Overview of vulnerability CVE-2014-0006

Computer Security (MTAT.03.134)

Joosep Järve

OOS (Swift)
OpenStack Object Storage (Swift) is a widely-used open-source object storage system that can be used as a stand-alone system or as a part of a cloud compute environment. It runs on standard Linux distributions and on standard x86 server hardware. It has eventual consistency architecture, which makes it ideal for building massive highly distributed infrastructures with lots of unstructured data serving global sites. It provides object storage in virtual containers, which allows users to store and retrieve files (arbitrary data) in applications via an industry-standard RESTful http API, by HTTP requests directly to the API or by using one of many client libraries in languages as Java, Python, Ruby or JavaScript. The objects can have extensive indexable and searchable metadata. The service's distributed architecture supports horizontal scaling and redundancy as failure-proofing is provided through software-based data replication.

The vulnerability
On the 2nd of January, 2014 a SwiftStack (a commercial private cloud application using OpenStack Swift) employee Samuel Merritt reported and suggested a fix for a timing attack flaw in the way the swift TempURL middleware responded to arbitrary TempURL requests. TempURL compares a user-submitted string (HMAC digest) against a list of valid strings to determine whether or not to allow access to an object. The string comparison uses Python's built-in string comparison, which short-circuits evaluation on the first differing character, which may let an attacker perform timing analysis. Only Swift setups enabling the TempURL middleware are affected.

An attacker with knowledge of the targeted object name and the object account could use this flaw to obtain a secret URL to this object, which was intended to be publicly shared only with specific recipients, if the object had the TempURL key set.

The vulnerability was introduced along with the TempURL middleware in Swift version 1.4.6, which was released in February 2012 and affects versions 1.4.6 through 1.8.0, 1.9.0 through 1.10.0, and 1.11.0.
The vulnerability has received a CVSS v2 Base score of 4.3.

The exploitability subscore was rated at 8.6.

Exploitability metrics:

Access Vector: Network; The vulnerability is bound to the network stack and the attacker does not require local network access or local access.

Access Complexity: Medium; The access conditions are somewhat specialized: The attacking party is limited to a group of systems or users at some level of authorization, possibly untrusted; Some information must be gathered before a successful attack can be launched; The affected configuration is non-default and is not commonly configured; The attack requires a small amount of social engineering.

Authentication: None; Authentication is not required to access and exploit the vulnerability.

The impact subscore was rated at 2.9. Impact metrics:

Confidentiality Impact: Partial; There is considerable informational disclosure. Access to some system files is possible, but the attacker does not have control over what is obtained, or the scope of the loss is constrained.

Integrity Impact: None; There is no impact to the integrity of the system.

Availability Impact: None; There is no impact to the availability of the system.

Temporal and environmental score metrics were not defined.

**The fix**
The patch to fix the vulnerability was to use a constant time comparison function (streq_const_time) from swift.common.utils instead of Python’s built-in comparison when evaluating TempURL.
swift/common/middleware/tempurl.py

```python
from swift.proxy.controllers.base import get_account_info
from swift.common.swob import HeaderKeyDict, HTTPUnauthorized
from swift.common.utils import split_path, get_valid_utf8_str,

- registerSwiftInfo, getHmac
+
+ registerSwiftInfo, getHmac, streq_const_time

#: Default headers to remove from incoming requests. Simply a whitespace
@
@@ -284,7 +284,13 @
def _call__(self, env, start_response):
                   request_method='PUT'))
     else:
         hmac_vals = self.get_hmacs(env, temp_url_expires, keys)
- if temp_url_sig not in hmac_vals:
+ # while it's true that any() will short-circuit, this doesn't affect
+ # the timing-attack resistance since the only way this will
+ # short-circuit is when a valid signature is passed in.
+ if not is_valid_hmac:
+     return self._invalid(env, start_response)
+     self._cleanIncomingHeaders(env)
+     env['swift.authorize'] = lambda req: None
```

Source: https://github.com/openstack/swift/commit/75463988931e4095530f6b13389c254096eb485, screenshot taken on 31.05.2014

References
https://swiftstack.com/openstack-swift/architecture/
https://review.openstack.org/#/c/67185/
https://bugs.launchpad.net/swift/+bug/1265665
https://github.com/openstack/swift/commit/75463988931e4095530f6b13389c254096eb485