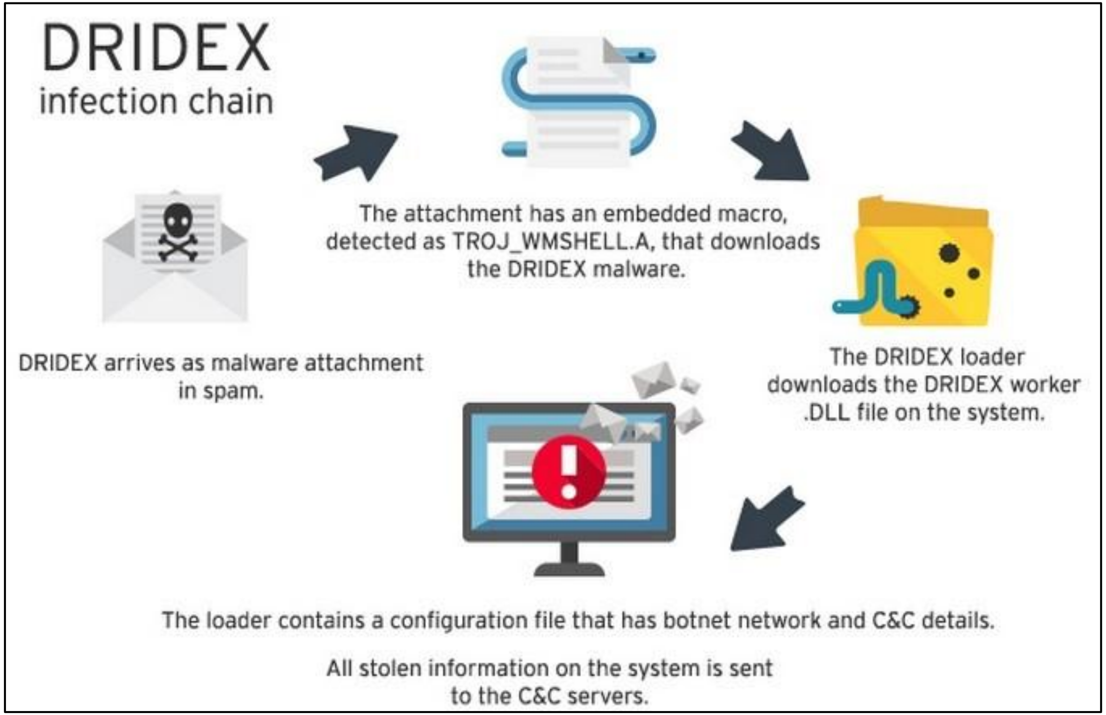
Detecting Malware in TLS traffic

MSc Project Presentation — Olivier Roques

Background

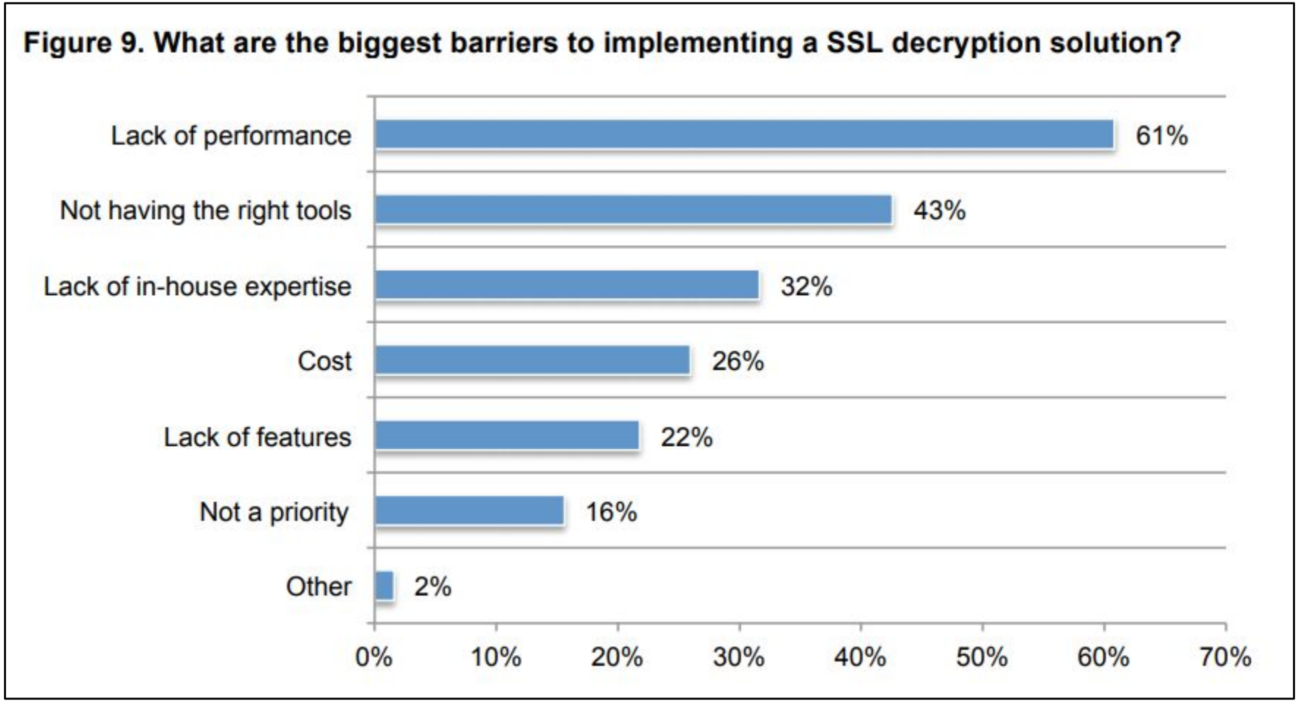
Malware in TLS



Dridex Infection Chain
Source: Trend Micro

- **91%** of pages loaded over HTTPS in the USA in August 2019 (Google)
- **21.5%** of malware used TLS in one way or another in May 2017 (Cisco)

Current Detection Techniques



A survey of 1023 companies regarding TLS decryption platforms

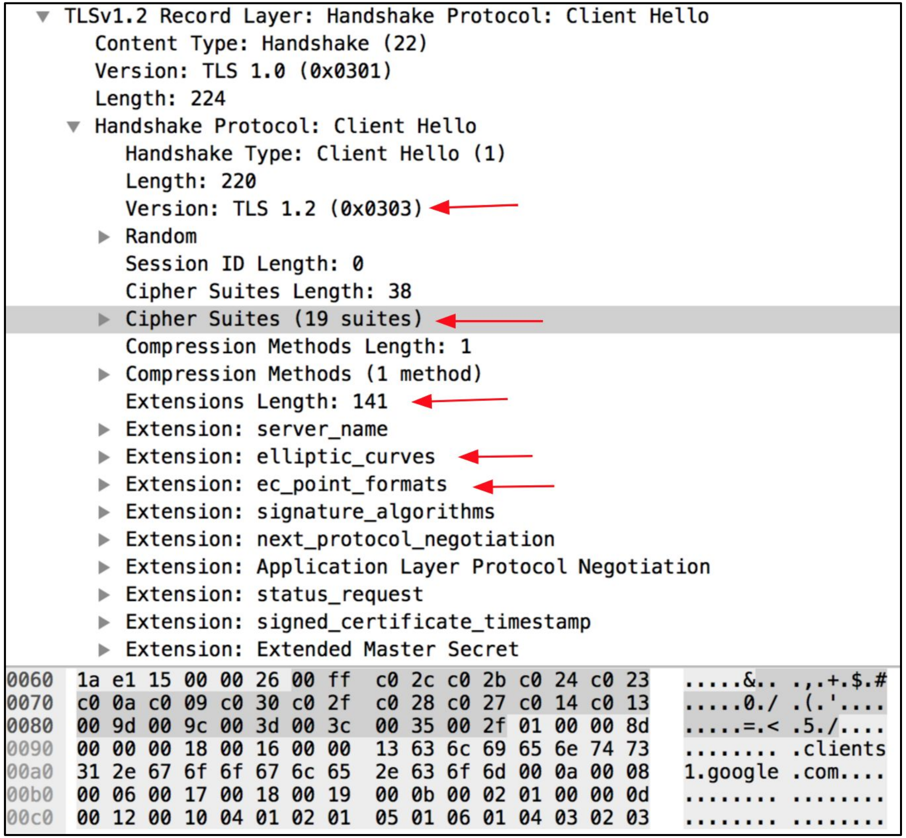
Source: Ponemon Institute

Project’s Objectives

1. **Gathering and arranging capture files** of malware and benign activities
2. Developing a set of tools to automatically **filter and extract features**
3. **Creating and training a TLS classifier** configurable by the user
4. **Integrating the classifier** into a real intrusion detection system (Lastline’s)

lastline

The TLS Protocol & JA3



A ClientHello packet sent by the client
Source: Salesforce

JA3 string format
`Version,Ciphers,Extensions,EllipticCurves,EllipticCurvePointFormats`

Example of a JA3 string
`769,47-53-5-10-49161-49162-49171-49172-50-56-19-4,0-10-11,23-24-25,0`

Resulting JA3 fingerprint
`ada70206e40642a3e4461f35503241d5`

Data Collection and Features

Repartition of Training Flows

Malware Datasets

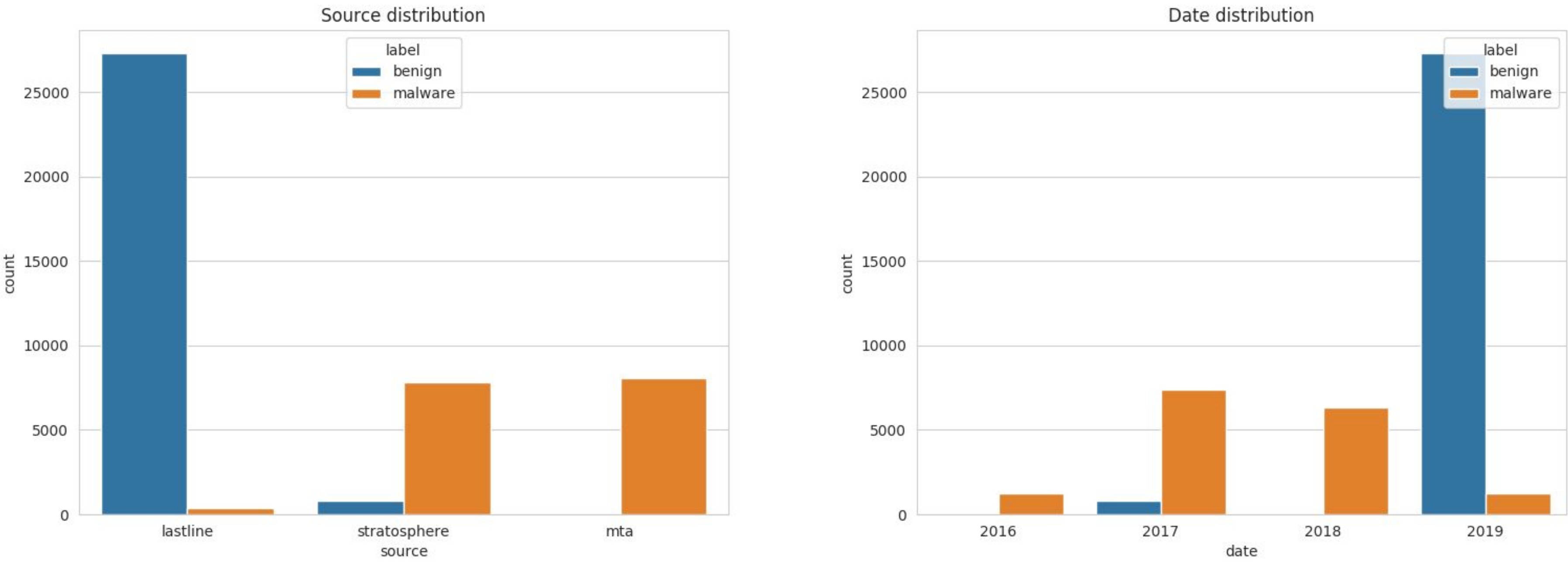
Link to the datasets

Source	Before Filtering	After Filtering	Reduction
malware-mta	17442	8080	-53.7%
malware_stratosphere	507175	7848	-98.5%
malware_lastline	2998	347	-88.4%
TOTAL	**527615**	**16275**	**-96.9%**

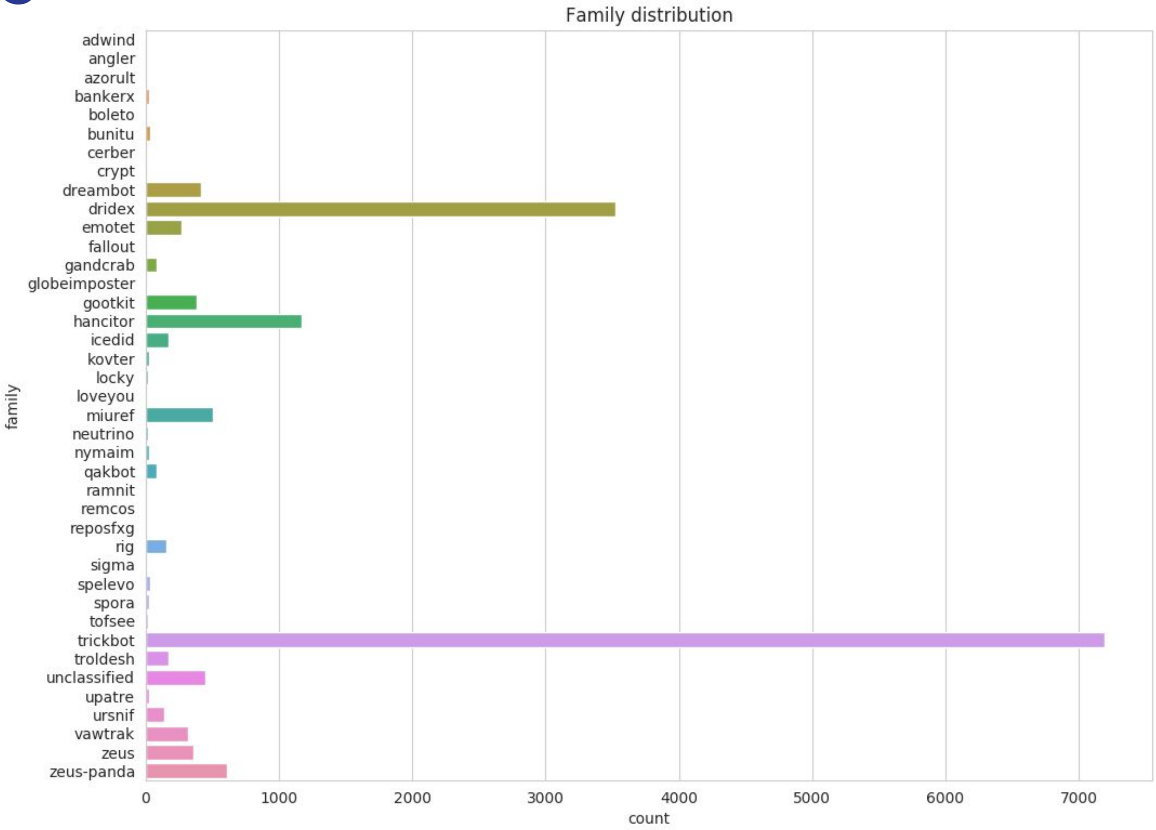
Benign Datasets

Source	Before Filtering	After Filtering	Reduction
benign_lastline-london	104728	15688	-85.0%
benign_lastline-redwood	68858	11627	-83.1%
benign_stratosphere	2427	821	-66.2%
TOTAL	**176013**	**28136**	**-84.0%**

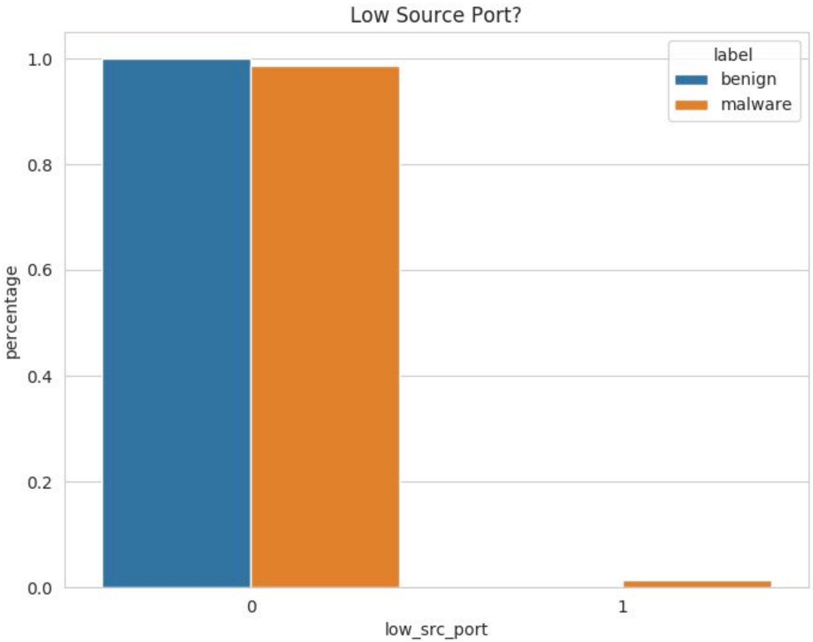
Source and Date of Training Datasets

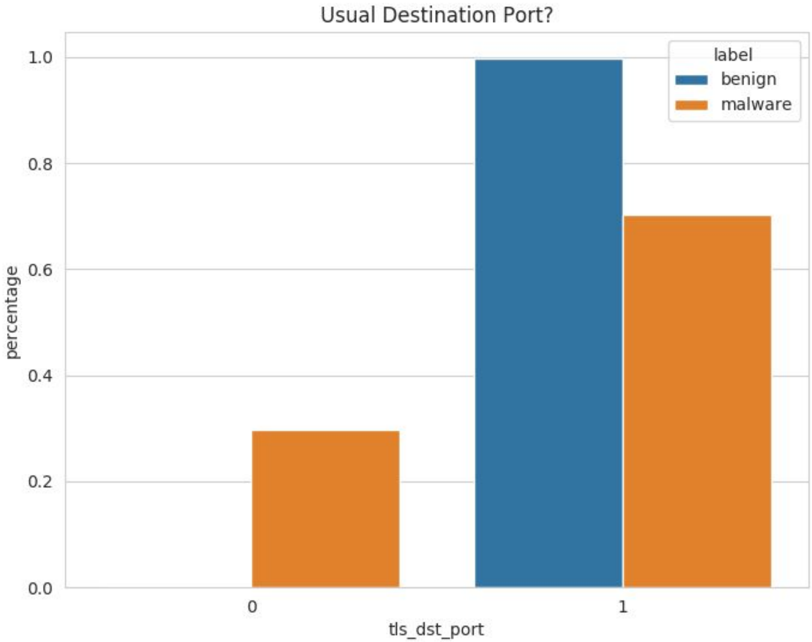


Training Dataset: Malware Families

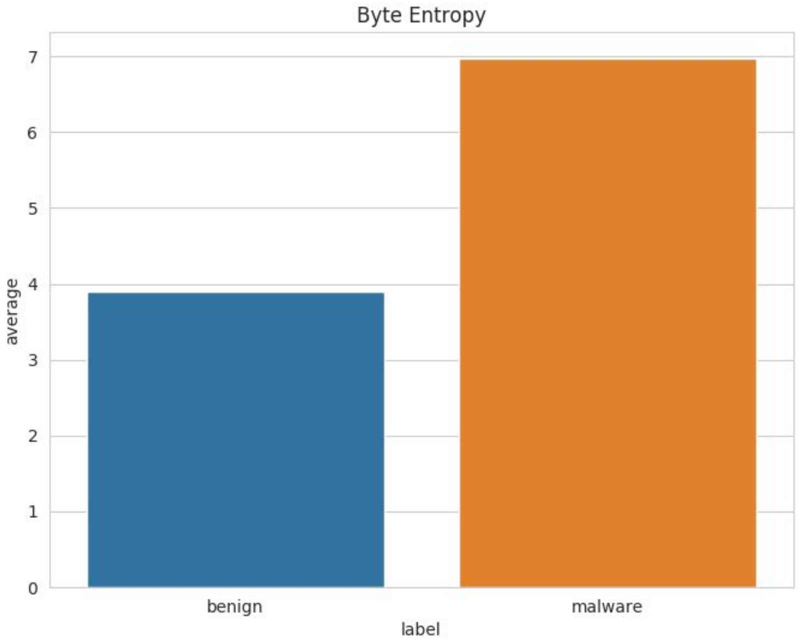


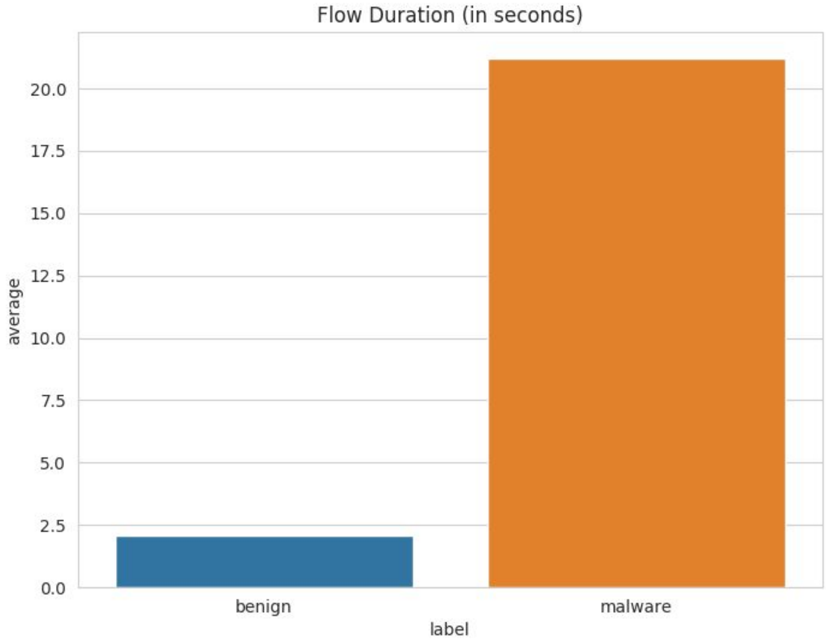
Port Differences



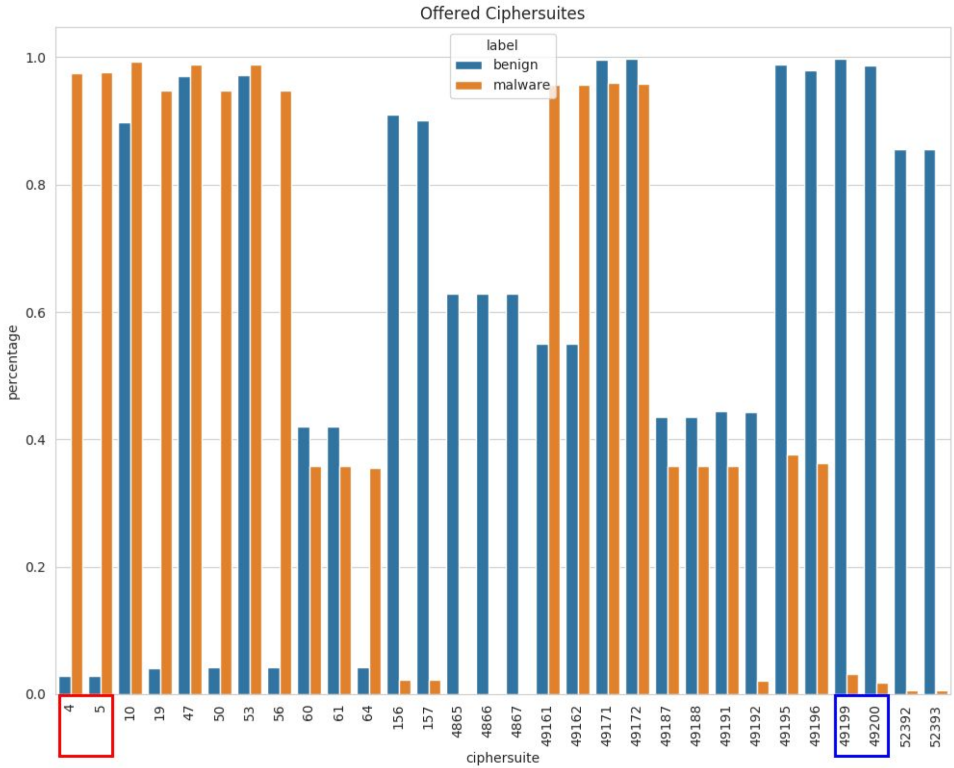


Entropy and Duration Differences





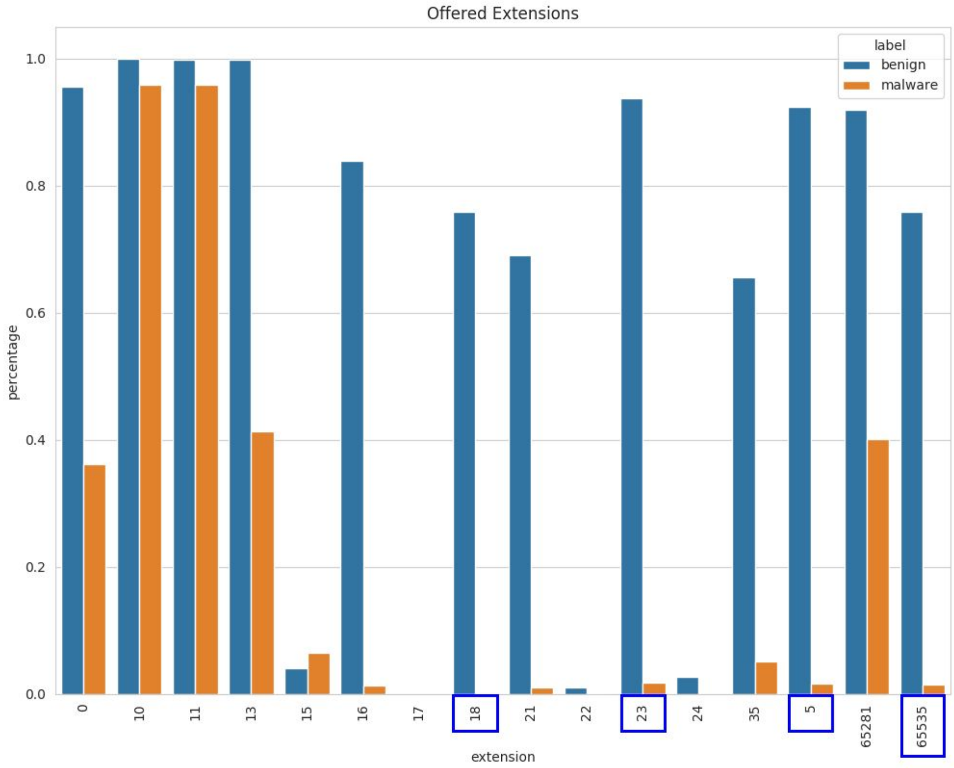
TLS Ciphersuites Differences



Notable Ciphersuites:

- `49199 — TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`: a secure ciphersuite
- `49200 — TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`: idem
- `4 — TLS_RSA_WITH_RC4_128_MD5`: an insecure ciphersuite due to the use of the deprecated RC4 algorithm
- `5 — TLS_RSA_WITH_RC4_128_SHA`: idem

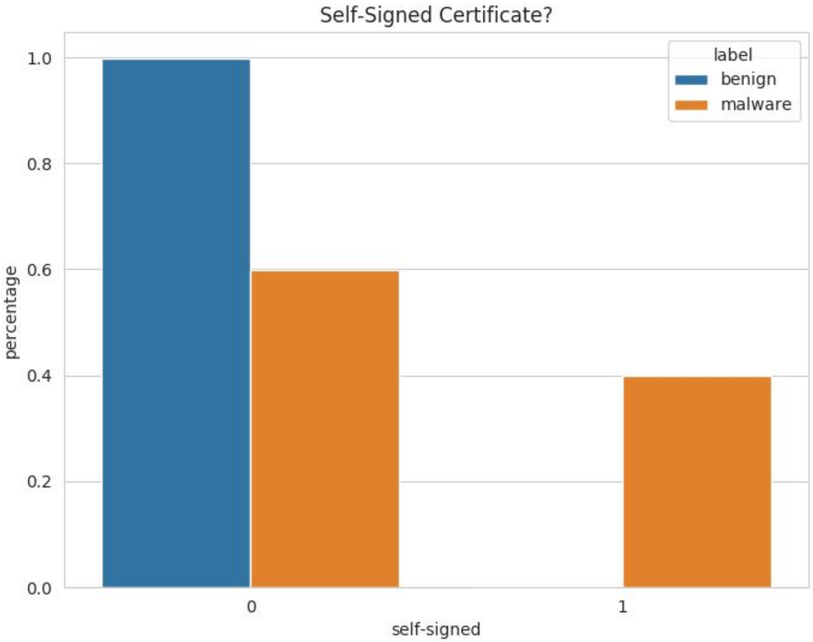
TLS Extensions Differences

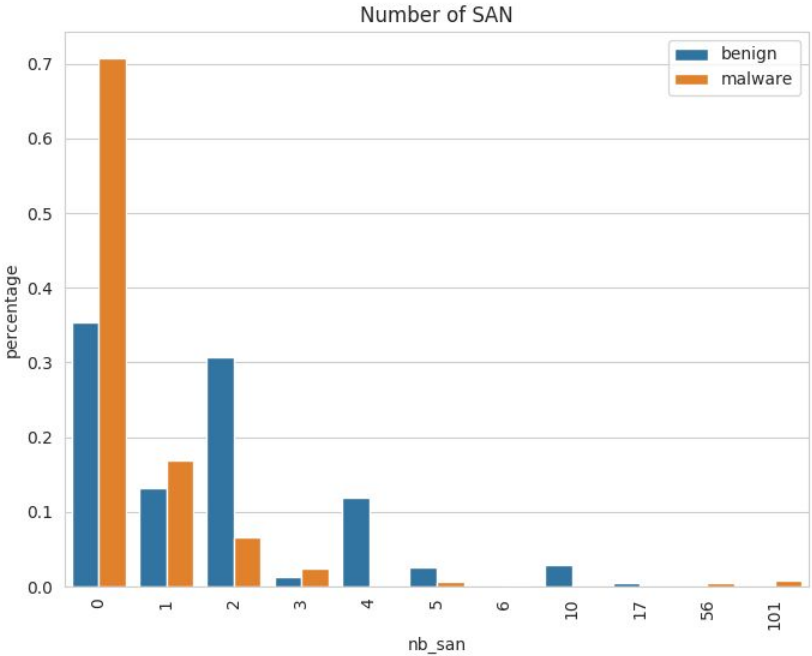


Notable Extensions:

- `5 – status_request`
- `18 – signed_certificate_timestamp`
- `23 – extended_master_secret`
- `65535 – unknown`: placeholder for extensions seen in testing but absent or ignored from the training set. Mostly seen in benign traffic because of Chrome's *GREASE* mechanism which inserts random extensions to make sure webservers ignore unknown values.

Certificate Differences

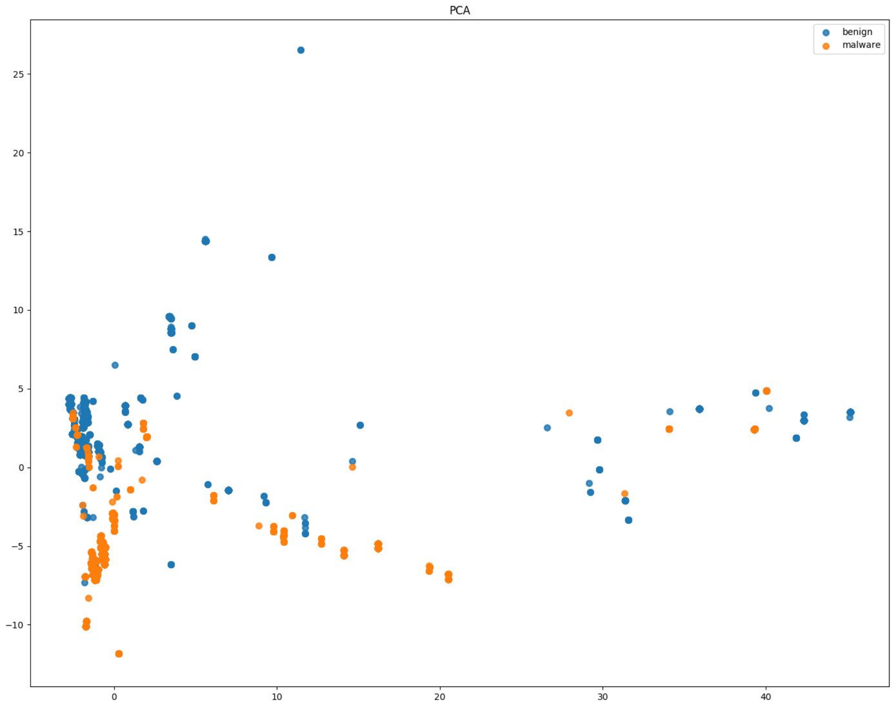




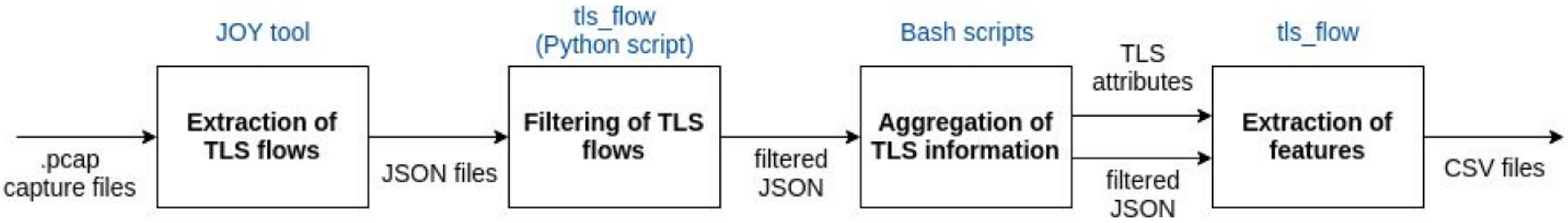
Selected Features

Feature	Size	Dynamic?	In reduced set?	Type
Ephemeral src port	1	No	Yes	Boolean
TLS dest port	1	No	Yes	Boolean
Nb of inbound bytes	1	No	No	Integer
Nb of outbound bytes	1	No	No	Integer
Nb of inbound packets	1	No	No	Integer
Nb of outbound packets	1	No	No	Integer
Flow duration	1	No	No	Integer
SPL	100	No	No	Stochastic matrix
SPT	100	No	No	Stochastic matrix
Byte dist mean	1	No	No	Float
Byte dist std	1	No	No	Float
Byte entropy	1	No	No	Float
Ciphersuites	146	Yes	Yes	Binary vector
Extensions	16	Yes	Yes	Binary vector
Nb of extensions	1	No	Yes	Integer
Supported Groups	36	Yes	Yes	Binary vector
Point Formats	4	No	Yes	Binary vector
Client's key length	1	No	No	Integer
Certificate's validity	1	No	Yes	Integer
Certificate's nb of SAN	1	No	Yes	Integer
Self-signed certificate	1	No	Yes	Boolean
Total	417		208	

Data Visualization: PCA



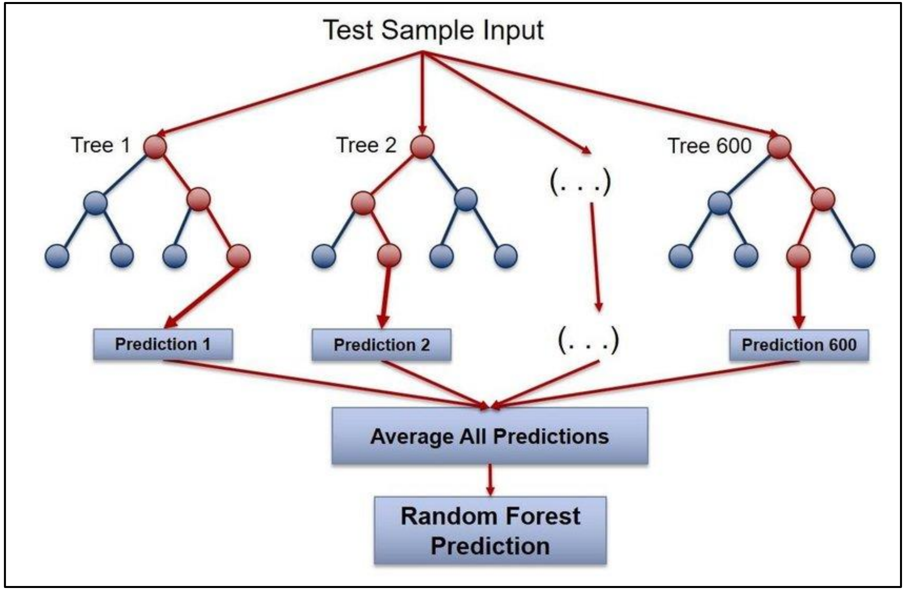
Extraction and Filtering Pipeline



```
olivier@olivierLT:~/tls-malware-detection/datasets$
```

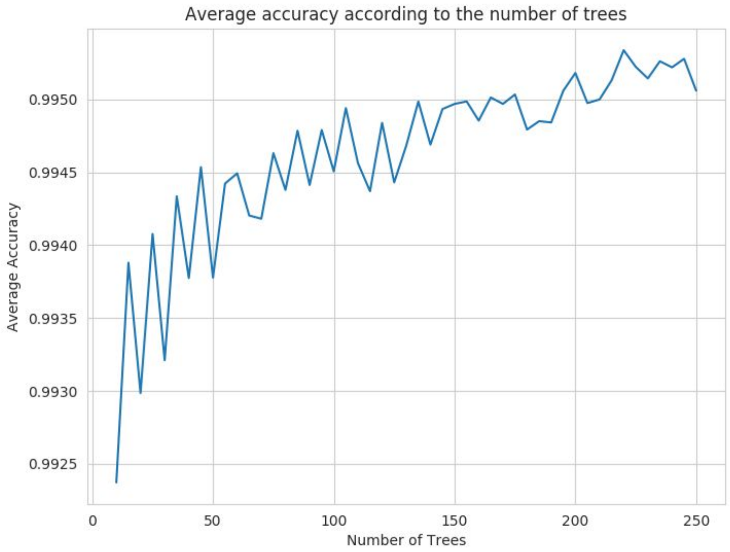
TLS Classifier

The Classifier: Random Forest



Random Forest visualization

Source: Evaluation and Comparison of Machine Learning Techniques for Rapid QSTS Simulations (Logan Blakely, 2018)



⇒ Final choice: **130 trees**

Results: 10-Fold Cross-Validation

```
Testing set size: 4441
Average accuracy: 0.9995
Average weighted precision: 0.9996
Average weighted recall: 0.9995
Average weighted f1-score: 0.9995
```

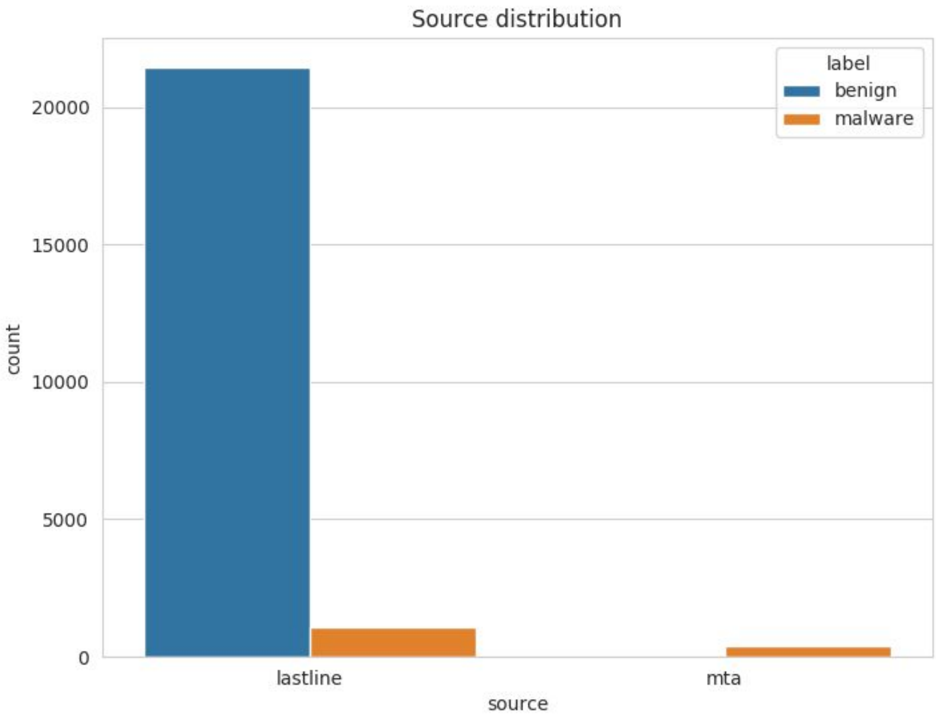
Results of 10-fold cross validation

High performances due to ***temporal experimental bias***.

The model assumes that flows are **identically distributed in time.**
⇒ Training folds will include flows **anterior and posterior** to flows in the testing fold.
⇒ Results in **inflated scores**

Fresh Testing Datasets

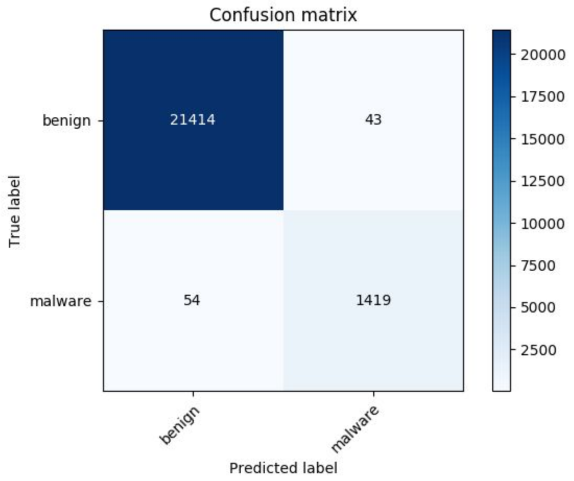
	Number of flows
malware_test	1473
benign_test	21457

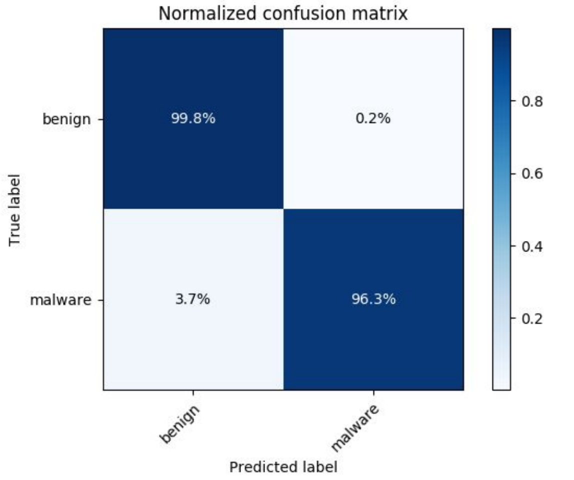


Results: Fresh Dataset

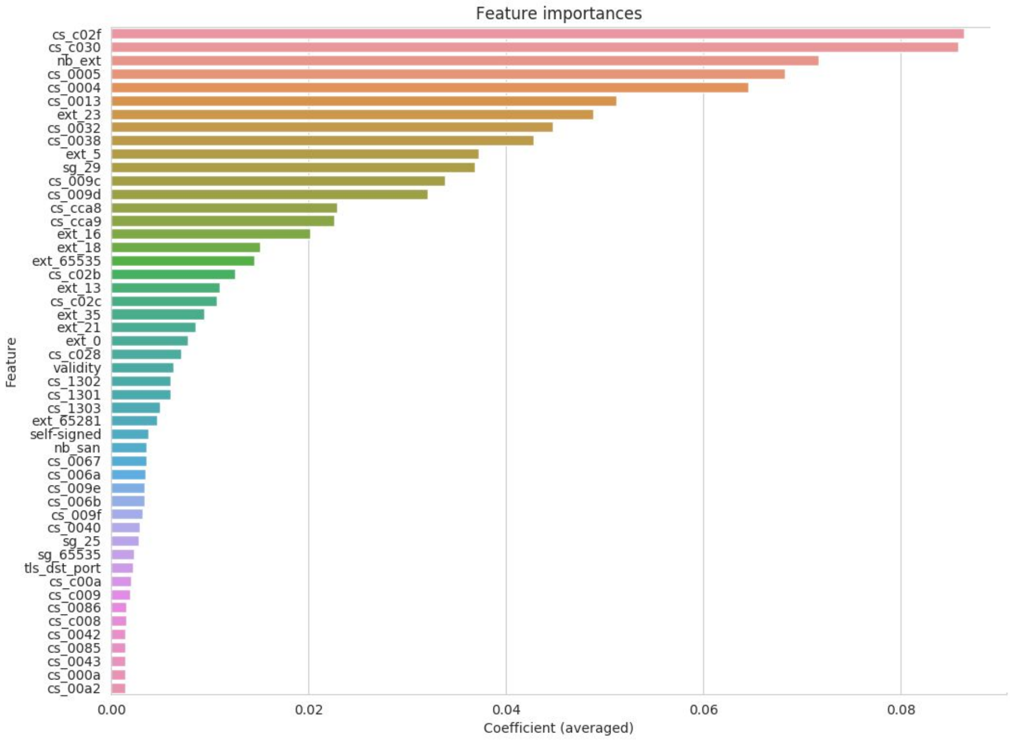
```
[INFO] Stratification based on families enabled
[INFO] Prediction threshold: 0.50
              precision    recall  f1-score   support

      benign     0.9973    0.9975    0.9974     21457
     malware     0.9639    0.9613    0.9626      1473

    accuracy                         0.9952     22930
   macro avg     0.9806    0.9794    0.9800     22930
weighted avg     0.9952    0.9952    0.9952     22930
```

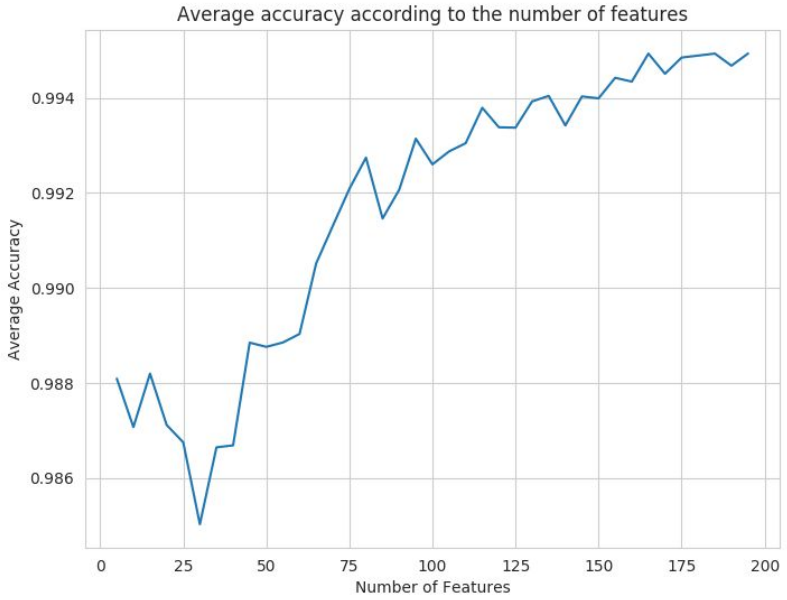
Results on the fresh dataset



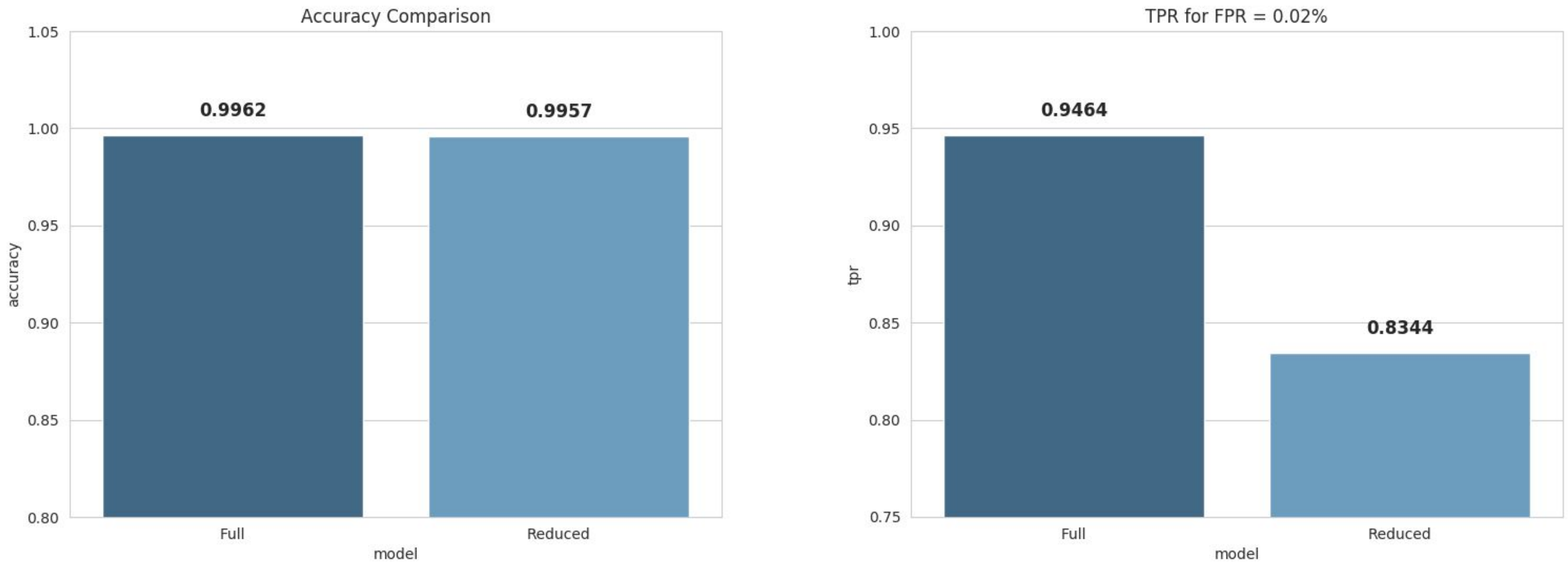


Best Features

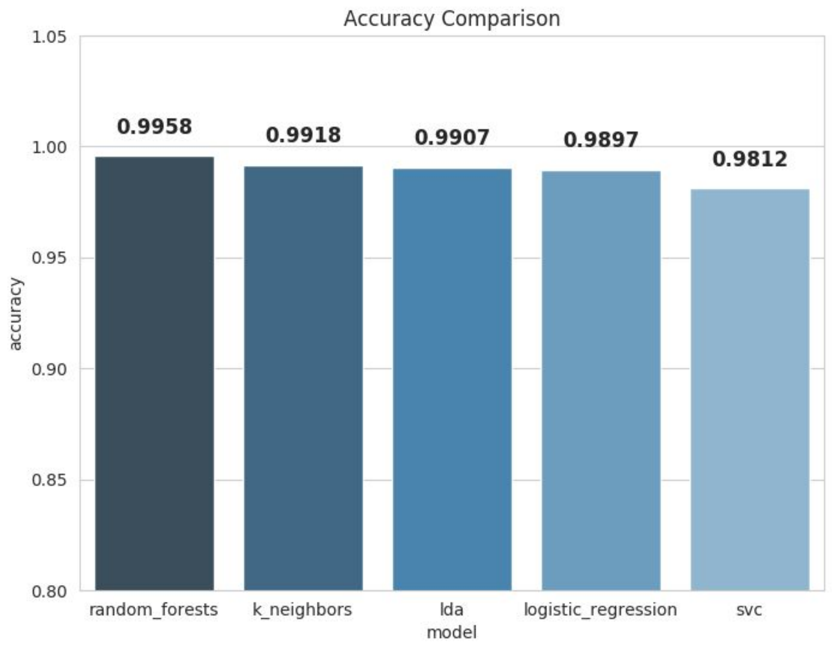


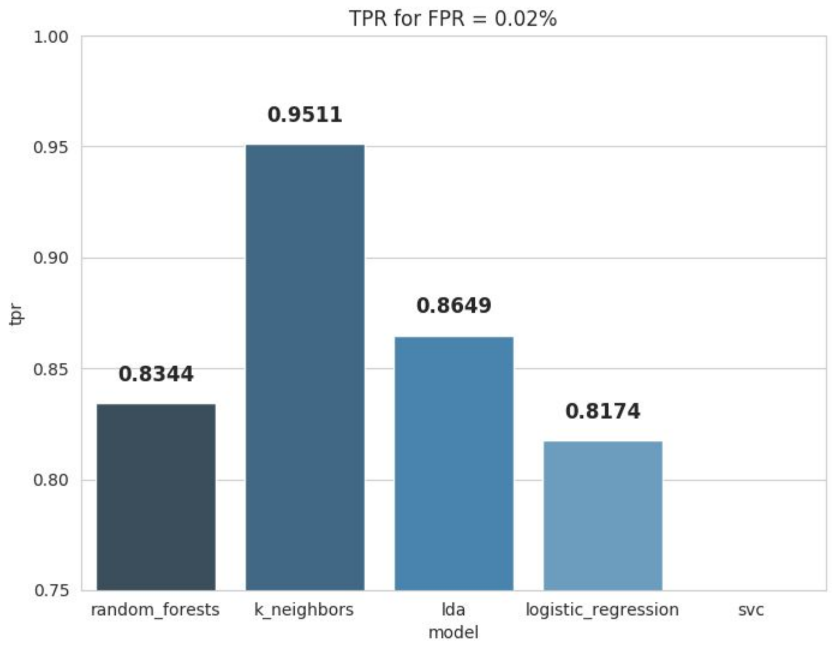


Influence of Features



Model Comparison





The TLS Detector

Lastline's IDS, ***Llanta***, is made of:

- **Sensors** that collect the traffic
- **Detectors** that receive the traffic and raise alerts

The classifier was integrated into Llanta in four steps:

1. The trained classifier was converted into a **Debian package**
2. Creation of a Python module to **transform raw TLS flows into TLS vectors**
3. **Development of a detector** which:
    a. Creates TLS vectors from raw TLS flows
    b. Forwards vectors to the classifier and collects its predictions
    c. Raises an alert when more than a certain number of flows are classified as malicious
4. **Extensive documentation was written** for the internal documentation platform. Capture files and datasets were arranged and saved to Lastline's data repository.

Discussion

Related Work

Category	Feature	Type
Flow Metadata	Source port	Integer
	Destination port	Integer
	Number of inbound bytes	Integer
	Number of outbound bytes	Integer
	Number of inbound packets	Integer
	Number of outbound packets	Integer
	Duration of the flow	Integer
Distribution	Sequence of packet lengths	Stochastic matrix
	Sequence of packet times	Stochastic matrix
	Byte distribution	Length-256 Array
TLS Metadata	List of ciphersuites	Binary vector
	List of TLS extensions	Binary vector
	Client's public key length	Integer
	Selected cipher suite	Integer
	Selected extensions	Binary vector
	Number of SAN	Integer
	Validity (in days)	Integer
	Certificate self-signed or not	Boolean

Cisco's features for TLS malware detection

Source: Deciphering Malware's use of TLS (without Decryption)

	Reduced set		Full set	
	0.5	**0.9**	**0.5**	**0.9**
Cisco	97.67%	80.76%	99.35%	85.80%
Lastline	96.13%	83.44%	97.11%	94.64%

Malware recall of Lastline's and Cisco's classifier for different thresholds

Limitations: Base Rate Fallacy & Number of FP

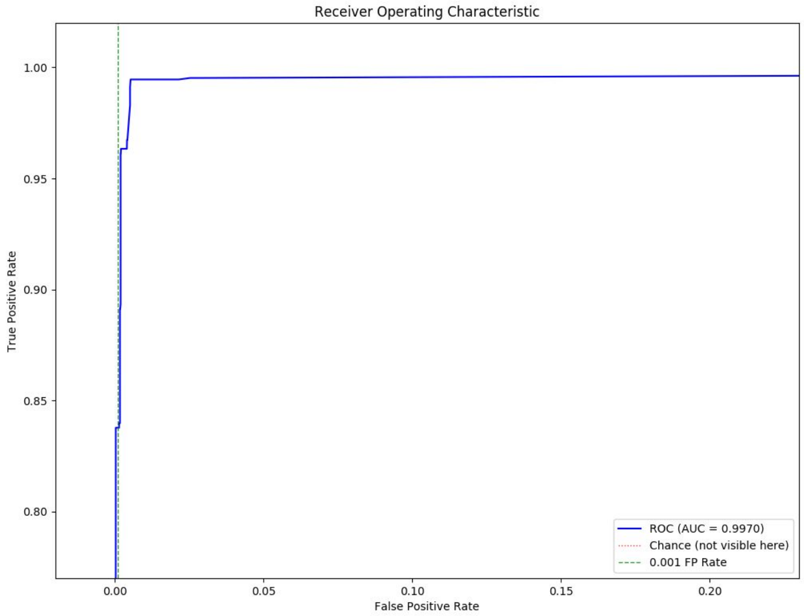
Bayes' theorem:

$$P(I \mid A) = \frac{P(A \mid I) \cdot P(I)}{P(A)} = \frac{P(A \mid I) \cdot P(I)}{P(A \mid I) \cdot P(I) + P(A \mid \bar{I}) \cdot P(\bar{I})}$$

The goal is to **maximize *P(I | A)***, the probability that a host is infected given that an alarm was raised

- *P(A | I)*: probability that an alarm is raised when an infection happens
  _P(A | I)_ = True Positive Rate (_TPR_)
- *P(A | Ī)*: probability that an alarm is falsely raised
  _P(A | Ī)_ = False Positive Rate (_FPR_)
- *P(I)*: probability that a TLS flow is malicious
  _P(I)_ = 0.00005 (estimate)

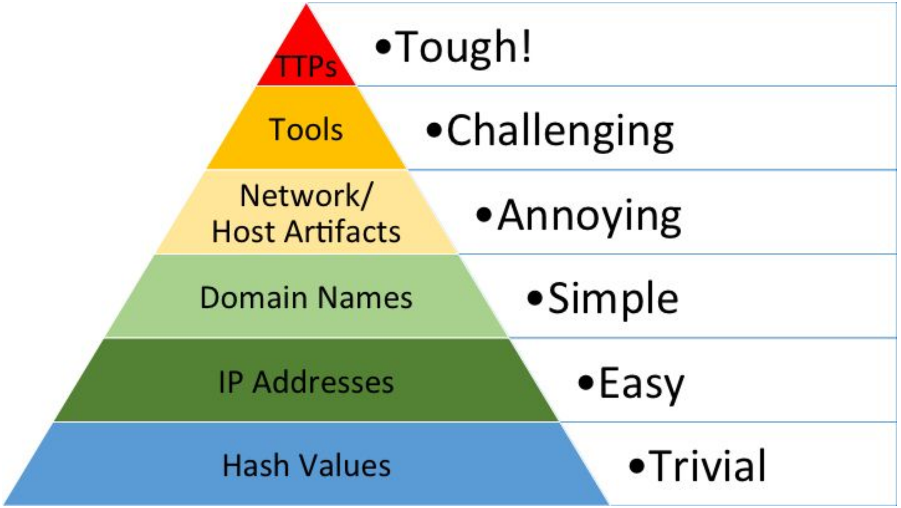
Threshold	*TPR*	*FPR*	*P(I | A)*
0.50	0.9946	0.0134	**0.37%**
0.92	0.8344	0.0002	**17.26%**



Limitations: Robustness of Features

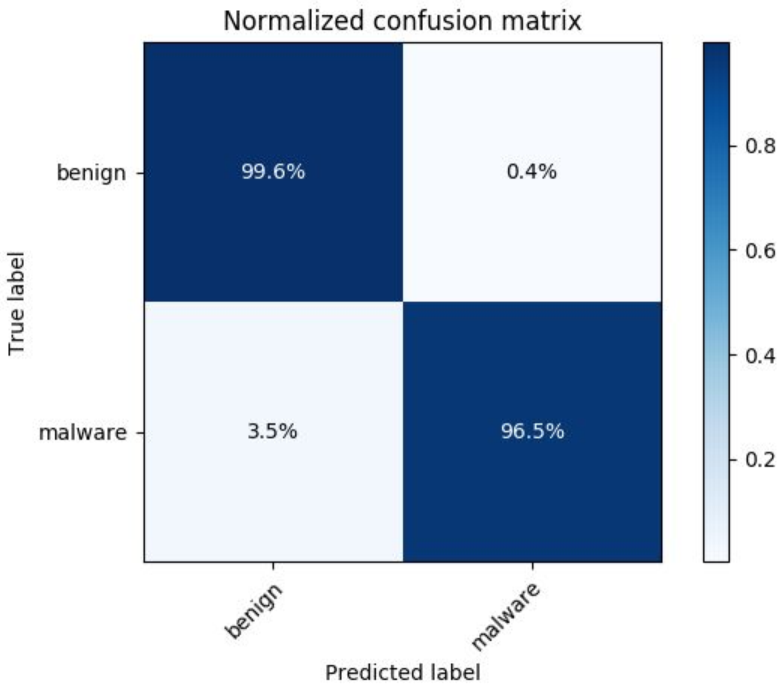
Feature	Score
Source and Destination ports	1
Bytes in and out	3
Packets in and out	3
Duration	3
Sequence of Packet Lengths	3
Sequence of Packet Times	3
Byte Distribution	3
Entropy	1
Ciphersuites	2
Extensions	2
Number of Extensions	2
Elliptic Curve Groups	2
Elliptic Curve Point Formats	2
Client's Key Length	1
Certificate's Validity	1
Certificate's Number of SAN	1
Self-Signed Certificate	1

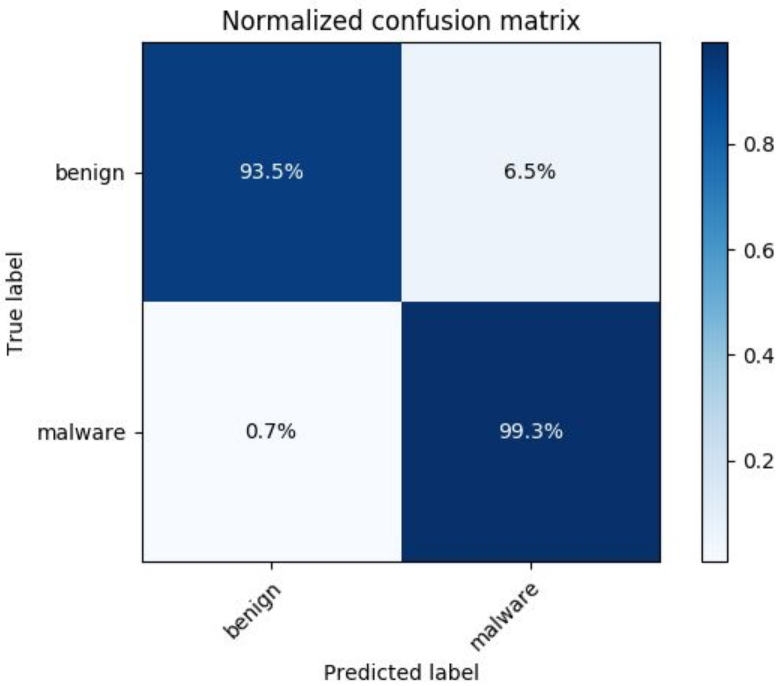
- **Robustness of features**, with an arbitrary score from weak (1) to robust (3).



The Pyramid of Pain: an empirical method to evaluate the robustness of IoCs.
Source: Enterprise Detection & Response

Limitations: Influence of the Benign Dataset





Trained on: London & Redwood offices
Tested on: Redwood office

Trained on: London office
Tested on: Redwood office

Possible Improvements

- **Automate the collection of capture files** both benign and malicious
- **Use the full set of features** to increase robustness and TPR
- **Use features from other protocols** (DNS, HTTP) to improve robustness and TPR
- **Combine the classifier with other detectors** (DGA, JA3) to reduce false positives

Weight	Feature
3.38	DNS Suffix `org`
2.99	DNS TTL `3600`
2.62	TLS Ciphersuite `TLS_RSA_WITH_RC4_128_SHA`
2.28	HTTP Field `accept-encoding`
1.95	TLS Ciphersuite `SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA`
1.78	HTTP Field `location`
1.38	DNS `Alexa: None`
1.21	TLS Ciphersuite `TLS_RSA_WITH_RC4_128_MD5`
1.12	HTTP Server `nginx`
1.11	HTTP Code `404`

Cisco's top 10 features from different protocols

Source: Identifying Encrypted Malware Traffic with Contextual Flow Data

JA3 Detector

How it works:

1. The detector **extracts JA3** from all TLS flows
2. A **profile of typical JA3 hashes is build** for each host
3. After that training period, **JA3 hashes absent from a host's profile and its neighbors'** are flagged as suspicious

```
-> anomalous JA3 fingerprint (a1674500365bdd882188db63730e69a2): None
-> anomalous JA3 fingerprint (54328bd36c14bd82ddaa0c04b25ed9ad): hola_svc
-> anomalous JA3 fingerprint (a0e9f5d64349fb13191bc781f81f42e1): Malware Test FP: fake-font-update-for-firefox
-> anomalous JA3 fingerprint (4e30215bd4af6afe796e9ff893e7f3cd): None
```

Thank You!