IoT Security

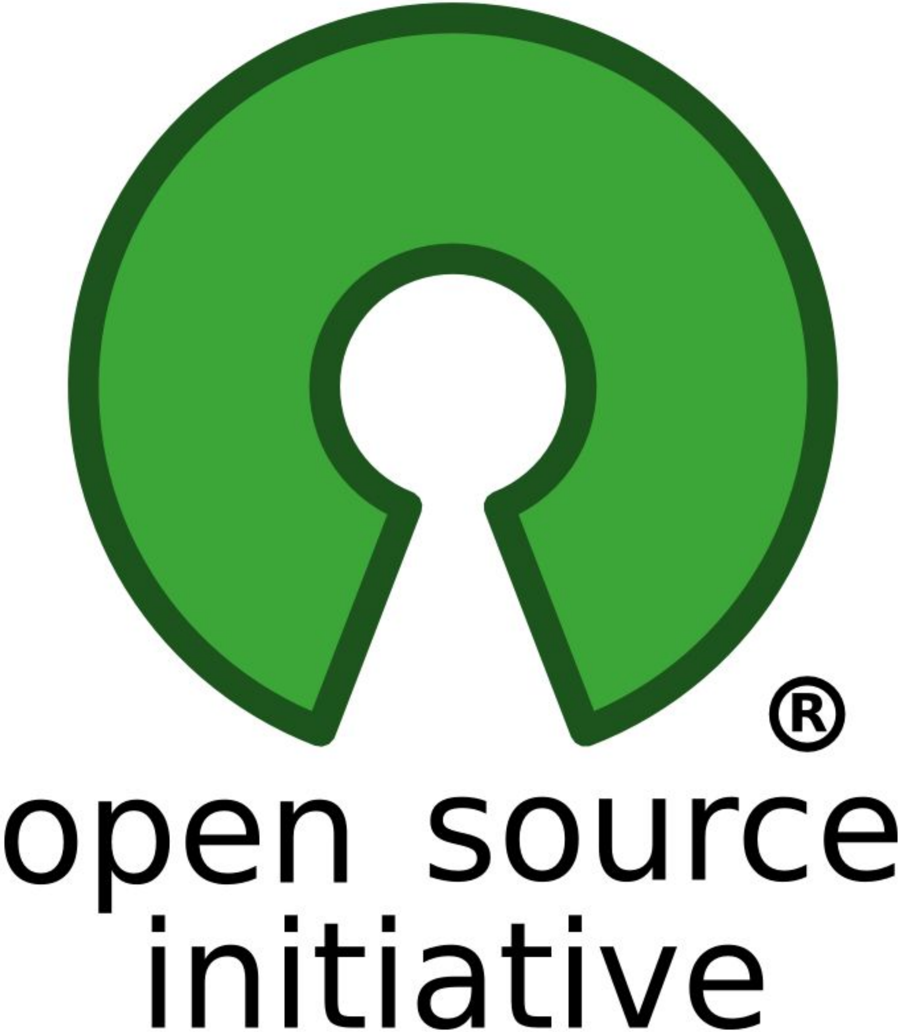
Terri Oda

IoT
The S Stands for Security

But, Terri, there’s no S in IoT.

Exactly.

twill
ely,
water. Lay flat to dry or
orth American product, dyed in
Tencel yarns 5/2,
y Lace Weight/
edle USA,
lain 24 epi twill
separately,
y or
duct, dyed in


®
open source
initiative

What makes IoT security challenging?

1. Things are small

Small processors. Small memory. Small disk. Small physical footprint. Small bandwidth. Small teams. Small code. Small part count. Small development time. Small price. Small performance. Small minds.

2. Things are experimental

“We don't know how this will be used”

“We’ll use this opportunity to use a new tool/framework/programming language”

“While we've got a small team, it’s a good time to try agile (for the first time)”

“Run this project like we're a startup”

“Let’s just throw it out there and see what sticks”

3. Things are exposed

“Everything’s on the internet!”
“We’re going to toss sensors in the forest”
“Your light bulb can talk to your phone and your doorbell and your smoke alarm…”
“This is going in a car.”
“For usability, we need this to connect and work right away, no password.”
https://www.shodan.io/

We’re using resources like it’s 1980.

But we’re doing it all differently (so we can’t use all the tools from the past)

And then we’re putting it in stuff that can kill people.

(I’m not going to take you to 1980 for the hands on portion.)

Reminder:
Ethical security requires consent.

Set up Mission 0 from https://github.com/terriko/sparklesecurity on your raspi (if you haven’t already) and then try at least Mission 1.

[Hands on portion ~ 20 minutes]

Choosing Better Open Source

Special IoT Edition

Developers are not choosing bad packages because they are malicious or stupid.

- Does it meet our needs now and will it do so in the future?
- Does it meet our licensing requirements?
- It is small enough for our desired footprint?
- Is it what others in the industry are using?
- Does it have a good tutorial so we can get someone ramped up to integrate it quickly?

Small processors. Small memory. Small disk. Small physical footprint. Small bandwidth. Small teams. Small code. Small part count. Small development time. Small price. Small performance.

It’s easy to see how security winds up taking a back seat.

How do we avoid that?

Simple Security Risk Assessment

1. Take a first look. Are there warning signs?
2. Check the contributors/activity.
3. Check how they handle security issues.
4. Look at the test suite.
5. Be aware of assumptions.

Step 1:
Take a First Look

First Look: key questions

- Have you read the readme, first pages of the website, and other readily available introductory information?
- Does the code appear to be held to good software development standards?
- Is this code used professionally or is it a hobby project?
- Are there any signs that there are known issues with this code?
- Does this code solve a personal problem for the developer, or is it robust enough for other use cases?
- Is this code active or is it an abandoned archive?
- Are there any warning signs?

The developers tell you to use something else

“use TweetNaCl.js (a TweetNaCl port to JavaScript) rather than this implementation, which is more likely to perform in constant time and has likely seen more eyes for review/audits.”

*******************************DISCLAIMER*******************************
This code is reference software only and is not feature complete. It should not be used in commercial products at this time. Intel makes no claims for the quality or completeness of this code

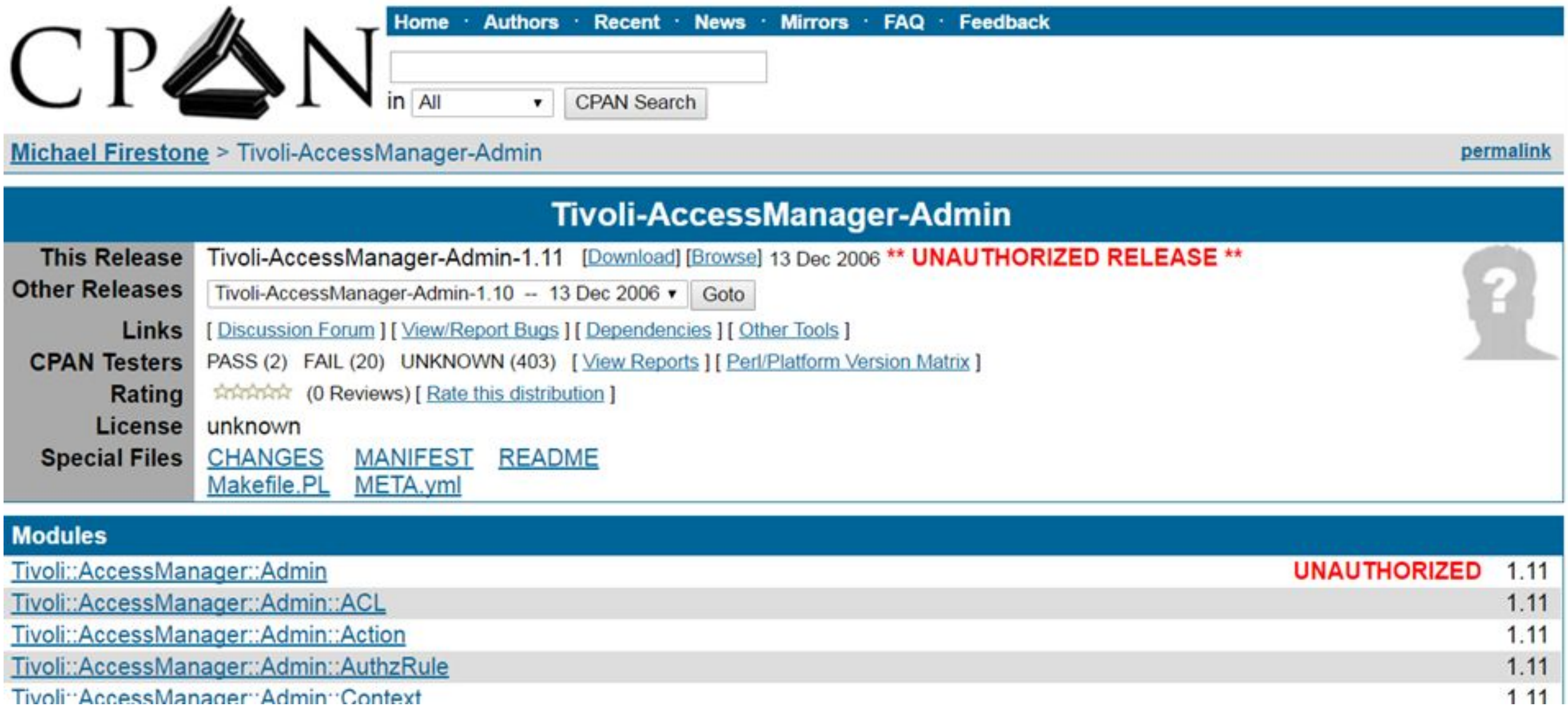
The developers tell you to use something else

“use TweetN
to JavaScrip
implementa
perform in c
seen more e

**   **************

T   omplete. It
s   ntel makes
n

Code of dubious provenance



Michael Firestone > Tivoli-AccessManager-Admin permalink

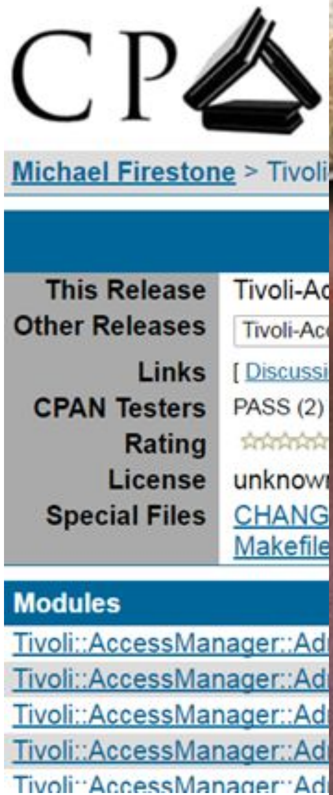
Tivoli-AccessManager-Admin

This Release	Tivoli-AccessManager-Admin-1.11 [Download] [Browse] 13 Dec 2006 ** UNAUTHORIZED RELEASE **
Other Releases	Tivoli-AccessManager-Admin-1.10 -- 13 Dec 2006 ▾ Goto
Links	[Discussion Forum] [View/Report Bugs] [Dependencies] [Other Tools]
CPAN Testers	PASS (2) FAIL (20) UNKNOWN (403) [View Reports] [Perl/Platform Version Matrix]
Rating	☆☆☆☆☆ (0 Reviews) [Rate this distribution]
License	unknown
Special Files	CHANGES MANIFEST README Makefile.PL META.yml

Modules

Tivoli::AccessManager::Admin	UNAUTHORIZED	1.11
Tivoli::AccessManager::Admin::ACL		1.11
Tivoli::AccessManager::Admin::Action		1.11
Tivoli::AccessManager::Admin::AuthzRule		1.11
Tivoli::AccessManager::Admin::Context		1.11

Code of dubious provenance



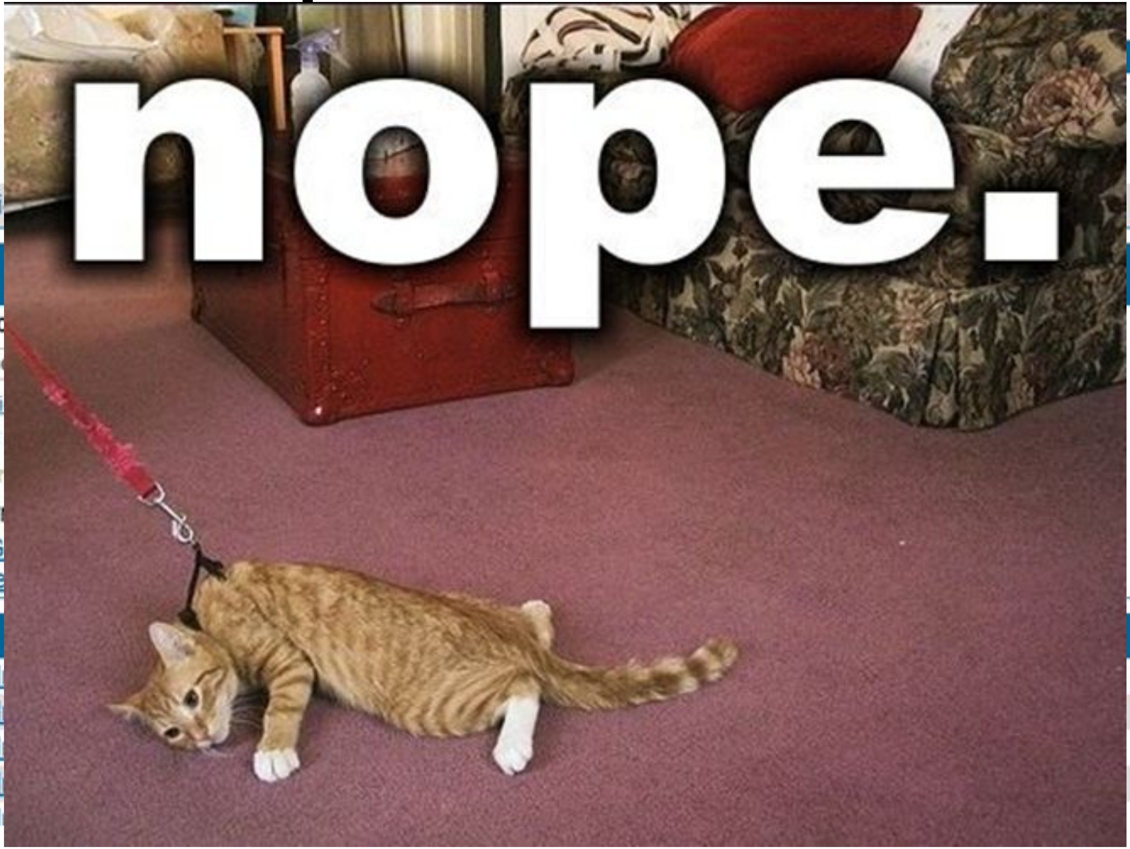
Michael Firestone > Tivoli

permalink

This Release	Tivoli-Ac
Other Releases	Tivoli-Ac
Links	[Discussi
CPAN Testers	PASS (2)
Rating	☆☆☆☆☆
License	unknown
Special Files	CHANG Makefile

Modules

Tivoli::AccessManager::Ad	UNAUTHORIZED	1.11
Tivoli::AccessManager::Ad		1.11
Tivoli::AccessManager::Ad		1.11
Tivoli::AccessManager::Ad		1.11
Tivoli::AccessManager::Ad		1.11



Code of Dubious Quality

"CryptoJS is a project that I enjoy and work on in my spare time, but unfortunately my 9-to-5 hasn't left me with as much free time as it used to. I'd still like to continue improving it in the future, but I can't say when that will be."

"Opencsv was developed in a couple of hours by Glen Smith."

Code of Dubious Quality

“CryptoJS is a
work on in my
unfortunately
with as much
still like to con
future, but I ca

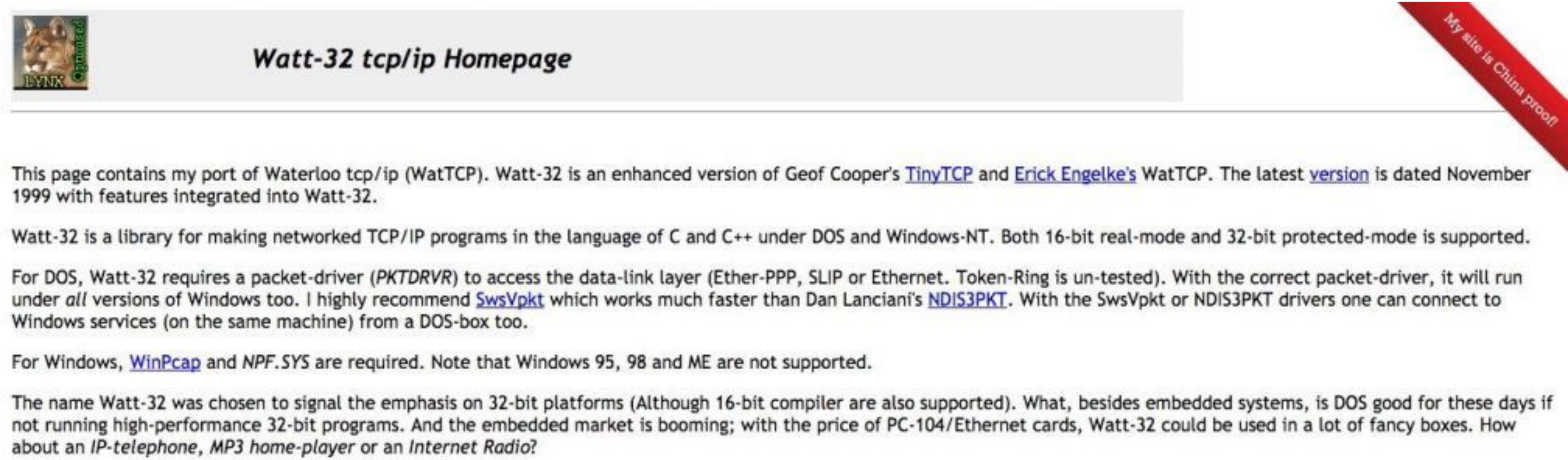
sv was
bed in a
of hours
Smith.”

Known Flaws

“This project was made as a “proof of concept” demonstration of how to detect apps on an iOS device, from early 2011. Since then, it has been extensively used in many apps, to the point where *Apple made the decision to ban* the excessive use of - canOpenURL:, *the method which iHasApp relies upon* to determine app installation. As a result, using a list of URL schemes for app detection is no longer a viable method.”

https://www.cvedetails.com/

Very Old Code



Watt-32 tcp/ip Homepage

This page contains my port of Waterloo tcp/ip (WatTCP). Watt-32 is an enhanced version of Geof Cooper's TinyTCP and Erick Engelke's WatTCP. The latest version is dated November 1999 with features integrated into Watt-32.

Watt-32 is a library for making networked TCP/IP programs in the language of C and C++ under DOS and Windows-NT. Both 16-bit real-mode and 32-bit protected-mode is supported.

For DOS, Watt-32 requires a packet-driver (*PKTDRVR*) to access the data-link layer (Ether-PPP, SLIP or Ethernet. Token-Ring is un-tested). With the correct packet-driver, it will run under *all* versions of Windows too. I highly recommend SwsVpkt which works much faster than Dan Lanciani's NDIS3PKT. With the SwsVpkt or NDIS3PKT drivers one can connect to Windows services (on the same machine) from a DOS-box too.

For Windows, WinPcap and *NPF.SYS* are required. Note that Windows 95, 98 and ME are not supported.

The name Watt-32 was chosen to signal the emphasis on 32-bit platforms (Although 16-bit compiler are also supported). What, besides embedded systems, is DOS good for these days if not running high-performance 32-bit programs. And the embedded market is booming; with the price of PC-104/Ethernet cards, Watt-32 could be used in a lot of fancy boxes. How about an *IP-telephone*, *MP3 home-player* or an *Internet Radio*?

Red Flag Words

“cJSON aims to be the dumbest possible parser that you can get your job done with.”

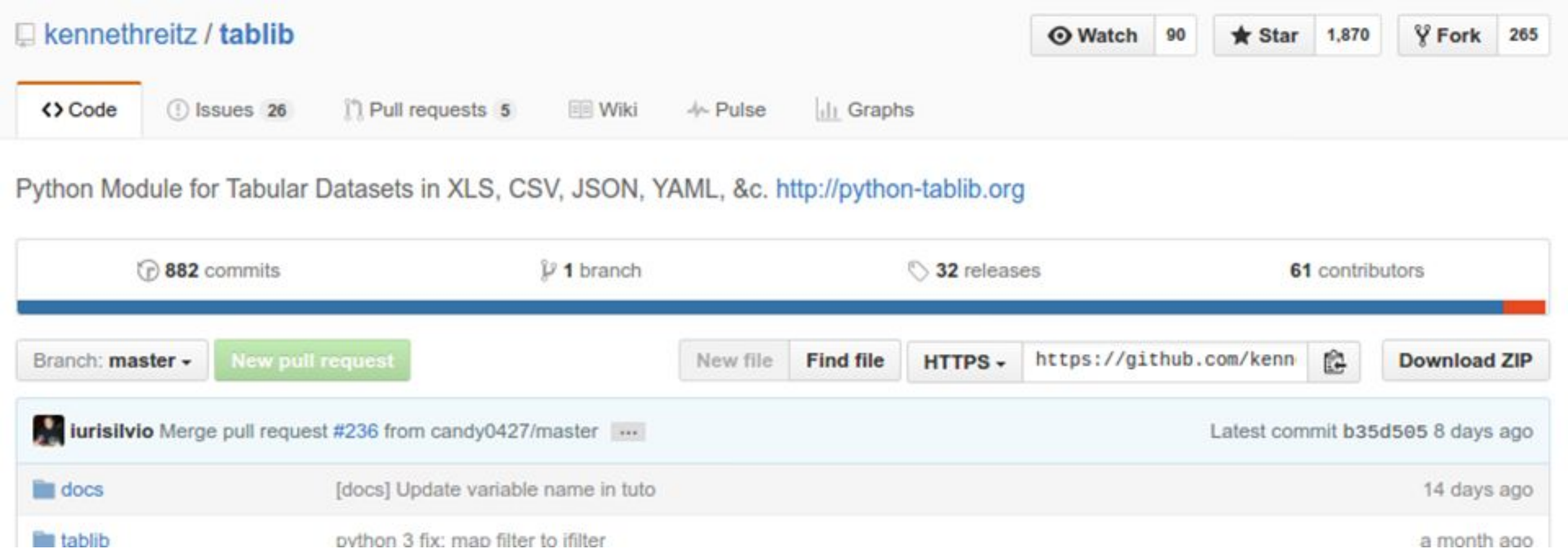
Red Flag Words

- “Elegant,” means to a security person: “We didn’t handle any edge cases.”
- “Lightweight,” means: “We cut out all the input validation.”
- “Fast,” means: “We cut out all the error checking.”
- “We wrote a new parser,” means all of the above.

Step 2:
Check the
Contributors & Activity

Contributors: Key Questions

- How many contributors are *active and significant*?
- Are the key maintainers doing this as part of a job or a hobby?
- Is this code actively maintained, or is it abandoned?
- How many checkins were there in the past year?
- *Are issues being fixed* on a regular basis?
- Who is doing the *code reviews*?
- Who takes over if the main maintainer gets sick?



kennethreitz / tablib

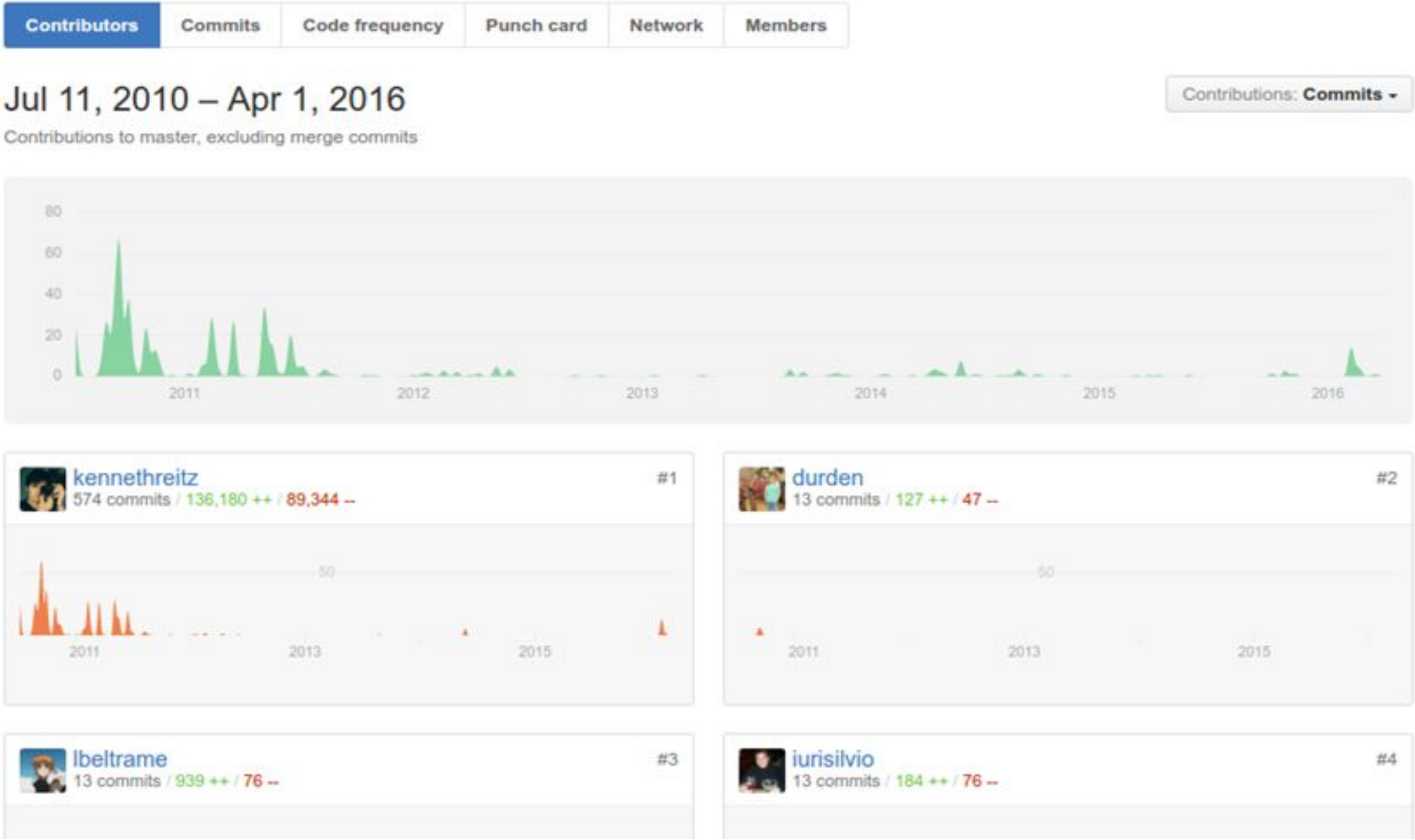
Python Module for Tabular Datasets in XLS, CSV, JSON, YAML, &c. http://python-tablib.org

882 commits 1 branch 32 releases 61 contributors

https://github.com/kenn

iurisilvio Merge pull request #236 from candy0427/master ... Latest commit b35d505 8 days ago

docs	[docs] Update variable name in tuto	14 days ago
tablib	python 3 fix: map filter to ifilter	a month ago



Jul 11, 2010 – Apr 1, 2016

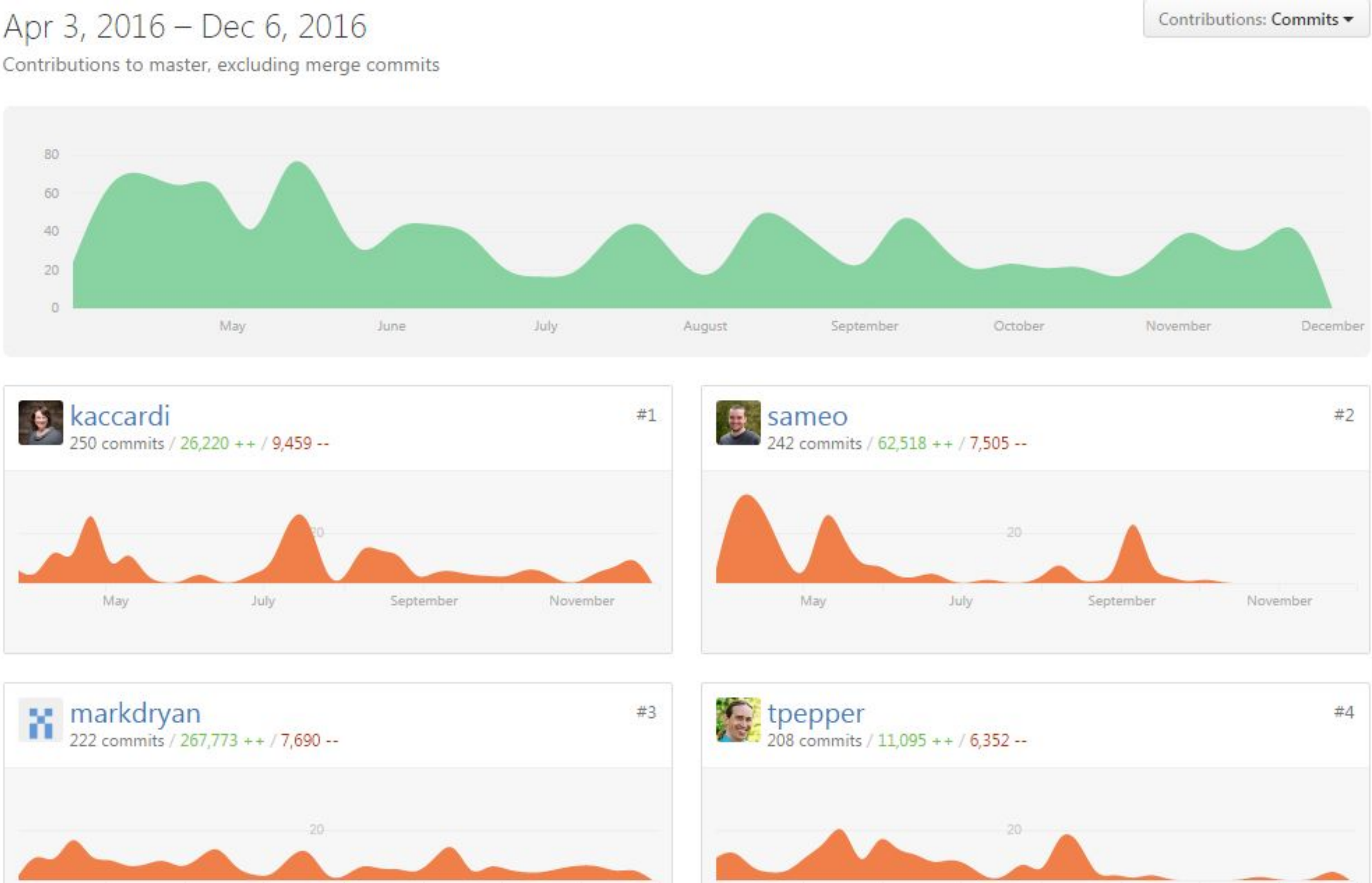
Contributions to master, excluding merge commits

kennethreitz
574 commits / 136,180 ++ / 89,344 --

durden
13 commits / 127 ++ / 47 --

lbeltrame
13 commits / 939 ++ / 76 --

iurisilvio
13 commits / 184 ++ / 76 --



Apr 3, 2016 – Dec 6, 2016

Contributions to master, excluding merge commits

Contributions: Commits ▾

kaccardi #1

250 commits / 26,220 ++ / 9,459 --

sameo #2

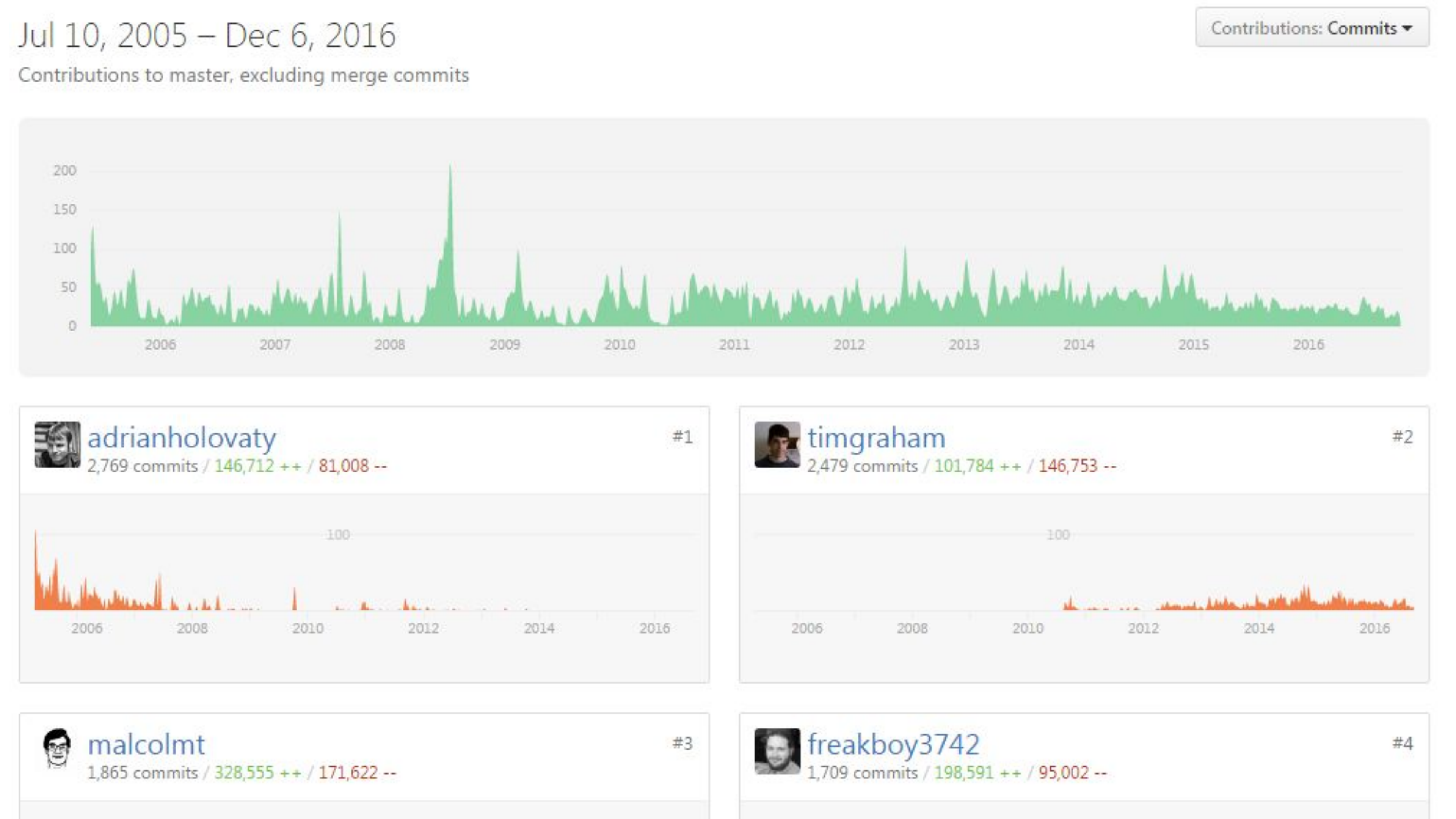
242 commits / 62,518 ++ / 7,505 --

markdryan #3

222 commits / 267,773 ++ / 7,690 --

tpepper #4

208 commits / 11,095 ++ / 6,352 --



Jul 10, 2005 – Dec 6, 2016

Contributions: **Commits** ▾

Contributions to master, excluding merge commits

adrianholovaty #1

2,769 commits / 146,712 ++ / 81,008 --

timgraham #2

2,479 commits / 101,784 ++ / 146,753 --

malcolmt #3

1,865 commits / 328,555 ++ / 171,622 --

freakboy3742 #4

1,709 commits / 198,591 ++ / 95,002 --

Step 3:
Check how they
handle security issues

Security Issues: Key Questions

- Is there a clear way to report security vulnerabilities?
  - An ideal procedure should involve a way to keep the vulnerability secret until a fix is found.
  - Typical good solutions can include sending an email to a special security mailing list or a bug tracker with special “security” flag.
  - If there is no way to report security issues specifically, assume the project has not thought about it (this is a bad sign).
- Is there evidence that vulnerabilities are fixed in a timely manner?
- Is there any explanation of what happens when a security issue is reported?

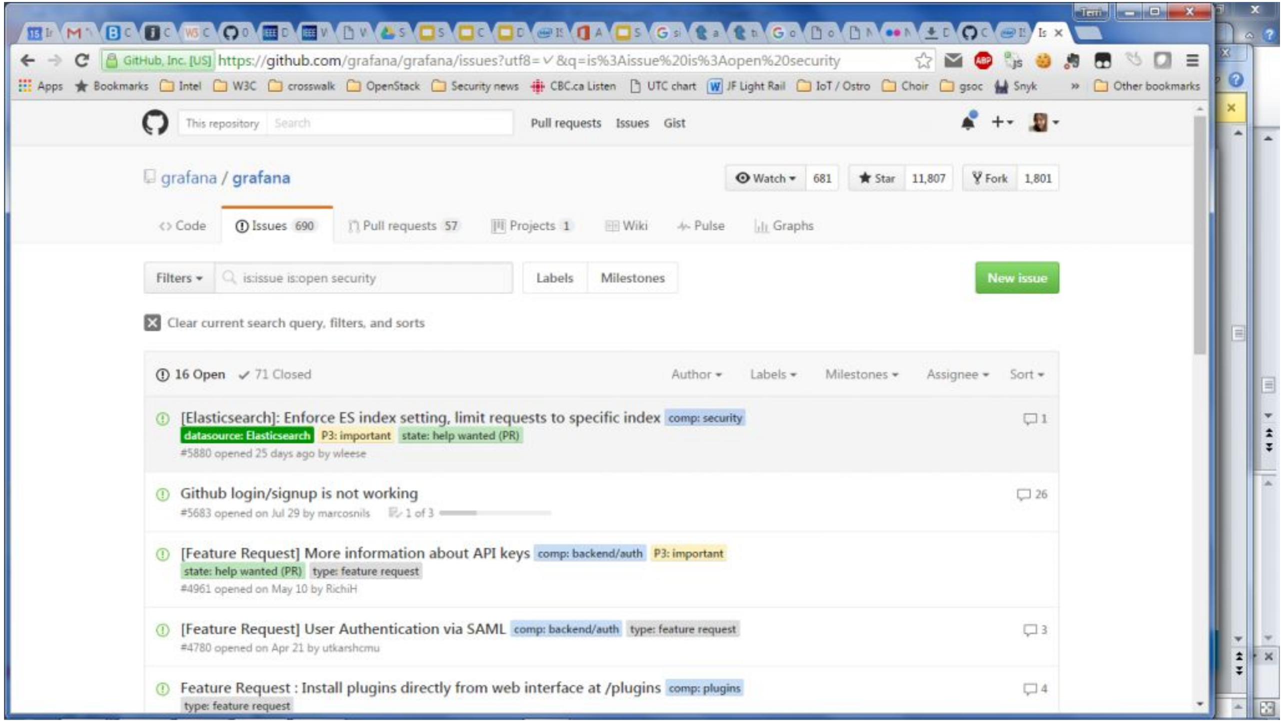
A Couple Good Examples

http://apache.org/security/

https://intel.com/security

No policy?

Try searching for open security issues in the bug tracker/mailing lists



16 Open ✓ 71 Closed

- [Elasticsearch]: Enforce ES index setting, limit requests to specific index `comp: security` `datasource: Elasticsearch` `P3: important` `state: help wanted (PR)` — 1
  #5880 opened 25 days ago by wleese
- Github login/signup is not working — 26
  #5683 opened on Jul 29 by marcosnils 1 of 3
- [Feature Request] More information about API keys `comp: backend/auth` `P3: important` `state: help wanted (PR)` `type: feature request`
  #4961 opened on May 10 by RichiH
- [Feature Request] User Authentication via SAML `comp: backend/auth` `type: feature request` — 3
  #4780 opened on Apr 21 by utkarshcmu
- Feature Request : Install plugins directly from web interface at /plugins `comp: plugins` `type: feature request` — 4

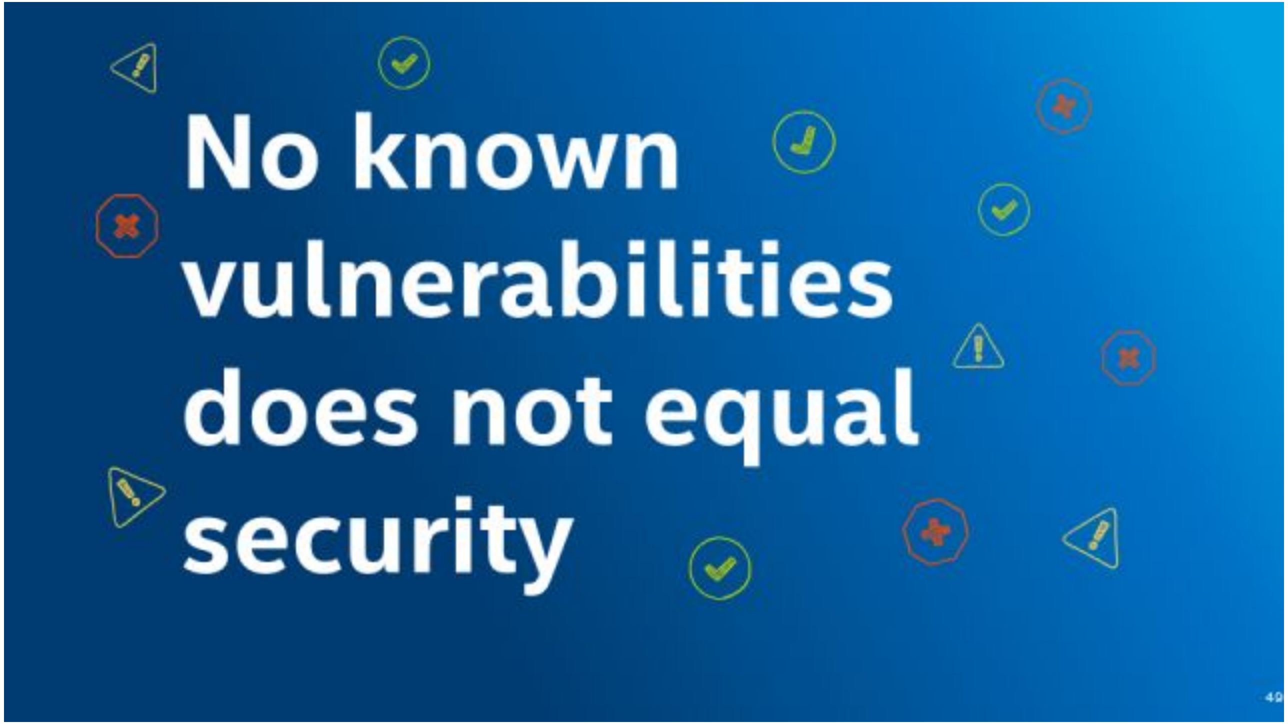
Step 4:
Look at the test suite

Test suite: Key questions

- Does this test suite cover *bad behaviour*?
- How comprehensive is this test suite?
- Do all tests pass?
- Is there continuous integration for tests?

Step 5:
Be Aware of Assumptions

Popularity does not equal security



No known vulnerabilities does not equal security

Good packages do not guarantee good dependencies

Package managers don’t always imply high quality

Good security reporting does not guarantee action

SECTION	GRADE	GRADING GUIDELINES
First look		A - Mentions security audit or other proactive security activity. B - No major warning signs, and code is used professionally. C - No major warning signs, but not widely used or not well-supported. D - Code has minor warning signs that need to be investigated in more detail. F - Code has known issues, major warning signs, or is abandoned
Contributors and activity		A - At least five significant, active contributors. B - More than two significant, active contributors. C - Only one major contributor who is active. D - Project has been inactive for nine months or less. F - Project has been inactive for more than one year.
Security issues		A - Project has had previous security issues and handled them quickly and well. Bonus if they also mention doing proactive security such as fuzz testing, static analysis, or security audits. B - Project has a plan for handling security issues but hasn’t had to use it much yet. C - Project does not have a plan for security issues but at least has an active bug tracker and issues get resolved. D - Project does not seem to resolve many open bugs. F - Project has open security issues that are not in the process of being resolved.
Test suite		A - Project has test suite with good coverage of positive and negative test cases set up as part of continuous integration, and test results are published for each build. B - Project has test suite with good coverage but no continuous integration. C - Test suite mostly covers positive test cases; very few or no error cases. D - Test suite has very low coverage or is only a few examples. F - No test suite.

First look	A - Mentions security audit or other proactive security activity. B - No major warning signs, and code is used professionally. C - No major warning signs, but not widely used or not well-supported. D - Code has minor warning signs that need to be investigated in more detail. F - Code has known issues, major warning signs, or is abandoned

Contributors and activity	A - At least five significant, active contributors. B - More than two significant, active contributors. C - Only one major contributor who is active. D - Project has been inactive for nine months or less. F - Project has been inactive for more than one year.

Security issues	A - Project has had previous security issues and handled them quickly and well. Bonus if they also mention doing proactive security such as fuzz testing, static analysis, or security audits. B - Project has a plan for handling security issues but hasn’t had to use it much yet. C - Project does not have a plan for security issues but at least has an active bug tracker and issues get resolved. D - Project does not seem to resolve many open bugs. F - Project has open security issues that are not in the process of being resolved.

Test suite	A - Project has test suite with good coverage of positive and negative test cases set up as part of continuous integration, and test results are published for each build. B - Project has test suite with good coverage but no continuous integration. C - Test suite mostly covers positive test cases; very few or no error cases. D - Test suite has very low coverage or is only a few examples. F - No test suite.

Activity:

Try a risk assessment on a random open source package!

Use one you care about, or go to

https://github.com/trending

(be aware, those aren’t always appropriate, just scroll on if they’re sketchy)

Scorecard:

https://github.com/sec-princess/WWCode-OSS-Study-Night-20180927/blob/master/OSS%20Component%20Scorecard.pdf

SECTION	GRADE	GRADING GUIDELINES
First look		A - Mentions security audit or other proactive security activity. B - No major warning signs, and code is used professionally. C - No major warning signs, but not widely used or not well-supported. D - Code has minor warning signs that need to be investigated in more detail. F - Code has known issues, major warning signs, or is abandoned
Contributors and activity		A - At least five significant, active contributors. B - More than two significant, active contributors. C - Only one major contributor who is active. D - Project has been inactive for nine months or less. F - Project has been inactive for more than one year.
Security issues		A - Project has had previous security issues and handled them quickly and well. Bonus if they also mention doing proactive security such as fuzz testing, static analysis, or security audits. B - Project has a plan for handling security issues but hasn’t had to use it much yet. C - Project does not have a plan for security issues but at least has an active bug tracker and issues get resolved. D - Project does not seem to resolve many open bugs. F - Project has open security issues that are not in the process of being resolved.
Test suite		A - Project has test suite with good coverage of positive and negative test cases set up as part of continuous integration, and test results are published for each build. B - Project has test suite with good coverage but no continuous integration. C - Test suite mostly covers positive test cases; very few or no error cases. D - Test suite has very low coverage or is only a few examples. F - No test suite.