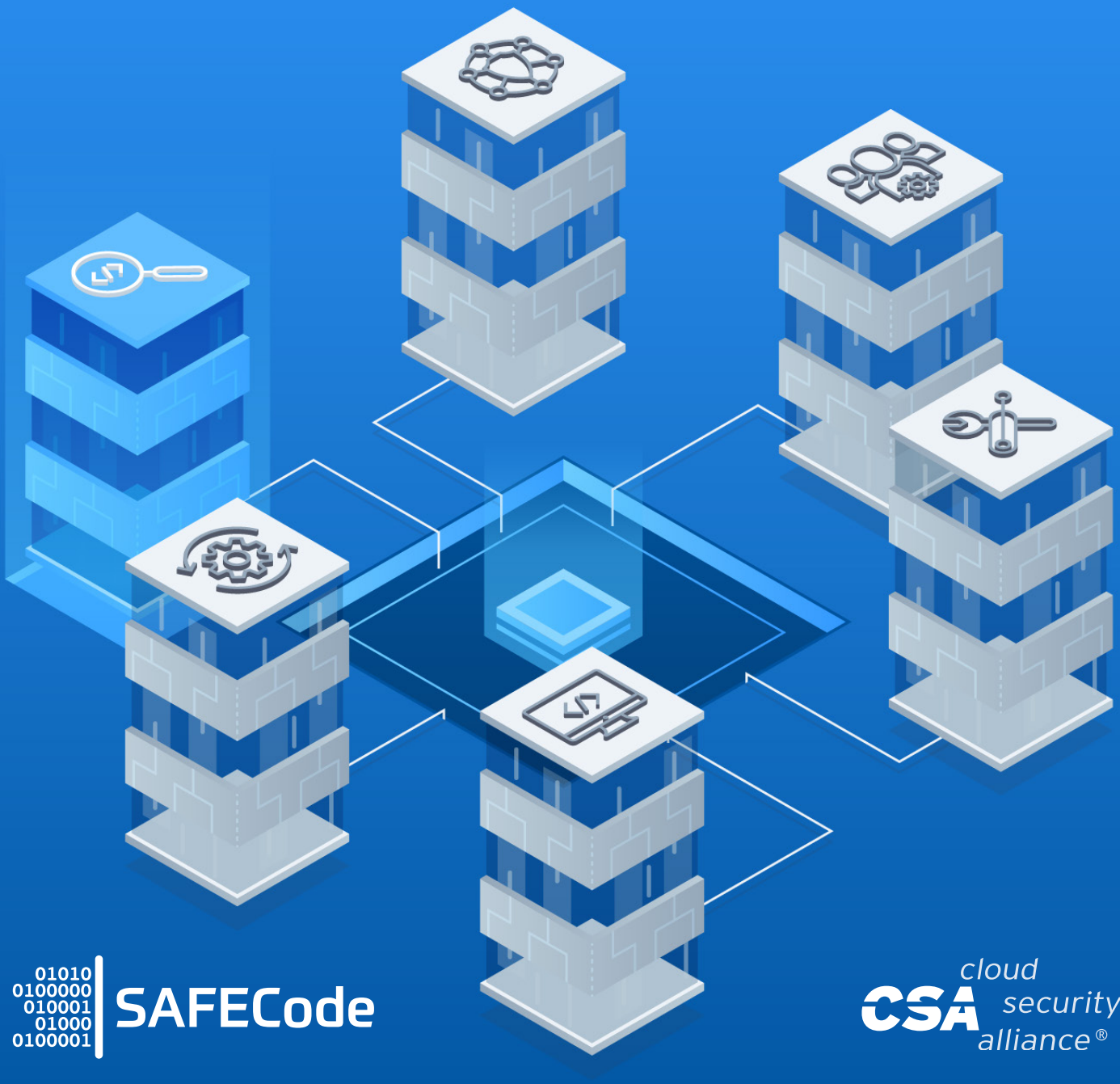
The Six Pillars of DevSecOps:

Measure, Monitor, Report, and Action



01010
0100000
010001
01000
0100001
SAFECode

CSA cloud security alliance®

The permanent and official location for the DevSecOps Working Group is
https://cloudsecurityalliance.org/research/working-groups/devsecops/

Acknowledgments

Lead Authors

Dan Gora
Roupe Sahans

Contributors

Alex Brown
Daniel Parkin
Michael Roza

Reviewers

Roland M
Julianna Tchebotareva
Timothy Thatcher
Udith W

CSA Analysts

Josh Buker

CSA Global Staff

Claire Lehnert

Table of Contents

Foreword

The Cloud Security Alliance and SAFECode are both deeply committed to improving software security outcomes. The paper Six Pillars of DevSecOps, published in August 2019, provides a high-level set of methods and successfully implemented solutions its authors use to build software at speed and with minimal security-related bugs. Those six pillars are:

Pillar 1: Collective Responsibility (Published 02/20/2020)
Pillar 2: Collaboration and Integration (Published 02/21/2024)
Pillar 3: Pragmatic Implementation (Published 12/14/2022)
Pillar 4: Bridging Compliance and Development (Published 02/08/2022)
Pillar 5: Automation (Published 07/06/2020)
Pillar 6: Measure, Monitor, Report and Action

The successful solutions that underpin each of these pillars are the subjects of a much more detailed set of joint publications by the Cloud Security Alliance and SAFECode. This paper is the final of the six follow-on publications.

Introduction

The implementation and maintenance of DevSecOps initiatives can take anywhere from a few months to years to implement. The continuous measurement of DevSecOps success and failure is the key differentiator when considering the wholesale change in people, processes, and tooling. The saying "you can't manage what you don't measure" has never been more true. Without actionable metrics and observability to measure performance, progress cannot be understood, success cannot be replicated, and failures cannot be recognized.

The ability to measure, monitor, report, and action are essential features of any successful security program to support effective decision-making. In this paper, we explore:

- **Making data observable**: Turning security data and metrics into observable data
- **Scenario-based review**: The concept of security observability applied to high and low-performing teams
- **Improvement through reporting**: Where to start on your security observability reporting journey

Goals

The Cloud Security Alliance's DevSecOps working group (WG) issued high-level guidance in "The Six Pillars of DevSecOps"[1] that advocated for a new approach to security. The six pillars are considered the critical focus areas for any organization looking to implement DevSecOps, with one of the pillars being "Pillar 6: Measure, Monitor, Report and Action".

The goal of Pillar 6 is to promote and demonstrate clear measurements for security performance in DevSecOps. This will enable security and digital leaders to determine the effectiveness of their security practices and how well security has been interwoven into the software development lifecycle.

Audience

The target audience of this document includes those involved in the management and operational functions of information security and information technology. This consists of the CISO, CIO, CTO, and the individuals involved in the following functional areas: platform engineering, DevOps, product teams, architecture, information security, governance, and compliance.

Making Data Observable

A system is observable if it emits useful data about its state, which is crucial for determining root cause[2]. Observability measures a system's current state based on the data it generates, such as logs, metrics, and traces. Observability aims to understand what's happening across all environments so issues can be detected and resolved to keep systems efficient and reliable. Organizations adopt observability to help detect and analyze the significance of events to their operations, software development lifecycles, application security, and end-user experiences[3].

The measurements of logs, metrics, distributed traces, and user experiences are the key pillars to achieving observability success[3]:

- **Logs**: Structured or unstructured text records of discrete events that occurred at a specific time
- **Metrics**: The values represented as counts or measures that are often calculated or aggregated over a period of time. Metrics can originate from various sources, including infrastructure, hosts, services, cloud platforms, and external sources
- **Distributed tracing**: Activity of a transaction or request as it flows through applications and shows how services connect, including code-level details
- **User experience**: Extends traditional observability telemetry by adding the outside-in user perspective of a specific digital experience on an application, even in pre-production environments

1 https://cloudsecurityalliance.org/artifacts/information-security-management-through-reflexive-security/
2 Observability vs. monitoring: What's the difference? | Dynatrace news
3 What is observability? Not just logs, metrics, and traces | Dynatrace news

Observability for Vulnerabilities

Identifying vulnerabilities becomes simple once product teams have mechanisms to scan with the right tools and workflows; for example, Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA). The challenge begins when teams scan, remediate, and report continuously.

Over time, the vulnerability backlog can increase — in some cases to the tens of thousands — until it becomes too difficult to manage and remediate. An effective vulnerability management program should detect and remediate vulnerabilities early in the development lifecycle to reduce the overall risk exposure to its platform/product. Observability for vulnerabilities requires you to identify the three following attributes.

1. Mean Time to Identify (MTTI)

> MTTI is the time taken to identify a vulnerability. A high MTTI indicates that vulnerabilities are detected later in the development lifecycle and thus will increase the cost of remediation and the risk of vulnerabilities being introduced into production. By identifying vulnerabilities early, developers can prevent exploitable vulnerabilities from being introduced to production environments.
>
> In Figure 2, the time to identify is the start date at the exposure window (10th January) and the date to identify said vulnerability (16th January); hence, the time taken to identify is 7 days. Across hundreds of vulnerabilities, the time to identify will vary; MTTI is the average of all your vulnerability identification values.

2. Mean Time to Remediate (MTTR)

> MTTR indicates the average time spent remediating a vulnerability after identification. MTTR helps to track if vulnerabilities are being identified and then "forgotten" by resolver teams. Like MTTI, MTTR can be tracked against severity and the volume of risk-accepted vulnerabilities.
>
> In Figure 2, the time to remediate is the difference in date of identification (16th January) and date of remediation (10th February); hence, the time taken to remediate is 26 days. Like MTTI, across hundreds of vulnerabilities, the time to remediate will differ; MTTR is the average of all your vulnerability remediation values.

3. Trending vulnerabilities against development velocity

> Is development velocity outpacing the ability to remediate vulnerabilities? Tracking this trend helps to identify if the development team is struggling to keep up with the pace of vulnerability remediation during development and operations and if the remediation

effort is keeping pace with the development velocity. MTTI/MTTR can be tracked against development velocity and severity scores. This can help identify if there is a disconnect between the development teams and the security teams in vulnerability management.

In Figure 2, the vulnerability exposure window (10th January - 10th February) spanned across two release windows. This value will help provide contextual information to help justify MTTI and MTTR values.

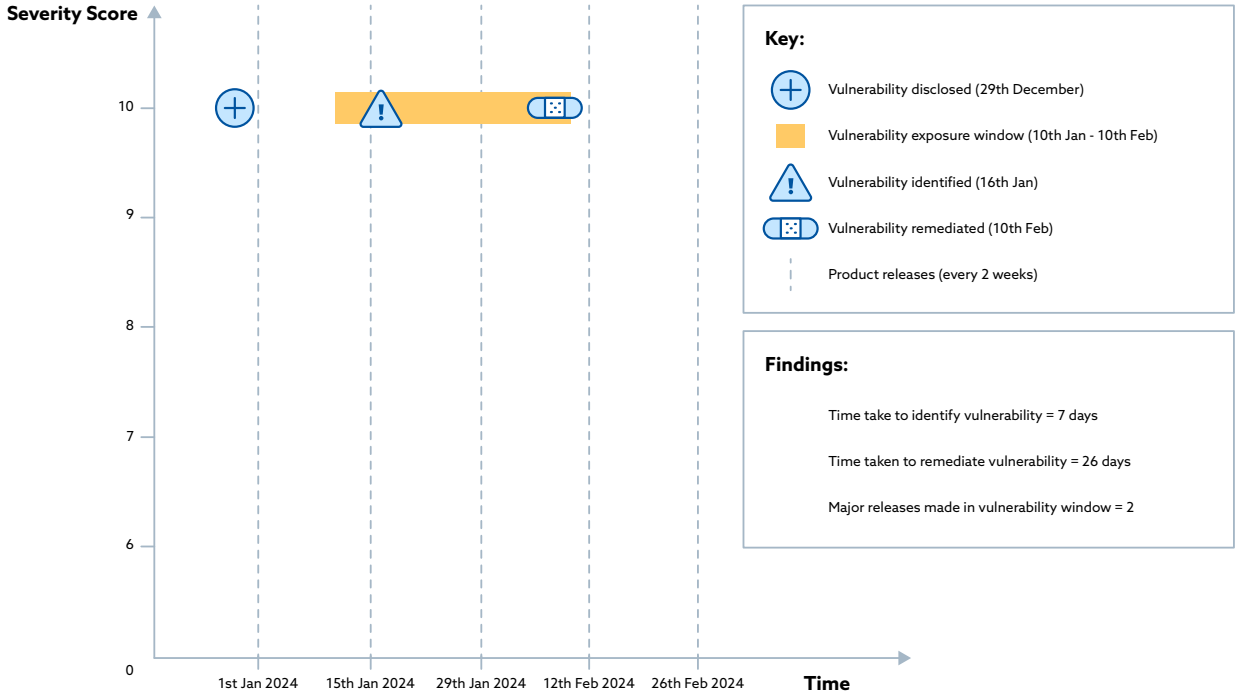


Figure 2: Metrics for vulnerabilities

Observability for Security Architecture

The following observability factors are designed to educate development teams on the acceptability of threat modeling exercises and their remediation activities. Organizations can employ these points to improve their threat modeling patterns and ceremonies.

4. Threat volume

Tracks the volume of threats identified from a threat modeling exercise, the type of threats, and corresponding controls. This helps to understand if a defense-in-depth approach is being adopted.

Figure 3 demonstrates against STRIDE (Spoofing, Tampering, Repudiation, Information

Disclosure, Denial of Service, Elevation of Privilege) the volume of controls applied to each threat theme. The defense-in-depth average is the control coverage average per STRIDE threat type. Information Disclosure in this example can be observed to be the most prevalent threat as it has the highest volume of controls mapped.

5. Control volume

Tracks each respective control against threats identified from a threat modeling exercise. This helps to understand if remediation activities are optimized to scale. For example, a single control has multiple uses.

Figure 4 demonstrates an example set of controls typically identified from a threat model and the usage of each control to address a threat. This measures the return on investment (ROI) for each control. In this example, Role-Based Access Control (RBAC) and security monitoring address the most threats and have the highest ROI, indicating which controls should be prioritized. Each control averages to address 5 to 6 threat use cases.

6. Guardrail volume

The volume of controls converted to guardrails in response to identified vulnerabilities/threats. This helps to understand if the environment is being secured to prevent a risk, threat, or vulnerability from occurring.

Figure 5 indicates the implementation of guardrails over time against controls, stressing the importance of preventative measures against threat use cases.

7. Threat model burndown

Measuring control implementation against threats is a key outcome of threat modeling. By tracking what has been identified against what is being implemented, teams can assess the effectiveness of threat modeling—verifying that value is being added to the development process.

Figure 5 represents a threat model burndown with 25 unaddressed threats reduced to 10 over 5 sprints (10 weeks) by control implementation. This type of measurement will qualify the success of a threat model exercise; in this example, 60% of controls implemented over 10 weeks represent good practice.

8. Security pattern adoption

Serves as a metric to evaluate the maturity of a security architecture by analyzing the implementation of security design patterns. A security pattern is a reusable solution that addresses common security challenges and vulnerabilities in software or system design; this

can form as a working document or a piece of reusable code (template).

Typically, a more mature organization will have more patterns and templates to ensure consistent design/engineering practices are followed. Well-scaled patterns can aid in reproducibility and consistency, reducing the number of recurring weaknesses and vulnerabilities being raised. A lack of up-take could indicate that teams are unaware of their existence or the pattern is not effective. In Figure 6, pattern/template volume is 20, and the adoption rate is 70%, indicating inefficiencies exist in 6 patterns.

9. Threat scenarios mapped to cyber risks

Identifies threats that inform risk statements - i.e., how many threats map to a risk statement. Unlike risks, threats cannot be mitigated; however, threats can be addressed and used to inform risks.

Figure 6 demonstrates this metric. In the example, the risk of sensitive data exposure maps to 14 threats; therefore, in theory, addressing the 14 threats should reduce the risk score presented to sensitive data exposure.

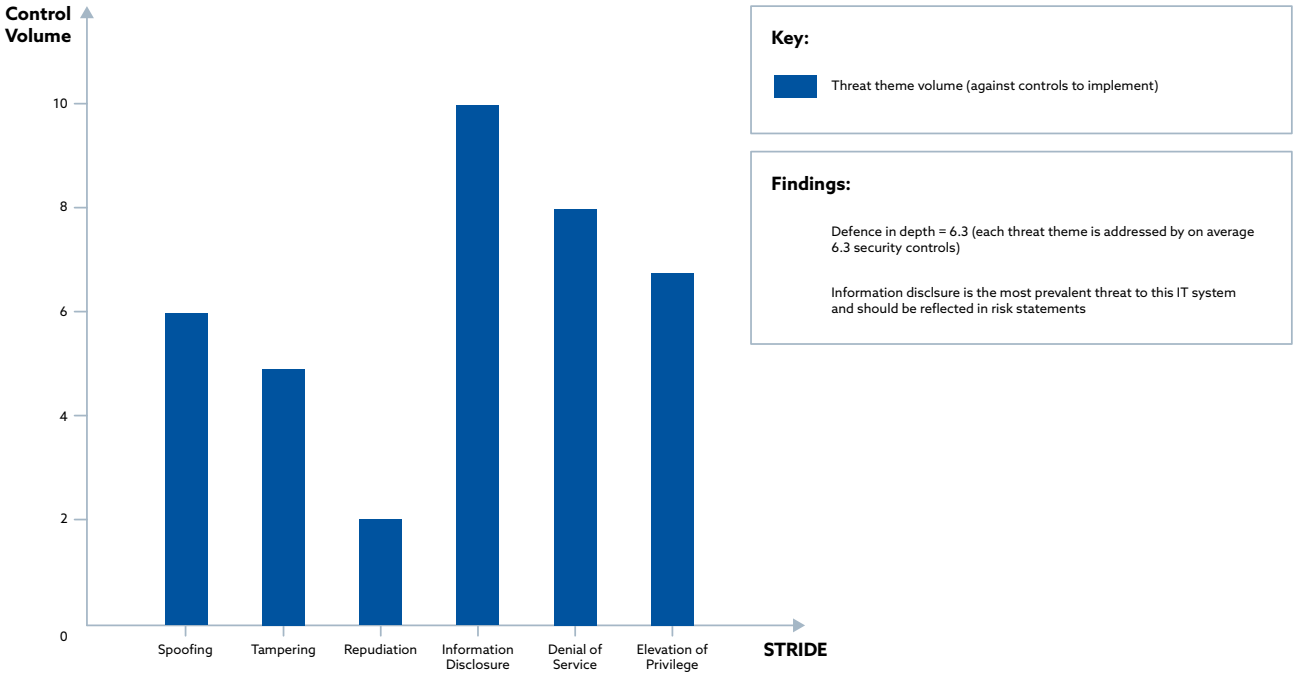


Figure 3: Metrics for control implementation across threat themes

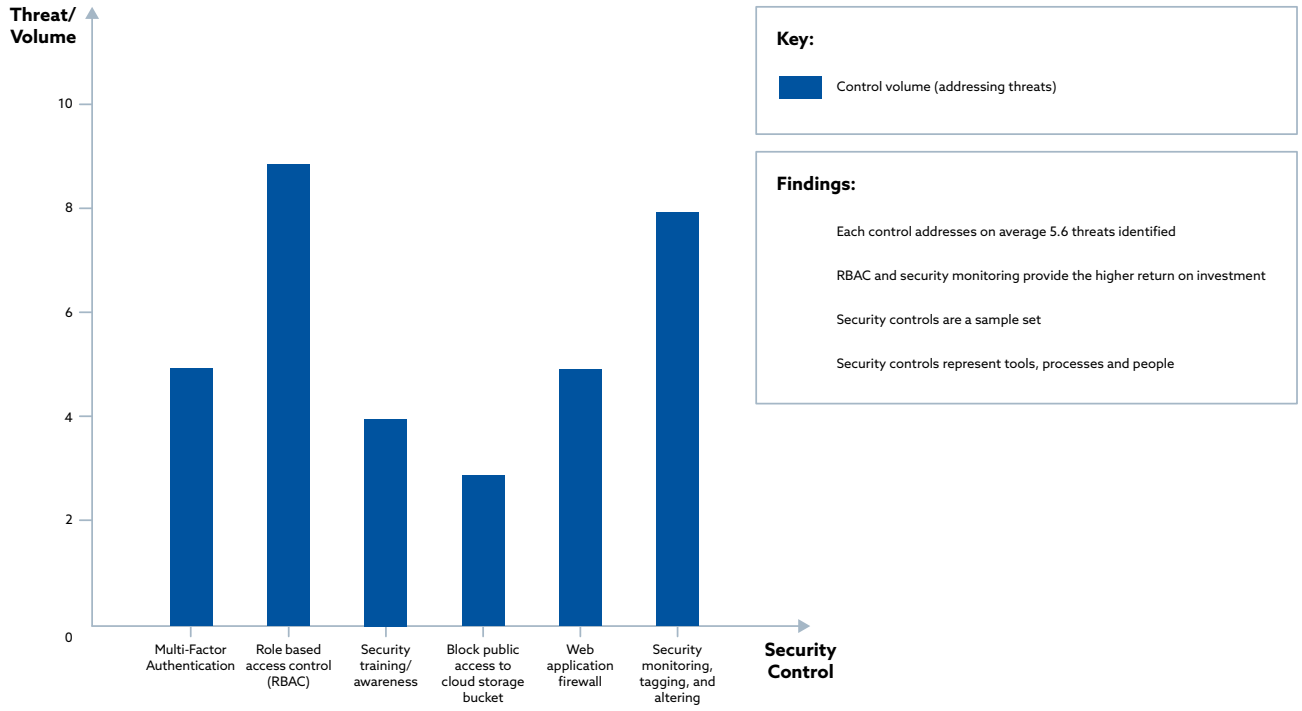


Figure 4: Metrics for security control reuse to address threats

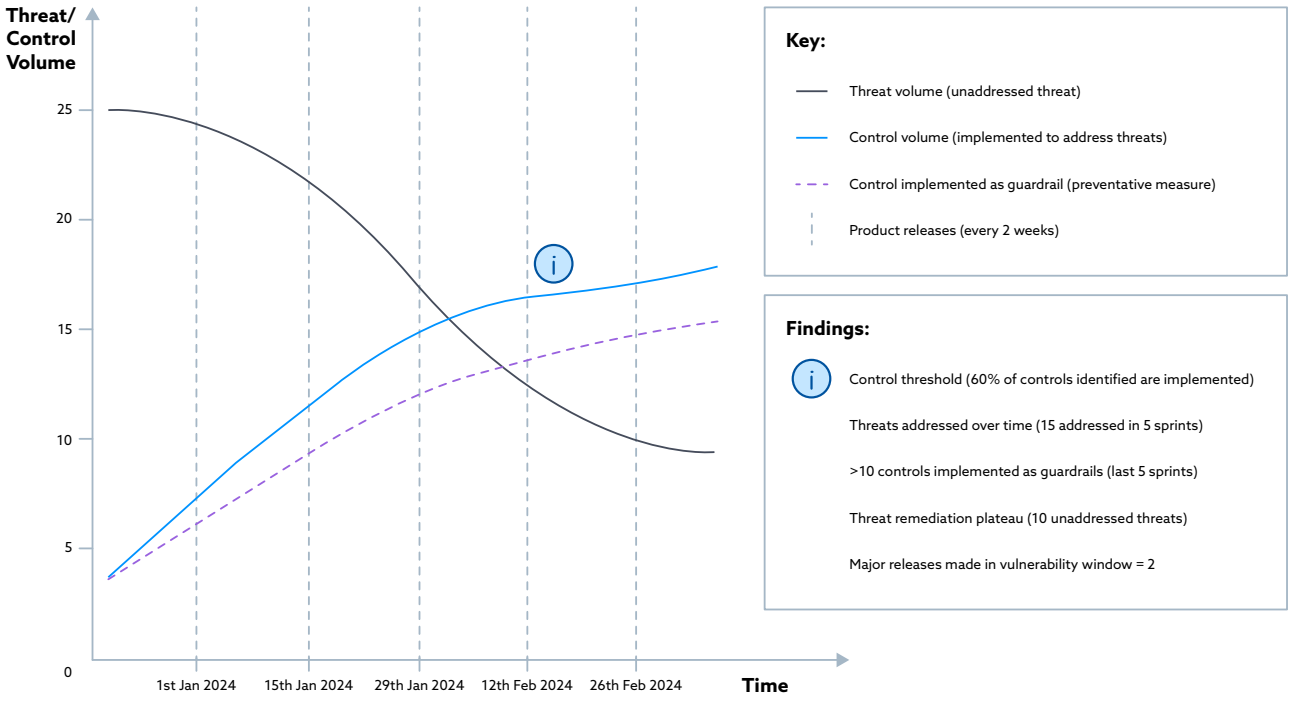


Figure 5: Metrics for control implementation against threats

Other Metrics:

A total of 31/34 threats map to 3 risk statements leading to an average of 11.3 for risk/threat mapping. 3 threats are immediately deprioritised as they don't attribute to a business risks.

Risk 1: 'risk of sensitive data exposure' is referenced in 14 threat statements

Risk 2: 'compromised data integrity' is referenced in 6 threat statements

Risk 3: 'Critical IT service unavailable' is referenced in 11 threat statements

Pattern/template volume is 20 with an adoption rate of 70% across the project. Indicating 6 patterns/templates need modifying or removing

Figure 6: Metrics mapping risks to threats and pattern adoption

Observability for Incident Response

Observability for Incident Response provides insight on alerting and detection and, in turn, leads to more detailed and informative post-incident reviews.

Mean time to detect (MTTD), mean time to contain (MTTC), and mean time to restore normality (MTTRN) are key metrics that allow organizations to act quickly when a failure occurs due to a security weakness. Security monitoring of the IT estate would allow organizations to capture these metrics, giving operational teams access to necessary data during an incident.

10. Mean time to detect (MTTD)

> MTTD is the time a team takes to discover a potential security incident. Organizations with a high MTTD introduce additional unwanted risk by not detecting incidents early. Early detection (identification) means teams can focus on response and recovery activities in a timely manner, and reduce the impact on production.
>
> In Figure 7, the time to detect (or identify) is the start date at the exposure window (31st December) and the date to determine said incident (1st January), hence the time taken to identify is two days. Across hundreds of incidents, the time to identify/detect will differ - MTTD is the average of all your incident detection values.

11. Mean time to contain (MTTC)

MTTC tracks the time it takes to contain a security breach/incident. This helps to understand how quickly the team can respond to security incidents. Containment is when a team can begin to limit and reduce the impact of an incident; avoiding impact on dependent services or lateral movement of a breach.

In Figure 7, the time to contain is from the detection date (1st January) to the containment date (6th January), hence the time taken to contain is 6 days. Across hundreds of incidents, the time to contain will differ - MTTC is the average of all your incident containment values.

12. Mean time to restore normality (MTTRN)

MTTRN indicates the average time spent to restore normality during incident remediation. A slower MTTRN can result in service disruptions, impacting an organization's ability to function or a product's ability to effectively work. In some cases, a lengthy MTTRN represents negligence when recovering service with sensitive data.

In Figure 7 the time to restore normality starts within the date to detect (1st January) and ends with the date to remediate said incident (25th January), hence the time taken to restore normality is 25 days. Across many incidents, the time to restore to normality will differ; MTTRN is the average of all your restoration values.

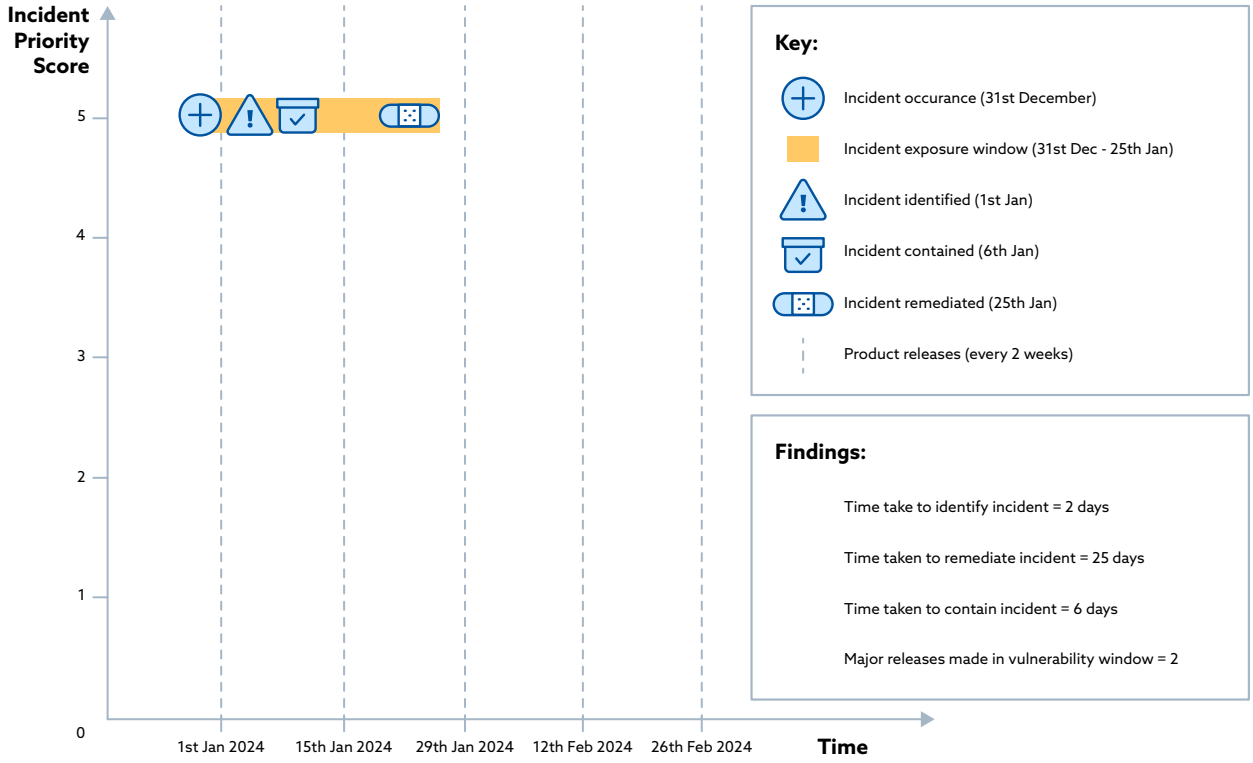


Figure 7: Metrics for incident response

Maturity Comparison Between Teams

Understanding the impact of DevSecOps practices on a team's performance requires a comprehensive approach to measuring, monitoring, reporting, and actioning metrics.

In order to provide an overview of the maturity and performance of different practices, illustrative scenarios of three teams operating at different maturity levels in different organizations and their impact on metrics are described. The scenario comparison uses the OWASP DevSecOps maturity model (DSOMM). Teams are labeled Alpha, Beta, and Charlie, with low, moderate, and high maturity and performance levels, respectively. The scenario comparison provides insights into the practical application and their impact on DevSecOps practices. The focus is not to label teams as 'good' or 'bad' but to illuminate the varying impacts of these practices on observable metrics and the overall maturity and performance.

Team "Alpha" - Low Maturity and Performance

The Alpha team operates at a low level of maturity in DevSecOps practices. This team operates between levels 1 and 2 of the DSOMM, possessing a basic understanding and adoption of relevant DevSecOps practices. Team Alpha is still in the early phases of adapting DevSecOps. The team has a slow response to remediate exploitable vulnerabilities and has an inconsistent approach to ensuring security is built in design. Delayed incident handling highlights the need for better security practices and training.

Team "Beta" - Moderate Maturity and Performance

The Beta team operates between DSOMM levels 2 and 3, showcasing moderate maturity in their DevSecOps practices. Although they've progressed beyond the foundational stages of DevSecOps, their approach to handling vulnerabilities can be described as moderately reactive rather than proactive. The commitment to embedding security during the design phase exhibits a level of inconsistency, occasionally allowing potential threats to be overlooked. Moreover, while their incident response is notably faster than less mature teams in the field, there still exists room for improvement.

Team "Charlie" - High Maturity and Performance

The Charlie team operates at a high level of maturity and performance, scaling on DSOMM levels 3 and 4 for DevSecOps practices. The metrics demonstrate advanced DevSecOps maturity. The practices and metrics of the Charlie team reflect not just a high level of technical proficiency but also a culture where security is prioritized and integrated at every step. This results in security being consistently embedded across all phases, reducing the attack surface and improving security posture.

While incident response is efficient and timely, team Charlie maintains a continuous improvement mindset. This ensures that the incident response can adapt to evolving security threats.

Security Observability Between Three Teams

Through comparative scenarios, the influence of maturity in performance becomes apparent. The scenarios underscore the importance of continuous measurement which forms the basis for informed decision-making, successful replication of practices, and recognition of areas for improvement. They ultimately serve as a guide for organizations to enhance their DevSecOps practices and culture.

Vulnerability observability across teams

Figure 8 proves the concept for averaging and scoring MTTI and MTTR using days elapsed, demonstrating improvement over time. **Figure 9** represents the comparison over time across teams, indicating high and low performance.

The Alpha team's low maturity practices result in a high number of observable vulnerabilities, while the Beta team's moderate maturity practices result in a reduced number. The Charlie team, with its high maturity practices, has the fewest observable exploitable vulnerabilities.

MTTI: The Alpha team takes an average of 20 days to identify a new vulnerability after its introduction. Due to established regular weekly vulnerability scans, the Beta team reduced this time to 15 days to identify new vulnerabilities. However, Beta are scanning a production codebase and are having to wait for the respective vulnerabilities to surface to production. The Charlie team is able to detect vulnerabilities with a low MTTI within a few days due to continuous integration and continuous deployment (CI/CD) practices that leverage automated security scans of non-production source code repositories.

MTTR: Once identified, vulnerabilities often remain unresolved for Alpha, hence the increase in MTTR over time. Beta has established a mitigation process, the MTTR reduced and plateaued at 15 days due to challenges in promptly addressing identified vulnerabilities. Charlie has reduced its MTTR continuously over time due to their effective remediation processes; developers dedicate time to vulnerability remediation.

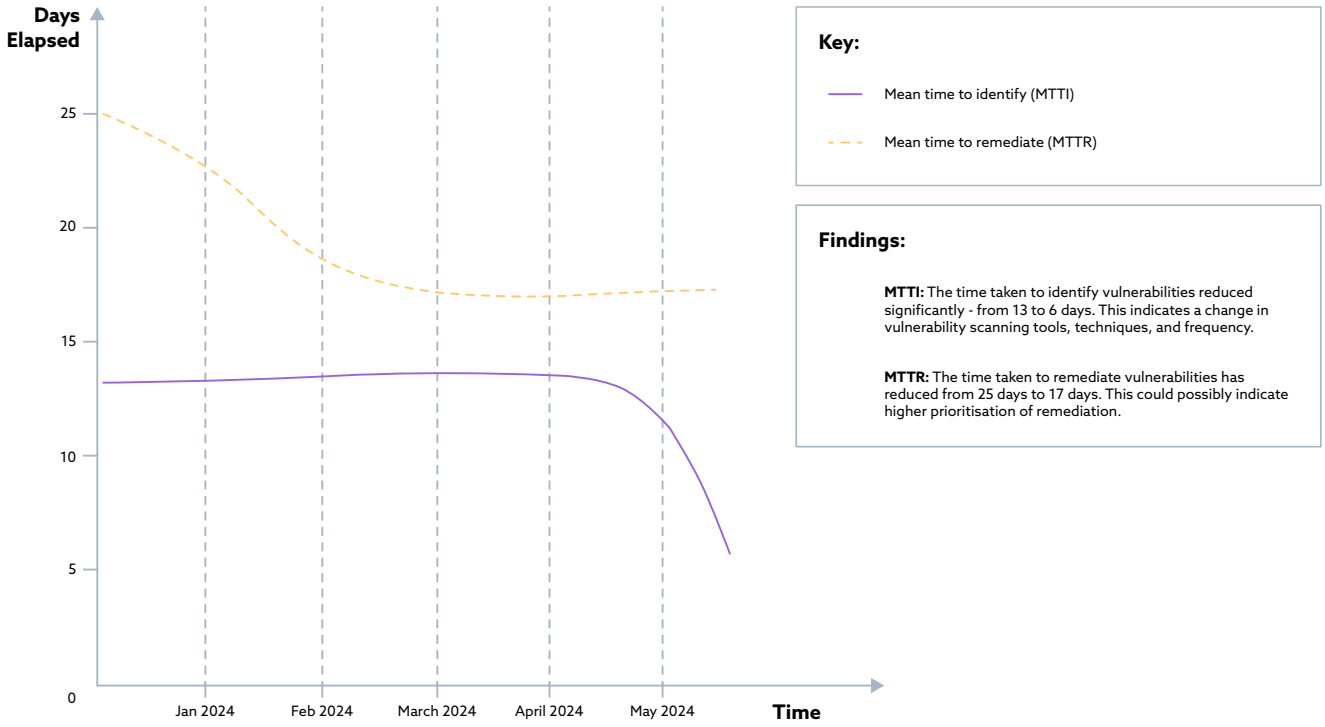


Figure 8: Conceptual observability for exploitable vulnerabilities

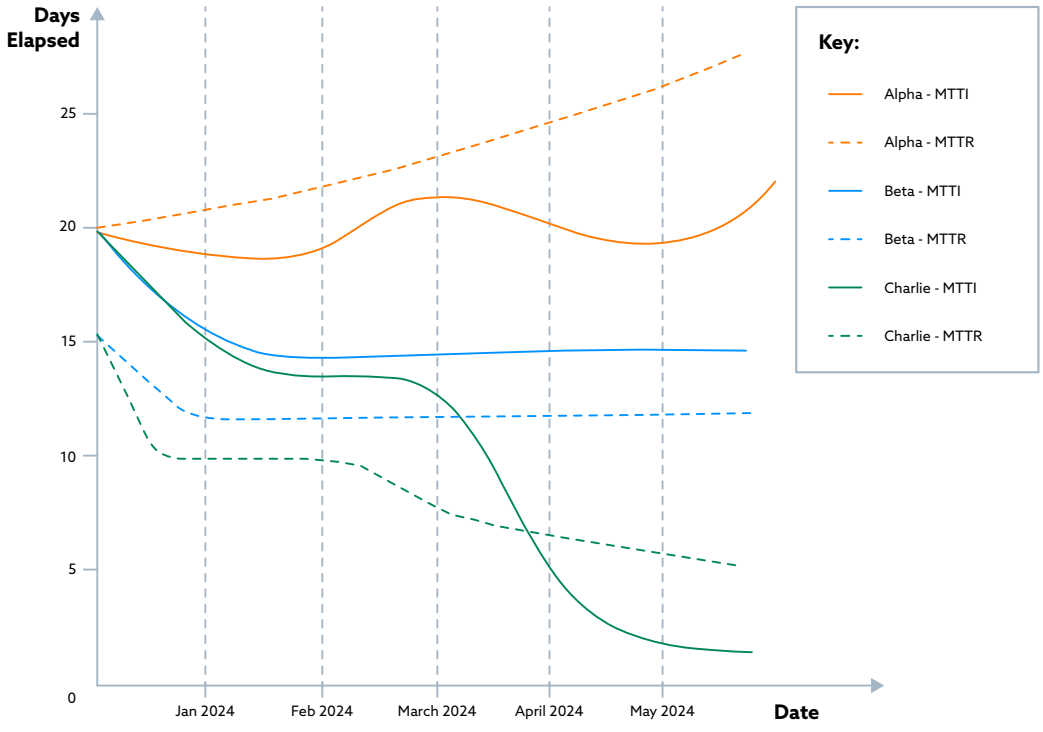


Figure 9: Comparison between teams for exploitable vulnerabilities

Security architecture observability across teams

Figure 10 proves the concept for averaging defense in depth and control re-usage scores, with a demonstration of improvement over time. **Figures 11 and 12** represent the comparison over time across teams, providing an indication of high and low performance for implementation and threat burndown.

The Alpha team's lack of formal security architecture practices results in poor implementation of controls, while the Beta team's moderate practices demonstrate gradual improvement. The Charlie team's robust practices result in a high level of security architecture observability where controls are consistently applied in a short timeframe.

Defense in depth and control re-usage: The Alpha team often identifies the same threats across multiple risk assessments, indicating a repetitive threat pattern without mitigation strategies. While the Beta team has made progress in identifying diverse threats during their assessments, delays in their threat modeling exercises sometimes lead to overlooked vulnerabilities. The Charlie team utilizes a well-defined threat modeling process to predict and mitigate diverse threats. While the process can be scaled across other development teams, regular retrospective sessions ensure that the team learns from past threat scenarios and enhances its threat mitigation strategies.

Threat and control volume: The Alpha team identifies necessary security controls during the design phase, but only a minority of these controls are implemented during production. For instance, while recognizing the need for encryption, they might only implement it at the storage level and not during data transit. The Beta team often realizes the importance of integrated security controls, such as encryption at both storage and transit levels. However, a delay between identification and implementation means that not all recognized controls get promptly embedded into the application. Identified Charlie security controls are promptly enforced. For instance, upon identifying the need for end-to-end encryption, the team ensures it's applied both at storage, transit and during processing. This is managed through automated deployment scripts that enforce a secure configuration.

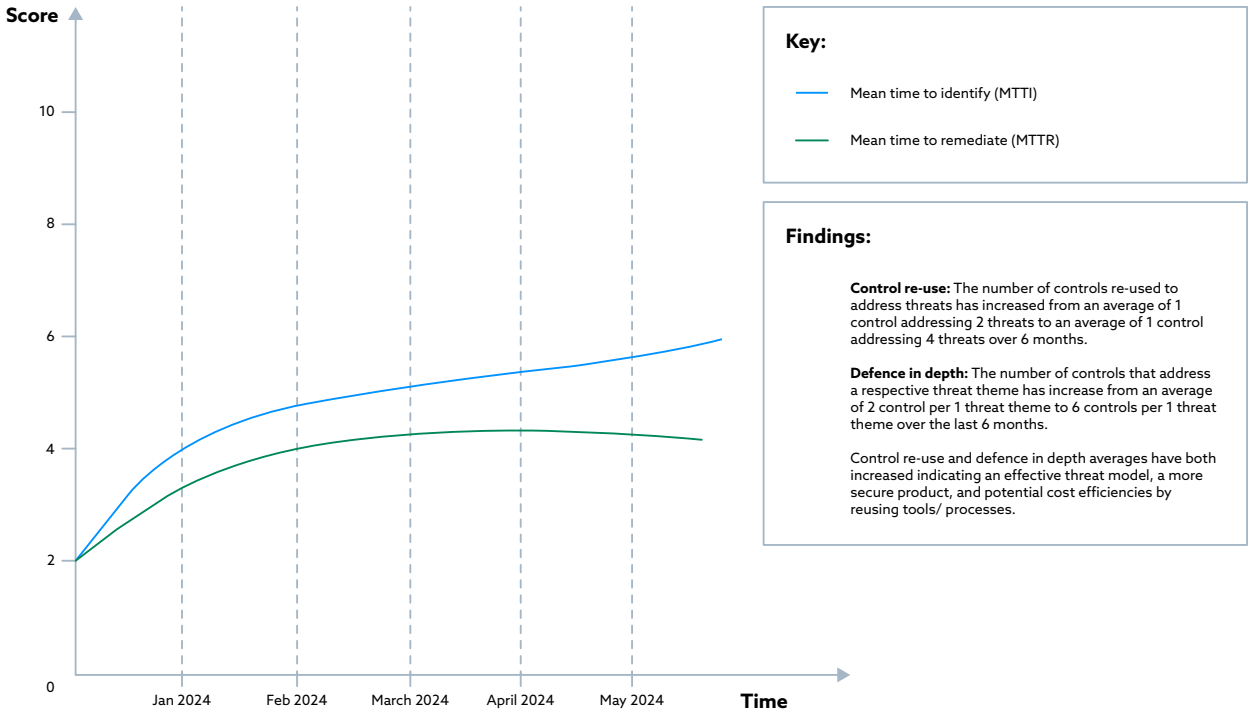


Figure 10: Conceptual observability for security architecture

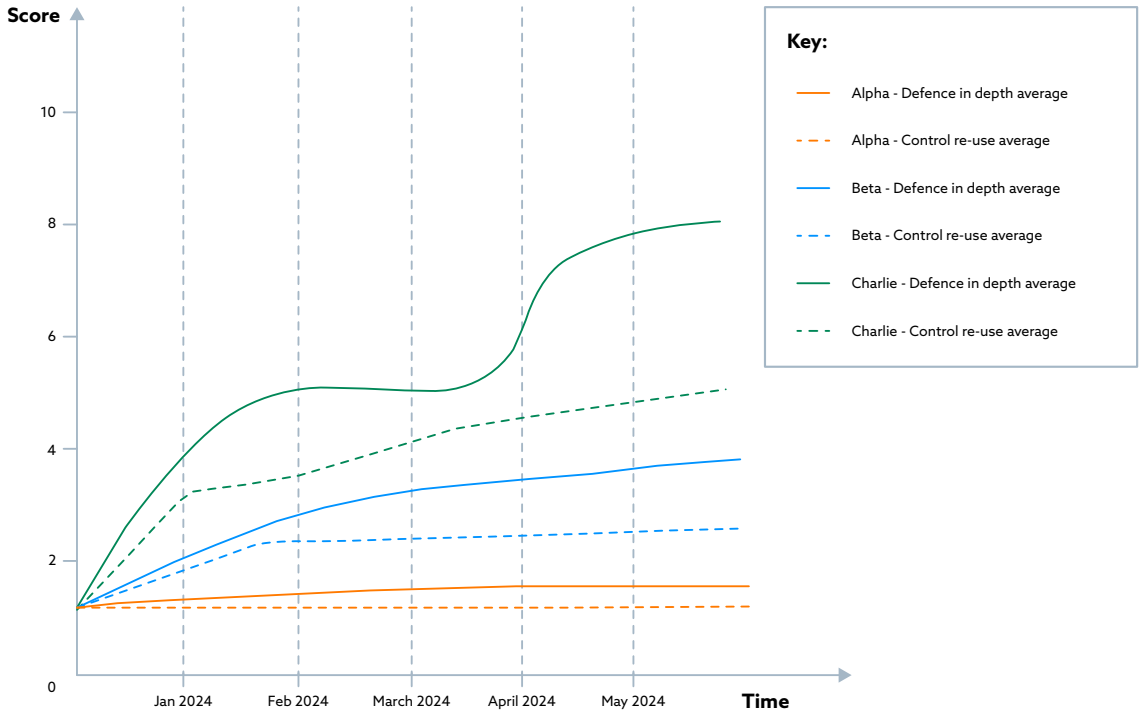


Figure 11: Comparison between teams for security architecture

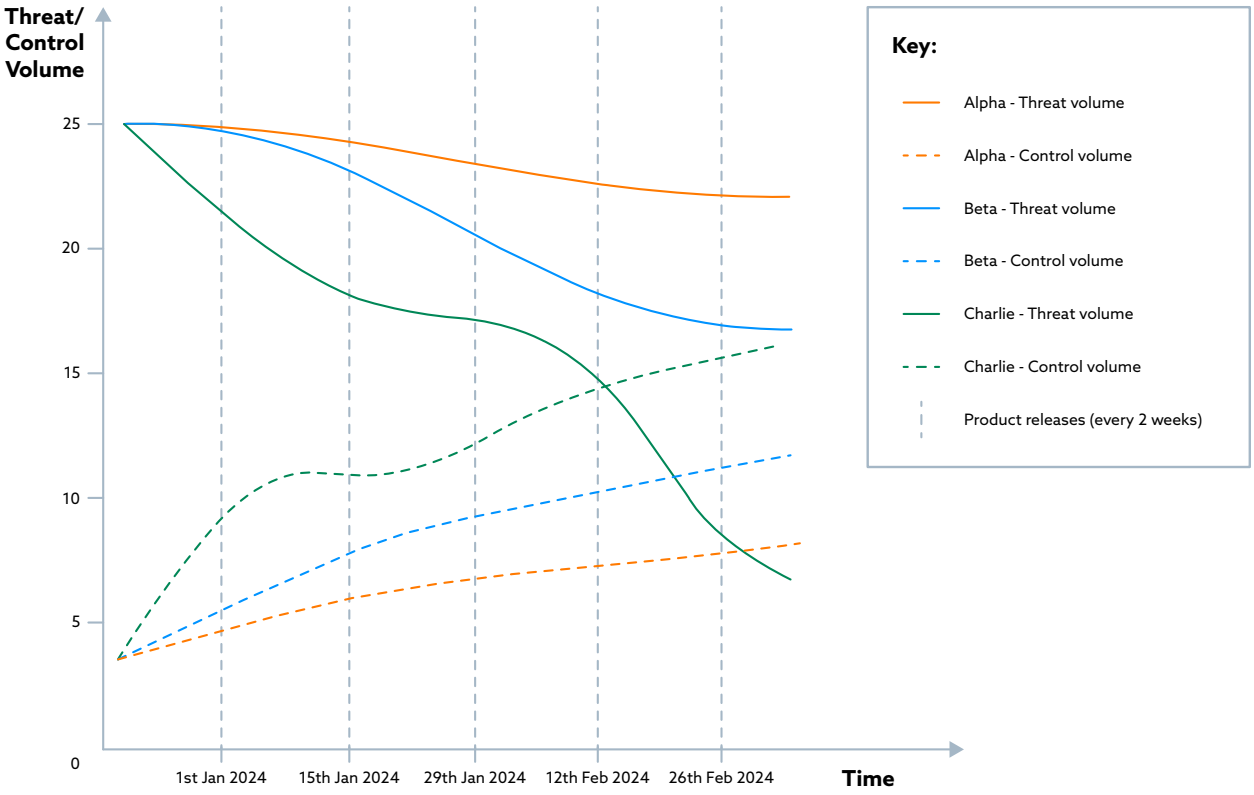


Figure 12: Comparison between teams - threat burndown

Incident response observability across teams

The Alpha team's lack of a formal incident response plan results in poor observability of incident response actions. The Beta team's basic plan improves observability, while the Charlie team's well-defined and regularly updated plan results in excellent incident response observability.

MTTD: The Alpha team lacks effective monitoring tools and practices, leading to a prolonged MTTD averaging at 19 days and increasing over time. Due to established monitoring capabilities, the Beta team reduced this time to 15 days to detect with indicators of improvement over time; however, insufficient process and staffing are resulting in a plateau to improvements. The Charlie team has a set of detection and response capabilities with robust processes, meaning it's able to sustain MTTD in a few days.

MTTC + MTTRN: Incidents often take time to resolve for Alpha and the backlog creates an overload that results in increased MTTC and MTTRN scores over time. Beta has established tooling in place that supports triage but are limited to making sustainable improvements over time. Charlie has reduced its MTTC and MTTRN continuously over time due to their effective use of tooling, people and processes; the project is able to pivot and reprioritize containment and restoration activities.

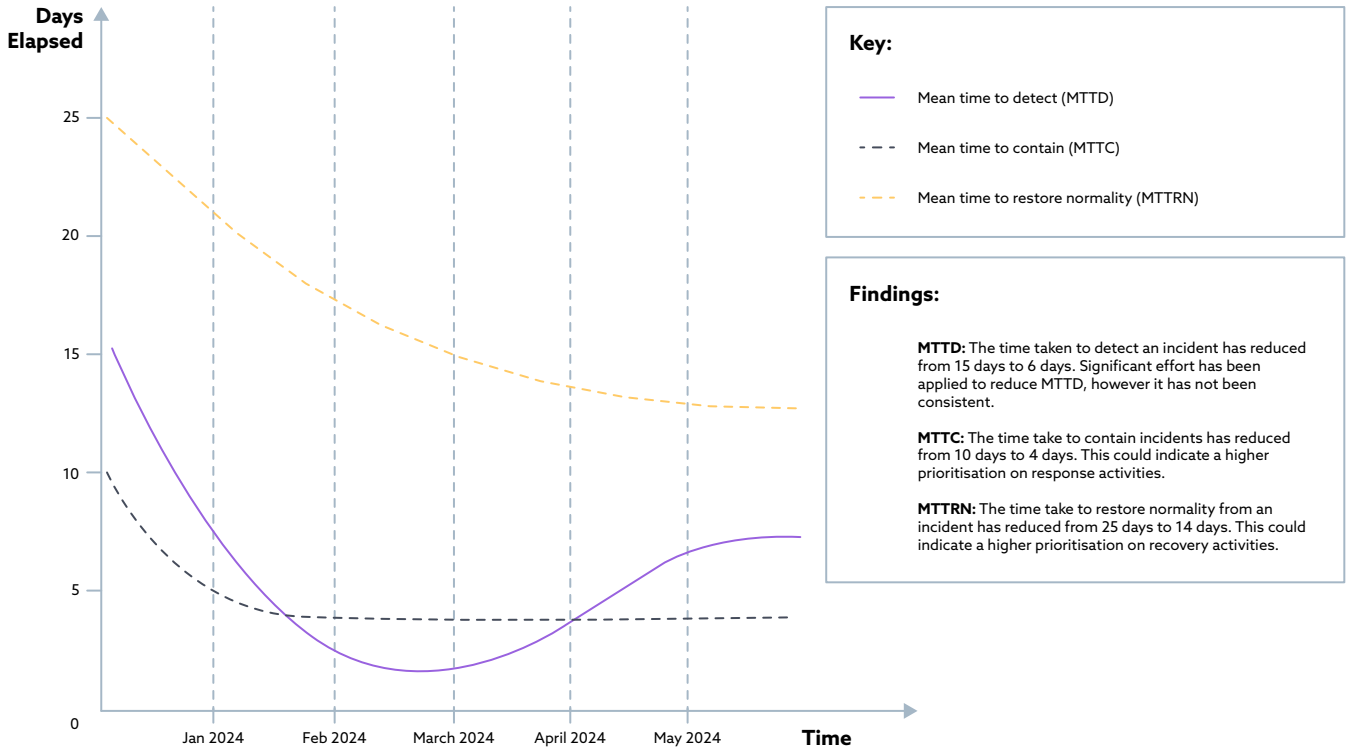


Figure 13: Conceptual observability for incident response

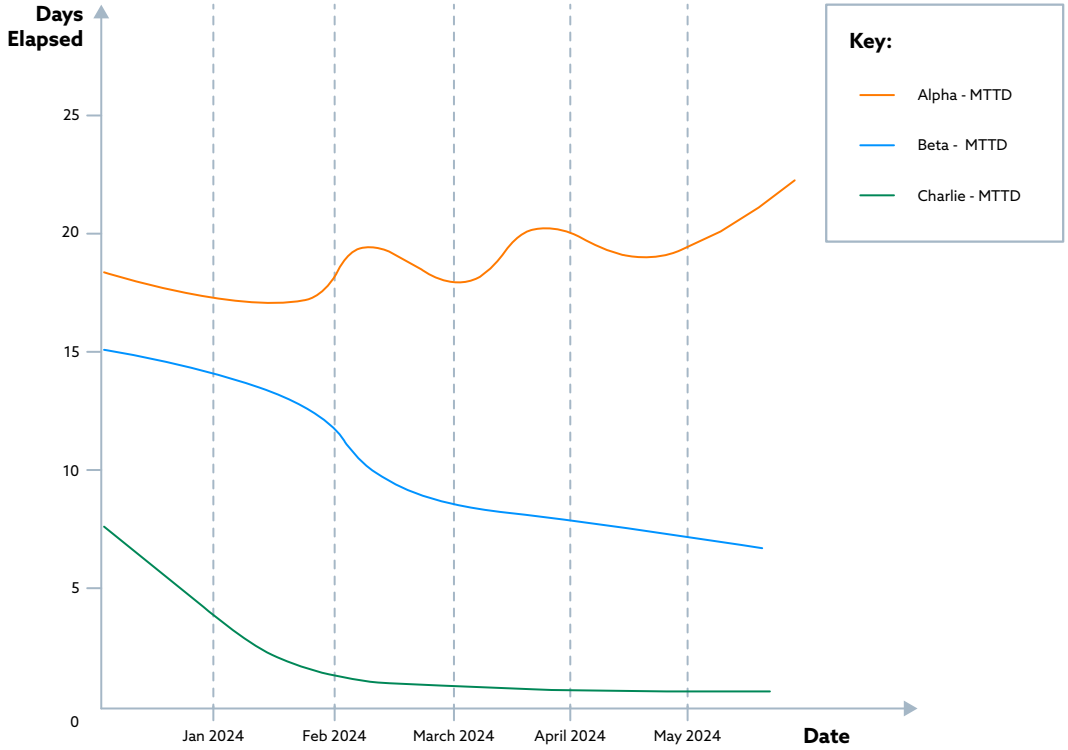


Figure 14: Comparison between teams - MTTD

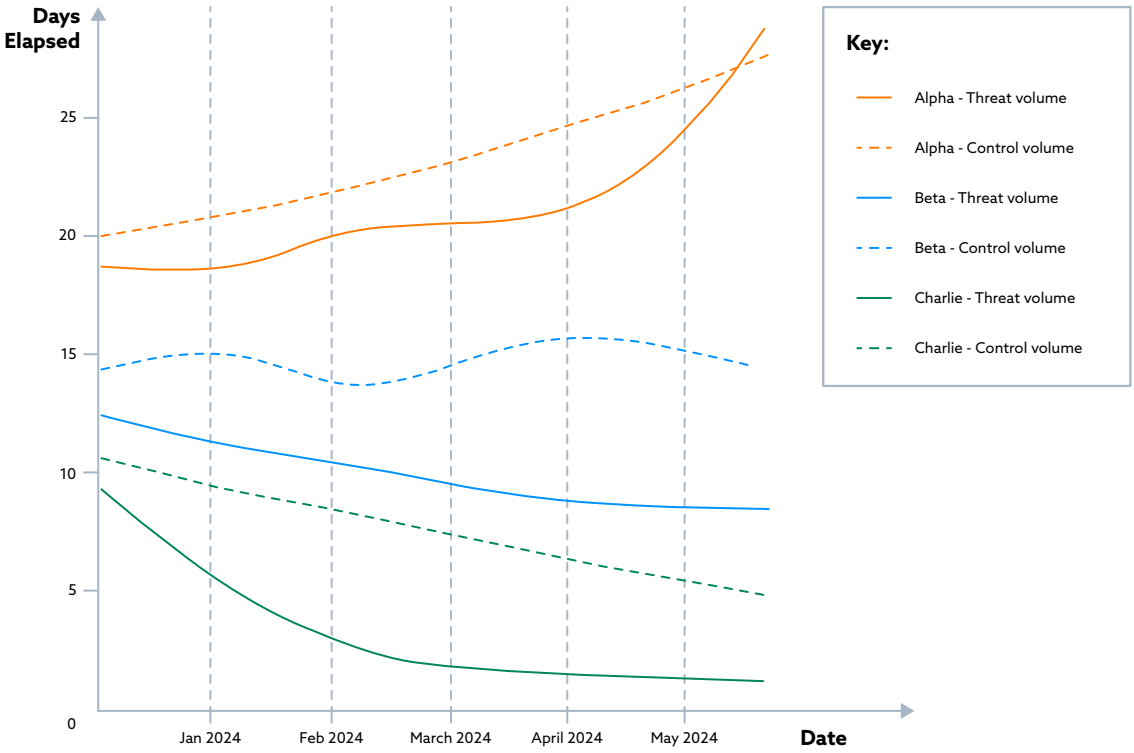


Figure 15: Comparison between teams - MTTC and MTTRN

Improvement Through Reporting

Reporting is a key aspect organizations should consider when embedding security, as it not only gives a view of the current security posture but also offers senior leaders the ability to assess and manage risks, security budgets, and resources. Continuous improvement through reporting is achieved by determining what is relevant for project teams versus what should be reported to senior management. It's so much more than simply sharing a report of vulnerabilities provided by a security solution.

Four principles and their application are described that should be considered when building out reporting to provide maximum benefit to the organization, with a roadmap to provide the journey to reporting success.

Principle - Make data accessible and observable

By gathering and representing informative data, an organization is able to assess the current security posture, allowing for trend analysis and risk mitigation. The benefits derived could include the following.

- **Increased Risk Awareness:** Increased risk awareness is a direct result of accessible and observable data. When data is readily available, a team can better identify potential risks and vulnerabilities, allowing for proactive risk mitigation strategies.
- **Increased Investment within Security:** Accessible data allows organizations to allocate resources more effectively by identifying areas where additional security measures are needed. This leads to increased investment in security, particularly through continuous improvement initiatives.
- **Threat Landscape Insight and Future Planning:** Data accessibility provides insights into the threat landscape, enabling organizations to plan for future security challenges. It empowers organizations to adapt and respond to emerging threats.
- **Faster Incident Response Activities:** Faster incident response activities are facilitated through accessible and observable data. When security incidents occur, having readily available data accelerates the mean time to detect (MTTD), mean time to contain (MTTC), and the mean time to restore to regular business operations (MTTRN).

Principle - Highlight Areas of Opportunities

This principle plays a crucial role in the realm of reporting, as it focuses on identifying areas within DevSecOps practices where improvement is needed. It serves as a means for enhancement opportunities. These opportunities are not limited to technical aspects alone. They may encompass a broader spectrum, including improvements in incident response plans, staff training, or adopting new security tools and practices. The benefits from this reporting include but are not limited to the following.

- **Enhanced Security Posture:** Identifying and addressing areas of improvement directly contributes to a more robust security posture. Organizations can better defend against potential threats by proactively addressing vulnerabilities and weaknesses.
- **Continuous Improvement:** Identifying and addressing areas of opportunity fosters a culture of continuous improvement within the organization. Teams can learn from past shortcomings and evolve their practices.
- **Tailored Solutions:** Reporting allows organizations to tailor their solutions to specific challenges and weaknesses. For example, if training gaps are identified, targeted training programs can be developed to address those specific needs.
- **Adaptation to Changing Threats:** As the threat landscape evolves, reporting enables organizations to adapt and respond effectively. It allows for timely adjustments in security practices to stay ahead of emerging threats.

Principle - Highlight change to drive continuous improvement

This principle underscores the importance of reporting as a mechanism for driving continuous improvement across an organization. It emphasizes the need for learning and development initiatives to enhance security awareness and culture. Implementing this principle is often manifested through comprehensive Learning and Development (L&D) programs for staff, which aim to elevate the overall security consciousness of the organization. The derived benefits from applying this principle can include the following.

- **Reinforced Learning Culture:** Emphasizing change through reporting enhances an organizational culture that values learning. By regularly reviewing and acting on reporting insights, employees are encouraged to continually acquire new skills and knowledge.
- **Dynamic Adaptation to Security Needs:** As reporting highlights the evolving requirements of an organization's security posture, it enables dynamic adaptation. This means that as new vulnerabilities are reported, the organization can quickly pivot its strategies and training programs to address these specific areas.
- **Enhanced Security Responsiveness:** With a focus on continuous improvement, the organization becomes more agile and responsive to threats. Learning from reports and implementing changes reduces the likelihood of repeat incidents, thereby enhancing the overall responsiveness to security incidents including MTTR, MTTD, and MTTRN.
- **Proactive Risk Management:** Continuous improvement driven by reporting allows organizations to transition from a reactive to a proactive approach. By analyzing trends and patterns, organizations can anticipate and mitigate risks before they materialize into breaches.

Principle - Encouraging Communication and Collaboration

The principle of encouraging communication and collaboration through reporting is central to building a unified approach. By offering an overview of the organization's security practices and DevSecOps activities, reporting acts as a way to enhance understanding and cooperation. This not only supports the current security needs but also prepares the organization to respond to and manage future security challenges effectively. The derived benefits from this collaborative reporting approach include the following.

- **Senior Leadership Engagement:** Reports that convey the organization's security status enable senior leadership to make informed decisions regarding security investments and initiatives. This involvement is critical for securing the necessary resources and attention for security-related matters.
- **Awareness and Risk Identification:** A widespread understanding of security risks is facilitated by consistent reporting. This awareness is essential for the early detection and management of potential security threats within the organization.
- **Development of Security Culture:** An improvement in security culture and awareness is a natural outcome of enhanced communication. As reporting sheds light on security issues and how they are being managed, it enhances a mindset where security becomes an integral part of the organizational culture.
- **Strengthening Team Relations:** Reporting bridges the gap between development and security teams, leading to better collaboration. When these teams are aligned on security goals and practices, the organization can address these challenges more efficiently.
- **Proactive Security Measures:** The integration of security SMEs into development teams, as reported, promotes proactive security measures. This 'shift left' approach ensures that security considerations are addressed early in the development process, thereby reducing potential vulnerabilities and streamlining remediation efforts.

Applying these principles for reporting

These four principles of making data accessible, highlighting opportunities, driving continuous improvement, and encouraging communication and collaboration are crucial to driving improvement through reporting. Together, these principles form a comprehensive approach to measuring and improving security throughout the DevSecOps lifecycle. When these principles are integrated into the reporting process, an organization can transform its security metrics from reactive data points to proactive instruments of security governance and strategic decision-making.

Source: [4]

4 Rajapakse, Roshan N., et al. "Challenges and solutions when adopting DevSecOps: A systematic review." Information and Software Technology 141 (2022): 106700.

Roadmap for Reporting

The compass for your journey

How can we shift our thinking to a 'security + performance = value' mindset?

1 Ultimate Future Ambitions

Measuring security by performance across your entire organisation and using security observability to manage risk, prioritise projects / activities and effectively allocate budget.

2 Today's Challenges

Understand the challenges your projects and organisation as a whole face- the root cause of low performance might not be the security team. Ensure you're able to collect data continuously.

3 Project Goals

There isn't a single tool that can provide you with security observability, so treat this like it's own project. Establish success criteria. If your project is aiming to build a product with aggressive time scales, balance it with the cost of cyber risk (e.g. data breach) and set your security observability targets appropriately.

4 Security observability milestones

Start by establishing a proof of concept with 1 project to win hearts and minds. If you're able to demonstrate observability for vulnerability management - the immediate next question will be where else can we apply this approach. Scale vertically to against your organisation's data points and work with what you have (don't procure tools just so you can demonstrate observability). Scale horizontally across multiple projects once you gain advocacy and leadership support.

Climbing Equipment

Ensure you're well equipped with the data points/ the tools to provide you with data before starting. You'll need data from vulnerability scanners, incidents will need to be recorded on IT service management systems and your security architects should already be threat modelling.

Ways of Working

Don't work in silo. Involve leadership and advocates from within and outside your organisation - this includes security and non-security stakeholders.

Conclusion

Implementing security into any digital project or organization is no easy feat. Businesses, incentivized to make money, are often pushing to release features faster, with market share at stake. The design and build of infrastructure and applications are measured against a complementary metric — performance (i.e., speed to deploy, time taken to create feature, time to test) — often a great fit to measure against business goals.

Security, in contrast, very effectively measures success by compliance rather than performance. The ability to improve a security function often leads to teams/products becoming more compliant. Compliance and performance can sometimes be opposing forces that require a delicate balance to avoid the perception that "security is a blocker."

This paper provides empirical evidence that three critical aspects of security and DevSecOps can be effectively evaluated using performance metrics. We calculate how well projects/organizations are doing on their ability to manage vulnerabilities, architect secure services, and respond to incidents, providing security observability.

Security observability provides an effective measurement for business leaders to inform them of security efficiencies and performance and, most importantly, where further investment and change are required.

The CSA DevSecOps working group will continue its research in providing security observability insights and scale the use cases identified in Appendix B.

Appendix A: Decomposing the Software Lifecycle

Organizations have a wide array of tools and solutions to choose from when implementing security into their Software Development Lifecycle (SDLC). In "Pillar 3: Pragmatic Implementation," we explored the different security activities to infuse security into the design, coding, testing, deploying, and monitoring stages of a Software Development Lifecycle (SDLC).

Using a framework-agnostic DevSecOps model focused on application development and platform security, we decompose the stages from **Figure 1,** where metrics can be identified to provide data (see Appendix B). While there are many different ways to define an SDLC and its stages, the five-staged SDLC (referenced in **Figure 1**: Design, Coding, Integration and test, Deployment, and Monitor) offers a universal view of software development.

We recognize that tools and solutions can be hard to deploy and challenging to operationalize and scale. Existing solutions may not provide actionable insights that can help mitigate security risks, or at least that's the perception. We unearth how to use metrics, derived from the DevSecOps stages, to understand security performance in DevSecOps-related activities.

Figure 16: Pillar 3 DevSecOps Activities

Appendix B: Measurement

From each stage in **Table 1**, we've identified an expansive list of possible metrics to capture that encourages holistic decision-making. Each metric reflects its common use by product and security teams across all industries. While Table 1 is not a comparison or reflection of best practices, these are the types of data that teams can collect during each stage of the SDLC.

Stage	Metric
Design	• Threat vector per component, function, and feature • Time to complete threat modeling exercise • Cost and time of addressing threats • Security control/threat ratio • % of re-used (existing) security controls • % of security controls that are product features • % of stories covering security requirements and acceptance criteria • Security story volume • Ratio of security story to functional story • Time to complete security user story • Average lead times for security work • % of issues detected and fixed during development • % of issues detected and fixed during testing • Risk identification volume at design • Risk mitigation volume as a result of design changes • % of risk resolution (mitigated, transferred, accepted, avoided) during design • Average time take to identify and document risks during design • Volume of architecture principles and artifacts • % of architectural principles met by product teams during design • Volume of security architecture reviews
Coding	• % of developer/engineer uptake in security and privacy training • Volume of lessons learnt sessions over a period of time • % of rejected code changes at pre-commit • % of notifications at post commit • % of code changes peer reviewed before merge • % of code that has been linted • Volume of total code linting errors • Time required to fix linting errors • % of open source components used that meet organizational approval / pass SCA scans • Volume of licensing issues identified • Volume of unaddressed licensing issues for products used in production • % of up-to-date dependencies • Volume of dependencies with known vulnerabilities

Coding	• Average time taken to update dependencies • % of IAC code scanned • % of source code scanned • Volume of checks across code base (by language) • Volume of rejected IaC deployments due to security misconfiguration • Volume of detected security findings • % of SAST false positive rate • Volume of security tickets to address security issues identified from SAST/ IaC scans • Volume of hardened containers • % of hardened containers used in production • Average frequency of hardened container updates • Volume of Container non-compliance coverage against CIS Benchmarks or other standard • % of unit test tickets that are security related • Throughput time of security unit tests • Passing rate of successful tests and failures • % of code and architectural changes that are peer reviewed for security vulnerabilities
Testing	• Volume of vulnerabilities detected in runtime by DAST, IAST, or continuous penetration test • Average lead time to perform different types of runtime tests by DAST, IAST, or continuous penetration test • Volume of high/critical severity of detected vulnerabilities • % of DAST/IAST false positive rate • Average application feature test coverage from DAST/IAST • Time to complete respective runtime tests like DAST, IAST, and penetration test • Ratio of high/critical runtime vulnerabilities compared to medium, low and informative • Volume of tests with responses of invalid, unexpected, or malformed content • % of API coverage during security runtime tests • Volume of vulnerabilities in container images • Volume of security incidents involving containers • Volume of security controls applied to containers • % pass rate of integrity checks
Deployment	• Deployment frequency and success rate • Average time to deploy infrastructure to production • Average time to identify configuration drift • Average time to remediate configuration drift • Average manual intervention steps required to repair configuration drift • Volume of guardrails applied to cloud resources • Volume of policy statements converted to guardrails (e.g., access control and resource management)

Deployment	• Volume of security incidents per environment • Volume of environment isolation tests • Average lifetime of secret before rotation • Volume of secrets stored in the CI/CD pipeline • Average time elapsed for security scanning triggered on the pipeline • Average time elapsed to deploy to production • % of pipeline deployment failures • % of hardened/approved images in use • % of secure configuration coverage on images (e.g., CIS Benchmark Level 1 vs. Level 2)
Monitoring	• Mean Time to Recover (MTTR) • Change Failure Rate • Service uptime for users • Volume of Security improvement opportunities identified post-release • Mean time to detect intrusion • Mean time to respond to intrusion • % of resources logged against STRIDE use cases • Volume of attack vectors (ATT&CK) from Breach Attack Simulation • Volume external endpoints complying to organizational security policy • % of production related incidents compared to non-prod • Average time to identify, investigate and resolve breaches • Volume of incidents by product • Mean Time to Remediate (MTTR) • % of vulnerabilities remediated in production • Average effort (t-shirt size) to remediate or patch • % of vulnerabilities remediated that have direct correlation to threat models

Table 1: Metrics

References

Sources: [5] [6]

Cloud Security Alliance,The Six Pillars of DevSecOps: Pillar 1, Collective Responsibility, February 2020, Available @ https://cloudsecurityalliance.org/artifacts/devsecops-collective-responsibility/

Cloud Security Alliance, The Six Pillars of DevSecOps: Pillar 2, Collaboration and Integration, February 2024, Available @ https://cloudsecurityalliance.org/artifacts/devsecops-automation/

Cloud Security Alliance, The Six Pillars of DevSecOps: Pillar 3, Pragmatic Implementation, September 2022, Available @ https://cloudsecurityalliance.org/artifacts/devsecops-automation/

Cloud Security Alliance, The Six Pillars of DevSecOps: Pillar 4, Bridging Compliance and Development, February 2022, Available @ https://cloudsecurityalliance.org/artifacts/devsecops-pillar-4-bridging-compliance-and-development/

Cloud Security Alliance, The Six Pillars of DevSecOps: Pillar 5, Automation, July 2020, Available @ https://cloudsecurityalliance.org/artifacts/devsecops-automation/

Cloud Security Alliance, Information Security Management through Reflexive Security: Six Pillars in the Integration of Security, Development and Operations, August 2019, Available @ https://cloudsecurityalliance.org/artifacts/information-security-management-through-reflexive-security/

Cloud Security Alliance, The Six Pillars of DevSecOps, Achieving Reflexive Security Through Integration of Security, Development and Operations, August 2019, Available @ https://cloudsecurityalliance.org/artifacts/six-pillars-of-devsecops/

The Forrester New Wave™: Cybersecurity Risk Rating Solutions, Q4 2018, Available @ https://go.forrester.com/blogs/examine-the-cybersecurity-risk-ratings-market-with-forresters-new-wave/

Forrester Research on DevOps Quality Metrics that Matter, Tricentis Sponsored 75 Common Metrics Ranked by Industry Experts, 2021 Available @, https://www.tricentis.com/resources/forrester-research-on-devops-quality-metrics/

A Measurement Companion to the CIS Critical Security Controls (Version 6) October 2015, Available @ https://CIS-Critical-Security-Controls-VER-6.0-10.15.2015.pdf

Measures and Metrics in Corporate Security, Security Executive Council, 2011, Available @ http://council.com/content/measures_metrics_mini_200801.pdf

5 Casola, Valentina, et al. "A security metric catalogue for cloud applications." Conference on Complex, Intelligent, and Software Intensive Systems. Springer, Cham, 2017.

6 Nichols, B. , 2021: The Current State of DevSecOps Metrics. Carnegie Mellon University's Software Engineering Institute Blog,. http://insights.sei.cmu.edu/blog/the-current-state-of-devsecops-metrics/ (Accessed December 27, 2022)

Carnegie Mellon University, Software Engineering Institute: The Current State of DevSecOps Metrics, March 29, 2021, Available @ https://insights.sei.cmu.edu/blog/the-current-state-of-devsecops-metrics/

GSA Tech Guides, DevSecOps Guide, July 2021, Available @ https://tech.gsa.gov/guides/dev_sec_ops_guide/

Information Technology Laboratory / Software and Systems Division, Software Quality Group, Metrics and Measures, available @ https://www.nist.gov/itl/ssd/software-quality-group/metrics-and-measures

OWASP DevSecOps Maturity Model DSOMM, DSOMM (owasp.org)

Model, Governance, Education and Guidance, Training and Awareness, Training and Awareness (owaspsamm.org)