실시간 악성파일 탐지 프로세스

BoB 8th 강성민

1. Linux 시스템

1) 사용법

[1] /home/investigate 폴더를 생성

[2] Gethistory.py, GetNetwork.py, GetProcess.py, GetService.py GetStrangeApacheAccess.py GetStrangeApacheError.py, GetVarLog.py, GetWho.py 실행

[3] 실행 이후 /home/investiage 하위에 각종 파일 생성 이후 분석 진행

2) 설계 방식

[1] NewInfohandler.py 내용

실시간 감시의 특성상 수많은 로그들을 전부다 기록할 순 없고 기록 범위를 최소화해야 함. 처음 실행 시 파일 사이즈를 저장한 뒤 파일 포인터를 이용해서 파일 사이즈가 변화하면 변화한 내용만을 기록할 수 있도록 작성함

```
def GetNewinfo_Size(file_path, write_path):
    file_size = os.path.getsize(file_path)
    file_size_compare = os.path.getsize(file_path)
    if file_size != file_size_compare:
        f = open(file_path, "rb")
        pointer1 = f.seek(file_size)
        pointer2 = f.seek(file_size_compare)

        f.seek(pointer1)
        data = str(f.read(pointer2-pointer1), encoding="utf-8")

        with open(write_path, "w+") as w:
            w.write(data)
```

그림 1. NewInfohandler..Py 내용

일부 파일들은 해당 내용이 적용되지 않아서 좀 더 비효율적이지만 비슷한 방식의 로

직을 적용

```python
def GetNewInfo(command, fileName):
    first_list = str(subprocess.check_output(command, shell=True), encoding="utf-8").split("\n")
    Dup_check_list = []
    while True:
        check_new_list = str(subprocess.check_output(command, shell=True), encoding="utf-8").split("\n")
        for item in check_new_list:
            if item in first_list:
                check_new_list.remove(item)

        for item in check_new_list:
            with open(fileName, "w+") as f:
                if item in Dup_check_list or item == "":
                    pass
                else:
                    f.write(item + "\n")
                    Dup_check_list.append(item)
```

그림 2. NewInfohandler.py 내용

[2] Apache2.py 내용

기본적으로 apache 로그를 사용할 것이라 가정함, apache 로그에서 앞서 설명한 로직과 같이 파일 사이즈를 기준으로 변화가 나타나면 해당 로그에 대한 검사를 진행할 수 있도록 함, 검사 진행은 미리 설정한 SQL관련 정규표현식, 웹쉘 관련 확장자, Css 공격, 파일명에 '.'이 두개 이상 찍혀 있는 경우를 필터링 함

```python
def Get_StrangeApache2log(log_full_path, write_full_path):
    result = subprocess.check_output('cat {}'.format(log_full_path), shell=True)
    Data_size = os.path.getsize(log_full_path)
    while True:
        Data_size = os.path.getsize(log_full_path)
        result = subprocess.check_output('tail -1 {}'.format(log_full_path), shell=True)
        str_result = str(result, "utf-8").replace("\n", "")
        Data_size_compare = os.path.getsize(log_full_path)

        if Data_size == Data_size_compare: #No new log
            pass
        else: #New log
            log = str_result + "\n"
            path = log.split(" ")[6]

            Sql_Check = check_File(path, Sql_injection_regx, write_full_path, log)
            Sus_Extension = check_File(path, Suspicious_Extension_regex, write_full_path, log)
            Css_Check = check_File(path, CssAttack_regx, write_full_path, log)
            Dot_Check = re.compile(Str_dot).findall(path)
            if len(Dot_Check) != 1:
                with open(write_full_path, "a") as f:
                    f.write(log)
                    print(log)
            else:
                pass
    f.close()
```

그림 3. Apache2.py 내용

[3] GetWho.py

앞서 설명한 방식을 활용하여 리눅스에 로그인한 사용자 정보를 출력하는 who에 변화가 나타나면 기록할 수 있도록 설계

[4] GetVarLog.py

/var/log 에는 다양한 로그들이 저장되어 있는데 그 중에서도 wtmp, btmp, laslog, xferlog, crom, secure, httpd/access_log, httpd/error_log를 수집하도록 함, 해당 로그들은 주기를 설정한 이후 총 3번 수집해서 향후 비교가 가능하도록 설정(주기는 본인이 설정 가능)

```
time_input = int(input("time cycle? : "))
for i in range(1, 4):
    CollectVarLog("wtmp", i)
    CollectVarLog("btmp", i)
    CollectVarLog("lastlog", i)
    CollectVarLog("xferlog", i)
    CollectVarLog("cron", i)
    CollectVarLog("secure", i)
    CollectVarLog("messages", i)
    CollectHttpdLog("access_log", i)
    CollectHttpdLog("error_log", i)
    CheckSetUID(i)
    time.sleep(time_input)
```

그림 4. 로그 수집

추가적으로 백도어 탐지를 위해서 SetUID가 설정된 파일을 찾아서 기록하도록 설정

DB 로그의 경우에는 기본적으로 general_log가 꺼져있고 웹 서버가 작동하는 상태에서 설정 변경을 위해 재시작하는건 불가하다고 판단하고 error log만을 수집

```
Service_list = GetServiceList()
if "mongodb" in Service_list:
    subprocess.call("cp /var/log/mongodb/error.log /home/investigate/", shell=True)

elif "mysql" in Service_list:
    subprocess.call("cp /var/log/mysql/error.log /home/investigate/", shell=True)

elif "mariadb" in Service_list:
    subprocess.call("cp /var/log/mariadb/error.log /home/investigate/", shell=True)
```

그림 5. DB 로그 수집

이후 마지막으로 네트워크, 프로세스, crontab 정보를 수집

```
subprocess.call("netstat -nalp > /home/investigate/final_netsta.txt", shell=True)
subprocess.call("ps -aux > /home/investigate/final_process.txt", shell=True)
subprocess.call("cp /var/spool/cron /home/investigate/", shell=True)
```

그림 6. 설정 주기 종료 이후 마지막 수집 내용

[5] GetProcess, GetNetwork, Gethisory

앞서 설명한 파일 크기 변화에 따른 차이에 따라 새로운 내용을 기록

```
def GetProcessInfo():
    GetNewInfo('ps -aux', "/home/investigate/new_process.txt")

if __name__ == "__main__":
    GetProcessInfo()
```

```
from NewInfohandler import GetNewInfo

def GetNetworkInfo():
    GetNewInfo('netstat -nalp', "/home/investigate/new_network.txt")

if __name__ == "__main__":
    GetNetworkInfo()
```

```
def Gethistory():
    #계정 파악
    User_list = str(subprocess.check_output('grep /bin/bash /etc/passwd | cut -f1 -d:', shell=True),encoding="utf-8").split('\n')
    while True:
        for user in User_list:
            if user == '':
                pass
            elif user == 'root':
                GetNewinfo_Size("/root/.bash_history","/home/investigate/root_history")

            else:
                GetNewinfo_Size("/home/{}/.bash_history".format(user),"/home/investigate/{}_history".format(user))

if __name__ == "__main__":
    Gethistory()
```

그림 7. GetProcess, GetNetwork, Gethisoty 내용들

2. 윈도우

뼈대만 짜고 실행까지 할 수 있도록 완성하지는 못했습니다. 진행한 것까지만 설명 드리겠습니다.

[1] 실행 프로세스 탐지

코드 실행 이후 실행되는 프로세스 만을 탐지하기 위해서 event id 감사를 통해

'4688'에 해당하는 프로세스 발견 시 리턴 하도록 설정함

```python
def GetNewProcess():
    while True:
        server = 'localhost'
        logtype = 'Security' # 새 프로세스 생성은 EventID는 4688로 Security에 포함
        hand = win32evtlog.OpenEventLog(server, logtype)
        flags = win32evtlog.EVENTLOG_BACKWARDS_READ|win32evtlog.EVENTLOG_SEQUENTIAL_READ
        total = win32evtlog.GetNumberOfEventLogRecords(hand)
        events = win32evtlog.ReadEventLog(hand, flags, 0)

        if events:
            for event in events:
                evtTime = str(event.TimeGenerated)
                dt = datetime.datetime(int(evtTime[0:4]),int(evtTime[5:7]),int(evtTime[8:10]),int(evtTime[11:13]),int(evtTime[14:16]),int(evtTime[17:19])).timestamp()
                    print("새 프로세스 생성 감지")
                    print("Source Name: ", event.SourceName)
                    print("Time generated: ", event.TimeGenerated)
    return event.SourceName
```

그림 8. 실행 프로세스 탐지 내용

[2] 프로세스 분석

프로세스에 대해 PE 파일 여부를 확인한 이후 sysinternals의 sigcheck를 이용해 디지털 서명 여부를 확인함, 추가적으로 IAT 테이블을 확인하여 사용하는 모듈을 확인

```python
def Check_PeNSigned(check_path, sigcheck_path):
    PeList = []
    NotSigned = []
    checkList = GetChecklist(check_path)
    for item in checkList:
        check_file = open(item, 'rb')
        byte = check_file.read(2)
        if byte == b'MZ':
            PeList.append(item)

        else:
            continue

    for Pe_file in PeList:
        cmd = [sigcheck_path, Pe_file]
        fd_popen = subprocess.Popen(cmd, stdout=subprocess.PIPE).stdout
        data = str(fd_popen.read().strip(),encoding="utf-8")
        if "Signed" in data:
            continue
        else:
            NotSigned.append(path)

    return NotSigned
```

그림 9. 디지털 서명 check

```python
def Check_IAT(check_path, Listdlls_path): #Listdlls.exe
    checkList = GetProcess()
    IAT_list = []
    for item in checkList:
        cmd = [Listdlls_path, item]
        fd_popen = subprocess.Popen(cmd, stdout=subprocess.PIPE).stdout
        try:
            data = str(fd_popen.read().strip(), encoding="utf-8")
            IAT_list.append(data)
        except:
            pass
```

그림 10. IAT 테이블 확인

해당 IAT 테이블에서 확인한 함수들과 악성코드들이 주로 사용하는 함수들이 얼마나 있는지 위험도 판단의 기준으로 삼고자 하였음

[3] 위험 프로세스 발견 이후

CLI 도구를 이용해 수집, 분석을 진행하고자 함

forecopy	프리패치, 웹 아티팩트, mft, 레지스트리 하이브 파일, 이벤트 로그 수집
Autorunsc	자동 실행으로 등록된 프로그램 조사
Strings	의심 프로세스이 asci, Unicode 조사
Psservice	서비스 점검
Psloglist	이벤트 로그 분석