

# CKAN Authenticated SSRF <= 2.9.11/2.10.4

## Vulnerability Information

Product: **Ckan**

Vendor: <https://github.com/ckan>

Affected Version(s): <= **2.9.11/2.10.4**

CVE ID: **TBD**

Description: **SSRF vulnerability in resource proxy functionality in Ckan <=2.9.11/2.10.4, allowing authenticated attackers to scan internal ports/hosts, and map the infrastructure environment.**

Vulnerability Type: **Server Side Request Forgery**

Root Cause: **User supplied property is not sanitized against common SSRF payloads when specifying the URL of external resources.**

Impact: **An authenticated attacker can scan ports/hosts of the internal network, and map the infrastructure environment. At the time of discovery, there were about 1000 instances on the Internet.**



## Reproduction Steps

1. Use grep to search potential vulnerable code:

```
(root@kali)-[~/Desktop/ckan]
# grep -iR "requests.get(" --include=*.py
ckan/model/license.py:         response = requests.get(license_url, timeout=timeout)
ckan/lib/search/__init__.py:     response = requests.get(
ckan/lib/search/__init__.py:     response = requests.get(url, timeout=timeout)
ckan/lib/captcha.py:     response = requests.get(recaptcha_server_name, params, timeout=timeout)
ckanext/resourceproxy/tests/test_proxy.py:         result = requests.get(url, timeout=30)
ckanext/resourceproxy/tests/test_proxy.py:         result = requests.get(url, timeout=30)
ckanext/resourceproxy/tests/test_proxy.py:         requests.get(url, timeout=1)
ckanext/resourceproxy/blueprint.py: r = requests.get(url, timeout=timeout, stream=True)
ckanext/resourceproxy/blueprint.py: r = requests.get(
ckanext/datapusher/logic/action.py: r = requests.get(url,
```

2. Take a closer look into the code:

```
<...SNIP...>
resource_id = data_dict[u'resource_id']
log.info(u'Proxy resource {id}'.format(id=resource_id))
try:
    resource = get_action(u'resource_show')(context, {u'id': resource_id})
except logic.NotFound:
    return abort(404, _(u'Resource not found'))
url = resource[u'url']

parts = urlsplit(url)
if not parts.scheme or not parts.netloc:
    return abort(409, _(u'Invalid URL.'))

timeout = config.get('ckan.resource_proxy.timeout')
```

```
max_file_size = config.get(u'ckan.resource_proxy.max_file_size')
response = make_response()
try:
    did_get = False
    r = requests.head(url, timeout=timeout)
    if r.status_code in (400, 403, 405):
        r = requests.get(url, timeout=timeout, stream=True)
<...SNIP...>
```

url is a user supplied property, and no input sanitization are employed.

3. To exploit the vulnerability, resource proxy plugin should be enabled:

<https://docs.ckan.org/en/2.9/maintaining/data-viewer.html#resource-proxy>

4. The vulnerability requires authentication, and the user should have specific permissions.

5. Add a view for a resource, specify the above internal URL.

image.png

image.png

6. Access the view, we can see hit logs. Attacker induces the server to make a request on his behalf.

```
(root@kali) - [~/Desktop]
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.1.64 - - [12/Jul/2024 17:44:35] code 404, message File not found
192.168.1.64 - - [12/Jul/2024 17:44:35] "GET /ssrf_pwn HTTP/1.1" 404 -
```

7. Stop the HTTP listener, and switch to a TCP listener.

image.png

8. Preview the view again, and the listener captures the access log against.

image.png

Interestingly, since it is a non-http port, the preview keeps loading.

image.png

The difference in response time can indicate whether a port is open, and whether the port is a http/https port. In this way, attackers can weaponize this vulnerability to scan internal network's hosts and ports.

---

Revision #2

Created 2 August 2024 18:47:09 by winslow

Updated 2 August 2024 23:02:26 by winslow