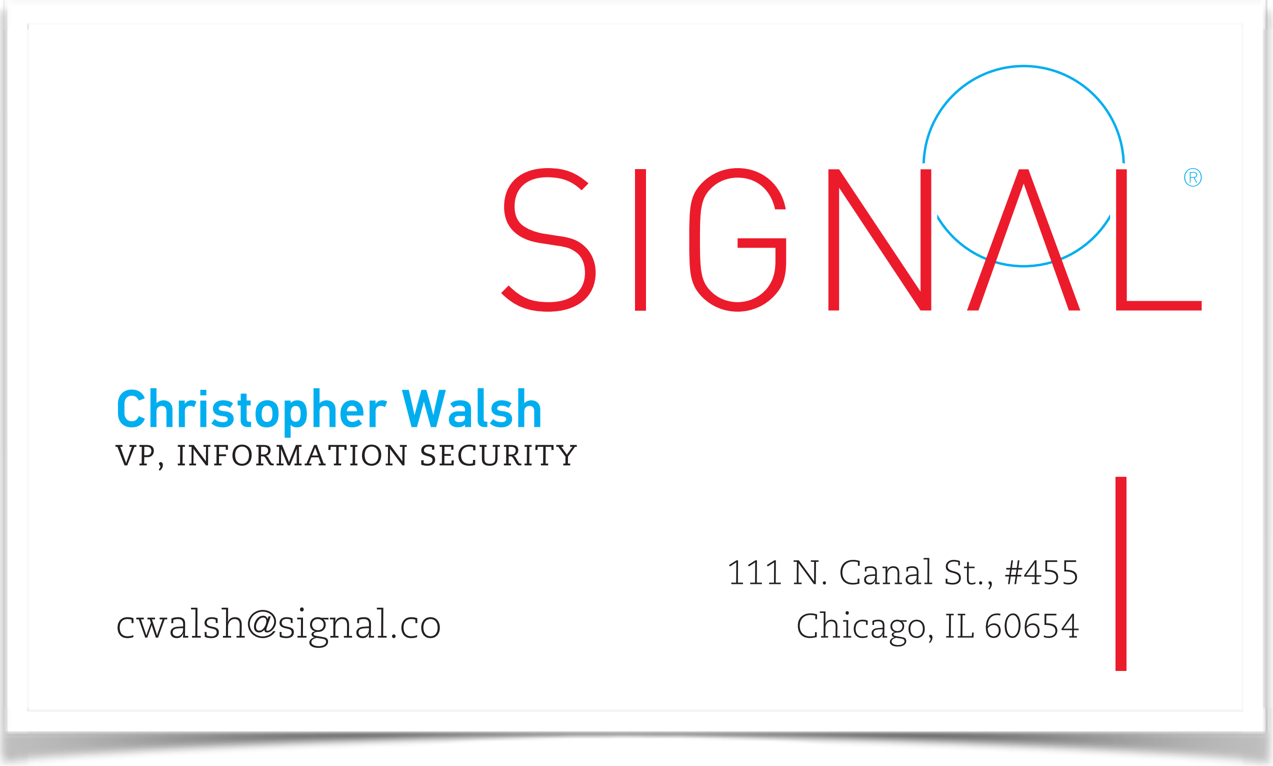
I’m the Security Lead

(…and I’m here to help)

$ whoami



SIGNAL®

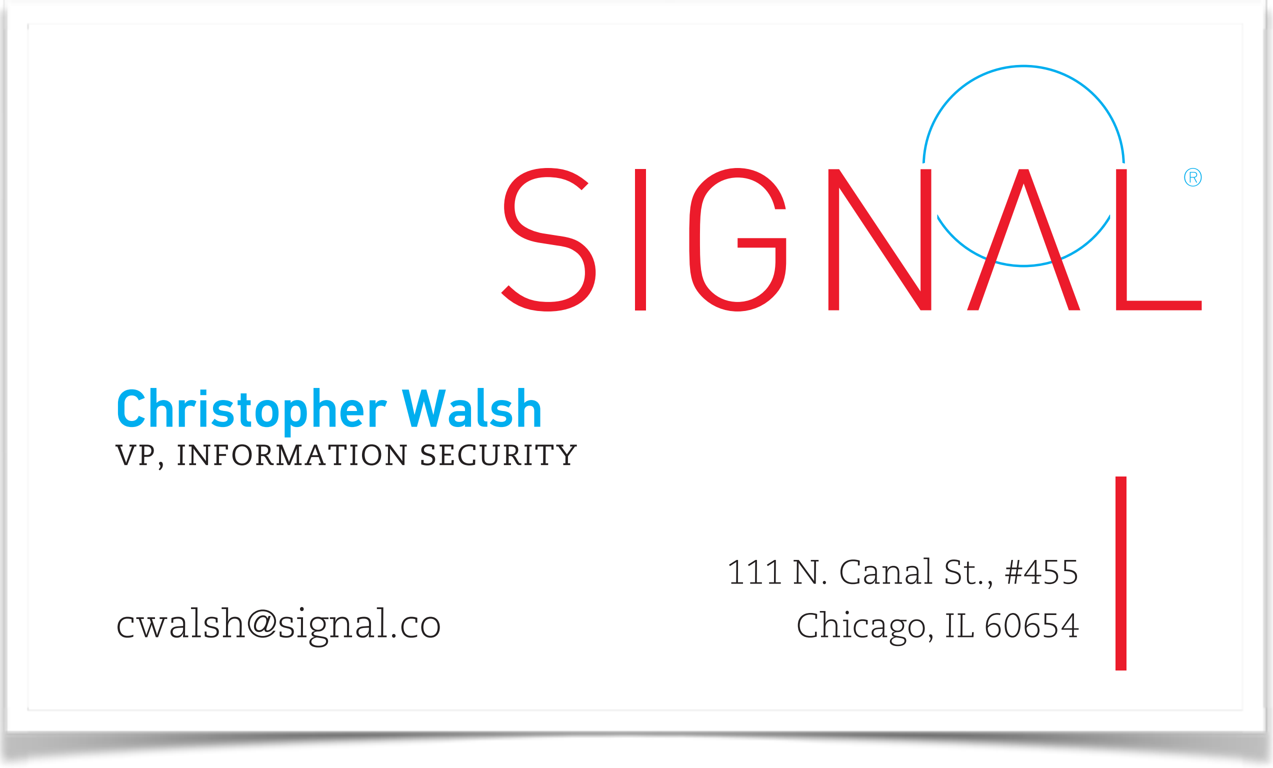
Christopher Walsh
VP, INFORMATION SECURITY

cwalsh@signal.co

111 N. Canal St., #455
Chicago, IL 60654

“Signal is the global leader in cross-channel marketing technology. Thousands of brands and digital agencies around the world rely on Signal’s patented technology to transform data into insights and engage with customers across the web, mobile devices and beyond – all in real time.”

```
$ whoami
```



SIGNAL®

Christopher Walsh
VP, INFORMATION SECURITY

cwalsh@signal.co

111 N. Canal St., #455
Chicago, IL 60654

(We’re an advertising technology company)

$ cat /etc/motd

Traditional “infosec” concerns and responsibilities are in many ways compatible with DevOps.

Success has social and technical elements.

I’ll tell you a (true!) story about my ongoing journey, abstracting lessons I think are generally applicable.

`$ cat /etc/issue`

As a newcomer to the neighborhood, successfully making friends involves observation, humility, trust-building, and contributing to (and subsequently influencing) shared goals.

Actions speak louder than words, but acting *properly,* like speaking politely, depends on unwritten rules.

Identifying these values is the first step to success.

Imposing external values leads to misery.

$ locate values

- ✓ Continuous value delivery
- ✓ Changing requirements are welcome
- ✓ Deliver working code as quickly as possible
- ✓ Business involvement
- ✓ Culture of trust
- ✓ Face-to-face interaction
- ✓ Progress measured by working code
- ✓ Constant pace (not death march, not idling)
- ✓ KISS
- ✓ Team self-organization
- ✓ Continuous, organic, process improvement

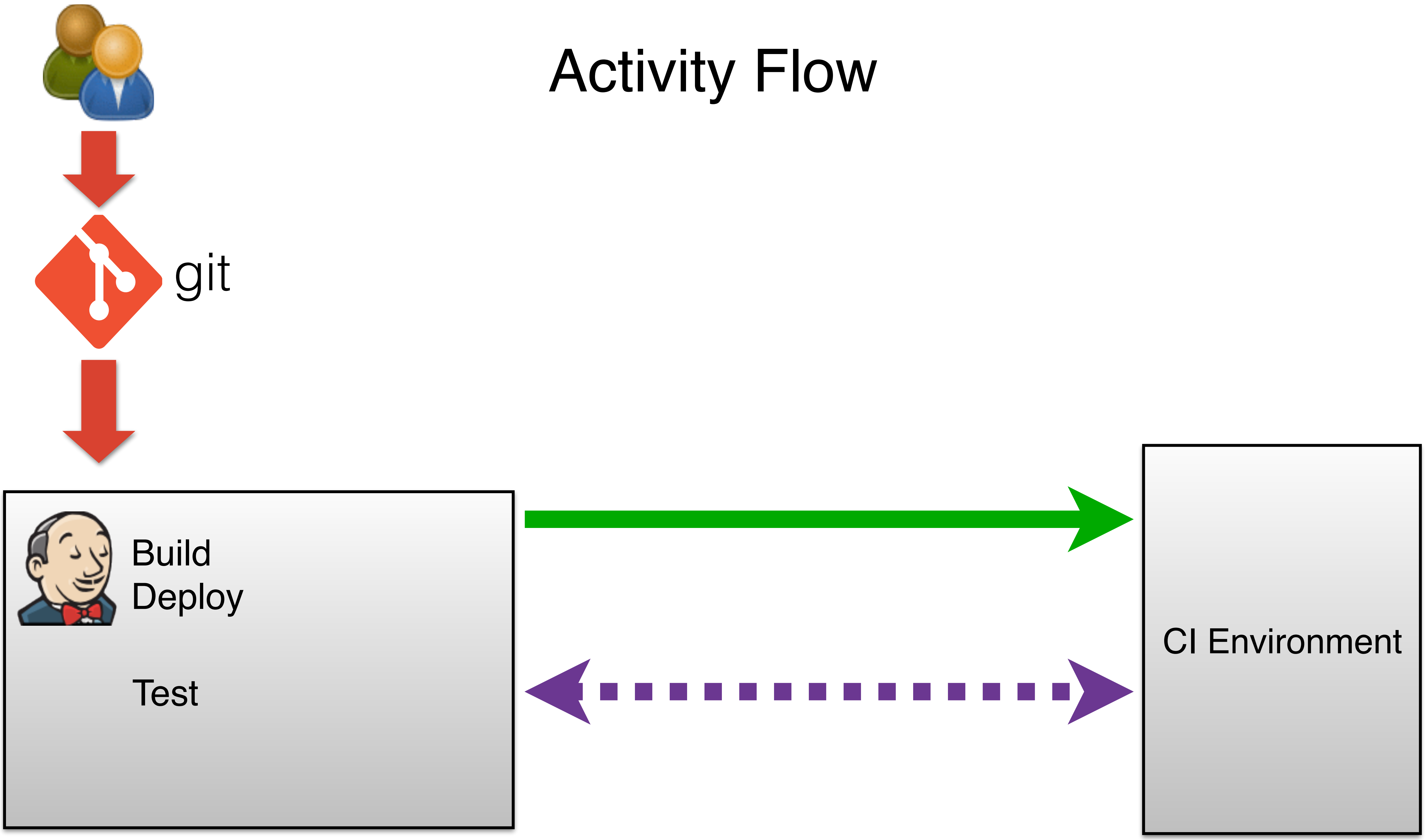
`$ locate values`

- ✓ Continuous value delivery
- ✓ Changing requirements are welcome
- ✓ Deliver working code as quickly as possible
- ✓ Business involvement
- ✓ Culture of trust
- ✓ Face-to-face interaction
- ✓ Progress measured by working code
- ✓ Constant pace (not death march, not idling)
- ✓ KISS
- ✓ Team self-organization
- ✓ Continuous, organic, process improvement

DevOps

- ✓ Continuous value delivery
- ✓ Changing requirements are welcome
- ✓ Deliver working **systems** as quickly as possible
- ✓ Business involvement
- ✓ Culture of trust
- ✓ Face-to-face interaction
- ✓ Progress measured by **tested systems**
- ✓ Constant pace (not death march, not idling)
- ✓ KISS
- ✓ Team self-organization
- ✓ Continuous, organic, process improvement

Activity Flow



git

Build
Deploy

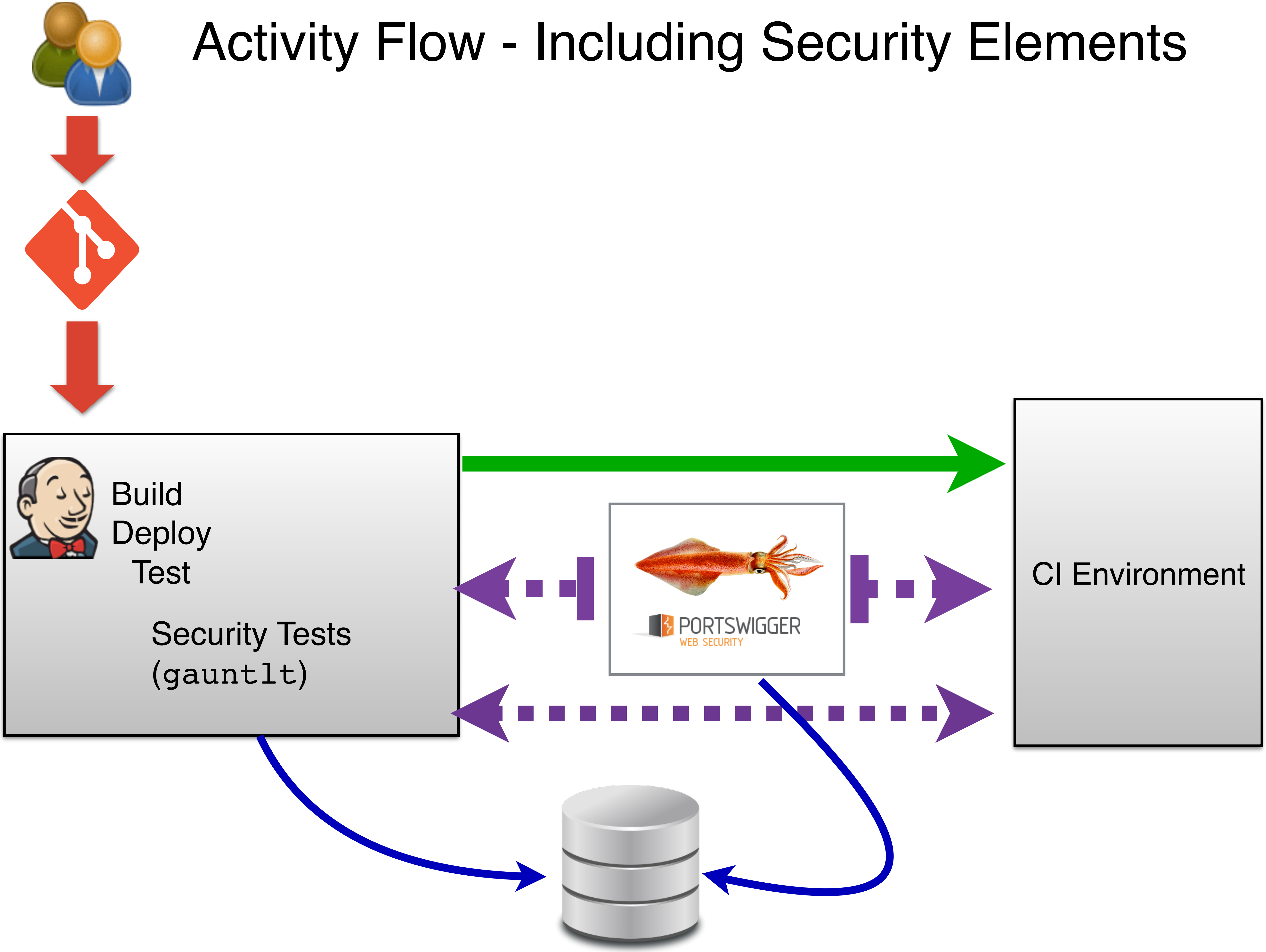
Test

CI Environment

That was easy…so let’s add some security

…but not the wrong kind!

Activity Flow - Including Security Elements



Build
Deploy
Test

Security Tests
(gauntlt)

CI Environment

Security tests as part of CI Pipeline

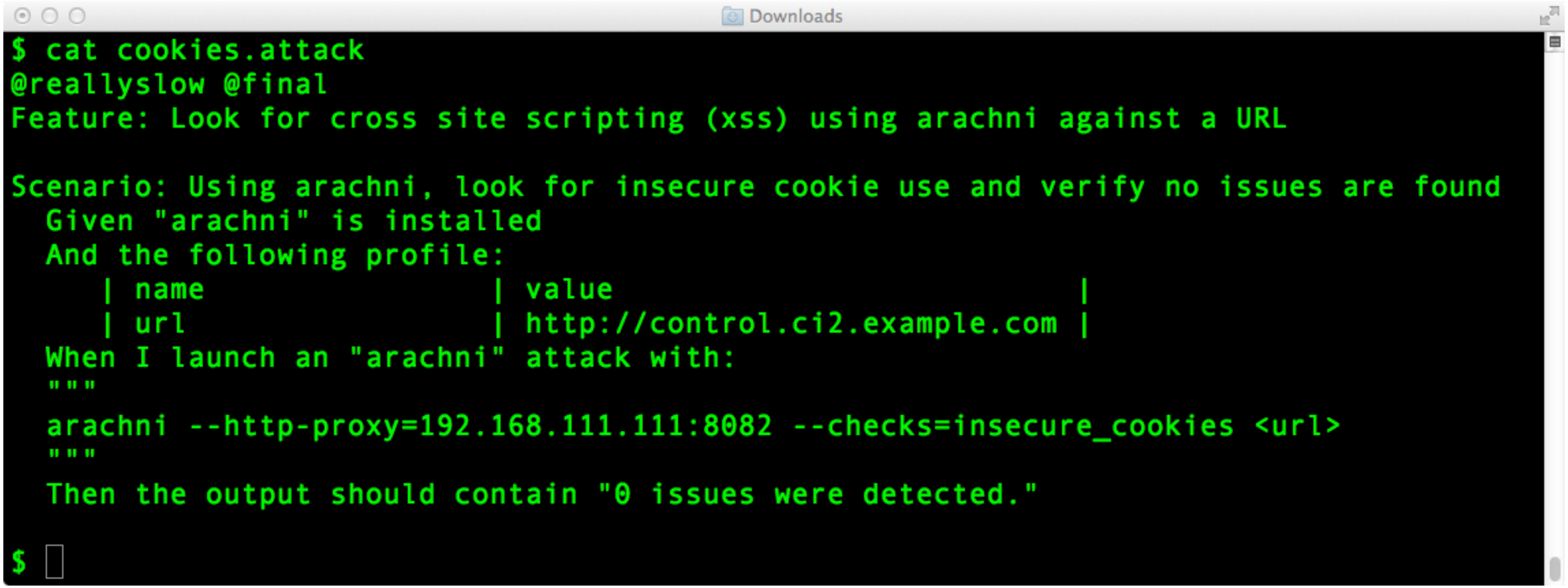
- Triggered by “normal” activity (e.g. smoke test)
- Uses scriptable tools, e.g. gauntlt, with built-in support for:
    - arachni (sometimes run thru a Burp proxy)
    - curl
    - dirbuster
    - sqlmap
    - …

Gauntlt - `gauntlt.org`

Uses attack scripts written in Gherkin.

Makes it easy to write tests - housekeeping is done for you.

Since our non-security tests use Cucumber, my stuff “fits in”, and I can find local help. *This is big for me!*



```
$ cat cookies.attack
@reallyslow @final
Feature: Look for cross site scripting (xss) using arachni against a URL

Scenario: Using arachni, look for insecure cookie use and verify no issues are found
  Given "arachni" is installed
  And the following profile:
     | name                | value                          |
     | url                 | http://control.ci2.example.com |
  When I launch an "arachni" attack with:
  """
  arachni --http-proxy=192.168.111.111:8082 --checks=insecure_cookies <url>
  """
  Then the output should contain "0 issues were detected."

$
```

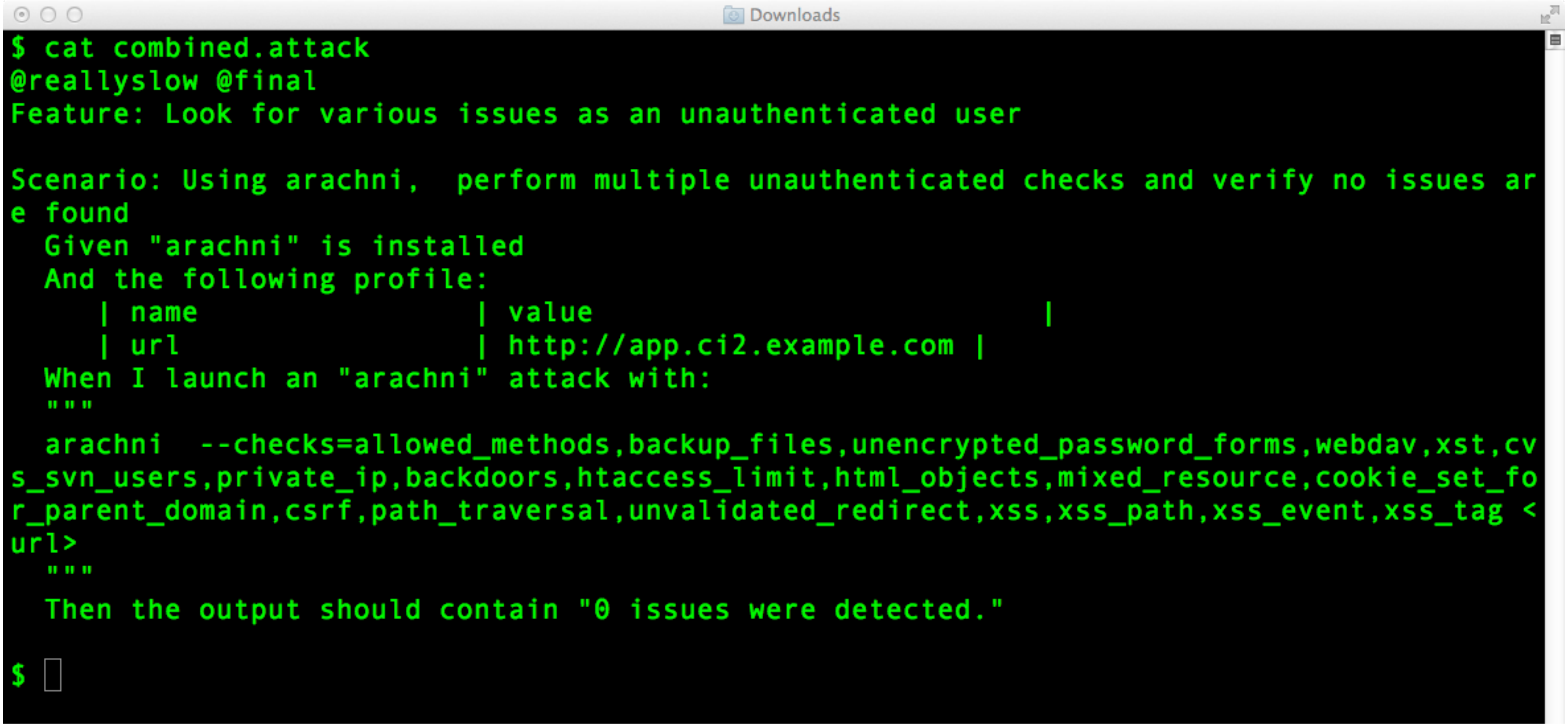
arachni - `www.arachni-scanner.com`

Extensive Ruby framework for web app pen-testing and security evaluation.

directory_listing,
backup_directories,
interesting_responses,
allowed_methods,
localstart_asp,
common_directories,
private_ip,
hsts,
captcha,
insecure_cookies,
mixed_resource,
emails,
html_objects,
cookie_set_for_parent_domain,
password_autocomplete,
credit_card,
http_only_cookies,
unencrypted_password_forms,
cvs_svn_users,
form_upload,
ssn,
backdoors,
http_put,
backup_files,
common_files,
origin_spoof_access_restriction_bypass,
xst,
htaccess_limit,
webdav,
rfi,
xss_dom,
path_traversal,
xss_dom_script_context,
xss_dom_inputs,
csrf,
no_sql_injection_differential,
sql_injection_differential,
xss_path,
xpath_injection,
code_injection,
xss_event,
code_injection_timing,
xss_tag,
unvalidated_redirect,
sql_injection,
os_cmd_injection_timing,
session_fixation,
ldap_injection,
source_code_disclosure,
os_cmd_injection,
sql_injection_timing,
response_splitting,
no_sql_injection,
file_inclusion,
xss_script_context,
xss,

Many checks! Easy to incorporate in gauntlt attack scripts

Example scripted check for multiple issues: CSRF, XSS, Cookie problems, invalidated redirects, stray files, etc.

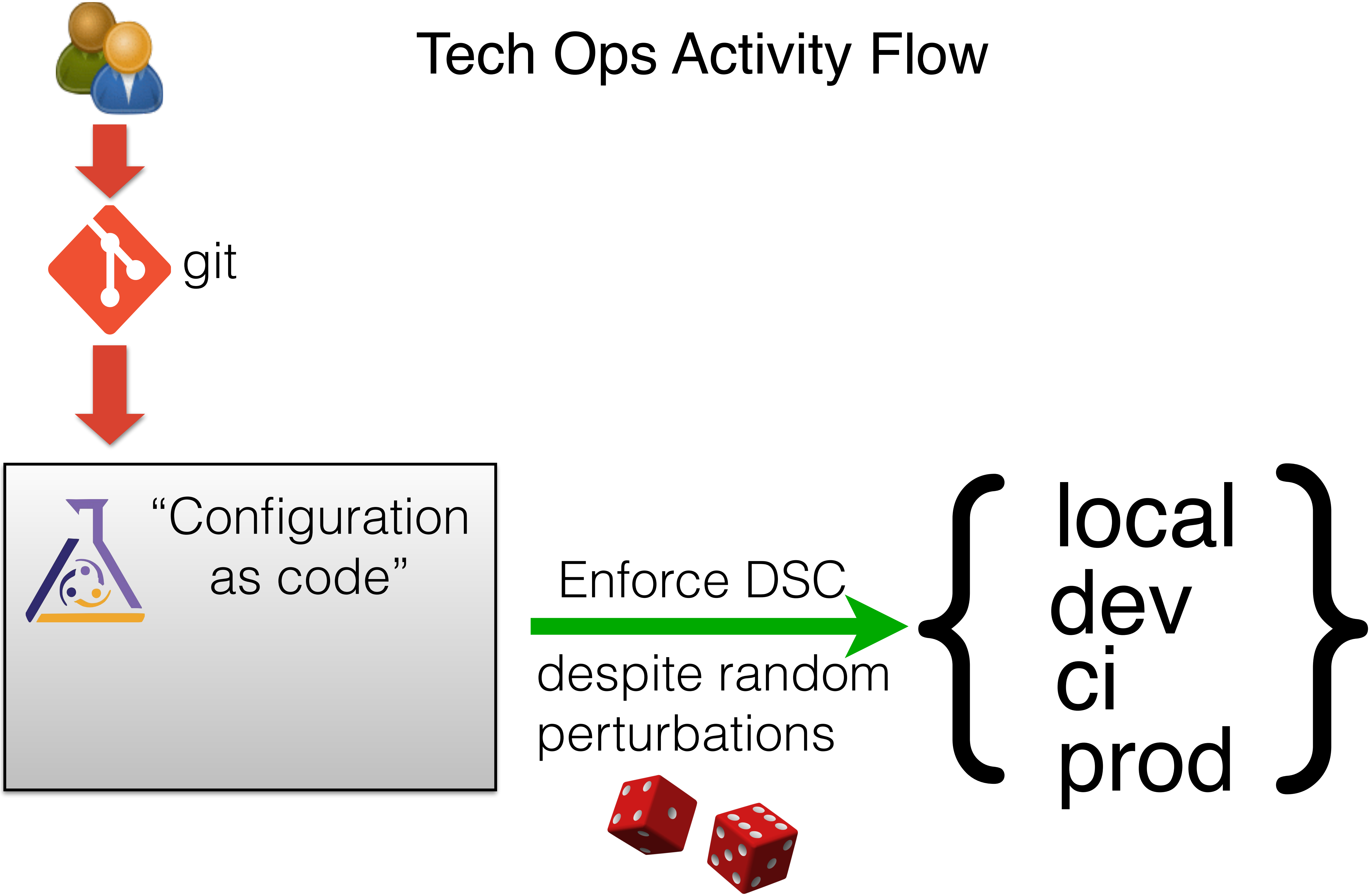


```
$ cat combined.attack
@reallyslow @final
Feature: Look for various issues as an unauthenticated user

Scenario: Using arachni,  perform multiple unauthenticated checks and verify no issues ar
e found
  Given "arachni" is installed
  And the following profile:
     | name                | value                          |
     | url                 | http://app.ci2.example.com |
  When I launch an "arachni" attack with:
  """
  arachni  --checks=allowed_methods,backup_files,unencrypted_password_forms,webdav,xst,cv
s_svn_users,private_ip,backdoors,htaccess_limit,html_objects,mixed_resource,cookie_set_fo
r_parent_domain,csrf,path_traversal,unvalidated_redirect,xss,xss_path,xss_event,xss_tag <
url>
  """
  Then the output should contain "0 issues were detected."

$
```

Benefits

- ✓ Issues are found before customers can experience them.
- ✓ Smaller units of remediation work facilitate flow
- ✓ Use of existing tools/techniques means security work “fits in”

Tech Ops Activity Flow



git

“Configuration as code”

Enforce DSC

despite random perturbations

{ local dev ci prod }

Challenges

- Complexity/dependencies —> Slowness —> WIP
- Common mechanism —> Speed, but with LCD capabilities

Choices

- Degree to which heterogeneity is tolerated
- Address complexity via architecture (e.g., microservices)?

Audit Considerations

Change Management

- Fine-grained audit logs due to automation.
- Standard (pre-authorized) changes.
- Same way, every time.
- Config as code + code review = all config changes reviewed

Least Privilege

- Automation —> *fewer* ppl, doing *fewer* things manually

Asset Management/CMDB

- Comes “free” with infrastructure automation!

“But Chris, we’re not an Agile/DevOps shop…”

Yeah, but you can influence that…

- “Go DevOps” with a low-risk POC

- Introduce influencers to DevOps ideas
    - Bring *The Phoenix Project* to the office

Credits

Mordac: http://search.dilbert.com/comic/Mordac%20The%20Preventer
Agile: http://agilemanifesto.org/principles.html
DevOps Principles: http://theagileadmin.com/2010/10/15/a-devops-manifesto/
Easy Button: http://www.staples-3p.com/s7/is/image/Staples/s0105150_sc7?$splssku$
Jenkins logo: http://jenkins-ci.org/
git logo: http://twitter.com/jasonlong
puppet logo: https://puppetlabs.com/company/news/media-kit
dice: https://openclipart.org/detail/25207/two-red-dice