**Bolt CMS <= 3.7.0漏洞利用**

# 一、漏洞描述

## 1、漏洞简述

Bolt CMS <= 3.7.0存在xss和远程代码执行漏洞。允许登录用户上传文件并更改文件名为\*.Php

# 二、机理分析

## 1、XSS漏洞

**影响版本：Bolt CMS <= 3.7.0**

**漏洞编号：**无

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、远程代码执行漏洞

**影响版本：Bolt CMS <= 3.7.0**

**漏洞编号：**无

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'

# 三、攻击路径

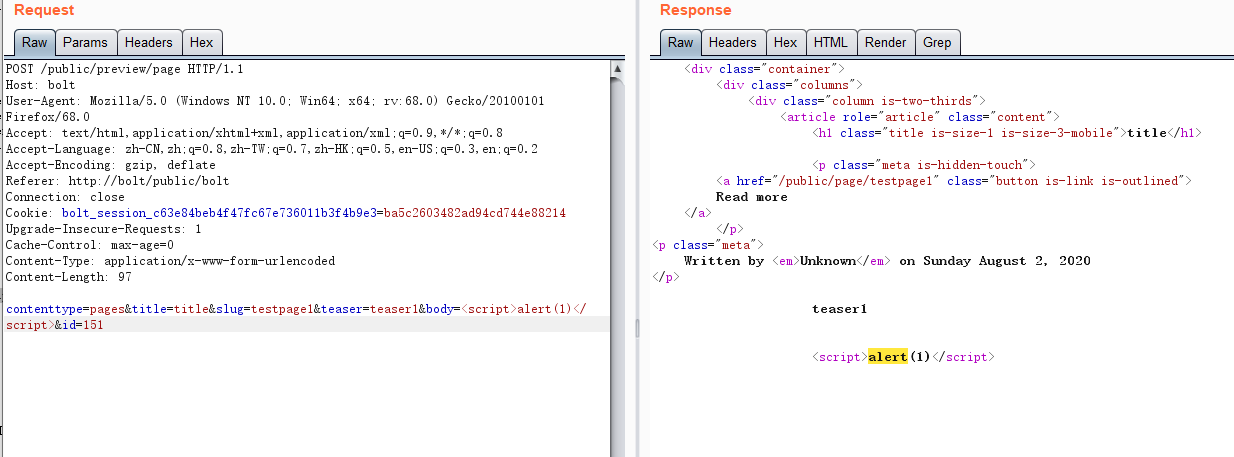
## 1. xss

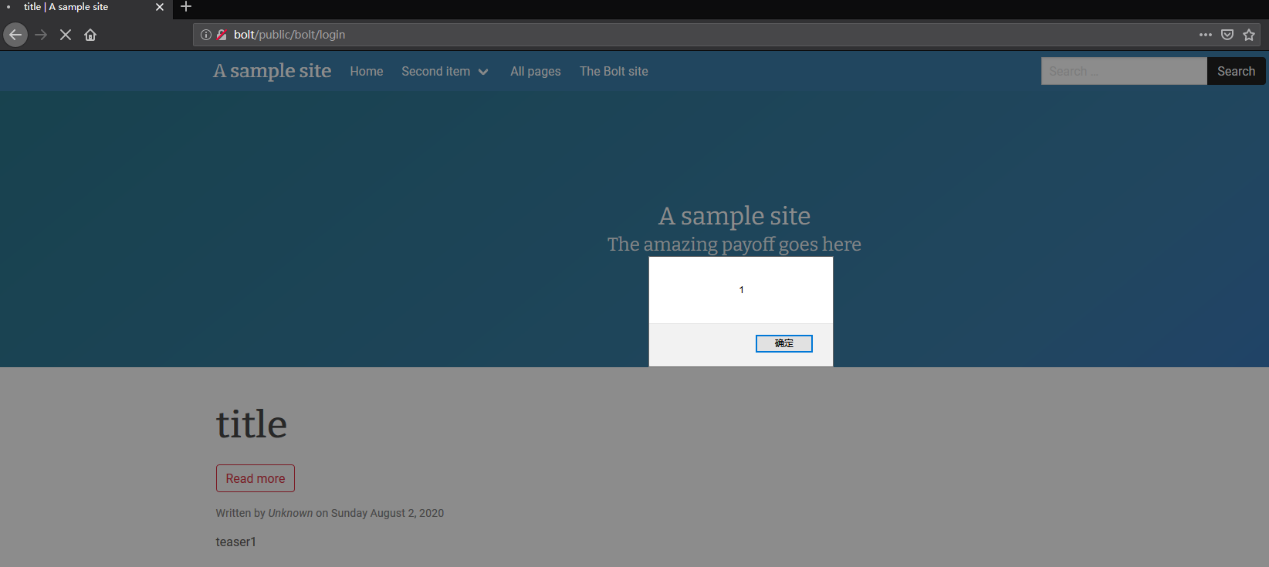
构造payload

POST /preview/page HTTP/1.1

Host: localhost

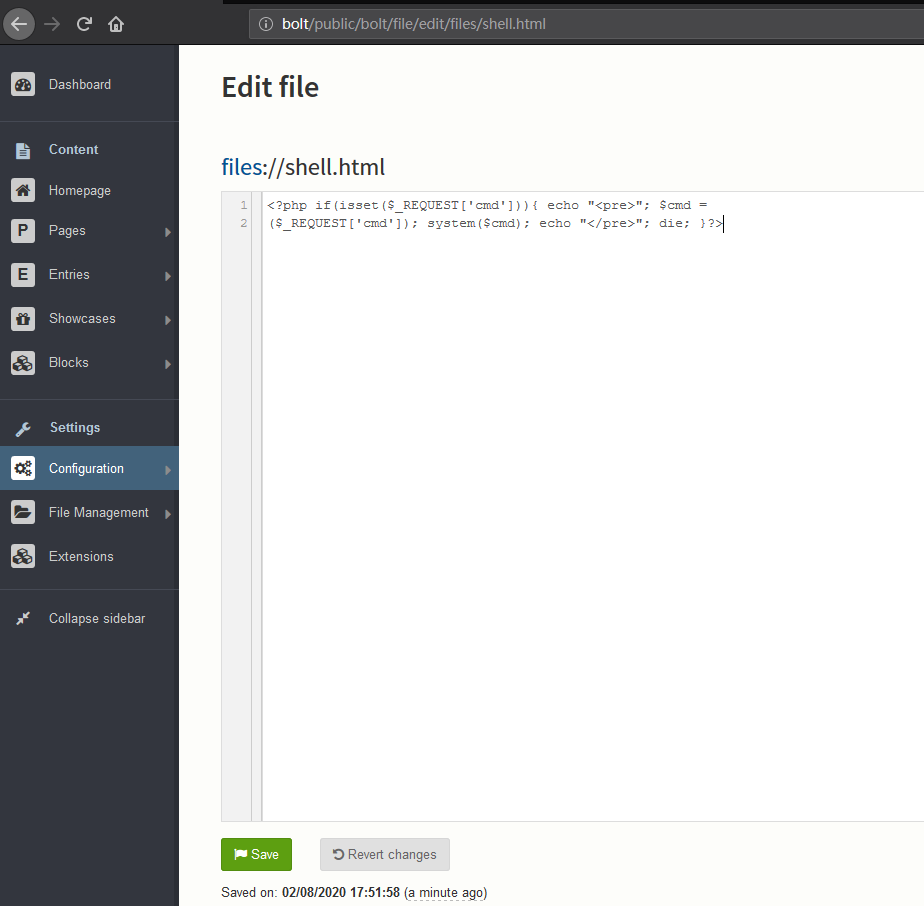
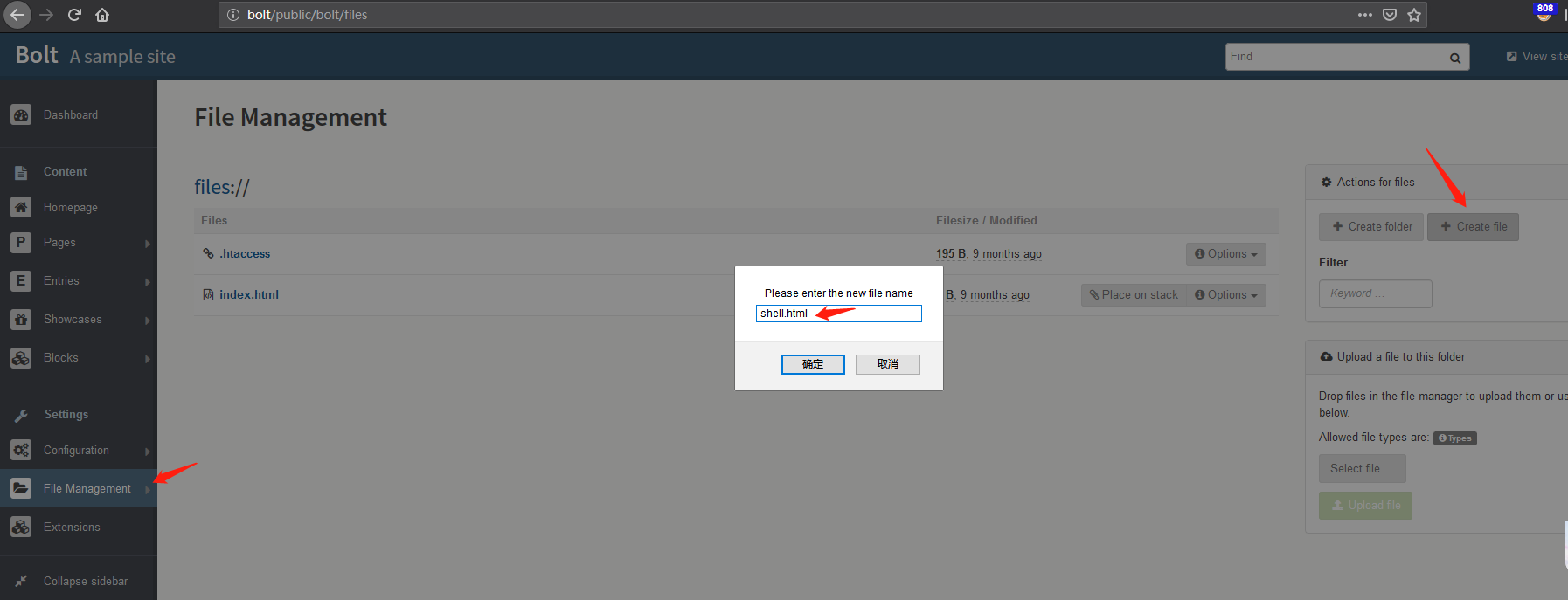
contenttype=pages&title=title&slug=testpage1&teaser=teaser1&body=<script>alert(1)</script>&id=151



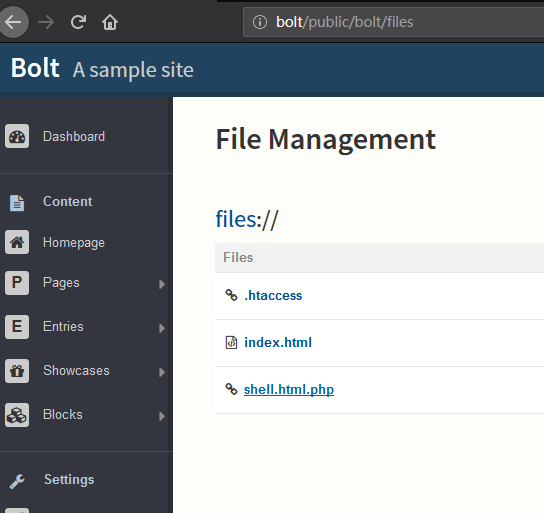
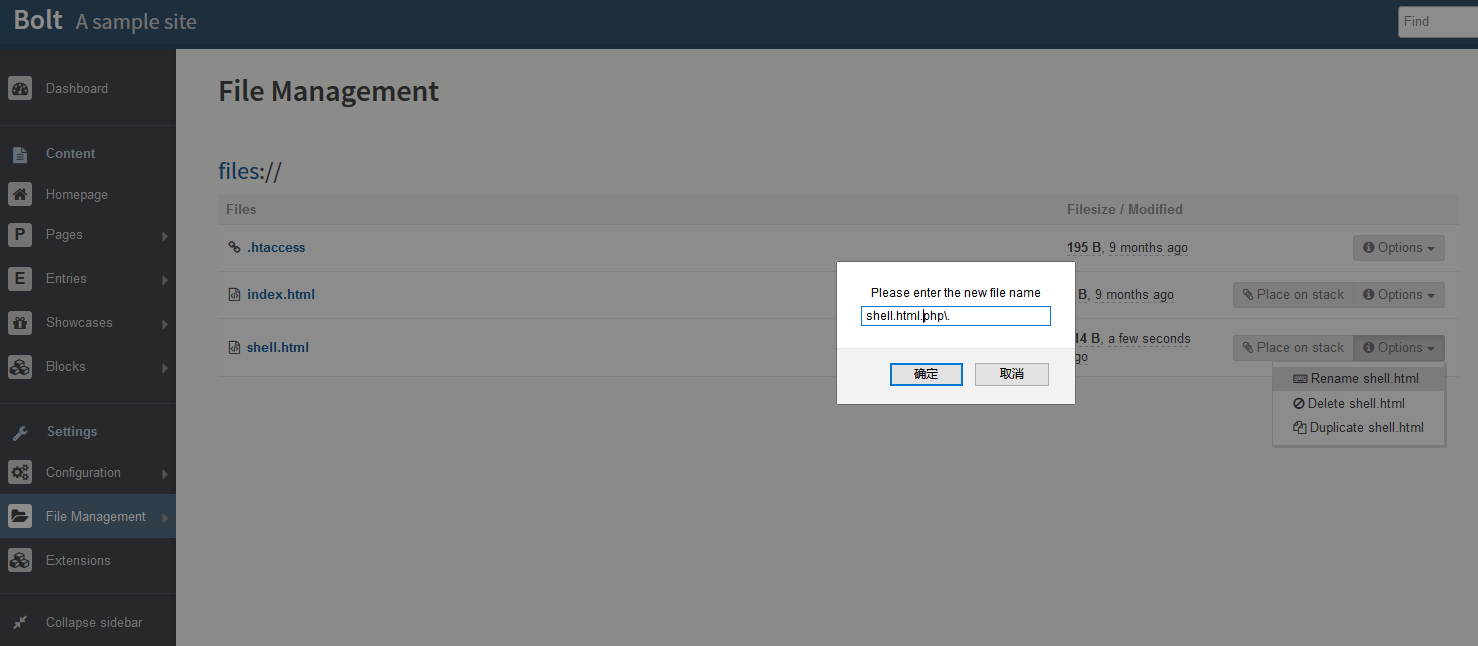


## 2.远程代码执行

创建一个文件，然后编辑这个文件，写入木马保存。



  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



# 四、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. 远程代码执行

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。