time.

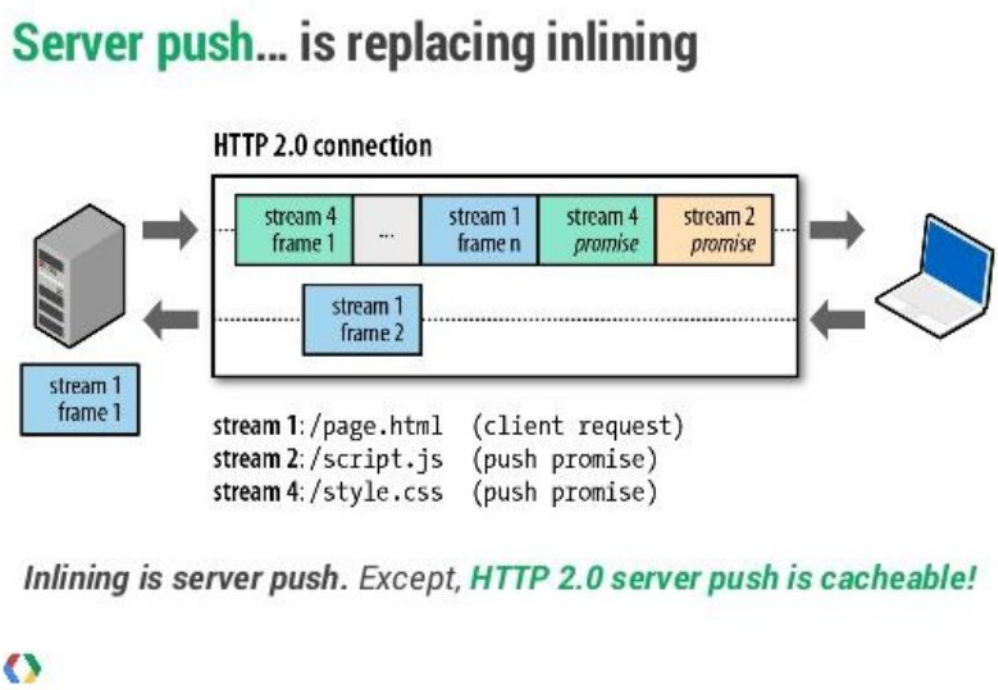

FIGURE 5. Server Push is Cacheable. (17)

2.2 The future of HTTP/2 and the most promising payoffs

HTTP/2 comes to solve many defects of previous versions of HTTP, HTTP/1.1 and HTTP/1.0.

With HTTP/2 we can make sure that web pages or applications will work/perform better and with fewer resources on both the client and server, including more benefits like:

- Encrypting: Web pages and Applications running over HTTP/2 have a better performance over secure connections
- Optimizing the TCP layer: In response to packet loss, the Applications should be designed with a TCP layer implemented to account for the switch from multiple TCP connections to a single long-lived one.
- Undoing HTTP/1.1 best practices: Many “best practices” delivered over HTTP/1.1 (e.g. image spriting) are not necessary sub-improvements.
- Deciding what and when to push: Applications should balance performance and utility over the new server push capabilities in HTTP/2. *(1)*

2.3 SPDY

SPDY (pronounced speedy), is a secure web transport protocol that makes the page load faster by multiplexing many transactions onto one TLS connection which means using fewer TCP connections to transport a web content. *(12)*

SPDY is not a replacement for HTTP, rather it modifies HTTP by reducing Latency by making multiple requests and responses over a single connection and by improving the security by using TLS. SPDY has been used as a base for HTTP/2 and the core developers of SPDY have been involved in developing HTTP/2. The latency will drop if SPDY is replaced by HTTP/2, instead the usage of both will ensure that users will continue to get the best possible experience. But that does not mean that HTTP/2 without SPDY is impossible! Usually, people they treat SPDY and HTTP/2 as the same thing, but that is not the case. Http/2 has started from SPDY and developed from there to create a new standard protocol which is incompatible with SPDY and HTTP/1 but close to them. *(27)*

2.4 Comparison of HTTP/2 and SPDY

TABLE 1.Comparison of Http2 & SPDY. (10)

SPDY	**HTTP/2**
SPDY uses the general purpose DEFLATE algorithm.	HTTP/2 uses HPACK which was specifically designed to compress headers.
Small response messages	Large response messages
SPDY does not use the ALPN extension.	HTTP/2 uses ALPN extension to avoid an additional network round-trip, which makes faster encrypted connections.
Multiplexing can happen just on one host at a time.	Multiplexing can happen on multiple hosts at the same time.
Slower Page Load.	Faster Page Load.
Unsecure Compression. SPDY leaves vulnerabilities in its current compression methods.	Secure Compression**.** HTTP/2 uses HPACK to prevent vulnerabilities.
Prioritization.	Improved Prioritization.

SPDY requires SSL to use the protocol and get the benefits.	HTTP/2 does not require SSL neither TLS.

3 HTTP/2 SECURITY

3.1 HTTP/2 safety

Implementing TLS with HTTP/2 will increase safety. HTTP/2 with TLS 1.3 or higher needs only support to the Server Name indication (SNI). SNI indicates and matches the name of the certificate with the name of the page to ensure security and safety. For example when you visit a website, the connection is made with a web server by providing the name and the domain name of the webpage. But when making an TLS connection, the browser requests a digital certificate from the web server in order to match the name of the certificate with the name of the page which you are trying to make connection with. If they do not match, then you will not be able to access the website and you will get a warning message. Also, it may indicate a man-in-middle-attack.

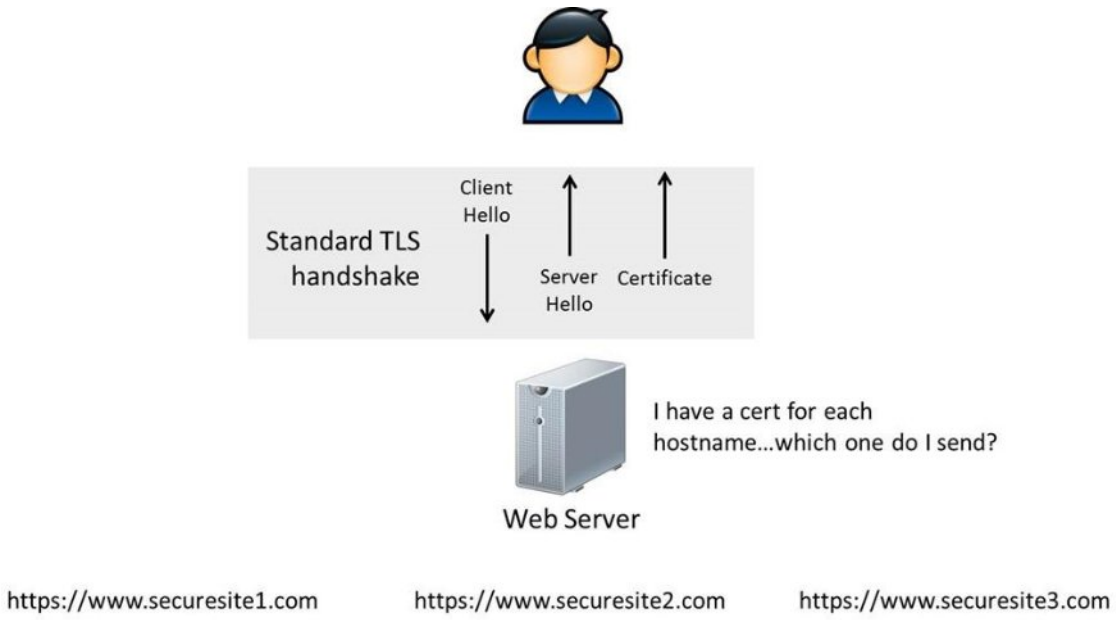

FIGURE 6. Server Name Indication. (25)

- In HTTP/2, cipher suites (i.e encryption algorithms used by hosts to establish a secure communication) should be AEAD. AEAD stands for Authenticated Encryption with Associated Data which describes several modes of operation to provide integrity, authenticity and confidentiality on data which increase safety.

  Compression in HTTP/2 is unnecessary as HTTP/2 provides secure compression features in which it forces all HTTP headers to be sent in a compressed format, thus reducing the amount of information that needs to be exchanged between the server and the browser.
- Http2 over TLS 1.2 must disable a renegotiation, because HTTP/2 has specific TLS requirements and it will be using a safer implementation of TLS by preventing *session renegotiation* attacks. This discovered vulnerability could be used to manipulate the data received by a client or by a server. *(22)*

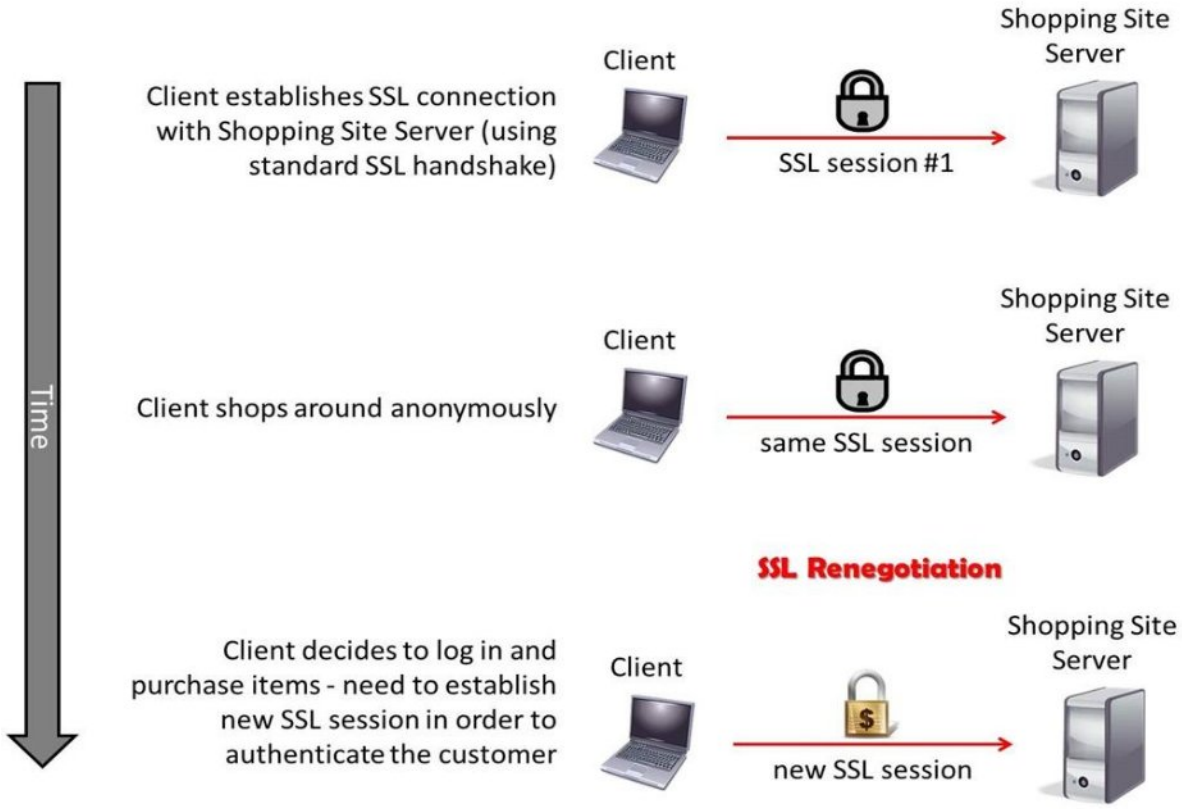

FIGURE 7. Renegotiation. (26)

3.2 New Risks of HTTP/2

The new key features of HTTP/2 compared to HTTP/1 has improved performance, but at the same time it has an introduced additional attack surface to the HTTP implementation.

3.2.1 HPACK

Probing Dynamic Table State:

Hpack reduces the length of header field encodings, which reduce the amount of data that is required to send HTTP requests and responses. The attacker can probe the compression context used to encode the header fields in order to define the header field and observe the length of these fields. This makes the attacker modify the requests in order to confirm guesses about the dynamic table state. A dynamic table or an Array is a random access, variable-size list data structure that allows elements to be added or removed. This kind of attack is also possible over the Transport Layer Security (TLS), since it provides a limited protection for the length of the content.

Applicability to HPACK and HTTP:

Attacks modeled on CRIME force the guess to match the entire header field value rather than individual characters. This attack can happen anytime that two distrustful entities control requests and response over one single HTTP/2 connection. For example, when an intermediary sends requests from multiple clients on a single connection toward an origin server or it takes responses from multiple origin servers and place them on a shared connection toward a client. *(18)*

3.2.2 CRIME

Crime, stands for a Compression Ratio Info-leak Made Easy. It exploits the data compression scheme over the TLS and SPDY Protocol in order to decrypt user authentication cookies from HTTPs (HTTPS Secure) traffic by means of brute-force, which makes the attacker perform a session hijacking.

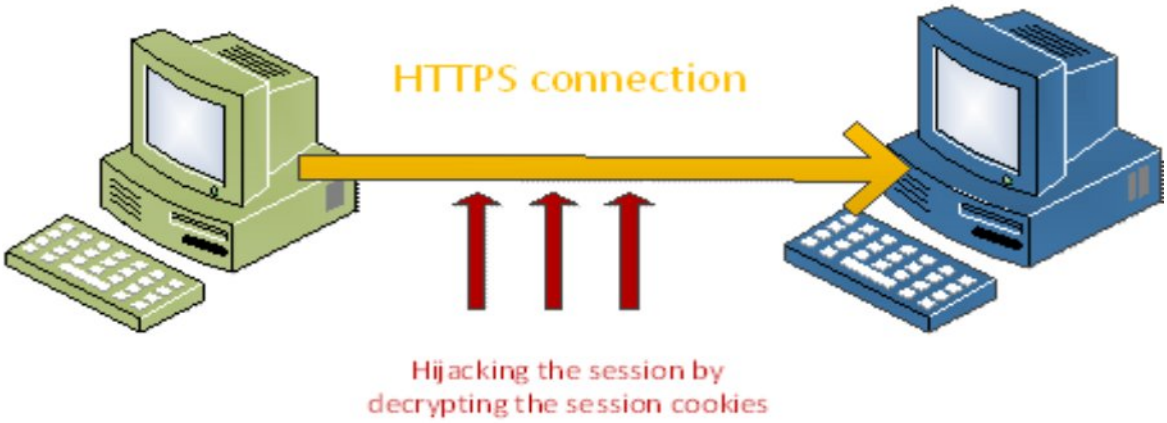

FIGURE 8. IT Security Training & Resources by InfoSec Institute. (19)

This occurs when the attack code of the attacker forces the victim's browser to send HTTPS requests to a targeted website to determine the value of the victim's session cookie. Since the attack code can't read the session included in the request because of the security mechanism in the browsers. CRIME is also possible over SSL and TLS because they use also compressions (DEFLATE).

However, we can still prevent CRIME Attack by disabling or preventing the use of the compressions by for example disabling the compression of SPDY requests or by using the protocol negotiation features of the TLS protocol to prevent the use of the data compression. *(9)*

3.2.3 Downgrade-attack

Downgrade-attacks have been always a problem with SSL/TLS family protocols. Downgrade attack is a crack method that tries to downgrade an encrypted connection to make it easier to be exploited, by other words it abandons a high quality mode or operation to an old lower quality mode. Usually, these kinds of attacks are often implemented as part of a man-in-the- middle attack by enabling cryptographic attack.

Below in figure 9, we can see an example of an active downgrade attack, where Firefox and Chrome are both switched to lower SSLv3 connections. Opera was not susceptible to the active attack. This attack was tested while connecting to Facebook using the latest versions of Firefox, Chrome and Opera.

FIGURE 9. TLSv1.2 Vs SSLv3 in Browsers. (23)

Another downgrade attack is SSH Man-in-the-Middle downgrade famous example, in which the Man in the middle or attacker forces the client and the server to use insecure SSH1 rather than SSH2, which is a more secure protocol. The attacker modifies the answer of the server which tricks the client and makes him think that the server supports only SSH1, thus forcing the client to open an SSH1 link. And in this case, the attacker will get the personal information of the client such as passwords in login due to the weak authentication mechanism.

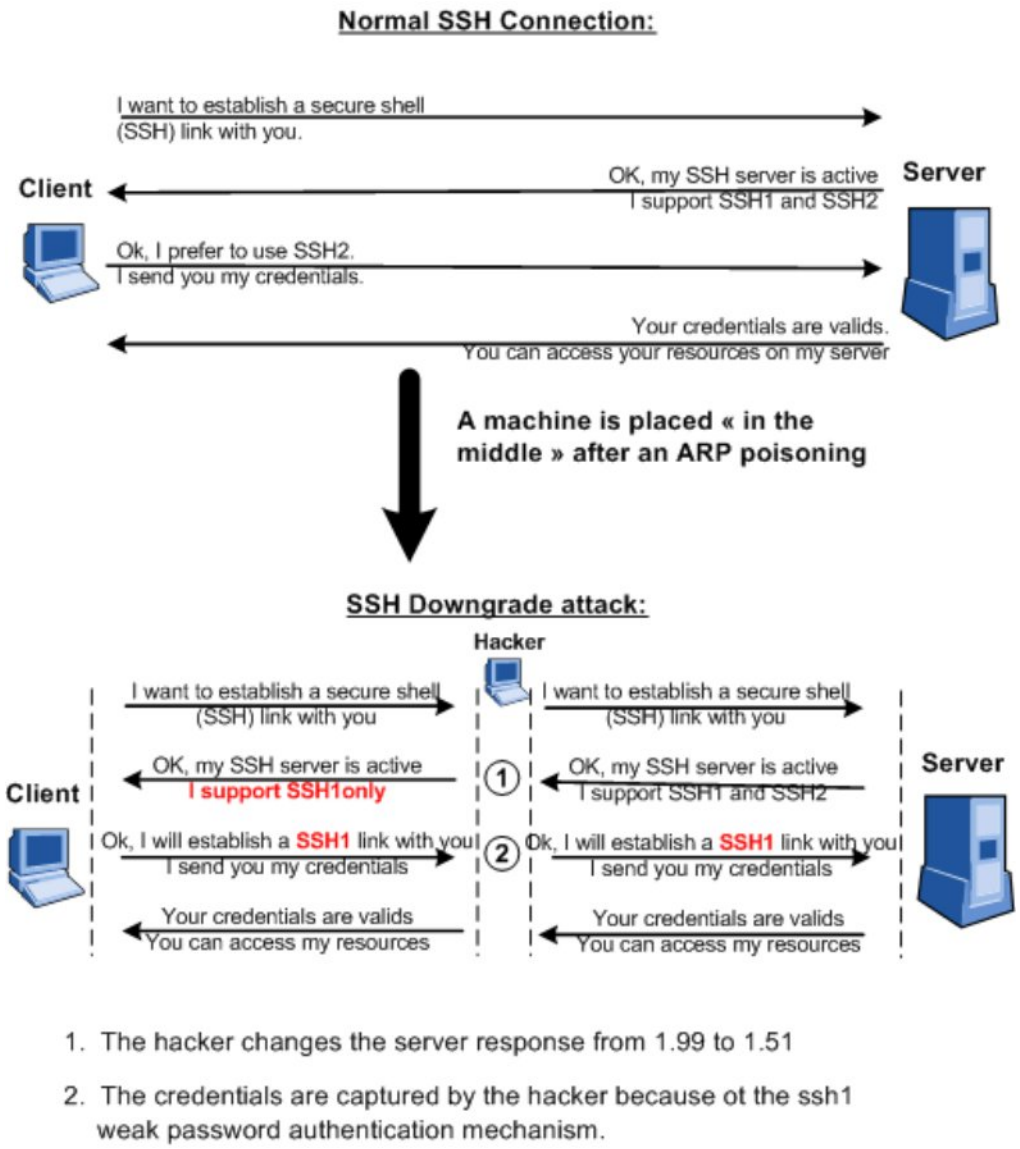

FIGURE 10. SSH Downgrade-attack. (24)

TLS has allowed the attackers or the Man in middle to force the client to use a downgrade attack to a weaker mechanism having an algorithm, like MD5. TLS has introduced a new Signature And Hash Algorithm field in the *Server* Key Exchange *message to allow the server to specify which signature and hash* algorithms the client must use. However, removing a backward compatibility is often the only way to prevent downgrade attacks. (See figure 10). *(24)*

3.2.4 Malformed Frames

There are many vulnerabilities in Mozilla Firefox that could allow an unauthenticated attacker to use a DOS attack (Denial Of Service).

This kind of vulnerability in HTTP/2 is due to the improper handling of malformed framed by an affected software. With this vulnerability, the attacker is able to exploit it by letting the user or client to follow a malicious link by misleading instructions. A successful exploit could trigger an interger underflow condition that will make the browser abort resulting in Dos Attack. Mozilla confirmed the vulnerabilities in a security advisory and released software updates.

Dos stands for Denial of Service. It is an attack method in which the hacker attempts to prevent users from accessing into the service/system. In a Dos attack, the hacker sends a lot of messages asking the server to authenticate requests that have invalid return addresses. This causes the server to wait before closing the connection since the server cannot find the return addresses which the hacker is sending. Thus, when the connection closes, the hacker continues sending more authentication messages with invalid return addresses in order to make the server wait to begin again, keeping the server busy.

For example, the hacker can attack a website and disrupt its normal functions by sending a flood of messages or an overload requests to the website that will slow down the website's response time and its data handling capacity which results in a system crash. Dos is a criminal offense even if attempted as a prank. *(20)*

In figure 11 below, the attacker is overloading the target server with requests which exceed the target server's capacity causing the server to slow down.

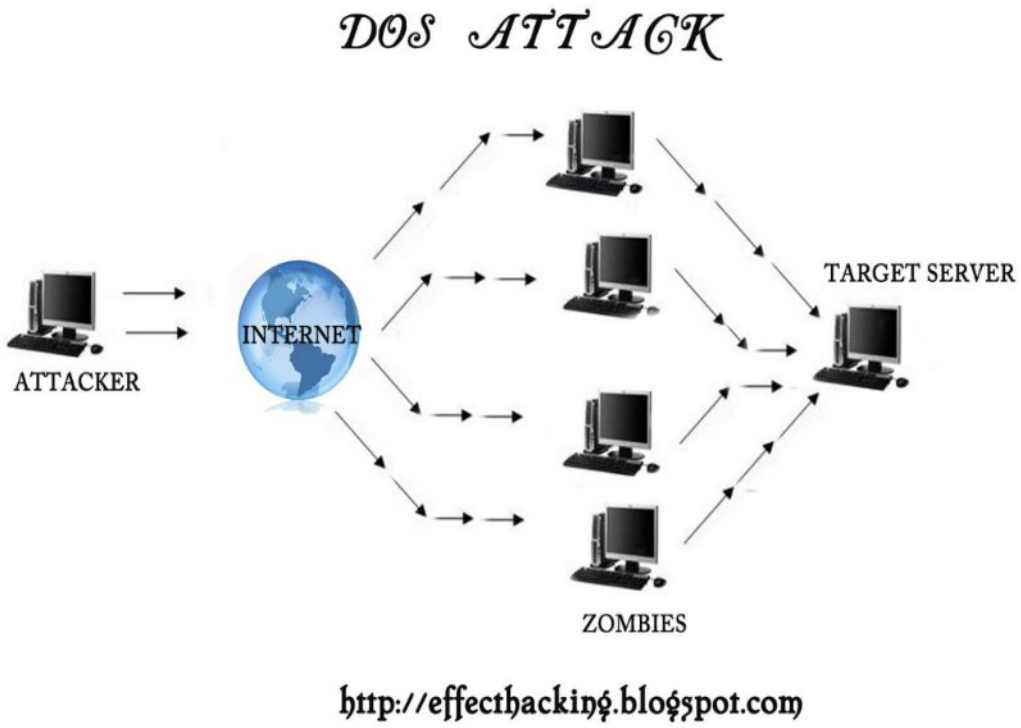

FIGURE 11. Denial of Service Attack. (21)

3.3 How Dos attacks can be launched against an HTTP/2 server

A study investigated if a given HTTP/2 server shows the effects of resource consumption upon the exploitation of the vulnerability of the protocol presented in the previous section. Four parameters are being monitored: the CPU usage, memory consumption, network throughput, and packet loss. When a computing resource is under stress, the network throughput can go down, while the rest of the indicators can go up. Attacks are, by itself not legitimate traffic.

Hence, the desired malicious HTTP/2 packets based on several parameter values to model the attack were crafted and a packet generator at the client-end was implemented.

A modified client implementation was used in a way that it sends crafted HTTP/2 packets representing the attack.

A logging tool collectl was used in this study to monitor the first three parameters mentioned above; the tool captured the CPU usage (in percentage), memory consumption (in MB free), and network throughput (in KB per second and the number of packets per second).

Three students in School of Computer and Security in Austria have used Collectl for data transfer and analysis as it generates accurate results at the cost of a low processing overhead. To monitor the fourth parameter, packet loss, a series of ICMP ping messages were sent to the server during the attack.

Pinging has showed the percentage of packets lost when the command is stopped abruptly. To test if a Dos attack was successful, a simple page request was sent from a second client terminal to the server.

When a server resource is under a Dos attack, the server does not send a response page to the client. A simple page would contain a simple ‘Hello World’ message in its HTML tagged.

On the other hand, investigations were conducted through this study, in order to answer the following questions: how Dos attacks can be launched against an HTTP/2 server; how many servers can a single malicious client attack simultaneously; and how can a time-delay factor in attack traffic make the attack stealthier.

Investigation 1: The Attack The first investigation was to examine out how a Dos attack can be launched against an HTTP/2 server. The following lab configuration was set up. Two VMware Player virtual machines were used: one hosted a client (as the attacker) and the other hosted a server (as the victim).

Each virtual machine was configured to have 1 processor core with 1 GB RAM, and ran the Ubuntu 14.10 Linux distribution. The client and the server were connected through a 100 Mbps virtual network. The two virtual machines were run on the same host machine, with Windows 7 Enterprise and 4GB RAM. Results can be found in the figure12.

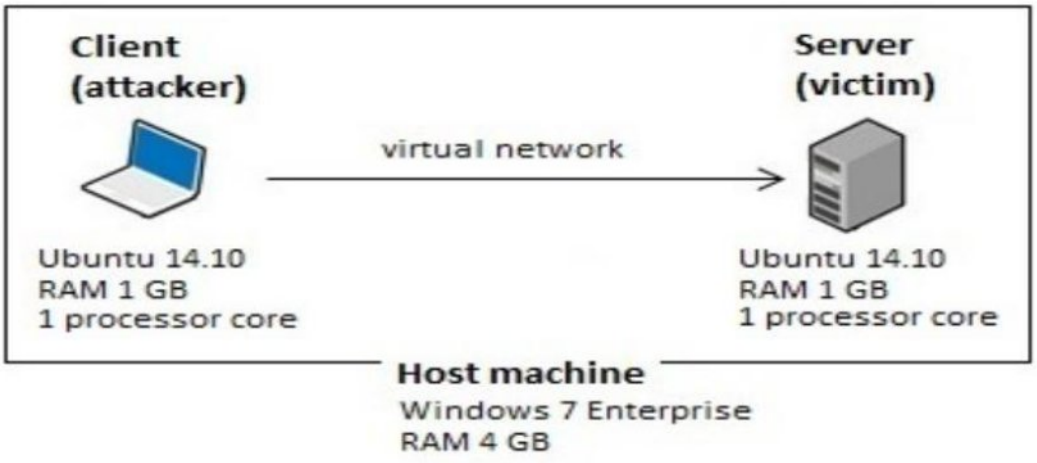

FIGURE 12. The Configuration of the lab for investigation 1 and 3. (3)

Test Case	CPU		KBIn		PktIn	
	ave	*s.d.*	*ave*	*s.d.*	*ave*	*s.d.*
1	98.56	6.29	403.98	45.67	274.06	31.81
2	94.80	17.23	320.40	88.38	224.04	78.20
3	88.39	24.22	305.35	120.79	219.94	103.73
4	97.99	8.82	321.42	127.00	223.71	121.04
5	98.14	7.46	324.59	121.26	226.41	122.60

FIGURE 13. Computing resource consumption during attacks. (4)

In this investigation, five observations were noted when Dos attacks were stimulated against the HTTP/2 server. For each observation, the packet generator sent one test case of crafted HTTP/2 frame packets against the server. Hence, there were five test cases of crafted HTTP/2 frame packets in total. These are: Test Case 1: 2M PING frame packets sent to the victim. Test Case 2: 2M WINDOW_UPDATE frame packets were transmitted on stream 0, with a random window size-increment. Test Case 3: 2M WINDOW_UPDATE frame pack-

ets on stream 0, with a fixed window-size-increment. Test Case 4: 10K WINDOW_UPDATE frame packets on 200 different stream IDs, with a random window-size increment. Test Case 5: 10K WINDOW_UPDATE frame packets in 200 different stream IDs, with a fix window-size increment. Each experiment was repeated 30 times to reduce the variance in the results obtained, and to improve the overall confidence in the findings.

The next section presents the results of the observations. It is important to note that Test Case 1 required sending PING frame packets at the application level, as not to confuse it with the PING messages sent at the network level, notoriously known to cause Dos attacks. The PING frame is part of the HTTP/2 protocol. In order to send packets in different stream IDs as done through Test Cases 4 and 5, a HEADER frame was sent to the receiver, to open a new stream ID. Hence there were 200 HEADER frame packets sent to create 200 stream IDs. Other test cases were attempted, but the results were not in accord with the interest of this paper. *(7)*

3.4 The Evolution Of HTTP

The Web was based on the hypertext transfer protocol (HTTP) and the hypertext mark-up language (HTML). There were numerous precursors in the form of distributed hypertext systems, but in the true Internet tradition the simplicity and openness of the original HTTP and HTML standards allowed them to be readily implemented in forms that could be made to interoperate across the network. In 1991 HTTP 0.9 was published and basic HTTP in 1992, which later became known as HTTP 1.0.

In 1993, a Mosaic web browser, the most global Web Browser, came as a major boost. It is easy to use and it brought multimedia web pages to life. Google Chrome, Internet Explorer, Safari, and Mozilla Firefox retain many of the characteristics of the original Mosaic graphical user interface (GUI), such as the URL bar and back/forward/reload buttons. As the use of the web snowballed, HTTP 1.0 (fully specified in 1996) attracted attention from network researchers and they discovered considerable space for improvement.

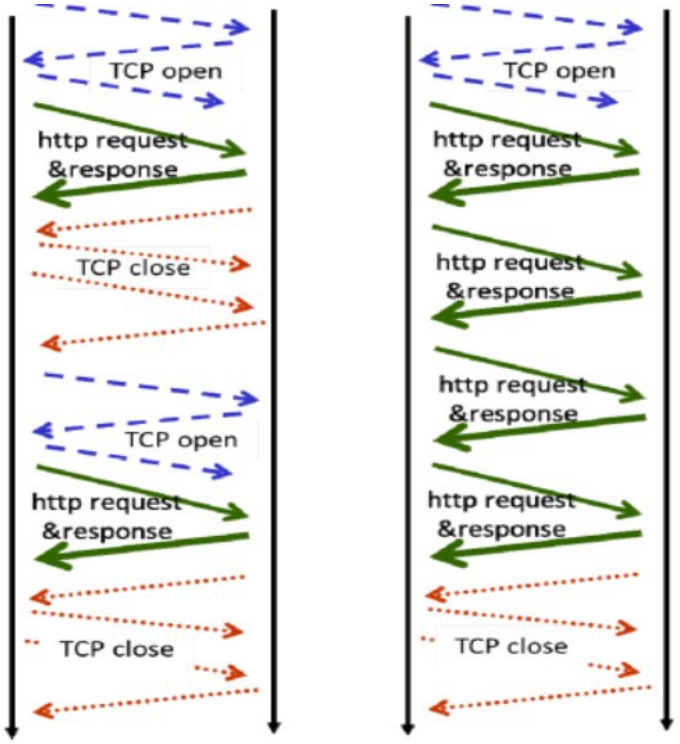

FIGURE 14. HTTP 1.0 & HTTP 1.1. (5)

HTTP 1.0 is a “stop and wait” protocol. In other words, if a web page consisting of some text and a few images was to be built and rendered, then multiple TCP connections were needed and enterprising browser designers decided to open these in parallel, reducing the overall Page Load Time (PLT), resulting in a better user experience.

HTTP 1.1 has improved over the previous version in the areas of: Caching, Bandwidth optimization, Connection management, Message transmission, Internet address conservation, Error notification, Security, Integrity & Authentication, and Content negotiation. For example, IP address conservation has been improved through the use of virtual hostnames for servers, specified in the new header “host” field. In practice HTTPS (HTTP over SSL or TLS) was adopted rather than the proposed HTTP 1.1 mechanisms for security. The concern over the inefficient use of TCP was addressed by improved connection management in the form of persistent connections. This means that a single TCP connection

between a client and a web server can be kept open to support multiple HTTP request and response interactions. (See figure 15 below)

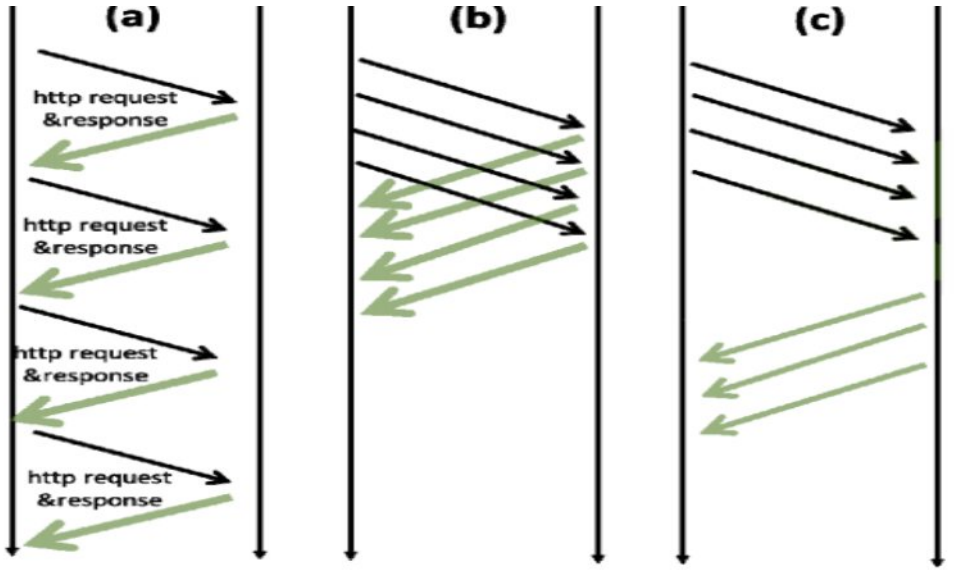

FIGURE 15. a) Stop & Wait; b) Pipelining; c) Head of Line Blocking. (5)

HTTP/2 is a major development to HTTP 1.1, it has been developed based on motivation of the need to improve the Page Load Time (PLT) of modern, large, and complex web pages.

Average page sizes and their complexity in terms of the number of objects have grown from approximately 10 Kbytes in 1995 to 1600 Kbytes in 2014, And from two objects (HTTP 1.0) in 1995 to over one hundred objects in 2014. Today the Web page encapsulates tens to hundreds of resources collected from multiple domains. Users access the Web from diverse device form factors, while browsers have improved dramatically, a constant throughout this evolution is the underlying application layer protocol HTTP designed at a time of far less page complexity with pages taking longer to load. Studies over the past five years suggest even 100 milliseconds additional delay can have a negative effect on Web use, spurring interest in improving Web performance.

TABLE 2. The Summary of the major differences between HTTP 1.0, HTTP 1.1, HTTP/2. (6)

HTTP 1.0	HTTP 1.1	HTTP/2
"Stop and Wait", strictly sequential processing of requests and responses over TCP	"Stop and Wait", strictly sequential processing of requests and responses over TCP	Full duplex streams of binary frames over TLS/TCP
PDU: HTTP Message	PDU: HTTP Message	PDU: HTTP/2 Frame (10 Types)
New TCP connection opened for each Request/Response pair Browsers seek performance gain by opening multiple parallel TCP connections, even between client and server in same domain	Persistent TCP connections specified and adopted Pipelining specified but not mandatory and not adopted Browsers continue to open multiple parallel TCP connections within same domain	Aim: One persistent TCP Connection per domain Multiple concurrent streams within the TCP connection Pipelining mandatory Stream Multiplexing and Prioritization Dynamic stream dependencies and reprioritization
Caching, Content compression option	Caching, content compression option	Caching, Content compression
		Header Compression Server Push Flow Control

4 Conclusion:

HTTP/2 is the future of the web and an exciting new option for web applications. HTTP/2 is a huge step toward making the web faster and more responsive, and it has already been adopted by some major web browsers. The current version of Chrome supports HTTP/2 by default, and so does the current version.

It provides strong support for more secure, simpler, faster site. We may find new and different performance techniques that help our web application under http/2, expecting to find lively online discussions about the best ways to use the new protocol. We hope this white paper is a useful early step in your journey toward gaining the simplicity, site performance improvements, and security offered by HTTP/2 for our web applications, Since the benefits of http/2 are many, but the updated protocol will require developers to change some their ways. HTTP/2 is without a doubt the direction the web is moving towards in terms of networking protocol that is able to handle the resource needs of today's websites. While SPDY was a great step forward in improving HTTP1.1, HTTP/2 has since further improved the HTTP protocol that has served the web for many years.

Given you have a server that supports the HTTP/2 protocol, we can start serving content over this protocol to users that are accessing your content through a supported browser. For browsers that do not support HTTP/2, they will continue to be delivered content through the old protocol. Using the HTTP/2 protocol will help make websites faster and overall will improve the web's user experience. *(14)*

REFERENCES

1. Akamai. 2015. Future of web. Date of retrieval 3.12.2016. https://http2.akamai.com/

2. Hoffman, B. 2014. Date of retrieval 9.3.2016. HTTP/2 request and response multi-plexing within a shared connection. https://moz.com/blog/http2-a-fast-secure-bedrock-for-the-future-of-seo

3. Allison, C. Hussein B. Http 1.0 & Http 1.1. 2015. Date of re-trieval 15.5.2016. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7344059

4. Allison, C. Hussein B. The Summary of the major differences between HTTP 1.0, HTTP 1.1, HTTP/2. 2015. Date of retrieval 15.5.2016. http://ieeexplore.ieee.org/stamp/stamp.jsp?

5. Allison, C Hussein, B. 2015. a) Stop & Wait; b) Pipelining; c) Head ofLine Blocking. Date of retrieval 15.5.2016. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7344059

6. Adi, E. Baig, Z. Peng Lam, C. Hingston, P. 2015. Computing Re-source Consumption During Attacks. Date of retrieval 15.5.2016. http://ieeexplore.ieee.org/stamp/stamp.jsp?_6666tp=&arnumber=7292994

7. Adi, E. Baig, Z. Peng Lam, C. Hingston, P. 2015. Date of retrieval 6.5.2016. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7292994

8. Adi, E. Baig, Z. Peng Lam, C. Hingston, P. The Configuration of the lab for investigation 1 and 3. 2015. Date of retrieval 5.5.2016. http://ieeexplore.ieee.org/stamp/stamp.jsp?8tp=&arnumber=7292994

9. King, G. What is SPDY.2013. Date of retrieval 12.3.2016 https://lincolnloop.com/blog/what-is-spdy/

10. Wagnon, J. 2013. Server Name indication. Date of retrieval 23.3.2016. https://devcentral.f5.com/articles/ssl-profiles-part-7-server-name-indication

11. Wagnon, J. 2013. Renegotiation. Date of retrieval 24.3.2016. https://devcentral.f5.com/articles/ssl-profiles-part-6-ssl-renegotiation

12. Dorfman, J. 2015. HTTP2 and SPDY Comparison. Date of retrieval 12.3.2016. https://www.maxcdn.com/blog/spdy-http2-shift/

13. Constantin, L. 2012. Feature to Hijack HTTPS sessions. Date of retrieval 28.3.2016. http://www.computerworld.com/article/2492541/security0/-crime--attack-abuses-ssl-tls-data-compression-feature-to-hijack-https-sessions.html

14. NGINX. HTTP/2 The Future Of the Web. 2015. Date of retrieval 16.6.2016. https://www.nginx.com/wpcontent/uploads/2015/09/NGINX_HTTP2_White_Paper_v4.pdf

15. O'Reilly. 2013. Http/2. Date of retrieval 8.3.2016 http://chimera.labs.oreilly.com/books/1230000000545/ch12.html

16. O'Reilly. 2013. HTTP/2 binary framing layer. http://chimera.labs.oreilly.com/books/1230000000545/ch12.html

17. O'Reilly. 2013. Server push is cacheable. Date of retrieval 3.12.2016. http://chimera.labs.oreilly.com/books/1230000000545/ch12.html#HTTP2_PUSH

18. R, Peon. 2015. Header Compression for HTTP/2. Date of retrieval 28.3.2016. https://http2.github.io/http2-spec/compression.html#compression.based.attacks

19. R, Mazerik. 2013. IT Security Training & Resources by InfoSec Institute. CRIME Attack. Date of retrieval 28.3.2016. http://resources.infosecinstitute.com/beast-vs-crime-attack/

20. Technopedia. Unknown. Denial Of Service Attack. Data of retrieval 4.4.2016. https://www.techopedia.com/definition/24841/denial-of-service-attack-dos.tp=&arnumber=7344059

21. TRICKOVISTA. 2014. DoS Attack. Date of retrieval 4.4.2016. https://trickovista.wordpress.com/author/trickovista/

22. Unknown. 2015. Renegotiation. Date of retrieval. 24.3.2016. https://wiki.mozilla.org/Security:Renegotiation

23. Unknown . 2008. Downgrade-attack. Date of retrieval 4.4.2016. http://openmaniak.com/ettercap_filter.php

24.Unknown. 2008. Downgrade-attack. Date of retrieval 4.4.2016. http://openmaniak.com/ettercap_filter.php

25. Beal, V. 2003. HTTP. Date of retrieval 8.3.2016 http://www.webopedia.com/TERM/H/HTTP.html

26. Lui, W. 2015. Http/2 Header Compression. Date of retrieval 9.3.2016. http://www.slideshare.net/walterliu7/http2-introduction

27. Wikipedia. 2016. SPDY. Date of retrieval 3.12.2016. https://en.wikipedia.org/wiki/SPDY