INSECURE DESERIALIZATION

By: Vie

CONTEXT: WHAT IS SERIALIZATION

Taking a complex object (and its fields) or data structure and converting it into a flatter format, which can be sent and recieved as a stream of bytes

Deserialization is the opposite process - taking the bytes and turning them into a functional copy of the object.

THE VULNERABILITY

When user-controllable data is deserialized by an application
↳ Can enable an attacker to manipulate serialized objects in order to pass harmful data

IDENTIFYING THE EXPLOIT

Try to identify what serialized data looks like
Watch for where user-input is sent

PYTHON-PICKLING

- Pickle module used for serializing (pickling) and deserializing (unpickle) Python object structure
- Pickle data format is a printable ASCII representation (although a binary format can be specified in newer prot

USAGE

- **Serializing**
  - create a pickler
  - call pickler's `dump()` method
  - pickle.dump(obj, file)
    ↳ writes pickled representation of obj to file.
  - pickle.dumps(obj)
    ↳ returns pickled representation without writing to a file
- **Deserializing**
  - create an unpickler
  - call unpickler's `load()` method
  - pickle.load(file)
    ↳ Read a string from file and interpret as a pickle data stream, reconstructing then returning original object
  - pickle.loads(string)
    ↳ process string as an object hierarchy

__REDUCE__

- When the pickler encounters issues in serializing an object, it will try to refer to an obj's override of the `__reduce__` method for a directive
- __reduce__ should return a string/tuple
  ↳ tuple should be:
  - a callable
  - tuple of arguments to call the callable on
- pickle will serialize each of these pieces seperately, then on unpickling, will call the callable on the provided arguments to construct the new object

EXAMPLE

```
import cPickle
import subprocess
import base64

class RunBinSh(object):
    def __reduce__(self):
        return (subprocess.Popen,(('/bin/sh',),))

print base64.b64encode(cPickle.dumps(RunBinSh()))
```

we override reduce method to open a shell.

When this gets unpickled, '/bin/sh' will immediately run.

JAVA-BINARY SERIALIZATION

- Java serializes in binary format
- Java serialized objects have unique signatures:
  - starts with AC ED 00 05 (hex), or rO0 (base64)
  - content-type header of HTTP response is application/x-java-serialized-object

- Any class that implements 'java.io.serializable' can be serialized and deserialized.

PHP

- mostly uses human-readable string format
  - ↳ letters represent the data type
    numbers represent the length of each entry

USAGE

- Serialize(): PHP object → string
- Unserialize(): string → original object

STRING STRUCTURE

- Basic structure of a PHP serialized string is "data type:data"

  - b:BOOLEAN boolean

  - i:INTEGER integer

  - d:FLOAT float

  - s:STRING_LENGTH:"STRING_CONTENT" string

  - a:NUM_ELEMENTS:{ELEMENTS} array

  - O:NAME_LENGTH:"CLASS_NAME":NUM_PROPERTIES:{PROPERTIES} object

EXAMPLE

O:4:"User":2:{s:8:"username";s:6:"vickie";s:6:"status";s:9:"not admin";}

- O:4:"User" — object called "User"
- :2: — 2 properties:
  - username: vickie
  - status: not admin
- s:8:"username";s:6:"vickie" — 1st property
- s:6:"status";s:9:"not admin" — 2nd property

DESERIALIZATION 'MAGIC' METHODS - WAKEUP & DESTRUCT

- If the class of the serialized object implements any method named __wakeup() and __destruct(), those methods will be executed automatically when the object is deserialized
  - ↳ __wakeup(): when an object is instantiated from a class in memory (after unserialize()), it will search for a function called '__wakeup()', then execute code in that function, which will reconstruct any resources the object may have
  - ↳ __destruct(): called to clean up the object when no reference to it exists.
- if we control the serialized string which will be passed to unserialize(), we may be able to control the application flow by manipulating the values passed into __wakeup or __destruct
  - ↳ called: "PHP object injection"

EXAMPLE

```
class Example2
{
  private $hook;
  function __construct(){
          //some PHP code...
    }
  function __wakeup(){
          if (isset($this->hook)) eval ($this->hook);
    }
}
...
$user_data = unserialize($_COOKIE['data']);
...
```

↑ RCE can be achieved. A user-provided object is passed into unserialize();
The example2 class has the magic __wakeup() function that will run eval() on user-provided input

We just need set our data cookie to a serialized example2 object with the 'hook' property set to whatever PHP code.

```
class Example2
 {
    private $hook = "evilFunc();";

 }
```

```
print urlencode(serialize(new Example2));
```

Need to use URL encode, since we're injecting via URL

When this object is passed to unserialize()...

- object is passed to program as data cookie
- unserialize(data)
- unserialize() instantiates a new Example2 object.
- Example2 class has a __wakeup() implementation, so __wakeup() is called
- wakeup looks to see if $hook is set. It calls eval($hook);