ΕΡΓΑΣΙΑ ΣΤΑ ΠΛΑΙΣΙΑ ΤΟΥ ΜΑΘΗΜΑΤΟΣ: ΑΣΦΑΛΕΙΑ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

**Project #1**

****

**ΥΣ13 ΕΑΡΙΝΟ 2017**

σΠΥΡΟΣ ΑΝΑΣΤΑΣΙΟΥ 1115201300007

ΑΓΓΕΛΟΣ ΤΟΥΜΑΣΗΣ 1115201300177

**ΠΕΡΙΕΧΟΜΕΝΑ**

* **ΠΡΟΛΟΓΟΣ**
* **ΠΡΟΣΤΑΣΙΑ** 
  + *SQL Injection*
  + *Cross-site Scripting (XSS)*
  + *Cross-Site Request Forgery (CSRF)*
  + *Remote File Injection (RFI)*
* **ΕΠΙΘΕΣΗ** 
  + *SQL Injection*
  + *Cross-site Scripting (XSS)*
  + *Cross-Site Request Forgery (CSRF)*
  + *Remote File Injection (RFI)*
* **ΣΥΜΠΕΡΑΣΜΑΤΑ**
* **ΒΙΒΛΙΟΓΡΑΦΙΑ**

**ΠΡΟΛΟΓΟΣ**

Η εφαρμογή του eclass v2.3 ελέχθηκε για πιθανά προβλήματα ασφάλειας όπως ζητείται στην εκφώνηση. Τα προβλήματα είναι κατηγοριοποιημένα σε SQL Injection, Cross-site Scripting (XSS), Cross-Site Request Forgery (CSRF) και Remote File Injection (RFI). Γνώμονας της διόρθωσης αυτών των κενών ασφάλειας ήταν να παραμείνει η **λειτουργικότητα** της εφαρμογής όπως προσφέρεται.

**ΠΡΟΣΤΑΣΙΑ**

Το χρονικό διάστημα μέχρι τις 21 Απριλίου, αφιερώθηκε στην κατανόηση της λειτουργίας της εφαρμογής και τη μελέτη της προστασίας του μαζί με την επίλυση των προβλημάτων που παρουσιάζουμε παρακάτω.

**SQL Injection**

προβλημάτων που παρουσιάζουμε παρακάτω

**Cross-site Scripting (XSS)**

Ξεκινώντας την εξερεύνηση τέτοιων ευπαθειών δοκιμάσαμε σε φόρμες με πεδία να γράψουμε html script της μορφής: <script> alert(‘ Problem here! ’); </script> , τα οποία στέλναμε, παρατηρώντας κάθε φορά το αποτέλεσμα τους όταν παρουσιάζονταν πίσω τα στοιχεία που δώσαμε. Το παραπάνω πρόκειται για *Persistent XSS*. Αρχεία με το παραπάνω πρόβλημα ήταν ……………………………………………………………………………………

Στη συνέχεια προσπαθήσαμε να προσθέσουμε στις καταλήξεις συνδέσμων με php αρχεία τα script tags και παρατηρήσαμε ότι η εκτενής χρήση της μεταβλητής $\_SERVER["PHP\_SELF"] οδηγούσε στην δυνατότητα ενός χρήστη να προσθέσει στο url κάτι της μορφής /%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E το οποίο θα μεταφραζόταν σε /"><script>alert('hacked')</script> . Το αποτέλεσμα αυτής της ενέργειας θα ήταν να κλείνει σε forms το action κομμάτι και να προστίθεται ένα script που θα έκανε alert το μήνυμα hacked. Ο τρόπος για να αντιμετωπίσουμε αυτό το πρόβλημα σύμφωνα με το <https://www.w3schools.com/php/php_form_validation.asp> είναι η χρήση htmlspecialchars() όποτε και αυτό κάναμε.

Όπως αναφέρθηκε η λανθασμένη χρήση αυτής της μεταβλητής ήταν εκτενής, αλλά η διόρθωση βασικών αρχείων που γίνονταν include σε ακόμα περισσότερα διόρθωσε το πρόβλημα. Σημαντικά αρχεία που διορθώθηκαν ήταν τα:

* modules/dropbox/index.php
* modules/forum\_admin/forum\_admin.php
* include/baseTheme.php
* student\_view.php
* modules/agenda/myagenda.php
* modules/phpbb/ viewtopic
* modules/phpbb/ editpost
* modules/phpbb/index
* modules/phpbb/newtopic
* modules/phpbb/reply
* modules/phpbb/viewforum

Πέρα από τα παραπάνω διορθώθηκαν και τα εξής :

* modules/auth/newprof.php
* modules/auth/formuser.php
* modules/auth/contactprof.php
* modules/auth/courses.php
* modules/admin/eclassconf.php
* modules/admin/mailtoprof.php
* modules/admin/addfaculte.php
* modules/admin/change\_user.php
* modules/admin/adminannouncements.php
* modules/course\_tools/course\_tools.php
* modules/course\_info/delete\_course.php
* modules/unreguser/unreguser.php
* modules/unreguser/unregcours.php
* modules/user/user.php
* modules/profile/profile.php

Εκ των οποίων μερικά έγιναν με τη σκέψη της σωστής χρήσης της μεταβλητής, άσχετα αν οδηγούσαν στην επιτυχή χρήση του script.

Τα παραπάνω αποτελούν δείγμα *Non-Persistent XSS*

Σημαντικό είναι να αναφέρουμε και το τρίτο είδος XSS το *DOM-based XSS*.

………………………………………………………..

Same origin policy kai cookies????

**Cross-Site Request Forgery (CSRF)**

Ιδέα για το παρακάτω πρόβλημα αποτέλεσε ένας χρήστης ή ένας διαχειριστής που θα άνοιγε κάποιο link ή θα πατούσε submit σε κάποια κακόβουλη εφαρμογή το οποίο θα τον οδηγούσε να καταστρέψει ή να διαμορφώσει πράγματα για τον λογαριασμό του ή ακόμα και να τον διαγράψει τελείως.

H χρήση μεταβλητών όπως η $require\_login = true; (ή αντίστοιχα η $require\_admin = true; ), δεν είναι αρκετή αφού αν ο χρήστης είναι συνδεδεμένος στον λογαριασμό του σε άλλη καρτέλα στον ίδιο browser και ανοίξει το link, αυτό θα πετύχει το σκοπό του.

O τρόπος να αποφύγουμε κάτι τέτοιο (σύμφωνα με τις διαφάνειες της 3ης διάλεξης) είναι η χρήση ενός μυστικού (hidden) token που στέλνεται μαζί με την συμπλήρωση ενός form ή με την επιλογή ενός link από το eclass.

Έτσι κάναμε generate ένα token το οποίο στέλναμε μόλις ο χρήστης ήθελε να εκτελέσει κάποια συγκεκριμένη ενέργεια.

Ο τρόπος που έγινε generate το token είναι ως εξής:

$token = md5(uniqid(rand (), TRUE));

$\_SESSION['token'] = $token;

Και στέλνεται ως:

<input type="hidden" name="token" value="$token" />

Δυστυχώς το παραπάνω αντιμετωπίζει προβλήματα αφου:

Το αποτέλεσμα της rand() μπορεί να είναι προβλέψιμο, το uniqid() προσθέτει μόνο μέχρι 29 bits entropy και ο md5 δεν προσθέτει καθόλου entropy αφού λειτουργεί ντετερμινιστικά. Τρόποι να λυθούν αυτά θα ήταν η χρήση νεότερης έκδοσης της php ή εξωτερικές βιβλιοθήκες, όμως θεωρήθηκε ότι ξεφεύγουν από τον στόχο της εργασίας όποτε έγινε η παραδοχή ότι ο επιτιθέμενος δεν θα προσπαθήσει να βρει τον τρόπο με τον οποίο παράγονται τα token ώστε να τον μιμιθεί.

Εν συνεχεία, κατά την εκτέλεση του αιτήματος που γίνεται από τη φόρμα με κάποιο SQL query προσθέσαμε την συνθήκη:

if (isset($\_SESSION['token']) && $\_POST['token']==$\_SESSION['token'])

Με το 2ο μέρος της συνθήκης τσεκάρουμε αν είναι ίδια τα token ώστε να αποφανθούμε αν πρόκειται για CSRF επίθεση. Το πρώτο μέρος υπάρχει στην περίπτωση που ο χρήστης δεν έχει πλοηγηθεί ακόμα στη συγκεκριμένη σελίδα που θα γίνει η επίθεση και έτσι δεν έχει αποκτήσει ακόμα token.

Αρχεία που τροποποιήσαμε είναι τα:

* modules/unreguser/unreguser.php
* modules/unreguser/unregcours.php
* modules/admin/unreguser.php ?????????????? TODO
* modules/admin/delcours.php
* modules/admin/addfaculte.php
* modules/admin/change\_user.php
* modules/admin/adminannouncements.php
* modules/profile/profile.php
* modules/course\_info/infocours.php
* modules/course\_info/delete\_course.php
* modules/course\_info/refresh\_course.php

…………………more TODO

**Remote File Injection (RFI)**

προβλημάτων που παρουσιάζουμε παρακάτω

**Διοίκηση Ολικής Ποιότητας**

Η *Δ.Ο.Π. (Διοίκηση Ολικής Ποιότητας)* σαν νέος τρόπος οργάνωσης των επιχειρήσεων, ξεκίνησε να εφαρµόζεται στην πράξη από το 1949, από την Ένωση Ιαπώνων Επιστηµόνων οι οποίοι είχαν άµεσο στόχο τη βελτίωση της παραγωγικότητας.

Το εν λόγω εγχείρηµα επιχειρήθηκε και στις ΗΠΑ, µε χρονική υστέρηση 30 περίπου

**ΕΠΙΘΕΣΗ**

Κατά τη φάση της επίθεσης (22 Απριλίου μέχρι 28 Απριλίου), η αντίπαλη ομάδα που μας δόθηκε ήταν η « » και ο κύριος στόχος μας ήταν να δούμε πόσες από τις ευπάθειες που αναφέραμε κατάφεραν να εντοπίσουν και να διορθώσουν.

**SQL Injection**

προβλημάτων που παρουσιάζουμε παρακάτω

**Cross-site Scripting (XSS)**

προβλημάτων που παρουσιάζουμε παρακάτω

**Cross-Site Request Forgery (CSRF)**

προβλημάτων που παρουσιάζουμε παρακάτω

**Remote File Injection (RFI)**

προβλημάτων που παρουσιάζουμε παρακάτω

Κατά τη διάρκεια του δεύτερου παγκόσµιου πόλεµου, υπήρξαν ποιοτικά προβλήµατα σε πολλές να υιοθετήσει τις BS 5750 ως διεθνή πρότυπα.Έτσι οι BS 5750 έγιναν ο ISO 9000.

**ΣΥΜΠΕΡΑΣΜΑΤΑ**

Οι παραπάνω έννοιες είναι μείζονος σημασίας για την επιστήμη της Διοίκησης Έργων και ο λόγος είναι τα μοντέλα που αναφέρθηκαν καθώς πραγματοποιούνται σύμφωνα με καθολικά πρότυπα της ΙSO. Καταλαβαίνουμε λοιπόν ότι, ο γενικός στόχος είναι η τήρηση και η βελτίωση των προδιαγραφών (χαρακτηριστικών) των προϊόντων ή υπηρεσιών που προσφέρονται, με τη χρήση των ανάλογων εργαλείων, έτσι ώστε να καλύπτονται οι συνεχώς μεταβαλλόμενες απαιτήσεις των πελατών.

Συμπερασματικά, η παραπάνω εργασία πετυχαίνει το σκοπό της καλύπτοντας τις δοσμένες προϋποθέσεις που έπρεπε να τηρηθούν

**ΒΙΒΛΙΟΓΡΑΦΙΑ**

**Διοίκηση Ολικής Ποιότητας**

Δρ. ΘΕΟΔΩΡΟΣ Κ. ΘΕΟΔΩΡΟΥ ‘Η Διοίκηση Ολικής Ποιότητας (Δ .Ο.Π.) στις

επιχειρήσεις και στην εκπαίδευση

**ISO 9000**

Διαχείρισης Καινοτοµίας’