python http.server open redirect vulnerability

phithon / 2018-01-02 21:43:47 / 浏览数 3873 安全技术 漏洞分析 顶(0) 踩(0)

python http.server open redirect vulnerability

（本文来自 https://www.leavesongs.com/PENETRATION/python-http-server-open-redirect-vulnerability.html 支持一下先知论坛的改版~）

Github账号被封了以后，Vulhub也无法继续更新了，余下很多时间，默默看了点代码，偶然还能遇上一两个漏洞，甚是有趣。

这个漏洞出现在python核心库http中，发送给官方团队后被告知撞洞了，且官方也认为需要更多人看看怎么修复这个问题，所以我们来分析一下。

0x01 http.server库简单分析

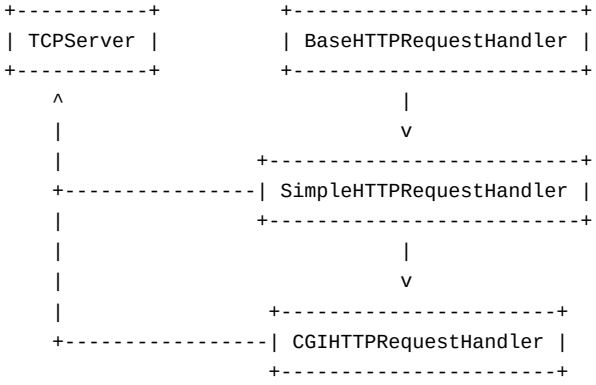
众所周知Python有一个一键启动Web服务器的方法：

```
python3 -m http.server
```

在任意目录执行如上命令，即可启动一个web文件服务器。其实这个方法就用到了`http.server`模块。这个模块包含几个比较重要的类：

1. `HTTPServer`这个类继承于`socketserver.TCPServer`，说明其实HTTP服务器本质是一个TCP服务器
2. `BaseHTTPRequestHandler`，这是一个处理TCP协议内容的Handler，目的就是将从TCP流中获取的数据按照HTTP协议进行解析，并按照HTTP协议返回相应数据包。
3. `SimpleHTTPRequestHandler`，这个类继承于`BaseHTTPRequestHandler`，从父类中拿到解析好的数据包，并将用户请求的path返回给用户，等于实现了一个静态
4. `CGIHTTPRequestHandler`，这个类继承于`SimpleHTTPRequestHandler`，在静态文件服务器的基础上，增加了执行CGI脚本的功能。

简单来说就是如下：



我们看看`SimpleHTTPRequestHandler`的源代码：

```
class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):
   server_version = "SimpleHTTP/" + __version__

   def do_GET(self):
       """Serve a GET request."""
       f = self.send_head()
       if f:
           try:
               self.copyfile(f, self.wfile)
           finally:
               f.close()

   # ...

   def send_head(self):
       path = self.translate_path(self.path)
       f = None
       if os.path.isdir(path):
           parts = urllib.parse.urlsplit(self.path)
           if not parts.path.endswith('/'):
               # redirect browser - doing basically what apache does
               self.send_response(HTTPStatus.MOVED_PERMANENTLY)
               new_parts = (parts[0], parts[1], parts[2] + '/',
                            parts[3], parts[4])
```

```
                new_url = urllib.parse.urlunsplit(new_parts)
                self.send_header("Location", new_url)
                self.end_headers()
                return None
            for index in "index.html", "index.htm":
                index = os.path.join(path, index)
                if os.path.exists(index):
                    path = index
                    break
            else:
                return self.list_directory(path)
        # ...
```

前面HTTP解析的部分不再分析，如果我们请求的是GET方法，将会被分配到do_GET函数里，在do_GET()中调用了send_head()方法。

send_head()中调用了self.translate_path(self.path)将request
path进行一个标准化操作，目的是获取用户真正请求的文件。如果这个path是一个已存在的目录，则进入if语句。

如果用户请求的path不是以/结尾，则进入第二个if语句，这个语句中执行了HTTP跳转的操作，这就是我们当前漏洞的关键点了。

0x02 任意URL跳转漏洞

如果我们请求的是一个已存在的目录，但PATH没有以/结尾，则将PATH增加/并用301跳转。

这就涉及到了一个有趣的问题：在chrome、firefox等主流浏览器中，如果url以//domain开头，浏览器将会默认认为这个url是当前数据包的协议。比如，我们访问http:/

所以，如果我们发送的请求的是GET //baidu.com HTTP/1.0\r\n\r\n，那么将会被重定向到//baidu.com/，也就产生了一个任意URL跳转漏洞。

在此前，由于目录baidu.com不存在，我们还需要绕过if
os.path.isdir(path)这条if语句。绕过方法也很简单，因为baidu.com不存在，我们跳转到上一层目录即可：

```
GET //baidu.com/%2f.. HTTP/1.0\r\n\r\n
```

如何测试这个漏洞呢？其实也很简单，直接用python3 -m
http.server启动一个HTTP服务器即可。访问http://127.0.0.1:8000//example.com/%2f%2e%2e即可发现跳转到了http://example.com/%2f../。

0x03 web.py任意URL跳转漏洞

那么，虽然说python核心库存在这个漏洞，不过通常情况下不会有人直接在生产环境用python -m http.server。

Python框架web.py在处理静态文件的代码中继承并使用了SimpleHTTPRequestHandler类，所以也会受到影响。

我们可以简单测试一下，我们用web.py官网的示例代码创建一个web应用：

```
import web

urls = (
   '/(.*)', 'hello'
)
app = web.application(urls, globals())


class hello:
   def GET(self, name):
       if not name:
           name = 'World'
       return 'Hello, ' + name + '!'


if __name__ == "__main__":
   app.run()
```

然后模拟真实环境，创建一个static目录，和一些子目录：

```
static
■■■ css
■   ■■■ app.css
■■■ js
   ■■■ app.js
```

运行后，直接访问http://127.0.0.1:8080////static%2fcss%2f@www.example.com/..%2f即可发现已成功跳转。

web.py的具体分析我就不多说了，由于请求必须有/static/前缀，所以利用方法有些不同，不过核心原理也无差别。

上一篇：Misc 总结 ----隐写术之电... 下一篇：企业安全建设—模块化蜜罐平台的设计...

1. 0 条回复
   - 动动手指，沙发就是你的了！