CVE-2020-28919: Stored XSS in Checkmk 1.6.0p18

19/12/2021

I've discovered a trivial stored XSS vulnerability in Checkmk 1.6.0p18 during an on-site penetration test and disclosed it responsibly to the tribe29 GmbH. The vendor promptly confirmed the issue, fixed it and announced an advisory. I've applied for a CVE, but didn't get around explaining the vulnerability in detail, therefore I'm publishing this blog post to complete the process.

Summary

- CVSS Score: 5.4 (Medium)
- CVSS: `CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N`
- Affected versions: 1.6.0p18 and earlier
- Fixed version: 1.6.0p19, 2.0.0i1
- Vendor advisory: Werk #11501
- Affected component: cmk/gui/htmllib.py:Escaper
- Bug fix: 87ceb966, e7fd8e4c

Impact

The vulnerability requires an authenticated attacker with permission to configure and share a custom view. Given these prerequisites, they can inject arbitrary JavaScript into the view title by inserting a HTML link with a JavaScript URL. If the attacker manages to trick a user into clicking that link, the JavaScript URL is executed within the user's browser context.

There is a CSP policy in place, but it does not mitigate inline JavaScript code in event handlers, links or script tags. An attacker could therefore obtain confidential user data or perform UI redressing.

The vulnerable code has been identified in versions below 1.6.0p18, such as 1.6.0 and older. It is unclear in which version the vulnerability has been introduced, therefore it's recommended to update to 1.6.0p19/2.0.0i1 or newer.

Detailed description

The Checkmk GUI code uses a WordPress-style approach to handle HTML: User input is encoded using HTML entities, then selectively decoded with a regular expression looking for simple tags. As a special case, the `<a>` tag gets its `href` attribute unescaped as well to enable hyperlinks. The attribute is not checked for its protocol, thereby allowing URLs such as `javascript:alert(1)`.

```
class Escaper(object):
    def __init__(self):
        super(Escaper, self).__init__()
        self._unescaper_text = re.compile(
            r'&lt;(/?)(h1|h2|b|tt|i|u|br(?: /)?|nobr(?: /)?|pre|a|sup|p|li|ul|ol)&gt;')
        self._quote = re.compile(r"(?:&quot;|&#x27;)")
        self._a_href = re.compile(r'&lt;a href=((?:&quot;|&#x27;).*?(?:&quot;|&#x27;))&gt;')

    [...]
    def escape_text(self, text):
        if isinstance(text, HTML):
            return "%s" % text  # This is HTML code which must not be escaped

        text = self.escape_attribute(text)
        text = self._unescaper_text.sub(r'<\1\2>', text)
        for a_href in self._a_href.finditer(text):
            text = text.replace(a_href.group(0),
                                "<a href=%s>" % self._quote.sub("\"", a_href.group(1)))
        return text.replace("&amp;nbsp;", "&nbsp;")
```

The above code is used for HTML generation. To exploit it, I started looking for a HTML form and found that when editing a custom view, no user input validation is performed on the view title (as opposed to the view name).

```
def page_edit_visual(what,
                     all_visuals,
                     custom_field_handler=None,
                     create_handler=None,
                     load_handler=None,
                     info_handler=None,
                     sub_pages=None):
    [...]
    html.header(title)
    html.begin_context_buttons()
    back_url = html.get_url_input("back", "edit_%s.py" % what)
    html.context_button(_("Back"), back_url, "back")
    [...]
    vs_general = Dictionary(
        title=_("General Properties"),
        render='form',
        optional_keys=None,
        elements=[
            single_infos_spec(single_infos),
            ('name',
             TextAscii(
                 title=_('Unique ID'),
                 help=_("The ID will be used in URLs that point to a view, e.g. "
                        "<tt>view.py?view_name=<b>myview</b></tt>. It will also be used "
                        "internally for identifying a view. You can create several views "
                        "with the same title but only one per view name. If you create a "
                        "view that has the same view name as a builtin view, then your "
```

```
                    "view will override that (shadowing it)."),
                regex='^[a-zA-Z0-9_]+$',
                regex_error=_(
                    'The name of the view may only contain letters, digits and underscores.'),
                size=50,
                allow_empty=False)),
            ('title', TextUnicode(title=_('Title') + '<sup>*</sup>', size=50, allow_empty=False)),
            [...]
        ],
    )
    [...]
```

Fix

Checkmk 1.6.0p19 and 2.0.0i1 parses the URL and validates its scheme against an allowlist before unescaping. JavaScript URLs are therefore left unescaped and not made clickable:

```
def escape_text(self, text):
    if isinstance(text, HTML):
        return "%s" % text  # This is HTML code which must not be escaped

    text = self.escape_attribute(text)
    text = self._unescaper_text.sub(r'<\1\2>', text)
    for a_href in self._a_href.finditer(text):
        href = a_href.group(1)

        parsed = urlparse.urlparse(href)
        if parsed.scheme != "" and parsed.scheme not in ["http", "https"]:
            continue  # Do not unescape links containing disallowed URLs

        target = a_href.group(2)

        if target:
            unescaped_tag = "<a href=\"%s\" target=\"%s\">" % (href, target)
        else:
            unescaped_tag = "<a href=\"%s\">" % href

        text = text.replace(a_href.group(0), unescaped_tag)
    return text.replace("&amp;nbsp;", "&nbsp;")
```

Timeline

- 2020-10-11: Initial contact with vendor
- 2020-10-12 - 2020-10-14: Further clarification with vendor
- 2020-10-20: Vendor advisory Werk #11501 has been released
- 2020-10-26: Vendor notified me about a patch for Checkmk 1.6.0p19
- 2020-11-17: Applied for CVE
- 2020-11-18: Received CVE-2020-28919
- 2021-12-19: Released blog post
- 2022-01-15: NVD published