A critical review of: Slow rate denial of service attacks against HTTP/2 and detection

Bastien DHIVER
bfrd2@kent.ac.uk

April 4, 2018

Nikhil Tripathi and Neminath Hubballi. Slow rate denial of service attacks against http/2 and detection. *Comput. Secur.*, 72(C):255–272, January 2018

Contents

1 Introduction

In this paper [1], Tripathi and Hubballi seek to assess the security level of the HTTP/2 protocol against Slow Rate Denial-of-Service attacks. In order to do that, a comparative study with the previous version of the protocol (HTTP/1.1) is conducted. Five novel attacks against popular HTTP/2 capable web servers are described and their effects measured. Lastly, a statistical abnormality measurement technique to detect the proposed attacks is discussed and evaluated.

This critical review will evaluate the research conducted by the authors, consider its applicability for a real-life environment and finally, suggest avenues for improvement.

2 Authors and Institution

The research was conducted by two Indian researchers, Nikhil Tripathi and Neminath Hubballi. Both authors carried this research at the Indian Institute of Technology Indore (IIT) in India. Nikhil is currently a Ph.D. student in the department of Computer Science and is interested in topics such as Computer Networks and Network Security as shown by his publications[1]. One particular area of interest for him is Slow Rate Denial-of-Service attacks. On the other hand, Neminath is an assistant professor in the department of Computer Science and Engineering and Nikhil's supervisor. His interests vary from Network security, System Security, Cloud Security and dependable systems. He previously worked for HP, Infosys Lab and Samsung R&D.

3 Background

Background on Denial-of-Service attacks and the HTTP/2 protocol is provided to the reader before getting to the heart of the matter. Both topics are well introduced.

3.1 (Distributed) Denial-of-Service attacks

Four types of attacks are described along their limitations from malicious client's perspective. Attacks can be performed at both, the transport and the application layer. A good overview is presented and references are given while doing so. This is a good choice to present attacks at various layers as the most commonly seen are the Transport-level DDoS Flooding attacks. We can get an idea of the proportions as shown from a recent Akamai report [2] on Figure 1. They succeed to pinpoint the drawbacks of non-slow rate attacks which are their easy detection and the resources needed for an attacker to perform such attacks (network bandwidth, computational resources). They chose to target the number of available slot in web servers connection queue which is smart because their limited number make them more vulnerable and the effects, really powerfull. In this process, looking as much as possible as legitimate traffic is the key to avoid detection. DoS attacks frequency is growing as well as their intensity and the associated costs as report Neustar[2]. The threat is real, the financial impacts costly in addition to the loss of customers and the image of the company damaged.

3.2 HTTP/2

The HTTP/2 protocol is quickly introduced with a description of essentials frames along a diagram representing an exchange during a basic transaction. This helps the reader to quickly get an overview of how the protocol works. The cleartext version of HTTP/2 (h2c) is not used by web browsers though, TLS is a De Facto standard, even if it wasn't included in the standard[3].

[1] https://scholar.google.co.uk/citations?user=Ur2fiEUAAAAJ

[2] https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q4-2017-state-of-the-internet-security-report.pdf

[3] https://http2.github.io/faq/#does-http2-require-encryption

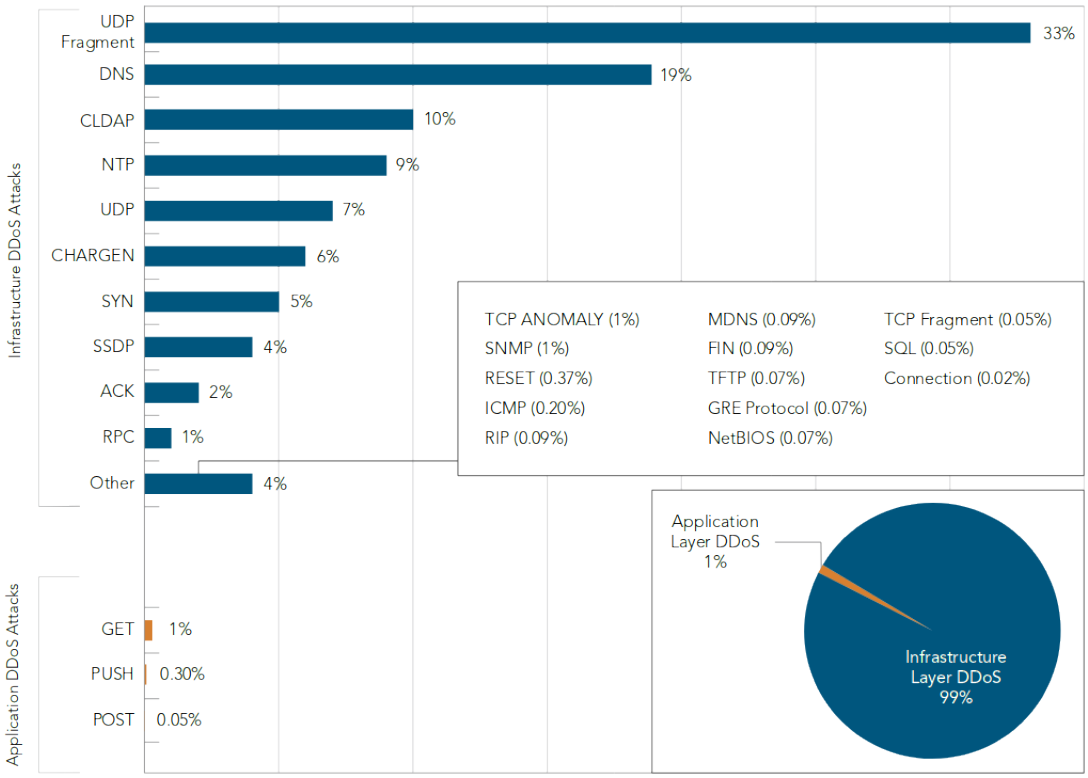


Figure 1: DDoS Attack Vector Frequency, Q4 2017 from Akamai

With that in mind, the exchanges between a server and a client are different, as shown by this Wireshark network capture I made on Figure 2. The decryption process is described in this presentation[4]. We can see that after the TCP and the TLS handshakes, the server is the one to send the first frames. Taking that into account and updating the diagram could have been a good idea here.

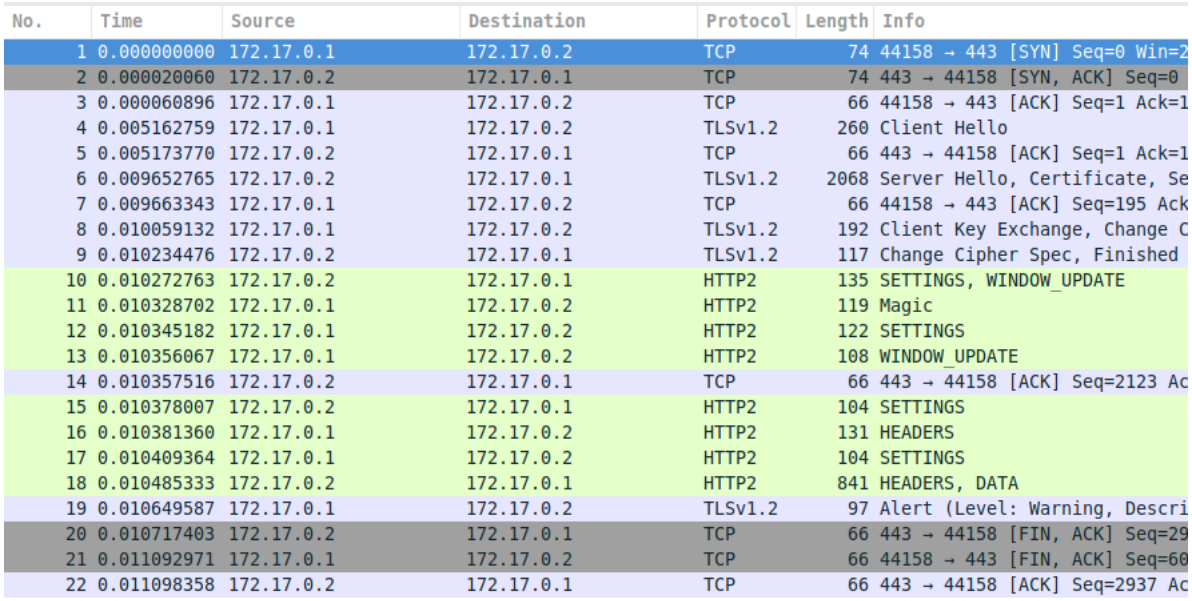
4 Reviewing main points of the article

A review of three important parts will be made, considering the testbed set up, the proposed attacks and their detection.

4.1 Testbed

The authors selected four popular web servers that supports the HTTP/2 protocol, namely Apache, Nginx, H2O and Nghttp2. The web servers were installed on a penetration testing GNU/Linux distribution named Kali. This doesn't reflect environments used in production but since this distribution is based on Debian, it's not an issue. At the time of this writing, the Debian distribution is used by 30% of the websites hosted on a GNU/Linux system[5], just behind Ubuntu with it's 37%, which is based on Debian as well. The Proofs-of-Concept attacks were implemented in Python by the authors which makes sense since it's commonly used to demonstrate and validate such assumptions. A well-known benchmarking tool named h2load was used to create and sent legitimate traffic. The architecture set up is very simple but sufficient to test and measure the attacks effects. Monitoring could have been put in place to provide graphs of the target resources usage.

[4]https://sharkfesteurope.wireshark.org/assets/presentations17eu/15.pdf
[5]https://w3techs.com/technologies/details/os-linux/all/all



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.1	172.17.0.2	TCP	74	44158 → 443 [SYN] Seq=0 Win=2
2	0.000020060	172.17.0.2	172.17.0.1	TCP	74	443 → 44158 [SYN, ACK] Seq=0
3	0.000060896	172.17.0.1	172.17.0.2	TCP	66	44158 → 443 [ACK] Seq=1 Ack=1
4	0.005162759	172.17.0.1	172.17.0.2	TLSv1.2	260	Client Hello
5	0.005173770	172.17.0.2	172.17.0.1	TCP	66	443 → 44158 [ACK] Seq=1 Ack=1
6	0.009652765	172.17.0.2	172.17.0.1	TLSv1.2	2068	Server Hello, Certificate, Se
7	0.009663343	172.17.0.1	172.17.0.2	TCP	66	44158 → 443 [ACK] Seq=195 Ack
8	0.010059132	172.17.0.1	172.17.0.2	TLSv1.2	192	Client Key Exchange, Change C
9	0.010234476	172.17.0.2	172.17.0.1	TLSv1.2	117	Change Cipher Spec, Finished
10	0.010272763	172.17.0.2	172.17.0.1	HTTP2	135	SETTINGS, WINDOW_UPDATE
11	0.010328702	172.17.0.1	172.17.0.2	HTTP2	119	Magic
12	0.010345182	172.17.0.1	172.17.0.2	HTTP2	122	SETTINGS
13	0.010356067	172.17.0.1	172.17.0.2	HTTP2	108	WINDOW_UPDATE
14	0.010357516	172.17.0.2	172.17.0.1	TCP	66	443 → 44158 [ACK] Seq=2123 Ac
15	0.010378007	172.17.0.2	172.17.0.1	HTTP2	104	SETTINGS
16	0.010381360	172.17.0.1	172.17.0.2	HTTP2	131	HEADERS
17	0.010409364	172.17.0.1	172.17.0.2	HTTP2	104	SETTINGS
18	0.010485333	172.17.0.2	172.17.0.1	HTTP2	841	HEADERS, DATA
19	0.010649587	172.17.0.1	172.17.0.2	TLSv1.2	97	Alert (Level: Warning, Descri
20	0.010717403	172.17.0.2	172.17.0.1	TCP	66	443 → 44158 [FIN, ACK] Seq=29
21	0.011092971	172.17.0.1	172.17.0.2	TCP	66	44158 → 443 [FIN, ACK] Seq=60
22	0.011098358	172.17.0.2	172.17.0.1	TCP	66	443 → 44158 [ACK] Seq=2937 Ac

Figure 2: Decrypted HTTP/2 exchange between Nginx and curl displayed in Wireshark, client IP address is 172.17.0.**1**

4.2 The attacks

Five novel and effective attacks targeting the number of free connection slots available of popular web servers are described by the authors along with the measured effects. Clear interaction diagrams at protocol level are provided for every attack and it really helps to understand how they are conducted and how they can be implemented. All of them requires a single HTTP payload with no more than four frames in order to cause the desired effects. This confirms the low level of interaction required to perform slow rate DoS. In the majority of cases, they require the client to re-send the payload after a specific timeout as the connections are closed by the server. We can consider them as novel because no mentions of such attacks on HTTP/2 were found so far in the literature.

4.2.1 Effects

The measured effects put Apache as the more vulnerable of the four tested servers. Their results claimed that only 150 connections were able to create a DoS scenario no matter which attack is performed on this web server. It also put in perspective that some servers had infinite timeout values depending on the attacks except for the web server Nghttp2. The paper does not focus on CPU, RAM, network usage and processes states of the targeted system, it could have strengthened the stealth claims that were made. An observation I had is that the behaviour of web servers are different in terms of timeout values and frames sent back, this can easily lead to software fingerprinting. The authors tested the attacks effects over TLSv1.2 as well which is realistic. Even if TLSv1.3 has just been approved by the IETF two weeks ago[6], TLSv1.2 is and will stay the default for a little while. Researchers used the Python Scapy packet manipulation program to perform the TLS handshake and then sent hardcoded payloads. It would have been so much easier to wrap the TLS context over the TCP socket to avoid the overhead mentioned. A comparison of slow rate DoS attacks in HTTP/1.1 and HTTP/2 is made with the Apache web server and focuses on structurally similar attacks such as Slow read, Slow message body and slow header. The slow message body and slow header are effective in both cases but the slow read attack is only effective on HTTP/2.

[6]https://www.ietf.org/mail-archive/web/ietf-announce/current/msg17592.html

4.3 Detection

Current web traffic is collected during the testing phase and then compared with the legitimate traffic previously collected from the training phase. A set of five features closely linked to the protocol is defined. A distance measurement technique using chi-square test statistics is used to measure the deviation between the recorded phases. The authors developed a program to extract the required features but no code was provided… Their program uses a Java library, why not use the Python Scapy library they mentioned page 9? This library is well suited for packet manipulation and inspection. One claimed contribution of the authors on page 2 is:

> We also propose a novel detection mechanism to detect these Slow Rate DoS attacks.

Well, the usage of the chi-square method for detecting anomalies in network traffic with profiles and feature selection isn't new at all [3] and has been used since 2003 at least.

4.3.1 Setup

A sample website has been put in place, with images and an upload form to perform their tests. Few volunteering users were asked to access the sample website. The setup seams credible enough. Apache was chosen to perform the tests which isn't a bad choice regarding its adoption over the web[7] as shown by Figure 3.

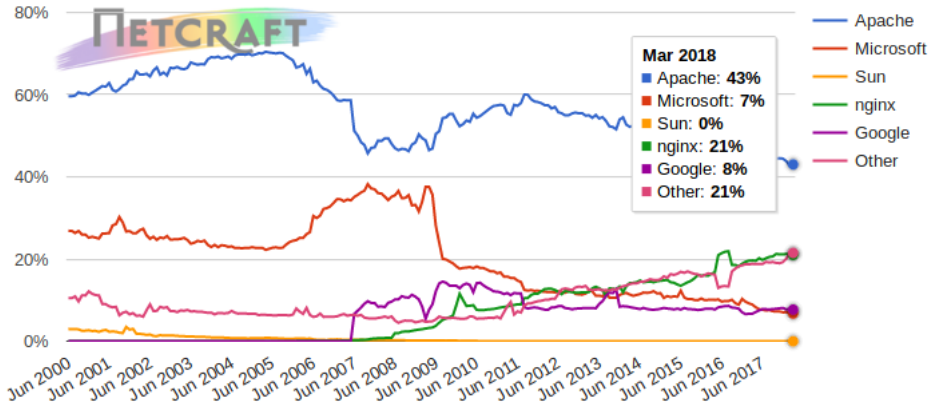


Figure 3: Percentages of websites using various web servers by Netcraft

It's also worth noting that the proposed technique used for detection with encrypted HTTP/2 traffic which uses intercepting proxy could not work out of the box depending on software implementations due to the deployment of TLSv1.3[8].

4.3.2 Detection Performance

Recall and False Positive Rate are used as performance metrics and graphs are provided. The detection scheme will have a high accuracy detection only if the connection pool space is consumed for a definite amount of time. The sending rate is so slow that the measured traffic profile is almost similar to normal HTTP/2 traffic. The detection performance depends on the threshold chosen for χ^2 value and the size of the window period (ΔT) chosen for monitoring features. Since the attacks mimic real traffic, this threshold will eventually be set to a lower value which can lead to a false positive increase. The overall effects look a bit mixed unfortunately.

[7] https://news.netcraft.com/archives/2018/02/13/february-2018-web-server-survey.html
[8] https://www.imperialviolet.org/2018/03/10/tls13.html

5 Prior work

Authors are well aware of the relation of their work with existing literature.

5.1 Vulnerabilities in HTTP/2 protocol

I only could find two papers by Adi and al. [4][5], a Ph.D. thesis [6] and a report by Imperva [7] that discussed DoS attacks against HTTP/2. The researchers found the same resources. The thesis was not available at the time their research was published. A comparison of the proposed attacks with the attacks discussed in Adi et al. is provided. A small mistake is made, they confused the number of packets with the number of connections required. Tripathi and Hubballi give credibility of their attacks compared to the described ones since the latter are based on flooding which makes the detection trivial and the impact is a reduction of Quality (RoQ) only. The two other resources mentioned mainly reported implementation issues that have been patched since.

5.2 Anomalies in encrypted network traffic

Observing properties such as inter-packet arrival time and time gaps between a request and its corresponding response can be used to detect anomalies in encrypted network traffic. Why didn't the authors explore the subject a little more? No intercepting proxy is needed and it could be sufficient to detect the proposed attacks.

5.3 Chi-square test

Chi-square test is widely used to detect network intrusions, port scanning and botnet control servers (C2) among others with a decent accuracy. This is the only detection test mentioned in the paper, more literature review could have help to mention a couple of alternatives such as machine learning techniques.

6 Applicability of Research

The paper is fairly recent, it has been submitted in 2017, published in 2018 and referenced on ACM[9]. HTTP/2 deployment and adoption is growing to reach nearly 25% as shown by Figure 4. This is a timely subject and not a week/month goes by without hearing about DoS attacks and their growing impacts. Authors used real web servers that are widely used all over the Internet, setup credible testbeds and demonstrated the real effects of the proposed attacks. This research can be the basis of building detection mechanisms for real world production environments once plugins for web servers and/or proxies will appear.

7 Criticisms

7.1 Proof-of-Concept

The authors claimed to have implemented the attacks in Python, but no links or references to the code are provided whatsoever. I had to implement them to verify the veracity of the claimed effects. The code is published on my GitHub account[10]. I conducted tests with the Apache 2.4.23 and Nghttp2 1.14.0 with the same setup as the authors. The software have been fetched from the original websites and compiled in an isolated environment. I noted several differences with the authors' results as shown by Table 1.

[9] https://dl.acm.org/citation.cfm?id=3162826

[10] https://github.com/Dhiver/SlowRate-HTTP2-DoS

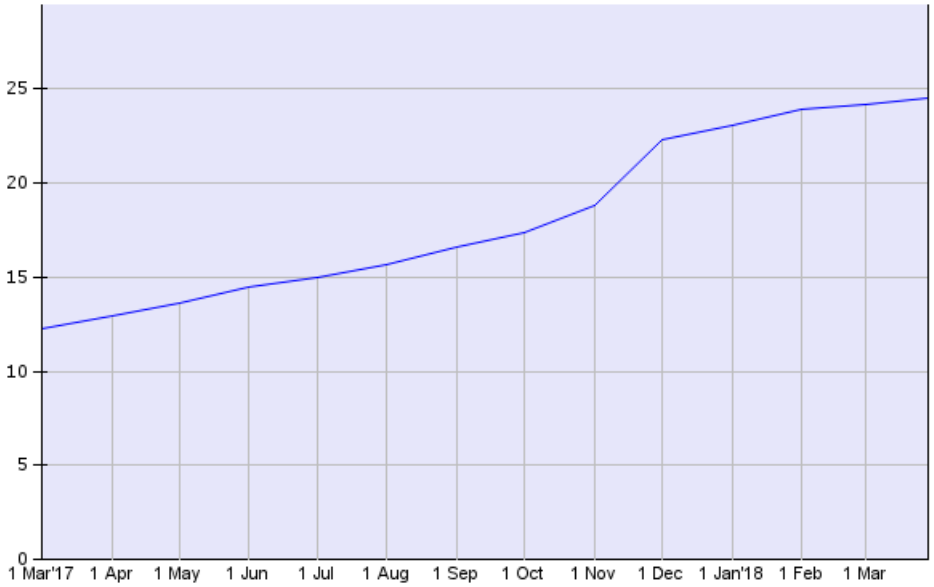


Figure 4: Usage of HTTP/2 for websites by W3Techs.com (29 Mar 2018)

Server	Attack 1	Attack 2	Attack 3	Attack 4	Attack 5	№ conn
Apache	60	120	60	60	5	400
Nghttp2	10	10	10	10	10	>3000

Table 1: Connection waiting time in seconds and № at servers for the five attacks

The waiting time values are less spectacular than the announced ones as well as the number of connections required to feel the whole queue unfortunately.

7.2 Erwin Adi Ph.D. Thesis

"Denial-of-service attack modelling and detection for HTTP/2 services"

The thesis [6] lists four novel flood attacks against HTTP/2. Four machine learning techniques to conduct traffic analysis that has been proven suitable for detection are used (Naïve Bayes, Decision Tree, JRip and Support Vector Machines). Flash-crowded traffic is used to populate the dataset of legitimate network traffic. Different features less specific to HTTP/2 protocol were used such as the number of connections and the flow information. However, the mentioned attacks require a high number of packets and are a more prone to detection as they are based on flooding attacks. Machine learning techniques had high accuracy in classifying network traffic thus making them suitable to detect DoS attacks. Detection is not only based on protocol specific features, other properties are useful such as, the entropy of requests per source IP addresses, the traffic flow (number of packets, their size, …), the behaviour of hosts and the statistics on the traffic. The same detection performance metrics are used here, the recall and FPR rates. The attacks described by Tripathi and Hubballi are much more subtle than what's described by Adi.

7.3 Personnal opinion

Here is my opinion in addition to the comments I have left throughout the sections. Tripathi and Hubballi demonstrated that a single client can create a DoS of widely used HTTP/2 web servers. Their selection of web servers is a good one. The paper is effortless to read, of good

quality and very specific to HTTP/2. It was a good idea to focus on the number of available slot connection. However, where are the Proofs-of-Concept?? The PoC of the attacks, the feature extraction code and the detection system code are missing. I had to implement the attacks in order to verify the claims… How do you expect to be credible of you don't provide concrete proofs? The cleartext version of HTTP/2 is used to describe the interactions but it differs from a real-life environment. When I first read the paper the attacks seemed quite simple and spectacular, but after running my tests I was a bit disappointed. Nevertheless, good job!

8 Avenues for Improvement

In addition to the comments I made in the upper sections, I will mention a few more areas for improvement. Only the chi-square statistical test is mentioned. Today everything is about machine learning, we want to read about machine learning techniques! A Future Works section could have been a good opportunity to talk about the release of a web server detection module for example. No countermeasures were provided at all. We would have liked to see advices such as the configuration of various timeouts in web servers to mitigate the effects in some extents. We also wonder if the discoveries have been reported to the software companies.

9 Conclusion

New layer seven attacks were described in this research. I was interested in learning more about the HTTP/2 protocol and had little knowledge about Slow Rate DoS attacks. This reading awakened my curiosity on both subjects. An overall view is that the HTTP/2 protocol security assessment is still in its very preliminary stages. We are hoping to see more studies in the next couple of months!

10 References

[1] Nikhil Tripathi and Neminath Hubballi. Slow rate denial of service attacks against http/2 and detection. *Comput. Secur.*, 72(C):255–272, January 2018.

[2] Worldwide ddos attacks & cyber insights research report. https://ns-cdn.neustar.biz/creative_services/biz/neustar/www/resources/whitepapers/it-security/ddos/neustar-2017-worldwide-ddos-attacks-cyber-insights-research-report.pdf. Accessed: 2018-03-31.

[3] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred. Statistical approaches to ddos attack detection and response. In *Proceedings DARPA Information Survivability Conference and Exposition*, volume 1, pages 303–314 vol.1, April 2003.

[4] 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)*, pages 1–5, Aug 2015.

[5] Erwin Adi, Zubair A. Baig, Philip Hingston, and Chiou-Peng Lam. Distributed denial-of-service attacks against http/2 services. *Cluster Computing*, 19(1):79–86, March 2016.

[6] Erwin Adi. *Denial-of-service attack modelling and detection for HTTP/2 services*. PhD thesis, Edith Cowan University, 2017 (accessed Febuary 21, 2017).

[7] In-depth analysis of the top four flaws of the next generation web protocol. https://www.imperva.com/docs/Imperva_HII_HTTP2.pdf. Accessed: 2018-03-31.