CVE监控之Python代码实现

浮萍 / 2018-01-05 16:56:00 / 浏览数 4040 技术文章 企业安全 顶(0) 踩(0)

0x01 前言

前几天在先知上看到伪全栈式安全研发：CVE监控这篇文章，就想着也实现一下代码进行最新CVE的监控。语言采用了Python，数据库也为Mongodb数据库。代码和实现的

主要包括以下几个方面。

1. 获取最新的CVE列表和详情
   主要采用了python的requests模块和BeautifulSoup模块。
2. 将最新的CVE信息存入数据库
   数据库使用了Mongodb，采用了pymongo模块。
3. 通过邮件发送最新的CVE信息
   发送邮件采用了smtplib模块。
4. 定时执行任务
   使用了linux的crontab来实现。

0x02 实现过程

1. 获取最新的CVE列表和详情

访问https://cassandra.cerias.purdue.edu/CVE_changes/today.html ，可以获取每天新增的CVE信息。

通过查看源代码，发现没html没什么规律可言，都是些超链接。要想获取最新的列表，可以通过取文本中间的方法来获取。
这里需要获取`New entries:`和`Graduations`之间的内容。然后通过BeautifulSoup来解析其中的超链接。
主要代码如下：

```
def getCVES():# ■■■■■CVE■■
   try:
       url = 'https://cassandra.cerias.purdue.edu/CVE_changes/today.html'
       res = requests.get(url, headers=headers, timeout=60)
       CVEList_html = getMiddleStr(res.text, 'New entries:', 'Graduations')
       soup = BeautifulSoup(CVEList_html, 'html.parser')
       for a in soup.find_all('a'):
           print(a['href'])
           print(a.string)
   except Exception as e:
       print(e)
```

获取文本中间内容的代码：

```
def getMiddleStr(content, startStr, endStr): # ■■■■■■■■
   startIndex = content.index(startStr)
   if startIndex >= 0:
       startIndex += len(startStr)
       endIndex = content.index(endStr)
   return content[startIndex:endIndex]
```

运行效果：

超链接的地址是CVE的详情。随便进入一个查看效果。
例如：http://cve.mitre.org/cgi-bin/cvename.cgi?name=2017-0874

这里需要记录的信息有：CVE-ID、Description、Assigning CNA和Date Entry Created。

通过查看网页源码发现，所有需要记录的信息在一个表格里面。但该页面有很多table，而且没有明显的标识来区分。而该table在div中，可以通过id来获取。
CVE-ID可以直接通过`soup.find(nowrap='nowrap').find('h2').string`获取。其他的几个信息可以通过获取相应tr中的td中的内容获得。

这样就可以获取最新的CVE列表和详情。

2. 将最新的CVE信息存入数据库

数据库采用了Mongodb。安装方法apt-get install mongodb
然后启动数据库

```
mkdir /var/data/ #■■■■■■■■
mongod --port 65521 --dbpath /var/data/ --bind_ip 127.0.0.1 #■■mongodb■■■■■■■■■■■■■■■
mongo 127.0.0.1:65521/mydb
db.createUser({user:'tass',pwd:'liehu',roles:[{role:'dbOwner',db:'mydb'}]}) #■■■■
```

Mongodb数据库插入一条数据，一般使用的是insert。

```
db.test.insert({"title":"test1", "blog_cont":"test1"})
```

如果我们想实现一个如果title存在，就对数据进行更新，不存在，就插入。可以这样来实现。

```
db.test.update({"title":"test2"}, {$set:{"title":"test2", "blog_cont":"test2"}}, {upsert:true})
db.test.update({"title":"test1"}, {$set:{"title":"test1", "blog_cont":"test3"}}, {upsert:true})
db.test.find()
```

执行完成后最终有两条数据，title分别为test1和test2，对应的内容为test3和test2.

因此在插入数据的时候，我们可以直接使用db.test.update({"title":"test2"}, {$set:{"title":"test2", "blog_cont":"test2"}}, {upsert:true})这种方式来实现。

更新只需更改data内容即可。

> 为了数据库的安全性，使用--bind_ip 127.0.0.1来设置数据库仅本地可以连接。更多mongodb数据库的配置可以参考MongoDB Mongodb.conf 配置Auth。

3. 通过邮件发送最新的CVE信息

发送邮件这里用到了smtplib。
发送邮件比较简单，就直接贴代码了。

```
def sendEmail(mail_msg):  # ■■■■■
   sender = 'from@163.com' # ■■■
   password = 'password' # ■■■■■■
   receiver = 'receiver@163.com' # ■■■
   message = MIMEText(mail_msg, 'plain', 'utf-8') #■■■■■■
   message['From'] = sender
   message['To'] = receiver

   subject = '■■CVE■■'
   message['Subject'] = Header(subject, 'utf-8')

   try:
       smtpObj = smtplib.SMTP('smtp.163.com')
       smtpObj.login(sender, password)
       smtpObj.sendmail(sender, receiver, message.as_string())
       print('■■■■■■■')
   except smtplib.SMTPException:
       print('Error: ■■■■■■■')
```

4. 定时执行任务

直接使用linux下的crontab来完成。
例如设置每天早上7点执行，可以这样设置：

```
0 7 * * * python /myJob/CVE-Monitor.py >> /log/CVE-Monitor.log
```

> 根据https://cassandra.cerias.purdue.edu/CVE_changes/
> 看到today.html更新的时间是明天的06:53，对应北京时间是19:53。若想及时获取，可以更换时间为20:00.

5.完善和优化

到这里监控脚本完成的差不多了，剩下就是如何来融合一起并改善了。
为了方便发送邮件内容和插入数据库，我们新建类CVEInfo。主要代码如下：

```
class CVEInfo:
   def __init__(self,url, cveid, description, company, createdate):
       self.url = url
       self.cveid = cveid
       self.description = description
       self.company = company
       self.createdate = createdate

   def show(self):
       return '<p><b>■■■■■■</b><a href="'+self.url+'">'+self.cveid+'</a></p><b>■■■■■■</b>'\
           +self.company +'<br><b>■■■■■■</b>'\
           +self.createdate+'<br><b>■■■■■■</b>'\
           +self.description + '<br><br><hr/>'

   def add(self):
       data = {
           'cveid': self.cveid,
           'description': self.description,
           'company': self.company,
           'createdate': datetime.strptime(self.createdate, "%Y%m%d"),
           'addDate': time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time())),

       }
       return data
```

为了美观，将邮件以html方式发送

```
message = MIMEText(mail_msg, 'html', 'utf-8')
```

邮箱收到的效果：

查看数据库数据：

从上面两张图片可以看到有三十多个，但我们有时候并不是都需要看。我们可以根据Description中关键信息来进行过滤，仅仅将我们需要关注的CVE信息发送到邮箱或进行
如下图为获取CVE-2017-8295的信息。

然后修改main方法，根据是否有关注的CVE信息来决定邮件的内容。
这里先用本地服务器为例，新建today.html文件，其中包含CVE-2017-9805和CVE-2017-16241。

运行代码结果打印了一条包含了我们的关键字的数据。
邮件中的内容如下所示：

这样就能过滤其他CVE信息，仅仅记录我们关注的内容了。

0x03 总结

本文主要用到了BeautifulSoup解析网页和mongodb数据库的使用，然后就可以将想要的内容保存到数据库中。脚本并不限于在此处使用，也可以修改一下抓取其他网站内
代码地址：https://github.com/fupinglee/MyPython/blob/master/work/CVE-Monitor.py
查询的功能就不做了，若想实现其他功能，可以自行增加和修改。

0x03 参考

[1]https://xianzhi.aliyun.com/forum/topic/1694/
[2]http://blog.csdn.net/guoxingege/article/details/47339885

上一篇：渗透技巧——模拟IE浏览器下载文件 下一篇：ColdFusion反序列化漏洞分...

1. 1 条回复


o0xmuhe 2018-02-26 13:39:31

不错不错，结合tg bot，可以做一个预警了。

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板