An Introduction to Virtual Isolated Networks

Jiann-Ching Liu
Computer Center, National Central University, Jhong-Li, Taiwan, R.O.C.
Email: center5@cc.ncu.edu.tw

摘要

虛擬隔離網路 (Virtual Isolated Networks), 簡稱 *VIN*, 是在區域網路上提供虛擬隔離的連線, 讓使用這個連線的電腦能在一個安全的虛擬隔離環境下, 使用虛擬隔離網路所提供的網路服務。

在這篇論文中, 除了導入虛擬隔離網路的概念外, 也提供其實作的方式及應用。在實作上, 將介紹虛擬隔離網路底層的網路通訊協定的需求及連通至網際網路閘道的設計；在應用上, 我們以解決一個有安全漏洞的 Windows 系統如何透過虛擬隔離網路的服務, 在一個實體不安全的區域網路下進行視窗線上更新的應用, 這個實例說明了虛擬隔離網路的實用價值。

關鍵詞： 虛擬隔離網路, *VIN*,
Virtual Isolated Networks

Abstract

Virtual Isolated Networks (*VIN*) provide safe, virtually isolated network links between each *VIN* client and specialized *VIN* gateway on top of traditional local area networks, in which *VIN* clients can use network services on *VIN* safely.

This essay introduces newly developing concepts of *VIN*, its implementation details and applications. The implementation of *VIN* requires an underlying network protocol and specialized *VIN* gateways; all these details will be discussed in this article. Also, we demonstrate how vulnerable *Windows XP* systems can be safely updated online via *VIN*, even if they are attached to non-safe local area networks.

Keywords: *VIN*, Virtual Isolated Networks

1. Introduction and Motivation

These days, it seems like new security holes on *Windows* systems are being found by computer hackers every day; regularly updating your *Windows* systems to prevent them from being influenced by computer viruses, Internet worms and hackers, etc. becomes more and more important every day. However, windows updates are much more convenient via directly connecting to network updates online rather than pre-downloading several patch files and installing them offline. Therefore, most computer users try to apply patches to their systems by directly connecting to the Internet to updating online. Ironically, using vulnerable systems to update online allows these systems to get intruded upon by computer viruses or Internet worms very quickly.

If you are a well trained engineer, you know how to avoid online update problems, of course, by pre-downloading patch files, installing software to protect your computer in advance, putting your computer under the protection of a firewall or IDP, etc. The problems become unmanageable if you need to service hundreds of users and computers. How do you help them to deal with their own problems? To solve these problems, we propose the concepts of *Virtual Isolated Networks* (*VIN*), and hope them will solve the problems.

The concepts behind *VIN* are to provide virtually isolated network links between individual networks users and *VIN* gateways. Each *VIN* client is virtually separated by other computers on the same LAN and is not possibly have real connection between other *VIN* clients either. This ensures *VIN* clients don't get influenced by any other computers; all network connections between *VIN* clients and the outside world must pass through a *VIN* gateway. A network manager for *VIN* only has to make sure the *VIN* gateway is set properly; every individual user can get a safety link via *VIN*. Therefore, with *VIN*, the management cost is minimized for network managers; those who aren't familiar with computers and networks security will also be happy using *VIN*, which provides them with a safe way of updating their *Windows* systems online conveniently.

2. Requirements and Design Issues

Figure 1 shows the simple concepts of the *VIN*; as the figure shows, A, B, C and D are computers attached to a local area network. Computers A and B are normal LAN computers; they are connected to each other and can access the Internet freely. However, Computers C and D are on the *VIN*; although it seems like they are connected to the same LAN as computers A and B, they cannot connect to other computers without passing through a *VIN* gateway. Therefore, Computers C and D are considered isolated from the local area networks.

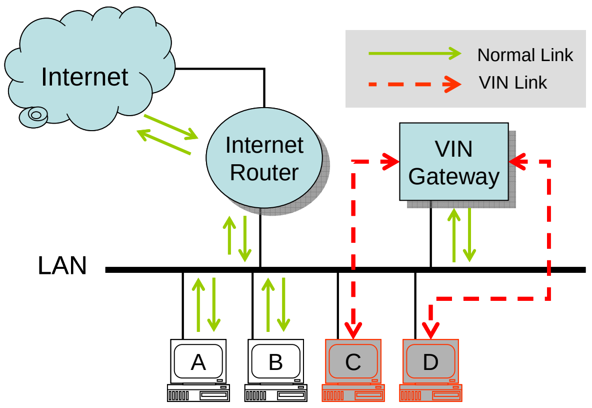


Figure 1. Virtual Isolated Networks

To achieve the goal of isolating each computer from the others in the LAN through use of *VIN*, the following conditions are required:

1. The *VIN* underlying network protocol must be a LAN-based, point-to-point TCP/IP connection. It is better if it is an existing protocol, or if it can be newly developed in the event that a suitable protocol cannot be found.
2. *VIN* must provide a specialized gateway, which ensures that each *VIN* is separate from the others, and allows managers to easily define their own policies within it.
3. The *VIN* protocol should have a client side authentication mechanism to ensure that the user's identity is valid.
4. The *VIN* protocol should have a server side authentication mechanism to avoid *VIN* clients being cheated by fake servers.

Fortunately, it hasn't taken a lot of effort that PPP over Ethernet (PPPoE) [1] is the protocol of choice; it is a well developed protocol and is already built-in on *Windows XP* systems; it meets the requirements of 1 and 3. Therefore, the focus of the remainder of this article will be how to build a *VIN* gateway to make the whole system work.

3. Implementation

Since PPPoE has been chosen as *VIN* underlying protocol, a *VIN* gateway must provide a PPPoE server in it. To achieve this goal, we choose *Linux* systems plus *rp-pppoe* software to build our *VIN* gateways. The following procedures briefly describe its installation steps:

1. Find a computer and install a *Linux* system in it. Make sure PPP [2] is supported by your *Linux* kernel or kernel module.
2. Download *rp-pppoe* package from the following URL and install to the system. *http://www.roaringpengiun.com/pppoe/*
3. Set an `/etc/ppp/pppoe-server-options` file, such as

```
lcp-echo-interval 10
lcp-echo-failure 2
ms-dns 10.0.0.1
```

After all these steps had been completed, the basic requirements of the *VIN* gateway can be easily started by issuing the following command:

```
pppoe-server –I eth0
```

There are two strategies to choose from on *VIN*, The first is proxy-based, and second is NAT-based. For *VIN* clients, a proxy-based strategy is very limited in the way that it accesses the network resources, and requires users to explicitly set the proxy-server on their Internet browser or other proxieable Internet applications. On the contrary, NAT-based *VIN* is more flexible and relatively transparent to *VIN* clients. For different strategies, the setting procedures are different, but the goals are the same: to isolate each *ppp* interface and to allow them to access the Internet within the paradigms set by the manager.

3.1 Proxy-based *VIN*

For a proxy-based *VIN*, Internet access for clients is limited by sending their requests to *VIN* gateways. On *VIN* gateways, a proxy-server, for example *squid*, is required to serve the requests from the *VIN*.

The steps for setting a proxy-based *VIN* gateway include the following additional steps:

1. Download and install *squid* on the *VIN* gateway.
2. Change *squid.conf* to make sure *VIN* access to the *squid* server can be allowed, for example:

```
acl localnet  src 10.0.0.0/255.0.0.0
```

3. Remember not to turn on the IP-forwarding option on a proxy-based *VIN* gateway.

Since IP-forwarding should be turned off on a proxy-based *VIN*, each *VIN* is considered isolated from the others; it therefore meets the desired requirements.

3.2 NAT-based *VIN*

For a NAT-based *VIN*, Internet access is done by network address translation [3] technology with an IP forwarding option turned on. To meet the *VIN* requirements, certain access controls must be enforced on *VIN* gateways. Within the *Linux* platform, *IP Netfilter* can deal with NAT functionally, and limit access control as well; proper setting of *IP Netfilter* is a key issue on a NAT-based *VIN* gateway.

The additional steps for setting NAT-based *VIN* on *VIN* gateways require a long *shell script* to get *IP Netfilter* working properly; please refer to the *Appendix* at the end of the essay to see the complete script.

The script assumes *eth0* as an external interface connecting to normal LAN; *ppp+* ("+" is a wildcard character) represents each *VIN* connection. Lines 1 to 21 are basic settings to initialize *IP Netfilter*. Lines 23 and 24 allow those networks in which traffic has already had been established to pass through *VIN* gateways. Lines 26 and 27 allow all traffic initiated by *VIN* clients to perform DNS queries or access proxy servers. Lines 33, 34, 42 and 43 masquerade IP packets sent to remote port 80 (http) and port 443 (https). Under this simple script, *VIN* clients are allowed to access DNS and proxy servers directly and browse web pages by network address translation.

3.3 *VIN* on multiple LANs

If your networks are formed by several VLANs, do you need more than one *VIN* gateway on each VLAN? The answer is yes and no. The answer is yes because if you have more than one VLAN, it means that you have larger networks environments, (for capacity and performance issues) you do require more *VIN* gateways to satisfy the environments; the answer is no because a single *VIN* gateway can serve more than one VLAN simultaneously.

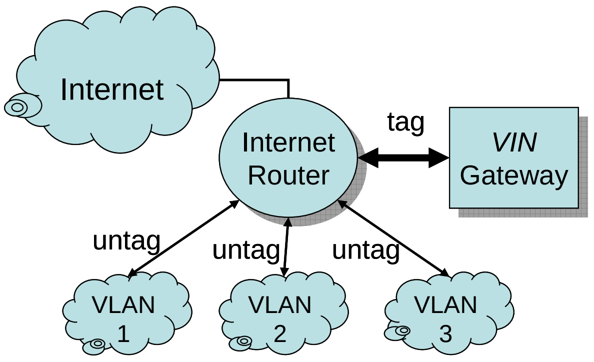


Figure 2. *VIN* gateway for multiple VLAN

You can install several network interface cards (NIC). For example, you can attach a NIC to each VLAN you have on your *VIN* gateway. You can also attach single NICs to a switch with a VLAN enable tag. The former solution is quite straightforward and the latter solution is quite simple. Figure 2 demonstrates how a *VIN* gateway can serve multiple VLANs.

The following steps show a way that a single NIC *VIN* gateway can service multiple VLANs:

1. Turn on 802.1q tag VLAN option on the port of *Ethernet switch/router* which your *VIN* gateway is directly connected with, and allow all the VLANs you want to pass through the port of the *Ethernet switch/router* by tagging.
2. Use *vconfig* command on your *VIN* gateway to make your network interface, say eth0, into several logical network interfaces, each logical interface representing each VLAN, of course.
3. Allow your *pppoe-server* program to listen on these logical network interfaces by issuing a command with several –I options, with each –I option followed by a VLAN interface name, to enable every VLAN interface you have.

3.4 *VIN* setting on client side

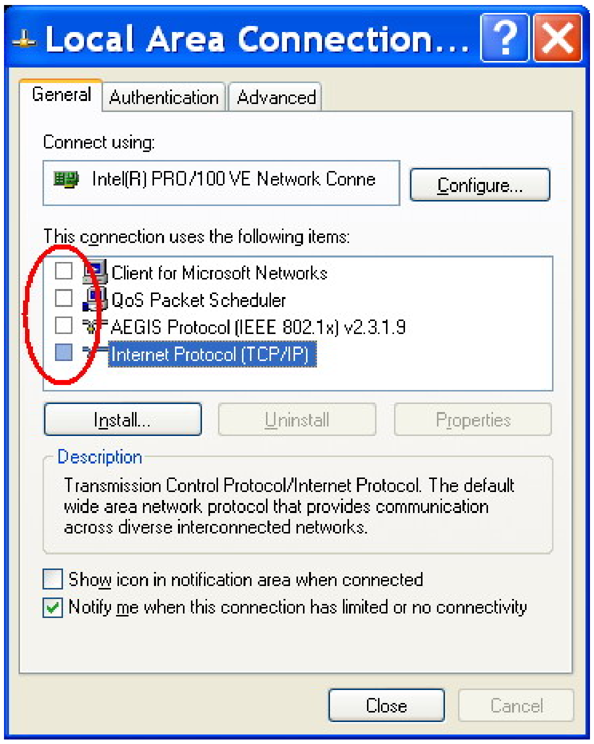
One of the main goals of providing the *VIN* is to reduce the management cost; it is strongly recommended that the *VIN* settings be as simple as possible. If users can do the setting themselves, it will make the *VIN* more popular. Therefore, the complexity of setting procedures is one of the key issues to make a success of *VIN*.

Since *PPPoE* is the underlying protocol for *VIN* on current implementation, the *Windows XP* systems and its later versions of *Microsoft* Windows-based systems can be easily adjusted to connect to *VIN*. For earlier versions of Windows systems such as *Windows 2000/98* may require additional software programs and more steps to use *VIN* correctly. In this essay, only directions configuring a *PPPoE* connection on *Windows XP* systems:

1. Open the **Control Panel**. Click the **Network and Internet Connection** option, then choose **Network Connections**. ***Note:*** *If the Control Panel is in* ***Classic View****, double click the* ***Network Connections*** *icon,*
2. Click the **File** menu and choose **New Connection** to start the New Connection Wizard. Select **Connect to the Internet** and click **Next**.
3. Select **Set up my connection manually** and click **Next**.
4. Select **Connect using a broadband connection that requires a username and password**. Click Next.
5. In the **ISP Name** field, enter a name for the connection. Click **Next**.

6. Enter your **Username** and **Password** into the appropriate fields. Click **Next**.
7. If you would like an icon to the PPPoE connection added to the desktop, select that option. Click **Finish** to complete the wizard.

As mentioned above, there are two types of *VINs*. One is proxy-based *VIN*, and the other is NAT-based *VIN*. NAT-based *VIN* is transparent to its user, that is, additional settings don't need to be dealt with. On the contrary, proxy-based *VIN* requires its user to explicitly set the proxy-server on the Internet setting. Since setting a proxy server for Internet Explorer is quite usual these days, it can be considered quite straightforward.


Local Area Connection...

General | Authentication | Advanced

Connect using:
Intel(R) PRO/100 VE Network Conne

This connection uses the following items:
- [ ] Client for Microsoft Networks
- [ ] QoS Packet Scheduler
- [ ] AEGIS Protocol (IEEE 802.1x) v2.3.1.9
- [ ] Internet Protocol (TCP/IP)

Description
Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

- [ ] Show icon in notification area when connected
- [✓] Notify me when this connection has limited or no connectivity

Figure 3. Disable all protocols on physical interface

After successfully setting PPPoE on Windows systems, don't forget to turn off all protocols on the physical Ethernet interface, as shown in figure 3, before you plug a network cable in to your computer. This is a very important step; if say TCP/IP is still available on the physical Ethernet interface, it means your computer can be accessed to the outside by TCP/IP; the risk of being intruded upon still exists.

Since all the protocols have been turned off, there is no absolute way to communicate with your computer except via a *VIN* gateway. Consequently, your computer is considered to be isolated from others, even if it is actually connected on dangerous physical networks.

4. Discussion

Using *Virtual LAN* [4] technologies can isolate a computer by assigning a different VLAN id, but it requires a network administrator to do the setting on demand. In addition, different clients should have their own unique VLAN id to prevent these clients from being influenced by their neighborhoods. Obviously, this kind of approach is very inconvenient. On the contrary, *VIN* solutions allow client computers to isolate themselves anytime they desire.

Sometimes, *VIN* can be built for special purpose. For example, allowing users to update *Windows* systems online safely. One example is to restrict *VIN* clients to accessing only on "***.microsoft.com**" and "***.windowsupdate.com**." For different types of *VINs*, the approaches are different.

4.1. Access control for Proxy-based *VIN*

For proxy-based *VIN*, users access the Internet indirectly through a proxy server on the *VIN* gateway. Therefore, access control is done by the proxy server.

If *squid* has been chosen as the proxy server software; it can be set to access-list in the configuration file. Here is a setting example:

```
acl aclurl url_regex -i ^http://[-_\.a-
zA-Z0-9]*\.microsoft\.com
acl aclurl url_regex -i ^http://[-_\.a-
zA-Z0-9]*\.windowsupdate\.com
acl aclurl url_regex -i ^[-_\.a-zA-Z0-
9]*\.microsoft\.com
acl aclurl url_regex -i ^[-_\.a-zA-Z0-
9]*\.windowsupdate\.com
```

4. 2 Access control for NAT-based *VIN*

Access control of NAT-based *VIN* relies deeply on the *IP Netfilter*; it can accept IP or subnet as its access list, of course. However, in this case, FQDN partial matching cannot be done by using *IP Netfilter* alone. To achieve this goal, we need something else to do the job.

It is apparent that if *VIN* systems can understand the destination's FQDN and its corresponding IP address, *IP Netfilter* still very useful to control the network traffic. The most straightforward way to figure out the destinations' FQDNs is to let *VIN* gateways serve as DNS servers for the *VIN* client.

Instead of modifying the DNS server program, simple DNS proxy program can be written and allowed to run in the *VIN* gateways. This small program redirects DNS query requests to an external DNS server, gets the DNS query responses from the server and sends them back to the original issuers. By analyzing DNS query responses, this program will comprehend the FQDN and its corresponding IP address as well. By matching this FQDN to the administrator's pre-specified patterns, say "***.microsoft.com"**, this program determines whether it should write this IP address to an access list or not.

To enable access control on NAT-based *VIN* gateways, execute the script shown in the *Appendix* with an ACL environment variable set to "Y;" it will enable line 30 and line 46 of the script. Line 30 drop all packets with mark "10" and line 46 will mark all packets coming from interface *ppp+* on the pre-routing queue of *IP Netfilter* as "10." To control which destination addresses are accessible, simply issue the following command:

```
iptables –t mangle –A  \
      PREROUTING –i ppp+ \
      -d <ip_addr> \
      -j MARK –set-mark 0
```

Since all these packets were marked by "10" and the rule discards all packets marked "10," the command simply lets the desired destination IP marked "0," and it passes through the *IP Netfilter* without being discarded.

4.3 Comparisons

Table 1. Comparisons between *VIN's* implementations

	Proxy-based *VIN*	**NAT-based *VIN***
IP forwarding	off	on
IP-based ACL	proxy server configuration	IP Netfiliter
DNS-based ACL	proxy server configuration	IP Netfiliter + DNS proxy
Application	Limited on proxieable services	Less restriction than proxy-based *VIN*
Client setting	PPPoE setting + setting proxy server explicitly	PPPoE setting only

Table 1 shows a brief comparison between each of the *VIN* implementations. Choosing the best type of *VIN* will strongly depend on the services you want to provide. However, it is possible to build a hybrid type of *VIN*, and it will take advantage of both implementations.

5. Future Work and Conclusions

By nature, a Proxy-based *VIN* will cache web pages and downloaded data; it will save a lot of network bandwidths and can respond to clients quickly if the cache is hit. For Windows update services, this kind of feature seems like a pretty good choice. Therefore, a proxy-based *VIN* was chosen to serve dormitory networks in National Central University [5]. It has already been used for over 8 months (Since January 2005), and has been proven to work well. From students' points of view, there is a safety network they can use to update their Windows systems safely; from administrators' points of view, the online update paradox has been solved, In addition, it is a really, really low-cost maintenance solution indeed.

We propose the *VIN* concepts, and use PPPoE as its underlying protocol, PPPoE is good, but still not good enough to meet all requirements we desired. As a result of lacking server authentication mechanism under PPPoE, it is possible to build a fake *VIN* on the LAN to cheat users. As the *VIN* becomes more and more widely used someday, we may be able to make some improvements to PPPoE to make a better *VIN* for the years to come.

References

[1] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D. and R. Wheeler, "*A Method for Transmitting PPP Over Ethernet (PPPoE)*", RFC 2516, February 1999.
[2] Simpson, W., Editor, "*The Point-to-Point Protocol (PPP)*", STD 51, RFC 1661, July 1994
[3] K. Egevang, Cray Communications, P. Francis, NTT, "*The IP Network Address Translator (NAT)* ", RFC 1631, May 1994.
[4] IEEE, "*Draft Standard for Virtual Bridge Local Area Networks*" P802.1Q/D1, May 16, 1997.
[5] Please refer to https://website.cc.ncu.edu.tw/dormnet/

Appendix

Shell script to set *Netfilter* on a NAT-based Virtual Isolated Networks

```
#!/bin/sh

IPT=/sbin/iptables
INTF="eth0"
IINTF="ppp+"
# ACL=Y

for i in filter nat mangle; do
    $IPT -t $i -F
    $IPT -t $i -X
done

$IPT -t filter -P INPUT   ACCEPT
$IPT -t filter -P FORWARD ACCEPT

# Create new user chain
$IPT -t filter -N tcpin
$IPT -t filter -N accfwd

TCPIN="$IPT  -t filter -A tcpin"
ACCFWD="$IPT -t filter -A accfwd"

$TCPIN  -m state --state ESTABLISHED,RELATED -j ACCEPT
$ACCFWD -m state --state ESTABLISHED,RELATED -j ACCEPT

$TCPIN -i ${IINTF} -p tcp --dport 3128 -m state --state NEW -j ACCEPT
$TCPIN -i ${IINTF} -p udp --dport   53 -m state --state NEW -j ACCEPT

if [ ”$ACL” = ”Y” ]; then
    $ACCFWD -i $IINTF --match mark --mark 10 -j DROP
fi

$ACCFWD -p tcp -i $IINTF -o $INTF --dport   80 -j ACCEPT
$ACCFWD -p tcp -i $IINTF -o $INTF --dport  443 -j ACCEPT

$IPT -t filter -A INPUT   -j tcpin
$IPT -t filter -A FORWARD -j accfwd
$IPT -t filter -A INPUT   -j DROP
$IPT -t filter -A FORWARD -j DROP

# IP Masquerade for TCP 80/443
$IPT -t nat -A POSTROUTING -o $INTF \
       -s 10.0.0.0/8 -d 0/0 -p tcp --dport  80 -j MASQUERADE
$IPT -t nat -A POSTROUTING -o $INTF \
       -s 10.0.0.0/8 -d 0/0 -p tcp --dport 443 -j MASQUERADE

if [ ”$ACL” = ”Y” ]; then
    $IPT -t mangle -A PREROUTING -i $IINTF -j MARK --set-mark 10
fi

echo 1 > /proc/sys/net/ipv4/ip_forward
```