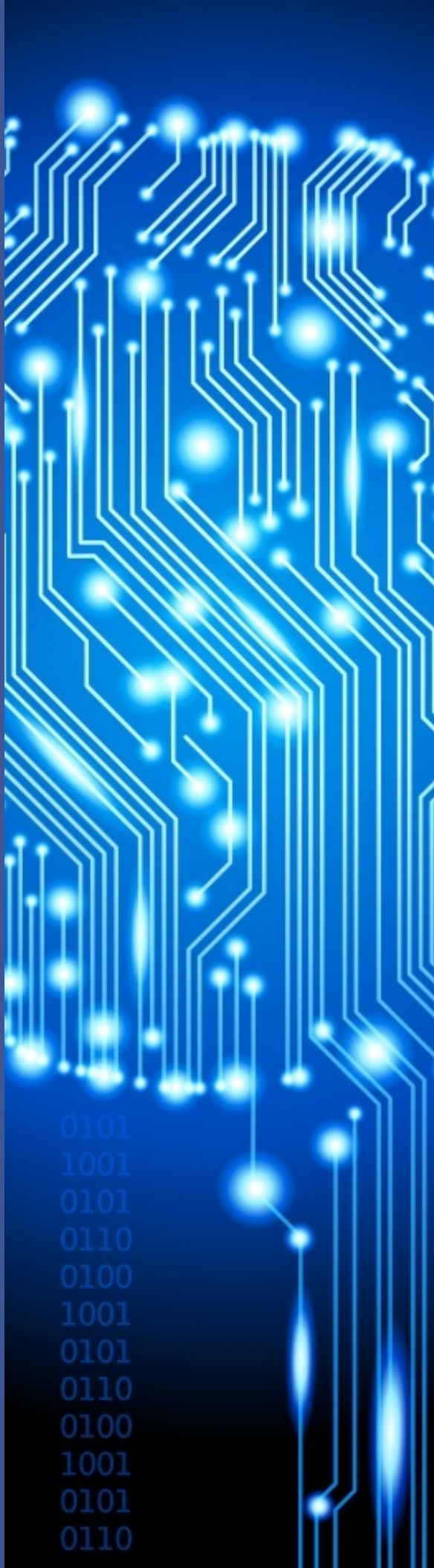
The Application of Artificial Intelligence in CyberSecurity

David Vassallo, CTO

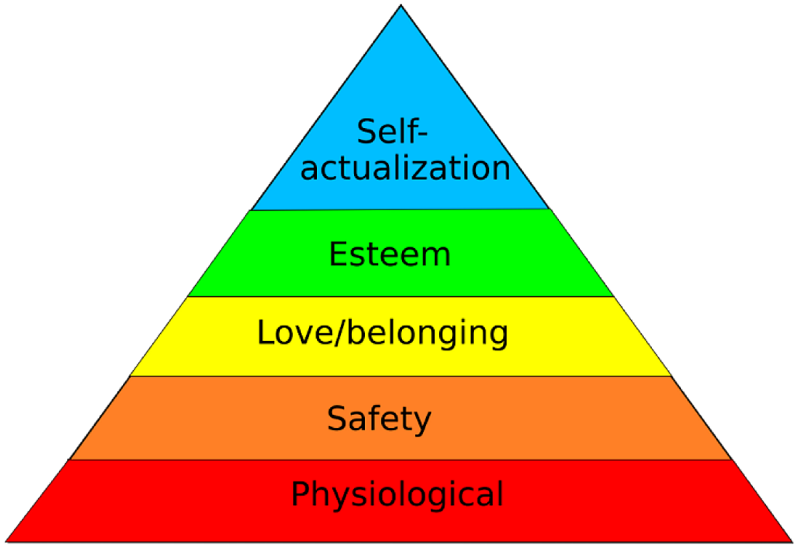


Dedicated to those who silently watch over our Systems

Contents

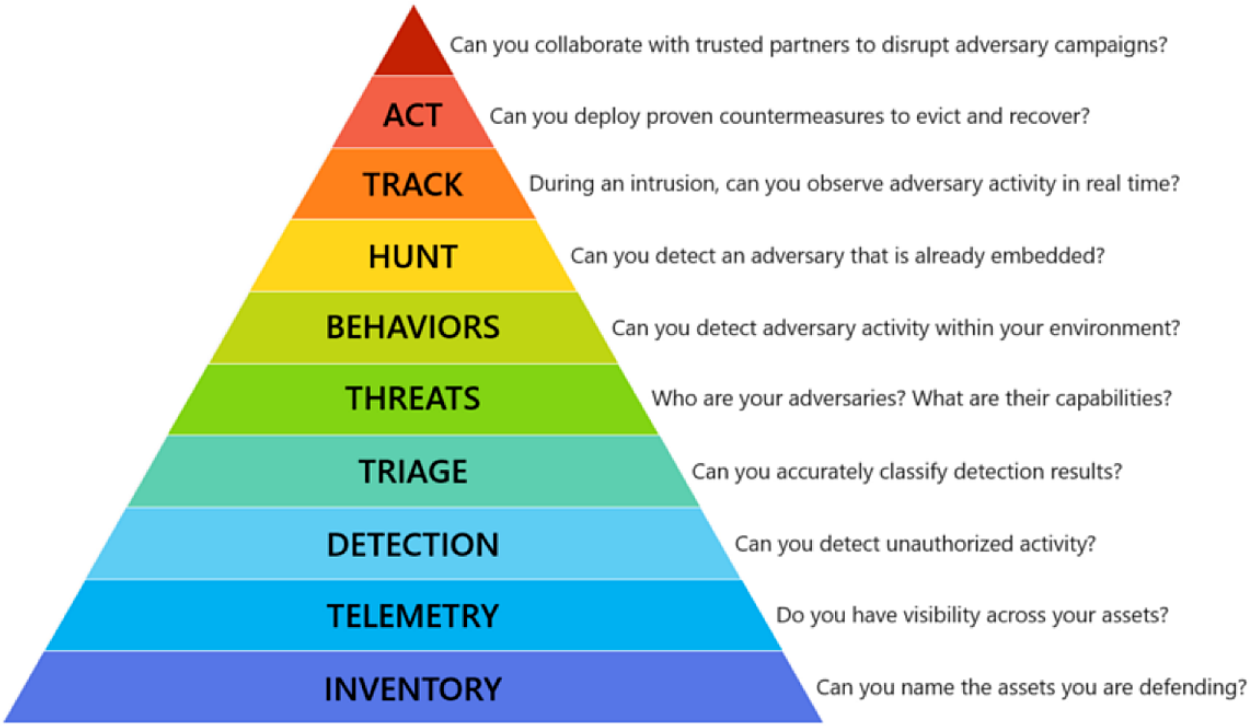
We are failing at the CyberSecurity hierarchy of needs

Many of us are probably familiar with the concept of the “hierarchy of needs”. The concept is usually depicted as a pyramid with humans’ more essential needs — such as bodily function — at the bottom, with more non-essential but rewarding needs towards the top:



Marlow’s Hierarchy of needs (https://en.wikipedia.org/wiki/Maslow's_hierarchy_of_needs)

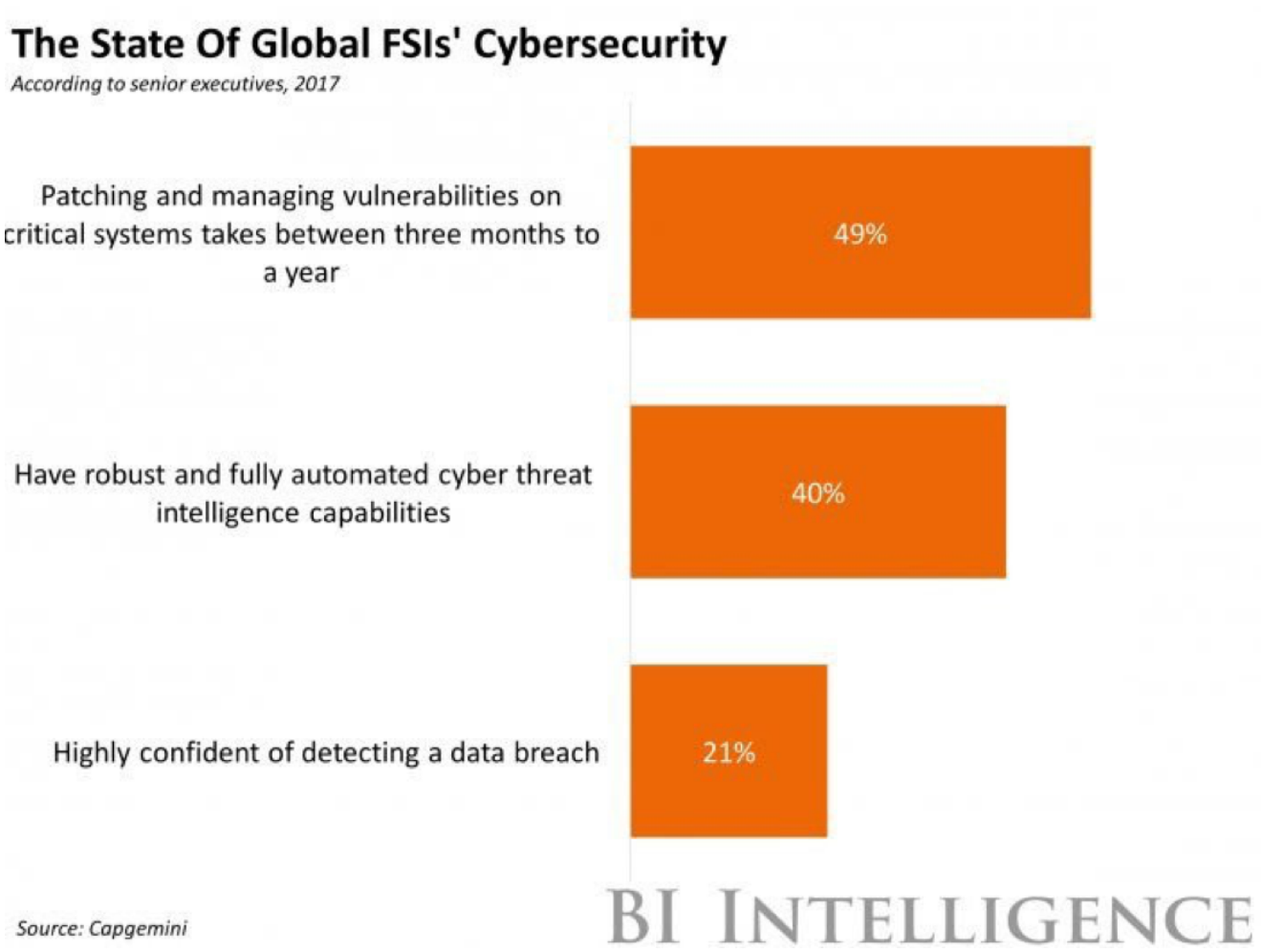
What if we extend this concept of a hierarchy of needs into CyberSecurity? It would help business owners and risk management teams asses how to approach implementing a holistic approach to protecting their business. Matt Swann went ahead and did just that. Here is his version of the Cyber Security hierarchy of needs:



I believe it's an excellent prioritization strategy to follow—but I probably can't name a handful of organizations that I've worked with or for who implement more than the first two layers properly.

With enough discipline and automation in place, it is possible to achieve the first two basic layers successfully. Even basic actions like running periodic scans in your network or using centralized host provisioning help in this aspect. Simple NMAP scans will help you answer "*Can you name the assets you are defending?*", while provisioning tools like SCCM, Chef or Puppet can ensure that telemetry agents are installed on all your assets, feeding data into a central location.

The next layer "*Detection*" is where things get interesting. How confident are you in your organization's ability to "*detect unauthorized activity*"? A recent article from business insider reveals that only 21% of financial services institutions can answer this question confidently.



Another striking observation made from the above chart

40% of respondents think they "*have robust and fully automated*" systems in place, but from those **only about half** of the respondents trust these systems to detect a breach. Even if you are in the minority of companies that have countermeasures in place, businesses seem not to trust these countermeasures very much—and we're still on just the third layer of needs for an effective security strategy—seven more layers to go.

One theory to explain this observation is that most systems out there today rely solely on signatures and rules. If your adversary uses different attacks to those you have rules for, they can slip past undetected. On the other hand, systems based solely on anomalies are known to have difficulty coping with false positives, and with false positives already a problem in signature based systems ("Squealing" anyone?) we're left with the thought that clearly, as an industry, security providers are failing at the hierarchy of needs.

Like most other problems, I believe that the best approach is a balanced approach. We should leverage the strengths of both signature-based and anomaly-based systems to cancel each other's weaknesses. Anomaly-based systems can detect novel attacks that signature based systems cannot, while the latter can help reduce the amount of false positives by providing additional context to alerts (you would be much more likely to consider an anomaly as interesting if the IP address involved has just triggered a "scanning" signature rule). This is approach is the one we have decided to adopt here at CyberSift, with some very promising results.

While effective tools and Intrusion Detection Systems are just one piece of the puzzle, a properly designed IDS will help an organization define effective security practices and response plans that all together are the basis of tackling the security hierarchy of needs appropriately.

CyberSift—*Security made intelligent. A hybrid IDS which leverages both signature & anomaly data mining techniques to simplify CyberSecurity.*

First Steps in applying machine learning to InfoSec

The intersection between machine learning [ML] and information Security [InfoSec] is currently quite a hot topic. The allure of this intersection is easy to see, security analysts are drowning in alerts and data which need to be painstakingly investigated and if necessary acted upon. This is no easy process and as was seen in the now infamous Target hack, more often than not alarms go by unnoticed. ML promises to alleviate the torrent of alerts and logs and (ideally) present to the analyst only those alerts which are really worthwhile investigating.

This is by no means an easy task however the rise of several enabling factors has made this goal reachable to the average InfoSec professional:

- Cloud Computing
- Big Data technologies such as Hadoop
- Python (and other language) libraries like Scikit-Learn [1] which abstract away the nuances of Machine Learning and Data Mining
- Distributed Data/Log collection and search technologies such as ElasticSearch [2]

From personal experience the process of learning about machine learning can be daunting, especially to those not of a mathematical background. However, in this series of articles I plan on outlining my learning process and enumerating the various excellent resources that are freely available on the internet to help anyone interested in getting started in this exciting field.

A good introduction into this field is a talk by @j_monty and @rsevey about "**Using Machine Learning Solutions to Solve Serious Security Problems**" which can be found here:

https://youtu.be/48O6L_DfE2o

The talk really whets your appetite for this field. A small distinction that should be pointed out is the difference between "**machine learning**" and "**data mining**". Data mining is the process of turning raw data into actionable information, while machine learning is one of the many tools/algorithms that help in this process. The presenters mention using WEKA [3] to get started in the field and get to grips with understanding the data that will eventually power our algorithms and machine learning. Before anything else, it will be very useful to manually try some data mining techniques to understand our data, which algorithms to apply to this data for best results and understand the challenges and rewards of doing so. This will allow us to better understand which machine learning algorithms we can later apply to infosec related data such as logs, pcaps and so on.

So it would seem WEKA is as good a place as any to get started! Some quick research turns up a hidden gem.... an online course from the creators of WEKA on how to use the program: https://weka.waikato.ac.nz/dataminingwithweka/preview

The course may not be open when reading this, however the course videos are still available on YouTube and this should be your first stop:

http://www.cs.waikato.ac.nz/ml/weka/mooc/dataminingwithweka/

Note: if you need to find the datasets the instructor is using (the WEKA installation from the Ubuntu repositories do not include these), then you can find them here:

http://storm.cis.fordham.edu/~gweiss/data-mining/datasets.html

References

[1] Scikit-learn : **http://scikit-learn.org/stable/**

[2] ElasticSearch: **https://www.elastic.co/**

[3] WEKA: **http://www.cs.waikato.ac.nz/ml/weka/**

The importance of data mining in the field of CyberSecurity

In a very interesting article on TechCrunch, Michael Schiebel writes about the various ways in which security analysts can learn from data scientists. He makes a couple of points that are worth highlighting.

Today, hacking is a much more complex art than it used to be: It no longer only involves just scanning and penetrating the network via a vulnerability. Yet the traditional security tools used by most companies are often inadequate because they still focus on this.

As any security professional can attest to, hacking nowadays has become easier than ever. Just a few years ago, script kiddies were relegated to using the venerable Nmap and brute force programs like THC Hydra. Nowadays it's a different story. There are a plethora of highly sophisticated (and effective) exploit tools such as Metasploit, the Social Engineering Toolkit and Powershell Empire. These tools are easy to learn, easy to extend, and excellent at what they do. Not only that—most of the tools are free and open source. At any stage of the attack lifecycle hackers can find amazing tools to help them do their job.

Yet we as CyberSecurity vendors are lagging behind especially when it comes to tool-sets. As Michael states:

"Most tools are still role-based, with signatures, detection and response rules. That's their downfall."

Again, we couldn't agree more. Signature based tools still play an important part in Cyber defense, but the defense-in-depth principle requires us to deploy tools which can mitigate those threats which pass through our outer rings of defense. Luckily, Cyber defense tools are evolving, with the help of open-source innovation in both

security and big data fields.

Focus on the abnormalities

This is what it's all about. Effectively finding abnormalities in your network has a couple of very important benefits to your organization:

- It forces you to be more aware of your networks and systems. You are required to investigate abnormalities and effectively determine if an abnormality is expected or malicious. The more aware you are of your environment, the less time it takes you to realize when something goes horribly wrong (like in the event of a hack…)
- With the proliferation of advanced attack vectors (like steganographic attacks) and cloud computing, it's very easy for hackers to use legitimate services to carry out their attacks in such a way as to avoid tripping signature based alarms. Signatures that target AWS or Twitter would be triggered so many times that they would be ignored, even though they are potential avenues of attack already being exploited by hackers. Abnormality detection systems can flag connections which use these services in weird ways (too much data being transferred, too many connections being done, periodic connections to previously unused endpoints, and so on…)

At this stage it's important to note that abnormalities do not automatically mean malicious activity… an anomaly based system highlights those events that deviate from the norm. There are several examples of genuine anomalies which are not malicious:

- Marketing executes a successful campaign resulting in a flood of connections to your webservers
- A misconfiguration is introduced during one of your changes to a backup system which causes high volume traffic to flow through the wrong network path
- Your organization engages with customers in new markets, leading to your network having new traffic patterns to previously non-contacted countries and Autonomous Systems

These are practical examples of how an anomaly based system increases your team's awareness of the environment. This leads me to prefer referring to anomaly based systems as ***"cyber-awareness"*** platforms rather than simple "cyber-defense".

The real problem in most organizations is that too much security alert data is coming in too fast.

Michael again hit the nail on the head here. If your security analysts are investigating too much data, then no wonder we're seeing alarming headlines such as:

Most companies take over six months to detect data breaches (by ZDNet)

Anomaly based IDS help your analysts focus on those alarms that can be important, reducing their mitigation time and efficiency—and at the end of the day this is what translates to cost savings for the organization.

Here at CyberSift we are building next generation anomaly detection systems which are based on the above principles and add an effective layer of defense which counters new threats as they emerge without the need of signatures or rules, all the while increasing your team's cyber-awareness of their systems and networks. Stay tuned for exciting developments...

Read the full article "What your security scientists can learn from your data scientists to improve cybersecurity" here.

OSSEC Hyperalerting

Every security professional has heard of the PCI Security Standard. The standard is mainly applicable to systems that store, use and transmit payment card details, and it lists out a number of controls, such as:

- 10.6 Review logs and security events for all system components to identify anomalies or suspicious activity.
- 11.4 Use intrusion-detection and/or intrusion-prevention techniques to detect and/or prevent intrusions into the network.

Common sense really—but legally binding in some industries. One of the PCI controls (10.6.1) states:

Review the following at least daily: All security events, Logs of all critical system components, etc.

That is a tall order right there.

Let's pick on one small component of an overall PCI strategy: **Host-based Intrusion Detection [HIDS]**, a very popular—open-source—example being OSSEC. Below is a graph showing the number of OSSEC alerts *generated by a single server,* over the course of about 8 hours:

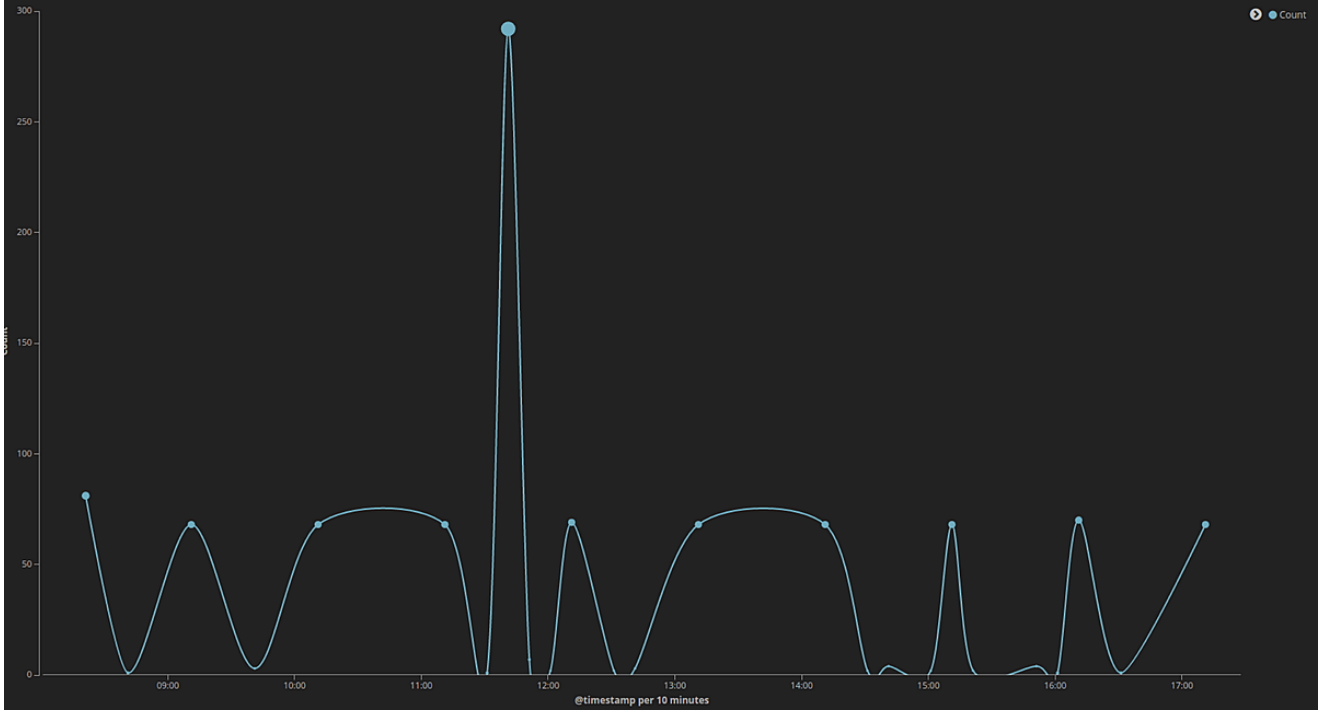


Fig 1: OSSEC alerts over time plotted by CyberSift

That's about **1023 events**. Granted, we didn't spend much time tuning our installation—but then again, which of your over-worked security analysts has got the time? This brings across the point why that particular PCI control is a tall order—imagine having to sift through thousands of alerts every day to figure out if there's anything unusual.

There is a helping hand however... ever heard of "*hyperalerting*"? Computer science researchers define hyperalerts as follows [1]:

A network intrusion hyperalert is an aggregation of related alerts. Several different intrusion alerts may be related to one attack.

Imagine a system which aggregates your OSSEC data into a neat timeline and groups similar logs together, making it easy to view alerts at a glance. We upgrade from a simple graph shown above to something more like an alert timeline shown below:

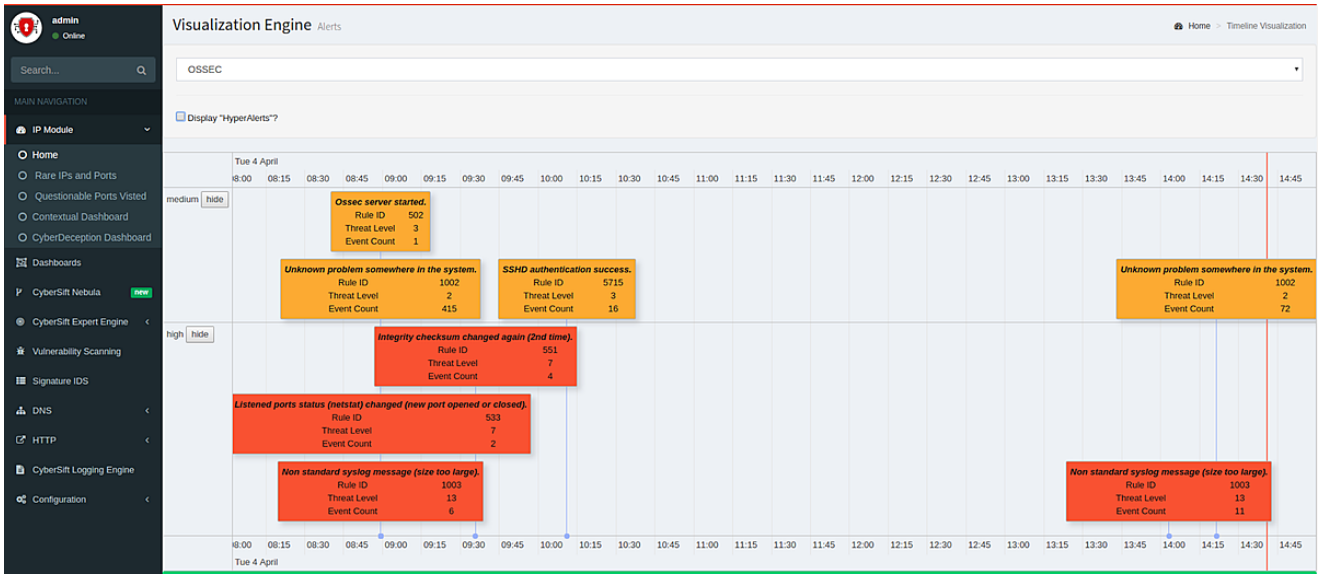


Fig 2: CyberSift Visualization Engine Summarizing OSSEC Alerts

Much better! CyberSift automatically groups alerts with the same ID together, and clearly demarcates severity levels. Presenting the alerts on a timeline allows the analyst to quickly identify “hotspots” of anomalous activity. Clicking on summarized anomaly will bring up the original logs that caused the alert:

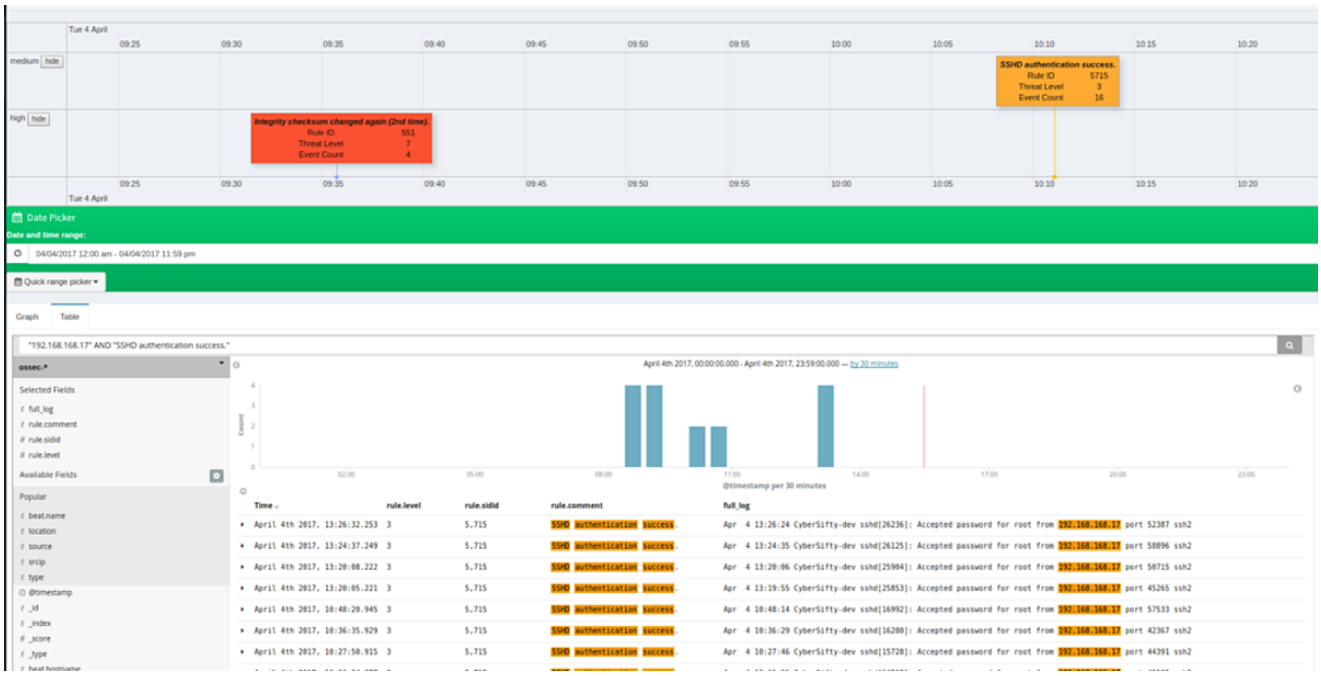


Fig 3: Original OSSEC logs filtered and displayed

But what if we can take this one step further? The summary is a good start but it doesn't help dealing with the sheer volume of generated logs. CyberSift applies anomaly detection to hyperalerts as well, detecting unusual "**transitions**". A transition is one hyperalert followed by another in a particular sequence. In other words, CyberSift can highlight anomalous groups of alerts that don't usually appear together for a given system.

Let's look at an example, this time we switch on the "**Display HyperAlerts**" option and we something like this

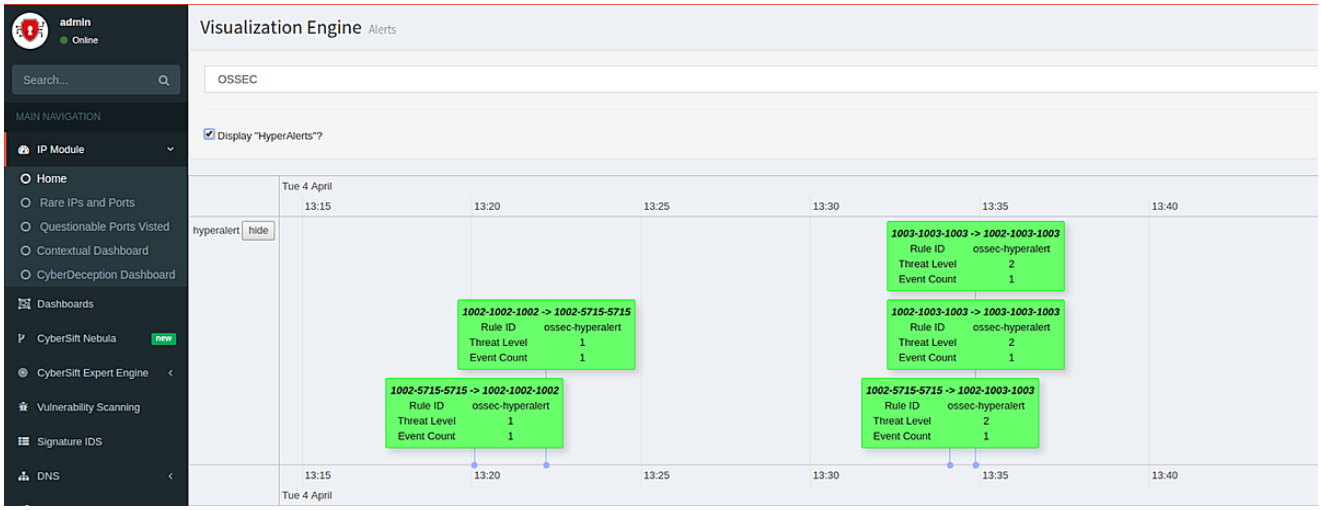


Fig 4: CyberSift Anomaly Hyperalerts

In the screenshot above, we see CyberSift highlighting groups of unusual alerts that occurred close together. Clicking on a hyperalert will bring up the details of the event that caused the anomaly:

Time	Abnormal_Transition
April 4th 2017, 09:20:21.776	5715-5715-5715 -> 1002-1002-5715

Table JSON

t	Abnormal_Transition	5715-5715-5715 -> 1002-1002-5715
t	Full_Logs	Apr 4 09:01:43 CyberSifty-dev sshd[10058]: Accepted password for root from 192.168.168.17 port 40342 ssh2 Apr 4 09:04:36 CyberSifty-dev sshd[10341]: Accepted password for root from 192.168.168.17 port 40543 ssh2 Apr 4 09:08:37 CyberSifty-dev sshd[10762]: Accepted password for root from 192.168.168.17 port 53700 ssh2 Apr 4 09:20:15 CyberSifty-dev sshd[11528]: Accepted password for root from 192.168.168.17 port 52322 ssh2 Apr 4 09:20:50 CyberSifty-dev critical-stack-intel: critical-stack 09:20:50 [#033[32mDEBUG#033[0m] #033[3 ty rated IP Blocklist, *errors.errorString=Feed missing: #033[31mcritical-stack-intel-153-threatcrowd.org- [0m) Apr 4 09:20:55 CyberSifty-dev critical-stack-intel: critical-stack 09:20:55 [#033[32mDEBUG#033[0m] #033[3 itten: A Campaign with 9 Lives (2015-11-09), *errors.errorString=Feed missing: #033[31mcritical-stack-inte es-(2015-11-09).bro.dat#033[0m, string=#033[0m)
t	SourceAddress	192.168.168.170
t	_id	AVs3-zdDPq5pItQGVfgK
t	_index	alerts
#	_score	-
t	_type	ossec
t	severity	1
⏲	timestamp	April 4th 2017, 09:20:21.776

In the above example, we have an abnormal transition

“5715–5715–575 -> 1002–1002–5715”

which, by referring to Fig 2 above, we know means “SSHD authentication success” (SID 5715) and “Unknown problem somewhere in the system” (SID 1002). This group of events is indeed anomalous because the system usually doesn’t see so many consecutive SSHD authentications

Hyperalerting can significantly decrease the number if important incidents that your security team need to investigate. In the below graph we can see the time savings introduced by hyperalerts when monitoring OSSEC alerts. Note how OSSEC outputs upwards of 70 alerts per hour for a single server, while hyperalerts cut down the alerts to a much more manageable 5–7 alerts per hour… often none at all!

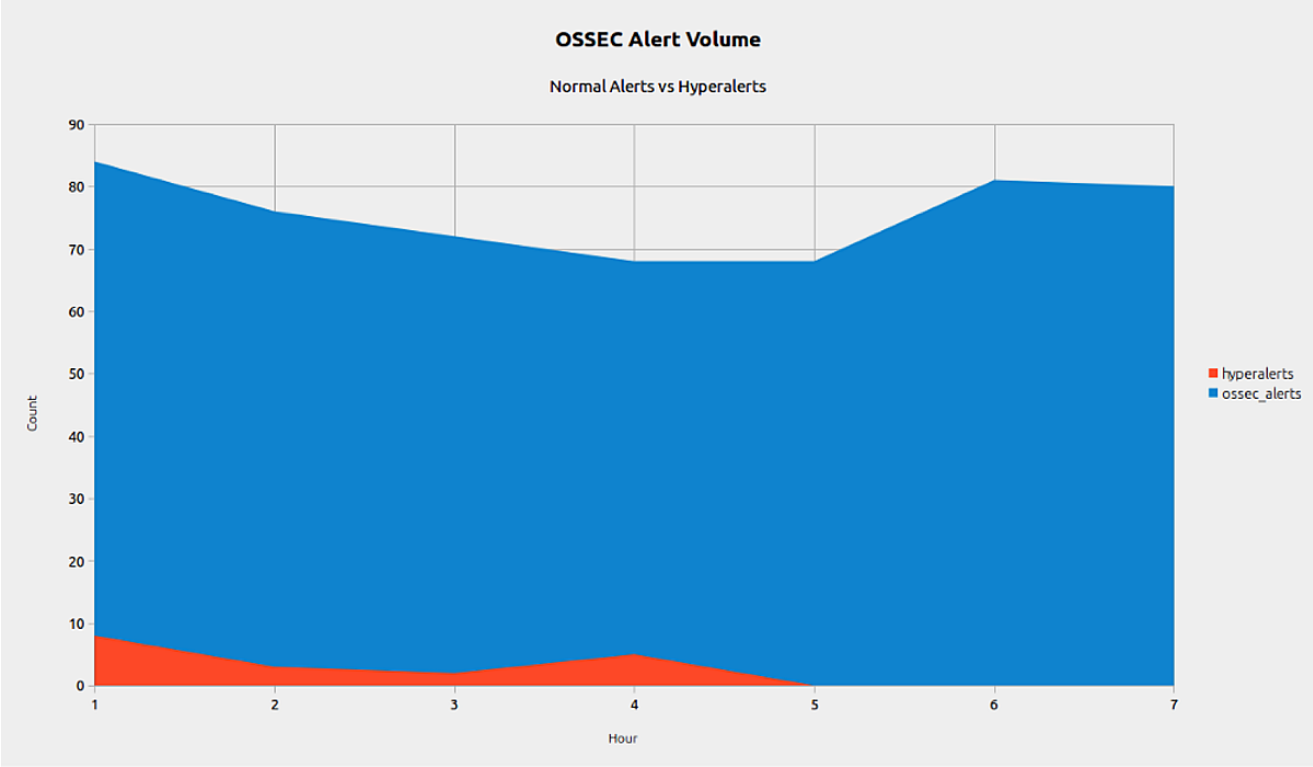


Fig 6: Hyperalerts reduce the number of investigations necessary

References:

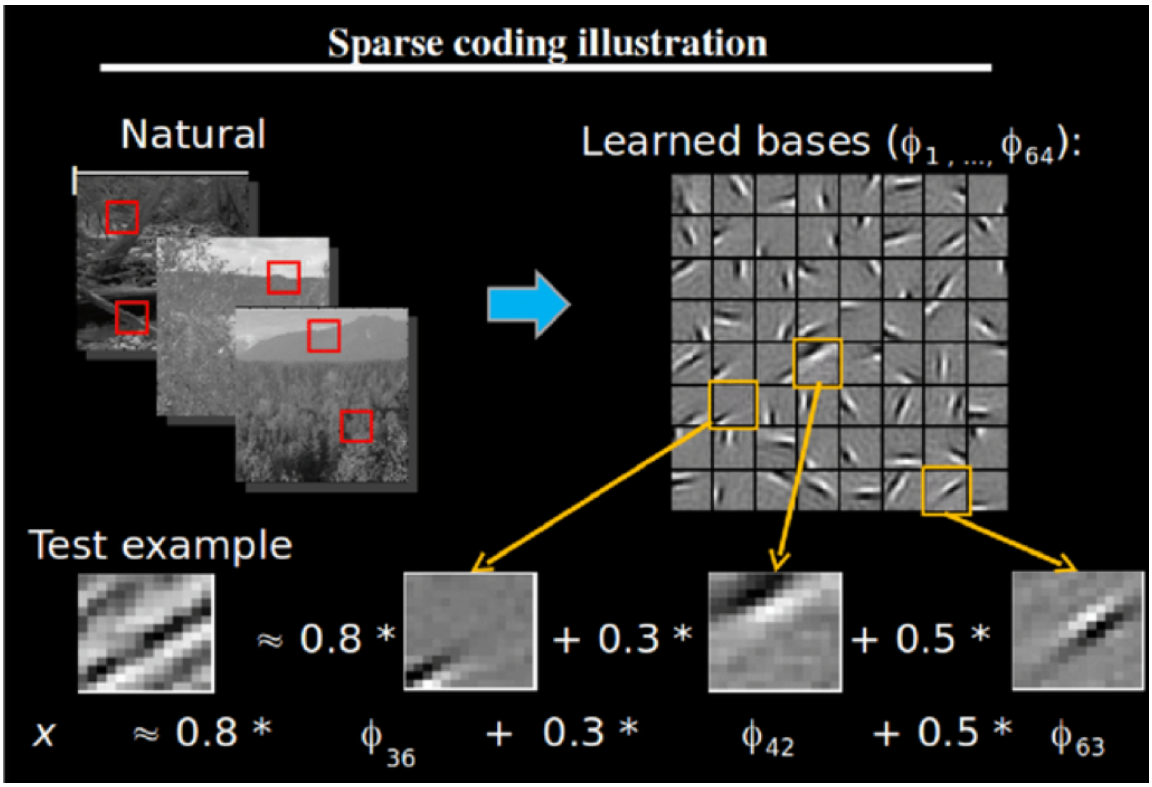
[1] de Korvin, A., Chen, P. and Hu, C., A Genetic Algorithm Approach for Analyzing Network Intrusion Hyperalerts.

Cyber Security: Sparse coding and anomaly detection

In the physical world, we often translate visual data from one “dimension” to another. For example, looking at the picture below, on the left hand side we see a view using night vision—and we’re still unable to pick out any “anomalies”. The anomaly (standing person) becomes pretty clear when we translate the night google data to use infrared instead, and as can be seen on the right hand side, though we lose some image detail we are now easily able to pick out our “anomaly”

In machine learning, we spend a lot of time trying to find “dimensions” to represent our data in such a way as to make the anomalies we’re looking for stand out far more than if we leave the data in it’s original form. There are a multitude of dimensions we can use, the one presented in this article is called “**Sparse Coding**”.

The essence of sparse coding can be explained by examining the figure below:



Imagine we have a set of data (images of a forest in the figure above). We can pass this data through a "**dictionary learner**". The job of the dictionary learner is to decompose our data into a set of unique "**bases**" or "**atoms**". Just like in the real world, a language dictionary can be used to construct sentences. Any sentence I write can be decomposed into individual words that can subsequently be looked up in a dictionary.

Similarly in our previous example above, any picture can be decomposed into bases or atoms which can be found in the dictionary we just built from our training data. In the specific example in the figure, the bottom "test example" is expressed in terms of three basis, each in different proportions (0.8 for the first one, 0.3 for the second one, and 0.5 for the last one)

Applying this to Cyber Security

Intuitively, such a system will struggle to express data it has never seen before—because it lacks the words or basis to decompose this data. Similarly, unusual or uncommon data will be expressed using a different set of words than those used to express common or normal data. Let's test this theory.

Take the following practical scenario:

You collect data logs from your firewall, every 5 minutes. Being a good DevOps engineer, you write a quick script to summarize this data, converting all the data in a 5 minute time windows to:

- *The destination BGP AS number (because tracking each individual destination IP provides too many entries...)*
- *The bytes transferred between your network and the destination AS number during those 5 minutes*
- *The number of clients in your network that communicated with the destination AS number*

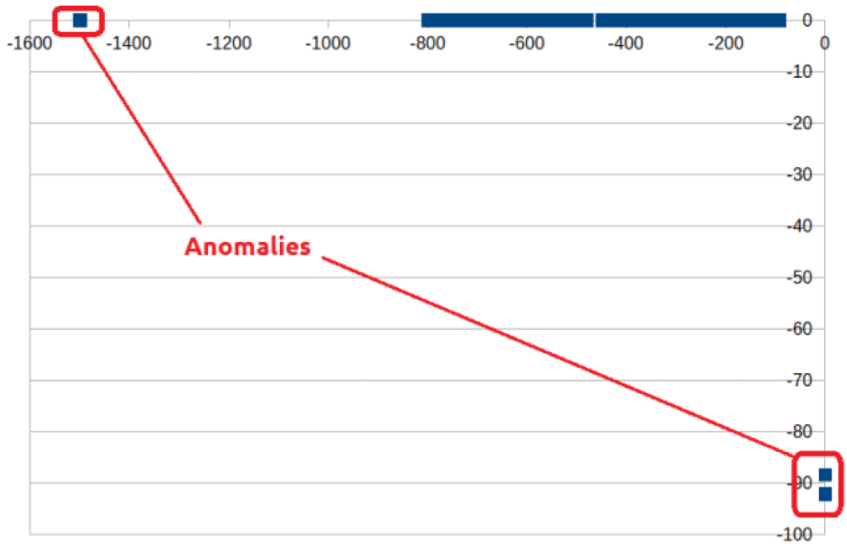
You would end up with a dataset that looks something like the below. I built the below data set by using LibreOffice calc, randomly generating numbers for each entry.

The only difference being the last entry, where I purposely entered an anomalous entry for demo purposes.

Now, you are required to find from within these entries any anomalies or weird data. Ideally, you should be able to use your work to calculate if future data points are anomalies or not.

We can apply the sparse coding principles I introduced in this article, as follows—using python, pandas and scipy:

The above code is basically using sparse coding to translate our data from one dimension to another (keep in mind that when doing so we usually can pick out details that are usually hidden, as in our night vision vs infrared example). The resulting data is shown at the end of the article, but it's easier to visualize the data as a plot, shown below:



We immediately note three anomalies. One translates to the purposely anomalous data point I inserted into the end of our toy data set (as expected), while the other two are anomalies introduced by the random numbers generated. If we examine these further, it turns out that both these anomalies come from AS number "200", which typically has "number of bytes transferred" being over 100. However for these two cases the number of bytes transferred turned out to be lower than expected—at about 80.

And there you have it—a quick and easy way of detecting anomalous data from firewall logs. Not only that, but you can use the dictionary generated by your code to see if new data points are anomalous or not. Of course this method doesn't cover all cases and probably has its own set of problems but it's a very good start considering the minimal amount of work we just put in.

At CyberSift we develop more advanced techniques which leverage machine learning and artificial intelligence to perform anomaly detection as we presented above—but on a much more advanced scale and in a more user friendly manner. Check us out!

PS: Resulting data after sparse coding:

- Machine Learning
- Cybersecurity

CYBERSIFT

What to know more?

REQUEST A FREE TRIAL