Cache-Based Characterization: a Low-Infrastructure, Distributed Alternative to Network-Based Traffic and Application Characterization

Anatoly Shusterman*, Chen Finkelstein, Ofir Gruner, Yarin Shani, Yossi Oren

Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract

It is important for network operators to carry out traffic and application characterization to gain insights into the activity of their networks. Several studies proposed methods that extract features from network traffic to characterize it, or to classify the application that produced it, based on a "man in the middle" network interception point that can analyze the entire network traffic of an organization. This network topology, however, is increasingly becoming irrelevant, due to mobile and remote traffic joining the corporate network by passing through VPN channels or relay networks.

In this work we propose an edge-oriented lightweight traffic characterization method, based on measuring contention on the last-level CPU cache. In contrast to previous traffic characterization methods, which track network traffic from a central location, our method performs measurements directly on user machines, using an unprivileged JavaScript-based webpage. Our evaluation shows that the accuracy of our cache-based method is equivalent to that of network-based methods, both over VPN and over non-VPN networks.

Keywords: Communication Network Protocols, Network Security and Privacy, Side Channel Analysis, Cyber-Physical Systems

1. Introduction

The growth of large local corporate computing networks, and the increase in the number of workers accessing these networks, have been accompanied by increasingly complex LAN maintenance and network security requirements. The presence of anomalous network activity on an organizational network can stem from computer malfunction, the malicious activity of an infected machine, an intentional or unintentional internal data breach made possible using social engineering, and more. A delay in detecting such abnormal network behavior might interfere with an organization's business activity by preventing its network from supporting the guaranteed quality of service (QoS), or allowing its secrets and assets to be leaked to their opponents. Thus, it is important for network operators to carry out traffic and application characterization to gain insights into the activity of their networks.

The traffic characterization task has been traditionally performed in a centralized location by a device capable of analyzing the network activity of the entire organization. This "man-in-the-middle" (MITM) approach has increasingly been challenged by traffic joining the corporate network by passing through VPN channels or relay networks. Such traffic might be encrypted, obfuscating its ports and IP addresses, or have modified statistical attributes, such as packet size, directions, and timing.

The COVID-19 pandemic and the resulting isolation policies have further complicated this centralized approach. As companies encourage their employees to work from home, with some of them even leveraging the pandemic to close most of their offices and shifting towards working from home permanently, there is no longer a simple way to intercept and monitor all organizational network traffic. Under these circumstances, MITM traffic sniffers are not feasible, as they would have to be installed at the home of every employee. Software-

*Corresponding author.
E-mail addresses: shustera@post.bgu.ac.il (A. Shusterman), chenfi@post.bgu.ac.il (C. Finkelstein), ofirgru@post.bgu.ac.il (O. Gruner), yarinab@post.bgu.ac.il (Y. Shani) yos@bgu.ac.il (Y. Oren)

based data loss prevention (DLP) tools, which are highly-privileged local agents which monitor the activity of corporate computers, are also less practical on computers and mobile devices which are not under the administrative control of the organization. Furthermore, a new risk to organizational networks has appeared, in which users connect to their corporate networks using VPN, and then use these corporate networks as relay points for file downloads, media streaming, and other unsanctioned activities. There is a need for a network analysis approach which is focused on the end user and not on a central approach, and which is light-weight enough to be run on non-corporate systems with minimal installation requirements.

A field of studies related to traffic characterization is the website fingerprinting field, where researchers aim to identify the specific website surfed to by end users [32, 25, 27]. These studies exploit the side-channel patterns in network flows and the timing of web page rendering, which is influenced by the structure of web pages (e.g., images, style, scripts). In these studies, researchers used machine learning techniques and statistical analysis to map these unique patterns to a specific web page and thus violate users' privacy.

In addition to network-based website fingerprinting, there has been a series of works on performing website fingerprinting based on local side channels, as measured by an attacker-controlled website running on the target machine [23, 2, 10, 35, 19]. In particular, cache-based website fingerprinting attacks were proposed by Shusterman et al. [29]. Instead of analyzing websites' network traffic, their work measured the CPU's last level cache contention over time during web page rendering. This measurement was performed without installing any software on the target machine. Instead, unprivileged JavaScript code, running on the target machine's standard web browser, was used to measure cache activity. Their method demonstrated accuracies similar to those of network-based cache fingerprinting, and furthermore demonstrated robustness against network website fingerprinting countermeasures.

Since network-based website fingerprinting is complemented very well by cache-based website fingerprinting, we were motivated to apply the same approach to network-based traffic and application classification. In particular, this study is aimed at answering the following research question: *Can the accuracy of network-based traffic and application classification be achieved with light weight cache-based methods?*

We answer the question in the affirmative, by describing a traffic and application classification system built on cache-based methods, which has comparable performance to network-based methods without the limitations of the MITM architecture.

The contributions of our study are as follows:

- We propose a method for remote characterization of network traffic and applications using the last-level cache side channel. We measure the accuracy of our system using standard metrics and show that it can perform traffic characterization with an accuracy of around 90% and application characterization with an accuracy of around 83%. This performance is similar to that of network-based methods, while requiring neither an external interception point, nor any privileged software installed on the remote machine.

- We analyze the minimal length of cache contention traces for the purpose of traffic characterization, and show that a measurement period of 10 seconds is enough to provide reasonable accuracy.

- We show our proposed method can perform traffic analysis on cross-traffic data, with and without a VPN, when training is performed without a VPN, and vice versa. In this cross-traffic setting, the cache based approach has a distinct advantage, providing a detection rate that is almost 5 times higher than the network-based approach in the traffic characterization setting, and almost 8 times higher in the application characterization setting.

- We publish a unique open-source dataset [30] containing more than 600GB of cache contention traces for various application profiles, correlated with network traffic data traces for the same activities, as well as code reproducing the results. This dataset can be used by other researchers to evaluate and validate additional methods for cache- and network- based traffic characterization.

Our work is the first time cache-based approaches have been harnessed to the task of application characterization. It improves on state of the art in

network-based application characterization by providing an approach that provides comparable accuracy to network-based approaches, without requiring the financial, legal, and technical challenge of forcing all traffic to flow through a MITM interception point.

2. Related Work

2.1. Network Traffic and Applications

There are two types of network traffic analysis. The first is **traffic characterization**, which identifies the general activity of the machine's user, whether that be browsing the Internet, sending an email, chatting, streaming videos, transferring files, communicating over VOIP, sending P2P data, etc. The second is **application classification**, which identifies a specific application, such as browsers like Chrome or Firefox browser, or a chat application (e.g., Google Hangouts, ICQ, or Skype), etc.

Two main traffic characterization methods have been proposed for network-based traffic analysis [34]: payload-based and feature-based methods.

The **payload-based method** requires knowledge of the protocols and the packet or flow structure. Prior research used characteristics like port numbers [8], and the matching of protocol structures and magic words that could be parsed by regular expressions [18]. Such methods could therefore mainly be applied on non-encrypted traffic [28].

The **feature-based method** uses the communication patterns and packet statistics rather than the packet structure. Moore et al. [21] described 249 feature discriminators; the main discriminators were the packet size, packet direction, and socket tuple, and statistical information derived from them, such as the average, mean, and standard deviation. As these features are derived from network traffic patterns, they can be obfuscated using tools like VPNs and relay networks (e.g., Tor network [6]) or traffic fingerprinting countermeasures [9, 36, 4] which are robust to regular packet encryption.

Machine learning (ML) algorithms are applied to make use of these features for traffic characterization, as the patterns and connections between the features might be complex or difficult to define manually. Research performed in this field has used different combinations of features and ML algorithms to classify network characteristics.

Draper-Gil et al. [7] proposed a method in which the statistical features of traffic flows are extracted, and then C4.5 and KNN classifiers are applied to characterize types of traffic. The authors used a two-stage classifier to address the challenge of detecting two different types of patterns in the VPN and non-VPN traffic; the first classifier determines whether the traffic has VPN characteristics, and the second one determines the type of the traffic.

Adopting their time based features, Caicedo-Muñoz et al. [5] applied the traffic characterization method to the QoS domain, integrating their domain knowledge of the traffic marking process. They suggested using a **per-hop behavior** (PHB) labeling system that is compatible with the QoS traffic marking process, and evaluated machine learning classifiers using this labeling system.

In contrast to the previously mentioned study [7], Yamansavascilar et al. [37] examined the application identification problem. They proposed performing feature selection using the χ^2 and CFS [13] methods before the classification.

While previous papers used statistical features on the network flows, Lotfollahi et al. [17] proposed using the raw packet data bytes without the IP address as the input to a convolutional neural network (CNN) classifier. Their model classified every packet to a specific application or web activity and outperformed the previously suggested methods [7, 37].

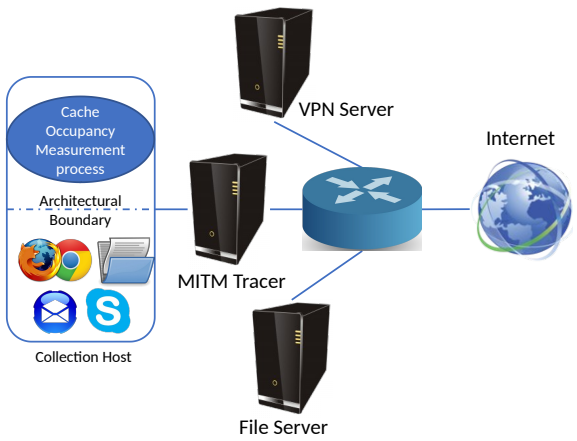


Figure 1: Experimental setup – The collection host run an application process along the cache contention measurement process. The MITM network tracer collects the network-associated data with the cache contention traces. The file server and the VPN server are connected to the default gateway.

2.2. Prime+Probe Last Level Cache

The Prime+Probe attack is a cache-based side-channel attack originally designed by Osvik et al. [24]. In [16] the authors demonstrated how a type of Prime+Probe attack can be used to extract data from the LLC. The Prime+Probe attack requires the attacker process to run on the same hardware as the victim process. This attack has two phases; in the first phase, the attacker process **primes** the cache by allocating a large buffer (called an eviction set) and accessing every location in this buffer, effectively overwriting all of the data in the cache. In the second phase, the attacker process **probes** the cache by periodically accessing every location of the eviction set and measuring the access time, while the victim process runs in the background. Since the victim process must evict the attacker's data from the cache in order to perform its own calculations, longer access times are indicative of victim process activity in the specific cache set.

The authors used this information leakage to distinguish between square and multiplication operations in the RSA algorithm, allowing them to extract the cryptographic key. They also performed a Prime+Probe attack to create a covert channel between two processes on the same machine.

The covert channel presented in [16] requires that the sender and receiver agree to use the same cache set, which makes the communication initiation setup very complicated. A different approach for LLC-based covert channel, presented by Maurice et al. [20], measured the entire LLC contention rather than that of a single cache set. While this method is slower, it makes the transmission of information possible in noisy settings such as the cross-VM channel.

The work of Oren et al. [23] showed the feasibility of creating eviction sets using JavaScript code, despite the virtualization layer provided by the browser's sandbox. Therefore they were able to perform the Prime+Probe attack over the web. Their methods performance achieved spatial resolution similar to the native attack presented in [16].

As a result of the Spectre and Meltdown [11, 14] attacks, browser vendors reduced the resolution of the JavaScript timer by three orders of magnitude. Therefore, the precise side-channel attack performed in [23] is no longer feasible.

A recent study of Shusterman et al. [29] used cache occupancy channel to perform a website fingerprinting attack, that identifies which webpage the victim visits. They were able to measure the entire LLC contention using a Prime+Probe attack implemented in JavaScript. They showed that when this cache measurement is done while at the same time the target machine is rendering a webpage, the resulting cache trace can be mapped to the webpage with high accuracy. This result was recently extended to work using CSS instead of JavaScript, making it feasible even if JavaScript is blocked on the target machine [31].

2.3. Towards Cache Traffic Characterization

Although traditional network-based traffic characterization methods perform well, they have very limited scalability in large organizations, where analyzing the entire organization's network traffic is difficult to impossible. Capturing large-scale organizational network traces requires a powerful server placed in series to the organization gateway, with the ability to monitor, store and process a vast amount of traces. Such an implementation might act as a bottleneck, reducing the performance of the network. Another limitation of this method is the ease of manipulating the network traffic. As workers might want to use applications prohibited by the company, they can manipulate the network traffic characteristics using padding and noise injection methods. One such method is BuFLO [4, 3], whose authors noted that there is a tradeoff between network performance and security: while padding packets and transmitting dummy packets on the local network produces random network patterns that reduce website fingerprinting classifiers' performance, they also produce overhead that reduces the network capabilities. In [29], the authors compared the performance of a network-based classifier to an LLC-based classifier when applying this countermeasure. The results showed that the network-based classifier's performance worsens, whereas the cache-based classifier is only slightly affected by this countermeasure. In addition to the above, due to the isolation policies followed the COVID-19 pandemic, there has been a global shift from working at a company's offices to working from home, and according to Bloom [1], this change is here to stay. In these circumstances, MITM traffic sniffers are not feasible, as they would have to be installed at the home of every employee. In contrast, cache contention measurement can be easily achieved remotely without intrusive software running on the user's computer or home router.

3. Methods

3.1. Security Model

In our work, we assume that the defender is an organization providing Internet access to its users. These users can be located on the premises of the organization, or they may be connected remotely to the organization through a VPN. The adversary in our case is a user interested in running prohibited applications, while taking advantage of the organization's network connectivity. We assume that the adversarial user can install and run arbitrary applications on the remote computer. The user can therefore also generate arbitrary network traffic on the defender's network. Specifically, the adversarial user can attempt to bypass the organizational traffic inspection firewall, by masquerading the prohibited network traffic as legitimate protocols. Since the user's computer is only partially under the organization's control, it might be difficult to install standard monitoring and data-loss prevention (DLP) tools on this computer, which require administrative privileges to operate. The objective of the defender in this security model is to identify the application running on the user's computer, under these challenging software installation limitations and unreliable network traffic patterns, Our security model is inspired by several recent cases in which computers running inside large organizations were recruited via malware into running attacks, such as distributed denial of service (DDoS) or ransomware. Our method can complement standard malware detection countermeasures, by offering a lightweight way of detecting nonstandard applications running on computers inside an organizational network.

3.2. Approach

To perform cache-based application classification, we measure the cache contention over time, as suggested by Shusterman et al. [29]. We run the cache contention measurement code in parallel with the tracked application, on the endpoint client. When the cache contention trace collection is complete, we send it to a remote data collection server, where standard machine learning tools are used to classify the trace to one of a set of known applications.

As we are the first to use cache contention side channel data in order to perform traffic characterization, We adopt the methodology of [29], collecting network traces with a MITM tracer machine as we collected the cache contention data. In this method, we can compare traffic characterization methods from literature applied on network data to the results obtained using our method on cache contention data. We stress that we collect the MITM network traces only for comparison with related work, and do not use them directly in our analysis.

As suggested in [7], we collected the data in two modes: with and without a VPN. The VPN measurements were conducted when both , the collection host and the the file server are connected to the network behind the VPN server on the same local network.

3.3. Machine Learning Evaluation.

In this work, we compared cache based traffic analysis to the network based traffic analysis methods proposed in the literature. We used two types of models in our evaluation: A classical machine learning model, and a deep learning model.

For the **classical model** experiments, we used the model that outperformed the others on traffic characterization task. We extracted the features mentioned in Section 3.7 from cache contention traces sampled at a rate of 500 Hz. The collection time of each cache trace was 10 seconds, for a total of 5000 data points per trace. The models we compared were Gradient Boosting, Random Forest and SVM classifiers, as implemented in the scikit-learn 0.21.3 repository for Python 3.7 [26]. In terms of feature hyper-parameters, for the Gradient Boosting classifier we used 500 estimators and a learning rate of 0.1, for the Random Forest Classifier we used 200 estimators, and for the SVM classifier we used the RBF kernel. The results reported in table 1 show that the best performing classical classifier was Random Forest. We thus decided to use this classifier for the remainder of the paper.

Table 1: Classical Cache-Based Classifiers Comparison

Classifier	**F1-Score**
Gradient-Boosting	83±1.5
Random Forest	86.9±0.9
SVM	64.2±1.2

We now describe our **deep learning** mode experiments. Our cache contention dataset is high-dimensional, and requires many hours of human work for collection. This motivated us to use a

neural network with a shallow structure and a small amount of weights, to prevent having the network overfit the relatively small training dataset. The network we evaluated is illustrated in Figure 2. Its architecture was proposed by Shusterman et al. [29], and we applied some fine tuning to make it better fit our data. It consists of two convolutional blocks of a convolutional layer with the ReLU activation function and a max pooling layer, an LSTM block which consists of an LSTM layer and a Dropout layer, and a classification block which consists of dense layer with softmax activation function. The convolutional blocks extract local features out of the input vector and reduce the dimensionality. the LSTM block extracts temporal features out of the local features. Finally, the softmax layer outputs the probability of a trace belong to each class. This model was implemented using the TensorFlow 2.1 repository in Python 3.7. Full details of the hyper-parameter tuning of the LSTM network are provided in Appendix .3.

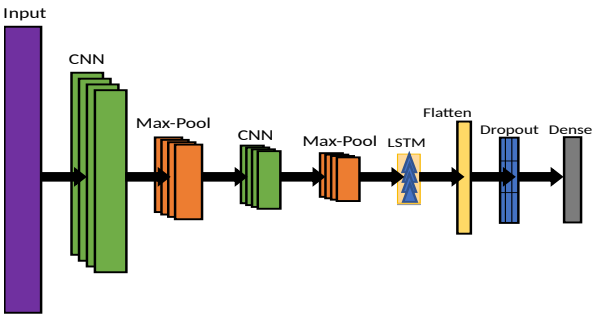


Figure 2: LSTM classifier architecture for cache contention data

To evaluate the models, we used the F1-score and reported results of stratified data with 10-fold cross-validation to reduce bias.

3.4. Experiments

To evaluate the effectiveness of our technique in traffic characterization and application classification we conducted a series of four experiments. Our first experiment is used to calibrate our trace length and extract features. Next, we evaluate our techniques and compare them to the state of the art methods from the literature.

Trace length optimization. To understand the tradeoff between the length of the cache contention trace fed to the classifier and its performance, where a smaller input would provide a shorter trace collection time and a larger input might improve the accuracy. The purpose of this experiment was to determine the minimum trace length that provides adequate classifier performance for the traffic characterization task, and the trace length that provides an optimal balance with the classifier's performance.

Traffic Characterization Evaluation. We assessed the performance of the Random-Forest and LSTM classifiers trained on the cache contention data to distinguish between eight categories of traffic: file transfer, VOIP-video, VOIP-audio, chat, email, P2P, streaming, and web browsing. We compared them to the network traces classifiers proposed in previous works [7, 17] on two types of traffic: with and without a VPN.

Application Classification Evaluation. We compared the performance of classifiers trained on cache contention datasets in the task of application classification to the classifiers proposed in previous works [7, 17] . For this task, we labeled our data according the 26 applications that created it (described in Appendix .2).

Cross-Traffic Classification. We examined the ability of a classifier trained on VPN data to classify non-VPN traffic data and vice versa. This experiment was also aimed at demonstrating the robustness of a classifier trained on non-VPN traffic when coping with previously unseen VPN network traffic. This case was measured using F1 score when considering the model's most probable result, and when the result appears in the top-3 most probable classes. The top-3 evaluation method assumes that there is a prior distribution that might be helpful in choosing the most probable class out of three.

3.5. Hardware Configurations

The data collection setup for cache contention and network traces presented in Figure 1 consists of the following four machines, each of which runs the Ubuntu 18.4 operating system:

Collection host – A machine with an Intel Core i7-6600U CPU at 2.60GHz with 16GB RAM. This machine run cache contention data measurements script on Firefox browser (version 77.0).

MITM tracer – A machine with an Intel Core i5-4570 CPU at 3.20GHz with 8GB RAM. This machine has two network cards that are bridged; one of the ports is connected to the endpoint user, and

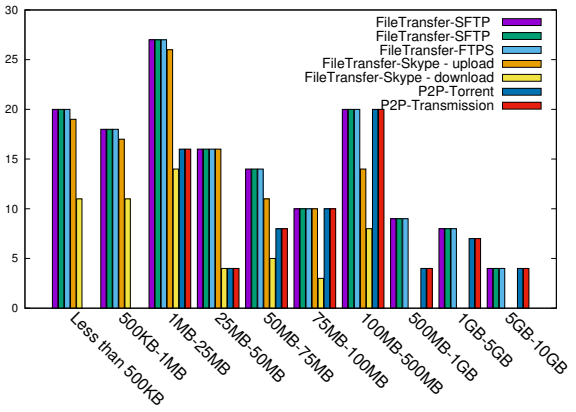


Figure 3: File transfer data distribution — size of files transferred when collecting data traces in non-VPN network.

the other is connected to the default gateway. The network traces were collected with tcpdump application (version 4.9.3).

File server – A machine with an Intel Core i7-6600U CPU at 2.60GHz with 16GB RAM. It contains the files transmitted to the user machine.

VPN server – A machine with an Intel Xeon E5-2650 CPU at 2.00GHz with 40GB RAM. This machine runs the The OpenVPN server (version 2.4.4) for the VPN measurements.

3.6. Data Collection

We collected the cache contention and network traces of different endpoint activities: downloading and uploading data, making VoIP audio and video calls, chatting , emailing and transferring P2P data. The network traffic and cache contention side-channel data is collected when performing each of these tasks separately . Some of the activities were performed automatically, while imitating the real world, such as web browsing and streaming, and the other activities were performed manually. In total, each activity has a total of two hours of data collected with a VPN, and two hours of data collected without a VPN; among them, the time of running applications performing the same activity is equal. The application details are as follows:

File transfer. — We used three different file transfer methods to collect data: FTPS, SFTP, and Skype. For each application we captured separately the download and the upload data. To support coverage and diversity in the data collection, different types of files were transferred: jpeg, mp4, mp3, zip, pdf, exe, and bin, and the size of the files varied, as seen in the distribution presented in Figure 3. The collection setup had the client and the file server machines connected to the same LAN.

For the FTPS method, we used the ProFTPD server (version 1.3.5e) [22] and a FileZilla client application [12] (version 3.28.0). For the SFTP protocol, we used OpenSSH (version 7.6p1) both for the client and the server machines. Another method used was the Skype for Linux [33] application (version 8.57.0.116). In contrast to the other methods, Skype can only send files that are a maximum of 300MB each time, so this reduced the transferred file size variance.

VoIP-Audio — The following applications were used to collect VoIP-audio data: Skype version 8.60, Facebook on the Firefox 77.0 browser, and Skype for Linux version 8.60. This data was collected during an audio-only human conversation, collecting cache contention data and network traffic only from one side of the conversation.

VoIP-Video — Data was collected during VoIP video conference calls on three applications: Zoom for Linux (version 5.0.4), Skype for Linux (version 8.60), and Google Meet on the Chrome browser (version 80.0).

Web Browsing — Browsing data was collected using the Chrome 80.0 and Firefox 77.0 browsers. To simulate real-world web browsing, we used the Alexa Top 100 most popular websites, and ran a Python crawler that randomly accessed web pages while clicking some links on the page. To mimic human behavior, we simulated the dwell time, which refers to the time a user spends viewing a document after clicking on it using a search engine link. In this simulation, we used the Weibull distribution [15] with a shape parameter of three multiplied by 65 seconds; this takes an average of one minute per web page. We clarify that we did not collect data from browsing over Tor network, or with the Tor browser.

Chat — Chat data was collected when using five different applications: ICQ (version 10.0.8172) and Skype (version 8.60) native applications, and WhatsApp, Facebook, and Google Hangouts chats (the data from the latter three was collected while the apps were running on the Firefox 77.0 browser).

Email — The email data was collected using Thunderbird (version 68.8), from three protocols: SMTP, IMAP, and POP3. When collecting SMTP data, we tracked the keystrokes and email transmission, and for POP3 and IMAP, incoming mail protocols did not include the keystrokes or trans-

mission.

Peer-to-Peer – P2P data was collected using two different P2P applications - qBittorrent (version v4.2.5) and Transmission (version 2.92). 3 presents the downloaded file size distribution.

Streaming — Streaming data was collected using two different streaming platforms: YouTube and Vimeo, both of which were running on the Firefox 77.0 browser. We created a playlist in YouTube and a watch later playlist in Vimeo, both of which contained different sized videos, and played the playlists when collecting the cache contention data.

3.7. Feature Extraction

Before feeding the cache contention data to the classifier, we transformed the raw data to a feature vector. Following Yamansavascilar et al. [37], these features were validated as contributing to the model performance using a the χ^2feature selection test. These are the features we selected:

Cache contention histogram bins – Figure 4 presents the cache contention distribution when performing various network activities. The cache contention is measured as the time it takes to prime the last level cache. The cache measurement script runs on Firefox 77.0 and used its 2-msec performance.now timer. As seen in the figure, different web activities have different cache contention distributions.

Cache contention Δs histogram bins – thirteen features were created by using calculating the delta between every two adjacent cache contention measurements, then extracting the histogram bins from these difference vectors.

Temporal features – statistical features were extracted from the raw data, e.g., average, standard deviation, mean absolute deviation, skewness, kurtosis, and zero-crossing rate.

Spectral features – statistical features were extracted over spectral representation of the raw data. For the transformation to the spectral representation, we used the absolute values of the FFT (fast Fourier transform) function on the raw data. Then we extracted statistical features, e.g., spectral centroid, spectral entropy, spectral spread, spectral skewness, spectral kurtosis, spectral flatness, and spectral irregularity. These statistical functions are described in Appendix .4.

3.8. Reproduction and Comparison

Although our proposed data collection method relies on the cache contention side channel rather than network traces, we compare the performance of our machine learning models to the the network traffic characterization and application classification methods proposed in recent studies [7, 17]. Before applying the preprocessing stages of previous works, we filtered the irrelevant traffic of several known ports (see a list of these ports in Appendix .1.) from the network traces. When evaluating the methods proposed by Draper-Gil et al. [7], we used their Naïve Bayes model with the features mentioned in their paper. When evaluating the methods proposed by Lotfollahi et al. [17], we used their CNN model. As in [17], we masked the IP addresses to avoid bias toward a specific address.

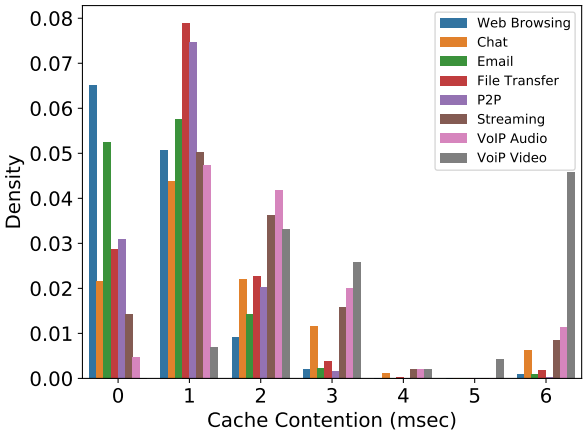


Figure 4: Distribution of cache contention over 2 hours of each protocol without VPN.

4. Results

4.1. Performance vs Time Optimization

In our analysis of the traffic characterization results, we first evaluate the influence of the trace length on the model performance. In this experiment, we evaluate several models with different cache contention feature vector lengths, from one second to 1.5 minutes. Figure 5 shows the tradeoff between the cache contention trace length and the model performance.

The Y-axis indicates the average F1 score of the random forest model with VPN data based on 10-fold cross-validation, and the X-axis indicates the cache contention trace length. Based on this experiment, we can see that the classifier's best performance is obtained when the cache contention trace size is 20 seconds, with an F1 score of 87.1%. We

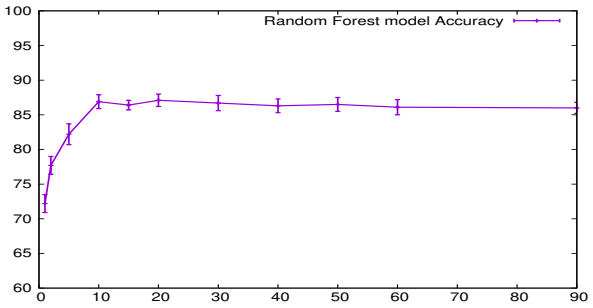


Figure 5: The tradeoff between cache contention trace length and the random forest model performance.

can also see that a trace length of one second performs adequately and achieves a score of 70%. The trace length we use for the rest of experiments is 10 seconds, a length which achieves an F1 score that is just 0.2% lower than that of a 20 second trace, but its collection time is two times shorter.

4.2. Evaluation of Traffic Characterization Models

Here we compare the performance of our classifiers trained on cache contention traces to the classifiers trained on network traces when performing the traffic characterization task. For classification of cache traces, we use the Cache-Random-Forest and Cache-LSTM models as described in Section 3.3. For the network traces classification, we use classifiers from the literature: Network-Naïve-Bayes, based on Draper-Gil et al. [7], and Network-CNN, based on Lotfollahi et al. [17]. The results in Table 2 show that both neural network models (Cache-LSTM and Network-CNN) perform better than their classical machine learning counterparts (Network-Naïve-Bayes and Cache-Random-Forest). Also we can see that the Cache-LSTM classifier has similar results on VPN and non-VPN data, which means that it is not affected by the difference in the traffic type.

When looking at VPN vs non-VPN traffic in results in table 2, we can see that, with the exception of the Network-Naïve-Bayes classifier, traces are easier to classify when data collected on VPN network using the other methods. These results show that despite the presumed security of the VPN network, it provides cleaner traces in both of the side channels, allowing better classification of a web activity.

Further, we investigate the weaknesses of the deep learning models that performed best for every dataset: the Cache-LSTM and Network-CNN models. The confusion matrices in Figures 6,7 show the misclassification of these models. In this figure, the vertical axis is the true label, and the horizontal axis is the predicted label. The color and the percentage displayed a the matrix present the probability of a class in the vertical axis to be classified as a class on the horizontal axis.

In the confusion matrices we can detect 3 types of behaviors – Classes with high percentage of classification as the true class, classes with low percentage of misclassification distributed equally over the other labels, indicating a class which some of its labels were hard to identify, and finally a case in which several classes have similar traces, and then these classes have high percent of misclassifications between them and low percent of misclassifications with other labels. In the Cache-LSTM confusion-matrix with non-VPN data 6a we can see a confusion between Emailing and Browsing traces. In the confusion matrix of Network-CNN with non-VPN data 6b we can see two types of behaviors: a confusion of between Emailing and all other labels outside of VoIP-Audio and VoIP-Video, and the File-Transfer activity which has a small percent of misclassification with Browsing, Chatting and Streaming.

When analyzing the confusion matrices of VPN data(see figures 7), the Cache-LSTM classifier in Figure 7a misclassifies FTP and P2P labels as Browsing, and the Browsing label might be misclassified with P2P traces. The Network-CNN model classifications on VPN traces exhibit a small amount of misclassifications of Chatting traces with the Emailing label.

4.3. Evaluation of Application Classification

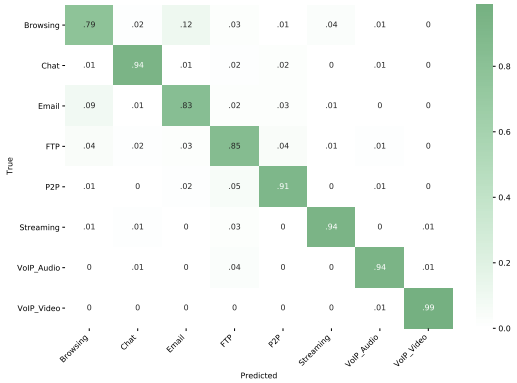
Here we compare the performance of cache and network classifiers in application classification task. The results of the experiment show that the Cache-LSTM outperforms the other classifiers in non-VPN data whereas Network-CNN is better in classification of the VPN data.

Further analysis of Table 2) shows that the non-VPN dataset is more challenging for classifiers than the VPN dataset, as in the traffic characterization task. The Cache-LSTM model achieves the best F1 score for the non-VPN dataset; this is followed by the Network-CNN model. We found that the classical models obtain poor results in this task, but the Cache-Random-Forest model performs better than Network-Naïve-Bayes. For the VPN dataset, the

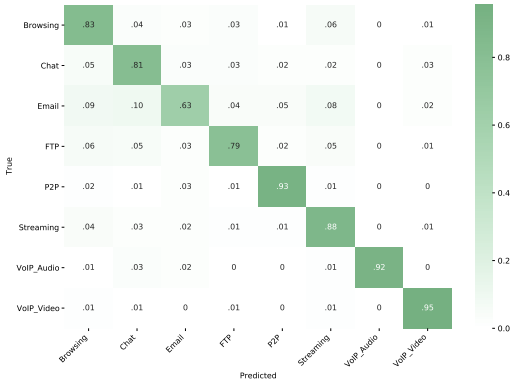
Table 2: Model Performance comparison

		Classical Models		Deep Learning Models	
Task		**Cache-Random-Forest**	**Network-Naïve-Bayes** [7]	**Cache-LSTM**	**Network-CNN** [17]
Traffic	Non-VPN	62.2±4.8	72.9±1.2	**89.6±1.3**	84.2±0.2
Characterization	VPN	86.9±0.9	57.1±1.5	90.7±0.1	**97.9±0.1**
Application	Non-VPN	55.4±2.8	24.9±0.2	**83.2±3.3**	77.1±4.5
Classification	VPN	79.2±1.4	31.6±0.3	84.7±2.0	**97.3±1.7**

Figure 6: Traffic Characterization on No VPN data Confusion Matrices.



(a) Traffic Characterization on Cache Contention data confusion on LSTM classifier.



(b) Traffic Characterization on Network data confusion on CNN classifier [17].

Network-CNN model achieves the best result, and this is followed by the Cache-LSTM model.

The confusion matrices of deep learning models with non-VPN data are presented in Figure 9. In Figure 8a , which presents the confusion matrix of the Cache-LSTM model, it can be seen that the applications for which there is the most confusion are from within traffic characterization categories, such as Chrome and Firefox browsers, which produce similar cache contention characteristics, as well as qBittorrent and Transmission, which are both used for P2P transmissions.

Another interesting insight is that although all of the chat and email traces of cache contention data were captured, along with the effect of the keystrokes on the keyboard, the confusion between them is not high. When examining the confusion matrix of the Network-CNN classifier in Figure 8b, we can see that it struggles with classifying the chat-WhatsApp and email-smtp categories, achieving less than 15% accuracy, and with classifying chat-skype, achieving 41% accuracy for that category.

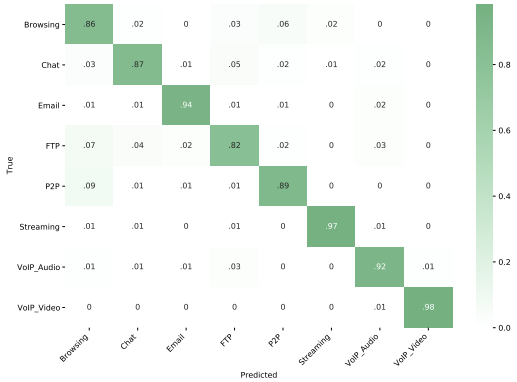
When examining both confusion matrices of the VPN and non-VPN data (see Figures 8a,9a), the LSTM model trained on cache contention data experiences confusion within each activity, such as browsing, chatting, and file transferring. The confusion matrix of the Network-CNN classifier is presented in Figure 9b; The confusion in this model is mainly between the P2P applications: qBittorrent and Transmission.

A detailed analysis of per label classification in table 3 shows that both Network-CNN and Cache-LSTM classifiers do not have significant difference between precision and recall inside any label. This result confirms that there is no classification bias toward any label.

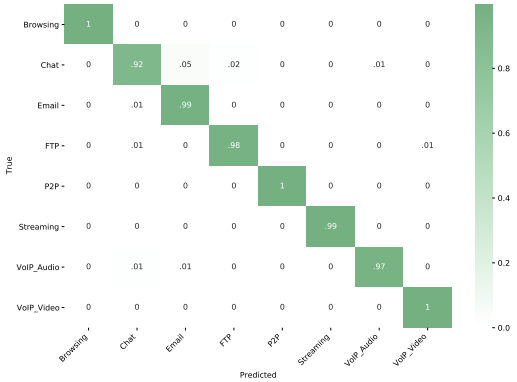
4.4. Cross Traffic Classification

In the experiment on the cross-traffic classifier, we examine the ability of a classifier trained on non-VPN dataset traces to classify VPN dataset traces and vice versa. We discuss the results for both cases together, as they achieved similar performance. In this experiment, only the deep learning models that performed better than the classical models are evaluated. The advantage of the Cache-LSTM classifier over the Network-CNN classifier can be seen in Table 4. On the traffic characterization task,

Figure 7: Traffic Characterization on VPN data Confusion Matrices

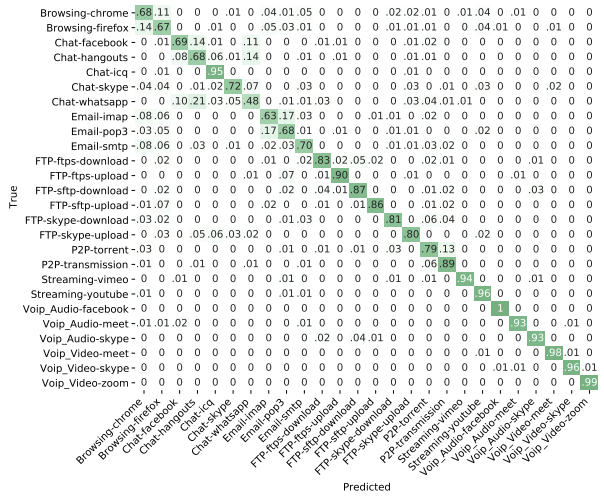


(a) Traffic Characterization on Cache Contention data confusion on LSTM classifier.

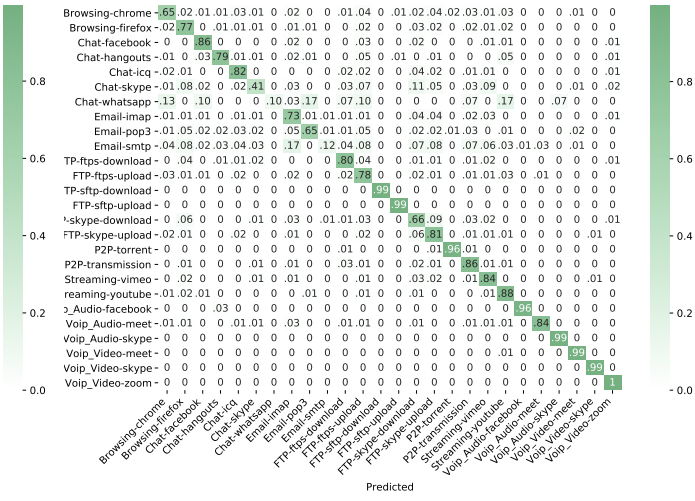


(b) Traffic Characterization on Network data confusion on CNN classifier [17].

Figure 8: Application Classification on VPN data Confusion Matrices.

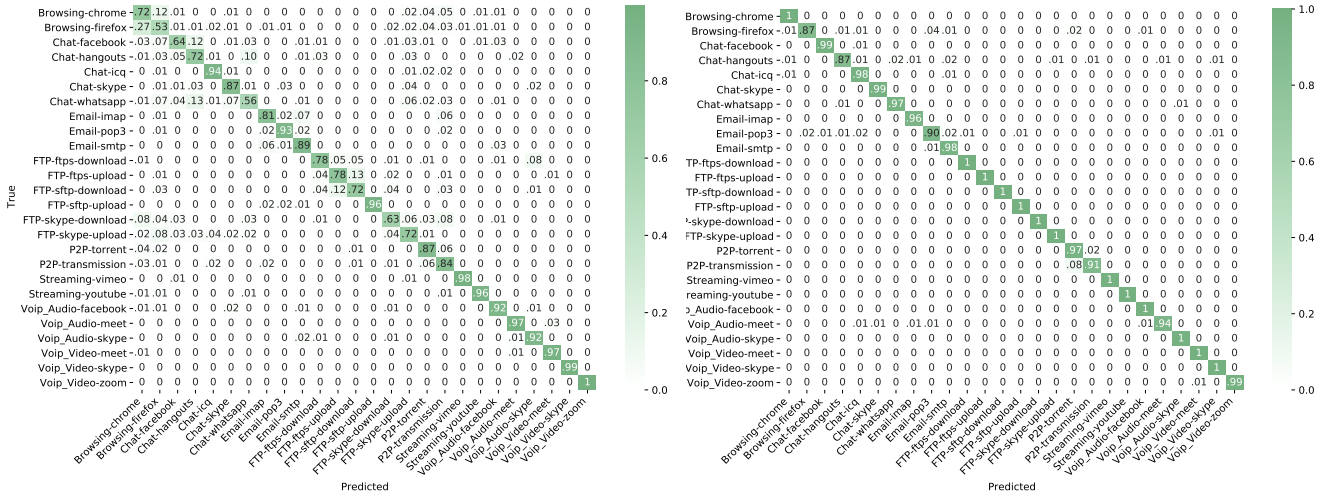


(a) Application Classification on Cache Contention data confusion on LSTM classifier



(b) Application Classification on Network data confusion on CNN classifier [17].

Figure 9: Application Classification on No VPN data Confusion Matrices



(a) Application Classification on Cache Contention data confusion on LSTM classifier

(b) Application Classification on Network data confusion on CNN classifier [17].

the Cache-LSTM classifier obtains a top-1 F1 score that is 30% higher than that of the Network-CNN classifier. When we observe the top-3 F1 score, the case in which there is some prior information on the user's activity on the network, we can see that the results of the classifiers increase to 75.1% and 41.7% for cache and network classifiers respectively. On the application classification task, the results of the Network-CNN classifier drop to the base rate for both the top-1 and top-3, whereas the Cache-LSTM classifier achieves a rate significantly above the base rate, with results of over 25% and over 50% for the top-3 F1 score.

5. Discussion

Our results show that cache-based characterization can be used as a low-infrastructure alternative to network-based traffic and application characterization. The cache-based method achieves similar results to the network-based method in most scenarios, and is significantly better in detecting traffic in the cross-traffic classification task.

5.1. Real-World Practicality

When deploying our proposed system in a real-world production setting, two additional factors need to be considered: the performance and communications overhead related to a large-scale system deployment, and the effects of adversarial users who intentionally wish to circumvent the classification system.

The machine learning training process is performed offline, and as such has no effect on the dynamic runtime performance of the system. The only overheads we therefore need to consider are related to the online behaviour of the system, and can be split into two parts – the communications overhead between the endpoints and the server, and the inference time on the server itself. Each cache trace, which is the item of data sent from the endpoints to the server, consists of 5000 16-bit integer values, collected over a period of 30 seconds. Even if we use very inefficient uncompressed JSON encoding for this data, the worst-case outgoing bandwidth from each endpoint is less than 10 kilobits per second, a value unlikely to have an impact on the endpoint's network performance. The inference speed of the cache LSTM classifier, as measured on our dual socket Intel Xeon E5-2660 server, is less than 25 microseconds per inference, equivalent to over 40K classifications per second. We can therefore conclude that the runtime performance of the system is appropriate for large-scale deployment.

Network-based classifiers can be intentionally circumvented by adversarial users, for example by hiding the BIND version for the DNS server, by using non-standard ports or proxy servers, or by applying traffic shaping and filtering methods such as BuFLO [4, 3], which completely remove all signal

Table 3: Application Classification per label Comparison

	NoVPN				VPN			
	Cache-LSTM		Network-CNN		Cache-LSTM		Network-CNN	
	precision	recall	precision	recall	precision	recall	precision	recall
Browsing-chrome	0.67	0.69	0.71	0.65	0.64	0.73	0.97	1.00
Browsing-firefox	0.69	0.68	0.71	0.77	0.65	0.53	0.87	0.90
Chat-facebook	0.74	0.70	0.81	0.86	0.74	0.65	0.98	0.99
Chat-hangouts	0.57	0.68	0.77	0.78	0.70	0.73	0.87	0.85
Chat-icq	0.91	0.95	0.83	0.82	0.83	0.95	0.95	0.98
Chat-skype	0.83	0.72	0.50	0.38	0.81	0.85	0.99	0.99
Chat-whatsapp	0.57	0.49	0.10	0.10	0.72	0.57	0.97	0.97
Email-imap	0.66	0.64	0.74	0.73	0.85	0.81	0.98	0.96
Email-pop3	0.68	0.67	0.75	0.66	0.93	0.93	0.84	0.88
Email-smtp	0.77	0.69	0.38	0.13	0.87	0.89	0.94	0.98
FTP-ftps-download	0.83	0.83	0.83	0.81	0.83	0.78	0.99	1.00
FTP-ftps-upload	0.91	0.91	0.69	0.78	0.82	0.80	1.00	1.00
FTP-sftp-download	0.86	0.88	0.98	0.99	0.76	0.72	0.99	1.00
FTP-sftp-upload	0.95	0.86	0.98	0.99	0.98	0.96	0.99	1.00
FTP-skype-download	0.76	0.82	0.63	0.67	0.73	0.63	0.99	1.00
FTP-skype-upload	0.79	0.79	0.69	0.80	0.63	0.71	0.99	1.00
P2P-torrent	0.82	0.79	0.95	0.96	0.85	0.87	0.91	0.97
P2P-transmission	0.84	0.89	0.79	0.86	0.78	0.84	0.96	0.91
Streaming-vimeo	0.98	0.94	0.78	0.84	0.99	0.98	0.99	1.00
Streaming-youtube	0.89	0.96	0.82	0.88	0.97	0.96	1.00	1.00
Voip Audio-facebook	0.97	0.99	0.99	0.96	0.92	0.92	0.98	0.99
Voip Audio-meet	0.95	0.93	0.96	0.84	0.96	0.97	0.98	0.93
Voip Audio-skype	0.95	0.94	0.99	0.99	0.93	0.92	0.99	1.00
Voip Video-meet	0.98	0.98	0.95	0.99	0.96	0.97	0.97	1.00
Voip Video-skype	0.97	0.97	0.97	0.99	0.99	0.99	0.98	1.00
Voip Video-zoom	0.98	0.99	0.93	1.00	1.00	1.00	0.99	0.98
Weighted Average	0.84	0.83	0.83	0.84	0.86	0.85	0.96	0.97

from the network trace, making classification impossible. Similarly, cache-based classifiers can also be targeted and misled by adversarial users. For example, the collection script itself may be disabled by the user, for example by the use of a customized script blocker. The user may also attempt to run noise-generating processes on the system that artificially increase the noise level of the cache contention signal, making analysis more difficult. As shown in related works, both of these approaches cannot completely defeat our approach. Cache contention measurements have been shown to be possible even with scripting completely disabled [31], meaning that as long as the user is accessing company-provided websites, his computer can be analyzed, even with a script blocker active. Cache-based measurements have also been shown to withstand traffic shaping countermeasures, and mechanisms that introduce noise to the cache can only attenuate, but not completely defeat, cache-based website fingerprinting [29].

5.2. Evaluation in a Work-From-Home Setting

One unique advantage of our approach is its ability to classify the traffic of computers located outside the corporate network. This trend of employees working from home is increasingly prevalent due to the repercussions of the COVID-19 pandemic.

Table 4: Model Performance comparison

		Network-CNN [17]		**Cache-LSTM**	
Task		**Top-1**	**Top-3**	**Top-1**	**Top-3**
Traffic	Train: Non-VPN / Test: VPN	15.5±1.3	41.7±2.2	50.3±2.3	75.1±3.0
Characterization	Train: VPN / Test: Non-VPN	13.5±1.8	36.7±2.3	48.6±2.7	77.9±2.5
Application	Train: Non-VPN / Test: VPN	4.6±0.8	15.22±1.8	24.0±1.8	51.1±2.4
Classification	Train: VPN / Test: Non-VPN	3.7±0.7	11.5±1.8	26.6±2.4	51.9±2.3

To evaluate this setting, we set up a Wireguard VPN connection between an endpoint in Israel and a VPN server in the United States, and used this setting to capture a sequence of 10-minute web browsing sessions, using both the Chrome and the Firefox browsers. Next, we attempted to classify the traffic type of these captured traces, using an LSTM classifier trained in a local deployment setting. Even without modifying the classifier model, the pre-trained LSTM classifier was able to classify these traces with a recall rate of 60.1±4.8%, a value significantly higher than the base rate of 12.5%.

5.3. Comparing our Approach to Network-Based Classifiers

Network-based traffic classification is a useful tool which is deployed and used by many network operators and organizations. It is interesting to consider the pros and cons of our cache-based approach, when compared to classical network-based traffic classification. In terms of raw accuracy, as our results demonstrate, the cache-based method has similar or better performance to the network-based method. The main difference between the network-based method and the cache-based method is in the added infrastructure requirement – the network-based method requires adding a costly MITM interception point that collects and analyzes all traffic entering and leaving the organization. Once this extra hardware is installed, no additional changes need to be made to the endpoints themselves, and a single unit of interception hardware can analyze the traffic of the entire organization. A user inside the organization can defeat the MITM interception point by using an alternative connection to the Internet, for example a cellular hot spot. The cache-based method, on the other hand, does not require this interception point. Instead, it requires that a small amount of unprivileged JavaScript code be run on each computer to be analyzed. The cache-based method is applied to all of the user's activity, regardless of the type of network connection used.

Comparing the two approaches, the network-based approach may be easier to deploy for organizations with the financial, legal, and technical ability to force all traffic to flow through a MITM interception point, and to force standard hardware and networking configurations on all users inside the organization. The cache-based approach, on the other hand, may be more appropriate for companies interested in occasionally analyzing the traffic patterns of a subset of users without an expensive investment in monitoring hardware.

5.4. Dataset Availability

The most important resource for traffic characterization studies is a labeled dataset. Collecting unlabeled network or cache contention traces might be easy, but collecting a labeled dataset requires hands-on manual data collection and labeling performed by humans in order to preserve authentic traffic characteristics. Currently, the most commonly used public dataset in the area of traffic characterization is the ISCXVPN2016 [7] dataset. The dataset compiled for our research may contribute more widely to the areas of network traffic analysis and cache contention-based traffic analysis. We captured traces of cache contention data related to network traffic data. The over 600GB dataset we created allowed us to compare various methods; it also promotes research in both areas. Furthermore, this dataset can be used by other researchers to evaluate and validate their methods; it can also help them avoid bias towards PCAP features such as IP addresses and ports, and also avoid bias towards specific world regions and network infrastructure parameters. Our full dataset of network trace PCAP files, the related JSON files containing the

labeled cache contention data, together with code reproducing the results, is available online [30].

5.5. Conclusions

In this work, we investigated the ability of cache contention traces to perform traffic characterization and application classification. While previous studies performed these tasks on data that could only be collected from within the organizational network, the proposed side channel data collection method expands the borders of traffic characterization, allowing us perform this task remotely, without being dependent on the user's current network.

We collect unique datasets of cache contention traces with their related network traces. These datasets contain traces collected under two types of network configurations: with and without VPN.

We propose two classifiers for the performance of traffic characterization and application classification tasks, a classical machine learning classifier and deep learning classifier. Our deep learning classifier, working on cache contention data, achieves results comparable with the state-of-the-art models applied to network traces.

While our classifier of cache-based classified achieves similar results on both settings of VPN and non-VPN dataset, the network classifier performane degrades while classifying non-VPN data. We can speculate that the network VPN data is less noisy than the non-VPN, since it is run on a more isolated environment.

Finally, we evaluate the classifiers on different settings of cross-traffic classification, in which the classifiers trained on non-VPN data are tested on VPN traces, and vice versa. In this case, cache-contention traces outperform the state-of-the-art network traffic characterization model, and show that they are more resilient to switching between network settings.

Acknowledgements

This research was supported by Israel Science Foundation grants 702/16 and 703/16 and by the Ben-Gurion University Negev-Tzin Scholarship Program.

References

[1] Nicholas Bloom. *How working from home works out*, 2020. URL `https://siepr.stanford.edu/research/publications/how-working-home-works-out`. 4

[2] Jo M. Booth. Not so incognito: Exploiting resource-based side channels in JavaScript engines. Bachelor thesis, Harvard, April 2015. 2

[3] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a distance: website fingerprinting attacks and defenses. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 605–616. ACM, 2012. doi: 10.1145/2382196.2382260. URL `https://doi.org/10.1145/2382196.2382260`. 4, 12

[4] Xiang Cai, Rishab Nithyanand, and Rob Johnson. Cs-buflo: A congestion sensitive website fingerprinting defense. In Gail-Joon Ahn and Anupam Datta, editors, *Proceedings of the 13th Workshop on Privacy in the Electronic Society, WPES 2014, Scottsdale, AZ, USA, November 3, 2014*, pages 121–130. ACM, 2014. doi: 10.1145/2665943.2665949. URL `https://doi.org/10.1145/2665943.2665949`. 3, 4, 12

[5] Julian Andres Caicedo-Muñoz, Agapito Ledezma Espino, Juan Carlos Corrales, and Álvaro Rendón Gallón. Qos-classifier for VPN and non-vpn traffic based on time-related features. *Comput. Networks*, 144:271–279, 2018. doi: 10.1016/j.comnet.2018.08.008. URL `https://doi.org/10.1016/j.comnet.2018.08.008`. 3

[6] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In Matt Blaze, editor, *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320. USENIX, 2004. URL `http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html`. 3

[7] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. Characterization of encrypted and VPN traffic using time-related features. In Olivier Camp, Steven Furnell, and Paolo Mori, editors, *Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016, Rome, Italy, February 19-21, 2016*, pages 407–414. SciTePress, 2016. doi: 10.5220/0005740704070414. URL `https://doi.org/10.5220/0005740704070414`. 3, 5, 6, 8, 9, 10, 14

[8] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S. C. Diot. Packet-level traffic measurements from the sprint ip backbone. *IEEE Network*, 17(6):6–16, 2003. 3

[9] Marc Juárez, Mohsen Imani, Mike Perry, Claudia Díaz, and Matthew Wright. Toward an efficient website fingerprinting defense. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I*, volume 9878 of *Lecture Notes in Computer Science*, pages 27–46. Springer, 2016. doi: 10.1007/978-3-319-45744-4_2. URL `https://doi.org/10.1007/978-3-319-45744-4_2`. 3

[10] Hyungsub Kim, Sangho Lee, and Jong Kim. Inferring browser activity and status through remote monitoring of storage usage. In *ACSAC*, 2016. 2

[11] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *IEEE SP*, pages 1–19, 2019.

4

[12] Tim Kosse. Filezilla - the free ftp solution. URL https://filezilla-project.org/. 7

[13] A. Meena Kowshalya, R. Madhumathi, and N. Gopika. Correlation based feature selection algorithms for varying datasets of different dimensionality. *Wirel. Pers. Commun.*, 108(3):1977–1993, 2019. doi: 10.1007/s11277-019-06504-w. URL https://doi.org/10.1007/s11277-019-06504-w. 3

[14] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading kernel memory from user space. In *USENIX Security*, pages 973–990, 2018. 4

[15] Chao Liu, Ryen W. White, and Susan T. Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In Fabio Crestani, Stéphane Marchand-Maillet, Hsin-Hsi Chen, Efthimis N. Efthimiadis, and Jacques Savoy, editors, *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 379–386. ACM, 2010. doi: 10.1145/1835449.1835513. URL https://doi.org/10.1145/1835449.1835513. 7

[16] Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B. Lee. Last-level cache side-channel attacks are practical. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 605–622. IEEE Computer Society, 2015. doi: 10.1109/SP.2015.43. URL https://doi.org/10.1109/SP.2015.43. 4

[17] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. Deep packet: a novel approach for encrypted traffic classification using deep learning. *Soft Comput.*, 24 (3):1999–2012, 2020. doi: 10.1007/s00500-019-04030-2. URL https://doi.org/10.1007/s00500-019-04030-2. 3, 6, 8, 9, 10, 11, 12, 14

[18] Evangelos P. Markatos, Spyros Antonatos, Michalis Polychronakis, and Kostas G. Anagnostakis. Exclusion-based signature matching for intrusion detection. In *In Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN*, pages 146–152. ACTA Press, 2002. 3

[19] Nikolay Matyunin, Yujue Wang, Tolga Arul, Kristian Kullmann, Jakub Szefer, and Stefan Katzenbeisser. Magneticspy: Exploiting magnetometer in mobile devices for website and application fingerprinting. In *WPES*, pages 135–149. ACM, 2019. 2

[20] Clémentine Maurice, Christoph Neumann, Olivier Heen, and Aurélien Francillon. C5: cross-cores cache covert channel. In Magnus Almgren, Vincenzo Gulisano, and Federico Maggi, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment - 12th International Conference, DIMVA 2015, Milan, Italy, July 9-10, 2015, Proceedings*, volume 9148 of *Lecture Notes in Computer Science*, pages 46–64. Springer, 2015. doi: 10.1007/978-3-319-20550-2_3. URL https://doi.org/10.1007/978-3-319-20550-2_3. 4

[21] Andrew Moore, Denis Zuev, and Michael Crogan. Discriminators for use in flow-based classification. Technical report, 2013. 3

[22] John Morrissey, T Saunders, Mark Lowes, Daniel Roesen, and Michael Renner. Proftpd: Highly configurable gpl-licensed ftp server software. URL http://www.proftpd.org/. 7

[23] Yossef Oren, Vasileios P. Kemerlis, Simha Sethumadhavan, and Angelos D. Keromytis. The spy in the sandbox: Practical cache attacks in javascript and their implications. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1406–1418. ACM, 2015. doi: 10.1145/2810103.2813708. URL https://doi.org/10.1145/2810103.2813708. 2, 4

[24] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of AES. In *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006. 4

[25] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website fingerprinting at internet scale. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016. URL http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/website-fingerprinting-internet-scale.pdf. 2

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 5

[27] Vera Rimmer, Davy Preuveneers, Marc Juárez, Tom van Goethem, and Wouter Joosen. Automated website fingerprinting through deep learning. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018. URL http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-1_Rimmer_paper.pdf. 2

[28] Justine Sherry, Chang Lan, Raluca Ada Popa, and Sylvia Ratnasamy. Blindbox: Deep packet inspection over encrypted traffic. *Computer Communication Review*, 45(5):213–226, 2015. doi: 10.1145/2829988.2787502. URL https://doi.org/10.1145/2829988.2787502. 3

[29] A. Shusterman, Z. Avraham, E. Croitoru, Y. Haskal, L. Kang, D. Levi, Y. Meltser, P. Mittal, Y. Oren, and Y. Yarom. Website fingerprinting through the cache occupancy channel and its real world practicality. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2020. 2, 4, 5, 6, 13

[30] Anatoly Shusterman, Chen Finkelstein, Ofir Gruner, Yarin Shani, and Yossi Oren. Datasets for cache-based and network-based traffic and application characterization, 2021. URL https://dx.doi.org/10.21227/bq8s-ps56. 2, 15

[31] Anatoly Shusterman, Ayush Agarwal, Sioli O'Connell, Daniel Genkin, Yossi Oren, and Yuval Yarom. Prime+probe 1, javascript 0: Overcoming browser-based side-channel defenses. In *USENIX Security Symposium*. USENIX Association, 2021 (to appear). 4, 13

[32] Payap Sirinam, Mohsen Imani, Marc Juárez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In

David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1928–1943. ACM, 2018. doi: 10.1145/3243734.3243768. URL `https://doi.org/10.1145/3243734.3243768`. 2

[33] Skype Technologies. Skype - communication tool for free calls and chat. URL `https://www.skype.com/`. 7

[34] Petr Velan, Milan Cermák, Pavel Celeda, and Martin Drasar. A survey of methods for encrypted traffic classification and analysis. *Int. J. Netw. Manag.*, 25(5):355–374, 2015. doi: 10.1002/nem.1901. URL `https://doi.org/10.1002/nem.1901`. 3

[35] Pepe Vila and Boris Köpf. Loophole: Timing attacks on shared event loops in Chrome. In *USENIX Sec*, pages 849–864, 2017. 2

[36] Tao Wang and Ian Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In Engin Kirda and Thomas Ristenpart, editors, *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, pages 1375–1390. USENIX Association, 2017. URL `https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao`. 3

[37] Baris Yamansavascilar, M. Amaç Güvensan, Ali Gökhan Yavuz, and M. Elif Karsligil. Application identification via network traffic classification. In *2017 International Conference on Computing, Networking and Communications, ICNC 2017, Silicon Valley, CA, USA, January 26-29, 2017*, pages 843–848. IEEE Computer Society, 2017. doi: 10.1109/ICCNC.2017.7876241. URL `https://doi.org/10.1109/ICCNC.2017.7876241`. 3, 8

Appendices

Appendix .1. Filtered ports

Ports and their protocol names, which were filtered before the training of network based models to enhance the classifier performance:

Prot Number	Protocol
53	DNS
67	DHCP
68	DHCP
123	NTP
137	NetBios
138	NetBios
139	NetBios
161	SNMP
427	SLP
546	DHCPv6
547	DHCPv6
1900	SSDP
5353	mDNS
5355	LLMNR

Appendix .2. Application Classification Labels

chrome
facebook
firefox
ftps-download
ftps-upload
hangouts
icq
imap
meet
pop3
sftp-download
sftp-upload
skype
skype-download
skype-upload
smtp
torrent
transmission
vimeo
whatsapp
youtube
zoom

Appendix .3. Hyperparameter tuning

Table .5: Hyperparameters search space of the LSTM neural network:

Hyperparameter	**Value**	**Space**
Optimizer	Adam	Adam,Adamax, Adadelta, RMSprop
Learning rate	0.001	0.001–0.002
Loss function	Categorical Crossentropy	
Epochs	50–80	Early stop by accuracy
Batch size	128	32 – 256
Input units	10000	500 – 90000
Convolution layers	2	1–4
Kernel size	16,8	2–31
Filters	256	2–512
Strides	3	1–4
Pool size	4	2–8
LSTM units	32	4 – 128

Appendix .4. Feature Extraction

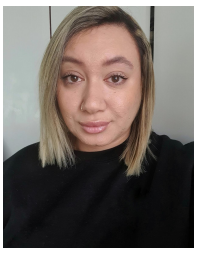
Table .6: Features function definition

Feature	Function
Spectral Centroid	$\frac{\sum_{N-1}^{n=0} f(n)x(n))}{\sum_{N-1}^{n=0} x(n))}$
Entropy	$-\sum_{x\in 0} x \log_2 x)$
Skewness	$\frac{\mu^3}{\sigma^3}$
Kurtosis	$\frac{\mu^4}{\sigma^4}$
Spectral Flatness	$\frac{exp(\frac{1}{N})\sum_{n-0}^{N-1} lnx(n)}{\frac{1}{N})\sum_{n-0}^{N-1} x(n)}$
spectral irregularity	$\sum_{k=2}^{N_b/2} \lvert a_{n,k} - a_{n,k-1}\rvert$

Anatoly Shusterman is a Ph.D student in the Department of Software and Information Systems Engineering in Ben-Gurion University of the Negev, Israel. He specializes in Machine Learning and Big Data Analytics. His current research interests include Cache Contention Side Channel Analysis and Deep Learning Algorithms.

Chen Finkelstein received her B.Sc degree from the Department of Software and Information Systems Engineering in Ben-Gurion University of the Negev, Israel.

Ofir Gruner received his B.Sc degree from the Department of Software and Information Systems Engineering in Ben-Gurion University of the Negev, Israel.



Yarin Shani received her B.Sc degree from the Department of Software and Information Systems Engineering in Ben-Gurion University of the Negev, Israel.

Yossi Oren received his M.Sc. degree in Computer Science from the Weizmann Institute of Science, Israel, and his Ph.D. degree in Electrical Engineering from Tel Aviv University, Israel, in 2008 and 2013 respectively. He is a Senior Lecturer (Assistant Professor) with the Department of Software and Information Systems Engineering in Ben-Gurion University, Israel. His research interests include implementation security (power analysis and other hardware attacks and countermeasures; low-resource cryptographic constructions for lightweight computers) and cryptography in the real world (consumer and voter privacy in the digital era; web application security).