A Code Execution Vulnerability in Google App Engine SDK for Python: A Security Advisory

Adi Sharabani, IBM Security Strategist and Architect
E-mail: `adish@il.ibm.com`, Twitter: `@adisharabani`, Blog: `blog.watchfire.com`

October 11, 2011

Introduction

Google App Engine [1] is a great technology allowing web developers to develop their own web applications, test them in their internal framework, and deploy them to Google's appspot.com domain. The Google App Engine framework allows developers to write their web site logic in Python, and offers several frameworks specially created for this. In addition, Google App Engine provides an SDK Console via web that acts as an administration console for the newly written application. This advisory lists 4 different vulnerabilities, one in admin console and three others in the Google python API, which allow a remote attacker to gain full code execution on the developer's machine. These severe issues have been communicated to Google, and a fix was released last month on Sep 12, 2012 (in version 1.5.4).

CSRF on the Admin Console

Each App Engine application has an internal SDK Console deployed as a web interface (e.g. `http://localhost:8080/_ah/admin`), and this interface can interact with the development server. The SDK Console has many different CSRF [2] vulnerabilities, allowing a remote hacker to redirect the victim's browser to this web based console and perform actions on behalf of the victim. For example, an important pages that is vulnerable to CSRF attack is the "Interactive Console" (`http://localhost:8080/_ah/admin/interactive`) which allows the execution of code by the web server itself, as if it was written as a web application. An attacker could thus lure a victim to enter his/her website, while running Google App Engine, and redirect the victim's browser to the vulnerable CSRF page, executing any python script on the victim's machine.

The following snippet presents malicious HTML code that could exists on attacker's site, and redirect victim's browser to execute the python command on the victim's machine.

```
<form method="POST" action="http://localhost:8080/_ah/admin/interactive/execute">
<textarea id="code" name="code">
import os
print "Page is vulnerable to CSRF"
print "Server's environment variables:",os.environ
</textarea>
</form>
<script>document.forms[0].submit()</script>
```

The following sections will demonstrate three different vulnerabilities in these restriction mechanisms which allow an attacker to gain remote code execution privileges. Combining any of these three vulnerabilities with the CSRF vulnerability results in full remote code execution on machines running the local Google App Engine server.

Command execution due to improper security restrictions in FakeFile object

One of the restrictions Google has applied to control file open actions within the development app server has been implemented using a FakeFile object. It seems that during initialization this object is created, and Google uses it to control which open modes (e.g. read, write, etc.) are allowed and which directories are valid for those actions. By importing the dev_appserver object, and rewriting the ALLOWED_MODES and ALLOWED_DIRS of this FakeFile object, an attacker could create and override any file on the system. Creating files in Start > Program files > Startup or overriding autoexec.bat could allow an attacker to execute arbitrary code on a victim's machine. Obviously similar concepts could apply to other operation systems if user has sufficient privileges.

The following snippet presents sample code that would create a file on the server's machine:

```
from google.appengine.tools import dev_appserver
dev_appserver.FakeFile.ALLOWED_MODES = frozenset(['r', 'rb', 'U', 'rU', 'w'])
dev_appserver.FakeFile.ALLOWED_DIRS.add("C:/")
open("C:/hacked","w").write("You've been hacked.\n")
```

Combined with the CSRF vulnerability mentioned above, a remote attacker could cause a victim to send a request to its vulnerable Google App Engine web interface, and then execute any arbitrary code on the victim's machine.

Command execution using direct access to _original_os object

Google App Engine's framework restricts the use of os.popen to execute commands on its developer application server. This is done by overriding the os module with a wrapper that doesn't have the popen method. However, for internal reasons, Google App Engine's developer decided to keep a reference to the original os module. Leveraging this reference, an attacker could execute the popen command and gain full control over the victim's machine.

The following code snippet presents sample code that would open the calc application.

```
from google.appengine.tools import dev_appserver
os = dev_appserver.RestrictedPathFunction._original_os
os.environ["COMSPEC"]= "C:\\WINDOWS\\system32\\cmd.exe"
os.popen("c:\\windows\\system32\\calc.exe")
```

Command execution using google.appengine.api.blobstore.os object

Similar to the previously described vulnerability, an attacker could access the original os module, as it was saved as a part of other objects in Google's App Engine API. The importance of these problems is that they might exist on the appspot.com server itself, and not just on the developer webapp server deployed with Google App Engine (Read the conclusion of this article for more on this issue).

The following code snippet presents sample code that would open the calc application on the victim's machine,

```
from google.appengine.api.blobstore import file_blob_storage
os = file_blob_storage.os
os.environ["COMSPEC"]= "C:\\WINDOWS\\system32\\cmd.exe"
os.popen("c:\\windows\\system32\\calc.exe")
```

Conclusion

The aforementioned vulnerabilities allow a remote hacker to gain full code execution privileges over users running non-patched Google's App Engine on their machine. The obvious attack vector described in this advisory is to leverage a CSRF vulnerability in the Interactive Console page to execute this code.

Vulnerability Scoring

The CSRF issue which facilitates the code execution attack was logged under CVE-2011-1364 [3]. The CVSS base score given to this issue is 9.3, and temporal score of 8.1.[1]

Vendor Response

As always, Google was very quick and efficient with providing a fix for its users. Google has indicated that Google App Engine v1.5.4, which was release on Sep 12, 2012, contains the fix for the above issues.

Acknowledgments

We would like to thank the Google team for the efficient and quick way in which they handled such security issues.

References

[1] Google App Engine. `http://code.google.com/appengine/`.

[2] CSRF: Cross-Site Request Forgery vulnerability. `http://en.wikipedia.org/wiki/Cross-site_request_forgery`.

[3] CVE-2011-1364. `http://xforce.iss.net/xforce/xfdb/69958`.

[1]CVSS Vector used for this calcualtion is: AV:N/AC:M/Au:N/C:C/I:C/A:C/E:H/RL:OF/RC:C