National Defense-ISAC

Software Security Controls – Metrics Automation

October 21, 2021

Authors:
Raul Barreras
Paul Keim
Waldemar Pabon
Joe Vega
Andrew Zuehlke

Reviewers:
Will Jimenez
Renee Stegman

Executive Summary

The implementation of security controls in the Software Development Lifecycle (SDLC) provides not only protection to the software produced but also offers critical information to assist with security vulnerability remediation. Despite the value of data associated with security vulnerabilities identified by the different security controls in the SDLC, most solutions save scan results in isolation even when having multiple tools producing data. For organizations to have a unified view of risk and compliance levels, it is important to centralize data captured as well as define metrics that would help increase the efficacy of the different teams and the organization.

Establishing a metrics strategy allows an organization to measure what is or is not working. The trends identified by this data will assist in establishing strategies to mitigate recurrent findings and in implementing the overall direction. The harmonization behind the usage of metrics is a key aspect of metrics management. The metrics strategy must consider the technical, operational, and organizational goals and priorities. Therefore, the definition of the metrics and the scope associated with each component becomes an important ingredient in the development of a metrics management system. Successfully centralizing the capture and processing of metrics ultimately helps correlate important aspects of the software delivery pipeline that can empower an organization to understand the current state of your enterprise.

The journey towards implementing metrics that leverage security controls scan results and data in the SDLC, must follow a building block approach. First, the organization needs to identify

which metrics will help improve the security stance of applications and consider how this information impacts the different roles involved in the detection, remediation of security vulnerabilities, and compliance. Developers will have different concerns compared to the overall compliance overview needed for executive leadership. Nevertheless, all the data captured can support decision-making at all levels and can be captured from the same data sources.

Once the metrics are identified, a data store must be designed and implemented. Selecting a schema that fits within the desired scenario will help the implementation team understand how to define the data store for the measures and the pieces of information that will be used to evaluate and present the metrics. For the purpose of this paper a star schema was used to drive the implementation discussion, though a traditional data warehouse structure can also help support reporting requirements for the metrics. In a business intelligence (BI) setting, the star schema is the most commonly used for this scenario (Choi, n.d.).

Now that we have defined the data store structure for the data warehouse, the automation of data aggregation becomes the next step. Here, leveraging the use of Application Programming Interface (API) services is an important aspect of the implementation, as it will create efficiencies and cost-saving scenarios for the organization by minimizing the human involvement in the process. The organization should plan to identify which API services are needed to help support the storing of the API-related information required to support the data visualization. After the implementation of the data aggregation, an organization is in a good position to start with the

implementation of data visualizations that support the analysis and decision-making. For data visualization, multiple strategies exist, such as dashboards, scorecards, and reports. Regardless of which option is chosen, organizations need to ensure the right data is included to ensure all involved parties have the information needed to understand what needs to be fixed and how a program can be improved moving forward. However, having too much data or too little information can hamper the ability to identify risk and compliance levels.

Finally, a communication plan needs to be established to ensure metrics reach their intended audiences in a meaningful and impactful way that helps reduce the risk of security vulnerabilities in the organization. Some approaches may require dashboards to be widely available for different key roles. In other instances, email or slide decks in presentations may be the preferred option. Each organization needs to evaluate which method works best for each role and the strategic objective of maintaining a strong security stance in application security. All the steps presented in this paper will help any organization to follow a phased approach towards the implementation of an application security metrics program. The aspiration behind a metrics program is to help bring the number of security findings down in production environments while improving the efficiency of development teams.

Table of Content

Introduction

Objective

In previous Software Security Automation white papers published by ND-ISAC, the Application Security Working Group introduced various aspects of implementing and maintaining security controls as part of a software development life cycle (SDLC). The first white paper, "Software Security Automation: A Roadmap toward Efficiency and Security," introduces and offers guidance in terms of the security controls needed when automating software security in both traditional waterfall and agile scrum methodologies, as well as in organizations transitioning into a DevSecOps practice; "Software Security Automation: Security Controls Evaluation Criteria" builds upon the first white paper and provides a quantitative approach to evaluating SDLC security tools against the needs of the organization; the third white paper, "Remediation Workflow Automation," explores a standard process—and the corresponding remediation steps—that should be used by teams when addressing security vulnerabilities in the software delivery process.

To highlight the benefits and effectiveness of implementing security controls as part of the SDLC, this white paper complements the previous papers by examining metrics from security controls, as well as metrics that can be extracted or deduced from security control data when cross-referenced with peripheral information. Metrics can be used to measure the progress of a program, identify areas needing attention, and better understand overall risk, among many

others. This paper explores what can be done with metrics, proposes a method for consolidating data in a central hub, highlights metrics to form effective communication for target audiences, and presents opportunities to incorporate metrics in visualizations.

This paper does not focus on a specific tool or vendor, but instead uses a generalized approach to identifying common metrics and overarching ideas that are essential to an effective SDLC program. Ultimately, however, each organization—and even individual teams within a single organization—must define the applicable controls and metrics for their specific circumstances.

Audience

The audience for this white paper includes software and security engineers, software and security architects, product managers, senior managers, and executives responsible for the selection, implementation, integration, and continued management of software security automation tools in the organization.

Structure of the paper

This paper begins by introducing a number of the most commonly used metrics from security controls. Additional metrics that can be correlated from security controls data are also presented. With such a wide audience, this paper highlights which metrics are most applicable for varying levels of leadership. A method for establishing a data warehouse structure to store metrics is then introduced, followed by a data collection process, establishing a process for visualizing data and identifying a strong communication plan. Finally, a comparison of native and custom visualization tools is presented.

Security Controls Metrics Automation

Security Control Data Collection Process

Implementing a data collection for the application security metrics that will be used by an organization requires strategic planning and execution. A proposed process below is composed of four basic steps that will enable the organization to be effective in the implementation of a metrics program for application security. Figure 1 provides a high-level overview of the process required to establish an application security metrics program that will help the organization understand the risk exposure and the overall compliance level.

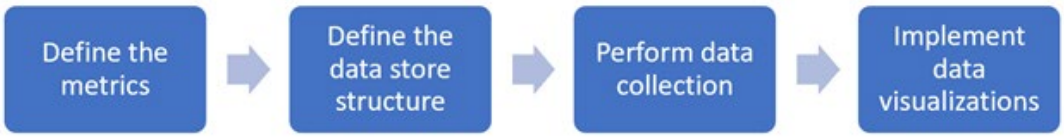

Figure 1 Process to establish an application security metrics program

First, an organization needs to identify and define the metrics needed to understand the level of risk and compliance. Since the security controls will provide most of the data used during the data collection process, it is important to identify the gaps associated with the criteria to be used for analyzing the data, as well as any potential measure needed that may not be directly included in a security control's scan results. This understanding will help define whether the strategy will require the use of native data reported by the security controls, use of a cross-reference datastore, or finding an alternative source that helps illustrate the required relationships to

enrich the data analysis. The definition of the metrics is an important prerequisite for the data collection, as it defines the requirements for the data store.

Right after the metrics have been identified based on what the organization is trying to measure, the automation process for the data collection can be delineated. During the data collection, an organization needs to understand where the data is, which mappings are needed, and how the data collection process will be executed. Validations to ensure data integrity in the metrics should also be carefully considered to ensure data presented to the end-user is meaningful. The data collection process will ensure the information regarding the measures and the analysis criteria will be loaded to their corresponding data stores.

Once the data collection process has been implemented, an organization can start with the implementation of the visualization components that will support data analysis. Measuring the data will require sharing results with the end-users. A proper component for data visualization must be identified based on the different roles and needs in an organization. Dashboards, scorecards, and reports are some of the most common components used during the visualization of the metrics.

It becomes important to understand how the implementation of metrics aligns results with business objectives and compliance requirements. Being able to measure and analyze the security state of software as it progresses through the software delivery pipeline is an important

step when identifying risk exposure. Metrics evaluation will help the organization create strategic approaches to minimize the impact of previously identified security vulnerabilities in software using a lesson learned approach. This approach will drive operational efficiencies while improving the quality of the software being developed. The next several sections in the paper provide a detailed approach to define the metrics needed, automate the collection of data, and define which visualization tools are needed to help implement a comprehensive measurement program for application security.

Security Control Metrics

Early in the implementation of security controls, some of the most important metrics come directly from the security controls themselves. These metrics provide information that can help identify actionable results and influence action. Where possible, metrics from security controls should be made into a feedback loop that is as automated as possible, using recommendations from the first three white papers. [1][2][3]

Metrics focus on numerical data produced by tools used throughout the SDLC and can vary depending on the type of security control involved. For example, a Static Application Security Testing (SAST) solution may produce a statistic for the number of vulnerabilities before compiling the source code, whereas the equivalent statistic for Dynamic Application Security Testing (DAST) or Interactive Application Security Testing (IAST) would highlight the number of vulnerabilities identified after a program has been compiled. This section will explore a few of the metrics available from security controls; a more in-depth assessment of a few of these metrics can be found in the Metrics per role section of this paper.

One of the most common metrics available from all vendors and across all tools is "severity;" not all solutions utilize the same scoring system, but they offer metrics around the number of

[1] Application Security: Remediation Workflow Automation: https://ndisac.org/blog/appsec-remediation-workflow-automation/

[2] Application Security: Security Controls Evaluation Criteria: https://ndisac.org/blog/software-security-automation-security-controls-evaluation-criteria/

[3] Software Security Automation: A Roadmap Toward Efficiency and Security: https://ndisac.org/blog/software-security-automation-a-roadmap-towards-efficiency-and-security/

vulnerabilities for a given severity. For example, many tools categorize vulnerabilities as either critical, high, medium, and low; a key metric immediately available after a scan completes is the number of vulnerabilities detected in each severity category. Often, scanning tools align the severity with the Common Vulnerability Scoring System (CVSS); in these circumstances, each vulnerability will be assigned a score between 0.0 and 10.0, which can then be used to calculate averages for severity.

Another metric available from security controls is the number of vulnerabilities identified for a given scanning tool. This can help identify where additional attention may be needed. Similarly, security controls can associate a vulnerability with an OWASP Top 10 threat category; looking at categories with the highest number of vulnerabilities can indicate an area that may need additional attention.

Additional Metrics

Additional metrics that are created by security controls include, but are not limited to:

Metric Name	Definition
Vulnerabilities per 1k Lines of Code (LOC)	Number of vulnerabilities per 1000 lines of code. This metric allows an organization to develop a more specific risk number/score for an application. This will give visibility on the projects that are the most vulnerable based on the size of the project. This will show the organization teams that might need help or additional secure code training. Also, if the principles are followed that were outlined in the previous paper, like code reuse, less vulnerable code will make it into new applications and only clean code will be reused vs vulnerable code propagating. See **Security Defect Density** in the **Process Compliance Metrics** section for the organizational view of this metric
Vulnerabilities per Application/Team	Number of vulnerabilities for a given application or development team. This is a supplemental metric that can be used in conjunction with the previous metric or standalone. As stated previously, this metric takes the previous metric a step further by looking at the project as a whole and determining the total vulnerabilities. After the aggregation of the findings, the development team that is responsible for the application is identified, and all of the applications they are responsible for, which allows the security team and executive management to see the development teams that are consistently producing non-secure applications.

Vulnerabilities
 	
Comparison (Custom
 	
vs Libraries)
	Number of vulnerabilities in custom code versus vulnerabilities identified in libraries. This is an important metric as developers of modern applications typically leverage open-source libraries for certain functionality vs creating and maintaining the function within their custom code.
	
	This metric looks at findings in both areas in a few ways:
	
	1. Static Application Security Testing of custom code
	2. Static Application Security Testing of open source/library (where applicable or allowed)
	3. Software Composition Analysis of open-source library
	    1. This includes:
	        1. Community support review
	        2. Bad Actors / Contributors to the code base
	        3. Community development practices
	        4. Known CVEs against library

Vulnerabilities Comparison (New vs Repeat Vulnerabilities)	Number of new vulnerabilities versus the number of repeat vulnerabilities. With this metric, security and development teams can see if development teams are remediating security vulnerabilities without introducing new findings, also it allows the security team to identify recurring findings that appear to be common for a specific development team or their organization. When repeat findings are identified, they should be addressed in secure code training and/or in secure coding standards for the organization, this can be as simple as providing examples of the vulnerable code and common ways to mitigate and address them.
Remediation Threshold Compliance	Time to remediate vulnerabilities; this can include the average, maximum, and minimum time. This metric will function to support defined organization remediation timelines. Thus, findings are tracked from when they are first identified to when they are remediated. This allows the security team and development team to see when findings were found, when and how they were addressed (if at all), and how many sprints or releases it took to incorporate the official fix. Organizations can then show metrics on developer, security, or executive leadership dashboards, showing how well teams are complying with defined and established timelines and if any team(s) continuously miss or do not comply with these timelines.

Stale/Dead Code Identifications (LOC)	Identification of dead or unused code within an application, specifically the lines of code count and offending code. This is an important and telling metric as it will show development teams the part of their application that either goes untouched or unused. It also shows areas where the development team can remove the code or refactor the application to be more efficient. From a security perspective, this is important as this stale or dead code can and usually does have vulnerabilities that need to be addressed and prioritized.

Justify the Implementation

Every organization should look at the security control metrics, the additional metrics above and the metrics mentioned in the sections below as critical to show not only the progress of their AppSec programs but also the maturity of their program. The advanced metrics described in this paper provide benefits to all teams and people (i.e., mgmt.) that have a stake in the process, as you will see in this paper, we show how developers, managers and executive leadership can benefit or use these metrics to inform their decision.

Regarding the implementation of the additional metrics above specifically, these metrics allow security management and executive leadership to see high-level metrics and very specific metrics tied to applications and development teams. This can then be used to perform remediation actions like targeted training sessions, development team mentoring, security team shadowing or expediting enforcement of secure coding strategies and frameworks.

Define important metrics per role (Organization, Team, Developers)

Metrics per role

We defined several types of metrics: native, process compliance, and organizational compliance metrics. Metrics can have different meanings depending on who will use them. We describe these metrics, allowed values, and explain the importance of each of them for developers, managers, and executive leadership.

Developers	Managers	Executive leadership
Helps developers understand which bugs to fix first. Knowing the severity of the findings, developers can plan their work, establish an implementation plan, and have an estimate of how long it may take to solve the issue.	Knowing the severity of the findings helps managers plan how to improve processes, determine if additional resources are needed, negotiate delivery times with stakeholders, and discover training opportunities.	Executive leadership abstracts severity to provide a more condensed assessment and understand trends, risks, and SLA compliance. Can use this metric to measure whether the organization is getting better or worse at releasing secure software.

Native Metrics

Severity (Type and Level)

Security Control implemented in the SDLC categorize applications using risk scores or severity. There is no standardized format to represent severity; some tools use nominal values (Severity Type), and others use numerical scores (Severity Level). Severity Type accepted values are

Critical, High, Medium, or Low, although some products could include 'Info'. Severity Level is a numerical value, from 1 to 5, and can be related to a Severity Type, normalizing the output of different tools. Over time, this metric should show a downward trend.

Scan Tool

This metric will collect the class of the tool used to generate the finding. Valid values are: SCA, SAST, DAST, IAST, RASP. Over time, this metric should show a greater number of findings in the tools used in SDLC initial stages.

Developers	Managers	Executive leadership
Knowing the number of vulnerabilities detected by each class of security controls will help developers fine-tune the security controls in the SDLC to support early detection.	Provides a better understanding of the coverage provided by the tools. The identification of the stages in which failures are being discovered helps to find gaps, provide insight into new security controls or tools with different feature sets that could be needed.	This metric shows the extent of how the software automation investment is being used.

OWASP Top 10 Threat Category

This metric collects the OWASP Top 10 Threat Category that describes a finding, including "Unknown" for findings which category does not belong to OWAP Top 10. This metric supports awareness, training, and proactive protection efforts.

Allowed values:

- Injection
- Broken authentication
- Sensitive data exposure
- XML external entities (XXE)
- Broken access control
- Security misconfigurations
- Cross-site scripting (XSS)
- Insecure deserialization
- Using components with known vulnerabilities
- Insufficient logging and monitoring
- Unknown

Over time, this metric should show a downward trend in each category.

Developers	Managers	Executive leadership
Helps identify common strategies to avoid specific security threats or even entire classes of security threats.	Helps identify training opportunities.	Helps identify risk and exposure.

Threat Vector Details

A threat vector is a path used by a malicious actor to exploit a vulnerability. Threat vectors can be extracted from the details provided by the tool.

Developers	Managers	Executive leadership
This metric helps identify remediation efforts needed, fine-tune the security controls in the SDLC to support early detection and decide priorities.	This metric helps identify remediation strategy needs and decide the best course of action.	This metric helps executives understand attack surface and potential risks to the business. For example, Threat Vector information or metrics could be used to identify security control/capability gaps, which in turn can be used to prioritize near-term or future resource/spend decision for the organization.

3rd Party Library/Framework (Software Composition Analysis)

This metric will have the name of the vulnerable library detected by the Software Composition Analysis (SCA) Tool and helps understand the use of each library per project, program, division, etc. This criterion will help the organization quickly identify the risk exposure after any industry security advisory.

Developers	Managers	Executive leadership
This metric helps understand vulnerable libraries needing a patch.	This metric helps with the identification of library usage and remediation efforts. It also provides managers with information on license agreements compliance.	Executive leadership will use this metric to identify risks associated with security advisories.

Process Compliance Metrics

Security Defect Density

This metric calculates the number of security defects found in every thousand (millions, depending on the codebase) lines of code. Over time, this metric should show a downward trend.

Developers	Managers	Executive leadership
Helps to perform a self-assessment of the software crafting process.	Helps evaluate the security practices implemented.	Helps evaluate cost and business impact associated with defects, understand skills gaps, or process issues.

Flaw Creation Rate

This metric measures the rate at which vulnerabilities are created as detected by security tool automation within the SDLC process. This metric should be compared against the average time to resolve a security flaw. Over time, this metric should be lower than the Average Resolution Time.

Developers	Managers	Executive leadership
Helps to perform a self-assessment of the software crafting process.	Helps evaluate the security practices implemented and identify training opportunities.	Helps detect trends and understand the effectiveness of the organization's application security program

Average Resolution Time

This metric is calculated on the time elapsed between a vulnerability appearance and the point where this vulnerability is flagged as 'Solved'. Over time, this metric should show lower than the established SLA.

Developers	Managers	Executive leadership
Helps to perform a self-assessment of the software crafting process.	Helps evaluate the security practices implemented and identify training opportunities.	Validates SLA compliance.

Organizational Compliance Metrics

Window of Exposure

Measures how much time it takes for the organization to remediate a vulnerability from the moment it gets disclosed, to the moment the vulnerability is patched. Software Security Automation plays a vital role in the identification process. Over time, this metric should show a downward trend.

Developers	Managers	Executive leadership
Helps to perform a self-assessment of the software crafting process.	Helps evaluate the security practices implemented and identify training opportunities.	Helps evaluate the security practices implemented and the organization's security posture.

Program Participation Level

This metric measures the participation within the organization including how many business units/projects have full/partial/no software automation in place. Over time, this metric should show an upward trend toward full coverage.

Developers	Managers	Executive leadership
N/A	N/A	Provides a picture of the application security program's maturity level.

Root Cause Analysis

This metrics measures whether the vulnerability is associated with third-party libraries vs custom code. Over time, this metric should show a downward trend in both cases.

Developers	Managers	Executive leadership
Helps identify common strategies to avoid specific security threats or even entire classes of security threats.	This metric should help understand where to focus investments decision over time.	N/A

SDLC Phase Detection

Measures in which phase of the SDLC the most vulnerabilities (higher percentage of issues) originate. Over time, this metric should show a lower number of findings in the SDLC stages.

Developers	Managers	Executive leadership
This metric helps identify remediation efforts needed, fine-tune the security controls in the SDLC to support early detection and determine priorities.	This metric should help understand where to focus investments decision over time.	Provides a picture of the application security program's maturity level.

Establish a Data Warehouse structure to store metrics

To fully support a metrics architecture that will be able to serve the different audiences, the design of the data warehouse storing the data to support decision-making becomes an important aspect of the data aggregation strategy. The security controls are going to be generating information that is important to help each role understand the state of the company's enterprise. Developers may need to understand the type of security vulnerability found in each project to plan for proper remediation. Managers may need to understand which projects are using a specific vulnerable component so they can initiate a communication process that will lead to remediation efforts. Executive leadership may be interested in correlation and trends associated with the high severity findings. As illustrated, even though the different roles may have different interests, all those concerns can be served using the same data collected from the security controls.

Designing the data store to support the metrics strategy requires the identification of key pieces of information which will help support the analysis process. Understanding a metric requires the use of context. For example, how many high severity findings were identified last month vs a year ago. The measure # of severity findings will be evaluated using a date context (last month, a year ago). Table 1 provides an example of those pieces of information which will be used in the paper to guide the implementation discussion of the metrics.

Table 1

Metrics Data Warehouse - Key Components

Data	Comments
Severity (Type and Level)	Helps with the analysis associated with severities.
Date	Provides a point-in-time contextual aspect of security findings that will help evaluate trends, contrast specific periods (i.e., last vs current month), etc.
Scan Tool	Help provide visibility regarding the efficacy of the security controls in the SDLC. When having all the recommended security controls (SAST, SCA, DAST, IAST, etc.) in place, reviewing the findings per each control could help identify efficacy levels. For example, a high number of Cross-Site Scripting security findings in the DAST may indicate a gap in the SAST coverage.
Threat Vector Details	CVE information, OWASP categorization, etc. will help understand the type of risk associated with the findings.
Organizational Structure	Help understand the maturity level of security in the software being created by the different teams in the organization.

When designing the data warehouse, depending on the approach selected, the layout of the database schema may require the data to be segregated in different data structures or tables. If the implementation requires an OLAP approach, a star schema will hold the measure values in the center stage table, while the information providing context for the analysis would remain in the tables surrounding the measures table. Figure 2 provides an example of the star schema in an OLAP implementation.

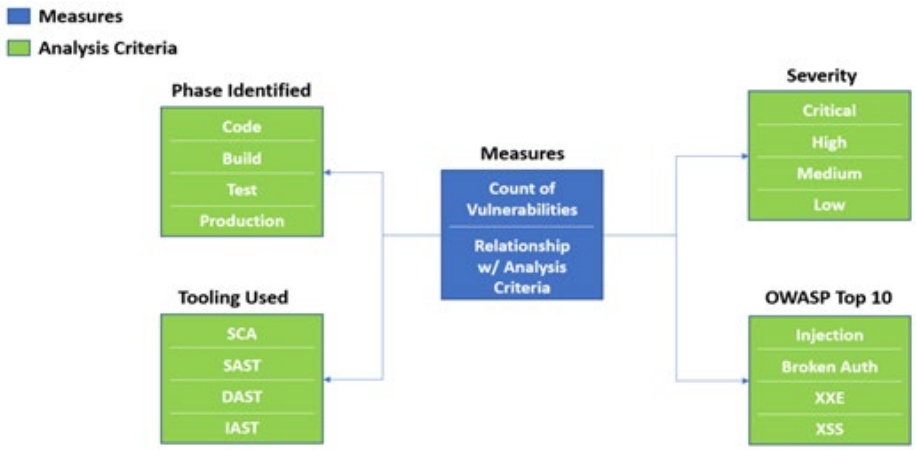

Figure 2 OLAP Star Schema

For the sake of facilitating the discussion, this paper will use a star schema approach for the data warehouse design. The database design will contain a measures table that will hold the metrics we are going to evaluate. Since we are going to be evaluating metrics associated with security findings from the security controls in the SDLC, our measures table needs to be able to store information about each security finding as well as the relationship of the analysis criteria we will use to evaluate the results. For example, to count the number of high severity findings in a report, the measures table for the security findings needs to store a row for each of the security findings. Each row will need to relate to the different analysis criteria needed to support the organization's metrics strategy. Using the information from Table 1, we can quickly derive a relationship between the measures and the analysis criteria. Figure 3 provides a breakdown of the measures table and the different columns required.

Measure Row Id
OwaspTop10Key
ScanResultDateKey
ThreatDetailsKey
SeverityKey

Figure 3 Relationship between the measures table and the analysis criteria tables

When evaluating the structure of the measures table, it becomes clear that to support a star schema there is a need to maintain a close relationship between each measure and the different analysis criteria tables. Each column ending with "Key" represents a foreign key to a table that will hold the unique values to use when trying to segment the information for the analysis. For example, the table presented in Figure 4 provides the layout for the Severity analysis criteria table. The layout of the analysis table is very simple, one column for the key, another column to hold the description that will be used to help with the visualization of the metrics. By creating this link between the measures table and the analysis criteria table, any visualization component can take advantage of the key relationship and help relate the different criteria that can be used to segment the analysis.

SeverityKey	Description
1	High
2	Medium
3	Low

Figure 4 Analysis Criteria table for Severity

Once the different data tables are created to hold the measures information as well as the analysis criteria tables, the organization will be ready to start working with the process that will perform the data collection and will populate all the required data in both the measures tables as well as the analysis criteria tables.

Establish a process to load the data

One of the important aspects of having access to the data from the different security controls in the SDLC is the data load process. The data load into the data warehouse is an important aspect of supporting metrics and analysis. To take advantage of the security controls, organizations need to select tools centralizing the data needed for the metrics. Security controls centralizing their data capture as the development teams run the scans during the DevOps pipeline stages offer a better mechanism for any integration effort through the use of their API services. The exposure of data through API services by security controls provides a great opportunity to leverage automation activities which will remove human friction from the process, creating efficiencies along the way.

There are instances where security controls working in standalone scenarios may lack API service exposure. These scenarios will create friction towards any automation initiative as the human factor will still be a requirement in the generation of the data in a format needed for the data load process. When organizations are evaluating the acquisition of security controls to support better application security, considering the impact associated with the lack of integration

capabilities such as the lack of API service must be at the forefront of the evaluation process. The lack of this type of integration through API services will hamper the ability of the organization to create efficiencies as well as affecting the maturity of automation initiatives.

In those instances where the organization has no short-term plan to replace their standalone security controls and metrics are an important goal, teams can still partially automate the data load. Assuming a security control working in standalone mode, the organization can still create processes to monitor a specific folder where the scan results can be dropped. From there, the process will take care of the data load. Even though this is not a scalable solution, it provides some level of feasibility while the organization transitions into security controls providing API services to access the required data.

As the organization prepares itself for the data load automation through the API services, two aspects of the implementation become important considerations:

- Mapping of the data
- Frequency

The mapping of the data will help define, for data structures associated with each security control, which fields will be mapped to the data warehouse tables and which fields will be needed to extract from mapping tables. For example, security control data could be missing information regarding the relationship with division, business unit and program. Here, the use of

a cross-reference table mapping the project with those pieces of information will help automation activities identify which key values should be included in the data warehouse. Hence, the data capture process to support metrics will become an exercise of mixing native data included in the results from the scans of each security control and the cross-reference tables that will help enrich the analysis and the metrics.

As an automated activity, the data load can be performing a data pull regularly. From a frequency standpoint, defining how often do we need to refresh the data in the data warehouse will determine the frequency needed to perform the pull of information. For instance, an organization may decide it will be beneficial to evaluate metrics every day versus every couple of weeks. This distinction will clearly define the strategy implemented in regard to how often the data capture is performed. The definition of the frequency will not have any effect on the process needed to perform the data pull, as it becomes a consideration on how often we repeat the data capture exercise. Once the data has been captured in a data warehouse structure, the organization will be ready to define the visualization aspect of the metrics.

Establish a process to visualize the data

Now that we have defined which metrics can be captured from the security controls, we have designed a data warehouse to store the metrics data as well as the data load automation that will ensure the data store holds the right pieces of information, the next logical step is to define the metrics visualization. There are several strategies organizations can use to implement the

visualization. There are some tools in the industry that help organizations to visualize the metrics, there are multiple approaches that could be used by organizations:

- Dashboard
- Scorecard
- Metrics Report

Organizations need to define how metrics will be exposed to facilitate the analysis process. The first option is through a metrics dashboard. A dashboard provides a consolidated view for the organization to logically present information in a centralized structure which will help measure those critical aspects of the application security domain. Understanding how the various needs of different teams (for example, manager concerns vs developers) is important, as it drives the discussion on which pieces of data to include in the dashboard for the respective target audiences and which key performance indicators are needed. Defining the data requirements per each role will help the implementation team understand which pieces of information will be included in the dashboard. For example, a developer dashboard can combine the following data components:

- Severity
- Scan Tool
- OWASP Top 10 Threats
- Vulnerable Libraries

Figure 3 provides an example of how to integrate these pieces of information to help developers with their respective concerns.

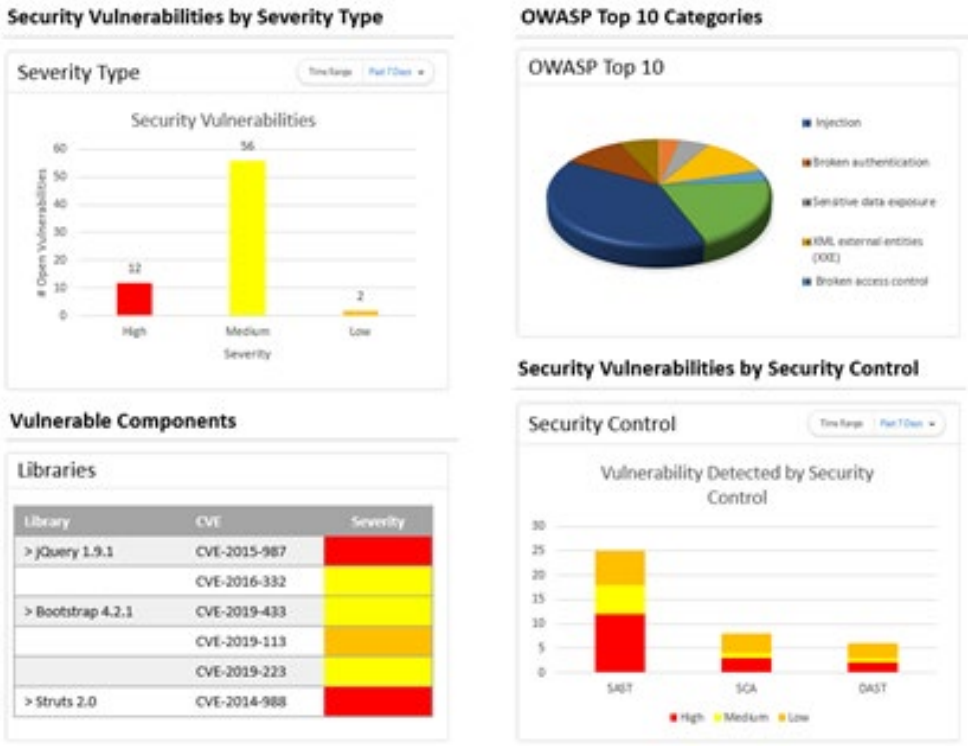

Figure 3 Developer Metrics Dashboard Example

The implementation of dashboards enables organizations to enhance the visibility associated with the security findings identified by the security controls in the SDLC. One key feature that organizations must include in the use of dashboards is the drill-down capability. Having drill-down capabilities will allow each role to move from the generic to the specific. Any tool selected by the organization to create a dashboard and present metrics should provide drill-down for better decision-making and timesaving efficiencies.

Another visualization strategy organization can implement to help with the evaluation and alignment of the metrics are scorecards. The use of scorecards allows an organization to align the security organizational goals as it evaluates the metrics captured against pre-established goals. Scorecards become a great asset to help visualize trends, contrasting the current state of application security against expectations, and aligning the operational improvements needed to improve the overall application security posture. Even though the benefits of the scorecard span across the organization, we must acknowledge the positive impact against organizational compliance as this will benefit the most from the implementation of a scorecard visualization tool. Figure 4 provides an example of a simple scorecard capturing the distribution of security vulnerabilities by severity while offering visibility regarding the trend.

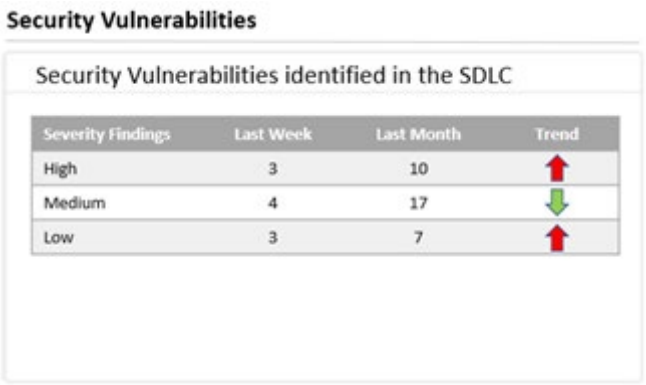

Figure 4 Scorecard representing Security Vulnerability by Severity

One final strategy which could be used by organizations to visualize the metrics is using reports. Implementing reports for the different roles defined in this paper should combine different metrics elements (key performance indicators, trending, etc.) to ensure they serve as a vehicle to measure security goals, objectives, and risk. The details provided in the report should avoid

being too detailed or vague as it may hamper the ability to provide decisive information. Technical details such as threat vector details should be reserved for development teams while more high-level information should be included to help the organization evaluate its compliance level and risk. Having the right data included in the reports will ensure effective oversight of the overall application security posture.

Communication Plan

At this point in our process, we have determined the key metrics and how to obtain them from our tooling. Now our next step will be determining how to communicate those metrics to our customers. First, we will determine our target audiences. In this paper, we are using the following audiences:

- Information Security leadership
- Development Leadership
- Development Teams

Each of these audiences will have different metrics to be communicated, either in the scope of the data or the specific metrics. As an example, overall code coverage is a useful metric for Security Leadership as it lets them see how much of the code is covered, but a Dev Lead will be more interested in how much of their code is covered, concerned only with their scope. Likewise, Security Leadership is unlikely to care much about the operational or technical metrics that a Development team would like to see around time spent scanning.

Next, consider what the appropriate presentation methods for these audiences may be. Some example methods will be:

- Automated Web Dashboards
- E-mailed Reports
- Slide Decks
- Auto-generated PDF

Choosing the right presentation for your data is vital for communicating effectively. Users may skip over regularly e-mailed reports, even if they contain critical information, as the regular appearance desensitizes users from the potential impacts of the contained vulnerabilities. Likewise, automated web dashboards may not be regularly visited if not part of a user's routine. Consider what makes the most sense for each of your audiences. For our purposes we will use the following pairings:

- Automated Web Dashboards – Development Teams
- E-mailed Reports – Development Leadership
- Slide Decks – Information Security Leadership

We have chosen the automated dashboards for the development teams as they will most likely need the most up-to-date information and will likely visit at irregular times, so the most real-time and available form makes sense for them. Likewise, we chose e-mailed reports for

development leadership as a regular communication mechanism from Security primarily to keep engagement focused.

Finally, we selected a manual process for our leadership for a few reasons. First, our leadership is likely to need to present these metrics in multiple different situations and the slide deck format makes the metrics portable across executive communications. Second, it allows us to review the metrics before leadership consuming them, enabling an opportunity to understand the metrics more and be better prepared to answer any questions or build a story around what the metrics are showing.

An important thing to consider is that the usefulness of metrics also changes – consider meeting with your audiences on a regular (quarterly, annual, etc.) basis to gather their feedback and see if they have additional use cases or metrics requests from you.

Built-In Visualization Tools vs Custom Approach

There are instances where organizations already possess data visualization tools. This scenario presents the best course of action as no additional effort for the implementation of the data visualization tool may be necessary. Since the software is already available, it becomes a matter of engaging with the resources who already know how to use the tool to facilitate the time to market of the implementation of the metrics. From an efficiency standpoint, this scenario

enables the organization to leverage existing knowledge on how to use the data visualization tool without having to spend any time in learning activities.

Some of the built-in visualization tools provide multiple options. These tools allow teams to create different types of charts, use drag-and-drop capabilities to create effective visualizations and integrate natively with common solutions used in the industry for collaboration. These native integrations support the creation of efficiencies as less time is required trying to integrate the metrics with tools commonly used by the organization at different levels.

Despite these benefits, some organizations may lack visualization tools to fully implement the metrics platform. If this is the case, a custom implementation approach may be required. Some database platforms in the industry already include as part of their licensing all the tools needed to define a data warehouse, create the data collection process as well as define and implement the reports to use when visualizing metrics. This is a scenario where organizations can leverage all features available and maximize the use of the software while maintaining a cost-effective approach. From an implementation standpoint, learning curves will be a key concern as there would be a need to have resources with the knowledge to help implement the data collection process as well as the reports. If no formal reporting platform exists, leveraging existing capabilities in the database will require proper architecture and deployment to ensure long-term sustainability.

Conclusion

Establishing a metrics program for application security requires the identification of the measures that will enable not only the developers but also the organization to understand the risk and compliance level associated with the software delivery process and their security stance. This paper provides a roadmap for the use of the scan results provided by the security controls implemented in the SDLC which will help identify threat vectors in code, vulnerable components, and malicious behaviors. Understanding and defining the metrics that are important to the organization will help with the design of the data store that will be used to capture those pieces of information enabling better visibility and planning.

Automation can be integrated as part of the data collection process using API services exposed by the different security control platforms. The data collection automation will help capture not only the measures but also the analysis criteria to be used during the data visualization. Data visualization components such as dashboards, scorecards and reports will help structure the information in a meaningful way to ensure the analysis process can support better decision-making while providing operational efficiencies. Finally, communicating the metrics throughout the organization becomes key as the metrics programs will ensure better compliance levels and stronger security posture in the software delivery pipeline.

References

Choi, Y. M. (n.d.). Intro to OLAP: On-Line Analytical Processing.

http://www.cis.drexel.edu/faculty/song/courses/info%20607/tutorial_OLAP/definition.htm