# Survey of Encrypted Network Traffic Fingerprinting Techniques

A Thesis

Presented in Partial Fulfillment of the Requirements for the

Degree of Master of Science

with a

Major in Computer Science

in the

College of Graduate Studies

University of Idaho

by

Scott Jeffery

Major Professor: Michael Haney, Ph.D.

Committee Members: Robert Hiromoto, Ph.D.; Min Xian, Ph.D.

Department Administrator: Terry Soule, Ph.D.

May 2020

# Authorization to Submit Thesis

This thesis of Scott Jeffery, submitted for the degree of Master of Science with a Major in Computer Science and titled "Survey of Encrypted Network Traffic Fingerprinting Techniques," has been reviewed in final form. Permission, as indicated by the signatures and dates below is now granted to submit final copies for the College of Graduate Studies for approval.

| | | |
|---|---|---|
| Advisor: | Michael Haney, Ph.D. | Date |
| Committee Members: | Robert Hiromoto, Ph.D. | Date |
| | Min Xian, Ph.D. | Date |
| Department Chair: | Terry Soule, Ph.D. | Date |

# Abstract

Inspecting network traffic has been a staple technique of Cybersecurity tools for many years. This ability to review packet contents as they traverse an organization's network is hindered by the ever increasing use of encrypted communications. Without the visibility of deep-packet inspection, automated systems are unable to determine if network connections pose a threat to organizational interests or if they are supporting necessary day-to-day interactions.

One solution is a man-in-the-middle configuration, where an organization decrypts all traffic traversing its borders, however, this is cumbersome and computationally expensive as network speeds increase.

This thesis aims to survey the current landscape of "in the dark" network traffic fingerprinting, where encrypted payloads remain opaque to automated analysis leaving only network flow, packet header, and inferred metadata available for traffic classification.

# Acknowledgements

Thanks to my family for all of their support and sacrifices which made this work possible.

Thanks to Dr. Michael Haney for courses that were always worth the price of admission.

# Table of Contents

# List of Tables

# List of Figures

# List of Acronyms

**AES** Advanced Encryption Standard

**AIM** AOL Instant Messenger

**API** Application Programming Interface

**BYOD** Bring Your Own Device

**CDF** Cumulative Distribution Function

**CDN** Content Delivery Network

**CNN** Convolutional Neural Network

**CPD** Change Point Detection

**CPU** Central Processing Unit

**DBSCAN** Density-based Spatial Clustering of Applications with Noise

**DDoS** Distributed Denial of Service

**DNS** Domain Name System

**DPI** Deep-Packet Inspection

**EFF** Electronic Frontier Foundation

**FBI** Federal Bureau of Investigation

**FN** False Negative

**FP** False Positive

**FTP** File Transfer Protocol

**GMM** Gaussian Mixture Modeling

**GPU** Graphics Processing Unit

**HMM** Hidden Markov Model

**HTTP** Hyper-Text Transport Protocol

**HTTPS** Hyper-Text Transport Protocol Secure

**IANA** Internet Assigned Numbers Authority

**ICMP** Internet Control Message Protocol

**ID3** Iterative Dichotomiser 3

**IDS** Intrusion Detection System

**IP** Internet Protocol

**IPAT** Inter-Packet Arrival Time

**IPS** Intrusion Prevention System

**IRC** Internet Relay Chat

**ISP** Internet Service Provider

**ISRG** Internet Security Research Group

**JSON** JavaScript Object Notation

**k-Means** k-Means

**k-NN** k-Nearest Neighbors

**K-S** Kolmogorov-Smirnov

**ML** Machine Learning

**NIDS** Network Intrusion Detection System

**NLP** Natural Language Processing

**NSA** National Security Agency

**OS** Operating System

**OSI** Open System Interconnect

**P2P** Peer-To-Peer

**PCAP** Packet Capture

**PDF** Probability Density Function

**PGP** Pretty Good Privacy

**PHAD** Packet Header Anomaly Detection

**QoS** Quality of Service

**RC4** Rivest Cypher 4

**RIPPER** Repeated Incremental Pruning to Produce Error Reduction

**RSA** Rivest-Shamir-Adleman

**SMTP** Simple Mail Transfer Protocol

**SSH** Secure Shell

**SSL** Secure Sockets Layer

**SVM** Support Vector Machines

**TCP** Transmission Control Protocol

**TLS** Transport Layer Security

**TN** True Negative

**TP** True Positive

**TTL** Time To Live

**UDP** User Datagram Protocol

**VoIP** Voice over Internet Protocol

**VPN** Virtual Private Network

**WPA** Wi-Fi Protected access

**XML** Extensible Markup Language

# Chapter 1: Introduction

Human kind has never been so interconnected, nor so dependent on these interconnections, as we are today. The world-wide proliferation of Internet connected devices has affected nearly every aspect of our daily lives, from the conveniences, such as on-demand streaming television and 24-hour banking access, to the life-saving, such as on-line medical consultations and e911 emergency calling.

Along with the benefits of a connected world comes new types of threats. Interacting with others around the globe includes the potential for interactions with those who might seek to do us harm. For the early Internet, these potential ramifications were limited by the benign nature of connected devices. The first worms and viruses were able to penetrate systems and pilfer or delete data, but the fledgling global network was simply not reliable or fast enough to be entrusted with critical infrastructure or sensitive information. Over time Internet connections dropped in price, increased in reliability, and proliferated across the globe. Dramatically expanded storage capacities made it economically feasible to amass enormous amounts of data and make it available over the public network. The culmination of fast, reliable and ubiquitous Internet connections with ever-cheaper processing power led to an explosion of connected devices. In a short period of time everything from mom-and-pop credit card transactions to critical electrical power grid controls were leveraging the cheap and reliable global network for daily operations.

## Cyber-Attacks Go Dark

Unfortunately, attaching business critical systems to the global network also meant more attractive, and potentially lucrative, targets for cyber-attacks. Initially, Internet malware was unlikely to cause disruption. Early worms and viruses, such as the Morris Worm or Michaelangelo Virus, might cause network outages or affect a small number of systems [30], but the risks were mostly limited to one-off inconveniences and annoyances. Eventually attacks evolved in prevalence and sophistication, culminating with extremely profitable ransomware attacks and Distributed Denial of Service (DDoS) for hire as an emerging business model [37]. Attackers began leveraging encryption as a means to obscure their activities, or, in the case of ransomware like CryptoLocker [33], as an integral part of their attacks.

## A Post-Snowden World

In 2013 Edward Snowden altered public perception with the release of National Security Agency (NSA) documents detailing how the governmental agency had gathered information on millions of Americans [16]. What was previously considered constitutionally protected communications was now fair game for

agency collection without any warrants or judicial overview.

The public responded by turning to encrypted communications platforms. Previously obscure messaging applications like Telegram, WhatsApp, and Signal Messenger saw their user base growing rapidly while existing services added encryption protections. The WhatsApp platform alone saw monthly usage numbers explode from 200 million in 2013 to 1.5 billion in 2017 [10].

At the same time a coalition formed with the goal of encrypting the web. In 2014 well-known sponsors like the Electronic Frontier Foundation (EFF), the Mozilla Foundation, Cisco, Google, and others came together to form the Internet Security Research Group (ISRG) and it's "Let's Encrypt" certificate authority service. By 2016 Let's Encrypt began providing freely available encryption certificates, enabling web-site owners to support encryption protected Hyper-Text Transport Protocol Secure (HTTPS) connections for their users.

These and other efforts to implement ubiquitous encryption were impactful enough that they prompted both Federal Bureau of Investigation (FBI) director James Comey and U.S. Attorney General William Barr into action against the "Going Dark" problem [12][16]. In 2014 Comey explained that he believes the "law hasn't kept pace with technology" and that, due in part to these post-Snowden changes, "real-time communication and stored data are increasingly encrypted" which he called "Going Dark." Barr later echoed these sentiments in 2019 stating: "I am here today to tell you that, as we use encryption to improve Cybersecurity, we must ensure that we retain society's ability to gain lawful access to data and communications when needed to respond to criminal activity."

In fact, recent Google transparency reports [17], collected from their Chrome Browser, confirm the increasing prevalence of encrypted communications, showing that today over 80% of browsing sessions leverage encrypted HTTPS communications, nearly doubling since 2017.

## Cybersecurity's ability to protect

Reliance on network technologies for critical business processes places the burden on Internet Service Providers (ISPs) and enterprise network administrators to provide reliable and secure services. The ability to offer reliable service requires network administrators understand the nature of the traffic they carry; weeding out threats while allowing legitimate traffic to pass. Initially this was accomplished with Firewalls and virus scanners. As attacks became more sophisticated these early tools evolved into Intrusion Detection Systems (IDSs), and later Intrusion Prevention Systems (IPSs), capable of dynamically watching a network for anomalous behaviors and, in the case of an IPS, making system changes to protect against threats. These tools worked for a time, but, again, cyber threats evolved and, this time, administrators answered with Deep-Packet Inspection (DPI) [8], extending the traditional

IDS/IPS model by examining packet payloads rather than relying on the usual header information with IP addresses, ports, etc. Of course DPI's ability to dig down to the packet payload level is most effective with unencrypted traffic.

## Encrypted network traffic classification

In our post-Snowden world, with more and more communications protected by encryption, DPI tools no longer have the payload visibility required to remain effective. The more that communications are protected by encryption the less DPI is able to determine which connections are malicious and which are valid traffic. Meanwhile, service providers, still expected to provide robust and secure network access, are searching for a viable alternative to DPI.

Enterprise scale networks might attempt to mitigate this lack of visibility using Secure Sockets Layer (SSL) interception to unwrap encrypted packets traversing their networks. Specialized network appliances are configured as a proxy between communicating systems. The appliance acts as a "go-between", establishing a separate SSL encrypted channel to each end-point over which messages are collected, decrypted, re-encrypted, and forwarded to the other side. Communications between the two end-points are still protected, but the appliance is able to view the un-adulterated messages as they are passed along, providing an opportunity to detect malicious activity in the process.

This method is only feasible where at least one endpoint is owned by the enterprise and can have the proxy's encryption certificate inserted into the endpoint's trusted cryptographic store. Organizations with policies that allow Bring Your Own Device (BYOD) or those that provide service outside of a corporate network (such as cell-phone carriers, hotel WiFi services, and ISPs) are still left in the dark.

Another, more universal, option is leveraging Data Science techniques to profile network traffic behaviors without the need for decryption, DPI, or proxying appliances. With this technique, Machine Learning (ML) models are "trained" to recognize certain types of network traffic based on the behaviors of the end-points rather than the content of the messages. Behaviors useful for model training might be any combination of available data or meta-data left untouched in the encrypted communication. These include source and destination Internet Protocol (IP) addresses, Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port number, packet size, inter-packet delivery time, session duration, etc. Despite the obfuscated nature of encrypted communications, many useful attributes are still available or can be derived from un-encrypted traffic elements.

For example, an ML Model might "learn" to differentiate an encrypted, interactive Secure Shell (SSH) session from a Netflix streamed movie based on the fact that SSH sessions tend to generate many small packets sent between endpoints, transmitting one key stroke from the client and the resulting response

from the server, as opposed to Netflix streams which are largely comprised of maximum size packets flowing from the Netflix server to the client. A network administrator may not be able to determine the SSH command transmitted or the Netflix movie being watched, but they receive enough visibility to determine if anyone was expected to SSH into the corporate HR system or if company policy allows for Netflix streaming to an employee's computer during work hours.

Applying ML techniques to network traffic classification has been studied for many years but has only recently become an economically feasible solution thanks to advances in Computer Science, Statistics, ML, and specialized hardware. In many ways we can thank the computer gaming industry for a large portion of these advances as the high-end Graphics Processing Unit (GPU) chips, which make game graphics more realistic, employ the same types of mathematics used for ML model training. Gamer demand for these specialized processors dropped the price, spurred innovations, and opened the door for Data Science specialists to explore techniques that would have otherwise been too complex for standard computer Central Processing Units (CPUs).

## Thesis purpose and goals

This thesis aims to survey the history and current state of encrypted traffic classification techniques looking at effectiveness and implementation complexity. A total of twenty published papers, spanning more than fifteen years, are reviewed and categorized into a taxonomy. The taxonomy approach ensures techniques are compared to other similar techniques, in an apples-to-apples fashion, as many different approaches aim for different outputs or work under varying assumptions around their inputs.

# Chapter 2: Taxonomy

Khalife et. al. [26] proposed a taxonomic approach for the research community to discuss traffic analysis and classification methods under a common framework. Figure 2.1 is a graphical representation of that taxonomy, describing the full chain of features for a given classification technique from input data type, through the classification technique, and resulting outputs.

## Methodologies categorized as taxonomy three-tuples

Using this three-tier hierarchy, each surveyed classification method can be characterized and systematically categorized, grouping common techniques into three-tuple representations (input, technique, and output). For instance, a publication describing techniques to fingerprint anomalous Transport Layer Security (TLS) connections based on the unique, clear-text list of cypher-suites supported by each end-point would belong to the (Input: Traffic Payload → Technique: Payload Inspection → Output: Flow → Anomaly) three-tuple category. Both the Input and Technique tuples indicate this method's reliance on the ability inspect the TLS negotiation payloads and extract the list of advertised cypher-suites. The Output tuple indicates the end goal is to flag anomalous end-points by profiling their list of supported cypher-suites and comparing those to know-good devices.

## Comparing apples to caramel apples

Where a given methodology might fit into more than a single taxonomic category, both three-tuples are assigned. For instance, a publication might describe how combining both Heuristic and Supervised Machine Learning techniques achieves more accurate results than either technique individually. In this case both the (Technique: Heuristic) and (Technique: Supervised Machine Learning) categories apply. By allowing multiple three-tuple labels, this multi-classed method can later be compared to other methods tagged with either (Technique: Heuristic) or (Technique: Supervised Machine Learning) as they share a similar foundation, even if one method might only apply a subset of the other. Presumably the synergy achieved with a combination of Heuristic and Supervised Machine Learning techniques is greater than either of the constituent parts alone, and comparing the combination to the individual techniques is necessary and prudent.

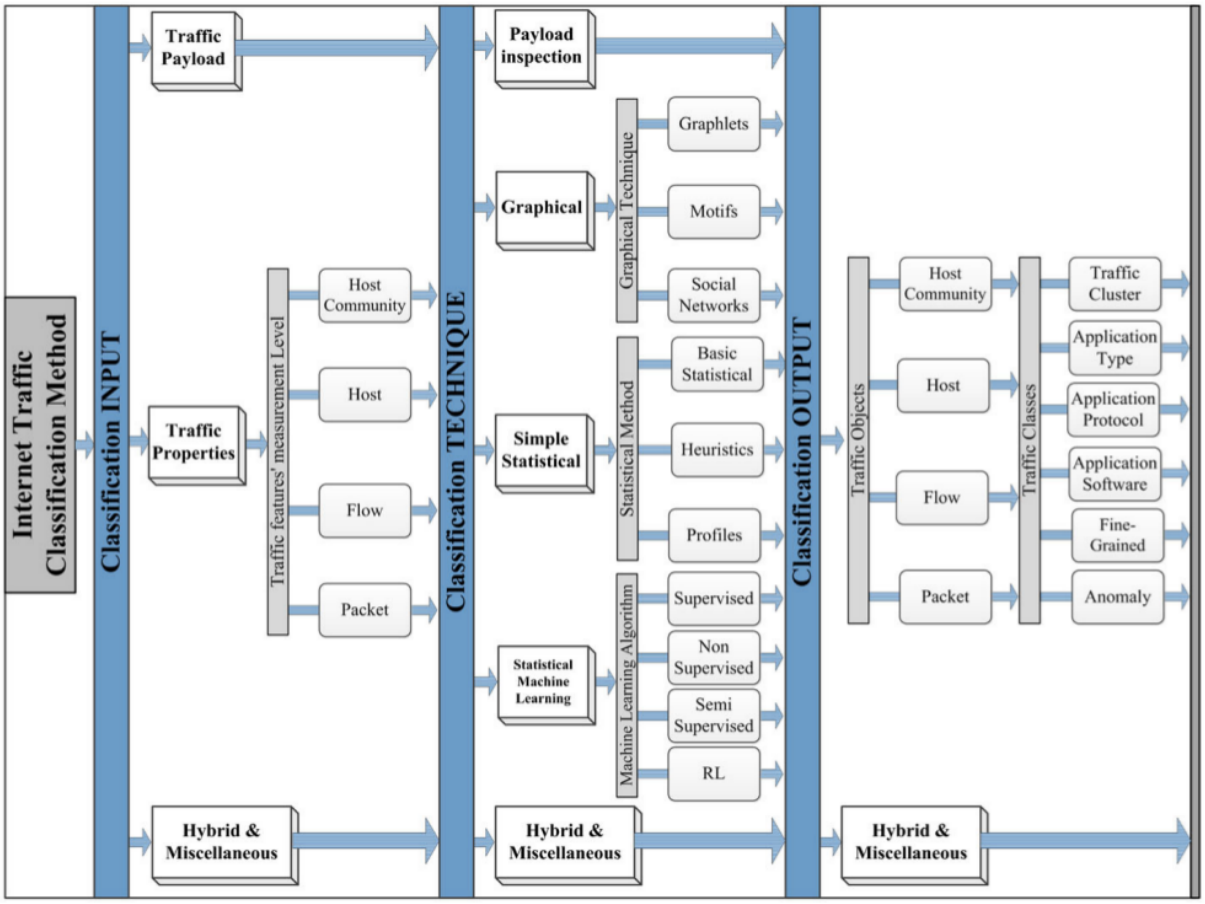



Figure 2.1: Khalife et al. Taxonomy [26]

| Classification Input | | |
|---|---|---|
| Traffic Payload | | use of application data |
| Traffic Properties (Measurement Level) | Host Community | graph metrics (diameter, connection degree) |
| | Host | number of connections, opened ports |
| | Flow | flow size, flow duration |
| | Packet | packet sizes, inter-arrival times |
| Hybrid & Miscellaneous | | combination of inputs, external knowledge |

Table 2.1: Velan, et al. Taxonomy Inputs [41]

| Classification Technique | | |
|---|---|---|
| Payload Inspection | | DPI, examination of first N bytes |
| Graphical Techniques | Graphlets | relationship between features (ports, addresses) |
| | Motifs | patterns of communication |
| | Social Networks | graph of communication |
| Statistical Method | Basic Statistical | probability density functions of e.g. packet sizes |
| | Heuristics | port-based classification |
| | Profiles | host profiling, usage of packet sizes and direction |
| Machine Learning Algorithm | Supervised | Hidden Markov Models, Naive Bayes, k-nearest neighbor, support vector machine |
| | Non-Supervised | clustering of unlabeled traffic, k-nearest neighbor |
| | Semi-Supervised | clustering of mixed traffic, k-nearest neighbor |
| | Reinforcement Learning | - |
| Hybrid & Miscellaneous | | combination of methods, external knowledge |

Table 2.2: Velan, et al. Taxonomy Techniques [41]

## Taxonomy terms explained

The Khalife taxonomy [26] diagram (Figure 2.1) uses condensed terms that need some extra context for clarity. Velan, et al. [41] built tables of examples, for each taxonomy layer, in their earlier survey paper (2014). These are replicated in Tables 2.1, 2.2, and 2.3 and are included here to provide a quick reference reminder for each term's context.

## Taxonomy Terms Defined

While the Velan, et al. tables are useful as a quick reference, they are still not sufficient to fully disambiguate the taxonomic terms. Appendix A elaborates on both Khalife, et al. and Velan, et al. with expanded definitions and examples where applicable.

–

| **Classification Output** | |
|---|---|
| **Traffic Objects** | |
| Host Community | host community is assigned a class, e.g., community of HTTP servers |
| Host | host is assigned a class |
| Flow | flow is assigned a class |
| Packets | packet is assigned a class |
| **Traffic Classes** | |
| Traffic Cluster | bulk or small transactions |
| Application Type | game, browsing, chat |
| Application Protocol | HTTP, HTTPS, FTP |
| Application Software | client software such as Mail client, FTP client or web browser |
| Fine-Grained | Skype voice call, Google search, Facebook chat |
| Anomaly | port scan, brute-force attack |
| Hybrid & Miscellaneous | combination of outputs or classes, external knowledge |

Table 2.3: Velan, et al. Taxonomy Outputs [41]

# Chapter 3: Statistics and Machine Learning Primer

This chapter is intended to provide a basic background for the most referenced statistical and machine learning techniques among the surveyed publications. The following machine learning algorithms are categorized according to the Khalife taxonomy [26], grouped into Supervised (requiring labeled input data), Non-Supervised/Unsupervised (not requiring labeled input data), and Semi-Supervised (using a combination of labeled and un-labeled data) machine learning techniques. Descriptions of any non-machine learning techniques referenced in the surveyed papers (e.g. statistical or Gaussian modeling approaches) are included at the end of this chapter.

## Supervised Machine Learning Techniques

Supervised Machine Learning algorithms are able to learn a mapping from each input data record to a target output attribute, typically a class label. This is usually accomplished in iterative phases where a model is repeatedly tested with some portion of the input set, output class predictions are evaluated, and model parameters are adjusted to compensate for prediction errors. With each iteration the model's predictive error rates should decrease, producing an updated model which better fits the available data.

Since each training iteration compares the model's predictions to some previously determined answers, every input record must be labeled with the correct output class beforehand. In this way the training of the model is "supervised" by the correctly labeled outputs, continually refining the model's effectiveness by measuring predictive errors.

### Logistic Regression

Logistic Regression [39] is a probability based classification algorithm. It can be used to determine the probability of a record belonging to a given class or the probability of an event occurring based on previous observations. With this algorithm an "S" shaped curve, called the "sigmoid" or "logistic" function, is adjusted to best fit all previous observations. This sigmoid curve is bounded between 0 and 1 and represents the probability of class membership between 0% and 100%.

Applied to encrypted network classification, Logistic Regression can be useful for answering questions like "is this an encrypted packet based on its entropy attributes?" or "is this flow an SSH connection based on average payload size?"

### Naive Bayes Classifier

Naive Bayes [35] [39] is one of the simplest algorithms and is based on Bayes' theorem. The theorem defines the probability of an event or class membership based on prior observations and takes the mathematical form:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.1}$$

Where:

- P(A|B) is the probability of A occurring given B is true
- P(B|A) is the probability of B occurring given A is true
- P(A) is the probability of observing A
- P(B) is the probability of observing B

Naive Bayes has an interesting feature in that it is an "incremental learner" [35]. This means a Bayesian model can be updated as new data elements are received without re-processing every training element again. When applied to large sets of network traffic data this feature can be very useful as the entire training set does not need to stored in a computer's memory for the a Bayesian model to be updated.

### Random Forests

Random Forests [39] can be used in several capacities, such as classification or regression, and improve upon traditional decision trees by applying ensemble learning techniques to reduce over-fitting.

The core algorithm is a modified form of "bagging" where input samples are split into a tree structure by dividing elements on a single randomly selected feature at each step. This process repeats many times to build a large number of trees, or a random forest, each of which may or may not be good estimators of the original data. When tree building is complete, new samples are fed into each of the trees where each tree's output prediction is counted as one vote. The result with the highest number of votes is taken as the final prediction for the system.

Taking votes from many poor learners to build a good learner is a foundational assumption in ensemble learning and, in this case, has the added benefit of reducing the over-fitting problem as many of the generated trees are poor learners in the first place.

### C4.5 Algorithm

C4.5 is a decision tree classifier which recursively constructs the tree model that gives the best information gain [27] [2]. Based on Iterative Dichotomiser 3 (ID3), C4.5 tries to find small (or simple) decision trees by measuring the entropy of the data elements assigned to each node and calculating the information gain (or entropy loss) if that node were split during the next recursive call. The recursive process repeats until a stopping condition occurs, such as all elements assigned to a node are of the same class.

Once the recursion is complete, and a tree is constructed, branch pruning is used to reduce the over-fitting problem and the increased possibility of classification errors that result from it. Tree branches that do not provide any useful information are replaced with a leaf node containing all elements from all branch nodes.

### RIPPER

Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [27] is a three-stage rule-induction algorithm that, like other rule-based algorithms, attempts to build a set of If-Then style rules that properly classify input data. The process begins by ordering input classes by increasing prevalence. A rule is found that separates the first (least prevalent) class from all others. Second, the newly created rule is then pruned using a pruning measure formula based on the number of positive and negative examples. This process is repeated for each input class until the entire input set is covered by rules classifying each element. Finally, one extra pass reviews each rule optimizing where possible.

According to [27], as of 2018 RIPPER is considered state of the art in rule induction and has the added bonus of generating rule sets that are human interpretable.

### k-Nearest Neighbors

Unlike many classifiers, k-Nearest Neighbors (k-NN) [39] does not train a model. Instead, this algorithm places the labeled input data in an n-dimensional Cartesian space and clusters that data with a known clustering algorithm. As new data elements become available, they are assigned a class based on the Euclidean distance from the new observation to its "k" nearest neighbors. If "k" is one, the new element is assigned the class of its single nearest neighbor. If "k" is greater than one, the new element is assigned the same class as the majority of its "k" nearest neighbors.

The effect of this is to partition the Cartesian space with each segment assigned a class label. Any new data that falls within the partitioned space, where decision boundaries are defined by the majority

of nearest neighbors, is then assigned that class.

### Support Vector Machines

Like other classifiers, Support Vector Machines (SVMs) [39] attempt to partition data elements into classes on an n-dimensional Cartesian space. SVMs segregate classes by finding decision boundaries, based on carefully selected observations, that maximize margins between the decision boundary and the observations. This maximal margin approach allows for some outliers to be mis-classified as long as they don't affect the overall classification accuracy.

For overlapping data or observations that are not linearly separable, SVMs map low-dimensional data to higher dimensions and then calculate the decision boundary that best fits the new structure.

### AdaBoost

AdaBoost [39] [13] (short for Adaptive Boosting) is one among a large family of boosting algorithms. AdaBoost uses an ensemble learning technique to combine a group of weak-learners into one strong-learner by assigning some of the weak-learners a higher weighted input to the final result. In this case a weak-learner is a machine learning algorithm with only slightly better results than random guessing.

Many weak-learners are created over several iterations each time taking into account the previous learner's error rate. The error is determined my measuring how many samples the previous weak-learner mis-classified and this error rate determines how much weight that weak-learner will be given in the final product based on formula 3.2:

$$Weight = \frac{1}{2} log(\frac{1 - TotalError}{TotalError}) \tag{3.2}$$

Ultimately AdaBoost leverages the idea that a group of weighted weak-learners perform better, and are faster to train, than one strong-learner.

### Change Point Detection (Supervised)

Change Point Detection (CPD) [3] looks for distinct shifts in time series data. These shifts could represent anything from a problematic change in patient health to a new trend in financial markets.

Typically change point detection methods come in two categories: on-line and off-line. Off-line change point detection refers to analysis of static datasets that have already been collected. Off-line algorithms can process the entire dataset at once and are often more accurate because of it. On-line change point detection refers to analysis of real-time streaming data and typically aims for immediate responses to any changing conditions. Since the on-line algorithm variants must quickly react to incoming data points

they typically process only the most recently received data or a small history window in the data stream.

CPD can be accomplished with both supervised and unsupervised algorithms. Aminikhanghahi, et al. [3] describe several supervised machine learning methods used for Change Point detection including Multi-Class and Binary-Class Classifiers (Decision Tree, Support Vector Machine, Naive Bayes, Gaussian Mixture Modeling, etc.)

### Hidden Markov Models

Hidden Markov Models (HMMs) [45] attempt to model the state of a system, which is assumed to follow a Markov process, using only imperfect indirect observations. In this case a Markov process is a set of system states and the probability that a system will transition between those states in continuous time.

HMMs fall under the Supervised Machine Learning category as some information must be known about the hidden system's states, state transition probabilities, and the imperfect observation probabilities before HMMs are useful for predictions.

Applied to encrypted traffic classification, HMMs are useful in modeling network connection states such as encrypted protocol handshakes. In this case the the state of the connection is hidden within an encrypted channel, however the size of TCP packets flowing between participants are observable and can indicate the probable current system state based on these indirect observations [45].

### XGBoost

XGBoost is useful for both regression or classification problems [9] and starts out with an initial prediction which can be any starting value. Observed samples are compared to the initial prediction and an XGBoost tree is fitted to the residual sample values. All residual samples are assigned to a tree node and the node's similarity score is calculated, including a regularization parameter. The node is then split at some point, placing some of the residual sample values into one child node and the rest into another. The same similarity score is again calculated for each child node. To determine if the split provides a better fit to the input data an overall gain score is calculated for that split. This is repeated for all possible division points, splitting the samples and calculating overall gain for that split, until the highest possible gain is found. This process is repeated, splitting each leaf node and calculating the best gain values until a maximum tree height limit is reached or until a minimum leaf membership threshold (least number of samples a leaf node must contain) is crossed.

After the full tree is built, a pruning round is used to determine if the tree grew too large. Each leaf node is evaluated against a tree complexity parameter, gamma. If the leaf node's gain is less than the

gamma parameter it is pruned. This effectively ensures the final splitting of observations provides a high enough gain to be worth keeping.

### Convolutional Neural Networks

Typically applied to image recognition and computer vision problems [46], this specialized version of deep neural network is particularly good at sub-sample pattern detection and it is this strength that makes them useful across multiple disciplines. Like other types of neural networks, Convolutional Neural Networks (CNNs) have an input layer, output layer, and several hidden layers. Hidden layers, known as "filters", are applied convolutionally, moving around the input data to match sub-patterns defined by each filter. Pattern detection is typically done with a multiplication or dot product between the filter and input data and the output from convolved filter application is sent to the next network layer to detect increasingly complex patterns.

Periodically a pooling layer may be applied to combine detected patterns and reduce data dimensionality. This process can help streamline computational complexity and speed up overall network performance.

Finally, the detected patterns are passed to the output layer where match likelihood scores are presented for each class. Typically the highest scoring class is the assumed match for the given input data.

## Unsupervised/Non-Supervised machine learning techniques

Unsupervised Machine Learning algorithms map input data records to an output grouping by looking for patterns in a given dataset. Typically this is accomplished by examining how records interrelate with each other, forming natural clusters of records that share similar attributes. The machine learning algorithm calculates cluster boundaries allowing for new data to be quickly assigned to a grouping, even if the new record's attributes do not clearly indicate a group assignment by falling near a cluster boundary.

Since unsupervised models do not compare detected clusters and calculated boundaries to some predetermined answer there is no need for labeled input data, and thus no "supervision" of the algorithm.

### K-Means

The K-Means algorithm is an unsupervised machine learning technique that attempts to partition features into non-overlapping subgroups of similar elements. Similarity is determined solely by proximity to other data in an N-Dimensional Euclidean space and the number of assumed subgroups is a required input to be determined by other means.

K-Means [39] arranges available features in an N-Dimensional Euclidean space and starts with k

assumed centroids among the data, each of which represents one cluster, or subgroup, of similar elements. The position of each centroid is refined as the feature data nearest each centroid are assigned membership to that cluster and the centroid is repositioned to the arithmetic mean of the surrounding data. This process repeats until cluster membership no longer changes and the centroids no longer move relative to the nearby data points.

### Spectral Clustering

Spectral Clustering [39] [6] turns a clustering problem into a graph partitioning problem by mapping feature data onto graph nodes and then working on the generated graph. Observed data elements are transformed into a similarity graph with each of the graph's edge values representing the similarity between the two connected nodes. Since members of the same cluster may fall far apart in the current dimensional space, the graph is projected into a lower dimensional space, revealing which features are closer in the reduced space. Finally, the transformed data is clustered using a traditional technique like K-Means.

The added steps make Spectral Clustering more effective on complex cluster shapes that traditional techniques struggle with. Despite the additional complexity, Spectral Clustering is computationally lightweight as most of the calculations are accomplished with matrix math.

### Density-Based Spatial Clustering of Applications with Noise

Like most clustering algorithms, Density-based Spatial Clustering of Applications with Noise (DBSCAN) attempts to partition data points in an N-Dimensional space into groups with each element assigned group membership. DBSCAN takes two parameters: neighborhood radius and a minimum cluster size. A random point is selected as a start node and any neighboring points are added to a cluster if they lie within the neighborhood radius from the start node. This process continues, checking for more nearby points around the newly added cluster members until no new elements can be reached within the pre-determined neighborhood radius distance. Any un-clustered points are then used as a new starting point and the process repeats. If a cluster does not have enough members to meet the minimum cluster size parameter it is marked as anomalous.

### Change Point Detection (Unsupervised)

Change Point Detection [3] looks for distinct shifts in time series data. These shifts could represent anything from a problematic change in patient health to a new trend in financial markets.

Typically change point detection methods come in two categories: on-line and off-line. Off-line change point detection refers to analysis of static datasets that have already been collected. Off-line algorithms

can process the entire dataset at once and are often more accurate because of it. On-line change point detection refers to analysis of real-time streaming data and typically aims for immediate responses to any changing conditions. Since the on-line algorithm variants must quickly react to incoming data points they typically process only the most recently received data or a small history window in the data stream.

Change Point Detection can be accomplished with both supervised and unsupervised algorithms. Aminikhanghahi, et al. [3] describe several unsupervised machine learning methods used for Change Point detection including Likelihood Ratio, Subspace Model, Probabilistic Methods, and Clustering techniques.

## Semi-Supervised machine learning techniques

Semi-Supervised machine learning techniques take a hybrid approach, feeding both labeled and unlabeled data to an unsupervised algorithm. This approach gives the algorithm a head-start by providing the data groupings which are already identified in the labeled records. Afterward, the clusters and decision boundaries can be further refined as un-labeled records are processed.

## Other statistical techniques

Beyond the typical machine learning techniques, some surveyed works leverage other algorithms and methodologies. The most prevalent of those are reviewed here.

### Reinforcement Learning

Reinforcement Learning is neural network based machine learning that, unlike typical supervised deep neural networks, does not require a labeled input set [38] [24]. Instead this class of algorithms reinforces "correct" learning with a reward mechanism while discouraging "incorrect" learning with a penalty. The reward/penalty feedback system is referred to as a "policy network" and takes the place of the gradient descent approach in traditional supervised neural network methods.

While reinforcement learning saves time spent collecting and labeling the training data required for supervised learning, it can often take longer to train a successful reinforcement learning network as many erroneous attempts must be made before happening upon a rewarding sequence of outputs. This is compounded by the need to tweak any penalty system to ensure a beneficial series of decisions followed by an incorrect one are not completely thrown out due to a single mistake.

### Gaussian Mixture Modeling

Probability Density Functions (PDFs) are often useful in modeling a given dataset, however, when observed data doesn't follow the normal probability distribution pattern (likely because more that one

class is represented in observed data) it is sometimes beneficial to combine two or more PDFs into a Gaussian Mixture model [39]. This is simply the process of adding two distributions together, in n-dimensional space, to form a new (and better fitting) distribution for modeling the data.

Just as with any PDF, the component Gaussians have a mean and co-variance matrix describing the centroid and overall shape. By combining the PDFs we can calculate the probability that a new data point belongs to one of the component distributions much like any other classification algorithm.

### Markov Chains

Traditional probability theory focuses on experimental outcomes where the result of a previous measurement does not affect the next. Markov Chains [18] provide a theoretical basis for systems where a previous outcome may affect the next. In this model a system is defined by a set of states and transition paths between them. Each state transition carries the probability that the transition will occur. In graph theory the states are the graph nodes and the transitions are weighted edges between those nodes, with the weight representing the probability of a state transition between the two connected nodes occurring.

Applied to encrypted traffic analysis, Markov Chains are useful for modeling state transitions in network connections, especially when looking for unique protocol initialization fingerprints. For example, a TLS connection moves through several states during encrypted tunnel initialization (connect, client hello, server hello, certificate exchange, etc.) that might be unique to the TLS implementation. By watching many protocol negotiations, and calculating the overall probability of each observed state change in the negotiation, an implementation's behaviors can be modeled as a Markov Chain and compared to other implementations by analyzing the differences in the Markov Chain structure.

## Network traffic analysis as a machine learning problem

At its core, machine learning works by placing features, or attributes, of known data elements into an n-dimensional space, with each feature mapped to one of the dimensional axes. For instance, when leveraging machine learning techniques in determining cancer cell malignant/benign status, one might amass a collection of previous cellular biopsy images and map cell width to one axis, cell height to another axis, and cell shape (round, ovular or irregular) to a third axis. Once the available features are arranged across the n-dimensional feature space, machine learning algorithms can go to work finding interesting patterns and correlations within the data. Hopefully, this search results in a more efficient way to determine cancer malignancy.

Applying machine learning to encrypted network classification is achieved in a very similar manner. Rather than mapping cell features to an n-dimensional space, network traffic flow, packet, or payload

features are arranged on the dimensional axes. A Data Scientist might choose to map a connection's average transmit rate, average receive rate, and duration when analyzing traffic flows or they might choose to map source port, destination port, and sliding window acknowledgement counts when processing packet data. In analyzing network traffic, there are many possible features and potentially much, much more data available in Packet Capture (PCAP) files.

Moore, et al. [5] enumerated many of the relevant features, which they termed "discriminators", for analyzing network flow information. They came up with nearly 250 unique attributes useful for machine learning application. These include elements such as port, packet inter-arrival time, size, control bytes, total packets sent, total packets received, total SYN packets, total FIN packets, and many more.

Given how large PCAP files can grow, one might imagine that attempting to process 250 unique features for each packet sent across the wire could quickly moves this type of machine learning classification into the realm of Big Data and Super-Computers.

## Evaluating Machine Learning Model Effectiveness

Once a machine learning model has been trained on a given dataset, it must be evaluated for effectiveness. There are several terms, used among the surveyed publications, that describe model efficacy and should be understood.

### The four classification outcomes [26]

**True Positive (TP)** - Machine learning algorithm correctly predicted the positive class. E.g. the algorithm correctly identifies a known-benign network connection as benign.

**False Positive (FP)** - Machine learning algorithm incorrectly predicted the positive class. E.g. the algorithm incorrectly identifies a known-malicious network connection as benign.

**True Negative (TN)** - Machine learning algorithm correctly predicted the negative class. E.g. the algorithm correctly identifies a known-malicious network connection as malicious.

**False Negative (FN)** - Machine learning algorithm incorrectly predicted the negative class. E.g. the algorithm incorrectly identifies a known-benign network connection as malicious.

### Confusion Matrix [26]

**Confusion Matrix** - A table of correct and incorrect machine learning predictions which forms the basis of several metrics formulas below. In Table 3.1, if the positive class is "Benign" and the negative class is "Malicious" then the confusion matrix shows five True-Positive, two False-Positive, three False-

| | | Actual Class | |
|---|---|---|---|
| | | Benign | Malicious |
| Predicted Class | Benign | **5** | 2 |
| | Malicious | 3 | **4** |

Table 3.1: Confusion Matrix Example

Negative, and four True-Negative predictions.

## Efficacy Metrics [26]

Using the numbers from Table 3.1 several useful metrics can be calculated.

**Precision** - Ratio of accurate positive predictions to all positive predictions

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

**Recall** - Ratio of accurate positive predictions to all positive class elements

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

**Accuracy** - Ratio of accurate predictions to all predictions

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.5}$$

**F1-score** - Combines Precision and Recall into one score

$$F1 = \frac{Precision * Recall}{Precision + Recall} \tag{3.6}$$

# Chapter 4: Encryption Effects on Machine Learning Features

Attempting to classify encrypted network traffic brings up several questions about encryption's effect on packets, payloads, and flows. Does encryption affect inter-packet arrival times? Are packet headers affected or obfuscated? How does payload size change when encrypted? How does packet entropy change when encrypted? The answers to these and many other questions have a decided effect on the machine learning techniques that may be applied for traffic classification.

Overall, the effects of encryption depend on the manner in which communications are encrypted. Relative to the Open System Interconnect (OSI) network model, encryption may be implemented in the lower layers, as with Wi-Fi Protected access (WPA) on wireless networks, or may be implemented in the higher layers, as with SSH and SSL. In general, the lower on the OSI model encryption is introduced the more effect it has on available machine learning features.

Specifics of the encryption algorithm used also affect how traffic is altered. For instance the Advanced Encryption Standard (AES) algorithm operates only on 128-bit blocks of data [23]. If the clear text data is smaller than the AES block size, the data is padded up to the required size before processing. This means the original payload is significantly altered before being enciphered, answering one of the previous questions.

## Effects of SSL encryption

Bernaille et al. [7] studied classification of SSL encrypted connections and came to the conclusion that "SSL does not modify significantly the the number of packets, their size, and their inter-arrival time." Figure 4.1 shows generated packets, from zero to 100 bytes, and the resulting encrypted packet sizes using three common encryption algorithms. As noted above, the AES algorithm output appears as a stair-step function due to the fact that AES is a block-cypher system which pads the input up to the next block size multiple before encoding. The two Rivest Cypher 4 (RC4) based algorithms (RC4 and RCA) are stream cyphers and require no padding so they show a more consistent relationship to the original input data. According to their research, that relationship remains a consistent 25 byte increase for RC4 and 21 byte increase for RCA even as the original packet size grows large.

Understanding this relationship between input and encrypted output means a simple compensation step of subtracting the correct number from the encrypted packet size returns the original packet length. Even with AES, a good approximation can be had by subtracting the mean of the stair-step range. All that

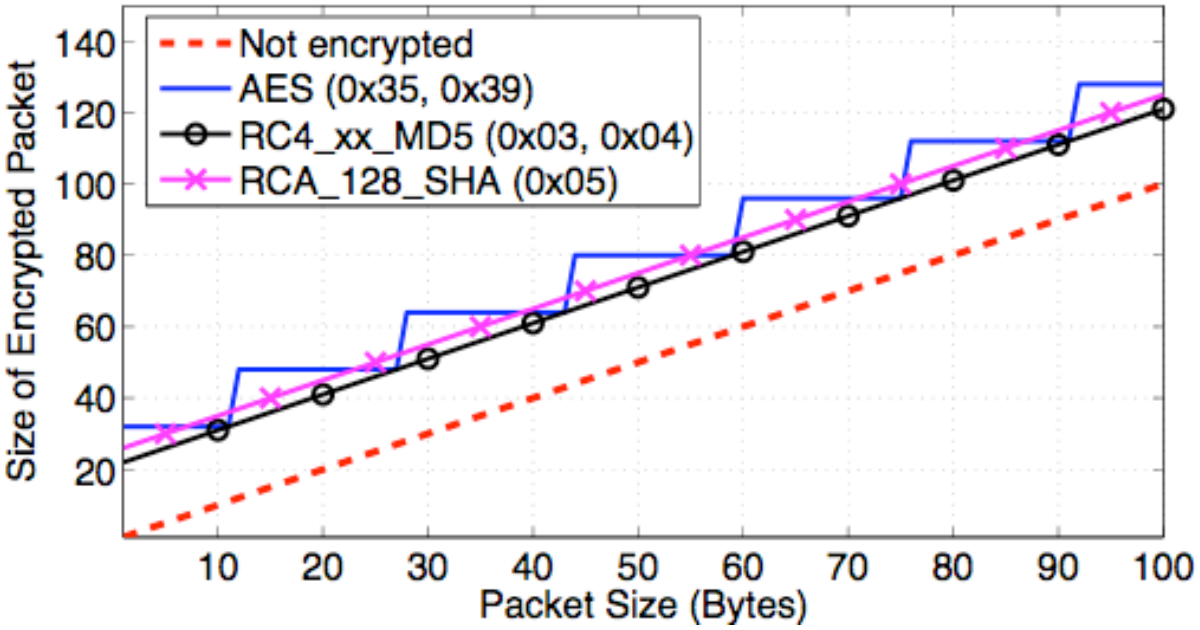



Figure 4.1: Bernaille et al. SSL Encryption Effects [7]

must be known is which algorithm was used to encrypt the connection for the appropriate adjustments to be made. Luckily, with SSL connections, the encryption algorithm is negotiated in plain-text. If a packet capture includes the initial SSL negotiation, the agreed upon algorithm can be compensated for appropriately.

## Combined Encryption and Compression

Podtburg [34] simulated a network carrying Pretty Good Privacy (PGP) encrypted payloads. The study provides more insight on encrypted payload size effects by noting a particular Rivest-Shamir-Adleman (RSA) encryption algorithm "increased some files by nearly one hundred percent." Cipher-text output which is larger than the plain-text input will likely increase the number of packets required to send a message and, potentially, increase the size of each packet used for transport.

However, increased payload size was not the only relevant side-effect. Podtburg also observed potential payload size decreases due to the fact that PGP compresses a message as well as encrypts it. The result of combining compression and encryption is a contention between encryption increasing payload size and compression decreasing it, each having a varying degree of influence depending on algorithms used.

This same paradigm is likely to play a role in TLS encrypted web traffic as well. RFC 3749 [19] defines compression methods as an integral part of the TLS specification. TLS encrypted channels may be both encrypted and compressed, in much the same way as PGP messages are, making it difficult to determine

if packet payload size or flow length has been altered by one or both of these competing influences.

# Chapter 5: Encrypted Vs. Compressed Traffic

If systems like PGP and TLS both compress and encrypt data, a method to detect and differentiate each must be found in order for encrypted traffic classification to be most effective. Many techniques are available to this end. Some are only appropriate for specific use cases while others generalize more readily.

This chapter reviews the most suitable methods for aiding in traffic classification.

## Entropy Measurements

As White, et al. stated [44]: "Our first instinct, when faced with the problem of measuring the uniformity of a set of samples, was to use entropy-based measures." On the surface this makes perfect sense as one would expect encrypted payloads to exhibit far greater entropy than un-encrypted traffic. While this is generally true of encryption, it ignores the fact that binary files (e.g. PNG images, MP3 files, ELF programs, etc.) and compressed files (e.g. ZIP archives, GZIP compressed files, etc.) also show increased payload entropy. White, et al. determined that "... accurate entropy testing requires significantly more samples than is practical in our setting, and is less efficient than more direct methods based on the samples themselves rather than on a derived statistic such as entropy."

## Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (K-S) test computes the difference between two Cumulative Distribution Functions (CDFs) showing the amount of "disagreement" between them. Statistically, one would expect encrypted or compressed data to be randomized to the degree that a computed CDF across all payload bytes (summing counts of values 1, 2, 3, ..., 255 in the data) would result in a nearly uniform distribution. This is especially true of encrypted data where enciphered outputs should appear as pseudo-random noise with a nearly uniform distribution. Armed with this uniformity assumption, a K-S test can be used to compare network traffic to a known uniform distribution, showing how far from uniform the payload data CDF lies and, consequently, how likely it is that the observed data is encrypted or compressed.

## Chi-Square Technique

An alternative to the K-S test, Chi-Square also compares two probability distributions. Similarity or difference is measured as a confidence percentage, estimating how often the CDFs diverge by chance.

For encrypted payloads, the Chi-Square test is used in the same manner as the K-S test. Captured payload values are mapped to a CDF which is then compared to the uniform distribution. Lower similarity

scores indicate a lack of encryption or compression while higher scores indicate the opposite.

## Likelihood Ratio Test

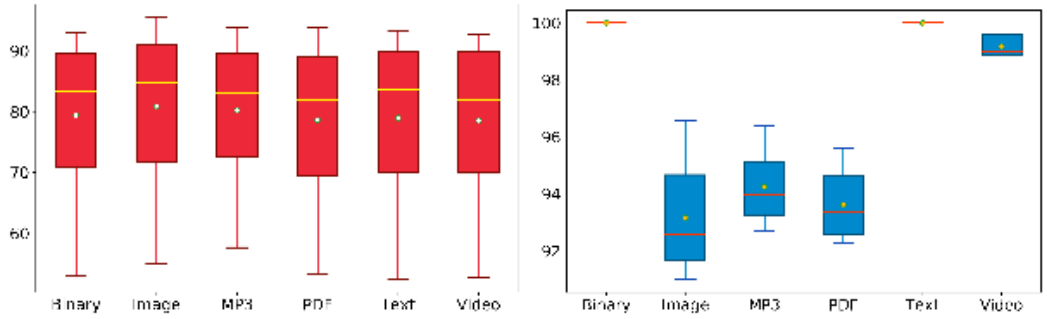
The Likelihood Ratio Test compares two CDF models by contrasting at the maximization of likelihood on one distribution against a bounded constraint likelihood measurement from the other. When the ratio of these two likelihood measurements is low, it indicates that the two models are very similar and will only vary by less than the sample error.

Again, this test is a way to compare the CDF generated from network traffic payloads to determine if the distribution is statistically similar to the uniform distribution. As with the K-S and Chi-Square methods, similarity indicates the presence of encryption or compression.

## Technique Effectiveness

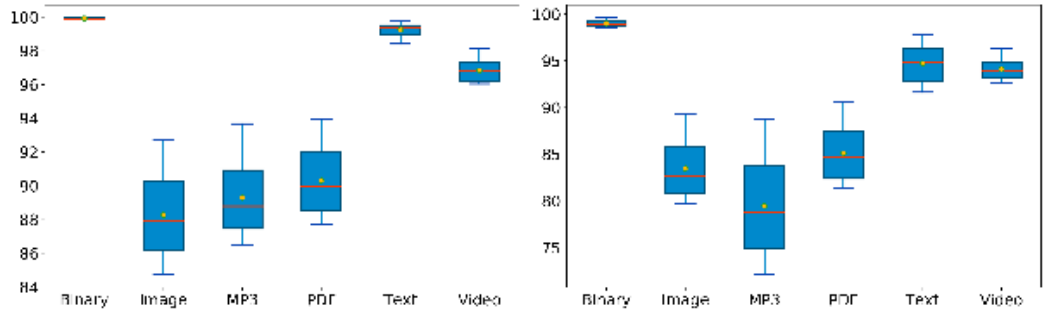
Casino, et al. coded a combination of techniques into their open-source project known as HEDGE (High Entropy DistinGuishEr) [15]. Figure 5.1 shows accuracy measurements for detecting encryption or compression on a range of file sizes and types. With varying degrees of accuracy, their system is able to detect the difference between compression and encryption which is extremely useful in deciding how to apply machine learning techniques to the network traffic classification problem.

White, et al. [44] performed similar test on a variety of techniques including Chi-Square, a discrete variant of the K-S test, Likelihood Ratio, and entropy based measurements. The goal of this work was not to detect encryption and compression separately, only to determine if the given data is "opaque", meaning obfuscation by either encryption or compression. Figure 5.2 compares True-Positive and False-Positive rates for a variety of techniques. Data points clustered in the top-left corner are more desirable as those techniques produced more True-Positive results with relatively low False-Positive errors.



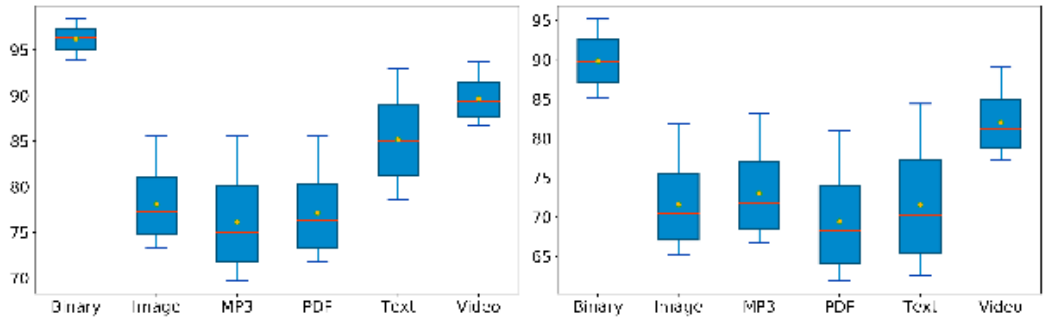

(a) % of accuracy for encrypted detection in 64KB files.

(b) % of accuracy for compressed detection in 64KB files.



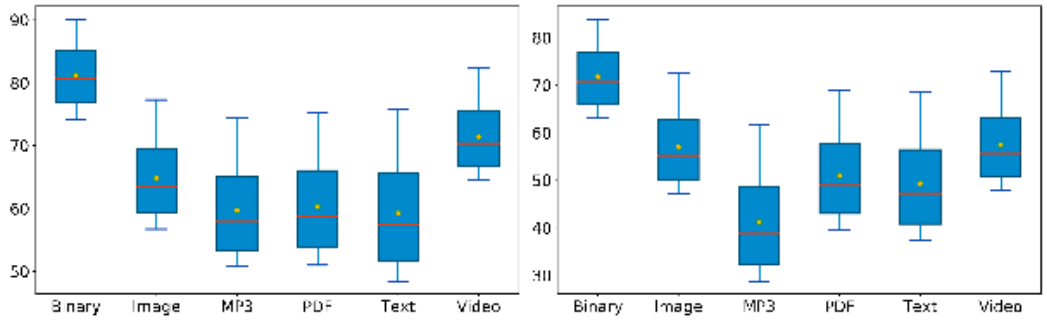

(c) % of accuracy for compressed detection in 32KB files.

(d) % of accuracy for compressed detection in 16KB files.




(e) % of accuracy for compressed detection in 8KB files.

(f) % of accuracy for compressed detection in 4KB files.




(g) % of accuracy for compressed detection in 2KB files.

(h) % of accuracy for compressed detection in 1KB files.

Figure 5.1: Casino et al. HEDGE system accuracy [15]

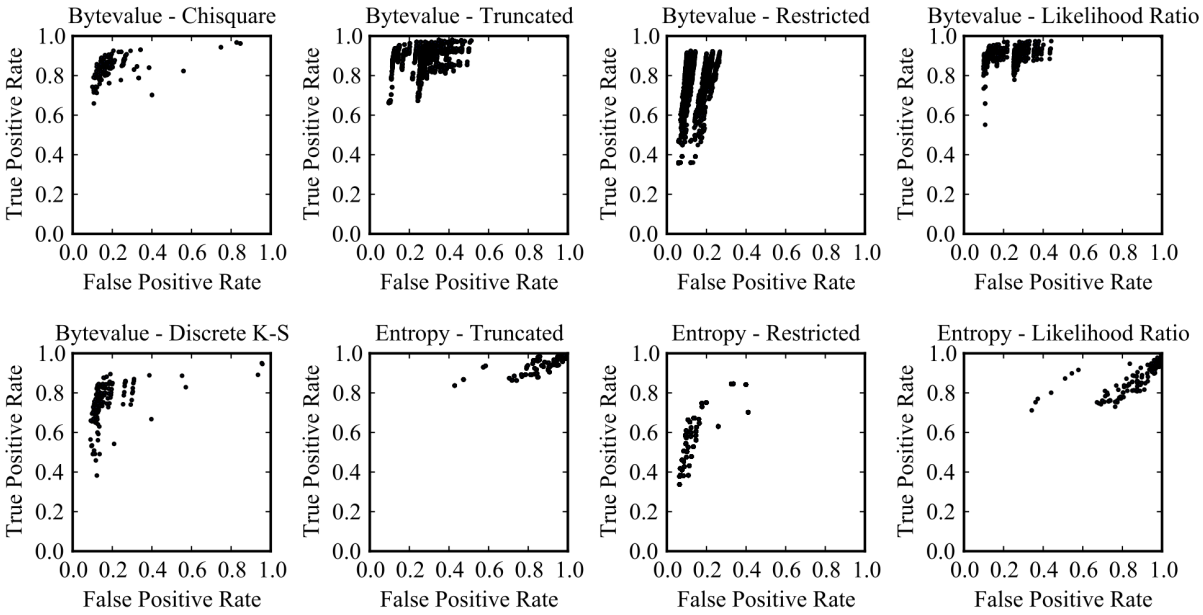



Figure 5.2: White et al. positive rate comparison [44]

# Chapter 6: Payload Analysis Techniques

## Supervised Payload Analysis Techniques

### Payload Entropy Approach

Khakpour and Liu [25] use an information theory approach to separate traffic flows into three categories: text, encrypted, or binary based on payload entropy measurements. Categorization is based on the observation that text payloads have minimal entropy, binary payload entropy is somewhat higher, and encrypted payloads have maximal entropy. Entropy, in this case, is measured as a set of vectors which are then used as a features in machine learning classification using both decision tree and SVM algorithms. A comparison of the two algorithms determined that SVM provides higher accuracy with less false positives.

Since this method relies on accurate entropy measurements, an extra step was added to remove application layer headers for well-known applications (HTTP, SMTP, IMAP and POP) using signature based detection. Without this extra step the repeated header information introduced bias and resulted in more prevalent misclassification of packets.

## Unsupervised Payload Analysis Techniques

### Natural Language Processing Approach

Zhang, et al. [47] treat binary payload data like a Natural Language Processing (NLP) problem by applying an unsupervised algorithm known as ProWord. Normally encrypted payloads would thwart this type of word-based pattern analysis, however, with their modified version of the Voting Experts [11] algorithm the clear-text negotiation phases for many encryption schemes mean this technique remains relevant.

Applying this technique, each application protocol is assumed to be its own type of language with unique characters, words, and grammar. The transmitted byte sequence payloads are scanned with a sliding window approach, leveraging the voting experts method of entropy based word boundary detection. Of course, when parsing through binary payload data these words are not the words we think of in spoken language. Instead the algorithm is detecting byte stream patterns that are common among application protocols.

Candidate feature word sets are assembled in a probability of occurrence tree structure. To reduce

memory usage, a pruning process removes the lowest frequency nodes which are unlikely to represent valid feature words. The result is a list of most probable feature words for a given application protocol which can then be used for off-line protocol detection in newly observed streams.

## Semi-Supervised Payload Analysis Techniques

### Spectral Clustering Approach

Liu, et al. [29] combine payload based and flow based features to increase overall classification performance. The authors evaluate a number of potential feature sets determining that ten features, most of which are related to transmitted data sizes, provide the best correlation to the application types of interest (HTTP, FTP, Game, HTTPS, Skype, Emule, PPlive, Xunlei).

To best utilize available payload data, the authors studied the effects of limiting payload data analysis to the first n-bytes. They determined that the first 40 payload bytes provided sufficient accuracy while not overburdening the machine learning algorithm with needless data.

Since most publicly available traffic capture datasets have payload content removed, the authors opt for capturing their own training data on a campus network. Three separate network traces are performed, each containing several thousand samples for each of the eight application types.

For the semi-supervised classification setup, the authors evaluated labeled training set size on overall accuracy. For each of the three network traces a DBSCAN clustering algorithm is seeded with 1000, 4000, 8000, and 10000 labeled samples. As the number of labeled samples increased so did overall accuracy, typically increasing nearly 10% between 1000 and 10000 labeled data points but only about 2% between 8000 and 10000. The authors opt to use 8000 labeled flows in a trade-off between accuracy and manual labor.

Overall system performance is very respectable with over 90% accuracy and at least 80% precision for each of the application type classifications when utilizing the combined payload and flow feature sets.

## Other Payload Analysis Techniques

### Supported Cipher Profiling

Husák, Čermák, Jirsík, and Čeleda take a very straight-forward approach to SSL and TLS fingerprinting by noting different clients support only a subset of the available cipher suites [22]. This, combined with the fact that encrypted connection parameters are negotiated in clear text, means it is possible to watch the SSL or TLS handshake process, including the transmitted list of supported ciphers, to infer

the user-agent of the client.

Both manually initiated connections from known clients (curl, wget, browsers, etc.) as well as a one-hour capture of campus traffic are analyzed to build a dictionary mapping supported cipher suites lists to user-agents. In total over 85 million HTTPS connections are processed, covering desktop, mobile, and unknown client systems, to achieve a 95.4% client type retrieval rate.

# Chapter 7: Packet Analysis Techniques

## Supervised Packet Analysis Techniques

### Supervised Method Comparison

Alshammari and Zincir-Heywood [2] ask if it is possible to identify packets from two encrypted applications (SSH and Skype) using a limited set of packet header information not including TCP/UDP port number or IP address. Both C4.5 and AdaBoost supervised machine learning algorithms are applied to several private and publicly available data sources including university network traffic, the DARPA99 dataset, and others. A total of 39 features are selected for model training and fifty runs of each model are evaluated using different training parameters for each.

Overall C4.5 shows a stronger performance, in both high detection rate and low false positives, for identifying both SSH and Skype encrypted traffic but did not perform as well as a genetic programming algorithm which achieved a 98% detection rate with 0.8% false positives.

### Hidden Markov Model Approach

Akhtar and Kamran [1] attempt to keep traffic classification as light-weight as possible by leveraging HMMs on four simple statistical features: mean Inter-Packet Arrival Time (IPAT), standard deviation of IPAT, mean packet size, and standard deviation of packet size. They focus on identifying four different applications: YouTube, Email, Skype, and an on-line game.

HMMs are trained from a custom generated dataset on dedicated machines. Four statical features were calculated for each of the four applications resulting in sixteen training features. Since HMM models have user selectable parameters, the authors use an iterative approach to parameter selection, continually testing and refining parameters with each iteration until an optimized set is found.

Ultimately the packet size features provides better results than IPAT, reaching as high as 92% traffic identification as opposed to IPAT which reached a respectable 87%.

### Multilevel Classifier Approach

Zhao, et al. [48] propose a multilevel classifier able to identify target mobile application traffic while simultaneously excluding unknown application traffic. This is accomplished with a three-stage architecture which combines flow, packet, and payload features, many of which are limited to the initial set of packets in the flow. The first stage is implemented as a coarse-grained binary classifier meant to

divide flows into two groups: target or other. This allows for the removal of as much irrelevant data as possible while passing relevant traffic along to the next stage. The second stage implements a fine-grained classifier designed to distinguish between the target applications. This stage is implemented as an "N+1" classifier returning which one of N applications most likely generated the captured traffic or an "ambiguous" label for for unidentified packets. Lastly, the third stage focuses on removing any false positives generated by the previous two layers and aims to eliminate any misclassified targets. This stage is implemented as a series of application pair checks, comparing the probability of matching applications in a one versus one fashion. With both the second and third stages focused on classifying application traffic from different perspectives, the authors believe the "strict prediction criteria of the third layer will enable the classifier to eliminate non-target instances effectively."

Overall the reported classification precision for encrypted application traffic was consistently in the high 90% range, however, the authors note that the second stage classifier excluded many true positive classifications which resulted in a low recall score for some applications.

### Random Forest Approach

Wang, et al. also focus on inferring smartphone application usage using side-channel information leakage techniques [43]. The authors train a Random Forest classifier using 20 packet level features (10 from transmitted packets and 10 from received packets), including some basic statistical inference such as packet inter-arrival times, average sizes, standard deviation of size, etc. Applications of interest are tested on both Apple iOS and Android platforms and are limited to detecting eight activity categories (browsing, gaming, multimedia, on-line chat, social network, dating, financial, and medical) across thirteen applications (Chrome browser, Boom Beach, YouTube, Songza, Facebook Messenger, Tencent QQ, Snapchat, Facebook, Twitter, Tinder, Mint, CDC News, and Medscape).

The authors generate a training dataset for each application by monitoring up to 300 seconds of application usage on a wireless network. Training sets are used to build a random forest classifier where both the overall classification accuracy and Out-of-Bag (validation) performance range between 69% (Boom Beach Game) to the high 90% range (YouTube, FaceBook Messenger, and Tencent QQ) with a very respectable overall accuracy in the mid 90% area.

### Hybrid Supervised Algorithms

Wright, Monrose, and Masson [45] take a hybrid approach for analyzing one or more application protocols carried in an encrypted tunnel. This effort starts with a k-NN supervised classifier on three packet features: timing, size, and direction. The classifier is trained on a random day's data from a

collected dataset leveraging TCP or UDP port number to determine ground truth. Protocols of interest are limited to Hyper-Text Transport Protocol (HTTP), HTTPS, Simple Mail Transfer Protocol (SMTP), SSH, AOL Instant Messenger (AIM), File Transfer Protocol (FTP) and Telnet.

When a given encrypted tunnel carries more than one application protocol type, the authors employ HMMs to determine the number and type of encapsulated protocols within the flow. An HMM is trained for each of the relevant protocols and the model that best matches the packet sequence is taken as the best matching protocol assumed to be contained within the encrypted tunnel.

## Unsupervised Packet Analysis Techniques

### Gaussian Mixture Modeling

Hsiao and Ibanez [21] look at unsupervised techniques for improved Network Intrusion Detection Systems (NIDS). Two techniques are compared, Gaussian Mixture Modeling (GMM) and a purely probability based approach known as Packet Header Anomaly Detection (PHAD). Packets are analyzed based solely on header content and the probability of header values deviating from expected norms. The GMM technique determines how unlikely it is for a packet, with its observed features, to be seen and flags those packets with the least likely probability as anomalous. PHAD takes this one step further generating a score for each field in an observed packet and uses a probability based calculation to determine if the packet is anomalous. This calculation includes a time based parameter which assumes anomalous events will occur in bursts rather than individually and sporadically.

The techniques are tested against the 1999 DARPA Intrusion Detection Dataset where the PHAD system shows a 35 times performance increase over the GMM technique. This is attributed to the non-stationary nature of the time parameter included with the PHAD technique and supports the hypothesis that anomalous events happen in bursts.

## Semi-Supervised Packet Analysis Techniques

### Semi-Supervised Method Comparison

Bernaille, Teixeira, and Salamatian [28] focus on fingerprinting SSL encrypted applications with the least number of packets possible. Three clustering methods (K-Means, GMM, and Spectral Clustering on HMMs) are evaluated against two sets of network captures, one with only packet headers and the other with full payloads.

Application fingerprinting is limited to payload size data for the first four packets of application layer

data after the SSL tunnel is established. This method better supports real-time application detection as the classification data is almost immediately available rather than waiting for calculated metrics such as connection length or mean throughput that are available only after the connection has completed.

Where clustering algorithms might assign multiple applications to the same cluster an additional disambiguation heuristic is added in the form of an Internet Assigned Numbers Authority (IANA) well-known port number. In this way, the technique falls back on the connection's TCP or UDP port number where the first four packets do not clearly point to a single application's behavior.

Of the three clustering algorithms evaluated, the combination of GMM and the port based heuristic correctly classified over 98% of known application behaviors, providing the best trade-off in model complexity and accuracy.

### Three-Step Classifier

Bernaille and Teixeira [7] expand on their previous work in [28] to focus on detecting SSL encrypted traffic and the application protocols within them. Classification is accomplished with a three-step process consisting of SSL connection recognition, detecting the first packets containing application data, and fingerprinting of encrypted application messages.

The SSL detection step relies on observing an SSL connection setup, which is done in clear-text. A simple set of heuristic rules, based on bit and byte values in the "client hello" and "server hello" packets, allow for distinguishing between SSLv1, SSLv2, SSLv3, and TLS encrypted connections.

After an encrypted connection is identified, the SSL packet stream is monitored until a packet with content type 23 is seen. This indicates the SSL negotiation is complete and application layer messages have begun.

Finally, the application protocol is fingerprinted in the same manner as described in [28], based on the size of the subsequent encrypted application layer messages. Since the clustering algorithm in [28] is initialized with pre-labeled cluster members, effectively "training" the clustering algorithm, this technique is categorized as semi-supervised rather than unsupervised.

## Other Packet Analysis Techniques

### Packet Header Profiling

Mahoney and Chan [31] propose an anomaly detection system, known as PHAD, which relies only on packet header information to detect anomalous network traffic. The system learns the typical range of values for 33 packet header fields from Ethernet, IP, TCP, UDP, and Internet Control Message

Protocol (ICMP) protocols. Armed with this set of observed packet header values, PHAD is able to monitor network traffic looking for any values that fall outside the normal ranges. The system assumes that anomalous events tend to occur in bursts and, using an anomaly probability function, scores each packet's anomalous fields and the likely hood of observing that anomalous value on the network.

The authors train the PHAD system using seven days of attack-free network traces from the publicly available DARPA 1999 off-line IDS data set. With training complete, the system was evaluated for the ability to detect exploits and attacks in the DARPA test set. Of 201 recorded attacks PHAD detected 67 of them, with a false positive rate of 10 per day. Many of these were detected by anomalous Time To Live (TTL) field values.

# Chapter 8: Flow Analysis Techniques

## Supervised Flow Analysis Techniques

### Data Omnia Approach

Anderson and McGrew [4] focus on detecting encrypted malware traffic using many available indicators in a "data omnia" approach. Rather than limiting to a single type of machine learning feature, this work combines Domain Name System (DNS) query context, client advertised TLS cipher suites, packet metadata (packet inter-arrival times and payload size) and HTTP header information to increase overall classification performance. Supervised machine learning techniques, including logistic regression and SVMs are compared with the SVM showing "no statistical improvement" to offset its increased computational complexity.

Training datasets for both malware and benign TLS connections were used to train machine learning models. Malware data was collected over a four month period in 2016, from a commercial user submission and malware testing site, resulting in over 21 thousand TLS flows from suspected malicious contributions. Benign data was collected from a large enterprise DMZ network over a five day period resulting in over one million recorded TLS connections. These flows were then pared back to those with available DNS query data and HTTP contexts resulting in 13 thousand malicious and nearly 43 thousand benign connections.

Applying logistic regression to the collected array of machine learning features results in a reported 99.993% total accuracy rate for malicious traffic detection with a very nearly 0% false detection rate. Even with reduced feature sets the total accuracy remained above 95%, however, false detection rates rise significantly.

### Random Forest Approach

Taylor, et al. [42] turn their attention to application detection on Android smartphone devices. This poses some extra challenges over desktop or server application fingerprinting as the smartphone ecosystem relies more heavily on smaller Application Programming Interface (API) calls using Extensible Markup Language (XML) or JavaScript Object Notation (JSON) formatted requests. These mobile platforms also consolidate some extra functionality, such as advertising retrieval and caching as Operating System (OS) level functions. This means some network traffic, that would be indicative of application use, is not directly attributable to the application process and DNS queries are less useful when pulling from distributed Content Delivery Networks (CDNs).

The authors apply Random Forest learning to 54 statistical flow level features, including kurtosis, skew, variance, standard deviation, etc. for both incoming and outgoing packets. Due to the fact that some observed flows are not part of the training data, and would subsequently fall into an "unknown" sample classification, reinforcement learning is applied to detect and label ambiguous flows.

Together, the Random Forest and reinforcement learning combination achieve precision and accuracy scores approaching the mid 90% area which seems to be on par with other surveyed publications.

### Supervised Method Comparison

Lashkari, et al. [20] analyze seven application types in two distinct scenarios: encrypted traffic flows encapsulated within a Virtual Private Network (VPN) link and encrypted traffic flows not encapsulated within a VPN link. The goal is to identify the seven relevant application types (web browsing, e-mail, chat, video streaming, file transfer, Voice over Internet Protocol (VoIP) calls, and Peer-To-Peer (P2P) network communications) using 23 flow based features including the mean, min, max, and standard deviation of forward, backward, and bi-directional IPAT as well as overall flow duration, active time, idle time, bytes per second, and packets per second.

Both the C4.5 decision tree and k-NN classifiers are applied to 28GB of custom generated traffic logs to determine if application behaviors can be identified in both scenarios (VPN encapsulated and non-VPN encapsulated) with similar results from each algorithm. With the first scenario the authors show that, using time-based flow features alone, they are able to distinguish between VPN and non-VPN encapsulated traffic with precision scores in the high 80% range. The second scenario shows both C4.5 and k-NN performed approximately equally in trying to identify application types within a mixed packet capture, achieving precision scores up to the mid 70% range.

### Convolutional Neural Network Approach

Rezaei and Liu [36] propose a multi-task CNN based approach where the system predicts three output classes: Bandwidth, Traffic Class, and Duration. These three predicted values are selected for their usefulness in Quality of Service (QoS) resource planning at the ISP level.

To accomplish this goal, the authors take advantage of CNNs ability to detect patterns in an input stream no matter where in the stream the pattern appears. A series of CNN filter and pool layers are constructed with a fully-connected output layer providing all three classifications as its output.

Two publicly available datasets are used to train the system: the QUIC Dataset from University of California at Davis and the ISCX VPN-nonVPN dataset from University of New Brunswick. A minimal set of time-series features are extracted from the training sets including packet length, inter-arrival time,

and direction for the first $k$ packets in each flow. Not all records in the training data are labeled with application information but the authors compensate for this by leveraging a correlation between the bandwidth and duration metrics to aid in training traffic classification.

Despite the limited number of time-series features and packets utilized from each flow, the multi-task CNN approach shows an improvement over comparable single-task machine learning efforts. Bandwidth and duration classification accuracy increased well over a percentage point and traffic classification boosted as much as 10% in many cases, despite the lack of labeled data.

## Unsupervised Flow Analysis Techniques

### Spectral Clustering Approach

Amoli [40] applies unsupervised flow analysis to real-time NIDS. This is accomplished with a two part system; one leverages the DBSCAN unsupervised technique to detect anomalous DDoS network traffic in real-time and the other tracks down centralized master nodes controlling their bot-net over HTTP or the Internet Relay Chat (IRC) network. By dividing the workload into two systems the real-time anomaly detection can continue while botnet controller hunting takes place elsewhere.

The authors show that the DBSCAN unsupervised algorithm is particularly well suited to zero-day attack detection achieving a 100 percent detection rate with a false-positive rate of just over three percent and an overall precision of 97.83

## Semi-Supervised Flow Analysis Techniques

### Two-Layer Classifier

Erman, Mahanti, Arlitt, Cohen and Williamsen [14] propose a semi-supervised traffic classification technique for both off-line and real-time network flow analysis. The system consists of a learner component and a classifier component. The learner builds a mapping from network flow statistics to application type based on partially labeled training data and the k-Means (k-Means) clustering algorithm. The classifier then uses the semi-supervised learner model to classify new connection statistics as they become available.

Fully labeled training data can be extremely cumbersome and expensive to obtain. Conversely, unlabeled training data is relatively easy to capture. By leveraging a semi-supervised approach to the learner component this technique is able to build an accurate and updatable classifier model quickly.

The authors show that flow data alone is able to achieve accuracy scores as high as 98% and, since this technique uses only network flow statistics, is unhindered by encrypted payloads.

# Other Flow Analysis Techniques

## Pure Statistical Approach

Munit, et al. [32] combine a purely statistical analysis approach, with port based heuristics, to classify flows as VoIP or non-VoIP traffic. VoIP applications of interested are limited to Google Hangouts, Facebook VoIP, Yahoo Messenger, Skype, and Viber.

Many VoIP protocols/applications utilize well-known TCP port ranges including: SIP (5060-5070), H.323 (1718-1720), MGCP/Megaco/H.248 (2427, 2944), Skype Login (80, 443), and Skype Authentication (33033). Traffic on these ports provides a good indication of VoIP application usage, however, encrypted tunnels obfuscate this simple heuristic requiring more sophisticated detection techniques.

The authors compare statistical properties for VoIP and non-VoIP application flows from several public and private traffic traces. Patterns for the above VoIP protocols are contrasted against non-VoIP flows including BitTorrent, antivirus updates, streaming television, streaming audio, FTP, text chat applications, web browsing, and e-mail services. Specifically the authors calculate mean packet size, standard deviation of packet size, maximum inter-packet delay, mean inter-packet delay, standard deviation of inter-packet delay, overall data rate, and packet entropy for flows captured in the traffic traces. Traffic comparisons lead to several observations which the authors then encode as rule-sets for flow classification:

1. $PacketRate > 15 packets/sec$
2. $DataRate < 0.5 Mb/s$
3. $50 \leq Mean(PacketSize) \leq 210 bytes$
4. $0 \leq S.D(PacketSize) \leq 75 bytes$
5. $Mean(PacketSize) \geq S.D(packetsize)$
6. $Mean(PacketSize) \geq 3.0$
7. $0 < MaxDiffTime \leq 0.8 seconds$
8. $0 < Mean(DiffTime) \leq .09 seconds$
9. $0 < S.D(DiffTime) \leq 0.25 seconds$

These rules form a decision tree where all of the first six rules must be true followed by at least two of the last three for a flow to be labeled as VoIP. Rules are applied to five second capture increments. If a definitive label cannot be determined, another five seconds of flow data are evaluated. After three ambiguous evaluation intervals a flow is assumed to be non-VoIP.

The authors report an extraordinary 100% accuracy and 0% false-positive rate for all VoIP applications of interest in their own captured traffic traces, however, only flows greater than five seconds can be classified which meant many VoIP flows were simply ignored during evaluation.

# Chapter 9: Conclusions

Machine learning and statistical analysis have brought network traffic fingerprinting a long way since the early days of port-based traffic identification. The twenty publications reviewed in this Thesis are only a small sampling of available works and were selected to cover as much of the Khalife, et al. taxonomy [26] and research history as possible.

These surveyed publications run the gamut of supervised, unsupervised, semi-supervised, and basic statistical techniques applied to payload, packet, and flow based analysis. They are able to detect specific application protocols, application types, software, or even anomalous behaviors, achieving extremely accurate results in their target contexts.

## Successes despite encryption

Of the three analysis types (packet, payload, and flow), encrypted communications has the most adverse effects on payload analysis techniques. Encrypted payload data becomes completely obscured, confounding traditional DPI analysis techniques. Despite these hindrances, several research efforts show promising results. Khakpour and Liu [25] are still able to leverage SVM for three-class classification (Text, Encrypted, or Binary Traffic) and Zhand, et al. [47] are able to side-step payload obfuscation by applying NLP techniques. Others take advantage of the fact that most encryption systems rely on an unencrypted negotiation and setup phase which allows for early traffic classification before the stream becomes fully opaque.

Packet analysis techniques suffer less hindrances, than payload techniques do, when encryption is introduced. Overall packet size metrics tend to increase as encryption algorithms pad enciphered bytes up to a pre-determined block size multiple before transmission. Despite these changes, Alshammari and Zincir-Heywood [2] are able to differentiate encrypted SSH and Skype connections using a limited set of packet header information while Akhtar and Kamran [1] show promising results by observing several packet size based statistics, such as mean and standard deviation, to detect four application types. Hsiao and Ibanez [21] describe a NIDS able to detect malicious traffic based entirely on packet header probabilities. Lastly, Bernaille, Teixeira, and Salamatian [28] profile SSL encrypted applications using payload size data from the least number of packets possible.

Finally, flow based analysis methods may see added latency between packet inter-arrival times (as the act of encrypting takes extra CPU cycles to compute the cypher-text) but, overall, the effects on analysis are minimal as compared to packet or payload analysis. Even with the latency changes, Taylor, et al. [42] show a method to identify smartphone application usage by inspecting 54 statistical flow-based

features, including kurtosis, skew, variance, and standard deviation, while Lashkari, et al. [20] are able to profile VPN encapsulated traffic leveraging 23 flow-based features including mean, max, min, and standard deviation for packet inter-arrival times.

## Future work ideas

Despite many successes, over a large number of years, current research still mainly focuses on applying various statistical techniques and measuring detection accuracy. Some work has been done to combine techniques into multi-layer classifiers, leveraging ML algorithms assembled in series to obtain better scores for a specific use-case and desired output, but there seems to be little focus on generalizing these capabilities into a comprehensive system. A few publications, such as Mahoney and Chan [31] or Casino, et al. [15], describe how such a system might be designed or provide example scripts, leaving full implementations for future efforts.

The field of Cybersecurity threat intelligence would also benefit greatly from a unified traffic fingerprinting framework based on these reviewed techniques. In much the same way that file hashes are computed, shared, and used to detect malicious binary files hiding in computer system, an equivalent network traffic fingerprint might be generated and distributed to help detect known harmful actors hiding in network traffic.

## Summary Matrix of Reviewed Works

Velan, et al. [41] have created a table structure to summarize works analyzed using the Khalife, et al. taxonomy [26]. This section borrows that table structure listing taxonomy attributes for each surveyed publication.

| Reference | Pub. Year | Input | | | | | Num. Features | Feature Select | Technique | | | | | | | Method(s) | Output | Enc. Prot. Ident. | Real-Time Ident. | Data Set | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Traf. Payload | Traf. Properties | | | | | | Payload Insp. | Graphlets | Stats. | | Machine | | | | | | | Public | Private | Real | Artificial | Ground Truth |
| | | | Host Com. | Host | Flow | Packet | | | | | Basic | Heuristic | Supervised | Un-Super. | Semi-Super | | | | | | | | | |
| [31] | 2002 | | | | | ✓ | 33 | | | | ✓ | | | | | Basic Statistical | P → A | | | ✓ | | ✓ | | known |
| [28] | 2006 | | | | | ✓ | 4 | | | | | | | | ✓ | HMM, GMM, Spectral Clustering | P → AP | | ✓ | | ✓ | ✓ | | signature, known |
| [45] | 2006 | | | | ✓ | ✓ | 3 | | | | | | ✓ | | ✓ | HMM, k-NN | F → AP | ✓ | | | ✓ | ✓ | | port |
| [7] | 2007 | | | | | ✓ | 3 | | | | | | | | ✓ | HMM, GMM, Spectral Clustering | P → AP | ✓ | ✓ | | ✓ | ✓ | | signature, known |
| [14] | 2007 | | | | ✓ | | 25 | ✓ | | | | | | | ✓ | Clustering | F → AT | | ✓ | | ✓ | ✓ | | signature |
| [2] | 2011 | | | | ✓ | ✓ | 39 | ✓ | | | | | ✓ | | | C4.5, AdaBoost | P → AT | | | ✓ | ✓ | ✓ | | signature, port |
| [29] | 2012 | ✓ | | | ✓ | | 10 | ✓ | ✓ | | ✓ | | | | ✓ | DBSCAN | P → AT | | | | ✓ | ✓ | | known |
| [25] | 2013 | ✓ | | | | | 10 | ✓ | | | ✓ | | ✓ | | | Entropy, SVM, CART | F → AT,AP | ✓ | | | ✓ | ✓ | ✓ | known |
| [40] | 2013 | | | | ✓ | ✓ | 23 | | | | ✓ | | | ✓ | | Change Point Detect, Clustering | F → A | | ✓ | ✓ | ✓ | ✓ | ✓ | known |
| [47] | 2014 | ✓ | | | | | - | | ✓ | | | | | ✓ | | Voting Experts [11] | P → AP | | | | ✓ | ✓ | | known |
| [43] | 2015 | | | | | ✓ | 20 | ✓ | | | ✓ | | ✓ | | | Random Forest | P → AS | | | | ✓ | ✓ | | known |
| [22] | 2016 | ✓ | | | | | 1 | | ✓ | | | ✓ | | | | Heuristics | F → AS | ✓ | | | ✓ | ✓ | ✓ | known |
| [4] | 2016 | ✓ | | | ✓ | ✓ | >4 | | ✓ | | | ✓ | ✓ | | | Heuristic, logistic regression, SVM | F → A | ✓ | | | ✓ | ✓ | | signature, known |
| [32] | 2016 | | | | ✓ | ✓ | 9 | ✓ | | | ✓ | ✓ | | | | Basic Statistical | F → AT | | ✓ | ✓ | ✓ | ✓ | | known |
| [20] | 2016 | | | | ✓ | | 23 | | | | | | ✓ | | | C4.5, k-NN | F → AT | ✓ | | | ✓ | ✓ | | known |
| [42] | 2017 | | | | ✓ | ✓ | 54 | ✓ | | | | | ✓ | | | Random Forest, Reinforcement Learning | F → AS | | | | ✓ | ✓ | | known |
| [1] | 2017 | | | | ✓ | ✓ | 16 | | | | | | ✓ | | | HMM | P → AS | | | | ✓ | ✓ | | known |
| [21] | 2017 | | | | | ✓ | 33 | ✓ | | | ✓ | | | ✓ | | GMM, Basic Statistical | P → A | | | ✓ | | ✓ | | known |
| [48] | 2019 | ✓ | | | ✓ | ✓ | 37 | ✓ | ✓ | | | ✓ | | | ✓ | Random Forest, XGBoost | P → AS | | | | ✓ | ✓ | | known |
| [36] | 2019 | | | | ✓ | ✓ | 3 | | | | ✓ | | ✓ | | | Convolutional Network | P → AT | | ✓ | ✓ | | ✓ | | known |

Output column legend: A Anomaly; H Host; AP Application Protocol; F Flow; AS Application Software; FG Fine-Grained; AT Application Type; TP Traffic Payload

Table 9.1: Analysis methods matrix - Velan, et al. format [41]

# References


[1] Naveed Akhtar and Muhammad Kamran. Lightweight internet traffic classification based on packet level hidden markov models. *International Journal of Advanced Computer Science and Applications*, 8, 01 2017.

[2] Riyad Alshammari and A. Zincir-Heywood. Can encrypted traffic be identified without port numbers, ip addresses and payload inspection? *Computer Networks*, 55:1326–1350, 04 2011.

[3] Samaneh Aminikhanghahi and Diane J. Cook. Change-point detection without needing to detectchange-points? *Knowledge and Information Systems*, 51, 2016.

[4] Blake Anderson and David McGrew. Identifying encrypted malware traffic with contextual flow data. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, AISec '16, pages 35–46, New York, NY, USA, 2016. ACM.

[5] Denis Zuev Andrew Moore and Michael Crogan. Discriminators for use in flow-based classification. *Queen Mary Research Online*, 2005.

[6] Michael Jordan Andrew Ng and Yair Weiss. On spectral clustering : analysis and an algorithm. *Neural Information Processing Systems : NIPS 2001*, 2001.

[7] Laurent Bernaille and Renata Teixeira. Early recognition of encrypted applications. In Steve Uhlig, Konstantina Papagiannaki, and Olivier Bonaventure, editors, *Passive and Active Network Measurement*, pages 165–175, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[8] Chris Brook. What is deep packet inspection? how it works, use cases for dpi, and more. `https://digitalguardian.com/blog/what-deep-packet-inspection-how-it-works-use-cases-dpi-and-more`, 2018.

[9] T. Chen and C. Guestrin. Xgboost: a scalable tree boosting system. In *22nd ACM SIGKDD International Conference*, pages 785–794, 08 2016.

[10] J. Clement. Number of monthly active whatsapp users worldwide from april 2013 to december 2017. `https://www.statista.com/statistics/260819/number-of-monthly-active-whatsapp-users/`, 2019.

[11] Paul R. Cohen and Niall M. Adams. An algorithm for segmenting categorical time series into meaningful episodes. In *IDA*, 2001.

[12] James B. Comey. Going dark: Are technology, privacy, and public safety on a collision course? https://www.fbi.gov/news/speeches/going-dark-are-technology-privacy-and-public-safety-on-a-collision-course, 2014.

[13] Bradley Efron and Trevor Hastie. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. Cambridge Univeristy Press, 2016.

[14] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, Ira Cohen, and Carey Williamson. Offline/realtime traffic classification using semi-supervised learning. *Performance Evaluation*, 64:1194–1213, 10 2007.

[15] Kim-Kwang Raymond Choo Fran Casino and Constantinos Patsakis. Hedge: Efficient traffic classification of encrypted and compressed packets. *Transactions on Information Forensics and Security*, 2019.

[16] Sean Gallagher. The snowden legacy, part one: What's changed, really? https://arstechnica.com/tech-policy/2018/11/the-snowden-legacy-part-one-whats-changed-really/, 2018.

[17] Google. Google transparency report: Https encryption on the web. https://transparencyreport.google.com/https/overview?hl=en, 2019.

[18] Charles M Grinstead and J Laurie Snell. Grinstead and snell's introduction to probability. http://math.dartmouth.edu/%7Eprob/prob/prob.pdf, 2006.

[19] Network Working Group. Transport layer security protocol compression methods. https://tools.ietf.org/html/rfc3749, 2004.

[20] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Mamun, and Ali Ghorbani. Characterization of encrypted and vpn traffic using time-related features. In *3rd International Conference on Information Systems Security and Privacy*, 02 2016.

[21] Luke Hsiao and Stephen Ibanez. Machine learning for network intrusion detection. In *Preprint*, 2017.

[22] Martin Husák, Milan Čermák, Tomáš Jirsík, and Pavel Čeleda. Https traffic analysis and client identification using passive ssl/tls fingerprinting. *EURASIP Journal on Information Security*, 2016(1):6, Feb 2016.

[23] Steven Iveson. Ipsec bandwidth overhead using aes. https://packetpushers.net/ipsec-bandwidth-overhead-using-aes/, 2013.

[24] Andrej Karpathy. Deep reinforcement learning: Pong from pixels. `https://karpathy.github.io/2016/05/31/rl/`, 2016.

[25] Amir Khakpour and Alex Liu. An information-theoretical approach to high-speed flow nature identification. *Networking, IEEE/ACM Transactions on*, 21:1076–1089, 08 2013.

[26] Jawad Khalife, Amjad Hajjar, and JesÃºs DÃaz-Verdejo. A multilevel taxonomy and requirements for an optimal traffic-classification model. *International Journal of Network Management*, 24, 03 2014.

[27] E V Kotelnikov and V R Milov. Comparison of rule induction, decision trees and formal concept analysis approaches for classification. *Journal of Physics Conference Series*, 1015, 2018.

[28] Renata Teixeira Laurent Bernaille and Kave Salamation. Early application identification. In *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, CoNEXT '06. ACM, 2006.

[29] Huisheng Liu, Zhenxing Wang, and Yu Wang. Semi-supervised encrypted traffic classification using composite features set. *Journal of Networks*, 7, 08 2012.

[30] John Love. A brief history of malware – its evolution and impact. `https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/`, 2018.

[31] Matthew Mahoney and Philip Chan. Phad: Packet header anomaly detection for identifying hostile network traffic. 05 2002.

[32] Suneel Munir, Nadeem Majeed, Salaser Babu, Irfan Bari, Jackson Harry, Zahid Masood, and Masood. A joint port and statistical analysis based technique to detect encrypted voip traffic. *International Journal of Computer Science and information security*, 14:117–131, 03 2016.

[33] Andy Norton. Crypto-malware – a look at the latest malware threat. `https://www.lastline.com/blog/crypto-malware-a-look-at-the-latest-malware-threat`, 2018.

[34] Stephanie Podtburg. Simulated effects of pgp encryption on network performance. *Transactions on Information Forensics and Security*, 2001.

[35] Foster Provost and Tom Fawcett. *Data Science for Business: What you need to know about data mining and data-analytic thining.* by O'Reilly Media, 2013.

[36] Shahbaz Rezaei and Xin Liu. Multitask learning for network traffic classification. *CoRR*, abs/1906.05248, 2019.

[37] Taylor Smith. The new face of ddos attacks: Bigger, badder, & available as-a-service. `https://www.pivotpointsecurity.com/blog/new-ddos-as-a-service-attacks/`, 2018.

[38] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2015.

[39] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (12th printing)*. Springer-Verlag New York, 2017.

[40] Payam Vahdani Amoli. Unsupervised network intrusion detection systems for zero-day fast-spreading attacks and botnets. *International Journal of Digital Content Technology and its Applications*, 10:1–13, 03 2016.

[41] Petr Velan, Milan Cermak, Pavel Celeda, and Martin Drasar. A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management*, 25, 2015.

[42] Mauro Conti Vincent Taylor, Riccardo Spolaor and Ivan Martinovic. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security ( Volume: 13 , Issue: 1 , Jan. 2018 )*, 13:63–78, 04 2017.

[43] Qinglong Wang, Amir Yahyavi, Bettina Kemme, and Wenbo He. I know what you did on your smartphone: Inferring app usage over encrypted data traffic. *Institute of Electrical and Electronics Engineers Inc.*, pages 433–441, 12 2015.

[44] Andrew White, Srinivas Krishnan, Michael Bailey, Fabian Monrose, and Phillip Porras. Clear and present data: Opaque traffic and its security implications for the future. In *Network and Distributed System Security Symposium - NDSS Symposium 2013*, 02 2013.

[45] Charles V. Wright, Fabian Monrose, Gerald M. Masson, and Philip Chan. On inferring application protocol behaviors in encrypted network traffic, 2006.

[46] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[47] Zhuo Zhang, Zhibin Zhang, Patrick Lee, Yunjie Liu, and Gaogang Xie. Proword: An unsupervised approach to protocol feature word extraction. pages 1393–1401, 04 2014.

[48] Shuang Zhao, Shuhui Chen, Yipin Sun, Zhiping Cai, and Jinshu Su. Identifying known and unknown mobile application traffic using a multilevel classifier. *Security and Communication Networks*, 2019, 01 2019.

# Appendix A: Taxonomy Terms Defined

**Traffic Payload** - Network communication contents beyond Layer 4 in the the OSI Model. Ignores network routing, addressing, port segregation, etc. and focuses on the content of the transmitted message rather than the mechanics of transport and delivery.

**Traffic Properties** - Network communication contents below Layer 5 in the OSI Model. Focuses on information used in the mechanics of transport and delivery (network routing, addressing, port segregation, etc.) rather than the content of the transmitted message.

**Host Community** - Properties of networked devices participating in the same application or of the same class. This might be as broad as hosts serving web HTTP content, or as specific as hosts running the Apache Web Server.

**Host** - Properties of a single networked device's communications including address, opened ports, number of connection, etc.

**Flow** - Attributes associated with a series of transmitted messages that, together, comprise a communication channel. Typically, communication networks do not support message transport as a single transmission, but instead break a message into smaller segments which are re-assembled at the receiver. From these segments many properties may be directly or indirectly inferred, including supported transmit speed, supported receive speed, flow size, flow duration, error tolerance, etc.

**Packet** - The smallest unit of network message transmission. Typically includes addressing information (e.g. source and destination host addresses) as well as all or part of the message content.

**Payload Inspection** - Communication analysis which looks at the content of messages sent rather than information related to network transport and delivery mechanics. Due to the segmented nature of network transmissions, payload inspection may require re-assembly of a network flow in order to obtain the full content of the original message. Often referred to as Deep-Packet Inspection DPI.

**Graphlets** - An induced subgraph representing network host communication relationships as graph nodes and edges.

**Motifs** - A partial subgraph representing network host communication relationships as graph nodes and edges.

**Social Networks** - A graph representation of network host interactions which can detect host communities running the same application. These are often useful in detecting hosts participating in a P2P network.

**Basic Statistical** - Using fundamental statistical techniques to classify encrypted traffic. These basic techniques do not include any "Machine Learning" methods, but rather, focus on statistics building blocks

such as probability distributions (Poisson, Exponential, Bernoulli, etc.) used to model traffic attributes (packet size, inter-packet arrival times, payload size, etc.)

**Heuristics** - A statistical approximation or shortcut usually expressed as a set of rules. For example, assuming that port 22 traffic carries an SSH connection would be a simple, port-based heuristic classification.

**Profiles** - Formalized metrics generated from understood normal behaviors. For example, identifying TLS negotiation payloads by their typical size, direction, and placement at the beginning of a flow.

**Supervised Machine Learning** - Statistically based traffic modeling where the model is built with a combination of network traffic and both "correct" and "incorrect" labels. Using these pre-determined labels, the model is continually refined to provide the best possible predictions on available training data, while keeping the ability to generalize and provide accurate predictions on, as-yet, unavailable future data. The pre-determined labels are what make this method "supervised."

**Non-Supervised Machine Learning** - Statistically based traffic modeling where the model is built only with network traffic. Requires no pre-determined labels and, thus, no supervision.

**Semi-Supervised Machine Learning** - Statistically based traffic modeling where the model is built with network traffic and a combination of labeled and un-labeled data. Typically, providing some labeled data can aid in the training process.

**Reinforcement Learning** - Statistically based traffic modeling where the model is built in an iterative, trial-and-error process. The model training process aims to balance exploration of un-probed solution space against exploitation of current capability. Between these two is the notion of "cumulative reward" driving the model to maximize the balance at each iteration.

**Traffic Cluster** - A collection of traffic flows that share common characteristics. For example, SSH and Telnet traffic might be clustered together as "interactive session" communications.

**Application Type** - A collection of traffic who's generating application share a common usage pattern. For example, instant messenger traffic would point to an application type of "Chat Application." [15]

**Application Protocol** - A collection of traffic who's generating application share a common protocol. For example, web browsing traffic would point to an application protocol of HTTP or HTTPS.

**Application Software** - A collection of traffic who's generating application share a common characteristic. For example, traffic related to the reading of e-mails would point to an application software label of "E-Mail Client".

**Fine-Grained** - A collection of traffic which points to a specific application (Facebook Messenger, Minecraft Server, etc.)

**Anomaly** - Any system behavior that falls outside an expected norm.