Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ασφάλεια Βάσης Δεδομένων σε Διαδικτυακές Εφαρμογές** **Database Security to Web Applications**
Ονοματεπώνυμο Φοιτητή	**Ελένη Κιουλαφή**
Πατρώνυμο	**Σπυρίδων**
Αριθμός Μητρώου	**ΜΠΠΛ/ 08040**
Επιβλέπων	**Παναγιώτης Κοτζανικολάου, Λέκτορας Τμήματος Πληροφορικής**

Ημερομηνία Παράδοσης **Μάρτιος 2014**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Παναγιώτης Κοτζανικολάου Λέκτορας	Δουληγέρης Χρήστος Καθηγητής	Δημήτριος Βέργαδος Επίκουρος Καθηγητής

Περίληψη

Το θέμα της παρούσας εργασίας είναι η Ασφάλεια Βάσης Δεδομένων σε Διαδικτυακές Εφαρμογές. Στην εργασία γίνεται αναφορά στις Απαιτήσεις Ασφάλειας των βάσεων δεδομένων και περαιτέρω ανάλυση των κυριότερων από αυτές. Επίσης περιγράφονται κάποιες από τις πιο γνωστές Απειλές Ασφάλειας για ένα σύστημα βάσεων δεδομένων καθώς και ορισμένα μέτρα που μπορούν να εφορμοστούν ώστε να επιτυχγάνεται όσο το δυνατόν καλύτερη ασφάλεια. Τέλος παρουσιάζεται η υλοποίηση μιας βάσης δεδομένων τα στοιχεία της οποίας διαχειρίζονται μέσω μιας Web εφαρμογής.

Abstract

The object of this paper is the Database Security to Web Applications. In this paper there is a reference to Security Requirments of databases and futher analysis of the main ones. The second part of the paper describes the most known security threats to a database system and a number of measures that can be implemented to achieve the best possible security. Finally there is a presentation of the implementation of a database whose information managed through a web application.

Περιεχόμενα

Κεφάλαιο 1°

1 Εισαγωγή

Η ευρύτατη χρήση της τεχνολογίας Συστημάτων Βάσεων Δεδομένων σε όλο και περισσότερα συστήματα υπολογιστών, σε διάφορες εφαρμογές, καθιστά απαραίτητη την ανάπτυξη μεθόδων για την ασφάλεια των δεδομένων μιας βάσης. Ιδιαίτερα σε πληροφοριακά συστήματα που χρησιμοποιούν εφαρμογές διαδικτύου για τη διαχείριση των πληροφοριών, για να εξασφαλιστεί η ασφάλεια ολόκληρου του πληροφοριακού συστήματος θα πρέπει τα δεδομένα που υπάρχουν στα συστήματα αυτά να προφυλάσσονται από κινδύνους όπως η διαρροή πληροφοριών σε μη εξουσιοδοτημένους χρήστες ή η διαγραφή σημαντικών πληροφοριών από τη βάση.

Με τον όρο ασφάλεια των βάσεων δεδομένων καθορίζουμε τους μηχανισμούς αποφυγής της μη επιθυμητής διαρροής (disclosure), μεταβολής (modification) και καταστροφής (destruction) των αποθηκευμένων δεδομένων.
Η ανάγκη μηχανισμών ασφάλειας αναζητείται όλο και περισσότερο από επιχειρήσεις και οργανισμούς. Αυτό οφείλεται στο γεγονός ότι στην πλειοψηφία τους οι επιχειρήσεις/ οργανισμοί χρησιμοποιούν συστήματα Βάσεων Δεδομένων σε συνδυασμό με Διαδικτυακές Εφαρμογές για την αποθήκευση και διαχείριση κρίσιμων εταιρικών δεδομένων, η απώλεια ή μη διαθεσιμότητα των οποίων μπορεί να οδηγήσει σε ολέθρια αποτελέσματα. Κατά συνέπεια τα συστήματα Βάσεων Δεδομένων πρέπει να διασφαλισθούν κατάλληλα ώστε να μπορούν να αντιμετωπίζουν ενδεχόμενες πράξεις νοθείας ή κλοπής, οι οποίες μπορεί να οδηγήσουν σε απώλεια της εμπιστευτικότητας (confidentiality), της ακεραιότητας (integrity) ή της διαθεσιμότητας (availability) των δεδομένων. Η νοθεία και η κλοπή επηρεάζουν αρνητικά όχι μόνο το περιβάλλον του συστήματος Βάσεων Δεδομένων αλλά και ολόκληρη την επιχείρηση. Η απώλεια της εμπιστευτικότητας οδηγεί σε αποκάλυψη ευαίσθητων προσωπικών δεδομένων ή δεδομένων κρίσιμης σημασίας για την επιχείρηση. Η απώλεια της ακεραιότητας έχει ως αποτέλεσμα την εμφάνιση άκυρων ή διαβρωμένων δεδομένων που ίσως επηρεάσουν σοβαρά τη μελλοντική λειτουργία της επιχείρησης. Απώλεια της διαθεσιμότητας υφίσταται όταν δεν είναι δυνατή η προσπέλαση των δεδομένων ή του συστήματος ή και των δύο, γεγονός που μπορεί να επηρεάσει σοβαρά για παράδειγμα την οικονομική απόδοση ή την αξιοπιστία μιας επιχείρησης ή ενός οργανισμού ιδιαίτερα μάλιστα αν αυτοί παρέχουν ηλεκτρονικές υπηρεσίες εικοσιτετράωρης διαθεσιμότητας. Σε κάποιες περιπτώσεις, οι ιδιότητες της εμπιστευτικότητας, της ακεραιότητας και της διαθεσιμότητας είναι τόσο στενά συνδεδεμένες, ώστε μία πράξη που οδηγεί σε απώλεια μίας εξ αυτών των ιδιοτήτων θα οδηγήσει άμεσα και σε απώλεια μιας άλλης.

Η ασφάλεια ενός Συστήματος Βάσεων Δεδομένων αποτελεί και το θέμα της παρούσας εργασίας. Πιο συγκεκριμένα θα γίνει Ανάλυση των Απαιτήσεων Ασφάλειας ενός ΣΔΒΔ, θα παρουσιαστούν κάποιες από τις κύριες Απειλές Ασφάλειας (π.χ. SQL Injections) και θα γίνει αναφορά σε εργαλεία και μηχανισμούς που μπορούν να χρησιμοποιηθούν για την προφύλαξη των δεδομένων ενός τέτοιου συστήματος.

Με σκοπό μια πιο ολοκληρωμένη μελέτη του θέματος στο τέλος της εργασίας θα παρουσιαστεί μια βάση δεδομένων για τη διαχείριση των πελατών μιας εταιρίας Τηλεπικοινωνιών, τα δεδομένα της οποίας θα διαχειρίζονται μέσα από μια web εφαρμογή από εξουσιοδοτημένους χρήστες. Για την κατασκευή της βάσης και της web εφαρμογής θα χρησιμοποιηθούν οι γλώσσες προγραμματισμού SQL, PHP, HTML, Javascript στο περιβάλλον WAMPP.

1.1 Σκοπός και Στόχοι της εργασίας

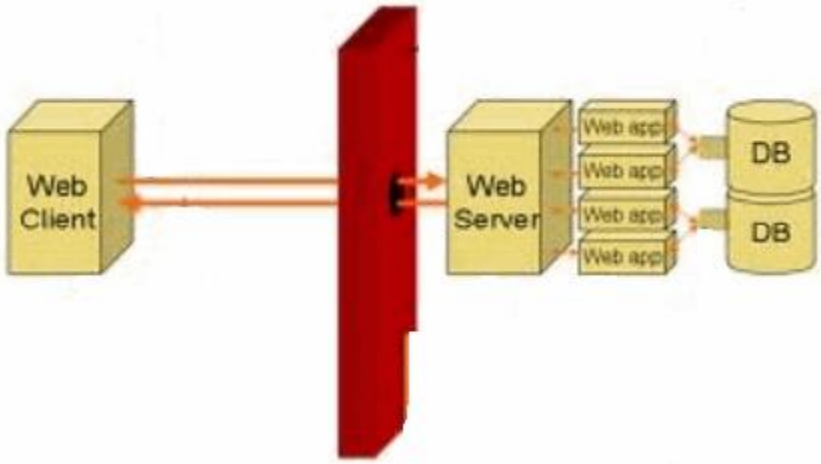
Με βάση τα παραπάνω, στους στόχους της εργασίας αυτής περιλαμβάνονται τα ακόλουθα:

1. Η θεωρητική ανάλυση των κινδύνων ασφάλειας και των αδυναμιών των βάσεων δεδομένων.
2. Η παρουσίαση και αξιολόγηση διάφορων μηχανισμών και εργαλείων για ασφάλεια των δεδομένων μιας βάσης.

3. Η υλοποίηση μιας βάσης δεδομένων, τα στοιχεία της οποίας θα διαχειρίζονται μέσω μιας web εφαρμογής και η οποία θα ενσωματώνει μηχανισμούς ασφάλειας των δεδομένων της.

1.2 Αρχιτεκτονική Εφαρμογής

Για τη διαχείριση των δεδομένων της βάσης που υλοποιείται στα πλαίσια της εργασίας δημιουργήθηκε μια Web εφαρμογή. Στο σχήμα 1 φαίνεται η αρχιτεκτονική μιας Web εφαρμογής που χρησιμοποιείται για το σκοπό αυτό. Παρακάτω θα περιγραφεί κάθε κομμάτι της εφαρμογής που φαίνεται στο σχήμα 1 και πώς αυτά συνεργάζονται για να έχουν οι ιστοσελίδες την τελική τους μορφή.



Σχήμα 1. Αρχιτεκτονική Εφαρμογής

1.2.1 Web Client (Πελάτης)

Ο Web Client χρησιμοποιεί μια εφαρμογή που ονομάζεται web browser. Μέσω του πρωτοκόλλου HTTP επικοινωνεί με τον web server και χρησιμοποιώντας τον web browser εμφανίζει τις HTML σελίδες που του στέλνει ο web server στην οθόνη. Web Client είναι για παράδειγμα οι προσωπικοί ή και φορητοί υπολογιστές.

1.2.2 Web Server (Διακομιστής ή Εξυπηρετητής)

Ο Web Server είναι ένας υπολογιστής που τρέχει λογισμικό το οποίο του δίνει τη δυνατότητα να απαντά σε αιτήσεις για έγγραφα και άλλα δεδομένα [7]. Τα προγράμματα που ζητούν και αναπαριστούν τα έγγραφα (όπως οι browsers) είναι οι clients (πελάτες) και μπορεί να τρέχουν στον ίδιο υπολογιστή με τον server ή σε σύνδεση μέσω δικτύου. Όταν ο Web Server λάβει ένα αίτημα για έγγραφα ή δεδομένα επεξεργάζεται το αίτημα αυτό και το μεταβιβάζει στην web εφαρμογή (web application) για επεξεργασία. Ανάλογα με το αίτημα η web εφαρμογή επιστρέφει την απάντηση και ο Web Server την στέλνει στον client. Κάποιοι από τους πιο γνωστούς Web Servers είναι οι: NCSA, Apache, CERN, Netscape και ο IIS (Internet Information Server) της Microsoft.

1.2.3 Web Application (Εφαρμογή Διαδικτύου)

Οι Web εφαρμογές είναι τα προγράμματα που επιτρέπουν στους clients να υποβάλουν αιτήματα για εισαγωγή ή ανάκτηση δεδομένων από τη βάση. Η web εφαρμογή επεξεργάζεται τα αιτήματα που λαμβάνει από τους χρήστες μέσω ένος server, όπως αναφέρθηκε και παραπάνω, ανατρέχει στη βάση δεδομένων (εάν αυτό φυσικά είναι απαραίτητο) και στη συνέχεια δημιουργεί μια προσαρμοσμένη σελίδα, την οποία μέσω του web server βλέπουν τελικά οι clients.

1.2.4 Data Base (Βάση Δεδομένων)

Η Βάση Δεδομένων είναι ένα σύστημα αποθήκευσης και διαχείρισης δεδομένων, η ασφάλεια του οποίου είναι το κύριο θέμα αυτής της εργασίας. Με τη βάση δεδομένων συνδέεται η Web εφαρμογή (ιδανικά μετά από ικανοποιητικούς ελέγχους πρόσβασης που εξασφαλίζουν την ασφάλεια των δεδομένων της) ώστε η δεύτερη να απαντήσει στα αιτήματα των clients. Για την εκπόνηση της εργασίας χρησιμοποιήθηκε, όπως αναφέρεται και στην επόμενη ενότητα, το Σύστημα Διαχείρισης Βάσεων Δεδομένων MySQL. Η δομή της βάσης, το είδος των δεδομένων που θα αποθηκεύει καθώς και η σύνδεση των δεδομένων με την Web εφαρμογή παρουσιάζονται στο κεφάλαιο 5.

1.3 Εργαλεία που χρησιμοποιήθηκαν για την εφαρμογή

Για την δημιουργία της εΒάσης και την ανάπτυξη της Web εφαρμογής χρησιμοποιήθηκε η πλατφόρμα **WAMP**. Το WAMP (Windows, Apache, MySQL and PHP) είναι ενα ολοκληρωμένο πακέτο server το οποίο παρέχει τα τέσσερα βασικά στοιχεία ενός Web server: μια βάση δεδομένων (MySQL), ένα Web Server (Apache), το λογισμικό για Web scripting (PHP) και τρέχει σε περιβάλλον Windows. Το WAMP περιλαμβάνει τις τελευταίες εκδόσεις του Apache, της MySQL και της PHP, ενώ περιλαμβάνει και αλλα χρήσιμα εργαλεία όπως Xdebug, PhpMyadmin, SQLBuddy, webGrind. Τα τρία βασικά εργαλεία που χρειαζόμαστε για την εφαρμογή (APACHE, MySQL & PHP) θα μπορούσαμε να τα βρούμε στο δίκτυο (είναι open source λογισμικό), αλλά η διαδικασία του να τα κατεβάσουμε και να τα εγκαταστήσουμε ένα ένα είναι πιο δύσκολη και χρονοβόρα. Το WAMP εγκαθιστά εύκολα και γρήγορα όλα τα απαραίτητα προγράμματα.

1. 3.1 APACHE

Ο Apache HTTP, γνωστός και απλά ως Apache, είναι ένας εξυπηρετητής. Ο Apache είναι ένας από τους δημοφιλέστερους εξυπηρετητές ιστού, εν μέρει γιατί λειτουργεί σε διάφορες πλατφόρμες όπως τα Windows, το Linux, το Unix και το Mac OS X. Κυκλοφόρησε υπό την άδεια λογισμικού Apache και είναι λογισμικό ανοιχτού κώδικα. Συντηρείται από μια κοινότητα ανοικτού κώδικα με επιτήρηση από το Ίδρυμα Λογισμικού Apache (Apache Software Foundation).
Ο Apache χρησιμοποιείται και σε τοπικά δίκτυα σαν διακομιστής συνεργαζόμενος με συστήματα διαχείρισης Βάσης Δεδομένων οπως Oracle, MySQ. Στο WAMP περιλαμβάνεται η έκδοση 2.4.4 του Apache.

1. 3.2 MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακής βάσης ανοικτού κώδικα, όπως λέγεται (relational database management system - RDBMS), που χρησιμοποιεί την Structured Query

Language (SQL), την πιο γνωστή γλώσσα για την προσθήκη, την πρόσβαση και την επεξεργασία δεδομένων σε μία Βάση Δεδομένων.
Είναι ανοικτού κώδικα (open source) και οποιοσδήποτε μπορεί να κατεβάσει την MySQL και να την διαμορφώσει σύμφωνα με τις ανάγκες του, σύμφωνα πάντα με την γενική άδεια που υπάρχει. Η MySQL είναι γνωστή κυρίως για την ταχύτητα, την αξιοπιστία και την ευελιξία που παρέχει.
Η MySQL αυτή τη στιγμή μπορεί να λειτουργήσει σε περιβάλλον Linux, Unix, και Windows.
Το WAMP περιλαμβάνει την έκδοση 5.6.12 της MySQL για την διαχείριση των βάσεων δεδομένων.

1.3.3 PHP

Η PHP είναι μια γλώσσα προγραμματισμού για τη δημιουργία σελίδων web με δυναμικό περιεχόμενο. Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή του Παγκόσμιου Ιστού (π.χ. Apache), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο που θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα HTML.

Η PHP (PHP Hypertext Preprocessor), προεπεξεργαστής υπερκειμένου PHP, είναι μια γλώσσα script, ο κώδικας της οποίας ενσωματώνεται σε μια HTML σελίδα και εκτελείται στην πλευρά του διακομιστή, είναι δηλαδή μια server side scripting language, όπως συνηθίζεται να λέγεται. Ο PHP κώδικας μεταφράζεται στον Web Server και δημιουργεί HTML έξοδο την οποία βλέπει ο επισκέπτης (client).

Η PHP όπως και η MySQL είναι μια ανοικτού κώδικα (open source) γλώσσα, οπότε και αυτήν μπορούμε να την εγκαταστήσουμε και να τη χρησιμοποιήσουμε δωρεάν. Η έκδοση της PHP που περιλαμβάνεται στο WAMP είναι η 5.4.12.

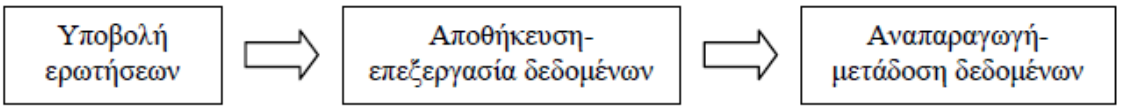
1.4 Δομή της εργασίας

Η εργασία αυτή είναι χωρισμένη σε 6 κεφάλαια. Στο $2^{ο}$ κεφάλαιο γίνεται αναφορά στις απαιτήσεις ασφάλειας των βάσεων δεδομένων και ανάλυση των κυριότερων από αυτές. Στο $3^{ο}$ κεφάλαιο περιγράφονται οι Απειλές που μπορεί να δεχθεί μια βάση δεδομένων και πως αυτές μπορούν να οδηγήσουν στην παραβίαση της ασφάλειας, και κατόπιν γίνεται αναφορά σε κάποια μέτρα που μπορούν να ληφθούν για την προστασία της βάσης. Στο $4^{ο}$ κεφάλαιο γίνεται αναλυτική περιγραφή των εργαλείων που χρησιμοποιήθηκαν για την εφαρμογή που θα παρουσιαστεί στην εργασία και κάποιων θεμάτων ασφάλειας για αυτά. Στο $5^{ο}$ κεφάλαιο περιγράφεται η δομή της εφαρμογής, οι λειτουργίες που εκτελεί και γίνεται αναφορά σε κάποια μέτρα ασφάλειας που λαμβάνονται για την προστασία της βάσης και της web εφαρμογής μέσω της οποίας διαχειρίζονται τα στοιχεία της. Τέλος, στο κεφάλαιο 6 καταγράφονται τα συμπεράσματα από την εκπόνηση αυτής της εργασίας.

Κεφάλαιο 2°

2. Ανάλυση Απαιτήσεων

Η βάση δεδομένων είναι για τους χρήστες της ένα εργαλείο για να αποθηκεύουν, να επεξεργάζονται και να μεταδίδουν δεδομένα, σύμφωνα με το μοντέλο που παρουσιάζεται στο ακόλουθο σχήμα:



Σχήμα 2. Μοντέλο Επεξεργασίας δεδομένων [11]

Τόσο κατά τη φάση της επεξεργασίας, όσο και κατά τη φάση της μετάδοσης χρησιμοποιούνται

ειδικοί μηχανισμοί για τη διασφάλιση ότι:

1. οι δοσοληψίες θα εκτελεστούν στο σύνολό τους ή καθόλου
2. θα εφαρμόζονται όλοι οι κανόνες ακεραιότητας που έχουν ορισθεί για τη βάση δεδομένων.

Οι απαιτήσεις ασφάλειας ενός συστήματος διευκρινίζονται με τη βοήθεια μιας πολιτικής ασφαλείας και επιβάλλονται από τους μηχανισμούς ασφάλειας. Οι απαιτήσεις αυτές μπορεί να είναι διαφορετικές για κάθε Σύστημα Βάσεων Δεδομένων. Ένα σύστημα μπορεί να χαρακτηριστεί σύστημα Υψηλού ή Χαμηλού Κινδύνου (High or Low Risk System) ανάλογα με το είδος των στοιχείων της βάσης, το επίπεδο συσχετισμού τους, τον τρόπο διαχείρισής τους, τη δυνατότητα πρόσβασης των χρηστών ή ακόμα και τους κανονισμούς και τις απαιτήσεις λειτουργικότητας της βάσης. Για να αποφασιστεί η πολιτική ασφάλειας και να καθοριστούν οι μηχανισμοί ασφάλειας για το σύστημα βάσεων δεδομένων θα πρέπει να γίνει ανάλυση των απαιτήσεων ασφάλειας με βάσει τους παράγοντες που αναφέρθηκαν παραπάνω. Παρόλο που πρακτικά για κάθε σύστημα οι απαιτήσεις είναι διαφορετικές, υπάρχουν κάποιες βασικές απαιτήσεις που θα πρέπει να ικανοποιεί κάθε σύστημα βάσεων δεδομένων. Οι κυριότερες από αυτές είναι [10] :

1. **Φυσική ακεραιότητα της βάσης (physical database integrity)**
2. **Λογική ακεραιότητα της βάσης (logical database integrity)**
3. **Ακεραιότητα των πεδίων της βάσης (element integrity)**
4. **Πιστοποίηση των χρηστών (user authentication)**

5. **Διαθεσιμότητα (availability)**

6. **Εμπιστευτικότητα (confidentiality)**

7. **Έλεγχος προσπέλασης (access control)**

8. **Ασφάλεια/Πεποίθηση (assurance)**

9. **Πληρότητα Δεδομένων (data integrity)**

10. **Εξ' αναφοράς Πληρότητα (referential integrity)**

11. **Πληρότητα Ύπαρξης**

2.1 Ακεραιότητα

Μια από τις πιο βασικές απαιτήσεις από των συστημάτων βάσεων δεδομένων είναι η ακεραιότητα των δεδομένων. Τα δεδομένα πρέπει να διασώζονται σε περιπτώσεις βλαβών υλικού και δυσλειτουργιών του λογισμικού (στο μέτρο του δυνατού βεβαίως), οι τροποποιήσεις πρέπει να γίνονται μόνο από εξουσιοδοτημένους χρήστες και κάθε φορά να επιστρέφονται τα δεδομένα που έχουν αποθηκευτεί. Αν υπάρχει οποιαδήποτε παραβίαση της ακεραιότητας, οι ενδιαφερόμενοι χρήστες πρέπει τουλάχιστον να ειδοποιούνται.

2.1.1 Φυσική ακεραιότητα της βάσης (physical database integrity)

Τα δεδομένα της βάσης πρέπει να είναι προστατευμένα από φυσικά προβλήματα (π.χ. πτώση τάσης ρεύματος), ούτως ώστε να είναι δυνατή η επανάκτηση των δεδομένων μετά από μια φυσική καταστροφή. Το σύστημα θα πρέπει να παρέχει μηχανισμούς ώστε κατόπιν εμφανίσεως τέτοιων περιστατικών να είναι δυνατή η ανάκαμψη από το σφάλμα και η ανάκτηση των δεδομένων.

Η τήρηση εφεδρικών αντιγράφων είναι ένας σημαντικός μηχανισμός υποστήριξης της φυσικής ακεραιότητας, η αντιγραφή δηλαδή των περιεχομένων των δίσκων όπου φυλάσσεται η βάση δεδομένων σε άλλα φυσικά μέσα αποθήκευσης (π.χ δισκετες, CD, εξωτερικούς δίσκους κ.λπ.). Για τα εφεδρικά αντίγραφα είναι σημαντικό να μπορούν να λαμβάνονται όταν η βάση δεδομένων βρίσκεται σε λειτουργία, με άλλα λόγια να είναι δυνατόν να διενεργούνται δοσοληψίες στη βάση δεδομένων όσο διαρκεί η λήψη του εφεδρικού αντιγράφου. Παράλληλα, είναι επιθυμητό να υπάρχουν ταχείες διαδικασίες ανάκαμψης, δηλαδή ο χρόνος που μεσολαβεί από την έναρξη της διαδικασίας ανάκαμψης μέχρι να τεθεί το σύστημα σε πλήρη διαθεσιμότητα να είναι κατά το δυνατόν μικρότερος.

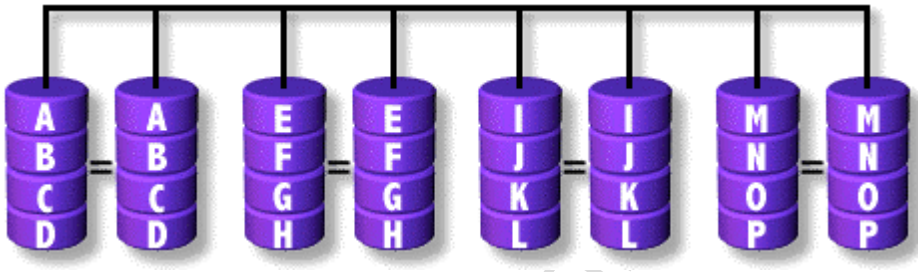
Τα εφεδρικά αντίγραφα βάσεων δεδομένων διακρίνονται σε δύο κατηγορίες, τα φυσικά αντίγραφα και τα λογικά αντίγραφα. Τα φυσικά αντίγραφα αποτυπώνουν τα περιεχόμενα των δίσκων, όπως ακριβώς τα αποθηκεύει η βάση δεδομένων, χωρίς να ενδιαφέρονται για τη λογική τους δομή. Τα φυσικά αντίγραφα λαμβάνονται σε μικρότερο χρόνο και αποκαθίστανται ταχύτερα, συνήθως όμως απαιτούν να διακόπτεται η λειτουργία της βάσης δεδομένων κατά τη λήψη τους και – ανάλογα με το σύστημα βάσης δεδομένων – είναι πιθανόν να μπορούν να λειτουργήσουν μόνο σε σύστημα «όμοιο» με αυτό από το οποίο ελήφθησαν. Τα λογικά αντίγραφα αποτυπώνουν τα δεδομένα της βάσης σε μορφή που αντικατοπτρίζει τη λογική τους δομή, χωρίς να αποτυπώνουν τον επακριβή τρόπο αποθήκευσης των δεδομένων στους δίσκους. Απαιτούν περισσότερο χρόνο για να ληφθούν και η αποκατάστασή τους διαρκεί περισσότερο, ωστόσο μπορούν εν γένει να λαμβάνονται ακόμα και ενώ η βάση δεδομένων

λειτουργεί και μπορούν να λειτουργήσουν και σε συστήματα «ανόμοια» προς αυτό από το οποίο ελήφθησαν.

Μία δεύτερη τεχνική για την υποστήριξη της φυσικής ακεραιότητας είναι η χρήση τεχνολογίας **RAID (Redundant Array of Inexpensive Disks)** [wikipedia]. Η τεχνολογία RAID χρησιμοποιεί πλεονάζοντες δίσκους για την αποθήκευση των δεδομένων. Αυτό γίνεται αντιγράφοντας στους επιπλέον δίσκους αθροίσματα ελέγχου και διόρθωσης. Η αντιγραφή γίνεται έτσι ώστε να εξασφαλίζεται ότι αν σε κάποιον από τους δίσκους προκληθεί βλάβη το σύστημα μπορεί να συνεχίσει τη λειτουργία του. Η τεχνολογία **RAID** - για ότι έχει να κάνει με την ασφάλεια - χωρίζεται στις εξής τρεις κατηγορίες :

- **RAID 1** – χρήση κατοπτρικών δίσκων

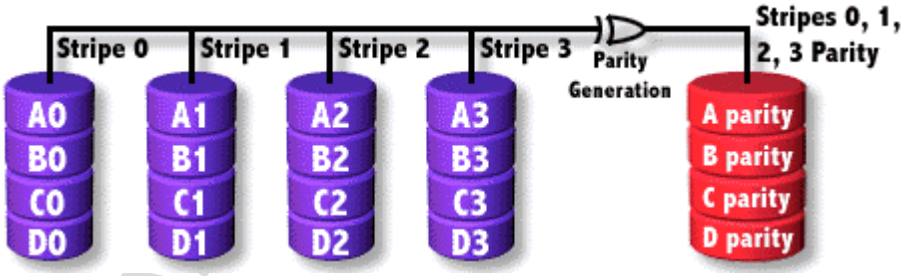
  Για κάθε δίσκο χρησιμοποιείται ένας ίσης χωρητικότητας. Το σύστημα αντιγράφει τα δεδομένα και στους δύο δίσκους, όπως φαίνεται στο σχήμα που ακολουθεί:



Σχήμα 3. Χρήση RAID 1 [11]

- **RAID 3** – παράλληλη μεταφορά και ψηφία ισοτιμίας

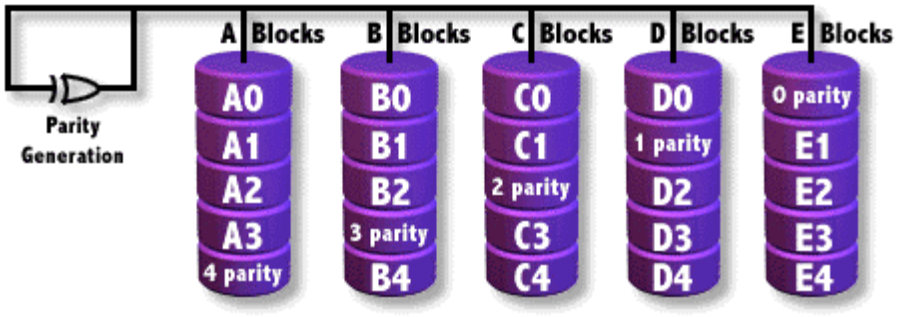
  Τα δεδομένα διαμοιράζονται σε πολλαπλούς δίσκους (stripe 0-3 στην εικόνα που ακολουθεί) και ψηφία ισοτιμίας δημιουργούνται και εγγράφονται στον πλεονάζοντα δίσκο:



Σχήμα 4. Χρήση RAID 3 [11]

- **RAID 5** – ανεξάρτητοι δίσκοι δεδομένων με κατανεμημένα ψηφία ισοτιμίας

  Κάθε φυσικός δίσκος περιέχει τόσο διαμερίσεις δεδομένων όσο και διαμερίσεις ψηφίων ισοτιμίας. Η ανεξαρτησία των δίσκων επιτρέπει τη μεγιστοποίηση της ταχύτητας ανάγνωσης και την αύξηση της ευελιξίας.



Σχήμα 5. Χρήση RAID 5 [11]

2.1.2 Λογική ακεραιότητα της βάσης (logical database integrity)

Πρέπει να διατηρείται σε κάθε περίπτωση η λογική ακεραιότητα της βάσης. Για παράδειγμα η διατήρηση της λογικής ακεραιότητας της βάσης εγγυάται ότι η μεταβολή της τιμής ενός από τα πεδία της δεν επηρεάζει τις τιμές των άλλων πεδίων, παρά μόνο εφόσον κάτι τέτοιο έχει προβλεφθεί. Οι ακόλουθες ενότητες περιγράφουν βασικούς μηχανισμούς οι οποίοι βοηθούν στην διατήρηση της λογικής ακεραιότητας της βάσης δεδομένων.

1. **Πεδία ορισμού**. Θα πρέπει να ελέγχεται ότι οι τιμές που αποθηκεύονται στα πεδία συμφωνούν με τον τύπο των πεδίων που έχει οριστεί από τον δημιουργό της βάσης. π.χ. δεν θα πρέπει σε πεδία χαρακτήρων να εισάγονται αριθμοί ή το αντιστροφο.
2. **Αποδοχή ή όχι τιμών null**. Σε πεδία τα οποία έχουν οριστεί ώστε να μην είναι κενά (null) δεν μπορεί να μην εισάγονται τιμές, π.χ. [11]

```
customerId varchar(20) not null
```

η εντολή ορίζει ότι το πεδίο customerId δεν μπορεί να είναι κενό. Πρέπει η βάση δεδομένων να εξασφαλίζει ότι για τα πεδία αυτά θα παρασχεθούν τιμές κατά την εισαγωγή των δεδομένων.

3. **Εύρος τιμών.** Ορισμένα δεδομένα ορίζονται ώστε να δέχονται τιμές μέσα από ένα συγκεκριμένο πεδίο ορισμού, μέσα απο εντολές ελέγχου, π.χ [11]

```
grade number(2) check(grade >= 0 and grade <= 10)
isSecure char(1) check(isSecure = 'y' or isSecure = 'n')
```

4. **Σχέση μεταξύ τιμών διαφορετικών πεδίων στην ίδια εγγραφή.** Για τις τιμές των πεδίων υπάρχουν κάποιοι λογικοί περιορισμοί ή και περιορισμοί που ορίζονται από τους σχεδιαστές (π.χ. αριθμιτικοί) στις μεταξύ τους σχέσεις π.χ. [11]

```
check (endDate is NULL or endDate > startDate)
```

5. **Έλεγχος μοναδικότητας τιμών.** Σε κάθε πίνακα υπάρχουν πεδία τα οποία χαρακτηρίζουν τις εγγραφές και πρέπει να είναι μοναδικά (π.χ. το πρωτεύον κλειδί). Αυτό θα πρέπει να ορίζεται από τη βάση δεδομένων. π.χ. [11]

```
customerId number(10, 0) unique
countryId varchar(3), passportNo varchar(10)
primary key(countryId, passportNo)
```

6. **Έλεγχος αναφοράς τιμών.** Οι τιμές που αποθηκεύονται σε έναν πίνακα πρέπει να υπάρχουν σε στήλη/στήλες του άλλου πίνακα [11]:

```
countryId varchar(3) references coyntry(countryId)
foreign key(countryId, passportNo)
references immigrant(countryId, passportNo)
on update cascade
```

7. **Έλεγχος μεταβάσεων κατάστασης.** Όταν υπάρχουν διαδικασίες που επεμβαίνουν στα δεδομένα της βάσης και τα αλλάζουν θα πρέπει να ελέγχεται ότι οι τιμές που προκύπτουν από την αλλαγή αυτή ικανοποιούν τους ορισμούς που έχουν οριστεί. Για παράδειγμα αν σε μια εταιρία θέλουν τα έξοδα να μην υπερβαίνουν τα 10.000€ θα πρέπει κάθε φορά που αυξάνεται το ποσό των εξόδων να ελέγχεται εάν ικανοποιεί την απαίτηση της εταιρίας για παράδειγμα ένας εταιρικός κανόνας απαγορεύει την αύξηση μισθών άνω του 20%, ο μόνος τρόπος να διασφαλίσουμε την εφαρμογή του κανόνα είναι να ελέγχουμε την κατάσταση της βάσης πριν και μετά την ενημέρωση:

```
if (costs > 10.000) then
raise_application_error(10, 'Costs are grater than 10.000€')
```

Οι δοσοληψίες είναι ένας μηχανισμός απαραίτητος για την εξασφάλιση της ακεραίοτητας των δεδομένων. Οι δοσοληψίες είναι πακέτα εντολών τα οποία θα εκτελεστούν είτε στο σύνολό τους (όλες οι εντολές που περιλαμβάνουν) είτε καθόλου (καμία από τις εντολές). Για την υλοποίηση των δοσοληψιών χρησιμοποιούνται κυρίως τεχνικές τήρησης ημερολογίων, οι οποίες διακρίνονται σε δύο βασικές κατηγορίες [11]:

- Ημερολόγια αναίρεσης με πρότερη εγγραφή των δεδομένων. Τα δεδομένα γράφονται άμεσα, ενώ στο ημερολόγιο φυλάσσονται πληροφορίες σχετικά με το ποιες πράξεις εγγραφής πρέπει αναιρεθούν (και πως) για να καταργηθούν συνολικά ημιτελείς δοσοληψίες.

- Ημερολόγια επανάληψης με ύστερη εγγραφή των δεδομένων. Τα δεδομένα γράφονται μόνο μετά το πέρας της δοσοληψίας, ενώ στο ημερολόγιο τηρούνται πληροφορίες για τις εγγραφές που πρέπει να γίνουν για να ολοκληρωθούν ημιτελείς δοσοληψίες.

Ένας άλλος μηχανισμός ακεραιότητας είναι ο έλεγχος ταυτοχρονισμού. Ο έλεγχος ταυτοχρονισμού φροντίζει ώστε να μην υπάρχουν παρεμβολές ανάμεσα σε δοσοληψίες που προσπελαύνουν ταυτόχρονα τα ίδια δεδομένα. Ως παράδειγμα θεωρήστε τις δύο ακόλουθες δοσοληψίες [11]:

```
select sum (balance) from account;
update account set balance = balance + interest, interest = 0
```

η πρώτη από τις οποίες υπολογίζει το συνολικό ταμειακό υπόλοιπο όλων των λογαριασμών, ενώ η δεύτερη κεφαλαιοποιεί τους τόκους. Για τις δοσοληψίες αυτές είναι σημαντικό η δοσοληψία ανάγνωσης (υπολογισμού συνολικού ταμειακού υπόλοιπου) να «διαβάσει» όλα τα ταμειακά υπόλοιπα είτε πριν την κεφαλαιοποίηση είτε μετά την κεφαλαιοποίηση, και σε καμία περίπτωση μερικά πριν και μερικά μετά, καθώς αυτό θα οδηγήσει σε εννοιολογικά εσφαλμένο αποτέλεσμα. Για τον έλεγχο ταυτοχρονισμού χρησιμοποιούνται τεχνικές κλειδώματος δεδομένων (όπου η κάθε δοσοληψία αποκλείει τις άλλες από συγκρουόμενες προσβάσεις στα δεδομένα που αυτή χρησιμοποίησε μέχρι την περάτωσή της) ή τεχνικές χρονοσήμων (όπου η κάθε δοσοληψία σημαδεύει χρονικά τα δεδομένα που προσπελαύνει, ελέγχοντας παράλληλα και τα χρονόσημα που έθεσαν άλλες δοσοληψίες).
Ένα τελευταίο σημείο που σχετίζεται με τη λογική ακεραιότητα της βάσης δεδομένων είναι ο χρόνος διενέργειας του ελέγχου για το αν τηρούνται οι περιορισμοί ακεραιότητας. Οι περισσότεροι έλεγχοι μπορούν να διενεργηθούν κατά τη στιγμή της ενημέρωσης της βάσης δεδομένων – π.χ. η τιμή ενός πεδίου είτε θα είναι είτε δεν θα είναι σε μία ορισμένη περιοχή. Μία συγκεκριμένη κατηγορία ελέγχων ωστόσο μπορεί να διενεργηθεί μόνον στο πέρας της δοσοληψίας. Ας θεωρήσουμε ως παράδειγμα [11] τον περιορισμό «μία εταιρεία πρέπει να έχει ακριβώς έναν πρόεδρο», που υλοποιείται με τις κάτωθι εντολές:

```
select count(*) into :numPresidents
from employee where position = 'President';
if (:numPresidents <> 1) then /* error */ ...
```

Η αλλαγή προέδρου μπορεί να γίνει με τις εντολές:

```
update employee set position = 'President'
where empId = :newPresident;
```

Παρατηρήστε ότι μετά την πρώτη εντολή ο περιορισμός δεν ισχύει (δεν υπάρχει κανείς εργαζόμενος με θέση προέδρου), οπότε αν το σύστημα διενεργήσει άμεσα τον έλεγχο θα απορρίψει την ενημέρωση. Η αντιστροφή της σειράς των εντολών δεν είναι λύση, καθώς μετά την εκτέλεση της πρώτης εντολής και πάλι ο περιορισμός δεν θα ισχύει (θα υπάρχουν δύο πρόεδροι). Η μόνη λύση είναι να διενεργείται ο έλεγχος μόνο μετά το πέρας και των δύο εντολών οι οποίες θα πρέπει να περιέχονται σε μία δοσοληψία.

2.1.3 Ακεραιότητα των πεδίων της βάσης (element integrity)

Η Ακεραιότητα των πεδίων της βάσης εγγυάται ότι οι τιμές των επί μέρους πεδίων της βάσης είναι ακριβείς (σωστές).

Η ακεραιότητα των στοιχείων της βάσης είναι η ορθότητα τους ή η ακρίβεια τους. Οι εξουσιοδοτημένοι χρήστες είναι υπεύθυνοι για την εισαγωγή σωστών δεδομένων στη βάση. Παρ' όλα αυτά, οι χρήστες κάνουν λάθη στην συλλογή των δεδομένων, στην επεξεργασία των αποτελεσμάτων και στην εισαγωγή των τιμών. Τα Σ.Δ.Β.Δ. μπορούν να βοηθήσουν τον χρήστη να αντιληφθεί τα δεδομένα καθώς αυτά εισέρχονται και να τα διορθώσει.Αυτό μπορεί να γίνει με τρεις τρόπους:

1. Μπορεί να εφαρμοστεί έλεγχος πεδίων (field checks), που κάνει έλεγχο για τις κατάλληλες τιμές σε μια θέση. Ένα πεδίο μπορεί να παίρνει μόνο αριθμητικές τιμές ή πρέπει να είναι μεγαλύτερο από το άθροισμα δυο άλλων πεδίων.
2. Με τον έλεγχο πρόσβασης (access control). Μια βάση δεδομένων πρέπει να διατηρεί δεδομένα από διαφορετικές πηγές. Πριν την ανάπτυξη των βάσεων δεδομένων, πλεονασματικά δεδομένα μπορεί να έχουν αποθηκευτεί σε διαφορετικά μέρη. Έτσι σε κάθε αλλαγή ενός δεδομένου έπρεπε να ενημερωθούν όλες οι πηγές, ενώ με τις βάσεις αυτό μπορεί να γίνει μόνο με μια κεντρική ενημέρωση και όλοι οι χρήστες να έχουν τα σωστά δεδομένα.

3. Ο τρίτος τρόπος για τη διατήρηση ενός στοιχείου μιας βάσης είναι ένα πρόχειρο αλλαγής για τη βάση (change log). Το ημερολόγιο αλλαγών είναι μια λίστα για κάθε αλλαγή που συμβαίνει στη βάση. Το ημερολόγιο περιέχει και τις αυθεντικές και τις τροποποιημένες τιμές. Με αυτό το ημερολόγιο ο διαχειριστής της βάσης μπορεί να αναιρέσει οποιαδήποτε αλλαγή που ήταν λαθεμένη.

2.2 Πιστοποίηση των χρηστών (user authentication)

Πριν επιτραπεί στο χρήστη η πρόσβαση στη βάση δεδομένων πρέπει να καθοριστεί η ταυτότητά του (identity) με ένα αξιόπιστο τρόπο, και μετά να ελεγχθεί αν ο συγκεκριμένος είναι εξουσιοδοτημένος χρήστης για την βάση δεδομένων. Υπάρχουν δυο επιλογές μηχανισμών για την αναγνώριση και αυθεντικοποίηση των χρηστών που είναι οι εξής:

1. Από το μηχανισμό αυθεντικοποίησης του server της βάσης δεδομένων.
2. Από το σύστημα ασφάλειας του λειτουργικού συστήματος.

2.2.1 Πιστοποίηση μέσω της Βάσης Δεδομένων

Κάθε χρήστης έχει ένα όνομα (username) και ένα συνθηματικό (password). Αυτό το συνθηματικό μπορεί να αποθηκεύεται μέσα στην βάση δεδομένων χρησιμοποιώντας έναν αλγόριθμο κρυπτογράφησης DES (Data Encryption Standard). Για να συνδεθεί με τη βάση δεδομένων ένας εξουσιοδοτημένος χρήστης του λειτουργικού συστήματος πρέπει να την τροφοδοτήσει με το δικό του username και password της βάσης.

Στα κατανεμημένα συστήματα, ένα συνθηματικό που περνάει από το ένα μηχάνημα στο άλλο μπορεί να θέσει την ασφάλεια σε κίνδυνο. Εάν το συνθηματικό περνάει σε καθαρό κείμενο, δηλαδή δεν έχει κρυπτογραφηθεί, οποιοσδήποτε που παραφυλάει να κρυφακούσει τα δεδομένα μπορεί να διαβάσει και το συνθηματικό και έτσι να γίνει εξουσιοδοτημένος χρήστης.

Για το λόγο αυτό κάποια συστήματα χρησιμοποιούν ένα κλειδί για να κρυπτογραφήσουν το συνθηματικό του χρήστη πριν αυτό διέλθει στο server. Κάθε προσπάθεια σύνδεσης χρησιμοποιεί ένα ξεχωριστό κλειδί για την κρυπτογράφηση, κάνοντας την κρυπτογράφηση πιο δύσκολα να αποκρυπτογραφηθεί. Αφού το κρυπτογραφημένο με το κλειδί συνθηματικό περάσει στον server, τότε αυτός το αποκρυπτογραφεί και το επανακρυπτογραφεί χρησιμοποιώντας τον αλγόριθμο DES και συγκρίνει αυτό με το συνθηματικό που είναι ήδη καταχωρημένο στη βάση. Εάν αυτά ταιριάζουν, τότε ο χρήστης συνδέεται επιτυχώς με τη βάση.

2.2.2 Πιστοποίηση Μέσω Του Λειτουργικού Συστήματος

Κάθε χρήστης έχει ένα όνομα (username) για τη βάση δεδομένων, αλλά δεν έχει συνθηματικό. Για να συνδεθεί στη βάση δεδομένων ένας κατάλληλα εξουσιοδοτημένος χρήστης του λειτουργικού συστήματος δεν είναι απαραίτητο να διευκρινίσει το username και password της βάσης. Αντί για αυτό, όταν ο χρήστης προσπαθεί να συνδεθεί στην βάση, το εξουσιοδοτημένο username του λειτουργικού συστήματος μεταβιβάζεται από το ασφαλές λειτουργικό σύστημα στο σύστημα της βάσης δεδομένων. Το σύστημα της βάσης επαληθεύει ότι το username που δόθηκε είναι εξουσιοδοτημένο να συνδεθεί στην βάση.

Ένα από τα πλεονεκτήματα της χρήσης της αυθεντικοποίησης μέσω του λειτουργικού συστήματος αντί αυτής μέσω της βάσης δεδομένων είναι ότι το username που έχει ο χρήστης είναι το ίδιο και στο λειτουργικό σύστημα και στη βάση δεδομένων. Αυτό απομακρύνει την ανάγκη να διατηρούνται ξεχωριστά οι πληροφορίες αναγνώρισης και αυθεντικοποίησης του χρήστη και στο λειτουργικό και στον server της βάσης και είναι πιο εύκολο στον χρήστη να λογοδοτεί για τις ενέργειες του μέσα στο σύστημα.

2.2.3 Σύγκριση Μεταξύ ΣΔΒΔ και ΛΣ Για Την Ασφάλεια

Και το σύστημα διαχείρισης βάσης δεδομένων και το λειτουργικό σύστημα είναι εξίσου υπεύθυνα για την προστασία των πόρων έναντι της μη εξουσιοδοτημένης αναφοράς και τροποποίησης. Παρ' όλα αυτά, υπάρχουν διαφορές που κάνουν τους μηχανισμούς προστασίας

του λειτουργικού συστήματος να μην είναι εξολοκλήρου κατάλληλοι για την ασφάλεια της βάσης δεδομένων. Έτσι:

1. Το λειτουργικό σύστημα προστατεύει τους φυσικούς πόρους, όπως για παράδειγμα, αρχεία, σελίδες, τους κυλίνδρους του δίσκου και τις μαγνητικές ταινίες. Το σύστημα διαχείρισης βάσης δεδομένων χρειάζεται να προστατεύσει λογικές πηγές, όπως για παράδειγμα συσχετίσεις, εγγραφές και τμήματα.
2. Το λειτουργικό σύστημα προστατεύει τους πόρους του καθ' ολοκληρία, ενώ το σύστημα διαχείρισης βάσης δεδομένων χρειάζεται να έχει την δυνατότητα να προστατεύει μέρος των πόρων, όπως είναι τα πεδία, οι εγγραφές και τα υποσχήματα μέσα στην ίδια βάση.
3. Το λειτουργικό σύστημα κάνει διάκριση μεταξύ της πρόσβασης για διάβασμα και αυτής για γράψιμο. Το σύστημα διαχείρισης βάσης δεδομένων είναι απαραίτητο να διακρίνει μεταξύ επιπρόσθετων τύπων πρόσβασης, όπως για παράδειγμα πρόσβαση βάση του περιεχομένου και βάση των συμφραζομένων.

2.3 Διαθεσιμότητα (availability)

Η Διαθεσιμότητα εγγυάται ότι οι εξουσιοδοτημένοι χρήστες μπορούν γενικά να προσπελάσουν άμεσα την βάση και τα δεδομένα για τα οποία είναι εξουσιοδοτημένοι. Ανάλογα με τη χρήση των δεδομένων μιας βάσης μπορεί να απαιτείται υψηλή διαθεσιμότητα ή όχι. Με τον όρο υψηλή διαθεσιμότητα εννοούμε ότι η βάση δεδομένων πρέπει να είναι διαθέσιμη 24 ώρες την ημέρα 7 ημέρες την εβδομάδα. Μπόρει όμως για κάποια επιχείρηση να είναι αποδεκτό η επαναφορά της βάσης να γίνει σε μερικά λεπτά ή ακόμα και μετά από κάποιες ώρες. Οι βάσεις δεδομένων μπορεί να απειλούνται από λάθη υπαλλήλων, από βλάβες του συστήματος ή από φυσικές καταστροφές. Σε κάθε περίπτωση θα πρέπει να αποφευχθεί η απώλεια πληφορίας που μπορεί να προκληθεί από τις παραπάνω αιτίες. Για αυτό το λόγο είναι απαραίτητη η δημιουργία αντιγράφων ασφαλείας, ώστε η βάση δεδομένων να μπορεί να επαναφερθεί το συντομότερο δυνατόν και να είναι διαθέσιμη στους χρήστες. Η δημιουργία αντιγράφων σε συνδυασμό με ένα μηχανισμό άμεσης επαναφοράς ενός ΣΔΒΔ μπορεί να εγγυηθεί στους χρήστες τη διαθεσιμότητα της βάσης και την αποφυγή απώλειας πληροφορίας.

2.4 Εμπιστευτικότητα (Confidentiality)

Η Εμπιστευτικότητα εγγυάται ότι τα δεδομένα είναι προσπελάσιμα μόνο από άτομα τα οποία είναι εξουσιοδοτημένα να προσπελάσουν τα δεδομένα και τα έχουν ανάγκη για την εργασία που εκτελούν στη βάση. Η Εμπιστευτικότητα στα διάφορα συστήματα επιβάλλεται μέσω των μηχανισμών της ταυτοποίησης, της πιστοποίησης χρηστών και των υποχρεωτικών και διακριτικών μηχανισμών ελέγχου πρόσβασης.

2.5 Έλεγχος προσπέλασης (access control)

Ο έλεγχος προσπέλασης εγγυάται ότι οι χρήστες της βάσης μπορούν να προσπελάσουν μόνο τα δεδομένα εκείνα για τα οποία έχουν εξουσιοδοτηθεί. Οι διάφοροι τύποι χρηστών μπορεί έτσι να περιοριστούν σε ορισμένους χώρους και τρόπους προσπέλασης, ανάλογα με τις ανάγκες τους (π.χ. μόνο διάβασμα). Τα ΣΔΒΔ παρέχουν δύο τύπους ελέγχου προσπέλασης το διακριτικό έλεγχο προσπέλασης (Discretional Access Control, DAC) και τον υποχρεωτικό έλεγχο προσπέλασης (Mandatory Access Control, MAC) για τη διαχείριση του προνομίου προσπέλασης δεδομένων. Αν και ο διακριτικός έλεγχος προσπέλασης παρέχεται από τα

περισσότερα εμπορικά ΣΔΒΔ, είναι πολύ λίγα τα ΣΔΒΔ που υποστηρίζουν τον υποχρεωτικό και πολύ αυστηρότερο τύπο ελέγχου προσπέλασης. Στη συνέχεια θα αναλύσουμε τις δύο αυτές διαφορετικές δυνατότητες

2.5.1 Διακριτικός Έλεγχος Προσπέλασης

Ο τυπικός τρόπος εφοδιασμού διακριτικού ελέγχου πρόσβασης σε ένα σύστημα Βάσεων Δεδομένων βασίζεται στην Εκχώρηση και Ανάκληση προνομίων (privileges) μέσω εντολών SQL. Ένα προνόμιο επιτρέπει σε κάποιο χρήστη να δημιουργήσει ή να προσπελαύνει (δηλαδή να αναγνώσει, να αποθηκεύσει ή να τροποποιήσει) ένα αντικείμενο της Βάσης Δεδομένων ή συγκεκριμένες συνιστώσες του ΣΔΒΔ. Τα προνόμια εκχωρούνται στους χρήστες μόνο αν προαπαιτούνται στα πλαίσια των καθηκόντων εργασίας τους, δηλαδή αν οι χρήστες δεν μπορούν να διεκπεραιώσουν την εργασία τους χωρίς τα προνόμια αυτά. Η καταχρηστική εκχώρηση μη αναγκαίων προνομίων μπορεί να οδηγήσει σε διάρρηξη της ασφάλειας του συστήματος. Τα προνόμια που ορίζονται από το πρότυπο SQL είναι τα εξής:

- SELECT: το προνόμιο ανάκτησης δεδομένων από ένα πίνακα
- INSERT: το προνόμιο εισαγωγής νέων πλειάδων σε ένα πίνακα
- UPDATE : το προνόμιο τροποποίησης αποθηκευμένων πλειάδων σε ένα πίνακα
- DELETE: το προνόμιο διαγραφής αποθηκευμένων πλειάδων από ένα πίνακα
- REFERENCES: το προνόμιο αναφοράς γνωρισμάτων ενός συγκεκριμένου πίνακα σε περιορισμούς ακεραιότητας
- USAGE: το προνόμιο χρήσης πεδίων ορισμού.

Τα προνόμια SELECT και UPDATE μπορούν να περιορισθούν σε συγκεκριμένες στήλες ενός πίνακα, μη επιτρέποντας την ανάγνωση ή αντιστοίχως την ενημέρωση των υπολοίπων στηλών του πίνακα. Ομοίως, το προνόμιο REFERENCES μπορεί να περιορισθεί σε συγκεκριμένες στήλες ενός πίνακα, επιτρέποντας οι στήλες αυτές να σε περιορισμούς ξένου κλειδιού για τη δημιουργία ενός νέου πίνακα, χωρίς να επιτρέπει να ισχύσει αυτό και για τις υπόλοιπες στήλες του πίνακα.

Η SQL υποστηρίζει το διακριτικό έλεγχο προσπέλασης μέσω των εντολών GRANT και REVOKE. Ο χρήστης που δημιουργεί ένα πίνακα με την εντολή CREATE TABLE, γίνεται αυτομάτως ο ιδιοκτήτης (owner) του πίνακα και αποκτά αυτοδικαίως όλα τα προνόμια επί του αντικειμένου αυτού. Οι υπόλοιποι χρήστες αρχικά δεν έχουν κανένα προνόμιο στο νέο πίνακα. Για να αποκτήσουν και άλλοι χρήστες προσπέλαση στον πίνακα, πρέπει ο ιδιοκτήτης του πίνακα να τους εκχωρήσει τα απαραίτητα προνόμια μέσω της εντολής GRANT. Το ΣΔΒΔ παρακολουθεί ανελλιπώς την εκχώρηση και ανάκληση προνομίων σε άλλους χρήστες, και διασφαλίζει ότι κάθε στιγμή οι χρήστες που διαθέτουν τα κατάλληλα προνόμια θα μπορούν να προσπελάσουν ένα αντικείμενο.

Όταν ένας χρήστης δημιουργεί μια όψη με την εντολή CREATE VIEW, γίνεται αυτομάτως ο ιδιοκτήτης της όψης αλλά δεν αποκτά απαραιτήτως τα πλήρη δικαιώματα στην όψη αυτή. Για τη δημιουργία της όψης, ο χρήστης πρέπει να διαθέτει το προνόμιο της επιλογής (SELECT) σε όλους τους πίνακες που συμμετέχουν στη δημιουργία της όψης. Ωστόσο, ο ιδιοκτήτης της όψης αποκτά τα προνόμια αυτά σε κάθε πίνακα που συμμετέχει στη δημιουργία της συγκεκριμένης όψης.

Εκχώρηση Προνομίων

Όπως αναφέρθηκε και προηγουμένως, για την εκχώρηση προνομίων επί των αντικειμένων μίας Βάσης Δεδομένων σε άλλους χρήστες χρησιμοποιείται η εντολή GRANT. Συνήθως η εντολή αυτή καλείται από τον ιδιοκτήτη ενός πίνακα. Η σύνταξή της είναι η εξής:

```
GRANT <λίστα_προομίων> | ALL PRIVILEGES
ON  <λίστα_αντικειμένων>
TO  <λίστα_χρηστών | PUBLIC>
[WITH GRANT OPTION];
```

Η λίστα προνομίων σχηματίζεται από ένα ή περισσότερα από τα επόμενα προνόμια, διαχωρισμένα μεταξύ τους με κόμματα:

- SELECT [<λίστα_στηλών>]
- DELETE
- INSERT
- UPDATE [<λίστα_στηλών>]
- REFERENCES [<λίστα_στηλών>]
- USAGE

Η λίστα αντικειμένων μπορεί να περιλαμβάνει τα ονόματα ενός ή περισσοτέρων πινάκων, όψεων, πεδίων ορισμού, κ.τ.λ., η λίστα_στηλών το όνομα μίας ή περισσοτέρων στηλών ενός πίνακα ή μιας όψης και η λίστα_χρηστών το αναγνωριστικό όνομα χρήστη ενός ή περισσοτέρων χρηστών ή ομάδων χρηστών. Για ευκολία, η εντολή GRANT επιτρέπει τη χρήση της δεσμευμένης φράσης ALL PRIVILEGES για την εκχώρηση σε χρήστες όλων μαζί των προνομίων επί ενός ή περισσοτερων αντικειμένων. Η εντολή GRANT παρέχει επίσης την λέξη κλειδί PUBLIC για την εκχώρηση προνομίων σε όλους τους υπάρχοντες και μελλοντικούς εξουσιοδοτημένους χρήστες του ΣΔΒΔ.

Η παράμετρος WITH GRANT OPTION χορηγεί στους χρήστες της λίστας λίστα_χρηστών το δικαίωμα να εκχωρήσουν και σε άλλους χρήστες τα προνόμια που απέκτησαν στα αντικείμενα που αναφέρονται στη λίστα_αντικειμένων. Αν και αυτοί οι χρήστες χορηγήσουν τα προνόμια αυτά με την παράμετρο WITH GRANT OPTION, οι χρήστες που θα αποκτήσουν τα προνόμια μπορούν επίσης να τα χορηγήσουν σε άλλους χρήστες. Αν η δεσμευμένη φράση παράμετρο WITH GRANT OPTION δεν εμφανίζεται σε μια εντολή GRANT, τότε ο χρήστης που αποκτά τα συγκεκριμμένα προνόμια δεν έχει τη δυνατότητα να τα χορηγήσει σε άλλους χρήστες. Με τον τρόπο αυτό, ο ιδιοκτήτης ενός αντικειμένου διατηρεί πλήρη γνώση για το ποιοι χρήστες και με ποια προνόμια έχουν δικαιώματα χρήσης αντικειμένων που ο ίδιος δημιουργήσε.

Ανάκληση Προνομίων

Για την ανάκληση προνομίων επί αντικειμένων χρησιμοποιείται η εντολή REVOKE. Η εκτέλεση της εντολής REVOKE μπορεί να ανακαλέσει όλα ή μερικά από τα προνόμια που είχαν εκχωρηθεί παλαιότερα μέσω της εντολής. Η σύνταξη της REVOKE είναι εξής:

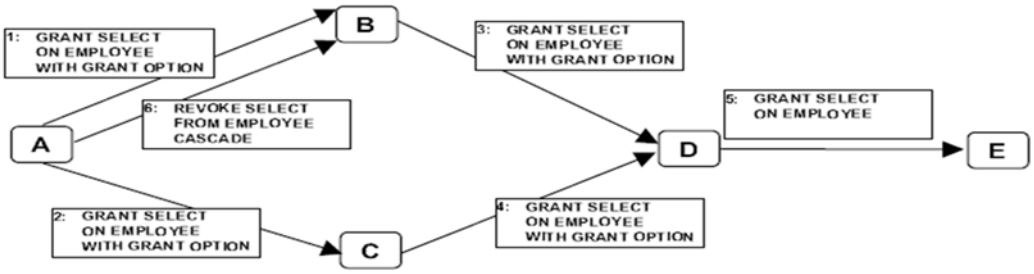
```
REVOKE [GRANT OPTION FOR] <λίστα_προνομίων>  |  ALL PRIVILEGES
ON  <λίστα_αντικειμένων>
FROM <λίστα_χρηστών>  |  PUBLIC
[RESTRICT  |  CASCADE];
```

Η έκφραση ALL PRIVILEGES αναφέρεται σε όλα τα προνόμια που είχαν χορηγηθεί στους χρήστες της λίστας λίστα_χρηστών από το χρήστη που ανακαλεί τα προνόμια αυτά. Η προαιρετική δήλωση GRANT OPTION FOR επιτρέπει το προνόμιο που είχε εκχωρηθεί μέσω της δήλωσης WITH GRANT OPTION της εντολής GRANT να ανακληθεί ξεχωριστά από τα ίδια τα προνόμια επί των αντικειμένων.

Με την προσθήκη της έκφρασης RESTRICT, η εντολή REVOKE θα αποτύχει αν το προνόμιο έχει χορηγηθεί από το χρήστη που το κατέχει και σε άλλους χρήστες ή αν το χρησιμοποίησε για τον ορισμό όψεων. Η έκφραση RESTRICT είναι η προεπιλεγμένη έκφραση. Με την προσθήκη της έκφρασης CASCADE η εντολή REVOKE θα εκτελεσθεί και αυτομάτως θα ανακαλέσει τα προνόμια από κάθε χρήστη που απέκτησε τα προνόμια αυτά από το χρήστη από τον οποίο ανακαλούνται και θα καταστραφούν όλες οι όψεις που σχηματίσθηκαν με βάση τα προνόμια αυτά.

Η εκτέλεση μίας εντολής REVOKE από ένα χρήστη A προς ένα άλλο χρήστη B, δεν επηρεάζει τα προνόμια που έχουν χορηγηθεί στο χρήστη B από άλλους τρίτους χρήστες. Συνεπώς, αν ένας χρήστης C έχει χορηγήσει στο χρήστη B το προνόμιο που ο χρήστης A ανακαλεί, τότε η εκχώρηση του προνομίου από το χρήστη C επιτρέπει στο χρήστη B να κάνει ακόμη χρήση του συγκεκριμένου προνομίου.

Στο παράδειγμα του Σχήματος (6), ο χρήστης A ως ιδιοκτήτης μίας υποθετικής σχέσης employee εκχωρεί το προνόμιο SELECT στους χρήστες B και C (βήματα 1 και 2 αντιστοίχως) με τη χρήση της παραμέτρου WITH GRANT OPTION. Οι χρήστες B και C με τη σειρά τους χορηγούν το προνόμιο αυτό στο χρήστη D (βήματα 3 και 4, αντιστοίχως). Στη συνέχεια ο χρήστης D εκχωρεί το προνόμιο SELECT επί της σχέσης employee στον χρήστη E (βήμα 5). Όταν λοιπόν ο χρήστης A ανακαλεί το προνόμιο SELECT από το χρήστη B με την παράμετρο CASCADE (βήμα 6), το προνόμιο ανακαλείται από το χρήστη B, όχι όμως και από το χρήστη D διότι ο χρήστης D έχει λάβει το ίδιο προνόμιο και από το χρήστη C . Αν ο χρήστης C δεν είχε εκχωρησει αυτό το προνόμιο στο χρήστη D, τότε η ανάκληση θα είχε διαδοχικά αφαιρέσει το προνόμιο SELECT ON employee τόσο από το χρήστη D όσο και από το χρήστη E.



Σχήμα 6. Συνέπειες χρήσης της εντολής REVOKE

Ο διακριτικός έλεγχος προσπέλασης, αν και αποτελεσματικός, διαθέτει την αχίλλειο πτέρνα του, καθώς υπάρχει τρόπος ώστε ένας μη-εξουσιοδοτημένος χρήστης να εξαπατήσει έναν εξουσιοδοτημένο χρήστη υποκλέπτοντας ευαίσθητα δεδομένα του. Για παράδειγμα, ας υποθέσουμε ότι υπάρχει ένας χρήστης A που έχει υπό την ιδιοκτησία του ένα πίνακα R για τη διατήρηση και επεξεργασία ευαίσθητων δεδομένων του και ένας κακόβουλος χρήστης B που δεν διαθέτει προνόμια προσπέλασης επί του πίνακα R. Στη συνέχεια, έστω ότι ο κακόβουλος χρήστης B δημιουργεί εναν πίνακα S και εκχωρεί το προνόμιο INSERT επί του S στον χρήστη A χωρίς μάλιστα ο χρήστης A να γνωρίζει το γεγονός αυτό. Έπειτα, ο κακόβουλος χρήστης B τροποποιεί κάποια εφαρμογή λογισμικού που χρησιμοποιεί ο εξουσιοδοτημένος χρήστης A για την επικοινωνία του με τον πίνακα R και την προσπέλαση των δεδομένων του (π.χ. ένα διαδραστικό γραφικό περιβάλλον) και ενσωματώνει κάποιες κρυφές εντολές για την αντιγραφή ευαίσθητων δεδομένων από τον πίνακα R, όπου έχει πρόσβαση μόνο ο χρήστης A, στο νέο πίνακα S που δημιουργήθηκε από το χρήστη B. Με τον τρόπο αυτό, ο χρήστης B που είναι μη εξουσιοδοτημένος για ανάγνωση του πίνακα R επιτυγχάνει τελικώς να αποκτήσει ένα αντίγραφο του πίνακα αυτού. Στο τέλος της διαδικασίας ο κακόβουλος χρήστης B, για να καλύψει πλήρως τις ενέργειες του, μπορεί να τροποποιήσει και πάλι την εφαρμογή λογισμικού που χρησιμοποιήσε ως δούρειο ίππο (trojan horse) για την εξαπάτηση του εξουσιοδοτημένου χρήστη A, επαναφέροντας την στην αρχική της μορφή, δηλαδή χωρίς τον ενσωματωμένο κρυφό κώδικα εντολών.

Είναι φανερό ότι χρειάζεται μία διαφορετική πολιτική ασφάλειας για την αντιμετώπιση αυτής της σοβαρής αδυναμίας του διακριτικού ελέγχου προσπέλασης. Η πολιτική αυτή είναι ο υποχρεωτικός έλεγχος προσπέλασης που αναλύεται στη συνέχεια.

2.5.2 Υποχρεωτικός Έλεγχος Προσπέλασης

Ο υποχρεωτικός έλεγχος προσπέλασης βασίζεται σε πολιτικές που δεν μπορούν να μεταβληθούν από τους χρήστες του ΣΔΒΔ. Κάθε αντικείμενο Βάσης Δεδομένων συνδέεται με ένα βαθμό ασφαλείας (security class) και κάθε χρήστης με ένα βαθμό εξουσιοδότησης (security clearance) για την προσπέλαση αντικειμένων συγκεκριμένων βαθμών ασφαλείας. Το ΣΔΒΔ ελέγχει αν ένας χρήστης επιτρέπεται να αναγνώσει ή να αποθηκεύσει ένα αντικείμενο με βάση κανόνες που συγκρίνουν το βαθμό εξουσιοδότησης του χρήστη με το βαθμό ασφαλείας του αντικειμένου. Αυτοί οι κανόνες διασφαλίζουν ότι τα ευαίσθητα δεδομένα δεν θα κοινοποιηθούν ποτέ σε κάποιο χρήστη που δεν διαθέτει την κατάλληλη εξουσιοδότηση. Το πρότυπο της SQL δεν περιλαμβάνει μέχρι σήμερα υποστήριξη εντολών για την επιβολή του υποχρεωτικού ελέγχου προσπέλασης.

Το πιο δημοφιλές πρωτόκολλο για την περιγραφή του υποχρεωτικού ελέγχου προσπέλασης ονομάζεται πρωτόκολλο Bell-LaPadula και βασίζεται στους γνωστούς όρους υποκείμενο, αντικείμενο, βαθμός ασφαλείας και εξουσιοδότηση ασφαλείας. Στη συνέχεια της ενότητας θα θεωρηθεί ότι ο βαθμός εξουσιοδότησης ενός υποκειμένου L ή ο βαθμός ασφαλείας ενός αντικειμένου O συμβολίζεται με την έκφραση class(L) και class(O), αντιστοίχως. Επίσης, για λόγους απλότητας θα θεωρηθεί ότι υπάρχουν τέσσερις βαθμοί ασφαλείας: ο άκρως απόρρητος (top secret, TS) ο απόρρητος (secret, S), ο εμπιστευτικός (confidential, C) και ο αδιαβάθμητος (unclassified, U). Τέλος για τη διαβάθμιση αυτή θα ορισθεί η ταξινόμηση TS > S > C > U, όπου A > B σημαίνει ότι ο βαθμός A βρίσκεται σε υψηλότερο επίπεδο ασφαλείας από τον βαθμό B.

Το πρωτόκολλο Bell – LaPadula θέτει δύο περιορισμούς σε όλες τις αναγνώσεις και αποθηκεύσεις επί των αντικειμένων της Βασής Δεδομένων:

1. **Περιορισμός Απλής Ασφαλείας** (όχι αναγνώσεις προς τα επάνω – no reads up): Στο υποκείμενο L επιτρέπεται η ανάγνωση του αντικειμένου O μόνο αν class(L) ≥ class(O). Για παράδειγμα, σε ένα υποκείμενο με εξουσιοδότηση βαθμού ασφαλείας S επιτρέπεται η ανάγνωση απορρήτων αλλά και αδιαβάθμητων δεδομένων, αλλά σε ένα υποκείμενο με εξουσιοδότηση U δεν επιτρέπεται η ανάγνωση απόρρητων δεδομένων.
2. **Περιορισμός-*** (όχι εγγραφές προς τα κάτω – no writes down): Στο υποκείμενο L επιτρέπεται η εγγραφή στο αντικείμενο O μονο αν class(L) ≤ class(O). Για παράδειγμα, σε ένα υποκείμενο με εξουσιοδότηση S επιτρέπεται η εγγραφή μόνο σε αντικείμενα με βαθμό ασφαλείας S και επάνω. Με τον περιορισμό αυτό μία εφαρμογή δούρειου ίππου δεν είναι δυνατό να υποβιβάσει το βαθμό ασφαλείας μίας απόρρητης πληροφορίας, αντιγράφοντας την και διαρρέοντας την σε αντικείμενα με χαμηλότερο βαθμό ασφαλείας.

Αν σε κάποια Βάση Δεδομένων ταυτόχρονα με το διακριτικό, καθορισθεί και ο υποχρεωτικός έλεγχος προσπέλασης, τότε οι προηγούμενοι περιορισμοί εφαρμόζονται προσθετικά. Συνεπώς, για την ανάγνωση ή αποθήκευση ενός αντικειμένου, ο χρήστης θα οφείλει να κατέχει τα κατάλληλα προνόμια που παρέχονται μέσω της εντολής GRANT της SQL και την κατάλληλη εξουσιοδότηση ασφαλείας που ορίσθηκε για το χρήστη αυτό από το διαχειριστή της Βάσης Δεδομένων.

[Πηγή παραδειγμάτων: Συστήματα Βάσεων Δεδομένων-Θεωρία και πρακτική εφαρμογή(Ι.Μ. – Α.Π.)]

2.6 Σύντομη Αναφορά στις υπόλοιπες απαιτήσεις ακεραιότητας

Ασφάλεια/Πεποίθηση (Assurance)
Παρέχει ένα βαθύτερο επίπεδο εμπιστοσύνης για τη σωστή λειτουργία και αποδοτικότητα των χαρακτηριστικών της Εμπιστευτικότητας, Ακεραιότητας και Διαθεσιμότητας του συστήματος. Η Ασφάλεια αυτή επιτυγχάνεται με τη χρήση ηχητικών και μηχανικών τεχνικών ασφαλείας.

Πληρότητα Δεδομένων (Data Integrity
Το πληροφοριακό περιεχόμενο βάσης δεδομένων είναι αποδεκτό μόνο όταν ικανοποιούνται συγκεκριμένες προϋποθέσεις πληρότητας των καταχωρημένων στοιχείων. Όταν συμβεί παραβίαση της συνθήκης πληρότητας των δεδομένων, το σύνολο του πληροφοριακού περιεχομένου της βάσης είναι άχρηστο.

Εξ' αναφοράς Πληρότητα (Referential Integrity)
Η Εξ' αναφοράς Πληρότητα έχει να κάνει με την λογική σύνδεση του περιεχομένου των πινάκων. Αυτό σημαίνει ότι η συνθήκη της εξ' αναφοράς πληρότητας επιβάλλει για την κάθε στιγμή που το ξένο κλειδί παίρνει τιμή η τελευταία να είναι ήδη καταχωρημένη στον πίνακα που περιέχει το αντίστοιχο κύριο κλειδί.

Πληρότητα Ύπαρξης
Έχει να κάνει με την δέσμευση του κάθε πίνακα να έχει ένα κύριο κλειδί, έτσι ώστε να προσδιορίζεται μονοσήμαντα η κάθε εγγραφή. Η τιμή του κύριου κλειδιού πρέπει να μην είναι NULL (δηλαδή το πεδίο να είναι κενό).

Κεφάλαιο 3°

3 Απειλές & Μέτρα Ασφάλειας ενός Συστήματος Βάσεων Δεδομένων

Οι βάσεις δεδομένων, κυρίως σε επίπεδο εταιριών/οργανισμών, περιέχουν το μεγαλύτερο – και πιο ευαίσθητο – όγκο δεδομένων κάτι που τις καθιστά πρωταρχικό στόχο για τους εισβολείς (attackers).

Σε αυτό το κεφάλαιο θα δοθούν ορισμοί για τους όρους Απειλή (threat), Αδυναμία (Vulnerability) και Exploit και θα προσπαθήσουμε να εξηγήσουμε το πως μπορούν, αν δεν αντιμετωπιστούν, να οδηγήσουν στην παραβίαση της ασφάλειας μιας βάσης δεδομένων. Επίσης θα γίνει αναφορά σε κάποια Μέτρα Ασφάλειας που μπορούν να ληφθούν από τον διαχειριστή του συστήματος, ώστε να διορθωθούν όσο είναι δυνατόν οι όποιες αδυναμίες του συστήματος και να υπάρχει η υποδομή αντιμετώπισης των απειλών που μπορεί να δεχθεί ένα ΣΔΒΔ.

3.1 Ορισμοί

Αδυναμία (Vulnerability) είναι ένα τρωτό σημείο στο λογισμικό (software), το υλικό (hardware) ή τις διαδικασίες (configurations) ενός συστήματος, το οποίο μπορεί να επιτρέψει την πρόσβαση σε μη εξουσιοδοτημένους χρήστες στους πόρους του συστήματος ή να οδηγήσει στην απώλεια δεδομένων λόγω λάθος χειρισμού από κάποιο χρήστη.

Σύμφωνα με τις τελευταίες εκθέσεις ασφαλείας κάποιων από τις κυριότερες εταιρείες ασφάλειας, οι hackers θεωρούν τα τρωτά σημεία (τις αδυναμίες) μιας βάσης δεδομένων ως τα κύρια ελαττώματα που μπορούν να εκμεταλλευτούν προκειμένου να παρακάμψουν την άμυνα του στόχου τους.

Exploit είναι μια τεχνική που εκμεταλλεύεται τις αδυναμίες ενός συστήματος για να επιτεθεί σε αυτό και να παραβιάσει την ασφάλειά του. Ο όρος αυτός αναφέρεται μιας επιτιχυμένη επίθεση αυτού του είδους. Πολλοί hackers κρατούν λίστα με τα exploits τους και κατόπιν τα γνωστοποιούν (υπερηφανευόμενοι θα έλεγε κανείς) στο δίκτυο μαζί με τις ευπάθειες που ανακάλυψαν. Όταν ένα exploit εκμεταλλεύεται μια αδυναμία σε ένα λειτουργικό σύστημα ή μια εφαρμογή, οι κατασκευαστές του συστήματος δημιουργούν ένα “patch” (κώδικα επιδιόρθωσης ή επίθεμα) προκειμένου να προστατέψουν τους χρήστες από τα exploits. Οι χρήστες του συστήματος είναι υπεύθυνοι για την εγκατάσταση αυτών των ενημερώσεων οι οποίες συνήθως είναι διαθέσιμες μέσω διαδικτύου. Η μη εγκατάσταση ενός patch εκθέτει τα συστήματα βάσεων σε πιθανές παραβιάσης ασφάλειας.

Απειλή (threat) εσωτερική ή εξωτερική είναι η πιθανή αιτία που μπορεί σε συνδυασμό με μία αδυναμία να προκαλέσει αποτυχία ασφάλειας ενός συστήματος, δηλαδή παραβίαση των δεδομένων. Οι απειλές μπορεί να είναι εκούσιες ή ακουσίες. Εκούσιες είναι οι απειλές που σχεδιάζονται και υλοποιούνται από ένα πρόσωπο με σκοπό την διάρηξη στην ασφάλεια ενός συστήματος βάσεων δεδομένων. Για παράδειγμα η ύπαρξη ενός exploit είναι απειλή. Ακούσιες είναι οι απειλές που δεν έχουν προκληθεί σκόπιμα αλλά από λάθη στο σχεδιασμό ή τη χρήση συστημάτων ή ακόμα και από καταστροφές η αντιμετώπιση των οποίων δεν είχε προβλευθεί, όπως για παράδειγμα μια φυσική καταστροφή.

Παρακάτω περιγράφονται κάποιες από τις πιο γνωστές απειλές και αδυναμίες στην ασφάλεια των βάσεων δεδομένων με κυριότερη από αυτές να θεωρείται η **SQL Injection**.

3.2 Υπερβολικά και Αχρησιμοποίητα Προνόμια (Excessive and Unused Privileges)

Σε αυτή την περίπτωση χρήστες (ή εφαρμογές) έχουν προνόμια τα οποία υπερβαίνουν τις απαιτήσεις της εργασίας τους. Οι χρήστες μπορεί να καταχραστούν αυτά τα δικαιώματα και να εκτελέσουν λειτουργίες εκτός των αρμοδιοτήτων τους. Για παράδειγμα, ένας εργαζόμενος στη γραμματεία του Πανεπιστημίου που η δουλειά του απαιτεί μόνο δυνατότητα ανάγνωσης (read only) των βαθμολογιών των φοιτητών, μπορεί να εκμεταλλευτεί υπερβολικά δικαιώματα μετατροπής (update) με σκοπό να παραποιήσει τις βαθμολογίες κάποιων φοιτητών.

Το ερώτημα που γεννάται εδώ είναι πώς μπορεί ένας χρήστης να καταλήξει με Υπερβολικά προνόμια διαχείρισης για μια βάση δεδομένων. Αυτό συνήθως συμβαίνει γιατί οι μηχανισμοί ελέγχου προνομίων για τους διάφορους ρόλους δεν είναι σωστά ορισμένοι ή δεν διατηρούνται. Ως αποτέλεσμα μπορεί να χορηγούνται στους χρήστες γενικευμένα ή προεπιλεγμένα (default) προνόμια, τα οποία όμως να υπερβαίνουν κατά πολύ τις απαιτήσεις της εργασίας τους. Αυτή η αδυναμία όμως δημιουργεί περιττούς κινδύνους για το ΣΔΒΔ.

3.3 Κατάχρηση Προνομίων (Privilege Abuse)

Οι χρήστες μπορεί να προβούν σε κατάχρηση των δικαιωμάτων που τους παρέχονται από τη βάση για κακόβουλο ή μη εξουσιοδοτημένο σκόπο. Ας θεωρήσουμε για παράδειγμα μια εσωτερική εφαρμογή μιας κλινικής, η οποία χρησιμοποιείται για τον έλεγχο αρχείων μεμονομένων ασθενών μέσω μιας Web εφαρμογής. Οι χρήστες έχουν τη δυνατότητα να βλέπουν το ιστορικό της υγείας ενός μόνο ασθενούς, η πρόσβαση σε λίστες πολλών διαφορετικών ασθενών δεν είναι δυνατή και επίσης δεν επιτρέπεται η δημιουργία αντιγράφων των αρχείων. Ωστόσο ένας «αδίστακτος» χρήστης θα μπορούσε να παρακάμψει αυτούς τους περιορισμούς με το να συνδεθεί με τη βάση χρησιμοποιώντας έναν εναλλακτικό client όπως το MS-Excel. Χρησιμοποιώντας το Excel και τους κωδικούς πρόσβασης που του έχουν δοθεί για τη βάση θα μπορούσε εύκολα να αποθηκεύσει τα αρχεία όλων των ασθενών σε ένα laptop.

3.4 Κλιμάκωση προνομίων (Privilege escalation)

Η Κλιμάκωση προνομίων είναι μια διαδικασία που χρησιμοποιούν οι hackers για να διεισδύσουν σε ένα ΣΔΒΔ. Εκμεταλλευόμενοι ελλατώματα στο σχεδιασμό και τις διαδικασίες του λειτουργικού συστήματος ή μιας διαδικασίας ακόμα και SQL, διαλέγουν ένα λογαριασμό που δεν είναι πολύ γνωστός και αρχικά δεν έχει πολλά δικαιώματα και αρχίζουν να κλιμακώνουν (αυξάνουν) τα δικαιώματα αυτού του λογαριασμού ώστε να αποκτήσουν δικαιώματα διαχειριστή (administrator) στη βάση δεδομένων. Για την αποφυγή τέτοιου είδους επιθέσεων οι επιχειρήσεις χρησιμοποιούν ένα συνδυασμό συστημάτων πρόσληψης εισβολής (IPS - Intrusion Prevention System [4]) και ελέγχων πρόσβασης σε επίπεδο ερωτημάτων (query-level access control).

3.5 Malware

Το Malware είναι ένα είδος κακόβουλου λογισμικού το οποίο χρησιμοποιείται για να διακόψει τη λειτουργία του υπολογιστή να κλέψει ευαίσθητες πληροφορίες ή να διεισδύσει σε ιδιωτικά συστήματα υπολογιστών. Μπορεί να εμφανιστεί με τη μορφή κώδικα, script ή και ως άλλο λογισμικό. Οι διάφοροι hackers του διαδικτύου χρησιμοποιούν προηγμένες επιθέσεις που συνδυάζουν πολλαπλές τακτικές όπως share phishing emails και malware για να διεισδύουν σε οργανισμούς και να υποκλέπτουν ευαίσθητα δεδομένα. Αυτό γίνεται μέσω των εξουσιοδοτημένων χρηστών, οι οποίοι, αγνοώντας ότι η συσκευή τους έχει μολυνθεί από κακόβουλο λογισμικό, γίνονται αγωγοί ώστε οι hackers να έχουν πρόσβαση στο δίκτυό τους και στα ευαίσθητα δεδομένα των βάσεων τους.

3.6 Ασθενές Ίχνος Ελέχου (Weak Audit Trail)

Η αυτόματη καταγραφή κάθε συναλλαγής μιας βάσης με ευαίσθητα δεδομένα θα έπρεπε να είναι μέρος του συστήματος οποιασδήποτε βάσης δεδομένων. Η αποτυχία συλλογής λεπτομερών εγγραφών ελέγχου της δραστηριότητας της βάσης αποτελεί σοβαρό οργανωτικό κίνδυνο σε πολλά επίπεδα. Οργανισμοί με ασθενείς (ή και καθόλου) μηχανισμούς ελέγχου των βάσεων δεδομένων τους θα είναι σε αντίθεση με τις συνήθεις απαιτησεις της εποχής. Για αυτό το λόγο πολλές επιχειρήσεις στρέφονται σε φυσικά εργαλεία ελέγχου που τους παρέχονται από τους προμηθευτές των ΣΔΒΔ τους ή βασίζονται στις λύσεις που υπάρχουν σε διάφορα εγχειρίδια. Αυτές οι προσεγγίσεις δεν καταγράφουν τις λεπτομέριες που είναι απαραίτητες για να υποστηρίξουν ελέγχους και ανίχνευση επιθέσεων. Επίσης οι φυσικοί ελεγκτικοί μηχανισμοί είναι γνωστοί για την κατανάλωση πόρων της CPU και στου σκληρού δίσκου αναγκάζοντας έτσι πολλούς οργανισμούς να περιορίσουν ή και να εξαλείψουν τους ελέγχους. Τέλος, οι περισσότεροι φυσικοί ελεγκτικοί μηχανισμοί είναι μοναδικοί για κάθε πλατφόρμα βάσεων δεδομένων. Για παράδειγμα [16] τα logs της Oracle διαφέρουν από αυτά του MS – SQL και αυτά διαφέρουν από τα logs του DB2. Σε οργανισμούς με διαφορετικά περιβάλλοντα βάσεων δεδομένων για παράδειγμα αυτό είναι ένα σημαντικό εμπόδιο στην εφαρμογή μιας ενιαίας διαδικασίας ελέγχου.

Όταν οι χρήστες έχουν πρόσβαση στη βάση δεδομένων μέσω Web εφαρμογών (όπως SAP, Oracle κ.λ.π.) μπορεί να είναι δύσκολο να καταλάβουμε με ποιον χρήστη συνδέεται η δραστηριότητα πρόσβασης στη βάση. Οι περισσότεροι μηχανισμοί ελέγχου δεν έχουν καμία επίγνωση του ποιος είναι ο τελικός χρήστης επειδή όλη η δραστηριότητα είναι συνδεδεμένη με το όνομα του χρήστη για τη Web εφαρμογή. Έτσι, η όλη ανάλυση παρεμποδίζεται γιατί δεν υπάρχει καμιά σύνδεση με τον πραγματικό χρήστη.

Τέλος, οι χρήστες με πρόσβαση administrator είτε νόμιμα είτε κακόβουλα μπορούν να απενεργοποιήσουν τους φυσικούς ελέγχους της βάσης με σκοπό να αποκρύψουν «δόλιες» δραστηριότητες. Τα «καθήκοντα» ελέγχου ιδανικά θα έπρεπε να διαχορίζονται από τους διαχειριστές (administrators) και την πλατφόρμα της βάσης ώστε να είναι σαφής ο διαχωρισμός των αρμοδιοτήτων.

3.7 Έκθεση Μέσων Αποθήκευσης (Storage Media Exposure)

Τα μέσα αποθήκευσης αντιγράφων ασφαλείας είναι συνήθως εντελώς απροστάτευτα από επιθέσεις. Ως αποτέλεσμα πολλές παραβιάσεις ασφάλειας περιλαμβάνουν κλοπή cd ή παλαιότερα δισκετών με αντίγραφα δεδομένων (back up disks). Επιπλέον η παράλειψη ελέχγου και παρακολούθησης της δραστηριότητας των διαχειριστών που έχουν χαμηλού επιπέδου πρόσβαση σε ευαίσθητα δεδομένα μπορεί να βάλει τα δεδομένα αυτά σε κίνδυνο. Λαμβάνοντας κατάλληλα μέτρα ασφαλείας για τα αντίγραφα ευαίσθητων δεδομένων αλλά και παρακολουθώντας την δραστηριότητα των χρηστών με δικαιώματα πρόσβασης σε αυτά τα δεδομένα (υψηλού ή χαμηλού επιπέδου) είναι μια πολύ καλή πρακτική ασφάλειας βάσεων δεδομένων.

3.8 Εκμετάλλευση Ευάλλωτων Βάσεων Δεδομένων (Exploitation of Vulnerable)

Ένα συχνό φαινόμενο είναι οι ευάλωτες βάσεις δεδομένων που εξακολουθούν να έχουν προεπιλεγμένους λογαριασμούς και προεπιλεγμένη διαμόρφωση παραμέτρων, χωρίς να έχουν υποστεί κάποια επιδιόρθωση (patch) που θα βελτιστοποιούσε την ασφάλειά τους ενάντια σε πιθανές επιθέσεις. Οι πιθανοί εισβολείς γνωρίζουν πολύ καλά πως να εκμεταλλεύονται αυτές τις αδυναμίες για να εξαπολύουν επιθέσεις εναντίον οργανισμών/εταιριών. Δυστυχώς πολύ συχνά

οι οργανισμοί/εταιρίες επιλέγουν να διατηρήσουν τη διαμόρφωση της βάσης και να μην προβούν σε επιδιορθώσεις ακόμα και όταν αυτές είναι διαθέσιμες. Παρόλο που επιδιορθώσεις για βάσεις δεδομένων μπορεί να είναι διαθέσιμές για μεγάλο διάστημα παίρνει πολύ καιρό στις εταιρίες να προβούν στις απαραίτητες επιδιορθώσεις με αποτέλεσμα τα ευαίσθητα δεδομένα τους να είναι ευάλλωτα σε επιθέσεις.

3.9 Αδυναμίες πρωτοκόλου μιας Βάσης Δεδομένων (Database protocol vulnerabilities)

Οι αδυναμίες στα πρωτόκολα μιας βάσης δεδομένων μπορεί να οδηγήσουν σε μη εξουσιοδοτημένη πρόσβαση, διακοπή ή διαθεσιμότητα δεδομένων. Για παράδειγμα [16] ο ιός Slammer της SQL εκμεταλλεύτηκε μια αδυναμία του πρωτοκόλου του Microsoft SQL Server για να εκτελέσει έναν κώδικα επίθεσης με στόχο τους διακομιστές της βάσης δεδομένων.

Οι επιθέσεις πρωτοκόλλου μπορούν να αντιμετωπιστούν με την ανάλυση και επικύρωση των επικοινωνιών της SQL ώστε να επιβεβαιώνεται ότι αυτές είναι κατάλληλες.

3.10 Άρνηση Υπηρεσίας (Denial of service)

Η Άρνηση Υπηρεσίας (DoS [16]) είναι μια γενική κατήγορία επιθέσεων στην οποία η πρόσβαση σε εφαρμογές διαδικτύου ή δεδομένα αρνείται στους προβλεπόμενους χρήστες. Συνθήκες Άρνησης Υπηρεσίας μπορούν να δημιουργηθούν με πολλές τεχνικές. Η πιο συνηθισμένη σε περιβάλλοντα βάσεων δεδομένων είναι η υπερφόρτωση των πόρων του διακομιστή όπως η μνήνη και η CPU γεμίζοντας το δίκτυο με ερωτήματα για τη βάση τα οποία τελικά προκαλούν τη συντριβή του διακομιστή.

Τα κίνητρα πίσω από τις επιθέσεις DoS συχνά συνδέονται με «απάτες εκβιασμού» στις οποίες ένας απομακρυσμένος εισβολέας προκαλεί επανειλημμένα συντριβή στον server μεχρι το «θύμα» να ανταποκριθεί στις απαιτήσεις του. Όποια και αν είναι η πήγη, η Άρνηση Υπηρεσίας αποτελεί μια σοβαρή απειλή για τα ΣΔΒΔ.

3.11 SQL Injection

Η SQL Injection είναι ένας τύπος exploit όπου ο επιτιθέμενος προσθέτει SQL κώδικα στη φόρμα εισόδου χρηστών μιας εφαρμογής (login form) εισάγωντας μέσω αυτής **ερωτήματα SQL** στη βάση τα οποία αν εκτελεστούν θα του δώσουν πρόσβαση στα δεδομένα ή και τη δυνάτοτητα παραποίησης τους.

Ένα SQL ερώτημα (SQL query) είναι ένα αίτημα ή μια εντολή για μια ενέργεια που θέλουμε να εκτελεστεί στη βάση δεδομένων. Τυπικά σε μια φόρμα εισόδου του χρήστη σε μια εφαρμογή γίνεται πιστοποίηση του χρήστη αυτού (user authentication). Όταν λοιπόν οι χρήστες δίνουν το username και το password τους στα “text boxes” της φόρμας εισόδου συνήθως αυτά εισάγονται σε μια εντολή SELECT. Εάν οι τιμές που δόθηκαν από το χρήστη είναι οι αναμενόμενες από το σύστημα τότε επιτρέπεται η πρόσβαση στο χρήστη, ενώ εάν οι τιμές που δόθηκαν δεν πιστοποιούνται από τη βάση δεν επιτρέπεται η πρόσβασή σε αυτήν. Όμως οι περισσότερες φόρμες εισόδου δεν έχουν μηχανισμούς που να μην επιτρέπουν την εισαγωγή μέτα-χαρακτήρων στα “text boxes” με αποτέλεσμα οι επιτιθέμενοι να χρησιμοποιούν τις φόρμες εισόδου για να στείλουν τις εντολές που θέλουν στη βάση δεδομένων.

Τα αποτελέσματα από μια επίθεση στην SQL σε μια τρωτή περιοχή μπορούν να κυμανθούν από ένα λεπτομερές μήνυμα λάθους, το οποίο αποκαλύπτει την τεχνολογία που χρησιμοποιείται, ή επιτρέποντας στον επιτιθέμενο να έχει πρόσβαση στις μη προσβάσιμες περιοχές της βάσης με την παραποίηση του ερωτήματος σε μια πάντα-αληθή Boolean τιμή. Ακόμα μπορεί να του επιτρέψει την εκτέλεση εντολών διαχείρισης σε ένα σύστημα βάσεων δεδομένων.

Η απειλή των SQL Injections εμφανίστηκε λόγω της αύξησης των αυτοματοποιημένων εργαλείων. Στο παρελθόν ο κίνδυνος ήταν κάπως περιορισμένος γιατί η μέθοδος ενός exploit (και κατ' επέκταση μια SQL Injection) έπρεπε να δημιουγηθεί και να διεξαχθεί με το χέρι (manually), ένας hacker έπρεπε να εισάγει το SQL ερώτημα του στη φόρμα εισόδου. Τώρα όμως που υπάρχουν διαθέσιμα αυτοματοποιημένα προγράμματα για την εκτέλεση SQL Injections η πιθανότητα παραβίασης της ασφάλειας μιας βάσης με αυτή τη μέθοδο είναι εξαιρετικά μεγάλη. Με τα παραπάνω εργαλεία, που διατιθενται πολλές φορές ελεύθερα μέσω του διαδικτύου, δίνεται η δυνατότητα ακόμα και σε χρήστες με περιορισμένες γνώσεις να «κατεβάσουν» ένα τέτοιο εργαλείο να το εφαρμόσουν σε ένα Web Site και αυτόματα να κάνουν "download" ολόκληρη τη βάση δεδομένων που τρέχει από πίσω χωρίς καμία γνώση του θέματος.

Παρακάτω παρατίθενται κάποιες από τις πιο γνωστές τεχνικές για SQL Injections.

3.11.1 Τύποι SQL Injection

Οι συνέπειες των των επιθέσεων τύπου SQL Injection ποικίλουν από τη συλλογή ευαίσθητων δεδομένων μέχρι τον χειρισμό τους μέσα στη βάση και από την εκτέλεση εντολών στο σύστημα μέχρι και την άρνηση υπηρεσίας μιας εφαρμογής. Επίσης οι συνέπειες μιας τέτοιας επίθεσης εξαρτώνται από το είδος της βάσης, το στόχο του επιτιθέμενου καθώς και τους ρόλους και τα δικαιώματα της SQL στη βάση.

Οι ερευνητές χωρίζουν τις επιθέσεις αυτές σε τρεις κατηγορίες:

1. **Επιθέσεις πρώτης τάξης (First Order Attack)**

   Ο επιτιθέμενος μπορεί απλά να εισάγει εντολές SQL μέσω των οποίων ο τροποιποιημένος κώδικας που προκύπτει να εκτελεστεί άμεσα δίνοντας του πρόσβαση στη βάση.

2. **Επιθέσεις δεύτερης τάξης (Second Order Attack)**

   Σε αυτή την περίπτωση η οι τιμές που δίνονται περιέχουν μια SQL Injection που δεν εκτελείται άμεσα αλλά αποθηκεύεται σε μια αξιόπιστη δομή της βάσης όπως η γραμμή ενός πίνακα. Σε κάποιες περιπτώσεις εισάγεται ένα αίτημα το οποίο εγκαθιστά μια έγκυρη δήλωση SQL. Στη συνέχεια ένα άλλο μέρος αυτού του αιτήματος, χωρίς ελέγχους για SQL Injections εκτελεί αυτή την αποθηκευμένη εντολή SQL και προκαλεί την παραβίαση στη βάση.

3. **Πλευρικές επιθέσεις Lateral Injections**

   Σε αυτό τον τύπο επίθεσης ο επιτιθέμενος μπορεί να χειριστεί εσωτερικές συναρτήσεις της βάσης. Για παράδειγμα [15] χρησιμοποιώντας τη συνάρτηση "To_Char()" μπορεί να αλλάξει τις τιμές των περιβαλλοντικών μεταβλητών "NLS_Date_Format" ή "NLS_Numeric_Characters". Όταν το γράμμα "d" συνενώνεται με ένα string αυτόματα καλείται από την SQL η συνάρτηση To_Char(), η οποία το μετατρέπει σε format που καθορίζεται από την NLS_Date_Format, όποτε το συναντάει στο τέλος ενός string. Εδώ ο εισβολέας αλλάζει τις τιμές των περιβαλλοντικών μεταβλητών NLS_Date_Format σε ότι θέλει και αυτό βοηθά στην εκτέλεση της παράπλευρης επίθεσης αφού αντί για την αρχική μεταβλητή NLS_Date_Format τώρα εκτελείται αυτή που έχει οριστεί από τον εισβολέα.

3.11.2 Παραδείγματα SQL Injection

Υπάρχει μια σειρά από διαφορετικές τεχνικές που χρησιμοποιούνται στις επιθέσεις SQL. Κάθε τεχνική χαρακτηρίζεται από μια συγκεκριμένη υπογραφή. Σε αυτό το κεφάλαιο παρουσιάζονται τρεις τεχνικές SQL Injection.

1. **Εκμετάλλευση μη ύπαρξης φίλτρου χαρακτήρων διαφυγής(escape characters)**

   Αυτή η μορφή Injection είναι πιθανή όταν ο χρήστης δεν έχει προνοήσει να «φιλτράρει» τα δεδομένα που δίνοται σε μια φόρμα εισόδου και να αποκλείσει επικύνδυνους χαρακτήρες όπως οι χαρακτήρες διαφυγής.

   Ας θεωρήσουμε για παράδειγμα την παρακάτω δήλωση SQL:

   ```
   statement = "SELECT * FROM users WHERE name ='" + userName + "';"
   ```

   Αυτή η εντολή επιστρέφει τις εγγραφές του πίνακα “users” που έχουν το όνομα (πεδίο name) που δίνει ο χρήστης. Ωστόσο εάν η μεταβλητή που “userName” που θα δοθεί είναι κατασκευασμένοι από κακόβουλο χρήστη η παραπάνω SQL δήλωση μπορεί να κάνει περισσότερα από αυτά για τα οποία σχεδιάστηκε αρχικά. Για παράδειγμα αν η μεταβλητή που θα δοθεί είναι:

   ```
   ' or '1'='1
   ```

   η εντολή που θα τρέξει τελικά είναι:

   ```
   SELECT * FROM users WHERE name = '' OR '1'='1';
   ```

   Η παραπάνω εντολή είναι έγκυρη και θα επιστρέψει όλες τις γραμμές του πίνακα users, μιας και η δήλωση WHERE '1'='1' είναι πάντα “True”. Αν τώρα στις γραμμές του πίνακα users είναι αποθηκευμένα όλα τα usernames και passwords των χρηστών ένας εισβολέας μπορεί να αποκτήσει έυκολα πρόσβαση σε αυτά με δίνοντας απλά ‘ ‘ ή ‘1’ στη φόρμα εισόδου [www.w3schools.com].

2. **Εκμετάλλευση μη ύπαρξης ελέγχου του τύπου των τιμών εισόδου.**

   Εάν ο σχεδιαστής δεν έχει προνοήσει ώστε κατά την εισαγωγή τιμών σε μεταβλητές να γίνεται έλεγχος του τύπου των τιμών αυτών (αν δηλ. είναι numeric, character κ.λ.π.) μπορεί να επιτραπεί για παράδειγμα η εισαγωγή συμβόλων ή λέξεων σε ένα πεδίο που θα έπρεπε να δέχεται μόνο ακέραιους αριθμούς. Οι περισσότερες βάσεις δεδομένων υποστηρίζουν την δήλωση «πακέτων εντολών της SQL», δηλάδη τη δήλωση πολλών εντολών μαζί οι οποίες διαχωρίζονται συνήθως με ένα ελληνικό ερωτηματικό (;). Σε μια τέτοια περίπτωση η παραπάνω παράλειψη μπορεί να προκαλέσει σοβαρή ζημιά στη βάση. Αν για παράδειγμα έχουμε τη δήλωση:

   ```
   statement := "SELECT * FROM userinfo WHERE id =" + a_variable + ";"
   ```

   Ενώ είναι ξεκάθαρο ότι ο δημιουργός της εντολής θεώρησε ότι η μεταβλητή “id” είναι αριθμός. Αν όμως στην πραγματικότητα είναι ένα αλφαριθμητικό (string) τότε ο τελικός χρήστης μπορεί να χειριστεί αυτή την εντολή όπως θέλει παρακάμπτοντας την ανάγκη για τους χαρακτήρες διαφυγής. Αν για παράδειγμα ορίσει την μεταβλητή “a_variable” ως:

   ```
   1;DROP TABLE Suppliers
   ```

Η εντολή που θα εκτελεστεί είναι:

```
SELECT * FROM userinfo WHERE id=1;DROP TABLE users;
```

Και έτσι ένας μη εξουσιοδοτημένος χρήστης θα προκαλέσει τη διαγραφή ενός ολόκληρου πίνακα της βάσης δεδομένων [4].

Η μεγάλη σημασία που έχει για τις εταιρίες τους οργανισμούς άλλα και τους ιδιώτες η προστασία των κρίσιμων δεδομένων μιας βάσης οδήγησε στην ανάγκη εξεύρεσης τρόπων άμυνας απέναντι στις τεχνικές που απειλούν την ασφάλεια των δεδομένων αυτών. Η ανάγκη αυτή φυσικά οδήγησε τους προγραμματιστές να αναζητήσουν τους τρόπους για να προστατέψουν μια βάση δεδομένων όσο είναι δυνατόν από ενδεχόμενες επιθέσεις ή ευπάθειες.

3.12 Προσδιορισμός και Αξιολόγηση Αδυναμιών

Ένα από τα πρώτα πράγματα που θα πρέπει να κάνει ένας διαχειριστής βάσεων δεδομένων για την ασφάλεια της βάσης είναι να εντοπίσει που μπορεί να υπάρχουν αδυναμίες στη βάση και να τις αξιολογήσει. Θα πρέπει να εντοπίσει δηλαδή τις αδυναμίες που μπορούν να εκθέσουν τη βάση για παράδειγμα σε SQL Injections, τις πιθανές αδυναμίες ή παραλείψεις σε διαδικασίες πιστοποίησης ή τα τρωτά σημεία που μπορεί να υπάρχουν στις διαδικασίες που χρησιμοποιούνται στη βάση.

Αφού εντοπιστούν οι όποιες αδυναμίες είναι απαραίτητο να αξιολογηθούν και να ιεραρχηθούν με βάση την σημασία τους και την ευαισθησία των δεδομένων. Η αξιολόγηση των αδυναμιών μπορεί να βασιστεί σε γνωστά εργαλεία όπως το Common Vulnerability Scoring System (CVSS). Η αξιολόγηση βοηθάει στη σωστή ιεράρχηση του κινδύνου ώστε η διαχείριση των αδυναμιών που εντοπίστηκαν να οργανωθεί από σωστά ξεκινώντας από εκείνες με το μεγαλύτερο ρίσκο και αφήνοντας στο τέλος εκείνες με το μικρότερο. Μια τέτοια αξιολόγηση θα έδινε τη μεγαλύτερη επικυνδηνότητα σε αδυναμίες που μπορεί να αφήσουν απροστάτευτη τη βάση σε SQL Injections.

Από τη στιγμή που θα εντοπιστούν και θα ιεραρχηθούν οι αδυναμίες στο σύστημα βάσης δεδομένων θα πρέπει σύμφωνα και με την ιεραρχηση να ελεγχθεί εάν υπάρχει τρόπος να αντιμετωπιστούν, για παράδειγμα με ένα patch. Εάν δεν έχει δοθεί patch για το τρωτό σημείο που εντοπίστηκε θα πρέπει να γίνει χρήση εικονικών (virtual) patches. Η χρήση εικονικών patches θα προστατεύσει τη βάση από πιθανά exploits χωρίς να χρειαστεί να γίνουν αλλαγές στη δομή και τις λειτουργίες της. Και εδώ σημαντικό είναι να δοθεί προτεραιότητα σε αδυναμίες υψηλού κινδύνου που μπορεί να διευκολύνουν διαδικασίες DoS και SQL Injections.

3.13 Διαχείριση Δικαιωμάτων

Θα πρέπει να υπάρχει σωστή διαχείριση στα δικαιώματα που δίνονται στους χρήστες. Αρχικά θα πρέπει να συγκεντρωθούν πληροφορίες για τα δικαιώματα όλων των χρηστών. Ποιος χρήστης έχει ποια δικαιώματα, ποιος χορήγησε τα δικαιώματα και σε ποιους καθώς και σε ποια δεδομένα έχει δικαιώματα ο κάθε χρήστης είτε είναι υψηλού είτε χαμηλού επιπέδου. Η συγκέντρωση των πληροφοριών αυτών σε μια λίστα θα βοηθήσει της πρόσβασης των χρηστών σε ευαίσθητα δεδομένα.

Επίσης είναι θα πρέπει να εντοπιστούν και να αρθούν Υπερβολικά Δικαιώματα (Excessive Privileges) και να γίνει έλεγχος και αν κριθεί απαραίτητο διαγραφή των Αδρανών χρηστών.

Μπορεί λοιπόν αρχικά να γινεί μια έρευνα για να βρεθούν οι χρήστες στους οποίους έχουν χορηγηθεί δικαιώματα που δεν χρειάζονται για να κάνουν τη δουλειά τους όπως και οι χρήστες που για οποιονδήποτε λόγο δεν είναι πια ενεργοί. Αυτή η διαδικασία θα βοηθήσει να επιβεβαιωθεί ότι όλοι οι χρήστες έχουν μόνο τα δικαιώματα που χρειάζονται και όχι περισσότερα και ότι δεν υπάρχουν ανενεργοί λογαριασμοί με δικαιώματα σε ευαίσθητα δεδομένα. Οι hackers πολύ συχνά επιλέγουν να μιμηθούν χρήστες με δικαιώματα χαμηλού επιπέδου για να αποκτήσουν πρόσβαση στα ευαίσθητα δεδομένα της βάσης. Η μείωση των Υπερβολικών δικαιωμάτων λοιπόν μπορεί να βοηθήσει στην καλύτερη προστασία των ευαίσθητων δεδομένων.

3.14 Παρακολούθηση και Αποκλεισμός

Η παρακολούθηση της δραστηριότητας σε ένα σύστημα βάσης δεδομενων μπορεί να βοηθήσει στον εντοπισμό διαροής δεδομένων, μη εξουσιοδοτημένων SQL συναλλαγών και επιθέσεων του συστήματος σε πραγματικό χρόνο. Σε όλες αυτές τις περιπτώσεις μπορούν να uv real-time alerts ή να τερματίζεται άμεσα η σύνδεση του χρήστη. Επίσης η χρήση πολιτικών ασφάλειας που επιθεωρούν τη δραστηριότητα της βάσης και αναγνωρίζουν πρότυπα που αντιστοιχούν σε γνωστές επιθέσεις όπως η DoS μπορεί να βοηθήσει στην πρόληψη κάποιων επιθέσεων.

Διαδικασίες που μπορούν να εντοπίσουν Ασυνηθιστή Δραστηριότητα στη βάση δεδομένων μπορούν επίσης να φανούν χρήσιμες. Η καθιέρωση ενός σφαιρικού προφίλ για κάθε χρήστη της βάσης βοηθά να εντοπιστεί εύκολα μια ενέργεια που δεν ταιριάζει στο προφίλ ενός χρηστή. Η βάση μπορεί να δίνει μια ειδοποίηση (alert) σε μια τέτοια περίπτωση ή να μπλοκάρει προσωρινά το συγκεκριμμένο χρήστη μέχρι να επιβεβαιωθεί ότι η βάση είναι ασφαλής. Μια τέτοια διαδικασία παρέχει στη βάση δυνατότητα ανίχνευσης επιθέσεων DoS, SQL Injections, malware κ.λ.π.

Επισής επειδή οι web εφαρμογές είναι ο πιο κοινός φορέας SQL Injections η χρήση ενός Web Application Firewall (WAF) κρίνεται απαραίτητη κυρίως σε βάσεις που διαχειρίζονται μέσω διαδικτυακών εφαρμογών. Μια WAF θα αναγνωρίσει μια επίθεση SQL Injection.

3.15 Προστασία και Κρυπτογράφηση Δεδομένων

Τα ευαίσθητα δεδομένα της βάσης θα πρέπει να προστατεύονται με οποιονδήποτε τρόπο. Η αρχειοθέτηση των δεδομένων και εκτός της βάσης είναι μια διαδικασία που θα πρέπει να γίνεται περιοδικά για τα δεδομένα της βάσης. Η αποθήκευση των δεδομένων σε φυσικά μέσα αποθήκευσης (π.χ. εξωτερικούς σκληρούς δίσκους) μπορεί να σώσει σημαντικές πληροφορίες από κινδύνους που δεν εχουν ακόμα εμφανιστεί και πιθανώς δεν μπορούν να αντιμετωπιστούν.

Κρυπτογράφηση Δεδομένων (Data Encryption)

Κρυπτογράφηση είναι η διεργασία μετασχηματισμού ενός μυνήματος σε μια ακατανόητη μορφή με τη χρήση ενός κρυπτογραφικού αλγόριθμού, έτσι ώστε αυτό να μην είναι αναγνώσιμο από τρίτα μέρη (εκτός νόμιμου παραλήπτη) [13]. Η αντίστροφη διαδικασία, δηλαδή η αποκρυπτογράφηση (decryption) είναι η διεργασία ανάκτησης του αρχικού μηνύματος (αναγνώσιμη μορφή) από μια ακατανόητη έκδοση του που είχε παραχθεί μέσω μιας διεργασίας κρυπτογράφησης και απαιτεί τη γνώση του κατάλληλου κλειδιού. Η αποκρυπτογράφηση εκτελείται από κάποιο εξουσιοδοτημένο μέρος, σε αντίθεση με την κρυπτανάλυση.

Η κρυπτογράφηση είναι μια τεχνολογία που χρησιμοποιείται για την προστασία των δεδομένων μια βάσης. Η κρυπτογράφηση μπορεί να εφαρμοστεί στα περιεχόμενα της βάσης μέσω εσωτερικών συναρτήσεων της βάσης ή εξωτερικά χρησιμοποιώντας μια άλλη εφαρμογή. Η κρυπτογράφησης μιας βάσης δεδομένων μπορεί ταξινομηθεί σε δύο βασικούς τύπους:

Διαφανής/Εξωτερική Κρυπτογράφηση (Transparent/External Encryption): όρος για την κρυπτογράφηση όλοκληρης της βάσης δεδομένων. Αυτό παρέχεται από τις συναρτήσεις κρυπτογράφησης του συστήματος βάσης δεδομένων. Ορισμένοι προμηθευτές βάσεων προσφέρουν κρυπτογράφηση σε επίπεδο στηλών και πινάκων αλλά το πιο σύνηθες είναι η κρυπτογράφηση του συνόλου των δεδομένων της βάσης. Ονομάζεται «Διαφανής» Κρυπτογράφηση Δεδομένων γιατί είναι αόρατη στους χρήστες και τις εφαρμογές που χρησιμοποιούν τα δεδομένα και δεν απαιτεί αλλαγές στη δομή της βάσης. Η αρχική κύρια χρήση της είναι για την αποφυγή έκθεσης ευαίσθητων δεδομένων στην περίπτωση απώλειας φυσικών μέσων αποθήκευσης. Η Διαφανής κρυπτογράφηση προστατεύει τη βάση από μη εξουσιοδοτημένους χρήστες αλλά δεν προστατεύει τα δεδομένα από εξουσιοδοτημένους χρήστες.. Η κλασική περίπτωση χρήσης για αυτό το μοντέλο κρυπτογράφησης είναι η κρυπτογράφηση αριθμών πιστωτικών καρτών σε μια βάση. Ο στόχος είναι να παρέχεται προστασία έναντι ακούσιας έκθεσης δεδομένων ή και να επιβληθεί διαχωρισμός ρόλων σε διαπιστευμένους χρήστες της βάσης. Το μειονέκτημα είναι ότι αυτές οι παραλαγές δεν είναι αόρατες στη βάση και συνήθως απαιτούν αλλαγές στον κώδικα της βάσης. Η κεντρική ιδέα είναι να κρυπτογραφούνται με αυτή τη μέθοδο μόνο τα δεδομένα που ορίζονται ώς ευαίσθητα από το διαχειριστή της βάσης μειώνοντας τις συνέπειες και τις αλλαγές στον κώδικα της βάσης. Το πώς αυτό μπορεί να επιτευγχθεί εξαρτάται από τον τρόπο διαχεισης κλειδιών, τη χρήση εσωτερικής ή εξωτερικής κρυπτογράφησης και το πώς οι εφαρμογές χρησιμοποιούν τη βάση.

Κρυπτογράφηση Χρηστών/Δεδομένων(User/Data Encryption): όρος που περιγράφει την κρυπτογράφηση συγκεκριμένων στηλών, πινάκων ή ακόμα και δεδομένων μέσα στη βάση. Λέγεται και Κρυπτογράφηση Χρηστών γιατί τα κρυπτογραφημένα δεδομένα ανήκουν και διαχειρίζονται ανά χρήστη. Η κλασική περίπτωση χρήσης για αυτό το μοντέλο κρυπτογράφησης είναι η κρυπτογράφηση αριθμών πιστωτικών καρτών σε μια βάση. Ο στόχος είναι να παρέχεται προστασία έναντι ακούσιας έκθεσης δεδομένων ή και να επιβληθεί διαχωρισμός ρόλων σε διαπιστευμένους χρήστες της βάσης. Το μειονέκτημα είναι ότι αυτές οι παραλαγές δεν είναι αόρατες στη βάση και συνήθως απαιτούν αλλαγές στον κώδικα της βάσης. Η κεντρική ιδέα είναι να κρυπτογραφούνται με αυτή τη μέθοδο μόνο τα δεδομένα που ορίζονται ώς ευαίσθητα από το διαχειριστή της βάσης μειώνοντας τις συνέπειες και τις αλλαγές στον κώδικα της βάσης. Το πώς αυτό μπορεί να επιτευγχθεί εξαρτάται από τον τρόπο διαχεισης κλειδιών, τη χρήση εσωτερικής ή εξωτερικής κρυπτογράφησης και το πώς οι εφαρμογές χρησιμοποιούν τη βάση.

Συναρτήσεις κατακερματισμού (hash functions H) είναι συναρτήσεις που παίρνουν ένα όρισμα μεταβλητού μήκους (m) και επιστρέφουν ένα αλφαριθμητικό σταθερού μήκους (h), h=H(m). Με τη χρήση τέτοιων συναρτήσεων μπορούν να κρυπτογραφηθούν κωδικοί χρηστών μιας βάσης δεδομένων ώστε να μην είναι δυνατή η αποκάλυψη τους σε πιθανές επιθέσεις. Κάποιες από τις πιο γνωστές συναρτήσεις κατακερματισμού είναι οι MD5 (16), SHA (16) και SHA-1 (16).

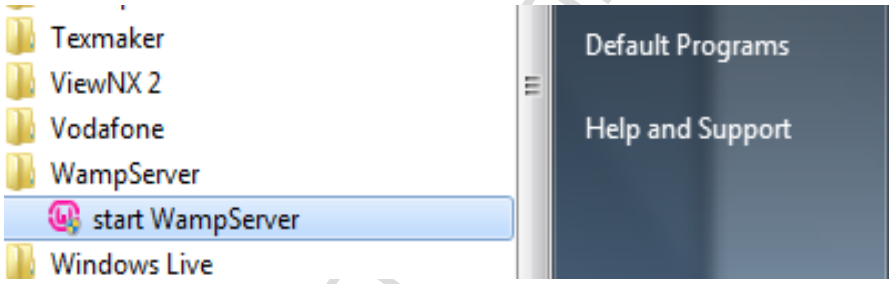
Κεφάλαιο 4°

4 Αναλυτική περιγραφή Εργαλείων - Θέματα Ασφάλειας

Όπως αναφέρεται και στο πρώτο κεφάλαιο για τη εκπόνηση της πρακτικής εφαρμογής χρησιμοποιήθηκε το πακέτο WAMP. Στο κεφάλαιο αυτό περιγράφεται πιο αναλυτικά η χρήση των εργαλείων του WAMP στην εφαρμογή και αναφέρονται κάποιες από τις ευπάθειες που έχουν εντοπιστεί από προγραμματιστές για το πακέτο και τα εργαλεία που αυτό παρέχει.

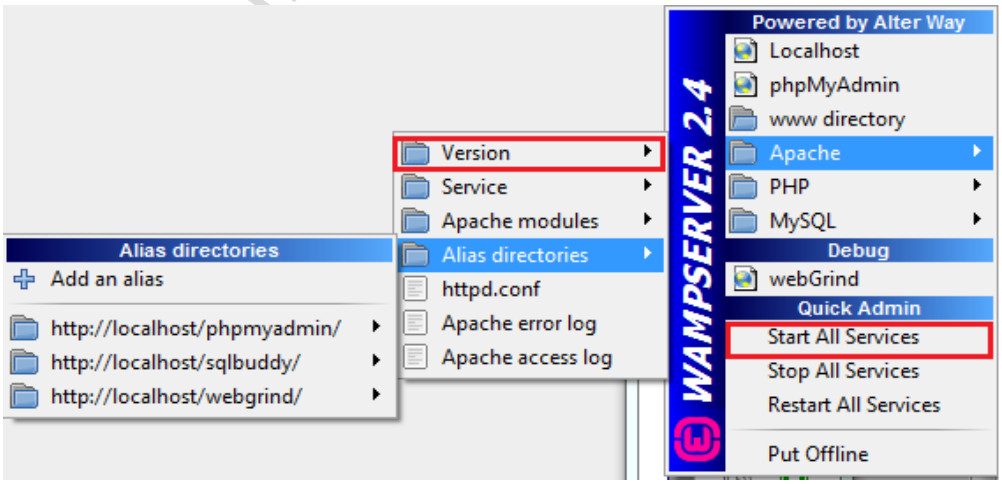
4.1 Περιγραφή του πακέτου WAMP που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής

Αφού το WAMP εγκατασταθεί στον υπολογιστή εύκολα, σαν ένα απλό πρόγραμμα, από την επιλογή “All Programs” των Windows μπορεί κάποιος να ανοίξει το παράθυρο διαλόγου του (Σχήμα 2.) και να επιλέξει “Start All Services” για να ενεργοποιήσει όλες τις εφαρμογές και τα εργαλεία του server (Σχήμα 3.).



Εικόνα 1. Εμφάνιση του WAMP Server από τα Windows

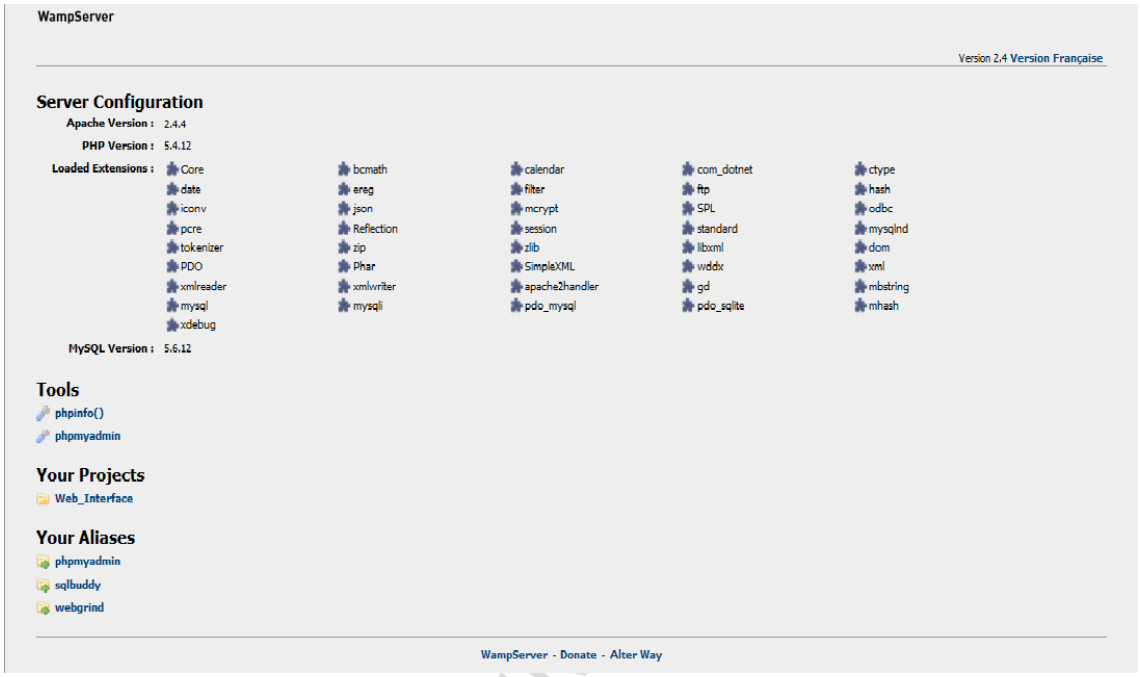
Στο παράθυρο διαλόγου του WAMP επίσης ο χρήστης μπορεί να δει εν συντομία τις εφαρμογές και τα εργαλεία του, τις εκδόσεις και τις δομές των εφαρμογών αυτών να ανοίξει την κεντρική τοπίκη σελίδα (localhost) (Σχήμα 4) με όλες τις πληροφορίες του του πακέτου και ανοίξει όποιο εργαλείο χρειάζεται.



Εικόνα 2. Παράθυρο Διαλόγου του WAMP Server

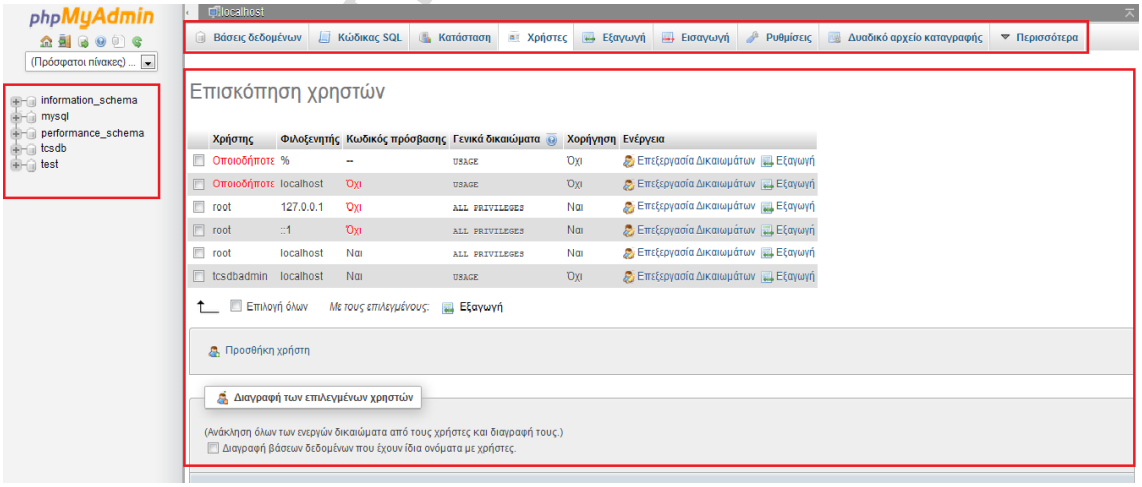
Σχήμα 3. Παράθυρο Διαλόγου του WAMP.

Επιλέγοντας είτε από το παράθυρο διαλόγου είτε από την κεντρική σελίδα διαχείρισης του WAMP το εργαλείο “phpMyAdmin” ο χρήστης μπορεί να ξεκινήσει την δημιουργία και διαμόρφωση μιας βάσης δεδομένων. Στη σελίδα του phpMyAdmin ο χρήστης έχει μεγάλη ποικιλία επιλογών. Μπορεί να δημιουργήσει μια εντελώς νέα βάση από την αρχή ή να επιλέξει



Εικόνα 3. Κεντρική σελίδα του WAMP.

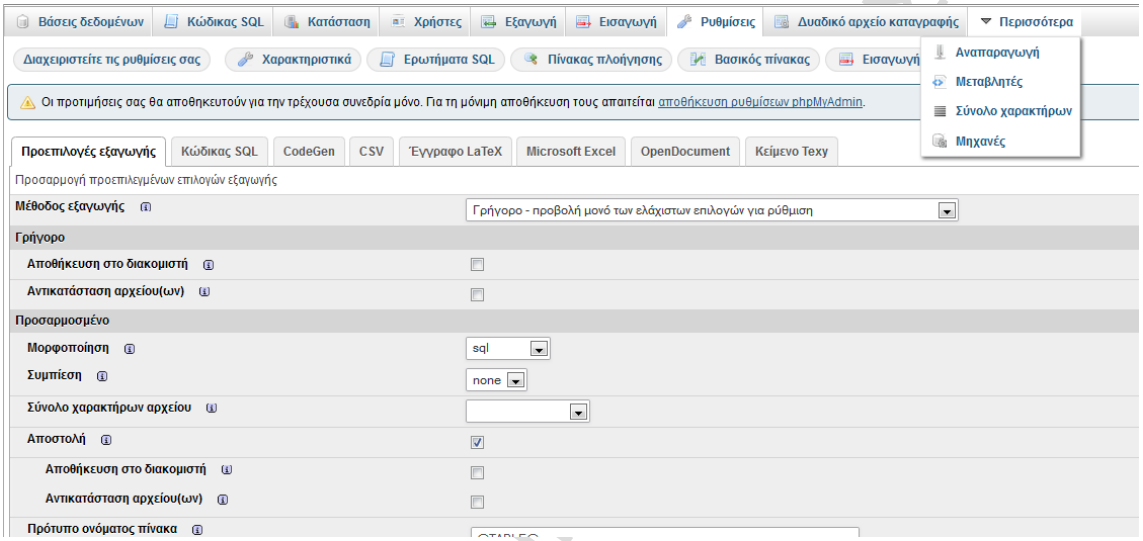
να εισάγει μια βάση που ήδη έχει δημιουργηθεί (εάν φυσικά έχει σώσει την δομή της με το σωστό τρόπο). Από αυτή την οθόνη ο χρήστης βλέπει όλες τις λειτουργίες που μπορεί να τρέξει μέσω του εργαλείου και τις βάσεις που ήδη υπάρχουν στον server.



Εικόνα 4. phpMyAdmin Tool

Οι επιλογές που δίνει το εργαλείο είναι πάρα πολλές. Κάποιες από τις κυριότερες αναφέρονται εδώ. Ο χρήστης μπορεί :

- Να δει όλες τις βάσεις που είναι εγκατεστημένες στον server.
- Να δώσει εντολές γράφοντας κώδικα SQL για τη δημιουργία νέων βάσεων ή τη διαχείριση αυτών που ήδη υπάρχουν στον server.
- Να δει τους χρήστες τα δικαιώματα που έχουν και να τα επξεργαστεί.
- Να εξάγει σε μορφή sql (π.χ. tcsdb.sql) μια ολόκληρη βάση δεδομένων και αντίστοιχα να εισάγει μια βάση δεδομένων στον server από ένα αρχείο sql.
- Να αλλάξει τις ρυθμίσεις του εργαλείου όπως τον εξυπηρετούν όπως για παράδειγμα το πώς θα εξάγεται μια βάση δεδομένων όταν επιλέγει κάποιος να εκτελέσει μια Εξαγωγή (Σχήμα 6)



Εικόνα 5. Αλλαγή Ρυθμίσεων για την εκτέλεση Εξαγωγής μιας βάσης δεδομένων στο phpMyAdmin

Το εργαλείο αυτό χρησιμοποιήθηκε για τη δημιουργία της βάσης και τη σύνδεσή της με την Web εφαρμογή.

Για την μορφή της εφαρμογής (τύπου CRM) που κατασκευάστηκε για την διαχείριση των δεδομένων της χρησιμοποιήθηκαν οι γλώσσες HTML και Jaasript.

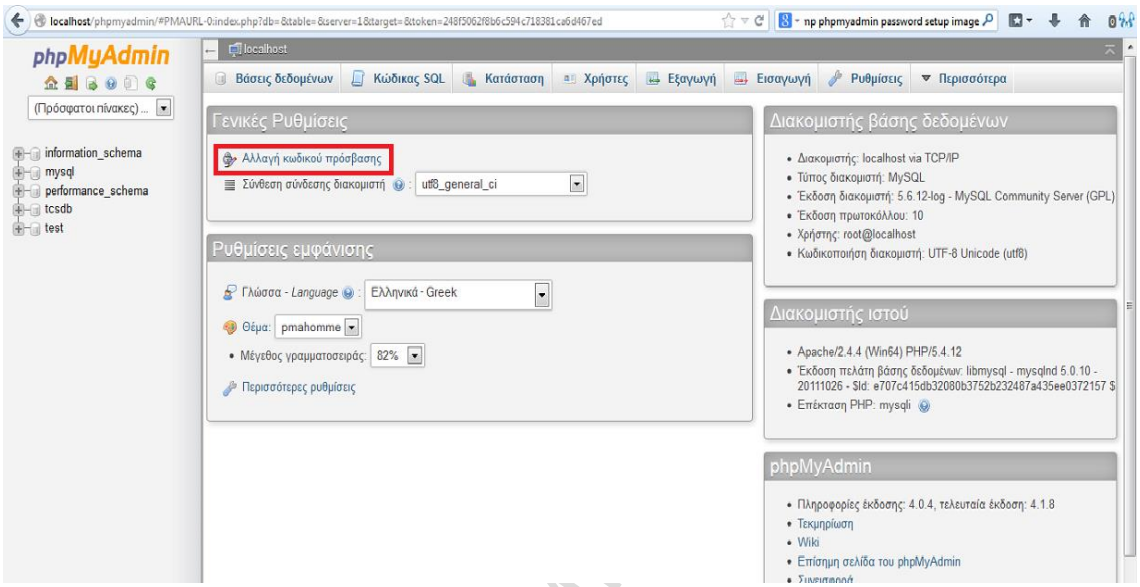
Στην επόμενη ενότητα παρουσιάζονται κάποια θέματα ασφάλειας για το πακέτο WAMP.

4.2 Θέματα ασφάλειας στις εφαρμογές του πακέτου WAMP

4.2.1 Εργαλείο PhpMyAdmin

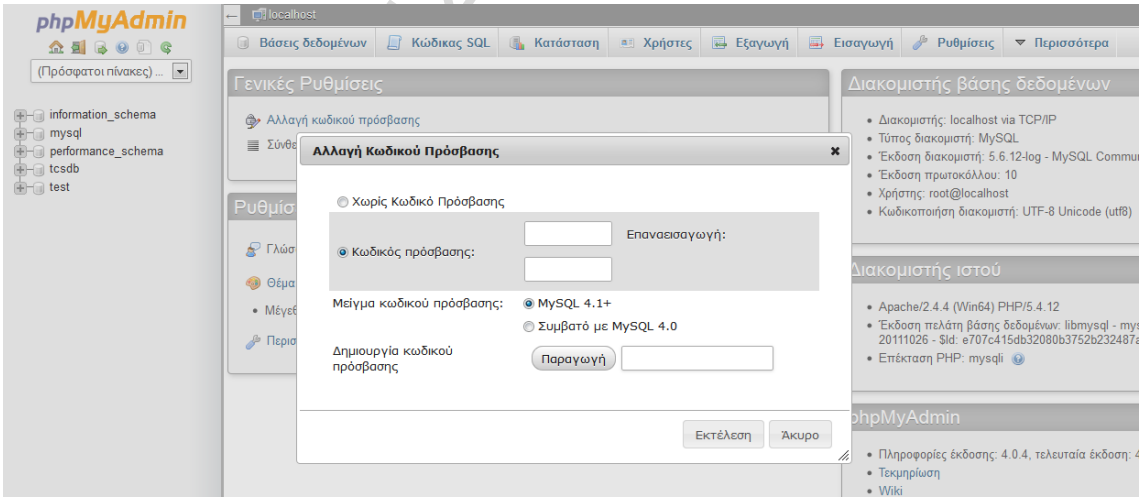
΄Εχει παρατηρηθεί από τους προγραματιστές ότι συνήθως όταν κάποιος εγκαθιστά το WAMP δεν κάνει σωστά αυτή την εγκατάσταση. Συνήθως ο χρήστης απλά εγκαθιστά το πακέτο με τις default προδιαγραφές χωρίς να δίνει php/mysql administrator κωδικό. Αυτή η διαδικασία δίνει τη δυνατότητα σε οποιοδήποτε χρήστη να χρησιμοποιήσει την κονσόλα **PhpMyadmin** και να εκτελέσει λειτουργίες όπως create / update / delete στον mysql server. Αυτή η παράλειψη μπορεί να επιτρέψει σε κάποιον επιτιθέμενο με τις κατάλληλες εντολές να αποκτήσει πλήρη έλεγχο στον web server και να και να καταστρέψει ένα ολόκληρο σύστημα βάσεων δεδομένων. Παρόλο που ο κίνδυνος είναι πολύ σοβαρός ο αριθμός των χρηστών που απλά κατεβάζουν εγκαθιστούν και χρησιμοποιούν το WAMP χωρίς να δημιούργησουν τους απαραίτητους

κωδικούς για τις php/mysql είναι μεγάλος. Για να είναι ασφαλείς οι βάσεις που πρόκειται να διαχειριστεί κάποιος μέσα από τον server θα πρέπει πρώτα απ' όλα πριν ξεκινήσει να χρησιμοποιεί οποιδήποτε από τα εργαλεία του να δημιουργήσει κωδικό πρόσβασης. Το εργαλείο PhpMyAdmin δεν πρόκειται ποτέ να ζητήσει password για να επιτρέψει στον χρήστη την πρόσβαση στις εφαρμογές, είναι κάτι που θα πρέπει ο χρήστης να κάνει μόνος του. Για να δώσει νέο κωδικό πρόσβασης ο χρήστης μπορεί από την κεντρική σελίδα του PhpMyAdmin (Γενικές Ρυθμίσεις) να επιλέξει «Αλλαγή Κωδικού πρόσβασης» :



Εικόνα 6. Αλλαγή Κωδικού πρόσβασης στο phpMyAdmin 1ο βήμα

Και κατόπιν στην οθόνη που θα εμφανιστεί να δώσει τον κωδικό πρόσβασης που επιθυμεί:



Εικόνα 6. Αλλαγή Κωδικού πρόσβασης στο phpMyAdmin 2ο βήμα

4.2.2 APACHE

Η ασφάλεια του server ο οποίος χρησιμοποιείται για την διαχείριση των δεδομένων της βάσης με μιας web εφαρμογης είναι πολύ σημαντική. Ο Apache HTTP server σε θέματα ασφάλειας έχει καλό ιστορικό και μια ολόκληρη κοινωνία προγραμματιστών που ασχολείται συνεχώς με θέματα ασφάλειας που μπορεί να προκύψουν και την άμεση αντιμετώπισή τους. Παρ' όλα αυτά είναι αναπόφευκτο ότι για κάθε έκδοση του λογισμικού που εκδίδεται θα εμφανιστούν κάποια προβλήματα κατά τη χρήση του.

Για παράδειγμα στις 23 Μαΐου 2013 δημοσιεύτηκε από την ομάδα προγραμματιστών του Apache μια ευπάθεια στην ενότητα "mod_dav ", με τίτλο "mod_dav Crash", για τις εκδόσεις 2.4.4, 2.4.3, 2.4.2, 2.4.1 του Apache Server. Σε αυτές τις εκδόσεις του server η συνάρτηση δεν καθορίζει σωστά εάν το πρωτόκολο DAV (Distributed Authoring and Versioning) είναι ενεργοποιημένο για ένα URI (Uniform Resource Identifier) το οποίο επιτρέπει σε απομακρυσμένους επιτιθέμενους να προκαλέσουν "Άρνηση Υπηρεσίας" (segmentation fault) μέσω ενός συγχωνευμένου αιτήματος στο οποίο το URI έχει ρυθμιστεί ώστε να διαχειρίζεται από την συνάρτηση "mod_dav_svn" αλλά ένα συγκεκριμένο href χαρακτηριστικό στην XML (Extensible Markup Language) αναφέρεται σε ένα non-DAV URI. Αυτή η ευπάθεια διορθώνεται στην έκδοση 2.4.6 του Apache.

Άλλη μια ευπάθεια ανακοινώθηκε από την ίδια ομάδα στις 22 Ιουλίου της ίδιας χρονιάς αυτή τη φορά για την "mod_session_dbd". Ένα ελάτωμα την «ανάγκασε» να προχωρήσει στην αποθήκευση κάποιων λειτουργιών για μια ενότητα χωρίς να λαμβάνει υπόψην τα "dirty flag" (ειδοποιήσεις για αλλαγές) και την απαίτηση για νέο κωδικό ενότητας (Session ID). Και αυτό το πρόβλημα έχει επιλυθεί στον Apache 2.4.6. [http://httpd.apache.org - Apache httpd 2.4 vulnerabilities]

Ενδεχομένως η χρήση μιας ενημερωμένης έκδοσης του WAMP με τη νέα έκδοση του Apache (2.4.6) θα ήταν ένα τρόπος να αποφύγει κάποιος πιθανές επιθέσεις όπως αυτές που περιγράφονται παραπάνω.

4.2.3 PHP

Η php είναι μια γλώσσα προγραμματισμού και από μόνη της δεν μπορεί να προστατεύσει μια βάση δεδομένων. Αυτό που μπορεί να γίνει με τη χρήση της php είναι να δημιουργηθούν αλγόριθμοι σε php που θα ελέγχουν την πρόσβαση και τη διαχείριση των δεδομένων της βάσης μέσω μιας web εφαρμογής.

Μπορούν να καθοριστούν οι συνδέσεις μέσω SSL (Secure Socket Layer) για την κρυπτογράφηση των επικοινωνιών client/server για αυξημένη ασφάλεια της βάσης ή να χρησιμοποιηθούν SSH (Secure Shell) για την κρυπτογράφηση της σύνδεσης των χρηστών με τη βάση δεδομένων. Εάν έστω και ένα από τα παραπάνω εφαρμοστεί η επικοινωνία των χρηστών με τη βάση είναι κρυπτογραφημένη και θα είναι δύσκολο για πιθανούς εισβολείς να παρακολουθήσουν αυτές τις επικοινωνίες και να πάρουν πληροφορίες για τη βάση.

Στην περίπτωση που ένας εισβολέας καταφέρει να διαπεράσει το server και να αποκτήσει πρόσβαση στη βάση είναι επικίνδυνο (όπως έχει ήδη αναφερθεί) να υποκλέψει ή να αλλοιώσει ευαίσθητα δεδομένα της. Ενάς τρόπος να αντιμετωπιστεί αυτή η απειλή είναι να κρυπτογραφηθούν τα δεδομένα της βάσης (κεφ.3.14), όμως πολύ λίγες βάσεις δεδομένων προσφέρουν αυτό τον τύπο κρυπτογράφησης. Μια εύκολη λύση για αυτό το πρόβλημα είναι να δημιουργηθεί ένα πακέτο κρυπτογράφησης το οποίο μπορεί να χρησιμοποιηθεί μέσω της php. Η php προσφέρει αρκετές συναρτήσεις, όπως αυτές στις βιβλιοθήκες **Mcrypt** και **Mhash** οι οποίες καλύπτουν μια ευρεία ποικιλία αλγορίθμων κρυπτογράφησης. Τα δεδομένα κρυπτογραφούνται πριν την εισαγωγή τους στη βάση δεδομένων και αποκρυπτογραφούνται κατά την ανάκτηση τους. Μερικές από τις συναρτήσεις κρυπτογράφησης της php είναι οι εξής: mhash, mhash_count, mhash_keygen, mcrypt_cbc, mcrypt_cfb κ.λ.π.

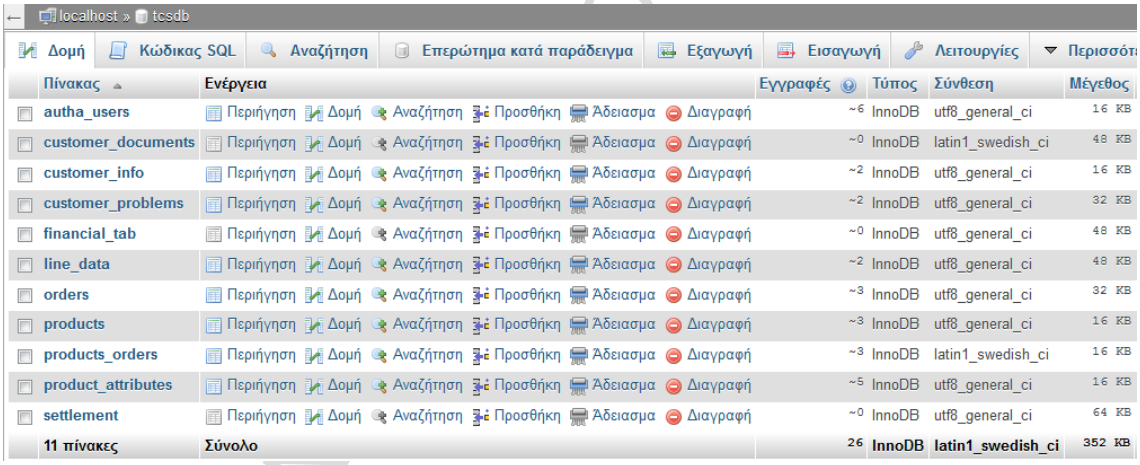
Κεφάλαιο 5

5 Μελέτη περίπτωσης: Παράδειγμα εφαρμογής για εταιρεία τηλεπικοινωνιών

5.1 Η Βάση Δεδομένων

Σε αυτό το κεφάλαιο παρουσιάζεται η υλοποίηση μιας βάσης δεδομένων τα στοιχεία της οποίας διαχειρίζονται μέσω μιας Web εφαρμογής. Η Βάση Δεδομένων διαχειρίζεται τα δεδομένα των πελάτων μιας εταιρίας τηλεπικοινωνιών και συνδέεται με μια εφορμογή CRM (Customer Relationship Management) που έχει ονομαστεί “Telecomunication Company”.

Η ιδέα ήταν η δημιουργία ενός συστήματος μέσα από το οποίο διάφοροι χρήστες με διαφορέτικα δικαιώματα μπορούν να διαχειρίζονται και να έχουν πρόσβαση σε συγκεκριμένα δεδομένα της βάσης. Η δημιουργία της βάσης έγινε με το εργαλείο phpMyAdmin. Πριν δημιουργηθεί η βάση (σύμφωνα και με το κεφάλαιο 4.1.1) δημιουργήθηκε ένας μοναδικός κωδικός για τη MySQL ο οποίος έχει πρόσβαση σε οποιαδήποτε βάση δεδομένων υπάρχει στον server. Κατόπιν δημιουργήθηκε η βάσηδεδομένων “tcsdb” της οποίας η δομή φαίνεται στο παρακάτω σχήμα (Σχήμα 9):



Εικόνα 7. Δομή της βάσης tcsdb

Η βάση αποτελείται από 11 πίνακες.

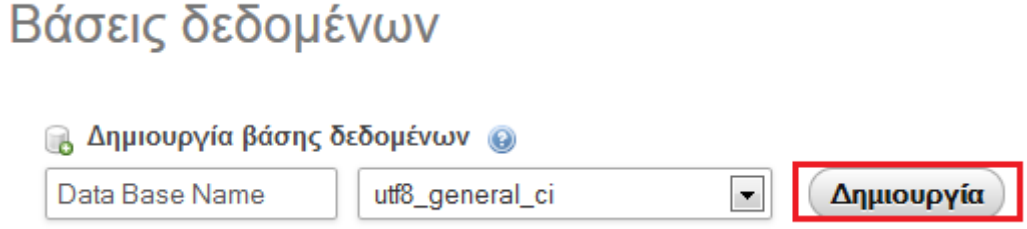
1. **autha_users :** σε αυτό τον πίνακα αποθηκεύονται τα usernames και passwords των χρηστών που θα έχουν πρόσβαση στη βάση. (πεδία: username, password)
2. **customer_documents:** σε αυτό τον πίνακα αποθηκεύονται τα έγγραφα που έχει αποστείλει ο πελάτης στην εταιρία. (πεδία: document_id, document_type, document_title, attachment_date, ξένα κλειδιά: customer_id, order_id)
3. **customer_info:** σε αυτό τον πίνακα αποθηκεύονται όλα τα προσωπικά δεδομένα του πελάτη (πεδία: customer_id, customer_first_name, customer_last_name, customer_identity_number, customer_tax_number, customer_birth_date, customer_job, customer_contact_number, customer_mobile_number, customer_2nd_contact_number,

customer_email, customer_type, customer_street, customer_street_number, customer_city, customer_postcode)

4. **customer_problems:** σε αυτό τον πίνακα αποθηκεύονται όλα τα προβλήματα ή και τα αιτήματα που μπορεί να καταγραφούν για τον κάθε πελάτη (τεχνικά, λογιστικά κ.α.) (πεδία customer_problem_id, customer_problem_creation_date, customer_problem_resolution_date, customer_problem_status, customer_problem_category, customer_problem_subcategory, customer_problem_subject, customer_problem_comments, ξένα κλειδιά: customer_id, order_id)
5. **financial_tab:** αυτός ο πίνακας είναι η οικονομική καρτέλα του πελάτη. Εδώ φαίνονται οι οφειλές, οι πληρωμές ή και οι πιστώσεις του. (πεδία: bill_id, type_of_record, date_of_issue, expiration_date, invoice_file, current_ballance, payment_credits, ξένα κλειδιά: customer_id, order_id)
6. **line_data:** σε αυτόν τον πίνακα αποθηκεύονται τα στοιχεία της τηλεφωνικής σύνδεσης του πελάτη. (πεδία: line_number, sip_number, IP_address, urban_center, internet_username, internet_password, router_serial_number, router_mac_address ξένα κλειδιά: order_id, customer_id)
7. **orders:** εδώ αποθηκεύονται όλες οι παραγγελίες που είναι στο ονόμα του πελάτη. (πεδία: order_id, order_type, order_creation_date, order_activation_date, order_deactivation_date, ξένο κλειδί: customer_id)
8. **products:** εδώ αποθηκεύονται τα προϊόντα που προσφέρονται στον πελάτη. (product_name, product_price)
9. **product_attributes:** στον πίνακα αυτό αποθηκεύονται τα διάφορα χαρακτηριστικά των προϊόντων. (πεδία: product_name, attribute_name)
10. **products_orders:** εδώ αποθηκεύονται οι παραγγελίες του πελάτη με τα προϊόντα που έχει επιλέξει να προσθέσει ο πελάτης στην κάθε παραγγελία. (πεδία: product_name, order_id)
11. **settlement:** εδώ αποθηκεύονται οι διακανονισμοί που μπορεί να έχει ο πελάτης. (πεδία: settlement_id, settlement_creation_date, settlement_due_date, settlement_payment_day, settlement_ammount, settlement_status, customer_id, order_id, bill_id)

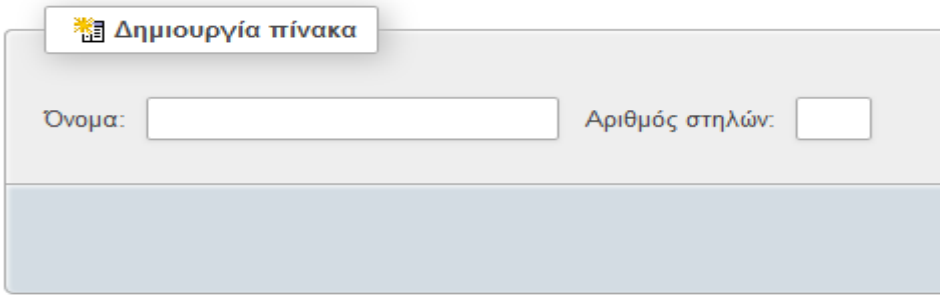
Για τη δημιουργία της βάσης στο phpMyAdmin ακολουθούμε τα παρακάτω βήματα:

Επιλέγουμε “Βάσεις Δεδομένων” και στην οθόνη που εμφανίζεται δίνουμε το όνομα και τη σύνθεση της βάσης. Στην υλοποίηση έχει χρησιμοποιήθεί σύνθεση utf8_general_ci ώστε στα πεδία της βάσης να μπορούν να αποθηκεύονται και ελληνικά.



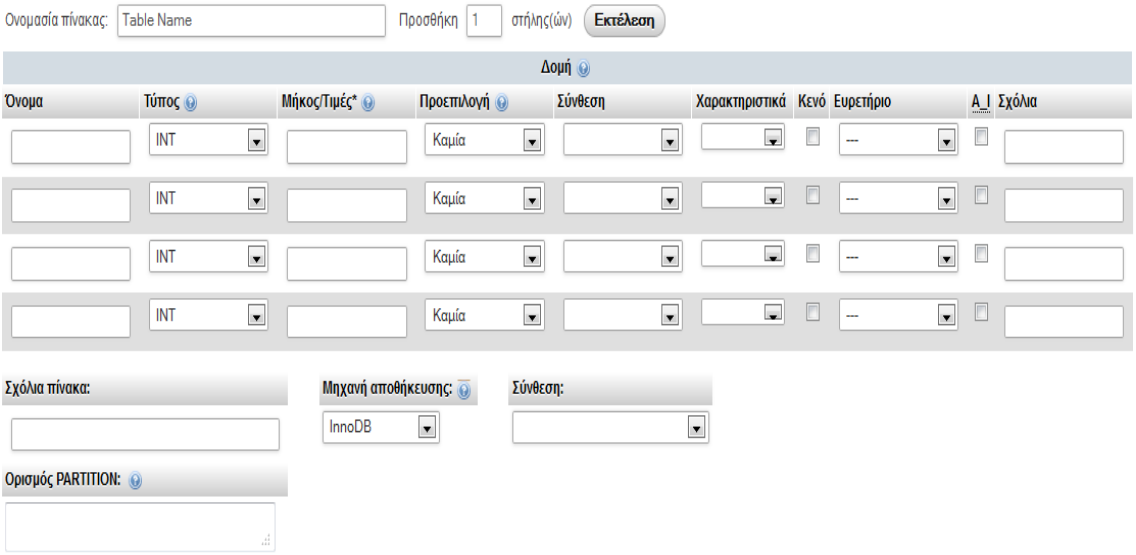
Εικόνα 8. Δημιουργία Βάσης Δεδομένων στο phpMyAdmin

Κατόπιν στην ενότητα “Δημιουργία Πίνακα” δίνουμε το όνομα και τον αριθμό των στηλών του πίνακα και στην οθόνη που εμφανίζεται συμπληρώνουμε αναλυτικά τα στοιχεία των πεδίων του πίνακα.



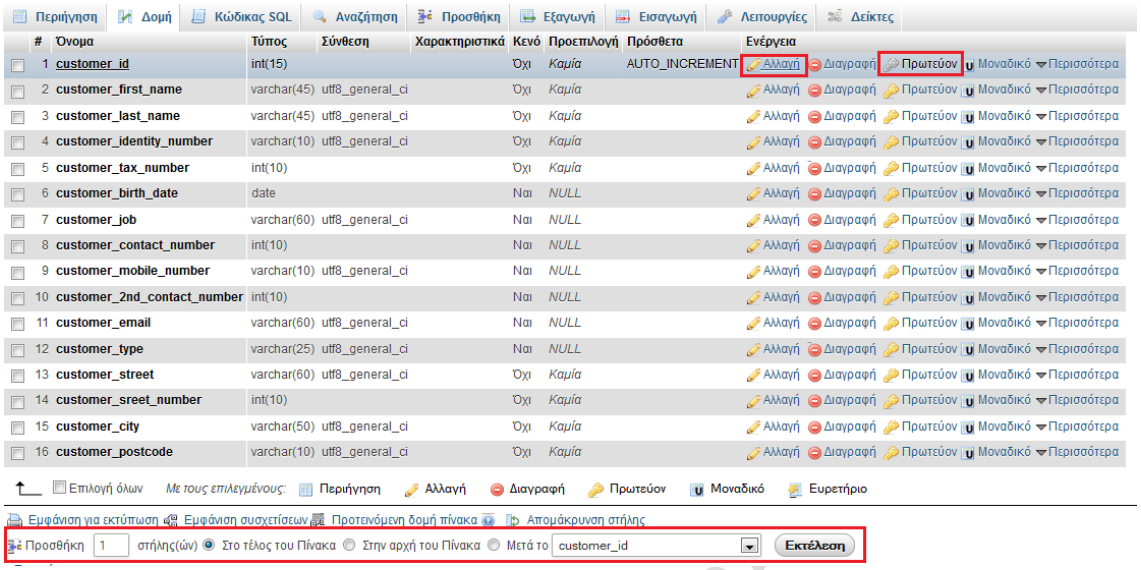
Εικόνα 9. Δημιουργία πίνακα στο phpMyAdmin

Για κάθε πεδίο δίνουμε όνομα και επιλέγουμε τον τύπο, το μήκος και τη σύνθεση για το κάθε ένα. Επίσης εδώ ορίζουμε αν το πεδίο μπορεί να πάρει την τιμή null ή όχι (Κενό), αν η τιμή του πεδίου θα αυξάνεται αυτόματα (A_I) ορίζουμε πρωτεύοντα και μοναδικά (unique) κλειδιά και αν υπάρχουν ειδικά χαρακτηριστικά. Η οθόνη δίνουμε όλες τις παραπάνω εντολές είναι η παρακάτω:



Εικόνα 10. Ορισμός πεδίων πίνακα στο phpMyAdmin

Όταν ολοκληρώσουμε την διαδικασία αυτή επιλέγουμε αποθήκευση και από την επιλογή "Δομή" μπορούμε να βλέπουμε ανά πάσα στιγμή τη δομή του πίνακα και να κάνουμε αλλαγές σύμφωνα με τις ανάγκες που προκύπτουν. Σε τέτοιες περιπτώσεις βέβαια θα πρέπει να προσέχουμε τις συνδέσεις μεταξύ των πινάκων και τις λειτουργίες που είχαμε ορίσει αρχικά στη βάση. Για παράδειγμα αν για κάποιο λόγο θέλουμε να διαγράψουμε ένα πεδίο ή ένα πίνακα από τη βάση θα πρέπει να ελέγξουμε με αν υπάρχουν συνδέσεις (unions) και πώς επηρεάζονται από την διαγραφή αυτή. Στο παρακάτω σχήμα βλέπουμε τη μορφή της δομής του πίνακα "customer_info" της βάσης. Εδώ ξεχωρίζει κανείς την ένδειξη που μπορεί κάποιος να επιλέξει για να κάνει αλλαγές στη βάση, μπορεί να δει ποιο είναι το πρωτεύον κλειδί να δει τις συσχετίσεις του πίνακα αυτού με άλλα στοιχεία της βάσης και να προσθέσει ή να διαγράψει πεδία του πίνακα.



Εικόνα 10. Δομή Βάσης Δεδομένων στο phpMyAdmin

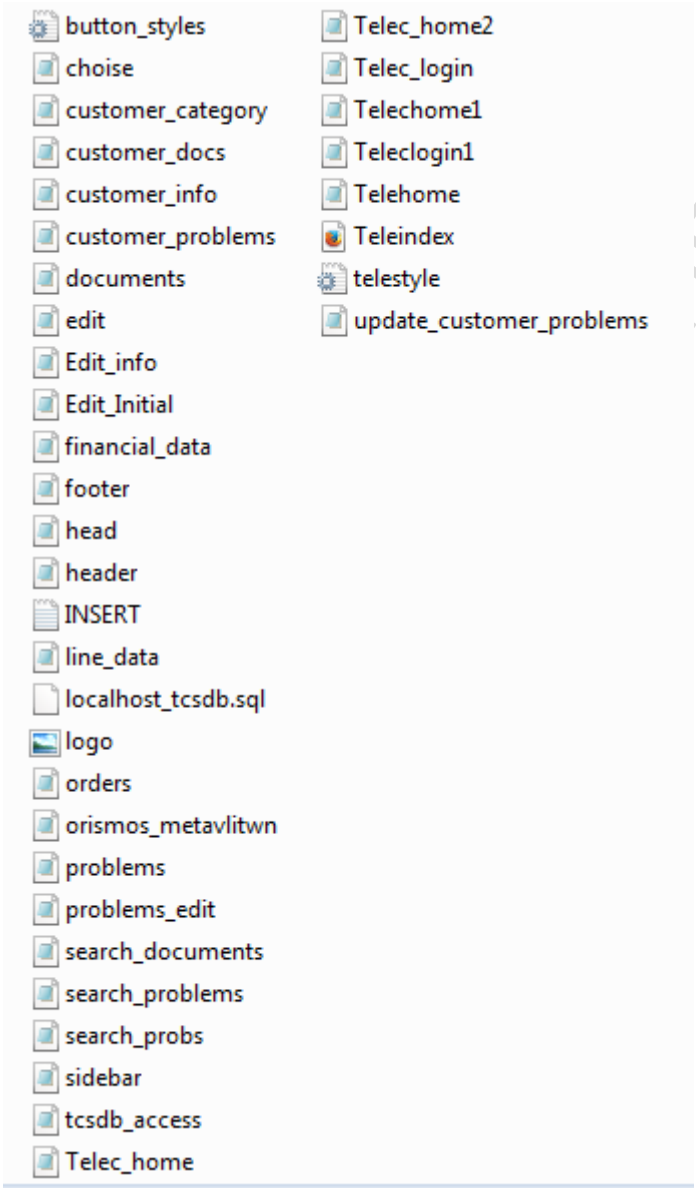
Το εργαλείο phpMyAdmin παρέχει τη δυνατότητα διαχείρισης των στοιχείων και της δομής μέσα από ένα παραθυρικό περιβάλλον όπως παρατηρήσαμε παραπάνω παρ' όλα αυτά έαν ο χρήστης θέλει να γράψει εντολές σε κώδικα SQL μπορεί να το κάνει ανά πάσα στιγμή κλικάροντας την επιλογή "Κώδικας SQL". Επίσης κάθε φορά που εκτελείται μια εντολή SQL στο εργαλείο στο επάνω μέρος της σελίδας εμφανίζεται ο κώδικας της εντολής που εντολή που εκτελέστηκε και μπορούμε να τον αποθηκεύσουμε για μελλοντική χρήση. Για παράδειγμα όταν δημιουργήσαμε τον πίνακα "customer_info" μέσω του εργαλείου η εντολή SQL που εκτελέστηκε ήταν:

```
CREATE TABLE IF NOT EXISTS `customer_info` (
  `customer_id` int(15) NOT NULL AUTO_INCREMENT,
  `customer_first_name` varchar(45) NOT NULL,
  `customer_last_name` varchar(45) NOT NULL,
  `customer_identity_number` varchar(10) NOT NULL,
  `customer_tax_number` int(10) NOT NULL,
  `customer_birth_date` date DEFAULT NULL,
  `customer_job` varchar(60) DEFAULT NULL,
  `customer_contact_number` int(10) DEFAULT NULL,
  `customer_mobile_number` varchar(10) DEFAULT NULL,
  `customer_2nd_contact_number` int(10) DEFAULT NULL,
  `customer_email` varchar(60) DEFAULT NULL,
  `customer_type` varchar(25) DEFAULT NULL,
  `customer_street` varchar(60) NOT NULL,
  `customer_sreet_number` int(10) NOT NULL,
  `customer_city` varchar(50) NOT NULL,
  `customer_postcode` varchar(10) NOT NULL,
  PRIMARY KEY (`customer_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;
```

Αξίζει να σημειωθεί ότι οι χρήστες που δημιουργήθηκαν για τη βάση έχουν πολύ συγκεκριμένα δικαιώματα και οι κωδικοί τους είναι κρυπτογραφμημένοι (χρήση της εντολής sha1()) για μεγαλυτέρη ασφάλεια. Επίσης όλα τα πεδία των πινάκων της βάσης είναι συγκεκριμένου τύπου και έχουν προκαθορισμένο μήκος.

5.2 Η Web εφαρμογή (CRM)

Σε αυτή την ενότητα περιγράφεται ο κώδικας που χρησιμοποιήθηκε για την μορφοποίηση της εφαρμογής CRM καθώς και για την εκτέλεση των διεργασιών της βάσης μέσω αυτής. Ξεκινώντας παρατίθενται όλα τα αρχεία (.php, .html & css) που δημιουργήθηκαν για την εφαρμογή.



Εικόνα 10. Αρχεία Υλοποίησης της βάσης δεδομένων και της εφαρμογής

Στα αρχεία “button_styles.css” και “telestyle.css” καθορίζεται η μορφοποίηση των κεντρικών δομών της εφαρμογής. Στο παρακάτω σχήμα βλέπουμε ένα μέρος του αρχείου “telestyle.css” στο οποίο καθορίζονται για παράδειγμα το χρώμα που θα υπάρχει πίσω από τη σελίδα (backround: 93A5C4;) το μέγιστο και ελάχιστο πλάτος του κεντρικού μέρους και το ότι δεν θα έχει πλαίσιο. Παρακάτω υπάρχει ο κώδικας για τη μορφή του header (σε επόμενο σημείο του αρχείου και του footer), της στήλης που θα εμφανίζεται αριστερά από το κεντρικό μένου της σελίδας, το backround του κεντρικού μέρους και υπάρχουν και μορφοποιήσεις για τη γραματοσειρές και άλλα στις επόμενες γραμμές του κώδικα.

```
.container {
    width: 90%;
    max-width: 1260px;/* a max-widt
    min-width: 780px;/* a min-width
    background: #93A5C4;
    margin: 0 auto; /* the auto val
}
/* ~~ the header is not given a wid
.header {
    background: #6F7D94;
    padding-bottom: 0px;
}

.sidebar1 {
    float: left;
    width: 15%;
    background: #93A5C4;
    padding-bottom: 10px;
}
.content {
    padding:0px 0;

    background: #93A5C4;
}
```

Στο δεύτερο .css αρχείο που χρησιμοποιούμε καθορίζεται η μορφή των “buttons” που χρησιμοποιούνται στην εφαρμογή.

```
.button {
    /*basic styles*/
    width: 250px;  height: 50px;  color: white; background-color: #99CF00;
    text-align: center;  font-size: 30px;  line-height: 50px;

    /*gradient styles*/
    background: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#99CF00), to(#6DB700));
    background: -moz-linear-gradient(19% 75% 90deg,#6DB700, #99CF00);

    /*border styles*/
    border-left: solid 1px #c3f83a;
    border-top: solid 1px #c3f83a;
    border-right: solid 1px #82a528;
    border-bottom: solid 1px #58701b;
    -moz-border-radius: 10px;
    -webkit-border-radius: 10px;
    border-radius: 10px;
    -webkit-gradient(linear, 0% 0%, 0% 100%, from(#99CF00), to(#6DB700))
}

.button h3 {
    font-size: 20px;
    line-height: 35px;
    font-family: helvetica, sans-serif;
}

.button p {
    font-size: 12px;
    line-height: 4px;
    font-family: helvetica, sans-serif;
}
```

Εδώ καθορίζεται το μέγεθος τους (width, height) το μέγεθος και η μορφή των γραμμάτων(font-size, font-family) το χρώμα τους και κάποια άλλα στοιχεία μορφοποίησης.

Με τη χρήση των .css αρχείων επιτυγχάνεται ο καθορισμός των μορφών αυτών μόνο μια φορά από τον χρήστη και για τις υπόλοιπες σελίδες που θα δημιουργηθούν απλά θα καλούνται αυτά τα αρχεία στην αρχή του κώδικα των php αρχείων και θα περνάνε τη μορφοποίηση στο site.

Το αρχείο “header.php” καθορίζει την κλάση header η οποία χρησιμοποιείται για τη μορφή της κεφαλίδας του CRM. Ο κώδικας html – javascript είναι ο εξής:

```
<div class="header">

 <align="left" a href="http://localhost/Web_Interface/Teleindex.html" class="header">
 <img src = "http://localhost/Web_Interface/logo.jpg"  width="15%" style="background: #8090AB; display:block;"/>

   <div align="right">
   <script language = "javascript">
   var now=new Date();
   var nday = now.getDay();
   var ndate = now.getDate();
   var nmonth = now.getMonth()+1;
   var nyear = now.getFullYear();
   document.writeln(ndate + "/" + nmonth + "/" +nyear);
   </script>
   </div>
   </div><!-- end .header -->
```

Η κλάση header σε συνδυασμόμε τα css αρχεία μας δίνει την κεφαλίδα του CRM που βλέπετε στην επόμενη εικόνα:

Εικόνα 11. Header εφαρμογής

Αντίστοιχα ο κώδικας για την κλάση footer που προσδιορίζει τι θα εμφανίζεται στο κάτω μέρος της σελίδας είναι:

```
<html>

<div class="footer">
  <p align="center">Telecomunication company - Athens - tel: 2110000000 - email:</p>
<!-- end .footer --></div>

</html>
```

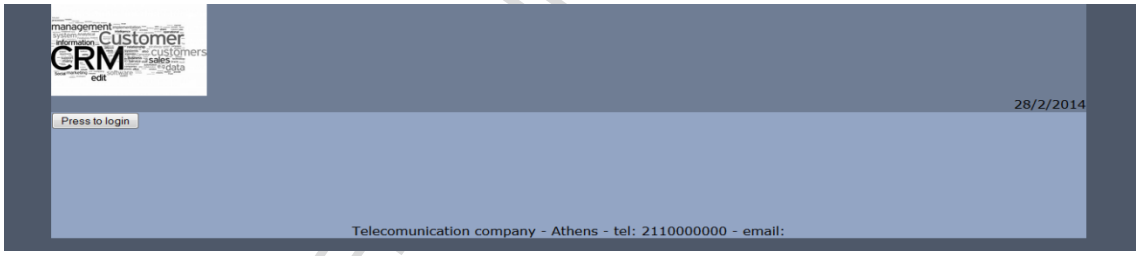
Αυτή η κλάση χρησιμοποιείται σε κάθε αρχείο που έχουμε φτιάξει για τις διάφορες σελίδες της εφαρμογής και έτσι σε όποια σελίδα και αν είμαστε στο τέλος της σελίδας εμφανίζεται η παρακάτω εικόνα:

Εικόνα 12. Footer εφαρμογής

Αυτός είναι ο κώδικας της αρχικής σελίδας της Web εφαρμογής

```
<head>
<link rel="stylesheet" type="text/css" href="telestyle.css"/>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8-generic-ci">
<title>Telecomunication company</title>
</head>
<body>
<div class="container">
  <div class="header">

    <align="left" a href="http://localhost/Web_Interface/Teleindex.html"> <img src = "http://localhost/Web_Interface/logo.jpg"
    width="15%" padding-bottom: 10px; style="background: #8090AB; display:block; "/>
    <div align="right">
    <script language = "javascript" >
    var now=new Date();
    var nday = now.getDay();
    var ndate = now.getDate();
    var nmonth = now.getMonth()+1;
    var nyear = now.getFullYear();
    document.writeln(ndate + "/" + nmonth + "/" +nyear);
    </script>
    </div> <!--end right-->
  </div> <!--end header-->

  <div class="content">
    <form method-="post" action="Teleclogin1.php">
    <table width=0 cellpadding=0 cellspacing=0 border=0><tr>
    <td colspan="4" align="center">
    <input type="submit" value="Press to login" class="button">
    </td>
    </tr>
    </table>
    </form>
```

Ο οποίος όταν εκτελεστεί μας δίνει:



Εικόνα 13. Αρχική Σελίδα

Εάν κάποιος επιλέξει να κάνει login στην εφαρμογή εμφανίζεται η σελίδα

Εικόνα 14. Φόρμα πιστοποίησης

που είναι η φόρμα εισαγωγής για το όνομα και τον κωδικό του χρήστη που χρειάζονται για την είσοδο στην εφαρμογή.Ο κώδικας για τη φόρμα είναι:

```
<form method="post" action="Telec_login.php">
<p>Username<br><input type="text" name="name"></br></p>
<p>Password<br><input type="password" name="password"></br></p>
<p><input type="submit" value="Log In"></p>
</form>
```

Και ο κώδικας που διαβάζει τα δεδομένα που εισάγει ο χρήστης και συνδέεται με τη βάση για να ελέγξει ότι το όνομα χρήστη και ο κωδικός είναι έγκυρα ώστε να δώσει πρόσβαση στον συγκεκριμένο χρήστη είναι:

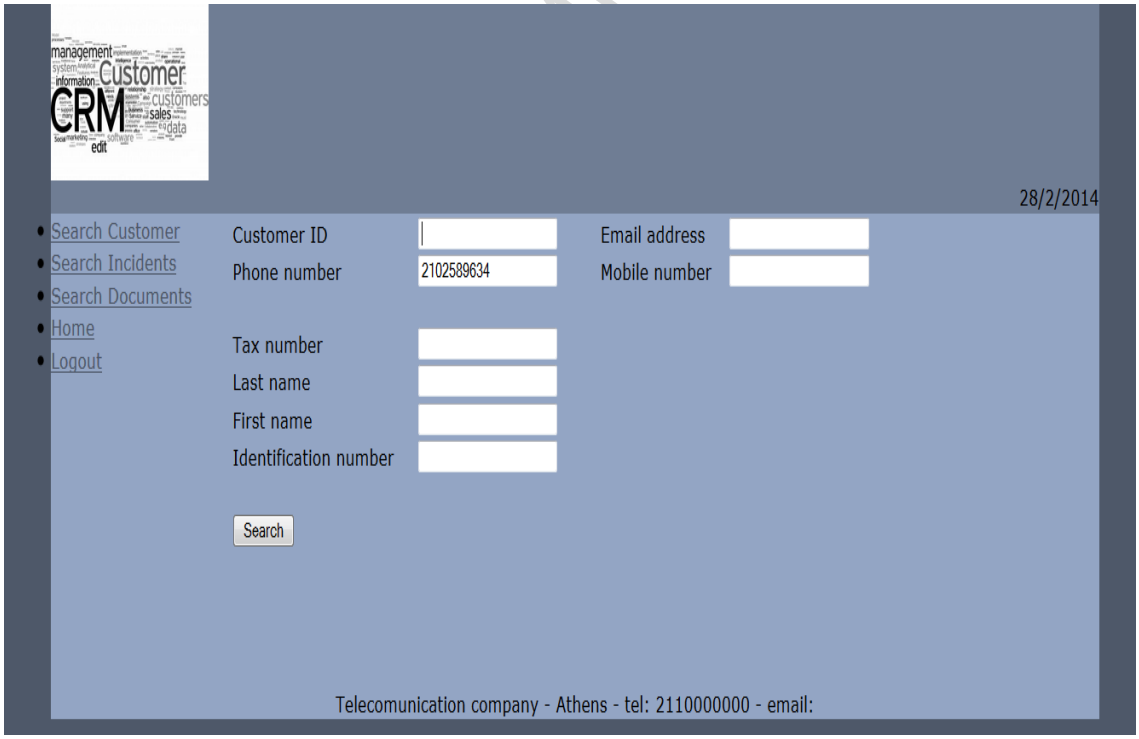
```
<?php
    $name = $_POST['name'];
    $password = $_POST['password'];
    if(empty($name)||empty($password))
    {
?>
    <form method="post" action="Telec_login.php">
    <p>You must enter a valid username and password !</p>
    <p>Username<br><input type="text" name="name"></br></p>
    <p>Password<br><input type="password" name="password"></br></p>
    <p><input type="submit" value="Log In"></p>
    </form>

<?php
    }
    else
    {
        $mysql = mysqli_connect('localhost', 'root', 'elen2013');
        if(!$mysql)
        {   echo 'Cannot connect to database';
            //exit;
        }
        //Select database
        $selected = mysqli_select_db($mysql, 'tcsdb');
        if(!$selected)
        {   echo 'Cannot select database.';
            //exit;
        }
         $query= "select count(*) from autha_users where
                  username = '$name' and
                  password = sha1('$password')";
         $result= mysqli_query($mysql, $query);
         if(!$result)
         {  echo 'Cannot run query.';
```

```
            //exit;
        }
    $row = mysqli_fetch_row($result);
    $count = $row[0];
    if ($count > 0)
    {   header("location: http://localhost/Web_Interface/Telechome1.php");
    }
    else
    {   header("location: http://localhost/Web_Interface/Teleclogin1.php");
        //echo 'Fail to connect';
        //exit;
    }
}
?>
```

Όπως φαίνεται και στον κώδικα αυτό η php χρησιμοποιεί την συνάρτηση sha1() για να αποκρυπτογραφήσει τον κωδικό ώστε να μπορέσει να αναγνωριστεί από τον πίνακα κωδικών της βάσης. Στην περίπτωση που ο χρήστης δεν αναγνωριστεί η εφαρμογή τον ενημερώνει ότι θα πρέπει να δώσει έγκυρους κωδικούς με το μήνυμα:

«You must enter a valid username and password !»

Εάν ο χρήστης είναι όντως εξουσιοδοτημένος η εφαρμογή αναγνωριζεί τα στοιχεία του και του δίνει πρόσβαση στη σελίδα αναζήτησης πελατών που φαίνεται παρακάτω:



Εικόνα 15. Σελίδα Αναζήτησης πελατών

Όπως μπορείτε να παρατηρήσετε ο χρήστης έχει τη δυνατότητα να αναζητήσει τον πελάτη με πάρα πολλά δεδομένα για την καλύτερη εξυπηρέτηση και του πελάτη αλλά και των χρηστών. Ο κώδικας που χρησιμοποιήθηκε για την μορφή και τη λειτουργία της σελίδας αυτής είναι:

```
<html>
  <?php include 'head.php' //dhmiourgia head?>

<body>

<div class="container">
  <?php include 'header.php' //dhmiourgia header ?>

<div class="content">
   <?php include 'sidebar.php' //dhmioyrgia sidebar epilogwn?>



    <form method="post" action="Telec_home.php">
      <table width="65%" height="30%" border="0">
        <tr>
          <td>   </td>
          <td>Customer ID</td>
          <td><input type="text" name="customerid"></td>
          <td>   </td>
          <td>Email address</td>
          <td><input type="text" name="emailaddress"></td>
        </tr>
        <tr>
          <td>   </td>
          <td>Phone number</td>
          <td><input type="text" name="phonenumber"></td>
          <td>   </td>
          <td>Mobile number</td>
          <td><input type="text" name="mobilenumber"></td>
        </tr>
```

```
    </tr>
    <tr><td> </td></tr>
    <tr>
      <td>   </td>
      <td>Tax number</td>
      <td><input type="text" name="taxnumber"></td>
    </tr>
    <tr>
    <td>   </td>
    <td>Last name</td>
    <td><input type="text" name="lastname"></td>
    </tr>
    <tr>
    <td>   </td>
    <td>First name</td>
    <td><input type="text" name="firstname"></td>
    </tr>
    <tr>
    <td>   </td>
    <td>Identification number</td>
    <td><input type="text" name="idnumber"></td>
    </tr>
    <tr><td> </td></tr>
    <tr>
    <td>   </td>
    <td><input type="submit" value="Search" class="button";></td>
    </tr>
  </table>
  </form>
```

Μόλις ο χρήστης δώσει τα στοιχεία και επιλέξει αναζήτηση (search) εμφανίζεται μια σελίδα με τις διάφορες πληροφορίες στις οποίες έχει πρόσβαση ο χρήστης. Στις εικόνες που παρατίθενται ο χρήστης είναι ο administrator ο οποίος έχει πρόσβαση σε όλες τις πληροφορίες του πελάτη και επίσης έχει δικαιώματα εισαγωγής/διαγραφής και ενημέρωσης για όλα τα στοιχεία της βάσης. Η εκόνα που εμφανίζεται στο χρήστη για να επιλέξει ποιες πληροφορίες του πελάτη τον ενδιαφέρουν είναι αυτή του Σχήματος 11. Για να εμφανιστεί φυσικά η εικόνα που βλέπουμε στο σχήμα 11, στον πελάτη από πίσω τρέχει και πάλι ένα αρχείο php το οποίο:

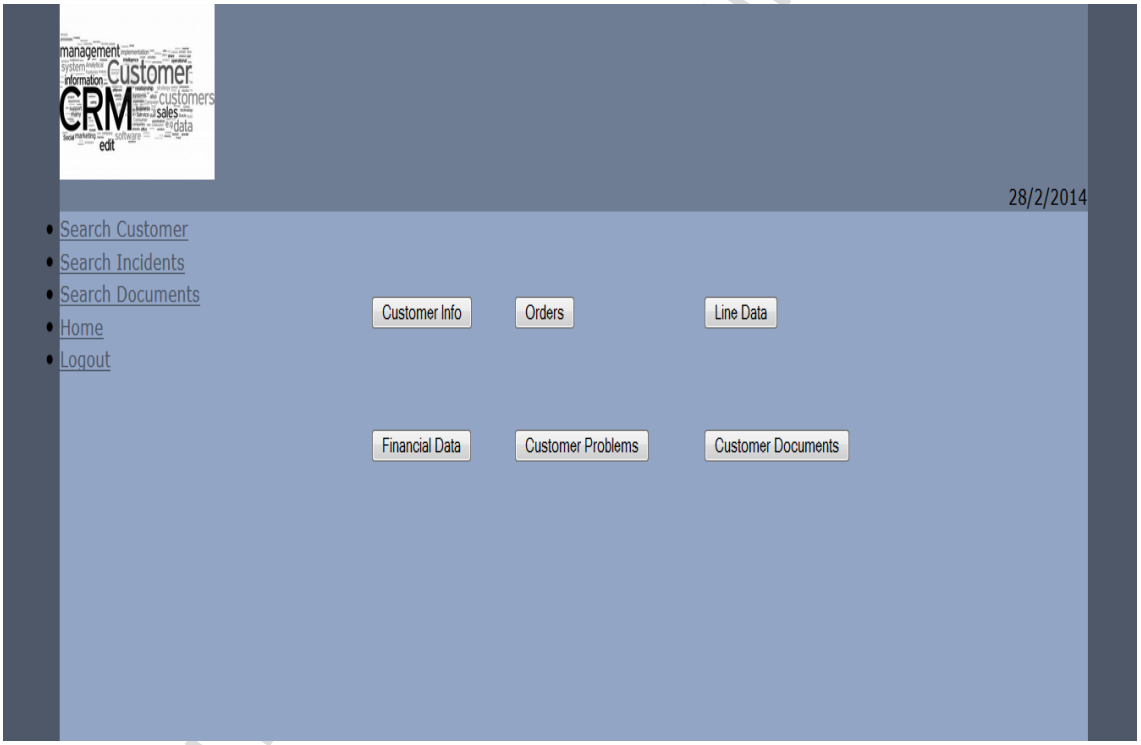
- Αναζητά το στοιχείο του πελάτη που έδωσε ο χρήστης ώστε να εντοπίσει την εγγραφή στη βάση και να επιστρέψει τις ανάλογες πληροφορίες
- Στην περίπτωση που τα στοιχεία του πελάτη δεν βρεθούν δεν επιστρέφει τίποτα
- Στην περίπτωση που αναγνωρίσει τα στοιχεία του χρήστη διαμορφώνει ανάλογα τη σελίδα με τις κατηγορίες των δεδομένων του πελάτη και την εμφανίζει στην οθόνη.

Ο κώδικας που υλοποιεί όλα τα παραπάνω είναι:

```php
<?php

    session_start();

    //orismos global metavlitwn
    $_SESSION['phonenumber'] = $_POST['phonenumber'];
    $_SESSION['customerid']=$_POST['customerid'];
    $_SESSION['emailaddress']= $_POST['emailaddress'];
    $_SESSION['mobilenumber'] = $_POST['mobilenumber'];
    $_SESSION['taxnumber'] = $_POST['taxnumber'];
    $_SESSION['lastname'] = $_POST['lastname'];
    $_SESSION['firstname'] = $_POST['firstname'];
    $_SESSION['idnumber'] = $_POST['idnumber'];

    //orismos topikwn metavlitwn
    include 'orismos_metavlitwn.php';


    //An den dothei kanena stoixeio epistrefei sthn arxikh selida
    if(empty($phonenumber)&& empty($customerid)&& empty($emailaddress)&& empty($mobilenumber)&&
    empty($taxnumber)&& empty($lastname)&& empty($firstname)&& empty($idnumber))
    {
        header("location: http://localhost/Web_Interface/Telechome1.php");
    }

    //An oxi syndeetai me th vasi gia na eleksei ta dothenta stoixeia
    $mysql = mysqli_connect('localhost', 'tcsdbadmin', 'tcs@adm1');
    if(!$mysql)
    {
        echo 'Cannot connect to database';
```

```php
    }
    //Select database
    $selected = mysqli_select_db($mysql, 'tcsdb');
    if(!$selected)
    {   echo 'Cannot select database.';
        //exit;
    }

    $query_1= "select * from line_data where
    line_number = '$phonenumber'";

    $result_1= mysqli_query($mysql, $query_1);

    //an to result_1 einai null h an exei 0 grammes dhl to query_1 trexei
    //alla de dinei apotelesma trexei to query_2
    if ((!$result_1) or (mysqli_num_rows($result_1) < 1))
    {
        $query_2= "select * from customer_info where
                    customer_id = '$customerid'or
                    customer_email = '$emailaddress' or
                    customer_mobile_number = '$mobilenumber' or
                    customer_tax_number = '$taxnumber' or
                    customer_last_name = '$lastname' or
                    customer_first_name = '$firstname' or
                    customer_identity_number = '$idnumber'";


        $result_2= mysqli_query($mysql, $query_2);

        //an to result_1 einai null h an exei 0 grammes (dhl to query_1 trexei
```

```
        //an to result_1 einai null h an exei 0 grammes (dhl to query_1 trexei
        //alla de dinei apotelesma) epistrefei sthn arxikh selida
        if((!$result_2) or (mysqli_num_rows($result_2) < 1))
        {
            header("location: http://localhost/Web_Interface/Telechome1.php");
        }

            //Diaforetika syndeetai me to arxeio poy perilamvanei ta
            //stoixeia tou pelath poy vrethike
            else
            {
                header("location: http://localhost/Web_Interface/Telec_home2.php");
            }
    }

    //An to result_1 den einai keno kai yparxei kataxwrhsh sth vash syndeetai me to arxeio
    //poy perilamvanei ta stoixeia tou pelath poy vrethike
    else
    {
        header("location: http://localhost/Web_Interface/Telec_home2.php");
    }

?>
```



Εικόνα 16. Κατηγορίες Αποθηκευμένων Πληφοριών του πελάτη

Από αυτή την οθόνη ο χρήστης μπορεί να επιλέξει να πάρει πληροφορίες από μία ή από όλες τις κατηγορίες με τα δεδομένα του πελάτη. Για να είναι η σελίδα πιο λειτουργική και να μην χρειάζεται ο χρήστης να δίνει κάθε τα στοιχεία του πελάτη για κάθε πληροφορία που χρειάζεται ξανά και ξανά έχει χρησιμοποιηθεί το αρχείο “orismos_metavlitwn.php” στο οποίο, οποιοδήποτε στοιχείο και αν δώσει ο πελάτης αποθηκεύται προσωρινά σαν τοπική μεταβλητή και χρησιμοποιείται και κατόπιν το σύστημα αποθηκεύει σε μια άλλη, επίσης προσωρινή μεταβλητή, τον κωδικό του πελάτη (customer id) για να δίνει πληροφορίες για τον συγκεκριμένο πελάτη μέχρι ο χρήστης να επιλέξει κάποια από τις επιλογές αριστερά της σελίδας (Σχήμα 12) για να

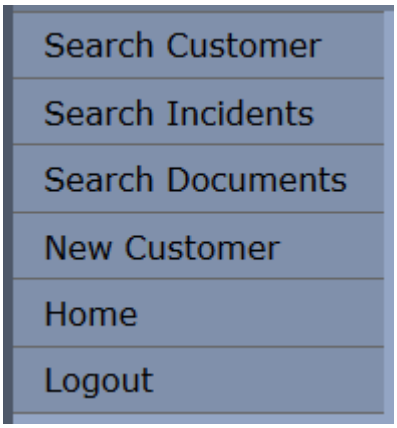
διαλέξει άλλο πελάτη ή να βγει τελείως από την εφαρμογή. Ο κώδικας του παραπάνω αρχείου είναι:

```php
<?php

    //orismos topikwn metavlitwn
    $phonenumber=$_SESSION['phonenumber'];
    $customerid=$_SESSION['customerid'];
    $emailaddress=$_SESSION['emailaddress'];
    $mobilenumber=$_SESSION['mobilenumber'];
    $taxnumber=$_SESSION['taxnumber'];
    $lastname=$_SESSION['lastname'];
    $firstname=$_SESSION['firstname'];
    $idnumber=$_SESSION['idnumber'];

    $mysql = mysqli_connect('localhost', 'tcsdbadmin', 'tcs@adm1');
    if(!$mysql)
    {
        echo 'Cannot connect to database';
        //exit;
    }
    //Select database
    $selected = mysqli_select_db($mysql, 'tcsdb');
    if(!$selected)
    {   echo 'Cannot select database.';
        //exit;
    }

$phone_cus_id= "select customer_id from line_data where line_number = '$phonenumber'";
$cs_1= mysqli_query($mysql, $phone_cus_id);


//an to cs_1 einai null h an exei 0 grammes dhl to query_1 trexei alla de dinei apotelesma trexei to query_2
if ((!$cs_1) or (mysqli_num_rows($cs_1) < 1))
{
```


```php
if ((!$cs_1) or (mysqli_num_rows($cs_1) < 1))
{
```

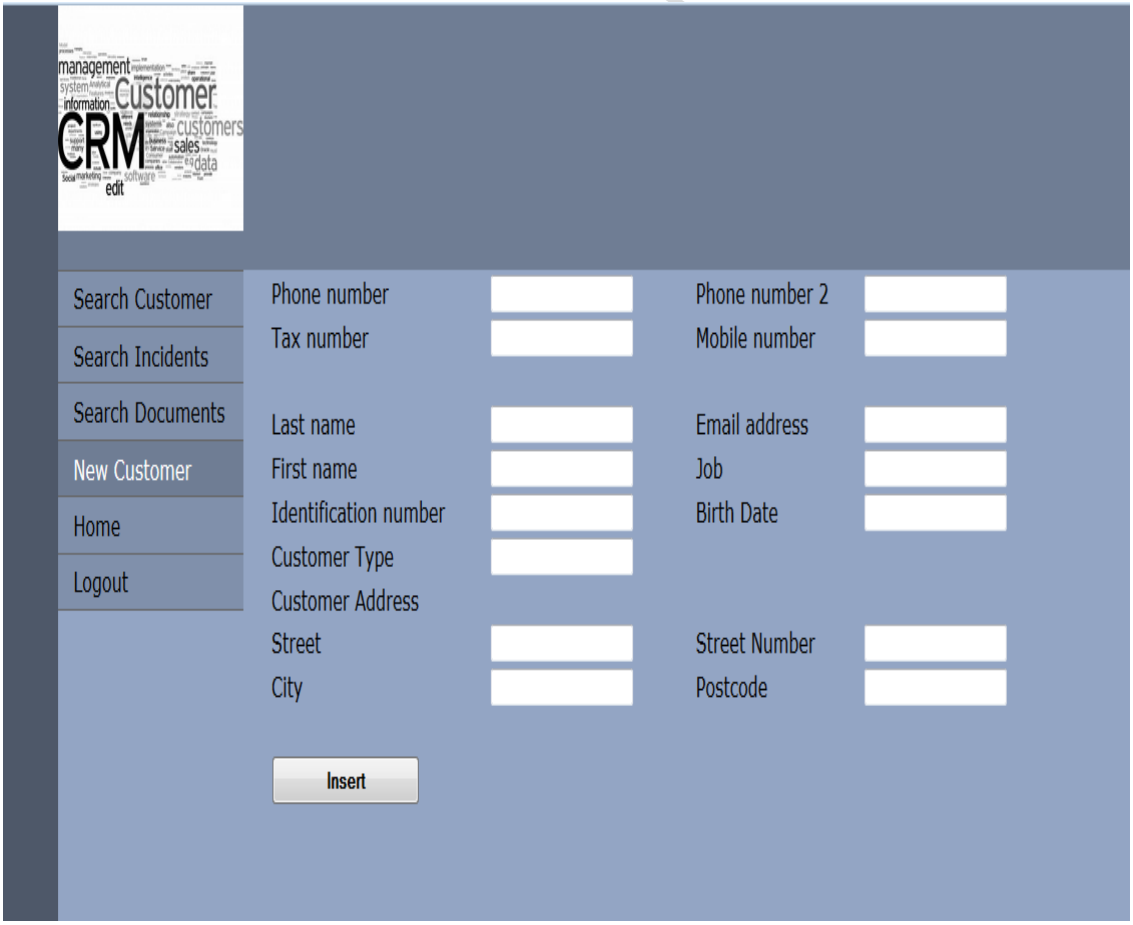

```php
  $cusinf_cus_id = "select customer_id from customer_info where
                customer_id = '$customerid'or
                customer_email = '$emailaddress' or
                customer_mobile_number = '$mobilenumber' or
                customer_tax_number = '$taxnumber' or
                customer_last_name = '$lastname' or
                customer_first_name = '$firstname' or
                customer_identity_number = '$idnumber'";

  $cs_2= mysqli_query($mysql, $cusinf_cus_id);
  if((!$cs_2) or (mysqli_num_rows($cs_2) < 1))
        {
            echo 'No data exist';
        }

  while($row2 = mysqli_fetch_array($cs_2))
            {
                $cust_id=$row2['customer_id'];  //an o xristis exei dosei opoiodhpote allo
                                                //stoixeio ektos apo thl to customer_id orizetai apo ton pinaka customer_info
                //echo $cust_id;
                echo '<br>';}
}
else
{   while($row = mysqli_fetch_array($cs_1))
            {
                $cust_id=$row['customer_id'];   //an o xristis exei dosei arithmo thlefwnou to customer_id
                                                //orizetai apo ton pinaka line_data
                //echo $cust_id;
                echo '<br>';}
}
```

Μετά από αυτή τη διαδικασία το σύστημα δίνει στον χρήστη τις πληροφορίες που αναζητά τρέχοντας για κάθε αναζήτηση σε στοιχεία διαφορετικού πίνακα της βάσης ένα αρχείο της μορφής όπως αυτή του “line_data.php” ο κώδικας του οποίου παρατίθεται παρακάτω.



Εικόνα 17. Επιλογές αναζήτησης και επιστροφής της εφαρμογής

Επίσης ο χρήστης μπορεί να εισάγει δεδομένα από την web εφαρμογή στη βάση όπως φαίνεται στην παρακάτω εικόνα:



Εικόνα 1. Φόρμα εισαγωγής δεδομένων στη βάση

Μέρος του κώδικα που εκτελείται για την εμφάνιση της φόρμας εισαγωγής δεδομένων φαίνεται παρακάτω:

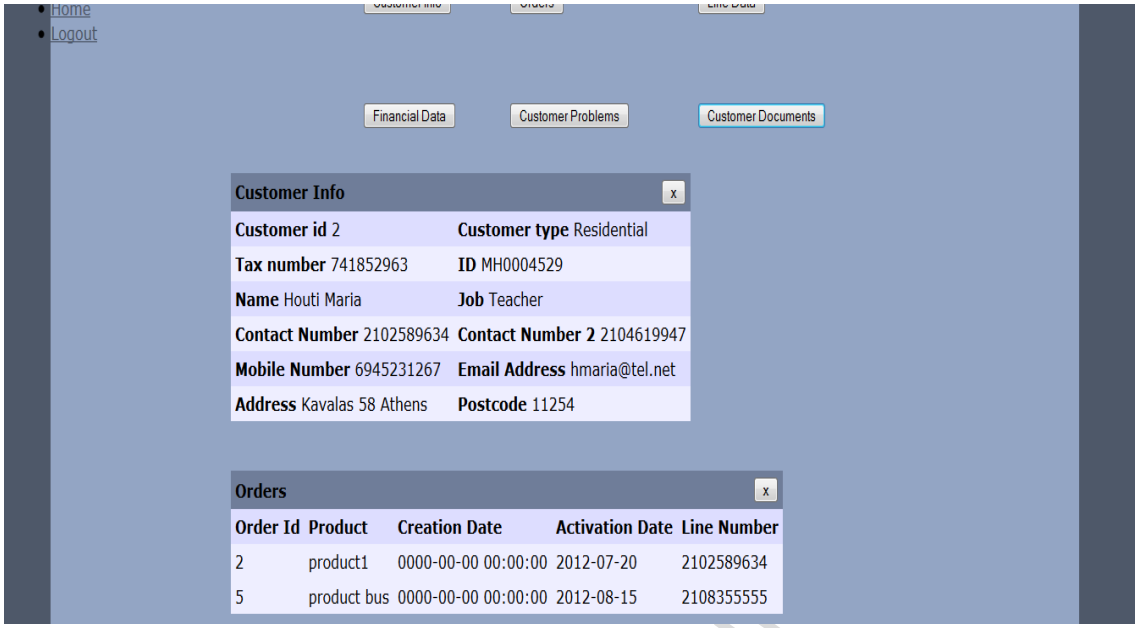
```
<div class="content">
    <form method="post" action="New_Customer_Insert_2.php">
     <table width="65%" height="30%" border="0">
       <tr>
         <td>   </td>
         <td>Phone number</td>
         <td><input type="text" name="phonenumber"></td>
         <td>   </td>
         <td>Phone number 2</td>
         <td><input type="text" name="phonenumber_2"></td>
       </tr>
       <tr>
         <td>   </td>
         <td>Tax number</td>
         <td><input type="text" name="taxnumber"></td>
         <td>   </td>
         <td>Mobile number</td>
         <td><input type="text" name="mobilenumber"></td>
       </tr>
       <tr><td> </td></tr>
       <tr>
       <td>   </td>
       <td>Last name</td>
       <td><input type="text" name="lastname"></td>
       <td>   </td>
         <td>Email address</td>
         <td><input type="text" name="emailaddress"></td>
       </tr>
       <tr>
       <td>   </td>
       <td>First name</td>
       <td><input type="text" name="firstname"></td>
```

Επίσης οι εντολές SQL που εκτελούνται μέσω της php για να καταγραφούν τα στοιχεία στη βάση είναι :

```php
    $phonenumber=$_POST['phonenumber'];
    $emailaddress=$_POST['emailaddress'];
    $mobilenumber=$_POST['mobilenumber'];
    $taxnumber=$_POST['taxnumber'];
    $lastname=$_POST['lastname'];
    $firstname=$_POST['firstname'];
    $idnumber=$_POST['idnumber'];
    $birthdate=$_POST['birthdate'];
    $job=$_POST['job'];
    $phonenumber_2=$_POST['phonenumber_2'];
    $type=$_POST['type'];
    $customer_street=$_POST['customer_street'];
    $customer_street_number=$_POST['customer_street_number'];
    $customer_city=$_POST['customer_city'];
    $customer_postcode=$_POST['customer_postcode'];


    $mysql = mysqli_connect('localhost', 'tcsdbadmin', 'tcs@adm1');
    if(!$mysql)
    {
        echo 'Cannot connect to database';
        //exit;
    }
    //Select database
    $selected = mysqli_select_db($mysql, 'tcsdb');
    if(!$selected)
    {   echo 'Cannot select database.';
        //exit;
    }
```

```php
$customerinsert =
"INSERT INTO customer_info (customer_identity_number, customer_tax_number,
customer_birth_date, customer_job, customer_contact_number, customer_mobile_number,
customer_2nd_contact_number, customer_email, customer_type, customer_street,
customer_street_number, customer_city, customer_postcode)
VALUES ('$firstname','$lastname', '$idnumber', '$taxnumber', '$birthdate',
'$job', '$phonenumber','$mobilenumber','$phonenumber_2','$emailaddress',
'$type','$customer_street','$customer_street_number','$customer_city','$customer_postcode')";
$customerinsert = mysqli_query($mysql, $customerinsert);

$customerinsert_2 = "INSERT INTO customer_info (customer_first_name, customer_last_name)
VALUES (AES_ENCRYPT('$firstname','153'),AES_ENCRYPT('$lastname','154'))";
$customerinsert_2 = mysqli_query($mysql, $customerinsert_2);

echo "Data Entry was Successful";

?>
```

Στην παρακάτω εικόνα φαίνεται μια ενδεικτική εικόνα που έχει ο χρήστης με τις πληροφορίες του πελάτη όταν έχουν εκτελεστεί σωστά όλες οι παραπάνω διαδικασίες.



Εικόνα 18. Πληροφορίες από τη βάση στη Web εφαρμογή.

Όπως αναφέρθηκε και νωρίτερα ο κώδικας που εκτελείται για κάθε πίνακα πληροφοριών του πελάτη που εμφανίζεται στην παραπάνω εικόνα είναι της μορφής (από αρχείο line_data.php):

```
<?php

$linenumber = "select * from line_data where customer_id = '$cust_id'";
$linenumb = mysqli_query($mysql, $linenumber);
//$row5 = mysqli_fetch_array($linenumb);


        echo "
        <table id=\"lndata\" border=\"0\" width=\"65%\" cellpadding=\"5\" cellspacing=\"0\" align=\"center\" style=\"display:none\">
        <tr bgcolor=\"#6F7D99\">
        <td align=\"left\"><b>Line Data</b></td><td> </td><td> </td><td> </td><td> </td>
        <td align=\"right\"><input type=\"button\" id=\"buttontel\" value=\"x\" onclick=\"tablehide('lndata')\" /></td>
        </tr>
        <tr bgcolor=\"#ddddff\">
        <td align=\"left\"><b>Line Number</b></td><td align=\"left\"><b>Sip Number</b>
        <td align=\"left\"><b>Urban Center</b></td><td align=\"left\"><b>IP Address</b></td>
        <td align=\"left\"><b>MAC Address</b></td><td align=\"left\"><b>Order ID</b></td>
        </tr>";
        while($row5 = mysqli_fetch_array($linenumb))
        {
        echo "
        <tr bgcolor=\"#eeeeff\"><td align=\"left\"> ".$row5['line_number']."</td><td align=\"left\"> ".$row5['sip_number']."</td>
        <td align=\"left\">".$row5['urban_center']."</td><td align=\"left\"> ".$row5['IP_address']."</td>
        <td align=\"left\"> ".$row5['router_mac_address']."</td><td align=\"left\"> ".$row5['order_id']."</td>
        </tr>";
        }
        echo "
        </table>
        <br> </br>";
?>
```

Κεφάλαιο 6ο

6.1 Συμπεράσματα

Στην εργασία αυτή παρουσιάστηκαν κάποια από τα πιο βάσικα θέματα ασφάλειας βάσεων δεδομένων. Έγινε ανάλυση των απαιτήσεων ασφάλειας που πρέπει να γνωρίζει ο διαχειριστής ενός Συστήματος Βάσεων Δεδομένων και παρουσιάστηκαν ορισμένα μέτρα που μπορούν να εφαρμοστούν ώστε να διασφαλιστεί η ασφάλεια των δεδομένων.

Στο κεφάλαιο που αναφέρονται οι απαιτήσεις ασφάλειας μπορεί να διαπιστώσει κανείς είναι ότι οι απαιτήσεις για ασφάλεια ενός ΣΔΒΔ αφορούν όλα τα επίπεδα του συστήματος (π.χ. δομές, δεδομένα κ.λ.π.). Για παράδειγμα δεν αρκεί μόνο να διασφαλιστεί η ακεραιότητα των δεδομένων μιας βάσης θα πρέπει να γίνονται και έλεγχοι πρόσβασης σε αυτή, να υπάρχουν και μηχανισμοί κρυπτογράφησης και πολλές άλλες λειτουργίες ώστε να ικανοποιούνται οι απαιτήσεις που έχουν τεθεί. Η ανάλυση των απαιτήσεων ασφάλειας της βάσης δεδομένων είναι απαραίτητη προϋπόθεση για το σχεδιασμό ενός συστήματος που θα είναι λειτουργικό και θα ικανοποιεί τις απαιτήσεις των χρηστών του ενώ παράλληλα θα διασφαλίζει στους διαχειριστές του ότι υπάρχουν οι μηχανισμοί εκείνοι που προστατεύουν τις ευαίσθητες πληροφορίες και τις διεργασίες που είναι απαραίτητες για αυτούς.

Με βάση κάποιες από τις πιο γνωστές απειλές – αδυναμίες στα συστήματα βάσεων δεδομένων δόθηκαν ορισμένα από τα μέτρα ασφάλειας που προτείνονται για την προστασία των ευαίσθητων δεδομένων ενός τέτοιου συστήματος. Από αυτό το κεφάλαιο σε συνδυασμό με τις ευπάθεις του WAMP και των εφαρμογών του, όπως αυτές παρουσιάζονται στο κεφάλαιο 4, θα μπορούσε κανείς να συμπεράνει ότι η σωστή Διαχείριση των δικαιώματων των χρηστών μιας βάσης είναι ένα από τα πιο βασικά σημεία που θα πρέπει να προσέξει ο δημιουργός ενός τέτοιου συστήματος. Ιδιαίτερη βαρύτητα – χωρίς αυτό να σημαίνει ότι άλλες απειλές ασφάλειας πρέπει να αγνοηθούν - θα πρέπει να δίνεται στην πρόληψη των SQL Injections (π.χ. με μηχανισμούς που μπορούν να αναγνωρίζουν αυτού του είδους τα πρότυπα) καθώς αποτελεί μια από τις πιο συχνές και καταστροφικές απειλές για τα συστήματα βάσεων δεδομένων.

Όπως αναφέρθηκε και στην αρχή αυτής μελέτης οι βάσεις δεδομένων των οποίων η διαχείριση γίνεται μέσω Web Εφαρμογών αποτελούν κύριο μέσο αποθήκευσης και διαχείρισης πληροφοριών σε πάρα πολλά συστήματα και σε πολλούς διαφορετικούς τομείς της κοινωνίας. Για αυτό το λόγο, και με δεδομένο ότι η τεχνολογία των υπολογιστών και των λειτουργιών που γίνονται μέσω Web εφαρμογών εξελίσεται με ραγδαίους ρυθμούς θα πρέπει το θέμα της ασφάλειας των δεδομένων να μην αποτελεί πολυτέλεια αλλά βασικό στοιχείο για το σχεδιασμό και την διατήρηση της ορθής λειτουργίας ενός συστήματος βάσεων δεδομένων.

6.2 Μελλοντική Εργασία

Σσε αυτή την ενότητα καταγράφονται κάποιες πτυχές του θέματος που δεν παρουσιάστηκαν σε αυτή την εργασία και θα μπορούσαν να αποτλέσουν το αντικείμενο μιας επόμενης μελέτης που θα γίνει σε συνέχεια της παρούσας.

Σε μια μελοντική εργασία θα μπορούσε να γίνει περαιτέρω ανάλυση των των πρακτικών για την ασφάλεια μιας βάσης δεδομένων. Η αναλυτική παρουσίση των συναρτήσεων που παρέχονται από την MySQL για την κρυπτογράφηση δεδομένων, στηλών ή και πινάκων είναι ένα ενδιαφέρον θέμα. Επίσης ο έλεγχος των μηχανισμών ασφάλειας που εφαρμόζοτναι σε μια βάση δεδομένων με την υλοποίηση εικονικών επιθέσεων στο σύστημα ώστε να φανεί στην πράξη πώς αντιδρά η βάση σε πιθανούς κινδύνους (όπως για παράδειγμα σε μια SQL Injection) και αν όντως οι μηχανισμοί που έχουν τεθεί είναι αποτελεσματικοί, θα μπορούσε να αναπτυχθεί ως επέκταση αυτής της εργασίας.

Σε επόμενη εργασία θα μπορούσε επσίσης να γίνει αναφορά και αναλυτική παρουσίαση των μηχανσιμών ασφάλειας που παρέχονται από άλλα συστήματα διαχείρισης βάσεων δεδομέων όπως για παράδειγμα η Oracle.

Βιβλιογραφία

1. The Apache Sotware Foundation (http:www.apache.org)
2. wikipedia (httpen.wikipedia.rg)
3. PHP: Data Base Security (http:www.php.org)
4. Παραδείγματα: w3schools.com (http://www.w3schools.com)
5. “Top 10 Data Base Threats” Impreva (http:www.impreva.com)
6. IT Security (http: netappsec.blogspot.gr)
7. Τεχνολογίες Διαδικτύου Αρχές Λειτουργίας & Προγραμματισμός Εφαρμογών στο Διαδίκτυο (Δουληγέρης Χρήστος, Κοπανάκη Εύη, Μαυροπόδη Ρόζα)
8. Ανάπτυξη Web Εφαρμογών με Php και MySQL (Luke Welling & Laura Thomson, Απόδοση: Γκλάβα Μαρία)
9. Data Base Security (Database Security Consortium)
10. ΣΗΜΕΙΩΣΕΙΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΑΣΦΑΛΕΙΑΣ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ (Δρ Ελευθέριος Μπόζιος)
11. ΠΡΟΣΤΑΣΙΑ ΚΑΙ ΑΣΦΑΛΕΙΑ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ (κ. Βασιλάκης)
12. Συστήματα Βάσεων Δεδομένων-Θεωρία και πρακτική εφαρμογή (Ιωάννης Μανωλόπουλος, Απόστολος Ν.Παπαδόπουλος)
13. Database Security and Encryption (Iqra Basharat, Farooque Azam, Abdul Wahab Muzaffar)
14. Understanding SQL Injection (http://www.cisco.com/web/about/security/intelligence/sql_injection.html)
15. SQL Injection Knowhow (http://www.codeproject.com/Articles/206814/SQL-Injection-Knowhow)
16. Impreva Data Security (http://blog.imperva.com)
17. Stackoverflow (http://stackoverflow.com/questions)

18. MySQL (http://dev.mysql.com/doc/refman/5.1/en/encryption-functions.html)