Bandit and Gosec – Security Linters

All Things Open 2018

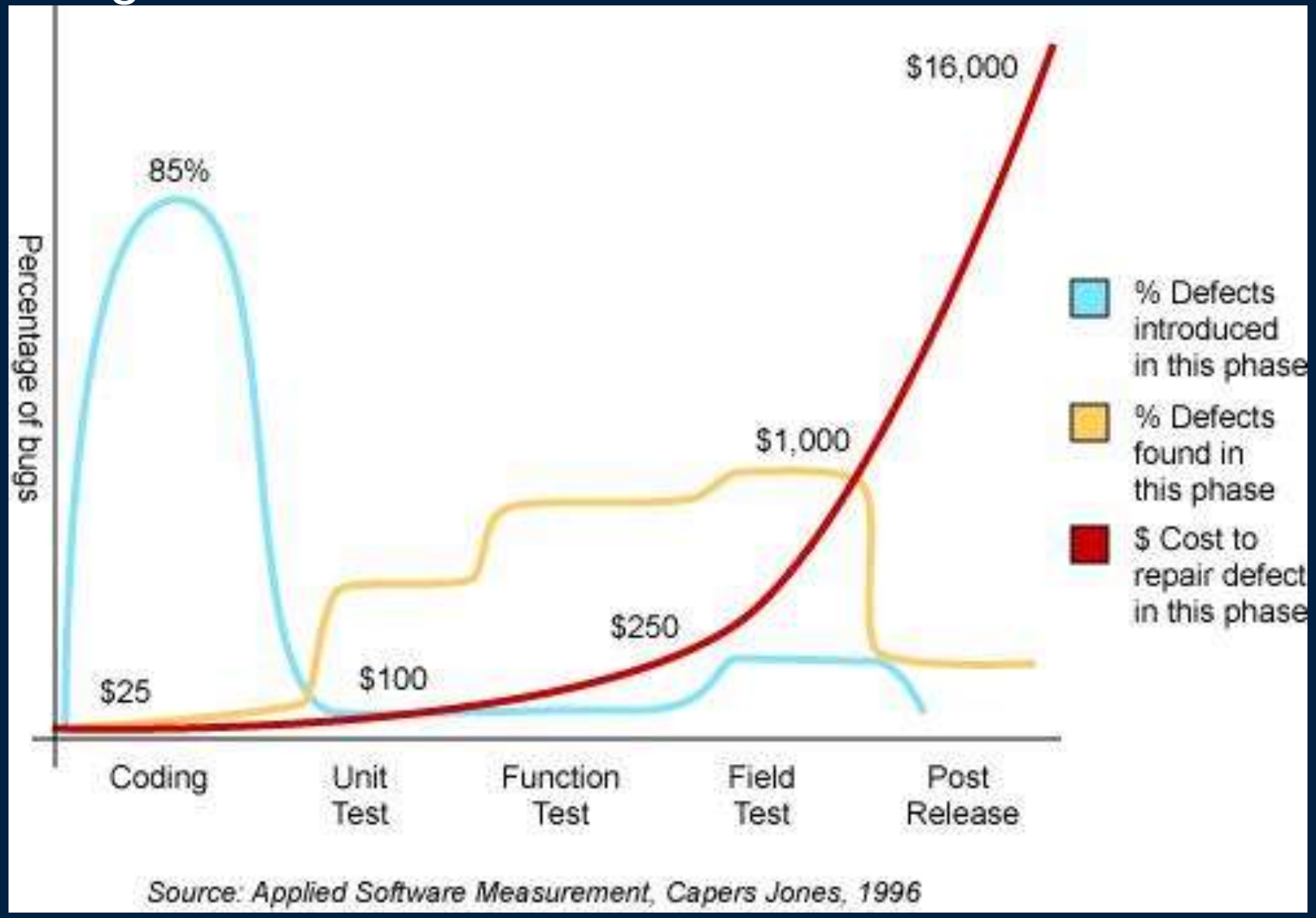
Eric Brown

Staff Open Source Engineer / Open Source Technology Center

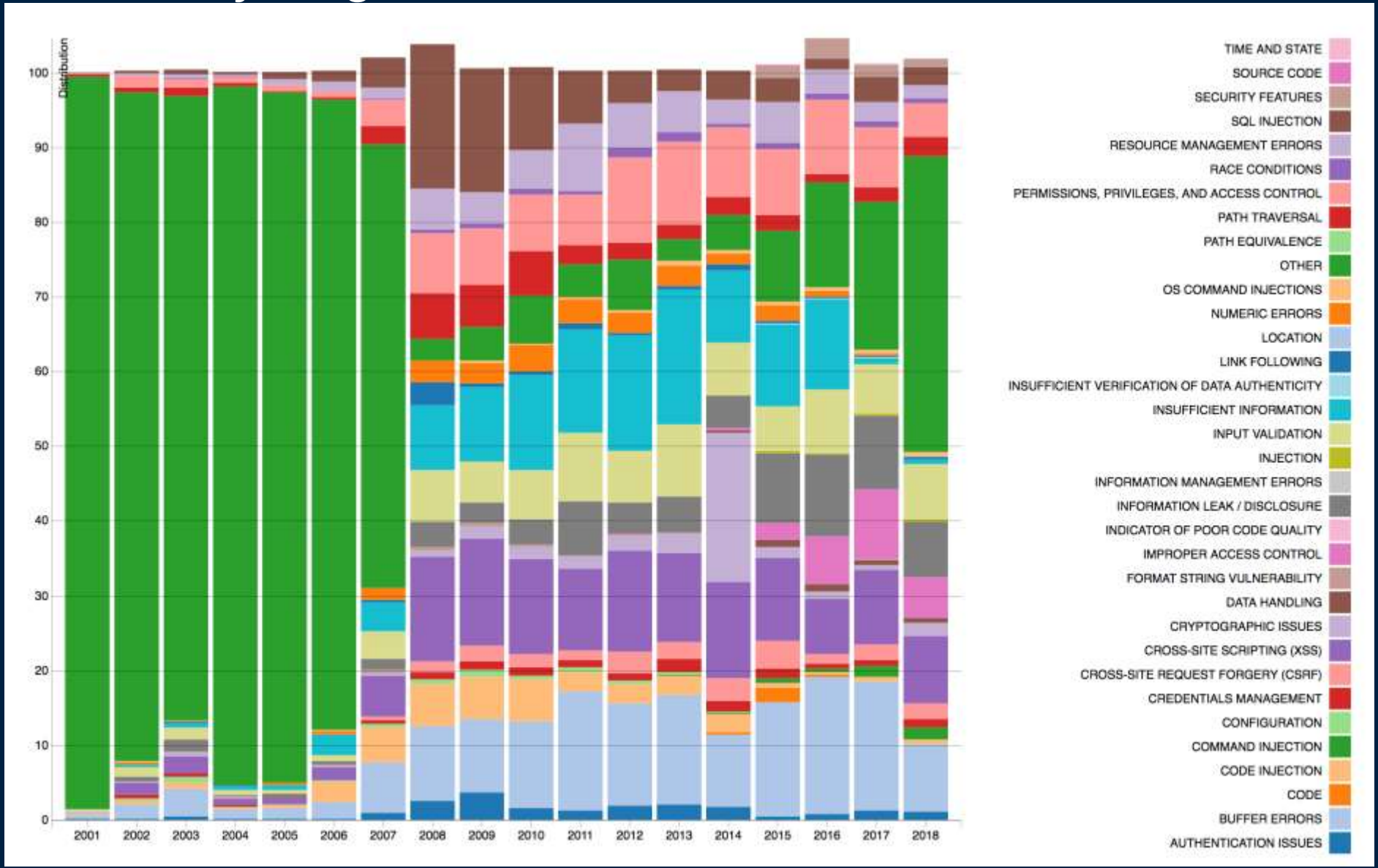
October 22nd 2018

Moar Security

Cost to Fix Bugs



Types of Security Bugs



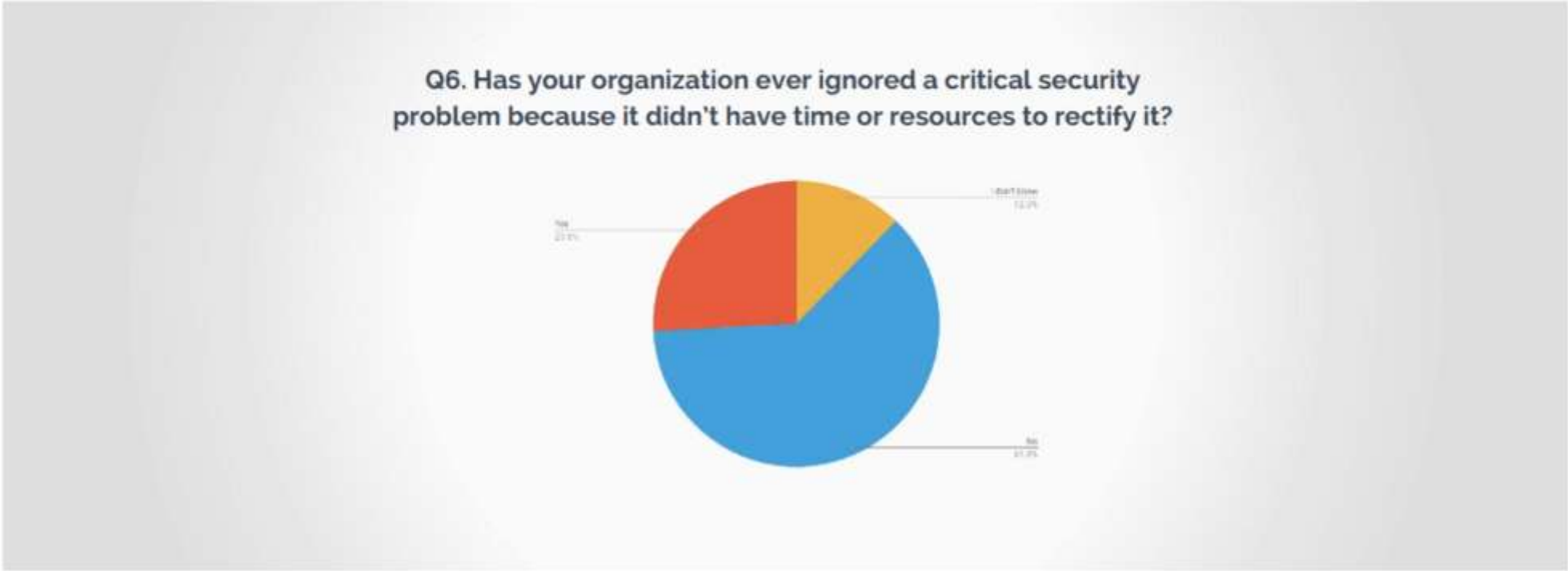
Source: National Institute of Standards and Technology

What’s the Struggle?

26% of Companies Ignore Security Bugs Because They Don’t Have the Time to Fix Them

By Catalin Cimpanu

May 10, 2018 10:00 AM 3



Source: https://www.bleepingcomputer.com/news/security/26-percent-of-companies-ignore-security-bugs-because-they-don-t-have-the-time-to-fix-them/

How to Catch Security Bugs Earlier

Education

How to Catch Security Bugs Earlier

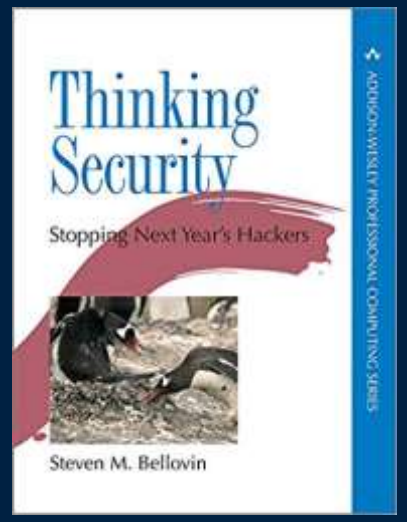
Education

- Courses

How to Catch Security Bugs Earlier

Education

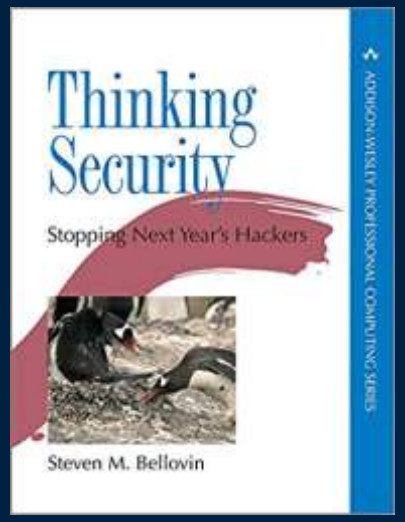
- Courses
- Books



How to Catch Security Bugs Earlier

Education

- Courses
- Books
- Conference talks ☺



How to Catch Security Bugs Earlier

Tooling

How to Catch Security Bugs Earlier

Tooling

- Unit testing

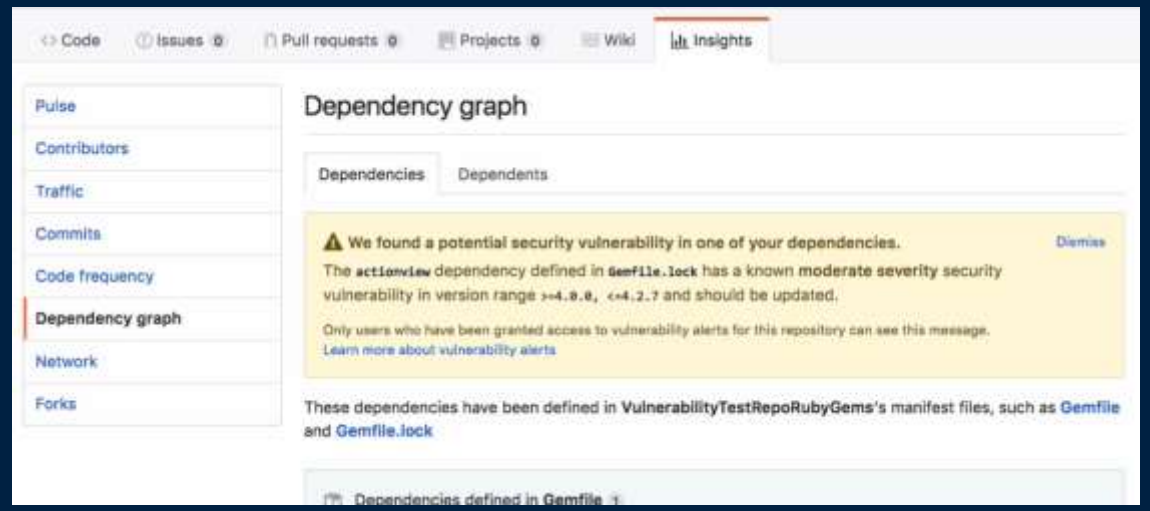
Travis CI

How to Catch Security Bugs Earlier

- Unit testing
- Dependency checking

Tooling

Travis CI

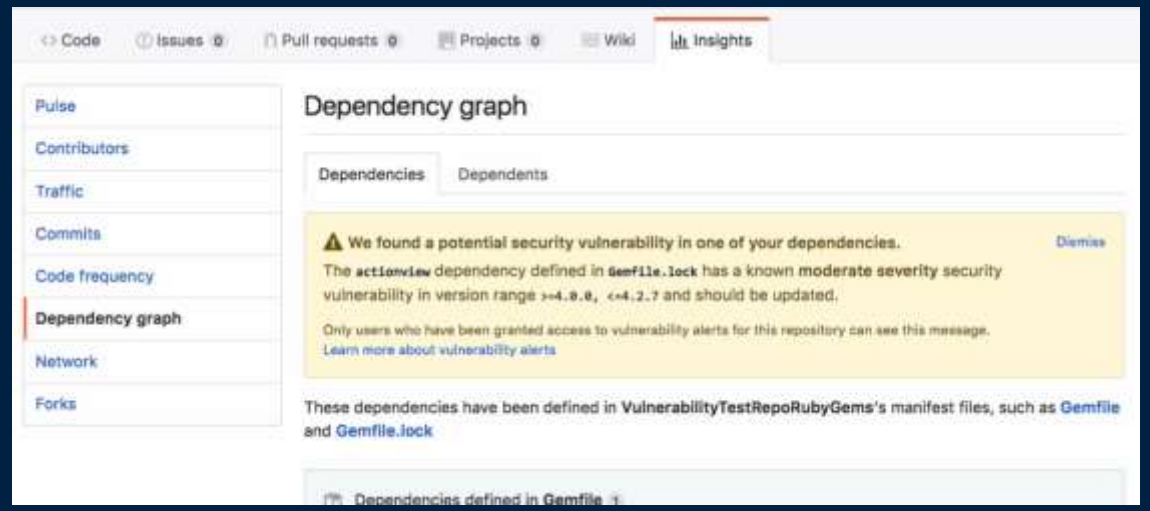


How to Catch Security Bugs Earlier

Tooling

- Unit testing
- Dependency checking
- Linters

Travis CI



What is a Linter?

- A.K.A Static Code Analysis (SCA)
- Scans source code statically without running or building the code
- Linters focus on finding common mistakes
- Might make use of a compilers AST (abstract syntax tree)
- Popular examples:

checkstyle

Pylint
Star your Python code!

ESLint

When to Run Linters?

- When coding
  - Find linter plugins to your favorite IDE
  - Fix as you code
- As part of a CI (continuous integration) system
  - Catch mistakes when code is pushed to a repository for review
  - Travis-CI, Circle CI, Jenkins, etc

Downsides

- Linters are imperfect
  - False positives
- Extra time
  - Requires time to interpret the results
  - Requires time during CI to scan code

What is Bandit?

- A Python security linter
- Project started early 2015
- Originally designed for OpenStack but now part of the Python Code Quality Authority
- Python 2.7, 3.5, 3.6, 3.7 compatible
- Runs on Linux and macOS
- Easy to write new plugins
- Low resource requirements
- Runs quickly

Bandit: Issues It Finds

- Finds common security issues in Python code
  - Use of assert
  - Hardcoded passwords
  - Command injection
  - Insecure temporary file usage
  - Promiscuous file permissions
  - Usage of unsafe functions/libraries
  - Binding to all interfaces
  - Weak cryptography
  - Bad SSL versions
  - Requests without certificate validation
  - Use of insecure protocols

Bandit: Formatters

- Many different output formatters
  - CSV
  - HTML
  - JSON
  - Text
  - XML
  - YAML
  - Custom: using predefined tags

Bandit: Other Features

- Customizable filters on severity and confidence levels
- Allows creation of profiles to scan a subset of plugin tests
- Adjust lines of context shown in output
- Group results by file or vulnerability
- Output delta reports of previous scans
- Allows marking false positives using the "# ***nosec"*** comment

Bandit: Config

GitHub, Inc. (US) | https://github.com/Netflix-Skunkworks/historical/blob/master/tox.ini

am - Open ... LWN.net OpenStack GitHub 中級4-1秋 | aomi3 n... Travel Deals, Cou

```
[testenv:bandit]
basepython = python3
skip_install = true
deps =
    bandit
commands =
    bandit --ini tox.ini -r historical

[bandit]
skips = B101
```

Bandit: Config

```
browne-a01:workspace browne$ bandit-config-generator -o bandit.yaml
[ INFO]: Successfully wrote profile: bandit.yaml
```

demo1.py | ssh_no_host_key_verification.py | bandit.yaml

```yaml
    - subprocess.run
  subprocess_without_shell_equals_true:
    no_shell:
    - os.execl
    - os.execle
    - os.execlp
    - os.execlpe
    - os.execv
    - os.execve
    - os.spawnlp
    - os.spawnlpe
    - os.spawnv
    - os.spawnve
    - os.spawnvp
    - os.spawnvpe
    - os.startfile
    shell:
    - os.system
    - os.popen
    - os.popen2
    - os.popen3
    - os.popen4
    - popen2.popen2
    - popen2.popen3
    - popen2.popen4
    - popen2.Popen3
    - popen2.Popen4
    - commands.getoutput
    - commands.getstatusoutput
    subprocess:
    - subprocess.Popen
    - subprocess.call
    - subprocess.check_call
    - subprocess.check_output
    - subprocess.run
  try_except_continue:
    check_typed_exception: false
  try_except_pass:
    check_typed_exception: false
  weak_cryptographic_key:
    weak_key_size_dsa_high: 1024
    weak_key_size_dsa_medium: 2048
    weak_key_size_ec_high: 160
    weak_key_size_ec_medium: 224
    weak_key_size_rsa_high: 1024
    weak_key_size_rsa_medium: 2048
```

Bandit: Dependents

Bandit: Integrations

- SublimeLinter-bandit – Sublime Text linter plugin
- flake8-bandit – Flake8 plugin
- pyreportcard – Report card of Python projects quality
- bandit-plugin- - Hudson/Jenkins plugin

Wishlist:

- Visual Studio Code
- Vim
- Emacs
- Atom

Bandit: Sublime Text Linter

~/workspace/bandit/examples/weak_cryptographic_key_sizes.py — workspace

```
50  # Also incorrect: without keyword args
51  dsa.generate_private_key(512,
52
53  1 error
    bandit: B505 - DSA key sizes below 1024 bits are considered breakable.
54
55  rsa.generate_private_key(3,
```

```
16  rsa.generate_private_key(public_exponent=65537,
17                           key_size=2048,
18                           backend=backends.default_backend())
19  pycrypto_dsa.generate(bits=2048)
20  pycrypto_rsa.generate(bits=2048)
21  pycryptodomex_dsa.generate(bits=2048)
22  pycryptodomex_rsa.generate(bits=2048)
23
24  # Also correct: without keyword args
25  dsa.generate_private_key(4096,
26                           backends.default_backend())
27  ec.generate_private_key(ec.SECP256K1,
28                          backends.default_backend())
29  rsa.generate_private_key(3,
30                           4096,
31                           backends.default_backend())
44                           backend=backends.default_backend())
45  pycrypto_dsa.generate(bits=1024)
46  pycrypto_rsa.generate(bits=1024)
47  pycryptodomex_dsa.generate(bits=1024)
48  pycryptodomex_rsa.generate(bits=1024)
49
50  # Also incorrect: without keyword args
51  dsa.generate_private_key(512,
52
53  1 error
    bandit: B505 - DSA key sizes below 1024 bits are considered breakable.
54
55  rsa.generate_private_key(3,
56                           512,
57                           backends.default_backend())
58  pycrypto_dsa.generate(512)
59  pycrypto_rsa.generate(512)
60  pycryptodomex_dsa.generate(512)
61  pycryptodomex_rsa.generate(512)
62
63  # Don't crash when the size is variable
64  rsa.generate_private_key(public_exponent=65537,
65                           key_size=some_key_size,
66                           backend=backends.default_backend())
67
```

no_host_key_verification.py
nonsense.py
nonsense2.py
nosec.py
okay.py
os-chmod-py2.py
os-chmod-py3.py
os-exec.py
os-popen.py
os-spawn.py
os-startfile.py
os_system.py
paramiko_injection.py
subprocess_shell.py
telnetlib.py
tempnam.py
try_except_continue.py
try_except_pass.py
unverified_context.py
urlopen.py
weak_cryptographic_key_sizes
wildcard-injection.py
xml_etree_celementtree.py
xml_etree_elementtree.py
xml_expatbuilder.py
xml_expatreader.py
xml_lxml.py
xml_minidom.py
xml_pulldom.py
xml_sax.py
xml_xmlrpc.py

bandit(error), pep8(ok), pylint(1|3), Missing module docstring (missing-docstring), Line 1, Column 1

Spaces: 8 Python

Bandit: Sublime Text Linter

Package Control

Search

BROWSE

SublimeLinter-bandit

by SublimeLinter ST3

SublimeLinter plugin for Python, using bandit

LABELS linting, SublimeLinter, python, security

Details

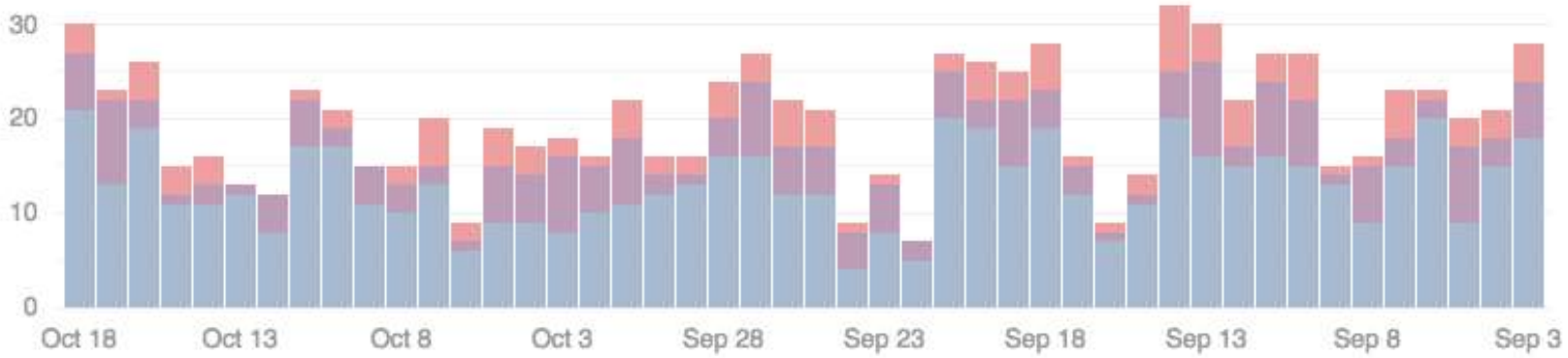
VERSION 1.1.1
HOMEPAGE github.com
ISSUES github.com

MODIFIED 7 months ago

LAST SEEN 1 hour ago
FIRST SEEN 2 years ago

Installs

TOTAL 12K WIN 8K OS X 3K LINUX 1K



Bandit Alternatives

- Very few Python security linters
- Others:
  - RATS (rough-auditing-tool-for-security) - unmaintained as of 2013
  - HP Fortify - not free

Bandit in Action

```
import paramiko
import re


class Oa:
    def __init__(self, host, username, password):
        self.host, self.username, self.password = host, username, password

    def getid(self, name):
        host, username, password = self.host, self.username, self.password
        s = paramiko.SSHClient()
        s.set_missing_host_key_policy(paramiko.AutoAddPolicy)
        s.connect(host, username=username, password=password)
        stdin, stdout, stderr = s.exec_command('show server list')
        id = None
        for line in stdout:
            if name.lower() in line.lower():
                matchid = re.search('([0-9]*) %s.*' % name.lower(),
                                    line.lower())
                id = matchid.group(1)
                break
        s.close()
        return id
```

Bandit in Action

```
bash-3.2$
```

Bandit in Action

```
bash-3.2$
```

Contributing to Bandit

- https://github.com/PyCQA/bandit
- Bandit is participating in Hacktoberfest
- Core maintainers:
  - Myself – ericwb
  - Ian StapleTon Cordasco – sigmavirus24
  - Gage Hugo – ghugo
  - Luke Hinds – lukehinds
- Ideas:
  - Documentation could be improved
  - More integrations with IDEs

What is Gosec?

- A Golang security linter
- Project started mid 2016
- Started by one of the creators of Bandit
- Runs on Linux and macOS
- Low resource requirements
- Runs quickly

Gosec: Issues It Finds

- Finds common security issues in Go code
  - Hardcoded credentials
  - Binding to all interfaces
  - SQL injection
  - Command injection
  - Insecure temporary file usage
  - Promiscuous file permissions
  - Usage of unsafe functions/libraries
  - Weak cryptography
  - Bad TLS/SSL versions
  - Ignoring host keys

Gosec: Formatters

- Many different output formatters
  - CSV
  - JSON
  - Text
  - XML
  - YAML

Gosec: Other Features

- Customizable filters on severity and confidence levels
- Allows creation of profiles to scan a subset of plugin tests
- Group results by severity
- Allows marking false positives using the "# ***nosec"*** comment
- Tool to generate TLS rules according to Mozilla recommendations
  - https://statics.tls.security.mozilla.org/server-side-tls-conf.json

Gosec: Integrations

- SublimeLinter-contrib-gometallinter – Sublime Text linter plugin
- gometallinter – collection of linters
- – via a gometallinter plugin

Visual Studio Code

Wishlist:

- Go Report Card
- Vim
- Emacs
- Atom

Gosec: Sublime Text Linter

~/workspace/go/src/github.com/hashicorp/vault/builtin/logical/pki/cert_util.go — workspace

```
1 warning
gometalinter: warning - Blacklisted import crypto/sha1: weak cryptographic primitive,MEDIUM,HIGH (gosec)
```

```
cert_util.go:
    10:1    warning gometalinter:warning    Blacklisted import crypto/sha1: weak cryptographic primitive,MEDIUM,HIGH (gosec)
  1377:1    warning gometalinter:warning    Use of weak cryptographic primitive,MEDIUM,HIGH (gosec)
```

Gosec: Visual Studio Code

cert_util.go — workspace

EXPLORER

OPEN EDITORS
- validation.go go/src/github.com/hashico...
- cert_util.go go/src/github.com/hashi... 2
- User Settings ~/Library/Application Sup...

WORKSPACE
- cryptography
- docs
- fail2ban
- frisk
- go
  - bin
  - pkg
  - src
    - github.com
      - acroca
      - derekparker
      - golang
      - hashicorp
        - vault
          - .github
          - .hooks
          - api
          - audit
          - builtin
            - audit
            - credential
            - logical
              - aws
              - cassandra
              - consul
              - database
              - mongodb
              - mssql
              - mysql
              - nomad

OUTLINE

validation.go | cert_util.go | User Settings

```go
package pki

import (
    "bytes"
    "context"
    "crypto"
    "crypto/ecdsa"
    "crypto/rand"
    "crypto/rsa"
    "crypto/sha1"
    "crypto/x509"
    "crypto/x509/pkix"
    "encoding/asn1"
    "encoding/base64"
    "encoding/pem"
    "fmt"
    "net"
    "net/url"
    "regexp"
    "strconv"
    "strings"
    "time"

    "github.com/hashicorp/errwrap"
    "github.com/hashicorp/vault/helper/certutil"
```

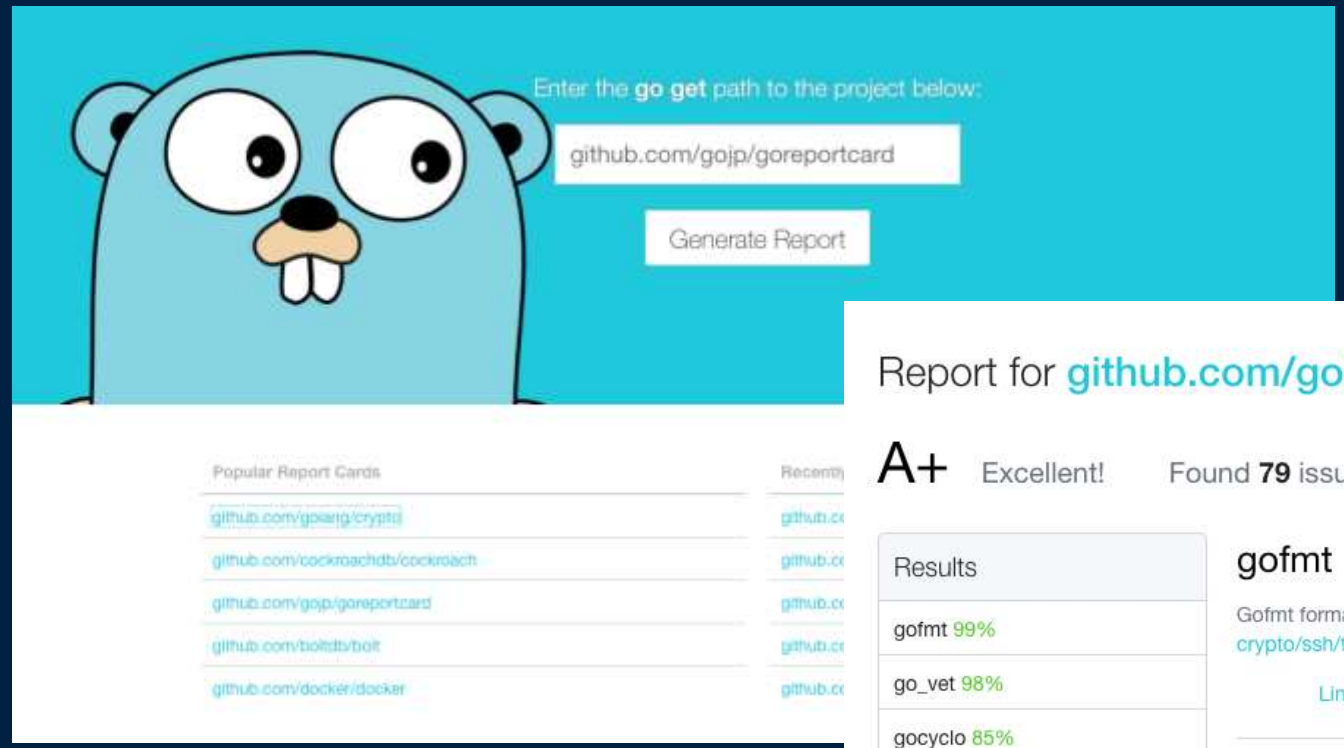
PROBLEMS 2 | OUTPUT | DEBUG CONSOLE | TERMINAL

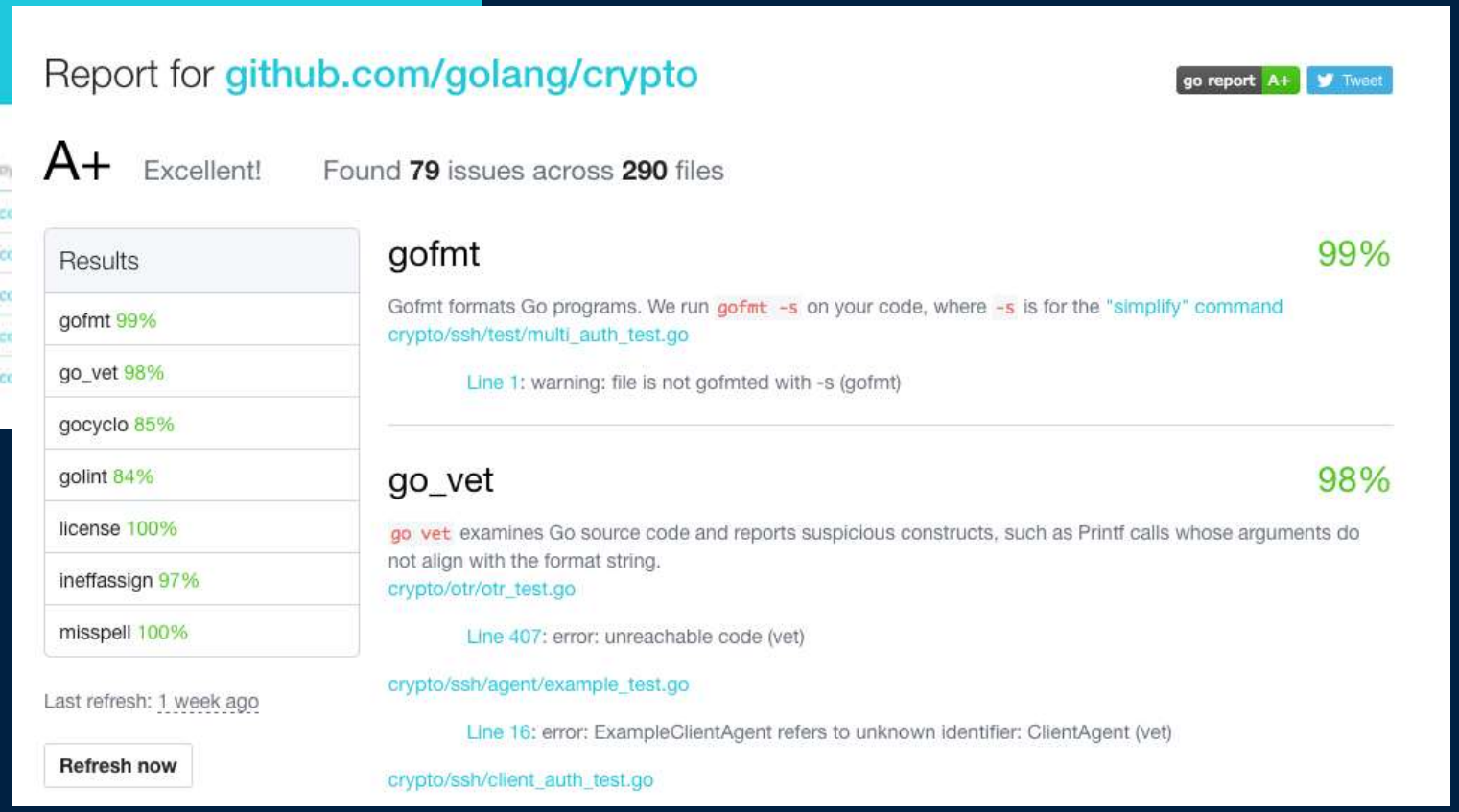
Filter. Eg: text, **/*.ts, !**/node_...

cert_util.go go/src/github.com/hashicorp/vault/builtin/logical/pki 2
- Blacklisted import crypto/sha1: weak cryptographic primitive,MEDIUM,HIGH (gosec) (10, 2)
- Use of weak cryptographic primitive,MEDIUM,HIGH (gosec) (1377, 2)

master+ | Python 2.7.14 64-bit | 0 2 | Ln 10, Col 18 (13 selected) | Tab Size: 4 | UTF-8 | LF | Go | 1

Gosec: Go Report Card





Gosec in Action

```go
package main

import (
    "crypto/rand"
    "crypto/rsa"
    "fmt"
)

func main() {
    reader := rand.Reader
    key, _ := rsa.GenerateKey(reader, 512)
    publicKey := key.PublicKey
    fmt.Println("Public key: ", publicKey)
}
```

Gosec in Action

```
bash-3.2$
```

Gosec: TLS rules

```
browne-a01:go browne$ bin/tlsconfig
browne-a01:go browne$
```

```
browne-a01:ericwb browne$ go generate ./...
browne-a01:ericwb browne$
```

```go
package rules

import (
    "go/ast"

    "github.com/securego/gosec"
)

// NewModernTLSCheck creates a check for Modern TLS ciphers
// DO NOT EDIT - generated by tlsconfig tool
func NewModernTLSCheck(id string, conf gosec.Config) (gosec.Rule, []ast.Node) {
    return &insecureConfigTLS{
        MetaData:     gosec.MetaData{ID: id},
        requiredType: "crypto/tls.Config",
        MinVersion:   0x0303,
        MaxVersion:   0x0303,
        goodCiphers: []string{
            "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305",
            "TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305",
            "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384",
            "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
        },
    }, []ast.Node{(*ast.CompositeLit)(nil)}
}

// NewIntermediateTLSCheck creates a check for Intermediate TLS ciphers
// DO NOT EDIT - generated by tlsconfig tool
func NewIntermediateTLSCheck(id string, conf gosec.Config) (gosec.Rule, []ast.Node) {
    return &insecureConfigTLS{
        MetaData:     gosec.MetaData{ID: id},
        requiredType: "crypto/tls.Config",
        MinVersion:   0x0301,
        MaxVersion:   0x0303,
        goodCiphers: []string{
            "TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305",
            "TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305",
            "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
            "TLS_DHE_RSA_WITH_AES_128_GCM_SHA256",
            "TLS_DHE_RSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
            "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",
            "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384",
```

Gosec: Contributing

- https://github.com/securego/gosec
- Ideas:
  - Documentation could use some work
  - More integrations with IDEs

What is <Yet-To-Be-Named>?

- GitHub App
- Uses the GitHub Checks API
- Currently utilizes these linters:
  - Bandit
  - Gosec
- Automatically scans Pull Requests

<Yet-To-Be-Named> in Action

```
bash-3.2$
```

<Yet-To-Be-Named> in Action

Changes from all commits ▾ Jump to... ▾ +3 −4 Diff settings ▾ Review changes ▾

```
12 - dsa.generate_private_key(key_size=2048,
13                            backend=backends.default_backend())
```

```
12 + dsa.generate_private_key(key_size=512,
13                            backend=backends.default_backend())
```

Check warning on line 13 in examples/weak_cryptographic_key_sizes.py

LocalForewarnDev / security-linter

B505:weak_cryptographic_key

DSA key sizes below 1024 bits are considered breakable.

```
14   ec.generate_private_key(curve=ec.SECP384R1,
15                           backend=backends.default_backend())
16   rsa.generate_private_key(public_exponent=65537,
17                            key_size=2048,
18                            backend=backends.default_backend())
19   pycrypto_dsa.generate(bits=2048)
20 - pycrypto_rsa.generate(bits=2048)
```

```
14   ec.generate_private_key(curve=ec.SECP384R1,
15                           backend=backends.default_backend())
16   rsa.generate_private_key(public_exponent=65537,
17                            key_size=2048,
18                            backend=backends.default_backend())
19   pycrypto_dsa.generate(bits=2048)
20 + pycrypto_rsa.generate(bits=512)
```

Check warning on line 20 in examples/weak_cryptographic_key_sizes.py

LocalForewarnDev / security-linter

B505:weak_cryptographic_key

RSA key sizes below 1024 bits are considered breakable.

```
21 - pycryptodomex_dsa.generate(bits=2048)
```

```
21 + pycryptodomex_rsa.generate(bits=512)
```

Check warning on line 21 in examples/weak_cryptographic_key_sizes.py

LocalForewarnDev / security-linter

B505:weak_cryptographic_key

RSA key sizes below 1024 bits are considered breakable.

```
22 - pycryptodomex_rsa.generate(bits=2048)
23
24   # Also correct: without keyword args
25   dsa.generate_private_key(4096,
```

```
22
23   # Also correct: without keyword args
24   dsa.generate_private_key(4096,
```

vmware

<Yet-To-Be-Named> in Action

Changes from all commits ▾ Jump to... ▾ +25 −0 Diff settings ▾ Review changes ▾

MVrachevApp / security-linter

B105:hardcoded_password_string

Possible hardcoded password: 'ajklawejrkl42348swfgkg'

```
16 +         print("Nice password!")
17 +
18 + def doLogin(password="blerg"):
19 +     pass
20 +
```

Check warning on line 20 in different_mistakes.py

MVrachevApp / security-linter

B107:hardcoded_password_default

Possible hardcoded password: 'blerg'

```
21 + def NoMatch3(a, b):
22 +     pass
23 +
24 + os.chmod('/etc/passwd', 0227)
```

Check failure on line 24 in different_mistakes.py

MVrachevApp / security-linter

B103:set_bad_file_permissions

Chmod setting a permissive mask 0227 on file (/etc/passwd).

```
25 + os.chmod('~/.bashrc', 511)
```

Check failure on line 25 in different_mistakes.py

MVrachevApp / security-linter

B103:set_bad_file_permissions

Chmod setting a permissive mask 0777 on file (~/.bashrc).

Thank You!

Please be sure to rate my talk on the All Things Open App

Email: browne@vmware.com
IRC: browne
GitHub github.com/ericwb