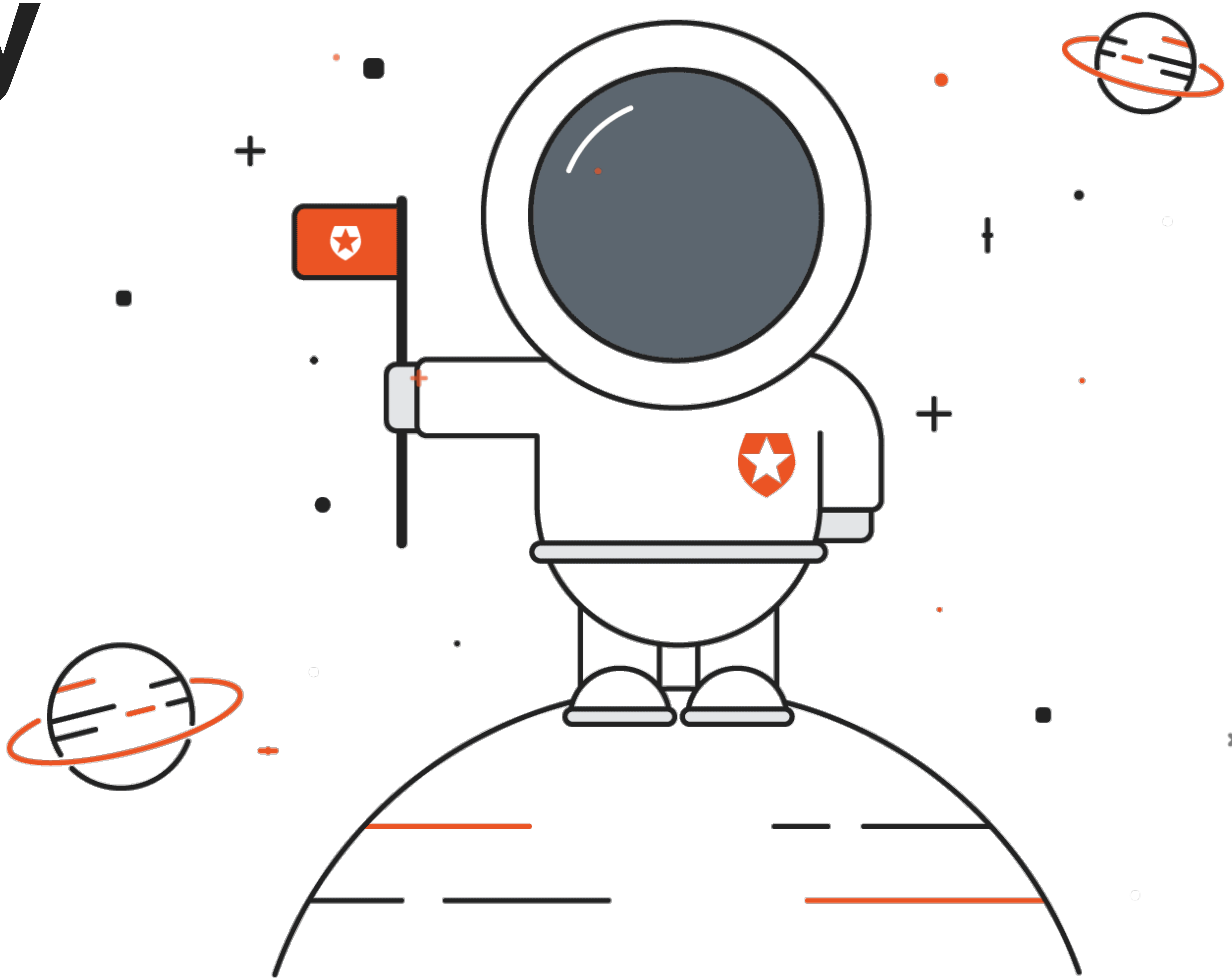
Python Security

Best Practices



Security Matters

@itrwyss

US Data Breaches Statistics

First half of 2019

54%
Increase

3,800+
were reported

3.2 billion
Just 8 of those

@itrwyss

@itrwyss

Had Been Uploaded

38,000
Driver's Licenses

3,200
Passport Details

@itrwyss

Had Stolen

146.6 million
Names and Dates of Birth

145.5 million
Social Security Numbers

99 million
Address

209,000
Payment Card Numbers and Expiration Dates

@itrwyss

GDPR

Data Protection Officer (DPO)

Compliance

25. maj 2018

Databrud

Persondata

@itrwyss

The EU - U.S. Privacy Shield

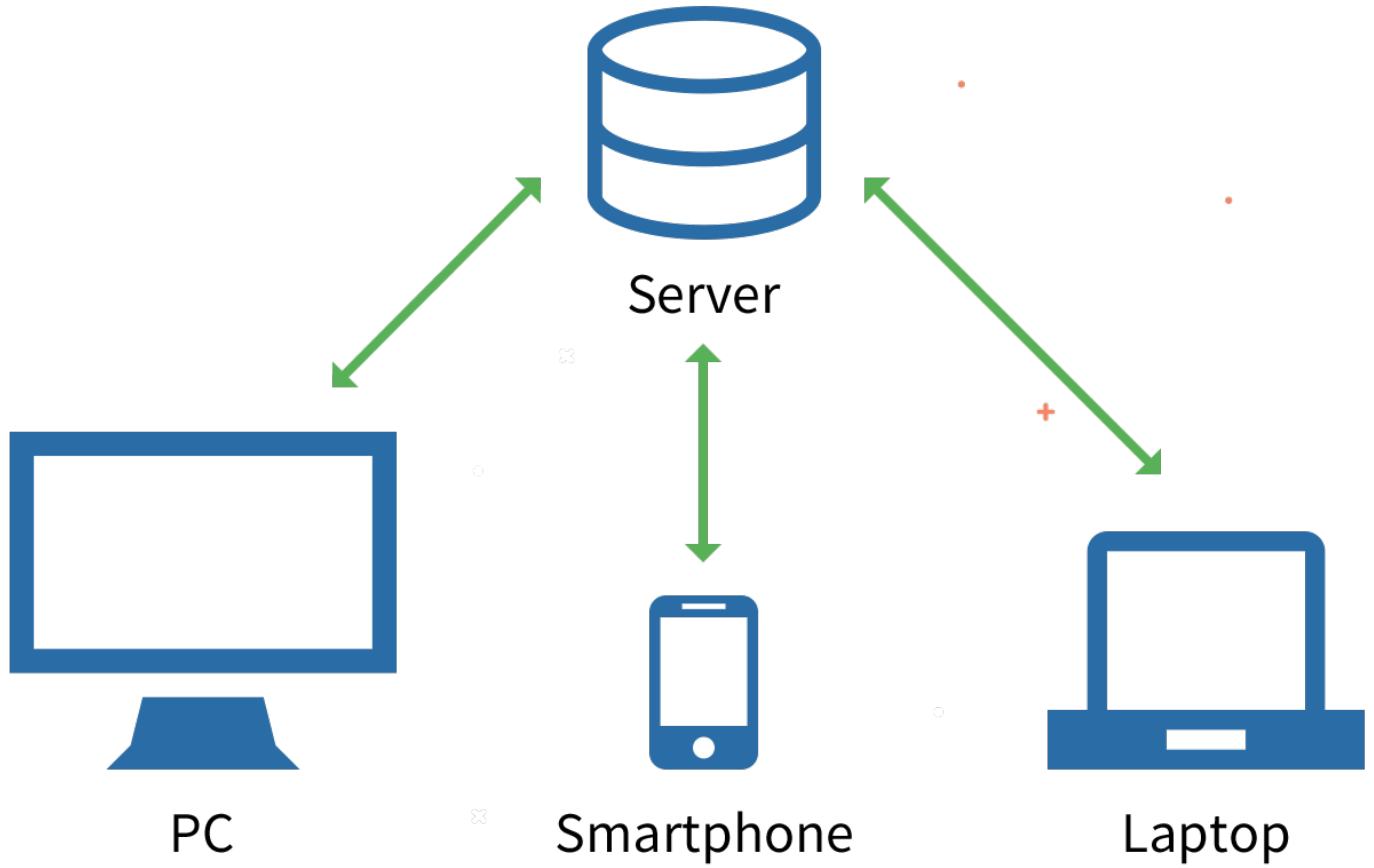
TrustArc

@itrwyss

Security **must** be a <u>goal</u>.

@itrwyss

Security is a **Team Effort**



@itrwyss

Best Practices

Python ~~Security~~

@itrwyss

Mercedes Wyss
@itrjwyss

Community Leader
JDuchess Guatemala & Devs+502

Co-Lead
PyLadies and Women in Data Guatemala City

Mozilla Hispano & Guatemala

Chief Technology Officer (CTO) at Produactivity
Full Stack Developer
(Backend, Android, Frontend)

Auth0 Ambassador &
Oracle Groundbreaker Ambassador

Duke's Choice Award
2016 Winner
Java

AMBASSADOR

Oracle Groundbreaker Ambassador

Devs +502
Comunidad Desarrolladores en Tecnologías < Google > en Guatemala

JDuchess
Guatemala

Code Best Practices

@itrwyss

Use the latest version

- If we don’t upgrade, we leave us open to security vulnerabilities.
  - Python (for **Python 2** support ends on Jan 1, 2020)
  - Framework (**Django 1.11** supports ends on Apr, 2020).
  - Libraries (The last update of **pycrypto** was on Jan 20, 2014)

@itrwyss

Use Python 3

Also the latest version

- Python 2 support ended on January 1, 2020.
- Python 3 was released in 2008.
- Migrate from Python 2 to Python 3 is challenge. Do it as soon as posible.
- Check python.org for most recent release.

@itrwyss

Keep up-to-date on vulnerabilities

- The soon as we know about a vulnerability, we need to act fast to solve them.
    - Upgrade to a new version.
    - Patching.
    - Change our code.
- https://vuln.whitesourcesoftware.com/, Github Bot, Snyk, Pipenv vulnerability checker.

@itrwyss

Use up-to-date libraries

- The fact that a library is so popular doesn't mean it is the best option.
- Check the last update.

@itrwyss

github.com/dlitz/pycrypto

or jump to...

Pull requests Issues Marketplace Explore

dlitz / pycrypto

Watch 125 Star 2.1k Fork 557

Code Issues 143 Pull requests 60 Actions Projects 0 Security 0 Insights

The Python Cryptography Toolkit https://www.dlitz.net/software/pycrypto/

952 commits 14 branches 0 packages 18 releases 23 contributors View license

Branch: master New pull request Create new file Upload files Find file Clone or download

wbolster and dlitz Increase attempts for recovering RSA (p,q) from (n,e,d) ... ✓ Latest commit 7acba5f on Jun 20, 2014

Doc	Release v2.7a1	7 years ago
LEGAL	Move LEGAL/copy/README to COPYRIGHT and update it.	11 years ago
build-aux	Regenerate autoconf files	6 years ago
lib/Crypto	Increase attempts for recovering RSA (p,q) from (n,e,d)	6 years ago
m4	Use autoconf to generate compiler options	7 years ago
src	Increase attempts for recovering RSA (p,q) from (n,e,d)	6 years ago
tools	Fix tools/create-pythons.sh building Python on Linux 3.x or later (sy...	6 years ago
.gitignore	Add support for tox	6 years ago
.travis.yml	Update .travis.yml to test against most Python versions we support (L...	6 years ago
ACKS	Update ChangeLog	9 years ago
COPYRIGHT	Use autoconf to generate compiler options	7 years ago

@itrwyss

github.com/Legrandin/pycryptodome
Pull requests
Issues
Marketplace
Explore
Legrandin / pycryptodome
Used by 11.3k
Watch 49
Star 1.2k
Fork 216
Code
Issues 35
Pull requests 10
Actions
Projects 0
Wiki
Security 0
Insights
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download

A self-contained cryptographic library for Python https://www.pycryptodome.org

2,451 commits | 19 branches | 0 packages | 77 releases | 75 contributors | View license

JeremyRand and Legrandin README: Use HTTPS for homepage link — Latest commit 2233a1a 19 days ago

.circleci	First minimal CI	12 months ago
Doc	Fix big endian counter in image	last month
appveyor	Fix suffix for 64 bit installers	7 months ago
lib/Crypto	Add reference to RFC3279 for (EC)DSA format	last month
src	Isolate NIST ECC tables into translation units	3 months ago
travis	Add sysroot flag to fix OSX builds on travis.io	4 months ago
.gitignore	Bug fixes on RsaKey class	5 years ago
.travis.yml	Fix macOS SDK path for notarization	3 months ago
AUTHORS.rst	typo	16 months ago
Changelog.rst	Fix typo in Changelog	3 months ago

@itrwyss

Be careful when downloading packages

- Instead when PyPI has a procedure for reporting security concerns, not always is up-to-date.
- Double-check that we are using legitimate and updated packages.
- Regardless of whether we use (pip or Pipenv).

@itrwyss

Watch your import statements

- Python imports are very flexible, but that flexibility may has a security cost.
    - Absolute import (entire path)
    - Relative import (explicit or implicit).

```
from safe_package import safe_module

from ..some_package import less_danger
```

@itrwyss

Review your dependency licenses

- To ensure that your project is sustainable and to protect yourself, know what licenses your dependencies use and comply with the terms.

@itrwyss

Never include password in commit

Any kind of credential

- For example, use Git-Secrets or something similar.
- Encrypt sensitive information.
- May 2019, for example, a hacker stole hundreds of passwords saved in plain text in GitHub repositories and demanded a ransom of 0.1 Bitcoin each.

Be careful with String formatting

- When we have user input.

```
from string import Template
greeting_template = Template(“Hello World, my name is $name.”)
greeting = greeting_template.substitute(name=”Hayley”)
```

```
“Hello World, my name is Hayley.”
```

@itrwyss

Protect against SQL injections

- SQL injection is one of the OWASP Top Ten reasons of hacks.
- SQL injections can also drop sensitive data from insecure tables.
- Tip, make queries with object-relational mapping (ORM).

@itrwyss

Handle request safely

- HTTP request are typically handled in Python through the request library.
- We can bypass the SSL certificate verification (only do it on dev), when you trust in the source.

```
requests.get("https://python.org", verify=False)
```

@itrwyss

Deserialize selectively

- Don’t deserialize data from an untrusted source, can result in malicious code. Like using pickle module, part of the standard library.
- There is no way to know whether the object structure that you are “unpickling” is malicious until it is too late.
- This behavior was recently found in NumPy–a popular package for scientific computing.

Set debug = false

- Some Python frameworks (like Django), debug is set to tru by default in new projects.
- Debug can be helpful in development to show errors in our code, but isn't a good practice in production.

@itrwyss

Security Tools

@itrwyss

Use a virtual environment

- Ideally the developer's local environment should be identical to the production environment.
- We can use pip freeze, use a requirements.txt to set up production. This is easy, but not a security conscious option.
- Pinning dependencies, freeze a project to a moment in time.
- Instead of installing packages globally, we use a virtual environment for each project.

@itrwyss

Use Virtualenv

Use a virtual environment

- Supports an isolated Python environment by creating a separate folder for packages used in the specific project.
- Each project’s packages are isolated from each other.

@itrwyss

Use Pipenv

Use a virtual environment

- Tool that manages the competing interests o having a predictable environment and has an up-to-date environment.
- Use two-file system that separates abstract dependency declarations.
- Manages your installations on a virtual environment, displays a dependency tree, and check dependencies for known vulnerabilities.

Use Poetry

Use a virtual environment

- It helps you declare, manage and install dependencies of Python projects, ensuring you have the right stack everywhere.
- It allows projects to have deterministic dependencies with specific package versions, so they build consistently in different places.

Scan Your Code with Bandit

- Bandit scans each .py file and builds a corresponding abstract syntax tree (AST).
- For example, can detect if we are using a framework with debug setting true.
- We can use it locally or as part of a CI/CD pipeline.
- Test plugins, Blacklist plugins, Report formatters.

@itrwyss

```
patch() function to monkey-patch xmlrpclib and mitigate XML vulnerabilities.
   Severity: High   Confidence: High
   Location: ./xml_xmlrpc.py:1
1       import xmlrpclib
2       from SimpleXMLRPCServer import SimpleXMLRPCServer
3

--------------------------------------------------
>> Issue: [B506:yaml_load] Use of unsafe yaml load. Allows instantiation of arbitrary objects. Consider yaml.safe_load().
   Severity: Medium   Confidence: High
   Location: ./yaml_load.py:6
5           ystr = yaml.dump({'a' : 1, 'b' : 2, 'c' : 3})
6           y = yaml.load(ystr)
7           yaml.dump(y)

--------------------------------------------------

Code scanned:
        Total lines of code: 811
        Total lines skipped (#nosec): 14

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 134
                Medium: 119
                High: 64
        Total issues (by confidence):
                Undefined: 0
                Low: 8
```

@itrwyss

Use Pyntch

- Is a static code analysis, can identify potential runtime errors before actually running a code.
- Pyntch gathers the following information:
    - Possible types of objects
    - Functions or instances methods
    - Calling locations
    - Uncaught exceptions

@itrwyss

Use Sqreen

- Checks our application for packages with malicious code.
- Checks for legitimate packages with known problems or outdated versions.

@itrwyss

https://github.com/itrjwyss/PythonSecurity/

https://www.facebook.com/itrjwyss

@itrjwyss