COUNTERCEPT

MACHINE LEARNING

A threat hunting reality check

Contents

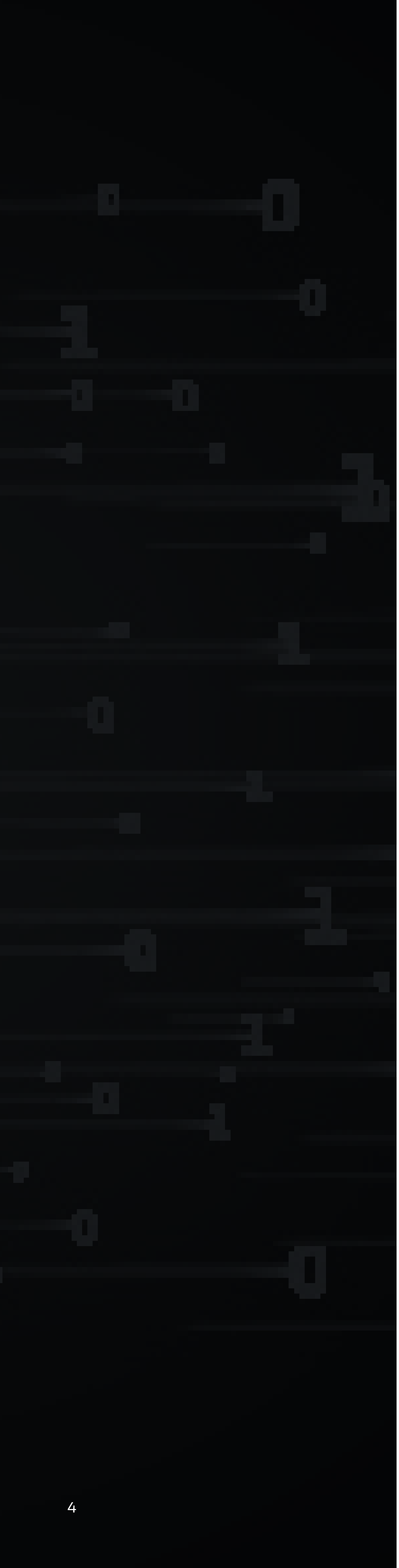
Abstract

Machine learning is a field of computer science dating back to the 1950s but its popularity increased significantly in the 1990s.

In more recent years, it has become a marketing buzzword in a variety of fields, including cybersecurity – yet Gartner listed machine learning at the very top of the 'Peak of Inflated Expectations' in its 2016 Hype Cycle for Emerging Technologies. While machine learning can be powerful when correctly applied and when solving practical problems to which it is well suited, it is often misconstrued as a one-size-fits-all replacement technology able to solve a range of difficult problems on its own.

This paper gives an introduction to the high-level concepts of machine learning and the typical ways in which it is applied to attack detection in the cybersecurity industry. It covers the problems commonly encountered and gives practical examples derived from Countercept's experience of applying machine learning techniques as part of our threat hunting platform. In this way, we demonstrate that machine learning is currently more of an enhancement technology for solving specific security problems than a one-size-fits-all replacement technology. In particular, it will not replace the requirement for a highly experienced attack detection team.

If you are not an expert on machine learning, we hope to take you from the peak of inflated expectations, through the trough of disillusionment and straight onto the plateau of productivity.



What is Machine Learning?

Although a complex and diverse subject, in general terms machine learning is a means by which computers make predictions or discover new information without being explicitly programmed to do so.

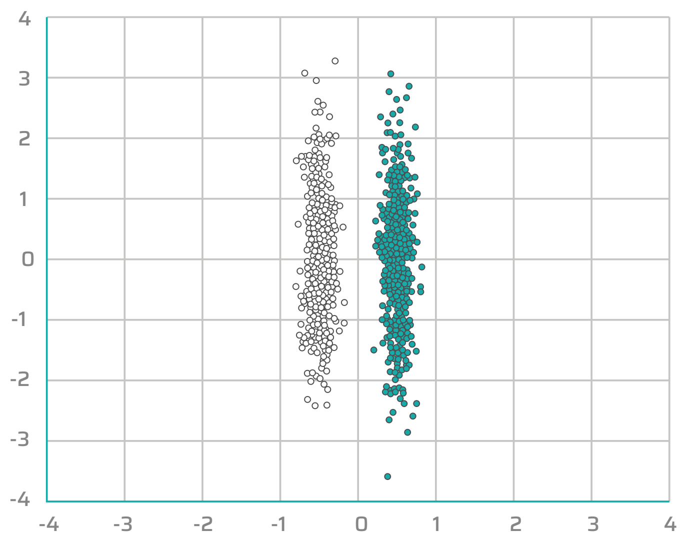
The machine derives its behavior from data, rather than from an explicit set of rules. Usually, it comes down to spotting patterns in numerical data.

In the majority of cases, machine learning falls into one of two categories:

- **Supervised Learning**
  A set of training data is provided with correctly assigned outputs that enable the algorithm to learn how to predict future outputs from new data: for example, facial recognition to match a photo against a known database.
- **Unsupervised Learning**
  This involves identifying trends or patterns within a set of unlabeled data, perhaps by identifying clusters of similar data or by identifying anomalies that do not seem to fit the common theme. An example might be the analysis of water consumption in a given region over time, with the aim of detecting unusual surges caused by burst underground pipes.

Consider the following simple two-dimensional data set. Here we have two classes of data labeled in white and blue respectively and plotted on a graph.

The possible goal of a supervised learning algorithm would be to learn the difference between the white and blue classes. If successful, when a new sample is supplied, the algorithm could accurately predict whether the sample should be white or blue. On the other hand, an unsupervised algorithm might be fed this data without blue or white labels and it should still be possible to identify two primary clusters of similar data – and perhaps a few samples that belong to neither group. The algorithm wouldn't know that one cluster is called 'white' and one is called 'blue', just that there is an observable difference between the two.



How Does This Apply to Attack Detection?

There are many problems in the security industry that could benefit from the help of machine learning.

But there are two areas where security vendors and marketing departments have focused their efforts in recent years.

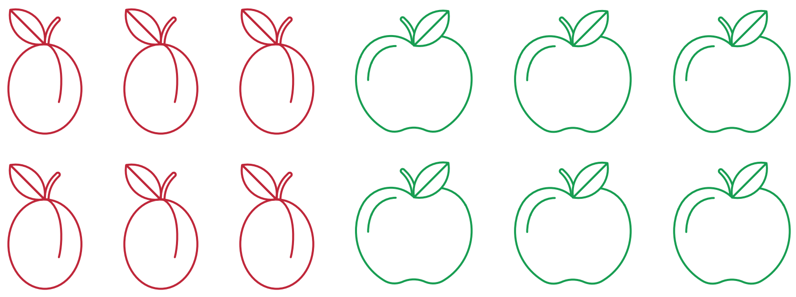
- **Next-Generation Antivirus**
  This area has a fairly broad definition and by no means all approaches are based on machine learning. However, some key players are focused on classifying files, processes and certain types of system behavior as malicious, using machine-learning techniques. The basic premise is that machine learning can develop a more generalized way of identifying malicious behavior that could prove more effective than identifying traditional AV signatures for known malware families. Being heavily malware-detection orientated, off-the-shelf 'next-gen' antivirus products tend to be endpoint focused.
- **User and Entity Behavior Analytics (UEBA)**
  Products in this space are focused on detecting malicious activity by establishing a baseline of normal behavior on the network. Many of these approaches use machine-learning techniques and are generally focused on passive network data or traditional log sources. The premise is that users or machines often have a predictable behavioral profile that can be observed and modeled over time, and hence behavior that deviates from this profile can be considered as a potential compromise or evidence of an insider at work.

Basic Concepts in Machine Learning

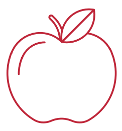
Before considering the security-specific use cases in more detail, it's worth understanding some of the basic concepts and problems generally encountered when using machine learning in any field, to better understand how they affect security applications.

Representative Sampling

An important concept is the idea of representative sampling. In other words, whatever data you supply for training or baselining needs to provide an accurate picture of reality. Consider the following samples as the data supplied when training an algorithm to classify images as apples or plums:



Let's say we trained an algorithm on this data and then tested its performance by asking it to classify the following:



A potential outcome is that the model misclassifies this image as a plum. The data supplied in training was not representative of the wider variation of plums and apples and the model might identify color as the key differentiating factor. While we supplied multiple examples of red plums and green plums, we failed to supply any examples of red apples, yellow plums, etc. Consequently, the model would not realize that color was an unreliable differentiating factor. Similar problems can occur with an unbalanced data set, where certain types of features are represented but in significantly differing proportions.

Underfitting and Overfitting

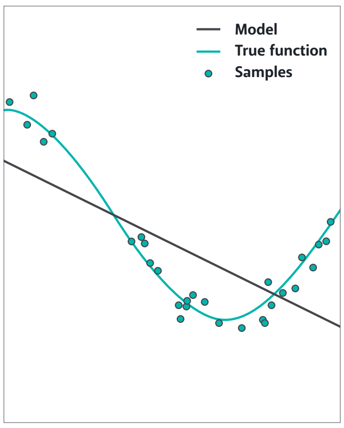
Another potential problem with machine learning is either using a model that is unable to express the complexity of your data properly or one that forms an overcomplicated explanation too affected by unrepresentative examples that better explains the training data, but performs poorly when tested against real-world data.

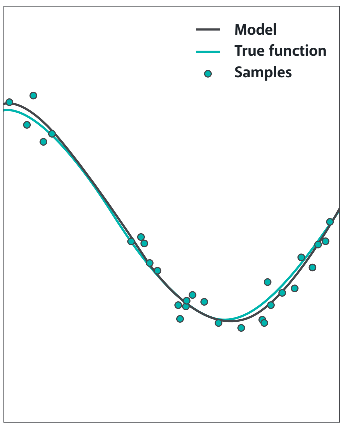
Consider the data below.

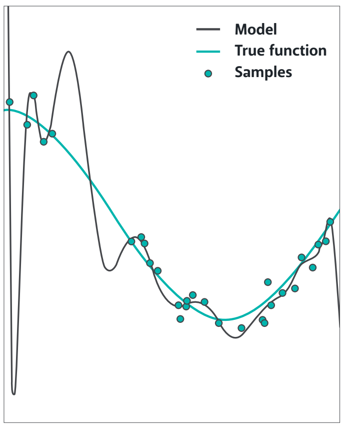
The middle graph is an example of a good fit and is what we're hoping to achieve, with our model closely following reality.

The graph on the left is an example of underfitting, where the model is only capable of seeing the world in straight lines but the underlying data has a more complicated relationship. This would result in a poorly performing model.

Meanwhile, the graph on the right is an example of overfitting, where the algorithm has formed an overly complicated model to more accurately explain the training data. However, it fails to accurately explain the underlying data trend and would perform poorly when tested against previously unseen data. Only the graph in the middle represents an algorithm that has produced a good fit to the data.

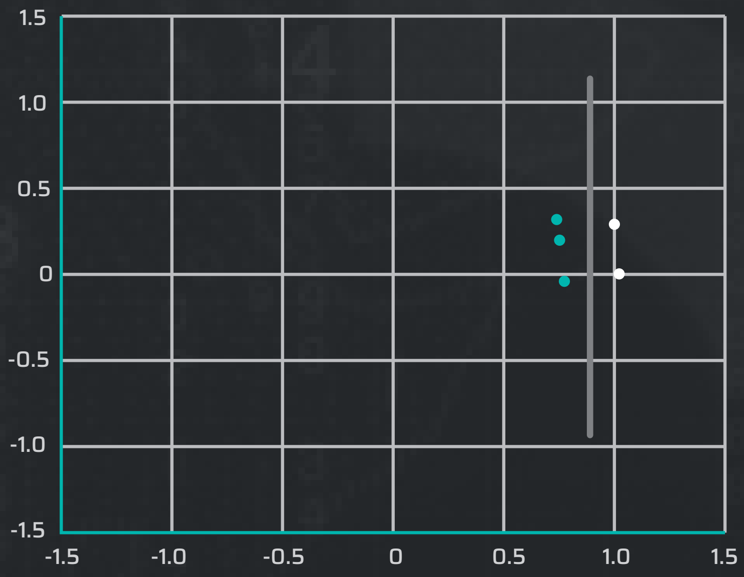




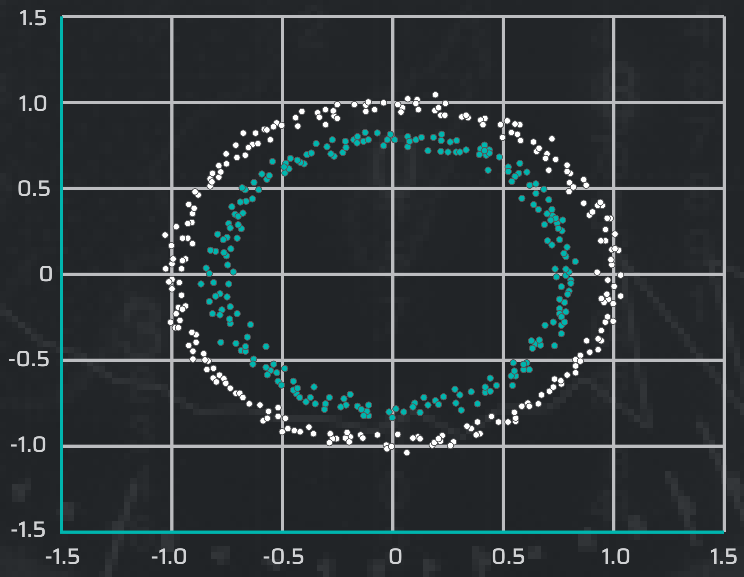


Sample Size and the 'Curse of Dimensionality'

While this also relates to the idea of representative sampling, the actual number of samples available for training or baselining is vital for building accurate models. Let's say you were tasked with drawing a boundary line to separate the following data:

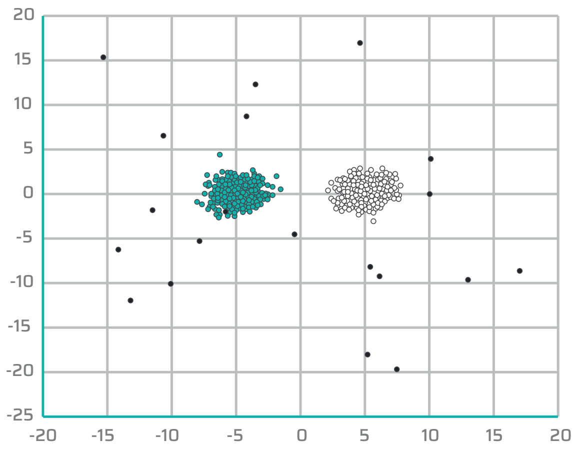


There's very little data to go on but your best guess would probably be to draw a vertical line down the middle and say that blue is to the left and white is to the right. However, with a more realistic sample size, we see that the data distribution is very different and a vertical boundary line is totally incorrect:

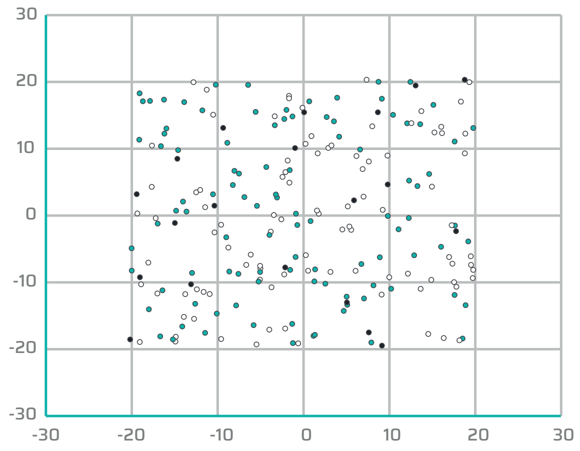


Learnable Problems

It's common for people to see complicated mathematical notations, hear unfamiliar technical terms or witness self-driving cars and consequently think of machine learning and related topics as black magic, where anything is possible. Vendors with a focus on machine learning-based approaches and products are quick to capitalize on this perception, but the reality is that machine learning is not magic; it's simply applied mathematics. In order for it to be useful, it needs to be applied to a problem well suited to machine learning – and the problem itself needs to be learnable.



For example, consider the data set shown above. Let's assume that blue and white samples are normal and we are interested in automatically detecting the anomalous black samples. In this case, blue and white clearly cluster in certain regions and almost all black instances are clearly identifiable as being far from these clusters. This problem is learnable, in that we should be able to model this in such a way as to identify black samples with reasonable confidence.



On the other hand, consider the data above. In this case, there are no discernible clusters, the black samples seeming well mixed with the white and blue, and it's unlikely that we could determine automatically whether a sample should be black. The problem is not learnable, at least not from the above data. We would need additional data features to provide some separation in order to differentiate between the normal and anomalous samples.

Applications to Security Use Cases

We'll now look back at the two key areas of attack detection commonly tackled with machine-learning techniques (next-generation antivirus and UEBA), and consider how the basic concepts discussed above apply in each case.

First, next-generation antivirus.

In order to accurately describe an executable file format in a manner whereby similar variations can be easily clustered, we are likely to need a relatively high number of features – and hence we risk facing the curse of dimensionality. On the other hand, there is an immense variety of malware families out there, with new ones springing up every day, and they can be collected relatively easily. Consequently, we are likely to have a very high sample size that should address this concern.

However, ideally we need a class of legitimate software to include in the training. This is arguably a harder problem, as there is a huge amount of software in use across the world and gathering representative samples for legitimate software is difficult, at least outside the major software types commonly used by organizations. With this is mind, let's consider a few scenarios:

- **Brand-New Variant of the Andromeda Malware Family**
  We'd expect our algorithm to be well trained using a large number of examples of Andromeda, and some key generic traits, highly specific to the Andromeda family, are likely to have been learnt. Without specific effort by the malware authors to bypass our model, we might expect this to be a good candidate for detection. Here we have high sample sizes and relatively representative sampling.

- **Malicious Use of Legitimate Remote Administration Software**
  This could even be software contained within our 'legitimate software' group. We might be able to identify it but we certainly can't explicitly block execution of any software that could potentially be misused. This is not really a suitable case for preventative next-generation antivirus solutions.

  Malware is considered malware because a human has decided its principle purpose is to commit malicious actions. However, there are plenty of examples of dual-purpose software that could be misused. Is a port scanner malicious? Is an SSH client malicious? Our model cannot see who is sitting behind the keyboard and it cannot judge intent, only behavior. Hence this is potentially not a learnable problem, at least not without other data for context.

- **Bespoke Malware Used in Targeted Attacks**
  In this case we can expect a custom malware family, previously unseen, and so we won't have representative samples. Unless it is largely derived from a known family, it's unlikely to be anything other than anomalous in our model, but since much of the software used in the real world is unknown to us, this situation will occur fairly frequently.

  We need to bear in mind that before we automatically block malware, we have to be very, very sure that it is indeed malware or we risk disrupting business operations. A smart threat group might deliberately take known "good" software and make only minor changes to embed their own Trojan functionality, in which case it's more probable that it would be classed as legitimate, not malicious software. The threat group is also likely to test beforehand that it can bypass common commercial antivirus technologies and 'next-gen' approaches are no different.

  Here we have an example of non-representative sampling, as our model has never seen anything like it before. It's also possible that it will have been deliberately constructed to appear similar to something we consider legitimate. Hence we'd be unlikely to detect and prevent this threat.

Moving on to UEBA (user and entity behavior analytics), what type of modeling might we usefully apply to network and log data?

The obvious use cases to consider are the elements of the Cyber Kill Chain that relate to command and control, lateral movement, and data exfiltration. Perhaps we can detect some of the following scenarios?

1. A system performing internal network reconnaissance, e.g. traditional port scanning, DNS enumeration and more modern LDAP/AD-related reconnaissance.

2. A user account operating from a new system or accessing services that it does not usually access, potentially indicating lateral movement or actions on objectives.

3. A system requesting an unusually large amount of data from an internal system (data theft) or transferring an unusually large amount of data to the internet (data exfiltration).

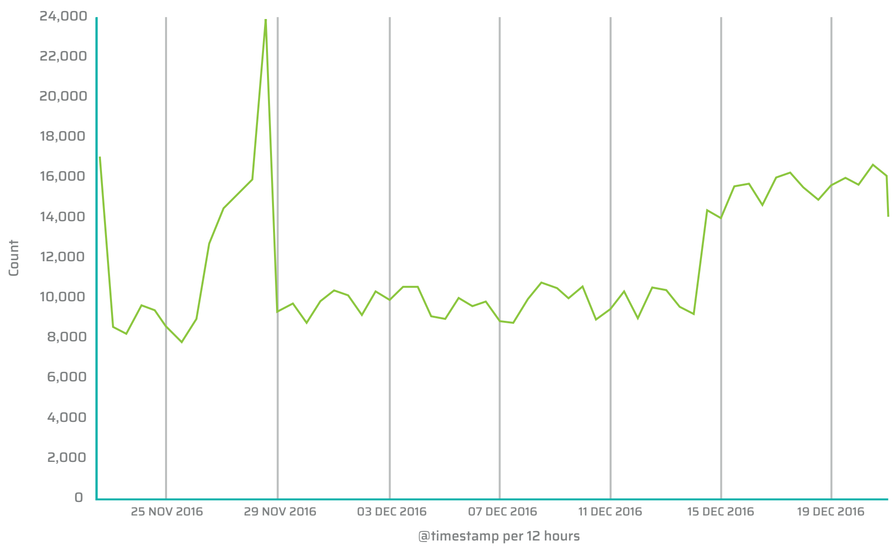
4. A system beginning regular communication with an IP address or domain that it hasn't previously connected to, which could be command and control traffic.

A question we need to address at an early stage is precisely how do we model user and entity behavior? Do we model individual users and entities? Or do we model them in groups of similar types? And how do we accurately identify a user or entity? When it comes to users, it's fairly straightforward as usernames tend to be uniquely identifiable, but in the case of entities it's not quite so simple.

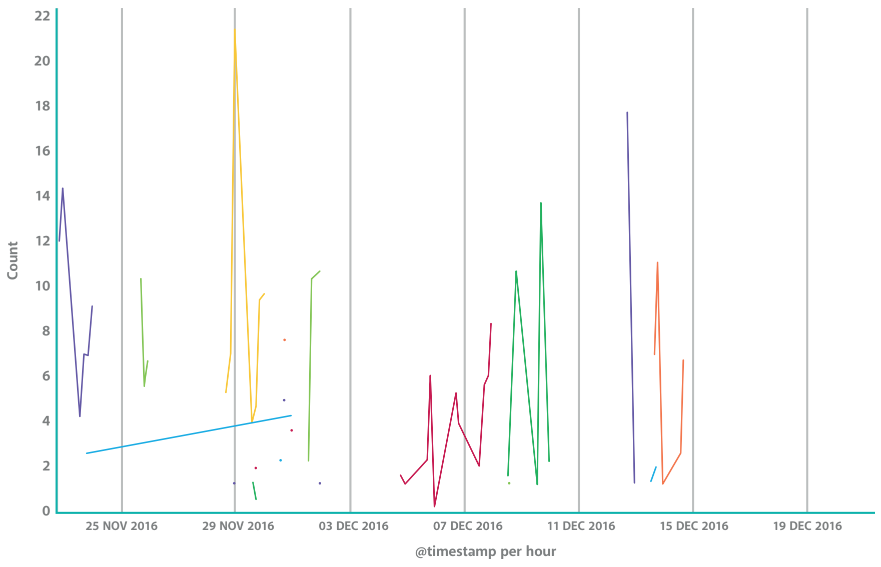
If we receive information directly from endpoints, we have a better idea of which behavior is related to a specific endpoint. For log data, we might have uniquely identifiable hostnames or we might only have IP addresses. For network data, in most cases we will only have IP addresses.

Modeling entities as single IP addresses generally works relatively well for servers that have static IP addresses and remain in the same physical location. However, the far more important scenario is the (usually) much larger endpoint estate. In these cases, DHCP is the norm and users flip between wired and WiFi connections, travel between offices, and connect remotely via the VPN. While it is true that correlation with DNS and DHCP network data can help map IP addresses to unique hostnames, this is by no means an easy task. It requires complete coverage of network traffic down to the intra-subnet layer for office subnets, which is very difficult to achieve on a large corporate network.

For example, looking at log activity from a machine account for a server over a 30-day window might show activity similar to the following. This was taken from activity for an Exchange Server:



A line of a specific color identifies a specific source IP address for logon events related to this user account. In the example above, we can clearly see a pattern of 24/7 activity from a single IP address, as we might expect to see for a server, which is a relatively easy situation to model. If this account were to be accessed from a different IP address, the activity would be easy to identify as highly anomalous.

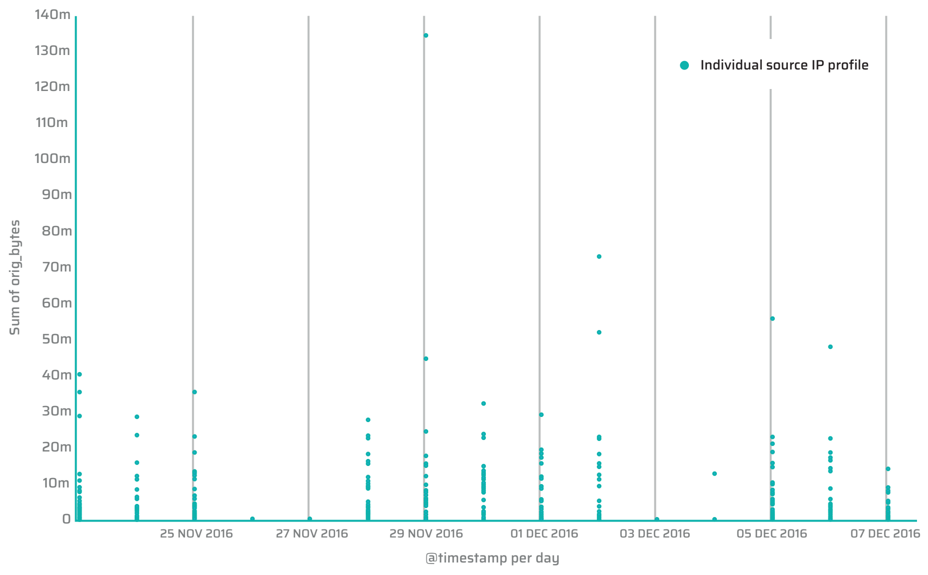


However, consider a standard business user with a laptop over the same period. As the data on the previous page shows, we can see activity from multiple IP addresses from multiple subnets and it's a much trickier situation to model. While we can see clear gaps that correspond to weekends, and also that there generally seems to be only a single IP address in use at any one time, it's still a more challenging scenario. If we see two IP addresses in use on the same day, it might be cause for concern but there could easily be a legitimate explanation, such as a user flipping between wired and WiFi networks, traveling to another office for a meeting, going home and working from the VPN that evening, or perhaps there's been a DHCP lease renewal during the day. If we knew the exact start and stop times for our data, we could ascertain that there was no overlap, but the reality is that the data we receive represents discrete events that relate to the time at which network traffic or log entries are generated, and the boundaries are somewhat fuzzy. As we can see, the problem is far more complicated when considering mobile users rather than fixed servers.

As a general rule, the more specific and sensitive we want to make our models, the more features we need to add to our data and this in turn increases the data dimensionality. If we aim to model behavior down to the individual user or entity, we also face the challenge of very small sample sizes and as we saw earlier, this is a problem for machine learning. So we are generally faced with a trade-off between a highly sensitive model that is extremely false-positive prone, and a more general model that can't detect everything but produces a smaller number of quality results, all of which are worth manual investigation.

When it comes to user data, particularly from Windows event logs, we often have full hostnames and login-type details that allow us to more reliably track a mobile user. This makes it easier to model down to the individual user. However, for network data it is far more problematic to track specific entities and we have larger numbers of possible features to consider. Modeling at the subnet level is potentially an easier approach and has the added bonus of significantly increasing our sample size, whereas modeling on a few days or weeks of data for a single entity is highly unlikely to provide a complete description of behavior. However, it does assume that the profile of any given subnet is relatively similar and unimodal. The presence of multiple profiles on the same subnet could cause issues.

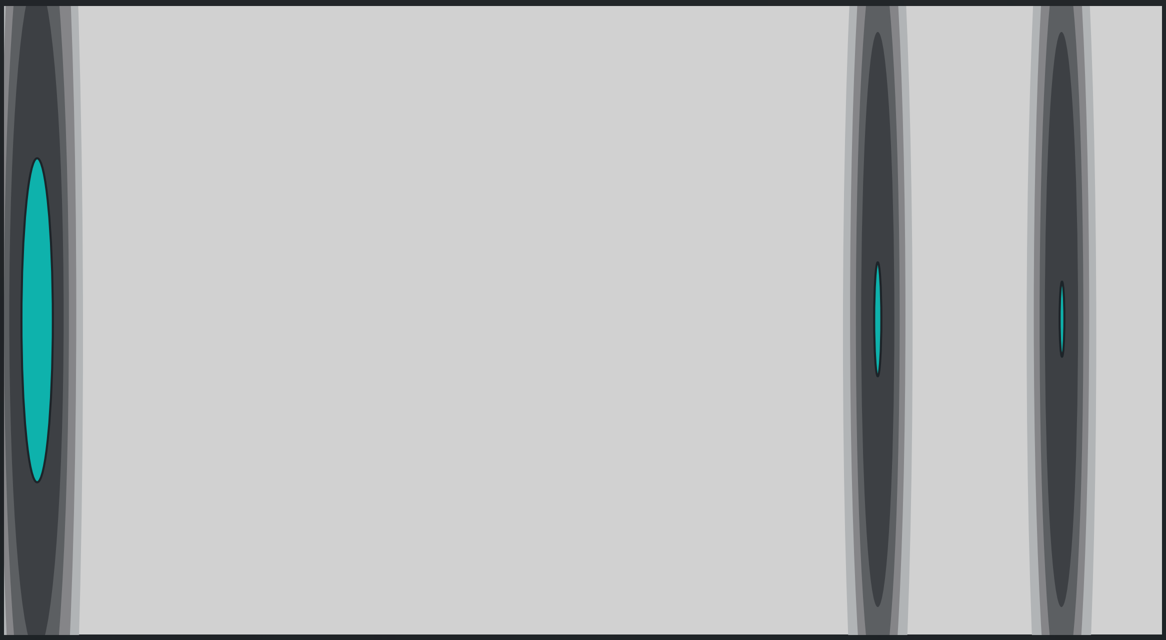
For example, let's look at internet-bound traffic volumes per day from a user workstation subnet:



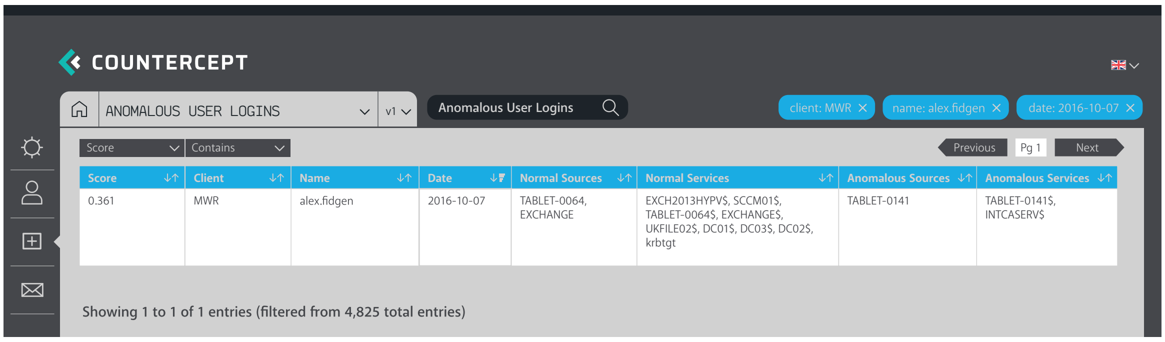
As the data on the previous page shows, most IP addresses on this given subnet are clustered around a similar region each day, with a few clear outliers, and histograms of the same data show a skewed normal distribution. As long as we make allowances for outliers in our training set, modeling per subnet over a rolling window of days should quite easily allow us to identify systems on a particular subnet that break the usual pattern and upload a comparatively large amount of data to the internet.

However, clusters will not always follow this pattern. Unless we specifically configure our solution with detailed subnet data from our network and keep it updated as network changes are made (no small task on a huge enterprise network), we could end up with mixed usage profiles being treated as the same subnet. Even if the subnet information is accurate, there could be cases where multiple usage profiles co-exist on the same subnet.

For example, consider a subnet where most systems have a fairly similar usage profile but one or two systems make huge backups nightly. This could lead to a scenario where either we learn that a huge range of data transfers is normal, or we are continually told that the backup servers are behaving anomalously, with neither outcome being desirable. Instead, we need to avoid this scenario by automatically identifying multimodal usage profiles and handling them accordingly.



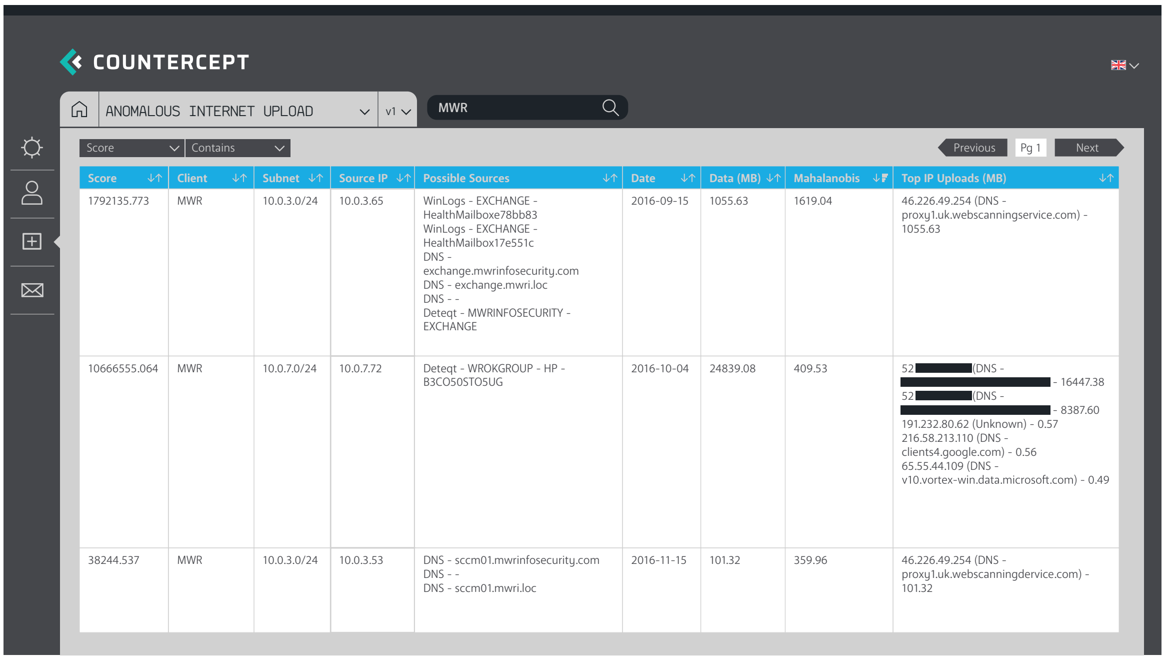
The graph above is an example of a simple model of internal data transfer on a network exhibiting this exact problem – and incorporating a pre-clustering step. In this case, three normal regions have been correctly identified from the training data and everything outside those is considered anomalous. At Countercept, we use a similar approach across various types of network data, considering multiple features, in order to identify a range of potentially malicious behavior on our clients' networks. A few examples from our parent company MWR's network are given below. Certain details have been modified or obscured for security purposes but these are results from a real global network.



Score	Client	Name	Date	Normal Sources	Normal Services	Anomalous Sources	Anomalous Services
0.361	MWR	alex.fidgen	2016-10-07	TABLET-0064, EXCHANGE	EXCH2013HYPV$, SCCM01$, TABLET-0064$, EXCHANGE$, UKFILE02$, DC01$, DC03$, DC02$, krbtgt	TABLET-0141	TABLET-0141$, INTCASERV$

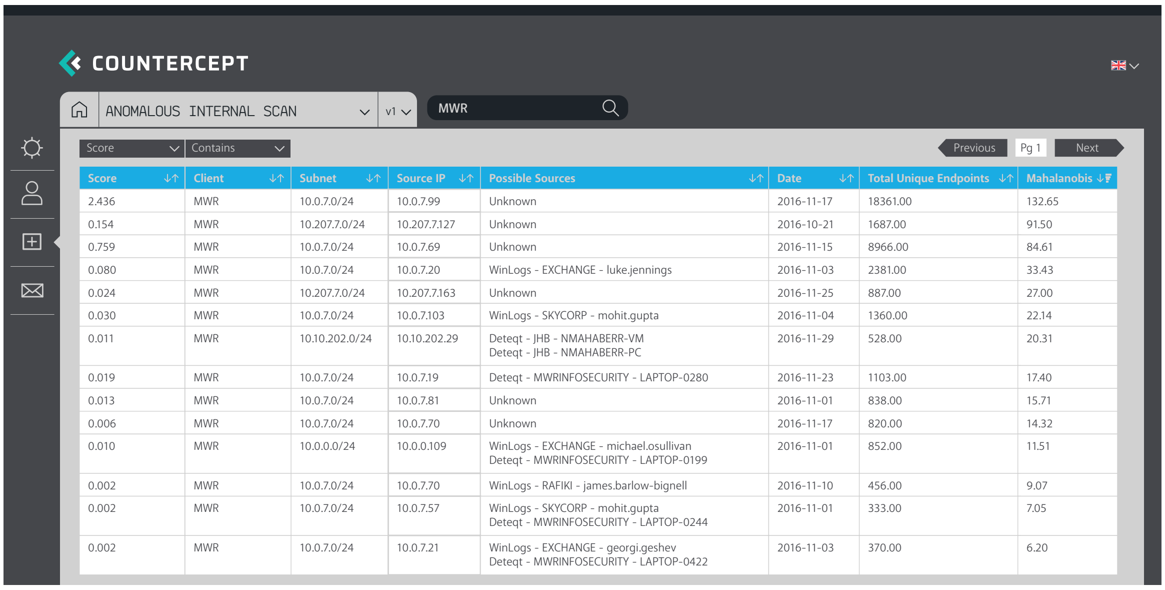
Showing 1 to 1 of 1 entries (filtered from 4,825 total entries)

Here we can see an alert being raised for anomalous user behavior. This user has a relatively predictable behavioral pattern but was seen operating from a different tablet device to access an internal certificate server. This is exactly the type of behavior you would expect to raise an alert, although in this case it was due to the user being issued with a new tablet and new client certificates being issued by the internal certificate server. In a 100,000-user enterprise environment, we will inevitably see many instances of non-malicious anomalies but the same model will also highlight malicious behavior – such as a user account being used for lateral movement, involving systems it does not normally access.



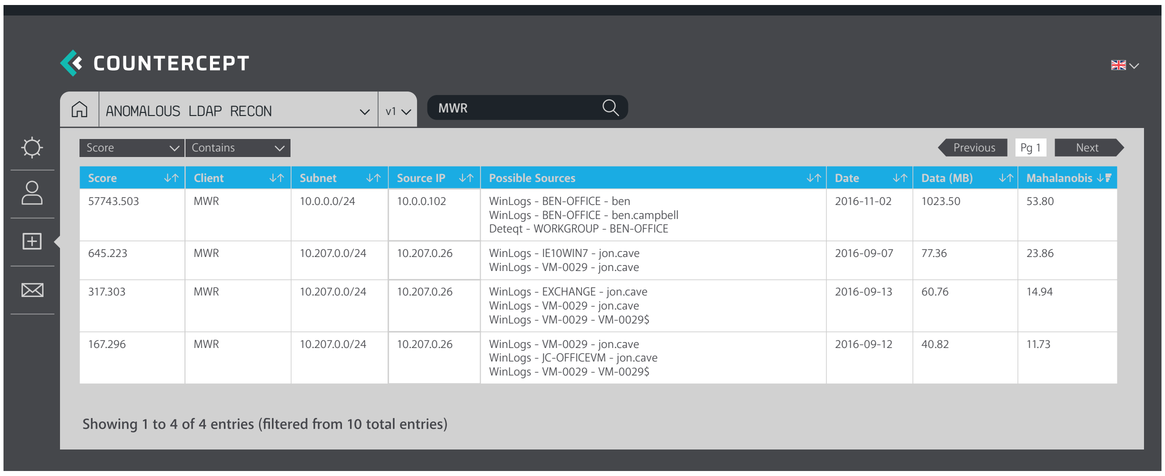
Score	Client	Subnet	Source IP	Possible Sources	Date	Data (MB)	Mahalanobis	Top IP Uploads (MB)
1792135.773	MWR	10.0.3.0/24	10.0.3.65	WinLogs - EXCHANGE - HealthMailboxe78bb83 WinLogs - EXCHANGE - HealthMailbox17e551c DNS - exchange.mwrinfosecurity.com DNS - exchange.mwri.loc DNS - - Deteqt - MWRINFOSECURITY - EXCHANGE	2016-09-15	1055.63	1619.04	46.226.49.254 (DNS - proxy1.uk.webscanningservice.com) - 1055.63
10666555.064	MWR	10.0.7.0/24	10.0.7.72	Deteqt - WROKGROUP - HP - B3CO50STO5UG	2016-10-04	24839.08	409.53	52[redacted](DNS - [redacted] - 16447.38 52[redacted](DNS - [redacted] - 8387.60 191.232.80.62 (Unknown) - 0.57 216.58.213.110 (DNS - clients4.google.com) - 0.56 65.55.44.109 (DNS - v10.vortex-win.data.microsoft.com) - 0.49
38244.537	MWR	10.0.3.0/24	10.0.3.53	DNS - sccm01.mwrinfosecurity.com DNS - - DNS - sccm01.mwri.loc	2016-11-15	101.32	359.96	46.226.49.254 (DNS - proxy1.uk.webscanningdevice.com) - 101.32

Meanwhile, the data above shows some results from an internet upload model that looks for signs of data exfiltration. We can see 1GB uploaded from an Exchange Server to a web proxy that has been identified as extremely anomalous – far more so than 25GB uploaded from a different subnet, as it's unusual to see internet traffic from internal servers, whereas user subnets tend to have much more internet activity. In this case, it was the result of an internal penetration test to check that the activity would be identified.



Score	Client	Subnet	Source IP	Possible Sources	Date	Total Unique Endpoints	Mahalanobis
2.436	MWR	10.0.7.0/24	10.0.7.99	Unknown	2016-11-17	18361.00	132.65
0.154	MWR	10.207.7.0/24	10.207.7.127	Unknown	2016-10-21	1687.00	91.50
0.759	MWR	10.0.7.0/24	10.0.7.69	Unknown	2016-11-15	8966.00	84.61
0.080	MWR	10.0.7.0/24	10.0.7.20	WinLogs - EXCHANGE - luke.jennings	2016-11-03	2381.00	33.43
0.024	MWR	10.207.7.0/24	10.207.7.163	Unknown	2016-11-25	887.00	27.00
0.030	MWR	10.0.7.0/24	10.0.7.103	WinLogs - SKYCORP - mohit.gupta	2016-11-04	1360.00	22.14
0.011	MWR	10.10.202.0/24	10.10.202.29	Deteqt - JHB - NMAHABERR-VM Deteqt - JHB - NMAHABERR-PC	2016-11-29	528.00	20.31
0.019	MWR	10.0.7.0/24	10.0.7.19	Deteqt - MWRINFOSECURITY - LAPTOP-0280	2016-11-23	1103.00	17.40
0.013	MWR	10.0.7.0/24	10.0.7.81	Unknown	2016-11-01	838.00	15.71
0.006	MWR	10.0.7.0/24	10.0.7.70	Unknown	2016-11-17	820.00	14.32
0.010	MWR	10.0.0.0/24	10.0.0.109	WinLogs - EXCHANGE - michael.osullivan Deteqt - MWRINFOSECURITY - LAPTOP-0199	2016-11-01	852.00	11.51
0.002	MWR	10.0.7.0/24	10.0.7.70	WinLogs - RAFIKI - james.barlow-bignell	2016-11-10	456.00	9.07
0.002	MWR	10.0.7.0/24	10.0.7.57	WinLogs - SKYCORP - mohit.gupta Deteqt - MWRINFOSECURITY - LAPTOP-0244	2016-11-01	333.00	7.05
0.002	MWR	10.0.7.0/24	10.0.7.21	WinLogs - EXCHANGE - georgi.geshev Deteqt - MWRINFOSECURITY - LAPTOP-0422	2016-11-03	370.00	6.20

However, by modeling numbers of systems and ports contacted, rather than data volumes, we can easily identify traditional internal reconnaissance in the form of ping sweeps and port scans. In this case, it's easy to identify even relatively conservative and stealthy port scanning without false positives, as opposed to extremely noisy traditional IDS systems that also miss stealthier scanning activities. All the above results are due either to internal penetration testing exercises or network diagnostic activities by DevOps staff. The 'Total Unique Endpoints' column in this case represents the number of IP:port combinations that were contacted.



Score	Client	Subnet	Source IP	Possible Sources	Date	Data (MB)	Mahalanobis
57743.503	MWR	10.0.0.0/24	10.0.0.102	WinLogs - BEN-OFFICE - ben WinLogs - BEN-OFFICE - ben.campbell Deteqt - WORKGROUP - BEN-OFFICE	2016-11-02	1023.50	53.80
645.223	MWR	10.207.0.0/24	10.207.0.26	WinLogs - IE10WIN7 - jon.cave WinLogs - VM-0029 - jon.cave	2016-09-07	77.36	23.86
317.303	MWR	10.207.0.0/24	10.207.0.26	WinLogs - EXCHANGE - jon.cave WinLogs - VM-0029 - jon.cave WinLogs - VM-0029 - VM-0029$	2016-09-13	60.76	14.94
167.296	MWR	10.207.0.0/24	10.207.0.26	WinLogs - VM-0029 - jon.cave WinLogs - JC-OFFICEVM - jon.cave WinLogs - VM-0029 - VM-0029$	2016-09-12	40.82	11.73

Showing 1 to 4 of 4 entries (filtered from 10 total entries)

A slightly more interesting example is the modeling of LDAP traffic specifically. Modern internal reconnaissance activities tend to avoid port scanning in favor of querying legitimate services such as LDAP, in order to gain vast swathes of information about the makeup on an enterprise network, using tools such as PowerView. LDAP traffic generated in response to group policy is relatively regular and predictable, and the majority of anomalies here have been generated in response to the use of PowerView or similar tools during penetration tests.

What are These Techniques Blind to?

Throughout this paper, we've discussed some of the problems faced when applying machine-learning techniques to attack detection. Here we consider some more specific scenarios. At the highest level, we have three key problem areas:

1. Malicious activities we can't detect because – with the data we have – they appear normal.
2. Non-malicious activities that regularly appear abnormal and generate a great many false positives.
3. Malicious activities that can't be detected because we simply don't have the data.

So let's consider two sides of the story and look at a best-case and worst-case scenario, and how they might appear in the context of some machine-learning approaches.

Best-Case Scenario

1 An endpoint is compromised via a phishing attack that uses a new variant of a known Remote Access Trojan family. It begins making noisy, regular communications with an IP address – one with which the network has never before communicated – in a distant nation.

2 This same endpoint then begins noisily port scanning the internal network, using a stolen local administrator account hash to access other workstations in a different office, until it locates a domain administrator.

3 Domain administrator credentials are then stolen and used to log in to some key database servers via the original compromised endpoint as a pivot.

4 Large multi-terabyte data transfers are then made from these servers to internet addresses, with the clear aim of stealing the contents of large databases.

We can expect to detect various aspects of the activity in this case.

Our next-gen AV solution might detect the newer variant of the known malware family in the original compromise. The noisy port scanning is likely to stand out significantly from normal network traffic and should be easy to identify as anomalous, as should the sudden use of a local administrator account to log in to large numbers of workstations. The domain admin logins to the key database servers from the original compromised endpoint are likely to be seen as anomalous, since the compromised user was not a database administrator.

Finally, the multi-terabyte data transfers should be easily identifiable as anomalous, as they are from IP addresses that do not usually communicate with the internet at all. Additionally, in terms of data visibility, most of these activities are likely to involve network traffic crossing key network boundaries – where we are most likely to have network sensor coverage.

Worst-Case Scenario

1 An endpoint is compromised via a phishing attack using a bespoke Trojan while the user is working remotely from home. The user has been targeted specifically and much of the information desired by the threat actor is in Office documents and emails stored locally, which are immediately stolen and do not represent large data volumes. Google Chat is used as the command and control channel and only makes regular connections when in active use, otherwise remaining dormant.

2 When the now-infected endpoint is reconnected to the internal enterprise network, the Trojan is used to send a new phishing email internally, targeting a member of the desktop support staff and consequently compromising their endpoint. This account can be used to gain remote access to any desktop on the estate.

3 Specific LDAP queries and internal Wiki access are used to find additional information concerning specific employees of interest to the threat actor, pivoting via the newly compromised desktop support staff member's endpoint.

4 The desktop support credentials are then used to remotely access the workstations of the targets of interest, select documents, and steal emails. Further Trojans are installed and the threat actor then goes quiet – until at some point in the future they decide to pull more documents remotely.

In this case, our machine-learning models are of much less use.

For a start, much of the activity occurred outside the network, or internally on workstation subnets where there's unlikely to be proper network sensor or log coverage compared with key network boundaries and critical servers. Our next-gen antivirus product is unlikely to highlight any concerns, as the Trojan used is completely bespoke and has probably already been tested to ensure it functions correctly with commercially available antivirus technologies. Even if the command and control channel could be detected, is Google Chat really going to be seen as malicious? It's likely that many systems on the network are already communicating with this service.

Initial lateral movement is then achieved by an internal phishing email. At network level, the compromised endpoint will simply be talking to Exchange as usual and not making any direct communications or logins to other systems. The end result is a newly compromised member of the desktop support staff. At this point, highly selective internal reconnaissance makes light use of services in a way that's likely to be seen as normal. It's probably usual for the desktop support staff member to access employee workstations on a daily basis, and in a manner that would only be predictable if tied up with user phone calls and ticketed support requests. The use of this account from this endpoint to remotely access a small number of employees' workstations is unlikely to be seen as anomalous – even if network sensor and log coverage is available for those subnets, which it may well not be. Finally, the quantity of additional data stolen is relatively small, and well within the normal profile of those systems as a result of internet browsing. Trojans are now in place and credentials obtained such that, in future, the threat actor needs only to select new documents and emails that interest them, with no need for further activity; all will appear to be business as usual.

Conclusion

Machine learning is a powerful tool of great benefit when used correctly to help solve specific security problems.

However, it is by no means a silver bullet able to address the entire field of attack detection. There are significant limitations that affect machine-learning approaches; in particular, enterprise networks are highly anomalous environments and, since anomalies are common, so are false positives.

On the other hand, not all malicious activities are anomalous from the perspective of the data sources normally available to machine-learning-orientated attack detection systems, so false negatives are also common. Finally, even for the specific problems for which machine-learning approaches are useful, they cannot entirely replace the human operator; they simply enhance their capabilities. In Countercept's experience, anomalous results produced by machine-learning techniques, while very useful, are often the most time-consuming and challenging to identify as malicious or non-malicious.

For an effective enterprise-scale attack detection system, it's vital to have the right data sources across endpoints, logs and network, appropriate use case-orientated analysis techniques for these data sets, and highly skilled, offensively trained staff to interpret and investigate properly. Machine-learning techniques can be used to enhance analysis techniques in some areas of attack detection, but there are many areas where other approaches are more effective – and as always, the human element is indispensable.

Importantly, the available data is critical, since no machine-learning model can learn anything useful unless it can be determined from the data itself. Hence any vendor promoting a machine learning-based product with the message that it can automatically solve all your attack detection problems, based on a single data source, should be treated with healthy skepticism, as has been the case with any supposedly 'silver bullet' solutions throughout the history of the security industry.

COUNTERCEPT

COUNTERCEPT

info@countercept.com
www.countercept.com

Connect with us