补档THM-Command Injection(命令注入漏洞)-学习

本文相关的TryHackMe实验房间链接： https://tryhackme.com/room/oscommandinjection

通过学习相关知识点：了解允许你通过易受攻击的应用程序执行命令的漏洞及其补救措施。

H2 什么是命令注入?

在本文中，我们将讨论一类Web漏洞--命令注入漏洞，我们需要了解这个漏洞是什么，并会展示它的影响以及它给web应用程序带来的一些风险。

然后，你还能够将这些知识付诸实践，即：

- 如何发现命令注入漏洞
- 如何为不同操作系统构建payload进行测试以及利用命令注入漏洞
- 如何在web应用程序中防护命令注入攻击
- 最后，将相关理论应用于练习中。

首先，让我们先了解什么是命令注入。 命令注入是滥用web应用程序的行为从而实现在操作系统上执行命令，在执行系统命令时将获得与OS设备上的web应用程序运行时相同的权限。 例如，在以名为 joe 的用户身份运行的 Web 服务器上实现命令注入将在对应的操作系统用户 joe 权限下执行命令——从而获得用户 joe 所拥有的任何系统权限。

命令注入漏洞也可被称为“远程代码执行”(RCE)，因为攻击者通过命令注入漏洞 可以尝试诱使web应用程序执行他们所提供的一系列有效载荷（在web应用程序中远程执行代码），而无需直接访问目标机器本身（例如：通过交互式 shell的方式）。Web 服务器将在运行该web应用程序的用户的权限下以及相关访问控制级别下 处理并执行 攻击者所提供的payload代码。

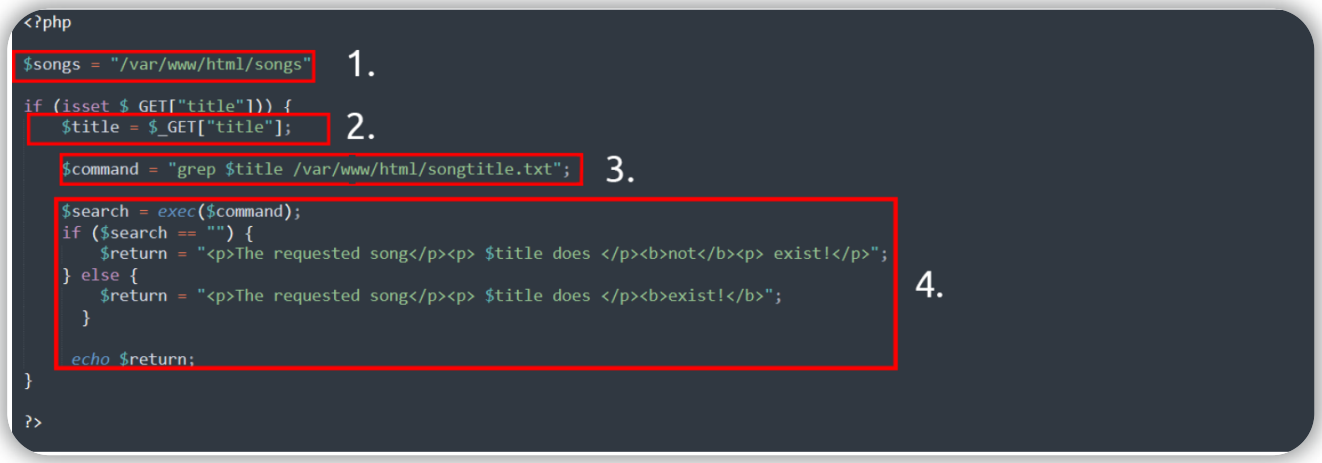
RCE意味着攻击者可以直接与易受攻击的目标操作系统进行交互，因此攻击者可能会尝试读取目标系统或用户的文件、数据以及类似的东西。例如，滥用web应用程序以执行命令 `whoami` 以列出应用程序正在运行的用户帐户信息就是利用命令注入漏洞的一个例子。

命令注入是 2019 年 Contrast Security 的 AppSec 情报报告中的十大漏洞之一（ Contrast Security AppSec.，2019 ）；此外，OWASP 框架也在不断将这种性质的漏洞列为 Web 应用程序的十大漏洞之一（ OWASP 框架 ）。

H2 发现命令注入漏洞

此漏洞的存在是因为应用程序经常使用编程语言（如 PHP、Python 和 NodeJS）中的函数来向计算机操作系统传递数据并在操作系统上进行系统调用操作。 例如，从字段中获取输入信息并在系统文件中搜索相关条目。 以下面的代码片段为例：

在此代码片段中，web应用程序会获取用户在名为 `$title` 的输入字段中所输入的数据，然后在操作系统上的文件目录中搜索对应的歌曲名称，我们可以将其分解为几个简单的步骤。



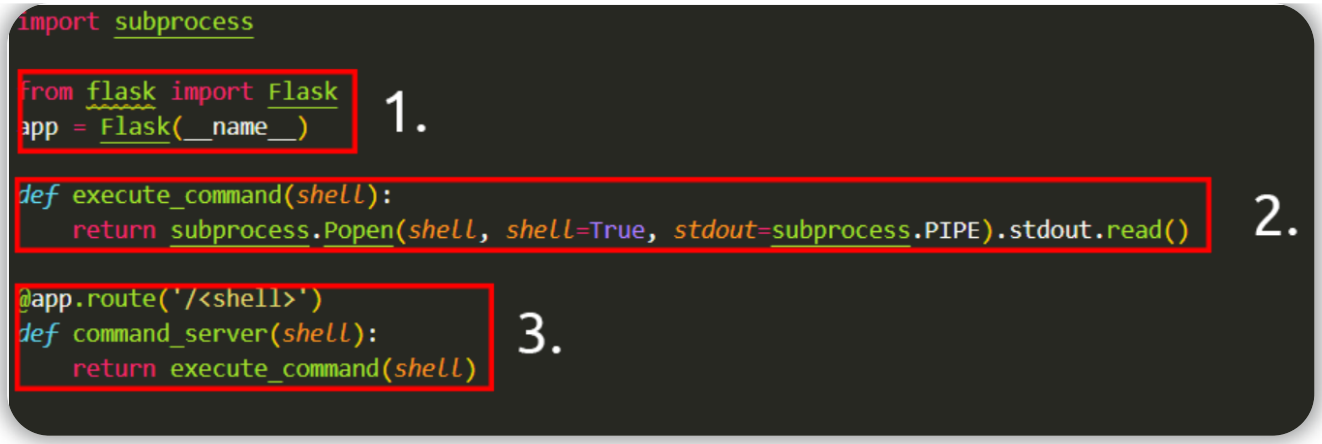

```php
<?php
$songs = "/var/www/html/songs"
if (isset $ GET["title"])) {
    $title = $_GET["title"];
    $command = "grep $title /var/www/html/songtitle.txt";
    $search = exec($command);
    if ($search == "") {
        $return = "<p>The requested song</p><p> $title does </p><b>not</b><p> exist!</p>";
    } else {
        $return = "<p>The requested song</p><p> $title does </p><b>exist!</b>";
      }
     echo $return;
}
?>
```

1. 该web应用程序将 MP3 文件存储在指定的操作系统目录下。
2. 用户输入他们想要搜索的歌曲名称，web应用程序将此输入存储到 `$title` 变量中。
3. 此 `$title` 变量中的数据被传递给命令 `grep` --从名为*songtitle.txt*的文本文件中搜索以找到用户希望搜索的歌曲条目。
4. 从*songtitle.txt* 文件中搜索得到的最终查找结果将决定该web应用程序通知用户目标歌曲存在与否。

在现实环境下，以上文件通常会存储在数据库中；所以上图代码片段只是应用程序从用户处获取输入信息并 以此与应用程序所对应的操作系统发生交互的一个示例。

攻击者可以通过注入他们自己选择的命令并让应用程序执行该命令 来滥用此web应用程序，攻击者可以要求web应用程序从一些敏感的文件中读取数据，而不是简单地使用 `grep` 来搜索 songtitle.txt 中的条目。

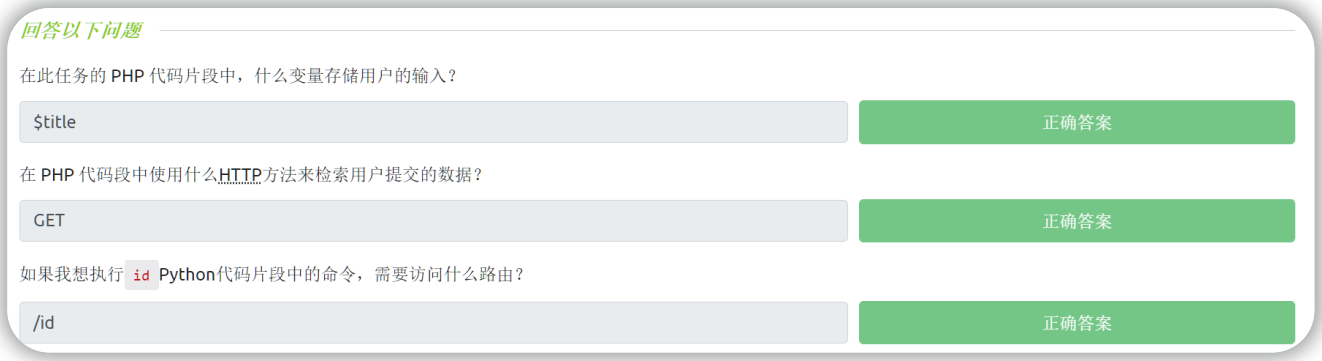
无论web应用程序使用何种编程语言，都可能以这种方式滥用应用程序，只要web应用程序处理并执行攻击者所构造的payload，就会导致命令注入。 例如，下面的代码片段是一个用 Python 编写的应用程序。



请注意，你不需要了解这些应用程序背后的语法，此处适当概述了这个 Python 应用程序的工作步骤。

1. "*flask*"包用于设置网络服务器
2. 定义函数，该函数使用"*subprocess*"包在设备上执行命令
3. 在web服务器中使用一个路由来执行所提供的任何内容。 例如，如果要执行 `whoami` 命令，我们需要访问 `http://flaskapp.thm/whoami`

答题



H2 利用命令注入漏洞

你通常可以通过应用程序的行为来确定是否能够发生命令注入攻击。

使用用户输入的数据来填充系统命令的应用程序，通常可以和意想不到的应用程序行为进行结合。例如，shell 运算符 `;`、 `&` 和 `&&` 将组合两个（或多个）系统命令并同时执行它们。

命令注入主要可以通过以下两种方式进行检测：

1. 盲注类型的命令注入（无回显）
2. 详细的命令注入（有回显）

下面中定义了这两种方法，第四、五小节将更详细地解释这些方法。

Blind（盲注类型） 这种类型的命令注入 在你测试 payload 时应用程序并没有直接输出结果;因此你将必须调查应用程序的行为以确定你的 payload 是否成功执行。

Verbose（详细类型） 这种类型的命令注入，在你测试 payload 时能够从应用程序获得直接反馈；例如，当你运行 whoami 命令以查看应用程序在哪个用户下运行时，Web 应用程序将直接在页面上输出具体的用户名信息。

检测盲注类型的命令注入（无回显）

盲注类型的命令注入在发生命令注入攻击时，并没有可见的输出显示，因此不会立即引起注意。 例如，当你执行了一条命令，但 Web 应用程序并没有输出任何消息。

对于这种类型的命令注入，我们需要使用会导致一些时间延迟的payload；例如，ping 和 sleep 命令。以ping命令 为例，根据你指定的 ping 次数，应用程序将挂起 x 秒。

另一种检测盲注类型-命令注入的方法是强制输出一些结果，这可以通过使用诸如 `>` 之类的重定向运算符来完成，如果你对此不熟悉，建议查看 Linux 基础模块知识点。 例如，我们可以告诉 Web 应用程序执行诸如 `whoami` 之类的命令并将其重定向到一个新建文件中；然后我们可以使用诸如 `cat` 之类的命令来读取这个新创建的文件的内容。

测试命令注入通常很复杂，可能需要经过多次尝试，尤其是 Linux 和 Windows 两种操作系统所使用的命令语法还存在不同之处。

使用 `curl` 命令也是测试命令注入的好方法，这是因为你可以使用 `curl` 命令在payload中向应用程序传递数据。 以下面的代码片段为例，通过一个简单的 curl payload 传递数据到web应用程序就可以进行命令注入攻击。

```
curl http://vulnerable.app/process.php%3Fsearch%3DThe%20Beatles%3B%20whoami
#curl http://vulnerable.app/process.php?search=The Beatles; whoami
```

检测详细类型的命令注入（有回显）

检测此种类型的命令注入通常可以用很简单的方法，详细类型的命令注入是指应用程序会向我们提供有关正在发生或正在执行的payload的反馈或输出。

例如，包含 `ping` 或 `whoami` 等命令的payload的执行结果会直接显示在 Web 应用程序的页面上。

有用的payload

以下是Linux 和 Windows两种操作系统中的一些有价值的有效载荷（payload）。

Linux

`whoami`	查看应用程序在哪个用户下运行。
`ls`	列出当前目录下的内容，你可能会找到配置文件、环境文件（令牌和应用程序密钥）等文件。
`ping`	此命令将调用`web`应用程序并挂起，这将有助于测试`web`应用程序是否存在盲注类型的命令注入漏洞。
`sleep`	这是测试`web`应用程序是否存在盲注类型的命令注入漏洞的另一个有用的`payload`。
`nc`	`Netcat`可用于在易受攻击的`web`应用程序上生成反向`shell`，你可以使用此立足点围绕目标机进行导航以获取其他服务、文件或潜在的提权方法。

Windows

`whoami`	查看应用程序在哪个用户下运行。
`dir`	列出当前目录下的内容，你可能会找到配置文件、环境文件（令牌和应用程序密钥）等文件。
`ping`	此命令将调用`web`应用程序并挂起，这将有助于测试`web`应用程序是否存在盲注类型的命令注入漏洞。
`timeout`	此命令也能调用`web`应用程序并挂起，如果未安装 `ping` 命令，它对于测试 `web`应用程序 是否存在盲注类型的命令注入漏洞也很有用。

答题

回答以下问题

如果我想确定应用程序以哪个用户身份运行，我会使用什么负载？

whoami — 正确答案

我会使用什么流行的网络工具来测试Linux机器上的盲命令注入？

ping — 正确答案

我将使用什么有效载荷来测试Windows机器的盲命令注入？

timeout — 正确答案

H2 防护命令注入攻击

可以通过多种方式防止命令注入，从尽可能少地使用编程语言中具有潜在危险的函数或库，到不依赖用户输入并过滤用户输入；下面的示例是在 PHP 编程语言环境下，但是，相同的原则也可以扩展到其他编程语言环境。

减少使用易受攻击的函数

在 PHP 中，有许多函数可以与操作系统发生交互，并通过 shell 执行命令，包括：

- Exec--执行一个外部程序
- Passthru--执行外部程序并且显示原始输出
- System--执行外部程序并且显示输出

以下面的代码片段为例，在此处，web应用程序将只接受和处理输入到表单中的数字，这意味着不会处理任何输入的命令，例如 whoami。

```
<input type="text" id="ping" name="ping" pattern="[0-9]+"></input> 1.
<?php
echo passthru("/bin/ping -c 4 "$_GET["ping"].'); 2.

?>
```

1. 该web应用程序将只接受特定的字符格式（数字 0-9）
2. 然后该web应用程序将继续执行这个全是数字的数据。

在默认情况下，类似的函数会接受字符串内容或者用户的数据输入，并将执行 在目标系统上被提供的任何内容，因此任何未经适当检查而使用这些函数的web应用程序都容易受到命令注入攻击的影响。

对输入数据进行净化（sanitisation）

清理web应用程序所使用的用户输入是防止命令注入的好方法。 这是指定用户可以提交的数据格式或类型的过程。 例如，可以指定仅接受数字数据或删除任何特殊字符（如 >、& 和 /）的输入字段。

在下面的代码片段中，PHP 函数 filter_input 用于检查通过输入表单提交的任何数据是否为数字，如果不是数字，则视为无效输入。

```php
<?php

if (!filter_input(INPUT_GET, "number", FILTER_VALIDATE_NUMBER)) {

}
```

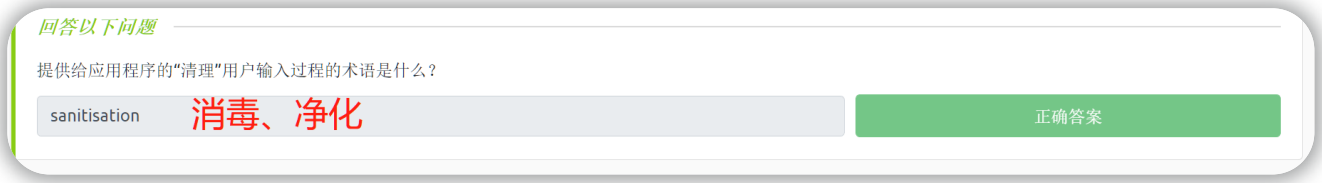
绕过过滤器

web应用程序可以采用多种技术来过滤和清理从用户输入中获取的数据，这些过滤器将限制你使用特定的有效载荷； 但是，我们可以滥用web应用程序背后的逻辑来绕过这些过滤器。

例如，web应用程序可能会去掉我们所使用的payload中的引号，我们此时可以尝试改用引号的十六进制值来替代payload中的引号；虽然最终使用的payload数据格式与预期的不同，但仍然可以得到该payload成功执行的对应结果。

```
$payload = "\x2f\x65\x74\x63\x2f\x70\x61\x73\x73\x77\x64"
```

答题

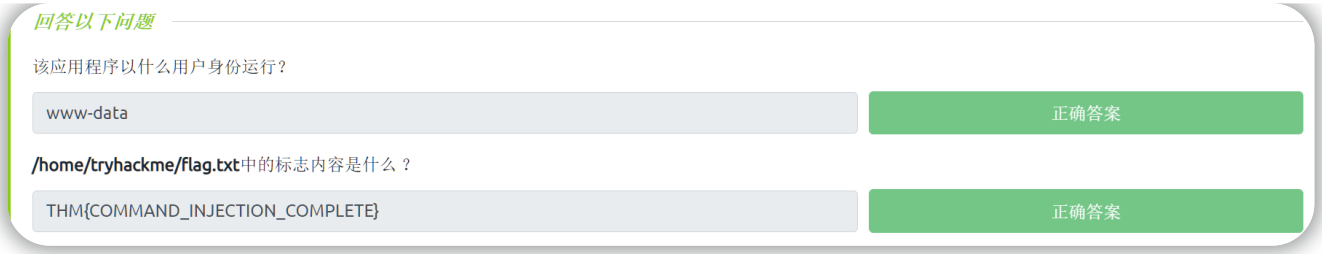


H2 命令注入漏洞-练习

启动目标机器，在分屏中可见的网站上所托管的web应用程序上测试一些有效载荷，以测试命令注入漏洞。 如果你遇到困难或者希望探索一些更复杂的有效载荷，请参阅此备忘单： https://github.com/payloadbox/command-injection-payload-list

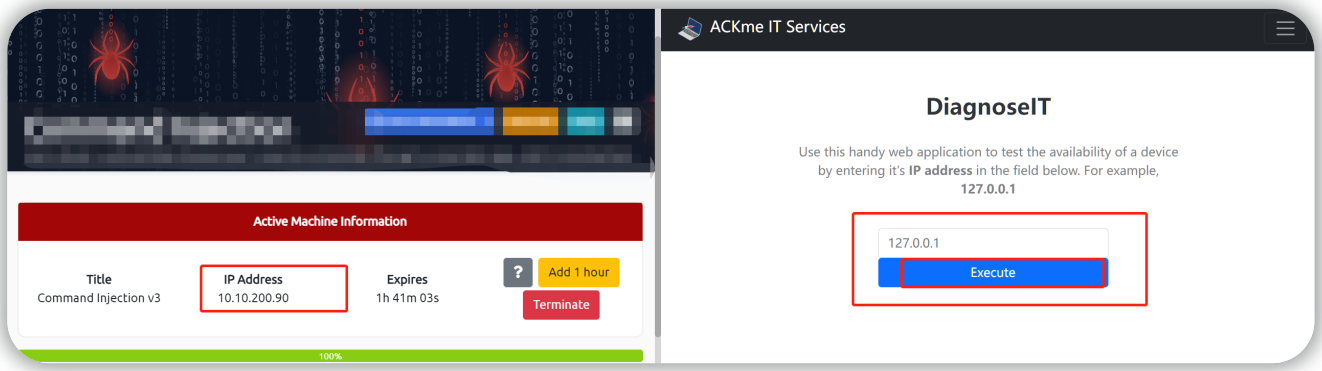
在 `/home/tryhackme/flag.txt` 中找到flag的内容，你可以使用有效载荷来实现这一点——建议尝试多种有效载荷。

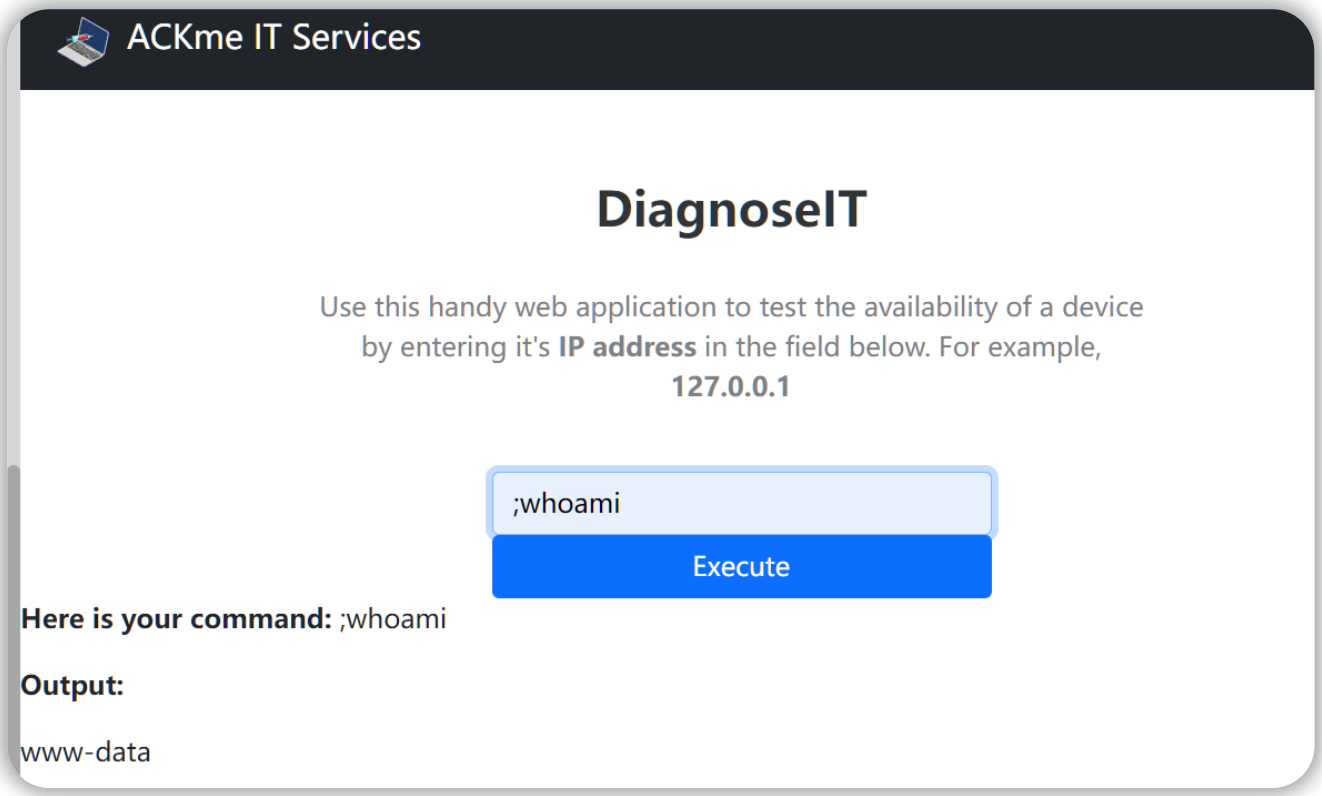
答题



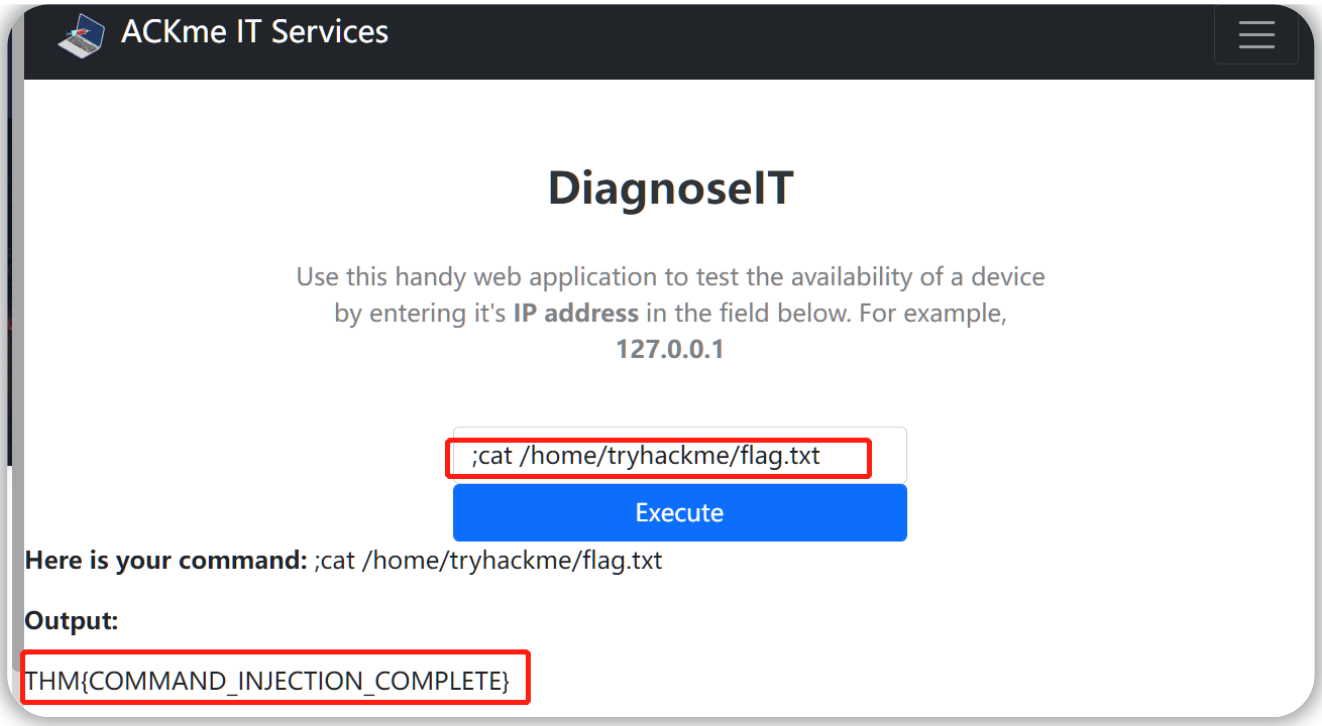
启动目标机器，并查看分屏中的web应用程序：

通过在下面的字段中输入设备的 IP 地址，使用这个简易的 Web 应用程序来测试目标设备的可用性。 例如，127.0.0.1





> www-data



THM{COMMAND_INJECTION_COMPLETE}