# ZIP 作成時の UNICODE によって分離された文字を使ったサニタイズ回避法について

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

2010 年 04 月 28 日

Ver. 1.0

*Copyright 2010 NTT Communications Corporation*

# 1. 調査概要

Web アプリケーションなどで ZIP ファイルを生成する際、圧縮対象のファイル名を適切にサニタイズしないと、不用意に「..\」などの文字列が混入し、脆弱性を有する古い展開ツールを使っている利用者がそのファイルを展開する際に、予期せぬディレクトリ上にファイルが展開される可能性があることを検証した。
システム開発時に圧縮ライブラリを使用する際、ファイル名の文字コードについても注意した上で、システム開発を行う必要がある。

# 2. Webアプリケーションと ZIP作成機能

## 2.1. Webアプリケーションと ZIP作成機能

Web アプリケーションによって、ファイルをアップロードする機能を有する場合、稀にアップロードしたファイルを ZIP 圧縮した状態で、ダウンロード可能となる Web サイトがある。
ZIP 圧縮することで、通信量の削減や、Microsoft 社の Web ブラウザ Internet Explorer のファイル内容で判断する機能による XSS 誘発を防ぐなどを目的としていると思われる。

また、Windows上ではZIP形式で圧縮されたファイルのファイル名には、UNICODEではなくANSIコードを使用しているため、UNICODEによって分離された文字(円記号[u00A5])を使ったサニタイズ回避テクニックを用いられることで、ZIP圧縮する際のサニタイズ処理が回避され不正なファイル名が忍び込む危険性がある(図 2.1-2～図 2.1-4)。

一部の展開ツールでは、セキュリティ侵害行為とならないように、予期しない場所へのファイルの展開はできないようになっている(図 2.1-5～図 2.1-8)。

逆に一部の展開ツールでは、圧縮されたファイル名の指示通りに展開してしまうツールも未だに存在する(図 2.1-9～図 2.1-10)。

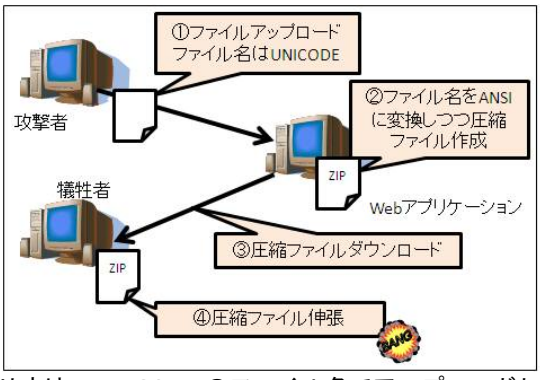


**図 2.1-1 : 攻撃シナリオは、UNICODE のファイル名でアップロードした圧縮ファイルが、犠牲者のホスト上で伸張される際に、ディレクトリトラバーサル攻撃が成立する可能性がある**

*Copyright 2010 NTT Communications Corporation*


**図 2.1-2 :**「文字コード表」を使って、UNICODE の円記号をファイル名に使う

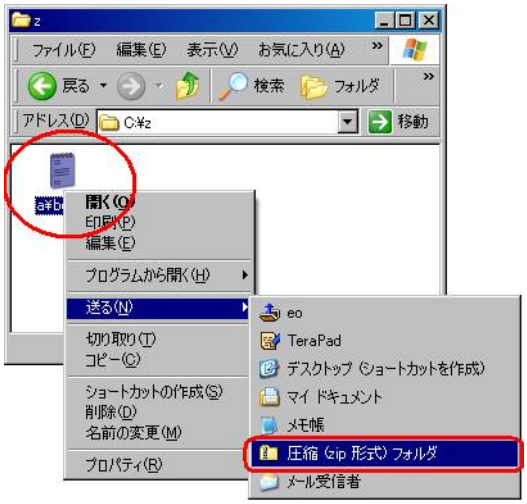


**図 2.1-3 :** ファイル名に UNICODE で分離された円記号を含むファイルを MS-WindowsXP SP3 標準の「ZIP フォルダ」で圧縮してみる

*Copyright 2010 NTT Communications Corporation*

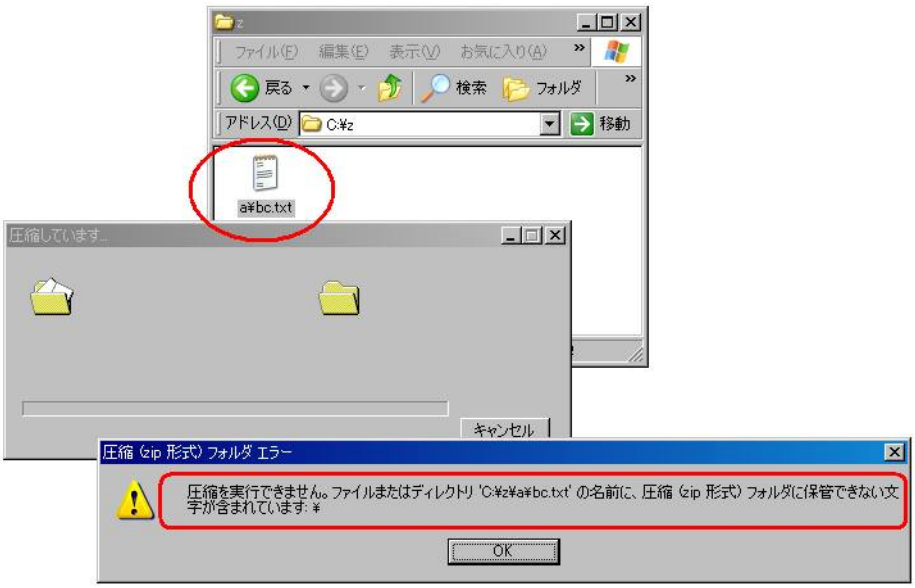


**図 2.1-4 :** 図 2.1-3の結果。UNICODEによって分離された円記号を
ファイル名に含むファイルは圧縮できないようだ

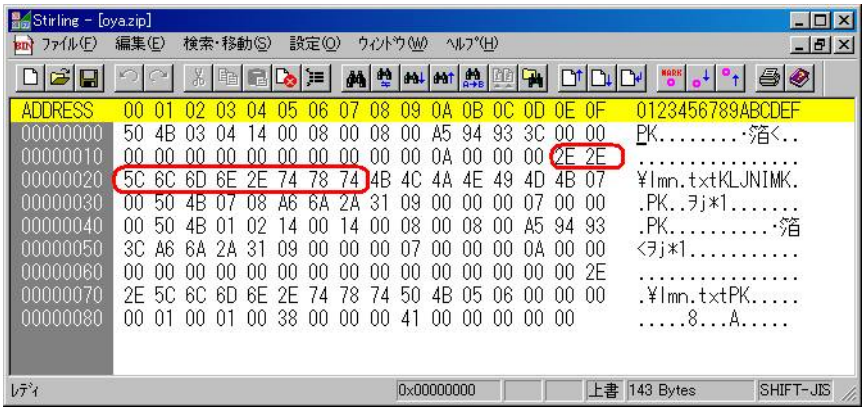


**図 2.1-5 :** 「..」を含む相対パスのファイル名[..\lmn.txt]が圧縮されている ZIP ファイル


**図 2.1-6 :** 図 2.1-5をデスクトップ上に展開しようとするとeo1.5.2 は
「..」を無視し、そのままデスクトップ上に展開した

*Copyright 2010 NTT Communications Corporation*

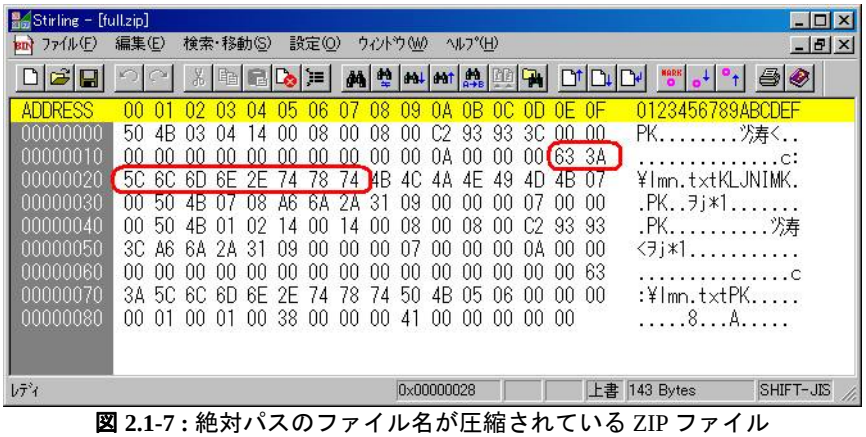


**図 2.1-7 :** 絶対パスのファイル名が圧縮されている ZIP ファイル

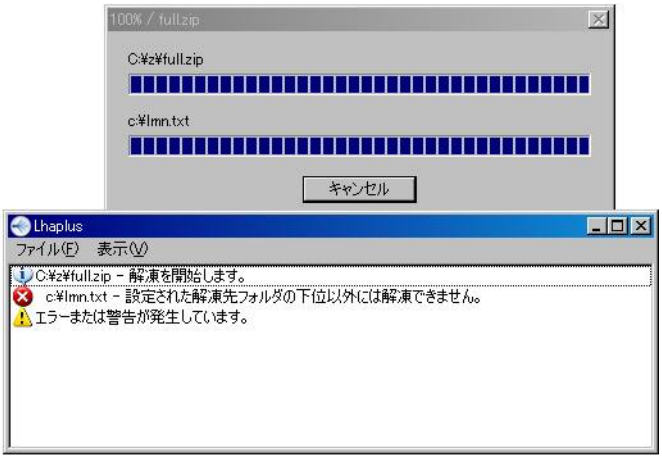


**図 2.1-8 :** 図 2.1-7をデスクトップ上に展開しようとするとLhaplus1.57 は
このようなエラーが表示されて、展開に失敗する


**図 2.1-9 :** 相対パス形式となっている図 2.1-5を今度は、Lhaplus1.57 で、
ZIPファイルのあるディレクトリ上(c:\x\y)に展開してみる

*Copyright 2010 NTT Communications Corporation*


**図 2.1-10 :** 図 2.1-9の結果。Lhaplusは相対パス形式については、
指示通り([c:\x\y]の一つ上のディレクトリ[..\lmn.txt]なので、[c:\x])に展開するようだ

# 3. Javaの場合

Java の場合、JDK 標準の java.util.zip を使うことで、ZIP アーカイバを扱うことができる。
しかしながら、圧縮ファイル名は UTF-8 に変換されてしまうため、Windows 上で展開する場合、日本語文字を含むファイル名が文字化けしてしまう。
Java には、JDK 標準以外にも、Ant パッケージの org.apache.tools.zip を使うことでも ZIP アーカイバを扱うことができる。こちらは、文字コードを指定できるなど、JDK 標準より柔軟である。
この Ant を使用して ANSI 文字コードのファイル名として圧縮処理を行う場合、UNICODE によって分離された文字を使ったサニタイズ回避テクニックが有効なため、ファイル名を一度 ANSI コードへ変換し、その上で UNICODE に変換しなおした上(Java コード上の文字コードは UNICODE であるため)でサニタイズ処理を行うことが求められる。

## 3.1. java.util.zipの場合

検証環境

- MS-WindowsXP SP3
- JDK 1.6.0_06

*Copyright 2010 NTT Communications Corporation*

```
import java.io.File;
import java.io.FileOutputStream;
import java.util.zip.ZipOutputStream;
import java.util.zip.ZipEntry;
// import org.apache.tools.zip.ZipOutputStream;
// import org.apache.tools.zip.ZipEntry;

public class ZipTest{
 public static void main(String[] argv){
  int i;
  System.out.println("usage : ZipTest ZIPFileName FileNameString ArchivedData Flg");
  System.out.println(" Flg = 1 -> Replace[u005c]->[u00a5]");
  if(2 < argv.length){
   String myData = argv[2];
   String myFileNameStr = argv[1];
   byte[] myDataHako = myData.getBytes();

   if(3 < argv.length){
    System.out.println("Before String = " + myFileNameStr);
    char[] tempHako = myFileNameStr.toCharArray();
    for(i=0;i<tempHako.length;i++){
     System.out.print(tempHako[i] + "(" + Integer.toHexString((int)tempHako[i]) + ") ");
     if(Character.valueOf(tempHako[i]).equals('¥¥') == true){
      char[] t = Character.toChars(165);
      tempHako[i] = t[0];
     }
    }
    myFileNameStr = "";
    for(i=0;i<tempHako.length;i++){
     myFileNameStr += Character.toString(tempHako[i]);
    }
    System.out.println("¥nAfter String = " + myFileNameStr);
    tempHako = myFileNameStr.toCharArray();
    for(i=0;i<tempHako.length;i++){
     System.out.print(tempHako[i] + "(" + Integer.toHexString((int)tempHako[i]) + ") ");
    }
    System.out.println("");
   }
   try{
    File zipFile = new File(argv[0]);
    ZipOutputStream myZipOutputStream = new ZipOutputStream(new FileOutputStream(zipFile));
//    myZipOutputStream.setEncoding("MS932");
    myZipOutputStream.putNextEntry(new ZipEntry(myFileNameStr));
    myZipOutputStream.write(myDataHako);
    myZipOutputStream.closeEntry();
    myZipOutputStream.close();
   }catch(Exception e){
    e.printStackTrace();
   }
  }
  System.out.println("End");
 }  }
```

**図 3.1-1：検証コード(java.util.zip[JDK 標準]の場合)**

*Copyright 2010 NTT Communications Corporation*

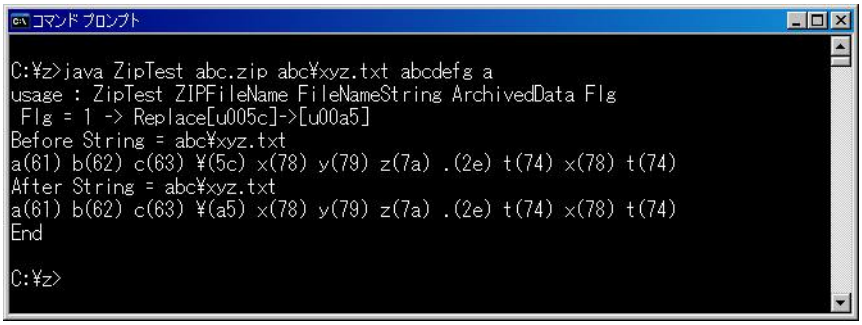


**図 3.1-2 :** 図 3.1-1の実行結果

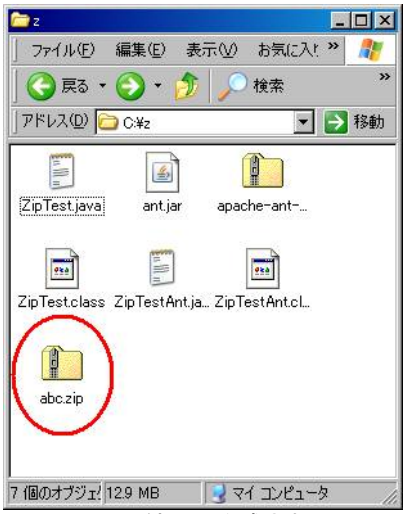


**図 3.1-3 :** 図 3.1-2の結果、作成されたZIPファイル


**図 3.1-4 :** 図 3.1-3の中身をMS-WindowsXP標準の「ZIPフォルダ」で中身を確認すると
ファイル名文字化けしている

*Copyright 2010 NTT Communications Corporation*

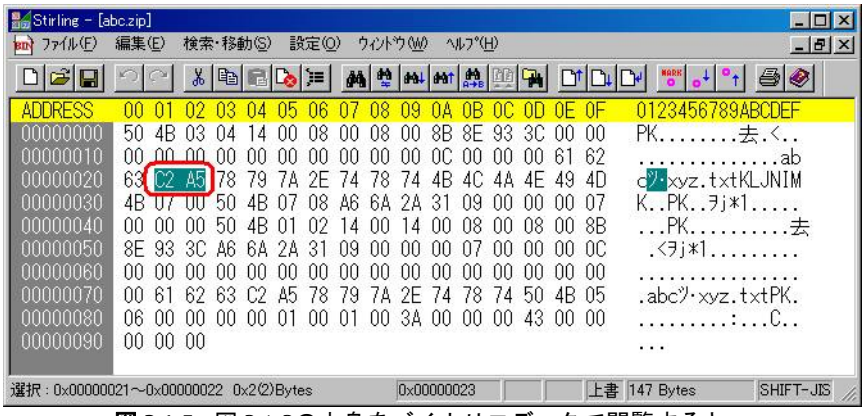


**図 3.1-5 :** 図 3.1-3の中身をバイナリエディタで閲覧すると、

「円記号」が「c2 a5」(UTF-8)となっている

## 3.2. org.apache.tools.zip(Antパッケージ)の場合

検証環境

- MS-WindowsXP SP3
- JDK 1.6.0_06
- Ant 1.8.0

*Copyright 2010 NTT Communications Corporation*

```
import java.io.File;
import java.io.FileOutputStream;
// import java.util.zip.ZipOutputStream;
// import java.util.zip.ZipEntry;
import org.apache.tools.zip.ZipOutputStream;
import org.apache.tools.zip.ZipEntry;

public class ZipTest{
 public static void main(String[] argv){
  int i;
  System.out.println("usage : ZipTest ZIPFileName FileNameString ArchivedData Flg");
  System.out.println(" Flg = 1 -> Replace[u005c]->[u00a5]");
  if(2 < argv.length){
   String myData = argv[2];
   String myFileNameStr = argv[1];
   byte[] myDataHako = myData.getBytes();

   if(3 < argv.length){
    System.out.println("Before String = " + myFileNameStr);
    char[] tempHako = myFileNameStr.toCharArray();
    for(i=0;i<tempHako.length;i++){
     System.out.print(tempHako[i] + "(" + Integer.toHexString((int)tempHako[i]) + ") ");
     if(Character.valueOf(tempHako[i]).equals('¥¥') == true){
      char[] t = Character.toChars(165);
      tempHako[i] = t[0];
     }
    }
    myFileNameStr = "";
    for(i=0;i<tempHako.length;i++){
     myFileNameStr += Character.toString(tempHako[i]);
    }
    System.out.println("¥nAfter String = " + myFileNameStr);
    tempHako = myFileNameStr.toCharArray();
    for(i=0;i<tempHako.length;i++){
     System.out.print(tempHako[i] + "(" + Integer.toHexString((int)tempHako[i]) + ") ");
    }
    System.out.println("");
   }
   try{
    File zipFile = new File(argv[0]);
    ZipOutputStream myZipOutputStream = new ZipOutputStream(new FileOutputStream(zipFile));
    myZipOutputStream.setEncoding("MS932");
    myZipOutputStream.putNextEntry(new ZipEntry(myFileNameStr));
    myZipOutputStream.write(myDataHako);
    myZipOutputStream.closeEntry();
    myZipOutputStream.close();
   }catch(Exception e){
    e.printStackTrace();
   }
  }
  System.out.println("End");
 }  }
```

**図 3.2-1：**検証コード(org.apache.tools.zip[Ant]の場合)

文字コードを指定するメソッド以外、図 3.1-1とほとんど同じである

*Copyright 2010 NTT Communications Corporation*

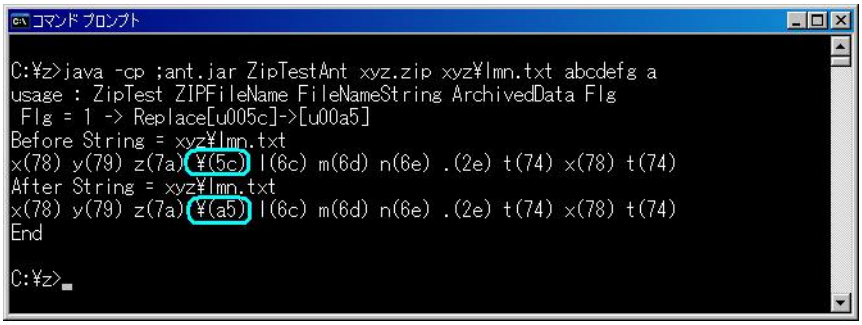


**図 3.2-2 :** 図 3.2-1の実行結果

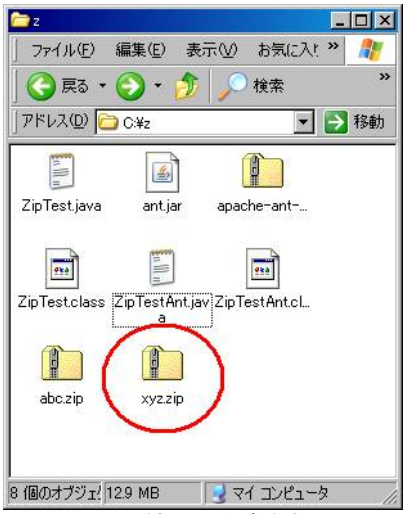


**図 3.2-3 :** 図 3.2-2の結果、作成されたZIPファイル


**図 3.2-4 :** 図 3.2-3の中身をMS-WindowsXP標準の「ZIPフォルダ」で中身を確認すると
サブフォルダが存在している

*Copyright 2010 NTT Communications Corporation*

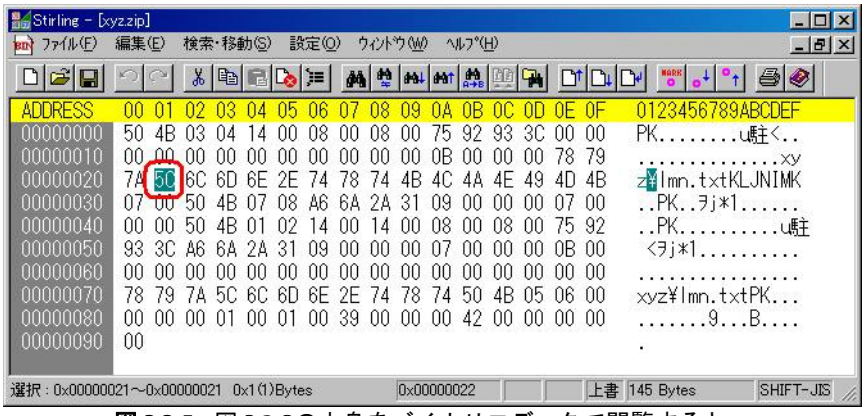


**図 3.2-5 :** 図 3.2-3の中身をバイナリエディタで閲覧すると、
「円記号」が「5c」(つまり 0x5cに縮退している)となっている

# 4. PHPの場合

PHP で、ZIP 圧縮を行う場合、PHP 標準の zip 圧縮ライブラリ(Win32 版：php_zip.dll)と、phpMyAdmin がインストールされた環境であれば、phpMyAdmin のライブラリ(zip.lib.php)を使う場合と、二通りの方法がある。

これらを使って、ファイル・アップロード機能があり、アップロードされたファイルを ZIP 圧縮する UTF-8(UNICODE)の Web ページを作成し、挙動の確認を行った。

検証の結果、標準の zip 圧縮ライブラリ、phpMyAdmin のライブラリ共に、ファイル名の文字コードを自動変換する機能を有していないことを確認した。
よって、共に ZIP ファイル内部の圧縮されたファイルのファイル名は、与えられた文字コード UTF-8 のままであり、本文書が期待するサニタイズ回避テクニックは使用することができないことを確認した。
つまり、プログラマが明示的に文字コード変換処理を行う必要があり、その際に文字コード変換前にサニタイズ処理を行うようなコーディングをしていれば、本文書が期待するサニタイズ回避テクニックが有効に働くものと思われるが、プログラマのそのような行動は、稀であると思われる。

## 4.1. php_zip.dll (Win32) の場合

検証環境

- MS-Windows2000 SP4
- Apache 2.2.11 for Win32
- PHP 5.2.13 for Win32

*Copyright 2010 NTT Communications Corporation*

```
<html>
         <head>
          <meta http-equiv="content-type" content="text/html;charset=utf-8">
          <title>test1</title>
         </head>
         <body>
<?php
$display_data = "";
$zip_name = "test1.zip";
if(isset($zip_name)){

        if(is_uploaded_file($_FILES['upfile']['tmp_name'])){
                $filename = $_FILES['upfile']['name'];
                $file_content = $_FILES['upfile']['tmp_name'];
                $contents = file_get_contents($file_content);
                $zip = new ZipArchive();
                $result = $zip->open($zip_name , ZipArchive::CREATE);

                if($result === TRUE){
                        $zip->AddFromString($filename , $contents);
                        $zip->close();
                }
                $hex_content = bin2hex($filename);
                $count = strlen($hex_content) / 2;

                for($i=0;$i<$count;$i++){
                        if($i==0){
                                $t=0;
                        }else{
                                $t=$i * 2;
                        }
                        $data = substr($hex_content , $t , 2);
                        $display_data = $display_data . $data . " ";
                }
        }
}
?>
                <form enctype="multipart/form-data" method="post" action="">
                アップロード：<br>
                <input type="file" name="upfile"><br>
                <input type="submit">
                </form>
                <hr>
                <?php echo $filename; ?><br>
                <?php echo $display_data; ?>
        </body>
</html>
```

**図 4.1-1：検証コード**(org.apache.tools.zip[Ant]の場合)

*Copyright 2010 NTT Communications Corporation*


**図 4.1-2 :** ZIP ファイルの保存先フォルダ


**図 4.1-3 :** 図 4.1-1のWebページ、ここでtest.txtをアップロードする

*Copyright 2010 NTT Communications Corporation*

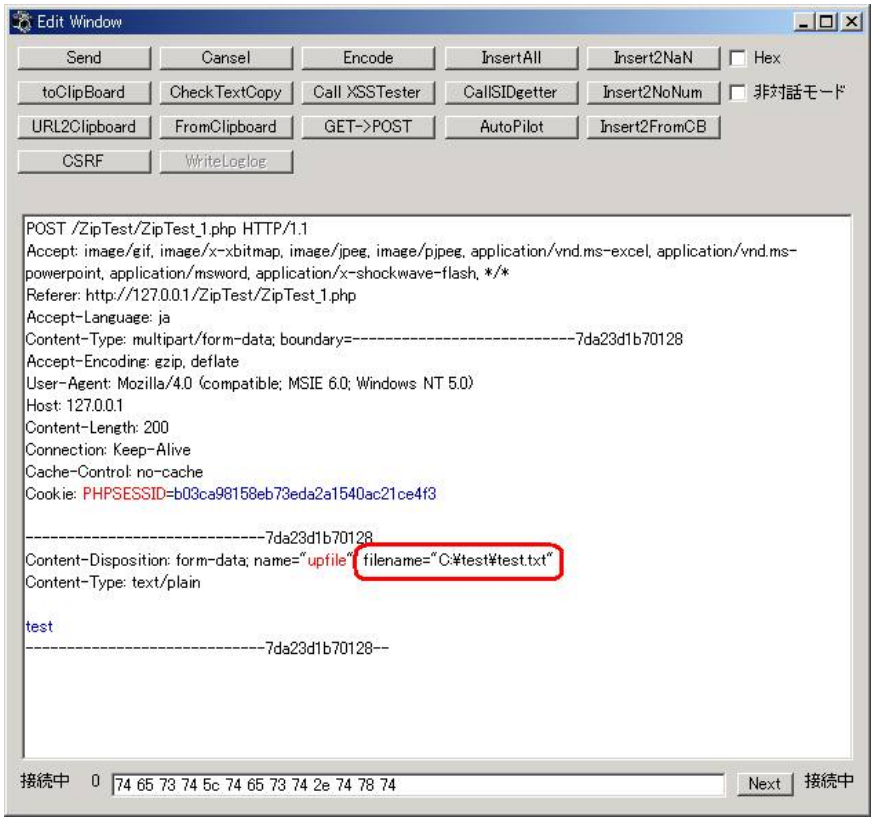


**図 4.1-4 :** 図 4.1-3のHTTPリクエスト

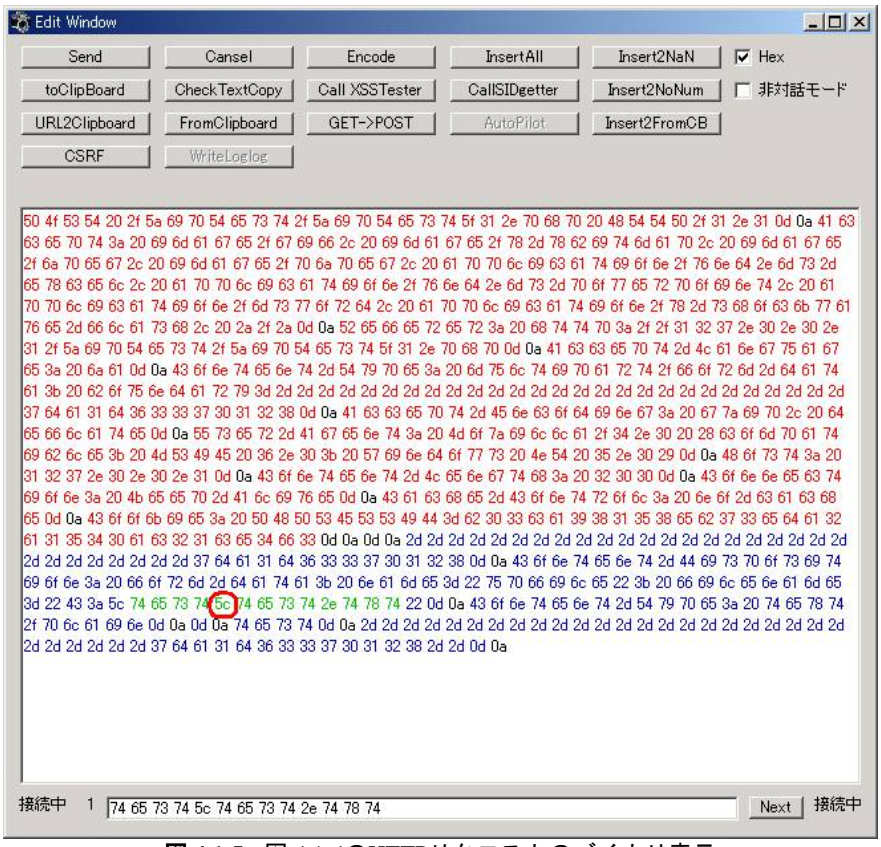


**図 4.1-5 :** 図 4.1-4のHTTPリクエストのバイナリ表示

*Copyright 2010 NTT Communications Corporation*

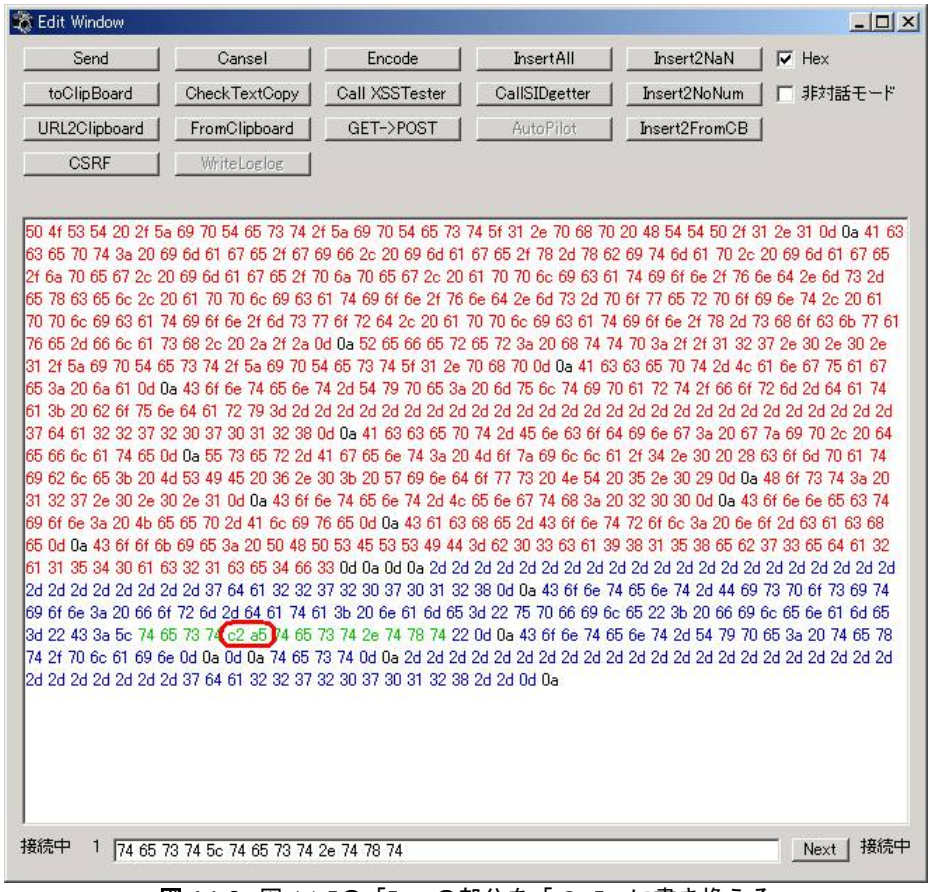


**図 4.1-6 :** 図 4.1-5の「5c」の部分を「c2 a5」に書き換える

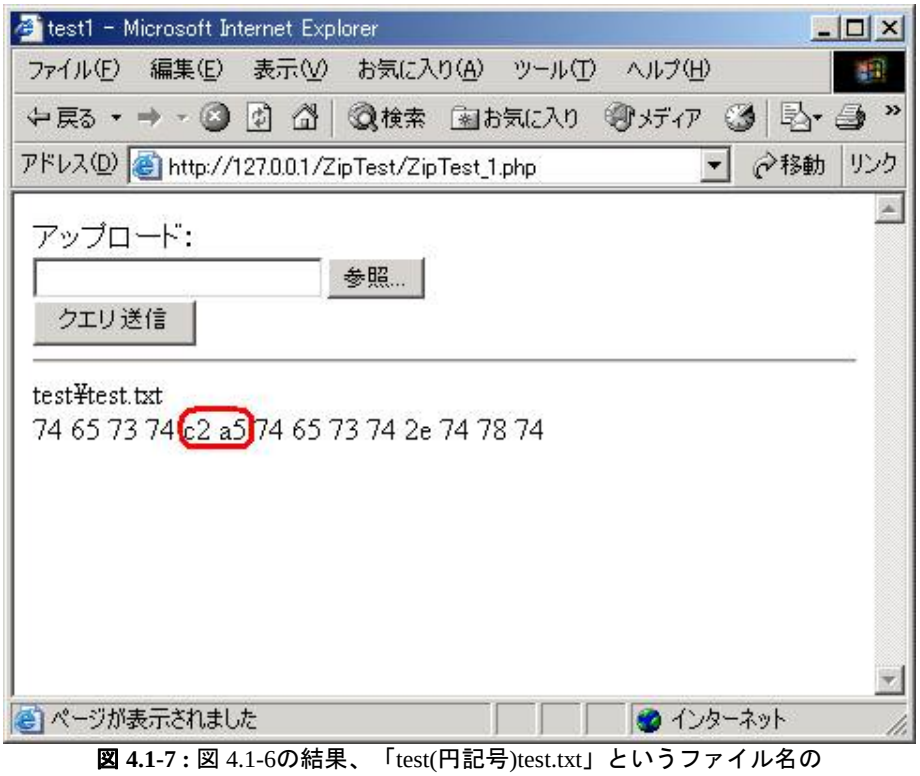


**図 4.1-7 :** 図 4.1-6の結果、「test(円記号)test.txt」というファイル名の
ファイルが圧縮されたことになっている

*Copyright 2010 NTT Communications Corporation*


**図 4.1-8 :** 図 4.1-7の後の図 4.1-2には「test1.zip」というファイルが生成された

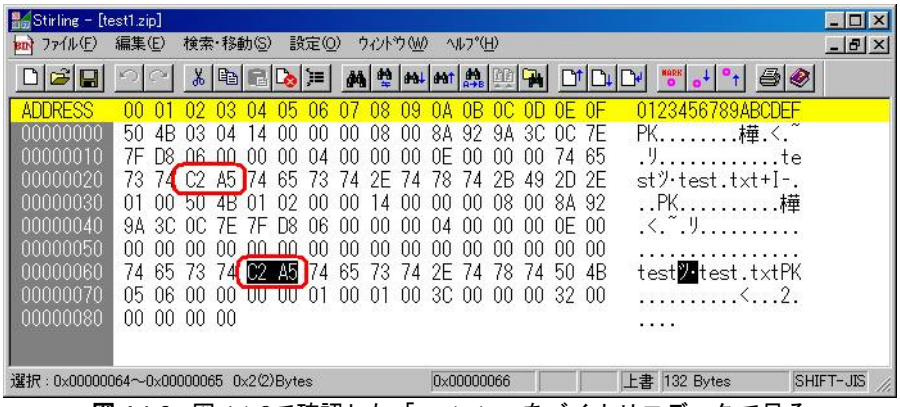


**図 4.1-9 :** 図 4.1-8で確認した「test1.zip」をバイナリエディタで見る

このように、zipファイル内の圧縮されたファイル名は、特に文字コードが変換されることなく、

そのままUTF-8 の文字コードで格納されている

## 4.2. zip.lib.php (phpMyAdmin) の場合

検証環境

- MS-Windows2000 SP4
- Apache 2.2.11 for Win32
- PHP 5.2.13 for Win32
- phpMyAdmin 3.3.2

*Copyright 2010 NTT Communications Corporation*

```
<html>
         <head>
          <meta http-equiv="content-type" content="text/html;charset=utf-8">
          <title>test2</title>
         </head>
         <body>
<?php
$display_data = "";
$zip_name = "test2.zip";
if(isset($zip_name)){

        if(is_uploaded_file($_FILES['upfile']['tmp_name'])){
                $filename = $_FILES['upfile']['name'];
                $file_content = $_FILES['upfile']['tmp_name'];
                require_once('zip.lib.php');
                $zipfile = new zipfile();
                $handle = fopen($file_content, "rb");
                $contents = fread($handle, filesize($file_content));
                fclose($handle);
                $zipfile -> addFile( $contents, $filename );
                $zip_buffer = $zipfile->file();
                $handle = fopen($zip_name, "wb");
                fwrite($handle, $zip_buffer );
                fclose($handle);

                $hex_content = bin2hex($filename);
                $count = strlen($hex_content) / 2;

                for($i=0;$i<$count;$i++){
                        if($i==0){
                                $t=0;
                        }else{
                                $t=$i * 2;
                        }
                        $data = substr($hex_content , $t , 2);
                        $display_data = $display_data . $data . " ";
                }
        }
}

?>
                <form enctype="multipart/form-data" method="post" action="">
                アップロード：<br>
                <input type="file" name="upfile"><br>
                <input type="submit">
                </form>
                <hr>
                <?php echo $filename; ?><br>
                <?php echo $display_data; ?>
        </body>
</html>
```

**図 4.2-1：検証コード**(org.apache.tools.zip[Ant]の場合)

*Copyright 2010 NTT Communications Corporation*

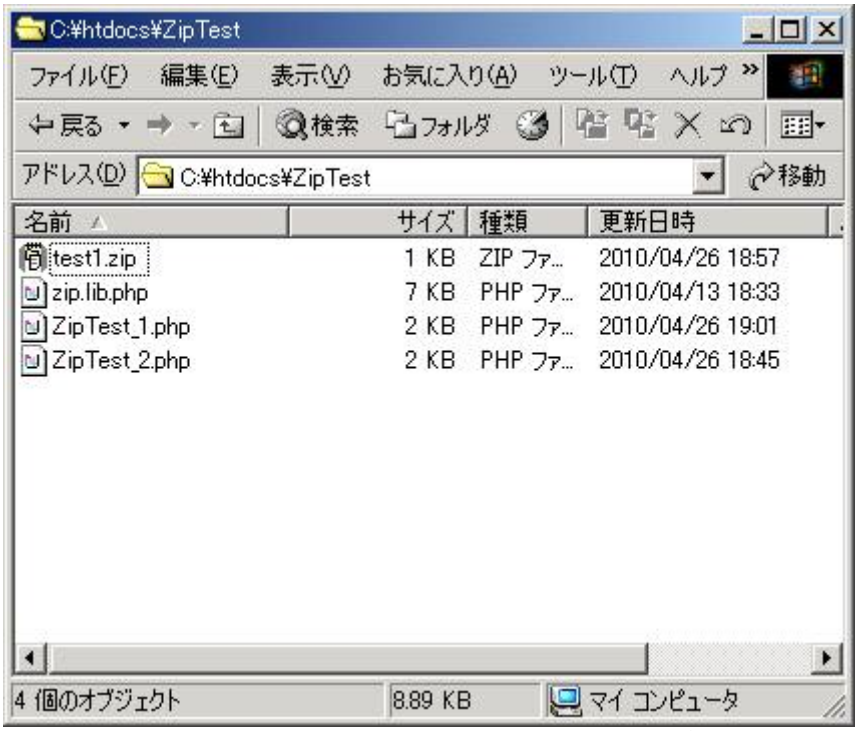


**図 4.2-2 :** 検証前の ZIP ファイルの保存先フォルダ

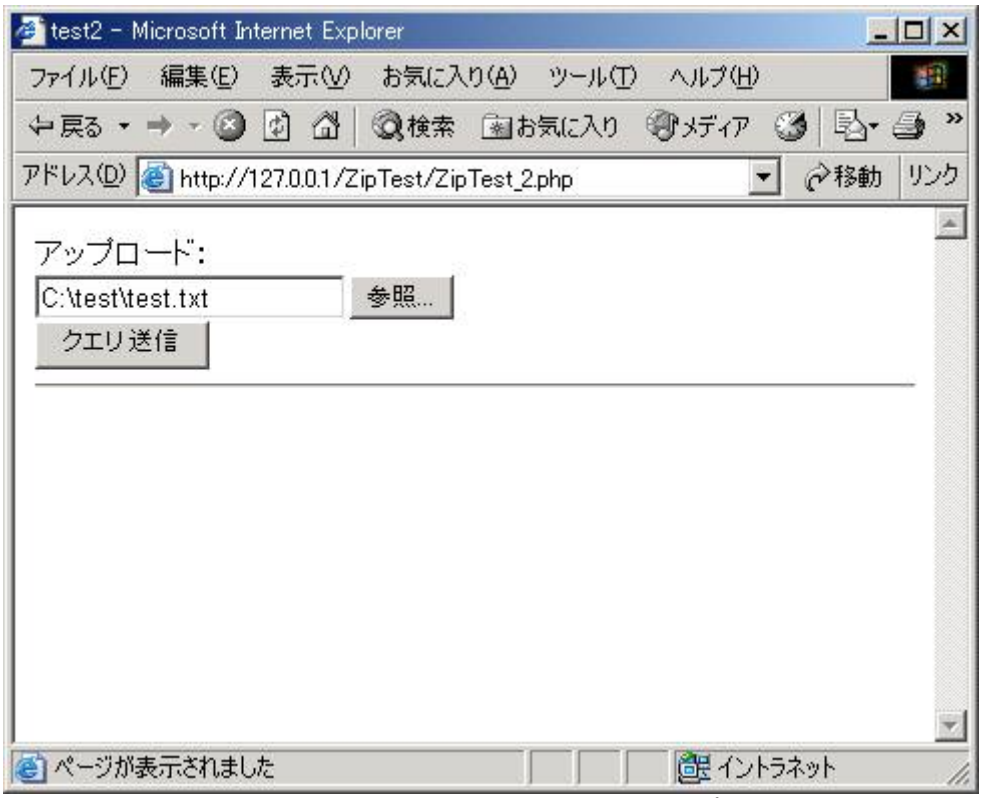


**図 4.2-3 :** 図 4.2-1のWebページ、ここでtest.txtをアップロードする

*Copyright 2010 NTT Communications Corporation*

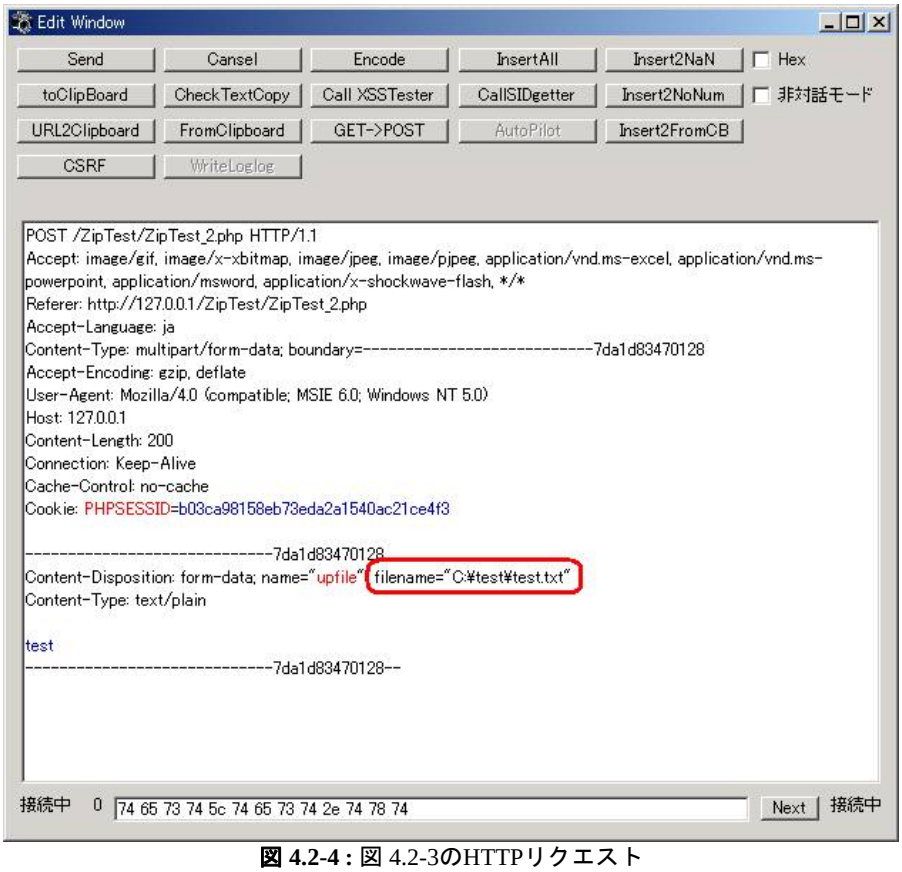


**図 4.2-4 :** 図 4.2-3のHTTPリクエスト

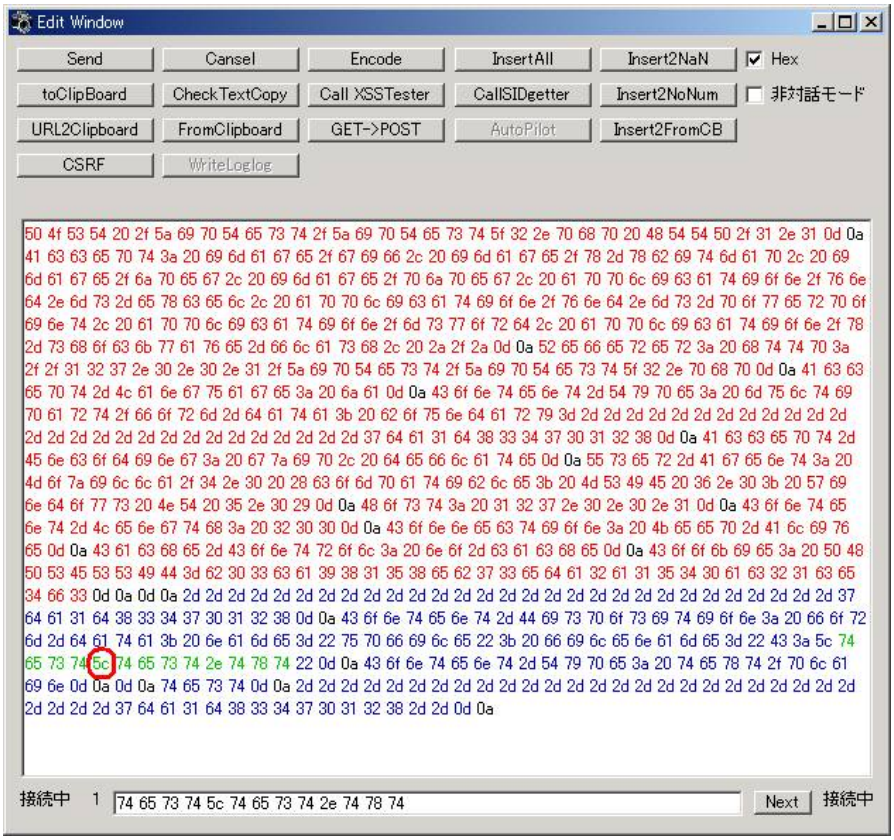


**図 4.2-5 :** 図 4.2-4のHTTPリクエストのバイナリ表示

*Copyright 2010 NTT Communications Corporation*

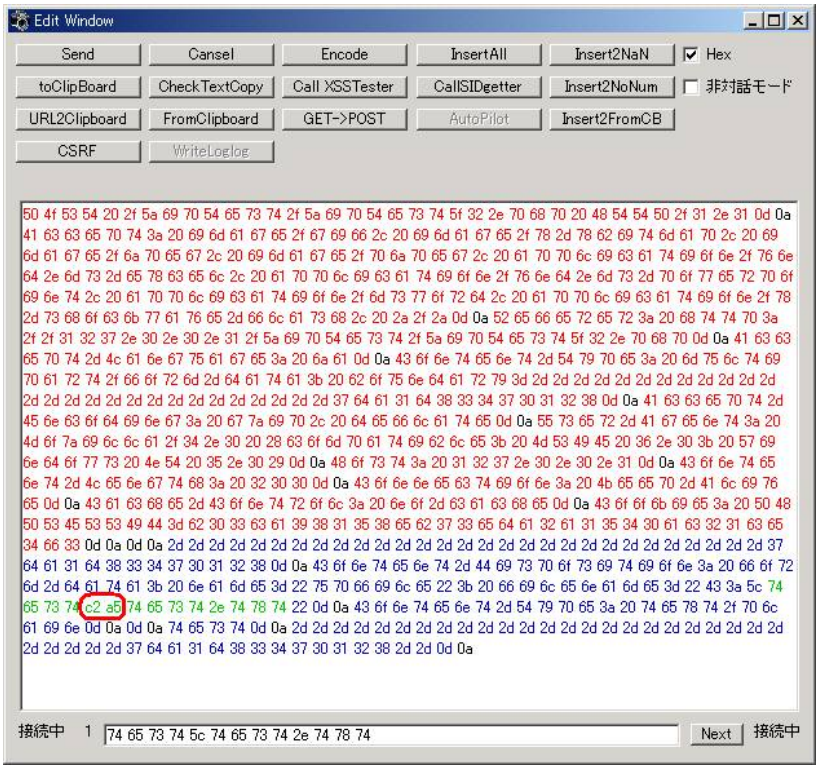


**図 4.2-6 :** 図 4.2-5の「5c」の部分を「c2 a5」に書き換える

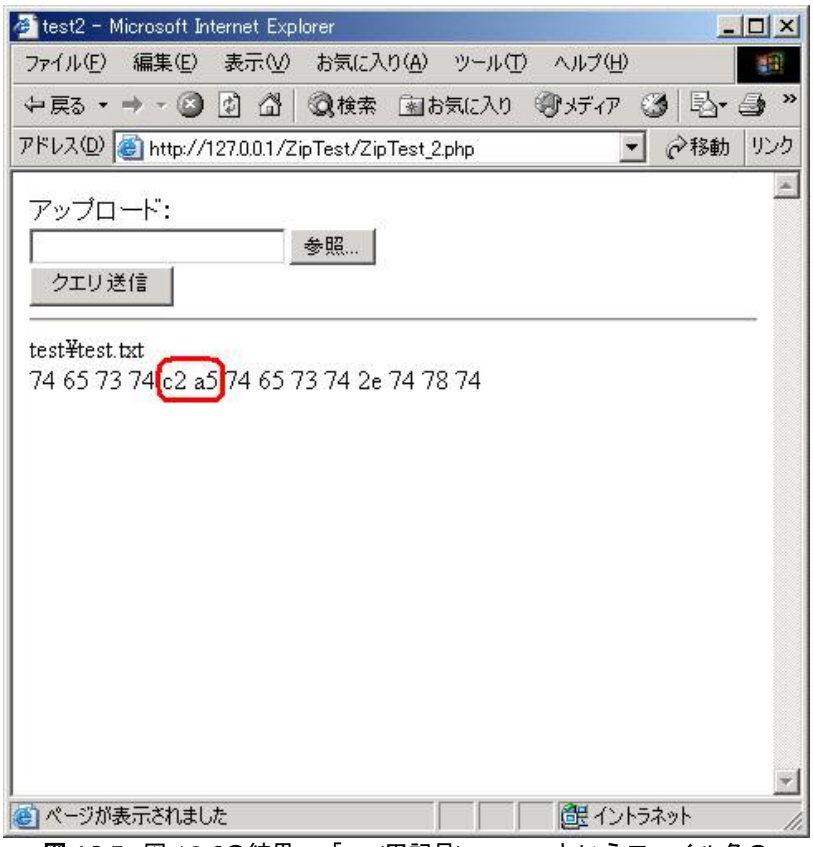


**図 4.2-7 :** 図 4.2-6の結果、「test(円記号)test.txt」というファイル名の
ファイルが圧縮されたことになっている

*Copyright 2010 NTT Communications Corporation*

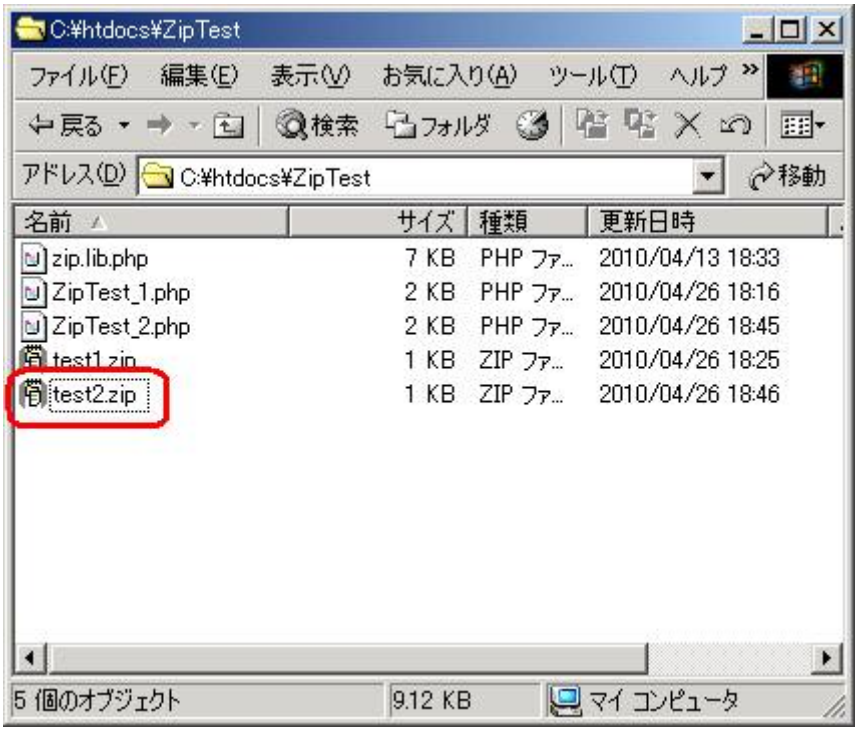


**図 4.2-8 :** 図 4.2-7の後の図 4.2-2には「test2.zip」というファイルが生成された

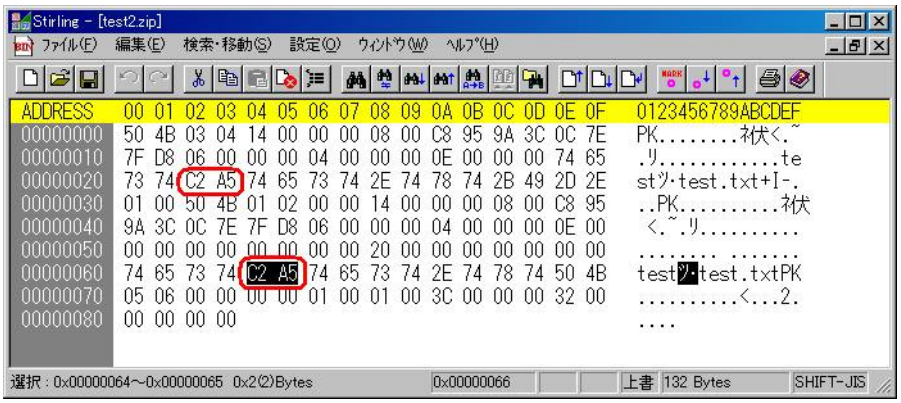


**図 4.2-9 :** 図 4.2-8で確認した「test2.zip」をバイナリエディタで見る

このように、zipファイル内の圧縮されたファイル名は、特に文字コードが変換されることなく、

そのままUTF-8 の文字コードで格納されている

*Copyright 2010 NTT Communications Corporation*

## 5. まとめ

現時点では、MS-Windows の ZIP フォルダなど MS-Windows 上で動作する展開ツールのほとんどが UNICODE に対応していない以上、ZIP ファイル内部に圧縮して保存するファイルのファイル名には、ANSI コードを選択せざるを得ないだろう。
システム設計がこのような場合、Webアプリケーション開発者を含めZIP圧縮を行うアプリケーションに携わっている開発者の方で、UNICODEのファイル名が汚染データ[1]である場合、一旦ファイル名をUNICODEからANSIコードへ変換した後で、サニタイズ処理[2]を実施しなければ、本文書で指摘したようなZIP展開時に想定していないディレクトリへZIP圧縮されたファイルが展開されるだろう。

## 6. 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
本城 敏信

## 7. 参考

1. UNICODE とセキュリティ
http://openmya.hacker.jp/hasegawa/public/20041030/unicode-and-security.pdf
2. UTF-8.jp
http://www.utf-8.jp/
3. Unicode とサニタイジング回避テクニック ver1.6
http://rocketeer.dip.jp/secProg/unicodebug007.pdf
4. 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
5. IPA ISEC セキュアプログラミング講座 ver1 8-1.Windows パス名の落とし穴
http://www.ipa.go.jp/security/awareness/vendor/programmingv1/b08_01.html
6. IPA ISEC セキュアプログラミング講座 ver1 7-7. Unix パス名の安全対策
http://www.ipa.go.jp/security/awareness/vendor/programmingv1/b07_07.html
7. IPA ISEC セキュアプログラミング講座 ver1 8-3. NTFS のセキュリティ機能と落とし穴
http://www.ipa.go.jp/security/awareness/vendor/programmingv1/b08_03.html

[1] 汚染データ：入力元が信用できない汚染されているかもしれないデータ
[2] サニタイズ処理：この場合はファイルパスなので、「\」や「/」を含んでいるかどうか、NTFS ストリーム指定があるかどうかなどのバリデーション(入力チェック)を指すだろう

*Copyright 2010 NTT Communications Corporation*

## 8. 履歴

- 2010 年 04 月 28 日 ：ver1.0 最初の公開

## 9. 最新版の公開URL

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

## 10. 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

*Copyright 2010 NTT Communications Corporation*