# On-demand secure service with virtual machines

Myungho Jung
*myungho.jung@utah.edu*
*University of Utah*

Nowadays, tens or hundreds of virtual machines are running on a host with the development of virtualzation. For security, users want to run programs on virtual machines to avoid access from others. Instead of running virtual machines at all times, it saves significant resources in the host by running virtual machines on demand. In this project, an automated system to manage virtual machines is implemented with OpenVPN. In addition to efficiency, it can be applied to larger system through scalable and flexible design.

## I. INTRODUCTION

In a virtualized environment, users can build many virtual machines on a server and need to access them on isolated network. If a host runs a large number of guest OS, it requires a lot of resources such as public IP, listening ports, and memory for a server processes. Even if it provides secure private networks, it will waste unnecessary resources as the number of users is growing.

To resolve the problem, we restricted the environment as there is only has a server with a public IP. Port forwarding may be a solution; however, there should be better approaches when it comes to flexible and scalable system. Thus, it is also assumed that users should be able to access any services without changing port numbers.

The problem can be resolved in different network layers. First, in the network layer, routing table with Network Address Translation(NAT) may be a solution. Even if a server only has an IP address, packets can be routed to designated virtual machines through NAT rules. The router changes the destination IP depending on the source IP and vice versa. In addition, a central server is necessary to authenticate users. For example, an OAUTH2 server connected to the router can change the table after authentication. Remote Authentication Dial-In User Service(RADIUS) and single sign-on(SSO) can also be a authentication server for various services. Finally, a VPN server offers securely isolated networks from outside. The problem is that users connected to a server should be on the same network even if they cannot communicate with each other. For this project, various solutions are considered and compared, and finally, an bridged VPN server integrated with OpenFlow is suggested.

## II. PROBLEM STATEMENT

As the performance of computers have improved and small virtual machines and containers are introduced, the large number of hosts are running on a machine. As a result, people started to need to operate many virtual machines on their own private network. Although it would be a solution to run more servers on a host, it is a waste of resources if a small number of connections are concurrently made. Even if a server only has a public IP and limited memory capacity, there are various solutions to implement the system. We will discuss on them in the next sections.

## III. RELATED WORKS

Gang *et al.* designed a Single Sign-On system to identify clients using IP address[**?** ]. Even if the first version was vulnerable to IP spoofing attack, they overcame the problem by authenticating and encrypting packets with certificates.

Jitsu[**?** ] is one of the related projects in Xen group. It triggers to start unikernel virtual machines when a user requests. The structure of service would be similar to this project; however, DNS server just starts a virtual machine and returns the address of it without authentication.

RADIUS is another solution to centralize authentication for different services. FreeRADIUS[**?** ] which is an open source RADIUS project has been developed. It can be applied to most Linux applications thanks to a project integrating with Pluggable Authentication Module(PAM).

Finally, VPN provides bidirectional authentication between server and client. OpenVPN[**?** ] is one of the open source projects and it provides SSL-based encryption and authentication. Therefore, the server can trust IP or MAC address of clients and also detect changes.

## IV. APPROACH

There are three approaches to implement the system. First, although each user's packets destines to the same IP address, the router in the server can modify destination and source IP addresses of packets. On top of that, an authentication server is necessary to guarantee that an attacker cannot spoof IP address. VPN also makes it possible to provision isolated network with authentication.

### A. Routing by source IP address

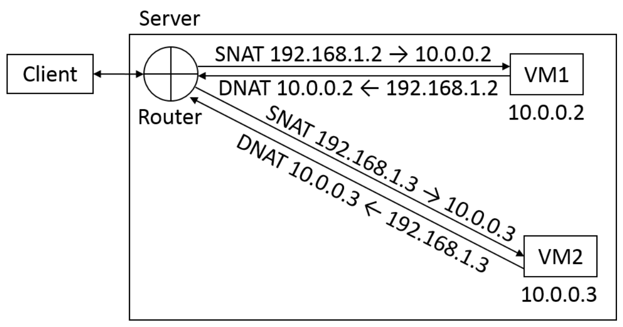
Server
Client
Router
SNAT 192.168.1.2 → 10.0.0.2
DNAT 10.0.0.2 ← 192.168.1.2
VM1
10.0.0.2
SNAT 192.168.1.3 → 10.0.0.3
DNAT 10.0.0.3 ← 192.168.1.3
VM2
10.0.0.3

FIG. 1. A router with NAT rules based on source IP address

As a naive approach, IP address can be used to identify users. It is appropriate when a server and clients are connected with LAN and fixed IP address are allocated. In the figure **??**, the NAT rules are defined in the router and it changes the source and destination IP address of incoming and outgoing packets. However, the problem is that IP address in packets is exposed and easily spoofed by attackers. Therefore, additional authentication mechanism is necessary to improve security.

### B. Routing with authentication

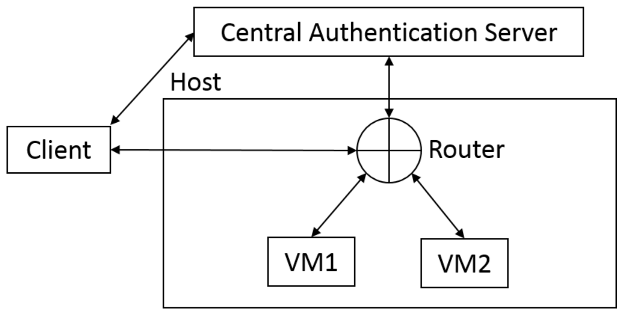

FIG. 2. A router with authentication server

Second, a central server can authenticate users and notify the router of the result. There are various solutions to implement it. First, a RADIUS server can act as a comprehensive authenticator. It helps to centralize users profiles into central database. It can be applied to various Linux applications using PAM. Similarly, Single Sign-On system can centralize the authentication of services. Though most of them are developed for web services, Kerberos[**?** ] based SSO can be used as a central login system. Similarly, OAUTH2 server works for HTTP services.

These systems may provide secure access to a virtual machine, they have common problems. First, it needs two public IP or ports for authentication and service. Otherwise, it would cause a problem when a user's IP address changes. Second, a user cannot access multiple virtual machines with a public IP. Finally, even if the authentication system exists, source and destination IP addresses on packets cannot be encrypted and it is vulnerable to man in the middle attack. Therefore, IP address should be used as only public key.

### C. Virtual Private Network

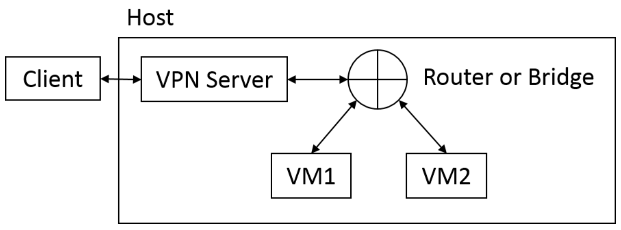

FIG. 3. A VPN server connected to virtual machines

Users can access isolated network using VPN. It provides private network with any number of virtual machines. The problem is that the amount of required memory is a multiple of users. It causes the server to sharply slow down when a large number of users concurrently access. However, a lot of memory will be saved if we can route packets in the server.

To make the idea work, first solution is attaching a VLAN tag to incoming packets and tap devices on virtual machines of a user. Then, only matching packets can pass the tap device. However, there would be a network performance issue if many users use different tags since packets should be broadcasted. Another problem is the number of possible tags is limited as 4096 because of the number of allocated bits.

It is possible to separate networks if the network traffic is controlled by OpenFlow. For this project, an automated system is implemented using OpenVPN scripts. The scripts are triggered when users connect, disconnect, and change status like MAC address. The system is efficient and secure because it only requires a small amount of fixed pre-allocated memory and a listening port, and provides isolated network for each user. Also, it simplifies the server side which makes it scalable and flexible.

## V. IMPLEMENTATION

The system is implemented using OpenVPN and tested on Ubuntu 14.04 LTS server with Xen hypervisor. Virtual machines are automatically started when users connect to VPN server. By doing so, the deployment process becomes simple, for example, new version of virtual machines are served by uploading image files on server and making a new connection. After a user's VPN tunnel is established, the user can access his or her own virtual machines. It makes users feel like their guest OS is always running on the server. Because the default action in disconnection is stopping virtual machines, some users may have a problem in executing scheduled jobs or background process. It will be dealt with in section **??**. More than a user can have the same subnet because not only is it working on network layer 2 but also user bridges are located in different network namespace, which prevents from IP confliction between users.

### A. Network Structure

After a user successfully authenticates to VPN server, virtual machines become active with connection to a bridge. The figure **??**. shows a case that three users are connected to the network. In this example, **br0** is a Open vSwitch connected to tap device of VPN server. And three bridges **br-user1**, **br-user2**, and **br-user3** are linked to the **br0** through patch ports. In the example, user 1 has access to **VM1** and **VM2**, user 2 has access to **VM3** and **VM4**, and the user 3 has access to **VM5** in the figure.

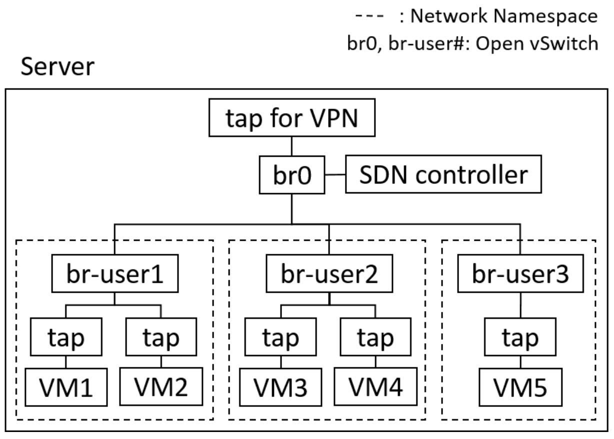

FIG. 4. An example of VPN network on server

A server and clients have a tap device and make a connection with it. Although different clients are connected to the same tap device, OpenFlows defined on **br0** pass frames to each user's bridge and thus, private network is isolated from each other. Even if configuration files are distributed to clients, it is not hard for users to change MAC address of tap device. To prevent malicious activity like crafting frames, OpenFlow[**?** ] rules should be added to the bridge **br0** on VPN server to allow only valid packets, otherwise drop them. SDN controller helps to create and manage flows and Floodlight[**?** ] which is open source controller is used in this project. When a user is authenticated by VPN server, the server starts the user's virtual machines and adds flows in network layer 2. Users can access their guest OS only after the flows are inserted because every ARP and IP packet is rejected by default. Though bridges are linked with patch ports in this project because those are on the same host, bridges in remote hosts can be connected with tunnel which will be useful in the cloud system.

### B. OpenVPN

OpenVPN supports bridged and routed mode which are working on network layer 2 and 3, respectively. In this project, bridged OpenVPN server is selected to cooperate with Open vSwitch and OpenFlow. When a user connects to VPN server, the server executes scripts to authenticate user, create bridge, and add flows. Fortunately, OpenVPN supports various plugins to authenticate with external applications. In this project, MySQL database is used for storing user profile.

Next script in Listing **??**. is triggered after a client is successfully authenticated.

Listing 1. Script for client connection

```
#!/bin/sh
# Usage: connect.sh <central bridge> <user
   bridge> <network namespace>

# Bridges
br-center = $1
br-user = $2
# Patch ports
p1 = '$br-center'-to-'$br-user'
p2 = '$br-user'-to-'$br-center'
# Network namespace
ns-user = $3

# Add patch ports
ovs-vsctl add-br $br-user
ovs-vsctl add-port $br-center $p1 \
-- set interface $p1 type=patch \
options:peer=$p2
ovs-vsctl add-port $br-user $p2 \
-- set interface $p2 type=patch \
options:peer=$p1
```

```
# Create network namespace and put bridge
ip netns add $ns-user
ip link set $br-user netns $ns-user
ip netns exec $ns-user ifconfig lo up
ip netns exec $ns-user ifconfig $br-user up

# Start virtual machines
start-vm.sh $username $br-user
# Add flows
add-openflow.sh $username $br-user
```

This script creates a new user bridge and adds patch ports to it and the central bridge to link them. After that, create new network namespace and put the user bridge into it. Even if the two bridges are in different network namespace, they can communicate with each other through the patch port. However, we want to only allow the owner to access virtual machines. To do this, the last two scripts add flows after starting the virtual machines on the bridge.

Listing 2. Script for client disconnection

```
#!/bin/sh
# Usage: disconnect.sh <central bridge> <user
     bridge>

# Bridges
br-center = $1
br-user = $2
# Patch ports
patch = ’$br-center’-to-’$br-user’

# Stop virtual machines
stop-vm.sh $username $br-user
# Remove flows
del-openflow.sh $username $br-user

# Remove bridge and ports
ovs-vsctl del-br $br-user
ovs-vsctl del-port $br-center $patch

# Remove network namespace
ip netns del $ns-user
```

When a user disconnects, the server will stop the virtual machines and remove the bridges. Although it may cause performance issue if users frequently connect and disconnect, it will save allocated memory and improve network performance by maintaining only required bridges and flows. Scripts related to virtual machines and OpenFlows will change depending on hypervisor and SDN controller and thus, those are not included in this paper.

### C. Xen hypervisor and Rump kernels

In order to manage the status of virtual machines, hypervisor is a useful tool providing a variety of APIs. Xen[**?** ] is one of the popular open source hypervisors for virtual machines and compatible with most operating systems. However, it requires much computing resources to run general virtual machines. To simulate the environment where many virtual machines are running on a host, Rump Kernel[**?** ] which is unikernel virtual machine with the NetBSD kernel is used. Although the unikernel is similar to containers like docker, it includes its own kernel like regular operating systems which makes the host secure from kernel attack. Unikernel usually consists of an application and required frameworks and thus, needs a small amount of cpu and memory. In the project, the status of rumpkernel is controlled by rumprun tool in OpenVPN script.

### D. Open vSwitch and Floodlight SDN controller

Open vSwitch[**?** ] is a virtual switch providing bridged network for virtual machines. Static Flow Pusher which is one of applications of Floodlight helps to manage flows. It provides REST API with json type request and response which makes it easy to handle data.

Since there is only a tap device on the server side, we cannot trust any information in packets except source IP address and TCP port thanks to SSL tunneling. First, we need 2 rules to drop every IP and ARP packets by default. And, 4 rules per user are required to be added to allow and route packets in both directions. The rules protect the server from attacks manipulating packets from both of VPN client and guest OS.

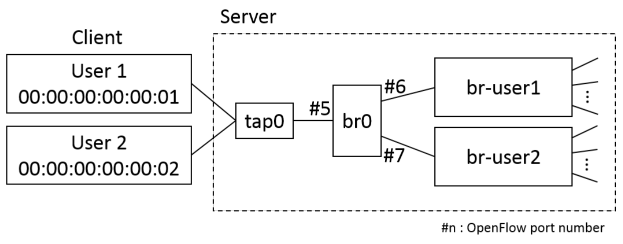

FIG. 5. An example that two users connected to a server

In the figure **??**, user 1’s packets should be passed to br-user1 bridge and user 2’s packets should go to br-user2. To accomplish it, next OpenFlow rules are required be added to **br0**.

- Default drop rules
  - priority: 1
  - match
    - protocol: IP, ARP
  - action
    - drop
- Flow for packets from user1
  - priority: 10
  - match
    - in port: 5
    - source MAC: 00:00:00:00:00:01
    - protocol: IP, ARP
  - Action
    - output: port 6
- Flow for packets to user1
  - priority: 10
  - match
    - in port: 6
    - destination MAC: 00:00:00:00:00:01
    - protocol: IP, ARP
  - Action
    - output: port 5
- Flow for packets from user2
  - priority: 10
  - match
    - in port: 5
    - source MAC: 00:00:00:00:00:02
    - protocol: IP, ARP
  - Action
    - output: port 7
- Flow for packets to user2
  - priority: 10
  - match
    - in port: 7
    - destination MAC: 00:00:00:00:00:02
    - protocol: IP, ARP
  - Action
    - output: port 5

These rules help to drop unnecessary and malicious packets and for users to securely access their network.

### E. Database

MySQL server is required for storing user data and authentication. First, user table is created at first and a user is registered by inserting data into table. The structure of the table looks like table **??** although it is simplified.

| user_id | username | password | mac_addr | ip_addr |
|---|---|---|---|---|
| 1 | user1 | ***** | 00::01 | 10.0.0.10 |
| 2 | user2 | ***** | 00::02 | 10.0.0.20 |

TABLE I. An example of users table

Username and password are necessary to authenticate users. MAC address should be updated when a user connects, disconnects, or change it. By doing so, the server can prevent from impersonating other users. Actually, IP address is not required because users can change it without being noticed by the server if it is bridged mode. However, IP spoofing attack is useless because of flow rules in network layer 2.

### F. Sharing networks between users

Although users can share networks in client side, there are advantages in connecting bridges in the server. Networks can be simply linked by making a connection between bridges by patch ports, VXLAN, or GRE tunnel. First, it is secure because users don't have to expose address of their host running client to others. And server can force to disconnect bridges whenever the owner wants. In addition, it is beneficial for network performance between virtual machines because they can communicate without passing VPN tunnel.

## VI. SECURITY

Basically, private networks must be secure from outside as well as between users thanks to SSL tunneling based on certificates and OpenFlow rules. The packet switching occurs in layer 2 network and each virtual machines are connected to bridge through tap devices. This makes it difficult to modify packets in guest OS for the malicious purpose. On tap of that, the entire network topology can be only seen and controlled by host OS.

### A. MAC address spoofing

**Algorithm 1** Learn-address script

```
if operation = add or update then
    if address already exists in user DB then
        return failure
    else
        Update the user's address
    end if
else if operation = delete then
    Change the user's address to null
end if
return success
```

Flow rules are based on source and destination MAC address and thus, attackers may try to spoof MAC address to access other user's network. However, if a user change the address of tap device, OpenVPN detects the change and rejects the packets if the address is already assigned to other user. To be specific, the **learn-address** script of OpenVPN server is triggered when a client connects or changes MAC address and it can refuse the connection by returning failure code. OpenVPN passes the operation, address, and user ID to the script as parameters. By using this, if a new MAC address of a user overlaps with one of other users connected, the server rejects the later one.

### B. Packet Crafting

On the guest OS, an attacker may try to manipulate MAC address of packets and send them to other network. However, the packets cannot pass the patch port on the central bridge according to OpenFlow rules. On top of that, users cannot access the network devices on host OS. Of course, some hypervisors had vulnerabilities escalating privileges from guest to host OS like VENOM[**?** ] before. It is already patched in most of virtualizations and also it is beyond the scope of this paper.

## VII. TEST

This program was tested on Ubuntu 14.04 server with Xen hypervisor 4.4. The overall architecture is showed in the figure **??**. Three users can access the server by VPN client. The central bridge br0 and each user's bridge, br-user1, br-user2, and br-user3, are connected with a pair of patch ports. User bridges are located in separate network namespace, ns-user1, ns-user2, and ns-user3, respectively. Internal IP is allocated to each virtual machine and they are linked with a bridge. OpenFlows are only defined in br0 to distribute frames.

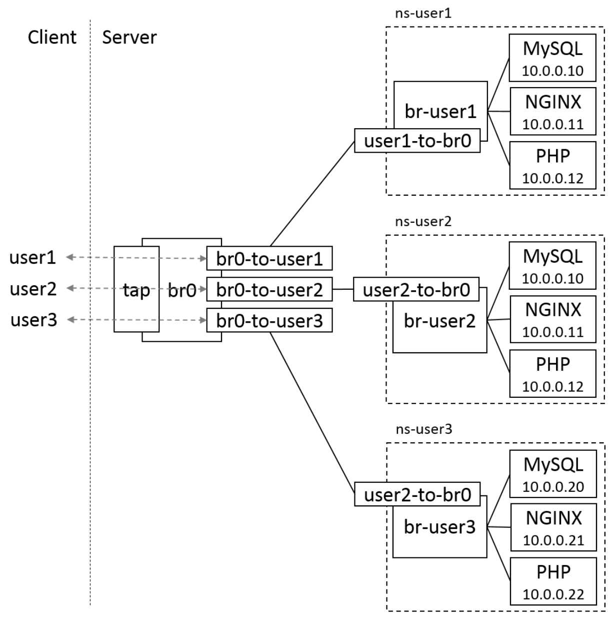

FIG. 6. Network structure in test environment

For the test, three users are registered and each user has three Rump kernel virtual machines with Wordpress. Because each user's wiki title is different, we can easily check which virtual machines a client can access. A user is allowed to access only his or her own virtual machines according to the OpenFlow rules inserted by OpenVPN scripts. As expected, it was confirmed that each users could open their Wordpress page through web browser. Although virtual machines of user1 and user2 have the same IP addresses, IP addresses doesn't conflict because packet flow is blocked in network layer 2. On top of that, each user has different network namespace which makes it passible to create individual ARP table.

MAC address spoofing attack is also simulated. It is assumed that user1 tries to mimic user3's address. If the user knows the address, it can be easily done with a command like 'sudo ifconfig tap0 hw ether <address>'. However, user1 could not access user3's network and VPN server printed error messages written in learn-address script. It indicates that address spoofing attack is not accomplished by manipulating address in packets.

In the server, user1 and user3 can share networks by connecting br-user1 and br-user3 with patch ports. After just adding ports, user3 could access user1's Wordpress and vice versa. However, user1 and user2 cannot link their bridges because IP addresses of virtual machines overlaps and it will cause IP confliction.

One of the important factor is service response time. The performance was measured by time taken to

| Status | Response Time(seconds) |
|---|---|
| First time(not cached) | 31.56 |
| After cached | 5.206 |

TABLE II. Response time from Wordpress VMs

receive a response from Wordpress server after VPN connection was established. Although the virtual machines were not cached at first booting, the booting time became shorter next time. The result is shown in table **??**. It was tested 20 times and averaged for each. The response time is acceptable on average except the case that the virtual machines were not cached. In addition, this examples need to start up three virtual machines which means it will take shorter when one or two virtual machines are booted.

## VIII. DRAWBACKS

In the project, we anticipated that users want to stop their virtual machines after VPN connection is closed. However, some of the users may like to keep running for scheduled jobs. Instead of stoping or suspending the virtual hosts, minimizing hardware resource would be better if the hypervisor supports changing the virtual hardware of running guest OS. And it will be restored to normal status when the user comes back like sleep mode in laptop PC.

## IX. FUTURE WORKS

Basically, virtual machines of each user cannot communicate each other. By connecting two bridges, two users can simply share the network. However, Xen-Cap[**?** ] can help to provide find-grained sharing between user's virtual machines. On top of that, sharing process only needs a step of granting capability to other user. By doing so, users can share any type of resources with others.

## X. CONCLUSION

Through the project, various methods to serve virtual machines with limited resources are investigated. In conclusion, VPN with OpenFlow is one of the best systems when it comes to security and scalability although routing packets by source IP with central authentication server would be better in network performance. In addition, it is showed that the system is scalable enough to be applied to the virtual private cloud system in the future. With the advent of unikernels and Microservices, this system will be helpful to save resources in virtulization on private networks. The research presented in this report was supported by the National Science Foundation under the Grants No. 1319076.

[1] Project floodlight. `http://www.projectfloodlight.org/`.

[2] Rump kernels. `http://rumpkernel.org/`.

[3] Venom. `http://venom.crowdstrike.com/`.

[4] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 37(5):164–177, 2003.

[5] Alan DeKok. Freeradius, 2008.

[6] Anil Madhavapeddy, Thomas Leonard, Magnus Skjegstad, Thomas Gazagnaire, David Sheets, Dave Scott, Richard Mortier, Amir Chaudhry, Balraj Singh, Jon Ludlam, et al. Jitsu: Just-in-time summoning of unikernels. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 559–573, 2015.

[7] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[8] Steven P Miller, B Clifford Neuman, Jeffrey I Schiller, and Jermoe H Saltzer. Kerberos authentication and authorization system. In *In Project Athena Technical Plan*. Citeseer, 1988.

[9] Yathindra Naik. Xen-cap: A capability framework for xen. 2013.

[10] Ben Pfaff and Bruce Davie. The open vswitch database management protocol. 2013.

[11] James Yonan. Openvpn, 2007.

[12] Gang Zhao, Dong Zheng, and Kefei Chen. Design of single sign-on. In *E-Commerce Technology for Dynamic E-Business, 2004. IEEE International Conference on*, pages 253–256. IEEE, 2004.