# WDAC

Notes from the field

# Speakers


**Kim Oppalfens**

CIO, OSCC

Kim.oppalfens@oscc.be

@thewmiguy

**Tom degreef**

Extraordindary Consultant, OSCC

@tomdegreef

1-2-3; 2-3; 1-3; 2-3

# Welcome

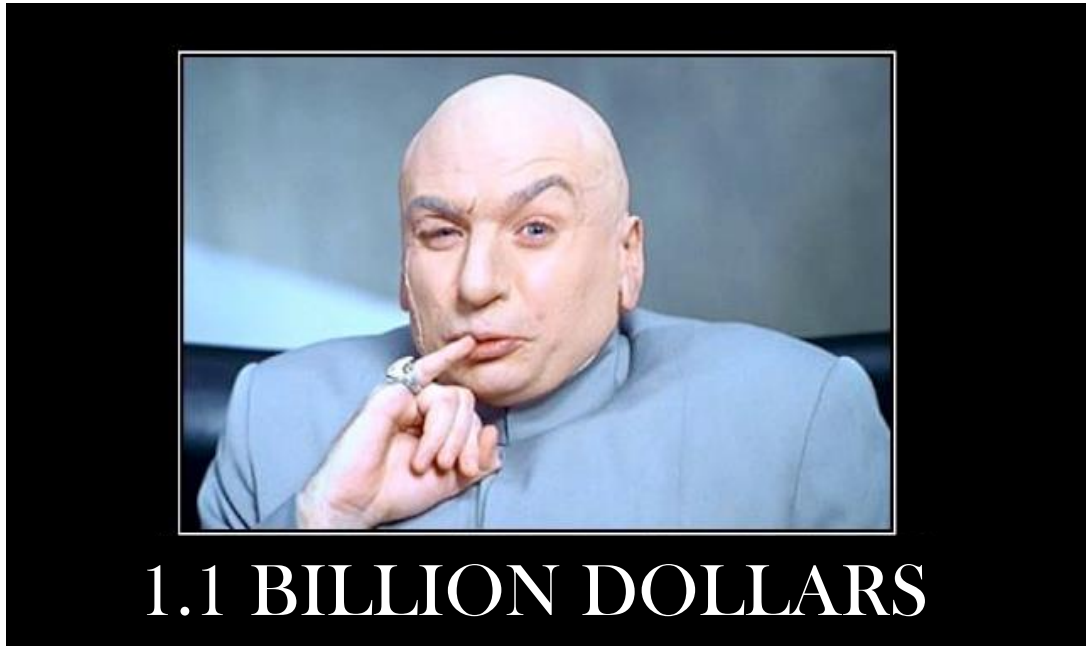
- Benefits
- Benefits over Applocker
- OSCC Base Policy
- Gotchas
- Handling Scripts
- Catalogs
- AppControl.AI
- Training

# Application Allowlisting benefits

# Ransomware cost

- Based on CoveWare Ransomware report
- Protects against 94% of automated malware & ransomware attacks (*)
  - Enforces solid processes and security practices

(*) 72.23% of all statistics are invented on the fly



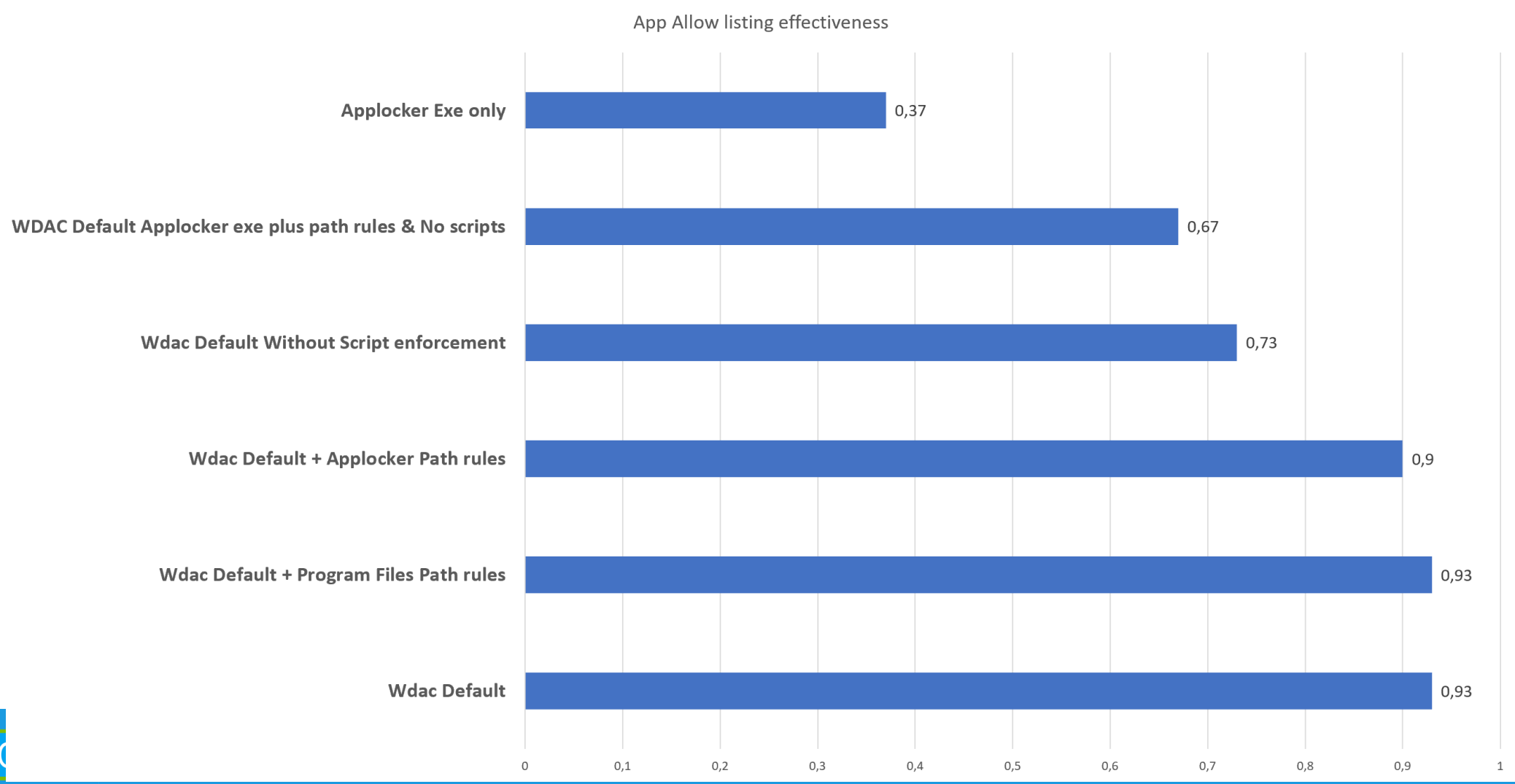
# Application Allow listing

- A paradigm shift
- Code integrity aka
  - Non-jailbroken Windows
  - “Modify an electronic device to remove restrictions imposed by the manufacturer or operator to allow the installation of unauthorized software.”

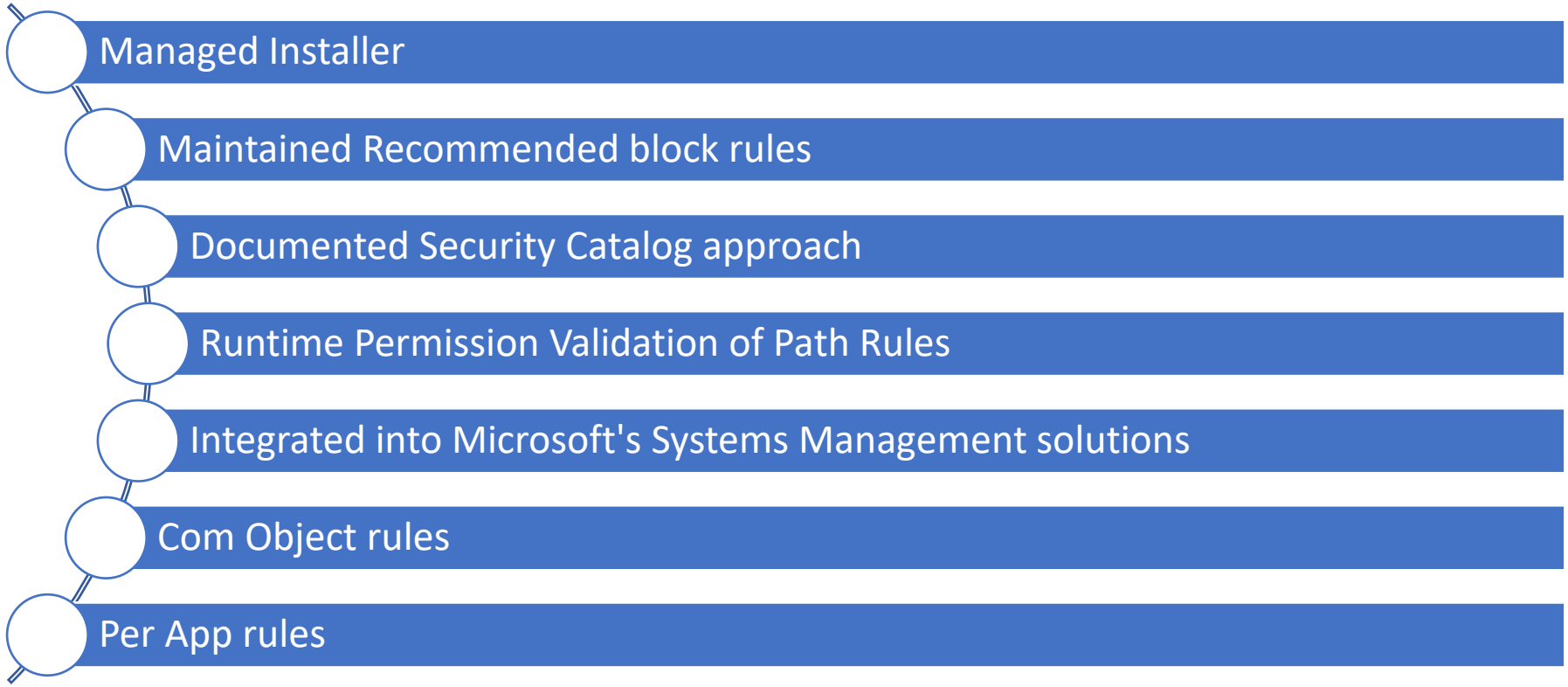
Aaron Margosis: “Can severely limit what an exploited vulnerability in a user program can accomplish.”

Kim Oppalfens: “Non-limited code execution will almost certainly result in full system compromise over time.”

# Recap – All of the stats



# WDAC Benefits over Applocker



# WDAC vs Applocker in Real live Printer Nightmare

- OS Trusted based on signatures vs OS Trusted based on path rules
- TL;DR
  - 0-Day
  - Remote Code Execution + Privilege escalation to LocalSystem
  - Took Microsoft 3 months to patch all vectors
  - Bug in remote driver installation allows regular user to install printer drivers and **drop DLL's**
- Demystifying the PrintNightmare vulnerability (sygnia.co)
- CVE-2021-34527 - Security Update Guide - Microsoft - Windows Print Spooler Remote Code Execution Vulnerability
- Stopped by WDAC, DLL would not be allowed to execute
  - DLL is dropped in Windows\System32\spool\drivers which is allowed in default applocker rules

# Getting started

Building your base policy

# Rule options – OSCC's advice

- 0 - Enabled:UMCI
- 3 - Enabled:Audit Mode
- 4 – Prevent “Flighted” builds (EG: Windows insider)
- 6 - Enabled:Unsigned System Integrity Policy
- 9 - Enabled:Advanced Boot Options Menu
- 12 – Required: Enforce Store Applications
- 13 – Managed installer
- 16 - Enabled:Update Policy No Reboot

- 2 - Required:WHQL
- 19 - Enabled:Dynamic Code Security

# Recommended Rules

- Code signer certificate
- C:\Windows\Assembly FilePath rule
- Microsoft recommended block rules
- Deny Heartbeat executable
- (Inbox) Windows Store app rules

# Picking-order for generating trust

Managed Installer

- No additional work needed

Security Catalogs

- Self-Documenting

(Supplemental) Policy

- Document properly

# Create your initial packaged app policy

```
Get-AppxPackage | Out-GridView -PassThru | %{$Rules += New-CIPolicyRule -Package $_}

New-CIPolicy -Rules $rules -FilePath .\packagedapppolicy.xml -MultiplePolicyFormat -UserPEs

Set-CIPolicyIdInfo -BasePolicyToSupplementPath "basepolicy.xml" -FilePath .\packagedapppolicy.xml
```

# Create package app policy

Using PowerShell

# Gotchas (1)

- The Windows Store
  - Python?
  - Signature based vs Package family name
  - Blocking store access?
- File hash vs Authenticode hash
- The Self-Updating app paradox
  - Failed app upgrades are seldomly problematic
  - FilePublisher rules (+Product Names)
  - Path rules if needs be

# Gotchas (2)

- Fixed in PSADT 4.0 (Just released)
- PSADT C# File in AppDeployToolkitMain.PS1

```
$PackagePath = 'C:\PSADT-PrusaSlicer'
$tempfolder = "c:\temp"

#Convert .cs file to .dll using csc.exe
$PSADT_CSfile = Get-ChildItem -Path "$PackagePath\AppDeployToolkit" -Filter '*.cs'
Copy-Item -Path $($PSADT_CSfile.fullname) -Destination $tempfolder -Force start-process -FilePath "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe"
    -ArgumentList "/target:library /out:$tempfolder\AppDeployToolkitMain.dll $tempfolder\$($PSADT_CSfile.name)" -Wait -WorkingDirectory "$PackagePath\AppDeployToolkit"
Move-Item -Path "$tempfolder\AppDeployToolkitMain.dll" -Destination "$PackagePath\AppDeployToolkit" -Force Remove-Item $tempfolder\$($PSADT_CSfile.name) -Force

#replace the reference to appdeploytoolkitmain.cs with appdeploytoolkitmain.dll in appdeploytoolkitmain.ps1
$psadtmain = $psadtmain -replace "AppDeployToolkitMain.cs", "AppDeployToolkitMain.dll"
Set-Content -Path "$PackagePath\AppDeployToolkit\AppDeployToolkitMain.ps1" -Value $psadtmain -Encoding utf8 -Force
```

# The unsurmountable challenge of code signing

## Create your own cert without a PKI

```
New-SelfSignedCertificate -Type CodeSigningCert -Subject 'Application control
signing cert' -CertStoreLocation Cert:\CurrentUser\my
```

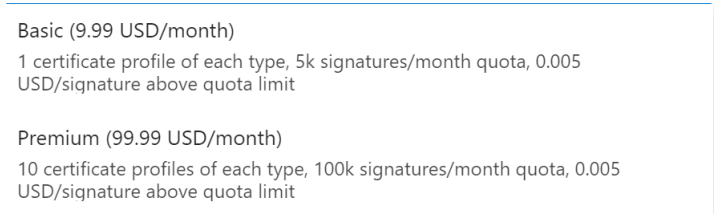
***#!!! Add cert to Trusted Store !!!***

## Sign a file using that cert

```
Set-AuthenticodeSignature -Certificate $(gci Cert:\CurrentUser\My -CodeSigningCert
|? {$_.Subject -like 'CN=Application control signing cert'}) -TimestampServer
http://timestamp.digicert.com -HashAlgorithm sha256 -FilePath "file"
```

# Azure Trusted Signing accounts

- Fresh out of private preview
- Service to allow codesigning files with Microsoft managed certificates
- Developed with WDAC in mind
- Automation friendly
  - Plenty of integrations (PowerShell, SignTool, Github actions)
- Pricing



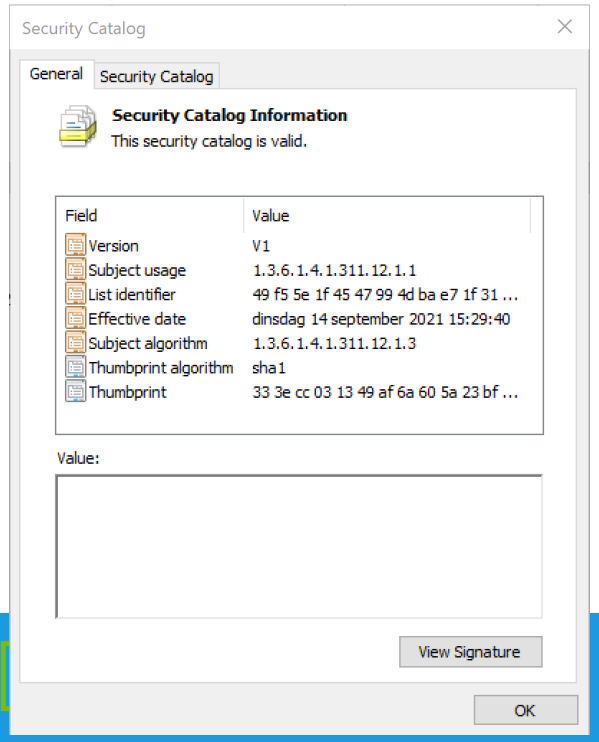

- Trusted Signing documentation | Microsoft Learn

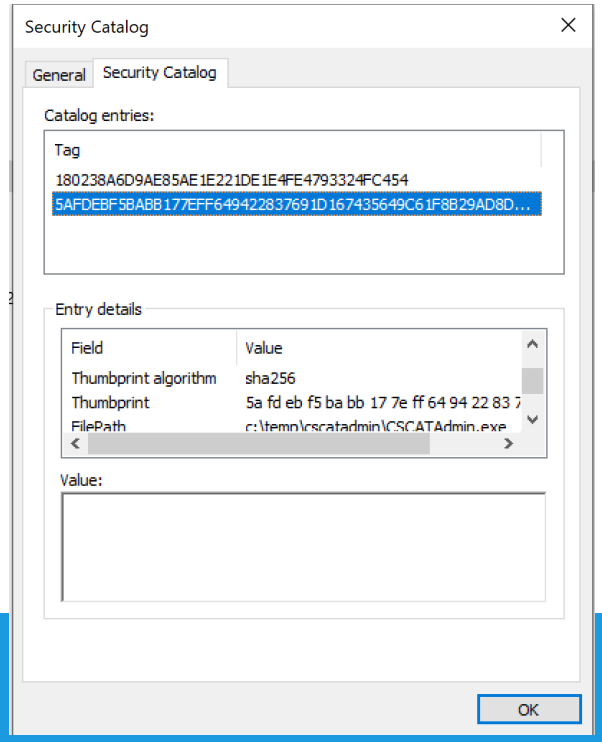
# Auto sign PowerShell

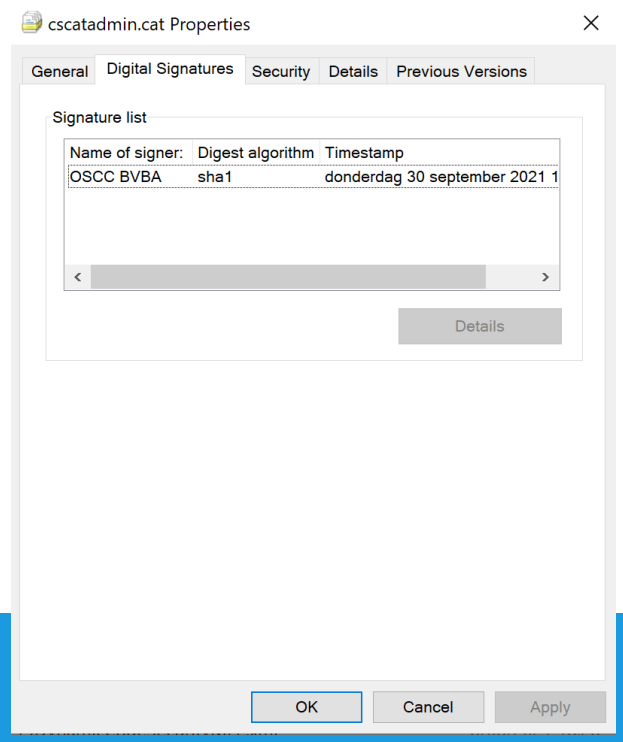
Using VsCode

# Handling Backlog

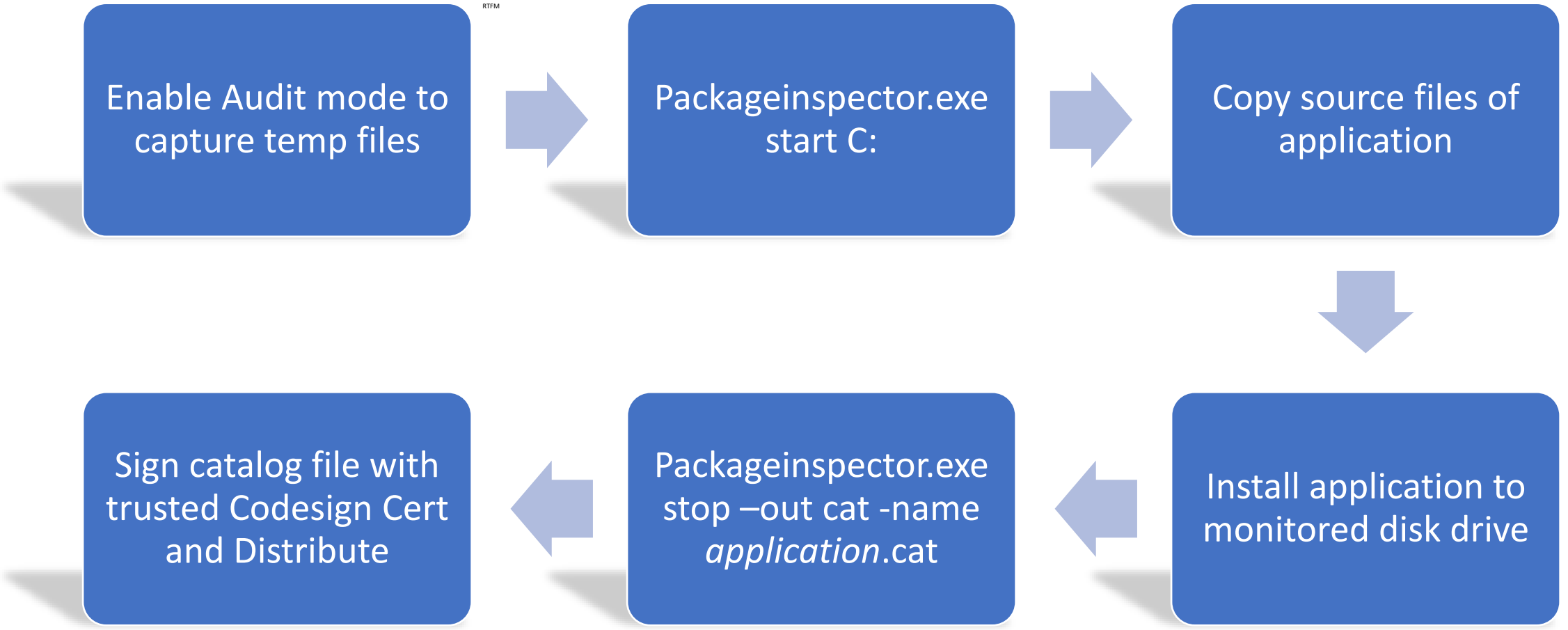
- .cat files, Introduced 2 decades ago with Windows 2000
- Enforcement started in Windows XP
  - Driver signing is Kernel Mode Code Integrity
- Windows Defender Application Control adds User mode Code Integrity



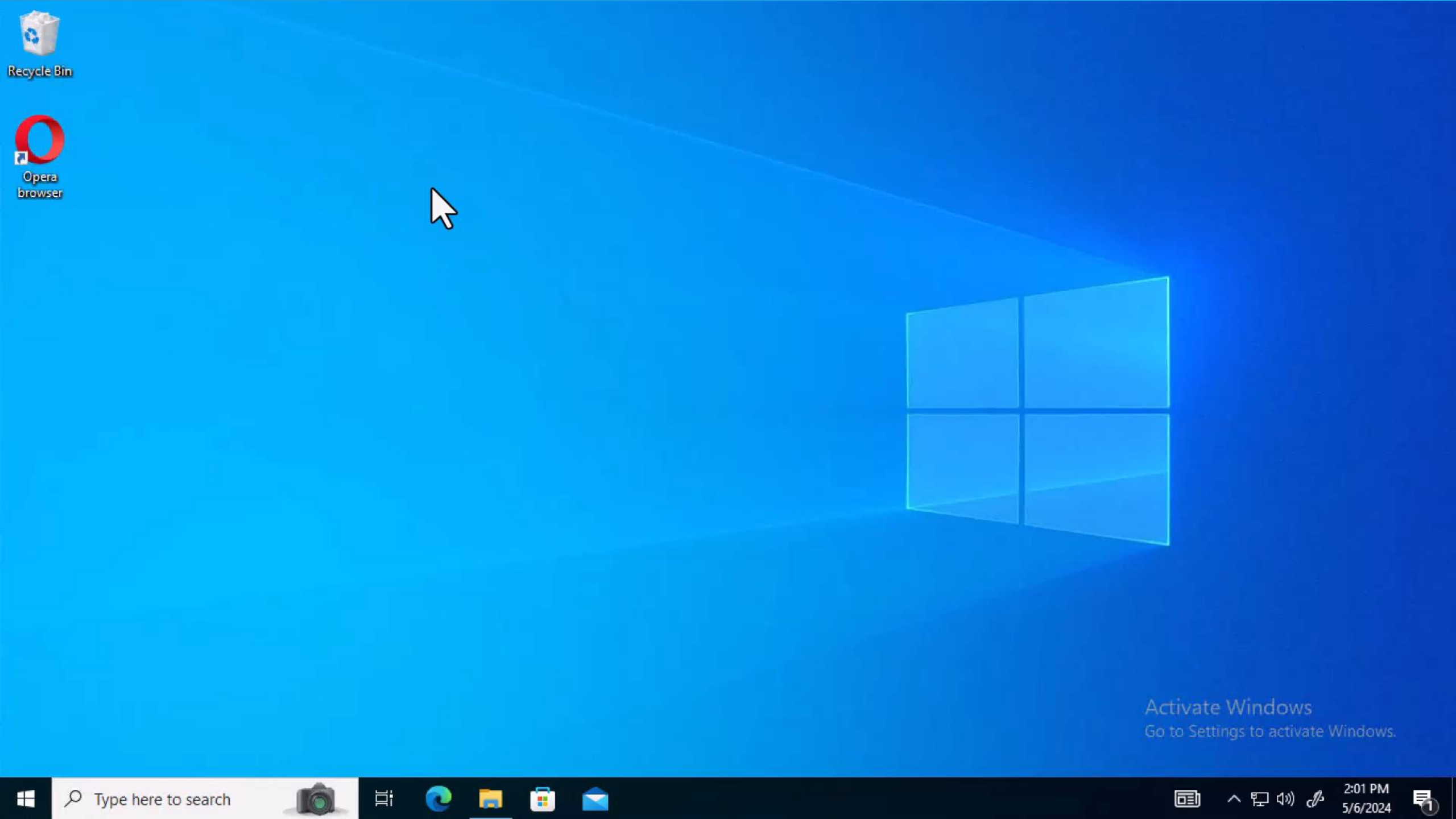



# Make app trusted with signed Catalog monitoring

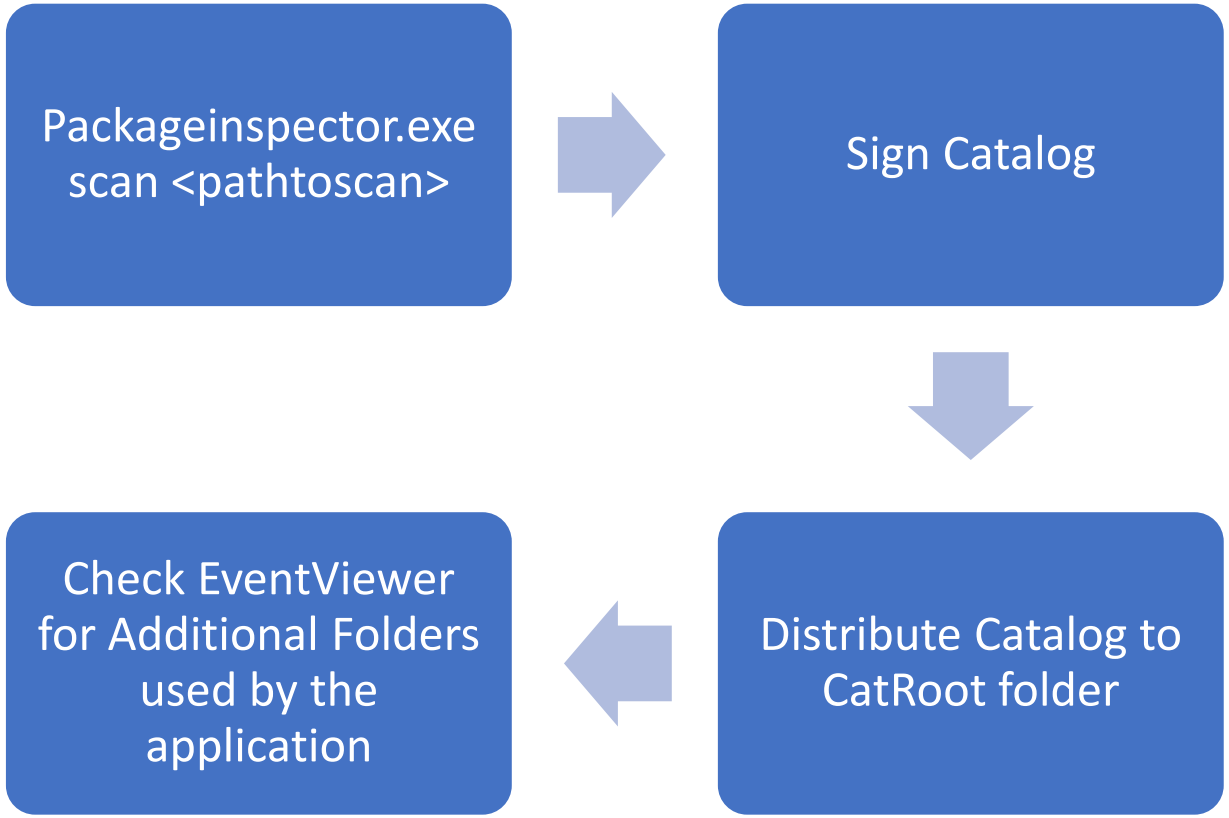


# PackageInspector

Start / Stop

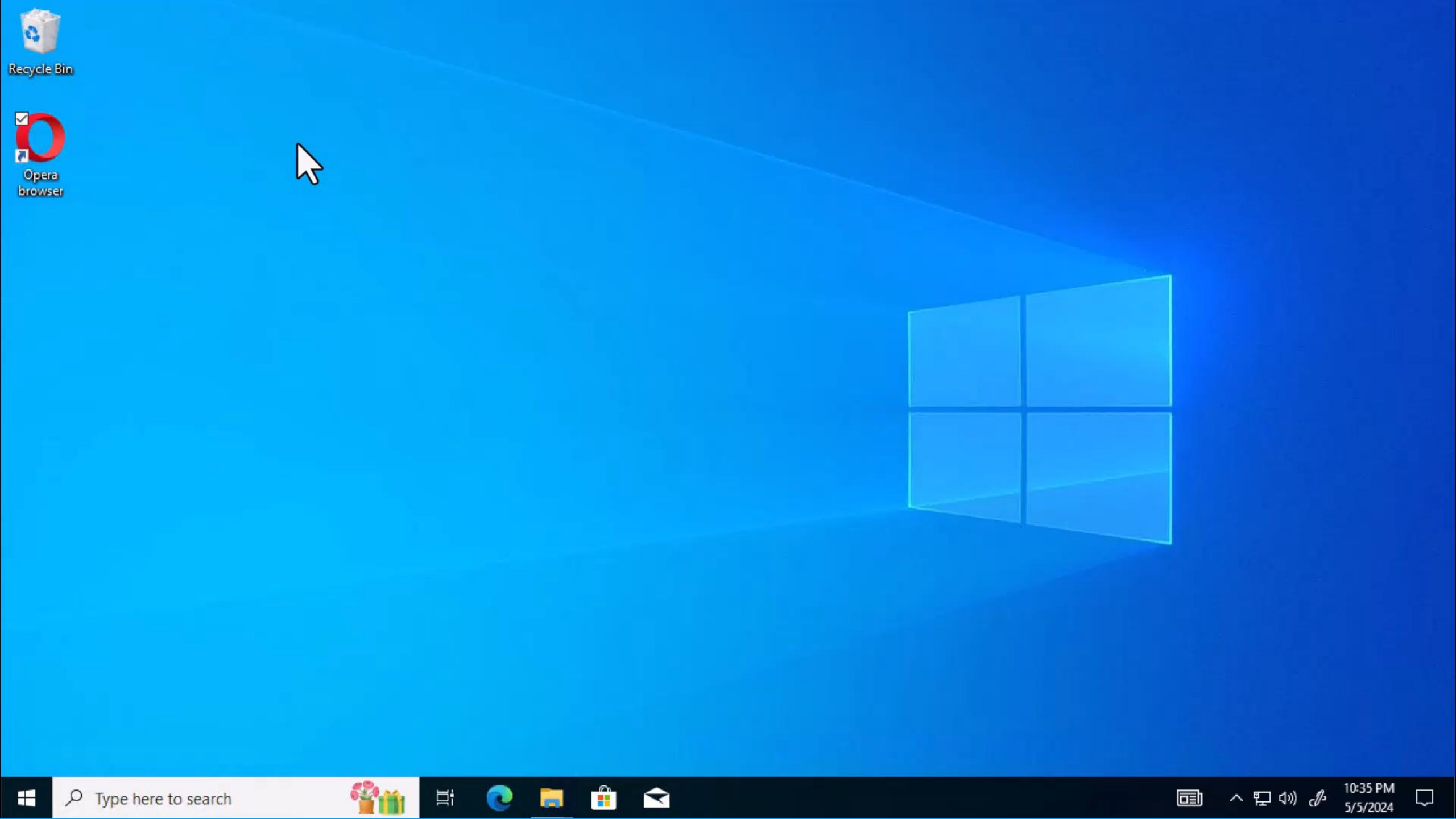

Recycle Bin
Opera browser
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:01 PM
5/6/2024

# Make app trusted with Catalog Scan



# Assisted backlog handling

AppControl.Ai


Recycle Bin
Opera browser
Type here to search
10:35 PM
5/5/2024

# WDAC Masterclass at ViaMonstra

Starts April 2025 – Mail Tom.Degreef@oscc.be for 10% discount