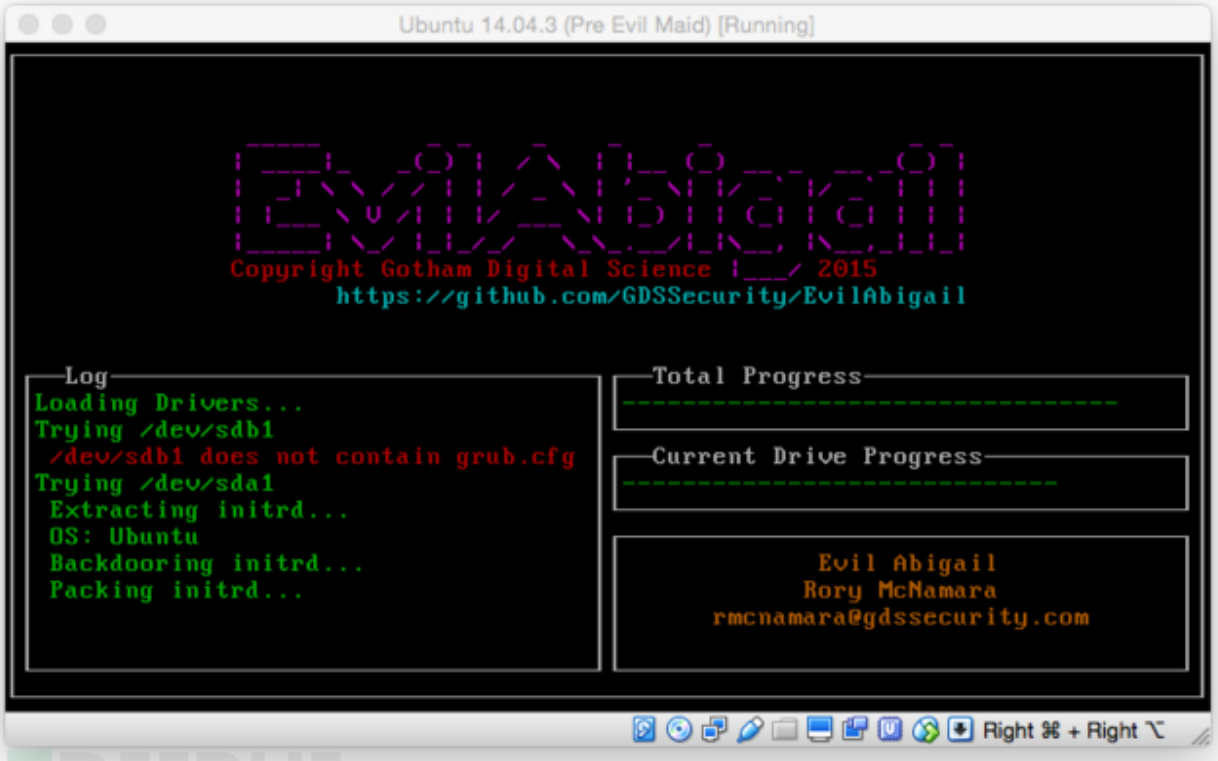

最近，我们高谭数码科技(GDS)的研究人员讨论了针对Linux系统全盘加密的冷启动攻击，大家都认为这种攻击是可行的，但执行这么一次攻击有多难？攻击的可行性有多少呢？

攻击

由于市面上没有现成的工具能够执行这种攻击，所以我们自己制作了工具，并把它命名为EvilAbigail。Evil maid攻击可以针对任何操作系统。此次的研究我们针对的是使用LUKS全盘加密的Linux系统。

一般来说，当Linux系统使用了全盘加密后，会由一小块分区还是没有加密，这个区域就是用来解密和引导加密磁盘的。这个分区会挂载在/boot，并且包含内核和初始RAM磁盘(initrd)。虽然攻击内核或者bootloader也是可行的，但是我们还是针对initrd进行了攻击。

initrd是指一个临时文件系统，它在启动阶段被Linux内核调用。initrd主要用于当root文件系统被挂载之前，进行准备工作。initrd 中包含了是解密和挂载root文件系统所需要的目录和可执行程序的最小集合。一旦initrd任务完成，它就会执行pivot_root，从而将initrd根文件系统卸载掉,并挂载真正的根文件系统。

一般来说，initrd是一个通过gzip压缩的cpio镜像。我们测试的基于Debian的操作系统是这样，但基于RedHat的操作系统 (Fedora, RHEL, CentOS)现在使用的是dracut，包含一个未压缩的cpio镜像。基于Debian的initrds 会用ash shell脚本执行启动，而dracut则会用systemd和它所关联的配置方法。

为了执行我们的攻击，我们选择使用一个基于LD_PRELOAD的bootkit，但是其实也可以注入恶意的内核或可执行文件中。我们使用LD_PRELOAD的主要目标是对刚刚解密完成的root文件系统中的第一个可执

行文件注入一个共享对象。第一个可执行文件通常是/sbin/init，PID一般会是1。进行攻击最简单的方法就是修改init脚本，导出这个环境变量，这样执行pivot_root的时候环境变量就设置好了。因为当文件系统更改的时候还得在合适的时候（解密之后）把共享对象复制到新系统中。把以下这两行放入initrd的init脚本中，插在切换文件系统之前：

```
cp  /hack.so  /${rootmnt}/hack.so
export  LD_PRELOAD=/hack.so
```

之所以这样可行是因为真正的root文件系统是在临时root文件系统下解密挂载的，这先于pivot，并且rootmnt变量是用挂载点位置填充的。但是，在这之前需要把目标文件系统重新挂载成读写，因为默认是只读的。在我们的例子中我们对init脚本进行了修改，修改了脚本分析内核命令行的地方，因此无论提供的参数是什么，root文件系统都是读写方式挂载。另一种方法是在注入的命令中添加mount -o remount,rw /${rootmnt}。

不过基于dracut的initrds中不存在注入点，因为init的可执行文件是个二进制文件而非shell脚本。这就给我们带来了三个问题，只有克服了这三个问题我们才能够注入到pid 1进程中。

三个问题

第一个问题是有关复制我们的二进制文件到解密的root文件系统中。这个问题是三个问题中最好解决的一个。我们可以加两个ExecPre指令到负责pivote文件系统的systemd服务文件中。这基本就相当于前面提到的插入脚本的方法。第一个命令会以读写方式重新挂载root文件系统，第二个执行复制操作。

第二个问题有关LD_PRELOAD。因为我们不是在修改shell脚本，我们不能把环境变量传递给这个进程（因为它是由内核调用的），因此加载我们的共享对象就有点棘手了。最简单的办法就是，先把init二进制文件移动到另一个位置，然后在它原来的位置插入我们自己的shell脚本，最后再执行原来的二进制文件。我们只需要两行代码，第一行导出LD_PRELOAD，第二行执行原来的systemd二进制文件。请注意，这样注入的是initrd中的pid 1进程，而不是最终root文件系统的pid 1。

第三个问题就是，在调用switch-root命令之前，systemd会用clearenv()函数清除所有环境变量。因为这个函数是标准库的一部分，我们就可以重写这个函数，让被注入的进程会调用我们的函数而不是原来的函数。我们不关心真正清除环境变量，我们写的clearenv()函数会清除所有环境变量，然后把我们的LD_PRELOAD变量注入到环境中。由于clearenv()只会被调用一次，我们的修改不会导致任何副作用。

解决了以上这三个问题之后，我们的共享对新就会被复制到加密的root文件系统中，我们的LD_PRELOAD会被注入到目标文件系统的pid 1进程中。接下来我们就可以获取用于解密的用户密码。

对于Debian的initrds，可执行文件会要求用户输入密码，然后解密、挂载root文件系统。我们可以把我们的脚本注入到pipeline进程中从而获取密码。

至于systemd，它会通过Unix domain sockets使用一种更复杂的进程间通信。我们选择攻击文件系统的挂载，而非这个进程。这又是一个库函数，它在动态加载库中，从systemd里调用，所以我们可以hook这个函数。解密硬盘的函数叫做crypt_activate_by_passphrase。这个函数会把密码作为char数组。通过hook这个函数，我们可以获取到密码。我们要包裹这个原来的函数，所以我们用dlsym打开真正的函数，并且调用它。不过在此之前我们会保存密码以便以后取回。

为了“保存”密码，我们简单地把密码加到我们之后要复制到root文件系统的共享对象中。之所以选择这种方法是因为我们已经知道这个文件存在，并且会被复制过去。采用这种方法还会减少我们接触到的磁盘文件的数量。为了获取密码，我们之后会读取我们自己文件末尾（通过LD_PRELOAD变量定位），然后把它设置为我们的反弹shell中PASSWORD环境变量的值，因此（以meterpreter为例）通过‘getenv PASSWORD’命令可以获取用于解密磁盘的密码。由于我们所有的目标主机都默认安装了python，所以我们就使用了python meterpreter反弹shell。

解决方案

解决这种问题有很多方法。但是即使使用了这些解决方法，如果攻击者能够物理接触计算机，并且有足够的时间重刷BIOS/UEFI，那也是防不住的。

第一种方法是把bootloader、内核和initrd放在外置的U盘上，然后从U盘上启动从而替代/boot分区。但对用户来说这个方法很糟糕，因为他们离开笔记本的时候要记得拔掉U盘，如果没有卸载的话还要安全卸载/boot分区。更新的时候也很麻烦，要插上U盘才能更新initrd/内核。

另一种方案则是彻底关闭从外置媒体启动系统。这样就不存在自动化攻击的可能性了，但是某些情况下，如对于包含shell的Debian initrds，还是可以从initrd人工挂载和修改initrd的。这可以通过自动键盘式设备完成，这样就绕过了无法通过外置媒体启动系统的限制。

另外，还可以开启BIOS启动密码，这样没有密码的人就无法启动计算机了。

不过如果攻击者有足够的时间把硬盘拆下，然后用他们自己的笔记本启动硬盘的话，后两种方案也不管用了。

最后，最安全的方案就是将SecureBoot拓展到initrd。SecureBoot可以验证bootloader和内核，如果能够验证经过签名的initrd的话，要不留痕迹地在/boot分区修改任何东西都会很困难。但是如果攻击者可以通过刷BIOS/UEFI关闭secureboot的话，这种方案也没用了。

防止这类攻击最好的方法就是不要让你的设备落入攻击者们的手中，我们的PoC攻击可以在2分钟内攻陷所有的目标主机，但在现实世界中，攻击者也可以做出攻击特定目标的payload，这样的payload只用几秒就可以攻击设备。

相关代码和有关我们工具的更多细节：

https://github.com/GDSSecurity/EvilAbigail

*** 参考来源GDS Blog，vulture翻译，文章有修改，转载请注明来自FreeBuf黑客与极客（FreeBuf.COM）**