UntrustIDE: Exploiting Weaknesses in VS Code Extensions

Elizabeth Lin, Igibek Koishybayev, Trevor Dunlap, William Enck, Alexandros Kapravelos

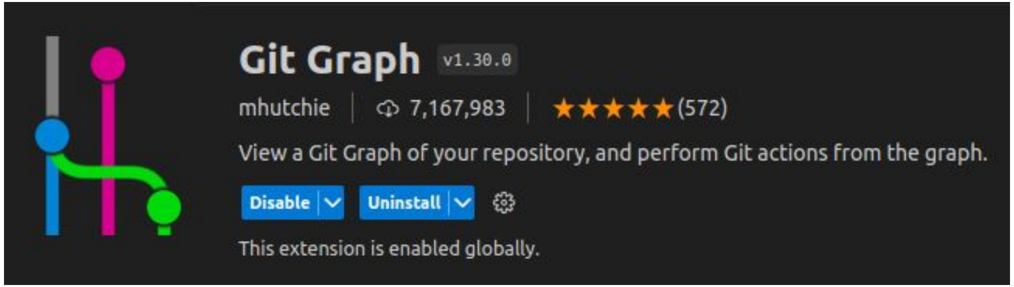
VS Code

- Microsoft VS Code is the most popular IDE used by developers
  - Built on Electron framework
  - Stack Overflow survey reports 73.71% of devs use VS Code
- Popularity stems from the marketplace of over 50K extensions written by third-party developers
  - Support for programming languages
  - Linters
  - Databases / Containers

Example Extension

Git Graph

- View git graph
- Perform git actions: push, pull, merge, etc
- Branch actions
- View file diffs
- Compare commits

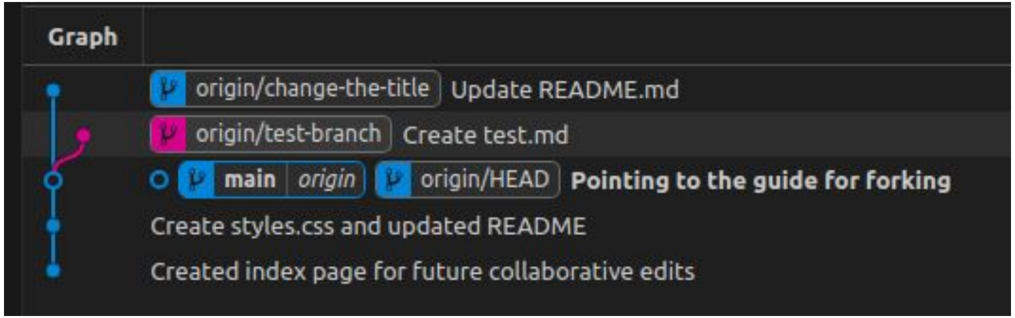


Git Graph v1.30.0

mhutchie | 7,167,983 | ★★★★★ (572)

View a Git Graph of your repository, and perform Git actions from the graph.

This extension is enabled globally.



origin/change-the-title Update README.md

origin/test-branch Create test.md

main origin origin/HEAD **Pointing to the guide for forking**

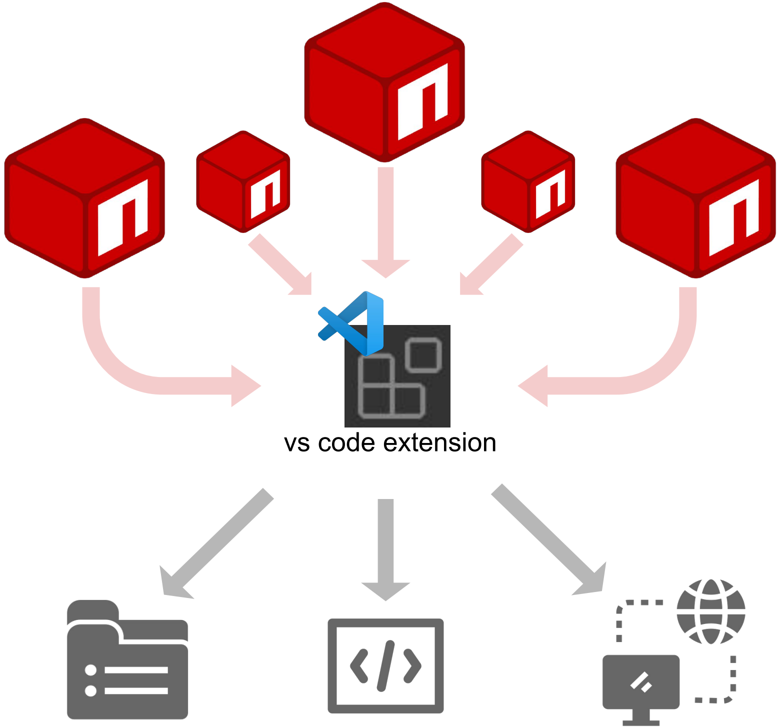
Create styles.css and updated README

Created index page for future collaborative edits

What is the state of **exploitable** vulnerabilities in VS Code extensions?

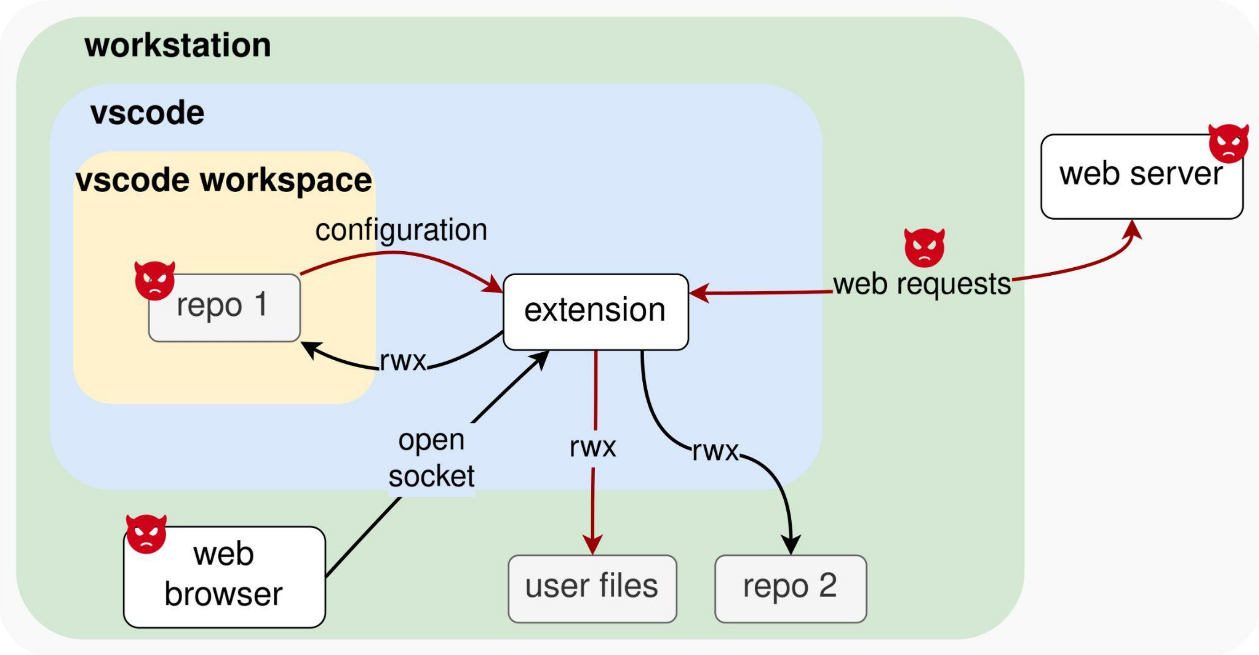
VS Code Extension Capabilities

- Node.js application
- npm modules
- File access
- Network access
- Run web servers
- Shell commands and scripts



Threat Model

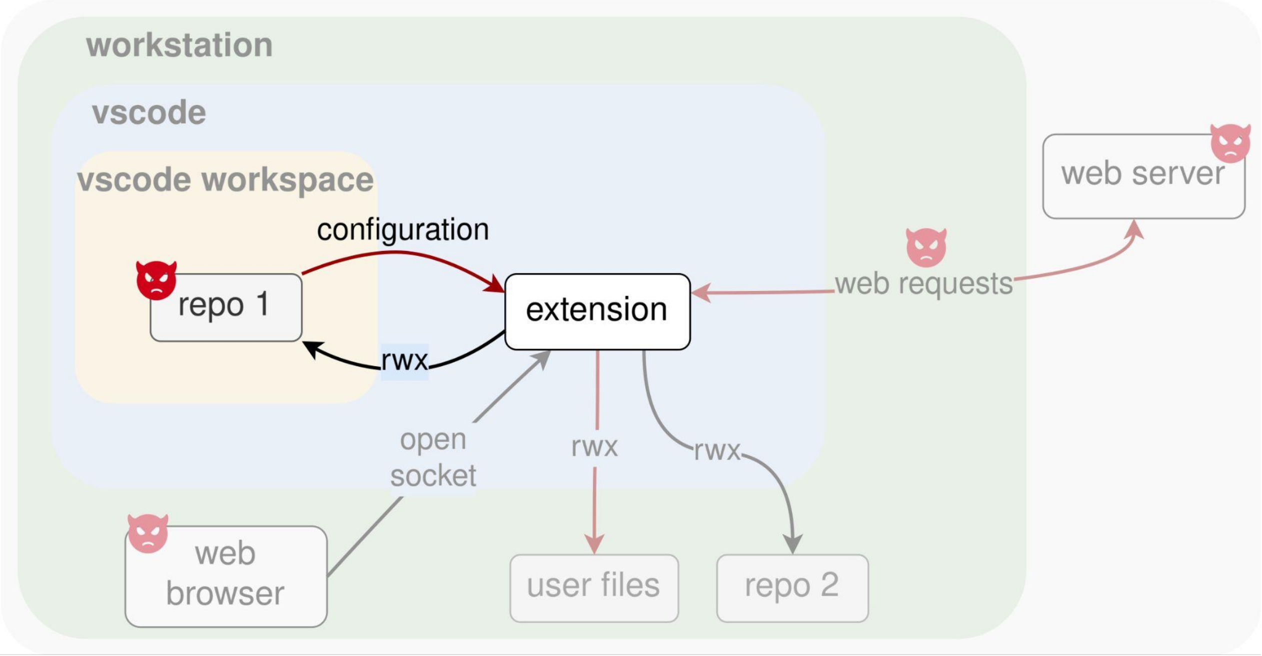
- Users are benign
- Extension developers are benign
- Git repo is untrusted
- External attacker



Threat Model

Attack Vector

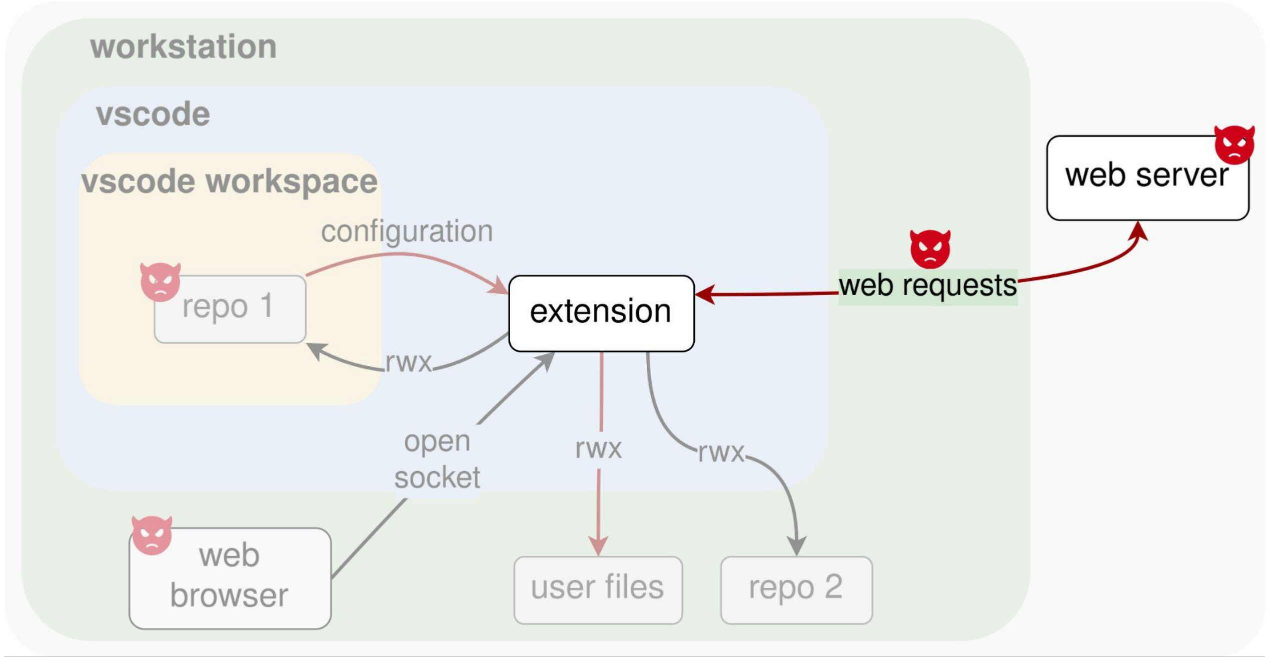
- Workspace Setting
- File Read



Threat Model

Attack Vector

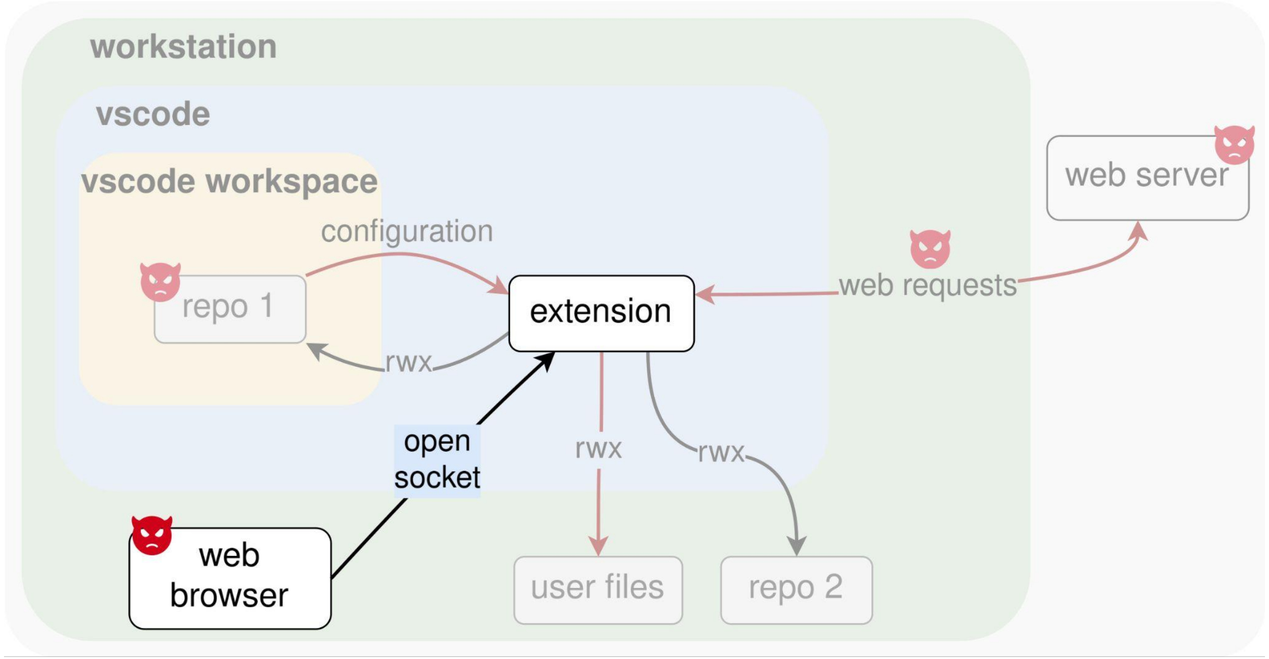
Network Response



Threat Model

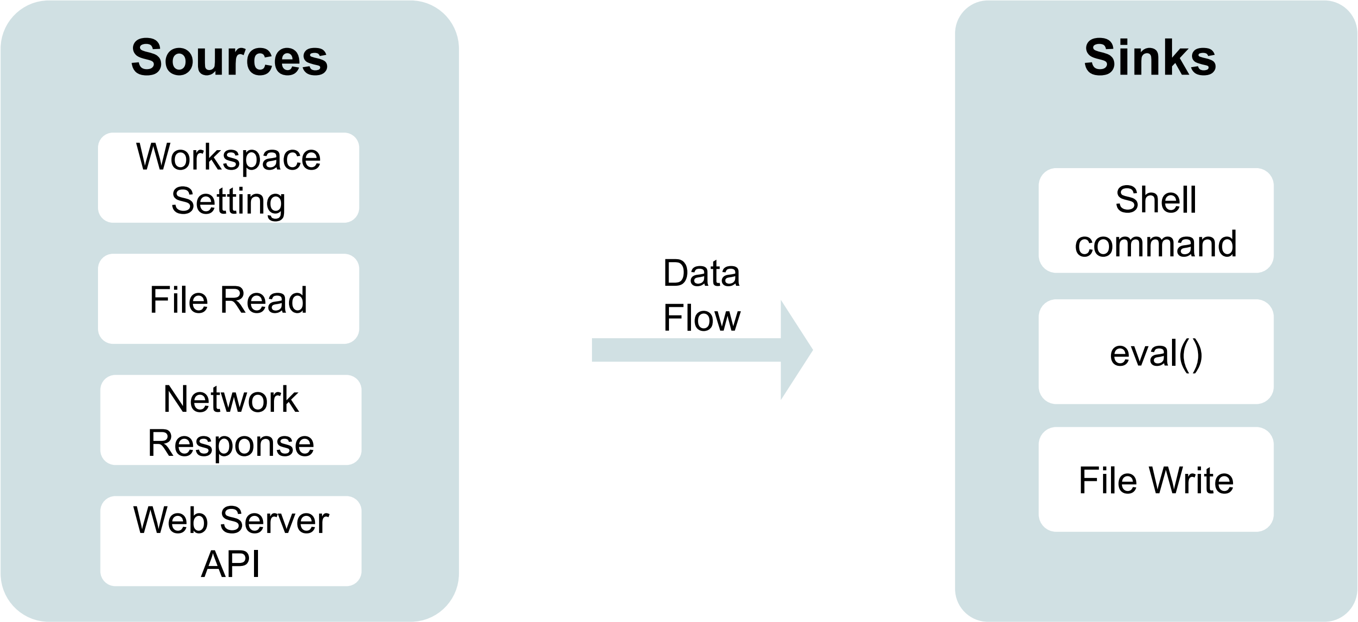
Attack Vector

Web Server API



Vulnerability Discovery using Data Flow Analysis

- Data flow analysis of 4 taint sources to 3 taint sinks



Sources

Workspace Setting

File Read

Network Response

Web Server API

Data Flow

Sinks

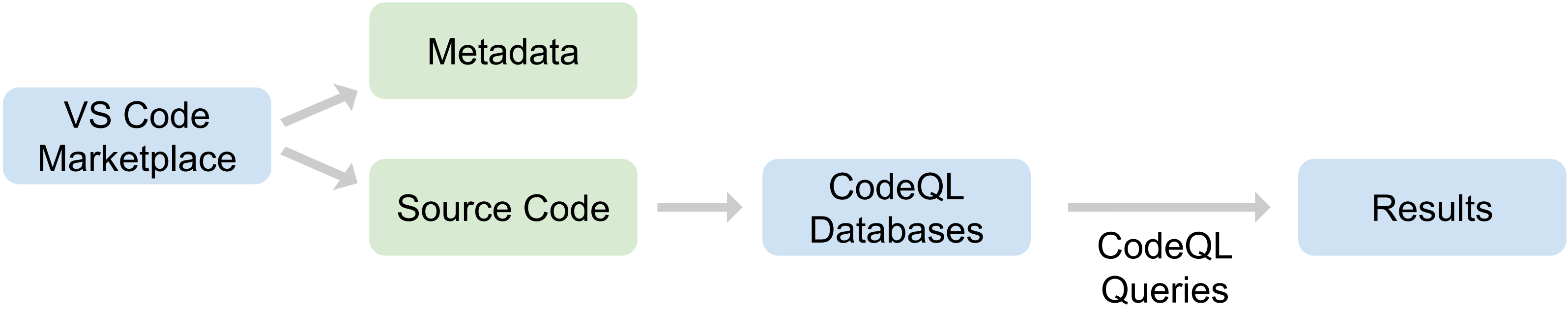
Shell command

eval()

File Write

Data Flow Analysis

- We constructed 12 base queries in CodeQL
- Applied additional filter queries when necessary



VS Code Marketplace

Metadata

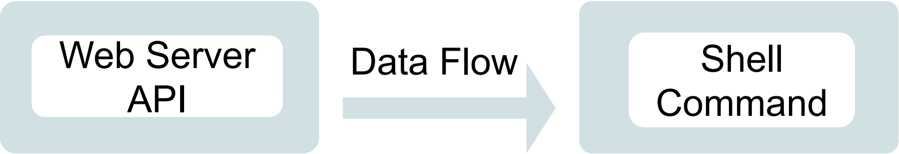
Source Code

CodeQL Databases

CodeQL Queries

Results

Example



Web Server API

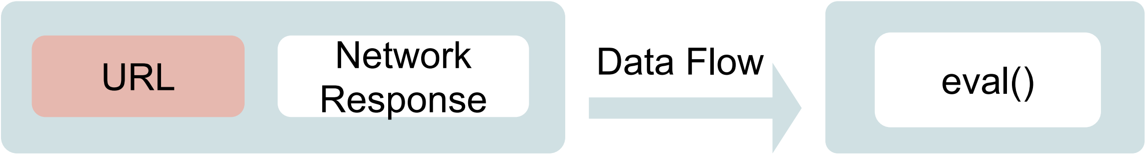
Data Flow

Shell Command

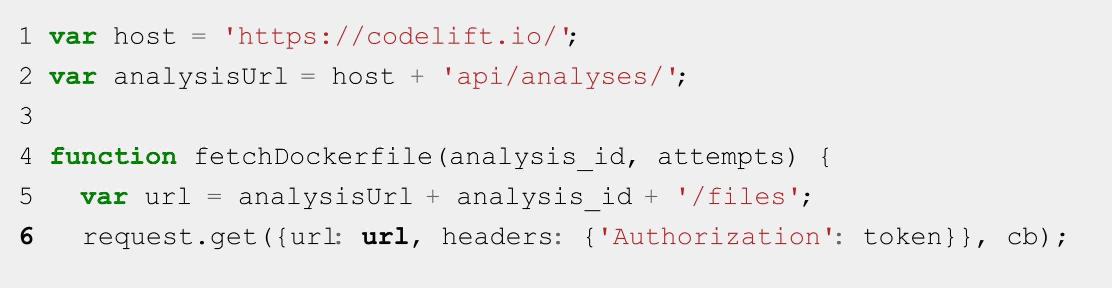
```
class Configuration extends TaintTracking::Configuration {
  Configuration() { this = "Configuration" }
  override predicate isSource(DataFlow::Node source) {
      source = any(Http::RouteHandler rh).getARequestNode()
  }
  override predicate isSink(DataFlow::Node sink) {
      exists(SystemCommandExecution shell | sink = shell. getACommandArgument())
  }
}

// Find flows from web server to shell command
from
Configuration cfg, DataFlow::PathNode source, DataFlow::PathNode sink
where
cfg.hasFlowPath(source, sink) and not source.getNode() = sink.getNode()
select
sink.getNode(), source, sink, "shell command depends on $@.",
source.getNode(), "route handler"
```

Challenges



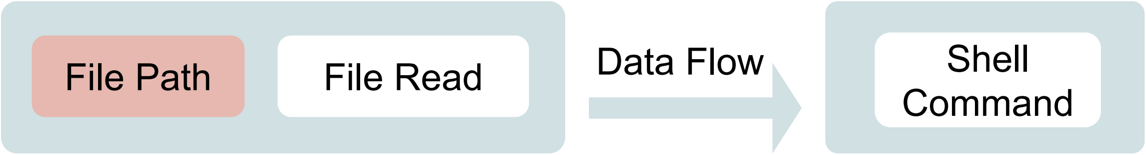
- Determining whether URL is insecure
- Http URLs and URLs with values from untrusted sources are considered insecure
- **Solution:** chain another CodeQL data flow query



```
var host = 'https://codelift.io/';
var analysisUrl = host + 'api/analyses/';

function fetchDockerfile(analysis_id, attempts) {
  var url = analysisUrl + analysis_id + '/files';
  request.get({url: url, headers: {'Authorization': token}}, cb);
```

Challenges

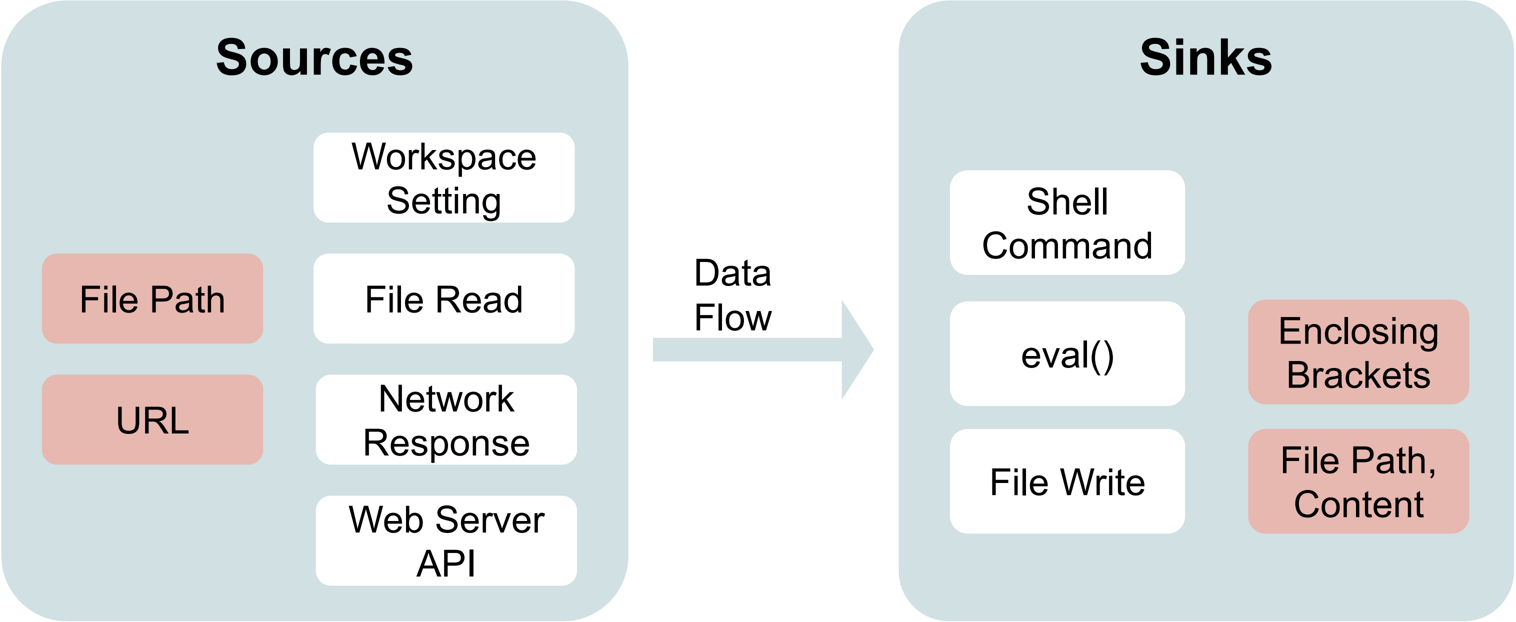


- Determining file path
- Files in VS Code workspace considered untrusted
- Filtering requires some manual effort

```
// taint source
function getCtestPath(cwd) {
  const match =
    fs.readFileSync(cacheFilePath).toString().match(CTEST_RE);
  ...
}
// taint sink
const ctestProcess = child_process.spawn(ctestPath, [
  '--show-only=json-v1',
  ...(!!buildConfig ? ['--build-config', buildConfig] : []),
  ...args, ],
  { cwd }
);
```

Filters

- 4 types of filters applied
- Includes both automated and manual filtering

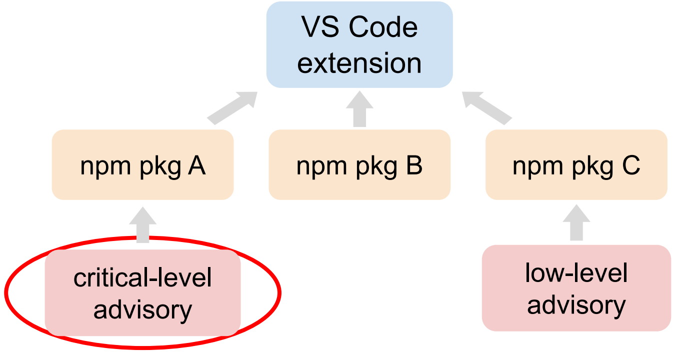


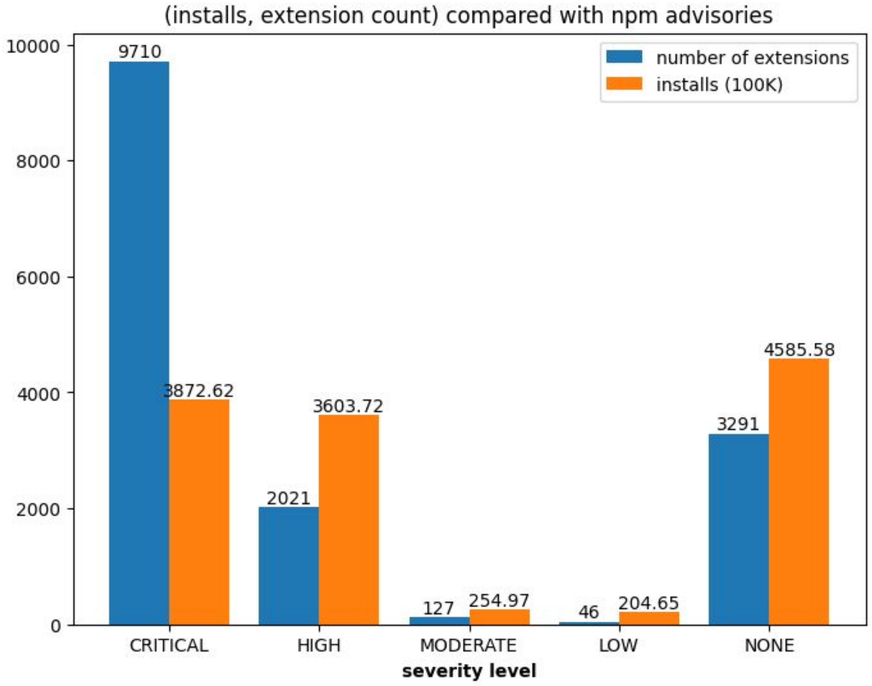
Empirical Study

- Two key research questions:
  - **RQ1:** Are there vulnerabilities in dependencies imported by the extension?
  - **RQ2:** What are the exploitable vulnerabilities (data flows) in the extension itself?
- Dataset collected Feb 2023:
  - 39K total extensions
  - 22K extensions included code

RQ1: Vulnerable npm Dependencies

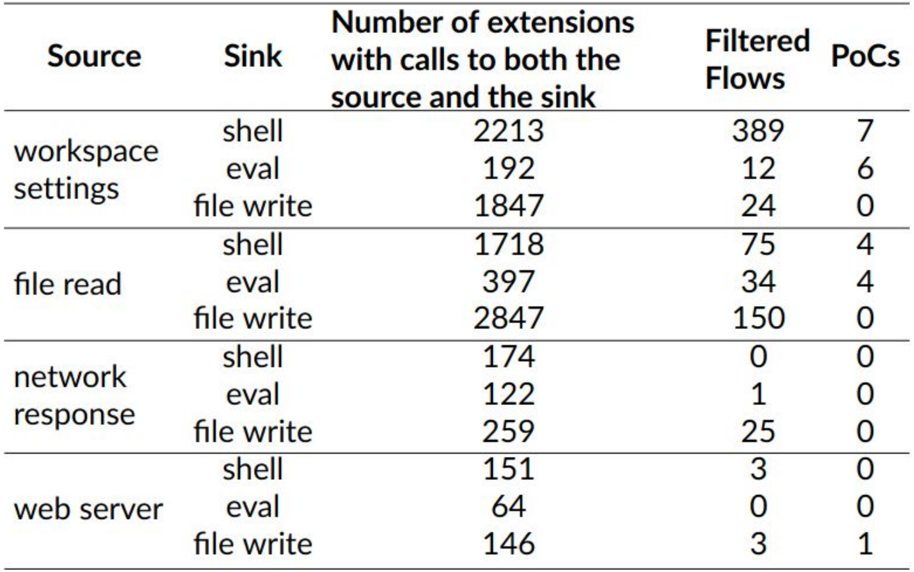
More than 9000 extensions import dependencies with **critical-level** advisories.





RQ2: Data Flow Analysis

716 dangerous data flows

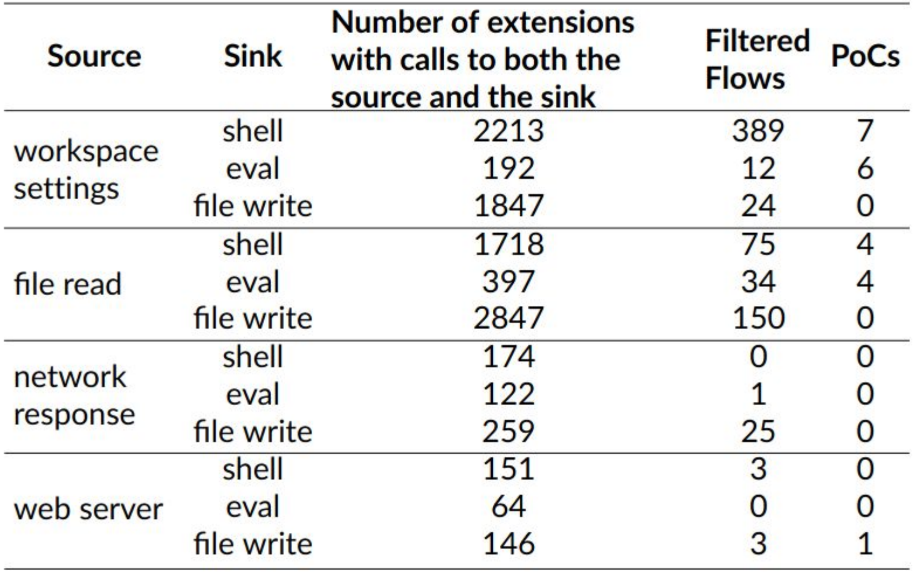


Source	Sink	Number of extensions with calls to both the source and the sink	Filtered Flows	PoCs
workspace settings	shell	2213	389	7
	eval	192	12	6
	file write	1847	24	0
file read	shell	1718	75	4
	eval	397	34	4
	file write	2847	150	0
network response	shell	174	0	0
	eval	122	1	0
	file write	259	25	0
web server	shell	151	3	0
	eval	64	0	0
	file write	146	3	1

RQ2: Data Flow Analysis

21 extensions with verified code execution exploits

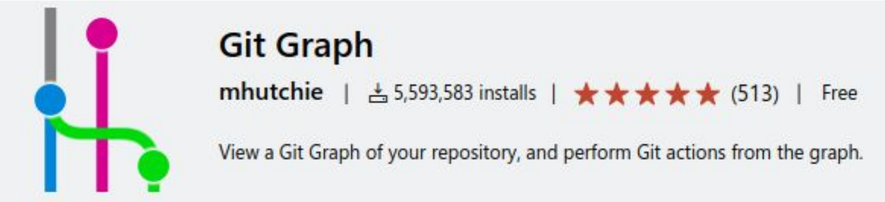
Verified PoC exploits impact more than **6 million** installations



Source	Sink	Number of extensions with calls to both the source and the sink	Filtered Flows	PoCs
workspace settings	shell	2213	389	7
	eval	192	12	6
	file write	1847	24	0
file read	shell	1718	75	4
	eval	397	34	4
	file write	2847	150	0
network response	shell	174	0	0
	eval	122	1	0
	file write	259	25	0
web server	shell	151	3	0
	eval	64	0	0
	file write	146	3	1

** We contacted the developers of the 21 extensions with verified exploits to notify them of the vulnerabilities. 6 developers responded and confirmed vulnerabilities, 3 extensions released new fixed versions.*

Example Vulnerability



Git Graph

mhutchie | 5,593,583 installs | ★★★★★ (513) | Free

View a Git Graph of your repository, and perform Git actions from the graph.

```
// taint source
get gitPaths() {
  const configValue = vscode.workspace.getConfiguration('git').get('path', null);
  ...
}
// taint sink
function getGitExecutable(path) {
  return new Promise((resolve, reject) => {
  resolveSpawnOutput(cp.spawn(path, ['--version'])).then((values) => {
  ...
}
```

PoC Exploit

VS Code Workspace

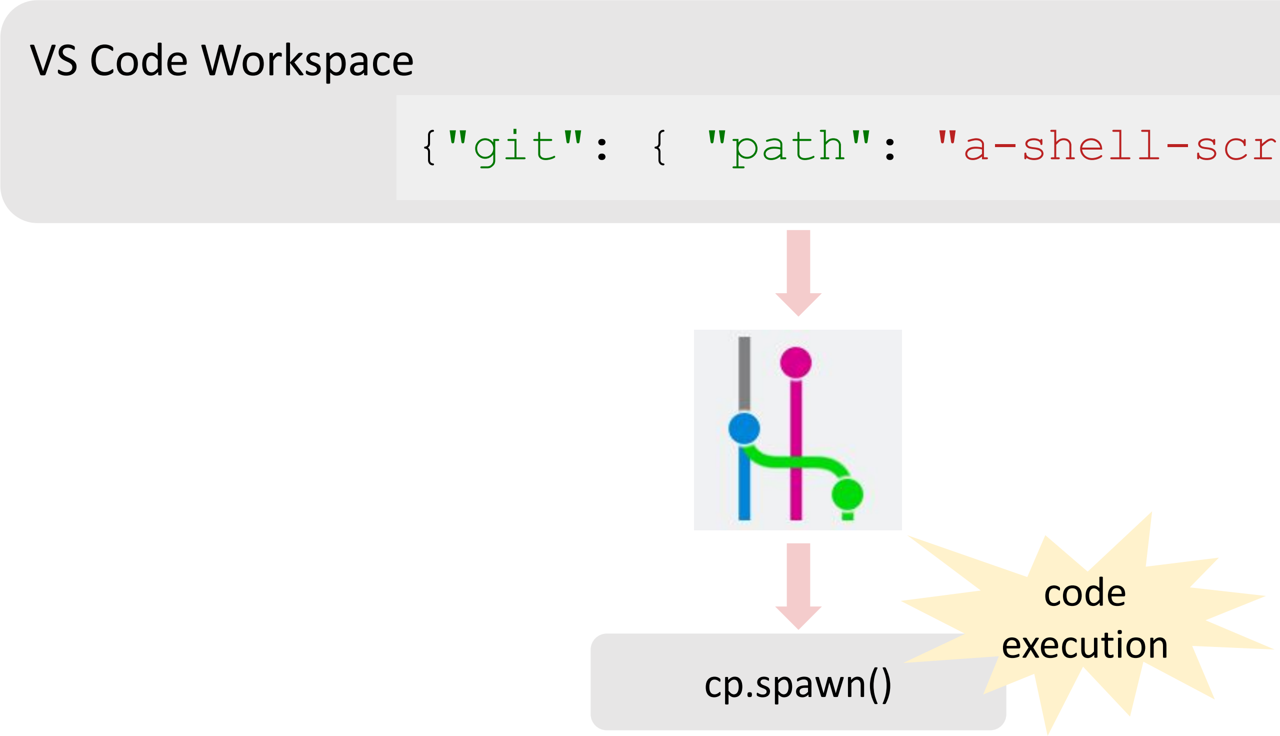
```
{"git": { "path": "a-shell-script.sh"}
```

PoC Exploit

VS Code Workspace

```
{"git": { "path": "a-shell-script.sh"}
```

PoC Exploit



VS Code Workspace

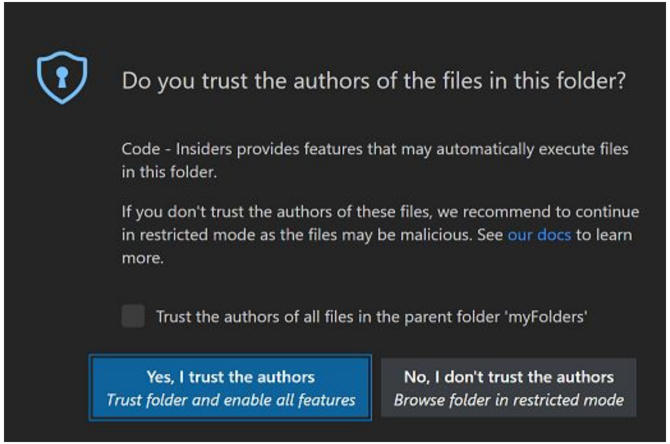
```
{"git": { "path": "a-shell-script.sh"}
```

cp.spawn()

code execution

VS Code Workspace Trust

- Restricted mode prevents code execution for **most** extensions
- However, restricted mode is not a full solution
- Disables or limits valuable
  - VS Code functionality for tasks
  - Debugging
  - Extensions
  - Workspace settings



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

Code - Insiders 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.

□ Trust the authors of all files in the parent folder 'myFolders'

Yes, I trust the authors
Trust folder and enable all features
No, I don't trust the authors
Browse folder in restricted mode

Summary

Our verified exploits impact more than **6 million** installations.

Data flows from workspace settings and files are most likely to be exploited.

One fourth of extensions import **critical-level vulnerabilities** in their dependencies.

Questions?

Our verified exploits impact more than **6 million** installations.

Data flows from workspace settings and files are most likely to be exploited.

One fourth of extensions import **critical-level vulnerabilities** in their dependencies.

Elizabeth Lin
elizabethtlin.com
etlin@ncsu.edu

UntrustIDE repository
github.com/s3c2/UntrustIDE