The Open Source Way

Episode 09 - Fosstars

SAP Open Source

Transcript

Karsten Hohage: Welcome to The Open Source Way. This is SAP's podcast series in which we'll talk about the difference that open source can make. In each episode we'll talk to a different expert and we'll talk to them about why they do it the open source way. I'm your host, Karsten Hohage and in this episode, I'm going to talk to Michael Bernhardt and Artem Smotrakov - sorry I had to concentrate on that - about Fosstars. We will learn how it helps you find out more about the security and other aspects of open source components. To our guests: Michael works in the role of program manager for SAP's open source security strategy and is DevSecOps lead in SAP's corporate security organization. In his spare time, he plays around with something he calls semantic light. We will come to that in a minute. Artem is a security engineer at SAP who helps developers to use open source in a safer way. He also does things around electronics in his spare time. But Michael, maybe first, you do semantic light in your spare time. What is that? Hello, by the way, and welcome.

Michael Bernhardt: Thanks so much Karsten, and thanks for inviting me here and funny that you touch upon that. What is semantic light? If I invited you to my home, you would see what is semantic light and you might find this all around the house in here. So the semantic light, it all started off with kind of the experience or with the notion that continuously I was checking for status updates, whether this was the share price or whether this was the weather forecast. And what I perceived is, that this is a constant distraction when you're looking at your mobile phone. And that's what got me going in, kind of starting to read a little bit about how the brain processes light and what I there found is that light is perceived by the left temporal lobe that is responsible for the language and the light perception is separate to that, kind of in the right hemisphere. And so it's not interfering. And that's what makes it pretty handy so to say, to have real-time status data, kind of discrete.

Karsten Hohage: That means you have lamps at home, they change from red to green, depending on whether the SAP stock price is rising or falling or something or…?

Michael Bernhardt: You're right, you're right, there are some weather indications, there are some stock prices, or if I leave the house, then it also tells me did I close all my

windows and just the same kind I see can be applied also to a business world, to an enterprise world on the most relevant information for your corresponding role.

Karsten Hohage: OK, you're married, right?

Michael Bernhardt: I'm married, I'm still married.

Karsten Hohage: OK.

Michael Bernhardt: It's kind of my way to get family acceptance of me playing around with the little raspberries.

Karsten Hohage: I was just going to say you just needed some reason to kind of do hardware software interfaces because, yeah, the job just isn't enough, right?

Michael Bernhardt: That's very much true. It's inspiring to have something else and still keep up a good marriage.

Karsten Hohage: OK, is that similar to the stuff you're doing when you say, Artem, you like to play around with electronics at home?

Artem Smotrakov: Yeah, something like that I also play with some raspberry pies and microcontrollers with some light, some time, and there is a lot of other stuff.

Karsten Hohage: OK, good. So with that established, what kind of guys you are. Let me just come to a little note before we start into our actual topic. We will be talking about matters related to security. Thus, I need to kind of make the following disclaimer. If someone in this following podcast claims something to be secure, it's probably correct but don't take it for granted. If someone in this podcast claims something to be able to assess if something else is secure, it's probably correct but do not necessarily take it for granted. If someone in this podcast claims anything at all, do believe that it's to our best knowledge any time but see above do not necessarily rely on it with your system security. You all know why I have to say that. And with that said, we can come to the actual topic finally. So either of you who wants to start? What is Fosstars?

Artem Smotrakov: Yeah, sure, I can start. Fosstars is a java-based open source project that allows engineers and other domain experts to define various ratings for open source projects. For example, that's my favorite example, security rating for open source project and those ratings help to check how well some particular properties are implemented in open source, for example, security.

Karsten Hohage: OK, so security would just be one aspect and Fosstars really is, how would you call that, a framework to plug in certain assessment tools?

Artem Smotrakov: Good question, you're completely right. Fosstars is a framework that allows to define multiple ratings and allows you to define ratings to assess different sides of open source projects.

Michael Bernhardt: And let me let me add to that: Kind of, what Artem already mentioned as to the framework, once you apply Fosstars and you have that rating, what it allows here is that you have the comparable rating and it gives you the justification for introducing a new open source component with high confidence. It allows you also to assess continuously during the life cycle of your product the state of this open source component. And that gives you criteria at hand that is part of what we call in the security domain "threat modelling", where we assess on what, how stable, how secure the product is. It gives you the perfect criteria in the discussion with the corporate security guys that are always nudging you: "How secure is a product?". So you have a good kind of statement at hand why you selected a certain component, why is it still secure, why it might not be on the latest version, but still can be considered as secure and good kind of high quality enough.

Karsten Hohage: OK, maybe before we go into some more details concerning the framework and also concerning the security and/or other examples, let me first ask: It's kind of a strange name - how did that come about?

Artem Smotrakov: Yeah, I can briefly explain. Fosstars consists of two words: "Foss" and "stars". "Foss" means free and open source software and "stars" refer to a number of stars that we can give for an open source project. Let's say, one star is not too good. Two stars is okay and more than two is good. So that kind of reflected the idea that

Fosstars tries to assess open source projects, giving them some amount of stars, for example.

Karsten Hohage: And is that really what it what it turns out as a result? Do I get one-star, two-star, three-star, five-star software or what's the rating scale?

Artem Smotrakov: Yeah, not really. So in the beginning, we wanted to actually use stars to represent ratings for open source project but in the end we decided just to go with human readable labels and not only limit us with a number of stars but historically we decided to call it Fosstars.

Karsten Hohage: So how, now that we've talked about the name, how did Fosstars come to life anyway? Why was it needed? And what are the challenges that it tackles?

Michael Bernhardt: So let me take the question and I go back to the very beginning of when the whole project started. The whole discussion on Fosstars arised from the debate, that you might have seen from Eric Raymond, so-called "Linus's Law" and the discussion with Jim Zemlin from the Linux Foundation where it was about open source communities and the security of them, speaking the so-called famous "eyeballs statement" that they brought up. So: "Give me enough eyeballs, all bugs are shallow" and Jim Zemlin responded to that: "In these cases, the eyeballs weren't really looking". And he referred to that in particular on the so-called openSSL case, Heartbleed, that was there something around 2015, 2016. And also we had some discussions before. I remember, for example, the discussion with Gordan Haff from RedHat and also reading into the work of David Wheeler from the Core Infrastructure Initiative. And this generally raised the question about - when we talk about open source and how we integrate it in our products - is just looking into the vulnerabilities, is a proper vulnerability management enough? The state of an open source component that you have integrated can change tomorrow if there is a new known vulnerability coming up tomorrow. So that changed completely your risk perception that you had the day before. And so what we did is, we started with a survey inside of SAP, in the end there were two thousand colleagues responding to that. And the results are very comparable to what we have seen from other firms kind of reporting. Bring up reports from Snyk and the DataSecOps Report. So one of the finding was that open source may be seen as a risk during the development if there's not enough guidance how to keep it...

Karsten Hohage: And, I assume, if there are not enough eyeballs who have looked at it, right?

Michael Bernhardt: That's very much true. The question is, how much eyeballs can you invest and how do you come to the conclusion that there are enough eyeballs, exactly. And they kind of concluded that from what they knew back then, they stated that they refer to the eyeballs being given in public blocks, being kind of what is written generally about the component in the internet. And they refer to the comments by colleagues, so their close nearby colleagues. What do they see as the right component and...?

Karsten Hohage: Sorry, just for my understanding here. So I have two different ways of thinking about Fosstars at this point of your explanation: Is it either Fosstars provides artificial eyeballs or does Fosstars assess how many eyeballs have looked at something?

Michael Bernhardt: Good question. The way I would compare it is that, and this might link to some kind of upcoming blog that we plan to release, is the so-called mobile manufacturing. Toyota is referring to it as the so-called Obeya. Obeya is a big room where a number of different people in different roles, the quality manager, the architect, the license expert, the security expert, look at a certain part, a certain process, the production process, the third party part, kind of element, that is supposed to be integrated. And they judge this based on their expertise and give it a process of how it should be assessed so that in the end, the production process assures that the whole manufacturing is, in the end, delivering a secure and kind of well-established product.

Karsten Hohage: OK, now the entire approach of somehow trying to assess the maturity, the security, the performance, whatever of open source components, as we said, it's a framework. Our example is security but you can probably do other aspects as well. That approach probably has been there before. On the one hand, there is eyeballs and on the other hand, I assume there are also other tools who try things like that. What was the particular reason for you, for us to drive the Fosstars project? What were you missing? What wasn't there?

Michael Bernhardt: We looked into what the market considered state-of-the-art back then. And what we found is that on the one hand, we found reports and we had direct exchanges with our partners that have established so-called security review boards. For example, Capital One is one of the companies which claims of having established this kind of, what I mentioned as the Obeya, this kind of group sitting together and judging on open source components. We also know, for example, from Siemens, Siemens Healthcare, in kind of critical components, critical assets to their supply chain, to their final product, they take the extra effort of bringing people together and judge on that. We also learn from the concept of where companies do dedicated forms of open source components to really restrict it. Kind of a very heavy process of putting in the effort to kind of nail it down to your particular need, to avoid that anything which kind of might not be used but might introduce a flaw is kind of irrigated. It's kind of removed.

Karsten Hohage: You made that sound as if this was sometimes even overdoing it, or is it the way it's good and supposed to be?

Michael Bernhardt: I would put it like this: If you have a highly critical area of your product development and is essential where it's really the core of your application, you can do this investment. But considering that nowadays development projects, they consist - also in the enterprise world - around 60 to 90, even going beyond 90% of open source. That's what makes it a challenge to assess each and every component. And you need efficient tools and an efficient way to do that. Where we found kind of with the Fosstars approach, we have the automated approach, pulling the right criteria, giving it a rating which is consumable by a number of people and a number of experts also when they are not from the security domain and give them the clear understanding, is this the right component for my dedicated use case or not.

Karsten Hohage: All right, but basically, we can't make a general statement: "You always do it this way or that way". It depends on whether it is a critical component, then you have, I don't know, you have what the open source community has said, what your tool has said, like Fosstars for instance, and you have additional real humans look upon it. And in other places, you may just want to trust the open source community's assessment plus your tool’s assessment or something, right? Depending on the level of criticality.

Michael Bernhardt: Exactly, what Fosstars definitely provides you is the baseline where you can put the first judgment on. Anything which requires, or has a dedicated security need to go deeper into that, you can do the investment. But like I mentioned, Fosstars is the basis.

Karsten Hohage: So if we spoke in the ways of semantic lights, it's basically Fosstars either puts on a red, green or yellow light, and depending on that, you don't go by this assessment alone but then you either take a look or kind of think: “OK, this is where I take a look when I've looked at everything else”.

Michael Bernhardt: Exactly.

Karsten Hohage: OK, and maybe back to you Artem, how does Fosstars do this?

Artem Smotrakov: Yeah, basically the approach is pretty simple, I try I'll try to briefly explain it. As I mentioned earlier, Fosstars is a framework and this framework allows a domain expert, for example, a security engineer, to define a rating for open source software or to define a model, in other words. So then Fosstars does the following: Fosstars starts gathering publicly available information about this open source project and then put this information to that model defined by a domain, the domain expert. And then Fosstars does some magic inside, with a defined model, we can discuss it in detail maybe later, and it outputs two things, just two things: A score, this is a number from zero to ten and the label. So a score means how well specific property, for example, security is implemented in an open source project. Zero means relatively bad or bad. Ten means good. And if you got something in the middle, it's something in the middle, and then Fosstars tries to interpret this score for you by assigning the case a specific human readable label, for example, it may be good, bad or moderate, depending on the score, which is output by the model.

Karsten Hohage: When you say, for instance, you gather publicly available information using security as an example or something else if you wish. What kind of publicly available information would that be? For example?

Artem Smotrakov: Yeah, great question. In general, we are trying to find variables and data about open source project and try to incorporate it in our models in Fosstars. I'll

give you a couple of examples. So if you're talking about open source security rating, we're trying to gather information about what kind of security tools are used in open source project. For example, it can be static code analysis, dynamic tools for dynamic analysis, fuzzers. Another example that we're trying to check: If The Open Source Way project has security policy that describes how vulnerabilities may be reported to project maintainers. But besides this stuff, which is directly related to security, we're also trying to see, for example, how active a project and how big a community is and things like that.

Karsten Hohage: How many eyeballs?

Artem Smotrakov: Yeah, exactly. How many eyeballs, yes.

Karsten Hohage: Yeah, but then I take it, what it does is really not actually scanning the open source components code for, by some way of magic or these days called artificial intelligence, finding vulnerabilities in the code. But it's rather around the processes and the tools and everything that has been used to make this code secure. Right?

Artem Smotrakov: Yes, you are absolutely correct. Fosstars doesn't scan the actual code for vulnerabilities but you're right, it tries to identify what kind of tools and processes around these tools, open source community has established for a specific open source project.

Karsten Hohage: So in the end, one would have to say the rating that it has as a result is more like: "This code does have the potential to be secure and this code doesn’t have the potential to be secure", because it doesn't actually look in, I don't know, what protocol is being used here and there or does it do that as well?

Artem Smotrakov: The logic here is that if we identify that a project uses a lot of security tools, it has security policy, it's very active, it gives us signs that this project takes good care about security. And this is good in general to prevent vulnerabilities to be introduced to the project in the future. And that's what we're trying to assess, how likely, how secure it is to use open source component in an application.

Karsten Hohage: OK, makes sense, of course, because the tools that are being used to actually make the code secure, you don't have to redundantly do this again but rather gather the information that they have produced and what has been used, right?

Artem Smotrakov: Yes, exactly.

Karsten Hohage: I'm just trying to see if I understand it correctly. OK, cool.

Michael Bernhardt: Let me add one point to that, and this really goes down to not only tools and how the community internally checks on their coding, but this goes also into beyond that of how the community is interacting with the broader community in terms of processes. I'll give you one example. Nowadays, we have something like a private channel where open source communities, well-established open source communities, can be contacted if somebody finds a security bug. Normally, that is done by the community stating on there, in there README.md, it's part of their README.md or as part of their policy, the security file, they are stating how the community can be contacted if such a finding was identified. If that was not the case, you can imagine somebody who finds a security bug might have no real understanding how he can approach someone from the community and think this internally. And that would mean that maybe then he's posting it in a blog, maybe he's posting this as a GitHub issue publicly and everybody could see this beforehand.

Karsten Hohage: Or maybe they just get frustrated and don't report it at all, if it's unclear how and where to report it to, right? I mean, I can relate to that as an end user of things, if I don't know, I'm being contacted by a strange telephone number at work or something. I'm supposed to report this to someone. If it's easy to find out where to report it, I do that. If it's not easy to report it, I kind of think: "OK, did I say anything? No, I didn't say anything. Well, no need to report it, OK?"

Michael Bernhardt: This is definitely not the situation as a community that we want to be in and as a kind of consumer of that component. We don't want a component to act like that.

Karsten Hohage: Right, so here again, you assess the potential, is it easy or is it well-defined how security issues are being reported? And then the assumption is then they

are probably also being reported. OK, I get it, I think. So how then in the end, is there any verification like based on these kind of aspects, is there any verification that a rating is correct, or like, at least near correct or something?

Artem Smotrakov: Yeah, that's a very important point. I think this is very important to make sure that a rating model that's built on top of Fosstars produces meaningful data. So to make sure that our ratings produce meaningful ratings, we have a special bunch of tests. This is known-answer-tests. It's pretty simple thing. So we define a set of data and we define an expected score and an expected label. And then we just feed this data to the model and see if the actual output score match with the expected one. That's pretty simple. And we define the multiple test vectors like that. And to start, so we just define a data set which represents a very bad open source project so that we definitely expect a bad score, a low score and a bad label. And we define then a data set that represents a good open source project. And we expect for this good artificial open source project, a high score with a good label. And then we define some test vectors that represent open source projects somewhere in the middle between zero and ten. And once we introduce something new in our model or change it or tune it a bit, we run this test vectors and see if our produced actual output scores fit to this expected ranges.

Karsten Hohage: Let me see if I understood that correctly, basically two ways of verification. One is producing artificial projects that have an expected rating. And step two, basically is human assessment of has the expectation been fulfilled, right?

Artem Smotrakov: Yeah, yeah.

Karsten Hohage: OK. Michael, you seem to want to add something here.

Michael Bernhardt: I want to add, it's important to say that just a rating doesn't fall from the sky. A rating and the perception of how it reflects security is a continuous learning path. And so what we did is also inside of SAP, we revalidated those scorings with product teams. Artem was outlining on the automation of continuously kind of having this proof that the rating still applies and is still accurate but also the metrics and the weightage inside the rating of all kind combined is something which needs to be revalidated with real projects. And for that, in the end, it very clearly shows and we have seen this as part of one of the first assessment that we did with a team, there were quite

a heavy load of bad ratings and then those questions arise. What do I do as a developer now with all those bad ratings? Is the rating correct? Am I using such an outdated or kind of flawed or non-active project? Or do I need a kind of a process guidance to overcome that, to move the components from bad to good? And so what I want to express is that the automation is one key of that but it's also as part of the internal community inside of SAP, as part of the outside community this discussion of how the rating in the end assembles in the correct form.

Karsten Hohage: OK, and breaking this down to a very simple detail, maybe, would you then discuss things like, for instance, you are saying: “OK, our tool says you don't have a defined channel to report issues”, and the team says: “Yes, we do. Your tool just didn't find it”.

Michael Bernhardt: That could be the way, that could be one of the findings. Right now, I would say we are kind of in the neat situation that, for example, when we look to GitHub, GitHub is right now doing a streamlining of security processes, not only for security, important to say, but also in particular, they are aiming at making the communities applying a standard process which helps to run the open source community overall more secure. By having a security.md file, kind of giving out the policy of how to create it, how to formulate it by having, for example, advisories being given by the community once the want to disclose it. That's one channel. And all these dedicated channels is what can be easily accessed by an automated tool. Speaking of Fosstars.

Karsten Hohage: Then we were speaking about projects now with which also you, or against which you, basically have to tune the Fosstars output. If I want to use Fosstars for rating my project, how do I get started?

Artem Smotrakov: Yeah, we have a command line tool, first of all, that you can download and just pass a URL to your open source project, GitHub URL, for example, and Fosstars is going to automatically collect data and output everything to your terminal. So using this command line tool is just a first option, another option, if you don't want to download the tool and run it, you can just check out a report that we built for kind of well-known open source projects. This report contains a lot of Apache projects, Eclipse project, Spring projects. So you can just check out this report in our

GitHub repository. And we have some plans to add a GitHub action that runs Fosstars, so we have some plans to implement a Maven plug-in that people can use to automate Fosstars checks in their CI/CD pipelines.

Michael Bernhardt: And to give some further insight of how we apply this inside of SAP, I was already mentioning, for example, the concept of threat modelling, where we apply this kind of collecting the information, what components we have, and have them rated by Fosstars, which allows us then as part of the assessment to look at particular into those which have a kind of not so good, bad rating, a moderate rating. Also, in addition, Artem was mentioning the CI/CD pipeline, but also it comes to the next step that when you want to release when you want to deploy your product. It's about how do you validate it and that Fosstars is kind of taken there or integrated into our tool sets there and allowing that we get this kind of a report before deployment, before delivery, making the judgment: "Is the product in a state that is good enough from our, or against our, security principle?"

Karsten Hohage: OK, then, with these at least first level instructions or tips on how you start using Fosstars, let me come to my usual final question. If from this session today, you would want people to take two to three main takeaways back home. Which ones would that be?

Artem Smotrakov: So ratings defined with Fosstars is a model - we all know that all models are wrong but some of them are useful and we're trying to build a useful one to do that. We are looking for feedback and feel free to reach out to us.

Karsten Hohage: OK, here's one, join the community or give feedback, that's point one. Michael, your turn.

Michael Bernhardt: I love that conclusion. I couldn't agree more. The three points that I would mention here is assess your open source dependencies early. Make it easy for developers to get the right and concise criteria for the justification and do not only concentrate on vulnerabilities and avoid the perception that security is the final gatekeeper for your delivery and for your deployment of your product in the end.

Karsten Hohage: OK, then it's my turn to say thank you Michael and thank you, Artem, for being here. That was interesting. I hope the same is true for everyone out there. Everyone out there, thank you for listening to The Open Source Way. Please listen in again the next time. Please spread the word - we have actually switched our rhythm, by the way. We're not going bi-weekly any longer. We will always be on the last Wednesday of the month, a new episode is coming out. Thanks again, Michael. Thanks again, Artem. Let's all say bye bye.

Michael Bernhardt: Thanks so much.

Artem Smotrakov: Thank you, bye.

Karsten Hohage: Thank you.

Michael Bernhardt: Bye.