International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 04 Issue: 04 | April -2020 ISSN: 2582-3930

Deserialization in Web Application

Archana Mudaliar [1] , Dr.Priyanka Sharma [2]

[1]Student Master in Technology Cyber Security , [2]Director (R&D) Raksha Shakti University

[1]Department of Information Technology and Telecommunication, Raksha Shakti University

***Abstract* - In recent years there is a surge of serialization-based vulnerabilities in web applications which have resulted in serious incidents such as exposing private data of millions of individuals , logic manipulation or arbitrary code execution. Although there have been some efforts in addressing this problem, there is still no unified solution that is able to detect implementation-agnostic vulnerabilities. We aim to help penetration testers and students as well as to identify and test serialization vulnerabilities on future penetration testing engagements via consolidating research for serialization penetration testing techniques. In addition to that, serialization typically implemented in various platform application server and also web Application. However, this technique had some vulnerabilities and it was discovered in many application server, methods in various web applications. Furthermore, we also introduce the first deserialization test environment which can be used to test deserialization vulnerability detection tools and for educational purposes.**

***Keyword* - Deserialization, Web application vulnerability, php, java, python, serialization.**

I. INTRODUCTION

Insecure deserialization is a vulnerability that occurs when untrusted data are deserialized and used to abuse the application logic, inflict denial of service (DoS) attacks, or even execute arbitrary code. This class of vulnerabilities is included in the ten most critical web application security risks of OWASP. In order to understand what insecure deserialization is, we first must understand what the serialization and deserialization functionalities are. Complex modern systems are highly distributed, as the components communicate with each other and share information (such as moving data between services, storing information, etc.), the native binary format is not ideal for transmission. Serialization, also known as marshaling, refers to a process of converting a native binary object into a format that can be easily stored (for example saved to a file or a database), sent through data streams (for example stdout), or sent over a network. The format in which an object is serialized into, can either be binary or structured text (such as XML, JSON, YAML, etc.) with JSON and XML being two of the most commonly used serialization formats within web applications. On the other hand, deserialization is the exact opposite of serialization, that is, transforming serialized data coming from a file, stream or network socket back to an object identical to the one that the deserialized data came from. Serialization operations are extremely common in architectures that include APIs, microservices, and client-side MVC (Model View Controller).

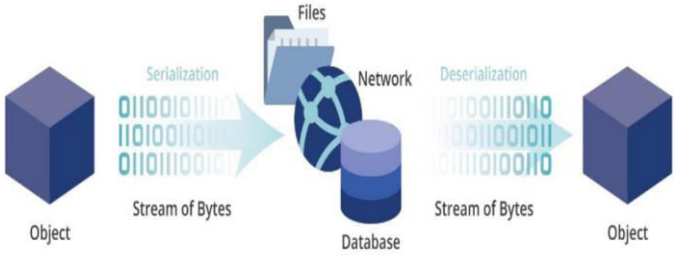

Fig 1. Serilization and Deserialization

Web applications make use of serialization and deserialization regularly and most programming languages even provide native features to serialize data (especially into common formats like JSON and XML). This process is safe as long as the data and objects used, come from trusted and bug-free sources. Using data from any other provenance poses the risk of a malfunction or a deserialization attack since the incoming serialized data could potentially conceal malicious instructions that will force the deserializing program to execute them. It is frequently possible for an attacker to abuse these deserialization features when the application is deserializing untrusted data that the attacker controls. Successful insecure deserialization attacks could allow an attacker to carry out denial-of-service (DoS) attacks, authentication bypasses and remote code execution attacks. Deserialization attack occurrences are abundant both in the past and in recent years where seven documented CVEs related to insecure deserialization are presented, the most recent one was in 2019. It should be noted that the large scale data breach that happened to Equifax in 2017 rooted in insecure deserialization of the struts framework. Furthermore the researchers identified insecure deserialization as a core threat to smart grid systems due to their nature, which requires the transmission of data between nodes that should be serialized and then unserialized in their destination, thus providing a larger than usual attack surface.

Deserialization attacks, despite Java, also affect other languages such as PHP and Python. A PHP Object Injection vulnerability occurs when not sanitized input is used during the deserialization of data in a given web application. The PHP functionalities serialization and deserialization that allow for data storage of any type in a simple string. This format makes it easy to transfer complicated data structures and is often misused to create multidimensional cookies and similar data structures. Since PHP allows deserialization of arbitrary objects, an attacker might be able to inject a specially prepared object with an arbitrary set of properties into the application. Depending on the application implementation, an attacker could trigger internal PHP magic functions which in turn could lead to several vulnerabilities such as code injection, SQL injection, path traversal and application denial of service, depending on the context. Furthermore, python is also vulnerable to these same exploitations through its deserialization functionality. More specifically, an attacker that can control the input to a deserialization python function (pickle.loads(serialized_data)) can forge serialized data that will force the system to run any arbitrary code in the context of the web application. Object injection in PHP is quite common and many recent vulnerabilities have affected large scale applications. More specifically, both WordPress (CVE-2018-20148) and Drupal (CVE-2019-6338) were affected by object injection in vulnerabilities that allowed attackers to execute code, manipulate files and privilege escalation. Other web applications affected include PHPMailer(CVE-2018-19296),Alienvault (CVE-2016-8580) and OpenPSA2(CVE-2018-1000525)

The risk raisers, when an untrusted deserialization user inputs by sending malicious data to be de-serialized and this could lead to logic manipulation or arbitrary code execution.

II. PROGRAMMING LANGUAGE SUPPORT SERIALIZATION

They are many Object-oriented programming support serialization either by using syntactic sugar element or using interface to implement it.This study consented on deserialization vulnerabilities in Java, ruby and php as well as how can these bugs detected, exploit, and Mitigations techniques.

A) Deserialization vulnerability in Java

Java provides serialization where object represented as sequence of bytes, serialization process is JVM independent, which means an object can be serialized in a platform and deserialized on different platform. Java implements serialization using class interface **Java.io.Serializable**, to serialize an object to implement classes **ObjectInputStream** , **ObjectOutputStream** those classes contains several

methods to write/read objects.readObject it is the vulnerable method that leads to deserialization vulnerability it takes serialized data without any blacklisting.

ObjectOutputStream	ObjectInputStream
writeObject: The method writeObject is used to write an object to the stream	readObject: Read an object from the ObjectInputStream.
writeUTF: Primitive data write of this String in modified UTF-8 format.	readUTF : Reads a String in modified UTF-8 format

Fig 2. Difference between ObjectOutputStream and ObjectInputStream

B) Deserialization vulnerability in Python

Python also provides serialization objects like Java and it has many modules including Pickle, marshal, shelve, yaml and finally json it is a recommended module when doing serialization and deserialization. We could observe differences between Java and Python in deserialization vulnerability, Python does not depend on code flow to create payload it simply deserializes all classes this behavior may lead to RCE Serialization which could be found in parameters or cookies. Firstly we explore Pickle taking into account what is mentioned in Python documentation. The pickle module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure. Deserialized untrusted data can compromise the application.

Warning: The pickle module is not secure against erroneous or maliciously constructed data. Never un-pickle data received from an untrusted or unauthenticated source.

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

Fig 3. Pickle module provides functions

C) Deserialization vulnerability in PHP

PHP is like Java and Python, PHP also supports serialization and issues of using serialization it has two methods to implement serialization and deserialization we explore theme and also a case where we can make web application vulnerable. Serialize it simply by converting object to bytes that could be stored. unserialize it simply by converting bytes to object again from here came vulnerability like Python and Java which serializes untrusted data to expose web application. Exploit deserialization in Java depends on code flaw and in Python doesnt depend on any flow in code, but in PHP depends on code flow inside magic methods. Serialization could be found in parameters, cookies. Magic method which are used are as follows.

- __reduce__ to reconstruct our payload when it deserializes something like PHP but it depends on code flaw after calling magic method.
- __sleep is called when an object is serialized and must be returned to array.
- __wakeup is called when an object is deserialized.
- __destruct is called when PHP script end and object is destroyed.
- __toString uses object as string but also can be used to read file or more than that based on function call inside it.

III. CONCLUSION

In this work, we analyzed the need for a more sound and complete approach when it comes to detecting deserialization vulnerabilities. Although this class of vulnerabilities has produced large-scale attacks with a considerable economic and social impact, there is still a lack when it comes to complete and automatic deserialization vulnerability detection tools and related research.

.

REFERENCES

[1]https://www.owasp.org/index.php/Deserialization_of_untrusted_data

[2] Babak Amin Azad, Pierre Laperdrix, and Nick Nikiforakis. 2019. Less is more: quantifying the security benefits of debloating web applications. In 28th { USENIX } Security Symposium ({ USENIX } Security 19). 1697–1714.

[3] Davide Balzarotti, Marco Cova, Vika Felmetsger, Nenad Jovanovic, Engin Kirda, Christopher Kruegel, and Giovanni Vigna. 2008. Saner: Composing static and dynamic analysis to validate sanitization in web applications. In 2008 IEEE Symposium on Security and Privacy (sp 2008). IEEE, 387–401.

[4] https://github.com/mbechler/serianalyzer

[5] Brianwrf. 2015. hackUtils. Retrieved September 1, 2019 from https://github.com/ brianwrf/hackUtils

[6] https: //github.com/ikkisoft/SerialKiller

[7] https://github.com/apache/commons-io/blob/master/src/main/java/org/apache/commons/io/serialization/