consunet®

Whitepaper
Anatomy of a Trusted DevSecOps Factory

Executive Summary

Ensuring Trust at Every Stage of the Software Development Lifecycle — A Holistic Approach to Secure Development Practices

The digital landscape is continuously evolving, and organisations are grappling with how to protect their software from security threats. Data breaches, privacy concerns, exfiltration of commercial intellectual property (IP), and loss or corruption of key data and systems, all contribute to damaging customer trust. Security threats directly impact current and future revenue or create increased disruption and delays with significant cost overruns.

Traditional software development lifecycles (SDLCs) separate security practices from the development process. This separation introduces vulnerabilities into the software product, delays releases, and increases development costs. Organisations require a new approach to assuring customer trust and delivering secure software.

A solution to these challenges has emerged, a new paradigm—DevSecOps.

DevSecOps is a modern software development approach that embeds security practices within existing software development and operations processes. This whitepaper defines the elements of DevSecOps and how each element addresses trust, security, and support concerns, while simultaneously transforming ideas into live production solutions for users and customers.

A successful transition to the DevSecOps approach requires an understanding of the core principles of software development, knowledge to discern DevSecOps practices from traditional security applications, and comprehension of how DevSecOps can effectively implement strategies to combat cyber threats. This whitepaper provides guidance on the elements of building a complete DevSecOps Factory to incorporate security throughout the SDLC and enhance trust levels of the end customer product.

Contents

1 Introduction

As software development rates accelerate, the need for secure software development practices increases. Traditional Waterfall methods have been gradually replaced by Agile continuous improvement methods, and more recently augmented with software development and systems operations (DevOps) practices. These approaches, however, are still not meeting the current need to scale and deliver secure software fast.

1.1 DevOps to DevSecOps

The intent of DevOps is to shorten the software development lifecycle, improve product quality, and accelerate software time to market. DevOps considers the delivery of user features as well as the deployment, operation, and support activities required for the product, thus reducing the gap between development and operations for seamless and continuous software delivery to customers.

As DevOps gained popularity, the industry adopted Agile methods to deliver software fast, however security was still considered a barrier to development rather than an enabler or core feature. As a result, many development environments and resulting products had catastrophic risks and impacts such as privacy data breaches, loss of commercial source code, and corruption and loss of data. It became clear that security could no longer be considered a separate phase in the SDLC. This understanding led to the emergence of DevSecOps, an approach focused on integrating security practices seamlessly into the DevOps lifecycle activities.

Shifting Left

Embedding security activities throughout the SDLC promotes shorter, more controlled development, while maintaining effective and auditable security. This approach is referred to as 'Shifting Left' where security activities are considered ‘to the left’ of the SDLC as opposed to only testing for security flaws at the final stage. This approach mitigates the potential for security flaws to continue through subsequent development stages, enhancing the overall security posture of the developed application, providing a proactive monitoring approach to identify and correct security concerns early.

Integrating security throughout the development process improves efficiency, eliminating the need for large-scale, time-consuming security checks at the end of the SDLC. Legacy security testing conducted at the end of the SDLC requires teams to revisit work, whereas 'Shifting Left' flags vulnerabilities earlier for remediation during design and development, and reduces the need for costly architecture and development rework.

Shared Responsibility

DevSecOps promotes a culture of shared security responsibility. All team members are accountable for maintaining and enhancing security in the product and operation environments. This shared responsibility facilitates improved collaboration, innovative problem-solving, and ultimately, more secure software.

Consunet's Trusted DevSecOps Approach

Consunet's DevSecOps approach builds upon existing DevSecOps processes to include activities, practices, technologies, and governance that contribute to increase, verify, and sustain trust levels. This forms Consunet's Trusted DevSecOps Approach—a holistic end-to-end development and security solution achieved with rapid agility. Consunet's Trusted DevSecOps Approach demonstrates value measured by the following metrics:

- **Mean-time to Production** – the time between idea generation to capability deployment.
- **Deployment Speed** – the speed to deploy new secure versions.
- **Deployment Frequency** – the frequency of version deployments for rapid customer capability enhancements.
- **Production Failure Rate** – the likelihood of products failing in deployment.
- **Mean-time to Recover** – the time for a product to recover from a failure.
- **Software Vulnerability Patching Speed** – the time to patch internal software or source library vulnerabilities and fix running software.

This whitepaper describes the elements required to achieve a Trusted DevSecOps Approach including the technologies, processes, and governance practices recommended to be applied in all product development and delivery stages, providing customers with the value they seek—without compromising security or privacy.

2 Understanding DevSecOps

DevSecOps is the concept of incorporating Security activities at every stage in the DevOps process. From planning, coding, building, and testing, to releasing, deploying, operating, and monitoring, Security is considered continuously. These stages are represented in the DevSecOps Infinity Loop, Figure 1. Applying security throughout the development process eliminates surprises, failed deployments, and most significantly—security breaches from threat attack vectors. DevSecOps reduces the likelihood of security being applied as an afterthought and provides opportunities to identify security issues upfront, reducing development costs and the potential for project overruns.

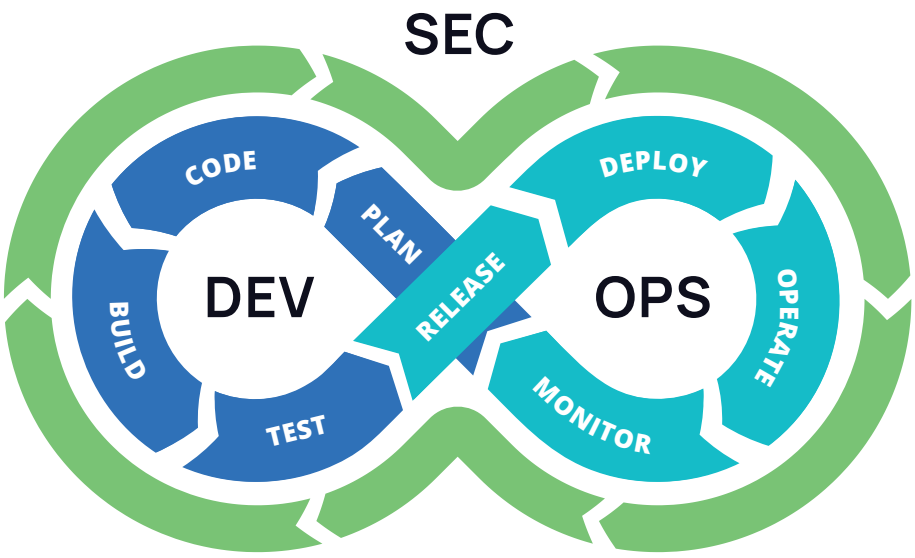


Figure 1 - DevSecOps Infinity Loop

DevSecOps is achieved by ensuring security activities are the responsibility of both development and operations team members (both manual and automated) and includes security checks, code scanning, event monitoring, security controls and patterns/approaches to provide increased security. By embedding security into all phases of the development and operations processes, DevSecOps cultivates a business culture where development, security, and operations teams collaborate to maintain a secure, trustworthy, and speedy application delivery pipeline and operating environment, maximising uptime.

Threat Attack Vectors
Threat attack vectors are individuals or groups that cause digital damage to computer systems software and networks by exploiting vulnerabilities. Examples of threat attack vectors include phishing, ransomware, and malware attacks.

DevSecOps protects against a range of threat attack vectors:

- **Supply Chain Attacks** – By ensuring robust security measures are put in place at every stage of the development and deployment chain, DevSecOps prevents infiltration into systems through an outside partner or service provider.
- **Access to Development Chains for Malicious Injection** – DevSecOps practices limit access to the development environment and continuously monitor for any suspicious activities, preventing attackers from injecting malicious code.
- **Exfiltration of IP and Private Data** – DevSecOps secures sensitive data and IP by incorporating security measures right at the beginning of the development process.
- **Exploitation of Common Vulnerabilities and Exposures (CVEs) in Environment Services and Libraries** – DevSecOps proactively identifies, and patches known CVEs in environmental services, libraries, and developed products as part of the continuous integration/continuous deployment processes.
- **Poor Security Coding Practices Resulting in CVEs in Products** – By integrating secure coding practices and code scanning throughout the development process, DevSecOps identifies potential security concerns in software so they can be remediated prior to testing and deployment.
- **Injection of Malware into Production Through Compromised Deployment Practices** – The emphasis on secure deployment practices and continuous monitoring in DevSecOps reduces the risk of malware being introduced during deployment.

Challenges of Implementing DevSecOps

Despite its benefits, implementing DevSecOps can pose significant challenges for organisations and involves fundamentally shifting mindsets and processes. Successful implementation of DevSecOps requires an investment in training, tools, and education, to foster a continuous learning and improvement culture. Implementing DevSecOps requires a long-term commitment to security by all business teams.

Given the rising prevalence of cyber threats, this investment is not only valuable, but essential.

3 Building Trust

Many organisations lack the time and expertise to construct a fully comprehensive DevSecOps Factory.

Rather than implementing specific tools or processes, Consunet conducts activities that incrementally build trust with customers, trust between developers, trust within agile delivery teams, and ensure that trust is never eroded by threat attack vectors. This forms Consunet's Trusted DevSecOps Factory that can be incrementally adapted to meet team and customer needs.

Consunet's Trusted DevSecOps Factory

Trusted DevSecOps Environments
DevSecOps enables rapid conversion of ideas into live, secure, and trusted customer outcomes and services. Bringing ideas into Consunet's Trusted DevSecOps Factory creates high value products for customers without compromising security or privacy.

A DevSecOps Factory is not constructed overnight. It must evolve based on practice, learnings and experience with the potential products and their pitfalls. A DevSecOps factory should be tailored to meet specific customer, product, and environmental needs.

Consunet's Trusted DevSecOps Factory as shown in Figure 2, provides a blueprint to inform and guide organisations on the core elements to commence building trust with customers. The DevSecOps Factory should consider:

- **DevSecOps Services** – The core technologies and services used to access the environment (e.g. authorisation and authentication), to manage and monitor the environment, and to develop secure capability through the collection of available DevSecOps service functions. These include Pre-flight, Pre-Scan, Build/Test, Assemble, Scan, Publish/Deploy, and Observe.
- **Product CI/CD Pipelines** – DevSecOps Services are used to access shared secure and trusted code and artifact repositories through repeatable and streamlined Continuous Integration / Continuous Delivery (CI/CD) pipelines. Each pipeline reuses the DevSecOps services and is configured to meet each product's specialised needs and customer outcomes.
- **Secure Infrastructure** – Core physical environment(s) to store, host, process, and access the development and runtime environments. This includes the physical server and boundaries, network connectivity and integration points, and the ongoing availability, capacity and maintenance environments. Consunet can design, build and accredit for environments operating at a variety of security classifications to meet product(s) and customer(s) needs whether working in defence, financial, emergency services or other sensitive environments.
- **Collaborative Community** – Consunet's Trusted DevSecOps Factory also considers the ownership, IP Frameworks, shared Code/Artifact repositories, and works planning with external vendor and supplier environments, with the objective to deliver an interoperable, cohesive, and trusted outcome for customers and user communities in an efficient, extendable, and flexible manner.
- **Open Architectures/Solutions** – The use of Open-Source solutions (hardware and software), interoperability standards, and open architectures based on componentised or service-based solutions, enables organisations to quickly adapt and support robust and emerging capabilities or integration between products.
- **Trusted Governance and Processes** – Consistent and repeatable processes and practices to ensure security concerns are applied throughout the planning, design, development, testing, deployment, operation, and support activities, achieved by development and operations activities, and the IT Infrastructure Library (ITIL) for support activities. Other practices include the use of hardened common infrastructure images, trusted and regularly scanned repositories, and mandated DevSecOps security scanners and reporting services as part of Product CI/CD Pipelines. These practices all contribute to increasing trust levels and support the ongoing accreditation, authorisation, and authentication of the factory and the products developed within.

All elements combined provide a secure, trusted DevSecOps Factory and can be designed and configured to provide increasing level of assurance depending on customer environment risks.

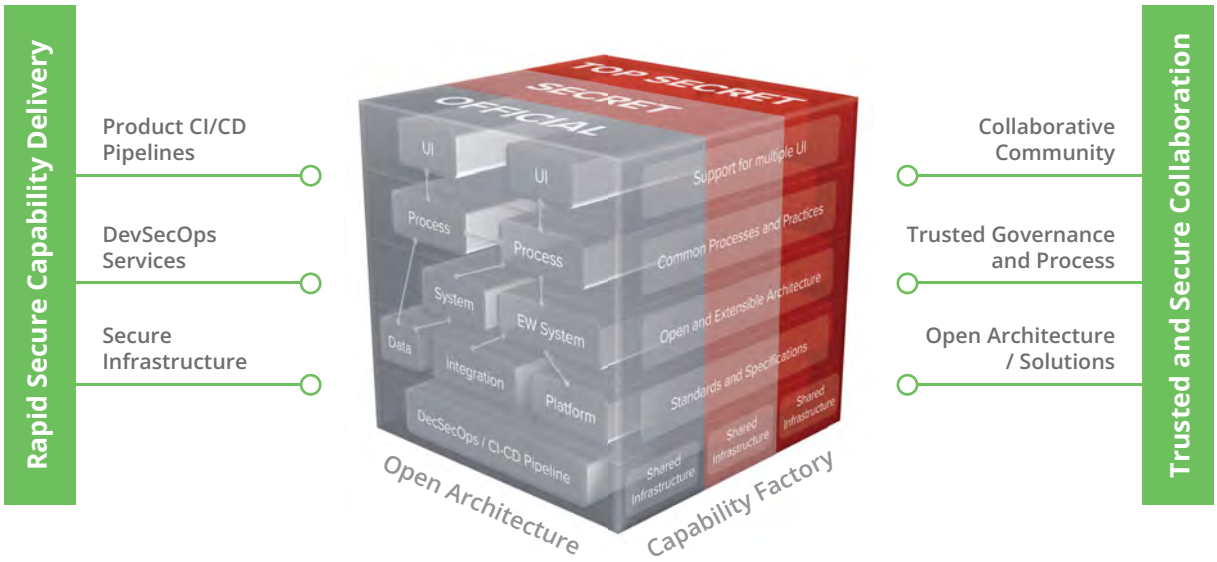


Figure 2 - Consunet's Trusted DevSecOps Factory

Assurance Levels

Assurance Levels are a means of measuring trust within a SDLC, and are described as follows:

Low Assurance – Low Assurance environments are when only minimal security controls are applied to safeguard against key threats to the environment, data and products. Typically, these are used in prototyping environments, or where the risk impact of breach, data loss or exploitation is extremely low. This is not a recommended approach, unless the development environment is considered a short-term, throw-away environment where complete loss, exfiltration or corruption can be tolerated.

Moderate Assurance – Moderate Assurance environments are recommended for when the resulting products are expected to be 'operational deployable' or 'enterprise ready' and operated by end users who require a certain level of confidence in the provenance and security of the product or service. The focus on this whitepaper is about how Trusted DevSecOps can achieve a Moderate Assurance level environment.

High Assurance – High Assurance environments are recommended for when additional security process, practices and technical controls are required to further increase the fidelity and verification of the pipeline, surrounding governance, test/production environments, and resulting products and services used by end users. This type of environment is required for products and services that deal with private data, financial services/data, commercially sensitive data, and national security related products and services.

This whitepaper describes the elements required to build a Trusted DevSecOps Factory to achieve both Moderate Assurance and High Assurance levels.

4 Building a Trusted DevSecOps Factory

To build a Trusted DevSecOps Factory and deliver secure, trusted software for customers, the following elements must be considered:

- **Supply Chain** – The Supply Chain is the first stage of the development pipeline that protects libraries, containers and sources feeding into the development environment and is the first line of defence in building a base level of trust. The Supply Chain includes the use of trusted repositories and is when security reviews and scans are conducted.
- **Continuous Integration / Continuous Deployment (CI/CD)** – Traditionally considered the focus of DevSecOps, CI/CD only represents a part of the overall Trusted DevSecOps Factory as shown in Figure 2; however, CI/CD is the powerhouse that provides the core automation, scanning, building and testing services to facilitate the timely transition of ideas to end user functionality; while building trust with scans, validation, checks and monitoring.
- **Monitoring** – Monitoring is focused on the operational capability and is when end users provide the critical feedback into the Supply Chain and CI/CD elements for the continuous improvement cycle as per the DevSecOps infinity cycle loop.
- **Lifecycle and Support** – Ongoing management of the environment, infrastructure, environment services, reusable services and security operations ensure that the capacity, performance and capabilities of the DevSecOps Factory meets current and future product/project needs.
- **Governance and Collaboration** – To ensure that auditing, policies, processes, and collaboration activities are considered, Governance and Collaboration provides the final assurance that all elements are working seamlessly together.

Each element is shown in Figure 3 and described in the following sub-sections.

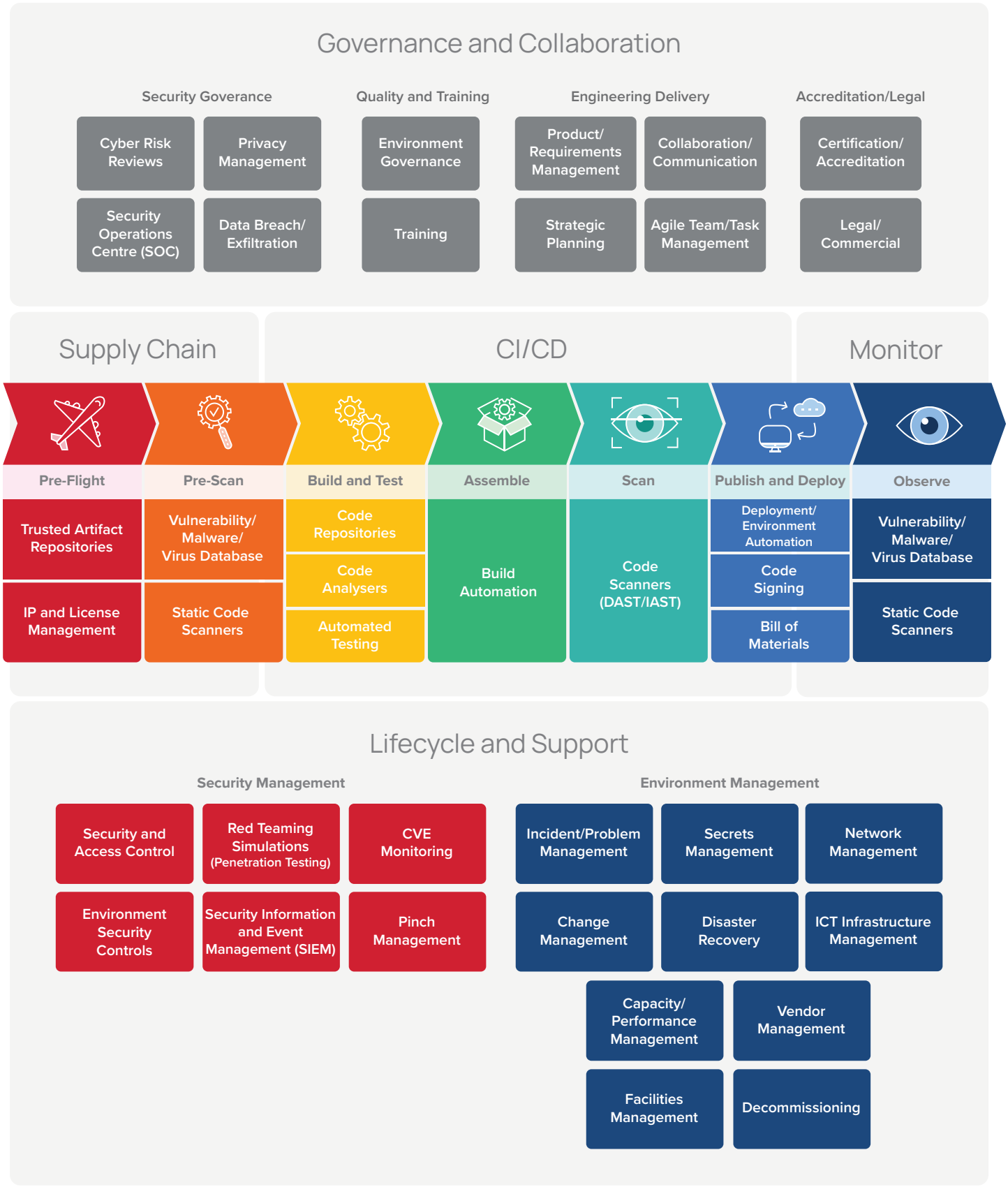


Figure 3 - Consunet's Trusted DevSecOps Factory Elements

4.1 Supply Chain

Malicious code can enter the SDLC via a compromised library, container image, or tool, creating vulnerabilities and in worst cases—facilitating the deployment of hidden malware. This can be controlled by securing the Supply Chain.

DevSecOps Supply Chain security involves securing all components in the software development process by implementing validation and verification processes. The raw materials (e.g. code, libraries, dependencies) used to build software, and the tools and systems used in the development, testing, and deployment process, are passed through these processes.

The Trusted DevSecOps Factory must provide security around source code repositories, libraries, containers and images used within development, or deployed within the environment. Validated and verified artifacts are stored in trusted repositories and regularly scanned. These are the Pre-Flight and Pre-Scan activities that then feed subsequent development build pipelines.

CASE STUDY

Log4j Supply Chain Compromise

The Log4j hack was a significant cybersecurity event that took place in late 2021. It centred around the exploitation of a critical vulnerability in the widely used Apache Log4j Java-based logging utility. This utility was heavily integrated into enterprise software globally, across a variety of industries and sectors.

The vulnerability, known as Log4Shell and officially identified as CVE-2021-44228, allowed an attacker to execute arbitrary code remotely. This effectively provided the attacker with control over the targeted system. The vulnerability's potential for damage was greatly amplified due to the broad adoption of Log4j in numerous software applications.

The discovery of the Log4j vulnerability triggered a wave of urgency as organisations hurried to patch their systems. However, due to the widespread use of Log4j, this was a Herculean task. In many instances, Log4j was embedded within third-party components, complicating the patching process. Following the release of an initial patch by Apache, a subsequent vulnerability was identified, necessitating another round of patches. This incident underscored the vital importance of staying current with patching and vulnerability management, as well as the potential risks associated with relying heavily on a single utility.

The Log4j hack is considered a watershed moment for cybersecurity. It demonstrated how interconnected and fragile digital infrastructure can be. The incident served as a stark reminder that vulnerabilities can exist within even the most trusted software components. When these components are as pervasive as Log4j, the effects are far-reaching.

Ultimately, the Log4j incident highlighted the need for comprehensive security strategies. These strategies should encompass not only reactive measures to deal with incidents as they occur, but also proactive approaches. By identifying and mitigating vulnerabilities before they can be exploited, organisations can significantly strengthen their defence against cybersecurity threats.

4.1.1 Pre-Flight

The pre-flight phase is crucial for setting the foundation for the rest of the development and deployment process. It involves ensuring that the tools, source libraries, containers and images used in development and operation of the environment are secure and reliable. This is achieved through Trusted Artifact Repositories and reviews of IP and licenses. Close relationships to activities within the Security Governance element provides validation and verification support in the form of Cyber Risk Review activities.

Trusted Artifact Repositories

Through Cyber Risk Reviews, standardised artifacts are verified, validated, and stored in artifact repositories, providing a trusted source for developers. Trusted Artifact Repositories enable developer and environment support teams to focus on the use of the artifacts with confidence that they have already been scanned. This eliminates artifact duplication and the potential for individuals and teams to consider artifact scanning and hardening as too difficult and skipped entirely exposing the artifact to unnecessary security risks and threats.

Artifacts include:

- **Operating Systems (OS)** – Specifically preselected versions of Linux or Windows that suits the individual organisation's security posture. These OS images must have the security controls and settings to 'harden' these images and have the monitoring hooks so when used, are automatically integrated into the operational monitoring environment.
- **Virtual Machines (VM)** – It is logical to package an operating system with a suite of applications or an application runtime. Once these are hardened and scanned, VM images should be stored in a trusted artifact repository.
- **Containers** – Applications packaged as containers include their application runtime and are executed within a container runtime/orchestration. These containers should be scanned and reviewed to check their embedded libraries, software applications and contents are suitable for running in the environment.
- **Base Software / Libraries** – Essential software and dependencies required for the application to run. This may include programming language runtimes (e.g. Java, Python), web servers (e.g. Apache, Nginx), databases (e.g. MySQL, PostgreSQL), and other commonly used tools and libraries or software used in development or environment operation.
- **Custom Codes / Libraries** – Organisations may have in-house tools, themes, libraries and settings to include to encourage re-use and provide a trusted repository for sourcing the latest versions of these shared resources.

The trusted artifact repository has multiple uses. It is used as a source for development activities, as interim storage throughout the CI/CD pipeline, and as part of the deployment and operation of the production runtime environments. The trusted artifact repository must be treated with a high level of security, as a compromised artifact repository can easily undermine all other parts of the DevSecOps trust chain.

Trusted Artifact Repository Maintenance
Artifact repositories require regular maintenance and upkeep. This involves regularly scanning repositories for newly identified CVEs, adding patched/updated versions, and removing compromised artifacts. Updates to security controls and tools may also affect artifacts so security updates are embedded into the source artifacts and therefore automatically adopted throughout the DevSecOps Factory.

Intellectual Property and License Management

In the context of DevSecOps, 'defined IP' refers to Intellectual Property (IP) that is documented, protected, and maintained within an organisation. It includes any proprietary software, code, algorithms, processes, or other creative works that are owned by the organisation. In DevSecOps, it is important to have clear policies and procedures to safeguard and manage IP throughout the software development and deployment lifecycle.

Licenses, on the other hand, in the context of DevSecOps, refer to the permissions and rights granted to users or organisations for the use, distribution, and modification of software or IP. Licenses play a crucial role in open-source software, as they define the terms and conditions under which the software can be used, modified, and distributed. When adopting open-source components or libraries in a DevSecOps environment, it is essential to understand and comply with the licenses associated with those components to ensure legal and ethical usage. Developers must also consider the trickle-down costs or implications when including libraries, software or even source code into their projects as some may result in additional licensing costs that need to be factored into product licensing costs for end users.

The Trusted DevSecOps Factory therefore requires the following practices for managing IP and licenses:

- **IP Management** – Establishing clear policies for identifying, documenting, and protecting IP throughout the software development and deployment process. This includes identifying ownership, managing licenses, and defining usage rights.
- **License Compliance** – Ensuring that all software components and dependencies used in the development pipeline comply with their respective licenses. This involves tracking and managing open-source licenses and understanding any obligations or restrictions imposed by those licenses.
- **Auditing and Risk Assessment** – Conducting regular audits and risk assessments to identify potential IP violations or license compliance issues within the development ecosystem. This mitigates legal risks and ensures compliance with relevant regulations and standards.
- **Education and Training** – Providing education and training to development teams and stakeholders about IP rights, license obligations, and best practices for managing IP and licenses within the DevSecOps process.

By effectively managing defined IP and licenses, organisations can ensure legal compliance, protect their IP assets, and maintain a secure and reliable DevSecOps environment. IP and License Management should be considered early on in the design and development process to reduce surprises discovered late in the release or operations phases.

License Poisoning

'License Poisoning' is a term used in the context of open-source software development to describe a situation where code licensed under a restrictive open-source license, such as the GNU General Public License (GPL), is incorporated into a proprietary codebase.

This code 'poisons' the proprietary license because the GPL and similar licenses have a 'copyleft' provision that requires derivative works to be licensed under the same terms. This means that the inclusion of GPL-licensed code in a proprietary project can potentially force the entire project to be released under the GPL, effectively turning the proprietary software into open-source software.

The OpenWRT project as an example, was affected by License Poisoning. In 2003 it was discovered Linksys had used components from the Linux project that were copyright (or copyleft) under the GPL, forcing them to release the firmware for their range of WRT54G routers.

4.1.2 Pre-Scan

The pre-scan phase refers to the initial preparation and setup activities conducted before any development. Pre-scanning ensures the source artifacts are verified before entering the artifact repository, also identifying risks and vulnerabilities. For environments that are isolated from external networks completely or where verified Malware/Virus definitions are required, this phase is also responsible for the management and maintenance of these source definition databases. By effectively executing the pre-scan phase, organisations can ensure a streamlined and focused security scanning process, leading to early detection and mitigation of vulnerabilities in the software development lifecycle.

Vulnerability, Malware, and Virus Database Scanners

Trusting in a single poorly validated scanner can lead to disastrous results. Vulnerability, malware, and virus database scanners are used to identify known threat adversary tactics and techniques. Each scanner is focussed on a specific set of potential threats or detection approaches. These scanners are also responsible for ensuring that source definitions and databases are regularly updated (global threat attack vectors work across time zones) and used on all artifacts contained within the Artifact Repository(s) or introduced into the development environment.

These scans are often conducted as part of Cyber Risk Reviews prior to the approval/denial decision when considering new software, libraries or other development tools, products and services. Also used as part of the network traffic analysers, these scans identify malicious network-based activity between networks or within the non-production and production environments within the Trusted DevSecOps Factory (see Network Management section).

Static Code Scanners

Another key scanner at this stage of the DevSecOps pipeline are static code analysers or Static Application Security Testing (SAST) tools. When used at the Pre-scan phase, code analysers are focused on finding issues with source libraries and code that are planned to be incorporated into development activities. As the name suggests, SAST tools analyse the source code, byte code and binary code when they are not running to detect code weaknesses or vulnerabilities. Different types of Static Code scanners are used to support different languages and frameworks, or to detect different types of vulnerabilities or code with poor security practices.

Code scanners are important to the Trusted DevSecOps Factory and are used multiple times in Supply Chain, Build, Test, and Publish phases of the development process. Continuous scanning identifies new concerns throughout the development process and product operations.

4.2 Continuous Integration / Continuous Deployment

CI/CD is the original focus of DevOps and the core element to the development process. CI/CD covers the activities of building/compiling the software, assembling the solution elements, scanning the code, publishing, and deployment to either non-production or production environments. All activities are automated and facilitated by CI/CD pipeline and deployment automation tools. These pipelines and automation tools rely on the Trusted Artifact Repositories and Code Repositories for sourcing trusted artifacts, sourcing code, and storing the final artifacts. The goal of the CI/CD pipeline is to automate all steps and streamline development to deployment with enforced security and quality gates so developers can focus on functionality and capability delivery, whilst accessing the security and quality benefits provided by the CI/CD pipelines.

4.2.1 Build and Test

Secure-by-design and secure-by-default principles are used to achieve secure programming practices. Languages that contain memory-safe and thread-safe features assist to strengthen the integrity of applications. Code Repositories, Code Analysers and Automated testing are the elements within a Trusted DevSecOps Factory that store the securely developed code and conduct the automated scans and tests to identify any security concerns to achieve the secure-by-design and secure-by-default principles.

Code Repositories

The source code repository is central to a DevSecOps environment, serving as both a collaborative platform and a definitive source of truth for developed products. Source Code Management (SCM) tools empower developers to simultaneously contribute to the same codebase (even the same functions) without inadvertently overriding another developer's work. An essential feature of SCM tools is the ability to create branches, enabling features or changes to be developed in isolation, and to be later merged into the main codebase. Additionally, these tools foster a robust code review process, where before changes are merged, they are reviewed and approved which ensures that quality and security are maintained at every step. By conducting code reviews, teams can spot bugs, logical flaws, and potential vulnerabilities early, reducing the cost and consequences of addressing these issues at later stages. Using clear guidelines, constructive feedback, and the assistance of automated tools, code reviews can significantly elevate the code's maintainability, scalability, and security. When code repositories are used with automated code analysers, code scanners, automated testing and automated build through the CI/CD pipelines provided by the Build Automation, developers gain access to automated security, quality, review and collaboration tools to streamline development activities, identify issues early, and can focus on user functionality development.

Development Environment

The development environment is the entry point for creating software products. It is the platform where code is written, tested, and deployed if required, for the first stages of quality assurance. To achieve a high level of integrity, the development environment itself must be secure and trusted. This requires implementing multiple layers of security controls tailored to safeguard the developers' workstations. Such controls can act as barriers that prevent the execution of malicious code, whether it arrives in the form of malware, unauthorised remote access, or even as code snippets that may inadvertently be copied and pasted during the development process. Endpoint security focuses on securing individual workstations or devices that connect to the development environment. This includes installing antivirus software, firewalls, application controls and other intrusion detection and prevention systems to the developers' endpoints (e.g. laptop, desktop or virtual machines). These controls can monitor network traffic and system behaviour to detect and block malicious activities, thereby offering a layer of protection against potential security threats.

An Integrated Development Environment (IDE) is more than a tool for writing code; it is a comprehensive platform that augments a developer's efficiency, accuracy, and collaboration capabilities. A notable advantage of many modern IDEs is their built-in linting and formatting tools. Linters actively scan code for potential errors or deviations from best practices, while formatters ensure that the code adheres to a consistent style, enhancing its readability and maintainability. Using linters and formatter in combination with real-time security plugins, IDEs can proactively identify and prevent vulnerabilities, ensuring that the code is functional, clean, and secure from the outset. Furthermore, IDEs often incorporate integrated version control interfaces, fostering transparency, traceability, and collaborative efficiency in the codebase. All this occurs before code is pushed into source code repository which prevents bad code from entering the CI/CD pipeline.

Standardised and templated development environments facilitate fast onboarding of developers and ensure consistency between development teams so developers spend less time building and debugging their development environment and can focus on building product functionality and improvements.

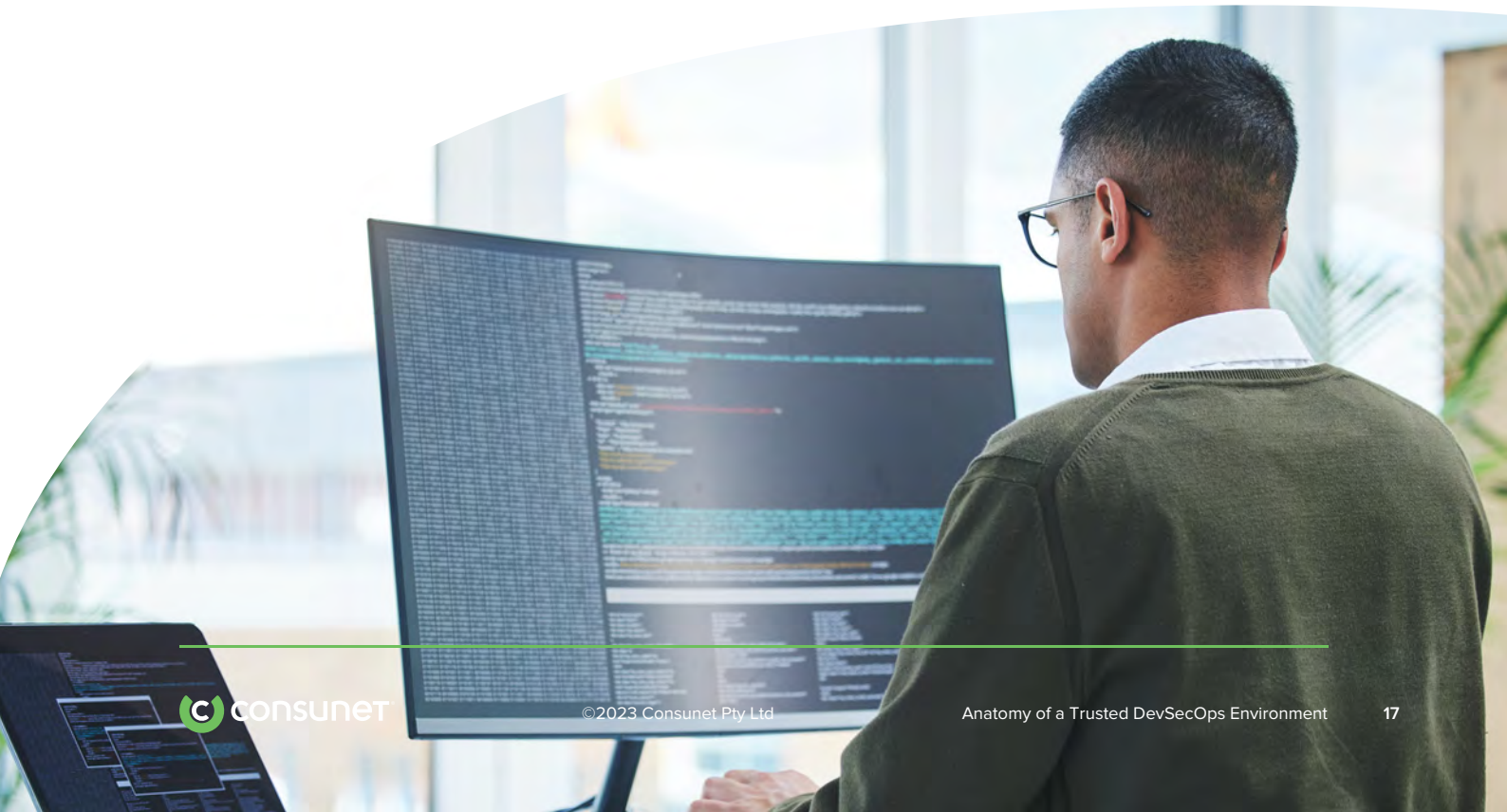
Code Analysers

A secure development environment and using an IDE is not enough to ensure clean, maintainable and secure code. Once pushed to source control, the CI/CD pipeline takes over and performs additional checks and analysis. Linters enforce coding standards by identifying both syntactic and stylistic issues, ensuring the code is not only secure but also maintainable and consistent with established best practices. Software Composition Analysis (SCA) tools focus on external dependencies, especially open-source components, verifying that vulnerabilities are not introduced. SAST analyses application source code, byte code or binary code, while in a non-running state to detect security vulnerabilities such as SQL Injection or buffer overflows.

Automated Testing

Unit tests dive deep into individual pieces of code, be it functions or methods, validating that they operate as intended in isolation. On the other hand, integration tests ascertain that these discrete components properly interact with one another, guaranteeing the software's functionality as an integrated whole. A pivotal aspect of both testing methods lies in the utilisation of effective test data. While real-world data scenarios can provide valuable insights into how a system would behave in production, the use of actual customer data poses significant privacy concerns. Given the stringent regulatory landscape around data protection and the ethical imperative of preserving user privacy, it is vital to avoid direct testing on genuine customer datasets. Instead, randomised, anonymised data offers a more suitable alternative. Such data replicates the diversity and complexity of real-world data without compromising individual privacy, allowing for comprehensive testing that respects and upholds user confidentiality.

Automating tests as much as possible ensures consistency, repeatability and agility. Once set up, automated tests execute in the same manner every time, eliminating the variability introduced by human testers. This repeatability ensures that regression issues can be identified instantly when code changes are introduced. Secondly, automation dramatically speeds up the testing process ensuring that defects are detected and addressed early in the development cycle. Furthermore, automating tests frees up valuable human resources, enabling them to focus on more complex and nuanced testing scenarios, enhancing overall software quality.



4.2.2 Assemble

The assemble phase in a CI/CD pipeline encapsulates several critical processes that transform code into deployable artifacts, ensuring their integrity, security, and consistency throughout the deployment. This is achieved through Build Automation, which acts as the engine for many of the DevSecOps processes.

Build Automation

By automating the build process, changes in the source code are compiled and packaged in a consistent and repeatable manner, ready for testing and deployment. Errors and incompatibilities are detected early in the CI phase, leading to faster feedback loops for developers. In the CD phase, automated builds guarantee that what's being deployed is consistent, having undergone a standard process. Each pipeline execution employs a unique runtime environment, ensuring independence, and preventing artifact cross-contamination. Access to resources and environments is branch-dependent; main branch pipelines access production, while merge request and development pipelines are confined to staging and development environments respectively. This safeguards access to production systems from potential malicious code. During the build, binaries or libraries created between steps are stored in a distinct cache for each run. Required build dependencies are sourced from trusted artifact storage, and post-build artifacts are directed to a staging repository. Multiple pipelines provide support for tailored product-focus operating environments, parallel project development and support for specialised security, AI, testing, and other auxiliary systems to be integrated.

GUIDANCE: Container Hardening

Container images or VM templates should be secure-by-default, the best practices being:

- **Minimal Services** – Limiting the container or virtual machine to only essential services curtail its attack surface, reducing vulnerability points.
- **Streamlined Images** – Starting with a minimal base image and layering only necessary components diminishes the potential threat landscape.
- **Hardened Images** – Using best practices around application control, hardening of application configuration, and hardening of operating system configuration.
- **Non-Root Processes** – Containers should run processes as non-root users, reducing risks of potential compromises escalating to the host system.
- **Immutable containers** – Containers that are not changed at runtime; when an update is required (for patching or application upgrades), the entire container is replaced, simplifying maintenance, and ensuring repeatability in deployments.

4.2.3 Scan

Application security testing can assist software developers in identifying security vulnerabilities in their applications. Security scans should encompass multiple methodologies such as SAST, Dynamic Application Security Testing (DAST), and Interactive Application Security Testing (IAST), each playing a fundamental role in identifying and mitigating vulnerabilities. Each method serves a distinct purpose, tailored to securing different threat attack vectors and should all be conducted to provide comprehensive test coverage.

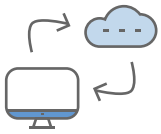
Code Scanners

Choosing the right security scanning tools is important to effectively identify vulnerabilities and risks. There are various tools available for different types of scans such as DAST, IAST, SCA, and more. The selected tools need to be configured properly based on the requirements of the application or system. The following provides a summary of recommended tools:

- **Static Application Security Testing (SAST)** – SAST analyses application source code, byte code or binary code, while they are in a non-running state to detect the security vulnerabilities such as SQL Injection or buffer overflows. At this stage of the pipeline, SAST will be used to scan the assembled binaries.
- **Dynamic Application Security Testing (DAST)** – DAST is a security testing methodology that evaluates a web application in its running state, typically from an external perspective, without access to the underlying code. This usually includes attempting to perform SQL injection attacks and Cross-Site Scripting attacks (XSS) or fuzzing of inputs to Application Programming Interfaces (APIs) to find unintended behaviour.
- **Interactive Application Security Testing (IAST)** – IAST analyses code for security vulnerabilities while the application is run by an auto-test suite, human tester, or any activity 'interacting' with the application functionality. This is similar to DAST where it looks at the application from an external perspective but with access to the underlying running code through debuggers or access to the internal state of the application. This provides visibility on the behaviour of the product with various inputs as another way to find vulnerabilities in the code.
- **Software Composition Analysis (SCA)** – SCA analyses the components, libraries and binaries that are used to create the full application. This is used to create a Software Bill of Materials (SBOM) which can be checked for license violations or known CVEs.

To achieve comprehensive coverage and uncover a wide range of vulnerabilities in depth, it is advisable to use a variety of scanning tools. Furthermore, for each type of scan, employing multiple tools of the same type can enhance detection capabilities and remove any potential detection bias.

4.2.4 Publish and Deploy



The 'publish and deploy' stage in a CI/CD pipeline is crucial for finalised code artifacts and deployment to appropriate environments and repositories. Publishing involves signing the produced packages and creating the accompanying SBOM to enhance their security and credibility. The deployment method for these packages varies based on the nature of the product. For instance, a library might be uploaded to a trusted artifact repository, while an application might necessitate the setup of an entire runtime environment like Kubernetes for execution, which is coordinated through the deployment automation services.

Code Signing

Code signing involves leveraging cryptographic techniques to attach a digital signature to software code. When a piece of software is signed using a developer's private key, recipients can use the corresponding public key to verify the code's source. This verification process establishes a chain of trust, ensuring that the software has not been maliciously modified since it was last signed. For added assurance, signed code should then be stored in the trusted repository. These layers of trust build confidence that what is deployed is the correct intended code.

Software Bill of Materials

The Software Bill of Materials (SBOM) provides a comprehensive inventory of components, libraries, and modules that are incorporated within a software product. Each entry in an SBOM typically includes details such as the component's name, version, origin, and applicable license, ensuring transparency around both IP and usage rights. An SBOM helps developers, security professionals, and end users to understand what's running in a given software environment. This documentation is invaluable for security assessments, as it aids in identifying known vulnerabilities associated with specific components. Additionally, in the event of security incidents or breaches, having a structured SBOM accelerates the process of pinpointing compromised elements. As software development increasingly leans towards the use of reusable components, maintaining an up-to-date and accurate SBOM is paramount for ensuring transparency, compliance, and enhanced security throughout the software's lifecycle. Storing all dependencies specified in the SBOM within the trusted artifact repository further enhances trust and security. This approach mitigates potential attacks on external repositories that could compromise the product.

Regulations in finance, healthcare, and critical infrastructure sectors often mandate organisations to validate their software against known vulnerabilities or licensing benchmarks. A thorough SBOM enables organisations to efficiently ensure and showcase compliance, guaranteeing that software elements are transparent, appropriately licensed, and align with industry standards.

Deployment and Environment Automation

Orchestrating modern applications designed using containers and microservices with distributed deployment designs can be difficult to deploy as each component needs to consider the relationship between itself and other components to act as a whole. Use of development, deployment and environment automation tools achieve this in conjunction with the following strategies and tools:

- **Infrastructure as Code** – The use of deployment architecture defined as a set of scripts and configurations that can be 'executed' to deploy an environment. This is achieved using infrastructure deployment and automation tools such as Terraform or Ansible or even custom developed scripts and solutions.
- **Configuration as Code** – The use of a source code repository to store the configuration of infrastructure and applications. This can then be used by Continuous Delivery tools to set up and configure environments.
- **Container Orchestration** – Container Orchestration provides a way to manage the full lifecycle of containerised applications, scheduling them to run and performing checks to ensure the health of these containers.

- **API Gateways** – API Gateways act as the entry point to applications, directing requests and managing microservices. This allows the API endpoints to change with the API gateway coordinating traffic.
- **Service Mesh** – Service Mesh is an infrastructure layer facilitating seamless communication between services, ensuring fault tolerance, observability, and security.
- **Messaging Services** – Messaging Services streamline inter-service communication by providing a resilient and scalable method of passing messages in a decoupled manner, guaranteeing data integrity. Typically, this is achieved using a publisher-subscriber architecture: the message bus receives messages from publishers, and then subscribers retrieve those messages.
- **Integration Services** – Integration Services are used to connect, orchestrate and facilitate data transfer between different systems, applications, or databases. They streamline the flow of data by performing Extract, Transform and Load (ETL) operations to ensure consistent information exchange across disparate platforms.

Once deployed, the running products need to be put through rigorous testing to ensure that the code works as a whole, meets business requirements, and performs well under load. Testing types in this phase are:

- **Integration Testing** – When individual units or components of a software application are combined and tested as a group. Its primary purpose is to identify any discrepancies, inconsistencies, or communication issues between these integrated units. By ensuring that integrated components work cohesively, integration testing detects interface defects and guarantees that data flows correctly between modules.
- **Performance Testing** – Used to determine how a system performs as a whole in terms of responsiveness and stability under a particular workload. This can be used to determine which parts of the application are bottlenecks and should be optimised or can be scaled up or down.
- **Acceptance Testing** – Serves to confirm the alignment of the product with specified business requirements. It validates the software's functionality, usability, and performance against the set criteria. By acting as a final checkpoint before release, acceptance testing verifies that the software meets stakeholder needs and industry standards, mitigating potential business risks associated with product misalignment.
- **Secure Content Automation Protocol (SCAP)** – A collection of standards for expressing and automating security policy, vulnerability management, and measurement. These can be used to automate security checks against known best practices such as the Australian Cyber Security Centre's Essential Eight.

These testing environments can also be used for running the code scanning from the Scan phase where dynamic operation of the product application is required (e.g. a live instance running).

Continuous Deployment is automating the steps of deploying the product into an environment. Continuous Delivery builds upon this and is specifically deploying the application to production in a way that minimises disruption and downtime. There are several methods to achieve this:

- **Go/No Go** – This deployment approach uses a stage gate that acts as a final validation point, determining the readiness of the release for production.
- **Rolling Releases** – Rolling Releases are updates that are continuously rolled out to users without a fixed schedule or version number. Instead of waiting for a major release, enhancements, bug fixes, and new features are delivered as they are developed and deemed stable. This approach ensures that users always have access to the latest improvements and reduces the need for large-scale, disruptive updates.
- **Staged Releases** – Also known as Canary Releases, Staged Releases involve deploying new software updates to a subset of users initially, rather than to the entire user base at once. By releasing changes in phases or 'stages', developers can monitor and address any potential issues with minimal impact. If no problems arise, the update is gradually rolled out to larger portions of the user base, ensuring stability and a smoother user experience. This approach reduces the risk of widespread issues, enabling developers to catch and rectify problems before they affect all users.

- **Blue/Green Deployment** – The simultaneous operation of two distinct full production environments (the existing version and the imminent release). This facilitates uninterrupted releases by moving users from one to the other. A/B testing with live users can also be achieved using this deployment method by using an API gateway or service mesh to send the traffic from a portion of users to the new deployment.

4.3 Monitor

Although visualised at the end of the pipeline in Figure 3, the Monitor stage can also be considered as the beginning of the process as shown in the infinity DevSecOps loop in Figure 1. The Monitor stage is the key step for observing possible security concerns, performance concerns, and capacity concerns. This stage will support investigation/remediation actions by the Security Operations Centre, Engineering and Environment Management teams to capture impacts, failure modes and trends. This improves application/product performance for end users, schedules proactive maintenance actions, and enables planning for future system capacity or product features/functions. As a result, the Monitor stage is responsible for providing visibility of the Supply Chain security, the performance and outcomes for the CI/CD pipelines, the performance of all the non-production and production environments, and the status of vulnerabilities and cyber security threats. A DevSecOps environment provides clear reports, alerts and investigation tools to achieve these outcomes.

4.3.1 Observe

Observability is achieved through Configuration Management to ensure that verified and approved changes flow into the production runtime environments and through continuous monitoring and reporting functions using logs, reporting and auditing functions within the environment and software products. These observability functions apply to the production environment for end user products, the underlying environments used in production, and also support the development DevSecOps services. Treating the development environment as a production environment elevates the importance of product development teams, as delays and impacts to developers has a direct impact on continuous delivery for end users.

Configuration Management

The goal of configuration management is twofold: ensure that teams supporting users of the products have the required reference materials and configuration to rectify issues or support user requests; and provide a quality gate before release to production that all necessary checks and balances have been completed. Reference material may include asset registers, application user/system documentation, database, container, infrastructure, security, IP registers, software licenses, and related technical data. This practice is often supported with IT Service Management systems and document version control systems.

To automate the configuration management register, Infrastructure as Code and Configuration as Code principles are recommended. These principles manage the configuration/deployment of the environment and elements as code/scripts within code repositories and include them as part of the patch release. This 'as Code' approach allows for change and release management processes to include not only traditional product configuration management, but also the Infrastructure Deployment / Management Code as a coherent release bundle. This approach provides an additional security trust level as the Infrastructure Code is reviewed, validated and security checked along with the product code, incrementally increasing trust levels for the environment itself and not only the deployed user software.

Configuration Management also supports legal and licensing requirements through technical data management, IP registers, license registers and how these elements are used within the environment's and product's configuration hierarchy. Additional license and compliance scanning tools can be used to ensure adherence with governance guidelines and rules if required for the product/environment’s trust assurance level.

Observability

Observability is focused on proactively and retroactively identifying issues (both current state and historical issues) through live monitoring of the environment systems and services and the progress of processes/workflows. This enables support and development teams to identify capacity or performance trends, and to access historical logs for fault finding or root cause analysis. Observability is achieved through a central logging solution with an associated reporting, dashboard and searching function. Often application/product specific logging and reporting solutions are used in conjunction with the central logging solution for deep dive analysis activities but with shorter log retention periods than the central logging solution.

This solution will also support other functions within the Trusted DevSecOps Factory such as:

- **Security Governance** – for security event analysis, threat intelligence analysis and related security operations centre activities.
- **Environment Management** – for capacity planning, incident/problem management, network management and ICT service management.
- **Engineering Management** – for product performance analysis, product bug analysis and user behaviour analysis.

For more advanced DevSecOps environments that require higher assurance levels, the use of observability functions for security governance is required. This changes the focus of the Monitor element from reporting and analysis to proactively alerting or automating response approaches. Security monitoring can include traditional security measures such as networking logs, Web Application Firewall (WAF) logs, Intrusion Prevention System (IPS) and vulnerability scanners but also the use of embedded security within the products/applications deployed referred to as Runtime Application Self-Protection (RASP). RASP is where developers have embedded the security analysis, logging and alerting functions within the application itself to provide runtime application protection and support threat intelligence analysis used in the security governance activities.

4.4 Governance and Collaboration

Governance is critical in setting a culture of success within organisations. The goal of governance is not to control, but rather facilitate and guide the delivery of quality solutions to end users with high levels of trust and security. Success is achieved through both bottom-up and top-down governance activities; bottom-up activities including secure coding practices applied by developers on product development; and top-down strategic governance activities including holistic policies/procedures/audits/organisation structures that are applied across products, projects, teams and environments.

Governance continues after the product is in the operating environment, often referred to as the Authority to Operate (ATO) governance gate. Governance activities support early idea generation, guide design and build activities, and guarantee that quality and security reviews are conducted, and gates are passed. Governance ensures audits are performed to identify areas for improvement or address concerns when they arise. The four key governance domains of Security Governance, Quality and Training, Engineering Delivery, and Accreditation/Legal services are described further.

4.4.1 Security Governance

Security Governance provides continuous security services throughout the DevSecOps environment and to the products developed and deployed within these environments. Security is considered as a core element of each activity and not as an afterthought. Security governance activities are considered as additional support functions to the security elements contained in other activities within the complete DevSecOps environment. For example, Cyber Risk Reviews provide additional specialist skills to the existing design activities already considering security as part of the product design activities during Build and Test. Security governance is tailored and organised to best provide supporting security services and guardrails so Developers using the environment can focus on secure coding/design and delivering customer product functionality, whilst relying on Cyber Security expertise to support them as part of the security governance roles and activities. This approach allows for a balanced, risk-based approach to be applied to best delivery, agile streamlined capability outcomes while building trust and maintaining security.

Cyber Risk Reviews

Supply chain risks are an often-overlooked cyber security threat source when designing and building new products or operating DevSecOps environments. To increase the security posture, Cyber Risk Reviews should be completed on all software products, libraries, container images, operating systems and other artefacts that are used to either operate the development environment or as part of product development and operation. These reviews support the Pre-flight and Pre-scan activities and are focused on considering the pedigree of the item, contributors to the development, current and historical vulnerabilities, vendor performance and reputation.

Security Operations Centre

The Security Operations Centre (SOC) provides an overarching view of the Trusted DevSecOps Factory's security posture. The SOC uses the Observability function to provide log analysis and Security Information and Event Management (SIEM) functionality. The SOC is also responsible for ongoing threat modelling and vulnerability analysis through monitoring of external cyber threat reporting and the outputs of the various security scans conducted in the Trusted DevSecOps Factory's Pre-Scan, CI/CD Scan, and dynamic scan activities. When vulnerabilities are identified, or cyber incidents occur, the SOC is responsible for the communication, coordination and reporting activities to remediate these occurrences in a timely manner, and for updating processes, tools and practices to bolster the cyber security fitness levels. Over time, security access and configuration can often drift or the needs change. To identify and remediate these gaps, the SOC is responsible for conducting regular audits of access permissions and security configurations to ensure that they remain appropriate and coordinate actions to update and close these gaps.

Privacy Management

The solutions/products developed within the Trusted DevSecOps Factory are dependent on the type of data stored and processed in production and non-production environments. Data must be considered for special handling controls. Types of data that need special handling include:

- **Personally Identifiable Information (PII)** – Identification numbers on drivers' licenses or other documents that can be distinguished or traced to the individual.
- **Personal Information (PI)** – Details about a person such as date of birth, names and addresses.
- **Financial data** – Company or individual financial information, credit card numbers or other financial information which require special handling requirements.
- **Sensitive/Classified data** – Classified data, that if released to uncleared individuals or groups could have personal, national or company negative impacts.

For some data, 'need to know' principles apply. This means restricting storage or access to the owner of the data and mandatory parties only. Another implication is that during development some data is unavailable or is suitably redacted/obfuscated to reduce the risks if data is breached, exfiltrated, corrupted or lost. Suitable communication plans must be in place for reporting, notifying and adhering to the applicable laws for systems and customer locations. A suitable privacy policy should be in place for how to manage, control and handle privacy related data both in production and non-production environments.

Scans conducted during CI/CD and the logging and monitoring phases within the Observability activities can provide active and automated audits of the development and production environments. Scans identify data requiring additional security controls and can also provide alerts on data stored incorrectly in the scanned location (e.g. finding credit card details in development databases).

Data Breach and Exfiltration

In the event of a data breach, users, customers, and relevant authorities must be contacted. Processes and contact detail registers must be in place to support this. To proactively mitigate data breaches and exfiltration actions the non-production and production environments must have suitable logging and monitoring in place to monitor key data items and movement of these items within and between network boundaries. Transfers between network zones should be closely monitored with regular audits and reviews of the monitoring rules, access credentials and logs to ensure they remain appropriate. Additional proactive actions are penetration tests conducted as part of the ongoing security management function within the Lifecycle and Support domain to identify that controls are effective.

4.4.2 Quality and Training

Organisations build DevSecOps with the required set of processes, governance, and technology, but these can erode over time through a lack of maintenance and the bypassing of tools and controls, undermining the purpose of the environment and compromising the trust outcomes. Two critical focus areas to ensure this degradation does not occur is Continuous Improvement through the Environment Governance and Quality Management System, and the ongoing Training of developers, system administrators, project delivery members and associated users of DevSecOps. Through regular reviews, processes, practices and controls, DevSecOps can be improved and optimised to the changing needs of customers, developers and the security landscape. Training provides the means to onboard new people and update individuals on the best practices of using DevSecOps, or to capture feedback for future improvements.

Environment Governance

Environment Governance considers the roles responsible for the standing operational procedures (SOPs), policies, and controls required to achieve the key target outcomes of the Trusted DevSecOps Factory. These roles are:

- **Chief Information Security Officer** – responsible for the overall security posture of the Trusted DevSecOps Factory and resulting products.
- **Engineering Manager** – responsible for the development practices and the quality of the resulting products and their inherent security.
- **Environment Manager** – responsible for the optimised availability of the underlying infrastructure, DevSecOps services and technologies used to conduct development and run the non-production/production products and services.

Additional roles are required for project delivery, development, and systems administration to fully achieve end-to-end delivery, support, and operation of the Trusted DevSecOps Factory; however, are outside the scope of this whitepaper.

The Quality Management System (QMS) contains a complete set of policies, SOPs, handbooks and training guides that are regularly reviewed and updated. This QMS provides the authoritative reference library for all aspects of the Trusted DevSecOps Factory.

Environment Governance also considers the overall architecture and growth of the Trusted DevSecOps Factory so the tools, services and systems can expand or contract based on the customer needs and product needs. The Environment Management functions then facilitate these architectural changes.

Training

Training covers all aspects of the Trusted DevSecOps Factory and uses the guides, procedures and details contained within the QMS. Training is provided to developers, systems administrators, project managers, and all other team members in the available tools and processes and the Trusted DevSecOps Factory's objectives and principles. Individuals are empowered to apply security and trust practices, provide feedback, and implement improvements to increase the trust levels of the capability.

4.4.3 Engineering Delivery

The primary objective of a Trusted DevSecOps Factory is to turn great ideas into great customer outcomes. The engineering delivery function achieves this by using the Factory's services, processes, and technology environments to build customer products and services. The engineering delivery function covers the planning, coordinating, communicating and reporting elements of building the solution and can be applied to existing agile or waterfall-based project management approaches.

Product and Requirement Management
Prioritising customer requirements, architectural backlogs, environment development and support tasks are essential in transitioning ideas into secure trusted products for users. Requirements engineering, systems engineering, solution architecture, and product management activities use the Agile Teams/Task management, collaboration/communication, and strategic planning functions to achieve these goals. The DevSecOps Factory must include tools to support requirements management, feature management, and bug management to support the future product design and needs development to incrementally deliver product functionality to customers.

Open Standards, Open API, and flexible products all enable interoperability between products as well as products working within the customer's target ecosystems.

Strategic Planning
Engineering delivery must consider not only the short and medium term needs of the product but also the long-term and strategic priorities and context. Strategic planning considers elements such as:

- All products being developed within the Factory and the interplay between products and their individual roadmaps as a whole.
- Strategic priorities of the enterprise and customer contexts.
- Geo-political and financial environment and trends.

Implementation of this strategic plan can be achieved using Collaboration/Communication tools (e.g. documents, chat platforms) or if available within the road mapping and task management functions (e.g. Planning Increments, enterprise backlogs).

Collaboration and Communication
Effective development, delivery, support and management requires efficient and effective communication and collaboration. To facilitate this outcome, the Trusted DevSecOps Factory should include collaboration and communication services such as:

- Documentation libraries and file shares.
- Secure File Transfer and processes for the secure transfer, scanning and auditing of files entering and existing in the environment.
- Online documentation or wiki service for collaborative editing of shared procedures, guides, documentation, references.
- Online/Realtime communication such as text chat and/or online video conferencing functions.
- Group calendars for planning events, meetings and other time-based activities.

All these services provide methods for teams to work together on content, capture reference materials and contribute to a common knowledge repository in alignment with each product's communication plan.

Agile Teams and Task Management

The need for multiple teams and developers to plan and track completed tasks remains essential. The DevSecOps Factory must include suitable tools that support delivery methodologies for task tracking, monitoring delivery progress, managing finance, managing risks, and planning long-term roadmaps. Examples of these tools are Scaled Agile Framework (SAFe), Scrum, Kanban, and Extreme Programming. These tools are used throughout the Factory as part of early planning and prioritisation, design and architecture activities, implementation and testing, deployment management, ongoing support and operations, and even governance controls and auditing. As a result, the task management solution will become a focus point for teams and members.

4.4.4 Accreditation and Legal

Although not typically considered a part of a Trusted DevSecOps Factory the legal aspects of the software, libraries and products use to design, build and operate can have a significant financial or functional impact if incorrectly managed. Likewise, the environment and products built and deployed within the DevSecOps Factory should be subjected to certification and accreditation activities to check that the required controls have been considered, implemented appropriately and are reviewed regularly.

Certification and Accreditation

Certification is used to verify that the environment, product or process applies a particular standard, set of controls and adherence to a certification program. Accreditation is used to verify an appropriate level of quality within the applicable standard. Often certification and accreditation are used interchangeably and depending on the external governance requirements for the Trusted DevSecOps Factory itself or the operating environment of the resulting Products, there may be certain certifications and accreditations to be acquired before or while using the Factory and products. Related programs and standards to be considered include:

- **Australian Infosec Registered Assessors Program (IRAP)** – A program for completing IRAP security assessments against the Information Security Manual (ISM) requirements and used as input into subsequent Australian Defence and government agency certification and accreditation processes.
- **Cybersecurity Maturity Model Certification (CMMC)** – US Department of Defence maturity model used for certifying the level of cyber security maturity based on the processes and controls applied within the system.
- **National Institute of Standards and Technology (NIST) SP 800-161** – Cybersecurity Supply Chain Risk Management Practices for Systems and Organisations providing guidance on managing ICT related supply chain risks.
- **ISO/IEC 27001:2022** – Part of the ISO/IEC 2700 family of standards with policies and procedures for the management of organisation's information risks with the ISO/IEC 27001:2022 standard focused on the Information Security Management System to control information security risks.

This is a sample list as different standards, certification and accreditation programs exist for each geographical and political region and must be analysed and considered for the specific DevSecOps Factory use cases, and the products developed and deployed within. Additional industry specific items may also need to be considered for defence, finance, health, or other user communities handling personally identifiable and sensitive data.

Legal/Commercial

Intellectual and legal terms must be considered when selecting products and libraries. To facilitate this, the DevSecOps Factory governance element maintains an IP and license register including where and how these terms are being used (see Configuration Management in the Monitor domain).

Other Legal and Commercial aspects that need to be considered are how environment access controls are defined to support suitable segregation and export controls (e.g. International Traffic in Arms Regulation (ITAR)), confidentiality limitations, privacy requirements, financial regulations, security

release/access limitations, and other local or international laws and regulations. These apply either to the operation of the capability or the use of the products within.

As the products evolve and develop over time, and as new customers come on board or existing customers leave, the need to regularly review and update the Legal and Commercial elements exists to ensure that coverage remains and is up to date with the latest legal and commercial requirements.

4.5 Lifecycle and Support

DevSecOps is a holistic approach to the entire software development lifecycle, which not only ensures software is delivered faster using agile methods, but also guarantees a value-added, end user experience. DevSecOps includes the development and deployment phases as well as the ongoing management and maintenance phases crucial for sustaining security and performance, which when optimised, provides an outstanding experience for an organisation's customer community.

4.5.1 Security Management

Automation and advanced security tools have revolutionised how organisations approach cybersecurity by flagging known vulnerabilities and enforcing security protocols efficiently. Automation on its own however, is not a complete security solution. Automation lacks the nuanced reasoning that a human can provide.

Security professionals perform in-depth environment and code reviews and identify subtle irregularities or complex threats that automated tools overlook. Their role is critical in the ever-evolving landscape of cyber threats, where the limitations of automation can be exploited by sophisticated attackers. An effective Security Management strategy must strike a balance between leveraging cutting-edge tools and nurturing a skilled team of experts capable of addressing the details that machines miss. The following security management tools and practices operate in conjunction with the Security Governance function described earlier in this whitepaper.

Identity and Access Management

Identity and Access Management (IAM) ensures that only authorised users can access specific resources and perform permitted actions within a computing environment. It encompasses a range of capabilities such as authentication, authorisation, user provisioning, role-based access control, and auditing. IAM is not only focussed on controlling access to traditional infrastructure components (servers and databases), it also includes managing permissions for CI/CD pipelines, configuration management systems, and API endpoints within microservices architectures.

Integrating IAM into DevSecOps practices adds a robust layer of security by preventing unauthorised access and potential breaches. Automation plays a critical role in IAM; automated IAM solutions can dynamically adjust permissions based on policies and real-time data, enabling a highly responsive security posture. For example, short-lived credentials can be automatically rotated, reducing the window of opportunity for attackers even if they steal credentials. This can be further improved by using 'Security as Code,' wherein security configurations and controls are treated as version-controlled code and can be audited and refined continually.

Effective onboarding and offboarding processes are critical. During onboarding, automated provisioning and initial security training ensure that new team members have the necessary, but minimal, permissions to perform their jobs, thereby adhering to the principle of least privilege.

Offboarding should include automated deprovisioning of access rights to eliminate potential security risks.

Best practices in this domain include:

- **Multi-Factor Authentication (MFA)** – MFA combines something a user knows (password), with something a user has (a device to receive a token), or with something a user is (biometric verification). MFA adds a complex security layer to prevent attackers gaining unauthorised access.
- **Least Privilege Access** – Granting users only the permissions they need to perform their jobs. This reduces the risk of a user inadvertently or intentionally compromising security.
- **Role-Based Access Controls (RBAC)** – Assigns permissions to roles rather than individuals. Users can be assigned one or more roles, making it easier to manage and audit permissions.
- **Secrets Management** – Store secrets such as API keys, tokens, and passwords in a secure and encrypted secrets management service instead of embedding them in code or using configuration files.
- **Single Sign-On (SSO)** – SSO integrates with IAM to reduce password fatigue and simplify the management of multiple services.
- **Using Certificates for Authentication** – Certificates add more authentication than relying solely on traditional username/password-based credentials. This is especially relevant for machine-to-machine interactions where these can be stored and managed from the secrets management server.

Environment Security Controls

To maintain the Trust and Security of a DevSecOps environment, it is essential to have appropriate security controls to achieve the desired assurance level. These controls are used to defend against a variety of threat attack vectors within both the development and production environments and include:

- **Application Control** – Software that ensures only approved programs can be installed and run, reducing the risk of malware or unauthorised applications.
- **Firewalls and Access Control Lists (ACLs)** – Firewalls act as the barrier between all networks and untrusted networks, filtering out unauthorised or potentially harmful traffic. ACLs provide granular control over network resources, specifying which users or system processes are granted access to specific network resources.
- **Micro Segmentation** – Micro segmentation takes the above firewall and ACL controls to the next level by isolating not only the networks but individual workloads and restricts communication paths between them. Doing so further limits the potential for lateral movement by attackers.
- **Web Application Firewalls (WAFs)** – Specialised firewalls that focus on securing web application layers, protecting against attacks such as SQL injection and cross-site scripting. These can add additional protection to applications by acting as a gatekeeper for requests, blocking malicious requests before they reach the services and products behind them.
- **Mutual TLS (mTLS)** – Provides two-way authentication between services. Unlike traditional Transport Layer Security (TLS), which authenticates the server to the client, mTLS also authenticates the client to the server, enhancing trust and data integrity by ensuring that both parties involved are who they say they are.
- **Load Balancing and Scaling** – Distributes incoming network or application traffic across multiple servers, thereby mitigating the risk of any single point of failure and defending against Distributed Denial of Service (DDoS) attacks. This can be combined with automated scaling of services that enables workloads to dynamically respond to even larger attacks or support expected customer peak user workloads.

Additional security controls should also be applied but must be tailored to the specific developer and product user community, balancing security with achieving customer outcomes.

Red Teaming Simulations (Penetration Testing)

Penetration testing is an authorised, simulated cyber-attack on a computer system, network, or application, intended to uncover vulnerabilities, weaknesses, and security gaps. Conducted either by internal teams or third-party experts, the objective of penetration testing is to identify areas of concern before malicious actors can exploit them. By actively finding security flaws and vulnerabilities, the overall security posture is improved and flaws proactively remediated. This provides proactive maintenance of security controls for development and production environments, upholding high levels of trust as the active testing provides validation that existing controls are effective.

Security Information and Event Management

Security Information and Event Management (SIEM) is a comprehensive approach to providing real-time analysis of security alerts generated by various hardware and software components. In a DevSecOps context, SIEM solutions play a crucial role in centralising the collection and interpretation of security-related data, such as logs from firewalls, network devices, and applications. By correlating disparate data points and applying sophisticated analytics, SIEM tools can identify abnormal patterns or potential security incidents that may require immediate attention.

Common Vulnerabilities and Exposures Monitoring

Monitoring Common Vulnerabilities and Exposures (CVEs) is crucial, not only for the software under development, but also for the entire DevSecOps environment including servers, workstations, databases, and networking equipment. Vulnerabilities can exist at multiple layers, and neglecting to continuously monitor CVEs across all components can leave an organisation and customers exposed to security risks. Automated scanning tools assist in timely identification and prioritisation of vulnerabilities, enabling swift mitigation or patching and this CVE Monitoring provides the active monitoring and use of these scans to proactively patch, upgrade, replace and actively manage the risk exposure identified by CVEs.

Patch Management

Given that new vulnerabilities are discovered regularly, maintaining an up-to-date patch management strategy is essential for security resilience. This extends beyond application code to include operating systems, middleware, firmware, and even networking equipment. However, rapid patching should be balanced with testing to ensure that patches don't introduce new issues or conflicts. Moving straight to the latest patch to get the 'latest and greatest' features may also introduce more vulnerabilities or stability issues into the environment and therefore suitable patch management practices should be in place working alongside change management to deploy these patches.

4.52 Environment Management

Secure and trusted computing environments rely on a variety of specialised tools that require ongoing management. The overhead costs associated with environment management are significantly reduced by implementing robust processes and governance protocols. Furthermore, employing automation for tasks such as system updates, security scans, and routine maintenance can also contribute to operational efficiency, thus further minimising costs and risks. By balancing a well-structured management framework with the strategic use of automation, organisations can maintain a high level of security and reliability without overwhelming their systems administration teams. The following covers the full range of environment management actions to proactively rectify issues, perform regular updates, and ensure that performance and capacity needs are met through to planned obsolescence and decommissioning of elements no longer required.

Incident Management

Incident Management focuses on restoring normal service operation as quickly as possible following an unplanned disruption or degradation in service quality. This entails a systematic approach to identifying, classifying, and resolving incidents, often using a ticketing system to track progress and communication. On the other hand, problem management delves deeper to identify the root causes of incidents in an effort to prevent future occurrences. Effective problem management often results in modifications to standard operating procedures, updates to documentation, or even changes in the IT architecture.

Change Management

Change Management ensures any alterations to the system or its environment are introduced in a controlled and coordinated manner. Whether changes are instigated by new feature deployments, system upgrades, or compliance requirements, the process involves a series of steps from planning and approval to testing, implementation, and post-implementation review. The objective is to minimise the risk of negatively impacting the stability and integrity of the existing IT infrastructure while maximising the value of the change. This is achieved through a formalised change approval process, often involving a Change Advisory Board to assesses the risks and benefits associated of the proposed change. Rigorous documentation and communication are also key elements, as they keep stakeholders informed and provide a record of alterations for future reference or audits. By managing changes in a structured way, organisations can align their IT capabilities with business objectives more effectively, mitigate risks of service disruption, and foster a culture of continuous improvement, and if required roll-back to previous versions without significant disruption to the user community of environment.

Secrets Management

Secrets Management focuses on safeguarding sensitive information such as API keys, credentials, and tokens. This data is crucial for system operations but poses a significant security risk if compromised. Traditional methods of storing secrets in configuration files or environment variables are increasingly seen as inadequate, given the complexity of modern architectures. Secrets management tools and services provide a more secure and centralised way to store, access, and manage secrets. These solutions often include features such as encryption-at-rest, access control policies, audit trails, and automated rotation of secrets to minimise the risk of exposure or misuse.

Disaster Recovery

Disaster Recovery is designed to ensure the quick restoration of essential functions in the aftermath of significant system failures, data corruption, malware attacks such as ransomware, or other catastrophic events.

An integral component of this is backup management, which involves regularly creating, storing, and testing copies of critical data and system configurations in geographically dispersed locations. These backups serve as a lifeline for data restoration in cases where corruption or loss occurs. To mitigate the risks of data corruption, whether accidental or intentional, it is crucial to configure systems to automatically identify inconsistencies. Upon detection, administrators can then leverage the backup service to rectify the corruption.

In many disaster recovery strategies, having a secondary physical site for data storage and system operations is another key element. The secondary location, often known as a 'hot,' 'warm,' or 'cold' site depending on its state of readiness, serves as a backup operation environment and can be switched to in the event of a disaster and provide business continuity, although often still in a degraded capacity. Additionally, a comprehensive disaster recovery plan should be in place, outlining the specific steps to be taken during and after a disaster. This plan usually details the roles and responsibilities of key personnel, as well as the procedures for switching to the secondary site, notifying stakeholders, and restoring normal operations. The plan should be routinely tested and updated to account for new risks and evolving infrastructure or customer needs.

Network Management

Network Management is crucial for maintaining a stable, secure, and efficient IT infrastructure used during development and often in highly networked or network accessible products. Using a variety of tools and methodologies, Network Management provides fault detection, performance monitoring, and configuration management. These activities not only ensure optimal network performance but also enable pre-emptive issue resolution supporting incident and problem management activities. By aligning these network management practices with ITIL guidelines, organisations can better meet service level agreements (SLAs) and governance standards. This integrated approach creates a network environment that is agile and robust, adapting to both daily operational needs and long-term projects, making it a vital component of secure and trusted environment.

ICT Infrastructure Management

Eventually, an organisation may grow to a size where manually provisioning and tracking its IT footprint becomes unfeasible and effective ICT Infrastructure Management and automation is required. ICT Infrastructure Management involves tracking, managing, and optimising organisational ICT assets throughout their lifecycle. This can range from tangible assets (e.g. servers and networking hardware), workstations and real estate, to intangible assets such as virtual machines, containers, databases, software licenses and IP. The primary goal of ICT Infrastructure Management is to ensure that ICT assets are efficiently used and maintained, thereby providing maximum value. Effective asset management not only reduces operational costs but aids in risk mitigation, especially when it comes to compliance with legal and regulatory requirements. Tools such as Configuration Management Databases and Enterprise Asset Management systems should be leveraged to automate many aspects of this function, providing a real-time view into asset status and performance.

Virtual Machines, Physical Machines or Containers

The choice between Virtual Machines (VMs), physical machines and running containers (using orchestration tools like Kubernetes) is contingent on a range of factors including workload characteristics, scale, management complexity, and application architecture.

VMs offer strong isolation and flexibility, as each instance runs its own operating system, making them ideal for monolithic applications or those with specific OS-level requirements. However, VMs typically consume more resources and have longer boot times, which are a concern in environments that require rapid scaling. On the other hand, using Kubernetes for orchestrating containerised applications offers benefits such as resource efficiency, quick start-up times, and seamless scaling. However, Kubernetes comes with a steep learning curve, its own set of complexities and introduces additional overhead. Using bare metal, physical machines, to run workloads, enables software to use the full computational power of these machines but compromises the agility and flexibility as software is locked down to run only on specific hardware. Additionally, to cut down on software licensing fees, the use of physical machines can limit costs especially when software is licensed based on central processing unit (CPU) cores. The hardware used to run VMs or containers typically have significant CPU capacity, increasing the costs. The choice between the three options should be based on specific needs, the scale of operation, and the expertise available as part of the initial and ongoing infrastructure management activities.

Capacity and Performance Management

Capacity and Performance Management focusses on ensuring that IT resources are optimised to meet both current and future organisational needs. This involves continuously monitoring system performance, processing and utilisation trends, analysing data, and making informed decisions about hardware and software scaling. By aligning capacity planning with performance metrics, organisations can not only fulfill their current requirements but also proactively prepare for future demand. Effective management in these areas maintains high availability and optimal system performance, while also providing cost efficiencies and mitigating the risks of system failures or slowdowns.

Facilities Management

Facility Management takes on a multi-faceted role extending beyond physical infrastructure oversight. It ensures that the physical spaces housing critical IT assets, such as data centres and network rooms, are not only efficient but also secure. Physical security measures, such as access controls, surveillance systems and alarms, are vital components of this approach. In addition, Facility Management often involves achieving and maintaining various industry accreditations related to security and operational standards depending on the sensitivity and classification of the data.

Vendor Management

Vendor Management involves the selection of vendors based on cost and capability, but also their adherence to security standards and compliance requirements. This ensures that all third-party components, software, and services meet organisational and regulatory security benchmarks. To fortify the supply chain, organisations may require vendors to possess specific security accreditations or undergo regular audits. Contracts often include clauses that specify security expectations, incident response protocols, and penalties for non-compliance. By tightly integrating vendor management, organisations can effectively mitigate risks, ensuring that all elements of the supply chain uphold the same security and quality standards, thus strengthening the overall Trusted DevSecOps Factory.

Decommissioning

Decommissioning is the planned phase-out or retirement of a system, application, or infrastructure component. This process involves multiple steps, such as data migration, asset sanitisation, configuration reversals, and access revocations, often requiring coordination across various teams including Development, Operations, and Security. Decommissioning is not simply turning off a switch; detailed planning and execution are needed to ensure that no data is lost or exposed and that resources are optimally reallocated. Failing to correctly decommission systems can lead to problems such as orphaned resources consuming budget, unnoticed vulnerabilities exposing risks, and residual data left open to be compromised and 'die', resulting in increased overheads and at worst, threat attack vectors compromising the Trusted DevSecOps Factory and products. Decommissioning also requires the development of specialised documentation and procedures for compliance.

5 Conclusion

DevSecOps is more than a set of tools—it is a complete solution framework for an organisational culture shift. DevSecOps redefines and reshapes existing software development and operational processes by embedding security as a fundamental component at every stage of the development cycle, building trust with every pipeline stage and ongoing production operations. With ever-increasing cyber threats in the digital landscape, it is with a matter of urgency that organisations make this shift to not only protect their assets, but to ensure faster, more reliable, and more secure software applications for customers.

As highlighted in this whitepaper, many organisations contend with building a complete DevSecOps solution on their own due to a lack of expertise and resources. Fortunately, implementing DevSecOps is not an all-or-nothing venture. Organisations can start small, and gradually integrate security checks into their existing DevOps pipeline, thereby cultivating a culture of shared responsibility for security across all teams over time.

DevSecOps enhances security, improves compliance, and reduces risk, strengthening the overall security posture for software applications. Consunet are leaders in implementing DevSecOps and can assist organisations in this transition by identifying how to uplift or enhance existing DevOps processes through maturity reviews, specialised consulting services, or delivery of a complete Trusted DevSecOps Factory tailored to developer and customer needs. Every step taken to integrate security into existing DevOps processes aligns organisations with the critical, global purpose of securing customer and end user information in the uncertain and evolving digital domain.

consunet®

To build the cyber and spectrum security future

Head Office
44 Waymouth Street
Adelaide SA 5000

Postal Address
GPO Box 449
Adelaide SA 5001

Contact
+61 8 8234 8819
contact@consunet.com.au

www.consunet.com.au