Haystax TECHNOLOGY
Security Analytics **Redefined**

Three Security Analytics Approaches That (Mostly) Don't Work

How to Properly Apply Bayesian Networks, Machine Learning and Rules-Based Systems to Meet Today's Security Challenges

Introduction

Digital technologies have changed the face of business and government, and will continue to do so at an even faster pace. They drive innovation, boost productivity, improve communications and generate competitive advantage, among other benefits.

However, there is a dark side to this digital revolution that has now come clearly into focus. McKinsey estimates that cyber-attacks will cost the global economy $3 trillion in lost productivity and growth by 2020, while theft, sabotage and other damage inflicted by trusted personnel continue to cost organizations in lost revenues, revealed secrets and damaged reputations.

The field known as 'Security Analytics' is widely seen as offering solutions that will spare beleaguered organizations the reputational, financial and physical costs of all kinds of threats. But the reality is that today's security products are mostly ineffective, while the threats continue to morph and multiply and the attackers continue to outmaneuver or overwhelm the defenders. Even underlying doctrines and policies are far behind the times.

Security Analytics Approaches That Don't Work

There are three widely deployed analytical approaches that are frequently cited as effective tools in the security analytics community: Bayesian networks, machine learning and rules-based systems. We are intimately familiar with these approaches, because they form the core of our model-based Constellation Analytics Platform™. Unfortunately, we also see numerous implementations where they simply don't produce good results, don't scale or are too hard to work with.

In this white paper we will first explore in detail why each approach is so challenging, and then describe how we've addressed those challenges in our products.

Here's a quick summary of the three approaches.

1. Bayesian probability theory states that it's possible to predict with surprising accuracy the likelihood of something happening (or not happening), in a transparent and analytically defensible way. A *Bayesian inference network*, or model, captures every element of a problem and calculates possible outcomes mathematically. The harder the problem, the better it works – at least in theory. In reality, a typical approach is to gather a roomful of PhDs and spend a lot of time and money building a Bayesian network. Then, with even greater effort and more man-hours, the Bayesian network is turned into software by a roomful of coders. The resulting product is something the user struggles even to understand, let alone use. Not surprisingly, there's an emerging camp that claims Bayesian networks are old-fashioned and not suited to solving today's security challenges – especially now that machine learning is available.
2. *Machine learning*, in Arthur Samuel's classic definition, "gives computers the ability to learn without being explicitly programmed." It can be used, for example, to uncover hidden insights from historical relationships and trends found in data. While that may have excited the original adherents, we've had over 50 years to discover some of the limitations:
    - There are no real, generalizable approaches to machine learning.
    - Correlation isn't all it's cracked up to be in a world of black-swan scenarios and asymmetric threats.
    - Most machine-learning solutions come 'black boxed,' which is highly unpopular among users who have to make critical decisions, and then defend them.
    - Hasn't science taught us to start with a hypothesis? There's no such luxury with machine learning.
3. Much simpler – and more common – than Bayesian networks and machine learning, *rules-based systems* have their own inherent drawbacks. Because they're typically binary, the outputs tend to be too coarse-grained for the often subtle threats they're trying to detect and identify. This leads to a proliferation of red flags, most of them false positives, which then leads to a proliferation of pricey analysts. Try instead to create rules for special cases and you get a proliferation of rules. Paralysis reigns, and the world is still not safer.

The good news is that these three approaches can be harnessed in meaningful ways – as long as they're thoughtfully built, combined and applied.

Bayesian Networks

Bayesian networks are among the more sophisticated capabilities applied to security analytics, relying as they do on decision science and statistics rather than software engineering expertise. But in many implementations, the effort seems to focus on backrooms full of PhDs doing the hard math, rather than on tools that make Bayesian networks accessible to subject-matter experts (SMEs), who are usually not mathematicians. The resulting black-boxed solutions can be hard to comprehend, and often lack a suitable platform and user interface for easy access and operational use by the people who would most benefit from them.

Bayesian networks provide the capability to efficiently represent probabilistic knowledge about a complex problem domain. They then use that knowledge to reason intelligently in that domain under conditions of uncertain, incomplete or even contradictory data.

A major challenge is developing the probabilistic knowledge that defines the Bayesian network for that problem. One way to develop that knowledge is to learn the Bayesian network's structure or parameters from data. This is feasible only when there is a suitably labeled data set available. When no data is available from which to learn, the knowledge in a Bayesian network must be derived from domain knowledge.

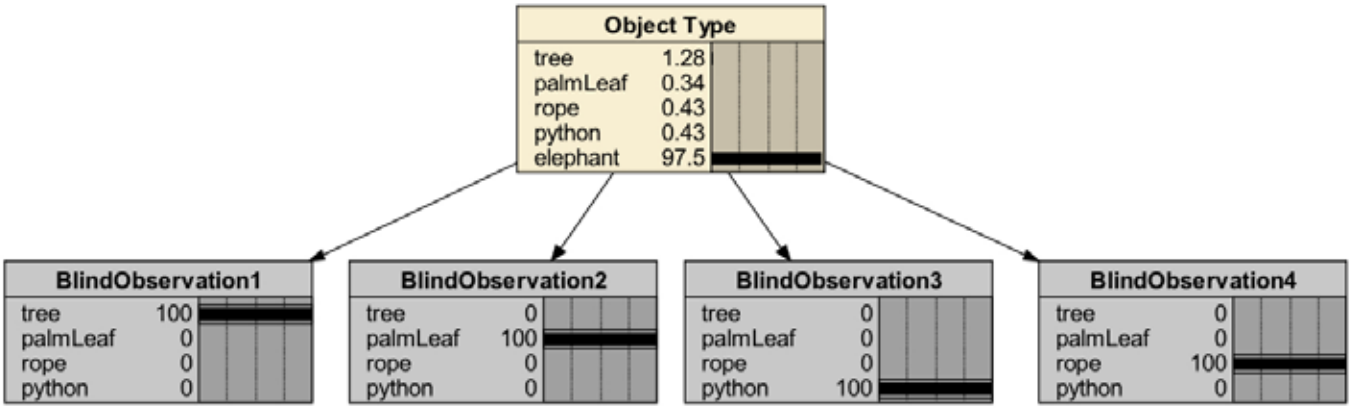
Typically, modeling experts don't have the required domain knowledge and thus must extract the information from an organization's internal experts, policy documents, knowledge bases and/or other sources. This knowledge elicitation is a time-consuming and expensive process, tying up domain experts and delaying implementation of the solution.

At Haystax Technology, we've developed a knowledge representation approach that dramatically simplifies the knowledge elicitation process for a common class of Bayesian networks, and allows SMEs to express domain knowledge in a natural way that does not require extensive interaction with Bayesian modeling experts.

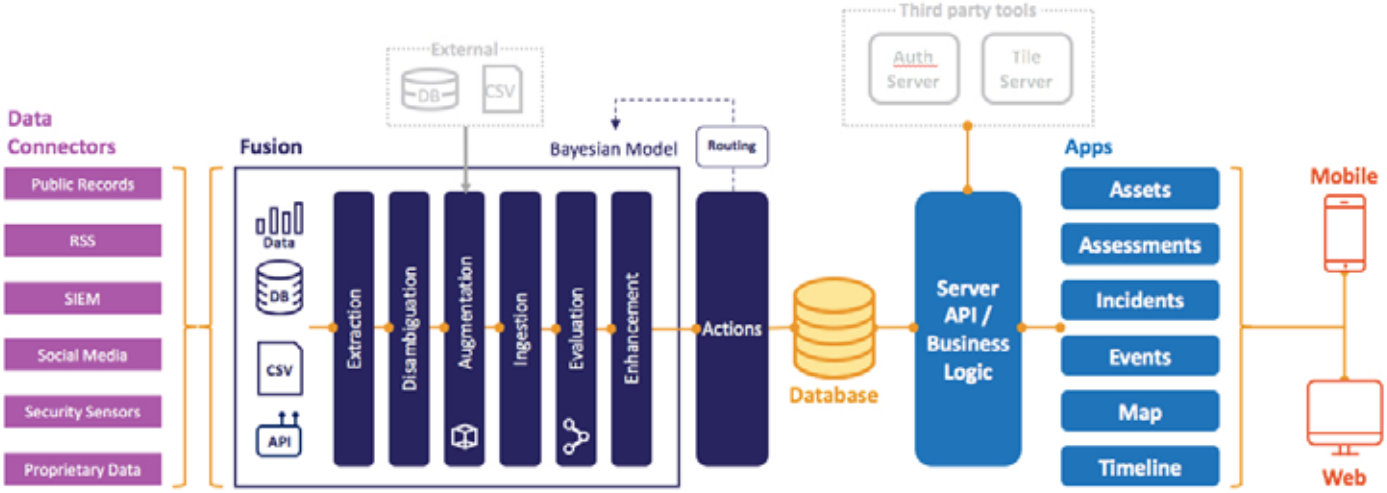
Our approach involves first identifying the important *concepts* in the problem domain, then specifying qualitative relationships between the concepts and finally using software to automatically assemble the qualitative knowledge into a quantitative Bayesian network (see example below, which uses the classic 'blind-men-and-an-elephant' scenario). The important concepts become binary random variables in the Bayesian network, and we then qualitatively define relationships between concepts.

The most common relationship between concepts is *indication* – that is, 'A is an indicator for (or against) B,' or alternatively, 'A is evidence for (or against) B.' A concept can be an indicator for more than one indicated concept. Other relationships include *summary* ('A is a summary of B1, B2, … Bn), *mitigation* ('C mitigates the influence of A on B') and *relevance* ('D makes A relevant to B').

For each of the above, our approach uses a qualitative specification of the strength of the indication (or evidence) and polarity – that is, *positive*: evidence for, or *negative*: evidence against. Once we have a qualitative representation of the knowledge, the knowledge is automatically assembled into a quantitative Bayesian network.



The final step of exploitation of a Bayesian network is a software architecture designed to support reasoning with analytical models, which facilitates extracting evidence from data streams and data sources, applying evidence appropriately to the Bayesian network, performing inference in the Bayesian network and then presenting relevant inference results to the user (see architecture image below).



Haystax Technology's Constellation Analytics Platform™ was built to run Bayesian networks at scale, ingesting, analyzing and processing large amounts of structured and unstructured data – both streaming and in batches – from a wide variety of data sources. Intuitive interfaces and intelligent alerting ensure the results reach the people who need the information, when they need it.

Machine Learning

Machine learning has revolutionized our world. It powers our smart phones, determines which advertisements we view, and soon it will dominate the automobile industry through self-driving cars. Even Google now wants all of its engineers to know at least some machine learning.

But how does machine learning compare to the Bayesian model-first approach taken by Haystax? And are these concepts necessarily at odds with each other?

In practice, the machine learning approach typically looks something like this:

1. *Identify the response variable*: This is the variable to be predicted (e.g., 'banking transaction is compromised, or not'; 'patient is infected with the Zika virus, or not')
2. *Get data*: The more the better.
3. *Select features*: Features are a particular subset of data, or calculations from data, that are hypothesized to be predictive. For instance, if we're predicting whether a news story is about politics or sports, the nouns used in the article (e.g., 'Congress' or 'football') would probably make good features. Words like 'the' or 'a,' on the other hand, would not.
4. *Choose a machine-learning algorithm*: For classification problems (e.g., 'patient has Zika, or not'), examples of classification machine-learning algorithms are: logistic regression, naïve Bayes, neural network and support vector machines. If the response variable is continuous (e.g., prices of houses), algorithms such as a regularized form of linear regression might be appropriate.
5. *Tune hyper-parameters*: In this stage, the machine learning algorithm will generally have parameters that need tuning. These are often selected by a technique such as cross validation.
6. *Test*: Finally, it is important to test the performance of the algorithm. Usually, a technique like k-fold cross validation is used to compute metrics such as error rates, precision, recall, etc.

It is not uncommon to find practitioners trying different types of machine-learning model (e.g., a support vector machine instead of logistic regression) before stumbling upon a model that seems to work best. But, as almost any practitioner will tell you, great features always trump a fancy machine-learning model. And there are dangers in machine learning as well. One is that correlation doesn't necessarily imply causality. In fact, some journals have even taken the position of considering banning p-values altogether, because they claim the test for reliable results was too easy to pass.

It's important to realize, however, that machine learning and model-first approaches don't have to be at odds. As an example, at one stage Google engineers were attempting to solve the problem of recommending the right coffee shops by learning from data. From their initial data set they found that, on the whole, user satisfaction and distance traveled were negatively correlated, which makes sense given that most people do not want to walk for hours to get their caffeine fix. But when they tried using different machine-learning algorithms, they found that the trend lines produced didn't match this obvious answer.

At this point, they decided to try a model-first approach. They started with the hypothesis that the function they wanted to learn from the data set should be monotonically decreasing – i.e., that less distance walked was always better than more, assuming an equal quality of coffee-drinking experience. This very basic model-first approach led to a brand new and vastly superior set of results; Google still learned from the data set, but the calculations were made within the context of the new model.

At Haystax, we've witnessed similar benefits of taking a model-first approach. For instance, we worked on the problem of detecting compromised transactions for a major international bank. It was hypothesized that specific signatures – such as if an unusual HTML tag name was present – would be indicative of malware. Now, of course there are benign explanations for the appearance of an unusual HTML tag. Most of the time it was harmlessly injected by the customer's browser. We exploited this knowledge using our modeling framework, and then used the bank's data to estimate parameters such as the conditional probability that a particular HTML tag name would be observed for a given page URL. As a result, although two different HTML tag names might end up with the same anomalous score, our top-level hypothesis will react differently depending on mitigating concepts.

Christopher Bishop also recommends a type of 'model-first' approach in his paper on model-based machine learning. The idea is simply that in any problem you model you should start with concepts, which can then link to other concepts. If the links are made to have direction, you can determine that one concept causes the other to occur. Simply by constructing this graph, you are using a model-first approach.

Now it may turn out that the topology of the graph is identical to the topology of a hidden Markov model. If this is the case, then by all means, use a hidden Markov algorithm to estimate the parameters. If the model yields poor results, this can actually be exciting because when we revise the model we can review assumptions that are too weak or too strong, or determine whether other concepts are now relevant. And if after revision the results improve, we've now learned more about the underlying process that generates the observed data.

Rules-Based Systems

Rules-based systems use 'if-then' rules to derive actions. For example, if the fact that 'Sally is 22 and unemployed' is matched to the rule 'If a person is between 18 and 65 and is unemployed, they can claim unemployment,' the system would conclude that 'Sally can claim unemployment.'

One advantage of these systems is that they are relatively easy to understand, and can be built to represent expert judgment on simple or complicated subjects. Another is that cause-and-effect are transparent. Even though the 'if-then' reasoning can become complex, a domain expert can verify the rule base and make adjustments where necessary.

That said, there are three critical weaknesses to rules-based systems and, unfortunately, they overwhelm the advantages:

1. Rules engines *do not scale*. They must logically become almost as complicated as the problem the system is trying to solve. Unlike machine learning, rules cannot be learned and must instead be added manually. In practice, a 'rules explosion' occurs where a series of seemingly simple rules becomes a complex net of conflicting rules, overlapping rules and so on. As a result, rules-based systems become very hard to understand in the aggregate, and the more knowledge a user adds by adding more rules, the more complex and opaque the system becomes.
2. These systems *don't handle incomplete or incorrect information* very well, and data that does not have an associated rule will be ignored. This means that rules-based systems are particularly bad at detecting 'unknown unknowns' like new derivations of malware, or new diseases.
3. Rules-based systems *don't know what to do with variables that have an infinite number of possible values*. And unfortunately, there are many of these variables: a person's weight, for example, or the time it takes to complete a task, or the price of a gallon of gas.

Arbitrarily converting these continuous variables into discrete variables means potentially missing patterns or deriving false patterns. For example, a rule flagging someone who's more than 90 days' delinquent on a single debt payment may miss someone else who is chronically delinquent on debt in the last five years, but never more than 89 days late for any one account. Who is riskier?

At Haystax, we don't use a rules-based system as the foundation of our risk analytics models. Nonetheless, we use rules in two ways:

- In all models there are hard rules (e.g., 'you must be a US citizen to obtain a clearance'), which can be implemented as policy or compliance restrictions regardless of the amount of risk they may indicate.
- We use rules *after* the analytics have occurred to sort or highlight information. For example, we use rules to dictate what we'd like to display on our dashboards (see image below), or in our reports.

In general, the Haystax approach is to use rules to reason on *processed* data, not as a way to *process* data.

A Security Analytics Approach That Does Work

To be sure, Bayesian networks, machine learning and rules-based systems are applied successfully in many software systems across many domains, so clearly the techniques themselves are sound. (Machine learning, for example, has been used by Google to recognize objects within an image and automatically create captions for them.) But they don't always work for security analytics, and that is primarily because each technique has weaknesses that have yet to be resolved appropriately for that kind of application. These include:

Specialized knowledge

Security analytics is a complex and technical field, requiring specialized knowledge of risk-management concepts, network systems, log files and analytic techniques. Similarly, Bayesian networks, machine learning and rules-based systems – as well as the tools needed to implement them – are equally complex. Sadly, security analytics personnel possessing all of these skills are both rare and expensive, and without them your team will naturally be limited by what it knows, or by the challenge of translating its needs to the experts available.

Opacity

Machine learning and rules-based systems don't map directly to security analytics problems. Even simple implementations are complex enough to prevent practitioners from understanding solutions based on them. In the case of machine learning, the statistics and math behind each of multiple techniques and the rationale for selecting one technique over another are lost or forgotten once a choice is made. With rules-based systems, the sheer volume of rules creates a cognitive burden that prevents comprehensive understanding. Ultimately, this results in systems that are difficult to grasp, and improve only incrementally over time.

Specialized solutions

As practiced today, all three approaches are bound by the following significant constraints:

- Machine learning is dependent on data and thus is unable to offer solutions in cases where data is scarce or non-existent.
- Rules-based systems either produce far too many false positives or negatives, or require so many rules as to be inapplicable beyond one specific configuration.
- Bayesian network-based solutions, on the other hand, force practitioners to operate at the level of individual assertions and prevent larger and more meaningful networks from being constructed.

Despite their limitations, though, each of these techniques offers unique strengths that would need to be present in an ideal security analytics solution:

- Bayesian networks: Domain conceptual alignment and ability to reason on incomplete data.
- Machine learning: Sheer power and ability to cope with massive quantities of data.
- Rules-based systems: Intuitive simplicity and ease of getting started quickly.

What's needed is a solution that *combines these strengths while also compensating for the individual weaknesses* of each approach. Solutions that provide software libraries or toolboxes for developers fail because they're aimed at the wrong audience. The ideal solution would be aimed at domain users, and would provide simple interfaces for applying these powerful analytic tools to security analytics problems.

At Haystax, we've taken that user-focused approach. In our Constellation Analytics Platform™, domain experts can interact with simple tools using their own domain concepts and knowledge to create Bayesian network-based security analytics solutions that integrate data directly, or transform it via machine learning. Using an easy-to-learn language, both direct rules and probabilistic inference can operate side by side and be integrated into a single model to create easily traceable, transparent and generalizable models for domains of interest. Machine learning and data science techniques can then be applied in simple and separable cases, ensuring that implementation decisions are easy to capture.

Here in more detail, then, is a security analytics solution that *does* work.

Model first . . .

Hundreds of years of experience have taught us that experiments without hypotheses mostly produce the wrong data, and attempting to build understanding from too much of the wrong data will generally yield poor results.

Instead, we've created a simple language that allows domain experts to clearly state what they know to be true, using their own terms and concepts. The results are easily understandable descriptions of a domain at any level of abstraction. These descriptions can be written, shared, stored, version-controlled and mapped to source material or interviews. When compiled by our system, we get a fully formed, functional Bayesian inference network – in simple terms, a model (see image below).

Our modeling approach is successful because we enable analysts to work in their own domain, using their own terminologies and the qualitative statements they make every day. In addition, we provide a number of relationship 'patterns' that capture typical relationship types among concepts.

For example, one concept may indicate another weakly, strongly or even absolutely (water falling from the sky, for instance, is a strong indicator of rain). Other pattern relationships include inverse, mitigation and summaries.

Rules, such as those found in a rules-based system, can be expressed as absolute indicators on model nodes. In this manner, we merge the techniques of Bayesian inference networks and rules-based systems into a single coherent framework for expressing meaning from data.

Our approach is more effective than rules-based systems because we can intermingle rules-based inference with more nuanced probabilistic inference in the same model – perhaps even for the same concept.

... Then go get the data

Creating a model first provides a structure for identifying the data needed to drive the model. This is a valuable step because it enables us to consider the data that's needed and to ignore the data that is unlikely to be useful. Only after creating a model do we consider the specific data we might have on hand, and how we can apply it to our security analytics problem.

Data comes in many formats, at different velocities and in widely varying volumes. In any problem, how that data gets applied to the model is a key part of developing a successful solution. Some data, such as personal information about an individual (see image below), changes infrequently, and can be used directly in the model. Where a person lives, for example, when considered with salary records, can be an indicator of excessive wealth.

Other data, such as network user activity, changes frequently but is easily accessible. Such data can be highly indicative of malicious activity, but equally it may only be useful when considered against historical activity of the same type. Accessing a network or

downloading files, for example, is usually normal activity, but when it happens outside business hours or at a time that is atypical for a given person it could be an indicator of something more malicious.

In the Constellation Analytics Platform™, we provide software tools for describing an ontology of the things that can exist, and how data sources map to create instances of them. This ontology forms the foundation of the solution, enabling data sources to map into the security analytics problem domain and enabling the analytic model to make inferences on ontology instances. The Constellation platform orchestrates all of these actions, enabling the system to operate transparently, consistently and reliably.

Data-source mappings can be direct (e.g., age, birth date or an employment event) or complex in cases where data science or other techniques are required to map the data. Such mappings can be expressed as simple configuration files or source code, as desired, and thus can be arbitrarily complex. Data-source mappings result in instantiations of ontology objects that the system creates and persists over time.

When needed, machine-learning and data-science techniques can be used in data-source mappings to capture specific indicators or to extract indicators based on learned features in data. In the Constellation platform, machine-learning techniques applied to data tend to repeat existing patterns, simplifying their implementation because there are examples to look at.

Applying new data to the system via the creation of ontology objects triggers evaluation of one or more Bayesian inference models. During evaluation, ontology objects map to specific indications in the inference model(s) and are applied considering all aspects of their definition. In this way, two similar events that occur at different points in time can result in dramatically different interpretations by the model.

Typical Constellation security analytics ontologies are developed around a core object such as a person, building or computer system. Around that object, associated objects (e.g., events for a person or computer system) can be indicated by the various data sources and created as part of the mappings defined for the data source. No specific considerations or concessions need to be made during model implementation to accommodate scaling, so the best mapping of data to model via the ontology can always be used. With Constellation, even novice implementers are able to create solutions that operate at Internet scale.

A better analytical outcome

Our system offers a radically different approach to security analytics, one that allows Constellation users to gain the benefits of three key security analytics techniques *without* their drawbacks. Constellation combines Bayesian inference networks, machine learning and rules-based systems in a platform that provides the best features of each, with technical innovations that eliminate the weaknesses of more conventional implementations.