Security of WebAppli&Mail

2007 年 03 月 27 日

Ver. 1.0

目　次

1　WebAppli で使う Mail 送信モジュールの安全な使い方

1.1 本文書の目的

本文書は、Web アプリケーションで利用されている各種メール送信用[モジュール | オブジェクト | 関数]の安全な使い方について検討する。
本文書で対象としている脅威は、主に以下である。

- 単一メールアドレス指定のデータ入力箇所に複数のメールアドレスを入力されて、メール爆弾の踏み台となる脅威
- メールヘッダ・インジェクション
- SMTP インジェクション
- メール文書の切捨て

上記のいずれもが、悪用される確率および発現した場合の深刻度、共に低くセキュテリィ上の深刻な問題点ではない。
しかしながら、それゆえにインパクトが低く、「セキュテリィ」というテーマであまり扱われていないのではないだろうか。一方で[システム開発者 | プログラマ]には、深刻度が低くても必要な情報であると感じ、本文書を作成した。

2　WebAppli で使う Mail 送信モジュールの安全でない使い方によって発生しうる脅威

2.1 メール爆弾の踏み台

一般的な Web アプリケーションの入力フォームで電子メールアドレスを利用者に入力してもらう場合、単一のメールアドレスを入力してもらうことを想定していると考えられる。
この入力画面において、単一のメールアドレスのみならず、複数のメールアドレスが入力可能であった場合、また Web アプリケーションがそれら複数のメールアドレスへ一斉に同報通知してしまう場合、不正行為者は一度の HTTP アクセスによって大量のメールをばらまくことが可能であり、そのような Web ページはメール爆弾の踏み台になる危険性がある。

2.2 メールヘッダ・インジェクション

汚染データ(※)をメールヘッダ(To や From や Subject が代表)の一部として使う場合、改行コードを挿入可能であれば、メールヘッダに任意の情報(例: Reply-To ヘッダなど)が追加される危険性がある。
また、「(改行)(改行)」の後に続けてメールボディ(本文)も挿入可能であれば、メール本文が任意のメールを送信できるという問題にもなる。

[※] 汚染データ
ここでは、利用者から Web アプリケーションへ入力された安全な書式であると検証されていないデータと定義する。「汚染されているかも知れないデータ」とした方がより正確であるような気がする。

2.3 SMTP インジェクション

最終的に、汚染データも SMTP の一部として使われる。よって、細工された汚染データによって、任意の SMTP コマンドを発行させる攻撃である。

2.4 メール文書の切捨て

Web アプリケーションから送信されるメールの多くは、各種管理情報がフッタに記述されることが多い。
(例：「不要な方は××してください」など)
汚染データをメール本文として利用する場合、Web アプリケーション開発者の不注意によって、それらフッタの情報がメール本文に反映されない場合がある。
SMTP においてメール本文を送る場合、メール本文の終了を示すのが「.(ピリオド)」だけの行である。つまり、Web ページが本文が入力可能であり、かつ「.」だけ行をそのまま SMTP として流している場合が該当する。

3　WebAppli で使う Mail 送信モジュールの具体的な安全策

3.1 Perl + Sendmail の場合

UNIX/Linux システムの CGI プログラムの開発言語として Perl が使われてる。
Perl では、OS 上のメール送信コマンド「Sendmail」コマンドを使用してメールを送信することが一般的である。

3．1.1 OS コマンド・インジェクション

OS コマンドを使用するため「OS コマンドインジェクション」には注意する必要がある。

これについては、以下を参照すること。
IPA ISEC セキュアプログラミング講座 [4-2.] Perl の危険な関数
http://www.ipa.go.jp/security/awareness/vendor/programming/a04_02.html

3．1.2 メールヘッダ・インジェクション

「3．1.1 OS コマンド・インジェクション」の続きである。
図 3．1.2-1のように汚染データを「Sendmail」コマンドの引数とはしないで呼び出す方法が一般的であろう。
つまり、「Sendmail」コマンドを「-t」オプションで起動し、メール本文を標準入力として与えて、メール本文中の送信先メールアドレスを sendmail に解釈させメール送信を実行させる。
という手順が一般的であろう(図 3．1.2-1)。

図 3．1.2-1を見てわかるように、汚染データを送信先メールアドレスとしてメール本文のヘッダに配置している。
メールヘッダは改行コードをデリミタにして「ヘッダ名」「：」「ヘッダ値」という書式になっている。

このように、メールヘッダに汚染データを配置させる場合、改行コードを禁止するサニタイズ処理が必要である。

```
${ret} = open(OUT,"|/usr/lib/sendmail -t")
if(${ret} eq undef){ print “error”; exit;}
print OUT "To: <<入力されたメールアドレス>>¥n";
print OUT "From: admin¥@exsample.com¥n";
print OUT "Subject: ThisIsTestMail¥n¥n";
print OUT "This is Test";
close(OUT);
__END__
```

図3．1.2-1：Perl でのメール送信の一般的手法

sendmail コマンドを「-t」オプションで起動すると、

入力データ(メールアドレスなど)をコマンド引数として与える必要がないため

「OS コマンド・インジェクション」を無視できる。

(日本語処理はコードが煩雑になるため省略している)

このコード例の場合では、入力データとして「メールアドレス(改行)メールヘッダ」と

いうような書式で与えられた場合、任意のヘッダを挿入可能となる

頻繁に利用するメールヘッダは、以下であろう。

- **To ヘッダ (送信先メールアドレス)**
- **From ヘッダ (送信元メールアドレス)**
- **Subject ヘッダ (メールタイトル[題])**

当然だが、これ以外のヘッダであったとしても、汚染データをメールヘッダとして利用する場合は、改行コードを禁止するサニタイズ処理を行うことが必要である。

3．1.3　複数メールアドレス指定

「3．1.2 メールヘッダ・インジェクション」の続きである。
図 3．1.2-1を見て分かるように、送信先メールアドレスに汚染データを配置している。
To ヘッダには、カンマ区切りでメールアドレスを複数指定することができる。よって、汚染データによって、そのような指定が行われた場合、複数箇所にメールが送信される結果となる。
このことは一度の Web アクセスで大量のメールを送信できるということになり、不正行為者にとってはメール爆弾の発信基地として利用される危険性がでてくるだろう。

送信先アドレスは複数のメールアドレスを配置できる、という明示的な仕様であれば、本項はセキュテリィ問題ではない。しかし、多くの Web アプリケーションでは、汚染データに複数のメールアドレスを配置可能であるという仕様は稀であると思われる。
このように一つのメールアドレスしか配置できないという暗黙的な仕様に対して、複数のメールアドレスが配置可能になっているのであれば、機能過多であり、この余分な機能がいつセキュテリィ侵害事例として発現するかもしれず、対策しておくことを推奨する。

対策として、入力データが一つのメールアドレスであるかどうかを事前に検証すること。

3．1.4　メール本文の切り捨て

「3．1.3 複数メールアドレス指定」の続きである。

今度は、以下の Perl コードを考えてみる。

```
if (!(open(OUT,"|/usr/lib/sendmail -t"))) { print “error”; exit; }
print OUT "To: support¥@exsample.com¥n";
print OUT "From: admin¥@exsample.com¥n";
print OUT "Subject: ThisIsTestMail¥n¥n";
print OUT <<入力データ>>;
print OUT €¥34¥nThis Mail is Sended WebSystem”；
close(OUT);
```

図3．1.4-1 : Perl でのメール送信の一般的手法
入力データを元に「送信先」を解析する。
メールヘッダは全てハードコードしてあり、入力データはメール本文のボディにのみ配置される
(日本語処理はコードが煩雑になるため省略している)

図 3．1.4-1 では、「OS コマンド・インジェクション」「メールヘッダ・インジェクション」「メール爆弾の踏み台」にもならない。
しかし、このプログラムの入力データとして図 3．1.4-2のようなデータを与えた場合、最終行のメール本文のボディのフッタ部分が、送信メールから削除されるのである。

```
Hello
.
これ以降のデータは、上のピリオドだけの行によって送られるメールに含まれない。
```

図3．1.4-2：図 3．1.4-1に与えるデータ
ピリオドだけの行を与えることでそれ以降のデータを切り捨てる事ができる場合がある

これは、SMTP プロトコルでは「.」(ピリオド)だけの行がメール本文の終了を意味するため、入力データに「.」(ピリオド)だけの行が含まれたことで、メール本文がそこで途切れてしまうからである。

対策は、以下のように「-i」オプションを追加して「Sendmail」コマンドを呼び出すことである。

```
if (!(open(OUT,"|/user/lib/sendmail -t -i"))) { &error('error'); }
print OUT "To: support@exsample.com¥n";
print OUT "From: admin@exsample.com¥n";
print OUT "Subject: ThisIsTestMail¥n¥n";
print OUT <<入力データ>>;
print OUT ¥¥¥¥nThis Mail is Sended WebSystem”；
```

図3．1.4-3：図 3．1.4-1の対策版
「-i」オプションは、標準入力から与えられた「.」(ピリオド)だけの行を
メール本文の最後とは見なさないというオプションである。

3．1.5　メールアドレスの正規表現

メールアドレスがどうかの検証(Validate)には正規表現を使うのが Perl では一般的であろう。

さて、以下の正規表現は汚染データがメールアドレスかどうかを検証することができるであろうか。

```
$mail_address =~ /[a-zA-Z0-9_-]+\@[a-zA-Z0-9_-]+(\.\w+)+/;      (※)
```

実は正しくない。この正規表現では「user01@exsample.com,user02@exsample.com」も正しいメールアドレスとなってしまう。

カンマ区切りで複数のメールアドレスを入力を可能とする仕様の Web **アプリケーションは稀であろう。よって、この正規表現では、一般的(かつ暗黙的な)仕様「メールアドレスという入力ボックスには一つのメールアドレス」を必要十分条件で満たしていないことになる。**
このように、メールアドレスを一つだけ許可するという仕様を満たすには、

```
$mail_address =~ /^[a-zA-Z0-9_-]+\@[a-zA-Z0-9_-]+(\.\w+)+$/;    (※)
```

にする必要がある。最初と最後に「^」と「$」をつけただけであるが、これが結構忘れやすいので注意したい。

本項の主旨は、メールアドレスのチェック用に作成された正規表現のいくつかで、一つのメールアドレスかどうかの判定が抜けているものがあるので注意して欲しい。という事である。

===

(※) : **この正規表現は的確ではない、的確な正規表現については直下の** Web **サイトや書籍を参考にして自らのシステムにマッチする正規表現を探って欲しい。**

蛇足: メールアドレス完全に正しく正規表現することに関して、以下の URL **が参考になる。[システム開発者 | プログラマ]は一読しておくことを推奨する。**

- **メールアドレスに一致する正規表現**
  http://www.tt.rim.or.jp/~canada/comp/cgi/tech/mailaddrmatch/
- **メールアドレスの正規表現**
  http://www.din.or.jp/~ohzaki/perl.htm#Mail

3.2 Win32 + CGI の場合(BlatJ)

Win32 上の Web サーバ上の CGI の場合、サードパーティ製の「BlatJ」を使う場合が一般的であろう。
本項では、ver1.8.2+J2.8 を使用した。
Win32 版の Perl でのメール送信は以下のようなコードで記述できる。

```
#! /usr/local/bin/perl
${fileName} = "temp.txt";
if(open(FILE,">${fileName}")){
        print FILE <<メール本文>>;
        close(FILE);
        $ret = system("blatj.exe ${fileName} -t <<送信先アドレス>> -s <<メールタイトル>> -server <<SMTP
        サーバアドレス>> -f <<送信元アドレス>>");
        print $ret;
}
__END__
```

図3.2-1：「BlatJ」を使ったメール送信のコード例
この図では、メール本文を固定ファイル名のファイルに保存しているが、
実際の場面では、CGI は同時実行される可能性があるので、
乱数を使ってファイル名を生成するなど重ならないようにする必要がある。
また、一時ファイルの保存ディレクトリや、BlatJ.exe からの戻り値の解析
([正常|異常]終了)などにも注意が必要である。

(※) BlatJ の配布元　　http://www.piedey.co.jp/softs/blat.html

3．2.1 OS コマンド・インジェクション

「BlatJ.exe」は、サードパーティ製ではあるが、OS コマンドであるため「OS コマンドインジェクション」には注意する必要がある。
特に、

- 送信先メールアドレス(-t / -c / -b)
- 送信元メールアドレス(-f)
- メールタイトル(-s)

らは、BlatJ.exe コマンドの引数として与える必要があり、汚染データをそれらコマンド引数として配置する場合が多いと思われるため、そのときには注意する必要がある。

「OS コマンド・インジェクション」については、以下を参照すること。
IPA ISEC セキュアプログラミング講座 [4-2.] Perl の危険な関数
http://www.ipa.go.jp/security/awareness/vendor/programming/a04_02.html

3．2.2 複数メールアドレス指定

マニュアルには、To/Cc/Bcc プロパティ共に「,(カンマ)」区切りでの複数メールアドレスの指定が可能であると書かれている。

また、改行コードをデリミタとすることも可能である。この点については、マニュアルからは見あたらなった。

という事で、これらのプロパティに汚染データを配置する場合、「,(カンマ)」および「改行コード(Cr)と(Lf)」について考慮しなければ、複数のメールアドレスを指定される危険性があることに注意する必要がある。

3．2.3 メールヘッダ・インジェクション

BlatJ.exe でメールヘッダ・インジェクション可能な引数は以下である。

名前	可能性
-t	不可
-c	不可
-c	不可
-s	可
-f	不可

これらのプロパティに汚染データを配置させる場合、改行コードを禁止するサニタイズ処理が必

要である。
ただし、「To ヘッダ」などを追加して複数メールの送信はできないようであるが、返信先を指定する「Reply-To」ヘッダなどは追加可能である。

最後に「(改行)(改行)」によってメールボディのインジェクションもヘッダと同様に可能である事を確認した。

3．2.4　SMTP インジェクション

メールヘッダインジェクション可能な変数に対して、「.」だけの行を与えられるかどうか観察した結果、「..」というヒドゥンドットアルゴリズムによってエスケープされることを確認した。

名前	可能性
-t	—
-c	—
-c	—
-s	不可
-f	—

3．2.5　メール本文の切捨て

メール本文に「.」だけの行を与えることで、それ以降のメール本文を切り捨てる攻撃は出来ないことを確認した。

3.3 IIS + ASP の場合(ComBlatJ)

IIS+ASP でメール送信を行う場合、いくつかの方法が一般的にあり、その中の「ComBlatJ」というサードパーティ製 COM を使う場合について検討したい。
本項では、ver1.8.2+J2.8 を使用した。
「ComBlatJ」を使ったメール送信は、図 3.3-1 のようなコードで実装される。

```
Option Explicit
Dim obj
Set obj = WScript.CreateObject("Blat.Send.1")
obj.IsError = False
obj.ErrorMessage = ""
obj.To = <<送信先メールアドレス>>
obj.From = <<送信元メールアドレス>>
obj.Subject = <<メールのタイトル>>
obj.Body = <<メール本文>>
obj.Send
If obj.IsError = True Then
        WScript.Echo obj.ErrorMessage
End If
Set obj = Nothing
WScript.Quit
```

図3.3-1：「ComBlatJ」を使ったメール送信のコード例

(※) ComBlatJ の配布元　http://www.piedey.co.jp/softs/comblat.html

3．3.1 複数メールアドレス指定

マニュアルには、To/Cc/Bcc プロパティ共に「,(カンマ)」区切りでの複数メールアドレスの指定が可能であると書かれている。

また、改行コードをデリミタとすることも可能である。この点については、マニュアルからは見あたらなった。

という事で、これらのプロパティに汚染データを配置する場合、「,(カンマ)」および「改行コード(Cr)と(Lf)」について考慮しなければ、複数のメールアドレスを指定される危険性があることに注意する必要がある。

3．3.2　メールヘッダ・インジェクション

ComBlatJ でメールヘッダ・インジェクション可能なプロパティは以下である。

名前	可能性
To	不可
Cc	不可
Bcc	不可
Subject	可能
From	可能
Impersonate	不可
ExtraHeader	可能(マニュアルに可能である事が明記済)
Organization	

これらのプロパティに汚染データを配置させる場合、改行コードを禁止するサニタイズ処理が必要である。
ただし、「To ヘッダ」などを追加して複数メールの送信はできないようであるが、返信先を指定する「Reply-To」ヘッダなどは追加可能である。

最後に「(改行)(改行)」によってメールボディのインジェクションもヘッダと同様に可能である事を確認した。

3．3.3　SMTP インジェクション

メールヘッダインジェクション可能な変数に対して、「.」だけの行を与えられるかどうか観察した結果、「..」というヒドゥンドットアルゴリズムによってエスケープされることを確認した。

名前	可能性
To	―
Cc	―
Bcc	―
Subject	**不可**
From	**不可**
Impersonate	―
ExtraHeader	**不可**
Organization	

3．3.4　メール本文の切捨て

メール本文に「.」だけの行を与えることで、それ以降のメール本文を切り捨てる攻撃は出来ないことを確認した。

3.4 IIS + ASP の場合(CDO for NTS)

IIS+ASP でメール送信を行う場合、いくつかの方法が一般的にあり、Microsoft ExchangeServer に付属する CDO「Microsoft Collaboration Data Objects」のサブセット版「Microsoft Collaboration Data Objects for WindowsNTServer」を使う場合について検討したい。
本項では、Microsoft-Windows2000Server SP4 上で実験した。

「CDO for NTS」を使ったメール送信は、図 3.4-1 のようなコードで実装される。

```
Option Explicit
Dim obj
Set obj = WScript.CreateObject("CDONTS.NewMail")
obj.From = <<送信元メールアドレス>>
obj.To = <<送信先メールアドレス>>
obj.Subject = <<メールタイトル>>
obj.Body = <<メール本文>>
obj.Send
Set Obj = Nothing
Wscript.Quit
```

図3.4-1：「CDO for NTS」を使ったメール送信のコード例

(※) MS-Windows2003 以降では、CDO for NTS はサポート対象外のようである。

3．4.1 複数メールアドレス指定

To/Cc/Bcc プロパティに「;(セミコロン)」をデリミタで複数アドレスを指定することが可能である。

という事で、これらのプロパティに汚染データを配置する場合、「;(セミコロン)」について考慮しなければ、複数のメールアドレスを指定される危険性があることに注意する必要がある。

3．4.2　メールヘッダ・インジェクション

CDO for NTS でメールヘッダ・インジェクション可能なプロパティは以下である。

名前	可能性
To	可能
Cc	可能
Bcc	可能
Subject	可能
From	可能

これらのプロパティに汚染データを配置させる場合、改行コードを禁止するサニタイズ処理が必要である。
ただし、「To ヘッダ」などを追加して複数メールの送信はできないようであるが、返信先を指定する「Reply-To」ヘッダなどは追加可能である。

3．4.3　メール本文の切捨て

メール本文に「.」だけの行を与えることで、それ以降のメール本文を切り捨てる攻撃は出来ないことを確認した。

3.5 IIS + ASP の場合(BASP21)

IIS+ASP でメール送信を行う場合、いくつかの方法が一般的にあり、その中の「BASP21」というサードパーティ製 COM を使う場合について検討したい。
「BASP21」を使ったメール送信は、図 3.5-1 のようなコードで実装される。
本項では、ver2003/10/10 を使用した。

```
Option Explicit
Dim obj
Dim errStr
Set obj = WScript.CreateObject("BASP21")
errStr = obj.SendMail(<<SMTPServ>>,<<送信先メールアドレス>>,<<送信元メールアドレス>>,<<メールタイトル>>,<<メール本文>>,"")
Set obj = Nothing
WScript.Quit
```

図3.5-1：「BASP21」を使ったメール送信のコード例

(※) BASP21 の配布元　http://www.hi-ho.ne.jp/babaq/basp21.html

3．5.1 複数メールアドレス指定

マニュアルには、第二引数の送信先アドレスの指定において、タブ区切りの複数メールアドレスの指定が可能であると書かれている。

という事で、これらのプロパティに汚染データを配置する場合、「タブ」について考慮しなければ、複数のメールアドレスを指定される危険性があることに注意する必要がある。

3．5.2　メールヘッダ・インジェクション

BASP21 でメールヘッダ・インジェクション可能なプロパティは以下である。

名前		可能性
To	第二引数	可能(SMTP コマンド)
From	第三引数	可能(SMTP コマンド)
Subject	第四引数	可能

これらのプロパティに汚染データを配置させる場合、改行コードを禁止するサニタイズ処理が必要である。

最後に「(改行)(改行)」によってメールボディのインジェクションもヘッダと同様に可能である事を確認した。

3．5.3　SMTP インジェクション

以下のプロパティで SMTP インジェクションが可能である。

名前		可能性
To	第二引数	可能
From	第三引数	可能
Subject	第四引数	可能

メールヘッダ・インジェクションと同様、改行コードを禁止するサニタイズ処理が必要である。

3．5.4　メール本文の切捨て

メール本文に「.」だけの行を与えることで、それ以降のメール本文を切り捨てる攻撃は出来ないことを確認した。

3.6 .NET Framework2.0 の場合

.NET Framework2.0 の場合、System.Net.Mail 空間を使用するのが、一般的であろう。
本項では、Windows2000Sp4 上で、.NET Framework SDK 2.0.50727、C# 8.00.50727.42 (MS-Visual C#2005)を使用した。
図 3.6-1では、さまざまなクラスを使うことなく、SmtpClient クラスの Send メソッドだけを使う場合である。

```
using System;
using System.Net.Mail;
class test{
   static void Main(){
      SmtpClient smtpObj = new SmtpClient("mail.insi.co.jp");
      smtpObj.Send(<<送信元メールアドレス>>,<<送信先メールアドレス>>,<<メールタイトル>>,
<<メール本文>>);
      Console.WriteLine("Done");
 }          }
```

図3.6-1：C#で System.Net.Mail を使ったメール送信のコード例 1

実際には、try～catch 文でエラーをハンドリングしたりする必要があるはず。

3．6.1 複数メールアドレス指定

「,(カンマ)」指定で第二引数の送信先メールアドレスに複数アドレスを指定することが可能である。
という事で、この第二引数に汚染データを配置する場合、「,(カンマ)」について考慮しなければ、複数のメールアドレスを指定される危険性があることに注意する必要がある。

3．6.2　メールヘッダ・インジェクション/SMTP インジェクション

メールヘッダ・インジェクション/SMTP インジェクション共にできないことを確認した。

名前		可能性
送信先	第一引数	不可
送信元	第二引数	不可
タイトル	第三引数	不可

3．6.3　メール本文の切捨て

メール本文がデフォルトで「content-transfer-encoding: quoted-printable」となるため、「.」だけの行を挿入できない(「(改行)」→「=0D=0A」に変換される)。
よって、メール本文を切り捨てるような攻撃は成功しなかった。

(※) これは、バグのようである。KB927858 で修正されるらしいが、現時点ではそのような KB は見つからない。
参考: [.NET 2.0]日本語(ISO-2022-JP) を Content-Transfer-Encoding: 7bit で送信する方法
http://blogs.sqlpassj.org/mitsugi/archive/2006/02/23/16237.aspx

3.7 Java の場合

Java の場合、Sun から提供されている JavaMail API を使うのが一般的であろう。
本項では、JDK 1.5.0_06 + JavaMail 1.4 + JAF 1.1 (Win32)を使用した。

```
import java.util.Properties;
import java.io.UnsupportedEncodingException;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
public class MailTest{
        public static void main(String args[]){
                String SMTPSrv = <<SMTPServer アドレス>>;
                String FromAddr = <<送信元メールアドレス>>;
                String ToAddr = <<送信先メールアドレス>>;
                        // プロパティオブジェクトを構築する
                Properties propObj = System.getProperties();
                propObj.put("mail.smtp.host",SMTPSrv);
                propObj.put("mail.host",SMTPSrv);
                        // プロパティオブジェクトに基づいて、Session オブジェクトをロード
                Session sessionObj = Session.getInstance(propObj,null);
                        // Sessiion オブジェクトに基づいてメッセージオブジェクトをロード
                MimeMessage msgObj = new MimeMessage(sessionObj);
                try{
                        // 送信元メールアドレスを InternetAddress オブジェクトにラップ
                        InternetAddress objFrom = new InternetAddress(FromAddr);
                        // メッセージオブジェクトに送信元を指定
                        msgObj.setFrom(objFrom);
                        // メッセージオブジェクトにメールタイトルを指定
                        msgObj.setSubject("テスト","ISO-2022-JP");
                        // メッセージオブジェクトに本文を指定
```

```
                msgObj.setText("This is test","ISO-2022-JP");
                // 送信先を指定
                msgObj.setRecipients(Message.RecipientType.TO,ToAddr);
                Transport.send(msgObj);          // 送信
                System.out.println("sended¥n");
            }catch(Exception e){    System.out.println("error¥n");    }
        }       }
```

図3.7-1：JavaMail を使ったメール送信のコード例

3．7.1　複数メールアドレス指定

JavaMail API の一般的な使い方として、開発者が入力されたデータから複数アドレスを取得する場合、一度 InternetAddress クラスの配列に展開してから、その配列を setRecipients()メソッドに割り当てる方法が一般的であろう。このようなコードの場合、開発者自身が一つの変数に複数のアドレスが含まれることを想定して JavaMail API を用いているためセキュリティ上の問題はないといえる。
しかしながら、String クラスをそのまま割り当てる方法で setRecipients()メソッドを呼び出すコードを書いた多くの開発者は、String クラスには一つのアドレスだけであると想定しているのではないだろうか(図 3.7-1でもそのような意図で String クラスをそのまま渡している)。このような場合、複数のメールアドレスが含まれているという想定外の現象が起きるためそのまま複数のアドレスへメールが送信されるのであれば、セキュテリィ上の問題である。
実際にテストしてみたところ「,(カンマ)」区切りで複数アドレスにメールを送信することができることを確認した。
(「;(セミコロン)」「改行(¥r¥n)」「 (スペース)」は AddressException が発生した)

という事で、汚染データをメールアドレスとして用いる場合「,(カンマ)」について考慮しなければ、複数のメールアドレスを指定される危険性があることに注意する必要がある。

JavaMail API の場合は、折角アドレスを解析してくれる InternetAddress.parse()メソッドかあるので、それを使うのが最も適切であると考える。一度 InternetAddress クラスの配列に展開してから、InternetAddress クラス配列の 0 番目のメールアドレスだけ使うというコードを追加することで、InternetAddress クラスがメールアドレスを解析する機能を使って、入力データを解析できるのではないだろうか。

```
InternetAddress objAddr[] = InternetAddress.parse(ToAddr);
msgObj.setRecipients(Message.RecipientType.TO,objAddr[0].getAddress());
```

図3．7.1-1：図 3.7-1のコードの「// 送信先を指定」の直下の一行を
このように修正すれば、メールは一度の呼び出しで一通だけ送信されるように制限できる

3．7.2 メールヘッダ・インジェクション

JavaMail API の MimeMessage クラスでメールヘッダ・インジェクション可能なメソッドは以下である。

メソッド名	可能性
setSubject()	可能
setHeader()	可能
addHeader()	可能
setRecipients()	不可

これ以外の setXXXX メソッドでも可能かも知れないが、例えば setSentDate()の引数に外部から渡される汚染されたデータが与えられる場面を考えられないため、実験をしていない。
(当然であるが、そのような場合に直面した場合は、読者自らが実験し安全な方法を探って欲しい)

ただし、「To ヘッダ」などを追加して複数メールの送信はできないようであるが、返信先を指定する「Reply-To」ヘッダなどは追加可能である。

最後に「(改行)(改行)」によってメールボディのインジェクションもヘッダと同様に可能である事を確認した。

3．7.3　SMTP インジェクション

メールヘッダインジェクション可能な変数に対して、「.」だけの行を与えられるかどうか観察した結果、「..」というヒドゥンドットアルゴリズムによってエスケープされることを確認した。

メソッド名	可能性
setSubject()	不可
setHeader()	不可
addHeader()	不可
setRecipients()	―

3．7.4　メール本文の切捨て

メール本文に「.」だけの行を与えることで、それ以降のメール本文を切り捨てる攻撃は出来ないことを確認した。

3.8 PHP + mb_send_mail() [Win32]の場合

PHP の場合、メール送信が標準機能として実装されており、それを使うのが一般的であろう。
本項では、PHP5.2.0/5.1.1 for Win32 を使用した。
日本語環境であれば、mb_send_mail() 関数を利用するのが一般的であろう(図 3.8-1)。
送信元メールアドレスは、php.ini で指定するか、オプションの第四引数で指定する。

```
<?
mb_send_mail(<<送信先メールアドレス>>,<<メールタイトル>>,<<メール本文>>,<<拡張ヘッダ>>)
?>
```

図3.8-1 : PHP **のメール送信のコード例**

3．8.1 複数メールアドレス指定

マニュアルには「,(カンマ)」指定で第一引数の送信先メールアドレスに複数アドレスを指定することが可能である。
という事で、この第一引数に汚染データを配置する場合、「,(カンマ)」について考慮しなければ、複数のメールアドレスを指定される危険性があることに注意する必要がある。

3．8.2 メールヘッダ・インジェクション

mb_send_mail() 関数の第四引数には、その他のヘッダを指定することができる。
マニュアルによれば、第四引数には、改行コードで複数のヘッダを指定可能であると明記している。
よって、第四引数に汚染データを配置させる場合、改行コードを禁止し、不正行為者が望む任意のヘッダを挿入されないようサニタイズ処理が必要である。

第二引数(メールタイトル)については改行コード込みで MIME エンコードされたため、任意のヘッダを挿入することはできなかった。

しかし、AL-Mail32 ver1.13a では、あたかも任意ヘッダが与えられたかのように振舞った。(これは、メールクライアントがサブジェクトの MIME デコードを行った際に改行コードを評価してしまうからで、メール送信モジュールの側だけの問題でもないだろう)。
よって、そのようなメールクライアントをターゲットとした場合のみ可能性がある(この場合はボディの挿入も可能である)。
また、mail() 関数では、MIME エンコードはされないものの、改行コードが別の文字に置換され、改行コードを使ったメールヘッダインジェクションはできないことを確認した。

mb_send_mail() 関数でメールヘッダ・インジェクション可能なプロパティは以下である。

名前		可能性
To	第一引数	不可
Subject	第二引数	不可 (改行コード込で MIME エンコード、または改行コードを別文字へ置換)
拡張ヘッダ	第四引数	可能(マニュアルに可能である事が明記済)

ただし、「To ヘッダ」などを追加して複数メールの送信はできないようであるが、返信先を指定する「Reply-To」ヘッダなどは追加可能である。

最後に第四引数に対して「(改行)(改行)」によってメールボディのインジェクションか可能かどうか調査したが、「(改行)(改行)」は一つの「(改行)」に置換されるため、メールボディを与えることはできないことを確認した。

3．8.3　メール本文の切捨て

メール本文に「.」だけの行を与えることで、それ以降のメール本文を切り捨てる攻撃は出来ないことを確認した。

しかし、第四引数の拡張ヘッダの指定で「.」の指定が可能であった。
この現象を用いて、メール本文を切り捨てたメールを送信可能である。

対策としては、mb_send_mail() 関数の第四引数に汚染データを配置する場合、行頭の「.」を「..」にするヒドゥンドットアルゴリズムを用いてエスケープすることである。

3.9 PHP + mb_send_mail() [Linux/Unix]の場合

PHP の場合、メール送信が標準機能として実装されており、それを使うのが一般的であろう。
本項では、PHP5.2.0 (configure オプション = “--enable-mbstring”) (CentOS4.4) を使用した。
php.ini は php.ini-dist をベースに使い、「sendmail_from」だけを設定した。
「sendmail_path」は指定していない状態(デフォルト値の “sendmail -t -i”)で実験した

日本語環境であれば、mb_send_mail() 関数を利用するのが一般的であろう(図 3.9-1)。
mb_send_mail() 関数(mail()関数)は、Win32 版と Linux/Unix 版では実装が異なるため、Win32 版とは別に本項では Linux/Unix 版で実験を行った。
送信元メールアドレスは、php.ini で指定するか、オプションの第四引数で指定する。

```
<?
mb_language("Ja");
mb_internal_encoding("EUC-JP");
mb_send_mail(<<送信先メールアドレス>>,<<メールタイトル>>,<<メール本文>>,<<拡張ヘッダ>>)
?>
```

図3.9-1 : PHP **のメール送信のコード例**

3．9.1 複数メールアドレス指定

マニュアルには「,(カンマ)」指定で第一引数の送信先メールアドレスに複数アドレスを指定することが可能である。
またそれ以外に「 (半角スペース)」「改行コード」が、デリミタとして使用可能であった。
という事で、この第一引数に汚染データを配置する場合、「,(カンマ)」「 (半角スペース)」「改行コード」について考慮しなければ、複数のメールアドレスを指定される危険性があることに注意する必要がある。

3．9.2 メールヘッダ・インジェクション

mb_send_mail() 関数の第四引数には、その他のヘッダを指定することができる。
マニュアルによれば、第四引数には、改行コードで複数のヘッダを指定可能であると明記している。
よって、第四引数に汚染データを配置させる場合、改行コードを禁止し、不正行為者が望む任意のヘッダを挿入されないようサニタイズ処理が必要である。

同様に第二引数(メールタイトル)についても改行コードを付与することで任意のヘッダを挿入することが可能であった。

mb_send_mail() 関数でメールヘッダ・インジェクション可能なプロパティは以下である。

名前		可能性
To	第一引数	不可
Subject	第二引数	可能 メールボディも可能
拡張ヘッダ	第四引数	可能(マニュアルに可能である事が明記済) メールボディも可能

ただし、「To ヘッダ」などを追加して複数メールの送信はできないようであるが、返信先を指定する「Reply-To」ヘッダなどは追加可能である。

最後に第四引数に対して「(改行)(改行)」によってメールボディのインジェクションか可能かどうか調査したが、Win32 版と異なり、メールボディを与えることが可能であることを確認した。
第四引数に汚染データを配置する場合「(改行)(改行)」が入らないようなサニタイズ処理が必要である。
そもそも汚染データを拡張ヘッダとして利用する場合、一つのヘッダとして使う場合が大部分だろう。このような状況を想定した場合、「改行コード」は禁止されるはずであり(でなければ他のヘッダをインジェクションされる危険性がある)、そのような対策を何がしかの方法で施していれば、大概の場合で「(改行)(改行)」のインジェクションは阻止されるだろう。

3．9.3 メール本文の切捨て

メール本文に「.」だけの行を与えることで、それ以降のメール本文を切り捨てる攻撃は出来ない

ことを確認した。

また、拡張ヘッダを指定する第二、第四引数を使って、メールボディを指定可能であったが、「.」が「..」に置換されメール本文の切捨て攻撃はできないことを確認した。

3.10　PHP + Mail.php(PEAR)の場合

PHP の場合、メール送信が標準ライブラリ(PEAR)機能として提供されており、それを使うのも一般的であろう。

本項では、以下の環境で実験した

- PHP5.2.0 for Win32
- Mail 1.1.14
- Auth_SASL 1.0.2
- Net_SMTP 1.2.8
- Net_Socket 1.0.6
- PEAR 1.4.5

ソースコードは以下のようになる(図 3.10-1)。

```
<?
require_once "Mail.php";
$params['host'] = '127.0.0.1';
$params['port'] = '25';
$toAddr = <<送信先メールアドレス>>;
$headers['From'] = <<送信元メールアドレス>>;
$headers['Subject'] = <<メールタイトル>>;
$body = <<メール本文>>;
$mail_object =& Mail::factory('SMTP',$params);
$flg = $mail_object->send($toAddr,$headers,$body);
if($flg == true){     echo "successe\n";    }
echo "end\n";
?>
```

図3.10-1：PHP&PEAR **のメール送信のコード例**

3．10.1 複数メールアドレス指定

マニュアルには配列、または「,(カンマ)」指定で第一引数の送信先メールアドレスに複数アドレ

スを指定することが可能である。
という事で、この第一引数に汚染データを配置する場合、「,(カンマ)」について考慮しなければ、複数のメールアドレスを指定される危険性があることに注意する必要がある。

3．10.2 メールヘッダ・インジェクション

send() メソッドの第ニ引数には、その他のヘッダを指定することができる。
マニュアルによれば、第ニ引数には、連想配列にて指定すると明記している。
配列名がヘッダ名になり、配列の値がヘッダ値になると書かれている。
連想配列の値に改行コードを挿入した結果、改行コード以降のデータがメールデータから消されることを確認した。ただし、改行コードは有効であり、そのヘッダ値をセットする以降のコードでセットした連想配列の値は、メールのボディ部分に配置される現象を確認した。
任意のヘッダ挿入はできないが、PHP プログラマが想定しているヘッダをメールのボディ部分にずらすことができるということである。
よって、攻撃の脅威は高くないが、send()メソッドの第二引数に汚染データを配置する際に改行コードのサニタイズ処理が必要である。

send() メソッドでメールヘッダ・インジェクション可能なプロパティは以下である。

名前		可能性
To	第一引数	不可(メールアドレスとして正しくないと判断した場合「SMTP RSET」を SMTP サーバへ送る)
拡張ヘッダ	第二引数	不可(ただし、以降のヘッダをボディ部へずらすことが可能)

3．10.3 メール本文の切捨て

メール本文に「.」だけの行を与えることで、それ以降のメール本文を切り捨てる攻撃は出来ないことを確認した。

また、Mail::factory の第一引数に「sendmail」を指定した場合、第二引数の「sendmail_args」

のデフォルト値は「-i」である。このことから PHP から sendmail を呼び出した場合、メール本文の途中切捨て攻撃は起きないであろう。
PHP プログラマが明示的に「sendmail_args」を指定する場合は、「-i」を忘れてはいけない。

このことは、php.ini の「mail function」タブの「sendmail_path」においても同様である。
「sendmail_path」のデフォルトは「sendmail -t -i」と「-i」が付与しているため、デフォルト状態では、メール本文途中切捨て攻撃はおきないもとと思われる。
また、PHP プログラマが php.ini を編集する際、「sendmail」コマンドのオプションとして「-i」を指定することを忘れてはいけない。

3.11 socket を直接操作する場合

アプリケーションの開発環境に、求めている機能を満たすメール送信関数がない、などの理由で、socket(winsock)を直接操作する(つまり、自前で SMTP コマンドを操作する)ことでメールを送信する場合もあるかと思う。

そのような場合、当然であるが、汚染データによって、想定外の SMTP コマンドが発行されないように気をつけたプログラミングが必要である。
利用する SMTP コマンドは以下であろう。

- EHLO <<ホスト名>> または HELO <<ホスト名>> コマンド
- MAIL FROM <<送信元メールアドレス>> コマンド
- RCPT TO <<送信先メールアドレス>> コマンド
- DATA コマンド
- QUIT コマンド

SMTP コマンドの区切りは改行コードである。
よって、メールアドレスなどに汚染データを配置する場合は、改行コードを挿入されないようにサニタイズ処理が必要である。
簡単に改行コード(Cr と Lf)を削ってしまう、またはデータの検証として改行コードが含まれていた場合にはエラー処理へ移行するなどの対策が考えられる。

DATA コマンドの後は、「.(ピリオド)」だけ行までがメールコンテンツ(メールのヘッダとボディ[本文])である。メール本文、メールヘッダに汚染データとして「. (ピリオド)」だけの行がないようにサニタイズ処理を行うこと。
この場合は、SMTP の仕様で行頭の「.(ピリオド)」」は「..(ピリオド二つ)」にエスケープするヒドゥンドットアルゴリズムによるエスケープ処理を実装するのが最も適切な対処方法である。
また、メールヘッダに与える汚染データには改行コードが含まれないようなチェックも必要である。

4　執筆者など

4.1 本文書の免責事項

本文書に記述されてる情報の利用は、読者の責任に帰するものとする。
本文書で行った実験は、執筆者の実験環境で確認したものである。モジュールの細かいバージョン、実験に利用した送信用メールサーバ/受信用メールサーバなどの状況によっては、現象が異なる可能性がある。

4.2 執筆者

■ active@window.goukaku.com

4.3 更新履歴

最初のバージョン：2007 年 03 月 27 日

4.4 本文書の最新バージョン

http://rocketeer.dip.jp/secProg/

4.5 仕様とバグについての考察

例えば送信先メールアドレスを指定する場合に、カンマ区切りなど複数のメールアドレスを指定できる場合が多い。
これについては、マニュアルに明記されている場合が多く、マニュアルに明記されている以上、[関数 | API | モジュール]の利用者(つまりプログラマ)が注意すべき事項であり、セキュテリィホールであるとはいえない。

しかしながら、メールタイトルを指定する場合に改行コードがそのままメール本文に与えられ、結果として他の任意のメールヘッダを指定できる、という現象はセキュテリィホールと言えるのではないだろうか。
セキュテリィホールではなくても、バグという主張はできるのではないだろうか。
しかし、発生確率/深刻度ともに低く見積もられ(筆者自身も低いと考えている輩の一人であるが)、これらのバグ修正には時間がかかるものと思われる。
[関数|API|モジュール]側で対策が行われることが望ましいが、そもそもメールアドレスやメールタイトルの「改行」は不要のものであり、アプリケーション側でサニタイズ処理(削る、別の文字に置換する、エラー処理へ移行する)を行えばよいものである。

4.6 本文書を一般化

筆者自身が本文書を作成する上でおこなった実験の本質的な部分は以下である。

- メール本文は、空行を挟んで「ヘッダ」と「ボディ」に分かれる
- 「ヘッダ」は、一行単位で管理情報(ヘッダ)が配置している(送信元、送信先、タイトル、返信先…)
- SMTP の仕様の中で、メール本文をサーバへ送信する際の、メール本文の終了を示すサインが「.」だけの行である
- SMTP の仕様の中で、MAIL FROM コマンド、RCPT TO コマンドは、送信元メールアドレス、送信先メールアドレスをオプションに持つ
- SMTP の仕様の中で、コマンドの区切りは改行である
- 大部分の Web アプリケーションでは、「送信元」「送信先」「メールタイトル」「メール本文」が Web アプリケーション利用者(不正行為者も含)から与えられる

以上の仕組みを把握した上で

- 「送信先」の複数指定の方法について、マニュアルで明記されているか
- 「ヘッダ」として配置する「送信元」「送信先」「タイトル」に改行を挿入した場合、受信メールのヘッダはどのような状態になるのか
  (改行以降の文字列が別のヘッダとして認識されていないか)
- メール本文途中に「.」だけの行を配置した結果、受信メールの本文ではそれ以降がメール本文から消えていないか。

- **「送信元」「送信先」は、SMTP の「MAIL FROM」「RCPT TO」コマンドのオプションとしても使用するため、それらに改行コード(SMTP コマンドのデリミタ)を与えることで、SMTP のやり取りに入り込めないか。**

以上の観点で実験を行った。本文書以外のメール送信モジュールについても同様な観点で検査すればよいだろう。

以　上