Zip Slip漏洞综述

angel010 / 2018-06-11 01:38:23 / 浏览数 6025 技术文章 技术文章 顶(1) 踩(0)

Zip Slip是一个广泛存在的关键存档提取（critical archive
extraction）漏洞，该漏洞允许攻击者在系统中任意写文件，尤其是会导致远程命令执行。Snyk安全团队的研究人员6月5日发现并公布了该漏洞的细节，该漏洞影响上千个

其实许多的生态系统中都存在该漏洞，包括JS、Ruby、.net和Go。但该漏洞在Java环境中尤为流行，因为java环境中没有对存档文件进行高效处理的中心库函数。缺乏这样

该漏洞可以用伪造的含有目录遍历文件名（e.g. ../../evil.sh）的存档进行利用。Zip slip漏洞会影响许多的存档格式，包括tar, jar, war, cpio, apk, rar, 7z等。

Zip
Slip是一种可以从存档中提取文件的目录遍历漏洞利用形式。目录遍历漏洞的基础是攻击者可以获取目标文件夹外的文件系统的部分访问权限。然后攻击者可以覆写可执行文

利用应用流

利用该漏洞的两个必要条件是：
1）有恶意顶存档文件；
2）提取代码不会执行验证性检查。

首先，在提取时，zip文件的内容需要有一到多个会打破目标目录的文件。在下面的例子中，我们可以看到zip文件的内容。一共有2个文件，good.sh文件和evil.sh文件。go
..命令，会发现仍然在root目录中，所以在遍历到敏感文件前，恶意路径会含有许多层的../来有可能到达root目录。

```
5 Tue Jun 5 11:04:29 BST 2018 good.sh
20 Tue Jun 5 11:04:42 BST 2018 ../../../../../../../../tmp/evil.sh
```

该zip文件的内容被手动伪造的。虽然zip允许用户向这些路径中增加文件，但存档创建工具是不允许的。但有了好的工具，很容易就可以用这些路径创建文件。
漏洞利用的第二个条件是有从存档中提取文件的功能，这里既可以用自己的代码也开始用库函数。该漏洞存在于提取代码不验证存档中的文件路径有效性时。含有漏洞的代

```
1:   Enumeration<ZipEntry> entries = zip.getEntries();
2:   while (entries.hasMoreElements()) {
3:       ZipEntry e = entries.nextElement();
4:       File f = new File(destinationDir, e.getName());
5:       InputStream input = zip.getInputStream(e);
6:       IOUtils.copy(input, write(f));
7:   }
```

第4行e.getName()是目标目录dir连接在一起，没有进行有效性验证。此处，zip存档到达evil.sh，会将zip的入口处的全路径加到目标目录中，这会导致evil.sh写入目标目录

你会受到漏洞影响吗？

如果你使用的库存在zip slip漏洞或工程中含有有漏洞的代码，都会在不验证目录遍历的情况下从存档中提取文件。
Synk维护了一个github存档库，列出了所有有zip slip漏洞的工程：
https://github.com/snyk/zip-slip-vulnerability

采取什么措施?

下面是检查工程中是否含有zip slip漏洞的代码的步骤：

1. 搜索项目中是否存在有漏洞的工程

Java

Java生态系统不提供含有存档文件高级处理的中心库。流行的Oracle和Apache commons-compress
API被用来提供一些存档支持，但不公开提供完全提取的能力。研究人员发现Java生态系统比其他生态系统的存档库函数更多，而且许多库都是有漏洞的。
漏洞代码示例：

```
1:   Enumeration<ZipEntry> entries = zip.getEntries();
2:   while (entries.hasMoreElements()) {
3:       ZipEntry e = entries.nextElement();
4:       File f = new File(destinationDir, e.getName());
5:       InputStream input = zip.getInputStream(e);
6:       IOUtils.copy(input, write(f));
7:   }
```

验证代码示例:

```
1:   String canonicalDestinationDirPath = destinationDir.getCanonicalPath();
2:   File destinationfile = new File(destinationDir, e.getName());
3:   String canonicalDestinationFile = destinationfile.getCanonicalPath();
4:   if (!canonicalDestinationFile.startsWith(canonicalDestinationDirPath)) {
5:       throw new ArchiverException("Entry is outside of the target dir: " + e.getName());
6:   }
```

Groovy

与Java类似，Groovy在不同的工程代码库中也有有漏洞的代码段，以为使用了有漏洞的Java存档处理库。
漏洞代码示例：

```
1:   final zipInput = new ZipInputStream(new FileInputStream(self))
2:   zipInput.withStream {
3:    def entry
4:    while(entry = zipInput.nextEntry)  {
5:      final file = new File(dest, entry.name)
6:      file.parentFile?.mkdirs()
7:      def output = new FileOutputStream(file)
8:      output.withStream {
9:        output << zipInput
10:     }
11:     unzippedFiles << file
12:   }
13: }
```

验证代码示例:

```
1:   final canonicalDestinationDirPath = destinationDir.getCanonicalPath()
2:   final destinationfile = new File(destinationDir, e.name)
3:   final canonicalDestinationFile = destinationfile.getCanonicalPath()
4:   if (!canonicalDestinationFile.startsWith(canonicalDestinationDirPath)) {
5:       throw new ArchiverException("Entry is outside of the target dir: ${e.name}")
6:   }
```

JavaScript

JavaScript有很多的中心库，能够从存档中提取文件，研究人员发现的有漏洞的库已经修复了。
要说明的是join命令将两个路径参数结合在一起，并返回解析后最短的路径。
漏洞代码示例：

```
1:   self.on('entry', function(entry) {
2:     entry.pipe(Writer({
3:       path: path.join(opts.path,entry.path)
4:   }))
```

验证代码示例:

```
1:   var filePath = path.join(targetFolder, entry.path);
2:   if (filePath.indexOf(targetFolder) != 0) {
3:     return;
4:   }
```

.Net

.Net生态系统也有中心库函数来执行提取功能。事实上，核心.Net库中的代码会检查Zip slip漏洞。
漏洞代码示例：

```
1:   public static void WriteToDirectory(IArchiveEntry entry,
2:                                        string destDirectory,
3:                                        ExtractionOptions options){
4:     string file = Path.GetFileName(entry.Key);
5:     string destFileName = Path.Combine(destDirectory, file);
6:     entry.WriteToFile(destFileName, options);
7:   }
```

验证代码示例:

```
1:   destFileName = Path.GetFullPath(Path.Combine(destDirecory, entry.Key));
2:   string fullDestDirPath = Path.GetFullPath(destDirectory);
3:   if (!destFileName.StartsWith(fullDestDirPath)) {
4:      throw new ExtractionException("Entry is outside of the target dir: " + destFileName);
5:   }
```

Go

Go生态系统只有一个有漏洞的库，并且在研究人员公布该问题的2天内就修复了。要说明的是join命令将两个路径参数结合在一起，并返回解析后最短的路径。
漏洞代码示例：

```
1:   func (rarFormat) Read(input io.Reader, dest string) {
2:     rr := rardecode.NewReader(input, "")
3:     for {
4:       header := rr.Next()
5:       writeNewFile(filepath.Join(dest, header.Name), rr, header.Mode())
6:     }
7:   }
```

验证代码示例:

```
1:   func sanitizeExtractPath(filePath string, destination string) error {
2:      destpath := filepath.Join(destination, filePath)
3:      if !strings.HasPrefix(destpath, destination) {
4:        return fmt.Errorf("%s: illegal file path", filePath)
5:     }
6:     return nil
7:   }
```

Ruby & Python

研究人员没有在Ruby和Python生态中找到有漏洞的代码段和库函数。事实上，python的zipfile是有漏洞的，但在2014年就修复了。Ruby也存在各种各样的漏洞，也在之前
因为缺乏高级的提取API，所有有很多库没有正确的使用，也会造成一些漏洞的产生。

2. 在应用build pipeline中添加Zip Slip安全测试

如果不通过直接或者递归依赖来搜索是否存在有漏洞的库，那么可以选择snyk这样的依赖性漏洞扫描工具。在开发生命周期中加入安全测试也是一个好的实践，比如在开发、
slip漏洞。

其他有漏洞的工程

因为不同生态使用不同的库，所以很多工程其实都是存在漏洞的。其中，上千个工程含有系统的漏洞代码样本或漏洞库函数，最主要的有Oracle, Amazon, Spring/Pivotal, Linkedin, Twitter, Alibaba, Jenkinsci, Eclipse, OWASP, SonarCube, OpenTable, Arduino, ElasticSearch, Selenium, Gradle, JetBrains等。

https://res.cloudinary.com/snyk/image/upload/v1528192501/zip-slip-vulnerability/technical-whitepaper.pdf

上一篇：【译】Metasploit：如何使... 下一篇：利用CVE-2017-8890实现...

1. 0 条回复
   - 动动手指，沙发就是你的了！