

Learning to Listen: Machine Learning for Adaptive Wireless Adversary Detection

About Me

- @JohnDunlap2
- Security Researcher
- Reverse Engineer
- Avid collector of bad software
- Work for GDSSecurity doing code review / RE / Research

I swear this will not be a buzzword laden talk

BINGO!		
Deep Learning	Inference	Tensorflow
Training	FREE	Keras
AI Revolution	Cyber	APT

A Wild Gabe Appears



Gabe is a wireless Ninja.

```
Applications ▾ Places ▾ Terminal ▾ Thu 04:05 en1 [1] [ ] [x]
```

```
root@kali: ~/eaphammer
```

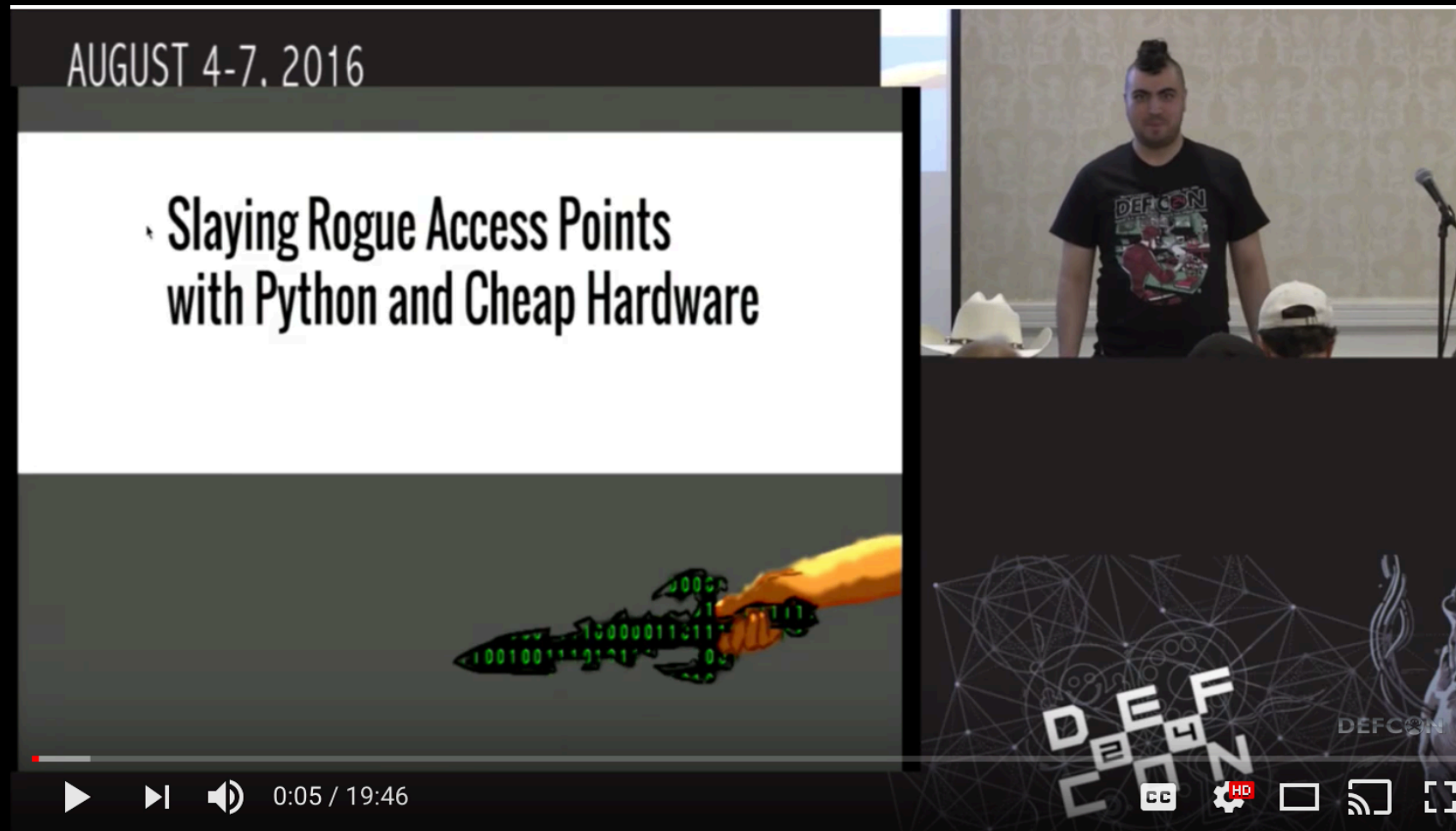
```
File Edit View Search Terminal Help
```

```
openssl pkcs12 -in server.p12 -out server.pem -passin pass:`grep output_password server.cnf | sed 's/.*=//;s/^ *'`  
-passout pass:`grep output_password server.cnf | sed 's/.*=//;s/^ *'`  
openssl verify -CAfile ca.pem server.pem  
server.pem: OK  
openssl x509 -inform PEM -outform DER -in ca.pem -out ca.der  
root@kali:~/eaphammer# ./eaphammer -i wlan1 --channel 4 --auth ttls --wpa 2 --essid AndroidCorp --creds
```

v0.0.7

```
[*] stopping network-manager service.  
100% | 4/4 [00:04<00:00, 1.00s/it]  
[*] stopping wpa_supplicant service.  
100% | 4/4 [00:04<00:00, 1.00s/it]  
Error: NetworkManager is not running.  
[*] Reticulating radio frequency splines...  
100% | 1/1 [00:01<00:00, 1.00s/it]  
Configuration file: /root/eaphammer/conf/hostapd-wpe.conf  
Using interface wlan1 with hwaddr 08:11:22:33:44:00 and ssid "AndroidCorp"  
wlan1: interface state UNINITIALIZED->ENABLED  
wlan1: AP-ENABLED  
press enter to quit...[]
```

Two years ago he gave this talk



AUGUST 4-7, 2016

Slaying Rogue Access Points with Python and Cheap Hardware

The video player shows a presentation slide with a title and a date. The slide features a graphic of a hand holding a glowing green object with binary code. The video player interface includes a progress bar, play/pause button, volume icon, and a timestamp of 0:05 / 19:46. The video content shows a man in a DEFCON t-shirt standing at a podium with a microphone, with a cowboy hat and a white cap on the podium. The background of the video shows a DEFCON logo and a network diagram.

0:05 / 19:46

Gabe invented a way to take out rogue Aps.
That's Metal!



Sentry Gun

s0lst1c3 / sentrygun

Watch 10Star 49Fork 27

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Rogue AP killer

36 commits

4 branches

0 releases

2 contributors

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

s0lst1c3 Merge pull request #2 from Abdulwahaab710/patch-2

Latest commit 8b03e96 on Nov 30, 2017

docs/img	addings docs	2 years ago
.gitignore	Initial commit	3 years ago
README.md	Fixing markdown hearders	8 months ago
configs.py	evil twin detection using tx varation works reliably	2 years ago
network_tools.py	splitting client and server into two packages	2 years ago
pip.req	splitting client and server into two packages	2 years ago
sentrygun.py	Update sentrygun.py	2 years ago
sg-calibrator.py	evil twin detection using tx varation works reliably	2 years ago
sniffer.py	cleaned up output	2 years ago
whitelist.txt	splitting client and server into two packages	2 years ago

What does it do?

- Detects Rogue Aps
- Tries to **locate** them
- Tries to DOS them

How is it Tracking Rogue A.Ps?

- Spatially distributed sensor array
- Do benchmarking on the network in a trusted environment to obtain normal signal strength levels
- Flag abnormal behavior
- Location is inferred from signal strength relative to a known sensor position

Basically arithmetic mean

```
ssids = calibration_table['ssids']
for s in ssids:

    for bssid in ssids[s]['bssids']:

        tx_list = ssids[s]['bssids'][bssid]['packets']
        tx_list_len = ssids[s]['bssids'][bssid]['len']

        mean = sum(tx_list) / tx_list_len
        max_dev = max(abs(el - mean) for el in tx_list)

        upper_bound = mean + (max_dev * N_TIMES_MAX_DEV)
        lower_bound = mean - (max_dev * N_TIMES_MAX_DEV)

        ssids[s]['bssids'][bssid]['mean'] = mean
        ssids[s]['bssids'][bssid]['max_dev'] = max_dev
        ssids[s]['bssids'][bssid]['upper_bound'] = upper_bound
        ssids[s]['bssids'][bssid]['lower_bound'] = lower_bound
```

This actually worked quite well in practice

- Gabe was able to find and eliminate rogue AP's in his tests
- Because of the multi-sensor setup he was able to locate attackers

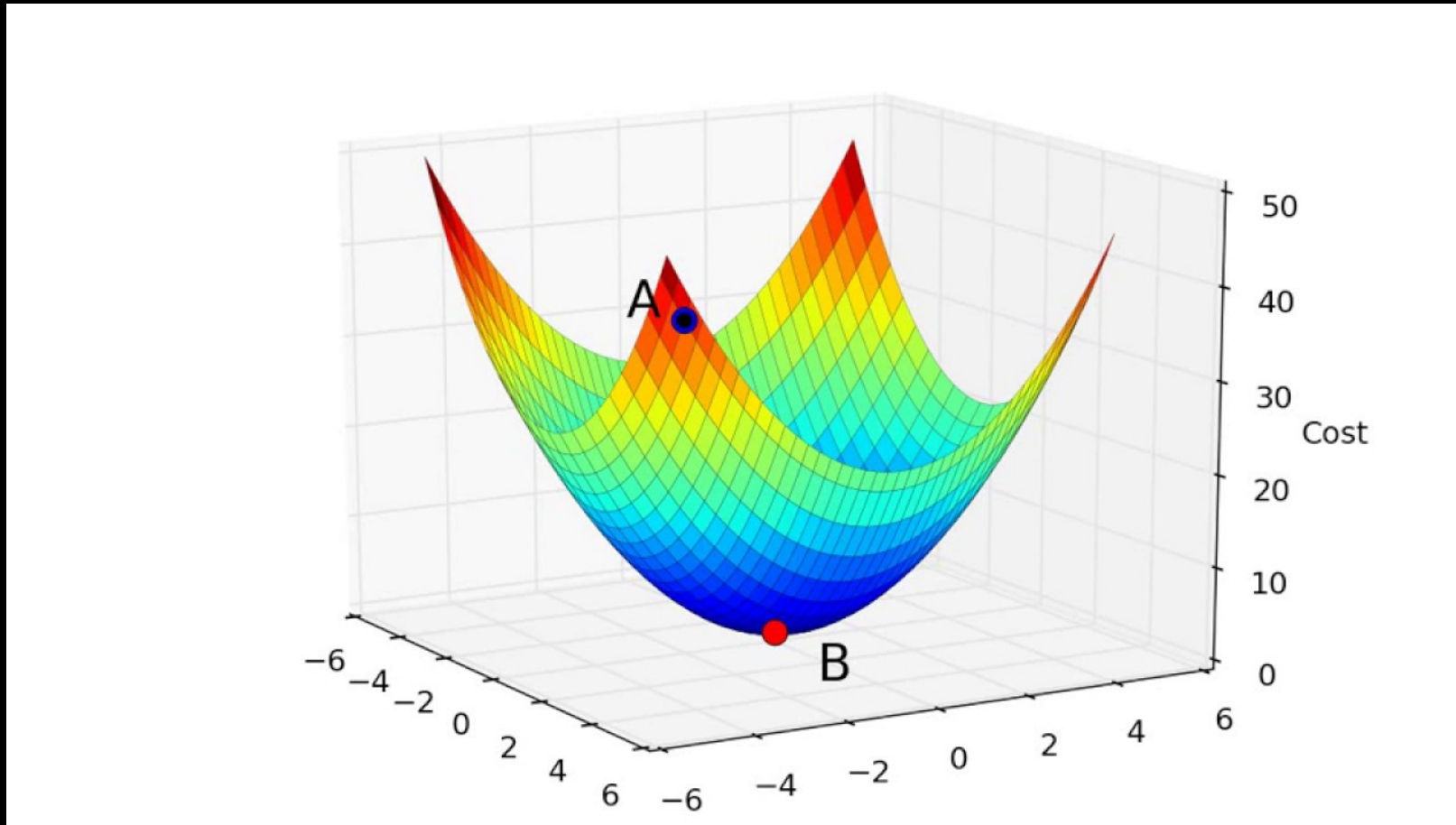
But I'm one of *those* engineers



So my mind demanded more

- More math
- More features
- Deeper characterization of attackers

How about machine learning



The idea:

- Add machine learning algorithms to Gabe's tool to train a model to better identify and classify attackers
- Try to identify signals that don't fit well within Gabe's algorithm.
- Predict recurring attack types and locations

The Prior Work

- Surely someone has tried something like this in the literature
- Turns out there are a lot of close calls but no exact matches

Why is ML Good for this?

- Classification
- Over time

YANG, SONG AND GU: ACTIVE USER-SIDE EVIL TWIN ACCESS POINT DETECTION USING STATISTICAL TECHNIQUES 2012

- “We propose to exploit the intrinsic communication structure and property of evil twin attacks. Furthermore, we propose two statistical anomaly detection algorithms for evil twin detection, TMM and HDT. In particular, our HDT improves TMM by removing the training requirement. HDT is resistant to the environment change such as network saturation and RSSI fluctuation.”
- HDT – Hop Differentiation Technique
- TMT - Trained Mean Matching
- Uses SPRT - Sequential Probability Ratio Test
- Learning and non-learning variants
- Able to measure difference between rogue and non-rogue without a whitelist by measuring statistical properties of the wifi radio signal itself (e.g s/n ratio)
- Client side approach - each workstation runs this
- Seeks to distinguish one hop versus multi-hop communications.

Kim,Seo,Shon,Moon: A novel approach to detection of mobile rogue access points (2013)

- Round trip time analysis
- Specifically detects differences between mobile and wired networks by analyzing RTT
- Assuming the attacker is exfiltrating via 3G, the difference in latency should be a dead giveaway
- Uses ICMP packets for the timing analysis

Jana, Kaserer: On Fast and Accurate Detection of Unauthorized Wireless Access Points Using Clock Skews

- Detects endpoints via differences in clock skew
- Classifier is trained on whitelisted network and compares expected results to the live network
- Gets clock skew from Time Synchronization Function (TSF) timestamps in the IEEE 802.11 beacon/probe response messages sent by the AP
- Server based. Requires a whitelist and training.

Learning

The side channels

- Clock Skew
- Signal strength
- Timestamp

Picking a good network

- This is time series data
- Data has both long time scale and short time scale attributes
- Our adversary may be evolving to counteract the model
- Adversaries may be extremely heterogeneous

The Network

- LSTM – Long short term memory
- Adapted from LSTMs used to in medical diagnostics time series data
- LSTMs are good because they can capture long term and short term time series data

LSTM

Long short-term memory (LSTM) units are units of a **recurrent neural network (RNN)**. An RNN composed of LSTM units is often called an *LSTM network*. A common LSTM unit is composed of a **cell**, an **input gate**, an **output gate** and a **forget gate**. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.

LSTM networks are well-suited to **classifying**, **processing** and **making predictions** based on **time series** data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the exploding and **vanishing** gradient problems that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, **hidden Markov models** and other sequence learning methods in numerous applications^[citation needed].

The Features

- Training label
- Signal strength
- Timestamp
- Time packet received

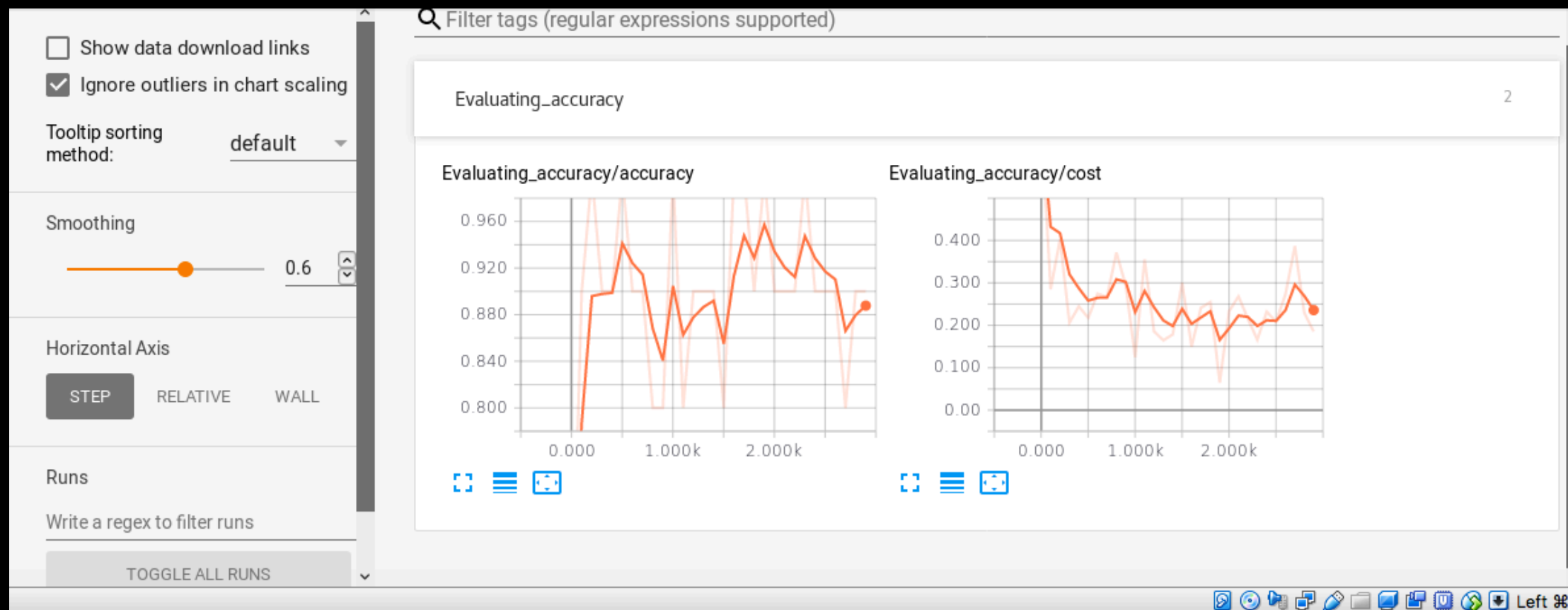
The Training

- 2k rounds of training.

Avoiding Overtraining

- Used a variety of rogue AP's
- Largest data set possible

The fancy graph



The Results

- Up to 90% accuracy!
- LSTM detects recurring patterns in the data
- No signs of overtraining

Did we do better than before?

- Gabe's algorithm has a FP rate of up to 50%
- Single packets may either live inside the mean or legitimate packets may live outside of it.
- With better training ML may be able to do in depth fingerprinting with extremely low FP rates.