2017

Intelligent feature selection for detecting http/2 denial of service attacks

Erwin Adi
University of New South Wales

Zubair Baig
Edith Cowan University

Recommended Citation

Adi, E., & Baig, Z. (2017). Intelligent feature selection for detecting http/2 denial of service attacks. DOI: https://doi.org/10.4225/75/5a84e0b095b40

DOI: 10.4225/75/5a84e0b095b40
Adi, E., & Baig, Z. (2017). Intelligent feature selection for detecting http/2 denial of service attacks. In Valli, C. (Ed.). (2017). The Proceedings of 15th Australian Information Security Management Conference, 5-6 December, 2017, Edith Cowan University, Perth, Western Australia. (pp. 57-63).

INTELLIGENT FEATURE SELECTION FOR DETECTING HTTP/2 DENIAL OF SERVICE ATTACKS

Erwin Adi[1], Zubair Baig[2]
[1]Australian Centre for Cyber Security, School of Engineering and Information Technology, University of New South Wales, Australia
[2]Security Research Institute, School of Science, Edith Cowan University, Perth, Western Australia
e.adi@adfa.edu.au, z.baig@ecu.edu.au

Abstract

Intrusion-detection systems employ machine learning techniques to classify traffic into attack and legitimate. Network flooding attacks can leverage the new web communications protocol (HTTP/2) to bypass intrusion-detection systems. This creates an urgent demand to understand HTTP/2 characteristics and to devise customised cyber-attack detection schemes. This paper proposes Step Sister; a technique to generate an optimum network traffic feature set for network intrusion detection. The proposed technique demonstrates that a consistent set of features are selected for a given HTTP/2 dataset. This allows intrusion-detection systems to classify previously unseen network traffic samples with fewer false alarm than when techniques used in literature were employed. The results show that the proposed technique yields a set of features that, when used for network traffic classification, yields low numbers of false alarms.

Keywords: HTTP/2, feature selection Denial of Service, machine learning

INTRODUCTION

Hypertext Transfer Protocol (HTTP) has been the standard for web browser communication since the end of the 20th century. Until recently, most web communications were reliant on HTTP version 1.1, which was designed to transfer texts. As current web sites render large sized content such as audio and video, users routinely experience slow web browsing experience through HTTP/1.1. Hence, the new version, HTTP/2 (Belshe, Peon, & Thomson, May 2015), was designed to deliver web services at higher transfer rates, enhancing the end-user experience.

HTTP/2 has been accepted as the next generation standard for web communications. The protocol had its preliminary version deployed in 2010, was formally published in 2015, and is currently deployed by approximately 10 million websites globally, or 19% of all the websites (Usage of HTTP/2 for websites, November 2017). Major web browsers such as Mozilla, Chrome and Microsoft Edge support the protocol; and popular web sites such as Google services, Facebook and Twitter operate their web services employing the protocol.

As with all technological advances, the threat posed by the adversary class is ever existing. Adversaries can send a large volume of HTTP traffic towards a target HTTP/2 web service, causing resource exhaustion and eventual prevention of access for legitimate users. Such exploit is commonly known as a flood-based attack, or a Denial of Service attack. To detect flood-based attacks, previous studies have applied machine learning techniques. These techniques can learn from data samples, adapt to new environments, produce rule sets, and predict the class of unseen data. In detecting flood-based attacks, these known techniques construct a model of legitimate (normal) traffic, and classify attack traffic as instances where feature values extracted from the network traffic deviate from the baseline legitimate model.

A selection of studies that have previously classified traffic into flood-based and normal (Moore & Zuev, 2005; Mukherjee & Sharma, 2012; Baig, Sait, & Shaheen, 2013; Katkar & Kulkarni, 2013; Al-Jarrah et al., 2014) employed feature ranking and selection techniques to reduce the complexity associated with large scale data classification. Techniques such as Information Gain (Kullback & Leibler, 1951) and Gain Ratio are commonly used for feature *ranking*. The higher the rank of a feature, the more relevant the features are for the traffic classification process. Subsequently, the order of the rank number can be applied to *select* a set of most relevant features.

The main issue introduced with feature ranking and selection based on the above techniques is that they do not always yield the same subset of relevant features for varying datasets (Witten & Frank, 2005, p. 154). Specifically, different members of *{1 . . . n}* features are yielded, given different datasets. Over time, additional dataset samples are required to extend previous knowledge, to construct models that represent the current situation, and to classify with an acceptable degree of accuracy. Hence, there is a need to select a chosen set *{1 . . . n}* of features that will yield a consistent list of members in the set, regardless of the data set choice. This study proposes a feature selection technique to address the aforementioned need. Specifically, this study proposes selection of a chosen set of features, given two different datasets that describe the characteristics of HTTP/2 traffic, to perform better than when HTTP/1.1 features are processed and ranked. The significance of this study is twofold: first, it proposes a novel feature selection technique for HTTP/2 traffic; and second, it examines how the technique can be applied to analyse HTTP/2 traffic based on variable datasets.

BACKGROUND

In detecting HTTP/1.1 flood-based attacks, reported studies have found that attack traffic showed higher number of packets when compared to the number of TCP connections that were established for a given target web service (Jung, Krishnamurthy, & Rabinovich, 2002; Ni, Gu, Wang, & Li, 2013). The introduction of HTTP/2 changed this concept significantly. The novel HTTP/2 mechanisms for communicating parties can be leveraged by adversaries to create previously unseen attack vectors to disrupt these services.

A recent study (Adi, Baig, & Hingston, 2017) showed that HTTP/2 flood-based attacks can be modelled to mimic the number of TCP connections observed on legitimate network traffic. The flood traffic, when examined through methods known in the literature to detect such attacks (Kumar, Joshi, & Singh, 2007; Lakhina, Crovella, & Diot, 2005; Rahmani, Sahli, & Kamoun, 2012), yielded a high number of False Alarms, indicating that the attack traffic mimicked normal traffic, thereby bypassing intrusion-detection systems. This showed that an urgent challenge exists to understand the characteristics of HTTP/2 flood-based attack traffic.

The above study (Adi et al., 2017), henceforth named Stealthy Attack Model, also proposed a set of 42 features as input to machine learning techniques to differentiate attack from normal network traffic. The study showed that the proposed set of features yielded fewer False Alarms compared to when HTTP/1.1 features were ranked and used for classification. These features were obtained from the statistical properties of captured HTTP/2 packets, which were grouped based on 3 features, namely, count, size, and lapse. The count features were obtained from the number of packets captured in an instance of time. The size feature was the total number of bytes of packets captured in an instance. The lapse feature was the time difference between the connection initiation of a packet (indicated by a SYN packet) and the time when the packet was captured. The study ranked the features with both Information Gain and Gain Ratio techniques, and analysed machine learning classification performance when applied on varying sets of ranked features. It showed that machine learning techniques such as Naïve Bayes, Decision Trees, JRip, and Support Vector Machines can be employed to classify HTTP/2 traffic into attack and normal class with good accuracies.

False Alarms is a notable performance measure employed by studies in the field of intrusion detection, to classify traffic into flood-based and normal (Wang, Zhang, Hei, Ji, & Ma, 2016; Manzoor, Kumar, et al., 2017; Suhasaria, Garg, Agarwal, & Selvakumar, 2017; Suganya, 2016; Osanaiye, Choo, & Dlodlo, 2016; Latif, Abbas, Latif, & Masood, 2015). False Alarms, described by the equation shown below, is defined as the percentage of instances incorrectly classified out of the total number of the whole instances S in a dataset. The False Positive FP is the ratio of normal traffic that a machine learning technique incorrectly identifies as attack traffic. The False Negative FN is the ratio of attacks that the technique incorrectly identifies as legitimate traffic.

$$False\ Alarm = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{S}} \times 100\%$$

The percentages of False Alarms are illustrated on the Y-axis of a graph, as a function of 1 to $\{1 \ldots n\}$ features, presented on the X-axis of the graph. A lower percentage of False Alarms implies a better performance of the classifier. Observations based on human intervention are required to analyse which set of features are best employed for classification tasks to yield the lowest percentage of False Alarms. This was demonstrated in the Stealthy Attack Model analysis: different features sets were chosen to obtain the desired performance when different machine learning techniques were employed (Adi et al., 2017, Sec. 5.3).

The aforementioned study demonstrated that the results were observed and analysed to compare the performance when different sets of features were selected. On the other hand, the study reported in this paper places the observation before the classification results are obtained. The study also highlights that relationships exist between various features of a data set, and leverages this relationship to select relevant features.

DATA PROCESSING AND FEATURE SELECTION

Machine learning performance can be optimised through analysing relationships that exist in the data (Witten & Frank, 2005, p. 78). Learning involves examining relationships between objects rather than between instances. Of particular interest in this study is the concept of a sister in any family. Traditionally, when translated to a dataset representation, a sister is technically described as when two people can be labelled as *sister = yes* or *sister = no*. However, this representation can be prohibitive in terms of storage cost: describing the sister relationship of 100 people would require 100^2 = 10,000 instances. Describing such a relationship in real datasets poses a further problem: the number of objects that form relationships in datasets are unknown (Witten & Frank, 2005, p. 48). The solution to this problem is to inspect the objects and their relationships in the dataset before selecting features, to identify whether the two objects have the same parents (Witten & Frank, 2005, p. 48). This is illustrated in Table 1.

Table 1. A representation of a sister relationship

Name	Parent 1	Parent 2	Name	Parent 1	Parent 2	Sister
P	A	B	Q	A	B	Yes

Table 1 also lists P and Q as two objects, to examine if they are sisters. For the purpose of simplicity, the illustration ignores the gender of the objects. The table describes that P and Q are sisters because they have the same parents. This observation serves as the framework of the proposed feature selection technique, which is presented in the next section.

PROPOSED FEATURE SELECTION TECHNIQUE

This study observes that a sister relationship exists between count and size feature groups of the two datasets that were acquired from the Stealthy Attack Model (namely Stealthy Attack-1 and Stealthy Attack-2 that are discussed in the next subsection).

Referring to Table 1 as the framework, this study proposes that *P = count* and *Q = size*. These two parameters are sisters since they have the same parents. Parents were defined based on features of network traffic. One parent (A) was defined from packets generated by bots, that are uniform in their number of packets and packet sizes, compared to those generated by heterogeneous devices. In the Stealthy Attack Model, the bots were implemented by at most two C-libraries, i.e. curl (Stenberg, 1996–2016) and nghttp2 (Tsujikawa, 2015). This contrasts with real network traffic which is generated by different devices from different manufacturers, having different hardware/software implementations. Real network traffic generates heterogeneous packet sizes. Instead, the bots were generated through a model that dictates how client-to-server packets are formed. Particularly, Stealthy Attack Model launched only one HTTP/2 packet type (namely window update packet), where the packet size was defined by the C-libraries that implemented the bots. Furthermore, there were a limited number of bots that generated such network packets, causing the model to generate a less dispersed number of packets compared to normal traffic. This is illustrated in Figure 1, which is adapted from the Stealthy Attack Model study (Adi et al., 2017, Sec. 5.2).

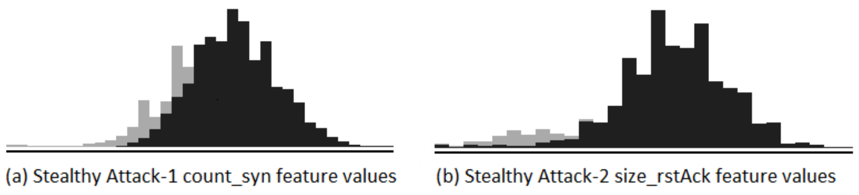

Figure 1. Distribution of attack (grey) and normal (black) feature values

Figure 1 illustrates the distribution of two different feature values, in this example, the count_syn (the number of SYN packets in an instance) and the size_rstAck (the total size of RST ACK packets in an instance), for the stealthy attack dataset. The X-axis represents various ranges of feature values, while the Y-axis describes the number of instances in the dataset that encompass such range of values for a given feature. The grey bars represent attack traffic generated by bots, while the black bars represent normal traffic. Figure 1 shows that the normal traffic is more distributed than the attack traffic: the black bars span wider than the grey bars; while the black bars forms a distributed bell-curve, the grey bars are lower relative to the black bars, and plane relative to the X-axis. This illustrates that the attack traffic is less heterogeneous in terms of packet count and sizes, when compared to normal.

The other parent (B) was defined as attacking bots attempting to mimic normal flows of normal data traffic. In the Stealthy Attack Model, two parameters were defined to mimic normal traffic: a stealthy factor and a delay between connections. The stealthy factor *sf* was defined as a flood of HTTP/2 packets launched with a probability of *1/sf*.

A delay between connections of *d* was prescribed as *d* ms added between successive TCP connections of a given session. These parameters defined the bots capable of controlling the number of packets launched from a malicious client towards a target server. As the bot traffic was controlled (i.e., operated below the radar), the number of packets and their sizes are less heterogeneous than those of normal traffic (as Figure 1 shows).

Henceforth, *P* and *Q* are sisters, since they both were derived from non-heterogeneous bots that attempted to mimic normal traffic.

Step Sister Algorithm

This study proposes the Step Sister technique to select a set of features based on inter-sister relationships, given two different datasets. The precondition of the algorithm is to have two sets of ranked features. In this study, the datasets Stealthy Attack-1 (S1) and Stealthy Attack-2 (S2) were ranked through application of the Information Gain technique (Kullback & Leibler, 1951).

The output of the algorithm is to have a *ChosenSet* of features; this set is initially assigned to an empty set. While there is no definition of how many features is sufficient, a finite number of 5 features was considered in this study for demonstration.

Two sets of ranked features were employed from the Stealthy Attack datasets (Adi et al., 2017). These are the Stealthy Attack-1 dataset and the Stealthy Attack-2 dataset. These features were ranked through application of the Information Gain technique, resulting in two ranked lists, S1 and S2, respectively. For the purpose of this study, the two features chosen to analyse HTTP/1.1 traffic, i.e. the count_app and the count_syn features (explained in the next subsection), were not selected as part of the *ChosenSet*. Furthermore, because this study observed that the count group has a sister association with the size group, this study disregarded the size_app and the size_syn features. These are shown on lines 5 - 7 of the algorithm.

To define a sister relationship, the algorithm examines the feature groups and feature types. Feature groups are features that share the same characteristics such as “count” and “size”. Feature types signify the packet types where the features were extracted from. For example, rstAck is a feature type, which is extracted from a network packet carrying RST ACK flag. Hence, a feature named count_rstAck is obtained from the number of packets (i.e. count) carrying RST ACK flags, observed in an instance of time. The sister association is found to be true when a feature *f* from one of the ranked list *S* being examined belongs to the “count” group, and the same type of feature where its group is “size” is found in the *ChosenSet*. The converse situation where *f* belongs to the “size” group, also demonstrate a sister association when the same type of feature of group “count” is found in the *ChosenSet*.

Henceforth, the Step Sister algorithm can be simplified as follows. The first step is to clean the two ranked list *S1* and *S2*, by eliminating a feature is its sister is already in the *ChosenSet*, starting from the highly ranked feature from each list. It selects a feature from the two lists *S1* and *S2* that is ranked higher. These steps iterate until there is a handful number of features in the *ChosenSet* with size *n*.

Stealthy Attack Datasets

The two attack models proposed in the Stealthy Attack Model were named Stealthy Attack-1 and Stealthy Attack-2. The Stealthy Attack datasets described attack and normal HTTP/2 traffic classes: the attack data was generated out of two attack models; and the normal data was obtained from simulating 5,200 bots that mimicked human behaviour when online.

The Stealthy Attack-2 dataset extended the Stealthy Attack-1 dataset through employing attacking bots that mimicked the distribution value of a highly relevant feature observed from the Stealthy Attack-1 data analysis. In both datasets, the features were ranked through employing both Information Gain and Gain Ratio algorithm. The result showed that 42 features could detect HTTP/2 flood-based attacks better, i.e. fewer False Alarms than when two HTTP/1.1 features were employed. The two HTTP/1.1 features used as a comparison were count_app, i.e. the number of Application Data packet observed in an instance, and count_syn, the number of TCP connection initiation observed in an instance.

RESULTS AND ANALYSIS

The *ChosenSet* from the Step Sister algorithm yielded the following set of features:

- size_rstAck, is the total size (in KB) of packets carrying RST-and-ACK TCP flags observed in one instance.

- size_tlsKey, is the total size (in KB) of TLS packets carrying key exchange observed in one instance
- count_encryptedAlert, is the total number of TLS packets carrying Encrypted Alert flags observed in one instance.
- lapse_rstAck max, is the maximum duration of time between packets carrying RST-and-ACK TCP flags within an observed instance, and a packet carrying SYN flag signifying its first connection initiation.
- size_tlsHello, is the total size (in KB) of TLS packets carrying Hello packet type observed in one instance.

The above set of features was employed to classify traffic described for both datasets, i.e. the Stealthy Attack-1 and Stealthy Attack-2 datasets. The classification analysis employed Weka (University of Waikato, 1993–2016), software that provides a collection of machine learning techniques, to analyse the classification performance in terms of False Alarm. Four machine learning techniques were employed: Naïve Bayes (NB), Decision Tree J48 (DT), JRip, and Support Vector Machines (SVM). Two other feature ranking algorithms that were employed in the Stealthy Attack Model, Information Gain (IG) and Gain Ratio (GR), serve as a comparison to analyse the classifier performance.

The Step Sister algorithm did not aim to yield better performance than what the Stealthy Attack Model study yielded. This is shown in Table 2. The False Alarm yielded by the machine learning techniques was compared when the features were selected and ranked by the Step Sister (SS) algorithm, Information Gain (IG) and Gain Ratio (GR). The values obtained for the SS columns were the outcomes of this study, while the values shown for the IG and GR columns were the outcomes of the Stealthy Attack Model study. To compare the correct values, only the most relevant 5 features were selected from the list of ranked features by IG and GR. The table shows that the numbers in column SS are not consistently lower than the values on the other columns. Hence, the SS algorithm did not seek to optimise the performance obtained in the Stealthy Attack Model.

Table 2. False Alarms produced (%) when different algorithms were employed

	Stealthy Attack-1			Stealthy Attack-2		
	IG	GR	SS	IG	GR	SS
NB	0.2892	0.2410	0.2410	0.0519	0.0519	0.0778
DT	0.0482	0.0723	0.0482	0.0519	0.0519	0.0519
JRip	0.0482	0	0.0723	0.0259	0.0259	0.0259
SVM	0	0	0	0	0.0778	0

Table 3. False Alarms produced (%) when different feature sets were employed

	Stealthy Attack-1		Stealthy Attack-2	
	HTTP/1.1	SS	HTTP/1.1	SS
NB	0.2651	0.2410	0.5189	0.0778
DT	0.1687	0.0482	0.2335	0.0519
JRip	0.1205	0.0723	0.2335	0.0259
SVM	0	0	0.3373	0

Consistent results can be seen when the machine learning performance was compared to those when the HTTP/1.1 features were employed. This is illustrated in Table 3. All of the False Alarm yielded by the machine learning techniques were lower when the features were selected by the SS algorithm, compared to those when HTTP/1.1 features were employed. Hence, the *ChosenSet* of five features consistently yielded better performance.

The *ChosenSet* features are more consistent than the set of features employed in the Stealthy Attack Model study: first, the Step Sister algorithm selected the same *ChosenSet* of features to be employed by machine learning techniques. This is different to the methods employed by the Stealthy Attack Model study, where different sets of features *{1 . . . n}* must be selected. Second, in the Stealthy Attack Model study, observations were required to choose the size n of the set *{1 . . . n}*. In contrast, the same five features selected by the Step Sister algorithm in this study were employed by the machine learning techniques. This demonstrated the aim of the study: to yield the same set of features to analyse different datasets.

CONCLUSION

This study proposed a technique, namely, *Step Sister*, which yielded a set of features to aid in machine learning-based classification of HTTP/2 flooding traffic. The analysis employed two datasets that described the characteristics of HTTP/2 flooding and legitimate traffic, respectively. Since the Step Sister technique yielded a consistent set of features for machine-learning based classification, intrusion-detection systems that employ this technique can operate without requiring human intervention to choose the size and members of the feature set. Furthermore, the proposed technique was tested on two varying datasets. The study demonstrated that the Step

Sister technique analysed both datasets to yield a chosen set of features. Machine learning techniques that employed these set of features yield lower False Alarms than when techniques known in literature were employed to analyse HTTP/2 traffic.

REFERENCES

Adi, E., Baig, Z., & Hingston, P. (2017). Stealthy denial of service (DoS) attack modelling and detection for HTTP/2 services. *Journal of Network and Computer Applications, 91,* 1–13.

Al-Jarrah, O., Siddiqui, A., Elsalamouny, M., Yoo, P., Muhaidat, S., & Kim, K. (2014, June 30-July 3). Machine-learning-based feature selection techniques for large-scale network intrusion detection. *Distributed Computing Systems Workshops, Madrid, Spain* (p. 177-181). IEEE Xplore.

Baig, Z. A., Sait, S. M., & Shaheen, A. (2013). GMDH-based networks for intelligent intrusion detection [Journal Article]. *Engineering Applications of Artificial Intelligence*, *26* (7), 1731-1740.

Belshe, M., Peon, R., & Thomson, M. (May 2015). *Hypertext Transfer Protocol version 2 (HTTP/2)* (Report No. RFC 7540). Internet Engineering Task Force (IETF).

Jung, J., Krishnamurthy, B., & Rabinovich, M. (2002, May 7-11). Flash crowds and Denial of Service attacks: Characterization and implications for CDNs and web sites. In *Proceedings of the 11th International Conference on World Wide Web, Honolulu, Hawaii* (p. 293-304). New York: ACM.

Katkar, V. D., & Kulkarni, S. V. (2013, December 12-14). Experiments on detection of denial of service attacks using naive bayesian classifier. In *International Conference on Green Computing, Communication and Conservation of Energy, India* (p. 725-730). IEEE Xplore.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22* (1), 79–86.

Kumar, K., Joshi, R., & Singh, K. (2007). A distributed approach using entropy to detect DDoS attacks in ISP domain. In *International Conference on Signal Processing, Communications and Networking, Honolulu, Hawaii* (pp. 331–337). IEEE Xplore.

Lakhina, A., Crovella, M., & Diot, C. (2005). Mining anomalies using traffic feature distributions. In *ACM SIGCOMM Computer Communication Review* (Vol. 35, pp. 217–228).

Latif, R., Abbas, H., Latif, S., & Masood, A. (2015). EVFDT: an enhanced very fast decision tree algorithm for detecting distributed denial of service attack in cloud-assisted wireless body area network. *Mobile Information Systems*, *2015*.

Manzoor, I., & Kumar, N. (2017). A feature reduced intrusion detection system using ANN classifier. *Expert Systems with Applications*, *88*, 249-257.

Moore, A. W., & Zuev, D. (2005). Internet traffic classification using bayesian analysis techniques [Conference Proceedings]. In *ACM SIGMETRICS Performance Evaluation Review* (Vol. 33, p. 50-60). ACM.

Mukherjee, S., & Sharma, N. (2012). Intrusion detection using naive bayes classifier with feature reduction [Journal Article]. *Procedia Technology, 4, 119*-128.

Ni, T., Gu, X., Wang, H., & Li, Y. (2013). Real-time detection of application-layer DDoS attack using time series analysis [Journal Article]. *Journal of Control Science and Engineering, 2013,* 4.

Oikonomou, G., & Mirkovic, J. (2009, June 14-18). Modeling human behavior for defense against flash-crowd attacks. In *IEEE International Conference on Communications, Dresden, Germany* (pp. 1–6). IEEE Xplore.

Osanaiye, O., Choo, K. K. R., & Dlodlo, M. (2016). Distributed denial of service (DDoS) resilience in cloud: review and conceptual cloud DDoS mitigation framework. *Journal of Network and Computer Applications*, *67*, 147-165.

Rahmani, H., Sahli, N., & Kamoun, F. (2012). Distributed Denial-of-Service attack detection scheme-based joint-entropy [Journal Article]. *Security and Communication Networks*, *5* (9), 1049-1061.

Saleh, M. A., & Abdul Manaf, A. (2015). A novel protective framework for defeating HTTP-based denial of service and distributed denial of service attacks. *The Scientific World Journal.*

Stenberg, D. (1996–2016). *cURL* [Software]. Retrieved from https://curl.haxx.se/download.html

Suhasaria, P., Garg, A., Agarwal, A., & Selvakumar, K. (2017). Distributed Denial of Service Attacks: A Survey. *Imperial Journal of Interdisciplinary Research*, *3*(3).

Tsujikawa, T. (2015). *Nghttp2: HTTP/2 C library* [Computer Program]. Retrieved from https://nghttp2.org/

University of Waikato. (1993–2016). *Weka (version 3.8)* [Software]. Retrieved from http://www.cs.waikato.ac.nz/ml/weka/downloading.html

Usage of HTTP/2 for websites. (n.d.). Retrieved November 02, 2017, from https://w3techs.com/technologies/details/ce-http2/all/all

Wang, Y., Zhang, Y., Hei, X., Ji, W., & Ma, W. (2016). Game strategies for distributed denial of service defense in the cloud of things. Journal of Communications and Information Networks, 1 (4), 143{155. Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques* [Book]. Morgan Kaufmann.

Zhou, W., Jia, W., Wen, S., Xiang, Y., & Zhou, W. (2014). Detection and defense of application-layer DDoS attacks in backbone web traffic [Journal Article]. *Future Generation Computer Systems*, *38,* 36-46.