電子メール送信者のなりすましの抑制

96H021 江口裕一

目 次

1 はじめに **1**

2 電子メール送信者のなりすまし **2**
2.1 なりすましの定義 2
2.1.1 なりすます方法 2
2.2 抑制対象としている環境 4
2.2.1 IP マスカレード 4
2.2.2 POP before SMTP 4

3 なりすましの抑制 **5**
3.1 なりすましの抑制方法 5
3.2 ゲートウェイマシンの設定 9
3.3 中継用プログラム 9
3.3.1 stone 10
3.3.2 認証情報の保持 10
3.3.3 なりすましの判別 19
3.3.4 認証情報の削除 22
3.3.5 なりすましへの対応 24

4 考察 **25**

5 まとめ **27**

1　はじめに

近年、インターネット利用人口が増加し生活に密着しだすとともに、電子メールも生活に欠かせないものに定着しつつある。しかし、一般的に使われている電子メールクライアントソフトにおいて、ユーザが架空の電子メールアドレスや他人の電子メールアドレスを用いて電子メールを送信すれば、誰でもが容易に送信者を偽ることができてしまう。特に、IP マスカレードを導入した環境では、電子メールサーバから各クライアントマシンの IP アドレスを特定できないため、送信元電子メールアドレスを偽った送信者を特定するのが困難である。

本研究では、これらの問題を解決するために、IP マスカレードを導入した環境で電子メール送信者のなりすましを抑制するのを目的とする。

2　電子メール送信者のなりすまし

電子メールを利用している一般ユーザは、他人がなりすまして送ってきた電子メールであっても送信者名と送信元電子メールアドレスを見て送信者を判断してしまうことがほとんどである。一般的には、ユーザの使用しているマシンの IP アドレスを特定できるため、電子メール送信者の特定ができるように電子メール送信サーバを設定することが可能である。しかし、IP マスカレードを導入した環境ではクライアントマシンの IP アドレスを特定できないために、多数のユーザがマシンを共用しているような本学の演習室等の環境では、なりすましを行った送信者を特定することが困難である。

また、ユーザが自分自身の電子メールアドレスを設定し間違えることにより、ユーザが不本意になりすまし電子メールを送信してしまうこともある。このようなユーザがさらに送信先の電子メールアドレスを間違えていた場合、エラーを通知する電子メールをユーザへ送ることが出来なくなってしまう。つまり、エラーを通知するための電子メールが届かなくなり、管理者の元へエラーを通知する電子メールが送られることになる。

2.1　なりすましの定義

ユーザが受信チェックする際に使用するユーザ名に対応する電子メールアドレス以外は全てなりすましであるものとする。つまり、root など管理者権限で着信チェック時には一般的に使われないようなユーザ名の電子メールアドレスを使用して送信した場合にも、送信元のユーザ名で電子メールの着信チェックが行われていなければ、なりすましであるものとする。

2.1.1　なりすます方法

MS-Windows などで使用されている一般的なクライアントソフトを使って、電子メール送信者が他人になりすますことは非常に容易である。電子メールアドレスを架空のものや他人のものを記述したのち電子メールを送信するとなんの問題もなく電子メールは届き、送られてきた電子メールの送信元電子メールアドレスはなりすましたものになる。

以下に、本学の演習室で使用されている AL-Mail を用いて例をあげる。ユーザは s96h021@sub.osaka-sandai.ac.jp という電子メールアドレスを所有しており、ユーザが所有していない hogehoge@sub.osaka-sandai.ac.jp という電子メールアドレスで電子メールを送信する方法である。

1. 図 1のように自分自身の電子メールアドレスを書き換える。
2. 電子メールを新規に作成する。
3. 電子メールを送信する。
4. 図 2のように差出人が hogehoge@sub.osaka-sandai.ac.jp である電子メールが電子メール受信者の元へ届く。

図 1: クライアントソフトの設定

図 2: なりすましメール

2.2 抑制対象としている環境

本研究が電子メール送信者のなりすましを抑制する対象としているのは、多数のユーザが複数のコンピュータを共用していて、クライアントマシンに IP マスカレードが施されている本学の演習室のような環境である。

2.2.1 IP マスカレード

IP マスカレードとは、2 つのネットワークがルータマシン等を介して隣接している環境で、一方のネットワークに属しているマシンが、ルータマシンの IP アドレスを用いて、もう一方のネットワークに属するマシンと通信するための機能である。特に、ローカル IP アドレスしか持たない複数のクライアントマシンがゲートウェイマシンの持つ 1 つのグローバル IP アドレスを利用してインターネットに接続させる場合に使用されることが多い。この場合、インターネット上にある電子メールサーバからはクライアントマシンの個々の IP アドレスを特定することができないということになる。

2.2.2 POP before SMTP

現在、インターネット接続プロバイダーなどでは電子メール送信サーバを第三者に悪用されないように外部ネットワークから外部ネットワークへ電子メールを送信する際の電子メール送信サーバの使用を制限しているところがほとんどである。その際に、POP before SMTP を導入して外部からの電子メール送信サーバを使用するのが自プロバイダー会員かどうかを判別するようにしている。

POP before SMTP とは、電子メール着信チェックが正常に行えたユーザが使用しているクライアントマシンに割り当てられてる IP アドレスを電子メールサーバ内に記録しておき、一定時間に、その IP アドレスを持つクライアントマシンから電子メールサーバを使用することを可能とするものである。つまり、一定時間経過すると電子メールサーバ内に記録されたそのクライアントマシンの IP アドレスが消去されるということである。この一定時間というのを 15 分程度に設定しているプロバイダーが多いようである。

また、この POP before SMTP を使用するとなりすましを行った電子メール送信者を特定することが可能になる。しかし、IP マスカレードを導入した環境では電子メールサーバから参照するクライアントマシンの IP アドレスは複数マシン単位に同一のものとなってしまう。つまり、このような環境で電子メールサーバに POP before SMTP を使用したとしても、本来期待される効果が発揮できないことになる。

3　なりすましの抑制

本研究は、電子メール送信者のなりすましを抑制するために、クライアントマシンの IP アドレスを直接参照できるゲートウェイマシンに、POP before SMTP の概念を応用して使用する。つまり、ゲートウェイマシンにユーザが電子メールの着信チェックをした際の認証情報を記録しておき、ユーザが電子メールを送信する際になりすましが行われていないかを判別するために利用する。

3.1　なりすましの抑制方法

電子メール送信者のなりすましを抑制するにあたって、図 3のような環境を想定した。この環境は、クライアントマシンがインターネット上の電子メールサーバを利用する際、IP マスカレードを施しているゲートウェイマシンの IP アドレスを用いるものである。

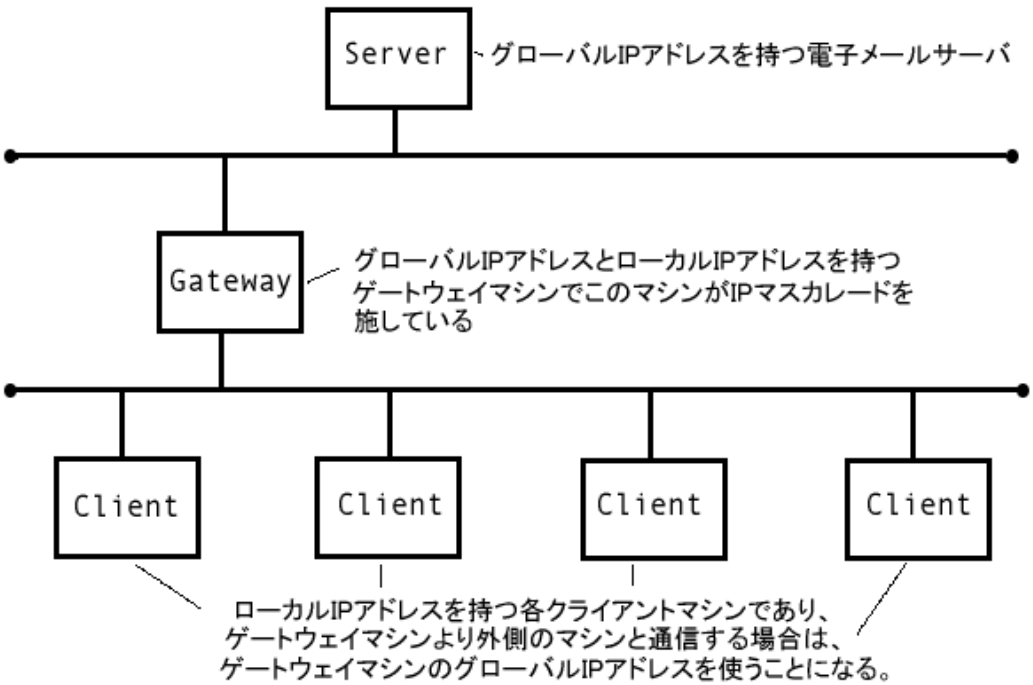

図 3: 想定した環境

電子メール送信者のなりすましを抑制する方法は次のような順序で行う。

1. ゲートウェイマシンは、ユーザが電子メールの着信チェックする際に、クライアントマシンが直接電子メールサーバへ POP3 の接続を試みるのを、ゲートウェイマシン内で動作している中継プログラムを介して接続するようにする。

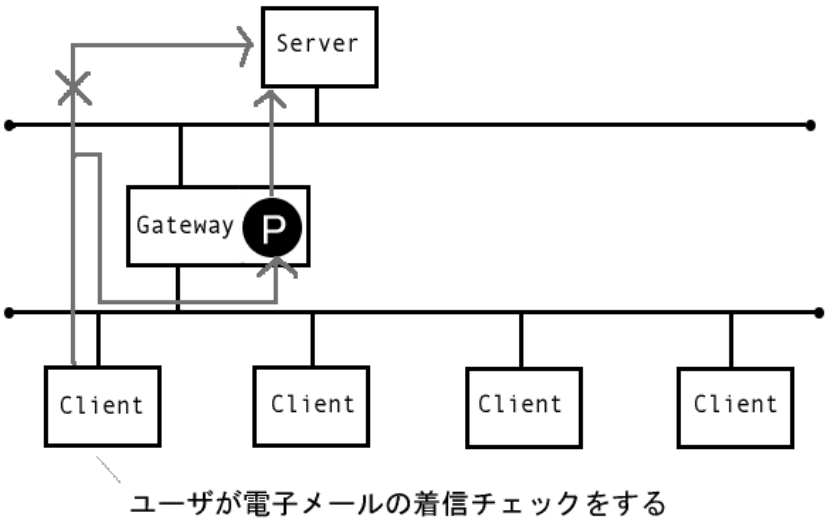

図 4: ユーザの着信チェック

2. 中継プログラムは、ユーザの受信メールチェックを中継する際に、着信チェックをした時刻、ユーザの使用しているクライアントマシンのローカル IP アドレス、ユーザの電子メールアドレスをゲートウェイマシンのファイルに書き込む。

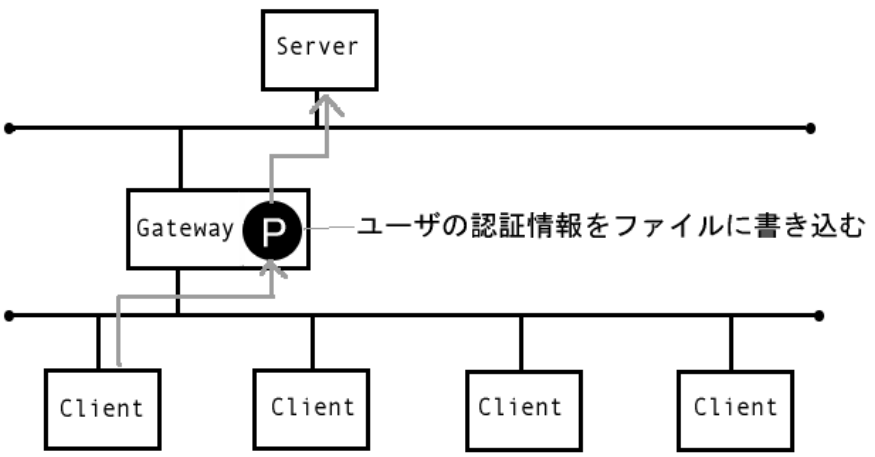

図 5: 認証情報の保持

3. ゲートウェイマシンは、ユーザが電子メールを送信する際に、クライアントマシンが直接電子メールサーバへ SMTP の接続を試みるのを、ゲートウェイマシン内で動作している中継プログラムを介して接続するようにする。

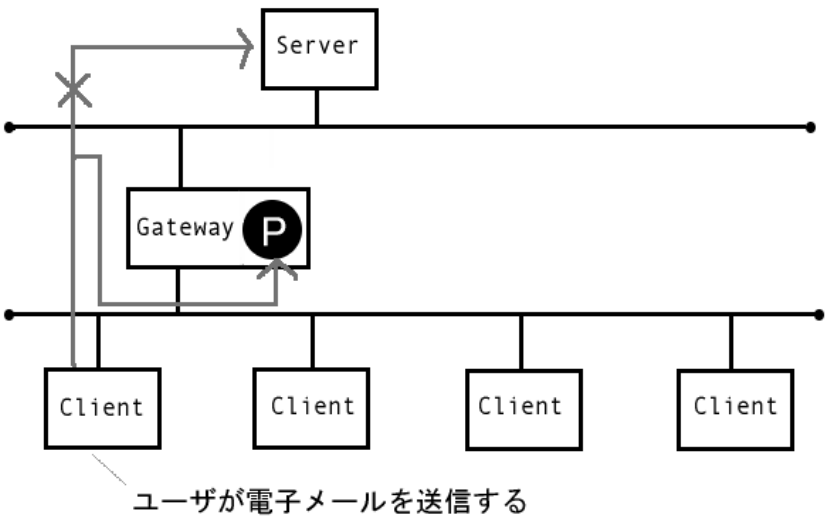

図 6: ユーザの電子メール送信

4. 中継プログラムは、送信する電子メールの送信元電子メールアドレスとクライアントマシンの IP アドレスがゲートウェイマシン内のファイルに一致するものがあるか調べる。

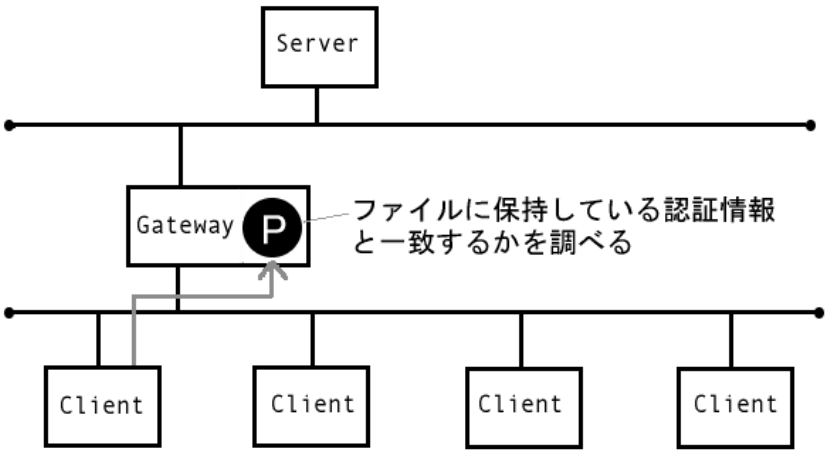

図 7: なりすましの判別

5. ゲートウェイマシンのファイル中に一致するものがある場合は、電子メールを電子メールサーバへの中継を続行する。

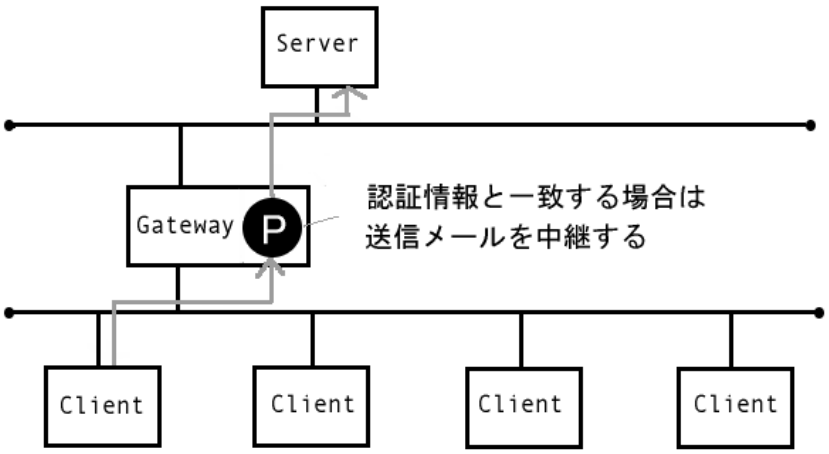

図 8: なりすましのない場合

6. ゲートウェイマシンのファイル中に一致するものがない場合は、電子メールサーバへの中継を中断し、クライアントマシンで動作する電子メールクライアントソフトへエラーであることを伝える。

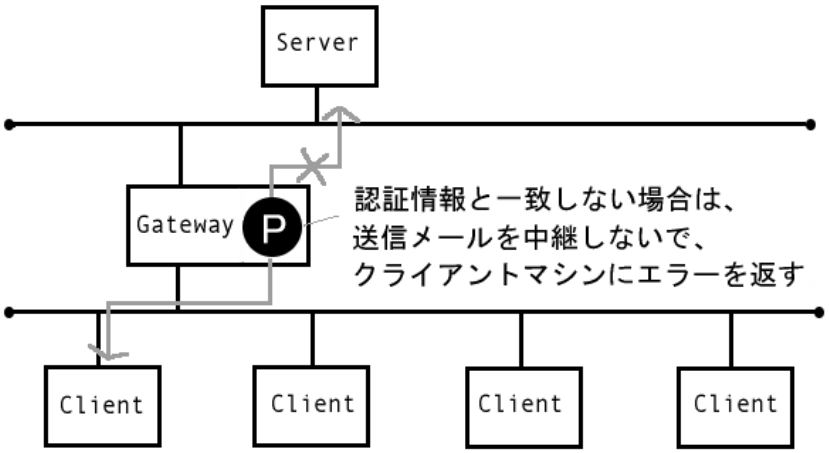

図 9: なりすましのある場合

7. 中継プログラムはユーザが電子メールの着信チェックをしてから 15 分経過した後、ゲートウェイマシンのファイルに書き込んだ情報を消去する。

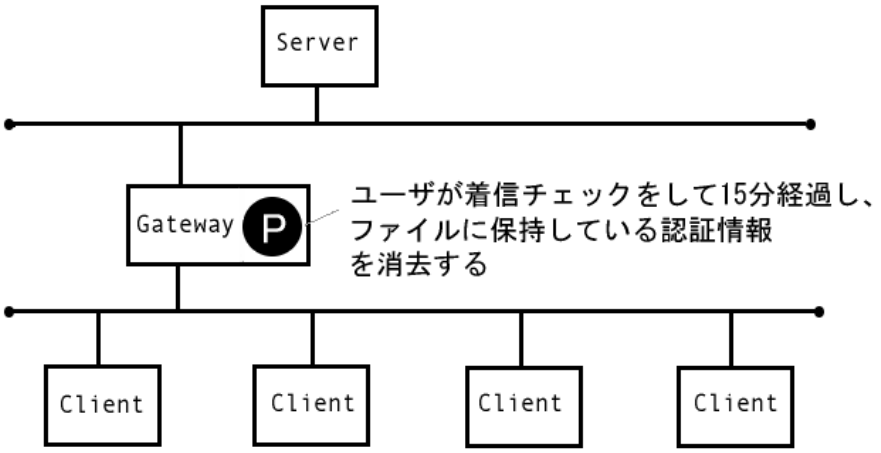

図 10: 認証情報の消去

3.2 ゲートウェイマシンの設定

ゲートウェイマシンは、クライアントマシンが電子メールサーバへ接続しようとした場合に、ゲートウェイマシンで動いている中継プログラムへ接続するように設定しなければならない。ここでは、実際の設定例として Linux のカーネル 2.0 系で ipfwadm というプログラムを使用した場合を挙げる。また、ここで挙げる設定例は演習室内から電子メールサーバに接続する際は、どのような電子メールサーバへ接続する設定であったとしても全てが中継プログラム経由で中継プログラムに設定された電子メールサーバへしか接続させない環境の設定である。ゲートウェイマシンの 8110 番ポートで POP3 用の中継プログラムが動いており、8025 番ポートで SMTP 用の中継プログラムが動いているものとする。

/sbin/ipfwadm -I -a accept -P tcp -D 0/0 110 -r 8110
/sbin/ipfwadm -I -a accept -P tcp -D 0/0 25 -r 8025

ゲートウェイマシンは、クライアントマシンから電子メールサーバの POP3 が動いている 110 番ポートへの接続要求があった場合に自らの 8110 番ポートへ接続させ、クライアントマシンから電子メールサーバの SMTP が動いている 25 番ポートへの接続要求があった場合に自らの 8025 番ポートへ接続させるという意味である。

3.3 中継用プログラム

本研究を進める上で、中継プログラムは以下の機能を満たすことが必要にある。

- 電子メールサーバとクライアントマシンの間で POP3 や SMTP を中継する
- 電子メールの受信チェックした際の情報をマシン内に保存する
- 送信元電子メールアドレスが保存されてるものと一致するか判別する
- 受信チェック時に保存した情報を一定時間経過後に削除する

3.3.1 stone

本研究で使用した中継プログラムは、stone(http://www.gcd.org/sengoku/stone/) というプログラムに着信チェック時の認証情報を蓄えて送信元電子メールアドレスに偽りがないか判別できる機能を付け加える形で estone というプログラムを作成した。

3.3.2 認証情報の保持

ユーザが電子メールの着信チェックをした際に得られる情報をゲートウェイマシン内に残す方法は以下の順で行う。

1. 中継プログラムがクライアントマシンからの接続を受け付けた時点で、クライアントマシンの IP アドレスを外部変数に書き込む。
2. クライアントマシンから'USER ユーザ名' と送られてきたら、ユーザ名を外部変数に書き込む。
3. 'PASS パスワード' を電子メールサーバへ中継し、電子メールサーバから'+OK' と送られてきたら、時刻・クライアントマシンの IP アドレス・ユーザ名に対応する電子メールアドレスをファイルに書き込む。

以下に認証情報を保持するために stone に追加・変更した estone のプログラムソースを示す。

```
----------宣言部分----------
typedef struct _Pair {
        :
        :

/* 構造体 Pair に着信チェックや電子メールの送信がどの程度進んでるかを
 * 判別するための変数 check、着信チェック時のユーザ名を入れる変数 user、
 * クライアントマシンの IP アドレスを入れる変数 ip_addr を宣言する
 */
    int check;
    char user[128];
    char ip_addr[16];

        :
        :
} Pair;

        :
        :

/* POP3 と SMTP のどちらを中継しているのかを判別するための変数 Eproto */
int Eproto = 0;

/* 着信チェック時の情報を削除するためにプログラムの指定 */
char timer_prog[128] = "/usr/local/sbin/estone_timer";

/* ユーザの電子メールアドレスのホスト名部分を指定する */
```

```
char mail_serv[128] = "sub.osaka-sandai.ac.jp";

/* 着信チェック時の情報を格納するファイルの指定 */
char auth_file_pass[128] = "/etc/estone/pop.auth";
```

```
————————————メイン関数————————————
int main(argc,argv)
int argc;
char *argv[];
{

    :
    :

/* プログラムを動作させる時の引数に'-P' とあれば、
 * POP3 を中継しながら、ユーザの認証情報を指定された
 * ファイルに格納する動作を estone にさせるために
 * 変数 Eproto に 1 を入れる。
 */
        case 'P':
          Eproto = 1;
          break;

/* プログラムを動作させる時の引数に'-S' とあれば、
 * SMTP を中継しながら、ユーザのなりすましを判別する
 * 動作を estone にさせるために変数 Eproto に 2 を入れる。
 */
        case 'S':
          Eproto = 2;
          break;

    :
    :

}
```

```
————————————関数 doaccept————————————-
Pair *doaccept(stonep)
Stone *stonep;
{

    :
    :

  char ip_addr[16];

    :
    :

/* クライアントの接続を受け付けたら、３２ビットのビット列である
 * IP アドレスを、８ビットごとに区切り１０進数にする関数 addr_client を
 * 呼び出し、変数 ip_addr に格納する。
 */
  sprintf(ip_addr,"%s",addr_client(&from.sin_addr));

    :
    :

/* POP3 もしくは、SMTP を中継している場合は、
 * 構造体 Pair 内の変数 ip_addr にクライアントの IP アドレスを入れ、
 * 中継の進行具合を把握するための変数 check を初期化する。
 */
  if(Eproto ≠ 0){
    sprintf(pair1→ip_addr,"%s",ip_addr);
    sprintf(pair2→ip_addr,"%s",ip_addr);
    pair1→check = pair2→check = 0;
  }

    :
    :

}
```

——————関数 addr_client——————

```
/* 変数 add_cliennt は、32 ビットのビット列を 8 ビットごと 4 区切りにして
 * 10 進数へ変換する関数である。
 * つまり、クライアントの IP アドレスが 32 ビットのビット列が格納された
 * addr を引数にとり、192.168.196.126 のような人間が見やすい形に
 * 変換して文字型変数 str に格納し、これを戻り値とする。
 */
char *addr_client(addr)
struct in_addr *addr;
{
    static char str[STRMAX];
    union {
    u_long    l;
    unsigned char    c[4];
    } u;
    struct hostent *ent;
    u.l = addr→s_addr;
    sprintf(str,"%d.%d.%d.%d",u.c[0],u.c[1],u.c[2],u.c[3]);
    return str;
}
```

```
————————関数 dowrite————————
int dowrite(pair)
Pair *pair;
{
        :
        :

/* 変数 Eproto に 0 が入っていた場合は estone は stone と同じ動作をする。*/
      if(Eproto == 0){
        len = send(pair→sd,&pair→buf[pair→start],pair→len,0);

/* プログラムを動作させる際に'-P' オプションが指定されていた場合、
 * 変数 Eproto に 1 が入っており、クライアントマシンもしくは POP3 が動作している
 * 電子メールサーバへ情報を中継するとともに、関数 pop_estone を呼び出し、
 * 中継している認証情報を指定されたファイル内へ書き込む。
 */
      }else if(Eproto == 1){
        len = send(pair→sd,&pair→buf[pair→start],pair→len,0);
        pop_estone(pair);

/* プログラムを動作させる際に'-S' オプションが指定されていた場合、
 * 変数 Eproto に 2 が入っており、SMTP が動作している電子メールサーバへ
 * 情報を中継する判別する変数 smtp_estone を呼び出し、
 * 送信元電子メールアドレスに偽った記述がないか判別する。
 */
      }else if(Eproto == 2){
        len = smtp_estone(pair);
      }

        :
        :
}
```

```
————————-関数 pop_estone————————-
/* pop_estone は、中継している POP3 の情報の中から、
 * ユーザ名を取り出し、ゲートウェイ内の指定されたファイルへ、
 * 中継した時刻・クライアントマシンの IP アドレス・ユーザの電子メールアドレスを書き込む。
 * また、pair->check と pair->pair->check は中継の進行状況を把握するための変数である。
 */
void pop_estone(pair)
Pair *pair;
{
    int n;
    FILE *auth_file;
    time_t the_time;
    char run_timer[130];

/* pair->check が 0 の場合で、クライアントマシンから、
 * 'USER ユーザ名' が送られてきた場合、pair → user にユーザ名を入れ、
 * 中継の進行状況を把握するために、
 * pair->check と pair->pair->check に 1 を入れる。
 */
    if(pair→check == 0){
      if(pair→buf[pair→start] == 'U'){
        if(pair→buf[pair→start+1] == 'S'){
          if(pair→buf[pair→start+2] == 'E'){
            if(pair→buf[pair→start+3] == 'R'){
              if(pair→buf[pair→start+4] == ' '){
                n = 5;
                while((pair→buf[pair→start+n] ≠ '\xd') && (pair→buf[pair→start+n] ≠ '\xa')){
                  pair→user[n−5] = pair→buf[pair→start+n];
                  n++;
                }
                pair→user[n−5] = '\0';
                pair→check = pair→pair→check = 1;
              }
            }
          }
        }
      }

/* pair->check が 1 の場合で電子メールサーバから '+OK' が送られてきた場合、
 * 次にユーザからパスワードが送られてくることを記すために、
 * pair->check と pair->pair->check に 2 を入れる。
 * また、'+OK' でない場合は、pair->check と pair->pair->check に 0 を入れる。
 */
    }else if(pair→check == 1){
      if(pair→buf[pair→start] == '+'){
```

```
        pair→check = pair→pair→check = 2;
      }else{
        pair→check = pair→pair→check = 0;
      }

/* pair->check が 2 の場合で、クライアントマシンから、
 * 'PASS パスワード' が送られてきた場合、
 * 次に、電子メールサーバからユーザ認証が成功したかの
 * メッセージが返ってくることを記すために、
 * pair->check と pair->pair->check に 3 を入れる。
 */
    }else if(pair→check == 2){
      if(pair→buf[pair→start] == 'P'){
        if(pair→buf[pair→start+1] == 'A'){
          if(pair→buf[pair→start+2] == 'S'){
            if(pair→buf[pair→start+3] == 'S'){
              if(pair→buf[pair→start+4] == ' '){
                pair→check = pair→pair→check = 3;
              }
            }
          }
        }
      }

    /* pair->check が 3 の場合で電子メールサーバから '+OK' が送られてきた場合は、
     * ユーザの受信チェックの際の認証が完了したということである。
     */
    }else if(pair→check == 3){
      if(pair→buf[pair→start] == '+'){

        /* 現在の時刻を調べる */
        the_time = time((time_t *)0);

        /* '時刻 クライアントマシンの IP アドレス ユーザの電子メールアドレス'
         * を宣言部分で指定されたファイルへ書き込む。
         */
        auth_file=fopen(auth_file_pass,"a");
      fprintf(auth_file,"%ld %s %s@%s\n",the_time,pair→ip_addr,pair→pair→user,mail_serv);
      fclose(auth_file);

        /* ファイルへ書き込んだ情報を一定時間後に消去するため
         * のプログラムを動作させる。
         */
        sprintf(run_timer,"%s &",timer_prog);
      system(run_timer);
```

```
        /* 次に syslogd に POP3 での認証をしたことを伝える。 */
        if(Syslog){
          openlog("estone",LOG_PID,LOG_MAIL);
          syslog(LOG_INFO,"POP3 %s %s@%s",pair→ip_addr,pair→pair→user,mail_serv);
          closelog();
        }

      }

      /* ユーザ認証手続きが終了したことを記すため、
       * pair->check と pair->pair->check へ 4 を入れる。
       */
      pair→check = pair→pair→check = 4;

    }
     return;
}
```

3.3.3 なりすましの判別

ユーザが電子メールを送信する際に、中継プログラムは以下の順でなりすましの判別を行う。

1. 中継プログラムがクライアントマシンからの接続を受け付けた時点で、クライアントマシンの IP アドレスを外部変数に書き込む。
2. クライアントマシンから'MAIL FROM:<ユーザの送信元電子メールアドレス>' と送られてきたら、ユーザの送信元電子メールアドレスを外部変数に書き込む。
3. ゲートウェイマシンに保存されているファイルにクライアントマシンの IP アドレスとユーザの電子メールアドレスの一致するものが存在するかどうか調べる。
4. 一致するものがある場合は、中継を続ける。
5. 一致するものがない場合は、電子メールサーバへ'QUIT' と送り、さらにクライアントマシンへ'451 Authorization Failed' と送り中継を中断する。

以下に stone へ追加した estone のプログラムソースのなりすましを判別する部分を示す。

——————-関数 smtp_estone——————

```
/* 関数 smtp_estone は、ユーザの送信元電子メールアドレスに
 * 偽りがないかを確認するための関数である。
 * 'MAIL FROM:< ユーザの電子メールアドレス >' が送られてくるまで
 * 電子メールサーバとクライアント間を中継し、
 * 'MAIL FROM:< ユーザの電子メールアドレス >' が送られてきた段階で、
 * ユーザの電子メールアドレスに偽りがあるかを判別し、
 * 偽りがない場合は中継を続行、偽りがある場合は中継を中断する。
 */
int smtp_estone(pair)
Pair *pair;
{
    int m,n,auth_check,len;
    char mail_from[11],mail_addr[240],ip_mail_addr[256],temp_line[256];
    FILE *auth_file;

      /* pair → check が 0 の場合で'MAIL FROM:' が送られてきたかを調べる */
      if(pair→check == 0){

        /* 先頭の文字列が'M' ではなく明らかに'MAIL FROM:' で
         * ない場合は、中継を続行する
         */
        if(pair→buf[pair→start] ≠ 'M'){
          len = send(pair→sd,&pair→buf[pair→start],pair→len,0);

        /* 先頭の文字列が'M' の場合は、'MAIL FROM:' の可能性があるので
         * 調べる。
         */
        }else{
          n = 0;
```

```
  while(n < 10){
  mail_from[n] = pair→buf[pair→start+n];
    n++;
  }
  mail_from[n] = '\0';
  if(strcoll(mail_from,"MAIL FROM:")){

    /* 'MAIL FROM:' ではない場合は中継を続行 */
  len = send(pair→sd,&pair→buf[pair→start],pair→len,0);

  /* 'MAIL FROM:' の場合、変数 mail_addr に
   * 送信元電子メールアドレスを書き込む。
   */
}else{
    while(pair→buf[pair→start+n] ≠ '\x3c'){
      n++;
    }
    n++;
    m = 0;
    while(pair→buf[pair→start+n] ≠ '\x3e'){
      mail_addr[m] = pair→buf[pair→start+n];
      m++;
      n++;
    }
    mail_addr[m] = '\0';

    /* ゲートウェイ内の指定したファイルにクライアントの IP アドレスと
     * 送信元電子メールアドレスが一致するものがあるかを判別し、
     * 一致するものがある場合には、auth_check をインクリメントする。
     */
    sprintf(ip_mail_addr,"%s %s",pair→ip_addr,mail_addr);
    auth_check = 0;
    auth_file=fopen(auth_file_pass,"r");
    while((fgets(temp_line,128,auth_file) ≠ NULL) && (auth_check == 0)){
      temp_line[strlen(temp_line)−1] = '\0';
      if(strstr(temp_line,ip_mail_addr) ≠ NULL){
        if(strcoll(strstr(temp_line,ip_mail_addr),ip_mail_addr) == 0){
          auth_check++;
      }
      }
    }
    fclose(auth_file);

      /* auth_check が 0 でない場合は、ユーザの送信元電子メールアドレスが
       * 正しいということなので、中継を続行するとともに、
```

```
        * syslog へ送信電子メールを中継するメッセージを伝える。
        * また、ユーザの電子メールの中継を以後も続行することを記すために、
        * pair->check と pair->pair->check に 1 を入れる。
        */
       if(auth_check ≠ 0){
        len = send(pair→sd,&pair→buf[pair→start],pair→len,0);
        pair→check = pair→pair→check = 1;
        if(Syslog){
         openlog("estone",LOG_PID,LOG_MAIL);
         syslog(LOG_INFO,"SMTP %s",ip_mail_addr);
         closelog();
        }

       /* auth_check が 0 の場合は、ユーザの送信元電子メールアドレスに
        * 偽りがあるということなので、中継を中断するために、
        * 電子メールサーバへ'QUIT' を送るとともに、
        * syslog へ送信電子メールを中継を中断したメッセージを伝える。
        * また、ユーザの電子メールの中継を以後は中断することを記すために、
        * pair->check と pair->pair->check に 2 を入れる。
        */
       }else{
        send(pair→sd,"QUIT\xd\xa",6,0);
        len = pair→len;
        pair→check = pair→pair→check = 2;
        if(Syslog){
         openlog("estone",LOG_PID,LOG_MAIL);
         syslog(LOG_INFO,"AUTH ERR %s",ip_mail_addr);
         closelog();
        }
       }
     }
    }
   /* pair->check が 1 なので、送信メールの中継を続行する。 */
   }else if(pair→check == 1){
   len = send(pair→sd,&pair→buf[pair→start],pair→len,0);

   /* pair->check が 2 なので、送信メールの中継を中断するために、
    * クライアントマシンへ'451 Authorization Failed' と送る。
    */
   }else if(pair→check == 2){
    send(pair→sd,"451 Authorization Failed\xd\xa",26,0);
    len = pair→len;
   }
   return len;
}
```

3.3.4 認証情報の削除

さらに、この作成した中継プログラムは、認証情報を書き込んだ一定時間経過後にこれを消去する。ユーザがそのクライアントマシンで電子メールを着信チェックしたことのある他のユーザの電子メールアドレスを用いて電子メールを送信することを防ぐためである。本研究ではこの一定時間を 15 分 (900 秒) とした。

以下に estone のプログラムソースの認証情報をゲートウェイマシンのファイルから削除するために作成した部分を示す。

```
————————————estone_timer.c————————————-
/* estone_timer.c は、POP3 を中継している estone から呼び出される
 * プログラムで認証情報が書き込まれたファイルから一定時間
 * が経過した情報を削除する。
 */
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#include<unistd.h>

#define POP_BEFORE_SMTP_TIME 900

main(void)
{
  FILE *AUTH_Temp,*AUTH_Bak;
  char auth_temp[] = "/etc/estone/pop.auth";
  char auth_bak[] = "/etc/estone/pop.auth.bak";
  char temp_line[128],temp_line2[64],change_bak[128];
  int n;
  time_t the_time;

  /* 呼び出された時刻を記録する */
  the_time = time((time_t *)0);

  /* POP_BEFORE_SMTP_TIME で指定された一定時間待機する。 */
  sleep(POP_BEFORE_SMTP_TIME);

  /* 指定されたファイル内にある情報の内で、
   * このプログラムが呼び出された時刻よりも古いものがあれば
   * ファイルから削除する。
   */
  AUTH_Temp=fopen(auth_temp,"r");
  AUTH_Bak=fopen(auth_bak,"w");
  while(fgets(temp_line,128,AUTH_Temp) ≠ NULL){
    n = 0;
    while(temp_line[n] ≠ ' '){
      temp_line2[n] = temp_line[n];
      ++n;
    }
```

```
    temp_line2[n] = '\0';
    if(the_time < strtol(temp_line2,NULL,0)){
      fprintf(AUTH_Bak,"%s",temp_line);
    }
  }
  sprintf(change_bak,"mv %s %s",auth_bak,auth_temp);
  system(change_bak);
}
```

3.3.5　なりすましへの対応

中継プログラム estone がなりすましを発見した場合に、

1. ユーザが自分自身の電子メールアドレスの設定を間違えた
2. ユーザが着信チェックをせずに、電子メールを送信しようとした
3. ユーザが悪意を持って自分自身の電子メールアドレスを偽った

といった 3 つの可能性が挙げれる。

特に上記の 2 のような場合に、送信メールを最初から書き直す必要があるとユーザに不便を強いることになる。estone は送信メールがなりすましと判別した場合に、ユーザが着信チェックを正常に行ったのちに、直ちに再送できるように、エラー番号 451 をクライアントマシンへ図 11のように通知する。

図 11: なりすましへの対応

4　考察

本研究で提案した方法を研究室内でテストした結果、ユーザが以下のように送信元電子メールアドレスを偽った電子メールを送信することができた。(図は AL-Mail を使用してなりすましを行おうとした場合の結果である。)

- ユーザが電子メールアドレスのユーザ名部分を偽っていた場合 (図 12)

図 12: ユーザ名部分のなりすまし

- ユーザが電子メールアドレスのホスト名部分を偽っていた場合 (図 13)

図 13: ホスト名部分のなりすまし

- 上記の両方を偽っていた場合 (図 14)

図 14: 全く架空のアドレス

また、本研究で提案した方法は、ユーザが電子メールの着信チェックを行った情報を一定時間ゲートウェイマシンに保持することにより成り立っている。よって、ユーザが着信チェックしてから、一定時間内に他のユーザがそのマシンを用いて、そのクライアントを用いて着信チェックを行ったユーザの電子メールアドレスを用いて電子メールを送信することが可能となってしまう。本研究がこの一定時間を 15 分とした根拠は、POP before SMTP を取り入れているインターネット接続プロバイダーの多くが着信チェックした際に得た情報を保持する時間を 15 分としており、本研究で提案した方法も POP before SMTP の概念的な部分を取り入れていることにある。

よって、今後の課題としては、以下が挙げられる。

- 実際に一般ユーザがコンピュータを使用している演習室等での運用テスト
- 本研究のために作成したプログラムの安定性の向上
- ユーザの着信チェック時の情報を蓄える時間の検証

5　まとめ

本研究では、IP マスカレードを導入した環境で電子メール送信者のなりすましが行われた場合に、なりすましを行った人物を特定することが困難であることに着目し、電子メール送信者がなりすましを行えないような方法を提案した。また、本研究で考えた方法を用いて研究室内でテストした段階では、電子メール送信者のなりすましを抑制するのに一定の効果が得られた。今後、実際に一般ユーザが使用している環境での運用テスト、本研究のために作成したプログラムの安定性の向上、ゲートウェイマシンでユーザの電子メール着信チェック時の情報を蓄えておく時間の検証等が課題としてあげられる。

謝辞

本研究を進めていく上で、御協力及び、御指導頂いた藤井 信夫教授、中村 孝講師、大垣 斉講師、そして情報安全工学研究室の各研究生の方々に深く感謝の意を表します。

参考文献

[1] Neil Matthew / Richard Stones, 訳：葛西 重夫, Linux プログラミング, ソフトバンク パブリッシング,1999

[2] 金内 典充 / 今安 正和, 監修：羽山 博, UNIX ネットワークプログラミング, オーム社,1993

[3] Stone のページ,[http://www.gcd.org/sengoku/stone/]

[4] RFC821,Simple Mail Transfer Protocol

[5] RFC1939,Post Office Protocol - Version 3

付録

estone インストールマニュアル

estone は、アプリケーションレベルの TCP と UDP の パケットリピーターである stone を元に作成してものです。estone は、電子メールユーザのなりすましを抑制するとともに、電子メールユーザが送信元電子メールアドレスの設定を間違えて送信することを抑制します。

使用方法

(詳細な使用方法は運用マニュアルを見て下さい)
estone [-P/-S] [-l] [電子メールサーバ]:[電子メールサーバのポート] [esotne を動かすポート]
-P : POP3 中継機能
-S : SMTP 中継機能
-l : syslog 使用

設定方法

1. 任意の場所に認証情報を保存するためのディレクトリを作成します。

   (例) mkdir /etc/estone

2. 必要に応じて estone_timer.c に認証情報を保存するファイルを書き変えて下さい。(既に存在するディレクトリを記述して下さい。) デフォルト値は”/etc/estone/pop.auth”と”/etc/estone/pop.auth.bak”です。

   同じく estone_timer.c の認証情報を破棄する時間を書き変えて下さい。デフォルト値は”900”秒です。以下の行を書き換えます。

   #define POP_BEFORE_SMTP_TIME 900

3. esotne_timer.c をコンパイルします。コンパイル後、任意の場所に置いてください。

   (例) cc estone_timer.c -o estone_timer

   mv estone_timer /usr/local/sbin/

4. estone.c の以下の部分を必要に応じて書き換えてください。これらは、estone.c 内を’zone1-2’ という文字列を検索するとすぐ見つかるはずです。

   認証情報を保存するファイルの場所：デフォルト値は”/etc/estone/pop.auth”

   estone_timer の場所：デフォルト値は”/usr/local/sbin/estone_timer”

   中継する電子メールアドレスのホスト名部分：デフォルト値は”sub.osaka-sandai.ac.jp”

5. estone.c を make します。estone というファイルが出来ます。任意の場所に置いて下さい。

   (例) make linux

   mv estone /usr/local/sbin/

付録 2

estone 運用マニュアル

このマニュアルには、IP マスカレード環境にあるクライアントマシンを使用するユーザが送信元電子メールアドレスを偽って電子メール送信したり、ユーザ自身の電子メールアドレスの設定を間違えて電子メールを送信することを抑制するための運用例等が書かれています。

estone は、以下の 3 つで構成されています。

- estone 本体の POP3 中継 (認証情報の記録機能)
- estone 本体の SMTP 中継 (ユーザの電子メールアドレス判別機能)
- estone_timer(認証情報の削除機能)

estone を使う場合に以下のメリットがあります。

- 各クライアントマシンの電子メールクライアントソフトの設定は何ら変更する必要がない。
- 電子メールサーバの設定も何ら変更する必要がない。(現段階でサポートするプロトコルは POP3 と SMTP です。)

このマニュアルが想定しているのは、以下のマシンがある環境です。

- ローカル IP アドレスしかもたないクライアントマシン (IP マスカレードを使用してインターネットに接続するマシン)
- グローバル IP アドレスとローカル IP アドレスを持つゲートウェイマシン (IP マスカレードを使用する設定がされているマシン)
- グローバル IP アドレスを持つ電子メールサーバ

運用例

ここで示す運用例は、以下のような環境の例です。必要に応じて変更して下さい。

- 電子メールサーバの IP アドレスは、133.64.2.13 である。
- 電子メールサーバの 110 番ポートで POP3 が動作している。
- 電子メールサーバの 25 番ポートで SMTP が動作している。
- ゲートウェイマシンの 8110 番ポートで estone の POP3 中継機能を動作させる。
- ゲートウェイマシンの 8025 番ポートで estone の SMTP 中継機能を動作させる。
- クライアントマシンが属するネットワークは 192.168.1.0/24 である。

1. ゲートウェイマシンの起動時に estone が起動するように、rc.local に以下の記述を追加します。

   /usr/local/sbin/estone -P -l 133.64.2.13:110 8110 &

   /usr/local/sbin/estone -S -l 133.64.2.13:25 8025 &

2. ユーザが電子メールサーバへ接続しようとすると、estone につながるようにゲートウェイマシンを設定するために、rc.local に以下の記述を追加します。

   (Laser5Linux6.0 や、RedHat6.0 等の Linux カーネル 2.2 系を使用している場合)

   /sbin/ipchains -A input -p tcp -s 192.168.1.0/24 -d 0/0 110 -j REDIRECT 8110

   /sbin/ipchains -A input -p tcp -s 192.168.1.0/24 -d 0/0 25 -j REDIRECT 8025

   (VineLinux1.1 等の Linux カーネル 2.0 系を使用している場合)

   /sbin/ipfwadm -I -a accept -P tcp -S 192.168.1.0/24 -D 0/0 110 -r 8110

   /sbin/ipfwadm -I -a accept -P tcp -S 192.168.1.0/24 -D 0/0 25 -r 8025

3. マシンを reboot して実際にクライアントマシンから着信チェックをしてみて下さい。(ただし、telnet で着信チェックを行った際は estone が正常に着信チェックをしたとカウントしない場合があります。)

   /etc/estone/pop.auth に以下のような記述があれば、estone の POP3 中継機能は正常に動作しているといえます。

   950821804 192.168.1.3 s96h021@sub.osaka-sandai.ac.jp

4. 実際にユーザの電子メールアドレスの設定を架空のものにして電子メールを送信して下さい。電子メールクライアントソフト上に、以下のどちらかに該当するようなメッセージが出たら、estone の SMTP 中継部分は正常に動作しているといえます。

   ”Authorization Failed”

   ”451 Authorization Failed”

5. ’-l’ を付けて estone を起動している場合は、/var/log のメール関連のログが残るファイルに以下のような書き込み残っています。上から、POP3 の認証が成功したとき、メールを送信できたとき、メールの送信が estone によって抑制された場合となります。

   Feb 08 05:53:33 sub estone[687]: POP3 192.168.1.3 s96h021@sub.osaka-sandai.ac.jp

   Feb 08 05:21:58 sub estone[688]: SMTP 192.168.1.3 s96h021@sub.osaka-sandai.ac.jp

   Feb 08 05:36:32 sub estone[688]: AUTH ERR 192.168.1.3 eguchi@sub.osaka-sandai.ac.jp