Lecture #6: Network Security & Monitoring #2

WPI CS4516 Spring 2019 D term

Instructor: Lorenzo De Carli (ldecarli@wpi.edu)

(slides include material from Christos Papadopoulos, CSU)

Previously on CS4516

- Attack recap
  - Large-scale DDoS attacks
    - Take hosts offline using **sheer volume of traffic**
    - CDNs based on **reverse proxying** can protect
  - Attacks against hosts
    - In the early days, **worms**
    - Nowadays, **malware**
    - …also, **remote exploits** (yep, still a problem)
    - Protection: **network monitoring tools**
- **Intrusion detection/prevention systems** (IDS/IPS)
  - Identify attack traffic by performing **deep packet inspection (DPI)**

Previously on CS4516 /2

- **Signature-based IDS:** remember those?
- Perform DPI using byte-level patterns in packets' payloads
- Initially, based on **fixed string matching**
  - **Example:** Code Red attack consists of constant string: *GET /default.ida?NNNNNNNNNNNNNNNNN…*
- Later, extended to use **regular expressions**
  - Sommer & Paxson, "Enhancing Byte-level network intrusion detection signatures with context", CCS 2003

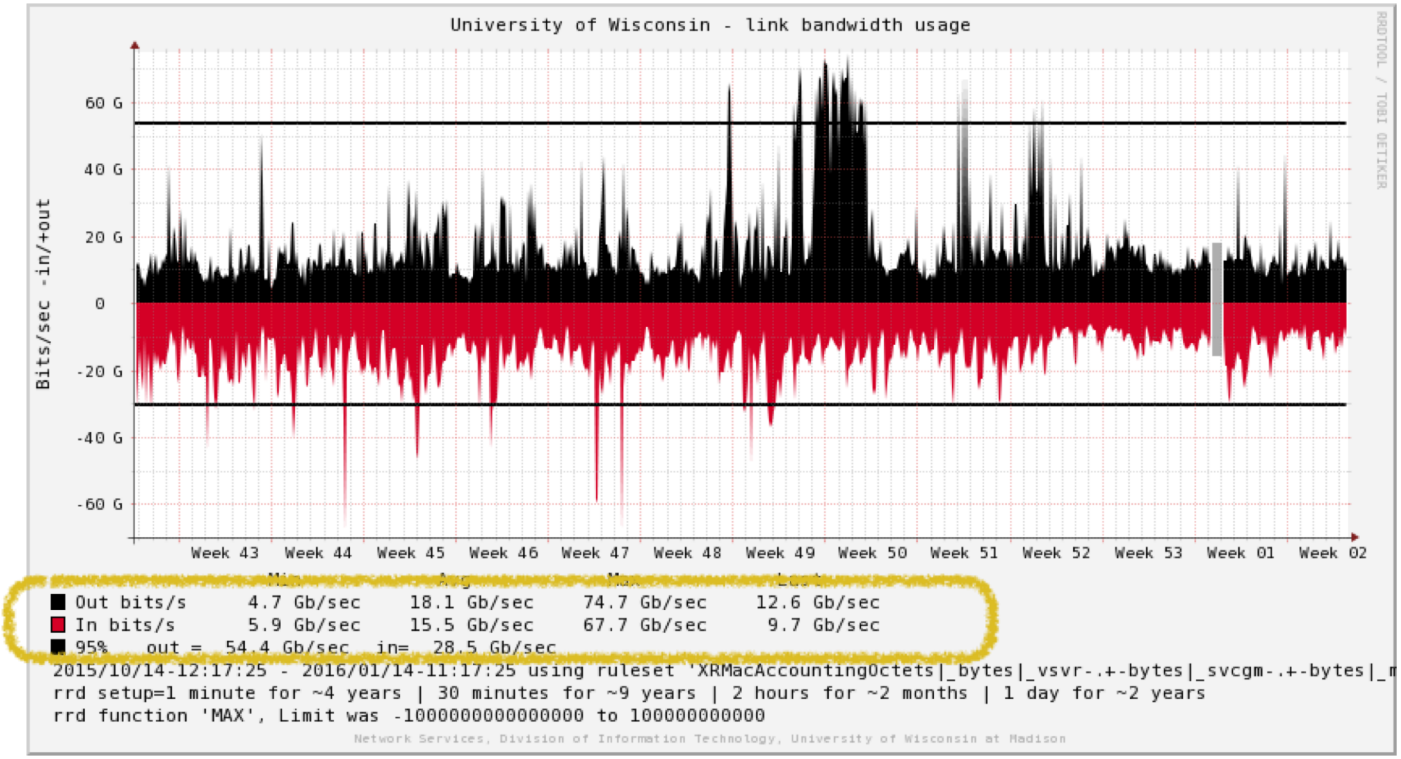
Previously on CS4516 /3

- **Policy-based IDSs**
- Perform DPI by **analyzing and correlating network events**
- Separates **mechanism** (protocol parsing and generation of network events)…
- …from **policy** (attribution of meaning, if any, to sets of one or more events)

Some more issues in Intrusion Detection

Parallelizing intrusion detection

- Issue: regardless of the approach (signature- or policy-based), network links of large organizations carry too much traffic for a single IDSs



Cycles available to analyze each packet at 20Gb/s: **~250**

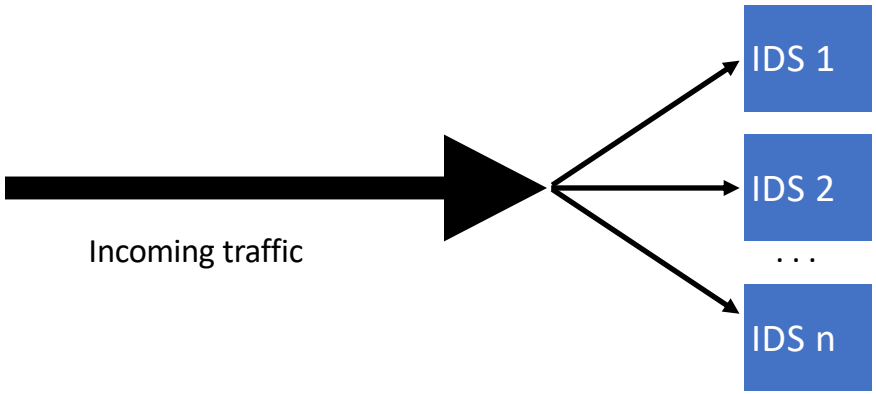
Cycles necessary to analyze a DNS message in Bro: **~4000**

Why is this a problem?

- **Effectiveness!**
- An IDS that cannot “keep up” with the rate of data seen on the wire must drop packets
- Dropping packets entails the possibility of missing attacks

Parallelizing intrusion detection/2

- **Conclusion:** need multiple IDS's for the same link



- What is the best way to split traffic?
  - Simplest approach: randomly distribute packets/flows to IDS nodes

Parallelizing intrusion detection/3

- **Problem with naive parallelization?**
  - Attacks that span multiple packets/connection may go undetected
- **Solutions:**
  - Allow IDS nodes to share state (Vallentin et al., 2007)
  - Split traffic in an attack-aware fashion (Kruegel et al., 2002; Sommer et al., 2009; De Carli et al., 2014)

Data recording

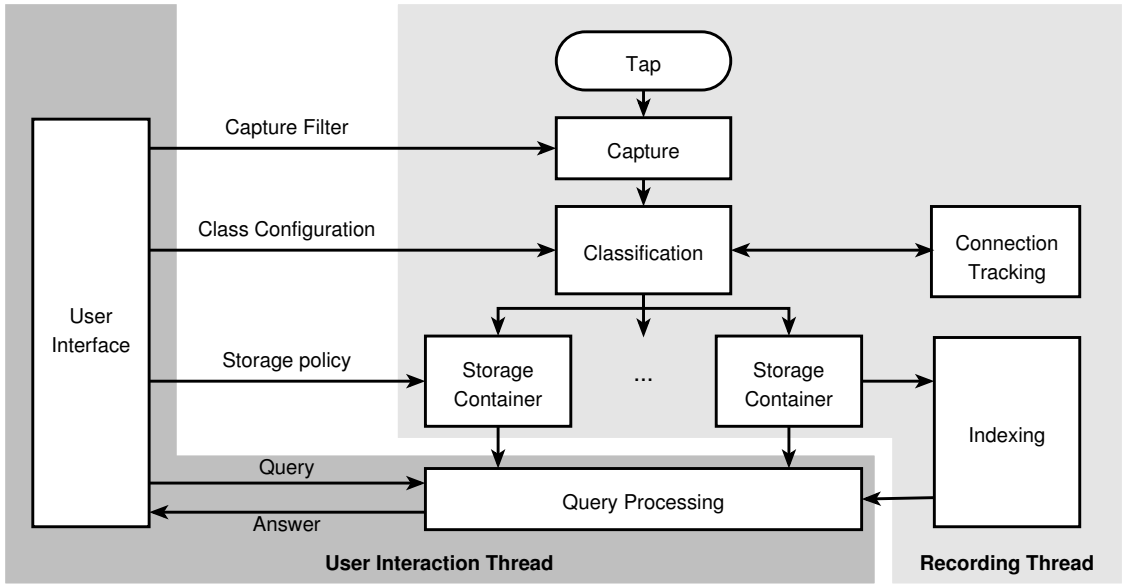
- An attack have been detected, now what?
- Detection per se does not provide crucial information:
  - Was the attack successful?
  - How much damage was done?
- The ability to perform **forensics** is an essential component of a security plan!

Data recording/2

- Network monitoring system needs to record data to enable forensics
- Obviously can’t record all traffic forever
  - Privacy implications
  - Just technically impossible!

The Time Machine

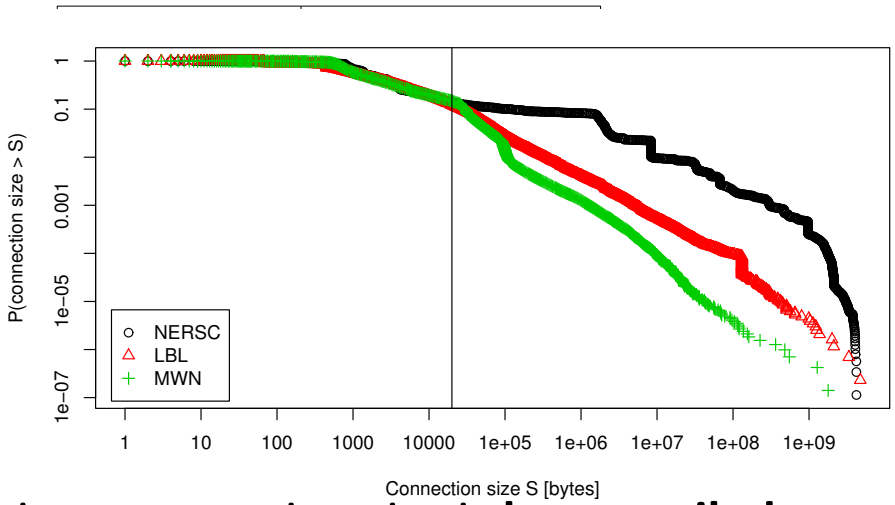
- Kornexl et al., “Building a Time Machine for Efficient Recording and Retrieval of High-Volume Network Traffic”



- Idea: build a system to efficiently filter and record only **relevant network data**

The Time Machine/2

- What constitutes “relevant network data”?



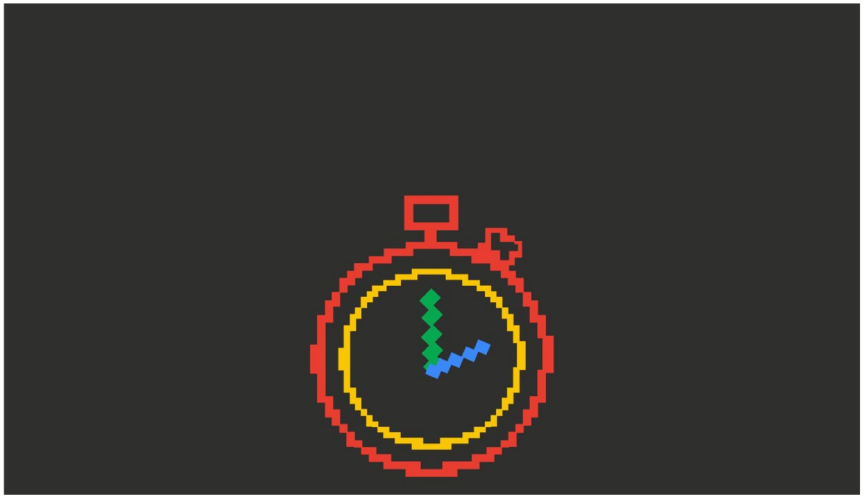
- Observation: connection size is **heavy-tailed**
  - LBL dataset: **12% connections generate 96% of data**
- Idea: **only record the a small amount of data (e.g. 20KB) for each connection**
  - Small connection recorded in full
  - Reasonable characterization of large connections **(attacks are at the beginning anyway)**

Modern developments

- Service storing **“network telemetry”** in the cloud
  - DHCP/DNS logs
  - Flow records
  - Packet captures
  - Logs
  - …
- **Advantages:** can keep unlimited history, can run state-of-the-art threat detection algorithms
- Disadvantages?

LILY HAY NEWMAN SECURITY 03.04.19 04:10 PM

AN ALPHABET MOON SHOT WANTS TO STORE THE SECURITY INDUSTRY'S DATA



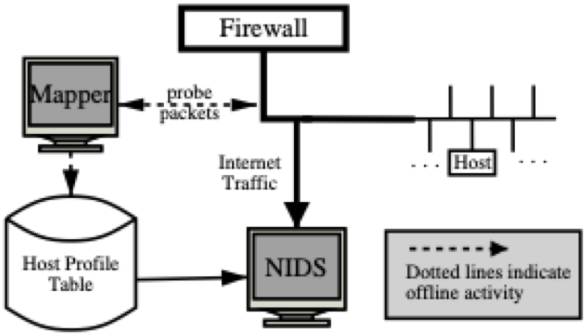
Wired, 3/4/2019

Stream reconstruction

- TCP/IP vagaries can be used to cause IDS to **“see” a different stream that a victim host**
- Example: send benign packet w/ TTL N (s.t. N is enough to read IDS, but not end-host); then send duplicate packet w/ TTL N+1 and attack payload
  - **IDS may discard duplicate, but victim will suffer attack**
- Many other techniques
  - MTU, overlapping TCP sequence numbers, etc.

Stream reconstruction/2

- Potentially serious problem (form of **IDS evasion**)
- Some solutions in literature:
- Active mapper (Dharmapurikar & Paxson, 2003): determines host parameters (MTU, #hops, version of TCP stack), provides them to IDS



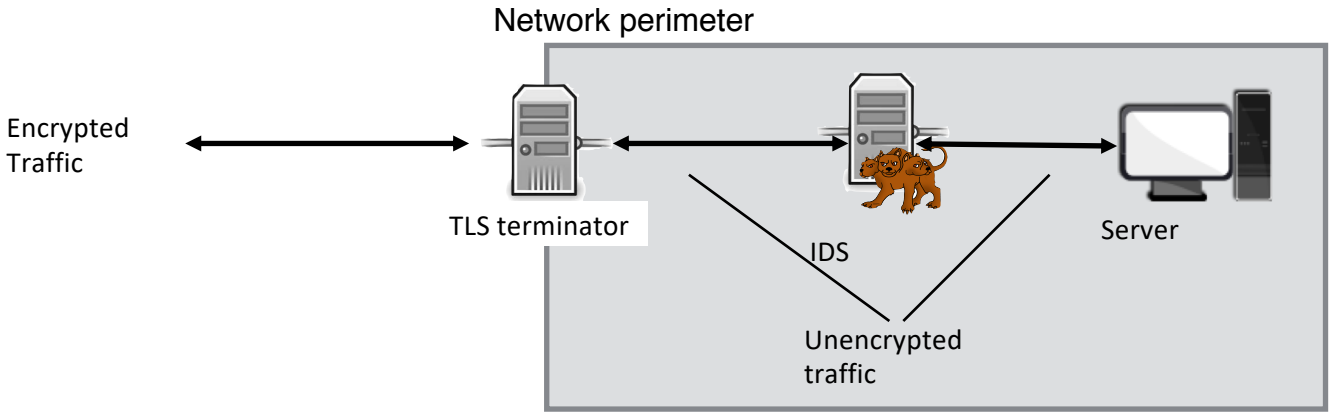
- Robust TCP normalization (Vutukuru et al., 2008): actively remove inconsistency from TCP stream
  - **Issues?**

What about encryption?

- Modern network communications tend to be encrypted
  - E.g. HTTP -> HTTPS
- DPI on encrypted traffic cannot be directly performed
  - Why?
- Can still do firewalling based on packet headers, but it may not be enough

Dealing with encryption

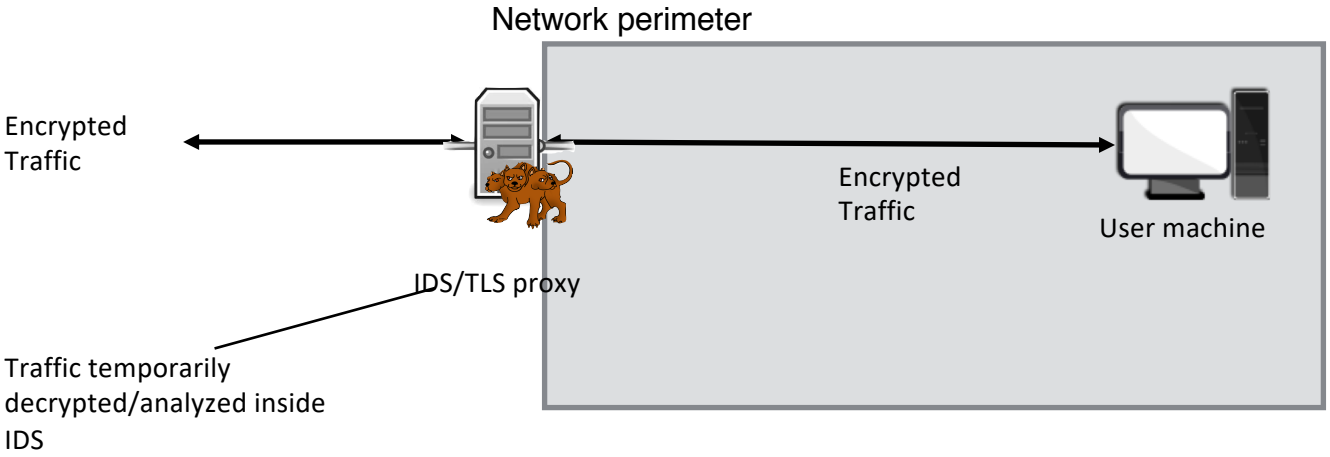
- Possible solution: TLS termination



- May be acceptable for traffic to/from servers
- May not be for traffic to/from user machines
  - User may want guarantees that their traffic remains private even within their organization network
  - Regulations (e.g. HIPAA) may require traffic to remain encrypted at all times

Dealing with encryption/2

- Possible solution #2: TLS proxying



- How does it work?
  - Install root certificate in user machines
  - Proxy terminate TLS connections, analyze traffic, re-encrypt with its own key
  - Client does not complain because it trusts root certificate

Is TLS proxying a good idea, technically?

- Unclear; many TLS proxying products are riddled with bugs/vulnerabilities

The Security Impact of HTTPS Interception

Zakir Durumeric[*∨], Zane Ma[†], Drew Springall[*], Richard Barnes[‡], Nick Sullivan[§], Elie Bursztein[¶], Michael Bailey[†], J. Alex Halderman[*], Vern Paxson[‖∨]

[*] University of Michigan [†] University of Illinois Urbana-Champaign [‡] Mozilla [§] Cloudflare
[¶] Google [‖] University of California Berkeley [∨] International Computer Science Institute

***Abstract*—As HTTPS deployment grows, middlebox and antivirus products are increasingly intercepting TLS connections to retain visibility into network traffic. In this work, we present a comprehensive study on the prevalence and impact of HTTPS interception. First, we show that web servers can detect interception by identifying a mismatch between the HTTP User-Agent header and TLS client behavior. We characterize the TLS handshakes of major browsers and popular interception products, which we use to build a set of heuristics to detect interception and identify the responsible product. We deploy these heuristics at three large network providers: (1) Mozilla Firefox update servers, (2) a set of popular e-commerce sites, and (3) the Cloudflare content distribution network. We find more than an order of magnitude more interception than previously estimated and with dramatic impact on connection security. To understand why security suffers, we investigate popular middleboxes and client-side security software, finding that nearly all reduce connection security and many introduce severe vulnerabilities. Drawing on our measurements, we conclude with a discussion on recent proposals to safely monitor HTTPS and recommendations for the security community.**

connection to the destination server. We show that web servers can detect such interception by identifying a *mismatch* between the HTTP User-Agent header and the behavior of the TLS client. TLS implementations display varied support (and preference order) for cipher suites, extensions, elliptic curves, compression methods, and signature algorithms. We characterize these variations for major browsers and popular interception products in order to construct heuristics for detecting interception and identifying the responsible product.

Next, we assess the prevalence and impact of HTTPS interception by applying our heuristics to nearly eight billion connection handshakes. In order to avoid the bias inherent in any single network vantage point, we analyzed connections for one week at three major Internet services: (1) Mozilla Firefox update servers, (2) a set of popular e-commerce websites, and (3) the Cloudflare content distribution network. These providers serve different types of content and populations of users, and we find differing rates of interception: 4.0% of Firefox update connections, 6.2% of e-commerce connections, and 10.9% of

(NDSS 2017)

Some problems with TLS proxying

- Root certificate is all-powerful
  - What could go wrong?
- TLS stack of end-hosts usually incorporates modern security design (no outdated cyphers, well-known bugs fixed etc.)
  - TLS stack of interception box may not

Activity!

- Create groups of 4 to 5 people
- Come up with:
  - Objections against deep packet inspection (why is it a bad idea in the modern Internet?)
  - Answers to these objections (why it may still be a good idea?)
- Will go through groups randomly
  - Each round, one group will presents a reason why DPI is a bad idea and another group will attempt to debate it
  - When a group does not have an answer, we'll skip to next group

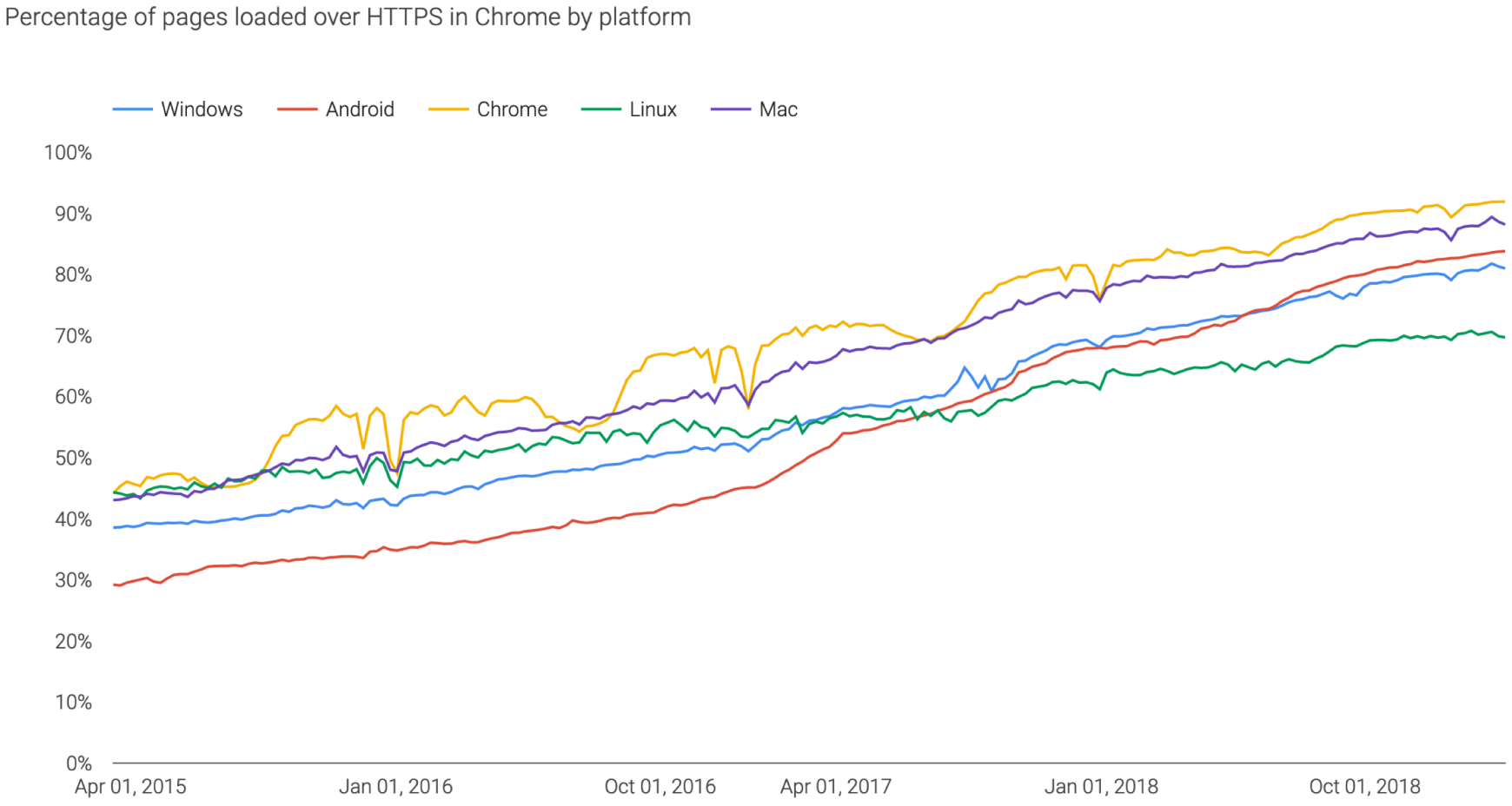
The end-to-end encryption debate

End-to-end encryption

Deep Packet Inspection

Richard Westall, Sword of Damocles, 1812 (oil on canvas)

Encryption is prevalent



(Source: https://transparencyreport.google.com/https/overview?hl=en)

E2e encryption debate: a summary

Get off my back you snoop! End-to-end encryption 4ever!

Hacktivist/privacy-sensitive user

If you do not have anything to hide, sure you don't mind me taking a look…

Network administrator/law enforcement officer

The e2 encryption debate /3

- What is the **problem** here?
  - On one hand, the public has an **understandable expectation** that their **communication remains private**
    - End-to-end accomplishes this
  - On the other hand, **network security tools** have traditionally relied on the **ability to inspect communications** (signatures, protocol parsers, etc.)
- The two goals (privacy & security) are – to an extent – at odd with each other!

The case for e2e encryption

- Intercepting network traffic is not all that difficult
- People use Internet for many **sensitive tasks**, such as personal relationships and banking
- Risks:
  - Privacy violations
  - Impersonation/account compromission
    - Authentication itself is not enough!
    - E.g., sidejacking attack

Does this apply to every scenario?

- Can you think of **situations where access to network traffic may make sense**?
- Hint:
  - No expectation of communication secrecy and/or,
  - Secrecy does not benefit the user

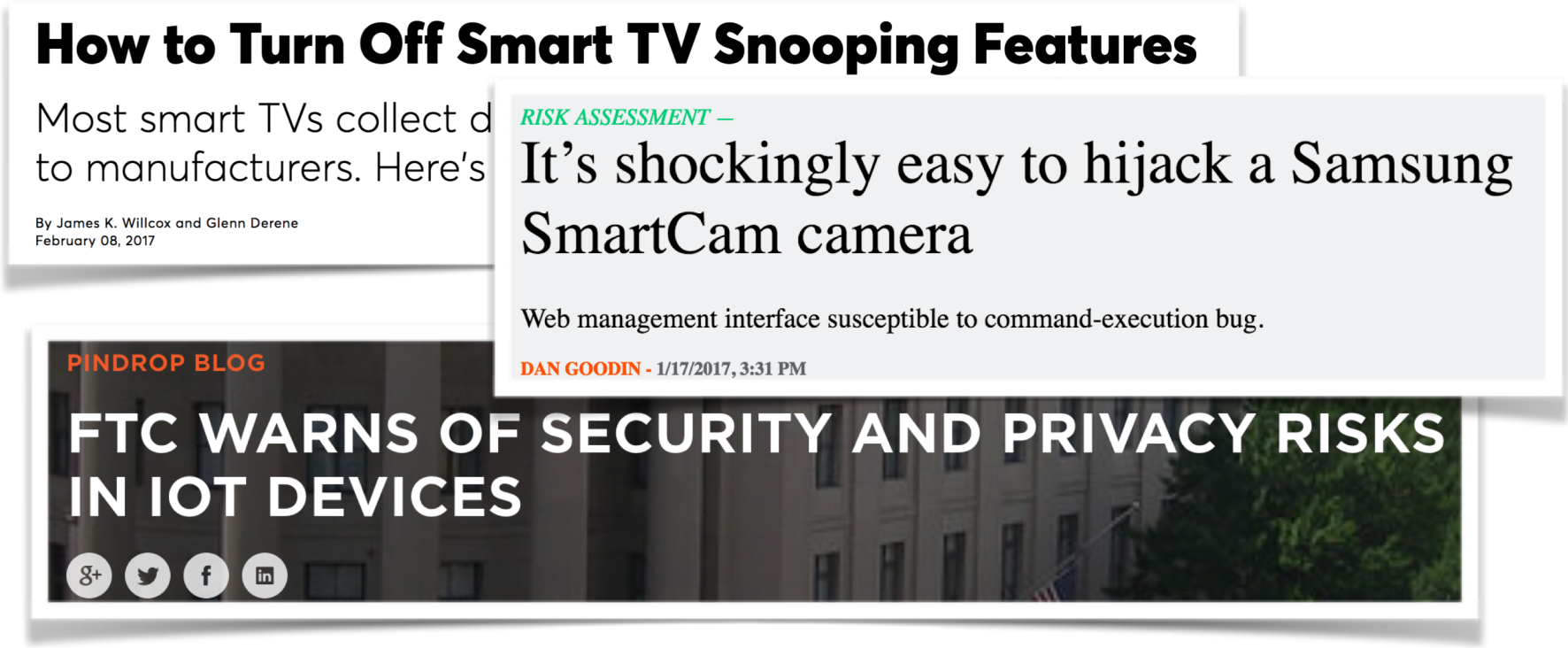
Case #1: no exp. of secrecy

- Workplace (to an extent)

- Military/defense contractor network

Case #2: wrong type of secrecy

- IoT/smart devices



- Network traffic != user activity

Possible solutions /1

- **Interception**
  - Currently require hacks
  - TLS 1.3 makes it even harder
  - Allowing it at protocol level would require new protocols
- Computation on encrypted data
  - For example, string search

TLS 1.3 and security

- TLS 1.2 supports static RSA and Diffie-Hellman
  - Can pre-share key w/ interception box
  - No root certificates etc.
  - TLS 1.3 does not support them anymore
- TLS 1.2 sends SNI field in cleartext
  - “Server name indicator”: domain name the client wishes to contact
  - TLS 1.3 encrypts SNI

Possible solutions/2

- Experimental/academic:
  - “Interception-friendly” TLS: allows user to selectively specify who can intercept traffic and which parts of the traffic can be decrypted
  - Intrusion detection on encrypted data: alternate encoding of traffic (e.g., sequence of encrypted tokens)
    - Allows simple analyses to be performed
    - Difficult to do anything more than keyword search

Possible solutions/3

- Ignore **packet payloads**/focus on **other features**
- Historically, the first form of traffic filtering
- **Firewalling** based on header field values
  - E.g., block ports from/to specific ports and addresses
- Can you think why it would not work?
  - Many traditional application-level protocols (e.g., FTP) now **run on top of HTTP/HTTPS**
  - **Widespread use of CDNs and cloud services** make IP addresses less significative

Possible solutions/4

- Extract features from **flow metadata** (packet lengths, inter-packet timing, etc.)
- Train **machine learning algorithms** to recognize **activity of interest**
  - More powerful than header-based rules
  - Does not need access to decrypted payloads

Let’s talk about applications of ML to network traffic analysis

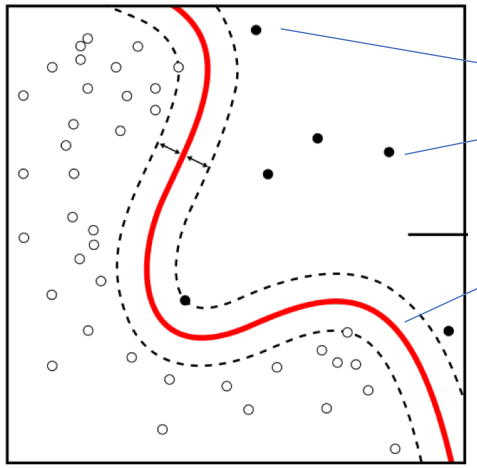
Traffic classification

- In general, the act of **mapping network flows to classes of interest**
- Several ways to do it, but the most popular is to **use statistical and/or machine learning techniques**
- **Supervised:** detect known types of traffic of interest (assigned reading)
  - Needs labeled samples

Anomaly detection

- Application of statistical and/or machine learning techniques to **identify unexpected network behavior**
- Typically **unsupervised/semi-supervised**
- Goal: detect “anything out of the ordinary”
- Detect **flows that deviate from expected behavior of system**
- Needs definition of “**normal behavior**”

ML refresh



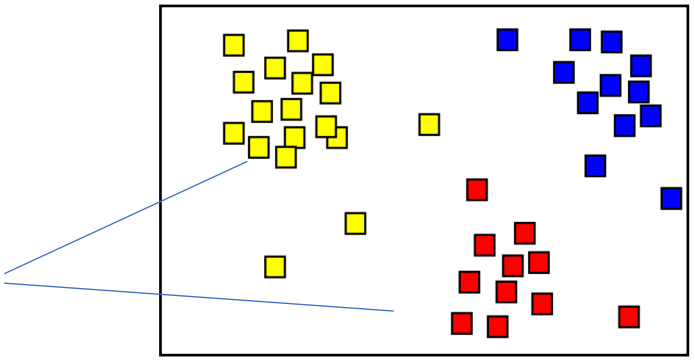
https://commons.wikimedia.org/w/index.php?curid=47868867

Classification (supervised):

- Convert flows to points (vectors) in **feature space**
- Decide to which class a vector begin using a decision boundary determined during training

Anomaly detection (unsupervised/semi-supervised):

- Convert entities to points (vectors) in **feature space**
- Identify different classes of flows that lie far from “normal flows”
- Use clustering, 1-class classification, etc.



ML refresh/2

- A whole host of algorithms have been proposed/used for classification problems
  - SVM, random forests, neural networks, …
  - For the purpose of this lecture, you can think of them as a black box that takes a point and maps it to a class
- Similarly for anomaly detections
  - For the purpose of this lecture, you can think of them as black boxes that identify sets of “strange” points
    - “Strange” == “lie far away from points representing normal flows”

Can you explain me why traffic classification != anomaly detection?

A bit of history/1

- **Portnoy et al., 2001:** features derived from TCP flow metadata (duration, #bytes, etc.)
  - **Single-linkage clustering** to identify classes of traffic
- **PAYL (Stolfo et al., 2004):** uses simple statistics
  - Compute **mean, std of frequency of all possible 256 byte values** in normal traffic
  - Samples w/ **values that significantly differ from normal traffic** are flagged

A bit of history/2

- **Poseidon (Bolzoni et al., 2006):** like PAYL but uses self-organizing maps to pre-process features
- **McPad (Perdisci et al., 2009):** ensemble of multiple 1-class classifiers
  - Anomaly detector
  - Features: frequency distribution of byte n-grams
  - Trains multiple one-class SVMs and combines their output
- **Many others!**

A bit of history/3

- **Commercial approaches**?
  - Sure, lots of them
  - Not many details on their inner workings

- In general, many past approaches are now **outdated by their assumptions**, and/or generate **high false positives**

What about traffic classification?

- There has been work, but until HTTPS became prevalent there were easier ways to do it
- Can you think of some?
  - IP/Port-based
  - Regexp-based
  - Protocol parsing

Modern times

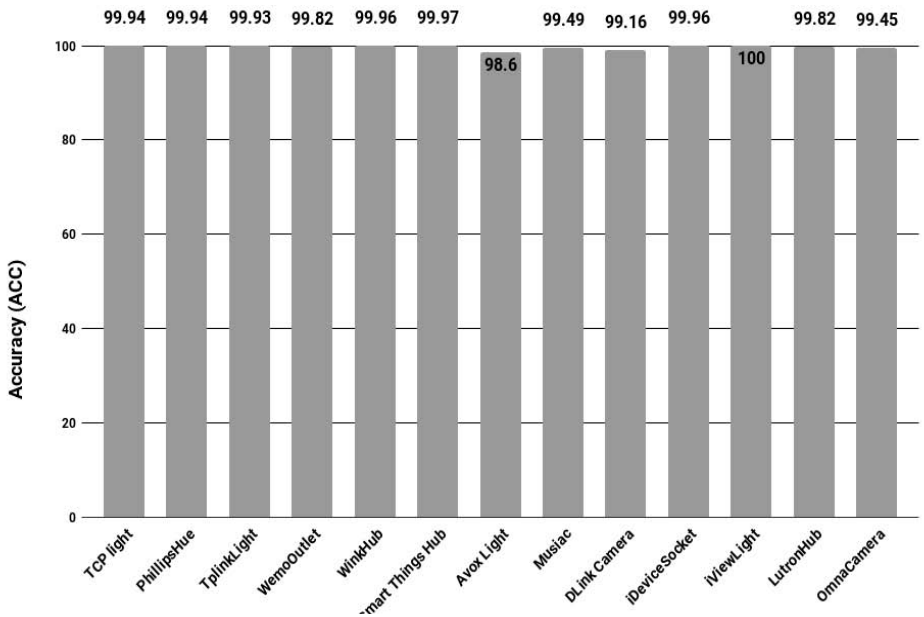
- Why do we suddenly care more about **machine learning in networking**?
  - **Encryption** has made understanding traffic harder
  - Also, now we have:
    - **More data**
    - **More powerful learning techniques**

“Things” you may want to classify

- **Flows**, but not just that!
  - **Users**
  - **Websites**
  - **Device** make/model/OS
- Each goal requires different features/algorithmic tuning

Example 1: IoT fingerprinting

- Bezawada et al., “IoTSense: Behavioral Fingerprinting of IoT Devices” (2018)
- Goal: identify make/model of IoT device from traffic
  - Traffic classification problem (flow->device)
- Approach:
  - Features: layer-III/IV/VII protocol type; TCP payload length/entropy; TCP window size
  - Algorithm: gradient boosting



Example 2: web browsing analysis

- Herrmann et al., "Website fingerprinting", 2009
- Goal: identify website generating encrypted traffic
- Approach:
  - Feature: frequency distribution of packet sizes
  - Algorithm: Naïve Bayes (with some tricks)

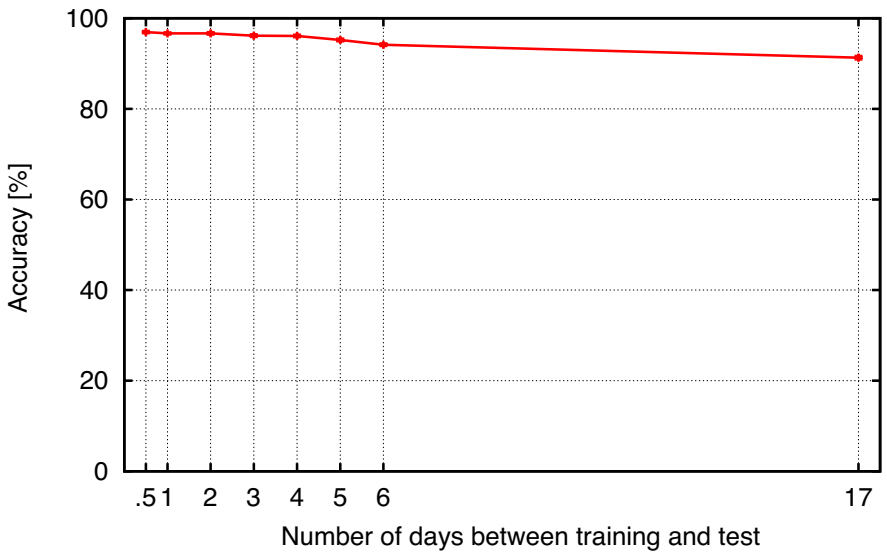


Figure 5: Accuracy for various time spans between training and test

Some common problems

- For anomaly detection, “normal behavior” is a slippery concept
  - **Users change habits**
  - Getting a **clean baseline** is difficult
    - How do you guarantee that training data is free of malicious traffic?
- Output should be **produced quickly** in order to be useful
  - But datasets are huge

Problems/2

- For some attacks, **correlation over long timescales** is necessary
  - But who has memory/power to do that?
- **False positives** still a problem
  - The curse of rare events (AKA “base-rate fallacy”)
- Further readings:
  - https://techcrunch.com/2016/02/29/machine-learning-is-not-the-answer-to-better-network-security/
  - http://news.bbc.co.uk/2/hi/uk_news/magazine/8153539.stm

Problems/3

- Dataset generation
  - Supervised classification requires **labeled data**
  - Generating enough samples may be difficult
  - Device fingerprinting particularly problematic
  - **Getting clean data** may be difficult
- Feature selection
  - Are characteristics of the traffic used for classification **really significant**? Or just an **accident of the dataset?**

Let’s talk about today’s reading

Android app fingerprinting

- Another **classification problem**
- Can you think of concrete applications?
  - Identify apps (good & bad)
  - Identify specific users from app set (good & bad)
  - Network management & engineering (good)
  - Marketing research (bad ;-))

Why is this different?

- The closest operation is **website fingerprinting**…
- …but that can leverage:
  - IP addresses (to an extent)
  - Flow size distribution, who tends to be website-specific
- Can’t do that here!
  - Can you explain why?

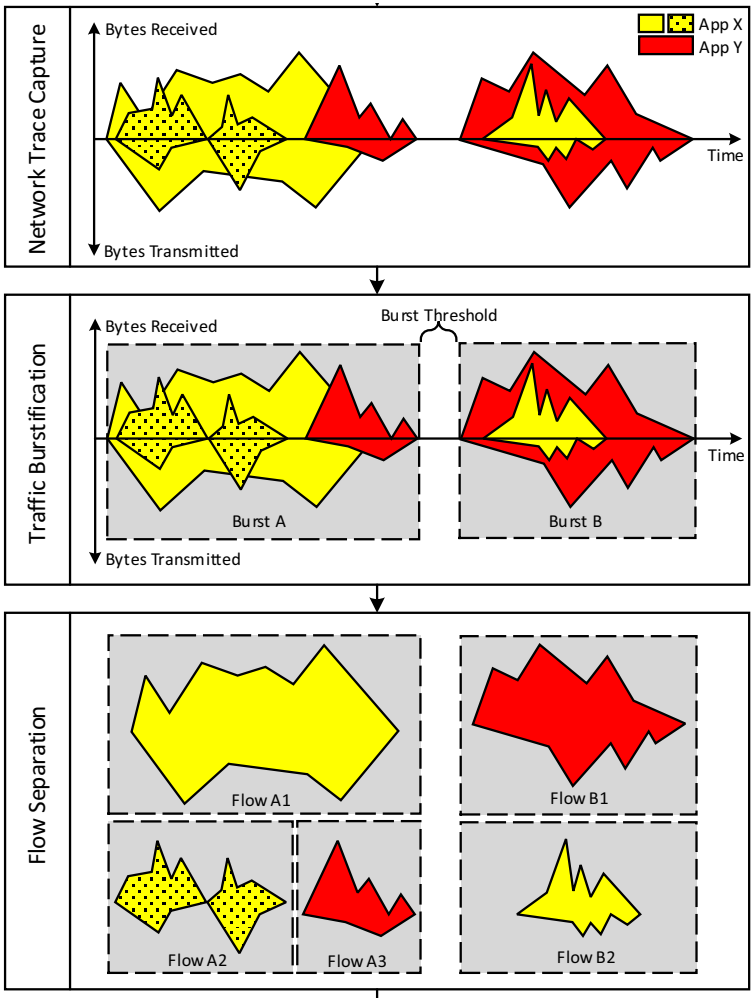
Problems

- Want to do this in **near-real time**
  - But app sessions can be long
  - Certain approaches are computationally expensive

- **Cannot rely on payload data**
  - Can’t rely on IP addresses either
  - Why?

Approach

- **“Burstification”**
  - Look at flows, chop them when there is a long enough silence (1s)
- Keep sample size **small** (only a few packets per classifier decision)
- Allows **timely classification**



Approach/2

- Features: **packet lengths**
  - Either individual, combined into feature vectors…
  - Or statistical properties of a sequence of packet lengths
- Classifier(s):
  - **Support Vector Machines**
  - **Random Forests** (seem to work better)
- Approaches:
  - **Closed-world classifiers** (map flows to one of many possible apps)
  - **Open-world classifiers** (map flows to either a specific app or “anything else”)

Approach/3

- **Tricks:**
  - **Drop TCP errors** (retransmissions etc.)
    - Prevent noise from seeping into training/evaluation data
  - **Truncate long flows**
  - **Discard very short flows** (incomplete conversations are not meaningful)
  - **Discard low-confidence classifier decisions**

Results

TABLE 5: Table summarising multi-class classifier performance when classification validation is used.

Classification Approach	**Prediction Probability Threshold**		
Per flow SVC	**0.5**	**0.7**	**0.9**
Precision	95.1%	97.2%	99.7%
Recall	92.4%	92.3%	90.4%
Accuracy	95.0%	99.1%	99.6%
% Flows classified	45.5%	26.0%	3.4%
Per flow Random Forest	**0.5**	**0.7**	**0.9**
Precision	95.4%	98.1%	99.5%
Recall	95.0%	98.0%	99.4%
Accuracy	94.7%	98.3%	99.5%
% Flows classified	80.6%	71.3%	62.3%
Single Large SVC	**0.5**	**0.7**	**0.9**
Precision	79.8%	79.7%	65.0%
Recall	73.2%	70.2%	66.5%
Accuracy	89.6%	91.1%	93.0%
% Flows classified	5.9%	2.1%	0.7%
Single Large Random Forest	**0.5**	**0.7**	**0.9**
Precision	95.9%	98.4%	99.7%
Recall	94.9%	98.0%	99.6%
Accuracy	95.2%	98.2%	99.6%
% Flows classified	86.5%	79.4%	72.0%