Department of Education

DevSecOps

A culture of workflow cyber security

Classification: OFFICIAL

Queensland Government

Contents

Executive summary

In today's digital landscape, software security is not just a necessity but a cornerstone for safeguarding critical systems and sensitive data against malicious threats. It plays a vital role in maintaining the integrity, availability, and confidentiality of information, which are crucial for the trust and safety of individuals and organisations alike.

Embracing a proactive approach, DevSecOps (Development, Security, and Operations) stands out as a pivotal strategy for the swift delivery of secure, high-quality digital services. This approach, *which integrates security practices from the very beginning of the development and operations lifecycle*, is instrumental in facilitating early threat detection. Such early integration significantly reduces risks and costs associated with cybersecurity threats. Moreover, the inherent automation within *DevSecOps accelerates development processes* while ensuring robust security through continuous monitoring.

This document encompasses a thorough market scan, comparing various solutions to tailor DevSecOps to the department's specific needs. The aim is to craft a practical and cost-effective approach to cybersecurity, adapting to the unique challenges and requirements faced in today's rapidly evolving cyber landscape.

Introduction to DevSecOps

In the dynamic world of software development, the intersection of speed, innovation, and security is where DevSecOps finds its significance. DevSecOps, an abbreviation for Development, Security, and Operations, represents a paradigm shift in how organisations approach software development and cybersecurity.

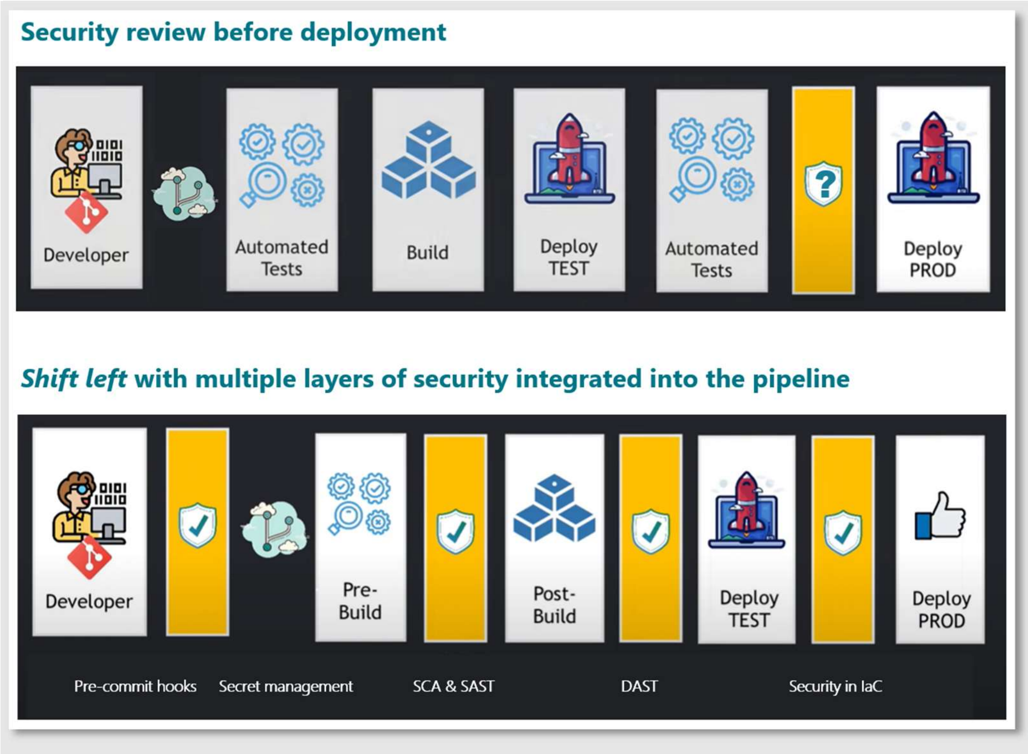
Traditionally, *security has often been a siloed* aspect of the development process, *typically addressed at the end of the software development lifecycle* (SDLC). This approach, while structured, tends to delay product releases and may overlook evolving security threats that arise during the development process. DevSecOps disrupts this model by embedding security at every stage of the SDLC, starting from initial design to deployment and operations.

This integration is not merely about tooling; *it's a cultural shift*. DevSecOps fosters a 'Security as Code' culture, where security is treated as an integral part of the development process, not as an afterthought. It encourages continuous collaboration between development, security, and operations teams. This collaborative approach ensures that security considerations are not just tacked on but are an intrinsic part of the development process.

One of the key tenets of DevSecOps is the 'shift left' principle. It emphasises the need to address security early in the development cycle. By shifting left, security testing and compliance monitoring happen in tandem with coding, significantly reducing the likelihood of security issues in the released product.

The benefits of DevSecOps are multifaceted. It enables organisations to rapidly deploy secure and compliant software, reduces the cost and time involved in addressing security issues, and enhances overall software quality. Perhaps most critically, it helps in managing and mitigating the risks in a landscape where cybersecurity threats are increasingly sophisticated and pervasive.

In summary, DevSecOps represents a strategic and necessary evolution in the approach to software development. By integrating security practices into every phase of the SDLC, it allows organisations to *balance the need for speed and innovation in software development with the imperative of maintaining robust cybersecurity measures*.



[1]

[1] Screenshots stitched from “What is DevSecOps?” by TechWorld with Nana

Current state analysis

The current security landscape in software development is marked by rapidly evolving risks and technologies. This dynamic environment demands a security approach that is both robust and proactive. Our existing security framework, however, shows several shortcomings:

- **High-level architectural reviews**: These abstract reviews often miss crucial details. They lack depth in analysing the inner workings of software, which can lead to undetected vulnerabilities lurking beneath the surface.
- **Use of open-source libraries**: While open-source libraries nested within closed-source components offer numerous benefits, they also pose significant security risks. These risks are often overlooked, and ironically, attempts to secure these libraries by 'freezing' versions can lead to further vulnerabilities due to lack of continual patching.
- **Peer code reviews**: Although peer reviews are essential for collaboration and knowledge sharing, they frequently fall short in identifying complex security issues. This is often due to a lack of specialised security expertise among peers.
- **Annual penetration tests**: These often expensive and time-consuming tests do not cover the entire spectrum of code vulnerabilities. In a fast-paced development world, a year is a long time, and software can become exposed to new risks between these annual checks. Moreover, many of the existing security tools were developed before the emergence of current technological trends.

Given these challenges, there is an urgent need to upgrade the tools and methodologies used by developers. This upgrade is essential for ensuring the safety and security of software in an environment where threats are continuously evolving and increasing in complexity.

DevSecOps vs traditional cyber security

Aspect	Traditional cybersecurity	DevSecOps
Integration timing	Security integrated towards the end	Early integration from the outset of the project
Automation	Often relies on manual testing and checks	High emphasis on automation of security testing and monitoring
Collaboration	Siloed teams with segregated duties	Strong collaboration among Dev, Sec, and Ops teams
Culture	Security is often seen as a separate task	Fosters a culture of 'Security as Code' and shared responsibility
Feedback loop	May lack continuous feedback mechanisms	Continuous feedback loop for rapid response and improvement
Policy enforcement	Manual policy and compliance enforcement	Policy as Code and Compliance as Code, automated enforcement
Development speed	May slow down due to late-stage security checks	Accelerated with security integrated in the CI/CD pipeline
Resilience & recovery	May have longer recovery times	Improved resilience and rapid recovery from incidents
Cost efficiency	Higher costs due to late-stage remediation and manual checks	Potential cost savings due to early detection and automation
Continuous monitoring	Periodic monitoring and assessments	Continuous monitoring and real-time security analytics

Keys for an holistic DevSecOps approach

1. **Security in infrastructure as code:** Ensuring secure infrastructure configurations from the start.
2. **Secret management:** Securely manage and store secrets and credentials.
3. **Pre-commit hooks:** Enforce code quality and security standards before code is committed to the repository.
4. **Source composition analysis:** Scan and manage open-source dependencies for vulnerabilities.
5. **Static application security testing (SAST):** Analyse the source code for security vulnerabilities.
6. **Dynamic application security testing (DAST):** Test applications in runtime for security issues.
7. **Web application firewall (WAF):** An additional layer of protection for web applications by filtering malicious traffic.
8. **Container security:** Scan container images for vulnerabilities.
9. **Continuous integration/continuous deployment (CI/CD) security:** Integration of security checks within the CI/CD pipeline.
10. **Threat modelling:** Practices and tools for identifying and addressing security threats in the design phase.
11. **Security orchestration, automation, and response (SOAR):** Automating incident response and remediation.
12. **Security information and event management (SIEM):** Real-time monitoring and threat detection.
13. **Identity and access management (IAM):** Managing user access and permissions securely.
14. **Runtime application self-protection (RASP):** Protection against runtime attacks.
15. **Secure code review tools:** In-depth code review and vulnerability detection.

Strategic pivot to DevSecOps

Transitioning to a DevSecOps model requires a strategic pivot that encompasses not just the adoption of new tools and technologies but also a shift in organisational culture and processes. This strategic pivot should be executed through several key steps:

Assessment and Planning: Begin by conducting a thorough assessment of the current security practices and infrastructure. This should include identifying the gaps in the existing system and understanding the specific needs of the organisation. Based on this assessment, develop a detailed plan that outlines the objectives, timeline, and key milestones for the transition to DevSecOps.

Cultural Shift and Training: DevSecOps demands a cultural shift where security is integrated into every aspect of development and operations. This involves training and sensitising the development, operations, and security teams about the importance of security in the development lifecycle. It's crucial to promote a culture of collaboration and shared responsibility for security.

Integration of Security into the SDLC: Security practices should be 'shifted left', meaning they are introduced earlier in the software development lifecycle. This includes integrating security tools and practices into the Continuous Integration/Continuous Deployment (CI/CD) pipelines and ensuring that security checks are a part of the regular development process.

Automation and Continuous Monitoring: Implement automation tools for security testing, monitoring, and compliance. Automation is key in DevSecOps to ensure continuous security without slowing down the development process. Continuous monitoring of the infrastructure and applications for any security threats should be established.

Feedback Loops and Continuous Improvement: Establish feedback mechanisms for continuous learning and improvement. Security is an evolving field, and the DevSecOps approach should be flexible enough to adapt to new threats and technologies. Regular reviews and updates to the security practices should be an integral part of the process.

Stakeholder Engagement and Communication: Engage all stakeholders, including management, development teams, and IT staff, in the transition process. Effective communication is vital to ensure everyone understands their role in the new DevSecOps environment.

Measuring Success and Adjusting Strategies: Define metrics to measure the success of the DevSecOps initiatives. These metrics could include the number of security incidents, the time taken to identify and fix vulnerabilities, and the impact on the software delivery timelines. Regularly review these metrics and adjust strategies as necessary.

This strategic pivot to DevSecOps is not just a one-time project but an ongoing journey towards a more secure, efficient, and resilient software development process. By taking these steps, organisations can ensure a smooth and successful transition to a DevSecOps model, resulting in faster, safer, and more reliable software delivery.

DevSecOps implementation options

Each of the below options offers a different level of integration and coverage, and the choice depends on the department’s current commitment on needs, resources, and readiness for change. The Minimal Impact Option is suited to start with basic DevSecOps practices, the Balanced Option seeks a more thorough approach without extensive disruption, and the Comprehensive Option aims at a complete DevSecOps transformation.

The options are presented as a pathway to assist in a prolonged or phased adoption approach.

Minimal impact option

Objective: To integrate essential DevSecOps practices with minimal disruption to existing workflows.

Components:

Static Application Security Testing (SAST): Using Lint, GitHub, SonarQube or similar tools for source code analysis.

Dynamic Application Security Testing (DAST): Implementing automated testing during runtime.

Security in Infrastructure as Code: Ensuring secure configurations from the start.

Secret Management: Using tools like Azure Key Vault for secure management of secrets and credentials.

Benefits: Offers a basic level of security integration without major changes to current processes.

Challenges: May not cover all aspects of DevSecOps, leading to potential gaps in security.

Balanced option

Objective: A more integrated approach that balances security, efficiency, and impact on existing systems.

Components:

> *All components from the Minimal Impact Option.*
>
> Software Supply Chain Management: Managing and securing the software supply chain.
>
> Continuous Integration/Continuous Deployment (CI/CD) Security: Integration of security checks within CI/CD pipelines.
>
> Incident Response: Implementing effective incident response mechanisms.

Benefits: Provides a comprehensive security approach while maintaining operational efficiency.

Challenges: Requires more resources and may have a moderate impact on current workflows.

Comprehensive Option

Objective: Full-scale implementation for thorough integration of DevSecOps.

Components:

All components from the Balanced Option.

Container Security: Ensuring the security of containerised applications.

Identity and Access Management (IAM): Managing user access and permissions securely.

Security Information and Event Management (SIEM): Real-time monitoring and threat detection.

Security Orchestration Automation and Response (SOAR): Automating incident response and remediation.

Web Application Firewall (WAF): Additional layer of protection for web applications.

Threat Intelligence: Using advanced threat intelligence tools.

Secure Code Review Tools: In-depth code review and vulnerability detection.

Benefits: Achieves a high level of security and compliance, covering all aspects of DevSecOps.

Challenges: Could be resource-intensive and may require significant changes to existing processes.

Appendix

Available tools and services

The table below shows the combination of tools that would provide us with minimal, balanced and comprehensive cover.

Note: The column indicating the presence of an existing service or tool relies is based on a review of Content Manager and is only as current as those records.

	Minimal cover	Balanced cover	Comprehensive cover	Existing in DoE	Provider/tool
Static application security testing (SAST)	✓	✓	✓	✓	SonarQube*
Source composition analysis	✓	✓	✓	No	-
Dynamic application security testing (DAST)	✓	✓	✓	✓	SonarQube*
Security in Infrastructure as Code	✓	✓	✓	✓	SonarQube*
Software supply chain management	✓	✓	✓	No	-
Secret management	✓	✓	✓	✓	Azure DevOps & Azure Key Vault
Continuous integration/continuous deployment (CI/CD) security		✓	✓	✓	SonarQube

	Minimal cover	Balanced cover	Comprehensive cover	Existing in DoE	Provider/tool
Incident response		✓	✓	No	-
Container security			✓	✓	SonarQube*
Identity and Access Management (IAM)			✓	✓	Azure AD
Security information and event management (SIEM)			✓	✓	Intalock
Security orchestration, automation, and response (SOAR)			✓	No	-
Web application firewall (WAF)			✓	✓	CITEC
Threat intelligence			✓	✓	Threat Intelligence Pty Ltd
Secure code review tools			✓	No	-

* We have access to these services with SonarQube but there is a very low adoption rate across the department

Market scan and analysis

The following tables comprises both established leaders in Static Application Security Testing (SAST) products, as recognised by Gartner's Magic Quadrant, and tools already in use or provided by software vendors we currently have a relationship with.

	CheckMarx SAST	GitHub Advanced Security	Snyk Code	SonarQube	Sonatype	Synopsys	Veracode
Existing DoE relationship	No	Yes	No	Yes	No	No	No
Founding/ownership	Founded 2006	Founded 2008 and owned by Microsoft since 2018	Founded in 2015	SonarQube is owned by SonarSource, which was founded in 2008	Founded in 2008	Founded in 1986	Founded in 2006
Location/s	• Private cloud in Atlanta, Georgia, US	• Boston, Massachusetts, US • Tel Aviv, Israel • Ottawa, Canada • Zurich, Austria • London, England	•	•	•	•	•
Integration	• DevOps • GitHub • IDE	• DevOps • GitHub • IDE	• DevOps • GitHub • IDE	• DevOps • GitHub • IDE	• DevOps • GitHub	• DevOps • GitHub	• DevOps • GitHub
Cost	From $59,000/year	≈ $80/month/active committers	≈ $85/month/active committers	Per line of code subscription Current subscription for 5 million lines ($60,714.80)	$675 per team member	$675 per team member	?

The table below captures the various features included with each product.

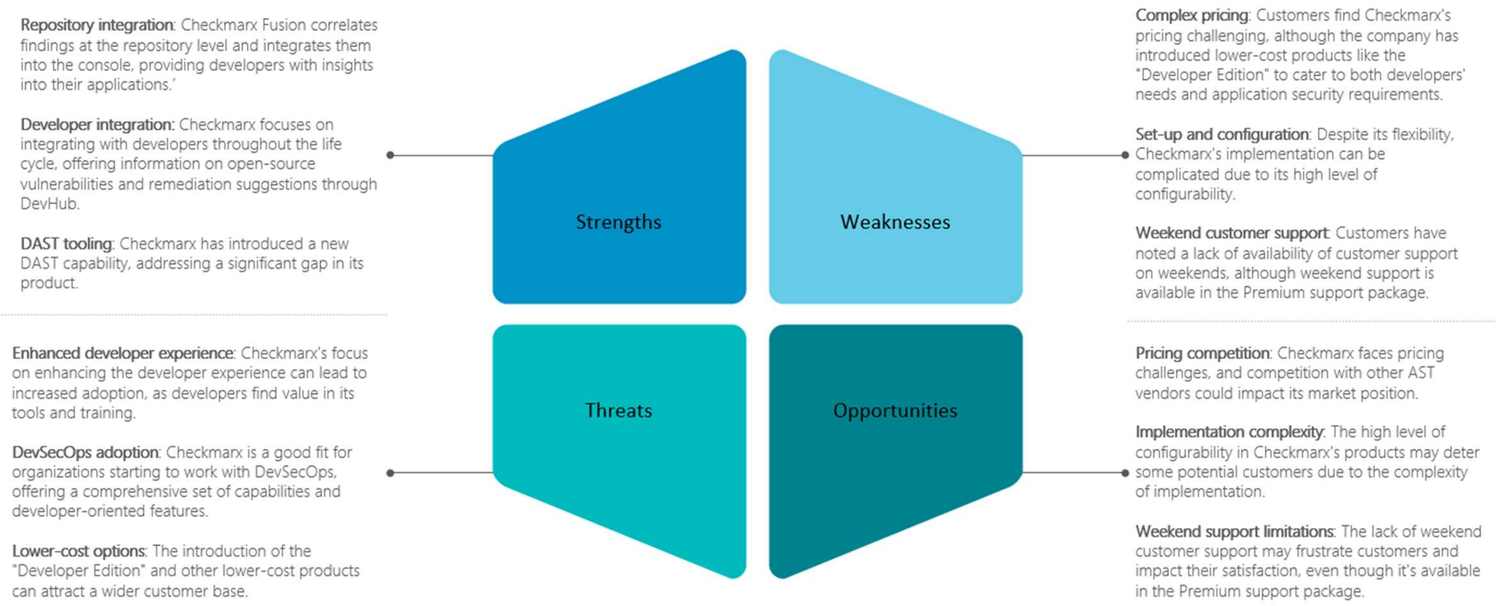
	Feature	CheckMarx SAST	GitHub Advanced Security	Snyk Code	SonarQube	Sonatype	Synopsys	Veracode
Objective measures	**Code scanning**	Yes	Yes	Partial	Yes	Yes	Yes	Yes
	Customisation							
	Automation							
	Language support	Various	Various	Various	Various	Various	Various	Various
	Static analysis	Yes	Yes	No	Yes	Yes	Yes	Yes
	Dynamic analysis	Yes	Yes	Partial	Yes	Yes	Yes	Yes
	Software composition analysis	Yes	Yes	Yes	No	Yes	Yes	No
	Dependency scanning	Yes	Yes	Yes	No	Yes	Yes	No
	Container scanning	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Licence compliance	No	Yes	Yes	No	Yes	Yes	Partial
	Code remediation	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Vulnerability detection							
	Continuous monitoring							
	CI/CD integration	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	IDE integration	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	False positive reduction							
	Third party integration							
	Reporting & analytics	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Scalability							
	Mobile support							
	Pricing model	Commercial	Commercial	Freemium / Commercial	Open-source / Commercial	Commercial	Commercial	Commercial
Subjective measures	**Gartner Magic Quadrant**	Leader	Challenger	Leader	[Not included] Peerspot #1	Niche player Forester Wave leader	Leader (#1)	Leader
	Support & documentation							
	Ease of use							
	Community & user feedback							

Product scan using the Gartner Magic Quadrant

The Gartner Magic Quadrant is a research methodology and graphical representation that provides a snapshot of a market's direction, maturity, and participants. It evaluates companies based on their completeness of vision and ability to execute. The quadrant has four categories: Leaders, Challengers, Visionaries, and Niche Players. Companies are positioned in one of these based on their performance, helping businesses assess the competitive landscape and make informed decisions when choosing vendors or solutions.

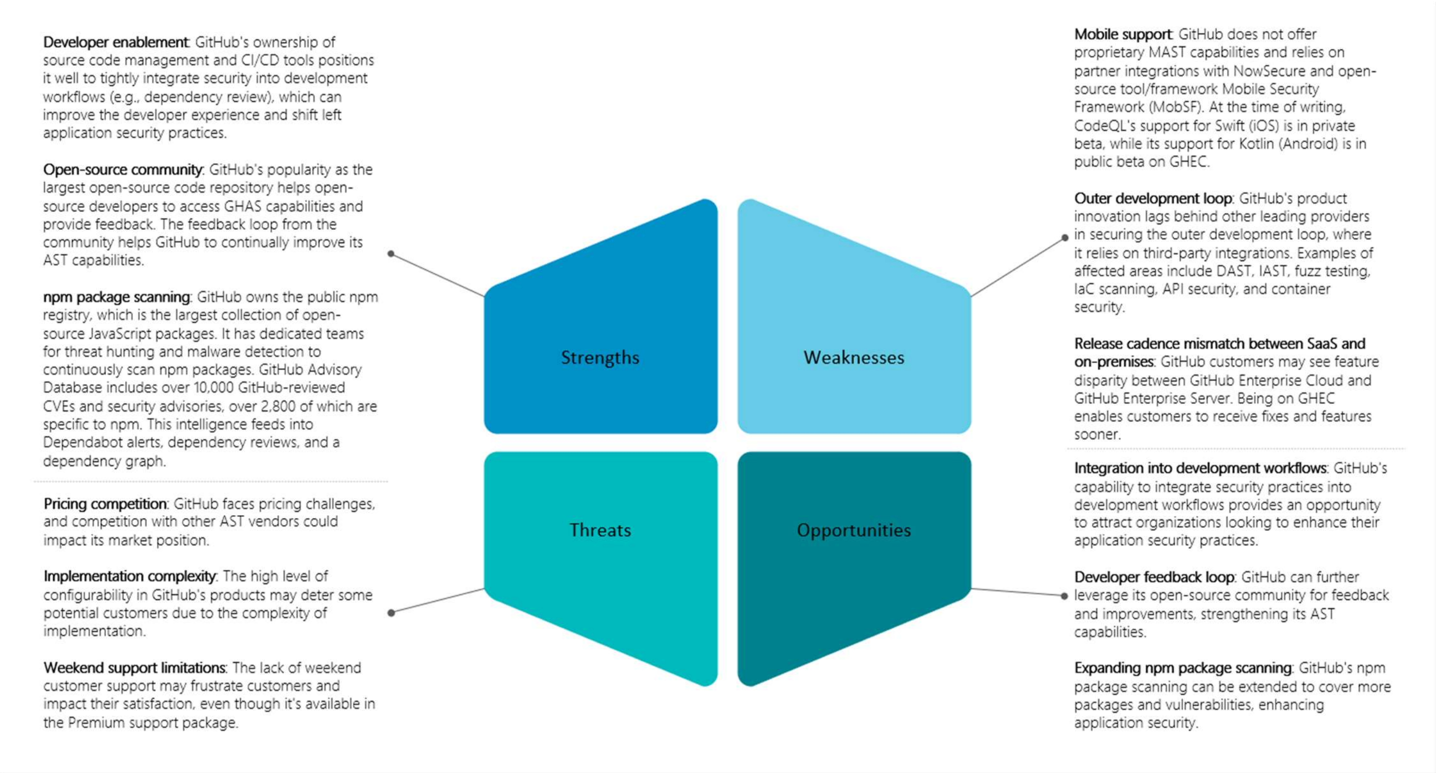
Checkmarx

Checkmarx Advanced Security is a leading cybersecurity company that specialises in providing comprehensive application security solutions, offering static application security testing (SAST), dynamic application security testing (DAST), and software composition analysis (SCA) to help organisations identify and mitigate vulnerabilities in their software applications throughout the development lifecycle.



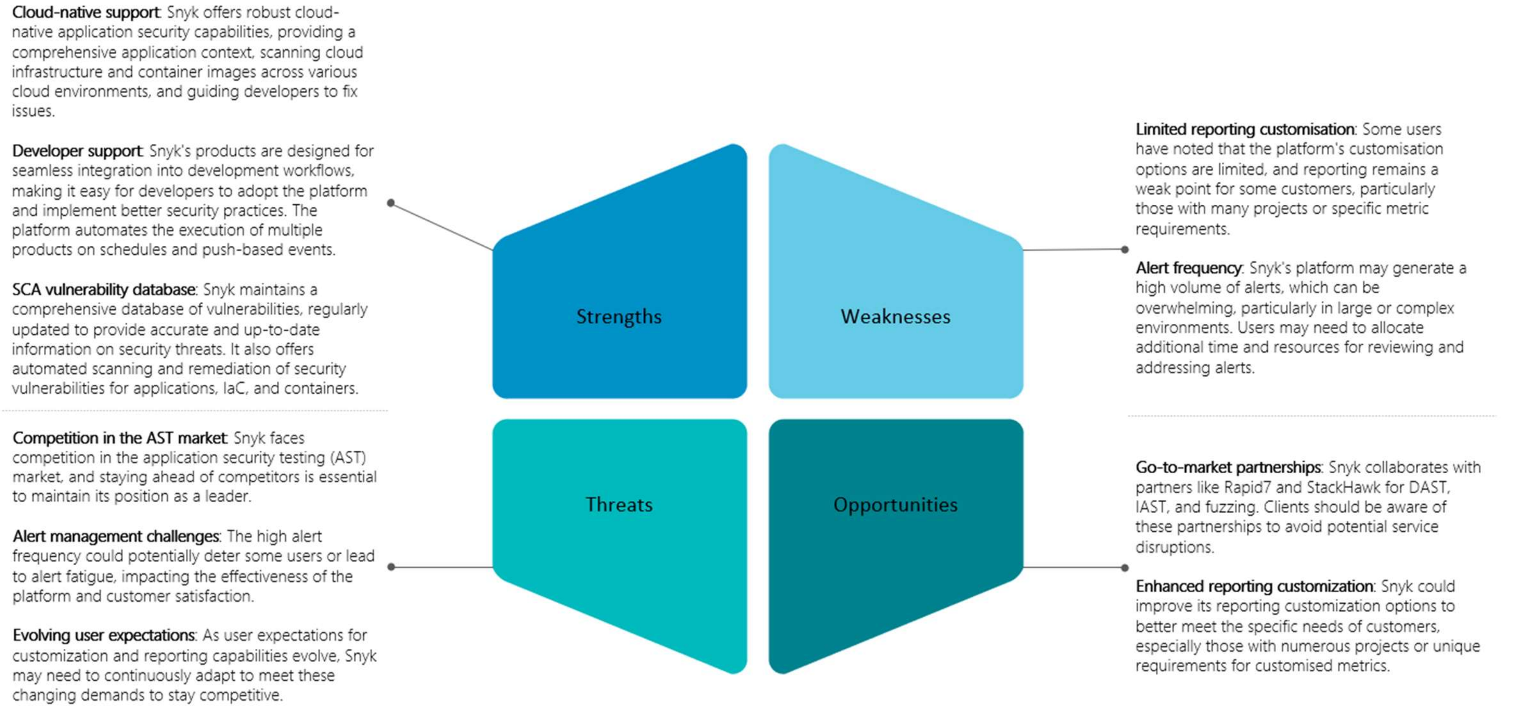
GitHub Advanced Security

GitHub Advanced Security is a suite of security features integrated into GitHub to help developers and organisations identify and remediate security vulnerabilities in their code, with a focus on providing code scanning, secret scanning, and dependency scanning to enhance the security of software development workflows.



Snyk Code

Snyk's paid plans provide enhanced capabilities for identifying and fixing vulnerabilities in open-source dependencies and container images, catering to organistions heavily reliant on third-party code.



SonarQube

Paid editions of SonarQube offer advanced features for code quality, maintainability, reliability, and security analysis, making it suitable for organisations looking to improve software quality.

A note on the adoption of SonarQube

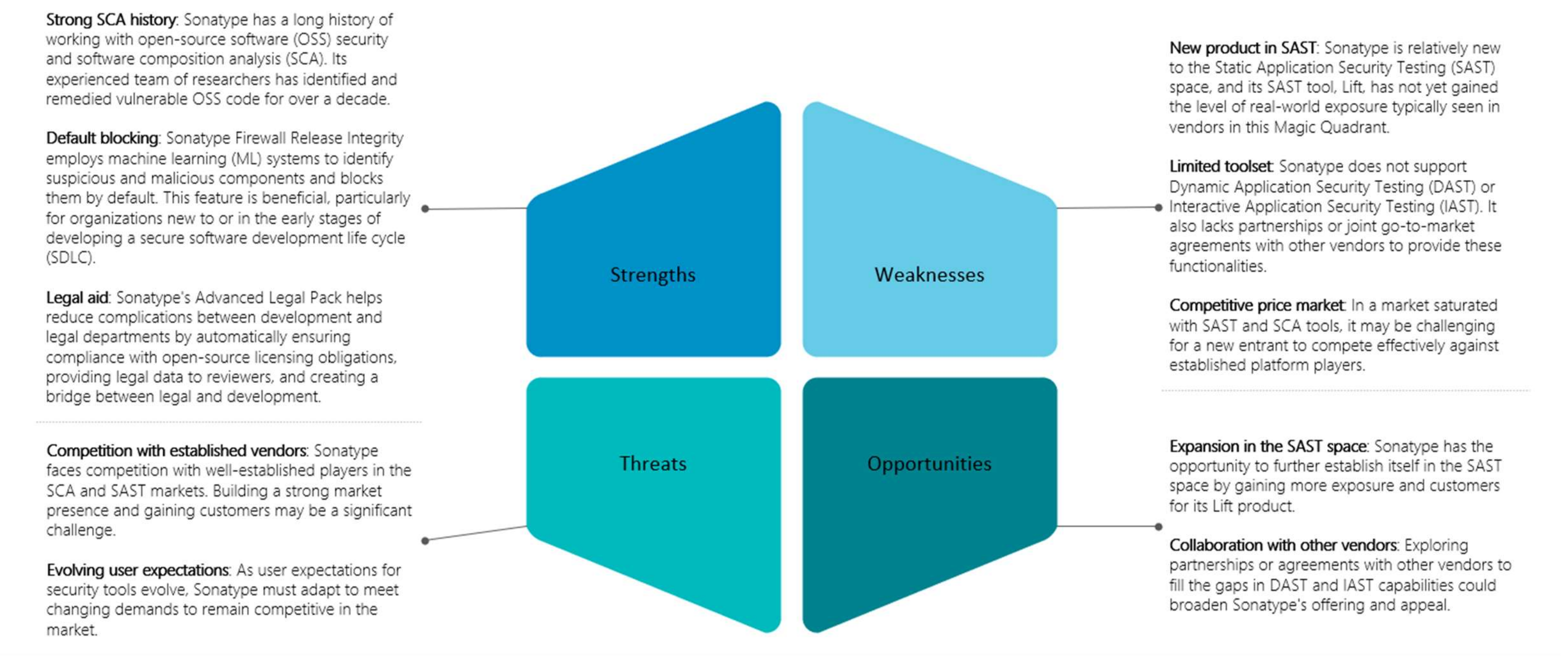
The Department of Education's software development journey began with an early embrace of SonarCloud, driven by the goal of enhancing code quality and security. Subsequently, a transition to SonarQube was necessitated by legislative requirements. To streamline operations and reduce maintenance efforts, we opted for the Azure App Service platform. Although a SonarQube Docker image exists, organisational infrastructure limitations currently preclude its use.

Management of deployments is efficiently handled through Azure DevOps pipelines, minimising the need for user intervention. This approach has generally been low-maintenance. However, the recent introduction of automated smoke testing, while intended to bolster application quality, has introduced overhead due to the inherent test instability, possibly stemming from application changes.

Unfortunately, the adoption rate of these tools remains low, which has impeded the realisation of expected benefits. Research indicates the potential for substantial advantages, but the absence of widespread adoption and support presents a challenge in achieving full certainty. In light of these circumstances, a comprehensive review is warranted, potentially leading to the exploration of alternative tools with greater organisational backing.

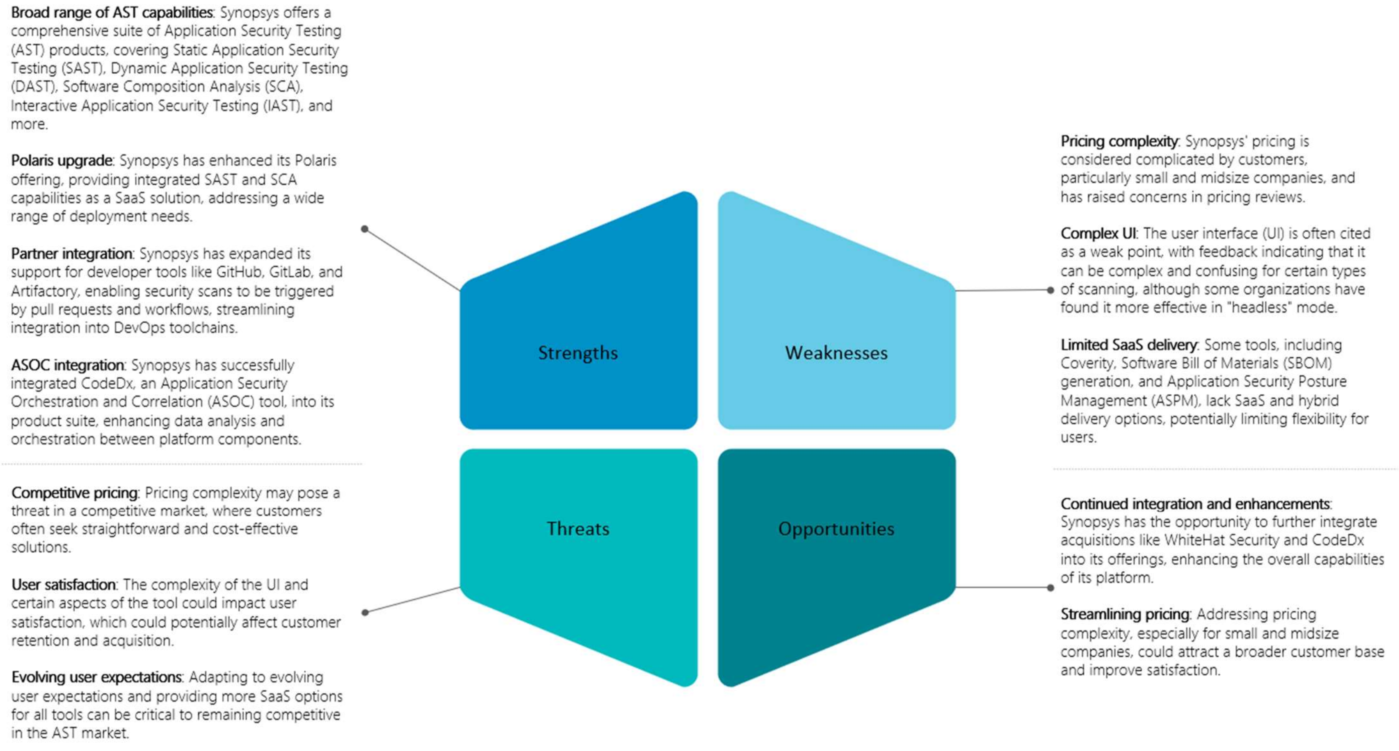
Sonatype

Sonatype is a software company specialising in DevOps and open-source governance solutions, focusing on helping organisations manage and secure their software supply chains.



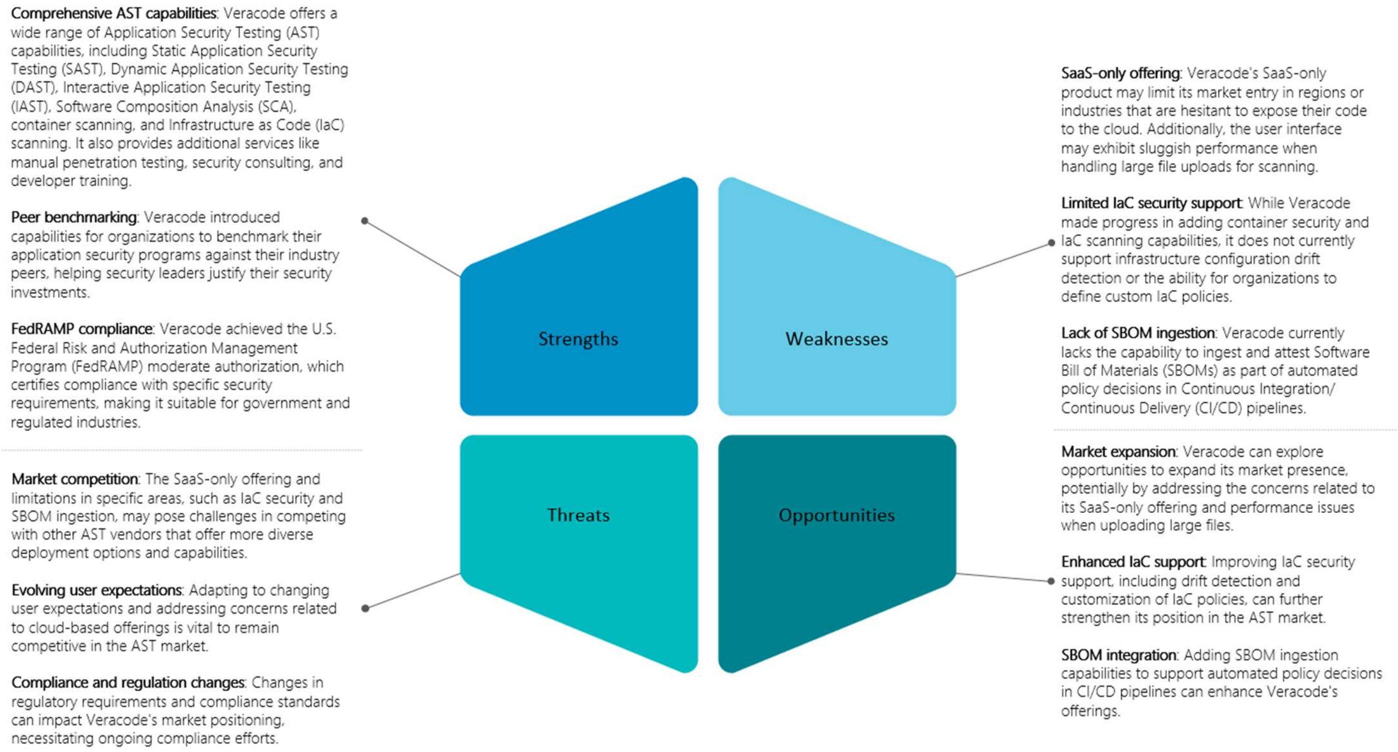
Synopsys

Synopsys offers paid versions of its software security tools, providing an extensive suite of solutions, including static analysis, dynamic analysis, and software composition analysis, suitable for large enterprises requiring a comprehensive approach to security.



Veracode

Veracode's paid offerings include a comprehensive suite of application security testing (AST) tools, covering static analysis, dynamic analysis, and software composition analysis, making it ideal for organisations with diverse security needs.



Glossary of terms

Key terms	Definition
API testing	The process of verifying that an application's programming interfaces (APIs) function correctly, securely, and efficiently, ensuring they adhere to specified standards and requirements.
Application security posture management (ASPM)	A comprehensive cybersecurity practice that involves assessing, monitoring, and improving the security posture of an organisation's applications to protect against vulnerabilities and threats, with the goal of enhancing overall security and compliance.
AST (Application security testing)	The practice of evaluating software applications to identify and address potential security vulnerabilities and weaknesses, ensuring that the applications are robust and protected against security threats and breaches.
AST, Dynamic (DAST)	A method of assessing the security of a web application by evaluating it in its running state to identify vulnerabilities and weaknesses, often by simulating real-world attacks from the outside, without access to the source code.
AST, Interactive (IAST)	A testing method that assesses the security of an application by continuously monitoring and analysing the application's behaviour during runtime to identify vulnerabilities and security issues. IAST combines elements of both Static AST (SAST) and Dynamic AST (DAST) to provide real-time insights into an application's security.
AST, Mobile (MAST)	The process of assessing the security of mobile applications, such as those on smartphones and tablets, to identify vulnerabilities and potential

	security risks, ensuring the protection of sensitive data and user privacy.
AST, Static (SAST)	a security testing technique that involves analysing the source code, bytecode, or binary code of a software application to identify vulnerabilities, security flaws, and potential weaknesses before the application is executed, helping to ensure the security and integrity of the software.
Container security	Safeguarding the integrity, confidentiality, and availability of applications and their associated components that are deployed within containers, to prevent vulnerabilities, unauthorised access, and data breaches.
Fuzzing	A testing technique used to discover vulnerabilities and flaws in software, especially in applications or systems that handle data inputs.
Integrated development environment (IDE) integration	The capability of a software tool or platform to seamlessly work within or alongside an IDE, allowing developers to enhance their coding and development processes, access additional features or services, and streamline their workflow directly from their development environment.
Intrastructure-as-code (IaC)	A methodology that involves managing and provisioning infrastructure, such as servers, networks, and databases, through code and automation tools, streamlining the deployment and management of IT resources within software development and operations.
Software composition analysis (SCA)	A practice that involves examining and assessing the software components and third-party libraries used in a software application to identify any vulnerabilities, licensing issues, or security risks,

	helping organisations maintain the security and compliance of their software products.
Software supply chain security (SSCS)	Practices and measures aimed at ensuring the security and integrity of the software supply chain, which includes the development, distribution, and deployment of software. This involves protecting against potential vulnerabilities, threats, or attacks at various stages of the software supply chain to maintain the overall security of software applications.