DevSecOps

Putting the ‘Sec’ into DevOps

December 2017

TABLE OF CONTENTS

Executive Summary

Approximately 25% of companies do not perform any security testing on the products they develop. With the ongoing growth in the number of breaches recorded each year, it is important that companies not only adopt security testing as a standard, but ensure they choose the right security testing strategy to support their organisations.

There are many techniques that help introduce security into the DevOps lifecycle, each has its own set of benefits and constraints. Therefore, it is important to evaluate each technique and associated tools to ensure they match the organisation's security testing strategy, languages and risk appetite.

Introduction

Your customers are putting more and more pressure on you to deliver quicker, therefore you have decided to adopt an agile approach to delivering products in a continuous cycle of design, development and testing. In fact, you are going to embrace DevOps for even greater efficiency, wrapping development and operations into a single workflow. Automation plays a major role in this new way of working. However, security validation in this fast-paced world may be wrongly sacrificed for speed

A recent report suggests that 20% of companies do not test for vulnerabilities[i]. In the same report, it is suggested that a shortage of security skills is a problem. Therefore, as teams start to work at a faster pace, security testing will be omitted more frequently by the pressure of delivering products to customers.

The purpose of this paper (which was presented to the DevSecOps gathering on 6 December 2017) is to bring to your attention the techniques and tools that will help you develop secure software in the DevOps lifecycle.

There are tools, some more mature than others, that can help teams integrate security into the delivery pipeline. This paper will help you understand what you should be evaluating to help you stay one step ahead of hackers and your competitors.

The Problem

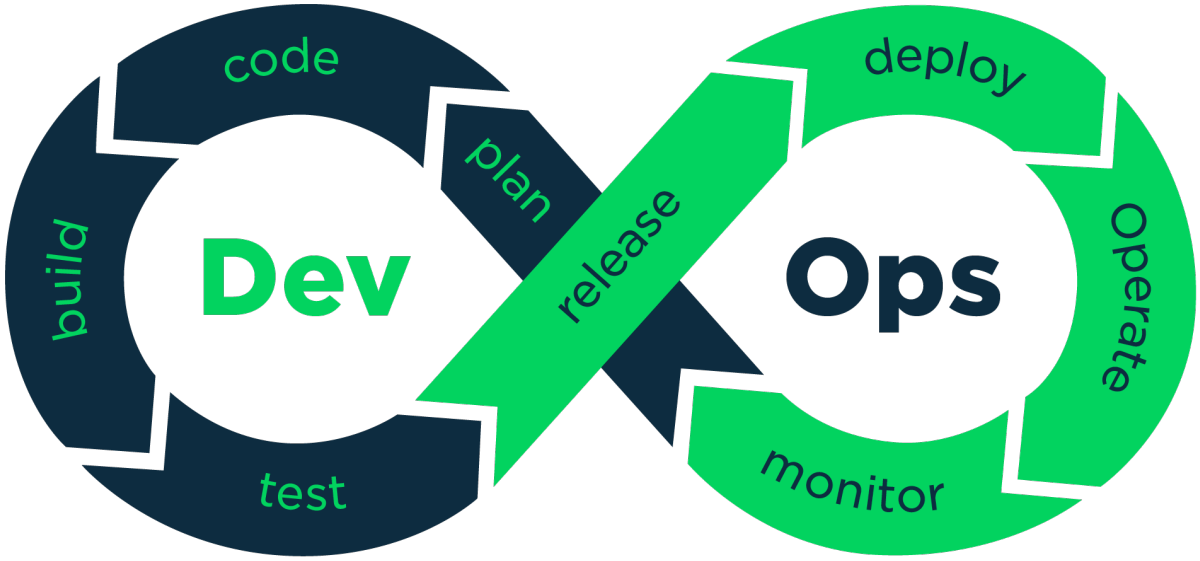
As companies move to new ways of working, they expect to be able to adopt the tools to accelerate their pipeline. In the world of automated security testing there are many tools available covering multiple technologies and multiple techniques. However, the market has yet to reach a level of maturity to support the fast-paced changes that companies are making.

Companies are faced with a number of challenges to integrate security testing into the DevOps pipeline, particularly the lack of support for the languages and technologies used by these teams. Relatively new concepts such as containers as well as the increased use of open source software can leave parts of the delivery cycle untested and often exposing vulnerabilities. In a world where the number of successful attacks is increasing, exposing businesses and customers to financial loss and reputational damage, software security must cover each phase of the SDLC. If companies fail to integrate security into each step of the delivery cycle, the real risk of your company or products being exposed by an attacker will become extremely high.

However, by careful selection of tools and techniques integrated into your DevOps delivery cycle, you can reduce this risk and develop more secure software. In many industries, not only does this improve financial and reputational stability, it will also satisfy industry regulators.

The Solution

Adopting a number of techniques and tools at each stage of the SDLC will allow teams to continue to deliver secure software at a fast pace. Each phase of the DevSecOps cycle has its own challenges, but these can be overcome by understanding how security testing plays a part in each of these phases.



Planning

The planning phase is where the new features or changes to existing features are designed. Identifying potential vulnerabilities at this stage of the delivery pipeline is the most cost-effective way to secure your applications. This is achieved through a technique called Threat Modelling. In an agile cycle, the easiest way to threat model is to introduce STRIDE[ii] analysis into the discussions between developers, architects, product owners and security experts. Whiteboard sessions are the most effective way of analysing the architecture to identify potential vulnerabilities in the design of the feature or application.

Nevertheless, there have been attempts to develop tools that facilitate this process such as Microsoft's Threat Modelling Tool and Continuum's Irius. The main limitations with these tools is their dependency or architectural drawings. In

the fast-paced world of DecOps, drafting architectural drawings prior to development is not best practice. Therefore, the manual process of threat modelling fits better with DevOps way of working.

Coding

During the coding phase, there are many techniques and technologies that help developers keep the number of potential vulnerabilities low. These range from peer reviews, to manual or automated code reviews and designing bespoke non-functional unit tests.

Peer reviews should be a two-way conversation between coders while the code is being designed and written. Through these discussions about the code and observations made by peers, potential issues can be talked through and remediated before the code is committed to your source code repository. This also leads to the sharing of knowledge between developers which improves the security culture and awareness within the development team. However, these types of reviews are dependent on the security skills and experience of the collective coders.

Independent code reviews can be manual or they can be automated. Manual code reviews are similar to peer reviews, but are normally carried out by security experts after the code is written and often after the code is saved to a source code repository. Feedback can be a slow process. Automated testing is the most efficient methodology; the technique favoured for this is Static Application Security Testing (SAST). SAST tools inspect source code directly to identify potential security vulnerabilities within the code base. Although reporting of potential vulnerabilities is relatively quick, the flipside is the large number of false positives that are presented following a scan. This is due mainly due to the assumptions that the tools make about how the code functions at runtime. Good SAST tools will not only identify vulnerabilities in the code, but will also provide guidance on how to remediate the issues and educate developers on how to develop secure code.

Developers who have a good understanding of security vulnerabilities could integrate non-functional test cases into automated unit test suites. However, in

order for this to be successful, developers must write a set of tests that cover a large number of possible outcomes.

Integrating all three practices into the coding phase of the DevOps lifecycle offers a very high level of defence against coding vulnerabilities with individual components and libraries.

A number of vendors offer SAST tools, each with its own unique algorithms, IDE integration and reporting methodologies. It is important that you evaluate these tools to verify that they support the languages and tools your development teams use.

Building

Once the coding phase is completed, the process moves onto the build phase, which is an opportunity to assess the security of all the connected components or libraries in the application or services. A technique that may be considered at this phase is Dynamic Application Security Testing (DAST). This involves running specific scanning tools against the components of an application in a running state and verifying the results using application security experts. The most significant advantage of DAST over SAST is the reduction of false positives. However, the results are not instant, which suggests this may not fit into a fast-paced DevOps environment. Nevertheless, it has been shown to be a highly effective way of reducing the number of potential security bugs being deployed to production.

Testing

Testing an application in a DevOps environment usually involves running functional tests as part of an automated quality assurance process. In the world of security testing, an emerging technique is Interactive Application Security Testing (IAST). There are two types of IAST: active (or induced) testing and passive (self-induced) testing. The former relies on DAST to activate the IAST process; the latter monitors applications passively while being tested. As mentioned above, the use of DAST in a continuous delivery cycle is not effective due to the slow turnaround in delivering results. Passive IAST works with the existing automation process leveraging this capability to detect security issues as they are found. This means passive IAST is a very effective tool to use in the DevOps cycle.

This is an emerging security testing technology but there are tools that are being made available by vendors. Most notably, Contrast Security which has significant movement in the Gartner Magic Quadrant[iii].

Releasing

As the DevOps cycle moves from the development phases into the operations phases, the types of security analysis move away from application testing and into the hosting, integration and environmental testing.

During the release phase, applications may be integrated into web-hosting environments such as Apache and could, potentially, integrate with other third party or open source components. These are not under direct control of your development teams which means that security testing should focus on the known security status of the third-party component. It is important that open source components are patched against known vulnerabilities and if only unpatched versions are available, these need to be identified and reported back to the development teams. Testing against the Common Vulnerabilities and Exposures (CVE) database requires a different set of tools that identify the open source software used in your products and comparing them against the version of the open source software within the CVE database to determine whether they are vulnerable to known exploits.

There are tools that offer automated integration into the DevOps process to facilitate this process. The growth in use of Free and Open Source Software (FOSS) is driven by the need for service and application providers to delivery business functionality faster. Therefore, developers use FOSS to make use of their out-of-the-box features, in order to focus on adding business value to their applications and services. In addition to this growth, the number of vulnerabilities being exploited within FOSS is also growing. Therefore, open source scanning has become an important element in delivering secure products and should be adopted in your DevOps lifecycle.

Deploying

Deploying code into production is a risky part of the pipeline if not handled correctly. Assurances need to be made that the code that is pushed to production is the same code that has been security tested during the earlier phases.

Malicious tampering or unintentional mistakes in the deployment process can introduce vulnerabilities. Therefore, it is important that the complete deployment package is tested and the process is automated to avoid these risks.

Containers are fast becoming a standard way of deploying application and infrastructure code into production in a secure and standard way. This is an emerging technology, although there are several vendors now offering several Container technologies.

Operating

Once code has made it into production, validating the security of the applications and services has not ended. It is feasible that vulnerabilities exist in production, either ones that were difficult to detect prior to deployment into the live environment or ones that have emerged from new threats. There are three key methods to safely detect vulnerabilities that have made it to production.

The first and most common method is the use of internal or external penetration testing teams. Penetration tests should be periodically commissioned to validate your applications running in a production state (which is, not necessarily in the production environment). This type of testing is often targeted at new functionality or specific parts of the application or service.

Another, more expensive technique, is to offer bug bounties to ethical hackers who will attempt to attack your applications or services for a price and provide you with evidence of how to repeat the attack so that development teams can work on a fix. The advantage of this technique is that the bug bounty hunters are looking for any way into your system. They give you an opportunity to fix the issue and close the threat before a more sinister hacker exploits the vulnerability.

Finally, there are automated tools that can continuously validate your applications in production to identify new vulnerabilities that have been discovered since your application went live.

Monitoring

Another effective way to identify attacks that are in progress is to adopt System Information Event Monitoring (SIEM) techniques into your armoury. Tools and products that support this activity should provide the following features:

- Real time alerting, to notify your team that an attack is potentially under way.
- Data aggregation which pulls data from application and system logs to facilitate searching them for unusual behaviour or patterns.
- Dashboards that show your team in real time the current state of the application or system, making it easy to identify failed services, network traffic threshold breaches, failed login attempts etc.

These features can identify potential attacks such as Distributed Denial of Service (DDoS) attacks or brute force authentication attacks allowing your teams to close them down immediately.

Left to Right security testing

The key to delivering secure software to your customers is to ensure applications and services are fully tested during each phase of the DevOps cycle. However, the cost of this approach can be inhibitive or even greater than the loss associated with a breach. Therefore, it is important to understand the most cost-effective approach for your organisation. In simple terms, it is most cost effective to eliminate security vulnerabilities as early as possible in the delivery of a new feature or product. Spending time and resources at the design phase to undergo a threat modelling exercise is cheaper and likely to capture a large number of issues at a relatively low cost. The further you travel through the cycle, the more expensive and more difficult it becomes to identify and address potential security vulnerabilities.

However, techniques such as SAST, FOSS Scanning and Real-time monitoring are effective automated processes that will help you identify security issues and reduce the risk of attacks on your applications and services.

The Conclusion

Security testing is often overlooked or approached with little urgency in many organisations. Half-hearted attempts to introduce penetration testing as a strategy to validate products is misguided and rarely effective at making your products secure. Introducing security testing into each phase of the DevOps cycle will reduce the risk of major security breaches that incur huge financial losses or irreparably damage your reputation with your customers, employees and shareholders.

However, it is important that your company's appetite and cost for security does not outweigh the financial or reputational loss caused by a security breach. Therefore, you must select the right techniques and the right tools to keep your applications and services secure. Understanding your options is key to making the right choices.

Many tools are available on the market to assist you in securing your applications and services, each of which you should weigh up carefully. Some low hanging branches, such as threat modelling and simple monitoring of your applications should be considered by default. Application security testing tools are also relatively easy to integrate and, if implemented correctly, can be cost effective. SAST, DAST and IAST each bring their own set of advantages and disadvantages so it is important to understand which technique will benefit your organisation the most.

Your choice of tools is also dependent on the technologies and languages used in your organisation. Likewise, the scope of open source software used in your products determines the type of scanning required to identify vulnerabilities they harbour.

If you are serious about securing your products, evaluating the tools and techniques available to you is a valid first step. From low-cost to high-value, there are many products that will help you reduce the risk of exposing your products to attacks. The cost of security must be proportional to the value of your assets that you are protecting. However, these tools must also allow your teams to continue

delivering at the speed your stakeholders expect from your company and therefore, must integrate into your DevOps cycle.

Finally, like the products you develop, the automated security tools available to you are evolving and improving continuously. Therefore, it is important that you adopt a strategy to evaluate your security tooling regularly and adjust your strategy accordingly.

Putting the ‘Sec’ into DevSecOps is achievable and a necessity in this high risk, high paced world in which we find ourselves.

About Your Company

Dynaminet Ltd is a small independent company specialising in Digital transformation, Security, DevOps and Information Governance.

Email: info@dynaminet.co.uk

---

[i] https://www.trustwave.com/Company/Newsroom/News/New-Report-Shows-that-One-in-Five-Businesses-Don-t-Test-for-Security-Vulnerabilities/

[ii] STRIDE is an anagram for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privileges. See: Shostack, Adam – *Threat Modeling Designing for Security* (Wiley 2014)

[iii] https://www.prnewswire.com/news-releases/contrast-security-named-the-only-visionary-in-gartner-2017-magic-quadrant-for-application-security-testing-300417201.html