



ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ  
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ  
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εντοπισμός και αντιμετώπιση SQL ευπαθειών

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ  
ΤΣΟΥΓΓΟΥ ΑΡΓΥΡΩ

ΕΠΙΒΛΕΠΩΝ: ΔΡΑΖΙΩΤΗΣ ΚΩΣΤΑΝΤΙΝΟΣ

23 Σεπτεμβρίου 2024



ARISTOTETLE UNIVERSITY OF THESSALONIKI  
FACULTY OF SCIENCES  
COMPUTER SCIENCE DEPARTMENT

Detection and mitigation of SQL vulnerabilities

GRADUATE THESIS  
TSOUNGOU ARGYRO

SUPERVISOR:DRAZIOTIS KONSTANTINOS

23 September 2024



Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης  
Σχολή Θετικών Επιστημών  
Τμήμα Πληροφορικής

Copyright ©All rights reserved Tsoungou, 2024.

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα πτυχιακή εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στο πλαίσιο 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. 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.

Το περιεχόμενο αυτής της εργασίας δεν απηχεί απαραίτητα τις απόψεις του Τμήματος, του Επιβλέποντα, ή της επιτροπής που την ενέκρινε.

**Υπεύθυνη Δήλωση**

(Υπογραφή) .....

**Τσούγγου Αργυρώ**

# Περίληψη

Οι web εφαρμογές έχουν γίνει αναπόσπαστο μέρος της σύγχρονης κοινωνίας, διευκολύνοντας την επικοινωνία, το εμπόριο και την ανταλλαγή πληροφοριών. Ωστόσο, με την αυξανόμενη σημασία τους, προκύπτουν και σημαντικές προκλήσεις ασφάλειας. Η εργασία αυτή εστιάζει στην ασφάλεια των web εφαρμογών, με κύριο θέμα τις επιθέσεις SQL injection, μια από τις πιο γνωστές και κοινές απειλές.

Η εργασία ξεκινά με το θεωρητικό κομμάτι, το πρώτο κεφάλαιο ασχολείται με τον πιο διάσημο τρόπο αξιολόγησης ασφάλειας μιας εφαρμογής το penetration testing, στο δεύτερο αναλύεται η αρχιτεκτονική web των εφαρμογών και οι ευπάθειες στην ασφάλεια τους και στο τελευταίο εξηγείται τι είναι μια επίθεση SQL injection και περιγράφονται οι τύποι αυτής της επίθεσης. Στη συνέχεια ακολουθεί το πρακτικό κομμάτι το οποίο παρέχει πρακτικά παραδείγματα που δείχνουν τις διάφορες μορφές επιθέσεων SQL injection, όπως error-based injection, second-order injection και injection via cookies. Αναλύονται επίσης τρόποι αντιμετώπισης αυτών των επιθέσεων, όπως η χρήση των prepared statements. Ο συνδυασμός θεωρίας και πρακτικών παραδειγμάτων προσφέρει μια ολοκληρωμένη κατανόηση της ασφάλειας των web εφαρμογών, την αξιολόγηση της ασφάλειας τους σύμφωνα με τους κανόνες του pen testing και των τρόπων άμυνας κατά των επιθέσεων SQL injection.

## Λέξεις Κλειδιά.

Penetration Testing, Web Application Security, SQL Injection, SQL Injection Types

# Summary

Web applications have become a very important part of today's society, facilitating communication, online shopping and information sharing. However, with the increasing importance of web applications, significant security challenges also arise. This thesis focuses on the security of web applications, with the main theme being SQL injection attacks, one of the most well-known and common threats.

The thesis begins with the theoretical part, the first chapter is about penetration testing-a well-known way to assess the security of an application-, the second chapter describes the architecture of web applications and also analyzes their security vulnerabilities and the last chapter explains what an SQL injection attack is and describes its types. The practical part of the thesis provides practical examples showing the various forms of SQL injection attacks, such as error-based injection, second-order injection and injection via cookies. After every example there are shown ways to counter these attacks, such as the use of prepared statements. The combination of theory and practical examples offers a comprehensive understanding of the security of web applications, their security assessment according to the rules of pen testing and ways to defend against SQL injection attacks.

## Key Words.

Penetration Testing, Web Application Security, SQL Injection, SQL Injection Types

# Περιεχόμενα

<b>1</b>	<b>Εισαγωγή</b>	<b>5</b>
<b>2</b>	<b>Θεωρητικό Κομμάτι</b>	<b>7</b>
2.1	Penetration Testing . . . . .	7
2.1.1	Τι είναι το Penetration testing . . . . .	7
2.1.2	Τύποι Penetration testing . . . . .	8
2.1.3	Γνωστές Μεθοδολογίες του Penetration testing . . . . .	11
2.1.4	Εργαλεία του Penetration testing . . . . .	15
2.2	Web εφαρμογές και ασφάλεια . . . . .	17
2.2.1	Τι είναι μια web εφαρμογή . . . . .	17
2.2.2	Ευπάθειες web εφαρμογών . . . . .	20
2.3	SQL Injection . . . . .	22
2.3.1	Τι είναι μια επίθεση SQL injection . . . . .	22
2.3.2	Τύποι SQL injection . . . . .	23
<b>3</b>	<b>Προσομοίωση επιθέσεων SQL Injection</b>	<b>25</b>
3.1	SQL Injection Based on User Input . . . . .	26
3.2	Blind SQL Injection . . . . .	29
3.3	Error-Based SQL Injection . . . . .	31
3.4	Union-based SQL Injection . . . . .	33
3.5	Second-Order SQL Injection . . . . .	34
3.6	SQL Injection Based on HTTP Headers . . . . .	39
3.7	SQL Injection via Cookies . . . . .	41
<b>4</b>	<b>Επίλογος</b>	<b>44</b>

# ΚΕΦΑΛΑΙΟ 1

## Εισαγωγή

Στη σημερινή εποχή, το διαδίκτυο έχει γίνει απαραίτητο για πολλές βασικές λειτουργίες που χρησιμοποιούμε στην καθημερινή μας ζωή, με τις διαδικτυακές εφαρμογές να διαδραματίζουν κρίσιμο ρόλο σε οτιδήποτε, όπως τις εφαρμογές e-banking, τα μέσα κοινωνικής δικτύωσης και το ηλεκτρονικό εμπόριο. Καθώς οι οργανισμοί και οι περισσότεροι άνθρωποι βασίζονται όλο και περισσότερο σε αυτές τις εφαρμογές για να διεξάγουν τις δραστηριότητες αυτές και να διαχειρίζονται ευαίσθητα δεδομένα, η ανάγκη για την προστασία αυτών των εφαρμογών γίνεται πιο κρίσιμη από ποτέ. Η τεχνολογία συνεχώς εξελίσσεται, αλλά μαζί της εξελίσσονται και οι απειλές που τη στοχεύουν. Για αυτό τον λόγο ο χώρος του cybersecurity έχει πλέον αναδειχθεί σε ένα από τα πιο σημαντικά ζητήματα τόσο για τους προγραμματιστές όσο και για τις επιχειρήσεις και τους χρήστες.

Οι web εφαρμογές, από τη φύση τους, συχνά εκτίθενται στο κοινό, καθιστώντας τις πρωταρχικούς στόχους για κακόβουλους χρήστες. Αυτές οι εφαρμογές χειρίζονται συχνά μεγάλες ποσότητες ευαίσθητων δεδομένων και μπορούν να γίνουν στόχοι για εισβολείς που θέλουν να εκμεταλλευτούν αδυναμίες για κακόβουλους σκοπούς. Μεταξύ των διαφόρων μορφών επιθέσεων, η επίθεση SQL injection ξεχωρίζει ως μία από τις πιο διαδεδομένες και επιζήμιες. Η επίθεση αυτή περιλαμβάνει την εισαγωγή κακόβουλων SQL query μέσα στα πεδία εισόδου μιας web εφαρμογής, με αποτέλεσμα να διακινδυνεύει η βάση δεδομένων της εφαρμογής αυτής. Αυτός ο τύπος επίθεσης μπορεί να οδηγήσει σε μη εξουσιοδοτημένη πρόσβαση σε ευαίσθητες πληροφορίες, καταστροφή δεδομένων ή ακόμα και σε πλήρη παραβίαση του συστήματος.

Οι αυξανόμενες απειλές στην ασφάλεια των συστημάτων/εφαρμογών δημιούργησαν την ανάγκη για πιο ισχυρά μέτρα ασφάλειας, γεγονός που οδήγησε στην πρακτική των έλεγχων διείσδυσης (pen testing). Ο έλεγχος διείσδυσης είναι μια προληπτική προσέγγιση για τον εντοπισμό και την αντιμετώπιση των ευάλωτων σημείων ενός συστήματος προτού μπορέσουν να τα εκμεταλλευτούν κακόβουλοι χρήστες. Με την προσομοίωση σεναρίων επιθέσεων σε πραγματικό κόσμο, οι ελεγκτές (pen testers) μπορούν να αποκαλύψουν αδυναμίες στην άμυνα ενός συστήματος και να παρέχουν χρήσιμες πληροφορίες για τη βελτίωση της ασφάλειας του.

Το hacking , που συχνά συνδέεται με παράνομες και κακόβουλες δραστηριότητες, έχει εξελιχθεί σε μια πιο διαφοροποιημένη έννοια στον τομέα του cybersecurity. Ενώ το ανήθικο hacking στοχεύει στην παραβίαση συστημάτων για προσωπικό όφελος ή για να προκαλέσει βλάβη, το ηθικό hacking ή αλλιώς white hat hacking, χρησιμεύει ως κρίσιμο εργαλείο για τη διαφύλαξη των συστημάτων πληροφοριών. Οι ηθικοί hacker χρησιμοποιούν τις δεξιότητές τους για να εντοπίσουν και να διορθώσουν τα ευάλωτα σημεία, διασφαλίζοντας ότι οι οργανισμοί μπορούν να προστατεύσουν τα περιουσιακά τους στοιχεία από κακόβουλες επιθέσεις.

Καθώς οι web εφαρμογές συνεχίζουν να αυξάνονται σε πολυπλοκότητα και λειτουργικότητα, η ασφάλεια τους γίνεται πιο δύσκολη και απαιτητική. Οι προγραμματιστές πρέπει όχι μόνο να επικεντρωθούν στη δημιουργία μιας αποτελεσματικής εμπειρίας χρήστη, αλλά και να διασφαλίσουν ότι οι εφαρμογές τους είναι ανθεκτικές έναντι μιας ευρείας σειράς πιθανών απειλών. Η επίτευξη αυτής της ισορροπίας απαιτεί συνεχή εκπαίδευση, ενημέρωση και χρήση προηγμένων τεχνικών ασφαλείας, ώστε να προστατεύονται οι χρήστες και τα δεδομένα τους από τις συνεχώς εξελισσόμενες απειλές.



# ΚΕΦΑΛΑΙΟ 2

## Θεωρητικό Κομμάτι

### 2.1 Penetration Testing

#### 2.1.1 Τι είναι το Penetration testing

Ο έλεγχος διείσδυσης, γνωστός και ως penetration testing ή ethical hacking, είναι μια διαδικασία αξιολόγησης ασφάλειας στην οποία ένας ελεγκτής προσπαθεί συστηματικά να εντοπίσει και να εκμεταλλευτεί αδυναμίες που μπορεί να παρουσιάζονται σε ένα δίκτυο, μια εφαρμογή ιστού ή σε υπολογιστικό σύστημα. Στόχος αυτής της προσομοιωμένης επίθεσης είναι να αξιολογηθεί η αποτελεσματικότητα των μέτρων ασφάλειας του συστήματος και να εντοπιστούν τυχόν αδυναμίες που θα μπορούσαν ενδεχομένως να εκμεταλλευτούν κακόβουλοι χρήστες. Το penetration testing περιλαμβάνει διάφορα στάδια όπως αυτά του σχεδιασμού, της συλλογής πληροφοριών, της σάρωσης ευπάθειας, της εκμετάλλευσης και της αναφοράς.

Κατά τη φάση σχεδιασμού, καθορίζονται το εύρος και οι στόχοι της δοκιμής, διασφαλίζοντας ότι δεν παραβιάζουν τις πολιτικές ασφάλειας και τις απαιτήσεις συμμόρφωσης του οργανισμού. Η συλλογή πληροφοριών περιλαμβάνει τη συγκέντρωση όσων περισσότερων δεδομένων γίνεται για το σύστημα-στόχο, ώστε να εντοπιστούν πιθανά σημεία εισβολής. Η σάρωση ευπαθειών χρησιμοποιεί αυτοματοποιημένα εργαλεία για τον εντοπισμό γνωστών αδυναμιών, ενώ τεχνικές χειροκίνητων δοκιμών εφαρμόζονται για την αποκάλυψη πιο περίπλοκων ζητημάτων ασφαλείας που μπορεί να παραβλέψουν τα αυτοματοποιημένα εργαλεία.

Μόλις εντοπιστούν οι ευπάθειες, ο ελεγκτής επιχειρεί να τις εκμεταλλευτεί για να αποκτήσει μη εξουσιοδοτημένη πρόσβαση, να αποκτήσει παραπάνω προνόμια ή να υποκλέψει ευαίσθητα δεδομένα, προσομοιώνοντας τις ενέργειες των πραγματικών εισβολέων. Στη συνέχεια, τα αποτελέσματά και οι ευπάθειες που βρέθηκαν καταγράφονται σχολαστικά σε μια λεπτομερή αναφορά, η οποία παρέχει μια ολοκληρωμένη εικόνα των αδυναμιών που βρέθηκαν, των μεθόδων που χρησιμοποιήθηκαν για την εκμετάλλευσή τους και των πιθανών επιπτώσεων στον οργανισμό.

Η τελική φάση περιλαμβάνει την αποκατάσταση, όπου ο οργανισμός λαμβάνει συστάσεις που μπορούν να εφαρμοστούν για να διορθώσει τα ευάλωτα σημεία και να ενισχύσει τη συνολική του ασφάλεια. Οι τακτικοί έλεγχοι διείσδυσης είναι απαραίτητοι για τη διατήρηση της καλύτερης δυνατής ασφάλειας των συστημάτων και την προστασία ευαίσθητων δεδομένων έναντι των συνεχώς εξελισσόμενων απειλών από εργαλεία hacking αλλά και κακόβουλων χρηστών.

### 2.1.2 Τύποι Penetration testing

Υπάρχουν πολλοί τύποι penetration testing, ο καθένας από αυτούς κατηγοριοποιείται ανάλογα με την προσέγγιση του. Δηλαδή η κάθε κατηγορία εξαρτάται από τις πληροφορίες που έχει ο ελεγκτής για το σύστημα και από την τεχνική που ακολουθεί. Η κατηγοριοποίηση που ακολουθεί είναι σύμφωνα με το BSI (British Standards Institution)<sup>1</sup>.

#### 1. Τύποι penetration testing ανάλογα με το επίπεδο γνώσης του συστήματος από τον ελεγκτή

- **Black Box Testing :** Το pen testing τύπου black box περιλαμβάνει τον έλεγχο ενός συστήματος χωρίς καμία προηγούμενη γνώση γι' αυτό. Ο ελεγκτής ενεργεί όπως θα έκανε ένας εξωτερικός επιτιθέμενος, χωρίς πρόσβαση σε εσωτερικές πληροφορίες από το εσωτερικό του οργανισμού/εταιρείας. Αυτή η μέθοδος χρησιμοποιείται για να προσομοιώσει πραγματικά σενάρια επίθεσης και να κατανοήσει πώς μπορεί να επιτεθεί ένας εξωτερικός επιτιθέμενος την οργάνωση, ενώ εντοπίζει τυχόν ευπάθειες που μπορεί να εκμεταλλευτούν χωρίς εσωτερική γνώση. Αν και προσφέρει μια ρεαλιστική προσομοίωση εξωτερικής επίθεσης και βοηθάει στην αναγνώριση ευπαθειών που είναι δημόσια προσβάσιμες, έχει περιορισμένο εύρος, καθώς ο ελεγκτής δεν διαθέτει εσωτερική γνώση και μπορεί να μην αποκαλύψει βαθύτερες ευπάθειες μέσα στο εσωτερικό δίκτυο.
- **White Box Testing :** Το pen testing τύπου white box περιλαμβάνει τον έλεγχο ενός συστήματος με πλήρη γνώση γι' αυτό, όπως πρόσβαση στον πηγαίο κώδικα, στην αρχιτεκτονική δικτύου και σε άλλες εσωτερικές πληροφορίες. Αυτή η προσέγγιση επιτρέπει μια λεπτομερή εξέταση των πιθανών ευπαθειών. Για παράδειγμα, μια εταιρεία ανάπτυξης λογισμικού μπορεί να πραγματοποιήσει δοκιμή white box για να διασφαλίσει ότι ο κώδικάς της είναι ασφαλής και δεν κινδυνεύει από ευπάθειες πριν την κυκλοφορία μιας νέας εφαρμογής. Αυτή η μέθοδος χρησιμοποιείται για την παροχή μιας πλήρους εκτίμησης της ασφάλειας του συστήματος με πρόσβαση σε όλες τις σχετικές πληροφορίες και για τον εντοπισμό αλλά και την διόρθωση ευπαθειών που μπορεί να υπάρχουν κατά τη διάρκεια της διαδικασίας ανάπτυξης του συστήματος. Αν και προσφέρει μια εξαιρετικά λεπτομερή ανάλυση του συστήματος και βοηθάει στην ανίχνευση και αντιμετώπιση ευπαθειών νωρίς

---

<sup>1</sup><https://turingpoint.de/en/blog/classification-of-pentests-according-to-bsi/>

στη διαδικασία ανάπτυξης, απαιτεί σημαντικό χρόνο και πόρους και ενδέχεται να μην προσομοιώσει τα σενάρια πραγματικής επίθεσης όσο αποτελεσματικά όσο η τεχνική τύπου black box.

## 2. Τύποι penetration testing ανάλογα με την τεχνική

- **Network Pen Testing :** Το network pen testing περιλαμβάνει την αξιολόγηση της ασφάλειας της δικτυακής υποδομής ενός οργανισμού, συμπεριλαμβανομένων των δρομολογητών, των μεταγωγέων, των firewalls και άλλων συσκευών δικτύου. Ο βασικός στόχος είναι να εντοπιστούν ευάλωτα σημεία που θα μπορούσαν να εκμεταλλευτούν οι εισβολείς για να αποκτήσουν μη εξουσιοδοτημένη πρόσβαση ή να διακόψουν τις υπηρεσίες δικτύου.
- **Web Application Pen Testing :** web application pen testing επικεντρώνεται στον εντοπισμό ευάλωτων σημείων της ασφαλείας των web εφαρμογών. Ο ελεγκτής προσομοιώνει επιθέσεις οι οποίες θα φανέρωναν κάποιο σημείο που θα επέτρεπε sql injection, cross-site scripting (XSS) και "σπάσιμο" της αυθεντικοποίησης.
- **Mobile Apps Pen Testing :** Το mobile apps pen testing αξιολογεί την ασφάλεια των εφαρμογών που εκτελούνται σε κινητές συσκευές. Αυτή η δοκιμή περιλαμβάνει την εξέταση του πηγαίου κώδικα της εφαρμογής και της αποθήκευσης δεδομένων για ευπάθειες. Ο στόχος είναι να εντοπιστούν οι αδυναμίες που θα μπορούσε να εκμεταλλευτεί ένας επιτιθέμενος για να αποκτήσει πρόσβαση σε ευαίσθητα δεδομένα ή να χειραγωγήσει τη λειτουργικότητα της εφαρμογής.
- **Physical Pen Testing :** Το physical pen testing αξιολογεί τους φυσικούς ελέγχους ασφαλείας ενός οργανισμού, όπως συστήματα ελέγχου πρόσβασης και επιτήρηση (surveillance). Ο ελεγκτής προσπαθεί να παρακάμψει αυτούς τους ελέγχους για να αποκτήσει μη εξουσιοδοτημένη πρόσβαση σε εγκαταστάσεις και σε ευαίσθητα δεδομένα του συστήματος.
- **Social Engineering Pen Testing :** Το social engineering pen testing περιλαμβάνει προσομοίωση επιθέσεων που εκμεταλλεύονται την ανθρώπινη συμπεριφορά για να αποκτήσουν μη εξουσιοδοτημένη πρόσβαση σε πληροφορίες ή συστήματα. Ο ελεγκτής χρησιμοποιεί τεχνικές όπως phishing, pretexting και baiting για να εξαπατήσει τους υπαλλήλους ώστε να αποκαλύψουν ευαίσθητες πληροφορίες ή να εκτελέσουν ενέργειες που θέτουν σε κίνδυνο την ασφάλεια του συστήματος.

## 3. Τύποι penetration testing ανάλογα με την επιθετικότητα

- **Passive :** Το passive pen testing είναι μια μη παρεμβατική μέθοδος που συλλέγει πληροφορίες χωρίς να αλληλεπιδρά άμεσα με τα συστήματα-στόχους. Χρησιμοποιεί τεχνικές όπως OSINT, network sniffing και παθητική αναγνώριση για τη

συλλογή δεδομένων χωρίς την εκμετάλλευση ευάλωτων σημείων. Αυτή η προσέγγιση παρέχει μια βασική κατανόηση των πιθανών ευάλωτων σημείων με βάση τις παρατηρούμενες πληροφορίες. Είναι ιδιαίτερα χρήσιμο για αρχικές φάσεις αναγνώρισης αδυναμιών ή σε εξαιρετικά ευαίσθητα περιβάλλοντα όπου η ενεργή δοκιμή είναι περιορισμένη.

- **Cautious** :Το cautious pen testing δίνει προτεραιότητα στην ελαχιστοποίηση του κινδύνου για τα συστήματα-στόχους, χρησιμοποιώντας συντηρητικές τεχνικές και γνωστά exploits για την αποφυγή έντονων διαταραχών. Αυτή η προσέγγιση, η οποία συχνά περιορίζεται σε εύρος για τη διασφάλιση ασφάλειας, μπορεί να "χάσει" ορισμένες ευπάθειες. Είναι ιδανικό για περιβάλλοντα όπου ο χρόνος λειτουργίας και η σταθερότητα του συστήματος είναι ζωτικής σημασίας, όπως για παράδειγμα συστήματα υγειονομικής περίθαλψης.
- **Calculated** :Στο calculated pen testing ο ελεγκτής υπολογίζει πριν την προσομοίωση της επίθεσης το ποσοστό επιτυχίας της αλλά και τι συνέπειες θα έχει στο σύστημα-στόχο. Είναι κατάλληλο για οργανισμούς που αναζητούν βαθιά κατανόηση της ασφαλείας τους χωρίς σημαντικούς λειτουργικούς κινδύνους.
- **Aggressive** :Το aggressive pen testing προσομοιώνει τις επιθέσεις ώστε να είναι όσο το δυνατό πιο επικίνδυνες και ρεαλιστικές γίνεται. Χρησιμοποιεί προηγμένες και επιθετικές τεχνικές, με μεγάλη πιθανότητα να προκαλέσει διακοπές λειτουργίας του συστήματος ή καταστροφή δεδομένων, για να αποκαλύψει τα πιο κρίσιμα και ευάλωτα σημεία του συστήματος.

#### 4. Τύποι penetration testing ανάλογα με το εύρος

- **Full** :Το full pen testing είναι μια ολοκληρωμένη αξιολόγηση ολόκληρης της IT υποδομής ενός οργανισμού. Στόχος του είναι να ανακαλύψει όσο το δυνατόν περισσότερες ευπάθειες μπορεί να παρουσιάζονται σε δίκτυα, εφαρμογές, τελικά σημεία και μέτρα φυσικής ασφάλειας του οργανισμού. Αυτός ο τύπος pen testing προσομοιώνει μια πραγματική επίθεση με τον ελεγκτή να έχει πλήρη ελευθερία να εξερευνήσει και να εκμεταλλευτεί τυχόν αδυναμίες.
- **Limited** :Το limited pen testing εστιάζει σε συγκεκριμένα σημεία της IT υποδομής ενός οργανισμού, όπως συγκεκριμένες εφαρμογές, διακομιστές ή τμήματα δικτύου. Αυτή η στοχευμένη προσέγγιση στοχεύει στον εντοπισμό ευάλωτων σημείων εντός του καθορισμένου πεδίου εφαρμογής, καθιστώντας την κατάλληλη για την αντιμετώπιση γνωστών αδύναμων σημείων.
- **Focused** :Το focused pen testing στοχεύει πολύ συγκεκριμένα ευάλωτα σημεία ή σενάρια απειλών για να ελέγξει την ανθεκτικότητα συγκεκριμένων μέτρων ασφαλείας. Συχνά ο ελεγκτής είναι καθοδηγούμενος από ήδη γνωστά ζητήματα ή πρόσφατες πληροφορίες απειλών, Αυτή η συγκεκριμένη προσέγγιση επιβεβαι-

ώνει πόσο καλά λειτουργούν συγκεκριμένες άμυνες ή εξετάζει τις επιπτώσεις από συγκεκριμένα ευάλωτα σημεία.

## 5. Τύποι penetration testing ανάλογα με την προσέγγιση

- **Covert** : Σε ένα covert pen testing γνωστό και ως double-blind pen testing η επίθεση στο σύστημα γίνεται από κάποιο ελεγκτή χωρίς να έχουν ενημερωθεί οι υπάλληλοι υπεύθυνοι για την ασφάλεια του συστήματος. Αυτή η προσέγγιση στοχεύει στην προσομοίωση ενός πραγματικού σεναρίου επίθεσης, όπου ο ελεγκτής συμπεριφέρεται σαν αληθινός εισβολέας δηλαδή δεν έχει προηγούμενη γνώση της εσωτερικής υποδομής του συστήματος και πρέπει να βασίζεται σε τεχνικές εξωτερικής αναγνώρισης και εκμετάλλευσης.
- **Overt** : Το overt pen testing περιλαμβάνει τη διεξαγωγή των επιθέσεων με την πλήρη γνώση και συνεργασία της ομάδας των υπαλλήλων υπεύθυνους για την ασφάλεια του οργανισμού. Ο ελεγκτής παρέχεται με λεπτομερείς πληροφορίες σχετικά με εσωτερικά συστήματα, δίκτυα και εφαρμογές.

## 6. Τύποι penetration testing ανάλογα με το σημείο εκκίνησης

- **Internal** : Ο έλεγχος τύπου internal testing περιλαμβάνει την προσομοίωση επιθέσεων από το εσωτερικό του δικτύου του οργανισμού. Αυτός ο τύπος pen testing είναι απαραίτητος για την κατανόηση του τρόπου με τον οποίο ένας εσωτερικός χρήστης, όπως ένας υπάλληλος ή κάποιος που έχει καταφέρει να παραβιάσει την εξωτερική ασφάλεια του δικτύου, θα μπορούσε να εκμεταλλευτεί τυχόν ευπάθειες που υπάρχουν στο σύστημα. Για παράδειγμα, μια βιομηχανική επιχείρηση μπορεί να πραγματοποιήσει internal pen testing για να διασφαλίσει ότι οι υπάλληλοι δεν μπορούν εύκολα να αποκτήσουν πρόσβαση σε ευαίσθητα τμήματα του δικτύου ή να διαταράξουν με κάποιο τρόπο το σύστημα.
- **External** : Ο έλεγχος τύπου external testing επικεντρώνεται στην αξιολόγηση της ασφάλειας των εξωτερικών πόρων ενός οργανισμού, όπως διακομιστές, firewalls και άλλες υποδομές δικτύου που είναι εκτεθειμένες στο διαδίκτυο. Αυτού του είδους η δοκιμή είναι κρίσιμη για την αναγνώριση ευπαθειών που θα μπορούσαν να εκμεταλλευτούν επιτιθέμενοι εκτός του εσωτερικού δικτύου του οργανισμού. Για παράδειγμα, μια εταιρεία που προσφέρει υπηρεσίες online banking θα επέλεγε external pen testing για να διασφαλίσει ότι η πλατφόρμα της, οι διακομιστές web και η σχετική υποδομή είναι ασφαλείς από επιθέσεις όπως DDoS, SQL injection και μη εξουσιοδοτημένη πρόσβαση.

### 2.1.3 Γνωστές Μεθοδολογίες του Penetration testing

Καθώς το penetration testing είναι πολύ σημαντικό για την ασφάλεια συστημάτων είναι απαραίτητο να υπάρχει κάποια μεθοδολογία για την διεξαγωγή του. Γνωστές μεθοδολογίες

οι οποίες παρέχουν λεπτομερείς οδηγίες και βέλτιστες πρακτικές για τη διεξαγωγή ολοκληρωμένων pen test είναι οι OWASP Testing Guide (OTG), Penetration Testing Execution Standard (PTES), NIST SP 800-115 και OSSTMM (Open Source Security Testing Methodology Manual). Παρακάτω θα αναλυθούν οι φάσεις από κάποιες από αυτές τις μεθοδολογίες.

Η μεθοδολογία Penetration Testing Execution Standard (PTES) παρέχει ένα ολοκληρωμένο πλαίσιο για τη διεξαγωγή pen testing, διασφαλίζοντας ότι όλα τα βήματα της διαδικασίας εκτελούνται μεθοδικά και σωστά. Αυτή η ενότητα θα αναλύσει τις επτά διαφορετικές φάσεις του PTES, καθεμία από τις οποίες είναι απαραίτητη για τον εντοπισμό και την αντιμετώπιση των αδυναμιών ασφαλείας στην υποδομή ενός οργανισμού.

### **1. Pre-Engagement**

Η αρχική φάση περιλαμβάνει την προετοιμασία και τον σχεδιασμό της επίθεσης που πρόκειται να προσομοιωθεί. Οι ελεγκτές συγκεντρώνουν τα κατάλληλα εργαλεία, λειτουργικά συστήματα και λογισμικό που χρειάζονται, ανάλογα με το είδος και την έκταση του pen test.

### **2. Information Gathering**

Σε αυτή τη φάση, ο ελεγκτής συγκεντρώνει πληροφορίες για τα συστήματα-στόχους, τόσο από τον οργανισμό όσο και από δημόσιες πηγές, όπως τα μέσα κοινωνικής δικτύωσης. Αυτή η συλλογή πληροφοριών είναι σημαντική για τα network pen testing καθώς βοηθά στον εντοπισμό πιθανών αδυναμιών και σημείων εισόδου.

### **3. Threat Modelling**

Η μοντελοποίηση απειλών (threat modelling) είναι μια συστηματική προσέγγιση για τον εντοπισμό και την αντιμετώπιση πιθανών απειλών που στοχεύουν ένα σύστημα ή μια εφαρμογή ενός οργανισμού. Αυτή η φάση περιλαμβάνει την αξιολόγηση των δυνατοτήτων των πιθανών εισβολέων και τον σχεδιασμό της κατάλληλης ασφάλειας για την προστασία των πιο ευάλωτων τμημάτων του οργανισμού.

### **4. Vulnerability Analysis**

Σε αυτή την φάση οι ελεγκτές εντοπίζουν και αξιολογούν τα ευάλωτα σημεία που προκύπτουν από ελαττώματα/αδυναμίες στην ασφάλεια των συστημάτων του οργανισμού. Χρησιμοποιούν έναν συνδυασμό αυτοματοποιημένων εργαλείων και χειροκίνητων μεθόδων για να αποκαλύψουν αδυναμίες που θα μπορούσαν να εκμεταλλευτούν κακόβουλοι χρήστες. Ο στόχος είναι να εντοπιστούν τα πιο επικίνδυνα ευάλωτα σημεία και να δοθεί στον οργανισμό μια ξεκάθαρη εικόνα των περιοχών που χρειάζονται άμεση προσοχή και διόρθωση.

### **5. Exploitation**

Σε αυτή την φάση οι ελεγκτές δοκιμάζουν να εκμεταλλευτούν τα ευάλωτα σημεία που βρέθηκαν στις προηγούμενες φάσεις για να δουν αν μπορούν να χρησιμοποιηθούν

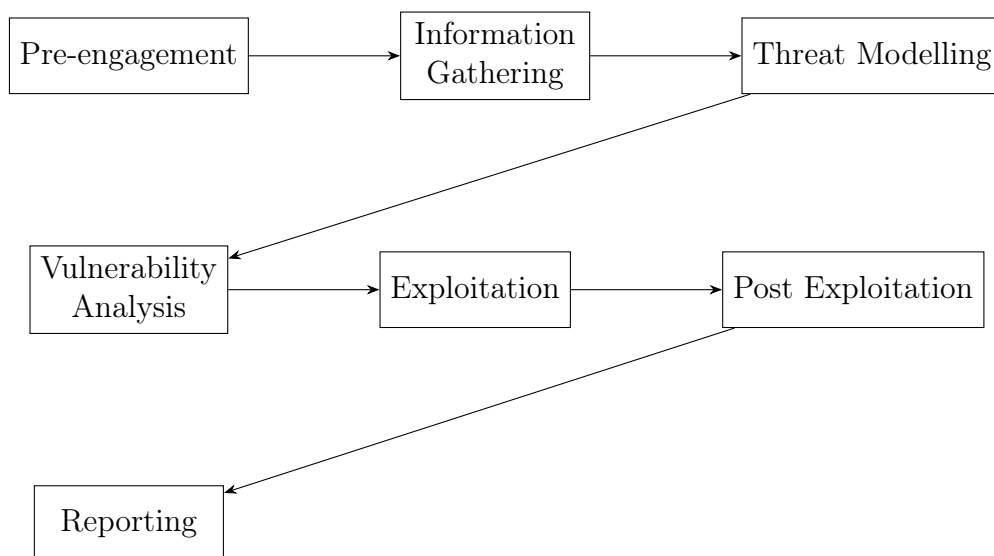
ώστε να παραβιαστεί η ασφάλεια του οργανισμού. Αυτό περιλαμβάνει την προσομοίωση πραγματικών επιθέσεων για να ελέγξουν αν οι αδυναμίες που εντοπίστηκαν μπορούν να αξιοποιηθούν. Η επιτυχία αυτής της φάσης εξαρτάται από την ακρίβεια της προηγούμενης ανάλυσης και δείχνει τα αποτελέσματα που θα είχε μια πραγματική επίθεση.

## 6. Post Exploitation

Μετά την επιτυχή εκμετάλλευση των αδυναμιών, ο ελεγκτής αξιολογεί το επίπεδο πρόσβασης που απέκτησε και την αξία των παραβιασμένων συστημάτων. Αυτή η φάση δείχνει πόσο μακριά θα μπορούσε να φτάσει ένας εισβολέας μέσα στο δίκτυο και ποιος θα ήταν ο αντίκτυπος στη συνολική ασφάλεια του οργανισμού. Ο ελεγκτής επίσης διορθώνει τυχόν αλλαγές που έγιναν κατά τη διάρκεια της δοκιμής για να διασφαλίσει ότι δεν θα μείνουν πίσω ανεπιθύμητες επιπτώσεις.

## 7. Reporting

Η τελική φάση περιλαμβάνει τη συγκέντρωση των αδυναμιών που βρέθηκαν και τις επιπτώσεις που θα είχαν αν συνέβαιναν σε μια ολοκληρωμένη αναφορά. Στην αναφορά αυτή σημειώνεται τι δοκιμάστηκε, ποιες μέθοδοι χρησιμοποιήθηκαν, τα ευάλωτα σημεία που βρέθηκαν και τα βήματα που έγιναν για την εκμετάλλευσή τους. Παρέχει επίσης προτάσεις για τη μείωση των κινδύνων, βοηθώντας τον οργανισμό να βελτιώσει την ασφάλειά του.



**Diagram 1 : Penetration Testing Execution Standard (PTES) Phases**

Το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST)<sup>2</sup> παρέχει μια δομημένη προσέγγιση για την διεξαγωγή pen testing, σχεδιασμένη να βοηθά τους οργανισμούς να αξιο-

<sup>2</sup>Elain Barker, Technical Report NIST Special Publication 800-57 Part 1

λογούν την ασφάλεια των συστημάτων τους. Η μεθοδολογία NIST χωρίζεται σε τέσσερις βασικές φάσεις: Planning, Discovery, Attack και Reporting. Θα αναλυθούν παρακάτω.

### 1. Planning

Η πρώτη φάση είναι το θεμελιώδες βήμα της μεθοδολογίας NIST. Σε αυτή τη φάση, η ομάδα που θα κάνει τον έλεγχο και ο οργανισμός συμφωνούν για το τι ακριβώς θα δοκιμαστεί, όπως ποια συστήματα, δίκτυα και εφαρμογές θα ελεγχθούν, και ποια όρια θα πρέπει να τηρηθούν. Επίσης, καθορίζονται και άλλοι παράμετροι όπως ο χρόνος που θα διαρκέσει η δοκιμή, οι πιθανοί κίνδυνοι που μπορεί να εμφανιστούν και τα εργαλεία ή το προσωπικό που θα χρειαστεί.

### 2. Discovery

Στην φάση αυτή οι ελεγκτές επικεντρώνεται στη συλλογή όσο το δυνατόν περισσότερων πληροφοριών σχετικά με το σύστημα-στόχο. Η συλλογή πληροφοριών γίνεται τόσο με παθητικές τεχνικές, όπως τον έλεγχο ήδη δημόσιων πληροφοριών και σάρωση δικτύου, όσο και με ενεργητικές τεχνικές, όπως συστήματα ανίχνευσης για ανοιχτές θύρες, υπηρεσίες και άλλα εκμεταλλεύσιμα σημεία εισόδου.

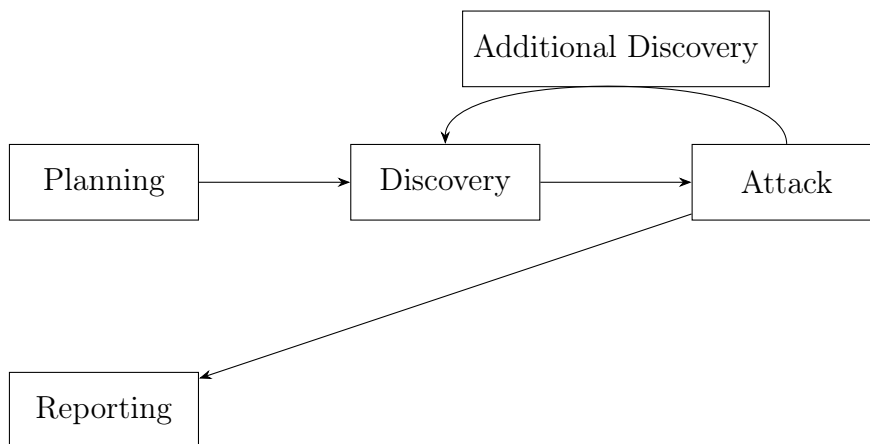
### 3. Attack

Στη φάση της επίθεσης, ο ελεγκτής χρησιμοποιεί τις πληροφορίες που συγκεντρώθηκαν κατά τη διάρκεια του Discovery για να προσπαθήσει να εκμεταλλευτεί τα ευάλωτα σημεία που βρέθηκαν. Αυτή η φάση προσομοιώνει τις ενέργειες ενός πραγματικού εισβολέα, χρησιμοποιώντας διάφορα εργαλεία και τεχνικές για να παραβιάσει τις άμυνες του οργανισμού. Ο στόχος είναι να αποκτήσει μη εξουσιοδοτημένη πρόσβαση σε συστήματα, δεδομένα ή δίκτυα για να δείξει τον πιθανό αντίκτυπο μιας επιτυχημένης επίθεσης. Η φάση αυτή είναι συχνά το πιο σημαντικό μέρος ενός pen test, καθώς παρέχει αποδείξεις για τα κενά ασφαλείας και την πιθανή εκμετάλλευσή τους από κάποιον κακόβουλο χρήστη.

### 4. Reporting

Στην τελευταία φάση συντάσσεται μια αναλυτική αναφορά με τις αδυναμίες που ανακαλύφθηκαν και τα αποτελέσματά της εκμετάλλευσής τους. Η αναφορά περιλαμβάνει συνήθως μια περίληψη για τα ανώτερα του οργανισμού, που περιγράφει τη συνολική κατάσταση της ασφαλείας και τα βασικά ευάλωτα σημεία, καθώς και μια πιο τεχνική ενότητα που παρέχει σε βάθος ανάλυση των συγκεκριμένων ευρημάτων. Επίσης περιέχει και προτάσεις που θα βοηθήσουν στον μετριασμό των κινδύνων που εντοπίστηκαν, βοηθώντας τον οργανισμό να βελτιώσει την ασφάλεια των συστημάτων του.





**Diagram 2 : NIST SP 800-115 Phases**

Η μεθοδολογία OWASP (OTG) προσφέρει έναν οργανωμένο τρόπο διεξαγωγής pen testing , περιλαμβάνοντας βασικά στάδια όπως συλλογή πληροφοριών, δοκιμές ρυθμίσεων και αξιολόγηση ευπαθειών. Στόχος της είναι να εντοπίσει και να μειώσει πιθανούς κινδύνους ασφαλείας σε κάθε φάση της ανάπτυξης λογισμικού. Τέλος η μεθοδολογία Open Source Security Testing Methodology Manual (OSSTMM) παρέχει μια πιο ευρεία προσέγγιση, καλύπτοντας όχι μόνο web εφαρμογές , αλλά και άλλους τομείς της ασφάλειας, όπως τη φυσική και δικτυακή ασφάλεια. Δημιουργήθηκε από το Institute for Security and Open Methodologies (ISECOM) και δίνει έμφαση σε μια αυστηρή διαδικασία ελέγχου με λεπτομερείς οδηγίες για την αξιολόγηση της ασφάλειας σε διάφορους τομείς. Και οι δύο μεθοδολογίες είναι σημαντικές για την ενίσχυση της ασφάλειας ενός οργανισμού, καθεμία προσφέροντας μοναδικά πλεονεκτήματα στη διεξαγωγή pen testing.

#### **2.1.4 Εργαλεία του Penetration testing**

Όπως αναφέρθηκε και στα παραπάνω κεφάλαια οι ελεγκτές που διεξάγουν τα penetration test χρησιμοποιούν εκτός από μεθοδολογίες διάφορα εργαλεία για να αυτοματοποιήσουν διαδικασίες όπως την εύρεση ευάλωτων σημείων ή ακόμα και την εκμετάλλευσή τους. Κάποια από τα πιο γνωστά εργαλεία είναι το NMap , Wireshark , OWASP ZAP , John the Ripper , Metasploit ,SQLmap και Burp Suite. Τα περισσότερα από αυτά είναι open-source και σε κάποιες διανομές Linux όπως τα KALI και Parrot OS είναι προ εγκατεστημένα.

#### **Εργαλεία για την σάρωση πληροφοριών**

Πρόκειται για εργαλεία που χρησιμοποιούνται στην αρχική φάση ενός pen test όπου ελεγκτής συγκετρώνει πληροφορίες για το σύστημα. Η χρήση εργαλείων σε αυτή την φάση είναι ιδιαίτερα χρήσιμη καθώς βοηθούν τον ελεγκτή να κατανοήσει καλύτερα την τοπολογία του δικτύου, τις υπηρεσίες που εκτελούνται σε ένα σύστημα και να εντοπίσει πιθανές εισόδους στο σύστημα. Το NMap είναι ένα ευρέως χρησιμοποιούμενο εργαλείο σάρωσης δικτύου που

βοηθά στην ανακάλυψη κεντρικών υπολογιστών, υπηρεσιών και ανοιχτών θυρών, παρέχοντας πληροφορίες για τα λειτουργικά συστήματα και τις εκδόσεις υπηρεσιών που εκτελούνται στο δίκτυο. Ένα ακόμα χρήσιμο εργαλείο το Maltego χρησιμοποιείται για εξόρυξη δεδομένων η οποία βοηθά στην οπτικοποίηση των σχέσεων μεταξύ σημείων δεδομένων, καθιστώντας το ιδιαίτερα χρήσιμο για τη συλλογή πληροφοριών ανοιχτού κώδικα (OSINT). Επιπλέον, το Google Dorking είναι μια τεχνική που χρησιμοποιεί ειδικούς τελεστές αναζήτησης στο Google για να βρει πληροφορίες που δεν είναι άμεσα εμφανείς σε ιστότοπους αλλά έχουν καταχωρηθεί από τις μηχανές αναζήτησης.

### **Εργαλεία για σάρωση ευπαθειών**

Στις αρχικές φάσεις ενός pen test ο ελεγκτής χρησιμοποιεί εργαλεία για να εντοπίσει ευάλωτα σημεία σε ένα σύστημα. Γνωστό εργαλείο για σάρωση ευπαθειών ικανό να εντοπίζει ένα μεγάλο εύρος ζητημάτων ασφαλείας, όπως γνωστές ευπάθειες, λανθασμένες ρυθμίσεις και μη ενημερωμένο λογισμικό, είναι το Nessus. Το OpenVAS είναι μια εναλλακτική λύση του Nessus καθώς παρέχει παρόμοια λειτουργικότητα για τον εντοπισμό αδυναμιών ασφάλειας και είναι open-source.

### **Εργαλεία για εκμετάλλευση ευάλωτων σημείων**

Η φάση του exploitation είναι από τις πιο κρίσιμες φάσεις ενός pen test αφού από τα αποτελέσματα αυτής της φάσης θα κριθεί πόσο σημαντική είναι η ευπάθεια που βρέθηκε και πως θα επηρεαστεί το σύστημα σε περίπτωση που κάποιος την εκμεταλλευτεί. Ένα από τα πιο σημαντικά εργαλεία σε αυτή τη φάση είναι το Metasploit Framework, το οποίο έχει την ικανότητά να προσομοιώνει πολύπλοκες επιθέσεις σε διάφορες πλατφόρμες. Επιτρέπει στους ελεγκτές να εκμεταλλεύονται συστηματικά τα ευάλωτα σημεία, παρέχοντας πολύτιμες πληροφορίες για τη κατάσταση ασφαλείας του στόχου. Άλλο ένα διαδεδομένο εργαλείο είναι το SQLmap, ειδικά σχεδιασμένο για να αυτοματοποιεί τον εντοπισμό και την εκμετάλλευση ευπαθειών τύπου SQL injection σε web εφαρμογές.

### **Εργαλεία για web εφαρμογές**

Για τη διεξαγωγή web application pen tests, υπάρχουν εργαλεία που διευκολύνουν τη διαδικασία και την καθιστούν πιο αποτελεσματική. Δύο από τα πιο ευρέως χρησιμοποιούμενα είναι το Burp Suite και το OWASP ZAP (Zed Attack Proxy). Το Burp Suite προσφέρει λειτουργίες όπως proxy server, σαρωτή ευπαθειών και επαναλήπτη (repeater), επιτρέποντας στους ελεγκτές να παρακολουθούν, να τροποποιούν και να αναλύουν την κυκλοφορία ιστού, να εντοπίζουν ευάλωτα σημεία και να αυτοματοποιούν επαναλαμβανόμενες εργασίες. Από την άλλη, το OWASP ZAP είναι μια open-source εναλλακτική λύση που επίσης παρέχει ισχυρές δυνατότητες σάρωσης για την ανίχνευση ευπαθειών σε web εφαρμογές. Είναι φιλικό προς το χρήστη και υποστηρίζει τόσο αυτοματοποιημένες όσο και μη αυτόματες δοκιμές, καθιστώντας το ιδανικό για αρχάριους και έμπειρους pen testers.

## Εργαλεία για σπάσιμο κωδικών πρόσβασης

Το σπάσιμο κωδικού πρόσβασης είναι ένας από τους βασικότερους τρόπους να αξιολογηθεί η ασφάλεια ενός συστήματος. Ο ελεγκτής επιχειρεί να αποκτήσει πρόσβαση ,για παράδειγμα σε λογαριασμό κάποιου άλλου χρήση χωρίς την έγκρισή του, αποκωδικοποιώντας τον κατακερματισμένο κωδικό πρόσβασης. Δύο κορυφαία εργαλεία σε αυτόν τον τομέα είναι το John the Ripper και το Hashcat. Το John the Ripper είναι γνωστό για την ταχύτητα και την ευελιξία που προσφέρει, καθώς λειτουργεί με πολλούς διαφορετικούς τύπους αλγορίθμων κρυπτογράφησης και κατακερματισμού. Χρησιμοποιεί μεθόδους όπως dictionary attacks και brute force attacks για να σπάσει αποτελεσματικά τους κωδικούς πρόσβασης. Το Hashcat, από την άλλη, ξεχωρίζει επειδή για να επιταχύνει τη διαδικασία χρησιμοποιεί την GPU. Αυτό το καθιστά ιδιαίτερα χρήσιμο εργαλείο σε περιπτώσεις περίπλοκων αλγορίθμων κατακερματισμού που απαιτούν πολλούς πόρους. Υποστηρίζει επίσης πολλούς τύπους κατακερματισμών και προσφέρει προηγμένες μεθόδους επίθεσης, όπως hybrid attacks και rule-based attacks, καθιστώντας το πολύ αποτελεσματικό στην ανάκτηση κωδικών πρόσβασης από διάφορα κρυπτογραφημένα δεδομένα.

## 2.2 Web εφαρμογές και ασφάλεια

### 2.2.1 Τι είναι μια web εφαρμογή

Μια web εφαρμογή ή αλλιώς διαδικτυακή εφαρμογή είναι μια εφαρμογή λογισμικού που εκτελείται σε έναν web server και είναι προσβάσιμη μέσω web browser. Σε αντίθεση με τις παραδοσιακές εφαρμογές ,οι οποίες είναι εγκατεστημένες στον υπολογιστή ή σε κάποια κινητή συσκευή, οι web εφαρμογές είναι προσβάσιμες από οποιαδήποτε συσκευή με σύνδεση στο διαδίκτυο και συμβατό πρόγραμμα περιήγησης(web browser).

Οι web εφαρμογές μπορεί να ποικίλλουν από απλούς ιστότοπους με διαδραστικές λειτουργίες, όπως φόρμες επικοινωνίας, έως πολύπλοκες πλατφόρμες όπως ιστότοποι μέσων κοινωνικής δικτύωσης ή συστήματα e-banking. Χρησιμοποιούν έναν συνδυασμό τεχνολογιών front-end (user interface) και back-end (server-side).

### Πως λειτουργεί μια web εφαρμογή

Οι web εφαρμογές αποτελούνται από δύο δομικά μέρη, το client και το server. Ο client ,που είναι συνήθως ένας web browser, αλληλεπιδρά με τον server για την εκτέλεση ενεργειών στην εφαρμογή. Το μέρος του client είναι υπεύθυνο για το user interface μιας εφαρμογής, δηλαδή είναι το κομμάτι με το οποίο αλληλεπιδρά ένας χρήστης. Όταν για παράδειγμα ένας χρήστης κάνει κλικ σε ένα κουμπί ή υποβάλλει μια φόρμα, το front-end(client) στέλνει ένα αίτημα στο back-end(server), με τη σειρά του ο server επεξεργάζεται αυτά τα αιτήματα, εκτελεί την απαραίτητη λογική, ανακτά ή ενημερώνει δεδομένα από μια βάση δεδομένων και στη συνέχεια στέλνει μια απάντηση στο front-end. Αυτή η απάντηση μπορεί να είναι οτιδήποτε,

από τη φόρτωση μιας νέας σελίδας έως την ενημέρωση ενός μέρους της ιστοσελίδας χωρίς πλήρη επαναφόρτωση.

Καθώς οι web εφαρμογές έχουν ως σκοπό την καλύτερη εμπειρία για τον χρήστη πρέπει να βρουν έναν τρόπο ώστε να διαχειρίζονται όσο πιο αποδοτικά γίνεται τις περιόδους σύνδεσης και την αποθήκευσή δεδομένων/προτιμήσεων ενός χρήστη. Εδώ μπαίνουν στο "παιχνίδι" τα cookies. Τα cookies διαδραματίζουν ιδιαίτερα σημαντικό ρόλο στη διαχείριση περιόδων σύνδεσης και στην αποθήκευση δεδομένων από το μέρος του client. Ένα cookie είναι ένα μικρό κομμάτι δεδομένων που στέλνει ο server στο browser του client, το οποίο στη συνέχεια αποθηκεύεται τοπικά στη συσκευή του χρήστη. Τα cookies χρησιμοποιούνται για διάφορους σκοπούς, όπως την απομνημόνευση των προτιμήσεων των χρηστών, την διατήρηση της σύνδεσης των χρηστών στην εφαρμογή ή η παρακολούθηση της συμπεριφοράς των χρηστών σε πολλές περιόδους σύνδεσης. Για παράδειγμα, όταν συνδέεστε σε έναν ιστότοπο και επιλέγετε την επιλογή "Remember me", ρυθμίζεται ένα cookie για να σας κρατά συνδεδεμένους ακόμα και αφού κλείσετε το browser σας. Τα cookies μπορούν επίσης να χρησιμοποιηθούν για αναλυτικά στοιχεία και διαφημίσεις, βοηθώντας τις web εφαρμογές να προσφέρουν εξατομικευμένες εμπειρίες με βάση τις παλαιότερες προτιμήσεις των χρηστών.

Οι βάσεις δεδομένων αποτελούν ένα από τα σημαντικότερα μέρη μιας web εφαρμογής καθώς προσφέρουν ένα δομημένο τρόπο αποθήκευσης, ανάκτησης και διαχείρισης δεδομένων. Όταν ένας χρήστης αλληλεπιδρά με μια εφαρμογή—υποβάλλοντας μια φόρμα, πραγματοποιώντας μια αγορά ή δημοσιεύοντας ένα σχόλιο—τα δεδομένα που δημιουργούνται αποθηκεύονται σε μια βάση δεδομένων. Οι βάσεις δεδομένων μπορούν να ταξινομηθούν σε τύπους SQL (Structured Query Language) και NoSQL. Οι βάσεις δεδομένων SQL, όπως η MySQL και η PostgreSQL, οργανώνουν δεδομένα σε πίνακες με προκαθορισμένα σχήματα, καθιστώντας τις ιδανικές για εφαρμογές που απαιτούν πολύπλοκα queries και σχέσεις μεταξύ των δεδομένων. Στην ουσία αυτές οι βάσεις δεδομένων διασφαλίζουν την ακεραιότητα της αποθήκευσης των δεδομένων μέσω δομημένων σχημάτων και σχεσιακών μοντέλων.

Οι βάσεις δεδομένων NoSQL, όπως η MongoDB, προσφέρουν μια πιο ευέλικτη προσέγγιση, αποθηκεύοντας δεδομένα σε μορφές όπως έγγραφα, ζεύγη key-values ή γραφήματα, καθιστώντας τις κατάλληλες για εφαρμογές που χειρίζονται μεγάλους όγκους μη δομημένων ή ημιδομημένων δεδομένων. Οι βάσεις δεδομένων NoSQL χρησιμοποιούνται συχνά σε καταστάσεις όπου η επεκτασιμότητα και η απόδοση είναι κρίσιμες, όπως σε αναλύσεις σε πραγματικό χρόνο, συστήματα διαχείρισης περιεχομένου και εφαρμογές μεγάλων δεδομένων. Επιτρέπουν πιο ευέλικτη ανάπτυξη και μπορούν να χειριστούν διαφορετικούς τύπους δεδομένων πιο αποτελεσματικά από τις παραδοσιακές βάσεις δεδομένων SQL.

## **Τεχνολογίες που χρησιμοποιούνται σε μια web εφαρμογή**

Οι web εφαρμογές βασίζονται σε μια ποικιλία τεχνολογιών για να λειτουργούν αποτελεσματικά και να είναι εύκολες στην χρήση από τον καθένα. Η ανάπτυξη τους βασίζεται σε ορισμένες βασικές γλώσσες προγραμματισμού, καθεμία από τις οποίες έχει έναν συγκεκριμένο ρόλο στη δημιουργία τους. Η HTML (HyperText Markup Language) είναι η βασική γλώσσα που

χρησιμοποιείται για τη δομή μιας web εφαρμογής, ορίζοντας τα βασικά δομικά στοιχεία μιας ιστοσελίδας, όπως επικεφαλίδες, συνδέσμους, εικόνες και πολυμέσα. Η HTML5, η τελευταία έκδοση της HTML, έχει βελτιώσει σημαντικά την ανάπτυξη ιστοσελίδων, εισάγοντας νέα σημασιολογικά στοιχεία, εργαλεία για την ενσωμάτωση πολυμέσων και API που επιτρέπουν τη δημιουργία πιο δυναμικών και εύχρηστων εφαρμογών.

Εκτός από την HTML, η CSS (Cascading Style Sheets) χρησιμοποιείται για το στυλ και την οπτική μορφοποίηση μιας web εφαρμογής, επιτρέποντας στους προγραμματιστές να ελέγχουν τη διάταξη, τα χρώματα, τις γραμματοσειρές και τη συνολική εμφάνιση μιας ιστοσελίδας. Η CSS επιτρέπει τον διαχωρισμό του περιεχομένου από το σχεδιασμό, καθιστώντας ευκολότερη τη διατήρηση και ενημέρωση των οπτικών πτυχών ενός ιστότοπου χωρίς να αλλοιώνεται η υποκείμενη δομή του. Οι προηγμένες λειτουργίες όπως το CSS Grid και το Flexbox έχουν δώσει περαιτέρω τη δυνατότητα στους προγραμματιστές να δημιουργούν σύνθετες διατάξεις που προσαρμόζονται σε διαφορετικά μεγέθη οθόνης ανάλογα με την συσκευή, διασφαλίζοντας ότι η ιστοσελίδα θα λειτουργεί ομαλά σε H/Υ, tablet και smartphone.

Άλλη μια γλώσσα προγραμματισμού που χρησιμοποιείται στην πλειοψηφία των web εφαρμογών, κυρίως για το back-end κομμάτι, είναι η JavaScript. Η JavaScript επιτρέπει στους προγραμματιστές να δημιουργούν λειτουργίες όπως φόρμες επικύρωσης, animations, διαδραστικούς χάρτες και real-time ενημέρωση της εφαρμογής, βελτιώνοντας σημαντικά την εμπειρία χρήστη. Οι δυνατότητες της JavaScript επεκτείνονται περαιτέρω μέσω βιβλιοθηκών και φραμεворκς όπως τα jQuery, React, Angular και Vue.js, τα οποία απλοποιούν τη διαδικασία δημιουργίας πολύπλοκων user interfaces.

Στην πλευρά του server χρησιμοποιούνται επίσης γλώσσες όπως PHP, Python, Ruby και Java για να δημιουργούν τη λογική και τη λειτουργικότητα που απαιτούνται για τις web εφαρμογές. Αυτές οι γλώσσες βοηθούν στην επεξεργασία δεδομένων, τη διαχείριση βάσεων δεδομένων και την εκτέλεση πολύπλοκων λειτουργιών. Για παράδειγμα, η PHP χρησιμοποιείται συχνά για την ανάπτυξη ιστοσελίδων, σε συστήματα όπως το WordPress. Η Python, γνωστή για την ευκολία στη χρήση της, χρησιμοποιείται σε frameworks όπως το Django και το Flask για την κατασκευή εφαρμογών. Η Ruby, με το framework Rails, είναι δημοφιλής για την ταχύτητα ανάπτυξης web εφαρμογών, ενώ η Java, με την ευρεία χρήση της σε εταιρικές εφαρμογές, είναι γνωστή για την απόδοση και την ασφάλειά της.

Τέλος ένα ιδιαίτερα σημαντικό χαρακτηριστικό των σύγχρονων web εφαρμογών είναι η δυνατότητά τους να επικοινωνούν και να συνδέονται με άλλες υπηρεσίες λογισμικού. Αυτό γίνεται μέσω των Application Programming Interfaces (APIs). Τα API επιτρέπουν στις εφαρμογές να στέλνουν και να λαμβάνουν δεδομένα από άλλες υπηρεσίες, διευκολύνοντας πολλές λειτουργίες. Για παράδειγμα, μια ιστοσελίδα μπορεί να χρησιμοποιεί API όπως το Stripe ή το PayPal για να διαχειρίζεται με ασφάλεια τις πληρωμές. Επίσης, είναι συχνό να ενσωματώνονται API από πλατφόρμες κοινωνικής δικτύωσης, όπως το Facebook ή το Twitter, για να επιτρέπεται στους χρήστες να συνδέονται, για παράδειγμα μέσω Facebook, ή να μοιράζονται περιεχόμενο μέσα στην εφαρμογή. Επιπλέον, τα API μπορούν να χρησιμοποιηθούν για να εισάγουν δεδομένα από εξωτερικές πηγές, όπως πληροφορίες καιρού ή ειδήσεις,

που μπορούν να εμφανιστούν μέσα στην εφαρμογή.

### 2.2.2 Ευπάθειες web εφαρμογών

Οι πιο συχνές αδυναμίες σε μια web εφαρμογή εμφανίζονται στο σχεδιασμό, την υλοποίηση ή τη διαμόρφωση της εφαρμογής. Αυτά τα ευάλωτα σημεία ενέχουν σημαντικούς κινδύνους, οδηγώντας σε παραβιάσεις δεδομένων ή ακόμη οικονομικές απώλειες στην εφαρμογή. Σε αυτήν την ενότητα, θα αναλυθούν οι 10 πιο σημαντικές ευπάθειες που μπορεί να επηρεάσουν μια web εφαρμογή σύμφωνα με το OWASP (Open Worldwide Application Security Project)<sup>3</sup>.

#### Broken Access Control

Ο έλεγχος πρόσβασης (access control) διακυβεύεται όταν γίνεται δυνατή η μη εξουσιοδοτημένη πρόσβαση σε δεδομένα ή λειτουργίες, που συνήθως προστατεύονται από κάποιο user authentication. Η εκμετάλλευση αυτών των ευάλωτων σημείων μπορεί να επιτρέψει στους εισβολείς να πραγματοποιήσουν μη εξουσιοδοτημένες ενέργειες, όπως να κλέψουν ευαίσθητες πληροφορίες, να τροποποιήσουν λογαριασμούς χρηστών ή ακόμα και να αποκτήσουν πρόσβαση σε λειτουργίες διαχείρισης. Για την αποφυγή τέτοιων παραβιάσεων, είναι σημαντικό να εφαρμόζονται οι αρχές ελάχιστων προνομίων και να επιβάλλεται role-based access control (RBAC).

#### Cryptographic Failures

Οι κρυπτογραφικές αποτυχίες συμβαίνουν συνήθως όταν η κρυπτογράφηση είναι αδύναμη, με αποτέλεσμα την έκθεση προσωπικών δεδομένων. Αυτό μπορεί να οφείλεται στην χρήση πολύ παλιών αλγορίθμων κρυπτογράφησης, κακή διαχείριση κλειδιών ή αποτυχία κρυπτογράφησης ευαίσθητων πληροφοριών συνολικά. Για να αποφευχθεί αυτό, θα πρέπει να χρησιμοποιείται ισχυρή κρυπτογράφηση για την προστασία των δεδομένων τόσο κατά τη μεταφορά όσο και όσο είναι απλά αποθηκευμένα, η διαχείριση των κρυπτογραφικών κλειδιών θα πρέπει να γίνεται με ασφάλεια και δεν θα πρέπει να αποθηκεύονται περιττές ευαίσθητες πληροφορίες ή επιπλέον αντίγραφα.

#### Injection

Ευπάθειες τύπου injection παρουσιάζονται όταν αποστέλλονται μη αξιόπιστα δεδομένα σε έναν interpreter ως μέρος ενός command ή query, όπως SQL, NoSQL, OS ή LDAP injections. Οι εισβολείς μπορούν να εκμεταλλευτούν αυτά τα ευάλωτα σημεία για να εκτελέσουν μη εξουσιοδοτημένες εντολές ή να αποκτήσουν πρόσβαση σε δεδομένα. Για την αποφυγή τέτοιων επιθέσεων, συνίσταται η χρήση παραμετροποιημένων query, αποθηκευμένες διαδικασίες και επικύρωση εισόδου.

---

<sup>3</sup><https://owasp.org/www-project-top-ten/>

## **Insecure Design**

Αυτή η ευπάθεια εμφανίζεται όταν η ασφάλεια δεν έχει προτεραιότητα στην αρχή της ανάπτυξης μιας εφαρμογής. Μπορεί να οδηγήσει σε ζητήματα όπως insecure workflows ή έλλειψη βασικών ελέγχων ασφαλείας, δημιουργώντας δυσεύρετα ευπαθή σημεία μόλις η εφαρμογή είναι ενεργή.

## **Security Misconfiguration**

Η εσφαλμένη ρύθμιση παραμέτρων ασφαλείας (security misconfiguration) συμβαίνει όταν οι προεπιλεγμένες ρυθμίσεις ασφαλείας δεν ορίζονται ή διατηρούνται σωστά, οδηγώντας στην αύξηση των πιθανών ευάλωτων σημείων με την πάροδο του χρόνου. Αυτό μπορεί να περιλαμβάνει ζητήματα όπως προεπιλεγμένες ρυθμίσεις, ελλιπείς διαμορφώσεις, ανοιχτό χώρο αποθήκευσης στο cloud ή υπερβολικά λεπτομερή μηνύματα σφάλματος. Για να αποφευχθεί αυτό, οι οργανισμοί θα πρέπει να δημιουργήσουν ασφαλείς διαμορφώσεις, να αυτοματοποιήσουν την ανάπτυξή τους και να παρακολουθούν και να ελέγχουν τακτικά αυτές τις ρυθμίσεις.

## **Vulnerable and Outdated Components**

Η χρήση components με γνωστές αδυναμίες θέτει σε κίνδυνο την ασφάλεια ολόκληρης της εφαρμογής. Αυτό περιλαμβάνει παλιές βιβλιοθήκες, frameworks και άλλα components του λογισμικού. Για τη διαχείριση αυτών των κινδύνων, είναι απαραίτητη η τακτική ενημέρωση των ενοποιημένων που χρησιμοποιούνται στην εφαρμογή, επίσης συχνά security tests βοηθούν στον εντοπισμό αυτών των αδυναμιών.

## **Identification and Authentication Failures**

Αυτά τα περιστατικά προέρχονται από αποτυχίες στην αναγνώριση και τον έλεγχο ταυτότητας χρήστη, όπως κακή διαχείριση κωδικού πρόσβασης, αδύναμες πρακτικές ασφαλείας, έλλειψη multi-factor authentication (MFA) ή προβλήματα διαχείρισης περιόδου λειτουργίας. Για να αποφευχθεί η επίθεση σε αυτά τα σημεία, είναι σημαντικό να υπάρχουν ισχυροί έλεγχοι ταυτότητας των χρηστών και να υπάρχει σωστή διαχείρισή των περιόδων σύνδεσης του κάθε χρήστη.

## **Software and Data Integrity Failures**

Αυτή η κατηγορία αντιμετωπίζει τους κινδύνους που σχετίζονται με τις ενημερώσεις λογισμικού και την ακεραιότητα των δεδομένων. Για παράδειγμα, ένας κακόβουλος χρήστης μπορεί να εκμεταλλευτεί μη ασφαλείς μηχανισμούς ενημέρωσης. Για τον μετριασμό αυτών των κινδύνων, προτείνεται η χρήση ασφαλών διαδικασιών ενημέρωσης του συστήματος, η χρήση κρυπτογραφικών υπογραφών για την επικύρωση της ακεραιότητας του λογισμικού, καθώς και η ενσωμάτωση ελέγχων ασφαλείας σε αγωγούς CI/CD.

## Security Logging and Monitoring Failures

Η κακή καταγραφή και παρακολούθηση μπορεί να δυσκολέψει τον εντοπισμό ευάλωτων σημείων. Συχνά προβλήματα αυτής της περίπτωσης περιλαμβάνουν την έλλειψη καταγραφής αρχείων, την μη ασφαλή αποθήκευση τους (όπως η διατήρηση διευθύνσεων IP σε απλό κείμενο) και τη δυσκολία εντοπισμού παραβάσεων. Η αποτελεσματική ασφάλεια βασίζεται στην ύπαρξη υψηλής ποιότητας αρχείων καταγραφής, στην παρακολούθηση σε πραγματικό χρόνο για άμεση αντίδραση σε ειδοποιήσεις και σε ένα ισχυρό σύστημα ανάλυσης των αρχείων καταγραφής.

## Server-Side Request Forgery (SSRF)

Το SSRF (Server-Side Request Forgery) είναι μια ευπάθεια όπου ένας εισβολέας μπορεί να "ξεγελάσει" τον server ώστε να κάνει αυθαίρετα αιτήματα σε άλλα συστήματα, εκθέτοντας πιθανώς εσωτερικές υπηρεσίες και δεδομένα. Για προστασία από επιθέσεις SSRF βοηθάει η περιορισμένη πρόσβαση σε εξωτερικά δίκτυα μόνο όταν είναι απαραίτητη. Επιπλέον, ο οργανισμός πρέπει να διατηρεί το εσωτερικό του δίκτυο καλά τμηματοποιημένο από τα εξωτερικά δίκτυα.

## 2.3 SQL Injection

### 2.3.1 Τι είναι μια επίθεση SQL injection

Μια επίθεση SQL injection είναι μια επίθεση κατά την οποία ο επιτηθέμενος εκμεταλλεύεται ευπάθειες που υπάρχουν στα query της βάσης δεδομένων μιας web εφαρμογής εισάγοντας κακόβουλο SQL κώδικα σε πεδία εισόδου ή παραμέτρους. Αυτή η επίθεση -αν πετύχει - μπορεί να οδηγήσει σε μη εξουσιοδοτημένη πρόσβαση, διαρροή δεδομένων ή ακόμα και πλήρη έλεγχο της βάσης δεδομένων. Η επίθεση SQL injection, αν και είναι μια από τις πιο απλές και παλιές μορφές επιθέσεις παραμένει ιδιαίτερα επικίνδυνη για πολλές εφαρμογές λόγω κακής επικύρωσης εισόδου και λόγω "κακών" query. Ένα κλασικό παράδειγμα επίθεσης SQL injection περιλαμβάνει μια φόρμα σύνδεσης όπου ένας εισβολέας εισάγει «' OR '1'='1'» στο πεδίο κωδικού πρόσβασης, μετατρέποντας το SQL query σε μια συνθήκη που αξιολογείται πάντα ως αληθής, παρέχοντας ενδεχομένως μη εξουσιοδοτημένη πρόσβαση:

```
SELECT * FROM users WHERE username = 'admin' AND password = ''  
OR '1'='1' ;
```

Αφού εξηγήσαμε τι είναι μια επίθεση SQL injection, ας δούμε μερικά πραγματικά παραδείγματα όπου τέτοιες επιθέσεις είχαν σημαντικές επιπτώσεις. Στην επίθεση GhostShell, η ομάδα APT Team GhostShell στόχευσε 53 πανεπιστήμια χρησιμοποιώντας SQL injection, κλέβοντας και δημοσιεύοντας 36.000 προσωπικά αρχεία που ανήκαν σε φοιτητές, καθηγητές και προσωπικό. Ένα άλλο παράδειγμα αφορά τη ομάδα RedHack, η οποία χρησιμοποίησε SQL injection για να παραβιάσει έναν ιστότοπο της τουρκικής κυβέρνησης, διαγράφοντας



χρέη προς κυβερνητικές υπηρεσίες. Η παραβίαση του 7-Eleven είναι ένα ακόμη σημαντικό παράδειγμα όπου οι εισβολείς χρησιμοποίησαν SQL injection για να διεισδύσουν σε εταιρικά συστήματα κλέβοντας 130 εκατομμύρια αριθμούς πιστωτικών καρτών. Τελευταίο παράδειγμα είναι η παραβίαση του HBGary, που συνέβη από άτομα που συνδέονται με την ομάδα ακτιβιστών Anonymous, οι οποίοι χρησιμοποίησαν SQL injection για να κατεβάσουν τον ιστότοπο της ασφάλειας της εταιρείας. Αυτή η επίθεση έγινε ως αντίποινα για τον Διευθύνοντα Σύμβουλο της HBGary που δημοσίευσε ότι είχε ταυτοποιήσει μέλη των Anonymous.

### 2.3.2 Τύποι SQL injection

Υπάρχουν αρκετοί διαφορετικοί τύποι SQL injection, οι οποίοι θα αναλυθούν παρακάτω.

#### Union-based SQL Injection

Πρόκειται για την πιο συνηθισμένη μορφή SQL injection κατά την οποία ο εισβολέας εκμεταλλεύεται τον τελεστή UNION της SQL για να συνδυάσει τα αποτελέσματα δύο ή περισσότερων εντολών SELECT. Αυτό επιτρέπει στον εισβολέα να ανακτήσει δεδομένα από πολλούς πίνακες της βάσης δεδομένων με την χρήση ενός query. Αν η εισαγωγή του query είναι επιτυχής, ο εισβολέας μπορεί να αποκτήσει μη εξουσιοδοτημένη πρόσβαση σε ευαίσθητα προσωπικά δεδομένα που προστατεύονται διαφορετικά, όπως για παράδειγμα user credentials.

#### Error-Based Injection

Σε αυτή την περίπτωση ο εισβολέας εκμεταλλεύεται τα μηνύματα σφάλματος που δημιουργούνται από μια βάση δεδομένων όταν αποτυγχάνει να εκτελεστεί ένα query. Αυτή η μέθοδος είναι ιδιαίτερα αποτελεσματική έναντι των MS-SQL Servers. Ουσιαστικά ο εισβολέας προκαλεί την εφαρμογή να εκτελέσει queries που θα παράγουν σφάλμα και μέσω του μηνύματος που στέλνει το σφάλμα μπορεί να αποκαλύψει ακούσια πληροφορίες σχετικά με τη δομή της βάσης δεδομένων ή ακόμη και να περιλαμβάνει τα δεδομένα που ζητούνται από τον εισβολέα.

#### Blind SQL injection

Η blind SQL injection είναι η πιο δύσκολη μορφή injection επειδή δεν παράγει άμεσα μηνύματα σφάλματος ή εξόδους από τα οποία ο εισβολέας μπορεί να λάβει πληροφορίες. Αντίθετα, βασίζεται σε TRUE ή FALSE απαντήσεις από τη βάση δεδομένων για να λάβει πληροφορίες. Αυτό στην συνέχεια μπορεί να χωριστεί σε boolean-based SQL injection, όπου ο εισβολέας χρησιμοποιεί συγκεκριμένα SQL statements και εξάγει την πληροφορία που χρειάζεται για την βάση ανάλογα με την τιμή που λαμβάνει (TRUE/FALSE) και σε time-based SQL injection, όπου ο εισβολέας μετρά το χρόνο που απαιτείται για την εκτέλεση ενός query για να συμπεράνει εάν πληρούνται ορισμένες προϋποθέσεις.

Άλλος ένας τρόπος με τον οποίο μπορούν να ταξινομηθούν είναι με βάση την μέθοδο που χρησιμοποιεί ο εισβολέας για να εισάγει τα δεδομένα.

### **SQL Injection Based on User Input**

Αυτός ο τύπος επίθεσης είναι εφικτός όταν η εφαρμογή ενσωματώνει δεδομένα που προέρχονται από τους χρήστες σε queries της βάσης χωρίς κάποιον έλεγχο ότι αυτά είναι ασφαλή. Είναι η πιο απλή μέθοδος, καθώς τα πεδία εισαγωγής είναι από τους πιο συνηθισμένους τρόπους να εισάγει ο εισβολέας κακόβουλο SQL κώδικα.

### **SQL Injection Based on Cookies**

Η επίθεση αυτή περιλαμβάνει τον χειρισμό των δεδομένων που είναι αποθηκευμένα στα cookies ενός χρήστη, τα οποία στη συνέχεια αποστέλλονται στον web server και είναι πιθανό να χρησιμοποιηθούν σε queries της βάσης δεδομένων της εφαρμογής. Αυτή η περίπτωση είναι δυνατή μόνο αν τα cookies δεν έχουν επικυρωθεί σωστά.

### **SQL Injection Based on HTTP**

Στο injection αυτό ο εισβολέας αξιοποιεί το γεγονός ότι οι web εφαρμογές συχνά δέχονται δεδομένα από HTTP headers, όπως User-Agent ή Referer. Ο εισβολέας μπορεί να τροποποιήσει τα headers για να εισάγουν κώδικα SQL σε queries που εκτελούνται από την εφαρμογή με την χρήση αυτών των header.

### **Second-Order SQL Injection**

Πρόκειται για μια ιδιαίτερα ύπουλη μορφή επίθεσης καθώς ο επιτιθέμενος εισάγει κακόβουλο κώδικα που δεν εκτελείται αμέσως. Αντίθετα, ο κώδικας που εισάγεται αποθηκεύεται στη βάση δεδομένων και εκτελείται αργότερα όταν ανακτηθεί για την εκτέλεση κάποιας λειτουργίας. Αυτός ο τύπος injection είναι πολύ δύσκολο να εντοπιστεί επειδή ο κώδικας δεν φαίνεται κάπου μέχρι να ενεργοποιηθεί από μια συγκεκριμένη ενέργεια ή query.

## ΚΕΦΑΛΑΙΟ 3

# Προσομοίωση επιθέσεων SQL Injection

Για μια πρακτική προσέγγιση της θεωρίας που έχει αναπτυχθεί παραπάνω, δημιουργήθηκε ένα website που προσομοιώνει ευάλωτα σημεία που μπορεί συνήθως να εκμεταλλευτεί κάποιος μέσω διαφόρων τύπων επιθέσεων SQL injection. Το website ως host τον τοπικό υπολογιστή χρησιμοποιώντας XAMPP, επιτρέποντας ένα ασφαλές περιβάλλον δοκιμών εκτός σύνδεσης. Το website χρησιμοποιεί MariaDB για την δημιουργία των βάσεων δεδομένων και αποθηκεύονται στο phpmyAdmin. Ο σκοπός αυτού του website είναι να παρέχει πρακτικά παραδείγματα επιθέσεων SQL injection σε ένα ελεγχόμενο περιβάλλον, τονίζοντας πώς μπορούν να εκτελεστούν οι διαφορετικοί τύποι επιθέσεων SQL injection σε ένα ευάλωτο website.

Η αρχική σελίδα αποτελείται από επτά κουμπιά/επιλογές, το καθένα έχει τίτλο έναν συγκεκριμένο τύπο επίθεσης SQL injection. Με την επιλογή ενός κουμπιού, ο χρήστης ανακατευθύνετε σε ένα σχετικό σενάριο, όπως μια σελίδα log in, μια φόρμα αναζήτησης ή ένα πεδίο εισαγωγής, σχεδιασμένο να είναι ευάλωτο στον συγκεκριμένο τύπο επίθεσης έγχυσης. Οι τύποι SQL injection που παρουσιάζονται είναι :Union-based injection, Error-based injection, Blind SQL injection, SQL Injection Based on User Input ,SQL Injection Based on Cookies, SQL Injection Based on HTTP headers και Second-Order SQL Injection. Παρακάτω φαίνεται η αρχική σελίδα του website:

## SQL Injection Demo

SQL Injection Based on User Input

Blind SQL Injection

Error-Based Injection

Union-based SQL Injection

Second-Order SQL Injection

SQL Injection Based on HTTP  
Headers

SQL Injection via Cookies

Στην συνέχεια θα αναλυθεί το κάθε σενάριο επίθεσης και θα παρουσιαστούν τρόποι με τους οποίους μπορεί να προστατευτεί μια σελίδα από αυτές τις επιθέσεις.

### 3.1 SQL Injection Based on User Input

Για τον πρώτο τύπο SQL injection δημιουργήθηκε μια σελίδα log in που επιτρέπει στους χρήστες να συνδεθούν παρέχοντας ένα όνομα χρήστη και έναν κωδικό πρόσβασης. Σε περίπτωση που το όνομα χρήστη και ο κωδικός είναι σωστά η σελίδα επιστρέφει το μήνυμά "Login successful!" ενώ στην περίπτωση που είναι λάθος επιστρέφει το μήνυμα "Invalid username or password". Στο backend αυτής της σελίδας για τον έλεγχο της σύνδεσης εκτελείτε ένα SQL query για να επαληθεύσει εάν ο παρεχόμενος κωδικός και όνομα χρήστη υπάρχουν στη βάση δεδομένων. Στην τρέχουσα μορφή της, η λογική της σύνδεσης είναι πολύ επιρρεπής σε επιθέσεις SQL injection λόγω του μη ασφαλούς τρόπου χειρισμού των δεδομένων που δίνονται από τους χρήστες.

Όταν ένας χρήστης υποβάλλει τη φόρμα σύνδεσης, δημιουργείται το ακόλουθο SQL query:

```
SELECT * FROM users WHERE username = '$username' AND password = '$password' ;
```

Σε αυτό το query, οι τιμές username και password λαμβάνονται απευθείας από την είσοδο του χρήστη και εισάγονται στο query χωρίς κατάλληλη επικύρωση ή sanitization. Ως αποτέλεσμα, κακόβουλοι χρήστες μπορούν να χειριστούν το query με τρόπους που ο προγραμματιστής δεν σκόπευε και να αποκτήσουν πρόσβαση χωρίς να γνωρίζουν κάποιο όνομα χρήστη και κωδικό που υπάρχουν στην βάση δεδομένων.

Για παράδειγμα, εάν ένας εισβολέας εισάγει 'OR '1'='1 και στα δύο πεδία ονόματος χρήστη και κωδικού πρόσβασης, το SQL query που προκύπτει έχει την παρακάτω μορφή:

```
SELECT * FROM users WHERE username = ' ' OR '1'='1 ' AND password = ' '
```

Σε αυτό το query η πρόταση 'OR '1'='1' είναι πάντα αληθής επειδή το '1'='1' είναι ταυτολογία. Ως αποτέλεσμα, ολόκληρη η συνθήκη WHERE αξιολογείται ως true και το query επιστρέφει όλους χρήστες που υπάρχουν στον πίνακα users της βάσης δεδομένων ανεξάρτητα από το αν το όνομα χρήστη ή ο κωδικός πρόσβασης είναι έγκυρα.

Αυτό επιτρέπει στον εισβολέα να παρακάμψει πλήρως τον έλεγχο ταυτότητας και να αποκτήσει μη εξουσιοδοτημένη πρόσβαση στο σύστημα, κάτι που αποτελεί κρίσιμο κίνδυνο για την ασφάλεια του συστήματος. Η επιτυχία αυτής της επίθεσης οφείλεται στο γεγονός ότι δεν γίνεται σωστός έλεγχος των δεδομένων που δίνει ο χρήστης. Όταν τα δεδομένα του χρήστη συνδέονται απευθείας σε ένα SQL query χωρίς κατάλληλο έλεγχο, οι εισβολείς μπορούν να προσθέσουν κακόβουλο κώδικα. Έτσι, αλλάζουν το ερώτημα και το κάνουν να λειτουργεί διαφορετικά από το κανονικό.

## Login

Username:

Password:

Login

Login successful!

Για την αντιμετώπιση αυτής της ευπάθειας, είναι απαραίτητη η εφαρμογή των prepared statements με parameterized queries. Τα προεπαρασκευασμένα διασφαλίζουν ότι τα δεδομένα που εισάγουν οι χρήστες αντιμετωπίζονται αυστηρά ως δεδομένα και όχι ως εκτελέσιμος κώδικας, αποτρέποντας έτσι επιθέσεις SQL injection.

Ακολουθεί ο τρόπος με τον οποίο το query που ταυτοποιεί τον χρήστη και ολοκληρώνει την σύνδεση μπορεί να επαναδιατυπωθεί χρησιμοποιώντας parameterized queries:

```
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ?  
AND password = ?");  
$stmt->bind_param("ss", $username, $password);  
$stmt->execute();  
$result = $stmt->get_result();
```

Με την χρήση prepared statements το SQL query δημιουργείται με placeholders, όπως το "?", και οι πραγματικές τιμές για τα username και password προστίθενται σε αυτά τα placeholders αφού το query έχει προετοιμαστεί. Αυτό εξασφαλίζει ότι οι τιμές που δίνει ο χρήστης αντιμετωπίζονται μόνο ως δεδομένα και όχι ως μέρος της εντολής SQL, αποτρέποντας έτσι την εισαγωγή κακόβουλου κώδικα.

Η μέθοδος bind\_param χρησιμοποιείται για να προσθέσει τα δεδομένα του χρήστη με ασφάλεια, δηλαδή διαχειρίζεται τυχόν επικίνδυνους χαρακτήρες, όπως εισαγωγικά, που θα μπορούσαν να χρησιμοποιηθούν σε επιθέσεις SQL injection. Με αυτό τον τρόπο, ακόμα κι αν κάποιος προσπαθήσει να εισάγει κακόβουλο κώδικα, η βάση δεδομένων θα τον δει ως απλό κείμενο και δεν θα τον εκτελέσει.

## 3.2 Blind SQL Injection

Για αυτόν τον τύπο επίθεσης δημιουργήθηκε μια φόρμα αναζήτησης χρηστών. Εισάγεται κάποιο όνομα χρήστη και αν βρεθεί στην βάση δεδομένων τότε εκτυπώνεται το μήνυμα "User found!" ενώ αν δεν υπάρχει εμφανίζεται το μήνυμα "No user found!.. Το backend επεξεργάζεται την είσοδο του χρήστη με την χρήση ενός SQL query, το οποίο ελέγχει εάν ένας χρήστης με το συγκεκριμένο όνομα χρήστη υπάρχει στη βάση δεδομένων. Ωστόσο, ο κώδικας είναι ευάλωτος σε επίθεσή Blind SQL Injection, έναν τύπο επίθεσης που επιτρέπει σε έναν εισβολέα να συμπεράνει πληροφορίες σχετικά με τη βάση δεδομένων έμμεσα, ακόμη και όταν δεν παρέχονται άμεσα μηνύματα σφάλματος ή κάποια έξοδος με πληροφορίες της βάσης δεδομένων. Στην τρέχουσα μορφή του, το εύαλωτο SQL query είναι το εξής:

```
SELECT * FROM users WHERE username = '$input';
```

Όπως φάνηκε και στην προηγούμενη περίπτωση το query λαμβάνει απευθείας τα δεδομένα του χρήστη και τα χρησιμοποιεί για την εκτέλεση της SQL εντολής χωρίς κάποιο sanitization. Στη συνέχεια, το σύστημα εκτελεί το query και επιστρέφει το αντίστοιχο μήνυμα. Αν και αυτό το query φαίνεται απλό, μπορεί να αξιοποιηθεί από κάποιον εισβολέα για μια επίθεση τύπου Blind SQL Injection. Η επίθεση αυτή διαφέρει από την τυπική επίθεση SQL injection

καθώς ο εισβολέας δεν λαμβάνει απευθείας σχόλια από τη βάση δεδομένων με τη μορφή μηνυμάτων σφάλματος ή δεδομένων. Αντίθετα, μπορεί να συμπεράνει πληροφορίες για την βάση δεδομένων βασιζόμενος στην απάντησή που δίνει γενικά το σύστημα, όπως το εάν ένας χρήστης βρέθηκε ή όχι.

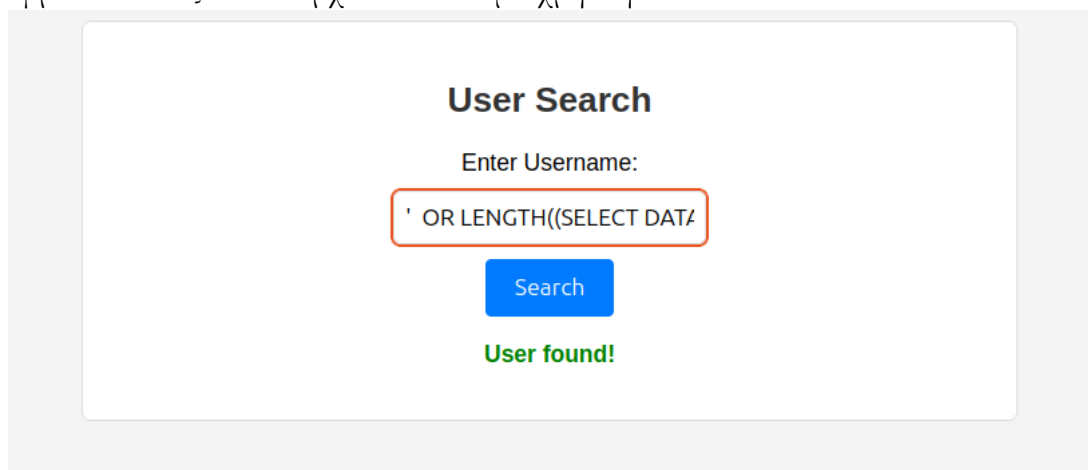
Σε αυτήν την περίπτωση, ένας εισβολέας μπορεί να δημιουργήσει ένα query που χειρίζεται τη λογική του υπάρχοντος query για να αποκαλύψει πληροφορίες σχετικά με τη δομή της βάσης δεδομένων, όπως το μήκος του ονόματος της. Για παράδειγμα, η ακόλουθη εντολή θα μπορούσε να εισαχθεί στο πεδίο αναζήτησης ονόματος χρήστη:

```
' OR LENGTH((SELECT DATABASE())) > 5 — '
```

Το παραπάνω query ελέγχει το μήκος του ονόματος της τρέχουσας βάσης δεδομένων εκτελώντας την εντολή (SELECT DATABASE()), η οποία ανακτά το όνομα της βάσης δεδομένων. Στη συνέχεια, ελέγχει εάν το μήκος του ονόματος της βάσης δεδομένων είναι μεγαλύτερο από 5. Τέλος η ακολουθία – είναι ο τρόπος εισαγωγής σχολίων στην SQL, έτσι όταν εκτελεστεί αυτό το query διασφαλίζεται ότι ο υπόλοιπος κώδικας θα αγνοηθεί με αποτέλεσμα να ελεγχθεί μόνο η συνθήκη που έχει εισαχθεί από τον εισβολέα. Το query έχει αυτή την μορφή :

```
SELECT * FROM users WHERE username = '' OR LENGTH((SELECT DATABASE())) > 5 — '
```

Εκτελείται σε δύο μέρη, αρχικά η συνθήκη username = '' ελέγχει για ένα κενό όνομα χρήστη, το οποίο δεν ταιριάζει σε κανέναν χρήστη της βάσης δεδομένων. Στην συνέχεια με την χρήση του OR γίνεται ο έλεγχος LENGTH((SELECT DATABASE())) > 5 ο οποίος επιστρέφει την τιμή True αν το μήκος του ονόματος της βάσης δεδομένων είναι μεγαλύτερο από 5, ενώ σε άλλη περίπτωση επιστρέφει την τιμή False. Έτσι αν αυτή η συνθήκη ισχύει το query επιστρέφει ένα αποτέλεσμα σαν να βρέθηκε χρήστης ενώ στην πραγματικότητα δεν έχει βρεθεί καθώς δεν υπάρχει κενό όνομα χρήστη.



Αυτή η επίθεση επιτρέπει στον εισβολέα να συμπεράνει το μήκος του ονόματος της βάσης



δεδομένων χωρίς ποτέ να δει απευθείας το όνομα. Επαναλαμβάνοντας παρόμοια query, ο εισβολέας θα μπορούσε να εξαγάγει περισσότερες λεπτομέρειες σχετικά με τη βάση δεδομένων και το περιεχόμενό της.

Για να μην υπάρχει η πιθανότητα να μπορεί κάποιος να εκτελέσει αυτή την επίθεση είναι σημαντικό να αποφεύγετε τη συμπερίληψη των δεδομένων που δίνει ο χρήστης απευθείας στο SQL query. Αν και είναι ένα απλό λάθος είναι πολύ συνηθισμένο αλλά και επικίνδυνο. Επομένως συνιστάται η χρήση prepared statements και parameterized queries. Ο ασφαλής τρόπος αναζήτησής ενός χρήστη είναι ο εξής

```
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ?")
;
$stmt->bind_param("s", $input); // "s" means the input is a
    string
$stmt->execute();
$result = $stmt->get_result();
```

Με αυτή την διατύπωση, χρησιμοποιούνται placeholders για το όνομα χρήστη, και η μέθοδος bind\_param δεσμεύει την είσοδο του χρήστη σε αυτό το σημείο. Έτσι, η είσοδος αντιμετωπίζεται ως απλό κείμενο και δεν μπορεί να αλλάξει τη δομή του query. Η μετάβαση σε prepared statements με parameterized queries αποτρέπει τους εισβολείς από το να χειραγωγούν το query μέσω εισαγωγής δεδομένων. Αυτή η πρακτική βελτιώνει τη συνολική ασφάλεια της εφαρμογής, καθιστώντας τις αλληλεπιδράσεις με τη βάση δεδομένων ασφαλείς. Η χρήση prepared statements είναι κρίσιμη για την προστασία ευαίσθητων πληροφοριών και την αποτροπή μη εξουσιοδοτημένης πρόσβασης, ακόμη και στις πιο απλές λειτουργίες, όπως τα πεδία αναζήτησης ή σύνδεσης.

### 3.3 Error-Based SQL Injection

Για αυτή την περίπτωση δημιουργήθηκε μια φόρμα αναζήτησης όπου οι χρήστες μπορούν να εισάγουν το id ενός προϊόντος για να ανακτήσουν τις αντίστοιχες λεπτομέρειες για αυτό, όπως το όνομα και την τιμή. Το backend επεξεργάζεται την είσοδο του χρήστη και με την χρήση ενός SQL query πραγματοποιεί αναζήτηση στη βάση δεδομένων για ένα προϊόν με το αντίστοιχο id. Ωστόσο, η φόρμα αναζήτησης είναι ευάλωτη σε Error-Based SQL Injection, μια μορφή SQL Injection όπου οι εισβολείς εκμεταλλεύονται μηνύματα σφάλματος βάσης δεδομένων για να εξαγάγουν πληροφορίες σχετικά με την δομή της βάσης δεδομένων. Το SQL query που είναι ευάλωτο είναι το εξής:

```
SELECT * FROM products WHERE id = '$id';
```

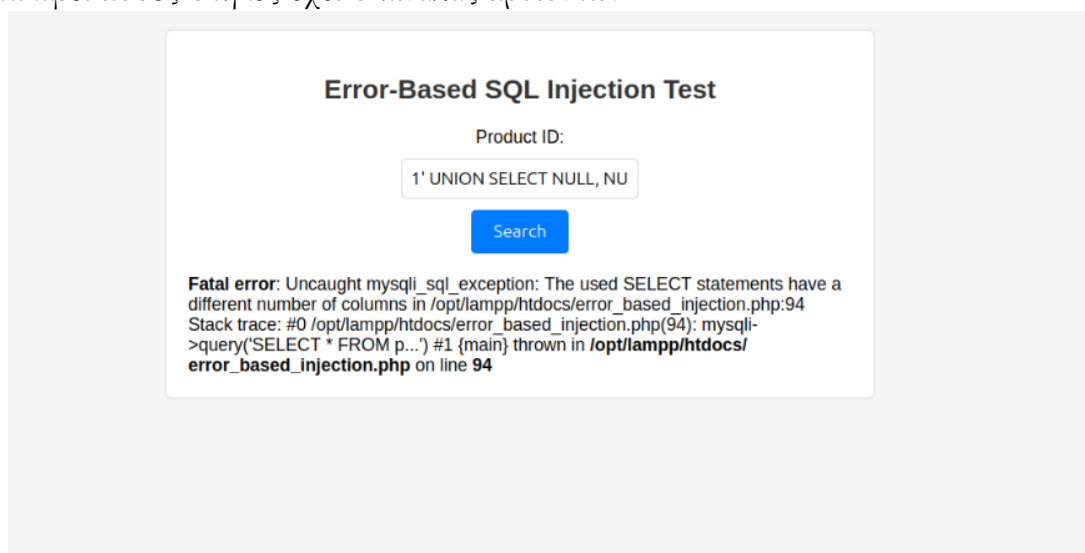
Όπως και στις προηγούμενες περιπτώσεις η αναζήτηση γίνεται χρησιμοποιώντας απευθείας την είσοδο δεδομένων του χρήστη. Οι εισβολείς μπορούν να εισάγουν queries στην φόρμα αναζήτησης και εκμεταλλεύονται τα μηνύματα σφάλματος που επιστρέφονται από τη βάση

δεδομένων για να κατανοήσουν τη δομή της και να αποκτήσουν σχετικές πληροφορίες όπως για παράδειγμα τον αριθμό των στηλών σε έναν πίνακα.

Για παράδειγμα, εάν ο εισβολέας εισάγει τα ακόλουθα στο πεδίο αναζήτησης προϊόντος:

```
1' UNION SELECT NULL, NULL — '
```

Αυτό το query επιχειρεί να συνδυάσει το αρχικό query : `SELECT * FROM products WHERE id = '$id'`; με την χρήση της εντολής `UNION SELECT`. Ο τελεστής `UNION` χρησιμοποιείται για να συνδυάσει το σύνολο αποτελεσμάτων δύο ή περισσότερων δηλώσεων `SELECT`, αλλά για να πετύχει αυτή η λειτουργία, και οι δύο εντολές `SELECT` πρέπει να επιστρέψουν τον ίδιο αριθμό στηλών. Ο εισβολέας δεν γνωρίζει πόσες στήλες έχει ο πίνακας επομένως με την εντολή `1' UNION SELECT NULL, NULL — '` δεν είναι σίγουρο ότι θα εκτυπώσει για παράδειγμα όλες τις στήλες του πίνακα. Εκεί που βασίζεται η επίθεση είναι στο μήνυμα λάθους που θα επιστρέψει η βάση δεδομένων από το οποίο μπορεί ο εισβολέας να καταλάβει πόσες στήλες έχει ο πίνακας προϊόντων.



Το παραπάνω μήνυμα λάθους δείχνει ξεκάθαρα ότι υπάρχει αναντιστοιχία στο πλήθος στηλών που 'ζητάει' ο εισβολέας και σε αυτό που υπάρχει όντως στον πίνακα της βάσης. Έτσι ο εισβολέας μπορεί να δοκιμάσει εντολή που ζητάει παραπάνω ή λιγότερες στήλες μέχρι να πετύχει τον αριθμό στηλών που υπάρχουν στην βάση. Για παράδειγμα το query : `1' UNION SELECT NULL, NULL, NULL — '` είναι επιτυχές επειδή πλέον επιλέγει τρεις τιμές `NULL` οι οποίες αντιστοιχούν στις τρεις στήλες στον πίνακα προϊόντων.

Όπως έχει αναφερθεί και παραπάνω η πιο αποτελεσματική λύση για προστασία ενάντια σε SQL injection είναι η χρήση των prepared statements με parameterized queries. Παρακάτω φαίνεται πως μπορούμε να υλοποιήσουμε με ασφάλεια την αναζήτηση του προϊόντος χρησιμοποιώντας το id που μας δίνει ο χρήστης.

```
$id = $_POST['id'];  
$stmt = $conn->prepare("SELECT * FROM products WHERE id = ?");
```

```
$stmt->bind_param("s", $id);  
$stmt->execute();  
$result = $stmt->get_result();
```

Με την χρήση των placeholders η είσοδος του χρήστη μεταβιβάζεται με ασφάλεια μέσω της μεθόδου `bind_param`. Αυτό διασφαλίζει ότι τα δεδομένα που εισάγονται από τον χρήστη αντιμετωπίζεται ως απλή συμβολοσειρά και δεν μπορούν να αλλάξει τη δομή του query, παρέχοντάς προστασία από την εισαγωγή κακόβουλου κώδικα. Επιπλέον, ο σωστός χειρισμός σφαλμάτων (π.χ αποφυγή αναλυτικού μηνύματος σφάλματος) είναι σημαντικός για την προστασία του συστήματος από error-based injections γιατί εμποδίζει τους εισβολείς από το να αποκτήσουν πληροφορίες για τη βάση δεδομένων μέσω μηνυμάτων σφάλματος.

### 3.4 Union-based SQL Injection

Μια union-based SQL injection χρησιμοποιεί τον τελεστή UNION για να συνδυάσει το αποτέλεσμα του αρχικού query με ένα ακόμη SELECT statement το οποίο αναχτά πληροφορίες για την δομή της βάσης δεδομένων. Για την επίδειξη μιας union-based επίθεσης δημιουργήθηκε μια φόρμα αναζήτησης προϊόντων. Ο χρήστης εισάγει στο πεδίο αναζήτησης το όνομα του προϊόντος και η αναζήτηση του επιστρέφει την τιμή. Ο τρόπος με τον οποίο λειτουργεί η αναζήτηση του προϊόντος καθιστά το σύστημα ευάλωτο σε επίθεσή SQL injection.

```
SELECT * FROM products WHERE name LIKE '%$search%';
```

Το παραπάνω query ενσωματώνει απευθείας τα δεδομένα που εισάγει ο χρήστης στο SELECT statement με αποτέλεσμα κάποιος κακόβουλος χρήστης να μπορεί να εισάγει SQL κώδικα και να αποσπάσει πληροφορίες για την βάση δεδομένων. Για παράδειγμα αν κάποιος χρήστης εισάγει το παραάτω χυερψ στην φόρμα αναζήτησης καταφέρνει να εκτυπώσει όλο τον πίνακα των προϊόντων της βάσης καθώς και το όνομα της βάσης.

```
' UNION SELECT NULL,NULL database() — '
```

### Product Search

Enter Product Name:

' UNION SELECT NULL,NULL

Search

Product: Union Product A - Price: 49.99  
Product: Union Product B - Price: 59.99  
Product: Union Product C - Price: 69.99  
Product: - Price: db\_union

Μια union-based επίθεση μπορεί πολλές φορές να συνδυαστεί με blind ή error-based injection αφού ο εισβολέας χρειάζεται πρώτα να αποκτήσει κάποιες πληροφορίες για το σύστημα. Για παράδειγμα και στην παραπάνω περίπτωση (error-based SQL injection) αφού ο εισβολέας μάθει πόσες στήλες έχει ο πίνακας μπορεί να επιλέξει τον σωστό αριθμό NULL τιμών και να αποσπάσει τις πληροφορίες που φαίνονται παραπάνω. Για την προστασία του συστήματος είναι απαραίτητη, όπως έχει αναφερθεί και παραπάνω, η χρήση prepared statements με parametrized queries.

```
$search = $_POST[ 'search ' ];  
$stmt = $conn->prepare( "SELECT * FROM products WHERE name LIKE ?  
    " );  
$search_param = "%{$search}%";  
$stmt->bind_param( "s", $search_param );  
$stmt->execute();  
$result = $stmt->get_result();
```

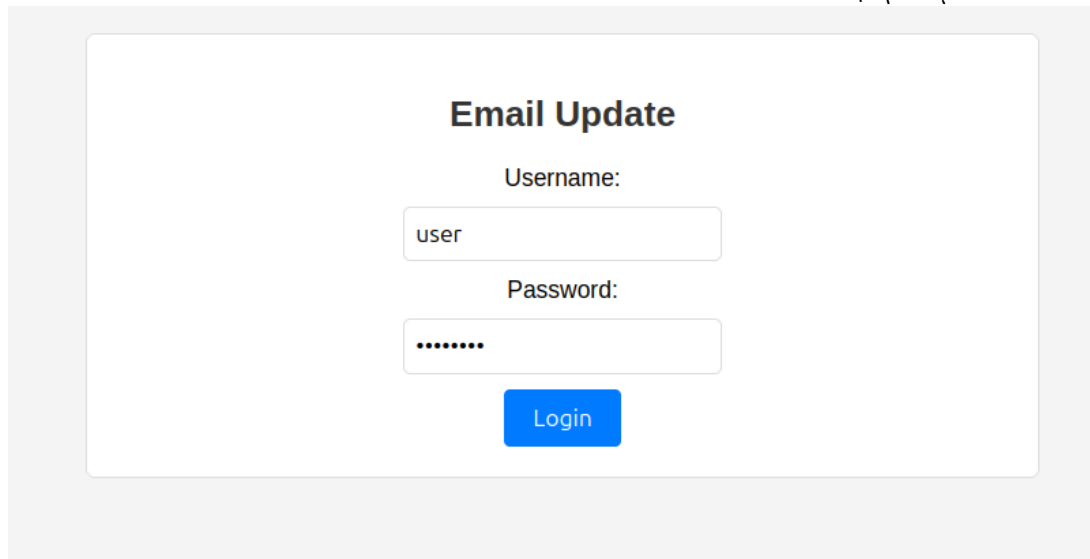
Η χρήση των prepared statements είναι ο πιο αποτελεσματικός τρόπος προστασίας ενός συστήματος από οποιαδήποτε μορφή SQL injection αφού διασφαλίζει ότι υπάρχει το απαραίτητο sanitization των δεδομένων που εισάγουν οι χρήστες σε οποιαδήποτε φόρμα(σύνδεσης ή αναζήτησης).

### 3.5 Second-Order SQL Injection

Αυτό το κομμάτι του website παρουσιάζει μια από τις πιο ιδιαίτερες μορφές επίθεσης SQL injection, την second order SQL injection. Χαρακτηρίζεται από την δυσκολία στον εντοπισμό της ειδικά όταν πρόκειται για περίπλοκες εφαρμογές. Η διαφορά αυτής της επίθεσης με τις επιθέσεις first order SQL injection (π.χ error-based, blind injection) είναι ότι τα αποτελέσματα

της δεν φαίνονται απευθείας. Στο παράδειγμα του website η επίθεση παρουσιάζεται μέσω μιας φόρμας ενημέρωσης email. Ένας κακόβουλος χρήστης μπορεί να χειραγωγήσει αυτήν τη φόρμα για να ενημερώσει το email ενός άλλου χρήστη (π.χ. του διαχειριστή) εισάγοντας ένα SQL query.

Η σελίδα αρχικά ζητάει από τον χρήστη να συνδεθεί ώστε να μπορεί να ενημερώσει το email του. Έστω ότι ο χρήστης υσερ θέλει να αλλάξει το email του διαχειριστή της σελίδας. Καθώς δεν μπορεί να παραβιάσει την λογική σελίδα για να συνδεθεί ως διαχειριστής και να το αλλάξει από εκεί συνδέεται κανονικά από τον δικό του λογαριασμό.



Από εκεί εισάγει το εξής χυερψ στην φόρμα ενημέρωσης email:

```
attacker@email.com' WHERE username = 'admin' — '
```

Email updated for user.

### Email Update

Enter New Email:

HERE username = 'admin' -- '

Update Email

Enter Username to Retrieve Email:

Retrieve Email

Logout

Η ενημέρωση φαίνεται να έγινε κανονικά για τον χρήστη user. Όταν όμως συνδεθεί ο διαχειριστής και προσπαθήσει να ανακτήσει το email του η σελίδα θα επιστρέψει το email που έβαλε ο χρήστης user στο παραπάνω query.

Email for admin: attacker@email.com

### Email Update

Enter New Email:

Update Email

Enter Username to Retrieve Email:

Retrieve Email

Logout

Αυτό συμβαίνει γιατί όταν ο user έγραψε το query στην φόρμα ενημέρωσης στο backend της σελίδας εκτελέστηκε το εξής query :

```
UPDATE users SET email = 'attacker@email.com' WHERE username = 'admin' —';
```

Επομένως ενημερώθηκε το email για τον χρήστη admin χωρίς να το έχει κάνει ο ίδιος του. Αυτό είναι ένα κλασικό παράδειγμα second-order injection, όπου η εισαγωγή του κακόβουλου κώδικα δεν δείχνει κάποιο αποτέλεσμα απευθείας, αλλά αποθηκεύεται και "ενεργοποιείται" αργότερα όταν εκτελείται κάποιο άλλο query (όπως η ανάκτηση email) αποκτά πρόσβαση στα δεδομένα που έχουν εισαχθεί από τον εισβολέα.

Γενικά για την προστασία από second-order injection πρέπει να υπάρχει σωστό sanitization όλων των αλληλεπιδράσεων με την βάση δεδομένων, είτε πρόκειται για εισαγωγή/ενημέρωση δεδομένων στην βάση ή και κατά και την ανάκτηση τους. Στο παραπάνω παράδειγμα πρέπει να υπάρχει προσοχή και στα δεδομένα που θα δώσει ο χρήστης αλλά και στο τι θα ζητήσει από την φόρμα ανάκτησης email.

Η πιο ασφαλής λύση είναι η προσθήκη prepared statements τόσο στο κομμάτι ενημέρωσης του email όσο και στο κομμάτι ανάκτησης ώστε να αποτραπεί η εισαγωγή κακόβουλου κώδικα με οποιαδήποτε μορφή. Επομένως οι παραπάνω εντολές θα είναι πιο ασφαλής αν εκτελούνται με αυτή την μορφή:

```

if (isset($_SESSION['username'])) {
    $new_email = $_POST['email'];
    $username = $_SESSION['username'];

    $stmt = $conn->prepare("UPDATE users SET email = ? WHERE
        username = ?");
    $stmt->bind_param("ss", $new_email, $username);
    if ($stmt->execute()) {
        echo "<p class='message success'>Email updated for
            $username.</p>";
    } else {
        echo "<p class='message error'>Error updating email:
            " . htmlspecialchars($conn->error) . "</p>";
    }
    $stmt->close();
}

```

Ο παραπάνω κώδικας φροντίζει να είναι ασφαλής τα δεδομένα που εισάγει ο χρήστης στο κομμάτι ενημέρωσης email.

```

if (!empty($username)) {
    $stmt = $conn->prepare("SELECT email FROM users WHERE
        username = ?");
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
            echo "<p class='message success'>Email for " .
                htmlspecialchars($username) . ": " .
                htmlspecialchars($row["email"]) . "</p>";
        }
    } else {
        echo "<p class='message error'>No user found with
            username: " . htmlspecialchars($username) . "</p>";
    }

    $stmt->close();
}
}

```

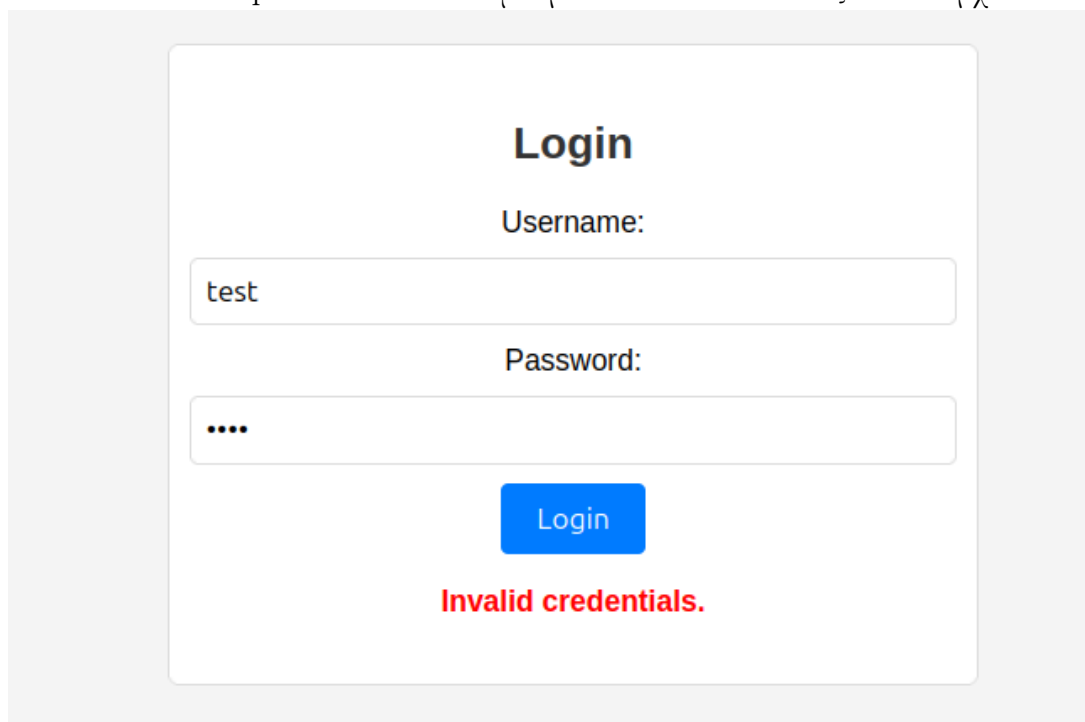


---

Τέλος παραπάνω δίνεται για την ασφαλή ανάκτηση της διεύθυνσής email του κάθε χρήστη. Με αυτές τις αλλαγές καμία μορφή SQL injection(first-order ή second-order) δεν είναι εφικτή.

### 3.6 SQL Injection Based on HTTP Headers

Τα περισσότερα website βασίζονται σε HTTP Headers για να διευκολύνουν την επικοινωνία μεταξύ client και server καθώς μέσω των header μεταφέρουν σημαντικές πληροφορίες όπως user agent και cookies. Μέσω αυτών των header επηρεάζεται ο τρόπος με τον οποίο επεξεργάζεται αιτήματα(requests) ο server καθώς πως μεταδίδονται οι απαντήσεις(responses) στους χρήστες. Ωστόσο αν μια εφαρμογή ή ένα website δεν έχει φροντίσει να υπάρχει σωστό sanitization τότε μπορεί κάποιος εισβολέας, με την χρήση εργαλείων όπως το curl ή το burp, να χειριστεί τιμές των headers όπως user agent για να εισάγει κακόβουλο περιεχόμενο. Στο παράδειγμα του website παρουσιάζεται μια σελίδα log in στην οποία στην αρχή ο χρήστης με username test και password test δεν μπορεί να συνδεθεί καθώς δεν υπάρχει.

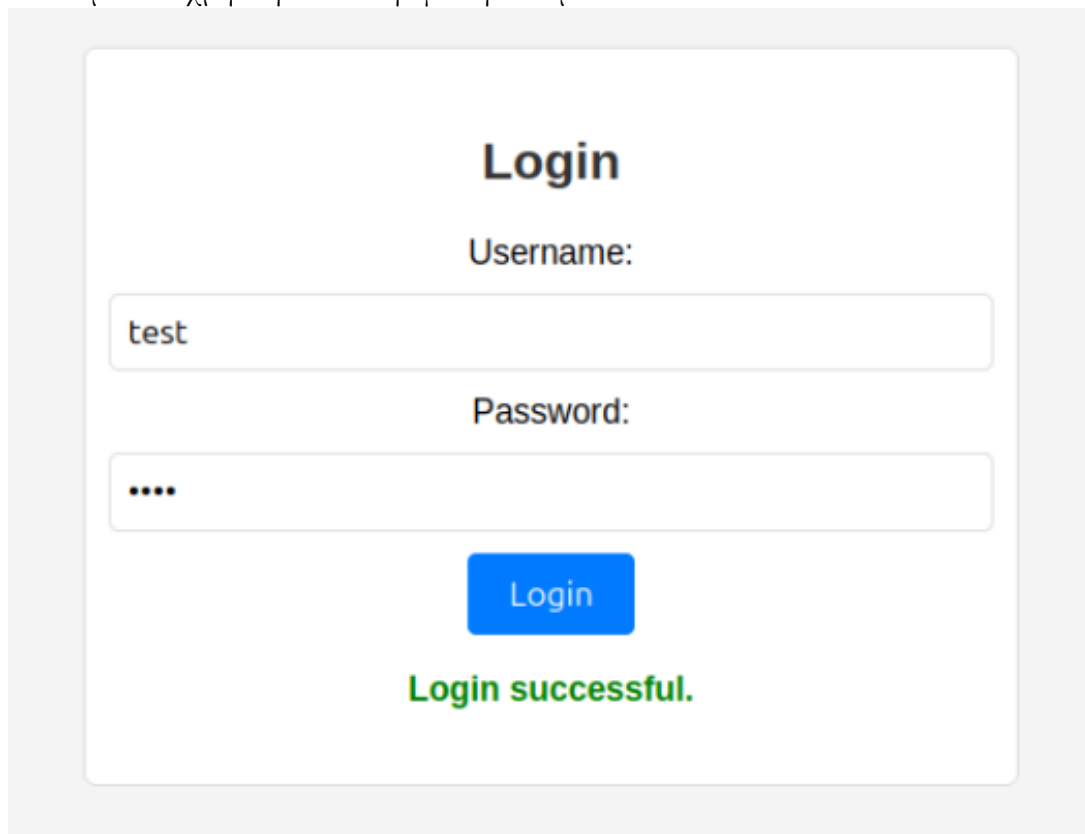


The image shows a web browser window displaying a login page. The page has a light gray background. In the center, there is a white rounded rectangle containing the login form. At the top of this rectangle, the word "Login" is written in bold black text. Below it, the label "Username:" is followed by a text input field containing the text "test". Underneath the username field, the label "Password:" is followed by a password input field with four dots "...." inside. Below the password field is a blue button with the text "Login" in white. At the bottom of the white rectangle, the text "Invalid credentials." is displayed in red. The entire login form is centered on the page.

Στην συνέχεια εκτελούμε την παρακάτω εντολή :

```
curl -H "User-Agent: Mozilla/5.0" -d "username=test&password=test" http://10.0.0.3/http_header_injection.php
```

Αν προσπαθήσουμε να συνδεθούμε αυτή τη φορά γίνεται καθώς μέσω της παραπάνω εντολής προσθέσαμε τον χρήστη test στην βάση δεδομένων.



Αρχικά η εντολή curl είναι ένα εργαλείο που χρησιμοποιείται για τη μεταφορά δεδομένων προς και από servers, είναι γνωστό εργαλείο για testing σε web apps και websites καθώς μπορεί κανείς εύκολα να κάνει HTTP αιτήματα. Σε αυτό το παράδειγμα :

- - H "User-Agent: Mozilla/5.0" :Αυτό το κομμάτι της εντολής κάνει τον HTTP header(- H) να μιμείται ένα αίτημα που προέρχεται από κάποιο browser. Ο header User-Agent προσδιορίζει τον client που κάνει το αίτημα και σε αυτήν την περίπτωση, έχει ρυθμιστεί να μοιάζει με αίτημα από το Mozilla Firefox (Mozilla/5.0), παρόλο που το αίτημα προέρχεται από το εργαλείο curl. Αυτό επιτρέπει στους εισβολείς να ελέγχουν τι στέλνεται στον server, και αν ο server βασίζεται σε κάποιον από αυτούς τους headers (π.χ. User Agent) για επεξεργασία μπορούν να το εκμεταλλευτούν για την εισαγωγή περιεχομένου.
- -d "username=test&password=test" :Η εντολή -d στο curl χρησιμοποιείται για τον

καθορισμό δεδομένων σε αιτήματα POST, στέλνοντας αυτά τα ζεύγη κλειδιού-τιμής σαν να είχαν υποβληθεί μέσω μιας φόρμας κάποιου website. Στο παράδειγμα αυτό στέλνει δεδομένα συγκεκριμένα για τα πεδία ονόματος χρήστη και κωδικού πρόσβασης.

Ουσιαστικά η παραπάνω εντολή στέλνει ένα αίτημα POST στην διεύθυνση `http://10.0.0.3/http_header_injection.php` με τα δεδομένα, όνομα χρήστη και κωδικό, που θέλει να εισάγει στην βάση.

Για την προστασία από τέτοιες επιθέσεις πρέπει αρχικά να φροντίσουμε ότι είναι ασφαλή τα δεδομένα που δέχεται η σελίδα από HTTP headers. Η χρήση της συνάρτησης `filter_var()` φροντίζει να υπάρχει σωστό sanitization για οποιαδήποτε τιμή δοθεί στον User-Agent header. Επομένως η εντολή θα πρέπει να είναι κάπως έτσι :

```
$user_agent = filter_var($_SERVER[ 'HTTP_USER_AGENT' ] ,  
    FILTER_SANITIZE_STRING);
```

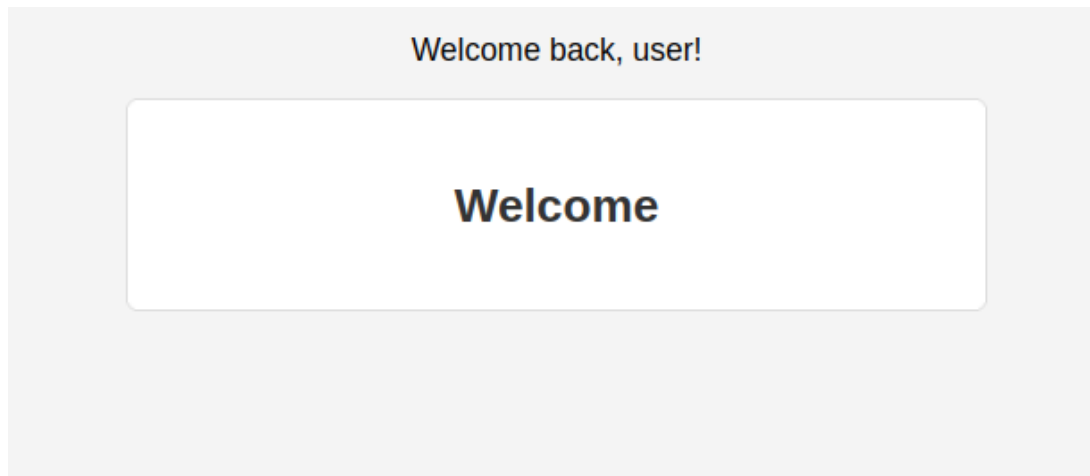
Για να είναι σίγουρα ασφαλής το website μπορούν επίσης να χρησιμοποιηθούν prepared statements όπου δεν υπάρχουν ήδη. Για παράδειγμα η εισαγωγή δεδομένων θα μπορούσε να υλοποιηθεί με τον εξής τρόπο :

```
$stmt = $conn->prepare("INSERT INTO login_attempts (username ,  
    password , user_agent) VALUES ( ? , ? , ? )");  
$stmt->bind_param("sss" , $username , $password , $user_agent);
```

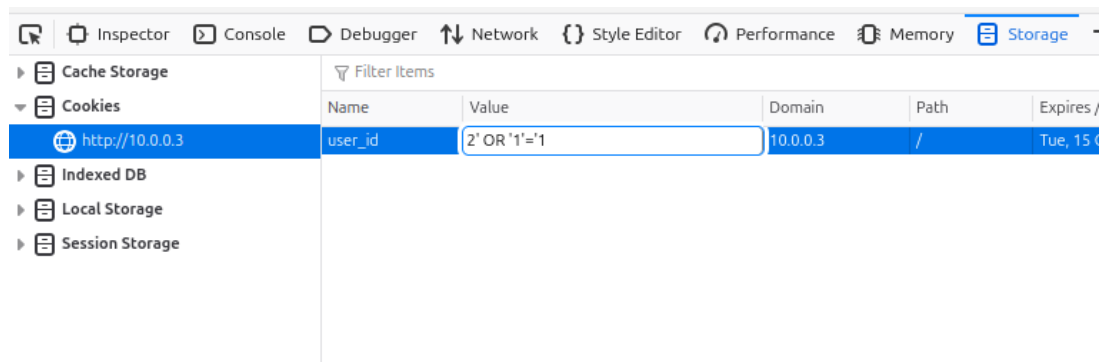
Με αυτές τις δύο αλλαγές είναι σίγουρο ότι το website δεν είναι ευάλωτο σε οποιαδήποτε μορφή επίθεσης SQL injection καθώς τα δεδομένα που δέχεται από headers είναι ασφαλή και με την χρήση των prepared statements βεβαιώνομαστε ότι καμία είσοδο από τον χρήστη δεν μπορεί να αλλάξει την δομή των SQL query που χρησιμοποιούνται για την εισαγωγή δεδομένων.

### 3.7 SQL Injection via Cookies

Για τον τελευταίο τύπο επίθεσης SQL injection δημιουργήθηκε μια log in σελίδα όπου αφού συνδεθεί ο χρήστης οδηγείτε σε μια απλή welcome σελίδα. Σε αυτή την σελίδα δημιουργείται ένα cookie για κάθε χρήστη σύμφωνα με το id του και χρησιμοποιείται για να τον καλωσορίσει στην σελίδα. Η σελίδα αυτή είναι ευάλωτη σε injection μέσω cookies επειδή η τιμή του cookie που δημιουργείται εισάγεται χωρίς κανένα sanitization στο SQL query που χρησιμοποιείται για την εκτύπωση του μηνύματος "Welcome back, \$username" όπως φαίνεται παρακάτω (π.χ. για τον χρήστη user).



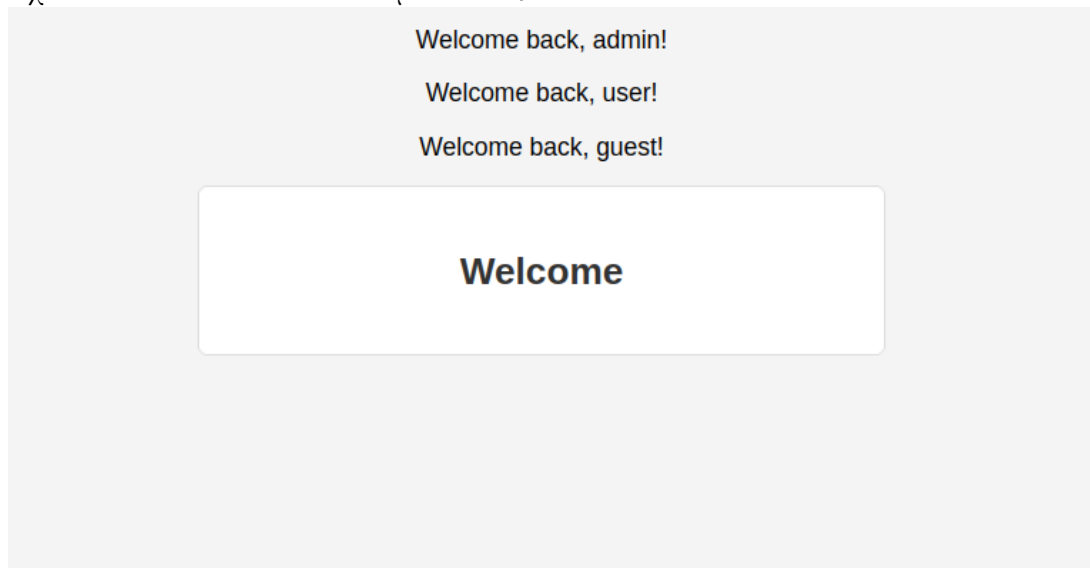
Κάποιος κακόβουλος χρήστης θα μπορούσε να παραποιήσει την τιμή του cookie και να εισάγει κακόβουλο SQL κώδικα ώστε να εξάγει πληροφορίες για όλη την βάση δεδομένων. Για παράδειγμα αν κάποιος εισάγει μια συνθήκη που ισχύει πάντα όπως '1'='1' σαν συνέχεια του αρχικού query μπορεί να κάνει την σελίδα να εκτυπώσει όλους τους χρήστες. Για να πραγματοποιηθεί αυτή η επίθεση το μόνο που χρειάζεται να κάνει ο εισβολέας είναι να αλλάξει την τιμή του cookie και να εισάγει τον κακόβουλο κώδικα με την συνεχώς αληθή συνθήκη. Η αλλαγή της τιμής του cookie μπορεί να γίνει από τα developer tools του browser (μπορούν να ανοίξουν με το κουμπί F12). Από εκεί πάμε στο στην επιλογή Στοραγε και στην συνέχεια δοκιες, εκεί θα εμφανίζονται όλα τα cookies του website και η τιμή μπορεί να αλλάξει κάνοντάς διπλό κλικ στην επιλογή value του cookie με όνομα user\_id. Τέλος για να αλλάξει σίγουρα η τιμή του cookie πρέπει να γίνει refresh της σελίδας.



Με αυτή την αλλαγή στην τιμή του cookie το SQL query παίρνει την παρακάτω μορφή :

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

Αυτό έχει ως αποτέλεσμα η βάση δεδομένων να επιστρέφει όλους τους χρήστες που περιέχει και να εκτυπώνει τα ονόματά τους.



Η επίθεση είναι εφικτή επειδή το cookie χρησιμοποιείται για την ανάκτηση του user id όπως φαίνεται στον κώδικα παρακάτω :

```
$user_id = $_COOKIE[ 'user_id' ];  
  
$sql = "SELECT * FROM users WHERE id = '$user_id'";  
$result = $conn->query( $sql );
```

Για να μην κινδυνεύει η σελίδα από επιθέσεις μέσω cookies είναι σημαντική, όπως έχει αναφερθεί ήδη, η χρήση των prepared statements γιατί διασφαλίζουν ότι δεδομένα που εισάγονται από τους χρήστες, όπως το "user\_id" που ανακτήθηκε από το cookie, συνδέονται με ασφάλεια στο query και δεν μπορούν να το τροποποιήσουν. Τέλος, είναι σημαντικό το validation των τιμών των cookies. Παρόλο που τα prepared statements βοηθούν, θα πρέπει να γίνεται και σωστό sanitization όλων των cookie πριν χρησιμοποιηθούν για να είναι σίγουρα στην σωστή τους μορφή, όπως για παράδειγμα το user\_id να έχει μόνο αριθμητικές τιμές. Με την εφαρμογή των παραπάνω ο κώδικας θα πρέπει να έχει τις παρακάτω αλλαγές :

```
if (isset($_COOKIE[ 'user_id' ]) && is_numeric($_COOKIE[ 'user_id' ])) {  
    $user_id = $_COOKIE[ 'user_id' ];  
    $stmt = $conn->prepare("SELECT * FROM users WHERE id = ?");  
    $stmt->bind_param("i", $user_id);  
    $stmt->execute();  
    $result = $stmt->get_result();
```

# ΚΕΦΑΛΑΙΟ 4

## Επίλογος

Με την αύξηση της χρήσης των web app, η ασφάλειά τους έχει γίνει βασική προτεραιότητα για πολλούς προγραμματιστές και οργανισμούς. Οι επιθέσεις SQL injection αναδεικνύουν τα κρίσιμα ευάλωτα σημεία που μπορούν να οδηγήσουν σε σοβαρές παραβιάσεις. Απλές παραλείψεις ασφάλειας μπορούν να απειλήσουν ένα ολόκληρο σύστημα, όπως αποδεικνύουν πρακτικά παραδείγματα. Για να βελτιωθεί η ασφάλεια, είναι σημαντικό να χρησιμοποιούνται λύσεις όπως prepared statements, validation εισόδου και ασφαλής χειρισμός των cookies. Καθώς οι τεχνολογίες εξελίσσονται, η προσέγγιση στην ασφάλεια πρέπει να προσαρμόζεται επίσης. Η συνεχής δοκιμές τύπου pen testing και ο τακτικός έλεγχος του κώδικα είναι απαραίτητα για την προστασία από σημερινές και μελλοντικές απειλές.

# Αναφορές

- [1] Ahmad Al-Ahmad, Belal Abu Ata, and Abdullah H. S. Wahbeh. Pen testing for web applications. *International Journal of Information Technology and Web Engineering*, July–September 2012.
- [2] Elain Barker. Recommendation for key management: Part 1: General. Technical Report NIST Special Publication 800-57 Part 1, National Institute of Standards and Technology, Gaithersburg, MD, 2016.
- [3] Matt Bishop. Education about penetration testing. Published in a collection edited by Matt Bishop and Deborah A. Frincke, 2007.
- [4] Dr. Patrick Engebretson. *The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy*. Syngress, second edition, 2013.
- [5] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso. A classification of sql injection attacks and countermeasures. In *Proceedings of the International Conference on Software Engineering (ICSE)*, College of Computing, Georgia Institute of Technology, 2006.
- [6] Bar Hofesh. Sql injection attack: How it works, examples and prevention, April 2022. Updated on January 23, 2024. Available at: <https://brightsec.com/blog/sql-injection-attack/>.
- [7] Inyong Lee, Soonki Jeong, Sang-Soo Yeo, and Jongsub Moon. A novel method for sql injection attack detection based on removing sql query attribute values. *Mathematical and Computer Modelling*, 2012.
- [8] Vincent Lui. Penetration testing: The white hat hacker. ISSA, 2007.
- [9] Arianit Maraj, Ermir Rogova, Genc Jakupi, and Xheladin Grajqevci. Testing techniques and analysis of sql injection attacks. In *Proceedings of the 2nd International Conference on Knowledge Engineering and Applications*, Prishtina, Kosovo, 2017.

- [10] Aleatha Shanley and Michael N. Johnstone. Selection of penetration testing methodologies: A comparison and evaluation. *School of Computer and Security Science, Security Research Institute, Edith Cowan University*, 2024.
- [11] Atefeh Tajpour and Mohammad JorJor zade Shooshtari. Evaluation of sql injection detection and prevention techniques. In *Proceedings of the 2010 Second International Conference on Computational Intelligence, Communication Systems and Networks*, Kuala Lumpur, Malaysia, 2010.
- [12] F. Valeur, D. Mutz, and G. Vigna. A learning-based approach to the detection of sql attacks. In *Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, Vienna, Austria, July 2005.
- [13] Clark Weissman. Penetration testing. Essay 11.