

Home

About



Blocking Visual Studio Code embedded reverse shell before it's too late



WINDOWS

SPLUNK

SOC

POWERSHELL



Overview

- Introduction
- Mitigation
 - Domains blacklist
 - Applocker
 - GPO
- Detection
 - Process

Search...

ipfyx

Yet another cybersecurity blog

READ MORE

Featured Posts

 Blocking Visual Studio Code embedded reverse shell before it's too late

Recent **Posts**

- D LINK DNS323 Fan Speed Control
- Bluekeep, why









Visual studio code tunnel

Introduction

Since July 2023, Microsoft is offering the perfect reverse shell, embedded inside Visual Studio Code, a widely used development tool. With just a few clicks, any user with a github account can share their visual studio desktop on the web. VS code tunnel is almost considered a lolbin (Living Of the Land Binary).

I am so glad that my users now have the ability to expose their computer with highly sensitive data right on the web, through an authentication I nor control, nor supervise. My internal network is now accessible from anywhere!

The worse part is that this tunnel can be triggered from the cmdline with the portable version of code.exe. An attacker just has to upload the binary, which won't be detected by any anti-virus since it is legitimate and singed windows binary.

It is therefore something to watch out for.

Execute it, open the link mentioned, log in to your github account and there is your reverse shell.

DEFENSE 2

ADMIN 1

CYBERSECURITY 1

Tags

POWERSHELL 2

WINDOWS 2

BASH 1

BLUEKEEP 1

BLUETEAM 1

CVE 1

DEFENSE 1

LINUX 1

RDP 1

SOC 1

SPLUNK 1

WSUS 1

POWERSHELL



The binary is definitely signed:



Notice how there are nor ProductName nor BinaryName value, we will use them later.

pfiatde blog post already did a great job at describing what an attacker can do so I won't say much more. Go check his awesome blog post.

Mitigation

But now, as a defender, how do we block or detect its usage?





If you're part of an organization who wants to control access to Remote Tunnels, you can do so by allowing or denying access to the domain global.rel.tunnels.api.visualstudio.com.

Yeah it's not that simple. It will indeed block most users but, from my testing, it won't block an attacker that has already established a tunnel once. Meaning the tunnel can be kept active or restarted at will despite this domain being blacklisted.

From my understanding, VScode contacts *global.rel.tunnels.api.visualstudio.com* to get its "clusters" : https://global.rel.tunnels.api.visualstudio.com/api/v1/clusters

Here is a sample of the resulting json:





- Dev Tunnels
 - global.rel.tunnels.api.visualstudio.com
 - [clusterId].rel.tunnels.api.visualstudio.com
 - [clusterId]-data.rel.tunnels.api.visualstudio.com
 - *.[clusterId].devtunnels.ms
 - *.devtunnels.ms

Blocking those domains will block and cut off any VS code tunnel:

- *.tunnels.api.visualstudio.com
- *.devtunnels.ms

Applocker

Applocker is Microsoft application whitelisting technology. When enabled and configured, with default rules for example, everything is blocked by default except for the executables and scripts defined in the preceding rules. Let's pretend we use Microsoft default rules for demo purpose.

To generate them, in *Group Policy Management Editor*, go to *Computer Configuration -> Windows Settings -> Security Settings -> Application Control Policies -> AppLocker*. Right Click on *Create Default Rules*.





Here is the resulting xml:

When using those rules, everything that is in Program Files and Windows is allowed, everything else is blocked. Administrators are allowed to execute anything, VSCode for instance. But if you were to define a rule that blocks VSCode, this rule would apply first, since "Deny rules" are applied before "Allowed rules".

Let's build a rule to block VScode altogether. The rule applies to Everyone (SID S-1-1-0), therefore to Administrators (SID S-1-5-32-544).

POWERSHELL

Get-AppLockerFileInformation 'C:\Users\ipfyx\AppData\Local\Prog





Dinary versionicange to materially viscode version

Warning: Do not import the following xml in Applocker. Imported alone, this rule will completely block your system. Add a Allow * rule beforehand.

Included in MS default rules, here is the resulting xml (called vscode.xml from now on):

```
<AppLockerPolicy Version="1">
  <RuleCollection Type="Appx" EnforcementMode="Enabled" />
  <RuleCollection Type="Dll" EnforcementMode="NotConfigured" />
  <RuleCollection Type="Exe" EnforcementMode="Enabled">
  <EilaDathBullo Id="021cc481_6017_4653_8f75_050b80acca20" Name
</pre>
```





Unfortunately, this rule won't apply to code tunnel because this binary doesn't have a product name (thanks Microsoft!):

```
Get-AppLockerFileInformation 'C:\Users\ipfyx\AppData\Local\Program
New-AppLockerPolicy: Les règles ne peuvent pas être créées. Le:
%OSDRIVE%\USERS\ipfyx\DOWNLOADS\VSCODE_CLI_WIN32_X64_CLI\CODE.E.
```

To be sure, let's test the previously generated vscode rule.

```
# Portable code tunnel is blocked but not by the rule

Test-AppLockerPolicy -XmlPolicy .\vscode.xml -Path .\code.exe -I
FilePath : C:\Users\ipfyx\Downloads\vscode_cli_win32_x64_I
PolicyDecision : DeniedByDefault
MatchingRule :

# Code tunnel is blocked but not by the rule

Test-AppLockerPolicy -XmlPolicy .\vscode.xml -Path 'C:\Users\ip
FilePath : C:\Users\ipfyx\AppData\Local\Programs\Microsof
PolicyDecision : DeniedByDefault
...
```

The *PolicyDecision* is *DeniedByDefault*, which means the rule we just built from the vscode binary does not apply to code-tunnel. If no rules matches, the default behaviour for applocker is to deny any execution. Code-tunnel is denied there, but it would not be if it was





```
# Export

Get-AppLockerFileInformation .\code.exe | New-AppLockerPolicy -I

# Code tunnel is blocked

Test-AppLockerPolicy -XmlPolicy .\vscode-tunnel-hash.xml -Path

FilePath : C:\Users\ipfyx\AppData\Local\Programs\Microsof

PolicyDecision : Denied

MatchingRule : code.exe
```

Warning: Do not import the following xml in Applocker. Imported alone, this rule will completely block your system. Add a Allow * rule beforehand.





Visual Studio has the so wanted features :

```
- Dev Tunnels - controls test functionality
```

But as of today (1.8.82), VScode doesn't. You can force automatic updates though! UpdateMode_default would be my goto.

```
gc 'C:\Users\ipfyx\AppData\Local\Programs\Microsoft VS Code\pol:
```

Detection





index=win sourcetype="XmlWinEventLog" EventCode=4688 code tunne:
| stats min(_time) as time_min max(_time) as time_max count as

host	SubjectUser	cmdline
sauteTomate	ipfyx	"c:\Users\ipfyx\AppData\Loca VS Code\bin\code-tunnel" tun license-termsname totallyNo

But the binary *code.exe* could be renamed. However, what an attacker can't change are the *tunnel* and *--accept-server-license-terms* options:

index=win sourcetype="XmlWinEventLog" EventCode=4688 tunnel according | stats min(_time) as time_min max(_time) as time_max count as or

	host	SubjectUser	cmdline
	sauteTomate	ipfyx	"c:\Users\ipfyx\AppData\Loca VS Code\bin\c0de.exe" tunne license-termsname totallyNo
	sauteTomate	ipfyx	"c:\Users\ipfyx\AppData\Loca VS Code\bin\code-tunnel" tun license-termsname totallyNo





Looking for suspicious child process

Great idea from lobas:

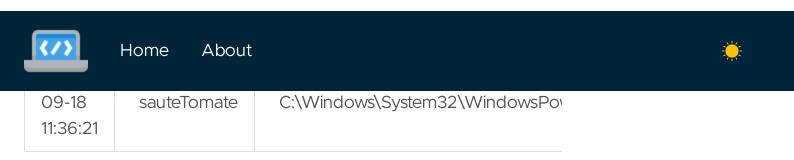
```
IOC: Process tree: code.exe -> cmd.exe -> node.exe -> winpty-ag
```

A straightforward search would be:

But once again, *code.exe* could be renamed

We could use the search above to look for suspicious child process from our process with *accept-server-license-terms* option :

```
index=win sourcetype="XmlWinEventLog" EventCode=4688 (cmd OR por
    [search index=win sourcetype="XmlWinEventLog" EventCode=4688
    | table host NewProcessId
    | rename NewProcessId as ParentProcessId
    | format]
| table _time, host, NewProcess, ParentProcess
```



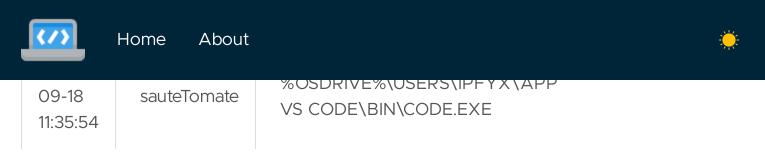
EDIT 20230924: license terms agreement is not mandatory. And even if it were, this could be bypass by creating the file 1 icense_content.json with the right content (see below). All we have left is detecting the string "*.exe*tunnel*" in a commandline, which is subject to false positive.

Applocker

EDIT 20230924

Since we defined some Applocker rules, we got some log from it! We could just search for that weird publisher value:

time host FilePath



Got you! There should not be that many binary signed without ProductName right? Let's count them quickly in System32 for example:

```
gci -Recurse C:\Windows\System32 -include *.exe |Get-AppLockerF:

Count: 20
```

Sigh. 20 binary names to masquerade as...

File creation

Another great idea from lobas:

```
IOC: File write of code_tunnel.json which is parametizable, but def
aults to: %UserProfile%\.vscode-cli\code_tunnel.json
```

license_consent.json file could also be watched.

```
POWERSHELL

PS > gc C:\Users\ipfyx\.vscode\cli\\code_tunnel.json

{"name": "sauceTomate", "id": "9s43zAc9", "cluster": "uks1"}

DS > gc C:\Users\ipfyy\ vscode\cli\\license consent icon

We would like to use third party code to improve the functionality of this website.
```





changed with the --cli-data-dir option.

```
.\code.exe tunnel help
...
GLOBAL OPTIONS:
--cli-data-dir <CLI_DATA_DIR> Directory where CLI metada
```

Web traffic monitoring

If you haven't blocked the domains I mentioned earlier, watch out for any HTTP traffic toward those domains.

Conclusion

A GPO parameter would be awesome but it is yet to be seen. An Applocker hash rule is not sustainable. Moreover, your non-domain-joined computers would not be concerned by neither of them.

Therefore, for now, we are only left to blocking those two bad boys:

```
*.tunnels.api.visualstudio.com
```

*.devtunnels.ms

Blocking Visual Studio Code embedded reverse shell before it's too late | ipfyx blog - 31/10/2024 16:53 https://ipfyx.fr/post/visual-studio-code-tunnel/

