Heather (00:12):

Welcome to the Hurricane Labs Podcast. I'm Heather, and today I have two of our pentesters, Meredith and Dennis, here to talk a little bit about what goes through their minds–what their process is–when attacking a network or an application. Thanks for joining me.

Dennis (00:29):

Hello, I'm Dennis.

Meredith (00:30):

Hello, I'm Meredith. It's great to be here.

Heather (00:33):

It's great to talk to you guys, Meredith. I think that we're gonna go ahead and start with you. Why don't you go ahead and walk us through a little bit about how you go about attacking a network?

Meredith (00:44):

Sure. So where I all start depends on the scope that we're given. So either we know exactly what our targets are and they are specific addresses that are given to us, or we are given a range and, or additionally we are inside of the network and our scope is whatever you can find, whatever you can touch, that is your scope. And so that is basically defined as, you know, white box versus gray box versus black box pentesting.

Heather (01:14):

Can you elaborate on what that means?

Meredith (01:16):

So black box testing is the, I would say one of the most common types of pen testing, where you are either given an IP address, a scope or neither of the two, and are just given a machine, either inside the network or external, that can access the resources that they want. And you are told, see what you can find, see what vulnerabilities you can find. You are essentially in a black box, you have no insight into what is where or what is running in that environment. Gray box testing, the next level up, will give you a bit of insight. It will often give you segmented scope stating this is our production network, this is our development network, this may be where we have PCI and compliance. You know, that is how it scope. This is our user network, and you got base insight into what may be running on each of those networks, any specific protected areas that you need to stay away from or take extra caution to. And then white box testing is essentially you know exactly what is on that network, you are given very strict details as to what is where, why it is there and often the controls that are believed to be in place so that either you cannot pivot from system to system. And white box is often used for compliance purposes when you are confirming vulnerabilities and compliance, because you do need to know where everything is to confirm that something is as expected or falls within the lines of compliance.

Dennis (03:03):

Yeah, it's like a spectrum. On one end black box is very much what hackers are facing. Real attackers have to face a black box. They don't expect a customers–a company–is going to give them any information. White box, you give the pentester a lot of information. And from this advantaged point can

they get in? If they can't get anything and they know everything to begin with, then you're in good shape. Some of pentesting, especially blackbox pen testing is guessing. And depending on what the pentester guesses, may just not guess as good as a real attacker. Additionally, depending on what they guess they may take a system down, whereas white box testing, they know what's going on and they can focus their guesses.

Meredith (03:55):

So pending one of those decisions that helps me know, okay, here's where I'm going to start, and I'm either going to focus in on the basis of scanning to see exactly what is running on all of these systems. And ideally I can figure out a bit of why it is there and what each system is. And so I like to use either some of the higher level tools like Qualys or Methods or one of the base tools that every pentester should have in their tool box, which is Nmap, just network and port scanning does have some simple scripts as well in there that you can throw at things. And that lets you know, okay, this system is, has, you know, these five ports open it's most likely to be a web server. This one's got 20 ports open. It's probably a poorly secured active directory server. And that lets you know what's running on there and you can look at the same results and say, okay, this web server is running this, let me go poke into sweat server. And that leads into the application discovery, and the next fun step of a pentest, which is attacking the application.

Dennis (05:06):

Yeah. That's where I come in. So pentesting and attacking often just comes back to like figuring stuff out. So in the very beginning, like Meredith was saying, we find hosts, find out what's running on them, and then we look into what in greater detail to what's running on them. It's impossible to actually test every application running on an internal network to test every byte of code that exists on some company's systems. So pentesters have to, and attackers have to focus on where they think the vulnerabilities are. Just for example, I had two weeks to work on one application and the, the JavaScript running on this application alone was hundreds of megabytes in size and ended up being more than 140,000 lines of code of JavaScript after it was beautified. So I can't, I can't read that much code in two weeks. So I have to focus on the functionalities that are most likely to cause problems. So for like a web application test, we generally don't look too closely at CSS ever. There is an occasion on occasion, you might use CSS or some CSS property if you get CSS injection somewhere. But for the most part CSS, isn't a good source of vulnerabilities, JavaScript is. So a great deal of the application testing is finding interesting places to poke around that. And then in a sense, asking those applications questions in a way that they'll answer. I mean that as an I learned this from one of our, I think our I guess he's not our most senior tester, but we have a pentester named Steve, who I was working on a SQL injection vulnerability and I could not figure out something. And so Steve showed me an injection that was pretty much benign. It didn't cause any problems, but it answered the question that I was having trouble with. It proved whether or not a single quotes or double quotes was being used. It proved what kind of encoding I need to use. And those essentially are the questions that you send at the application. You're asking, are you vulnerable to SQL injection? But you can't just say it in human words. So you have to send tick or when it goes one, but it's not always going to respond yes to that, even if it is vulnerable. So you have to try other things. And additionally, you don't want to be wasting your time trying to find SQL injection on a page that's just static HTML that the it's not even sending a request to the server, it's only once you hit the send button, it's only just doing stuff in JavaScript. So the beginning part of an application test is figuring out those types of things. Where, where do I want to look? Where do I want to spend time? So I spend a great deal of time just poking around at the application whatever it is, whether it's a web application or a FTP server or whatnot, figuring out what, why it is there, what information what it's legitimate purposes for.

Like, if it's an FTP server, why are they using this FTP server? What kind of files would be on there if it's just marketing material that anyone has, and I don't think I'm going to find a vulnerability, then I don't care too much to like test access restrictions. From that It's a matter of narrowing down what I really want to spend time on, what I think the client will get the most value from me testing. On occasion the test isn't always, sometimes I'm just giving an application to test and, and of course that's what I'm spending time on. So once I'm actually into a feature that I'm interested in that like I'm testing for SQL injection, or something like that, or I'm actually looking around at the website, one of the things that I'm always looking for is places where I think developers are still–have misunderstandings. So there's certain things that seem very easy in programming that are actually very difficult. And there's other things that seem difficult that are easy. I'm looking for those weird places like that, that I think a mistake could have occurred. One of the best examples of this as a URL, if a programmer is attempting to purse the URL themselves, then they're almost certainly going to get themselves in trouble. Nobody knows how to parse URLs. You can even URL lib two versus your lib three in Python, just one version away, disagree upon what constitutes a URL. So if I see somewhere where obviously URL out as being parsed or a URL would have to be parsed on the backend, then that's something I'm going to mess with a lot. And I'm going to try all the little tricks that I've learned over the years as to how you URL parsers mess up, depending on what the request is doing. Quite often, those kinds of things are good for server-side request forgery depending on what kind of attack I'm getting at. Depends on what kind of payloads I'm sending. If I end up getting code execution, then it's almost like starting all over again. Now I need to figure out what on that system do I have access to what tools are already available for me to do my bidding? So I probably want to upload a file to that server at some point, how can I do that? If it's Linux then does it have curl? It doesn't have curl. Does it have open SSL? You can use Open SSL S client to get, to make a connection like a raw encrypted connection and send files that way. Windows has its own little sneaky tools and command line. Can I get remote desktop to it and upgrade my shell to something more usable, those kinds of things. A lot of pentesting that the vast majority of pen testing is just searching around and poking and stuff and seeing what's there. What can you use and what can you make the system do? And how can you use those tools that you have available to you to make the system do something that the maintainers, the client did not intend for that system to do. A lot of vulnerabilities I find quite often aren't even like cross-site scripting or, or they don't have names, it's just that I made the system do something that the developer did not want it to do as, as far as I'm concerned. And so I notify them, 'Hey, maybe you don't want this computer to be giving this information out', or 'maybe you don't want this the computer to be searching your entire database'. Those kinds of things.

Heather (11:51):

This is not the first time in a podcast–it's actually probably the fourth or fifth time that JavaScript got dumped on. Why, why JavaScript? Why is that a key thing when you're looking for vulnerability?

Dennis (12:05):

If you're testing a web application, a great deal of like even internal testing is web application testing and JavaScript is the code that runs on the browser. So the server side code, I have no chance of seeing unless there is leaked or if it's like an open source software that I can go look it up. I can always see the JavaScript that's running in an application. And depending on how the application was created, sometimes the JavaScript gives you a great deal of secrets. So maybe the developer is working on a new endpoint, a new API on the server, and it's not available yet in the UI. So like you can't click the little job button down button and get to this new feature yet, but because he's, he's working on it it's seen in the JavaScript and see a URL or a location in the JavaScript for requests to be sent there. It's just not

implemented yet. A feature that's not completely implemented as a great source of vulnerabilities, a great source of trickiness, of problems. And so JavaScript can give you that, that stuff. I've also found. I found API keys and JavaScript. So the, in this case, the application was designed with a cloud storage backend, and that was for testing just to make it easy on the developers. So the developers to make it easy on themselves, but the API key inside of the JavaScript during testing, but when they migrated to a new storage system that was secured, didn't have API keys in the JavaScript, they left that JavaScript endpoint inside the JavaScript. It wasn't used by anything, but the API key showed up and I could go, I could access that storage device. Now in that test in particular, I was instructed to test only one application that cloud storage is out of scope. So I tested whether I could access, whether an essence, whether the API would tell me new authentication as in your API key is expired or whether the API would tell me, yeah, you're authenticated, what do you want me to do? And then from that, I notified the customer immediately to make sure that cause that that application I think was live. So I emailed the customer right away to make sure that they didn't have anything good on that server. I didn't check. I thought that was out of scope and make sure that they like revoked the keys and that kind of stuff. But JavaScript will often hint at things that's going on. And JavaScript is always JavaScript is written in the language of JavaScript. Whereas some of the applications I test are the website is not in English. So I can look at a website that is not in English, look at the JavaScript and figure out what's going on, look at the APIs and figure out what's going on. Even without Google translate. So JavaScript is, is often a great source of a great way to figure out what's going on, what the developer is thinking, how he goes about things. And it often discloses the type of technology being used. Some developers don't write JavaScript. They write a single language, which then is translated into JavaScript, server-side code and CSS at the same time. So in that case, quite often, I can find secrets and JavaScript. They did that. They did not intend to be there because they were only writing one language. They didn't realize that the, some of the information that they were writing would end up in the JavaScript.

Heather (15:44):

So once you are in, you mentioned looking for like interesting files or about know something that was of valuewhat is it that you're looking for exactly. Like, how do you distinguish between this? That's probably not going to have a whole lot that's good for me, versus this might be something worth having?

Dennis (16:03):

It's not always easy, especially. So there's some, you know, like I'm, I, our clients do any number of things. And so I don't always know whether like an ID number is important. So maybe I can steal that ID number from another user. Is that bad? So quite often for stuff like that, I I'll just report it and, and say, I don't know if this is bad, but this is something that I can do. And you can tell me whether this is an accepted risk in essence, that like, Oh yeah, we don't care about that number that everybody knows those numbers. But so like some of that depends on the industry. Obviously, if I can get social security numbers and that's bad, but is as a user's account number bad. So I tend to act with a great deal of humility in that and assume I don't know. And I do my best to notify the customer of anything that I think could possibly be scary. And then I trust the customer to tell me, yes, we do need to fix that, or we shouldn't fix that for their retest. So that also, there's a lesson there for the customers, for people getting a pentest, if there's something strange in your industry that a pentester probably does not understand such as an ID number or something like that, that is important is not exposed. Then let the pentester know beforehand. We have clients that tell us, we're interested in normal web application vulnerabilities on this website, but in particular, can you get the website to do this? Can you steal this information from a different user, those kinds of things? And then during testing, I can waive those

types of vulnerabilities more heavily and spend more time looking into those details and sending guesses that way.

Heather (18:01):

What advice would you both offer to a new pentester, who's, you know, they're trying to get their feet wet and get into the industry? What sort of advice could you offer?

Meredith (18:11):

Don't be afraid to try anything. I spent a lot of time working through web application basics and the basics of networking and the basics of server configurations for multiple different operating systems and, you know, try everything to find what you like. And then once you find what you really like that you can easily hone your skills and start going down a different rabbit holes that will lead you to your chosen brilliance.

Dennis (18:41):

Yeah. I agree with that. Pentesting is a huge field. You don't want to, I mean, you can make money anywhere. If you're going to be a pentester, you may as well enjoy it. So find the things that you like and try to focus on those things. But also you do want to stay somewhat general because you never know what you're going to run into. Security never stops there's new vulnerability types all the time. If you get into pentesting, you should realize that like you never going to stop learning, technology changes and you got to keep up with it, with new technology comes new vulnerabilities. And so you want to keep a handle on it. You just got to keep going, keep learning. Same for defense. You have to keep learning new stuff happens and you figure out what they like, who they like, and essentially build like an RSS feed or a Twitter profile of people that are producing smart stuff that interests them, that they can continue to keep learning the things that they need to learn.

Heather (19:45):

Alright. Well, thank you both for coming in, speaking with me today. I appreciate it. And that's all for now. So be sure to check out our resources on the links below and until next time. Stay safe.