文字コードに潜むセキュリティ

Internet Week 2010

Yosuke HASEGAWA

http://j.mp/yosuke

自己紹介

長谷川陽介 - はせがわようすけ

- ネットエージェント株式会社研究開発部
- Microsoft MVP
  for Consumer - Security 2005/10-2011/09
- Ineternet Explorer、Mozilla Firefoxを始め脆弱性を多数発見
- 最近は JavaScript の難読化を研究
  e.g. jjencode, aaencode
- http://utf-8.jp/

今日の話題

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

はじめに

文字コードとセキュリティ
何が関係あるの?

文字コードとセキュリティ 何が関係あるの?

- テキストパーサの使用頻度が増加
  - Webアプリケーションの台頭
  - XMLやJSONのようなテキストデータ
- レガシーな文字コードからUnicodeへの移行に伴う混乱
  - EUC-JPやShift_JIS等とUnicodeの混在

文字コードとセキュリティ 何が関係あるの?

- 機械的な処理以外でも問題に繋がる可能性
  - 視覚的に似ている文字
  - 利用者の錯誤を誘因
- 攻撃者にとっては強力な道具となり得る

今日の目的

- 開発者として、文字コードに関連する脆弱性について攻撃方法だけでなく対策するための方法についても学ぶ
  - 文字コードやクロスサイトスクリプティングといった基礎についてはすでに知っているものとして話をします。

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

比較の一致/不一致

比較の一致/不一致

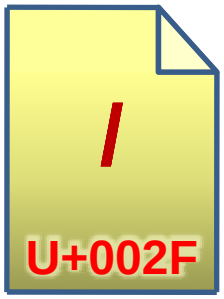
- 文字列の比較検出
  - セキュリティのための基本的な処理
  - 「安全な文字列の確認」や「危険な文字列の検出」など
- 開発者の意図と異なる比較結果
  - フィルタ処理のバイパス
  - 侵入検知(IDS等)の回避

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

UTF-8の非最短形式

- UTF-8では、ひとつの文字を複数のバイト列のパターンで表現可能
- 規格上は最短形式のみを許容



Valid

0x2F

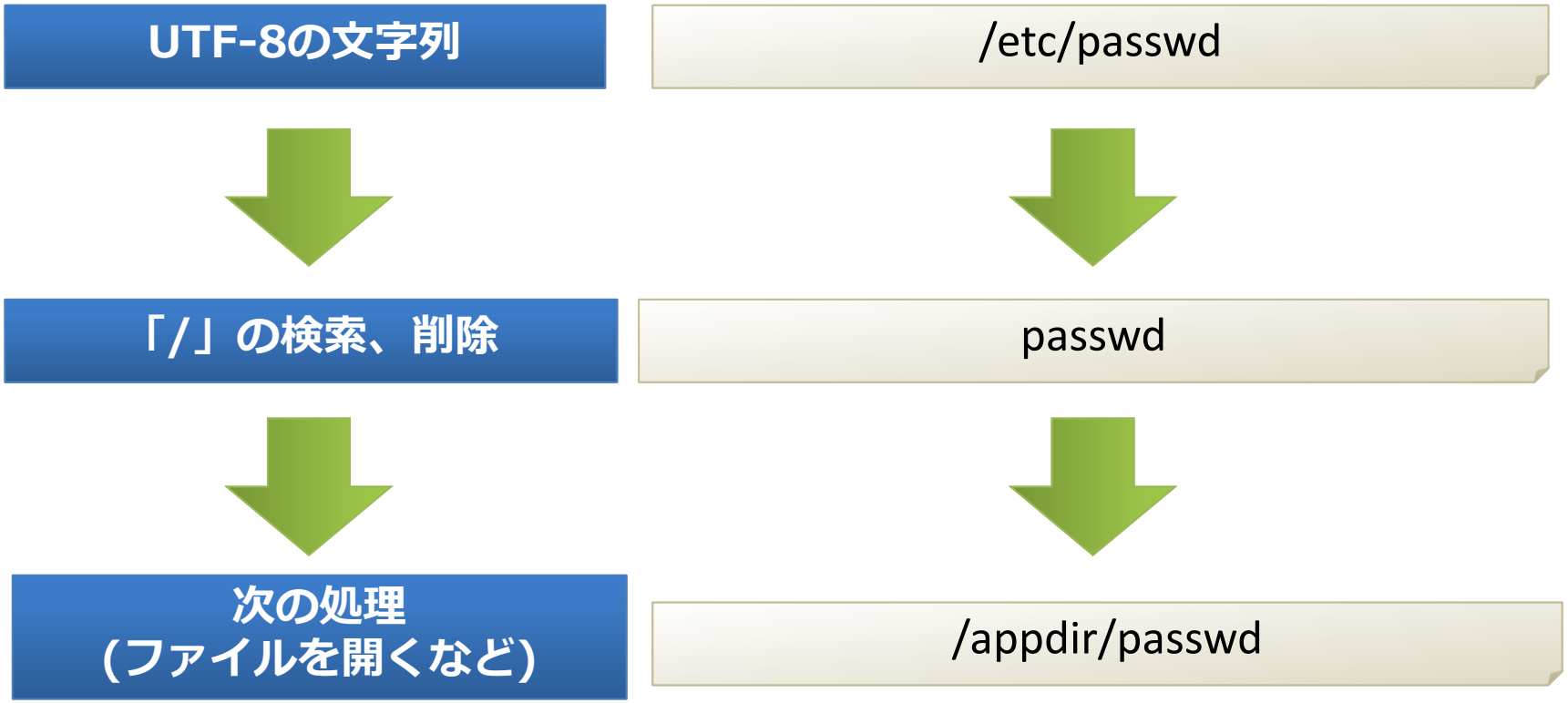
Invalid

0xC0 0xAF

0xE0 0x80 0xAF

0xF0 0x80 0x80 0xAF

UTF-8の非最短形式



404 エラー

UTF-8の非最短形式

UTF-8の文字列	..(0xC0 0xAF)etc(0xC0 0xAF)passwd
「/」の検索、削除	..(0xC0 0xAF)etc(0xC0 0xAF)passwd
次の処理 (ファイルを開くなど)	/appdir/../etc/passwd

非最小形式を許容

パストラバーサル

UTF-8の非最短形式

- 冗長なバイト表現を使ってフィルタを回避する攻撃手法
- 伝統的な攻撃手法のひとつ
  - MS00-057(IIS)などが有名
  - Nimdaで使用された

UTF-8の非最短形式

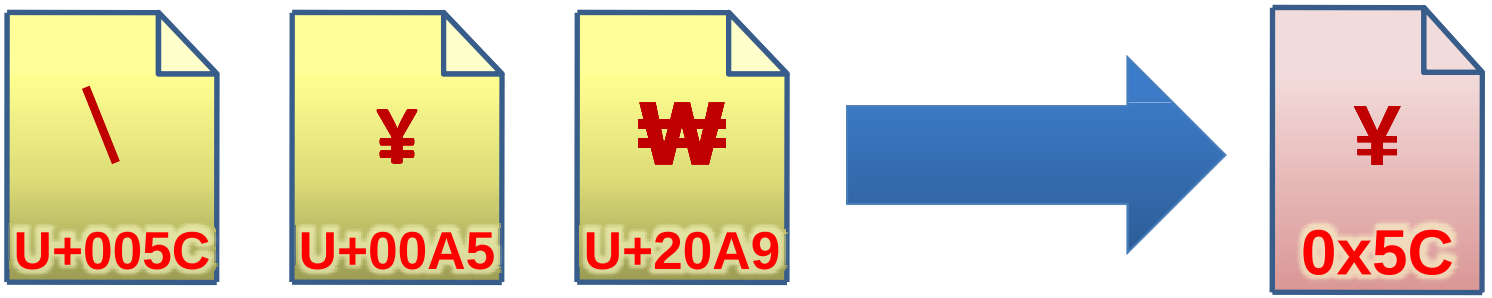
開発者としての対策

- UTF-16等に変換してから処理する
- 変換には自前のライブラリ等は使用しない
- メジャーなライブラリであれば通常は対策されている
  - 仮に問題があっても責任分界点が明確

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

多対一の変換

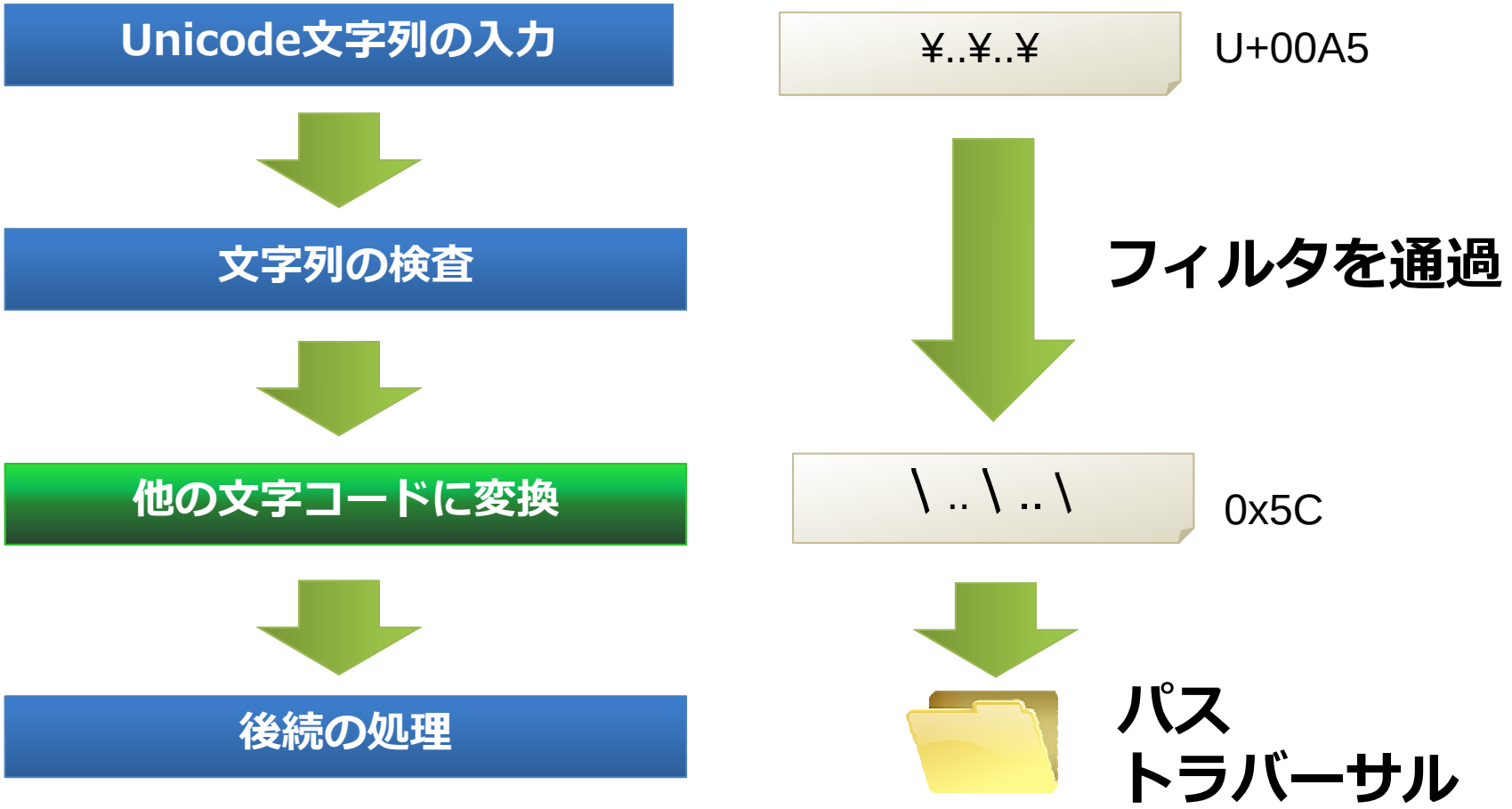


Unicode

Shift_JIS、
EUC-JPなど

- 文字集合の変換は多対一で行われることがある
- 特にUnicodeからの変換に注意

多対一の変換


Unicode文字列の入力
文字列の検査
他の文字コードに変換
後続の処理
¥..¥..¥
U+00A5
フィルタを通過
\ .. \ .. \
0x5C
パス
トラバーサル

多対一の変換

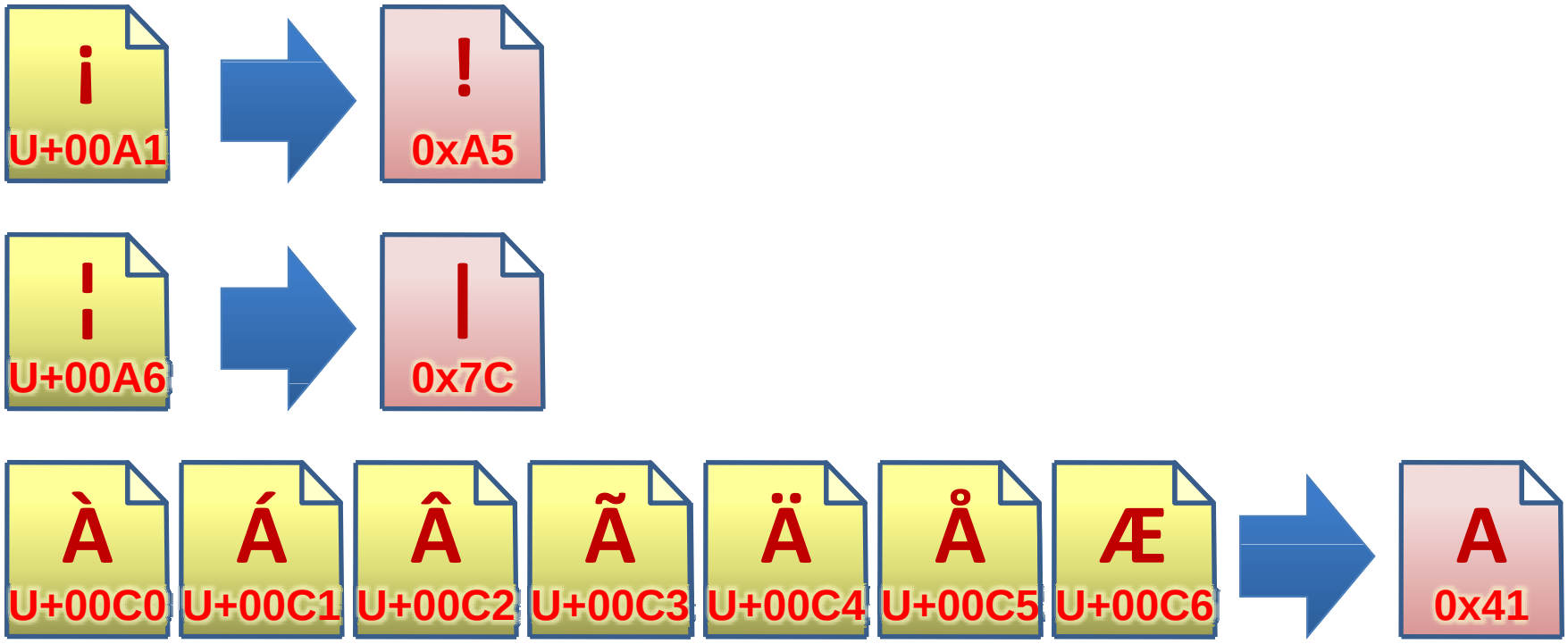
- 意図せず変換されることもある
  - NTFSはUnicodeでファイル名を保持
  - ANSIでファイル名を扱うAPIを呼び出す

多対一の変換

- ファイル名に¥(U+00A5)を使用可能
- Unicode非対応のアプリケーション/DLLではパストラバーサルが発生

多対一の変換

- 「¥」以外も多数の文字が多対一で変換



多対一の変換

開発者としての対策

- エンドツーエンドでUnicodeのまま文字列を扱い、文字集合は変換しない
- (変換するとしても)文字列の検査後には行わない

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

大文字と小文字

- こんな要望を受けたことはありませんか？

ユーザから入力される名前/ファイル名/パスワードは
大文字小文字を同一視して処理してください

大文字と小文字

- 大文字と小文字の同一視の定義は、言語文化により異なる

大文字と小文字

単語	一致	不一致
Gif / GIF	 アメリカ	 トルコ
Maße/MASSE	 ドイツ	 アメリカ
Maße / Masse	 スイス	 ドイツ  アメリカ

「Windowsプログラミングの極意」,株式会社アスキー,ISBN978-4-7561-5000-4,P.340より

大文字と小文字

- 全角文字と半角文字も考慮?
  - 全角大文字: ＡＢＣＤ
  - 半角大文字: ABCD
  - 全角小文字: ａｂｃｄ
  - 半角小文字: abcd

大文字と小文字

- 既存の文字列比較関数のルールを把握するのも難しい
- Visual C++ CRT / Win32 API

stricmp wcsicmp _stricmp _wcsicmp
_mbsicmp _stricmp_l _wcsicmp_l _mbsicmp_l
CompareString CompareStringEx
CompareStringOrdinal lstrcmpi IntlStrEqNI
StrCmpI StrCmpIC StrCmpNIC StrIsIntlEqual
....

大文字と小文字

開発者としての対策

- 大文字、小文字の差でセキュリティ上の分界点をつくらない
- 大文字小文字のルールの明確化と動作の確認

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

Unicode正規化

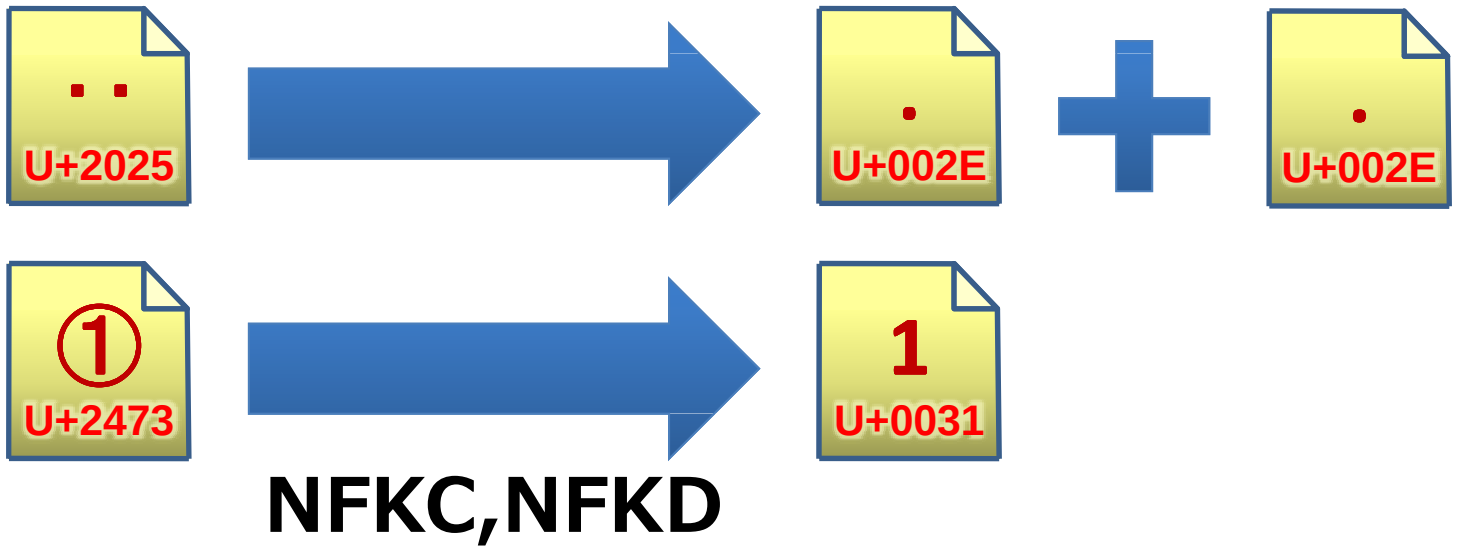
- Unicodeでは文字の分解、合成をサポート
  - 見た目は同じでもバイト列が異なる表現

Unicode正規化

- Unicodeでは4種類の正規化方法を規定
  - **NFC** Normalization Form Canonical Composition
  - **NFD** Normalization Form Canonical Decomposition
  - **NFKC** Normalization Form Compatibility Composition
  - **NFKD** Normalization Form Compatibility Decomposition

- 正規化した文字列から元の文字列は復元できない

Unicode正規化



- 正規化(NFKC、NFKD)により意味の異なるバイト列に変化

Unicode正規化


Unicode文字列の入力
文字列の検査
正規化(NFKD,NFKD)
後続の処理
¥･･¥･･¥
U+2025
フィルタを通過
¥..¥..¥
U+005C
パス
トラバーサル

Unicode正規化

開発者としての対策

- 文字列の検査後に正規化を行わない

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

不正なバイト列の埋め込み

- 不正なバイト列を与えたときの処理
  - 不正なバイト列を無視(切り捨てる)
  - 想定外の文字に変換
- こういった処理が脆弱性を生むことがある

不正なバイト列の埋め込み

- Firefox 2.0.0.12以前
  - charset=Shift_JIS のときに0x80を無視

```
<s[0x80]c[0x80]r[0x80]ipt>
    alert(1)
</s[0x80]c[0x80]r[0x80]ipt>
```

不正なバイト列の埋め込み

- IE 6, IE7, IE8
  - 0x00を無視する

```
<s[0x00]c[0x00]r[0x00]ipt>
    alert(1)
</s[0x00]c[0x00]r[0x00]ipt>
```

不正なバイト列の埋め込み

- IE6,7のMLang
  - 文字エンコーディング変換DLL
- 不正なバイト列を与えたとき
  - もとのバイト列に存在しない「"<>」などが生成され、XSSにつながる

不正なバイト列の埋め込み

- MLang：文字エンコーディング変換DLL
- IE内部では文字列をUnicodeで処理する

不正なバイト列の埋め込み

```
<meta http-equiv="Content-Type"
    content="text/html; charset=XXXXX" />
...
<input
value="(0xNN)(0xNN)(0xNN)onmouseover=alert(1)//
(0xNN)(0xNN)(0xNN)" type="text">
```

(0xNN)は文字コードXXXXXにおいて不正なバイト列

```
<input value="??"onmouseover=alert(1)//??""
type="text">
```

もとのバイト列に存在しない「"」が生成され、XSSにつながる

不正なバイト列の埋め込み

開発者としての対策

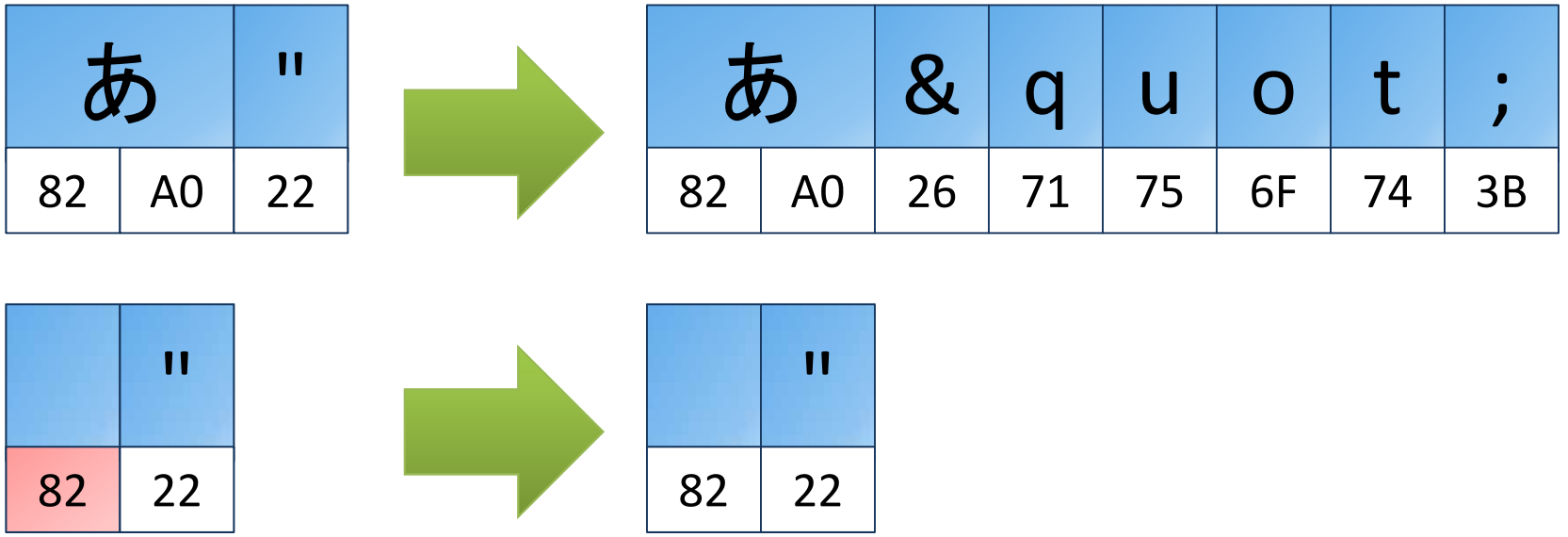
- 他の文字コードに変換
  - 内部:UTF-8 → 外部:EUC-JP等
- 文字列を適切なバイト列で構成する
  - PHP: mb_check_encoding など。

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

先行バイトの埋め込み

- マルチバイト文字の先行バイトだけを与えることでフィルタを回避



不正なバイト列の埋め込み

```
name:
  <input type=text value="[0x82]">
e-mail:
  <input type=text value=" onmouseover=...//">
```

- Shift_JISの先行バイトである 0x82 が後続のダブルクォートを無効にしている

不正なバイト列の埋め込み

```
UTF-8
 http://example.com/?%3cscript%20%E2%3Ealert(1);...
 http://example.com/?%E2%22onmouseover=alert(1)
Shift_JIS
 http://example.com/?%3cscript%20%81%3E%3ealert(1);...
EUC-JP
 http://example.com/?%3cscript%20%E0%3Ealert(1);...
 http://example.com/?%E0%22onmouseover=alert(1)
```

- IE8ベータ版ではXSS Filterを回避可能

不正なバイト列の埋め込み

開発者としての対策

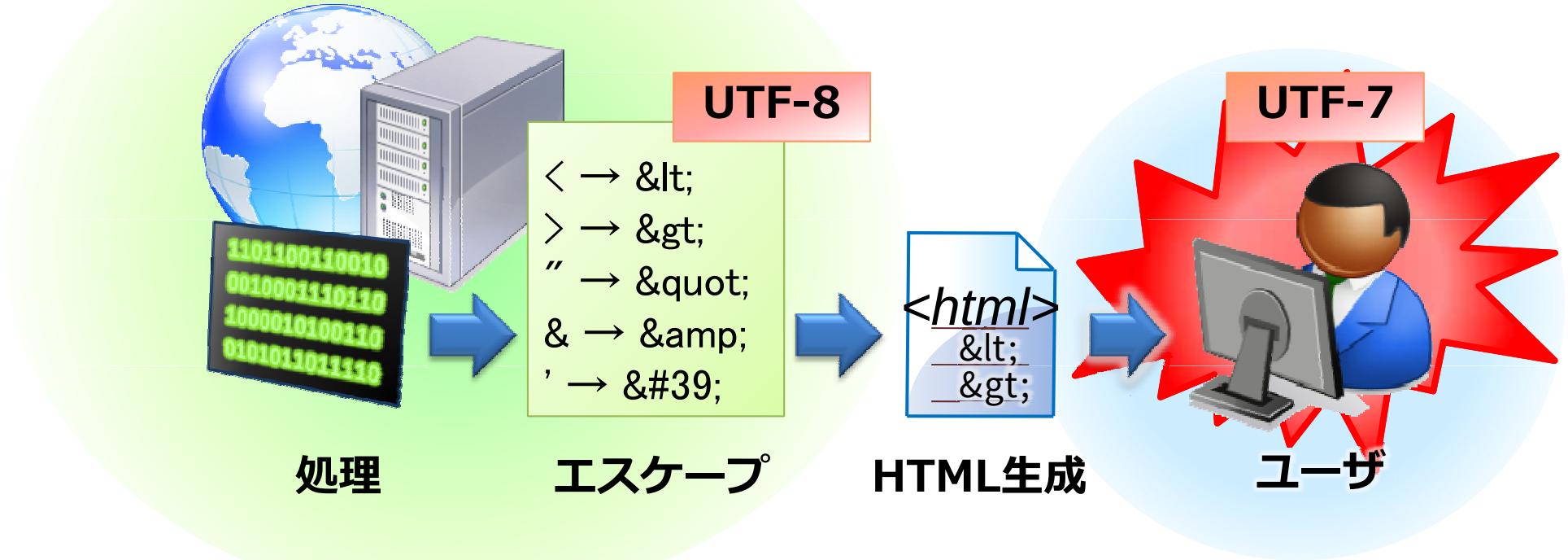
- 他の文字コードに変換
  - 内部:UTF-8 → 外部:EUC-JP等
- 文字列を適切なバイト列で構成する
  - PHP: mb_check_encoding, htmlspecialchars(第三引数も指定) など。

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

エンコード情報の不一致

- サーバ側とクライアント側で異なるcharsetと解釈される
  - 典型的には UTF-7 によるXSS



エンコード情報の不一致

- 典型的にはUTF-7によるXSSが有名
- charsetが不明瞭な場合にIEがUTF-7だと解釈することでXSSが発生

エンコード情報の不一致 UTF-7 XSS

- そもそもUTF-7とは?
  - Unicodeのエンコード形式のひとつ
  - 非ASCII文字や記号類がbase64でエンコードされる

```
<div class="main">
    abcdあいう
</div>
```

```
+ADw-div class+ADOAIg-main+ACIAPg-
    abcd+MEIwRDBG-
+ADw-/div+AD4-
```

エンコード情報の不一致 UTF-7 XSS

- メタキャラクタ(<>"など)を使わずにHTMLを記述できる
- IEにHTMLがUTF-7だと解釈させることで<script>が動作

```
<div>
+ADw-script+AD4- alert(1) +ADw-/script+AD4-
</div>
```

エンコード情報の不一致 UTF-7 XSS

- IEがHTMLをUTF-7扱いする条件
  - charsetが指定されていない
  - IEが理解できないcharsetが指定されている
  - 偽の<meta>を攻撃者が注入できる

エンコード情報の不一致 UTF-7 XSS

- charsetが指定されていない

```
HTTP/1.1 200 OK
Content-Type: text/html
...
<html>
<head>
  <meta http-equiv="content-type"
    content="text/html">
</head>
<body>
+ADw-script+AD4- alert(1) +ADw-/script+AD4-
...
```

エンコード情報の不一致 UTF-7 XSS

- IEが理解できないcharsetが指定されている

```
<meta http-equiv='content-type'
    content='text/html;charset=CP932'>
+ADw-script+AD4-
    alert(document.cookie);
+ADw-/script+AD4-
```

  - CP932/MS932/utf8/eucjp などは登録されていない

エンコード情報の不一致 UTF-7 XSS

- IEが理解できないcharsetが指定されている

  http://www.google.com/search?oe=CP932&q=%2bADw-...

  http://www.google.com/search?oe=CP950&q=%2bADw-...

  http://search.yahoo.com/search?eo=EUC&p=%2bADw-...

エンコード情報の不一致 UTF-7 XSS

- 偽の<meta>を攻撃者が注入できる
  - 本来の<meta>より前に偽の<meta>を攻撃者が注入

```
<title>+ADw-/title+AD4-
 +ADw-meta http-equiv+AD0-'content-type'
 content+AD0-'text/html+ADs-charset+AD0-utf-7'+AD4-
</title>
<meta http-equiv='content-type'
  content='text/html;charset=euc-jp'>
```

エンコード情報の不一致 UTF-7 XSS

- UTF-7 XSSへの開発者としての対策
  - charsetをHTTPレスポンスヘッダで明記
  - ブラウザが解釈可能なcharset名とする
  - <meta>より前に攻撃者がコントロールできる文字列を配置しない

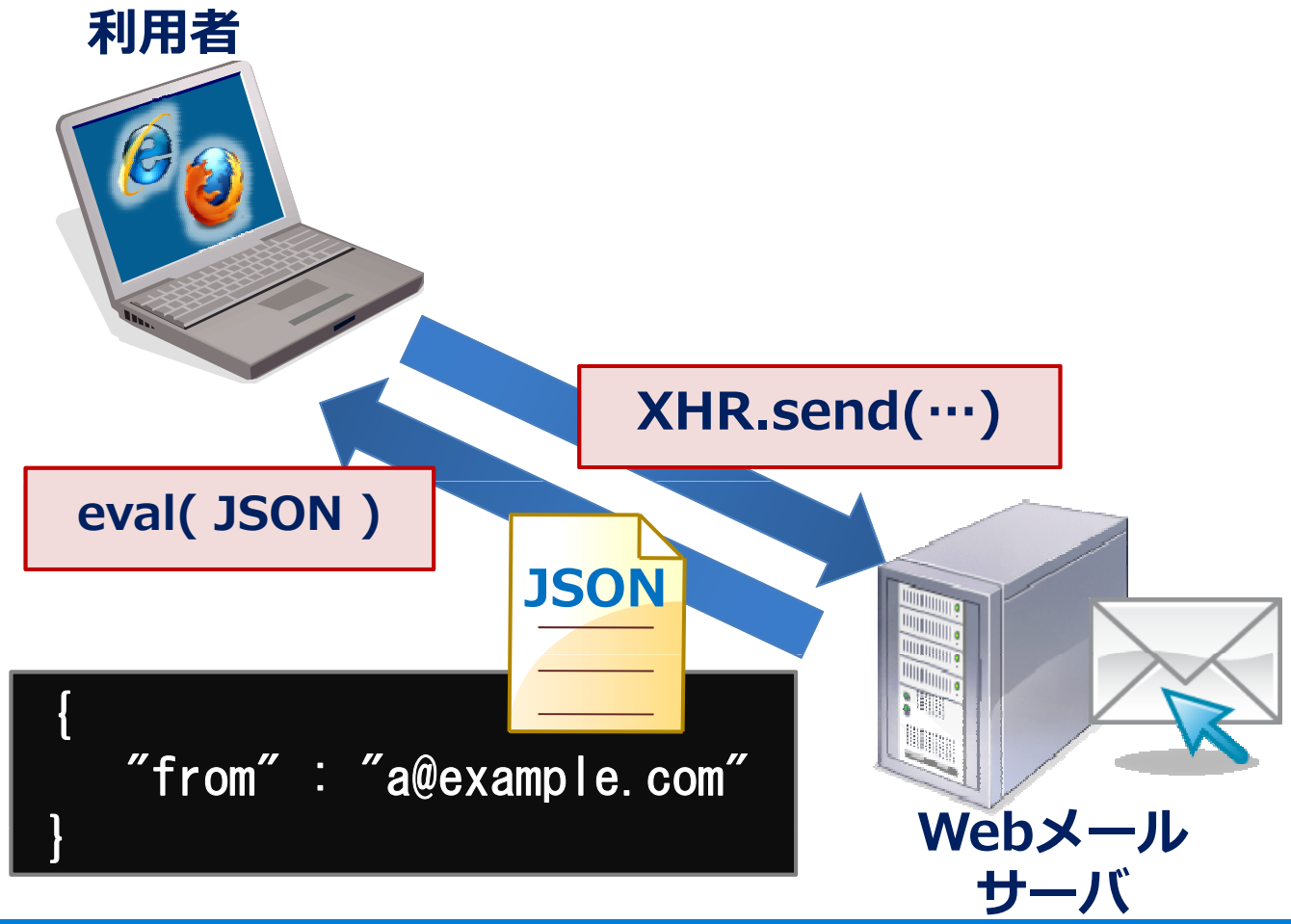
エンコード情報の不一致

UTF-7による問題はIEのXSS以外にも。

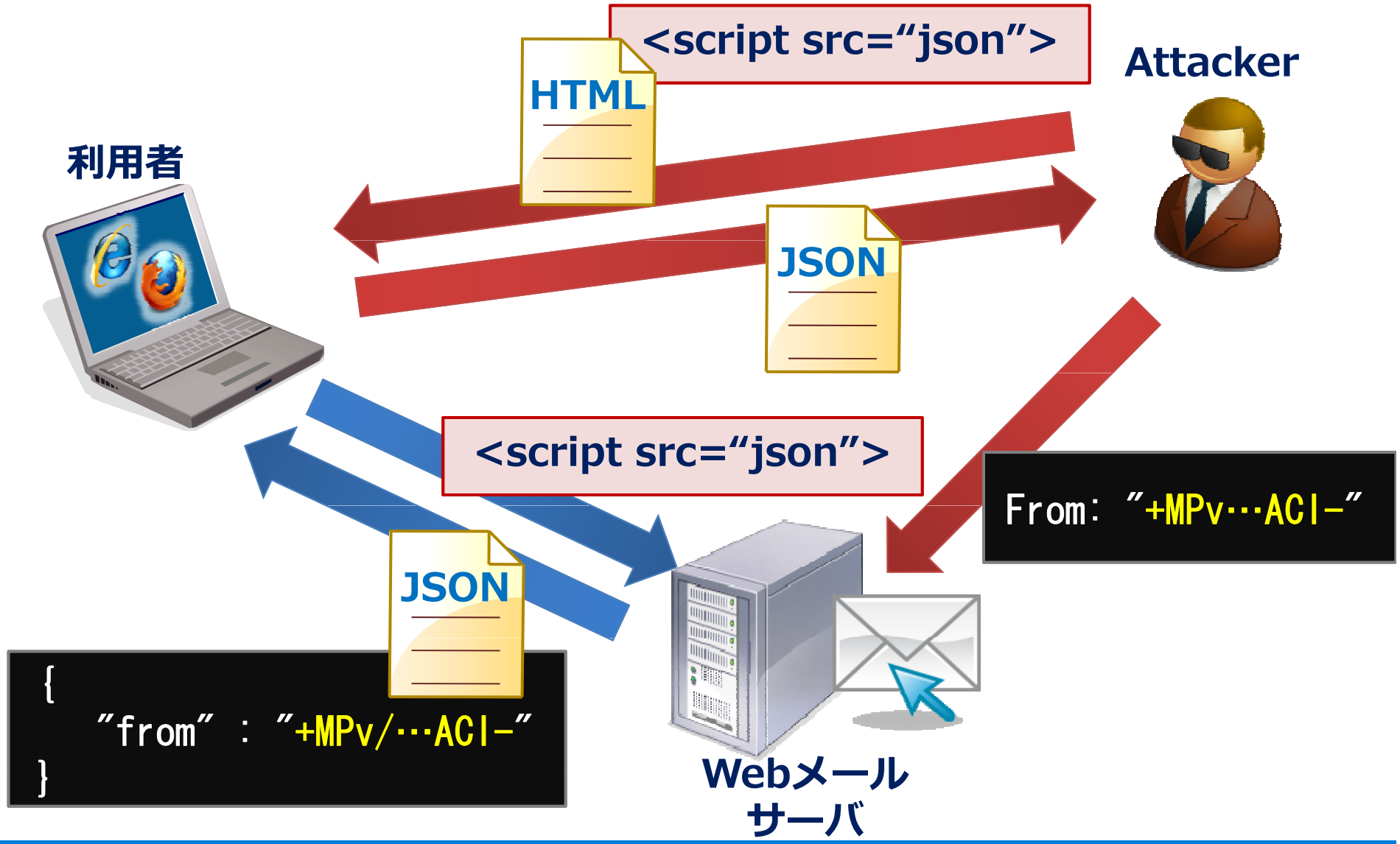
- UTF-7によるJSON Hijack
  - 機密情報を含むJSON
  - 攻撃者がJSON内の一部をコントロール可能
  - 例えばWebメールの新着通知など
- 攻撃者がJSON内のデータを盗み見できる

```
{
    "name" : "Yosuke HASEGAWA",
    "mail" : "hasegawa@utf-8.jp",
    "subject" : "Welcome to IW2010"
}
```

エンコード情報の不一致 JSON Hijack



エンコード情報の不一致 JSON Hijack



エンコード情報の不一致 JSON Hijack

```
[
 {
  "name" : "abc+MPv/fwAiAH0AXQA7-var t+AD0AWwB7AClAIg-:+ACI-",
  "mail" : "hasegawa@utf-8.jp"
 },
 {
  "name" : "John Smith",
  "mail" : "john@example.com"
 }
]
```

攻撃者により挿入された文字列

奪取対象のJSON：http://example.com/newmail.json

エンコード情報の不一致 JSON Hijack

```
[
 {
  "name" : "abc"}];var t=[{"":"",
  "mail" : "hasegawa@utf-8.jp"
 },
 {
  "name" : "John Smith",
  "mail" : "john@example.com"
 }
]
```

奪取対象のJSON：http://example.com/newmail.json

エンコード情報の不一致 JSON Hijack

```
[
 {
  "name" : "abc"}];var t=[{"":"",
  "mail" : "hasegawa@utf-8.jp"
 },
 {
  "name" : "John Smith",
  "mail" : "john@example.com"
 }
]
```

奪取対象のJSON：http://example.com/newmail.json

```
<script src="http://example.com/newmail.json" charset="utf-7">
<script> alert( t[ 1 ].name + t[ 1 ].mail ); </script>
```

奪取対象の作成した罠ページ

エンコード情報の不一致 JSON Hijack

```
Content-Type: application/json; charset=utf-8

[
 {
  "name" : "abc"}];var t=[{"":"",
  "mail" : "hasegawa@utf-8.jp"
 },
 {
  "name" : "John Smith",
  "mail" : "john@example.com"
 }
]
```

レスポンスヘッダでcharsetを明記

IE6,7では攻撃者が
指定した側が優先される

```
<script src="http://example.com/newmail.json" charset="utf-7">
<script> alert( t[ 1 ].name + t[ 1 ].mail ); </script>
```

奪取対象の作成した罠ページ

エンコード情報の不一致 JSON Hijack

開発者としての対策

- ❖HTTPレスポンスヘッダでcharsetを明記
- ❖JSON内の + を ¥u002B にエスケープ

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

7ビットエンコーディングの解釈

- Internet Explorer, Outlook Expressの問題
- エンコーディングがUS-ASCIIのとき、最上位ビットが無視される

7ビットエンコーディングの解釈

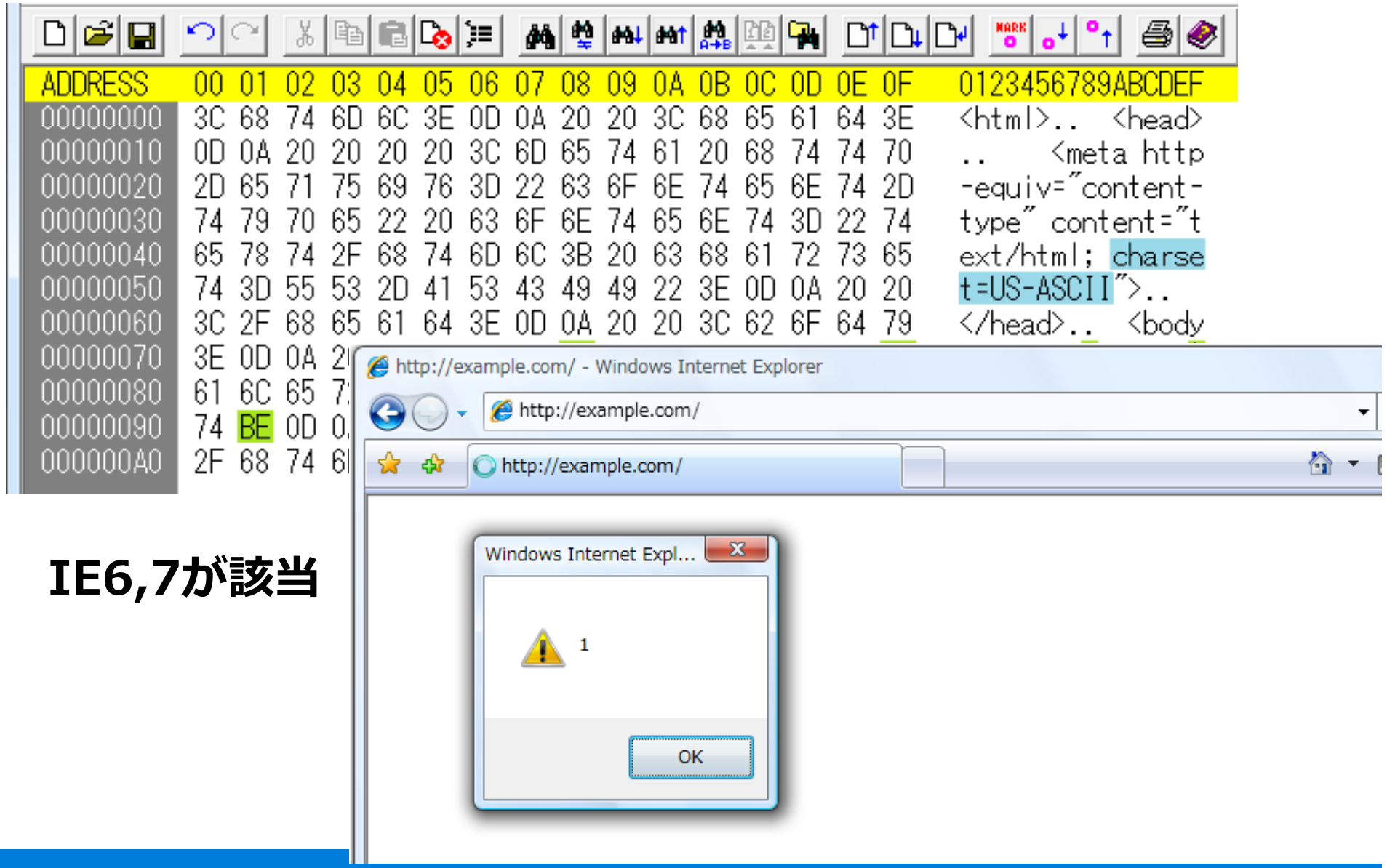
- US-ASCIIで最上位ビットが無視される

"	<	>
0x22	0x3C	0x3E
0010 0010	0011 1100	0011 1110

「	シ	セ
0xA2	0xBC	0xBE
1010 0010	1011 1100	1011 1110

ふたつの文字列は
等価として扱われる

7ビットエンコーディングの解釈



IE6,7が該当

7ビットエンコーディングの解釈

- Outlook Expressも同様
  - US-ASCIIの最上位ビットを無視

```
MIME-Version: 1.0
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit

This is test mail

begin 644 eicar.com
ﾍｶ#5/(5`EO$%O6S1<4%I8-3OH4%XI-T-#*3=])
$5)OT%2+5-4O4Y$O5)$+4%.
75$E625)54RU415-4+49)3$4A)$@K2"I#
`
end
```

M	ﾍ
0x4D	0xCD
0100 1101	1100 1101

6	ｶ
0x36	0xB6
0011 0110	1011 0110

7ビットエンコーディングの解釈

- 開発者としての対策
  - メールヘッダ、HTTPレスポンスヘッダでcharsetを明示
  - US-ASCIIを使用しない

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

表示上の欺瞞

表示上の欺瞞

- 人間に対する視覚的な効果
  - 利用者の錯誤を誘因
  - 攻撃者にとっては強力な道具となり得る

今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

視覚的に似た文字

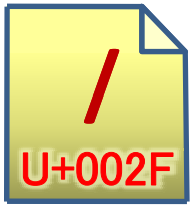
- 見た目の似ている文字
  - 例えば、数字の1(イチ)と小文字のl(エル)

http://bank1.example.com/
http://banklexample.com/

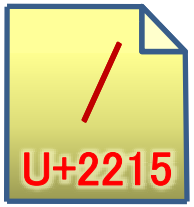
  - Unicodeだと、もっとたくさんの文字

視覚的に似た文字

❖スラッシュ



Solidus



Division Slash

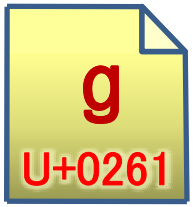
http://example.co.jp/t.example.com/foo/bar

Domain name

視覚的に似た文字

❖アルファベットの g

Lattin Small Letter G



Lattin Small Letter Script G

http://google.com/

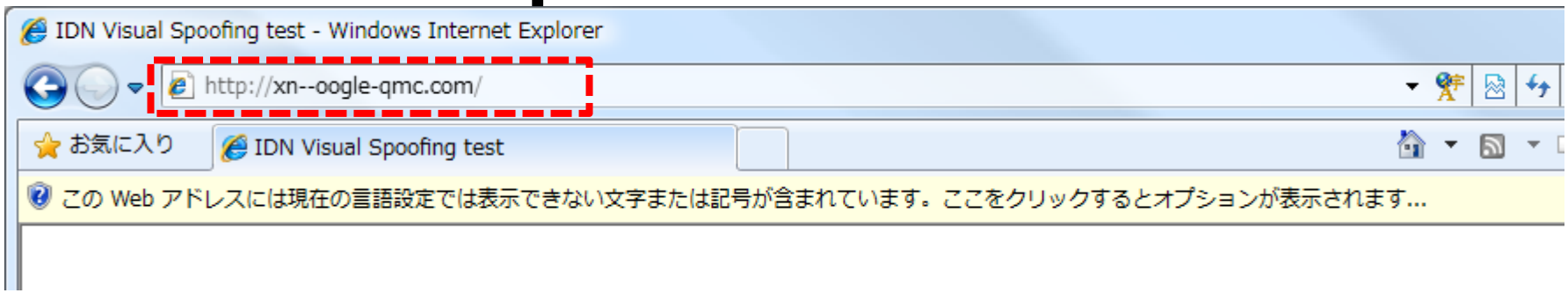
U+0261

視覚的に似た文字

- U+0261を実際にブラウザで表示

  http://google.com/
  http://xn--oogle-qmc.com

- Internet Explorer 8

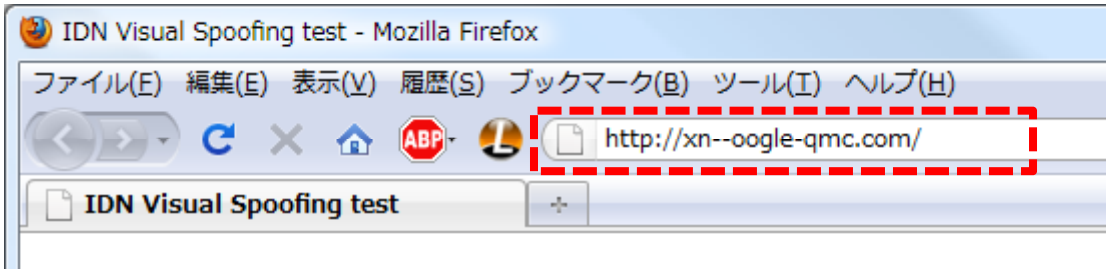


視覚的に似た文字

- U+0261を実際にブラウザで表示

  http://google.com/
  http://xn--oogle-qmc.com

- Mozilla Firefox 3.6

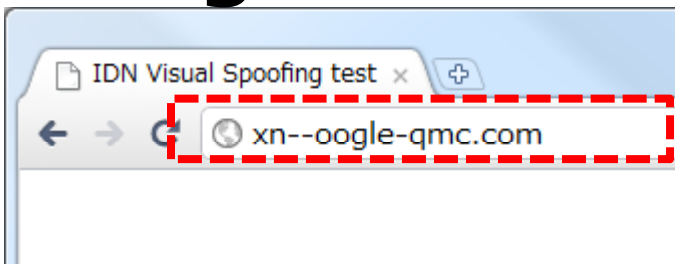


視覚的に似た文字

- U+0261を実際にブラウザで表示

  http://google.com/
  http://xn--oogle-qmc.com

- Google Chrome 7

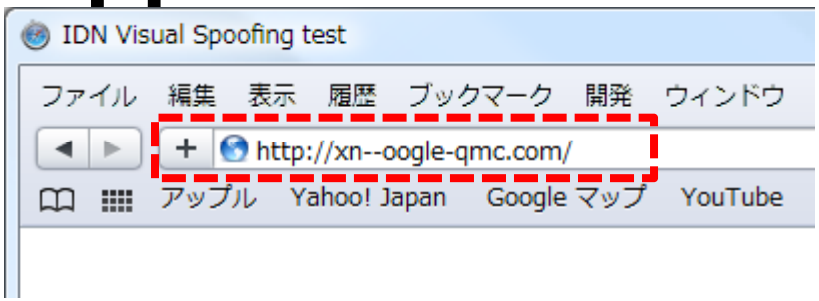


視覚的に似た文字

- U+0261を実際にブラウザで表示

  http://google.com/
  http://xn--oogle-qmc.com

- Apple Safari 5

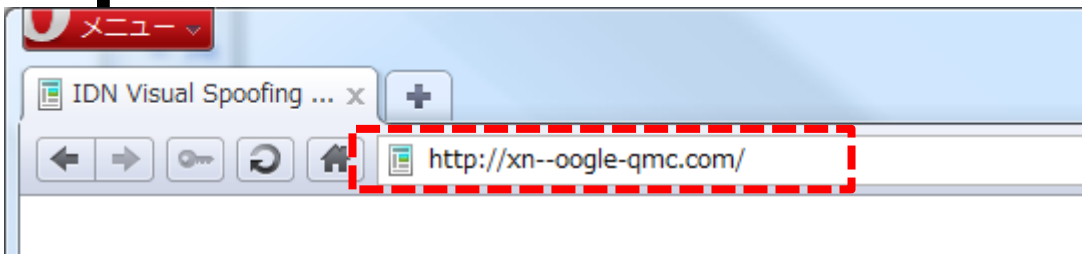


視覚的に似た文字

- U+0261を実際にブラウザで表示

  http://google.com/
  http://xn--oogle-qmc.com

- Opera 10

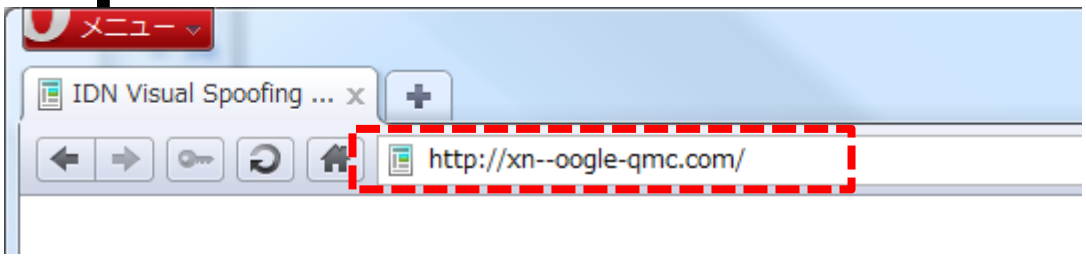


視覚的に似た文字

- U+0261を実際にブラウザで表示

  http://google.com/
  http://xn--oogle-qmc.com

- Opera 10



今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

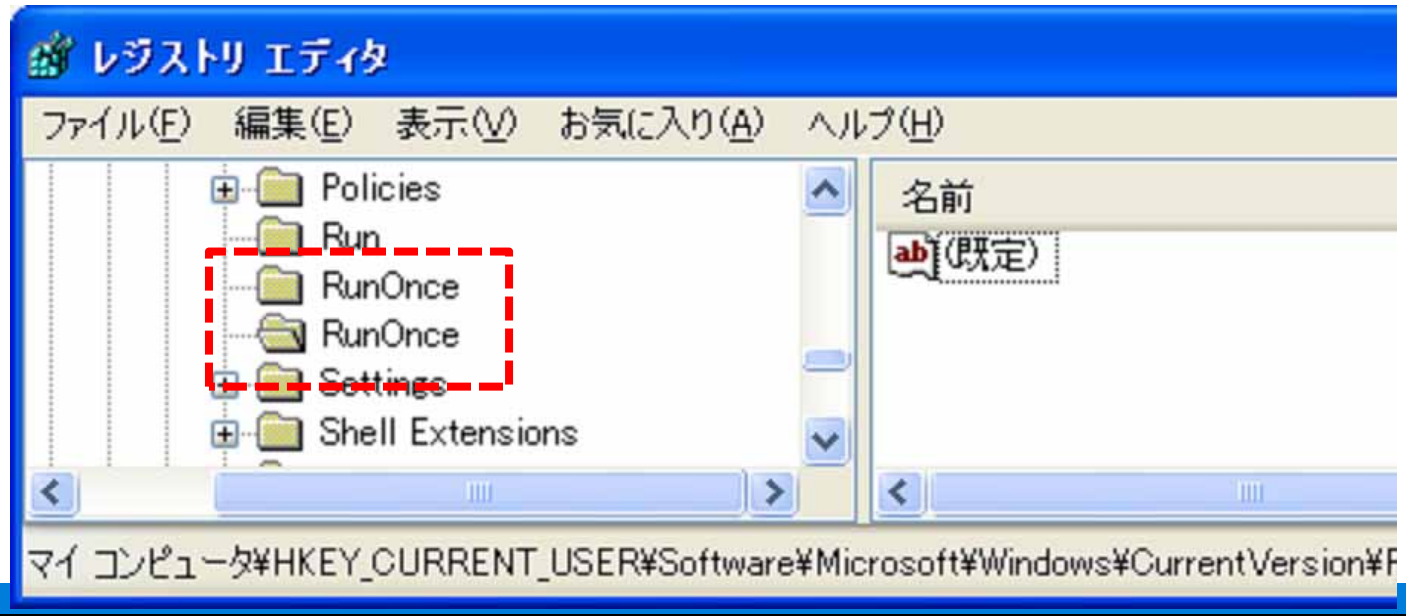
見えない文字

❖文字列に挿入しても見えない制御文字

U+200B	ZERO WIDTH SPACE
U+200C	ZERO WIDTH NON-JOINER
U+200D	ZERO WIDTH JOINER
U+202A	LEFT-TO-RIGHT EMBEDDING
U+FEFF	BYTE ORDER MARK (ZWNBSP)

見えない文字

- ファイル名やレジストリにも使用できる



今日の話題

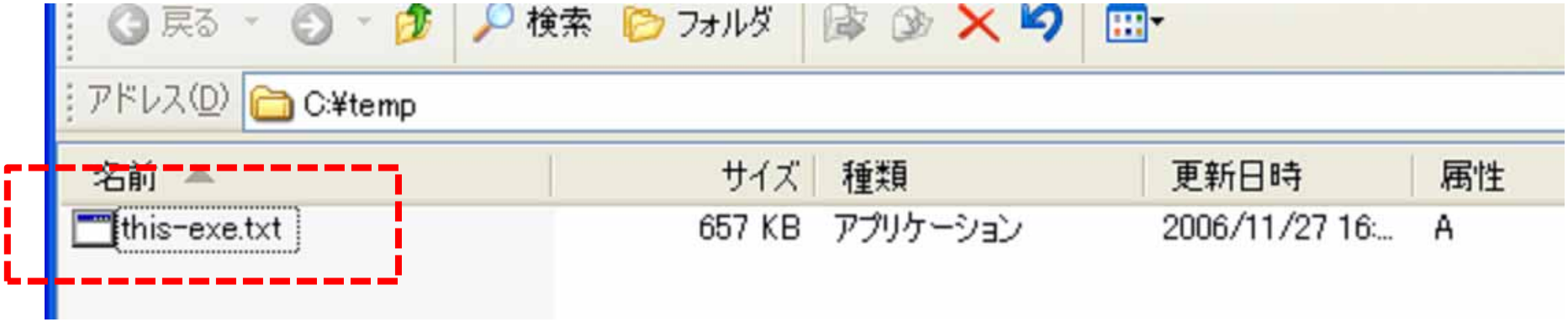
- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

双方向なテキスト

- Unicodeの双方向アルゴリズム
  - 文字列を部分的に左右反転して表示
  - U+202E - Right-to-Left Override;RLO

this-(U+202E)txt.exe	実際のバイト列
this-exe.txt	表示される文字列

双方向なテキスト



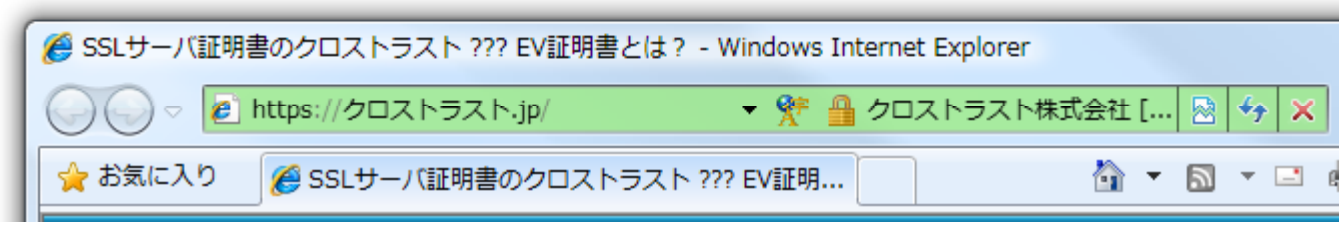
this-(U+202E)txt.exe

今日の話題

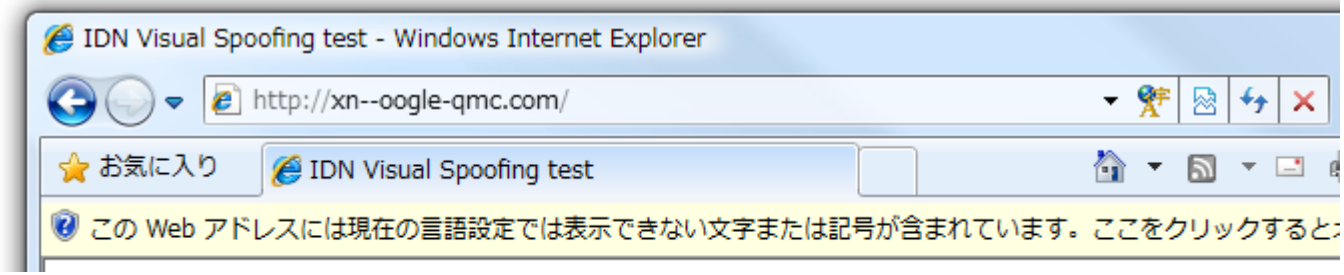
- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

表示上の欺瞞

- 対策
  - 複数の確認方法を用意する
  - SSLやEVSSLの活用



  - Punycodeでの表示



今日の話題

- はじめに
- 比較の一致/不一致
  - UTF-8の非最短形式
  - 多対一の変換
  - 大文字と小文字
  - Unicode正規化
  - 不正なバイト列の埋め込み
  - 先行バイトの埋め込み
  - エンコード情報の不一致
  - 7ビットエンコーディングの解釈
- 表示上の欺瞞
  - 視覚的に似た文字
  - 見えない文字
  - 双方向なテキスト
- まとめ

まとめ

まとめ

- 文字列の検査においては、検査後に文字コード変換や正規化を行わない
- 見た目だけに騙されない
- 文字コードを利用したセキュリティという分野は日本がトップレベルかつ未開拓

質問&連絡先

- メール
  - hasegawa@utf-8.jp
  - hasegawa@netagent.co.jp
- Twitter
  - @hasegawayosuke
- Web site
  - http://utf-8.jp/