Oops... Code Execution and Content Spoofing: The First Comprehensive Analysis of OpenDocument Signatures

Simon Rohlmann, Christian Mainka, Vladislav Mladenov, and Jörg Schwenk, *Ruhr University Bochum*

https://www.usenix.org/conference/usenixsecurity22/presentation/rohlmann

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

A Artifact Appendix

A.1 Abstract

The artifact evaluation consists of two parts: (1) evaluation of attacks on signed OpenDocument Format (ODF) documents and (2) evaluation of Document Signature Validator (DocSV).

To evaluate the attacks on signed ODF documents, we provide multiple proof of concept (PoC) files. By opening the PoC files the reviewer can evaluate the success of the attack. This success is either code execution via signed macros, content spoofing, or timestamp manipulation. To evaluate all attack classes, at least one Windows 10 system with the affected ODF applications is required. Optionally, a macOS, Linux, iOS, and an Android system is required to evaluate the artifacts under all analyzed ODF applications.

The second part of the artifact evaluation focuses on DocSV – our tool is capable to evaluate the signature status of signed documents in ODF, Office Open XML (OOXML), and Portable Document Format (PDF) formats. We provide a collection of test documents that can be used by the reviewers. Both the source code and the compiled executable are provided for this purpose. For the evaluation of the DocSV tool, a Windows 10 system with the respective ODF, OOXML and PDF applications to be tested is required.

A.2 Artifact check-list (meta-information)

- ☑ Run-time environment
  - Required: Windows 10
  - Optional: macOS Catalina, Ubuntu 20.04.3 LTS, iOS 15, and Android 10
- ☑ Software Installation
  - ODF office applications (see Section A.3.1)
  - DocSV executable (see Section A.3.1)
- ☑ Resources for the Evaluation
  - How much disk space required – approx. 50 GB
  - How much time is needed to prepare workflow – approx. 2h
  - How much time is needed to complete experiments – approx. 1h
- ☑ Code licenses (if publicly available)?: trial licenses are sufficient

A.3 Description

A.3.1 How to access

Access to the Vulnerable Applications Below we have listed links to the installation files of various ODF applications. Please note that not all ODF applications are freely available in the vulnerable version.

- Apache OpenOffice: choose version 4.1.8 here.
- IBM Lotus Symphony 3.0.1: here, fp2-Update: here.
- LibreOffice 7.0.4.2: here.
- Microsoft Office 2019: Microsoft Office 2019 can be downloaded as a trial version here.
- Collabora Online (CODE) 6.0-18: Virtual machine (VM) images for the online variant of Collabora are available here.

Artifacts Stable URL to all artifacts: `https://github.com/RUB-NDS/DocumentSignatureValidator/releases/tag/Artifact_Evaluation`

- PoC files for all attacks described in the paper are available here.
- DocSV source code and a compiled executable can be downloaded on GitHub.
- A collection with test files for DocSV is available here.

A.3.2 Hardware dependencies

N/A

A.3.3 Software dependencies

To evaluate all attack classes, at least a Windows 10 system with the vulnerable ODF applications is required. Optionally, a macOS Catalina, Ubuntu 20.04.3 LTS, iOS 15 and Android 10 system is required to be able to evaluate the artifacts under all analyzed ODF applications.

Due to their same code base, LibreOffice and Collabora Office cannot be installed on Windows at the same time.

For OpenOffice on macOS, the Mozilla Certificate Store must be associated to properly validate the trusted entity certificate (see configuration tutorial).

A.3.4 Data sets

N/A

A.3.5 Models

N/A

A.3.6 Security, privacy, and ethical concerns

N/A

A.4 Installation

ODF Signature Attacks All ODF applications (see Section A.3.1) need to be installed on a Windows 10 VM or on a physical machine. Optionally, the same for macOS Catalina, Ubuntu 20.04.3 LTS, iOS 15 and Android 10. To evaluate the PoC files on Collabora Online, run the prepared VM on VMware or VirtualBox and follow the instructions.

DocSV To evaluate DocSV, no installation is required, just run `DocumentSignatureValidator.exe`. In addition to ODF applications, DocSV can also be evaluated with PDF applications (Adobe Acrobat Reader DC and Foxit PDF Reader), as well as with OOXML applications (Microsoft Office).

Installation steps of Collabora Online (CODE) VM

1. Import the VM Image.
2. Change the network adapter to NAT in your virtualization software and add a port forwarding rule to forward port 443 to the VM.
3. Start the VM.
4. Enter a City.
5. Use the IP provided by DHCP or one of your own.
6. Select `Manage users and permissions directly on the system`.
7. Enter `test` as organization and add a mail address.
8. Choose a password.
9. Set `code.test.local` as FQDN.
10. Add `code.test.local` with IP `127.0.0.1` to the `etc/hosts` file of the operating system.
11. After the setup is finished, open `https://code.test.local` via the browser and register for a free license.
12. After that you can upload the ODF documents via `https://code.test.local/nextcloud/login`. Login: User name `Administrator`, Password: which was assigned during the installation.

A.5 Experiment workflow

Ground Truth First, the reviewers can see different documents that are: not signed, signed with a trusted key, signed with an untrusted key, and manipulated by invalidating the signature.

Configuration We need to trust the certificate of the `trusted.person.odf@gmail.com` and allow the execution of macros for this person. For this purpose, open the file `ODF_macro_signature_valid_and_trusted.odt`. A dialog then opens asking whether the creator `trusted.person.odf@gmail.com` should be trusted. To do this, check the "Always trust macros from this source" box and then press the "Enable Macros" button.

DocSV: Usage and Configuration DocSV uses XML configuration files for execution. Sample configuration files for evaluating various applications are available here.

- Input Files: In `<files><path></path></files>` users can specify the directory of the documents that will be analyzed. Test documents with different signature statuses and formats are available here.
- Results: The directory for saving the screenshots created during the check, as well as the `CSV` report are specified in `<output><path></path></output>`.
- Detection Rules: In `<sigvalidstring/>`, `<siginvalidstring/>` and `<sigproblem/>` the detection rules for each application can be specified. Note that the default configuration is dependent on the language. To avoid false results, use the English version of the office applications.
- Timeout: DocSV analyzes the application's process memory. To guarantee that the analyzed document is fully loaded into the memory, users can configure a timeout in `<wait/>`. Users with limited PC resources are encouraged to increase the timeout.

DocSV requires one parameter as input – the configuration file:

```
DocumentSignatureValidator.exe config_file_examples/
config_LibreOffice.xml
```

Prepare Foxit and Adobe for DocSV Foxit and Adobe use their own certificate stores, so unknown signer certificates must first be set up as trusted.

Adobe:

1. Open PDF test file containing a valid signature.pdf with Adobe Acrobat.
2. Open the Signature Panel.
3. Click on the arrow of the `Rev. 1...` signature.
4. Open `Signature Details`→`Certificate Details...`
5. Click on `Trust`→`Add to Trusted Certificates...` to trust the certificate.

Foxit:

1. Open PDF test file containing a valid signature.pdf with Foxit.
2. Open the Siganture Panel (left side `Manage digital signatures`).
3. Right click on the signature `Rev. 1...`→`Show Signature Properties`.
4. Click on `Show Certificate...`
5. Click on `Trust`→`Add to Trusted Certificates` to trust the certificate.

DocSV must be added to Foxit as a trusted app. Click on `File`→`Preferences`→`Trust Manager`→`Open Foxit PDF Reader from applications without valid digital signatures`→`Change Settings...` add the path to the DocSV .exe and click `Allow`.

A.6 Evaluation and expected results

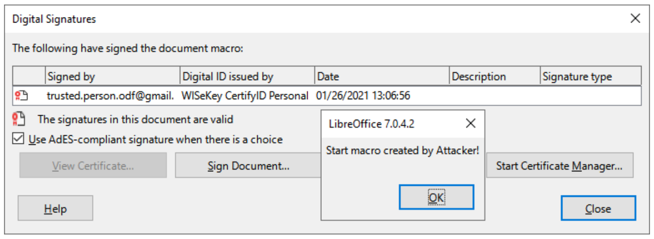
A.6.1 ODF Signature Attacks

All five attacks described in the paper can be evaluated. The PoC files are sorted by the vulnerable ODF applications for this purpose. For each attack class, two ODF documents are included, as well as two folders. Files starting with `01_` represent the initial document. Files starting with `02_` represent the documents manipulated by the attacker. The corresponding directories contain the unzipped ODF file.

01: Macro Manipulation with Certificate Doubling The attacker signs the document with its own key and thus can choose the macro code. The public key of a trusted entity (e.g., `trusted.person.odf@gmail.com`) is included to mask the document as trustful.

☑ **Execution**: Open the document `02_doc_macros_signed_by_attacker_manipulated.odt`.

☑ **Expected Result**: After opening the document, macro code is automatically executed which opens a simple message box. In `Tools→Macros→Digital Signatures...`, the trusted entity `trusted.person.odf@gmail.com` is displayed to the victim as the signer, even though the signature was created by the attacker.


Digital Signatures

The following have signed the document macro:

Signed by	Digital ID issued by	Date	Description	Signature type
trusted.person.odf@gmail.	WISeKey CertifyID Personal	01/26/2021 13:06:56		

The signatures in this document are valid

Use AdES-compliant signature when there is a choice

LibreOffice 7.0.4.2

Start macro created by Attacker!

Figure 1: Expected result for `01: Macro Manipulation with Certificate Doubling`

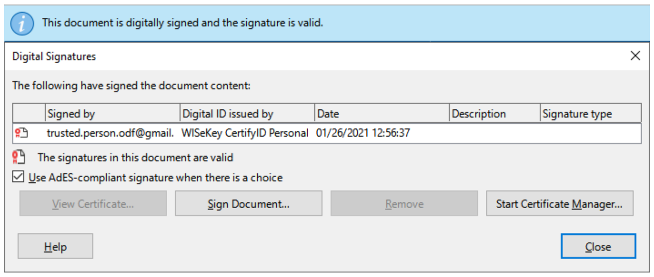
02: Content Manipulation with Certificate Doubling The attacker signs the document with its own key and thus can choose the content of the document. The public key of a trusted entity (e.g., `trusted.person.odf@gmail.com`) is included to mask the document as trustful.

☑ **Execution**: Open the document `02_doc_signed_by_attacker_manipulated.odt`.

☑ **Expected Result**: After opening the document, a valid and trusted document signature is displayed to the victim. Under `File→Digital Signatures→Digital Signatures...`, the trusted entity `trusted.person.odf@gmail.com` is displayed to the victim as the signer, even though the signature was created by the attacker.

03: Content Manipulation with Certificate Validation Bypass The attacker signs the document with its own key and thus can choose the content of the document. The attacker disables the verification of the certificate chain and successfully masks the document as trustful.

☑ **Execution**: Open the document `02_doc_signed_by_attacker_manipulated.odt`.


This document is digitally signed and the signature is valid.

Digital Signatures

The following have signed the document content:

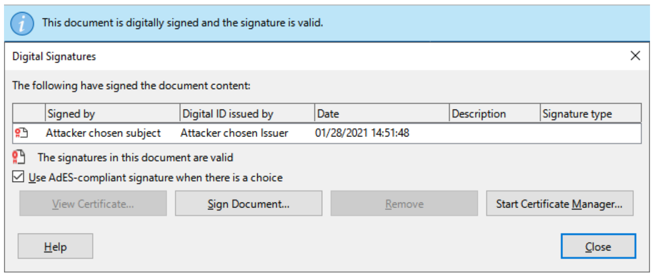
Signed by	Digital ID issued by	Date	Description	Signature type
trusted.person.odf@gmail.	WISeKey CertifyID Personal	01/26/2021 12:56:37		

The signatures in this document are valid

Use AdES-compliant signature when there is a choice

Figure 2: Expected result for `02: Content Manipulation with Certificate Doubling`

☑ **Expected Result**: After opening the document, a valid and trusted document signature is displayed to the victim. Under `File→Digital Signatures→Digital Signatures...`, a trusted entity arbitrarily chosen by the attacker is displayed to the victim as the signer, even though the signature was created by the attacker.


This document is digitally signed and the signature is valid.

Digital Signatures

The following have signed the document content:

Signed by	Digital ID issued by	Date	Description	Signature type
Attacker chosen subject	Attacker chosen Issuer	01/28/2021 14:51:48		

The signatures in this document are valid

Use AdES-compliant signature when there is a choice

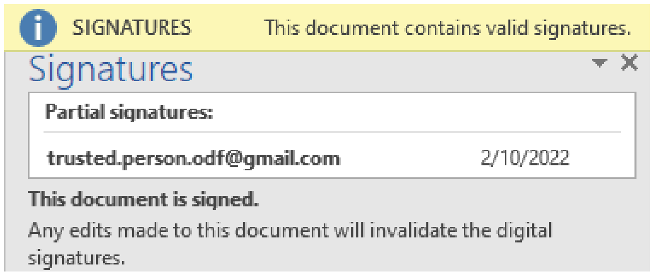
Figure 3: Expected result for `03: Content Manipulation with Certificate Validation Bypass`

04: Content Manipulation with Signature Upgrade The attacker possess an ODF with signed macros that is created by trusted entity. The attacker abuses the partial coverage of the digital signatures and manipulates the content of the document directly due to the missing integrity protection. Thus, the attacker can choose the content of the document arbitrarily.

☑ **Execution**: Open the document `02_doc_macros_signed_by_trusted_person_manipulated.odt` with Microsoft Office 2019.

☑ **Expected Result**: After opening the document, a valid and trusted document signature is displayed to the victim. Under `File→View Signatures`, the trusted entity `trusted.person.odf@gmail.com` is displayed to the victim as the signer.

05: Timestamp Manipulation with Signature Wrapping The attacker possess an ODF with signed content that is created by trusted entity. The attacker applies an XML Signature



SIGNATURES This document contains valid signatures.

Signatures

Partial signatures:

trusted.person.odf@gmail.com 2/10/2022

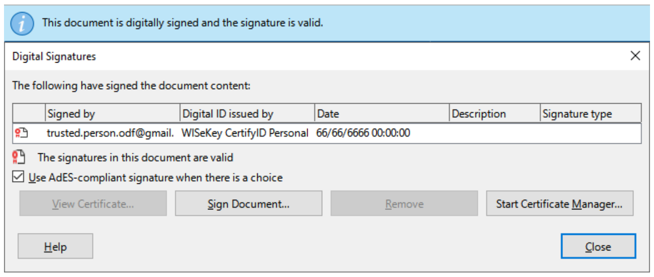
This document is signed.

Any edits made to this document will invalidate the digital signatures.

Figure 4: Expected result for `04: Content Manipulation with Signature Upgrade`

Wrapping attack. As a result the attacker can choose any timestamp for the signature.

- ☑ **Execution**: Open the document `02_doc_signed_by_trusted_person_manipulated.odt`.
- ☑ **Expected Result**: After opening the document, a valid and trusted document signature is displayed to the victim. Under `File→Digital Signatures→Digital Signatures...`, the trusted entity `trusted.person.odf@gmail.com` is displayed to the victim as the signer with the attacker's chosen timestamp `66/66/6666 00:00:00`.



This document is digitally signed and the signature is valid.

Digital Signatures

The following have signed the document content:

	Signed by	Digital ID issued by	Date	Description	Signature type
	trusted.person.odf@gmail.	WISeKey CertifyID Personal	66/66/6666 00:00:00		

The signatures in this document are valid

☑ Use AdES-compliant signature when there is a choice

Figure 5: Expected result for `05: Timestamp Manipulation with Signature Wrapping`

Further Macro Exploit Examples If the reviewers are interested in more powerful exploits, we created additional examples. These PoC files contain two more variants of the attack class `01` and are specially designed for Windows.

First variant `exe_download_execute`: The included macro downloads an `.exe` file from `https://github.com/attodf/odf-test` when the document is opened and saves it to `C\Users\%USERNAME%\AppData\Local\Temp`, then automatically executes the program. The program is harmless and does not contain any malicious code. It just outputs a text on the console. A working Internet connection is required for this variant.

Second variant `ransomware`: The included macro creates the file `example_ransomware.py` under `C:\Users\%USERNAME%\AppData\Local\Temp`. Then, this Python script is executed, using the Python environment of the respective office application, which can be found under `C:\Program Files\%ODF-Application%\program\python.exe`. This ransomware simulation serves as a PoC and is not supposed to do any damage, so it only creates a hashed file with `.hashed` extension from each file under `C:\Users\%USERNAME%\Desktop`. The function to delete the original files is not active in the Python code.

A.6.2 DocSV

DocSV can be used to check the signature status of signed documents in ODF, PDF and OOXML formats. DocSV is started by `DocumentSignatureValidator.exe` via the console. The configuration is done using an XML configuration file which must be passed as argument (see Section A.5). DocSV automatically opens the individual signed documents and determines the signature status through a memory analysis. The analysis results are exported as a `CSV` file. In addition, a screenshot of the opened document is also saved.

- ☑ **Execution:** Start DocSV and pass one of the prepared configuration files. DocSV will test the files stored in the folder test_documents. As part of the artifact evaluation, you can generate your own files and test these with DocSV.
- ☑ **Expected Result:** DocSV produces a report containing the result of the analysis and saves this report together with the taken screenshots in results_dovsv. Our collection of test documents contains one unsigned, signed, and manipulated document. DocSV should detect these correctly.

A.7 Experiment customization

N/A

A.8 Notes

The certificate of the trusted entity is valid until 11st May 2022. For the evaluation of the attacks after this date, it is necessary to reset the date of the operating system accordingly.

A.9 Version

Based on the LaTeX template for Artifact Evaluation V20220119.