Heather (00:12):

Welcome to The Hurricane Labs Podcast. I'm Heather, and today I'll be chatting with pentester, Dennis Goodlett, about a recent responsible disclosure experience he had. Dennis, thanks for joining me. Why don't you go ahead and tell us a little bit about the vulnerability.

Dennis (00:30):

So not long ago, I had a engagement with one of our clients and during the engagement I ran into a website that interested me and had some things about it seemed a little bit strange. And so I ended up playing with it quite a bit. The, the general idea is when a website does something tricky, then there's a better chance that it's going to be doing something wrong. It's kind of like you can't make a rock do much. It's hard to make a rock mess up, just a stone, but a intricate mechanical watch–there's all kinds of things you can do to mess it up. If you drop a hair inside of it, it'll mess up.

Heather (01:16):

Right, the more complicated, the more chances for breaking

Dennis (01:20):

Right. This website was doing enough that I thought that surely they messed up something somewhere. Soi wrote a web extension, it's called Eval Villain, and it's a web extension only for Firefox. Sadly, it doesn't work on Chrome, but it will monitor the common cross-site scripting sinks–that is common places where cross-site scripting Dom cross-site scripting will be added to the document object model. So like inner HTML, document dot write those kinds of things. And it keeps an eye on those things looking for user input. So while I was testing this website Eval Villain also pops up stack traces for these calls and in the stack trace, there was a call to a function which was obviously what parses URL parameters. So this is a great way to find unused URL parameters that can just cross reference that function inside the rest of the JavaScript and see if there's any URL parameters I've never seen yet. Those unused parameters might do something there that I find dangerous. So I grepped through the code and found an unused URL parameter that I had not seen being used at all. It had an interesting name and so, and there's some interesting code around it. So I ended up messing with it a little bit, but eventually once I get it to actually fit the format Eval Villain popped up again, before I even know what this perimeter really does Eval Villain popped up and said, Hey, it's being used in the Dom. So a little bit messing around later, and I get Dom cross-site scripting through this parameter. I know this was for a client test, but it turns out that this application was not created by my client, it was created by another company. So at the end of the test, as we do, we created a report to notify our client of all the findings that were present. So in that report is this Dom cross-site scripting explaining the impact and what can be done with it, and even how to fix it. The problem is though this isn't our client's code. I don't know that the client, maybe the client just throws this application away. Maybe the client adds a waf in front of it instead of trying to talk to the vendor who created this code. I don't know the what's going to happen on the other side of things. Now as the client, but their clients are busy. They're doing stuff, you know, they're, they have other things that they're worried about this test finished and they're moving on to the next thing. So they weren't real vocal back to me about disclosure to the vendor and that kind of stuff. They don't want to open that can of worms, but because now the client knows a vulnerability inside of a vendor product. It could be, I mean, I wouldn't accuse our client of this, but in theory, or a client could use that to attack competitors. Now, our client's not going to do that. Our client's just more interested in getting their code to work, but ethically I feel responsible for ensuring that the vendor gets this information. Even if our clients said, yeah, we're going to talk to the vendor directly. I really should

double-check with the vendor that this is actually happening. Sowe begin a disclosure process, and I hate the disclosure process. It's always terrible to try and find contact information, especially for some companies. So there's this neat–if you are a company you want to make the disclosure process as easy as possible. This took me a couple of weeks in order to disclose. And after a while, as you imagine spending a couple of weeks on this, after just trying to talk to somebody and as someone who isn't patient might start to think that public disclosure is the way to go, and that's not what you want for your company.

Heather (05:53):

Now, what if it makes sense for my company or for my site to limit access to our contact information for some reason. What could we do then to make it more accessible to someone trying to do a responsible disclosure?

Dennis (06:09):

So there is this neat idea called security.txt. If you Google security.txt, you'll find the write-up about it. And it's essentially a way that you can hide a little text file on your website that will notify everyone that knows where to look, how to talk to you in case you have a vulnerability. So if someone finds a vulnerability in your website through normal use, if someone finds a vulnerability in one of your products, then they know exactly who to talk to. This also lets you have a process for receiving this information to the right person and being able to verify it correctly. This company did, the vendor did not have a security.txt. So I searched through their website and ended up finding a Contact Us page. So that seems like the best idea. There was a phone number there. The first time I called I got it was later in the day, but I got put on hold for for 10 minutes and then I got an answering machine. So I did that a couple of times it didn't work, same idea each time. So I called back the next day and I got sent to support and there's different types of supports. And I don't understand which type of support constitutes a security vulnerability because this isn't even the correct way to contact them. It turns out. So eventually support notified me that I should make an account on the website so that I can create a ticket. So I did that now it takes me a while to make an account too, because I need to read the terms and service of this website. This is a matter that might cause legal trouble for myself or for my company. Some people really don't like disclosures. So if there's something inside of the terms and service that denies me the ability to disclose properly without breaking the law, then I can agree to those terms in service. So this is already a large annoyance

Heather (08:20):

That blows my mind a little bit like the website that you are trying to disclose to that's making it harder?

Dennis (08:26):

So the website, when you sign up for an account, there's a little button that's like, I agree to the terms and services and everyone clicks without reading, which myself included in most cases, but in this case, because I'm representing my company and this is, you know, there's a chance that they're going to try to prosecute me or something. I read through their entire terms and service. They had some stuff in there. Like for example, they had some stuff in there about not creating a fake account so that I can't do this anonymously, in essence, I have to represent myself. Which kind of stinks cause I would have enjoyed doing it anonymously just so I don't get spam stuff because also in there is there's essentially a line that said that the terms and service said that they would use my information for marketing and would be selling it to other marketing teams. So I am going to get spam because of this. Eventually when I do get

everything together, authenticated and all that, I still can't make a ticket. So I call support again about making a ticket now support can't figure it out either why I can't make a ticket. So they try to make a ticket in my behalf and they can't make a ticket in my behalf that doesn't work either. So in the meantime, I hit up the company's Twitter page and this is what I should have done from the beginning. I think talking to the marketing people is probably a good idea when it comes to disclosure, because I think they're very interested in getting you to the right people very quickly. So when I hit up the Twitter page, I immediately got an answer and was provided with an email address for their security team, which I had not seen on their website yet I had not seen anywhere else support did not recommend. So I finally have an email address for security specifically. So I email securities, the security email address, and they tell me that it is the security email address, but it's the wrong security email address that we needed a different that's for the product that I'm working with you need this other security email address. Now the first security email address made sense. It was security at the second one did not make any sense. I never would have guessed this email address. The security email address quite often, companies use security @ and then the company name, the company's domain in order to for disclosures. But this presents something of a difficulty because you don't want to just like email random email addresses, hoping that it's a real email. So I don't want to be sending my disclosure to security if there's no one there. So anyways security did have somebody there. They pointed me to this other email address, contacted the other email address. That was the place to talk to my first contact with the other email address was double checking. This is an appropriate way to disclose a medium level vulnerability. They agreed. And so I provided them with the details, including my recommended fix. This vulnerability was a little bit weird while the cross site scripting that I found could be fixed very easily, this perimeter ended up populating a variable that's used all over the place for all sorts of different things that could cause other vulnerabilities. So the cross site scripting I found is the most direct way to exploit this, but there's other places that could be exploited too. And, and so I made that clear and made my suggestion as to how to fix it, how to avoid the whole situation to begin with. I also made the suggestion that they modify how URLs are parsed so that it could not be hidden. So even if a new vulnerability shows up, their IDS should trigger when someone's trying to create the exploit. When you're trying to create an exploit against an IDS, you don't know what will it will block when you're trying to bypass it or bypass a web application firewall or something like that. So you have to try a lot of different things and inevitably you will get caught with something. And that makes noise, the more noise created by someone trying to create an exploit, the better the chance you will catch them before they create a valid exploit. So I made those suggestions. They were, they responded, "Hey, it's fixed now." And that was the end of it. They did not. They told me that because it's a medium level severity, they do not issue a CVE. Which means there won't be a big public disclosure about it. They'll just be a note inside of their change log. So if you don't keep an eye on the change log of all your different software that you're using, then you may not know that you need to upgrade in order to fix a vulnerability, which is kind of the point of CVEs to notify everyone. Also that was the end of the test with the client. The client test ended weeks before, so I didn't have access to this application anymore. So I couldn't test a fix. I don't even know if a client got the fix. I don't even know if the client approached the vendor independently. I couldn't tell the vendor who my client was because of nondisclosure. So it's a weird place to be in, but it's concerning because I've bypassed fixes a ton from developers, especially in situations like this, where things are a little bit off, because the way your L parsing has done there's characters that I can get a URL fragments do not do the same encoding as the rest of the URL is part of the, like the query string in a URL. So typical filtering won't work in some situations. So I can't play with that stuff to see if one of the other places where I was concerned with could still cause the vulnerability. I don't know what they did, so I don't necessarily know if it really is fixed. So if you're a company listening to this and someone comes forward to try and have a disclosure: one, you should, you should

make it easy on them as easy as possible on them so that they don't give up. They're trying to help you out. They want your product to be better. Be nice to them. Try to find a way that they can have contact information easily, such as security.txt that way they know where to go right away, also offer to let them test your fix, let them know what the fix is. That kind of information. It's a kindness to them. It shows respect to them, but it also, they've potentially spent hours weeks working on this exploit and they know quite a bit about what's going on and they can help you ensure that your fix actually works. Once you say that there's a vulnerability in your change log, it's a lot easier for someone to find where that vulnerability is from diffing code and they will be able to check also if the vulnerability was properly patched or not. So it's better to have a security tester who's on your side, check it beforehand.

Heather (16:00):

It's crazy to me how hard you had to work just to do them a favor, really. I mean, and alert them to this vulnerability.

Dennis (16:07):

Yeah. At least I didn't get prosecuted.

Heather (16:10):

Right. Yes. That's, that's a positive. Why do companies make it so hard? Do you think like, like you're trying to do them a favor, like you're helping them out and giving them an opportunity to fix something that's broken. So why are they making it so difficult on folks such as yourself?

Dennis (16:28):

So I'm not a programmer, but I think the programmer term for that is rust. So when you have a feature inside of your program that doesn't use very much, then it's going to end up rusting closed like a door hinge that isn't used ends up getting rusty, and then you can't open the door anymore. So I think that's a lot of the reason for this is a large part of this company is dedicated to doing, servicing customers and making sure that their product is working appropriately for customers. So all of their contact information is aimed towards that. 99% of the people that call are asking about why can't I log in? Why isn't this working? I thought my product was supposed to do this, but it can't those type of questions. And, and so that's what their developers pay attention to. That's what the people designing the website, that's what they pay attention to. So I think, especially in this case, cause once I, once I spoke to the security team, everything went beautifully. They had it fixed in a day. I think like their fix happened very quickly. It's one of the fastest from, from contact to fix that's the fastest I think I've ever had. But finding that magic email address was the troublesome part. And that's because they're like they, that company doesn't even think about it. And I'm not blaming them in that sense, but it's no wonder that just doesn't come to mind. You know, things like security.txt Is, is really cool in that regard that it makes it as long as you have it on your website. It'll take care of all that for you.

Heather (18:09):

Alright. Well, thank you Dennis, for joining me today and sharing your disclosure experience. I appreciate it. And that's all from us today. So be sure to check us out next time when I chat with a few of our teammates about their take on the security skills gap. Until then, stay safe.