.NET Framework 上の
SMTP Command Injection について

ＮＴＴ コミュニケーションズ株式会社
ＩＴ マネジメントサービス事業部
セキュリティオペレーションセンタ

2011 年 01 月 11 日

Ver. 1.0

1. 調査概要

.NET Framework 上の System.Net.Mail.AlternateView クラス、および System.Net.Mail.Attachment クラスには、SMTP Command Injection の脆弱性がある。
しかし、Microsoft としては、修正プログラムを公開する意思はないということである(次期バージョンである.NET Framework4.5 では修正済みの予定)。
よって、SMTP Command Injection を防止するためには、アプリケーション・プログラマがこれらのクラスを利用する際に、自らエスケープ処理を行わなければならない。
.NET Framework 上で電子メールを取り扱うプログラムを開発しているアプリケーション・プログラマへの注意喚起として、本文書を作成した。

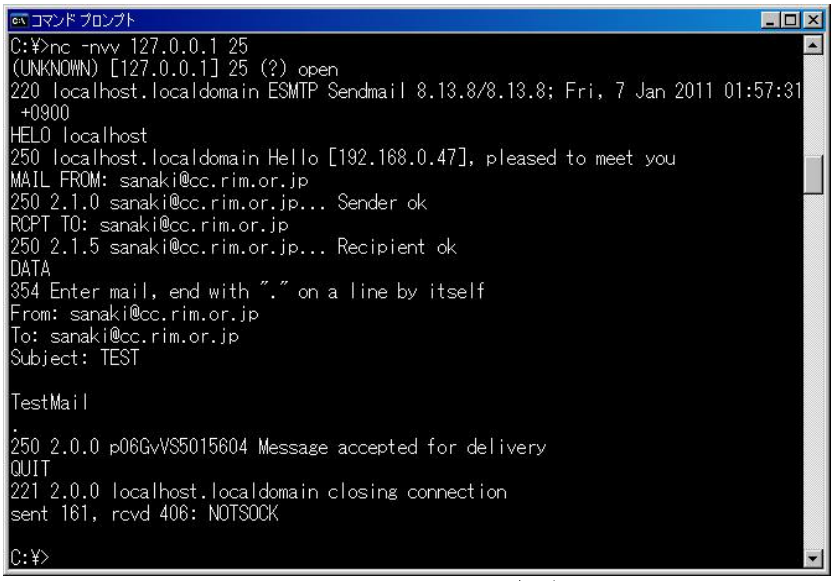
2. SMTP Command Injectionの検証

2.1. .NET Frameworkと電子メール送信処理

.NET Framework 上で、電子メールの送信を行うには、System.Net.Mail.SmtpClient クラスをそのまま使うだけでよい(第6章の15)。
しかし、これだけでは、日本国内で一般的に普及している文字コードが JIS コードである電子メールを送信することができない。
文字コードがJISコードの電子メールを送信するためには、System.Net.Mail.AlternateViewクラスを用いて、System.Net.Mail.MailMessageクラスを構築する必要がある(第6章の16)。
しかし、System.Net.Mail.AlternateView クラスの CreateAlternateViewFromString() メソッドには、メールコンテンツに対してのエスケープが行われていないため、任意の SMTP Command を送り込むことが可能となっている。
同様に、System.Net.Mail.Attachment クラスの CreateAttachmentFromString()メソッド、及び TransferEncoding プロパティを「7bit」にし、Base64 符号化などをしない状態で、ファイルを添付することでも、メールコンテンツのエスケープ漏れにより SMTP Command Injection が可能となっている。

2.2. SMTPコマンドの流れ

SMTP(Simple Mail Transfer Protocol) は、RFC821、RFC2821、そして RFC5321 で定義された、テキストベースで電子メールを送受信するプロトコルである。
メールのコンテンツ(メール自体のヘッダとボディ[本文])は、DATAコマンド後に送信する。その際、行頭の「.(ピリオド)」は「..(ピリオド二個)」にエスケープ(hidden dot algorithm)する必要がある(第6章の11と12)。



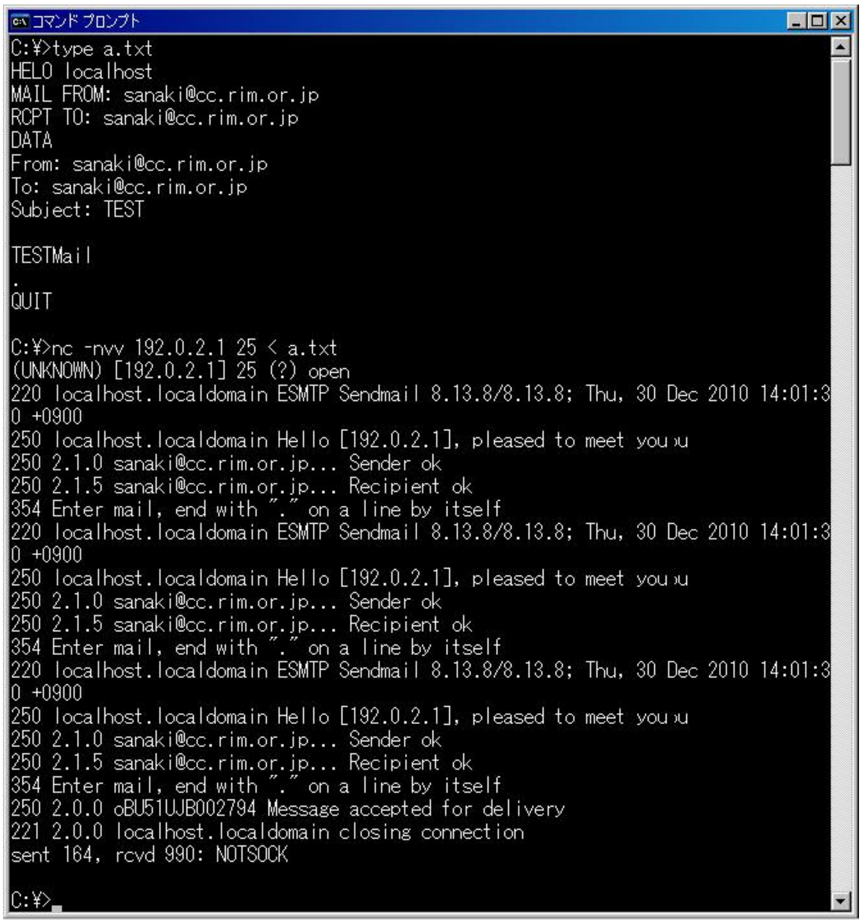
```
C:¥>nc -nvv 127.0.0.1 25
(UNKNOWN) [127.0.0.1] 25 (?) open
220 localhost.localdomain ESMTP Sendmail 8.13.8/8.13.8; Fri, 7 Jan 2011 01:57:31
 +0900
HELO localhost
250 localhost.localdomain Hello [192.168.0.47], pleased to meet you
MAIL FROM: sanaki@cc.rim.or.jp
250 2.1.0 sanaki@cc.rim.or.jp... Sender ok
RCPT TO: sanaki@cc.rim.or.jp
250 2.1.5 sanaki@cc.rim.or.jp... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
From: sanaki@cc.rim.or.jp
To: sanaki@cc.rim.or.jp
Subject: TEST

TestMail
.
250 2.0.0 p06GvVS5015604 Message accepted for delivery
QUIT
221 2.0.0 localhost.localdomain closing connection
sent 161, rcvd 406: NOTSOCK

C:¥>
```

図 2.2-1 : SMTP コマンドの例

「DATA」コマンドは「.」だけ行によって終了される



```
C:¥>type a.txt
HELO localhost
MAIL FROM: sanaki@cc.rim.or.jp
RCPT TO: sanaki@cc.rim.or.jp
DATA
From: sanaki@cc.rim.or.jp
To: sanaki@cc.rim.or.jp
Subject: TEST

TESTMail
.
QUIT

C:¥>nc -nvv 192.0.2.1 25 < a.txt
(UNKNOWN) [192.0.2.1] 25 (?) open
220 localhost.localdomain ESMTP Sendmail 8.13.8/8.13.8; Thu, 30 Dec 2010 14:01:3
0 +0900
250 localhost.localdomain Hello [192.0.2.1], pleased to meet you ou
250 2.1.0 sanaki@cc.rim.or.jp... Sender ok
250 2.1.5 sanaki@cc.rim.or.jp... Recipient ok
354 Enter mail, end with "." on a line by itself
220 localhost.localdomain ESMTP Sendmail 8.13.8/8.13.8; Thu, 30 Dec 2010 14:01:3
0 +0900
250 localhost.localdomain Hello [192.0.2.1], pleased to meet you ou
250 2.1.0 sanaki@cc.rim.or.jp... Sender ok
250 2.1.5 sanaki@cc.rim.or.jp... Recipient ok
354 Enter mail, end with "." on a line by itself
220 localhost.localdomain ESMTP Sendmail 8.13.8/8.13.8; Thu, 30 Dec 2010 14:01:3
0 +0900
250 localhost.localdomain Hello [192.0.2.1], pleased to meet you ou
250 2.1.0 sanaki@cc.rim.or.jp... Sender ok
250 2.1.5 sanaki@cc.rim.or.jp... Recipient ok
354 Enter mail, end with "." on a line by itself
250 2.0.0 oBU51UJB002794 Message accepted for delivery
221 2.0.0 localhost.localdomain closing connection
sent 164, rcvd 990: NOTSOCK

C:¥>_
```

図 2.2-2 : SMTP コマンドの例

SMTP Command Injection のためには、この図のように SMTP の状態を無視して、

TCP のストリームデータをそのまま SMTP コマンドとして扱うメールサーバが必要となる。

Microsoft 社の IIS-SMTP は、状態管理をしているようで、このような方法でメールを送れないようだ。

2.3. 検証プログラム

```
using System;
using System.IO;
using System.Text;
using System.Net.Mail;

class mailTest3{
 static void Main(string[] args){
  int minArgs = 5;
  System.Console.WriteLine("usage:");
  System.Console.WriteLine(" System.Net.Mail.AlternateView#CreateAlternateViewFromString TEST");
  System.Console.WriteLine("  mailTest2.exe 1 <<SMTPSrvHost>> <<FromMailAddress>> <<ToMailAddress>>
<<Subject>>  <<Filepath in MailBody>>");
  System.Console.WriteLine(" System.Net.Mail.Attachment#CreateAttachmentFromString TEST");
  System.Console.WriteLine("  mailTest2.exe 2 <<SMTPSrvHost>> <<FromMailAddress>> <<ToMailAddress>>
<<Subject>> <<Filepath in MailBody>>");
  System.Console.WriteLine(" System.Net.Mail.Attachment - AttachmentFile TEST");
  System.Console.WriteLine("  mailTest2.exe 3 <<SMTPSrvHost>> <<FromMailAddress>> <<ToMailAddress>>
<<Subject>> <<AttachmentFilePathName>>");
  if(0 < args.Length){
   if(0 < minArgs && minArgs < args.Length){
    System.Console.WriteLine("Start");

    String mySMTPSrv = args[1];
    String myFromMailAddress = args[2];
    String myToMailAddress = args[3];
    String myMailSubject = args[4];
    String myMailBody = String.Empty;

    if(args[0] == "1" || args[0] == "2"){
     StreamReader myStreamReader = new StreamReader(args[5],Encoding.GetEncoding("Shift_JIS"));
     myMailBody = myStreamReader.ReadToEnd();
     myStreamReader.Close();
    }

    System.Console.WriteLine("FromMailAddress: " + myFromMailAddress);
    System.Console.WriteLine("  ToMailAddress: " + myToMailAddress);
    System.Console.WriteLine("MailSubject: " + myMailSubject);
    System.Console.WriteLine("SMTP Server: " + mySMTPSrv);
    System.Console.WriteLine("BODY======");
    System.Console.WriteLine(myMailBody);
    System.Console.WriteLine("==========");

    Encoding myEncoding = Encoding.GetEncoding("iso-2022-jp");

    MailMessage myMailMessage = new MailMessage();
    myMailMessage.From = new MailAddress(myFromMailAddress);
    myMailMessage.To.Add(new MailAddress(myToMailAddress));

    myMailMessage.Subject = myMailSubject;

    switch(args[0]){
```

```
      case "1":
       AlternateView myAlternateView =
AlternateView.CreateAlternateViewFromString(myMailBody,myEncoding,System.Net.Mime.MediaTypeNames.Text.Plain);
       myAlternateView.TransferEncoding = System.Net.Mime.TransferEncoding.SevenBit;
       myMailMessage.AlternateViews.Add(myAlternateView);
       break;

      case "2":
       System.Net.Mail.Attachment myAttachment1 =
Attachment.CreateAttachmentFromString(myMailBody,System.Net.Mime.MediaTypeNames.Text.Plain,myEncoding,null);
       myAttachment1.TransferEncoding = System.Net.Mime.TransferEncoding.SevenBit;
       myMailMessage.Attachments.Add(myAttachment1);
       break;

      case "3":
       System.Net.Mail.Attachment myAttachment2 = new
Attachment(args[5],System.Net.Mime.MediaTypeNames.Text.Plain);
       myAttachment2.TransferEncoding = System.Net.Mime.TransferEncoding.SevenBit;
       myMailMessage.Attachments.Add(myAttachment2);
       break;
     }

     SmtpClient mySmtpClient = new SmtpClient();
     mySmtpClient.Host = mySMTPSrv;
     mySmtpClient.Send(myMailMessage);

     System.Console.WriteLine("End");
    }
   }
  }
}
```

図 2.3-1 : mailTest3.csc

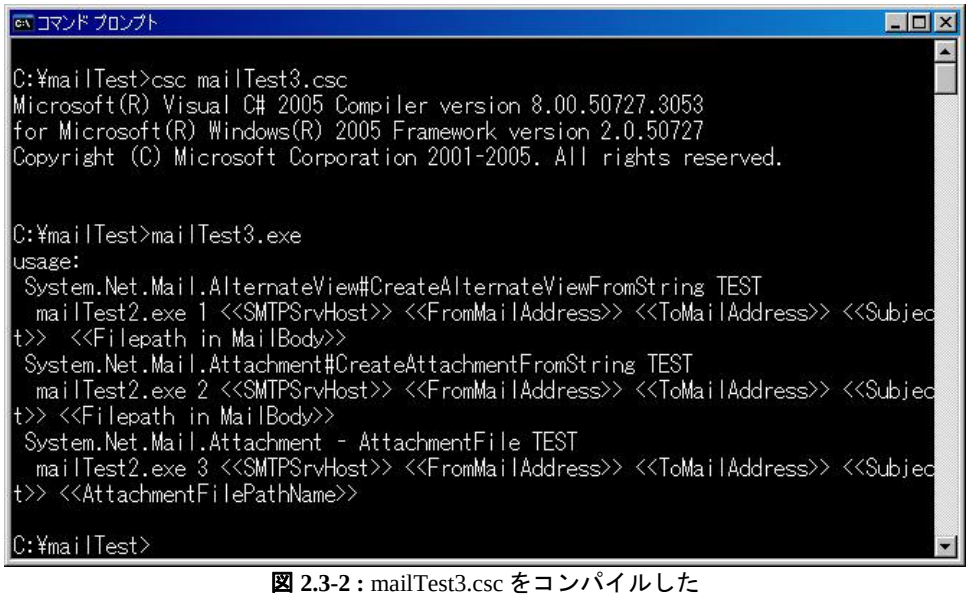


図 2.3-2 : mailTest3.csc をコンパイルした

```
C:¥>nc -nvv -L -p 25
listening on [any] 25 ...
connect to [192.0.2.1] from (UNKNOWN) [192.0.2.2] 1062
220
EHLO vs6sp6
250 ok
MAIL FROM:<sanaki@cc.rim.or.jp>
250 ok
RCPT TO:<sanaki@cc.rim.or.jp>
250 ok
DATA
354 ok
MIME-Version: 1.0
From: sanaki@cc.rim.or.jp
To: sanaki@cc.rim.or.jp
Date: 23 Sep 2010 16:37:24 +0900
Subject: SMTPTest
Content-Type: text/plain; charset=iso-2022-jp
Content-Transfer-Encoding: 7bit

SMTPTestBody
.
250 ok
 sent 39, rcvd 406
listening on [192.0.2.1] 90 ...
```

図 2.3-3：図 2.3-1の実行結果をサーバ側から見たログ。

AlternateViewクラスを使ったメール送信では、最後の「QUIT」コマンドがないようだ

2.4. AlternateView#CreateAlternateViewFromString() の場合

それでは、AlternateView#CreateAlternateViewFromString()の場合である。
図 2.4-2と図 2.4-3の結果から、CreateAlternateViewFromString()メソッドを使うことで、SMTP Command Injection によって、任意のメールを送信することができてしまう脆弱性が確認できる。

図 2.4-1：送信するメール自体

「.(ピリオド)」だけの行で SMTP の「DATA」コマンドを終了させ、

それ以降は SMTP コマンドが続いている

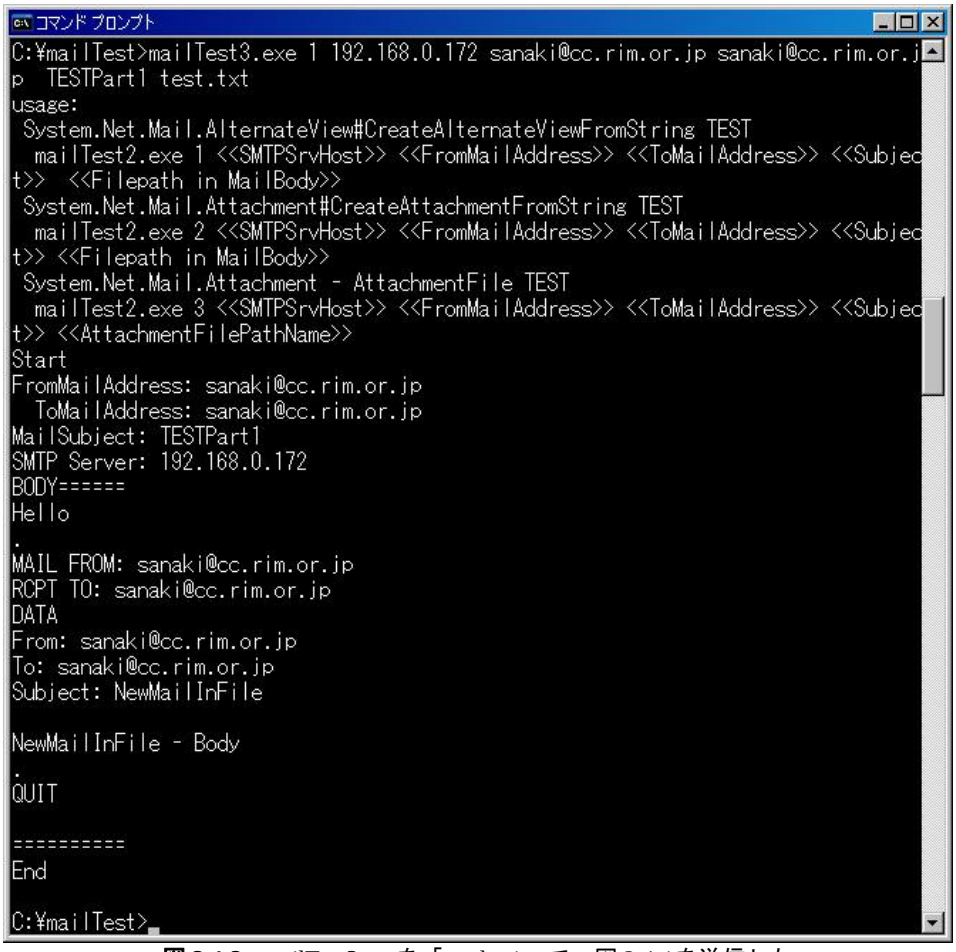


図 2.4-2 : mailTest3.cscを「mode=1」で、図 2.4-1を送信した

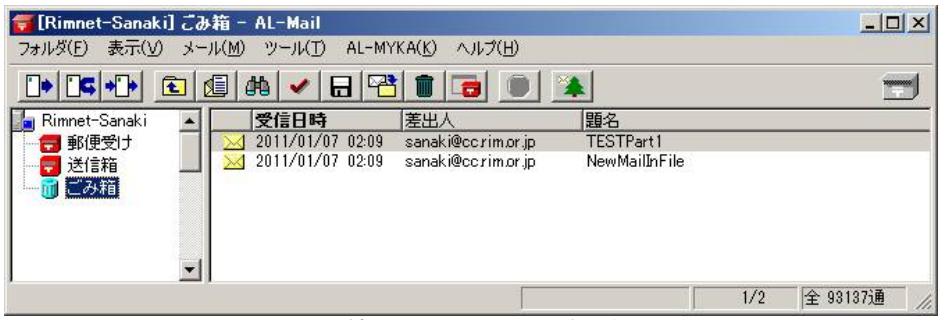


図 2.4-3 : 図 2.4-2の結果。二通のメールを受信したことから、

SMTP Command Injectionが成功してしまったことが分かる

2.5. Attachment#CreateAttachmentFromString() の場合

次は、Attachment#CreateAttachmentFromString()の場合である。
図 2.5-1と図 2.5-2の結果から、CreateAttachmentFromString()メソッドを使うことでも、SMTP Command Injection によって、任意のメールを送信することができてしまう脆弱性が確認できる。

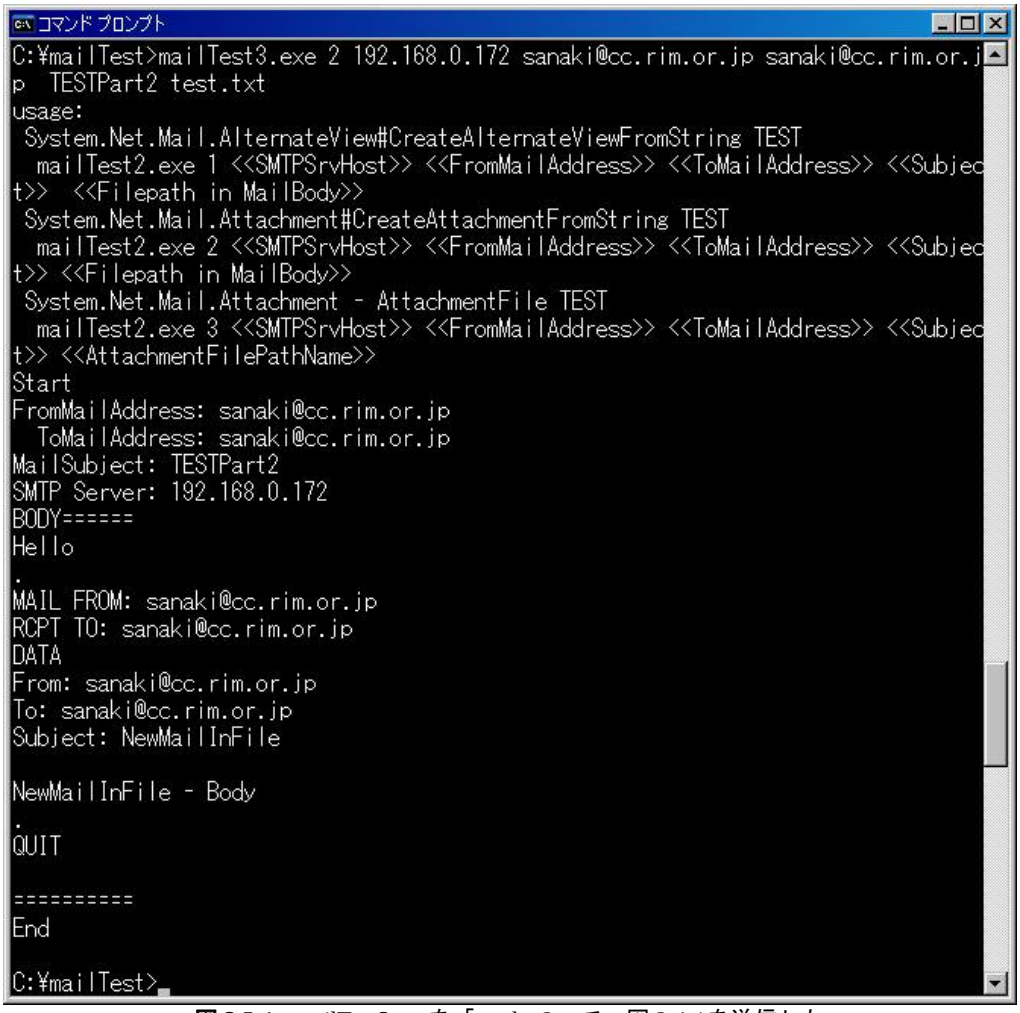


図 2.5-1 : mailTest3.cscを「mode=2」で、図 2.4-1を送信した

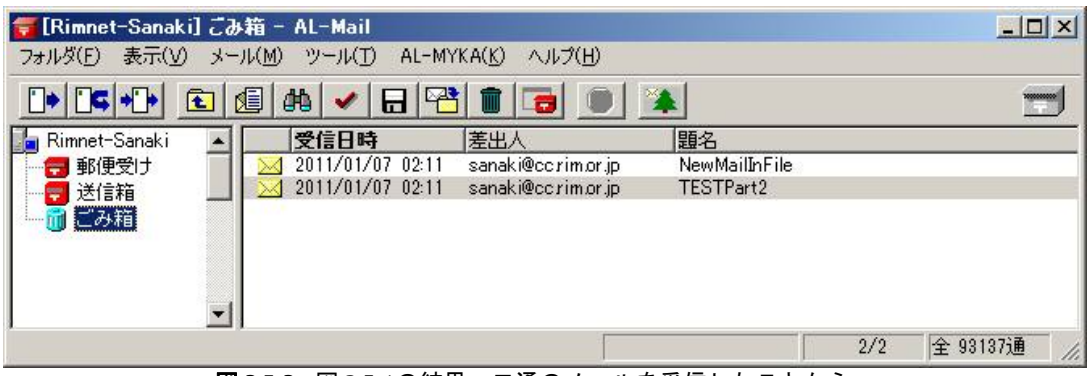


図 2.5-2 : 図 2.5-1の結果。二通のメールを受信したことから、

SMTP Command Injectionが成功してしまったことが分かる

2.6. Attachmentクラスを使ったファイル添付の場合(特殊な場面)

ファイルを添付する際に使われるクラスが、Attachmentクラスである(第6章の6)。このクラスを使うことで、添付ファイルはBase64 などの方法で符号化され、MIMEコンテンツの一つとしてMailMessageに割り当てられる。
一般的には、任意のファイルをそのまま Base64 などの方法でエンコードしてしまうだろうが、ここでは、少し特殊な場面を想定してみる。

例えば、ファイル拡張子を評価し.html ファイルであれば、エンコードしないで MIME コンテンツとして割り当てる、というようにプログラムされた場合である。
(Base64 などあらゆるエンコード処理では、サイズが大きくなるため、それをプログラマが嫌がり、エンコードする必要のない場合(全て 7bitASCII で収まると判断された場合など)、エンコードしないという仕様を想定してみた)
実際に、そのような場面でも使えるように、Attachment クラスには、TransferEncoding や ContentDisposition などのプロパティが設定できるようになっている。

図 2.6-4と図 2.6-5の結果から、添付ファイルを符号化しない場合でも、SMTP Command Injection によって、任意のメールを送信することができてしまう脆弱性が確認された。

図 2.6-1 : 送信するメール自体

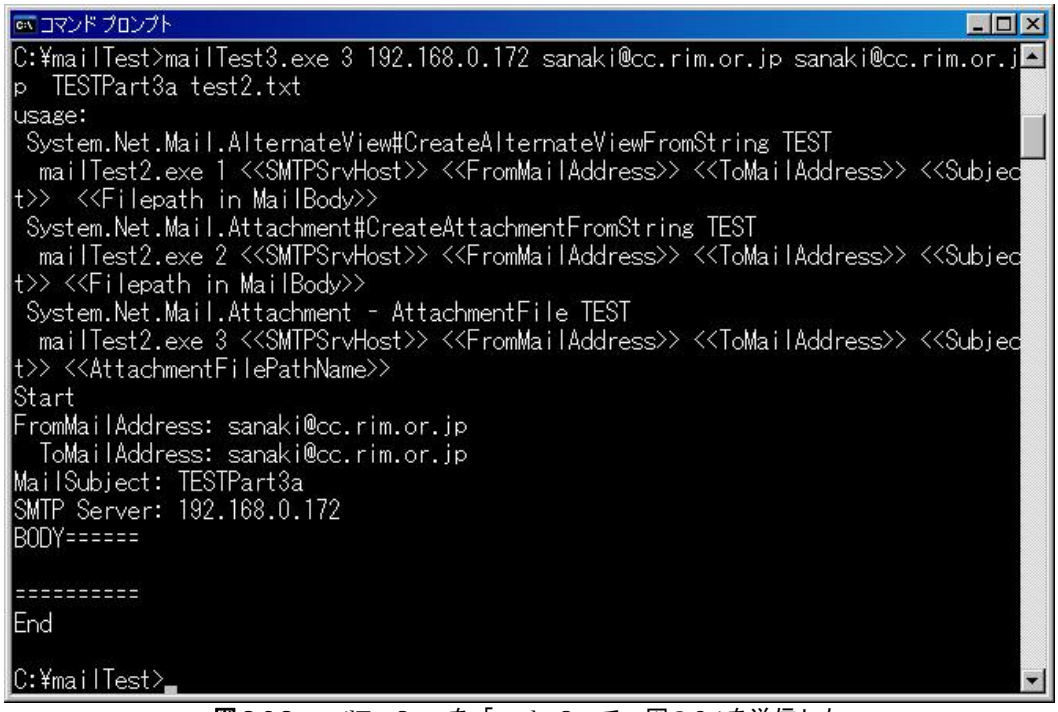


図 2.6-2 : mailTest3.cscを「mode=3」で、図 2.6-1を送信した

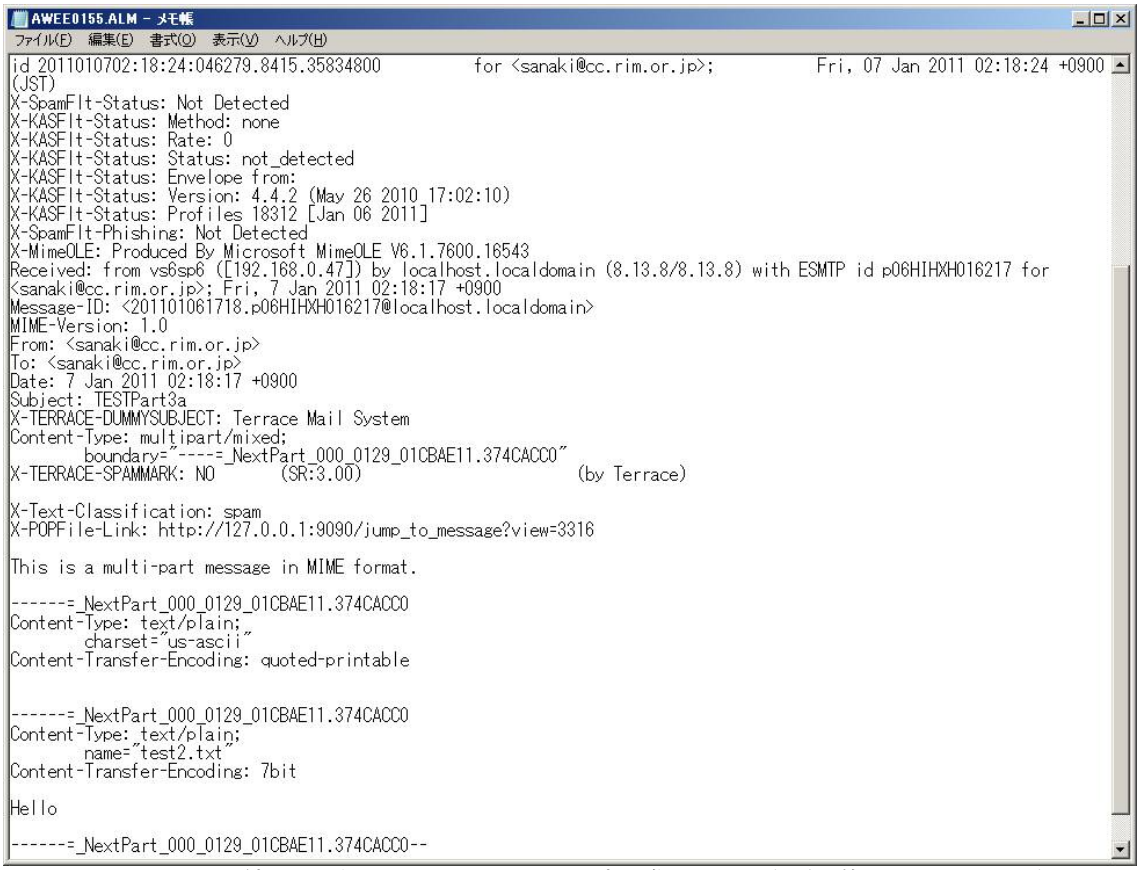


図 2.6-3 : 図 2.6-2の結果、受信したメール。MIMEにする際にBase64 などの符号化は必須ではない。

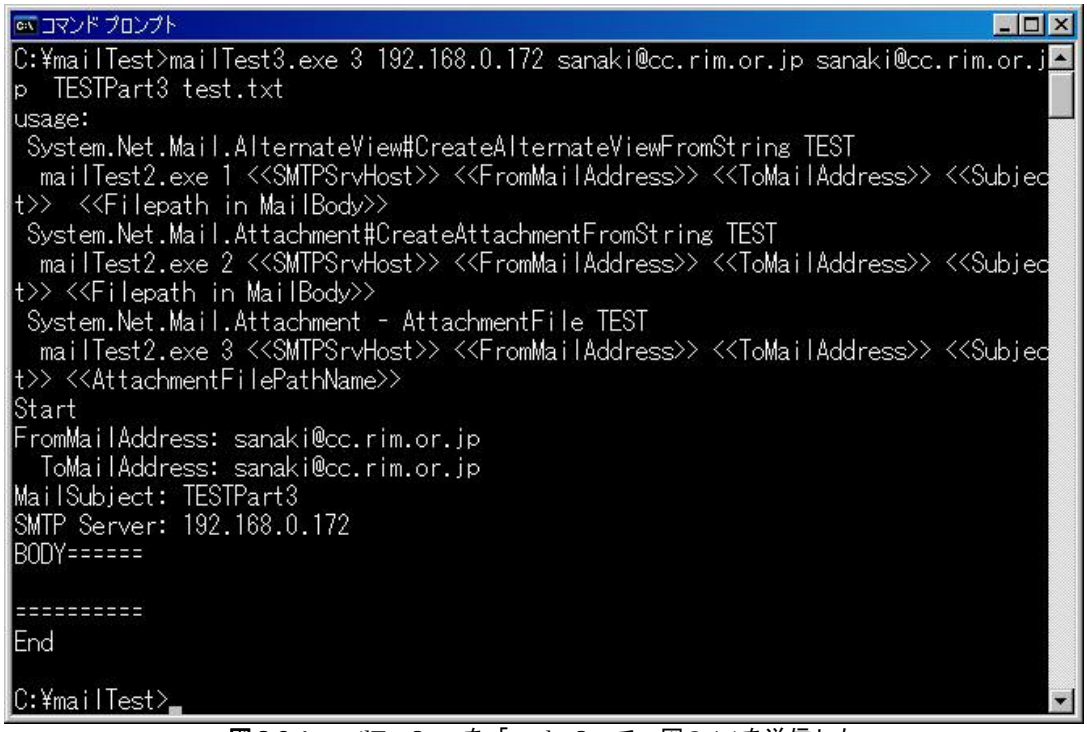


図 2.6-4 : mailTest3.cscを「mode=3」で、図 2.4-1を送信した

図 2.6-5 : 図 2.6-4の結果。二通のメールを受信したことから、

SMTP Command Injectionが成功してしまったことが分かる

3. 考察と対策(まとめにかえて)

図 2.3-1の検証プログラムのソースコードを見る限り、AlternateViewクラスよりは、MailMessageクラスのAlternateViewsコレクション及びAttachmentsコレクションのAdd()メソッドにおいて、hidden dot algorithm(第6章の11と12)が実施されていないことが原因であると推定される。

Microsoft 社は「対策は次期バージョンで実施する」との回答であった。つまり、現状の.NET Framework の全バージョンには、修正プログラムの提供は行われないということである。

本文書の執筆者の個人的な感覚としては、これは修正プログラムが必要な脆弱性であるとは思うのであるが、Microsoft 社としては、修正プログラムを提供する予定はないそうである。
この事は、アプリケーション・プログラマが、自らhidden dot algorithm(第6章の11と12)というエスケープ処理を行わなければならないということを意味しており、セキュリティ以前のライブラリの仕様として欠陥ではないだろうか。

.NET Framework 上で電子メールの送信機能を開発しているアプリケーション・プログラマが独自にエスケープ処理を実施する必要がある。
つまり、本文(エンコードしない添付ファイルも含)として与えるデータに対して

行頭の「≪.(ピリオド)≫」を「≪..(ピリオド二個)≫」

に置換する、という処理をアプリケーション・プログラマは自分自身で行うことが対策となる。
(ちなみに、変換対象の行は、「.(ピリオド)」だけ行ではなく、行頭が「.(ピリオド)」で始まる行である)

4. 付記 : 時系列

時系列は以下である。

1. 2007 年 03 月 26 日、BASP21 というレガシASP でよく使われるサードパーティのCOM コンポーネントにSMTP Command Injectionのバグが確認され、修正される
   (JVN #86092776)(第6章の 13)
2. 2010 年 09 月上旬、検証作業者の顧客(ASP.NET での Web メール機能アリ)に対して、セキュリティ診断を実施。この脆弱性を確認(顧客は自前のエスケープ処理を埋め込み対策済)。この脆弱性が.NET Framework のライブラリにあることを確認するのに時間がかかる
3. 2010 年 09 月 24 日、検証作業者は、日本の Microsoft 社へ報告
4. 2010 年 10 月 08 日以降、connect.microsoft.com で当該バグについてディスカッション
   (第6章の10)
   (検証作業者は気づかず)
5. 2010 年 11 月 26 日、Microsoft 社より、「次期バージョンの.NET Framework 4.5 で修正する」と回答を得る
6. 2010 年 12 月 21 日、検証作業者は、バグ情報(第6章の10)として公開されていることを認知
7. 2011 年 01 月 11 日、.NET Framework のアプリケーション・プログラマへ注意喚起として、Microsoft 社から許諾を得て、本文書を公開

5. 検証作業者

NTT コミュニケーションズ株式会社
IT マネジメントサービス事業部ネットワークマネジメントサービス部
セキュリティオペレーションセンター
　佐名木 智貴

6. 参考

1. セキュア Web プログラミング Tips 集(出版社:株式会社ソフト・リサーチ・センター)
   ISBN=978-4883732562
2. Security of WebAppli&Mail
   http://rocketeer.dip.jp/secProg/MailSecurity001.pdf
3. Security of HTTPHeader
   http://rocketeer.dip.jp/secProg/HttpSecurity003.pdf
4. System.Net.Mail.AlternateView クラス
   http://msdn.microsoft.com/ja-jp/library/system.net.mail.alternateview(VS.80).aspx
5. System.Net.Mail.AlternateView#CreateAlternateViewFromString メソッド
   http://msdn.microsoft.com/ja-jp/library/system.net.mail.alternateview.createalternateviewfromstring(VS.80).aspx
6. System.Net.Mail.Attachment クラス
   http://msdn.microsoft.com/ja-jp/library/system.net.mail.attachment(VS.80).aspx
7. System.Net.Mail.Attachment#CreateAttachmentFromString メソッド
   http://msdn.microsoft.com/ja-jp/library/system.net.mail.attachment.createattachmentfromstring(VS.80).aspx
8. System.Net.Mail.MailMessage クラス
   http://msdn.microsoft.com/ja-jp/library/system.net.mail.mailmessage(VS.80).aspx
9. System.Net.Mail.SmtpClient クラス
   http://msdn.microsoft.com/ja-jp/library/system.net.mail.smtpclient(VS.80).aspx
10. AlternateView.CreateAlternateViewFromString で CrLf インジェクションの脆弱性
    http://connect.microsoft.com/VisualStudioJapan/feedback/details/611673/alternateview-createalternateviewfromstring-crlf
11. RFC821 (4.5.2 に hidden dot algorithm の説明がある)
    http://www.sea-bird.org/doc/rfc_doc/rfc821-jp.txt
12. Sendmail VOLUME1 運用編 (出版社:オライリージャパン)
    ISBN=4-87311-176-5
    (P29 の脚注に「hidden dot algorithm」の説明がある)
13. JVN#86092776 BASP21 においてメールの不正送信が可能な脆弱性
    http://jvn.jp//jp/JVN86092776/index.html
14. BASP21
    http://www.hi-ho.ne.jp/babaq/basp21.html
15. .NET TIPS「.NET Framework 2.0 で電子メールを送信するには？[2.0 のみ、C#、VB]」
    http://www.atmarkit.co.jp/fdotnet/dotnettips/457sendmail2/sendmail2.html
16. .NET TIPS「JIS コード(JIS-2022-JP)でメールを送信するには？[2.0 のみ、C#、VB]」
    http://www.atmarkit.co.jp/fdotnet/dotnettips/696jismail/jismail.html

7. 履歴

- 2011 年 01 月 11 日：ver1.0 最初の公開

8. 最新版の公開URL

http://www.ntt.com/icto/security/data/soc.html#security_report

9. 本レポートに関する問合せ先

NTT コミュニケーションズ株式会社
IT マネジメントサービス事業部ネットワークマネジメントサービス部
セキュリティオペレーションセンター

e-mail: scan@ntt.com

以 上