种族状况脆弱性实验室

1 实验室概况

这个实验室的学习目标是让学生通过把他们从课堂上学到的关于脆弱性的知识转化为行动来获得关于种族条件脆弱性的第一手经验。 当多个进程同时访问和操作相同的数据时，会发生一个竞赛条件，执行的结果取决于访问的特定顺序。 如果特权程序具有竞争条件漏洞，攻击者可以运行一个并行进程来“竞争”特权程序，目的是改变程序的行为。

在这个实验室中，学生将得到一个具有种族条件漏洞的程序；他们的任务是开发一个利用漏洞并获得根特权的方案。除了攻击，学生将被引导通过几个保护方案，可以用来对抗种族条件攻击。学生需要评估这些方案是否有效，并解释原因。

2 实验室任务

2.1 初始设置

您可以使用我们预先构建的 Ubuntu 虚拟机执行实验室任务。如果您正在使用我们的 Ubuntu9.11VM，您可以跳过这个初始设置步骤。如果您正在使用我们的 Ubuntu11.04 或 12.04VM，您需要阅读以下内容。Ubuntu11.04 和 12.04 提供了一个内置的防止种族条件攻击的保护。 这个方案通过限制谁可以跟随系统链接来工作。 根据文档，“世界可写粘性目录中的符号链接”(例如。 如果跟随者和目录所有者与 symlink 所有者不匹配，则无法跟踪。 在这个实验室，我们需要禁用这个保护。 您可以使用以下命令实现这一点：

```
$sudosysctl-w kernel.yama.protected_sticky_symlinks=0
```

2.2 一项弱势方案

下面的程序是一个看似无害的程序。 它包含一个种族条件漏洞。

```
/*vulp.c*/

#包括<>
#包括<unistd.h>


主要()

{
    char*fn=“/tmp/XYZ”；
    字符缓冲[60]；
```

```
    文件*fp；

    /*获取用户输入*/scan f(“%50s”，缓冲区)；

    如果(！访问(FN，W_OK){

        fp=fopen(fn，“a+”)；
        写(“\n”，size of(char)，1，fp)；写(缓冲区，size of(char)，strlen（缓冲区），fp)；fclose(fp)；
    }
    否则打印 f(“无权限\n”)；
}
```

这是 Set-UID 程序的一部分(由 root 拥有)；它将一串用户输入附加到临时文件/tmp/XYZ 的末尾。 由于代码使用根特权运行，它仔细检查真正的用户是否实际拥有对文件/tmp/XYZ 的访问权限；这是访问()调用的目的。 一旦程序确保真正的用户确实有权，程序将打开文件并将用户输入写入文件。

看来程序在第一次查看时没有任何问题。但是，在这个程序中存在一个 RACE 条件漏洞：由于检查(Access)和使用(fopen)之间的窗口（模拟延迟），访问使用的文件可能与 fopen 使用的文件不同，尽管它们具有相同的文件名/tmp/XYZ。 如果恶意攻击者可以以某种方式使/tmp/XYZ 成为指向/etc/shadow 的符号链接，则攻击者可以将用户输入附加到/etc/shadow（请注意，程序使用根特权运行，因此可以覆盖任何文件）。

2.3 任务 1：开发种族条件脆弱性

您需要在上面的 Set-UID 程序中利用竞赛条件漏洞。 更具体地说，您需要实现以下内容：

1. 重写任何属于 root 的文件。
2. 获得根特权；也就是说，您应该能够做任何根可以做的事情。

2.4 任务 2：保护机制 A：重复

摆脱种族条件并不容易，因为检查和使用模式往往是必要的程序。而不是取消比赛条件，我们实际上可以增加更多的比赛条件，这样为了损害程序的安全性，攻击者需要赢得所有这些比赛条件。 如果这些比赛条件设计得当，我们可以成倍地降低攻击者的获胜概率。 基本思想是重复访问（)并打开(）几次；每次打开文件，最后通过检查它们的 i-node（它们应该是相同的）来检查是否打开了相同的文件..

请使用此策略修改易受攻击的程序，并重复您的攻击。 报告成功是多么困难，如果你还能成功的话。

2.5 任务 3：保护机制 B：最小特权原则

本实验室易受伤害程序的根本问题是违反最小特权原则。程序员确实理解运行程序的用户可能太强大，因此他/她引入了访问()来限制用户的能力。 然而，这不是适当的办法。 更好的方法是应用最小特权原则；也就是说，如果用户不需要某些特权，则需要禁用特权。

我们可以使用 seteuid 系统调用暂时禁用根特权，如果有必要，我们可以稍后启用它。 请使用此方法修复程序中的漏洞，然后重复您的攻击。 你能成功吗？ 请报告您的意见和解释。

2.6 任务 4：保护机制 C：Ubuntu 的内置方案

此任务仅适用于那些使用我们的 Ubuntu11.04 或 12.04VM 的人。正如我们在初始设置中提到的，Ubuntu11.04 和 12.04 提供了一个内置的保护方案，以防止种族条件攻击。 在此任务中，需要使用以下命令将保护打开：

$sudosysctl-w kernel.yama.protected_sticky_symlinks=1

在你的报告中，请描述你的观察。 还请解释以下几点：（1）为什么这种保护方案有效？（2）这是一个很好的保护吗？ 为什么或者为什么不？ （3）该计划有何局限性？

3 准则

3.1 两个潜在目标

在 vulp.c 中，可能有许多方法来利用种族条件脆弱性。 一种方法是使用漏洞将一些信息附加到/etc/passwd 和/etc/shadow。Unix 操作系统使用这两个文件对用户进行身份验证。如果攻击者可以将信息添加到这两个文件中，他们本质上有能力创建新用户，包括超级用户(通过让 uid 为零)。

/etc/passwd 文件是 Unix 机器的身份验证数据库。包含基本用户属性.. 这是一个 ASCII 文件，其中包含每个用户的条目。 每个条目定义应用于用户的基本属性。 当您使用 adduser 命令向系统添加用户时，该命令将更新/etc/passwdfile。

文件/etc/passwd 必须具有世界可读性，因为许多应用程序需要访问用户属性，如用户名、主目录等。 在该文件中保存加密密码意味着任何能够访问该机器的人都可以使用密码破解程序（如破解）来闯入他人的帐户。 为了解决这个问题，创建了影子密码系统。影子系统中的/etc/passwd 文件是世界可读的，但不包含加密密码。 另一个文件/etc/shadow 仅由 root 可读，它包含密码。

若要找出要添加到这两个文件中的字符串，请运行 adduser，并查看添加到这些文件中的内容。例如，下面是在创建一个名为 smith 的新用户之后添加到这些文件中的内容：

/etc/passwd：

史密斯：x：1000：1000：乔·史密斯，zzhome/smith：bin/bash

/etc/shadow：

smith：*1*Srdssdsdi*M4sdabPasdsdsdasdsdasdY/：13450：0：99999：7：：：

文件/etc/passwd 中的第三列表示用户的 UID。 因为 smith 账号是普通用户账号，所以其价值 1000 没有什么特别的.. 如果我们把这个条目改为 0，史密斯现在变成根。

3.2 创建符号链接

您可以调用 C 函数 symlink()在程序中创建符号链接。由于 Linux 不允许创建链接，如果链接已经存在，我们需要首先删除旧链接。 下面的 C 代码片段展示了如何删除链接，然后使/tmp/XYZ 指向/etc/passwd：

```
unlink("/tmp/XYZ");
symlink("/etc/passwd", "/tmp/XYZ");
```

您还可以使用 Linux 命令“ln-sf”创建符号链接。这里的“f”选项意味着，如果链接存在，请先删除旧链接。“ln”命令的实现实际上使用 unlink()和 symlink()。

3.3 提高成功率

RACE 条件攻击的最关键步骤（即指向目标文件的链接）必须发生在检查和使用之间的窗口内，即 vulp.c 中的访问和 fopen 调用之间。 由于我们不能修改易受攻击的程序，我们唯一能做的就是与目标程序并行运行我们的攻击程序，希望链接的更改确实发生在这个关键窗口内.. 不幸的是，我们无法达到完美的时机。 因此，攻击的成功是概率的。 如果窗口很小，成功攻击的概率可能很低。 您需要考虑如何增加概率（提示：您可以多次运行脆弱程序；您只需要在所有这些试验中获得一次成功）。

由于您需要多次运行攻击和脆弱程序，因此需要编写一个程序来自动化攻击过程。为了避免手动输入到 vulp，可以使用重定向。 也就是说，在文件中键入输入，然后在运行 vulp 时重定向此文件。 例如，您可以使用以下内容：外阴<档案。

3.4 知道攻击是否成功

由于用户没有访问/etc/shadow 的读取权限，因此无法知道是否修改了它。 唯一可能的方法是看它的时间戳。此外，如果我们停止攻击，一旦条目被添加到相应的文件将更好。 下面的 shell 脚本检查/etc/shadow 的时间戳是否已更改。 一旦注意到更改，它将打印一条消息。

```
#！/bin/sh

旧=‘ls-l/etc/阴影’
新=‘ls-l/etc/阴影’，而[“$旧”=“$新”]做
        新=‘ls-l/etc/阴影’
完成了
回声“停止... 阴影文件已更改“
```

3.5 不可取的情况

在测试攻击程序时，您可能会发现/tmp/XYZ 是以 root 为所有者创建的。如果发生这种情况，您已经失去了“种族”，即文件是由根创建的。一旦发生这种情况，就没有办法删除这个文件。这是因为/tmp 文件夹上有一个“粘性”位，这意味着只有文件的所有者才能删除该文件，即使该文件夹是世界可写的。

如果发生这种情况，您需要调整攻击策略，然后再试一次（当然，在手动从根帐户中删除文件之后）。发生这种情况的主要原因是，攻击程序在删除/tmp/XYZ 之后立即被关闭，但在它将名称链接到另一个文件之前。请记住，删除现有符号链接并创建新链接的操作不是原子的（它涉及两个单独的系统调用），因此如果上下文切换发生在中间(即在/tmp/XYZ 删除之后)，并且目标 Set-UID 程序有机会运行其 fopen(fn，“a+”)语句，它将创建一个以 root 为所有者的新文件。 想想一种策略，它可以最大限度地减少在该操作中间切换上下文的机会。

3.6 警告

在过去，一些学生在攻击过程中意外地清空了/etc/shadow 文件（我们仍然不知道是什么造成的）。 如果您丢失了阴影文件，您将无法再次登录。 为了避免这种麻烦，请制作原始阴影文件的副本。