Security Culture 1.1

Nick Miller

Contents

Frontispiece

Copyright and Licensee

Copyright (c) 2024 The OWASP Foundation.

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License. Please read and understand the license and copyright conditions.

Leaders

- Nick Miller

Reviewers

- Daniel Burn
- John Lowe

Introduction

This guide will discuss the importance and benefits of establishing a security culture when building an application security program. Security should be considered at each stage of the Software Development LifeCycle (SDLC), helping to create secure development practices.

This guide can be used by a broad range of roles: security; developers; testers and team leaders. The guide does not assume a particular SDLC, it can be used for any SDLC and software development approach.

This guide will introduce ideas and activities to help uplift security, such as threat modelling, but does not seek to be an authoritative source for these activities. References to more comprehensive material on these activities will instead be provided.

The first step to building an application security program is defining a **maturity goal** which sets a target to be measured against. To be able to meet the defined maturity goal, the appropriate resources need to be assigned. This requires having management buyin. The importance of security needs to be understood by roles from the development team manager to the product manager to ensure that security work can be prioritised accordingly.

Collaboration between the security team and development teams is essential. The security team expects the development team to write secure code, but they need to provide guidance on application security. With security team collaboration, security can be distributed across the development team.

A good way to distribute security across development teams is by using ”**Security Champions**”. A Security Champion acts as a point of contact for the developers to ask security questions. This helps scale out security, as there are often fewer security team members than development team members.

Activities help build the security mindset into development teams. Interactive labs and vulnerable by design applications help developers learn different security vulnerabilities and how to fix them. Capture the Flag (CTF) events and gamification are interesting and engaging activities to help build security.

Threat modelling is a useful task for developers to use to identify potential security issues early in the development lifecycle. This task need not be only performed by security architects, but can be done by developers in another way to scale out security.

Security testing is used to ensure code delivered is secure. However, tests can be added at multiple stages of the development lifecycle, not just the end. It is beneficial to involve the development team in the deployment of the security testing tools used. Ensure that the tool will provide value and have minimal false positives. It is also beneficial to have penetration testing reports that are provided to the development team be free from false positives and give details on how to remediate particular findings.

Lastly, the use of **metrics** is important to measure progress of the security uplift. Metrics from the Security Champions program, threat modelling, code review, security testing, and other activities are all useful to give an indication of how the application security uplift is improving over time. This allows adjustments to be made to the overall program to ensure the defined maturity goals are met.

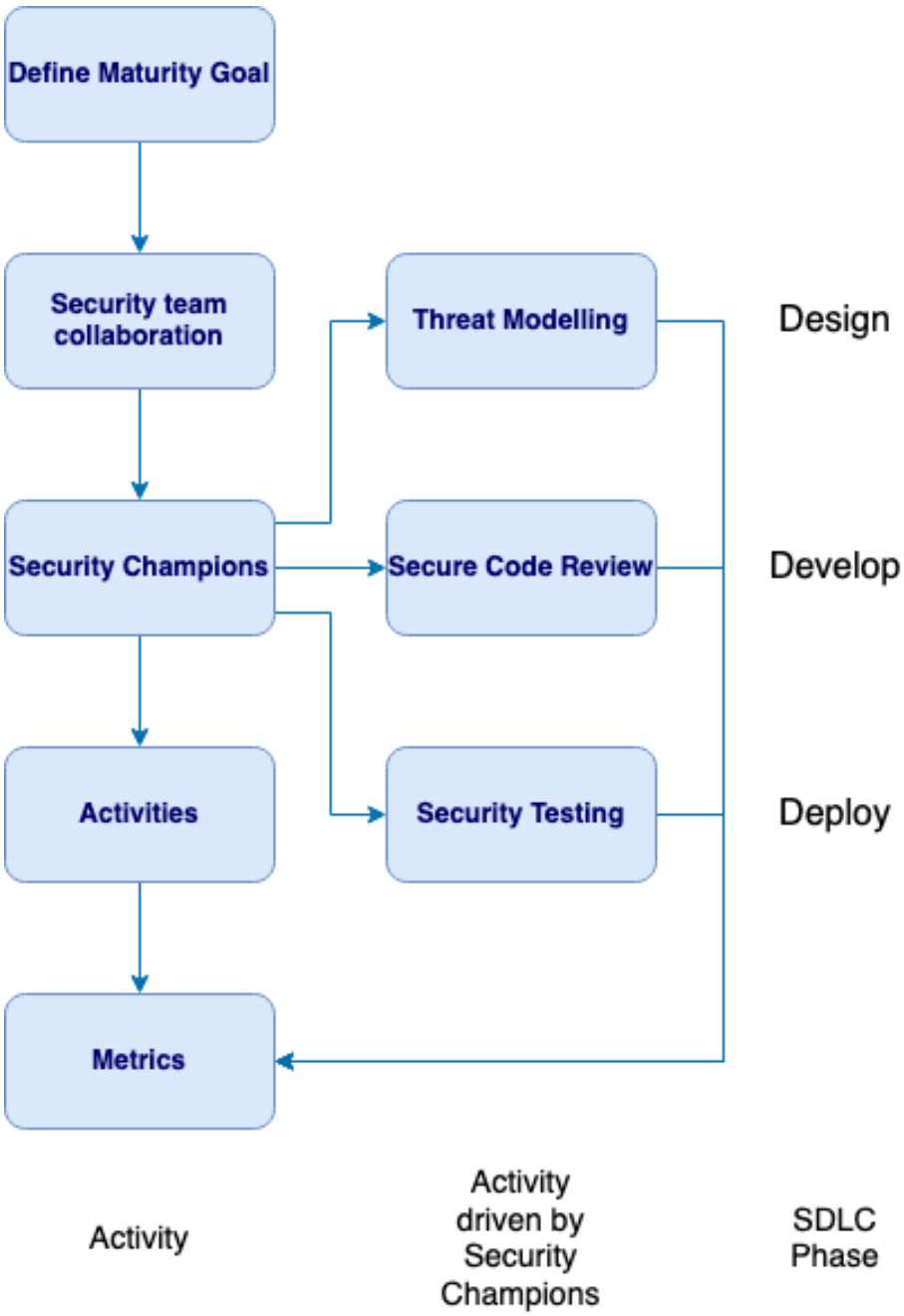

Figure 1-1: Security Activities Flow Chart

Why adding security in development teams is important

Security should not just be an afterthought, something bolted on at the end. It is important to add security early in the Software Development Lifecycle (SDLC). The earlier security is added into the project, the fewer vulnerabilities that will need to be fixed in later stages of the SDLC. The cost of fixing security issues increases the further along it is in the SDLC[1].

Rather than only thinking about security at the end of the SDLC, such as with a penetration test on delivered software, instead security should be added at each stage. During the Design phase, threat modelling can help identify possible security issues with a proposed solution architecture. During the Develop phase, secure code reviews can help find any vulnerabilities in the source code. During the Deploy phase security testing, such as automated security tests and penetration testing, can validate that the deployment is secure.

Figure 2-1: Security Activties by SDLC Phase

Developers are most appropriate to be responsible for application security, as they are the ones writing the code[2]. Developers have the most understanding of the locations of possible security issues. However, developers have varying degrees of security knowledge and experience. It is then essential to add security into the development process and ensure development team members have the skills necessary to deliver secure software. Developers can refer to the OWASP Developer Guide, a comprehensive reference guide to the areas of application security.

[1] OWASP. *Web Security Testing Guide 4.2.* 2020. *https://github.com/OWASP/wstg/releases/download/v4.2/wstg-v4.2.pdf*

[2] Ibid., 5.

Goal setting and security team collaboration

Define maturity goal

The first step to building an application security program is defining a maturity goal. Use a capability maturity model to measure current security knowledge and practices, and plan for an uplift. The particular goals that are set will depend on any security framework or standard that an organisation must comply with or aspires to achieve certification in. Use the OWASP Software Assurance Maturity Model (SAMM) to define the current state of application security for an organisation, and provide guidance on improving. Once the maturity model is created the appropriate resources should be assigned to be able to meet the set goals.

Following the SAMM guide, first the relevant stakeholders are interviewed to understand the current security state of practices, with a maturity level assigned for each practice. Once the current maturity level has been determined, the next step is to define a maturity target to implement. Planning utilises a roadmap involving multiple phases that have particular targets set for each phase. The roadmap is then communicated across the organisation and to management stakeholders. As the plan is implemented, the effectiveness of the improvements is measured.

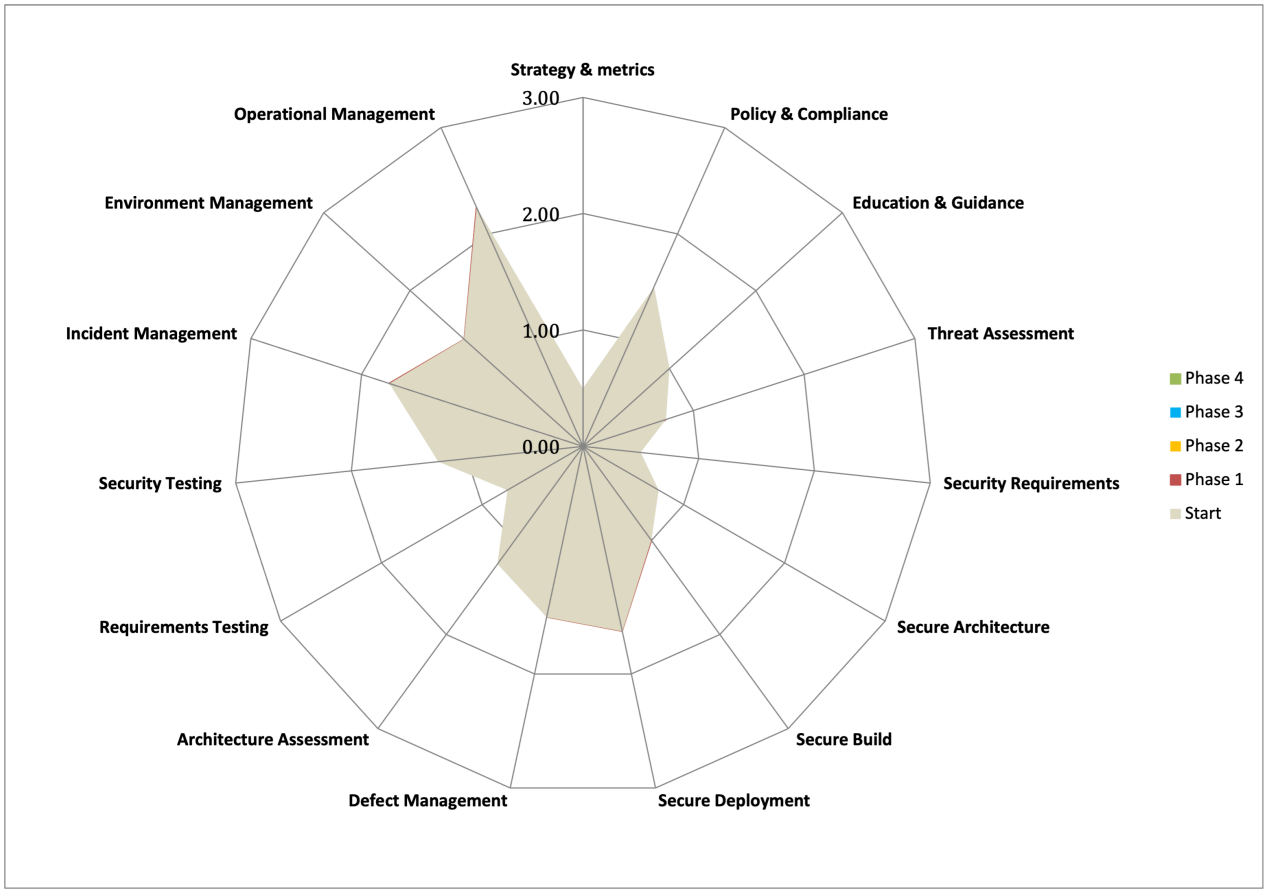

Figure 3-1: Maturity Level Graph

Management buyin and business process

When planning to uplift security it is important to have management buyin. Having the idea of security as a priority pushed down from the top of the organisation helps to legitimise the need for security and have it at the forefront of the minds of all organisation members. Roles representing the business, such as product owners should also have buyin. Introducing security

awareness at all levels of the organisation helps work to be prioritised. Development team managers and product owners will understand security is important and allocate resources to it.

Security work should be visible and be subject to normal business processes. By having security work use the same process, providing work hour estimates and including it in the roadmap alongside other work, it will ensure that security work is not forgotten about and is assigned the appropriate resources. The amount of security work to include in a roadmap will depend on the organisation's risk appetite. A financial organisation will be risk averse and require a significant investment in security work.

It is useful to have a risk register to track outstanding security risks. These are reviewed periodically by senior management to accept or mitigate as per the organisation's risk appetite.

Security and development teams working together

It is important for the security and development teams to work together. It can often be the case that teams are compartmentalised, working in silos. Instead, what is needed is collaboration between security and development teams. The development team may want to ensure they include security in their process, but do not know where to start and would benefit from guidance from the security team. Whereas the security team wants the development team to write secure code, but they may not have time to dedicate to application security. To address this issue, there needs to be collaboration between the security and development teams, as well as security distributed across the development team.

Security teams are often seen as a blocker. Security teams want to ensure systems are protected, but there is often a tradeoff between security and convenience. Security teams may default to disallowing a developer requested functionality. Effective security teams enable, rather than block[3].

It is important to keep an open communication channel between the security team and development team. This could be by having regular meetings where application security is discussed, with both teams able to contribute to the discussion. If your organisation uses an instant messaging tool to communicate, this can be an engaging, visible way to discuss security for both teams.

A security team member contributing code to a development team codebase can help build a good relationship between the two teams. This builds trust by showing the development team that the security team does have an understanding of writing code. In addition, showing how a vulnerability can be exploited in the context of an application that the developers work on can help application security seem relevant and useful. For example, a security team member may find a Cross-Site Scripting vulnerability in the developers' codebase and demonstrate an exploit. To remediate the vulnerability, they select a particular library to escape the html output that can be reused by the development team for this particular issue in future.

The security team should assign an appropriate priority rating when assigning work to a development team, rather than marking every item as urgent. Performing a risk assessment and assigning an appropriate risk rating is important. A risk rating considers the vulnerability and the impact it would have on the business. This can be calculated, for example, by using the OWASP Risk Rating Methodology. Having a risk rating helps the teams understand which

[3] Rich Smith, *Agile Application Security*, (USA: O'Reilly, 2017), 318.

items pose the most risk. When the development team understands how the risk ratings are calculated and the impact a particular vulnerability has on the business, it can be easier for the security team to have the attention of the development team in the prioritisation of the vulnerabilities remediation.

Use secure defaults and secure guard rails or paved roads as an effective way to ensure security is built into the development process[4]. Infrastructure used by developers should be set up in a secure manner, such that no additional work is needed to make the environment secure. Any modules that developers use should be secure by default. Rather than provide all options to the development team which may include insecure options, make available only the secure options that follow the organisation's security policy. Sample secure code or complete secure modules can be provided to developers to make it easier for them to integrate into their implementation. This makes the easy way to do something also the most secure way.

Security teams can ensure new solution architecture designs are secure by providing developers with sample solution architecture that has the required security controls in place. Security policies help guide developers in a secure design, but more beneficial is for the security team to provide a solution architecture, and ideally a sample application or code, that includes all the security controls they expect the developer to use. A developer can then simply take a sample application or solution architecture and incorporate it in a new solution, and the security team can be assured that the security controls they require will be in place.[5]

The speed at which security is introduced into the development team will depend on how much time is available to them and their current knowledge and interest. It is important for management to set aside the appropriate time for developers to focus on learning security, and for the security team to provide interesting relevant learning material. Improving security is a gradual process that needs maintenance and attention.

[4] OWASP, How to start an AppSec program with the OWASP Top 10. 2021. *https://owasp.org/Top10/A00_2021-How_to_start_an_AppSec_program_with_the_OWASP_Top_10/#stage-2-plan-for-a-paved-road-secure-development-lifecycle*

[5] Anunay Bhatt, Uncomplicate Security for Developers using Reference Architectures. 2021. *https://ab-lumos.medium.com/embedding-security-into-sdlc-using-reference-architectures-for-developers-29403c00fb3d*

Security Champions

It can be hard to introduce security across the development team using the security team alone. Information Security people do not scale across teams of developers. A good way to scale security and distribute security across the development teams is by using Security Champions.

Security Champions act as a single point of contact in regards to security for their team. The Security Champion is in contact with a member of the security team to ask for guidance. The Security Champion can monitor security best practices, and help the security team run and promote security uplift activities such as a CTF (Capture the Flag) event.

Security Champions help to improve the communication between the development teams and the security team. A Security Champion will know the pain points of their teammates' code bases and culture, they are then in a good position to present security in a way that directly connects with them. Security Champions are a key element of improving application security maturity as referenced in maturity frameworks such as OWASP SAMM.

It is important to have management buyin for the implementation of the Security Champions program. A Security Champion will need to have a percentage of time set aside from their regular position to spend on fulfilling this new role. A Security Champion will dedicate time to security initiatives, such that they have less time to spend on development. This time spent on security is worthwhile when weighed against the cost of not spending time on security, resulting in costly security issues to fix. When planning on implementing a Security Champions program, it may be beneficial to start with a proof of concept consisting of one or two Security Champions. When the program has been proven successful it can be rolled out across all development teams.

A large organisation that has a mature Security Champions program may next look towards establishing a Community of Practice for security.

The following section will describe the steps to build a Security Champions program as defined in the OWASP Security Champions Playbook. See also the OWASP Security Champions Guidebook that describes key principles for a successful Security Champions program.

1. Identify teams

Identify the teams involved that you want to add Security Champions to and how they work.

Identify team structure: In the initial step, the organisational structure is mapped to understand the teams that will be involved and their structure. For example, a team may consist of 5 engineers, 1 product owner, 1 QA and 1 team lead.

Identify technologies used: Identify what technologies are used in each team, such as programming language and framework.

Identify current security state: understand the existing security posture, so as to establish a baseline to use to measure improvement. This could describe any code reviews that are performed or automated security testing implemented.

Identify processes: Identify any existing communication channels, and what process is used to report a security issue and assign it to a team member.

2. Define the role

Define the developer participation requirements as a Security Champion, and clearly define the role. The responsibilities for a Security Champion depend on what the organisation's application security goals are and what the security team would like to embed into the Security Champion program. Refer to the identified current security state and detail what you would like to improve.

The Security Champion role may include:

Evangelise security: promoting security best practice in their team, imparting security knowledge and helping to uplift security awareness in their team

Contribute to security standards: provide input into organisational security standards and procedures

Help run activities: promote and run activities such as Capture the Flag (CTF) events, see Activities chapter

Conduct threat modelling: threat modelling consists of a security review on a product in the design phase, see Threat modelling chapter

Perform secure code reviews: raise issues of risk in the code base

Use security testing tools: conduct or verify automated security scanning and provide support to their team for the use of security testing tools, see Security testing chapter

See examples of what roles can contain at Security Champions session, OWASP summit 2017.

3. Nominate Champions

A Security Champion may be a developer, operations or QA role. Security Champions should be nominated, rather than assigned.

Management buyin: get agreement from management on the defined role responsibilities and time commitment of a Security Champion, such as 20% of their role.

Scale: 1 security staff member may support 5 Security Champions, who each support their team.

Assign to relevant team: It is useful to add Security Champions by technology platform, so that their specific technical skills can be used. For example, frontend; backend; mobile.

Converse with team manager and nominated team member: discuss with the team manager about possible Security Champion candidates. Nominations may come from the team manager as well as having a call for interested candidates. Then discuss with the candidates about the role and its benefits to see if they are a good fit. Discuss the benefits of learning application security and how it can help their career and stand out.

Present the Security Champions to the organisation: once the champions have been nominated, communicate the Security Champion program and its members to the organisation.

4. Set up communication channels

Establish a communication channel for all Security Champion members and their security team contact and organise periodic meetings.

Establish communication channel: create a communication channel that the Security Champions and their security contact can use. This can be whatever is appropriate for the organisation, such as a messaging chat application channel or email group.

Organise periodic meetings: periodic meetings are useful to discuss with the Security Champions on any issues they are having, and adjust goals as needed.

5. Build solid knowledge base

Establishing a knowledge base can help to have a single place to refer to for application security guidance.

A knowledge base may contain:

Organisation's security policy and procedures: links to relevant application security procedures such as the password policy, which cryptographic algorithms to use, and secure development procedures like how to conduct a code review or a particular library to use to escape output.

Reference material: useful external reference material can include OWASP Code Review Guide and OWASP Cheat Sheets. Security checklists can also be useful, such as OWASP Top Ten and OWASP Application Security Verification Standard (ASVS).

Creating a training program for Security Champions is also useful for building knowledge and skills in application security. The training format can range from references in the knowledge base to hands on labs. A training program can consist of different levels, each building on the previous one. Initial levels can contain broad security topics and expand out to specific platforms, such as mobile security.

6. Maintain interest

Keep the Security Champions motivated and engaged to ensure they can continue to enthusiastically promote security in their assigned team. Use activities such as Capture the Flag events to promote security. Recognise the work done by Security Champions and communicate their achievements to the wider organisation.

Activities: Security Champions can attend conferences together and a conference calendar can be created that contains relevant upcoming conferences. Security Champions can organise events such as Capture the Flag (CTF) and hackathons. Gamification can be used to help drive engagement - such as the use of a scoreboard during a CTF event, and the awarding of prizes like branded clothing or stickers. See Activities chapter.

Recognise and reward: It is important to recognise the achievements by the Security Champions, such as providing updates on progress in regular newsletters. Gamification can be used to indicate progress for the Security Champion, through the use of levels. This can help the Security Champion to have something to work towards and mark their success in their security journey. An organisation may formalise a Security Champion position with their HR department to help motivate individuals to join.

Measure progress: Security Champions can have goals to achieve, such as security Objectives and Key Results (OKR). For example, to close a percentage of critical and high vulnerabilities; conduct a certain number of threat modelling activities; complete security training modules. Defined maturity levels can be used to mark progress of individual Security Champions and the overall program. For example OWASP Top 10 Maturity Categories for Security Champions.

Activities

Use activities to build security skills and knowledge. Activities also help maintain enthusiasm and interest in security. This chapter will discuss different types of activities that can be used. Activities must be relevant and targeted to the particular developers and development teams.

Security teams can start by first **understanding the developers' security knowledge and experience**, and then teach the required security skills using appropriate engaging activities. Initially, a simple quiz asking developers for their familiarity with application security such as the OWASP Top Ten can be useful. The skills to focus on will draw from the maturity goals set.

Make Learning material relevant. Learning material should target the developers' relevant programming languages and platform. There may be some security aspects that are more applicable for a frontend developer compared to a backend developer. It is also important to know the individual's current skill level, to be able to provide learning material that is appropriate.

Provide learning material in **different formats**, as individual people learn in different ways. Some may prefer reading to videos. Learning material that includes interactive labs can be an engaging and interesting way to develop hands on application security skills.

The type of activities selected should also be **targeted** to the individual developers and development teams. Some may enjoy interactive hackathons and secure coding labs, while others may prefer audio and video shows.

Developers may find it hard to find somewhere to start learning security. Helping developers learn security need not be difficult and extensively time consuming. Short, practical, targeted learning activities can be a good way to build security knowledge. Starting with the OWASP Top Ten which addresses commonly found vulnerabilities, can be a good approach.

Security conferences

Application security conferences are a good way for developers to learn from security professionals. Presentations can include secure coding practices, new security vulnerabilities and new security tools.

Resources:

- OWASP Events Page

Security interest groups

Local application security groups can provide regular meetings for developers to meet and learn from each other.

Resources:

- OWASP Local Chapters

Security shows

Audio and video security shows provide regular presentations on various aspects of application security. They may provide updates on security events or in depth analysis of a particular

area of application security.

Resources:

- OWASP DevSlop Show
- OWASP Events, Chapters and Projects on YouTube
- OWASP Podcast

Security training courses and workshops

Security training may take the form of slides presented by a speaker or self directed learning modules. It is important that training is relevant and engaging to ensure uptake and deliver the security knowledge desired.

Resources:

- OWASP Application Security Curriculum
- OWASP Secure Coding Dojo
- OWASP Training Events

Secure coding labs and pen testing labs

Interactive labs are a good way to get developers to engage with security learning material. In a pen testing lab, a developer finds and exploits a vulnerability. Secure coding labs help developers learn secure coding techniques, how to avoid common security mistakes and remediate vulnerabilities. These are targeted to a particular programming language.

Vulnerable applications

Intentionally vulnerable applications give developers hands-on experience in finding and exploiting vulnerabilities. Intentionally vulnerable applications are created for educational purposes to contain vulnerabilities that are often found in real world applications. Vulnerable applications can be used as part of a group training event or independent training.

Resources:

- OWASP Juice Shop (written in JavaScript)
- OWASP WebGoat (written in Java)
- OWASP Vulnerable Web Applications Directory

CTF (Capture the Flag) events

A CTF (Capture the Flag) event is a competitive security challenge. Individuals or teams compete in tasks such as finding vulnerabilities in an intentionally vulnerable application. The winner of the CTF event is the individual or team who has achieved the most points, as a result of finding or exploiting vulnerabilities.

Resources:

- OWASP Juice Shop vulnerable app (CTF instructions)

Hackathon

A hackathon event can be used internally for an organisation to find vulnerabilities in their applications. Similar to a bug hunt, a hackathon event dedicates time for development team members to identify possible vulnerabilities that may exist in a system's architecture or deployed code. Hackathons can be a good opportunity to identify and address security debt that has built up that teams have not had time to address during their normal operations.

Gamification

Build security knowledge in a development team in an engaging way using gamification. Developers can be awarded points or badges for completing security learning material or demonstrating security knowledge. This may provide an incentive to complete security training. Receiving a badge for an accomplishment helps mark an individual's progress in application security. A competitive points system and team leaderboard can also help to provide some motivation to engage with security learning activities.

Resources:

- OWASP Security Pins

Threat modelling

Use threat modelling to identify potential security issues early in the development lifecycle. Identifying issues early results in a reduction of vulnerabilities in a deployed system. This section will discuss why it is useful for developers to do threat modelling, give a step by step guide of the threat modelling activity, and how to introduce the activity to development teams by using gamification.

The goal of the threat modelling activity is to create the security requirements for a system. When building the system, the security requirements are incorporated, with appropriate security controls addressing identified areas of vulnerability. Security testing is later used to ensure the security controls are in place, verifying that the security requirements have been met. This gives assurance that the system is secure.

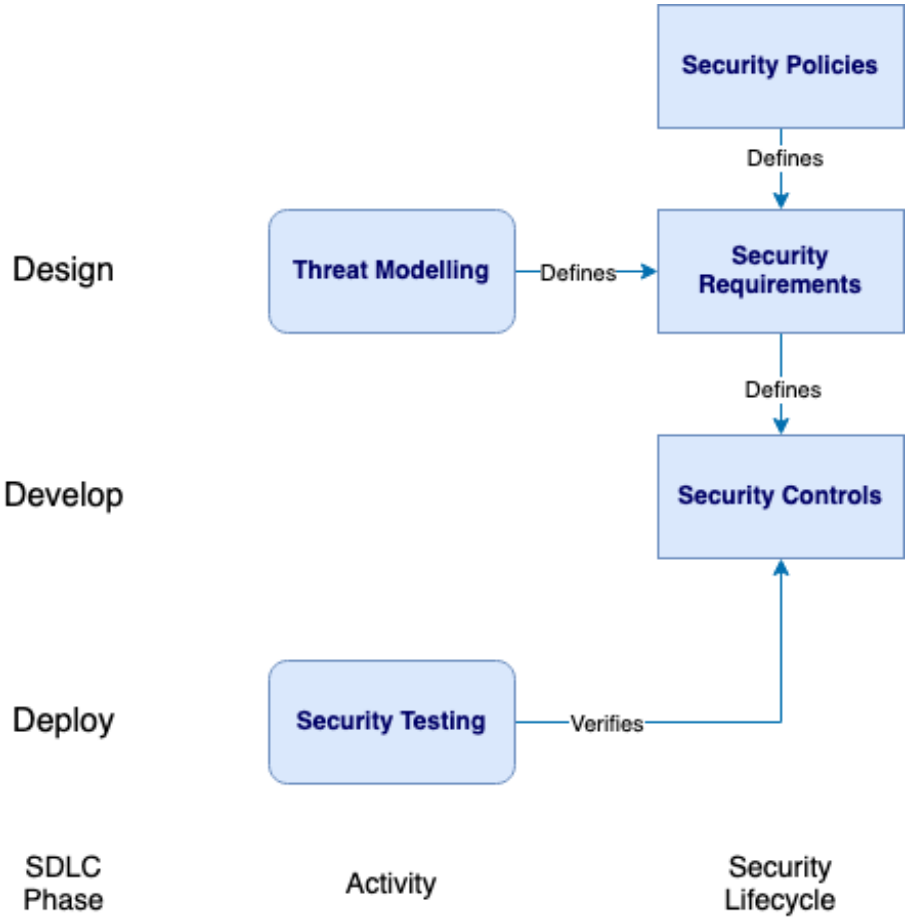

Figure 6-1: Security Requirements Flow Diagram

Why should developers perform threat modelling?

Security architects review solution designs and find security flaws. However, there is often a limited number of security architects and security professionals available to provide guidance, creating a bottleneck at the design phase. By introducing threat modelling into the development team, it can help to eliminate this bottleneck and scale out security.

When introducing the threat modelling activity, take into consideration the development teams workflow and current tools used. Consider using tools already in use rather than introducing a new tool which may take time to learn, in addition to learning threat modelling.

Threat modelling can be introduced to the development team by a security team member or

other team member knowledgeable in security, who is able to convey possible attack patterns and appropriate security controls. It can be helpful for this person to be present in the initial threat modelling activities to provide guidance, before the development team manages the activity themselves.

To introduce threat modelling to the development team, use a system that the team is familiar with.

Terminology

Terminology used as part of threat modelling and risk management:

- **Vulnerability**: a weakness in the software or missing security control
- **Threat agent**: an attacker who exploits a vulnerability
- **Risk severity**: the risk posed to the organisation by a particular vulnerability. Risk severity is calculated from the Impact and Likelihood values.
- **Impact**: the business impact resulting from a threat
- **Likelihood**: the likelihood of the threat occurring
- **Security Controls**: reduce Likelihood or Impact of a threat by addressing the associated vulnerability

For example, the threat of a Cross-Site Scripting attack (in which the *threat agent* causes unauthorised javascript to run in the victim's web browser) attempts to exploit a *vulnerability* in the lack of input validation and use of escaping functions in a web application. An appropriate *security control* is to use input validation and escaping libraries, and a Web Application Firewall (WAF).

A simplified step-by-step guide of the threat modelling process

1. Model the system

Draw a data flow diagram of the system. There are particular tools available that can be used in threat modelling to create a data flow diagram and indicate the threats, such as OWASP Threat Dragon. Also consider making use of any existing tools the development team is already familiar with, rather than introducing a new tool, which may take time to learn and adopt.

Example: threat model of a payment gateway A data flow diagram created in OWASP Threat Dragon showing the flow from a web browser using a merchant system to make a payment. The merchant system accesses a payment gateway, which uses a payment processing system, storing transaction details in a database.

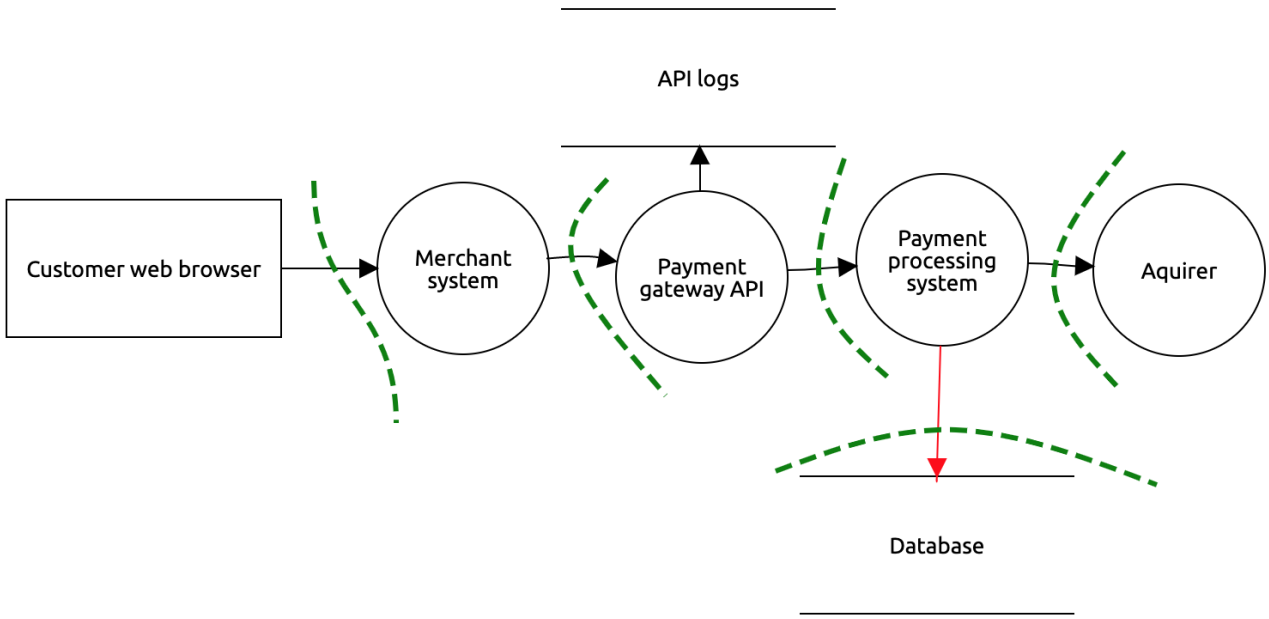

Figure 6-2: Example Threat Model

Threat model data flow diagrams use a rectangle to represent Actors. A circle is used to represent Processes, such as a web application or API. And a rectangle without vertical edges is used to represent data stores, such as a database or configuration files. Arrows are used to indicate the information flows between the Actors, Processes and data stores. Trust boundaries, dotted lines, are placed on the data flow diagram on information flows between systems to indicate where data changes its level of trust. Finally, identify any sensitive data that exists, either in information flows or data stores. It may help to consult the organisation's data classification policy when identifying the sensitive data.

2. Identify threat agents/vulnerabilities

- Define attack entry points (trust boundaries)
- Look for system vulnerabilities and threats for identified attack entry points (threats can be external threat agents or inside threat agents).
- To help identify possible threats and vulnerabilities, the STRIDE methodology can be used. If a development team is not yet familiar with the STRIDE methodology, it may be easiest to instead use a list of threats that the team is familiar with. A custom or standard checklist can be used focusing on either:
  - Vulnerabilities, such as OWASP Top Ten
  - Security requirements, such as OWASP Application Security Verification Standard (ASVS)
  - Attacks, such as OWASP Attacks list

Example: threat model of a payment gateway

- Define attack entry point: communication to the database
- Define vulnerability: sensitive data exposure (OWASP Top Ten A3:2017)
- Associated security requirement: encrypted communications required (OWASP ASVS v4.0.2-9.2.2)

3. Determine risk ratings

- Define impact and likelihood of each threat. This can be either a quantitative measure, such as OWASP Risk Rating Methodology, or qualitative using for example low; medium; high ratings.
- Determine risk rating for each threat (calculated from impact and likelihood)
- Rank threats by severity rating

Example: threat model of a payment gateway

- Risk rating for missing encrypted communication to the database is determined to be Low

4. Determine mitigations

- Determine risk treatment strategy for each risk: reduce, transfer, avoid, accept
- Agree on risk mitigation with risk owner and stakeholders
- Select appropriate controls according to strategy. Consider using the OWASP Top Ten Proactive Controls
- Write security tests to test the controls are in place
- Reevaluate risk rating after controls applied
- Periodically retest risk

Example: threat model of a payment gateway

- Risk mitigation selected is to encrypt the communication to the database (OWASP Proactive Control C3)

For further guidance on threat modelling consult the following resources:

- OWASP Threat Modeling community page
- OWASP Threat Modeling Cheat Sheet

Gamification

Use gamification as a way to introduce the threat modelling activity to the development team. Gamification can ensure all members of a team are involved and given the opportunity to provide input. A security team member can initially run and record the results of the gamification activity, before handing over to the development team.

The OWASP Cornucopia card game is designed to help developers think about possible threats in a solution design, and derive a set of security requirements to build against. Team members are each dealt cards which describe particular threats. They then take turns trying to make a case for their particular threat posing a risk to the solution design, scoring points if they are able to do so.

OWASP Cornucopia uses threats grouped into areas that are particularly relevant to software developers, such as authentication; authorisation; data validation. The threats are derived from OWASP Application Security Verification Standard (ASVS) and OWASP Web Security Testing Guide. Using OWASP Cornucopia can be useful when it is desired to have security requirements aligned with these standards.

Security testing

Use security tests to verify that the required security controls are in place, as defined in the security requirements. This chapter will discuss the selection of security tools; adding security tests into the development pipeline; the types of testing and tools that can be used; vulnerability management; and the use of penetration testing. For a detailed guide on how to conduct security testing refer to the OWASP Web Security Testing Guide.

Using security tools

Care should be taken when selecting and embedding security tools. Security tools should be easy for a developer to use, ideally embedded within a developer native tool, rather than be operated solely by an application security engineer. The security team should tune code scanner tools to ensure a minimum of false positives are reported such that the tool provides value, and does not waste the developers' time. Too many false positives could lead to developers ignoring true positives.

Security tools should give a good indication of what the found issues are and how to fix them. Security tool findings can be provided directly to the developers, to allow for rapid feedback on any security issues. In some cases it will be necessary first for the security team to provide an analysis of the findings.

Security tests can be configured to fail a code build if the tests do not pass. For development on a new codebase it can be useful to add in security tests from the start, such that developers are used to these checks.

Adding security tests into the development pipeline

Security tests have traditionally been conducted after code has been deployed. However, security tests can also be added in earlier phases of the SDLC, such as develop and commit[6]. Security tests can run as the code is written, with feedback delivered directly in the IDE (Integrated Development Environment). Security tests can happen at commit time, checking for known insecure patterns in the source code before being added to the code repository or merged to the main branch, this is known as Static Application Security Testing (SAST). Security tests are run at build time, such as checking for any vulnerabilities in libraries, known as Software Composition Analysis (SCA); or vulnerabilties in container images. Security tests run at deploy time, which allows automated testing on a running application, known as Dynamic Application Security Testing (DAST).

[6] Scott Gerlach, Developer's struggle with security, OWASP 20th Anniversary. 2021.

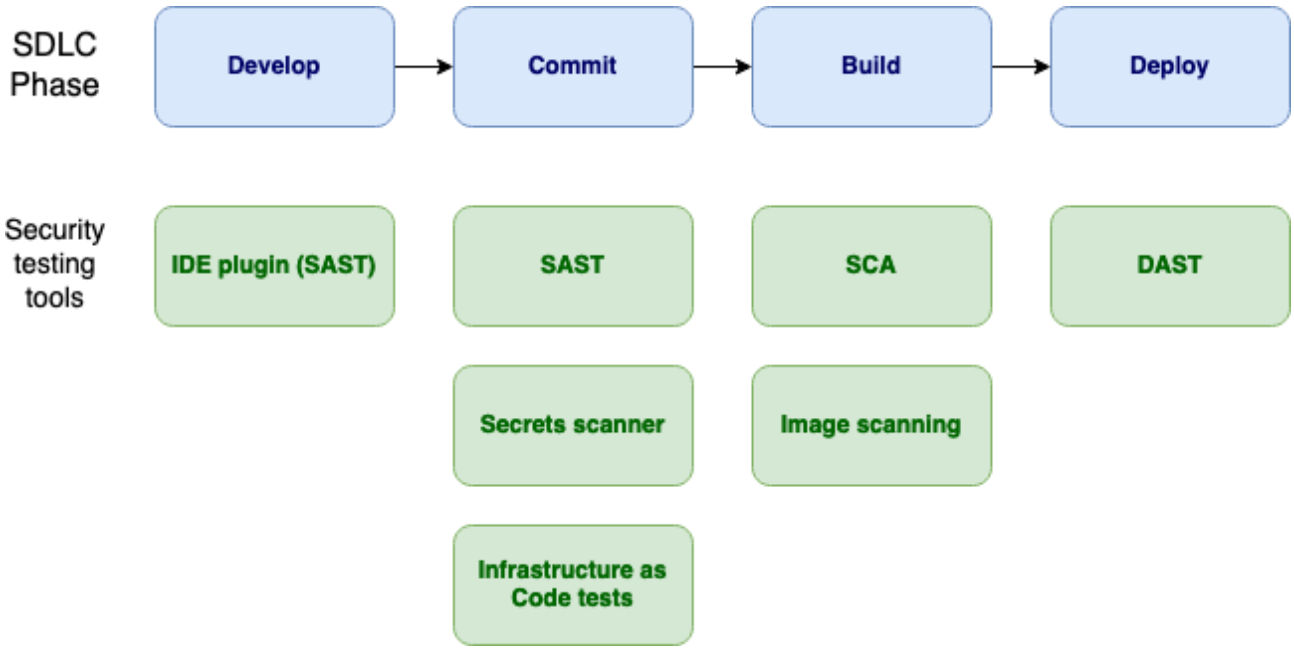

Figure 7-1: Security Testing Tools by SDLC Phase Diagram

Types of security tests

This section will provide a list of the OWASP projects that can be used in the different SDLC phases. See also Free for Open Source Application Security Tools

Checks at coding time

- IDE plugin: part of a Static Application Security Testing (SAST) tool that highlights secure coding recommendations in real time within the developer's Integrated Development Environment as they write code

Tests at commit time

- Static Application Security Testing (SAST): provides feedback upon commit of source code. For more details see OWASP Source Code Analysis Tools
- Secrets scanner: check for sensitive data such as passwords and api keys that may appear in committed source code or configuration files
- Infrastructure as Code (IaC) analysis: infrastructure resources can be defined and created from static files, providing the opportunity to run security checks when the files are committed before the infrastructure changes are made.

Tests at build time

- Software Composition Analysis (SCA): Check for vulnerabilities in third party libraries used by the application. For more details see OWASP Component Analysis
  - OWASP Dependency track
  - OWASP Dependency check
- Image scanning: Container images can be scanned for vulnerabilities before deployment

Tests at deploy time

- Dynamic Application Security Testing (DAST): tests performed on the running application. Tests can be conducted in a non-production environment before moving to production. For more details see OWASP Vulnerability Scanning Tools
  - OWASP ZAP

Penetration testing

A penetration tester plays the role of the attacker to find and exploit vulnerabilities. This helps provide a more accurate risk rating than vulnerability scans alone. Although penetration testing occurs at the end of the SDLC, the results of the penetration test can provide feedback for tests in the earlier phases. Such as additional rules for SAST and DAST scanners, and to use SCA to confirm vulnerabilities found by the penetration test[7].

A penetration test report should clearly detail found vulnerabilities, and how to fix them. It is also helpful to show how the vulnerability was exploited. This helps a developer test that their fix has worked.

A security team needs to help the development team interpret the penetration test report and provide guidance. An application security engineer may first check the report to remove any false positives before assigning developers to address the found vulnerabilities.

Bug bounty programs

Another approach to discover exploitable vulnerabilities is to implement a bug bounty program. An organisation with a bug bounty program invites bug hunters or security researchers to provide vulnerability details in return for a bounty reward, possibly monetary, for the finding.

It is important to have a well defined program detailing the type of findings that will be accepted and the systems in scope for testing.

An example bug bounty program can be found at the OWASP Bug Bounty page

For more details see the OWASP Vulnerability Disclosure Cheat Sheet

Vulnerability management and Application Security Posture Management (ASPM)

As vulnerabilities are identified from security testing tools they need to be recorded and managed. As mentioned in the threat modelling section, vulnerabilities should be defined with an Impact and Likelihood risk rating. Risk ratings use a quantitative rating such as the OWASP Risk Rating Methodology, or qualitative using for example low; medium; high. When vulnerabilities are assigned a risk rating, this allows their remediation to be prioritised accordingly. An organisation may implement a required timeframe that vulnerabilities of a particular risk rating are to be remediated.

Extending on from Vulnerability management, the goal of Application Security Posture Management (ASPM) is to prioritise the remediation of vulnerabilities that will have a business

[7] Daniel Krasnokucki, Feedback loop in DevSecOps - mature security process and dev cooperation, OWASP 20th Anniversary. 2021.

impact. Rather than trying to fix all vulnerabilities, there should be a focus on remediating the vulnerabilities with the greatest risk to the organisation. Factors that can determine a higher risk rating include whether a vulnerability is exploitable, publicly accessible, or if the vulnerability exists on a critical asset.

An Application Security Posture Management tool aggregates vulnerabilities from various security testing tools. This can be useful to help build security culture as developers and security engineers will have a single location to access all vulnerabilities, or all findings could be exported to a tool that developers are already using. An Application Security Posture Management tool makes it easier to assign particular vulnerabilities to relevant teams in an automated way, for example based on code repository.

- OWASP Defect dojo

Metrics

Use metrics to measure the progress of the security uplift program. According to OWASP SAMM metrics "evaluate the effectiveness and efficiency of the application security program"[8]. Metrics help to improve the security maturity by measuring the progress on the defined maturity goals, and allow for adjustments to the security program to ensure the maturity goals are met.

Metrics are used to tell a story to justify an action, such as a security budget, and argue for change[9]. Metrics are a useful tool to present to management to highlight where there may be gaps in the organisation's security posture that require additional resources to address, or to show that resources spent on security are having the desired effect of reducing risk to the organisation.

To start, examine business processes to see where they can be quantified and measured. Metrics are measured over time, such as quarterly, to show a trend. It is important to know what the desired direction for the metric to move is, up or down, and what activity may have caused significant movement.[10]

Take note of which Software Development Lifecycle (SDLC) phase has identified the most security issues. It is more expensive to fix security issues later in the development cycle.[11] Aim to identify issues sooner in the SDLC in accordance with maturity goals.

Example metrics

Design - Threat Modelling

- number of threat models or threat modelling activities conducted
- number of findings from threat models

Develop - Code Reviews

- number of code reviews conducted
- number of findings from code reviews

Deploy - Security testing

- number of findings from penetration testing (internal vs vendor)
- number of findings from vulnerability scanning
- number of findings remediated
- time taken to remediate a vulnerability, does it meet an SLA (An organisation may require vulnerabilities to be remediated in a certain time frame in accordance with the vulnerabilities risk rating)

Security team collaboration

- number of secure by default modules created
- number of secure architecture designs created and provided to developers

[8] OWASP. *SAMM.* *https://owaspsamm.org/model/governance/strategy-and-metrics/stream-b/*
[9] Marcus Ranum, AppSec California. 2016. *https://www.youtube.com/watch?v=yW7kSVwucSk*
[10] Ibid.
[11] OWASP. *Application Security Guide for CISOs.* *https://owasp.org/www-pdf-archive/Owasp-ciso-guide.pdf*

Security champions

- number of security champions onboarded

Training activities

- number of training activities introduced to developers
- number of developers onboarded to training activities
- number of developers engaged in particular training activities

Example graph using metrics

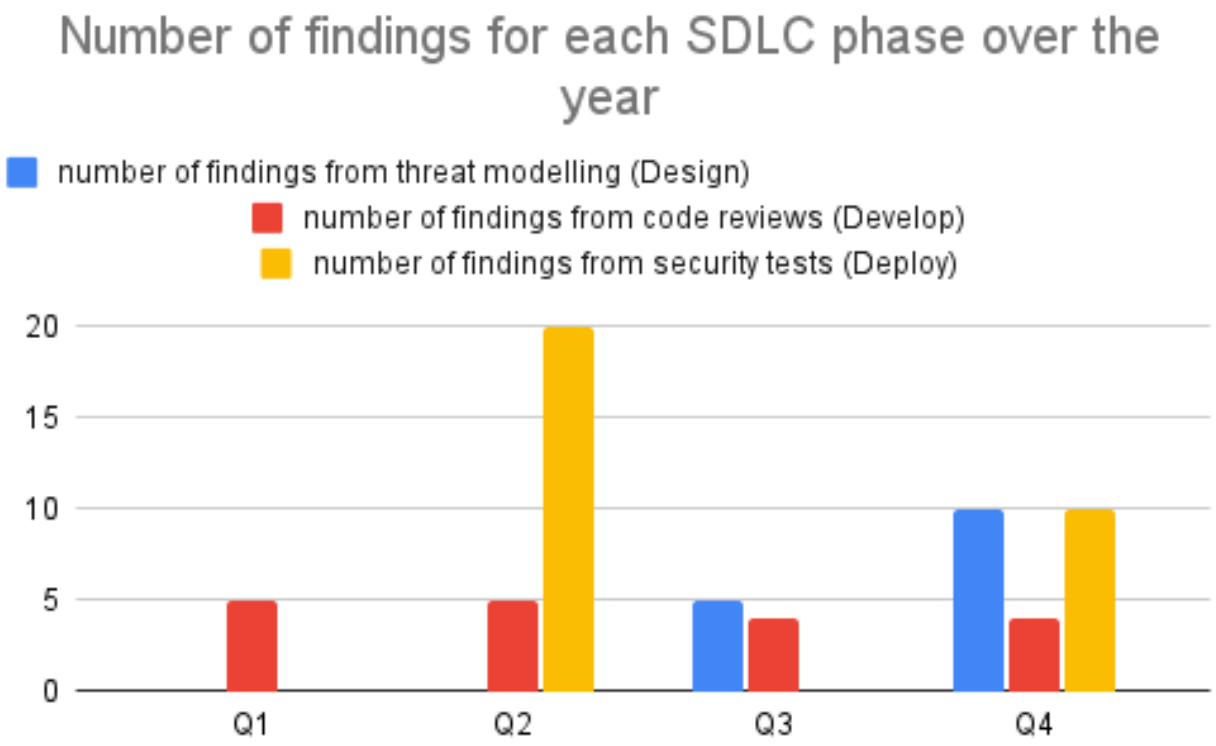

Figure 8-1: Example Metrics Graph

The metric for the number of findings from threat modelling in the Design phase increases in Q3 upon the threat modelling activity implementation for the organisation. This metric increase is expected and desired. The metric further increases in Q4 as the threat modelling activity is scaled out by Security Champions, an expected and desired increase.

The metric for the number of findings from security tests increases in Q2 as a result of a penetration test. This metric increase is expected. Upon a further penetration test in Q4, the metric has decreased from it's earlier record. This metric change is desired, as it represents a cost saving. The cost of addressing security findings increases at later stages of the development lifecycle.

Appendix: OWASP projects

A list of OWASP projects related to the chapters in this guide. Also refer to the Application Security Wayfinder, part of the OWASP Integration Standards project.

Area	OWASP Projects
Define maturity goal	OWASP Software Assurance Maturity Model (SAMM)
Security Champions	OWASP Security Champions Playbook
	OWASP Security Champions Guidebook
Activities	OWASP Juice Shop
	OWASP Secure Coding Dojo
Threat modelling	OWASP Threat Dragon
	OWASP Cornucopia
Security testing	OWASP Web Security Testing Guide
	OWASP Dependency Track
Metrics	OWASP CISO Guide