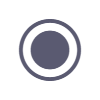
**Teva & Snyk \_ developer training-20250603\_100239-Meeting Recording**

June 3, 2025, 7:02AM

56m 29s

 **Omer Shliva** started transcription

 **Nathan Hart** 0:15  
Awesome.  
Yeah.

 **Chen Zilberman** 0:22  
Someone I was still waiting for more people to join.  
See what?

 **Syuret Pashov** 0:30  
Yeah, I think there might be a few more people that are missing.  
Let me just check and they are going to join.

 **Chen Zilberman** 1:31  
Yeah.

 **Syuret Pashov** 1:35  
Well, I think we have about maybe 95% of the people at the set of the meeting.  
So I guess we we can start anybody wants to join, I mean we have the recording as well. So that should be good enough.

 **Chen Zilberman** 1:47  
Cool.  
Cool. So nice to meet you all.  
My name is hen.  
I manage the Teva account here at Snake.  
We also have Nathan here as our solution engineer and we need this technical session before we begin.  
It will be great to hear from you a little bit to understand. Is there anyone here, whoever used code security tools in the past?

 **Nikolay Angelov** 2:10  
In Teva, we were utilizing Veracode solution.  
This is what we have for the last five years.  
We have some we see some pros and cons of using Veracode.  
So that's why we are exploring other solutions as well.

 **Chen Zilberman** 2:34  
OK. And maybe you can share a little bit more 'cause the purpose of this session is first to match a value you already liked and then add more to replace one another.

 **Nikolay Angelov** 2:46  
Well, basically for us.  
It's very important to to get the essence of the search scanning because.  
We observe in some scanning solutions, basically provide a lot of information that basically we as developers don't have a control over it like some severity findings from third party libraries that.  
Let's say we can or we are not allowed to upgrade for some reasons and basically.  
The overall score is those risks must be accepted.  
Let's say that we have 5 libraries that have very that has issues, but basically it's a grade that we cannot solve or this can be done at later stage.  
But as overall report, I think this shouldn't be.  
Don't pass criteria or something that is blocker because.  
For me, it's important to have a control on the on the status of the scanning whether.  
The scanning.  
The scan is compliant or or not so.  
I'm interested about the the sneakers.  
To what kind of a control we have on the, the scanning and the configuration?  
For the solutions, because in the when the team was configuring.  
Sneak there were like, I don't know, more than 1000 findings.  
You. It's normal that we cannot fix 11000 warnings, but for example we can handle 300 which are more critical and important so.  
For me, it's interesting.  
To understand whether we can focus on specific ones rather than have so much noise in the in the report.  
From my point of view.

 **Chen Zilberman** 4:57  
OK.  
So if to summarize, to make sure I capture that.  
So some of the things you would like to see improved is first of all, when it's too noisy. So that takes us to prioritization and and moreover the context of the security risk for you to understand and like what's the impact? Why do you need to fix it?  
What is the vulnerability like?  
What is the noise scale injection for example?  
We will go over today.  
And moreover, it's the remediation advice.  
So you don't need to go and search for that, rather have as many automations as you can.  
Correct.

 **Nikolay Angelov** 5:34  
Yes, yes.

 **Chen Zilberman** 5:35  
Have you ever used it in the developer workflow?  
The IDE, the CLI.  
Have you ever used the scanner in the in the developer workflow?

 **Nikolay Angelov** 5:45  
Sneak.  
No, no.

 **Chen Zilberman** 5:53  
So when did you got?  
When did you get all those vulnerabilities?  
When it like part of a pity or already when it's in production.

 **Nikolay Angelov** 6:03  
Just I'm not aware of whether, let me explain what is the current status.  
Most of the team here is.  
Basically have sneak installed from the last week.  
They they run it.  
They they play with it, but basically they haven't use it in the workforce. We are starting doing this.  
This to be part of the process.

 **Chen Zilberman** 6:31  
Never been done before.  
That's my question.

 **Nikolay Angelov** 6:34  
Yes, yes.

 **Chen Zilberman** 6:35  
OK.  
So we'll go over and kinda understand what it is. So some of the value of that would be to make sure that you're not just coding code encoding.  
Try to merge it.  
Realize it has vulnerabilities and then you need to start from scratch rather to be part of your initial coding process.  
Cool. So I think we can start if anyone else has any questions or things you want to add today.

 **Nikolay Angelov** 7:04  
Yeah, guys, if you have any questions, feel free to raise it to him and not done so they can reply.  
Basically, to understand more about the platform.

 **Chen Zilberman** 7:16  
Yep. So I'll start with just.  
I don't think we will need the full hour, but I'll start with just the 5 minutes to kind of understand. Explain what snick is while we're here and then we'll jumps straight away to the developer workflow specifically.  
Cool.  
So are you saying my slide?

 **Syuret Pashov** 7:41  
Yep.

 **Chen Zilberman** 7:42  
Yeah. Good.

 **Nikolay Angelov** 7:42  
Yeah, we see it.

 **Chen Zilberman** 7:44  
Good. So this is the SNP platform.  
What you can see here in purple are the different scanners. While it is different scanners, it's still the same scan, right?  
So it's part of the platform of sync and in one scan you can get snyk code result which is s s result like the Novus QL injection or your proprietary code that we check for risk. The open source which is the third party both for vulnerabilities and for LIC.  
Issues configurations.  
It's the container will scan your base images and the infrastructure as code will scan your IAC file, the telephone, the cloud formation, whatever it is you're using, and the only thing that is not part of the platform at the moment is because it's a recent acquisition of a.  
Company name probably, which is the sneak API in Web which is simulate dust. The penetration testing on your web interface now.  
The guardrails.  
The scanner itself is not sufficient enough by itself.  
To gain the self the Secure software development life cycle and we have couple of complementary elements.  
So the sneak essentials will help. Mostly the security teams to understand about your asset. If there is a new report that was created, who are the developers contributing to that report and some automation.  
Snyk learn is the education platform. We'll go over that at the end of the session.  
That's for you to.  
Both product training and security training, the analytics is the ROI, the KPI.  
It's meantime to remediate how many critical vulnerabilities have we merged?  
How many have we fixed?  
So these are all the statistics that help us to improve the overall code security and the security intelligence.  
Basically, the database of vulnerabilities.  
So we have our own security research team that helps sanitize the result that you're seeing and also the remediation advice.  
Now there are a couple of places where we can implement.  
This security guardrails in Cors do software.  
Development lifecycle. Today, we're going to focus on the left side on the ID, maybe a little bit also on the PL. but you should know that there are more guardrails to make sure that your code is scanned right side of the of the development. However, finding those V.  
When they're already in production means you need to stop your Sprint.  
Start coding from scratch.  
So finally it earlier.  
It's also cheaper for the organization, preventing backlog that is gonna grow if you're not finding it early.  
But we also know that you know you might be going through the policy.  
All good, but a few days after a few years after we might find some.  
You know a new log for Jay and you exploit maturity to an already existing vulnerabilities.  
So there is some merits to keep on monitoring your code.  
On the right side of the screen.  
Any questions on that?  
Hello.  
Cool. So we can jump to Nathan Sessions.  
Feel free to jump in with questions with anything you want to learn more because this session is for you.  
So the more questions you'll ask with the more interactive and value we can provide to you?

 **Nathan Hart** 11:01  
Awesome. Thanks hen.  
I just quickly introduce myself.  
My name is Nathan.  
I'm a staff solutions engineer here at sneak.  
I've been at Sieg for about four nearly 4 1/2 years now, and yeah, I'll be taking through this developer focused session on sneak like Hen said. Please do feel free to reach out if there are any questions either in the chat or just.  
Yeah, here in the in the call.  
Let's get into it.  
Hen already took you over most of what? Sneak is in terms of the platform, so I'm not going to go over that today too much.  
I just did want to note though.  
Everyone can see my slides correct.  
Cool. I did want to note that for the purposes of this demo and interactive session, we're going to be focused mostly on static analysis. OK, so sneak code and sneak open source scanning, which are the main components of this.  
Particular interactive session.  
You'll also notice that I've already put in the chat.  
The documentation to the IDE plugins OK.  
In your own time, if you want to.  
In the background you can actually go to the docs page here.  
I'll just get it up on my screen.  
And you can go and download the relevant extension to your coding ecosystem.  
So that's Visual Studio.  
Your code, Jetbrains, you know intellijp et cetera.  
Visual Studio for the net developers and eclipse if there's any Eclipse developers there.  
Feel free to do that in the background and we will continue on with the session.

 **Syuret Pashov** 12:53  
I think most of the guys probably done that already.

 **Nathan Hart** 12:57  
Most of the guys have downloaded it already.

 **Syuret Pashov** 12:59  
Yeah, if not all of them.  
You just it was part of the the Brexit we did prior to the meeting there. Most of the guys were already you know have this on board to their IDs primary vs vs code, yeah.

 **Nathan Hart** 13:13  
Fantastic. That's that's awesome to hear.  
OK. Thanks, tiara. And that's awesome to hear.  
So we won't have to go over that step today.  
I just want to very briefly go over what we're going to cover today in terms of sneak in the sdlc in the, in the software development lifecycle. Hen already went through this, but I just want to focus on on we're going to be doing today, we're actually going.  
To start at the Git repository level, OK, the Git repository level is where you see the projects in sneak and integrating.  
Repositories into sneak actually brings with it a huge advantage.  
Webhooks are set up and each git repository which allows us to then monitor each repository with PR checks. OK.  
Seeing the PR checks we will cover cicd. OK, so sneak in the cicd process and to finish we're going to go and look at the best part of snyk, OK, the sneak IDE plugins which are really, really cool.  
And it's good to know that you guys mostly downloaded these.  
Cool.  
Awesome. So let's get right into it.  
For the actual purposes of this demo, if someone does want to.  
Follow along with the same process that I'm doing.  
You can actually download this repo.  
I will put it here in the chat if anyone does want to follow along with this, OK.  
So you can feel free to fork that repo and copy it to your own code environment.  
Cool.  
Just gonna switch here into a different.  
Umm.  
Into a different org, OK.  
Fantastic. This is the sneak application in the web UI.  
We're not going to stay here for too long, but I just wanted to show you something very important that does happen when we integrate with sneak in the UI, OK.  
So first of all, you can actually manage projects to scan.  
This might not be part of your everyday workflow OK, but I did want you to understand the process that happens when you integrate with sneak OK for example.  
I'm actually going to select a couple of repos here to integrate.  
Sneak is going to import these repos into my UI OK.  
I can actually see the live import status of each.  
Of each repo here.  
So we can see already.  
You know, sneak has started scanning the build Gradle file.  
You know, open source manifest file for Gradle project and if we look at this again probably see some code assets soon.  
Let's go into projects. I can actually see. There's already 2 projects associated with this altro J project and we can see the code analysis here. OK.  
So that's that's really great to see. Code analysis is the actual static analysis on the repo.  
Now of course, you can actually review the results in the Web app and sneak, but we're not going to actually review them here.  
We're going to look at different ways throughout the software development life cycle to go and review the results.  
And make fixes in your own coding environment.  
This was just important for me to show you here so that we get the idea that once you do import a repo into sneak, the web hooks are set up on this in order to monitor in the git side. OK, in order to monitor the changes to the.  
Repo.  
Cool. I'm actually going to go to my own environment now, OK, my IDE environments. I just going to go and copy this code across, OK.  
So cool. Let's have a look here.  
I'm going to clone the Skip repository into my Visual Studio code, OK.  
And from there.  
Let's make a new folder for this.  
New folder.  
Fantastic.  
Cool. So what we have here is a pretty standard JavaScript application, OK?  
Fairly simple, and there's already been some.  
Dependencies added here.  
Now already what we can actually do.  
Is see within this application OK we have the IDE plugin scanned here OK.  
We're gonna come back to this IDE plugin, but it's good to know that.  
You know, sneak is able to scan at this stage already.  
What we're going to do right now is actually add purposefully add a vulnerable package to this repository. OK and we're not going to scan it with seek in the meantime. OK, I'm doing that on purpose in order for you to see how the merge request works.  
Cool. So I'm going to go and install.  
A package called Axios.  
Into this.  
Into this package Jason file OK and I'm actually going to go for a vulnerable version. OK, this version of Axios is actually quite old.  
And we can see later on in the vulnerability database what versions of AXIOS or different packages are vulnerable and what versions aren't.  
But let's go and install this anyway.  
Awesome, I can see now that we've actually installed this particular access package. OK, you can see here it's been installed at the old version. OK, I've pinned it to the older version, which is purposefully an older version because I actually wanted to show you what happens when I.  
Introduce the vulnerability into this. The next thing I'm going to do here in my terminal is actually, you know, do get, add to the manifest file, OK.  
And we're going to commit those changes, OK, with a message of new vulnerability.  
And today is Tuesday. Tuesday. Fantastic.  
Cool. Let's go ahead and actually push this, OK?  
And I'm going to give that a new branch name.  
Wrong branch.  
Choose day. Cool.  
Fantastic. Let's go have a look at that git repo over in the web browser.  
Cool. So I see that I have actually committed a new branch. OK to this repo and the pull request has been made OK, we've got this message here that I've created back in the editor and we're gonna create the pull request.  
Now it's really important to notice that this branch.  
And this repo in fact was monitored by snyk. OK, we can actually see the same project here and snyk if I go back.  
OK.  
We've got sneak goofy. OK, which is the same repo that you're seeing here now because I have already imported this repo into the sneak web app.  
You can actually see here.  
You can actually see here that we have.  
Sorry, I have my kids in the background here, but of home disturbances the downfalls are working from home.  
In any case, you can actually see here that we have.  
Sneak security test on this particular repo. If I click into this OK, it forwards us on to sneak telling us what the particular vulnerability is and you can see this that the package dot Jason file has been tested.  
We can view the test page OK.  
And you can see all of these Axios introduced vulnerabilities that were introduced by my poll request. OK.  
So this is really really useful in terms of scanning those vulnerabilities. And you'll notice that only the Nets new vulnerabilities.  
Were scanned and reported upon. OK, so while there were a lot of existing vulnerabilities within this particular project.  
It only reported on the net new ones.  
Does anyone have any questions for now?  
No. OK, cool. Fantastic.

 **Nikolay Angelov** 22:52  
No.

 **Nathan Hart** 22:55  
Cool. So let's actually get into fixing these vulnerabilities and and going into sort of more fixed flows. Obviously reporting on vulnerabilities and seeing the vulnerabilities important. You can also see at scale within the sneak web app the reports to see in a particular project the vulnerabilities that exist.  
Or across many projects.  
But focusing on earlier in the software development life cycle.  
Is really where we want to be at in terms of being developers who are responsible and fixing their own code. OK, so you can see here.  
That we.  
Are reporting on the vulnerability OK introduced in this particular Axios package version? OK 0.28.1 the one that I purposefully introduced?  
You can actually see here in terms of where it is fixed.  
OK.  
So what version it's fixed in?  
We also report this, of course, in the vulnerability database, so sneak maintains its own vulnerability database. If you're wondering about the source of this information, we maintain our own vulnerability database with all the packaged versions, et cetera. OK. And also explaining, you know, why this is vulnerable, right?  
So really helping your developers understand what in this particular version of this package is problematic.  
OK.  
If we just go back to that vulnerability page.  
You'll also notice this link here. This link takes you on to sneaklearn.  
Sneaklearn is our vulnerability, our security training platform for developers. OK, so within sneaklearn, by the way, all of you have access to this already, OK?  
Within SIG learn, there's many topics to cover, OK.  
There's security education topics.  
There's also product training, OK, specific product training and you can actually do these by individual lessons.  
On vulnerabilities.  
Or by learning paths. OK, so particular whole courses on how classes of vulnerabilities OK.  
For example, you can take the owos top ten security course for open source software.  
Let's go back to that particular lesson that I was on before. OK, so we can see this is a particular vulnerability lesson on prototype pollution in this instance. OK, for the JavaScript ecosystem, which is the ecosystem that I was working on.  
It explains to you very clearly you know what the prototype pollution is.  
What the vulnerability is, OK?  
Exactly what we're going to cover in this lesson.  
And we can actually interact with this particular.  
You know, vulnerability in the sandbox environment, which is really great.  
You can see that we're getting these.  
These this information back.  
This is really really useful in terms of actually understanding how these vulnerabilities work. Sneak is giving you the opportunity here to actually, you know, work with these vulnerabilities in a safe, sandboxed environment.  
So that's really good as well.  
I'm just going to post the URL here in the chat and feel free in your own time.  
To explore sneak, learn really, really useful content for your developers and and for you guys.  
Really, I mean to make you more secure developers.  
Any questions so far?

 **Syuret Pashov** 26:40  
Maybe, perhaps we could see some vulnerability or report of such of SAS scanning, because we've seen so far like open source vulnerabilities.  
You know where you require to upgrade to sort of version et cetera, et cetera.

 **Nathan Hart** 26:54  
Definitely. So let's have a look at a quick look at sneak static analysis tests in the web UI and then we'll look at it will compare it to how that looks in the in the IDE. OK.  
So here we do have code analysis OK.  
And this is a particular Java project which I imported. OK you can actually see exactly. You know what we have covered in this static analysis scan.  
OK.  
It's also really, really useful to understand exactly what analyzed files we have analysed here. OK so.  
HTML, Java, JavaScript in this case, OK, and the percentage of files scanned.  
Also here you can also see exactly what kind of vulnerabilities we are covering here.  
And and going to the sneak learn lessons. OK. So again, we're providing this sneak learn lesson links.  
In this particular project as well, which is code analysis.  
The last thing I wanted to note about code analysis and you guys are going to see this in the IDE as well, is that when we're scanning code analysis projects, OK, we are looking at complete data flows, OK.  
So we're looking at the whole source to source to sync data flow. OK, throughout your code project and you'll notice this actually takes us through different files within repo.  
So we're not just scanning individual lines of syncs of code here.  
You can actually see how we're doing this throughout the whole project and throughout the whole.  
Repo. Really.  
Over here in fix analysis, you can actually see the details of this particular vulnerability. Best practices for prevention and example fixes which are scraped from open source examples around the web.  
The best part of this though is in the IDE.  
So let's skip over to the idea and actually see what that looks like over there.

 **Nikolay Angelov** 29:00  
So basically you recommend this to not to be done through the web interface.

 **Nathan Hart** 29:05  
Yeah, for for developers.  
So for I mean for for most of you, I would really recommend that you use the IDE plugins.  
It's a much shorter cycle from finding the vulnerability to actually fixing it.  
So if we look over here and I think I'm just going to go and open a new window for this, let's open a new window and.  
Hang on.  
Awesome.  
So I have a sneak the sneak plug in here.  
It's now scanning my code OK.  
We're gonna focus in on code security for this particular one.  
The code scans are also generally pretty fast.  
That's interesting.  
Of course it's a technical sessions, so something has to go wrong.  
Let's try and open this in a different window.  
OK, cool. We actually have some code security issues here in this in this demo environment. OK. And we can see that there's seven issues discovered, OK.  
Let's just close that out.  
Awesome.  
Awesome. Cool.  
I'm just going to close my terminal as well.  
So you guys can see this?  
So we have a path traversal issue here.  
OK, first of all, you can see the complete data flow of how this is working, just like in the web app where we reviewed the data flow of how this particular vulnerability is working from the data source. OK all the way through just data sync, we can act.  
Understand here exactly where the vulnerability is flowing through. OK from this, from where the variable was introduced.  
All the way through to the end data sync.  
OK.  
So this is really, really important. Intend is that understanding the actual data flow of the issue? Of course we're also providing issue overview here, so helping you understand exactly what the issue is and how to you know possibly prevent it and the fixing analysis is really the best.  
Part of the idea because here we actually have AI generated fixes OK.  
This is using sneaks.  
On LLM model. OK it's not your your code. It's not just ChatGPT or some other publicly available model.  
Insert it into our application here and what we're going to be doing here is generating AI fixes OK, which are applicable to your own code base. OK, in this particular example, it says we couldn't generate fixes that made our highball standards. So there are some.  
You know, quality Gates when looking at these from a sneak side. OK, when analyzing the code.  
Let's go over to my original JavaScript project and we'll run a code security scan here.  
Cool. So we found that original.  
Package dot Jason vulnerability in the open source. OK, we'll go into code security soon.  
You can actually see here that we've got 134 total and four new.  
This is another nice filter button, OK, that allows you to focus on the code that already exists in the repo, or only the net new issues that you have introduced.  
If we go into the whole code, security issues.  
Cool. We say that there's a couple of issues found. OK. So we do have this. No SQL injection over here.  
Here you can see fixed code examples. OK, these are open source generated examples from like you saw on the web app.  
Here in the dataflow part, you can actually see exactly again the data flow. In this case it's one liner.  
Let's close out this channel just to make things clearer.  
And over here we have this lightning symbol which does indicate that we have a generate AI fix available here.  
This one's a fairly simple.  
I cut out for a moment.  
Can everyone hear me?

 **Syuret Pashov** 33:59  
Yeah, you're back.

 **Nathan Hart** 34:01  
Awesome. Anyone seen my screen?

 **Syuret Pashov** 34:05  
Yes, we do.

 **Nathan Hart** 34:06  
Awesome. So here you can actually see.  
He he you can actually see we've generated this fix AI generated fix OK which gives you a few different options for.  
Concealing the secret. In this case, we can actually go.  
And, you know, apply these fixes as well.  
So sneak can make changes to the code.  
Make it super super easy to actually, you know, go and remediate these issues within the code base.  
So that's an example of, you know, the code generation abilities of sneak within the app, OK.  
The other thing is within the open source world. OK, so going back for for example to that same axios file, let's have a look at that one.  
So here I'm gonna filter out just the four new ones that I introduced.  
We can see this Axios package and we can see where this is fixed OK.  
So there's a couple of fixes that could give us.  
A couple of upgrade paths that could give us a fix for this particular issue.  
We could either do a minor change, which is likely to be compatible with our application.  
You can see the upgrade is from 0.28.1 to 0.29 point 0.

 **Syuret Pashov** 35:25  
Nathan, can I? Can I have a question?  
So you mentioned about the SAT back to scanning that whenever certain vulnerabilities introduced.  
Sneaky Valley in the entire the entire flow.  
It's not just particular lines of the code, you know to effectively suggest some remediation for that, but do we have that?  
Kind of similar thing for open source vulnerabilities like where you're required to jump to a newer version.  
Like would it take the context of the application?  
Like what will actually be the impact of, you know, upgrading to a newer version for this?

 **Nathan Hart** 36:01  
Yeah, it's a really good question.  
So first of all, we don't have AI fixes for open source because generally AI the the open source fixed flow is a lot more simple, doesn't really require a particular AI engine.  
In this case, it's a package upgrade.  
Now sneak will give you if applicable, upgrade paths which are helpful OK in both cases. If you're wanting to do a minor upgrade like this one.  
You can go and upgrade to 0.29 point.  
Zero. You know we can do basically from there and npmi rescan the project, and hopefully that'll get fixed.  
A better upgrade path in this case might be the major version upgrade. If you are comfortable with rewriting the code so that you can, you know, get a get a major version upgrade. So in this case I'll go one point 6.4.  
And you can see, you know.  
If we do an impairment install, it's if this works.  
Npmi OK.  
We're basically changing that package Xiao's to a major version upgrade.  
Then we go, scan, rescan, open source pack.  
Scan again.  
And hopefully we should see the AXIOS issue.  
Basically disappear.  
Let's have a look here.  
Yeah, we don't have the axios.  
Hang on, we might have the access issue. OK, it looks like it's still scanning the old packet version. The package dot the package lock file.  
Not might not be In Sync here, but in principle you would see that disappear.  
Does that answer your question? Sure.

 **Syuret Pashov** 37:53  
What sort of? I mean, the actual mediation is really straightforward. Like you just submit the version that doesn't have the vulnerability, but was more of interested about what would be the impact of the, you know, the software that is using this package that will be using that package.  
Like what will be the required remediation within the actual code?  
Like would it do something like that?  
Because as Nikolay said, like most of the time we are unlocked in specific version because of the software that is being.  
That depends on. This is not.  
So the the problem is not to upgrade to newer version but the actual impact on the software this will have.  
I was wondering if you guys will do any of that next steps like for us.

 **Nathan Hart** 38:38  
For sure.  
A really good question.  
So understand the impact of upgrading that file, especially or that open source package, especially when you might have dependencies that you are restricted to use because of competitive compatibility issues.  
So sneak cannot definitively tell you what the impact of upgrading or not upgrading will be.  
A major of a minor version upgrade or a patch version upgrade is less likely to affect the current code base than a major version upgrade, right?  
If we go, I do want to show you something in a similar vein.  
It doesn't exactly answer your direct question, but sneak does has have the ability to understand reachability.  
What is reachability? I'm gonna go here into the reports to filter outreachable issues.  
What does reachability in in a sneak sense?  
Reachability defines basically if there is a vulnerable function being called from that particular package. OK, so it's another filter that you guys can be using to understand the impact of using a particular package, and if you are using problematic.  
Packages in terms of reachability or if you're using the vulnerable functions within those packages.  
So over here I've got this package with Jason file. I'm going to actually filter out the reachable issues only and you can see there's a lot of them. OK, so we have a few different reachable package issues.  
Again, what is the reachability here? It means that the developer or the the code is using a particular function which has been shown to be connected to the particular vulnerability in the package. OK.  
This can help you in making a more secure decision on whether to use the package or not.  
It is worth noting that just because the package is not marked as reachable or unreachable.  
It doesn't mean that you know the the package is totally.  
To be used so that is important to note. OK, but here you can see the particular the particular function that is being used and it can help you make a more secure decision on whether to actually use this function or this package or not.

 **Syuret Pashov** 40:59  
Makes sense?  
Thank you very much.

 **Nathan Hart** 41:02  
I also, I'm just kind of rooms for a moment.  
Hang on.  
Again, apologies for the background noise folks.  
I'm just gonna also point out the different workflows that we can use. OK.  
So going back, I just want to go back to that.  
Particular slide cool.  
So going back to this slide, you know what we have covered today in terms of the Git repository, OK, the merge code, PR checks, OK, scanning those PR checks and the IDE flows.  
There is one more flow that we can use, OK and that is cicd, OK.  
So using the sneak CLI.  
In the either in your local environment, which I'll show you a demo of, or in the cicd.  
Before I go on to this last part of Cicd and the sneak CLI, are there any other questions?  
Cool. So let's go into sneak in the CLI and the CLI CD. Super super useful.  
Workflows to use again.  
They are later on in the in the software development life cycle, a bit more further away from.  
Being instantly able to fix your code, but really useful in any case.

 **Syuret Pashov** 42:40  
I have a question before I jump into that, sorry.

 **Nathan Hart** 42:42  
Sure.

 **Syuret Pashov** 42:43  
So on the dashboard, it's Nick.  
Can we sort of distinguish between where the results are coming from?  
Where is from CLI. Sorry from CSD processor from the sorry the static codes on the actual repo.  
Can we tell where it's coming from?

 **Nathan Hart** 42:59  
Yeah, of course.  
So basically you have filters here which are showing the integrations so you can slow here in the projects page exactly where it's coming from, whether that Azure reposit.

 **Syuret Pashov** 43:01  
Yeah.  
OK, cool.  
So one wasn't over like the other.  
The previous result, I mean, let's say we have this cut in the the cut length.  
So the static replay then we have this more dynamic thing where it runs in the pipeline as with the CLI of snake, OK.

 **Nathan Hart** 43:26  
Yep, definitely.  
So you can see how I filtered out only the cicd cicli flows here which you can see here. All the results from my CLI flow.

 **Syuret Pashov** 43:34  
Look up.

 **Nathan Hart** 43:40  
Yeah, you can do this by way at the level of the project page, but also at the reports page, right?

 **Syuret Pashov** 43:41  
Thank you.

 **Nathan Hart** 43:46  
So if you're wanting to get like a report of, for example.  
Particular.  
Place where it's being introduced from. You can go and.  
Filter by not type.  
So many filters here.  
I always get confused what to use.  
Hang on, I've got assets, projects.  
I'll have to get back to you on how I sneak products. You can actually filter through sneak products here. So if you're just wanting the open source test for example or the container tests or C codes, you can actually filter it out in the reports page.

 **Syuret Pashov** 44:24  
Mm hmm.  
OK. Thanks.

 **Nathan Hart** 44:30  
Cool. Let's get into the IDE.  
I'll post the link for the CLI here in the chat if anyone does want to download the sneak CLI into their own environment and give this a test.  
It's a great idea.  
Cool. Let's have a look over here.  
I'm just going to share my.  
My terminal window.  
Cool. Let's see what code files I have here.  
OK.  
So I've got a few different code files.  
Here we're looking at the same.  
JavaScript repo that we used at the start and to install the.  
C cli.  
I just use this code flow right without the Docker file bot. It's just an auto complete part. But in any case you can go in and you can go and install this from npm.  
By going npmi minus G and then snyk you will then be taken through an authentication flow, OK, and if you're not, you can always do sneak auth and what I'm going to do once that's installed in my environment, it is already installed.  
I'm going to actually go and do sneak test.  
What is snyk test doing here?  
Exactly what we saw in the IDE, but just helping us to understand exactly this vulnerability which we introduced before, OK.  
Here you can see this access package and the upgrade remediation advice as well here.  
OK.  
Now this is useful in in in within the CLI, but sometimes you may want those results actually ported over to the sneak platform so I can actually let's clear this out and I'm going to do a sneak monitor command. What is that going to do that's actually going.  
To give me a URL to go and have a look at the vulnerabilities within the URL within the UI.  
Just switching.  
Screens to share here.  
Hang on.  
Stop sharing call share.  
Cool. So you can see now that I have actually.  
Monitored this particular project just a few seconds ago.  
The sources from Cicd and you can see the same Axios vulnerabilities coming in from that package which we scanned again with a sneak learn links. Really really useful here.  
The other thing to note about the sneak CLI is of course you can also do a sneak set, a sneak code test, OK.  
So if I just go sneak code test, we're going to be scanning the code flows within that repo. OK, so the actual static analysis on the proprietary code.

 **Nikolay Angelov** 47:29  
Nathan, I have a question.

 **Nathan Hart** 47:32  
Yeah, of course.

 **Nikolay Angelov** 47:32  
Do we have some kind of a report at the end of the day, based on the scanning?  
So.  
We can see in the time how the trend is going on, how the fixes are being done simply to to see where we are as overall.

 **Nathan Hart** 47:55  
Yeah, definitely.  
There's a few different reports.  
Surfaces can use to achieve this.  
OK.  
So first of all, in terms of the actual issues you can see here.  
The issues you can see the issues also over time, OK.  
This is really really useful for judging your overall security posture. OK, total open issues.  
Let's go for the last year, for example.  
OK, so I'm gonna look at the whole.  
Trends of.  
Of these issues, over time we can split this into identified and resolved, OK.  
Let's just look at it identified for example.  
Ideally you would want to see the resolved going up. Of course at a faster rate than they are identified.  
Exposure window time to resolve by week as well as other metrics. OK, now what you would I. I think what you're asking is about the.  
Particular repos that have been tested for by developers in the IDE.

 **Syuret Pashov** 49:05  
Maybe to add to that question, shall the this graphic here will take.  
The actual source of the code, like GitHub or whatever it says and give us the overview of how we do compared to what we had a week ago for that specific report, for example, correct.

 **Nathan Hart** 49:24  
Yep, you could definitely.  
You could definitely use the filters to achieve that. Yeah 100% you can go to the reports for a particular project.  
So let's let's actually do that.  
Let's view a report of just gonna filter back everything back in.  
Let's go to sneak goof.

 **Syuret Pashov** 49:46  
For example, like Project X on GitHub, you know it's there's a code sitting there.  
So a week ago we had 10 vulnerabilities.  
Now have two.

 **Nathan Hart** 49:55  
Yep.

 **Syuret Pashov** 49:55  
You know something to see the visibility, how we do in terms of projects on, yeah.

 **Nathan Hart** 50:01  
Yep, for sure.  
You could definitely just filter out, you know, here I've got just this one project filtered out.  
Project Origin from GitHub Project Target and you can see all of the open issues that I've been.  
This project and of course I can choose, you know, to go and change that report view to issue summary and add those same filters to see what the issue is over time of that of that project.  
Over here you can actually see the developers who are using these.  
So this is really useful in terms of tracking, you know who's fixing and who's who's using the plugins within their own environment.  
So you see total scans by developers here.  
I'm the only developer in my environment.  
It's a very lonely company that I have.  
But you can see here exactly how many scans I've created over each.  
Period.  
In which?  
You know, environments you can see.  
Exactly the adoption by individual users.  
So that's really, really useful to see in terms of developer adoption of these tools.  
The other thing that we can actually do is if you're looking at sort of more.  
Advanced analytics that you're wanting to understand among your different projects, among your different teams?  
We do have enterprise analytics, so enterprise analytics is looking at your whole application in the macro.  
Here we're only focusing in on critical and high vulnerabilities.  
We can actually also stretch this out to a year or longer, and you can actually understand, you know what my exposure.  
Is the difference vulnerabilities management, prevention coverage OK? How many?  
How many projects I have monitored out of my whole code base?  
Really, really useful to understand and also gauge developer adoption amongst your different organizations.  
Any other questions?  
I believe we have 5 minutes. I do want to give hen some time to wrap up.

 **Nikolay Angelov** 52:26  
Guys, guys any any questions?  
Someone like to ask.  
Uh.

 **Chen Zilberman** 52:38  
Cool.  
Nathan, I think I think you're still showing your screen by the way.

 **Nathan Hart** 52:42  
I am indeed I am indeed.  
Here we go.  
So we'll just we'll.

 **Chen Zilberman** 52:46  
Jim yeah.

 **Nathan Hart** 52:50  
I'll just do a very, very quick wrap up of what we have covered. Again, you can see the.  
The product.  
Links in the in the chat from sneak learn to the Ccli IDE plugins definitely use them.  
Today we covered mostly sneak code and sneak open source workflows for developers OK across the whole sdlc, OK?  
So really understanding through the IDE PR checks and git repo CLI cicd we didn't get so much into.  
We can leave that for another session if you guys are interested understanding how to implement the CLI into your pipelines, but we did cover quite a bit today and of course my cell phone HANA here for any questions. OK, so curet and Omer both have our emails and.  
Know how to get in touch with us?  
Feel free to reach out if you feel like another session like this could be useful or a different focus session.

 **Syuret Pashov** 53:49  
Yeah. So on on, we have I think a separation with you.  
I'm not sure if you'll be done about that.  
We we need to do some work on the structuring, on organization of next year. We'll talk more about that next time.  
I think Thursday notification, Fred was Thursday afternoon.

 **Chen Zilberman** 54:06  
Thursday, Yeah.

 **Syuret Pashov** 54:07  
Yeah. OK, Kim, we also bring that in there.  
We might need some help with that.

 **Chen Zilberman** 54:12  
No, Nathan won't be.  
It would be Wesley, but don't worry, I'll be there.

 **Syuret Pashov** 54:15  
OK.

 **Chen Zilberman** 54:15  
So OK.

 **Syuret Pashov** 54:15  
That's fine, that's fine.  
Anybody from outside? Thank you.

 **Chen Zilberman** 54:18  
Yeah, I think.  
Although like I haven't heard a lot of questions, I hope it was very intuitive for you and always clear just to say that there is a a lagging of couple of hours between the IDE results to the platform UI.  
So if you wanna look at the same day, you need to know that it's gonna take a few hours, but also the statistics you've asked in the and Nikolay about the overall code security posture over time.  
Is as it is.  
Good to see over time. So let's say we implemented the ID plug in today.  
I would go in again in the 1st of July to see like the trending over time. How many new vulnerabilities have we merged?  
But also something to bear in mind in terms of statistics that when we see good adoption of IDE plug in, we're seeing that the overall number of vulnerabilities fixed are going down and the meantime to remediate is going up.  
That's because we fixed it before it was even merged.  
So now we have time to clean the backlog.  
So those are vulnerabilities that will sit in your code for a long, long time.  
And now we're able to fix it.  
So these are couple of of metrics to see and we can read it together.  
There are.  
We'll send all the links in the summary e-mail.  
There is also the courses in the SNC LAN for the ID. If anyone needs a bit more training on that on the CLI, everything.  
And yeah, thank you for making up the time.  
Looking forward to see the results and feel free to reach out with any questions.

 **Syuret Pashov** 55:50  
I have one last question.  
So is it fair to say that the CLI, the extension and let's say PR checks on GitHub will produce the same results for example?  
OK, cool.

 **Nathan Hart** 56:04  
Yeah, the tests are consistent across most environments for open source. Yeah, sorry for code static analysis. Open source. There might be some variation in some languages between CLI and Git.  
That's a different discussion, though, yeah.

 **Syuret Pashov** 56:19  
All right.  
Thank you.

 **Nathan Hart** 56:20  
Cool. Thank you.

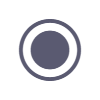
 **Nikolay Angelov** 56:22  
Thank you, everyone. Thank you. Bye.

 **Syuret Pashov** 56:22  
Thank you.  
Thank you very much.  
Bye bye.

 **Chen Zilberman** 56:24  
Thanks. So have a great day.

 **Krum Kuzmov** 56:26  
Thank you, will.

 **Nikolay Angelov** 56:27  
Bye bye.

 **Omer Shliva** stopped transcription