



Unbreakable ABAP? Vulnerabilities in custom ABAP Code

Markus Schumacher, Co-Founder Virtual Forge GmbH

10.- 12. März 2010
Print Media Academy, Heidelberg

Virtual Forge GmbH - <http://virtualforge.de>

- „GmbH“ since 1.1.2006, headquarters in Heidelberg
- Long-lasting consulting experience
- Application security, focus SAP from day 1
- Code Profiler, <http://www.codeprofilers.com>
- SAP audits and code reviews
- Book: „Sichere ABAP-Programmierung“, <http://sap-press.de/2037>
- Trainings



Agenda

- ABAP development - risks in Web applications (example)
 - ABAP/BSP vs. OWASP Top 10
- Examples of vulnerabilities in custom coding
 - Business Server Pages
 - ▶ Inline ABAP in HTML
 - ▶ HTMLB-Tag-Library
 - Open SQL
 - ▶ Dynamic Open SQL
 - ▶ SQL-Injection
- Conclusion

ABAP in a Nutshell

- Exists since ~30 years
- COBOL-like syntax
- “grown language”
 - several programming paradigms at the same time
 - very context sensitive, no reserved keywords
- DB-independent SQL-dialect built in
- Code is stored in DB
- Development environment developed in ABAP
 - Code stored on server
 - Access via transaction SE80
 - Transport management

ABAP and Open Source (but not free!)

- Sourcecode is completely available in a SAP installation
 - ▶ „SAP standard“ code plus custom coding
- Customers can change code
 - ▶ Copy, rename, and modify code
 - ▶ Change SAP standard code („modification“)
- ABAP allows several development frameworks
 - ▶ Customers write their own code in order to adapt the standard to their needs („customizing“)
 - ▶ Custom development for non-standard business processes
 - ▶ 3rd party add-ons

Frontend-Technologies

- Dynpro
 - ▶ Written in ABAP
 - ▶ Requires proprietary UI (SAP GUI)
 - ▶ Similar to X11 paradigm

- Internet Transaction Server (ITS)
 - ▶ 1st Web-Technologie of SAP
 - ▶ Development almost stopped, but widely used

Frontend-Technologies

- Business Server Pages (BSP)
 - ▶ HTML with embedded ABAP (similar to JSP)
 - ▶ Several programming paradigms incl. MVC
 - ▶ Widely used, customers still build new applications

- Web Dynpro (ABAP | Java)
 - ▶ UI-independent framework, „point & click“ programming for UI design
 - ▶ Developer can't embed his own HTML/JavaScript
 - ▶ Developer can't cause a vulnerability. But he also can't avoid them

Frontend-Technologies

- Web GUI
 - ▶ HTML-version of regular Dynpros (SAP GUI)
 - ▶ Earlier version on top of Internet Transaction Server
 - ▶ Today as plugin of SAP Web Application Server

- ... external systems (via JCo or RFC), Adobe Flash, Microsoft Silverlight, PHP, Python, etc.

Further Technologies

- File access
- Database access (OpenSQL, Native SQL)
- Remote access
 - HTTP, FTP, Email, ...
 - Messaging (PI/XI)
 - Web Services (SOAP)
 - RFC - Remote Function Call

- Whatever you need – SAP has it, but be aware of the little differences

SAP Web Technology

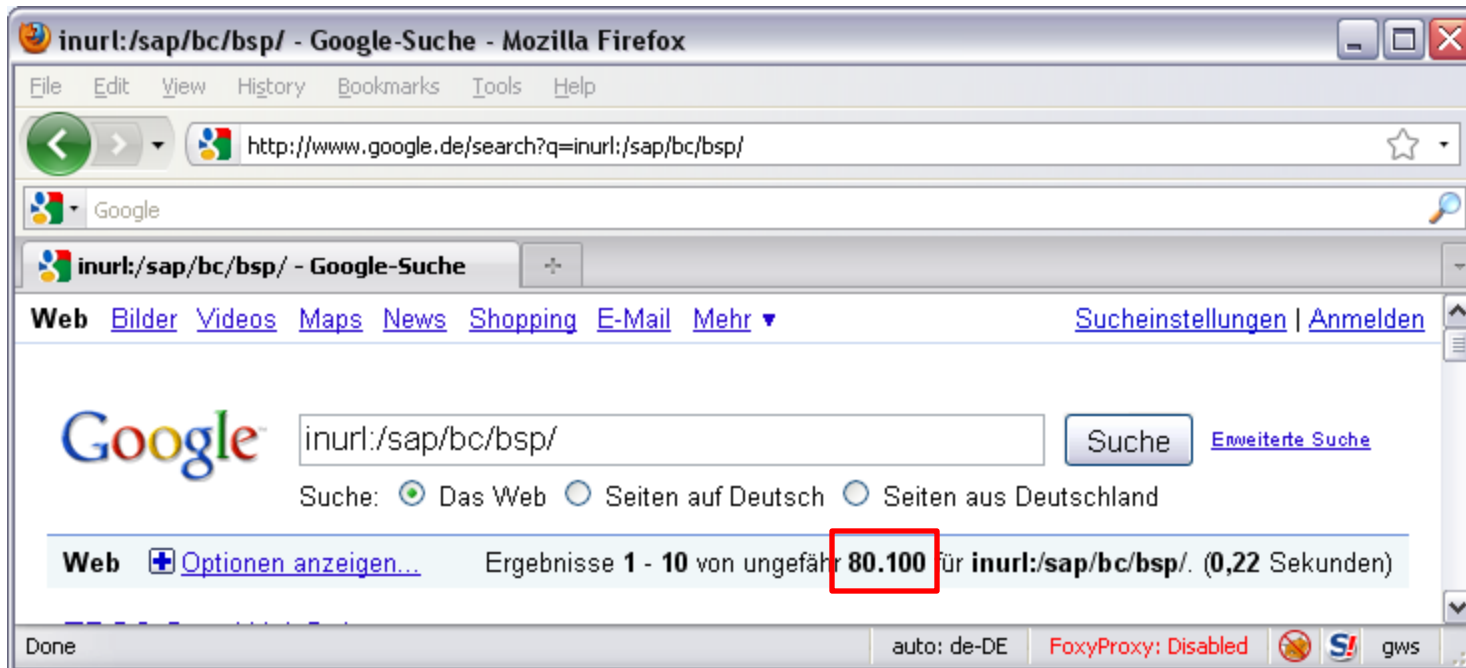
SAP NetWeaver Web Application Server (Web AS):

- Supports Single Sign On (SSO)
- SSO-ticket stored in cookie (MYSAPSSO2)
 - ▶ By default issued for path / and domain.tld
 - ▶ By default neither httpOnly, nor secure
- Development of your own HTTP-Handler possible
 - ▶ BSP, Web Dynpro, WebGUI are HTTP-Handler
- Configuration via profile parameter (*report* RZ11) and *transaction* SICF
- Blacklist implementation filters <script, %00 and other patterns

Business Server Pages (BSP)

Finding BSP applications:

- <http://www.google.de/search?q=inurl:/sap/bc/bsp/>



Business Server Pages (BSP)

- mentor.com
- erco.org
- sap-ag.de
- beiersdorfgroup.com
- mybayerjob.de
- heraeus.com
- wacker.com
- heidelberg.com
- knorr-bremse.com
- ottopersonalsysteme.de
- skyguide.ch
- eads.com
- bsr.de
- kuka.de
- kpmg.de
- daad.de
- euhreka.com
- umdasch.com
- celesio.com
- pflegedienst-navigator.de
- oebb.at
- salzburg-ag.at
- whirlpool.com
- volkswagen.de
- pharma.com
- wa.gov
- brucepower.com
- jetblue.com
- suzukiautoco.com
- singaporepower.com
- kaufland.de
- clavis-bonn.de
- albatha.ae
- vodafone.com
- iom.int
- wlw.de
- erecruiting-randstad.de
- lieferantensuchmaschine.com
- audi.de
- blanco.de
- festo.com
- vhv.de
- otto.de
- abb.de
- ruv.de
- holcim.com
- mannheim.de
- softsurvey.de

Business Server Pages (BSP)

OWASP Top 10	Potentially vulnerable?
A1 – Cross-Site Scripting (XSS)	Yes
A2 - Injection Flaws	Yes
A3 - Malicious File Execution	Yes
A4 - Insecure Direct Object Reference	Yes
A5 - Cross Site Request Forgery (CSRF)	Yes
A6 - Information Leakage and Improper Error Handling	n/a
A7 - Broken Authentication and Session Management	n/a
A8 - Insecure Cryptographic Storage	n/a
A9 - Insecure Communications	n/a
A10 - Failure to Restrict URL Access	n/a

Agenda

- ABAP development - risks in Web applications (example)
 - ABAP/BSP vs. OWASP Top 10
- Examples of vulnerabilities in custom coding
 - **Business Server Pages**
 - ▶ **Inline ABAP in HTML**
 - ▶ **HTMLB-Tag-Library**
 - Open SQL
 - ▶ Dynamic Open SQL
 - ▶ SQL-Injection
- Conclusion

Business Server Pages

Preventing Cross-Site Scripting by Encoding/Escaping

- in Plain-HTML-Pages
 - ▶ ABAP-Encoding-Functions (`CL_HTTP_UTILITY`)
 - ▶ BSP-Page Attribute (`forceEncode`)
- in Pages with HTMLB-Taglib
 - ▶ Tag-Attribute (`forceEncode`)

Business Server Pages – Plain HTML

```
1 <%@page language="abap" %>
2 <% DATA: name TYPE string.
3     name = request->get_form_field( 'name' ).
4 %>
5 <html>
6 <head><title>HTML mit eingebettetem ABAP</title></head>
7 <p>Hello <%= name %> </p>
8 <body>
9 </body></html>
```


Business Server Pages – Plain HTML

```
1 <html>
2   <head><title>HTML mit eingebettetem ABAP</title>
   </head>
3   <body>
4     <p>Hello Guest</p>
5   </body>
6 </html>
```

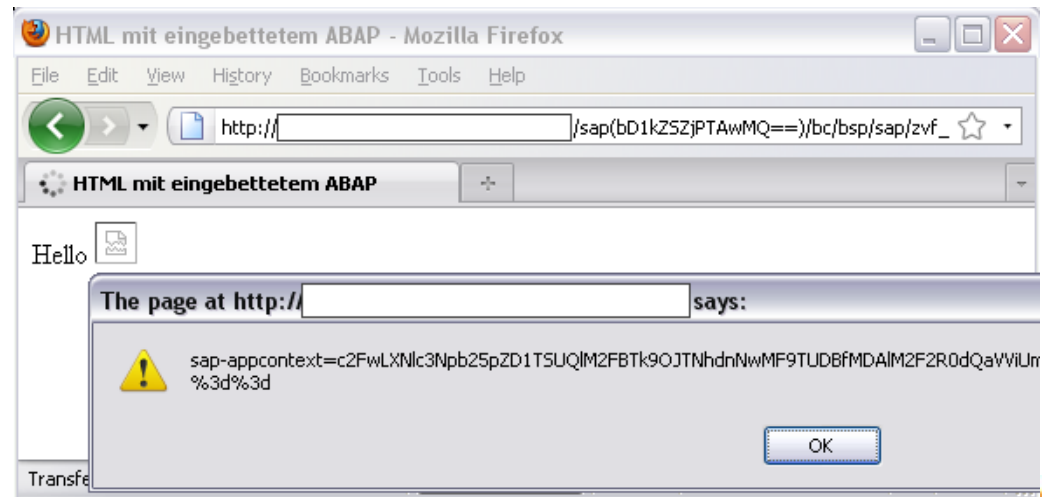


Business Server Pages – Plain HTML

Cross-Site Scripting Vulnerability:

```
http://.../example0.htm?name=
```

```
1 <html>
2 <head><title>HTML mit eingebettetem ABAP</title></head>
3 <p>Hello <img src= onerror="alert(document.cookie);"> </p>
4 <body>
5 </body></html>
```



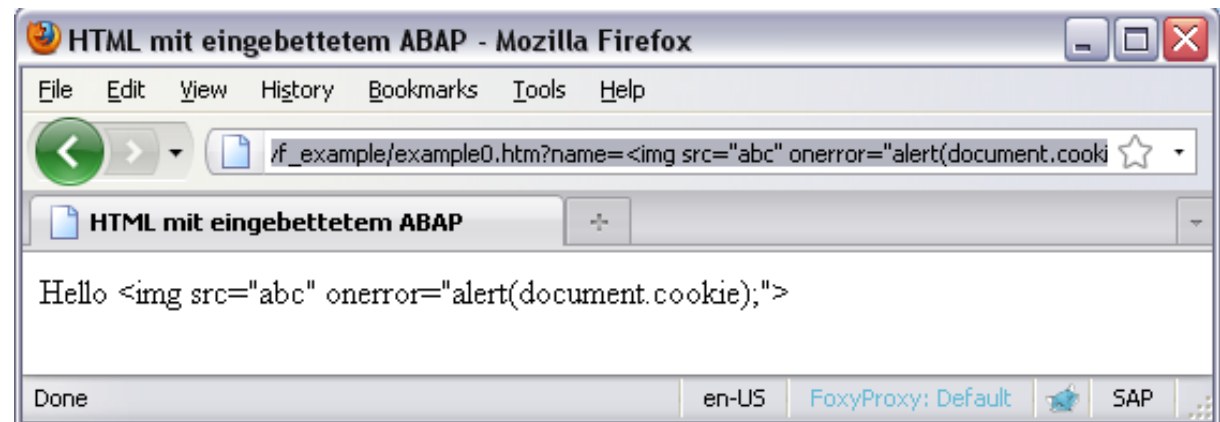
Business Server Pages – Plain HTML

```
1 <%@page language="abap" %>
2 <% DATA: name TYPE string.
3     name = request->get_form_field( 'name' ).
4     name = CL_HTTP_UTILITY=>escape_html( name ).
4 %>
5 <html>
6 <head><title>HTML mit eingebettetem ABAP</title></head>
7 <p>Hello <%= name %> </p>
8 <body>
9 </body></html>
```

Business Server Pages – Plain HTML

Prevented Cross-Site Scripting Schwachstelle:

- `http://.../example0.htm?name=`
- 1 `<html>`
 - 2 `<head><title>HTML mit eingebettetem ABAP</title></head>`
 - 3 `<p>Hello <img src="abc" onerror="alert(document.cookie);`
`"> </p>`
 - 4 `<body>`
 - 5 `</body></html>`



Business Server Pages – Plain HTML

- Encoding of data by
 - `CL_HTTP_UTILITY=>escape_html()`
 - `CL_HTTP_UTILITY=>escape_javascript()`
 - `CL_HTTP_UTILITY=>escape_url()`
 - `<%HTML=$VAR %>`, `<%URL= ... %>`, `<% ...`
- Pro:
 - ▶ Prevents XSS-Vulnerabilities
- Contra:
 - ▶ Every single output must be encoded according to the HTML-context
 - ▶ High effort, error prone

Business Server Pages – Plain HTML

```
1 <%@page language="abap" forceEncode="html"
2 DATA: name TYPE string.
3 name = request->get_form_field( 'name' ).
4 %>
5 <html>
6 <head><title></title></head>
7 <p>Hello <%= name %> </p>
8 <body>
9 </body></html>
```

Business Server Pages – Plain HTML

Preventing Cross-Site Scripting by

- `<%page forceEncode="{html|url|javascript}">`
- Global encoding via page attribute
- All output is encoded in the same way, no distinction between HTML-context (JavaScript, URL, ...)

Counterexample:

```
<a href="<%= request->get_form_field( 'user' ). %>">  
  Link</a>
```

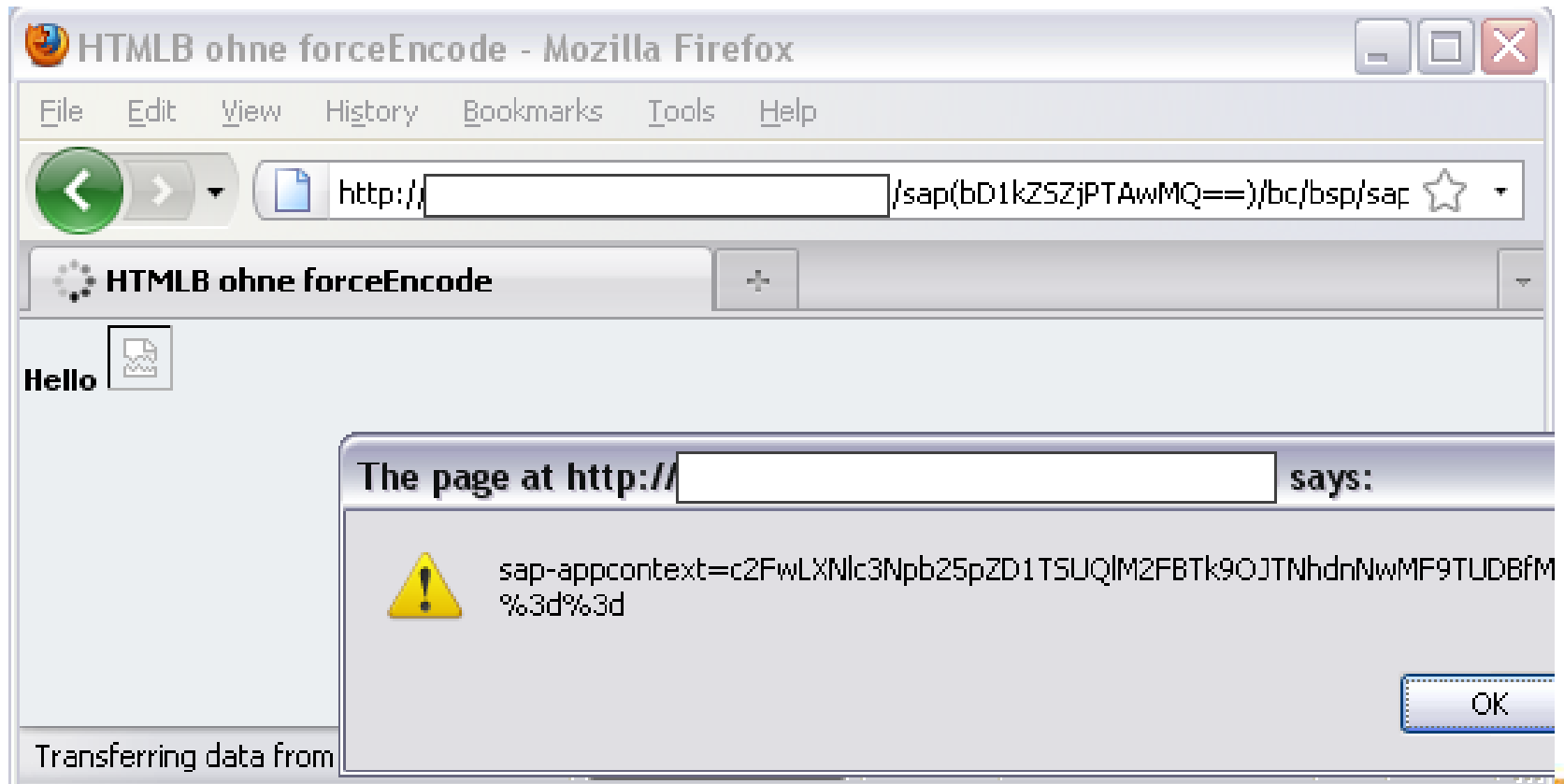
```
http://.../test.htm?user=javascript:document.write( ...
```

Business Server Pages – HTMLB

```
1  <%@page language="abap" %>
2  <%@extension name="htmlb" prefix="htmlb" %>
3  <% DATA: name TYPE string.
4     name = request->get_form_field( 'name' ). %>
5  <htmlb:content design="design2003">
6     <htmlb:page title = "HTMLB ohne forceEncode">
7         <htmlb:form>
8             <htmlb:textView      text      = "Hello <%= name %>"
9                                 design     = "EMPHASIZED" />
10        </htmlb:form>
11    </htmlb:page>
12 </htmlb:content>
```


Business Server Pages – HTMLB

- `name=`

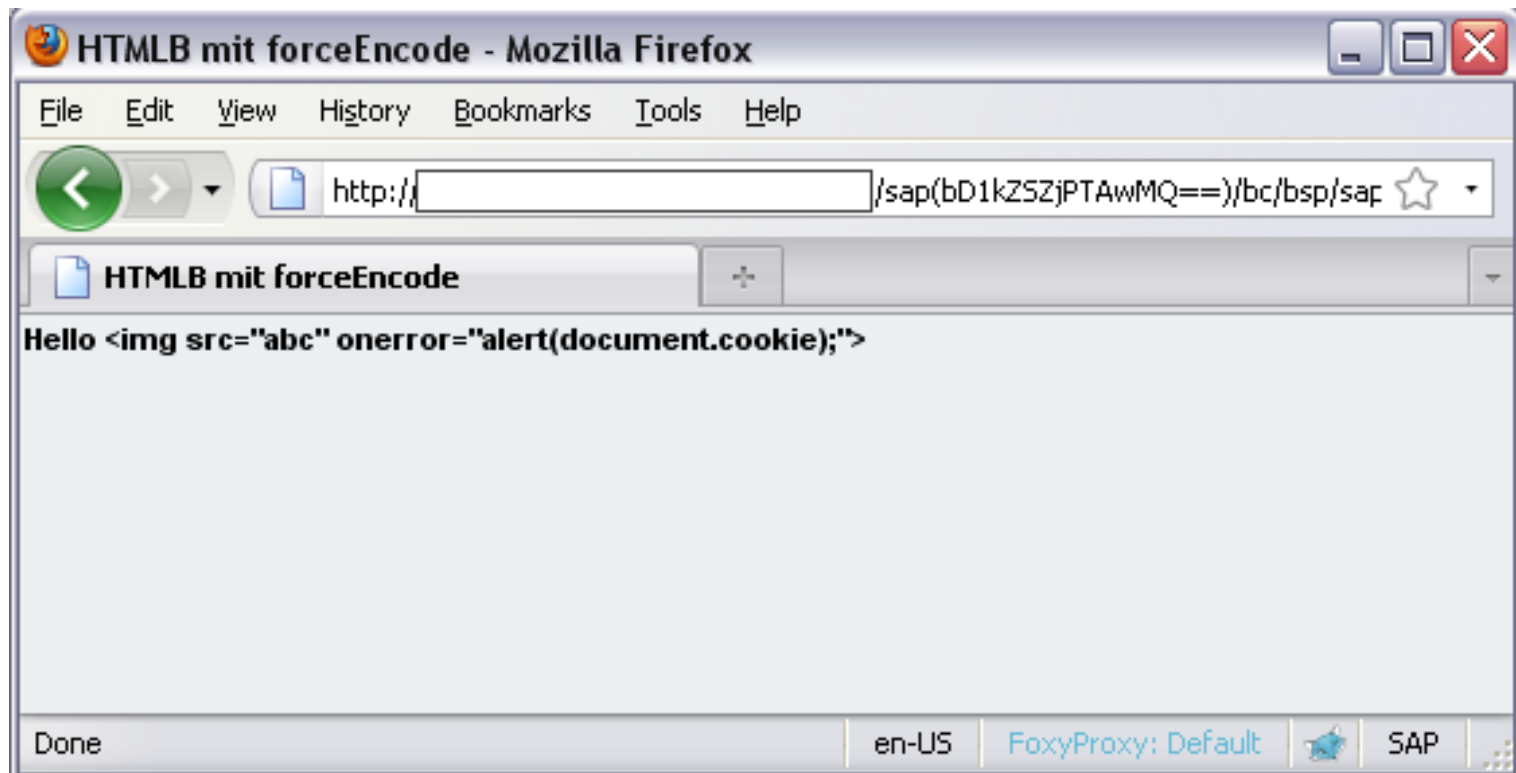


Business Server Pages – HTMLB

```
1  <%@page language="abap" %>
2  <%@extension name="htmlb" prefix="htmlb" %>
3  <% DATA: name TYPE string.
4     name = request->get_form_field( 'name' ). %>
5  <htmlb:content design="design2003" forceEncode="ENABLED">
6     <htmlb:page title = "HTMLB mit forceEncode">
7         <htmlb:form>
8             <htmlb:textView      text          = "Hello <%= name %>"
9                                 design        = "EMPHASIZED" />
10        </htmlb:form>
11    </htmlb:page>
12 </htmlb:content>
```

Business Server Pages – HTMLB

- `name=`
``



Business Server Pages

Preventing Cross-Site Scripting in Plain-HTML

- Encoding with methods (`CL_HTTP_UTILITY`)
 - ▶ High effort, error prone
- Encoding via page attribute (`forceEncode`)
 - ▶ Not specific for HTML-context, no complete coverage of attacks

Preventing Cross-Site Scripting in HTMLB

- Tag-attribute `forceEncode` per default deactivated, must be set explicitly

Agenda

- ABAP development - risks in Web applications (example)
 - ABAP/BSP vs. OWASP Top 10
- Examples of vulnerabilities in custom coding
 - Business Server Pages
 - ▶ Inline ABAP in HTML
 - ▶ HTMLB-Tag-Library
 - **Open SQL**
 - ▶ **Dynamic Open SQL**
 - ▶ **SQL-Injection**
- Conclusion

Open SQL

- Open SQL built in ABAP
- Internally converted to prepared statements
- SQL-statement and user data separated, no SQL-Injection possible

```
1  SELECT * FROM ZCCINFO
2  INTO l_zccinfo
3  WHERE uname = l_uname
4  AND ta_date = l_date.
```

Dynamic Open SQL - Example

Show your recent transactions - Mozilla Firefox

File Edit View History Bookmarks Tools Help

virtualforge.de:8000/sap(bD1kZ5ZjPTAwMQ==)/bc/bsp/sap/zvf_ccdata_demo/details.htm?input_year=2008&input_month=00

Show your recent transactions

Please select year and month to filter your transactions:

Year 2008 Month All Submit

Results:

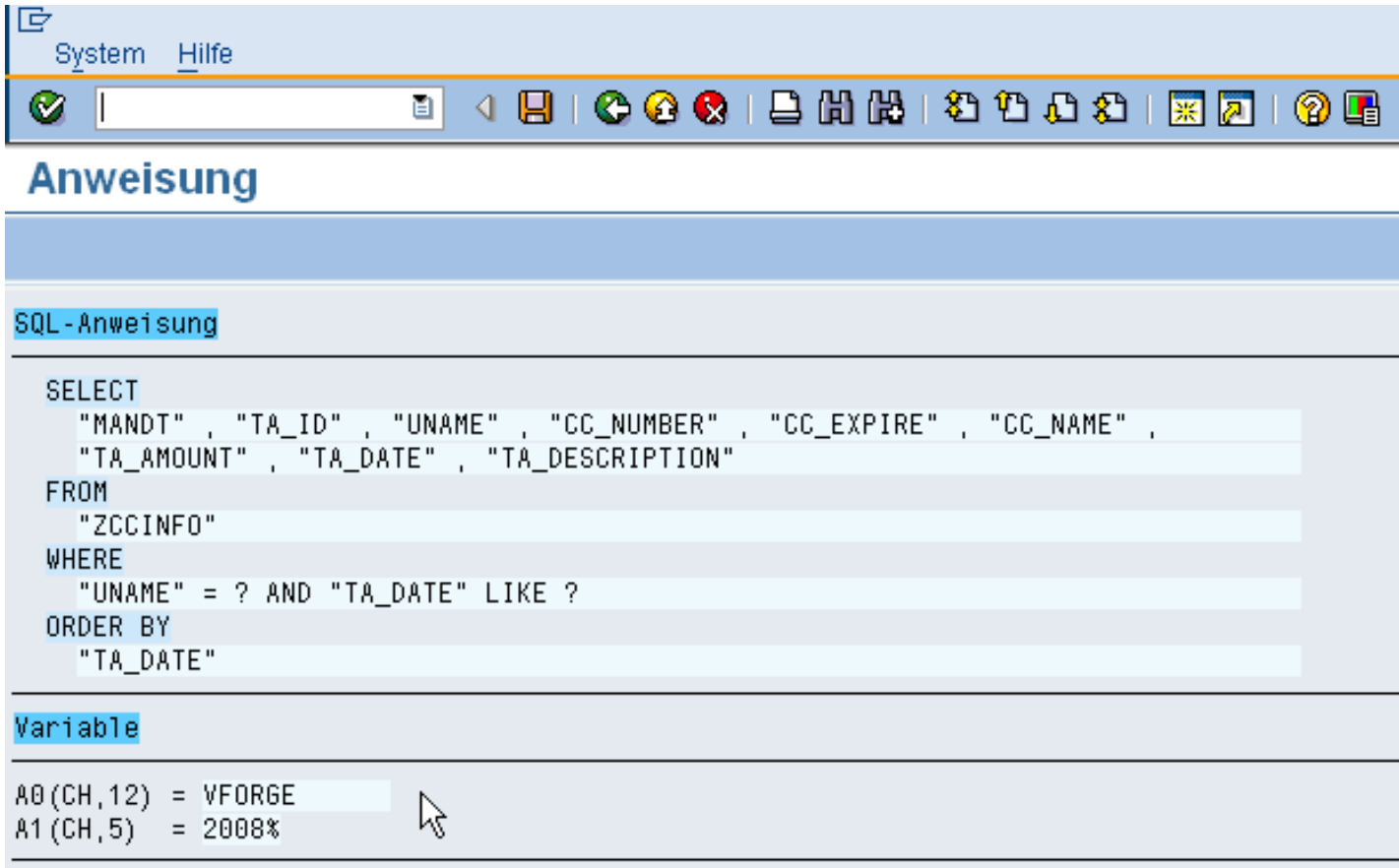
Your recent credit card transactions

CC_NUMBER	CC_EXPIRE	CC_NAME	TA_DESCRIPTION	TA_AMOUNT	TA_DATE
4149253627344523	11.09	PETER JACKSON	DRUG STORE	65,00	15.01.2008
4149253627344523	11.09	PETER JACKSON	PLANE TICKET	80,00	19.01.2008
4149253627344523	11.09	PETER JACKSON	THANK YOU FOR SHOPPING AT WALMART	180,00	08.02.2008
4149253627344523	11.09	PETER JACKSON	BMW OSTER SAYS THANK YOU FOR YOUR VISIT	2870,00	12.02.2008
4149253627344523	11.09	PETER JACKSON	CRAFTSMAN BILL	420,00	14.03.2008
4149253627344523	11.09	PETER JACKSON	LACOSTE STORE SAYS THANK YOU	360,00	27.03.2008
4149253627344523	11.09	PETER JACKSON	BAKERY	78,00	07.04.2008
4149253627344523	11.09	PETER JACKSON	FLEUROP FLOWER ORDERING	90,00	10.04.2008
4149253627344523	11.09	PETER JACKSON	AMAZON SAYS THANK YOU FOR PURCHASING A DVD PLAYER	1234,00	18.04.2008
4149253627344523	11.09	PETER JACKSON	OPTICIAN	230,00	04.05.2008

Seite 1 von 1

Done en-US FoxyProxy: Default SAP

Open SQL



The screenshot shows the SAP Open SQL interface. At the top, there is a menu bar with 'System' and 'Hilfe'. Below it is a toolbar with various icons for file operations and help. The main area is titled 'Anweisung' and contains a text editor with the following SQL query:

```
SELECT
  "MANDT" , "TA_ID" , "UNAME" , "CC_NUMBER" , "CC_EXPIRE" , "CC_NAME" ,
  "TA_AMOUNT" , "TA_DATE" , "TA_DESCRIPTION"
FROM
  "ZCCINFO"
WHERE
  "UNAME" = ? AND "TA_DATE" LIKE ?
ORDER BY
  "TA_DATE"
```

Below the query, there is a section titled 'Variable' with the following definitions:

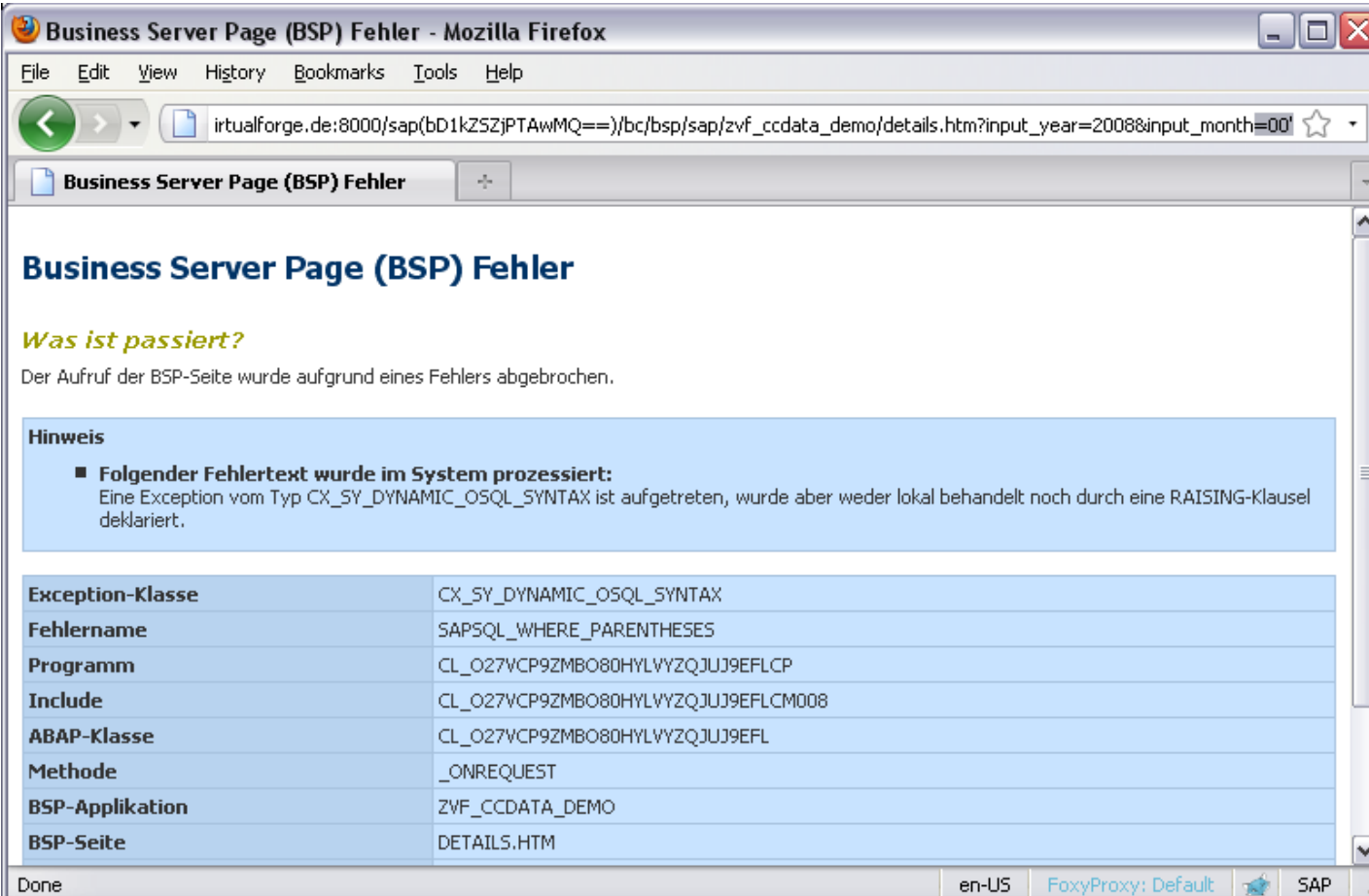
```
A0 (CH,12) = VFORGE
A1 (CH,5) = 2008%
```


Dynamic Open SQL

- Interprets String literal as SQL-Statement
- No encoding functions
 - ▶ User data can't be separated from SQL-commands
 - ▶ SQL-Injection very likely, when user data is part of dynamic SQL-Statement

```
1 SELECT (l_felder) FROM (l_table)
2 INTO l_zccinfo
3 WHERE (l_where).
```

Dynamic Open SQL - Example



Business Server Page (BSP) Fehler

Was ist passiert?

Der Aufruf der BSP-Seite wurde aufgrund eines Fehlers abgebrochen.

Hinweis

- **Folgender Fehlertext wurde im System prozessiert:**
Eine Exception vom Typ CX_SY_DYNAMIC_OSQ_L_SYNTAX ist aufgetreten, wurde aber weder lokal behandelt noch durch eine RAISING-Klausel deklariert.

Exception-Klasse	CX_SY_DYNAMIC_OSQ_L_SYNTAX
Fehlername	SAPSQL_WHERE_PARENTHESES
Programm	CL_O27VCP9ZMBO80HYLVY2QJUJ9EFLCP
Include	CL_O27VCP9ZMBO80HYLVY2QJUJ9EFLCM008
ABAP-Klasse	CL_O27VCP9ZMBO80HYLVY2QJUJ9EFL
Methode	_ONREQUEST
BSP-Applikation	ZVF_CCDATA_DEMO
BSP-Seite	DETAILS.HTM

Dynamic Open SQL - Example

Browser window: Show your recent transactions - Mozilla Firefox

Address bar: 0/sap(bD1kZ5ZjPTAwMQ==)/bc/bsp/sap/zvf_ccdata_demo/details.htm?input_year=2008&input_month=00'OR mandt LIKE '%

Form: Please select year and month to filter your transactions:
 Year: 2008 | Month: All | Submit

Results:

CC_NUMBER	CC_EXPIRE	CC_NAME	TA_DESCRIPTION	TA_AMOUNT	TA_DATE
4130865326190251	11/10	SARAH BLANKS	THE DRUGSTORE SAYS THANK YOU	24,00	03.01.2008
2810491793517364	02/08	LINDA OKE	IMAGE MODE SAYS THANK YOU	45,00	04.01.2008
2810418361048230	10/08	MAILING	THAI STORE	120,00	06.01.2008
5110282427041850	12/12	ABRAHAM SMITH	HOSPITAL ACCOUNT	490,00	07.01.2008
5067392017603901	08/10	PETER MARSHALL	FODYS SAYS THANK YOU	89,00	07.01.2008
3104827308174926	06/09	LISA IVKIC	THE FITNESS COMPANY SAYS THANK YOU FOR YOUR VISIT	45,00	07.01.2008
3106916205915295	05/10	BEN LANDS	HAIRDRESSER	120,00	07.01.2008
4182018462916318	07/12	TONI STANSFIELD	DELL SAYS THANK YOU FOR YOUR ORDER	1400,00	11.01.2008
6206194916490153	04/12	PIERRE CARDIN	CRAFTSMAN BILL	4980,00	12.01.2008
5194028301639618	10/09	DAVID GARNER	FLEUROP FLOWER ORDERING	50,00	12.01.2008
5104784134789216	10/11	JOHN DOE	DENTAL BILL	1750,00	12.01.2008
4182018462916318	07/12	TONI STANSFIELD	STARBUCKS FRANKFURT	17,00	13.01.2008

Browser status bar: Done | en-US | FoxyProxy: Default | SAP

Dynamisches Open SQL

System Hilfe

Anweisung

SQL-Anweisung

```
SELECT
  "MANDT" , "TA_ID" , "UNAME" , "CC_NUMBER" , "CC_EXPIRE" , "CC_NAME" ,
  "TA_AMOUNT" , "TA_DATE" , "TA_DESCRIPTION"
FROM
  "ZCCINFO"
WHERE
  "UNAME" = ? AND "TA_DATE" LIKE ? OR "MANDT" LIKE ?
ORDER BY
  "TA_DATE"
```

Variable

```
A0(CH,12) = VFORGE
A1(CH,4)   = 2008
A2(CH,4)   = 001%
```

Summary Open SQL

- Dynamic Open SQL can easily lead to SQL-Injection-Vulnerabilities
 - ▶ No encoding functions
 - ▶ Prepared-Statement-Injection
- Avoid dynamic Open SQL in ABAP whenever possible

Conclusion: Covered Topics

OWASP Top 10	TODO
A1 - Cross Site Scripting (XSS)	✓
A2 - Injection Flaws	✓
A3 - Malicious File Execution	X
A4 - Insecure Direct Object Reference	X
A5 - Cross Site Request Forgery (CSRF)	X
A6 - Information Leakage and Improper Error Handling	-
A7 - Broken Authentication and Session Management	-
A8 - Insecure Cryptographic Storage	-
A9 - Insecure Communications	-
A10 - Failure to Restrict URL Access	-

Conclusion: Take Aways

- SAP-Web-Frontends as example
 - Widely used, processing of business-critical data
- SAP-Web-Frontend-Technologies covered in this talk:
 - ▶ **Business Server Pages (BSP)**
 - ▶ Web Dynpro
 - ▶ Internet Transaction Server
 - ▶ Own HTTP-Handlers
 - ▶ ...
- High efforts for writing secure ABAP code!
 - Step 1: Understand how known vulnerabilities relate to SAP
 - Step 2: Understand what to do

Questions



Literature

- **“Sichere ABAP-Programmierung” - SAP Press, 2009**
Wiegenstein, Schumacher, Schinzel, Weidemann
<http://www.sap-press.de/2037>
- **“ SAP Documentation”** - <http://help.sap.com/>
- **“Secure Programming – ABAP” - SAP AG, 2004**
<http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/17a4f828-0b01-0010-8da1-d18bb60ec2bf&overridelayout=true>
- **“Security Scanner for ABAP”** - <http://codeprofilers.com/>
- **“vMovie: Security Knowledge on Stage”**
<http://secure-abap.de/media>