

Your Suite of Open Source Security Linters

SCaLE 17x

Eric Brown
Open Source Engineering Manager

vmware®

March 10th, 2019

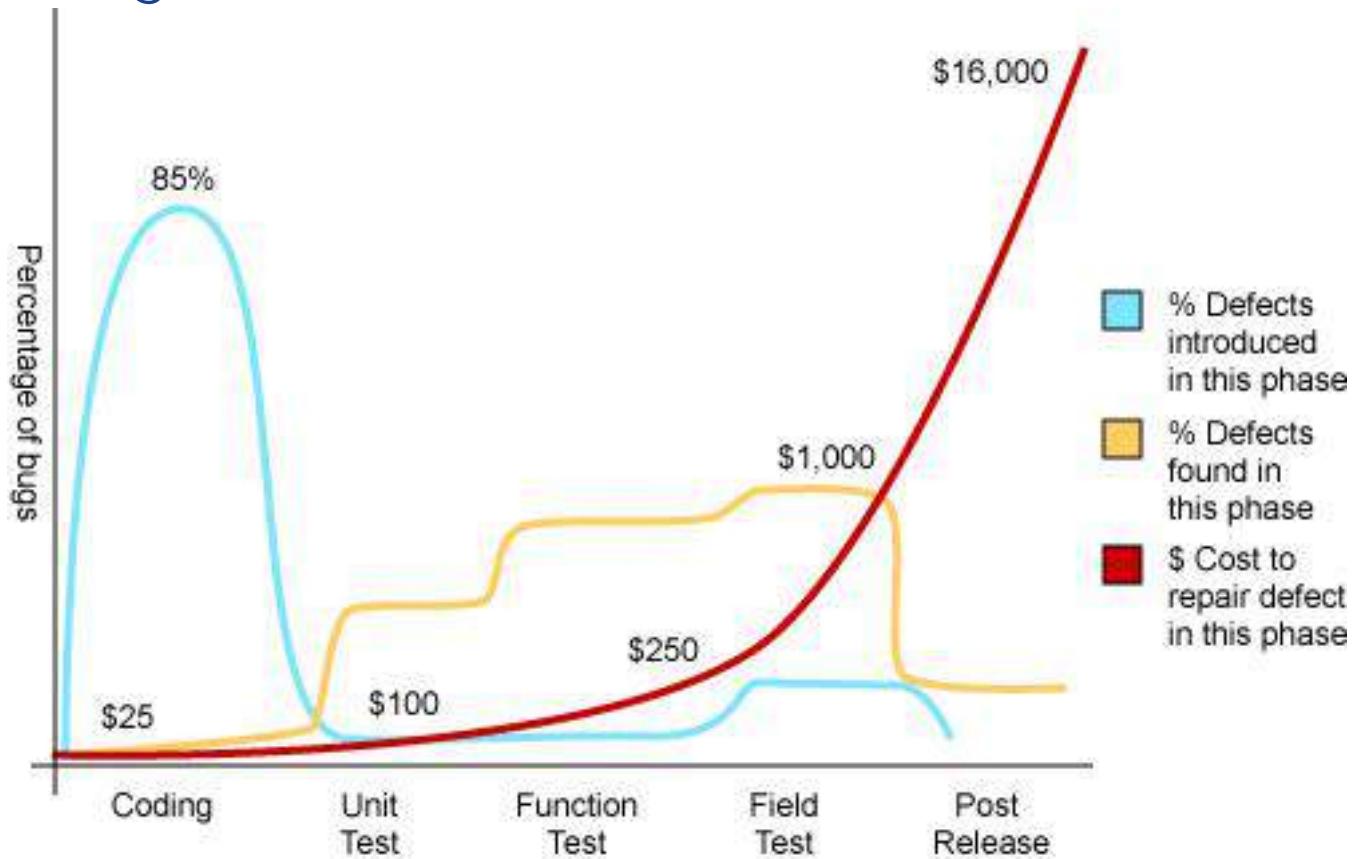
Agenda

- Why is this important?
- What is static code analysis?
- Tools available
 - Python
 - Golang
 - JavaScript
 - Others
 - GitHub

Moar Security

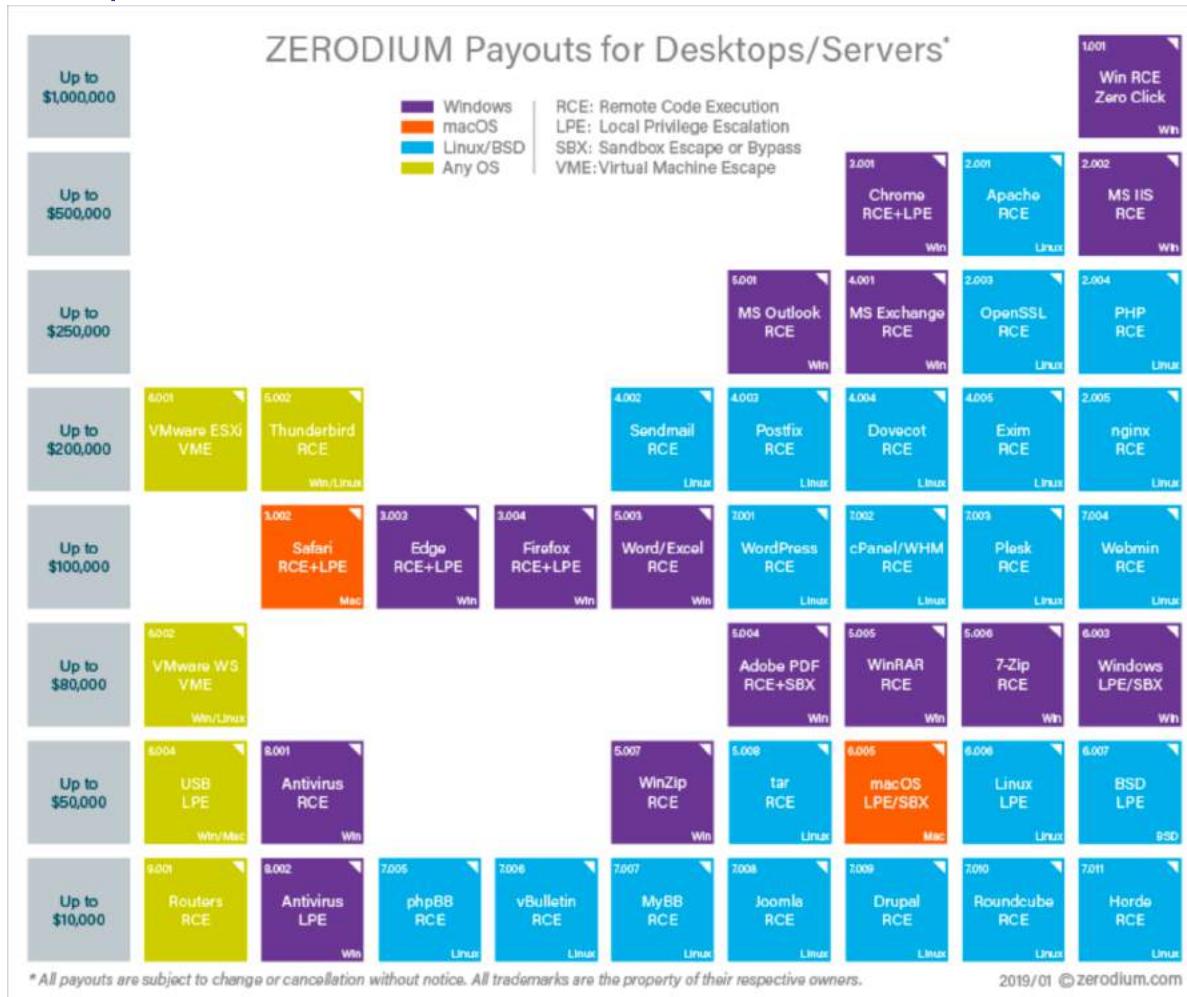


Cost to Fix Bugs

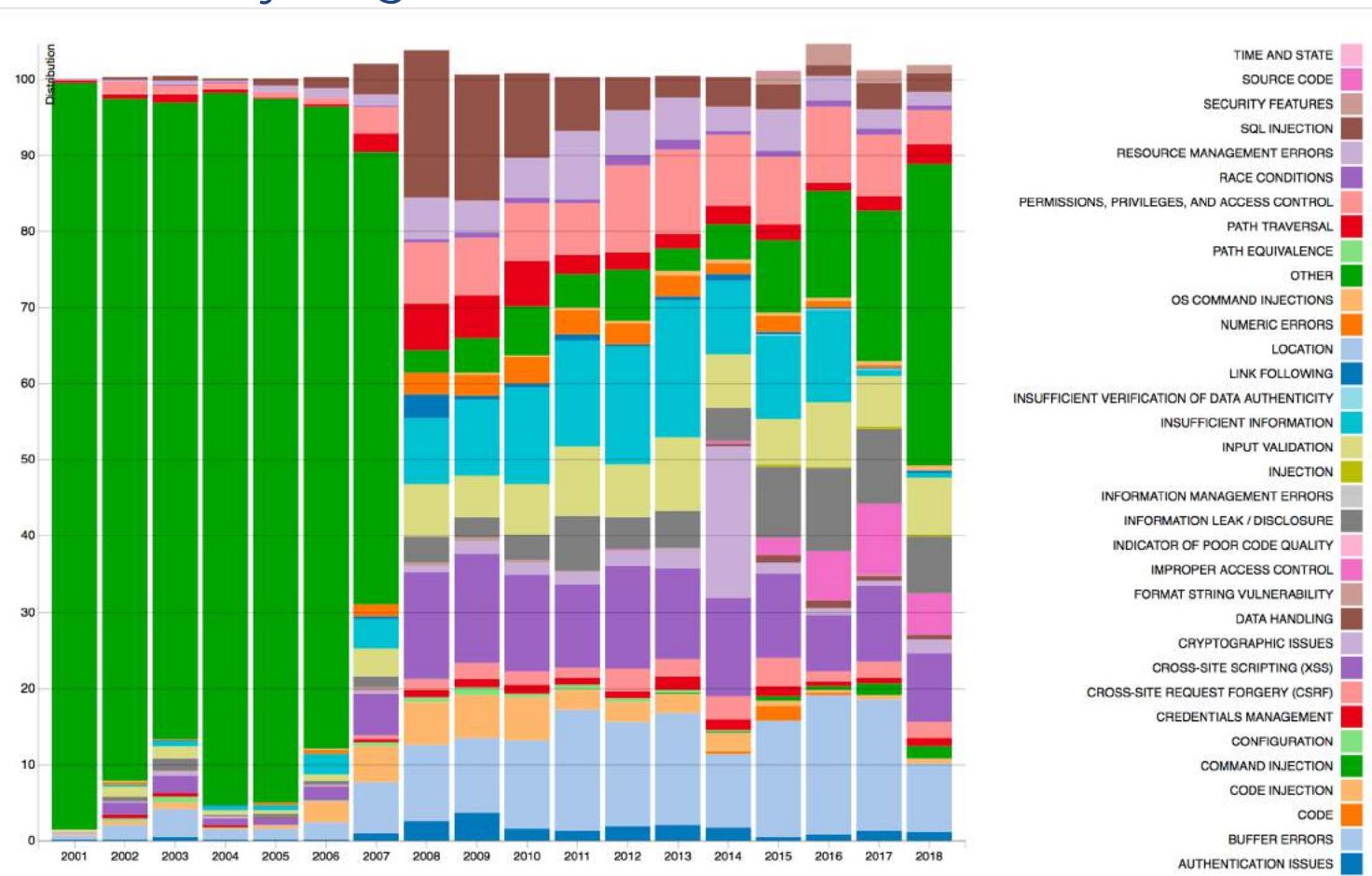


Source: *Applied Software Measurement*, Capers Jones, 1996

Business Impact \$\$\$



Types of Security Bugs



What's the Struggle?

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

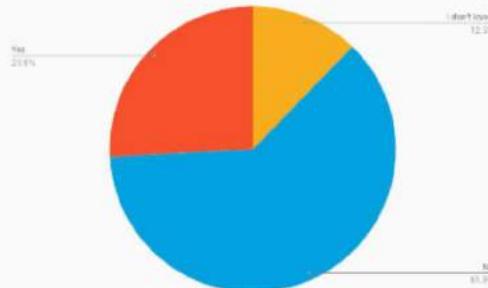
By [Catalin Cimpanu](#)

 May 10, 2018

 10:00 AM

 3

Q6. Has your organization ever ignored a critical security problem because it didn't have time or resources to rectify it?



How to Catch Security Bugs Earlier



Education

How to Catch Security Bugs Earlier



Education

A screenshot of a Pluralsight course page. The course title is "Security for Hackers and Developers" by Dr. Jand DeHott. The page includes a "Get Started" button, a brief description of the course, and a "RELATED TOPICS" section with links to Secure Coding, Security Development Lifecycle, Security Testing, Code Auditing, Fuzzing, and Exploit Development. Below the course title, there is an "ASSESSMENT DETAILS" section showing a graph of a user's performance. The graph has three colored areas: a light blue area at the top labeled "Expert 201-300", a green area in the middle labeled "Proficient 101-200", and a yellow area at the bottom labeled "Novice 0-100". A blue bar at the bottom represents the "4th percentile". The user's rating is shown as a red bar with a value of "0" and a pink diamond icon. The text "Security for Hackers and Developers : Expert" is displayed above the graph.

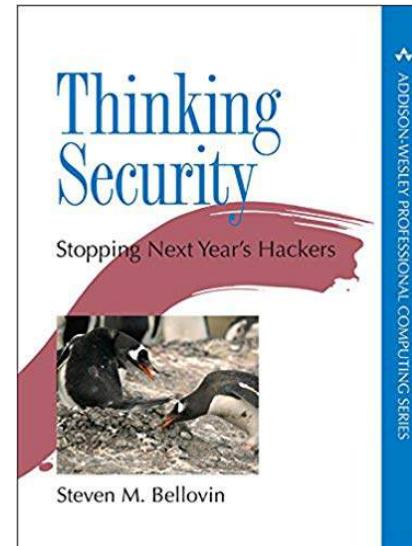
vmware®

How to Catch Security Bugs Earlier



Education

The screenshot shows the Pluralsight course page for 'Security for Hackers and Developers' by Dr. Jand DeHott. The course is marked as 'PATH' and 'Expert'. The description states: 'Security for Hackers and Developers lays the foundation for anyone interesting in creating secure software and systems, or anyone interesting in hacking computer systems.' A 'Get Started' button is visible. On the right, there's an 'ASSESSMENT DETAILS' section showing a graph of a user's rating (0) at the 41st percentile, with categories for Expert (201-300), Proficient (101-200), and Novice (0-100).

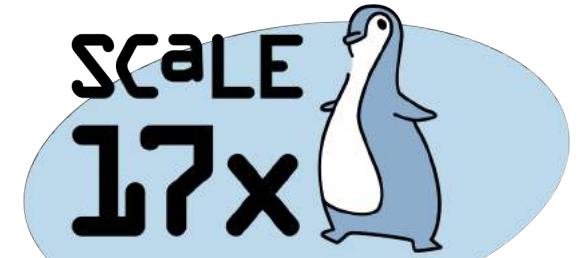
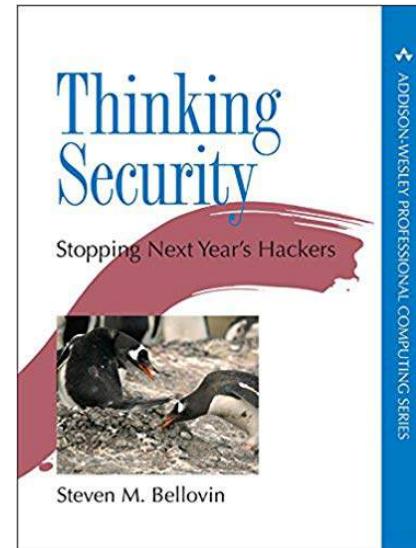


How to Catch Security Bugs Earlier



Education

The screenshot shows the Pluralsight course interface for 'Security for Hackers and Developers'. The course is authored by Dr. Jand DeHott. The description states: 'Security for Hackers and Developers lays the foundation for anyone interesting in creating secure software and systems, or anyone interesting in hacking computer systems.' Below the description is a 'Get Started' button. To the right, there is an 'ASSESSMENT DETAILS' section showing a graph of a user's rating (0) on a scale from 0 to 300, with the user being categorized as 'Expert'. The graph also shows the distribution of other users across 'Expert', 'Proficient', and 'Novice' levels.



How to Catch Security Bugs Earlier



Tooling

How to Catch Security Bugs Earlier



○ Unit testing

Tooling



Travis CI

How to Catch Security Bugs Earlier



- Unit testing
 - Dependency checking

Tooling



Travis CI

Code Issues Pull requests Projects Wiki Insights

Pulse Contributors Traffic Commits Code frequency Dependency graph Network Forks

Dependency graph

Dependencies Dependents

⚠ We found a potential security vulnerability in one of your dependencies. Dismiss

The `actionview` dependency defined in `Gemfile.lock` has a known moderate severity security vulnerability in version range `>=4.0.0, <=4.2.7` and should be updated.

Only users who have been granted access to vulnerability alerts for this repository can see this message. [Learn more about vulnerability alerts](#)

These dependencies have been defined in `VulnerabilityTestRepoRubyGems`'s manifest files, such as `Gemfile` and `Gemfile.lock`

Dependencies defined in `Gemfile` 1

How to Catch Security Bugs Earlier



- Unit testing
 - Dependency checking
 - Static Code Analysis

Tooling



Travis CI

Code Issues Pull requests Projects Wiki Insights

Pulse

Contributors

Traffic

Commits

Code frequency

Dependency graph

Network

Forks

Dependency graph

Dependencies Dependents

⚠ We found a potential security vulnerability in one of your dependencies. Dismiss

The `actionview` dependency defined in `Gemfile.lock` has a known **moderate severity** security vulnerability in version range `>=4.0.0, <=4.2.7` and should be updated.

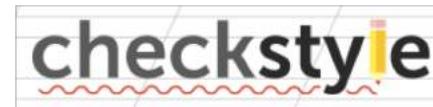
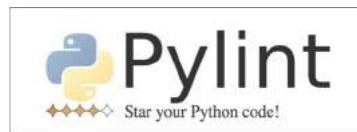
Only users who have been granted access to vulnerability alerts for this repository can see this message. [Learn more about vulnerability alerts](#)

These dependencies have been defined in `VulnerabilityTestRepoRubyGems`'s manifest files, such as `Gemfile` and `Gemfile.lock`

Dependencies defined in `Gemfile` 1

What is a Linter?

- Also known as Static Code Analysis tools
- Scans source code statically without running or building the code
- Linters focus on finding common mistakes
- Typically make use of a compiler's AST (abstract syntax tree)
- Popular examples:



When to Run Linters?

- When coding
 - Plugin 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 😞

- False positives
- Supplements, not replaces, human code review
- Extra time
 - Time to interpret the results
 - Time spent by CI to scan code

Bandit

Your Python Security Linter



What is Bandit?

- For Python
- 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

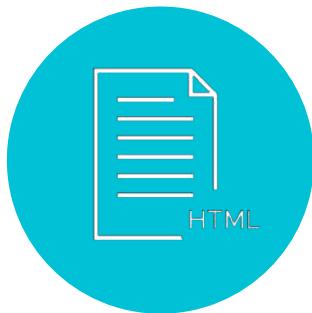
Bandit: Issues It Finds

- 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



CSV



HTML



JSON



Text



XML



YAML



Custom

Bandit: Other Features

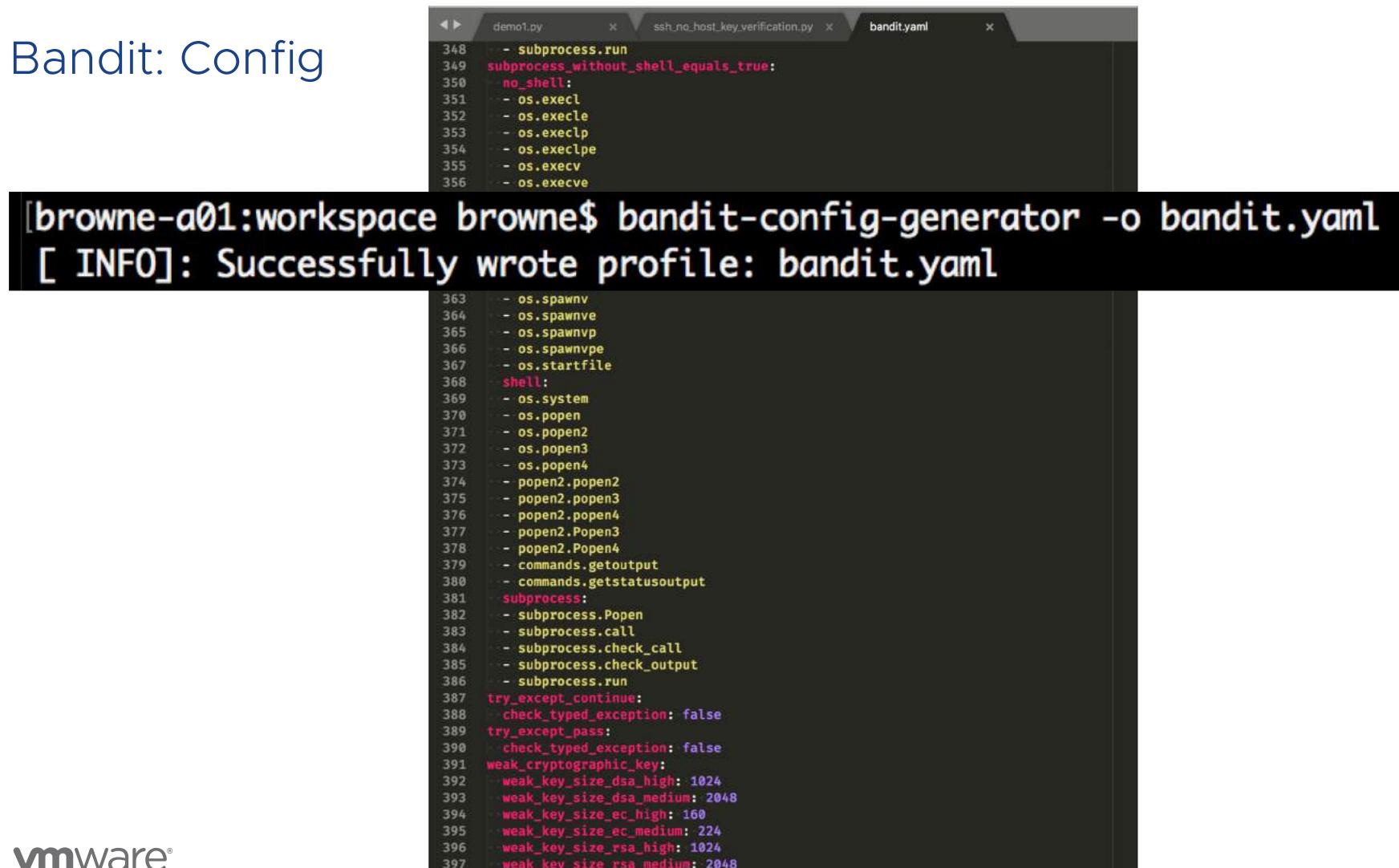
- Filtering on severity and confidence levels
- Allows creation of profiles to scan a subset of plugin tests
- Adjust lines of context shown in output
- Group results by file or vulnerability
- Output delta reports of previous scans
- Allows marking false positives using the “*# nosec*” comment

Bandit: Config

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

```
58  python setup.py check -- --skip-sdist
59
60  [testenv:bandit]
61  basepython = python3
62  skip_install = true
63  deps =
64      bandit
65  commands =
66      bandit --ini tox.ini -r historical
67
68  [bandit]
69  skips = B101
```

Bandit: Config



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

```
363: - os.spawnv
364: - os.spawnve
365: - os.spawnvp
366: - os.spawnvpe
367: - os.startfile
368: shell:
369: - os.system
370: - os.popen
371: - os.popen2
372: - os.popen3
373: - os.popen4
374: - popen2.popen2
375: - popen2.popen3
376: - popen2.popen4
377: - popen2.Popen3
378: - popen2.Popen4
379: - commands.getoutput
380: - commands.getstatusoutput
381: subprocess:
382: - subprocess.Popen
383: - subprocess.call
384: - subprocess.check_call
385: - subprocess.check_output
386: - subprocess.run
387: try_except_continue:
388: - check_typed_exception: false
389: try_except_pass:
390: - check_typed_exception: false
391: weak_cryptographic_key:
392: - weak_key_size_dsa_high: 1024
393: - weak_key_size_dsa_medium: 2048
394: - weak_key_size_ec_high: 160
395: - weak_key_size_ec_medium: 224
396: - weak_key_size_rsa_high: 1024
397: - weak_key_size_rsa_medium: 2048
```

Bandit: Dependents

Dependency graph

Dependencies Dependents

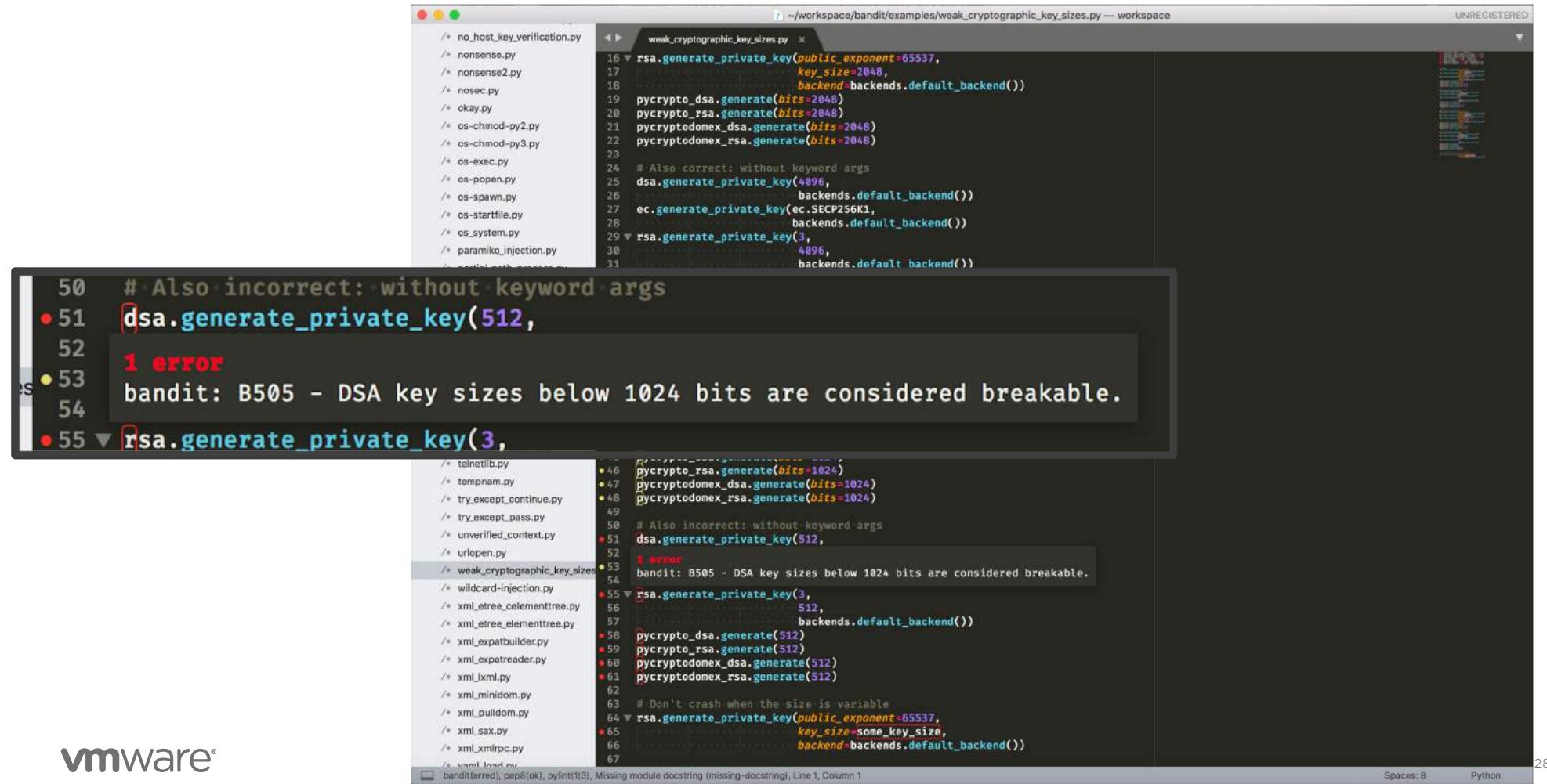
Repositories that depend on **bandit**

 1,259 Repositories  108 Packages 

Bandit: Integrations

- [SublimeLinter-bandit](#) – Sublime Text linter plugin
- [flake8-bandit](#) – Flake8 plugin
- [pyreportcard](#) – Report card of Python projects quality
- [bandit-plugin](#) – Hudson/Jenkins plugin
- [vscode-python](#) – Plugin for  * Added Feb 26th!

Bandit: Sublime Text Linter



The screenshot shows a Sublime Text window with a dark theme. The left sidebar lists several Python files: 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, and weak_cryptographic_key_sizes.py. The main editor area displays the code for weak_cryptographic_key_sizes.py. A red box highlights line 51: `dsa.generate_private_key(512,`. A red box also highlights line 55: `rsa.generate_private_key(3,`. A callout box points to line 53, which shows a red error message: `bandit: B505 - DSA key sizes below 1024 bits are considered breakable.`. The status bar at the bottom shows: bandit(error), pep8(ok), pylint(t13), Missing module docstring (missing-docstring), Line 1, Column 1. The bottom right corner shows: Spaces: 8, Python, and page 28.

```
weak_cryptographic_key_sizes.py
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 pycryptodome_dsa.generate(bits=2048)
22 pycryptodome_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())
50 # Also incorrect: without keyword args
51 dsa.generate_private_key(512,
52
53 1 error
54 bandit: B505 - DSA key sizes below 1024 bits are considered breakable.
55 ▼ rsa.generate_private_key(3,
56     512,
57     backends.default_backend())
58 pycrypto_dsa.generate(512)
59 pycrypto_rsa.generate(512)
60 pycryptodome_dsa.generate(512)
61 pycryptodome_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())

```

Package Control

Search

cmd+shift+p

BROWSE

SublimeLinter-bandit

by SublimeLinter **ST3**

Only works with Sublime Text 3

SublimeLinter plugin for Python, using bandit

LABELS [linting](#), [SublimeLinter](#), [python](#), [security](#)

Details

VERSION 1.1.1
HOMEPAGE github.com
ISSUES github.com
MODIFIED 12 months ago
LAST SEEN 1 hour ago
FIRST SEEN 2 years ago

Installs

TOTAL 15K WIN 10K OS X 3K LINUX 2K

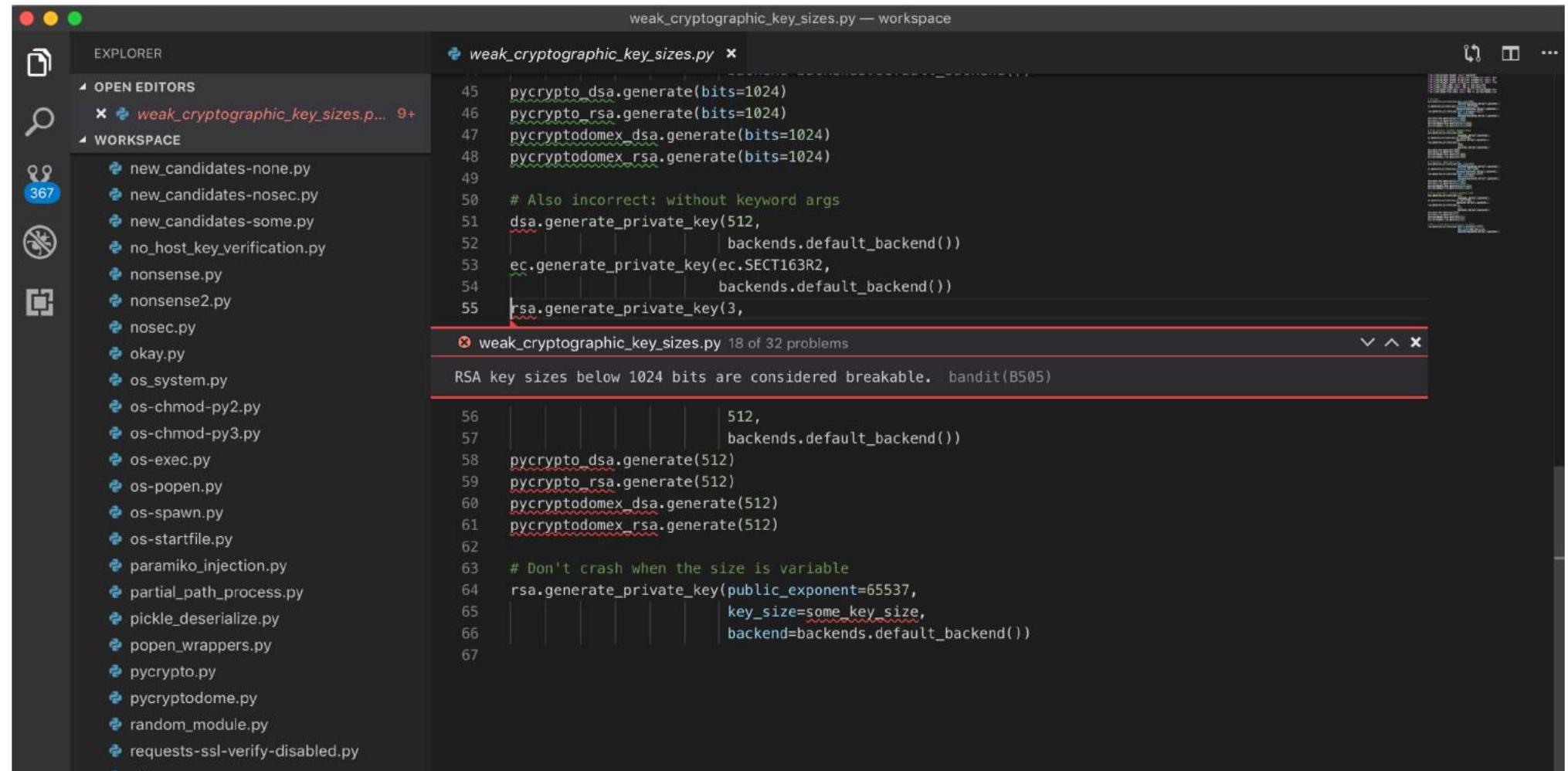


Readme

SOURCE raw.githubusercontent.com

vmware®

Bandit: VSCode



The screenshot shows the VSCode interface with a Bandit analysis results overlay. The left sidebar shows the Explorer with various Python files listed under 'OPEN EDITORS' and 'WORKSPACE'. The main editor area shows a Python script named 'weak_cryptographic_key_sizes.py'. A red arrow points to the line 'rsa.generate_private_key(3,' in the code, which is flagged as a problem. The status bar at the bottom indicates '18 of 32 problems'.

```
weak_cryptographic_key_sizes.py — workspace
weak_cryptographic_key_sizes.py x
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                         backends.default_backend())
53 ec.generate_private_key(ec.SECT163R2,
54                         backends.default_backend())
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
```

weak_cryptographic_key_sizes.py 18 of 32 problems

RSA key sizes below 1024 bits are considered breakable. bandit(B505)

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
```

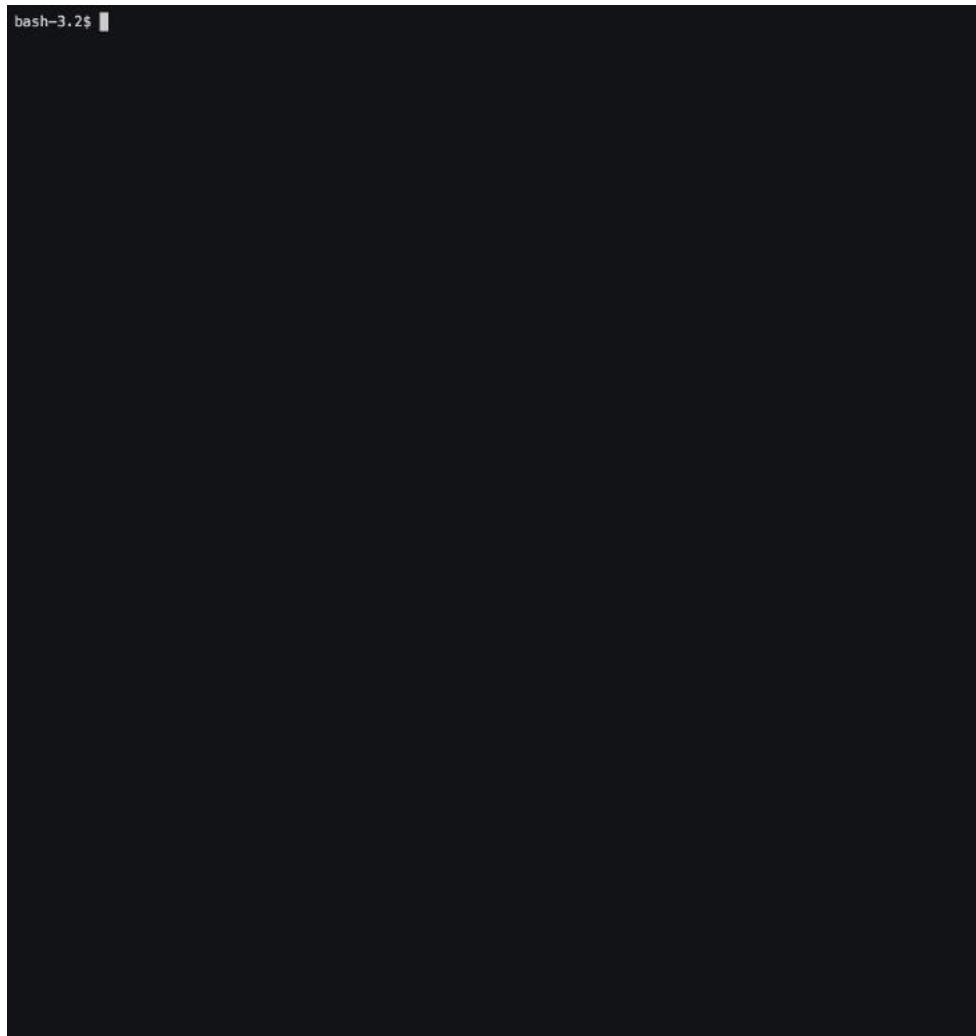
Bandit in Action

```
import paramiko
import re

class Dev:
    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("(\\d-\\d*) No.*" % name.lower(),
                                     line.lower())
                id = matchid.group(0)
                break
        s.close()
        return id
```

Bandit in Action



Bandit in Action



Contributing to Bandit

- <https://github.com/PyCQA/bandit>
- Core maintainers:
 - Myself – ericwb
 - Ian StapleTon Cordasco – sigmavirus24
 - Gage Hugo – ghugo
 - Luke Hinds – lukehinds
- Ideas:
 - Documentation could be improved
 - Integrations with more IDEs
 - Fix suggestions
 - Quick Fix - VSCode
 - Suggested changes - GitHub

Gosec

Your Golang Security Linter



vmware®

What is Gosec?

- For Golang
- Project started mid 2016
- Started by one of the creators of Bandit
- Runs on Linux, macOS, and Windows
- Low resource requirements
- Runs quickly
- Pluggable

Gosec: Issues It Finds

- 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



CSV



HTML



JSON



Text



XML

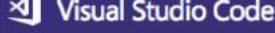


YAML

Gosec: Other Features

- Filtering on severity 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>
- Gotchas:
 - Only scans projects in your \$GOPATH

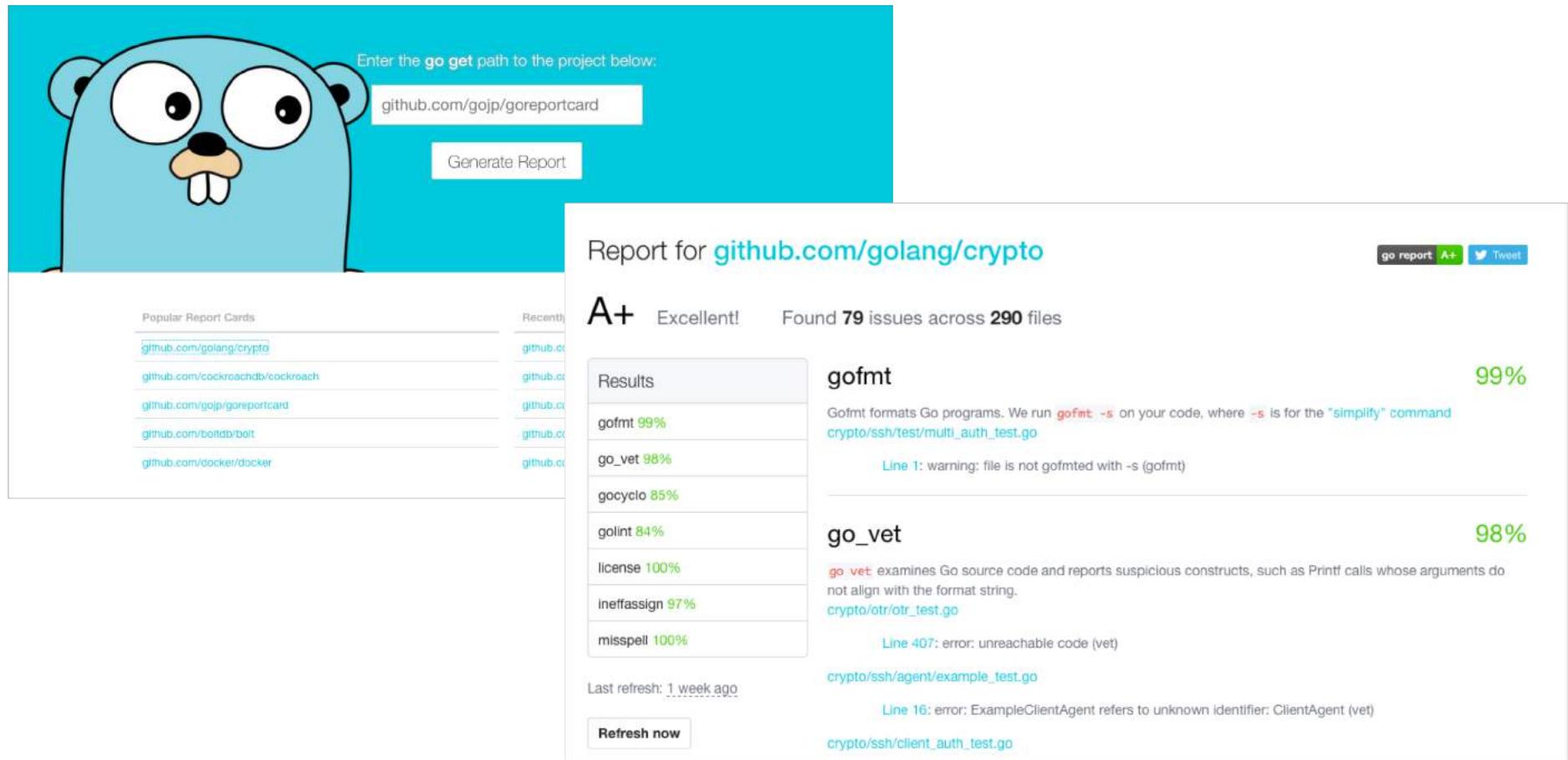
Gosec: Integrations

- [SublimeLinter-contrib-gometallinter](#) – Sublime Text plugin
- [SublimeLinter-golangcilint](#) – Sublime Text plugin
- [gometallinter](#) – collection of linters
- [golangcilint](#) – collection of linters
-  – via gometallinter or golangci-lint plugin
- [go-plus](#) – Atom via golangci-lint plugin

Wishlist:

- Go Report Card

Gosec: Go Report Card

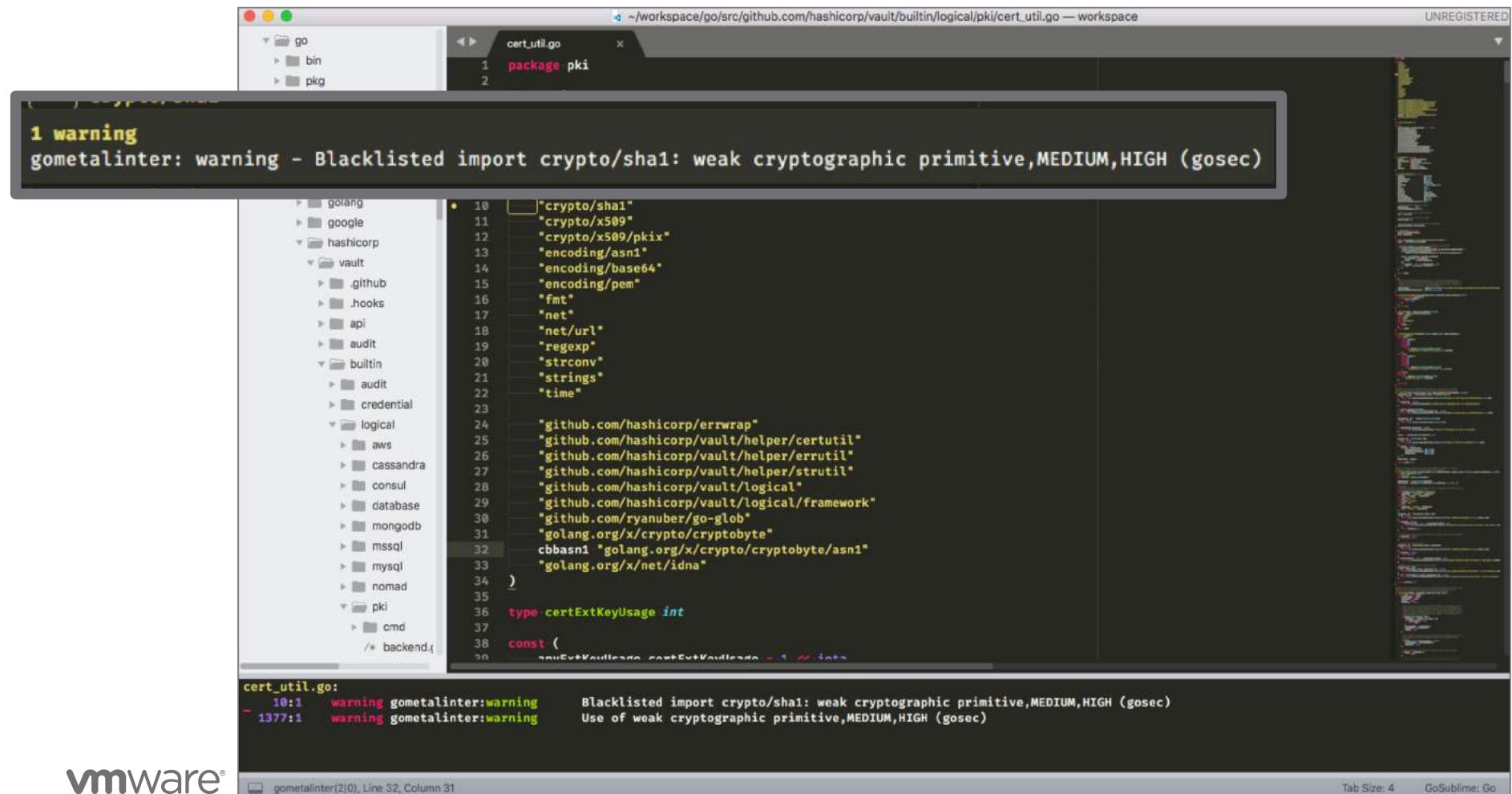


The screenshot shows the Go Report Card interface. On the left, there's a large blue cartoon gopher character. The main form area has a teal header with a text input field containing "github.com/gojp/goreportcard" and a "Generate Report" button. Below this, the report card for the github.com/golang/crypto repository is displayed. The card shows a grade of **A+** and the message "Excellent! Found 79 issues across 290 files". The report card is divided into sections for different tools:

- gofmt**: 99% (highlighted in green). Description: "Gofmt formats Go programs. We run `gofmt -s` on your code, where `-s` is for the "simplify" command". Example: `crypto/ssh/test/multi_auth_test.go`. Error: `Line 1: warning: file is not gofmted with -s (gofmt)`.
- go_vet**: 98% (highlighted in green). Description: "go_vet examines Go source code and reports suspicious constructs, such as Printf calls whose arguments do not align with the format string". Example: `crypto/otr/otr_test.go`. Error: `Line 407: error: unreachable code (vet)`.
- license**: 100% (highlighted in green).
- ineffassign**: 97% (highlighted in green).
- misspell**: 100% (highlighted in green).

At the bottom, there's a "Refresh now" button, a note about the last refresh (1 week ago), and social sharing buttons for "go report" and "Tweet".

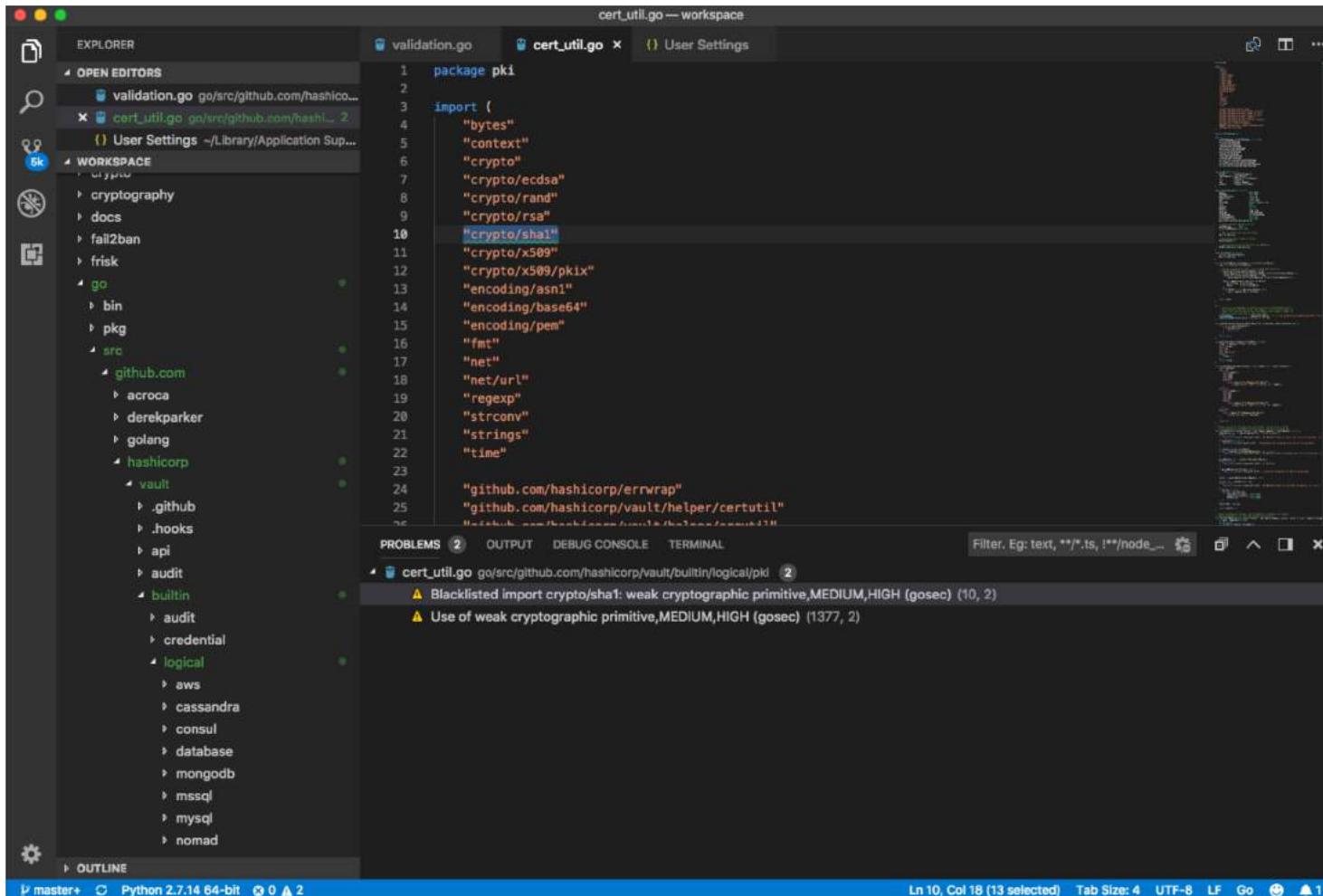
Gosec: Sublime Text Linter



The screenshot shows a Sublime Text window with the following details:

- File Path:** ~/workspace/go/src/github.com/hashicorp/vault/builtin/logical/pki/cert_util.go
- Code Editor:** The main editor shows the Go code for `cert_util.go`. The code includes imports for `crypto/sha1` and `crypto/x509`, and defines a `certExtKeyUsage` type and a `const` block.
- Linter Output:** A modal window displays a single warning from `gometalinter`: "Blacklisted import crypto/sha1: weak cryptographic primitive, MEDIUM, HIGH (gosec)".
- File Explorer:** The sidebar shows the project structure with packages like `go`, `bin`, and `pkg`, and sub-packages like `hashicorp/vault` and `logical/pki`.
- Right Panel:** A vertical panel on the right shows a list of other open files in the workspace.
- Bottom Status Bar:** The status bar shows "gometalinter(2/0), Line 32, Column 31", "Tab Size: 4", and "GoSublime: Go".

Gosec: Visual Studio Code



The screenshot shows the Visual Studio Code interface with the 'cert_util.go' file open. The code is as follows:

```
1 package pki
2
3 import (
4     "bytes"
5     "context"
6     "crypto"
7     "crypto/ecdsa"
8     "crypto/rand"
9     "crypto/rsa"
10    "crypto/sha1"
11    "crypto/x509"
12    "crypto/x509/pkix"
13    "encoding/asn1"
14    "encoding/base64"
15    "encoding/pem"
16    "fmt"
17    "net"
18    "net/url"
19    "regexp"
20    "strconv"
21    "strings"
22    "time"
23
24    "github.com/hashicorp/errwrap"
25    "github.com/hashicorp/vault/helper/certutil"
26)
```

The 'PROBLEMS' tab shows two warnings from Gosec:

- Blacklisted import crypto/sha1: weak cryptographic primitive, MEDIUM, HIGH (gosec) (10, 2)
- Use of weak cryptographic primitive, MEDIUM, HIGH (gosec) (1377, 2)

The status bar at the bottom shows: master+ Python 2.7.14 64-bit 0 ▲ 2, Ln 10, Col 18 (13 selected) Tab Size: 4 UTF-8 LF Go, and a green progress bar.

Gosec in Action

```
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

```
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$ [REDACTED]
```

Gosec: TLS rules

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

```
1 package rules  
2  
3 import (  
4     "go/ast"  
5     "github.com/securego/gosec"  
6 )  
7  
8 // NewModernTLSCheck creates a check for Modern TLS ciphers  
9 // DO NOT EDIT -- generated by tlsconfig tool  
10 func NewModernTLSCheck(id string, conf gosec.Config) (gosec.Rule, []ast.Node) {  
11     return &insecureConfigTLS{  
12         MetaData: gosec.MetaData{ID: id},  
13         requiredType: "crypto/tls.Config",  
14         MinVersion: 0x0303,  
15         MaxVersion: 0x0303,  
16         goodCiphers: []string{  
17             "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",  
18             "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",  
19             "TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305",  
20             "TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305",  
21             "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",  
22             "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",  
23             "TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384",  
24             "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384",  
25             "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",  
26             "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",  
27             "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",  
28         },  
29         []ast.Node{(*ast.CompositeLit)(nil)}  
30     }  
31  
32 // NewIntermediateTLSCheck creates a check for Intermediate TLS ciphers  
33 // DO NOT EDIT -- generated by tlsconfig tool  
34 func NewIntermediateTLSCheck(id string, conf gosec.Config) (gosec.Rule, []ast.Node) {  
35     return &insecureConfigTLS{  
36         MetaData: gosec.MetaData{ID: id},  
37         requiredType: "crypto/tls.Config",  
38         MinVersion: 0x0301,  
39         MaxVersion: 0x0303,  
40         goodCiphers: []string{  
41             "TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305",  
42             "TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305",  
43             "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",  
44             "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",  
45             "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",  
46             "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",  
47             "TLS_DHE_RSA_WITH_AES_128_GCM_SHA256",  
48             "TLS_DHE_RSA_WITH_AES_256_GCM_SHA384",  
49             "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256",  
50             "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",  
51             "TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA",  
52             "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

eslint-plugin-security

Your JavaScript Security Linter



vmware®

What is ESLint-Security-Plugin?

- A security plugin for ESLint
- Project started mid 2015
- Runs on Linux, macOS, and Windows
- Ignore false positives with `// eslint-disable-line`
- Configure via `.eslintrc` file
- Supports auto-fixing

eslint-plugin-security: Formatters

- So many!
 - Checkstyle
 - Codeframe
 - Compact
 - HTML
 - Jslint-xml
 - JSON
 - Junit
 - Stylish (default)
 - Table
 - Tap
 - Unix
 - visualstudio

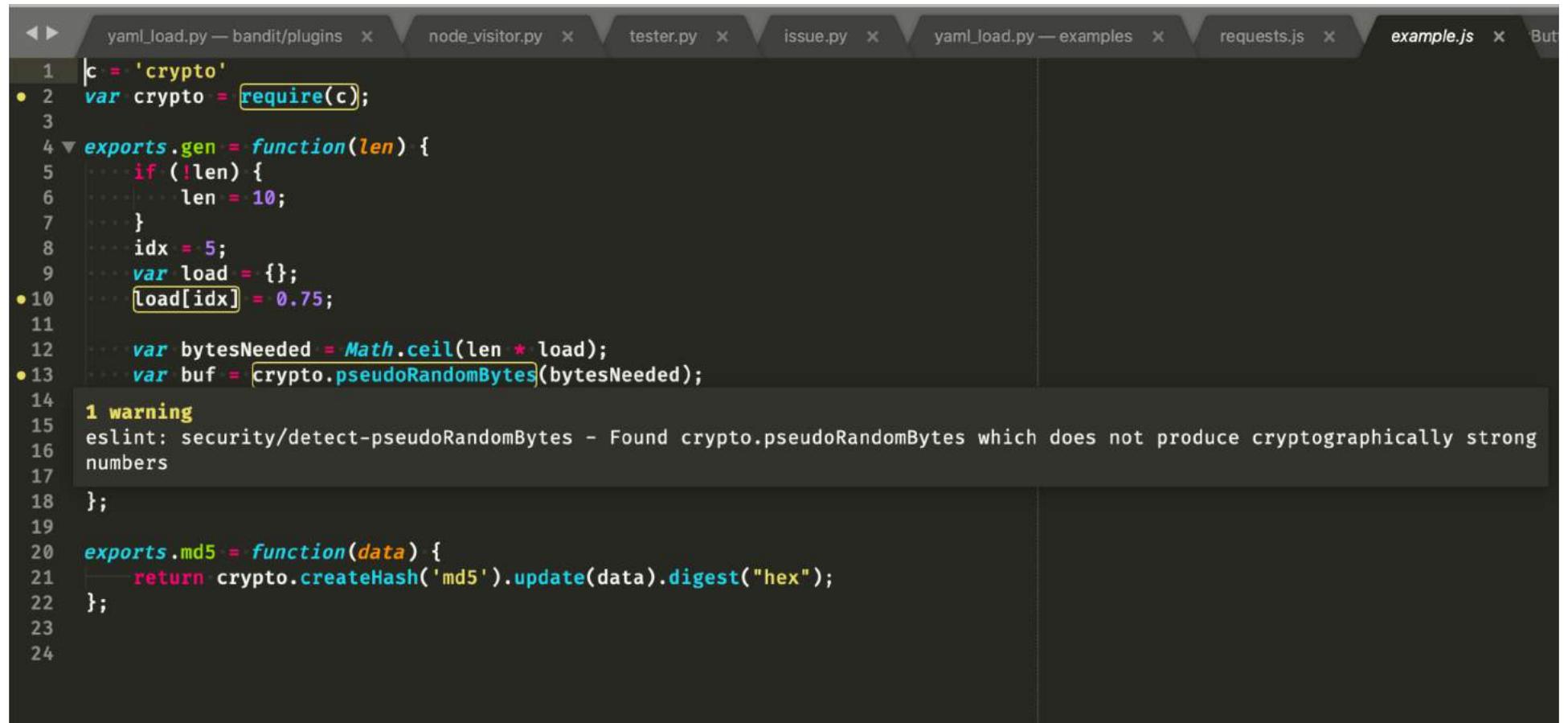
eslint-plugin-security: Issues It Finds

- Unsafe regex
- Buffer with `noAssert` flag
- Instances of `child_process` and `exec()`
- Disabled escape markup
- `eval()` with an expression
- CSRF middleware setup before method-override
- Variable in filename argument of `fs`
- Use of `RegExp(variable)`
- Use of `require(variable)`
- Object injection
- Possible timing attacks
- Use of `pseudoRandomBytes()`

eslint-plugin-security : Integrations

- [SublimeLinter-eslint](#) - Sublime Text plugin
- [vscode-eslint](#) –  Visual Studio Code
- [linter-eslint](#) – Atom plugin

eslint-plugin-security: Sublime Text Linter

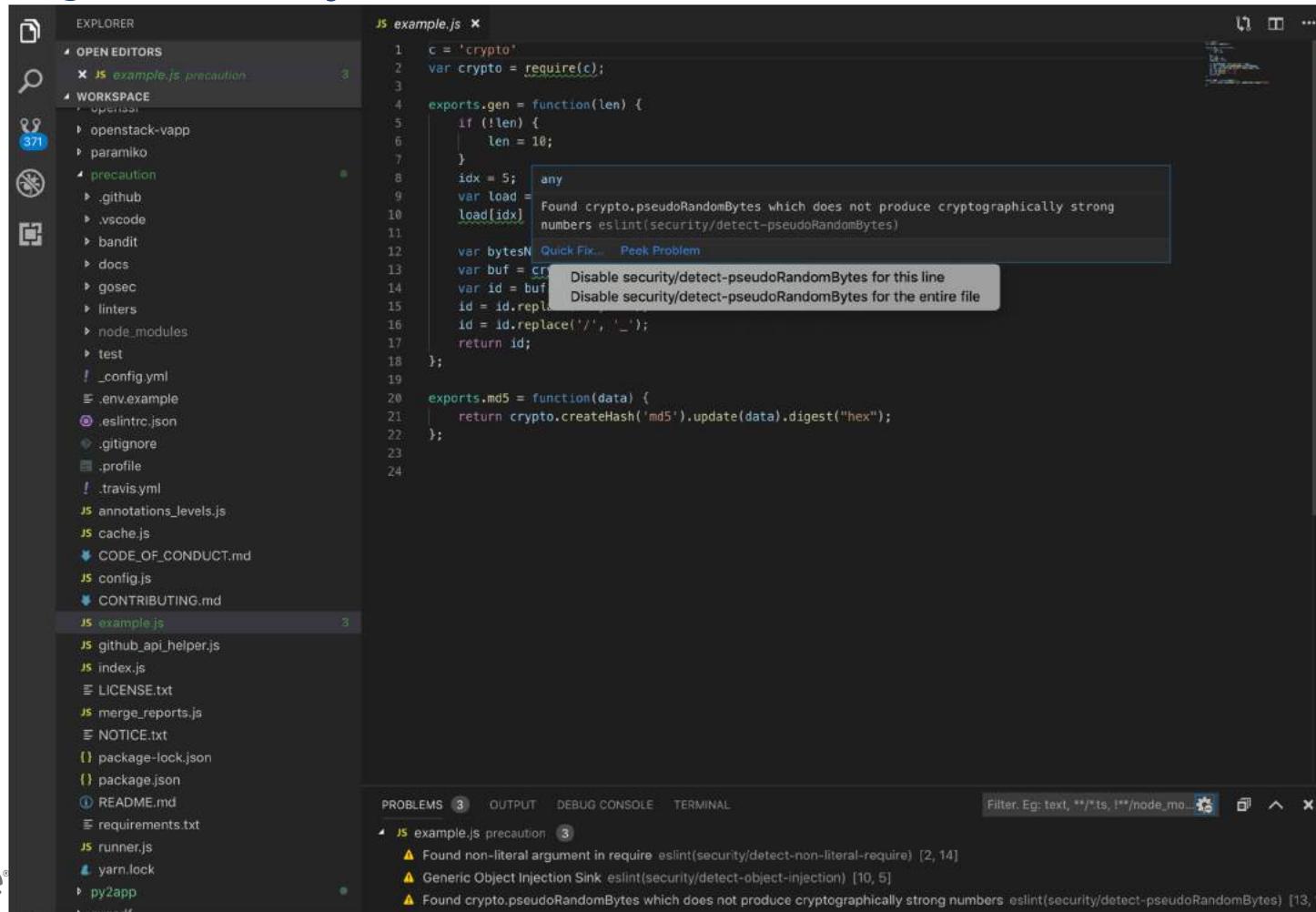


The screenshot shows a Sublime Text window with multiple tabs at the top: yaml_load.py — bandit/plugins, node_visitor.py, tester.py, issue.py, yaml_load.py — examples, requests.js, and example.js. The yaml_load.py — examples tab is active. The code editor displays a file with the following content:

```
1 |c = 'crypto';
2 |  var crypto = require(c);
3 |
4 ▼ exports.gen = function(len) {
5   if (!len) {
6     len = 10;
7   }
8   idx = 5;
9   var load = {};
10  load[idx] = 0.75;
11
12  var bytesNeeded = Math.ceil(len * load);
13  var buf = crypto.pseudoRandomBytes(bytesNeeded);
14
15  1 warning
16  eslint: security/detect-pseudoRandomBytes - Found crypto.pseudoRandomBytes which does not produce cryptographically strong
17  numbers
18 }
19
20 exports.md5 = function(data) {
21   return crypto.createHash('md5').update(data).digest("hex");
22 }
23
24
```

A yellow box highlights the line `var buf = crypto.pseudoRandomBytes(bytesNeeded);`. A tooltip or status bar at the bottom of the editor window displays the warning: `eslint: security/detect-pseudoRandomBytes - Found crypto.pseudoRandomBytes which does not produce cryptographically strong numbers`.

eslint-plugin-security: VSCode



The screenshot shows the VSCode interface with the eslint-plugin-security extension. The Explorer sidebar on the left lists various workspace files and folders, including `example.js` and `precaution`. The `example.js` file is open in the editor, showing a function that uses the `crypto` module. A tooltip is displayed over the `load[idx]` line, stating: "Found `crypto.pseudoRandomBytes` which does not produce cryptographically strong numbers `eslint(security/detect-pseudoRandomBytes)`". Below the editor, the Problems panel shows three ESLint errors:

- Found non-literal argument in `require` `eslint(security/detect-non-literal-require)` [2, 14]
- Generic Object Injection Sink `eslint(security/detect-object-injection)` [10, 5]
- Found `crypto.pseudoRandomBytes` which does not produce cryptographically strong numbers `eslint(security/detect-pseudoRandomBytes)` [13, 11]

eslint-plugin-security in Action

```
c = 'crypto'
var crypto = require(c);

exports.gen = function(len) {
  if (!len) {
    len = 10;
  }
  idx = 5;
  var load = {};
  load[idx] = 0.75;

  var bytesNeeded = Math.ceil(len * load);
  var buf = crypto.pseudoRandomBytes(bytesNeeded);
  var id = buf.toString('base64').substring(0, len);
  id = id.replace('+', '-');
  id = id.replace('/', '_');
  return id;
};

exports.md5 = function(data) {
  return crypto.createHash('md5').update(data).digest("hex");
};
```

eslint-plugin-security in Action

```
c = 'crypto'
var crypto = require(c);

exports.gen = function(len) {
  if (!len) {
    len = 10;
  }
  idx = 5;
  var load = [0];
  load[idx] = 0.75;

  var bytesNeeded = Math.ceil(len * load);
  var buf = crypto.pseudoRandomBytes(bytesNeeded);
  var id = buf.toString('base64').substring(0, len);
  id = id.replace(/\+/g, '-');
  id = id.replace(/\//g, '_');
  return id;
};

exports.md5 = function(data) {
  return crypto.createHash('md5').update(data).digest('hex');
};
```

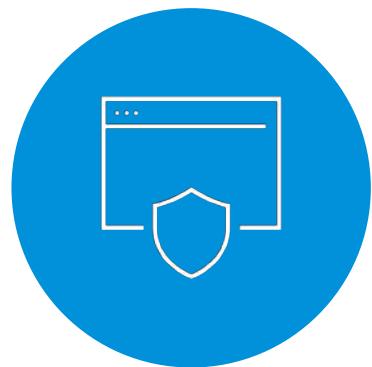
eslint-plugin-security in Action

```
browne-a02:precaution browne$ eslint --plugin security example.js  
/Users/browne/workspace/precaution/example.js  
  2:14  warning  Found non-literal argument in require  
                security/detect-non-literal-require  
 10:5   warning  Generic Object Injection Sink  
                security/detect-object-injection  
 13:15  warning  Found crypto.pseudoRandomBytes which does not produce cryptographically strong numbers  security/detect-pseudoRandomBytes  
  
✖ 3 problems (0 errors, 3 warnings)
```

eslint-plugin-security: Contributing

- <https://github.com/nodesecurity/eslint-plugin-security>
- Ideas:
 - Needs maintainers
 - More tests
 - More severity – currently only warnings

Others



vmware®



OWASP.org

https://www.owasp.org/index.php/Source_Code_Analysis_Tools

Open Source or Free Tools Of This Type

- [Bandit](#) - bandit is a comprehensive source vulnerability scanner for Python
- [Brakeman](#) - Brakeman is an open source vulnerability scanner specifically designed for Ruby on Rails applications
- [Codesake Dawn](#) - Codesake Dawn is an open source security source code analyzer designed for Sinatra, Padrino for Ruby on Rails applications. It also works on non-web applications written in Ruby
- [FindBugs](#) - (Legacy - NOT Maintained - Use SpotBugs (see below) instead) - Find bugs (including a few security flaws) in Java programs
- [FindSecBugs](#) - A security specific plugin for SpotBugs that significantly improves SpotBugs's ability to find security vulnerabilities in Java programs. Works with the old FindBugs too,
- [Flawfinder](#) Flawfinder - Scans C and C++
- [Google CodeSearchDiggity](#) - Uses Google Code Search to identifies vulnerabilities in open source code projects hosted by Google Code, MS CodePlex, SourceForge, Github, and more. The tool comes with over 130 default searches that identify SQL injection, cross-site scripting (XSS), insecure remote and local file includes, hard-coded passwords, and much more. *Essentially, Google CodeSearchDiggity provides a source code security analysis of nearly every single open source code project in existence – simultaneously.*
- [Graudit](#) - Scans multiple languages for various security flaws.
- [LGTM](#) - A free for open source static analysis service that automatically monitors commits to publicly accessible code in: Bitbucket Cloud, GitHub, or GitLab. Supports C/C++, C#, COBOL (in beta), Java, JavaScript/TypeScript, Python
- [PMD](#) - PMD scans Java source code and looks for potential code problems (this is a code quality tool that does not focus on security issues)
- [Progpiilot](#) - Progpiilot is a static analyzer tool for PHP that detects security vulnerabilities such as XSS and SQL Injection.
- [PreFast](#) (Microsoft) - PREfast is a static analysis tool that identifies defects in C/C++ programs. Last update 2006.
- [Puma Scan](#) - Puma Scan is a .NET C# open source static source code analyzer that runs as an IDE plugin for Visual Studio and via MSBuild in CI pipelines.
- [.NET Security Guard](#) - Roslyn analyzers that aim to help security audits on .NET applications. It will find SQL injections, LDAP injections, XXE, cryptography weakness, XSS and more.
- [RIPS](#) - RIPS is a static source code analyzer for vulnerabilities in PHP web applications. Please see notes on the sourceforge.net site.
- [phpcs-security-audit](#) - phpcs-security-audit is a set of PHP_CodeSniffer rules that finds flaws or weaknesses related to security in PHP and its popular CMS or frameworks. It currently has core PHP rules as well as Drupal 7 specific rules.
- [SonarQube](#) - Scans source code for more than 20 languages for Bugs, Vulnerabilities, and Code Smells. SonarQube IDE plugins for Eclipse, Visual Studio, and IntelliJ provided by SonarLint.
- [SpotBugs](#) - This is the active fork replacement for FindBugs, which is not maintained anymore.
- [VisualCodeGrepper \(VCG\)](#) - Scans C/C++, C#, VB, PHP, Java, and PL/SQL for security issues and for comments which may indicate defective code. The config files can be used to carry out additional checks for banned functions or functions which commonly cause security issues.

Precaution

One GitHub App to Run Them All



What is Precaution?

- Created late 2018
- GitHub App
- Uses the GitHub Checks API
- Automatically scans and annotates Pull Requests
- Support for:



Coming soon...

bandit/examples at master · ericwb/bandit

GitHub, Inc. (US) https://github.com/ericwb/bandit/tree/master/examples

Search

sql_statements-py36.py	Fix sql injection check for f-strings	3 months ago
sql_statements.py	Alter SQL injection plugin to consider .format strings	2 years ago
ssl-insecure-version.py	Remove the check for PROTOCOL_SSLv23	4 years ago
subprocess_shell.py	Add subprocess.run to B602	8 months ago
telnetlib.py	Introduce wildcards to blacklist_calls plugin	4 years ago
tempnam.py	add os.tempnam() / os.tmpnam() to blacklist	8 months ago
try_except_continue.py	Added try_except_continue plugin	3 years ago
try_except_pass.py	Adding test for Try, Except, Pass	4 years ago
unverified_context.py	Blacklist call of ssl._create_unverified_context	2 years ago
urlopen.py	Some spelling error need to be fixed	3 years ago
weak_cryptographic_key_sizes.py	Add Cryptodome to blacklist and weak ciphers/hash	2 years ago
wildcard-injection.py	Adding a test for partial paths in exec functions	4 years ago
xml_etree_clementtree.py	Update example files to work on Python 2 & 3	4 years ago
xml_etree_elementtree.py	Update example files to work on Python 2 & 3	4 years ago
xml_expatbuilder.py	Add XML vulnerability checking	4 years ago
xml_expatreader.py	Add XML vulnerability checking	4 years ago
xml_lxml.py	Add XML vulnerability checking	4 years ago
xml_minidom.py	Update example files to work on Python 2 & 3	4 years ago
xml_pulldom.py	Update example files to work on Python 2 & 3	4 years ago
xml_sax.py	Update example files to work on Python 2 & 3	4 years ago
xml_xmlrpc.py	Update example files to work on Python 2 & 3	4 years ago
yaml_load.py	Fast fix for PyCQA#286	9 months ago

© 2019 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)

[Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

Precaution: Contributing

- <https://github.com/vmware/precaution>
- Ideas:
 - Java support
 - C / C++ support
 - Needs a configuration mechanism



Thank You!



browne@vmware.com



github.com/ericwb



@VMWopensource