

*Unraveling Unicode:
A Bag of Tricks for Bug Hunting*

Black Hat USA

July 2009

Chris Weber

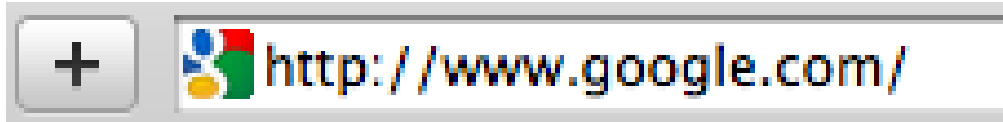
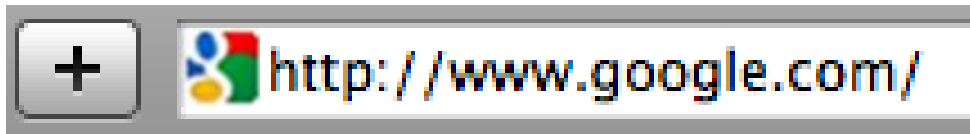
www.lookout.net

chris@casabasecurity.com

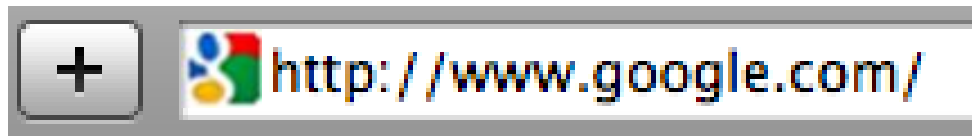


Casaba Security

Can you tell the difference?



How about now?



The Transformers

When good input turns bad

<scr*i*pt>

becomes

<scr*i*pt>

Agenda

Unicode Transformations

Agenda

- Unicode crash course
- Root Causes
- Attack Vectors
- Tools
 - Find Unicode issues in Web-testing
 - Visual Spoofing Detection

Unicode Transformations

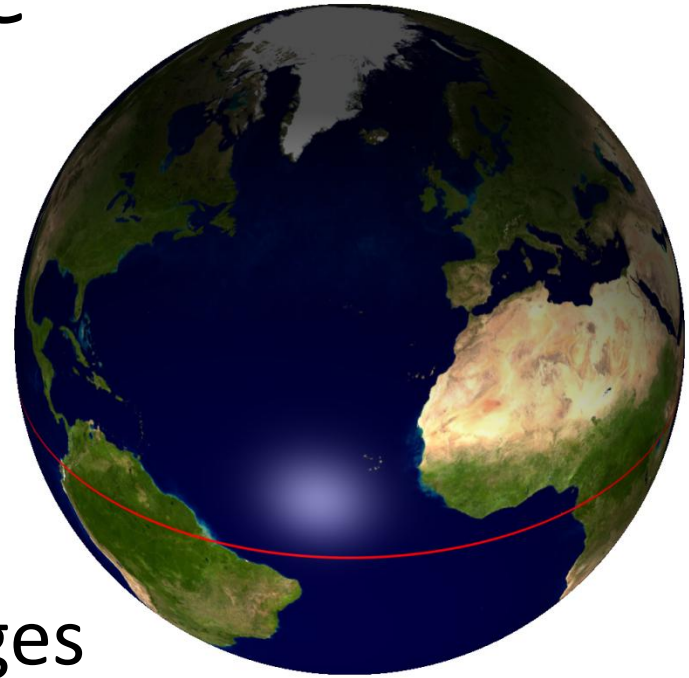
Agenda

- Unicode crash course
- Root Causes
- Attack Vectors
- Tools

Unicode Crash Course

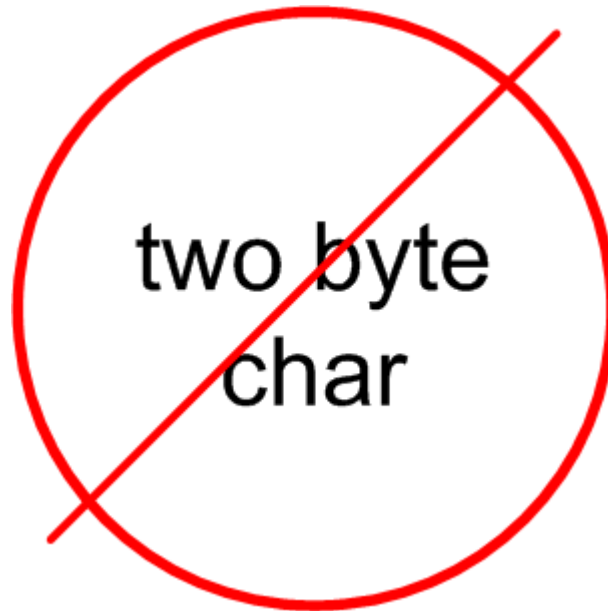
The Unicode Attack Surface

- End users
- Applications
- Databases
- Programming languages
- Operating Systems



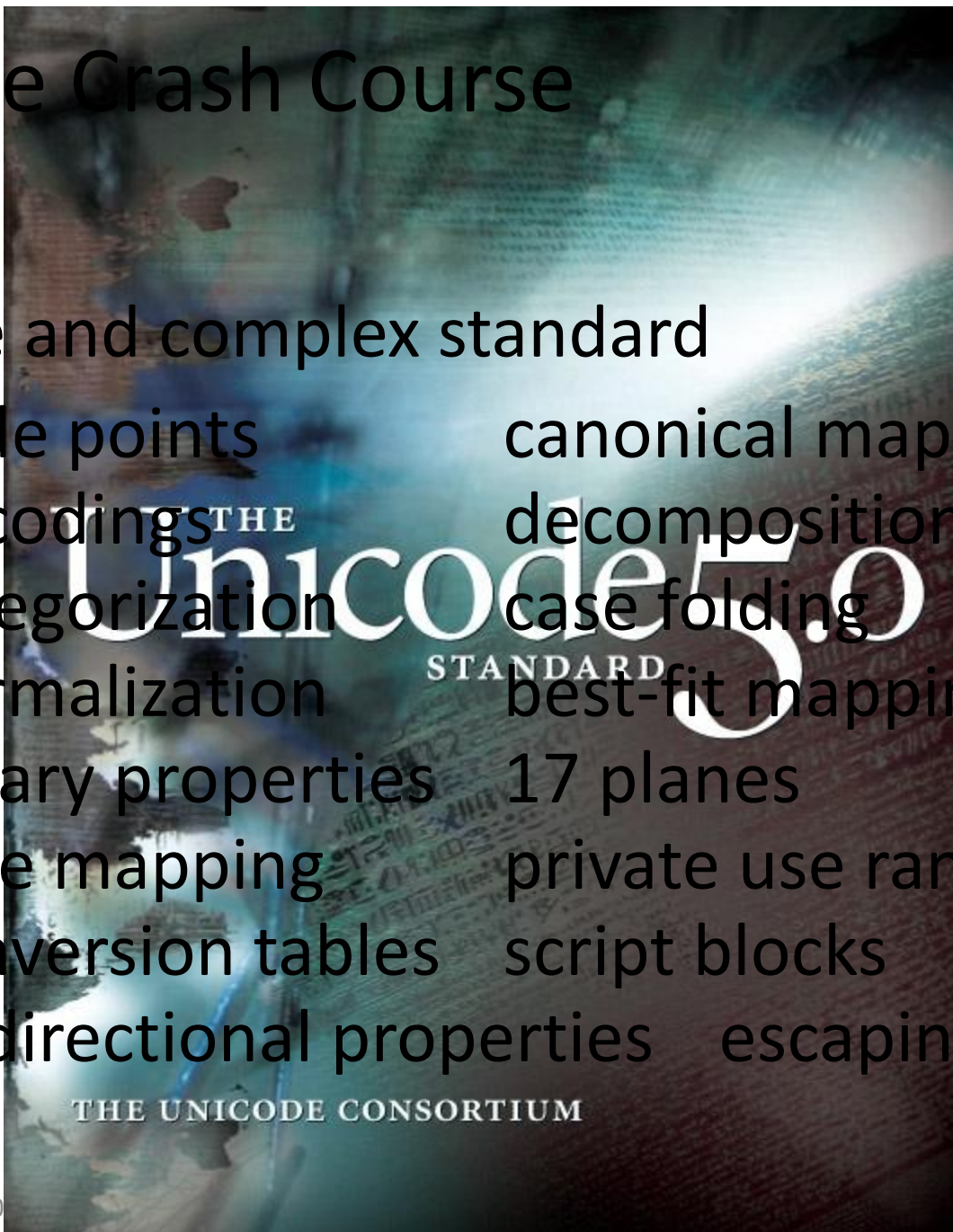
Unicode Crash Course

Unthink it



Unicode Crash Course

- A large and complex standard
 - code points
 - canonical mappings
 - encodings
 - decomposition types
 - categorization
 - case folding
 - normalization
 - best-fit mapping
 - binary properties
 - 17 planes
 - case mapping
 - private use ranges
 - conversion tables
 - script blocks
 - bi-directional properties
 - escapings



Unicode Crash Course

Code pages and charsets

Shift_jis

Gb2312

ISCII

Windows-1252

ISO-8859-1

EBCDIC 037



Unicode Crash Course

Ad Infinitum

- Unicode can represent them all
- ASCII range is preserved
 - U+0000 to U+007F are mapped to ASCII

Unicode Crash Course

Code points

- Unicode 5.1 uses a **21-bit scalar** value with space for over 1,100,000 **code points**:

U+0000 to U+10FFFF

Unicode Crash Course

Code Points

A = U+0041

Every character has a unique number

Unicode Crash Course

Category: Lu (Letter, Uppercase)

ToLower: U+0061

ToUpper: U+0041

Script: Basic Latin

A

U+0041

Latin capital letter A

Decomposition Type: none

Mapping: none

Binary Properties:

Hex Digit

Alphabetic

Lowercase

ID Start ...

Unicode Crash Course

Category: Ll (Letter, Lowercase)

ToLower: U+017F

ToUpper: U+0053

Script: Latin Extended-A

ŕ

U+017F

Latin small letter long S

Decomposition Type: <compat>

Mapping: U+0073

Binary Properties:

Alphabetic

Lowercase

ID Start ...

Unicode Crash Course

Encodings

UTF-8

- variable width **1 to 4 bytes** (*used to be 6*)

UTF-16

- Endianness
- Variable width **2 or 4 bytes**
- Surrogate pairs!

UTF-32

- Endianness
- Fixed width **4 bytes**
- Fixed mapping, no algorithms needed

Unicode Crash Course

Encodings and Escape sequences

U+FF21 FULLWIDTH LATIN CAPITAL LETTER A

`%EF%BC%A1`

`Ａ`

`Ａ`

`\xEF\xBC\xA1`

`\uFF21`

Unicode Transformations

Agenda

- Unicode crash course
- Root Causes
- Attack Vectors
- Tools

Unicode Transformations

Agenda

- Unicode crash course
- **Root Causes**
- **Attack Vectors**
- Tools

Unicode Transformations

Overview

- Unicode crash course
- Root Causes
 - Visual Spoofing and IDN's
 - Best-fit mappings
 - Normalization
 - Overlong UTF-8
 - Over-consumption
 - Character substitution
 - Character deletion
 - Casing
 - Buffer overflows
 - Controlling Syntax
 - Charset transformations
 - Charset mismatches
- Tools

Root Causes

Visual Spoofing

- Over 100,000 assigned characters
- **Many lookalikes** within and across scripts

A A A A A A A A A A A

Attack Vectors

IDN homograph attacks

Some browsers allow .COM IDN's

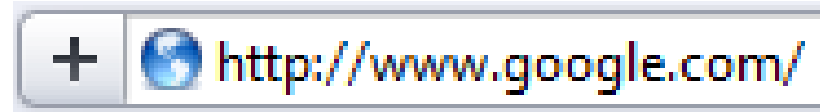
based on script family

– (Latin has a big family)

Attack Vectors

IDN homograph attacks

Safari



Attack Vectors

IDN homoglyph attacks

Opera



Attack Vectors

IDN homograph attacks

www.google.com is not www.google.com

Latin
U+0069

Latin
U+0261

gg

Root Causes

The state of International Domain Names

ICANN guidelines v2.0

– Inclusion-based

– Script limitations

– Character limitations

Deny-all default seems to be the right concept.

A script can cross many blocks. Even with limited script choices, there's plenty to choose from.

Great for domain labels, but sub domain labels still open to punctuation and syntax spoofing.

Attack Vectors

Visual spoofing Vectors

- Non-Unicode attacks
- Confusables
- Invisibles
- Problematic font-rendering
- Manipulating Combining Marks
- Bidi and syntax spoofing

Attack Vectors

Non-Unicode homoglyph attacks

rn can look like **m** in certain fonts

www.mullets.com is not www.rnullets.com

Latin
U+006D

Latin
U+0073 U+006E

Attack Vectors

Non-Unicode homoglyph attacks

Are you using mono-width fonts?

0 and O

1 and l

5 and S

Attack Vectors

Non-Unicode homoglyph attacks

Classic long URL's

```
http://login.facebook.intvitation.videomessageid-  
h048892r39.sessionnfbid.com/home.htm?/disbursements/
```

Attack Vectors

Single-script and The Confusables

www.αpple.com

// All Latin using Latin small letter Alpha 'α'

www.ƒacebook.com

// Mixed Latin/Greek with lunate sigma symbol 'ƒ'

www.аЬс.com

// All Cyrillic 'abc'

Attack Vectors

IDN homograph attacks

Browsers whitelist .ORG



Attack Vectors

IDN homoglyph attacks

Others don't necessarily but...



Attack Vectors

IDN homograph attacks

www.mozilla.org is not www.mozílla.org

Latin
U+0069

Latin
U+00ED

Attack Vectors

IDN Syntax Spoofing with / lookalikes

http://www.google.com/ path / file?.nottrusted.org




FULLWIDTH SOLIDUS
U+FF0F

(This case doesn't work anymore)

Attack Vectors

IDN Syntax Spoofing with / lookalikes

http://www.google.com/path/file.nottrusted.org



SOLIDUS
U+002F

(Normalized to a / U+002F)

Attack Vectors

IDN Syntax Spoofing with / lookalikes

http://www.google.com/path/file.nottrusted.org

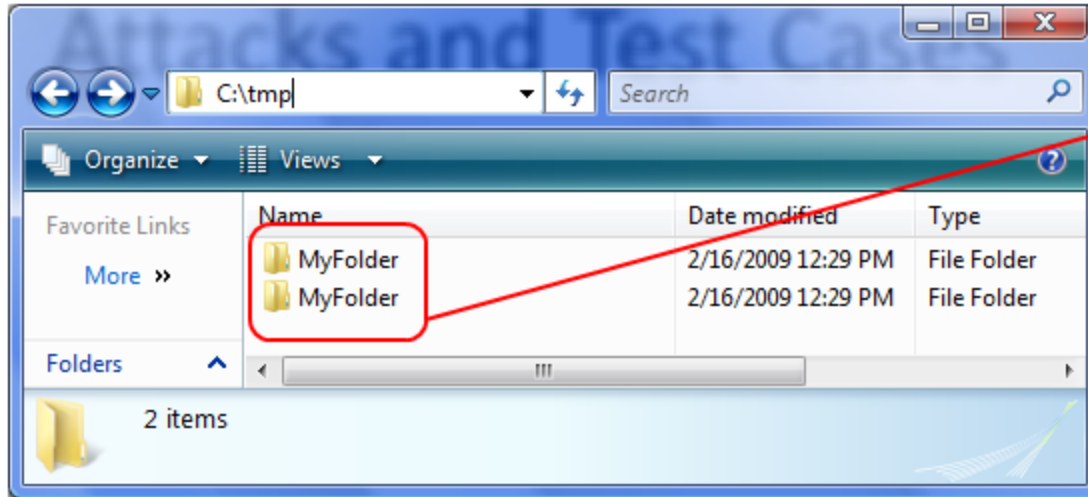


Katakana No
U+FF89

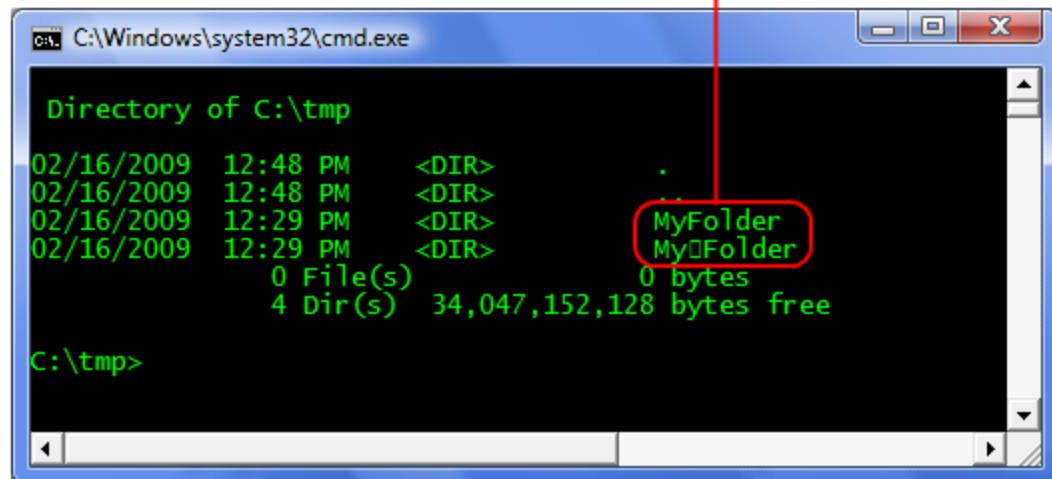
(However punctuation not required...)

Attack Vectors

The Invisibles

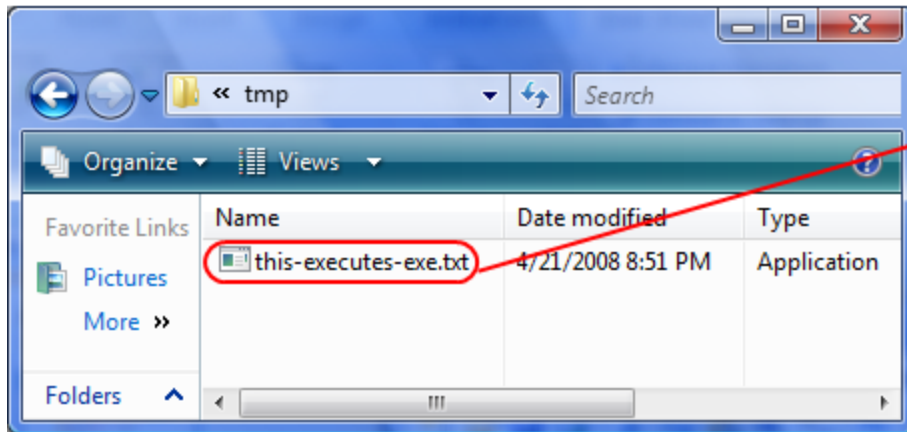


MyFolder
My[U+FEFF]Folder



Attack Vectors

Visual Spoofing with Bidi Explicit Directional Overrides



this-executes-[U+202E]txt.exe

Root Causes

Best-fit mappings

Commonly occur in charset transformations and even innocuous API's

Impact: Filter evasion, Enable code execution

When σ becomes s

U+03C3 GREEK SMALL LETTER SIGMA

When ' becomes ´

U+2032 PRIME

Root Causes

Guidance for Best-Fit mappings

- Scrutinize character/charset manipulation API's
- **Use** `EncoderFallback` **with** `System.Text.Encoding`
- **Set** `WC_NO_BEST_FIT_CHARS` **flag with**
`WideCharToMultiByte()`
- **Use Unicode end-to-end**

Case Study: Social Networking

Best-fit mappings

- A popular social networking site in 2008
- Implemented complex filtering logic to prevent XSS
 - **Attack:** Filter evasion, code execution
 - **Exploit:** Bypass filtering logic with best-fit mappings to leverage cross-site scripting
 - **Root Cause:** best-fit mappings

Case Study: Social Networking

Best-fit mappings

`-moz-binding()`

was not allowed, but....

– [**U+ff4d**]`oz-binding()`

would best-fit map!

Root Causes

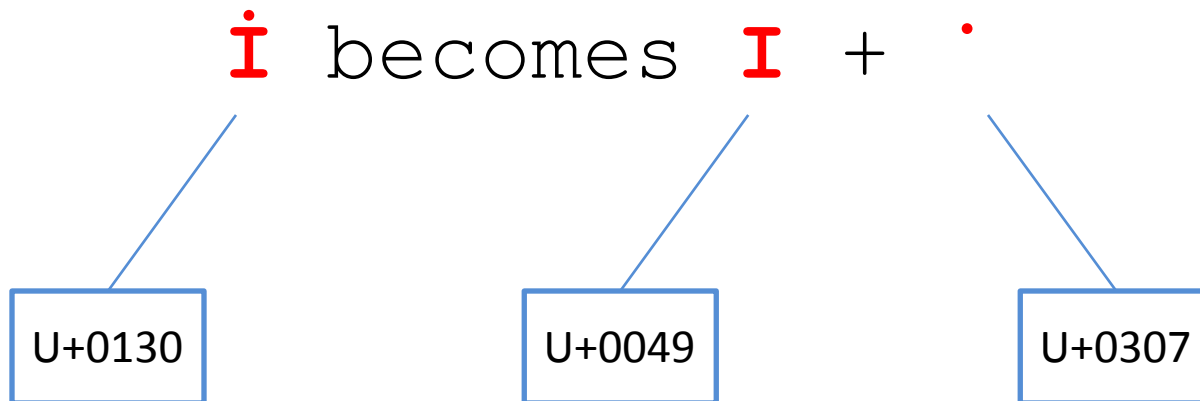
Normalization

Normalizing strings **after** validation is **dangerous**

Impact: Filter evasion, Enable code execution

Root Causes

Normalization

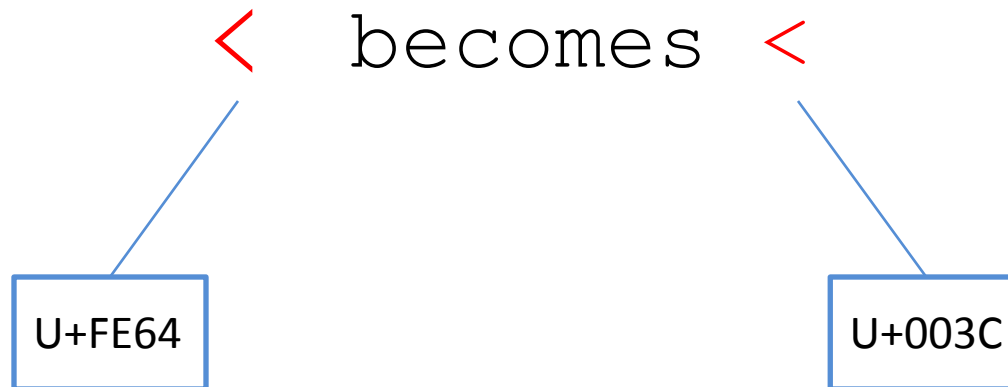


Root Causes

Normalization

But are there dangerous characters?

You bet... with NFKC and NFKD you could control HTML or other parsing



Root Causes

Normalization



`toNFKC ("< script>") = "<script>"`

Root Causes

Guidance for Normalization

Normalize strings **before** validation

NFKC first defense against Visual spoofing

Root Causes

Non-shortest form UTF-8

Non-shortest or overlong UTF-8

Impact: Filter evasion, Enable code execution

Application gets

%C0%A7

OS/Framework sees

%27

Database gets

'

Root Causes

Guidance for Non-shortest form UTF-8

- Unicode **specification forbids**
 - **Generation** of non-shortest form
 - **Interpretation** of non-shortest form for BMP
- Validate UTF-8 encoding (throw on error)

Attack Vectors

Directory traversal

How many ways can you say 

Attack Vectors

Normalization compatibility forms:

U+2024 U+2024 U+FF0F

%E2 %80 %A4 %E2 %80 %A4 %EF %83 %BF

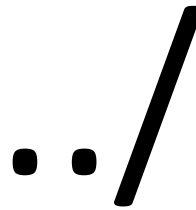
../

Best-fit mapping Windows-1252:

U+FF0E U+FF0E U+2215

%EF %BC %8E %EF %BC %8E %E2 %88 %95

. ./



UTF-8:

U+002E U+002E U+002F

%2E %2E %2F

../

UTF-8 overlong:

U+002E U+002E U+002F

%C0 %AE %C0 %AE %C0 %AF%

../

Root Causes

Handling the Unexpected

- Unassigned code points
 - U+2073
- Illegal code points
 - Half a surrogate pair
- Code points with special meaning
 - U+FEFF is the BOM
- **Impact:** Filter evasion, Enable code execution

Root Causes

Handling the Unexpected: Over-consumption

Over-consuming **ill-formed byte sequences**

** Big problem with MBCS lead bytes*

<41 C2 3E 41> becomes

<41 41>

Root Causes

Handling the Unexpected: Over-consumption

```
 "onerror="alert(1)"<br />
```

becomes

```
 "onerror="alert(1)"<br />
```


Root Causes

Handling the Unexpected: Character-substitution

Correcting insecurely rather than failing

- Substituting a ‘.’ or a ‘/’ would be bad

Root Causes

Handling the Unexpected: Character-deletion

“deletion of noncharacters” (UTR-36)



Root Causes

Handling the Unexpected: Character-deletion

<scr [**U+FEFF**] ipt> becomes <script>

Root Causes

Solutions for Handling the Unexpected

- Fail or error
- Use **U+FFFD** instead
 - A common alternative is ‘?’, which can be safe

Attack Vectors

Filter evasion

- Bypass filters, WAF's, NIDS, and validation
- Exploit delivery techniques
 - E.g. Cross-site scripting (buffer overflow of the Web)

Case Study: Apple and Mozilla

Safari and Firefox BOM consumption

- **Attack:** Filter evasion, code execution
- **Exploit:** Bypass filtering logic with specially crafted strings to leverage cross-site scripting
- **Root Cause:** Character deletion

```
<a href="java[U+FEFF]script:alert('XSS')>
```

Can be nastier:

```
<a h[U+FEFF]ref="java[U+FEFF]script:al[U+FEFF]ert('XSS')>
```

A Closer Look: The BOM


Category: Cf [Other, Format] 

Script: Common

Line Break: WJ [Word Joiner]

BOM
U+FEFF

ZERO WIDTH NO-BREAK SPACE
(BYTE ORDER MARK)

 Binary Properties:
Default Ignorable Code Point

UTF-8: EF BB BF

UTF-16LE: FF FE

UTF-16BE: FE FF

Root Causes

Casing

- Attackers manipulate casing operations to inject otherwise prohibited characters
- Casing can multiply the buffer sizes needed
- **Impact:** Filter evasion, Enable code execution

Root Causes

Casing

```
toLowerCase("i") == "i"
```

```
toLowerCase("script") == "script"
```

Root Causes

Casing

```
len(x) != len(toLower(x))
```

Root Causes

Guidance for Casing

- Perform casing operations **before** validation
- Leverage existing frameworks and API's
 - ICU, .Net

Root Causes

Buffer Overflows

- Incorrect assumptions about string sizes (chars vs. bytes)
- Improper width calculations
- **Impact:** Enable code execution

Root Causes

Buffer Overflows

Casing - maximum expansion factors

Operation	UTF	Factor	Sample	
Lower	8	1.5	Å	U+023A
	16, 32	1	A	U+0041
Upper	8, 16, 32	3	ï	U+0390

Source: *Unicode Technical Report #36*

Root Causes

Buffer Overflows

Normalization- maximum expansion factors

Operation	UTF	Factor	Sample
NFC	8	3X	□ U+1D160
	16, 32	3X	ﻧﻮ U+FB2C
NFD	8	3X	ï U+0390
	16, 32	4X	ï̈ U+1F82
NFKC/NFKD	8	11X	صلی اللہ علیہ وسلم U+FDFA
	16, 32	18X	

Source: *Unicode Technical Report #36*

Root Causes

Guidance for Buffer Overflows

- Know the difference between bytes and chars
- Secure coding
- Leverage existing frameworks and API's
 - ICU, .Net

Root Causes

Controlling Syntax

- White space and line breaks
 - E.g. when U+180E acts like U+0020
- Quotation marks
- **Impact:** Filter evasion, Enable code execution

Attacks and Exploits

Controlling syntax

- Manipulate HTML parsers and javascript interpreters
- Control protocols

Case Study: Opera

- Unicode formatter characters exploited for XSS
 - **Damage:** Filter evasion, controlling syntax
 - **Exploit:** Bypass filtering logic with specially crafted characters to leverage cross-site scripting.
 - **Root Cause:** Interpreting “white space”
 - A problem with HTML 4.0 spec?

Case Study: Opera

```
<a href=# [U+180E] onclick=alert () >
```

Case Study: Opera

Category: Zs [Separator, Space] 🚩

Script: Mongolian

Line Break: GL [Non-breaking ("Glue")]

MVS
U+180E

MONGOLIAN VOWEL SEPARATOR

Binary Properties:

🚩 White Space

Grapheme Base

Root Causes

Guidance for Controlling Syntax

- Question specifications
- Be careful...

Root Causes

Specifications

1) Character stability

- IDNA/Nameprep based on Unicode 3.2

2) Designs

- Specs are carefully designed but not always perfect
 - This could have been a problem:
 - “When designing a markup language or data protocol, the use of U+FEFF can be restricted to that of Byte Order Mark. In that case, any U+FEFF occurring in the middle of the file **can be ignored**, or treated as an error. ”
- HTML 4.01
 - Defines four whitespace characters and explicitly leaves handling other characters up to implementer.

Root Causes

Charset Transformations

- Converting between charsets is dangerous
- Mapping tables and algorithms vary across platforms
- **Impact:** Filter evasion, Enable code execution, Data-loss

Root Causes

Guidance for Charset Transformations

- Avoid if possible
- Use Unicode as the broker
- Beware the PUA mappings
- Transform, case, and normalize prior to validation and redisplay

Root Causes

Charset Mismatches

- Some charset identifiers are ill-defined
- Vendor implementations vary
- User-agents may sniff if confused
- Attackers manipulate behavior
- **Impact:** Filter evasion, Enable code execution

Root Causes

Charset Mismatches

Content-Type: charset=**ISO-8859-1**

Attacker-controlled input

<meta http-equiv="Content-Type" content="text/html;
charset=**shift_jis**" />

Root Causes

Guidance for Charset Mismatches

- Force UTF-8
- Error if uncertain

Unicode Transformations

Agenda

- Unicode crash course
- Root Causes
- Attack Vectors
- Tools

Unicode Transformations

Agenda

- Unicode crash course
- Root Causes
- Attack Vectors
- **Tools**

Tools

- **Watcher**
 - Passive Web-app security testing and auditing
- **Unibomber**
 - XSS autopwn testing tool

Tools

Watcher – Some of the Passive Checks Included

- Unicode transformation hot-spots
- User-controlled HTML
- Cross-domain issues
- Insecure cookies
- Insecure HTTP/HTTPS transitions
- SSL protocol and certificate issues
- XSS hot-spots
- Flash issues
- Silverlight issues
- Information disclosure

Tools

Watcher by Casaba Security

Totals (Alerts, Individual Issues)

Alert Filter: **Informational** High: 7, 19 Medium: 12, 18 Low: 0, 0 Information

Severity	Session ID	Type	URL
Medium	14	Set-Cookie HTTPOnly Attribute Not Set	www.nottrusted.com/wat
Medium	14	Set-Cookie Secure Attribute Not Set	www.nottrusted.com/wat
Medium	14	Set-Cookie Loosely Scoped Domain	www.nottrusted.com/wat
Informational	17	Charset not UTF-8	www.nottrusted.com/wat
High	18	Invalid Unicode ByteStream	www.nottrusted.com/wat
High	18	Null Bytes in ByteStream	www.nottrusted.com/wat
Medium	20	Flash allowScriptAccess Value	www.nottrusted.com/wat
Informational	21	Charset not UTF-8	www.nottrusted.com/wat
Medium	21	Flash crossdomain.xml Insecure Domain Reference	www.nottrusted.com/wat
Informational	30	Charset not UTF-8	www.nottrusted.com/wat
Medium	30	Silverlight clientaccesspolicy.xml Insecure Domain Reference	www.nottrusted.com/wat
High	32	User Controllable Location Header (Open Redirect)	www.nottrusted.com/wat
High	38	User Controllable Charset	www.nottrusted.com/wat

Clear Selected Results (All results if none selected) Export to XML

Invalid Unicode ByteStream

Risk: High

An invalid UTF-8 ByteStream was detected for request:
www.nottrusted.com/watcher/CheckPasvUnicode.php

The following issue(s) were identified:

- 1) An invalid 2 character UTF-8 ByteStream was found at byte position 481
Invalid byte(s): C3 75

Tools

Watcher - Web-app Security Testing and Auditing

<http://websecuritytool.codeplex.com>

Tools

Unibomber – runtime XSS testing tool

- Deterministic testing
- Auto-inject payloads
- Unicode transformers
 - < > ‘ “, etc.
- Detect transformations and encoding hotspots

Thank you!



Casaba Security

www.casabasecurity.com

Chris Weber

Blog: www.lookout.net

Email: chris@casabasecurity.com

LinkedIn: <http://www.linkedin.com/in/chrisweber>