Using Test Suites for Static Analysis Alert Classifiers

featuring Lori Flynn and Zach Kurtz as interviewed by Suzanne Miller

----

Suzanne Miller: Welcome to the SEI Podcast Series, a production of Carnegie Mellon University's Software Engineering Institute. The SEI is a federally funded research and development center sponsored by the Department of Defense and operated by Carnegie Mellon University. Today's podcast is available on the SEI website at sei.cmu.edu/podcasts.

My name is Suzanne Miller. I am a principal researcher here at the SEI. Today I am very pleased to introduce colleagues Lori Flynn and Zach Kurtz who work at our CERT Division. Before we get started with the topic of today's interview, tell me a little bit about how did you end up here and working on what you are working on. What is about the SEI that drew you to this kind of work?

Lori Flynn: Let's see. I have worked on code analysis in the past and probably the most notable thing that I did was co-invent a method for polymorphic program detection. The secure coding group at SEI does a lot of work with static analysis and so I'm excited to continue and expand on the previous kind of work that I did.

Suzanne: Okay. And Zach, what brought you here?

Zach Kurtz: Well, before the SEI, I had a meandering career in various things that were quantitative, and I eventually ended up calling myself a data scientist and came here as a general-purpose data scientist with no background in cybersecurity, but I have been learning a lot.

Suzanne: There is a lot to learn in cybersecurity and a lot of data. So that sounds like a good match. Today we are here to talk about a new tool that is on its way from on static analysis and a classification scheme. That is where the data part comes in, related to that. So why don't you tell us about this project and why is this project involved in the cybersecurity realm instead of just the normal sort of static analysis code quality things that we normally do.

Lori: Yes, so for the last two and a half years or so I have been leading related projects that are attempting to develop classifiers for static analysis alerts where we are trying to accurately predict whether an alert is true or false to try to save time.

Usually determining if an alert is true or false is a time-consuming process. There are a lot of alerts that come out of static analysis tools and limited money from organizations to pay for effort for people to analyze if they are true or false. If we can improve automation of handling static analysis alerts, we can help to secure code by fixing real flaws.

Suzanne: I know from my own background that was one of the adoption barriers to using static analysis tools is that you got lots of false positives and filtering through what's the real problem and what are things that really can be ignored in whatever case we're working in. Even when you are not talking about security, if you are just talking about code quality kinds of analysis, you still have that issue.

So this could be really, really important to improve in the adoption of static analysis kinds of tools, especially in cybersecurity where we have so many vulnerabilities that need to be accounted for. So the classification scheme, I imagine, is a big part of this in terms of, *How do we make these false positives more visible or less visible*? I guess depending on how you looks at it, but making it clear which ones are the false positives from the true issues. You want to tell us a little bit about how that plays out, Zach?

Zach: Yes, so making a classifier for which alerts are most likely to be true positives depends on a couple things. One is you have to have a good set of features on the input data on the alerts, descriptors of each alert that you could feed into a classifier to decide. The others, you have to have a good classification framework. We have, so far, tested a few different classifiers and the one that seems to work the best for now is a gradient boosting machines implementation. I am using Microsoft's open source LightGBM at the moment but there's others that would work comparably. I think for big improvements in accuracy, it's not so much the choice of classifier but improving the feature set in the future.

Suzanne: Okay. So what is the impact of that scheme that you are using on how programmers actually do their work? Do they need to change the way they do their work to improve the feature set going in or is it, *We take all comers* in terms of how we deal with this, and we use the classification scheme to work through what we're being given. How is that playing out?

Zach: Oh, this should have no effect on how programmers work.

Suzanne: They'll be very glad to hear that. That is a big deal because a lot of times we come up with these new technologies and the first thing we say is, *And now you have to do this in addition to whatever else you were doing*. So if they can use these tools without actually having to change the way that they would normally do the coding and the analysis of it, then, that's a good thing.

Zach: I see. The one piece of overhead is that they would need to create the feature sets that the classifier needs, which means running some analyzers that develop features based on any input code set. The system we developed should eventually get to the point where it can take any features they have available to feed the classifier.

Suzanne: So there is a little bit of work to do, but that is also automated for the most part?

Zach: There are metrics tools that automatically compute features for code bases. Any other features they could add could be as beneficial as well.

Lori: There are many different static analysis tools that can be used. There are many different code metrics tools that can be used. Right now we are using those two types of tools to develop features. In the future, we plan to use other types of tools that have been shown (in other people's published research) to improve classifier accuracy. So there are dynamic analysis tools and some graph analysis tools that have also been shown to be helpful in increasing classifier accuracy.

Suzanne: So you are working on both fronts, the classifier accuracy and the performance of the analyzer itself, in terms of being able to automate in a way that gives information that is relevant to the coder, right, not just to the metrics tool itself.

Lori: So, let's see. Now is a good time to talk about the evolution of this project. In the first year of this project, the focus was trying to create classifiers using output from multiple static analysis tools and using the names of the static analysis tools that provided alerts for a particular flaw as features. We explored whether that would improve the accuracy of the classifiers, and we found that it did.

That was not surprising because other research has shown that static analysis flaw-finding tools tend to have a somewhat overlapping set of flaws that they find, but looking at the Venn diagram, there are usually a lot of flaws that are not in the overlap. By using the results of multiple static analysis tools, we are both able to identify more flaws, and sometimes having information about particular combinations of tools providing alerts helped us to create more accurate classifiers.

Suzanne: So what are some of the challenges of doing this kind of work, trying to improve the classifiers, trying to find the data that would be useful in expanding the predictability of the classifier framework? What are some of the things you have run in to?

Zach: One of the key challenges we are working on is how to best use test suite data, which is on essentially fake synthetic code, to use that synthetic data to pre-train a classifier so that it becomes more accurate on real data on any new application. This is largely experimental, but it is a field. The field is called transfer learning, where you take data from one domain and try to use it to predict out in another domain. There is not always a solution, but we can try. It works best when the features in the training dataset are similar to the features in the test dataset. So if you are measuring things like the number of lines of code in a file—not that that is an actual feature we'd use, but just for an example—maybe the training dataset, the lines of code just ranged from 1 to 100 typically for some reason. Then in the test data or in the target domain where you actually want to apply this in a user's codebase, maybe the range of lines is more like 100 to 10,000. I don't know what is a reasonable number. But obviously, those numbers in the training set are not really comparable to...

Suzanne: Yes, they are not at the right scale.

Zach: Right. A very simple transfer learning technique is to rescale the target domain variables to be on a similar scale. You just divide everything in the target domain so that they are ranging from 1 to 100 just like they are in the training set. So you could do that kind of thing. There are much more technical transformations out there that are kind of reminiscent of principal component analysis. Transfer component analysis is one thing that has been published. So you just transform the target domain data to look more like the training data domain, and it might work better.

Now I have just started to look at that. We haven't really integrated that yet. There are some other techniques like when you begin a new project in a company that wants to use this, at first they won't have any data. So they might want to use the test suite data to train their classifier, but over time they gradually acquire more and more in-house labeled data. So as they get more of that, the test suite data will become less relevant.

So how do you manage the tradeoff between using the test suite data and the new data as the ratio of that availability switches? If you have just one new data point and lots of test data, we still need to give a lot of weight to the test suite data. There is an optimal way to balance which dataset gets used more until eventually it is basically just using the in-house data.

Suzanne: So that is almost its own research path in terms of figuring out what the balance or what is the point at which you want to turn off test suite data.

Zach: Right.

Suzanne: And so you are going to be busy for a while.

Suzanne: Yes.

Suzanne: What is the most non-intuitive thing that you have discovered as part of this work? As we talked about, we expected that we would see some overlap in the tools and things like that, but what was a surprise in terms of what you have done so far?

Lori: Well, so that was a novel result. There is a lot of published research on single static analysis tools. I think there were only three papers I could find on Google Scholar that used more than one static analysis tool, and none combined sophisticated classification algorithms in the way we did. So that was a novel result even though we were suspecting that we would get the results we did.

Then last year, we worked on using test suites like Juliet and STONESOUP, test suites that are intended to test static analysis tools to see which flaws they find. There are flaw taxonomies like [MITRE's] Common Weakness Enumeration, CWEs, and there are many aspects of a particular named flaw. These test suites test the tools to find combinations of control flow and data flow and type flow to figure out which tools get stumped with complicated combinations of those and which are still able to find flaws and which tools never find the flaws, even in really simple cases.

So the novel thing last year was using the test suites to generate our alerts. The reason that we did that is because in the first year of the project, we ran into this problem with the classifiers being very bad at classifying alerts for flaws that they had no data for. So we have a set of audit archive data that CERT people have manually audited alerts for code bases. Experts have determined whether the warnings from the tools are true or false positives.

So we had this big alert audit archive, and we used that to create classifiers that first year. But for flaws that we do not have audited data for, our classifiers were very bad, which is not surprising also, but it's very expensive to have auditors audit for every alert. You have to have *trues* and *falses* to create classifiers.

The idea with using the test suites was to see if we could- by automatically generating labeled data true or false for alerts using the test suites, can we create accurate classifiers for all kinds of flaws that we do not have manual data for? So, the test suites come along with some metadata that says, *On this line of this file, there is this exact flaw, CWE-190*. So we created some scripts that just parsed through the static analysis output and labels the alerts. This would probably be a good time for Zach to talk about some of the classifier accuracy results from that work.

Zach: Sure. I don't have all the numbers off the top of my head, the classifiers that we trained on the test suite data, these artificial alerts with labels, were pretty accurate when they predicting on other artificial alerts, which is not too surprising. You have area under the receiver operator characteristic curve of something like .98. It was good.

The challenge then is does that generalize out of that dataset to a new dataset, and it did not generalize very strongly. I think there is some signal there, but it is a bit weak in our initial results.

Suzanne: That means you get more research. Do you have any insights yet as to why that may be an issue?

Zach: The test suites are constructed very artificially but with some realism.

Suzanne: And they are focused on a subset of the possible vulnerabilities, right?

Zach: Right. So a typical test file might be much shorter or cleaner than a typical real-world code file. I am not sure what all the differences are. It might focus on one particular vulnerability instead of a mixture of them. So the structure of the test suites is considerably different than code in the wild. Yet, there could be some similarities, so we are still digging there.

Suzanne: So I imagine there is a lot more noise in the code in the wild is part of what you are probably dealing with.

Zach: There is more noise and different kinds of noise.

Suzanne: Yes. And figuring out how to get the classifiers to recognize the noise versus things that we really need to pay attention to is the challenge.

Zach: Yes.

Suzanne: It is all about noise.

Lori: Exactly. In last year's work, we were using the Juliet test suites, the Juliet C/C ++ test suite, which has small programs, very small programs, about one-to- three, short files long. So they are not very typical. This year, we are starting to add to our dataset with STONESOUP data. The STONESOUP test suites uses real open source code bases. By real I mean naturally-developed code bases. For instance, like a version of Eclipse that are then artificially injected with flaws. So they are much larger, and they are naturally-developed except for the artificial stuff. We have yet to see the effect of that data on the classifiers. That will be interesting.

Another thing that we are trying, that Zach is trying, is to use cross-project classification techniques. There are techniques that have been used by others and developed by others for developing labeled data in one type of project and using that data effectively to enhance production of classifiers for another set of data. That is a big focus of Zach's work this year.

Suzanne: So you have been playing around working with a bunch of different sort of open source publically available kinds of things. Have you been testing this? Have you got any organizations that you can talk about that you have actually piloted this with, in terms of actual coders trying to use this improve the security stance of the code they are producing?

Lori: Yes. There are four organizations. We cannot name them. Sorry. But there are four organizations who are collaborators with us. Three have previously provided us with a sanitized version of their data. We gave them a sanitizer, and we gave them a tool that their auditors use to mark static analysis alerts *true* or *false*. That tool that we produced is called SCALe. That tool exports data into a database, and we provided a script that sanitizes that database, where it removes all messages that kind of code snippets. For every place in the database where a file name or a directory name or a function name is mentioned...

Suzanne: Something identifiable.

Lori: Yes. We do a SHA-256 hash with a salt that they create and they keep so that the version of the database they gave us was sanitized, nothing sensitive was there. But we still were able to create classifiers—create and apply classifiers using that data because for some uses of features you want to know if the function name is the same, or you want to know certain things about a depth of a file. So that data was still available in the sanitized databases.

Suzanne: So you are getting out into the real world with this, so that is good. What are your plans, if any, for allowing some open-source customers, some other folks that would like to understand how to use this in their work? What kind of plans do you have for making something like this available to them? Would they need to work with you directly to create things or is there going to be sort of a self-service kind of way of working with the tools as they become available?

Lori: There is a version of SCALe coming out of our project. SCALe is a tool that is a framework for static analysis alert auditing that has a frontend GUI that allows views of static analysis alerts and of code and has a backend that exports to a database, like I said. The Secure Coding team has been developing that tool for a total of about nine years. For the last two and a half years, we have done the development of that tool. It has always been closed-source during all of the time it's been developed by the secure coding team, but that is about to change.

A February version of our projects' SCALe prototype is about to get published on GitHub sometime soon [Click here to download SCALe on GitHub]. We are doing some other exciting things with SCALe and with classifier development. Actually we are building an architecture that will enable many types of systems like DHS SWAMP [Software Assurance Marketplace] or SCALe Army CERDEC's SwAT, many different types of systems that provide this auditing framework will be able to plug into our architecture and with certain API calls will be able to create classifiers and create sophisticated alert prioritization schemes. If they want...

Suzanne: So they can create their own triage schemes essentially.

Lori: Exactly, sophisticated triage scheme. And our intent is to eventually put all of that on GitHub...

Suzanne: You don't control all of those.

Lori: Yes. So I can't say when that will happen or if for sure…

Suzanne: Even getting one version out is significant to the people that are trying to prevent vulnerabilities from occurring in their world. I think that there will be several of our listeners that are going to be interested in what these tools can do for them. So beyond the blog post and other things, can people get in touch with you? Are you open to new projects to work with on pilots with this technology? So that seems like that would be a place if people think that they may have a fit or they may have a need for the kind of triage.

I know a lot of coders that I know would like, *Please, please, please triage my code, so that I don't have to look at all these hundreds of alerts*. I think this is very exciting in terms of the usability of this kind of technology because that has been the big issue all these years is nobody wants to look at the reams and reams of alert data to figure out which ones are real.

Lori: Absolutely. People have been working on alert classifiers for a while now, and there are some really great methods. Right now it takes a stats expert or a team of them to set that up. And it really helps if you have a huge dataset. Google uses classifiers with FindBugs. Yes, they have super high accuracy rates doing that, but they also have a really enormous wonderful database. We are hoping that using this test suite trick to generate labeled data and creating this architecture that just regular folks can plug their tools into that we can help them out.

Suzanne: I think that is very helpful for the folks that are trying to use these kinds of tools to actually triage what is going on. I look at a lot of the work I do we work with development tests, operational tests, certification authorities. They are always looking for ways to compress the time that it takes to actually be able to say, *Yes, this is, you know, you have done the due diligence on this and you have eliminated as many vulnerabilities as is reasonable based on what we know*. And so these kinds of tools can be really critical to that whole process of compression. So that is very exciting.

I want to thank both of you for joining us today. I hope that our listeners and our viewers will be able to have some interesting times working with the tools as they come out and that they will also read and look at some of the other work that you have done in relating to this. Some stats experts out there may want to get in on some of the details of where some of this has been going. We will be providing links to all of the transcripts here but also to links to GitHub when that becomes available as well as to the blog post and other things that are related to this work. So they'll be plenty of resources for people to look at in relationship to this. So I hope that'll be helpful to your project.

I want to make sure people know that there is a blog post [a series of blog posts]that Lori and Zach have authored on this. The easiest way to find that is to go our blog area and search on Lori's last name, *Flynn, F-L-Y-N-N* and the things that are related to this should come up. So that will be a useful way for you to find new things.

This podcast will be available on the SEI website, sei.cmu.edu/podcasts, also on Carnegie Mellon University's iTunes U site and the SEI's YouTube channel. We have one. Thank you for listening. Thank you for viewing, and thank you for joining us today.