Demystify AI Security Products with a Universal Pluggable XAI Translator

Kailiang Ying*
Syracuse University

Tongbo Luo
Robinhood Markets, Inc

Xuguang Liu
Palo Alto Networks

Xinyu Xing
JD.com

ABSTRACT

In the past years, we witnessed a dramatic rise in the platforms and apps based on machine learning and artificial intelligence. Inevitably, nearly every security product claims to be powered by deep learning technology and achieve an incredible detection rate. A considerable number of security products leverage deep learning to achieve an incredible detection rate. If these machine learning detection models perform well, we prefer to blindly trust them. In some safety-critical cases, such as online banking transactions or high-quality services, we may want to know why the decision was made and possibly pay for the interpretability. Knowing the "why" can help SOC teams learn more about the threat and determine the consequences of it. However, the lack of transparency has limited their decisions to be understood by the security operation team.

Confused by the various fancy terms advertised by security companies, the dilemma faced by customers is how to determine the quality of these products and how to choose the suitable one. Previous studies have proposed various ways to evaluate many kinds of ML-based security products (e.g malware detection, cloud-based, endpoint av). Our presentation tends to bridge the research-to-practice gap by sharing our experience when evaluating the real-world vendor's products.

Keywords

Deep Learning, AI-based Security Product, XAI

1. INTRODUCTION

In the past year, we witnessed a dramatic rise in the platforms and apps based on machine learning and artificial intelligence. AI technology has impacted from software and the internet industry to other verticals such as healthcare, legal, manufacturing, automobile, and agriculture. This trend also applied to the security industry, all types of security products (e.g., malware detection engine, firewall IPS/IDS engine and endpoint sandbox) begin to leverage AI technology to escalate their power for better detection rate and quicker response speed. However, at the same time, due to the more and more complex model, especially some deep neural network based engines, it is extremely hard to understand the decision made by the products. Typical concerns including the reaction like "How on earth did the endpoint product quarantine my document?" from a frustrated customer, or a complaint like "How does the firewall failed to detect such a naive XSS attack?" from Dev-Ops team. As a result, an accountable explanation is becoming the necessary part along with the "binary" malicious-or-benign result.

The first generation artificial intelligence based security products have relied on complex modeling based on iterative stages of pattern recognition. AI results are often uncanny in their accuracy but have been criticized because the methodology for how the results were calculated can't be explained. This lack of explainability raises a question about how much trust can be placed in AI results, particularly for mission-critical applications. "The crucial thing most AI solution providers fail to incorporate is explainability. You have to ensure there's no bias in AI algorithms. This is a very important factor in any AI model" mentioned by CTO at IBM India [19]. Similarly, COO and CTO of SAS also said that "explainability of AI is part of a larger effort toward fair, accountable and transparent AI systems. The issues about algorithmic decision making are not new, but the conversation has ramped up in recent years. AI is bringing automated decisions to new domains such as medical diagnostics and autonomous driving, and is building systems that are more complex, less transparent and highly scalable. That combination makes us uneasy.

The main purpose of XAI is to solve the common issue in the machine learning field which is that even the designer failed to explain why the model arrived at a specific decision for the given input. In the academic field, researchers proposed various approaches to break this barrier and make the model not black-box any more. In this paper, we focus on using XAI to peek inside the complicated machine learning black-box and explain security products. Indeed, our work is not the first one to explore this field, several works had already demonstrated the benefit of explaining security products. Inspired by those amazing works and combining the scenarios in our daily work, we found out there is still a big gap between academic research in this field and demandings in the industry world.

Our Angle: Using XAI to explain products is very useful

*Currently working at Google

© 2021 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

in multiple scenarios. The most common case is when the security company leverages XAI to explain their own products to either provide more insight of the detection result to customers or help their engineering teams to improve products. The majority of the XAI work targeting this scenario when designing their tools. The assumption in this scenario is to assume the model as a white-box so that the model can be loaded into the XAI program and the whole training/testing datasets are available too. In addition to this common scenario, there is another use case where customers want to leverage XAI to evaluate the vendor's security products in order to help the purchasing decision. Usually, it is conducted by the company's product team. Besides deploying them to the company's testing environment and feeding benign and malicious samples to monitor the FP and FN, they can also leverage XAI to further peek inside the ML-based security product by checking whether their detection result is reasonable or not.

Our contribution is to share lesson learned when we use XAI to evaluate security products and identify potential XAI research direction to fill in business need.

2. BACKGROUND

XAI is one of a handful of current DARPA programs expected to enable "third-wave AI systems" [7]. To improve the transparency of deep neural networks, researchers start to work on explanation methods to interpret the classification results. Most existing works focus on non-security applications such as image analysis or natural language processing (NLP). Given an input image, the explanation method explains the classification result by pinpointing the most impactful features to the final decision. Common approaches involve running forward propagation or backward propagation in the network to infer important features. More advanced methods produce explanations under a black-box setting where no knowledge of classifier details is available. Explainability enables rules to be established and for the methodology to become more transparent. We can even take the output of those machine learning algorithms and then turn them into explainable rules. Then you can say well, the reason I'm not giving you this loan is because of these factors. Now, instead of just applying the formula, you can also use additional rules. Of course, you have rules for how you want to deal with customers, and you apply the rules and then you can use continuous machine learning to see if your actions were positive or negative for the bank or for your customer.

The main focus of this paper is not to invent new AI explanation method but to apply the suitable XAI method to explain the deep-learning based security product. The main idea of existing AI explanation approaches [5, 12, 8] are to approximate the local decision boundary using a linear model to infer the important features. To explain the prediction result of a security product, we learn an interpretable model locally around the prediction and frame the task as a submodular optimization problem. An essential criterion for explanations is that they must be interpretable. A possible interpretable representation is the presence or absence of a byte or a word. XAI keeps a balance between interpretability and fidelity by minimizing the locality-aware loss while having the complexity be low enough to be interpretable by humans. In the real world, most commercial software are closed source and XAI can only treat them as black boxes. Such a criterion requires the explainer to be model-agnostic. Thus, to learn the local behavior of model as the interpretable inputs vary, XAI draws instances around the original instance by drawing nonzero elements uniformly at random with a limit distance. XAI uses a mixture regression model that can approximate both linear and non-linear decision boundaries given enough randomly selected samples. This gives us the flexibility to optimize the local approximation for a non-linear boundary. We also use 'fused lasso' as a penalty term to capture the dependency between adjacent features. In this way, the XAI can produce high-fidelity explanation results for non-linear and feature dependency of DNN models like security products.

Unfortunately, most of the commonly used machine learning models do not support introspection into their decision process or prvoide explanations of their prediection/bias.

2.1 Attribution-based XAI

Attribution is one approach to interpretability, which highlights input dimensions that are influential to a neural network's prediction [3].

- Depends on where is the XAI method focusing on, XAI methods could target on a local instance or trying to understand the model as a whole.

- Based on the core algorithms to generate the explanation, the tools can be categorized as either gradient-based or perturbation-based methods. Gradient-based methods, such as saliency maps, saliency relevance maps, and class activation maps, calculate the gradient during the backpropagation stage using partial derivatives of the activations from the neural network to generate attributions. Perturbation-based algorithms derive the explanation by randomly perturbing the features in the input data instance and observing variations towards the output class, which only needs to one forward pass.

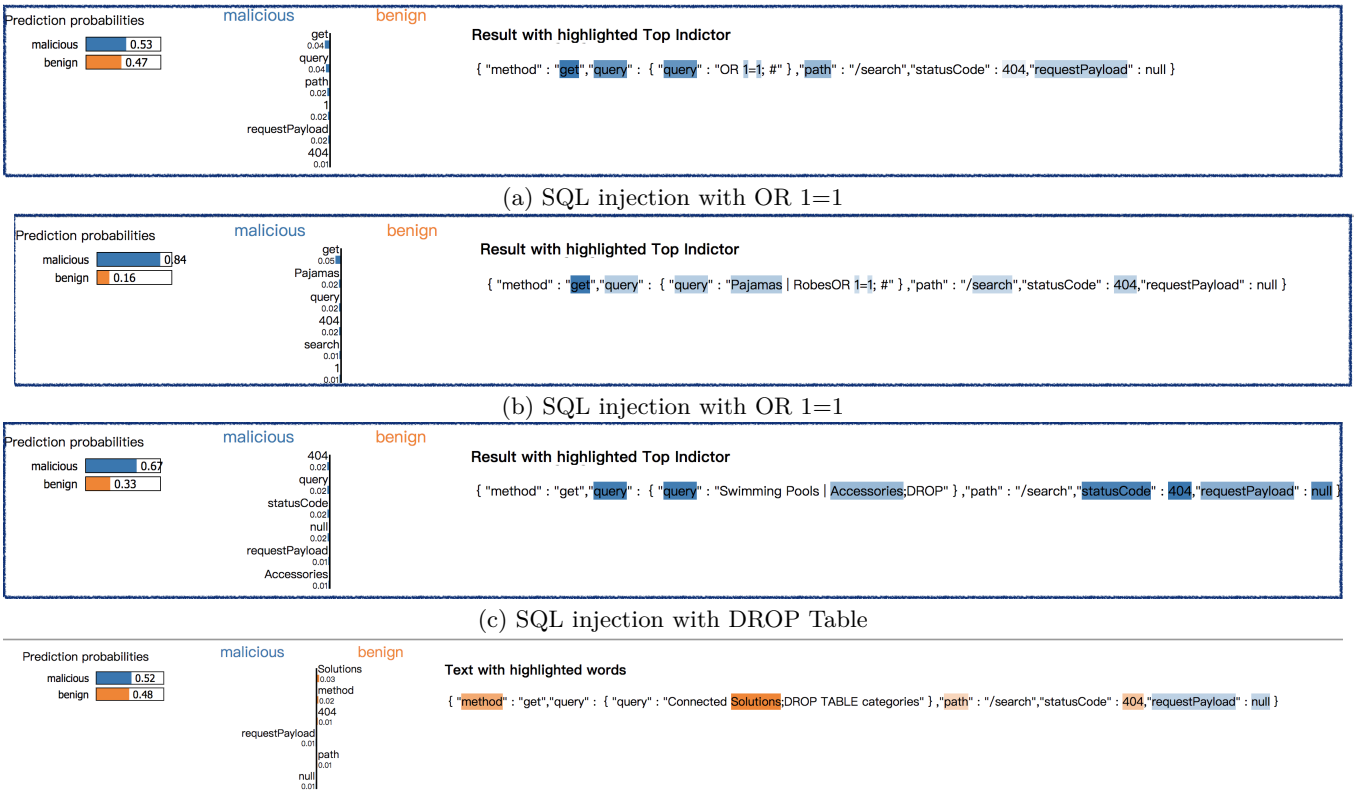
- Based on whether the XAI methods are integrated to the model or not, the tools can be categorized as either model-intrinsic or model-agnositc. Model-agnositc can be applied to any model in general which is described in the following section in detail.

2.2 Popular Model-Agnositc XAI tools

LIME: One of the popular XAI tool is called LIME [12], which refers to Local Interpretable Model-Agnostic Explanations. The idea of this approach is quite intuitive. Instead of training a global surrogate model, LIME focuses on training a `local` surrogate model to explain individual predictions of black-box machine learning model. By generating a dataset consisting of perturbed samples around the target sample and the corresponding predictions, LIME trains an interpretable model (e.g. regression model with lasso), which is weighted by the proximity (e.g distance) of the sampled instances to the target sample.

Local surrogate models with interpretability constraint try to minimize the loss to the original model f and keep the complexity as low as possible, which can be expressed as follows: explanation(x)=$\arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g)$

LIME tool provides extensions for various types of data, including images, tabular and text. In addition, because



(a) SQL injection with OR 1=1

(b) SQL injection with OR 1=1

(c) SQL injection with DROP Table

(d) SQL injection with Create User

Figure 1: explanation of various of SQL injection Attacks

LIME does not leverage the features extracted from the original sample but the presence or absence of entities in the sample, it is extremely valuable to explain the model which trained with non-interpretable features. For example, the NLP tasks usually convert the text words into the feature space (e.g word2vec [2]) and feed the embbeddings to the model. But LIME is still able to measure the importance of each word in the explaination.

Although LIME seems promising, it has few shortcomings which make it not reliable when explaining security products [21]. The first concern is that the toll has the issue to provide consistent explanation for the same sample. The explanations that come out can be different when we repeat the sampling process multiple times for a same sample. This is mainly because the explanation is highly depend on the points that is randomly sampling in the locality. The second concern is the sampling process used in LIME. Since it just purely sampled data points from a gaussian distribution, all the correlation between the features are ignored. This lead to the problem similar with "permutation importance" approach that the `unlikely` data points can be used to learn local explanation models. Since lots of samples used by security products are highly structured (e.g. binary, http headers), we proposed a solution to address this issue.

SHAP: Another state-of-art model-agnostic XAI approach is called SHAP, which leverages an important concept, Shapely Values, in cooperative games from game theory to explain the sample by computing the contribution of each feature to the prediction output. To get the shapely value of a feature i, ideally, we need to calculate weighted sum over all possible feature value combinations other than i. However, with the large number of features, such a requirement is computationally infeasible, therefore, the implementation takes advantage of different model's structures to achieve the optimizations in practice. For example, TreeSHAP takes advantage of the hierarchy in a decision tree's features and support approximation using Monte-Carlo sampling instead of calculating all possible coalitions.

However, only the `KernalSHAP` is model-agnostic. The big difference to LIME is the weighting of the instances in the regression model. LIME weighs the instances according to how close they are to the original instance. The more 1's in the coalition vector, the bigger the weight in LIME. However, SHAP weighs the sampled instances according to the weight which the coalition would get in the Shapley value estimation. Small coalitions (few 1's) and large coalitions (i.e., many 1's) get the largest weights.

Anchors: Anchor [13] is another model-agnostic approach to generate explanations by finding a decision rule that sufficiently anchors the prediction via a perturbation-based strategy. Unlike LIME, which trains a surrogate model, Anchor generates the IF-THEN stype explanation which is much easy-to-understand to humans. It leverages reinforcement learning techniques (multi-armed bandit problem) in combination with a graph search algorithm to reduce the number of model calls.

2.3 Explain Security Products

Previous worksdemonstrated how to explain different types of security products. [17] proposes the framework that is used to improve the interpretability of any IDS and shows the details of how it works. LEMNA [8] designs a high-fidelity explanation method to explain malware classifier and function start detector for binary reverse-engineering. Using a gradient-based approach to identify the most influential local features of Android malware detector [11]. [18] evaluated mulitple security products including *Drebin+*, *Mimicus+*, *DAMD* and *VulDeePecker*.

3. DESING AND IMPLEMENTATION

3.1 Challenges

In order to get a high quality explaination for security products, we have conquared the following challenges.

Challenge 1: Most of the tools provide adaptors for common input types such as image, text or tabular. The main functionality of these adapters is to reconstruct the randomly generated samples around the target data point. However, depending on the type of security products, the input could be binary files, http headers, traffic log etc. Therefore, we have implemented various adaptors for each type of security product.

Besides the lack of adaptor support, existing XAI tools are suitable for standalone model serving API. Therefore, as Figure 3.1 shows, they expect to receive a prediction function from the user to get the prediction result of each permuted sample. However, the majority of security products only provide the end-to-end services. For example, the malicious traffic detection product captures the traffic from the device directly, which means the only way to feed the data to the product is to send the real package from the device.

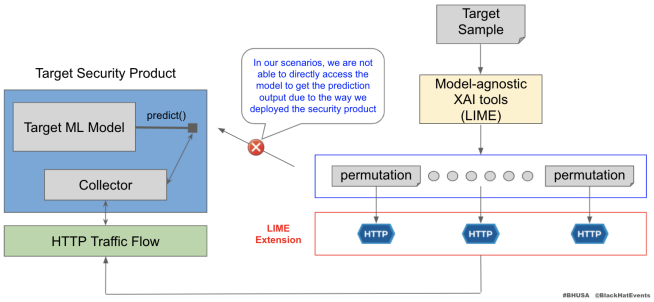


Figure 2: Customize Existing XAI Tools

Challenge 2: Another challenge is that the disadvantage of the sampling process used in LIME and KernalSHAP can be even worse when explaining security products with structured samples. Since these XAI tools simply leverage gaussian distribution to randomly generate samples based on the data to be explained. In our HTTP header case, since the header is highly structured and has grammar in it, Random sampling will generate a large number of invalid headers. It means that these invalid headers are unlikely to be in the training set, and the model's prediction is highly inaccurate which compromises the quality of explanation as well.

To mitigate this issue, we extend the LIME library by adding a module to replace the random sampling with structure-awareness sampling. As a result, our customization not only improves the speed of the tool but also the quality of the explanation. Our idea also applies to other types of samples like binary files, pdf files.

3.2 Explaining Malicious Request Detector

Security is a concern for any public facing web application. In order to assist web developers to defend against attempts from users looking to expose data or bring down an app, malicious request detection application can be used to analyze incoming requests to a target API and flag any suspicious activity. The app would also host a simple UI to display these flagged requests and provide the ability to take precautionary action when necessary. We launch multiple popular cyber-attacks and adopt XAI tool to explain why they successfully identify certain attacks and miss other cases.

Explaining SQL Injection Attack. SQL injection (SQLi) is an application security weakness that allows attackers to control an application's database – letting them access or delete data, change an application's data-driven behavior, and do other undesirable things – by tricking the application into sending unexpected SQL commands. Using SQL injection, attackers can obtain sensitive information like usernames, passwords, credit card numbers, medical records. SQLi attacks usually start with a craft request to the web application with special patterns embedded in a certain part of the URL, and once the backend services construct the SQL query based on this request, the malicious SQL statement will be injected and executed with the privilege as the web server.

Therefore, we adopt several commonly used techniques to craft request to launch the SQL injection attack, and the target DNN-based detection engine is able to detect nearly all of the attacks. However, when we further use our XAI tool to analyze the reason why they are detected, we observe that not every correct detection decision is made with solid and reasonable indicators.

- Pattern '`OR 1=1`': This pattern is injected to make the `WHERE` clause returns the first transaction in the database. If the target database stores the username and password for the web app, this injected SQL statement returns the first id from the users table no matter what the username and password are, which is very often the administrator. In this way, the attacker not only bypasses authentication but also gains administrator privileges. They can also comment out the rest of the SQL statement to control the execution of the SQL query further. The first two scenarios are based on this pattern, and from the explanation showed in Figure 1(a) and (b), the suspicious pattern '1=1' is identified by the detection engine as a strong indicator to mark the request as malicious.

- Pattern '`DROP TABLE`': As Figure 1(b) shows, for the injected SQL statement aims to drop tables in the database, although the tool determines the request as malicious with a higher confidence score, it fails to utilize the vital suspicious indicator `DROP` as the evidence but another neutral pattern the header.

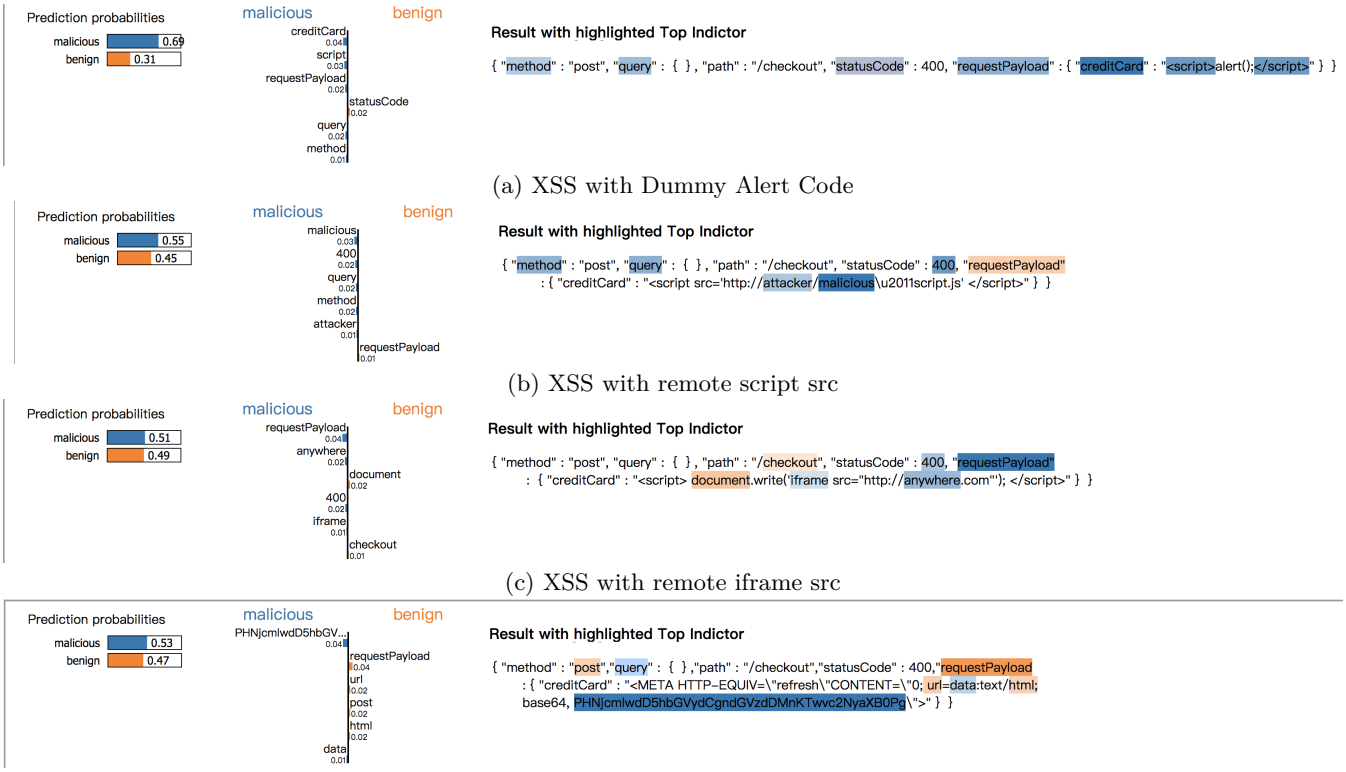


Figure 3: explanation of various of XSS Attacks

Explaining XSS Attack. Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites, which occurred when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it. An attacker usually crafts a `POST` request with the malicious script embedded in the body to the server, and once victim loads the page with the injected malicious script to the browser, it has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

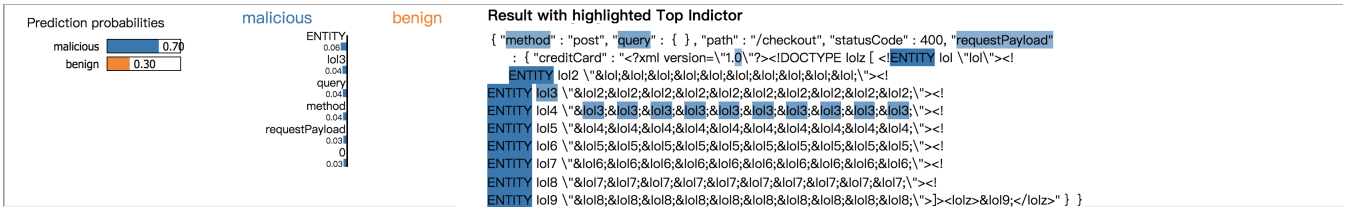
Therefore, we utilize multiple common methods to craft the request to launch XSS attack, and the target DNN-based detection engine managed to detect the majority of them. We further explained why the engine makes such a decision for XSS attack as follows:

- `XSS with Dummy Alert Code`: An alert statement is always used as demonstrate to test whether the injected script gets executed or not. At first, we explain a crafted request with this dummy alert code injected. As Figure 3(a) shows, the target tool not only identifies this request as malicious with high confidence (0.69), but also leverages the `<script>` tag in the injected part as the evidence, which aligns with the intuition of security experts.

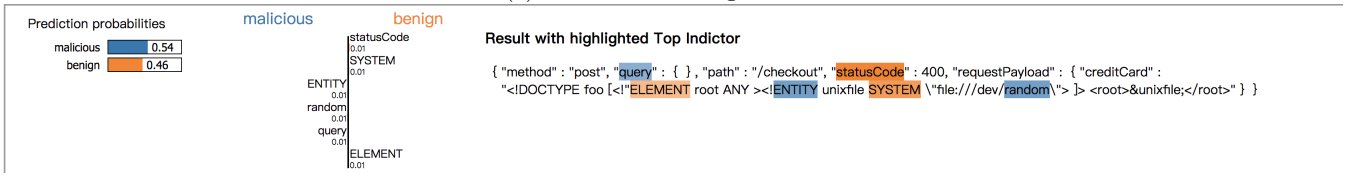
- `XSS with remote code`: To launch complicated attacks, attackers need to find a way to load script from a remote file, which always involves the `src` attribute of *script* or *iframe* tag. Figure 3(b) shows the explanation of a crafted request using *script* tag to load remote code; instead of using the *script* tag as the evidence, the tool leverages the domain name and file name from the remote address as the indicator. This information can be easily bypassed by attackers with a common name. Figure 3(c) shows the explanation of a crafted request that loads a malicious page into an *iframe*; the tool successfully captures the *iframe* tag as a suspicious pattern, but treat the word *document* belongs the injected script as a positive indicator. This explains why this request is in the boundary between benign and malicious (0.49 vs 0.51).

- `XSS with Base64 encoding evasion`: The odd thing about meta refresh is that it doesn't send a referrer in the header - so it can be used for certain types of attacks where you need to get rid of referring URLs. Additionally, with the directive URL scheme, attackers successfully hide any visibly sensitive pattern like the word SCRIPT or the JavaScript directive in it which is base64 encoded according to the FC 2397.

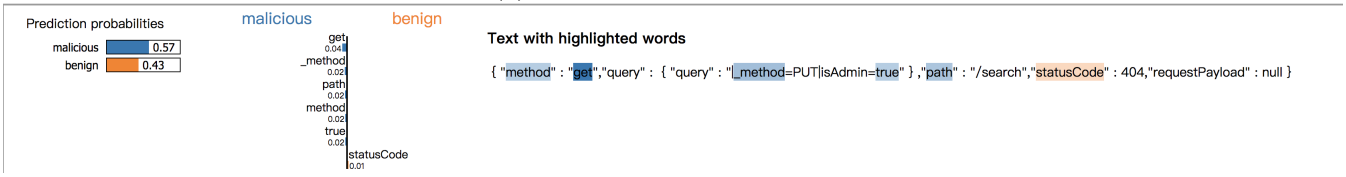
  Figure 3(d) shows the explanation of a crafted request with the code "`<script>alert('test3')</script>`" injected in the base64 format as *PHNjcmlwdD5hbGVyd*



(a) XXE Billion Laughs Attack



(b) XXE DoS under Unix servers



(c) SSRF Attack

Figure 4: explanation of various of XXE and SSRF Attacks

CgndGVzdDMnKTwvc2NyaXB0Pg. The encoded part is used as the strongest indicator to mark this request as malicious (highlighted in dark blue).

Explaining XXE Attack. An `XML external entity` (XXE) attack is one that can take place against a web app that parses XML input. Once the XML input contains a reference to an external entity and is then processed by an XML parser that has not been configured appropriately, an attacker is able to launch various of attacks including DoS, SSRF and etc.

- `XXE with Billion Laughs Attack`: Billion laughs attack [4], also referred to as an XML bomb or as an exponential entity expansion attack, is a type of denial-of-service (DoS) attack which is aimed at parsers of XML documents. Figure 4(a) shows an example consists of defining 10 entities and each defined as consisting of 10 of the previous entity, with the document consisting of a single instance of the largest entity, which expands to one billion copies of the first entity. The name the "billion laughs" comes from the fact that first entity is the string "lol". Due to the frequently cited entities, the victim computer memory quickly used up while process parsing the XML. Although various of defense mechanism was proposed to play against this kind of attack include capping the memory allocated in an individual parser if loss of the document is acceptable, or treating entities symbolically and expanding them lazily only when (and to the extent) their content is to be used, the best way to prevent it is blocked by firewall.

  The target tool is able to detect billion laughs attack, and the large number of occurrences of word `ENTITY` is high suspicious, in addition to the word `lol3`, which are highlighted as high-risk indicators that make the tool make the request as malicious with a higher confidence level (0.7).

- `XXE for denial-of-service under Unix servers`: On some systems, it is possible to mount a Denial of Service attack by telling the XML parser to read from the never-ending Unix-file `/dev/random`. XXE attacks can also be used to make the web server connect outwards using HTTP, or connect to internal servers not normally available from outside the firewall. Figure 4(b) shows the explanation of such an attack, and clearly, the key filename `/dev/random` is marked as a highly suspicious indicator.

Explaining SSRF Attack. Extracting parameters from HTTP message and getting resource URLs could be vulnerable to injection attacks that may change the semantics of the intended resource. In a Server-Side Request Forgery (SSRF), the vulnerable application composes the URL using data from the HTTP request that could also affect the scheme, host and port parts of the URL, so the vulnerable application acts as an open proxy/relay for the attacker. Two classes of attacks are relevant here: HTTP parameter/path pollution (HPPP) and Server-Side Request Forgery (SSRF). As Figure 4(c) shows, the tool manages to identify the suspicious keyword `_method` and `True`, but fails to detect the more important indicator, the parameter `isAdmin`.

3.3 Concept-based Explanation

Traditional XAI methods can only explain each individual sample. However, the core value of AI-based security products is to provide high-level intelligence on low-level raw features. Therefore, evaluating the quality of this high-level intelligence instead of quantifying the importance of each input feature of certain samples is vital to security buyers, which is known as `concept-level explanation` [6, 20]. By

```
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]}}
```

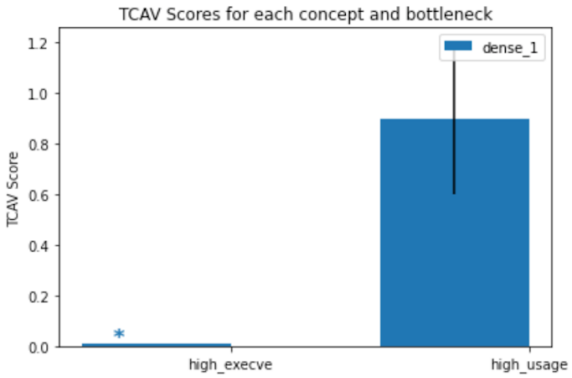


Figure 5: Concept based explain for syscall-based malware detector

deciphering the concepts they base their decisions on, we can characterize the global behavior of security products in a way understandable to humans, by explaining how they use concepts in arriving at particular decisions. Since such types of explanations are similar to human-level thinking, it is much easier to convince the whole security team to choose the good vendor.

A concept is characterized by a set of samples that have specific characteristics in common. We propose to leverage the concept distilled from domain experts and evaluate these concepts against security production. When we search concept based explanation solutions, we have the following requirements to effectively evaluate the security production.

- The concept based explainer can be universally pluggable to any security production.
- The concept based explainer can evaluate against blackbox AI model.
- The concept based explainer can customize the concept type.

With reasonable engineering effort, we are able to make the explainer TCAV[9] universal pluggable and adopt customized concept types. However, we are not able to find an academic solution that meets the second requirement which makes it impossible to evaluate real security products. The current concept level explainers require a whitebox model and knows the structure of the AI model. There is no concept based explainers can evaluate against blackbox AI model. We believe this is a research gap that future academic research could try to fill in and has practical business needs as well.

To evaluate our idea, we lift our requirement on the blackbox model and trained a surrogate AI model of system call malware detector. As shown in Figure 6, system-call malware detector predicts malware behavior based on the system call usage of each process. The detector takes the usage as numeric features and feeds them into AI model for prediction. Based on the performance report, our detection model is able to obtain a high accuracy malware detection model with over 99% accuracy when detecting malicious behavior. To test whether the AI model is able to detect several well known attacks, we picked two concepts, namely SQL injection, cmd injection. We think SQL injection has the following pattern: high file I/O, high memory usage, and high network throughput. For cmd injection, we think it has high execve usage. These features come from domain expertise and historical data.

Figure 5 is the evaluation result from TCAV. It shows the significant score of the concept on the last dense layer of the AI model. High_usage is the sql injection concept and high_execve is cmd injection concept. As you can see from the result, the SQL injection concept takes significant weight (TCAV score = 0.9) when detecting SQL injection behavior while cmd injection concept takes insignificant impact (TCAV score = 0.7) when detecting cmd injection. Although the AI model has high detection accuracy on cmd injection, the model does not make the judgement based on the human understanding. For SQL injection, we found that the model judgement basis is consistent with human understanding.

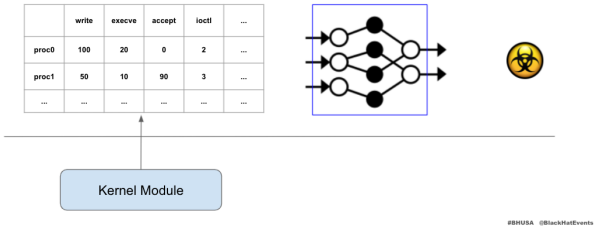
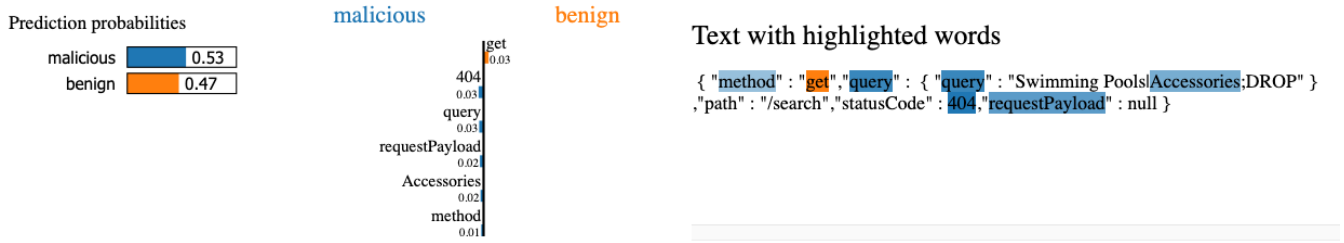


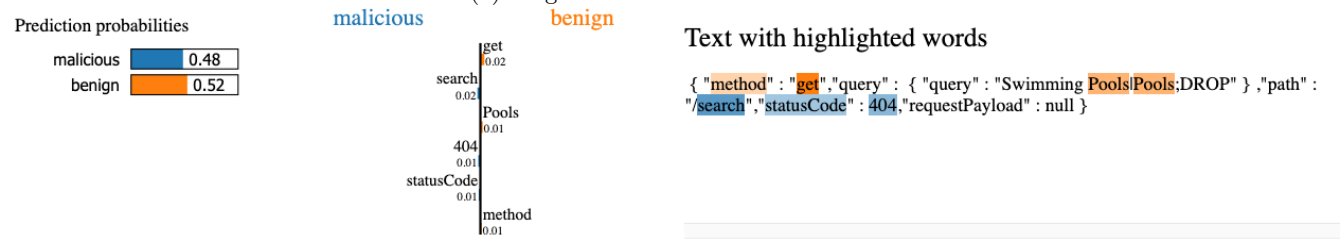
Figure 6: Syscall-based Malware Detection Tool

4. XAI ABUSED BY ATTACKERS

Based on some interesting observations when we explain the security products, we realized that the attacker is able to



(a) Original Malicious HTTP Header



(b) Adversarial Example crafted based on explaination

Figure 7: Crafting Advesarial Malicious Header using XAI

leverage the XAI technique to enhance their attack against AI-based security products [16, 1, 14].

4.1 Craft Adversarial Example using XAI

Adversarial examples [15, 10] are inputs that are similar to a correctly classified input, but which are misclassified by the machine learning models that an attacker has intentionally designed to cause the model to make a mistake. They pose a serious threat especially in security-critical autonomous systems such as self-driving cars. Using tradition approach to generate adversaial examples against commercial security product has the following disadvantages:

- Most approaches to generate adversarial examples tend to minimize the distance between the adversarial example and the instance to be manipulated, while shifting the prediction to the desired (adversarial) outcome. White-box attacks need to access the internal weights and leverage them to calculate the gradients of the target DNN model for generating adversarial examples. However, an attacker is highly unable to gain such knowledge of a comercial security product, which leads to the fact that the white-box attacks fail to work.

- Model-agnostic attacks only require the access to solely the input and output of a classifier (eg. access to the prediction function only) for constructing the adversarial examples. These works are mostly concerned with training a generative model on a target network so that the samples generated by them are adversarial using the white-box approaches. Moreover, for various types of defenses, such methods often require retraining their generator on a different substitute network. Even using the 'finite difference method for gradient estimation', we still need to generate tons of real samples and wire it up with the attack tools.

- The majority of approaches are targeted for image type input which does not have grammar-restrictions. For adversarial malware samples (e.g `HTTP headers` or `binary code`), besides fooling the security product, they also need to be the valid file without compromising the malicious functionalities. Prior methods failed to consider such constraints when generating the adversarial example.

Our idea is to leverage the samples with poor explanation, which means the security product successfully identified the sample as malware but using the wrong indicator or feature. We then replace the part of 'wrong-explaination' input with the benign data and lower the malicious score to bypass the detection but preserve the malicious behaviours.

For example, figure 7 shows a sample which the security product marked as malicious but based on unrelated features. The actual SQL injection script 'DROP' is not a high weight attribute for the AI model to label it as malicious. We then pick a high weighted attribute (i.e. 'Accessories') which we know is unrelated to SQL injection based on our domain expertise. By only modifying the value of this attribute to a benign score value (i.e. 'Pools'), we are able to preserve the malicious behavior while letting the AI model flip its label to benign.

4.2 Information leakage from security products

Since XAI is able to provide additional information besides the prediction result, we believe any insight provided by XAI could help attackers launch the real attack. Commercial security products often depend on hybrid technologies. Using a malicious HTTP header detector as an example, it is common for the detector to combine heuristic rules and a security AI model to analyze HTTP requests. The goal as an attacker is to understand the rule behind the security product in order to find a way to bypass the detector.

To simulate real security production setup, we trained a surrogate AI model for a malicious HTTP header detector. The model is able to perform at high accuracy. To obtain better performance during runtime detection, we intended to add a list of heuristic rules for the obvious SQL injection attack pattern in order to speed up the detection performance.

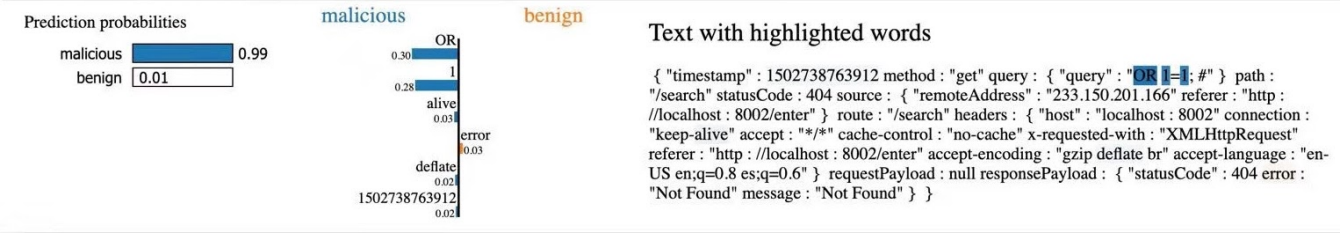


Figure 8: Information leakage from malicious HTTP detector

We use a LIME Text based explainer to 'debug' the malicious HTTP header detector. We found that XAI can easily identify the heuristic rule and the explanation result is quite deterministic. We run the experiment 10 times, LIME can accurately capture the rule with a high confident score. Figure 8 is an example with SQL injection attack. In this example, LIME can accurately highlight 'OR 1=1' which is one heuristic rule we put in the security product.

5. CONCLUSION

While more and more security products leverage deep-learning to improve the detection rate, it is necessary to dig deeper to figure out whether the AI models learn the correct indicator of malicious behaviors or just manage to just find the neutral difference between benign sample and malware. With the explainable artificial intelligence (XAI) techniques, we are able to examine the fundamental of AI-models trained by security products.

6. REFERENCES

[1] N. Carlini. Is ami (attacks meet interpretability) robust to adversarial examples? *arXiv preprint arXiv:1902.02322*, 2019.

[2] K. W. Church. Word2vec. *Natural Language Engineering*, 23(1):155–162, 2017.

[3] A. Das and P. Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[4] A. Ekholm. Den ljusnande framtid ar vard. 2010.

[5] W. Garcia, J. I. Choi, S. K. Adari, S. Jha, and K. R. Butler. Explainable black-box attacks against model-based authentication. *arXiv preprint arXiv:1810.00024*, 2018.

[6] A. Ghorbani, J. Wexler, J. Zou, and B. Kim. Towards automatic concept-based explanations. *arXiv preprint arXiv:1902.03129*, 2019.

[7] D. Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.

[8] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing. Lemna: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 364–379. ACM, 2018.

[9] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[10] A. Kurakin, I. Goodfellow, S. Bengio, et al. Adversarial examples in the physical world, 2016.

[11] M. Melis, D. Maiorca, B. Biggio, G. Giacinto, and F. Roli. Explaining black-box android malware detection. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 524–528. IEEE, 2018.

[12] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[13] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[14] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[15] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[16] G. Tao, S. Ma, Y. Liu, and X. Zhang. Attacks meet interpretability: Attribute-steered detection of adversarial samples. *arXiv preprint arXiv:1810.11580*, 2018.

[17] M. Wang, K. Zheng, Y. Yang, and X. Wang. An explainable machine learning framework for intrusion detection systems. *IEEE Access*, 8:73127–73141, 2020.

[18] A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck. Evaluating explanation methods for deep learning in security. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 158–174. IEEE, 2020.

[19] D. Weisinger. https://formtek.com/blog/explainable-ai-xai-turning-the-light-on-todays-black-box-ai-algorithms/. 2019.

[20] C.-K. Yeh, B. Kim, S. Arik, C.-L. Li, P. Ravikumar, and T. Pfister. On concept-based explanations in deep neural networks. 2019.

[21] Y. Zhang, K. Song, Y. Sun, S. Tan, and M. Udell. "why should you trust my explanation?" understanding uncertainty in lime explanations. *arXiv preprint arXiv:1904.12991*, 2019.