What is insecure deserialization?

Insecure deserialization is what happens when an application converts attacker-controlled bytes back into objects without verifying they are safe. Because deserialization can invoke constructors, magic methods, and property setters, a crafted payload can chain existing classes (a gadget chain) into arbitrary behavior, frequently remote code execution. It affects Java (via libraries like Commons Collections), PHP (unserialize with magic methods), Python (pickle), .NET, and Ruby, and the root cause is trusting serialized input.

HOW IT WORKS

01 The abuse and payload

Testers first spot serialized data, then build or reuse a gadget chain for the stack. Shown for defensive context:

- Spot it: Java serialized blobs start with the bytes `ac ed` (base64 `r00`); PHP serialized strings look like `O:4:"User":...`; Python pickle and .NET have their own signatures.
- Java: a tool like ysoserial generates a payload from a known gadget chain, for example `java -jar ysoserial.jar CommonsCollections5 'id'`, delivered where the app deserializes it.
- PHP: craft an object whose `__wakeup` or `__destruct` reaches a sensitive sink, a classic POP (property-oriented programming) chain.
- Python: an object whose `__reduce__` returns `(os.system, ('id',))` runs a command the moment `pickle.loads` processes it.

Because exploitation happens during parsing, it is easy to miss in a code read and is a priority target in web app penetration testing.

HOW TO DEFEND

- Prefer a data-only format such as JSON with a strict schema, and never native serialization for data that crosses a trust boundary.
- If native serialization is unavoidable, add integrity protection, sign the payload and verify it before deserializing, and restrict allowed classes with a strict allowlist.
- Keep gadget-bearing libraries patched and minimal; every extra library on the classpath is another potential gadget source.
- Isolate and monitor deserialization endpoints so an attempt is logged and contained.

SOURCES

[1] OWASP: Deserialization Cheat Sheet

[2] PortSwigger: Insecure deserialization

[3] MITRE ATT&CK: Exploit Public-Facing Application (T1190)

Find the gadget chain before an attacker does.

securelayer7.net/learn/web-exploitation/what-is-insecure-deserialization

Open online