An Overview Into DevSecOps Written for Rootstrap, Inc.

By Ravi Das

Introduction – What is Dev Sec Ops?

As the world of Cybersecurity starts to become more complex and dynamic to levels never seen before, there is now paramount pressure that is placed upon the IT Security teams across Corporate America to increase their vigilance. It's not just from the standpoint of thwarting off the bad guys that are trying to break in, but it is also trying to predict what future variants could potentially look like down the horizon, so that lines of defenses can be beefed up accordingly.

But now, everybody has a stake in this proposition – all the way from the C-Suite to the administrative assistant. There was one group that has stayed relatively immune from falling under the microscopic eyes of Cybersecurity, but this is now no longer the case.

This group is the software development teams. All their job has so far been to develop and compile the source code for the Web application that they have been tasked to create, and ship it off to the customer, under budget and right on time.

Because of this, security has long been an issue which has remain largely ignored. As a result, the Cyberattacker is finding ways to covertly sneak into the backdoors that are left behind, and staying in for long periods of time, very often going unnoticed.

Then once they feel comfortable in the environment they are, they move in a lateral fashion, deploying malicious payloads along the way which even the traditional antivirus and antimalware packages cannot capture.

Or they could start a data exfiltration process, in which small bits of the PII datasets are slowly extracted, once again going unnoticed. Because of recent attacks (most notably that of the Solar Winds hack), software developers are now feeling the heat to make sure that the source code they compile is secure in every aspect possible.

Thus, this is where the acronym "DevSecOps" is starting to come into play. It stands for "Development, Security, & Operations". The primary goal of this is to introduce and deploy automated security mechanisms into the entire lifecycle of the software development process.

If security was ever a concern in the past, it was something that was done at the very end, in a very haphazard fashion. One of the primary goals of DevSecOps is to introduce it at every level of development, so that each software module is thoroughly tested before moving onto the next one. Thus, the cascading effect of un remediated vulnerabilities and gaps is greatly mitigated.

Another key strength of DevSecOps is that it integrates not only the software development teams, but also the IT Security and Operations teams as well into one cohesive unit. This brings an extra set of eyes to help make sure that the nothing in the security process gets overlooked.

In other words, the siloed approach is now fully eradicated, and it has now become a shared responsibility, which leads credence to the DevSecOps motto: "Software, Sooner, Safer".

This allows for robust and secure code to be delivered without slowing down the software development cycle. Put another way: "DevSecOps helps enterprises to innovate securely at speed and scale."

(SOURCE: 1).

How To Implement Security Into DevSecOps

It is important to note that implementing a Cybersecurity mindset into your software development process is not something that can be deployed anywhere at any time. It must start very early on, preferably even before the application project has even started.

But most importantly, this kind of thinking must be adopted by all of the departments in your business. It's not just the IT Security team that has to believe in this framework, every employee must, because everybody has a key stake in keeping your business safe and secure.

But as it relates to DevSecOps, this proactive mindset has to be formally acknowledged and embraced in the planning stages of the software development cycle. From there, it then transcends in a lateral fashion until the coding is all done, and the project is ready to hand off to the client. For purposes of this whitepaper, a hypothetical software development process can be represented as follows:

1) Planning
2) Defining the Requirements
3) Designing & Prototyping
4) Development
5) Testing
6) Deployment

The above can formally be called the "Secure Software Development Lifecycle", or "S-SDLC" for short. Each step is reviewed as follows:

<u>Planning</u>

In this step, you have been assigned a project, and are in the process of assembling your software development together. This phase of the S-SDLC can be viewed as a macro one, as you are taking a holistic view of the kind of application that will be required, and defining the overall objectives of what needs to get done.

But most importantly, you are acknowledging the fact that security is going to be a top issue here, and you are laying down the foundations as to how the system of checks and controls will evolve. But also, you are also figuring the roles that the Operations and IT Security team will play in the S-SDLC.

<u>Defining the Requirements</u>

Obviously in this phase, you are formally defining the needs and wants of the client in the project, and mapping out the various software modules that will be needed to meet this objective. But also remember, that this is the key stage in which you will formally address the types of security issues that you think could evolve as development process evolves. It is very important that you take your time in this crucial phase, and this is one of the biggest areas in which you will need to involve the IT Security and Ops teams for their input.

This can also be referred to as the security forecasting stage. There will be issues of course that will come up of which your teams did not anticipate here. The goal here is to map out just about every what-if scenario that you can, so that any items of concern can be addressed quickly and efficiently. To help you in this process, there are various methodologies that are available, and the one that is most widely used is the Open Web Application Security Project, also known as "OWASP". As its name implies, this is an open-source platform in which the public can get access to the latest Cyber threat variants that are out there, and which are also ranked according to their degree of severity. You can gain access to this website here.

The bottom line is that before you can move forward, all of the teams have to come to a common consensus of the potential vulnerabilities and threats that they need to be on the lookout for as the source code is being developed and compiled.

Design & Prototype

It at this phase that you will start to implement the security controls into the various software modules, paying attention to these top three design philosophies:

1) The Principle of Least Privilege:

   This is the minimum rights, privileges, and permissions are established. In other words, end users will gain access to what ever they need in order to perform their daily job tasks, and nothing more than that. It is important that the source code be flexible and dynamic in this regard, as roles and titles do change among employees.

2) The Principle of Separation of Duties:

   With this concept, you are never giving away total, 100% control to just any one employee. Rather, it takes a few individuals to complete one large task, in a sequential fashion, based upon the rights, permissions, and privileges that they have granted. The source code that is being developed needs to have this kind of functionality implemented to it.

3) The Principle of Minimizing the Attack Surface Area:

   This simply means that that the source code that is being designed is clean and robust in nature, and most importantly it is not bloated in nature. For example, software developers like to use APIs in order to keep up to the timelines that have been established in the Planning phase. But there can become an overdependence on using more APIs than are necessary in this regard, which will make the overall application larger than what is necessary. What this translates into is that the Cyberattacker now has a much larger attack surface to penetrate into to spread their malicious payloads. But by having the source as "lean and mean" as possible, the attack surface greatly shrinks in proportion.

Development

As its name implies, this is the part of the S-SDLC in which the actual source code is compiled. The actual development process does not occur in just one huge chunk, rather it is done at the modular level, which was preestablished back in the Planning phase. As technology is rapid advancing at a rapid pace, so are the tools which are used to create the source code. In this regard, automation has become very important, not only to keep the project moving along, but also to reduce the amount of errors that could possibly occur.

Automation can replace many of the mundane and repetitive tasks that are involved, even when it comes to the security perspective. Some examples of this include the following:

- Continuous Integration: This is where the software developers submit each iteration of the source code that they have worked on into a central server, and is combined into one unit. It is not just a one-time deal, it can actually occur several times a day, depending upon the scope and magnitude of the development project. From here, automated builds and testing can then then take place, in an effort to track down any errors and vulnerabilities that exist in a very quick manner.

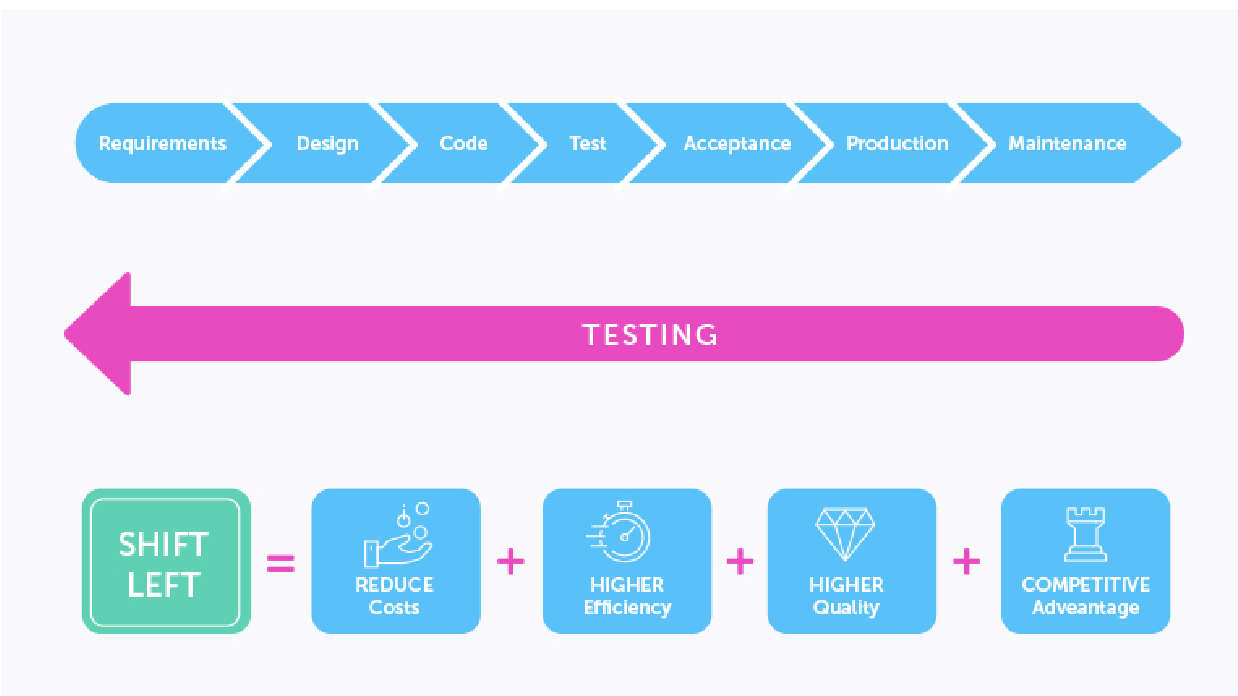
- Automated Security Testing: This is where Penetration Testing comes into play. With this, the primary objective is to find and locate any hard-to-find gaps, and remediate them quickly. There are many tools out there that can do this, such as Kali Linux or GitLab.

- Secure Code Repositories: This is especially useful for the storing of API Libraries, as reviewed earlier. Here, automated testing tools can double check that any APIs to be used in the S-SDLC are free from any bugs and are updated with the latest patches and upgrades.

<u>Testing</u>

In order to ensure the greatest level security in a software development project, each software module should be tested thoroughly tested, both from the standpoint of Penetration Testing and Threat Hunting. However this does not each module should be tested one at a time. This would simply take too much time to accomplish. Rather, the automated tools as described in the last subsection can be used to test these modules simultaneously, or in parallel. In the world of DevSecOps, this is technically known as "shifting left", because you are starting the testing process at the very beginning stages, rather than waiting until the end.

This is illustrated in the diagram below:



(SOURCE: 2).

Deployment

This phase where the hand off of the project to client actually occurs. In an ideal setting, the client should also test their new Web application for any weaknesses or backdoors that could have still been overlooked in the S-SDLC phases. But many times they won't, because they simply all is good and fine. Therefore, a critical aspect of DevSecOps is to conduct one last Penetration Test before the application is released into the production environment. There really is no need to involve any Blue or Purple teams at this stage, simply the Red Team will suffice. Of course, anything out of the ordinary should be fixed on the spot.

How To Implement Compliance Controls Into DevSecOps

As it has been reviewed thus far in this whitepaper, of the primary goals of DevSecOps is automation, from the very beginning to the very end of the S-SDLC. Although the emphasis here has been primarily that on security, automation can include other areas such as QA testing, compiling the source code, etc.

But a common question that often gets asked is if DevSecOps also includes a compliance component. The straight answer is no, it does not.

But given the dynamics of today's Cybersecurity threat landscape and data leakage issues, compliance is a very important point of consideration for DevSecOps. For example, if you create a web-based application, you inherently have landed some of the compliance responsibility because your software development implemented controls in the source code itself to help mitigate the risks of data leakage issues. Likewise, if your team has created applications that are internal to your organization, such as a company wide Intranet or any other employee facing portal that could contain PII datasets, then you are totally responsible for the compliance that goes with it. In these instances, compliance usually means that your company is abiding by the laws of the CCPA, GDPR, HIPPA, PCI-DSS, FedRAMP, etc., and that the appropriate controls have been put into place. But compliance actually goes far beyond that as well. It means also that you are keeping a proactive eye on these controls on a real time basis, and taking the needed steps to remediate any issues that come up.

So, what are the steps that you can take to prove to a federal auditor that your organization is in full compliance, especially when it comes to DevSecOps? Here are some recommended actions:

1) Documentation:

   Everything must be documented in the entire S-SDLC methodology. But most importantly, your documentation must include how the controls are put into place for the following:

   - Access Control;
   - Backup and Restoration;
   - Data Retention (this is includes all of your datasets, such as Data At Rest, Data In Motion, Data In Usage, etc.).

2) Change and/or Version Control:

These two terms can be used interchangeably with another, but the bottom line is that your DevSecOps team is keeping track of all of the requested and approved changes when it comes to the source code that is used to create the web application. This applies to both the Test and Production environments. GitOps is a widely used tool here. More information about this can be seen here.

3) Deploy Encryption:

One of the cardinal rules in DevSecOps is that you cannot just assume or take anything for granted. Meaning, when it comes to security, you always have to take the position that a malicious third party could be covertly monitoring your every move. Thus, you have to make sure that each and every item in the S-SDLC is encrypted, so that if something does fall into the wrong hands, there is not much that can be done to tamper with it, given that it is in a garbled state. In this regard, you want to use the highest levels of Encryption that are possible, such as the AES 256 Bit Encryption.

4) Vulnerability Assessments and Management:

It is important to keep in mind that that the S-SDLC is not just a one-time process. Meaning, once the final deliverable has been handed off to the client, the source code has to be continuously checked for any vulnerabilities that still may occur unknowingly, even after all of the testing has been done. This is again where automation can be a big boon. You can use automated Vulnerability Scanning tools to do this on a 24 X 7 X 365 basis, which is the recommended level of frequency at the present time. You can even take this one step further and even do a more exhaustive Penetration Test from time to time, using Kali Linux (as also reviewed earlier).

5) Security Monitoring:

This simply means that your DevSecOps teams have their eyes on any activity that appears to be abnormal from the established baselines. It will obviously take forever to do this manually, so your best here is to make use of both Artificial Intelligence and Machine Learning to once again to automate this process entirely, especially when it comes to filtering out the false positives. In this situation, these tools will be mining all of your log files, and presenting them into a SIEM console, so that your DevSecOps team can see everything holistically from one view, or dashboard.

6) Making sure that APIs are secure:

As also described earlier in this whitepaper, software development teams often rely upon the usage of APIs, so that the baseline code that is present in them can be repurposed and modified to fit the requirements of the project. If it had not been for APIs, a considerable amount of time would have to be allocated to create that source code from scratch. A popular way to gain access to such APIs is through open-source libraries. But the groups and forums that make these available leave the security checking and upgrades to the software development team. Therefore, you need to keep a central repository that accounts for all of the testing and upgrades that have been made to these APIs before they are implemented into the source code.

An automated tool you can use here is the Azure DevOps Artifacts, and more information about how to implement it and use it can be seen here. Although you can consult OWASP for the latest threats that are posed to APIs, a more comprehensive database for finding such information can be seen at the NIST National Vulnerability Database here.

The Strategic Benefits of DevSecOps

Now that we have examined in some detail as to how your organization can implement security and compliance into DevSecOps, the next question you are probably asking yourself (or for that matter, even your entire DevSecOps team), is what are the benefits to be gained from all of this? Obviously, there has to be a net gain from all of the hard work you have put in to make this into a reality, so here are some of the advantages that you will be able to yield from it:

1) Long term cost savings:

   By implementing security at every step of the way, you will in some fashion or another realize cost savings. Just imagine if you have never tested the web application, and it was released to the customer, and over a short period of time, they have discovered a number of bugs in the functionality of it. All fingers will be pointed back at you, and the cost will come out of your own bottom line to repair them. But by implementing security into your S-SDLC at the beginning stages, this probability from this from actually happening will be greatly minimized.

2) You will be able to deliver on time:

   As it has been discussed in this whitepaper, many software development teams often wait until the very end to security test the source code. Unless this has already been factored into the timeline, this can create a huge delay in the expected time of deliver. For instance, if there are many bugs that are discovered, then it simply means that much more time and extra will have to be devoted to remediating them completely. Also, by waiting until the very end, the chances are greater that a more haphazard job will be done, creating even more problems for your client in the end. But by testing on a modular basis, and in a parallel fashion, you can pretty much guarantee that the project should be delivered on time.

3) Everybody will feel a sense of responsibility:

   One of the biggest goals in Cybersecurity is to have companies take a much more proactive mindset, not only internally, but externally as well. In other words, the thinking that it is entirely the responsibility of the IT Security to make sure that all is protected needs to quickly disappear. The silos that have separated employees in this regard need to come down. Everybody has a responsibility to security, and by having the DevSecOps work as one harmonious unit, this will foster that kind of proactive mindset by showing that it can be accomplished, for the long term. In other words, a sense of "federation" will emerge, which will be a solution where everyone benefits.

4) An increased sense of perceived value:

   When customers look to purchase products and services, they not only look at the features and benefits of it, but in today's world, they are now looking at just how the secure the product actually is. A good example of this are the Internet of Things (IoT) that are designed for home use. Prospects are now looking at just how secure they are when it comes to connecting with other devices, whether it is a tangible product or even a software application. For example, by pointing out the security process and checking that have been put into the creation of the web application will not only lead to an increased perceived value of it, but it will also mean increased sales.

5) <u>Greater chances of being compliant</u>:

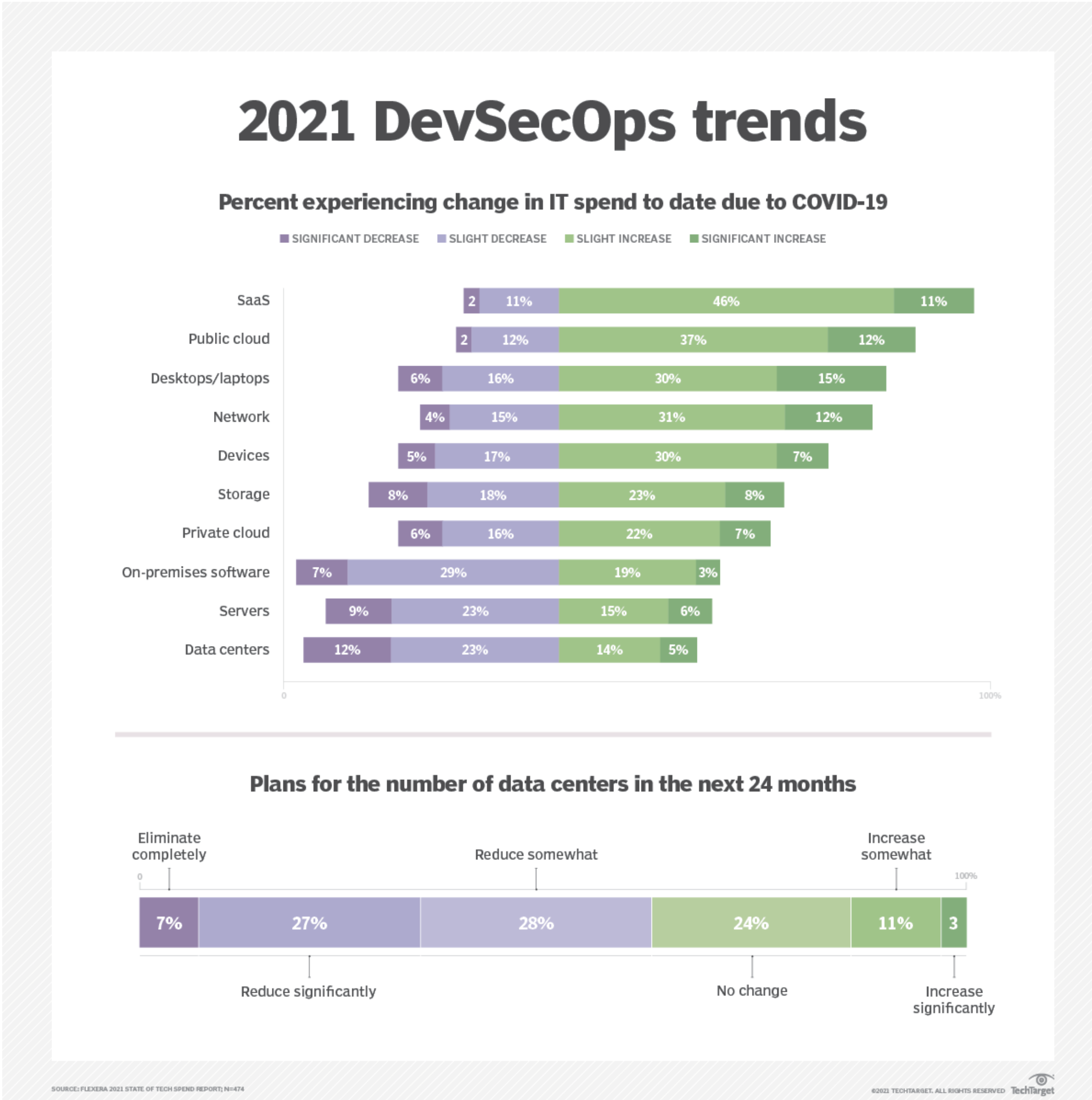
   It is important to keep in mind that the concepts of DevSecOps does not apply to just the world of the S-SDLC. It can also be used in other departments as well, most notably that of Accounting, Human Resources, and Finance. By using the same kind of framework, you should be able implement controls to help safeguard any confidential information and data that you have stored in your database in an efficient and prompt manner. By doing so, you will become compliant with the tenets as set forth the data privacy laws of the GDPR, CCPA, HIPAA, etc.

Conclusions - The Latest Trends In DevSecOps

As DevSecOps further evolves in 2022, there will be three key trends that are expected to impact it, which are as follows:

1) <u>A bigger move to the Cloud</u>:

   Although the benefits of the Cloud were realized before COVID-19, the pandemic has catalyzed the movement to it even more. For instance, the days of maintaining an On Premises Infrastructure will soon be gone, and it is expected that just about everything will be stored and accessed on some sort of Private, Public, or even Hybrid Cloud platform either in the AWS or Microsoft Azure. The DevSecOps model will be even more needed now than ever before, especially when it comes to preventing data leakages, and software development that will now fully evolve on Virtual Machines (VMs) and Virtual Desktops (VDs). In fact, this trend can be seen in the illustration below:



(SOURCE: 3).

In fact, this further adoption to the Cloud will fuel another trend that will strongly correlate to DevSecOps: Infrastructure as Code, also known as "IaC" for short. This is where Cloud platform is not managed by manually or even by automated processes, but rather it is done through specific configuration files which contain the specifications of your infrastructure.

2) <u>Edge Computing will take off</u>:

With Remote Workforce, pretty much all employees are scattered hundreds, if not thousands of miles apart from one another. Trying to gain access to the needed shared resources from a central VM can still take more time and processing power. As a result, you will see that VMs will be placed much closer to the points of where the remote employees reside at, geographically.

The end result is that not only access, but data processing times will be greatly reduced. This is also technically known as "Edge Computing". When this trend starts to occur in greater momentum, it will be one of the primary responsibilities of the DevSecOps teams to make sure that they are deployed and configured properly.

3) A greater dependency on others:

As the world becomes even more digital, the reliance upon using external, third-party vendors will grow even more. But gone are the days of implicit trust, now business entities are shifting towards a mindset of zero trust, with one of the primary drivers of this being the recent Solar Winds attack. Suppliers will now have to be carefully vetted to even greater degrees, and a fair share of this process will fall onto the shoulders of the DevSecOps team. In the end they will have to make sure that the security protocols implemented at third party are at least in par with the levels that you have set forth for your organization.

Conclusions

Finally in the end, the days of where software developers were not held accountable for security checking are now over. It is not just the responsibility of one team or one department, everybody has to come together in order to fully ensure that the lines of defenses are as beefed up as possible in order to mitigate any risks that can occur, whether it is in the external or internal environments.

The best way to start this is by assembling and maintaining a great DevSecOps team, and showing how teamwork and unity can work hand in hand, in unison with one another. This is a framework that is not going to go away, rather it is now going to be a fixture that will be permeated for a very long time to come. So, the sooner that your organization can adopt this, the better off you will be.

Sources

1) "Introduction to DevSecOps: Best Practices For Adoption", Contino.
2) https://www.testim.io/blog/shift-left-testing/
3) https://www.techtarget.com/searchitoperations/tip/3-DevSecOps-trends-to-keep-an-eye-on
4) https://www.ibm.com/cloud/learn/devsecops
5) https://owasp.org/
6) https://about.gitlab.com/topics/gitops/
7) https://www.redhat.com/en/blog/devsecops-compliance-make-your-auditors-job-easier
8) https://docs.microsoft.com/en-us/azure/devops/artifacts/start-using-azure-artifacts?view=azure-devops
9) https://ncp.nist.gov/repository
10) https://www.techtarget.com/searchitoperations/tip/3-DevSecOps-trends-to-keep-an-eye-on