Hello everybody. Wow, what an opportunity: a round table with 160 people! Let’s get started right away. In this session I would like to discuss the work we’ve done for SAMM on secure software development specifically for Agile. My goal is to share knowledge. I will be talking about the guidance but I also invite you, to share your challenges, your best practices, your lessons learned, so that we can learn from each other. There’s a good bunch of people joining today, so let’s benefit from that wisdom of the crowd. Because I definitely am not claiming to have all the answers. So, please, during the session, use the chat to pose questions, comments and to respond to the questions that I will have for you. Nessim will be so kind to assist me with processing your input. But let me first introduce myself.
I have been at the frontpage of OWASP for a while now. Not because I’m important, but by pure coincidence. It’s a picture of me presenting at Global Appsec Amsterdam.
SIG measures quality of software and guide organisations through shifting left, with offices in Amsterdam, New York, Antwerp, Copenhagen and Frankfurt.

ENISA : I worked on the recent report “Advancing software security in the EU”
You might have seen me or my team at one of the EU Appsec events. Let’s hope we can start doing that again in the future.
At the Integration project I work with Spyros, Elie and the team on providing overview over OWASP projects and an initiative called Common Requirement Enumeration where we link standards inside and outside OWASP at the level of requirements/controls.

But. Enough about me. Let’s talk about Agile secure development.

---

**Intro - The Integration project**

<table>
<thead>
<tr>
<th>Organization</th>
<th>Training</th>
<th>Requirements</th>
<th>Design</th>
<th>Implement</th>
<th>Verify</th>
<th>Respond</th>
</tr>
</thead>
<tbody>
<tr>
<td>- SAMM</td>
<td>- Webgoat</td>
<td>- Pyra</td>
<td>- HTML sanitizer</td>
<td>- Tools:</td>
<td>- Tools:</td>
<td>- Tools:</td>
</tr>
<tr>
<td>- Security champions playbook</td>
<td>- Shepherd</td>
<td>- Test dragon</td>
<td>- OWASP</td>
<td>- Dependency check</td>
<td>- MoBiSec</td>
<td>- Malicious activity report</td>
</tr>
<tr>
<td>- Pipeline tools</td>
<td>- Develop</td>
<td>- Technology boxes</td>
<td>- OWASP</td>
<td>- ZAP</td>
<td>- Code analysis</td>
<td>- Malicious activity report</td>
</tr>
<tr>
<td>- Glue</td>
<td></td>
<td>- Threat modeling</td>
<td>- OWASP</td>
<td>- Code analysis</td>
<td>- Code analysis</td>
<td>- Malicious activity report</td>
</tr>
</tbody>
</table>

owasp.org/www-project-integration-standards/
When I do a presentation on software development I typically would show this diagram. And through the years I increasingly got complaints and people correcting me, saying that this is completely not agile, and thereby everything I say about software engineering will be incorrect. So I often did this:
And it does Agile more justice, but it doesn’t provide the guidance that is needed. Even if I would draw a politically correct diagram, that would resonate with Devops, Waterfall, Agile and everybody:
... it would do justice to the nuances on a high level, but still wouldn’t answer the many questions that Agile teams have. And there are a lot of questions, because:
Agile security is hard

“We have no time for security in the short sprints, so we have security sprints.”

“The demo needs to show business features”

“We run these tools, so we’re good”

“Everything the developers should know is on the wiki. Let’s hope the best for the next pentest in 6 months”

“We made John responsible for security, so now it’s no longer my problem”

Agile security is hard, as illustrated by these quotes from some of the organizations I worked with.

*No time: So, traditionally, quality assurance has been a carefully designed single phase in software development, with its own bureaucracy. Of course there’s no time for that in sprints. And of course you can focus some sprints on security features, but also the non-functional security should be built in continuously. How else are you going to deliver working software? ‘Working’ doesn’t mean that it has the features that it needs to have. It needs to work in the real world. So the key is to find a way to make security fit.

*Demo: I often see an overfocus on features caused by the idea that sprint demos are about functions only. It’s important for the business to make sure security is regarded as a business value as well.

*Tools: using tools is a good practice but it’s important to understand the limitations of these tools.
Wiki: gathering requirements and guidelines is a good thing, but that’s not where it ends. You need an effective way for the team to get access to the relevant requirements based on the type of task they are working on. Having them on a wiki or in a document will not suffice.

There appears to be a strong need in the industry for guidance on how to make secure software development work in an Agile environment. So, how can we make it work?
Let’s have a quick look at what we are dealing with:

“How to make it work? Let’s look it up..

“Tracking and dealing with nonfunctional requirements like security and reliability is an unsolved problem in Agile development.”

So much for looking it up.

Audience: What sources are you using for process guidance?
Security in Agile (SCRUM)

Product Backlog → Sprint Planning → Sprint Backlog

Plan and prepare security

Sprint

Sprint review / retrospective → Shippable product

Daily Scrum

Implement security
You do many short iterations and every iteration you want to deliver software that is ready, with security built in.
So how do you squeeze all the necessary activities into a sprint: requirement selection, threat modelling, verification? What do you do with stories, with abuse stories and with the Definition Of Done? How do you get rid of the bureaucracy that is sometimes involved in how security teams and developers co-operate?
So, SAMM discusses business functions and practices, but at the same time it aims to be agnostic of the type of development approach, which is why Agile was not covered. This is a good decision to make the model elegant.
And therefore:
Agile guidance explains how SAMM practices work for Agile, in the form of best practices and pitfalls.
Since April 2018, I have been working on this project, in collaboration with the SAMM working group, clients and industry peers (eg. Michael Kuipers from Centric and Eric Nieuwland from Ictu). We did this based on our own experiences, by studying many organisations on what works and what doesn’t work, by doing observations, interviews and by looking into the many publications on this topic.

This got initiated because I was working on a Dutch government guideline on Agile security (CIP’s Agile Security Management) that was unknowingly repeating the work done withing SAMM and wanted to add agile best practices on top. I decided to go talk to the SAMM group and convince the government that the guideline should refer to SAMM and that we would extend SAMM with agile guidelines that we would then also add and translate to the Dutch publication.

I will show you a detailed example of guidance later, and we’ll discuss the main principles behind the guidance. But first: what does it look like and what does it cover.
This integration in the SAMM website was released recently, with the great help of Patricia Duarte.
Overview

This document explores how SAMM security practices work for Agile to attain the continuous building or of sufficient security during software development. It’s structured in the form of best practices and pitfalls.

Why SAMM Agile guidance

The Software Assurance Maturity Model (SAMM) is an OWASP flagship project on how to set up and grow a secure development process. It is agnostic of the development approach, which is why Agile is not explicitly involved. Nonetheless, there appears to be a strong need in the industry for guidance on how to make secure software development work in an Agile environment. How do you separate all the necessary activities into a sprint, i.e. commit a bit more than just a technical activity, and what are the implications for your current security strategy?

1. It provides questions on how the activity is slightly different for Agile.
2. It further provides more detail on how to implement activities in Agile (for example: how you make code reviews incremental), and what pitfalls to watch out for. Many introductions are made to practices when doing software security, especially in Agile situations.

History
Next step is to implement small links from the SAMM core model to Agile guidance sections. This allows the core model to stay nice and clean.
Agile secure development knows many pitfalls

#isolatedsecurity #pickduringplanning
#relyonthechampion #abusestorygalore
#silentchampion #skipmanualltest
#onlythreatmodel #latepentest
#requirementoverload #abuseCOS
#forgettheframework #skipsecurityrequirements

Audience: What are mistakes that you’d like to warn us for?
Of course: you need to automate testing as much as possible, both static and dynamic testing. So you need tools to scan your code and tools that try to break into your system. But then you’re not done because there’s still a lot of manual work that needs to take place.

The trick is to rely on expertise as little as possible by being more efficient, also to prevent people burning out. You sometimes here stories of one security expert for every 50 developers.

One way to become more efficient is to reduce the size of the whole security problem by: Building on proven technology: so you save time but mostly: you have less security to worry about.

Manual verification: code review and penetration testing

Close collaboration instead of quality gates. You don’t have time to go back and forth between development and test.

Instructions -> so the team doesn’t need constant expert help
The goal is to minimize cognitive load

In other words: there is no time in a sprint to get the development work done and then send it to security for testing. You want to shift left. This puts security professionals much more in an advisory and supportive role than in a role of a quality gate. This also means that all developers and testers should own security, including the product owner.

Let’s look at one of the key things you can do to streamline the manual work, and that has to do with Threat analysis vs hygiene.
This illustrates minimizing the dependency on security expertise, and the dependency of being smart and alert all the time.

If you are having dinner you first wash your hands. Such situational instructions are there to save time in thinking about what you need to do, and they also prevent things to be forgotten.

Please note that threat modelling is very useful. The point here is that if you can make it happen, it is helpful to not be depending on such analysis for trivial things. You shouldn’t need to threat model everything because many security countermeasures are always required.

There is a section by the way on incremental threat modeling in the Agile guidance.
Let’s have a look at an example guidance: how you deal with requirements.

The key is to minimize the daily instructions that developers need to work with, and to let as many as possible depend on the type of task they are working on, for them to be effective.

Audience question: channel = using a wiki, or cards, or tools like SKF or iriuskrusk. Or did you build something yourself? I suspect that Excel is going to be one of the answers.
The treatment of stories happens during creation of the story, and during backlog refinement. It should not be done during the planning meeting.

Definition of Ready: security checks should be there,
Definition of Done: security validation needs to be done
With that we are done discussing the Agile guidance. Thank you very much for your joining and if there's time we can take some more questions.

I would welcome your feedback on the notes and your own experiences very much. One final word from me: we talked about security, and this is an application security conference, but everything I have been saying should go for the other quality aspects of software engineering: maintainability, privacy, reliability, and performance efficiency. My personal overarching advice is to see if you can make your security program a software quality program. Good luck. Thank you.