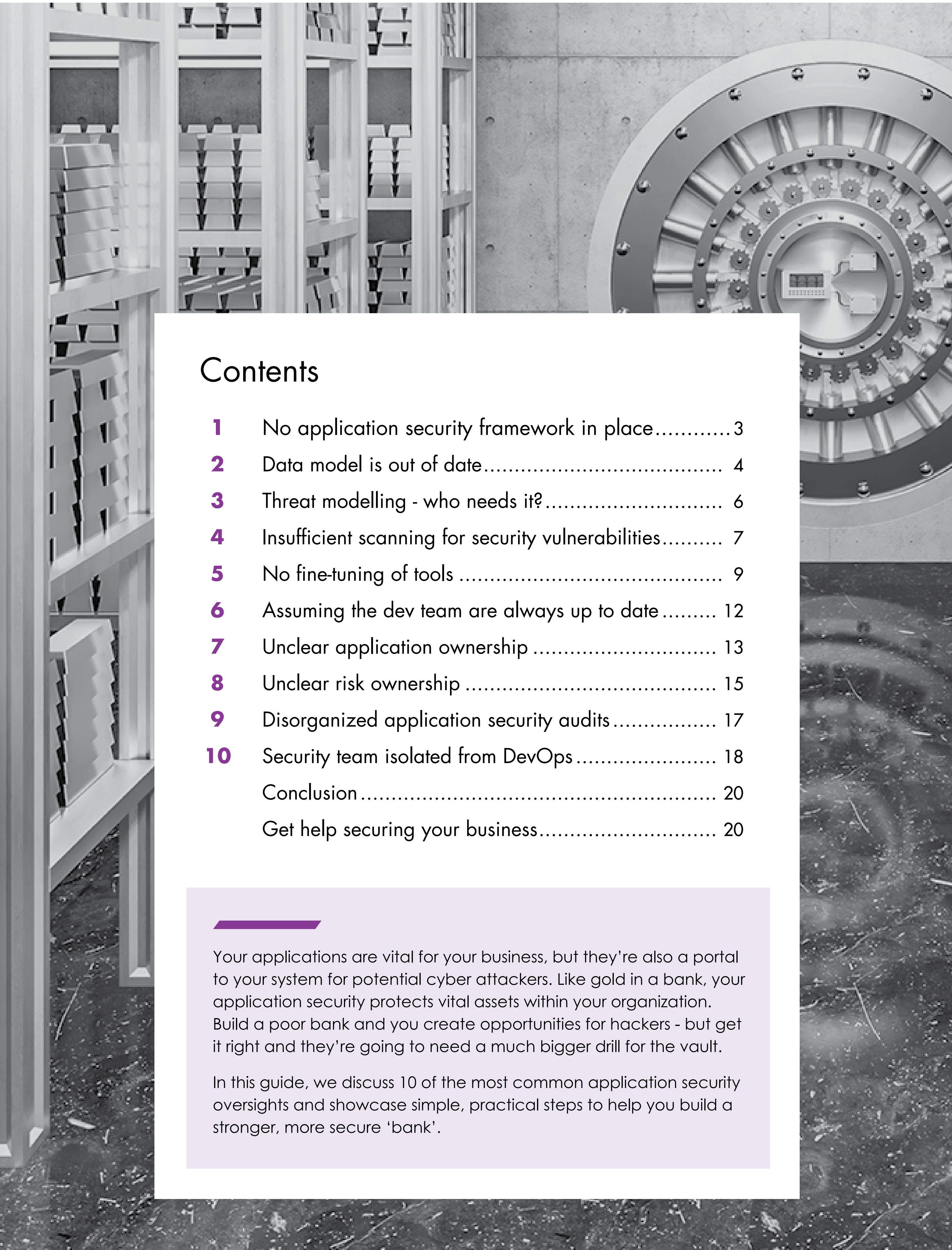
10 COMMON APPLICATION SECURITY OVERSIGHTS - AND WHAT TO DO ABOUT THEM

www.dig8ital.com

dig8ital



Contents

Your applications are vital for your business, but they're also a portal to your system for potential cyber attackers. Like gold in a bank, your application security protects vital assets within your organization. Build a poor bank and you create opportunities for hackers - but get it right and they're going to need a much bigger drill for the vault.

In this guide, we discuss 10 of the most common application security oversights and showcase simple, practical steps to help you build a stronger, more secure 'bank'.

1

No application security framework in place

You've built your business, assembled your dev team, and have decided to take security seriously – but you got stuck into building bank walls without erecting a frame behind them. Eventually you'll stack those bricks too high and they'll just blow over in the wind (and there's no chance they'll support a big metal vault door).

HOW TO FIX THIS OVERSIGHT:

Which framework to choose? Look for one that already utilizes some of the processes you're used to, so it's not an entirely alien way to work. Although any good framework is better than none at all.

You need an application security framework in place before building your bank; it will form the blueprint for everything we put inside it.

Application security frameworks are like comprehensive maturity checklists.

They offer important controls in every phase of the development life cycle, almost like step-by-step instructions full of best practices.

Frameworks are the foundation of good application security architecture; they're the plan behind your organization, and the strong frame behind the walls.

Everything else in this list follows having a framework in place.

There are many known application security frameworks already out there, for example:

- **OWASP SAMM v2:** Which helps organizations assess, formulate and implement software security strategies with a self-assessment model.
- **BSIMM v11:** Helps your team understand, plan and measure new software security initiatives with real-world data.
- **ISF AppSec Framework:** Helps companies improve security at all stages of the dev cycle by outlining best practice guidelines and offering support.

HOW DIG8ITAL DOES THINGS DIFFERENTLY

At dig8ital we've taken the best aspects of each of the common application security frameworks and combined them into a 'best of both worlds' service; by taking components from each framework, we can offer a more versatile, tailored service without sacrificing quality. That way we help build a 'bank' that's designed to suit each and every customer individually, based on their own unique requirements.

2

Data model is out of date

Customers, employees, partners, suppliers – so much data all processed within your business. Each set has its own database, and your various apps are constantly accessing these repositories or adding more information. But where is all that data? How is it protected? Which people and apps have access? This is the gold that your 'bank' is protecting. If you don't know where it's all stored and who has access, it could be at risk.

HOW TO FIX THIS OVERSIGHT:

If you don't have one already, develop a data flow diagram (DFD). This model will represent the flow of steps it takes to transfer data from an input to the repository.

Everytime you add a service or change an app, go back to the diagram and see where it fits in the model.

Here's one of the big issues: With so much data coming and going from your apps, it's common for information to become duplicated.

Each piece of duplicated data is a cost - it has to be stored somewhere. It's also a vulnerability, as now you're protecting it in two locations instead of one!

Duplicated data makes for a common target for would-be data robbers, as duplicates are often less protected than the original databases.
- Imagine storing gold in your vault, but then also a filing cabinet in the back office.

All of your apps should be protecting the data they process and store, but they can't if they don't know how to.

Make sure that the new change doesn't violate your data flow and put valuable gold somewhere unprotected!

GAINING EXTRA PROTECTION

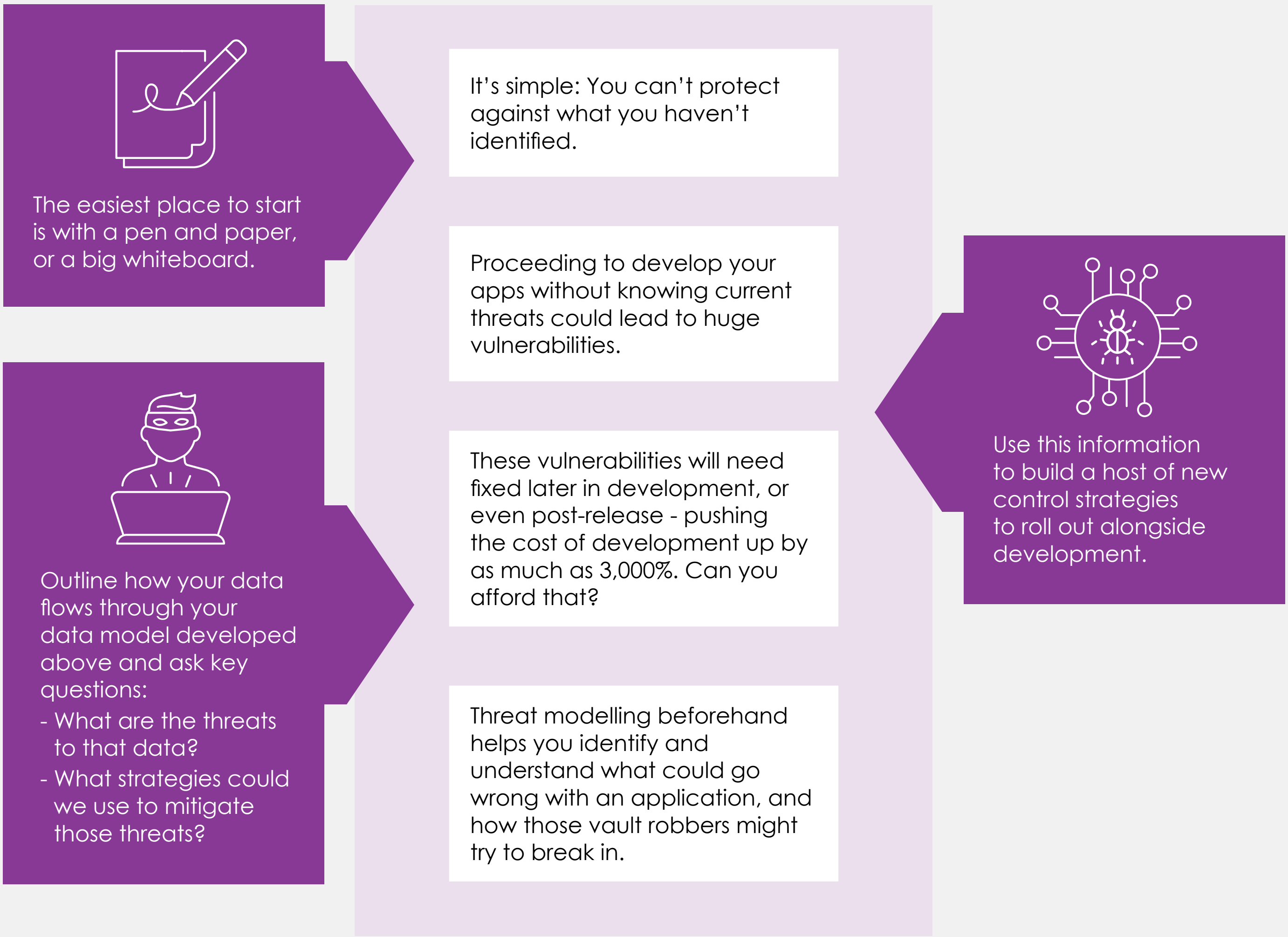
For that extra bit of protection, it's best practice to assign a data owner to each database so that someone is accountable and keeping an eye on it. For instance, HR could own employee data, the COO could take on customer data, and so on. For more, see page 10.

3

Threat modelling - who needs it?

You've been hiring the best talent you can find, so collectively your organization has extensive experience and you feel you know how to protect your vault. Your developers can guess the different ways robbers might try to break in, and have built controls to mitigate that risk. But what if they've missed something? And what if those threats change over time?

HOW TO FIX THIS OVERSIGHT:



The easiest place to start is with a pen and paper, or a big whiteboard.

It's simple: You can't protect against what you haven't identified.

Proceeding to develop your apps without knowing current threats could lead to huge vulnerabilities.

Use this information to build a host of new control strategies to roll out alongside development.

These vulnerabilities will need fixed later in development, or even post-release - pushing the cost of development up by as much as 3,000%. Can you afford that?

Outline how your data flows through your data model developed above and ask key questions:

- What are the threats to that data?
- What strategies could we use to mitigate those threats?

Threat modelling beforehand helps you identify and understand what could go wrong with an application, and how those vault robbers might try to break in.

COMMON THREAT MODELLING TOOLS

You may find that as your business grows, a whiteboard is too simple. Luckily, there are plenty of digital tools to help. Some of the common tools include:

1. Microsoft Threat Modeling Tool **2.** ThreatModeler **3.** OWASP Threat Dragon **4.** OWASP PyTM
5. IriusRisk **6.** SD Elements **7.** Tutamantic

4

Insufficient scanning for security vulnerabilities

So you know how the bank robbers will try to break in and you have a book of control strategies - it's time to let your dev team get started. But what if, during the development process, someone introduces an error in the code, or a hidden bug? Who's going to spot it? Who has time to scan each line of code to check?

HOW TO FIX THIS OVERSIGHT:

Application security vulnerability scanning translates threats and control strategies into programming language which can be checked and improved.

Without knowing how to scan for vulnerabilities, even after everything you've done you could still build a bank vault full of holes.

The reality is, coding – even under the guidance of a control strategy – can often introduce new and unseen errors.

These tools look through your coding patterns and tell you when they aren't secure.

They can also be customised to better suit your unique dev team – see page 7 for more.

COMMON VULNERABILITY SCANNING TOOLS

1. Raxis **2.** RIPS Technologies **3.** PVS-Studio **4.** Kiuwan **5.** Reshift **6.** Embold
7. CodeScene Behavioral Code Analysis **8.** Visual Expert **9.** Veracode **10.** Fortify Static Code Analyze

5

No fine-tuning of tools

Now your dev team has their list of strategies and an array of tools to scan for vulnerabilities. But everytime they scan the bank, their tool pings up a number of errors and fixing them only adds three more. They've agreed that most of the identified errors aren't actually a problem, so it's mostly just wasting effort. Over time, they use the scan less and less because, quite honestly, it's annoying to use.

HOW TO FIX THIS OVERSIGHT:

Code scanning tools need to be trained and fine-tuned to work effectively - to reduce those false positives and identify more critical vulnerabilities and architectural roots of vulnerabilities.

There will be a number of common errors in your code that you've agreed aren't a problem, so you can switch these alerts off.

There's no such thing as a magic wand - tools have to be used to be effective, which means knowing how to use them.

With so many apps helping you write and scan code, there will likely be a lot of security alerts popping up. Some critical, some minor. Either way, it's a lot of extra work.

If a scanning tool just creates more work, your bank builders might not want to use it. Fair enough!

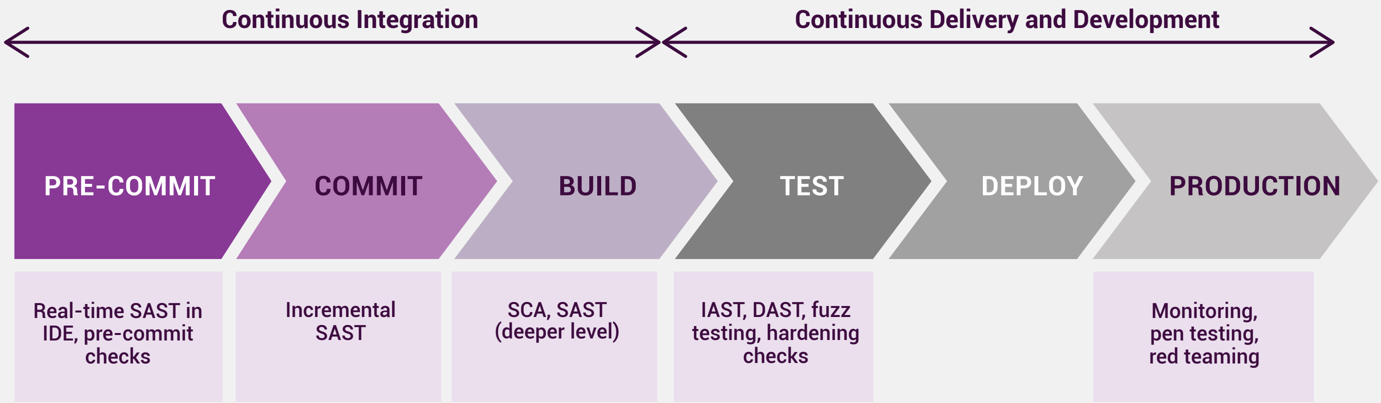
As you suppress superficial problems, you'll start to identify root cause problems. This is what you need.

Always look for the root cause. Three or four alerts could stem from one problem - by tackling that and ignoring the non-issues, you will work more effectively (and so will the tool).

5

Application Security Tools in the CI/CD Pipeline

From real-time static application security testing (aka SAST) to red teaming, there are a lot of tools and processes involved in the development of an app. While it may feel like a pain sometimes to sit down and work through each of these scans and penetrations, learning exactly where your vulnerabilities lie and the root cause of these problems can pay dividends in the long-run. As we mentioned earlier, leaving security bugs too late can increase your costs by 3,000% - nobody wants that!



Continuous Integration

Continuous Delivery and Development

PRE-COMMIT	COMMIT	BUILD	TEST	DEPLOY	PRODUCTION
Real-time SAST in IDE, pre-commit checks	Incremental SAST	SCA, SAST (deeper level)	IAST, DAST, fuzz testing, hardening checks		Monitoring, pen testing, red teaming

6

Assuming the dev team are always up to date

Your dev team are experts in their field, highly educated and fully qualified, with decades of experience between them. Their careers are paved with proud banks and highly secure vaults, and together they've thwarted more than a few robberies because of their skill. So ... what do they need extra training for?

HOW TO FIX THIS OVERSIGHT:

It's a common assumption to think that developers are always up to date with the latest trends in 'bank robbing techniques' (aka cyber security).

But, the reality is that most developers are too busy to sit and retrain every year. However, bank robbers are always inventing new ways to break in.

In addition, developers rarely get to see how their products are used. There's a disconnect between their work and the Ops team or the user. Do people enjoy being in the bank? Many devs wouldn't know.

This can lead to avoidable errors, or not knowing how to fix specific problems - not to mention simply not improving inefficiencies because no one has ever told them said inefficiencies exist. You simply can't know what you haven't been taught.

TRAINING WEBINARS

Invest in training webinars based on the technology that your devs are using. What is the latest version? What languages aren't secure anymore? What are the current trends?

USER FEEDBACK

Allow devs a day or two to spend with real app users to see if there are possible ways to abuse and misuse their products. Any discovered cases can be added to the next security update.

CONFERENCES

Sending developers to conferences is a great way to get them engaged in learning. While conferences aren't as in-depth as training sessions, they build awareness and give your team new things to look into.

7

Unclear application ownership

After putting so much work into building our bank, securing the vault and scanning for vulnerabilities, the bank's security still needs to be managed over its lifespan in case something goes wrong in the future. But whenever the prospect of managing that asset comes up, the response is always the same: "Not my job."

HOW TO FIX THIS OVERSIGHT:

Develop a new asset management bible that defines what an application owner should look like and how the role is designated.

In addition, consider the scope of their accountability so each owner clearly knows their own boundaries, and when to seek help (and where).

If ownership of an application is unclear, there's no accountability - no one taking care of it.

Most people won't take on this responsibility out of the goodness of their heart. They're not bad workers, just busy!

But without accountability, no one will keep the app updated, secure and functional. If something goes wrong, it can take a lot longer to identify and fix.

Every moment wasted fixing an issue is money spent, or customers displeased - not to mention regulators.

You can also note down the scope of their budget, for when they need to draw on funds to make important changes.

Asset owners should be assigned based on this guidebook. Think carefully about who has the experience, seniority and trust to own an asset, and make their position formal.

8

Unclear risk ownership

Your bank and its vault will always be at risk, and those risks could each have potentially devastating consequences to the business or its customers. But when no one knows who owns these risks, they might assume that everyone else does - which means no one is keeping an eye on them.

HOW TO FIX THIS OVERSIGHT:

Like before, you need a risk ownership bible that defines how to identify risk owners, how to designate them, their scope and their budget allowance.

Also like before, this must be a serious, formal process and not assigned on an ad hoc basis.

Risk ownership has the same problem as asset ownership when it comes to lack of accountability and consequences.

But in the case of risk specifically, it can be worse - risk is sometimes a nebulous, vague thing that employees have little prior knowledge of. Developers may sometimes spot problems with an information asset, but rarely risk.

Again, with no accountability there's no control, no security, and it takes longer to find and fix problems.

Risk owners must also be aware of a few extra factors:

- The risk appetite of the business.
- The organization's threat landscape.
- Current threat intelligence related to their role.
- How risk relates to the organization's objectives.

9

Disorganized application security audits

These days, companies have a lot of internal policies to follow - you can't build a bank without the paperwork. But if each piece of policy has an auditor trailing after it, eventually these auditors are going to be asking the same developers the same questions over and over, and you can just guess how said developers will respond to that.

HOW TO FIX THIS OVERSIGHT:

Take the time to audit your governance requirements - what regulations apply to your apps, what steps have you already taken to secure them and what steps remain?

Invest in governance, risk and compliance (GRC) software to help manage and automate this process.

It's common for organizations to build separate units to take care of important legislation (i.e. the GDPR) relating to the security of their data and/or quality of their apps.

But these assignments don't often take into consideration that there may already be a unit looking after some of the same issues, creating a duplication of effort.

Disorganized application security audits like this waste people's time. For example: In an automotive company, both the GDPR team and WP.29 team need to ask asset owners about customer data and its security. Those asset owners have to sit through two interviews for almost the same information.

Ensure that all governance teams are aligned on the organization's requirements, have been assigned access to the GRC software and are well-informed of what each other compliance unit is up to.

When legislation is updated, ensure that auditors first investigate if they have existing data that could be reused, whether their own or another team's.

COMMON GRC TOOLS

1. SAP Risk Management **2.** Axio360 **3.** OneTrust GRC **4.** IBM OpenPages **5.** ServiceNow GRC

10

Security team isolated from DevOps

How can bank builders build a vault if they don't have security experts helping them? And then when the bank opens, who is going to protect the bank on an ongoing basis? When security isn't integrated into the CI/CD pipeline, it's either missed entirely or acts as a barrier towards the end of development (putting those costs up).

HOW TO FIX THIS OVERSIGHT:

Involve security right from the design phase of an app.

Keep them included at each subsequent phase. Instead of fixing errors all at once, you fix little errors as you go.

Keep returning to your threat model to ask "What could go wrong?" at each phase of development.

Stay up to date on current threats using OWASP's annual top 10 list.

With no up-to-date security expertise in the DevOps teams, your people can't fix vulnerabilities that they aren't made aware of.

Even when DevOps teams are trained in security, errors can still slip through - it's not their job, and deadlines are short.

If security is left to a gate at the last minute, it can halt production and send people scrambling to the drawing board - a very costly exercise.

Regularly feed lessons learned from Ops back to Sec, who can then help Dev improve their work. This creates an infinite feedback loop that makes your team stronger over time.

Adopt secure SDLC best practices with techniques such as secure code repositories, pair coding, static code analysis, software component analysis, and learning SAST, DAST and IAST.

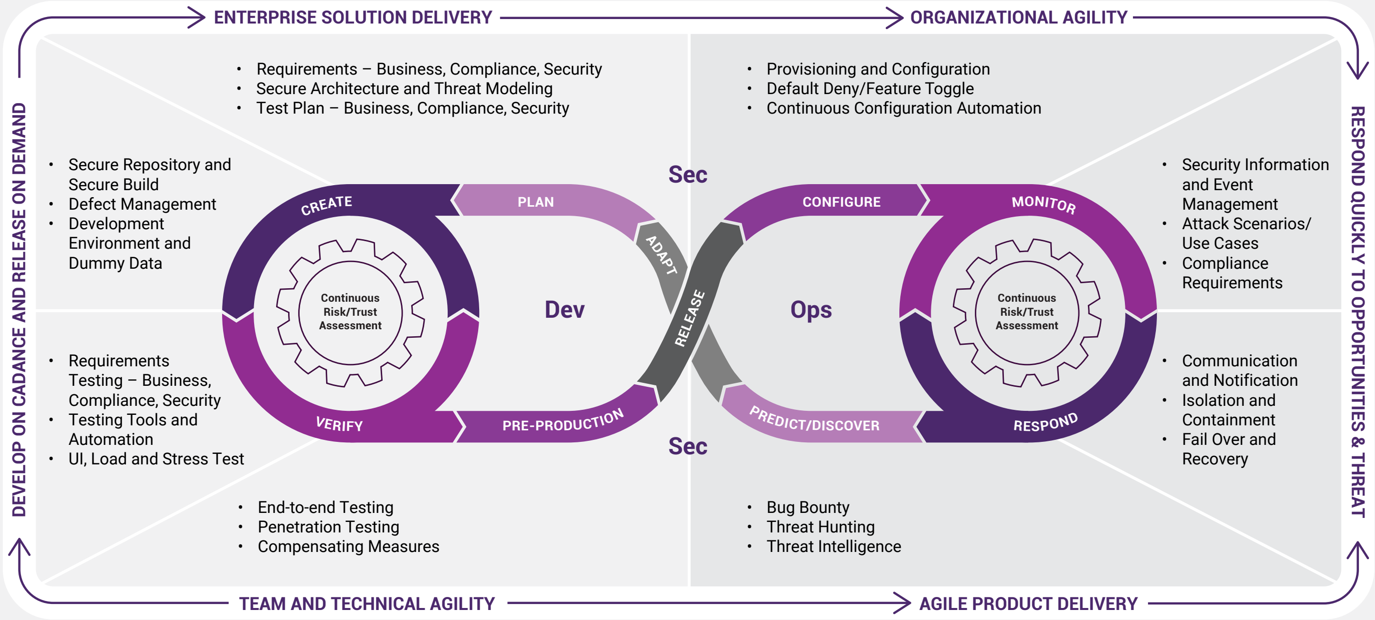
Consider going open source. It's not right for everyone, but having more eyes on your code could mean more people checking it for potential errors.

Identify skills gaps in the DevOps teams. Plugging these holes can reduce the size of security errors and turn people into security champions.

10

dig8ital's Application Security Framework

As we mentioned earlier, at dig8ital we've taken a 'best of both worlds' approach to application security and built a framework using best practices from the world's leading institutes. This helped us create the infinite DevSecOps loop, which as you can see seamlessly builds those vital security checks into the production line to create a constant loop of testing, monitoring and self-improvement, thus producing more secure applications.


ENTERPRISE SOLUTION DELIVERY
ORGANIZATIONAL AGILITY
DEVELOP ON CADANCE AND RELEASE ON DEMAND
RESPOND QUICKLY TO OPPORTUNITIES & THREAT
TEAM AND TECHNICAL AGILITY
AGILE PRODUCT DELIVERY
Sec
Sec
Dev
Ops
CREATE
PLAN
ADAPT
RELEASE
VERIFY
PRE-PRODUCTION
CONFIGURE
MONITOR
PREDICT/DISCOVER
RESPOND
Continuous Risk/Trust Assessment
Continuous Risk/Trust Assessment

- Requirements – Business, Compliance, Security
- Secure Architecture and Threat Modeling
- Test Plan – Business, Compliance, Security

- Provisioning and Configuration
- Default Deny/Feature Toggle
- Continuous Configuration Automation

- Secure Repository and Secure Build
- Defect Management
- Development Environment and Dummy Data

- Security Information and Event Management
- Attack Scenarios/ Use Cases
- Compliance Requirements

- Requirements Testing – Business, Compliance, Security
- Testing Tools and Automation
- UI, Load and Stress Test

- Communication and Notification
- Isolation and Containment
- Fail Over and Recovery

- End-to-end Testing
- Penetration Testing
- Compensating Measures

- Bug Bounty
- Threat Hunting
- Threat Intelligence

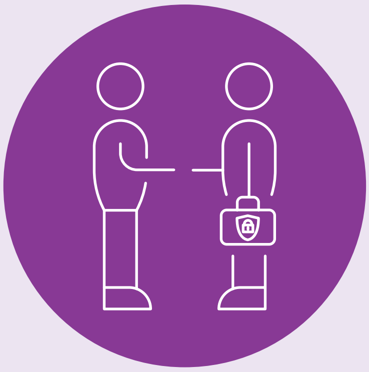
Conclusion

Your organization is like a bank, with critical assets that need protecting and a hoard of data stored in its vault. The key to building a strong bank that's resistant to robbers is to have a good plan - and stick to it.

Starting with a framework means you are following most if not all of what we've discussed in this list above. It's a checklist of best practices to follow, offering advice on strategies to use and how to use them, and will more than likely cover all of our points today.

In addition, knowledge is power. Finding those critical skills gaps and keeping team members up to date (and accountable) builds a highly cyber-aware culture and reduces the likelihood of someone making an avoidable mistake.

Get help securing your business



Even with the best advice and a checklist to follow, it's still hard to get application security right - and protect that 'bank' of yours.

This is where dig8ital comes in. We have vast expertise in the latest cyber security trends, including the evolving threat landscape and changing government legislation. Our security specialists have worked with some of Europe's biggest brands across sectors, and we're ready to help you, too.

To learn how we can build a better 'bank' together,
contact us today for a free maturity consultation.

W: dig8ital.com
E: contact@dig8ital.com

dig8ital