POP-CHAINS

"property-oriented programming" - fun with PHP's unserialize function.

- PHP object injection gadgets

CONTEXT

PHP Object Injection: PHP-specific insecure deserialization (POI)

POI relies on the fact that magic functions can be automatically invoked when an object is unserialized.

If user data is unserialized, an attacker can control all properties of an object and do all kinds of things.

EXPLOITS IN DETAIL

- Magic methods `__construct()` and `destruct()` are invoked during PHP deserialization.
  - ↳ The magic methods are the initial gadgets to start a POP chains

- They might call other functions, which will be other gadgets.

- The problem arises when we freely control the properties to a deserialized object.

Example

```
class File{
    public function __destruct() {
        $this->shutdown();
    }

    public function shutdown() {
        $this->handle->close();
    }
}
```

- We can construct a serialized string for a File object

- `__destruct()` is a magic method that is invoked automatically when an object is destroyed
  - ↳ In File, `__destruct()` calls shutdown(), which retrieves a property in the object called handle and then calls close() on it.

Say that there exists another class called Process:

```
class Process {
    public function close() {
        system('kill '. $this->pid);
    }
}
```

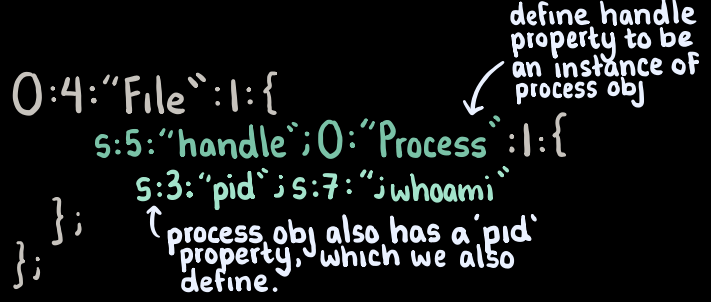
- Process has no magic methods, but it does have a property called pid, which is used in a system command in the close() function.

- Although Process' close() function looks attractive to exploit, we can't invoke it on our own...or can we?

If we have the following serial string...

```
O:4:"File":1:{
    s:5:"handle";O:"Process":1:{
        s:3:"pid";s:7:";whoami"
    };
};
```

- Serialized string that will reconstruct a File object
  - ↳ we define one of File's property: handle, and we specify that handle is a process object.
  - → we further define properties in handle, AKA the 'pid' property



- When this object is destroyed, the `__destruct()` method is called and the chain starts.
- The program calls shutdown(), which will look for a property called handle and invokes close(). The program sees that handle is an object of type 'Process' and invokes that class' ver of close()

·In process close(), we invoke a 'kill' sys command and concat the process property 'pid' to it. We defined 'pid' = 'whoami', so we will be able to execute a whoami command.

·What just happened?
↳ Chained together multiple snippets of code to do unintended things, using a magic method (from unserializing objects) to kick things off.
magic method = initial gadget

·We combined
- a class that, when garbage-collected, can invoke an arbitrary class' member method
- another class' member method that executes arbitrary PHP code.

·POP chains = ROP but with PHP properties invoked with a magic method.

EXPLOIT CHARACTERISTICS

·Magic methods, typically invoked during obj deserialization (POI)
·Control of object's properties, so we can point to other classes to run + chain together snippets of useful code.
·Presence of multiple classes to create objects out of
·Objects passed into PHP unserialize()

POP gadget examples

Arbitrary file write
```
@unlink($fileobject)
file_put_contents($this->file)
```

Code Execution
```
eval($this->injectobject);
```

Auth bypass
```
if (isset($this->obj)) return $this->
  obj->getValue();
($obj->check === $obj->secrethash)
  {echo "Pass";}
```

Type Juggling
```
if ($username == $adminName &&
  $pass == $adminpw){...}
```
↳ user input taken + treated as array and type conversion occurs.

NOTES