Detection of Virtual Private Network Traffic using Machine Learning

Shane Miller*, Kevin Curran and Tom Lunney

School of Computing, Engineering & Intelligent Systems, Ulster University, UK
*Corresponding Author email: miller-s1@ulster.ac.uk

Abstract: Detecting unauthorized users can be problematic for techniques that are available at present if the nefarious actors are using identity hiding tools such as anonymising proxies or Virtual Private Networks (VPNs). This work presents computational models to address the limitations currently experienced in detecting VPN traffic. A model to detect usage of virtual private networks (VPNs) was developed with a Multi layered perceptron neural network that was trained using flow statistics data found in the Transmission Control Protocol (TCP) header of captured network packets. Validation testing showed that the presented models are capable of classifying network traffic in a binary manner as direct (originating directly from a user's own device) or indirect (makes use of identity and location hiding features of VPNs) with high degrees of accuracy.

1. Introduction

Virtual Private Networks (VPNs) are a common method for criminals and other bad actors to disguise their online activities [1,2]. This is helped along by the increase in ease of use of VPNs; they are no longer just a tool for remotely accessing enterprise resources when travelling for work or when working from home. In fact, this could be a use-case for a criminal. If they wish to remotely access an enterprise network in order to steal company and trade secrets, they can use a VPN to hide their own location or to make it appear as if someone else was infiltrating the network [3]. There have been a few notable cases of this happening in recent years, such as the Sony Pictures incident from 2014, where confidential data including personal information about employees was stolen [4, 5]. Other attacks of note are the various data breaches which have been occurring for the last number of years, such as the LinkedIn breach [6]. Approximately 167 million account details including emails and passwords were stolen. It is not known whether the attacker(s) were using a VPN service to hide their location.

There are many anonymity technologies with most being based on networks called "mix" networks. These 'Mix networks' route packets in such a way as to make it extremely difficult a link between the source of the request. This works via through intermediaries and 'mixing' packets from participant. This makes it very difficult for eavesdroppers to trace end-to-end communications [7, 8]. Low latency systems include the popular anonymous communication system Tor as well as HTTP/SOCKS proxy services and Virtual Private Networks (VPNs) [9]. Systems such as Tor fall under the category of multi-hop anonymous communications models, while HTTP/SOCKS proxies and VPNs generally fall under the category of single-hop anonymous communication models. Proxy servers that are used to provide anonymisation are based on another type of proxy known as an "open" proxy. Open proxies are a proxy that is available to any user on the Internet. They are mostly used to set up anonymous proxy websites and categorised as a single-hop anonymous communication model. There are several different implementations of VPNs for providing anonymous communications [10, 11, 12]. The intended use for VPN implementations was to allow an organisation's workers to securely access internal network resources from outside of the internal network i.e. remote access. This is achieved through setting up a connection called a tunnel between the user's PC and the organisations servers. VPNs however can also be used as an anonymous communication system in an equivalent manner to an anonymous proxy server. The main difference between the two methods is in the VPN's tunnelled connection. The tunnelled connection between the user and the VPN server is encrypted.

IP blocking is a basic technique used to combat malicious threats to networks and it is one of the most common techniques for protecting networks [13]. Using this method, an IP address or a range of IP addresses can be blocked from accessing resources located on a web server or on an organisation's internal network. The IP block can be rendered ineffective by using proxies or VPNs. The user's IP address is typically sent out as a source IP address in the network packet containing the request to a web server. However, when using a proxy or VPN, this request is first sent to the proxy server which then forwards it on towards the web server. So, the blocked IP address of the user is not actually making any direct contact with the web server running the IP filter. The offending proxy or VPN IP address can be blocked, but this act of blocking the IP address can be made redundant. Upon discovering that their preferred proxy IP has been blocked, the user can simply switch to a different proxy or VPN provider. Unless preventative action is taken, which will cost a significant amount of time and effort, the user can continue to switch in order to maintain their access. Another method of securing networks is the use of Access Control Lists (ACL). These are usually implemented alongside IP blocking techniques. Network traffic is matched with an ACL to discriminate which network packets are forwarded onwards. Each packet is examined and compared to the policies outlined in the ACL to determine whether it should be allowed or blocked [14]. This is a very rigid form of network security that relies on a lengthy setup. Specifying what is acceptable and what is not takes a large amount of time due to the complexity and sheer number of network protocols that exist. Filtering based on the protocols included in the network packets can be rendered ineffective by VPNs due to how they encapsulate protocols within other protocols. Depending on the exact implementation of ACL, the network topology for the entire enterprise network will not be defined so the ACL cannot determine what is a member of the network. Proxies

can easily take advantage of this and the ACL is also susceptible to the user switching proxy provider to circumvent any blocks. Software based packet inspection is another method that can be used to detect and block usage of a proxy or a VPN. Deep Packet Inspection (DPI) is a popular method for securing networks against network packets containing malicious items such as viruses and other malware that are contained within payloads [15]. DPI examines and manages network traffic as it enters the network in a form of packet filtering that identifies, classifies and blocks packets that contain data (such as the aforementioned viruses) within their payload that goes against pre-arranged policies. This examination occurs at checkpoints located around the network and decisions based on rules assigned by an organisation occur in real-time based on the contents of the packet's payload. Previously, packet scanning software had the limitation of only scanning the packet's header, which contains the information necessary for transmission, but does not contain anything related to its contents. By scanning the packets contents, messages and other information can be extracted and used to identify the specific application or service it comes from. The rules that DPI algorithms operate by were string based, however using regular expression matching improves content scanning speeds [16]. As powerful as DPI can be, it is defeated by packets that make use of encryption to conceal their contents. VPNs are particularly effective at bypassing DPI as well as some proxies which support HTTPS, bringing the effectiveness of DPI into question [17, 18, 19, 20]. Over the past decade the research and networking communities have investigated and developed several classification approaches based on multiple algorithms. This has come about because the traditional approach of using TCP and UDP network ports to classify Internet applications has become less accurate. Newer applications that are being developed do not have ports registered to them by IANA and instead make use of ports that are already registered to other applications. The exhaustion of IP version 4 addresses has also contributed to this as organisations and application developers move to mitigate the effect [18]. Classification algorithms typically require training based on previously labelled data. For classification of network traffic, the network packets form the basis of the dataset. The contents can consist of unedited packet headers, with the information contained being used as the training features. They can also consist of statistical information calculated from streams of packets called flows. Efforts to classify Internet applications have largely been successful, with several datasets being created to represent most of the applications available. However, datasets representing anonymous communication systems are mostly non-existent and research into classification of anonymous traffic is still an emerging research area. A major limitation into classification of this type of network traffic is the use of encryption, which renders the payload of packets unusable as a training feature. Using machine learning capabilities and different feature formats, it should be possible to overcome this limitation. Packet header information such as the sequence and acknowledgement numbers and the general size of the data can potentially be used to train a machine learning algorithm. There is also the option of using flow-based features to enhance the potential training and detection accuracy of an algorithm [21].

The ability to detect whether a VPN has been used or not could be helpful in the pursuit of attackers such as those just mentioned. This research therefore presents methods that would aid in the detection of VPN technologies that are being used to hide an attacker's identity. While VPNs have legitimate uses, such as connecting to a business network from a remote location, they are still abused by criminals who use them to commit crimes whilst remaining undetected and unidentified. We demonstrate the detection and classification of VPN traffic using a Multi-layered Perceptron Neural Network for the classification of the traffic. Our approach can accurately identify threats in real time with as few false positive and negative results as possible.

2. Virtual Private Networks (VPNs)

A Virtual Private Network (VPN) provides private networks of resources and information over any public network [21, 22]. It enables a remote machine on network X to tunnel traffic, that might not normally be able to be sent across the Internet, to a gateway machine on network Y and appear to be sitting, with an internal IP address, on network Y. The gateway machine receives traffic to this internal IP address, and sends it back to the remote machine on network X [23]. This itself does not provide much security. Intercepting these tunnelled packets would still allow for the contents of the private packets to be intercepted and exposed by a third party. To overcome this, the private packets need to be encrypted and above that, some form of authentication needs to be used. VPN protocols vary in their support for encryption and authentication schemes. Each of the following sections will discuss some example algorithms and schemes supported by each VPN protocol.

2.1 PPTP

The Point-to-Point Tunnelling Protocol (PPTP) is a link layer VPN protocol that is designed to tunnel Point-to-Point Protocol (PPP) connections through an IP network, creating a VPN connection [10, 23]. The final packets are sent over IP from the client to the gateway PPTP server and back again. PPTP does not provide any methods for keeping data confidential or for providing strong authentication. The Microsoft implementation that was included with Windows NT provides a framework for negotiating authentication and encryption algorithms between server and client which relies upon existing negotiations contained within extensions and enhancements of PPP [25]. Some example authentication algorithms are the Password Authentication Protocol (PAP), the Challenge-Handshake Authentication Protocol (CHAP), MS-CHAPv1/v2, Microsoft's implementations of CHAP, and Extensible Authentication Protocol (EAP). CHAP and MS-CHAPv1/v2 have faced extensive scrutiny over the years [22, 23, 26, 27]. PAP transmits the username and password from the client through an unencrypted channel which leaves it vulnerable to eavesdropping attacks. This leaves it in the position where it can only be used as a last resort. Due to the vulnerabilities that have been found in the authentication and

encryption algorithms it uses, PPTP does not see widespread use anymore.

2.2 L2TP

The Layer 2 Tunnelling Protocol (L2TP) is also a link layer VPN that extends the PPP model by combining features of PPTP with features of the Layer 2 Forwarding (L2F) protocol [28]. L2TP functions similarly to PPTP. Higher level protocols, commonly PPP connections, are encapsulated within an L2TP tunnel by setting up an L2TP session. The L2TP packets in turn, including both the payload and the L2TP header are transported within a UDP packet. L2TP is also similar to PPTP in that it does not provide any methods for confidentiality or authentication and instead inherits existing protections from PPP. A protocol suite called IPsec was introduced to provide improved authentication and confidentiality over the PPP methods [12]. The original PPP methods used by L2TP were found to be vulnerable to a Denial of Service (Dos) attack which involved transmitting a request to stop the connection using the correct identification in order to terminate the VPN session [20]. This was a vulnerability that was solved in an updated version of L2TP called L2TP version 3 (L2TPv3). The new version included an optional authentication and integrity check that nullified the vulnerability. L2TP is often combined with another authentication and encryption protocol suite called Internet Protocol security (IPSec) [29].

2.3 IPsec

IPsec includes a collection of standardised protocols for mutual authentication between two hosts at the beginning of a VPN session and for the negotiation of cryptographic keys used to enable encryption for the session [29]. Data is kept secure by authenticating network packets to make sure of the integrity of the packet and that encapsulation has been implemented correctly. There are two modes in which IPsec can provide this functionality: transport mode and tunnel mode. In transport mode, the original packet is edited to include a new IPsec header in the original IP header. This additional header contains the information needed to perform authentication and integrity checking. In comparison, tunnel mode provides more flexibility. In tunnel mode, the entirety of each original IP packet is encapsulated inside a new IP packet consisting of a new IP header and the IPsec header [29]. This adds a layer of abstraction from the original IP packet's contents therefore providing confidentiality for the payload. To determine which mode is to be used during a connection, security information defining the modes that each end point supports needs to be exchanged. This is referred to as a security association. It contains information on the mode of IPsec to be used, the encryption algorithms to be used and the encryption keys used to set up the encryption. Exchange of this information is completed using the Internet Key Exchange (IKE) protocol.

2.4 OpenVPN

OpenVPN[1] [30] has a simple configuration and the mixture of enterprise-level security, usability and other features, plus its support for most of the operating systems that are available, it is widely regarded as among the best VPN solutions [31]. OpenVPN makes use of Hash-based message authentication codes (HMAC) in combination with the SHA1 hashing algorithm for ensuring packet integrity. OpenVPN has two authentication modes. In mode one, a pre-shared static key is used to provide authentication and encryption. In mode two, SSL/TLS mechanisms are used for authentication and key exchange[2] [31]. In static key mode, a pre-shared key is shared between both hosts before the tunnel is set up. This static key contains four independent sub-keys: HMAC send, HMAC receive, encrypt and decrypt. The preferred mode of operation is mode two which uses SSL/TLS. In this mode an SSL session is established requiring both hosts to present their own authentication certificate. If the authentication of the hosts succeeds, negotiation and exchange of the encryption/decryption and HMAC keys begins. Rather than the keys being static as in mode 1, in mode 2 the keys are randomly generated either by OpenSSL's *RAND_bytes* function or by using the TLS pseudorandom function (PRF) alongside random source material from both hosts. The keys are then exchanged over the SSL/TLS connection and the tunnel forwarding process begins. The data to be encrypted and transferred in the tunnel includes a 64-bit sequence number and the payload data consisting of an IP packet or Ethernet frame. Encryption of the tunnel packets is carried out using the Blowfish secret key block cipher [32]. OpenVPN then multiplexes the SSL/TLS session that is used for authentication and key exchange with the encrypted tunnel data. SSL/TLS is designed to operate using a reliable transport protocol so OpenVPN provides a reliable transport layer on top of UDP. The actual IP packets are tunnelled over UDP without an added reliability layer after they have been authenticated with a HMAC as the IP packet forwarder has been designed to operate over an unreliable transport layer.

3. VPN classification

A dataset consisting of TCP packets captured using the packet analysis tool Wireshark from an OpenVPN connection was created and tested using the exact same Azure machine learning tools. The results for this showed that the network was overfitting the problem as it was achieving 100% classification accuracy for both VPN traffic and non-VPN traffic. In external validation tests, the network was essentially guessing, as it was classifying every sample as having come from a VPN. In order to overcome this problem, it was hypothesised that a new dataset consisting of TCP flow records/statistics would be more appropriate for analysis. Flow statistics provide a high-level view of network communications by reporting the addresses, ports and byte and packet counts contained in those communications [42]. This data can be especially valuable when network traffic is being encrypted which can be the case with VPN traffic. Wireshark formed the basis of the packet capture for this newer dataset as was also the case for the first dataset. The

[1] https://openvpn.net/

[2] https://openvpn.net/index.php/open-source/documentation/security-overview.html

computer system used to capture the traffic was an Ubuntu 16.04 based virtual machine running on a Windows 10 host. The network connection used in the experiment is a virtualised Intel PRO gigabit ethernet card. Linux was used as it allows for a finer degree of control over some of the internal systems included such as the networking stack. Using some built in tools, it is easy to automate connections and disconnections to different networks and different network interfaces. This was a particularly helpful feature when dealing with the capture of VPN based packets. In normal operation, a connection to a VPN starts with a typical TCP "hello" sequence and key exchange. Once the connection is setup, it is only taken down whenever the user stops using the VPN. The connection is one long TCP connection between the user's machine and the VPN server.

3.1 OpenVPN using Stunnel

Stunnel[3] is an open source, multiplatform application that is designed to add SSL/TLS encryption capability to clients and servers that do not natively support the SSL/TLS protocols. While OpenVPN itself has support for SSL/TLS, techniques such as Deep Packet Inspection (DPI) have the potential to detect OpenVPN when using SSL/TLs [47]. Stunnel can be used to overcome this and present the traffic to DPI frameworks as normal SSL web traffic running on port 443. This gave rise to the question of whether a similar method of classification that was used to classify OpenVPN traffic using a neural network could also be trained to recognize OpenVPN traffic that was using Stunnel. To use Stunnel, the user must install and configure the application on both the OpenVPN server and on whatever OpenVPN client they are using to connect to the VPN. On Linux this involves installing the application by downloading the *stunnel4* package, creating and sharing a new OpenSSL certificate between the client and the server, creating and editing Stunnel config files and configuring the firewalls of both the server and client to allow the Stunnel traffic to be transported.

3.2 Dataset

As with the previous experiments, a dataset containing network traffic from Stunnel OpenVPN connections and non-VPN traffic is required to train the neural network. With the ground work already done with the setup of the OpenVPN server on AWS for the previous experiment, this was relatively simple. The Streisand VPN package also contained everything necessary to setup Stunnel for use with OpenVPN, only requiring a few configuration files to be modified. Once the VPN was setup and the connection stable, capture of the network traffic began using the same method as used for the OpenVPN data capture. Wireshark was used to capture network packets; the VPN was set to disconnect and reconnect every 10 minutes and automatic browsing script was used to generate traffic from the same selection of websites. Once the packets were captured, they were processed using the TCP flow export tool NetMate in order to gain flow statistics of the new data. The result of this data capture was a total dataset of 3,952 samples, of which 1,931 were Stunnel OpenVPN and 2,021 were non-VPN. This dataset was then loaded into Weka.

3.3 Feature Selection

Feature selection was applied to the capture data in order to reduce the number of features produced by NetMate. Again, the same Weka technique used for the OpenVPN experiment was used. This was the *CorrelationAttributeEval* model which was also operating under the same threshold of 0.5. The resulting features are displayed in Table 1. The feature selection for the Stunnel data appears to be largely different to the features selected for the original VPN dataset. Some attributes make a reappearance, such as *duration*, but with a different correlation coefficient. Some of the attributes selected this time have not been seen before which would seem to indicate that there is a difference in how Stunnel modifies the OpenVPN connection.

Attribute Name	Correlation Coefficient
min_fpktl	0.992
duration	0.937
max_fpktl	0.913
max_idle	0.78
max_biat	0.763
std_idle	0.719
max_fiat	0.673
mean_idle	0.575
min_idle	0.562
mean_fpktl	0.561
mean_active	0.512
max_active	0.511
std_fpktl	0.506

Table 1: Correlation Coefficients for Stunnel attributes

Following the same steps used in the previous experiment, the dataset was resampled into separate training, testing and validation sets. The training set contains 3160 samples, the testing set contains 633 samples and the validation set contains 127 samples after resampling.

3.4 Neural Network setup

For this experiment the goal was to examine how well the model developed in the previous experiment could also perform the same with network traffic from a different source. Therefore, the neural network model used in the previous experiment was reused without any modification. Weka was instructed to create a fully connected network with a hidden layer which sums together the number of attributes with the number of classes and divide the result by 2. In this instance there are 13 attributes and 2 classes which results in 15 divided by 2 which is 7.5. Weka rounds down to the nearest whole number so the number of hidden nodes is set to 7.

Once at this stage, the model is ready to be trained using the dataset. In the previous experiment, the model was trained, tested and validated using three resampled sets of data. The same method was used for this model with additional tests

[3] https://www.stunnel.org/

being run using 10-fold cross-validation and Leave One Out Cross Validation (LOOCV). On initial testing using these validation methods, the results gathered showed that the model was getting unrealistically high accuracy, possibly showing signs of overfitting of the model to the problem. To remedy this, the learning rate and then the momentum of the model were lowered from 0.1 to 0.01.

3.5 Results

Table 2, Table 3 and Table 4 show the results of each validation method used once the neural network had been finally trained using the updated configuration. Table 5, Table 6 and Table 7 show the confusion matrices for each of the tests.

Correctly Classified Instances	98.4252%
Incorrectly Classified Instances	1.5748%
Average True Positive Rate	0.968
Average False Positive Rate	0.000
Average Precision	1.000
Average Recall	0.968
Average F-Measure	0.984

Table 2: 80/20 split Validation test results

Table 2 shows the results gathered from Weka for the test that used an 80/20 percentage split on the dataset to create separate training, testing and validation sets. The results shown are taken from the final validation set test, which uses data that was kept separate from the training and tuning of the model in order to simulate as close as possible the real-world performance of the model. The overall accuracy of the model was shown to be 98.42%.

Correctly Classified Instances	97.8998%
Incorrectly Classified Instances	2.1002%
Average True Positive Rate	0.969
Average False Positive Rate	0.012
Average Precision	0.987
Average Recall	0.969
Average F-Measure	0.978

Table 3: 10 fold Cross Validation test results

Table 3 shows the results gathered from the test that used 10-fold cross validation to validate the model. For validation of this model the dataset was split into 10 equally sized subsamples or folds. Of these 10 subsamples, one is retained as the validation data for testing of the model and the remaining 9 subsamples are used as training data. This process is then repeated 10 times so that each of the folds is exactly once as the validation data. These results are then averaged to provide a single estimation of the performance of the model. The overall accuracy as shown by this validation is shown to be 97.89%.

Correctly Classified Instances	97.8239%
Incorrectly Classified Instances	2.1761%
Average True Positive Rate	0.968
Average False Positive Rate	0.012
Average Precision	0.987
Average Recall	0.968
Average F-Measure	0.978

Table 4: Leave One Out CrossValidation test results

Table 4 shows the results gathered from the test that used Leave One Out cross validation to validate the model. LOOCV involves a similar process to 10-fold Cross Validation where, instead of splitting the data into equal sized folds, only one sample is retained as the validation data, with the rest being used as training data. This process is repeated as many times as there are samples in the dataset i.e. until every single sample has been used as the validation data once. The overall accuracy achieved using this validation method was found to be 97.82%.

Classified as	*VPN*	*Normal*
VPN	60	2
Normal	0	65

Table 5: Confusion Matrix for 80/20 split Validation test

Table 5 shows the confusion matrix for the test that used an 80/20 percentage split on the dataset. It shows 60 samples were correctly identified as VPN, 65 samples were correctly identified as non-VPN and 2 were incorrectly identified as non-VPN. Interesting is the lack of samples that were incorrectly identified as VPN.

Classified as	*VPN*	*Normal*
VPN	1872	59
Normal	24	1997

Table 6: Confusion Matrix for 10 fold Cross Validation test

Table 6 shows the confusion matrix for the test that used 10-fold cross validation. It shows 1872 samples were correctly identified as VPN, 1997 samples were correctly identified as non-VPN, 24 samples were incorrectly identified as VPN and 59 samples were incorrectly identified as non-VPN.

Classified as	*VPN*	*Normal*
VPN	1870	61
Normal	25	1996

Table 7: Confusion Matrix for Leave One Out Cross Validation test

Table 7 shows the confusion matrix for the test that used LOOCV for validating the model. It shows 1870 samples were correctly identified as VPN, 1996 samples were correctly identified as non-VPN, 25 samples were incorrectly identified as VPN and 61 samples were incorrectly identified as non-VPN.

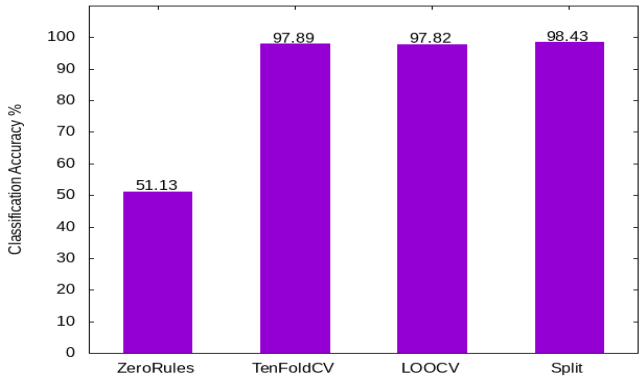

Figure 1: Graph comparing accuracies of different validation techniques against ZeroRules

The 80/20 split validation method was able to achieve an accuracy rate of 98.43%. Initially this would suggest that the 80/20 training and test split provides the best model, because the overall number of samples in the validation set is comparatively low, the results may not be reliable. This leaves the two types of cross validation to be compared to each other. 10-fold cross validation is one of the more popular forms of cross validation and is widely used. LOOCV is essentially cross validation where the number of folds that the data is sub-divided into is the same as the total number of samples in the dataset, in this case that would be 3952 folds. In the results the overall accuracies of the two methods are very close to one another. However, LOOCV has a much higher computation time when compared to 10-fold cross validation despite the individual fold computation time being lower. When 10-fold validation is used the model only has to be trained and tested once for each of the 10 folds, the model in this case must be trained and tested 3952 times when using LOOCV. Because the results of the two validation techniques are so close to one another, this means the benefits of LOOCV are possibly worthless. So, if we take the result of the 10-fold cross validation of 97.89% as the best indicator, it can be said that the neural network can accurately distinguish between an OpenVPN connection making use of Stunnel and normal non-VPN traffic. However, as noticed with the previous OpenVPN experiment, the confusion matrices for all of the validation methods used this time round show that the model is slightly too lenient, with a higher number of false negatives than false positives. Figure 1 shows the overall accuracies of each test to a test run without any rules applied. The ZeroRules method in Weka displays what the results would be in the event where everything is classified as one of the classes, in this case that was the normal class. Compared to the zero rules result, the neural network performs very well.

4. Conclusion

The aim was to investigate methods that would aid in the detection of VPN technologies that are being used to hide an attacker's identity. While VPNs have legitimate uses, such as connecting to a business network from a remote location, they are still abused by criminals who use them to commit crimes whilst remaining undetected and unidentified. Without a method to identify when a VPN is connecting to a web facing server, businesses could be vulnerable to having their network breached and having data stolen whilst being hindered in their ability to confidently say who stole it. This can be particularly detrimental to websites who deal with customer details and financial records. There are methods available for inspecting network traffic at the point of ingress and egress. An example of one of these methods is Deep Packet Inspection (DPI). It is closely related to another method called Shallow Packet Inspection (SPI), however SPI only has the ability to inspect the headers of network packets that are used to transport the packets to their destination. DPI goes a step further and inspects those headers and the actual content of the packet, which in the case of a HTTP packet could be a request for data from a website. A counter to DPI is the use of end-to-end encryption on the content of packets in order to hide those contents from prying eyes. This is done innocently enough with the goal being to stop potential man in the middle attacks from stealing sensitive data such as usernames and passwords or financial details as they are being transmitted. However, proxy and VPN technologies also have the ability to use encryption technologies with the use of IPSec and SSL/TLS. This increases the need for a method to identify these types of network traffic. Machine learning techniques are one way in which to accomplish this.

The experiments conducted to classify OpenVPN usage found that the Neural Network was able to correctly identify the VPN traffic with an overall accuracy of 93.71%. The further work done to classify Stunnel OpenVPN usage found that the Neural Network was able to correctly identify VPN traffic with an overall accuracy of 97.82% accuracy when using 10-fold cross validation. This final experiment also provided an observation of 3 different validation techniques and the different accuracy results obtained. Upon successful experiments conducted for the detection of Anonymising Proxy traffic, the focus was extended to include VPN traffic. The VPN technology OpenVPN was chosen as the focus for the experiments, which in turn found that the Neural Network was capable of classifying network traffic as either VPN traffic or as non-VPN traffic. This led to a further set of experiments which attempted to classify a form of OpenVPN traffic that made use of Stunnel to provide encryption. These found that a Neural Network trained on the Stunnel OpenVPN data could classify network traffic as either VPN traffic or non-VPN traffic. Again, the experiments were conducted in such as fashion as to eliminate bias where possible. This included keeping a portion of the captured dataset away from the training and tuning phases, so it could be used to simulate real world data that the model had never seen before.

References

1. Miller, S., Curran, K., Lunney, T. (2018) Multilayer Perceptron Neural Network for Detection of Encrypted VPN Network Traffic IEEE International Conference on Cyber Situational Awareness, Data Analytics and Assessment (Cyber SA 2018), 11-12 June 2018, Scotland, UK

2. Miller, S., Curran, K. and Lunney, T. (2018) 'Detection of Anonymising Proxies using Machine Learning', Special issue on Machine Learning for Cyber Security in Journal of Information Science (MDPI), ISSN 2078-2489, (Accepted) 2019

3. Geetha, S. and Phamila, A. V. (2016) Combating Security Breaches and Criminal Activity in the Digital Sphere. First. IGI Global. doi: 10.4018/978-1-5225-0193-0.

4. Peterson, A. (2014) 'The Sony Pictures hack, explained.', Washington Post, 18 December. Available at: https://www.washingtonpost.com/news/the-switch/wp/2014/12/18/the-sony-pictures-hack-explained/.

5. Pagliery, J. (2014) What caused Sony hack: What we know now, CNN. Available at: http://money.cnn.com/2014/12/24/technology/security/sony-hack-facts/ (Accessed: 6 December 2017).

6. Hunt, T. (2016) Observations and thoughts on the LinkedIn data breach, troyhunt.com. Available at: https://www.troyhunt.com/observations-and-thoughts-on-the-linkedin-data-breach/ (Accessed: 6 December 2017).

7. Chaum, D. L. (1981) 'Untraceable electronic mail, return addresses, and digital pseudonyms', Communications of the ACM. ACM, 24(2), pp. 84–90. doi: 10.1145/358549.358563.

8. Yang, M. (2015) 'De-anonymizing and countermeasures in anonymous communication networks', IEEE Communications Magazine, 53(4), pp. 60–66. doi: 10.1109/MCOM.2015.7081076.

9. Wood, D.. (1988) 'Virtual private networks', in 1988 International Conference on Private Switching Systems and Networks, New York, USA, pp. 132–136.

10. Zorn, G. (1999) 'Point-to-Point Tunneling Protocol (PPTP)', RFC 2637, pp. 1–57. Available at: https://tools.ietf.org/html/rfc2637 (Accessed: 12 January 2018).

11. Rawat, V. et al. (2001) Layer Two Tunneling Protocol {(L2TP)} over Frame Relay. doi: 10.17487/RFC3070.

12. Lawas, J. B. R., Vivero, A. C. and Sharma, A. (2016) 'Network performance evaluation of VPN protocols (SSTP and IKEv2)', in 2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN). IEEE, pp. 1–5. doi: 10.1109/WOCN.2016.7759880.

13. Thomas, K. et al. (2011) 'Design and evaluation of a real-time URL spam filtering service', in Proceedings - IEEE Symposium on Security and Privacy, pp. 447–462.

14. Cisco (2006) Access Control Lists: Overview and Guidelines. Available at: http://www.cisco.com/c/en/us/td/docs/ios/12_2/security/configuration/guide/fsecur_c/scfacls.html.

15. Dharmapurikar, S. et al. (2003) 'Deep packet inspection using parallel Bloom filters', IEEE Micro. IEEE Comput. Soc, pp. 52–61. doi: 10.1109/CONECT.2003.1231477.

16. Yu, F., Chen, Z., Diao, Y., Lakshman, T., Katz, R. (2006) 'Fast and memory-efficient regular expression matching for deep packet inspection', 2006 Symposium on Architecture For Networking And Communications Systems. New York, New York, USA: ACM Press, pp. 1–10. doi: 10.1145/1185347.1185360.

17. Sherry, J. et al. (2015) 'BlindBox', ACM SIGCOMM Computer Communication Review. ACM, 45(5), pp. 213–226. doi: 10.1145/2829988.2787502.

18. Dainotti, A., Pescape, A. and Claffy, K. (2012) 'Issues and future directions in traffic classification', IEEE Network, 26(1), pp. 35–40. doi: 10.1109/MNET.2012.6135854.

19. Miller, S., Curran, K. and Lunney, T. (2015) 'Traffic Classification for the Detection of Anonymous Web Proxy Routing', IJISR, 5(1), pp. 538–545. doi: 10.20533/ijisr.2042.4639.2015.0061.

20. Miller, S., Curran, K. and Lunney, T. (2016) 'Cloud-based machine learning for the detection of anonymous web proxies', in 2016 27th Irish Signals and Systems Conference, ISSC 2016. IEEE, pp. 1–6. doi: 10.1109/ISSC.2016.7528443.

21. García-Teodoro, P. et al. (2009) 'Anomaly-based network intrusion detection: Techniques, systems and challenges', Computers & Security, 28(1), pp. 18–28. doi: 10.1016/j.cose.2008.08.003.

22. Hawkes-Robinson, W. (2002) 'SANS Institute - Microsoft PPTP VPN Vulnerabilities - Exploits in Action'. SANS Institute. Available at: https://www.researchgate.net/publication/235927650_SANS_Institute_-_Microsoft_PPTP_VPN_Vulnerabilities_-_Exploits_in_Action (Accessed: 19 January 2018).

23. Schneier, B. and Mudge (1998) 'Cryptanalysis of Microsoft's point-to-point tunneling protocol (PPTP)', 5th ACM Conference on Computer and Communications Security, pp. 132–141. doi: 10.1145/288090.288119.

24. Farinacci, D. et al. (1994) 'Generic Routing Encapsulation over IPv4 networks', RFC1702, pp. 1–4. Available at: https://tools.ietf.org/html/rfc1702 (Accessed: 17 January 2018).

25. Simpson, W. (1996) 'PPP CHAP'. Network Working Group. Available at: https://tools.ietf.org/rfc/rfc1994.txt (Accessed: 19 January 2018).

26. Schmidt, J. (2012) A death blow for PPTP - The H Security: News and Features. Available at: http://www.h-online.com/security/features/A-death-blow-for-PPTP-1716768.html (Accessed: 19 January 2018).

27. Microsoft (2012) Microsoft Security Advisory 2743314 | Microsoft Docs, Microsoft Security Advisory. Available at: https://docs.microsoft.com/en-us/security-updates/SecurityAdvisories/2012/2743314 (Accessed: 12 January 2018).

28. Kazemi, K. and Fanian, A. (2015) 'Tunneling protocols identification using light packet inspection', in 2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC). IEEE, pp. 110–115. doi: 10.1109/ISCISC.2015.7387907.

29. Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>..

30. Feilner, M. (2006) Open VPN : building and operating virtual private networks. Packt. Publishing, ISBN: 190481185X, 2006.

31. Pohl, F. and Schotten, H. D. (2017) 'Secure and Scalable Remote Access Tunnels for the IIoT: An Assessment of openVPN and IPsec Performance', in, pp. 83–90. doi: 10.1007/978-3-319-67262-5_7.

32. Schneier, B. (1994) 'Description of a new variable-length key, 64-bit block cipher (Blowfish)', in. Springer, Berlin, Heidelberg, pp. 191–204. doi: 10.1007/3-540-58108-1_24.

33. Scarfone, K. and Mell, P. (2007) 'Guide to intrusion detection and prevention systems (idps)', NIST special publication, 800(2007), p. 94.

34. Lin, W.-C., Ke, S.-W. and Tsai, C.-F. (2015) 'CANN: An intrusion detection system based on combining cluster centers and nearest neighbors', Knowledge-Based Systems, 78, pp. 13–21. doi: 10.1016/j.knosys.2015.01.009.

35. Xiang, C., Yong, P. C. and Meng, L. S. (2008) 'Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees', Pattern Recognition Letters, 29(7), pp. 918–924. doi: 10.1016/j.patrec.2008.01.008.

36. Khan, L., Awad, M. and Thuraisingham, B. (2006) 'A new intrusion detection system using support vector machines and hierarchical clustering', The VLDB Journal, 16(4), pp. 507–521. doi: 10.1007/s00778-006-0002-5.

37. Özyer, T., Alhajj, R. and Barker, K. (2007) 'Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening', Journal of Network and Computer Applications, 30(1), pp. 99–113. doi: 10.1016/j.jnca.2005.06.002.

38. Ghosh, A. K., Schwartzbard, A. and Schatz, M. (1999) 'Learning Program Behavior Profiles for Intrusion Detection.', in Workshop on Intrusion Detection and Network Monitoring.

39. Samuel, A. L. (1959) 'Some Studies in Machine Learning Using the Game of Checkers', IBM Journal of Research and Development, 3(3), pp. 210–229. doi: 10.1147/rd.33.0210.

40. Khriplovich, I. B. and Pomeransky, A. A. (1998) 'Equations of Motion of Spinning Relativistic Particle in Electromagnetic and Gravitational Fields', EUA: Prentice Hall, 178, p. 640. doi: 10.1080/01422419908228843.

41. Russel, S. J. and Norvig, P. (2010) Artificial intelligence: a modern approach. Third Edit, EUA: Prentice Hall. Third Edit. doi: 10.1017/S0269888900007724.

42. Cisco (2018) Encrypted Traffic Analytics. Available at: https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/enterprise-network-security/nb-09-encrytd-traf-anlytcs-wp-cte-en.pdf (Accessed: 12 January 2018).

43. Liu, C., White, R. W. and Dumais, S. (2010) 'Understanding web browsing behaviors through Weibull analysis of dwell time', in Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval - SIGIR '10. New York, New York, USA: ACM Press, p. 379. doi: 10.1145/1835449.1835513.

44. Arndt, D. (2011) NetMate-flowcalc. Available at: https://dan.arndt.ca/projects/netmate-flowcalc/ (Accessed: 4 October 2018).

45. Stibler, S., Brownlee, N. and Ruth, G. (1999) 'RTFM: New Attributes for Traffic Flow Measurement', pp. 1–18. doi: 10.17487/RFC2724.

46. Frank, E., Hall, M. A. and Witten, I. H. (2016) 'The WEKA Workbench Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques" Morgan Kaufmann, Fourth Edition, 2016', Morgan Kaufmann, Fourth Edition. Available at: https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf (Accessed: 8 November 2017).

47. Deri, L., Martinelli, M., Cardigliano, A. (2014) 'nDPI: Open-source high-speed deep packet inspection', in 2014 International Wireless Communications and Mobile Computing Conference (IWCMC). IEEE, pp. 617–622. doi: 10.1109/IWCMC.2014.6906427.