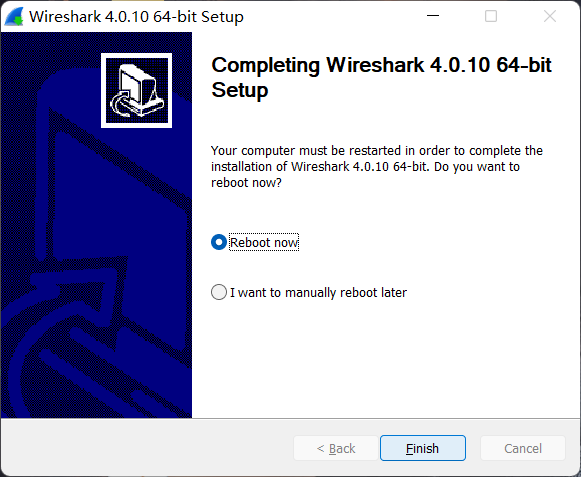
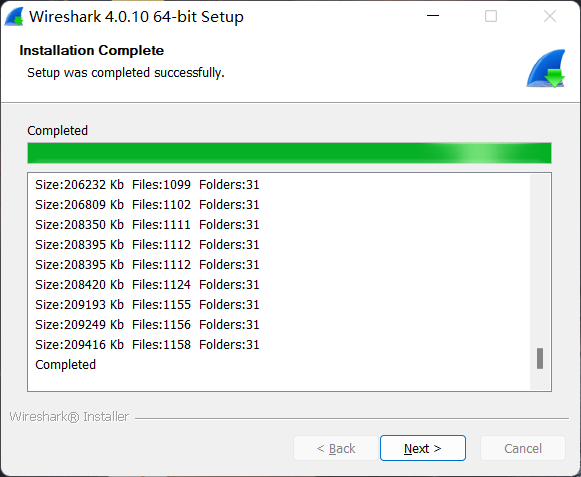
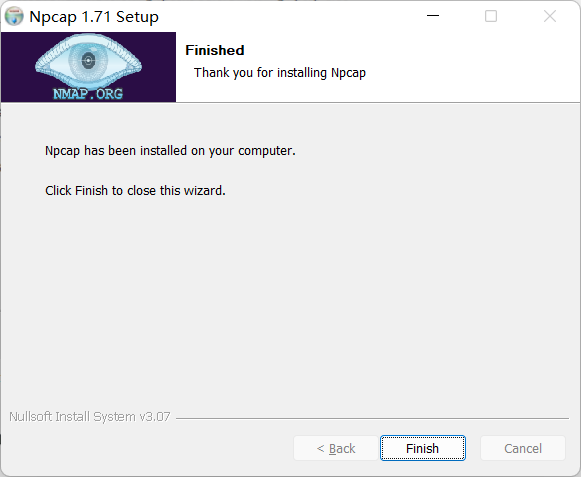
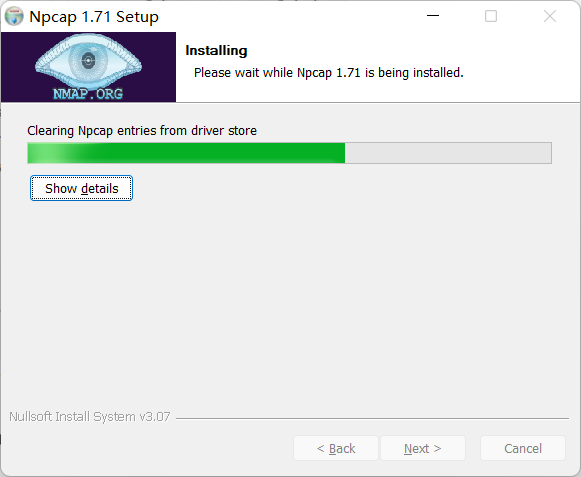
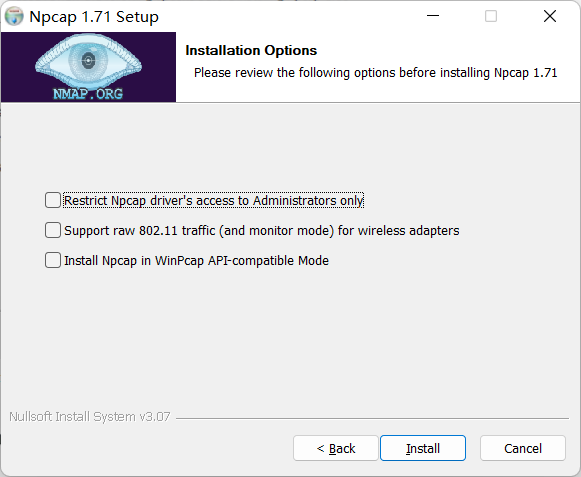
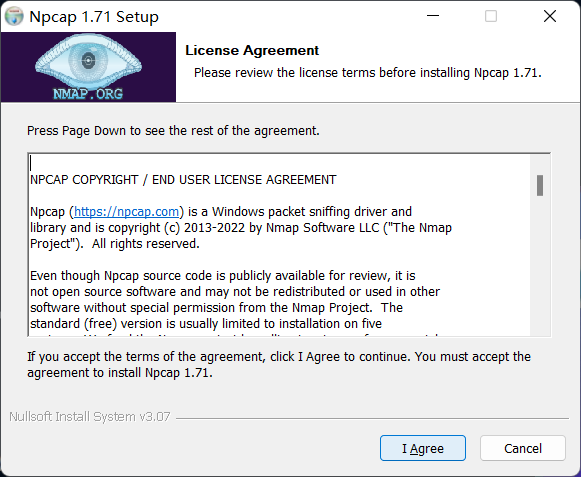
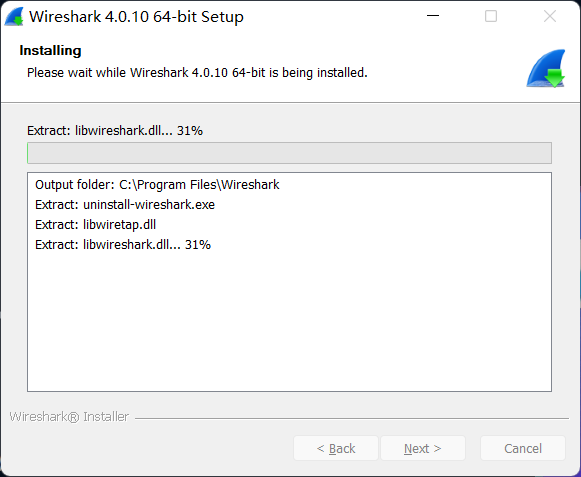
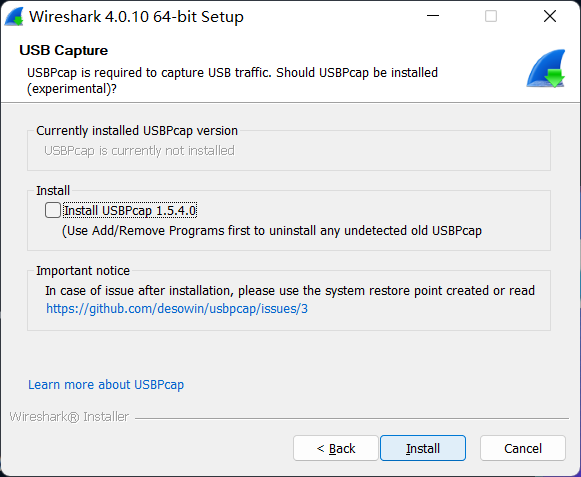
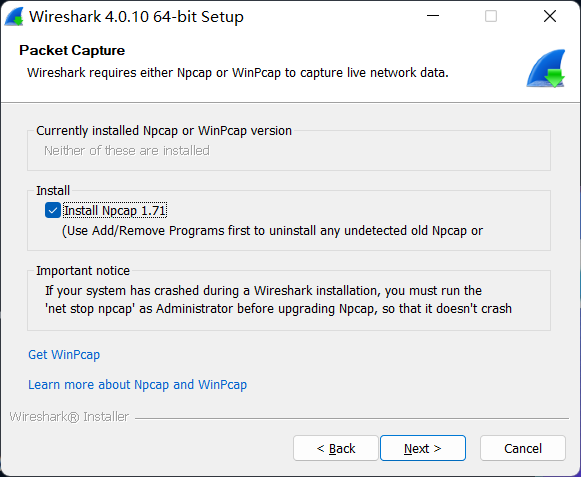
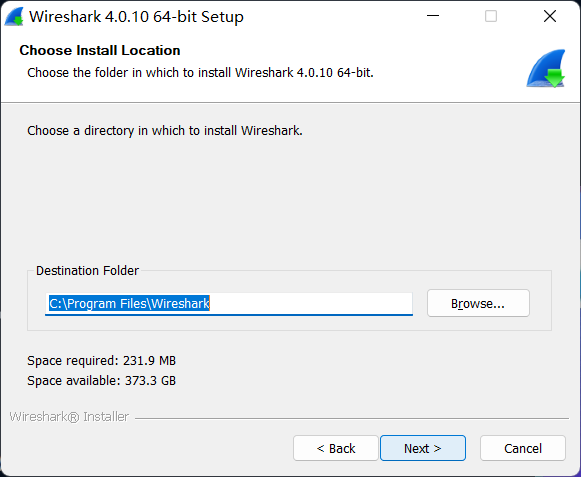
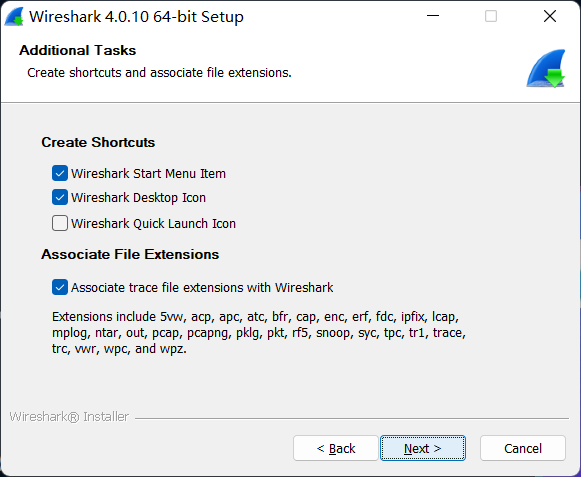
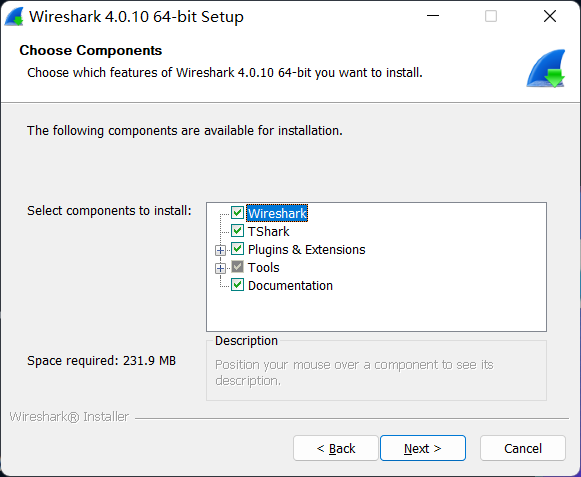
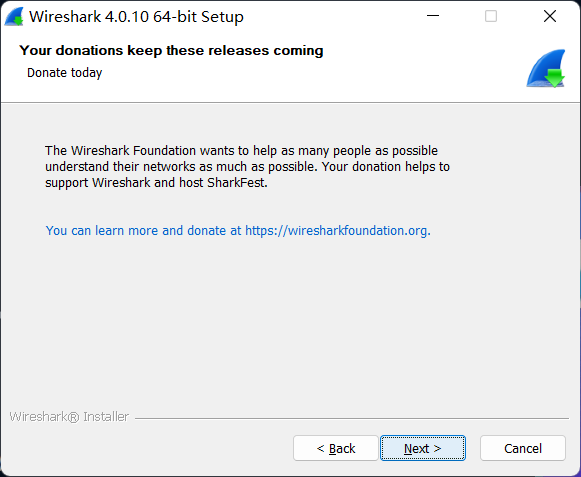
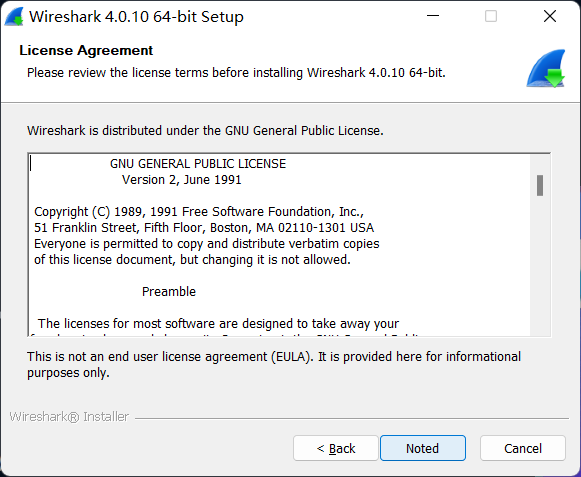
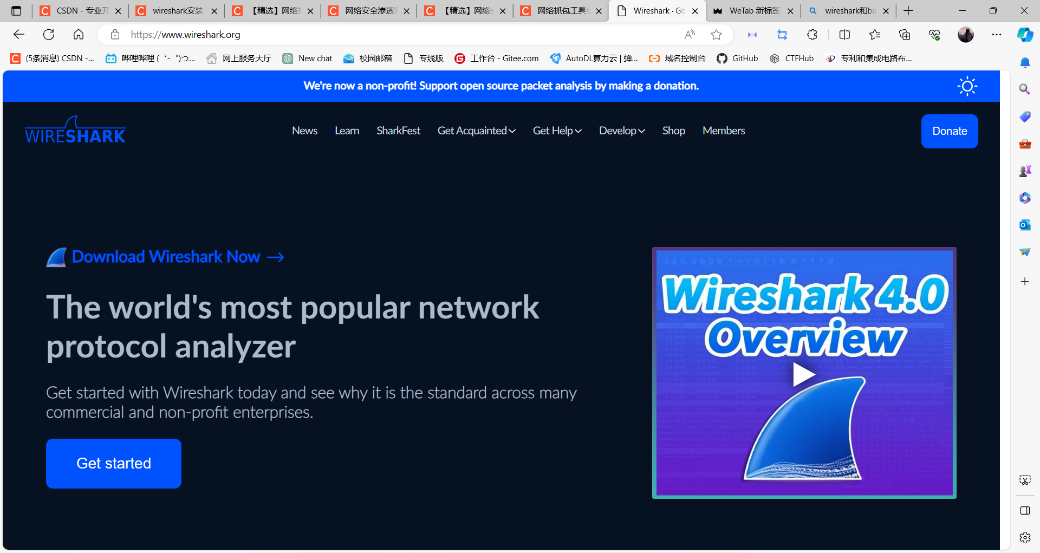
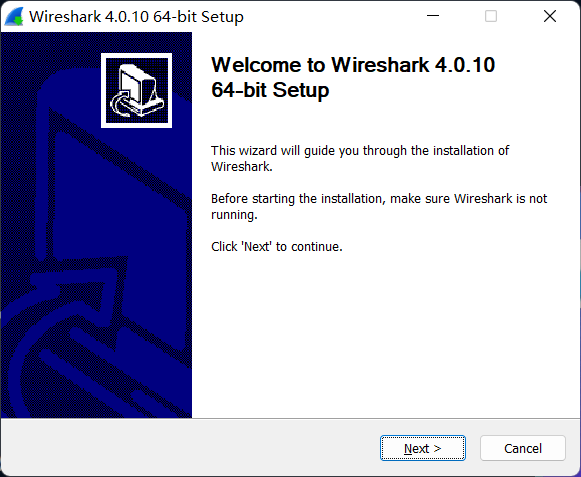
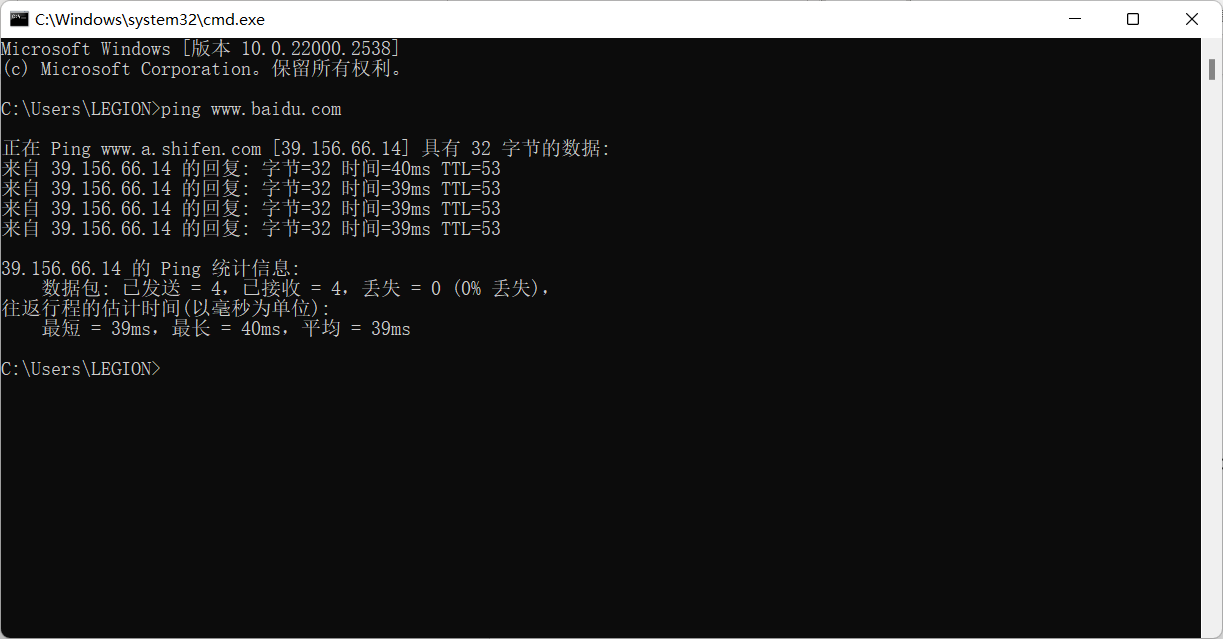
## 1：Wireshark的安装与使用

### 1.1安装过程以及截图

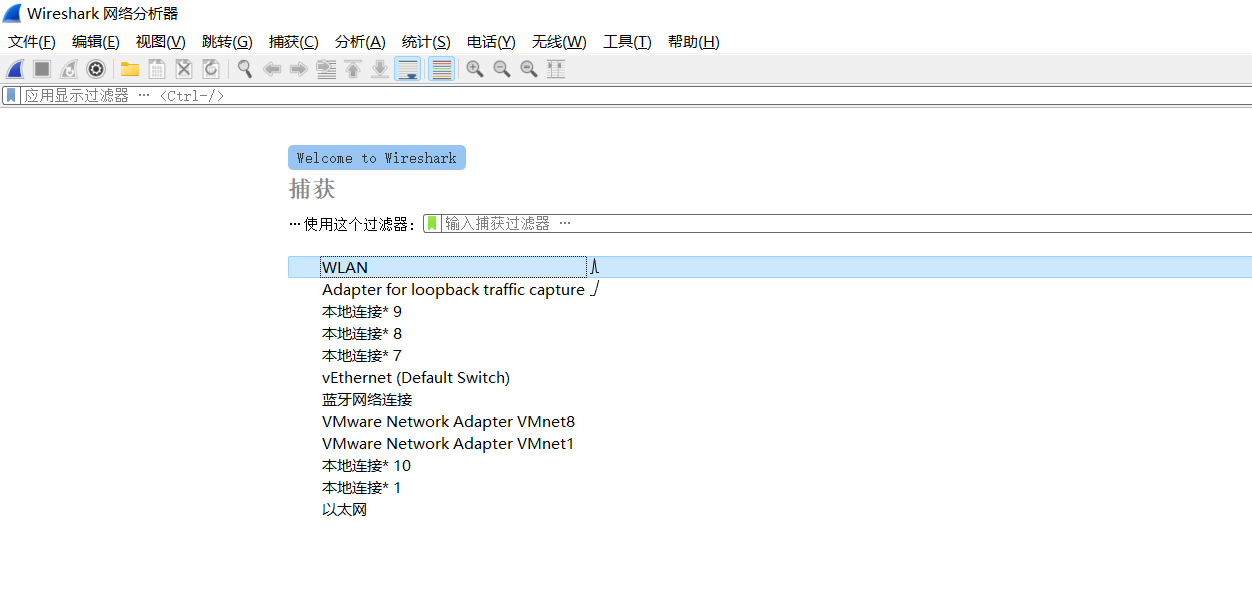


### 1.2 Wireshark演示一个TCP 三次握手的过程

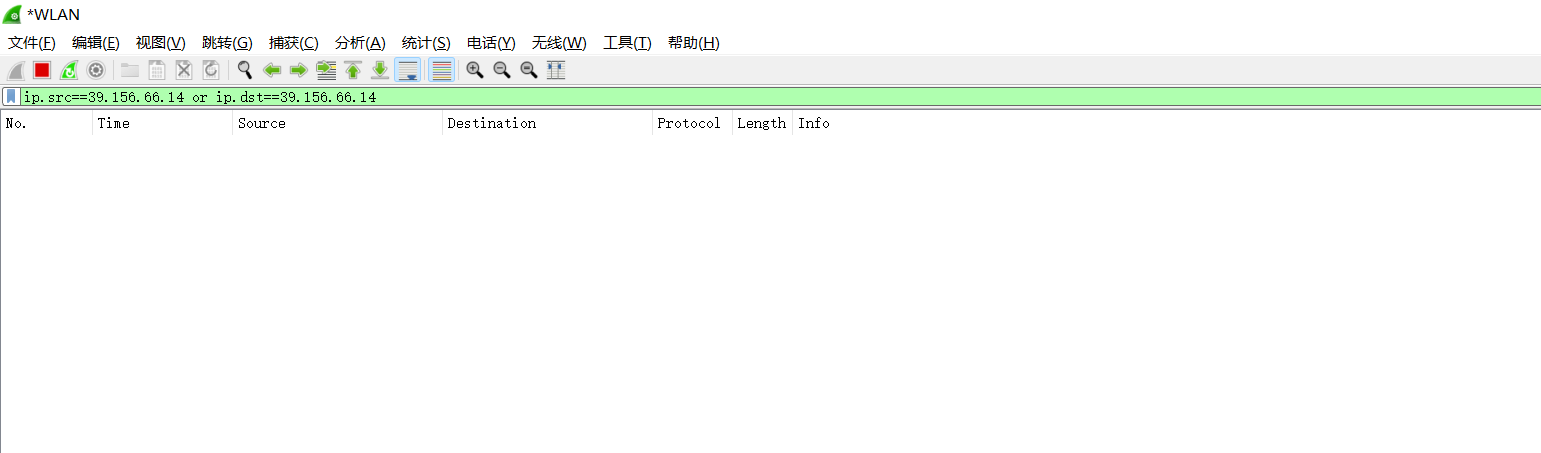
（1）通过ping指令获取目标网站https://www.baidu.com的ip: 39.156.66.14。



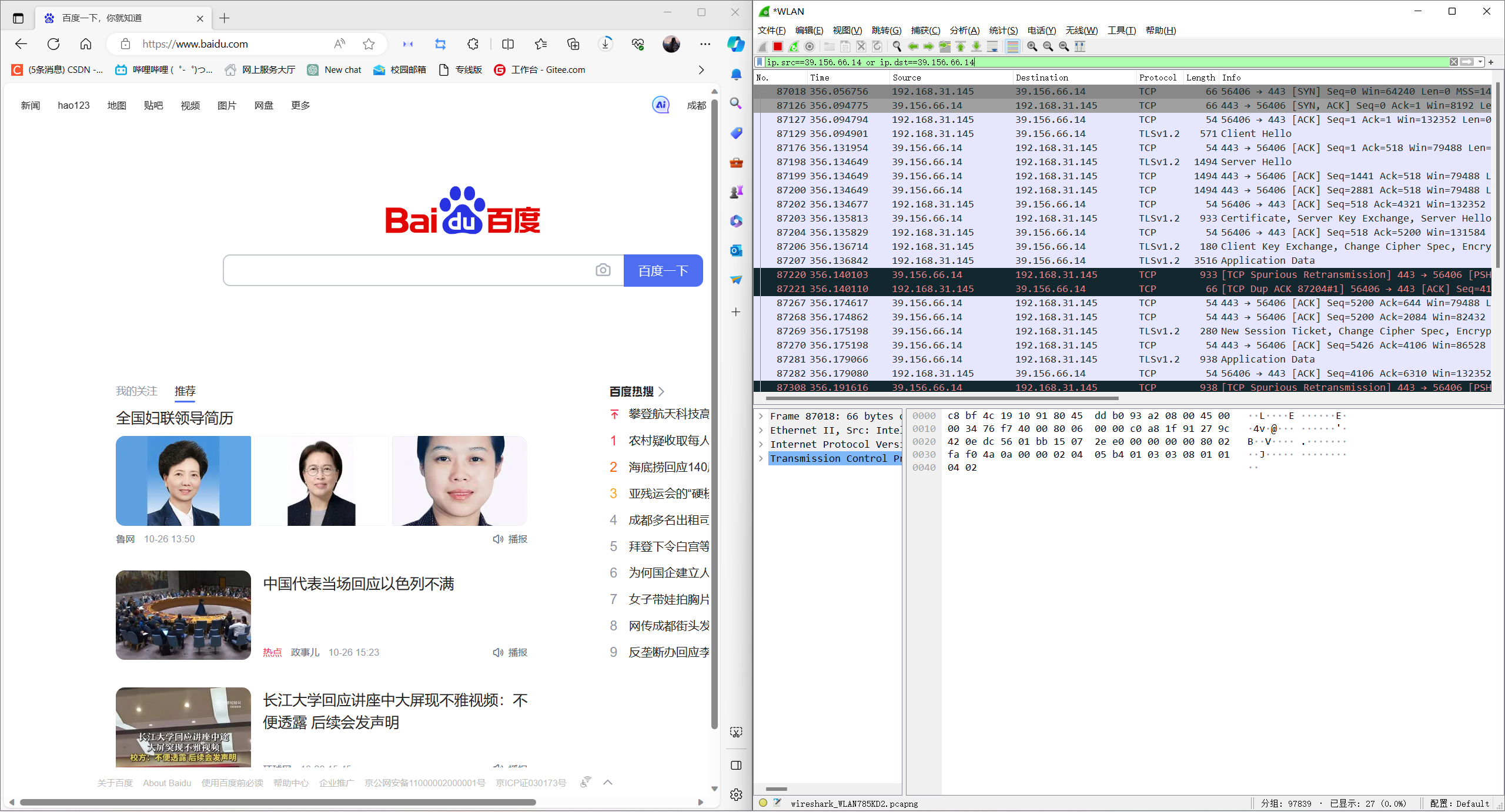
（2）打开Wireshark选择WLAN作为过滤器



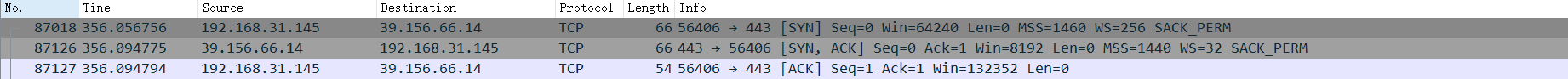
(3)筛选目标ip，此时因为没有访问记录所以看不到报文



（4）访问目标网址后即可筛选出对应的包

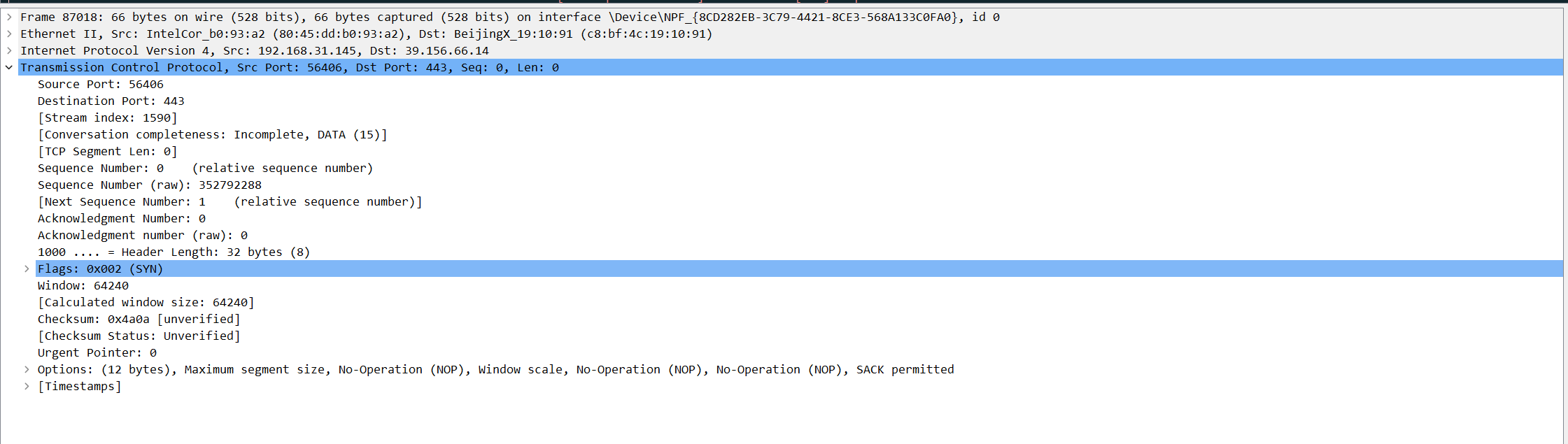


（5）查询时间最早的三条记录即为TCP三次挥手记录



（6）第一次握手

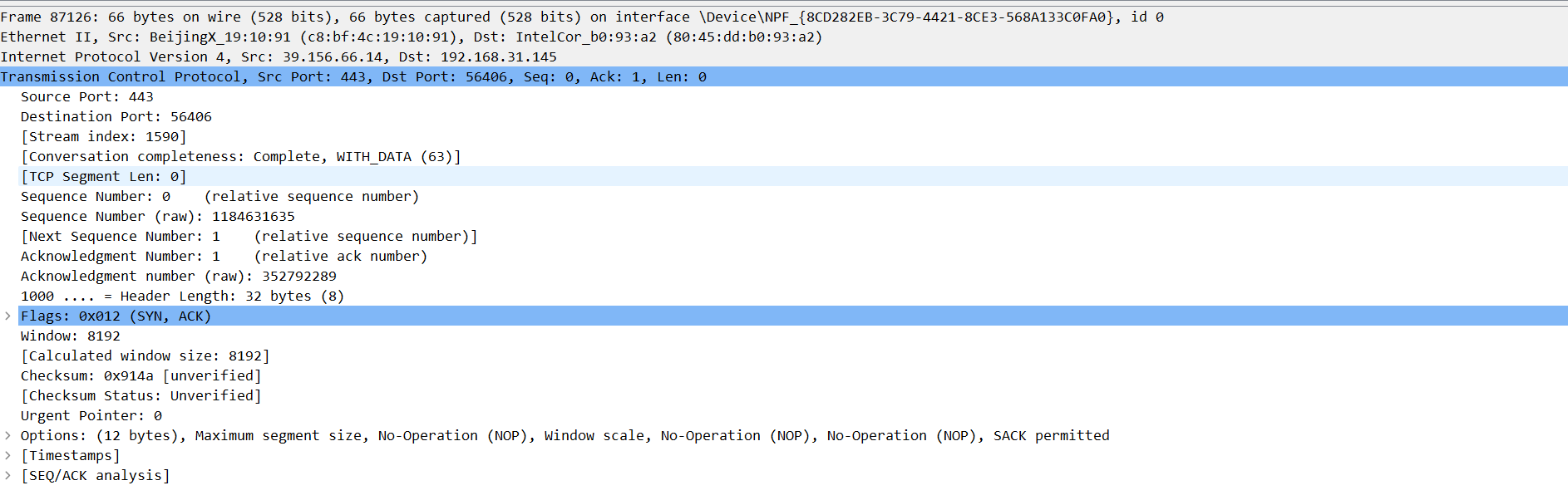




分析得第一次由主机向服务器发送SYN报文发送建立需求，SYN信号为1，序列号seq=0，下一个将要发送的序列号是1，然后确认号ack为0

（7）第二次握手

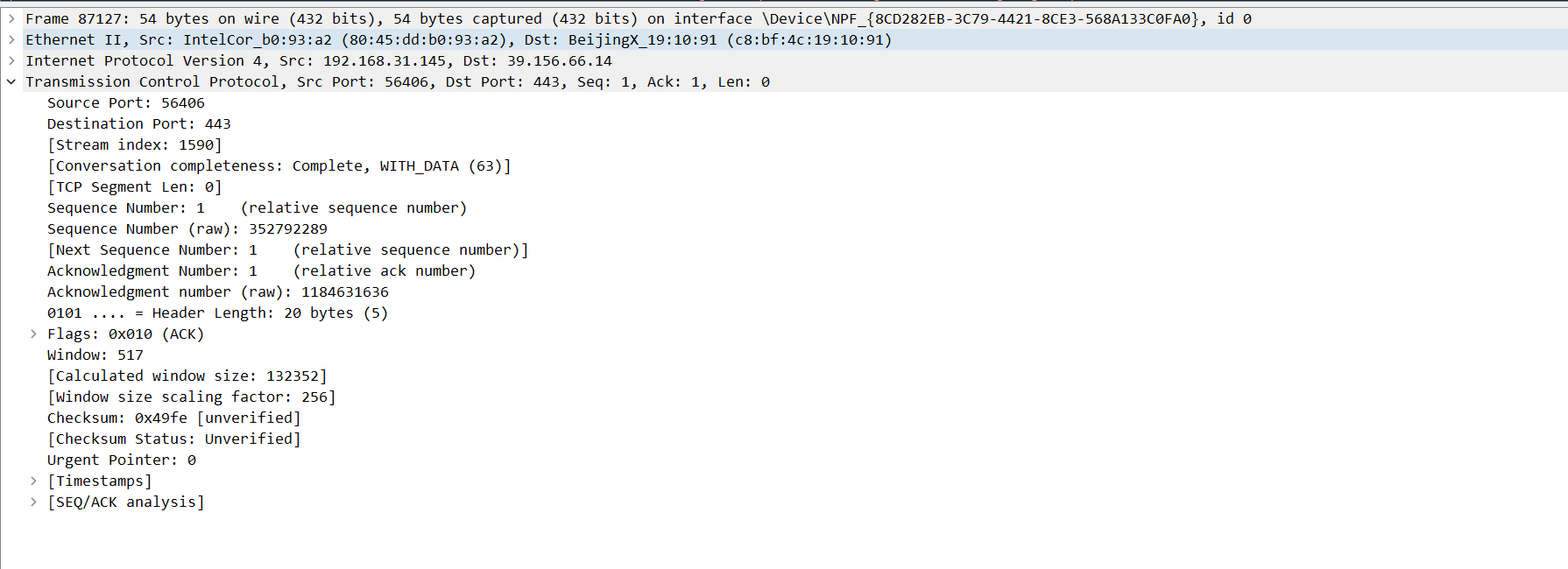




分析得第二次是由服务器向主机发送SYN+ACK报文发送确认信息，SYN信号为1，ACK反馈信号为1，序列号seq=0表示服务器发送给主机默认开始为0，ack=1表示成功接收到之前的序列并且希望下一个接收到的序列号为1，下一次发送的序列号为1

（8）第三次握手





分析得第三次是由主机向服务器发送ACK确认报文，ACK反馈信号为1，seq表示发送的序列号为1，ack=1表示确认收到序列号0的消息并且下一条希望接收到的消息序列号1，至此三次握手完毕成功建立TCP连接进行消息传输。

## 2.X-scan的安装与使用

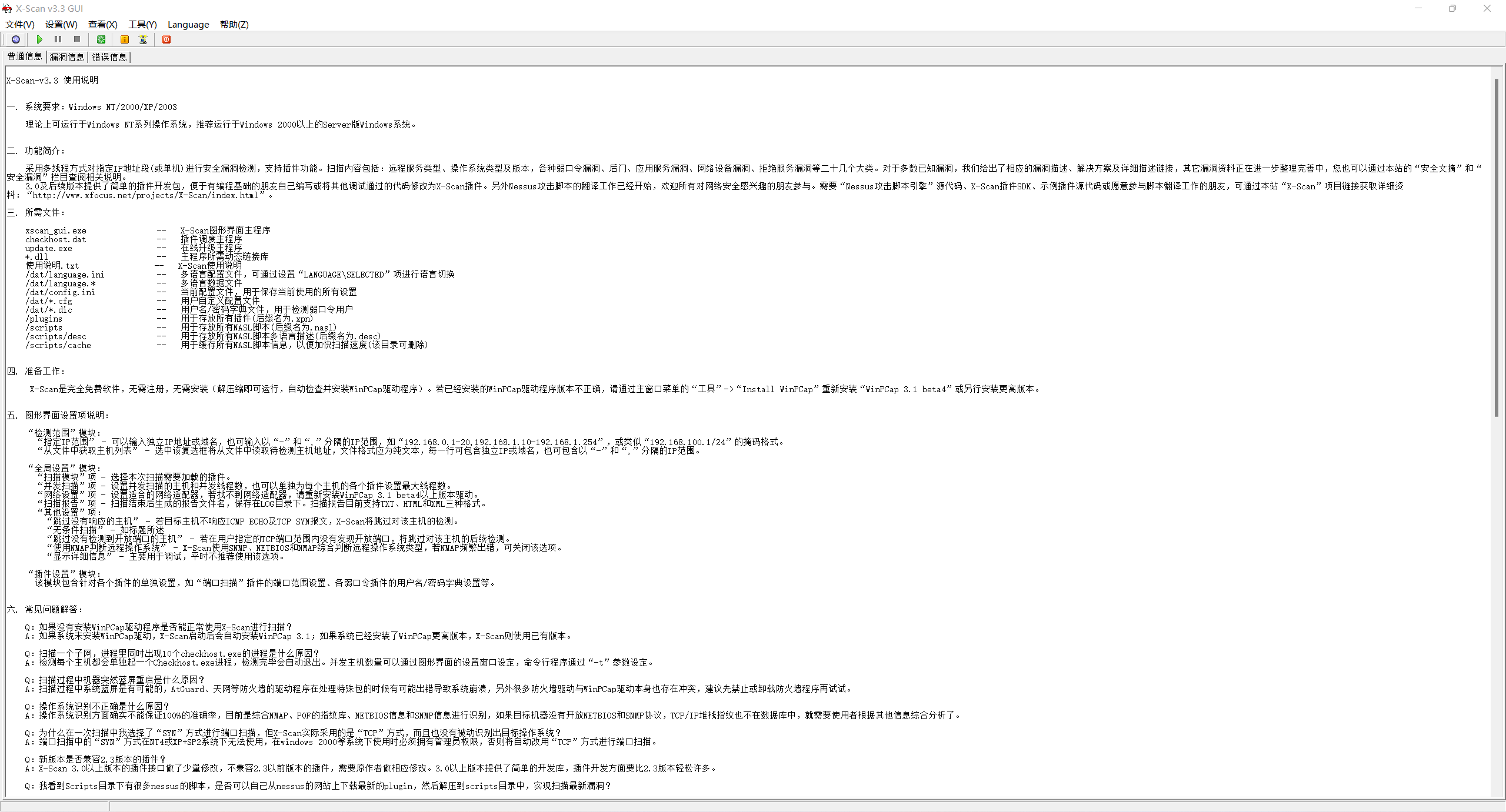
### 2.1 X-scan的安装与配置

获取x-scan.zip然后解压缩到本地，查看是否缺少相关ddl文件并补齐所有配置后，如下所示，打开xscan\_gui.exe进行使用

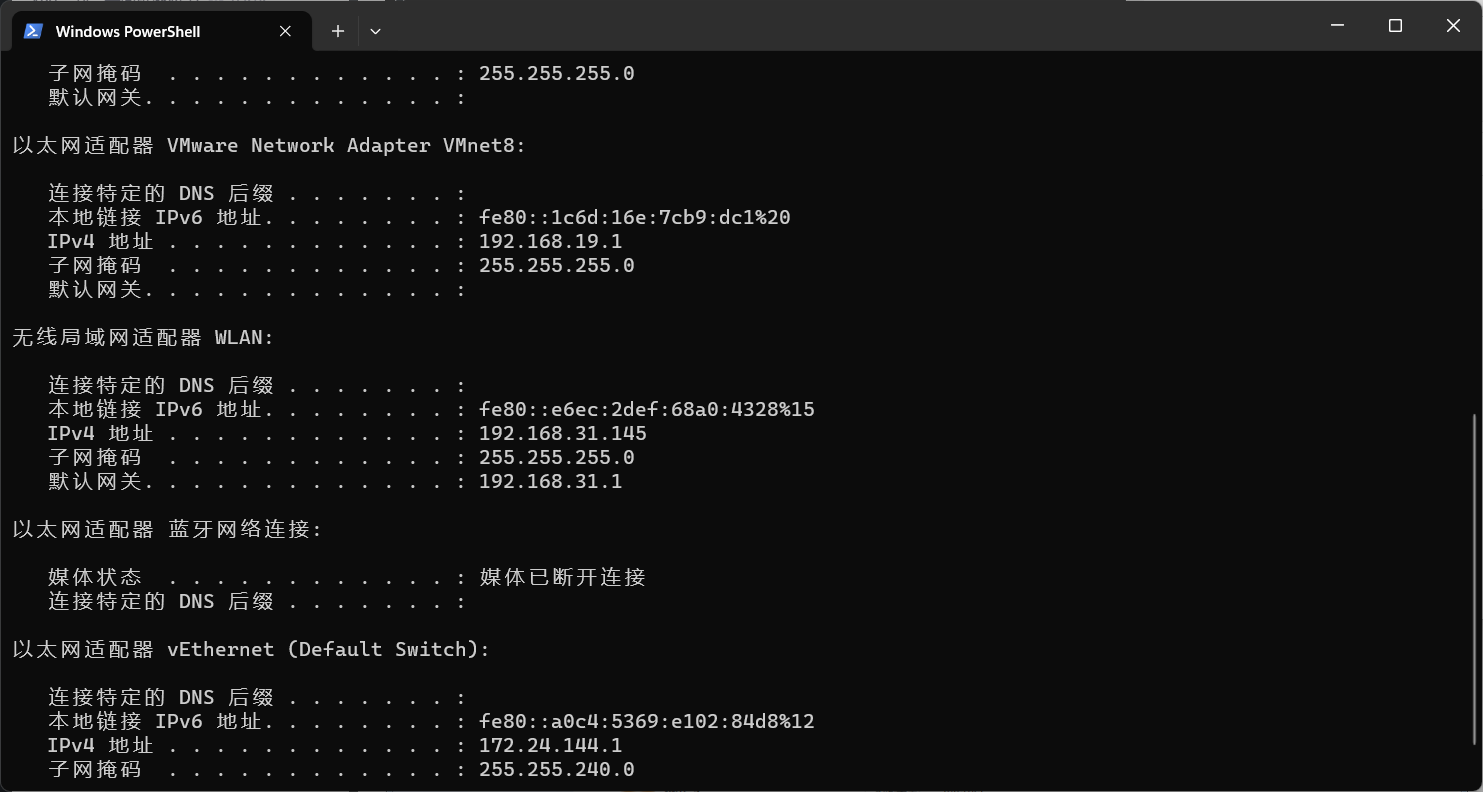


### 2.2 使用X-scan对指定的地址段进行扫描

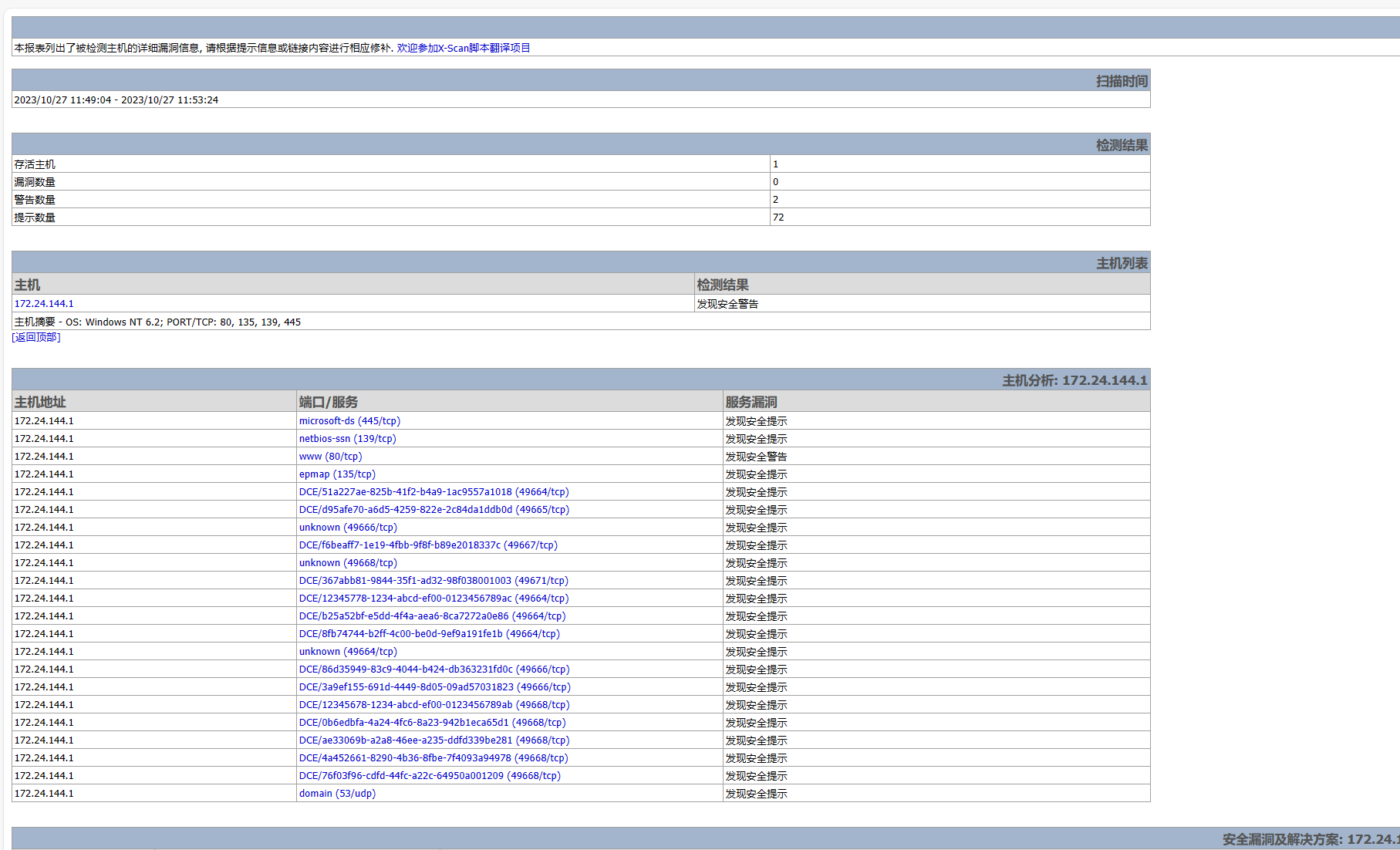
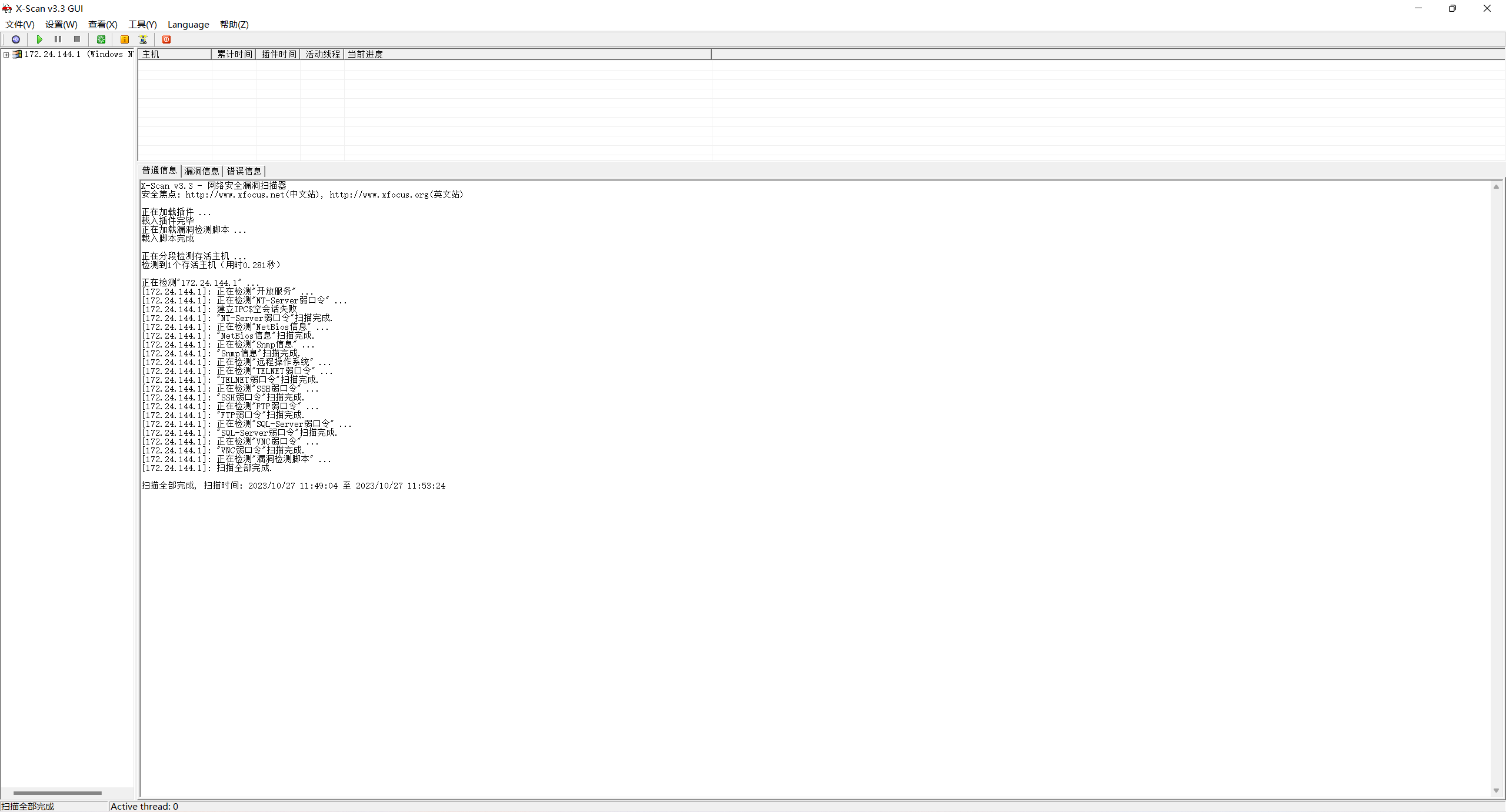
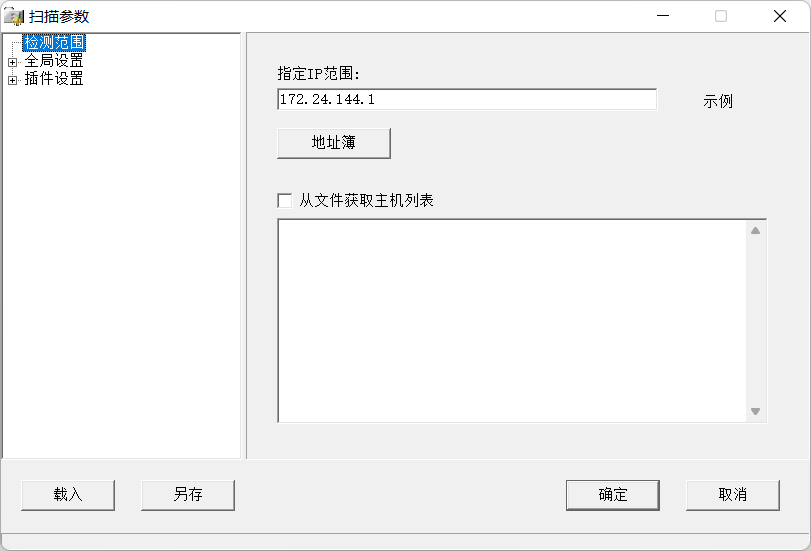
(1)打开xscan\_gui.exe



（2）在DOS窗口中输入ipconfig获取本机ip

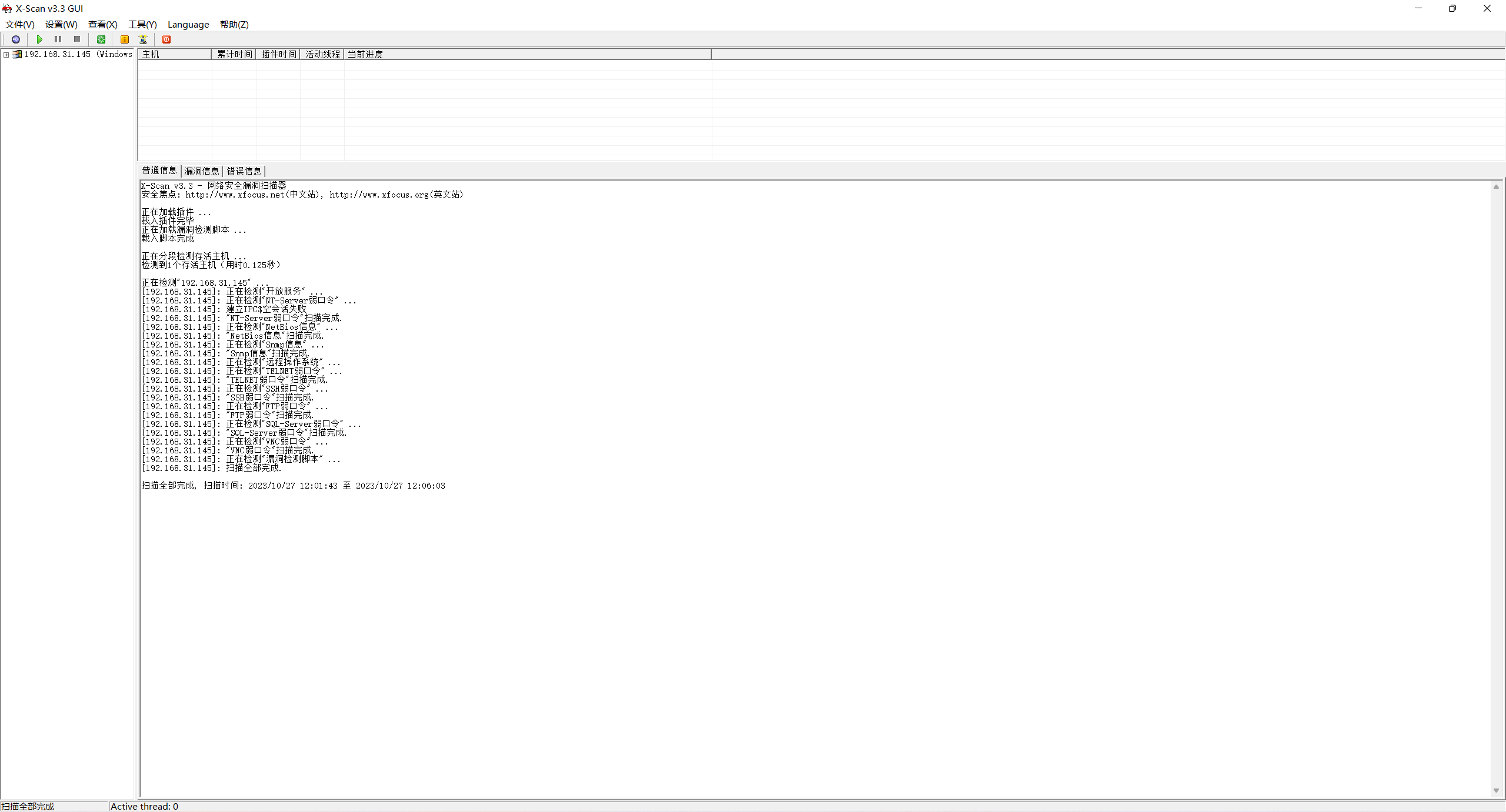
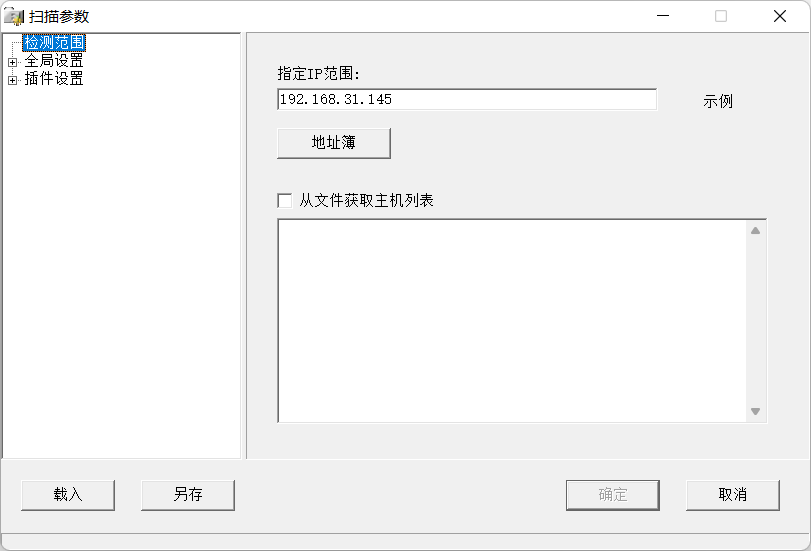


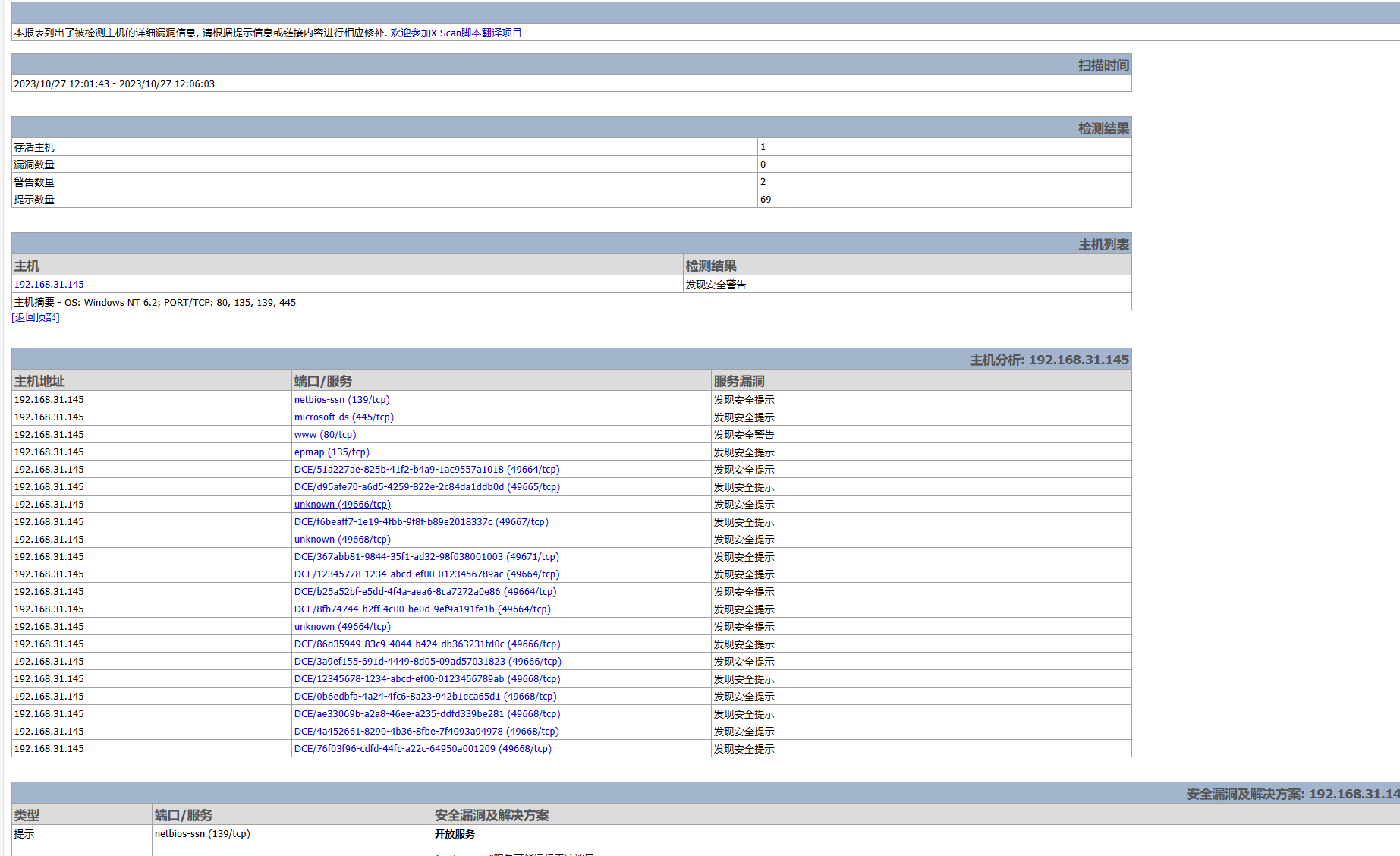
（3）在X-scan中设置ip参数实现扫描特定地址段，扫描以太网ip：172.24.144.1



分析得不存在漏洞

（3）在X-scan中设置ip参数实现扫描特定地址段，扫描WLAN无线ip：192.168.31.145





分析得到不存在漏洞

## 3. 缓冲区溢出

### 3.1 阐述缓冲区溢出的原理

缓冲区溢出通常发生在程序尝试写入超过目标缓冲区的数据时。程序通常在内存中分配一块固定大小的缓冲区用于存储数据，但如果程序没有正确地验证输入数据的大小，攻击者可以发送比缓冲区容量更大的数据，从而导致数据溢出到相邻的内存区域。攻击者可以在这些相邻内存区域中插入恶意代码，修改程序的控制流，或者损坏数据结构，从而危害系统的安全性和稳定性。

### 3.2 陈述缓冲区溢出的危害及解决办法

#### 3.2.1危害：

（1）攻击者可以执行恶意代码，控制受感染的程序，甚至整个操作系统。

（2）数据泄露：攻击者可能能够读取内存中的敏感数据，如密码、私钥等。

（3）服务拒绝：缓冲区溢出可能导致程序崩溃，拒绝为合法用户提供服务。

#### 3.2.2解决办法：

（1）输入验证：确保在将数据写入缓冲区之前对输入数据进行验证，确保其不会超出缓冲区的大小。

（2）使用安全的API：使用编程语言和库提供的安全函数来处理字符串和缓冲区，如strncpy替代strcpy。

（3）防御性编程：在程序中实施额外的安全措施，如堆栈保护、地址空间布局随机化（ASLR）和非执行栈，以减少攻击的成功率。

（4）定期安全审计：对代码进行定期的安全审计，以识别和修复潜在的缓冲区溢出漏洞。

### 3.3编写一段有缓冲区溢出风险的代码，并分析。

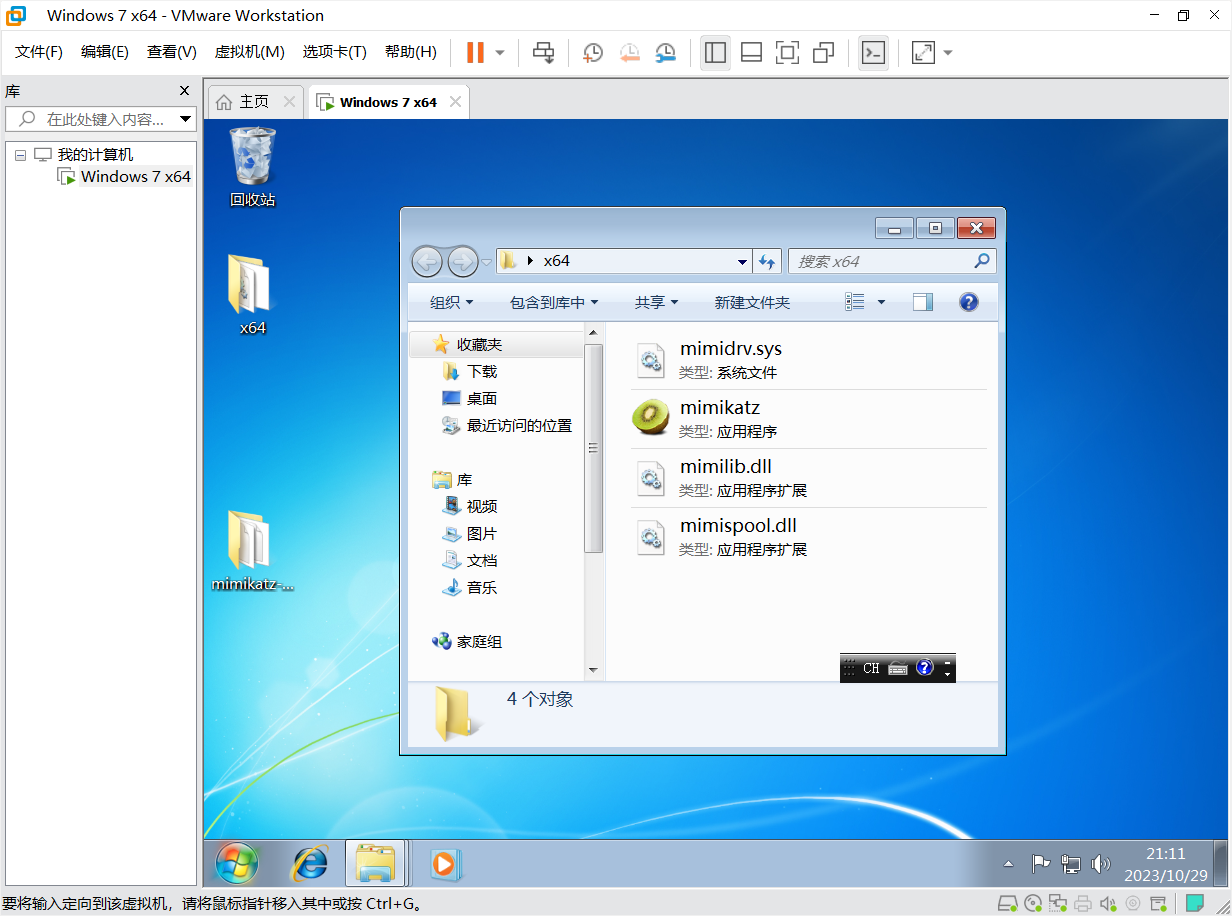
1. #include <stdio.h>
2. #include <string.h>
4. **int** main() {
5. **char** buffer[5]; // 缓冲区大小为5
6. **char** input[10];  // 输入数据
8. printf("输入数据：");
9. gets(input); // 不进行输入大小验证的危险函数
11. strcpy(buffer, input); // 潜在的缓冲区溢出
13. printf("拷贝后的数据：%s\n", buffer);
15. **return** 0;
16. }

在上述代码中，gets函数没有验证输入的大小，因此，如果用户输入的数据大于5个字符比如“overflow”，将导致buffer溢出。这种代码容易受到缓冲区溢出攻击，应该避免使用不安全的函数，如gets，并进行输入验证。可以改用fgets函数来限制输入数据的大小，从而减少风险。

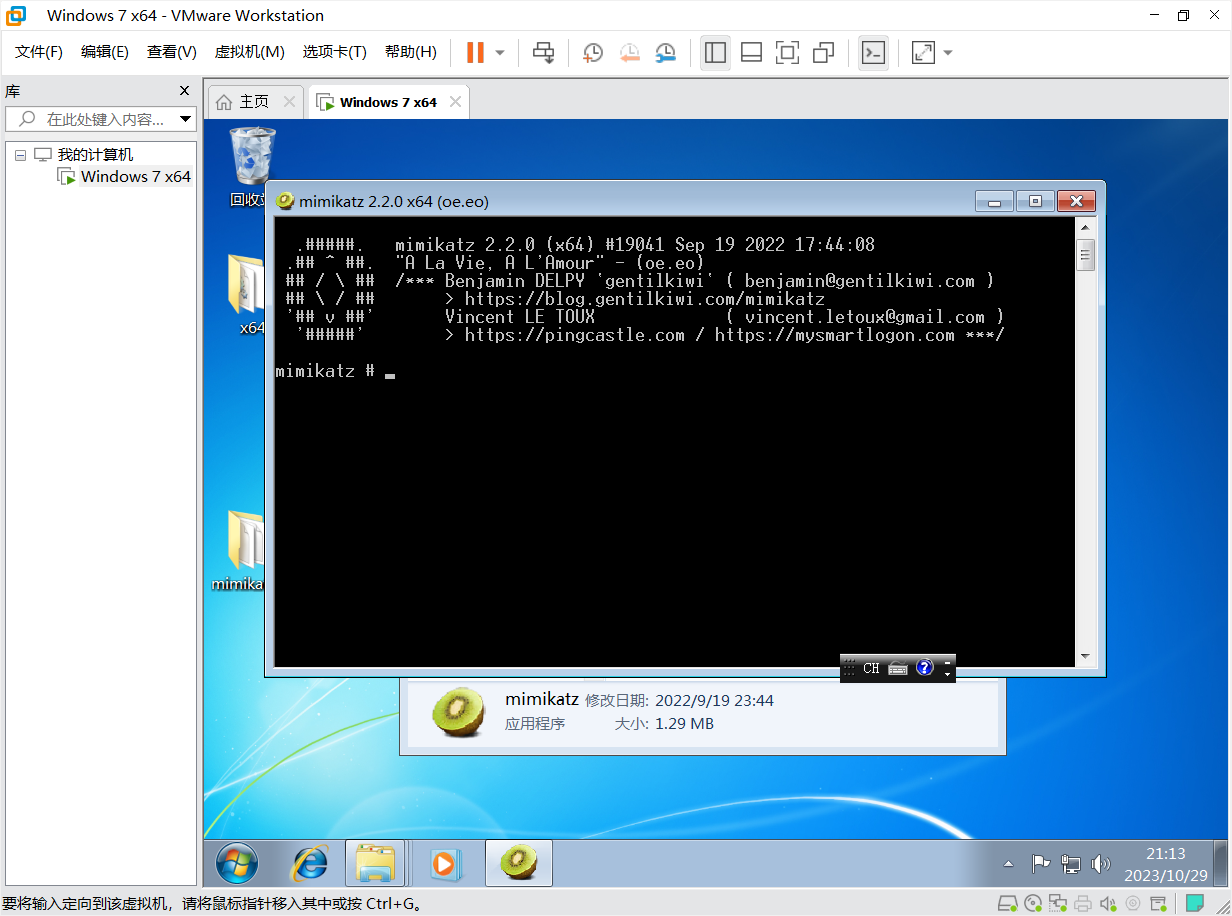
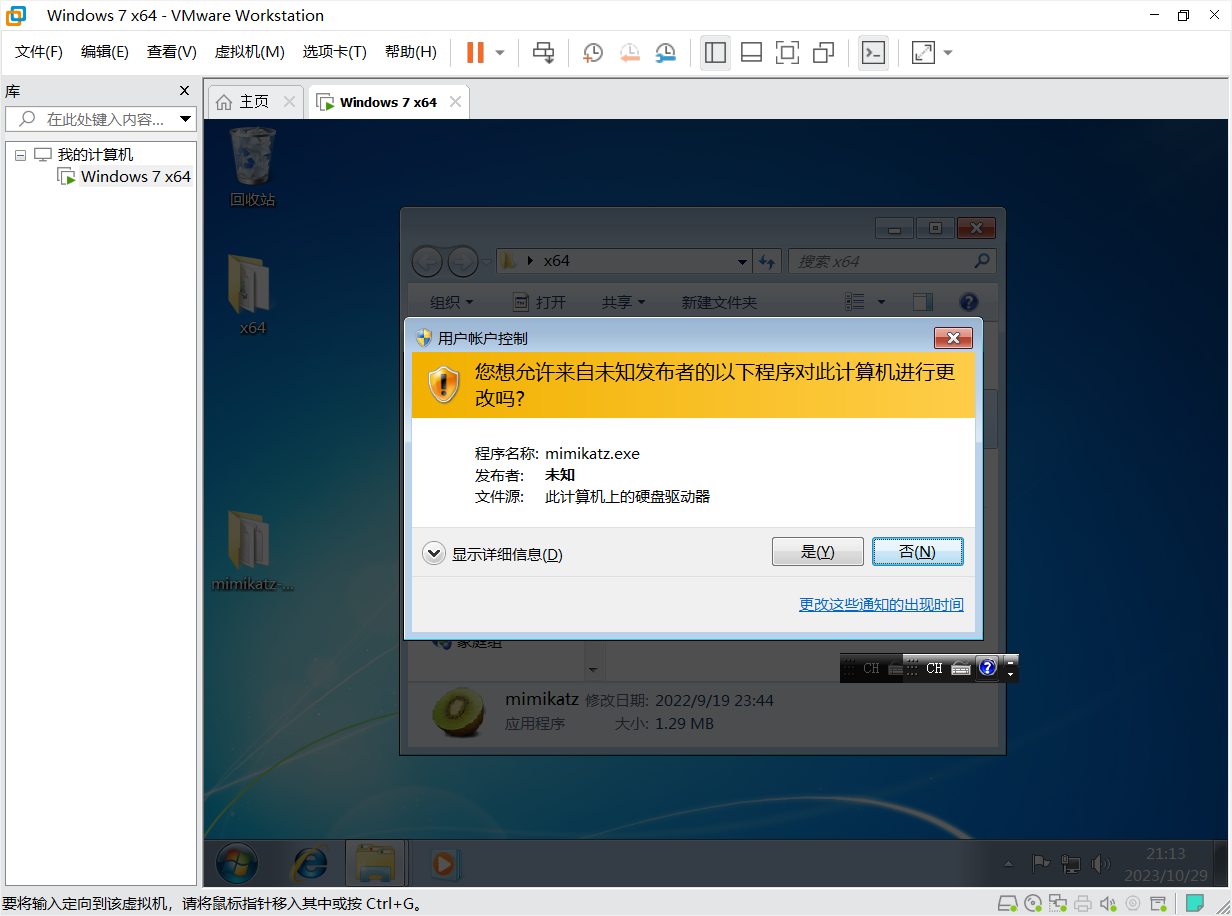
## 4. Windows密码获取实验

### 4.1 使用Mimikatz尝试获取Windows系统缓存的用户登录密码

（1）下载Mimikatz，并且移动到win7虚拟机中

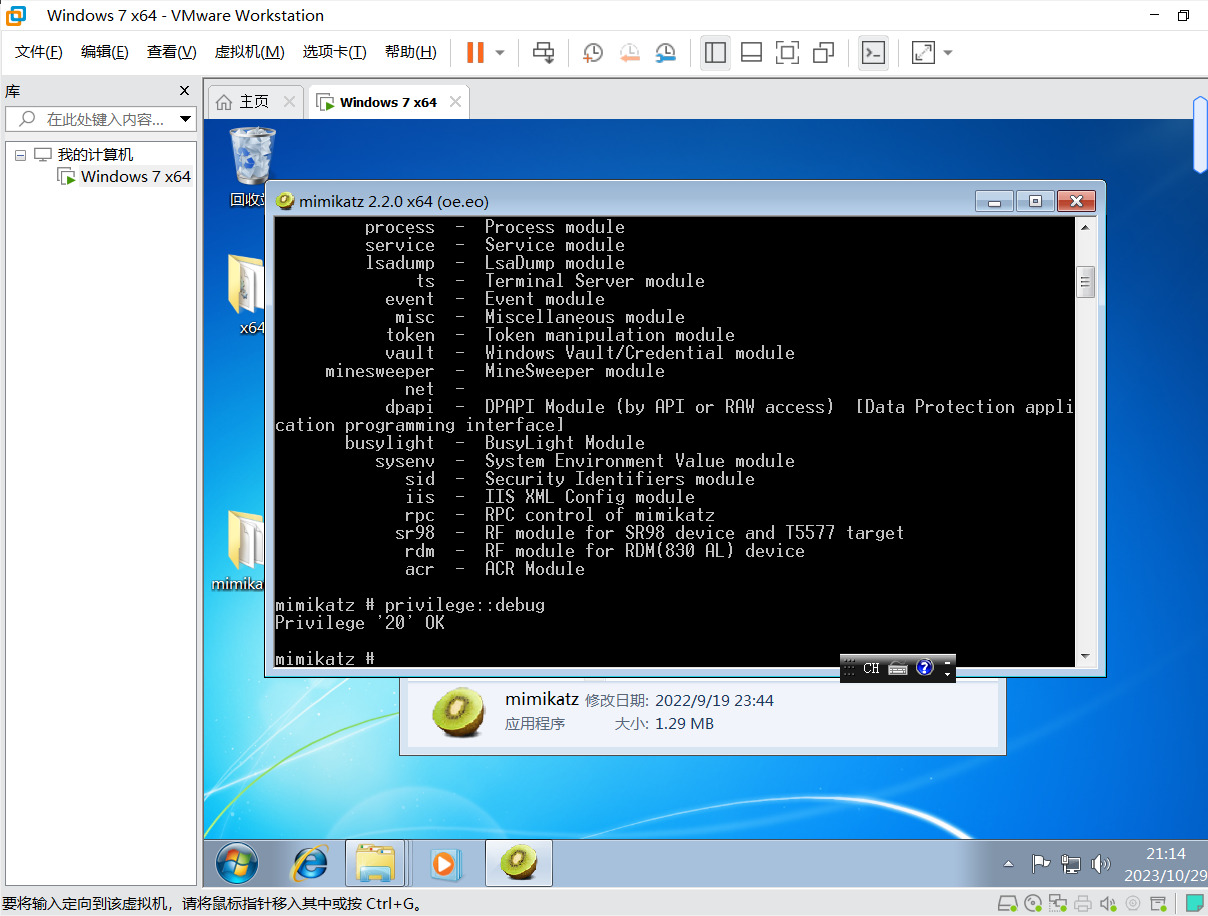


（2）右键通过管理员权限执行mimikatz.exe应用程序

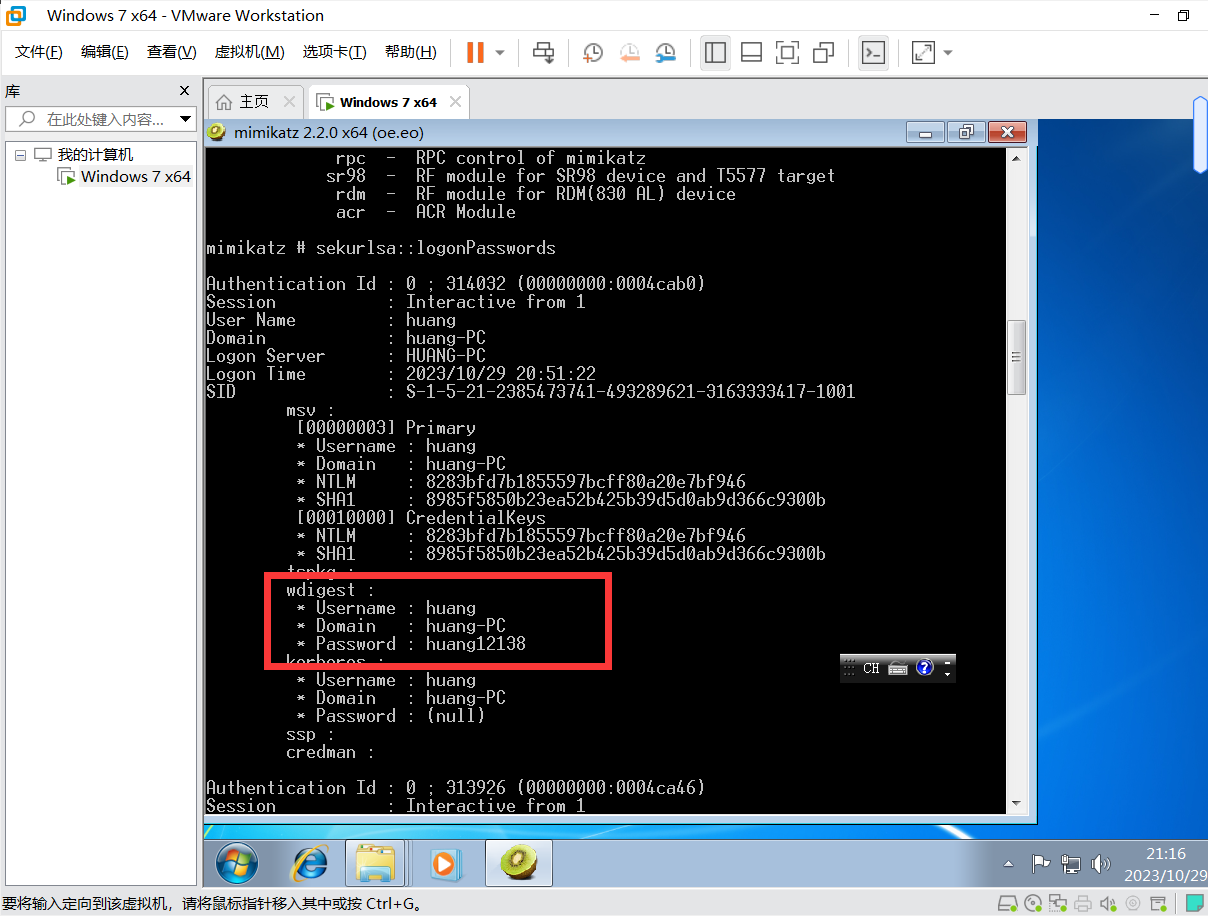


(3)使用管理员权限（system）执行以下命令

提升权限命令：privilege::debug



抓取用户密码命令：sekurlsa::logonPasswords



结果分析：成功获取用户名为：huang，登录密码为：huang12138

### 4.2简述该软件获取登录密码的原理

（1）实现方式

LSA Secrets: Mimikatz利用 Windows LSA（Local Security Authority）子系统中存储的密码散列来获取密码。这些散列通常用于存储本地计算机上的密码，包括本地用户账户和域账户的密码。Mimikatz可以提取这些密码散列，然后尝试对其进行离线破解。

Pass-the-Hash (PTH): Mimikatz 可以利用 Windows 中的 NTLM 散列（NTLM hash）来实施“传递哈希”攻击。这意味着它可以使用捕获到的密码散列，而不是明文密码，来进行身份验证。这样，攻击者可以绕过明文密码的需求，以获取访问权限。

Kerberos Tickets: 当用户登录到 Windows 系统时，他们通常会收到 Kerberos 认证票据，以便在会话期间进行身份验证。Mimikatz可以劫持这些票据，并在之后的时间内使用它们进行访问，而无需明文密码。

Cached Credentials: Windows 系统通常会缓存用户的登录凭据，以允许脱机登录。Mimikatz可以提取这些缓存的凭据，然后进行离线攻击，尝试破解密码。

（2）基本原理

Windows凭据管理系统：Windows操作系统使用Local Security Authority Subsystem Service (LSASS)来处理用户认证过程。LSASS负责管理用户登录凭据，包括明文密码、哈希值和其他相关信息。这些凭据存储在LSASS进程的内存中，以便系统能够验证用户的身份。

WDigest协议：WDigest是一种Windows认证协议，它用于在系统内部处理用户的明文密码。当用户登录时，Windows系统会使用WDigest协议来将密码加密存储在LSASS进程的内存中，以备稍后使用。

AES/3DES加密：WDigest协议使用AES（高级加密标准）或3DES（三重数据加密标准）等加密算法来加密用户密码。这些加密算法用于将密码变成密文，以便在内存中存储。

Mimikatz的逆向工程：Mimikatz的作者通过逆向工程技术，定位到LSASS进程内存中的密文密码以及解密所需的密钥。Mimikatz使用这些信息来提取和解密存储在内存中的用户密码。

总结来说，Mimikatz获取Windows密码的原理是通过逆向工程LSASS进程内存中的密文密码和相关密钥，然后使用这些信息来解密存储在内存中的用户密码。这使得Mimikatz可以成功提取用户的明文密码或密码哈希，这是一种潜在的安全风险，因此系统管理员和安全专家需要采取适当的措施来减轻Mimikatz等工具的威胁，如定期更新密码、限制对LSASS内存的访问、启用Windows Defender Credential Guard等。