# Defending Networks with Incomplete Information: A Machine Learning Approach

## Introduction and Abstract

Let's face it: we may win some battles, but we are losing the war pretty badly. Regardless of the advances in malware and targeted attacks detection technologies, our top security practitioners can only do so much in a 24 hour day. Even less, if you consider they need to eat and sleep. On the other hand, there is a severe shortage of capable people to do "simple" security monitoring effectively, let alone complex incident detection and response.

Enter the use of Machine Learning as a way to automatically prioritize and classify potential events and attacks as something can could potentially be blocked automatically, is clearly benign, or is really worth the time of your analyst.

In this Whitepaper we will present publicly for the first time an actual implementation of those concepts, in the form of a free-to-use web service. It leverages OSINT sources and knowledge about the spatial distribution of the Internet to generate a fluid and constantly updated classifier that pinpoints areas of interest on submitted network traffic and security tool log sources.

## Whitepaper Topics:

## Security Monitoring: We are doing it wrong

The amount of security log data that is being accumulated today, be it for compliance or for incident response reasons, is bigger than ever. Given the push on regulations such as PCI and HIPAA, even smallish and medium companies have a large quantity of machine-generated data stored in log management solutions no one is currently looking at.

There is a clear a surplus of data and a shortage of professionals that are capable of analyzing this data and making sense of it. This is one of the main criticisms that compliance and "check-box security" practices receive, and, of course, it is legitimate criticism because no one is safer for just accumulating this data.

Even when organizations have more advanced log management and SIEM solutions, there is a great difficulty in prioritizing what should be investigated as security events happen. These tools' functionality relies too deeply on very deterministic rules: if *something* happens in my network *X* amount of times, flag this as suspicious and or send me an alert. The problems arise from the fact that the *somethings* and the *X*s vary widely between organizations and also evolve (or devolve) over time in an organization itself. It is truly a Sisyphean effort.

But this is not exclusively a tool problem. There are a few really talented and experienced professionals that are able to configure one of these systems to perform. However, it usually takes a number of months or years and a couple of these SOC "supermen" working full-time to make this happen. And now we are adding Big Data to SIEM solutions? What chance do we have of finding a needle on a 1,000 times larger haystack?

But how many of these mythical and magical people exist? The new "security analyst" must now understand not only the intricacy of attack and defense but also be an accomplished "data analyst" or "data scientist" in order to work through massive amounts of data. I am not sure where they are, but I can assure you they are not working 24x7, night or weekend shifts on your local or subcontracted monitoring team.

We are doing it wrong.

This project introduces the idea of using Machine Learning techniques to mine information like this and help companies make informed decisions based on this treasure trove of information they have available.

The initial algorithms presented on this paper, by itself, may not yet outperform a (very well) trained analyst but:

- it's certainly better than no action at all;
- it can greatly enhance the analyst's productivity and effectiveness by letting him focus on the small percentage of data that is more likely to be meaningful.

## Machine Learning and the Robot Uprising

Machine learning is designed to infer relationships from large volumes of data. In a way, you do not design the classifications routines or the specific algorithm you are using, but let the data itself shape how the program should behave. The terms *learning* and *artificial intelligence* are usually associated with these kinds of routines because these processes try to mimic the way the brain learns by association and repetition. Once you have seen enough chairs, you will know when you see a completely different chair even if it has a different shape, color or is made of a different material.

Machine learning based systems are being used all around us:

- In sales and marketing, to serve us online ads, to suggest us products that are similar to the ones we or our friends have bought;
- In financial systems, fueling high-frequency trading applications looking for patterns in penny stocks or opportunities for arbitrage;
- In image processing, for example to convert an image of a text into a computer document (OCR).

Alas, they are not perfect: we always see an online ad that does not make sense ("what am I going to do with a carbon-fiber pogo stick?") or the market crashes on an algorithm bug, but the have a very good performance on problems that are otherwise intractable due to dataset sizes or response time necessary.

There is a huge number of algorithms, techniques, sub-groups and quasi-religious belief systems associated with Machine Learning, but for all intents and purposes we will limit our exploration to what are called "supervised learning" and "unsupervised learning" problems. The names are what they seem:

- In Supervised Learning, you are telling the algorithm what to expect from the training data. In other words, you need to tell it what results it should aim for when it sees similar data points. This is the basis of neural networks and other predictive algorithms;

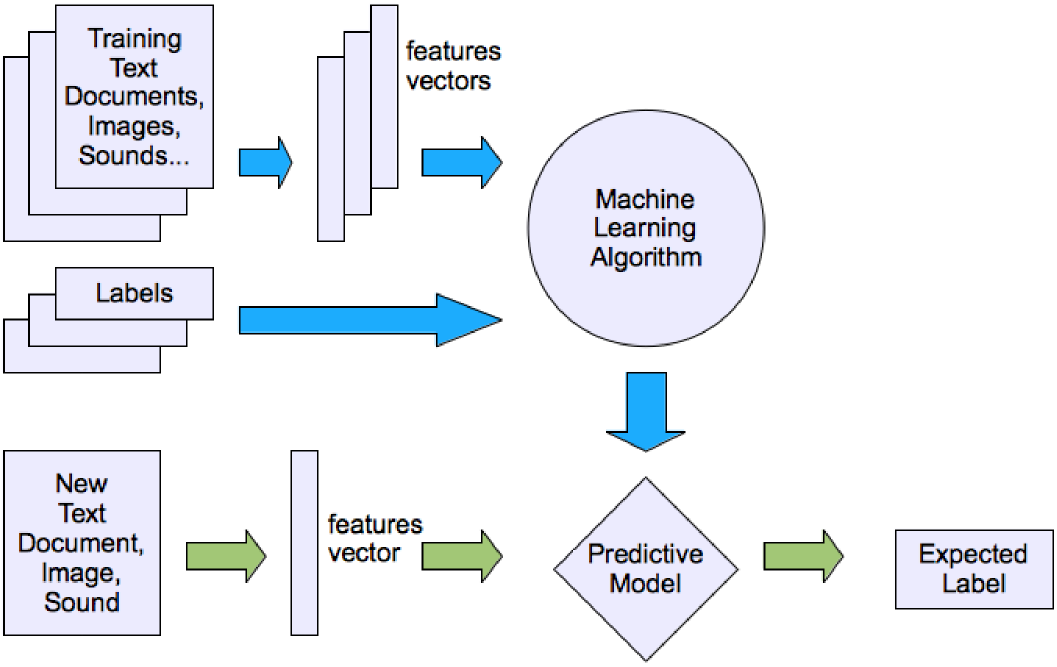
Training Text Documents, Images, Sounds...
features vectors
Machine Learning Algorithm
Labels
New Text Document, Image, Sound
features vector
Predictive Model
Expected Label

**Figure 1 - Supervised Learning (source:scikit-learn.github.io/scikit-learn-tutorial/general_concepts.html)**

- In Unsupervised Learning, the algorithm just receives the data and tries to infer relationships like similarity and proximity from the observations. This is more commonly used to either support anomaly detection (this data point is not like the others). It is also used to help figure out what is really relevant on the data set, on a process that is called "feature extraction".

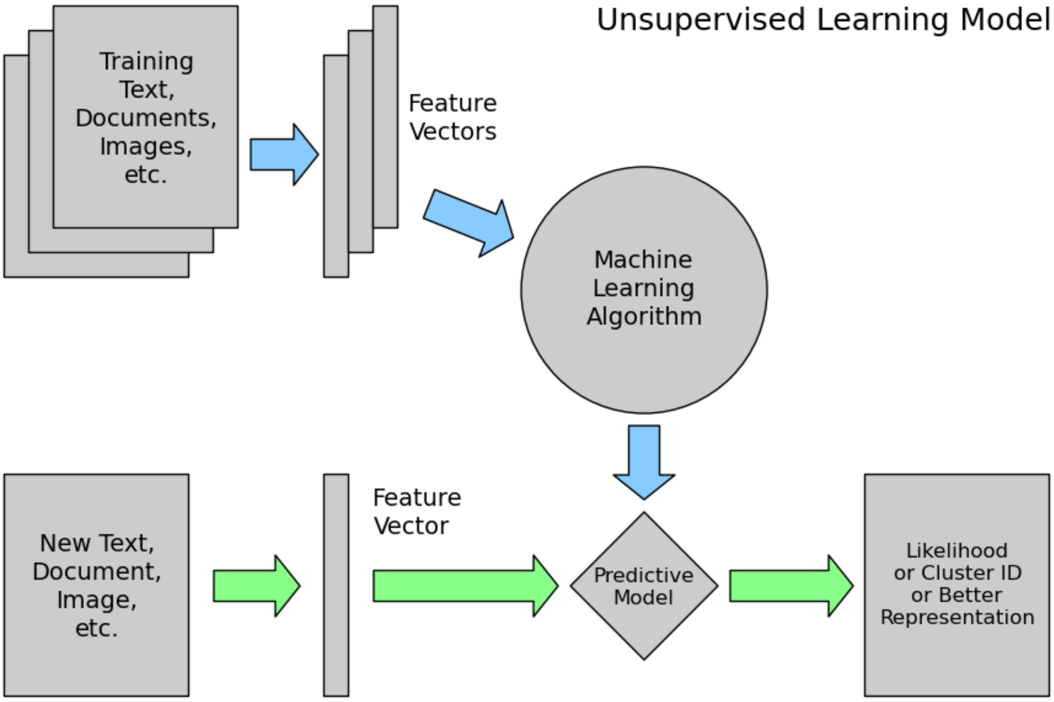
Unsupervised Learning Model
Training Text, Documents, Images, etc.
Feature Vectors
Machine Learning Algorithm
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Likelihood or Cluster ID or Better Representation

**Figure 2 - Unsupervised learning (source: scikit-learn.github.io/scikit-learn-tutorial/general_concepts.html)**

The thesis is that we can apply machine learning to parameterize the *something*s and the *X*s I mentioned before with very little effort on the human side. We could use Unsupervised Learning to find patterns in data that could generate new rules and relationships between occurrences in our networks. Or we could use Supervised Learning with the humans providing examples of "good" and "bad" behavior in their networks, so that the algorithm can then sift through gargantuan amounts of data and suggest other instances that are likely to be relevant in the same way.

There is actually a very good use case for Machine Learning in Information Security, which are spam filters. Nobody really talks about spam filters anymore, because, in a way, the problem has been satisfactory "solved", or actually reached an evolutional plateau. You still can't defeat actors that exploit the algorithm directly, such as sending a specially crafted phishing attack that looks like a regular e-mail. This is a more complicated problem, involving game theory and rational adversarial modeling [1].

Regardless, this is a neglected tool for Information Security by and large, and that is an odd choice, because it seems to be a good fit for the challenges we face day after day.

## More attacks = more data = better defenses

The great advantage of Machine Learning over more traditional approaches for Security Monitoring is that the predictive capabilities of the algorithm improve as you provide more data to its training processes. As your number of observations gets much larger than the complexity of the model you are trying to predict (usually referred in ML terms as dimensionality or VC-dimension), it gets very efficient indeed.

Let that sink in for a moment. The more you are attacked, the better your defenses can be. These categories of things have been deemed "Antifragile"[2], and they benefit from disorder and chaos. And disorder and chaos sound a lot like where we are today in Information Security monitoring.

This is a generalization, of course. The model that you choose to be trained, the specific math of the algorithm you choose all have direct impact on its performance and efficiency, and you need to choose the right “features” of the data wisely. But once you have a reasonable result on a smallish dataset, expanding the amount of data it is able to process will improve its efficiency.

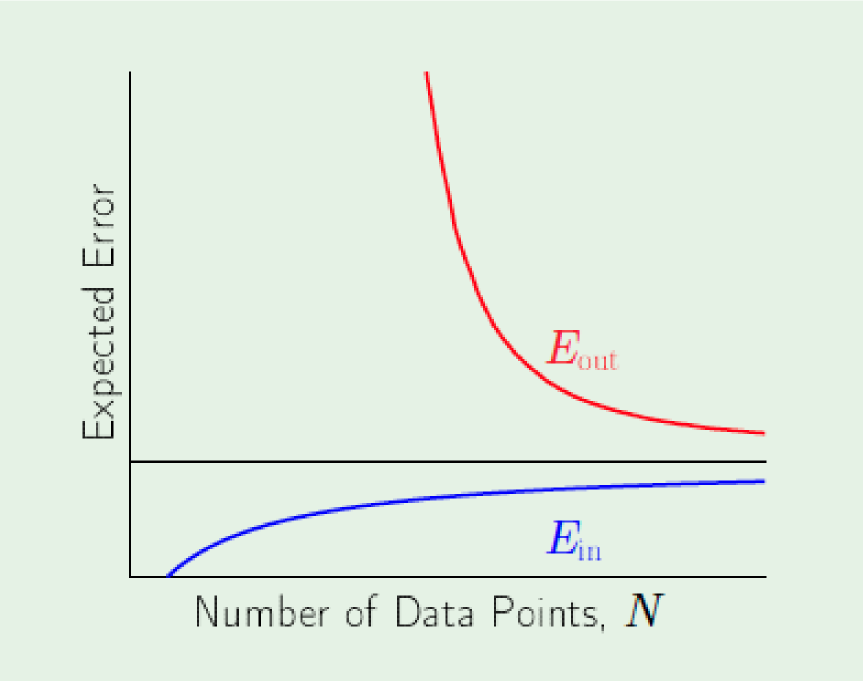

Figure 3 - Theoretical description of expected Ein (classification error in training set of the model) and expected Eout (classification error on new data that is predicted using the model). Source: Caltech, Learning from Data Course materials

Given the amount of log data I mentioned before that is just sitting inside the organizations, seems like a good deal to me.

# Designing a model to detect external agents with malicious behavior

## Data Collection

As an example of these capabilities, we present an implementation of a supervised learning model that we created based on public summarized log information made available by the SANS Technology Institute[3], more specifically by the DShield Project API[4].

The DShield project collects firewall logs from contributors all over the world and summarizes the blocked connections to pinpoint hosts that are engaging in vulnerability exploitation or port scanning in relation to these corporations.

Also, the data volumes are impressive, as we can see on the bulk data made available that there is a daily average of 1 million relevant summarized events. Back-of-the-envelope calculations have us estimate these summaries come from 30 million firewall blocks per day.

SANS offers an open API that provides access summarized anonymous information from this sources including information or malicious IP addresses, domains, AS, and the such, where malicious has a definition as simple as "this IP address has been hammering closed ports on these firewalls for the longest time now". Low hanging fruit it seems, but simple and effective triage.

## Model Intuition

This data is routinely used as an OSINT source for blacklisting IP addresses on some SIEM and Monitoring deployments. However, the effectiveness of this measure is debatable at best, because active attackers, and even scripted worms and bots will constantly change IP addresses.

However, they will not move very far.

There is a tendency of *bad behavior* being dependent on the topology of the Internet, that is if you have machines that are engaging on a specific type of behavior in the internet, their *neighbors* (as far as Organizational and ISP Autonomous Systems (AS) are concerned) are more likely to engage in this behavior.

This thesis is not only supported by common knowledge, one of the greatest inspirations of this work was a PhD Thesis paper by Moura called "Internet Bad Neighborhoods"[5] and a companion paper called "Internet Bad Neighborhoods Aggregation"[6] that talk about some experiments he and his collaborators have performed using traffic from RNP, the Brazilian academic research network. Same conclusions can be drawn by Google's Recent Safe Browsing Transparency Report [7].

But they will eventually change over time.

This information on bad behavior is time-sensitive. If a specific IP address attacked your organization a week ago, this is probably relevant information for the monitoring team if they should commit resources to investigating events from this IP address or nearby IP addresses. If it attacked yesterday, that is certainly more relevant to the team.

All Security Monitoring professionals will remember a few occasions when they would be on duty and "recognized" an IP address on the dashboard or event stream and some kind of "intuition" clicked that that was an event they had seen before and it should be followed up.

This is the intuition this algorithm intends to capture, using the data gathered from the DShield data to predict if an IP address is likely to be malicious or not.

## Feature Engineering

In order to create the features we need to the machine learning algorithm, we go through the following steps:

1. Initially we cluster the events we have by specific behaviors, such as blocked attempts on port 3389, 22 or 80 or port scans (where the same IP address attempts to connect to multiple ports). These clusters of events are deemed positive for bad behavior in our model, and each of the IP addresses get a rank of one for that day. These clusters from this example were arbitrarily chosen, but could have been selected by an unsupervised learning algorithm.
2. After we have identified these "bad" IP addresses per day, we need to add up their scores. However, we need to attenuate the contribution from each day as the days pass, as described in our time-based intuition component. For that, we apply an exponential function to the ranks of the IP address on each date, so that it will have a specific half-life.
    a. One of the behaviors we get from this is that if you choose a half-life around 5 to 7 days the contribution from IP addresses that last showed up their contribution will all but disappear in about 90 days or so. This helps the algorithm take in account changes in the Internet topology, or even different threat actors that might be starting to target the organization or getting bored and moving elsewhere.
3. Now that we have ranks for each IP address over time, we group them in sensible ways to exploit the neighborhood behaviors we discussed on the intuition. We compose the IP addresses by arbitrary net blocks (such as /16 or /24) and by which organizational and ISP AS they belong, being careful to normalize the rank by the number of possible IP addresses on the range. The contribution from each IP address must be proportional to the total number of IP addresses we have on the specific grouping.

With these calculations, we have all the features we need to train a classifier based on these IP address ranks where we select a group of these features

## Training the model

Given the rank calculations on each IP addresses, we can select a group of IP addresses from the DShield summary events and from contributed blocked logs. We select a few tens of thousands of observations assigning them the label of malicious event.

However, in order to properly train the algorithm, we require benign IP addresses as well. Otherwise any algorithm can just infer the trivial case that "everything is malicious".

In other to provide sufficient non-malicious IP addresses to train the model, we made use of the lists of Top 1 Million domains provided by Alexa [8] and the Google Safe Browsing [7] initiative. These are not perfect sources and malware domains have been known to creep in, especially when you approach the end of the list. We have found that the initial 20.000 to 40.000 entries hold really well, and provide a good diversity of origin countries. This helps the model not trivialize on specific regions of the world as bad actors given the predominance of US-based log sources on the DShield sampling.

With the dataset created with the selected features, it is just a matter of using your favorite classification algorithm. After some experimentation, we have selected Support Vector Machines[9], which do a very good job in separating non-linear groups when the features are numeric (which is our case here). We do not go into the details of the specific math machinery behind this, as it would deviate from the point of this document, but this specific algorithm can be trivially implemented with scientific languages or libraries in Java, Python or R [10].

As in many problems in analytics and machine learning, getting to a good answer is now usually the problem. The machinery is all there, but you need to ask a good question.

## Results

The model was trained using a technique called cross-validation.[11] That basically means that part of the training data of the model is not used to train the model, and is actually used to validate the results from the model trained on the other part of the data (which is then called the validation set). So in machine learning terms, if you do a 10-fold cross-validation, you are effectively dividing your dataset in 10 equally sized subsets, and training your model 10 times, always leaving one of the subsets out of the training. You then predict what the subset classification would be if you did not know it already using this recently trained model. The average of the classification errors you get from each subset (what percentage you got wrong) is called the cross-validation error, which is a good estimation of the model error on new data. [11]

For the model described above, using DShield data for the last 6 months, we are able to reach an <u>83 to 95%</u> of cross-validation accuracy.

- The results vary from the way we cluster the "bad behavior" of the data: for instance, it is much more efficient when we are specifically targeting ports 3389 or port 80, but is on the low ends of the results when we target port 22.
- There seems to be a high correlation on the amount of observations on the DShield data with this predictive power, as it is very skimpy on port 22, for example, and many days' worth of data have to be incorporated into the model for it to have enough data points to beat the "dimensionality issue" we described before. However we cannot say for sure if this is case without more data to test the models with.

When you generalize a model to completely new data, it is normal that a model loses accuracy, and sometimes it can be a deal breaker. However as we tested the trained models on log data provided by volunteers on the respective days of the trained models, they showed a 70 to 92%of true positive ratings (sensitivity/precision) on items that had previously marked as malicious and 95% to 99% true negative ratings (specificity/recall) [12].

Statistically speaking, that provides an odds likelihood [13]that an event pinpointed by the model in this example has 13.6 to 18.5 times more likelihood to be malicious than one that did not. That on itself could greatly assist the monitoring team that should be reviewing these logs.

Additional details on the accuracy, true positive and true negative ratios for the different experiments will be made available in the presentation itself during the conference.

## Future Direction and Improvements

The main challenges involve the gathering of more historical data to improve the performance of the model further. Additionally, using IP addresses for the correlation makes the detection susceptible to diversionary techniques such as open proxies, Tor and address spoofing (particularly for UDP), making those sources appear malicious as well over time if they are used for malicious purposes.

In order to develop this concept further, we have put together a project called MLSec (Machine Learning Security)[14] at https://www.mlsecproject.org, created to evolve new Machine Learning algorithms focused on assisting Information Security Monitoring.

We set up a free service where individuals and organizations submit logs extracted from their SIEMs or firewalls and have access to automated reports from the algorithms to pinpoint points of interest on the network. Their feedback on the detected events also helps us fine-tune the algorithms over time.

The service is being scaled out to accept more participants to contribute with log data and feature requests to improve the current models and to develop new ones.

## Acknowledgments and thanks

The author would like to thank:

- Sans Technology Institute, and their DShield Project
- Team Cymru, for their IP to ASN Mapping project and Bogon Reference
- MaxMind for their GeoLite data (http://www.maxmind.com)

## References

[1] "Sci vs. Sci: Attack Vectors for Black-hat Data Scientists, and Possible Countermeasures" - Joseph Turian - http://strataconf.com/strata2013/public/schedule/detail/27213

[2] "Antifragile: Things That Gain from Disorder" - Nassim Nicholas Taleb - http://www.randomhouse.com/book/176227/antifragile-things-that-gain-from-disorder-by-nassim-nicholas-taleb

[3] SANS Technology Institute - https://www.sans.org/

[4] DShield API - SANS Internet Storm Center - https://isc.sans.edu/api/

[5] Internet Bad Neighborhoods – G. Moura http://www.utwente.nl/en/archive/2013/03/bad_neighbourhoods_on_the_internet_are_a_real_nuisance.doc/

[6] Internet Bad Neighborhoods Aggregation - http://wwwhome.cs.utwente.nl/~sperottoa/papers/2012/noms2012_badhoods.pdf

[7]: https://www.google.com/transparencyreport/safebrowsing/

[8]: http://www.alexa.com/topsites

[9]: Support Vector Machines for Classification - Dmitriy Fradkin and Ilya Muchnik - http://paul.rutgers.edu/~dfradkin/papers/svm.pdf

[10] R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

[11] Cross-validation (statistics) - https://en.wikipedia.org/wiki/Cross-validation_(statistics)

[12] Sensitivity and Specificity - http://en.wikipedia.org/wiki/Sensitivity_and_specificity

[13] Likelihood ratios in diagnostic testing - http://en.wikipedia.org/wiki/Likelihood_ratios_in_diagnostic_testing

[14] MLSec Project - http://mlsecproject.org/