Intrusion Along the Kill Chain

Part I: On the State of things

Intrusion detection systems have been around for almost two decades as a way to attempt to fill in the gap for when preventative security fails. To practitioners of this arcane art the problem of accurately and consistently detecting attackers among the sea of noise feels no less worthy or indeed less difficult than that of historic searches for Atlantis or the Holy Grail. Years into the problem we have much technology to show for our work but sadly few approaches that work reliably, especially against the face of highly sophisticated modern attackers and their shifting behavior. This paper is an attempt to investigate some of the shortcomings of these classic approaches and offer some new ideas to approach this problem.

The Industry

Gartner estimates the information security market as having a total spend of about $35 Billion[1] for 2011. The exact amount of intrusion detection spending proves difficult to size accurately. However, IDC has forecast[2] $1.5 Billion being spent on SIEM systems alone in 2012. Regardless of precisely the way you count the numbers, it is clear that billions are being spent annually.

Enterprises of all kinds are buying the latest security appliance and SIEM products in an attempt to solve their Intrusion Detection problems. But does intrusion detection work? Two of the best technical reports on the state of the security industry are the Verizon Data Breach Report and the Mandiant M-Trends report. Both of these present data to help illuminate this subject.

Verizon

The 2012 Verizon Data Breach Investigations Report[3] is one of the best and most comprehensive analytical reports on the state of the security industry out there. It combines data from a multitude of sources including Verizon's own Incident Response work, as well as that of a number of Law Enforcement agencies from different countries in an attempt to distill trends and intelligence from the history of data breaches and compromises that they have collectively observed. The Verizon report does a good job of trying to include data from a cross section of businesses, from small to large.

---

[1] http://www.gartner.com/it/page.jsp?id=1844114
[2] http://www.qualys.com/docs/idc_VM_forecast2012.pdf
[3] http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2012_en_xg.pdf

First of all, and not surprisingly, 7 out of 10 targeted attacks leading to data breaches were against larger organizations[4]. These are the companies that we would expect would typically have invested in intrusion detection systems. So how well do our intrusion detection systems work at larger companies?

Average time to until an intrusion is discovered is one proxy for the efficacy of intrusion detection systems. According to Verizon, in 2011, half of intrusions to larger organizations took *months* or *years* to discover.

These same attacks that on average take this long to discover are executed on a dramatically different timeframe. The time between initial attack to initial compromise for larger organizations happen 71% of the time in "minutes" or less.

How long, after compromise does it take for our attacker to find and exfiltrate data? The news here is a little better as 75% of the time it takes days or longer in larger organizations. This is a great opportunity for practitioners of intrusion detection, as this is the time that attackers are rifling around your company looking for the data, users or systems they are after.

Unfortunately the numbers get much worse when you look at how compromises get discovered in the first place. Again, if we look at larger organizations, we find that half of discoveries are from 3rd parties. The remainder are discovered internally, but not by ways you might expect. About a third are discovered by non security employees noticing something is 'weird' and reporting it. Worse still, purpose built 'fraud detection mechanisms' are responsible for 5% total of detections of intrusions against larger organizations, just behind the 8% for routine manual log review.

This bears repeating: Intrusion detection systems are, on average, worse at detecting modern threats than humans doing adhoc log review and almost 6 times worse than your own users noticing 'suspicious behavior'. This is truly a sad state of affairs.

Mandiant

Mandiant is a company that also publishes an annual report. Their report is of a high quality as well but it is also somewhat more focused than that of Verizon as it primarily addresses advanced and targeted attacks. Their clients are almost exclusively large enterprises who have been compromised. In their most recent report[5] on their 2011 caseload they found that *only 6% of advanced intrusions were detected by internal processes*. Of course of that number I would guess that only a fraction was detected by a purpose built intrusion detection system.

---

[4] Larger Organizations are defined by Verizon as 1000 employees or more.
[5] http://www.mandiant.com/resources/m-trends/

If these numbers are to be believed, and I suspect that they are, then the utility of intrusion detection systems against modern targeted attacks couldn't be much worse. Is there a way out?

Part II: A glimmer of hope

The state of intrusion detection tools and best practices against modern targeted threats is so poor that we need to do nothing less than reinvent the field. In this section we discuss some ways to do just that.

First off, let's start at the beginning. What is intrusion detection anyway, and what problem are we trying to solve?

Intrusion detection is the art and science of finding actionable deviations between normal behavior and attacker behavior.

Perhaps another, more practical definition is:

Maximize your chance of getting lucky.

According to the data, detecting an intrusion is a low likelihood event. Let us now discuss how we maximize our chance of getting lucky and push the chances more in our favor.

Intrusion events are not binary

Often people will install an IDS sensor and then go about "tuning" it by turning off everything that makes an alert. This is the "*false positive fallacy*." This fallacy stems from thinking that only those things that rarely alert are of value; that intrusion detection is about generating a single perfect, actionable event.

The right way of thinking about intrusion events is to start by realizing that there is actually a continuum of event types. You can think of events as having a score that reflects how actionable they are. On the low extreme you can think of raw log data, pcap data, netflow data and so on as a 1. There is little dispute that it is an important part of the intrusion detection story, but the data itself isn't actionable without at least some kind of basic analysis. On the other extreme would be an event that is extremely actionable. These tend to be rare but they do happen. "Every time I see someone going to this dns name I want to be woken up in the middle of the night." These two extremes are familiar territory. Many security programs are based entirely on these two extremes. People collect logs, and try to make "wake me up in the middle of the night" events out of them.

The problem with this model is that there is a whole lot of interesting stuff between 1 and 10 that gets largely ignored by this common approach. The following list illustrates a few common examples of these "events in the middle" or low confidence events.

1. Snort event for a high confidence dns domain
2. Malicious PDF sent to a user
3. Logins for the same user from disparate geolocations
4. Netflow based alerts for known bad ip addresses
5. Specific Registry Modifications (i.e. Run Key)
6. Antivirus Alerts
7. Snort event for a blank useragent
8. Windows RDP Successful Login Event
9. Snort event that alerts on all encrypted outbound traffic
10. Pcap data, Raw Logs, Netflow etc.

When you change your mindset to allow these lower confidence events into your security program, you realize that a large portion of the intrusion detection job is basic instrumentation of your environment. This shifts the problem away from having to create only actionable events and allows for a much richer set of data to be part of the intrusion detection analysis later on. What most people do is try to instrument an event, and then spend a lot of effort trying to make each event a 1.

Too many people shy away from logging basic data about what users, machines, processes and so on are doing in lieu of chasing the silver bullet alert that will tell them for certain if an attacker is on their network. Instead, spend time thinking and instrumenting every possible indication of suspicious activity you could generate, regardless of how useful or confident you are in the utility of that data. You'll find this approach vastly easier and in the end more effective.

The Event Pipeline

Once you have taken steps to instrument your environment your next goal is use this sea of instrumented data to eventually build up to higher confidence correlated events. The event pipeline is a conceptual framework for doing this. The pipeline is illustrated in Figure .

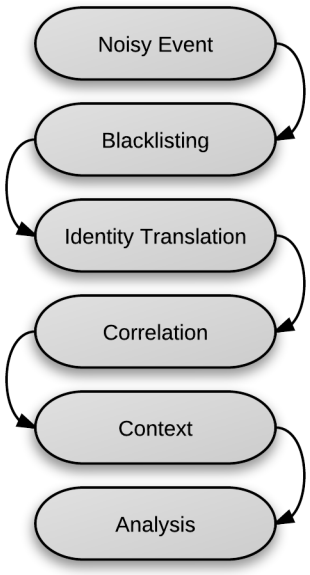

Figure 1: The Event Pipeline

Blacklisting

The first stage that our noisy events go through is known as Blacklisting. This is the stage where we remove known false positives from a signal. Unlike a typical approach to IDS, we do not squelch the event altogether. Instead we remove the known false positives by comparing the event to a list that we maintain and update over time. Let's say you wanted to look for all HTTP connections to the internet that have a blank useragent. Perhaps you write a snort signature that detects this network behavior. When you look at the results you discover a number of false positives. For example someone has a Fitbit and it has a blank useragent when it uploads its' users' data. Perhaps another user is running an

architectural design program from Taiwan that has a blank useragent when it looks for updates. After you blacklist these 2 examples from the original event, you've not got something that went from a confidence value of 2/10 to a 5/10, simply by removing the known noise.

Identity Translation: The apples and oranges problem

At this stage we have now instrumented our environment with dozens, hundreds or thousands of events of various confidence values. We have events derived from host data, network data and log data. In order to be able to view these different types of events through the same lens, we need to do identity mapping. In practice this is not easy, but having this capability is a fundamental requirement for an effective intrusion detection program. A stable and mature machine and user inventory system is a good starting place. One way to go about this is to map all events back to a user identity, but any approach that gets all events to have a common identity is fine. As events go through the pipeline, the identity translation stage is responsible for tagging events with this common translated identity. Let's take our example of the snort alerts for a blank useragent. As each of those alerts traverses the pipeline, the event should be tagged and attributed to each user account currently active on the system that sourced the traffic in question.

Correlation

The Attack Plane

Correlation is a complex topic that requires the introduction of a number of new concepts. The first of these is called the Attack Plane. The best way to think about an attack plane is that it is a way to conceptually group our sea of intrusion events together. The overall goal is to choose a grouping model that allows us to combine as many emanations from a real attacker as possible.

The simplest case of an attack plane is that of a specific host on your network. Let's say an attacker compromises one of your machines and installs malware. The malware modifies portions of the systems' registry and makes a network connection using a blank useragent. You have instrumented the useragent detection as we discussed above and have a fairly low confidence event (5/10) after blacklisting that triggered for that. The event has an internal source ip address of your host. You also have instrumented your registry such that you happened to get another low confidence event for the specific registry modification that occurred from the host.

In the case of this trivial example, we can choose an attack plane that is simply a host ip address on your network and stack the two events on the 'plane'. While these two events in isolation are interesting but not quite interesting enough, when viewed through the lens of the attack plane we can now see an opportunity for correlation.

Attack plane grouping is a great way to do event correlation and it allows you to gain value even from low confidence events. The problem with the trivial model above is that it doesn't capture attacker behavior completely. Our goal is to find an

attack plane that captures as much of our attackers' behavior as possible. This will allow us to maximize the number of events that we can include in our grouping.

The major source of limitation for the simple host-based model above is that modern attackers can often limit activity on a single host. Instead they opt to use a number of hosts and accounts in your environment as part of their attack. We can make a more intelligent choice of an attack plane if we think a little deeper about what attackers are actually doing.

The Kill Chain

So what do modern attackers do, and how can we accurately model their behavior? The Kill Chain is a conceptual model originally used by the US Military, and discussed in the context of information security by Huchins, Cloppert and Amin at Lockheed Martin.[6] The idea behind the Kill Chain is that there are a series of stages an attacker necessarily goes through in the course of setting up and executing an attack. Figure 2 shows an adapted version of the Kill Chain from the original work. Changes that have been made are primarily to add more focus to the time that an attacker spends within a given enterprise to make it clearer what our opportunities might be for detection.

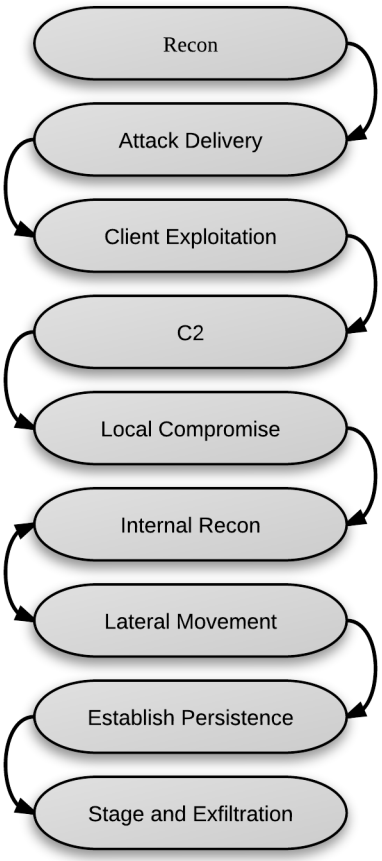

Figure 2: The Detection Kill Chain

If you have studied modern attacker behavior, the stages of the Kill Chain should be familiar. Recon is also known as "passive recon" where the attacker maps relationships between individuals, sets up C2 nodes and so on. Delivery is where the attacker will send the payload, for example in the form of a spear phishing email. Client exploitation is the detonation of that payload on the client, often exploiting weaknesses in browsers or 3rd party apps. The compromised host then engages in command and control communication. Next an attacker will usually grab credentials from the machine and possibly escalate privileges if necessary. The attacker will then set about discovering what systems are available internally, and what accounts and systems they need to spread to next to get to their target. Once they have determined this, they will spread laterally from machine to machine, and sometimes between accounts until they have privileges to access the data they need. At some point along the way, attackers will often make sure to leave one or more ways to

[6] http://papers.rohanamin.com/wp-content/uploads/papers.rohanamin.com/2011/08/iciw2011.pdf

gain re-entry into the network in case their original access vector is discovered. Finally once they find their target they will grab the data and send it out for offline analysis.

The Kill Chain model has a number of nice properties. Although it does change over time, it is far less volatile than the specific tools and techniques that an attacker uses which can change rapidly. It is this stability of this model that we will exploit in doing our correlation.

The Kill Chain as an Attack Plane

So now that we understand that an attacker's behavior can be described conceptually in the form of a series of stages, let's return to our discussion of attack planes and event correlation. Recall that we are attempting to optimize our chance of "getting lucky" at finding an intrusion by grouping events together in more and more sophisticated ways. The kill chain represents another way to group our low confidence events. We started of by simply grouping events that pertain to a single host in our environment together. Now, as we instrument our events, we also annotate them with the stage in the kill chain. Due to our identity mapping in a prior stage, we now have everything we need. First we group events that share a common identity. Next we try to observe clusters of events for the same identity that inhabit multiple stages in the kill chain.

I should note that the BotHunter[7] research project implements a simplified version of this same model. It uses the concept of instrumenting stages of malware behavior to more accurately detect malware.

Let's take an example to try to make this clearer. You have instrumented registry changes that indicate a host might be compromised. You also have an event for the blank useragent http connections we discussed. Finally you have a tool that creates low confidence events indicating access to some internal sensitive documentation. By themselves none of these events could ever solely indicate a compromise worthy of further investigation. However, let's apply our kill chain lens and follow the methodology laid out in our event pipeline. Each low confidence event would go through a process of individual blacklisting to remove egregious false positives. Next we would apply identity translation to each one. In the first case, registry changes, we might have only a hostname of the machine applied to the event. The translation system would apply identities to the event of all users that were seen on that machine at that time as well as the internal IP addresses of the system at that time. In the second case the event would come into the translation stage with only a source IP, which again you'd have to translate into usernames and apply to the event. In the final case we already have a username so we could apply IP addresses and hostnames to the event if desired. Now we have reached the correlation stage. The producer of the event would tag them with stage in the kill chain it belonged. The registry event would be 'client exploitation', while the blank useragent would be "C2" and the internal documentation events would be "internal recon". It is now

---

[7] http://static.usenix.org/event/sec07/tech/full_papers/gu/gu_html/

pretty easy to see how the kill chain can relate these events. Before they seemed like useless events that would generally be ignored. You simply line up the usernames and note that this is a correlated event with 3 possible stages seen in the attack progression.

Are there weaknesses to this approach? Absolutely. It is hard to build an accurate identity translation system and attackers change identities that can be difficult to follow. But likewise there are a large number of ways you can make this model vastly more sophisticated once the basic foundation is there. You can incorporate a much higher volume of events from your environment than you previously were just throwing away. Difficult to incorporate systems such as anomaly detectors fit into this model comfortably, as they are just another source of valuable, but often low confidence events. You can also experiment with alternate strategies for defining attack planes to find optimal fits against the event data.

Context

Going back to our original event pipeline, there are a few more stages after correlation is complete. When you finally have a list of high confidence events that you've produced we want to apply context. Context, also known as situational awareness, is additional data that you bring to bear on a given high-confidence event to aid manual analysis. The goal of context is to enable a human analyst to spend as little time as possible trying to make the determination to escalate an intrusion event. But what is context and where do you find it?

Your vendor products are an ecosystem

Most people don't think about their vendor products as being a major source of event context. Instead of just producing streams of alerts, many systems are valuable for the insight they offer into an alert produced by another subsystem. Let's take our example discussed during the correlation section above. Recall the case of the snort alert for a blank useragent . We had correlated it with alerts about documentation access and registry writes and found a possible kill chain match. Now that we are in the context stage with this correlated event, we can find ways to apply more information about the environment to this event to aid manual analysis. To that end, perhaps we add data from a machine inventory system to tell us about the running processes on that system. Perhaps we can create a simple visualization about the last 24 hours of authentication events for the user in question. Perhaps we can add some information about what patterns of network behavior the user or host was seen to have done around the time of the intrusion. Since we should have a relatively low amount of events at this stage ready for manual analysis, applying environmental context to each one should be no problem at all.

Analysis

Analysis is the stage where we finally have some things worthy of a human being's attention. In an ideal world, our poor human analysts are only looking at events that have a high quality score so we aren't wasting people's time with low confidence

events. Aided by our applied context, analysts would have everything they need when presented with the correlated event to make a decision.

Conclusion

Our current batch of Intrusion Detection products are clearly insufficient against today's targeted modern threats. This paper has laid out a series of models for how to rethink the problem from the ground up. Stepping back from our need to only instrument highly actionable events is the first and most important realization that we have outlined. The event pipeline provides a framework to understand how low confidence events can and should be assimilated into an effective intrusion detection program. We introduced the concept of the attack plane as a way to look at how to group events during the correlation stage. We present the kill chain as one highly sophisticated way to apply attack plane grouping to our events to better correlate them. Finally, we discussed the notion of applying context to our events so as to improve the quality of our analysis.