x-up-devcap-post-charset Header in ASP.NET to Bypass WAFs Again!

x-up-devcap-post-charset Header in ASP.NET to Bypass WAFs Again! - Soroush Dalili, soroush.me.

- Published: date not stated
- Original: <https://soroush.me/blog/x-up-devcap-post-charset-header-in-aspnet-to-bypass-wafs-again>
- Preserved from: https://soroush.me/blog/x-up-devcap-post-charset-header-in-aspnet-to-bypass-wafs-again (stored) on 2026-08-11
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

x-up-devcap-post-charset Header in ASP.NET to Bypass WAFs Again!

In the past, I showed how the request encoding technique can be abused to bypass web application firewalls (WAFs). The generic WAF solution to stop this technique has been implemented by only allowing whitelisted `charset` via the `Content-Type` header or by blocking certain encoding charsets. Although WAF protection mechanisms can normally be bypassed by changing the headers slightly, I have also found a new header in ASP.NET that can hold the `charset` value which should bypass any existing protection mechanism using the `Content-Type` header.

Let me introduce to you, the one and only, the `x-up-devcap-post-charset` header that can be used like this:

Copy

```
1POST /test/a.aspx?%C8%85%93%93%96%E6%96%99%93%84= HTTP/1.1

2Host: target

3User-Agent: UP foobar

4Content-Type: application/x-www-form-urlencoded

5x-up-devcap-post-charset: ibm500

6Content-Length: 40

7

8%89%95%97%A4%A3%F1=%A7%A7%A7%A7%A7%A7%A7
```

As it is shown above, the `Content-Type` header does not have the `charset` directive and the `x-up-devcap-post-charset` header holds the encoding's charset instead. In order to tell ASP.NET to use this new header, the `User-Agent` header should start with `UP`!

The parameters in the above request were create by the Burp Suite HTTP Smuggler, and this request is equal to:

Copy

```
1POST /testme87/a.aspx?HelloWorld= HTTP/1.1

2Host: target

3User-Agent: UP foobar

4Content-Type: application/x-www-form-urlencoded

5Content-Length: 14

6

7input1=xxxxxxx
```

I found this header whilst I was looking for something else inside the ASP.NET Framework. Here is how ASP.NET reads the content encoding before it looks at the `charset` directive in the `Content-Type` header:

https://github.com/Microsoft/referencesource/blob/3b1eaf5203992df69de44c783a3eda37d3d4cd10/System/net/System/Net/HttpListenerRequest.cs#L362

Copy

```
1if (UserAgent!=null && CultureInfo.InvariantCulture.CompareInfo.IsPrefix(UserAgent, "UP")) {
2    string postDataCharset = Headers["x-up-devcap-post-charset"];
3    if (postDataCharset!=null && postDataCharset.Length>0) {
4        try {
5            return Encoding.GetEncoding(postDataCharset);
```

Or

https://github.com/Microsoft/referencesource/blob/08b84d13e81cfdbd769a557b368539aac6a9cb30/System.Web/HttpRequest.cs#L905

Copy

```
1if (UserAgent != null && CultureInfo.InvariantCulture.CompareInfo.IsPrefix(UserAgent, "UP")) {
2    String postDataCharset = Headers["x-up-devcap-post-charset"];
3    if (!String.IsNullOrEmpty(postDataCharset)) {
4        try {
5            return Encoding.GetEncoding(postDataCharset);
```

I should admit that the original technique still works on most of the WAFs out there as they have not taken the request encoding bypass technique seriously ;) However, the OWASP ModSecurity Core Rule Set (CRS) quickly created a simple rule for it at the time which they are going to improve in the future. Therefore, I disclosed this new header to Christian Folini (@ChrFolini) from CRS to create another useful rule before releasing this blog post. The pull request for the new rule is pending at https://github.com/SpiderLabs/owasp-modsecurity-crs/pull/1392.

*References: *https://soroush.me/downloadable/request-encoding-to-bypass-web-application-firewalls.pdf https://www.slideshare.net/SoroushDalili/waf-bypass-techniques-using-http-standard-and-web-servers-behaviour https://soroush.secproject.com/blog/2018/08/waf-bypass-techniques-using-http-standard-and-web-servers-behaviour/ https://soroush.me/downloadable/Rare_ASP.NET_R

equest_Validation_Bypass_Using_Request_Encoding.pdf https://github.com/nccgroup/BurpSuiteHTTPSmuggler/
