IMPLEMENTATION OF AN ARP SPOOFING ATTACK USING THE NDIS DRIVER

Valeriu IONESCU[1], Adrian ZAFIU[1, 2]
[1]University of Pitesti, Romania
[2]Research Institute for Artificial Intelligence, Romanian Academy
[1]valeriu.ionescu@upit.ro, [2]azafiu@racai.ro

Keywords: ARP spoofing, NDIS driver, C#

Abstract: *The implementation of an Address Resolution Protocol (ARP) spoofing attack is a matter for concern in local networks, even if it is a known threat for a long time. The Network Driver Interface Specification (NDIS) API allows sending frames formatted in any way the user intended. This paper presents an ARP spoofing attack using the NDIS driver in promiscuous mode for network information query and frame sending using the C# language. The local network discovery stage is implemented and the protection of firewall solutions is investigated.*

1. INTRODUCTION

Network security is a current concern for any organization. The most important attacks come from the exterior of the network; however, there are many attacks just as damaging that can take place on the local network.

The Address Resolution Protocol (ARP) is used in local networks when a computer knows an IP address and tries to find the corresponding MAC address. This is necessary when a static MAC table is not kept by the network administrator, because the network is very large or keeps constantly changing.

ARP spoofing attacks are one of the many threats to the local network. They make the local computer think that is talking to a different computer in the local network. The purpose can vary from Denial of Service to various Man in the Middle attacks that target either all or special purpose equipment (such as gateways).

In order to implement an ARP spoofing attack the structure of the ARP communication is used. ARP has two components: an ARP request and an ARP reply. The ARP request is sent when the MAC address is not known. If an attacker sends an ARP reply that has an altered source MAC address, the destination will form new frames based on the received MAC. Layer 2 equipment, such as switches, help in this process by associating the modified MAC address with the port that it came on. Any further frames that will come to the switch will be sent to the attacker because decisions are based on the read altered MAC address, and not on layer 3 addresses (IP) which are still the correct ones. The traffic can be directed by the attacker as they see fit, the simplest being to ignore it resulting in a Denial of Service attack as seen in Figure 1.

Current ARP spoofing security solutions can be grouped in two categories:

- Solutions for small networks. Because of the use of mini-switches, the use of static IP addresses can be complemented with the use of static ARP entries for each computer.
- Solutions for large networks. The more expensive switching solutions should have the option to associate MAC addresses with switch physical ports, or, because there is usually a dedicated network administrator, a network monitoring tool can be used that warns about unusual ARP traffic.

Both are good solutions to the ARP spoofing but can be hard to implement, that is why many network administrators avoid them altogether.

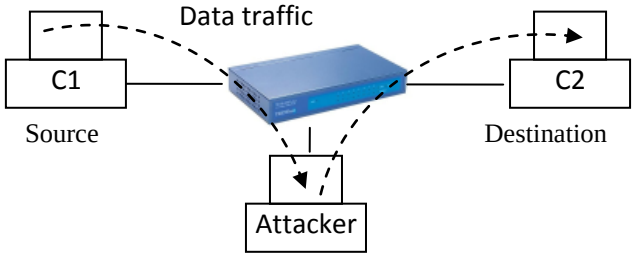


Fig. 1. Traffic path in a switched network after an ARP spoofing Man in the Middle attack

This paper presents the implementation of an ARP spoofing attack using C# because it's an easy to use language. The problems come from the fact that .NET does not have functions that allow full frame modification (to prevent spoofing tool development), and the Windows native functions that will be used for driver access are non-managed.

2. NDIS DRIVER

At the moment there are many tools that can implement an ARP spoofing attack such as: ARP-FILLUP, ARP0c or ArpSpyX [1,2] for all operating systems ranging from Windows , Linux to Mac OS.

For such implementations there are several tools available such as: SharpPcap or Pcap.net wrapper framework based on WinPcap [3] that provides its own networking driver. In Linux there is an even simpler solution by using the Raw Ethernet programming [4]

However for the target Windows environment, the Network Driver Interface Specification (NDIS) driver was used the in order to create the modified frames.

NDIS is an application programming interface (API) for network cards which provides a library of functions that can be used by MAC drivers as well as higher level protocol drivers in order simplify the development of both MAC and protocol drivers easier by hiding the underlying complexity.

NDIS had a continuous evolution, starting with version 2.0 found in MS-DOS, to version 6.2 found in Windows 7, while improving its capabilities (such as problematic Wi-Fi traffic monitoring [5,6]).

The NDIS driver requires modification in order to be able to send frames that had the source MAC address different from the current interface address because it was especially designed to avoid entering Promiscuous mode.[7]

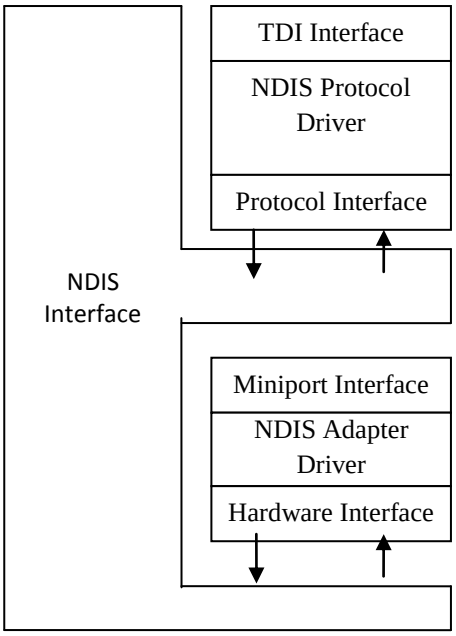


Fig. 2. The NDIS interfacing ranges from "Media Access Controller" (MAC) device driver to higher level protocol drivers [8][9]

3. SOFTWARE IMPLEMENTATION

There are many methods that allow sniffing in a switched network. This application allows both computer targeted attacks (where a single computer is attacked) and switch overflow attacks (where no computer in particular is targeted, but the switch which reads the traffic).

In the second situation the switch is flooded with false MACs that overflow the switch's port association table. In this case the switch can behave like a hub and send traffic on all its ports, traffic that can be sniffed.

The first component of the implementation is an ARP scanner of the network using the SendARP function, imported from the Windows iphlpapi.dll:

```
[DllImport("iphlpapi.dll", ExactSpelling = true)]
public static extern int SendARP(int DestIP, int SrcIP, byte[] pMacAddr, ref uint PhyAddrLen);
```

By using the function output, it can be determined if a computer has replied or not. The PingReply C# method can be easily firewall blocked so it is not that reliable. This allows us to create a list of the locally connected computers that will be available as targets.

Once the computer list is created this way it is necessary to access to the NDIS API information. This is made using the Windows methods from the kernel32 DLL. There is a good example on how to access and use NDIS in the Microsoft Driver Development Kit NDISPROT sample driver. Accordingly, the kernel32 library functions were imported in the C# application using the DllImport and the unsafe modifier in the method declaration [10] as shown below:

```
[DllImport("kernel32", SetLastError = true)]
private static extern IntPtr CreateFile(
        string _lpFileName,
        uint _dwDesiredAccess,
        uint _dwShareMode,
        uint _lpSecurityAttributes,
        uint _dwCreationDisposition,
        uint _dwFlagsAndAttributes,
        uint _hTemplateFile);
```

The first step is to get information about the NDIS status.

The function used to send a control code directly to a specified device driver is DeviceIoControl, and it causes the corresponding device to perform the corresponding operation. To retrieve a handle to the device, the CreateFile function was called with the name of the driver associated with a device. DeviceIoControl can accept a handle to a specific device. To specify the NDIS driver the following format was used: \\\\.\\\\NdisProt.

For this application the following control codes were used:

- IOCTL_NDISPROT_QUERY_BINDING = 0x12C80C in order to get information on the network cards present on the system;
- IOCTL_NDISPROT_OPEN_DEVICE = 0x12C800 in order to associate the network card with the handle opened (based on the data obtained with the previous query) with CreateFile.

The data read from the network interface uses the ReadFile function and the write of data to the network interface uses WriteFile. In the end, in order to close the driver the CloseHandle function was called.

The application structure is presented in Figure 3.

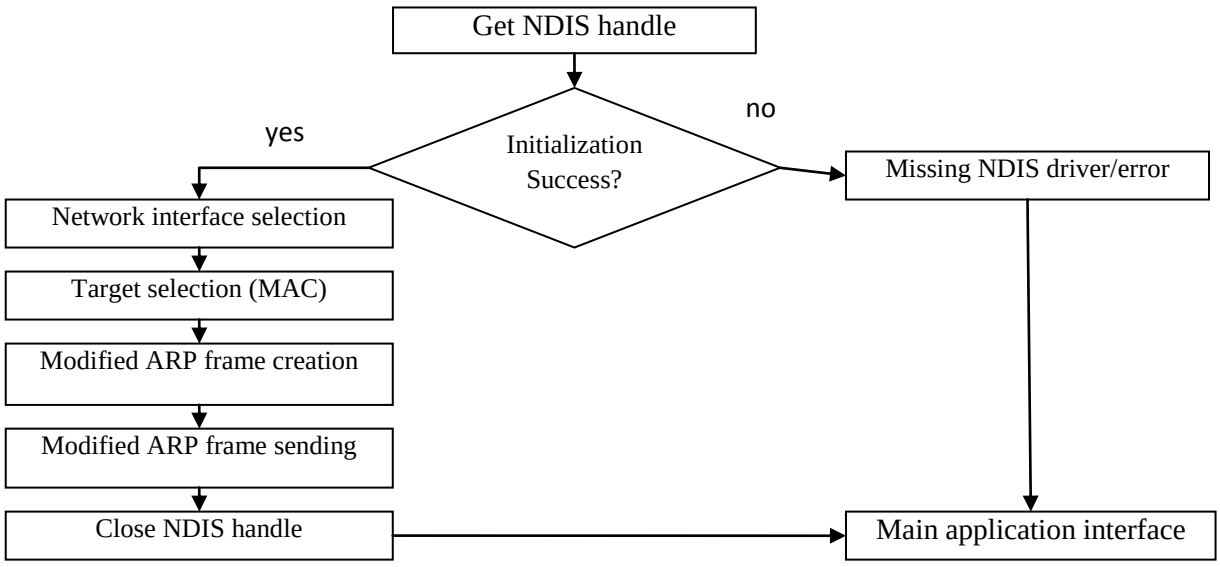


Fig. 3. Application architecture for modified ARP frame creation

Using the information gathered from the ARP scanner method, the target MAC is filled in and the user can choose only the erronous source MAC address that will be used and learned by the switch.

The two stages of the NDIS access are presented as follows:

- the NDIS information step that allows the selection of a valid network card from the multiple network interfaces existent on user's computer;
- the NDIS access for data writing, where the user has chosen the target device.

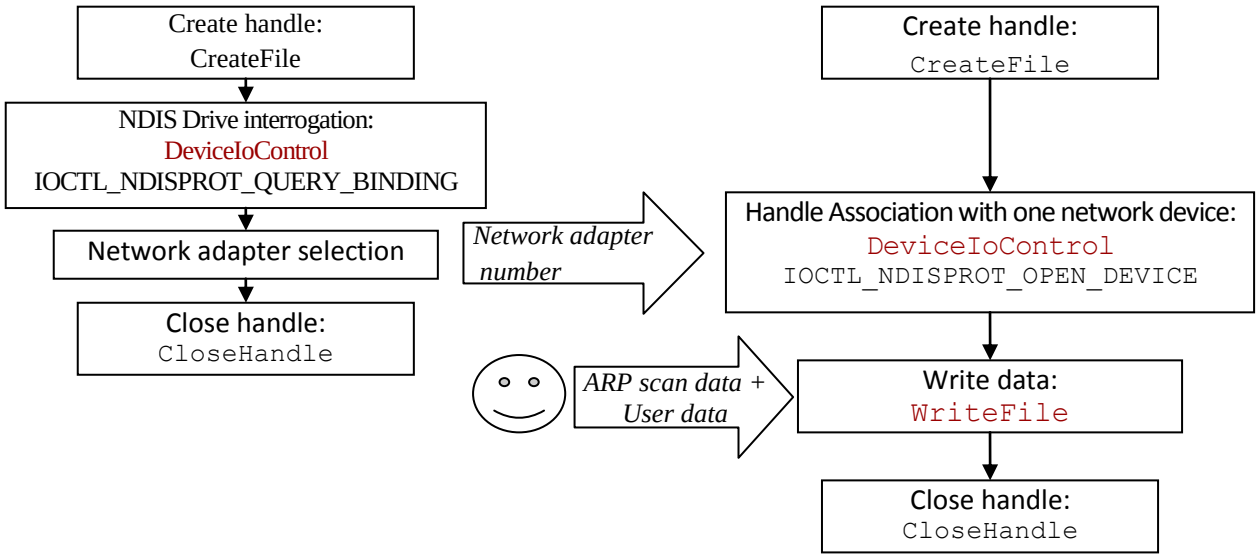


Fig. 4. The Network adapter selection and access in NDIS

The structure used for ARP reply frame creation is presented as follows:

```
public struct ARPreply  {
      public byte[] DestionationMac;
      public byte[] SourceMac;
      public byte[] DateARP1; //includes protocol ARP code, MAC address length, IP address IP length
      public byte[] sourceIP;
      public byte[] destionationIP;
                              }
```

In this structure, the DestionationMac vector was obtained as a result of target selection on the ARP scan process, and the SourceMac vector was read from the used dialog box. The IP addresses can be entered by the user and have the correct implicit values.

There is no other computation necessary in the frame creation process as all other necessary information has fixed values (such as ARP operation code 2 because it is an ARP reply).

4. TESTING AND RESULTS

The application was tested in a local network and was able to implement the proposed ARP spoofing attack. The monitoring was made using the Wireshark software.

In the following figure presents the network discovery stage is in progress.

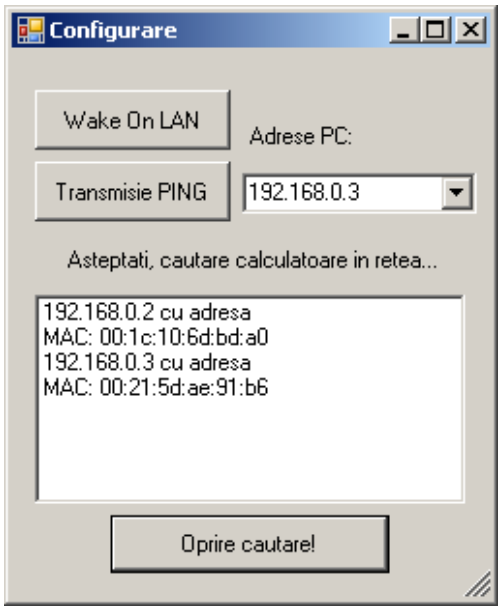


Fig 5. Network discovery stage using SendARP

When the user selects a MAC address, the following screen is opened that allows for the two types of attacks: a one time attack or continuous (flooding) attack when checkbox is selected.

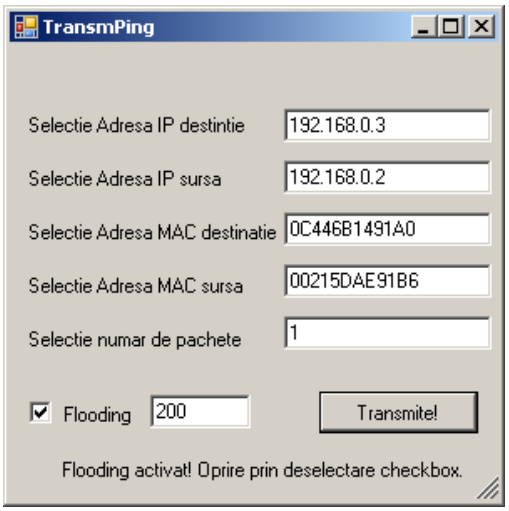


Fig 6. Interface for entering user data and selecting the attack method

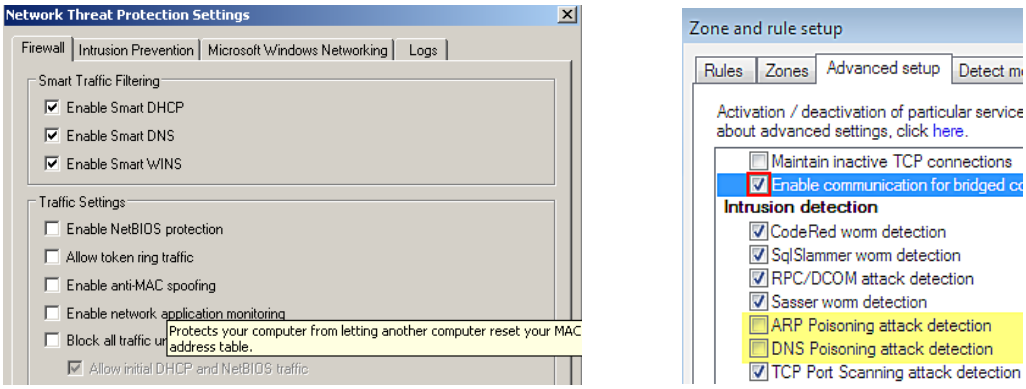
Various protection methods were tested in order to analyze the attack detection capabilities starting with the most common method is that is the firewall. There are many firewall solutions, but only few have explicit ARP spoofing protection, and even in that case this is limited to blocking unsolicited ARP replies (but allow ARP replies to legitimate ARP requests). Because ARP usage is required very often in a network that doesn't have static ARP entries, it was possible to send after several attempts a modified ARP frame as reply to a valid ARP request. Figure 7 presents two of these firewall solutions with ARP spoofing prevention methods.

Also, because the ARP spoofing process can target any of the following devices: user computer, switch and router, the success of a hardware ARP spoofing on any of them compromises the whole transmission chain. [11]

In this case there is no protection from receiving a modified ARP reply triggered by a legitimate ARP request.

In case other solutions for network security are impossible or impractical to implement, it is recommended to use tools like Arpwatch [12] or XArp [13] that can warn network administrators about the potential danger or ArpON that can apply policies for ARP table cleanup because there may be poisoned entries from the ARP spoofing attacks [14].

While tested, XArp was successfully able to detect the ARP poisoning attack and the corrupted network caches were detected.



(a) (b)

Fig. 7. ARP spoofing protection is limited and present in only few personal firewall solutions. Here (a) – Symantec Endpoint Protection and (b) – ESET Smart Security)

5. CONCLUSIONS

This paper presented a C# application that implements an ARP spoofing attack. ARP spoofing is still a powerful attack yet the attacker must gain access to the local network in order to be able to find and send the required information.

The NDIS API was used to successfully interrogate and send modified ARP frames.

In application testing it was found that most current personal protection solutions

offer ARP spoofing detection capability, but only few offer active protection measures.

Further software development of this software module will include frame monitoring and sorting and higher level attack implementation (such as erroneous SIP message construction).

6. REFERENCES

[1]. ***, "ARP0c connection interceptor", http://www.phenoelit-us.org/arpoc/, Accessed on: August 18, 2010

[2]. Allen Porter, "ArpSpyX- Monitor arp packets", http://thebends.org/~allen/arpspyx/, Accessed on: August 18, 2010.

[3]. CACE Technologies, "The WinPcap manual and tutorial for WinPcap 4.1.2", http://www.winpcap.org/docs/docs_412/html/main.html, Accessed on: August 18, 2010

[4]. Andreas Schaufler, "RAW ethernet programming",http://aschauf.landshut.org/fh/linux/udp_vs_raw/index.html, Accessed on: August 18, 2010

[5]. Printing Communications Assoc., Inc. (PCAUSA), "Native Wi-Fi Operation Traces", http://www.ndis.com/ndis-ndis6/default.htm, Accessed on: August 18, 2010

[6]. Microsoft Corporation, "Mobile Broadband Changes for Windows 7", December 10, 2009, http://www.microsoft.com/whdc/connect/MB/MBChangesWin7.mspx, Accessed on: August 18, 2010

[7]. Microsoft Corporation, "Considerations For Improving NDIS Driver Performance",http://msdn.microsoft.com/en-us/library/aa447432.aspx, Accessed on: August 18, 2010

[8]. Microsoft Corporation, "NDIS Miniport Drivers",http://msdn.microsoft.com/en-us/library/ff557065(v=VS.85).aspx, Accessed on: August 18, 2010.

[9]. Printing Communications Assoc., Inc. (PCAUSA), "NDIS.com Resources", http://www.ndis.com/resources/default.htm, Accessed on: August 18, 2010

[10]. Microsoft Corporation, "SafeFileHandle Class", http://msdn.microsoft.com/en-us/library/microsoft.win32.safehandles.safefilehandle(VS.85).aspx, Accessed on: August 18, 2010.

[11]. Sean Whalen, „An Introduction to ARP Spoofing",http://www.rootsecure.net/content/downloads/pdf/arp_spoofing_intro.pdf , April, 2001

[12]. Network Research Group, "arpwatch", Lawrence Berkeley National Laboratory, http://www-nrg.ee.lbl.gov, Accessed on: August 18, 2010

[13]. Christoph P. Mayer, XArp, "XArp - Advanced ARP Spoofing Detection", http://www.chrismc.de/development/xarp/index.html, Accessed on: August 18, 2010.

[14]. Andrea Di Pasquale, "SARPI - Static Arp Inspection",http://arpon.sourceforge.net/algorithms.html, Accessed on: August 18, 2010