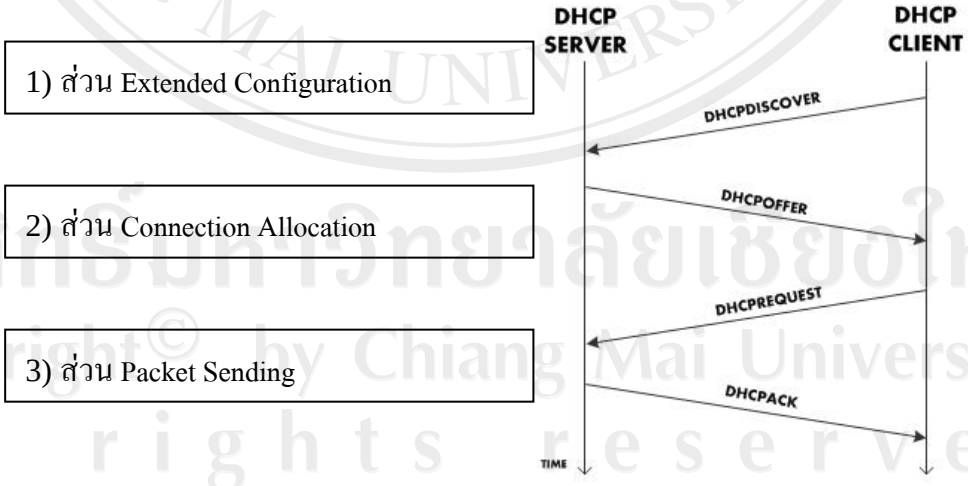
บทที่ 4
การพัฒนาการปรับปรุงการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล สำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะ

ในการพัฒนาการปรับปรุงการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล สำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะ สามารถแบ่งส่วนการพัฒนาออกเป็นสามส่วนหลัก ๆ ดังต่อไปนี้ ส่วนเพิ่มขยายในการจัดการกำหนดค่า (Extended Configuration) ทำหน้าที่ในการจัดการเรื่องการกำหนดค่าของการทำงานในส่วนของการปรับปรุง การทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลสำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะ, ส่วนการจัดการการจัดสรรการเชื่อมต่อ (Connection Allocation) ได้ทำหน้าที่ในเรื่องการจัดสรรการกำหนดค่าเครือข่ายให้แก่เครื่องลูกข่ายเพื่อปรับปรุง การทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลสำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะ, ส่วนจัดการการส่งข้อมูล (Packet Sending) ได้ทำหน้าที่ในการส่งข้อมูลที่ได้จากการจัดสรรการกำหนดค่าให้แก่เครื่องลูกข่าย และส่วนอื่น ๆ เพื่อช่วยให้การทำงานของไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลสำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะ ดังรูปที่ 4.1



รูปที่ 4.1 ส่วนในการปรับปรุงการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล เทียบกับไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลเดิม

4.1 การพัฒนาส่วนปรับปรุงการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล ส่วนเพิ่มขยายในการจัดการกำหนดค่า (Extended Configuration)

ส่วนเพิ่มขยายในการจัดการกำหนดค่า (Extended Configuration) ของการปรับปรุงระบบความปลอดภัยของไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล สำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะ ได้มีการทำงานเพิ่มเมื่อเครื่องแม่ข่ายเริ่มทำงานระบบไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล โดยเพิ่มในส่วนของการอ่านค่าข้อมูลการทำงานของไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลคือไฟล์ dhcp.conf มีรายละเอียดการแก้ไขไฟล์ ส่วนของการอ่านค่าข้อมูลการทำงานคือไฟล์ต้นฉบับ confpars.c ดังรูปที่ 4.2

```
case BLACKLIST:
next_token(&val, cfile);
blacklist = malloc(strlen(val));
strcpy(blacklist,val);
skip_to_semi(cfile);
break;
```

รูปที่ 4.2 ส่วนเพิ่มขยายในการจัดการกำหนดค่าไฟล์ข้อมูล Mac Address ของกลุ่มชนิดที่ไม่ให้บริการ (Blacklisted User)

ส่วนเพิ่มเติมการกำหนดค่าไฟล์ข้อมูล เครื่องลูกข่ายกลุ่มชนิดที่ไม่ให้บริการ (Blacklisted User) โดยมีขั้นตอนวิธีการทำงานคือ การจับจองการใช้งานหน่วยความจำของคำสั่ง malloc(strlen(val)), กำหนดค่าที่ได้ให้กับตัวแปร blacklist ของคำสั่ง strcpy(blacklist,val) และการอ่านค่าข้อมูลจากไฟล์เข้าสู่หน่วยความจำจากฟังก์ชั่น readblacklist() ดังรูปที่ 4.3

```
void readblacklist()
{
        FILE *fp = fopen(blacklist,"r");
        char buff[30] = {0};
        struct member *tmp,*tail;
        blacklists = NULL;
        while(fgets(buff,sizeof(buff),fp) != NULL)
        {
                tmp = malloc(sizeof(struct member));
                buff[strlen(buff)-1] = '\0';
                strcpy(tmp->mac,buff);
                tmp->next = NULL;
                if(blacklists == NULL)
                {
                        blacklists = tmp;
```

```
                    tail = blacklists;
                }
                else
                {
                    tail->next = tmp;
                    tail = tail->next;
                }
        }
        fclose(fp);
}
```

รูปที่ 4.3 ส่วนการอ่านค่าข้อมูล Mac Address ของกลุ่มชนิดที่ไม่ให้บริการ (Blacklisted User) จากไฟล์เข้าสู่หน่วยความจำ

ระบบจะทำการอ่านข้อมูลเครื่องลูกข่ายที่อยู่ในกลุ่มชนิดที่ไม่ให้บริการ (Blacklisted User) เพื่อใช้ในการตรวจสอบเครื่องลูกข่ายที่เข้ามาใช้บริการจากเครื่องแม่ข่ายไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล ซึ่งฟังก์ชั่นการทำงานอยู่ในไฟล์ต้นฉบับ readregistermac.c โดยการอ่านค่าข้อมูลของค่าที่กำหนดแล้ว ได้มีการตรวจสอบค่าว่าเป็นข้อมูลที่ถูกต้องถ้าค่าไม่ถูกต้องจะมีการแจ้งเตือน ดังรูปที่ 4.4

```
case DHCPGROUP:
next_token (&val, cfile);
g = atoi(val);
if (g > 2 || g < 0)
parse_warn ("unavailable group.");
else
group->dhcpgroup = g;
skip_to_semi (cfile);
break;
```

รูปที่ 4.4 ส่วนเพิ่มขยายในการจัดการกำหนดค่า ชนิดของกลุ่มของเครื่องลูกข่าย

ส่วนเพิ่มเติมการกำหนดค่าไฟล์ข้อมูลเครื่องลูกข่ายของแต่ละกลุ่มผู้ใช้งาน โดยมีขั้นตอนวิธีการทำงานคือ การจับจองการใช้งานหน่วยความจำของคำสั่ง malloc(sizeof(strlen(val))), กำหนดค่าที่ได้ให้กับตัวแปร group->shared_network->macdb ของคำสั่ง strcpy(group->shared_network->macdb,val) ดังรูปที่ 4.5

```
case MACDB:
next_token(&val, cfile);
group->shared_network->macdb = malloc(sizeof(strlen(val)));
strcpy(group->shared_network->macdb,val);
skip_to_semi(cfile);
break;
```

รูปที่ 4.5 ส่วนเพิ่มขยายในการจัดการกำหนดค่า ข้อมูล Mac Address ของกลุ่มเครื่องลูกข่ายของแต่ละกลุ่ม

การอ่านค่าข้อมูลจากไฟล์เข้าสู่หน่วยความจำจากฟังก์ชั่น readregismac() เพื่อใช้ในการตรวจสอบเครื่องลูกข่ายที่เข้ามาใช้บริการ เครื่องแม่ข่ายไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล ดังรูปที่ 4.6

```
void readregismac()
{
        FILE *fp;
        struct member *tmp, *tail;
        char buff[30] = {0};
        int i = 0;
        struct shared_network *shares = getallshared_network();
        while(shares != NULL)
        {
                shares->dhcpgroup = shares->group->dhcpgroup;
                sync();
                sync();
                printf("=> Read Mac File=> %d\n",(fp = fopen(shares->macdb,"r")));
                fp = fopen(shares->macdb,"r");
                fflush(fp);
                sync();
                sync();
                if(shares->dhcpgroup == 0)
                {
                        NOTMEMBERSHARED_NETWORK = shares;
                        shares = shares->next;
                        continue;
                }
                while(fgets(buff,sizeof(buff),fp) != NULL)
                {
                        tmp = malloc(sizeof(struct member));
                        buff[strlen(buff)-1] = '\0';
                        strcpy(tmp->mac,buff);
                        tmp->macdb = malloc(sizeof(shares->macdb));
```

```
                    strcpy(tmp->macdb,shares->macdb);
                    tmp->group = shares->dhcpgroup;
                    tmp->next = NULL;

                    if(members == NULL)
                    {
                        members = tmp;
                        tail = members;
                    }
                    else
                    {
                        tail->next = tmp;
                        tail = tail->next;
                    }
                }
                shares = shares->next;
                fflush(fp);
                fcloseall(fp);
                i++;
            }
}
```

รูปที่ 4.6 ส่วนการอ่านค่าข้อมูล Mac Address ของกลุ่มผู้ใช้งานจากไฟล์เข้าสู่หน่วยความจำ

จากรูปที่ 4.7 ตัวอย่างการกำหนดค่าการทำงานของการปรับปรุงการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล ได้มีส่วนเพิ่มเติมจากการกำหนดค่าของการทำงานของไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล คือการจัดการกำหนดค่าไฟล์ข้อมูล Mac Address ของกลุ่มชนิดที่ไม่ให้บริการ (Blacklisted User), การกำหนดกลุ่มเครื่องลูกข่าย, กำหนดชนิดของกลุ่มเครื่องลูกข่าย โดยมีความหมายคือ ค่า 0 คือกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบปรับปรุงระบบความปลอดภัย (Secured Public), ค่า 1 คือกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบส่วนตัวแบบปกติ (Non-secured Private), ค่า 2 คือกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบส่วนตัวแบบป้องกันความปลอดภัย (Secured Private) และกำหนดหนดค่าไฟล์ ข้อมูลเครื่องลูกข่ายของแต่ละกลุ่มผู้ใช้งาน ส่วนอื่น ๆ มีลักษณะการกำหนดค่าคล้ายๆ กับการกำหนดค่าการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลแบบเดิม

```
default-lease-time 43200;
max-lease-time 86400;
blacklist "/etc/dhcp/blacklist";                    // กำหนดหนดค่าไฟล์ ข้อมูลเครื่องลูกข่ายของ
BLACKLIST
group   {                                           // กำหนดกลุ่มเครื่องลูกข่า
        dhcpgroup 0;                                // กำหนดชนิดของกลุ่มเครื่องลูกข่าย
        shared-network NOTAMEMBER{
        subnet 192.168.0.0 netmask 255.255.128.0{
        range 192.168.0.13 192.168.63.255;
        option subnet-mask 255.255.255.252;
        option routers 192.168.0.1;
        option domain-name "public_zone.com";
        option domain-name-servers 192.168.0.1;
} } }
group{                                              // กำหนดกลุ่มเครื่องลูกข่าย
        dhcpgroup 1;                                // กำหนดชนิดของกลุ่มเครื่องลูกข่าย
        shared-network normalgroup{
        subnet 172.16.0.0 netmask 255.255.255.0{
        macdb "/etc/dhcp/1";  // กำหนดหนดค่าไฟล์ ข้อมูลเครื่องลูกข่ายของกลุ่มผู้ใช้งาน
        option subnet-mask 255.255.255.0;
        range 172.16.0.101 172.16.0.200;
        option routers 172.16.0.1;
        option domain-name "nonsecured_private_zone.com";
        option domain-name-servers 192.168.0.1;
} } }
group{                                              // กำหนดกลุ่มเครื่องลูกข่าย
        dhcpgroup 2;                                // กำหนดชนิดของกลุ่มเครื่องลูกข่าย
        shared-network securedmember{
        subnet 172.16.128.0 netmask 255.255.128.0{
        macdb "/etc/dhcp/2";  // กำหนดหนดค่าไฟล์ ข้อมูลเครื่องลูกข่ายของกลุ่มผู้ใช้งาน
        option subnet-mask 255.255.255.252;
        range 172.16.128.13 172.16.131.255;
        option routers 172.16.128.1;
        option domain-name "secured_private_zone.com";
        option domain-name-servers 192.168.0.1;
} } }
```

รูปที่ 4.7 ตัวอย่างการกำหนดค่าการทำงานของการปรับปรุงการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล

4.2 การพัฒนาส่วนปรับปรุงการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล ส่วนการจัดการการจัดสรรการเชื่อมต่อ (Connection Allocation)

ส่วนการจัดการการจัดสรรการเชื่อมต่อ (Connection Allocation) ของการปรับปรุงระบบความปลอดภัยของไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล สำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะ จะทำงานเมื่อเครื่องแม่ข่ายได้รับการร้องขอบริการไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลจากเครื่องลูกข่าย ได้เพิ่มการทำงานในส่วนการตอบกลับการให้บริการของไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลในส่วน Dhcpdiscover กับเครื่องลูกข่าย ซึ่งทำการแก้ไขไฟล์ dhcp.c มีรายละเอียดส่วนเพิ่มขยาย ดังรูปที่ 4.8

```
void dhcpdiscover (packet)
        struct packet *packet;
{
.....     // ส่วนข้อมูล Code ที่ไม่มีการเปลี่ยนแปลง
        current_group(mac);                    // ตรวจสอบชนิดของกลุ่มเครื่องลูกข่าย
        if(CURGROUP == -1)
        {
                printf("= Black List =\n");
                return;
        }
.....     // ส่วนข้อมูล Code ที่ไม่มีการเปลี่ยนแปลง
}
```

รูปที่ 4.8 ส่วนเพิ่มการจัดการการเชื่อมต่อของ Dhcpdiscover

เมื่อเครื่องแม่ข่ายในระบบได้รับการร้องขอบริการจากเครื่องลูกข่าย เครื่องแม่ข่ายจะทำการตรวจสอบ กลุ่มของเครื่องลูกข่าย โดยใช้ฟังก์ชั่น current_group() ซึ่งอยู่ในไฟล์ต้นฉบับ readregistermac.c โดยส่งค่า Mac Address ของเครื่องลูกข่ายที่ขอบริการไปตรวจสอบ โดยมีรายละเอียด ดังรูปที่ 4.9


```
int current_group(char mac[])
{
        struct member *tmp = members;
        CURGROUP = -1;
        CURMACDB = NULL;
        if(members == NULL)
        {
                CURGROUP = 0;
```

```
                return 0;
        }
        if(CURMACDB == NULL)
        CURMACDB = malloc(strlen(tmp->macdb));
        while(tmp != NULL)
        {
                if(strcmp(mac,tmp->mac) == 0)
                {
                        CURGROUP = tmp->group;
                        strcpy(CURMACDB,tmp->macdb);
        printf("In Current mac = %s and Tmp mac = %s Cur = %d\n",mac,tmp-
>mac,CURGROUP);
                        return CURGROUP;
                }
                tmp = tmp->next;
        }
        tmp = blacklists;
        while(tmp != NULL)
        {
                if(strcmp(mac,tmp->mac) == 0)
                {
                        CURGROUP = -1;
                        return CURGROUP;
                }
                tmp = tmp->next;
        }
        CURGROUP = 0;
        strcpy(CURMACDB,"");
        return CURGROUP;
}
```

รูปที่ 4.9 ฟังก์ชั่น current_group(mac) ตรวจสอบกลุ่มของเครื่องลูกข่าย

เมื่อระบบได้ทำการตรวจสอบกลุ่มของเครื่องลูกข่าย โดยกำหนดค่าเริ่มต้นให้กับกลุ่มของเครื่องลูกข่ายเป็น ค่า 0 คือกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบปรับปรุงระบบความปลอดภัย (Secured Public) ได้ตรวจสอบว่าเครื่องลูกข่ายอยู่ในกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบส่วนตัวแบบป้องกันความปลอดภัย (Secured Private) หรือกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบส่วนตัวแบบปกติ (Non-secured Private) ถ้าอยู่ในกลุ่มหนึ่งกลุ่มใดจะทำการส่งค่าของกลุ่มเครื่องลูกข่ายกลับ แต่ถ้าไม่อยู่ในกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบส่วนตัวแบบป้องกันความปลอดภัย (Secured

Private) และกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบส่วนตัวแบบปกติ (Non-secured Private) ระบบจะทำการตรวจสอบว่าเครื่องลูกข่ายอยู่ในกลุ่มชนิดที่ไม่ให้บริการ (Blacklisted User) ถ้าเครื่องลูกข่ายอยู่ในกลุ่มชนิดที่ไม่ให้บริการ (Blacklisted User) ระบบจะส่งค่า -1 กลับ ถ้าไม่อยู่ระบบจะส่งค่า 0 ซึ่งเป็นค่าเริ่มต้นกลับ

```
void dhcp_reply (lease)
struct lease *lease;
{
.....    // ส่วนข้อมูล Code ที่ไม่มีการเปลี่ยนแปลง
        if(CURGROUP == 0 || CURGROUP == 2)
        {
        mac = print_hw_addr(raw.htype,raw.hlen,raw.chaddr);
        subip = inet_ntoa(raw.yiaddr);
        mask = addrip(lease->subnet->group->options[DHO_SUBNET_MASK]-
>tree->data.const_val.data);
        if(state -> offer == DHCPACK)
        create_subinterface(mac,subip,P2PMASK);
        }
.....    // ส่วนข้อมูล Code ที่ไม่มีการเปลี่ยนแปลง}
```

รูปที่ 4.10 ส่วนการจัดการการจัดสรรการเชื่อมต่อ (Connection Allocation)

การจัดสรรการเชื่อมต่อ (Connection Allocation) ของกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบปรับปรุงระบบความปลอดภัย (Secured Public) และกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์ส่วนตัวแบบป้องกันความปลอดภัย (Secured Private) ระบบจะสร้าง Interface มารองรับการเชื่อมต่อระบบกับเครื่องลูกข่าย ซึ่งค่าเครือข่ายของเครื่องลูกข่ายจะแตกต่างไปจากไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลเดิม โดยฟังก์ชั่นcreate_subinterface () ในไฟล์ต้นฉบับ inet.c มีรายละเอียด ดังรูปที่ 4.11

```
void create_subinterface(mac,ip,netmask)
        char *mac;
        char *ip;
        char *netmask;
{
        char cmd[150] = {0};
        char nextip[30] = {0};
        char mask[30] = {0};
```

```
        struct iaddr tmp;
        char broadcast[30];
        struct stat st;
        int avilslot;
        struct subinterface *before;
        struct subinterface *newsub = dmalloc(sizeof(struct
subinterface),"sub_interface");
        memset(&tmp,0,sizeof(struct iaddr));
        memset(newsub,0,sizeof(newsub));
        avilslot = 0;
        if(checkmac(mac) == 0)
        {
                down_subinterface();
                return;
        }
        memset(newsub->ip,0,sizeof(newsub->ip));
        strcpy(newsub->mac,mac);
        strcpy(newsub->mask,netmask);
        strncpy(newsub->ip,ip,strlen(ip));
        strcpy(broadcast,piaddr(broadcast_addr(ipaddr(ip),ipaddr(netmask))));
        strcpy(newsub->broadcast,broadcast);
        newsub->number = 1;
        tmp = nextipaddr(ipaddr(ip));
        strcpy(mask,netmask);
        sprintf(nextip,"%s",piaddr(tmp));
        strcpy(newsub->nextip,nextip);
        if(sub_interface != NULL)
        {
                before = nextslot();
                if(before == sub_interface && before->number != 1)
                {
                        newsub->next = before;
                        sub_interface = newsub;
                }
                else
                {
                        newsub->number = before->number + 1;
                        newsub->next = before->next;
                        before->next = newsub;
                }
        }
        else
```

```
        {
                sub_interface = newsub;
        }
        memset(cmd,0,sizeof(cmd));
        sprintf(cmd,"ifconfig eth0:%d inet %s netmask %s broadcast %s",newsub-
>number,nextip,mask,broadcast,mask);
        // สร้าง Interface เพื่อรองรับการเชื่อมต่อกับเครื่องลูกข่าย
        printf("%s\n",cmd);
        system(cmd);
        memset(cmd,0,sizeof(cmd));
                                // สามารถสร้าง Script การจัดการแต่ละเครื่องลูกข่าย
}
struct iaddr ipaddr(ip)
        char *ip;
{
        struct iaddr rv;
        int i = 0;
        int j = 0;
        int k = 0;
        char token[3] = {0};
        memset(&rv,0,sizeof(rv));
        rv.len = 4;
        for(i=0; i<rv.len;i++)
        {
                while(ip[j] != '.' && ip[j] != '\0')
                        token[k++] = ip[j++];
                j++;
                rv.iabuf[i] = rv.iabuf[i] | atoi(token);
                memset(token,0,sizeof(token)) ;
                k = 0;
        }
        return rv;
}
char *addrip (str)
        unsigned char *str;
{
        static char pbuf [4 * 16];
        char *s = pbuf;
        int i;
        for (i = 0; i < strlen(str); i++) {
                sprintf (s, "%s%d", i ? "." : "", str[i]);
                s += strlen (s);
        }
        return pbuf;
}
```

```
struct iaddr nextipaddr(ip)
        struct iaddr ip;
{
        ip.iabuf[3] = (ip.iabuf[3] ^ 3);
        return ip;
}
struct subinterface *nextslot()
{
        struct subinterface *before;
        before = sub_interface;
        if(before->number != 1)
        return before;
        while(before->next != NULL)
        {
                if(before->number != before->next->number-1)
                break;
                before = before->next;
        }
        return before;
}
```

รูปที่ 4.11 ฟังก์ชั่น create_subinterface() ส่วนการจัดการการจัดสรรการเชื่อมต่อ

ในส่วนฟังก์ชั่น create_subinterface เป็นส่วนในการจัดสรรค่าต่างๆ ให้กับเครื่องลูกข่ายที่อยู่ในกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะ แบบปรับปรุงระบบความปลอดภัย (Secured Public) และกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์ส่วนตัวแบบป้องกันความปลอดภัย (Secured Private) ส่วนกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบส่วนตัวแบบปกติ (Non-secured Private) จะถูกกำหนดค่าลักษณะไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลแบบเดิม ซึ่งสามารถใช้ฟังก์ชั่นเดิม

4.3 การพัฒนาส่วนปรับปรุงการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล ส่วนจัดการการส่งข้อมูล (Packet Sending)

ส่วนจัดการการส่งข้อมูล (Packet Sending) ของการปรับปรุงระบบความปลอดภัยของไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล สำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะ จะทำงานเมื่อเครื่องแม่ข่ายตอบกลับการให้บริการไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลกับเครื่องลูกข่าย โดยเพิ่มการทำงานในส่วน Dhcp_reply ของข้อมูล DHCPPACK กับเครื่องลูกข่าย ซึ่งทำการแก้ไขไฟล์ต้นฉบับ dhcp.c มีรายละเอียดส่วนเพิ่มขยายดังรูปที่ 4.12

```
void dhcp_reply (lease)
        struct lease *lease;
{
.....    // ส่วนข้อมูล Code ที่ไม่มีการเปลี่ยนแปลง
        if(CURGROUP == 0 || CURGROUP == 2)
        {
                tmp = nextipaddr(lease -> ip_addr);
                if(state -> offer == DHCPACK)
                {
                        i = DHO_ROUTERS;
                        state -> options [i] = new_tree_cache ("routers");
                        state -> options [i] -> flags = TC_TEMPORARY;
                        state -> options [i] -> value = tmp.iabuf;
                        state -> options [i] -> len = tmp.len;
                        state -> options [i] -> buf_size = tmp.len;
                        state -> options [i] -> timeout = 0xFFFFFFFF;
                        state -> options [i] -> tree = (struct tree *)0;
                }
        }
.....    // ส่วนข้อมูล Code ที่ไม่มีการเปลี่ยนแปลง
}
```

รูปที่ 4.12 ส่วนจัดการการส่งข้อมูล (Packet Sending)

ในการจัดการการส่งข้อมูลของการปรับปรุงระบบความปลอดภัยของ ไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล สำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะ กลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบปรับปรุงระบบความปลอดภัย (Secured Public) และกลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์ส่วนตัวแบบป้องกันความปลอดภัย (Secured Private) โดยมีข้อมูลที่ได้จากส่วนการจัดการการจัดสรรการเชื่อมต่อ (Connection Allocation) แต่กลุ่มเครื่องลูกข่ายบนเครือข่ายคอมพิวเตอร์สาธารณะแบบส่วนตัวแบบปกติ (Non-secured Private)

จะถูกกำหนดค่าลักษณะไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลแบบเดิม ซึ่งสามารถใช้ฟังก์ชั่นการทำงานเดิมของโฮสต์คอนฟิกกิวเรชันโปรโตคอลได้

4.4 การพัฒนาส่วนปรับปรุงการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลส่วนอื่น

ส่วนการปรับปรุงการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลส่วนอื่นของการปรับปรุงระบบความปลอดภัยของไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล สำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะ ในส่วนนี้จะเป็นในส่วนของการช่วยเหลือเพื่อให้การปรับปรุงระบบความปลอดภัยของไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล สำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะทำงานได้ดีขึ้น

ส่วนที่ช่วยให้การทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลส่วนอื่น ของการปรับปรุงระบบความปลอดภัยของไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล สำหรับสภาพแวดล้อมเครือข่ายคอมพิวเตอร์แบบสาธารณะ เมื่อมีการปิดและเปิดเครื่องแม่ข่าย เครื่องลูกข่ายไม่จำเป็นต้องทำการร้องขอบริการ ไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลกับเครื่องแม่ข่ายใหม่ โดยเครื่องแม่ข่ายจะทำการบันทึกการให้บริการในไฟล์และอ่านข้อมูลขึ้นมาเพื่อสร้าง Interface รองรับการเชื่อมต่อเครื่องลูกข่าย เมื่อเครื่องแม่ข่ายเริ่มการทำงานของ ไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล การบันทึกการให้บริการไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอลไฟล์จะบันทึกเมื่อมีการใช้งานฟังก์ชั่น create_subinterface() ซึ่งอยู่ในไฟล์ต้นฉบับ inet.c โดยเรียกฟังก์ชั่น writesubinterface() ซึ่งอยู่ในไฟล์ต้นฉบับ readregismac.c โดยมีรายละเอียด ดังรูปที่ 4.13

```
void create_subinterface(mac,ip,netmask)
        char *mac;
        char *ip;
        char *netmask;
{
.....     // ส่วนข้อมูล Code ที่ไม่มีการเปลี่ยนแปลง
writesubinterface();
.....     // ส่วนข้อมูล Code ที่ไม่มีการเปลี่ยนแปลง
}
```

รูปที่ 4.13 การทำงาน writesubinterface() ในฟังก์ชั่น create_subinterface()

โดยระบบกำหนดค่า SUBINTERFACE เป็นค่าตัวแปรเพื่อกำหนดค่าไฟล์ที่เก็บข้อมูล subinterface ซึ่งมีค่าเป็น "/var/state/dhcp/subinterface" และเมื่อมีการเรียกฟังก์ชั่น writesubinterface() ระบบจะทำการบันทึกค่าข้อมูล subinterface ลงในไฟล์ ดังรูปที่ 4.14

```
#define SUBINTERFACE "/var/state/dhcp/subinterface"
void writesubinterface()
{
        FILE *fp = fopen(SUBINTERFACE,"wb");
        struct subinterface *tmp = sub_interface;
        if(tmp == NULL)
        {
                return;
        }
        while(tmp != NULL)
        {
                fwrite(tmp,sizeof(struct subinterface),1,fp);
                tmp = tmp->next;
        }
        fclose(fp);
}
```

รูปที่ 4.14 ฟังก์ชั่น writesubinterface() ในส่วนช่วยการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล

เมื่อเครื่องแม่ข่ายเริ่มการทำงานของไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล ระบบจะทำการอ่านข้อมูลในไฟล์ "/var/state/dhcp/subinterface" เพื่อทำการสร้าง Interface เพื่อรองรับการเชื่อมต่อกับเครื่องลูกข่าย โดยฟังก์ชั่น readsubinterface() ซึ่งอยู่ในไฟล์ต้นฉบับ readregismac.c โดยมีรายละเอียด ดังรูปที่ 4.15

```
void readsubinterface()
{
        FILE *fp = fopen(SUBINTERFACE,"rb");
        struct subinterface *tmp = malloc(sizeof(struct subinterface));
        struct subinterface *tail;
        char cmd[2000] = {0};
        struct stat st;
        while(fread(tmp,sizeof(struct subinterface),1,fp) > 0){
                if(sub_interface == NULL)
        {
                        printf("=> Start Read Old Sub Interface\n");
                        tail = tmp;
                        sub_interface = tail;
                }
                else
        {
                        tail->next = tmp;
                        tail = tmp;
```

```
                }
sprintf(cmd,"ifconfig eth0:%d inet %s netmask %s broadcast %s",tmp->number,tmp-
>nextip,tmp->mask,tmp->broadcast);          // สร้าง Interface เพื่อรองรับการเชื่อมต่อกับ
เครื่องลูกข่าย
                system(cmd);
                tmp = malloc(sizeof(struct subinterface));
                printf("=> End Read Old Sub Interface\n");
        }
        fclose(fp);
}
```

รูปที่ 4.15 ฟังก์ชั่น readsubinterface() ในส่วนช่วยการทำงานไดนามิกโฮสต์คอนฟิกกิวเรชันโปรโตคอล