USENIX'23 Artifact Appendix: Every Signature is Broken: On the Insecurity of Microsoft Office's OOXML Signatures

Simon Rohlmann
Ruhr University Bochum

Vladislav Mladenov
Ruhr University Bochum

Christian Mainka
Ruhr University Bochum

Daniel Hirschberger
Ruhr University Bochum

Jörg Schwenk
Ruhr University Bochum

A Artifact Appendix

A.1 Abstract

In our paper "Every Signature is Broken: On the Insecurity of Microsoft Office's OOXML Signatures", we present seven attacks, divided into five attack classes, on signed OOXML Word documents. The goal of each attack is to manipulate the displayed document content without invalidating the signature.

The attack classes Content Injection Attack (CIA), Content Masking Attack (CMA) and Legacy Wrapping Attack (LWA) are based on specification flaws that allow the attacker to correctly reference subsequently added files within the OOXML package. The root problem here consists of only partially signed relationship files. The Universal Signature Forgery (USF) and Malicious Repair Attack (MRA) attack classes exploit implementation flaws in the corresponding OOXML application. The reviewers can test the attacks under different Microsoft Office and OnlyOffice desktop versions during the evaluation. Since different versions of Microsoft Office cannot be installed simultaneously under one operating system, the applications must be installed individually. A bare-metal solution with macOS Monterey is required for the macOS versions of Microsoft Office and OnlyOffice Desktop.

We provide proof of concept (PoC) files for each attack to the reviewers. Here, the document numbered *01* is the source document, *02* is the first manipulation step (intermediate step), and *03* is the ready-to-run exploit. There is one exception for CIA, where the document *02* corresponds to the exploit.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The attacks are aimed at manipulating signed OOXML Word documents. The documents do not contain any macro code or similar that could influence the operating system. We have reported all vulnerabilities found during our investigations to Microsoft, OnlyOffice, as well as to the responsible standardization committee ISO/IEC JTC 1/SC 34. The vulnerabilities have been acknowledged by Microsoft. However, Microsoft has decided that the vulnerabilities do not require immediate attention. According to Microsoft, a potential fix in the future is not excluded. OnlyOffice has announced they are working on a fix.

A.2.2 How to access

- Stable URL: Link
- PoC files: Link
- To download Microsoft Office 2019, 2021 and 365, the Office Deployment Tool must be used with the appropriate configuration file:
  - Office Deployment Tool: Link
  - Configurations files: Link
- Microsoft Office 2016: Link
- Microsoft Office 2013: Link
- Microsoft Office for macOS 2019/2021/365: Link
- OnlyOffice Desktop: Link
- 7-Zip: Link
- Notepad++: Link

A.2.3 Hardware dependencies

A computer running Windows 10 and Ubuntu 22.04.1 or an equivalent Virtual Machine (VM) is required.

An Apple hardware running macOS Monterey is required to evaluate the attacks against Microsoft Office and OnlyOffice for macOS.

A.2.4 Software dependencies

Possibility to install the office packages.

A.2.5 Benchmarks

None

A.3 Set-up

A.3.1 Installation

The packages for OnlyOffice and Microsoft Office 2013, 2016, as well as the macOS versions of Microsoft Office can be downloaded and installed directly. For Microsoft Office 2019, 2021 and 365, the Office Deployment Tool must be installed. After installation, *setup.exe* must be started from the console and the configuration file must be passed as an argument: *setup.exe /configure config-file.xml*

Since the certificate used to sign the PoC files was only valid for three months and expired on September 5th, 2022, the host system date must be set accordingly to a date between June 7th and September 5th, 2022. If the date is not corrected accordingly, a recoverable signature is displayed in Microsoft Word instead of a valid document signature.

A.3.2 Basic Test

From the PoCs folder, open the file */5-1_CIA/01_document_signed_by_trusted_entity.docx*. Microsoft Office will display a valid document signature on opening the document if everything is set up correctly. If a recoverable signature is displayed, the date was not reset correctly or there is no internet connection available to verify the certificate.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): The attacker can manipulate a signed OOXML Word document. Microsoft Office for Windows displays the content selected by the attacker without invalidating the signature. This is proven by experiment **E1**. Sections 5, 6, and 7 of our paper contain the description of the attack technique, as well as our evaluation results. For this experiment, the attack classes CIA, CMA, LWA, USF, and MRA are relevant.

(C2): The attacker can manipulate a signed OOXML Word document. Microsoft Office for macOS displays the content selected by the attacker. The status of the signed document is displayed unchanged as a document protected by a digital signature. This is proven by experiment **E2**. Section 3 of our paper contains our evaluation results. Microsoft Office under macOS does not cryptographically check the signature, so direct document manipulation is possible. A dedicated attack technique is not required.

(C3): The attacker can manipulate a signed OOXML Word document. OnlyOffice Desktop for Windows, macOS, and Linux displays the content selected by the attacker without invalidating the signature. This is proven by experiment **E3**. Sections 5, 6, and 7 of our paper contain the description of the attack technique, as well as our evaluation results. For this experiment, the attack classes CIA, CMA, LWA, and MRA are relevant.

A.4.2 Experiments

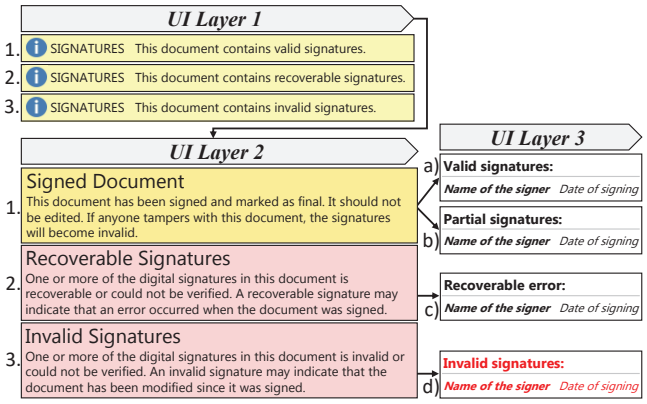

Figure 1: Representation of the different UI layers of Microsoft Office (Windows). A valid signature can combine the three UI layers *1. a)* or *1. b)*. *2. c)* corresponds to a signature with an unknown certificate state, e.g., if the certificate has expired, has been self-signed, or there is no Internet connection to validate the chain of trust. *3. d)* corresponds to an invalid signature, e.g., because an attacker modified the signed content. OnlyOffice Desktop provides only one UI layer to show the status of the document signature. This is essentially the same as the UI layer 3 shown and can reflect the combination *a)* or *d)*.

(E1): *[1 human-hour + 2 compute-hours + 10GB disk]: Manipulation of signed OOXML Word documents under Microsoft Office for Windows.*

Preparation: Download the provided PoC files to the system and launch a Windows version of Microsoft Office. All versions of Microsoft Office for Windows are equally vulnerable to the CIA, CMA, LWA, USF, and MRA attack classes. Microsoft Office 2013 has a bug that causes the UI layer 3 (Figure 1) to be left blank for any signed OOXML Word documents. The MRA attacks work on Microsoft Office 2019 and 2021 only if the office application has been activated/licensed. Set your host system to a date between June 7th and September 5th, 2022. The host system needs an Internet connection to validate the certificate trust chain.

Execution: One by one, open the files starting with *01* and *03* from the folders of the downloaded PoC files. The files *01* correspond to the unmanipulated original documents. Files *03* correspond to the documents signed

by the trusted entity and manipulated by the attacker. The CIA attack is an exception to this rule. Here, the file starting with *02* corresponds to the document manipulated by the attacker. Microsoft Office prompts to repair the documents when opening the files starting with *03* of attack class MRA. This repair prompt is part of the attack and must be confirmed. When closing, the document must not be saved, as this removes the attack vector.

Results: All documents starting with *01* show the content:

> *This document was created and validly signed by a trusted entity.*

All documents manipulated by the attacker contain the content:

> *This document has been manipulated by the attacker.*
> *The content is chosen by the attacker.*

The CMA attack in the *5-2_CMA_Font_Inj* subfolder is an exception. In the attacker document starting with *03*, all numbers were replaced with *6*, and the name was changed to *EVIL*.

For all attacks the UI layer 1 shows a valid signature (see Figure 2). Due to the use of test licenses of Microsoft Office applications, it may happen that the UI layer 1 is only displayed before or, in some versions, after the license query (to be canceled).

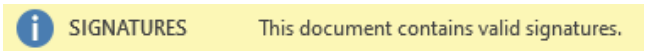

Figure 2: Valid signature under Microsoft Office's UI layer 1.

UI layer 2 is displayed after clicking on *File* (top left). The content of UI layer is the same as shown in Figure 3.

Figure 3: Valid signature under Microsoft Office's UI layer 2.

From UI layer 2, UI layer 3 can be opened by clicking on *View Signatures*. Here, the original signer *trusted.person.ooxml@gmail.com* is displayed as the signer for each manipulated document, although the content differs from the original signed document. For the CIA, CMA, LWA, MRA attack classes the signature type is specified as *Partial signature*, while for the USF attack class the signature type is specified as *Valid signature* (see *(a)* and *(b)* in Figure 1).

(E2): *[15 human-minutes + 10 compute-minutes + 5GB disk (for software)]: Manipulation of signed OOXML Word documents under Microsoft Office for macOS.*

Preparation: Download the provided PoC files to the system and launch a macOS version of Microsoft Office.

Execution: Open *01_document_signed_by_trusted_entity.docx* and *02_direct_manipulation_by_attacker.docx* files one by one.

Results: The document starting with *01* shows the content:

> *This document was created and validly signed by a trusted entity.*

The document manipulated by the attacker (starting with *02*) contains the content:

> *This document has been manipulated by the attacker.*
> *The content is chosen by the attacker.*

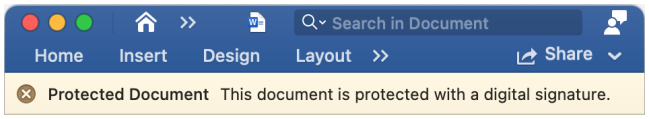

Figure 4: Message about a document protected by a signature in Microsoft Office for macOS.

Microsoft Office for macOS does not provide the UI layer described for the Windows variants. Microsoft Office for macOS only issues a message about a document protected by a signature (see Figure 4). This message also appears for the document manipulated by the attacker, although the included signature file (*sig1.xml*) does not contain any signature-relevant information. This shows that Microsoft Office does not perform any cryptographic verification of the signature.

(E3): *[30 human-minutes + 30 compute-minutes + 5GB disk)]: Manipulation of signed OOXML Word documents under OnlyOffice Desktop for Windows, macOS, and Linux.*

Preparation: Download the provided PoC files to the system and launch a version of OnlyOffice Desktop. All versions of OnlyOffice Desktop are equally vulnerable to the CIA, LWA, and MRA attack classes, as well as the style injection attack from the CMA attack class.

Execution: One by one, open the files starting with *01* and *03* from the folders of the downloaded PoC files. The files *01* correspond to the unmanipulated original documents. Files *03* correspond to the documents signed by the trusted entity and manipulated by the attacker. The

CIA attack is an exception to this. Here, the file starting with *02* corresponds to the document manipulated by the attacker.

Results: All documents starting with *01* show the content:

> *This document was created and validly signed by a trusted entity.*

All documents manipulated by the attacker contain the content:

> *This document has been manipulated by the attacker.*
> *The content is chosen by the attacker.*

The CMA attack in the *5-2_CMA_Style_Inj* subfolder is an exception. Since OnlyOffice has limited vulnerability to this attack, attackers can only hide existing content, but they cannot add new content. Thus, OnlyOffice displays the document manipulated by the attacker as an empty document.

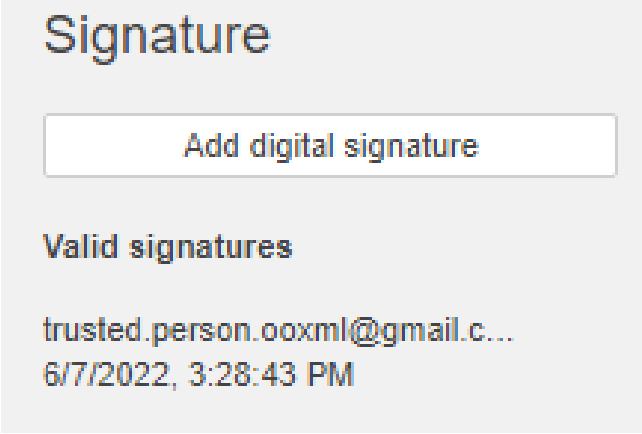

Figure 5: Valid signature under OnlyOffice's UI layer.

OnlyOffice shows the signer and the signature status on the right side after opening the signed documents. On Windows, macOS, and Linux, a valid signature is displayed. Under Windows, the signer *trusted.person.ooxml@gmail.com* is displayed directly, as shown in Figure 5. On macOS and Linux, the issuer of the certificate is displayed first. With a right-click on the issuer, the certificate details can be shown. The same signer (*trusted.person.ooxml@gmail.com*) is displayed within the subject section (*CN*) of the certificate.

(E4): *[1 human-hour)]: Reproducibility of the manipulation of the displayed content of a signed OOXML document.*

The following steps describe the procedure to reproduce that the displayed content is not protected by the signature and thus remains arbitrarily manipulable. This applies to the attack classes CIA, CMA, LWA, and MRA. The USF attack requires that correct hash values are generated for the displayed content. The procedure required for this is described in our paper. The manipulations require 7-Zip and Notepad++.

CIA:

1. right click on *02_CIA_manipulated_by_attacker.docx*. In the subitem 7-Zip click on *Open archive*.
2. in the archive in the subfolder *word/* extract the *people.xml"*.
3. now open *people.xml* with Notepad++.
4. between the first element *<w:t></w:t>* you will find the displayed text. This text can be manipulated arbitrarily.
5. now insert the manipulated *people.xml* back into the archive in the subfolder *word/* and overwrite the old file.
6. When opening the document, the inserted content is now displayed, while the signature status remains valid.

CMA Font Inj.:

1. right-click on *03_CMA_Font_Inj_manipulated_by_attacker.docx*. In the subitem 7-Zip click on *Open archive*.
2. extract the *document.xml.rels* in the archive in the subfolder *word/_rels/*.
3. now open *document.xml.rels* with Notepad++.
4. delete the entry *<Relationship Id="rId4" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/fontTable" Target="fontTable.xml"/>*. This will remove the link to the malicious embedded fonts.
5. now add the manipulated *document.xml.rels* back into the archive in the *word/_rels/* subfolder and overwrite the old file.
6. when opening the document, the content without malicious fonts is now displayed, while the signature status remains valid.

CMA Style Inj.:

1. right-click on *03_CMA_Style_Inj_manipulated_by_attacker.docx*. In the subitem 7-Zip click on *Open archive*.
2. in the archive in the subfolder *word/_rels/* extract the *document.xml.rels*.
3. now open *document.xml.rels* with Notepad++.
4. delete the entry *<Relationship Id="rId1" Type="http://schemas.openxmlformats.org/office*

Document/2006/relationships/styles" Target="styles.xml"/>. This will remove the link to the malicious style elements.
5. Now add the manipulated *document.xml.rels* back into the archive in the *word/_rels/* subfolder and overwrite the old file.
6. when opening the document, the content without malicious style elements is now displayed, while the signature status remains valid.

LWA:

1. right-click on *03_LWA_manipulated_by_attacker.docx*. In the subitem 7-Zip click on *Open archive*.
2. in the archive in the subfolder *word/* extract the *document.xml*.
3. now open *document.xml* with Notepad++.
4. between the first element *<w:t></w:t>* is the displayed text. This text can be manipulated arbitrarily.
5. now insert the manipulated *document.xml* back into the archive in the subfolder *word/* and overwrite the old file.
6. when opening the document, the inserted content is now displayed, while the signature status remains valid.

MRA DDA:

1. right click on *03_DDA_manipulated_by_attacker.docx*. In the subitem 7-Zip click on *Open archive*.
2. in the archive in the subfolder *word/* extract the *document2.xml*.
3. now open *document2.xml* with Notepad++.
4. between the first element *<w:t></w:t>* is the displayed text. This text can be manipulated arbitrarily.
5. now insert the manipulated *document2.xml* back into the archive in the subfolder *word/* and overwrite the old file.
6. When opening the document, the inserted content is now displayed, while the signature status remains valid.

MRA ETA:

1. right click on *03_ETA_manipulated_by_attacker.docx*. In the subitem 7-Zip click on *Open archive*.
2. in the archive in the subfolder *word/* extract the *document.xml*.
3. now open *document.xml* with Notepad++.
4. between the first element *<w:t></w:t>* is the displayed text. This text can be manipulated arbitrarily.
5. now insert the manipulated *document.xml* back into the archive in the subfolder *word/* and overwrite the old file.
6. when opening the document, the inserted content is now displayed, while the signature status remains valid.

Direct manipulation under macOS:

1. create a signed Word document using Microsoft Office for Windows.
2. right click on the document. In the subitem 7-Zip click on *Open archive*.
3. in the archive in the subfolder *word/* extract the *document.xml* and in the subfolder *_xmlsignatures/* extract the *sig1.xml*.
4. now open *document.xml* with Notepad++. between the first element *<w:t></w:t>* is the displayed text. This text can be manipulated arbitrarily.
5. now insert the manipulated *document.xml* again into the archive in the subfolder *word/* and overwrite the old file.
6. now open *sig1.xml* with Notepad++.
7. delete the entire content of *sig1.xml*.
8. insert the manipulated *sig1.xml* back into the archive in the subfolder *_xmlsignatures/* and overwrite the old file.
9. when opening the document, the inserted content is now displayed, while the document is still displayed as a document protected by a signature.

A.5 Notes on Reusability

Since the certificate used to sign the PoC files was only valid for three months and expired on September 5th, 2022, the host system date must be set accordingly to a date between June 7th and September 5th, 2022.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2023/.