Universitat Oberta de Catalunya

uoc.edu

Security checklists amb SCAP

Francesc Xavier Dantí Epinasa
MISTIC
Treball Final de Màster en Seguretat Empresarial

Consultor: Pau del Canto Rodrigo
Profesor: Victor Garcia Font

4 de juny de 2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Security checklists amb SCAP*
Nombre del autor:	*Francesc Dantí Espinasa*
Nombre del consultor/a:	Pau del Canto Rodrigo
Nombre del PRA:	Victor Garcia Font
Fecha de entrega (mm/aaaa):	**04/06/2019**
Titulación::	MISTIC
Área del Trabajo Final:	*M1.749 - TFM-Seguretat empresarial*
Idioma del trabajo:	Català
Palabras clave	*SCAP OpenSCAP security guides SGG checklist remediation customization baselines*

Resum del Treball:

Aquest treball consisteix en l'estudi del framework SCAP i com, amb les eines que ens ofereix, es creen les Security Checklists.

Veurem com SCAP defineix un marc idiomàtic per a que tots els sistemes relacionats amb la seguretat informàtica es puguin entendre: com definir els components de software, com definir les vulnerabilitats, els patch, els ajustos de configuració, etcètera.

Veurem com amb un llenguatge declaratiu com el XCCDF es poden crear SSG (SCAP Security Checklists) que seran re-aprofitables entre diferents sistemes i com s'hi poden definir els checks, ajustar-los per fer-los personalitzats a la normativa de cada necessitat (empresa o particular) i com es poden aprofitar per tal de definir accions per a solucionar els petits errors de configuració dels sistemes.

Veurem un exemple de com aplicar una avaluació en sistemes reals i com les accions de fix o «remediation» solucionen gran part dels problemes detectats. Podrem veure fins on arriben aquestes accions i què no poden solucionar.

Abstract:

This work consists in the study of the SCAP framework and how Security Checklists are created using it.

We will see how SCAP defines a language so that all systems related to computer security can be integrated: how to enumerate software components, vulnerabilities, patches, configuration settings, and so on.

We will see how XCCDF, a declarative language, can be used to create SSG (SCAP Security Checklists) that will be reusable between different systems and how you can define checks, adjust them to make them customized to the regulations of each security policy and how can be used to define actions to solve small system configuration errors.

We will see an example of how to apply an evaluation in real systems and how the actions of fix or remediation solve most of the problems detected. We can see what can be solved and what not wih this automated processes.

Índex

Llista de figures

1) Introducció

En aquest primer apartat explicarem el context del projecte: què hem volgut aconseguir amb aquest projecte i com ho hem volgut fer. En el capítol de conclusions hi descriurem fins on hem arribat.

1.1) Context i justificació

En un entorn on la seguretat informàtica està en constant evolució i creixement, és imprescindible dotar-nos de sistemes que ens permetin automatitzar les comprovacions a fer en els sistemes informàtics. Ja sigui per garantir que les configuracions satisfan els requisits de la normativa de seguretat, com per auditar si tenim sistemes que no estan al nivell de patch que seria d'esperar.

Un cop definida la política i normativa de seguretat d'una empresa, els responsables de seguretat s'ocuparan de crear una relació de comprovacions a fer en els seus sistemes per tal de poder verificar que els sistemes compleixen la normativa establerta.

Així, i només a títol d'exemple, un responsable de seguretat podria definir una relació de comprovacions en format de llista de checks. Per exemple:

a) No es permet l'accés SSH com a usuari 'root'.
b) La paraula d'accés dels usuaris ha de ser de més de 8 caràcters.
c) Totes les màquines tindran dos servidors NTP en xarxes diferents.
d) No s'admetran fitxers amb SUID si no formen part d'un paquet del sistema.
e) ...etcètera.

Podem imaginar-nos que en qualsevol empresa, aquesta relació de comprovacions a fer serà molt extensa. També és d'esperar que aquestes comprovacions s'hagin de fer periòdicament: cada cop que es fa algun canvi en un sistema o cada cop que s'ha de superar una auditoria. Quant de temps suposaria que els administradors de sistemes verifiquessin tot això a mà?

Durant molts anys, aquesta tasca s'ha fet de forma pràcticament manualment o automatitzada en part. En el primer cas, la tasca era físicament impossible en entorns gaire grans. En el segon cas, treballant amb eines basades en estats (ansible, puppet, SUSE Manager) s'ha pogut automatitzar fins a cert punt aquestes tasques, però sempre s'acabava amb scripts que depenien del sistema i no eren fàcilment exportables a altres entorns.

La solució passa per SCAP i les SCAP Security Guides (SSG). SCAP és un llenguatge que estandarditza tots els conceptes relacionats amb la seguretat informàtica. Les SSG, per la seva banda, són documents XCCDF en format XML que permeten crear checklists en un format compatible amb SCAP i, per tant: universals, exportables i personalitzables. Amb aquests formats, aquestes llistes són executables en qualsevol sistema de forma senzilla i programable.

Amb el mateix afany d'estalviar temps i aprofitar la expertesa dels qui dominen un sistema operatiu, aquestes comprovacions poden incloure les solucions als problemes coneguts per errors de configuració o deguts a problemes de seguretat.

En aquest Treball Final de Màster, farem una introducció a aquesta metodologia, veurem com es treballa amb ella, com s'aplica una política concreta de les SSG a un sistema i com es poden ajustar per personalitzar-les a les nostres necessitats.

Veurem que existeixen SSG públiques creades per experts en seguretat informàtica, per la comunitat i pels mateixos fabricants de software i sistemes operatius. Veurem com podem aprofitar-les pels nostres sistemes, quines diferències hi ha entre elles i fins a quin punt podem aprofitar-les per adaptar la feina feta als nostres sistemes.

També aprofitarem i treballarem amb els fix proposats per aquestes checklist i veurem fins a quin punt són capaces de solucionar gran part dels problemes de seguretat que es detecten en els sistemes.

1.2) Objectius del treball

En aquest Treball Final de Màster pretenem:

• Documentar la tecnologia SCAP, en concret de OpenSCAP i les SCAP Security Guides. Definició dels conceptes més importants, exemples i formes d'ús. Tenir clar l'estat de l'art d'aquesta tecnologia. Fins on arriba, quines alternatives hi ha al mercat i cap a on sembla que avança el sector en aquest sentit. Encaixa OpenSCAP amb els solucions que proposen els fabricants de sistemes operatius?

• Implementació d'un entorn virtual de laboratori, amb diferents sistemes operatius i aplicacions, a on poder fer les proves de funcionament de OpenSCAP. Aquest entorn de laboratori ha de permetre comprovar el bon funcionament del OpenSCAP amb comprovacions sobre CVE concrets i sobre polítiques predefinides.

• Ajustar o crear una política de seguretat que pugui aplicar-se a alguns o tots els sistemes operatius i comprovar-ne el bon funcionament en l'entorn de laboratori.

• Documentar tota la informació recopilada, els resultats obtinguts i les conclusions que hem tret de l'estudi.

1.3) Enfoc i metodologia

La experiència en el món de la tecnologia ens demostra que com millor s'aprenen els conceptes és practicant. Per aquest motiu, enfocarem la metodologia de treball de forma que s'entrellaçarà l'estudi amb una primera implementació en un únic equip. D'aquesta manera, aprendrem abans els problemes i limitacions de la tecnologia.

Un cop tinguem clars els conceptes, crearem el laboratori virtual i, en aquest, executarem les proves inicials per a comprovar que el sistema funciona.
Seguidament implementarem alguna política de seguretat i, finalment, les executarem en l'entorn de laboratori.

1.4) Llistat de tasques

Tenint en compte que podrem dedicar a aquest TFM un màxim de 10 hores setmanals de mitjana, planificarem les tasques de la següent forma:

• T1. Estudi previ. Conceptes clau, limitacions de la tecnologia, estat de l'art, alternatives. Temps Necessari Estimat (TNE): tres setmanes i mitja.
• T2. Implementació d'un entorn en el pc personal. TNE: una setmana.
• T3. Creació d'un entorn de proves. Amb un mínim de màquines virtuals per assegurar que el sistema funciona en diferents OS i diferents nivells d'actualitzacions. TNE: una setmana.
• T4. Implementació i comprovació d'un OpenSCAP en l'entorn de proves. Inclou la verificació amb plantilles preconfigurades i tests sobre CVE concrets. TNE: dues setmanes.
• T5. Creació de polítiques i aplicació en l'entorn de proves. TNE: tres setmanes.
• T6. Documentació de la memòria. TNE: cinc setmanes.

1.5) Planificació

Al iniciar el semestre vam fer una planificació molt optimista sobre el rendiment que tindríem amb les tasques T3, T4 i T5. La planificació original, entregada en la PAC1 era la següent:

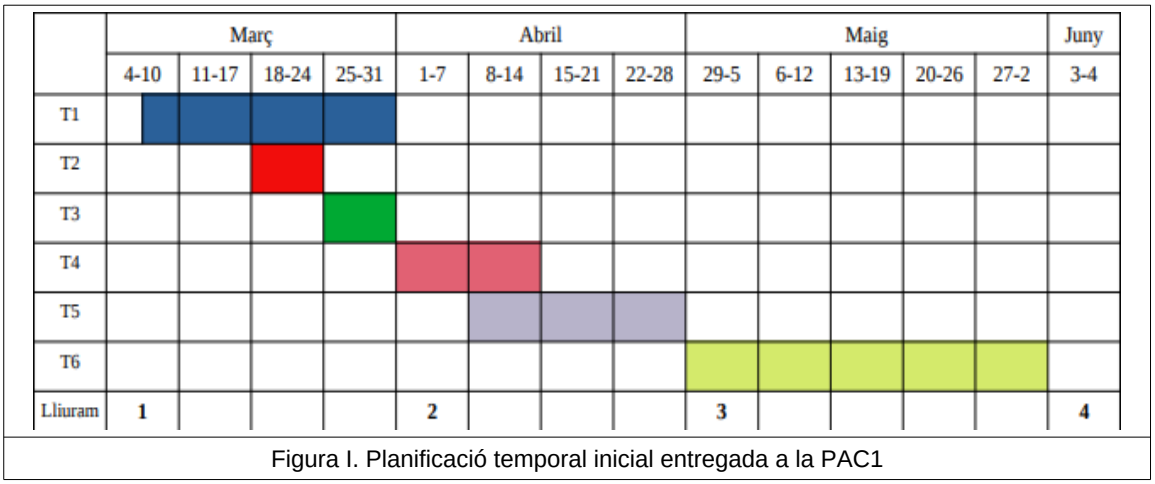

Figura I. Planificació temporal inicial entregada a la PAC1

La realitat és que la tasca T3 (Creació de l'entorn de proves) ens va portar forces mals de cap. L'entorn inicial s'havia dut a terme amb una màquina host basada en OpenSUSE, però un cop teníem les màquines virtuals creades vam veure que no podíem usar la paqueteria de SUSE per a treballar amb sistemes remots. Al ser un imperatiu laboral que tots els sistemes s'han de poder muntar amb paqueteria estàndard, vam decidir canviar de sistema operatiu basi i refer tota la infraestructura. Això ens va comportar una gran inversió de temps. Tocarem aquest assumpte en el capítol 2, de creació de l'entorn de treball.

A més, motius personals imprevistos ens han fet reduir el rendiment durant el mes de maig, fet que ens ha allargat els temps estimats per la T5 a pràcticament quatre setmanes i mitja.

Aquest fet va provocar que les tasques posteriors, que consistien en provar el OpenSCAP en l'entorn de proves i la creació de polítiques personalitzades, no es poguessin executar en paral·lel i que tota la planificació quedés desplaçada.

Per altra banda, la previsió inicial de com es redactaria la memòria s'ha demostrat totalment desencertada. Des del primer dia es comença a escriure la memòria, ja que gran part de la PAC1 ha acabat sent el capítol 1 de la memòria, així com la PAC2 i PAC3 ho són del capítol 2 i 3. Sigui com sigui, en la planificació hi representem les setmanes en que realment hem treballat amb la memòria.

El consell del tutor, de començar a redactar la memòria des del primer dia es comprova molt encertada i, segurament gràcies a aquest consell, podem fer la present entrega.

La realització temporal real del TFM ha estat doncs així, dedicant una mitjana de 5 hores setmanals al Treball.

	Març				**Abril**				**Maig**					**Juny**
	4-10	11-17	18-24	25-31	1-7	8-14	15-21	22-28	29-5	6-12	13-19	20-26	27-2	3-4
T1														
T2														
T3														
T4														
T5														
T6														

Figura II. Temporalització real de les tasques del TFM

2) Entorn de treball

Per tal de dur a terme totes les proves necessàries per generar aquesta memòria hem invertit força temps en generar un entorn de treball adequat, amb una varietat suficient de màquines virtuals com per poder fer proves en entorns molt diferenciats.

Al fer-ho, ens hem trobat amb algunes particularitats que ens han fet invertir temps en entendre perquè les coses no sempre són tan senzilles com semblen.

A fi i efecte que aquest temps invertit quedi reflectit per tal que qui llegeixi aquesta memòria pugui estalviar-se aquest temps, creiem necessari documentar breument el procés per a crear l'entorn virtual i explicar aquells problemes i dubtes que ens han anat sorgint.

2.1) Màquines virtuals amb Virtualbox

Un entorn virtual ens assegura poder disposar de tantes màquines com necessitem, executades en un hipervisor local o en un hipervisor remot.

En un entorn de treball final de màster que no es realitza en un entorn professional i en el que hem executat sempre un màxim de una o dues màquines a la vegada, la solució de Virtualbox local ha satisfet plenament els requisits marcats per a l'entorn de treball. Aquest sistema, mitjançant un mòdul de kernel especialment dissenyat, permet la execució concurrent de diferents màquines virtuals sobre el kernel de Linux, de forma que es poden executar múltiples màquines virtuals sobre el mateix maquinari.

Un cop instal·lat el programari de Virtualbox en la màquina host, que pot ser un pc qualsevol, tant amb Linux, Mac o Windows, podem afegir màquines virtuals noves tot ajustant la memòria, cpu i disc per cadascuna d'elles. Un cop fet això, s'introdueix la ISO d'instal·lació del sistema operatiu com is fos un DVD estàndard i es fa la instal·lació com si es traces d'una màquina física normal i corrent.

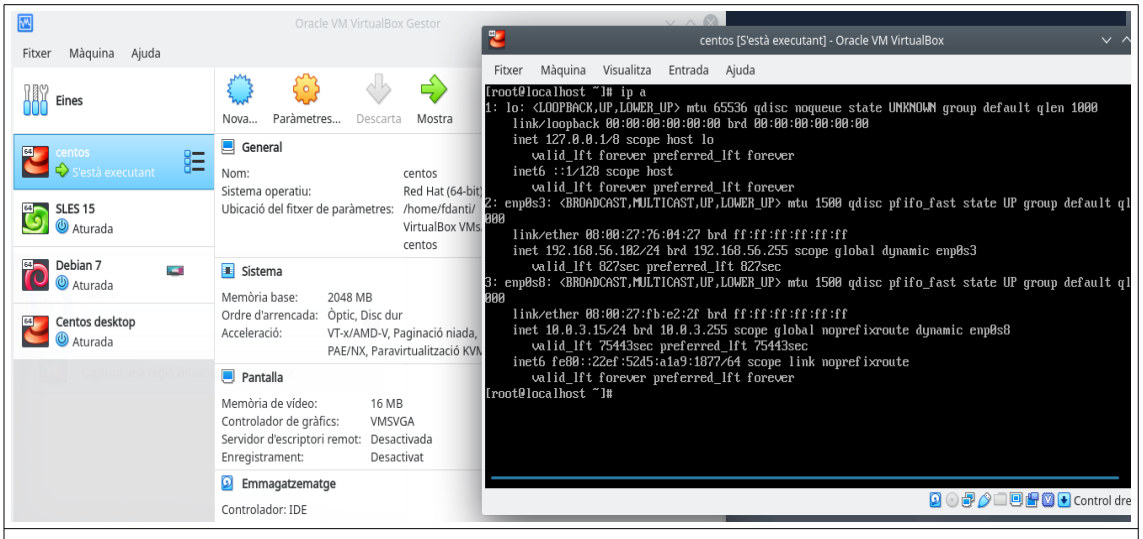

Figura III. Virtualbox amb diferents màquines virtuals instal·lades

La infraestructura que hem utilitzat per dur a terme aquesta memòria ha consistit en:

- Una màquina host, un portàtil personal, amb 8Gb de RAM i un Debian 9 de 64 bits.
- Una màquina virtual amb 2Gb de RAM i un CentOS7 de 64 bits sense entorn gràfic.
- Una màquina virtual amb 2Gb de RAM i un Debian 8 de 64 bits sense entorn gràfic.
- Una màquina virtual amb 4Gb de RAM i un SUSE SLES 15 de 64 bits sense entorn gràfic.

La motivació d'aquestes instal·lacions han sigut diferents factors:

- Hem decidit, per evitar haver de tenir sempre dues màquines virtuals engegades, que la màquina Host (el portàtil) seria la que faria les avaluacions remotes als altres sistemes. D'aquesta manera, d'una forma centralitzada podem avaluar totes les màquines virtuals d'una en una.
- Hem comprovat que Debian només inclou la paqueteria de SCAP en la versió 9 del seu sistema operatiu, però només es publiquen les SSG per a la versió 8. Aquest fet ha provocat que des de la màquina host no es pogués analitzar el mateix portàtil. Font[1] (secció «Preparing the targeted hosts»).
- Hem comprovat que la versió de «SCAP Workbench» de paqueteria de «OpenSUSE Leap» no està compilada per suportar la avaluació de sistemes remots. Aquest fet ha descartat la idea inicial de treballar amb un host de sobretaula més potent que tenia un OpenSUSE.
- Hem comprovat que el l'entorn gràfic del CentOS7 té problemes amb el ratolí i Virtualbox. No hem investigat més l'assumpte, perquè al cap i a la fi, el CentOS havia de ser un sistema remot amb el que hem treballat sempre per consola. Sigui com sigui, creiem necessari deixar-ne constància per a futurs treballs. Una primera idea d'usar el CentOS com a màquina per fer els avaluacions va ser descartada immediatament veient que cada 5 minuts es perdia el ratolí de la màquina virtual.

2.2) La connectivitat a xarxa de les les màquines virtuals

A nivell de xarxa, era imprescindible que la màquina host es pogués connectar amb les màquines virtuals directament.

A tal efecte, dins del Virtualbox, en l'apartat de «Gesto de xarxes amfitrió» hem creat una xarxa nova, amb nom vboxnet0 i amb adreçament privat 192.168.56.0/24, amb un servidor DHCP que servia adreces IP a partir de la 192.168.56.101.

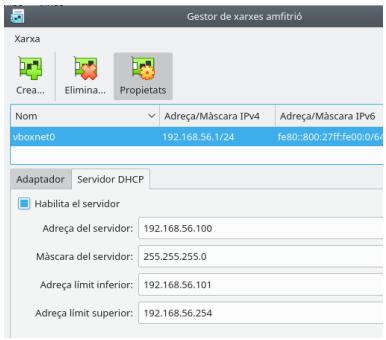

Figura IV. Virtualbox – Gestor de xarxes amfitrió

Per acabar, a les màquines virtuals hem ajustat la interfície de xarxa principal de forma que es connectés a la xarxa virtual que hem generat 'vboxnet0'. A fi i efecte que la màquina virtual pogués navegar per internet, en els moments en que havíem de descarregar algun programari, hem habilitat la interfície 2 de cada màquina virtual en mode NAT. D'aquesta manera, cada màquina virtual tenia dues IP: una de la xarxa 192.168.56.0/24 que ens donava accés des de la màquina host i una segona amb adreçament privat de la 10.0.0.0/8 que li donava accés a internet.

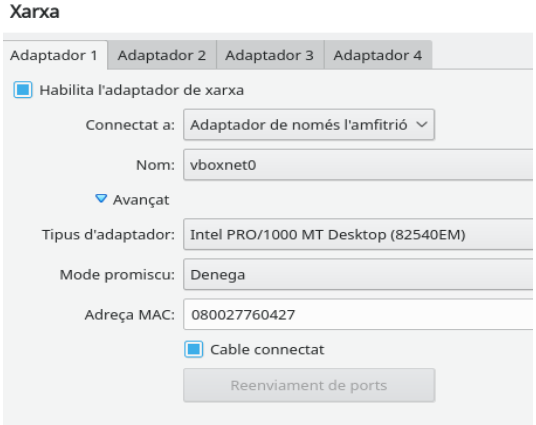

Figura V. Virtualbox – Gestor de xarxes amfitrió. Configuració de la interfície de xarxa principal de la VM

2.3) El programari a instal·lar

Un cop hem tingut totes les màquines instal·lades, hem procedit a la instal·lació del software a cadascuna d'elles.

A la màquina host, que és la que s'ocuparà de fer les avaluacions remotes, hi hem instal·lat el programari de OpenSCAP, «SCAP Workbench» i les «SCAP Security Guides» mitjançant la comanda:

```
sudo apt install libopenscap8 scap-workbench ssg-applications ssg-base \
                 ssg-devbderiver ssg-debian ssg-nondebian
```

Comanda per a instal·lar la infraestructura necessària en la màquina host

A més, per tal de poder fer avaluacions remotes des de la línia de comandes hem descarregat el script *openscap-ssh*, malgrat que hem vist que en la versió 1.2.3 de OpenSCAP ja s'hi inclou aquest programari (Font[2]).

Aquesta aplicació recopila tota la informació SCAP necessària per fer córrer la avaluació a la màquina remota, la envia a l'equip remot (en el nostre cas, la màquina virtual) via SSH i en recupera els resultats per tal de presentar l'informe.

A les màquines virtuals, per tal de dur a terme les avaluacions remotes ens caldrà instal·lar-hi el OpenSCAP, així com també el servidor de SSH per permetre la connexió remota. En cas que tinguin un firewall, caldrà obrir-lo per tal de permetre la connectivitat des de la màquina host.

3) Tecnologies usades

En aquest tercer capítol descriurem les tecnologies que hem usat en aquest Treball Final de Màster. Ho farem fins al detall en que haguem arribat durant aquests mesos de treball, assumint que no busquem ser exhaustius amb les definicions ni amb les opcions de les diferents tecnologies; en la major part dels subcapítols, hem arribat a veure una petita part de les opcions que ens ofereixen.

3.1) SCAP

SCAP[3] és el acrònim de «Security Content Automation Protocol» o «Protocol d'Automatització de Contingut de Seguretat». Creada per un grup del «*National Institude of Standards and Technology*» (NIST)[4] l'any 2006, és una especificació que permet expressar de forma estandarditzada, informació relacionada amb la seguretat informàtica: errors en el programari, configuracions problemàtiques i un llarg etcètera.

Hem d'entendre SCAP com una forma de definir la nomenclatura i format de tot un seguit de components relacionats amb la seguretat informàtica. La relació dels components amb que treballa SCAP són:

CVE, Common Vulnerabilities and Exposures. Defineix una enumeració comuna de les vulnerabilitats que apareixen en els sistemes informàtics. Defineix una nomenclatura estàndard per cadascuna de les vulnerabilitats que apareixen.

CCE, Common Configuration Enumeration. Defineix una enumeració comuna de vulnerabilitats de seguretat i també sobre erros en configuracions de sistemes.

CPE, Common Platform Enumeration. Defineix una enumeració comuna pel software. És crucial, a la hora d'identificar vulnerabilitats o erros de configuració, poder referenciar exactament quin es el software afectat.

XCCDF, eXtensible Configuration Checklist Description Format. Llenguatge, en format XML, amb la definició de checklists.

OVAL, Open Vulnerability and Assessment Language. Llenguatge per a definir informació sobre la configuració i avaluació de l'estat dels sistemes.

CVSS, Common Vulnerability Scoring System. Valoració numèrica d'una vulnerabilitat de seguretat.

OCIL, Open Checklist Interactive Language. Llenguatge per a implementar controls sobre dades emmagatzemades.

AI, Asset Identification. Definició del format per a la identificació i informació sobre els diferents actius existents.

ARF, Asset Reporting Format. Definició del format per a expressar informació sobre actius i l'anàlisi d'aquests.

CCSS, Common Configuration Scoring System. Valoració numèrica d'una configuració que afecta a nivell de seguretat. Similar a CVSS però aplicat a configuració.

TMSAD, Trust Model for Security Automation. Especificació per a implementar un model de confiança en SCAP sobre la integritat dels resultats generats.

Comprovem que es traca d'especificacions, sempre relacionades amb la seguretat informàtica i que, exceptuant el TMSAD, es poden classificar en enumeracions, llenguatges o valoracions numèriques.

El protocol SCAP ha tingut una gran adopció en el sector de la seguretat informàtica, motiu pel qual trobem contingut en aquest format per part de la majoria dels fabricants de software (sistemes operatius i també aplicacions) així com dels diferents CERT o equips de resposta a incidents informàtics. Vindria a ser l'estàndard *de facto* en el llenguatge de la seguretat informàtica.

3.2) SCAP Checklists

Hem vist com els formats definits per SCAP estandarditzen tot allò relacionat amb la seguretat informàtica. Això permet definir les polítiques i normatives de seguretat en un format comú, que es podran exportar entre diferents sistemes. Aquest mecanisme ens permetrà disposar o generar una guia de mesures de seguretat o «*checklists*» que convé o cal tenir implementades en els sistemes; ja siguin en forma de vulnerabilitats existents o d'ajustos en les configuracions.

Un checklist o guia de seguretat és una política escrita complint els requisits de SCAP, en format XML en el que bàsicament es treballa amb fitxers XCCDF i OVAL.
Els fitxers **XCCDF** són merament descriptius, sense comandes a executar directament però amb referència altres components SCAP.
Els fitxers **OVAL** contenen informació específica del sistema, de la seva configuració i del seu estat.
Entrarem en el detall del format d'aquests fitxers en el punt 2.4 i 2.5.

Les regles que s'especifiquen en una guia de seguretat provenen de guies de seguretat publicades per ens reconeguts en el sector de la seguretat informàtica com PCI DSS (estàndard de seguretat per a la indústria de targetes de pagament), STIG (Security Technical Implementation Guide) o United States Government Configuration Baseline (USBCB).

Aquest seguit de regles s'agrupen i ajusten en el que s'anomenen «profiles» o «*baselines*», que defineixen uns requisits diferents segons el nivell de seguretat que es vol definir.
Així, per exemple, la política pot definir que el password dels usuaris s'ha de canviar periòdicament. Un perfil molt laxe ens permetrà que aquest canvi sigui cada certs mesos; una perfil molt estricte ens dirà que aquest password s'ha de canviar cada pocs dies.

Aquests perfils o «*baselines*» poden ser definits pels mateixos fabricants de software, per ens externs o governamentals (DISA, NIST) i personalitzats en cada cas. Aquestes llistes permetran verificar el compliment de la normativa de seguretat que cada empresa tingui assignada. Així, per exemple, els entorns crítics com el militar o la banca hi haurà definides unes polítiques més estrictes que la que es podria tenir en una petita empresa familiar.
Per posar un exemple, la **Defense Information System Agency** (DISA), d'EUA, publica una política SCAP pels sistemes de les empreses que han de treballar amb ells. Amb aquesta política qualsevol pot auditar si els seus sistemes compleixen la normativa que la DISA exigeix.

La majoria de fabricants de software ofereixen polítiques de seguretat i existeixen, també, algunes guies oficials que es poden descarregar de la xarxa. Un responsable de seguretat, en principi no haurà de definir la política i normativa de seguretat de la empresa des de zero; hauria de ser suficient ajustant alguna política existent per adequar-la a les seves necessitats.

Tot seguit explicarem els tres mecanismes per a obtenir checklists que hem usat en aquest TFM. No pretén ser una llista exhaustiva de formes d'obtenir contingut SCAP; simplement són tres exemples de un ens privat, un de governamental i un fabricant que publiquen aquest tipus de material.

3.2.1) National Checklist Program Repository

El NIST és un ens governamental que s'ocupa, entre moltes d'altres tasques, de publicar, dins del seu projecte «*National Vulnerability Database*» (NVD)[4] i de forma oberta un repositori amb tot de checklist de diferents autoritats. Entre altres hi podem trobar fabricants de software com Microsoft, RedHat; fabricants de maquinari com HP o Kyocera i també autoritats governamentals dels EUA com la NSA o el mateix NIST.

El cercador ens permetrà descarregar el checklist en format xml per tal de poder aplicar-lo als nostres sistemes.
No és exhaustiu, però és un bon lloc per on començar a buscar.

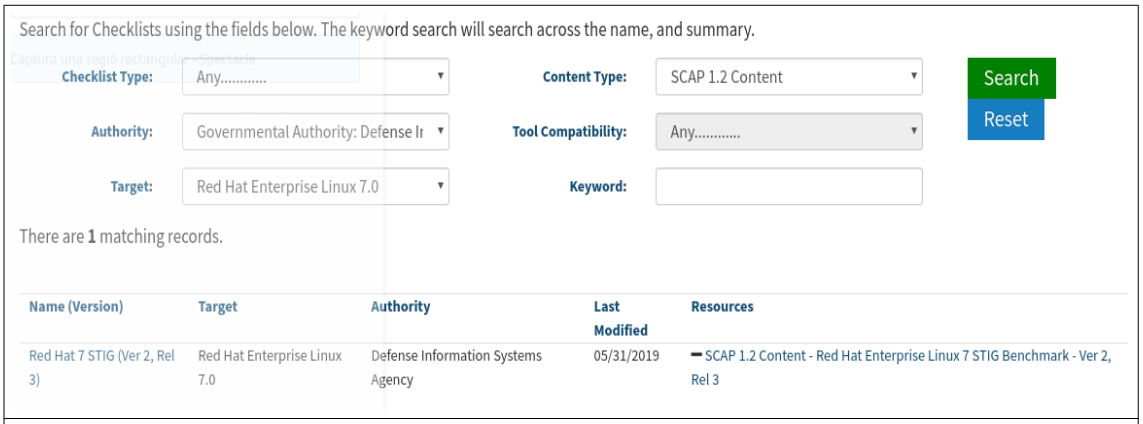

Figura VI. Cercador de checklists del NIST

3.2.2) SCAP Security Guide

La comunitat de OpenSCAP, dins del seu ventall de productes, ofereix de forma oberta una col·lecció de checklists de diferents autoritats com ara PCI-DSS, NIST SP-800-53 o ANSSI.
En el web de OpenSCAP[5] podem descarregar-les i usar-les en els nostres sistemes. En els sistemes Debian, la instal·lació es fa mitjançant quatre paquets del sistema:

Paquet	Contingut
ssg-debian	Polítiques de sistemes Debian
ssg-debderived	Polítiques per sistemes Ubuntu 14-16
ssg-nondebian	Polítiques per altres OS (RedHat, Fedora, SUSE, etc)
ssg-applications	Polítiques sobre aplicacions (Firefox, Java, etc)

Cal tenir en compte que les polítiques definides per OpenSCAP contenen alguns components que no formen part de la definició de SCAP i això pot provocar problemes a la hora d'usar-les amb altres aplicacions. Entrarem en més detall d'aquest assumpte en el punt 2.3 d'aquest capítol.

3.2.3) SUSE Linux Enterprise OVAL Information DB.

Finalment, parlarem de SUSE. Aquesta empresa, com la majoria dels fabricants de sistemes operatius (Microsoft, Debian, RedHat, etcètera), publiquen de forma regular tot de material SCAP per tal de poder fer validacions sobre els seus sistemes.
És una base de dades sobre incidents de seguretat, indexats per producte i un conjunt de RPM a usar per tal de complir amb els compromisos de seguretat del fabricant.
Dins del web de support de SUSE[6] podem descarregar el contingut OVAL pels diferents sistemes operatius de la empresa.

3.3) OpenSCAP

OpenSCAP, empresa que depèn de RedHat, ofereix un conjunt d'eines destinades a la seguretat informàtica.

El fet que el NIST els certifiqués[7] en la tecnologia SCAP les ha permès créixer molt en aquest sector i ara per ara, les eines de OpenSCAP són àmpliament usades en els sistemes basats en *nix. Això no vol pas dir que siguin els únics. Microsoft Corporation disposa de la seva pròpia eina nativa per treballar amb SCAP, basada en el Configuration Manager[8]. Empreses com ThreadGuard i Fortinet, també disposen d'eines de escaneig i validació basats en SCAP.

Ja hem introduït, el els capítols anteriors, el concepte de «*scanner*»: una eina que ens permetrà, entre d'altres coses, llegir polítiques de seguretat basades en SCAP i validar si un sistema la compleix o no.
El scanner de OpenSCAP 'oscap' permet fer moltes accions amb llenguatges de SCAP: treballar amb OVAL, XCCDF, CVVS, CPE, CVE i amb CVRF. En aquest treball ens centrarem amb les opcions que ens dóna amb OVAL i XCCDF, ja que aquests són els que formen part de l'objectiu.

En aquest apartat explicarem com instal·lar i com usar les eines que ens ofereix OpenSCAP per a dur a terme aquesta tasca: oscap i SCAP Workbench. Veurem que es tracta de dues formes d'atacar el mateix binari: per línia de comandes o a través d'una interfície més amigable.

3.3.1) Línia de comandes: oscap

La eina oscap funciona per línia de comandes i convé dominar-la si el que volem acabar fent és validacions periòdiques que s'executen autònomament. La línia de comandes facilita molt la creació de scripts que permeten automatitzar totalment la execució de comandes, ja sigui amb un servei degudament ajustat o via tasques programades.

La instal·lació d'aquest programa està suportada en la majoria dels sistemes operatius, ja siguin Linux com Windows. A la seva web de descàrregues[9] podem comprovar que hi ha paqueteria inclosa en els repositoris de la majoria dels fabricants de Linux (Debian, Fedora, RedHat). En el cas de SUSE, els repositoris del sistema també inclouen aquest programari, malgrat que no aparegui al web de OpenSCAP. En el cas de Debian, la instal·lació es fa mitjançant el paquet 'libopenscap8' Podem veure els passos seguits per fer la instal·lació del programari en el Annex I d'aquest treball.

Un cop instal·lada la eina, ens haurem d'assegurar de disposar de les guies de seguretat necessàries per a poder fer els anàlisis del sistema. En el nostre cas ens hem basat en la paqueteria de Debian per instal·lar la SCAP Security Guide de OpenSCAP. Així, mitjançant quatre paquets bàsics ens ofereixen les polítiques de seguretat necessàries pel nostre entorn de treball en la carpeta */usr/share/xml/scap/ssg/content.*

Un cop tenim la eina (scanner) i el contingut (SCAP Security Guides) instal·lats, comprovarem que a la carpeta de destí ens hi han aparegut tot de fitxers OVAL, OCIL, XCCDF i també un format que no havíem vist fins ara: DS (datastream). Els datastream són fitxers específics de OpenSCAP en el que el s'inclouen els continguts dels altres fitxers, amb la intenció de permetre una

més fàcil distribució dels fitxers. En el nostre cas, treballarem sempre que puguem sobre els fitxers natius. El motiu és senzill: és més fàcil navegar per fitxers més petits que per un de molt gran.

Avaluació de polítiques XCCDF amb 'oscap xccdf eval'.

Abans de poder executar un anàlisi sobre un sistema, ens caldrà decidir quin perfil de XCCDF volem executar. Entrarem en més detalls sobre els perfils en l'apartat 2.4.1, de forma que aquí ens limitarem a dir que una mateixa política de seguretat pot tenir definits diferents perfils, més o menys restrictius segons quina sigui la normativa a complir; ja que una normativa més estricta tindrà uns valors més restrictius en alguns aspectes. En el capítol 3 ens centrarem en la personalització d'aquestes polítiques.

Per comprovar quins perfils tenim disponibles en una política, podem usar la comanda '*oscap info*' seguida de la política que volem estudiar:

Figura VII. Execució de la comanda 'oscap info' sobre la política ssg-rhel7-xccdf.xml

Podem comprovar com la comanda info ens retorna, donada una política en format XCCDF, una relació dels perfils que hi ha disponibles.

Assumint que volem usar el primer dels perfils que ens han aparegut: «DISA STIG for Red Hat Enterprise Linux 7», podrem avaluar la política en la màquina mitjançant la comanda:

```
oscap xccdf eval --profile stig-rhel7-disa  --report /tmp/report.html ssg-rhel7-xccdf.xml
```

Exemple d'avaluació d'un XCCDF mitjançant oscap

Aquesta comanda avaluarà el fitxer XCCDF ssg-rhel7-xccdf.xml, amb el perfil stig-rhel7-disa i desarà els resultats en un format html en el fitxer /tmp/report.html

Podem avaluar també una màquina remota, passant els paràmetres adequats a la comanda 'oscap-ssh', que haurem de descarregar de la xarxa directament. En el següent exemple, estaríem avaluant la mateixa política que abans en un equip remot amb adreça IP 192.168.56.102:

```
oscap-ssh --sudo root@192.168.56.102 22 xccdf eval --profile stig-rhel7-disa  \
                                                    --report /tmp/report.html ssg-rhel7-xccdf.xml
```

Exemple d'avaluació remota d'un de XCCDF mitjançant oscap-ssh

OpenSCAP i SCE

Cal mencionar que OpenSCAP incorpora Script Check Engine (SCE), un llenguatge que no forma part de l'estàndard SCAP. Aquest fet porta a que el codi OpenSCAP que incorpora aquest llenguatge, no sigui compatible amb els altres fabricants.

Cal dir que apartar-se dels estàndards és -a priori- una mala idea, però cal matisar els motius que han portat a OpenSCAP a incorporar aquest llenguatge, ja que no ha estat perquè si.

Tal i com podem llegir al web de OpenSCAP sobre SCE[10], aquest llenguatge permet incorporar, dins de fitxers XCCDF, definicions que fan referència a scripts externs. Aquests scripts externs es poden usar a tal i com s'usen els scripts estàndard, executant-los i obtenint el resultat per tal de definir si un check es valida o no.

Aquest treball és la nostra primera incursió en aquest sector i no tenim prou coneixements per defensar un costat o un altre de la balança. Ara bé, cal remarcar que l'objectiu de OpenSCAP en permetre aquests scripts externs és facilitar la implementació de les tecnologies SCAP en sistemes on, sense el SCE seria molt costós d'iniciar processos d'integració; entenem que el SCE permetria començar a ordenar el parc tecnològic.

3.3.2) SCAP Workbench (GUI)

En l'apartat anterior hem vist com avaluar polítiques de seguretat mitjançant la línia de comandes. En aquest apartat farem la avaluació mitjanant una eina amb entorn gràfic (GUI).

SCAP-Workbench és una eina creada per l'equip de OpenSCAP, amb la intenció de facilitar la avaluació de polítiques amb un entorn més amigable que la consola. S'instal·la independentment del oscap i requereix d'un entorn gràfic per a poder funcionar.

Invocat mitjançant la comanda 'scap-workbench' ens obrirà una pantalla en la que directament ens oferirà triar quin contingut (quina política) volem carregar. Aquí tenim una diferència important respecte la línia de comandes: aquesta pantalla ens oferirà carregar directament els DataStream, no els XCCDF o OVAL directament. Recordem que un DataStream és un format de fitxer, reconegut per SCAP, en el que s'incorporen tots els fitxers necessaris (OVAL, XCCDF, etcètera) amb la intenció de facilitar la exportació de continguts per internet.

En el moment de fer aquest treball, SCAP Workbench no sap treballar directament amb fitxers XCCDF i anàlisis remots. El resultat d'intentar-ho és un missatge d'error que ens convida a usar el DataStream:

```
18:17:53  except  There was a problem setting up the scanner. There was a problem with
                  OscapScannerRemoteSsh! You can only use source datastreams for scanning remotely!
                  Remote scanning using plain XCCDF and OVAL files has not been implemented in SCAP
                  Workbench yet.
```

Figura VIII. Missatge d'error al intentar avaluar XCCDF amb SCAP Workbench directament contra un sistema remot

Així doncs, triarem el DataStream de CentOS

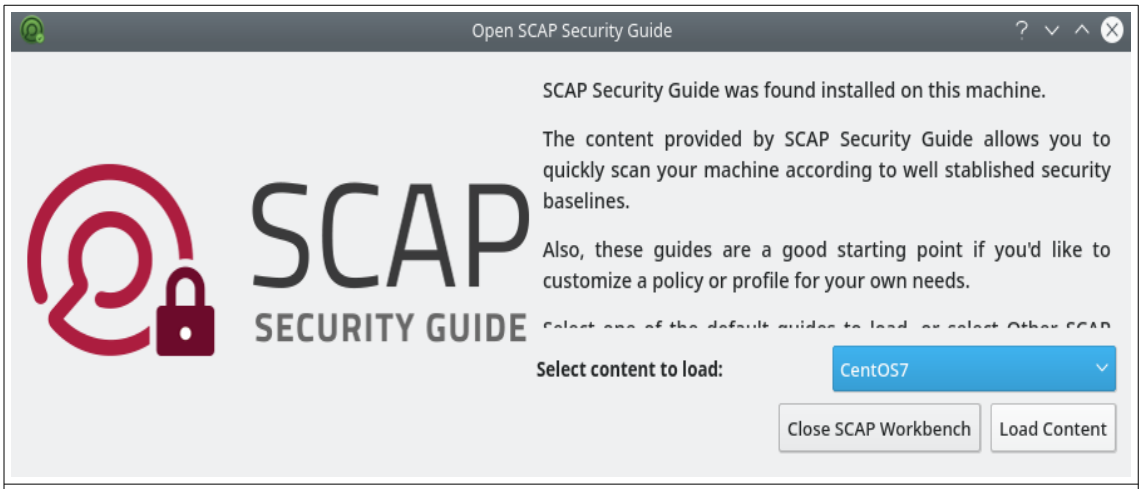

Figura IX. SCAP-workbench, de OpenSCAP – Seleccionar «CentOS» per poder obrir el DataStream

El DataStream de CentOS7 inclou dos XCCDF i haurem de triar quin volem. Al tractar-se d'una avaluació de proves, ens decantarem per la opció de scap_cref_ssg-rhel7-xccdf-1.2.xml.

Pel que fa al Profile, al tractar-se d'una avaluació de proves, ens decantarem pel perfil estàndard de «Red Hat Enterprise Linux 7», que inclou 51 tests.

Farem un anàlisi remot de la mateixa màquina que hem fet en l'apartat anterior (192.168.56.102) i obtindrem una pantalla com la següent:

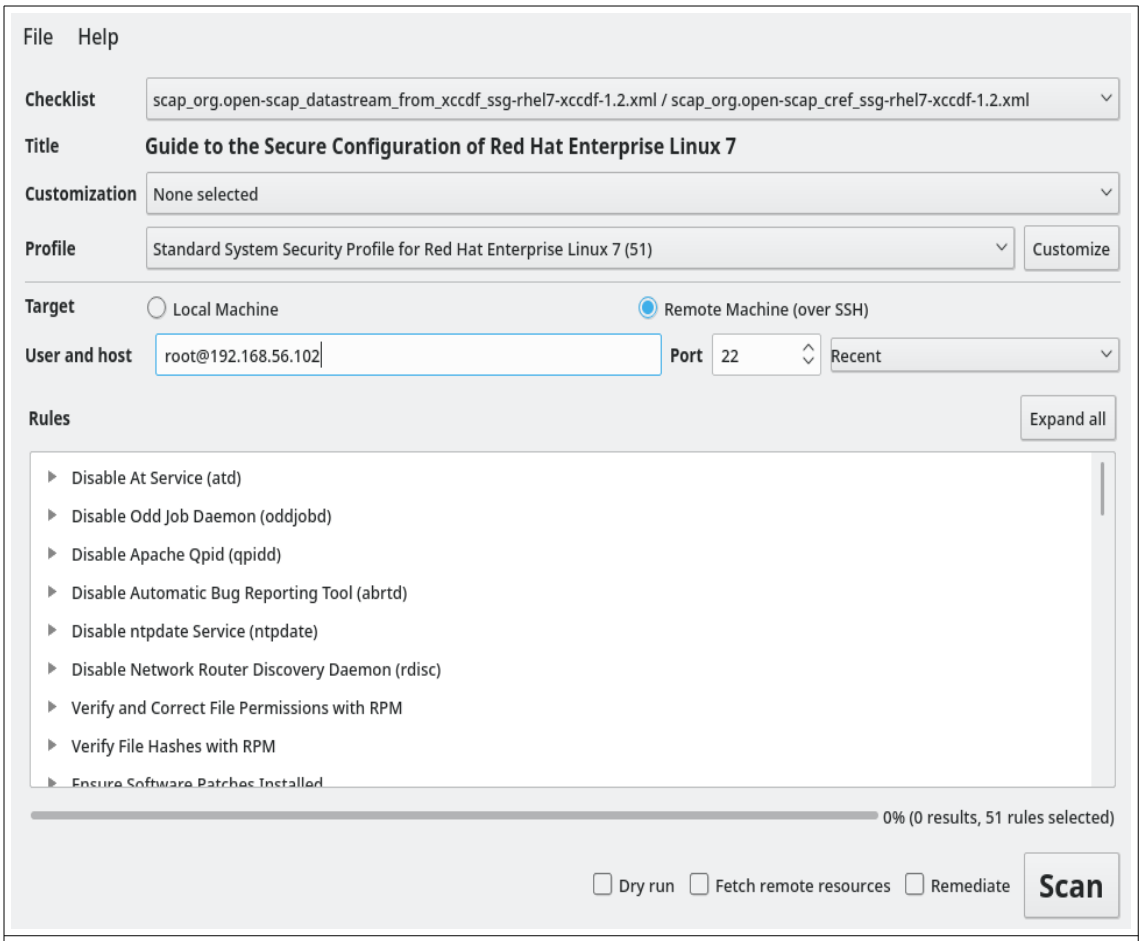

Figura X. Pantalla principal de SCAP-Workbench

Hem deixat desmarcades algunes opcions que volem comentar:

- **Dry run.** Amb aquesta opció marcada, la eina ens diria què farà però sense fer-ho. És molt útil perquè ens generarà les línies de comandes necessàries per a executar la avaluació. D'aquesta manera, d'una forma amigable podem generar el codi per crear scripts complexos que es poden automatitzar.
- **Fetch remote resources.** Aquesta opció permetria al scanner descarregar contingut OVAL remot, si així ho sol·licita la checklist. El checklist triat necessita descarregar contingut extern, però en aquesta avaluació de proves no permetrem que es descarregui. Això generarà un warning en la execució, però no la pararà..
- **Remediate.** Si marquem aquesta opció, el scanner intentarà solucionar els tests que hagin sortit 'failed'.

Amb les tres opcions desmarcades i llançarem el escaneig tot clicant a «Scan». Obrirem el report que es genera amb l'anàlisi i podrem observar els resultats:

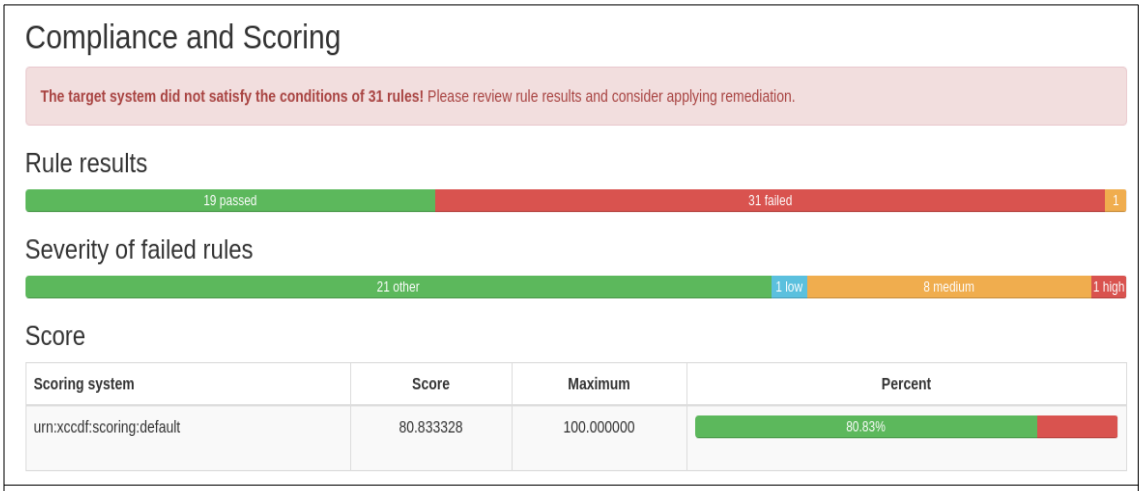

Figura XI. Resultat de la avaluació sobre el sistema remot CentOS amb el perfil de "Standard Red Hat Enterprise Linux 7»

Observem com de 51 tests del checklist, 19 han passat satisfactòriament, 31 han fallat (no superats) i un no s'ha pogut executar. El check que no s'ha pogut executar és perquè no s'ha pogut descarregar els continguts remots.

A títol de prova, tornem a fer la avaluació i aquest cop sol·licitem que se solucionin els checks que fallin (remediation).

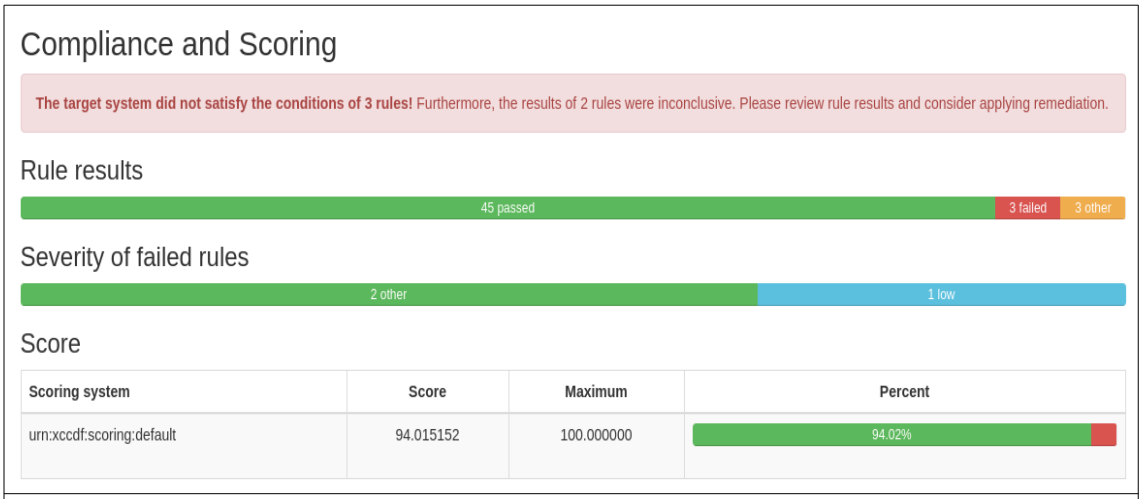

Figura XII. Resultats després d'aplicar les remediation automàtiques

Observem com després de la remediation, els checks superats passen a ser 45 de 51.

Al investigar amb els checks que han fallat, ens adonem que son assumptes difícilment solucionables mitjançant un script, ja que podrien afectar al sistema de forma molt negativa. Per posar alguns exemples que, a l'apartat de conclusions ens ajudaran una mica:

- Particionar. Algunes regles estableixen condicions sobre quins punts de muntatge han d'estar en una partició separada. Establir un remediation per aquests casos no és trivial, ja que particionar implica aturar serveis,

possiblement reiniciar la màquina i el sistema d'avaluació es limitarà a indicar-nos quines accions s'haurien de fer; no les farà.

- Deshabilitar el SUID en tots els fitxers que no hagin sigut instal·lats directament des d'un RPM confiable. El check del OVAL ens indicarà que hi ha algun fitxer amb SETUID i que no és d'un origen confiable. En aquest cas, no en canviarà el valor, perquè OVAL treballa amb estats i no es pot canviar l'estat d'un fitxer desconegut; així doncs, la avaluació ens informarà que hi ha un fitxer que no compleix els requisits, però no podrà solucionar-ho automàticament.

En el capítol 4, de personalització de les polítiques, veurem com usar el SCAP Workbench per a adaptar les polítiques a les necessitats de la nostra empresa.

3.4) OVAL

El Open Vulnerability Assessment Language (OVAL) és un llenguatge en XML que serveix per a definir comprovacions sobre vulnerabilitats i problemes de configuració, tot definint tests sobre estats del sistema. Un document OVAL està format per diferents seccions, cadascuna relacionada amb les altres mitjançant identificadors i que ens permeten reaprofitar parts del codi.
La part principal d'un document OVAL són les definicions de la vulnerabilitat, configuració, patch o inventari que durà a terme.
Aquestes definicions consten de dues parts; primerament hi ha una secció amb metadades amb la informació referent a la descripció: títol, sistemes afectats, referències externes, qui ha publicat la definició, etcètera. Tot seguit es defineixen els criteris aplicar sobre el sistema per comprovar si el sistema està afectat o no.

En el següent codi, veiem una definició en format OVAL pel CVE-2017-16612, que afecta als sistemes operatius SUSE Linux Enterprise de la versió 15 que tenen una libXcursor instal·lat. Aquest OVAL s'ha descarregat del web de SUSE.
Podem comprovar com el codi OVAL, en format XML, primerament defineix la classe de la definició; en aquest cas és de tipus «*vulnerability*» (marcat en color verd). Es fa una petita descripció del problema, especificant a quins sistemes operatius afecta, el títol i les referències corresponents al CVE, defineix uns criteris per a determinar si la màquina és o no vulnerable.
Aquests criteris s'ajuden de múltiples tests OVAL (en color blau) a executar i d'operadors booleans per a crear una lògica sobre aquests tests:

```
<definition id="oval:org.opensuse.security:def:201716612" version="1" class="vulnerability">
 <metadata>
 <title>CVE-2017-16612</title>
   <affected family="unix">
        <platform>SUSE Linux Enterprise Module for Basesystem 15</platform>
        <platform>SUSE Linux Enterprise Module for Basesystem 15 SP1</platform>
        <platform>SUSE Linux Enterprise Module for Desktop Applications 15</platform>
        <platform>SUSE Linux Enterprise Module for Desktop Applications 15 SP1</platform>
   </affected>
      <reference ref_id="CVE-2017-16612" ref_url="http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-16612"
source="CVE"/>
   <description>
      libXcursor before 1.1.15 has various integer overflows that could lead to heap buffer overflows when processing
```

```
malicious
      cursors, e.g., with programs    like GIMP. It is also possible that an attack vector exists against the related code in
cursor/xcursor.c in Wayland.
    </description>
    <advisory from="security@suse.de">
     <severity>Important</severity>
     <cve href="https://www.suse.com/security/cve/CVE-2017-16612.html">CVE-2017-16612</cve>
     <bugzilla href="https://bugzilla.suse.com/1065386">SUSE bug 1065386</bugzilla>
    </advisory>
  </metadata>
  <criteria operator="OR"><criteria operator="AND"><criteria operator="OR">
            <criterion  test_ref="oval:org.opensuse.security:tst:2009224670"  comment="SLES  Module  for  Desktop
Applications 15 is installed"/>
            <criterion  test_ref="oval:org.opensuse.security:tst:2009255099"  comment="SLES  Module  for  Desktop
Applications 15 SP1 is installed"/>
    </criteria>
          <criterion  test_ref="oval:org.opensuse.security:tst:2009224809"  comment="libXcursor1-32bit-1.1.15-1  is
installed"/>
   </criteria>
   <criteria operator="AND"><criteria operator="OR">
      <criterion test_ref="oval:org.opensuse.security:tst:2009223735" comment="SLES Module for Basesystem 15 is
installed"/>
       <criterion test_ref="oval:org.opensuse.security:tst:2009254629" comment="SLES Module for Basesystem 15
SP1 is installed"/>
    </criteria>
    <criteria operator="OR">
            <criterion  test_ref="oval:org.opensuse.security:tst:2009224088"  comment="libXcursor-devel-1.1.15-1  is
installed"/>
     <criterion test_ref="oval:org.opensuse.security:tst:2009224089" comment="libXcursor1-1.1.15-1 is installed"/>
    </criteria> </criteria> </criteria>
</definition>
```

Fragment d'una entrada OVAL per a SuSE Linux Enterprise Server 15.
Descarregat del web de suport de SUSE[6]

En aquest exemple, ressaltem en color blau, com els criteris acaben fent referència a uns tests a executar en el sistema. Aquests tests estan especificats en el mateix document OVAL en la secció de «tests» i tenen aquest format:

```
<rpminfo_test id="oval:org.opensuse.security:tst:2009224670" version="1" comment="sle-module-desktop-
applications-release is ==15" check="at least one" xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#linux">
    <object object_ref="oval:org.opensuse.security:obj:2009042723"/>
    <state state_ref="oval:org.opensuse.security:ste:2009061809"/>
</rpminfo_test>
```

Exemple de test en un fitxer OVAL

De la mateixa forma, trobem referències a un objecte (en vermell) obj:2009042723 i a un estat (en verd) ste:2009061809.

```
<rpminfo_object id="oval:org.opensuse.security:obj:2009042723" version="1"
xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5#linux">
            <name>sle-module-desktop-applications-release</name>
</rpminfo_object>
```

Exemple d'objecte rpminfo_object en fitxer OVAL

```
<rpminfo_state id="oval:org.opensuse.security:ste:2009061809" version="1" xmlns="http://oval.mitre.org/XMLSchema/
oval-definitions-5#linux">
   <version operation="equals">15</version>
</rpminfo_state>
```

Exemple d'un estat rpminfo_state en fitxer OVAL

Observem doncs com un fitxer OVAL consta d'unes definicions en les que, mitjançant referències a objectes i estats, es pot comprovar si un equip està afectat per una determinada vulnerabilitat o error de configuració.

Hem vist doncs que OVAL és un llenguatge declaratiu. No s'indica què s'ha de fer si no què es vol que es faci. Hi haurà eines, anomenades *scanner* que s'ocuparan de fer aquestes tasques, en funció de quin sigui el sistema operatiu en el que s'estiguin executant.
Això ens garanteix que el mateix fitxer OVAL servirà per a totes les infraestructures, independentment de quina tecnologia hi hagi al darrera: simplement ens caldrà un *scanner* degudament instal·lat i un fitxer OVAL amb el contingut desitjat.

3.5) XCCDF

El llenguatge XCCDF defineix validacions sobre els ajustos dels sistemes. Aquest llenguatge permet passar múltiples validacions de forma automatitzada, facilitant les auditories i anàlisis de sistemes de forma massiva i periòdica. És un llenguatge descriptiu, en el que s'hi explicitarà què es vol aconseguir i no el com, fet que permet poder utilitzar el mateix fitxer XCCDF en diferents plataformes.

Podem entendre un fitxer XCCDF com una relació de regles a complir, «*benckmark*» o «*checklist*» que al executar-la, ens dirà quines regles es compleixen favorablement i quines no.

Un document XCCDF té una estructura com la del següent exemple. Observarem que i ha gran part del codi plegat de forma que només es veu la estructura bàsica:

```
<?xml version="1.0" encoding="UTF-8"?>
<Benchmark xmlns="http://checklists.nist.gov/xccdf/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="DEBIAN-8" resolved="1" xml:lang="en-US" style="SCAP_1.1">
  <status date="2018-07-26">draft</status>
  <title xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">Guide to the Secure Configuration of Debian 8</title>
  <description xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">This guide presents a catalog of security-relevant
</description>
  <notice xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US" id="terms_of_use">Do not attempt to implement any of the settings in
</notice>
  <front-matter xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">The SCAP Security Guide Project<html:br xmlns:html="http://www.w3.org/1999/xhtml"/>
</front-matter>
  <rear-matter xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">Red Hat and Red Hat Enterprise Linux are either registered
</rear-matter>
  <platform idref="cpe:/o:debian:project:debian:8"/>
  <version update="https://github.com/OpenSCAP/scap-security-guide/releases/latest">0.1.39</version>
  <metadata xmlns:xhtml="http://www.w3.org/1999/xhtml">
  </metadata>
  <model system="urn:xccdf:scoring:default"/>
  <Profile id="standard">
  </Profile>
  <Profile id="anssi_np_nt28_restrictive">
  </Profile>
  <Profile id="anssi_np_nt28_minimal">
  </Profile>
  <Profile id="anssi_np_nt28_high">
  </Profile>
  <Profile id="anssi_np_nt28_average">
  </Profile>
  <Group id="remediation_functions">
  </Group>
  <Group id="services">
  </Group>
  <Group id="system">
  </Group>
</Benchmark>
```

Fragment del fitxer XCCDF de ssg-debian8. Inclòs dins la paqueteria de ssg de Debian

En aquest subapartat veurem el funcionament de els execucions de codi XCCDF que podríem resumir de la següent manera:

- Comprovar, via OCIL o OVAL, si es compleix el requisit de la regla: el permís del fitxer és l'esperat?
- Si així s'ha sol·licitat des de la aplicació *scanner*, executar el fix ja sigui via Ansible o via Shell Script.

Per tal d'entendre com funciona un XCCDF comentarem alguns dels elements bàsics mitjançant un exemple. L'exemple s'ha extret de la SCAP Security Guide de OpenSCAP, instal·lada en un sistema Debian mitjançant els paquets ssg-* de la distribució i, un cop instal·lat, es pot trobar a */usr/share/xml/scap/ssg/content/ssg-debian8-xccdf.xml*

3.5.1) Profiles

Els perfils són el mecanisme que té XCCDF per a reaprofitar codi. Es defineixen les regles a complir en general i, mitjançant la definició de perfils, es determinarà quins elements es comprovaran/validaran i amb quins valors.

```
<Profile id="standard">
      <title xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US" override="true">Standard System Security Profile for Debian 8</title>
     <description xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US" override="true">This profile contains rules to ensure standard security baseline of a Debian 8 system. Regardless of your system's workload all of these checks should pass.</description>
    <select idref="partition_for_tmp" selected="true"/>
    <select idref="partition_for_var" selected="true"/>
    <select idref="partition_for_var_log" selected="true"/>
    <select idref="partition_for_var_log_audit" selected="true"/>
    <select idref="partition_for_home" selected="true"/>
    <select idref="package_auditd_installed" selected="true"/>
    [...]
    <select idref="hw-install" selected="false"/>
    <refine-value idref="sshd_idle_timeout_value" selector="5_minutes"/>
  </Profile>
```

Fragment del Profile "standard". Font: ssg-debian8-xccdf.xml

D'aquest codi comentarem quatre elements que cal destacar:

- **<Profile>** Mitjançant l'element Profile, definim un nou perfil amb identificador «*standard*». Aquest perfil servirà per a customitzar les regles a les necessitats concretes, tot definint diferents elements <select> i <refine-value>.
- **<select>** Element que permet definir si un check es durà a terme o no dins del Profile que s'està definint. En la part visible de l'exemple podem comprovar com el perfil «*standard*» inclou múltiples regles a passar i exclou la «*hw-install*».
- **<refine-value>** Element que determina el valor d'un camp en el Profile que s'està definint, sobre-escrivint el valor per defecte. En el cas de l'exemple, amb color blau, el valor redefinit passa a ser el valor representat per un selector amb identificador «*5_minutes*», vinculat al camps «*sshd_idle_timeout_value*». Aquest selector el trobem definit en el mateix fitxer XCCDF, en l'apartat de serveis i ssh.

3.5.2) Agrupacions de regles - Groups

Pel que fa a la definició de Regles, el fitxers XCCDF mantenen una estructura jeràrquica en base als G**roups**.

Aquestes agrupacions permeten disposar d'una estructura comuna a tots els fitxers, facilitant la localització de regles. Així per exemple, en el mateix fitxer XCCDF anterior, trobem les següents agrupacions de regles:

```
<Group id="system">
   <title xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">System Settings</title>
      <description xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">Contains rules that check correct system settings.</description>
   <Group id="software">
    <title xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">Installing and Maintaining Software</title>
          <description xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">The following sections (...)
</description>
   </Group>
   <Group id="logging">
    <title xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">Configure Syslog</title>
      <description xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">The syslog service has been (...)
</description>
    <Group id="rsyslog_sending_messages">
     <title xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">Rsyslog Logs Sent To Remote Host</title>
      <description xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">If system logs are to be useful (...)
</description>
     <Rule id="rsyslog_remote_loghost" selected="false" severity="unknown">
      <title xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">Ensure Logs Sent To Remote Host</title>
     (...)
```

Fragment de codi XCCDF amb la jerarquia de grups per definir Rules. . Font: ssg-debian8-xccdf.xml

Aquesta jerarquia ens porta a situacions en que tenim grups definits que estan vuits, com és el cas del grup de «software» (en vermell a l'exemple). Ara bé, mantenir una estructura jeràrquica comuna facilita la automatització dels processos, a part de agilitzar els canvis que s'hagin d'aplicar en els fitxers, ja que gran part de la feina de definició ja està feta.

En color verd podem observar com es defineixen, jeràrquicament, dos subgrups «logging» i «rsyslog_sendind_messages», als que finalment, en color porpre, s'especifica una regla mitjançant el camp Rule.

3.5.3) Rules, fix i check

Per il·lustrar la definició de els regles, usarem una del grup <System>,<Permissions>,<Permissions_important_account_files> que ens permetrà veure com es defineix una regla que afecta als permisos del fitxer */etc/gshadow* d'un sistema *nix.

Com en el cas de l'exemple anterior, aquest fitxer es troba en el paquet sgg de Debian.

```
<Rule id="file_permissions_etc_gshadow" selected="false" severity="medium">
  <title xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US">Verify Permissions and ownership on gshadow
File</title>
  <description xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US"> ... </description>
  <reference href="http://www.ssi.gouv.fr/administration/bonnes-pratiques/">NT28(R36)</reference>
  <reference href="http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf">AC-6</reference>
  <rationale xmlns:xhtml="http://www.w3.org/1999/xhtml" xml:lang="en-US"> (…) Protection of this file is critical for
system security.</rationale>
  <fix xmlns:xhtml="http://www.w3.org/1999/xhtml" id="file_permissions_etc_gshadow"
system="urn:xccdf:fix:script:sh"
                                                                    complexity="low"
disruption="low" strategy="configure">
     chmod 0640 /etc/gshadow
  </fix>
  <fix xmlns:xhtml="http://www.w3.org/1999/xhtml" id="file_permissions_etc_gshadow"
system="urn:xccdf:fix:script:ansible"
                                                                    complexity="low"
disruption="low" strategy="configure">
    - name: Ensure permission 0640 on /etc/gshadow
      file:
        path="{{item}}"
        mode=0640
      with_items:
        - /etc/gshadow
      tags:
        - file_permissions_etc_gshadow
        - medium_severity
        - configure_strategy
        - low_complexity
        - low_disruption
        - NIST-800-53-AC-6
  </fix>
  <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
    <check-content-ref name="oval:ssg-file_permissions_etc_gshadow:def:1" href="ssg-debian8-oval.xml"/>
  </check>
  <check system="http://scap.nist.gov/schema/ocil/2">
    <check-content-ref name="ocil:ssg-file_permissions_etc_gshadow_ocil:questionnaire:1" href="ssg-
debian8-ocil.xml"/>
  </check>
</Rule>
```

Framgent de codi XCCDF amb una Rule definida. Font: ssg-debian8-xccdf.xml

Observem com la **Rule**, comença amb tot de metadades com són el títol, el rationale, referències i descripció.

Tot seguit comença amb els **fix** a aplicar i en aquest exemple observem com apareixen dos elements de fix: en color porpra el basat en shell script i en color verd el basat en Ansible. Sense entrar en molts detalls, direm que els shell scripts és un llenguatge programàtic amb instruccions a executar de forma imperativa. Per contra, Ansible és un llenguatge descriptiu en el que s'indica què es vol aconseguir i no els passos necessaris per a fer-ho; és agnòstic al sistema operatiu que hi hagi a sota.

Tant amb el fix basat en shell script com en ansible, el resultat és el mateix: s'assignaria un valor de 0640 en els permisos del fitxer */etc/gshadow*.

Finalment, però no menys important, dins de la Rule observem que s'hi defineix un apartat de **check** basats en OVAL i OCIL, marcats en color blau i taronja a l'exemple. Amb això comprovem què aporta SCAP: interoperabilitat entre diferents elements com son XCCDF, OCIL i OVAL per tal que es puguin usar conjuntament. Es delega a OVAL/OCIL el check sobre si s'ha d'aplicar o no el fix i XCCDF es pot dedicar simplement a solucionar-ho.

Aquests checks permetran definir quina comprovació s'haurà de fer en el sistema per determinar si cal o no aplicar el fix proposat per XCCDF. En el cas de OVAL, per exemple:

```
<check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
    <check-content-ref name="oval:ssg-file_permissions_etc_gshadow:def:1" href="ssg-debian8-oval.xml"/>
</check>
```

Check de OVAL definit en el XCCDF de la regla "file_permissions_etc_gshadow".

Si naveguem per dins el fitxer OVAL que ens referencia: sgg-debian8-oval.xml (en color blau), podrem trobar la referència al check oval sol·licitat ssg-file_permissions_etc_gshadow:def1 (en color verd), que fa referència al test oval:ssg-test_etc_gshadow:tst:1 (en vermell):

```
<ns0:definition class="compliance" id="oval:ssg-file_permissions_etc_gshadow:def:1" version="1">
    <ns0:metadata>
      <ns0:title>Verify /etc/gshadow Permissions</ns0:title>
      <ns0:affected family="unix">
      </ns0:affected>
        <ns0:description>This test makes sure that /etc/gshadow is owned by 0, group owned by 42, and has mode 0640. If
    the target file or directory has an extended ACL then it will fail the mode check.</ns0:description>
      <reference ref_id="file_permissions_etc_gshadow" source="ssg"/>
    </ns0:metadata>
    <ns0:criteria>
      <ns0:criterion test_ref="oval:ssg-test_etc_gshadow:tst:1"/>
    </ns0:criteria>
  </ns0:definition>
```

Definició de ssg-file_permissions_etc_gshadow:def:1. Font: sgg-debian8-oval.xml

Finalment, el test oval:ssg-test_etc_gshadow:tst:1 fa referència a un conjunt d'estats esperats pel fitxer */etc/gshadow*. En concret verifica el owner, el gid i el mode.

```
<ns3:file_test check="all" check_existence="all_exist" comment="/etc/gshadow mode and ownership"
                                                                    id="oval:ssg-test_etc_gshadow:tst:1" version="1">
    <ns3:object object_ref="oval:ssg-object_etc_gshadow:obj:1"/>
    <ns3:state state_ref="oval:ssg-_etc_gshadow_state_uid_0:ste:1"/>
    <ns3:state state_ref="oval:ssg-_etc_gshadow_state_gid_42:ste:1"/>
    <ns3:state state_ref="oval:ssg-_etc_gshadow_state_mode_0640:ste:1"/>
</ns3:file_test>
```

Estats del fitxer */etc/gshadow* definits via OVAL

Malgrat que la estructura inicialment sembli farragosa, permet un reaprofitament de codi que, al cap i a la fi, estalvia temps.

4) Modificar les checklists

En aquest capítol personalitzarem les checklist de SCAP Security Guides per comprovar com es podrien adequar a les nostres necessitats.

Veurem dues formes de fer-ho: mitjançant el tailoring i mitjançant la modificació de contingut XCCDF directament.

La primera opció ens permet fer petits ajustos sobre les guies existents. És la opció amb menys dificultat, perquè es limita a triar quines comprovacions existents s'han de dur a terme i amb quins valors s'han de comparar.
Ara bé, el tailoring no ens permetrà fer comprovacions sobre elements que no formen part de cap guia existent; ens caldrà tocar els fitxers XML directament si volem crear de zero un check inexistent.

4.1) Tailoring de checklists

En el cas del tailoring o customització de els guies existents, el que farem és treballar amb el SCAP Workbench per tal de fer petits ajustos sobre les guies existents. Això ens permetrà afegir i treure checks i modificar els valors que es volen assignar.

Com hem vist en el capítol anterior, les Security Guides de SCAP segueixen les recomanacions de estàndards (PCI-DSS, USGCB o DISA) que poden no adaptar-se completament a la nostra normativa de seguretat.
Farem aquesta personalització o «*tailoring*» usant la eina SCAP Workbench, que d'una manera gràfica permet ajustar quins checks volem executar i amb quins valors.

Començarem obrint el SCAP Workbench i triant una política per una de les nostres màquines. Per dur a terme aquest exemple usarem el DataStream de CentOS i triarem el perfil de «Standard System Security Profile for Red Hat Enterprise Linux 7» que conté un total de 51 checks.
Farem la avaluació sobre el sistema CentOS7 que tenim a la ip 192.168.56.102 que ja sabem, del capítol anterior, que superava 45 dels 51 tests.

Començarem la personalització clicant al botó de «customize», que ens mostrarà un diàleg en el que hem d'introduir el ID del nou Profile. El mateix diàleg ens informa de la importància de triar bé aquest identificador: no es pot canviar un cop assignat i ha de tenir un format preestablert de la forma:

xccdf_{**reverse_DNS**}_profile_{**rest of the ID**}

En el nostre cas, tractant-se d'un exemple que no s'exportarà a cap altre sistema, triarem el nom següent:

xccdf_local.fdantitfm.checklist_profile_standard

Acte seguit, se'ns obrirà el «Customization dialog» amb totes les opcions (916) disponibles en la guia de Red Hat Linux Enterprise. Aquí és on podrem triar quins checks volem passar al nostre sistema i quins valors han de prendre. En la Figura XIII mostrem com s'ajustaria el port en el que ha d'escoltar el SSH.

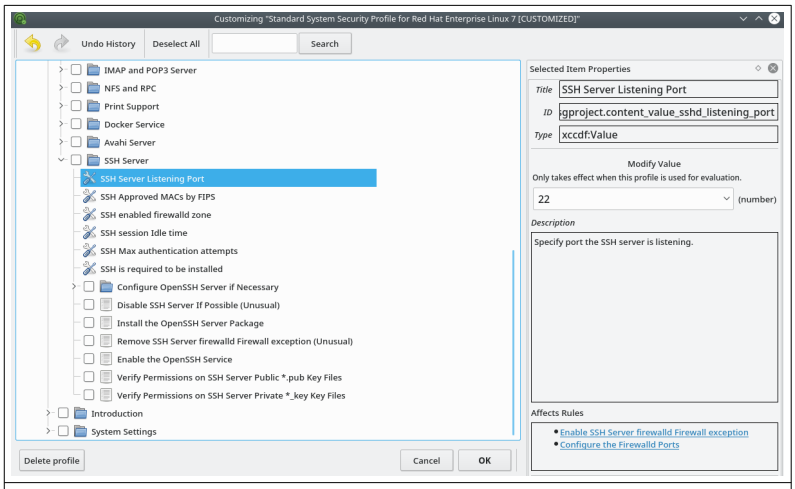

Figura XIII. Personalització del port del SSH

És important saber distingir quin tipus d'element estem habilitant o modificant en personalitzar un perfil. Distingirem els següents tipus d'elements:

Icona	Descripció
	Defineix un valor (tipus xccdf:Value)
	Defineix una regla (tipus xccdf:Rule)
	Defineix un grup (tipus xccdf:Group)

Cal destacar que modificar un valor per si sol no farà cap canvi en el sistema; només ajustant una regla podrem produir canvis en el sistema. Per tal de simplificar-nos aquesta gestió, els elements de tipus valor incorporen enllaços a les regles els utilitzen.

Per tal de simplificar els exemples al màxim, desmarcarem totes les regles que venen marcades per defecte en la guia, de forma que la nostra personalització tindrà només aquelles regles i valors que siguin necessaris. Aquesta decisió ens simplificarà al màxim els fitxers XCCDF.

En el següent exemple, pretenem fer el següent canvi en el sistema: **Volem ajustar el temps d'espera del SSH abans de tallar una connexió inactiva a 10 segons**.

Aquest paràmetre s'ajusta mitjançant el valor «*SSH session Idle time*» dins del grup de valors de SSH Server. Aquest paràmetre té, en la guia RHEL, un valor de 7200. L'ajustarem perquè prengui el valor de 10.

Observem en la Figura XIV com la aplicació ens informa que la regla associada és «*Set Idle Timeout Interval*» i ens dóna un enllaç directe per habilitar-la.

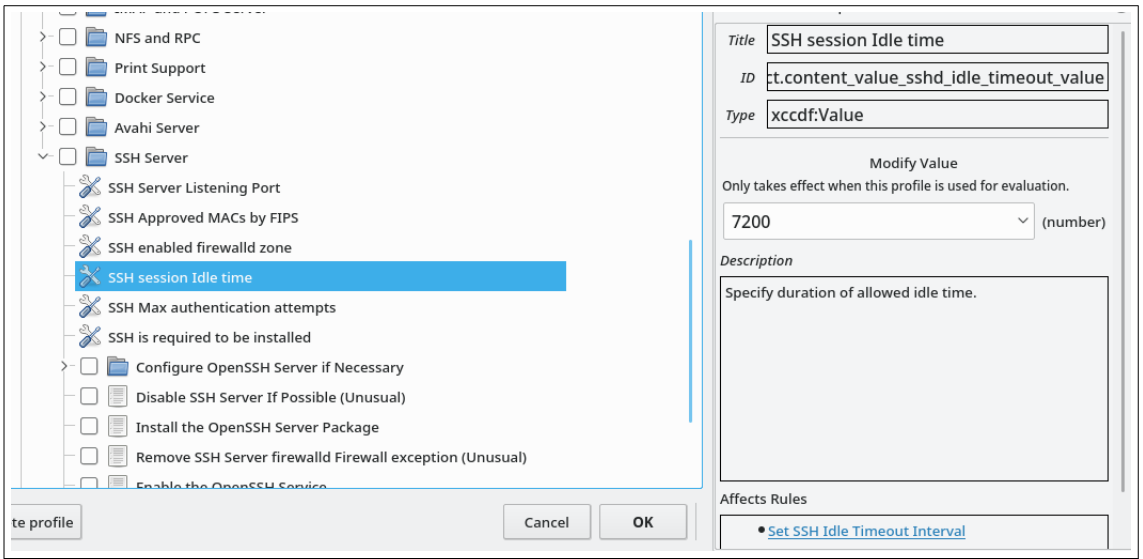

Figura XIV. Valor per «SSH session Idle Time» i regla associada

Anirem a aquesta regla, ja sigui navegant pel menú de les regles o clicant l'enllaç i la habilitarem.
Al fer-ho, tancarem el menú de personalització i observarem com en la pantalla principal ara se'ns informa que la nostra personalització consta d'una única regla(Figura XV).

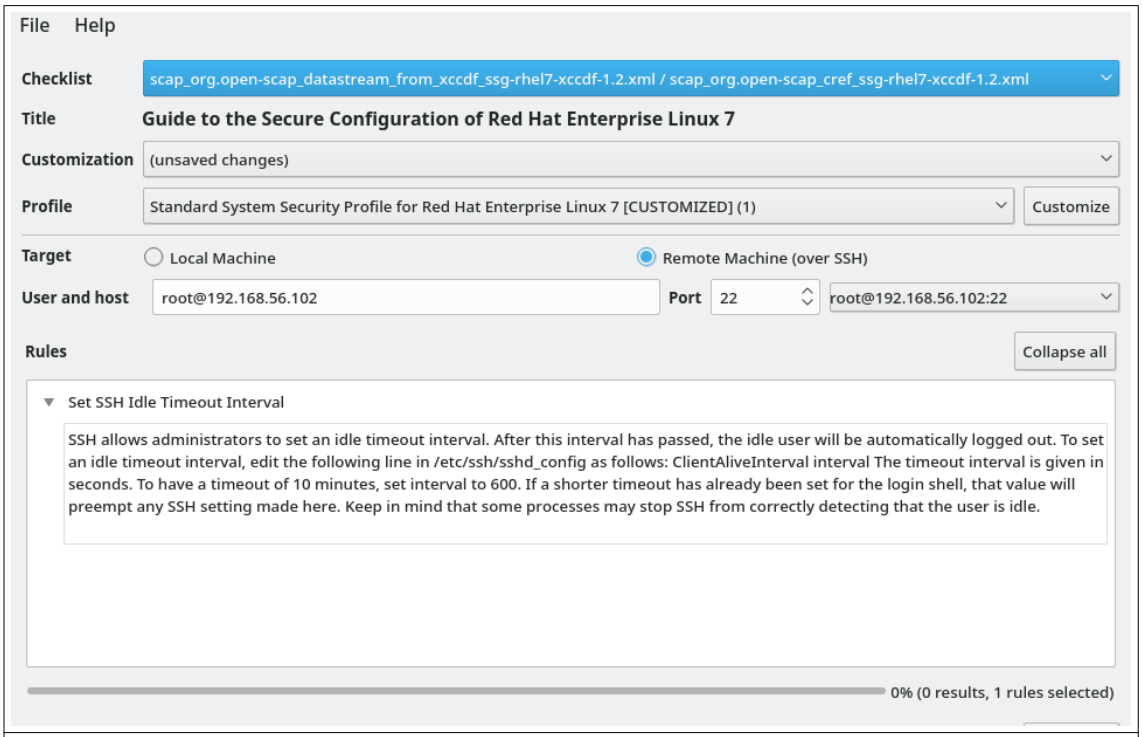

Figura XV. La personalització queda amb una sola regla habilitada

Marquem la opció de remediation i llancem la avaluació. Observarem com el resultat per la nostra regla passa a ser «*fixed*» i si anem a la màquina client, ens trobarem que en el fitxer */etc/ssh/sshd_config* s'hi han afegit les línies de text que mostrem en la Figura XVI:

```
# Per CCE-CCE-27433-2: Set ClientAliveInterval 10 in /etc/ssh/sshd_config
ClientAliveInterval 10
```

Figura XVI. CCE afegit en sshd_config per la Rule personalitzada

Observem doncs com SCAP Workbench ha modificat el fitxer de configuració de la màquina remota per satisfer els requisits de la nostra regla.

Com ja hem vist, amb SCAP tots els ajustos a la configuració estan identificats d'una manera unívoca mitjançant un CCE (Common Configuration Enumeration) i el WorkBench ha afegit la referència al CCE-27433-2 en aplicar el canvi en el fitxer de configuració.

Per tal de veure com ho ha fet, desarem els canvis fets en un fitxer i n'estudiarem el contingut. Per fer-ho, Cliquem a Fitxer→Desar la customització.

Aquest fitxer generat serà el que podrem exportar a altres sistemes per avaluar i, si cal, fixar els paràmetres de la nostra guia.

Per començar ens fixarem que el document desat es tracta d'un fitxer de DataStream, en el que el XML conté un node arrel de tipus Tailoring. Aquest és el tipus de contingut que permet fer ajustos sobre checklists, motiu pel qual, els sub-nodes són de tipus XCCDF.

Ens fixarem amb les darreres línies del fitxer generat:

```
<xccdf:select idref="xccdf_org.ssgproject.content_rule_sshd_set_idle_timeout" selected="true"/>
<xccdf:select idref="xccdf_org.ssgproject.content_group_ssh_server" selected="true"/>
<xccdf:select idref="xccdf_org.ssgproject.content_group_ssh" selected="true"/>
<xccdf:select idref="xccdf_org.ssgproject.content_group_services" selected="true"/>
<xccdf:set-value idref="xccdf_org.ssgproject.content_value_sshd_idle_timeout_value">10</xccdf:set-value>
```

Darreres línies del fitxer de customització o tailoring de SCAP Workbench

D'aquestes cinc línies, de tipus XCCDF, destacarem que a la primera, s'habilita la regla «*content_rule_sshd_set_idle_timeout*" mitjançant una instrucció de tipus «*select*» assignant-li un valor de «*true*».

En la darrera línia, mitjançant una instrucció de tipus «*set-value*», s'estableix el valor de la variable «*content_value_sshd_idle_timeout_value*» a 10.

4.2) Modificació manual dels continguts OVAL i XCCDF

La necessitat de tocar els fitxers OVAL o XCCDF directament apareixerà en el moment en que s'hagin de fer comprovacions sobre elements del sistema que no estan en cap altre guia existent.

No és l'objectiu d'aquest treball crear una llista de comprovacions completa, de forma que ens limitarem a modificar una guia existent existent per crear-hi un check inventat.
En el nostre cas, pretenem fer una checklist que comprovi que existeix el fitxer */tmp/fdanti* i que té els permisos 0600.

Cal recordar que el SCAP Workbench només sap treballar amb fitxers de DataStream, de forma que copiarem la guia de seguretat del CentOS7 a un nou fitxer per poder modificar-lo sense afectar a la original.

Dins d'aquest DataStream hi tindrem inclosos els nodes corresponents a les instruccions XCCDF i OVAL que volem ajustar. El contingut íntegre d'aquestes modificacions es poden trobar a l'Annex 1 d'aquest treball.

Començarem afegint una nova regla en el node XCCDF (Annex1, Regla XCCDF):

- Assignarem un ID=content_rule_file_permissions_tmp_fdanti a la nova regla.
- Ajustarem el títol, descripció i rationale per descriure l'objectiu de la regla.
- Crearem un node de fix, amb ID «file_permissions_tmp_fdanti», amb estratègia «configure» i que passarà al bash la instrucció «chmod 0600 /tmp/fdanti»
- Crearem un node check que apuntarà a una nova entrada OVAL definida per l'identificador «oval:ssg-file_permissions_tmp_fdanti:def:1»

A nivell de OVAL, haurem d'afegir el test sobre el fitxer en qüestió. Per aquest motiu:

- Crearem una entrada a la SGG de tipus OVAL amb el nom referenciat en l'apartat del XCCDF: «oval:ssg-file_permissions_tmp_fdanti:def:1» i que enllaçarà amb el check a executar. (Annex1, La definició OVAL arrel).
- Crearem el check OVAL a executar i que enllaçarà amb la definició del fitxer a tractar i els estats del fitxer. (Annex1, La definició OVAL del test)
- Definirem un element OVAL per ajustar el fitxer que volem avaluar. (Annex1, La definició del fitxer a avaluar)
- Definirem el estat amb el que volem que estigui el nou fitxer. (Annex1, La definició de l'estat del fitxer a avaluar)

A nivell de OCIL, haurem d'afegir els elements del llenguatge següents:

- Crearem un testaction per a retornar PASS o FAIL en cas de superar o no el check. (Annex1, Definició OCIL del resultat del check)
- Crearem un element de tipus «Question» per a poder expressar una preguta al usuari si és necessari. (Annex1, Definició OCIL de la «question»)

Amb aquests canvis, podem desar el fitxer modificat, i obrir-lo amb amb el SCAP Workbench. Ho fem així perquè la interfície gràfica ens facilita molt poder ajustar quines comprovacions executem i quines no.

Farem una customització per tal d'executar només el check que acabem de crear i comprovarem que ens apareix per pantalla.

Llancem el check i comprovem com s'executa en la màquina remota.

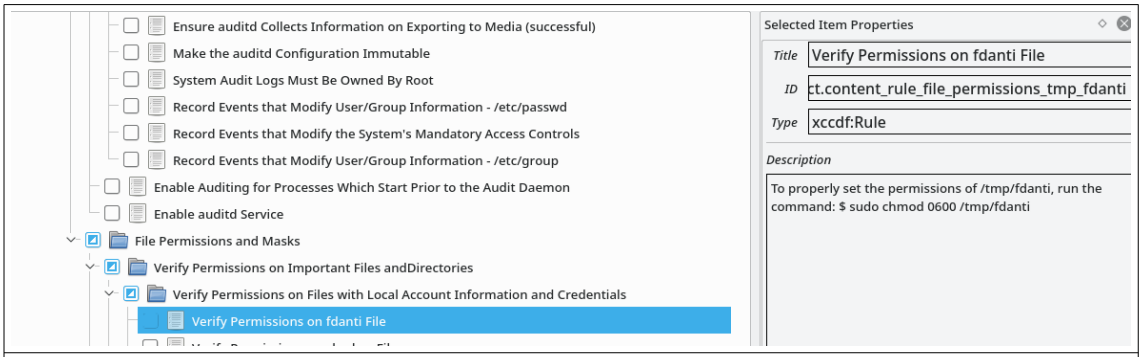

Figura XVII. Marquem només el check que acabem de crear

5) Conclusions

Amb aquest treball final de màster ens hem introduït a la tecnologia SCAP i com usar les seves eines per avaluar els sistemes mitjançant llistes de proves. Hem vist com es poden ajustar aquestes comprovacions per adaptar-les a les nostres necessitats mitjançant el Tailoring o l'ajust directe dels fitxers de DataStream.

Hem pogut comprovar com n'arriba a ser de farragós modificar el codi XCCDF i OVAL, ja que hi ha tot d'elements interrelacionats i modificar una petita part implica retocar molts elements: afegir una regla a XCCDF implica definir els corresponents tests en OVAL i també definir els elements de OCIL necessaris per a poder obtenir els resultats.
Ara bé, hem de dir que aquest llenguatge té els seus avantatges: es pot reaprofitar el codi de forma que s'eviten errors i unifica la forma de programar.
No hem d'oblidar que les checklist que hem vist es basen en l'estàndard SCAP i que com a tals, han de ser usables per a qualsevol altre sistema. D'aquí a que el seu domini pugui suposar una corba d'aprenentatge molt brusca.

A nivell de la feina feta, hem de ser crítics. Les expectatives que teníem a principi de curs s'han vist fustades per tot de circumstàncies que no havíem previst i que ja hem comentat en l'apartat de planificació: entorn de treball no adequat, paqueteria no suficientment actualitzada en SUSE).
Malgrat que haguem acomplert la planificació inicial, les expectatives que teníem eren superiors a això i ens hem deixat tot d'idees pel camí que hem hagut de deixar fora de l'àmbit d'aquest treball. La relació d'idees que volíem implementar, més enllà de la planificació inicial eren:

- Implementar aquesta solució en un entorn real, amb infraestructura basada en SUSE. No hem pogut sortir de l'entorn virtual local ni tampoc hem pogut veure com interactua SCAP amb el SUSE Manager.
- Investigar com actuar amb contenidors. Com passar checklists de SCAP a imatges de contenidors per poder decidir quan convé refer la imatge i procedir a la eliminació dels contenidors antiquats? En un futur proper, amb kubernetes apareixent per tot arreu, convé poder disposar d'eines com SCAP per comprovar els contenidors.
- Estudiar formes d'automatitzar la creació de regles pels checklist. Fer-ho manualment és una font d'errors que es podria evitar parametritzant mitjançant alguns scripts senzills

A nivell laboral, aquest treball m'ha obert la possibilitat d'implementar aquesta tecnologia en sistemes que tenim en producció, en un CPD amb més d'un miler de màquines virtuals. L'entorn és bàsicament SUSE, on existeixen solucions com SUSE Manager que permeten treballar amb llenguatge SCAP més enllà del que hem vist amb OpenSCAP. Tant si acabem implementant solucions basades en oscap com si ens decantem per SUSE Manager, el bagatge aconseguit amb aquest TFM m'anirà d'allò més bé per poder comparar funcionalitats i decidir quina de les tecnologies resulta més interessant en el nostre cas.

6) Glossari

Recollim en aquest apartat totes les abreviacions usades en la memòria.

AI	Asset Identification.
ARF	Asset Reporting Format.
CCE	Common Configuration Enumeration.
CCSS	Common Configuration Scoring System.
CERT	Computer Emergency Response Team.
CPE	Common Platform Enumeration.
CVE	Common Vulnerabilities and Exposures.
CVSS	Common Vulnerability Scoring System.
DHCP	Dynamic Host Configuration Protocol.
DISA	Defense Information System Agency.
GUI	Graphical user interface.
HTML	HyperText Markup Language.
ID	Identificador
NAT	Network Address Translation.
NIST	National Institude of Standards and Technology.
NSA	National Security Agency.
NTP	Network Time Protocol.
NVD	National Vulnerability Database.
OCIL	Open Checklist Interactive Language.
OS	Operating System.
OVAL	Open Vulnerability and Assessment Language.
PCI-DSS	Payment Card Industry Data Security Standard
SCAP	Security Content Automation Protocol.
SCE	Script Check Engine.
SETUID	Set User ID.
SLES	Suse Linux Enterprise Server.
SSH	Secure Shell.
TFM	Treball Final de Màster.
TMSAD	Trust Model for Security Automation.
TNE	En planificació temporal: Temps Necessari Estimat.
USGCB	United States Government Configuration Baseline
XCCDF	eXtensible Configuration Checklist Description Format.
XML	eXtensible Markup Language.

7) Bibliografia

Seguint el format APA

1.- Debian Wiki Team. (2017, Feb 2). **Using SCAP**. In Debian Wiki. Retrieved May 2019, from “https://wiki.debian.org/UsingSCAP”

2.- Martin Preisler. (2015, May 13). **Scanning remote machines with OpenSCAP**. Retrieved May 2019, from https://martin.preisler.me/2015/05/scanning-remote-machines-with-openscap/

3.- NIST. (2019) **Security Content Automation Protocol.** In NIST. Retrieved May 2019, from https://csrc.nist.gov/projects/security-content-automation-protocol

4.- NIST. (2019) **NVD-Home.** In NIST. Retrieved May 2019, from https://nvd.nist.gov/

5.- OpenSCAP comunity. (2016) **SCAP Security Guide**. In OpenSCAP. Retrieved June 1, 2019, from https://www.open-scap.org/security-policies/scap-security-guide/

6.- SUSE. (2919) **OVAL Descriptions.** In SUSE support. Retrieved May 2019, from https://www.suse.com/es-es/support/security/oval/

7.- NIST. (2019) **Product Validation Record**. In NIST. Retrieved May 2019, from https://csrc.nist.gov/projects/security-content-automation-protocol-validation-pr/validated-products-and-modules/142-red-hat-scap-1-2-product-validation-record

8.- Microsoft Corporation. (01/30/2019) **About SCAP extensions**. Retrieved May 2019, from https://docs.microsoft.com/en-us/sccm/compliance/plan-design/scap/about-scap

9.- OpenSCAP comunity. (2016) **OpenSCAP Download**. In OpenSCAP. Retrieved Feb 1, 2019, from https://www.open-scap.org/tools/openscap-base/#download

10.- OpenSCAP comunity. (2016) **SCE**. In OpenSCAP. Retrieved May, 2019, from https://www.open-scap.org/features/other-standards/sce/

S’ha usat altra bibliografia no referenciada en el text:

- Wikipedia contributors. (2019, May 13). **Security Content Automation Protocol**. In Wikipedia, The Free Encyclopedia. Retrieved June 1, 2019, from https://en.wikipedia.org/w/index.php?title=Security_Content_Automation_Protocol&oldid=896943249

- Wikipedia contributors. (2019, May 27). **APA style.** In Wikipedia, The Free Encyclopedia. Retrieved June 1, 2019, from https://en.wikipedia.org/w/index.php?title=APA_style&oldid=899013573

- OpenSCAP comunity. (2016) **SCAP Components**. In OpenSCAP. Retrieved June 1, 2019, from https://www.open-scap.org/features/scap-components/

- OpenSCAP comunity. (2016) **OpenSCAP**. In OpenSCAP. Retrieved June 1, 2019, from https://www.open-scap.org/

- NIST. (2019) **NVD-Home.** In NVD. Retrieved May 2019, from https://nvd.nist.gov/

- RedHat. (2019) **Documentation for Red Hat 7**. In Red Hat. Retrieved May 2019, from https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/security_guide/sect-defining_compliance_policy

- NIST. (2019) **OVAL Tutorial Definitions.** In NVD. Retrieved May 2019, from https://nvd.nist.gov/scap/docs/conference%20presentations/workshops/OVAL%20Tutorial%202%20-%20%20Definitions.pdf

- RedHat. (2019) **Security Guide – using SCAP Workbench**. In Red Hat. Retrieved May 2019, from https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/security_guide/sect-using_scap_workbench

- RedHat. (2019) **USGCB / STIG - DRAFT**. In Red Hat. Retrieved June 2019, from http://people.redhat.com/swells/scap-security-guide/tables/table-rhel7-nistrefs-ospp-rhel7.html

8) Annex 1. Modificacions al DataStream

Regla XCCDF:

```
<ns10:Rule id="xccdf_org.ssgproject.content_rule_file_permissions_tmp_fdanti"
selected="true" severity="medium">
        <ns10:title xml:lang="en-US">
                Verify Permissions on fdanti File
        </ns10:title>
        <ns10:description xml:lang="en-US">
                To properly set the permissions of <html:code>/tmp/fdanti</html:code>, run the
                command: <html:pre xml:space="preserve">$ sudo chmod 0644
                /tmp/fdanti</html:pre>
        </ns10:description>
        <ns10:rationale xml:lang="en-US">
                The <html:code>/tmp/fdanti</html:code> file contains nothing.
        </ns10:rationale>
        <ns10:fix complexity="low" disruption="low" system="urn:xccdf:fix:script:sh"
        id="file_permissions_tmp_fdanti" strategy="configure" >
                chmod 0600 /tmp/fdanti
        </ns10:fix>
        <ns10:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
                <ns10:check-content-ref href="ssg-rhel7-oval.xml" name="oval:ssg-
                file_permissions_tmp_fdanti:def:1" />
        </ns10:check>
</ns10:Rule>
```

La definició OVAL arrel:

```
<ns3:definition class="compliance" id="oval:ssg-file_permissions_tmp_fdanti:def:1"
version="2">
  <ns3:metadata>
      <ns3:title>Verify permissions on fdanti file</ns3:title>
      <ns3:affected family="unix">
        <ns3:platform>Red Hat Enterprise Linux 7</ns3:platform>
        <ns3:platform>Red Hat Enterprise Linux 6</ns3:platform>
      </ns3:affected>
      <ns3:description>Perms for /tmp/fdanti should be set correctly</ns3:description>
      <reference ref_id="file_permissions_tmp_fdanti" source="ssg" />
  </ns3:metadata>
  <ns3:criteria>
      <ns3:criterion test_ref="oval:ssg-test_file_permissions_tmp_fdanti:tst:1" />
  </ns3:criteria>
</ns3:definition>
```

La definició OVAL del test:

```
<ns7:file_test check="all" check_existence="all_exist" comment="Testing /tmp/fdanti permissions"
id="oval:ssg-test_file_permissions_tmp_fdanti:tst:1" version="1">
    <ns7:object object_ref="oval:ssg-object_file_permissions_tmp_fdanti:obj:1" />
    <ns7:state state_ref="oval:ssg-state_file_permissions_tmp_fdanti:ste:1" />
</ns7:file_test>
```

La definició OVAL del fitxer a avaluar:

```
<ns7:file_object comment="/tmp/fdanti" id="oval:ssg-object_file_permissions_tmp_fdanti:obj:1
                                                                                    version="1">
  <ns7:filepath>/tmp/fdanti</ns7:filepath>
</ns7:file_object>
```

La definició OVAL del estat del fitxer a avaluar:

```
<ns7:file_state id="oval:ssg-state_file_permissions_tmp_fdanti:ste:1" version="2">
        <ns7:suid datatype="boolean">false</ns7:suid>
        <ns7:sgid datatype="boolean">false</ns7:sgid>
        <ns7:sticky datatype="boolean">false</ns7:sticky>
        <ns7:uexec datatype="boolean">false</ns7:uexec>
        <ns7:gwrite datatype="boolean">false</ns7:gwrite>
        <ns7:gexec datatype="boolean">false</ns7:gexec>
        <ns7:owrite datatype="boolean">true</ns7:owrite>
        <ns7:oexec datatype="boolean">false</ns7:oexec>
</ns7:file_state>
```

Definició OCIL del resultat del check:

```
<ns9:boolean_question_test_action

id="ocil:ssg-file_permissions_tmp_fdanti_action:testaction:1" question_ref="ocil:ssg-
file_permissions_tmp_fdanti_question:question:1">

        <ns9:when_true>
          <ns9:result>PASS</ns9:result>
        </ns9:when_true>
        <ns9:when_false>
          <ns9:result>FAIL</ns9:result>
        </ns9:when_false>
      </ns9:boolean_question_test_action>
```

Definició OCIL de la «question»:

```
<ns9:boolean_question id="ocil:ssg-file_permissions_tmp_fdanti_question:question:1">
  <ns9:question_text>
    To check the permissions of /tmp/fdanti, run the command: $ ls -l /tmp/fdanti
  </ns9:question_text>
</ns9:boolean_question>
```