Security Development Lifecycle

Hausarbeit

vorgelegt von
Nathalie Costas Coronado, Stavros Giannis, Melike Demgül Öztürk, Hasan Hüseyin Sayan, Lenny Scheiding

Prof. Dr. Marcus Niemietz
Cyber Security Management
Clavis - Institut für Informationssicherheit

Entstanden an der

Hochschule Niederrhein
University of Applied Sciences
Wirtschaftswissenschaften
Faculty of Business Administration and Economics

Inhaltsverzeichnis

Abkürzungsverzeichnis

AA Architecture Assessment

API application programming interface

BSI Bundesamt für Sicherheit in der Informationstechnik

DM Defect Management

EM Environment Management

FSR Final Security Review

IM Incident Management

OM Operational Management

OWASP Open Web Application Security Projekt

RT Requirements-driven Testing

SAMM Software Assurance Maturity Model

SB Secure Build

SDL Security Development Lifecycle

SRA Security Risk Assessments

ST Security Testing

Zusammenfassung

Im Rahmen dieser Hausarbeit wird der Security Development Lifecycle, welcher im Jahr 2004 bei Microsoft integriert wurde und das OWASP-SAMM Model, welches im Jahr 2009 aktualisiert wurde, genauer erklärt. Der Anstieg vieler komplexer Systeme und die Vielzahl der immer steigenden Cyberangriffe und Schwachstellen zeichnen den Bedarf an den Hilfsmodellen wie der OWASP-SAMM und dem SDL Model.Daher werden diese und ähnliche Modelle fortführend aktualisiert und den aktuellen Bedingungen angepasst. Die Aufgabe der Modelle ist es, für einen Standard an Sicherheitsmaßnahmen zu sorgen, die bei der Entwicklung von Software integriert wurde. Beide Modelle haben das gleiche Ziel, aber benutzen im Kern verschiedene Praktiken, die sich in einigen Bereichen unterscheiden oder ähneln. Das Ziel der Hausarbeit ist es, diese Praktiken aufzuzählen und im Einzelnen zu erläutern. Dafür wird zunächst der SDL von Microsoft behandelt. Nach einer Einführung werden die Phasen im Detail erläutert. Daraufhin werden weitere Anforderungen beschrieben, sowie die Erhaltung von Anwendungssicherheit im Kontext des SDL. Im zweiten Teil der Arbeit wird das Modell SAMM v2 der Organisation OWASP präsentiert. Hier werden ebenfalls die Phasen verdeutlicht und erklärt. Durch die Erarbeitung beider Modelle wird es möglich, einen umfassenden Vergleich ziehen zu können und ein Verständnis für beide Modelle zu entwickeln. Abschließend wird ein allgemeiner Ausblick der sicheren Softwareentwicklung anhand beider Modelle gegeben und ein Fazit zu diesen gezogen.

1 Einführung

Jede Person will einen geschützten Aufenthalt im Internet haben. Um dies zu gewährleisten, gibt es heutzutage mehrere verschiedene Antiviren-Programme, die von der benutzten Software bereitgestellt oder empfohlen werden, um die Person vor Eindringlingen durch diverse Internetseiten oder auch E-Mails zu schützen. Damit eine dritte Partei keinen Zugriff auf jegliche personenbezogenen Informationen erhält und der Nutzer immer geschützt bleibt, muss die benutzte Software auch dementsprechende Sicherheit bieten. Damit eine sichere Software kein Zufall ist, muss diese auch mit der Intention codiert werden.

Die zu entstehende Software muss also durchstrukturiert und durchgeplant werden. Dies gelingt durch zwei verschiedene Methoden. Diese wären der Security Development Lifecycle und das SAMM-Modell.

Bei dem SAMM-Modell handelt es sich um einen Fragenkatalog, der die Rahmen der Sicherheit einer Software befragen soll. Durch dieses Modell können Unternehmen ihre Software kontrollieren und dadurch auch direkt verbessern. Hiermit kann man die Software in drei Reifegrade einteilen, wobei der dritte Reifegrad die höchste Sicherheit und kein Grad die niedrigste Sicherheit zeigt. Diese Sicherheitsgrade können durch die fünf Kernbereiche Governance, Design, Implementierung, Verification und Operations erreicht werden[1].

Eine weitere Methode ist der SDL. Diese Methode ist ein Prozess, den Microsoft im Jahr 2002 zum Leben gerufen hat, um die Sicherheit ihrer Software zu gewährleisten, da Windows in den Vorjahren häufig von Wurmattacken befallen wurde. Dies lag daran, dass fast jeder Haushalt den Kontakt zum Internet erhielt und PCs mehr verbreitet wurden. Dadurch gab es einen raschen Anstieg an bösartiger Software. Um verschiedene Schadsoftware zu entgehen, wurde das SDL entwickelt.[2]

Die erste Software, die nach dem SDL Prozess fertiggestellt und benutzt wurde, war Windows Vista. Windows Vista hat direkt Erfolge gezeigt, in dem es 53 Schwachstellen weniger als sein Vorgänger Windows XP aufwies. Es waren nur noch 66 Schwachstellen statt 119.

Nach diesem positiven Ergebnis des „Security Development Lifecycles“ wurde das SDL zu einem integrierten Bestandteil des Softwareentwicklungsprozesses. Heute

[1] vgl. Kerbl [2020]
[2] vgl. Eilers [2018]

über ein Jahrzehnt später ist das SDL nicht mehr wegzudenken, da es in jedem Entwicklungsprozess Bestandteil geworden ist.

2 Security Development Lifycycle

Der Security Development Lifecycle wurde im Jahr 2004 bei Microsoft integriert und spielt seit daher eine große Rolle bei der Entwicklung von neuer Software. Es ist ein Prozess, der universell auf jede Firma und Einrichtung anwendbar ist, welche sich mit der Entwicklung von Software beschäftigt. Der Zyklus bietet Richtlinien zur Erhaltung eines Minimalstandards, um Schwachstellen, in der Sicherheit der Software, vorzubeugen.
In den einzelnen Phasen des SDL werden praktische Verfahren angewendet, um den Datenschutz und die Sicherheit an jedem Zeitpunkt in der Entwicklung zu gewährleisten. Die drei Hauptkonzepte, die dabei von Microsoft ausgearbeitet wurden, sind die Schulung, kontinuierliche Prozessverbesserung und Verantwortlichkeit. Schulung stellt den Prozess dar, sich auf neue Technologien und die damit einhergehenden Gefahren vorzubereiten und zu informieren und dementsprechend in den SDL mit einzubauen. Dadurch wird der SDL innerhalb eines Unternehmens nicht statisch zu einem statischen Prozess, sondern steht selbst ständig im Wandel.
Der SDL wird bei Microsoft selbst bis heute in einhundert verschiedenen Ländern eingesetzt und wurde seit Entwicklung der Kernkonzepte auf jedes Microsoft Produkt angewandt. Ebenso soll dieser Prozess transparent und für jeden zugänglich sein. Aus diesem Grund hat Microsoft alle Praktiken und Informationen zum SDL zur freien Verfügung gestellt.

2.1 Anwendbarkeit des SDL

Damit der SDL von Microsoft ausgeführt werden kann, müssen in der Organisation, in der der Zyklus angewendet wird, klare Erwartungen an ein Projekt und dessen Typen festgelegt werden. Erst dann kann der SDL diese mit entsprechenden Kontrollen überprüfen. Laut Microsoft sind folgende Merkmale definiert, wenn eine Anwendung dem SDL unterworfen werden sollte.

- Bereich der Anwendung z. B. Geschäfts- oder Unternehmensumgebung
- Wenn durch die Anwendung vertrauliche oder personenbezogene Daten verarbeitet werden
- Bei Kommunikation mit anderen Netzwerken und dem Internet

Durch die ständige Weiterentwicklung von Computertechnologien ist es von Vorteil für das Projekt, wenn diese weiteren Kontrollen oder Merkmale auferlegt werden, die nicht vom SDL beschrieben werden.

Der SDL basiert auf vier Best Practices. Diese sind:

- Secure by Design
- Secure by Default
- Secure by Deployment
- Communications

Secure by Default steht dafür, dass jede Software eine Standardeinstellung für die Sicherheit hat, da jede Software Lücken aufweisen wird, die man jedoch durch die Standardisierung der Software so schwer wie möglich zugänglich machen will.
Secure by Deployment ist hierbei die fertige Software, die optimal installiert und eingerichtet werden kann.
Communications steht für den schnellen Austausch zu den Benutzern. Sollte eine Schwachstelle gefunden werden, können die Benutzer so schnell wie möglich darüber informiert und mit Patches versorgt werden.

Aus diesen vier Best Practices entstehen nun fünf verschiedene Phasen mit 16 Sicherheitsmaßnahmen. Diese fünf Phasen sind Anforderungen vor dem SDL, Anforderungen, Entwurf, Implementierung, Überprüfung und Veröffentlichung[1].

[1] vgl. Eilers [2018]

2.2 Phasen des SDL

2.2.1 Obligatorische Sicherheitsmaßnahmen

Sobald ein Softwareentwicklungsprojekt als SDL-Thema identifiziert wurde, muss das Entwicklungsteam die 16 erforderlichen Sicherheitsmaßnahmen erfolgreich abschließen, um dem SDL-Prozess von Microsoft zu entsprechen. Überdies wird es im Rahmen eines strengen jährlichen Überprüfungsprozesses kontinuierlich getestet. Die Liste der folgenden Methoden sollte jedoch immer die 16 hier beschriebenen Methoden enthalten und sind jeweils in verschiedenen Phasen unterteilt[2].

2.2.2 Phase 0: Anforderungen vor dem SDL

2.2.2.1 Sicherheitsmaßnahme 1: Schulungsforderungen

Alle Mitglieder des Softwareentwicklungsteams sollten angemessene Schulungen erhalten, um in Bezug auf Sicherheitsgrundlagen und aktuelle Trends in den Bereichen Sicherheit und Datenschutz auf dem Laufenden zu bleiben. Die Vorschulung vermittelt dem Fachpersonal das notwendige Hintergrundwissen[3].

2.2.3 Phase 1: Anforderungen

2.2.3.1 Sicherheitsmaßnahme 2: Sicherheitsanforderungen

Die Notwendigkeit, Sicherheit und Datenschutz zu antizipieren, stellt einen wichtigen Aspekt der sicheren Softwareentwicklung dar. Dazu gehören die Festlegung der Mindestsicherheitsanforderungen für die Anwendung, die beim Betrieb in der vorgesehenen Betriebsumgebung erfüllt sein müssen, sowie deren Spezifikation und Verteilung. Ein Überwachungssystem für Schwachstellen und Geschäftsaufgaben[4].

2.2.3.2 Sicherheitsmaßnahme 3: Quality Gates und Bug Bars

Die Qualitäts-Gateways und Bug Bars definieren die niedrigsten akzeptablen Qualitätsstufen für Datensicherheit und -schutz. Das Projektteam muss für jede Entwicklungsphase Qualitäts-Gateways aushandeln und diese dann von einem Sicherheitsberater genehmigen lassen. Weiterhin muss das Projektteam die Einhaltung der

[2] vgl. Microsoft [2010], S. 8
[3] vgl. Microsoft [2010], S. 8
[4] vgl. Microsoft [2010], S. 9

ausgehandelten Quality Gates nachweisen, um die abschließende Sicherheitsüberprüfung abzuschließen. Der Fehlerbalken ist ein Tor zur Qualität, das für das gesamte Softwareentwicklungsprojekt gilt[5].

2.2.3.3 Sicherheitsmaßnahme 4: Risikobewertung für Sicherheit und Datenschutz

Die Sicherheitsrisikobewertung (SRA) und die Datenschutzrisikobewertung sind obligatorische Prozesse, die funktionale Aspekte von Software identifizieren, die genau überwacht werden müssen[6].

2.2.4 Phase 2: Entwurf

2.2.4.1 Sicherheitsmaßnahme 5: Entwurfsanforderungen

Der einfachste Weg, die Zuverlässigkeit eines Projektdesigns zu beeinflussen, ist in den frühen Phasen seines Lebenszyklus. Es ist essenziell, Sicherheits- und Datenschutzaspekte während der Designphase zu berücksichtigen. Die Lösung von Sicherheits- und Datenschutzproblemen ist in den frühen Phasen eines Projektlebenszyklus weniger schwierig. Projektteams sollten die üblichen Praktiken vermeiden, Sicherheits- und Datenschutzfunktionen zu verschieben und relevante Probleme bis zum Ende der Projektentwicklung anzugehen[7]. Es ist möglich, unsichere Sicherheitsfunktionen zu implementieren. Sichere Funktionen sind definiert als Funktionen, deren Funktionalität zuverlässig auf Sicherheit ausgelegt ist, einschließlich einer strengen Validierung aller Daten vor der Verarbeitung oder einer kryptografisch robusten Implementierung von Bibliotheken für kryptografische Dienste. Der Begriff Sicherheitsfunktionen beschreibt eine Programmfunktionalität, die sich auf die Sicherheit auswirkt. Beispiele hierfür sind die Erstellung von Designspezifikationen für Sicherheit und Datenschutz, die Überprüfung der Spezifikation und die Bestimmung von minimalen Designanforderungen. Designspezifikationen sollten Sicherheits- und Datenschutzfunktionen definieren, die den Benutzern direkt zur Verfügung stehen[8].

2.2.4.2 Sicherheitsmaßnahme 6: Verringerung der Angriffsfläche

Die Verringerung der Angriffsfläche ist eng mit der Bedrohungsmodellierung verknüpft, auch wenn sie Sicherheitsprobleme von einer leicht anderen Perspektive betrachtet. Zur Verringerung der Angriffsfläche gehören das Abschalten oder Einschränken des

[5] vgl. Microsoft [2010], S. 9
[6] vgl. Microsoft [2010], S. 10
[7] vgl. Microsoft [2010], S. 10
[8] vgl. Microsoft [2010], S. 11

Zugriffs auf Systemdienste, die Anwendung des Prinzips der geringsten Rechte sowie ggf. der Einsatz von Verteidigungsschichten[9].

2.2.4.3 Sicherheitsmaßnahme 7: Bedrohungsmodellierung

Bedrohungsmodellierung wird in Umgebungen verwendet, in denen ein erhebliches Sicherheitsrisiko besteht. Es ermöglicht Entwicklungsteams, die Sicherheitsauswirkungen eines Designs im Kontext der beabsichtigten Betriebsumgebung und auf strukturierte Weise zu untersuchen, zu dokumentieren und zu diskutieren. Die Bedrohungsmodellierung ermöglicht auch die Berücksichtigung von Sicherheitsaspekten auf Komponenten- oder Anwendungsebene[10].

2.2.5 Phase 3: Implementierung

2.2.5.1 Sicherheitsmaßnahme 8: Verwenden genehmigter Tools

Alle Entwicklungsteams sollten eine Liste genehmigter Tools und der zugehörigen Sicherheitskontrollen, wie Compiler-/Linker-Optionen und Warnungen, definieren und veröffentlichen. Diese Liste sollte vom Sicherheitsberater des Projektteams genehmigt werden. Als allgemeine Regel sollten Entwicklerteams stets bestrebt sein, die neueste Version genehmigter Tools zu verwenden, um neue Funktionen und Sicherheit bei der Sicherheitsanalyse zu nutzen[11].

2.2.5.2 Sicherheitsmaßnahme 9: Verbieten unsicherer Funktionen

Viele beliebte Funktionen und APIs sind gegen die heutigen Bedrohungen nicht mehr sicher. Projektteams sollten alle Funktionen und API's analysieren, die in einem Softwareentwicklungsprojekt verwendet werden sollen, und diejenigen verbieten, die als unsicher identifiziert wurden. Sobald die Verbotsliste festgelegt ist, sollten Projektteams den gesamten Code mithilfe von Headerdateien, neuen Compilern oder Codeanalysetools auf verbotene Funktionen untersuchen und durch sichere Alternativen ersetzen[12].

[9] vgl. Microsoft [2010], S. 11
[10] vgl. Microsoft [2010], S. 11
[11] vgl. Microsoft [2010], S. 12
[12] vgl. Microsoft [2010], S. 12

2.2.5.3 Sicherheitsmaßnahme 10: Statische Analyse

Projektteams sollten eine statische Analyse des Quellcodes durchführen. Die statische Analyse des Quellcodes bietet eine skalierbare Möglichkeit, sicheren Code zu überprüfen, und trägt dazu bei, die Einhaltung der Richtlinien für sichere Codierung sicherzustellen. Statische Codeanalyse allein reicht oft nicht aus, um die manuelle Codeüberprüfung zu ersetzen[13].

2.2.6 Phase 4: Überprüfung

2.2.6.1 Sicherheitsmaßnahme 11: Dynamische Programmanalyse

Die Laufzeitüberprüfung von Softwareprogrammen ist notwendig, um sicherzustellen, dass ein Programm die Funktionalität hat, für die es entwickelt wurde. Im SDL-Prozess werden Laufzeittools wie AppVerifier in Verbindung mit anderen Techniken wie Fuzzing-Tests verwendet, um das gewünschte Maß an Sicherheitstests zu erreichen[14].

2.2.6.2 Sicherheitsmaßnahme 12: Fuzzingtests

Fuzzingtests sind eine spezialisierte Form der dynamischen Analyse, bei der Programmfehler durch absichtliches Einbringen fehlerhafter oder zufälliger Daten in eine Anwendung verursacht werden. Die Strategie des Fuzzing-Tests leitet sich aus dem Verwendungszweck der Anwendung und den Funktions- und Designmerkmalen der Anwendung ab[15].

2.2.6.3 Sicherheitsmaßnahme 13: Überprüfung von Angriffsflächen

Durch diese Tests wird sichergestellt, dass am System vorgenommene Design- oder Implementierungsänderungen berücksichtigt und neue Angriffsvektoren, die sich aus den Änderungen ergeben, getestet und adressiert werden[16].

[13] vgl. Microsoft [2010], S. 12
[14] vgl. Microsoft [2010], S. 12
[15] vgl. Microsoft [2010], S. 12 f.
[16] vgl. Microsoft [2010], S. 13

2.2.7 Phase 5: Veröffentlichung

2.2.7.1 Sicherheitsmaßnahme 14: Vorfallreaktionsplan

Jede Softwareversion, die den SDL-Anforderungen unterliegt, muss einen Vorfallreaktionsplan enthalten. Auch Programme, die zum Zeitpunkt der Veröffentlichung keine bekannten Schwachstellen aufweisen, können im Laufe der Zeit neuen Bedrohungen ausgesetzt sein[17].

2.2.7.2 Sicherheitsmaßnahme 15: Abschließende Sicherheitsüberprüfung

Die endgültige Sicherheitsfreigabe (FSR) ist eine sorgfältige Überprüfung aller Sicherheitsmaßnahmen, die für eine Softwareanwendung getroffen wurden, bevor sie freigegeben wird. FSR ist weder ein „Penetrationstest und Patch“ noch eine Gelegenheit, bisher vernachlässigte oder vergessene Sicherheitsmaßnahmen nachzuholen[18].

2.2.7.3 Sicherheitsmaßnahme 16: Veröffentlichung und Archivierung

Die Freigabe der Software hängt vom Abschluss des SDL-Prozesses ab. Der Veröffentlichung zugeordnete Sicherheitsberater muss bescheinigen, dass das Projektteam die Sicherheitsanforderungen erfüllt. Dies umfasst alle Spezifikationen, Quellcode, Binärdateien, private Symbole, Bedrohungsmodelle, Dokumentation, Notfallpläne, Lizenz- und Wartungsbedingungen für unsere Software, Drittanbieter und alle anderen Daten, die für die Wartung nach der Veröffentlichung erforderlich sind[19].

[17] vgl. Microsoft [2010], S. 13
[18] vgl. Microsoft [2010], S. 13 f.
[19] vgl. Microsoft [2010], S. 14

3 Weitere Anforderungen des SDL

3.1 Fehlerursachenanalyse

Bei einer Fehlerursachenanalyse wird in erster Linie die Sicherheit der fertiggestellten Software geprüft. Angesichts dessen ist sie kein klassischer Teil des Entwicklungsprozesses neuer Software. Mit einer solchen Analyse sollen gefundene Schwachstellen untersucht und behoben werden, um auch zu definieren, an welcher Stelle im Entwicklungsprozess es zu einem solchen Fehler kam. Ziel bei einer Fehlerursachenanalyse ist es den Fehler, durch Menschen oder Maschine, erstens zu beheben und die Lösung in einer überarbeiteten Form in den nächsten SDL des Unternehmens mit einzubringen.

Sicherheitslücken, die seit Veröffentlichung der fertiggestellten Software auftreten werden und noch nicht öffentlich bekannt gegeben wurden, werden „Zero-Day Threats“ genannt. Das Bundesministerium für Sicherheit in der Informationstechnik (BSI) wurde am 20.4.2021 durch die IT-Sicherheitsfirma „FireEye“ informiert, dass im Programm, Namens SonicWall E-Mail Security Appliance, Sicherheitslücken gefunden wurden, die bereits im März 2021 durch Unbekannte, ausgenutzt werden. Die Angreifer konnten nach Belieben Dateien einlesen, hochladen und administrative Benutzerkonten erstellen. Das BSI hat diese Gefahr als geschäftskritisch eingestuft. [1] Die Fehlerursachenanalyse soll unterstützen, solche Schwachstellen herauszufinden und zu schließen, bevor das fertige Produkt veröffentlicht wird.

3.2 Regelmäßige Prozessupdates

Damit gewonnene Kenntnisse aus Fehlerursachenanalysen in die Software mit implementiert werden, müssen regelmäßige Updates für die Software erfolgen. Um eine sichere Software zu entwickeln, darf die Sicherung dieser nicht statisch sein. Der Prozess für regelmäßige Updates sollte somit auch in den SDL eingebunden werden. Microsoft empfiehlt ein jährliches Update außer bei aktuell entdeckten Sicherheitsschachstellen, in einem solchen Fall sollte zum schnellstmöglichen Zeitpunkt die Lücke behoben werden.
Außerdem sollte Software für den Support und Wartung eingestellt wurden, nicht mehr in kritischen Bereichen eingesetzt werden, in denen sensible Daten verarbeitet werden

[1] vgl. Microsoft [2010], S. 15

oder die zur Erhaltung des Lebensstandards beitragen. Dabei sollte, sofern möglich, entweder die benutzte Software insgesamt gewechselt werden oder auf eine neue Version mit vorhandener Wartung gewechselt werden.[2]

[2] vgl. Microsoft [2010], S. 15

4 Erhaltung der Anwendungssicherheit

In jedem Unternehmen, in dem sichere Software nach dem SDL entwickelt wird, sollte über ein Mittel verfügen, welches die Einhaltung der benutzten Prozesse überprüft. Ein solches Mittel kann etwa die Sammlung von Entwicklungs- und Testdaten sein, die positiv zur Sicherheitsüberprüfung beitragen. Dabei spricht man von einer Anwendung, die Zugriff auf alle wichtigen Prozesse und Bestandteile des benutzten SDL-Prozesses, besitzt. Diese Bestandteile können von Entwurfsmustern bis hin zu Sicherheitsanalysen, Schwachstellenprotokolle und Prozessbescheinigungen reichen[1]. Anzumerken ist dabei, dass nur bestimmte Akteure im Unternehmen Zugriff auf diese Anwendung haben sollten. Ebenso wie eine klare Trennung und Verteilung von Rollen, sodass ein Sicherheitsberater beispielsweise keine Funktionen im Code direkt ändern und hochladen kann. Ebenso sollten die Leiter für Datenschutz und Sicherheit die Verantwortung für die Kategorisierung und Einbindung in Überwachungssysteme der benutzten Daten[2].
Anhand der Daten in einer solchen Anwendung können Datenschutzberater ein Framework für die abschließende Sicherheitsüberprüfung erstellen. Eine weitere Aufgabe dieser ist es, die angebenden Daten zu überprüfen und sicherzustellen, dass alles richtig eingetragen wurde. Ebenso wie die Sicherstellung, dass alle angegebenen Maßnahmen zur Schließung von Sicherheitslücken korrekt und richtig umgesetzt wurden. Um in größeren Unternehmen die Überwachungs- und Überprüfungsprozesse erfolgreich zu erfassen, müssen folgende Dinge definiert werden:

- Sicherheits- sowie Datenschutzanforderungen der Organisation
  - Beispielsweise keine bekannten Schwachstellen bei der Veröffentlichung, sogenannte Zero Day Threats
- Technische Anforderungen der Anwendung
- Kontext in der die Anwendung eingesetzt werden soll

Auch stellt die Verteilung von Ressourcen im Anwendungskontext eine Rolle. Bei Anwendungen, die in kritischen Einrichtungen eingesetzt werden sollen, empfiehlt es sich nicht monetär du personell an Überwachungsprozessen zu sparen, da dies in Zukunft unweigerlich zu Problemen der Anwendung führt. Auch empfiehlt sich ein

[1] vgl. Microsoft [2010], S. 16
[2] vgl. Microsoft [2010], S. 16

System zu integrieren, dass in einem kritischen Zeitpunkt benutzt werden kann, um bei der Analyse unterstützend zu sein[3].

[3] vgl. Microsoft [2010], S. 16

5 Software Assurance Maturity Model

5.1 Einführung SAMM Model

Die Open Web Application Security Projekt, kurz OWASP(OWASP), ist eine Organisation, die es sich zur Aufgabe gemacht hat, die Sicherheit von Webanwendungen und Services zu analysieren und zu optimieren. Dabei gibt die Organisation Hinweise zu Sicherheitsrisiken, Schwachstellen und bietet als Gegenmaßnahme verschiedene Modelle und Projekte an, mit der die Sicherheit vieler Dienste und Anwendungen gewährleistet werden kann. Das SAMM-Model ist eines dieser Richtlinien.
Die Abkürzung SAMM steht für das Model Assurance Maturity Model(SAMM) und ist eine Anleitung zur Softwaresicherheitsoptimierung der OWASP. Dieses Model ist ein international anerkannter Standard und bildet eine Basis zur sicheren Softwareentwicklung und Verbesserung. Das SAMM-Model 2.0 besteht aus fünf Bausteinen, nämlich Governance, Design, Implementation, Verification und Operations. Diese Bausteine bestehen aus jeweils drei Unterpunkten. Das OWASP-SAMM 2.0 ist die aktuelle Version. In den folgenden Arbeitspaketen wird auf die einzelnen Funktionen im Detail eingegangen.

5.2 Phasen des SAMM

5.2.1 Phase 1: Governance

Die Governance bietet eine Richtlinie für Organisationen in Bezug auf die Verwaltung der Softwareentwicklung und wichtigen Geschäftsprozessen. Kurz gefasst geht es hier um die sichere Gestaltung der Organisationsebene. Die Governance ist unterteilt in drei Unterpunkte, die im Folgenden genauer erläutert werden[1].

5.2.1.1 Strategy and Metrics

Die Formulierung und Erstellung einer Software umfasst viele verschiedene Aktivitäten und Prozesse. Einige dieser Prozesse können jedoch unproduktiv oder unwirksam sein und somit ins Leere laufen. In diesem Unterpunkt wird auf unproduktive oder unverhältnismäßige Aktivitäten aufmerksam gemacht. Die Strategy & Metrics erzielt daher die Erstellung eines effizienten und wirksamen Planes, um eine sichere Software zu ermöglichen.
Als Grundlage dient hier ein Software-Sicherheitsprogramm, dass dabei hilft den Plan zu erstellen und weitere Aktivitäten während der Umsetzung des SAMM-Models zu erstellen. Außerdem kann dadurch der Sicherheitsstatus unter Beobachtung gehalten werden und ein Gesamtüberblick ist gegeben.
Das Ganze wird in verschiedene Reifegrade unterteilt, in denen Schritt für Schritt erklärt wird, wie der Gesamtplan effektiv erstellt werden kann. Im Detail gibt es drei Reifegrade, die in eine Phase A, Create and Promote (Erstellen und Fördern) und in eine Phase B, Measure and Improve (Messen und Verbessern) unterteilt werden. Im Allgemeinen geht es darum Ziele zu identifizieren, die Risikotoleranz zu ermitteln, einen strategischen Plan zur Verfügung zu stellen, KPI's und Ziele festzulegen und um das Wachstum des Unternehmens zu steigern[2].

5.2.1.2 Policy and Compliance

In diesem Baustein geht es darum, ein Grundverständnis für behördliche Anforderungen und rechtliche Themen zu schaffen, d. h. die Compliance im Unternehmen ist im Fokus. Wichtig dabei ist es, die unternehmensinternen Standards zu kennen und aufrechtzuerhalten. Außerdem sind die Verpflichtungen von und gegenüber Drittanbietern ein wichtiger Bestandteil und dementsprechend zu handeln.
Der richtige Umgang mit genau diesen wichtigen Verpflichtungen und Anforderungen sind in diesem Unterpunkt ebenfalls in drei Reifegrade beschrieben. Diese Reifegrade sind in der Phase A Policy & Standards (Politik und Standards), sowie Phase B Compliance Management erfasst.

[1] vgl. OWASP [2020], S. 7
[2] vgl. OWASP [2020], S. 7

Zusammengefasst geht es darum, Governance- und Compliance-Treiber zu identifizieren, die für die eigene Organisation von Bedeutung sind. Entsprechend ist es wichtig, eine Sicherheits-Baseline zu entwickeln und die Richtlinien und Standards im Unternehmen zu kennen und zu formulieren. Sicherheitsanforderungen müssen im Unternehmen veröffentlicht und für alle Mitarbeiter zugänglich gemacht werden. Doch es genügt nicht, diese Anforderungen zu veröffentlichen, es bedarf auch eine genaue Beobachtung und Dokumentation dieser. Der Status der Einhaltung eben dieser Richtlinien und Standards bestimmt letztlich den Erfolg in der Organisation[3].

5.2.1.3 Education and Guidance

In diesem Baustein ist die Schulung, die Kompetenz, das Wissen und die entsprechenden Ressourcen im Fokus. Durch Fachwissen ist es den Mitarbeitern und Projektteams möglich Sicherheitsrisiken vorzubeugen und zu erkennen. Außerdem ist es wichtig, in die eigene Unternehmenskultur zu investieren und die Zusammenarbeit der Projektteams zu fördern. Das ist primär durch Kollaborationstools und durch die Transparenz zwischen Technologien und Tools ermöglicht.
Die Durchführung der Education & Guidance ist ebenfalls durch die Einteilung in drei Reifegrade möglich, die dann jeweils in Phase A, Training and Awareness (Training und Bewusstsein) und Phase B, Organization and Culture (Organisation und Kultur). In den Phasen A und B geht es darum, den Mitarbeitern entsprechende Ressourcen zur Verfügung zu stellen und diesen den Zugriff darauf zu ermöglichen. Es ist von Bedeutung Sicherheitsbewusstseinseinschulungen anzubieten. Es kann motivierend sein, in einem Projektteam einen Sicherheitschampion zu ernennen oder aber rollenspezifische Anleitungen bereitzustellen.
Des Weiteren ist das Angebot an Schulungsprogrammen und standardisierten internen Anleitungen ebenfalls unterstützend. Gerade Mitarbeiter der Softwaresicherheitsabteilung sind für die Aufrechterhaltung der Sicherheit im Unternehmen wichtig und sollten regelmäßig geschult werden[4].

5.2.2 Phase 2: Design

In diesem Geschäftsprozess wird das Design und die korrekte Zielsetzung des Unternehmens erarbeitet.
Dieser Geschäftsprozess ist in drei Bausteine unterteilt: Threat Assessment, Security Requirements und Security Architecture[5].

[3] vgl. OWASP [2020], S. 7
[4] vgl. OWASP [2020], S. 7
[5] vgl. OWASP [2020], S. 8

5.2.2.1 Threat Assessment

Die Erkennung und das Verständnis für Risiken in einem Unternehmen, vorwiegend in Bezug auf die zu entwickelnde Software und dessen Funktionalität sind eine Aufgabe, mit der sich das Threat Assessment beschäftigt. Bedrohungsanalysen und die Wahrscheinlichkeitsmessung von möglichen Angriffen ermöglichen ein effektiveres Arbeiten. Dies wird unterstützt von Priorisierungen und Entscheidungen zur Risikoakzeptanz. In den drei Reifegrade des Threat Assesment geht es darum, Bedrohungen zu identifizieren und das Anwendungsrisiko zu bewerten. Dabei geht es darum, die Wahrscheinlichkeit eines Angriffs und dessen Auswirkungen abzuschätzen. Dies geschieht mit Diagrammen, mit Bedrohungs-Checklisten und risikobasierten Bedrohungsmodellierungen.
Das Threat Assesment ist unterteilt in die Phase A Application Risk Profile (Anwendungsrisikoprofil) und Phase B Threat Modeling (Bedrohungsmodellierung). Der bewusste Umgang mit einer sicheren Software und sicheren Systemen wird unterstützt durch Schulungen und Tools zur Bedrohungsanalyse. Es ist wichtig eine gewisse Standardisierung im Umgang mit Bedrohungen einzuführen und die Analyse von möglichen softwarebezogenen Bedrohungen zu fördern. Außerdem muss das Thema rund um das Risikoinventar begleitet werden von einer kontinuierlichen Optimierung und Automatisierung dieser[6].

5.2.2.2 Security Requirements

In diesem Abschnitt ist der Fokus auf den typischen softwarebezogenen Anforderungen. Kurz gefasst geht es darum, die Sicherheitsanforderungen in Bezug auf sichere Software und dessen Handhabung zu thematisieren und zu behandeln. Außerdem werden Themen wie das Outsourcing, Lieferantenbeziehungen und Zulieferer Bedingungen behandelt, denn auch hier ist das jeweilige Unternehmen von spezifischen Anforderungen abhängig.
In den einzelnen Reifegraden werden die Software Requirements (Softwareanforderungen) und Supplier Security (Lieferantensicherheit) näher erläutert. Hohe Anwendungssicherheitsziele werden funktionalen Anforderungen zugeordnet und Lieferanten werden in Bezug auf organisatorischen Sicherheitsanforderungen bewertet. Die einzelnen Sicherheitsanforderungen werden entsprechend ihrer Wichtigkeit und Thematik strukturiert und von Entwicklerteams genutzt. Durch Sicherheit, die aufgrund von Lieferantenvereinbarungen erreicht wird, wird die Einhaltung der organisatorischen Anforderungen gewährleistet. Die gestellten Sicherheitsanforderungen sollten für alle Softwareprojekte, für das Outsourcing und für die Lieferantenbeziehungen eingehalten werden. Ein Anforderungsframework dient dabei unterstützend, da es von Projektteams und Entwicklerteams angewendet werden kann[7].

[6] vgl. OWASP [2020], S. 8
[7] vgl. OWASP [2020], S. 8

5.2.2.3 Security Architecture

Bei der Security Architecture geht es um die Sicherheit in Verbindung mit Technologien und Komponenten, die bei der Umsetzung der Software von Bedeutung sind. Die Zusammenstellung der Sicherheitskomponenten dienen als Grundlage und das Technologiemanagement befasst sich mit der Sicherheit unterstützender Technologien. Beide werden während der Entwicklung, Bereitstellung und Inbetriebnahme der Software genutzt. Das Ganze wird von Entwicklungsstacks, Bereitstellungstools und Betriebssysteme ergänzt und umgesetzt.
Die Security Architecture ist in Phase A, Architecture Design (Architektur-Design) und Phase B, Technology Management (Technologie Management), dessen Inhalte in drei Reifegrade unterteilt und beschrieben werden. Allgemein geht es hier darum, die Sicherheitslinien in den Software-Designprozessen zu berücksichtigen und Teams so zu schulen, dass sie sich in der Anwendung grundlegender Sicherheitsprinzipien auskennen. Gemeinsame Designmuster helfen bei der Standardisierung und bieten eine schnellere Sicherheitslösung. Es ist wichtig, die formale Kontrolle des Softwaredesignprozesses und die Validierung der Nutzung sicherer Kompetenzen zu überprüfen und zu beobachten. Die eingeführte Standardtechnologie sollte bei der gesamten Softwareentwicklung beibehalten werden[8].

5.2.3 Phase 3: Implementation

Das Arbeitspaket „Implementation“ hat den Fokus auf die Prozesse und Aktivitäten im Kontext der Erstellung und Bereitstellung von Softwarekomponenten durch ein Unternehmen und den damit verbundenen Fehlern. Das gemeinsame Ziel ist es, zuverlässig funktionierende Software mit minimalen Fehlern auszuliefern[9].

5.2.3.1 Secure Build

Die Secure Build (SB)-Praxis(SB) betont die Bedeutung der standardisierten, wiederholbaren Erstellung von Software und der Verwendung sicherer Komponenten, einschließlich Softwareabhängigkeiten von Drittanbietern.
Der erste Stream konzentriert sich darauf, jegliche Subjektivität aus dem Build-Prozess zu entfernen, indem eine vollständige Automatisierung angestrebt wird.
Der zweite Stream erkennt die Prävalenz von Softwareabhängigkeiten in modernen Anwendungen an. Es zielt darauf ab, sie zu identifizieren und ihren Sicherheitsstatus zu verfolgen, um die Auswirkungen ihrer Unsicherheit auf eine ansonsten sichere Anwendung einzudämmen[10].

[8] vgl. OWASP [2020], S. 8
[9] vgl. OWASP [2020], S. 9
[10] vgl. OWASP [2020], S. 9

5.2.3.2 Secure Deployment

Einer der letzten Schritte bei der Bereitstellung sicherer Software besteht darin, sicherzustellen, dass die Sicherheit und Integrität der entwickelten Anwendungen während der Bereitstellung nicht beeinträchtigt wird. Zu diesem Zweck konzentriert sich der erste Stream der Praxis darauf, manuelle Fehler zu beseitigen, indem der Bereitstellungsprozess so weit wie möglich automatisiert und der Erfolg von den Ergebnissen integrierter Sicherheitsüberprüfungen abhängig gemacht wird.
Der zweite Stream geht über die Bereitstellungsmechanismen hinaus und konzentriert sich auf den Schutz der Privatsphäre und Integrität sensibler Daten wie Kennwörter, Token und andere Geheimnisse, die für den Betrieb von Anwendungen in Produktionsumgebungen erforderlich sind[11].

5.2.3.3 Defect Management

Die Defect Management-Praxis (DM) hat den Fokus auf das Sammeln, Aufzeichnen und Analysieren von Softwaresicherheitsdefekten und deren Anreicherung mit Informationen, um Kennzahlen basierte Entscheidungen zu treffen.
Der erste Stream der Praxis befasst sich mit dem Prozess der Handhabung und Verwaltung von Fehlern, um sicherzustellen, dass freigegebene Software ein bestimmtes Sicherheitsniveau hat. Der zweite Stream konzentriert sich auf die Anreicherung der Informationen über die Fehler und die Ableitung von Metriken, um Entscheidungen über die Sicherheit einzelner Projekte und des gesamten Sicherheitsprogramms zu leiten[12].

5.2.4 Phase 4: Verification

Das Arbeitspaket „Verification“ umfasst die Prozesse und Aktivitäten, die sich darauf beziehen, wie eine Organisation während der Softwareentwicklung erzeugte Artefakte überprüft und testet. Dies umfasst in der Regel Qualitätssicherungsarbeiten wie Tests, kann aber auch andere Überprüfungs- und Bewertungsaktivitäten umfassen[13].

5.2.4.1 Architecture Assessment

Die Architecture Assessment (AA)-Praxis(AA) stellt sicher, dass die Anwendungs- und Infrastrukturarchitektur alle relevanten Sicherheits- und Compliance-Anforderungen angemessen erfüllt und identifizierte Sicherheitsbedrohungen ausreichend mindert.

[11] vgl. OWASP [2020], S. 9
[12] vgl. OWASP [2020], S. 9
[13] vgl. OWASP [2020], S. 10

Der erste Stream konzentriert sich auf die Überprüfung, ob die Sicherheits- und Compliance-Anforderungen, die in den Praktiken Policy & Compliance und Security Requirements identifiziert wurden. Zuerst ad-hoc (aus der Situation heraus) und dann systematischer für jede Schnittstelle im System erfüllt werden. Der zweite Stream überprüft die Architektur, zunächst auf Abwehr typischer Bedrohungen, dann auf spezifische Bedrohungen, die in der Threat Assessment-Praxis identifiziert wurden. In die fortgeschritteneren Form formalisiert die AA-Praxis den Überprüfungsprozess der Sicherheitsarchitektur und bewertet kontinuierlich die Wirksamkeit der Sicherheitskontrollen der Architektur, ihre Skalierbarkeit, sowie die strategische Ausrichtung. Identifizierte Schwächen und mögliche Verbesserungen werden an die Sicherheitsarchitektur-Praxis rückgekoppelt, um Referenzarchitekturen zu verbessern[14].

5.2.4.2 Requirements-driven Testing

Das Ziel des Requirements-driven Testing (RT)-Verfahrens(RT) besteht darin, sicherzustellen, dass die implementierten Sicherheitskontrollen wie erwartet funktionieren und die angegebenen Sicherheitsanforderungen des Projekts erfüllen. Dies geschieht, indem inkrementell eine Reihe von Sicherheitstest- und Regressionsfällen erstellt und regelmäßig ausgeführt wird.
Wichtig ist hierbei, dass die Aufmerksamkeit sowohl auf positive als auch auf negative Tests gerichtet wird. Zuerst wird verifiziert, dass die Sicherheitskontrollen der Anwendung die angegebenen Sicherheitsanforderungen erfüllen, und validiert deren ordnungsgemäße Funktion. Negatives Testen befasst sich mit der Qualität der Implementierung der Sicherheitskontrollen und zielt darauf ab, unerwartete Designfehler und Implementierungsfehler durch Missbrauchs- und Missbrauchstests zu erkennen[15].

5.2.4.3 Security Testing

Die Security Testing (ST)-Praxis(ST) nutzt die Tatsache, dass automatisierte Sicherheitstests zwar schnell sind und sich gut auf zahlreiche Anwendungen skalieren lassen. Jedoch tiefgreifende Tests basierend auf guten Kenntnissen einer Anwendung und ihrer Geschäftslogik oft nur durch langsamere, manuelle Experten „Security Testing“ möglich sind. Jeder Stream hat daher einen Ansatz im Kern.
Der erste Stream konzentriert sich auf die Etablierung einer gemeinsamen Sicherheits-Baseline, um sogenannte „Low Hanging Fruits“ automatisch zu erkennen. Je mehr Fehler die automatisierten Prozesse erkennen können, desto mehr Zeit haben Experten, ihr Wissen und ihre Kreativität einzusetzen, um sich auf komplexere Angriffsvektoren

[14] vgl. OWASP [2020], S. 10
[15] vgl. OWASP [2020], S. 10

zu konzentrieren und im zweiten Stream eingehende Anwendungstests zu gewährleisten.
Das Ziel dieser Praxis besteht darin, technische und Geschäfts-logische Schwächen in der Anwendung aufzudecken und sie unabhängig von den Anforderungen für Management und Geschäftsbeteiligte sichtbar zu machen[16].

5.2.5 Phase 5: Operations

Das Arbeitspaket „Operations“ umfasst die Aktivitäten, die erforderlich sind, um Vertraulichkeit, Integrität und Verfügbarkeit während der gesamten Betriebslebensdauer einer Anwendung und der zugehörigen Daten sicherzustellen[17].

5.2.5.1 Incident Management

In diesem Modell wird einen Sicherheitsvorfall definiert, als eine Verletzung oder die Gefahr einer unmittelbar bevorstehenden Verletzung der Sicherheitsziele eines Assets. Diese können aufgrund von böswilligem oder fahrlässigem Verhalten sein. Deshalb konzentriert sich die Incident Management (IM)-Praxis(IM) auf den Umgang mit verschiedene Sicherheitsvorfälle in Ihrer Organisation.
Monate oder sogar Jahre nach der ersten Sicherheitsverletzung wurden viele Sicherheitsvorfälle entdeckt. Während der „Verweilzeit“, bevor ein Vorfall erkannt wird, können erhebliche Schäden auftreten, die die Wiederherstellung erschweren. Deshalb konzentriert sich das erste Aktivitätsstream, Incident Detection, diese Verweilzeit zu verkürzen. Sobald festgestellt wird, dass Sie von einem Sicherheitsvorfall betroffen sind, ist es wichtig, diszipliniert und gründlich zu reagieren. So kann der Schaden begrenzt und so effizient wie möglich zum normalen Betrieb zurückzukehren[18].

5.2.5.2 Environment Management

Dieses Modell beschreibt, dass die Arbeit des Unternehmens nicht an der Anwendungssicherheit endet, sobald die Anwendung betriebsbereit ist. Für die verschiedenen Elemente des von Ihnen verwendeten Technologie-Stacks werden regelmäßig neue Sicherheitsfunktionen und Patches veröffentlicht, bis sie veraltet sind oder nicht mehr unterstützt werden.
Die meisten Technologien in jedem Anwendungsstapel sind standardmäßig nicht sicher. Dies ist häufig beabsichtigt, um die Abwärtskompatibilität oder die Einrichtung zu erleichtern. Aus diesem Grund erfordert die Gewährleistung des sicheren

[16] vgl. OWASP [2020], S. 11
[17] vgl. OWASP [2020], S. 11
[18] vgl. OWASP [2020], S. 11

Betriebs des Technologie-Stacks des Unternehmens die konsistente Anwendung sicherer Basiskonfigurationen auf alle Komponenten. Die Environment Management (EM)-Praxis(EM) konzentriert sich darauf, Ihre Umgebung sauber und sicher zu halten.
Schwachstellen werden während des gesamten Lebenszyklus der Technologien, auf die Ihr Unternehmen angewiesen ist, entdeckt, und neue Versionen, die diese beheben, werden nach verschiedenen Zeitplänen veröffentlicht. Daher ist es unerlässlich, Schwachstellen-Berichte zu überwachen und geordnete, zeitnahe Patches auf allen betroffenen Systemen durchzuführen[19].

5.2.5.3 Operational Management

Die Operational Management (OM)-Praxis(OM) beschäftigt sich mit Aktivitäten, um sicherzustellen, dass die Sicherheit während der gesamten Betriebsunterstützungsfunktionen aufrechterhalten wird. Obwohl diese Funktionen nicht direkt von einer Anwendung ausgeführt werden, hängt die Gesamtsicherheit der Anwendung und ihrer Daten von ihrer ordnungsgemäßen Leistung ab. Die Bereitstellung einer Anwendung auf einem nicht unterstützten Betriebssystem mit nicht gepatchten Schwachstellen oder das Versäumnis, Sicherungskopie sicher zu speichern, kann die in diese Anwendung integrierten Schutzmaßnahmen irrelevant machen.
Die von diesem Modell abgedeckten Funktionen umfassen, sind aber nicht beschränkt auf: Systembereitstellung, Verwaltung und Außerbetriebnahme; Bereitstellung und Verwaltung von Datenbanken; und Datensicherung, Wiederherstellung und Archivierung[20].

[19] vgl. OWASP [2020], S. 12
[20] vgl. OWASP [2020], S. 12

6 Ausblick

Im Allgemeinen sollte der SDL nicht als strikte Richtlinie zur Entwicklung von sicherer Software gesehen werden, sondern vielmehr als eine Art Universalanleitung. Jedes Unternehmen, unabhängig der Größe und Ressourcen, ist in der Lage den von Microsoft beschriebenen SDL, auf einer für das Unternehmen angemessenen Ebene, anzuwenden. Es soll als Möglichkeit dienen, mit wenig umständlichen Mitteln, Sicherheitspraktiken in die Softwareentwicklung zu integrieren. Zusammen mit der Automatisierung von Prozess und Einhaltung von Richtlinien werden Schritte dargestellt, die dem SDL entsprechen.
Obwohl der SDL von Microsoft entwickelt wurde, ist zu beachten, dass dieser allgemeingültig ist. Der SDL kann also auf jedem Betriebssystem, jeder Hardware und allen weiteren Entwicklungsmethodiken angewandt werden.
Ebenso ist eine einheitliche Verbesserung des gesamten Zyklus effizienter als die Verstärkung von nur einzelnen Teilbereichen. Besonders in Zeiten, in denen staatliche Konflikte über Internet ausgetragen werden, sind besonders kritische Infrastrukturen und deren Software ein Ziel für Angreifer. Microsoft stellt selbst den SDL als freien Prozess zur Verfügung, um die Sicherheit von Software und den Datenschutz zu verbessern. Dabei wurde der SDL bei vielen Projekten bereits erfolgreich angewandt. Im Prozess der Softwareentwicklung sollen die Sicherheitsmaßnahmen maßgeblich zur Sicherheit beitragen, aber ist so flexibel, dass auch weitere Richtlinien hinzugefügt werden können und zusammen mit dem SDL arbeiten. Im Fokus stehen dabei die Punkte Schulung, Prozess und Tools, welche die Vorhersehbarkeit von Gefahren und Kompetenz gegenüber Risiken verstärken.

Richtlinien, die den SDL maßgeblich unterstützen können, ist insbesondere das OWASP SAMM Modell. Das Ziel von OWASP SAMM ist ein effizientes System, welches die Analyse und die Optimierung von sichere Entwicklungen ermöglicht. Es zielt darauf ab, die Risiken jeder Organisation oder Unternehmen zu lösen. Denn ein unvollständiger Schutz kann fatale Auswirkungen für diese Unternehmen und Organisationen haben. Deshalb ist es sehr wahrscheinlich, dass in Zukunft das OWASP SAMM Model weiterhin aktualisiert wird, denn die Vielfältigkeit der Angriffe und Schwachstellen nimmt zunehmend zu, daher ist eine kontinuierliche Aktualisierung und Weiterentwicklung der Schutzmodelle besonders wichtig.

Literaturverzeichnis

Carsten Eilers. Microsoft security development lifestyle - eine einführung, November 2018. https://www.ceilers-news.de/serendipity/1010-Microsofts-Security-Development-Lifecycle-Eine-Einfuehrung.html. Zugriff am 21. Dezember 2021.

Thomas Kerbl. Sichere Software entwickeln mit OWASP SAMM, Januar 2020. https://www.heise.de/amp/hintergrund/Sichere-Software-entwickeln-mit-OWASP-SAMM-4918292.html, Zugriff am 21. Dezember 2021.

Microsoft. Vereinfachte Implementierung des Microsoft SDL, Februar 2010. https://www.microsoft.com/en-us/download/details.aspx?id=123790, Zugriff am 21. Dezember 2021.

OWASP. Presenting OWASP SAMM, Januar 2020. https://github.com/OWASP/samm/blob/master/Supporting%20Resources/v2.0/OWASP-SAMM-v2.0.pdf, Zugriff am 21. Dezember 2021.