

- أمور يجب عليك معرفتها مُسبقًا قبل قراءة هذه السلسلة:

○ ما هي لغة الترميز XML ؟ :

تجد الإجابة الوافية باللغة العربية في هذه الدورة

- الجزء الأول:

[/https://www.tech-wd.com/wd/2010/01/02/xml-first-lesson](https://www.tech-wd.com/wd/2010/01/02/xml-first-lesson)

- الجزء الثاني:

[/https://www.tech-wd.com/wd/2010/01/06/xml-second-lesso](https://www.tech-wd.com/wd/2010/01/06/xml-second-lesso)

- الجزء الثالث:

[/https://www.tech-wd.com/wd/2010/01/13/xml-third-lesson](https://www.tech-wd.com/wd/2010/01/13/xml-third-lesson)

- الجزء الرابع:

[/https://www.tech-wd.com/wd/2010/01/16/xml-fourth-lesson](https://www.tech-wd.com/wd/2010/01/16/xml-fourth-lesson)

- الجزء الخامس:

[/https://www.tech-wd.com/wd/2010/01/20/xml-fifth-lesson](https://www.tech-wd.com/wd/2010/01/20/xml-fifth-lesson)

○ ما هو ال Parser :

- تجد الإجابة باللغة العربية هنا :

<https://3alam.pro/ihanan95/articles/syntax-analysis-part-3>

السلام عليكم ورحمة الله وبركاته،

سلسلة المقالات هذه تُناقش بعض أنواع الثغرات المتعلقة بلغة الترميز XML لكن قبل البدء في مناقشة مختلف أنواع هذه الثغرات لابد من مقدّمة قد تكون طويلة بعض الشيء (لذلك تحلّى بالصبر معي قليلاً : [) ؛ الغرض من هذه المقدمة هو تمهيد لكل المفاهيم التي يجب علينا فهمها قبل الخوض في تفاصيل الثغرات، إن كنت تعتقد أنك على معرفة كافية بالأساسيات تستطيع القفز مباشرة لآخر السلسلة حيث أننا سننّبع هذا التدرّج في الشرح:

الجزء الأول: مقدمة

- استخدامات لغة الترميز XML - **مكتمل**
- Web Application and Web Services - **مكتمل**
- لغة الترميز XML وتطوير تطبيقات الويب
 - لغة الترميز XML في ال Web Application - **غير مكتمل**
 - لغة الترميز XML في ال Web Services - **مكتمل**

الجزء الثاني: XML Attacks

- الثغرات المتعلقة بال XML Parser
 - XML External Entity Injection (XXE) - **مكتمل**
 - XML Billion Laugh (BIL) - **غير مكتمل**
- الثغرات المتعلقة بال Web Application و Web Services
 - XML Injection (XMLi) - **غير مكتمل**

في كل جزء سنناقش العديد من الأمور بحول الله ..

نبدأ؟

بسم الله

الجزء الأول: مُقدّمة

• استخدامات لغة الترميز XML

لغة الترميز XML واسعة الاستخدام في مجالات عدّة ولا تقتصر فقط على الويب وتطبيقاته، بعض من هذه الاستخدامات:

تخزين البيانات:

تقوم بعض البرامج في نظم التشغيل باستخدام ال XML لتخزين البيانات، ومن ثم عمل بعض العمليات على هذه البيانات مثل البحث عن قيمة معينة ضمن هذه البيانات، أو ترتيب هذه البيانات وغيرها من العمليات.

ربما أبسط مثال في هذه الحالة برنامج Microsoft Access حيث يُتيح للمستخدم إدراج قاعدة بيانات مخزنة في صيغة ملف XML .

عرض وتمثيل المعادلات الرياضية:

نجد أيضًا استخدام آخر لهذه اللغة من أجل تمثيل المعادلات الرياضية في صفحات الويب، أو لعرض هذه المعادلات في برامج رياضية، إذا كنت مهتم بهذا المجال اطلع على:

MathML (<https://www.w3.org/Math/whatIsMathML.html>) .

عرض وتمثيل ال DNA :

يوجد أيضًا تطبيقات للغة الترميز XML في التقاطع بين المجال الطبي والمجال التقني:

Bioinformatics ، أحد أشهر هذه التطبيقات:

BSML (<http://xml.coverpages.org/bsml.html>) .

والقائمة تطول ولا تسع المساحة هنا لذكر تطبيقات هذه اللغة، لكن إذا كنت مهتم أرشح لك الاطلاع على هذه الصفحة التي تتضمن العديد من التطبيقات و وصف لكل استخدام:

<http://xml.coverpages.org/xml.html#applications>

• Web Application and Web Services

بعد أن تعرّفنا بشكل بسيط وعام على بعض تطبيقات لغة الترميز XML ، لنلقي الضوء الآن على الاستخدام الذي يهمنا في سياق هذه السلسلة ألا وهو **تطوير تطبيقات الويب**. وقبل الخوض في تفاصيل هذا الاستخدام، يجدر بنا التوقف قليلاً وتوضيح العلامات الفارقة بين مفهومين مهمين هنا، وهما: **Web Application** و **Web Services**.

: Web Application

تطبيقات الويب ما هي إلا "برامج" صُممت لتخدم فئة معينة من المستخدمين عن طريق القيام "بمجموعة معينة من المهمّات" التي يحتاجها هؤلاء المستخدمين ومن ثم إعادة "المُخرَج - **Output**" الناتج من إجراء هذه المهمة إلى المستخدم.

هذه البرامج بالطبع "الأدوات" المُستخدمة في بناءها و "طرق التواصل" معها لا تشبه طريقة التعامل مع البرامج الاعتيادية مثل محرّر النصوص أو الآلة الحاسبة وغيرها من البرامج ضمن نظام التشغيل الخاص بك.

لنجري مقارنة بسيطة بين هذين النوعين من البرامج حتى نفهم أوجه الاختلاف بينهما:
البرنامج في جهازك: كل ما تحتاجه للوصول له هو أن تذهب لقائمة البرامج في جهازك.
تطبيق الويب: كما ذكرنا هو "برنامج" لكن موجود على "جهاز" آخر، لذلك حتى تستطيع الوصول له أولاً يجب أن تملك "الوسيلة" التي تمكّنك من الوصول إلى هذا البرنامج في الجهاز الآخر، ويجب أيضاً أن تعرف "الطريق" الموصّل لهذا البرنامج.

أعتقد أننا قمنا بوضع الكثير من علامات التنصيص هنا، لنبدأ بفكّها واحده تلو الأخرى الآن

ذكرنا بدايةً أن تطبيقات الويب ماهي إلا "برامج" فهل هي فعلاً مجرد برنامج بالمعنى البسيط للكلمة؟
هي فعلاً برنامج لو أردنا النظر بصورة عامة، لكن من ناحية تطبيقية قد تكون **مجموعة من البرامج** التي تتواصل مع بعضها، وكل برنامج يؤدي مهمة معيّنة، قد تتساءل لماذا قد يتم تقسيم المهمّات بين برامج عدّة في الأساس؟ لما لا يكون برنامج واحد يؤدي جميع المهمّات؟ الإجابة هي أن كل مهمة من هذه المهمّات عملية معقّدة وفي الغالب تحتاج برنامج واحد خاص بها لإتمامها، الأمر يشبه فكرة

الدوال أو ال **Functions** في الكود الذي تقوم بكتابته، فأنت غالبًا لو أردت كتابة كود لأداء عدة مهمات ستقوم بعمل عدة دوال، وكل دالة تؤدي مهمة معينة، لكن في حالتنا هنا المهام لدينا معقدة جدًا أو بمعنى آخر كل مهمة يندرج تحتها العديد من المهمات المتفرعة لابد من عملها لإكمال المهمة الأساسية لذلك مثلما قسّمنا المهمات في مثال الكود على الدوال هنا قسّمنا المهام الأساسية على عدة برامج.

تخيّل أيضًا أن هذه المهمات زادت تعقيد أكثر، ماذا نقصد تحديدًا بالتعقيد؟ الأمر عند هذه النقطة يتفرّع لاحتمالات عدّة لكن سنركّز هنا على نقطتين متعلّقة بالتعقيد (Complexity) أولاً: سنعتبر أن هذه المهمة معقدة في حالة كانت تستهلك مساحة كبيرة من ذاكرة التخزين. ثانيًا: سنعتبر أيضًا أن المهمة معقدة إذا كانت تحتاج إلى معالج ذو سرعة عالية.

لنلخص الآن كل ما سبق، لدينا مهمات عدّة وكل مهمة في الأساس عبارة عن مجموعة من المهمات المتفرّعة، أيضًا بسبب هذا التفرّع فكل مهمة أساسية بحاجة إلى مساحة تخزين كبيرة جدًا و معالج ذو سرعة عالية .. هل تعتقد الآن أن جهازك البسيط قادر على إتمام هذه المهام؟ [؟

الإجابة تقودنا إلى تفسير ما قصدناه سابقًا بـ "الجهاز" ألا وهو الخادم أو ال **Server** ، فالجهاز الذي نعرف أنه قادر على حل مشكلة التعقيد (ذاكرة كبيرة ومعالج سريع) المرتبطة بتنفيذ هذه المهام هو ما يعرف اليوم بال **Server** ، ذكرنا سابقًا أن المهام معقدة وقد تكون موزّعة على عدّة برامج، لو أردنا تفسير هذا المعنى من منظور الخوادم، فالمهام قد تتمثّل بـ : مهمة استقبال الطلبات القادمة من المستخدمين و مهمة تخزين البيانات الخاصة بالمستخدمين ، كلا المهمتين قد تبدو بسيطة أو معقدة بناءً على عدد المستخدمين الذين تتم خدمتهم وبناءً على كمية البيانات التي يجب تخزينها، فلو كان عدد المستخدمين بسيط وكمية البيانات أيضًا بسيطة فخادم واحد قد يكفي لأداء المهمتين ، لكن في حالة كان عدد المستخدمين الذين يقومون بإنشاء الطلبات مع الخادم كبير وأيضًا البيانات التي يتم تخزينها كبيرة فالأفضل أن نفرّغ الخادم الأول لخدمة المستخدمين فقط ، وخادم آخر يقوم بأداء مهمة تخزين البيانات.

لا يزال لدينا أيضًا بعض المفاهيم المجردة التي تحتاج إلى تفسير مثل:

"الأدوات": ذكرنا أن هذه البرامج (تطبيقات الويب) تُبنى بأدوات، وهذه الأدوات ماهي إلا لغات

البرمجة الخاصة بتطبيقات الويب

"طرق التواصل": الآلية التي يستطيع المستخدم من خلالها التواصل مع تطبيقات الويب هو عن

طريق بروتوكول ال HTTP أو HTTPS

"الوسيلة": يحتاج المستخدم إلى وسيلة تمكنه من التعامل مع تطبيقات الويب عبر بروتوكول ال

HTTPS/HTTP ، وهذه الوسيلة قد تشمل شيئين ، أولًا: اتصال الأنترنت **Network Connection**

، ثانيًا: المتصفح **Browser** (يوجد أدوات أخرى قد تغني عن المتصفح لكن لسنا بصدد مناقشتها حتى الآن)، بالتالي سيستطيع المستخدم عن طريق برنامج المتصفح الموجود بجهازه بالتخاطب مع تطبيق الويب الذي يوجد في الخادم **Server** وكلا البرنامجين (المتصفح في جهاز المستخدم و تطبيق الويب) يتخاطبان ببروتوكول واحد لتبادل البيانات بينهما (HTTP أو HTTPS).

"الطريق": هل تذكر المثال الذي قمنا بذكره سابقًا حول البرنامج في جهازك؟ حتى يستطيع نظام

التشغيل الخاص بك ببدء هذا البرنامج وتشغيله فلا بد أن يتواجد البرنامج على جهازك وأن يتعرف

نظام التشغيل على هذا البرنامج وجميع ملحقاته عن طريق معرفة المسار **Path** الخاص بهذا البرنامج،

في عالم تطبيقات الويب الفكرة مشابهة إلى حد ما، فحتى يستطيع الخادم **Server** بخدمتك فلا بد أن

تقوم بتزويده بمسار البرنامج الذي سيقوم بمعالجة طلبك، وبما أن آلية التواصل مع هذا الخادم هي

عبر بروتوكول ال HTTP/HTTPS فطريقة تمرير المسار الخاص بالبرنامج هي بكتابة الرابط أو ما يعرف

بال URL .

: Web Services

نستطيع القول بأن ال **Web Services** هي نوع خاص من تطبيقات الويب ، ماذا نقصد بذلك؟
في الجزء السابق رأينا أن الطرف المستفيد من تطبيق الويب والذي يقوم بإرسال الطلبات هو **المستخدم User** ، في حالة ال **Web Services** الطرف المستفيد والذي يقوم بإرسال الطلبات لتطبيق الويب هو برنامج آخر، لنأخذ مثال آخر أيضًا، انظر للصورة التالية :

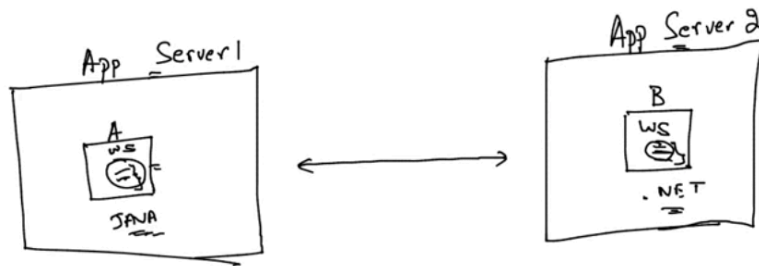


تخيل أنك تملك خادم ويب عليه مجموعة من ال **Classes** التي تؤدي مهمة ما، وفي داخل أحد هذه ال **Classes** يوجد لديك دالة اسمها **getProducts()** وظيفتها هذه الدالة هي إرجاع قائمة بالمنتجات، لو أردت أن تسمح للمستخدمين **Users** بأن يقوموا بالاستفادة وقراءة المخرج **Output** من هذه الدالة فأنت غالبًا ستقوم ببناء صفحات ويب وتعرض بها الناتج للمستخدمين، في هذه الحالة أنت قمت ببناء **Web Application** وبالطبع سيتمكن المستخدمين عن طريق المتصفح بعرض هذه الصفحات وقراءة المخرج من الدالة **getProducts()** .

الآن لنفترض أن صديقك يملك خادم ويب خاص به، وأراد أن يستفيد من النتائج التي تقوم بإرجاعها الدالة **getProducts()** (لاحظ هنا أن عملية ال **run** لهذه الدالة تتم في داخل الخادم الخاص بك) ، ماذا ستفعل حينها؟

قد تقول بسيطة سأقوم بإعطاء الكود المصدري الخاص بهذه الدالة وهو بدوره سيقوم بإدراجها في أحد ال **Classes** في الخادم الخاص به (أي عملية ال **run** للدالة **getProducts()** ستكون في الخادم الخاص بصديقك) ..

قد يكون هذا الحل مناسب في بعض الحالات، لكن ماذا لو أن الدالة **getProducts()** التي تعمل في خادمك تعتمد في عملها على عدة دوال أخرى وهذه الدوال موجودة في **Classes** مختلفة ؟ هل ستقوم أيضًا بنقل كامل الكود المصدري وتسليمه لصديقك ؟
ماذا أيضًا لو أن دالة ال **getProducts()** تعتمد في عملها على قراءة بعض البيانات من قاعدة بيانات ، وقاعدة البيانات هذه بالطبع موجودة أيضًا في خادمك!
في مثل هذه الحالات يأتي دور ال **Web Service** حتى يحل هذه المشكلة،
انظر للصورة التالية :



في الصورة أعلاه لنفترض أن الخادم الخاص بك هو ال **App Server 1** والخادم الخاص بصديقك هو ال **App Server 2** ، سيتخاطب البرنامج في خادم صديقك (والذي قد يكون برمج بلغة تختلف عن اللغة التي بنيت بها تطبيق الويب في خادمك) مع تطبيق الويب في خادمك ويقرأ منه المخرج فقط لا غير (عملية الوصول والقراءة وعرض النتائج أنت من تحكمها هنا).

• لغة الترميز XML وتطوير تطبيقات الويب

بعد أن تعرفنا على المفهومين السابقين، لنبدأ الآن التنقيب حول استخدام الـ XML في كل حالة.

لغة الترميز XML في الـ Web Application :

غير مكتمل

لغة الترميز XML في الـ Web Services :

ذكرنا سابقاً أن الـ **Web Services** ماهي إلا آلية تُمكن بقية المطورين من الوصول إلى المخرجات الخاصة بالكود المتواجد في خادمك (الاستفادة منه)، فأين موقع لغة الترميز XML في هذا السياق ؟ حتى يستطيع أي مطور الوصول والتخاطب مع الـ **Web Service** التي تقدمها لابد أن يوجد بروتوكول موحد بينهم يتبادلان الطلبات والردود من خلاله، وهذا البروتوكول يجب أن يعمل حتى وإن اختلفت اللغات خلف كلا الطرفين (**independence**) ، نقصد بالإختلاف هنا هو اختلاف اللغة التي يستخدمها المطور في تطبيقه عن اللغة التي قمت أنت باستخدامها في بناء الـ **Web Service**.

البروتوكول الذي يُتيح للطرفين التواصل مع إختلاف التقنيات هو الـ **Simple Object Access Protocol (SOAP)**. بروتوكول الـ **SOAP** في الطلبات **Request** والردود **Response** الخاصة به يستخدم الـ XML. أيضاً لابد من إخبار المطورين بالمعلومات التي يجب عليهم إرسالها (**Inputs**) إلى الـ **Web Service** حتى تُعالج طلبهم ويصلهم الرد مع النتائج (**Outputs**) ونستطيع إدارة هذا الأمر عن طريق الـ **Web Service Definition Languages (WSDL)** .

لنأخذ المثال التالي :

يوجد **Web Service** تتيح للمطورين قراءة البيانات الخاصة بالطقس عن طريق إرسال قيم الـ **latitude** والـ **longitude**، يستطيع المطور قراءة البيانات من هذه الـ **Web Service** وتوظيفها في تطبيقه كيفما يشاء ، في المثال التالي قمنا بإرسال طلب **Request** إلى هذه الـ **Web Service** ، لاحظ أننا مررنا القيم **latitude** و **longitude** إلى الدالة **NDFDgen** ، ولاحظ أيضاً أن هذا الطلب والذي تم تمريره عبر بروتوكول الـ **SOAP** هو مُضمّن في داخل الـ **Body** الخاص ببروتوكول الـ **HTTP**

```
POST /forecasts/xml/SOAP_server/ndfdXMLserver.php HTTP/1.1
content-type: text/xml; charset="utf-8"
...
```

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope ...>
  <env:Body>
    <ns0:NDFDgen>
      <latitude xsi:type="xsd:decimal">40.28</latitude>
      <longitude xsi:type="xsd:decimal">-79.49</longitude>
      ...
    </env:Body>
  </env:Envelope>
```

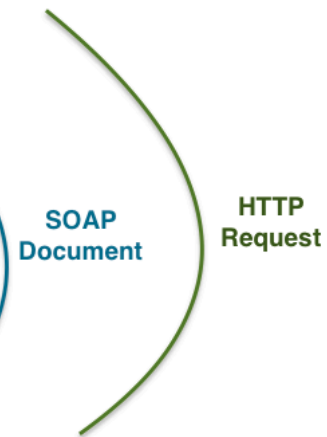


FIGURE 9.1 Portions of an HTTP request and embedded SOAP document sent to a weather-related web service.

بعد إرسال الطلب ، تم إستقبال هذا ال **Response** من ال **Web Service** ، ولاحظ أيضًا أن المخرجات من الدالة السابقة تم تمريرها عبر بروتوكول ال **SOAP** باستخدام لغة الترميز **XML**

```
HTTP/1.1 200 OK
Server: Apache/2.0.46 (Red Hat)
...
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope ...>
  <SOAP-ENV:Body>
    <NDFDgenResponse>
      <xmlOut xsi:type="xsd:string">
        ...
        <name>Daily Maximum Temperature</name>
        <value>47</value>
        ...
      </xmlOut>
    </NDFDgenResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

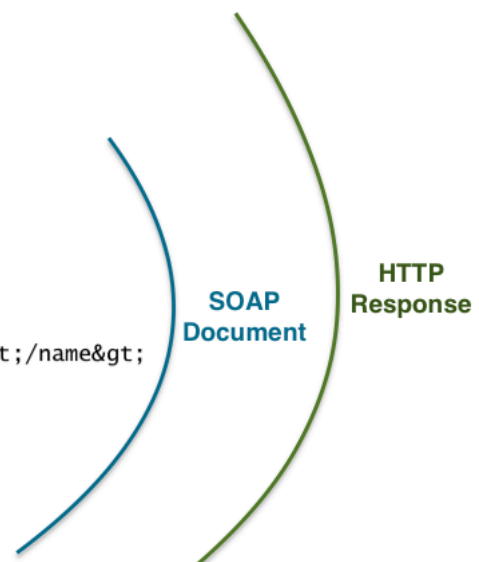


FIGURE 9.2 Portions of the HTTP response with embedded SOAP document received in response to the request of Figure 9.1.

ذكرنا سابقًا أن المطورين حتى يستطيعوا الإستفادة من أي **Web Service** فيجب عليهم أولاً أن يعرفوا ما الذي تتيحه لهم هذه ال **Web Service** ، أي ما هي الدوال **Functions** التي يستطيعون قراءة المخرجات **Output** منها ، وما القيم **Inputs** التي يجب تمريرها لهذه الدوال ، وما هي أنواع هذه المدخلات **Data Type** ، وأيضًا ما هو نوع المخرجات حتى يستطيع المطور التعامل معها بطريقة مناسبة في الكود الخاص به، وكل هذا يتم إدارته عبر ال **WSDL** ، فمطور ال **Web Service**

سيضع كل التفاصيل هذه في هذا ال **Document** ، لنأخذ الآن نظره على هذا الملف المرتبط بنفس ال **Web Service** في المثال السابق

```
<?xml version="1.0"?>
<definitions ...>
  ...
  <message name="NDFDgenRequest">
    <part name="latitude" type="xsd:decimal" /> توضيح للمُدخلات وأنواعها
    <part name="longitude" type="xsd:decimal" />
  ...
  </message>

  <message name="NDFDgenResponse">
    <part name="xmlOut" type="xsd:string" /> توضيح للمُخرجات وأنواعها
  </message>

  <portType name="ndfdXMLPortType">
    <operation name="NDFDgen">
      <documentation>...</documentation>
      <input message="tns:NDFDgenRequest"/> توضيح للدالة المتأخذة
      <output message="tns:NDFDgenResponse"/>
    </operation>
    ...
  </portType>
  ...
</definitions>
```

WSDL Document

FIGURE 9.3 Portions of a National Weather Service WSDL document defining an operation named NDFDgen along with two of its inputs and its output (obtained from <http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl>).

وبمناسبة ذكر ال **WSDL** هنا يجدر بنا الإشارة بأنه توجد **APIs** في كل لغة تتيح للمطورين التعامل مع ال **WSDL document** واختصار الكثير من العمليات مثل إنشاء ال **SOAP document** وتضمينه في داخل ال **HTTP request** ، وعند إستقبال ال **HTTP Response** من ال **Web Service** تساعد هذه ال **APIs** أيضًا في إستخراج ال **SOAP Document** .

الجزء الثاني: XML Attacks

قبل البدء في مناقشة مختلف أنواع الثغرات المتعلقة بلغة الترميز XML ، لنلقي نظرة شمولية (انظر للصورة الآتية) على هذه الأنواع ونعرف كل نوع منها ما الهدف الذي يُحدث به الضرر:

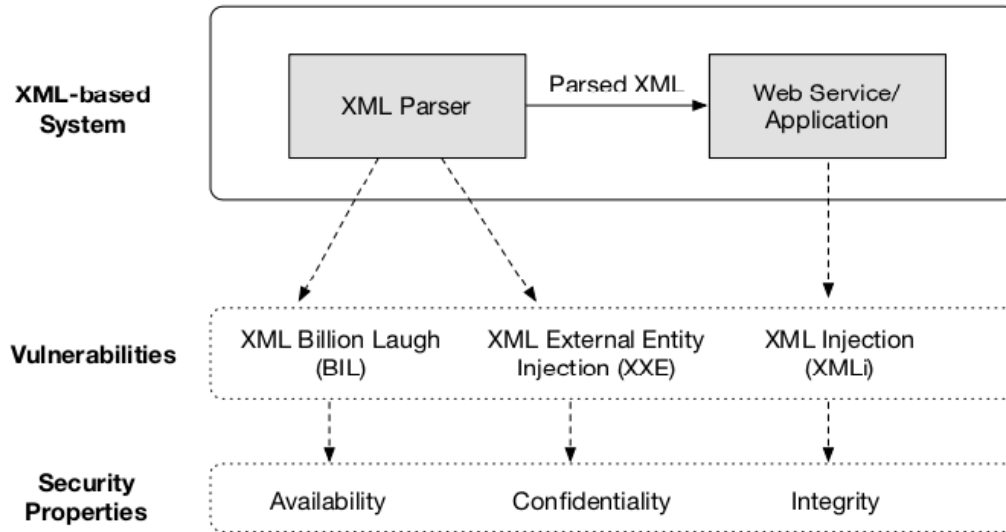


Figure 1.1: Vulnerabilities in XML-based System

• الثغرات المتعلقة بال XML Parser

تطبيقات الويب التي تعتمد على لغة الترميز XML في بعض أجزائها وعملياتها يقوم مطوريها بإستخدام XML Parser خاص باللغة المستخدمة في تطبيق الويب ، ال XML Parser يساعد في قراءة (Parsing) أكواد ال XML وجعلها مفهومه لتطبيق الويب حتى يستطيع التعامل معها، أي أن ال XML Parser أشبه ب API بين كود ال XML والكود الخاص بتطبيق الويب. فعوضًا عن أن يتعامل مطوّر الويب مع كود ال XML بشكل مباشر ويحاول تفسير كل سطر بنفسه، يقوم بإستخدام XML Parser يؤدي هذه الوظيفة، وبالمناسبة هذه الميزة هي أحد الأسباب التي جعلت لغة الترميز XML واسعة الإنتشار، فالعديد من لغات البرمجة لديها مكتبات خاصة بال XML Parser ، بالتالي عملية التخاطب بين تطبيقين من لغتين مختلفتين بغرض تبادل أو قراءة البيانات ستكون مهمة سهلة إلى حدٍ ما إذا أوجدنا لغة مشتركة بين هذين التطبيقين ، وفي هذه الحالة نقصد لغة الترميز XML .

يوجد العديد من الأنواع المختلفة للـ **XML Parser** ولسنا بصدد مناقشتها هنا، لكن هذه قائمة ببعض الـ **XML Parser** المفتوحة المصدر لبعض اللغات وأطر العمل:

:Java

https://docs.oracle.com/cd/B28359_01/appdev.111/b28394/adx_j_parser.htm#ADXDK3000

:Python

<https://docs.python.org/3/library/xml.html>

:PHP

<https://www.php.net/manual/en/book.xml.php>

: .NET Framework

<https://docs.microsoft.com/en-us/dotnet/standard/data/xml>

بعد هذه المقدمة حول الـ **XML Parser** ، لننتقل الآن للثغرات التي تحدث بسبب بعض نقاط الضعف في الـ **XML Parser** أو بسبب جهل مطور تطبيق الويب بطبيعة الخصائص التي يقدمها هذا الـ **Parser** وحدود إمكانياته.

: XML External Entity Injection (XXE)

قبل أن نتعرف على هذا النوع من الثغرات لنعرف بدايةً ما هو الـ **XML Entity** ؟
الـ **XML Entity** نستطيع إعتباره متغير يحمل بيانات وهذه البيانات قد تكون في نفس (Internal) ملف الـ **XML** أو في خارجه (External) .
هذا مثال على تعريف **ENTITY** ضمن ملف الـ **XML** :

```
<!DOCTYPE ARTICLE
[
  <!ELEMENT ARTICLE (TITLEPAGE, INTRODUCTION, SECTION*)>
  <!ELEMENT TITLEPAGE (#PCDATA)>
  <!ELEMENT INTRODUCTION (#PCDATA)>
  <!ELEMENT SECTION (#PCDATA)>

  <!ENTITY topics SYSTEM "Topics.xml"> —————> External Entity
  <!ENTITY title "A Short History of XML"> —————> Internal Entity
]
>
```

في المثال أعلاه عرّفنا نوعين مختلفين من الـ **XML ENTITY** وهما :

External ENTITY -

Internal ENTITY -

ما يهمنا هنا هو النوع الأول **External ENTITY** وهذا هو الـ **Syntax** الخاص بتعريفه :

```
<!ENTITY EntityName SYSTEM SystemLiteral>
```

- **EntityName** : يرمز لأسم هذا المتغير

- **SystemLiteral** : ترمز إلى المسار المتواجد به الملف، وهنا بإمكاننا استخدام أنواع مختلفة

من الـ **URI Scheme** مثل الـ **File** , **HTTP**

الآن بعد أن تعرّفنا على الـ **XML External Entity** ،

لنعرف ماهي ثغرة الـ **XML External Entity Injection** ؟

بما أنها ثغرة من نوع **Injection** فلا بد أن يتواجد "مكان" تأتي منه البيانات من المستخدم (أو من

تطبيق آخر) ومن ثم يتم خلط هذه البيانات مع الكود وترسل إلى الـ **Parser**

(إطلع على سلسلة مقالات الـ **Injection Attacks** لمزيد من التفاصيل حول هذه النقطة:

(<https://0xb1tbyte.github.io/2019/11/05/TheFundamentalCause.html>)

وفي حالتنا هنا تطبيق الويب يستقبل قيم من المستخدمين وهذه القيم قد يتم خلطها مع **XML**

Code، ولا يقوم تطبيق الويب بعمل فلترة كافية للمدخلات ممّا يتيح للمخترق حقن كود **XML**

وتحديداً نقصد هنا حقن **XML External Entity**

لنتوقف قليلاً عن الشرح ونبدأ بالجانب العملي حتى تتضح الصورة أكثر:

سنقوم بالتطبيق وحل تحدّي بسيط على هذه الـ **Machine** :

<https://www.vulnhub.com/series/xxe-lab,174>

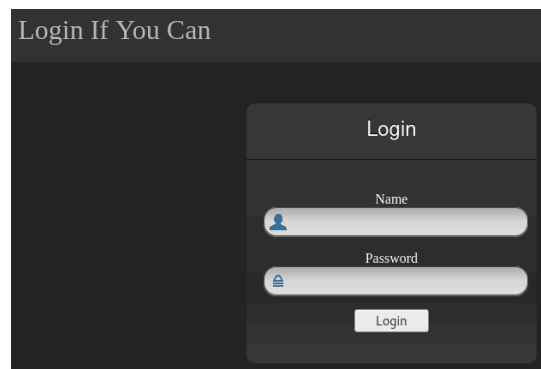
إعدادات المعمل ستكون كالآتي:

- تطبيق الويب المصاب بالثغرة: <http://172.16.220.134>

- الـ **Machine** التي نقوم بالإختبار من خلالها: Kali machine

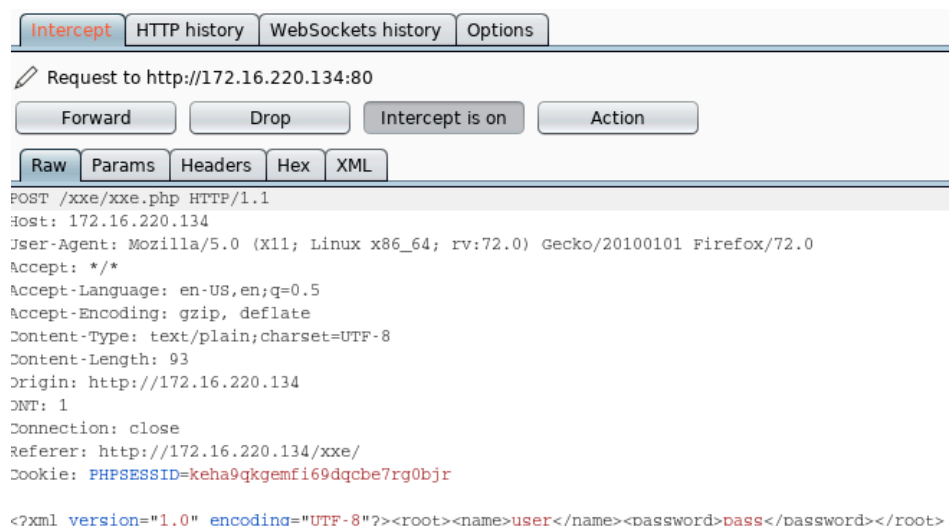
بعد الدخول على الصفحة الخاصة بالتطبيق (<http://172.16.220.134/xxe>) نجد واجهة

تسجيل الدخول هذه:



نحاول تسجيل الدخول وإعتراض الطلب عن طريق أداة **Burp Suite** قبل أن يتم إرساله إلى تطبيق

الويب حتى نفهم ما الآلية التي يعمل بها التطبيق



بعد إعتراض الطلب، نلاحظ أن البيانات الخاصة بالمستخدم يتم تمريرها عبر ال **XML** ، نُلقِي نظرة على ال **Source Code** ونجد الدالة **XMLFunction()** والتي تقوم ببناء ال **xml document** وتمريره إلى تطبيق الويب :

```
(index) X
19 function XMLFunction(){
20     var xml = '' +
21         '<?xml version="1.0" encoding="UTF-8"?>' +
22         '<root>' +
23         '<name>' + $('#name').val() + '</name>' +
24         '<password>' + $('#password').val() + '</password>' +
25         '</root>';
26     var xmlhttp = new XMLHttpRequest();
27     xmlhttp.onreadystatechange = function () {
28         if(xmlhttp.readyState == 4){
29             console.log(xmlhttp.readyState);
30             console.log(xmlhttp.responseText);
31             document.getElementById('errorMessage').innerHTML = xmlhttp.responseText;
32         }
33     }
34     xmlhttp.open("POST","xxe.php",true);
35     xmlhttp.send(xml);
36 };
```

الجزء الذي يهْمُنَا تحديداً في الكود هو أن المدخلات القادمة من المستخدم (اسم المستخدم وكلمة المرور) يتم خلطها مع كود ال **XML** ولا يوجد أي عملية فلترة مُسبقة لهذه المدخلات، ومن هنا نستطيع الحقن ، كِلا المتغيرين **name** و **password** مُصابين ونستطيع الحقن من خلالهما :

```
23     '<name>' + $('#name').val() + '</name>' +
24     '<password>' + $('#password').val() + '</password>' +
```

بعد تحليل الكود ، لنعود الآن إلى ال **Repeater** في ال **Burp Suite**

Request

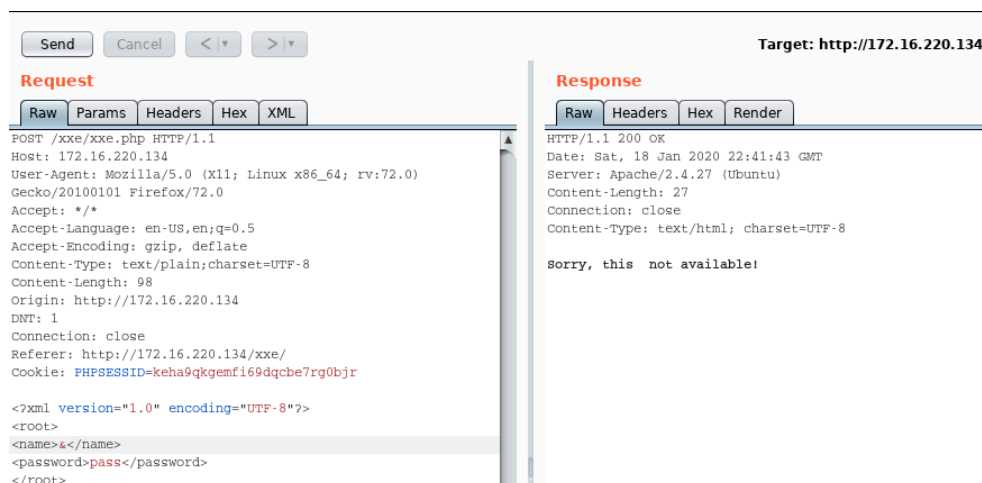
Raw	Params	Headers	Hex	XML
POST /xxe/xxe.php HTTP/1.1 Host: 172.16.220.134 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:72.0) Gecko/20100101 Firefox/72.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Content-Type: text/plain; charset=UTF-8 Content-Length: 93 Origin: http://172.16.220.134 DNT: 1 Connection: close Referer: http://172.16.220.134/xxe/ Cookie: PHPSESSID=keha9qkgemfi69dqcb7rg0bjr				
<?xml version="1.0" encoding="UTF-8"?><root><name>user</name><password>pass</password></root>				

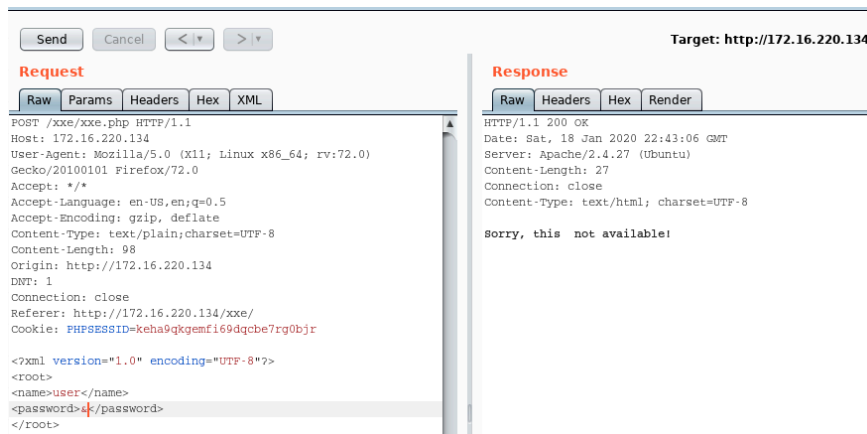
نبدأ الإختبار عن طريق تمرير أحد قيم ال **Meta-character** في ال **XML** مثل :

Table 4.1: XML Meta-characters

Character	Consequence
<	Opening a tag without closing it.
&	This is a character for escaping meta-characters, which makes an XML malformed when being used alone.
>	Closing a tag without opening it.
'	It makes the name specification syntactically incorrect when added to an attribute name.
"	Similar to the previous one.
<!--	This sequence of characters represents the beginning/end of a comment and is not allowed in attribute values.
]]>	This is a delimiter for the CDATA section and is not allowed in values of elements.

في حالة كان تطبيق الويب لا يقوم بعمل الفلترة للمدخلات فحقن أحد هذه القيم في المدخلات سيعمل على إحداث خطأ في البنية السليمة لملف ال **XML** ، مما يجعل ال **Parser** يُظهر لنا رسالة خطأ في ال **Response** ، في الخطوة الآتية مرّنا القيمة **&** ضمن اسم المستخدم أولاً حتى نتأكد أنه مصاب ، ومن ثم أعدنا المحاولة على المتغير الخاص بكلمة المرور .





نلاحظ أن رسالة الخطأ التي توقعناها من ال **Parser** لم تظهر ضمن ال **Response** ، لنبدأ الآن بحقن شيء آخر ، على سبيل المثال لنحاول الحقن بـ **XML External Entity** كالآتي:

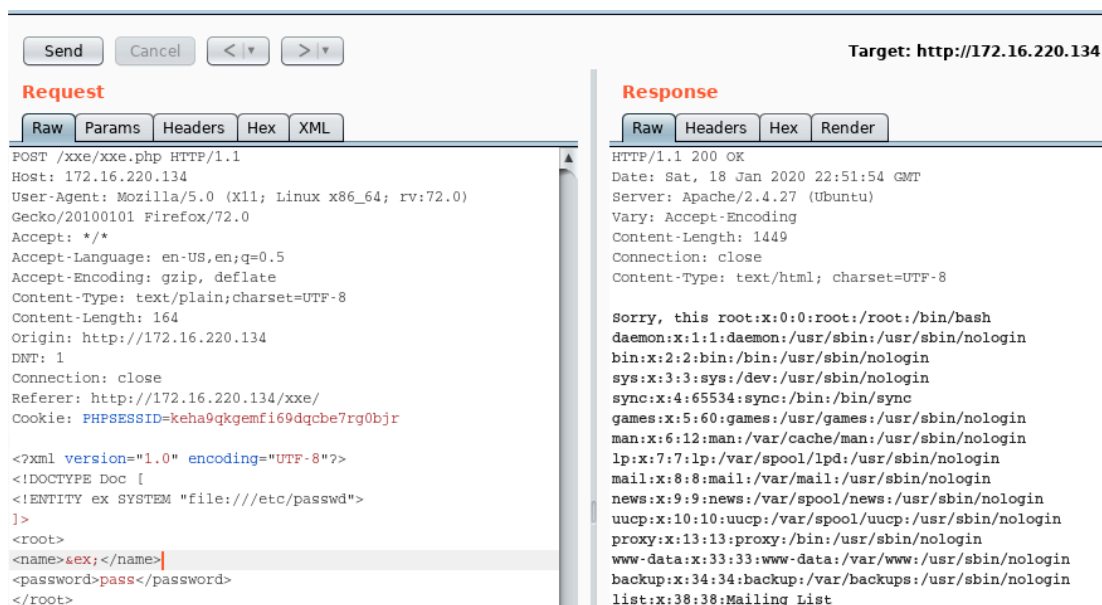
```
<!DOCTYPE Doc [
```

```
<!ENTITY ex SYSTEM "file:///etc/passwd">
```


ومن ثم نستدعي هذا ال **Entity** ضمن أحد قيم المدخلات

```
<name>&ex;</name>
```

ونرسل ال **Request** :



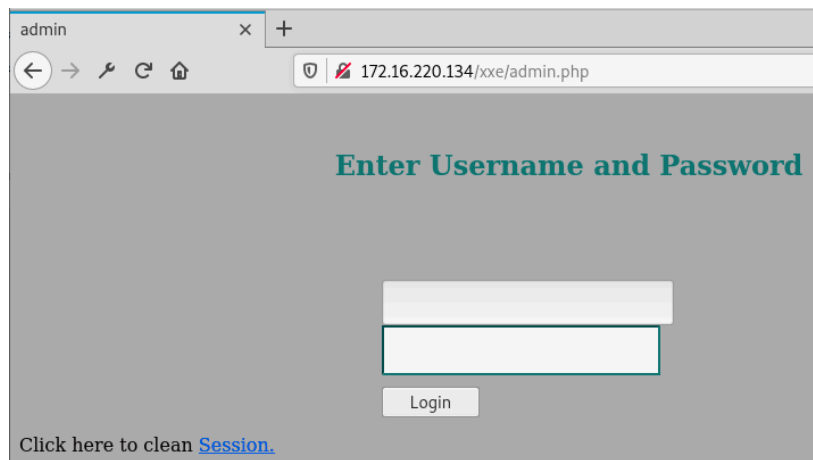
جميل!! ما الذي حصل هنا ؟

إستطعنا قراءة ملف ال **passwd** عن طريق تعريف **External Entity** يحمل المسار الخاص بهذا

الملف ، ومن ثم حقنا هذا ال **External Entity** في أحد قيم المدخلات

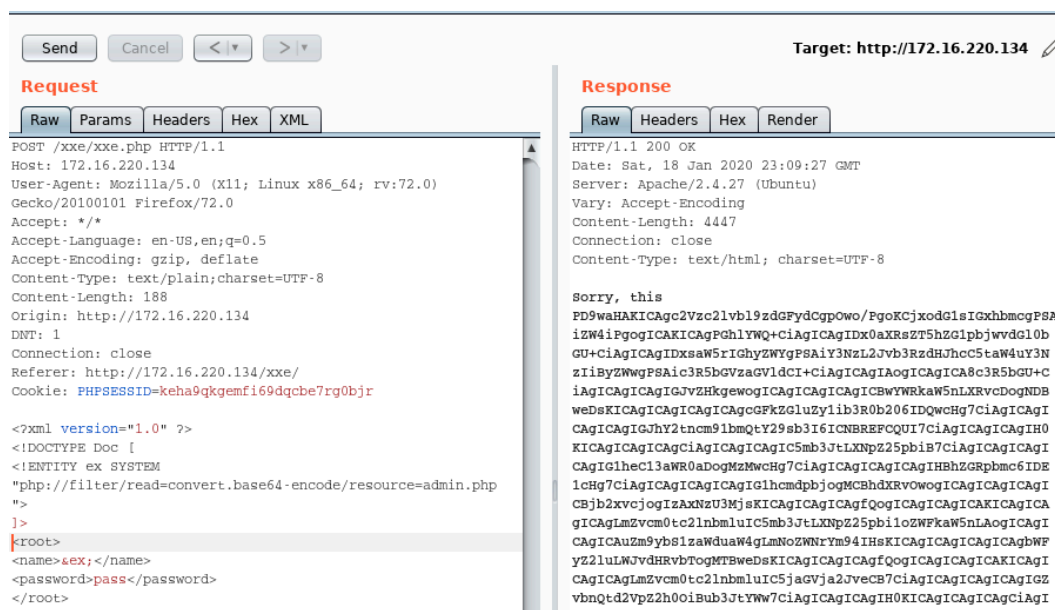
لننتقل لمستوى آخر من الحقن،

بعد قراءة محتوى ملف الـ **robots.txt** وجدنا صفحة تسجيل فرعية خاصة بالـ **admin**



بعد محاولات عدة لإختبار صفحة الدخول لنفس الثغرة وتحليل الكود لم نتوصل لشيء،

لكن ماذا عن قراءة محتوى ملف الـ **admin.php** عن طريق الثغرة في صفحة تسجيل الدخول الأولى؟
لنلقي نظرة!



إستطعنا قراءة محتوى الملف أيضًا!

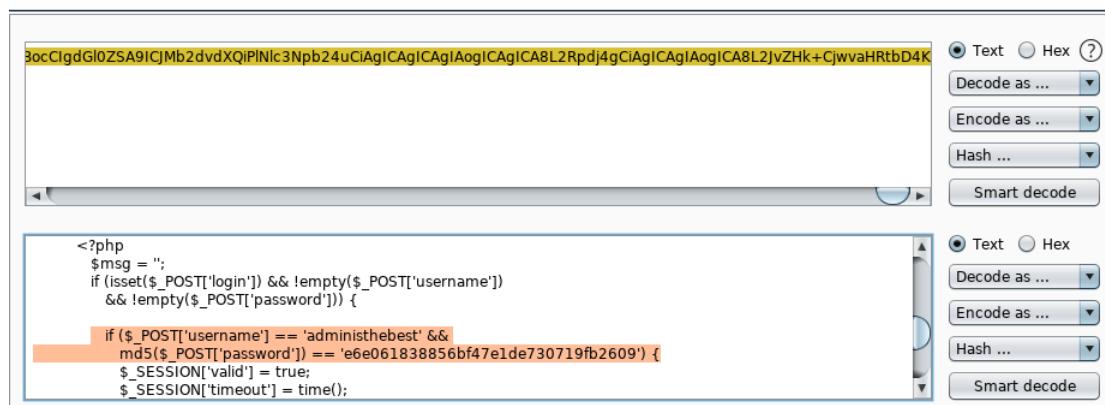
لاحظ أننا قمنا بعمل **Encoding** لمحتوى الملف كآلاتي (لمعرفة السبب وراء هذه الخطوة إطلع على

هذا الشرح: <https://github.com/0xb1tByte/eWPTXv1>

:([Journey/blob/master/XML%20Attacks/XML%20Attacks%20Notes.pdf](https://github.com/0xb1tByte/eWPTXv1/blob/master/XML%20Attacks/XML%20Attacks%20Notes.pdf)

<!ENTITY ex SYSTEM "php://filter/read=convert.base64-
encode/resource=admin.php">

بعد عمل **Decoding** لمحتوى الملف ، وجدنا هذه البيانات ضمن الصفحة



نقوم بتسجيل الدخول باستخدام هذه البيانات :

Enter Username and Password

Maybe Later

administhebest

••••••••

Login

ومن ثم نحصل على العلم :

Enter Username and Password

You have entered valid use name and password
Here is the **Flag**

|

Login

: XML Billion Laugh (BIL)

غير مڪتمل

• الثغرات المتعلقة بـ Web Services و Web Application

: XML Injection (XMLi)

غير مكتمل

• المراجع

كتب:

- **WEB TECHNOLOGIES A Computer Science Perspective** , Chapter 7 (XML) & Chapter 9 (Web Services)
- **The Web Application Hackers Handbook** , Chapter 10 (Injecting into XML Interpreters)

أوراق علمية:

- **Automated and Effective Security Testing for XML-based Vulnerabilities**, By : Sadeeq Jan
- **XXE Explanation and Exploitation**, By : Haboob Team

يوتيوب:

- **SOAP Web Services 01 - Introduction To Web Services :**
https://www.youtube.com/watch?v=mKjvKPlb1rA&list=PLqq-6Pq4lTTZTYpk_1DOowOGWJMIH5T39&index=1