DevSecOps – Agility with Security

Date: October 2016

Doc Ref: SOS-WP-DSO-161001

Authors: Murray Goldschmidt & Michael McKinnon

Table of Contents

Overview

With the rise of the DevOps movement, a chasm has emerged as it becomes evident that superfast and continuous software development and deployment is marginalising "traditional" security expertise, knowledge, and best practice.

This document aims to help security managers, software developers and business stakeholders understand what Dev"Sec"Ops represents and how it can help improve the security posture of your enterprise.

The information contained herein has been compiled from the collective experience of the security consultants at Sense of Security. Through previous engagements and security assessments, we have observed what does and doesn't work.

This paper is not intended to be a comprehensive "how-to" guide on how implementing the perfect DevSecOps environment, nor does it cover the tools and methodology that Sense of Security has devised when conducting such engagements.

However, we do hope this whitepaper provides you with valuable insight into the new and emerging theory of DevSecOps, as well as exposure to key objectives and discussion points that will continue to evolve over time.

Our contact details can be found online, and we look forward to receiving your feedback.

What is DevSecOps?

The term "DevSecOps" injects the domain of Information Security into the original concept of "DevOps", which itself is a compound term referring to the conjoining of Software Developers and IT Operations staff and systems, through collaboration and communication.

However, to properly understand the importance and advantages of the emerging DevSecOps concept, one needs to first understand the history and growth trajectory of DevOps, alongside the issues and challenges of integrating best practice security.

Since 2008 and growing steadily since, the DevOps movement has embodied a fast-paced engineering cultural practice in software development, more commonly used for cloud applications and deployments. Developers have access to efficient tools and agile production methods and this has created the foundation by which many well-known web applications now operate successfully.

DevOps is a common practice used to produce many of the world's largest web applications, such as those operated by Netflix, Etsy, Facebook and Amazon to name a few – as well as organisations ranging from start-ups through to large enterprises.

Arising from the need to deploy software quickly and with a high-degree of reliability, DevOps has been redefining software development such that it now includes many of the benefits of agile production methods, combined with the cost-saving automation of IT operations.

There are three main aspects of DevOps:

- People & Culture
- Practices & Process
- Tools & Automation

We cover these aspects in further sections of this document and explain how and why information security (the "Sec") should be incorporated into DevOps to achieve better outcomes for enterprises wishing to improve their security posture.

In summary, DevSecOps aims to be a movement that extends all of the organisational benefits achieved through a DevOps culture, with the addition of the benefits of Information Security – all without compromising on quality, speed, or efficiency.

Why is DevSecOps Important?

While many enterprises have invested in a DevOps culture and are benefiting from productivity improvements and better reliability of their applications, not a month goes by without seeing reports of yet another major data breach.

Despite DevOps teams utilising numerous tools and methods to build and deploy their infrastructure and their own custom applications, multiple issues remain in enterprises where security in the development life cycle is either ignored, overlooked, or not seen as a primary concern.

As a result, many enterprises and even some of the major DevOps software vendors are beginning to realise that incorporating security practices is a must-have, not a nice-to-have[1]. In our consulting practice, we have seen a wide range of issues affecting DevOps environments when it comes to security.

Security not a primary concern – for some enterprises, the risk of not improving their development practices from a security standpoint hasn't been fully considered, and at worse hasn't been considered at all. However, adopting a DevSecOps environment and identifying issues in the build cycle can lower attack risk, and improve the robustness of your entire enterprise.

Lack of secure coding awareness or best practice – insecure coding practices, even accidentally, can lead to injection flaws, inadvertent data exposure, cross-site scripting vulnerabilities, and authentication features that can be bypassed easily. We see this far too often, and we discuss in this document a number of ways it can be improved.

Too much focus on availability – rapid deployment of code can lead to DevOps practices that focus heavily on the reliability and availability of the application in production. This single-minded focus can overshadow security concerns. Availability is considered a tenet of good security, but so is Confidentiality and Integrity.

Supply chain issues with software libraries – approximately 90% of modern applications consist of third-party and open source software[2]. Easily obtained software components and libraries from untrusted sources provides exposure to vulnerabilities. Even when developers are diligent about using secure libraries, they may not be aware that those libraries use other libraries of their own, resulting in latent exposure.

Misconfiguration of systems – in the race to provision infrastructure and platforms quickly and automatically, it is critical that the configuration settings applied comply with security best practice. The skills gap when it comes to understanding specific hardening techniques for these systems is often lacking, and in many enterprises these are the low-hanging-fruit issues we see far too often.

Lack of audit trails and logging – developing an application with a view to fast deployment can result in a minimalist approach to auditing and logging features that might otherwise provide critical information during a security incident. Security teams that

[1] http://searchitoperations.techtarget.com/tip/Why-your-DevOps-process-should-have-security-baked-in
[2] "Continuous Acceleration: Why Continuous Everything Requires A Supply Chain Approach", Joshua Corman - https://www.its.fh-muenster.de/owasp-appseceu/2015/

are well-versed in incident response are sometimes cut-out of the DevOps design phases, where they need to be involved.

In summary, DevOps teams are making great strides in the areas of automating and streamlining the software lifecycle, but in general aren't focussing on security – this is why implementing a DevSecOps practice is important.

People & Culture

Enhancing an existing DevOps culture and introducing security to that dynamic for many enterprises can require a step change in communication with the existing (and sometimes isolated) security team.

Wider perceptions that sometimes affect internal security teams will need to change, as better integration with development practices is achieved, resulting in the dismantling of siloes. Security teams can no longer afford to operate in isolation.

When it comes to having an impact on the people and culture in a DevOps driven enterprise, it's the collaborative and communicative aspects that make it a profitable undertaking for many enterprises.

As DevOps is all about automation and velocity of service delivery, it is important that the security features adapt to this dynamic process. Information Security fundamentals remain, however their application to the DevOps lifecycle need to be fine-tuned so that security delivery is seen as transparent rather than a barrier to progress.

Security teams that rely on old approaches, that don't innovate, and don't contribute to the overall success of the organisation's endeavours will quickly become marginalised.

It's not all doom and gloom for security teams, though, as the domain of information security and the specialist knowledge required to achieve results continues to be highly sought after. This high value asset sitting inside enterprises needs to be fully utilised as a result.

Practices & Process

Apart from the people and tools used in a DevOps environment, it is largely the practices and processes that make it successful. Moreover, it is the systems thinking and procedural alignment with agile production methods that contributes significantly.

The procedural opportunities for embedding security within a DevOps environment become apparent when we study what "DevOps Done Well" looks like. (see Figure 1)

As we have alluded in People & Culture there are often barriers that exist between security teams and developers, but this also extends further to operations teams and management. Removing those barriers is naturally the founding stone of good DevOps.

Another key driver is the need for modern competitive enterprises to reduce their time to market, but we also see opportunities for tempering this with the need to ensure products and services are also secure.

Figure 1 - "DevOps Done Well" - Murray Goldschmidt

The need to identify issues and resolve them quickly applies not only to the pre-production pipeline, but already applies to the work of many security teams who are often the quickest at identifying issues that cascade back to many other teams.

The concept of quality at the source is a key concept, and is relevant to establishing a solid DevSecOps practice that protects the source repository.

Once a system component or portion of code has been adequately assessed and confirmed to meet all the security requirements, it should be endorsed as “Quality Approved” and can be re-used with confidence about its provenance. This can be further extended to “immutable systems” where a specific version of the system component cannot be changed, providing referential integrity to deployments derived from it.

Feedback for any application or system can originate from a number of places, both internally and externally. In a DevOps culture, feedback has become an important aspect of the development process as stakeholders at all levels are able to contribute their ideas and suggestions for improvement.

The Feedback loop is also enhanced through automation and tooling, where security defects are fed back to the preceding process for resolution. Once established, security testing through automation provides very effective visibility into the security status of the system at any point in time. Only defects will invoke a remediation process with all other processes flowing naturally through the service delivery pipeline at speed.

Lastly, removing silos and handovers are procedural improvements that good DevOps practice should bring about. Handovers cause a drag on velocity and without cross functional teams involved, handovers that separate teams will also introduce time delay, functional drift, and loss of accountability.

Tools, Automation and the DevOps Stack

A typical DevOps technology stack consists of a mix of tools from various vendors both open source and commercial. These vendors range from infrastructure and cloud platform providers, to makers of specialised development tools, services, and libraries covering:

- Infrastructure/Platform
- Code Repository
- Configuration Management
- Build and Orchestration / Provisioning
- Test Management
- Monitoring
- Deployment Environment

When it comes to major efficiencies, automation and orchestration has perhaps played the biggest part in DevOps - helping underpin the advancements in software development practices through the ability to repeat tasks over and over again.

Rarely is a technology product in a DevOps stack standalone, as each tool and platform invariably contains provision for automation or integration of some kind. This not only provides for speed in the development process by eliminating manual data handling, but also reliability and stability.

On the migration path to DevSecOps it cannot be underestimated just how important automation and orchestration features are in the stack when aiming to achieve a good overall security posture. Additionally, the security implications of using the tools themselves must also be understood properly.

In summary, automation is vital in the stack, and while some vendors are beginning to adapt their tools to cater for better security through DevSecOps, in our experience there is still room for improvement.

Software Development vs. Service Delivery

Software engineering practices have evolved in response to a changing technology landscape. As a result, the traditional view of the Software Development Life Cycle (SDLC) is arguably becoming outdated, particularly with the proliferation of web-based applications that are delivered as “services”. In fact, one could argue that SDLC now more appropriately stands for ***Service Delivery Life Cycle***.

We see this in some of the world’s largest public applications at scale, such as those operated by Netflix, Etsy, Facebook and Amazon that demonstrate on a frequent basis how web-based applications need to constantly evolve.

The notion of a service being delivered continuously means that modern software projects no longer require a fixed completion date. As researchers at Facebook openly acknowledge, “Sites like Facebook will never be completed. The mindset is that the system will continue to be developed indefinitely.”[3]

This never-ending-development mindset brings about a major shift in thinking when applying traditional security techniques and controls. Web applications in particular aren’t evolving only in response to demand for new user features and performance, but also for security – and this means having continuous security testing in place to match.

[3] https://research.facebook.com/publications/development-and-deployment-at-facebook/

And when it comes to developers, providing a quality application at scale requires that they're able to support the continuous operational use of their software. Developers can no longer avoid the responsibility of how their code performs in a production environment.

Therefore, providing a continuous Service Delivery through a robust DevSecOps environment must be achieved through providing Information Security practices that are delivered continuously and constantly, in step with the modern delivery lifecycle.

Securing the Continuous Development Chain

In DevSecOps security can no longer be implemented as a one-off or standalone undertaking, and therefore we must look for ways to automate and streamline many of the repetitive tasks that can be used to easily identify low-hanging-fruit issues.

In a development chain that functions continuously, the challenge is re-thinking and innovating the way security controls can be implemented. We identify here a number of key objectives in this endeavour.

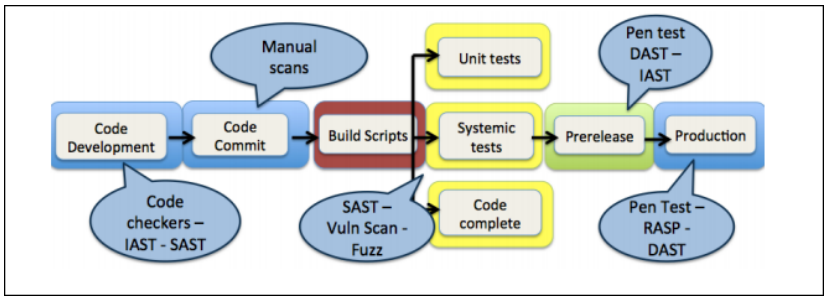

Figure 2 - Building a Tool Chain[4]

Security must keep up with speed of delivery

DevSecOps can only be successful if it contributes meaningfully to the existing DevOps movement without negating any of the benefits of speed or reliable service delivery.

Surround dynamic processes with protection

Each and every stage of the production lifecycle is a potential opportunity to provide protection. For example, dynamic or static code analysis can be automatically triggered immediately when code is committed to the source repository.

Build-in security testing automation

Security testing that is done right once, with tests set in code (just like your application code) is one of the best ways to begin realising the benefit of DevSecOps. Discard complex long range security roadmaps in favour of a "baked-in" security approach that addresses security incrementally in achievable and actionable chunks inside the core of your development pipeline processes.

Incremental but continuous improvement to security

Embedding security into the development lifecycle should be incremental – your security posture must always be improving and never weakening. We discuss this later as "immutable systems" and "phoenix upgrades".

[4] https://securosis.com/assets/library/reports/Security_Into_DevOps_Final.pdf

Quality at source, with frequent feedback

Maintaining quality at the source is based on the premise that if you close the gaps and variation within the development lifecycle you will eliminate problems from occurring, before they occur – this is practically achieved through triggered tests that are fully automated. Recycling known good configurations and source code modules also eliminates wasted development and deployment cycles and provides confidence about the integrity of the building blocks of your platform and the application delivered on it.

Securing the Layers of DevSecOps

The practical application of information security often involves the concept of layers, and appears in many successful security strategies such as defence in depth. When creating a DevSecOps environment it is just as important to understand these layers to maximise the benefits.

Accordingly, in a modern computing environment and with particular reference to DevSecOps, in our view there are usually three distinct layers to consider when it comes to a cloud based deployment.

Infrastructure

At the foundation of any application is the infrastructure layer, the security requirements of which should not be underestimated.

As DevOps teams take up more responsibility for automating the functions of provisioning and configuring infrastructure components, often the focus is on availability – at the potential sacrifice of good security.

For example, Google call it "Site Reliability Engineering"[5] – citing they think availability is one of the most important aspects, on the basis that if nobody can access the site, what good is it.

While we agree that availability is important, it should not come at the cost of unnecessarily exposing data, and no doubt Google have just as equally invested heavily in that area as well.

Automation and orchestration of Infrastructure such as the provisioning, configuration, and deployment of Infrastructure components is a huge advantage. But it can't be executed blindly, and this is where the addition of security expertise can significantly reduce the obvious risks.

Let's look at a few interesting and innovative security measures that can be implemented into existing DevOps practices that can have immediate impact on security posture.

Infrastructure as Code – programmatic execution that causes the provisioning of a platform, and even the infrastructure (e.g. provisioning of virtual machine images) is commonplace. From a security context though, system images must be secure at the source, pre-hardened and securely configured.

Use Immutable Systems – an immutable system by definition cannot be changed, and if deployed and implemented correctly can reduce or even eliminate external tampering of production infrastructure. Through the systematic separation of code and data, and making systems read-only, high levels of security can be achieved.

Adopt Phoenix Upgrades – just like the mythical "Phoenix" bird that would burn and rise again from the ashes, this concept involves a one-way path when it comes to

[5] https://www.wired.com/2016/04/google-ensures-services-almost-never-go/

upgrading production systems. Instead of applying patches and upgrades to running infrastructure, you only upgrade pristine system images, updating the approved version in the source repository as the new known-good, and then you "burn" or discard your production systems and replace them with the new images through a phoenix upgrade.

Implement Robust Auditing – often overlooked by DevOps teams who generally aren't required to investigate security incidents affecting the enterprise, it is important to implement robust auditing. For the infrastructure layer this should include an appropriate level of auditability for all administrator actions and all changes made to the fabric and the services running on it. This data then needs to be centrally logged. For example, in a cloud environment you would want to know who made changes to perimeter controls such as elastic load balancers, inbound and outbound access controls, all instances of remote access into the environment, changes to any storage buckets, and all privileged changes made to user accounts, and the policies enforced on them.

Embrace end-to-end continuous deployment – for existing DevOps teams that are not yet using automation or orchestration, from a security perspective they are missing out on many advantages. Automating security processes, some of which can be very time-consuming, such as hardening systems are a must these days. This item alone has been the catalyst for some to consider adopting DevSecOps practises.

Configuration and vulnerability management – along with the adoption of immutable systems, and phoenix upgrades, it is important to manage potential infrastructure vulnerabilities, continually assess for the introduction of new vulnerabilities in the environment, and ensure configuration settings are always known and controlled. Many security teams have extensive experience with this already; knowledge that can be instantly leveraged in a DevSecOps environment.

Platform

The definition of a platform in DevOps can include components ranging from the operating system (which overlaps with the infrastructure) to the web server or runtime environments (that overlap with the application).

Implementing an effective DevSecOps environment in the platform layer usually relies on taking advantage of automation in a highly dynamic environment. The ability to provision and destroy components automatically, quickly, and frequently are all critical factors for success.

Securing the platform at this layer is generally focused on configuration management which becomes even more relevant if the infrastructure has been implemented using the "immutable system" design discussed earlier.

Configuration settings and the inclusion of specific data files, such as security certificates and other such supporting files should be all that is needed at this layer.

For example, take the numerous tasks of configuring an Apache web server with a secure SSL/TLS configuration and deploying private keys – performed manually this is repetitive and prone to error and variation in process.

The alternative is automating and controlling the web server configuration file, along with scripts, private keys, and other files stored securely for integrity – but not only that – also triggering security tests that can be automated to ensure the SSL/TLS configuration complies to the highest standards at all times. A failed test here might indicate a need to address the configuration, or even upgrade the infrastructure.

Common functions performed at the platform layer can also include many of the configuration components and settings you would expect to see with the likes of Amazon AWS, Microsoft Azure, and other platform providers.

In summary, in a properly implemented DevSecOps environment at the platform layer you don't make changes to the platform itself, you make changes to the settings – where automation and control are vital.

Application

The application is something a DevOps team will already be very familiar with, having presumably authored much of the code behind it.

Providing security enhancements at this layer, which also includes the development process itself such as deployment, is critical to implement to ensure a consistent and reliable result.

We have already identified multiple areas of focus for enhancing application security in this whitepaper, and we present here another five to consider.

Continuous Integration – this is the practice of automated deployment combined with running a battery of pass/fail tests that will determine if a particular build will make it to the production environment. The production environment itself may be subject to a continuous integration process as well. In terms of integration of security, continuous integration can be leveraged effectively as the trigger point to provide many different security scans and tests in a DevSecOps environment.

Microservices – communication between multiple systems, with the use of "restful" application programmer interfaces (API's) has become mainstream allowing levels of integration never before seen – not only between application components, but between all systems both inside and outside the DevOps environment. However, the implications of interacting across system and application boundaries, sometimes with third-party services is associated with enhanced security risk around privacy and authorisation. Accordingly, all services need to be validated for security capability to defend against attacks.

Configuration Management – all applications require configuration settings of some kind or another, and often include highly confidential data such as access credentials (e.g. Amazon AWS Keys), sometimes incorrectly stored in the source code repository. It is imperative that developers are aware of the risks of poor configuration management habits, particularly around the storage of sensitive items.

Monitoring and Logging – security teams are better practised at understanding and interpreting log files and monitoring output in a production system. Developers should ensure they are collaborating with the appropriate security experts available to the enterprise at all times. Naturally, all applications need to log relevant actions including user identification, type of event, date and time, success or failure, and origination of event. All privileged actions must be auditable, including any changes made by an administrator.

Communication and Collaboration – a key part of the practice of DevOps is the efficient communication cross-team and up and down the organisational chart. Security teams that are being marginalised should be far better served in a DevSecOps environment, and with great benefit to the enterprise as a whole.

Summary

Underpinning the ideal concept of DevSecOps should be a solid foundation of risk management and information security expertise and knowledge – implemented in such a way that only the advantages are carried forward, such as the mission of maintaining velocity in the existing DevOps environment.

The principles for effective cybersecurity risk management continue to apply to every enterprise, and therefore the endeavour of creating a DevSecOps culture should be obvious – especially for those with a DevOps movement either emerging or established.

Accept there will always be levels of uncertainty – your enterprise will never have a perfect system, and there will always be some element of risk. Risk levels need to be set at a reasonable level which is appropriate for the enterprise.

The overarching objectives of an effective DevSecOps practice should be choosing a reasonable level of risk, and making security a part of every decision, ensuring that your user experience is fantastic, and that your business metrics are thriving as a direct result.

About Sense of Security

Sense of Security Pty Limited is an Australian based information security and risk management consulting practice delivering industry leading services and research to organisations throughout Australia and abroad.

Our strategic approach to security provides our clients with a capability to understand the security risks relevant to their organisation and knowledge to protect their information assets.

We provide expertise in governance & compliance, strategy & architecture through to risk assessment, assurance, incident response & security testing.

For more information, please contact us:

Web: www.senseofsecurity.com.au

Email: info@senseofsecurity.com.au

Phone: 1300 922 923