

TAMDA Zohir
BTS SIO 2 option SLAM
Bloc 3 - Cybersécurité

Le 24 septembre 2024



Enumération des logins

Sommaire

Mode non-sécurisé.....	3
Vérification d'un utilisateur inexistant.....	3
Vérification d'un utilisateur existant.....	6
L'attaque brute force.....	7
Mode sécurisé.....	10

Mode non-sécurisé

Le niveau de sécurité est définie à "0", ce qui signifie qu'il n'y a aucune vérification à faire côté client ni côté serveur. L'extension "Wsdl" va être utilisée pour intercepter des requêtes WSDL.

OWASP Mutillidae II: Keep Calm and Pwn On

Version: 2.11.15 Security Level: 0 (Hosed) Hints: Enabled Not Logged In

Burp extensions

Extensions let you customize Burp's behavior using your own or third-party Java code.

Add	Loaded	Type	Name
Remove	✓	Java	Wsdl

Vérification d'un utilisateur inexistant

L'URL "<http://172.16.61.5/webservices/soap/ws-user-account.php?wsdl>" a été tapée dans le navigateur pour permettre de récupérer des opérations dans le WSDL, l'interception est activé du côté de BurpSuite.

Pour récupérer les opérations, on peut parser le WSDL en cliquant sur "Parse WSDL" à partir du menu.

Request 172.16.61.5 GET http://172.16.61.5/webservices/soap/ws-user-account.php?wsdl

- Request Scan GET http://detectportal.firefox.com/
- Request Send to Intruder Ctrl+I GET http://detectportal.firefox.com/
- Request Send to Repeater Ctrl+R GET http://detectportal.firefox.com/
- Request Send to Sequencer GET http://detectportal.firefox.com/
- Request Send to Comparer GET http://detectportal.firefox.com/
- Request Send to Decoder
- Request Send to Organizer Ctrl+O
- Request Insert Collaborator payload
- Request Request in browser >
- Request Extensions > Wsdl > Parse WSDL

ws-user-account *

Operation

getUser
createUser
updateUser
deleteUser

```
<urn:getUser soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <username xsi:type="xsd:string">
    gero et
  </username>
</urn:getUser>
```

Résultat : on obtient 4 opérations :

- **getUser** qui sert à récupérer les informations de l'utilisateur.
- **createUser** qui sert à créer un utilisateur.
- **updateUser** qui sert à mettre à jour les informations de l'utilisateur.
- **deleteUser** qui sert à supprimer l'utilisateur.

En dessous de cette liste des opérations, on obtient un code sous format XML qui contient la balise “**username**” et cette dernière contient une valeur qui correspond au nom de l'utilisateur “**gero et**” par exemple.

Pour cela, faire un clic droit sur le code et cliquer sur “Send to Repeater” (*Envoyer au répéteur*).

Pretty	Raw	Hex
1 POST /webservices	Scan	Gecko/20100101 Firefox/
2 User-Agent: Mozilla/5.0	Send to Intruder	image/avif,image/webp
3 Accept: text/html	Send to Repeater	
4 Accept-Language:		
5 Accept-Encoding:		
6 Connection: keep-alive		

```

Request
Pretty Raw Hex
1 POST /webservices/soap/ws-user-account.php HTTP/1.1
2 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:120.0) Gecko/20100101
3 Firefox/120.0
4 Accept:
5 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: fr,fr-TR;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Connection: keep-alive
9 Upgrade-Insecure-Requests: 1
10 Priority: u=0, i
11 SOAPAction: urn:ws-user-account#getUser
12 Content-Type: text/xml;charset=UTF-8
13 Host: 172.16.61.5
14 Content-Length: 452
15
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="urn:ws-user-account">
<SOAP-ENV:Header>
<ns1:getUser>
<ns1:Body>
<ns1:getUserResponse xmlns:xsi="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="xsd:string">
    gero et
</ns1:getUserResponse>
</ns1:Body>
</ns1:getUser>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Tue, 01 Oct 2024 12:10:56 GMT
3 Server: Apache/2.4.61 (Debian)
4 X-Powered-By: PHP/8.0.310
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate
7 Pragma: no-cache
8 X-SOAP-Server: MuSOAP/0.9.11 (1.123)
9 Usr: Accept-Encoding
10 Content-Length: 559
11 Keep-Alive: timeout=5, max=100
12 Connection: Keep-Alive
13 Content-Type: text/xml, charset=ISO-8859-1
14
15
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns1="urn:ws-user-account">
<SOAP-ENV:Body>
<ns1:getUserResponse xmlns:xsi="urn:ws-user-account">
    <xsi:type="xsd:string">
        <accounts message="User gero et does not exist"></accounts>
    </xsi:type>
</ns1:getUserResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Le répéteur nous permet de modifier le code XML afin d'écrire le nom d'utilisateur et ainsi vérifier s'il existe ou pas.

Dans le cas où l'utilisateur "**gero et**" n'existe pas dans la base de données, la réponse sous format XML est claire: l'utilisateur gero et n'existe pas.

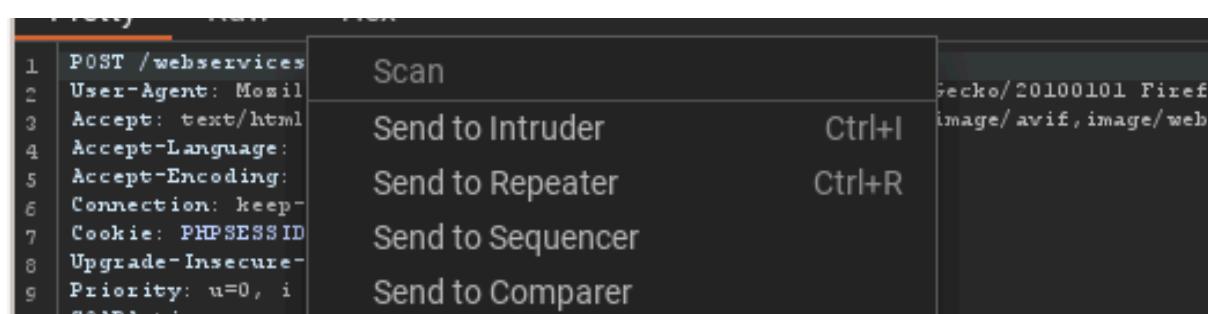
```

<return xsi:type="xsd:string">
    <accounts message="User gero et does not exist"></accounts>
</return>

```

La dernière étape qui est indispensable consiste à comparer les données pour chaque requête XML qu'on effectue. Par exemple, on peut comparer la réponse avec un utilisateur inexistant et l'autre avec l'utilisateur existant. C'est le rôle du comparateur (*Comparer*).

Pour cela, faire un clic-droit sur la réponse du code XML et cliquer sur "Send to Comparer" (Envoyer au comparateur).



Les deux requêtes vont être comparées dans la prochaine page.

Vérification d'un utilisateur existant

On souhaite vérifier que l'utilisateur nommé "**utilisateur1**" existe bien dans la base de données. Les procédures pour récupérer les opérations WSDL, mettre la requête dans le répéteur ainsi que mettre la réponse dans le comparateur sont les mêmes.

Si on veut envoyer la requête, cliquer sur le bouton "Send".

<pre><urn:getUser soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"> <username xsi:type="xsd:string"> utilisateur1 </username> </urn:getUser></pre>	<pre><accounts message="Results for utilisateur1"> <account> <username> utilisateur1 </username> <signature> une dÃ©dicace </signature> </account> </accounts></pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Requête

Réponse

Résultat : on obtient le nom d'utilisateur ainsi que sa signature "Une dédicace". Cette image démontre que l'utilisateur nommé "**utilisateur1**" existe. Côté comparateur, il y a deux requêtes. Avec ça, on peut les comparer grâce au bouton "**Words**" qui compare chaque mot pour ces deux requêtes.

The screenshot shows the 'Comparer' (Compare) tool interface. It has two main sections: 'Select item 1:' and 'Select item 2:'. Both sections display tables with columns for '#', Length, and Data. The Data column contains identical XML snippets. To the right of each table are buttons for Paste, Load, Remove, and Clear. Below the tables are 'Compare ...', 'Words', and 'Bytes' buttons.

#	Length	Data
17	1058	HTTP/1.1 200 OKDate: Tue, 01 Oct 2024 13:41:51 GMTServer: Apache/2.4.61 (Debian)X-Powered-By: PH...
18	965	HTTP/1.1 200 OKDate: Tue, 01 Oct 2024 13:42:11 GMTServer: Apache/2.4.61 (Debian)X-Powered-By: PH...

#	Length	Data
17	1058	HTTP/1.1 200 OKDate: Tue, 01 Oct 2024 13:41:51 GMTServer: Apache/2.4.61 (Debian)X-Powered-By: PH...
18	965	HTTP/1.1 200 OKDate: Tue, 01 Oct 2024 13:42:11 GMTServer: Apache/2.4.61 (Debian)X-Powered-By: PH...

A gauche, il y a la réponse sous format XML contenant le nom d'utilisateur "**utilisateur1**" ainsi que la signature "**Une dédicace**" surlignés en orange; tandis qu'à droite, il y a également la réponse sous le même format mais ce message qui est surligné en orange indique que l'utilisateur "**gero et**" n'existe pas dans la base de données.

```

Length: 1,058
Length: 965
<!--><account><username>utilisateur1</username><signature>une dÃ©dicace</signature>
<!--><accounts message="User gero et does not exist" /></return></ns1:getUs

```

L'attaque brute force

Pour réaliser cette attaque, il est nécessaire de créer un fichier texte contenant un ensemble de noms d'utilisateurs.

```

GNU nano 7.2
administrateur
admin
utilisateur
utilisateur1
Arithmatek
BetterEm
Bramsø
CVE
CWE
Cavalcanti

```

Pour charger ces utilisateurs, aller dans Intruder > Payloads et cliquer sur "Load...". Puis, charger le fichier texte contenant les noms d'utilisateurs, "dictionnaire_users.txt" par exemple

The screenshot shows the ZAP interface with the 'Proxy' tab selected. In the 'Payloads' section, a file selection dialog is open, showing a list of files in the 'Bureau' folder. The file 'dictionnaire_users.txt' is selected. Other files listed include TP Java, TP JS, Eclipse IDE.desktop, settings.desktop, and userfiles.desktop. The 'Nom fichier:' field is set to 'dictionnaire_users.txt' and the 'Type de fichier:' field is set to 'All files'.

Puis, on ne souhaite afficher que les utilisateurs existants dans la base de données; il faut les filtrer. Pour cela, aller dans l'onglet "Settings", se rendre dans la section "Grep - Extract" et cliquer sur le bouton "Add" (Ajouter). Puis, écrire dans le 1er champ de texte "Results for". Ce filtre permet de récupérer des données contenant le texte "Results for", s'il y en a pas, elles ne seront pas prises en compte.

The screenshot shows the OWASp ZAP interface with the 'Intruder' tab selected. In the 'Grep - Extract' section, the 'Extract the following items' checkbox is checked. Under 'Define start and end', the 'Start after expression' radio button is selected with the value 'Results for'. The 'Start at offset:' field is empty.

Après cela, dans l'onglet "Positions", ajouter l'URL concernée et coller le code XML.

The screenshot shows the OWASp ZAP interface with the 'Positions' tab selected. The 'Target' field contains the URL 'http://172.16.61.5/webservices/soap/ws-user-account.php?wsdl'. The 'Attack type' dropdown is set to 'Sniper'. On the right, there is a code editor window displaying a SOAP request payload:

```

POST /webservices/soap/ws-user-account.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:120.0) Gecko/20100101 Firefox/120.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: PHPSESSID=4jw8ijfm3nlbqjmuosnucv; showhints=1
Upgrade-Insecure-Requests: 1
Priority: -m=, i
SOAPAction: urn:ws-user-account#getUser
Content-Type: application/xml; charset=UTF-8
Host: 172.16.61.5
Content-Length: 455
Content-Type: application/xop+xml; charset=UTF-8; action="http://www.w3.org/2001/XMLSchema-instance"
Content-Transfer-Encoding: binary
Content-Location: http://www.w3.org/2001/XMLSchema-instance
Content-Format: http://schemas.xmlsoap.org/soap/envelope/
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header/>
<soapenv:Body>
<urn:getUser soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<username xsi:type="xsd:string">${user}</username>
</urn:getUser>
</soapenv:Body>
</soapenv:Envelope>

```

Après ça, sélectionner "**gero et**" dans ce code et cliquer sur le bouton "**Add \$**". Cela permet de remplacer "**gero et**" par chaque nom de l'utilisateur, ce qui est surligné en violet. Par la suite, on peut effectuer cette attaque en cliquant sur le bouton "Start attack".

6. Intruder attack of http://172.16.61.5/webservices/soap/ws-user-account.php?wsdl									
Attack Save Help ?									
Results	Positions	Payloads	Resource pool	Settings					
Intruder attack results filter: Showing all items									
Request ^	Payload	Status code	Response received	Error	Timeout	Length	Results for	Comment	
0		200	38			965			
1	administrateur	200	21			971			
2	admin	200	22			1040	admin"<account><username>		
3	utilisateur	200	13			968			
4	utilisateur1	200	18			1058	utilisateur1"<account><username>		
5	Arithmatey	200	17			967			

Résultat, on voit que 2 informations sont présentes dans la colonne "Results for", ce qui signifie que les utilisateurs "**admin**" et "**utilisateur1**" existent dans la base de données; tandis que d'autres qui n'ont aucune information dans cette colonne n'existent pas.

```

Length: 1,040          Length: 968
Text Hex Text Hex
= Results for admin"<account><username>admin</username><signature>g0|r0ot?</signature></account>
return xsi:type="xsd:xml"><accounts message="User utilisateur does not exist" /></return></ns1:getUserRespo

```

Grâce au comparateur, on peut voir en orange qu'avec le nom d'utilisateur et la signature présent dans cette ligne, l'attaquant a pu avoir ces informations pour pirater un compte utilisateur.

Mode sécurisé

Le niveau de sécurité est défini à 5. Les procédures sont identiques pour réaliser l'attaque brute force.



The screenshot shows a 'Word compare of #2 and #3 (7 differences)' interface. The left pane has a length of 1,050 and contains the XML fragment: <je="Results for admin"><account><username>admin</username><signature>g01 j001?</signature>. The right pane has a length of 1,073 and contains the XML fragment: <ige="Results for utilisateur1"><account><username>utilisateur1</username><signature>une j8#xc3:. Both panes have a 'Text' radio button selected.

Les informations, à savoir le nom d'utilisateur ainsi que la signature peuvent toujours être exploitable mais on constate qu'au niveau de la signature, certains caractères spéciaux sont codés, ce qui fait que l'on obtient pas en clair.

```
function doXMLEncodeQueryResults($pUsername, $pQueryResult, $pEncodeOutput) {
    global $Encoder;

    $lResults = "<accounts message=\"Results for ($pUsername)\">";
    $lUsername = "";
    $lSignature = "";

    while($row = $pQueryResult->fetch_object()){

        $pEncodeOutput?&$lSignature = $lUsername = $Encoder->encodeForHTML($row->username):$lUsername = $row->username;

        if(isset($row->mysignature)){
            $pEncodeOutput?&$lSignature = $Encoder->encodeForHTML($row->mysignature):$lSignature = $row->mysignature;
        } // end if

        $lResults.= "<account>";
        $lResults.= "<username>{$lUsername}</username>";
        if(isset($row->mysignature)){$lResults.= "<signature>{$lSignature}</signature>";}
        $lResults.= "</account>";

    } // end while

    $lResults.= "</accounts>";
```

```
    return $lResults;
```

```
} //end function doXMLEncodeQueryResults
```

Après analyse du code dans le fichier "**ws-user-account.php**", la fonction **doXMLEncodeQueryResults()** code les résultats provenant des requêtes SQL avant le l'envoi de la réponse sous format XML. Grâce à la variable globale **\$Encoder**, on peut coder les résultats en appelant la fonction **encodeForHTML()** à partir de cette variable. Cette fonction est utilisée pour coder le nom d'utilisateur ainsi que la signature.