CVE: CVE-2011-3224

Found By: Aaron Sigel of vtty.com and Brian Mastenbrook

Note:

We have attempted to fully explain how these bugs work below. Since we expect that's too remedial for some of you, here's the summary…

1. Help files from the Mac App Store contain AppleScript and Python payloads that can be MITMed during autoupdate resulting in execution of arbitrary commands for a remote attacker
2. When updating help, the Mac App Store insecurely writes and accesses locations in "/tmp/" with guessable filenames, which could result in local cross-user attacks

Affected Software:

Mac OS X v10.6.*
Previous versions may be affected but were not tested.

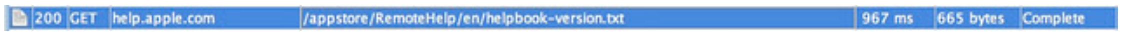
The details:

Man-in-the-middle (MITM) bugs are well known to security researchers and often lead to information disclosure that can result in session hijacking or leaking personal information. MITM attacks that result in execution of arbitrary commands on a victim's computer seem less common. This was not always the case. It used to be fairly common to see applications that had built in update mechanisms not bothering to use any secure method to grab new code. These days it is expected that new code will be validated with some form of cryptographic mechanism or be provided to the user over a secure channel before being executed. Without these safeguards in place it would be possible for attackers to hijack the update unless you had a fully trusted network connection between your computer and the vendor. Since that is not the case most of the time, most reputable software vendors implement these mechanisms to protect their updates. Unfortunately, this is not always done on Mac OS X when help books are updated.

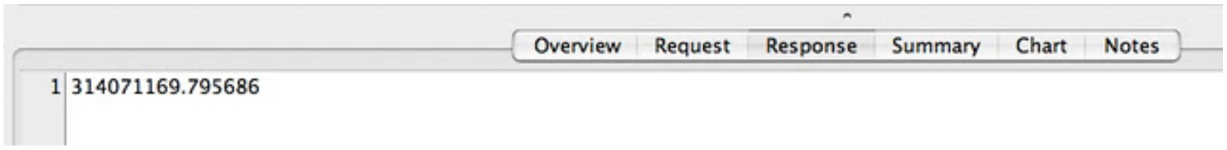
When the Mac App Store help book is opened, the help subsystem attempts to make sure that the documentation being displayed is the latest and greatest available. This is accomplished by a Python script that is distributed as part of the help book. Here's a snippet from the script:

```
# get the version number from the server
serverVersionURL = serverBaseURL + "helpbook-version.txt"
serverVersion =
NSString.stringWithContentsOfURL_encoding_error_
(NSURL.URLWithString_(serverVersionURL),
NSUTF8StringEncoding, None)
serverVersion = serverVersion[0]

# get the local version number
localVersionURL = directoryPath + "helpbook-version.txt"
localVersion =
NSString.stringWithContentsOfFile_encoding_error_
(localVersionURL, NSUTF8StringEncoding, None)
localVersion = localVersion[0]

# show the help if we do have the latest help
if serverVersion == localVersion:
        with open(statusFilePath, 'w') as statusFile:
                statusFile.write("NO_UPDATE_AVAILABLE")
        exit()
```

As you can see, if the version of the help document on the server is not the same as the local version, the update script will believe an update is available. Note that the help book version file is not signed in any way:

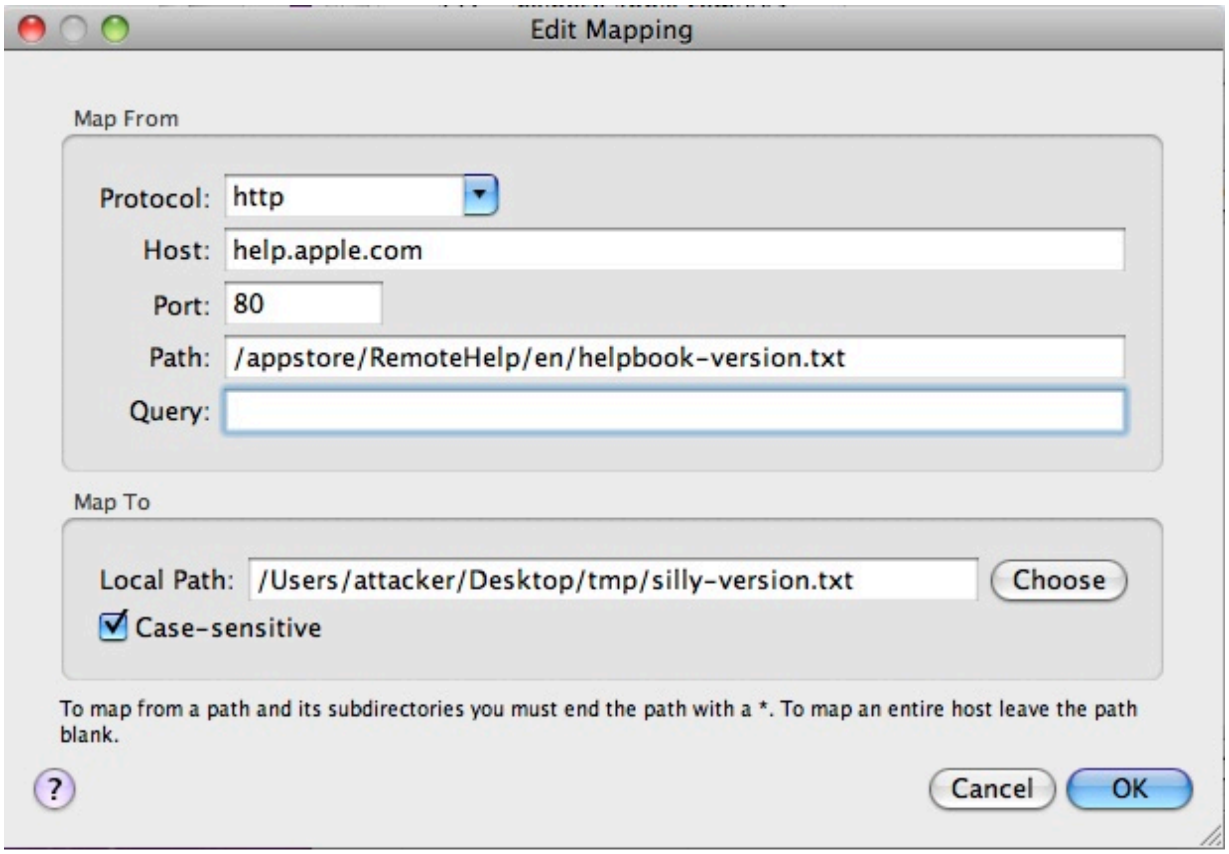


```
200  GET  help.apple.com  /appstore/RemoteHelp/en/helpbook-version.txt  967 ms  665 bytes  Complete
```



```
314071169.795686
```

By intercepting this request and serving a different version string, it is possible to trigger an auto update. This was tested using Charles Proxy to simulate the MITM in action by mapping the help book version request to serve a local file with a different version in it:


Edit Mapping

Map From

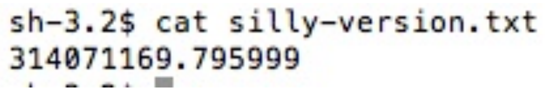
Protocol: http
Host: help.apple.com
Port: 80
Path: /appstore/RemoteHelp/en/helpbook-version.txt
Query:

Map To

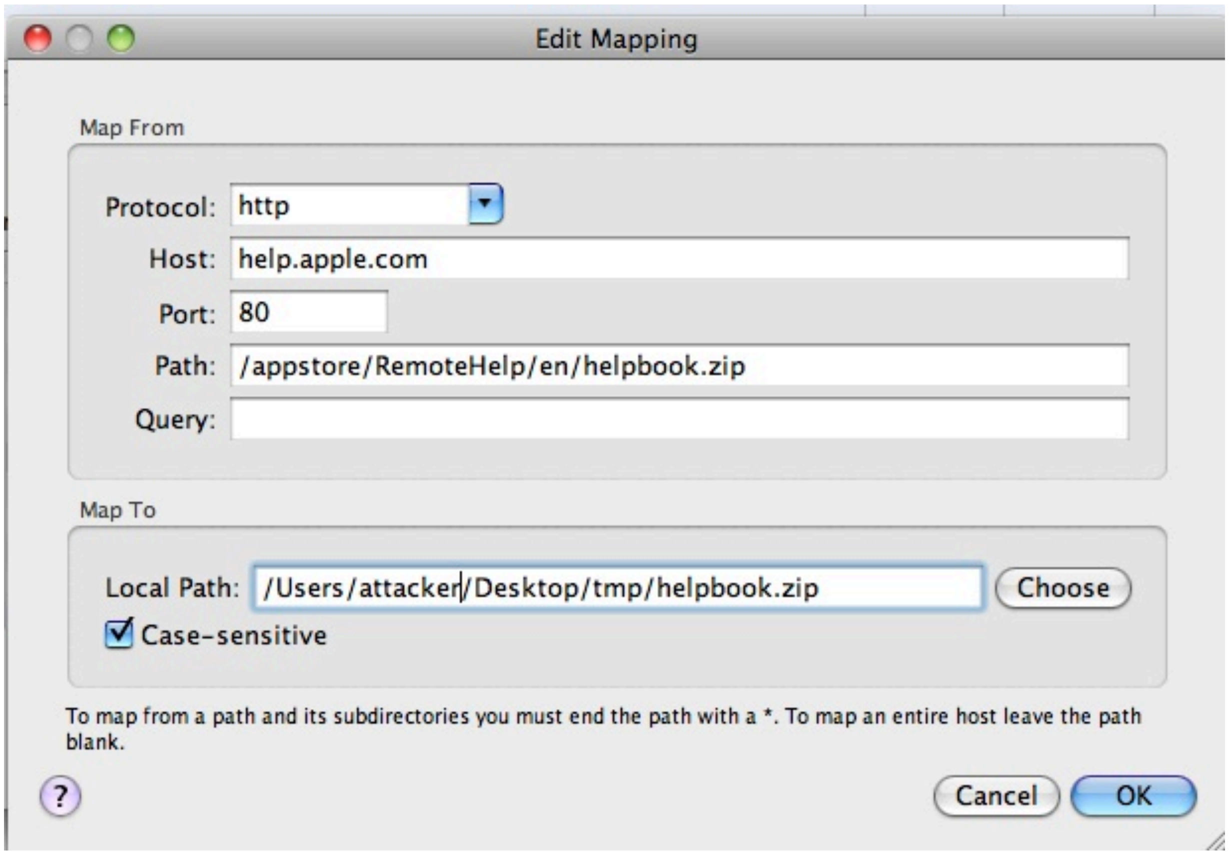
Local Path: /Users/attacker/Desktop/tmp/silly-version.txt
☑ Case-sensitive

To map from a path and its subdirectories you must end the path with a *. To map an entire host leave the path blank.

Content of "silly-version.txt":


```
sh-3.2$ cat silly-version.txt
314071169.795999
```

After passing this version test, the Python script proceeds to request a new help book archive and installs it. The help book archive "helpbook.zip" is downloaded from the remote server and installed. This new archive contains a full copy of the update script that has been performing this update. Next time an update occurs, our versions of the scripts will be executed. Charles Proxy was also used to demonstrate this part of the attack:



Edit Mapping

Map From

Protocol: http

Host: help.apple.com

Port: 80

Path: /appstore/RemoteHelp/en/helpbook.zip

Query:

Map To

Local Path: /Users/attacker/Desktop/tmp/helpbook.zip

☑ Case-sensitive

To map from a path and its subdirectories you must end the path with a *. To map an entire host leave the path blank.

In this version of "helpbook.zip", the "scripts/updatefrontend.py" script has been modified as follows:

```
#! /usr/bin/python

import objc, os, sys
from Foundation import *

+ os.system("open /Applications/Chess.app; /usr/bin/
touch /var/tmp/.HelpUpdateRan")
```

At this point we can see that the help subsystem will install malicious code provided by an attacker. The example above just runs Chess.app and creates a file to demonstrate that unsigned scripts have been executed. As demonstrated below, this is executed the next time that the help book is loaded. Of course it may take a while to happen unless we combine this attack with the "help:" URL scheme, which Safari will launch without any user interaction. Other good targets for attackers to target are "js/javascript.js" and "scripts/updatefrontend.scpt".

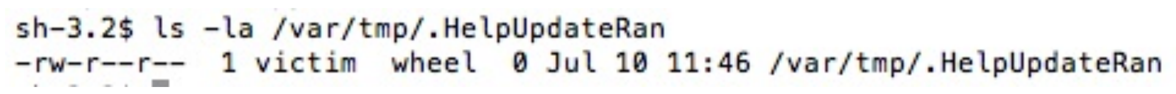
Here's the source of the test page used in the screen capture.

```
> cat ladieslove.html
<html>
<body>
<a href="help:openbook=com.apple.AppStore.help">chest
rockwell</a>
</body>
</html>
```

To see all of this in action, play the video below, which shows what happens once Mac App Store help is launched through Safari, triggering the injected Python commands to run:

There was a video here, which may still be live at:
http://vttynotes.blogspot.com/2011/10/cve-2011-3224-mitm-to-rce-with-mac-app.html

Also, the file in "/var/tmp" was created, owned by the victim:



```
sh-3.2$ ls -la /var/tmp/.HelpUpdateRan
-rw-r--r--  1 victim  wheel  0 Jul 10 11:46 /var/tmp/.HelpUpdateRan
```

Constraints and notes about this vulnerability:

1. Help book documents are only accessible via URL after they are registered, which occurs after being opened once. In order for the demonstration above to work, the help book from the Mac App Store must have been opened at least once.

2. This bug is an issue for more applications than just Mac App Store. It is an issue for any application that insecurely transfers help book content or does not validate it in some way before it is executed. Finding the other applications distributed with Mac OS X as part of the base distribution vulnerable to this issue is a (simple) exercise for the reader. An attacker could trigger several help books at once with the hope of infecting just one.

3. This demonstration was noisy and obvious, but an actual attacker could be much sneakier.

Local user attack:

Aside from the issue shown above, there's another far less serious security hole in the update process involving a file in "/tmp" that is insecurely created. When help documentation is checked and updated, the following is done:

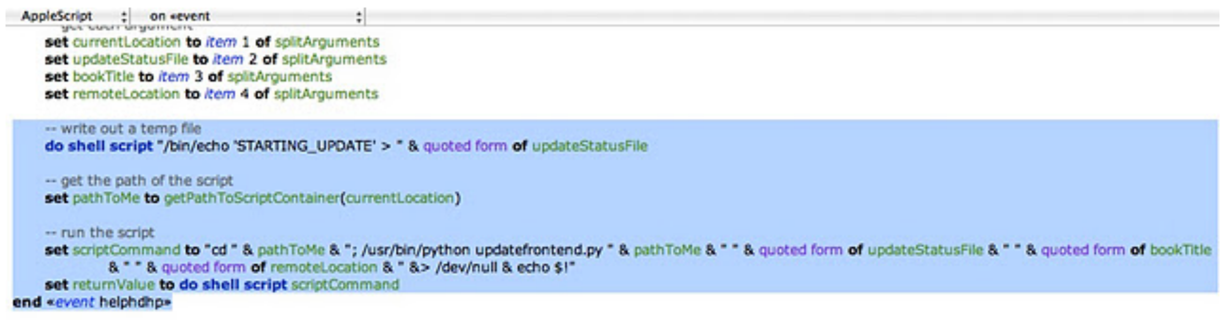
1. js/javascript.js creates an update filename:

```
 update_status_file:"/tmp/apd"+(new Date()).getTime()+"-update-status.txt"
```

2. This filename is then passed to the update scripts, which write to this file during update.

```
 setTimeout(function(){var d="help:runscript=/scripts/updatefrontend.scpt";d+="
string='"+location.href+",,,";d+=updateController.update_status_file+",,,";d
+=dataController.getSettingsStringForKey("FolderName")+",,,";d
+=dataController.getSettingsStringForKey("RemoteURL");d
+=localizationController.language+"/'";location=d; ….
```

This is vulnerable to traditional "/tmp" attacks, such as symbolic-linking the update status file to something the victim can write, and the attacker wants to clobber, create, or fill with a known value. Here's an example of where the Apple Script included in the help book insecurely writes to this file:



```
    set currentLocation to item 1 of splitArguments
    set updateStatusFile to item 2 of splitArguments
    set bookTitle to item 3 of splitArguments
    set remoteLocation to item 4 of splitArguments

    -- write out a temp file
    do shell script "/bin/echo 'STARTING_UPDATE' > " & quoted form of updateStatusFile

    -- get the path of the script
    set pathToMe to getPathToScriptContainer(currentLocation)

    -- run the script
    set scriptCommand to "cd " & pathToMe & "; /usr/bin/python updatefrontend.py " & pathToMe & " " & quoted form of updateStatusFile & " " & quoted form of bookTitle
        & " " & quoted form of remoteLocation & " &> /dev/null & echo $!"
    set returnValue to do shell script scriptCommand
end «event helphdhp»
```

Fix:

Apple has addressed these issues in Security Update 2011-006. It can be installed via Software Update.

Conclusion:

1. Apple should have signed and validated this code, or at the very least made sure that the help book was sent over a secure channel. Apple should scan all other help books they provide for variants of these issues. Apple should also reach out to third-party developers to help them avoid these same mistakes.

2. Install the security update that addresses this issue.

Reference:

1. Apple's advisory:

http://support.apple.com/kb/HT5002

2. Brian Mastenbrook's previous Help Viewer security issues:

http://brian.mastenbrook.net/display/30