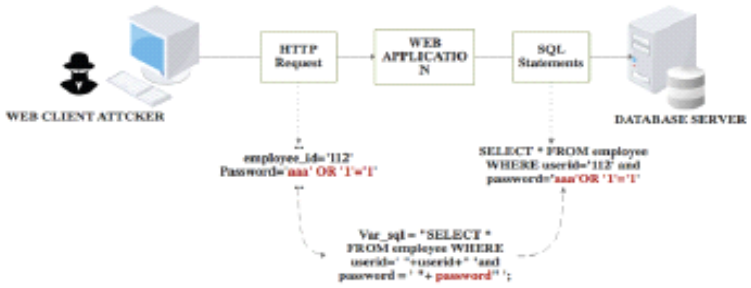
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΤΕΙ ΗΠΕΙΡΟΥ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ:

«Μελέτη και υλοποίηση επίθεσης εισαγωγής SQL ερωτήματος (SQL Injection) σε διαδικτυακές εφαρμογές και παροχή μηχανισμών προστασίας»



Τραγουδάρας Κωνσταντίνος

Α.Μ. : 11691

Επιβλέπων καθηγητής : Λιάγκου Βασιλική

ΣΕΠΤΕΜΒΡΙΟΣ 2018

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, Σεπτέμβριος 2018

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής
Βασιλική Λίαγκου

2. Μέλος επιτροπής

3. Μέλος επιτροπής

Ο Προϊστάμενος του Τμήματος
Στύλιος Χρυσόστομος

Δήλωση μη λογοκλοπής.

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Τραγουδάρας Κωνσταντίνος

Υπογραφή

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ θερμά την κ. Λιάγκου Βασιλική για την εποικοδομητική συνεργασία και καθοδήγηση κατά τη διάρκεια εκπόνησης της εργασίας.

Πίνακας περιεχομένων

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία θα μιλήσουμε για την ασφάλεια των Βάσεων Δεδομένων και θα γνωρίσουμε το SQL Injection.Η παρούσα πτυχιακή ασχολείται με επιθέσεις σε βάσεις δεδομένων διότι σήμερα είναι ένα από τα πιο κύρια σημεία των υπολογιστικών συστημάτων, στα οποία αποθηκεύονται σημαντικότατες πληροφορίες.Το SQL Injection αρχικά, είναι μια τεχνική έγχυσης κώδικα που χρησιμοποιείται για την επίθεση εφαρμογών που βασίζονται σε δεδομένα, στις οποίες εισάγονται κακόβουλες δηλώσεις SQL για διάφορους σκοπούς που θα δούμε παρακάτω και έπειτα θα κάνουμε μια μικρή αναφορά ιστορικά σε αυτό.Στο δεύτερο κεφάλαιο της πτυχιακής θα μελετήσουμε τους τύπους του SQL Injection και θα αναφερθούμε σε μικρά παραδείγματα SQL Injection με στόχο την καλύτερη κατανόηση τους. Στο τρίτο κεφάλαιο θα γνωρίσουμε τεχνικές εφαρμογής επιθέσεων, όπως για παράδειγμα την Trojan Horse και την Trojan Zebra, καθώς και μερικά από τα πιο γνωστά Sql injection tools.Στο τέταρτο και τελευταίο κεφάλαιο θα περιγράψουμε την υλοποίηση μιας φόρμας php που λαμβάνει τα στοιχεία των εργαζομένων του νοσοκομείου και τα περνά σε μια βάση δεδομένων.

.

ABSTRACT

In this thesis we will talk about database security and get to know SQL Injection. First we will define it and then we will refer to its history as well as decision making.In the second chapter we will study the types of SQL Injection and we will refer to small examples of SQL Injection for better understanding.In chapter three we will learn attack techniques such as Trojan Horse and Trojan Zebra and some of the best-known Sql injection tools.In the last chapter we will describe the implementation of a php form that receives the data of hospital workers and passes them to a database.

ΛΙΣΤΑ ΕΙΚΟΝΩΝ

Εικόνα 2.5 : Παράδειγμα κώδικα σε java

https://www.veracode.com/security/sql-injection

Εικόνα 2.6 : Παράδειγμα κώδικα σε java

https://www.veracode.com/security/sql-injection

Εικόνα 2.7 : Παράδειγμα κώδικα σε java

https://www.veracode.com/security/sql-injection

Εικόνα 2.8 : Παράδειγμα κώδικα σε java

https://www.veracode.com/security/sql-injection

Εικόνα 2.9 : Παράδειγμα κώδικα σε java

https://www.veracode.com/security/sql-injection

Εικόνα 2.10 : Παράδειγμα κώδικα σε java

https://www.veracode.com/security/sql-injection

Εικόνα 2.11 : Παράδειγμα κώδικα σε java

https://www.veracode.com/security/sql-injection

Εικόνα 2.12 : Παράδειγμα κώδικα σε java

https://www.veracode.com/security/sql-injection

Εικόνα 2.13 : Παράδειγμα κώδικα σε java

https://www.veracode.com/security/sql-injection

Εικόνα 3.1: BSQL Hacker

https://www.darknet.org.uk/2008/09/bsql-hacker-automated-sql-injection-framework/

Εικόνα 3.2: SQLmap

http://sqlmap.org/

Εικόνα 3.3: SQLninja

http://sqlninja.sourceforge.net/

Εικόνα 3.4: Safe3 SQL Injection

http://n3t5.blogspot.gr/2010/07/safe3-sql-injector-tool.html

Εικόνα 3.5: SQLSus

https://n0where.net/open-source-mysql-injection-sqlsus

Εικόνα 3.6: Mole

https://sourceforge.net/projects/themole/

Εικόνα 3.7:AMNESIA

http://slideplayer.com/slide/7087349/

Εικόνα 3.8: O JDBC- Checker

https://www.semanticscholar.org/paper/JDBC-checker%3A-a-static-analysis-tool-for-SQL%2FJDBC-Gould-Su/0743d913921a44750000889584a03b543e38825d

Εικόνα 3.9: To SQLrand

https://www.semanticscholar.org/paper/SQLrand%3A-Preventing-SQL-Injection-Attacks-Boyd-Keromytis/07c8dc37b1061784f3b55cf3ca5d2bc735e1693c

Εικόνα 4.1 :PHP

https://www.quora.com/What-is-the-step-by-step-process-to-become-a-PHP-developer

Εικόνα 4.2 : MySQL

https://c2.gr/blog/mysql/

Εικόνα 4.3 : Εγκατάσταση του EasyPHP

Εικόνα 4.4 : Εγκατάσταση του EasyPHP

Εικόνα 4.5 : Εγκατάσταση του EasyPHP

Εικόνα 4.6 : Εγκατάσταση του EasyPHP

Εικόνα 4.7 : Εγκατάσταση του EasyPHP

Εικόνα 4.8 : Εγκατάσταση του EasyPHP

Εικόνα 4.9 : Εγκατάσταση του EasyPHP

Εικόνα 4.10 : Εγκατάσταση του EasyPHP

1ο Κεφάλαιο: Εισαγωγή σε SQL Injection.

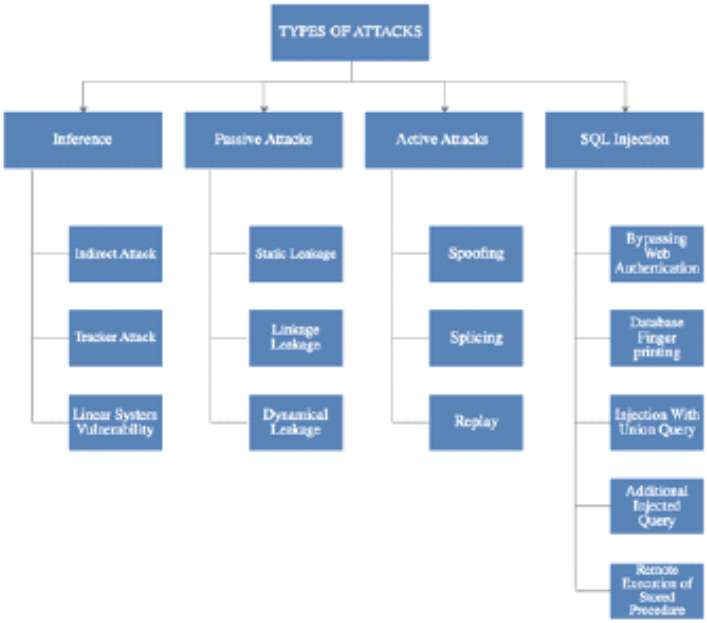
1.1 Ασφάλεια Βάσεων Δεδομένων

Μια βάση δεδομένων αποτελεί μια μέθοδο αποθήκευσης, ανταλλαγής καθώς και τροποποίησης δεδομένων. Η ασφάλεια των δεδομένων έδειξε να απασχολεί ιδιαίτερα τις επιχειρήσεις αλλά και τους απλούς χρήστες αφού καθημερινά διάφοροι κακόβουλοι χρήστες του διαδικτύου προσπαθούν να εκμεταλλευτούν τις μικρές ή μεγάλες ατέλειες της πλατφόρμας λογισμικού και των πρωτοκόλλων του Διαδικτύου. Με τον όρο ασφάλεια αναφερόμαστε στην προστασία της βάσης δεδομένων από την απώλεια δεδομένων, την παράνομη τροποποίηση δεδομένων καθώς και τη χρήση ακατάλληλων δεδομένων. Την σημερινή εποχή οι βάσεις δεδομένων μπορούν να χρησιμοποιηθούν σε πολλούς και διάφορους τομείς λόγου χάρη:

- Ιατρική
- Οικονομιά
- Εκπαίδευση
- Στρατό

Αυτές οι βάσεις δεδομένων εμπεριέχουν πολύ σημαντικές και ταυτόχρονα πολύ εμπιστευτικές πληροφορίες ώστε να υπάρξει πιθανότητα επιθέσεων. Μια βάση δεδομένων λοιπόν, καλείται να αντιμετωπίσει πολλά είδη επιθέσεων οι οποίες κατηγοριοποιούνται ως εξής:

- Inference attacks (Επιθέσεις συμπερασμάτων)
- Passive attacks (Παθητικές επιθέσεις)
- Active attacks (Ενεργές επιθέσεις)
- SQL Injection attacks (Επιθέσεις sql "ένεσης")



Εικόνα 1.1: Τύποι Επιθέσεων

Inference: Σε αυτό τον τύπο επίθεσης απαιτείται μόνο η άντληση χρήσιμων πληροφοριών από μη ευαίσθητα δεδομένα. Κατά κύριο λόγο οι χάκερς εδώ χρησιμοποιούν τις τρεις μεθόδους για να εκτελέσουν επίθεση. Την έμμεση επίθεση, την επίθεση tracker και την ευπάθεια συστήματος τακτικών γραμμών.Θαπρέπει να επισημάνουμε ότι όλοι οι χρήστες λαμβάνουν ευαίσθητα δεδομένα, έστω και σε μικρό ποσοστό.

Οι παθητικές επιθέσεις: Στιςσυγκεκριμένες επιθέσεις οι χάκερς ενδιαφέρονται μόνο για τα δεδομένα τα οποία έχουν ήδη παρουσιαστεί στη βάση δεδομένων. Συνοπτικά οι τρεις τρόποι σύμφωνα με τους οποίους έχουμε τη δυνατότητα να κάνουμε παθητική επίθεση είναι η στατική και δυναμική διαροή καθώς και η διαροή σύνδεσης.

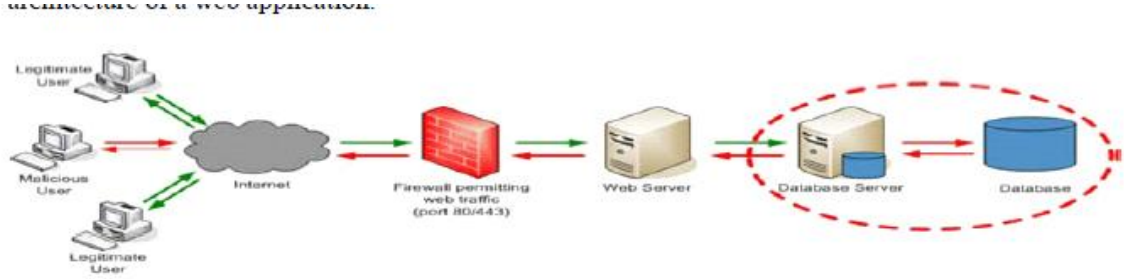
Οι ενεργές επιθέσεις: Σε αυτές τις επιθέσεις οι χάκερ δεν λαμβάνουν απλά τις πληροφορίες από τη βάση δεδομένων άλλα επιπλέον τροποποιούν τα πραγματικά δεδομένα λάθους.

Οι επιθέσεις SQL Injection: οι τρεις πρώτες επιθέσεις έχουν ως βάση το παραδοσιακό περιβάλλον του υπολογιστή. Με την πάροδο των χρόνων και την ανάπτυξη του Διαδικτύου, οι επιθέσεις, βασίζονται στο διακομηστή βάσης δεδομένων για εφαρμογή στο Διαδίκτυο. Η επίθεση η οποία στοχεύει στη βάση δεδομένων του

ιστού αποτελεί μια από τις βασικές επιθέσεις και δεν είναι άλλη από την επίθεση SQL Injection. Οι χάκερς μπορούν να επιτρέψουν στον διακομιστή να εκτελέσει τη δήλωση. Σύμφωνα με μελέτες οι χρήστες στο τέλος του 2016 έφτασαν τα 3 δισεκατομμύρια. Συνεπώς τα δεδομένα τα οποία αποθηκεύονται στη βάση δεδομένων του Ιστού είναι εξαιρετικά μεγάλα και σημαντικά. Συμπεραίνουμε λοιπόν ότι οι επιθέσεις SQL αποτελούν έναν πολύ σημαντικό και επικίνδυνο τύπο επίθεσης σε μια ενημερωμένη βάση δεδομένων.[4]

1.2 Ορισμός SQL Injection

Το SQL Injection αποτελεί μια τεχνική για την κακόβουλη εκμετάλλευση εφαρμογών που χρησιμοποιούν δεδομένα τα οποία με τη σειρά τους παρέχονται από τον πελάτη σε καταστάσεις SQL. Ένα SQL Injection θα πραγματοποιηθεί σε περίπτωση που ένας εισβολέας τοποθετήσει μια σειρά δηλώσεων SQL σε ένα ερώτημα που εισάγει ο χρήστης σε μια web-based εφαρμογή. Ο επιτιθέμενος έχει τη δυνατότητα να πάρει τα κενά ασφαλείας προγραμματισμού εφαρμογών και να περάσει κακόβουλες δηλώσεις SQL μέσω μια Web εφαρμογής για να εκτελέσει από τη βάση δεδομένων . Τα SQL Injection ανήκουν στην κατηγορία πλέον των πιο γνωστών επιθέσεων που συμβαίνουν στο Διαδίκτυο. Το SQL ερώτημα μεταφέρεται στο διακομιστή της βάσης δεδομένωνόπου και εκτελείται. Τα δεδομένα έχουν παράνομους χαρακτήρες χωρίς προειδοποίηση, όπως ερωτηματικά ή χαρακτήρες σχολίου, που ενδέχεται να οδηγήσουν σε απροσδόκητα ακόμη και επικίνδυνα ερωτήματα. Επίσης δεδομένα για τα οποία δεν ελέγχεται ο τύπος ενδέχεται να οδηγήσουν είτε σε εσφαλμένα είτε σε επικίνδυνα ερωτήματα. Σε περίπτωση που ένα ερώτημα αναμένει αριθμητική εισαγωγή, αλλά λαμβάνει κείμενο, η ελλατωματική είσοδος μπορεί να παρερμηνευθεί από τον διακομιστή βάσης ως εντολή κειμένου. Σε αυτή την κατηγορία σφάλματος, που προκύπτει από δεδομένα που δεν έχουν διαφύγει ορθά και ελέγχεται ο τύπος, αποτελούν μια ευπάθεια ασφάλειας η οποία πρέπει να διορθωθεί.Πρέπει να επισημανθεί ότι οιSQL Injection τύποι, έχουν αρκετές διαφορές απο τις παραδοσιακές επιθέσεις σε server. Σε μια επίθεση με έγχυση δεν υπάρχει καμία προσπάθεια πρόσβασης σε οποιοδήποτε είδος του διακομιστή. Ο βασικός στόχος του sql injection είναι η αξιοποίηση των ευπαθειών για την κατανόηση του διακομιστή και της ίδιας της βάσης δεδομένων από την εφαρμογή του Διαδικτύου. Στην παρακάτω εικόνα απεικονίζεται η θέση μιας τέτοιας ευπάθειας στην αρχιτεκτονική μιας διαδικτυακής εφαρμογής.



Εικόνα 1.2 : Η θέση μιας ευπάθειας στην αρχιτεκτονική μιας διαδικτυακής εφαρμογής

Η εν λόγο ευπάθεια χρησιμοποιείται με σκοπό να εκθέσει ευαίσθητα δεδομένα που θα επιτρέψουν την πρόσβαση σε μηχανήματα. Αναλυτικότερα, όπως αναφέραμε και προηγουμένως η τεχνική sql injection έχει στόχο την βάση και όχι τον διακομιστή.

Μπορούμε λοιπόν να κατανοήσουμε ότι όταν μιλάμε για SQL Injection αναφερόμαστε σε μια δυνατότητα εισαγωγής εντολών SQL στη μηχανή βάσης δεδομένων μέσω της υπάρχουσας εφαρμογής.

Το SQL Injection αποτελεί ένα ***«ελάττωμα»*** στη διαδικτυακή εφαρμογή όχι όμως στη βάση δεδομένων ή στον web server. Θα πρέπει να αναφέρουμε πως ανεξάρτητα από το πόσο καλά ασφαλισμένο είναι το σύστημα, ένας χάκερ έχει τη δυνατότητα να πάρει την πλήρη ιδιοκτησία της βάσης. [1][2]

1.3 Η ιστορία του SQL Injection

Μερικές από τις πιο σημαντικές ημερομηνίες στην ιστορία του SQL Injection συνοπτικά αναφέρονται παρακάτω:

- Το 1998 πραγματοποιήθηκε η πρώτη αναφορά στο Phrack 54.
- Το 2000 δημοσιεύτηκε το «πώς χάκαρα το Packetstorm – μια ματιά στο hacking wwwthreads μέσω SQL".
- Το 2002 ο Chris Anley δημοσίευσε το «Advanced SQL Injection σε εφαρμογές SQL Server.
- Το 2004 στο Blackhat το 0x90.org κυκλοφόρησε στο SQeaL.

1.3.1 Χρησιμότητα SQL Injection

Δύο από τις δυνατότητες του SQL Injection είναι:

- Η διαγραφή και τροποποίηση πληροφοριών από τη βάση.
- Η παράκαμψης ταυτότητας.

Η διαγραφή

Select productioninfo from table

Where product name= ‘whatever’;

DROP TABLE productinfo; --‘

Η παράκαμψη ταυτότητας

Select * from users

Where username=’user’ and password=’passwd’;

Select * from users

Where username=’admin’—‘and password=’whocares’; [2]

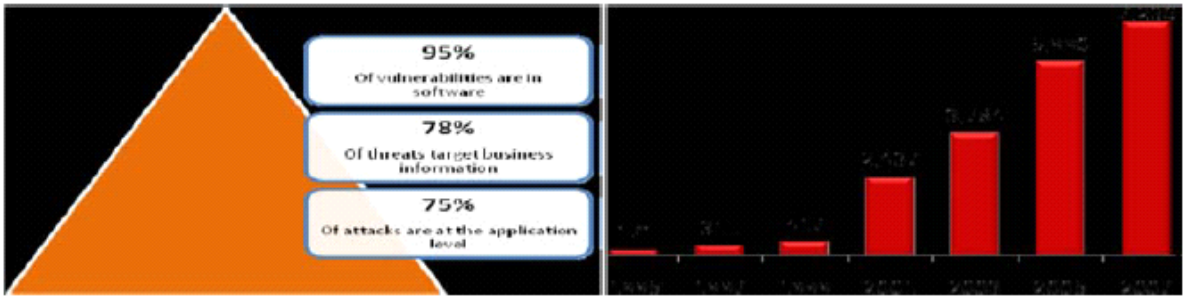
1.4 SQL Injection and Σχεδιαστικές αποφάσεις

1.4.1 Το λογισμικό

Η σημερινή εφαρμογή αποτελεί τη ***«νέα περίμετρο»*** της επιχείρησης. Με την βέλτιστη τεχνολογία ασφάλειας σε επίπεδο δικτύου (που σκληραίνει την περίμετρο του δικτύου) οι κακόβουλοι οι οποίοι επιτίθενται συγκεντρώνουν την προσπάθεια τους στην ***«απεργία»*** στα λιγότερα υπερασπιζόμενα σημεία. Παλαιότερα οι χάκερς ήταν ευχαριστημένοι με την απομάκρυνση ιστότοπων Web, την εξαφάνιση των επιθέσεων άρνησης παροχής υπηρεσιών καθώς και την εμπορία παράνομων αρχείων μέσω στοχοθετημένων δικτύων . Από την άλλη πλευρά οι σύγχρονοι επιτιθέμενοι είναι κερδοσκοπικοί. Όσον αφορά τα οικονομικά στοιχεία καθώς και τα δεδομένα

των πελατών έγιναν πολύ σημαντικά εμπορεύματα και οι εφαρμογές είναι πολύ ασφαλείς για την προστασία τους.

Στην παρακάτω εικόνα παρουσιάζονται οι πρόσφατες βιομηχανικές στατιστικές του εν λόγο λογισμικού .



Εικόνα 1.3: Πρόσφατες βιομηχανικές στατιστικές

Όπως μπορούμε να διακρίνουμε τα δεδομένα από το CERT,διαπιστώνουμε ότι ο αριθμός των τρωτών σημείων του λογισμικού έχει αυξηθεί σε πολύ μεγάλο βαθμό και έχει σκιάσει 7000 νέες αποκαλύψεις ευπάθειας λογισμικού το 2014. Σύμφωνα με την Gartner και την NIST το 95% των τρωτών σημείων στο λογισμικό , το 75% απευθύνεται σε επιχειρηματικές πληροφορίες και το 25% των επιθέσεων στοχεύουν στο επίπεδο εφαρμογής. Παρόλα αυτά οι περισσότερες επιχειρήσεις κατανέμουν λιγότερο από το 10% των κοστών ασφαλείας τους για την ασφάλεια των εφαρμογών.

1.4.2 Σχεδιασμένες αποφάσεις

Κάθε απόφαση διασπάται σε δυο επιμέρους μέρη. Στο πρώτο μέρος καλείται να εξηγήσει τα δυνητικά μειονεκτήματα τα οποία θα έχει μια επιλογή εν σύγκριση με την εναλλακτική λύση ενώ από την άλλη πλευρά στο δεύτερο μέρος καλείται να εξηγήσει τα πλεονεκτήματα της καθώς και να προσπαθήσει να βοηθήσει τον εκάστοτε αναγνώστη να κατανοήσει τον λόγο που λήφθησαν ορισμένες σχεδιαστικές αποφάσεις. [1]

2ο Κεφάλαιο : Τύποι SQL Injection

2.1 Εισαγωγή

Όσον αφορά τα είδη των επιθέσεων θα αναφέρουμε ότι για να μπορέσουμε να δώσουμε μια σαφή εικόνα των τεχνολογιών που χρησιμοποιούνται για να επιτύχουμε μια επίθεση εισαγωγής sql εντολής θα παρουσιάσουμε τις κατηγορίες καθώς και τη μεθοδολογία των sql injection επιθέσεων.

2.2 Τύποι SQL Injection

Η επίθεση SQL Injection είναι ιδιαίτερα ευέλικτη και διαφοροποιημένη όπως προαναφέραμε. Μπορούμε να την ταξινομήσουμε με δύο τρόπους ως προς:

- ***Τον στόχο***
- ***Το SQL ερώτημα που χρησιμοποιεί***

Ακολουθεί εκτενέστερη αναφορά σε καθεμία εκ των παραπάνω κατηγοριών.

2.2.1 SQL Injection ως προς τον στόχο

Με τον όρο "ως προς τον στόχο" πρέπει να διευκρινίσουμε ότι εννοούμε μέσα από αυτή την επίθεση δίνεται η δυνατότητα σε κάποιον κακόβουλο επιτιθέμενο να "τρέξει" εντολές SQL ενάντια σε ένα server – στόχο και στη συνέχεια να αποσπάσει αρκετά ευαίσθητες πληροφορίες (όπως για παράδειγμα κωδικοί πρόσβασης, ονόματα χρηστών, emails, αριθμοί πιστωτικών καρτών κ.α) μέσα από την βάση δεδομένων στην οποία και επιτίθεται. Παρακάτω βλέπουμε πέντε από αυτές τις ενέργειες:

- Η Ανάκτηση δεδομένων
- Η τροποποίηση δεδομένων
- Η αλλαγή σχήματος στη βάση δεδομένων
- Το κλείσιμο της σύνδεσης ανάμεσα σε διακομιστή και βάση δεδομένων
- Τα συστήματα ελέγχου εξ αποστάσεως

Ανάκτηση δεδομένων: αυτός ο τύπος αποτελεί έναν από τους πιο συνηθισμένους . Η εν λόγο επίθεση αποσκοπεί στην υποκλοπή δεδομένων τα οποία αποθηκεύονται σε

εφαρμογές διαδικτύου. Μερικά από αυτά τα δεδομένα που δύναται να υποκλαπούν είναι:

- ***Το SSN***
- ***Ο κωδικός της πιστωτικής***
- ***Ο τραπεζικός λογαριασμός***

Η τροποποίηση δεδομένων : συχνά οι χάκερς επιθυμούν την πρόσβαση σε δεδομένα με απώτερο σκοπό να επέμβουν στο περιεχόμενο τους και να τα αλλάξουν. Επί της ουσίας προσπαθούν είτε να τα διαγράψουν είτε να τα τροποποιήσουν τα δεδομένα σε σχέση με τα αρχικά .

Η αλλαγή σχήματος στη βάση δεδομένων: ο συγκεκριμένος στόχος δεν αποσκοπεί μόνο στα αρχεία αλλά και στις δομές των βάσεων δεδομένων.

Το κλείσιμο της σύνδεσης ανάμεσα σε διακομιστή και βάση δεδομένων: ο συγκεκριμένος τύπος επίθεσης θα τερματίσει τη λειτουργία της βάσης δεδομένων σε εφαρμογή στο Διαδίκτυο. Με αυτόν τον τρόπο εμποδίζεται η εξυπηρέτηση.

Τα συστήματα ελέγχου εξ αποστάσεως : η πιο επικίνδυνη έγχυση για το λόγο ότι επιτρέπει στους χάκερς να αποκτήσουν τον έλεγχο του συνολικού συστήματος.

2.2.2 Sql injection ως προς τον τύπο ερωτήματος

Όσον αφορά την κατηγοριοποίηση Sql injection ως προς τον τύπο ερωτήματος θα διαχωριστούν σε έξι υποκατηγορίες και πιο συγκεκριμένα σε: Tautologies, Illegal/Logically Incorrect Queries, Union Query, Piggy- back Queries, Alternate Encodings και Inference. Ακολουθεί εκτενέστερη αναφορά για καθεμία από τις προαναφερθείσες υποκατηγορίες.

***2.2.2.1 Επίθεση με βάση ταυτολογία*:** Αποτελεί ένα είδος επίθεσης όπου ο εισβολέας εισάγει ένα ερώτημα αξιολογώντας το πάντα ως αληθές. Η συγκεκριμένη επίθεση έχει να κάνει με καταχωρήσεις στη βάση δεδομένων, παράκαμψη της επαλήθευσης ταυτότητας και εξαγωγή δεδομένων.

Οι επιθέσεις ταυτολογιών αποτελούν τον πιο απλό και γνωστό SQLIA . Ο συγκεκριμένος τύπος εισάγει κακόβουλο κώδικα σε ερωτήματα SQL και επιπλέον τροποποιεί τη δήλωση κωδικοποίησης.

Η δήλωση συνθηκών του ερωτήματος SQL αποκαθιστά ότι τα ανακτημένα δεδομένα πληρούν μια και μόνο σημαντική προυπόθεση. Τα ανακτημένα δεδομένα θα εμφανίσουν όλα τα δεδομένα των γραμμών σε ένα πίνακα βάσεων δεδομένων αν η δήλωση συνθήκης διαγράφεται από επίθεση ταυτολογιών. Στη φόρμα σύνδεσης οι χρήστες καλούνται να εισάγουν αναγνωριστικό χρήστη και κωδικό πρόσβασης.

$sql= "SELECT * FROM user WHERE id= '$id' AND password= '$password'";

Σε περίπτωση που υπάρχει ευπάθεια επαλήθευσης στην είσοδο του χρήστη ένας χάκερ έχει τη δυνατότητα να ανακτήσει όλες τις σειρές δεδομένων από τον πίνακα χρηστών πληκτρολογώντας **x ή 1=1.** Πληκτρολογώντας **x** στο πλαίσιο κειμένου του κωδικού πρόσβασης η φόρμα σύνδεσης ορισμένες φορές μπορεί να λάβει τον κωδικό πρόσβασης του διαχειριστή καθώς και τον κωδικό πρόσβασης στον πίνακα χρηστών. Σε περίπτωση που ο διαχειριστής και το όνομα χρήστη και ο κωδικός πρόσβασης δεν είναι αποθηκευμένα σε πίνακα το ερώτημα SQL θα τροποποιηθεί και θα εκτελεστεί όπως φαίνεται παρακάτω:

$sql= 'SELECT * FROM user WHERE id= 'x' or '1=1'-- AND password= 'x';

Το x μπορεί να πάρει οποιαδήποτε τιμή διότι 1=1 είναι πάντα αληθές για τη δήλωση συνθηκών και το σχόλιο μιας γραμμής το σύμβολο διπλής παύσης "-" θα σχολιάσει όλα τα μέρη του ερωτήματος SQL μετά από αυτό και θα αγνοηθεί από την εκτέλεση SQL.

Υπάρχει και άλλη παράγωγη μορφή ταυτολογιών:

Admin'--; admin'#;' or 1=1--; ' or 1=1#; ') or '1'='1'--;') or ('1'='1'#; "or 1=1--; " or 1=1 or ""="; ' or (EXISTS); ' or username like '%;' or userid like '%; ' or username like '%;

Για να αλλάξετε τον κωδικό πρόσβασης του διαχειριστή χωρίς να προηγηθεί έλεγχος του παλιού κωδικού πρόσβασης:

UPDATE users SET pa s s word='1234' WHERE

Username=’admin’—‘ AND password =’1234’

Σε περίπτωση που ο τύπος δεδομένων εισόδου είναι ακέραιος SQLIA υπάρχει η δυνατότητα εκκίνησης χωρίς ενιαίο εισαγωγικό

SELECT * FROM user WHERE useid=12 or 1=1;

Ένα ακόμη πολύ επικίνδυνο θέμα ευπάθειας αποτελεί η χρήση cookies για την παροχή σεναρίων ως παραμέτρους που αγνοούνται από του προγραμματιστές τις περισσότερες φορές.

<?

$pass= $COOKIE[‘psswd’];

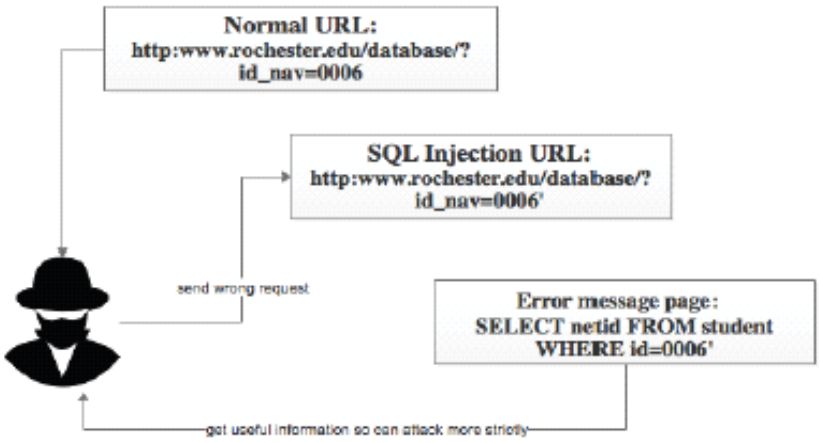
$user= mysql_query (‘SELECT password FROM

User WHERE pass = $psswd”);

?>

Οι χάκερς έχουν τη δυνατότητα να μπορούν να αλλάξουν τον κωδικό πρόσβασης χρήστη χρησιμοποιώντας τον παραπάνω κωδικό.

2.2.2.2 Illegal/Logical Incorrect Queries: Παράνομα/λογικά εσφαλμένα ερωτήματα. Οι χάκερς ορισμένες φορές χρησιμοποιούν το σφάλμα μηνύματος για να βρουν ευαίσθητες παραμέτρους στην εφαρμογή ιστού. Με αυτό τον τρόπο οι επιτιθέμενοι θα στέλνουν εσφαλμένα ερωτήματα (επί σκοπό). Στην εικόνα που ακολουθεί μπορούμε να διακρίνουμε ένα δείγμα παράνομου / εσφαλμένου ερωτήματος.



Εικόνα 2.1: Δείγμα παράνομου / εσφαλμένου ερωτήματος

Ο εν λόγο τύπος SQLIA αποτελεί έναν από τους πιο σημαντικούς τύπους επίθεσης. Αποτελεί το αρχικό βήμα για τη συλλογή σημαντικών πληροφοριών τόσο του τύπου όσο και της δομής του διακομιστή της βάσης δεδομένων. Οι χάκερς υποβάλουν σκόπιμα εσφαλμένα SQL queries για να επιτρέψουν στον διακομιστή βάσης δεδομένων να απορρίψει τα ερωτήματα και να εμφανίσει μήνυμα ανατροφοδότησης. Η εφαρμογή ιστού είναι ευάλωτη και οι χάκερς χρησιμοποιούν τεχνολογίες SQLIA για να εκμεταλλευτούν τη βάση δεδομένων και να εξάγουν δεδομένα από αυτή.

Database Type	Feed-back error messages
MySQL	Warning: mysql_errno(): supplied argument is not a valid MySQL
ODBC.ASP	Microsoft OLE DB Provider for odbc Drivers error '80040e21'
Oracle	SQLException: ORA-01722: invalid number
ODBC. C#	[Microsoft][ODBC SQL Server Driver][SQL Server] Unclosed quotation mark
.NET	Stack Trace: SqlException (0x80131904)
PostgreSQL	Warning: PostgreSQL query failed
ColdFusion	Invalid data for CFSQLTYPE

Εικόνα 2.2: Κοινά μηνύματα λάθους βάσης δεδομένων

Κάθε τύπος βάσης δεδομένων επιστρέφει διαφορετικό μήνυμα με αυτό τον τρόπο οι χάκερς μπορούν να ανιχνεύσουν ποιος είναι ο τύπος βάσης δεδομένων που υποστηρίζει την εφαρμογή ιστού ανάλογα με το μήνυμα (βλ. εικόνα 2.2).

***2.2.2.3 Union Query*:** ερώτημα της ένωσης. Με την χρήση της ένωσης οι εισβολείς μπορούν να επιτρέπουν την εξαγωγή πληροφοριών από μια βάση δεδομένων. Το Union query μπορεί να χρησιμοποιηθεί μόνο όταν και τα δύο ερωτήματα έχουν την ίδια δομή . Ο επιτιθέμενος μπορεί να δημιουργήσει μια εντολή SELECT παρόμοια με το αρχικό ερώτημα.

***2.2.2.4 Piggy- back Queries*:** ερωτήματα τα οποία υποστηρίζονται από το Piggy. Στο συγκεκριμένο τύπο έγχυσης οι εισβολείς χρησιμοποιούν οριοθέτηση ερωτήματος (λόγου χάρη ΄΄΄΄ ΄΄) για να δεσμεύσουν ερωτήματα. Ένας εισβολέας εισάγει επιπλέον ερώτημα στο αρχικό για να εξάγει δεδομένα από τη βάση. Μπορεί να προσθέσει ή να τροποποιήσει δεδομένα και να εκτελέσει απομακρυσμένες εντολές. Σε αυτή την

περίπτωση ο εισβολέας δεν θέλει να τροποποιήσει το αρχικό ερώτημα αλλά να συμπεριλάβει και νέα ερωτήματα που αφορούν το αρχικό.

***2.2.2.5 Alternate Encodings*:**εναλλασσόμενες κωδικοποιήσεις. Στη συγκεκριμένη τεχνική οι εισβολείς χρησιμοποιούν διαφορετική μέθοδο κωδικοποίησης λόγου χάρη Unicode, ASCII.

***2.2.2.6 Inferential*:** στον συγκεκριμένο τύπο επίθεσης καλείται επαγωγικό SQLi – τυφλό SQLi. Η επαγωγική έγχυση SQL εν αντιθέση με την Blind SQLi έχει την δυνατότητα εκμετάλλευσης περισσότερων του ενός επιτιθέμενους. Ο επιτιθέμενος δεν μπορεί να δει το αποτέλεσμα μιας τέτοιας επίθεσης αλλά μπορεί να αναδημιουργήσει τη δομή βάσεων δεδομένων . [3][4]

Ακολουθεί ο πίνακας ανά τεχνικές που ανιχνεύουν ή αποτρέπουν την SQLIA.

App	Que-ries	Precis-ion
Bookstore	68%	99.95
Forum	56%	99.93
Clasifieds	61%	99.94
NewsPortal	61%	99.92
JobPortal	61%	99.96

Εικόνα 2.3: Πίνακας ανά τεχνικές που ανιχνεύουν ή αποτρέπουν την SQLIA

3ο Κεφάλαιο: Παραδείγματα SQL Injection

3.1 Εισαγωγή

Σε αυτό το σημείο της εργασίας μας παρουσιάζουμε παραδείγματα Sql injection με απώτερο σκοπό να γίνουμε περισσότερο κατανοητοί. Πιο συγκεκριμένα παρουσιάζουμε παραδείγματα με χρήση της εντολής select, παράδειγμα σύνδεσης χρήστη σε έναν ιστότοπο, έγχυσης SQL βασισμένη σε καταγεγραμμένες δηλώσεις SQL, δυνατότητας χρήσης των παραμέτρων SQL και παραδείγματα επίθεσης και άμυνας, επίσης, παράδειγμα επίθεσης αβλαβούς ένεσης και τέλος πως μπορούμε να ανακτήσουμε ένα πίνακα και τα ονόματα των στηλών του.

3.2Παράδειγμα με χρήση της εντολής select

Στη συνέχεια παραθέτουμε ένα παράδειγμα στο οποίο δημιουργείται μια εντολή SELECT προσθέτοντας μια μεταβλητή txtUserId σε μια συμβολοσειρά επιλογής. Η μεταβλητή λαμβάνεται από την είσοδο του χρήστη (getRequestString)

txtUserId= getRequestString ("UserId");

txtSQL= "SELECT * FROM Users WHERE UserId= " + txtUserId;

Η SQL Injection με βάση το 1=1 είναι πάντα True.

Στη συνέχεια θα ανατρέξουμε στον προηγούμενο κώδικα που ο βασικός ρόλος ήταν η δημιουργία μιας δήλωσης SQL για την επιλογή ενός χρήστη με ένα συγκεκριμένο αναγνωριστικό χρήστη.

Ταυτότητα χρήστη : 105 OR 1=1

Έπειτα η εντολή SQL θα πάρει τη μορφή :

SELECT * FROM Users WHERE UserId = 105 OR 1=1;

Μπορούμε να διαπιστώσουμε ότι η παραπάνω SQL είναι έγκυρη και θα επιστρέψει όλες τις σειρές του πίνακα με όνομα Users και αυτό συμβαίνει για το λόγο ότι το OR 1=1 είναι πάντα αληθές άρα True.

Πότε μπορεί το παράδειγμα να γίνει επικίνδυνο;

SELECT UserId, Name, Password FROM Users WHERE UserId =105 or 1-1;

Σε αυτή την περίπτωση ένας χάκερ έχει τη δυνατότητα πρόσβασης σε όλα τα ονόματα των χρηστών καθώς και τους κωδικούς πρόσβασης σε μια βάση δεδομένων αρκεί μόνο να εισάγει **105 ή 1=1** στο πεδίο εισαγωγής.

SQL Injection με βάση το ΄΄΄΄=΄΄΄΄ είναι πάντα αληθής δηλαδή **True**

3.3 Παράδειγμα σύνδεσης χρήστη σε έναν ιστότοπο

Στη συνέχεια παρουσιάζεται ένα όνομα χρήστη **John Doe** και για κωδικό πρόσβασης χρησιμοποιούμε το **myPass**:

Όπως φαίνεται και παρακάτω:

Username: John Doe

Password: myPass

uName=getRequestString (΄΄username΄΄);

uPass= getRequestString (΄΄userpassword΄΄);

Το ερώτημα που θα τρέξει η sql μετά την παραπάνω είσοδο είναι:

sql= SELECT *FROM Users WHERE Name= ΄΄ ΄+ uName + ΄΄ ΄AND Pass = ΄΄ ΄+uPass+΄΄ ΄

Και το αντίστοιχο αποτέλεσμα στον πλοηγό του ιστότοπου είναι το εξής:

SELECT * FROM Users WHERE Name = ΄΄ John Doe ΄΄ AND Pass= ΄΄myPass΄΄

Τώρα θα δούμε μια διαφορετική περίπτωση στην οποία ένας χάκερ έχει τη δυνατότητα πρόσβασης σε ονόματα χρηστών καθώς και κωδικούς πρόσβασης σε μια βάση δεδομένων αρκεί μόνο να εισάγει ΄΄ **OR**΄΄ ΄΄=΄΄ στο πλαίσιο κειμένου ονόματος χρήστη ή τον κωδικό πρόσβασης.

Username: ΄΄ or ΄΄΄΄=΄΄

Password: ΄΄ or ΄΄΄΄=΄΄

Ο κώδικας του διακομιστή θα φτιάξει μια έγκυρη πρόταση SQL όπως αυτή που ακολουθεί:

SELECT *FROM Users WHERE Name= ΄΄΄΄ or ΄΄=΄΄ AND Pass = ΄΄΄΄ or ΄΄=΄΄

Η προαναφερθείσα SQL είναι έγκυρη και θα επιστρέψει όλες τις σειρές από τον πίνακα με όνομα Users και αυτό θα συμβεί για το λόγω του ότι η εντολή **or ΄΄=΄΄** είναι πάντα αληθής δηλαδή **True.**

3.4 Παράδειγμα Έγχυσης SQL βασισμένη σε καταγεγραμμένες δηλώσεις SQL

Οι περισσότερες βάσεις δεδομένων αποτελούνται από δέσμες εντολών SQL. Κάθε μια από τις δέσμες αποτελεί μια ομάδα δύο ή περισσότερων εντολών SQL, οι οποίες χωρίζονται με ερωτηματικά. Ακολουθεί ένα παράδειγμα δήλωσης SQL το οποίο θα επιστρέψει όλες τις σειρές από έναν πίνακα με όνομα Employees και έπειτα θα διαγράψει τον πίνακα με όνομα Producers.

SELECT * FROM Employees; DROP TABLE Producers

Εάν μπορούσαμε να έχουμε τον κώδικα του διακομιστή-server, θα ήταν ο εξής:

txtUserId= getRequestString (" UserId");

txtSQL= "SELECT * FROM Employees WHERE UserId = "+txtUserId;

Σε περίπτωση που δώσουμε ως είσοδο στο πεδίο του "Username": ***105; DROP TABLE Producers***, τότε μια έγκυρη εντολή SQL θα έχει δημιουργηθεί από τον διακομιστή-server σε μορφή:

SELECT * FROM Employees WHERE UserId =105; DROP TABLE Producers;

3.5 Παράδειγμα δυνατότητας χρήσης των παραμέτρων SQL

Σε περίπτωση που επιθυμούμε να προστατέψουμε μια τοποθεσία Web από SQL Injection μας δίνεται η δυνατότητα χρήσης των παραμέτρων SQL. Με τον όρο παράμετροι SQL αναφερόμαστε σε τιμές οι οποίες προστίθενται σε ένα ερώτημα SQL με ελεγχόμενο τρόπο.

txtUserId= getRequestString (" UserId");

txtSQL= "SELECT * FROM Users WHERE UserId = @0";

db.Execute (txtSQL, txtUserId);

Οι παράμετροι αναπαρίστανται στην εντολή SQL από έναν δείκτη @. Επίσης ο μηχανισμός SQL έχει τη δυνατότητα ελέγχου κάθε παραμέτρου με στόχο να βεβαιωθεί ότι είναι ορθή για τη στήλη του και αυτό το αντιμετωπίζει κυριολεκτικά. Παρακάτω βλέπουμε ακόμα ένα παράδειγμα:

txtNam= getRequestString ("CustomerName");

txtAdd= getRequestString ("Address");

txtCit= getRequestString (" City");

txtSQL= "INSERT INTO Customers (CustomerName, Address, City)

Values (@0, @1, @2)";

Db.Execute(txtSQL, txtNam, txtAdd, txtCit);[5]

3.6 Παραδείγματα επίθεσης και άμυνας

Σε αυτό το σημείο της εργασίας μας θα αναφέρουμε ορισμένα παραδείγματα επίθεσης και άμυνας. Ας υποθέσουμε ότι ο προγραμματιστής επιθυμεί να εμφανίσει τους αριθμούς λογαριασμούς καθώς και τα υπόλοιπα για το αναγνωριστικό του χρήστη που παρέχεται σε μια διεύθυνση URL. Σε αυτή την περίπτωση μπορούμε να γράψουμε σε java:

```
String accountBalanceQuery =
  "SELECT accountNumber, balance FROM accounts WHERE account_owner_id = "
  + request.getParameter("user_id");

try {
    Statement statement = connection.createStatement();
    ResultSet rs = statement.executeQuery(accountBalanceQuery);
    while (rs.next()) {
        page.addTableRow(rs.getInt("accountNumber"), rs.getFloat("balance"));
    }
} catch (SQLException e) { ... }
```

Εικόνα 2.4 : Παράδειγμα κώδικα σε java

Όπως μπορούμε να κατανοήσομε υπό φυσιολογική λειτουργία ο χρήστης με αναγνωριστικό ID 984 μπορεί να συνδεθεί καθώς και να επισκεφθεί την παρακάτω διεύθυνση: https://bankingwebsite/show_balances?user_id=984

Σύμφωνα με το παραπάνω μπορούμε να καταλάβουμε ότι το Account Balance Query θα είναι:

```
SELECT accountNumber, balance FROM accounts WHERE account_owner_id = 984
```

Εικόνα 2.5: Παράδειγμα κώδικα σε java

Το παραπάνω μεταβιβάζεται στη βάση δεδομένων και επιπλέον επιστρέφονται οι λογαριασμοί καθώς και τα υπόλοιπα για τον χρήστη 984 τα οποία θα εμφανιστούν επιπλέον γραμμές στη σελίδα. Ο εισβολέας έχει τη δυνατότητα αλλαγής του user_id :

```
0 OR 1=1
```

Εικόνα 2.6 : Παράδειγμα κώδικα σε java

Το παραπάνω οδηγεί στο λογαριασμό Balance Query :

```
SELECT accountNumber, balance FROM accounts WHERE account_owner_id = 0 OR 1=1
```

Εικόνα 2.7 : Παράδειγμα κώδικα σε java

Μόλις το εν λόγο ερώτημα μεταβιβαστεί στη βάση δεδομένων θα επιστρέψει όλους τους αριθμούς λογαριασμών και τα υπόλοιπα που αποθηκεύτηκαν και επιπλέον θα προστεθούν νέες γραμμές στη σελίδα. Σε αυτή την περίπτωση ο εισβολέας γνωρίζει τους αριθμούς λογαριασμών και τα υπόλοιπα του κάθε χρήστη.

Ο προγραμματιστής θα μπορούσε εύκολα να επιδιορθώσει το συγκεκριμένο θέμα ευπάθειας με τη χρήση μιας προετοιμασμένης δήλωσης :

```
String accountBalanceQuery =
  "SELECT accountNumber, balance FROM accounts WHERE account_owner_id = ?";

try {
    PreparedStatement statement = connection.prepareStatement(accountBalanceQuery);
    statement.setInt(1, request.getParameter("user_id"));
    ResultSet rs = statement.executeQuery();
    while (rs.next()) {
        page.addTableRow(rs.getInt("accountNumber"), rs.getFloat("balance"));
    }
} catch (SQLException e) { ... }
```

Εικόνα 2.8 : Παράδειγμα κώδικα σε java

Σε περίπτωση που ο εισβολέας επιθυμεί να δώσει τιμή η οποία δεν είναι ακέραιος τότε η εντολή statement.setInt() θα πετάξει ένα σφάλμα SQLException αντί να πραγματοποιηθεί ολοκλήρωση του ερωτήματος.

Ας υποθέσουμε ότι ένας προγραμματιστής εφαρμόζει μια φόρμα σύνδεσης και γράφει το παρακάτω:

```
String userLoginQuery =
  "SELECT user_id, username, password_hash FROM users WHERE username = '"
  + request.getParameter("user") + "'";

int user_id = -1;
HashMap userGroups = new HashMap();

try {
    Satement statement = connection.createStatement();
    ResultSet rs = statement.executeQuery(userLoginQuery);
    rs.first();
    user_id = rs.getInt("user_id");
    if (!
     hashOf(request.getParameter("password")).equals(rs.getString("password_hash"))
    ) {
        throw BadLoginException();
    }

    String userGroupQuery = "SELECT group FROM group_membership WHERE user_id = " + user_id;
rs = statement.executeQuery(userGroupQuery);
    while (rs.next()) {
        userGroup.put(rs.getString("group"), true);
    }
}
catch (SQLException e) { ... }
catch (BadLoginException e) { ... }
```

Εικόνα 2.9 : Παράδειγμα κώδικα σε java

Κανονικά ένας χρήστης θα παρέχει ένα όνομα χρήστη λόγου χάρη John καθώς και τον κωδικό πρόσβασης και το πρώτο ερώτημα που θα εκτελεστεί θα είναι:

```
SELECT user_id, username, password_hash FROM users WHERE username = 'john'
```

Εικόνα 2.10 : Παράδειγμα κώδικα σε java

Η βάση δεδομένων επιστρέφει το αναγνωριστικό χρήστη του John καθώς και τον κωδικό πρόσβασης του. Έπειτα η βάση δεδομένων παίρνει λίστα των ομάδων στις οποίες ανήκει ο John. Σε περίπτωση που ένας εισβολέας έχει τον

κωδικό John σε αυτή την περίπτωση ο επιτιθέμενος θα μπορούσε να είναι ο John!. Επίσης θα μπορούσαν να μετατρέψουν τον John σε διαχειριστή και να συνδεθούν χρησιμοποιώντας τον ακόλουθο ως όνομα χρήστη:

```
john';
INSERT INTO group_membership (user_id, group)
VALUES (SELECT user_id FROM users WHERE username='john', 'Administrator'); --
```

Εικόνα 2.11 : Παράδειγμα κώδικα σε java

Άρα το ερώτημα που θα διαβιβαστεί στη βάση θα είναι της μορφής:

```
SELECT user_id, username, password_hash FROM users WHERE username = 'john';
INSERT INTO group_membership (user_id, group)
VALUES (SELECT user_id FROM users WHERE username='john', 'Administrator'); --'
```

Εικόνα 2.12 : Παράδειγμα κώδικα σε java

Θα πρέπει να προσθέσουμε σε αυτό το σημείο ότι ο εισβολέας μπορεί να αλλάξει την ομάδα του John χωρίς να ολοκληρωθεί η διαδικασία σύνδεσης . Άρα ο εισβολέας ο οποίος δεν έχει μαντέψει τον κωδικό του John θα μπορούσε να διαγράψει λογαριασμούς και να επαναφέρει τους κωδικούς πρόσβασης .

```
String userLoginQuery =
  "SELECT user_id, username, password_hash FROM users WHERE username = ?";

…

try {
    PreparedSatement statement = connection.prepareStatement(userLoginQuery);
    statement.setString(1, request.getParameter("user"));
    ResultSet rs = statement.executeQuery();
…
```

Εικόνα 2.13 : Παράδειγμα κώδικα σε java

Το συγκεκριμένο ερώτημα δεν θα επιστρέψει αποτελέσματα όταν η παράμετρος χρήστη περιέχει μια επίθεση SQL Injection αφού η τιμή της παραμέτρου καθορίζεται από το Prepared Statement. [7]

3.7 Παράδειγμα επίθεσης αβλαβούς ένεσης και αξιολόγηση των αποτελεσμάτων

Εδώ θα δούμε πως μπορούμε να πειραματιστούμε με επιθέσεις αβλαβούς ένεσης για την βάση δεδομένων αλλά θα μας δείξουν ότι πρέπει να διορθώσουμε την σελίδα.

Για παράδειγμα έχουμε ένα site που αναζητά ένα άτομο σε μια βάση και ως αποτέλεσμα παρέχει πληροφορές επικοινωνίας όπως το παρακάτω:

http://myfakewebsite.com/directory.asp?lastname=chapple&firstname=mike

Υποθέτουμε ότι αυτή η σελίδα εκτελεί μια αναζήτηση βάσης δεδομένων χρησιμοποιώντας το εξής ερώτημα:

SELECT phone

FROM directory

WHERE lastname='chapple' and firstname='mike'

Έτσι μπορούμε να δοκιμάσουμε μια αλλαγή στη URL για να δούμε αν δουλεύει το SQL Injection όπως αυτό:

http://myfakewebiste.com/directory.asp?lastname=chapple&firstname=mike'+AND+(select+count(*)+from+fake)+%3eD+OR+'1'%3d'1

Αν η εφαρμογή δεν έχει προστασία από sql injection απλώς συνδέει αυτό το ψεύτικο πρώτο όνομα στη δήλωση SQL που εκτελεί κατά της βάσης δεδομένων με αποτέλεσμα το ερώτημα SQL μετά από SQL Injection να είναι:

SELECT phone

FROM directory

WHERE lastname='chapple' and firstname='mike'

AND (select count (*) from false)>0

OR '1'='1'

Πήραμε την ελευθερία της μετατροπής της URL-encoded μεταβλητής σε ισοδύναμα ASCII ώστε αν είναι πιο εύκολη η παρακολούθηση τους, πχ. το 3%d είναι η κωδικοποίηση URL για τον χαρακτήρα '='.

Εν κατακλείδι, όταν δοκιμάσουμε να τρέξουμε την σελίδα, αν η εφαρμογή είναι σωστή, θα απομακρύνει τα ' ' πριν περάσει το ερώτημα στη βάση. Έτσι, θα εμφανιστεί μύνημα σαν αυτό:

Error No user found with name mike+AND+(select+count(*)+from+fake)+%3e0+OR+1%3d1Chapple!

Αν όμως η εφαρμογή είναι ευάλωτη σε sql injection, θα περάσει τη δήλωση στη βάση δεδομένων με αποτέλεσμα μια από τις δύο πιθανότητες. Είτε ο server έχει λεπτομερές σφάλμα που δεν θα έπρεπε, θα εμφανιστεί μύνημα,

Microsoft OLE DB Provider for ODBC Drivers error '80040e37'

[Microsoft][ODBC SQL Server Driver][SQL Server]Invalid object name 'fake'.

/directory.asp, line 13 "

αλλίως θα εμφανίσει ενα μύνημα με πιο γενικό σφάλμα.

3.8 Παράδειγμα πίνακα σε MySQL 5

Σε αυτό το παράδειγμα θα δούμε πως μπορούμε να ανακτήσουμε ένα πίνακα και τα ονόματα στηλών του. Γι αυτό, θα χρειαστούμε το information_schema το οποίο ανακτά όλους τους πίνακες και τις στήλες στη βάση δεδομένων. Για να πάρουμε τους πίνακες χρησιμοποιούμε το όνομα πίνακα "table_name" και "information_schema.tables."

Για παράδειγμα, http://server/news.php?id=5 union all select 1,table_name,3 from information_schema.tables/*

Εδώ, αντικαθιστούμε τον αριθμό 2 με το table_name για να πάρουμε τον πρώτο πίνακα από το information_schema.tables που φένεται στην οθόνη. Τώρα πρέπει να προσθέσουμε ένα όριο στο τέλος του ερωτήματος για να μας εμφανίσει όλους τους πίνακες όπως στο παρακάτω παράδειγμα.

http://server/news.php?id=5 union all select 1,table_name,3 from information_schema.tables limit 0,1/*

Πρατηρούμε ότι έχουμε βάλει 0,1 για να πάρουμε ένα αποτέλεσμα ξεκινώντας από το 0. Για να δούμε τον δεύτερο πίνακα, αλλάζουμε το όριο 0,1 στο 1,1 όπως φένεται παρακάτω.

http://server/news.php?id=5 union all select 1,table_name,3 from information_schema.tables limit 1,1/*

Για τον τρίτο πίνακα αντίστοιχα επιλέγουμε όριο 2,1 και συνεχίζουμε μέχρι να βρούμε τις πληροφορίες που χρειαζόμαστε όπως db_admin, poll_user, auth, κλπ. Για να ανακτήσουμε τα ονόματα των στηλών, η μέθοδος είναι ίδια. Εδώ χρησιμοποιούμε "column_name" και "information_schema.columns". Η μέθοδος είναι παρόμοια όπως φένεται και εδώ:

http://server/news.php?id=5 union all select 1,column_name,3 from information_schema.columns limit 0,1/*

Χρησιμοποιώντας το πρόγραμμα αυτό, μπορεί κάποιος να έχει πρόσβαση στην βάση δεδομένων, χρησιμοποιώντας άλλη βάση δεδομένων. Για την

δεύτερη στήλη όπως αντιλαμβανόμαστε θα πρέπει να αλλάξουμε το όριο από 0,1 στο 1,1. Για παράδειγμα:

http://server/news.php?id=5 union all select 1,column_name,3 from information_schema.columns limit 1,1/*

Η δεύτερη στήλη εμφανίζεται οπότε συνεχίζουμε ούτω καθεξής ώστε να βρούμε τις κατάλληλες πληροφορίες που θέλουμε όπως username, user, login, password κλπ.

4ο Κεφάλαιο : Εργαλεία και τεχνολογίες για την εφαρμογή και την αποτροπή επιθέσεων Sql injection

4.1 Εισαγωγή

Σε αυτό το σημείο της εργασίας μας θα γνωρίσουμε μερικά από τα πιο γνωστά εργαλεία καθώς και τεχνολογίες για την εφαρμογή και την αποτροπή επιθέσεων Sql injection. Πιο συγκεκριμένα θα μελετήσουμε τις μεθόδους ανίχνευσης διαφυγής Trojan Horse, Trojan Zebra, White Space Manipulation και Comment Exploitation καθώς και τα εργαλεία : BSQL Hacker, SQLmap, SQLninja, Safe3 SQL Injection, SQLSus και το τερματικό περιβάλλον του Mole.

4.2 Ανίχνευση διαφυγής

Η ανίχνευση επιθέσεων SQL Injection μπορεί να καταστεί εφικτή με τη χρήση κατάλληλων τεχνικών μεθόδων που αποτρέπουν την πρόσβαση. Αυτό μπορούν να το επιτύχουν με την εισαγωγή κατάλληλων λέξεων κλειδιών ή ακόμη και υπογραφών που χαρακτηρίζονται ως κακόβουλες. Οι επιτιθέμενοι κρύβουν το περιεχόμενο του κακόβουλου μηνύματος (το οποίο εισάγουν με διαφορετικούς τρόπους) με απώτερο στόχο την αποφυγή της ανίχνευσης. Αυτό δύναται να επιτευχθεί με κατάλληλες τεχνικές και τεχνολογίες οι οποίες προσπαθούν να σταματήσουν τη διαρροή πληροφοριών. Ένας τέτοιος τρόπος προστασίας έναντι των επιθέσεων αποτελούν οι μηχανές ανίχνευσης διαφυγής. Οι συγκεκριμένες μηχανές έχουν ως σκοπό την απόκρυψη του κακόβουλου κώδικα που προσπαθεί να επιτύχει επίθεση δηλητηρίασης SQL . Οι μηχανές προσπαθούν να ανιχνεύσουν και να αποτρέψουν την παραπάνω ενέργεια.

4.2.1 Trojal Horse

Αν ο επιτιθέμενος χρησιμοποιεί Trojal Horse τότε ένα επικίνδυνο κομμάτι κώδικα κρύβεται μέσα σε μια έγκυρη αίτηση και δύναται να φανερωθεί μόλις γίνει αποδεκτή. Τα IPS και Firewalls αποτελούν συστήματα κατάλληλα για την παρεμπόδιση

απειλών. Τα εν λόγο συστήματα είναι σημαντικά για τον εντοπισμό κρυφών επιθέσεων πριν εκείνες γίνουν αποδεκτές και τους επιτραπεί η πρόσβαση . Με την πάροδο των χρόνων έχουν δημιουργηθεί μεγάλες λίστες οι οποίες αποτελούνται από λέξεις και χαρακτήρες που μπορεί να προκαλέσουν SQL Injection με αποτέλεσμα τη δημιουργία βάσεων δεδομένων υπογραφών. Στις υπογραφές το ταίριασμα των λέξεων χρησιμοποιείται για την παρεμπόδιση περιπτώσεων όπου ο επιτιθέμενος πολλές φορές δύναται να παραβιάσει με απόλυτη επιτυχία τους μηχανισμούς άμυνας της εφαρμογής. Οι λέξεις όπως λόγου χάρη DROP και UNION σε ορισμένες περιπτώσεις εκλαμβάνονται ως πιθανές επιθέσεις με αποτέλεσμα την απόρριψη τέτοιου είδους αιτήσεων. Με την χρήση του Trojal Horse οι επιτιθέμενοι απορρίπτονται από τα Firewalls με αποτέλεσμα να τους αρνείται η πρόσβαση στην εφαρμογή. Συνεπώς θα πρέπει να εντοπίσουν νέο τρόπο εισβολής σε αυτά τα συστήματα . Γι αυτό το λόγο δημιουργήθηκε το Trojan Zebra .

4.2.2 Trojan Zebra

Το Trojan Zebra έχει πολλές ομοιότητες με το Trojan Horse αλλά διαφέρει ως προς τον σχηματισμό. Τα συστήματα παρεμπόδισης απειλών μπερδεύονται από την εμφάνιση του Trojan Zebra και αυτό συμβαίνει για το λόγω του ότι η αναζήτηση αφορά συγκεκριμένες αντιστοιχίες. Με αυτό τον τρόπο γίνεται επιτρεπτή η πρόσβαση στο κέντρο δεδομένων όπου η κρυφή επίθεση πραγματοποιείται μέσα από τις βάσεις δεδομένων. Θα πρέπει να αναφέρουμε πως κέντρο δεδομένων ή αλλιώς Data Center θεωρείται ένας χώρος ο οποίος έχει σχεδιαστεί και υλοποιηθεί για την φιλοξενία υπολογιστών, διακομιστών (server) και συστημάτων δικτύωσης τα οποία μπορεί να χρησιμοποιήσει για να αποθηκεύσει ένας οργανισμός τα δεδομένα του πληροφοριακού του συστήματος.Την σημερινή εποχή οι επιθέσεις SQL Injection επιτυγχάνονται ολοένα και περισσότερο και αυτό υφίσταται για το λόγω του ότι το zebra και το horse έχουν πολλές ομοιότητες. Βέβαια έχουν μια σημαντική διαφορά σε σχέση με τις σημερινές βάσεις δεδομένων υπογραφών , το διαφορετικό πρότυπο. Στο Trojan Zebra οι επιτιθέμενοι χρησιμοποιούν την ευελιξία των διαφόρων παραμέτρων καθώς και την ποικιλία της γλώσσας με απώτερο σκοπό να αποκρύψουν παραδοσιακές επιθέσεις που αναγνωρίζονται από τα συστήματα παρεμπόδισης απειλών.[6]

4.2.3 White Space Manipulation

Οι περισσότερες μηχανές ανίχνευσης επιθέσεων SQL Injection έχουν ως βάση τους υπογραφές και επιπλέον μπορούν να ανιχνεύουν επιθέσεις. Οι επιθέσεις αυτές με τη σειρά τους διακρίνονται για την ποικιλία τους στον αριθμό και στην κωδικοποίηση των κενών τα οποία εμφανίζονται γύρω από τον κακόβουλο κώδικα. Βέβαια οι συγκεκριμένες μηχανές μειονεκτούν στο ότι δεν έχουν τη δυνατότητα να πραγματοποιήσουν την ανίχνευση κενών ανάμεσα στον ίδιο κώδικα. Τόσο το πρότυπο αντιστοιχίας όσο και η μέθοδος υπογραφών αναζητούν για ταίριασμα κειμένου που αποτελείται από ένα ή και περισσότερα κενά. Αποτυγχάνουν να ανιχνεύσουν το ίδιο ταίριασμα κειμένου αν δεν περιέχονται κενά μέσα στον κώδικα.

Το White Space Manipulation περιλαμβάνει τη διαγραφή των κενών ανάμεσα σε λέξεις σε μια αίτηση. Οι παρακάτω ενέργειες χρησιμοποιούνται με απώτερο σκοπό να μπερδέψουν την αντιστοιχία ανάμεσα στις λέξεις των συστημάτων ανίχνευσης:

- Tab
- Carriage return
- Line feed

Θα πρέπει να προσθέσουμε ότι η εν λόγο επίθεση λειτουργεί και αυτό συμβαίνει διότι η μηχανή ανάλυσης SQL των βάσεων δεδομένων δεν ενδιαφέρεται ούτε για τα κενά ούτε για τη διαμόρφωση των χαρακτήρων .

4.2.4 Comment Exploitation

Παλαιότερα οι επιτιθέμενοι έκαναν χρήση της διπλής κοινής ενωτικής σύνταξης όπως λόγου χάρη το '- -' για κάλυψη των ενεργειών τους. Ακόμη και σήμερα ορισμένες από τις μηχανές αναγνωρίζουν τις επιθέσεις με αποτέλεσμα να αλλάζουν την τακτική τους οι επιτιθέμενοι. Κατά συνέπεια ακολουθούν τη χρήση της σύνταξης σχολίου η οποία υποστηρίζεται από τη γλώσσα προγραμματισμού C /**/ για να παρακαμφθεί η ανίχνευση. Επίσης θα πρέπει να τονίσουμε ότι οι επιτιθέμενοι χρησιμοποιούν τον διαχωρισμό των εντολών που χρησιμοποιούνται μαζί με τις επιθέσεις . Η αναγνώριση του μπορεί να επιτευχθεί μέσω της αντιστοιχίας υπογραφών και σε ορισμένες περιπτώσεις διαχωρίζουν τις λέξεις κλειδιά. Χαρακτηριστικό παράδειγμα αποτελεί η εντολή UNION η οποία χρησιμοποιείται ώστε να προκληθεί SQL Injection. Η εν λόγο λέξη αποτελεί για τις μηχανές ανίχνευσης υπογραφών μια μόνιμη απειλή όσο

δεν χρησιμοποιεί κάποια λέξη κλειδί ή κάποιο κενό. Σε αυτή την περίπτωση για να αποφύγουν οι επιτιθέμενοι την ανίχνευση κάνουν χρήση δεικτών σχολίων ης γλώσσας C . Συνεπώς η λέξη UNION θα μετατραπεί σε UN /**/ION κατά συνέπεια θα παρακαμφθεί η ασφάλεια. Η συγκεκριμένη επίθεση επιτυγχάνεται για το λόγω του ότι μια πληθώρα μηχανών οι οποίες έχουν ως βασικό τους μέλημα την επεξεργασία SQL κώδικα σε βάσεις δεδομένων αφαιρούν τα σχόλια που προκύπτουν κατά την ανάγνωση πολύ πριν ξεκινήσει η διαδικασία επεξεργασίας statement[6].

4.2.5 Τρόποι για την αποφυγή επιθέσεων SQL Injection σε PHP

Μερικά βήματα που μπορούμε να ακολουθήσουμε έτσι ώστε να έχει η εφαρμογή μας προστασία κατά των επιθέσων SQL είναι:

Η εφαρμογή ελέγχου παραμέτρων σε όλες τις εφαρμογές. Για παράδειγμα αν ζητάμε από τον χρήστη να δώσει στην είσοδο έναν αριθμό πελάτη πρέπει να βεβαιωθούμε ότι η είσοδος είναι ένα νούμερο πριν τρέξει το SQL ερώτημα. Μπορούμε να περιορίσουμε τα δικαιώματα του λογαριασμού που εκτελεί ερωτήματα SQL. Αν ο λογαριασμός εκτελούσε ερωτήματα που δεν έχει άδεια, δεν θα επιτύχει. Τέλος, χρησιμοποιώντας διαδικασίες και τεχνικές που έχουμε αποθηκεύσει για να αποτρέψουμε την SQL Injection.

Οι επιθέσεις SQL Injection αποτελούν κάτι πολύ συνηθισμένο στον ιστό. Οι χάκερ χρησιμοποιούν κακόβουλα ερωτήματα SQL με απώτερο σκοπό να ανακτήσουν δεδομένα απευθείας από μια βάση δεδομένων. Ουσιαστικά προσπαθούν να εκμεταλλευτούν μια ευπάθεια ασφάλειας. Επίσης είναι πιο συνηθισμένες σε εφαρμογές όπου οι εισροές δεν φιλτράρονται σωστά. Οι τέσσερις αυτοί τρόποι είναι:

- ***Οι προετοιμασμένες δηλώσεις***
- ***Η αποφυγή Strings***
- ***Η χρήση trim() και strip_tags()***
- ***Χρήση PDO***

Οι προετοιμασμένες δηλώσεις: ο πιο εύκολος τρόπος να αποτρέψουμε προσβολές SQL Injection σε PHP με τη χρήση ‘Prepared Statements’ .

Η αποφυγή Strings: η αποφυγή χρήσης συμβολοσειράς βοηθάει στην αφαίρεση ειδικών χαρακτήρων για χρήση σε δηλώσεις SQL.

Η χρήση trim() και strip_tags(): είναι οι συμβατικοί τρόποι με τους οποίους μπορείτε να φιλτράρετε την είσοδο. Με τη χρήση και των δύο μαζί μπορούμε να αφαιρέσουμε πρόσθετους κώδικες που χρησιμοποιούνται γενικά από του χάκερς.

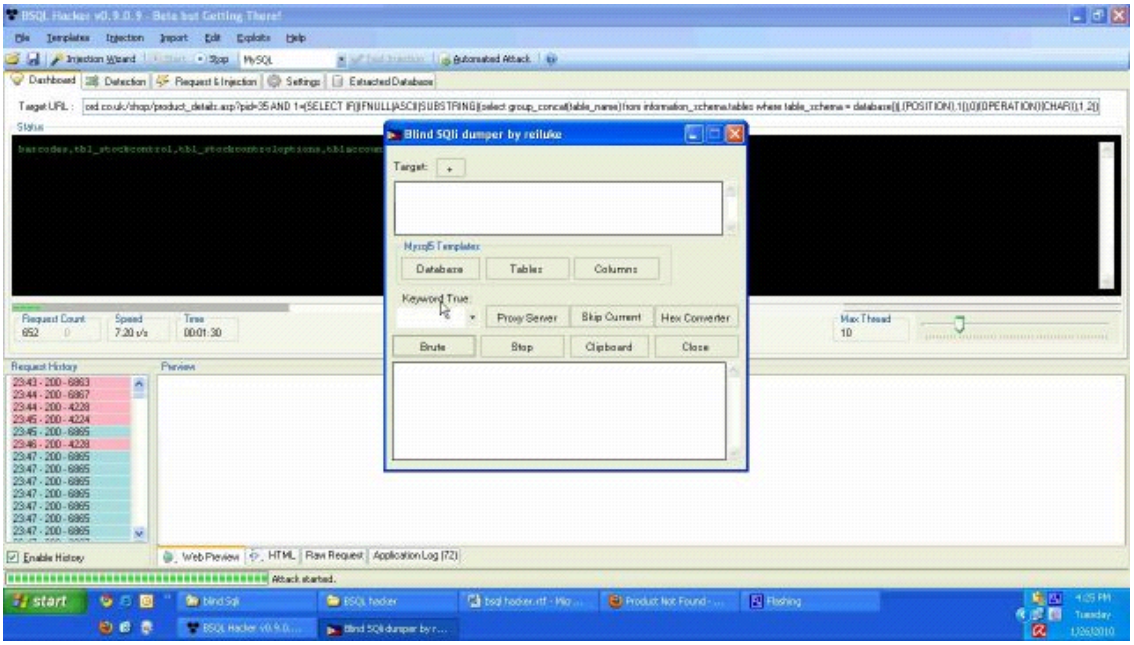
Χρήση PDO: τα PDO ή τα αντικείμενα δεδομένων PHP είναι πολύ χρήσιμα ίσως και τα πιο αποτελεσματικά στην πρόληψη των προσβολών SQL Injection.

4.3 SQL Injection tools

Στο υποκεφάλαιο αυτό της παρούσας πτυχιακής εργασίας θα γνωρίσουμε έξι από τα πιο γνωστά εργαλείο SQL Injection .

4.3.1 BSQL Hacker

Το BSQL Hacker αποτελεί ένα από τα πιο ωραία εργαλεία SQL Injection. Το εν λόγο εργαλείο βοηθά πάρα πολύ στο να αντιμετωπίσετε μια επίθεση ενάντια σε μια εφαρμογή ιστού. Το συγκεκριμένο εργαλείο διακρίνεται για τη γρηγοράδα του και την επίθεση με χρήση νημάτων για βέλτιστα αποτελέσματα.



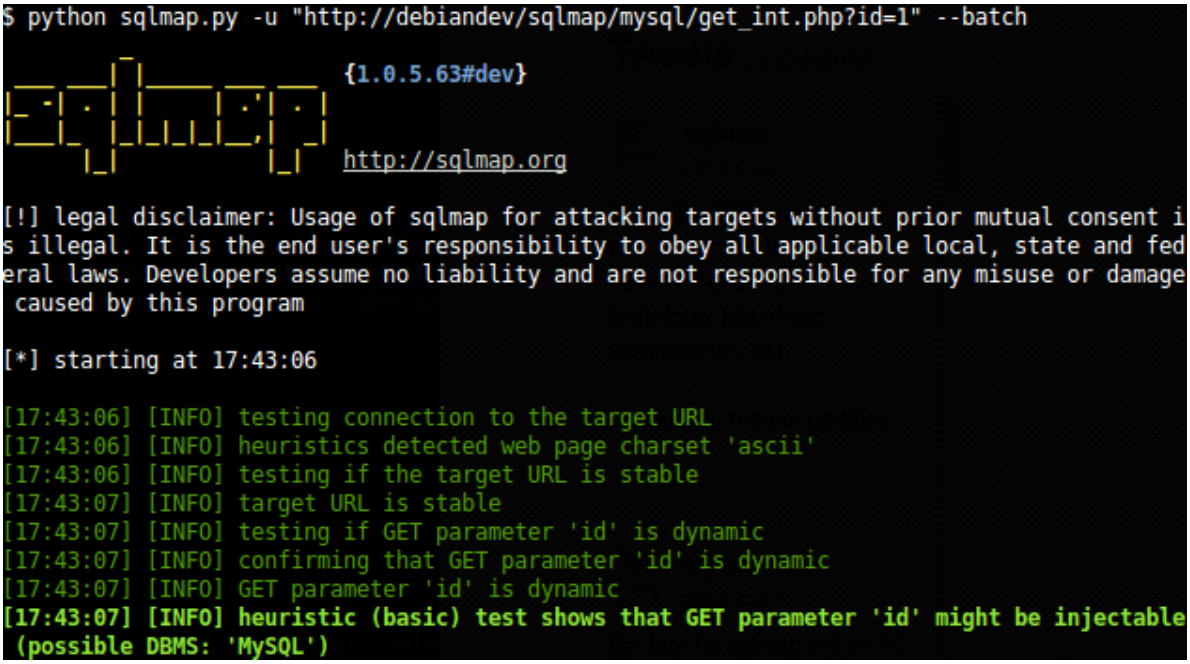
Εικόνα 3.1: BSQL Hacker

Υποστηρίζει τέσσερα διαφορετικά είδη επιθέσεων SQL Injection:

- Blind SQL Injection
- Time Based Blind SQL Injection
- Deep Blind
- SQL Injection Error Based SQL Injection

4.3.2 SQLmap

Το SQLmap αποτελεί ένα πολύ σημαντικό εργαλείο έγχυσης SQL το οποίο είναι ανοικτού κώδικα άρα διατίθεται δωρεάν. Θα πρέπει να αναφέρουμε ότι το συγκεκριμένο εργαλείο είναι εξαιρετικά εύκολο στο να εκμεταλλευτεί την ευπάθεια SQL Injection σε μια εφαρμογή web και στο διακομιστή της βάσης δεδομένων.



Εικόνα 3.2: **SQLmap**

Αποτελείται από μια πολύ ισχυρή μηχανή ανίχνευσης η οποία δύναται να ανιχνεύσει τα τρωτά σημεία που σχετίζονται με τα SQL Injection. Μερικές από τις βάσεις δεδομένων που περιλαμβάνει είναι οι :

- MySQL
- Oracle
- PostgreSQL
- Microsoft SQL Server
- Microsoft Access
- Sybase
- SAP MaxDB

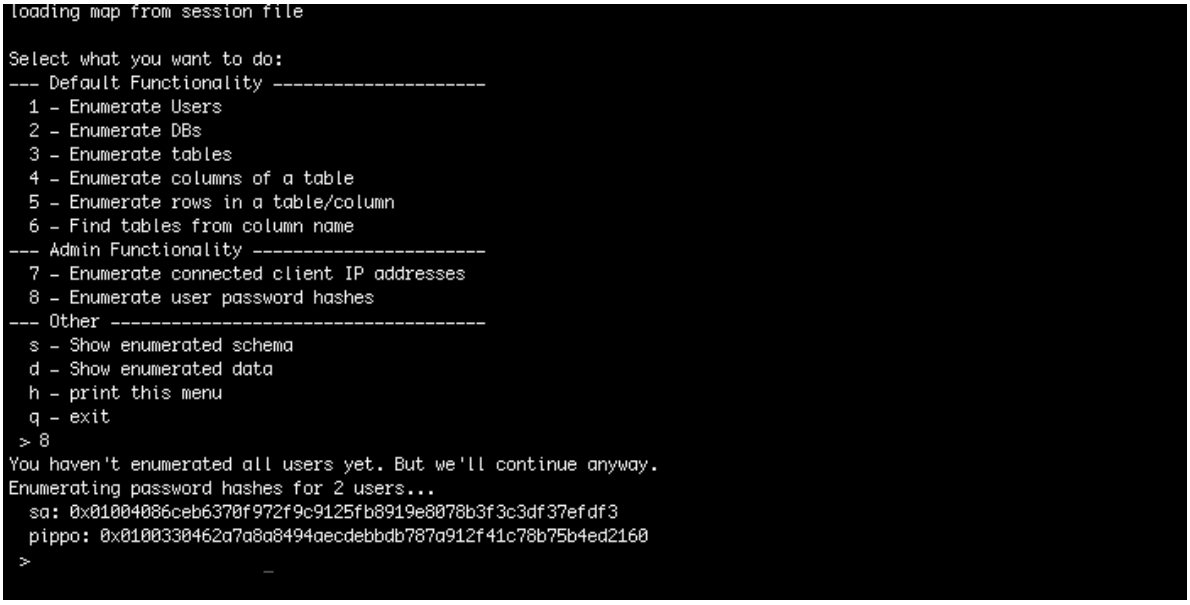
Μόλις συνδεθείτε με έναν από τους παραπάνω διακομιστές βάσης δεδομένων το συγκεκριμένο εργαλείο σας δίνει τη δυνατότητα να αναζητήσετε:

- Συγκεκριμένο όνομα της βάσης δεδομένων
- Συγκεκριμένους πίνακες
- Συγκεκριμένες στήλες

Κλείνοντας θα προσθέσουμε ότι ο διακομηστής βάσεων δεδομένων είναι εξαιρετικά μεγάλος και μπορεί να περιέχει πολλές βάσεις και πίνακες.

4.3.3 SQLninja

Το SQLninja είναι και αυτό ένα από τα πολύ σημαντικά εργαλεία του SQL Injection το οποίο εκμεταλλεύεται εφαρμογές ιστού που χρησιμοποιούν διακομιστή SQL ως διακομιστή βάσης δεδομένων . Το εν λόγο εργαλείο δύναται να μην μπορεί να βρει πρώτα τον τόπο της έγχυσης. Αλλά μόλις ανακαλυφθεί μπορεί εύκολα να αυτοματοποιήσει τη διαδικασία της εκμετάλλευσης και να εξαγάγει τις πληροφορίες από το διακομιστή βάσης δεδομένων.



Εικόνα 3.3: SQLninja

Το εν λόγο εργαλείο έχει τη δυνατότητα να προσθέσει απομακρυσμένες λήψεις στο μητρώο του OS Server της βάσης δεδομένων αντί να απενεργοποιήσει την πρόληψη εκμετάλλευσης δεδομένων . ο βασικότερος στόχος του είναι το να μπορεί να επιτρέψει στον εισβολέα να αποκτήσει απομακρυσμένη πρόσβασης σε ένα διακομιστή βάσης δεδομένων SQL. Ολοκληρώνοντας την αναφορά μας θα προσθέσουμε ότι το συγκεκριμένο εργαλείο είναι διαθέσιμο για τα παρακάτω λειτουργικά συστήματα:

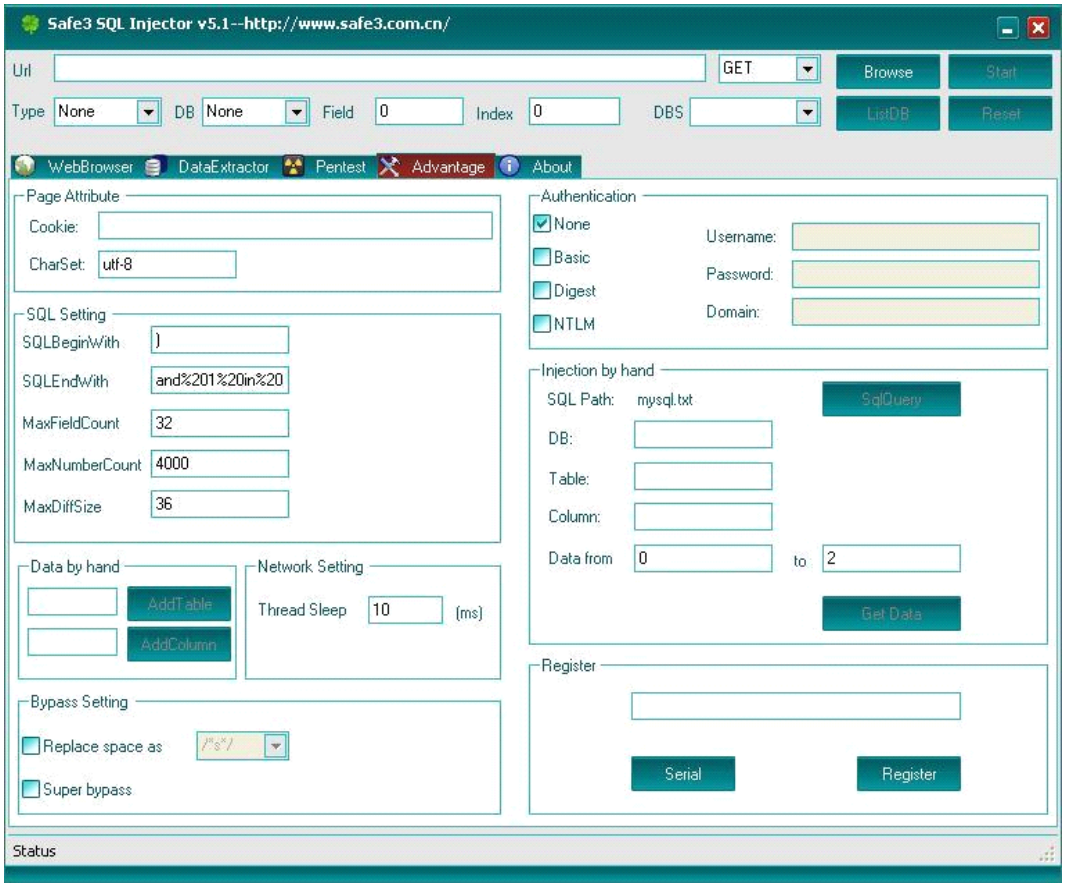
- Windows
- Linux
- FreeBSD
- Mac OS X
- iOS

4.3.4 Safe3 SQL Injection

Το Safe3 SQL Injection αποτελεί ένα εργαλείο τόσο εύχρηστο και ισχυρό. Το εν λόγο εργαλείο καθιστά αυτόματη τη διαδικασία SQL Injection το οποίο βοηθά τους

επιτιθέμενους να αποκτήσουν πρόσβαση σε έναν απομακρυσμένο διακομιστή SQL εκμεταλλευόμενοι την ευπάθεια SQL Injection. Δε θα μπορούσαμε να παραλείψουμε ότι το Safe3 SQL Injection έχει στη διάθεση του ένα σύστημα AI το οποίο αναγνωρίζει :

- Τον διακομιστή βάσης δεδομένων
- Τον τύπο έγχυσης
- Τον καλύτερο τρόπο ευπάθειας



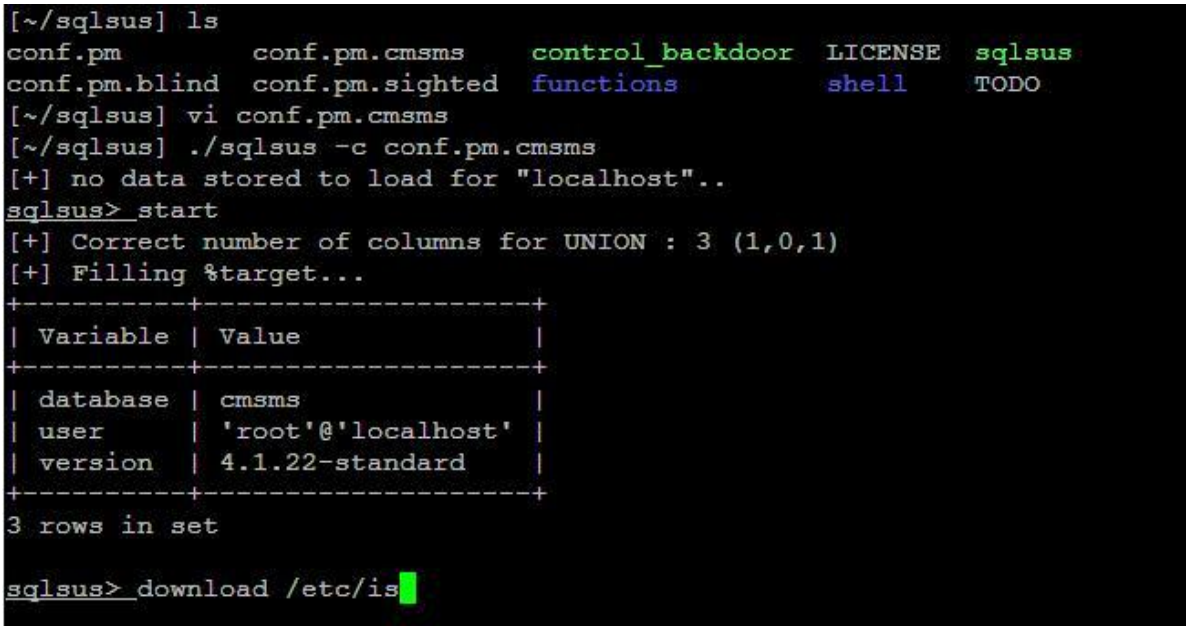
Εικόνα 3.4: Safe3 SQL Injection

Επιπλέον υποστηρίζει τα συστήματα διαχείρισης βάσεων δεδομένων:

- MySQL
- PostgreSQL
- Microsoft SQL Server
- Microsoft Access
- SQLLite
- Firebird
- Sybase
- Sap MaxDB

4.3.5 SQLSus

Το SQLSus είναι και αυτό ακόμη ένα πολύ σημαντικό εργαλείο έγχυσης SQL το οποίο είναι ανοικτού κώδικα. Ουσιαστικά αποτελεί ένα εργαλείο τόσο έγχυσης όσο και εξαγοράς MySQL. Το εν λόγο εργαλείο είναι γραμμένο σε Perl και έχετε τη δυνατότητα να επεκτείνετε τις λειτουργίες προσθέτοντας δικούς σας κώδικες. Διακρίνεται για τη γρηγοράδα και την αποδοτικότητα του. Επιπλέον χρησιμοποιεί έναν πολύ ισχυρό αλγόριθμό επίθεσης ο οποίος μεγιστοποιεί τα δεδομένα που χρησιμοποιήθηκαν. Προσφέρει μια διεπαφή εντολών που επιτρέπει να κάνετε τις δικές σας ερωτήσεις SQL και να πραγματοποιείτε επιθέσεις SQL Injection.



Εικόνα 3.5: SQLSus

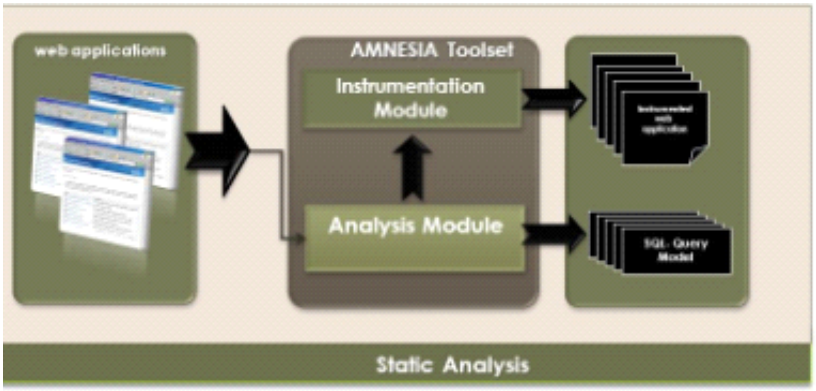
4.3.6 Mole

Το τερματικό περιβάλλον του Mole αποτελεί ένα εργαλείο αυτόματης εισαγωγής SQL το οποίο διατίθεται δωρεάν. Θα πρέπει αρχικά να βρείτε την ευάλωτη διεύθυνση URL και έπειτα να την μεταβιβάσετε στο εργαλείο. Το συγκεκριμένο εργαλείο μπορεί να ανιχνεύσει την ευπάθεια από τη δεδομένη διεύθυνση URL με τη χρήση τεχνικών ερωτήματος βασισμένες σε Boolean. Επιπλέον προσφέρει μια διεπαφή γραμμής εντολών η οποία είναι ιδιαίτερα εύκολη και απλή. Επιπροσθέτως παρέχει αυτόματη συμπλήρωση .[8]

Εικόνα 3.6: Το τερματικό περιβάλλον του Mole

4.4 AMNESIA

Το AMNESIA αποτελεί ένα εξαιρετικά χρήσιμο εργαλείο το οποίο έχει ως βάση μοντέλα που συνδυάζουν στατιστική ανάλυση και έλεγχο κατά το χρόνο εκτέλεσης.

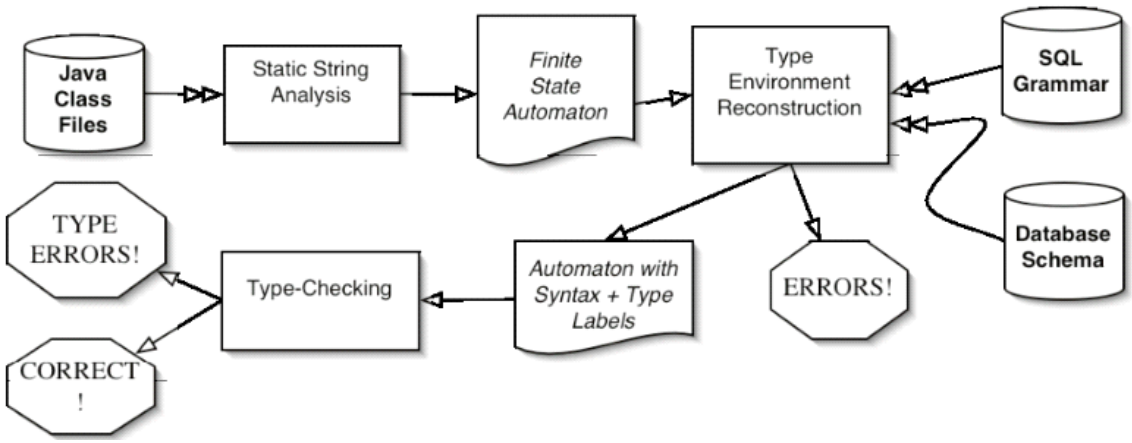


Εικόνα 3.7:AMNESIA

Αρχικά το AMNESIA χρησιμοποιεί στατιστική ανάλυση για να χτίσει τα μοντέλα των διαφορετικών τύπων ερωτημάτων τα οποία μπορεί μια εφαρμογή να παράγει με νόμιμο τρόπο σε κάθε σημείο πρόσβασης δεδομένων. Στη συνέχεια ελέγχει κάθε ερώτημα έναντι στα στατιστικά μοντέλα πριν σταλεί στη βάση δεδομένων. Όσον αφορά τα ερωτήματα τα οποία παραβιάζουν το μοντέλο προσδιορίζονται ως επιθέσεις έγχυσης και αποτρέπεται η εκτέλεση τους στη βάση δεδομένων. Η εν λόγο τεχνική αποδίδει βέλτιστα ενάντια σε επιθέσεις SQL έγχυσης. Το βασικότερο μειονέκτημα της είναι η τροποποίηση της αρχικής εφαρμογής ιστού με την ενσωμάτωση ελέγχων πριν από κάθε κλήση στη βάση δεδομένων. Βέβαια αξίζει να προσθέσουμε ότι η επιτυχία της τεχνικής εξαρτάται από την ακρίβεια της στατιστικής ανάλυσης για την οικοδόμηση των μοντέλων ερωτημάτων.[12]

4.5 Ο JDBC- Checker

Ο JDBC- Checker αποτελεί μια πολύ σημαντική τεχνική η οποία έχει ως βασικό σκοπό τον έλεγχο στατιστικών καθώς και την ορθότητα των δυναμικών παραγόμενων SQL ερωτημάτων.

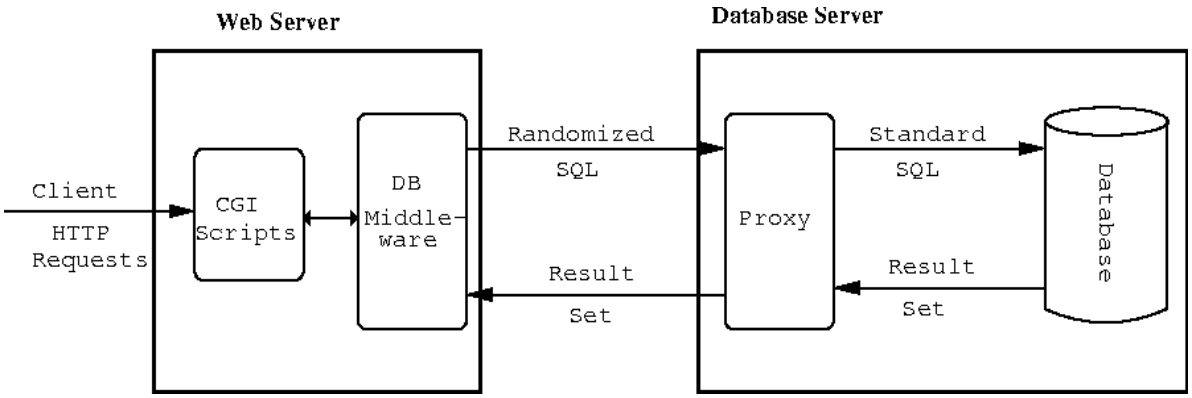


Εικόνα 3.8: Ο έλεγχος που παρέχει στη βάση δεδομένων ο JDBC- Checker

Αρχικά η εν λόγο τεχνική δεν χρησιμοποιήθηκε για την ανίχνευση γενικών επιθέσεων SQL έγχυσης παρόλα αυτά έχει την ικανότητα να μπορεί να αποτρέπει επιθέσεις που εκμεταλλεύονται τους κακούς συνδυασμούς τύπων σε ένα δυναμικά παραγόμενο ερώτημα. Ο JDBC- Checker μπορεί να ανιχνεύσει την αιτία ευπαθειών SQL έγχυσης στον κώδικα . Το βασικότερο μειονέκτημα ατής της τεχνικής είναι η περιορισμένη εμβέλεια στην ανίχνευση μόνο των ταυτολογιών και δεν μπορεί να ανιχνεύσει άλλους τύπους επιθέσεων.[13]

4.6 Το SQLrand

Το SQLrand είναι μια αρχιτεκτονική ανάπτυξης εφαρμογών σε sql. Παρέχει ένα πλαίσιο το οποίο επιτρέπει στους μηχανισμούς ανάπτυξης των εφαρμογών να δημιουργήσουν ερωτήματα SQL χρησιμοποιώντας τυχαίες εντολές στη θέση των κανονικών λέξεων κλειδιών της γλώσσας SQL.



Εικόνα 3.9: Το SQLrand

Πιο συγκεκριμένα ένα φίλτρο διακόπτει τα ερωτήματα που κατευθύνονται προς τη βάση δεδομένων και αποκωδικοποιεί τις λέξεις κλειδιά. Ο SQL κώδικας ο οποίος εγχέεται από ένα επιτιθέμενο δε θα έχει κατασκευαστεί χρησιμοποιώντας το τυχαίο σύνολο εντολών. Κατά συνέπεια οι εγχεόμενες εντολές θα οδηγήσουν σε λανθασμένο ερώτημα. Παρόλο που η εν λόγο τεχνική μπορεί να χαρακτηριστική εξαιρετική έχει πολλά μειονεκτήματα. Παρόλο που χρησιμοποιεί μυστικό κλειδί για την ταυτοποίηση των εντολών , η ασφάλεια της προσέγγισης εξαρτάται από το πόσο οι επιτιθέμενοι είναι σε θέση να ανακαλύψουν το κλειδί. Η προσέγγιση επιβάλει μια πρόσθετη υποδομή αφού απαιτεί την ενσωμάτωση ενός πληρεξούσιου για τη βάση δεδομένων στο σύστημα.[14]

5^{o} Κεφάλαιο :Υλοποίηση εφαρμογής

5.1 Τι χρησιμοποιήσαμε στην εφαρμογή

Στην εφαρμογή που υλοποιήσαμε χρησιμοποιήσαμε κάποια πολύ σημαντικά εργαλεία. Χρησιμοποιήσαμε ένα Php Interface για να φτιάξουμε μια βάση δεδομένων που θα είναι προσβάσιμη από το διαδίκτυο. Αυτή η βάση είναι συνδεδεμένη σε έναν Php Server. Χρησιμοποιήσαμε τον EasyPHP Devserver και Webserver όπου θα μιλήσουμε για αυτά παρακάτω. Αυτό σημαίνει πως μπορούμε να φιλοξενήσουμε οτιδήποτε απευθείας στον υπολογιστή μας και να το μοιραστούμε στο διαδίκτυο. Στην συνέχεια, ξεκινώντας την δημιουργία του κώδικα, χρησιμοποιήσαμε την γλώσσα προγραμματισμού Php και MySql. Τέλος χρησιμοποιήσαμε το phpmyadmin το οποίο διαχερίζεται τη MySql μέσω του Διαδικτύου ενώ έχουμε ακόμα και τη δυνατότητα να εκτελέσουμε άμεσα οποιαδήποτε δήλωση SQL. Στη συνέχεια θα κάνουμε μια σύντομη αναφορά σε καθένα από τα παραπάνω ώστε να κατανοήσει και ο μέσος αναγνώστης την εφαρμογή που υλοποιήσαμε.

5.1.1Η γλώσσα προγραμματισμού PHP

Η γλώσσα προγραμματισμού PHP (βλ. Hypertext Pre Processor) είναι μια γλώσσα η οποία ασχολείται με τη συγγραφή σεναρίων. Η συγκεκριμένη γλώσσα έχει την ιδιότητα να ενσωματώνεται μέσα σε κώδικα της γλώσσας HTML και επιπροσθέτως την ιδιότητα να εκτελείται από την πλευρά του server.

Εικόνα 4.1 : PHP

Η γλώσσα προγραμματισμού PHP έχει δανειστεί από τις γλώσσες:

- C
- Java
- Perl

Αλλά επιπλέον διαθέτει και μερικά δικά του και μοναδικά χαρακτηριστικά. Ο βασικός στόχος της PHP είναι να δώσει τις δυνατότητες στους web developers να δημιουργήσουν δυναμικές σελίδες.

Η γλώσσα PHP έχει πολλές δυνατότητες όπως και άλλα προγράμματα της τεχνολογίας CGI όπως:

- Η επεξεργασία των δεδομένων μιας φόρμας
- Η δημιουργία δυναμικού περιεχομένου ιστοσελίδων
- Η αποστολή cookies
- Η λήψη cookies

Παρόλο που οι δυνατότητες της γλώσσας PHP είναι πολλές αυτή που ξεχωρίζει είναι η χρήση της στις βάσεις δεδομένων. Μερικές από τις βάσεις δεδομένων τις οποίες υποστηρίζει η γλώσσα PHP είναι:

- MySQL
- Oracle
- PostgreSQL
- Solid
- Sybase
- Velocis
- Unix dbm[9]

5.1.2 Η γλώσσα MySQL

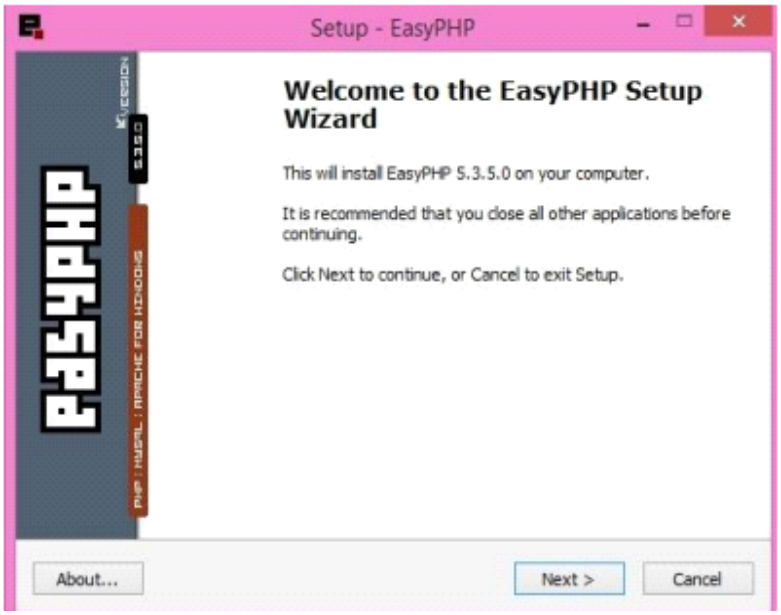
Η MySQL αποτελεί ένα λογισμικό ανοικτού κώδικα το οποίο έχει να κάνει με τη διαχείριση βάσεων δεδομένων. Παρέχει πολλές δυνατότητες στους χρήστες όπως :

- Η αποθήκευση δεδομένων
- Η οργάνωση δεδομένων[10]

Εικόνα 4.2 : MySQL

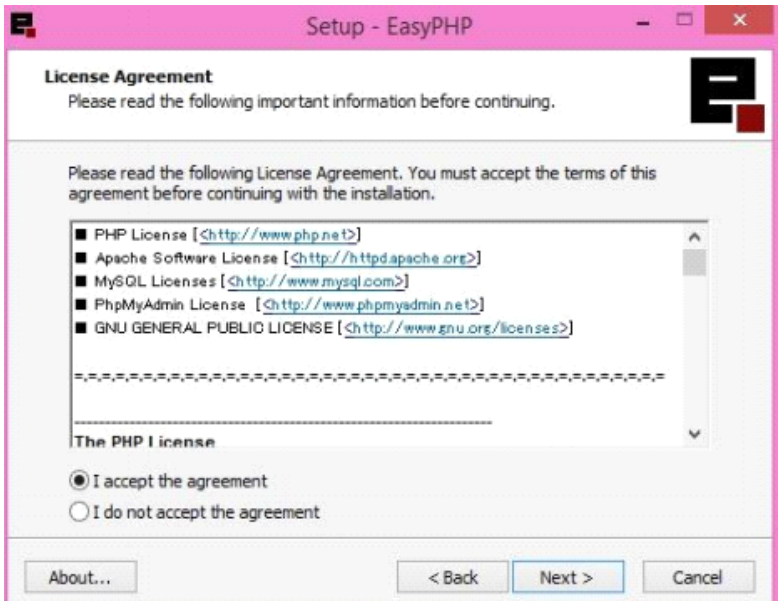
5.1.3 Ο εξυπηρετητής-server EasyPHP.

Στην εργασία μας χρησιμοποιήσαμε τον EasyPHP δηλαδή έναν server που περιέχει την PHP που μπορεί να υποστηρίξει την SQL. Ο server έχει τη δυνατότητα να είναι συνδεδεμένος στον υπολογιστή σε ένα εικονίδιο localhost ή στο internet.Ο Devserver εγκαθιστά ένα πλήρες και έτοιμο περιβάλλον ανάπτυξης κώδικα, και ο Webserver μετατρέπει τον υπολογιστή μας σε έναέτοιμο προς χρήση προσωπικό web hosting server. Στις επόμενες είκονες παρατίθενται τα απλά βήματα εγκατάστασης του EasyPHP. Επιλέγοντας "Next" ξεκιναεί η διαδικασία πριν την εγκατάσταση



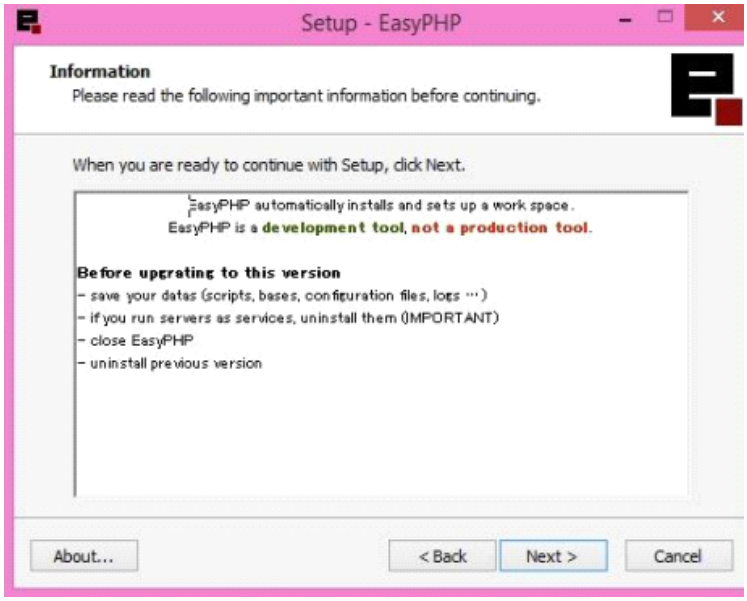
Στο επόμενο βήμα αφότου διαβάσουμε και κατανοήσουμε τους όρους χρήσης επιλέγουμε αποδοχή των όρων αυτών και στη συνέχεια "Next".

Εικόνα 4.3 : Εγκατάσταση του EasyPHP



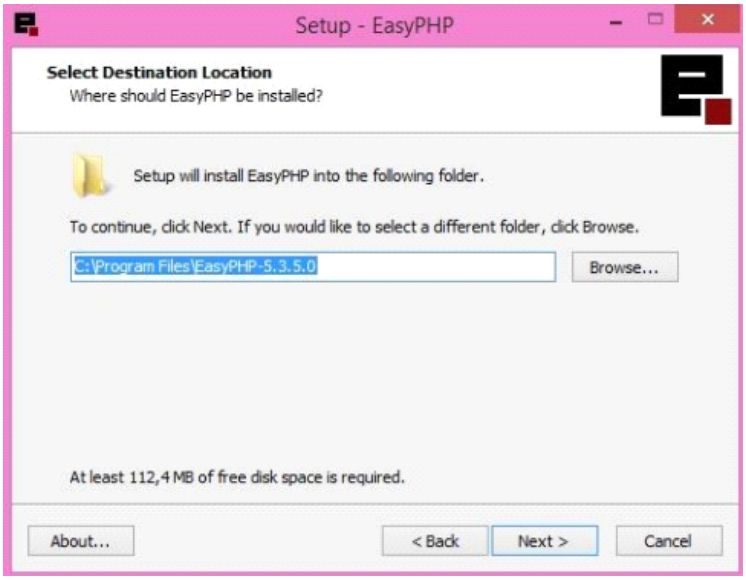
Όπως και προηγουμένως έτσι και τώρα προσέχουμε τις απαραίτητες πληροφορίες που μας δίνονται έτσι ώστε η εγκατάσταση να είναι επιτυχής. Συνεχίζουμε με "Next".

Εικόνα 4.4 : Εγκατάσταση του EasyPHP



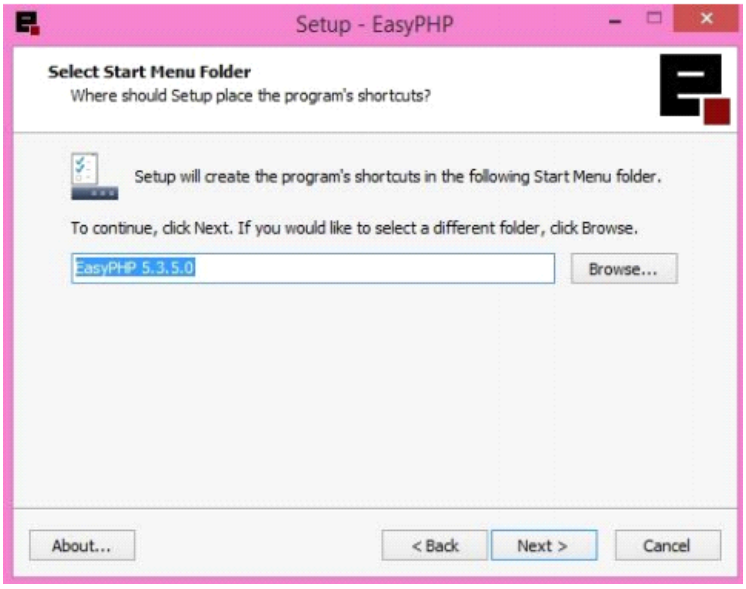
Εικόνα 4.5 : Εγκατάσταση του EasyPHP

Όπως είναι γνωστό για κάθε εγκατάσταση προγράμματος, θα πρέπει να επιλέξουμε την τοποθεσία που θέλουμε να αποθηκευτεί. Παρομοίως την επιλέγουμε και στη συνέχεια ξανά "Next".



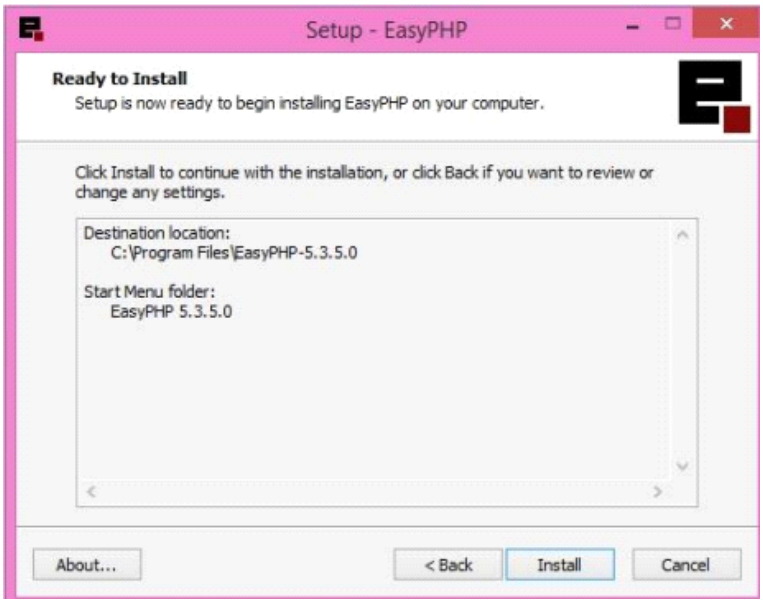
Εικόνα 4.6 : Εγκατάσταση του EasyPHP

Εδώ έχω το δικαίωμα να εμφανίζεται το εικόνιδιο σαν συντόμευση στην επιφάνεια εργασίας, επιλέγουμε "Next".



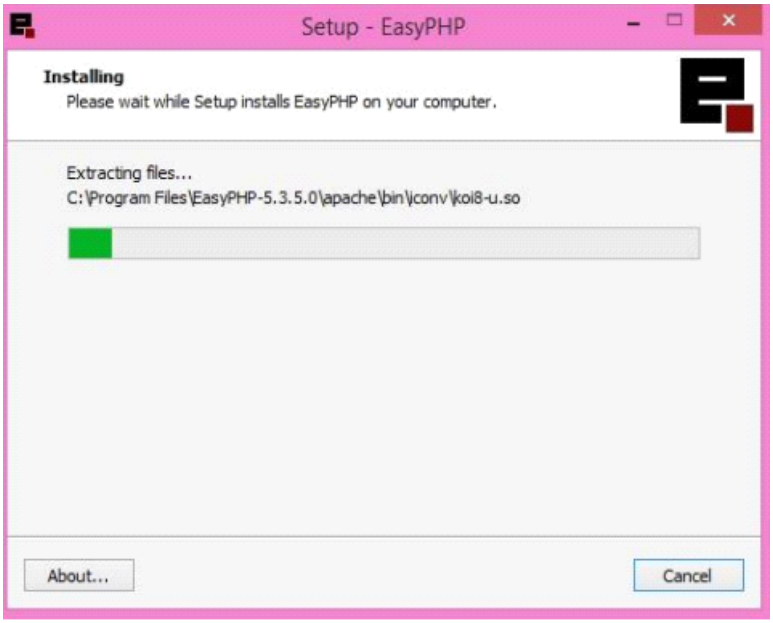
Φτάσαμε στο τελικό στάδιο πριν την εγκατάσταση. Μπορούμε να επιλέξουμε "back" για να πάμε σε προηγούμενες δραστηριότητες και να τις τροποποιήσουμε, αλλίως "Install" για εγκατάσταση.

Εικόνα 4.7 : Εγκατάσταση του EasyPHP



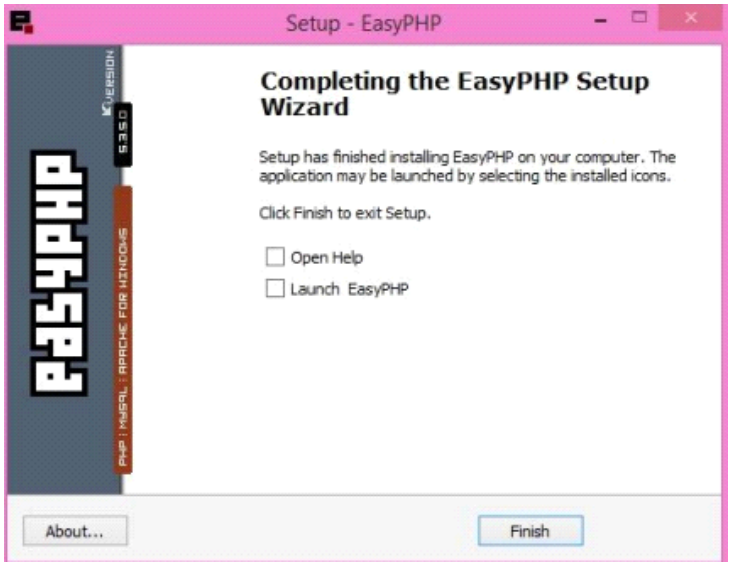
Αναμένουμε να ολοκληρωθεί η εγκατάσταση.

Εικόνα 4.8 : Εγκατάσταση του EasyPHP



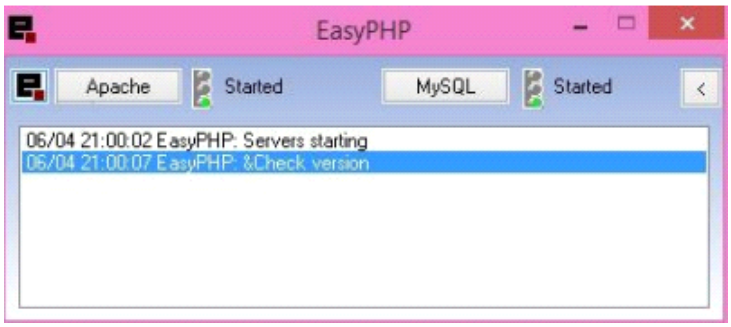
Και επιλέγουμε "Finish".

Εικόνα 4.9 : Εγκατάσταση του EasyPHP

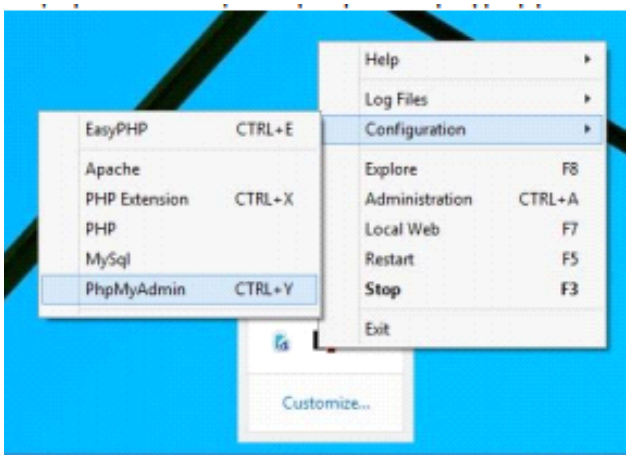


Όπως αναφέραμε στην αρχή του κεφαλαίου το EasyPHP αποτελείται από τον server Apache και την Mysql. Για την λειτουργία του θα πρέπει να είναι ενεργοποιημένα και αυτό διακρίνεται από την πράσινη σηματοδότηση.

Εικόνα 4.10 : Εγκατάσταση του EasyPHP



Εικόνα 4.11 : Εγκατάσταση του EasyPHP



Εικόνα 4.12 : Εγκατάσταση του EasyPHP

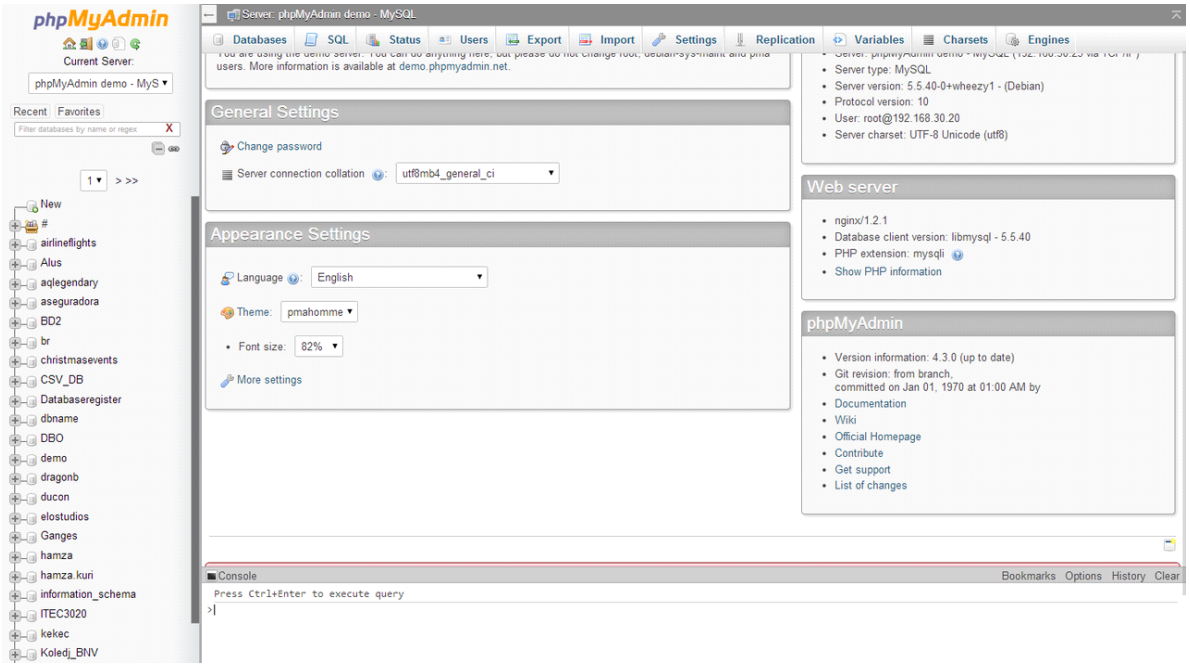
5.1.4 Το εργαλείο phpMyAdmin.

Το Phpmyadmin είναι ένα εργαλείο ελεύθερου λογισμικού γραμμένο σε PHP και αποτελεί ένα περιβάλλον από το οποίο μπορούμε να διαχειριστούμε εύκολα, γρήγορα και εξ αποστάσεως πρόσβαση σε οποιαδήποτε βάση δεδομένων τύπου sql. Το εργαλείο αυτό χρειαστήκαμε για να στήσουμε την βάση δεδομένων και να την διαχειριστούμε.

Αφού έχουμε εγκαταστήσει το Wamp server πηγαίνουμε στο Firefox πληκτρολογούμε localhost και στη συνέχεια πατάμε phpmyadmin όπως βλέπουμε στην εικόνα.

Με τη χρήση του μπορούμε να διαχειριστούμε βάσεις δεδομένων. Ουσιαστικά είναι μια εφαρμογή . Μερικές από τις λειτουργίες που παρέχει είναι :

- η δημιουργία, η διαγραφή και η επεξεργασία βάσεων δεδομένων
- Η δημιουργία, η διαγραφή και η επεξεργασία πινάκων
- Η δημιουργία, η διαγραφή και η επεξεργασία γραμμών
- Η δημιουργία, η διαγραφή και η επεξεργασία στηλών
- Η εκτέλεση ερωτημάτων SQL
- Η διαχείριση χρηστών



Εικόνα 4.13 : Phpmyadmin

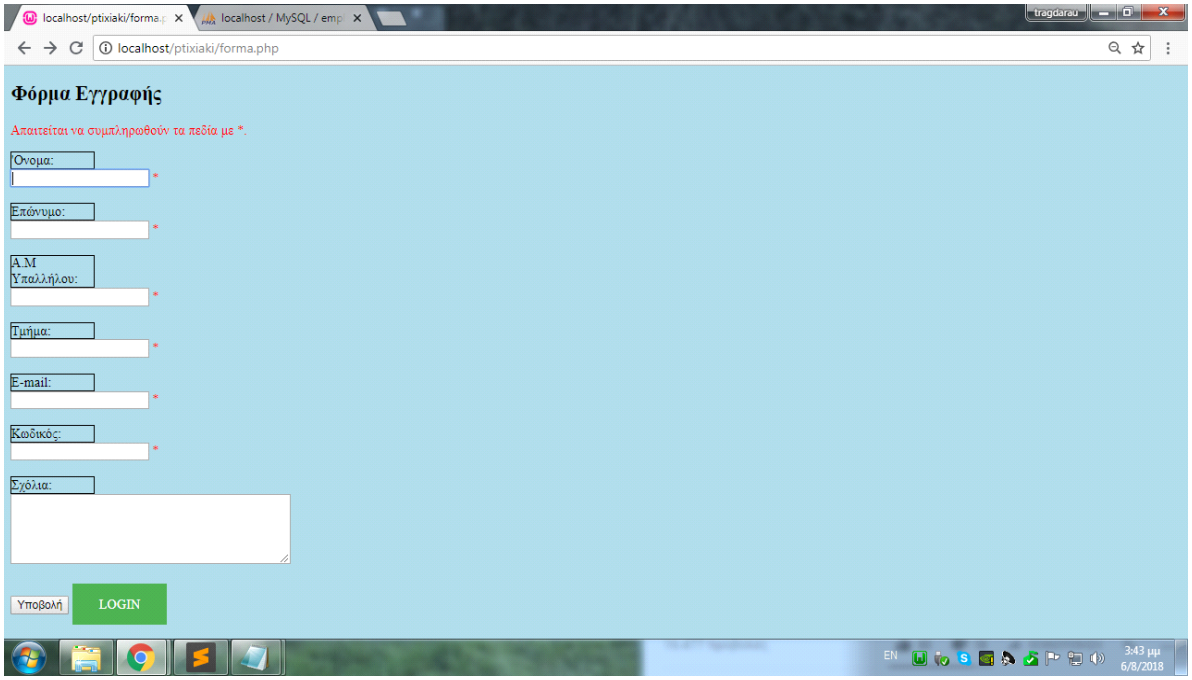
5.2Η υλοποίηση της εφαρμογής μας

Για να κατανοήσουμε την επίθεση θα πρέπει αρχικά να κατανοήσουμε τον τρόπο με τον οποίο λειτουργούν οι διαδικτυακές εφαρμογές. Οι διαδικτυακές εφαρμογές φιλοξενούνται διαμέσου ενός web server , ενός application server και ενός database server. Από τον browser όλα όσα απαιτούνται για να τρέξουν την εφαρμογή κωδικοποιούνται σε HTTP και στέλνονται στον εξυπηρετητή. Ό ρόλος του οποίου είναι η ανάλυση την πληροφορίας την οποία έχει λάβει και η επιστροφή των αντίστοιχων δεδομένων στην εφαρμογή. Η πληροφορία στέλνεται με μορφή κειμένου στον server και πολλές φορές καθιστά εφικτή και την αποστολή τόσο κακόβουλών όσο και εσφαλμένων πληροφοριών .

Για την ανάπτυξη της εφαρμογής χρησιμοποιήσαμε την γλώσσα προγραμματισμού html η οποία είναι η κύρια γλώσσα για τις ιστοσελίδες και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Επίσης χρησιμοποιήσαμε την γλώσσα CSS. Η CSS χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί σε μια γλώσσα σήμανσης όπως η Html. Η Php που χρησιμοποιήθηκε μας βοήθησε με δυναμικό περιεχόμενο έτσι ώστε να λειτουργεί η ιστοσελίδα. Τέλος

χρησιμοποιήσαμε sql που όπως γνωρίζουμε χρησιμοποιείται στις βάσεις δεδομένων για την διαχείριση των δεδομένων.

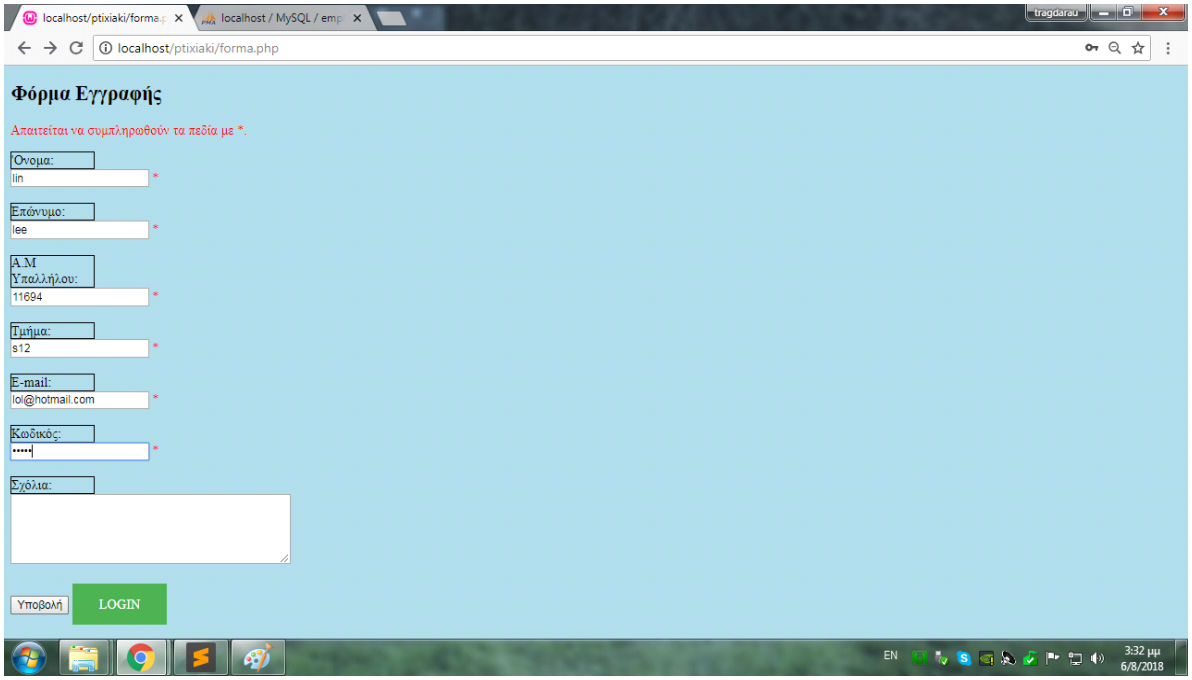
Η εφαρμογή αλληλεπιδρά με μια βάση δεδομένων ώστε να επιτευχθεί η αποθήκευση πληροφοριών αναφοράς.Ουσιαστικά ο χρήστης εισάγει δεδομένα τα οποία επεξεργάζονται από τη βάση δεδομένων. Με αυτό τον τρόπο ένας επιτιθέμενος έχει τη δυνατότητα να εισάγει κακόβουλη πληροφορία και έπειτα να προκαλέσει την εφαρμογή να του επιστρέψει πληροφορίες, (που στην ιδανική λειτουργία δεν πρέπει) είτε λόγο αδυναμίας είτε λόγο ευπάθειας.



Εικόνα 4.23 webapp

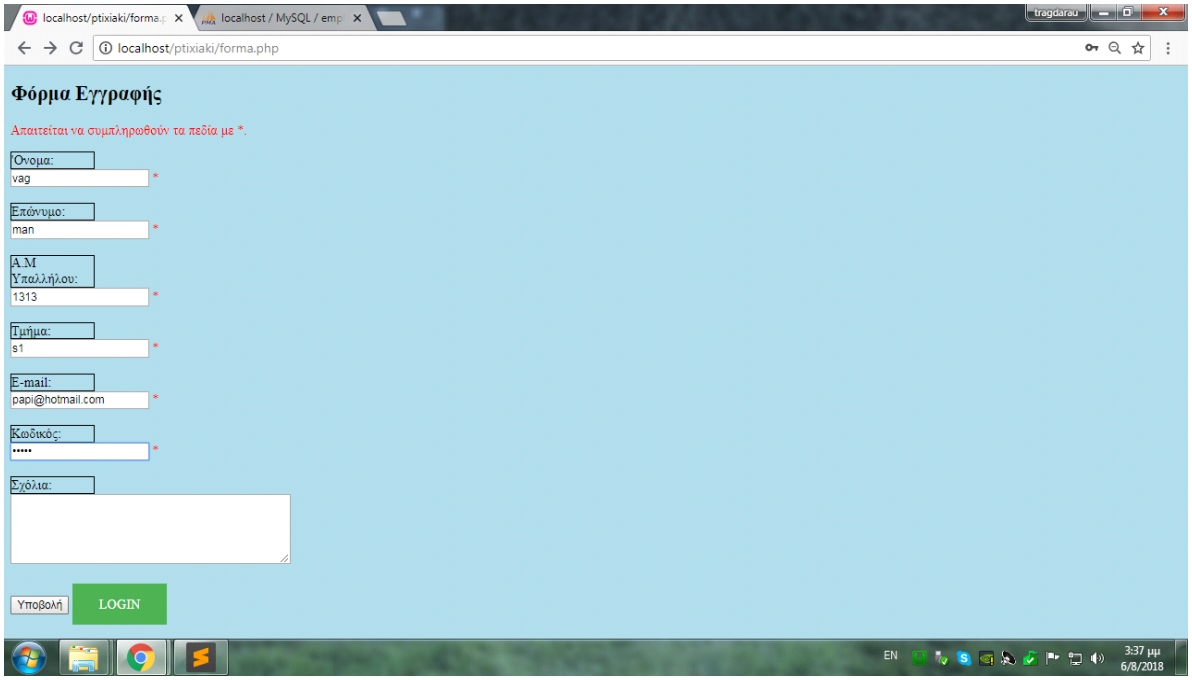
Η εφαρμογή όπως βλέπουμε στην επόμενη εικόνα, εισάγει τα στοιχεία (όνομα, επώνυμο, αριθμό μητρώου κλπ) υπαλλήλων ενός νοσοκομείου και πατώντας "Υποβολή", τα δεδομένα αποθηκεύονται στη βάση δεδομένων. Θα προσθέσουμε για παράδειγμα τον χρήστη με τα εξής στοιχεία:

Όνομα "lin", Επώνυμο "lee", αριθμό μητρώου "11694", ο οποίος εργάζεται στο τμήμα του νοσοκομείου "s12", με e-mail "lol@hotmail.com" και πληκτρολογεί τέλος τον προσωπικό του κωδικό.

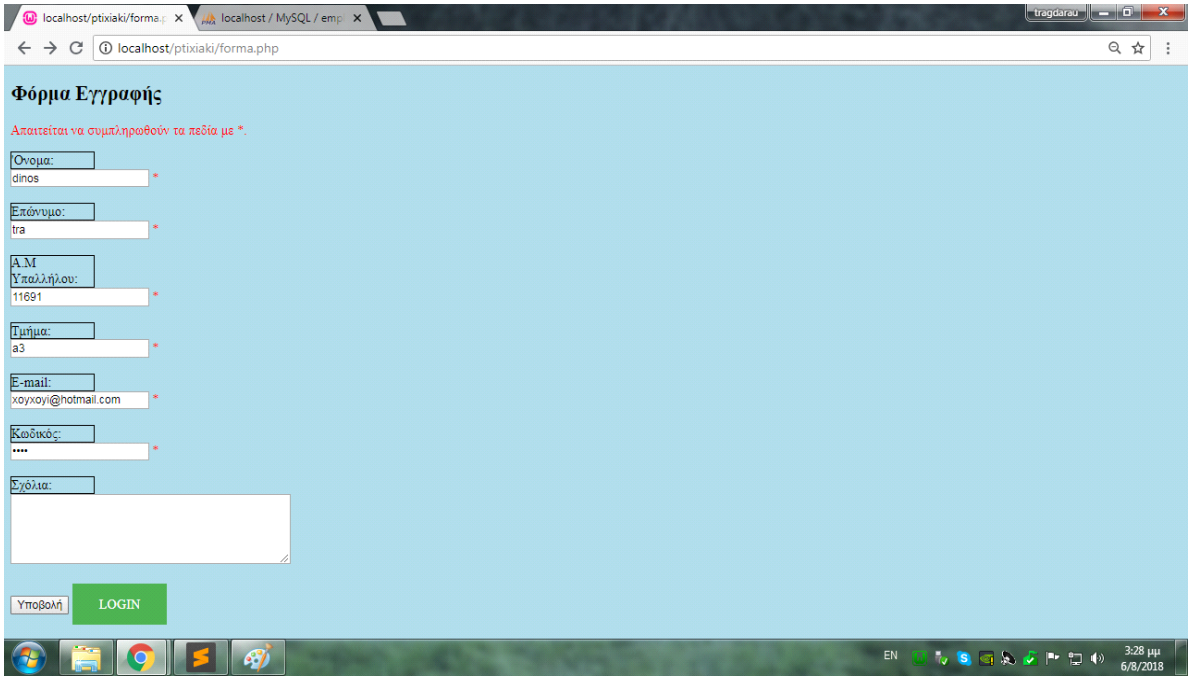


Εικόνα 4.24 login1

Ας καταχωρήσουμε τώρα δύο ακόμα υπαλλήλους με τα στοιχεία τους με παρόμοιο τρόπο.

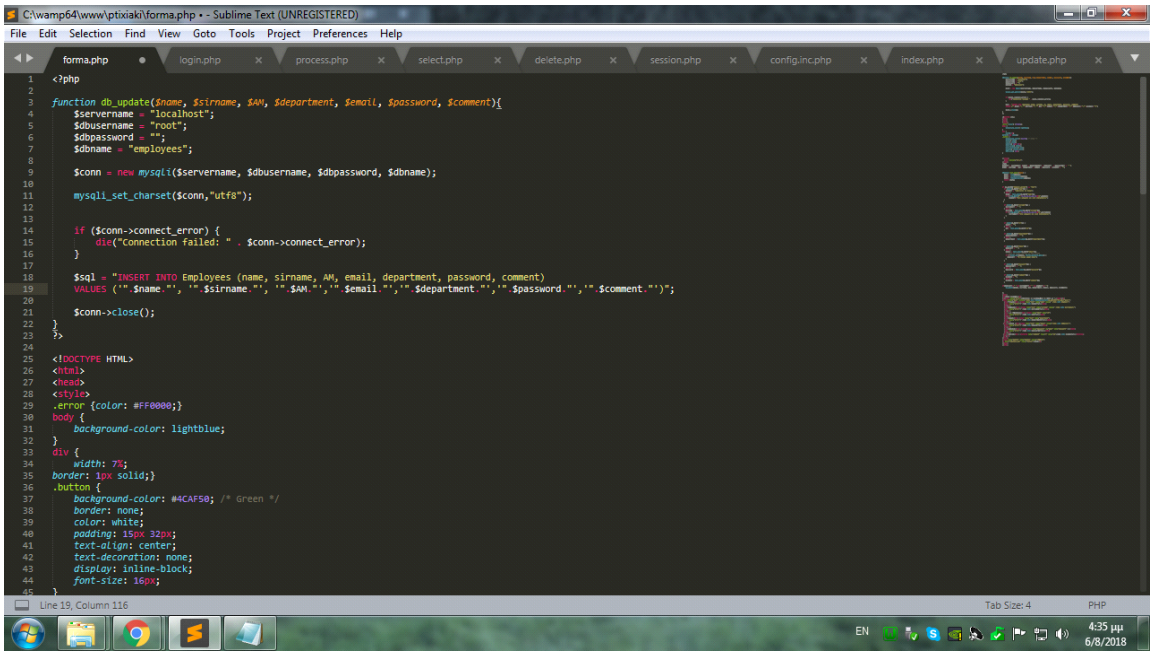


Εικόνα 4.25 login2

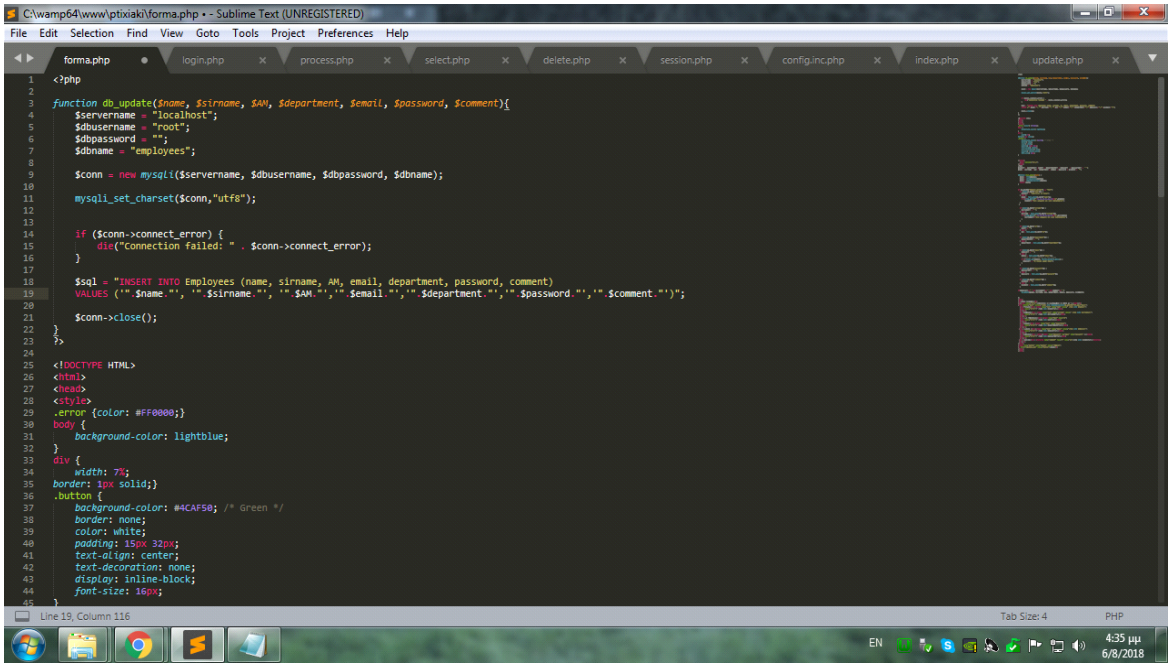


Εικόνα 4.26 login3

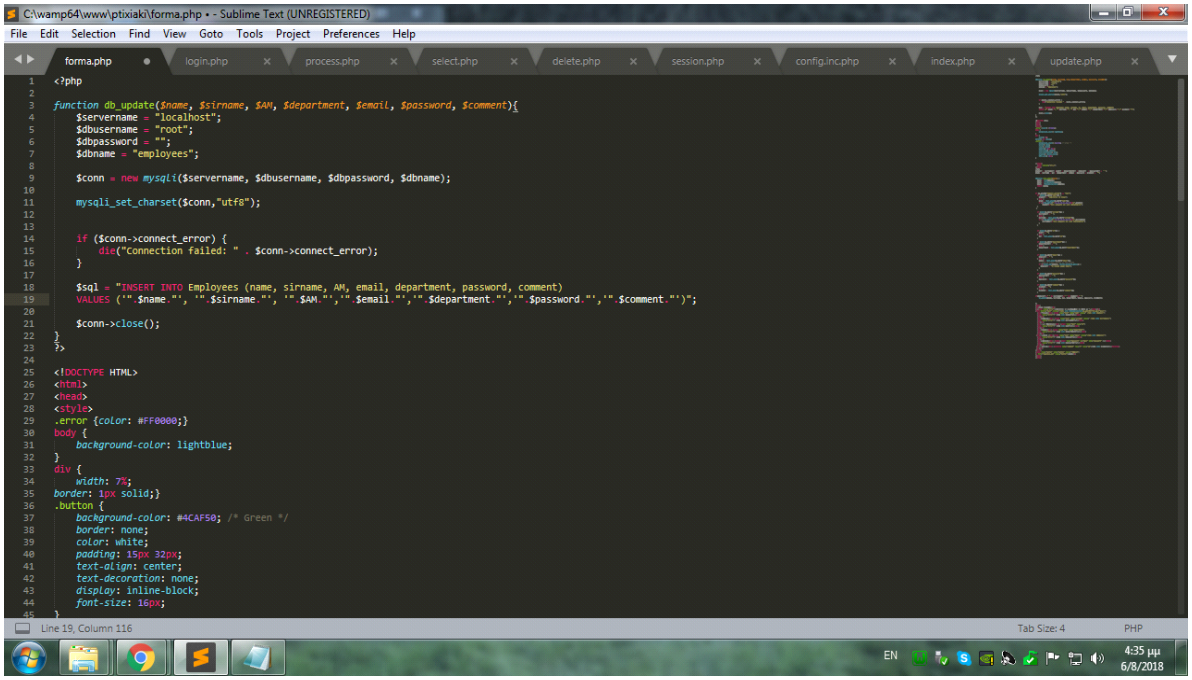
Η υλοποίηση εφαρμογής σελίδας διαδικτύου που εξυπηρετεί μια βάση για την καταχώρηση των υπαλλήλων φαίνεται στις παραπάνω εικόνες.



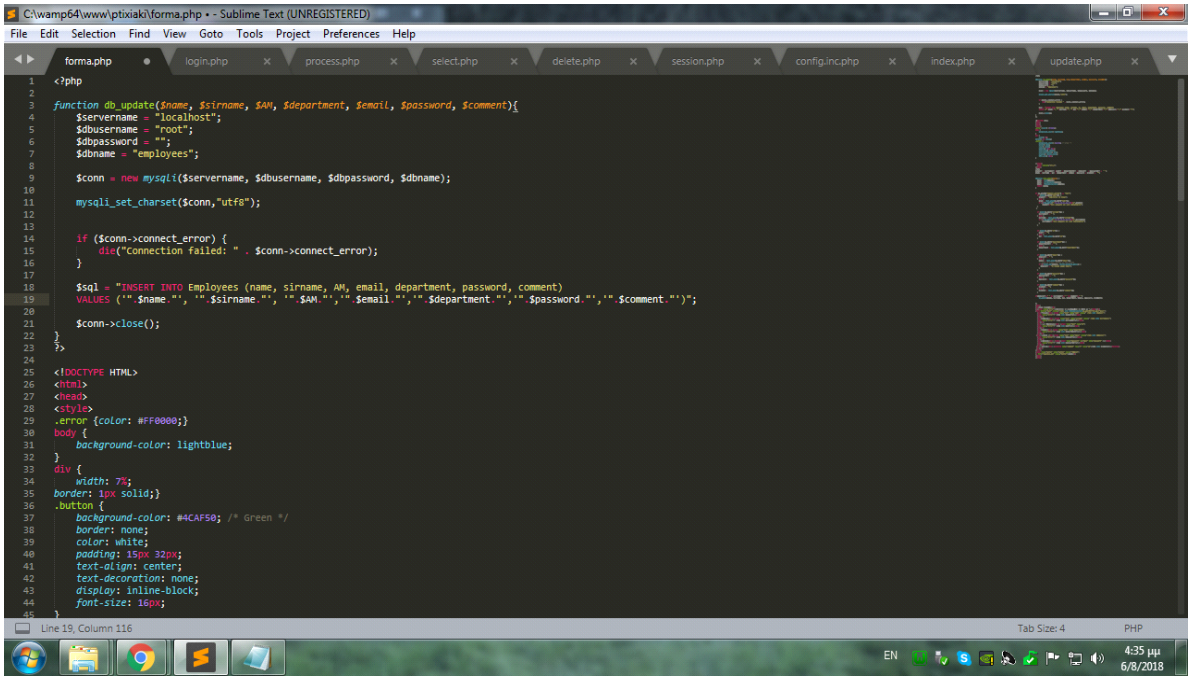
Εικόνα4.27codeform1.



Εικόνα4.28codeform2



Εικόνα4.29codeform3



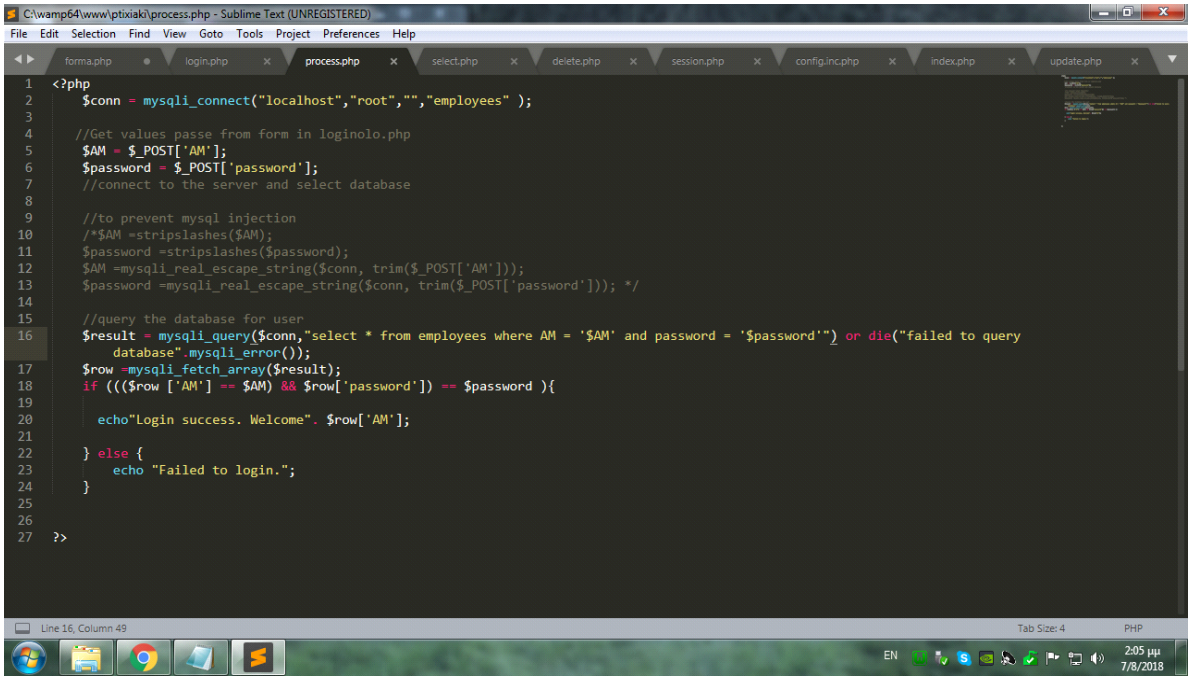
Εικόνα4.30codeform4

Έχουν εγγραφεί 3 υπαλλήλοι με τα στοιχεία τους, τα οποία βλέπουμε στις παραπάνω εικόνες. Χρησιμοποιήθηκε η γλώσσα Php για να γραφεί κώδικας που να υλοποιεί μια σελίδα που να επικοινωνεί με την βάση που υλοποιήθηκε σε sql, πιο συγκεκριμένα στην ειικόνα φαίνεται η διασύνδεση με την βάση.

Για να μεταφερθούν στην κύρια ιστοσελίδα του νοσοκομείου όπου χρειάζεται για να εργαστούν, πρέπει να κάνουν σύνδεση (LOGIN) με δύο προσωπικά τους στοιχεία, τον αριθμό μητρώου (AM) και τον κωδικό (password) τους. Συμπληρώνοντας, λοιπόν, τα δυο τους αυτά στοιχεία-μοναδικό αριθμό μητρώου και απόκρυφο κωδικό πρόσβασης-, και πατώντας το κουμπί "LOGIN" τρέχει ο κώδικας που φαίνεται στην εικόνα 4.19.

Εδώ πραγματοποιούμε μια επίθεση όπου κακόβουλος χρήστης θα προσπαθήσει να έχει πρόσβαση στα προσωπικά δεδομένα χωρίς να γνωρίζει τον μυστικό κωδικό από την βάση δεδομένων.

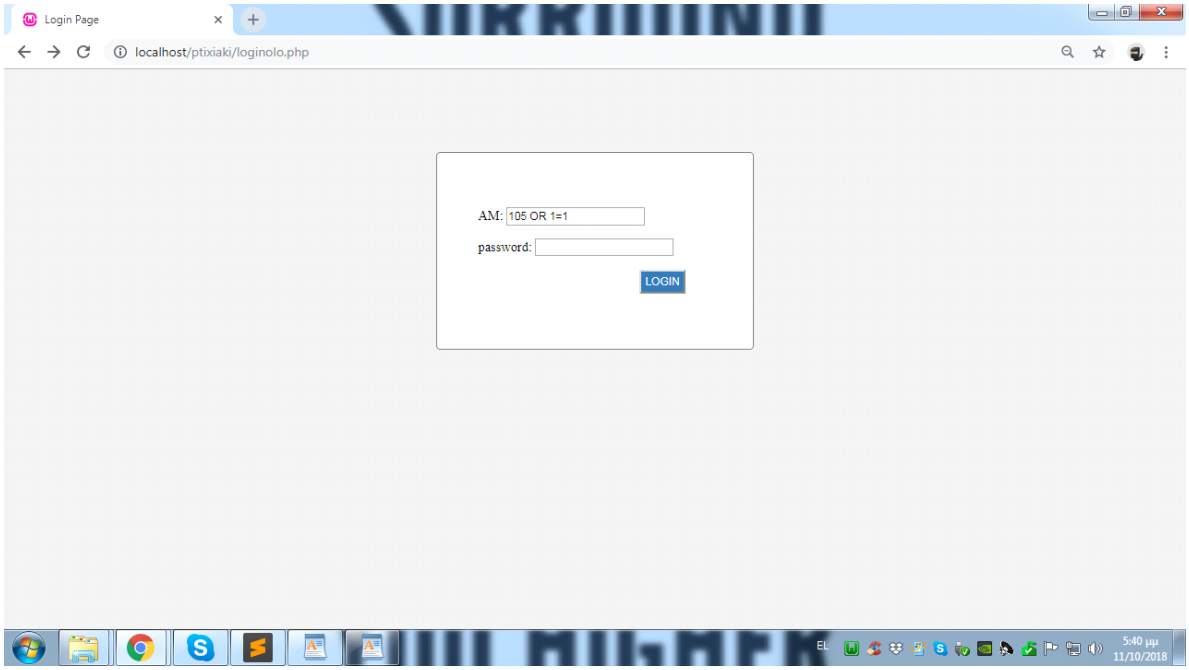
Σε αυτό τον κώδικα ο χρήστης δίνει τον παραπάνω προσωπικό του συνδυασμό. Στην παρακάτω εικόνα



τρέχει ο κώδικας στη βάση δεδομένων που επιστρέφει έναν έγκυρο συνδυασμό και επιτρέπει την πρόσβαση στην κύρια ιστοσελίδα. Η βάση δεδομένων αποθηκεύει τους κωδικούς για κάθε χρήστη και κάνει ανάκτηση με τον ΑΜ. Στην περίπτωση που δωθεί μη έγκυρος συνδυασμός εμφανίζεται μήνυμα απόρριψης.

Στον ίδιο κώδικα έχουμε βάλει σε σχόλια, από γραμμή 9 έως 14, τον κώδικα που αποτρέπει να γίνει mysql injection έτσι ώστε να παρατηρήσουμε το πως ένας κακόβουλος χρήστης χωρίς να γνωρίζει ένα συνδυασμό, έχει πρόσβαση στην ιστοσελίδα σαν πραγματικός εγγεγραμμένος υπάλληλος. Παρακάτω θα δούμε αναλυτικά δύο τρόπους Sql Injection στην εφαρμογή μας.

Ο πιο κοινός τρόπος για να έχει πρόσβαση στην κύρια σελίδα, είναι βάζοντας στο πεδίο του "AM" το εξής: "105 OR 1=1" όπως στην εικόνα

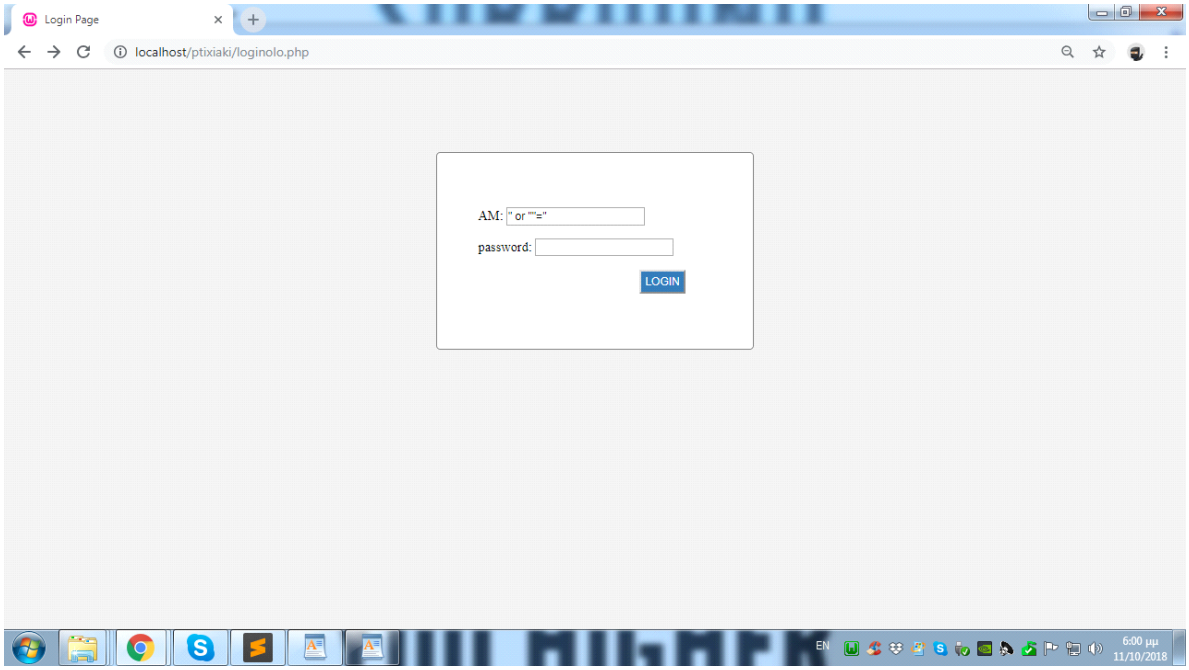


'Υστερα πατώντας το κουμπί "LOGIN" έχουμε πρόσβαση στην ιστοσελίδα χωρίς να γνωρίζουμε στην πραγματικότητα ένα συνδυασμό AM και password που βρίσκονται στη βάση δεδομένων. Αυτό γίνεται ως εξής:

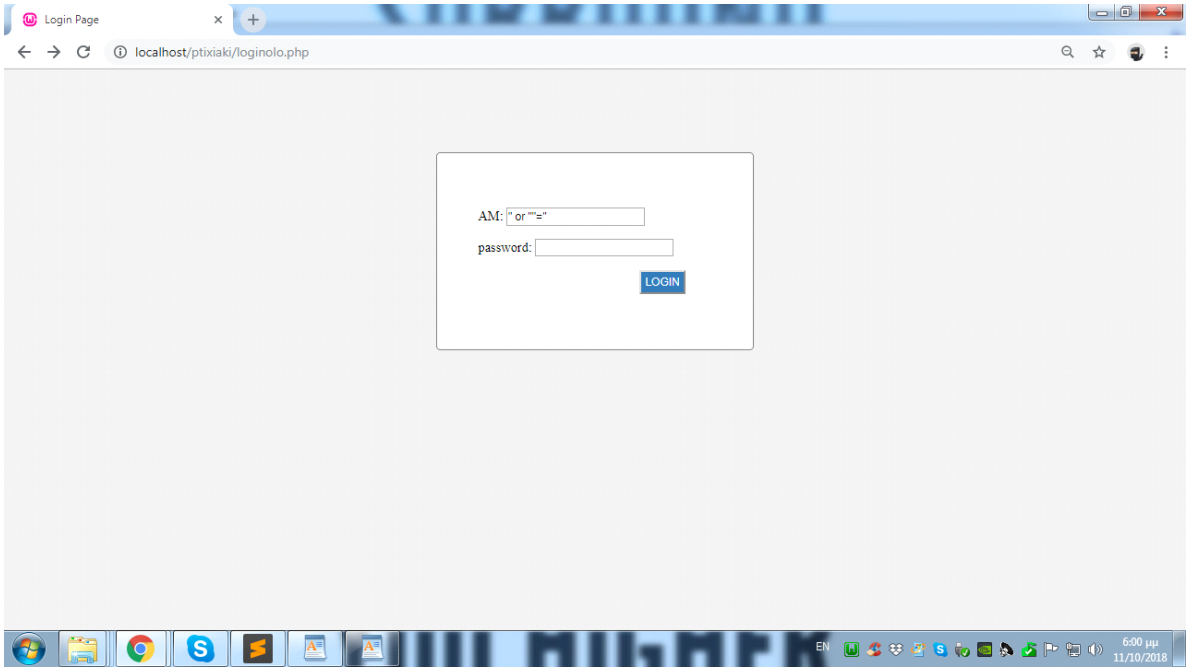
$result = mysqli_query($conn,"select * from employees where AM = '$AM' and password = '$password'") or die("failed to query database".mysqli_error());

Το παραπάνω κομμάτι κώδικα που θα τρέξει θα πάρει για AM μια συνθήκη που είναι πάντα αληθής (105 OR 1=1) αγνοώντας την τιμή του password.

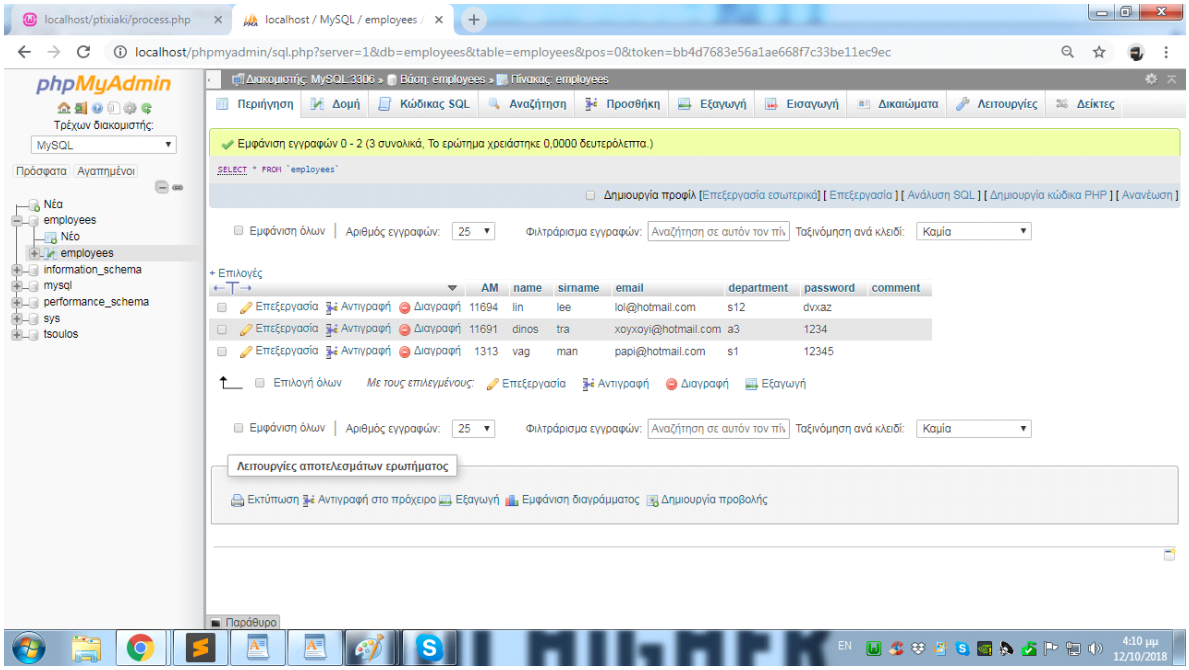
Ένας ακόμη τρόπος είναι βάζοντας στο textbox του AM το εξής input: " or ""=" όπως δείχνει η εικόνα



,έχουμε ξανά πρόσβαση στην σελίδα χωρίς όμως να εμφανίζει μύνημα που καλωσορίζει κάποιον εγγεγραμένο χρήστη με το όνομα του αλλά ανώνυμα όπως θα έπρεπε.

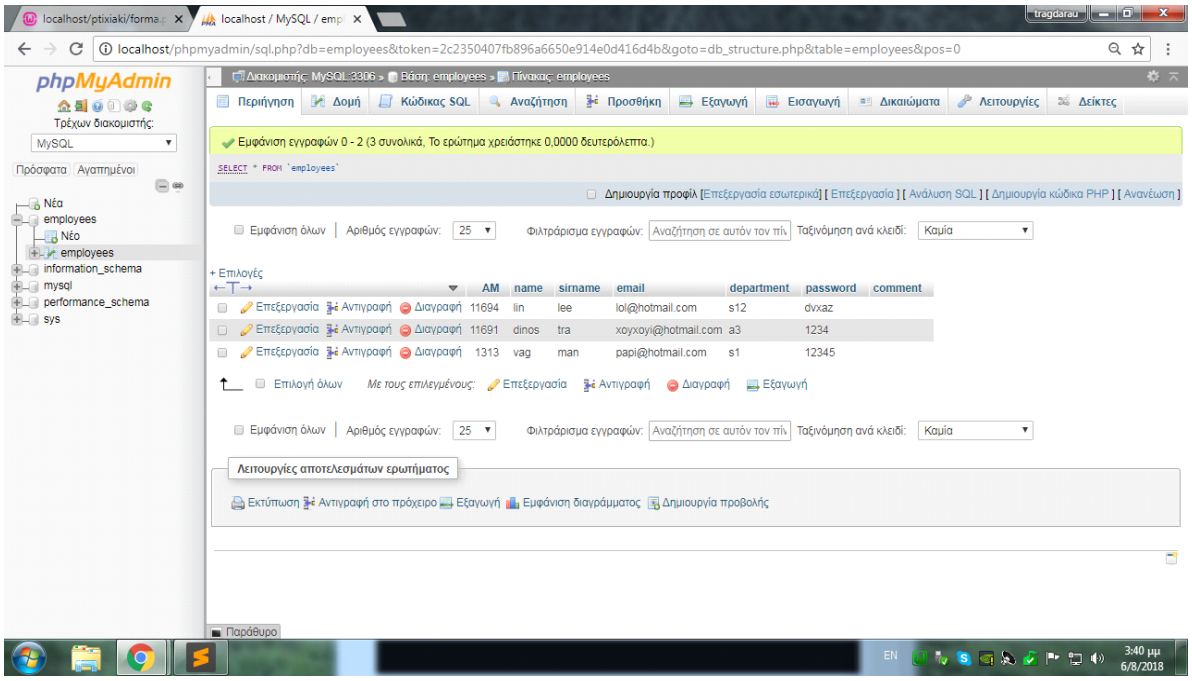


Η παράκαμψη της ταυτότητας δεν είναι η μοναδική κακόβουλη ενέργεια. Ένας χάκερ μπορεί να έχει ως σκοπό την διαγραφή του πίνακα των υπαλλήλων στη βάση δεδομένων όπως σε αυτή εδώ της εφαρμογής μας.

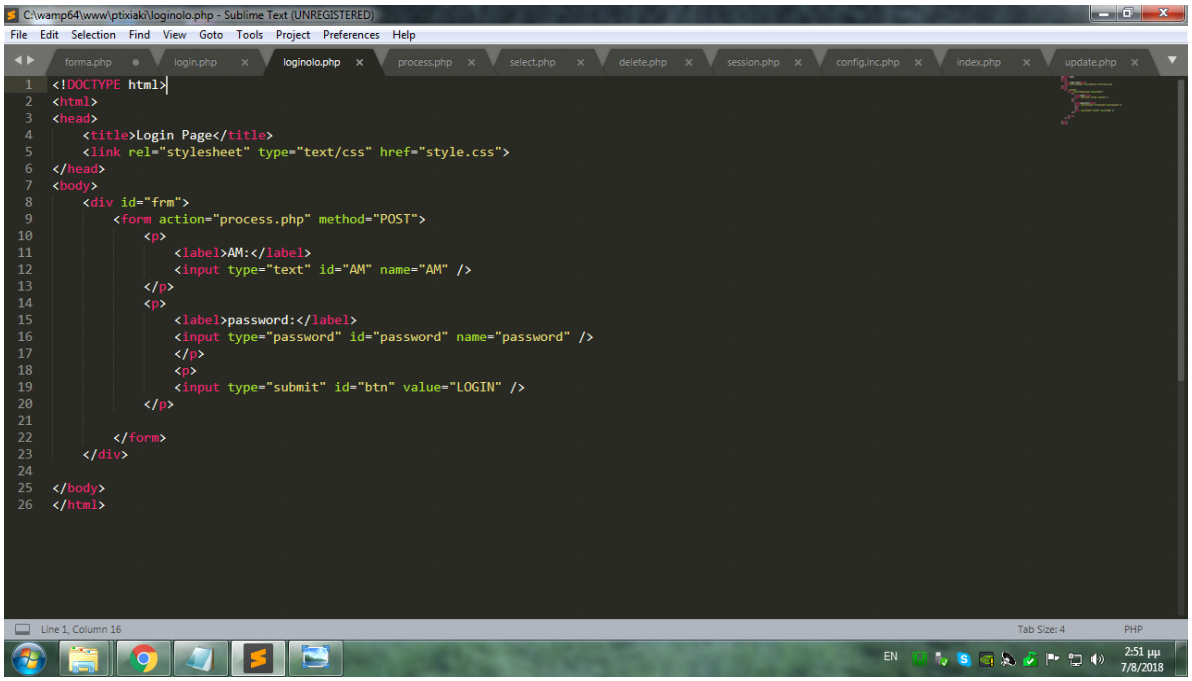


Σε αυτή την περίπτωση γράφει στο input: 105; DROP TABLE employees. Ο χάκερ θα μπορέσει να έχει πρόσβαση στην κύρια σελίδα αλλά ο πίνακας δεν θα διαγραφεί διότι η mysql_query() που τρέχει το πρόγραμμά μας, δεν επιτρέπει πολλαπλά ερωτήματα, όπως το παρακάτω:

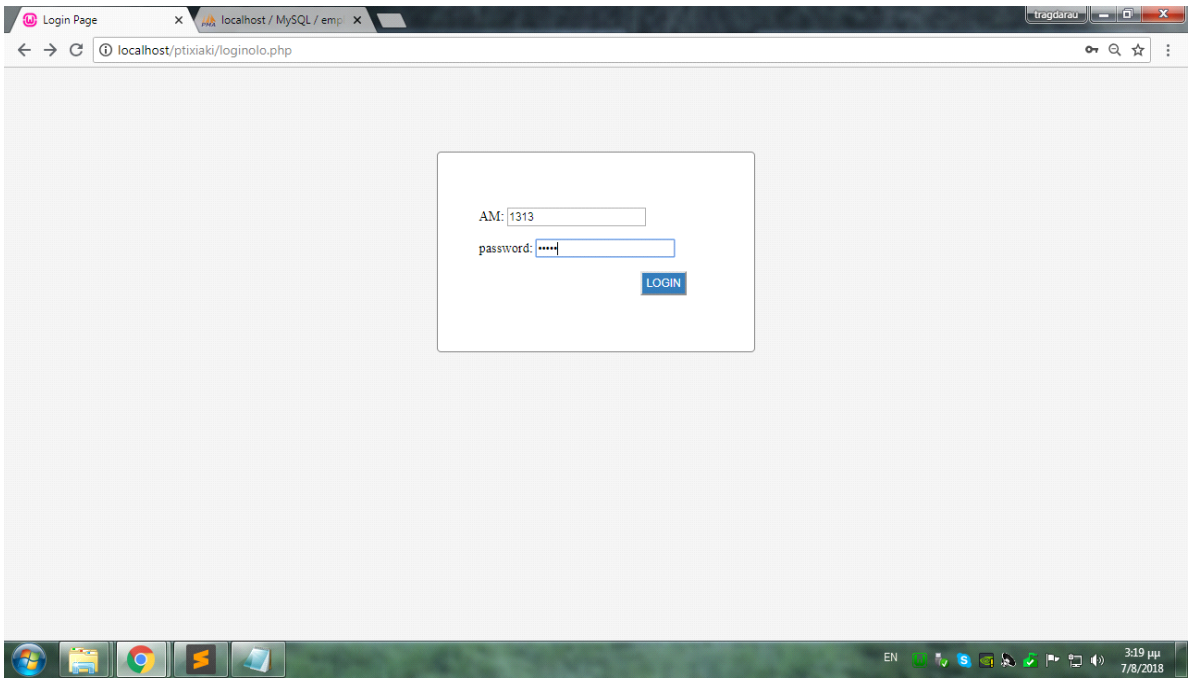
SELECT * FROM employees WHERE AM = 105; DROP TABLE employees;



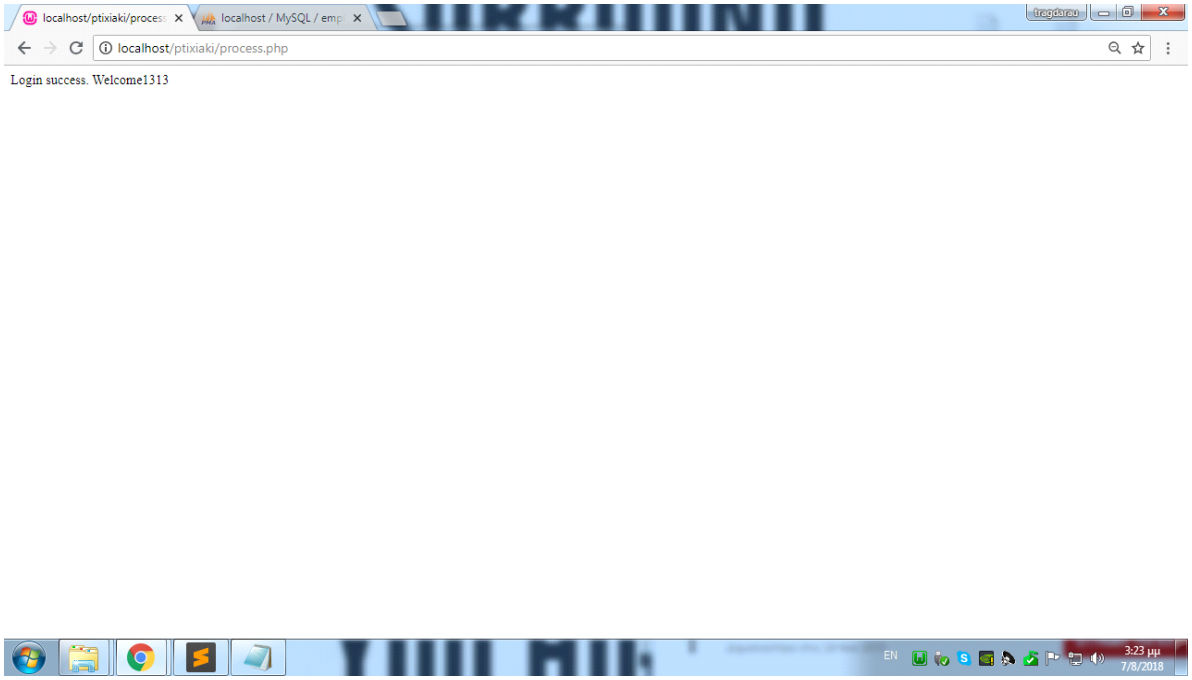
Αυτοί ήταν δύο τρόποι λοιπόν sql manual injection, ένας επιτυχής και ένας ανεπιτυχής στο να διαγράψει τον πίνακα, με πρόσβαση παρόλαυτα στην ιστοσελίδα.



Εικόνα 4.19: login

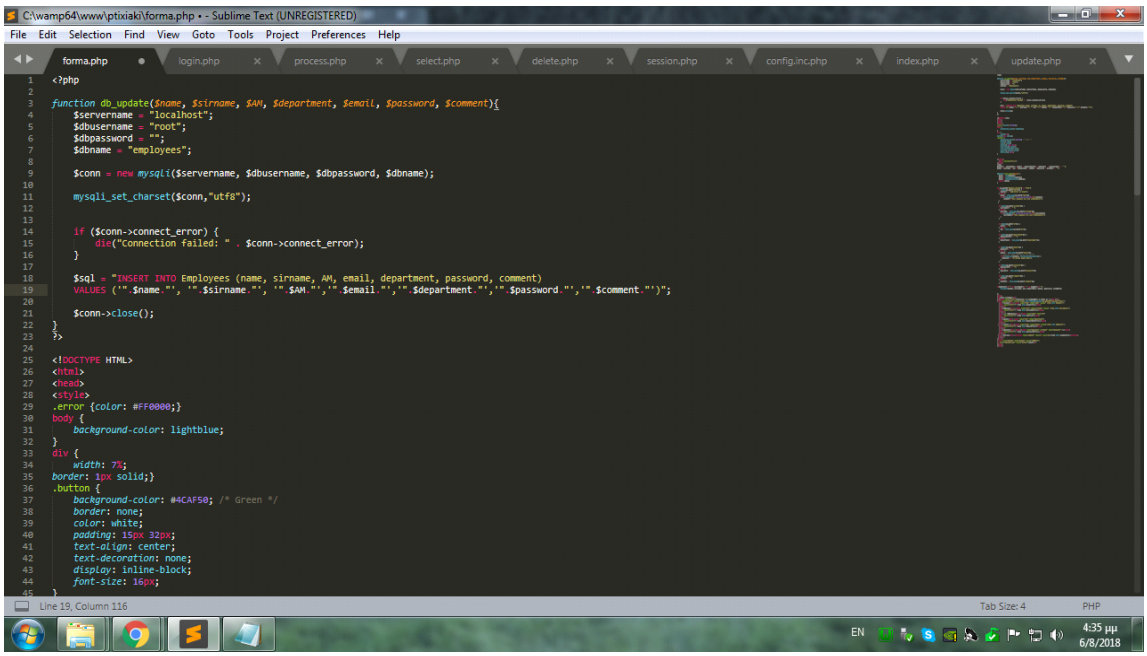


Εικόνα 4.20 : login_page

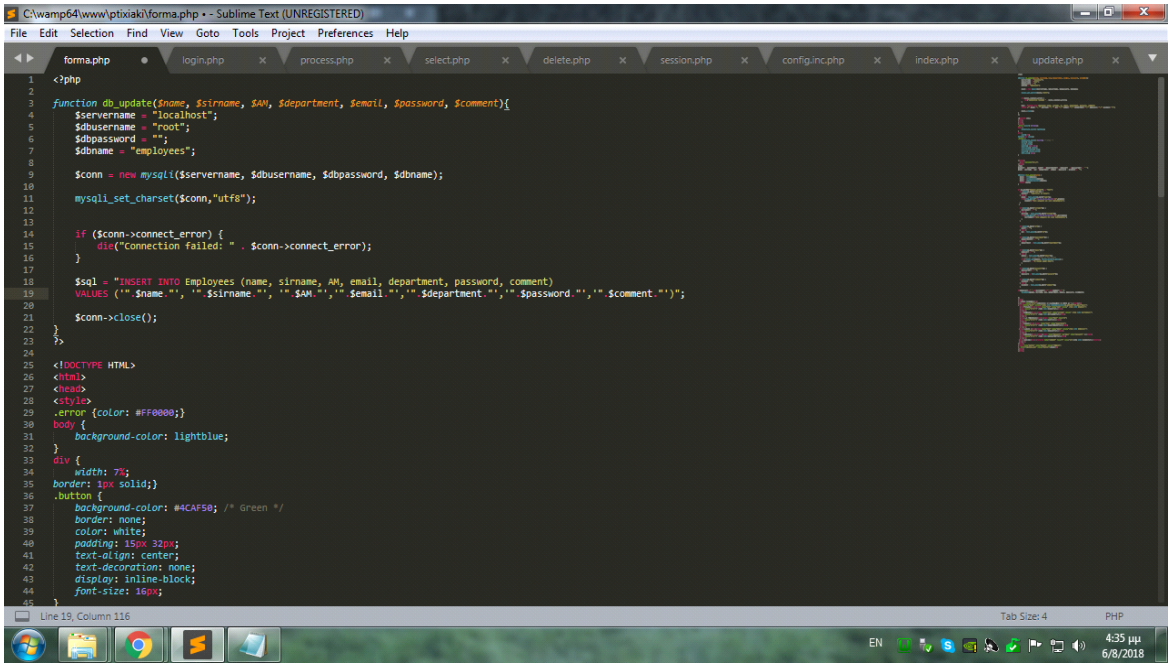


Εικόνα 4.21 : login_success

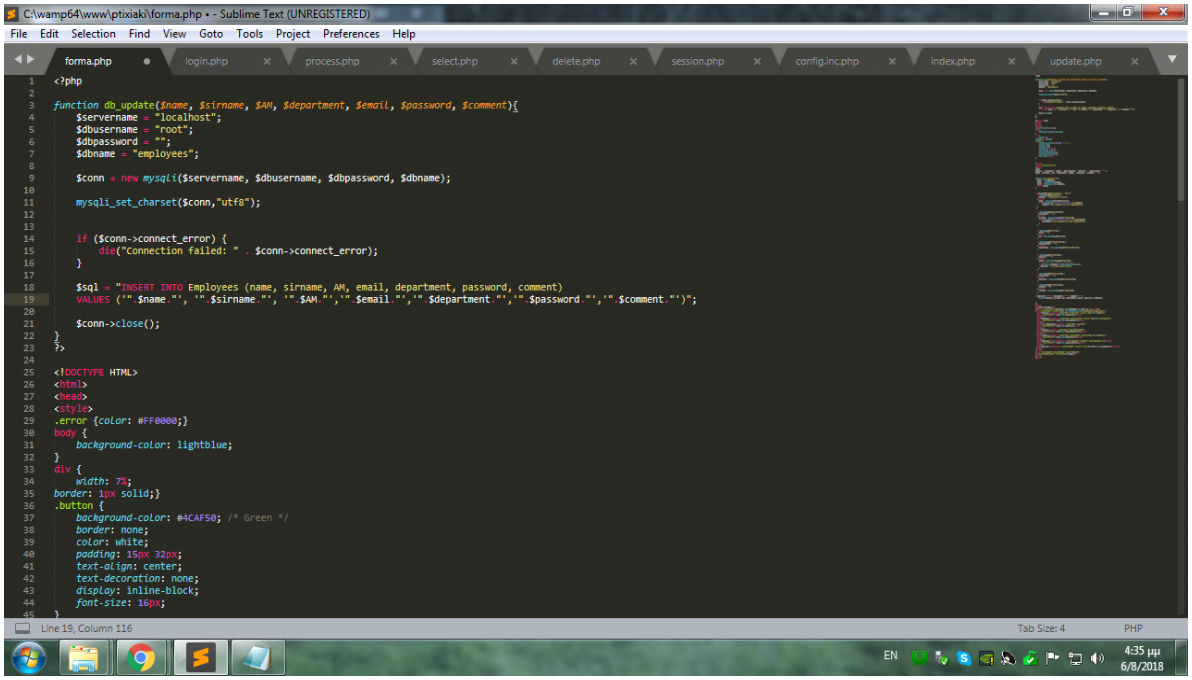
Εικόνα 4.22 : process



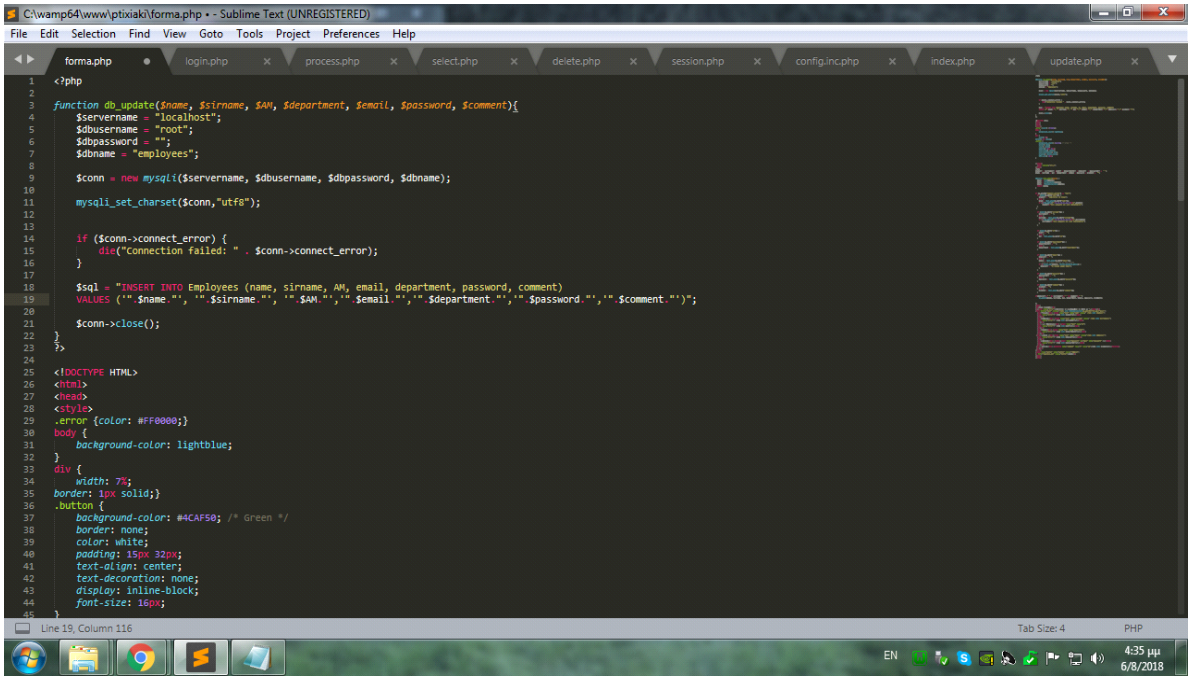
Εικόνα 4.23: codeforma1



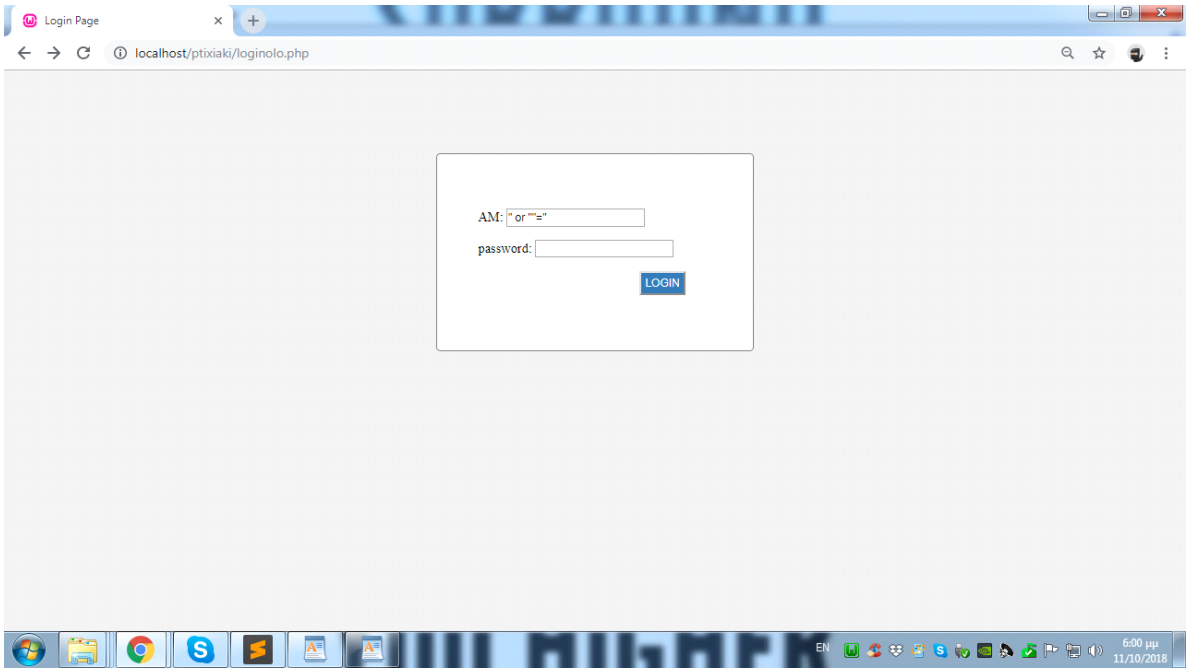
Εικόνα 4.24 :codeforma2



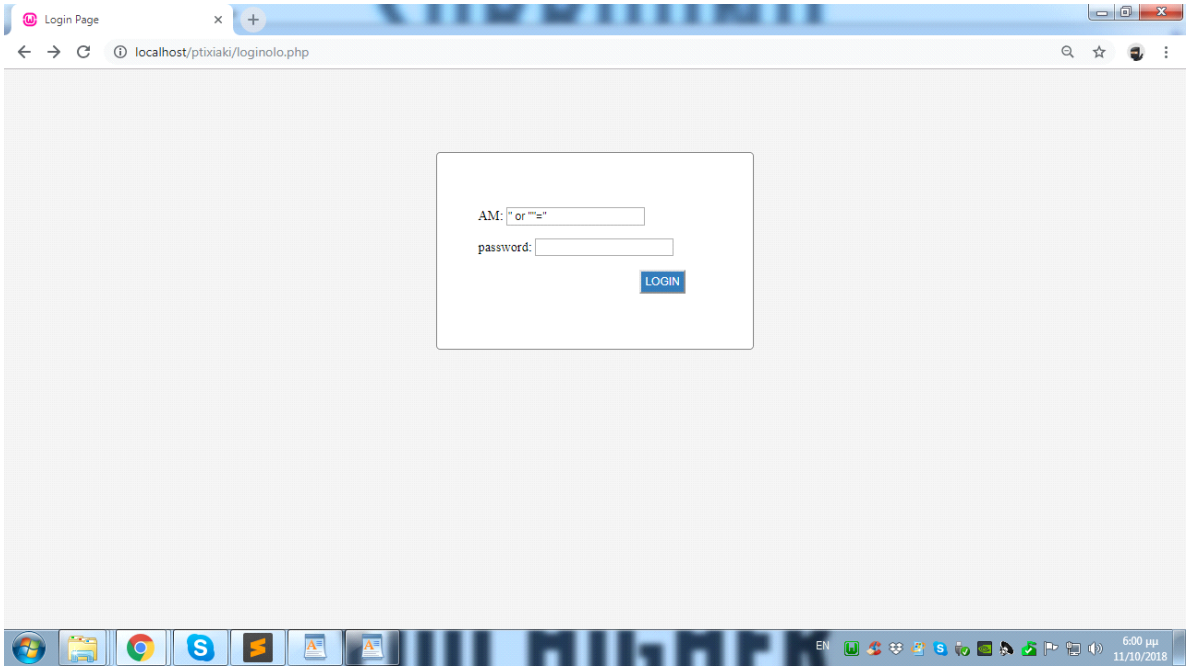
Εικόνα 4.25 :codeforma3



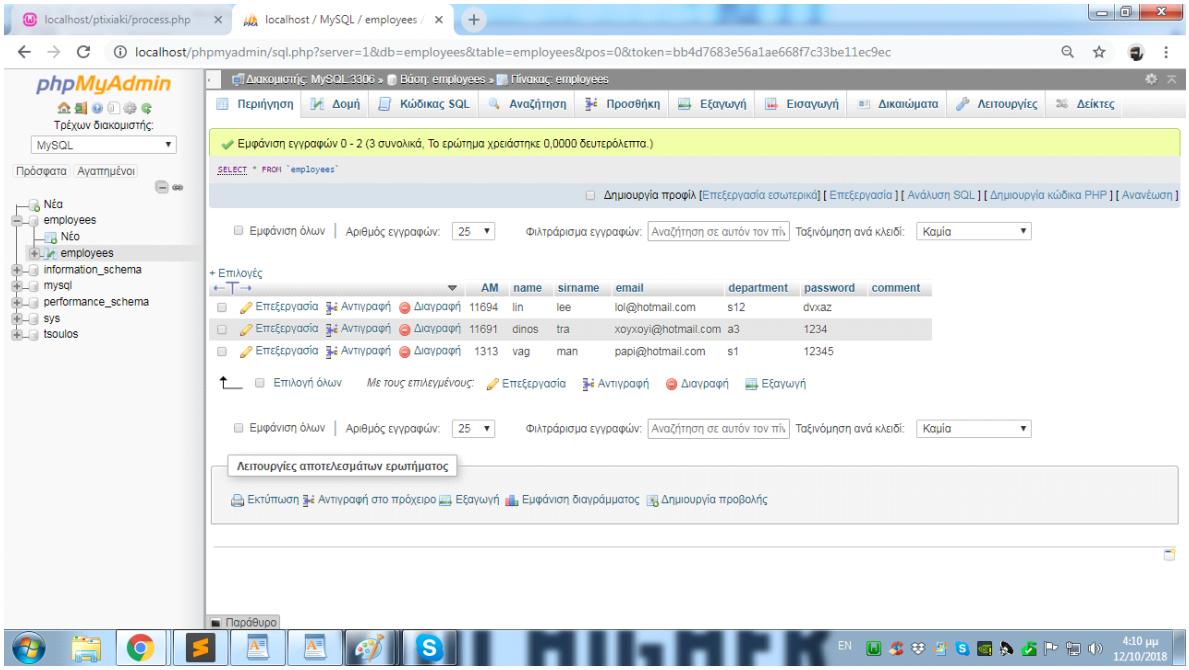
Εικόνα 4.26 :codeforma4



Εικόνα 4.28 :fakelogin1



Εικόνα 4.29 :fakeloginsuccess



Εικόνα 4.30 :droptablebefore

Συμπεράσματα

Κατά την εκπόνηση της παρούσας πτυχιακή εργασία μελετήσαμε την ασφάλεια των Βάσεων Δεδομένων και γνωρίσαμε το SQL Injection. Με τον όρο Το SQL Injection αναφερόμαστε σε μια τεχνική για την κακόβουλη εκμετάλλευση εφαρμογών που χρησιμοποιούν δεδομένα τα οποία με τη σειρά τους παρέχονται από τον πελάτη σε καταστάσεις SQL.

Στο δεύτερο κεφάλαιο γνωρίσαμε τους τύπους του SQL Injection. Τους διακρίναμε με βάση τον στόχο και με βάση το SQL ερώτημα που χρησιμοποιεί. Όσον αφοχά τη δεύτερη κατηγοριοποίηση γνωρίσαμε τους τύπους: Tautologies, Illegal/Logically Incorrect Queries, Union Query, Piggy- back Queries, Alternate Encodings και Inference. Στη συνέχεια αναφέραμε μικρά παραδείγματα SQL Injection με στόχο την καλύτερη κατανόηση τους.

Στο επόμενο κεφάλαιο είδαμε μερικά από τα πιο κύρια παραδείγματα sql injection. Τα πρώτα από αφορούσαν την σύνδεση χρήστη σε ιστότοπο, με χρήση εντολής select αλλά και έγχυσης sql βασισμένη σε καταγεγραμμένες δηλώσεις sql. Επίσης είδαμε παραδείγματα χρήσης των παραμέτρων sql, επίθεσης και άμυνας τα οποία πραγματοποιήθηκαν σε java και τέλος μερικά παραδείγματα αβλαβούς ένεσης και αξιολογήσαμε τα αποτελέσματα που μας δώθηκαν.

Στο τέταρτο κεφάλαιο γνωρίσαμε τις τεχνικές ανίχνευσης επιθέσεων.Η ανίχνευση επιθέσεων SQL Injection μπορεί να καταστεί εφικτή με τη χρήση κατάλληλων τεχνικών μεθόδων που αποτρέπουν την πρόσβαση με την εισαγωγή κατάλληλων λέξεων κλειδιών ή ακόμη και υπογραφών που χαρακτηρίζονται ως κακόβουλες. Έπειτα γνωρίσαμε έξι από τα πιο γνωστά εργαλείο SQLInjection. Πιο συγκεκριμένα τα: BSQL Hacker, SQLmap, SQLninja, Safe3 SQL Injection, SQLSus και Mole.

Τέλος στο πέμπτο και τελευταίο κεφάλαιο μελετήσαμε την γλώσσα php είναι μια γλώσσα η οποία ασχολείται με τη συγγραφή σεναρίων και τη mysql η οποία αποτελεί ένα λογισμικό ανοικτού κώδικα το οποίο έχει να κάνει με τη διαχείριση βάσεων δεδομένων. Έπειτα περιγράψαμε βήμα βήμα την εγκατάσταση του wamp και υλοποιήσαμε μια μικρή εφαρμογή. Η εφαρμογή ζητάει από τον υπάλληλο του

νοσοκομείου να εισάγει τα στοιχεία του:όνομα, επώνυμο, αριθμό μητρώου υπαλλήλου, το τμήμα στο οποίο ανήκει, e-mail, κωδικό πρόσβασης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Atiqur Rahman, Md. Mahbubul Islam. 2015. ***Security Assessment of PHP Web Applications from SQL Injection Attacks.*** Ανακτήθηκε στις 2 Μαίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: https://www.researchgate.net/publication/278242567_Security_Assessment_of_PHP_Web_Applications_from_SQL_Injection_Attacks

[2] Δρ. Μηνάς Δασυγένης. ***Προγραμματισμός Διαδικτύου.*** Πανεπιστημιακές Σημειώσεις. Πανεπιστήμειο Μακεδονίας. Ανακτήθηκε στις 2 Μαίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: http://arch.icte.uowm.gr/courses/webprogramming/oc_SQL_Injection-1.pdf

[3] Yao Chu Zhu. 2016. ***Exploring Defense of SQL Injection Attack in Penetration Testing.*** Ανακτήθηκε στις 2 Μαίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: http://aut.researchgateway.ac.nz/bitstream/handle/10292/10020/ZhuYC.pdf?sequence=3

[4] Fengxiang Lan , Jing Zhang***. Database Security – SQL Injection*** . Ανακτήθηκε στις 2 Μαίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: http://www.cs.rochester.edu/courses/261/fall2017/termpaper/submissions/04/Paper.pdf

[5]W3schools.com. ***SQL Injection*** . Ανακτήθηκε στις 6 Μαίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: https://www.w3schools.com/sql/sql_injection.asp

[6] Lori Mac Vittie. 2015. ***SQL Injection Evasion Detection*** . Ανακτήθηκε στις 6 Μαίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: https://worldtechit.com/wp-content/uploads/2015/07/f5-white-paper-sql-injection-evasion-detection-.pdf

[7] Veracode. ***Sql injection cheat sheet & tutorial: vulnerabilities & how to prevent sql injection attacks.***2017. Ανακτήθηκε στις 12 Μαίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: https://www.veracode.com/security/sql-injection

[8] Ismail Orhan. 2017 .***Best Free and Open Source SQL Injection Tools*** . Ανακτήθηκε στις 22 Μαίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: https://www.linkedin.com/pulse/best-free-open-source-sql-injection-tools-ismail-orhan-bsceng-ceh-1

[9] Κεντρο Πλη.νε.τ. ν. Φλωρινας. ***Η Γλώσσα Προγραμματισμού PHP.*** Ανακτήθηκε στις 22 Μαίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-Php-Analytical.html

[10] ***Ενα εισαγωγικό tutorial στην MySQL.*** 2017. Ανακτήθηκε στις 22 Μαίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: https://c2.gr/blog/mysql/

[11] Jeff Orloff. ***Mysql. Installing and configuring a wamp server on your computer.*** Ανακτήθηκε στις 22 Μαίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: http://www.developerdrive.com/2011/08/installing-and-configuring-a-wamp-server-on-your-computer/

[12] ***AMNESIA: Analysis and Monitoring for Neutralizing SQL-Injection Attacks.*** Ανακτήθηκε στις 2 Ιουνίου 2018. Διαθέσιμο στον δικτυακό ιστότοπο: http://slideplayer.com/slide/7087349/

[13] ***JDBC checker: a static analysis tool for SQL/JDBC applications***.Ανακτήθηκε στις 2 Ιουνίου 2018.Διαθέσιμο στον δικτυακό ιστότοπο: https://www.semanticscholar.org/paper/JDBC-checker%3A-a-static-analysis-tool-for-SQL%2FJDBC-Gould-Su/0743d913921a44750000889584a03b543e38825d

[14] ***SQLrand: Preventing SQL Injection Attacks*** .Ανακτήθηκε στις 2 Ιουνίου 2018.Διαθέσιμο στον δικτυακό ιστότοπο: http://web1.cs.columbia.edu/~angelos/Papers/sqlrand.pdf

ΠΑΡΑΡΤΗΜΑ

Εδώ θα δούμε τους κώδικες σε διάφορα παραρτήματα με βάση την λειτουργία τους.

1. Ο κώδικας αυτός αναπτύχθηκε για την είσοδο των στοιχείων των υπαλλήλων

```
<?php

function db_update($name, $sirname, $AM, $department, $email, $password, $comment){
	$servername = "localhost";
	$dbusername = "root";
	$dbpassword = "";
	$dbname = "employees";

	$conn = new mysqli($servername, $dbusername, $dbpassword, $dbname);

	mysqli_set_charset($conn,"utf8");


	if ($conn->connect_error) {
		die("Connection failed: " . $conn->connect_error);
	}

	$sql = "INSERT INTO Employees (name, sirname, AM, email, department, password, comment)
	VALUES ('".$name."', '".$sirname."', '".$AM."','".$email."','".$department."','".$password."','".$comment."')";

	$conn->close();
```

```
}
?>

<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
body {
    background-color: lightblue;
}
div {
    width: 7%;
border: 1px solid;}
.button {
    background-color: #4CAF50; /* Green */
    border: none;
    color: white;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
```

```
}



</style>
  <meta charset="UTF-8">
</head>
<?php
$nameErr = $sirnameErr =$AMErr = $departmentErr = $emailErr  = $passwordErr =
" ";
$name = $sirname = $AM = $department = $email = $password = $comment =  "";



function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}



if ($_SERVER["REQUEST_METHOD"] == "POST")
  if (empty($_POST["name"])) {
    $nameErr = "Απαιτείται το όνομα";
```

```
} else {
  $name = test_input($_POST["name"]);
        if (!preg_match("/^[A-Za-z0-9α-ωA-Ω ]+$/u",$name)){
                $nameErr="Μόνο γράμματα και κενά επιτρέπονται";
 }
}

if (empty($_POST["sirname"])) {
  $sirnameErr = " ";
 } else {
  $sirname = test_input($_POST["sirname"]);
        if (!preg_match("/^[A-Za-z0-9α-ωA-Ω ]+$/u",$sirname)){
                $sirnameErr="Μόνο γράμματα και κενά επιτρέπονται";
        }
 }


if (empty($_POST["AM"])) {
  $AMErr = "";
 } else {
  $AM = test_input($_POST["AM"]);
 }
```

```
if (empty($_POST["department"])) {
  $departmentErr = " ";
} else {
  $department = test_input($_POST["department"]);
}



if (empty($_POST["email"])) {
  $emailErr = " ";
} else {
  $email = test_input($_POST["email"]);
  // check if e-mail address is well-formed
  if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $emailErr = "Μη έγκυρη μορφή email";
  }
}


if (empty($_POST["password"])) {
  $passwordErr = " ";
} else {
  $password = test_input($_POST["password"]);
}
```

```
 if (empty($_POST["comment"])) {
  $comment = "";
 } else {
  $comment = test_input($_POST["comment"]);
 }

if($emailErr==" " && $sirnameErr==" " && $nameErr==" ")
        db_update($name, $sirname, $AM, $department, $email, $password, $comment);

?>
<body>
<h2>Φόρμα Εγγραφής</h2>
<p><span class="error">Απαιτείται να συμπληρωθούν τα πεδία με *.</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
 <div>'Ονομα:</div> <input type="text" name="name" value=" <?php echo $name;?>">
 <span class="error">* <?php echo $nameErr;?></span>
 <br><br>
 <div>Επώνυμο:</div> <input type="text" name="sirname" value=" <?php echo $sirname;?>">
 <span class="error">* <?php echo $sirnameErr;?></span>
```

```
<br><br>

<div> A.M Υπαλλήλου:</div> <input type="text" name="AM">

<span class="error">* <?php echo $AMErr;?></span>

<br><br>

<div>Τμήμα:</div> <input type="text" name="department">

<span class="error">* <?php echo $departmentErr;?></span>

<br><br>

<div> E-mail: </div><input type="text" name="email" value="<?php echo $email;?>">

<span class="error">* <?php echo $emailErr;?></span>

<br><br>

<div>Κωδικός:</div> <label><input type="password" id="pass" name="password" /></label>

<span class="error">* <?php echo $passwordErr;?></span>

<br><br>

<div> Σχόλια:</div> <textarea name="comment" rows="5" cols="40"><?php echo $comment;?></textarea>

<br><br>

</form>

<input type="button" name="button" value="Υποβολή">

<a href="loginolo.php" class="button">LOGIN</a>

</body>

</html>
```

2. Όταν πατήσει ο χρήστης το LOGIN, τότε θα μεταφερθεί σε σελίδα που τρέχει τον εξής κώδικα.

2.1. Το κομμάτι κώδικα σε html.

```
<!DOCTYPE html>
<html>
<head>
	<title>Login Page</title>
	<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
	<div id="frm">
		<form action="process.php" method="POST">
			<p>
				<label>AM:</label>
				<input type="text" id="AM" name="AM" />
			</p>
			<p>
				<label>password:</label>
				<input type="password" id="password" name="password" />
				</p>
				<p>
				<input type="submit" id="btn" value="LOGIN" />
```

```
                    </p>


                </form>
        </div>


</body>
</html>
```

2.2 Το αντίστοιχο κομμάτι κώδικα σε php για την ίδια σελίδα. Στον κώδικα αυτόν θα δούμε και σχόλια για την ενεργοποίση/απενεργοποίηση του sql manual injection.

```
<?php
        $conn = mysqli_connect("localhost","root","","employees" );


  //Get values passe from form in loginolo.php
        $AM = $_POST['AM'];
        $password = $_POST['password'];
        //connect to the server and select database


        //to prevent mysql injection
        /*$AM =stripslashes($AM);
        $password =stripslashes($password);
        $AM =mysqli_real_escape_string($conn, trim($_POST['AM']));
        $password =mysqli_real_escape_string($conn, trim($_POST['password'])); */
```

```
        //query the database for user
        $result = mysqli_query($conn,"select * from employees where AM = '$AM' and password = '$password'") or die("failed to query database".mysqli_error());
        $row =mysqli_fetch_array($result);
   if ((($row ['AM'] == $AM) && $row['password']) == $password ){


     echo"Login success. Welcome". $row['AM'];


   } else {
        echo "Failed to login.";
   }


?>
```

3. Όταν δώσει ο χρήστης ΑΜ και password πρέπει να γίωει έλεγχος για το αν ο συνδυασμός αυτός είναι έγκυρος ώστε να πραγματοποιηθεί η είσοδος. Αναπτύχθηκε με τον εξής κώδικα.

3.1
```
<?php
  include('config.php');
  session_start();


  $user_check = $_SESSION['login_user'];


  $ses_sql = mysqli_query($db,"select AM from admin where AM = '$user_check' ");
```

```
  $row = mysqli_fetch_array($ses_sql,MYSQLI_ASSOC);

  $login_session = $row['AM'];

  if(!isset($_SESSION['login_user'])){
    header("location:login.php");
  }
?>
```

,όπου το config.php είναι:

3.2

```
<?php
  define('DB_SERVER', 'localhost');
  define('DB_USERNAME', 'root');
  define('DB_PASSWORD', '1234');
  define('DB_DATABASE', 'employees');

$db=mysqli_connect(DB_SERVER,DB_USERNAME,DB_PASSWORD,DB_DATA
BASE);
?>
```