



Non conventional attacks – Some things your security scanner won't find...

Tom Van der Mussele
Security Analyst
Verizon Business Security Solutions
tom.vandermussele@verizonbusiness.com
+352691191974

OWASP

23/05/2011

Copyright © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document
under the terms of the OWASP License.

The OWASP Foundation

<http://www.owasp.org>

Non conventional attacks – Some things your security scanner won't find

■ Introduction

■ Non Conventional Attacks

- ▶ Attacking XML Streams
- ▶ Breaking out of the interface
- ▶ Exploiting Trust Relationships in Web Apps

■ Questions

Introduction

Tom Van der Mussele
Lead Security Analyst



Security Management Program – Application Certification
Luxembourg

tom.vandermussele@verizonbusiness.com

Non Conventional Attacks



Attacking XML Streams

■ What am I attacking?

- ▶ Applications use XML streams for dynamic data querying, providing standardized formats and consistent data handling – making the browsing session more interactive – « push » data to end user without having to 'refresh' the page (i.e. heartbeat, etc...)
- ▶ These applications rely heavily on Expressing Style Sheets (XSL) for data formatting and representation
- ▶ For example: Google Calendar, Google Docs,...

Attacking XML Streams

■ What is the goal?

- ▶ XML streams are data which is sent in a raw format and interpreted by the client side, rather than represented on the server side.
- ▶ The goal for attackers is to escape context of that specific end user to try and obtain data belonging to other users.
- ▶ XML streams are often more valuable to attackers since less content validation gets done at the server side level. Oftentimes layout templates are used as 'filters'.

Attacking XML Streams

■ Why don't scanners pick this up?

- ▶ XML streams are very application specific and make it difficult for scanners to differentiate between an authorized data set and an unauthorized data set.
- ▶ Scanners typically search for their own reflected input in the output stream or typically expect a certain response or error code

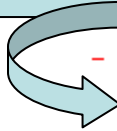
Attacking XML Streams

■ How do I achieve it?

- ▶ Analyze the XML data set to identify the structure of a valid response
- ▶ Utilize input manipulation (e.g. different templates), cookie modification, HTTP meta data (e.g. User-Agent / MSIE, FireFox, BlackBerry etc), changing parameter values (e.g. UserID, EmployeeID,...) to try and force a different response from the server

Attacking XML Streams - Examples

<https://bank.be/accountdetails.asp&markup=xml&template=index.xsl>

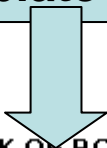


```
- <BANK>  
<BANKNAME>BANK OF BOB</BANKNAME>  
<BANKCODE>BOB</BANKCODE>  
<BANKASSET>P3B-F</BANKASSET>  
</BANK>
```

- By changing the parameter « index.xsl » to a non existing one « blah.xsl » the application was unable to format the XML anymore.

<https://bank.be/accountdetails.asp&markup=xml&template=blah.xsl>

- Due to loss of the formatting template, this resulted in disclosure of the full XML file



```
<BANKNAME>BANK OF BOB</BANKNAME>  
<BANKCODE>BOB</BANKCODE>  
<BANKASSET>P3B-F</BANKASSET>  
</BANK>  
<BANK>  
<BANKNAME>ALICE BANK</BANKNAME>  
<BANKCODE>AB</BANKCODE>  
<BANKASSET>O4C-K</BANKASSET>  
</BANK>  
<BANK>
```



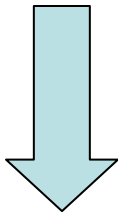
Attacking XML Streams - Examples

https://humane-bronnen.com/Router/Transaction/Erp?_EVT=CHG&_EID=1200

- By changing the parameter value of fields it is possible to test the least privilege principle. A scanner doesn't understand what data is being changed and will try the typical ` or 1>0--

https://humane-bronnen.com/Router/Transaction/Erp?_EVT=CHG&_EID=1337

- The attack resulted in changing another profile than was intended by the application



```
<XHR11.1>
- <HR11.1>
  <_PDL>TEST</_PDL>
  <_TKN>HR11.1</_TKN>
  <_PRE>TRUE</_PRE>
  <Message>Change Complete - Continue</Message>
  <MsgNbr>000</MsgNbr>
  <StatusNbr>001</StatusNbr>
  <FldNbr>_f0</FldNbr>
</HR11.1>
</XHR11.1>
```

Attacking XML Streams

■ How can these attacks be prevented?

- ▶ Mainly a design issue – Prevent it early on in the development process. User should never be able to explicitly specify the layout template.
- ▶ Securing Input: Implement least privilege principle on XML streams (tie XML query data to user session)
- ▶ Securing Output: Perform proper output validation on XML stream (e.g. bind context of session to the user session)

Breaking out of the interface

■ What am I attacking?

- ▶ Certain web apps contain front end interfaces written specifically to interact with 'legacy' backend systems such as iSeries, zSeries, Unix, ...
- ▶ Good examples are the front ends for CICS on iSeries

Breaking out of the interface

- What is the goal I am trying to achieve?
 - ▶ Escape the web based front interface and interact with the backend interface
 - More often than not, user based credentials are end to end (i.e. user logged on to web app is also replicated on back end)
 - Privilege escalation is limited, but not impossible – If you escape the interface, you can log in as different user
 - Functionality escalation is more probable – i.e. web front end restricts to a limited amount of function but actual account is more permissive

Breaking out of the interface

■ Why don't scanners pick this up?

- ▶ The application keeps presenting data and doesn't necessarily throw out an error, so it is difficult for a scanner to detect
- ▶ Many of these applications are very custom development, as such it's difficult for scanners to find known 'vulnerability patterns'

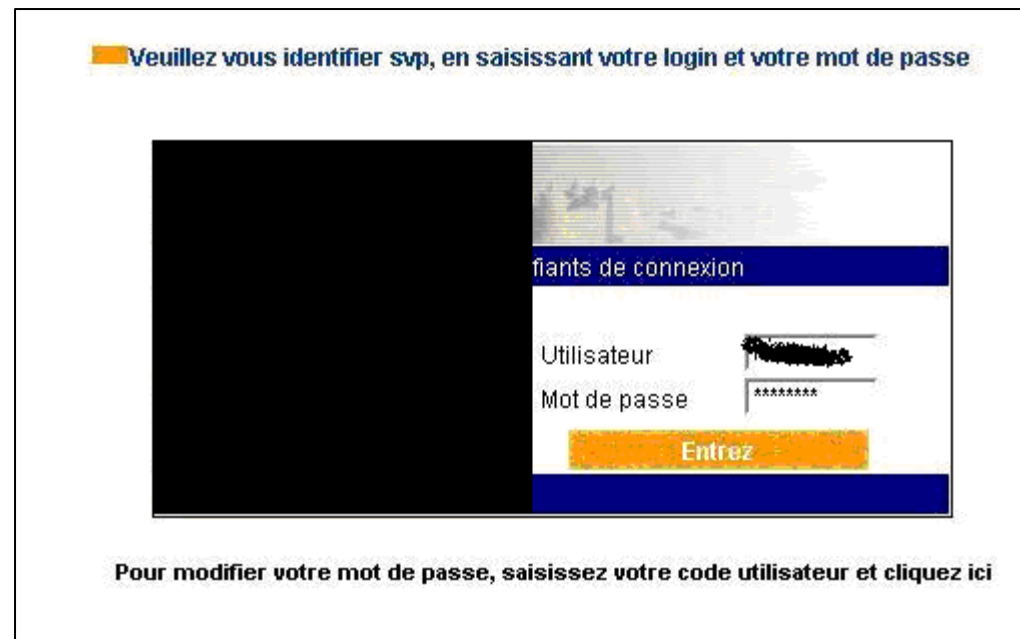
Breaking out of the interface

■ How did I achieve this?

- ▶ Using special characters or encoded characters interpreted by the Web Application Environment (i.e. Java) I was able to break out the interface restrictions
- ▶ This way I was also able to send Special command keys by hexing string (e.g. Function Keys – F1, F2,..)
- ▶ By changing client side parameters such as Terminal Type, Environment, Port, etc...

Breaking out of the interface - Example

- The front end looks like any other web interface, and interacts via a middleware with the legacy system



Breaking out of the interface

- By using encoded or special characters and/or changing client side parameters it is possible to jump out of the web front end and interacting directly with the legacy system

```
-----  
NoPers: _____  
Mot de passe: _____ Nouv1: _____ Nouv2: _____  
Firme: _____  
Droit-System: _____  
-----INDICATION / MESSAGES -----  
***** Erreur Système      Code-Retour :   70      *****  
*****  
***** Nbre de transactions actives Supprimées :    2 *****  
*****      F12 pour sortir d'*****  
-----  
----- ZSTM001 ----- CICS3 0878 -----
```

Breaking out of the interface

- How can these attacks be prevented?
 - ▶ Restrict input into the application, especially filter out any function keys and similar when using blacklisting
 - ▶ Limit the terminal types available to the user through the web frontend (i.e. some users should not be allowed to log on from the web front end)

Exploiting Trust Relationships in Web Apps

■ What am I attacking?

▶ SSL VPN components

- i.e. SSL VPN devices often have multiple applications hosted behind them – Obtaining a cookie to the SSL VPN (through a flaw in the back end application) may allow access to other services (i.e. OWA, Citrix, Juniper, ...)

▶ Single Sign On and Multi Factor Authentication Systems

- In house developed SSO applications which have explicit trust levels between initial authentication and subsequent authorization to components (i.e. initial challenge-response authentication, followed by a single sign on to back end apps)

Exploiting Trust Relationships in Web Apps

■ What is the goal I am trying to achieve?

- ▶ Accessing Authenticated Applications via the cookie of the SSL VPN
 - Multiple applications are provided through a single gateway (SSL VPN) and the goal is to access other applications by exploiting a vulnerability in one of the applications
- ▶ Accessing Authenticated Applications after Single Sign On Authorization
 - The Single Sign On will authorize the user to the back end applications using their individual authentication mechanisms

Exploiting Trust Relationships in Web Apps

■ How did I achieve this?

▶ SSL VPN attacks

- Exploit a vulnerability in one of the backend applications to retrieve the SSL VPN cookie (i.e. XSS on one of the web apps), rendering access to other the SSL VPN services for that user

▶ Single Sign On and Multi Factor Authentication Systems

- Accessing the authentication pages of the backend applications directly and logging in with different credentials than the Single Sign On

Exploiting Trust Relationships in Web Apps

■ How can these attacks be prevented?

▶ SSL VPN Cookie Stealing

- Set cookies to HTTPOnly

▶ Single Sign On and Multi Factor Authentication Systems

- Use cryptographic techniques such as signed values from the Single Sign On box to the back end application (these strings can then contain username information and can be compared to the active session).

More non conventional attacks

- Access Control Issues
- Access Rights Issues
- User-specific files
- Temporary files
- Session states
- ...

Questions

