HTTP/2 Tsunami: Investigating HTTP/2 Proxy Amplification DDoS Attacks

Beckett, D., & Sezer, S. (2017). *HTTP/2 Tsunami: Investigating HTTP/2 Proxy Amplification DDoS Attacks*. Advance online publication. https://doi.org/10.1109/EST.2017.8090411

Document Version:
Peer reviewed version

HTTP/2 Tsunami: Investigating HTTP/2 Proxy Amplification DDoS Attacks

David Beckett, Sakir Sezer
CSIT, Queens University Belfast, Northern Ireland

***Abstract*—Distributed Denial of Service (DDoS) attacks cause significant damage to computer systems by taking a system offline. Hypertext Transfer Protocol (HTTP), is the most commonly used protocol for web services. The HTTP protocol has recently received a major update to HTTP/2. This new protocol provides increased functionality, however this poses a threat from DDoS due to the larger attack surface.**

HTTP/2 implements novel compression techniques to reduce bandwidth, in this paper we explore this compression technology to providing understanding on its risk from DDoS, specifically in a HTTP/2 to HTTP/1 proxy deployment. We implement a testbed and measure the bandwidth to show that a amplification attack is possible which is comparable to the current largest amplification attacks.

***Index Terms*—DDoS, HTTP2, HPACK, Flood, Amplification, Attack, Apache, nghttp2, Nginx, Vulnerabilities.**

I. Introduction

Cyber attacks have become a major concern for society, as they are becoming more prevalent and harder to protect against. A common attack type is a Distributed Denial of Service attack, this aims to take a network or service offline so that it's services can no longer be accessed.

These DDoS attacks are becoming more common [1] and are increasing in magnitude. A common target for a DDoS attack is the network infrastructure, whereby the network links and devices are flooded with malicious packets to the extent were they can no longer carry out their designed functionality.

Botnets are commonly used as an attack source, this poses a serious issue as the infected bots are distributed in nature, and can belong to unaware members of the general public, as recently illustrated by the Mirai botnet [2].

A large portion of the internet is transmitted using the Hyper Text Transfer Protocol (HTTP). In 2015, this protocol received a substantial upgrade and was released as HTTP/2 [3]. The latest version of this protocol provides new functionality with the aim of improving user's quality of service. HTTP/2 introduces multiplexed streams allowing for multiple requests and responses to be sent together in the same packet. It also introduces a new compression methodology for the header content to minimise bandwidth usage.

This new protocol has seen adoption by many websites. A common approach is to use a HTTP/2 to HTTP/1 proxy, this allows for the older version of the protocol to be used on the existing web infrastructure, whilst providing the benefits of the new protocol between users and the website proxy. Content Delivery Networks (CDN) have also started to enable this proxy deployment method for many of their clients still running HTTP1 on their backend environment.

The benefits of HTTP/2 are well known, however there is currently a lack of coverage on the risk from DDoS. In this paper we provide an experimental study on the risk of attack due to the latest version of the protocol, relating to the use of it's compression technique HPACK [4], as this is a potential target for an amplification style attack.

II. HTTP/2 Protocol Overview

Internet services and connection speeds have significantly improved since the 1990s, however the HTTP protocol remained relatively unchanged. The protocol only allowed a single HTTP request to be active at 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, in particular due to large cookie values which were often duplicated in subsequent requests. Due to these issues, Google in 2012 proposed SPDY, an alternative to the legacy HTTP/1.1 protocol. SPDY aimed to solve these issues by introducing multiplexed requests and by introducing a new header compression technique. SPDY was formally adopted as the new HTTP protocol, HTTP/2 in 2015 and released as a RFC [3]. The following section will contain some of the changes that the protocol has introduced so that potential attacks can be exposed.

A. HTTP/2 Headers

The HTTP protocol communicates between the server and client using headers. These headers include basic information such as the website URL, the requested resource and user identification data such as cookies. Since the introduction of HTTP, these header sizes have significantly increased.

Websites now require a multitude of cookies, for gathering user analytics and for maintaining user sessions. User agents are also sent which identifies the web browser that the user is using. These values are typically the same for all requests, resulting in massive amounts of header data duplication.

HTTP/2 aims to solve this by introducing its own bespoke header compression format, HPACK[4]. HPACK reduces header duplication by implementing data lookup tables for common header values. Requests can reference the location of the data within the storage table to reduce data retransmission. These header values are encoded using Huffman Encoding to minimise their overall size for when full data is required to be sent.

Index	Header Name	Header Value
1	:authority	
2	:method	GET
3	:method	POST
4	:path	/
5	:path	/index.html
6	:scheme	http
7	:scheme	https
8	:status	200
9	:status	204
10	:status	206
11	:status	304
12	:status	400
13	:status	404
14	:status	500
15	accept-charset	
16	accept-encoding	gzip, deflate
17	accept-language	
18	accept-ranges	
19	accept	

Fig. 1. Extract from HPACK Static Table

HPACK introduces two header memory tables, static and dynamic. The static table contains common header values as defined by the RFC, such as common HTTP response codes and header field names as can be seen in Figure 1.

The dynamic table is connection specific, it is created for the life of each TCP connection. Each party manages their own copy of the table. After a value is stored within the dynamic table, the field's memory location can be referenced within future request minimising the retransmission of common header values. An example can be seen in Figure 2.

The dynamic table has an initial size of 4K Bytes as defined by the RFC however this can be increased or decreased. Current implementations however have kept this value at the default 4KB value.

B. HTTP/2 Stream Control

HTTP/2 introduces multiplexing of requests and responses, which are transmitted in streams as can be seen in Figure 3. Both parties can negotiate the maximum number of active streams. If a value is not defined by the clients, the initial value is assumed to be infinite. The RFC recommends a value of 100, however this can be increased or decreased through setting frames. In current implementations, the current chosen values are 100 for Apache and nghttp2, whilst Nginx uses 128.

C. HTTP/2 Deployment

Currently the top products for implementing HTTP/2 are Nginx and Apache [5]. IIS has HTTP/2 functionality but it is currently not as widely deployed. Apache also does not implement HTTP/2 with its own library, but instead uses the open source nghttp2 API.

HTTP/1.1	HTTP/2
1st Request for index.html	
GET /search.php **Host:** www.onlineshop.com **User-Agent:** Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99 Safari/537.36 **Cookie:**A=AIUaSmZgpspNZ-v0bA14Ew8_w54mc5UKIDlnrrXOMYZ1us—xJu85507vSWF5AwQ=	**:method:** GET **:path:** /search.php **:authority:** www.onlineshop.com **User-Agent:** Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99 Safari/537.36 **Cookie:**A=AIUaSmZgpspNZ-v0bA14Ew8_w54mc5UKIDlnrrXOMYZ1us—xJu85507vSWF5AwQ=
Subsequent Requests	
GET /image1.jpg **Host:** www.onlineshop.com **User-Agent:** Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99 Safari/537.36 **Cookie:**A=AIUaSmZgpspNZ-v0bA14Ew8_w54mc5UKIDlnrrXOMYZ1us—xJu85507vSWF5AwQ=	**:method:** GET **:path:** /image1.jpg **&A0** **&A1** **&A2**
& denotes reference to Dynamic Table ***Table does not reference Static Table for simplicity***	

Fig. 2. HTTP/2 Example showing referencing of Dynamic Table

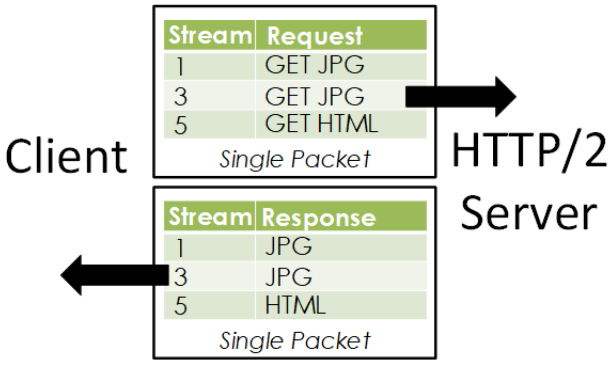

Fig. 3. HTTP/2 Streams

Due to the radical changes in the protocol, legacy web firewalls are unable to analyse the binary protocol due to the introduction of streams and the use of HPACK header compression. Many implementations instead use a proxy to use HTTP/2 between client and proxy, but still use HTTP/1.1 at the back-end, this still allows existing infrastructure such as existing web application firewalls and load balancers to be used whilst gaining the benefits of the new protocol.

CDNs such as Cloudflare have also enabled HTTP/2 by default for all their clients, resulting in all website visitors to use HTTP/2. If the website back-end hasnt been updated to the latest protocol, the proxy will convert the traffic to HTTP/1.1.

D. HTTP/2 Potential Vulnerabilities

DDoS attacks against HTTP are well known, however the new protocol introduces increased complexity which provides the possibility of new attack methodologies, as well as potentially enhancing existing attacks. Due to this risk, the RFC

contains a brief section dedicated to some of the DDoS issues the protocol introduces[3].

HTTP/2 has introduced multiplexed flows, along with flow management within the application layer. This new flow mechanism may allow for existing network layer attacks to be replicated within the application layer, for example if an attack floods the web server with flow window updates, this may consume the web servers processing power.

Historically, the HTTP request flood attack has been a popular attack form. Due to the multiplexing of requests in HTTP/2, this will give an attacker a enhanced attack target as they will be able to send more requests per packet.

Amplification attacks are a very common DDoS attack which seeks to consume the capacity of the target's network link by reflecting low bandwidth traffic off a reflection source which provides a larger data response. The use of header compression by HTTP/2 may give an attacker a method of amplification as the headers will have to be decompressed before processing. Current amplification factors from existing attacks can be seen in Table I.

TABLE I
TOP ATTACK BANDWIDTH AMPLIFICATION FACTORS [6]

Protocol	Factor
NTP	556.9
Chargen	358.8
QOTD	140.3
RIPv1	131.24
Quake	63.9
LDAP	46 to 55
DNS	28 to 54

If the attacker can create an amplification attack which causes the decompressed header data to exceed either the memory of the server or the bandwidth of the link, this will cause system instability. This can be achieved with low sized incoming headers if the attacker can find large decompression methods and if there is no upper limits set on the maximum decompression size.

III. RELATED WORK

C Rossow et al [7] analysed UDP amplification attacks to expose the risk of existing vulnerabilities with network protocols. Bait servers were deployed and back scatter analysis was performed to provide insight into their current usage. Using UDP as the network protocol for an amplification attack allows for IP spoofing to be used so that attackers can reflect off vulnerable servers to target any IP address.

R Sherwood et al [8] investigated the use of TCP for amplification attacks by generating optimistic ACKS causing the network link throughput to increase, resulting in the victim to consume their own network bandwidth.

B Sieklik et al [9] analysed the TFTP application for potential amplification attacks. They discovered a amplification factor of 60x which could generate a large amount of attack traffic for DDoS.

E Adi et al[10] [11] analysed HTTP/2 and it's risk from DDoS. 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 relating to DDoS [12]. 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.

IV. RATIONALE FOR CURRENT RESEARCH

DDoS is not a new topic of research, however existing research carried out on HTTP/1.1 does not provide sufficient coverage over the new HTTP/2 protocol. Currently the only published work on HTTP/2 is by E Adi et al[10][11] and the security vendor Imperva [12], however these do not provide complete coverage of the potential attack vectors of HTTP/2.

The changes in the new protocol dramatically change the overall operation of the protocol and has altered the threat posed by DDoS. There is currently no research into the risks posed on the risk posed by HTTP/2 proxies for bandwidth amplification attacks.

Amplification attacks seeks to consume the network link capacity of the target, so that it can no longer handle the traffic. We believe the use of header compression by HTTP/2 gives an attacker a method of amplification, allowing for the exploitation of the application layer to perform a network layer DDoS and this is the topic we have to chosen to investigate.

From obtaining knowledge of this attack, before it is seen by industry, we aim to provide researchers with a threat model so that defence systems may be proposed in the near future.

V. PROXY AMPLIFICATION ATTACK

HTTP/2 compression implements a dynamic table for header data storage. On a header value's first occurrence it can be stored in this table so that subsequent requests can instead reference the data. HTTP/1.1 does not have this compression functionality, therefore if a HTTP/2 to HTTP/1.1 proxy handles the requests, the requests will need to be expanded in full for HTTP/1.1. These small referenced HTTP/2 requests therefore will result in large sized HTTP/1.1 requests as can be seen in Figure 4. This could be exploited as an attack vector, to consume the network bandwidth of a back-end data link.

To understand this risk, we have performed amplification DDoS attacks on an experimental testbed. These simulations consist of an attacker exploiting the dynamic header compression by referencing stores values from the dynamic table.

A testbed environment was setup to allow for these experiments to be carried out. Sensors were positioned onto the links of the network so that both bandwidth sizes and packet rates could be recorded, allowing for the relevant amplification factors to be measured.

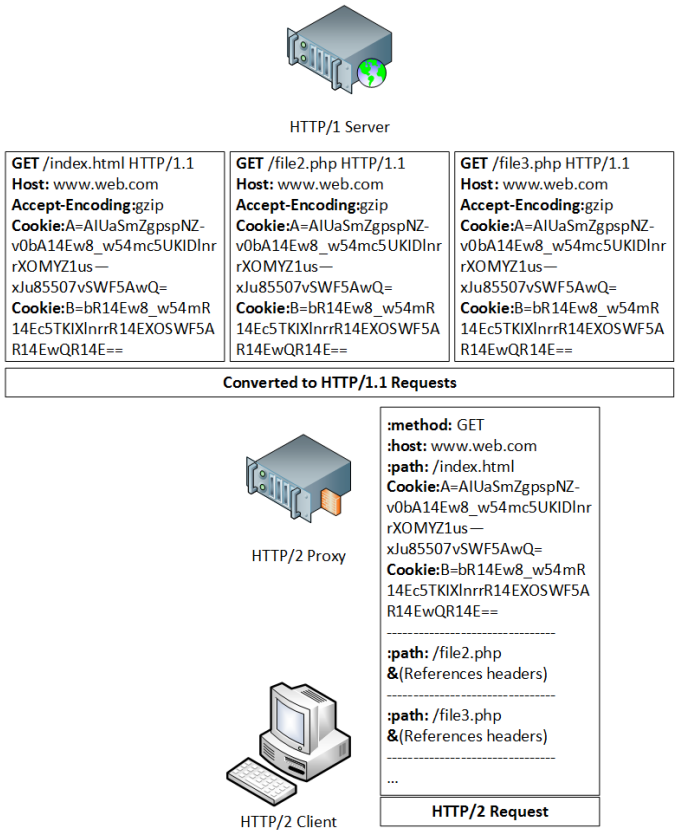

Fig. 4. Amplification of packets when using HTTP/1 to HTTP/2 Proxy

For amplification factors, we consider both packet and bandwidth amplification. If the proxy requires a larger quantity of packets on the HTTP/1 backend, this may cause it to crash as packet generation is also a bottleneck for network devices.

A. Testbed Architecture

The testbed comprised of a HTTP/2 gateway proxy which connects to a HTTP/1.1 back-end webserver. All simulations are carried out on a private cloud environment running Ubuntu 14.04 as an operating system. Apache 2.4.7 is used for the back-end HTTP/1.1 traffic. For the proxy, the two most common HTTP/2 proxies were tested, Nginx and nghttp2. IIS and native Apache do not currently provide proxy support.

The attack tool used was h2load, the tool was fed the necessary parameters to generate all of the following amplification attacks.

The tool, tcpdump captures the traffic at the attacker location and at the back-end location. Only packets containing a payload were measured to minimise the influence of deployment choices regarding the frequency of TCP ACKs. The byte size is measured for the entire packet including Ethernet, IPv4, TCP headers and HTTP payload.

B. Dynamic Table Memory Consumption

To obtain the optimum header pattern for an attacker, the compression of the HTTP header field must be fully understood to find the attack set with the maximum impact. Each HTTP request features multiple header values such as cookies, user agents and host name.

Each stored header record i, consumes memory within the dynamic table. The field name for each header record consumes F number of bytes, the corresponding field value consumes V bytes and each record requires an additional 32 Bytes overhead. Each character within the record consumes a single Byte. The total header size needs to be below the total maximum dynamic header size to allow for compression to operate. Using this information, the following formula was produced.

$$\sum_{i=0}^{N} F_i + V_i + 32(Bytes) < MaxDynamicHeaderSize \quad (1)$$

Due to the overhead of 32 bytes per cookie instance, an optimum attack pattern will aim to reduce the number of cookie instances and instead increase the byte size of each cookie to maximise the amplification power.

C. Amplification Attack Experiments

The objective of the experiment is to show the amplification risk posed by the decompression of HTTP/2 request headers. The max concurrent stream value affects the number of open streams a client can have at a point in time, if this value is large, a client should gain larger amplification due to more frequent requests which can reference the dynamic table.

Each HTTP/2 request requires several header values to facilitate the request, these can be seen in Table II. The cookie values chosen for the following experiments were obtained using equation 1. The cookie structure used for the following simulations can be seen in Table III.

1) Effect of Max Concurrent Requests on Amplification Factor: The aim is to demonstrate how an attacker can generate an amplification attack using the dynamic table size of 4KB. The simulation's first request contains 2 instances of 1940 character cookies, to fill up the 4KB table. The first request headers will be sent to the server in full, however HPACK will store these large values into its dynamic table. Subsequent requests may now reference these large cookie values from the dynamic table, whilst requiring a relatively small transmission size.

The recommended max concurrent value of 100 was chosen to illustrate the base amplification factor an attacker will have. Varying levels of simultaneous requests were sent to the server ranging from 1 request (with no subsequent requests) to 768.

The experiment is subsequently repeated using alternative max concurrent values of 128, 256 and 512 to demonstrate the affect this has on the amplification factor.

2) Results: After the experiments were executed, the amplification factor was calculated by calculating the ratio between the attackers traffic and the traffic produced by the proxy to the back-end server.

Figures 5 and 6 show the amplification factor vs numbers of active requests streams sent in that particular connection.

TABLE II
HTTP/2 REQUEST HEADER VALUES

Field Name and Value	Field Name Length (Bytes)	Field Value Length (Bytes)	Overhead (Bytes)	Total (Bytes)
Authority:www.csitweb.co.uk	9	17	32	58
User-Agent:h2load nghttp2/1.14.0-DEV	10	25	32	67
Cookie:CN=(100 Chars)	6	103	32	141

TABLE III
SCHEMATIC OF EXPERIMENT 1 REQUESTS

Request	Header Structure
1	C0=ABCDE(1940 chars) C1=ABCDE(1940 chars)
2	&C0 (reference to Dynamic table) &C1 (reference to Dynamic table)
...	...
N	&C0 (reference to Dynamic table) &C1 (reference to Dynamic table)

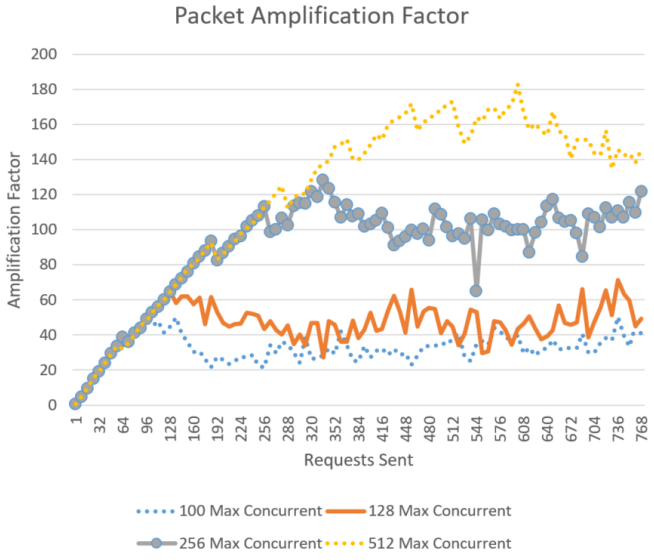

Fig. 5. Packet Amplification for varying max concurrent values

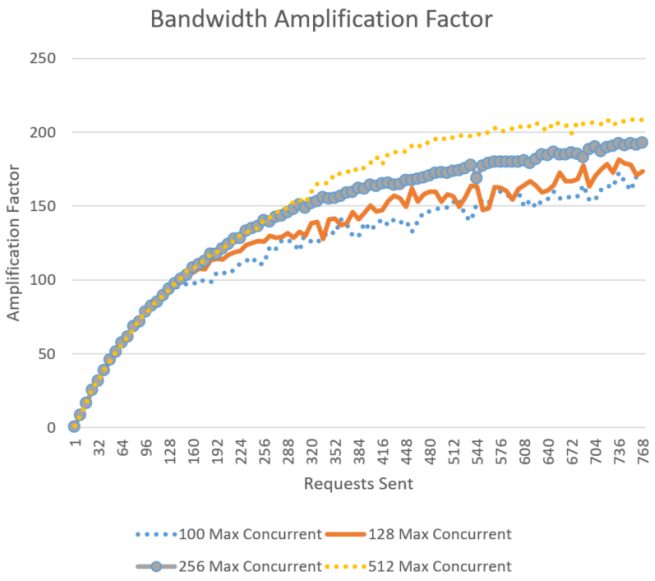

Fig. 6. Bandwidth Amplification for varying max concurrent values

TABLE IV
TABLE: AMPLIFICATION FACTOR FOR VARYING MAX CONCURRENT STREAM LIMITS. MAX DYNAMIC TABLE SET AT 4KB

	Amplification Factor at Max Concurrent limit	
Max Concurrent Setting	Bandwidth	Packet
100	79.2x	46.8x
128	94.4x	64.9x
256	140.6x	113x
512	196.3x	173x

It can be seen from the graph shown in Figure 5 that the attack amplification factor increases linearly for all scenarios until the sent requests matches the relevant max concurrency value. After the number of attack requests exceeds the max concurrency value, the amplification rate curtails and levels off. This is due to the extra HTTP/2 packet overhead required to facilitate the freeing of streams for the remaining requests.

Due to the loss of amplification and the extra computational complexity, we can assume the attacker will not send more requests than the max concurrent value in a single connection. We expect for an attacker to create multiple connections each sending the requests at the max concurrent value set by the server to maximise the impact of their amplification attack. Therefore the max amplification factor is taken from the graph at the max concurrent limit value. These values are shown in Table IV.

VI. CONCLUSION

The objective of the research is to understand, in a HTTP/2 deployment, the risk of a DDoS attack relative to the threat posed by the previous HTTP/1.1 version.

We have shown the vulnerability of a amplification attack in a HTTP/2 to HTTP/1 environment. This allows a sophisticated attacker to exploit the HTTP/2 header compression, HPACK to generate large packet payload onto the back-end data link.

The max concurrency value set by the server also influences the amplification power of the attacker. For 128 max concurrent value with a 4KB dynamic table, this gave a HTTP/2 to HTTP/1 amplification factor of 94.4x whilst 100 max concurrent value gave a amplification factor of 79.2x. For proxies, it is therefore safer to choose this lower limit of 100.

Another possible amplification attack vector emerges due to publicly available HTTP proxies. If they support HTTP/2

they can be used to amplify HTTP/2 compressed traffic to any HTTP/1.1 website. This opens up this attack to allow any HTTP/1.1 website to be attacked through reflection.

We believe we have benefited the research community by providing this information so that suitable defence mechanisms can be proposed. We hope to explore suitable detection strategies in our future work and analyse further attack methods relevant to HTTP/2.

HTTP/2 deployment is on the rise, however security research on this new protocol is inadequate, we have provided much needed research on this protocol so that future deployments can be implemented with parameters that can decrease its risk from DDoS.

REFERENCES

[1] A. Networks, "WORLDWIDE INFRASTRUCTURE SECURITY REPORT XII 2017," 2017. [Online]. Available: https://www.arbornetworks.com/insight-into-the-global-threat-landscape

[2] Incapsula, "Breaking down Mirai Botnet," 2017. [Online]. Available: https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html

[3] IETF, "RFC:7540, Hypertext Transfer Protocol Version 2 (HTTP/2)," 2015. [Online]. Available: https://tools.ietf.org/html/rfc7540

[4] IETF., "RFC:7541, HPACK: Header Compression for HTTP/2)," 2015. [Online]. Available: https://tools.ietf.org/html/rfc7541

[5] W3TECH, "Usage of HTTP/2 broken down by web servers," 2017, online; accessed 10-Mar-2017.

[6] US Computer Emergency Readiness Team (US-CERT), "UDP-based Amplification Attacks," https://www.us-cert.gov/ncas/alerts/TA14-017A, 2014, online; accessed 26-Aug-2014.

[7] C. Rossow, "Amplification hell: Revisiting network protocols for ddos abuse," in *2014 Network and Distributed System Security Symposium*, 2014.

[8] R. Sherwood, B. Bhattacharjee, and R. Braud, "Misbehaving tcp receivers can cause internet-wide congestion collapse," in *Proceedings of the 12th ACM Conference on Computer and Communications Security*, ser. CCS '05. New York, NY, USA: ACM, 2005, pp. 383–392. [Online]. Available: http://doi.acm.org/10.1145/1102120.1102170

[9] B. Sieklik, R. Macfarlane, and W. J. Buchanan, "Evaluation of tftp ddos amplification attack," *Computers & Security*, vol. 57, pp. 67–92, 2016.

[10] E. Adi, Z. A. Baig, P. Hingston, and C.-P. Lam, "Distributed denial-of-service attacks against http/2 services," *Cluster Computing*, vol. 19, no. 1, pp. 79–86, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10586-015-0528-7

[11] E. Adi, Z. Baig, C. P. Lam, and P. Hingston, "Low-rate denial-of-service attacks against http/2 services," in *2015 5th International Conference on IT Convergence and Security (ICITCS)*, Aug 2015, pp. 1–5.

[12] Imperva, "HTTP/2: In-depth analysis of the top four flaws of the next generation protocol," Aug, 2016.