#127 - How to Stop Bad Guys from Staying on Your Network (with Kevin Fiscus)

[00:00:00]

[00:00:00] **G Mark Hardy:** Hello, and welcome to another episode of CISO Tradecraft, the podcast that provides you with the information, knowledge, and wisdom to be a more effective cybersecurity leader. I'm your host, G Mark Hardy, and today I've got a special guest today. Kevin Fiscus, who's been with us before, if you have good memory, you'll remember that he was with us on episode number 43 on cyber deception.

Well, Kevin's been doing a lot more thinking since then. He's got some wonderful ideas that I'd like to share with you. So Kevin, welcome to the show.

[00:00:38] **Kevin Fiscus:** Thank you G Mark. I appreciate it. I guess this time, people have to suffer with looking at me in addition to just hearing me, so hopefully that won't drive everybody away.

[00:00:45] **G Mark Hardy:** And that's a good point. So if you're listening to us on our podcast, remember we have a YouTube channel. Don't forget to subscribe to YouTube. It makes sure that our numbers get up so we can get a little bit more control over our, our videos. And if you're listening to us on your favorite podcast channel, please continue to do so.

But [00:01:00] alike here and there would help us out a little bit because it helps us reach other listeners as well. So, Kevin, we work together for almost a decade as SANS instructors, and so we go back a long ways. But you've been doing a lot of stuff. Tell me a little bit more about maybe your background and, and what you're doing now.

[00:01:14] **Kevin Fiscus:** Just a, a brief overview. I've, I've been in the IT industry since 1990. Started off doing mainframe operations for the US Air Force. So I got to spend some time in in Misawa Airbase Japan. And that was a lot of fun and got out the Air Force and spent time doing a variety of very strange things like trying to sell knockoff perfume and overly expensive vacuum cleaners, but ultimately ended up getting back into it for a while.

Did kind of generalized IT stuff for a number of years, being the computer guy for a couple of small businesses. But then ultimately in early 2001, I want to say really shifted my focus to information security. And I've been working as an information security consultant for a [00:02:00] number of value added resellers deploying security technologies, but then also doing security consulting work.

And that's really what I've been focusing on, is trying to help organizations. First of all, understand where their security program is today. Like what, what does it look like for real today? And then in all honesty, many organizations already kind of know, but they have difficulty in prioritizing the decisions that they need to make.

Right? We see all these problems. We we're overloaded with vulnerabilities and new threats and new technologies and, and user errors. Where, where should we focus? So I've spent a lot of time. Over the last, let's say 10 to 15 years, really focusing on trying to help organizations understand where the strengths of their security program are, where the weaknesses of their security program are, and then trying to help prioritize what should be done next.

Whether that be identifying the low hanging fruit or more of the [00:03:00] strategic type of things, and. As part of that, I've, I've kind of self-analyzed. I've taken a lot of time going, we've got a problem when it comes to security, right? All of the metrics that are out there in terms of things like dwell time and cost of data breach and all this show that the bad guys. By and large are winning in a lot of ways. It's not universal, but they are winning in a lot of ways. And one of the biggest areas that I've seen is that bad guys compromise our networks and stay on there far too long before we manage to detect them. So, I started going down that trail of, well, why is this the case and what do we need to do to solve that problem for real?

And oddly enough have recently come across a some, some what to me are startling revelations of, wait a minute, we might be doing some things substantially wrong. We might [00:04:00] actually even have some of the wrong goals. And so that's kind of what I've been working on recently.

[00:04:06] **G Mark Hardy:** Now that's rather interesting I think for let's unpack some of that. So for CISOs who have to deal with. A, what do we have today in our environment? B, what are the bad guys doing? And then C, what do we need to do in the future? It says that you're looking at a lot of that. Well, one of the things, let's start out and with the idea of about the attackers and being in there and things such as that.

And so in our discussion previously had mentioned about things like the Ponemon cause. So Larry Ponemon study last year identified that the average time to identify a breach is 207 days, but. FireEye's Mandiant trends. Now, of course, Mandiant is no longer FireEye is, so the median dwell time is 28 days. So let's unpack that a little bit.

Of course, 28 days, but 207 days to breach it. So there's a pretty good chance that either the bad guys have been in there, done what they were going to do, taking their actions and kicked out. But then how long does it take for these attackers [00:05:00] to actually break out and start doing damage?

[00:05:03] **Kevin Fiscus:** Yeah, so it's kind of interesting when we look at the when you contrast the Mandiant M-Trends report. And the Ponemon Institute cost of a data breach report. You have wildly different numbers, right? time of 28 days versus 207 days. I am not a statistician. I don't pretend to be, nor do I really want to be a statistician.

But the thing to me that's interesting is the M-Trends report is based only upon incidents that were investigated by Mandiant and my general anecdotal evidence is that organizations that contract with FireEye slash Mandiant tend to be more robust with respect to their overall security program. So we can look at the M trends number almost as the, maybe not the best case, but the better case.

Whereas the Ponemon study just studies hundreds of organizations in over a dozen verticals. Across the globe. I think that that represents more of the average case. [00:06:00] So the way that I tend to kind of distill it down is that regardless as to what what study you look at, the amount of time that bad guys remain on our networks can be measured in weeks to months or at least the time before they're initially detected.

Now, what's really scary about that is if you look for example, just recently, the the CrowdStrike, excuse me, the CrowdStrike Global Threat Report just came out. For 2023, they show what's called the the average breakout time, and that's defined as the time between initial access and lateral movement.

So it's the time that the bad guy instills themselves in our networks, elevates their permissions, and then starts moving 84 minutes. Is the average breakout time. So what it means is bad guys get on our networks 84 minutes later, they're moving laterally, and we don't find out about them for weeks to months, 28 days to 207 days, somewhere in that ballpark.

[00:07:00] And that is a s that that is a horrifying prospect because think about this, how much time does it take for a bad guy to do catastrophic harm?

[00:07:11] **G Mark Hardy:** Not a whole lot of time at all, and in fact, if you take a look at. Our general approach that we'll do with, let's say the NIST cybersecurity framework. Identify, protect, detect, respond, recover. We're already talking about detection, but what if we, let's back up a little bit. Let's go back to the protection level.

Why? Why aren't we be able to protect effectively? What's, why do these controls not working?

[00:07:31] **Kevin Fiscus:** So, So there are a couple of reasons for, for protective controls not working. There are some really obvious ones, right? Any given protective control could be, could suffer from a vulnerability, right? It, it is possible that they're coded by human beings. So you have this potential for a vulnerability.

So a bad guy could bypass a tech, a technical control because they exploit some vulnerability. Great. But there are other reasons as well. First of all, [00:08:00] that protective control may not actually be implemented in an optimized way, right? It, it could lack visibility, it could be placed in the wrong location, it could be configured incorrectly accidentally, or just may.

Maybe it was configured on purpose. But it, the, the person who configured it didn't understand certain risk factors, things along those lines. So those are, are some kind of almost technological reasons. Another really simple reason why protective controls. Sometimes fail is that a very common avenue of attack is identity based, right?

If you look at even things like Zero Trust, which I'm a huge fan of, I think zero Trust is an amazing concept. But what happens if an attacker within even a well-implemented zero trust environment gets a valid identity? Right. If you have a valid identity and you're using valid protocols, if you're living off the land, so to speak, there's no reason for the protection to stop you.

And [00:09:00] that becomes a problem. All of these things are really well known. The one thing that I don't think a lot of people pay a lot of attention to is what I refer to as resistance. The idea is that, When a bad guy attacks, they don't succeed. For the most part. The first time they try something and it fails because a protective control has successfully stopped them.

But when that protective control does successfully stop the bad guy, that creates resistance to the bad guy. The bad guy can then look at that and go, all right. I can gain some intelligence about the target environment by understanding the resistance. In other words, the bad guy runs into our environment and they hit a wall.

Well, they know that in that type of interaction, they've hit a wall. So what do they do? We haven't caught them in any real way. We don't know who they are, so, All they have to do is try [00:10:00] again. Maybe they come back from a different IP address, but the fact of the matter is protective controls almost universally mean that the bad guy can come back and try again and again and again.

And every time they do that, they learn more about what works. And what doesn't work. It's kind of like, and I can't quote it, but I'll paraphrase the, the quote from Thomas Edison when he was trying to invent the light bulb and, and he talked about the number of times that he failed, and he said something along the lines of, well, I didn't fail that many times.

I just succeeded in figuring out which things didn't work.

[00:10:34] **G Mark Hardy:** It's kind of like a ctf, right? Where you go ahead and you try to capture the flag and this doesn't work. This doesn't work. You eventually either discover or stumble upon something that does work because you've gained some knowledge along the way.

[00:10:47] **Kevin Fiscus:** Yeah. And, and what's even worse is that, the way that I describe it is protective controls tend to be static and uniform. And what I mean by that, I'm not talking about the fact that you can, I mean, you [00:11:00] can update and improve your protective controls, but in the duration of any given attack, your protective controls do what they do, and they don't do what they don't do.

But they also tend to be uniforms. So if you have like an EDR platform, you tend to deploy that same EDR platform throughout your entire enterprise. You don't take 12% of your systems and deploy one E D R platform and 32% and deploy another one. So what that means is through that trial and error process, once the attacker figures out how to evade one instance of the EDR platform, they've effectively evaded every instance of that EDR platform.

So what what it boils down to is, The bad guys have a virtually infinite number of tries to get it right. Vulnerabilities, misconfigurations, identity problems

mean that they're going to find some avenue that does work, and once they find it, that method probably works universally throughout the entire environment.[00:12:00]

Now, all this means this is not. In, in my opinion, this isn't even necessarily a bad thing, it's just a thing that we need to be aware of. And what it means is we need to adhere to the age old paradigm of protection is ideal, but detection is a must. We can't rely exclusively on protective controls to work for all the reasons we've talked about here, and probably others.

So we have to be able to effectively and rapidly detect bad guys on our network so that we can mount an effective response in a timely manner so that we don't suffer the cost of that breach.

[00:12:41] **G Mark Hardy:** So if we look at it, then some of the issues with regard to our protective control. Roles, as you had said, just kind of trying to summarize here, is they tend to be static in uniform, so we're going to deploy the same EDR every place else. Attackers have the feedback loop, which you call resistance, allowing them to gain information just through black box testing, if you will.

I don't know what's in [00:13:00] it, but I get this response. And then after a while, either because of incorrect configuration or misconfiguration or maybe somebody deliberately put a little bypass in there for testing, they forgot to take out an attacker, potentially can do that. And then as we start out looking at some of those stats where the dwell time could be as long as a month, but yet an attacker gets, what, 84 minutes to go ahead and start their lateral movement.

So it seems like by the time we detect somebody in our. Enterprise, they've probably already done everything they're going to do. And if, if we're at the median and it's already game over, and yet somehow we go to general quarters, we start fighting a response, we try to kick the bad guys out, et cetera.

But I guess the question is, what causes some of these attacks to go? Undetected and allow an attacker to be in there for weeks or months instead of setting off an alert. I had something a couple weeks ago in one of my environments that I work as a CISO, where I had written [00:14:00] some rules that were kind of unique, but they triggered and one fired off, and it's like this.

This doesn't look right. So I went in there and contacted the user. It was like 1:00 AM in the morning when the time zone that person was in and said, Hey, did you use a wifi hotspot in such and such a location? And he said, well, I was

on international travel. I'm now over here in the Middle East. It's like, yeah, you ought change your password.

And a little bit later, after going ahead and trying to do some corrective things, it turned out that I got a threat Intel report saying the IP address from which there was a valid login.

[00:14:30] **Kevin Fiscus:** Mm-hmm.

[00:14:31] **G Mark Hardy:** Because he did get Phish for his credentials was associated with a known APT. good news was it was an at APT that was kind of, well, a lazy one if you will.

They got in there and he was one of our third party folks, so he didn't have MFA, although it did give me the political ammunition to go to the executive committee and said, we need MFA for all of these third parties, not just your employees. Because look what happened. A bad guy got in B, kicked him out pretty quickly.

But the point is [00:15:00] that the shields were down for a short period of time. So for us as CISOs, what we want to think about is can we set up these detect rules initially that they're really not prevent because we got prevent set up, but as we're saying, they get passed. So what is it about detection that either works or doesn't work and things that we can do better?

[00:15:19] **Kevin Fiscus:** Sure. So I'll start off with that. Most of what I said about protection also applies to detection, right? In other words, you could have detection set up incorrectly. You could have a vulnerability that the attacker could exploit that allows them to, you know, go under the radar of the, you know, the i d s or what, whatever the deal is.

So in other words, you could have technical flaws. You could have configuration flaws. And then also we have the identity issue, which depending upon how we're doing detection, we, you know, if somebody logs in, Let's put it this way, if somebody logs in with valid credentials, they're using valid protocols.

In your [00:16:00] example, if they were logging in from a relatively kind of consistent IP address, there's not really any reason to to generate a detection alert. But there's another and, and what I think is almost the bigger problem, I read a report the other day that talked about the average security operations center receives 11,000 alerts per day.

11,000 alerts per day doing the overly complicated math of dividing that by the number of minute minutes. That's 7.6 alerts per minute. In this same report, it says that a SOC analyst takes an average of 10 minutes to deal with each, each alert. If you add 11,000 alerts per day at 10 minutes per alert, that's something like two hundred and twenty eight eight hour shifts per day to be able to analyze all of those alerts.

So the idea is that even when our detective controls do actually generate an alert, in many cases it gets [00:17:00] lost in the noise of all of the other alerts that are being generated. So you have this issue of. Another thing that I, I read a study where it talks about, and I've seen various statistics, but false positives, an alert that wasn't actually anything bad.

I've seen some studies that show on average, one out of three alerts as a false positive, and I've seen some that say even more than that. So the idea is that, We can miss a lot of things either due to the attacker exploiting some vulnerability or some issue or because of misconfiguration.

Think about a lot of monitoring is done at the perimeter, but we don't see what's happening on our internal network, those kinds of things. So we can, that's said. We also generate a lot of false positives. The reason why we generate a lot of false positives, it's, it's intentional if we, even if it's not accepted as such.

The idea is when we're tuning [00:18:00] our detection capabilities, not just a specific technology, but overall we generally have to make some decisions. Would we rather have every alert that pops up, be a real thing, but potentially miss a whole bunch of things, or would we rather generate too many alerts, but make sure we capture everything?

And most organizations err on the side of more false positives so that we can try to eliminate false negatives. And if that was the case, that might be okay. But the fact of the matter is most organizations suffer from both false positives and false negatives. Alert overload, alert, fatigue, misconfigurations, attackers that know how to circumvent.

So we end up with this. Information overload. I, I heard somebody at one point in time, I forget it was a, a speech or an article where they talked about security information has become a big data problem. How do we filter through, so we, we implement technologies like [00:19:00] SIEM. To try to, and, and the, the messaging when SIEM came out was, we're going to normalize and we're going to correlate all this information so that we're going to take multiple alerts that

are related to each other from multiple sources, and if necessary, elevate the the importance of that particular alert.

That sounds great. I've not actually seen a SIEM that lives lived, lives up to the hype when they first came out. They're great technologies, but we still end up with this information overload. So alerts get missed. When they get caught, they get lost in the shuffle of all the other things. And I think, in my opinion, this is largely because we haven't adapted our approach to looking at security information accordingly. We still look at this, alright, we're going to collect everything in some big centralized bucket, apply some logic that hopefully allows us to differentiate between the [00:20:00] important everything and the not important everything. But we still are generating 11,000 alerts per day and, and that becomes the huge problem.

[00:20:08] **G Mark Hardy:** Well, what we look at then is in such a noisy environment, if we're getting 11,000 alerts, it suggests that our prevent function didn't stop the 10,500 of them that they should have. Of giving us a more manageable number. So it really comes back to how are we architecting our tool set? And also over time we learn how to tune these things and filter out.

And so I know, for example, I can use geofencing on my users. So if somebody has a valid credential, But that's how I detected this particular thing because somebody came in from a place that was not inside my geofence. They're like, well, okay, I know that people travel and they don't always tell me that they're on traveling, but it was worth a follow up and bingo, we caught it.

And so in that particular case, got some rapid detects. So it seems like maybe the obvious solution is we just go ahead and improve our fidelity of our, in our detection, and then we automate all these response [00:21:00] times accelerated. So that seems to be like a pretty good approach, right?

[00:21:03] **Kevin Fiscus:** And, and, and it is, but so when I say it is, look at all the tech, like think of all of the technologies where you reference them with an acronym that ends with DR right. EDR, XDR, NDR, ITDR. The DR is detection and response, and then incorporate things like SOAR, right where you have the security orchestration, automation and response.

The idea of all of these is we want to do, hopefully, a better job of detecting bad things and then automating the response, and that makes an enormous amount of sense on the surface. But here's the problem. Let's say that we did that perfectly right. Let's say that we won that battle. We figured out a way to detect bad guy activity with 100% accuracy.

I'm talking [00:22:00] about like, imagine we did this perfect. There are no false positives, there are no false negatives, and we get immediate and automated response.

[00:22:11] **G Mark Hardy:** And it's almost like a prevent now in a way because we're kicking them out. But you had a problem about prevent a few minutes ago, so please keep going.

[00:22:19] **Kevin Fiscus:** So the problem that you run into is if you actually managed to do that, you have turned your detective controls into preventative controls. You have now simply provided the attacker with that resistance or that feedback loop. So if we actually did detection as well as it could theoretically be done.

Then you actually just make it easier for the attacker. The fact is the attackers don't know they've been caught. When we don't know they're there. I mean like, it sounds stupid, but like the fact that it takes us weeks to months to detect the attacker means that the attacker doesn't have a way of [00:23:00] figuring out how they've been caught easily.

If we shortened the time between attacker action and then detection and then response, we make it easier for the attacker to figure out what they can. And cannot do. We give them that resistance, we give them that feedback loop. So if we actually achieved the proverbial holy grail of detection and response, we would actually lessen our security posture and make it easier for the attackers to figure out ways around things.

[00:23:31] **G Mark Hardy:** And that's initially counterintuitive, but when you walk through it, It makes sense because as we used to like to say in the military, the enemy gets a vote and also the, the plan is nothing but planning is everything. So from those general concept saying, it doesn't mean we should not try to go ahead and prevent and detect because we need to.

And that's an absolute essential. But what is there a [00:24:00] better model? There's just some other way to do this other than, as Einstein said, Your definition of insanity is doing the same thing over and over again expecting a different result.

[00:24:09] **Kevin Fiscus:** So I believe there is and, and in fact, I think that there are four core components that are required to do this security thing, right? It's a model that I'm kind of coining the term. Detection Oriented Security Architecture. Partially because it describes what I'm talking about and partially

because we get an acronym that we can pronounce of DOSA and we just need lots more acronyms in our life.

But the idea is I believe there are four components that are necessary for a detection oriented security architecture. The first of those components is what I call high fidelity low noise detection. Now, obviously we all want that, but. From a practical perspective, the technology we have available today, we don't have a lot of ability to do this.

High fidelity, [00:25:00] low noise detection sort of. We actually kind of do, we're just doing detection wrong. So when I look at information security, Information that was weird. But if I look at cybersecurity information, right, all of the information that is available to us as we, for example, respond to an incident, that information can get broken down into two categories.

Alerting and monitoring. Not, not shocking, but alerting is designed to say something bad is going on here. Whereas monitoring is just, I captured something. Now in many cases, we take our alerting information and our monitoring information and throw it all into the same repository and hope we can search through it and sort it out.

And what I mean, a specific example. Full packet capture. Full packet capture is a great thing to do if you want to do investigations. But full packet capture never tells you something's wrong. It just says, I captured stuff. Network monitoring [00:26:00] solutions, the majority of system and device logs fall into this category of contextual information.

We want it if we need to investigate, but it doesn't tell us anything necessarily proactively. Then we look at alerting information and we see things like IDS. We see things like UEBA user and entity behavioral analytics. We see certain aspects of endpoint protection, so on and so forth. These, these are things that make the red lights flash and say something bad is going on.

So the first thing that I think we need to do is intentionally differentiate between those two types of information. Now, the way that I like to describe it because it's clearer, is to say, take your your monitoring information and put it into one repository, and then take your alerting information and put it in a different repository.

You don't need to put it in two different [00:27:00] repositories as long as you label it or tag it. I don't really care, but differentiate clearly between monitoring and alerting. But the next step, and this is what I don't think a lot of people are

doing, what I want to do is I want to take, within the context of alerting, I want to differentiate between high fidelity and low fidelity alerts.

What I basically mean statistics as an example. If an alert is 90, 95 or higher percent likely to be real, that's high fidelity. If an alert is subject to false positives, I if it could be just noise, it's not high fidelity. So what I'm saying is what we also need to do is then separate our alerting. We want to take all of the high fidelity alerts.

And send them to a different repository or at least label them or tag them accordingly. I want those alerts to be ones where if it [00:28:00] goes off, you know, there is a problem now because those alerts are so specific, so focused, so high fidelity, you are going to miss things. In other words, I'm saying that the high fidelity errors on the side of we want to eliminate false false positives.

I'm okay with false negatives in that bucket because we have the other bucket, the low fidelity alerts that get treated, how we have always treated those alerts. So the idea is have some ability to say, if this alarm goes off, It's definitely bad. We have a lot of other alarms that are going off that might need to be investigated and analyzed and all that, but if this alarm goes off, it's really, really bad.

And what this gives organizations think about a, a typical organization, right? 24 by seven monitoring is expensive. [00:29:00] Now you might be large organization that can fill your own SOC. You might hire a managed security service provider that does 24x7 monitoring, but there's still a price tag. What if we tried to do 24x7 alerting?

What if we're a smaller organization that wants to be able to, I, I always like to joke, we want to be able to page our people when the alert goes off. I know nobody uses pagers anymore, but the fact of the matter is, if we're averaging 7.6 alerts per minute, you cannot send a text message to your security person at that rate.

But if you knew that, This particular category of alert was absolutely always actionable. Now we can do more effective 24x7 monitoring or alerting. We can get somebody involved much more quickly and and do so while reducing costs. That's not to say we don't need to look at the low fidelity stuff, but it says that if we change our view on that, we get a much [00:30:00] higher fidelity thing. So I'll That's, that's one step. But I'll pause because you you seem like you're, you're intrigued.

[00:30:06] **G Mark Hardy:** I am. So what we're looking at, in a way, we're sort of setting a squelch to say, I've got all this information coming in for those of us work with radios. Now you can reduce the static and only pick up the signal because it's really signal to noise ratio is what we're talking about is that we've got a very high noise ratio typically.

But when we look at this, if we do the physical equivalent, let's say a burglar alarm. So if I have a detector that looks for high frequency, Glass breaking, for example. Then if somebody smashes out a window, that's pretty easy to detect. Yeah, that's not a normal course of event and I should alert, but somebody slowly jimmying the back lock or picking a lock on the back door somehow is not going to generate that much noise.

And so you're right, I'll miss that potentially in terms of a false negative gets through my glass breaking noise, but I'm not going to get any anybody trying to come into that front window is going to definitely get picked up. So what we do,

[00:30:57] **Kevin Fiscus:** yeah, I was going to say, it's kind of like I have those ring [00:31:00] cameras and ring, you know, ring doorbells and stuff like that, and they go off. If somebody comes up to my house. But every once in a while they go off When a bug walks across. Yeah, exactly. Every once in a

[00:31:10] **G Mark Hardy:** This is my hot spare, by the way. It's fully charged.

[00:31:13] **Kevin Fiscus:** But it'll be like a bug will, will go, will go past it and it will generate alert.

And it always happens at two o'clock in the morning, so two o'clock in the morning. Next thing I know, there's somebody at your front door. I'm now awake. Flip on the thing. It's that kind of thing, right? It's, it's, that is designed to be an over that is designed to generate more false positives at the expense of not generating false negatives, but

[00:31:38] **G Mark Hardy:** So, so, so what we're talking about now then is, is taking this whole universe of detection functions, setting some threshold, squelch, however we want to call it, to say anything above this level is most definitely something bad. There's bad stuff down here, but we've already got a process to deal with that, and it's buried in the noise, and we're not going to get any better necessarily at this, but we'll definitely get better [00:32:00] at these things up here.

And so as a result, we create a special pipeline in our workflow. To say, deal with these right away, rather than them all getting lobbed into one big bucket, which to me sounds almost intuitive that if I'm running a sock, that I'm going to want to cherry pick those things that I care about. But what you're suggesting is, is security leaders, we want to definitively the state.

This is what we're looking for, this is what we care about, this is what I want everybody to, to go and deal with immediately. Everything else can be dealt with. As you know in the matter of course. So now what we're looking at trying to do is, are we then subject however to that same resistance issue that you had talked about earlier where, hey, we're jumping on it like a, a dog on a steak.

But then pretty quickly we realize, okay, don't put your steak on the kitchen floor because the dog's going to eat it. Let's put it someplace else.

[00:32:51] **Kevin Fiscus:** Yeah, so, so yeah, and, and I will say to, to jump off what you said. One thing that is really uncomfortable for decision makers [00:33:00] in organizations is to say, I don't care. If there are false negatives, right? False negatives are a horror show that means something bad happened and we didn't catch it. But in this alternate workflow, I don't care about false negatives.

All I want is high fidelity alerts. The false negatives get addressed in all of the other stuff, as you said. But to your point, all right, let's say that we do this, we get an alert. What do we do next? Right? So the problem that you run into is as a security practitioner, what do you do next? When you get that alert, you really have no responsible option.

Under most circumstances than to eject the bad guy, right? Because what are you going to do? Are you going to sit there and watch them because of the bad guy? At any instant could take action that causes catastrophic harm?

[00:33:56] **G Mark Hardy:** Go back and read the RSA case for really good reference [00:34:00] for something that happened like that. Yeah.

[00:34:01] **Kevin Fiscus:** And so if you sit there and you watch them and you're just like, well, I know they're there, but I don't want to provide feedback or resistant, so I'm just going to let them hang out for an hour and a half.

And after we do that, then we'll kick them off so they don't know what caught them in that hour and a half. They completely destroyed your production database. They embarrassed you publicly. They released all of your top secret

information, whatever the deal is, right. So we cannot do that. We cannot knowingly allow the attacker to hang out in our production environment.

But there's another element to my detection oriented design, and it boils down to what I call non-production resources. The idea of non-production resources. These could be anything in theory, but we're going to keep it simple just for this conversation. Non-production resource, think of a, a server, a workstation, a network device, an IoT device a SCADA system, [00:35:00] an ICS system.

But the key aspect of these devices is they have no other production value. What that means is two things. One, normal becomes defined as no interaction or no use. Any interaction or use is by definition, abnormal and therefore it becomes a high fidelity low noise detection. But the other thing is that if the attacker is interacting with my non-production resources, They cannot cause me any kind of substantial harm.

So if I become aware of the attacker, I know where they are and they're messing with my non-production resources, I can allow them to stay on my network for a longer period of time. And the other thing is that the non-production resources give me time to [00:36:00] respond. I might not have the ability to do up to the minute response.

I might be able to respond in five hours given my staffing. But if the attacker's playing with non-production resources, that becomes more okay.

[00:36:15] **G Mark Hardy:** And so what we have then is this reminds me of an episode called Cyber Deception that we recorded about a year and a half ago with a guy by the Wow. Same name, Kevin FIscus. So yeah, there's some, at least there's consistency here. So what we used to just call honeypots, In the old days, which is basically a decoy system that has no production or no business use.

It's designed to look very much like something that's useful. And back here on my bookshelf, I've got the Cuckoo's egg in here someplace by Clifford Stoll is probably one of the first people to do so and write about it. But now if we think about that, it doesn't require. A whole lot of capabilities, but we probably either have hardware kicking around or if we're in a, in a cloud environment, it can spin up virtual machines and, and no reason not to use something that [00:37:00] has enough things there that it looks normal.

I remember some of the early ransomware. Detection tools to say, am I on a vm? Well, how do you know? Well go look at the most recent documents

viewed because everybody looks at a document and if that has no recent document viewed, then it's probably a virtual machine. Don't detonate. Don't give yourself away.

Well, now what we're saying is create an environment where there is some Clifford stole like interesting stuff. It's. An ability to grab somebody, intercept them and say, Hey, they're looks like they're doing something they shouldn't. Let's flip them over here in the fact that they're interacting. Or, I mean, do you grab them and shift them over or do you just leave this thing as an attractive nuisance, so to speak, and hope that they go in there?

Much like executive salaries dot xls sitting in the home page of the finance server. Everybody works for finance, knows that that's a honey token, but. An attacker or a nosy employee would not. So we, which of those approaches would work best?

[00:37:54] **Kevin Fiscus:** Well, so before we jump into that, I, I will get there, but one thing that I do want to point out is within this, this [00:38:00] context of this deception, right? You can, you can absolutely build your own deception environment, whether it's through virtualization, cloud-based virtual compute instances containerization, docker containers whatever.

You can build it yourself. There are also plenty of commercial solutions if you just want to go buy it. And plug it in. You can do that as well at a variety of, of price points. So the question then becomes, how do we use this thing? Right? And there are a couple of things that I think are really important.

First of all, the way that I look at this concept of cyber deception, I kind of break it down into what I call cyber deception and what I call cyber trapping. Cyber trapping is simply placing these I, I tend to shy away from the term honey pot. It's a correct term, but it has so many connotations with it that people get kind of confused and, and they don't really understand it.

So I call them decoys. But they are honeypots. But the idea is if I [00:39:00] just plant decoys throughout my environment, now we could plant decoys that are standalone systems. We could plant things as you said that you know, employee salary or whatever, dot dot xls. We could plant what I typically refer to as Bates, which would be put on production systems that would generate an alert that is what I would refer to as cyber trapping.

And we're kind of looking at it as let's lay down a field of landmines and hope somebody steps on one. When we get into the deception piece, that's when we

start actually thinking about the attacker As a human being, the attacker makes decisions based on information that they get. So the idea is that we create these decoys or these baits to be interesting to the attacker, and as long as the decoys or baits are both interesting to the attacker and match up with the attacker's expectations.

The attacker would've no reason to question that. So what we're [00:40:00] basically making is attractive things. We can also place what are often called breadcrumbs on production systems. So think of like an ARP cache entry or an RDP profile that references one of these decoy systems. So the idea is we want to create an environment that funnels the attacker using the attacker's decision making process into these decoys or these baits.

Now as far as what you do with it, a lot of that depends upon your operational capabilities. If you have the capability of responding to an incident quickly and effectively, in other words, I'll just put it this way, if you measure your response time in minutes, then all I really need are marginally realistic, decoys and baits, because once interaction happens, it's a detect, and minutes later you are responding.

Now, if you measure your response time in hours or worse yet [00:41:00] days, then you need to create what are often referred to as higher interaction decoys. We need to create decoys that are more realistic and that are more interesting because we need to engage the attacker over a period of time, and we need to give ourselves the time in order to be able to To respond to them in order to be able to deal with them.

But there is a problem, and this problem could be solved in a variety of different ways. But the problem is the attacker interacts with a decoy to your point. Go look at the most recently opened documents or look at the web browser cache, or look at the ARP cache, or look at the DNS cache, or even look at file system timestamps.

If nobody's actually been using this decoy, it's going to be obvious to an aware and experienced attacker. The attacker has to be both aware and experienced, which goes in our favor [00:42:00] because again, if the attacker sees what they expect to see. And they are interested, then they probably aren't going to be analyzing file system timestamps.

They think they're winning already, but if they do discover some discrepancy that might move them off of the decoy and towards a production system. And that becomes a problem because what we really want is the roach motel, right?

We want the, you know, roaches check in, but they don't check out. So how can we do this?

And there's some commercial deception technologies that takes the, take the approach very technologically. When you interact with a decoy, they shuffle you off to, for example, a cloud-based environment where all the decoys live and they have access controls that don't allow you to leave. Right. But if you're just building that on your own, it's going to be much more difficult to do that.

So this is where the third aspect of my deception oriented security architecture comes in, and it's what I [00:43:00] call environment variability. The idea of environment variability is if you have an attacker, they have the ability to keep coming back and back and back and back using what they learned in their previous attacks to make their subsequent attacks more effective and better.

Well, what if when the attacker came back, they didn't see the same environment? What if they saw something different? Now with these decoys and this deception stuff, you could just change the decoys, change the IP addresses, change the names, do whatever you want to do. At least the decoys will now look different.

And because the attacker doesn't really know, which was a decoy and which was production, that can add a lot of confusion and uncertainty on the part of the attacker, which could just make them go away. But imagine this, and this is kind of something that I'm I, I'm working on, and it is similar to Zero Trust, but the idea is imagine that once we get that high fidelity low [00:44:00] noise detect, we know there's a bad guy in the attacker.

We know what IP address they're actively using. So I'm actually working on a proof of concept to do this, which is. Once we get the detect, we grab the IP address of the attacker who is actively on our network. We then push out a policy to all of the host-based firewalls of the production systems. The policy that we're doing is not block that IP address.

Instead, it is operate as basically a port level proxy to redirect that attacker's packets to a similarly constructed decoy. So the attacker the entire times believes that they're interacting with a production system. Let's just say that the attacker interacted with a production system. They found a website.

Cool. They haven't triggered any alerts yet. Then they trigger an alert. The attacker then goes back to that same [00:45:00] system. They get redirected to a decoy with a similar website. As far as the attacker's concerned, they're

interacting with the same IP address, the same ports, and it looks like the same website.

What we've actually done is we've redirected the attacker to an entirely non-production environment. But the attacker has no knowledge of that, and I don't just do that for one production system. I push this out to every production system. So what we basically do is once the attacker is detected, the production environment vanishes and everything they do redirects to decoys of varying levels of realism, depending upon your operational goals.

[00:45:44] **G Mark Hardy:** And that's fascinating. We're, we're just about out of time here, so let me try to summarize what you've been saying to see if I've captured it correctly. So first of all, we need high fidelity. Low noise detection, being able to have a threshold below which we know this stuff is definitely bad. And [00:46:00] we'll worry about the other false negatives down here.

We want to have incident information that we can access right away, so we've got all these other data sources that we can correlate with, want to have a non-production resource, basically our decoy environment or deception environment or whatever. And then the environment variability, which suggests that when attacker comes back to the same location where they've triggered an alert, They get sent over to, if you will, a parallel universe that looks pretty good.

And now what we do is we got better security. The attacker's dwell time has been significantly reduced because as soon as we catch them, because we're focusing on this, they're immediately pushed out of the production environment where they don't get a chance to go cause trouble. The cost of a breach should hopefully go down because yeah, there's still some lower level things, but more likely than that, we're grabbed all the most expensive stuff.

And then the attackers get less intel feedback. So they don't get that resistance because they're not seeing what it was that they might have. You've got better threat intelligence because now you know where the attackers are coming from, and at the same time, you're kind of blowing [00:47:00] sunshine at your attackers.

They're thinking that they're doing a great job when in fact they're not. Did did it capture all that?

[00:47:05] **Kevin Fiscus:** Yes. Yeah. And, and I think that that's the key piece. And, and the, the final point that I want to make is really that blowing sunshine

thing, right? Every control that we could possibly implement could be detected and circumvented by attackers. Right. I I'm not saying that like the decoys and all that cannot be circumvented.

I'm saying in order to circumvent the control, an attacker has to want to circumvent the control. They have to be aware of the control. And with utilizing this kind of deception and these non-production resources, we present the attacker with exactly what they expect. And not only what they expect, but what they desire.

So the idea is the human being that is the attacker never gets the resistance that causes them to get suspicious or to try to circumvent. So it's not that this stuff cannot be circumvented. It's that the attacker never wants [00:48:00] to, and that is a differentiating factor that no other security control that I've ever encountered could make that case when the attacker never desires evasion, avoidance or whatever because they think they have won the entire time.

And that's to that, that's, that's borderline magic.

[00:48:19] **G Mark Hardy:** I, I love that. Hey, Kevin, if somebody wanted a little bit more information on your deception oriented service arch or security architecture, how will they get in touch with you?

[00:48:27] **Kevin Fiscus:** Sure. So a couple things. I dunno if you can, you guys can read the thing behind so my, my company that's rolling this stuff out is called Deceptive Defense, deceptive defense.com. And then you can also reach out to me at K Fiscus, KFiscus@deceptivedefense.com. Also another kind of, I, I haven't updated in a while, but I'm planning to I've got a YouTube channel around cyber deception and the YouTube channel name is Take Back The Advantage.

[00:48:54] **G Mark Hardy:** Well, that sounds great. Well, thank you very much, Kevin. Appreciate your time and your insight, and glad to have you back on the show. And [00:49:00] for our listeners and our watchers, thanks for joining us here on another episode of CISO Tradecraft. We hope that we've been able to provide you with actionable information to help you better defend your enterprise.

And to get you thinking outside the box. Again, this is your host, G Mark Hardy. It's been a privilege to have you with us. If you're not following us, please follow us again, and until next time, stay safe out there.