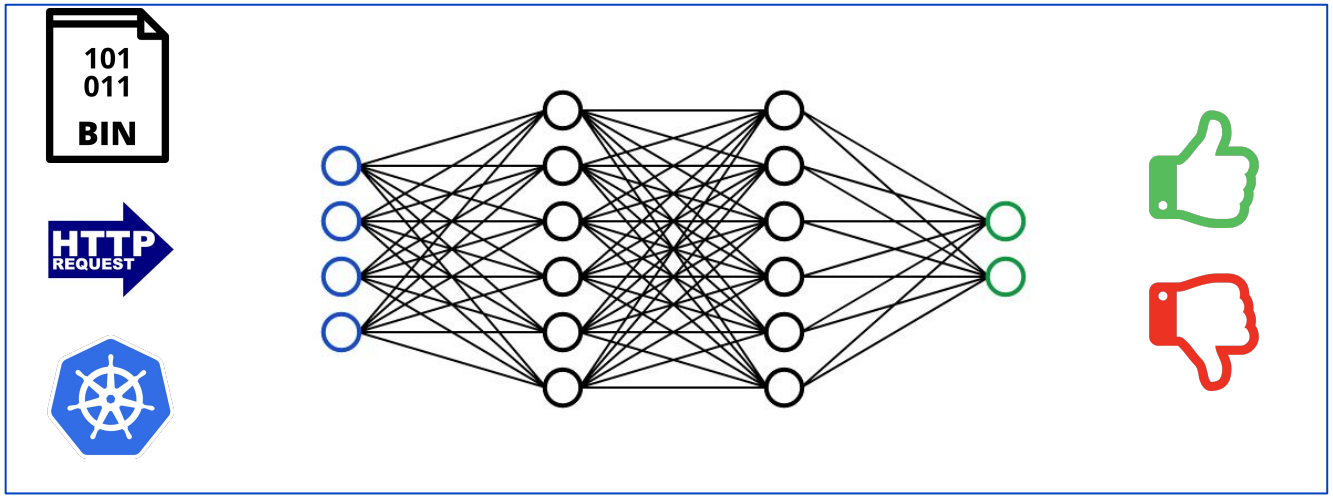
Demystify AI Security Products With a Universal Pluggable XAI Translator

Tongbo Luo, Kailiang Ying, Xinyu Xing, Xuguang (Luke) Liu

Motivation



Scenarios

Vendor

Customer

Attacker

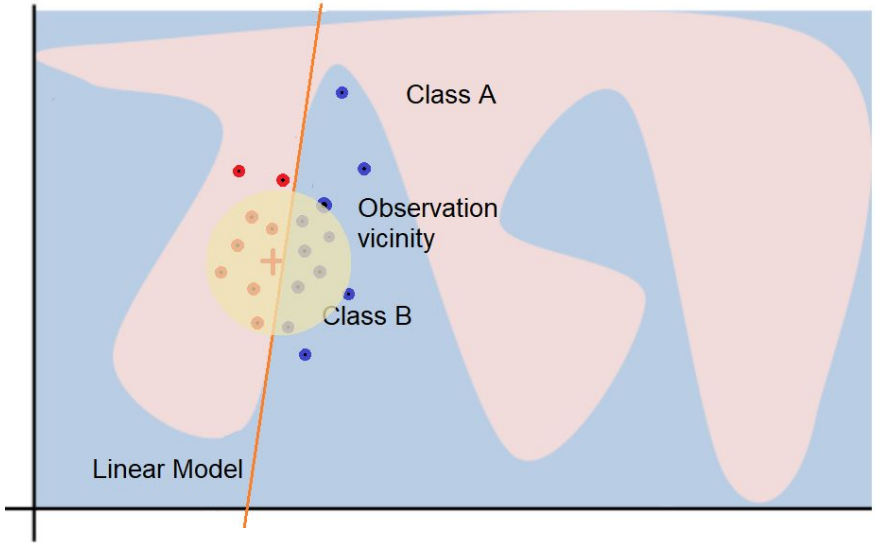
Key Takeaways

- Share lesson learned when we use XAI to evaluate security products
- Identify potential XAI research direction to fill in business need

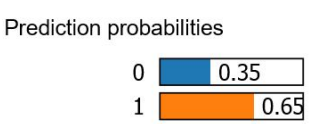
State-of-art XAI Tools

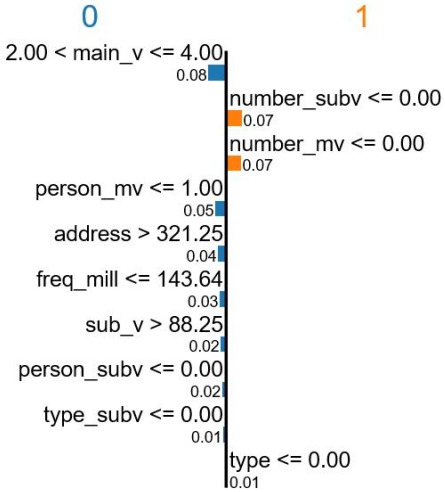
- LIME
- SHAP (KernalSHAP)
- Anchor

LIME --- Local Interpretable Model-Agnostic Explanations



$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g)$$





"Why Should I Trust You?": Explaining the Predictions of Any Classifier

LIME

Advantages

- Works for all types of data (images, tabular, text)

- Model was trained with non-interpretable features

Disadvantages

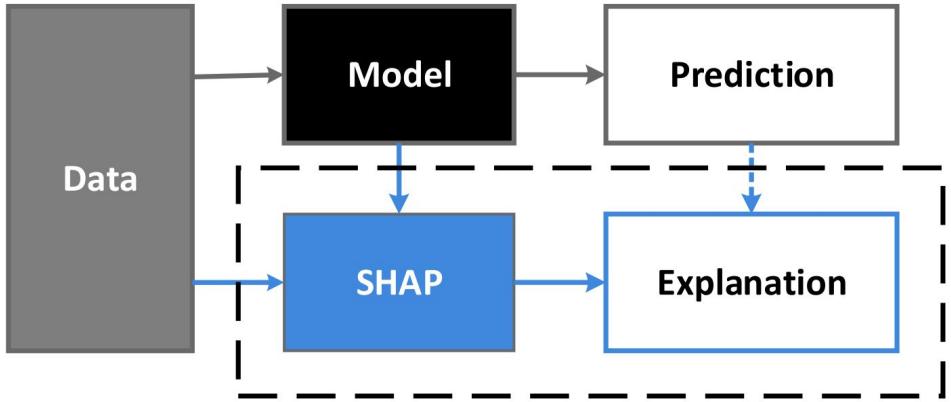
- Instability of the Explanations

- Sampling process

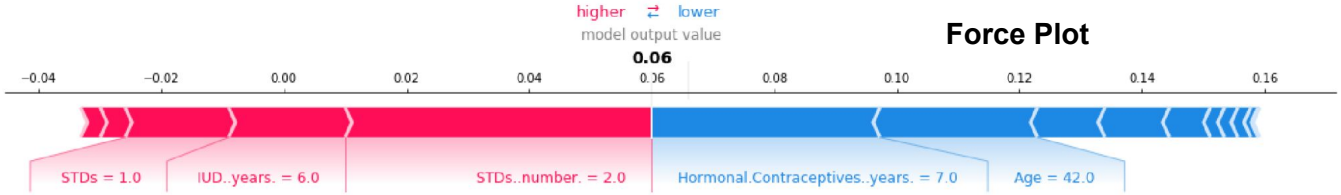
SHAP

SHAP (SHapely Additive exPlanations)

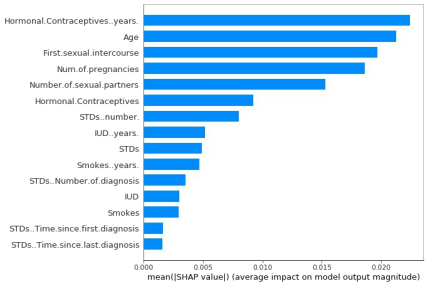
- Shapely Values (Game Theory)
- Visualization



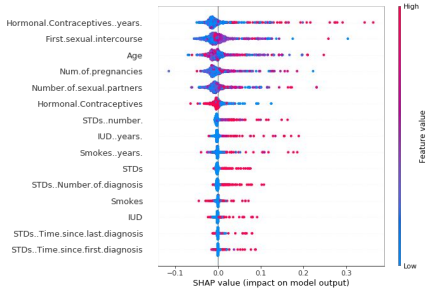
Build-in Visualization



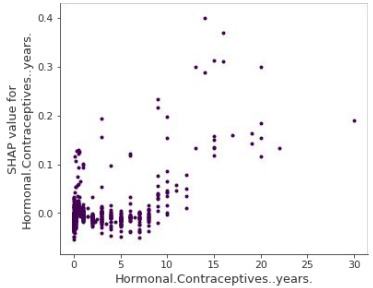
Force Plot



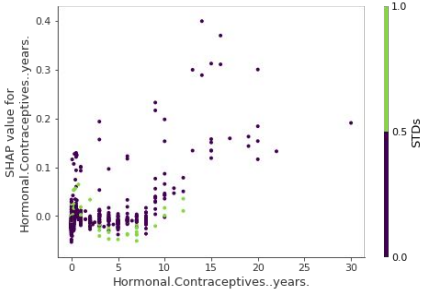
Feature Importance



Summary Plot

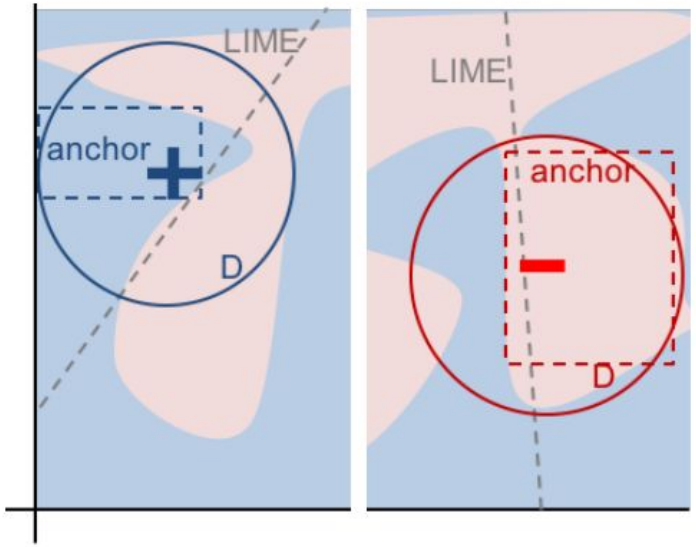


Dependence Plot

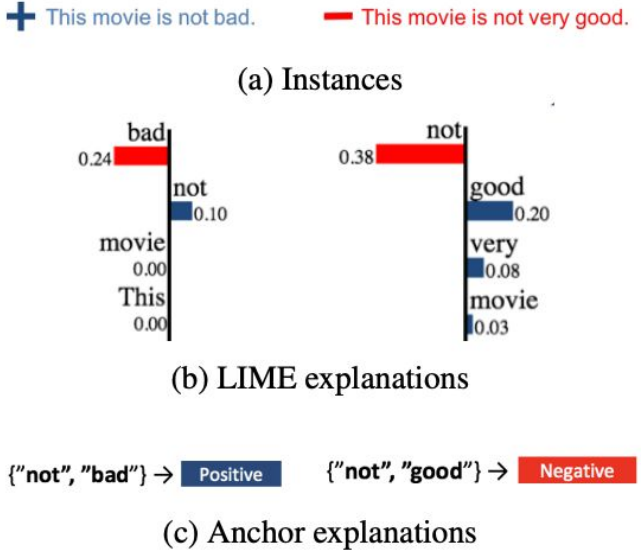


Interaction Values

Anchor



ref: tinyurl.com/2nd7w8e7



Anchors: High-Precision Model-Agnostic Explanations (AAAI'18)

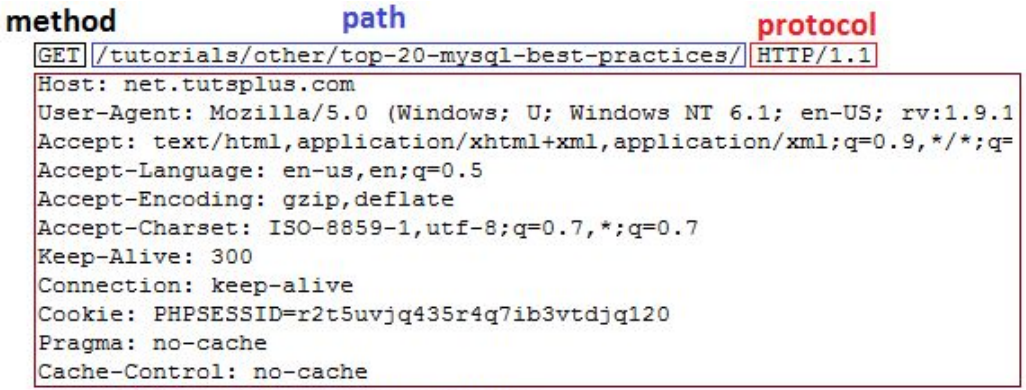
When XAI meets ML-based Security Product

- **Malicious HTTP header Detection Tool (DNN with text-type input)**
- **IDS (DNN with numeric features)**
- **Malicious Binary Detection (RNN-based Detection Model)**
- **Detection Malicious Cloud Activity**
- **System-call Detection (Concept-based Explanation)**

Detecting Malicious HTTP Requests

Common Attacks

Command Injection Attack
SQL Injection Attack
XSS



Assumption Gaps

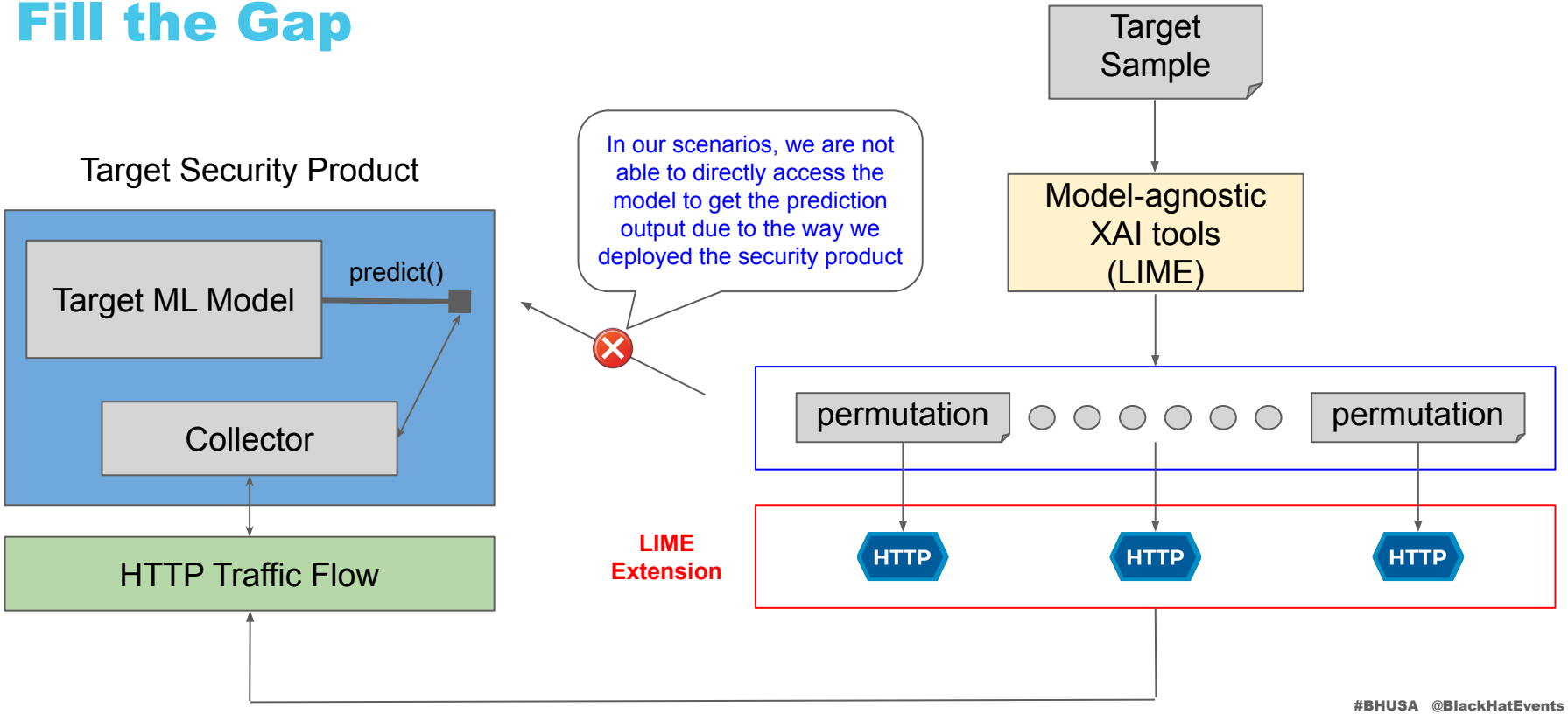
- **Lack of model knowledge**
  - Actual model's detail is confidential

Model-agnostic tools
(treat model as blackbox)

- **No direct access to model**
  - Trigger actual activity in the target system

Customize the XAI tools

Fill the Gap


Target Sample
Model-agnostic XAI tools (LIME)
In our scenarios, we are not able to directly access the model to get the prediction output due to the way we deployed the security product
Target Security Product
Target ML Model
predict()
Collector
HTTP Traffic Flow
permutation
permutation
HTTP
HTTP
HTTP
LIME Extension

Avoid Sampling Invalid Data

Header to be explained: { “method” : “get”, “query”: “Accessories; Drop”, “path” : “/search”, “statusCode”: “404”, “requestPayload”: “null” }

Invalid sampling with LIME_TEXT

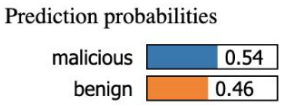
{ “” : “get”, “query”: “Accessories; Drop”, “path” : “/search”, “statusCode”: “404”, “requestPayload”: “null” }

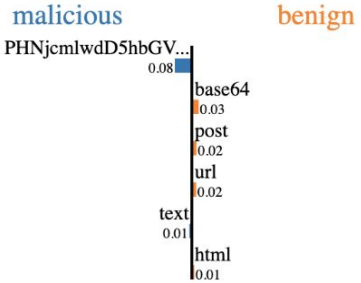
{ “method” : “”, “query”: “Accessories; Drop”, “path” : “/search”, “statusCode”: “404”, “requestPayload”: “null” }

Invalid sampling with LIME_TABULAR

{ “method” : “get”, “query”: “Accessories; Drop”, “path” : “/search”, “statusCode”: “404”, “requestPayload”: { "creditCard":"<iframe />”} }

Detecting Malicious Requests

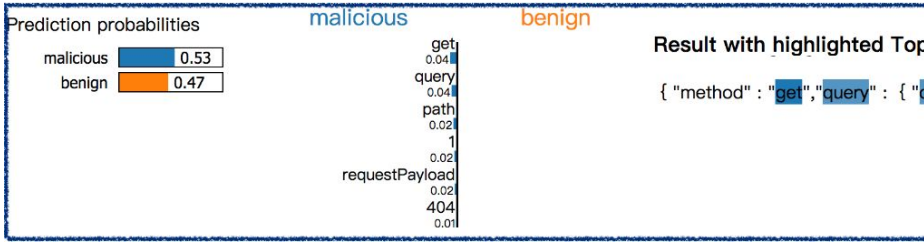


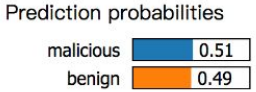


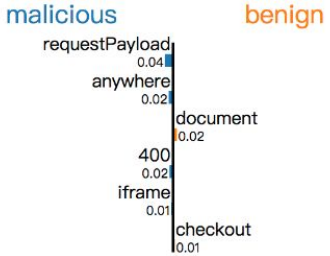
Text with highlighted words

{ "method" : "post","query" : { } ,"path" : "/checkout","requestPayload" : { "creditCard" : "|META HTTP-EQUIV="refresh"CONTENT="0;url=data : text/html;base64,PHNjcmlwdD5hbGVydCgndGVzdDMnKTwvc2NyaXB0Pg"| }

<script>alert('test3')</script>



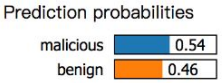


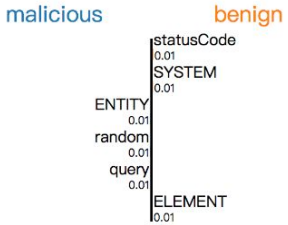


Result with highlighted Top Indictor

{ "method" : "post", "query" : { } , "path" : "/checkout", "statusCode" : 400, "requestPayload" : { "creditCard" : "<script> document.write('iframe src="http://anywhere.com"'); </script>" } }

XSS with remote iframe src





Result with highlighted Top Indictor

{ "method" : "post", "query" : { } , "path" : "/checkout", "statusCode" : 400, "requestPayload" : { "creditCard" : "<!DOCTYPE foo [<!"ELEMENT root ANY ><!ENTITY unixfile SYSTEM \"file:///dev/random\">]> <root>&unixfile;</root>" } }

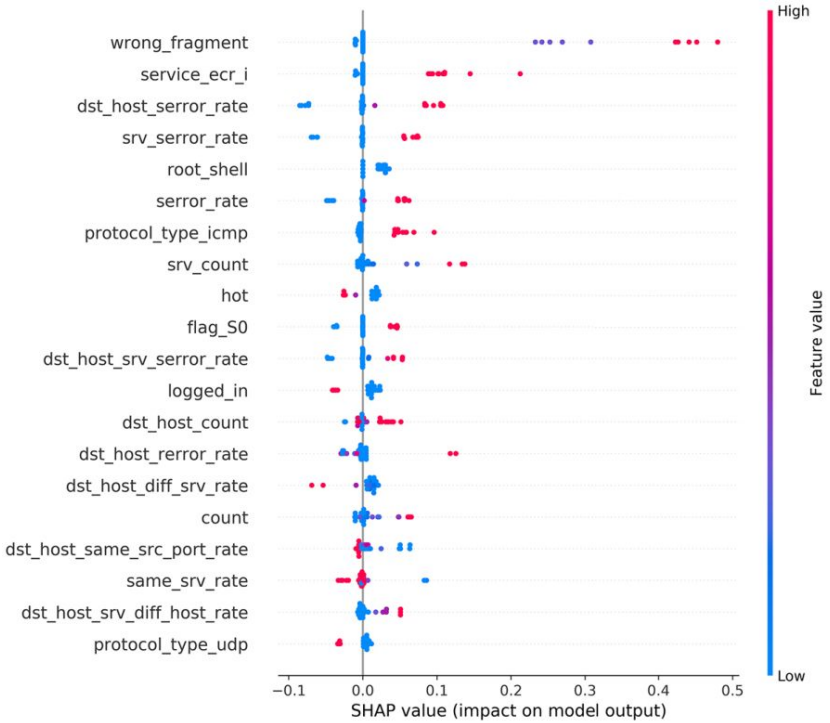
XXE DoS under Unix Server

ML-based IDS (Intrusion Detection System)

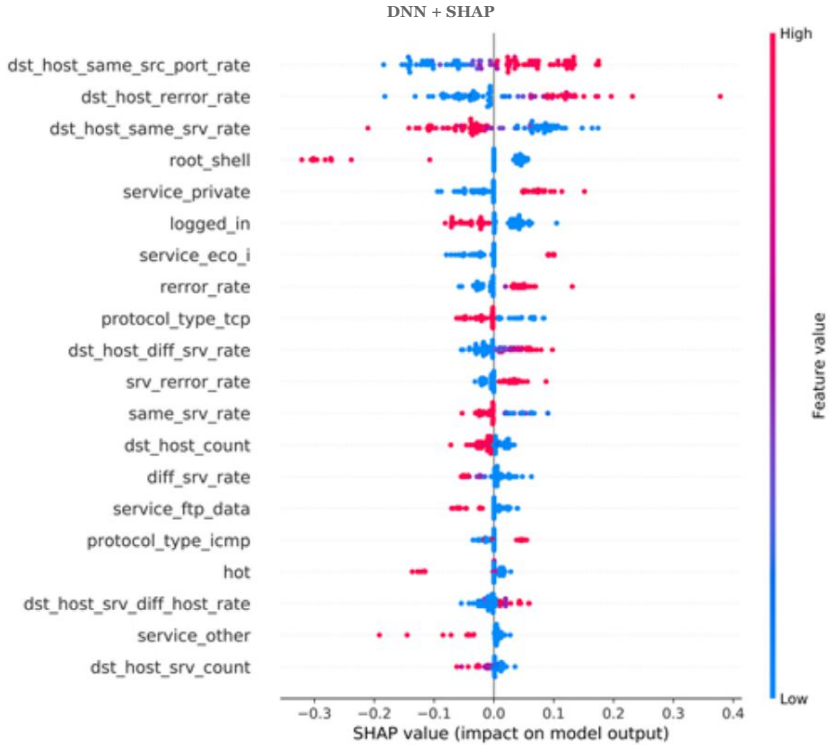
F#	Feature name	F#	Feature name	F#	Feature name
F1	Duration	F15	Su attempted	F29	Same srv rate
F2	Protocol type	F16	Num root	F30	Diff srv rate
F3	Service	F17	Num file creations	F31	Srv diff host rate
F4	Flag	F18	Num shells	F32	Dst host count
F5	Source bytes	F19	Num access files	F33	Dst host srv count
F6	Destination bytes	F20	Num outbound cmds	F34	Dst host same srv rate
F7	Land	F21	Is host login	F35	Dst host diff srv rate
F8	Wrong fragment	F22	Is guest login	F36	Dst host same src port rate
F9	Urgent	F23	Count	F37	Dst host srv diff host rate
F10	Hot	F24	Srv count	F38	Dst host serror rate
F11	Number failed logins	F25	Serror rate	F39	Dst host srv serror rate
F12	Logged in	F26	Srv serror rate	F40	Dst host rerror rate
F13	Num compromised	F27	Rerror rate	F41	Dst host srv rerror rate
F14	Root shell	F28	Srv rerror rate	F42	Class label

Common Features Used by ML-based IDS

Explain IDS Detection Result



(a) DoS



(b) Probe

Malware Detection (Binary Reverse-Engineering)

Hex Sequence

90	90	90	90	83	ec	4c

Decimal Sequence

144	144	144	144	131	236	76

Classifier

Classifier Output

0.01	0.01	0.01	0.01	0.99	0.01	0.01

LEMNA

90
90
90
90
83
ec
4c

Detect Cloud Malicious Activity

Network
- Activity to/from Known bad IPs
- Usual changes to traffic pattern
- Unsal outbound port usage

DNS
- Queries to known-bad domains

Host-based
- OS, Application, Security/Audit logs
- Endpoint security event

Network-device based
- FW/IDS/IPS “drop-in” solution logs/alerts

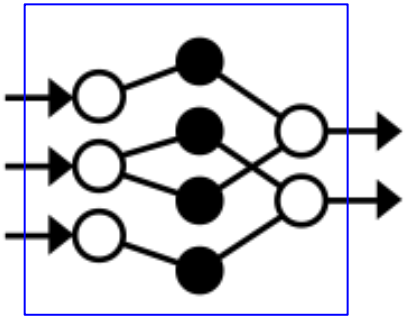
Cloud provider API Activity
- Multiple failed logins
- Simultaneous API access from different regions
- Attempted activity from terminated accounts/credentials/keys
- Uncommon service/API usage
- Credential/permission enumeration
- Changes to user accounts/logging/detection configurations
- Sensitive changes to user permission
- Internal IAM credentials used from external sources

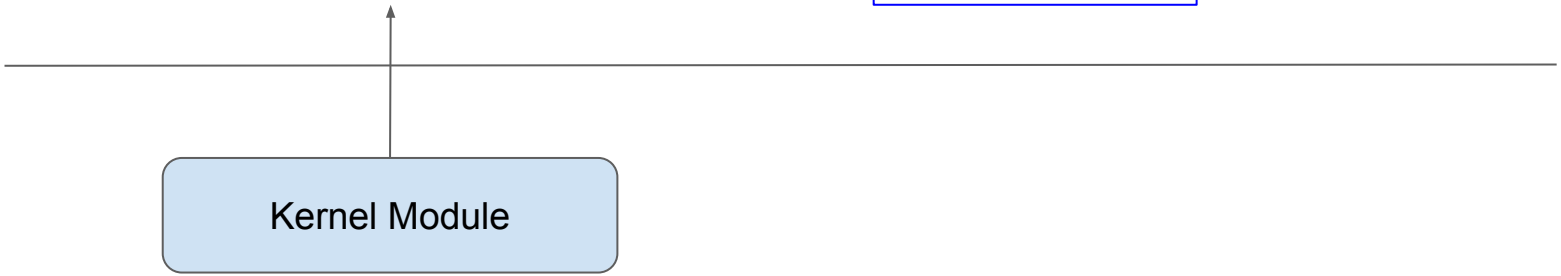
Concept-based Explanation

Instead of explaining individual sample, we think concept-level explanation is better when evaluating security products.

Malware Detection Model using System-call

	write	execve	accept	ioctl	...
proc0	100	20	0	2	...
proc1	50	10	90	3	...
...	...	...	...	...	...

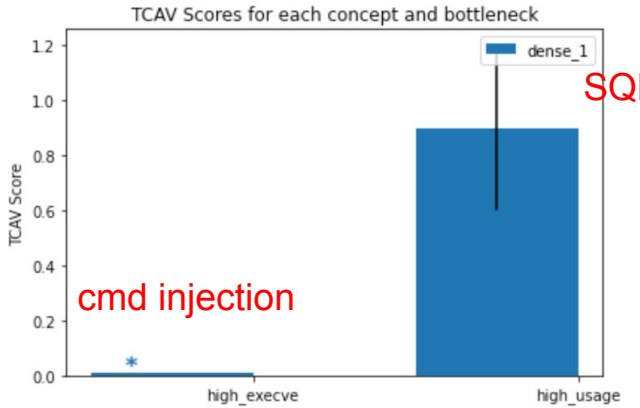




Global explain on system call detection

1. SQL injection: high file I/O, high memory usage, high network throughput
2. cmd injection: high execve usage

```
Class = Malicious.
  Concept = high_execve
    Bottleneck = dense_1. TCAV Score = 0.70 (+- 0.46), random was 0.54 (+- 0.49). p-val = 0.344 (not significant)
  Concept = high_usage
    Bottleneck = dense_1. TCAV Score = 0.90 (+- 0.30), random was 0.54 (+- 0.49). p-val = 0.030 (significant)
{'dense_1': {'bn_vals': [0.01, 0.8985], 'bn_stds': [0, 0.29953338712070143], 'significant': [False, True]}}
```



SQL injection

cmd injection

Gap: White-box model

Attacker's Perspective

1. Use XAI to Craft Adversarial Example
2. Use XAI to Leak Information from Security Product

Crafting Adversarial Example

- Model set up
  - white-box
  - black-box

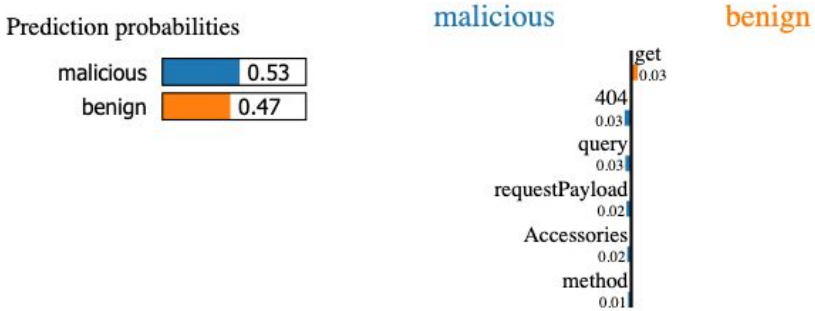
Estimate the gradient using finite difference

$$\mathrm{FD}_{\mathbf{x}}(g(\mathbf{x}), \delta) = \begin{bmatrix} \frac{g(\mathbf{x}+\delta\mathbf{e}_1)-g(\mathbf{x}-\delta\mathbf{e}_1)}{2\delta} \\ \vdots \\ \frac{g(\mathbf{x}+\delta\mathbf{e}_d)-g(\mathbf{x}-\delta\mathbf{e}_d)}{2\delta} \end{bmatrix}$$

- Adversarial Example must be “valid”
  - Satisfy the structure requirement
  - Keep the malicious behaviour

Generate Adversarial Example via XAI

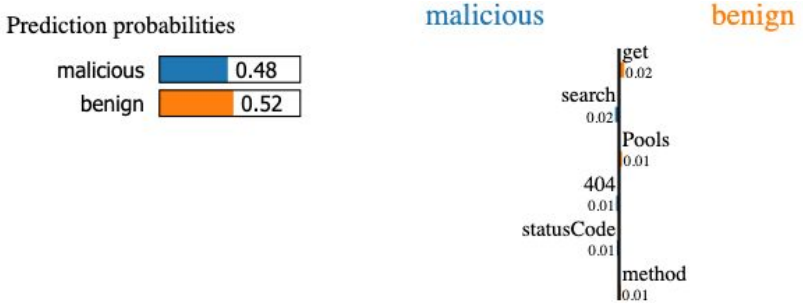
Original example



Text with highlighted words

{ "method" : "get","query" : { "query" : "Swimming Pools|Accessories;DROP" } ,"path" : "/search","statusCode" : 404,"requestPayload" : null }

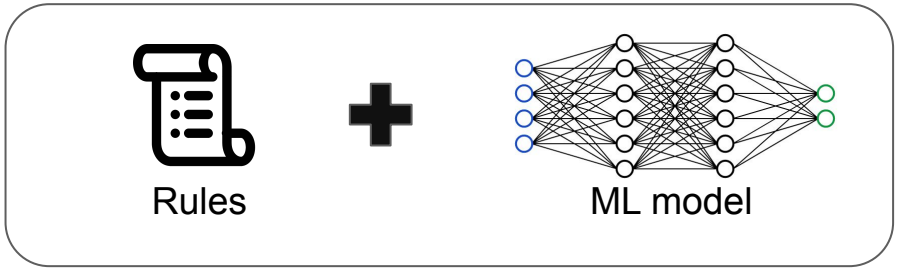
Adversarial example

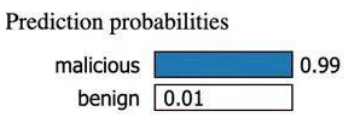


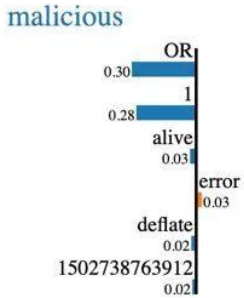
Text with highlighted words

{ "method" : "get","query" : { "query" : "Swimming Pools|Pools;DROP" } ,"path" : "/search","statusCode" : 404,"requestPayload" : null }

Leak Information from Hybrid Security Products





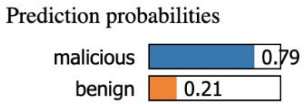


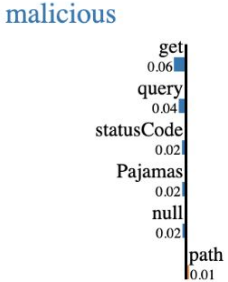
Text with highlighted words

{ "timestamp" : 1502738763912 method : "get" query : { "query" : "OR 1=1; #" } path : "/search" statusCode : 404 source : { "remoteAddress" : "233.150.201.166" referer : "http : //localhost : 8002/enter" } route : "/search" headers : { "host" : "localhost : 8002" connection : "keep-alive" accept : "*/*" cache-control : "no-cache" x-requested-with : "XMLHttpRequest" referer : "http : //localhost : 8002/enter" accept-encoding : "gzip deflate br" accept-language : "en-US en;q=0.8 es;q=0.6" } requestPayload : null responsePayload : { "statusCode" : 404 error : "Not Found" message : "Not Found" } }


Q&A

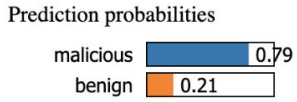
Flaky local explain

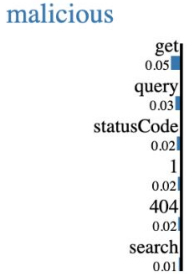




Text with highlighted words

{ "method" : "get","query" : { "query" : "Pajamas | RobesOR 1=1; #" },"path" : "/search", "statusCode" : 404, "requestPayload" : null }





Text with highlighted words

{ "method" : "get","query" : { "query" : "Pajamas | RobesOR 1=1; #" },"path" : "/search", "statusCode" : 404, "requestPayload" : null }