Using Machines to Exploit Machines

Harnessing AI to Accelerate Exploitation

Guy Barnhart-Magen
Ezra Caltum

@barnhartguy @aCaltum

Legal Notice and Disclaimers

This presentation contains the general insights and opinions of its authors, ~~Guy Barnhart-Magen~~ and Ezra Caltum. We are speaking on behalf of ourselves only, and the views and opinions contained in this presentation should not be attributed to our employer.

The information in this presentation is provided for informational and educational purposes only and is not to be relied upon for any other purpose. Use at your own risk! We makes no representations or warranties regarding the accuracy or completeness of the information in this presentation. We accept no duty to update this presentation based on more current information. We disclaim all liability for any damages, direct or indirect, consequential or otherwise, that may arise, directly or indirectly, from the use or misuse of or reliance on the content of this presentation.

No computer system can be absolutely secure.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

*Other names and brands may be claimed as the property of others.

~~@barnhartguy~~ @aCaltum

$ ID

Guy Barnhart-Magen

@barnhartguy

BSidesTLV Chairman and CTF Lead

Ezra Caltum

@acaltum

BSidesTLV Co-Founder

DC9723 Lead

LAS VEGAS
BSIDES

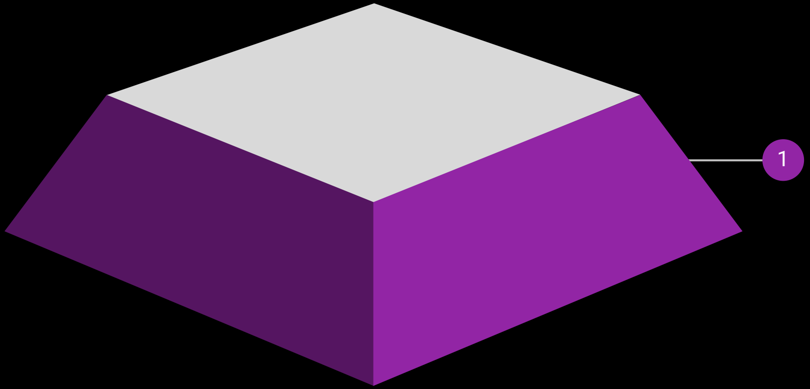
CRYPTO + PRIVACY
VILLAGE

44CON

t2.fi
INFOSEC

LKSAT
SKYTA
TALKS
ATSKY

OUR PROBLEM

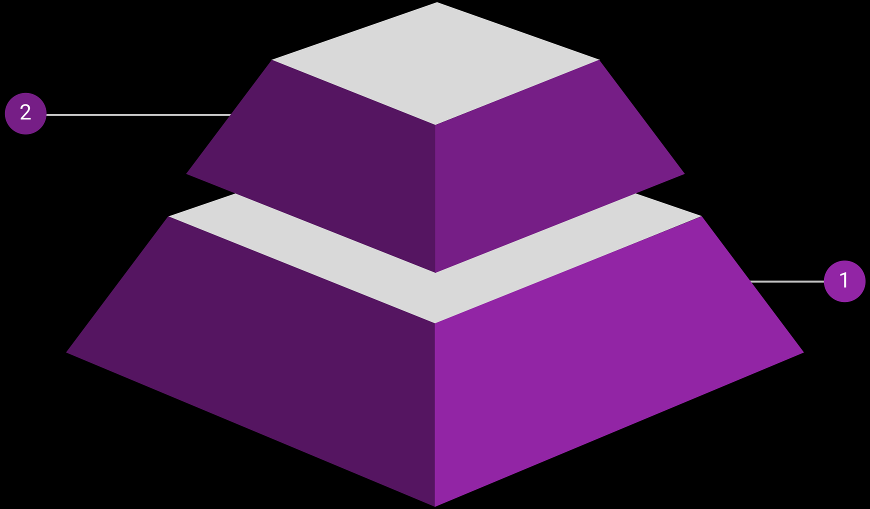


Fuzz Testing

Literally thousands of crashes to analyze

(good problem to have?)

OUR PROBLEM



Automation

Might miss something important, but helps reduce from thousands to hundreds of results

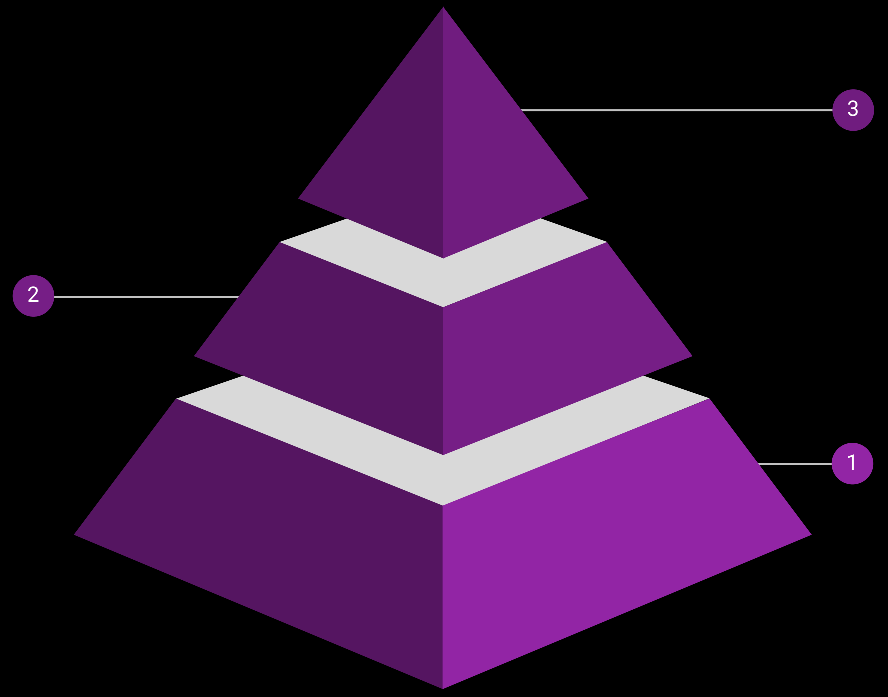
Fuzz Testing

Literally thousands of crashes to analyze

(good problem to have?)

@barnhartguy @aCaltum

OUR PROBLEM



Manual Analysis
Can only do a limited amount with limited researchers time

Automation

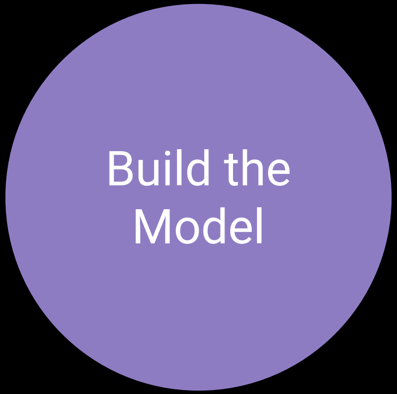
Might miss something important, but helps reduce from thousands to hundreds of results

Fuzz Testing
Literally thousands of crashes to analyze

(good problem to have?)

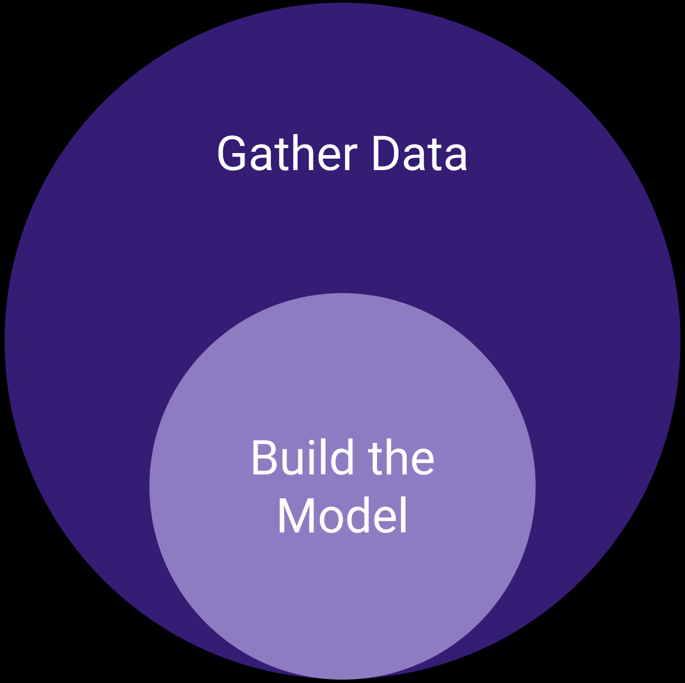
@barnhartguy @aCaltum

EFFORT BALANCE

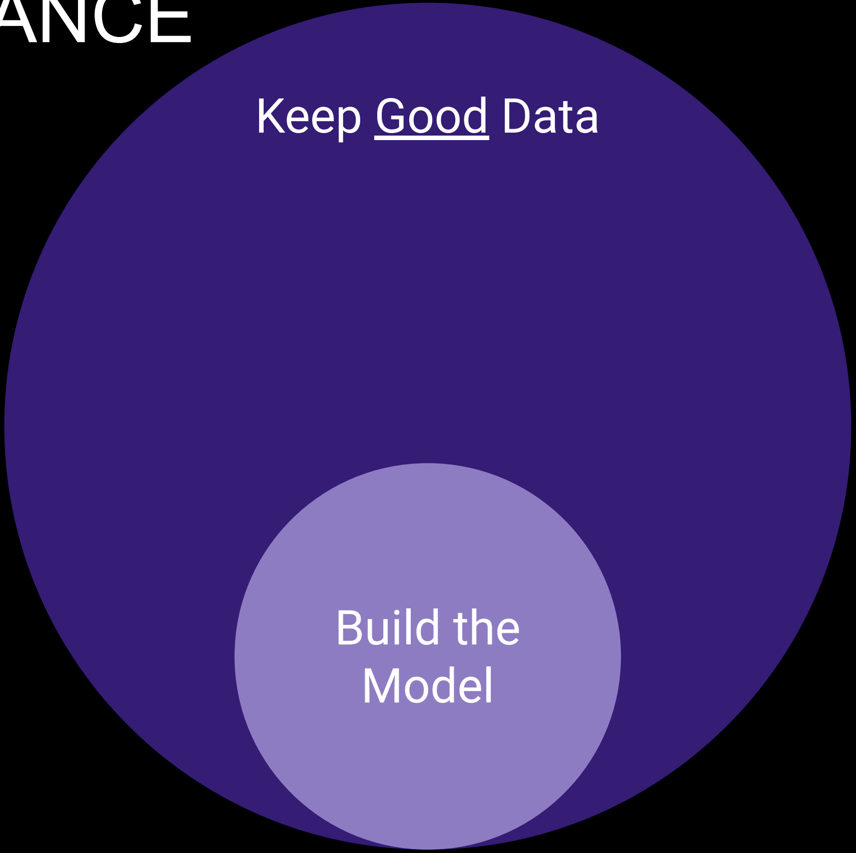


Build the Model

EFFORT BALANCE



EFFORT BALANCE



@barnhartguy @aCaltum

PROBLEM STATEMENT

PROBLEM STATEMENT

What is Australia?

PROBLEM STATEMENT

Can we create an ML model that can triage crashes and help us focus on the exploitable ones?

(we got a lot of crashes from AFL)

REVISED PROBLEM STATEMENT

Can we create an ML model that can outperform exploitable, based on the same data?

it should perform at least as well as exploitable

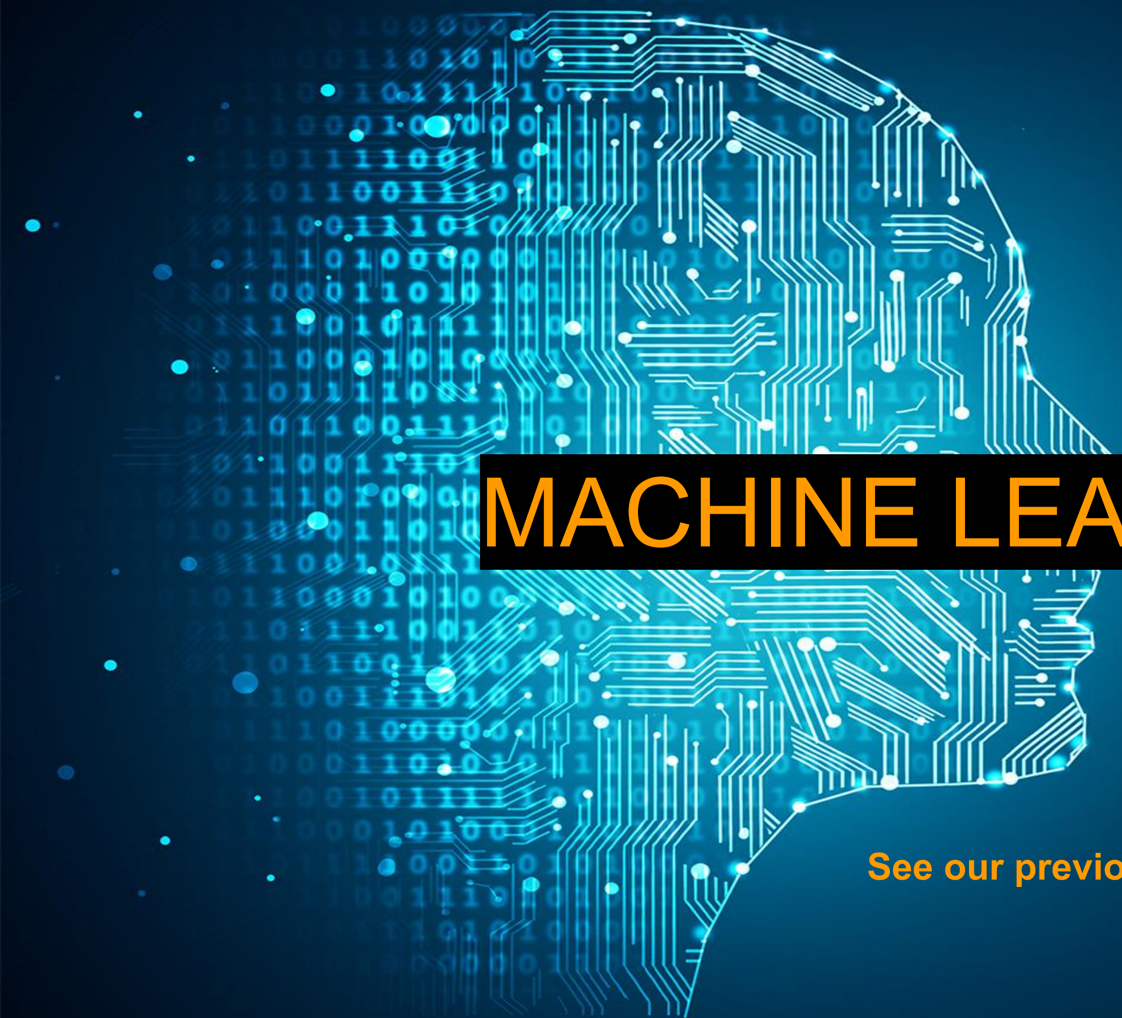
FULL DISCLOSURE

Limited dataset - but we tried anyway (no DL today)

We want to focus on the methodology

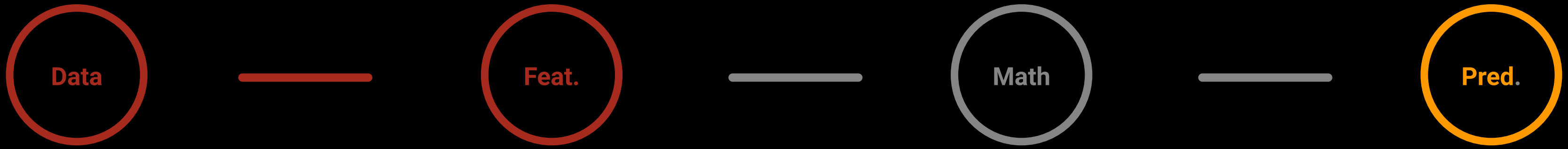
We can’t trust this results, but they are worth sharing

MACHINE LEARNING



See our previous talks on hacking machine learning systems
:-)

WHAT IS MACHINE LEARNING?



Data Ingestion

Normalizing and converting data to a canonical way for feature extraction

Feature Extraction

Analyzing the data and extracting the interesting features from it

Model Fitting

Repeatedly trying to improve model fit to the data observed

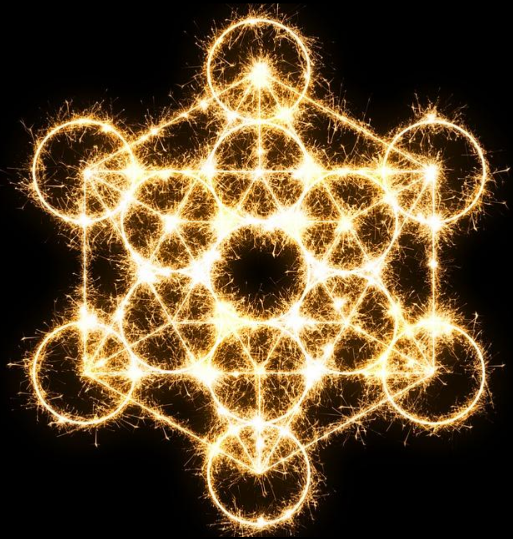
Predictions

Given a never seen before datum, what does the model predict it to be

MACHINE LEARNING

What it isn't:

- Magic
- A solution to every problem
- Difficult or Complex
- One of the holy VC buzzwords:
  - Blockchain
  - Cyber
  - Zero Trust



THE DIFFERENCE BETWEEN ML AND AI

If it is written in Python, it’s probably Machine Learning

If it is written in PowerPoint, it’s probably AI

EXAMPLE

EXAMPLE

Using Machines to Exploit Machines

Harnessing **AI** to Accelerate Exploitation

Everyone Confuses “AI” with “ML”

So do We
Sorry

WHAT IS IT GOOD FOR?

Finding patterns in a lot of data, patterns you did not expect (counter intuitive)

WHAT IS IT GOOD FOR?

Finding patterns in a lot of data, patterns you did not expect (counter intuitive)

Correlating different inputs you suspect are related somehow

WHAT IS IT GOOD FOR?

Finding patterns in a lot of data, patterns you did not expect (counter intuitive)

Correlating different inputs you suspect are related somehow

Abstracting a problem and throwing it at an algorithm, hoping for the best (e.g. being lazy)

PREDICTIONS

ML makes predictions based on previously seen data

Your data quality is important! (data is not information)

WHAT DO YOU GET?

How is this new sample I am testing now similar to all the other samples I’ve seen in the past?

Testing - extracting and then comparing features against your model

Crash Triage

Windows

An error has occured.We dont even know what is it.So cant fix it ,
and you have restart your computer.By the way if you restart your ,
computer you will in all open applications.
In the other hand on :)

Press Enter to return to Windows(It wont work), or

Press CTRL+ALT+DEL to restart your computer.

Error : 0E : 016F : BFF9B3D4

Uni.Q Design

A COMMON MORNING IN MY LIFE

- I start a fuzzing process overnight and go home

A COMMON MORNING IN MY LIFE

- I start a fuzzing process overnight and go home

- At first light in the morning (11:00) I drink a cup of coffee

A COMMON MORNING IN MY LIFE

- I start a fuzzing process overnight and go home
- At first light in the morning (11:00) I drink a cup of coffee
- I analyze the data from the crash dump with the help of a debugger

A COMMON MORNING IN MY LIFE

- I start a fuzzing process overnight and go home
- At first light in the morning (11:00) I drink a cup of coffee
- I analyze the data from the crash dump with the help of a debugger
- Based on my experience, and the output of some plugins, I classify the crashes as either exploitable or not

A COMMON MORNING IN MY LIFE

- I start a fuzzing process overnight and go home
- At first light in the morning (11:00) I drink a cup of coffee
- I analyze the data from the crash dump with the help of a debugger
- Based on my experience, and the output of some plugins, I classify the crashes as either exploitable or not
- I start developing a POC for the exploitable crashes.

A COMMON MORNING IN MY LIFE

- I start a fuzzing process overnight and go home ➔ No need for sleep for our AI overlords
- At first light in the morning (11:00) I drink a cup of coffee ➔ No need for coffee for our AI overlords
- I analyze the data from the crash dump with the help of a debugger ➔ Preprocessing phase prepares the data for the ML analysis
- Based on my experience, and the output of some plugins, I classify the crashes as either exploitable or not ➔ ML analyzes the data, based on its experience (training data), emits predictions (human intuition or heuristics)
- I start developing a POC for the exploitable crashes. ➔ Human minions will develop a PoC for the overlords

Our Data Set

CGC
CYBER
GRAND_CHALLENGE
MECHANICAL PHISH
SHELLPHISH
AUGUST 2016 | LAS VEGAS, NV
DARPA

DARPA CYBER GRAND CHALLENGE

We have 632 test cases that we know are exploitable

We ran `exploitable` against them and got:

- 607 were definitely exploitable
- 12 were probably exploitable
- 13 were unknown - the tool couldn’t reach a decision

SO, WHAT DOES A CRASH GIVE US?

EAX, EBX, ECX, EDX - general purpose (values, addresses)

ESP, EBP - Stack pointers

ESI, EDI - Source and Destination Index (for string operations)

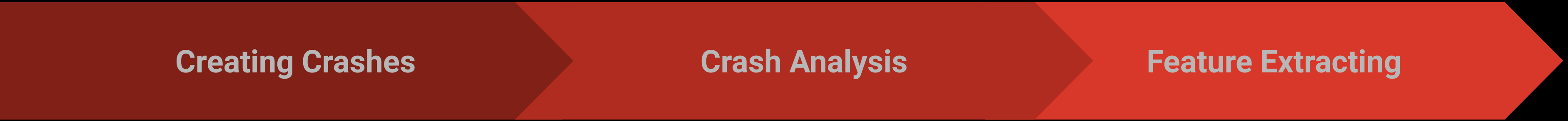
EIP - Instruction pointer

eflags - metadata (wasn’t actually useful at all, empty values)

CS, SS, DS, ES, FS, GS - Segment registers

Also a whole lot of other things which we didn't look at

OUR PROCESS



Creating Crashes

Crash Analysis

Feature Extracting

Running tests against a ~600 programs with known crashes, collecting the crash dumps

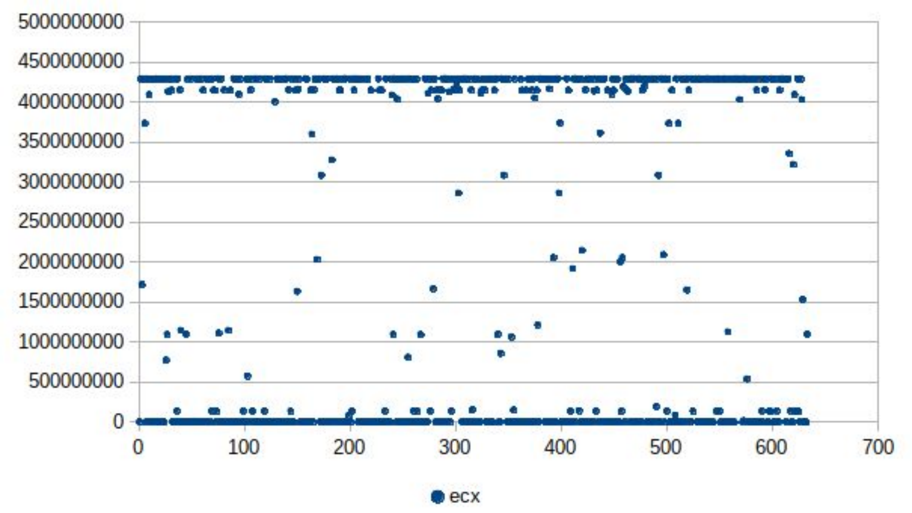
Analyzing the crash dumps using `exploitable`, collecting the stack and register values

Converting the data collected from the `exploitable` output to a canonical representation, extracting the features we cared about

PROBLEM

Register values are discrete and unrelated to each other

What can we learn from specific register values?



@barnhartguy @aCaltum

CLASSIFYING DATA

We tried breaking the values of the registers into three groups:

- High address range (kernel)
- Low address range (userland)
- Values

Bad results - data distribution not uniform :-(

BINNING

Dividing the values to evenly spaced bins

10 bins total, evenly distributed between [min_val, max_val]

This helps the model ignore specific values, and look at them as ranges

Good results :-)

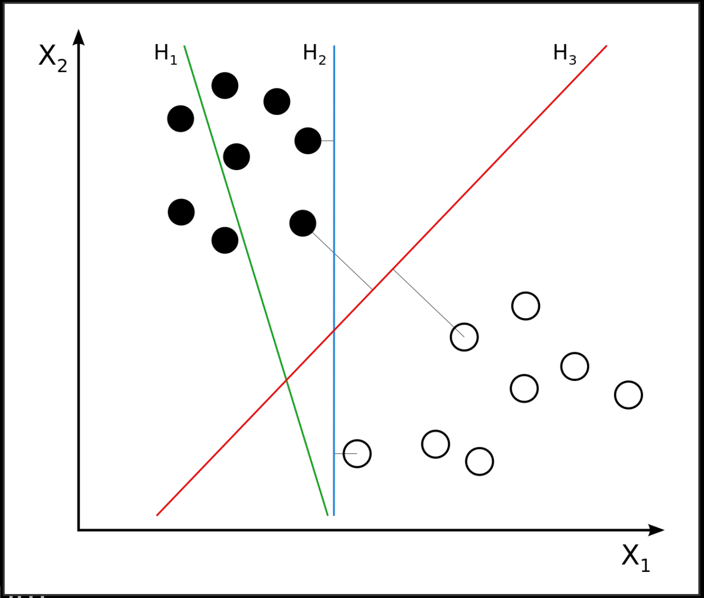
OneClassSVM

Train your major class (609 records, EXPLOITABLE)

Test your data against similarity to the model {-1,1}

+1 = very similar to the model

-1 = very not similar to the model



RESULTS - OneClassSVM

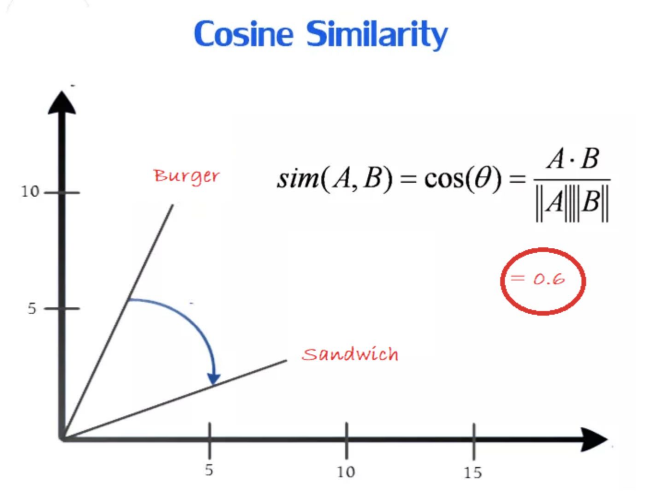
Anomaly detection using OneClassSVM: 23 records (from 25) are successfully recognized as belong to “exploit” class

- 23 records recognized as a major class:
    - 13 records previously labeled as “unknown”
    - 10 records previously labeled as “probably exploitable”
- 2 “probably exploitable” records identified as outliers

Class	1ClsSVM
Exploitable	+23
Probably Exploitable	2
Unknown	

COSINE SIMILARITY

- Cluster our data (609 records, EXPLOITABLE)
- Measure similarity between each data point (24 records) to the cluster
- We also used binning and not the actual register values



RESULTS - Cosine Similarity

We tried comparing using linear or centroid methods

Started with 9 register values, then adding the rest (15 register values, using binning)

~65% using values of 9 registers

~87% using values of 15 discretized registers

Class	CosSim Linear	CosSim Centroid
Exploitable	+16	+22
Probably Exploitable		
Unknown		

XGBoost

“Tree” that is built using the most contributing features

Very easy to explain how decisions are made, good for insights

Select 80% of the data (evenly sample from each group) for training, 20% for testing

RESULTS - XGBoost

95-99% accuracy

RESULTS - XGBoost

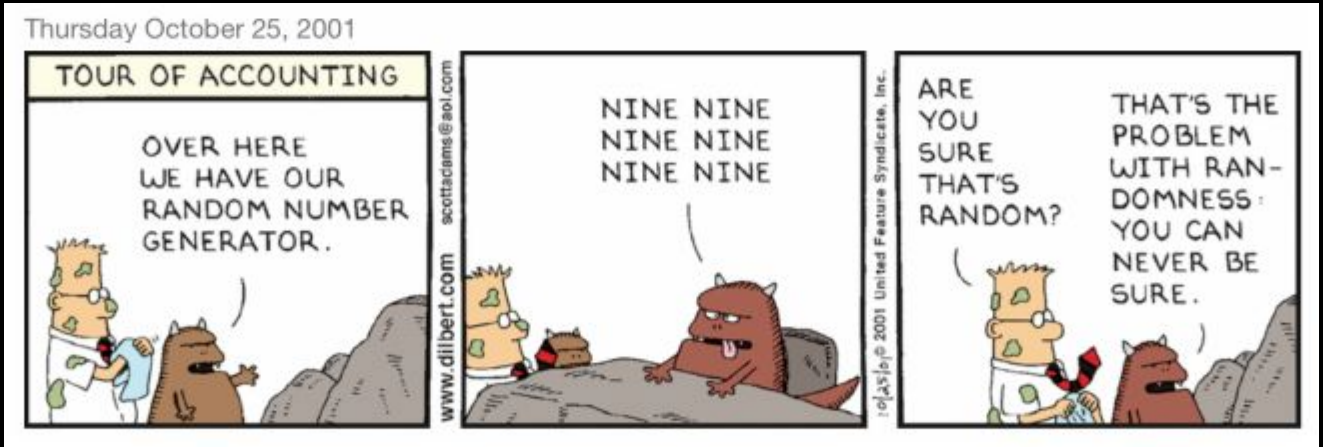
95-99% accuracy

This is not very good, you can get very high success rate guessing EXPLOITABLE all the time - be correct 96% of your guesses

RESULTS - XGBoost

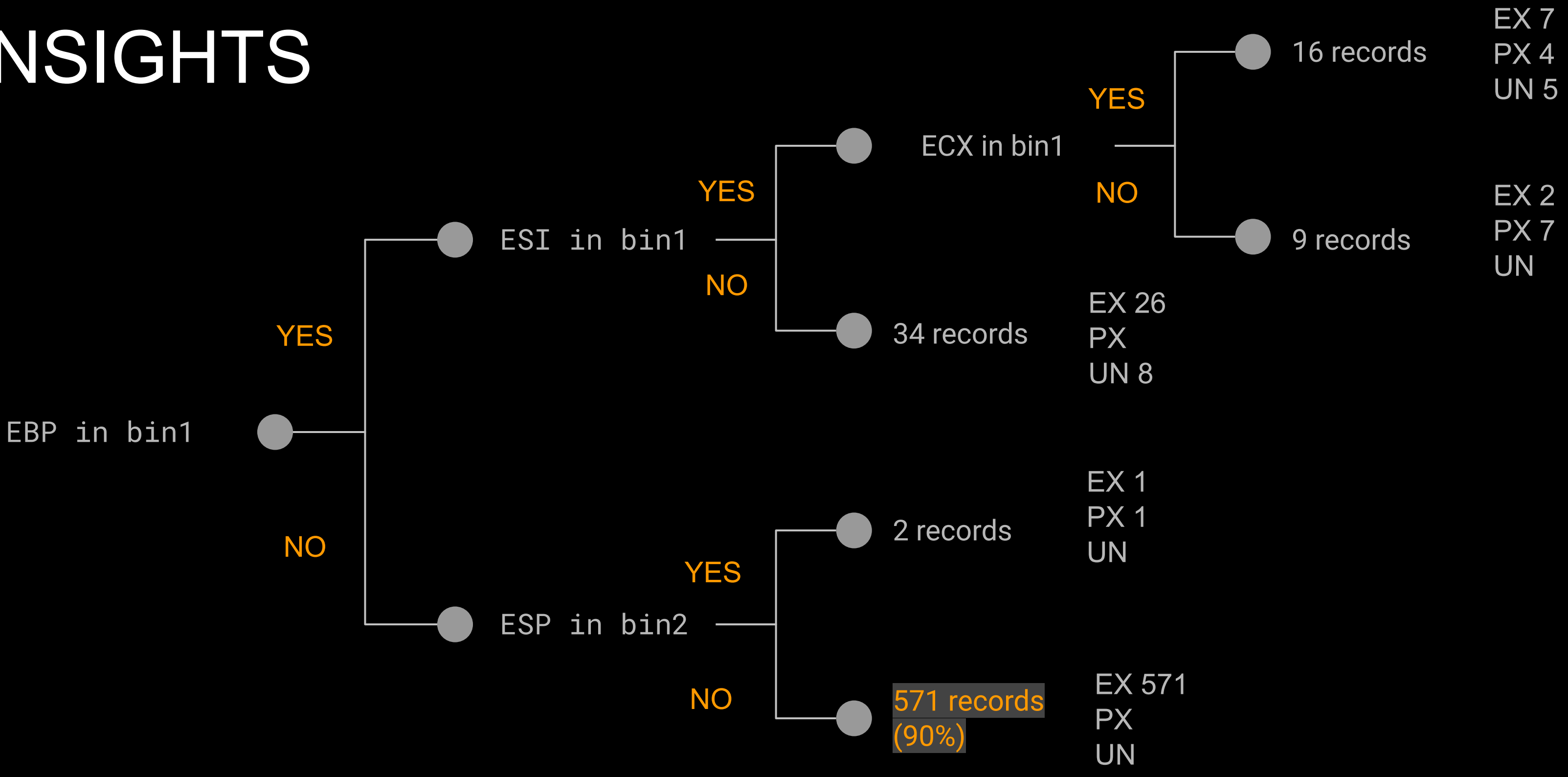
95-99% accuracy

This is not very good, you can get very high success rate guessing EXPLOITABLE all the time - be correct 96% of your guesses


Thursday October 25, 2001
TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
scottadams@aol.com
www.dilbert.com
NINE NINE NINE NINE NINE NINE
© 2001 United Feature Syndicate, Inc.
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS: YOU CAN NEVER BE SURE.

@barnhartguy @aCaltum

INSIGHTS



EBP in bin1

YES

ESI in bin1

YES

ECX in bin1

YES

16 records

EX 7
PX 4
UN 5

NO

9 records

EX 2
PX 7
UN

NO

34 records

EX 26
PX
UN 8

NO

ESP in bin2

YES

2 records

EX 1
PX 1
UN

NO

571 records (90%)

EX 571
PX
UN

@barnhartguy @aCaltum

RULE OF THUMB?

For 571 (90%) of our records, it is enough to test:
!(EBP in bin1) & !(ESP in bin2) to classify it as EXPLOITABLE

Does this make any sense?

Will this remain true with more data?

COMPARISON AGAINST `exploitable`

Built and tested against a set of heuristics - works very well

Out method shows that we can perform as well or better against the same data set

However, we need more data to give any certainty to these claims

HOW TO BUILD THIS YOURSELF

We released a whitepaper to explain our methodology and results

https://www.productsecurity.info/files/Whitepaper_SAS19.pdf

More research, and especially more data is needed!

CONCLUSIONS

ML is only as good as your dataset, you’re answering “how similar”

This is still a work in progress.

We don’t have enough non-exploitable crashes to test against

The insights we gathered are interesting, and merit a deeper look when more data is available

WHERE CAN WE USE THIS?

Feedback for bug trackers (impact/importance)

Feedback for vuln hunters - focus areas

Feedback for fuzzers - where to focus

MORE INSIGHTS

Data science is an art

We need to talk with people from different disciplines than us

ACKNOWLEDGEMENTS

Denis Klimov (PhD), Intel

Caswell, Brian, Lunge Technology - Cyber Grand Challenge Corpus

`exploitable` - https://github.com/jfoote/exploitable

Thank You!

@barnhartguy
@aCaltum