Disseny d'aplicacions segures

Josep Vañó Chic

PID_00208407

UOC
Universitat Oberta de Catalunya

Índex

Introducció

Per a desenvolupar aplicacions segures, s'ha de tenir en compte la seguretat en tot el procés de la creació del programari, no solament en la fase del desenvolupament en què s'escriu codi. S'ha de preveure la seguretat en totes les fases del projecte.

Dissenyar aplicacions segures implica introduir tasques, criteris, documentació, requisits, anàlisis, etc., propis de la seguretat, en el conjunt de totes les activitats que es duen a terme a l'hora de crear una aplicació.

En aquest mòdul farem una aproximació als aspectes que s'han d'implementar i de tenir en compte en el disseny d'aplicacions segures. L'enfocament el fem des de la perspectiva de la seguretat, és a dir, quins elements s'han d'incorporar en el conjunt de les activitats pròpies del desenvolupament de programari perquè sigui segur.

Per a dissenyar aplicacions segures es duen a terme una sèrie d'activitats que estan emmarcades en el que s'anomena *security development lifecycle* (SDL). S'ha de tenir en compte que l'aplicació de l'SDL, en part, es pot adaptar a les peculiaritats de cada organització, fins i tot del programari en qüestió. En aquest mòdul, l'SDL es mostra d'una manera genèrica per al desenvolupament de programari en general.

Objectius

En acabar la lectura d'aquest mòdul, heu d'haver assolit les competències següents:

1. Conèixer la importància de la incorporació de la seguretat en el cicle de vida del programari.

2. Distingir les tasques de seguretat en cada fase del cicle de vida del programari.

3. Comprendre la importància de l'anàlisi de riscos.

4. Conèixer el procés del modelatge d'amenaces.

5. Entendre les tècniques bàsiques de seguretat.

1. Cicle de vida del desenvolupament de programari segur

El primer pas d'una bona pràctica de seguretat és ser conscients que la seguretat és una part integral del procés de desenvolupament; per tant, perquè el programari compleixi els objectius de seguretat, ha d'integrar la seguretat en el cicle de vida de desenvolupament del programari.

La integració de la seguretat es pot fer mitjançant la inclusió de determinades activitats en els processos actuals d'enginyeria de programari de l'organització, sia el model en cascada, el model iteratiu i incremental o el model de desenvolupament àgil. Aquesta integració de la seguretat en el cicle de vida del programari és el que s'anomena *security development lifecycle*[1].

(1)SDL: *security development lifecycle*.

En aquest mòdul considerarem les fases d'un cicle de vida de desenvolupament genèric. Cal tenir en compte que els processos d'aquestes fases són gairebé sempre personalitzats per a adaptar-se a les necessitats del grup de desenvolupament de programari[2] i al model d'enginyeria escollit.

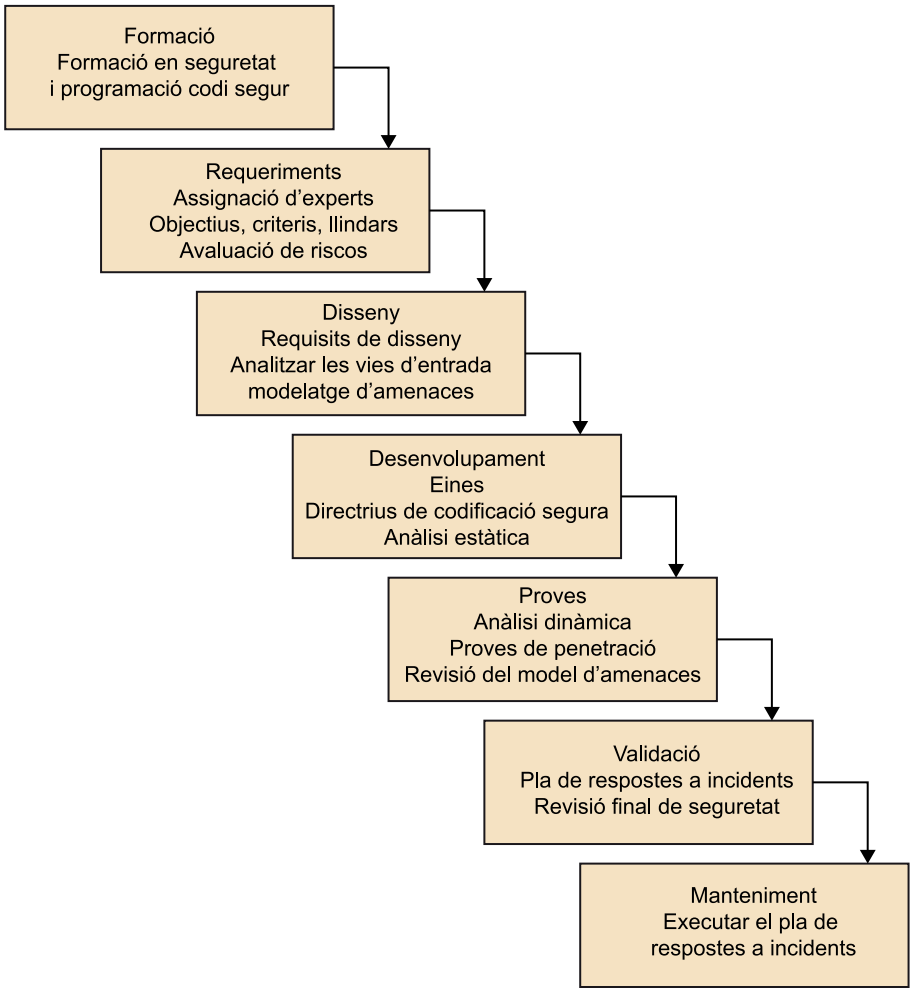
(2)SDLC: *software development lifecycle*.

Fases del cicle de vida del desenvolupament del programari:

Bibliografia

Trobareu diversa informació sobre SDL per Internet.

Podeu consultar The Open Web Application Security Project (OWASP).



1.1. Formació

En el desenvolupament d'un programari més bo, és fonamental comprendre els problemes en la seguretat. Amb la formació de l'equip de desenvolupament sobre aspectes bàsics i de les últimes tendències en seguretat i també sobre la programació de codi segur, s'aconsegueix que l'equip assoleixi un nivell més adequat de conscienciació sobre la seguretat i fa que augmenti el compromís en l'escriptura de codi segur. Així doncs, és important establir un pla de formació i també la freqüència que ha de tenir per a mantenir l'equip al dia sobre les novetats en seguretat.

Un altre aspecte que hem de tenir en compte és la incorporació de nous membres a l'equip. Durant les entrevistes amb el membres potencials de l'equip, es necessari fer preguntes sobre la seguretat per a avaluar el nivell de coneixement en aquest aspecte. Això ajuda a seleccionar els desenvolupadors que són conscients en seguretat i a avaluar la necessitat de formació addicional sobre la seguretat.

1.2. Requisits

Es considera que es necessita deu vegades més de temps, diners i esforç per a corregir una errada en la fase de desenvolupament que en la fase de disseny i deu vegades més en la fase de proves que en la fase de desenvolupament; per tant, s'ha d'incorporar el concepte de la seguretat des de les primeres fases del cicle.

Els objectius i requisits de seguretat i privacitat s'han de definir a l'inici del procés de desenvolupament de l'aplicació. Definint els requisits a l'inici, els equips de desenvolupament podran identificar les principals fites i resultats integrant la seguretat i la privacitat, de manera que facilitaran que la planificació s'alteri tan poc com sigui possible. Els objectius de la seguretat són propòsits i restriccions que afecten la confidencialitat, integritat i disponibilitat de les dades i l'aplicació. Un programari vulnerable posa en perill aquestes tres propietats que determinen que un sistema informàtic sigui fiable. Així doncs, en aquesta fase s'han de preveure les tasques següents:

- Assignar els experts en seguretat.
- Definir els objectius de seguretat.
- Establir els criteris de seguretat.
- Fixar els llindars de seguretat i els límits d'errors.
- Avaluar riscos i analitzar costos.

1) Assignar experts en seguretat

Identificar l'equip o persona que tindrà la responsabilitat de fer el seguiment i la gestió de la seguretat del producte i també de coordinar i comunicar l'estat de qualsevol problema de seguretat.

2) Definir els objectius de seguretat

Els objectius varien en funció de diversos factors, com, per exemple, el tipus d'usuari que utilitzarà l'aplicació, el tipus de dades que es tractaran o l'entorn d'execució.

Tot i que en una organització es poden establir un conjunt de categories d'objectius comuns, cal considerar que cada aplicació té objectius propis, de manera que és necessari fer un procés d'identificació dels objectius.

Per a identificar els objectius es pot començar per un document que doni resposta, per exemple, a les preguntes següents:

- Quina és l'audiència de l'aplicació?

- Quin tipus d'usuari la utilitzarà?

- Hi ha d'haver diferents nivells de requisits de seguretat en funció de l'usuari?

- És una aplicació interna de l'organització, externa o totes dues?

- On s'executarà l'aplicació? Internet? Xarxa? De quins sistemes de seguretat es disposa? Tallafoc? DMZ?

- Què es vol protegir?

- Quines implicacions hi ha si es comprometen els objectes que es volen protegir?

- Quins serveis d'infraestructura de seguretat del sistema operatiu o del sistema en general es poden aprofitar?

- Fins a quin punt ha de ser protegit l'usuari dels seus propis actes?

En el procés d'identificació, els objectius de seguretat es poden classificar segons el tipus d'objectiu amb les possibles vulnerabilitats de l'aplicació i els problemes potencials, com, per exemple:

Tipus d'objectiu	Qüestionari
Valors tangibles que s'han de protegir	• Hi ha comptes d'usuari i contrasenyes que s'han protegir? • Hi ha informació confidencial de l'usuari (per exemple, números de targeta de crèdit) que s'ha de protegir? • Hi ha propietat intel·lectual sensible que s'ha de protegir? • Es pot utilitzar aquest sistema com un conducte per a accedir a la resta del sistema de l'organització?
Valors intangibles que s'han de protegir	• Hi ha valors corporatius que es podrien veure compromesos per un atac a aquest sistema?
Requisits	• Hi ha polítiques de seguretat corporatives que s'han de complir? • Hi ha legislació sobre seguretat que s'ha de complir? • Hi ha legislació sobre privacitat que s'ha de complir? • Hi ha estàndards o normes que s'han de complir? • Hi ha restriccions o condicionants deguts a l'entorn de desenvolupament?
Requisits de la qualitat del servei	• Hi ha requisits específics de disponibilitat que s'han de complir? • Hi ha requisits específics de rendiment que s'han de complir?

3) Establir els criteris de seguretat

Fer l'anàlisi dels requisits de seguretat i privacitat, la qual ha d'incloure les especificacions de seguretat mínimes de l'aplicació en l'entorn operatiu previst, i també l'especificació i la implementació d'un sistema de seguiment dels elements de treball i de les vulnerabilitats de seguretat.

4) Fixar els llindars de seguretat i els límits d'errors

Definir els llindars de qualitat i els límits d'errors per a establir els nivells mínims acceptables de qualitat en matèria de seguretat i privacitat. Definint aquests criteris a l'inici del projecte, es comprendran més bé els riscos associats als problemes de seguretat, i els equips podran identificar i corregir els errors de seguretat durant el desenvolupament.

5) Avaluar riscos i analitzar costos

Implementar sistemes de seguretat i privacitat implica un augment del cost en el desenvolupament, però no implementar sistemes de seguretat pot comportar un cost molt més elevat, degut, per exemple, a la pèrdua d'informació confidencial, la no-disponibilitat del sistema a causa d'un atac aprofitant una vulnerabilitat o altres possibles situacions provocades servint-se d'un forat en la seguretat.

Amb l'avaluació dels riscos s'identifiquen els aspectes funcionals del programari que requereixen una revisió exhaustiva, en què s'inclou la informació següent:

- Quines parts del projecte requeriran models d'amenaça.
- Quines parts del projecte requeriran revisions de disseny de seguretat.
- Quines parts del projecte requeriran proves de penetració per part d'un grup extern a l'equip del projecte.
- Requisits d'anàlisis o de proves addicionals que es considerin necessàries per a mitigar els riscos de seguretat.

El nivell de privacitat és un aspecte que afecta la implementació de la seguretat; per tant, s'ha de mesurar el nivell d'impacte sobre la privacitat:

- **Alt risc de privacitat**. L'aplicació, la funció, el procediment o el servei emmagatzema, processa o transfereix dades d'identificació personal, canvia configuracions o associacions de tipus d'arxius o instal·la programari.

- **Risc mitjà**. Les dades tractades són anònimes.

- **Risc baix**. No hi ha res que afecti la privacitat ni hi ha cap tractament de dades personals ni anònimes, no es modifica cap configuració ni s'instal·la cap programari.

1.2.1. Metodologies i estàndards

La complexitat de dur a terme un pla de seguretat, i en conseqüència tenir-la en compte des de l'inici del cicle de vida del programari, fa que sigui necessari l'ús d'una metodologia.

Hi ha diversos sistemes de gestió de seguretat de la informació (SGSI), com, per exemple, Magerit i l'estàndard ISO 27002[3] (abans ISO 17799).

(3)ISO 27002 forma part de la sèrie de normes ISO 27000 que proporcionen un marc de gestió de la seguretat de la informació.#

L'ISO 27002 estableix un conjunt de dominis de control que cobreixen completament la gestió de la seguretat de la informació, en què cada domini es refereix a un aspecte de la seguretat de l'organització. Concretament, en el cas del desenvolupament de programari segur, es consideren aspectes de l'apartat 12, "Adquisició, desenvolupament i manteniment dels sistemes d'informació", que es desglossa en els punts següents:

12. Adquisició, desenvolupament i manteniment dels sistemes d'informació

12.1. Requisits de seguretat dels sistemes d'informació

12.1.1. Anàlisi i especificació dels requisits de seguretat

12.2. Processament correcte de les aplicacions

12.2.1. Validació de les dades d'entrada

12.2.2. Control del processament intern

12.2.3. Integritat dels missatges

12.2.4. Validació de les dades de sortida

12.3. Controls criptogràfics

12.4. Seguretat dels arxius del sistema

12.5. Seguretat en els processos de desenvolupament i suport

12.6. Gestió de la vulnerabilitat tècnica

Magerit[4] és una metodologia pública[5] desenvolupada pel Ministeri d'Administracions Públiques que cobreix els aspectes de l'anàlisi i gestió de riscos dels sistemes d'informació i que a la vegada es basa en diverses normes, entre les quals, l'ISO 27002. Aquesta metodologia consta de quatre fases:

(4)*Magerit* és l'acrònim de *metodologia d'anàlisi i gestió de riscos dels sistemes d'informació*.

(5)La seva documentació està disponible en el portal del Ministeri d'Administracions Públiques.

- Planificar l'anàlisi i gestió de riscos.
- Analitzar riscos.
- Gestionar el risc.

- Seleccionar salvaguardes.

1.3. Disseny

En la fase de disseny s'hi inclouen les especificacions funcionals i de disseny en termes de seguretat; a més a més, es fa l'anàlisi de riscos i el modelatge d'amenaces per a identificar les amenaces i vulnerabilitats en el programari.

En les especificacions funcionals es descriuen les característiques de seguretat o funcions de privacitat que estan directament exposades als usuaris, com, per exemple, la necessitat d'autenticació de l'usuari per a accedir a dades específiques o el consentiment de l'usuari abans d'executar una funció de privacitat d'alt risc. En les especificacions de disseny es descriu com s'han d'implementar aquestes característiques i funcionalitats, i a més a més com s'ha de fer per a implementar-les de manera segura.

En aquesta fase es preveuen les tasques següents:

- Establir els requisits del disseny de la seguretat.
- Analitzar les vies d'entrada dels atacants.
- Crear el modelatge d'amenaces.

1) Establir els requisits del disseny de la seguretat

Alguns dels exemples de les activitats en aquest punt són els següents:

- Crear les especificacions de disseny en matèria de la seguretat i privacitat.
- Especificar els requisits mínims del disseny criptogràfic.
- Descriure les característiques de seguretat o privacitat que seran exposades als usuaris.
- Descriure com s'han d'implementar de manera segura les funcionalitats i les diverses característiques del programari.

2) Analitzar les vies d'entrada dels atacants

Els atacs aprofiten els punts dèbils i les vulnerabilitats; per tant, és important limitar les possibilitats d'accés als atacants, i així reduir els riscos. En aquest cas s'ha de tenir en compte, per exemple, els aspectes següents:

- Tancar o restringir l'accés als serveis del sistema.
- Aplicar el principi de privilegis mínims.

- Configurar correctament les excepcions del tallafoc.

3) Crear el modelatge d'amenaces

Aquest procediment permet als equips de desenvolupament considerar, documentar i descriure de manera estructurada les implicacions del disseny en la seguretat. La creació del modelatge d'amenaces és un treball en equip que implica els dissenyadors, els desenvolupadors, l'equip de proves, els tècnics en seguretat, etc. Aquesta és la tasca principal d'anàlisi de seguretat que es fa en la fase de disseny de programari, a la qual destinarem un capítol en aquest mòdul.

1.4. Desenvolupament

El desenvolupament implica escriure i depurar codi en què l'objectiu és escriure codi amb la màxima qualitat possible. La qualitat va lligada amb la seguretat; no es pot dir que un codi és de qualitat si no incorpora la seguretat. En aquest apartat s'han de considerar els punts següents:

- Eines de desenvolupament.
- Directrius de codificació segura.
- Anàlisi estàtica.

1) Eines de desenvolupament

L'equip de desenvolupament ha de definir una llista de les eines aprovades i de les comprovacions de seguretat que hi estan associades, com, per exemple, els advertiments i les opcions del compilador. Aquesta llista l'ha d'aprovar l'assessor o responsable de seguretat del projecte. En general, s'ha de procurar usar l'última versió de les eines aprovades per a aprofitar les noves proteccions i funcions d'anàlisi de seguretat.

2) Directrius de codificació segura

És necessari definir un conjunt mínim de normes de codificació, i també la manera com s'han d'analitzar les biblioteques, les funcions i les interfícies de programació d'aplicacions (API) que es poden utilitzar conjuntament en un projecte de desenvolupament de programari i, a més a més, s'han de prohibir les que es determini que no són segures.

3) Anàlisi estàtica

A banda de la revisió de codi feta manualment, s'ha de tenir en compte l'anàlisi estàtica, que proporciona una capacitat escalable per a la revisió de codi segur i pot ajudar a comprovar que se segueixen les polítiques de codificació segura.

Tot i així, cal ser conscients de les fortaleses i debilitats de les eines d'anàlisi estàtica i que és necessari afegir una revisió humana; per tant, les eines d'anàlisi estàtica s'han de considerar com a eines d'ajuda i no pas com a substitució d'altres mètodes d'anàlisi i revisió.

1.5. Proves

La funció de les proves de seguretat és verificar que el disseny del sistema i el codi poden resistir l'atac. Així doncs, en aquesta fase es duen a terme les tasques següents:

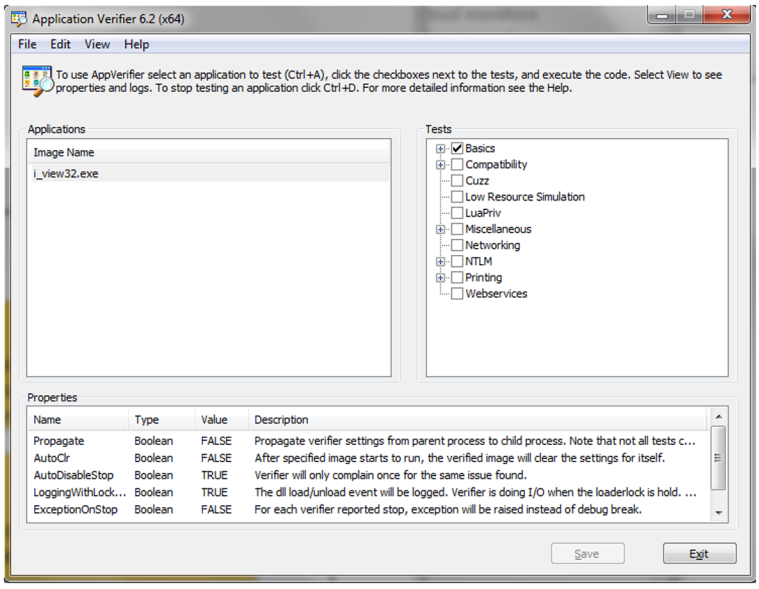
- Anàlisi dinàmica.
- Proves de penetració.
- Revisió del model d'amenaces.

1) Anàlisi dinàmica

És necessari comprovar el programari desenvolupat en temps d'execució per a assegurar que la funcionalitat es correspon amb el disseny.

En aquesta etapa s'ha de considerar l'ús d'eines, com, per exemple, AppVerifer[6], que supervisin el comportament de les aplicacions, i així poder detectar les possibles corrupcions de memòria, els problemes amb els privilegis dels usuaris o altres tipus de problemes de vulnerabilitat crítics. També s'ha de tenir en compte l'ús d'altres mètodes, com, per exemple, les proves d'exploració de vulnerabilitats amb dades aleatòries, dades amb format incorrecte o dades fora del rang previst.

[6] És una eina de Microsoft dissenyada específicament per a detectar errors (*bugs*) i ajudar a corregir corrupcions en la memòria i vulnerabilitats en la seguretat crítiques.



2) Proves de penetració

Aquest tipus de proves és de vital importància quan el programari s'utilitza en escenaris crítics. Es tracta de proves de caixa blanca que simulen les accions d'un *hacker*. L'objectiu d'aquestes proves és detectar vulnerabilitats degudes a errors de codificació, configuració o altres punts dèbils en la implementació.

3) Revisió del model d'amenaces

Sovint, una aplicació es desvia de manera significativa de les especificacions funcionals i de disseny definides en les fases de requisits i de disseny, i per tant és important fer una revisió del que es va elaborar en la fase de requisits i de disseny per a actualitzar el model d'amenaces segons els canvis que s'hagin produït. D'aquesta manera, es podran prendre en consideració les noves possibles amenaces i vulnerabilitats que poden aparèixer amb els canvis de disseny i es podran revisar i mitigar amb els criteris actualitzats.

1.6. Validació

La implementació requereix una sèrie de tasques per a poder posar finalment el programari en producció, entre les quals s'han de tenir en compte les següents:

- Elaborar el pla de respostes a incidents.
- Fer la revisió final de seguretat.

1) Elaborar el pla de respostes a incidents

Encara que s'hagin superat totes les proves, un cop s'ha posat l'aplicació en producció, pot passar que apareguin noves amenaces fins ara desconegudes o que no s'hagin previst o detectat alguns tipus d'amenaces existents. Així doncs, no s'ha de descartar que un cop el programari estigui en producció es detectin vulnerabilitats que s'hagin de corregir i, per tant, que s'hagi d'elaborar un pla de respostes a incidents. Aquest pla varia en funció del tipus d'organització i del tipus de programari; tot i així, però, a tall d'exemple podem indicar els següents:

- Identificar un equip de manteniment i d'emergències, si és necessari, que doni resposta a les vulnerabilitats.
- Idear un pla de serveis de seguretat per a codi heretat d'altres equips de l'organització.
- Preparar, si es dóna el cas, un pla de seguretat per a codi de tercers amb llicència.

2) Fer la revisió final de seguretat

Es tracta d'inspeccionar les activitats de seguretat fetes abans del llançament del programari. En aquest punt no es tracta de tornar a fer un test o de corregir errors, sinó d'avaluar els resultats de les eines, de les proves i del rendiment, tenint en compte els llindars de qualitat i límits d'errors prèviament determinats. La revisió de seguretat pot tenir tres resultats:

- **Revisió superada**. S'han corregit i mitigat tots els problemes de seguretat i privacitat identificats.

- **Revisió superada amb excepcions**. S'han corregit i mitigat els problemes de seguretat i privacitat identificats. Tot i així, es detecten problemes que no s'han pogut resoldre, els quals es registren i es preveuen corregir en una versió posterior.

- **Revisió amb remissió a una instància superior**. No es compleixen o no s'han corregit els problemes de seguretat i privacitat amb uns mínims acceptables. El responsable de seguretat no pot aprovar el projecte i no es pot posar en producció; per tant, s'ha de remetre a una instància superior per a la consegüent presa de decisions.

1.7. Manteniment

Al llarg de la vida del programari es van fent modificacions, sia correctives, adaptatives, evolutives o perfectives. Aquestes modificacions s'han de gestionar amb el que s'anomena **gestió del canvi**.

En aquesta fase de manteniment no ens hem d'oblidar de la seguretat, i s'han de continuar incorporant tots els elements de seguretat i privacitat de què hem tractat fins ara. En la gestió del canvi s'han d'incorporar totes les tasques i elements necessaris perquè les modificacions compleixin els requisits de seguretat i, a més a més, perquè l'aplicació continuï complint els llindars de seguretat i límits d'errors estipulats en el projecte.

1.8. Eines

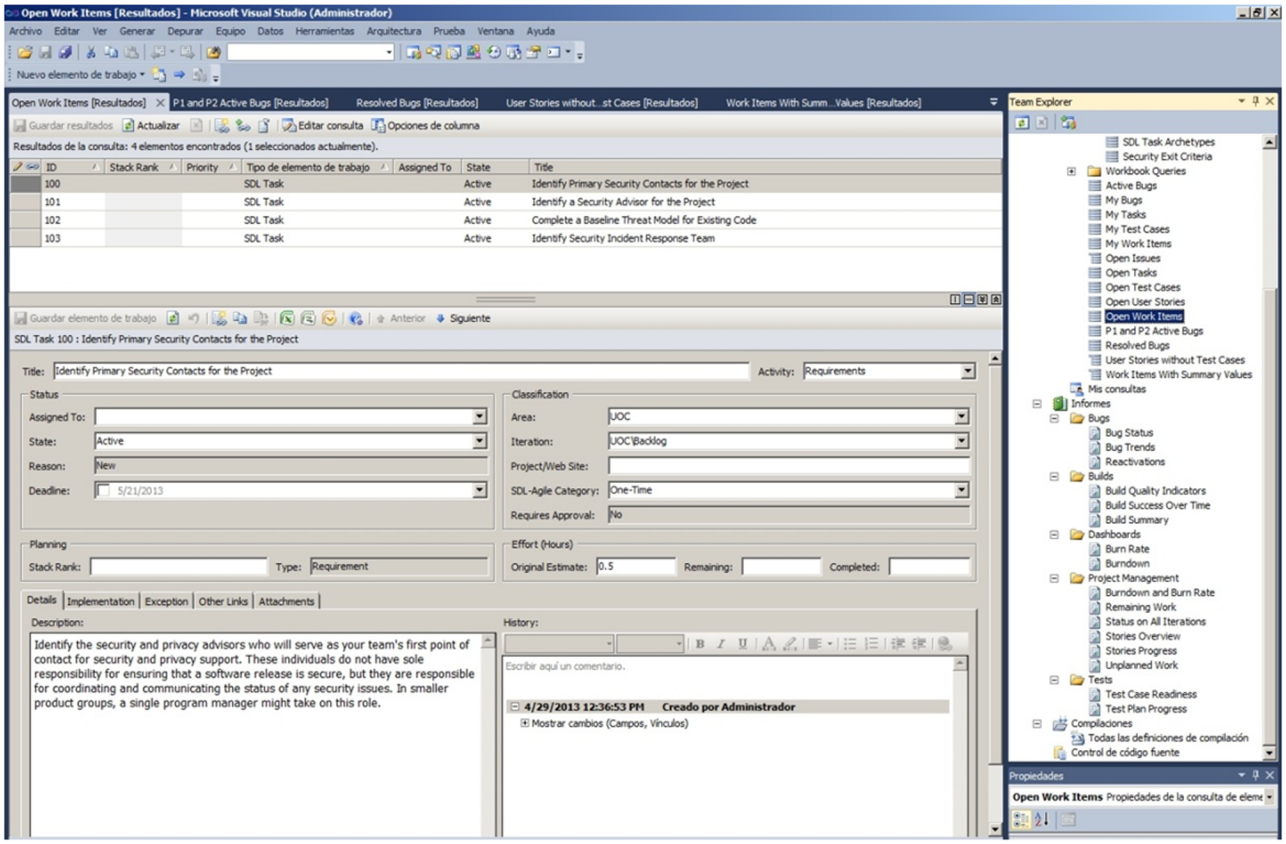
El fet d'incorporar la seguretat en el cicle de vida del programari implica gestionar, documentar, controlar i fer el seguiment de tot un conjunt de tasques durant tot el procés. En el mercat hi ha diverses eines per a les diverses plataformes de desenvolupament, les quals integren la seguretat en totes les fases, des de la fase de requisits fins a la fase de manteniment. A tall d'exemple, exposem l'eina de Microsoft MSF-Agile plus SDL Process Template for VSTS, que està disponible per a diverses versions de Visual Studio .NET amb Team Foundation Server. Es tracta d'unes plantilles per al model de metodologies àgils i per al model iteratiu i incremental amb les quals s'analitza el codi desenvo-

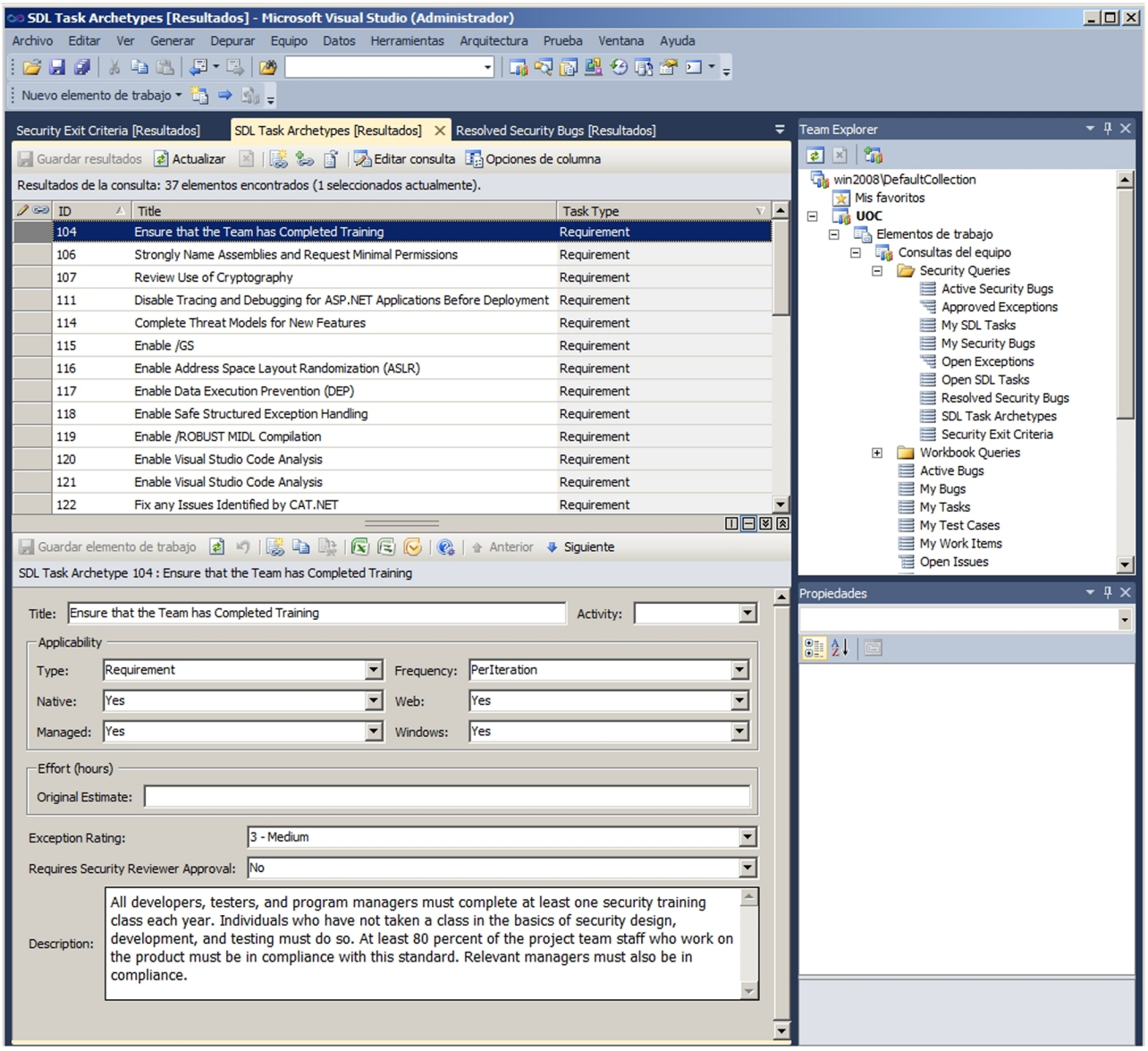
Bibliografia

Les plantilles MSF-Agile Plus SDL Process Template for VSTS es poden obtenir des del web de Microsoft.

Podeu ampliar la informació sobre aquestes plantilles al mateix web de Microsoft.

lupat per a assegurar que compleix les pràctiques de desenvolupament segur. Les plantilles també creen automàticament el flux de treball de seguretat dels elements de seguiment de processos com el del modelatge d'amenaces per a assegurar que aquestes activitats de seguretat importants no siguin omeses.




SDL Task Archetypes [Resultados] - Microsoft Visual Studio (Administrador)
Archivo Editar Ver Generar Depurar Equipo Datos Herramientas Arquitectura Prueba Ventana Ayuda
Nuevo elemento de trabajo
Security Exit Criteria [Resultados]
SDL Task Archetypes [Resultados]
Resolved Security Bugs [Resultados]
Guardar resultados Actualizar Editar consulta Opciones de columna
Resultados de la consulta: 37 elementos encontrados (1 seleccionados actualmente).
ID Title Task Type
104 Ensure that the Team has Completed Training Requirement
106 Strongly Name Assemblies and Request Minimal Permissions Requirement
107 Review Use of Cryptography Requirement
111 Disable Tracing and Debugging for ASP.NET Applications Before Deployment Requirement
114 Complete Threat Models for New Features Requirement
115 Enable /GS Requirement
116 Enable Address Space Layout Randomization (ASLR) Requirement
117 Enable Data Execution Prevention (DEP) Requirement
118 Enable Safe Structured Exception Handling Requirement
119 Enable /ROBUST MIDL Compilation Requirement
120 Enable Visual Studio Code Analysis Requirement
121 Enable Visual Studio Code Analysis Requirement
122 Fix any Issues Identified by CAT.NET Requirement
Guardar elemento de trabajo Anterior Siguiente
SDL Task Archetype 104 : Ensure that the Team has Completed Training
Title: Ensure that the Team has Completed Training
Activity:
Applicability
Type: Requirement
Frequency: PerIteration
Native: Yes
Web: Yes
Managed: Yes
Windows: Yes
Effort (hours)
Original Estimate:
Exception Rating: 3 - Medium
Requires Security Reviewer Approval: No
Description: All developers, testers, and program managers must complete at least one security training class each year. Individuals who have not taken a class in the basics of security design, development, and testing must do so. At least 80 percent of the project team staff who work on the product must be in compliance with this standard. Relevant managers must also be in compliance.
Team Explorer
win2008\DefaultCollection
Mis favoritos
UOC
Elementos de trabajo
Consultas del equipo
Security Queries
Active Security Bugs
Approved Exceptions
My SDL Tasks
My Security Bugs
Open Exceptions
Open SDL Tasks
Resolved Security Bugs
SDL Task Archetypes
Security Exit Criteria
Workbook Queries
Active Bugs
My Bugs
My Tasks
My Test Cases
My Work Items
Open Issues
Propiedades

2. Avaluació de riscos

S'ha de considerar que de risc, poc o molt, sempre n'hi ha. Aquest fet fa que en els projectes el risc s'hagi de gestionar i s'hagi de fer una anàlisi de riscos. Com que de risc n'hi ha, es tracta d'establir mecanismes d'observació i prevenció, plans per a mitigar o reduir els riscos i plans de contingències per al cas en què el risc deixi de ser risc, s'activi i es transformi en un problema real.

En el desenvolupament de programari, com en qualsevol projecte, les fases de l'anàlisi de riscos són les mateixes que en altres àmbits. Les metodologies que es poden aplicar són diverses; per exemple, Magerit, NIST, CRAMM o Octave. En general, les fases de l'anàlisi de riscos es poden resumir en les següents:

- Definir i valorar els actius afectats.
- Identificar les vulnerabilitats.
- Identificar les amenaces.
- Estudiar les salvaguardes.
- Determinar la probabilitat.
- Analitzar l'impacte.
- Determinar el nivell de risc.

A continuació exposarem breument algunes consideracions molt generals que s'han de tenir en compte en l'anàlisi de riscos en el cas del desenvolupament de programari.

D'entrada, per a fer una avaluació de riscos, s'ha d'assumir que el programari o el sistema serà atacat.

La quantitat de temps i esforç que destina un atacant a intentar trobar com atacar un sistema depèn de diversos factors, entre els quals es poden plantejar els següents:

- El valor de les dades que tracti el programari, com, per exemple, números de targetes de crèdit, dades personals que poden ser utilitzades, dades econòmiques.

- El programari, és d'ús d'una petita empresa o d'una multinacional?

- Quina serà la distribució del programari? D'ús exclusiu per a un client o es tracta d'una aplicació estàndard d'ús generalitzat?

- Quin serà l'entorn d'execució? En una xarxa local o d'accés per Internet?

Tenint en compte aquests factors, a més a més dels que es determinin, s'ha de decidir quin nivell de risc és acceptable. Per a una possible pèrdua de 1.000 euros, no justifica una inversió afegida en desenvolupament de 10.000 euros per a preveure tots els errors potencials de seguretat. Tot i tenir en compte aquest criteri, també s'ha de considerar el cost en concepte de pèrdua de reputació que pot patir l'empresa amb la conseqüent pèrdua de clients actuals i clients potencials.

Avaluar el risc

L'avaluació de riscos depèn molt del tipus de programari; tot i així, però, alguns dels factors que es poden considerar són els següents:

- Què és el pitjor que pot passar si l'aplicació és atacada amb èxit?
- Quin grau de dificultat tindria l'atacant per a aconseguir un atac amb èxit?
- Com seria de gran l'objectiu de l'atacant? Es tracta d'una aplicació amb un centenar de còpies venudes o està instal·lada per defecte en milers d'ordinadors?
- És vulnerable per defecte, ho és quan l'usuari fa un conjunt inusual d'opcions o es necessiten eines especialitzades?
- Quants usuaris es veurien afectats?
- Com s'accedeix a l'objectiu? L'execució, requereix accés a la xarxa local, accés local o accés per Internet?

Una avaluació de riscos dóna una idea de la probabilitat de ser atacats i del dany que podria causar un atac. El pas següent és esbrinar com es pot ser atacat, incloent-hi el sistema; no solament els atemptats al programari sinó també als servidors, a les dades, etc. Per a fer aquesta tasca es crea un model d'amenaces.

3. Modelatge d'amenaces

El modelatge d'amenaces és una tècnica que es pot utilitzar per a ajudar a identificar amenaces, atacs, vulnerabilitats i contramesures rellevants per a l'aplicació, de manera que permet determinar més bé els riscos a què pot ser exposat el programari i com es poden manifestar els atacs. L'objectiu és determinar quines amenaces requereixen que siguin mitigades i com es pot fer. L'activitat de modelatge d'amenaces ajuda a identificar el següent:

- Els objectius de seguretat.
- Les amenaces rellevants.
- Les vulnerabilitats rellevants i les contramesures que s'hi poden aplicar.

El modelatge d'amenaces, bàsicament, consisteix a revisar el disseny o l'arquitectura seguint una metodologia que faciliti trobar i corregir els problemes de seguretat actuals o futurs. Es tracta que els diferents actors que hi estan involucrats (desenvolupadors, equip de proves, gerència, administradors de sistemes, etc.) participin en el procés d'identificació de les possibles amenaces, tant prenent una actitud defensiva com posant-se en el paper d'un possible atacant. Seguint aquest enfocament, es força a tots aquests actors a explorar les debilitats que puguin sorgir o que ja siguin presents i a determinar les possibles contramesures.

A la vegada, el fet de dur a terme un modelatge d'amenaces permet identificar i complir els objectius de seguretat específics de cada entorn i facilita la priorització de tasques basant-se en el nivell de risc.

El modelatge d'amenaces és un procés iteratiu. S'ha de tenir en compte que és impossible identificar totes les possibles amenaces en una sola passada. A més a més, les aplicacions rarament són estàtiques sinó que es van millorant i adaptant a les noves necessitats de l'organització, i per tant, a mesura que el cicle de vida avança, s'afegeix més detall en el model:

- S'incrementa el detall del model a mesura que es descobreixen nous factors.

- Mentre es desenvolupa, les decisions de disseny i implementació revelen nous factors.

- Mitjançant les diverses fases del cicle de vida del desenvolupament del programari, s'ha de tenir en compte que poden aparèixer nous factors, fins i tot en la fase de manteniment, i també a l'hora de configurar el sistema i en l'ús posterior de l'aplicació.

3.1. Procés del modelatge d'amenaces

El procés del modelatge d'amenaces es compon de les fases següents:

- Identificar els actius.
- Definir l'arquitectura.
- Descompondre l'aplicació.
- Identificar les amenaces:
  - Identificar.
  - Documentar.
  - Valorar.
- Mitigar les amenaces.

3.1.1. Identificar els actius

Consisteix a identificar els actius que s'han de protegir. Aquests actius poden ser dades confidencials, bases de dades, pàgines web, disponibilitat del lloc web o del sistema, etc.

3.1.2. Definir l'arquitectura

L'objectiu d'aquesta etapa és documentar el funcionament de l'aplicació, l'arquitectura i configuració d'implementació i també les tecnologies que formen part de la solució; tot plegat, buscant possibles vulnerabilitats en el disseny o en la implementació de l'aplicació.

3.1.3. Descompondre l'aplicació

L'objectiu d'aquesta fase és aconseguir comprendre l'aplicació i saber com interactua amb les entitats externes, per a descobrir així les amaces amb més facilitat.

En aquesta etapa es fan les tasques següents:

- Identificar els límits de confiança.
- Analitzar i fer un diagrama de flux de dades.
- Identificar els punts d'entrada.
- Identificar codi privilegiat.
- Documentar el perfil de seguretat.

3.1.4. Identificar les amenaces

En aquet punt s'identifiquen les amenaces que poden afectar el sistema i comprometre els actius. Una de les maneres habituals de dur a terme aquesta tasca és reunir un grup amb membres dels equipis de desenvolupament i de proves i fer una sessió de pluja d'idees.

1) Identificar les amenaces

STRIDE[7] és un sistema desenvolupat per Microsoft per a classificar les amenaces de seguretat.

(7)STRIDE és l'acrònim de *spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege*.

Per a identificar les amenaces es poden utilitzar tres enfocaments bàsics:

a) Utilitzar STRIDE per a categoritzar les amenaces.

En el procés d'identificar amenaces és important fer-se preguntes com, per exemple, les següents:

- Un usuari no autoritzat, pot visualitzar dades confidencials?
- Un usuari amb privilegis només de lectura, pot modificar registres en la base de dades?
- Un usuari, pot fer ús d'algun component per a elevar els seus privilegis als d'administrador?

Per a facilitar la formulació d'aquest tipus de preguntes es pot utilitzar el sistema STRIDE agrupant aquestes preguntes en sis categories:

- ***Spoofing***: aquesta amenaça permet a un atacant fer-se passar per un altre, sia un usuari o un servidor.
- ***Tampering***: la manipulació de dades implica la modificació maliciosa de les dades.
- ***Repudiation***: aquesta amenaça implica que un usuari pot negar que ha fet una acció que en realitat ha fet, i no es pot demostrar que l'ha fet, és a dir repudiar.
- ***Information disclosure***: consisteix a divulgar informació a persones no autoritzades.
- ***Denial of service***: és la denegació de servei o accés a usuaris vàlids.

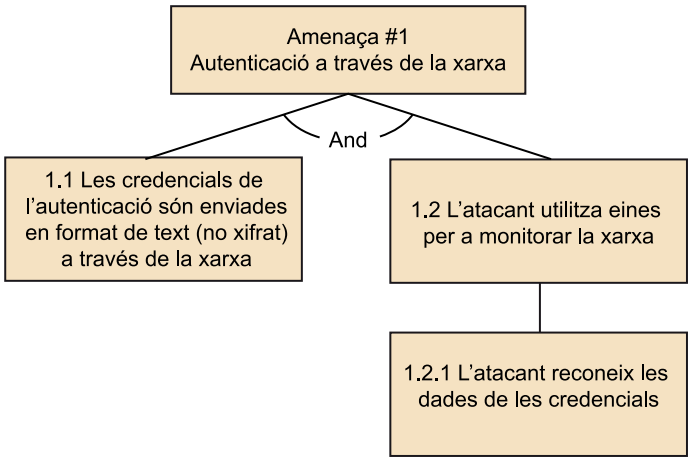
- ***Elevation of privilege***: en aquest cas, un usuari pot elevar els seus privilegis d'usuari i així fer tasques per a les quals no estava autoritzat.

b) Utilitzar llistes d'amenaces classificades

Amb aquest enfocament, primer de tot, es crea una llista d'amenaces comunes agrupades per tipus, per exemple per xarxa, per servidor i per aplicació. A continuació s'aplica la llista d'amenaces a la mateixa arquitectura de l'aplicació.

c) Arbres d'amenaces

Amb aquest sistema es tracta de descompondre una amenaça amb les diverses formes que pot utilitzar un atacant.



2) Documentar les amenaces

El pas següent en la identificació és documentar les amenaces. Per a fer-ho, s'ha d'utilitzar una plantilla en què s'inclouen els apartats següents:

- Descripció de l'amenaça.
- Objectiu de l'amenaça.
- Nivell de risc.
- Tècniques d'atac.
- Contramesures.

3) Valorar les amenaces

Un cop identificades i documentades, és necessari determinar les amenaces més importants per a prioritzar els esforços que hi cal destinar. Per a classificar les amenaces en funció de la importància que tenen, es fa una valoració de l'amenaça de manera que es pugui quantificar. Un cop valorades s'han de classificar en ordre decreixent.

Una manera senzilla de fer la valoració és puntuar la probabilitat i el dany potencial de l'amenaça amb valors del 0 al 10 i aplicar la fórmula següent:

$$\text{risc} = \text{probabilitat} \times \text{dany potencial}$$

En aquest cas s'obté una valoració del risc amb un interval del 0 al 100.

Una altra manera per a fer la valoració de l'amenaça és utilitzar el mètode DREAD[8]. En aquest cas es tracta d'afegir noves dimensions que ajudin a determinar quin és l'impacte que representa. Amb aquest mètode es valoren les qüestions següents per a cada amenaça:

[8]DREAD és l'acrònim de *damage potential, reproducibility, exploitability, affected users, discoverability*.

- **Dany potencial (*damage potential*)**: quina és la magnitud del dany que pot causar?
- **Reproductibilitat (*reproducibility*)**: és fàcil de reproduir l'atac?
- **Explotabilitat (*exploitability*)**: és fàcil que un atacant aconsegueixi explotar una vulnerabilitat?
- **Usuaris afectats (*affected users*)**: quants usuaris es veurien afectats?
- **Detectabilitat (*discoverability*)**: és fàcil trobar la vulnerabilitat?

En aquest tipus de valoració es poden utilitzar els valors alt, mitjà i baix, els quals es quantifiquen amb els valors 3, 2 i 1, respectivament.

Exemple

Amenaça	D	R	E	A	D	Total	Classificació
SQL injection	3	3	3	3	2	14	Alt

3.1.5. Mitigar les amenaces

L'objectiu de la gestió de riscos és reduir l'impacte que pot crear l'explotació d'una amenaça en l'aplicació o en el sistema. Això es pot fer amb una estratègia de mitigació de riscos per a respondre a les amenaces. En general, hi ha quatre opcions per a mitigar les amenaces:

- No fer res.
- Informar l'usuari de l'amenaça.
- Treure el problema.
- Solucionar el problema.

1) No fer res

Davant d'un risc baix, l'organització pot decidir adoptar aquesta opció. Tot i així, rarament és una bona opció perquè no sol ser la solució correcta, ja que el problema està latent en l'aplicació, i tard o d'hora serà descobert i al final s'haurà de solucionar el problema igualment.

2) Informar l'usuari de l'amenaça

L'alternativa d'informar l'usuari del problema, per exemple amb un quadre de diàleg, permet que aquest usuari pugui decidir si vol utilitzar la funció o no. Tot i així, aquesta opció pot ser problemàtica ja que l'usuari pot prendre la decisió menys encertada.

3) Treure el problema

Si no hi ha temps per a arreglar el problema, s'ha de considerar el fet de treure la funcionalitat problemàtica de l'aplicació i solucionar el problema per a la propera versió.

4) Solucionar el problema

El fet de seleccionar les tecnologies necessàries per a solucionar el problema, òbviament, és la millor opció, però també la més difícil, ja que implica més feina per als dissenyadors, els desenvolupadors i l'equip de proves.

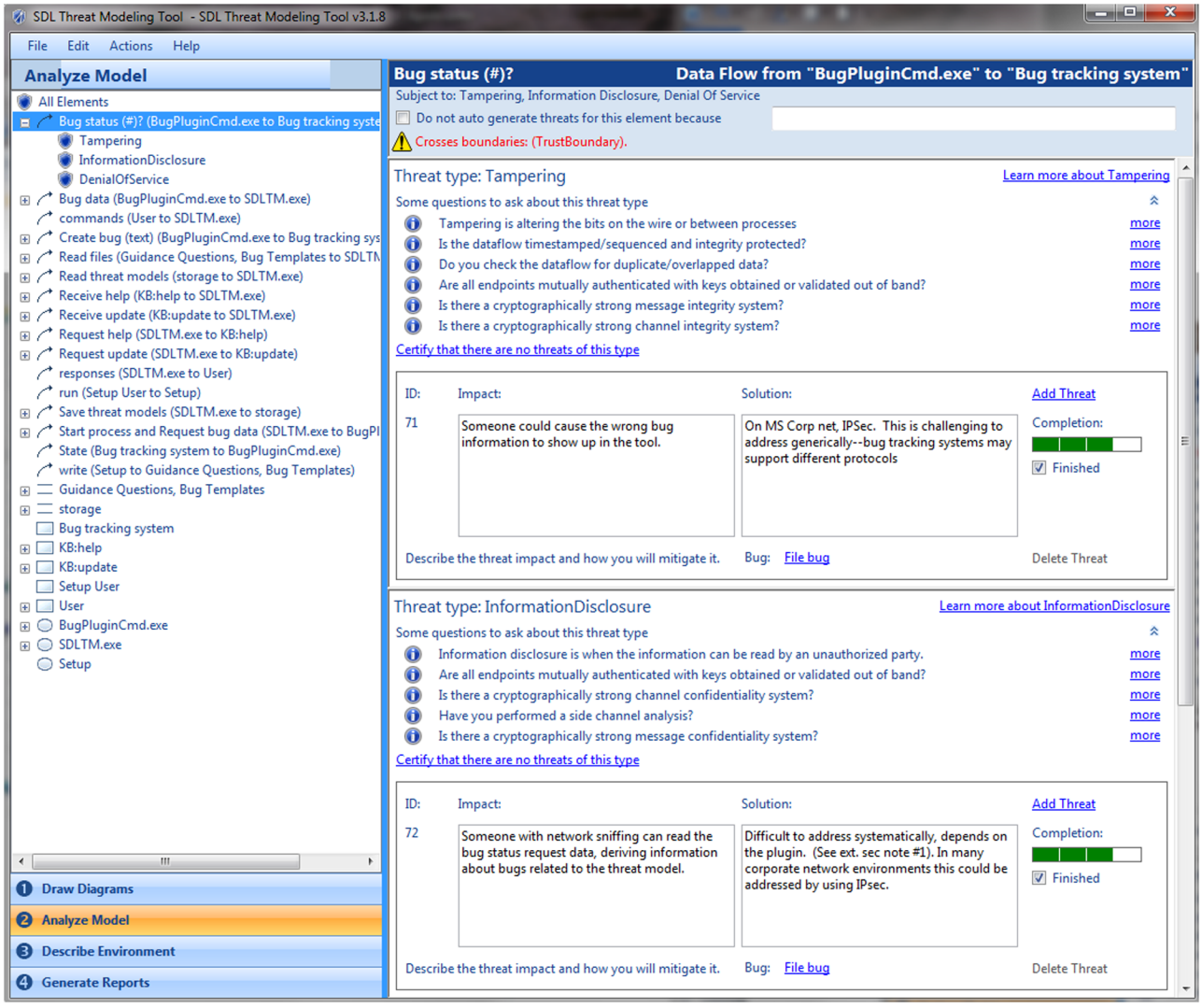
A continuació mostrarem una llista d'algunes tècniques de mitigació d'amenaces, utilitzant la classificació STRIDE:

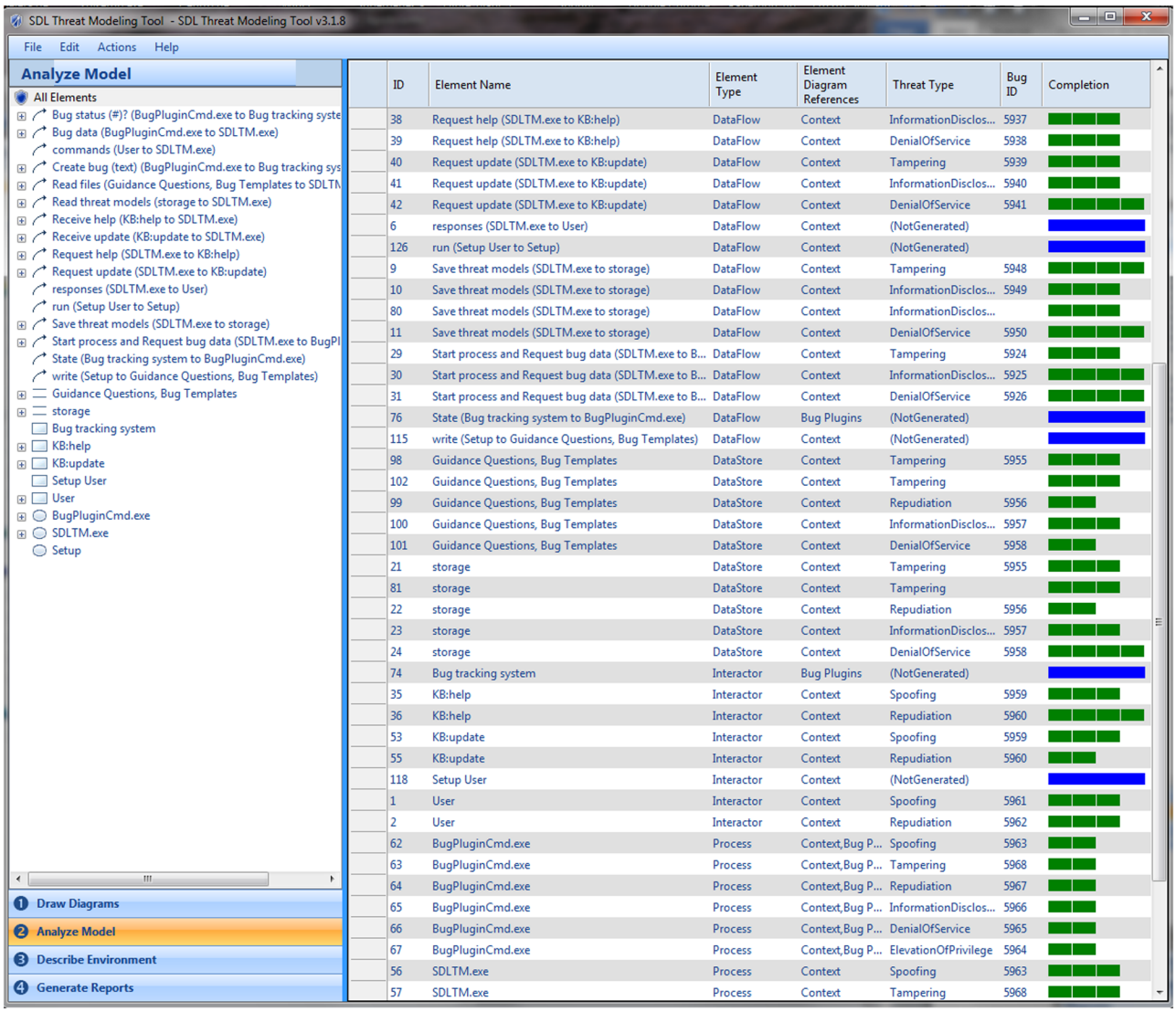
Tipus d'amenaça	Tècnica de mitigació
Spoofing	Autenticació Protecció de les dades confidencials
Tampering	Autorització Signatura digital *Hashes* *Tamper-resistant protocols*
Repudiation	Signatura digital
Information disclosure	Autorització *Privacy-enhanced protocols* Encriptació
Denial of service	Autenticació Autorització Filtratge Qualitat de servei (QoS)

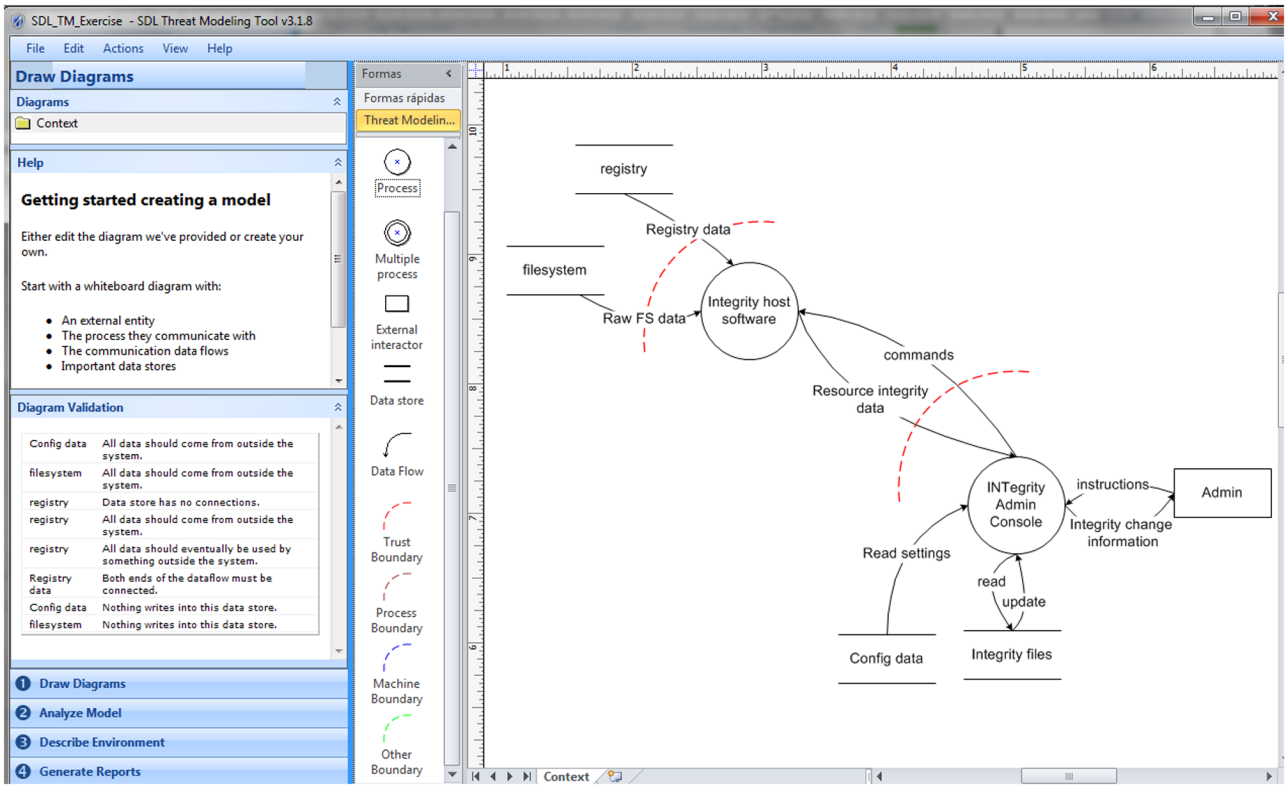
Tipus d'amenaça	Tècnica de mitigació
Elevation of privilege	Execució amb mínims privilegis

3.2. Eines

En les activitats del modelatge d'amenaces es requereix definir, identificar, documentar, elaborar diagrames, fer un seguiment de les activitats, etc. És a dir, és necessària una gestió completa i seguir metodologia. Hi ha eines per a facilitar que es puguin dur a terme totes les tasques i compartir a la vegada la informació amb tot l'equip que hi està involucrat. En aquest exemple es mostren les diverses opcions del programari SDL Threat Modeling Tool, que ha desenvolupat Microsoft. Aquest programari és gratuït, es pot baixar des del portal de Microsoft i permet dur a terme les diverses tasques que s'han de fer en el modelatge d'amenaces.




SDL Threat Modeling Tool - SDL Threat Modeling Tool v3.1.8
File
Edit
Actions
Help
Analyze Model
All Elements
Bug status (#)? (BugPluginCmd.exe to Bug tracking syste
Bug data (BugPluginCmd.exe to SDLTM.exe)
commands (User to SDLTM.exe)
Create bug (text) (BugPluginCmd.exe to Bug tracking sys
Read files (Guidance Questions, Bug Templates to SDLTM
Read threat models (storage to SDLTM.exe)
Receive help (KB:help to SDLTM.exe)
Receive update (KB:update to SDLTM.exe)
Request help (SDLTM.exe to KB:help)
Request update (SDLTM.exe to KB:update)
responses (SDLTM.exe to User)
run (Setup User to Setup)
Save threat models (SDLTM.exe to storage)
Start process and Request bug data (SDLTM.exe to BugPl
State (Bug tracking system to BugPluginCmd.exe)
write (Setup to Guidance Questions, Bug Templates)
Guidance Questions, Bug Templates
storage
Bug tracking system
KB:help
KB:update
Setup User
User
BugPluginCmd.exe
SDLTM.exe
Setup
1 Draw Diagrams
2 Analyze Model
3 Describe Environment
4 Generate Reports
ID
Element Name
Element Type
Element Diagram References
Threat Type
Bug ID
Completion
38 Request help (SDLTM.exe to KB:help) DataFlow Context InformationDisclos... 5937
39 Request help (SDLTM.exe to KB:help) DataFlow Context DenialOfService 5938
40 Request update (SDLTM.exe to KB:update) DataFlow Context Tampering 5939
41 Request update (SDLTM.exe to KB:update) DataFlow Context InformationDisclos... 5940
42 Request update (SDLTM.exe to KB:update) DataFlow Context DenialOfService 5941
6 responses (SDLTM.exe to User) DataFlow Context (NotGenerated)
126 run (Setup User to Setup) DataFlow Context (NotGenerated)
9 Save threat models (SDLTM.exe to storage) DataFlow Context Tampering 5948
10 Save threat models (SDLTM.exe to storage) DataFlow Context InformationDisclos... 5949
80 Save threat models (SDLTM.exe to storage) DataFlow Context InformationDisclos...
11 Save threat models (SDLTM.exe to storage) DataFlow Context DenialOfService 5950
29 Start process and Request bug data (SDLTM.exe to B... DataFlow Context Tampering 5924
30 Start process and Request bug data (SDLTM.exe to B... DataFlow Context InformationDisclos... 5925
31 Start process and Request bug data (SDLTM.exe to B... DataFlow Context DenialOfService 5926
76 State (Bug tracking system to BugPluginCmd.exe) DataFlow Bug Plugins (NotGenerated)
115 write (Setup to Guidance Questions, Bug Templates) DataFlow Context (NotGenerated)
98 Guidance Questions, Bug Templates DataStore Context Tampering 5955
102 Guidance Questions, Bug Templates DataStore Context Tampering
99 Guidance Questions, Bug Templates DataStore Context Repudiation 5956
100 Guidance Questions, Bug Templates DataStore Context InformationDisclos... 5957
101 Guidance Questions, Bug Templates DataStore Context DenialOfService 5958
21 storage DataStore Context Tampering 5955
81 storage DataStore Context Tampering
22 storage DataStore Context Repudiation 5956
23 storage DataStore Context InformationDisclos... 5957
24 storage DataStore Context DenialOfService 5958
74 Bug tracking system Interactor Bug Plugins (NotGenerated)
35 KB:help Interactor Context Spoofing 5959
36 KB:help Interactor Context Repudiation 5960
53 KB:update Interactor Context Spoofing 5959
55 KB:update Interactor Context Repudiation 5960
118 Setup User Interactor Context (NotGenerated)
1 User Interactor Context Spoofing 5961
2 User Interactor Context Repudiation 5962
62 BugPluginCmd.exe Process Context,Bug P... Spoofing 5963
63 BugPluginCmd.exe Process Context,Bug P... Tampering 5968
64 BugPluginCmd.exe Process Context,Bug P... Repudiation 5967
65 BugPluginCmd.exe Process Context,Bug P... InformationDisclos... 5966
66 BugPluginCmd.exe Process Context,Bug P... DenialOfService 5965
67 BugPluginCmd.exe Process Context,Bug P... ElevationOfPrivilege 5964
56 SDLTM.exe Process Context Spoofing 5963
57 SDLTM.exe Process Context Tampering 5968


SDL_TM_Exercise - SDL Threat Modeling Tool v3.1.8
File Edit Actions View Help
Draw Diagrams
Diagrams
Context
Help
Getting started creating a model
Either edit the diagram we've provided or create your own.
Start with a whiteboard diagram with:
An external entity
The process they communicate with
The communication data flows
Important data stores
Diagram Validation
Config data All data should come from outside the system.
filesystem All data should come from outside the system.
registry Data store has no connections.
registry All data should come from outside the system.
registry All data should eventually be used by something outside the system.
Registry data Both ends of the dataflow must be connected.
Config data Nothing writes into this data store.
filesystem Nothing writes into this data store.
1 Draw Diagrams
2 Analyze Model
3 Describe Environment
4 Generate Reports
Formas
Formas rápidas
Threat Modelin...
Process
Multiple process
External interactor
Data store
Data Flow
Trust Boundary
Process Boundary
Machine Boundary
Other Boundary
registry
Registry data
filesystem
Raw FS data
Integrity host software
commands
Resource integrity data
INTegrity Admin Console
instructions
Integrity change information
Admin
Read settings
read
update
Config data
Integrity files
Context

4. Tècniques de seguretat

En aquest apartat definirem algunes de les tècniques de seguretat que es poden aplicar en els sistemes d'informació. Es tracta de mostrar un breu resum de cada tecnologia sense entrar-hi a fons.

1) Autenticació

L'autenticació és el procés pel qual una entitat verifica que una altra entitat és el qui diu que és o el que diu que és. S'ha de tenir en compte que una entitat pot ser un usuari, un codi executable o un ordinador. L'autenticació requereix evidència en forma de credencials i proves, les quals poden ser de moltes maneres, com una contrasenya, una clau privada o, en el cas de l'autenticació biomètrica, una empremta dactilar.

2) Autorització

Un cop s'ha autenticat, en general aquesta entitat voldrà accedir als recursos que li permeti l'aplicació, com, per exemple, dades, fitxers, impressores o funcionalitats. En una mateixa aplicació, molts cops no tots els usuaris tenen el mateix nivell de privilegis i, per tant, l'aplicació ha de comprovar si l'usuari està autoritzat a fer una tasca o accedir a un recurs determinat. Una bona opció és que, depenent de l'usuari, l'aplicació mostri només les opcions i els recursos en funció dels privilegis de l'usuari i només pugui accedir a allò a què està autoritzat. Alguns dels mecanismes d'autorització són els següents:

- Llistes de control d'accés.
- Restriccions d'IP.
- Permisos de servidor.

a) Llistes de control d'accés

La implementació d'una *access control list* (ACL) es basa en una llista de permisos que s'associen a un objecte o recurs (dades, carpetes, fitxers, opcions, etc.) en funció de l'usuari o grup d'usuaris. En general, els sistemes operatius implementen aquest concepte, i en funció de l'aplicació cal valorar si s'ha de crear una llista de control d'accés per a la mateixa aplicació o utilitzar la que proporciona el sistema operatiu.

b) Restriccions d'IP

En aquest cas es tracta de donar o denegar accés als recursos en funció de l'adreça IP, de les subxarxes o del DNS.

c) Permisos de servidor

Molts servidors ofereixen la seva pròpia manera de control d'accés per a protegir els seus objectes. Per exemple, els servidors de bases de dades permeten a l'administrador determinar qui té accés a les taules, a les funcions, als procediments o a les vistes.

3) *Tamper-Resistant and Privacy-Enhanced Technologies*

La protecció a la manipulació i a la privacitat es refereix a la capacitat de protegir les dades a ser manipulades sia de manera maliciosa o accidental.

Alguns dels protocols i de les tecnologies que permeten aquesta protecció són els següents:

- SSL/TLS.
- IPsec.
- DCOM i RPC.
- Sistemes de fitxers xifrats.

a) SSL/TLS

Són protocols criptogràfics que proporcionen comunicacions segures per xarxa. Les dades són encriptades amb un codi d'autenticació del missatge per a proporcionar integritat a les dades. TLS és la versió d'SSL ratificada per la Internet Engineering Task Force (IETF).

b) IPsec

La funció d'aquest conjunt de protocols és la d'assegurar les comunicacions sobre el protocol IP autenticant o xifrant cada paquet IP.

c) DCOM

Distributed Component Object Model (DCOM) és una tecnologia propietària de Microsoft per a desenvolupar components de programari distribuït entre diversos ordinadors que es comuniquen entre si. L'aparició del *framework* .NET de Microsoft fa que les noves aplicacions en entorn Windows es decantin cap a utilitzar el *framework* .NET; tot i així, però, encara hi ha moltes implementacions que utilitzen DCOM.

d) Sistemes de fitxers xifrats

Són sistemes de fitxers especialitzats en encriptació que permeten als usuaris emmagatzemar dades xifrades en el disc dur de manera transparent. Alguns dels sistemes de fitxers xifrats són els següents:

- EFS, en el cas de Microsoft Windows Server, que opera en l'àmbit de sistema de fitxers.
- CFS, en el cas d'Unix, que opera en l'àmbit d'aplicació.
- TCFS, en el cas de Linux, que opera en l'àmbit de nucli.

4) Signatura digital

La signatura digital és una aplicació de la criptografia de clau pública que permet donar autenticitat d'origen a la informació enviada, assegurar-ne la integritat i impedir el repudi del qui signa.

Propietats de la signatura digital:

- Autenticitat: la signatura autentica el document.

- No-repudi: el qui ha signat un document no pot negar que l'ha signat.

- Integritat: el contingut del missatge juntament amb la clau privada s'utilitza per a crear la signatura. En cas de modificació posterior del missatge, la verificació de la signatura amb el nou missatge modificat no serà correcta.

A continuació mostrarem com opera una signatura digital:

Una **funció *hash*** o **funció resum** fa correspondre a un missatge m de mida variable una representació $H(m)$ de mida fixa.

Els algorismes més utilitzats són els MD5 i SHA-1.

Propietats:

- Funcionen amb entrades de mida variable.
- Longitud fixa del resum generat.
- Funció d'un únic sentit, és a dir que a partir del resum generat no es pot deduir l'entrada original.
- Funció amb distribució uniforme de les col·lisions (una col·lisió es produeix quan dues entrades diferents generen el mateix resum).

Signar:

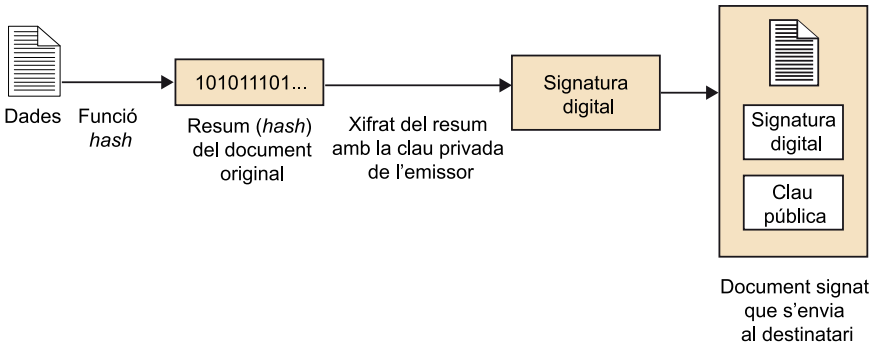
L'usuari A, per a signar un missatge m, aplica la funció *hash* H al missatge m i obté $H(m)$.

Signa $H(m)$ amb la seva clau privada SK i obté la signatura $s(m)$, en què $s(m) = SK(H(m))$.

S'afegeix $s(m)$ a m, i forma la signatura.

S'afegeix la clau pública PK de l'usuari A, perquè el destinatari pugui comprovar la signatura.

S'envia al destinatari el paquet format per (m, $s(m)$, PK).

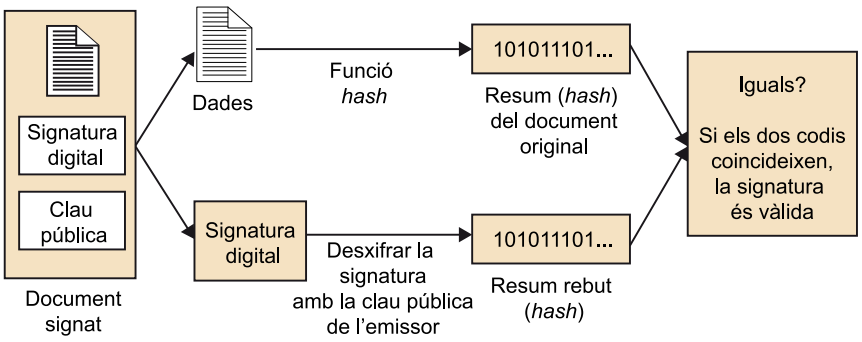


Verificar:

L'usuari B rep el missatge de A, calcula $H(m)$ i obté el resum 1.

Desxifra $s(m)$ amb la clau pública de A i obté el resum 2.

Es comparen els resums 1 i 2; si són iguals indica que el document és l'original signat. Com que les accions de *PK* i *SK* són inverses, el verificador podrà comprovar que el resultat sigui exactament el missatge signat.



5) Auditoria

L'objectiu de l'auditoria és recollir informació sobre el comportament, l'accés o denegació d'accés als recursos i objectes, l'ús de privilegis i altres accions de seguretat importants. Aquesta informació és emmagatzemada en un fitxer de registres (fitxer *log*) per a analitzar-la més endavant.

És important protegir aquests fitxers contra els possibles atacs, ja que l'atacant podria obtenir informació molt valuosa per a fer els seus atacs.

6) Filtratge i qualitat de servei

El filtratge implica inspeccionar les dades a mesura que es reben per a prendre la decisió d'acceptar o rebutjar el paquet. Aquesta funcionalitat és una de les que té el tallafoc i hi ha moltes amenaces que es poden mitigar amb aquest filtratge.

La qualitat del servei (Quality of Service) és un conjunt de components que permeten proporcionar un tractament específic a determinats tipus de trànsit.

7) Mínims privilegis

Una bona pràctica és la d'executar els processos just amb els privilegis necessaris i imprescindibles per a fer la tasca en qüestió.

Bibliografia

Apple Inc. (2013). *Risk Assessment and Treat Modeling* [en línia]. https://developer.apple.com/library/mac/#documentation/security/Conceptual/Security_Overview/ThreatModeling/ThreatModeling.html

Howard, M.; LeBlanc, D. (2002). *Writing Secure Code* (2a. ed.). Redmond (Washington): Microsoft Press.

Meier, J. D.; Mackman, A. (2005). *Security Engineering Explained: Paterns and Practices*. Redmond (Washington): Microsoft Press.

Microsoft (2013). *Microsoft Security Development Lifecycle (SDL) Process Guidance* [en línia]. http://msdn.microsoft.com

Microsoft (2013). *Security and Identity* [en línia]. http://msdn.microsoft.com

Microsoft (2013). *Threat Modeling* [en línia]. http://msdn.microsoft.com

The Open Web Application Security Project, OWASP (2013). *Application Threat Modeling* [en línia]. https://www.owasp.org/index.php?title=Application_Threat_Modeling&setlang=en

The Open Web Application Security Project, OWASP (2013). *Security Code Review in the SDL* [en línia]. https://www.owasp.org/index.php/Security_Code_Review_in_the_SDLC

The Open Web Application Security Project, OWASP (2013). *Threat Risk Modeling* [en línia]. https://www.owasp.org/index.php/Threat_Risk_Modeling