Adventures in Creating a Cybersecurity Dataset

Heather Lawrence

@infosecanon

Nebraska Applied Research Institute

About Me

- Data Scientist with NARI
- PhD student @ UCCS
- 6 Years USN
- Hack@UCF, NCCDC
- B-Sides Orlando Board Member, VetSec Board Member, DEF CON Goon, and Kernelcon volunteer

NEBRASKA APPLIED RESEARCH INSTITUTE
at the University of Nebraska

Outline

- Best Practices
- Motivation (Why we did the thing)
- Design (What we did)
- Challenges (Why this talk is labeled as an ‘adventure’)
- Resources
- Outro

Best Practices (Gharib et al 2016)

- Complete network configuration
- Complete traffic
- Labelled dataset
- Complete interaction
- Complete capture
- Available protocols
- Attack diversity
- Heterogeneity
- Feature set
- Metadata

Best Practices (Shiravi et al)

- Guidelines to creating your own dataset (Shiravi et al):
  - Up-to-date network-based data and protocols
  - Publicly available
  - Real network traffic
  - A variety of malicious and normal user behavior
  - Payload

Motivation – Why we did the thing

- Intrusion detection used to rely on signatures
- But signatures can be changed by changing trivial parts of the attack (generally the payload)
- Machine learning will save us!
- Wait… adversarial machine learning is a thing
- Several authors have complained about a lack of usable data (Sommer and Paxson 2010) as late as 2017
- Can’t test IDS-specific machine learning algorithms without usable data
    - Can’t compare results unless the data is open access

Brief Timeline in Adversarial ML

2004
“Adversarial Classification”
Dalvi, et al

2011
“Adversarial Machine Learning”
Huang, et al.

2012
“Poisoning attacks against SVMs”
Biggio, et al.

2014
“Intriguing properties of neural networks”
Szegedy, et al.

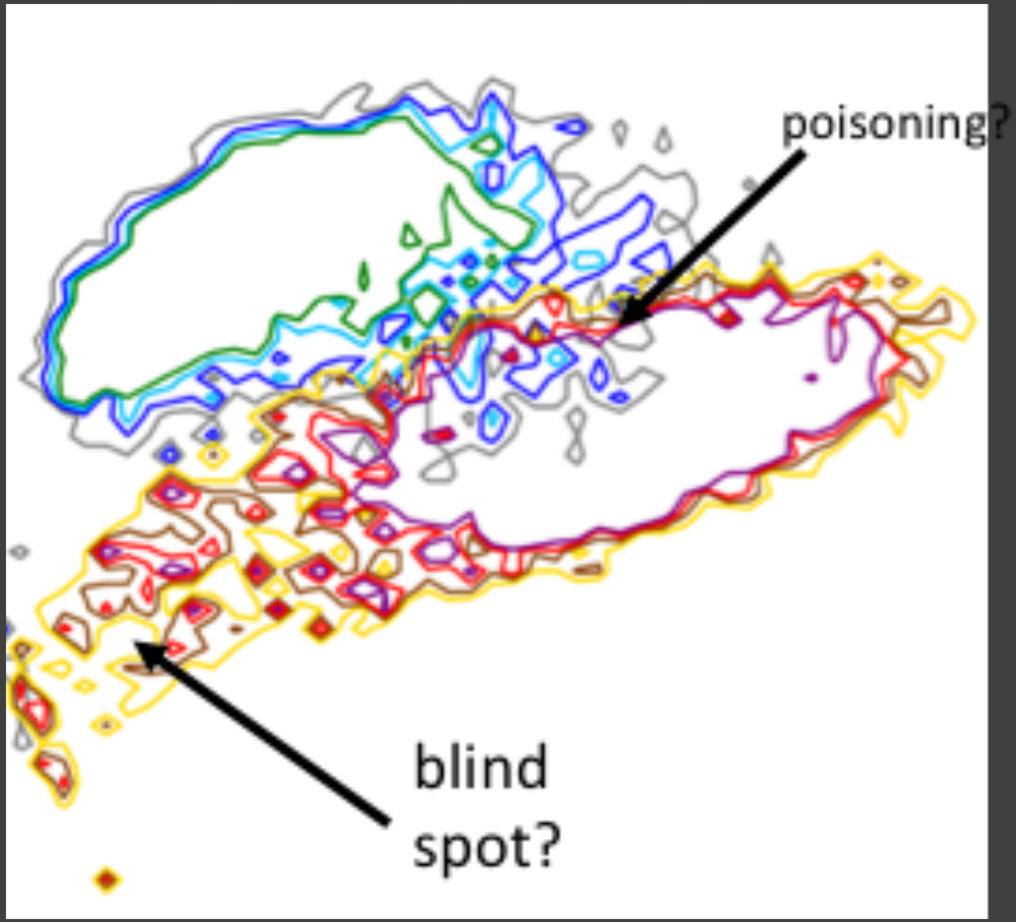
2016
“Transferability in ML: from phenomena to black-box attacks using adv samples”
Papernot, et al.

Types of Attacks

- Causative
  - Manipulation of training data
- Data Poisoning
  - Specially crafted attack points are injected into the training data
- Exploratory
  - Exploit the classifier itself
- Hybrid
  - A combination of the aforementioned

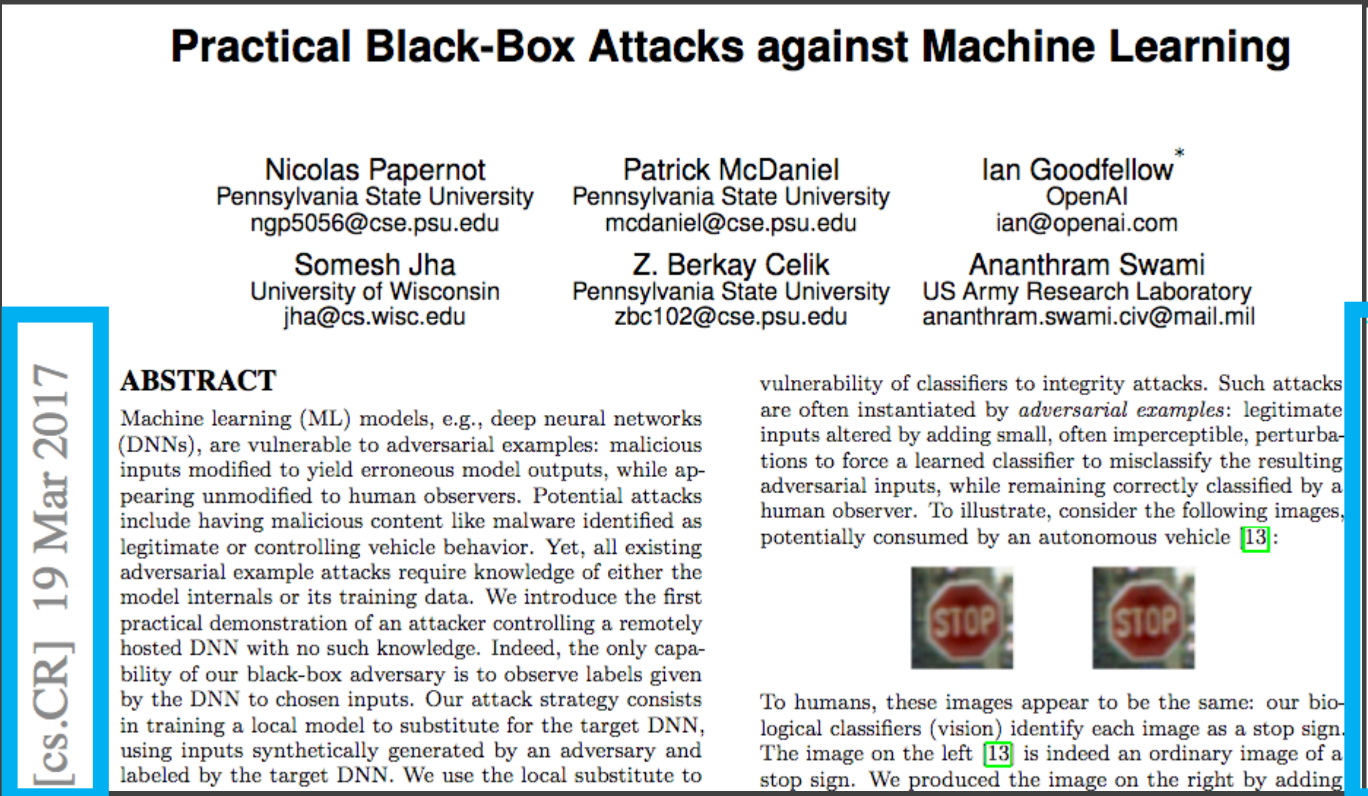
Blind Spots

- Regions in a model's decision space where the decision boundary is inaccurate
- Reason: No training data was provided
- Ongoing research area



Tully and Anderson, *Navigating the Labeling Bottleneck as Security Embraces AI,* RSA Conference 2018

Computer Vision vs. Intrusion Detection



[cs.CR] 19 Mar 2017

Practical Black-Box Attacks against Machine Learning

Nicolas Papernot
Pennsylvania State University
ngp5056@cse.psu.edu

Patrick McDaniel
Pennsylvania State University
mcdaniel@cse.psu.edu

Ian Goodfellow*
OpenAI
ian@openai.com

Somesh Jha
University of Wisconsin
jha@cs.wisc.edu

Z. Berkay Celik
Pennsylvania State University
zbc102@cse.psu.edu

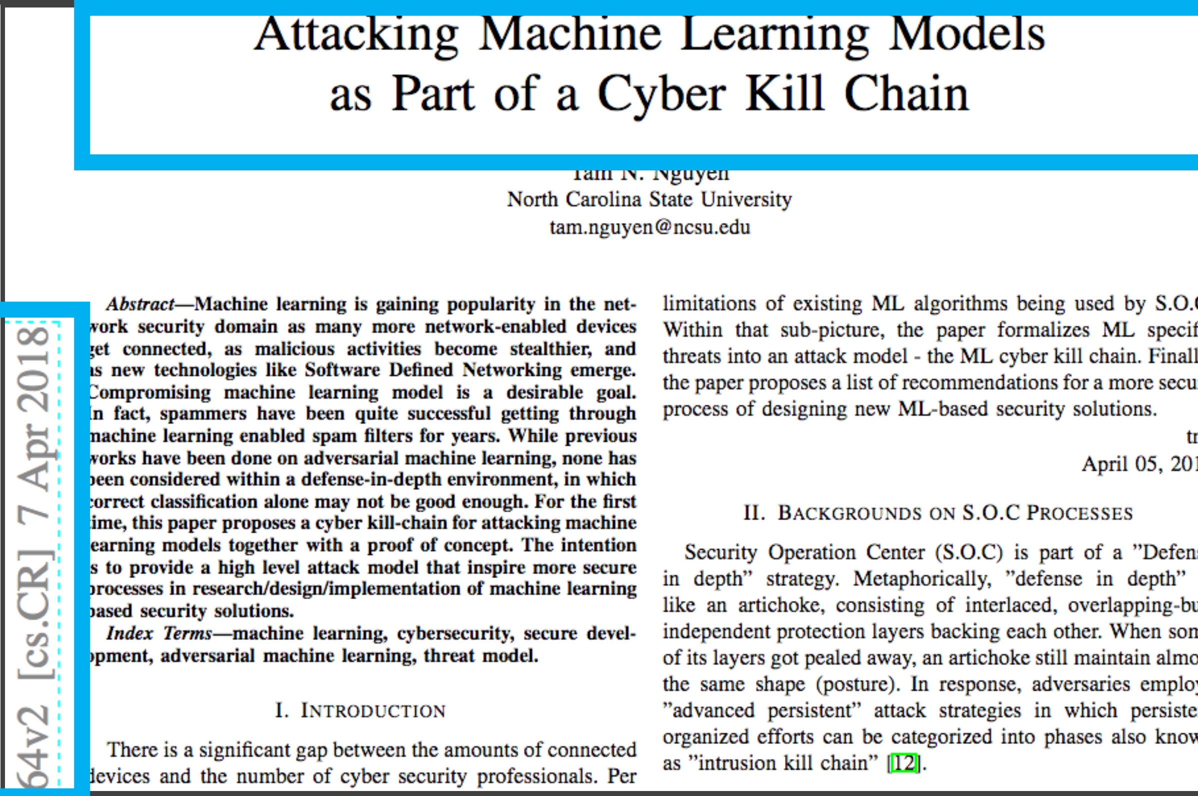
Ananthram Swami
US Army Research Laboratory
ananthram.swami.civ@mail.mil

ABSTRACT

Machine learning (ML) models, e.g., deep neural networks (DNNs), are vulnerable to adversarial examples: malicious inputs modified to yield erroneous model outputs, while appearing unmodified to human observers. Potential attacks include having malicious content like malware identified as legitimate or controlling vehicle behavior. Yet, all existing adversarial example attacks require knowledge of either the model internals or its training data. We introduce the first practical demonstration of an attacker controlling a remotely hosted DNN with no such knowledge. Indeed, the only capability of our black-box adversary is to observe labels given by the DNN to chosen inputs. Our attack strategy consists in training a local model to substitute for the target DNN, using inputs synthetically generated by an adversary and labeled by the target DNN. We use the local substitute to

vulnerability of classifiers to integrity attacks. Such attacks are often instantiated by *adversarial examples*: legitimate inputs altered by adding small, often imperceptible, perturbations to force a learned classifier to misclassify the resulting adversarial inputs, while remaining correctly classified by a human observer. To illustrate, consider the following images, potentially consumed by an autonomous vehicle [13]:

To humans, these images appear to be the same: our biological classifiers (vision) identify each image as a stop sign. The image on the left [13] is indeed an ordinary image of a stop sign. We produced the image on the right by adding



64v2 [cs.CR] 7 Apr 2018

Attacking Machine Learning Models as Part of a Cyber Kill Chain

Tam N. Nguyen
North Carolina State University
tam.nguyen@ncsu.edu

***Abstract*—Machine learning is gaining popularity in the network security domain as many more network-enabled devices get connected, as malicious activities become stealthier, and as new technologies like Software Defined Networking emerge. Compromising machine learning model is a desirable goal. In fact, spammers have been quite successful getting through machine learning enabled spam filters for years. While previous works have been done on adversarial machine learning, none has been considered within a defense-in-depth environment, in which correct classification alone may not be good enough. For the first time, this paper proposes a cyber kill-chain for attacking machine learning models together with a proof of concept. The intention is to provide a high level attack model that inspire more secure processes in research/design/implementation of machine learning based security solutions.**

***Index Terms*—machine learning, cybersecurity, secure development, adversarial machine learning, threat model.**

I. Introduction

There is a significant gap between the amounts of connected devices and the number of cyber security professionals. Per

limitations of existing ML algorithms being used by S.O. Within that sub-picture, the paper formalizes ML specif threats into an attack model - the ML cyber kill chain. Finall the paper proposes a list of recommendations for a more secu process of designing new ML-based security solutions.

April 05, 201

II. Backgrounds on S.O.C Processes

Security Operation Center (S.O.C) is part of a "Defen in depth" strategy. Metaphorically, "defense in depth" like an artichoke, consisting of interlaced, overlapping-bu independent protection layers backing each other. When son of its layers got pealed away, an artichoke still maintain almo the same shape (posture). In response, adversaries employ "advanced persistent" attack strategies in which persiste organized efforts can be categorized into phases also know as "intrusion kill chain" [12].

Motivation

- Intrusion detection used to rely on signatures
- But signatures can be changed by changing trivial parts of the attack (generally the payload)
- Machine learning will save us!
- Wait… adversarial machine learning is a thing
- Several authors have complained about a lack of usable data (Sommer and Paxson 2010) as late as 2017
- Can’t test IDS-specific machine learning algorithms without usable data
  - Can’t compare results unless the data is open access

Motivation

- Are there datasets out there that do this?

Related Work – KDD99

- Most cited dataset (also the oldest – 1999)
- Dataset was created by monitoring a simulated Air Force network for weeks
- Simulated dataset that doesn't reflect current attack techniques or methodologies
- Don't contain real packet headers or data

- Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. Analysis and results of the 1999 DARPA off-line intrusion detection evaluation. pages 162–182, 10 2000.

Related Work – SSH Attacks

- Dataset consisting of University of Twente campus network traffic (100 servers, workstations, and honeypots)
- Attacks and detections limited to SSH
- Contained flow data and host log files

- Rick Hofstede, Luuk Hendriks, Anna Sperotto, and Aiko Pras. SSH compromise detection using netflow/ip-fix. ACM SIGCOMM computer communication review, 44(5):20–26, 2014.

Related Work – UNSW-NB15

- Used IXIA PerfectStorm tool to generate nine families of attacks
- Traffic captured using tcpdump, distilled into netflows using Argus, and analyzed using Bro-IDS (now known as Zeek)
- Attack labels are generated programmatically using the IXIA tool
- 49 features, protocols include HTTP, FTP

- Nour Moustafa and Jill Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 military communications and information systems conference (MilCIS), pages 1–6. IEEE, 2015.

Related Work – AWID

- The Aegean Wi-Fi Intrusion Dataset is a curated 802.11 collection containing wireless benign and attack traffic
  - Attacks are tool generated
- Normal traffic is human generated
- Used Kali Linux to conduct penetration testing and Wireshark to log traffic

- Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Stefanos Gritzalis. Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset. IEEE Communications Surveys & Tutorials, 18(1):184–208, 2015.

Related Work – CTU-13

- Collection of 13 pcaps focused on botnet traffic
- Paper introduces a method of detecting botnet traffic (BotHunter)

- Garcia, Sebastian, et al. "An empirical comparison of botnet detection methods." *computers & security 45* (2014): 100-123.

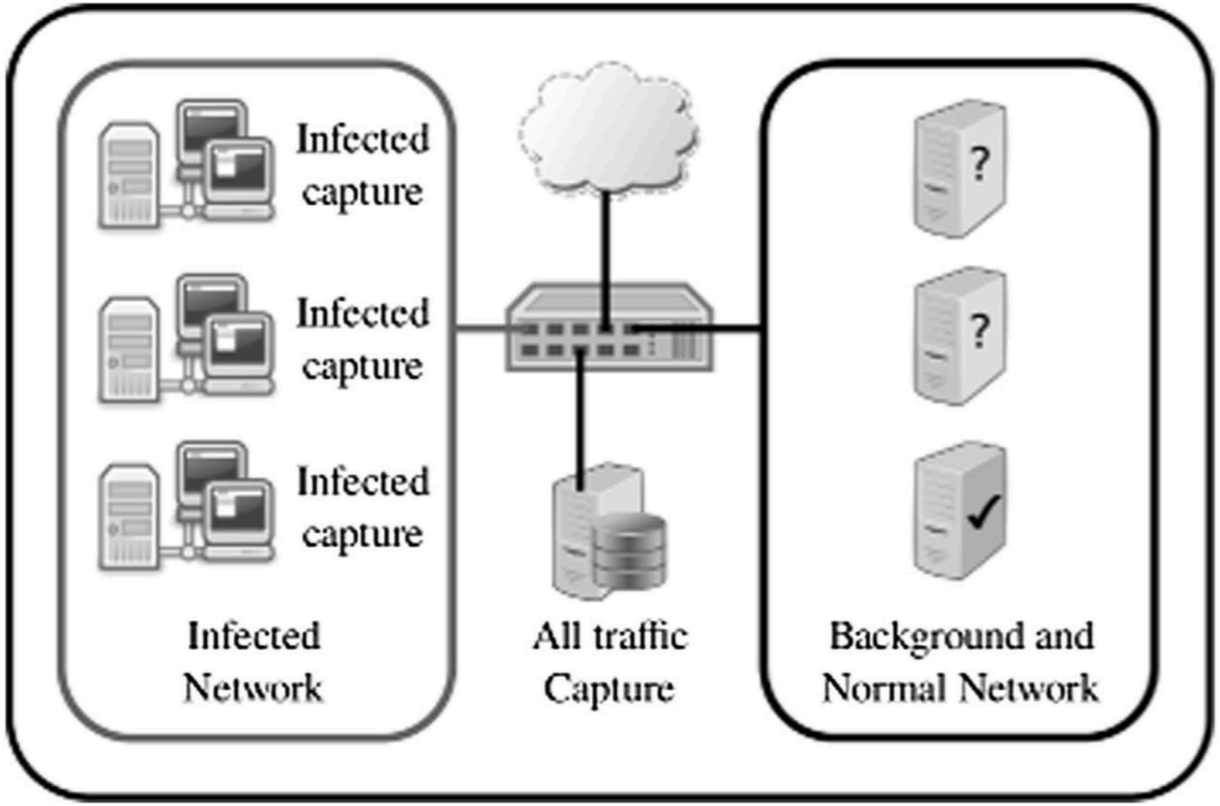


Fig. 3 — Testbed network topology.

Related Work – CICIDS 2017

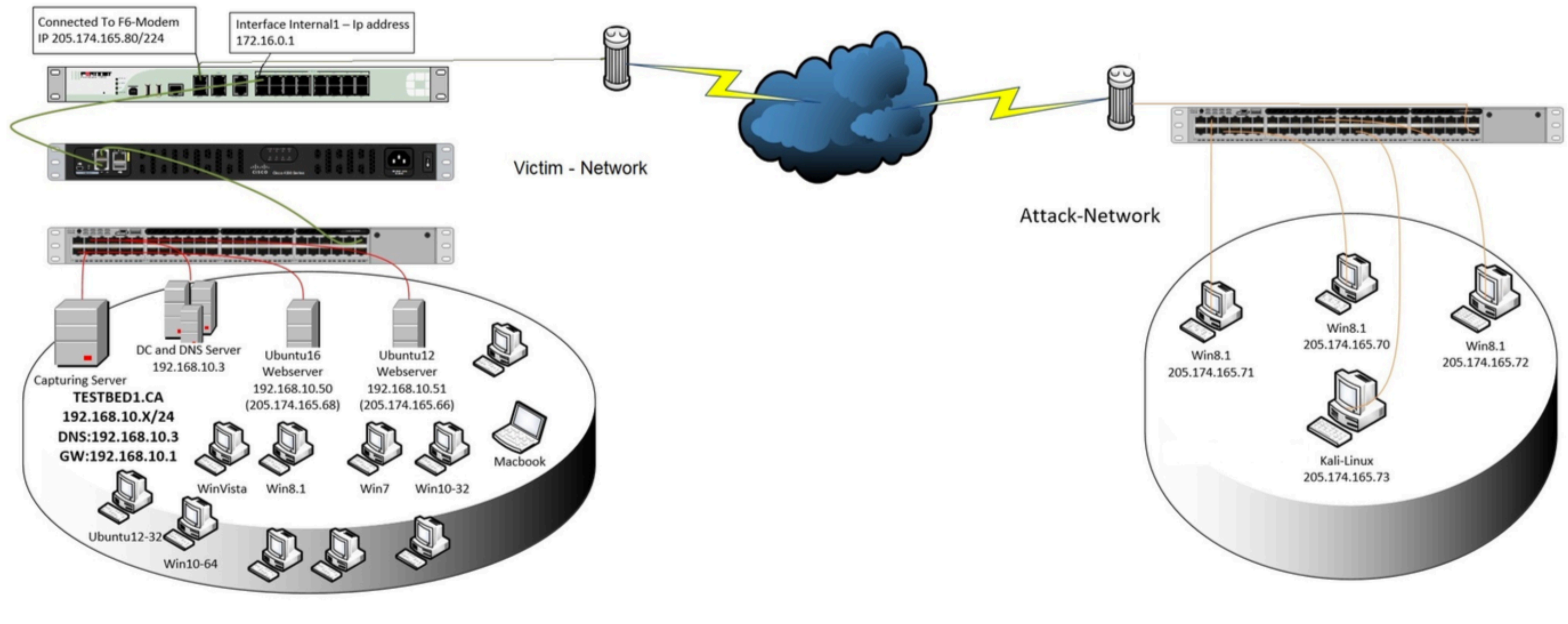


Figure 1: Testbed Architecture.

Sharafaldin, Iman, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward generating a new intrusion detection dataset and intrusion traffic characterization." ICISSP. 2018.

Motivation

- Are there datasets out there that do this?
  - Not really

Related Work - Datasets

- Thorough survey of network intrusion dataset papers (Ring et al)

- Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. A survey of network-based intrusion detection data sets. Computers & Security, 2019.
- https://arxiv.org/pdf/1903.02460.pdf

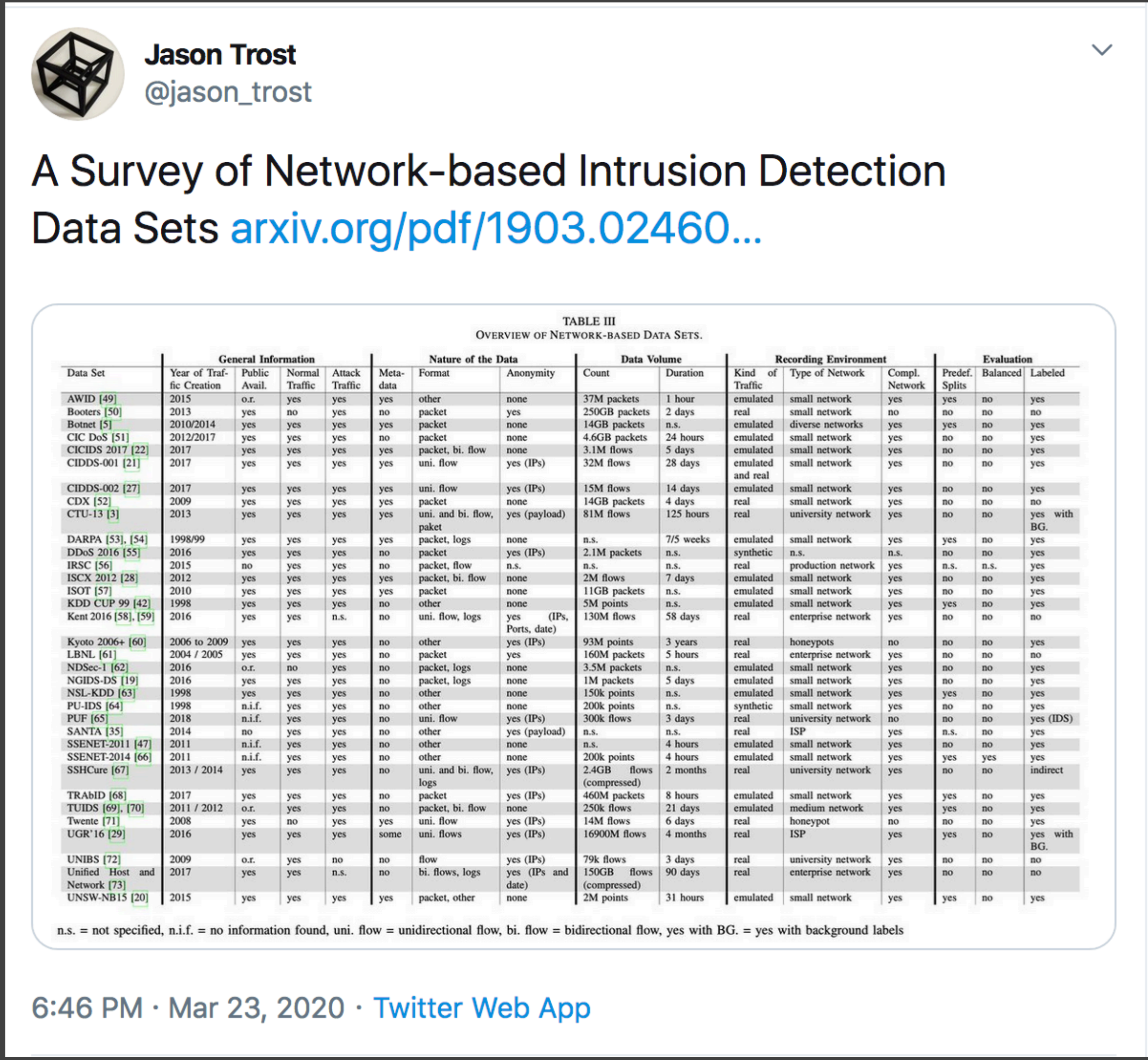


TABLE III
OVERVIEW OF NETWORK-BASED DATA SETS.

	General Information				Nature of the Data			Data Volume		Recording Environment			Evaluation		
Data Set	Year of Traffic Creation	Public Avail.	Normal Traffic	Attack Traffic	Meta-data	Format	Anonymity	Count	Duration	Kind of Traffic	Type of Network	Compl. Network	Predef. Splits	Balanced	Labeled
AWID [49]	2015	o.r.	yes	yes	yes	other	none	37M packets	1 hour	emulated	small network	yes	yes	no	yes
Booters [50]	2013	yes	no	yes	no	packet	yes	250GB packets	2 days	real	small network	no	no	no	no
Botnet [5]	2010/2014	yes	yes	yes	yes	packet	none	14GB packets	n.s.	emulated	diverse networks	yes	yes	no	yes
CIC DoS [51]	2012/2017	yes	yes	yes	no	packet	none	4.6GB packets	24 hours	emulated	small network	yes	no	no	yes
CICIDS 2017 [22]	2017	yes	yes	yes	yes	packet, bi. flow	none	3.1M flows	5 days	emulated	small network	yes	no	no	yes
CIDDS-001 [21]	2017	yes	yes	yes	yes	uni. flow	yes (IPs)	32M flows	28 days	emulated and real	small network	yes	no	no	yes
CIDDS-002 [27]	2017	yes	yes	yes	yes	uni. flow	yes (IPs)	15M flows	14 days	emulated	small network	yes	no	no	yes
CDX [52]	2009	yes	yes	yes	yes	packet	none	14GB packets	4 days	real	small network	yes	no	no	no
CTU-13 [3]	2013	yes	yes	yes	yes	uni. and bi. flow, paket	yes (payload)	81M flows	125 hours	real	university network	yes	no	no	yes with BG.
DARPA [53], [54]	1998/99	yes	yes	yes	yes	packet, logs	none	n.s.	7/5 weeks	emulated	small network	yes	yes	no	yes
DDoS 2016 [55]	2016	yes	yes	yes	no	packet	yes (IPs)	2.1M packets	n.s.	synthetic	n.s.	n.s.	no	no	yes
IRSC [56]	2015	no	yes	yes	no	packet, flow	n.s.	n.s.	n.s.	real	production network	yes	n.s.	n.s.	yes
ISCX 2012 [28]	2012	yes	yes	yes	yes	packet, bi. flow	none	2M flows	7 days	emulated	small network	yes	no	no	yes
ISOT [57]	2010	yes	yes	yes	yes	packet	none	11GB packets	n.s.	emulated	small network	yes	no	no	yes
KDD CUP 99 [42]	1998	yes	yes	yes	no	other	none	5M points	n.s.	emulated	small network	yes	yes	no	yes
Kent 2016 [58], [59]	2016	yes	yes	n.s.	no	uni. flow, logs	yes (IPs, Ports, date)	130M flows	58 days	real	enterprise network	yes	no	no	no
Kyoto 2006+ [60]	2006 to 2009	yes	yes	yes	no	other	yes (IPs)	93M points	3 years	real	honeypots	no	no	no	yes
LBNL [61]	2004 / 2005	yes	yes	yes	no	packet	yes	160M packets	5 hours	real	enterprise network	yes	no	no	no
NDSec-1 [62]	2016	o.r.	no	yes	no	packet, logs	none	3.5M packets	n.s.	emulated	small network	yes	no	no	yes
NGIDS-DS [19]	2016	yes	yes	yes	no	packet, logs	none	1M packets	5 days	emulated	small network	yes	no	no	yes
NSL-KDD [63]	1998	yes	yes	yes	no	other	none	150k points	n.s.	emulated	small network	yes	yes	no	yes
PU-IDS [64]	1998	n.i.f.	yes	yes	no	other	none	200k points	n.s.	synthetic	small network	yes	no	no	yes
PUF [65]	2018	n.i.f.	yes	yes	no	uni. flow	yes (IPs)	300k flows	3 days	real	university network	no	no	no	yes (IDS)
SANTA [35]	2014	no	yes	yes	no	other	yes (payload)	n.s.	n.s.	real	ISP	yes	n.s.	no	yes
SSENET-2011 [47]	2011	n.i.f.	yes	yes	no	other	none	n.s.	4 hours	emulated	small network	yes	no	no	yes
SSENET-2014 [66]	2011	n.i.f.	yes	yes	no	other	none	200k points	4 hours	emulated	small network	yes	yes	yes	yes
SSHCure [67]	2013 / 2014	yes	yes	yes	no	uni. and bi. flow, logs	yes (IPs)	2.4GB flows (compressed)	2 months	real	university network	yes	no	no	indirect
TRAbID [68]	2017	yes	yes	yes	no	packet	yes (IPs)	460M packets	8 hours	emulated	small network	yes	yes	no	yes
TUIDS [69], [70]	2011 / 2012	o.r.	yes	yes	no	packet, bi. flow	none	250k flows	21 days	emulated	medium network	yes	yes	no	yes
Twente [71]	2008	yes	no	yes	yes	uni. flow	yes (IPs)	14M flows	6 days	real	honeypot	no	no	no	yes
UGR'16 [29]	2016	yes	yes	yes	some	uni. flows	yes (IPs)	16900M flows	4 months	real	ISP	yes	yes	no	yes with BG.
UNIBS [72]	2009	o.r.	yes	no	no	flow	yes (IPs)	79k flows	3 days	real	university network	yes	no	no	no
Unified Host and Network [73]	2017	yes	yes	n.s.	no	bi. flows, logs	yes (IPs and date)	150GB flows (compressed)	90 days	real	enterprise network	yes	no	no	no
UNSW-NB15 [20]	2015	yes	yes	yes	yes	packet, other	none	2M points	31 hours	emulated	small network	yes	yes	no	yes

n.s. = not specified, n.i.f. = no information found, uni. flow = unidirectional flow, bi. flow = bidirectional flow, yes with BG. = yes with background labels

Experimental Setup

- Using the network anomaly detection paradigm…
  - “This traffic is benign”
- Type I - A rejection of the null hypothesis
  - False positive
  - Incorrect classification of benign traffic as malicious traffic
  - Increases operator fatigue
- Type II – A non-rejection of a false null hypothesis
  - False negative
  - Malicious traffic classified as benign
  - Allows malicious traffic on the network

What causes Type I / Type II errors?

True Positive

- Web surfing

True Negative

False Positive

False Negative

What causes Type I / Type II errors?

True Positive

- SQLi

True Negative

False Positive

False Negative

What causes Type I / Type II errors?

True Positive

True Negative

- Web crawling
- User actions

False Positive

False Negative

What causes Type I / Type II errors?

True Positive

True Negative

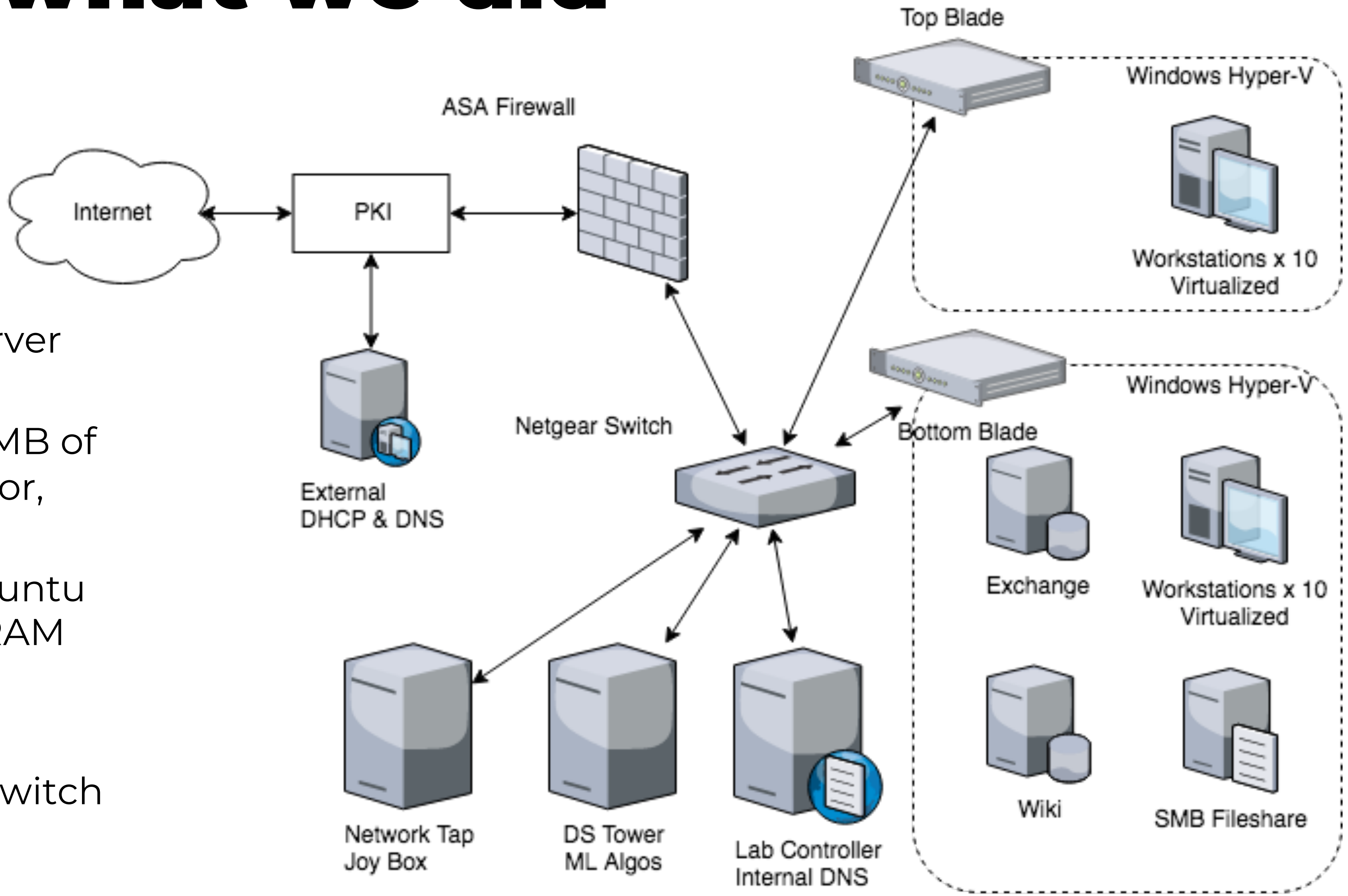
- Nmap
- Directory Traversal
- Delivery of shell

False Positive

False Negative

Design – what we did

- Used 3x server blades running Windows Server 2016
- Each drone had 1024MB of RAM, 1 virtual processor, 40Gb HD
- Network tap used Ubuntu 18.04 LTS with 32Gb RAM and 1TB HD
- ASA 5506 Firewall
- Netgear 24-port pro switch



Automated Data Generation

- Powershell scripts automated 'user' actions
  - 3 profiles: business, admin, engineering
- Randomly:
  - Browse to 30 Azure-hosted mirrors
  - Email another user (using the Exchange server)
  - R/W to a SMB Fileshare
  - Browse to Wiki addresses
- Automated malicious traffic generated by Kali Linux

Human Data Generation

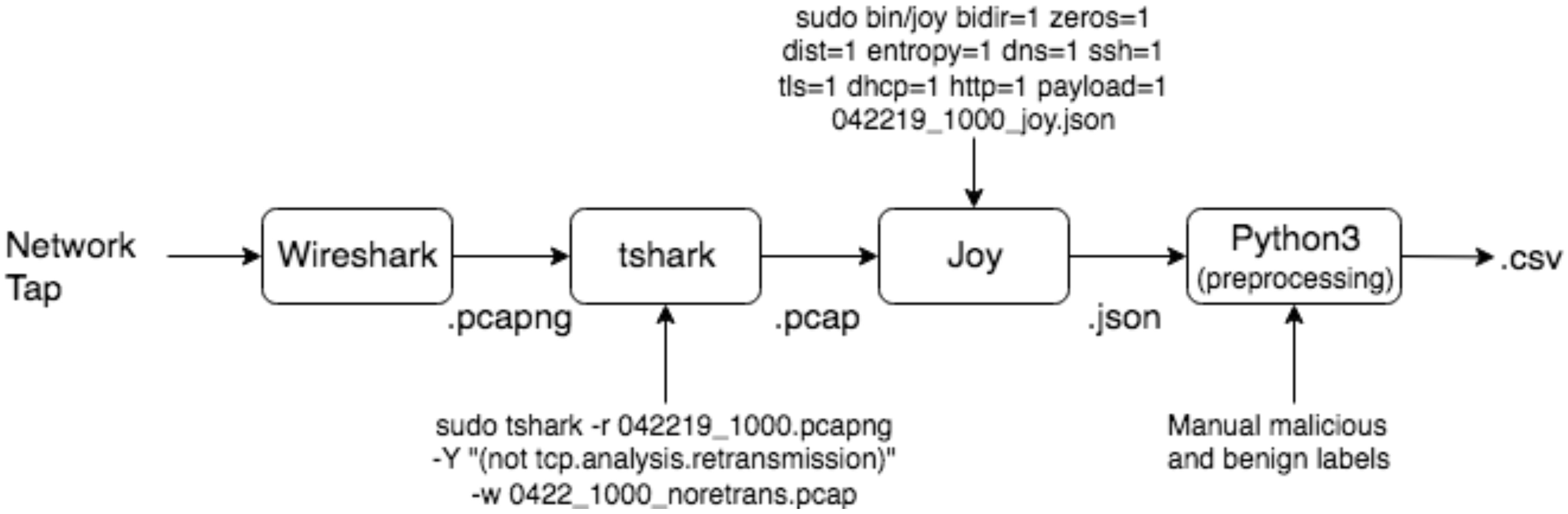
- 10 Human subjects browsed services for 30 minutes
  - Provides a means to compare human-generated benign traffic characteristics
- Malicious traffic was generated by the same human subjects for 1 hour
  - Used Burp Suite to leverage vulns in DVWA
  - Saved us from having to build our own vulnerable web app
  - Described their intent while attacking DVWA
  - Lowered the bar of knowledge for more participation

11.7 P7

Webpage	Wiki	Fileshare	Finding vulns	Using vulns
5	5	5	3	3

- I started my connection to DVWA
- I use the file upload to open a backdoor
- First I want to use weevely. I go to the directory cd /usr/share/weevely
- Run the Python ./weevely.py generate secret my.php. This creates a php script called 'my' with the password of "secret"
- Uploaded this script to DVWA
- Changed the filename in burp from php to jpg to bypass the image filter
- Enabled the back door
- Privileges determined to be nt authority system
- Made myself an account
- Made myself an admin
- Shutdown the box

Data Collection



Data Processing

- Network traffic captures were QA'd after gathering
- Attacks were verified via PCAP review
- Ways to identify the traffic were translated into pandas dataframes rules

```
def malicious(x):
# If traffic was from 10.10.10.4 AND the
protocol was ICMP, it’s malicious
if df.loc[df[’pr’] == 1.0]: df[’label’] == 1
```

Data Processing

- IPs were translated to the service provider that owns the space

```
if (ip.is_private):
    return 'private'
if ip in ipaddress.ip_network('137.48.0.0/16'):
    return 'UNO PKI'
if ip in ipaddress.ip_network('52.0.0.0/11'):
    return 'Amazon.com, Inc.'
```

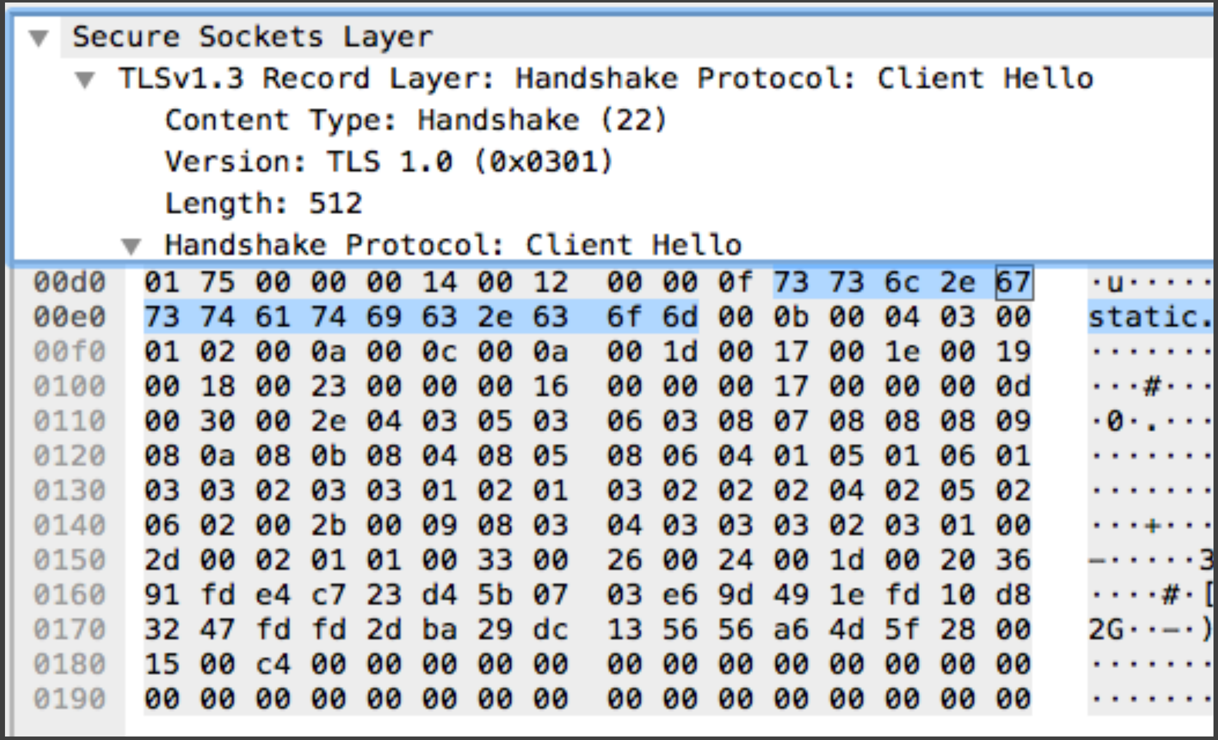
- Ports binned by major service while >1024 is reserved or dynamic

Reducing the unique IP feature space

- Even with reducing the amount of 3rd party advertiser traffic, the unique IP feature space was large
- Reduced by condensing traffic to the ICANN address space holder
- Used the MaxMind GeoLite2 database
- Expensive operation required multithreading and switch statements to reduce processing

Design Priorities

- Prioritized the ability to read headers in plaintext to use for machine learning features
- HTTP/2, TLS 1.3 needed further engineering
- Wanted more metadata from the TLS handshake including the `ClientHello` message



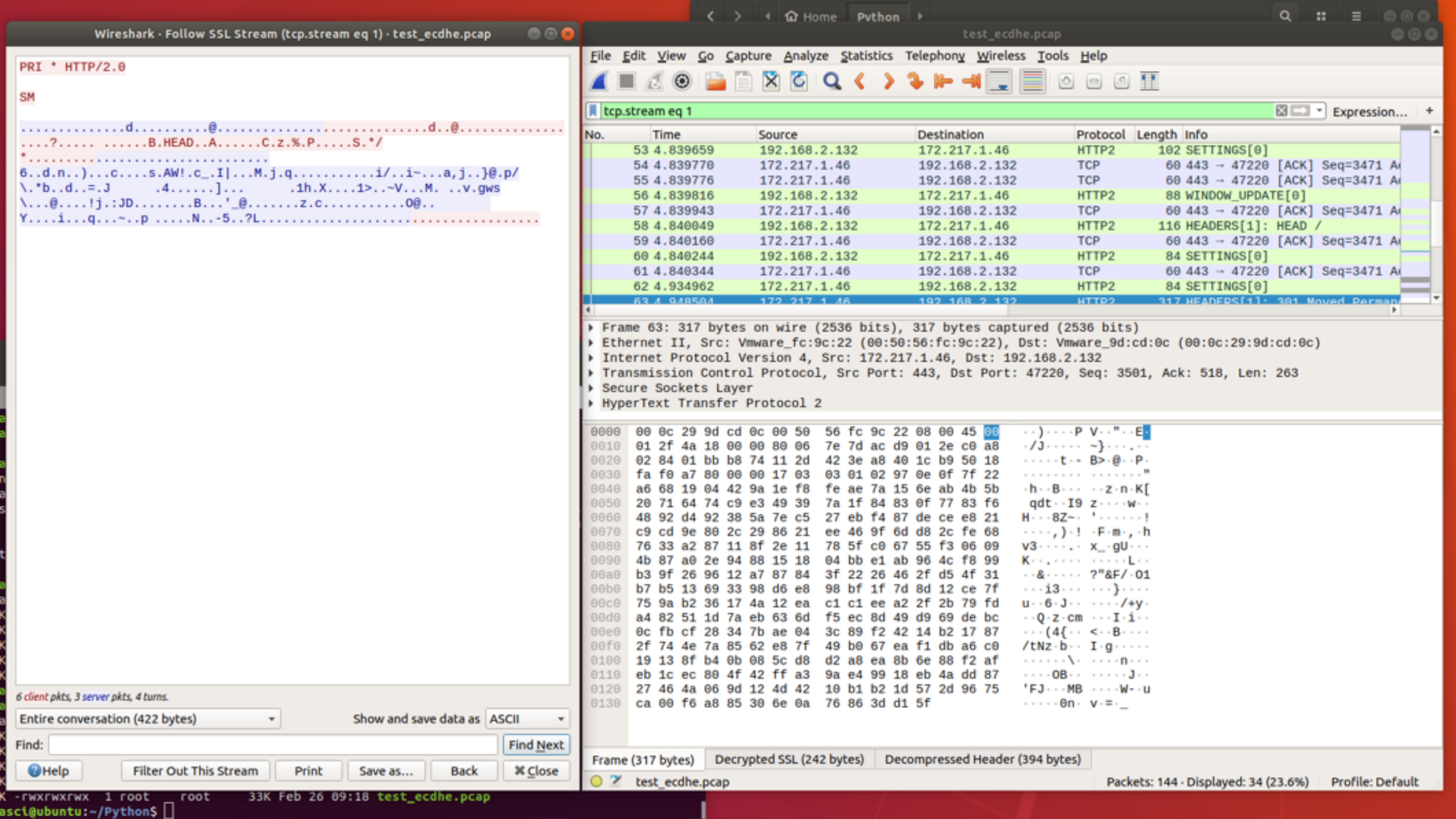
Challenges

- Based on the design priorities we came across a few challenges when it came to generating usable features

HTTP/2

- Originally, users browsed to a set of websites pulled from Alexa 100
  - But HTTP/2 was already enabled
  - HTTP/2 enhances the user experience by compressing the web traffic headers
- Removal of HTTP 2.0 headers was successful through cUrl

```
cUrl -I -tlsv1.2 -http1.1 https://www.google.com
```


Wireshark · Follow SSL Stream (tcp.stream eq 1) · test_ecdhe.pcap
PRI * HTTP/2.0
SM
6 client pkts, 3 server pkts, 4 turns.
Entire conversation (422 bytes)
Show and save data as ASCII
Find:
Find Next
Help
Filter Out This Stream
Print
Save as...
Back
Close
test_ecdhe.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 1
Expression...
No. Time Source Destination Protocol Length Info
53 4.839659 192.168.2.132 172.217.1.46 HTTP2 102 SETTINGS[0]
54 4.839770 172.217.1.46 192.168.2.132 TCP 60 443 → 47220 [ACK] Seq=3471
55 4.839776 172.217.1.46 192.168.2.132 TCP 60 443 → 47220 [ACK] Seq=3471
56 4.839816 192.168.2.132 172.217.1.46 HTTP2 88 WINDOW_UPDATE[0]
57 4.839943 172.217.1.46 192.168.2.132 TCP 60 443 → 47220 [ACK] Seq=3471
58 4.840049 192.168.2.132 172.217.1.46 HTTP2 116 HEADERS[1]: HEAD /
59 4.840160 172.217.1.46 192.168.2.132 TCP 60 443 → 47220 [ACK] Seq=3471
60 4.840244 192.168.2.132 172.217.1.46 HTTP2 84 SETTINGS[0]
61 4.840344 172.217.1.46 192.168.2.132 TCP 60 443 → 47220 [ACK] Seq=3471
62 4.934962 172.217.1.46 192.168.2.132 HTTP2 84 SETTINGS[0]
Frame 63: 317 bytes on wire (2536 bits), 317 bytes captured (2536 bits)
Ethernet II, Src: Vmware_fc:9c:22 (00:50:56:fc:9c:22), Dst: Vmware_9d:cd:0c (00:0c:29:9d:cd:0c)
Internet Protocol Version 4, Src: 172.217.1.46, Dst: 192.168.2.132
Transmission Control Protocol, Src Port: 443, Dst Port: 47220, Seq: 3501, Ack: 518, Len: 263
Secure Sockets Layer
HyperText Transfer Protocol 2
Frame (317 bytes)
Decrypted SSL (242 bytes)
Decompressed Header (394 bytes)
Packets: 144 · Displayed: 34 (23.6%)
Profile: Default
-rwxrwxrwx 1 root root 33K Feb 26 09:18 test_ecdhe.pcap
asci@ubuntu:~/Python$

TLS 1.3 vs. 1.2

- TLS 1.3 removed the cipher suites that plagued TLS 1.2 (like CBC)
- TLS 1.3 decryption requires ephemeral Diffie-Hellman keys that are established between the user’s endpoint and the webserver
- How can we provide more cleartext features for machine learning?

TLS 1.3 Downgrading

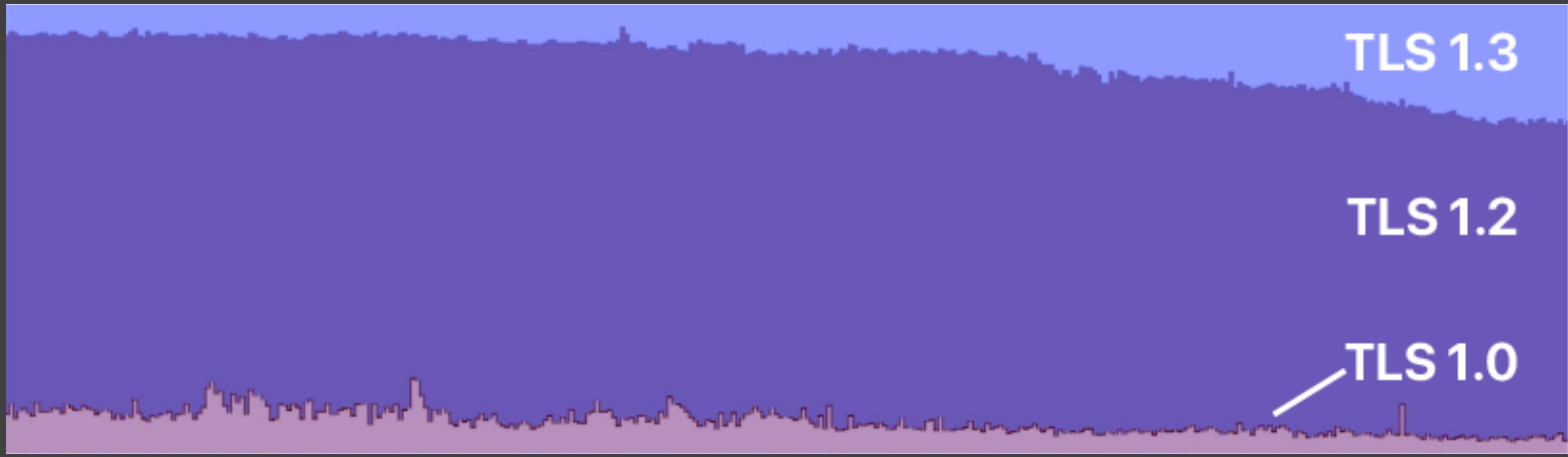
- Can’t we just use TLS 1.2 if we ask nicely?
- Asking many top Alexa websites to downgrade their cipher suites breaks the website
  - Would have had to cherry pick websites that have yet to upgrade to modern crypto suites
  - Some modern libraries only provide TLS 1.3[1] or frameworks support TLS 1.3 by default
    - And the share of TLS 1.3 is growing quicker than the adoption of 1.2

[1] https://github.com/facebookincubator/fizz)

TLS 1.3 Adoption

Browser	TLS 1.3 (%)
Chrome	30%
Firefox	27%
Safari	27%

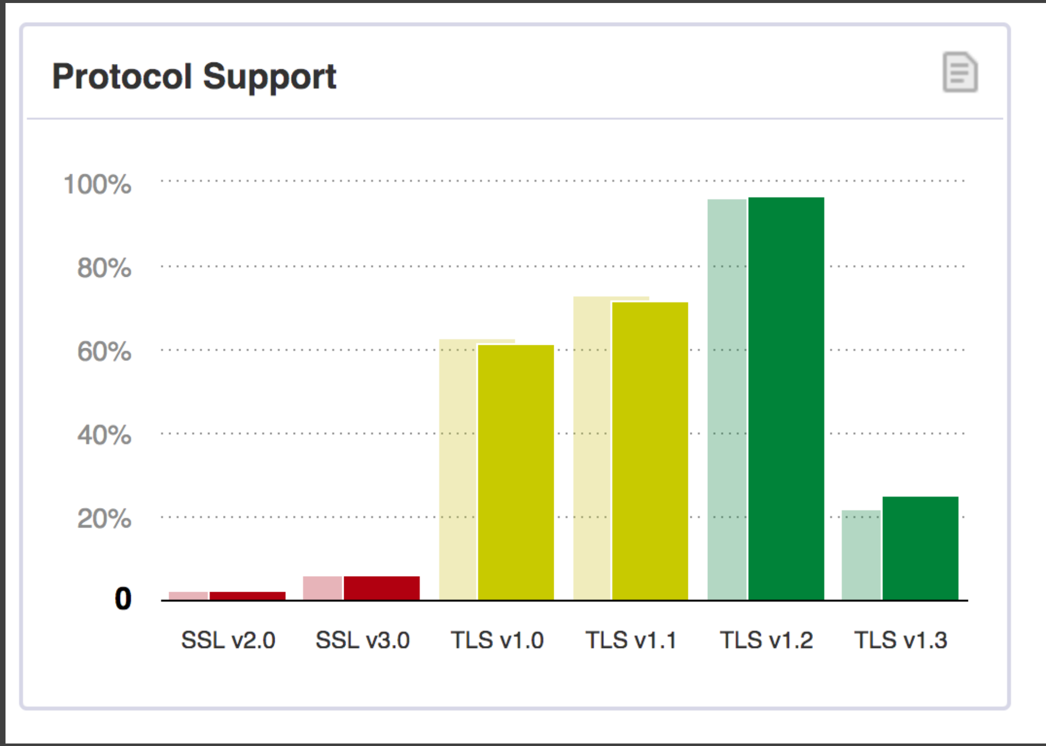
Percentage of TLS 1.3 connections amongst web browsers as of Aug 2019



Cloudflare TLS version trends from May 2018 to May 2019
https://ietf.org/blog/tls13-adoption/

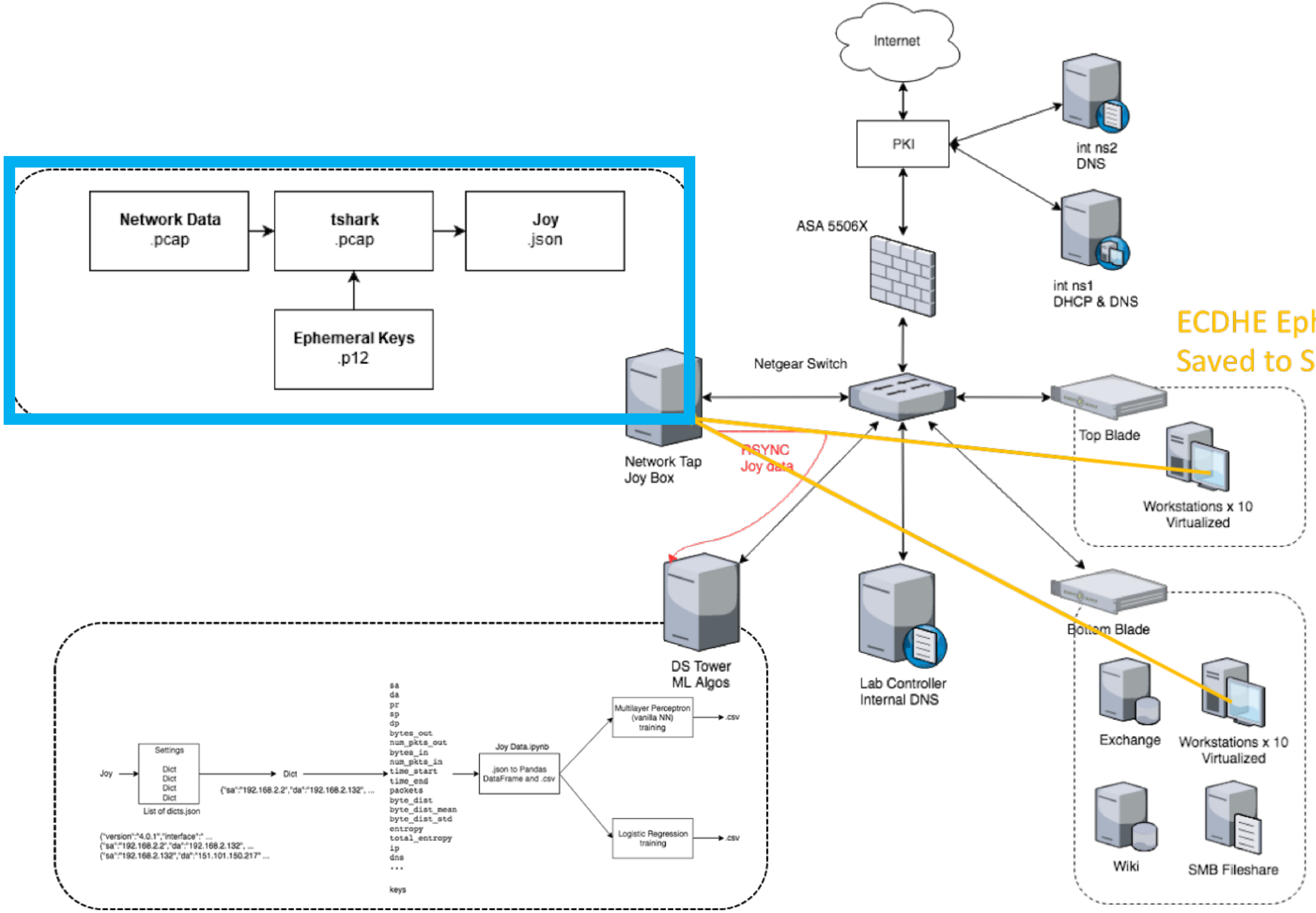
TLS 1.3 Adoption

- Qualys' SSL Pulse does a monthly scan across 150,000 SSL- and TLS-enabled websites and provides a dashboard of distributions and protocol support.
- TLS 1.2 still reigns supreme (for now)



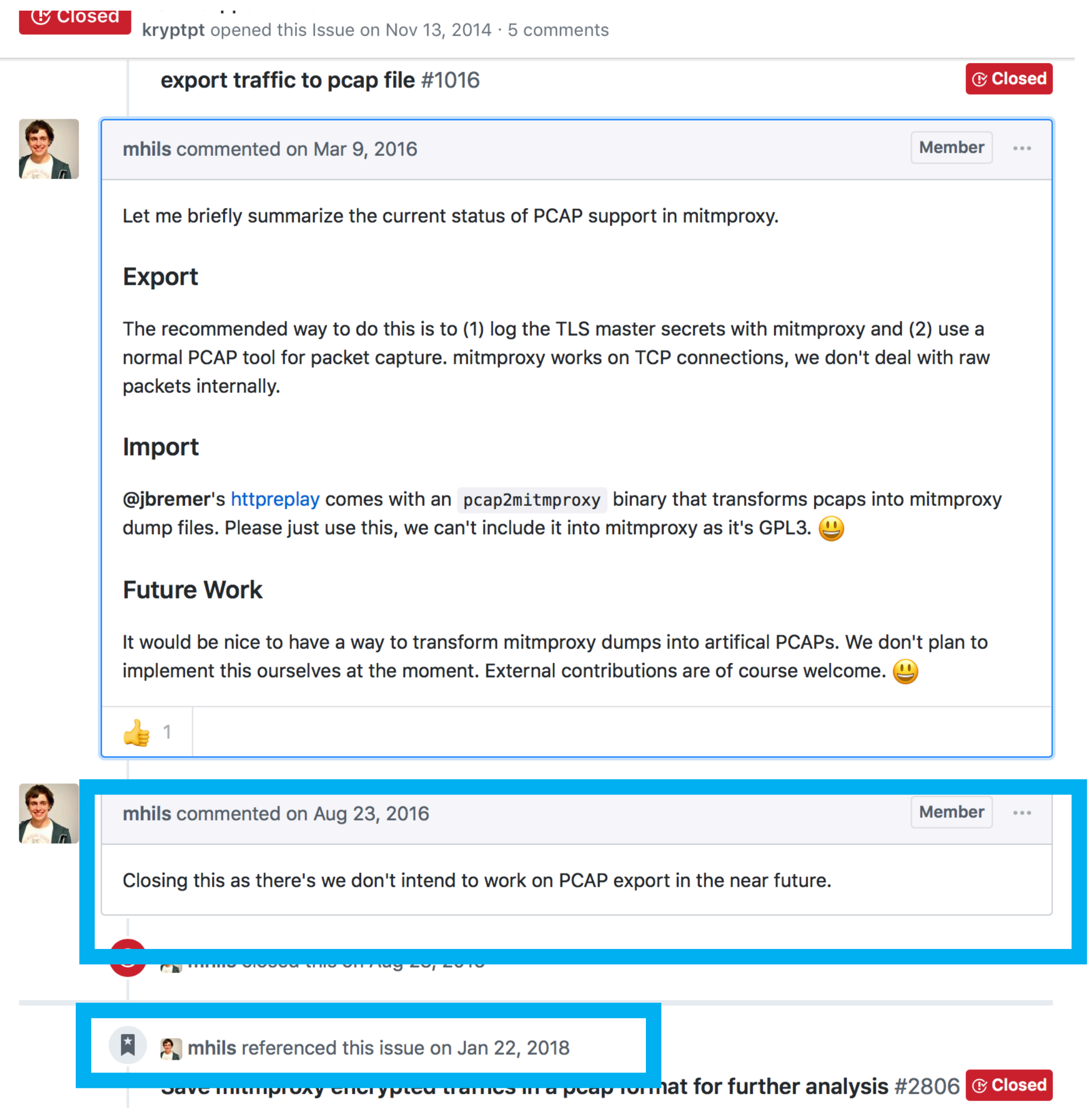
Qualys SSL Pulse
https://www.ssllabs.com/ssl-pulse/

ECDHE Ephemeral Keys Saved to SMB Share



TLS

- Can’t we just MiTM?
- We lost data visibility when using proxy servers (distinct IPs of user VMs) (MiTM proxy, Squid proxy) and neither of these proxies could save unencrypted traffic
    - https://github.com/mitmproxy/mitmproxy/issues/408#issuecomment-194415504
- Even with the keys, Wireshark could not save the decrypted pcaps *either*



For Future Reference

- Mitmpcap, a mitmproxy addon script, exports traffic to PCAP file, so you can view the decoded HTTPS or HTTP/2 traffic in other programs.
  - https://github.com/muzuiget/mitmpcap

TLS

- Asking nicely didn’t work
- MiTM didn’t work
- How do we can we provide those features?
  - Mirror websites in the cloud to control the cipher suite used
    - Con: Adds artificiality to data
    - Pro: Reduces the amount of third party advertiser traffic

- **Residual traffic was TLS 1.3 (OS updates)**
- **Majority of web traffic successfully uses TLS 1.2**

GeoIP Lookups

```
In [53]: OCSVM_model.fit_predict(X)
         OCSVM_model.score_samples(X) #Accuracy Rating

Out[53]: array([123.16765345, 114.61292808, 112.58887769,
                206.31248729, 198.57973916])

In [54]: #Y.mean() # null error rate

In [55]: # 1 - 0.001 = 99.99
         #coeff_df = DataFrame(list(zip(X.columns,np.tran

In [56]: # Split data
         #X_train,X_test,Y_train,Y_test = train_test_spli

In [57]: #log_model2 = LogisticRegression(solver='sag',ma
         #log_model2.fit(X_train,Y_train) # Fit new model

In [58]: #class_predict = log_model2.predict(X_test) # Ru

In [59]: #print(metrics.accuracy_score(Y_test,class_predi

In [60]: print("--- %s seconds ---" % (time.time() - star

         --- 1767.6108849048615 seconds ---
```

```
In [53]: OCSVM_model.fit_predict(X)
         OCSVM_model.score_samples(X) #Accuracy Rating

        7.65590373, 7.65948639, 3.28499574, 7.65545312, 1.99934744,
        7.65574556, 7.68814426, 7.65564252, 7.65594118, 7.70586689,
        7.56241694, 7.6667588 , 5.07693838, 7.65625653, 6.20145773,
        6.29527442, 8.44828022, 4.69793124, 8.63261821, 7.22238043,
        7.66436065, 7.65594264, 7.66013036, 7.65547255, 7.65547001,
        4.67214869, 8.4406612 , 7.64906761, 7.65590361, 2.05470579,
        7.67466059, 7.65606004, 7.65638857, 7.65605016, 5.69651846,
        7.65604011, 7.65602986, 1.99934744, 7.65638857, 7.65543142,
        7.65596433, 4.48024331, 7.68243619, 7.65600881, 7.65594118,
        7.65594118, 7.65623123, 7.65594118, 7.65623123, 7.65594118,
        7.65594118, 7.65594118, 7.655998  , 5.0461702 , 6.60880411,
        6.29527442, 7.67681701, 7.65598699, 7.65597578, 4.95800579,
        5.82514926, 8.0046963 , 6.82367588, 7.65638857, 7.67498135,
        7.65596705, 7.65595277, 7.65638857, 7.66428588, 7.67939458,
        8.42838423, 7.65594096, 8.57816055, 8.48246519, 8.23473639,
        7.41342345, 7.78831953, 8.39948691, 7.65592894, 6.32081566,
        7.82591689, 7.82652261, 7.82718125, 6.0362733 , 7.82718125,
        7.82591689, 7.82652261, 7.23798668, 7.82718125, 7.82652261,
        7.6592379 , 7.82591689, 7.82718125, 7.82652261, 7.03324173,
        7.82718125, 6.0362733 , 7.65893237, 7.65893237, 7.65893237,

In [54]: #Y.mean() # null error rate

In [55]: # 1 - 0.001 = 99.99
         #coeff_df = DataFrame(list(zip(X.columns,np.transpose(log_model.coef_))))

In [56]: # Split data
         #X_train,X_test,Y_train,Y_test = train_test_split(X,Y)

In [57]: #log_model2 = LogisticRegression(solver='sag',max_iter=1000)
         #log_model2.fit(X_train,Y_train) # Fit new model with training data

In [58]: #class_predict = log_model2.predict(X_test) # Run a prediction with X_test dataset

In [59]: #print(metrics.accuracy_score(Y_test,class_predict)) # Compare Y_test to prediction

In [60]: print("--- %s seconds ---" % (time.time() - start_time))

         --- 2.421576738357544 seconds ---
```

Hosting

- Although PCAPs are better than Netflows, PCAPs are much larger
- **Difficult to find a place to host this size of a dataset (60gb) as a research set unless you're paying for it**

Resources

- If you decide to provide a dataset to the community or need data to provide research to the community, these resources may help

Other cool datasets

- Malware
  - https://github.com/endgameinc/ember
  - http://amd.arguslab.org/
- Canadian Institute for Cybersecurity
  - Android Malware
  - DDoS
  - CICIDS
  - Botnet
  - https://www.unb.ca/cic/datasets/index.html

Possible Hosts

- Impact Cybertrust
  - https://www.impactcybertrust.org
- SNAP Large Network Dataset Collection
  - snap.stanford.edu/about.html
- networkrepository.com
  - Largest network repository across 30 domains (bioinformatics, etc)
- AWS Dataset Program
  - https://aws.amazon.com/opendata/public-datasets/
- AWS Glacier
  - ~ $210.6 for 10 years (60gb)
  - https://docs.aws.amazon.com/amazonglacier/latest/dev/uploading-archive-mpu.html

Why didn't you just use...

- The Wall of Sheep dataset
- The DEF CON dataset
- The NCCDC dataset (or any of the regional sets)

- None of these datasets are labelled!

Future Work

- Study machine learning algorithms trained using this data
  - Determine how to make them more robust against adversarial examples
- Operational Technology (OT)-specific protocols with serial and serial-over-Ethernet traffic
  - Or a hybrid IT-OT network
- Capture the Flag competitions could be used to gather more participation
  - Would need a separate virtual environment for each participant

Other Major References

- Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP ’10, pages 305–316, Washington, DC, USA, 2010. IEEE Computer Society.
- Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial machine learning. *In Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec ’11, pages 43–58, New York, NY, USA, 2011. ACM.
- 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*, pages 35–46. ACM, 2016.
- Blake Anderson and David McGrew. Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity*. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’17, pages 1723–1732, New York, NY, USA, 2017. ACM.
- Tomás Pevný, Martin Komon, and Martin Rehaky. Attacking the ids learning processes. pages 8687–8691, 10 2013.

Thank you!

- Questions?