Tema 6: Seguretat a les aplicacions

Seguretat Informàtica (SI-FIB)

Ton Rodriguez | SI | Setembre 2018

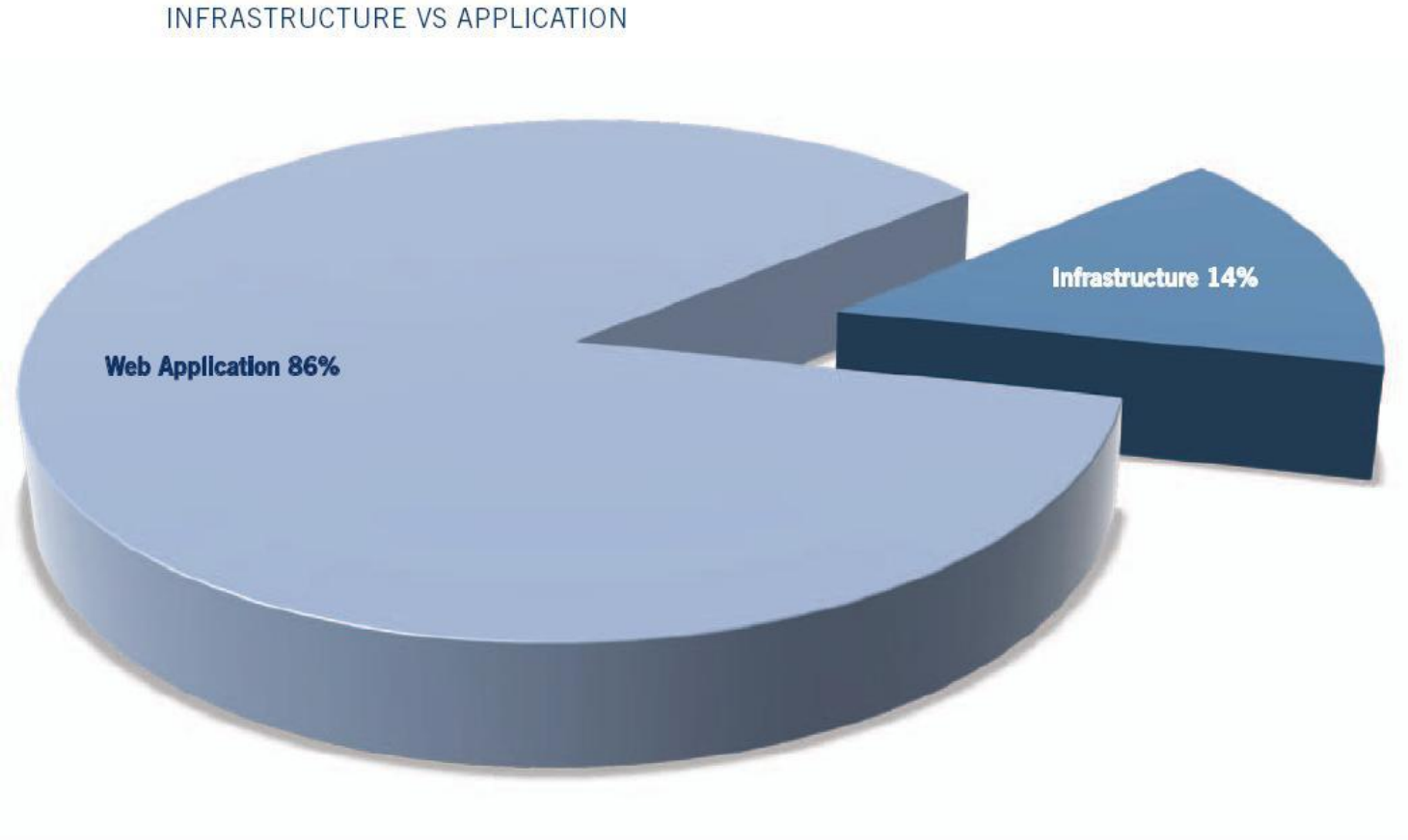
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

inLab FIB talent & tech

ESCERT

Introducció

Introducció



Areas of the compromised systems exploited.

UK Security Breach Investigations Report 2010 by 7safe

Introducció

- Causes de software insegur
  - Vulnerabilitats
  - Estructura organitzativa, processos de desenvolupament, tecnologia
  - Desenvolupadors
  - Requisits legals
  - Increment de la connectivitat y la complexitat

Introducció

- NO existeix un sistema 100% segur
  - Vulnerabilitats teòriques
  - Vulnerabilitats reals (exploits)

- La debilitat d'un sistema permet a un atacant violar la:
  - Confidencialitat
  - Integritat
  - Disponibilitat

Introducció

- No existeix la bala de plata.
- Assegurar un programa de seguretat rentable y exhaustiu mitjançant SDLC.
- Provar aviat i sovint.
- Entendre l'abast de la seguretat que requereix el projecte.
- Pensar creativament.
- Fer servir les eines adequades.
- Utilitzar el codi font quan estigui disponible.
- Documentar els resultats de les proves.

SDLC


CICLO DE VIDA DEL SOFTWARE
DEFINICIÓN DE NECESIDADES
ANÁLISIS
DISEÑO
CODIFICACIÓN
PRUEBAS
VALIDACIÓN
MANTENIMIENTO Y EVOLUCIÓN

Perquè programació segura?

- Les aplicacions web són la forma més fàcil d'atacar un sistema
- La seguretat **NO** pot ser només externa al desenvolupament
- Hem de ser conscients dels problemes i de les possibles solucions
- No hem de suposar que els usuaris utilitzaran la nostra aplicació únicament com nosaltres l'hem dissenyat.
- No podem confiar a cegues en el que ens arriba del navegador de l'usuari.


¡TO ER MUNDO E GÜENO!
¡ESA ESPAÑA TAN DIVERTIDA!
un film de
SUMMERS.
DAGA FILMS SA
UNA PRODUCCION ... M.A. S. A.-KALENDER FILMS INTERNATIONAL, S. A.-PARAGUAS FILMS, S. A.


TRUST NO ONE

Introducció

- OWASP (Open Web Application Security Project)
- Creada amb la finalitat de determinar y combatre les causes que fan que el software sigui insegur.
- Fundació sense ànim de lucre.
- La participació es gratuïta y oberta a tothom.
- Pretén establir mètodes de treball segurs a l'hora de desenvolupar aplicacions web
- Lliure de pressions corporatives

OWASP TOP10

OWASP TOP10

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

Errors de programació?

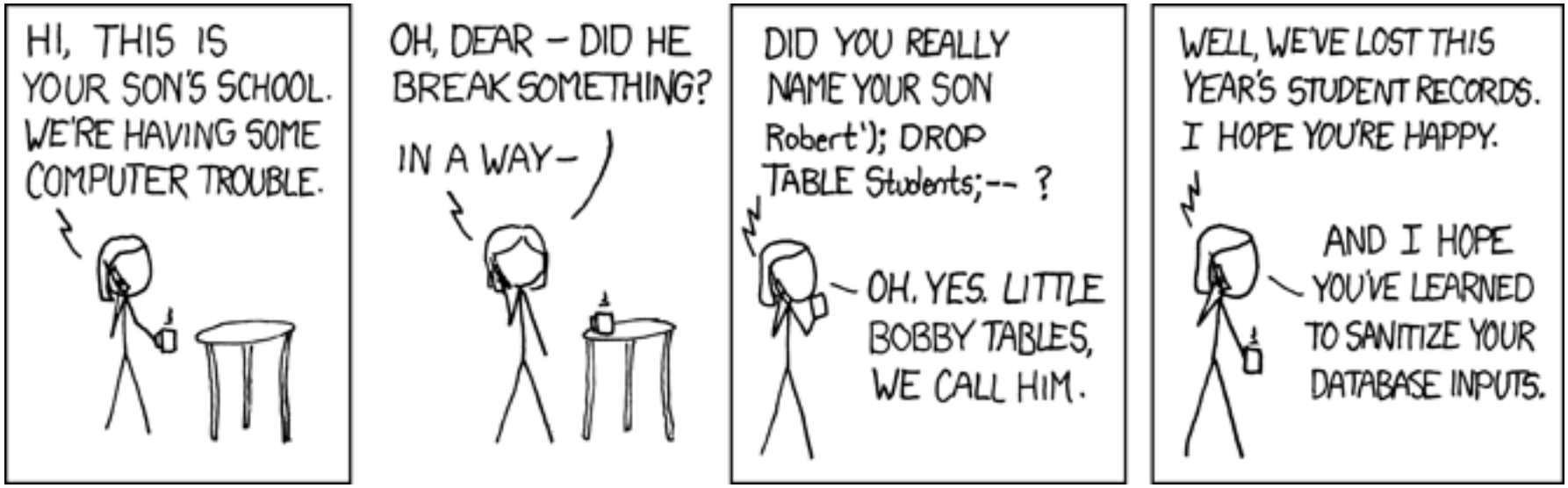
1. **Injection**
2. **Broken Authentication**
3. **Sensitive Data Exposure**
4. **XML External Entities (XEE)**
5. **Broken Access Control**
6. Security Misconfiguration
7. **Cross-Site Scripting (XSS)**
8. **Insecure Deserialization**
9. Using Components with Known Vulnerabilities
10. Insufficient Logging&Monitoring

Errors de programació?

1. **Injection**
2. **Broken Authentication**
3. **Sensitive Data Exposure**
4. **XML External Entities (XEE)**
5. **Broken Access Control**
6. Security Misconfiguration
7. **Cross-Site Scripting (XSS)**
8. **Insecure Deserialization**
9. Using Components with Known Vulnerabilities
10. Insufficient Logging&Monitoring

Injection

- Es produeix quan passem dades no validades a un intèrpret i l'enganyem per que les executi com si fossin comandes escrites pel programador



http://xkcd.com/327/

Injection - Exemple

- El cas típic es injecció de SQL degut a que concatenem codi SQL amb dades
  - "SELECT descripcio FROM taula WHERE id=" + id
    - id="1 OR 1=1"
    - id="1 UNION SELECT password FROM usuaris"
- Ojo: no només hi ha injecció de SQL
  - Injecció LDAP (tipic en els logins)
  - Injecció de shell script (executant comandes externes)

Quin és el problema?

- Construïm SQL concatenant strings

- Solució
  - SQL paramètric sempre que sigui possible
    - Es SQL amb forats pels valors
    - El motor de BD diferencia entre SQL i valors

- Mitja solució (quan no és possible)
  - Encoding dels paràmetres per SQL
    - Depenent de la base de dades a utilitzar.
    - No 100% segura

Errors de programació?

1. Injection
2. **Broken Authentication**
3. Sensitive Data Exposure
4. XML External Entities (XEE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging&Monitoring

Broken Authentication

- “Tots els problemes relacionats amb el procés de login, les sessions i les credencials"

- SSL als formularis de login
- Cookies "httpOnly" / "secure"
- Session Fixation
- URL Rewriting
- Emmagatzemament dels passwords

HTTPS

- Permet que les connexions vagin per un canal segur encriptat

- Tota web que demani username i password ha d'estar protegida amb SSL (i TLS!)
- No només el formulari de login, sinó TOT perquè circulen les cookies de sessió
- No tenir HTTPS permet "esnifar" les connexions
- Especialment problemàtic en xarxes WIFI

Cookies "Secure"

- Flag que fa que les cookies només s'enviïn per HTTPS
  - Exemple:
    - Activar-les a una aplicació Java en Tomcat, configurant el connector de HTTP normal

```
<Connector port="8080" .... secure="true"/>
```

HTTPOnly

- Permet que les cookies no es vegin en Javascript

- Per comprovar si està activat, teclegem a la barra de la URL
  - `javascript:alert(document.cookie);`
- Si apareix la sessió, som vulnerables
  - JSESSIONID
  - PHPSESSID
  - ...

- Exercici: Proveu a diverses webs...

HTTPOnly. Com ho solucionem?

- Quan s'envia la cookie de sessió s'ha d'afegir l'atribut "HttpOnly“
  - Tomcat 7, per defecte ho envia.
  - PHP. És un paràmetre del php.ini
    - http://www.php.net/manual/en/session.configuration.php#ini.session.cookie-httponly

Session Fixation

- Es produeix quan hi ha alguna forma de passar una sessió començada a un altre usuari via URL i la sessió no es regenera quan fem login.

- Error #1. Es permet passar una sessió per URL
- Error #2. No creem una nova sessió quan fem login

Guardar els passwords en clar

- Els passwords dels usuaris **MAI** s'han de guardar en clar!
- Sempre s'ha de guardar un hash i comparar el hash del password entrat amb el de la BD
- Hi ha diverses funcions de hash. Les mes habituals son MD5 i SHA1
  - Es recomanable utilitzar funcions mes segures, com SHA 256

És suficient?

- Raibow tables: https://crackstation.net/
  - Per passwords simples, tenir el hash = tenir el password en clar
- Solució: Salt
  - Concatenem un valor aleatori al password abans de fer el hash
  - Guardem aquest "Salt" com un valor associat a l'usuari.
  - Quan hem de validar l'usauri, consultem el seu Salt i el concatemem al password que ha entrat l'usuari abans de fer el hash

- https://crackstation.net/hashing-security.htm

Errors de programació?

1. Injection
2. Broken Authentication
3. **Sensitive Data Exposure**
4. XML External Entities (XEE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging&Monitoring

Sensitive Data Exposure

- Vol dir que per culpa del nostre programa es pugui accedir a informació relacionada amb el nostre sistema
  - Com guardem els nostres passwords de BD?
    - Estem exposant passwords dels usuaris?
  - Com tractem les excepcions?
  - Estem exposant estructura de la BD?

Exemple

Estado HTTP 500 - Error de Sintaxis en sentencia SQL "SELECT * FROM COMENTARIS WHERE AUTOR="[*]'"

---

type Informe de Excepción

mensaje Error de Sintaxis en sentencia SQL "SELECT * FROM COMENTARIS WHERE AUTOR="[*]'"

descripción El servidor encontró un error interno que hizo que no pudiera rellenar este requerimiento.

excepción

```
java.lang.RuntimeException: Error de Sintaxis en sentencia SQL "SELECT * FROM COMENTARIS WHERE AUTOR=''[*]'"
Syntax error in SQL statement "SELECT * FROM COMENTARIS WHERE AUTOR=''[*]'"; SQL statement:
SELECT * FROM COMENTARIS WHERE AUTOR=''' [42000-188]
        edu.upc.escert.curs.repositori.vulnerable.RepositoriComentaris.getComentarisFromSQL(RepositoriComentaris.java:62)
        edu.upc.escert.curs.repositori.vulnerable.RepositoriComentaris.getComentarisPerAutor(RepositoriComentaris.java:78)
        edu.upc.escert.curs.LlistaComentarisServlet.doGet(LlistaComentarisServlet.java:31)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:621)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:728)
```

nota La traza completa de la causa de este error se encuentra en los archivos de diario de Apache Tomcat/7.0.39.

Apache Tomcat/7.0.39

Solució

- No mostrar les pagines d'error per defecte
  - Crear una o més pàgines d'error personalitzades
  - (Java/Tomcat) Configurar error-pages a l'aplicació al web.xml

```
<error-page>
    <error-code>403</error-code>
    <location>/html/403.html</location>
</error-page>
<error-page>
    <location>/html/error.html</location>
</error-page>
```

Errors de programació?

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. **XML External Entities (XEE)**
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging&Monitoring

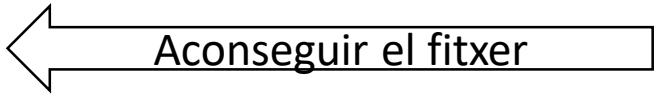
XML External Entities (XEE)

- Consisteix en aconseguir que un processador de XML rebi un fitxer maliciós o bé en mètodes que permetin incloure contingut maliciós a un XML que serà processat posteriorment.
  - Per defecte molts processadors antics permeten especificar URI externes.

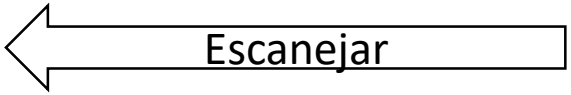
- Si s'aconsegueix explotar es pot aconseguir:
  - Extracció de dades internes
  - Denegació de servei
  - Peticions des del servidor
  - ...

Exemples

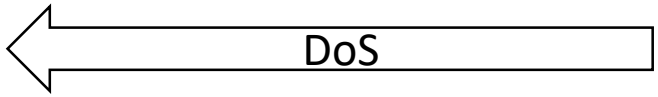
```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
 <!ELEMENT foo ANY >
 <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
 <foo>&xxe;</foo>
```




```
<!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>
```




```
<!ENTITY xxe SYSTEM "file:///dev/random" >]>
```



Solucions?

- Validar els fitxers XML d'entrada
- Actualitzar els parsers (processadors/serialitzadors)
- Deshabilitar el parsing de DTD
- Creació de llistes blanques
- Utilització de formats més simples com per exemple json

Errors de programació?

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XEE)
5. **Broken Access Control**
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging&Monitoring

Broken Access Control

- Primera vulnerabilitat que és totalment de **disseny** de l'aplicació

- Vol dir tenir accés a coses a les que no hauríem de tenir accés, ja sigui per veure-les o per modificar-les.
  - No gestionem bé les àrees que estan protegides.

Exemple

- Fem login a una aplicació de fòrum:
  - Només hauríem de poder esborrar els nostres comentaris, però...
  - http://www.exemple.com/esborrar?id=1
- Com ho solucionem?
  - La lògica del programa ha de validar aquests paràmetres que de fer son referències a objectes de la nostra aplicació
  - Assegurar-se si cal passar-lo o el podem obtenir al servidor
- Com gestionem que està protegit i que no?
- Proteccions declaratives sempre que sigui possible.

Com solucionar-ho? Seguretat declarativa

- Les URL protegides i els rols que han de tenir els usuaris han d'estar el més aïllats possible de la resta del programa

- Deixarem en mans del servidor
  - La gestió de les sessions
  - La comprovació del login una vegada donada la llista d'usuaris (anomenada Realm a Tomcat)
  - La comprovació dels rols
  - La redirecció cap a la pantalla de login
  - API genèrica per saber qui ha accedit per codi

Errors de programació?

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XEE)
5. Broken Access Control
6. Security Misconfiguration
7. **Cross-Site Scripting (XSS)**
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging&Monitoring

Cross-site scripting (XSS)

- Es produeix quan un atacant pot arribar a executar codi javascript escrit per ell a la pàgina que està mirant un altre usuari
  - Reflectit
  - Permanent

Moltes formes d'escriure codi Javascript

- `<script>alert("hola")</script>`
- `<a onmouseover="alert('hola')">hola</a>`
- `<img src="javascript:alert('hola')">`

- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

- **No n'hi ha prou amb carregar-se el tag script!**

Com evitar el XSS?

- **Mantra: Filter on input, escape on output**
  - http://lukeplant.me.uk/blog/posts/why-escape-on-input-is-a-bad-idea/
- Hem de tenir quan abans millor valors "nets" de l'entrada
- Hem d'evitar guardar a la base de dades valors escapats
- Hem de decidir com volem escapar depenent del context de sortida:
  - HTML
  - Un valor d'un atribut HTML
  - JSON
  - ...

Filtrar l'entrada

- Hem de filtrar segons la semàntica de l'aplicació, no pensant en caràcters estanys per HTML
  - **L'hospitalet** és un nom de ciutat vàlid
  - **D'Artagnan** és un cognom
  - **In & Out** és un nom de pel·lícula correcte
  - **Només si x<5 o x>10** és un comentari vàlid
  - **SELECT * FROM PROVA WHERE ID='1'** pot ser una resposta vàlida en un qüestionari sobre SQL
  - **Això es "important"** és un comentari vàlid
  - **Això es < b>important</ b>** és un comentari vàlid que pot arribar a la nostra aplicació si tenim un camp de rich text
- No podem filtrar a saco i carregar-nos tots els <, >, ' i "

HTML o no HTML?

- No Permeten HTML? (exemple: camp títol)
  - Filtrar l'entrada segons convingui
  - Codificar la sortida segons el lloc on l'haguem d'escriure

- Permeten HTML? (exemple: camp comentari)
  - Fer neteja (sanitization) sempre en base a una llista blanca
  - Per màxima seguretat, sanititzar a la sortida, però NO fer encoding

Errors de programació?

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XEE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. **Insecure Deserialization**
9. Using Components with Known Vulnerabilities
10. Insufficient Logging&Monitoring

Insecure Deserialization

- Es produeix quan al des-serialitzar un objecte que pot enviar un atacant es modifica el comportament de l'aplicació.
- Aquest objecte pot ser:
  - El contingut d'una RPC
  - Una cookie
  - Tokens d'una API
  - Una crida a un web service
  - ...
- O bé es crea un objecte maliciós o bé es modifica el contingut d'un que ja existeix

Exemple (modificació)

- En una aplicació PHP es crea una “supercookie” amb algunes dades de l’usuari i el rol
  - [username,id,role] -> [berta,42,user]
- Un atacant modifica l’objecte per canviar el comportament de l’aplicació
  - [berta,42,user] -> [berta,42,admin]
- Al des-serialitzar la “supercookie” l’aplicació es comporta com si la berta fos administradora.

Mesures a prendre

- Implementar verificacions d'integritat amb signatures per evitar la modificació
- Limitar i restringir els tipus de dades de les crides
- Des-serialitzar en entorns amb pocs privilegis
  - Monitoritzar aquests entorns
    - Generar logs de les excepcions
    - Monitoritzar el tràfic
    - Monitoritzar si hi ha usuaris des-serialitzant continuament

BONUS (2013): Cross-Site Request Forgery (CSRF)

- Es tracta de forçar que un usuari loguejat correctament realitzi una acció involuntàriament.

- Aprofita la confiança d'un servidor en el navegador d'un usuari legítim.

Cross-Site Request Forgery - Exemple

- Suposem que l'Aleix ha iniciat sessió a banc.example.com
- Rep el següent missatge de la Berta:

```
Ei Aleix! Mira això:
<img src="http://banc.example.com/nova_transferencia?
compte_origen=Aleix&quantitat=1000&compte_desti=Berta">
```

- El navegador de l'Aleix quan realitzi la petició de la imatge farà la petició que vol la Berta.
- Si es disposa d'una màquina intermitja es pot utilitzar amb diverses finalitats no relacionades amb l'aplicació, es un problema que va més enllà...

Com evitar-ho?

- Filtre que comprova la presencia d'un **token**, excepte per les pàgines d'entrada a l'aplicació
- Afegir el token a **CADA** url excepte les d'entrada
  - Això ho fa sol Tomcat si escrivim les url amb c:url

```
<filter>
   <filter-name>CsrfFilter</filter-name>
   <filter-class>org.apache.catalina.filters.CsrfPreventionFilter</filter-class>
   <init-param>
     <param-name>entryPoints</param-name>
     <param-value>/</param-value>
   </init-param>
 </filter>
```

Conclusions de programació segura

- Pensar en casos d'abús quan pensem en els casos d'ús
- Utilitzar sempre SQL paramètric
- Controlar les opcions de configuració de la nostra aplicació
- Utilitzar seguretat declarativa sempre que puguem
- "Escapar" o codificar quan escrivim un valor com a HTML, JSON, URL...
- Si tenim camps amb HTML, sanititzar amb llista blanca
- Mirar cada paràmetre amb lupa, què acceptem i què no
- Quan tenim un paràmetre que fa referència a alguna cosa... validar!
- I tot això es resumeix en: **desconfieu en el que us arriba!**

Vulnerabilitats

Gestió de les vulnerabilitats

- Es necessari conèixer, emmagatzemar i gestionar les vulnerabilitats que es descobreixen. D'aquesta manera podem:
  - Comprovar ràpidament si es coneix alguna vulnerabilitat per alguna plataforma.
  - Identificar exploits a partir de les vulnerabilitats.

Gestió de les vulnerabilitats – CPE

- Common Platform Enumeration.
- Definits per MITRE (www.mitre.org)
- Nom “estàndard” que s'assigna a un producte software, o sistema.
- Permet identificar-lo de manera única.

CPE - Exemple

- cpe:/a:microsoft:internet_explorer:11:beta
- cpe:/a:microsoft:internet_explorer:10
- cpe:/a:microsoft:internet_explorer:9
- cpe:/a:microsoft:internet_explorer:8

Gestió de les vulnerabilitats – CVE

- Common vulnerabilities and Exposures
- També definit per MITRE.
- Identificador que s'assigna a una vulnerabilitat.

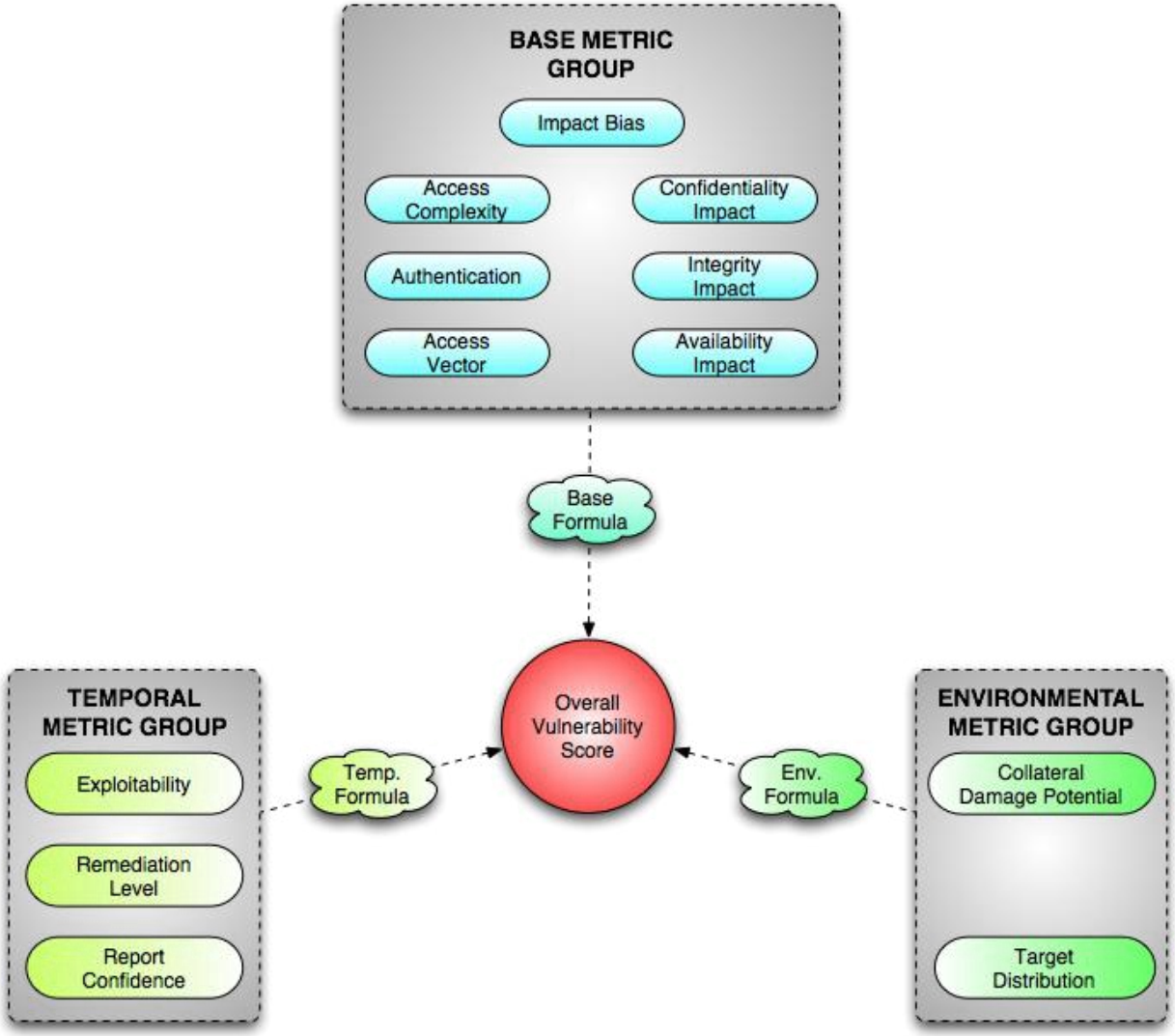
CVE - Exemple

- CVE – CVE-2014-0268
- CPE
  - cpe:/a:microsoft:internet_explorer:11:
  - cpe:/a:microsoft:internet_explorer:10
  - cpe:/a:microsoft:internet_explorer:9
  - cpe:/a:microsoft:internet_explorer:8
- Descripció
  - Hi ha una vulnerabilitat d'elevació de privilegis a Internet Explorer durant la validació de la instal·lació de fitxers locals i durant la creació segura de claus de registre.

Gestió de les vulnerabilitats – CVSS

- Definit l'any 2004 pel conjunt d'equips de resposta a incidents i de seguretat del FIRST.
- Pretén identificar i avaluar les vulnerabilitats.

CVSS - Mètriques



CVSS - Mètriques

- Exploitability Metrics
  - Access Vector (AV)*
    - Local (AV:L)
    - Adjacent Network (AV:A)
    - Network (AV:N)
  - Access Complexity (AC)*
    - High (AC:H)
    - Medium (AC:M)
    - Low (AC:L)
  - Authentication (Au)*
    - Multiple (Au:M)
    - Single (Au:S)
    - None (Au:N)
- Impact Metrics
  - Confidentiality Impact (C)*
    - None (C:N)
    - Partial (C:P)
    - Complete (C:C)
  - Integrity Impact (I)*
    - None (I:N)
    - Partial (I:P)
    - Complete (I:C)
  - Availability Impact (A)*
    - None (A:N)
    - Partial (A:P)
    - Complete (A:C)

* - All base metrics are required to generate a base score.

CVSS- Exemple

- CVSS Severity (version 2.0):
  - CVSS v2 Base Score: 4.3 (MEDIUM) **(AV:N/AC:M/Au:N/C:N/I:P/A:N)**
  - Impact Subscore: 2.9
  - Exploitability Subscore: 8.6

- CVSS Version 2 Metrics:
  - Access Vector: Network exploitable; Victim must voluntarily interact with attack mechanism
  - Access Complexity: Medium
  - Authentication: Not required to exploit
  - Impact Type: Allows unauthorized modification

Gestió de les vulnerabilitats – NVD

- El NIST disposa d'una base de dades amb totes les vulnerabilitats que han aparegut des de 2002.
- Permet la consulta directa o bé descarregar totes les dades en format XML.
- S'actualitza diàriament.
  - Afegeix els CVSS a posteriori, entre 8 i 48 hores després normalment.

NVD - Exemple

- L'entrada a la NVD d'una vulnerabilitat

CVE-2014-0268

- https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0268

Gestió de les vulnerabilitats

- De totes maneres tota aquesta informació és molt difícil d'aplicar a la nostra organització si no disposem d'un **sistema d'inventari** adequat que ens permeti saber:
  - Quins son els nostres actius?
  - On es troben físicament?
  - Quines funcions desenvolupen?
  - En quin estat es troben?

0-days

- Les vulnerabilitats 0-day son aquelles que a disposició de l comunitat (o no) sense que el fabricant del producte n'estigui al corrent:

  No hi ha "parche"

- Es tracta d'una divulgació de tots els detalls del problema de seguretat que permeten la creació d'exploits durant la finestra de temps disponible fins que el fabricant soluciona el problema.

  0-day -> exploits -> CRITICITAT ALTA

Tècniques d'anàlisi de codi

Tècniques de comprovació

- Inspeccions y revisions manuals
- Modelat d'amenaces
- **Revisió de codi**
- Proves d'intrusió

Anàlisi del codi

- Es bona idea realitzar analaisi del codi que escrivim
  - Millor si el fa una altre persona
    - Extreme programing!
  - O un procés...

- Es pot analitzar el codi de manera **estàtica** (caixa blanca) o **dinàmica** (en execució)

Anàlisi estàtic

- Permet comprovar:
  - Que es segueixen bones pràctiques de programació
  - Vulnerabilitats típiques
- Se li poden escapar errors de disseny propis del llenguatge o l'aplicació en si mateixa.
- Segons la fase del SDLC
  - Implementació
  - Testing

Eines d'anàlisi estàtic

- PMD
  - Open source
  - Java, Javascript, PLSQL, Apache Velocity, XML y XSL
  - Detecta duplicació de codi
  - Integració amb diversos IDEs (Maven, Eclipse, NetBeans, JBuilder, JDeveloper o IntelliJ IDE).
- SonarQube
  - Open source
  - Ofereix resum de la qualitat del codi segons diferents mètriques
- Coverity
  - Analitzador estàtic de codi Java, C/C++, C# o JavaScript
  - Gratuït per projectes open source

Eines d'anàlisi estàtic

- FxCop
  - de Microsoft pel .NET Framework basat en les guies de disseny de .NET
- FindBugs
  - Busca patrons d'error a codi Java
  - Pluguin pels principals IDEs (Eclipse, Maven, NetBeans, etc...)
- FlawFinder
  - Per C/C++
- RIPS
  - Per PHP
- Bandit
  - Per Python

https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

Anàlisi dinàmic

- Permet comprova el comportament de l'aplicació en un entorn d'execució real
- Poden aparèixer errors no detectats a l'anàlisi estàtic

Eines d'anàlisi dinàmic

- El debugger de tota la vida

- Application Verifier
  - Eina de Microsoft
  - Ara inclosa a les debug tools
  - Monitoritza l'ús de recursos del software i proposa canvis
- ProcessMonitor
  - Eina de sysinternals
  - Monitoritzem nosaltres mateixos que fa l'aplicació

Provar aviat i sovint...

- Metodologies àgils!
- TDD
  - Amb tot el que implica
    - Unit Test
      - Per diferents llenguatges (Junit (java), python...)
    - Jenkins
    - Etc...
- http://agiledata.org/essays/tdd.html

Auditories

Auditories

- Que es una auditoria?
- L'**auditoria** consisteix en la revisió sistemàtica d'una activitat o d'una situació per avaluar-ne el compliment de les regles o criteris objectius als que les mateixes han de ser sotmeses, segons la definició donada per la Real Academia de la Lengua Española.

- En el nostre cas consisteix en **revisar** la seguretat y la integritat d'aplicacions o servidors y **avaluar** les possibles vulnerabilitats trobades.

Auditories

- Tècniques d'auditoria:
  - Inspeccions y revisions manuals
  - Modelat d'amenaces
  - Revisió de codi
  - **Proves d'intrusió**

Auditories – Proves d'intrusió

- Avantatges
  - Pot ser ràpid i barat.
  - Requereixen un coneixement relativament menor que una revisió del codi font.
  - Comproven el codi que realment està exposat.
- Inconvenients
  - Massa tard des del punt de vista del SDLC
  - Proves només dels impactes frontals

I això es legal?

- Segons l'article 197 bis del codi penal en referència a l'accés no autoritzat a sistemes informàtics:
  - *1. El que por cualquier medio o procedimiento, vulnerando las medidas de seguridad establecidas para impedirlo, y **sin estar debidamente autorizado**, acceda o facilite a otro el acceso al conjunto o una parte de un sistema de información o se mantenga en él en contra de la voluntad de quien tenga el legítimo derecho a excluirlo, será castigado con pena de prisión de seis meses a dos años.*
  - *2. El que mediante la utilización de artificios o instrumentos técnicos, y **sin estar debidamente autorizado**, intercepte transmisiones no públicas de datos informáticos que se produzcan desde, hacia o dentro de un sistema de información, incluidas las emisiones electromagnéticas de los mismos, será castigado con una pena de prisión de tres meses a dos años o multa de tres a doce meses.*

Auditories – Proves d'intrusió

- Normalment segueixen un procés de l'estil:
  - Information gathering
  - Escaneig / enumeració
  - Anàlisi
    - Manual
    - Automatitzat
  - Explotació / obtenció d'evidències

Information Gathering

Information Gathering

- A l'hora de fer una auditoria es important obtenir el màxim d'informació possible sobre els servidors o els serveis auditats.
- No es tracta només d'informar-nos, sinó també de buscar "coneixements indeguts" com poden ser l'estructura interna de la xarxa, arxius que haurien d'estar ocults...
- Es a dir, volem trobar informació que no hauríem (o haurien) de trobar.

Information Gathering - DNS

- Mitjançant peticions DNS als servidors de l'objectiu es possible obtenir informació diversa:
  - Nombre de servidors de correu/DNS (redundància)
  - Serveis externs (caching, correu)
  - Subdominis
  - Servidors virtualitzats

- Hi ha diverses eines que permeten realitzar aquestes peticions:
  - nslookup
  - dig
  - fierce

DNS - dig

- Podem demanar pels registres habituals (A, NS, MX)
- Podem fer resolució inversa amb -x
  - Útil per descobrir noms alternatius o aplicacions web
- També podem indicar el servidor DNS que volem utilitzar amb "@[server]"
- Podem simplificar la resposta amb el flag "+short"

Information Gathering – Google Hacking

- Google hacking
  - Es una tècnica que utilitza el buscador de google amb paràmetres específics que permeten filtrar per url o certs strings que ens poden permetre descobrir fitxers interns de servidors que hagis estat indexats.

Information Gathering – Google Hacking

La cerca base

- Busquem paraules o frases intentant “tenir sort”
  - Moltíssima informació que no ens interessa
- Segur que tinc sort
  - La primera pàgina del resultat de la cerca
    - La més votada, la primera del ranking
- Cerca de paraules
  - Omet preposicions, articles...
    - Cerca “El museu de la ciència a Barcelona”
    - Cerca “ museu ciència Barcelona“
      - Es el mateix que “ museu + ciència + barcelona“
  - No distingeix entre majúscules y minúscules
    - Excepte OR
  - Correcció dels termes a buscar

Information Gathering – Google Hacking

- Per fer una cerca més precisa hem d'utilitzar els paràmetres que ens ofereix el cercador.
- Utilització de les cometes
  - Cerca "Universitat Politècnica de Catalunya"
- Utilització de l'operador "-"
  - Omissió de paraules " "Universitat Politècnica de Catalunya" –Escola "
- Utilització de OR
  - Cerca entre diversos termes
  - Resultat amb totes les pàgines on hi aparegui qualsevol d'ells.
- Avaluació de paràmetres d'esquerra a dreta.

Information Gathering – Google Hacking

- Existència d'operadores especials
  - operador:terme
  - Sense espais!
  - L'operador en minúscules!
- El terme pot ser:
  - Una paraula
  - Una frase (entre cometes)
- Existeixen operadors que s'han d'utilitzar sols
  - Començenamb la paraula reservada ALL
- No tots serveixen per a tot
  - Seccions
- Si es produeix un error de sintaxi s'entén com una terme més a buscar

Google Hacking – Prevenció

- Ser conscients de la informació que penjem
  - Configuracions por defecte
  - Fitxers de proves
- Ús de metatags
  - “<meta name=’GOOGLEBOT’ content=’NOARCHIVE’ />”
  - Noarchive: no es guarda a la caché
  - Llistat de robots: http://www.user-agents.org
- Utilització del fitxer robots.txt
  - Format del fitxer:
    - User-agent: *
    - Disallow: llistat de directoris
- Però....
  - filetype:txt inurl:robots.txt

Google Hacking – Automatització

- Eines d'automatització de google hacking
  - FoundStone Sitedigger
  - Apollo
  - Athena
  - Wikto
- GHDB. Google Hacking Database
  - Col·lecció de tècniques de Google Hacking
  - http://www.exploit-db.com/google-dorks/

Information gathering - Metadades

- Molts documents contenen informació “extra”
- Les metadades proporcionen informació sobre:
  - Usuaris
  - Localització (imatges)
  - IPs
  - Software
- Eines
  - stat
  - pdfinfo
  - file
  - https://metashieldanalyzer.elevenpaths.com/

Metadades - FOCA

- Fingerprinting Organizations with Collected Archives
- Combina i automatitza moltes tècniques
- Permet extreure i **organitzar** informació dels fitxers trobats.

Information gathering – Altres tècniques

- Deep Web & Pastebin
  - Sempre es bona idea buscar aparicions de l'entitat que estem auditant a la Deep Web (amb compte) i a Pastebin, ja que molts grups de… "moralitat dubtosa" acostumen a deixar allà la informació que obtenen.

Escaneig de xarxa i enumeració

Escaneig de xarxa i enumeració

- Una de les primeres coses que necessitem saber és quins serveis de l'entitat que auditem es troben "al descobert", es a dir, accessibles des de fora.
- Volem realitzar un escaneig per detectar quines Ips i ports estan oberts i quins serveix s'ofereixen a través seu.
- Hi ha diverses eines útils per automatitzar aquesta tasca.

Escaneig de xarxa i enumeració

- nmap
  - Eina invocable per línia de comandes de Linux molt potent per realitzar escàners de xarxa i ports.
    - nmap -v -A scanme.nmap.org
      - Scan detallat amb detecció de SO, versions, scripts i traceroutes.
    - nmap –F scanme.nmap.org
      - Scan ràpid que només detecta els ports oberts o filtrats.

Escaneig de xarxa i enumeració - nmap

- **Obert**. Indica que l'aplicació que ofereix el servei espera connexions o paquets al port
- **Tancat**. Els ports tancats no tenen cap aplicació escoltant a través seu però es podrien obrir en qualsevol moment.
- **Filtrat**. Indica que un firewall o algun altre obstacle a la xarxa està bloquejant l'accés al port i per tant nmap no el pot classificar com a obert o tancat.
- **No filtrat**. Els ports classificats com a no filtrats son aquells que responen als sondejos de nmap però pels quals l'eina no es capaç d'assignar un estat (obert o tancat). Només amb escàner ACK.

Escaneig de xarxa i enumeració - nmap

- Nmap informa de las combinacions d'estat **open|filtered** i **closed|filtered** quan no pot determinar en quin dels dos estats es troba un port.

- La taula de ports també pot incloure detalls de la versió de l'aplicació quan s'ha sol·licitat detecció de versions.

Escaneig de xarxa i enumeració - nmap

- Entrada:
  - scanme.nmap.org
  - microsoft.com/24
  - 192.168.0.1
  - 10.0.0-255.1-254
  - iL <fitxer_entrada>: Llegeix una llista de sistemes/xarxes del fitxer.

- Descobriment:
  - **sP**: Sondeig ping. Només determina si l'objectiu està "viu".
  - **P0**: Assumeix que tots els objectius estan "vius".
  - **-n**: no fa resolució DNS.

Escaneig de xarxa i enumeració - nmap

- Detecció del sistema operatiu:
  - **-O**
- Sondejar ports oberts per obtenir informació del servei/versió
  - **-V**
- Velocitat
  - **-T[0-5]**

Exemples nmap

- nmap scanme.nmap.org

```
Starting Nmap 6.47 ( http://nmap.org ) at 2015-11-09 13:55 CET
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.17s latency).
Not shown: 991 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
25/tcp    filtered smtp
80/tcp    open     http
135/tcp   filtered msrpc
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
593/tcp   filtered http-rpc-epmap
9929/tcp  open     nping-echo
31337/tcp open     Elite

Nmap done: 1 IP address (1 host up) scanned in 16.00 seconds
```

Exemples nmap

- `nmap -sP scanme.nmap.org`

```
Starting Nmap 6.47 ( http://nmap.org ) at 2015-11-09 16:26
CET
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.17s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds
```

Exemples nmap

- nmap -O scanme.nmap.org

```
Starting Nmap 6.47 ( http://nmap.org ) at 2015-11-09 16:31 CET
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.17s latency).
Not shown: 990 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
25/tcp    filtered smtp
53/tcp    filtered domain
80/tcp    open     http
135/tcp   filtered msrpc
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
593/tcp   filtered http-rpc-epmap
9929/tcp  open     nping-echo
31337/tcp open     Elite
Aggressive OS guesses: Linux 3.11 - 3.13 (95%), Linux 3.2 - 3.8 (95%), IPFire firewall 2.11
(Linux 2.6.32) (94%), Linux 2.6.32 (94%), Linux 3.1 - 3.2 (94%), DD-WRT v24-sp1 (Linux 2.4)
(93%), Linux 2.6.32 - 2.6.39 (92%), IGEL UD3 thin client (Linux 2.6) (92%), Linksys WRV200
wireless broadband router (92%), DD-WRT v24-sp2 (Linux 2.4.36) (92%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 18 hops

OS detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.42 seconds
```

Anàlisi manual

Anàlisi manual

- No podem trobar totes les vulnerabilitats existents mitjançant analitzadors automàtics ja que:
  - Generen falsos positius
  - No troben totes les vulnerabilitats possibles.
  - De vegades no distingeixen correctament situacions que a un expert li resulten obvies.

Anàlisi manual – OWASP

- OWASP (Open Web Application Security Project) es una comunitat oberta dedicada a habilitar les organitzacions per a desenvolupar, comprar y mantenir aplicacions confiables.

- Totes les eines, documents, fòrums i capítols d'OWASP son gratuïts per a qualsevol persona interessada en millorar la seguretat de les aplicacions.

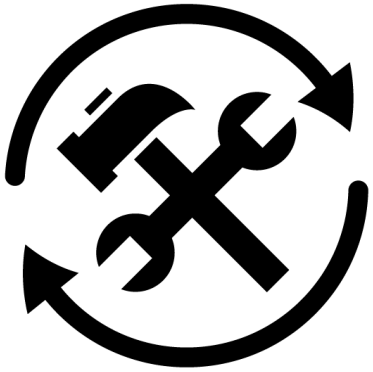
- https://www.owasp.org

Anàlisi manual

- Sol requerir coneixements específics del que s'està analitzant.
  - Information gathering!

- La intuïció i la creativitat de l'auditor poden jugar un paper important en alguns casos.

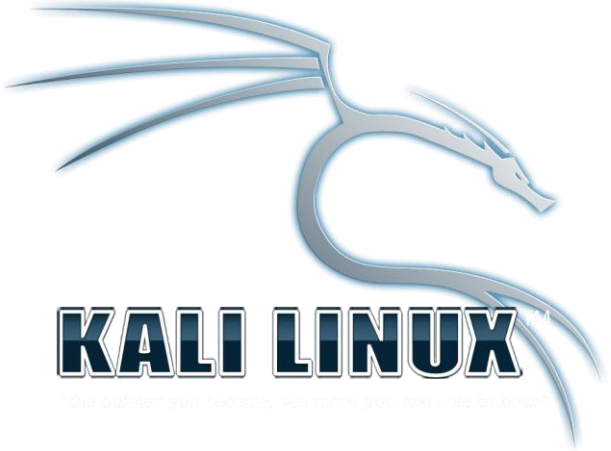
- La única manera d'aprendre'n es **practicant.**



Anàlisi automatitzat

Anàlisi automatitzat

- Hi ha moltíssimes eines disponibles que automatitzen alguns dels atacs més comuns.
- Algunes han arribat a un nivell de sofisticació considerable.
- A la distribució de Linux Kali (www.kali.org) disposem de moltes d'aquestes eines ordenades segons la seva finalitat.



Anàlisi automatitzat - WFuzz

- Wfuzz
  - Eina molt coneguda que permet atacar serveis mitjançant fuzzing i que és altament configurable.
  - Exemple d'invocació:
    - Wfuzz.py –c –z file,commons.txt –hc 404 –o html http://www.site.com/FUZZ 2> res.html

Anàlisi automatitzat - sqlmap

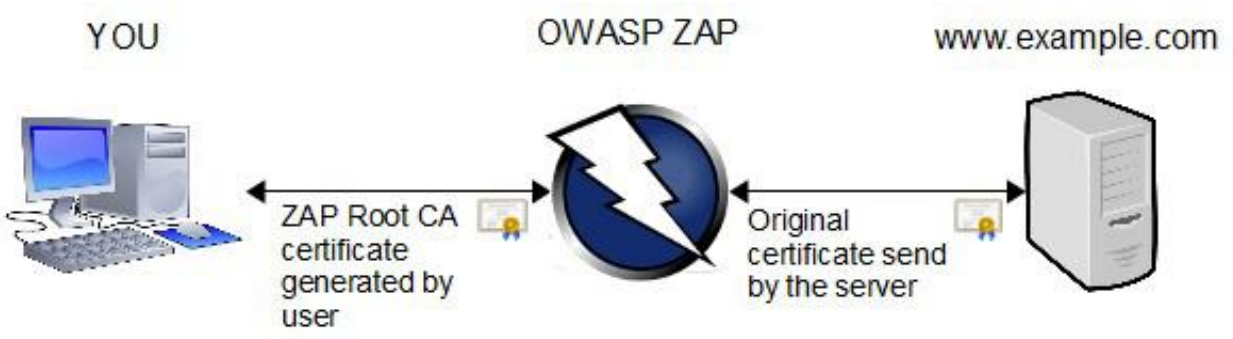
- sqlmap
  - Eina molt potent per realitzar tests de penetració a bases de dades.
  - Automatitza la detecció i explotació de moltes vulnerabilitats SQLinjection.
  - Escrita en Python
  - El codi es troba disponible a Github
    - https://github.com/sqlmapproject/sqlmap

Anàlisi automatitzat - sqlmap

- Alguns dels modificadors més utilitzats:
  - **-u URL**: Defineix la pàgina objectiu.
  - **-p**: Permet especificar el paràmetre vulnerable.
  - **--dbs**: Mostra el nom de totes les bases de dades.
  - **-D**: Selecciona la base de dades on es realitzaran les consultes.
  - **--tables**: Mostra el nom de les taules de la base de dades seleccionada.
  - **-T:** Selecciona la taula on es realitzaran les consultes.
  - **--columns**: Mostra el nom de les columnes de la taula seleccionada.
  - **--dump**: Extreu les dades de la taula seleccionada a un fitxer csv.
  - **--dbms**: Selecciona el motor de base de dades que utilitza la consulta (SQL, MySQL, ORACLE...)

Anàlisi automatitzat – OWASP ZAP

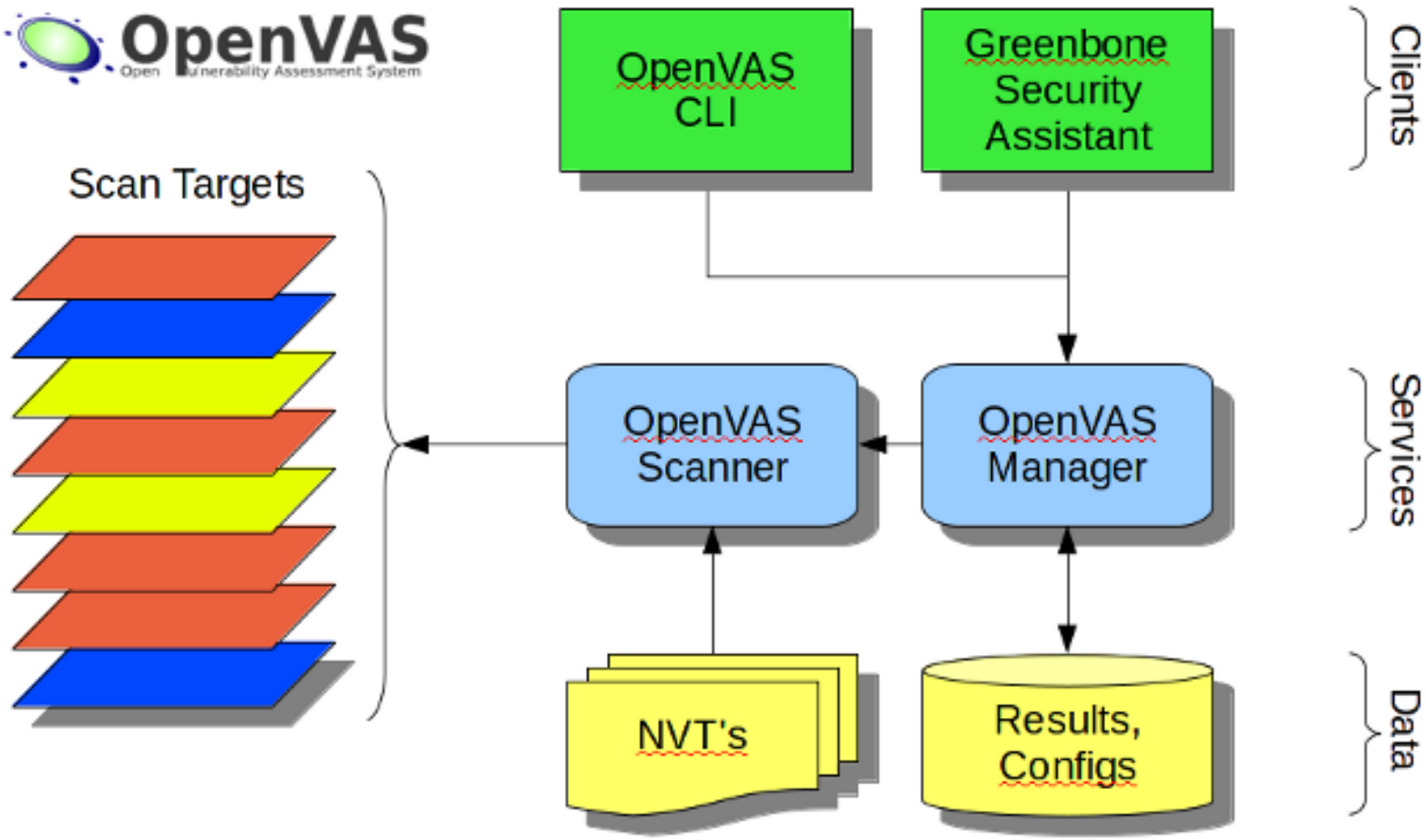
- OWASP Zed Attack Proxy
- Permet automatitzar l'anàlisi de vulnerabilitats en aplicacions web
  - Encara que també permet proves manuals
    - En aquest curs utilitzaré Burpsuite
- Fa de proxy entre el navegador i l'aplicació a auditar
  - S'han d'importar els certficats



Anàlisi automatitzat- OpenVAS

- Eina que permet ***escanejar*** una infraestructura, ***detectar*** els serveis que ofereix i ***automatitzar*** la seva ***explotació***.
- Útil per assegurar que l'entorn en el que s'executa l'aplicació també es segur.

OpenVAS – Arquitectura



Explotació de vulnerabilitats

Explotació de vulnerabilitats

- Un cop detectades les (possibles) vulnerabilitats tan sols ens queda explotar-les.
- Hem d'anar amb compte per diverses raons:
  - Hem de complir l'acord que tinguem amb el responsable del sistema.
  - Tot i tenir permís també podem "tombar" involuntàriament el sistema.

Explotació de vulnerabilitats

- El primer pas per explotar una vulnerabilitat és mirar si ja existeix algun exploit conegut.
- Algunes son relativament senzilles de comprovar:
  - Un fitxer que hauria d'estar ocult és accessible.
  - Podem injectar SQL en un camp concret.
- N'hi ha d'altres que necessiten fer ús d'exploit més complexes i també hem de mirar si ja existeixen:
  - Podem buscar-los a exploit-db
    - https://www.exploit-db.com/

  - Directament podem utilitzar metasploit.

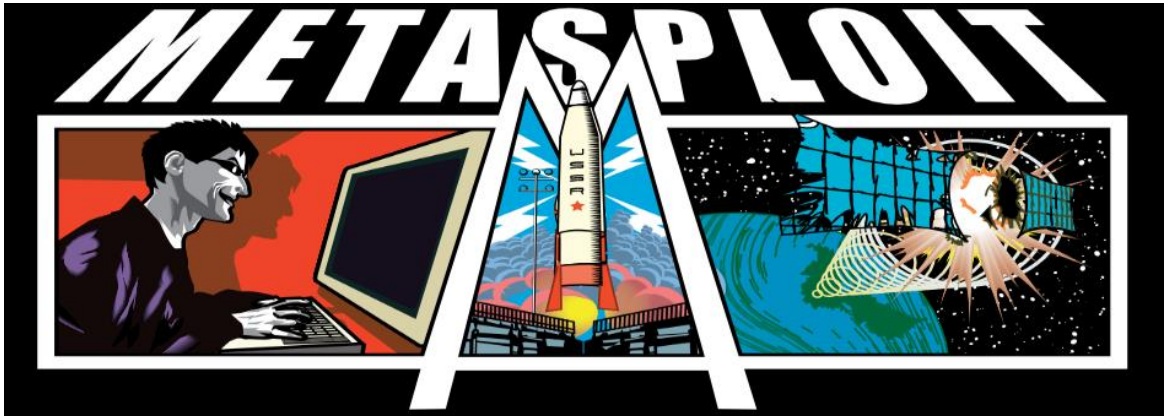
Explotació de vulnerabilitats

- Moltes vegades trobem l'exploit just per la vulnerabilitat que volem explotar.

- D'altres voldrem modificar el codi d'un exploit ja existent.

- En última instància pot ser que haguem d'escriure el nostre propi exploit.

Explotació de vulnerabilitats - Metasploit



- Es tracta d'un Framework amb diverses eines que permeten:
  - Desenvolupar exploits.
  - Executar exploits contra màquines locals o remotes.
  - Escanejar màquines cercant vulnerabilitats.
  - Recol·lectar informació sobre vulnerabilitats.
  - **Manté el context d'explotació.**

Explotació de vulnerabilitats - Metasploit

- El framework també conté un conjunt de binaris del tipus “msf” que permeten, entre d’altres:
  - Invocar una línia de comendes per interactuar amb Metasploit.
  - Executar una interfície gràfica per interactuar amb Metasploit.
  - Generació de payloads.
  - Ofuscació dels payloads mitjançant encoders.
  - Anàlisi de binaris.

Explotació de vulnerabilitats - Metasploit