The Holy Grail of “Systems for Machine Learning”

Teaming humans and machine learning for detecting cyber threats

Ignacio Arnaldo
PatternEx
San Jose, CA, USA
iarnaldo@patternex.com

Kalyan Veeramachaneni
MIT LIDS
Cambridge, MA, 02139, USA
kalyanv@mit.edu

ABSTRACT

Although there is a large corpus of research focused on using machine learning to detect cyber threats, the solutions presented are rarely actually adopted in the real world. In this paper, we discuss the challenges that currently limit the adoption of machine learning in security operations, with a special focus on label acquisition, model deployment, and the integration of model findings into existing investigation workflows. Moreover, we posit that the conventional approach to the development of machine learning models, whereby researchers work offline on representative datasets to develop accurate models, is not valid for many cybersecurity use cases. Instead, a different approach is needed: to integrate the creation and maintenance of machine learning models into security operations themselves.

1. INTRODUCTION

Over the past decade, various disciplines, from aircraft monitoring and healthcare to IOT engineering and cybersecurity, have taken up machine learning (ML) systems that monitor large amounts of data in order to draw conclusions and/or suggest interventions. By adopting such systems, stakeholders aim to leapfrog traditional post-hoc investigations by humans and achieve more proactive and predictive analysis and prevention. In particular, recent years have yielded a large corpus of research focused on using machine learning to tackle a wide range of cybersecurity use cases [13]. In an ideal scenario, machine learning would enable us to quickly detect precursors to adverse events, enabling institutions to either prevent or reduce the dwell time of cyberattacks.

A few years ago, our team at PatternEx set out to develop and deliver a cybersecurity machine learning platform. This paper summarizes the experience and establishes the key challenges faced along the way. The most striking observation is that a common modus operandi in both research and commercial ventures involves identifying a “*labeled dataset*”, generating a complex machine learning model, and declaring victory when high accuracy is achieved. However, the approaches generally suffer from a lack of realism and representativeness [3] and are rarely adopted in the real world. Moreover, access to representative data is but one of the roadblocks in cybersecurity. In fact, given the breadth of the challenges, we predict that making machine learning work in this domain will be the holy grail of machine learning systems development:

Challenge #1: Where art thou “labels”? A fundamental assumption undergirding current efforts in this area is that “labeled examples” of previous attacks are readily available or could be easily acquired via an interactive system. This assumption is misguided or academic at best. The acquisition of “labeled datasets” remains a key barrier to ML adoption, for several reasons: certain datasets pertaining to previous breaches cannot be made public due to political or privacy-related concerns, and the shifting nature of attacks makes it challenging to maintain up-to-date labeled datasets. We present different label acquisition strategies and their benefits, drawbacks, and challenges in Section 3.

Challenge #2: A giant leap from “AutoML” to scalable model deployments. The scale of model development required for ML-based cybersecurity solutions to be effective is unprecedented, and must be coupled with real-world operational settings and requirements. In a nutshell, supporting the ML-based detection of a wide range of threats introduces the need to simultaneously analyze a diversity of data sources and to learn and deploy a large number of disparate machine learning models (each possibly learned from minimal training examples catalogued on a daily basis). Meanwhile, real-time requirements demand a complex data processing infrastructure. The systems currently available for automating machine learning are dwarfed by the scale and specifics of these requirements, which we discuss further in Section 4.

Challenge #3: Real-time explanations. In real settings, complex, multi-stage attacks that are worth detecting usually require analyst investigations. This means that any findings made by ML models must be integrated into existing analyst workflows. In fact, the use of ML models for detection introduces challenges related to the interpretability and explainability of machine learning models. We describe more on this in Section 5.

Challenge #4: It’d take a village. The changing nature of the cyberthreat landscape introduces the need to create and maintain ML models on a continuous basis. This requirement involves new tasks and workflows for the cybersecurity operations workforce, which will need to be augmented with a new set of roles. We expand upon this observation in Section 6.

While usage of machine learning for cybersecurity has been piecemeal basis, the area of machine learning systems development has been similar. It seems like the community is chipping away one problem at a time, AutoML systems try to find best fit models [10], a separate system helps humans label datasets [25], another theme of work focuses on model development and deployment [22] and yet another set of systems focus on explaining models. Perhaps the most compelling learning of all for us has been that building a machine learning driven cybersecurity platform requires all of the above, which begs the question: would this be the holy grail of machine learning systems development? We argue that it is.

2. RELATED WORK

There is a large corpus of research focused on tackling cybersecurity use cases with machine learning. In particular, there are three cybersecurity problems for which the availability of labeled datasets has resulted in a proliferation of machine learning applications: file analysis for malware detection [13], botnet detection [29], and the domain or URL detection problem [6, 20].

However, the cyberattack detection problem spans a wide range of malicious activities. As of the day of this writing, MITRE's ATT&CK initiative [30] lists 282 techniques used by adversaries at different phases of an attack. There are isolated works demonstrating the feasibility and benefits of adopting machine learning for the detection of some of these techniques. To illustrate this point, we provide examples of works that tackle different stages of an attack: browser exploits [14], web application attacks [16], TOR connections [18] and VPN traffic [8], lateral movement [31] and credential access [12], HTTP [7] and DNS tunneling [9, 23], encrypted malware flows [1], web-based command and control [33], and data exfiltration [2, 19]. While many of these approaches are promising and convey meaningful insights into designing detection strategies, they do not discuss how their methods fit into the type of broader detection strategy needed in real-world security operations. In fact, the modeling approaches reported in these works must be coupled with appropriate support in order to function in real life. In this paper, we discuss these requirements and their associated challenges in depth, with a special focus on model deployment and maintenance and the integration of model findings into existing operational workflows.

3. LABEL ACQUISITION SCENARIOS IN CYBERSECURITY

In this section, we take a deep dive into cybersecurity's label acquisition challenge. To drive the discussion, we first classify common strategies for label acquisition in Figure 1. The figure shows a series of paths leading to the acquisition of labels (numbered from 1 to 7). In the following subsections, we describe the steps involved in each of these paths, along with their challenges, benefits, and drawbacks.

3.1 Real data versus lab

The first distinction involves whether the data originates in a real-world system or organization, or instead is the outcome of lab simulations. Lab simulations present several benefits: they are able to simulate a wide range of attacks, and result in the creation of high-quality (low-noise) data at a fast pace. This label acquisition strategy corresponds to path (1) in Figure 1. On the other hand, depending on the simulation strategy, it is possible data generated in this way will suffer from a lack of generality, realism, and representativeness [3]. In the authors' experience, this is particularly the case when the goal is to generate benign data – shown as path (2) in the figure – because it is challenging to replicate the great variety of legitimate activity and traffic patterns observed in real-world organizations. Therefore, relying on simulated data alone might prove insufficient to build detection models, which motivates label acquisition from real-world data.

3.2 Continuous monitoring

This refers to cases where the data has been monitored, analyzed, and investigated in a timely manner (in close-to-real time), as opposed to cases where the data has been analyzed and investigated *a posteriori*. Continuous monitoring occurs in most medium and large organizations, where a series of security solutions monitor the traffic and activity at the organization. Upon the detection of malicious activity, they either prevent further activities, or generate alerts that are then reviewed, investigated, and (if necessary) remediated by a security operations team.

3.3 Analyst confirmation

Alerts generated by continuous monitoring mechanisms are reviewed and investigated by security analysts. In most security operations, ticketing systems are used to monitor, track, and manage alert resolutions, and security analysts log the investigation outcomes upon closing the tickets that they are assigned. These logs correspond to a *label* that can be leveraged by AI security analytics (paths 3 and 4 in Figure 1). It is important to stress that labels are being captured in security operations today.

To put AI security analytics opportunities in context, here is a high-level view of the different mechanisms used to generate alerts in today's security operations:

3.3.1 Detection based on known malicious entities

This detection is performed based on indicators of compromise, such as signatures, file hashes, and IP and URL or domain blacklists (shown as ”Repeat entity” in the figure).

3.3.2 Detection based on known malicious activity patterns

Instead of looking for known malicious entities, this detection approach leverages existing knowledge of malicious activity patterns for detection (shown as ”Repeat pattern” in the figure). It is implemented predominantly as rules and (in isolated cases) as supervised machine learning models; each method presents its advantages and drawbacks. In a nutshell, rules are easy to interpret (more on this in Section 5) and do not require training data, whereas supervised models provide improved detection accuracy (if training data is available) at the cost of interpretability.

3.3.3 Detection based on anomaly detection

This last approach uses anomaly detection to surface rare activity patterns. These methods rely on baselining strategies, whereby historic activity patterns are modeled and new activities are assigned a score that captures their deviation from the norm. The analysis can be built on a population basis (by obtaining baselines for a group of entities), or on an individual basis (by obtaining an activity baseline for each entity). This approach can detect attacks where neither the entities or the patterns are known *a priori*, given that the as-

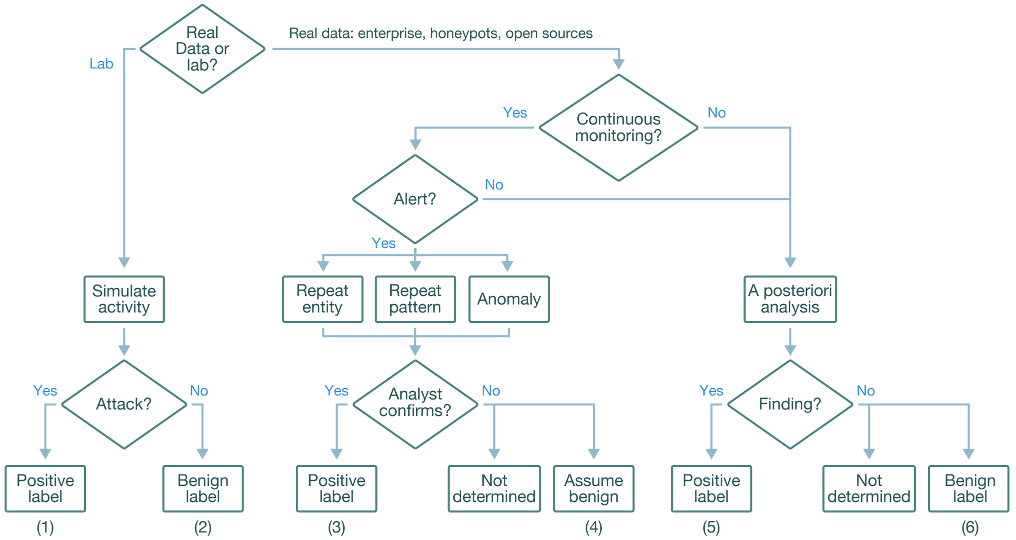


Figure 1: Label acquisition paths in cybersecurity. The acquisition of labeled data can be the result of lab simulations (1 and 2), analyst investigations (3 and 4), or a posteriori threat hunting exercises (5 and 6).

sociated activity patterns are expected to be different from previously observed activity. On the other hand, in practice, they also generate a large number of false alerts, given that not all anomalies actually correspond to malicious activities. For instance, activity patterns may present anomalies due to misconfigurations, seasonality, or legitimate changes in the activities of the monitored entities, among other reasons.

3.4 A posteriori analysis

As shown in Figure 1, there are two paths leading to a posteriori analysis, often referred to as *threat hunting* in the cybersecurity domain. The first path corresponds to cases where continuous monitoring is in place, i.e. activities that were monitored and analyzed, but were not alerted upon. The second case corresponds to scenarios where continuous monitoring is not taking place. The latter is currently a frequent situation: for instance, organizations today rarely perform security analytics on DNS traffic or cloud-based applications.

A posteriori investigations allow security researchers to perform in-depth investigations to understand whether unclassified activities correspond to attacks (path 5 in Figure 1) or to benign activities (path 6), with the benefit of not having to comply with deadlines imposed by service level agreements. The drawbacks of this approach are twofold. The first issue has to do with the timing of the investigations: it is hard to perform fully-fledged investigations after the fact because the state of the resources involved in the attack might have changed, and actors and attack infrastructure might not be active anymore. This often leads to a roadblock in these investigations, making it difficult if not impossible to assess the reach of the analyzed activities, and to determine whether they were malicious. The second limiting factor is operational: given that these one-off investigations require ad-hoc workflows, researchers might require additional tools and analysis infrastructure that are not readily available at their organizations, especially in cases where there is a lot of data to analyze.

4. THE MANIFOLD DEPLOYMENT LEAP

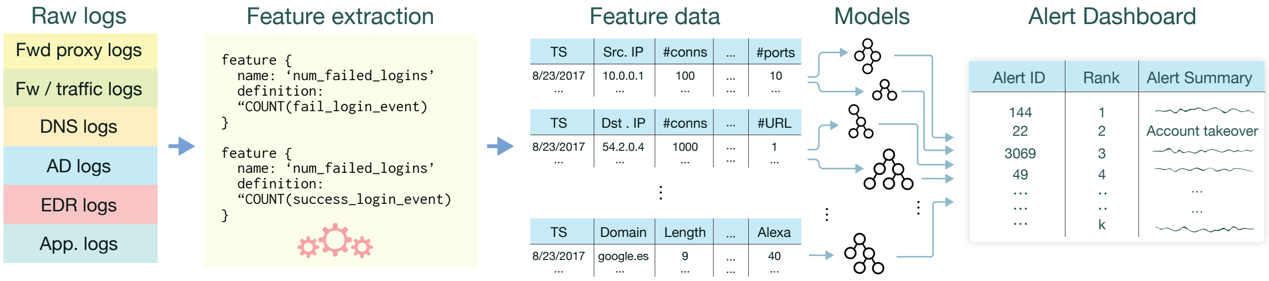
We introduce the challenges involved in building the processing infrastructure necessary to adopt a machine learning-based attack detection strategy for the cybersecurity domain, given the wide range of possible threats and data sources involved. Such infrastructure would need to ingest multiple data sources, analyze the activity of multiple entities, and leverage machine learning models to generate alerts.

4.1 A primer on machine learning pipelines

Machine learning pipelines involve a sequence of data transformations that have been devised by experts to score entities. Generally, we begin by extracting *features* from timestamped log data, and next obtain entity scores using these extracted features and machine learning models. The steps involved in ML pipelines are schematically represented in Figure 2 and detailed in the following.

Step 1: Ingestion of raw logs. *Logs* are files that register a time sequence of events associated with entities in a monitored system. Logs are generated in a variety of formats: `json`, `comma-separated-values`, `key-value` pairs, and `event logs` (event types are assigned unique identifiers and described with a varying number of fields). The first step entails parsing these *logs* to identify entities (e.g, IP addresses, users, domains, etc.) relationships between entities (one-to-many or many-to-many), timestamps, data types and other relevant information about the relational structure of the data. The data is then stored either in the relational data model (`database`) or as-is. Either way, in this step, a relational model is designed based on prior knowledge of how the data is collected and what it means.

Step 2: Feature extraction. Once the data has been parsed into a relational structure, the next step consists of extracting, for each entity instance, a series of *features* that describe its activity. For example, for any given IP address, we can average the number of bytes sent and received per connection over a time interval of an hour. The output of this phase is generally an entity-feature matrix, in which



```
feature {
  name: ‘num_failed_logins’
  definition:
  “COUNT(fail_login_event)
}

feature {
  name: ‘num_failed_logins’
  definition:
  “COUNT(success_login_event)
}
```

TS	Src. IP	#conns	...	#ports
8/23/2017 ...	10.0.0.1 ...	100 ...	... ...	10 ...

TS	Dst . IP	#conns	...	#URL
8/23/2017 ...	54.2.0.4 ...	1000 ...	... ...	1 ...

TS	Domain	Length	...	Alexa
8/23/2017 ...	google.es ...	9 ...	... ...	40 ...

Alert ID	Rank	Alert Summary
144	1	
22	2	Account takeover
3069	3	
49	4	
...	..	...
...	..	...
...	k	

Figure 2: A sequence of data transformations performed by machine learning pipelines for cyberthreat detection. From left to right: a wide range of data sources are ingested and features are extracted to describe the activity of the entities observed in the data (source IPs, destination IPs, domains, etc.). Finally, a set of machine learning models score the activity of each entity instance and populate an alert dashboard.

each row corresponds to an instance of an entity and each column is a feature.

Step 3: Entity scoring. Each entity receives scores generated by machine learning models. The scores represent how rare the activity of the entity is in the case of anomaly detection models, or how closely it resembles a malicious activity pattern in the case of supervised models.

Step 4: Alert generation. In the final layer, the scores generated by machine learning models are aggregated into alerts in a process known as *alert correlation* [21]. These alerts populate the alert dashboard and are reviewed by security analysts.

4.2 Variety of sources

The first challenge involved in incorporating machine learning into cyberattack detection is the variety of data sources in the cybersecurity space. In the authors' experience, organizations commonly store and index around 50 data sources, either for forensics or for live analysis. Such variety makes it challenging to ingest and model the data: on one hand, parsing logic needs to be developed for each data source, and on the other hand, the information useful for threat modeling needs to be identified for each source, which remains a time-consuming process that can only be performed by skilled domain experts.

This challenge becomes more acute when the goal is to build generic cybersecurity platforms, because it is then necessary to account for different products and versions of security devices and applications. In fact, for the categories shown in Figure 2 such firewalls, DNS, EDR etc., there are multiple solutions that generate logs, with different information content and different formats. Moreover, log formats are often augmented over time as improvements are made to the solutions' capabilities and new versions are released, which involves continuous maintenance of the ingestion logic.

4.3 Variety of threats

The biggest challenge for AI security analytics is arguably the variety of techniques used by attackers. As noted before, MITREs ATT&CK initiative [30], the most comprehensive resource to date, currently lists 282 techniques used by adversaries in different phases of an attack – an undercount, given that new techniques are continuously being implemented and used by adversaries.

As shown in Figure 2, analysis of the activity of common entities such as internal IPs, external IPs, domains, users etc., can be leveraged in combination with machine learning models to identify malicious techniques. However, it is important to note that the activity traces necessary to identify a given technique can be scattered across data sources. Therefore, the analysis of cybersecurity data introduces a *many-to-many mapping from data sources to entity activity*, as well as a *many-to-many mapping from entity to threat or technique.*

These observations suggest the need for modular pipelines in which multiple entities are modeled, and where the activity of a given entity is extracted from multiple data sources and fed to one or more machine learning models. The design and implementation of such a manifold system represents another challenge, because most machine learning-ready, end-to-end analytics solutions aim at supporting a reduced number of use cases.

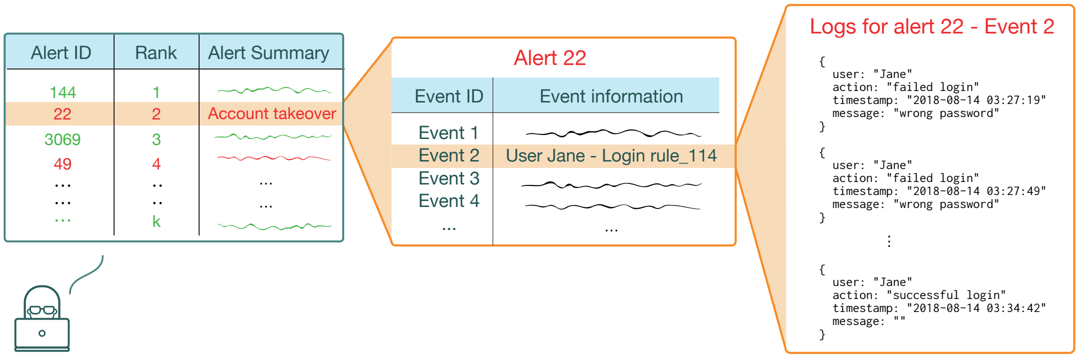
4.4 Close-to-real-time requirement

Due to regulation, policy, and/or service-level agreements, many organizations require incident response times to remain in the sub-hour range. This includes the threat detection step, which in the case of AI security analytics involves the steps described in Section 4.1, along with investigations and incident response. This means that threat detection needs to be performed in close-to-real-time; in the authors' experience, detection times in the 5 to 15 minute range after the event are acceptable in operational scenarios.

Despite advances in big data computing, the large scale of data remains a significant roadblock in the attempt to deliver close-to-real-time detections. While some data sources may be small in size, the daily volume of sources such as firewall, proxy, or DNS logs range from a few GBs for small and medium businesses to multiple TBs for large organizations. The ability to process data for machine learning model consumption at the multi-terabyte scale is achieved with distributed streaming computing frameworks which, in turn, introduce a series of maintenance challenges. (While these maintenance challenges can limit the adoption of AI security analytics, they are not unique to cybersecurity, and therefore we consider their analysis out of the scope of this paper.)

5. EXPLAINING ML-GENERATED ALERTS

In this section, we describe the process that security analysts currently follow while investigating threats and describe the challenges associated to integrating ML-generated alerts.



Alert ID	Rank	Alert Summary
144	1	
22	2	Account takeover
3069	3	
49	4	
…	..	…
…	..	…
…	k	

Alert 22

Event ID	Event information
Event 1	
Event 2	User Jane - Login rule_114
Event 3	
Event 4	
…	…

Logs for alert 22 - Event 2

```
{
  user: "Jane"
  action: "failed login"
  timestamp: "2018-08-14 03:27:19"
  message: "wrong password"
}

{
  user: "Jane"
  action: "failed login"
  timestamp: "2018-08-14 03:27:49"
  message: "wrong password"
}

         ⋮

{
  user: "Jane"
  action: "successful login"
  timestamp: "2018-08-14 03:34:42"
  message: ""
}
```

Figure 3: Schematic representation of the three-layer interface used by security analysts. The *alert summary view* (left) displays all the alerts. The *event view* (center) shows the set of events that were aggregated in the same alert. The *raw data view* (right) shows the logs, packets, files, process information, etc. for each event included in the alert.

5.1 Existing analyst interface

Security analysts review alerts and take appropriate actions to remediate actual threats. Alerts are generated either by security devices or analytics tools, and are aggregated in a single area for digestion by the analysts. Figure 3 shows a schematic representation of a common interface used by security analysts to review and investigate alerts. As shown in the figure, the context of each alert is displayed in three different layers. In the first and highest-level layer, referred to in this paper as the *alert summary view*, a dashboard displays all the alerts that need to be reviewed. Generally, each alert is contextualized with summary information such as the detected malicious activity, a risk score, a timestamp, and the entities involved in the alert. The second layer, the *event view*, shows the set of events that were aggregated in the same alert. The aggregation or *alert correlation* logic groups related entities and removes duplicates. (Note that when a new event takes place, this logic determines whether an existing alert is updated or a new alert created.) The third and lowest-level view shows the raw data (logs, packets, files, process information etc.) that triggered the detection mechanism for each event included in the alert. This last view is referred to as *raw data view*.

5.2 Incorporating ML-generated alerts

We now describe the challenges and strategies involved with generating the analyst views described above – alert summary view, event set view, and raw data view – when spurred by a detection by machine learning models.

5.2.1 Alert summary view:

When machine learning models are added to the existing set of detection strategies (signatures, blacklists, and rules), the generation of this view can remain for the most part unchanged. Given that ML models evaluate the activity of an entity over a predefined time window, it is straightforward to retrieve the entity information and the timestamp of any given detection.

In a case where supervised models are looking for repeating malicious activity patterns, the identified threat is given by the class predicted by the classifier. Score are computed as a function of the class probabilities returned by the classifier that triggered the alert, among other parameters.

The case of anomaly detection models is arguably more complex. The score is generally a metric that indicates how much an activity deviates from the norm. However, given that high scores indicate *rare activities*, it is harder to determine the threat represented by an activity that is rarely seen. This limitation arguably introduces a challenge in the operationalization of anomaly detection-based alerts, given that analysts are not informed of the specific threats identified in the analysis.

5.2.2 Event view:

This view is the result of the alert correlation logic, i.e. the logic that groups events generated by the different detection mechanisms. Given that this grouping is often performed based on the entities involved in the events and the time of the activities [21], the addition of ML-generated events into the logic does not introduce additional complexities.

On the other hand, this view shows the mechanism that triggered the alert. For instance, Figure 3 shows the events aggregated into *Alert 22*, where *Event 2* was triggered by a Login rule with id 114. One of the benefits of rule-based detection is that rules are interpretable, such that analysts understand the activity pattern identified by each rule. Achieving this with machine learning models is more complex. In the following section, we describe a series of alternatives that might allow for a similar view with machine learning models. Note that we adopt the classification of the approaches and terminology introduced in the survey of methods for explaining supervised models contributed by Guidotti *et al.* [11].

1. Use of interpretable models: Also referred to as *transparent box design*, this is the option to learn interpretable models, such as rules, linear models or decision trees. In this way, a human-readable representation of the models can be incorporated in place of the rules or signatures. In the case of decision trees and linear models, a second option consists of showing a summarized information of the *feature importance* instead of showing the full models.

2. Black-box explanation: There are two principal approaches to explaining black-box models (tree ensembles, support vector machines, and neural networks). The first consists of providing a summary of the *importance* of each of the features. For instance, this capability is discussed in the seminal paper introducing random forests [4], while works such as [35] are model-agnostic.

The second approach consists of showing, rather than the black-box model, an interpretable surrogate model (rules, decision trees, or linear models) that mimics the outputs of the black box. This approximation can be *global*, whereby a surrogate model is used for all the examples in the data, or *local*, whereby each region of the data is explained with a different surrogate model. This way, the user is still presented with an interpretable model. While initial works considered global approximation strategies, the lack of *fidelity* of the surrogate models for complex applications resulted in inconsistencies in the explanations. Recent works show improved results by focusing on local approximations [15, 32].

3. Outcome explanation: Similar to model explanations, there are two main methods for providing outcome explanations. The first relies on the analysis of the contribution of each feature to a given prediction. Note again that some specific models, such as random forests, incorporate this capability out of the box [4], while model agnostic methods have been developed to provide this information [26, 27, 32] for any model that generates output probabilities. For cases in which a surrogate model strategy is adopted, outcome explanation is achieved by showing the specific rule or decision path in a decision tree that triggered the prediction.

One of the shortcomings of these approaches when applied to cybersecurity is that they use the features associated with machine learning analysis as a basis to explain the models or the model outcomes. However, features created by data scientists are not necessarily interpretable by security analysts. Therefore, additional care is needed to ensure the use of interpretable features to generate explanations, and it is paramount to complement each machine learning alert with its corresponding raw data.

5.2.3 *Retrieving relevant raw data:*

To populate the third and lowest-level analyst view, it is necessary to retrieve the raw data corresponding to a machine learning alert. In cases where the alerted entity presents a reduced data footprint, this step is straightforward. In fact, it is enough to retrieve all the logs pertaining to the analyzed entity, and to rely on the analyst to triage the relevant information for the investigation.

However, in practice, the footprint of alerted entities is often large and manual triage is not efficient. For instance, the network activity of a given endpoint can be captured in thousands of log lines in firewall and DNS logs. In these common cases, in order to improve the efficiency of the investigation, it is necessary to reduce the set of logs displayed to as context for the analyzed alert. A simple option consists of designing predefined views of the raw data, generally using a hard-coded time window prior to the alert, and filtering logic to display a reduced set of selected logs, events (rows), and fields (columns). However, this approach can prove insufficient when the alert corresponds to an activity observed for the first time or to variations of existing attacks. In fact, in these cases, a hard-coded, predefined contextualization logic might fail to retrieve the raw data needed to investigate the alert.

Although they are more often used for text, images, and tabular data, certain methods in the explainable machine learning literature may apply here. In particular, those methods that retrieve the discriminant regions of the input data that determine the model outcome. These regions are referred to as *saliency masks* in the context of natural language processing and computer vision [17, 28, 34, 36], and are commonly retrieved via *sensitivity analysis* [5], whereby perturbations to the input data and their resulting model outcomes are jointly analyzed to determine the contribution of each input region to the model outcome.

```
Name                             Example 1   Example 2
------------------------------------------------------
date                             9/7/2017    9/8/2017
source_ip                        10.0.0.4    10.0.0.5
Num. NX domains                  16          14
Avg. NX domain length            22.13       21.07
Avg. NX domain digit ratio       0.25        0.17
Avg. NX domain consonant ratio   0.56        0.61
Avg. NX domain vowel ratio       0.12        0.15
Avg. NX domain entropy           3.99        3.97


Example 1: 9/7/2017

timestamp   source_ip   domain                              response_code
1:56:55     10.0.0.4    1k1r8daphzygb.com                   NXDOMAIN
1:57:01     10.0.0.4    6xzxrlq0sv218o4.com                 NXDOMAIN
1:57:04     10.0.0.4    d19pp1v-l-e1senafnh5pz.com          NXDOMAIN
7:59:51     10.0.0.4    7-hxc3u9urnjnf.com                  NXDOMAIN
10:05:05    10.0.0.4    sdhtbtlf1q.com                      NXDOMAIN
10:05:12    10.0.0.4    xj316pmqvj.com                      NXDOMAIN
10:05:12    10.0.0.4    2syufo8vs0jprllr1a6lmd8q32.com      NXDOMAIN
10:07:44    10.0.0.4    td-k3nctm5-byh4knfd7u.com           NXDOMAIN
10:07:47    10.0.0.4    yb7561d7lw40yq35n-d3.com            NXDOMAIN
10:08:01    10.0.0.4    30x3l0c-8.com                       NXDOMAIN
10:08:30    10.0.0.4    sie-tsuw57l2m.com                   NXDOMAIN
10:08:30    10.0.0.4    svczpy14t3.com                      NXDOMAIN
10:08:33    10.0.0.4    3tmg1apkz4n6pz3074m.com             NXDOMAIN
10:10:04    10.0.0.4    v6dutthgl9aq1tezqric3llp2rm.com     NXDOMAIN
10:10:14    10.0.0.4    t7hoyj6088be546f96h7s.com           NXDOMAIN
10:17:15    10.0.0.4    q9eyeqxnudngw06urnp2f2t2kt.com      NXDOMAIN
12:16:40    10.0.0.4    nu8zm4v2es8lbypbhg-g9nt6kof.com     NXDOMAIN

Example 2: 9/8/2017

timestamp   source_ip   domain                              response_code
1:09:21     10.0.0.5    s6qlhjacym8klak6qct7wf.com          NXDOMAIN
1:12:07     10.0.0.5    mezm62dw2s6.com                     NXDOMAIN
2:15:26     10.0.0.5    bv6wyeyfinr5cp71wra2.com            NXDOMAIN
3:42:26     10.0.0.5    dbqfeyvvjs.com                      NXDOMAIN
9:09:13     10.0.0.5    lnmymb6kje7fs.com                   NXDOMAIN
9:09:13     10.0.0.5    b1ld1gh0wa08e3o6kjy.com             NXDOMAIN
9:09:16     10.0.0.5    foi4jctjjuvidphz92.com              NXDOMAIN
9:09:54     10.0.0.5    hp3k9p0u4qrlz8.com                  NXDOMAIN
9:49:14     10.0.0.5    9l-qsckvr7tlxgwv38imph9bt.com       NXDOMAIN
9:51:11     10.0.0.5    8zmr-ksvlohihcvnx2.com              NXDOMAIN
9:51:11     10.0.0.5    rz1vxe75i2lq694.com                 NXDOMAIN
9:51:36     10.0.0.5    qegxtcgpuyyyqupngj7z.com            NXDOMAIN
9:52:42     10.0.0.5    rw8i-ikw.com                        NXDOMAIN
10:00:31    10.0.0.5    g--l3q9czusfvu-sdp0yevpk2p.com      NXDOMAIN
```

Figure 4: Contextualization of DGA detections with the features used for machine learning analysis and with relevant raw logs.

In the authors' experience, retrieving relevant logs is important for providing actionable ML-generated alerts, because features used for machine learning analysis are not always easy for security analysts to interpret. However, based on our review of related work, most papers in the machine learning literature focus on problems in which data takes the form of images, text, or tables, and do not consider the timestamped activity logs of relational or transactional data that are more relevant to cybersecurity use cases. While the underlying ideas behind these works represent promising approaches to retrieving relevant raw data in the cybersecurity applications, the lack of domain-specific demonstrations arguably introduces a challenge for the integration of machine learning alerts in security operations.

To illustrate this challenge, we next explore the detection of Domain Generation Algorithms (DGAs), a well-documented command and control mechanism [24]. DGAs are designed by attackers to generate dynamic domains that are then used as meeting points for compromised machines and remote attackers. The key idea is that the remote attacker and the compromised machine share the generation logic, and will therefore generate the same set of domains, each on its own side. The remote attacker then registers one of the many domains generated by the algorithm, while the malware in the compromised machines attempts to connect to each and every one of the generated domains until a communication channel is established with the remote attacker.

When such an attack takes place, compromised machines try to connect to an elevated number of random looking domains, as shown in Figure 4. In doing so, they generate DNS queries to resolve the domains, most of which fail because the domains do not exist (NXDOMAIN response code). In order to detect this type of attack with machine learning, we analyze the activity of internal IPs (source_ip in the figure) captured in DNS logs. In particular, we propose using the following features that quantify the characteristics of the NXDOMAINS queried by each IP:

1. Number of NX domains queried by the IP
2. Average length of the NX domains queried by the IP
3. Average digit ratio of the NX domains
4. Average consonant of the NX domains
5. Average vowel ratio of the NX domains
6. Average entropy of the NX domains

Figure 4, shows, for two examples, the values of the extracted features as well as the section of DNS logs used for the analysis. We argue that, in this case, the logs provide information that is more helpful and intuitive to analysts who are trying to determine the presence or absence of a DGA. As discussed above, the automated retrieval of the appropriate logs remains an open challenge for the machine learning and cybersecurity communities.

6. ROLES AND WORKFLOWS

In this section we briefly describe current security operation workflows, along with the adjustments needed to incorporate and maintain machine learning-based detection models.

6.1 Existing roles and workflows

Security operation centers (SOCs) split their analysts into three tiers. The first tier is comprised of the juniormost analysts, while the most skilled analysts are in tier three. First-tier analysts perform a first-pass screening and filtering of alerts in order to discard false positives and remediate common threats. For example, a common tier-1 workflow involves matching alerts against blacklists of known malicious entities, and adding rules to block further connections to the matched entities. In cases where the first tier analysts cannot conclude whether an event is malicious, they escalate the event for further investigation by a higher-level analyst. Tier-2 analysts perform investigations and remediations of escalated alerts, and can in turn pass on events to tier-3 analysts. Generally, only critical alerts requiring advanced investigation skills, such as reverse-engineering malware, are escalated to the last layer.

Apart from these layered investigation workflows, SOCs adopt continuous improvement strategies to maintain appropriate detection rates as the threat landscape evolves, managing the mechanisms (rules, threat intelligence sources etc.) that generate alerts so that mechanisms with inadequate detection rates are either updated or switched off. Update procedures include subscription-based feeds, as well as in-house continuous improvement processes developed by the operations team. For example, an outdated rule that generates an elevated number of false positives can be edited or deleted. Similarly, new detection mechanisms are constantly added as information about new attacks becomes available.

6.2 Emerging roles and workflows in AI security analytics

As described in previous sections, existing mechanisms for generating alerts do not predominantly rely on machine learning models. Consequently, current SOC responsibilities focus on alert investigations and remediations, but do not include some of the key roles and workflows necessary to operationalize machine learning models. As shown in Figure 5, ML workflows introduce three distinct roles: security analysts, model developers, and machine learning architects.

6.2.1 *AI security analytics workflows:*

As shown in Figure 5, AI security analytics operations interact with data at different levels. In a nutshell, the initial steps of big data analytics involve characterizing the activities of active entities across a variety of data sources. Next, the deployed models generate alerts that are integrated into the dashboard for analyst consumption. As such, analysts review ML-generated alerts in the same way they review alerts generated by other mechanisms such as rules, signatures, and blacklists. One of the key aspects of this process is that the investigation outcomes provided by analysts are stored together with the alert contexts and associated data. These annotated alerts are leveraged by machine learning model developers to create new models and improve existing ones. Finally, an AI security analytics architect decides on the model deployment strategy and identifies detection gaps. In the following, we describe in more detail the work carried out by the different AI security analytics actors.

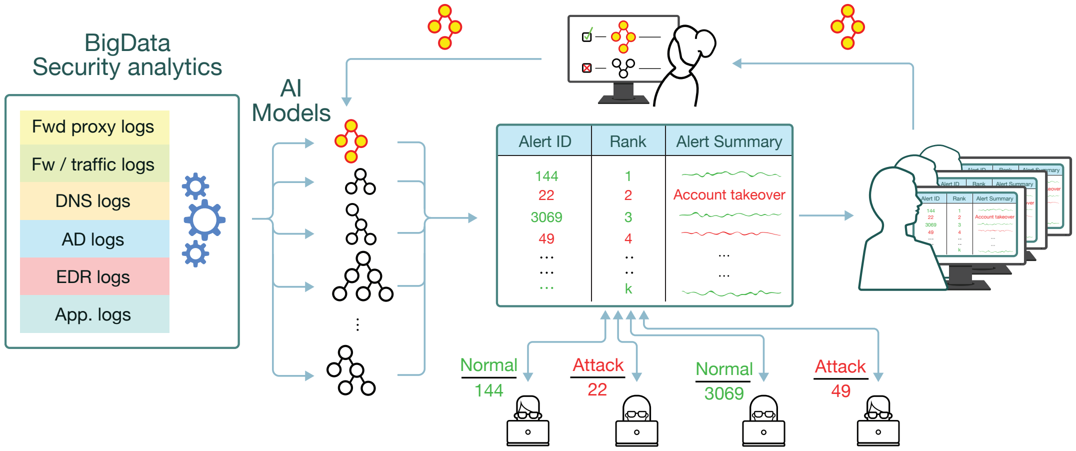
6.2.2 *Security analysts:*

The key responsibility of security analysts is still the investigation and remediation of alerts. Despite the challenges described in Section 5, the investigation and alert escalation workflows followed by security analysts are not highly impacted by the inclusion of ML-generated alerts.

6.2.3 *Machine learning model developers:*

Model developers are charged with the creation and continuous improvement of detection models. New models are created to detect new attacks, expanding the detection coverage. On the other hand, improvements to existing models target a better trade-off between true positives (malicious findings) and false positives (false alerts). Continuous improvement workflows are needed to counter model degradation caused by an evolving threat landscape.

In the authors' experience, the shortage of skilled model developers for cybersecurity represents one of the key challenges in the adoption of AI security analytics. Working in this area requires both an extensive knowledge of the cybersecurity domain, in order to understand attacks and how they manifest in data, along with an understanding of the



Alert ID	Rank	Alert Summary
144	1	
22	2	Account takeover
3069	3	
49	4	
…	..	...
…	..	...
…	k	

Figure 5: **AI security analytics operations:** Big data analytics characterize the activities observed from a variety of data sources. Next, the deployed machine learning models generate alerts that are integrated in the alert dashboard for analyst consumption. Processed data, in combination with investigation outcomes, are leveraged by model developers to create and improve models. Finally, an AI security analytics architect defines the deployment strategy.

strengths and limitations of machine learning models. While frameworks such as [30] describe a large number of attack techniques, developers need to identify opportunities where machine learning models can improve the detection capabilities of existing detection mechanisms.

6.2.4 *AI security analytics architect:*

The responsibilities involved in this role are twofold. AI security analytics architects are in charge of devising the machine learning deployment strategy, deciding which models to deploy in order to obtain an appropriate threat coverage and an acceptable cost (false positives) versus benefit (true positives) trade-off. They must also identify gaps in detection coverage due to a lack of models or degraded model performance. These findings are then translated into requirements for the development or improvement of models.

7. CONCLUSION

In this paper, we have deeply explored the challenges that currently limit the adoption of machine learning for threat detection in real-world security operations. The scope of these challenges is wide, and the solutions require not only technological developments, but also changes in the roles and processes involved in security operations. This paper has focused on the label acquisition challenge, the processing capabilities needed to address the many-to-many mappings of data sources to the wide range of threats that compose the cybersecurity domain, and the challenges involved in adding the alerts generated by machine learning models into existing investigation workflows. Finally, the paper discussed the operational changes in processes and roles needed to maintain a machine learning-based detection strategy over time.

8. REFERENCES

[1] B. Anderson and D. McGrew. Identifying Encrypted Malware Traffic with Contextual Flow Data. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, AISec '16, pages 35–46, New York, NY, USA, 2016. ACM.

[2] P. Awais Rashid, R. Ramdhany, M. Edwards, S. Mukisa Kibirige, M. Ali Babar, D. Hutchison, and R. Chitchyan. Detecting and Preventing Data Exfiltration. Technical report, Lancaster University, 04 2014.

[3] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani. Towards effective feature selection in machine learning-based botnet detection approaches. In *2014 IEEE Conference on Communications and Network Security*, pages 247–255, 2014.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[5] P. Cortez and M. J. Embrechts. Opening black box Data Mining models using Sensitivity Analysis. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 341–348, April 2011.

[6] M. Darling, G. Heileman, G. Gressel, A. Ashok, and P. Poornachandran. A lexical approach for classifying malicious URLs. In *2015 International Conference on High Performance Computing Simulation (HPCS)*, pages 195–202, July 2015.

[7] J. J. Davis and E. Foo. Automated Feature Engineering for HTTP Tunnel Detection. *Comput. Secur.*, 59(C):166–185, June 2016.

[8] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani. Characterization of Encrypted and VPN Traffic using Time-related Features. In *ICISSP*, 2016.

[9] G. Farnham and A. Atlasis. Detecting DNS Tunneling. SANS Institute InfoSec Reading Room, 2013.

[10] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and Robust Automated Machine Learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc., 2015.

[11] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv.*, 51(5):93:1–93:42, Aug. 2018.

[12] A. Hagberg, N. Lemons, A. Kent, and J. Neil. Connected Components and Credential Hopping in Authentication Graphs. In *2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems*, pages 416–423, Nov 2014.

[13] H. Jiang, J. Nagra, and P. Ahammad. SoK: Applying Machine Learning in Security-A Survey. *arXiv preprint arXiv:1611.03186*, 2016.

[14] A. Kapravelos, Y. Shoshitaishvili, M. Cova, C. Kruegel, and G. Vigna. Revolver: An Automated Approach to the Detection of Evasive Web-based Malware. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 637–652, Washington, D.C., 2013.

[15] S. Krishnan and E. Wu. PALM: Machine Learning Explanations For Iterative Debugging. In *Proceedings of the 2Nd Workshop on Human-In-the-Loop Data Analytics*, pages 4:1–4:6, New York, NY, USA, 2017. ACM.

[16] H. Lampesberger, P. Winter, M. Zeilinger, and E. Hermann. An on-line learning statistical model to detect malicious web requests. In *International Conference on Security and Privacy in Communication Systems*, pages 19–38. Springer, 2011.

[17] W. Landecker, M. D. Thomure, L. M. A. Bettencourt, M. Mitchell, G. T. Kenyon, and S. P. Brumby. Interpreting individual classifications of hierarchical networks. In *2013 IEEE Symposium on Computational Intelligence and Data Mining*, pages 32–38, April 2013.

[18] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani. Characterization of Tor Traffic using Time based Features. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, pages 253–262. INSTICC, SciTePress, 2017.

[19] Y. Liu, C. Corbett, K. Chiang, R. Archibald, B. Mukherjee, and D. Ghosal. SIDD: A Framework for Detecting Sensitive Data Exfiltration by an Insider Attack. In *2009 42nd Hawaii International Conference on System Sciences*, pages 1–10, Jan 2009.

[20] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani. Detecting Malicious URLs Using Lexical Analysis. In *Network and System Security - 10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings*, pages 467–482, 2016.

[21] S. A. Mirheidari, S. Arshad, and R. Jalili. Alert correlation algorithms: A survey and taxonomy. In G. Wang, I. Ray, D. Feng, and M. Rajarajan, editors, *Cyberspace Safety and Security*, pages 183–197, Cham, 2013. Springer International Publishing.

[22] MLflow. An open source platform for the machine learning lifecycle, 2019.

[23] A. Nadler, A. Aminov, and A. Shabtai. Detection of Malicious and Low Throughput Data Exfiltration Over the DNS Protocol. *CoRR*, abs/1709.08395, 2017.

[24] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla. A Comprehensive Measurement Study of Domain Generating Malware. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 263–278, Austin, TX, 2016. USENIX Association.

[25] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: rapid training data creation with weak supervision. *The VLDB Journal*, Jul 2019.

[26] 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*, KDD '16, pages 1135–1144, New York, NY, USA, 2016. ACM.

[27] M. Robnik-ikonja and I. Kononenko. Explaining Classifications For Individual Instances. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):589–600, May 2008.

[28] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[29] M. Stevanovic and J. M. Pedersen. On the use of machine learning for identifying botnet network traffic. *Journal of Cyber Security and Mobility*, 4(3):1–32, 2016.

[30] The MITRE Corporation. ATT&CK: Adversarial Tactics, Techniques & Common Knowledge, 2019.

[31] I. Ullah. Detecting Lateral Movement Attacks through SMB using BRO. Master's thesis, University of Twente, 2016.

[32] M. M. Vidovic, N. Görnitz, K. Müller, and M. Kloft. Feature importance measure for non-linear learning algorithms. *CoRR*, abs/1611.07567, 2016.

[33] M. Warmer. Detection of web based command & control channels. Master's thesis, University of Twente, 2011.

[34] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

[35] A. Zien, N. Krämer, S. Sonnenburg, and G. Rätsch. The feature importance ranking measure. In W. Buntine, M. Grobelnik, D. Mladenić, and J. Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 694–709, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[36] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling. Visualizing deep neural network decisions: Prediction difference analysis. *CoRR*, abs/1702.04595, 2017.