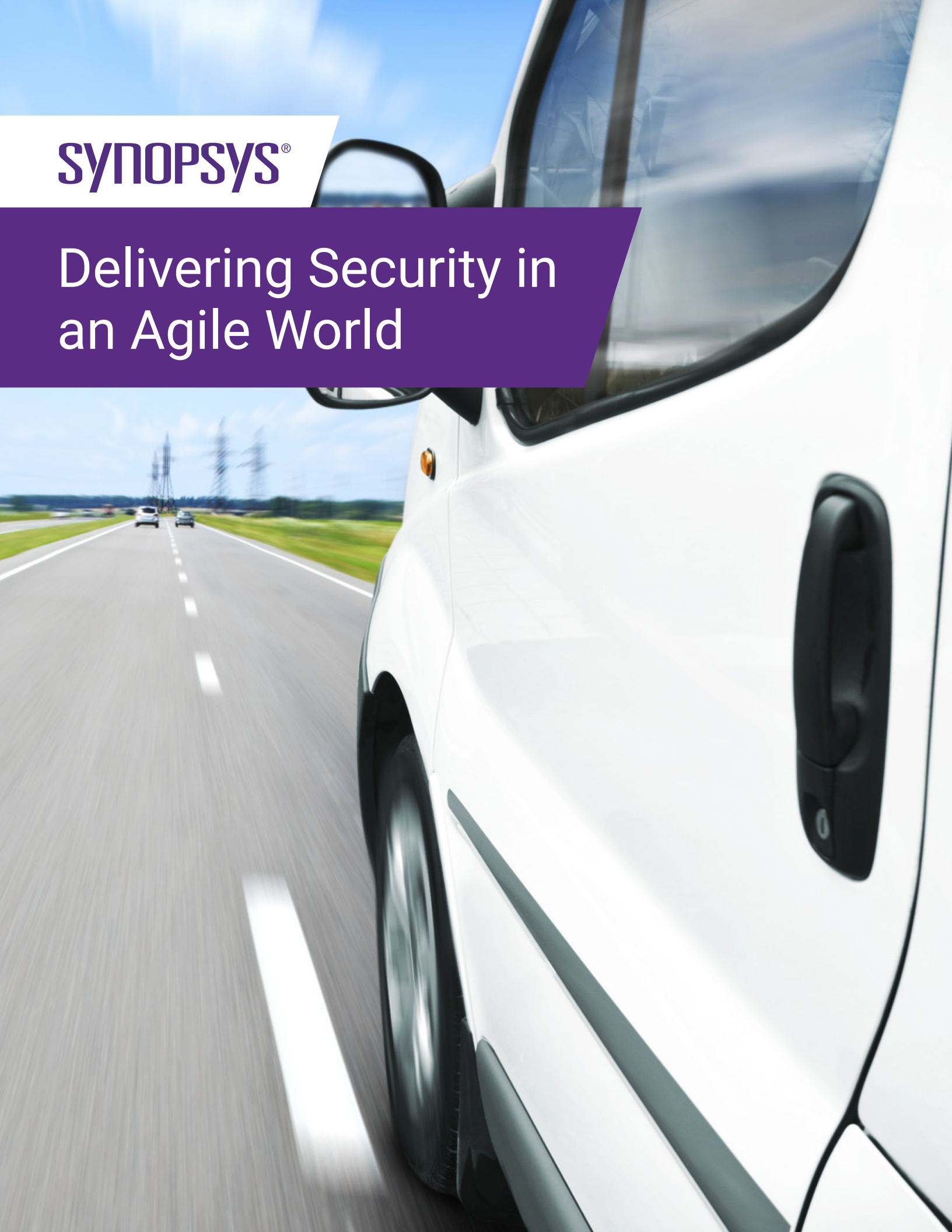
SYNOPSYS®

Delivering Security in an Agile World



Get your priorities straight

If you want to deliver software features in an agile way, it's critical to ensure the software you're delivering is secure. To understand how this works, imagine that your software development process is a shipping business. Poof! Now, instead of delivering software, you're delivering packages, albeit really important packages.

Of course you want to be the fastest, most reliable shipping service out there, so you adopt an agile model.

What makes Agile so great for your business? It's the way it goes about "filling the van." Here's what we mean by that.

Agile software teams usually have a backlog of user stories or features that they want to add to their software, right? In your imaginary shipping service, that backlog is your warehouse full of packages you need to deliver. Each package represents a feature that someone wants in your software. Some of these packages are very important and must be delivered immediately. Others don't have a critical delivery date, so you can squeeze them in whenever possible.

Each morning, you put out as many packages as can fit into the van, starting with the ones that need to be delivered that day. Your driver comes in, picks up the packages you've set out, and figures out the most efficient route to get them all delivered that day. Think of the driver's delivery day as a two- to four-week sprint in agile software development.

Keep on keepin’ on

As soon as the delivery van is packed up, it rolls out of the warehouse to begin its daily run. Only after the van has left do you realize that you intended to swap out a lower-priority package with a high-priority one that needs to be delivered as soon as possible. Now what do you do? Do you call the driver back? No way. The route is optimized to deliver the most packages in the least amount of time. Veering from that plan would slow down the process, and you’d risk not getting all the packages delivered on time. You must put it aside to add to the next run.

Likewise, with agile sprints, if you’re a week into your sprint, telling your team to drop what they’re doing and start something new would slow down the process and push back the release date. The better course of action is to add the new feature to the next sprint instead.

The driver delivering packages to the right addresses, on time, without losing them or breaking them is just like a software development team that delivers a well-defined set of features by the predetermined release date—only far less probable (at least in my neighborhood).

Don’t cram the van, man

You may not be able to change what’s going on right now, but you do have the freedom to change the order of what’s coming up next.

So while your driver is out delivering packages, you can reorganize and reschedule the remaining boxes in the warehouse and prepare them for tomorrow’s delivery. Similarly, agile software owners can pick features for future sprints and put them in order based on urgency.

When you’re selecting what items to deliver each day, it’s important to remember that the van can carry only so much stuff at a time. Even if you stack packages on the passenger’s seat or shove them underneath, eventually you’re going to run out of room. Likewise, agile development teams have a notion of “how big the van is.” Teams measure their velocity (how much work they can get done in a typical sprint), and they estimate the sizes of features and user stories (like the sizes of boxes). So if your team knows they can do 48 points of work in a sprint (or whatever metric they use), the sprint is no more stretchable than the sides of a delivery van.

If all your eggs don't fit in one basket

What if a box is so big that it can't be delivered in one van? Similarly, a software feature might be so big or complicated that you can't deliver the whole thing in a single sprint. No big deal; it happens all the time. Just open up the box and try to find smaller boxes inside that do fit in the van and can be shipped separately. In agile development, that translates to delivering the total feature set over several "delivery days" or sprints.

Of course, not every feature can be broken down easily. If someone has ordered a dozen eggs, but your van has room for only 10, that's fine. You can deliver 10 today and 2 tomorrow. However, you can't deliver half an egg. You can't deliver 10.5 eggs today and 1.5 eggs tomorrow, at least not without getting really messy. Likewise, there are limits to how you can break down software features. This is sometimes a perplexing limitation to nonprogrammers. Things that sound impossible to break down turn out to be easy, and things that sound easy to break down turn out to be hard.

Handle with care

You have a warehouse full of boxes with merchandise that needs to go to customers who are paying you. However, to get the boxes from point A to point B safely, you must implement security measures along the way.

Taking the time to fill the empty space in each box with packing peanuts and bubble wrap can be worth the extra time and effort. Security precautions on the part of the driver when loading and unloading the van is also worthwhile. Placing the heavy boxes on the bottom of the van and those labeled FRAGILE on top ensures the packages arrive on time and with the contents delivered as expected. The packing materials and driver precautions are like the security designs, security features, and security activities applied in software development.

Imagine how many items would arrive damaged every day if precautionary measures weren't implemented. When packages are damaged, the delivery of a replacement item has to be scheduled, which will take up space in the van on a future delivery day. Likewise, it takes time to implement corrections to insecure software features in future sprints.

The accumulation of replacement items that need to be delivered is called "technical debt." Each sprint, you typically plan to pay off some technical debt. Some sprints involve only new features, and other sprints focus heavily on paying off technical debt. Spending most of a sprint paying off technical security debt is like spending a whole day delivering replacements for items that arrived at their destinations broken. But how much packaging material is enough? Similarly, how much security work should you be doing right now, and how much can be saved for later?

Some boxes are obviously at more risk—those labeled FRAGILE that you've carefully placed on top of the heavy boxes in the van, for instance. These are like the software features in the backlog that are obviously security- related—they may deal with personal information, passwords, or confidential documents. But sometimes it's not obvious how valuable boxes are or how bad it would be if they arrived broken. One package may contain some cheap glass tchotchkes while another may carry an irreplaceable Ming vase or lovely leg lamp (which is definitely fra-gee-lay). Getting a handle on how vulnerable the boxes are and how much time and company money it will take to replace them is critical to figuring out how you'll need to secure each package throughout the shipping process.

On top of the delay, if you break that Ming vase, you're going to have one seriously miffed customer. At best, they'll just use a different delivery service in the future. Worse, they'll sue you for everything you've got. The same goes for insecure software. Boy, do you risk upsetting your customers if hackers use your software to steal their sensitive information!

When life gives you golf balls

Golf balls are round and small, which makes them difficult to package. Similarly, it is often difficult for an agile development team to deal with the kinds of artifacts that security teams produce. A PDF report from a code scanning tool might turn out 25,000 results. That's like someone bringing you a massive crate full of 25,000 golf balls, each individually addressed. Sure, you have plenty of room for them in the van, but no one is going to want to sit and sort through every golf ball one at a time—no matter how much sugar on top you add to your pretty please. It's absurdly inefficient.

Then you realize that all the white balls (e.g., cross-site scripting) go to one place, all the yellow ones (e.g., SQL injection) go to another place, and those four random pink ones (e.g., server configuration issues from a third-party software vendor) go to yet another address. Someone needs to sort—or triage—the golf

If security teams are going to interact with agile development teams, they have to deliver their results in a format that developers can use. Otherwise, it's like delivering a crate full of golf balls and asking someone to sort them.

balls to figure out how to deliver them efficiently. Similarly, someone has to triage the security defects and figure out which ones are legitimate, how to deliver the fixes, and what the fixes entail.

Security issues should be packaged in a way that makes it easier for developers to deliver. For example, if you put all the white golf balls in a big box and address it as "cross-site scripting in the e-commerce platform," the shipping company doesn't have to sift out all the white golf balls from the crate and pack and ship them. Likewise, packaging a bunch of related security issues into a Jira ticket or a card on the Kanban board makes it easier for the software team to prioritize and deliver it in a sprint.

Here are three tips for delivering security successfully in an agile world.

1. Security needs to meet developers where they work.

Provide security assessment results in a format that is consumable by the development team. Track those findings in a working system that the developers use daily. Maybe it's Jira, the Kanban board, or whatever sprint planning tool the developers use. Security teams might have their own systems as well (risk registers, vulnerability lists, threat models, etc.). But if you want the developers to spend time and effort on something, that something has to appear on their task list in a way they can use.

2. Agile software development methods work.

They are not sloppy, chaotic, or fast and loose. But the security artifacts that security people often want (threat models, architecture diagrams, risk lists, etc.) are not natural by-products of many agile teams' activities. It's not for lack of caring. Agile teams build things that get them directly to their goals. If you put security on their list of goals, then they will build things that get them to security.

3. The goal is to create secure software.

There is no need to make security artifacts for the sake of making security artifacts.

Anatomy of agile delivery

In agile software development	In delivering boxes of goods	Comments
Creating features	Delivering a box of goods from point A to point B	The more you ship right, the happier everyone is.
Creating secure features	Delivering a box of goods from point A to point B without breaking them	If you deliver insecure software features, you will end up delivering a correction in a future sprint. If it's secure to start with, you are more efficient.
A sprint	A day spent driving the van delivering goods	Once the van leaves the warehouse, or once the sprint starts, it's a really terrible idea to call the driver back and change what's on the van.
Software development team	The driver delivering the boxes	If you deliver something broken, you'll have to schedule a replacement in a future delivery/sprint.
The team's velocity	The van's capacity	The van, like a sprint, has a fixed size. There's only so much stuff that fits.
Feature backlog	Boxes waiting to be scheduled for delivery in the van	You can reorder, repack, and reschedule anything that hasn't been scheduled for delivery yet.
Kanban board, sprint plan, etc.	The manifest of boxes you plan to deliver	
Security controls, features, designs, etc.	Packaging materials to protect the goods	Too much packaging takes up valuable space to fit goods. Not enough packaging, and stuff leaks or breaks.
Technical debt	Boxes that have been waiting a long time to be shipped	Some features, security controls, and so on really do have to be shipped someday. If you put them off, they become technical debt.

If you want to deliver software features in an agile way, it's critical to ensure the software you're delivering is secure.

Learn more

The Synopsys difference

Synopsys Software Integrity Group helps organizations build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior. With a combination of industry-leading tools, services, and expertise, only Synopsys helps organizations maximize security and quality in DevSecOps and throughout the software development life cycle.

For more information go to www.synopsys.com/software.

Synopsys, Inc.
185 Berry Street, Suite 6500
San Francisco, CA 94107 USA

U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com

SYNOPSYS®
Silicon to Software™