10 Types of Application Security Testing Tools and How to UseThem

featuring Dr. Thomas Scanlon as Interviewed by Suzanne Miller

----

Welcome to the SEI Podcast Series, a production of the Carnegie Mellon University Software Engineering Institute. The SEI is a federally-funded research and development center sponsored by the US Department of Defense. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

Suzanne Miller: Hello, my name is Suzanne Miller. I'm a principal researcher in the Agile-in-Government Initiative at the SEI. I am here today to introduce Dr. Thomas Scanlon, who works in our CERT Division. He is a senior cybersecurity engineer over there, and today he is going to talk to us about application security testing. We are going to have a marathon, because there are 10 types. We will talk about whichever ones of those you want our viewers to know about. Before we get started with that, tell us a little bit about your background and how you came to be in this particular area of research.

Thomas Scanlon: Hello, it is good to be with you today. It is a pleasure to talk about this. Working for an FFRDC, I get to work with a variety of government military programs. One thing I have been doing a lot the last few years is, [the programs] want to increase our application security testing, and that leads to automating it, a lot of tooling. So, I have been helping them bring tooling into their programs. With a lot of these [programs], just knowing where to start is a big obstacle. There are a lot of options out there. I have been helping [the programs] pick through the options and what's appropriate, because what's appropriate in one place may not be appropriate at the other place. We go through what fits in each context.

Suzanne: When we talk about *types* of application security testing, what do we mean by *types*? What are the different types of security testing that these tools support?

Thomas: I put them into groups of 10 types of testing. That's just my designation, and I have seen listings that have as many as 27 different types.

Suzanne: That is a lot to process.

Thomas: For instance, the most basic one that people do is static code analysis. Those are tools that read your code. They have a copy of the source code, and they read through it and look for vulnerabilities and weaknesses. You can break that down into tools that look at source code, tools that that do static binary code, or bytecode analysis, so they look at the compiled code. That is where you can differentiate further, but at the end of the day it's a static code analysis, so the tool is at rest. It's not at a running state when we are doing this analysis. You may have the source code, you may have the binary, but you are doing tests . . .

Suzanne: So if you have a COTS product, for example, a commercial off-the-shelf product, you are going to be doing the binary. They are not likely to give you the source code.

Thomas: Correct.

Suzanne: There are cases in the government where we do use code that comes from other places, so that is an important aspect of being able to test the binary, not just the source code.

Thomas: Yes, and in the government, in particular, you find cases where the contract between the government and the software producer only allows for delivery of executables. They don't even get the source code. So then all you can do is testing on the binary. That also leads to the next most common one, which is dynamic testing. That is where you run the code, and you do testing with the code while it is running, like black-box testing. So you're going to run it, you're going to throw test cases at it, and you're going to see what happens, basically.

Suzanne: So you are looking at test cases that are geared towards the security aspect, not just performance.

Thomas: The tools do support some functional testing and security, so it's not always just security. Maybe some of the programs I work with--we are doing functional test cases and security. But yes, what you'll do is you'll throw test cases at it, and a lot of times you'll do what is called fuzzing, where they throw known invalid test cases at it, and they want to see what happens. *How does it fail, does it fail gracefully, is it resilient?* Because a lot of times, the weakness or the vulnerability comes about through not handling a failed test case . . .

Suzanne: Or anomalous data.

Thomas: All software is pretty good at getting through the normal use cases, it's when you get into those nooks and crannies that things go haywire.

Suzanne: It would be easy to think that static and dynamic are the only two. You are either code at rest or you are code running. What are some of the other types that are beyond the static and dynamic categories?

Thomas: I think there are principally three big types. So, we talked about static and we talked about dynamic; the third one is what they call origin analysis, or software composition analysis. What these tools do is, they look at all the components and libraries and things that you bring into your product to see if you are bringing in vulnerabilities into your product. What those do--they don't necessarily examine, say, source code--they are just examining the software materials, or the manifest of what's in your software, and they compare that to a list of known vulnerabilities; for instance, the NIST National Vulnerability Database is a common one they look at. So they say, *You are using these components and we know that there are vulnerabilities in those components, so you should either patch or upgrade or use a different component, or at least in some way address that*. Those are the three big types.

Suzanne: That type is a newer type. I talked to a government group 15 years ago that was interested in exactly this type of origin analysis. At that point we basically said, *Well, there's not really any automated way to do this*. So this is actually a big step forward: to be able to do origin analysis in an automated fashion.

Thomas: These tools have really grown recently. One of the hindrances to these tools in the past was that they involve a lot of data, because you are basically cataloging every software component out there and every known vulnerability, and those are continually updated; there's new vuls discovered every day, or introduced every day. Even for the systems that are out there that work really well now, they require a lot of data storage or a lot of traffic to do these comparisons.

Suzanne: It is a lot of bandwidth.

Thomas: The benefit is, one of your biggest exposure points can be known vulnerabilities. Your software may have vulnerabilities that nobody knows about, so your risk of getting exploited through those is low, but if you are using software components that have known vulnerabilities, adversaries are familiar with that. So those are ones you definitely want to patch.

Suzanne: In your research in these application-security-testing tools, is there advice that you would give to a program that's just starting to think about this? What is it that will make this journey a little easier? I'm not saying there is an "easy button," I know there isn't, but what are some things that they should either be aware of, cautious about, or should be really looking forward to in terms of how they navigate this space?

Thomas: Outside of doing some basic research and reading blogs and papers and things . . .

Suzanne: You have a new blog! I heard it's pretty popular, too. Blogs are one of our favorite ways of learning things today.

Thomas: There is a lot of information there. Besides learning information, the biggest thing I say to do is just get a tool in hand and exercise it a little bit. There's a lot of really good open-source tools in each of these classes, and there is also-- a lot of commercial tools have trial license and free licenses. So before you rush in and think you need a certain product or commit to buying a big product, just exercise the tools a little bit. You'll learn so much about the capabilities and the things you're looking for.

Suzanne: Some of our customers have environments that are difficult [for getting] tools approved. Do you have any advice for those kinds of environments? Are there pre-vetted kinds of lists of security-testing tools that are in confined, constrained environments, would be able to have easier access to?

Thomas: Some of the commercial tools are on different approved product lists within different . . .

Suzanne: So that's a place they need to look is, what are the approved products?

Thomas: For instance, in the military, each of the services has an APL [approved products list] or the equivalent, DHS has a TRM [technical reference model]. So you want to look at those approved products lists. Some products would be on there. There are also a lot of products that are developed by FFRDCs and other researchers from the government, so their approval is almost implicit, though you still want to check with your authorizing officials . . .

Suzanne: Your local authorizing officials . . .

Thomas: Correct. But that's a good place to get tools and to get them at no cost, to the tool itself anyway. That's a good place to start.

Suzanne: The first one is free. Then you get involved in it and then you end up—but that's okay, if it's useful to you then that's what you want to do. There are so many tools out there. Are there particular drawbacks to using these kinds of security-testing tools that people should be aware of, any cautions that you would want to give people?

Thomas: Yes. Right off the bat, if you get one of these tools and you exercise them a little, like I suggested, you will find out right away that the tools spew out a lot of output. And so, that's the thing that I have seen programs get caught off guard by. They did the right thing and they went and got the tools, they ran them . . .

Suzanne: *Now what do I do*?

Thomas: And then they had heaps, literally, of output. And then I've actually been asked to help them go through the output and make sense of it. So, be advised that you're going to get a lot of output. So for one, allow time in your budgets and your schedules for your developers, engineers to process that output, especially the first couple times through, it's not so easy. A lot of the tools have settings where once you've looked at an output or finding, you can mark it to not come up again, or mark it remediated; it will test it once, and then if it satisfies the test, it won't bring it up again. There are mechanisms, but the first time it really is labor intensive to go through these tools and go look at all the output.

Suzanne: One of the situations I have seen is that that output is not sensitive to your context. And so, there are things that it may warn you about that actually are okay in the particular environment that you're in, and the tool can't know that. It can't know that you have other kinds of mitigating things happening around the software. So, that's one of the reasons that you end up with a lot of warnings that you may actually not need to pay attention to directly, even though you need to make sure that they are mitigated in some other way.

Thomas: Absolutely. The first attempt to fix--it's easy to fix the code, but there's a lot of cases where there are other mitigating factors that you don't really need to fix the code. You just need to document that this doesn't need fixed for this reason, and then you can suppress that warning. The other thing is, a lot of the tool findings produce syntax-like errors. It's more of a style issue that they find. While it's good to write good-looking code, if you have a lot of competing priorities, it may not be that important to you. So, you can suppress those warnings. So as you use the tool, you can find things you can suppress.

Suzanne: One of the things I know from other kinds of pipeline sorts of things, you are doing one thing and then another and then another, and you are trying to get different kinds of testing done. What are some of the interactions that you find interesting between security-testing tools and other types of automated tools, automated build tools, automated performance managers? Have you seen any issues or any interesting sort-of factoids that come up when you put these tools into a pipeline with other tools?

Thomas: One of the biggest decisions you have to make, in those types of settings, is if you want the findings of the security tool to block your builds. There's a lot of programs that run the security tool as part of the nightly build process, or somewhere in their checking process. When the tool finds things, there's pros and cons to having to halt your build or halt your checking. So obviously, it can slow the process and defeat the whole purpose of doing the DevOps or the Agile.

On the other hand, if you just let the tool run and the findings get put aside to be dealt with later, then they don't get dealt with later. Typically what you see is, they'll put a prioritization on the findings, and so a high-level finding can block a build, but everything else gets set aside. Then you really just need a good business process to say, *We're going to address the findings we set aside once a week* or whatever schedule fits with your project.

Suzanne: We talked about the big categories, are there any particular tools that, out of the 10 types, that you would like to highlight as being something that our viewers may not have run into? So for example, I'm guessing that the origin-analysis tools are new to people that haven't really been doing much with supply-chain management. Are there any particular tool types that you go, *You probably haven't seen this before, but it's important*?

Thomas: I think the emerging space is a class of tools we call correlation tools, and what they do is they take the findings of multiple tools and correlate them together. It's usually a good practice, once you get a tool introduced and running, to run more than one tool. Because we find, there's studies that show this, that the intersection between two tools of the same class isn't as much as you would think. If I run two different static-code analyzers, they are going to find some of the same things, but they are [also] going to find things that the other doesn't find. It's sort of a best practice to run more than one tool when you can. A lot of times you achieve that by, maybe, running a commercial product and an open-source product, so it fits in your budget.

But when you have, I already said that you get a lot of output from one tool, then you run another tool and you get more output. So these correlation tools help you, so you can address when they do find the same finding it will correlate them together and address it once, and will help you manage the output. So a lot of them have really good interfaces for triaging your findings to assign the priorities. So that's an emerging class of tools. With that is something they call Application Security Orchestration. That's where, if you are going to run multiple tools, you will have sort of a tool manager tool that . . .

Suzanne: If I use these settings here, then make sure that those also get set over here, and things like that.

Thomas: Or, make sure my static-code-analyzer tool runs first, and then I'm going to run my dynamic code analyzer. So there's some tools like that, sort of like tools for tools.

Suzanne: So, it's the automation of the automation.

Thomas: Correct.

Suzanne: We're getting very meta here, this is good. So, open-source security tools, everyone can access. The paranoid in me says, *Well, if I run open-source security tools, am I actually exposing myself to bad actors who use the data about people using those tools in wreaking their havoc?* Is that a risk that I should be worried about, or am I just paranoid?

Thomas: You should be conscious of that. So, the security tool itself, if it's open source you have the code, so you can run a tool on a different tool . . .

Suzanne: Probably not a bad idea!

Thomas: Run it on there. Also, just using open-source components in general, there's some aversion for that reason, because anyone can contribute. Even if it's on the security tool, if it's just the library I'm using.

The flipside of the argument, though, is because it is open and transparent, the more eyes are looking at it, some people actually consider it safer because you have a lot of different people looking at it, then someone will find that.

Suzanne: I imagine there are some white-hat types of hackers that specialize in, *Let me see what's going on with these open-source security tools to make sure that they are safe for my use*.

Thomas: Correct. A lot of the open-source tools do come out of, like I mentioned, government projects or foundations that are dedicated to open-source software in general, where they are trusted in the community and they have their own vetting process for who can be contributors and how things get into the code base.

Suzanne: I'm feeling less paranoid now, thank you. I do have that streak, which is why I don't work in your field, because the more I would know, the more I would just not be able to sleep at night. So, I thank all of you who work in that field, because you keep me out of it.

So, we like to stress in our podcast, transition. Because if nobody knows, nobody can do anything. But besides the blogs and some of the writing that you've done on this, are there any other kinds of transition mechanisms that we are using to help people get access to some of the tools that we may have used or may have developed, and how do people get to know more about this if they are new to this area?

Thomas: I've spoken at a couple of conferences where I've presented similar material. We do have a software . . .

Suzanne: Are you going to be at FloCon?

Thomas: I am not.

Suzanne: Oh, man! Alright. But we still like FloCon, even though you are not going to be there.

Thomas: We like FloCon. This doesn't really exactly fit FloCon.

Suzanne: Fair enough.

Thomas: I recently spoke at the ISC Squared Security Congress about application-security-testing tools. SEI is hosting a software-assurance conference in a couple of weeks, where we have different creators of open-source testing tools presenting, so you can come . . .

Suzanne: Is that going to be here in Pittsburgh?

Thomas: It is in our Arlington office.

Suzanne: Arlington, okay.

Thomas: On November 27th. That is open to the public, free registration. You do have to register, but it is free registration. There's going to be five or six people that have open-source tools presenting, sort of 101 on their tool and how you can use it, where you can get more information.

Suzanne: So, this is an area where there is probably more information than you can digest rather than less, it sounds like.

Thomas: Correct. That was part of the motivation for the blog post and things about putting the tools. So, a lot of developers are familiar that there's tooling, and they know what to do when they get a tool report, because it is their language. They understand it, but they're not sure which tools to run and why, what fits their situations. Just sort of putting your arms around it, get it into buckets on paper, has been helpful to a lot of people.

Suzanne: Based on what we're seeing in terms of the hits on the blog posts, I think a lot of people need that. So thank you for writing that blog post. I just want to thank you for talking today. This is an area that, despite my paranoia, I have a lot of interest in how do we deal with all of these problems. So, I really appreciate all of you that work in this area.

For those of you that want to know more, please look at Dr. Scanlon's blog post on application security testing, and some of the other resources that he's talked about we'll have posted on our website, and I want to thank you again.

Suzanne: Great, thank you.