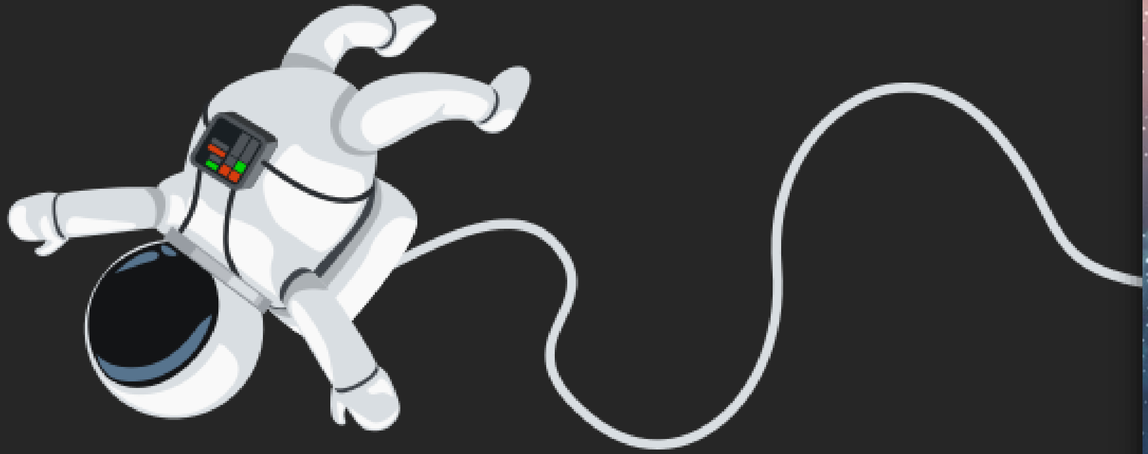
42: The answer to **life**, the **universe**, and **everything** **offensive security**

Who Are We?

Nick Landers
monoxgas



Will Pearce
moo_hax

Work at **Silent Break Security**

Research, Dev, Training, Ops, etc.

What is Machine Learning?

- Lots of magic that:

  gets investors, goes public, makes $

- Rebranded if/else statements
- Data Sheet Keywords:

  “Analyzes millions of X and adapts”

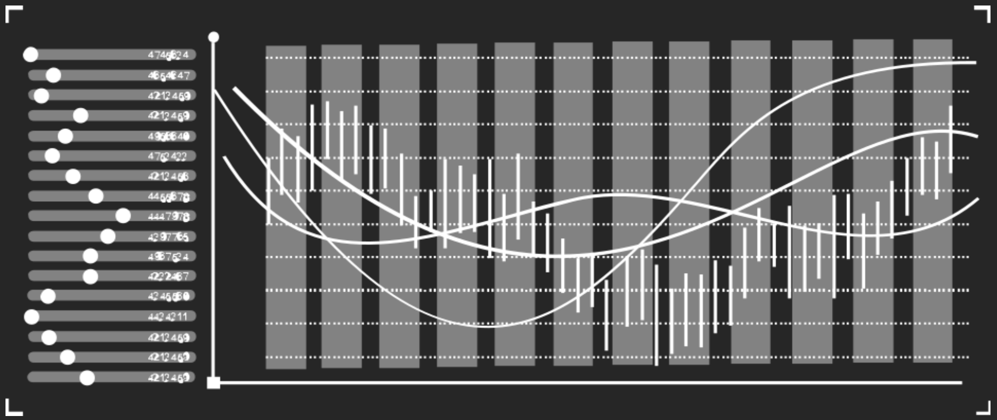
  “High degree of confidence”

  “Next Generation”

  “Scientists”

  “Central Cortex”

  “Neural Network”



What **really** is ML?

- Set of techniques that aim to:

  model a problem mathematically
- Stats + Maths + Computers
- Very old discipline – 1950’s (IBM)
- Predictions without explicit programming
- **Growing fast:** computing power, data aggregation, etc.

... Still magic ... But **mostly math**

Why Do We Care?

- It's coming to a field near you
  - No longer a math problem, it's an engineering problem
  - Our future will be fought with ML
- It can be really **really** awesome
  - Building relationships in non-congruent data
  - Bring out operator 6th senses
  - Crush huge amounts of data faster than humans
  - Can be as complex or as simple as you want to make it
  - Optimizing a traditionally "manual" line of work

What is “Offensive” ML?

“The application of ML to offensive security problems”

- Abusing control relationships * [**Neo4j/Kegra**]
- Obfuscating C2 as English [**Markov Obfuscate**]
- Detecting sandbox environments * [**Deep Drop**]
- Improving social media phishing [**SNAP_R**]
- Faster password guessing [**PassGan**]
- Metasploit exploit selection [**Deep Exploit**]
- Stealing models for evasion * [**Adversarial ML**]
- Automating timing attacks [**ParzelSec**]

* - to be discussed


Collision
Course

Starting with ML

Google “ML [literally anything] tutorial”

“Detecting Cats in Images with OpenCV”

“Auto-Generating Clickbait with RNNs”

https://github.com/ujjwalkarn/Machine-Learning-Tutorials/ (320+ links)

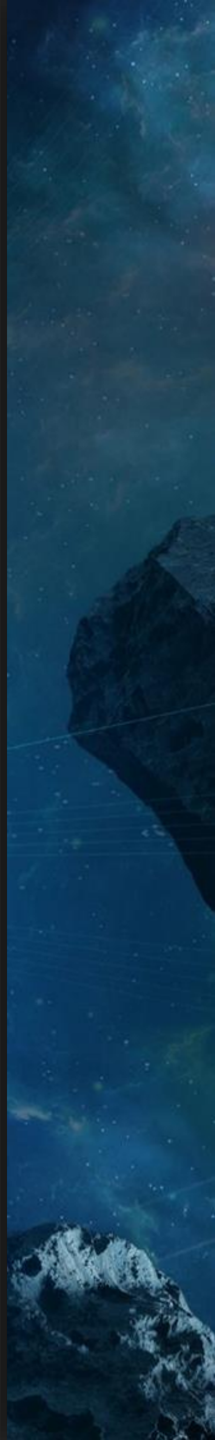
https://sgfin.github.io/learning-resources/ (200+ links)

https://github.com/josephmisiti/awesome-machine-learning (200+ links)

https://github.com/awesomedata/awesome-public-datasets (410+ links)

Starting with ML

1. Data is everywhere, what ~~can~~ should we collect?
2. What data is used by a human to solve/perform X?
3. Process the data and extract useful features.
4. Download Python + [ML stuff]. (Might need some GPUs)
   - NumPy / Pandas – Data processing and matrices
   - SciKit-Learn – Data analytics + Basic ML
   - TensorFlow – Full blown ML framework from Google
   - Keras – High-level wrapper for TensorFlow (also CNTK, Theano)
5. Write a 10 line script and ML your heart out.



Sandbox Detection – Case Study

- Sandboxes are a dangerous place
  - Popularity is rising
  - Preventing analysis is a priority
- Lots of current detection strategies
  - Enumerating host information
  - Automated behavior indicators
  - Network anomalies – server and client

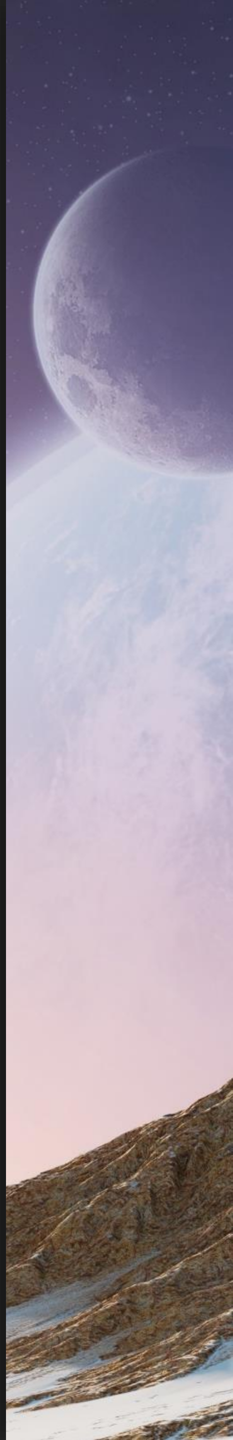
    ...

Need detection using minimal information

ML Concepts

Data – The raw information we gather

PID	User	Process
1	NT AUTHORITY\SYSTEM	smss.exe
236	NT AUTHORITY\SYSTEM	csrss.exe
120	NT AUTHORITY\SYSTEM	winlogon.exe
492	Admin-PC\Admin	explorer.exe
940	Admin-PC\Admin	procmon.exe
680	Admin-PC\Admin	dllhost.exe
772	Admin-PC\Admin	winword.exe



ML Concepts

Features – How we represent data to an algorithm

PID	User	Process
1	NT AUTHORITY\SYSTEM	smss.exe
236	NT AUTHORITY\SYSTEM	csrss.exe
120	NT AUTHORITY\SYSTEM	winlogon.exe
492	**Admin-PC\Admin**	explorer.exe
940	Admin-PC\Admin	**procmon.exe**
680	Admin-PC\Admin	dllhost.exe
772	Admin-PC\Admin	winword.exe

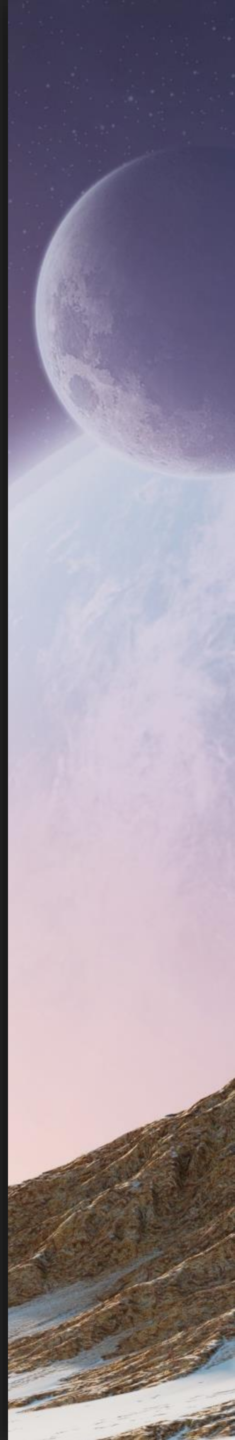
Feature	Value
bad_user:	1
sysinternals:	1
domain_member:	false

ML Concepts

Features – How we represent data to an algorithm

PID	User	Process
1	NT AUTHORITY\SYSTEM	smss.exe
36	NT AUTHORITY\SYSTEM	csrss.exe
312	NT AUTHORITY\SYSTEM	winlogon.exe
444	ACME\arthur.dent	explorer.exe
452	ACME\arthur.dent	winword.exe
1972	ACME\arthur.dent	chrome.exe
2928	ACME\arthur.dent	chrome.exe

Feature	Value
proc_count:	7
average_pid	906
compression:	85%
proc_per_user:	3.5



ML Concepts

Inputs – Our features for each sample

```
data = np.array([
        [33, 4, 8.25],
        [157, 1, 157],
        [195, 1, 195],
        [30, 4, 7.5],
        [34, 4, 8.5],
        [84, 1, 84]
])
```

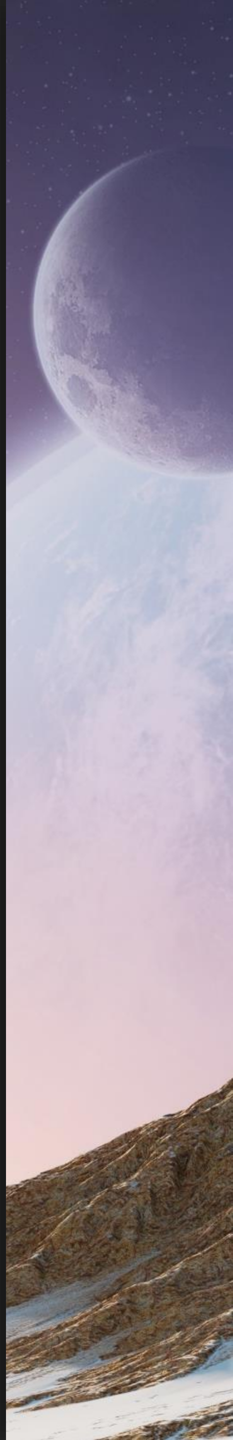
ML Concepts

Label – The thing we are predicting *

```
data = np.array([
        [33, 4, 8.25, 1],
        [157, 1, 157, 0],
        [195, 1, 195, 0],
        [30, 4, 7.5, 1],
        [34, 4, 8.5, 1],
        [84, 1, 84, 0]
])
```

0 = safe
1 = sandbox

* - we manually label data for training



ML Concepts

Classification – The strategy for assessing outputs

Binary: Grouping into 2 fixed categories

Multi-Label: Grouping into N categories

Regression: Targeting a continuous variable value

(1 – 999)

* - we usually have labeled data for training

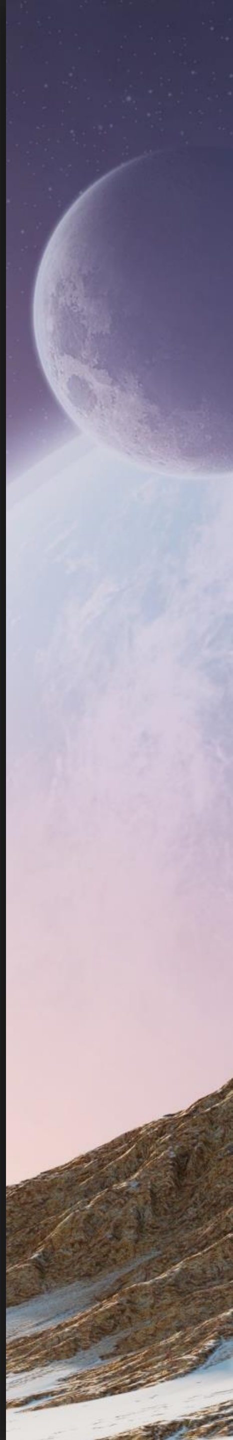
ML Concepts

Node – Smallest unit to carry an activation

* Calculated using an activation function

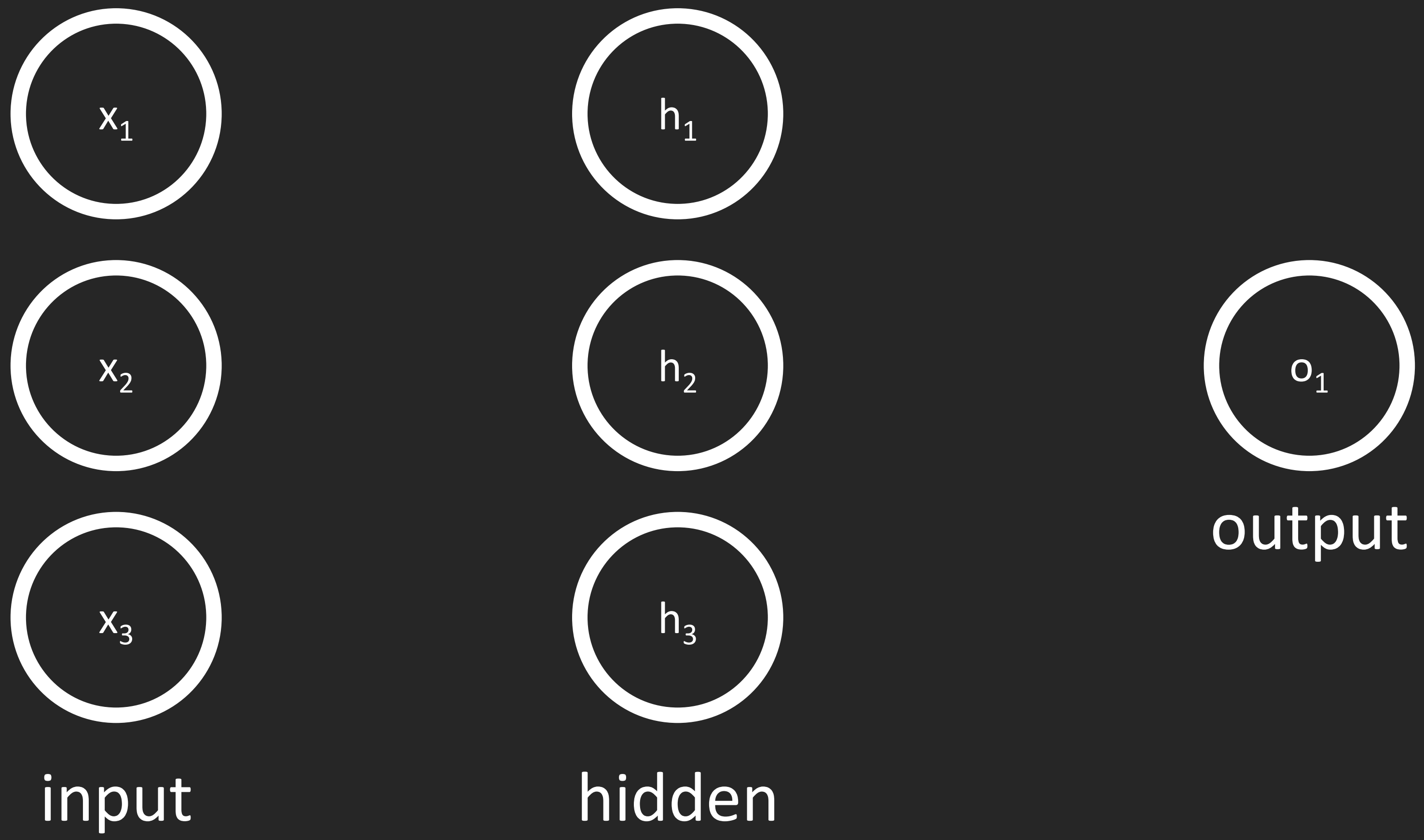
(like sigmoid)

$$\frac{1}{1 + e^{-x}}$$



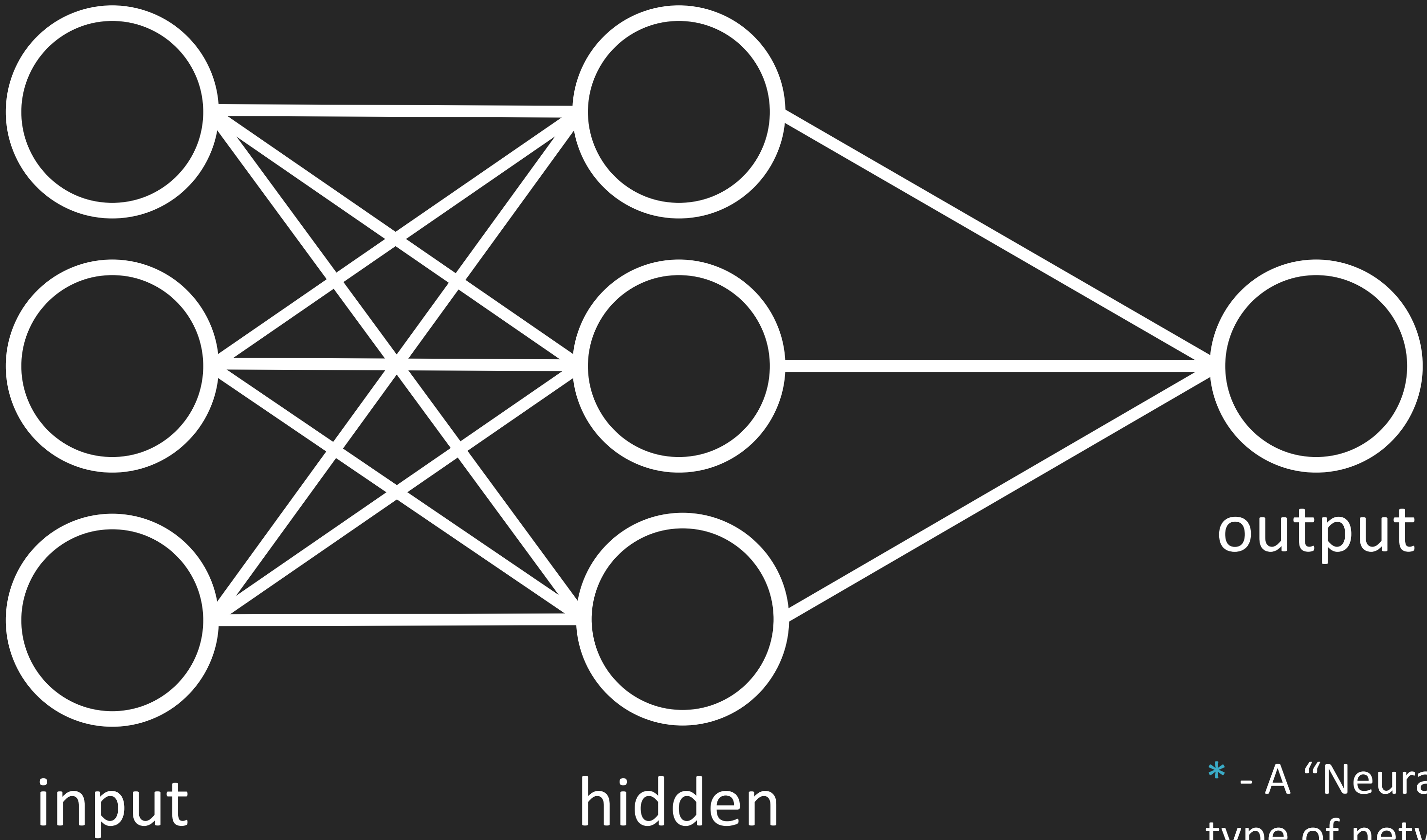
ML Concepts

Layer(s) – A parallel group of nodes



ML Concepts

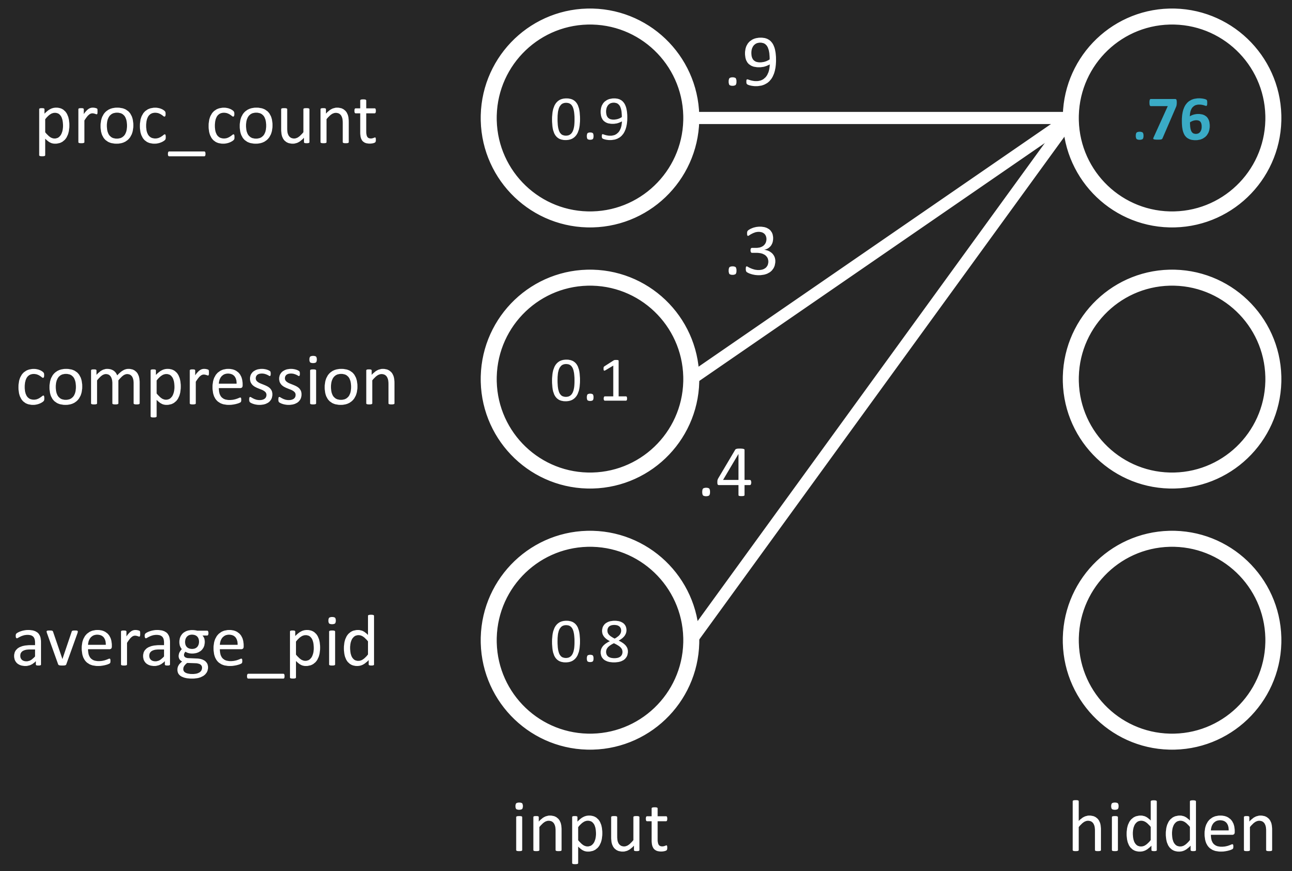
Network – A few layers strapped together *



* - A “Neural Network” isn’t the only type of network

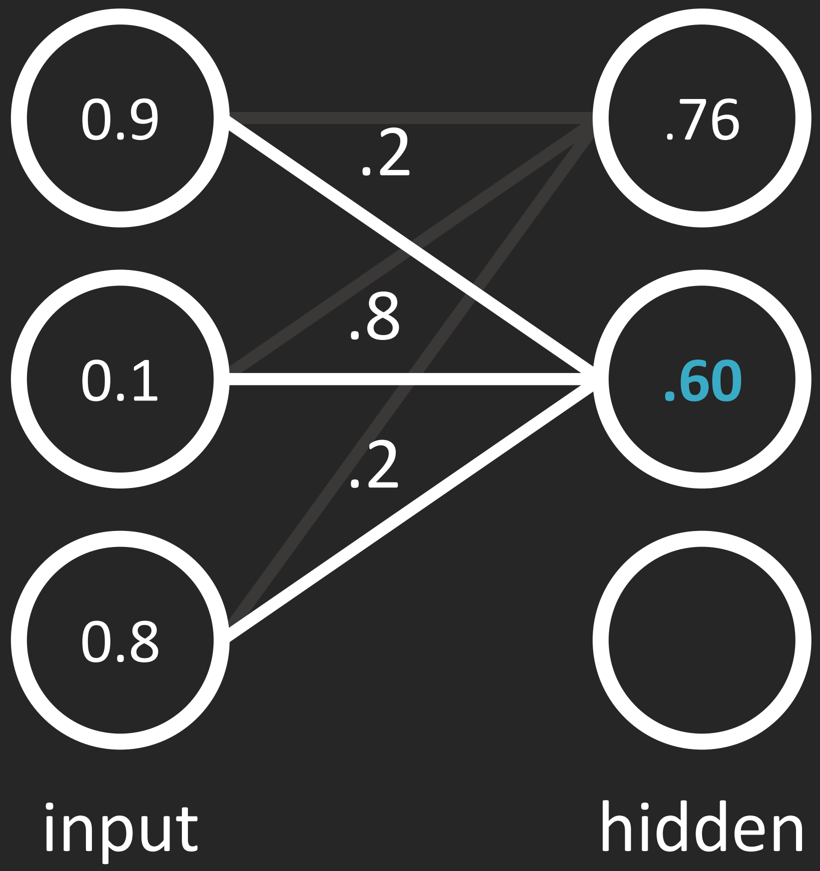
ML Concepts

Feed Forward – calculate the error of the network



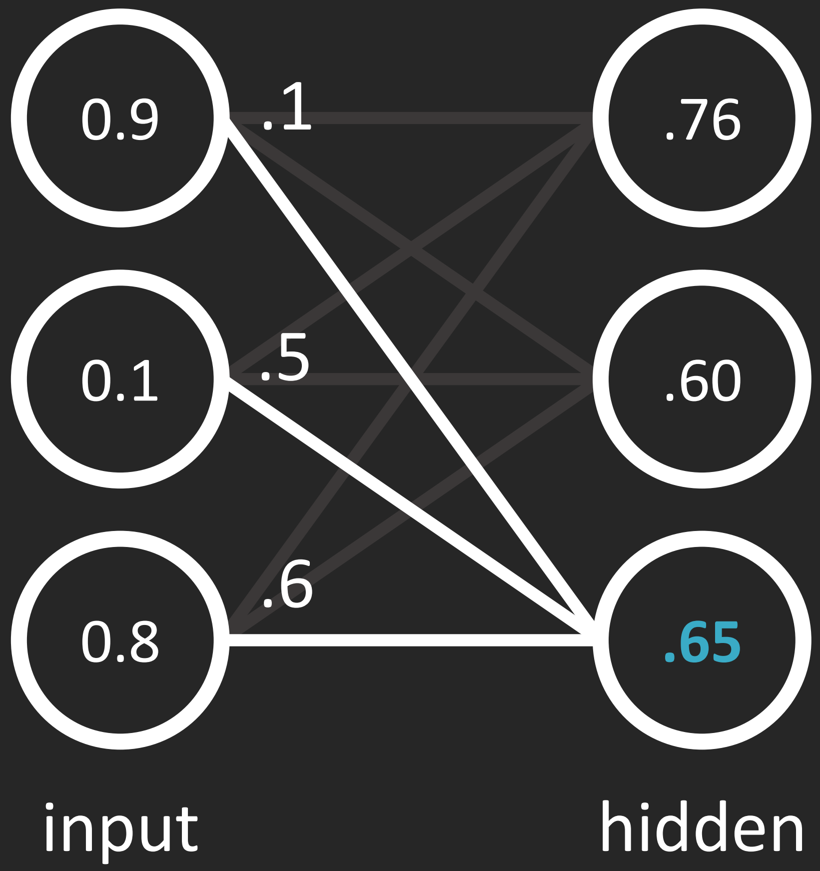
ML Concepts

Weighted sum is passed into the activation function



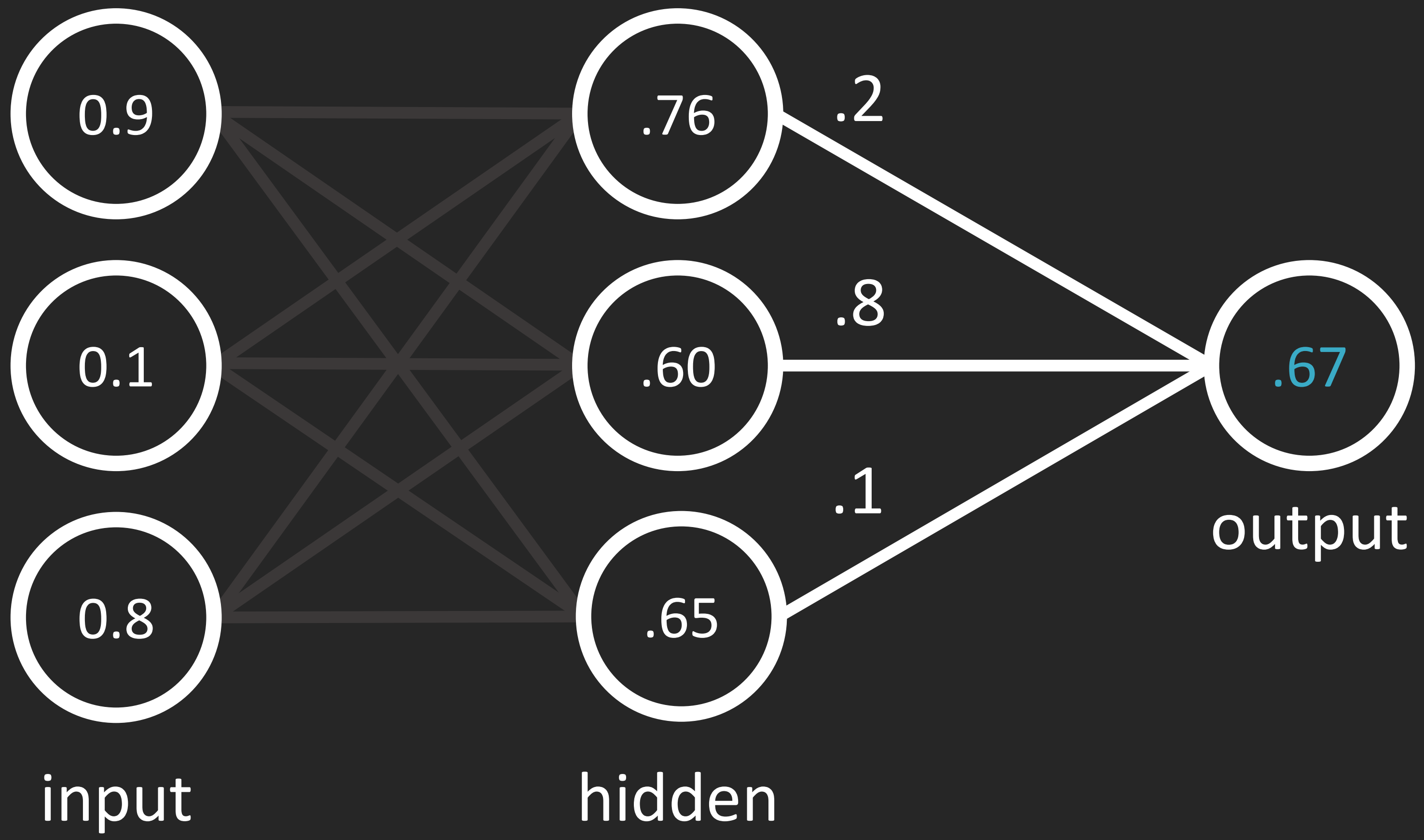
ML Concepts

Activations become inputs for the next layer



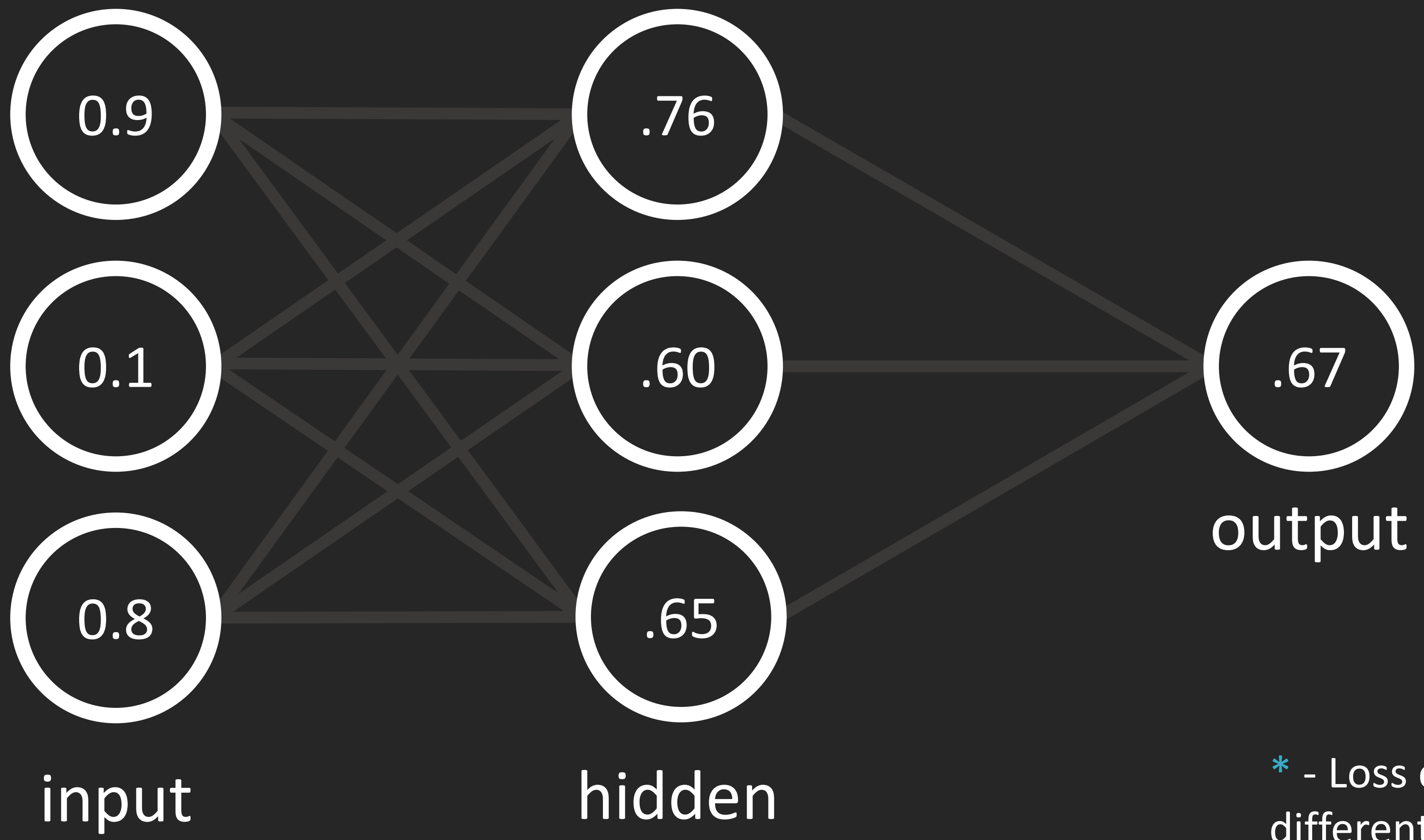
ML Concepts

Output – Final result of cascading activations



ML Concepts

Loss - The gap between the target label and output *



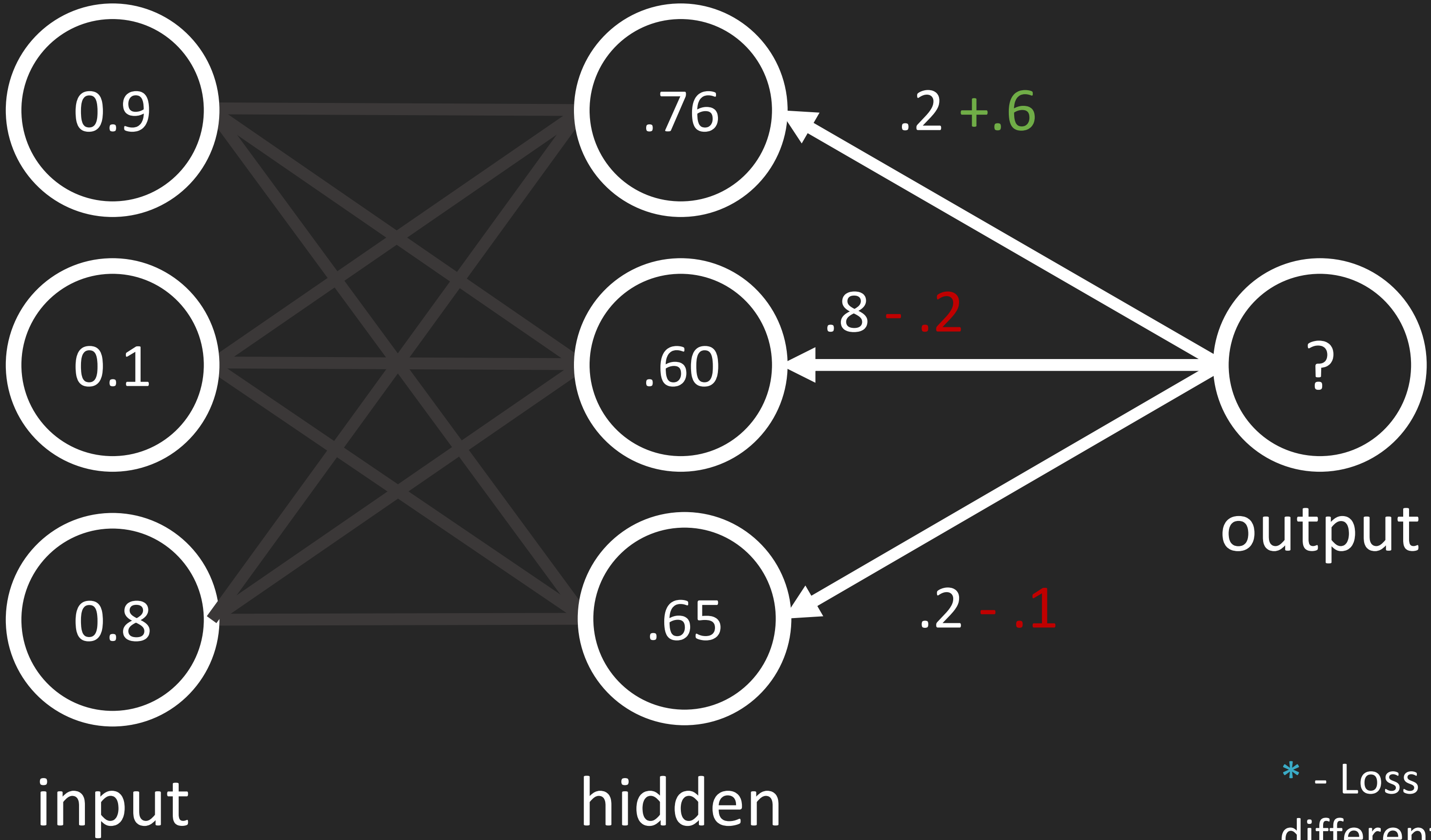
target: 1

loss = 0.33

* - Loss can be calculated many different ways

ML Concepts

Backpropagation – Updating weights to improve * (“learn”)



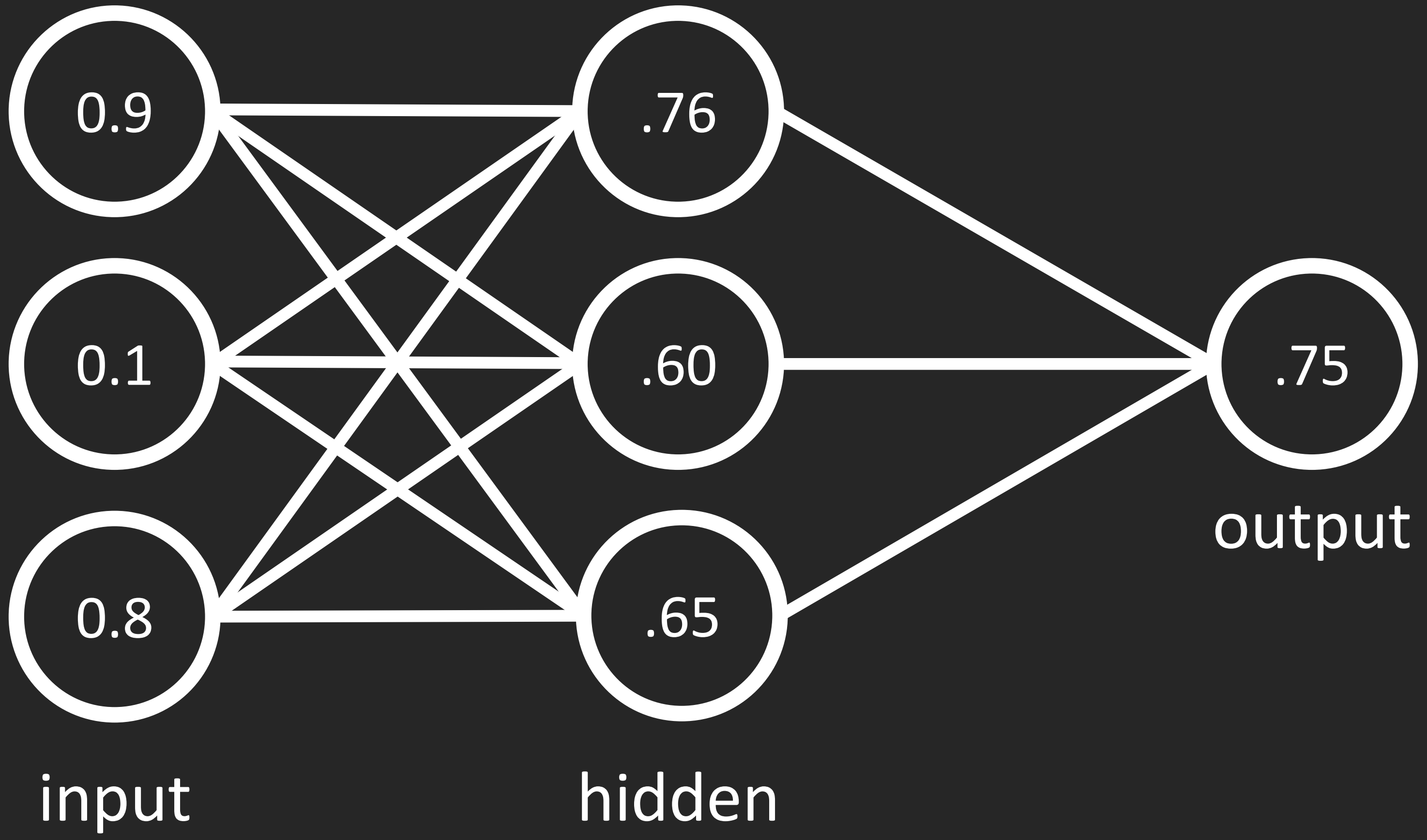
target: 1

loss = ?

* - Loss can be calculated many different ways

ML Concepts

Model – Network with learned weights



target: 1

~~loss = 0.33~~

loss = 0.25

Acc. + 0.08

Code – Neural Network

```
dataset = np.loadtext(features.txt)
features = dataset[:,0:3]
labels = dataset[:,3]

model = models.Sequential()
model.add(layers.Dense(3, activation="relu", input_dim=3))
model.add(layers.Dense(3, activation="relu"))
model.add(layers.Dense(1, activation="relu"))
model.compile(loss="binary_crossentropy", optimizer="adam")
model.fit(features, labels, epochs=10, batch_size=10)
```

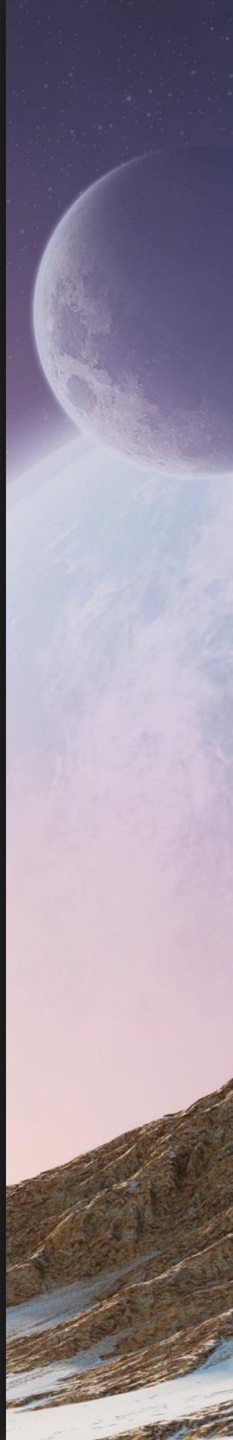
The Results

Total Samples: 130

Known Sandboxes: 18

Accuracy: 95%+

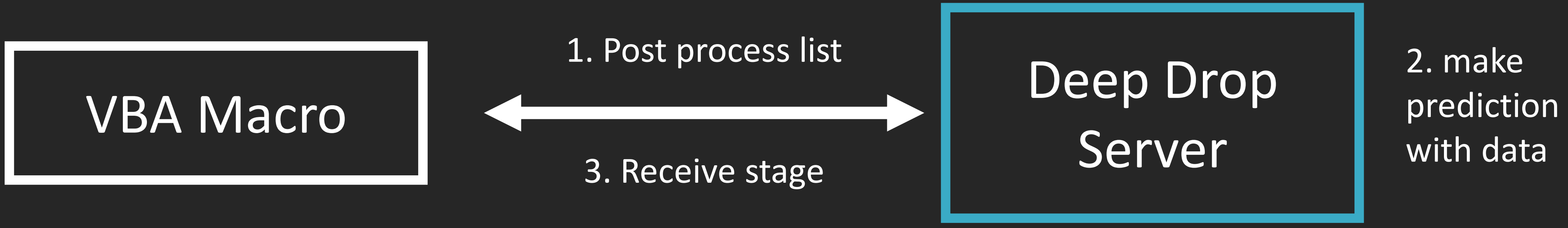
- Lots of options for feature combinations
- Ultimately a great problem for ML
- Doc2Vec / Decision Trees performed well too



Deep Drop - **ML-enabled dropper server**

https://github.com/MoooKitty/SchemingWithMachines/

- Initially released at BSides LV 19
- Holds a trained model for parsing outputs
- Written in Python + Flask + Keras
- Makes stage delivery decisions based on ML



Data Pipelines

Data Collection

Issue: Offenders traditionally don’t keep data

- Nobody is talking about this issue
- Should this remain an expectation?
- How can/should we anonymize data?
- Why are vendors allowed to keep data? (then sell it back)
- **Would sharing datasets further our field?**

Potential Solutions:

- Keep only model weights
- Hash features for storage
- Use models that don’t require previous data

Textual Data

- Defenders already work with parsed data
  - SIEM collection and alerting
  - E-Mail report buttons
  - AMSI integrations
  - Known environments
- Offenders are still in a textual world
  - Command line interfaces
  - Screen/Session logs
  - ASCII art for every tool
  - Reports and narratives

Feature Engineering

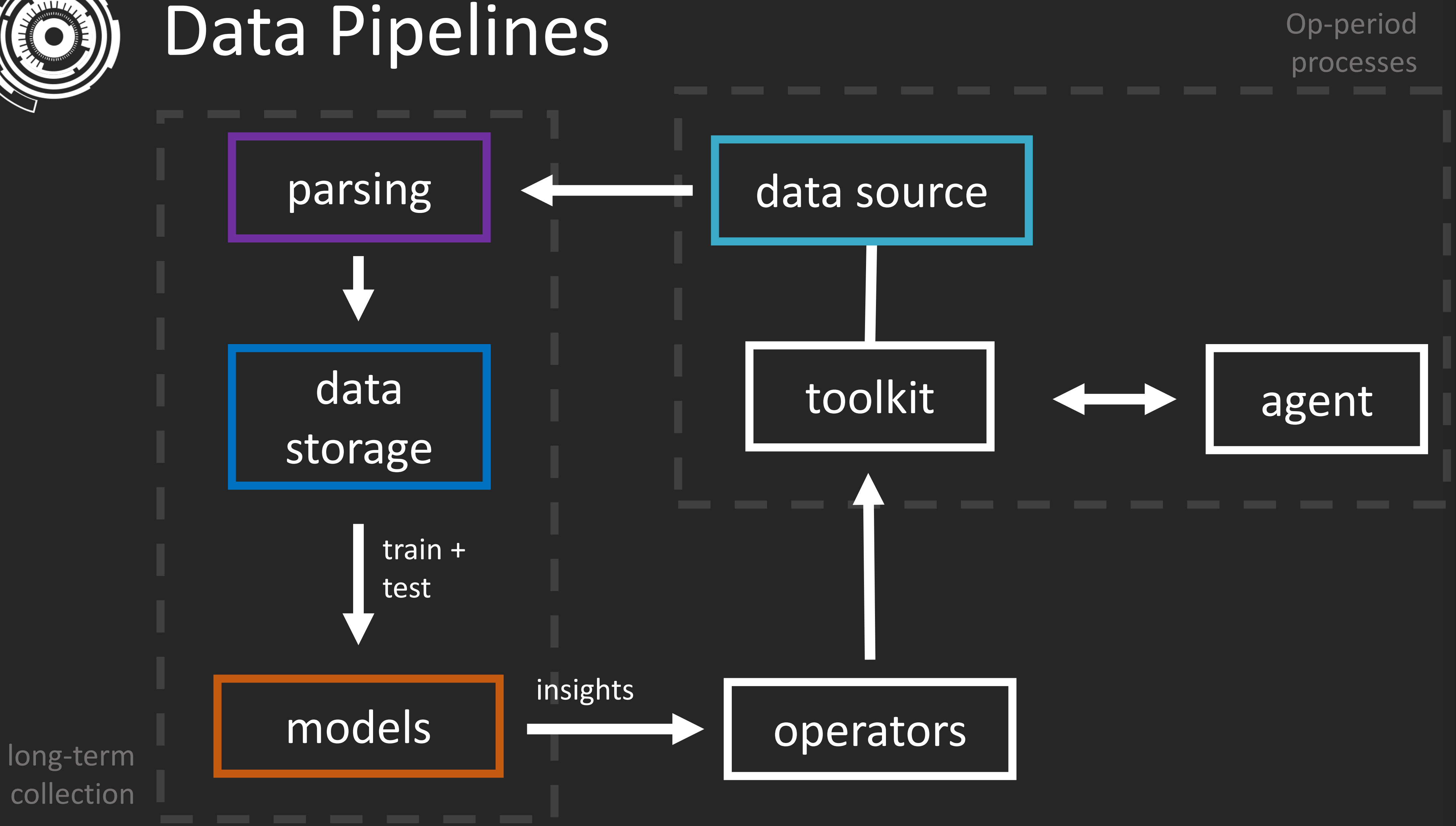
- Target important meta-properties **– domain knowledge**
- Store as much as possible & build features later
- Use data analysis to assist
  - “What does the distribution of commands look like?”
  - Do new features line up with existing labels?
  - Reduce correlated features - “noise”
  - Once trained, find features which don’t affect outputs

Essentially ... Lots of **basic statistics**

Data Pipelines

- Need systems for managing our data – long / short term
- This processing requires engineering
  - Start early to learn later
  - Implementations will vary by team TTPs
  - Solution will likely tie us down – agnosticism
  - Ideally passive – not interfering with ops
- Focus on high impact data
- Previous works on the subject:
  - https://github.com/ztgrace/red_team_telemetry
  - https://github.com/outflanknl/RedELK
  - https://github.com/SecurityRiskAdvisors/RedTeamSIEM

Data Pipelines



Team Integration

- Development is a requirement
  - Helpful if you also have/modify tool source
  - 2019 is the “year of C2” – shouldn’t be a problem
- Identify isolated jobs to begin delegating
  - Basic classification that a human already does
  - Suggestions that can augment decisions
- Basic statistics for ops
  - Average number of actions per operation
  - Count/distribution of commands and arguments
- Consider trust in the final solution

Into the Unknown

State of Attack Paths

- Path finding with information is solved

  Neo4j + Bloodhound

- **However, information will degrade**
  - Changes to Active Directory / Windows
  - Growing use of *nix in business
  - Network segmentation improvements
- **Networks are unknown, but discrete**
  - We **don't know** the user names, permissions, etc.
  - We **know** it's not infinite

Data Inference

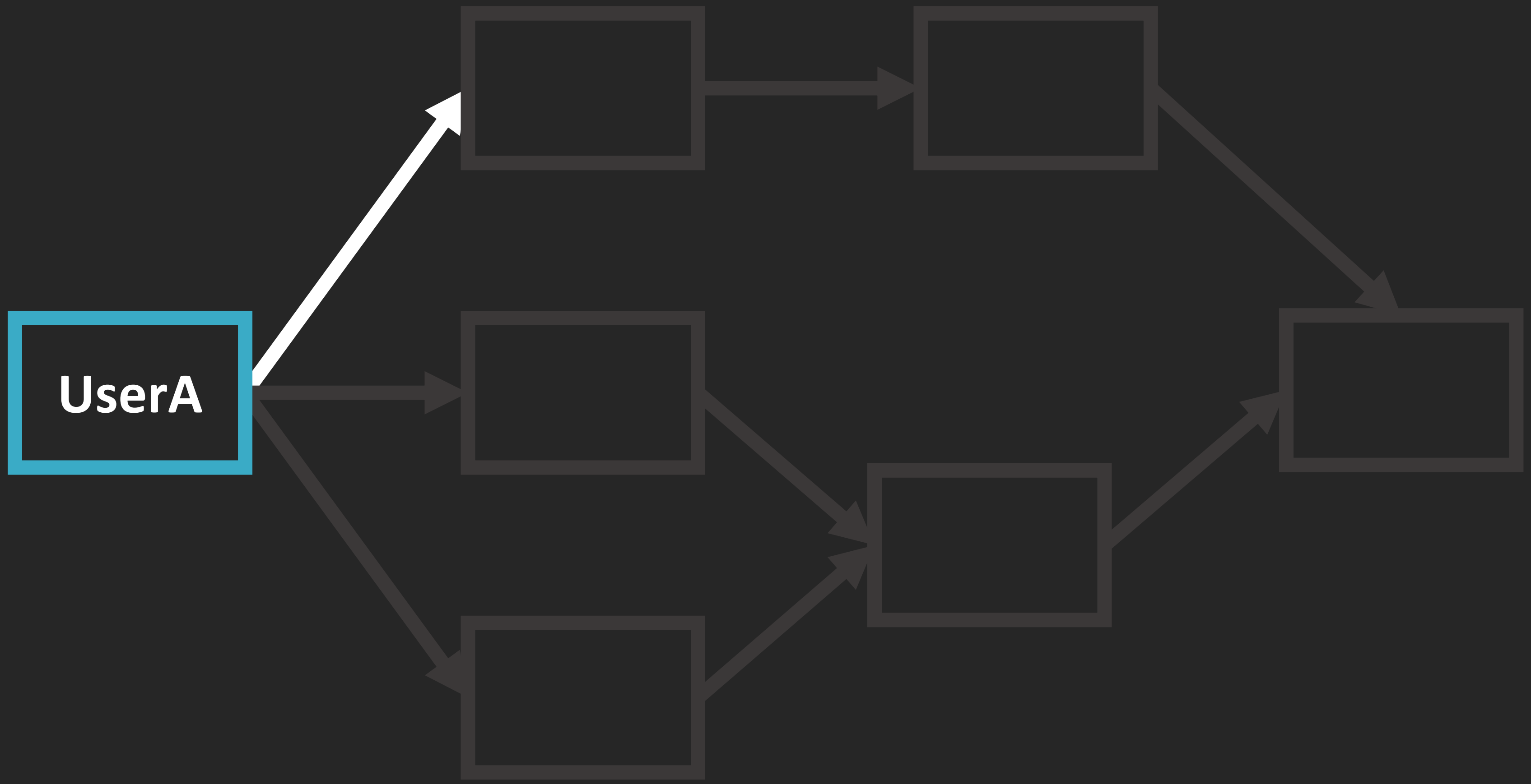
Networks **require** data organization

therefore

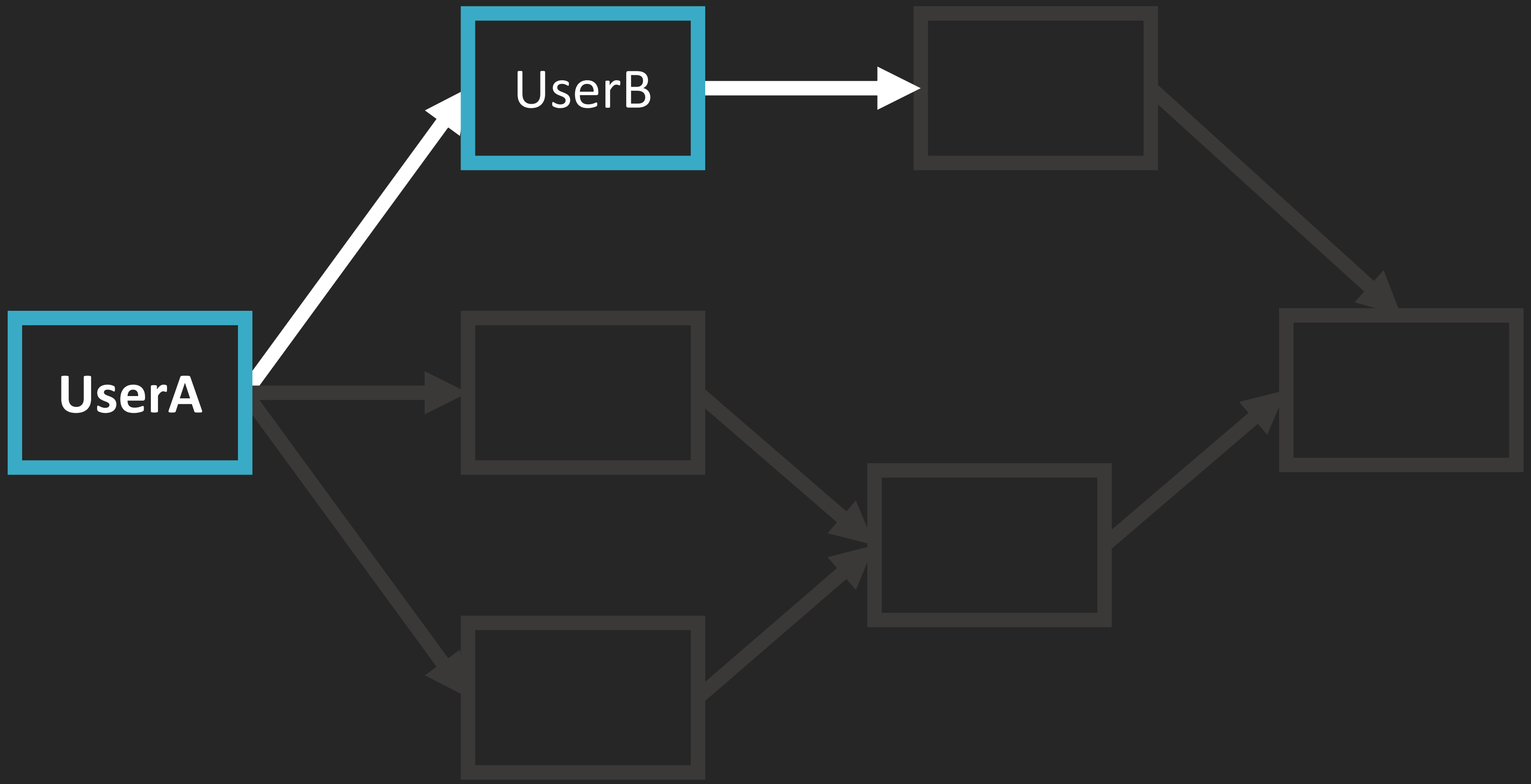
Networks **imply** data organization

- Networks (AD) generally use text-labeling
  - We’re all human, we expect it to be relational
- Can we infer these relationships?
  - Textual similarities
  - Mapped drives, local users, host information
  - LinkedIn, GitHub, public exposure ...

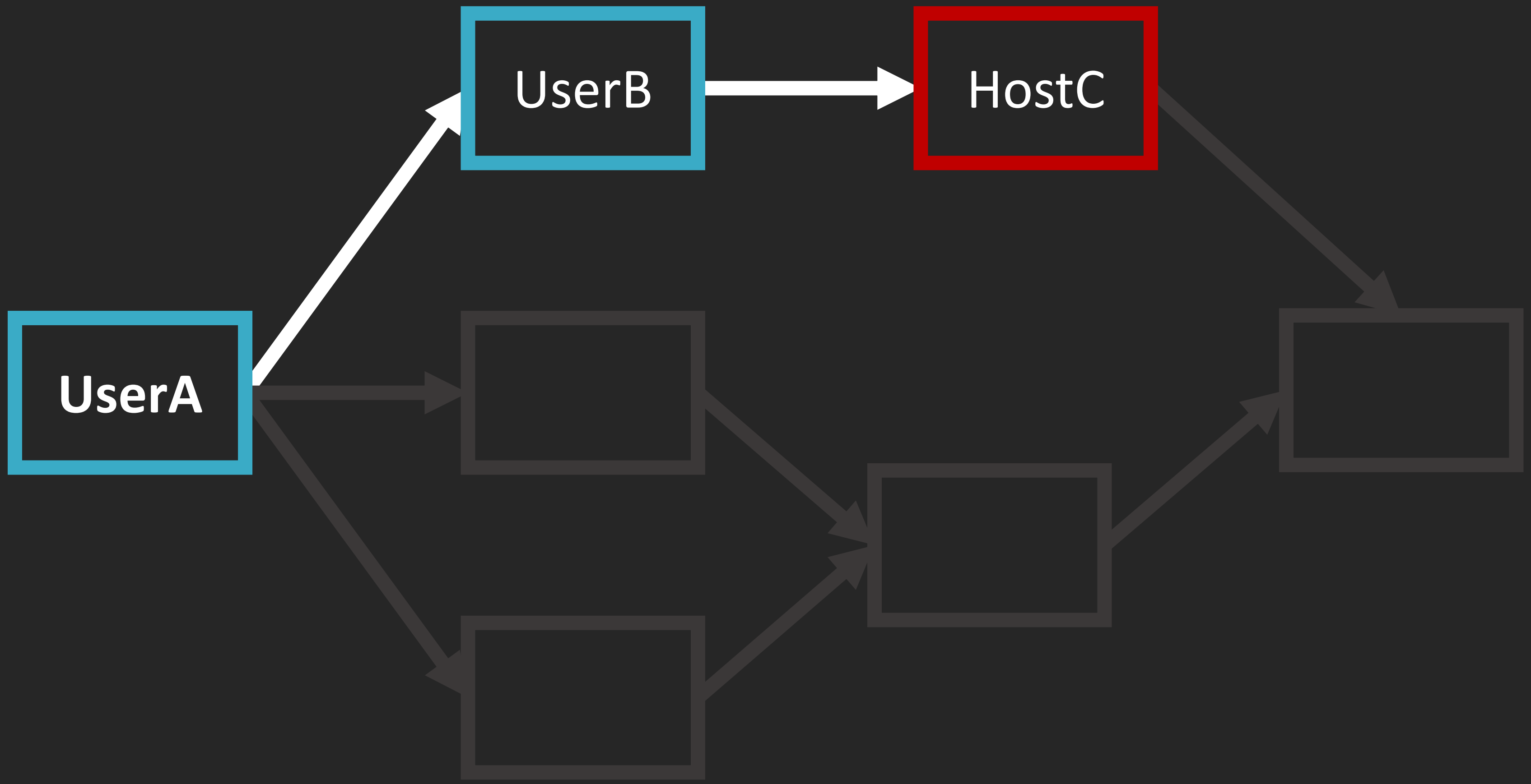
A Network of Probabilities



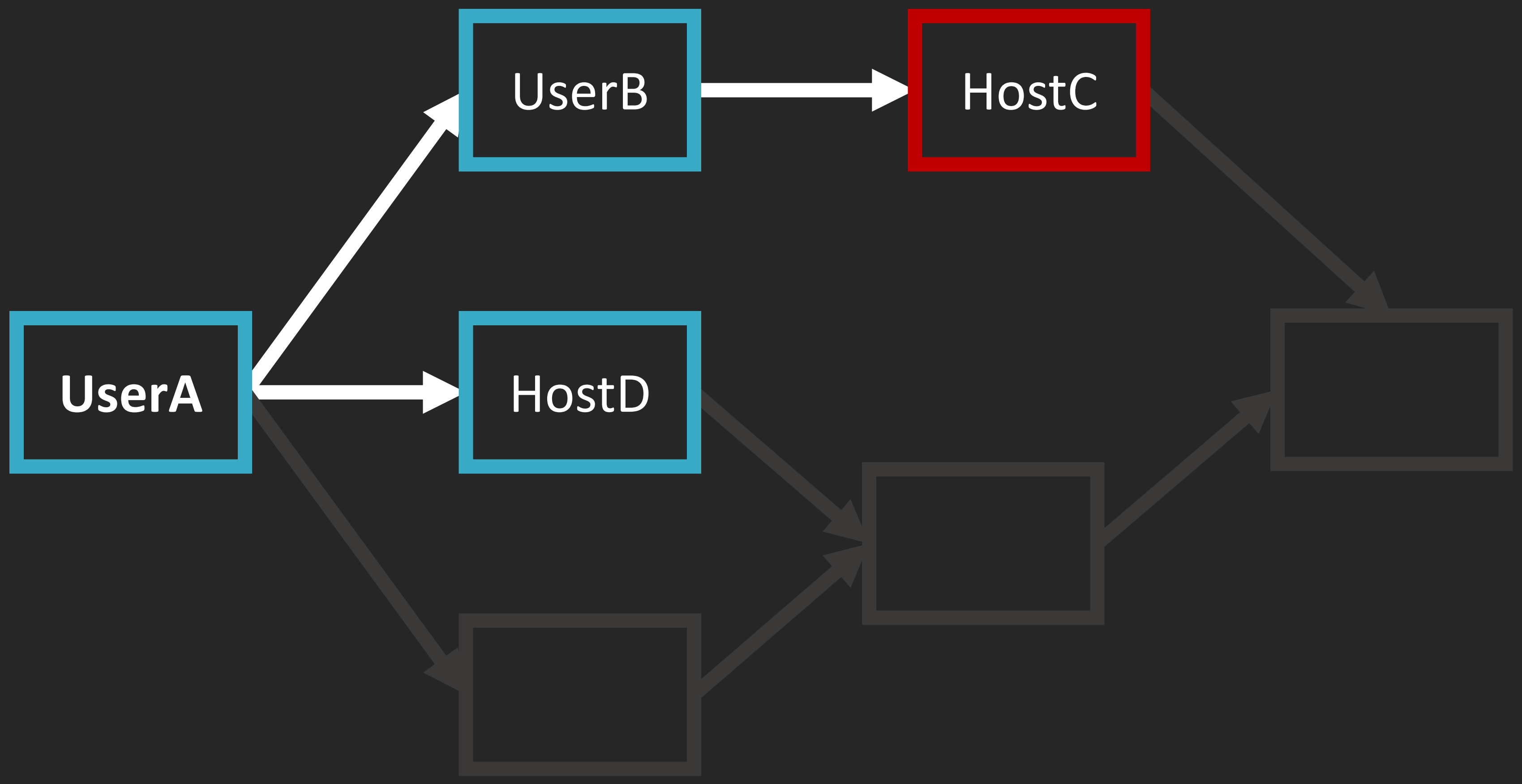
A Network of Probabilities



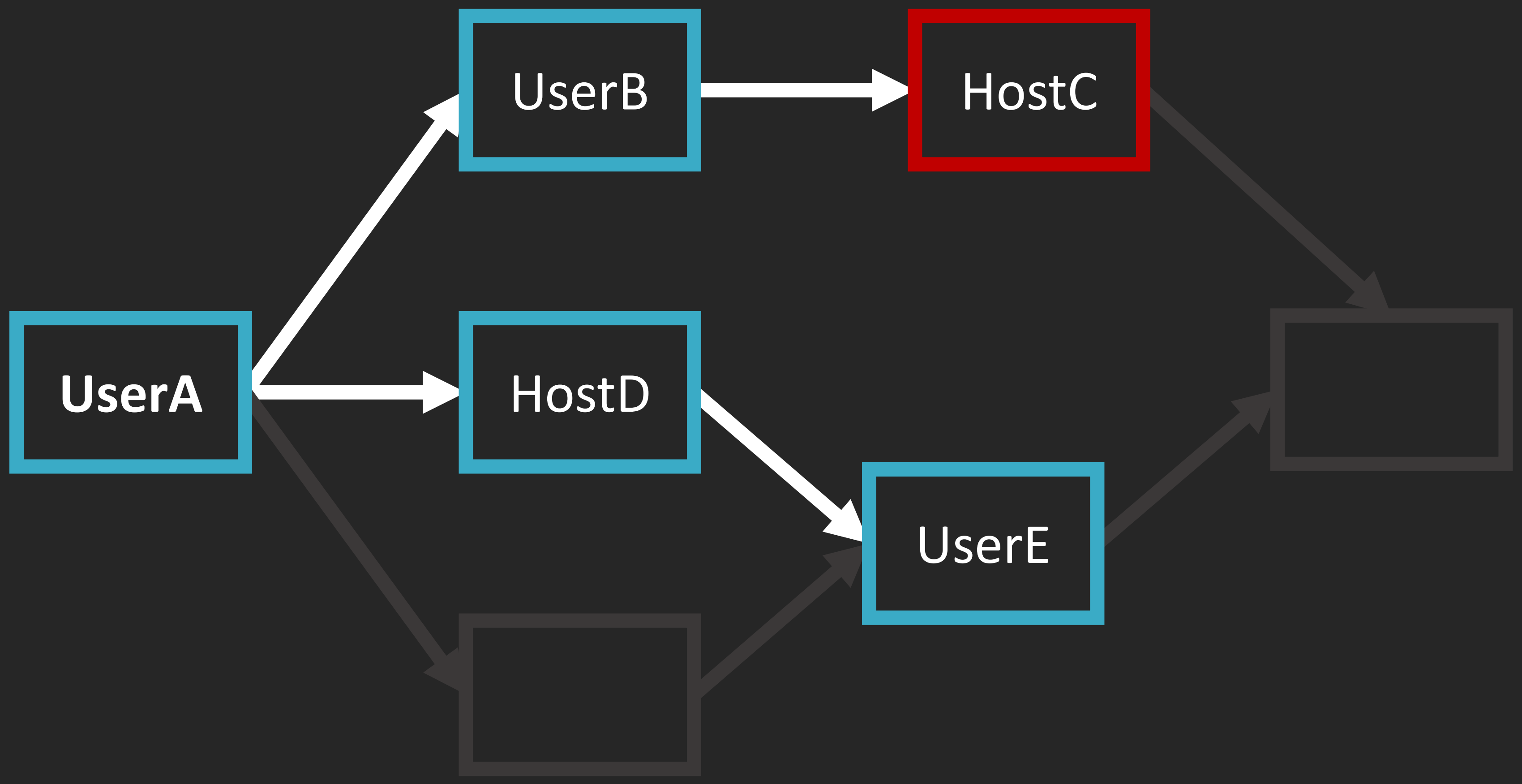
A Network of Probabilities



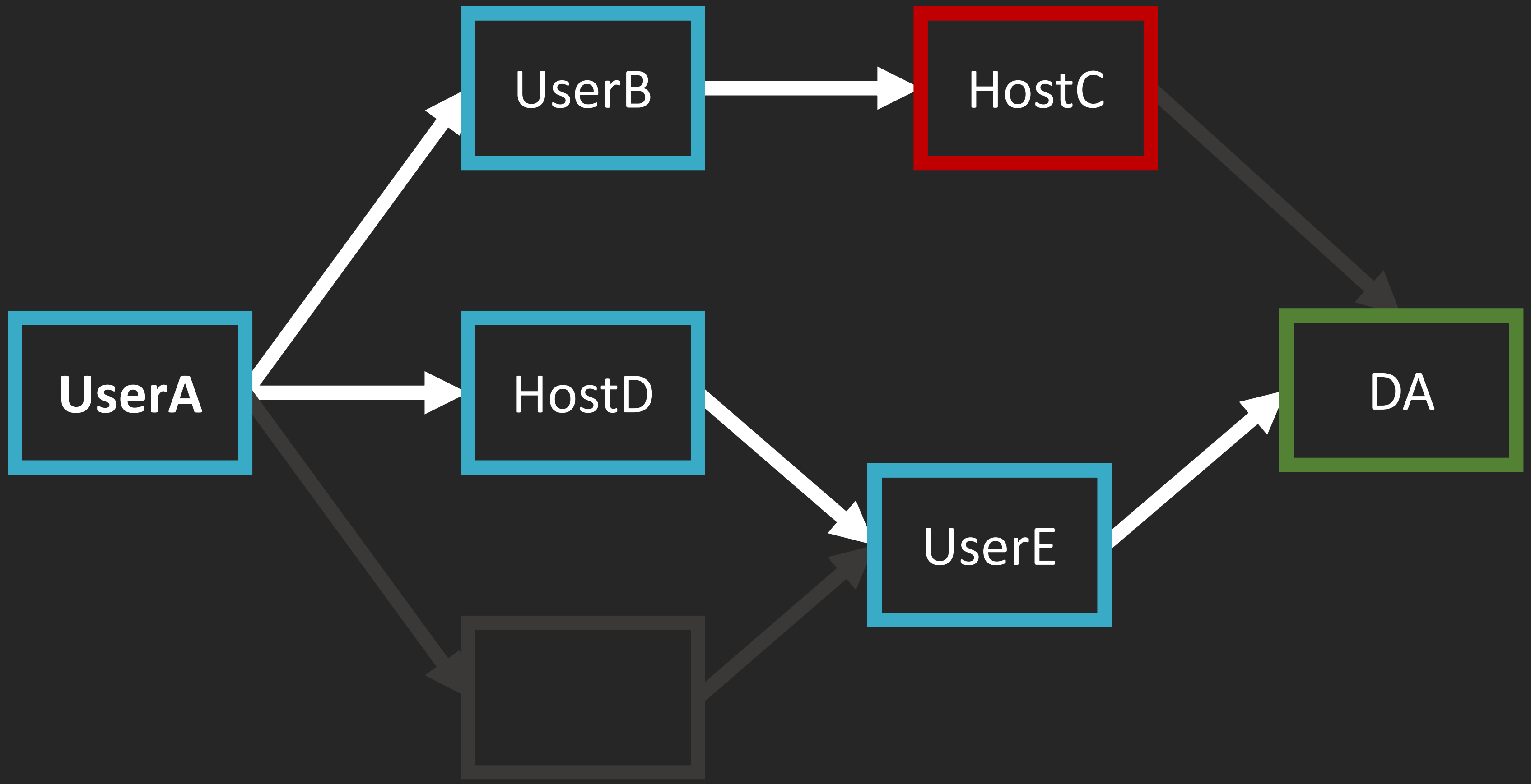
A Network of Probabilities



A Network of Probabilities



A Network of Probabilities



Mental Models

As operators, we build mental network maps

- We assume relationships based on:
  - Standardized textual labels
  - Experiences in the network itself
  - Pattern recognition – how **has** it been configured, how **will** it
- We act on these assumptions with queries
  - Validate access to a host
  - Verify the group membership of a user
  - Collect new attack surface via enumeration

Information creates confidence

Simulating Mental Models

Data + Heuristic Search + Simulation

- **Data** - Information from the current context
- **Heuristic** - Relationships between data points
  - Operator driven - flexible
  - We can “assume” relationships, or even new data points
  - Use algorithms to bring up the most relevant data
- **Simulation** - Select actions based on heuristics
  - Could include the “impact” of potential actions
  - Can assist an operator, or **become one** – active / passive

Data Layer

- Host/User/Group name information
- Active network connections
- Outbound RDP history
- Network queries
  - Active Directory with limited filters
  - Direct host service access
- Host-based Events
  - Event logs
  - ETW tracing
  - Custom tracking over time

Heuristics Layer

- How can we relate textual data?
- What strategies are used already?
- What would an operator care about?

We need some number to support our simulation

Heuristic: Simple ‘if’ Statement

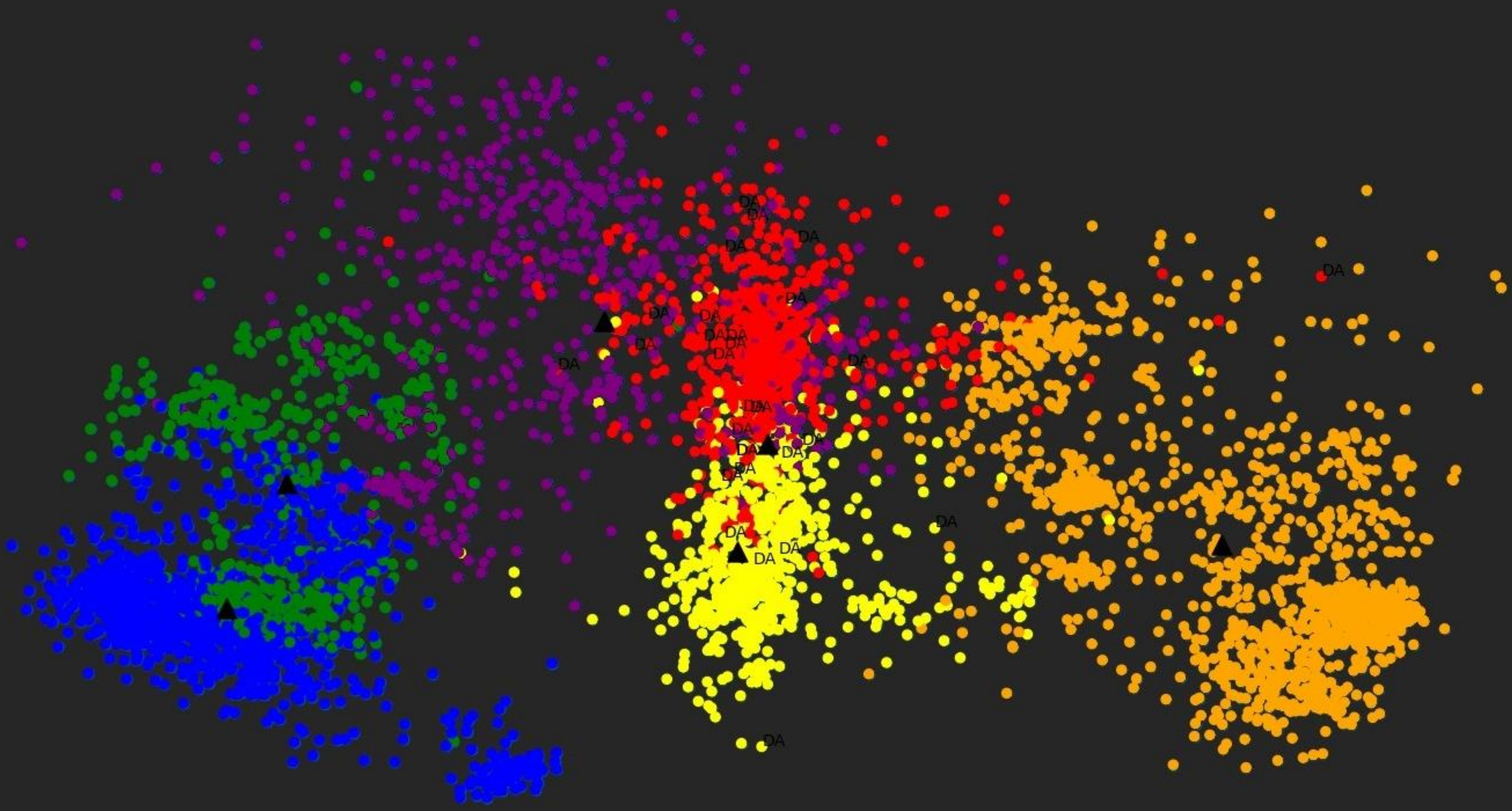
- Operator driven insights
- Doesn’t require complexity

```
if <output> in command_output:
        return new_state, reward
else:
        continue
```

Heuristic: Cosine Similarity

```
match (g:Group)
with collect(g) as groups
match (u:User)
with u, algo.ml.oneHotEncoding(groups, [(u)-[:MemberOf]->(memberof) | memberof]) as embedding
with {item:id(u), weights: embedding} as userData
with collect(userData) as data
call algo.similarity.cosine(data, {similarityCutoff: 0.7071, write: true, topK: 100})
yield nodes, p50, p75, p90, p99, p999, p100
match (u:User {name: "<USER>"})-[similar:SIMILAR]->(other)
return other.name as user, similar.score as score
```

Heuristic: Cosine Similarity



Heuristic: Levenshtein Distance

```
match (c:Computer)
match (g:Group)
with g,c, apoc.text.levenshteinSimilarity(g.name, c.name) as data
return g.name as group, c.name as host, data as score
order by score desc
limit 100
```

Heuristic: Levenshtein Distance

Group	**Hostname**	**Score**
SQL Developers	SQLDEV01	55
Domain Admins	DOM-PRINS	55
SQL Admins	SQLDEV01	44
VPN Users	DEVPN-B	38

Simulation: Shortest Path

Dijkstra's Algorithm

- Shortest distance from A->B
- See their awesome talks - the past 5 years
- Useful for map/path finding problems
- Currently what Bloodhound uses

not the only one we could use ...

https://github.com/andyrobbins/PowerPath

Simulation: Shortest Path

A-Star Algorithm

- Shortest distance from A->B + **Heuristic**
  - Helps us avoid particular paths
  - Ignore paths which might be unavailable – segmentation
  - Punish “noisy” paths

CALL **algo.shortestPath**.astar.stream((startNode:Node, endNode:Node, weightProperty:String, propertyKeyLat:String, propertyKeyLon:String,

{nodeQuery:'labelName', relationshipQuery:'relationshipName', direction:'BOTH', defaultValue:1.0})

YIELD nodeId, cost

Q-Learning for Automation

For a given action in a given state, the environment returns a new state, and a reward

- Basic reinforcement learning
- Allows an agent to learn optimal actions
- Strength is through the heuristic you use
- Requires some initial state – all weights are 0

New State: ?

Reward: ?

Environment

states:	\\SQLDEV01\c$	\\DEVPN-B\c$
Admin on SQLDEV01	**55**	0
Admin on DEVPN-B	0	**38**

New State: Admin on SQLDEV01

Reward: +55

Environment

Action:
ls \\SQLDEV01\\c$

Output:
Directory of ...

states:	\\SQLDEV01\c$	\\DEVPN-B\c$
Admin on SQLDEV01	55	0
Admin on DEVPN-B	0	38

New State: Shell on SQLDEV01

Reward: +500

Environment

Action:
Pivot to SQLDEV01

Output:
New Target on ...

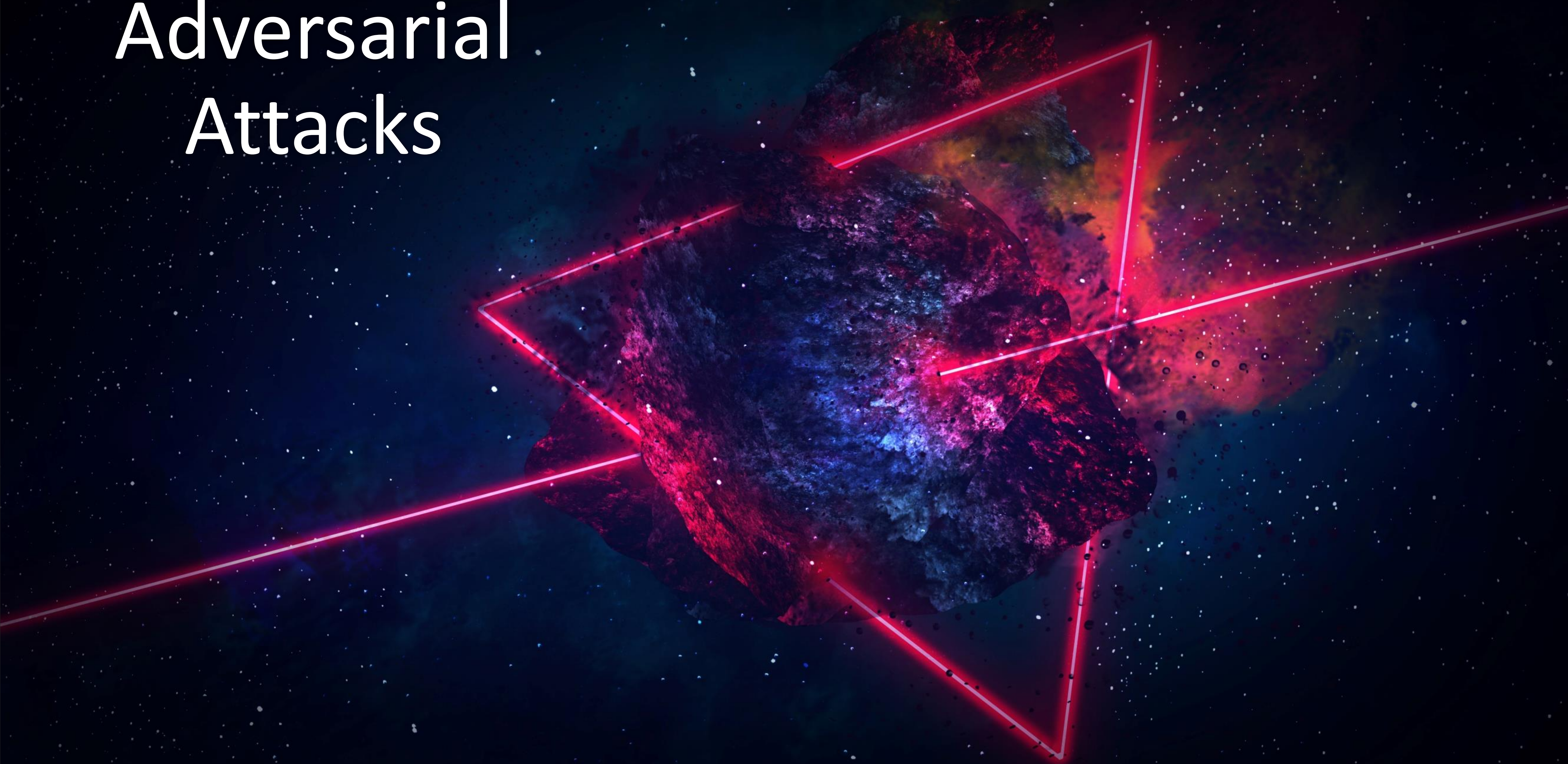
states:	...	Pivot to SQLDEV01
Admin on SQLDEV01		0
Admin on DEVPN-B		0
Shell on SQLDEV01		**500**
Shell on DEVPN-B		0

Back on the radar

Goal > **Data > Prior > Query > Posterior**

- We are all just data processors
- Attack graph theory is the future
  - Bloodhound/resiliency is like white-box code review
  - **Heuristic-based cyclic queries are black-box**
- With limited knowledge, we work in probabilities
- Given a sufficient process, Q-learning could op

Adversarial Attacks



Adversarial ML

“Attacking existing models”

Effective|Efficient vs. Secure|Robust

- Proven mainly in the lab
  - Not theoretical, just hard to build
  - Many demonstrations lack practical use
- Two basic approaches:
  - **White:** Access to the original model, architecture, etc.
  - **Black:** Access only to the outputs for a given input

Previous Works

- **DeepWordBug** - Black-box generation of adversarial text

  https://github.com/QData/deepWordBug

- **Cylance, I kill you!** - Client-side model reversing

  https://skylightcyber.com/2019/07/18/cylance-i-kill-you/

- **Good word attacks on statistical mail filters**

  https://ix.cs.uoregon.edu/~lowd/ceas05lowd.pdf

- **Robustness Toolbox** – Attacks, defenses, etc.

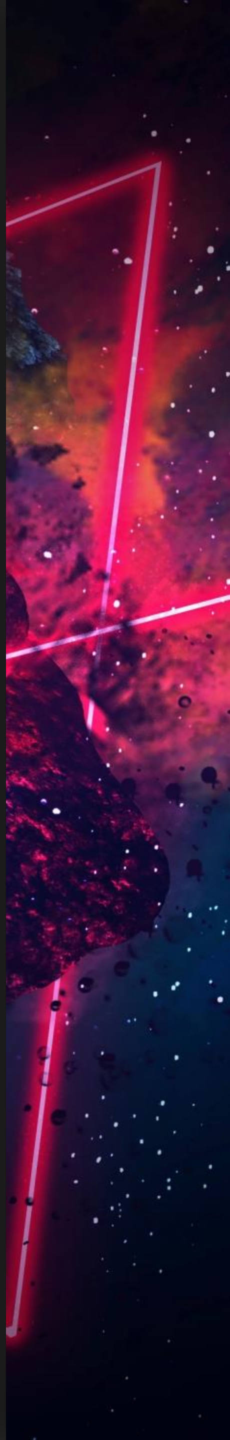
  https://github.com/IBM/adversarial-robustness-toolbox

proofpoint - Case Study

- **E-Mail security company**
  - Spam detection
  - Malware sandboxing
  - URL analysis
  - End user training
- Openly promote their use of ML (MLX, CLX)
- Supporting 230k+ domains – Rapid7 Sonar DNS data

  590 **gov** domains

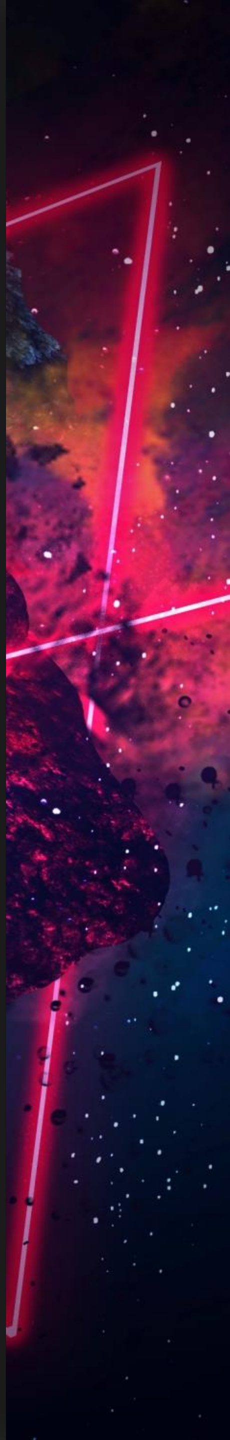
  2300 **edu** domains



proofpoint - Vulnerability

To: <reciever@domain.com>
From: <sender@domain.com>
Subject: Our Meeting
...
X-Proofpoint-Spam-Details: rule=nodigest_notspam policy=nodigest score=0 malwarescore=0 mlxlogscore=999 mlxscore=0 suspectscore=14 spamscore=0 impostorscore=0 adultscore=0 clxscore=593 priorityscore=0 phishscore=0 bulkscore=97 lowpriorityscore=97 classifier=spam adjust=0 reason=mlx scancount=1 engine=9.1.0-12345000 definitions=main-12345

Leaky inputs... tsk tsk



proofpoint - Attack (a)

1. Collect a dataset – Send X emails to steal scores

2. Copy the model - Use their outputs to duplicate

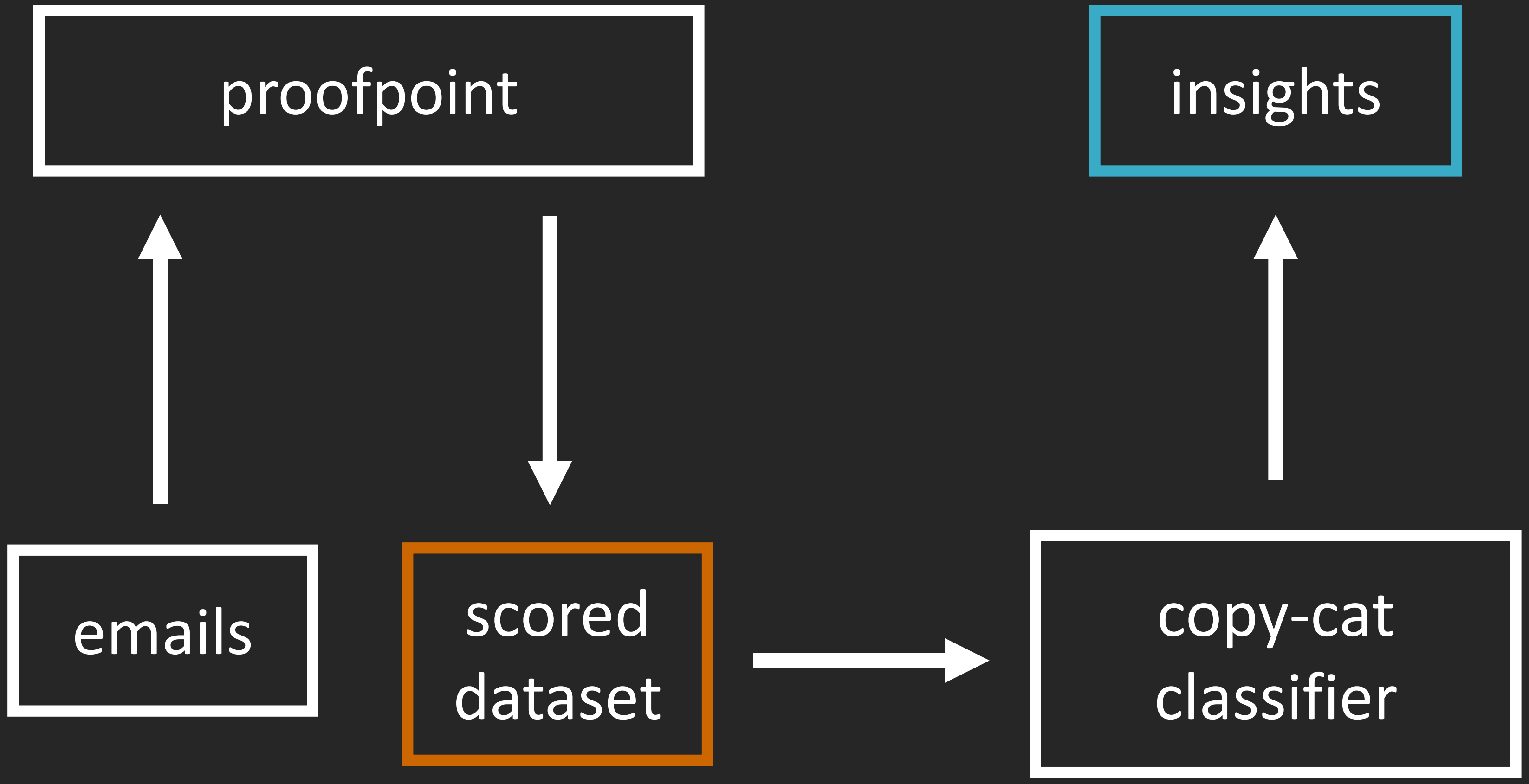
3a. Extract information from the model

- Take N highest/lowest emails, unique the words
- **Toggle inputs to discover the most impactful tokens**
- Invert the model mathematically *
- Randomly alter/add content and re-score

(char swaps, homoglyphs, tense)

* - https://r2rt.com/inverting-a-neural-net.html

proofpoint - Attack (a)



proofpoint - Attack (b)

1. Collect a dataset – Send X emails to steal scores

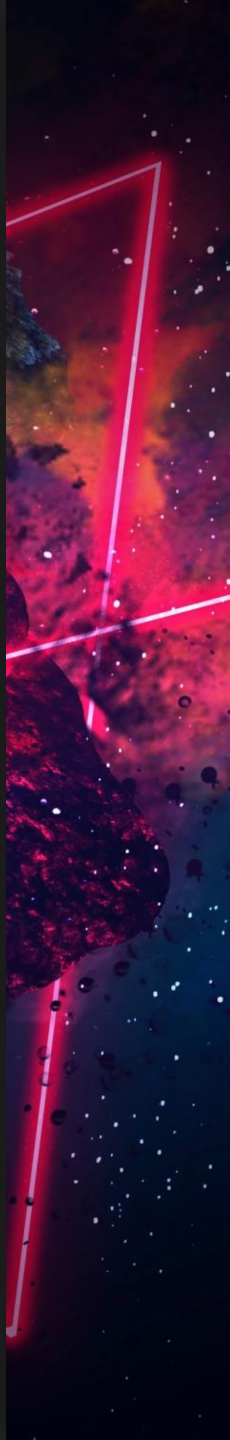
2. Copy the model - Use their outputs to duplicate

3b. Make a generator – Use our copy-cat to train

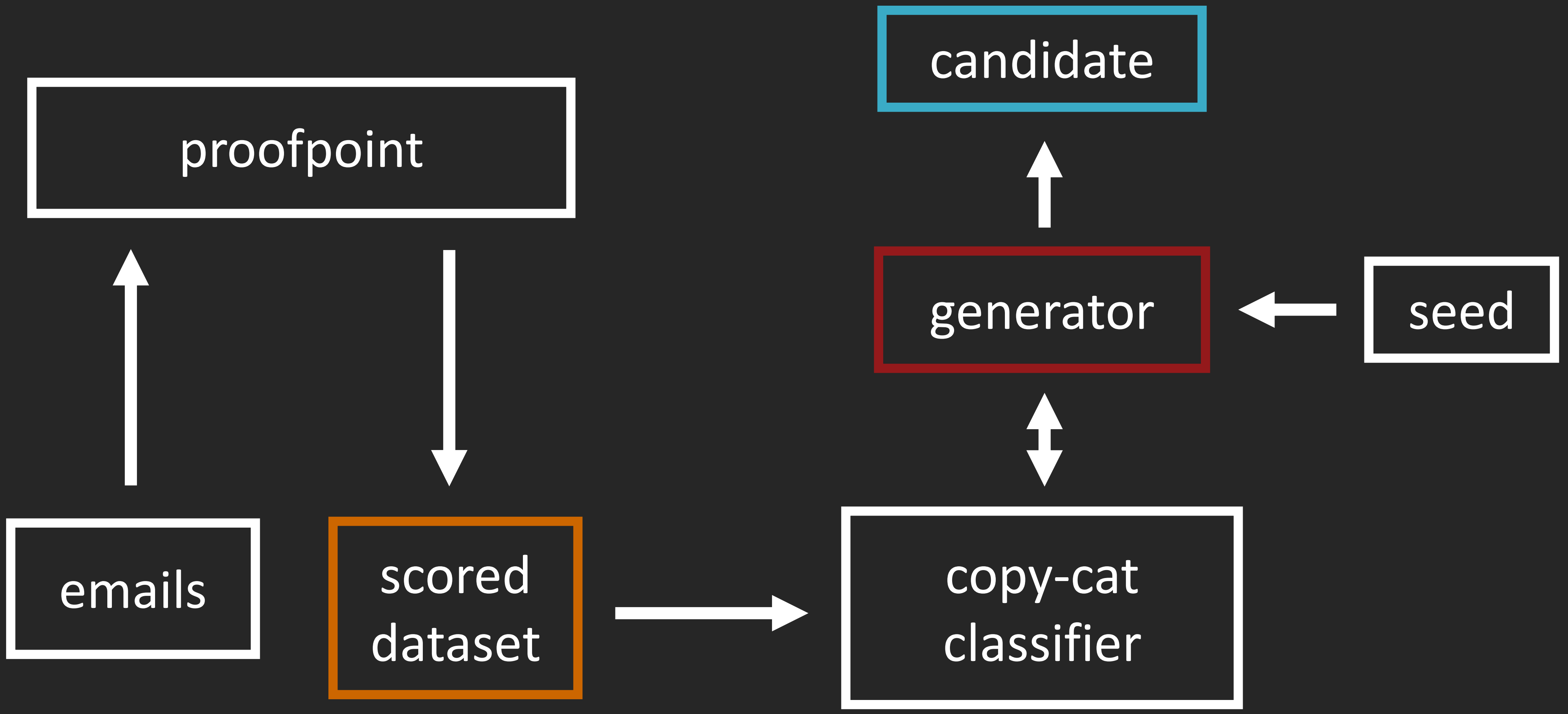
- Let it learn useful insights
- Target maximum score – custom loss function

4. Automate improvements

- “Fix” pre-written candidates
- Generate “good” content from scratch



proofpoint - Attack (b)



proofpoint - Challenges

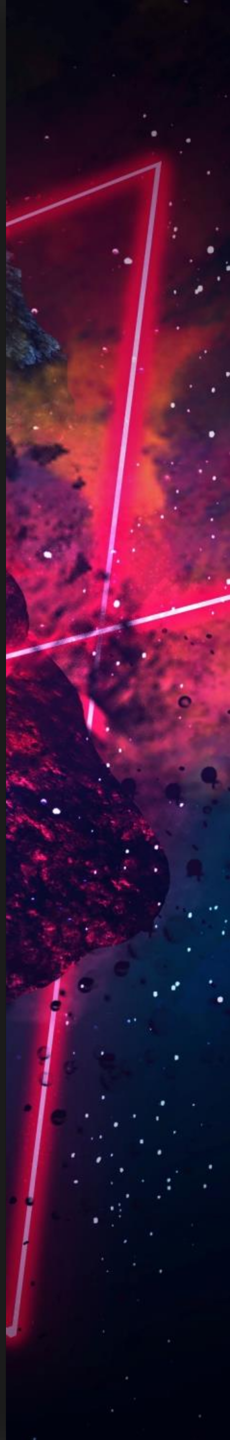
- **Initial email content**
  - Finding a sufficient dataset
  - Links/attachments have large effects on the score
- **Bulk email delivery**
  - Easy if we have a Proofpoint inbox – harder if we don’t
  - Total emails required for a sufficient dataset
  - Extraction process altering the scores
- **Final Outcomes**
  - Generators will likely create gibberish – human intervention
  - Bypassing a model is only one part of the “defenses”

1. Collect a dataset

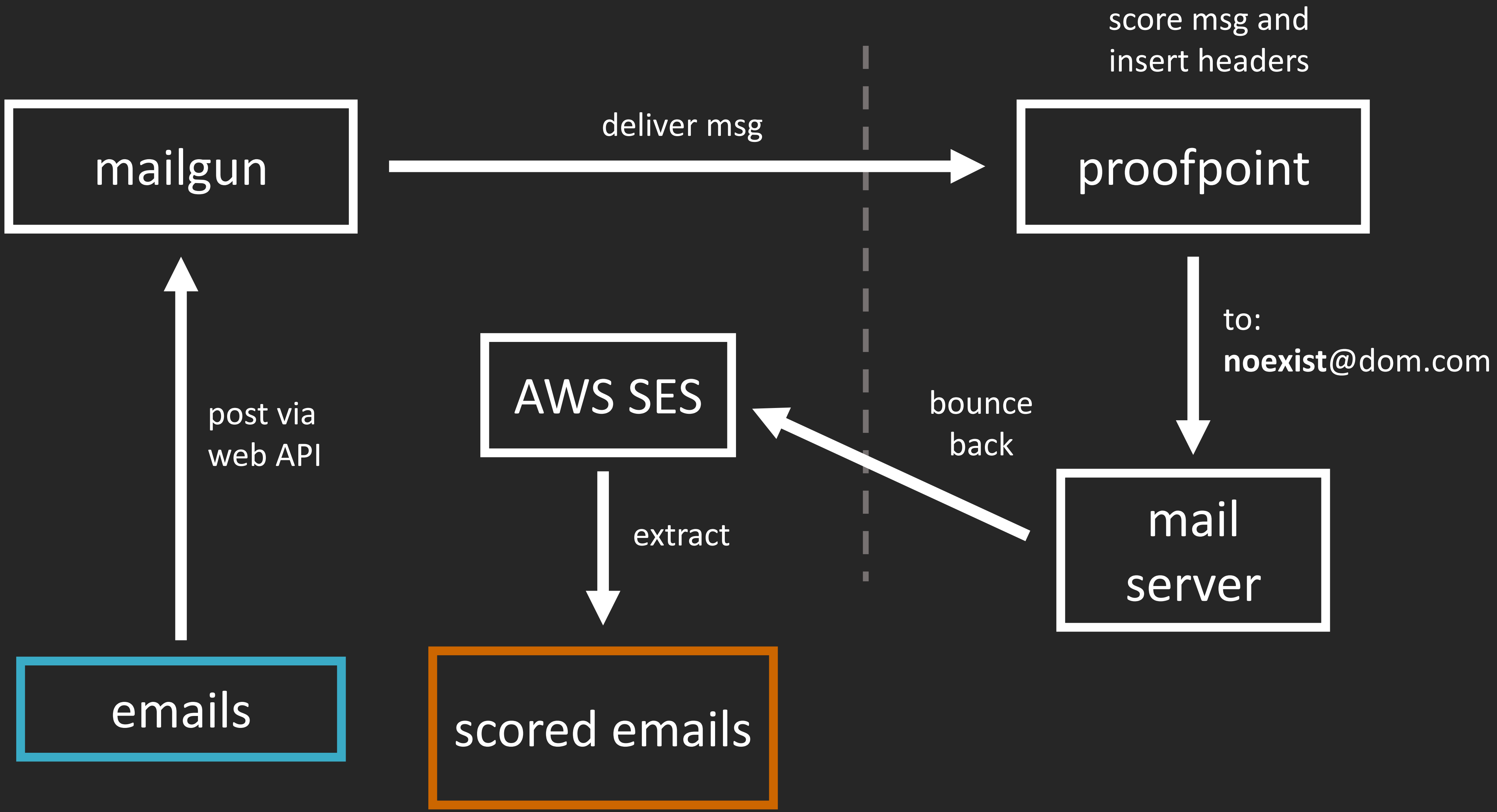
- Needed to gather inputs for scoring (a lot)
  - Enron dataset for text-based candidates
  - ISCX-URL-2016 for link-based candidates
- Use bounce-backs to collect the scores
  - Delivered using Mailgun
  - Received using AWS SES + S3 bucket

We ran multiple collection runs:

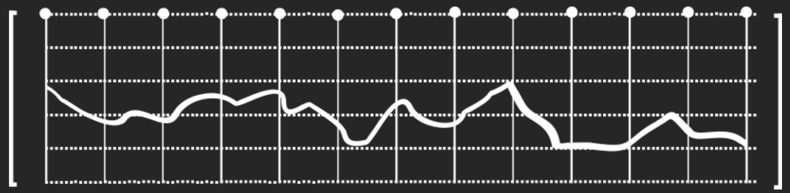
1. 5k pre-processed/scored samples from Enron
2. 13k Links inside a generic template
3. 15k raw subject + bodies from Enron inboxes



1. Collect a dataset



1b. Data Analysis



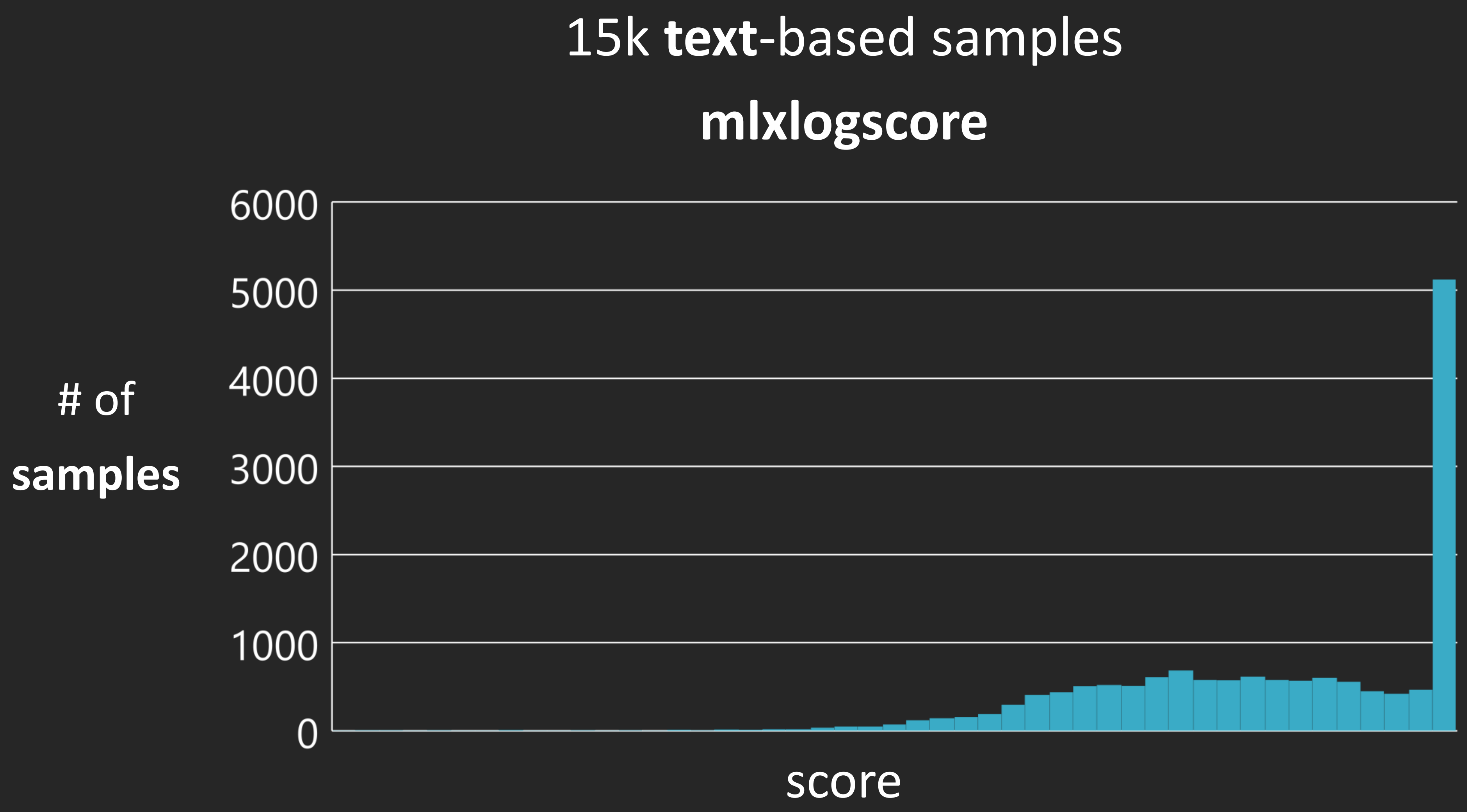
score	bulk	malware	priority	spam	phish	impostor	mlx	mlxlog	low-p	suspect	adult	clx
16	0	0	90	16	0	0	16	73	0	3	0	403
0	0	0	118	0	0	0	0	505	0	8	0	324
0	0	0	118	0	0	0	0	479	0	19	0	303
0	0	0	118	0	0	0	0	489	0	3	0	315
0	0	0	118	0	0	0	0	538	0	3	0	321
0	0	0	90	0	0	0	0	605	0	3	0	437
0	0	0	118	0	0	0	0	455	0	3	0	293
0	0	0	90	0	0	0	0	728	0	3	0	466
0	0	0	118	0	0	0	0	477	0	3	0	299
0	0	0	118	0	0	0	0	483	0	3	0	288
0	0	0	118	0	0	0	0	484	0	3	0	344
0	0	0	90	0	0	0	0	595	0	74	0	432
0	0	0	118	0	0	0	0	329	0	3	0	304

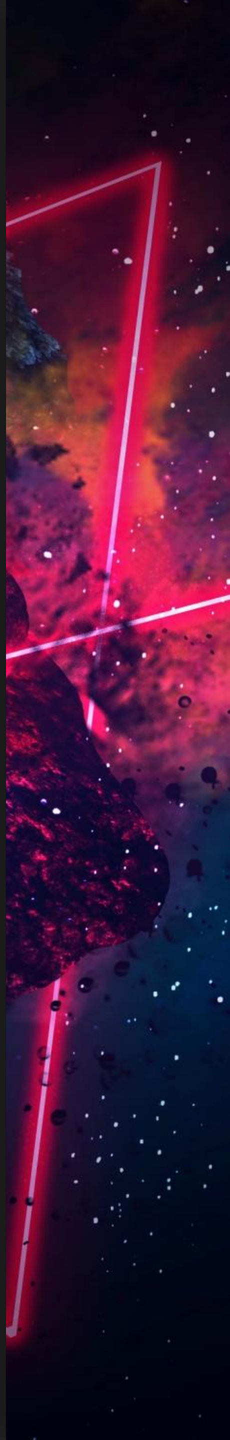
1b. Data Analysis

	score	*bulk*	*priority*	*spam*	*phish*	*mlx*	*mlxlog*	*low-p*	*suspect*	*adult*
bulk	-									
priority	-	-								
spam	1	-	-							
phish	-	-	-	-						
mlx	1	-	-	1	-					
mlxlog	-.2	-	-	-.2	-.1	-.1				
low-p	-	1	-	-	-	-	-			
suspect	-	-	-	-	-	-	-	-		
adult	-	-	-	-	-	-	-	-	-	
clx	-	-	1	-	-	-	-	-	-	-

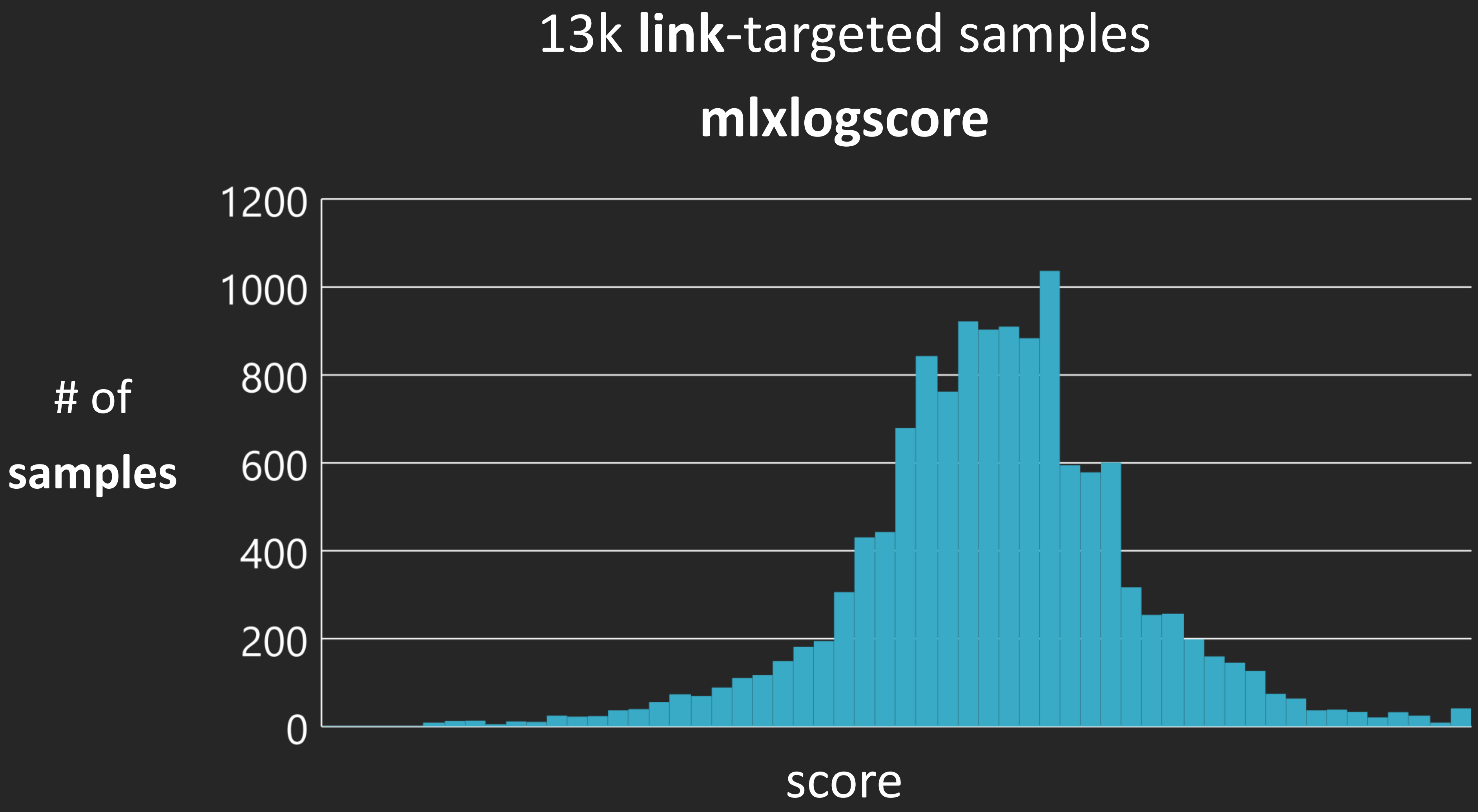
* - values < .1 have been omitted

1b. Data Analysis



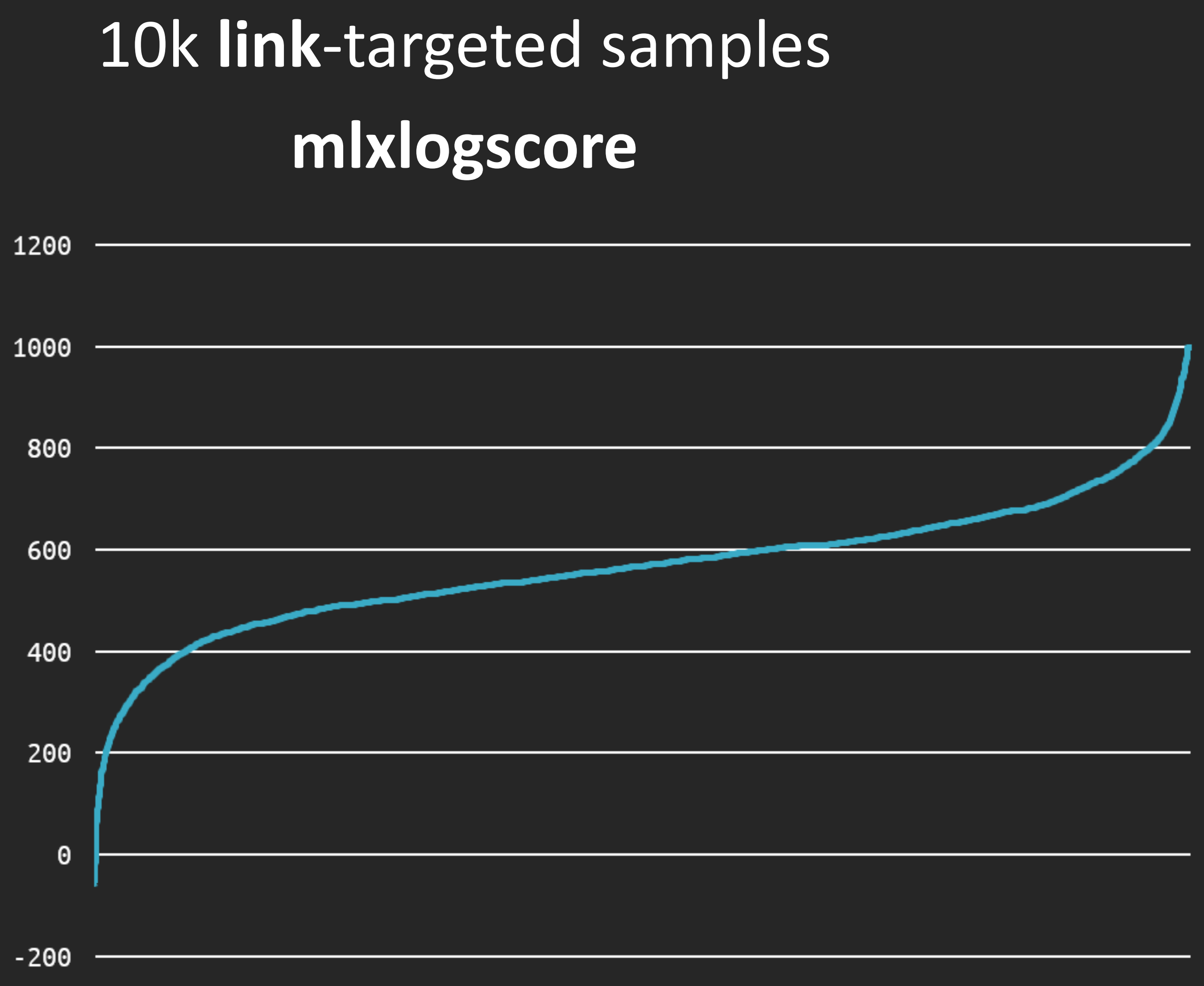


1b. Data Analysis



1b. Data Analysis

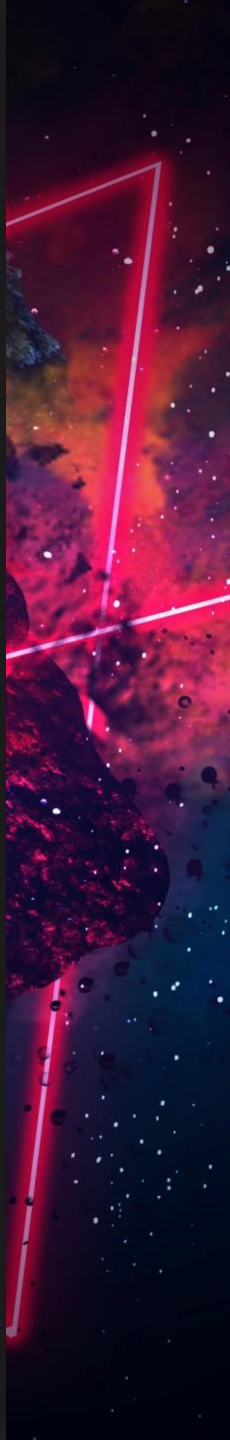
Activation function baby!



2. Copy the Model

- Select a label for training: **mlxlogscore**
  - Good distribution – at least for links
  - Previously scaled / activated
  - *Generally* between 1 and 999
  - Larger = "safer“

- Select some likely model emulators
  - **Neural Network** + Bag of Words (BOW) *
  - **LSTM** + Sequenced Text



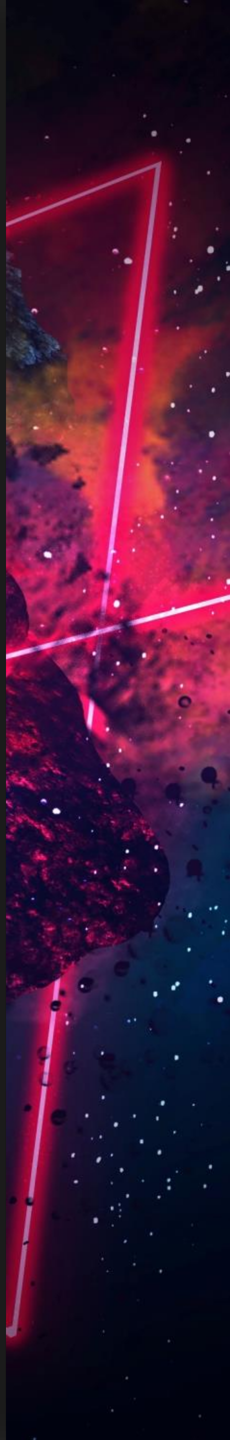
* - Also referred to as “one-hot” encoding

2. Copy the Model - Results

	Neural Network + BOW	LSTM + Sequences
Text targeted samples	69	91
Link targeted samples	**42**	96

(we didn’t plan this, we swear)

* showing scaled mean absolute error (mean **point** error)



3. Extract Information

- **Make text alterations and rescore**
  - Take a phishing email: "Click here for cats"

    Clikc here for cats (Swap)

    Click hare fur cats (Substitute)

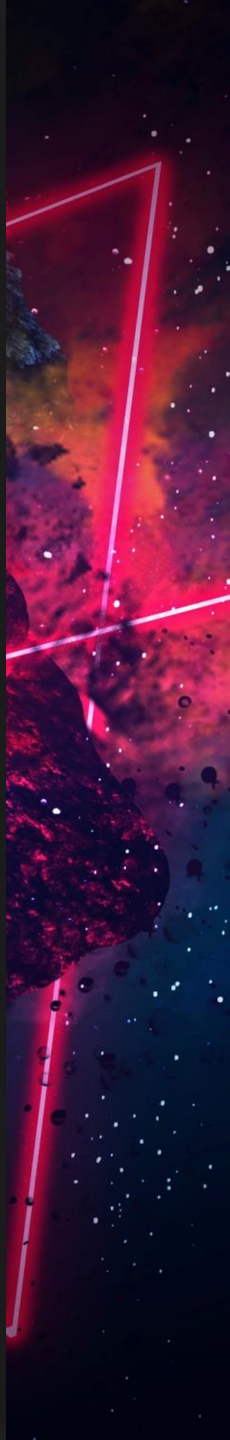
    Click her for cats (Delete)

    Click here for carts (Insert)
  - Final outputs need to make sense
- **Toggle input tokens and rescore**

  [1,1,1,1] - Click here for cats - 500

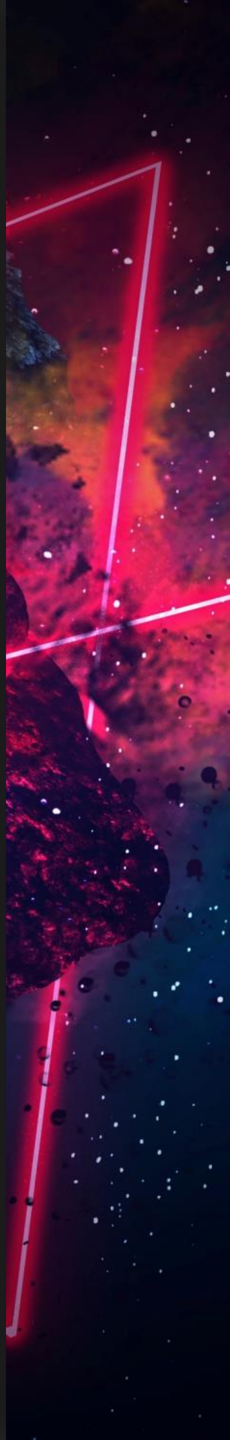
  [0,1,1,1] - here for cats - 490

  [1,0,1,1] - Click for cats - **300**
- **Score every possible combination of tokens** - fuzzing



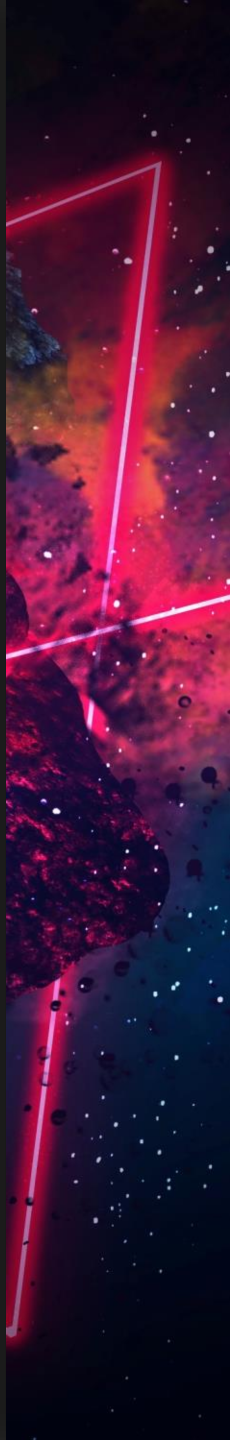
3. Extract Information

```
for sample in test_set:
    base = make_prediction(sample)

    for token in sample:
        altered = sample.toggle(token)
        test = make_prediction(altered)

        # Record a rolling score movement
        insights[word] += (base – test)
```



3. Extract Information - **Texts**

good	**meh**	**bad**
calculation	lisa	software
asset	digest	**99**
appreciated	piano	unsub
finalized	stems	**bridgeline**
tyson	architectual	absolutely
difficult	living	quantities
dial	smells	**hydro**
default	storms	proposal
lawyers	alcoholic	deposit
bids	broccoli	holden



3. Extract Information - **Links**

good	meh	bad
movies	cpanel1	cool
category	certificate	**citi**
ecnavi	area2	hc
xml	delores	**wp**
payment	verify2	license
docs	struggles	**includes**
shop	chinas	styles
dest	second	logon
kitchen	webserver	plugins
webapps	uniq	spreadsheet

Confirming our insights - **Texts**

Top 10 **highest** scoring words:

999

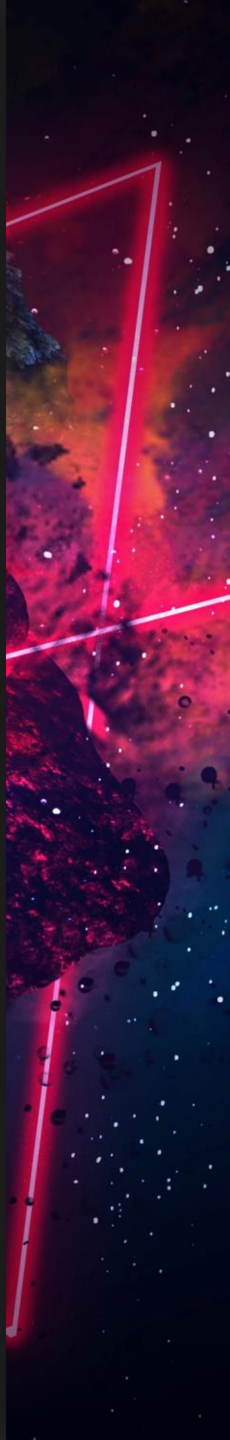
Random 10 words from the middle:

640

Top 10 **lowest** scoring words:

…

mx0a-000a1001.pphosted.com gave this error:
This message looks too much like SPAM to accept.



Confirming our insights - **Links**

https://neverexistdomain.com/**wp-includes**/file

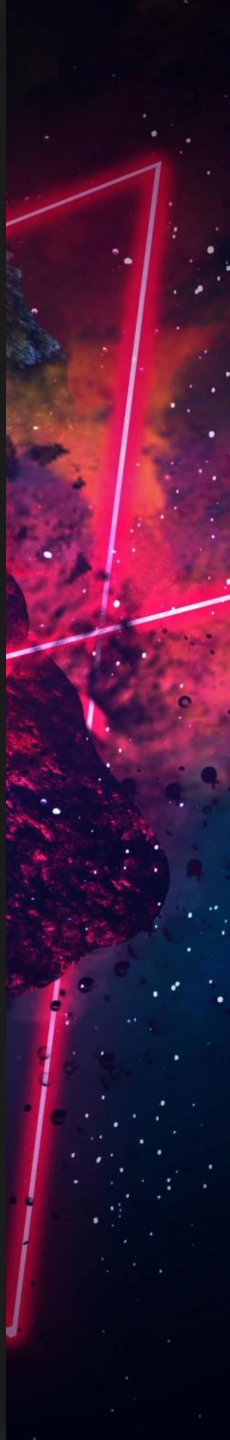
Predict: 378 | Real: **300**

https://neverexistdomain.com/**up-uncludes**/file

Predict: 600 | Real: **559**

https://.../ecnavi/category.xml?movies=payment

Predict: 999 | Real: **999**



Disclosure | Remediation

- **Models are interesting beasts**
  - Represent learned vectors – not always apparent
  - Difficult to retrain / rebuild
  - Black box with “magic” inside
- **What warrants responsible disclosure?**
  - What % is considered a viable bypass?
- **How does remediation occur?**
  - Can’t just add a signature
  - General models might work even without leaky outputs

Final Thoughts

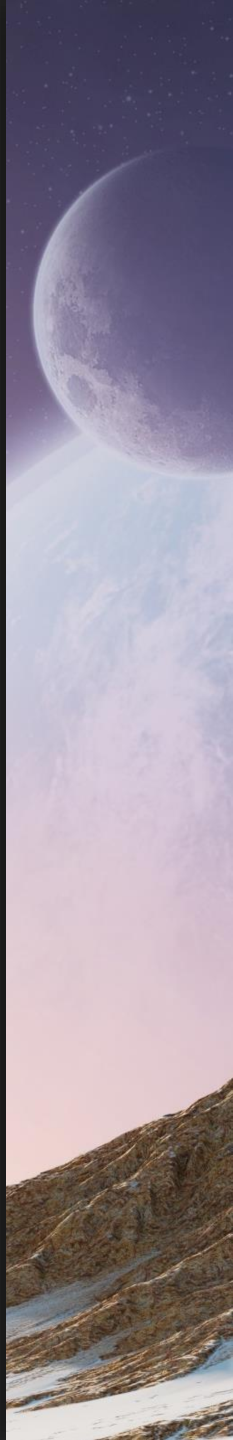
Real World Talk

- **Application Whitelisting**
  - Was cool until people realized there were bypasses
  - Will be a vendor pitch while it gets sales
- **EndGame ML Competition**
  - Simple bypasses for static analysis (sRDI, emojis)
  - Data scientists solving defensive problems
  - Static analysis is only a small part of the battle

“If you don’t understand X before ML, you won’t understand it after” - Will

Not All Bad ...

- **The next generation of malware**
  - Intelligent agents
  - Genetic programming client-side (JIT, variants on the fly)
  - Distributed API calls and hooks
  - Rootkits – layered defenses (clothing)
- **Securing and hardening models**
  - Helping vendors improve their products
  - Ensuring ML isn’t the next big security gap



Fun Projects

- What other defenses leak outputs?
  - Windows defender AMSI sampling
  - URL categorization
- Can we evade other mechanisms?
  - E-Mail attachments in transit
  - HTML content during site inspection
- What other offensive tasks can be offloaded?
  - File & directory enumeration
  - External reconnaissance
  - Chat/E-mail bot for phishing
- Add data extraction to Seatbelt for Neo4j

Final Thoughts

- Lots of fun work to be done, come play!
- Machine learning is here to stay, don’t sleep on it
  - It’s all a joke until you get caught
  - “Model bypasses” will become a part of offensive kits
  - ML understanding could quickly become a job requirement

“Might be nothing – could be everything.

Likely somewhere in the middle.

Time will tell.” - Will

Greetz

@culteredphish – Colleague & all around good guy

@tyler_robinson – Ex-Colleague & long time friend

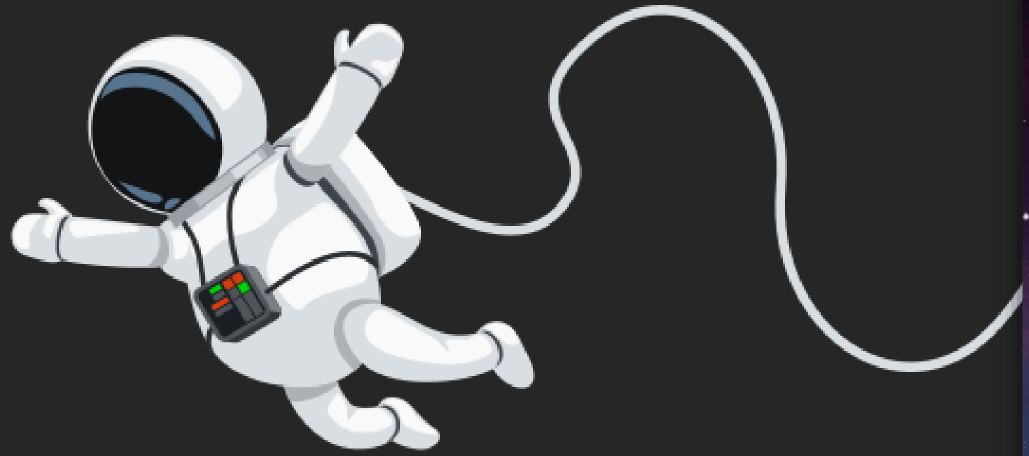
@rharang – Answered some helpful questions

@silentbreaksec – Company supporting this research

- Nancy Fulda of BYU
- Will’s Mother-in-law for babysitting

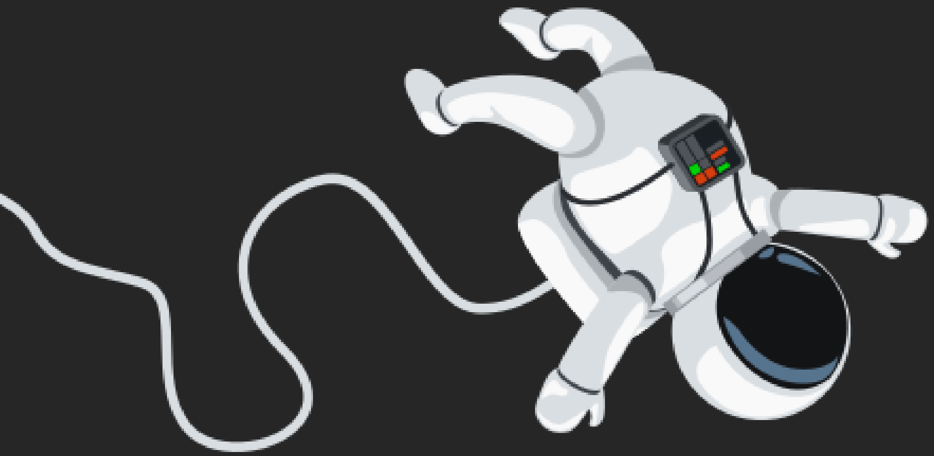
All of you for attending the talk

Find Us After



Nick Landers

@monoxgas



Will Pearce

@moo_hax

MooKitty

SILENTBREAK SECURITY

Soon: http://github.com/MooKitty/FourtyTwo

Resources

- **“Make your own Neural Network”** – Tariq Rashid
- **“3 Blue 1 Brown”** YouTube channel
- **“Jabrils”** YouTube channel

https://www.kaggle.com/

https://silentbreaksecurity.com/machine-learning-for-red-teams-part-1/

https://github.com/MoooKitty/RedML

So long and thanks for all the phish!