Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ασφάλεια Διαδικτυακών Εφαρμογών με τη χρήση του πλαισίου .NET** **Web Application Security using .NET Framework**
Ονοματεπώνυμο Φοιτητή	**Δημήτριος Πριάρης**
Πατρώνυμο	**Νικόλαος**
Αριθμός Μητρώου	**ΜΠΠΛ 12050**
Επιβλέπων	**Παναγιώτης Κοτζανικολάου, Λέκτορας**

Ημερομηνία Παράδοσης **Φεβρουάριος 2015**

Τριμελής Εξεταστική Επιτροπή

Παναγιώτης Κοτζανικολάου Λέκτορας	Μιχαήλ Ψαράκης Επίκουρος Καθηγητής	Κων/νος Πατσάκης Λέκτορας

Επιτελική Σύνοψη

Η παρούσα Μεταπτυχιακή Διατριβή ασχολείται με την ασφάλεια των διαδικτυακών εφαρμογών και ειδικότερα με τη μελέτη των τεχνολογιών ασφάλειας που παρέχει το πλαίσιο .NET. Από την οπτική του σχεδιαστή και του προγραμματιστή, η ασφάλεια των διαδικτυακών εφαρμογών περιλαμβάνει την ανάλυση των απαιτήσεων ασφάλειας σε διάφορα επίπεδα όπως είναι το επίπεδο του εξυπηρετητή ιστού, το επίπεδο της εφαρμογής και το επίπεδο της βάσης δεδομένων. Αρχικά παρουσιάζονται οι βασικές απαιτήσεις ασφάλειας πάνω στις οποίες πρέπει να στηρίζονται οι διαδικτυακές εφαρμογές. Στη συνέχεια αναλύονται οι κυριότερες απειλές που αντιμετωπίζουν οι διαδικτυακές εφαρμογές, σύμφωνα με το μοντέλο STRIDE. Επιπλέον, γίνεται ανάλυση των τεχνολογιών και των βιβλιοθηκών ασφάλειας που προσφέρει το πλαίσιο .NET, ενώ για κάθε υπηρεσία ασφάλειας αναλύονται οι εναλλακτικές τεχνολογίες που παρέχει το πλαίσιο. Στη τελική φάση της εργασίας αναπτύχθηκε μια δοκιμαστική διαδικτυακή εφαρμογή με τη χρήση του πλαισίου .NET, στην οποία εφαρμόζονται τεχνολογίες ασφάλειας που προσφέρει το πλαίσιο, με στόχο την ικανοποίηση των απαιτήσεων ασφάλειας που έχουν τεθεί.

Abstract

This thesis is concerned with web application security and more specifically with the study of the security technologies offered by the .NET framework. From the perspective of the web designers and developers, web application security includes the analysis of the security requirements at several layers, such as the web server, the application and the database server. Initially, the fundamental security requirements of the web applications are presented. Then, the most important security threats against web applications are analyzed, according to the STRIDE model. Furthermore, an analysis of the security libraries and security technologies of the .NET framework are presented; for each examined security service, the alternative security technologies offered by the .NET framework are analyzed. In the final stage of this thesis, a test web application was developed using the .NET framework, in which several security technologies are applied, in order to meet the security requirements set from the analysis phase.

Πίνακας περιεχομένων

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΠΙΝΑΚΑΣ ΠΙΝΑΚΩΝ

Κεφάλαιο 1°

1 Εισαγωγή

1.1 Ασφάλεια Διαδικτυακών Εφαρμογών

Η ασφάλεια διαδικτυακών εφαρμογών είναι ο κλάδος της Ασφάλειας Πληροφοριών που ασχολείται με την ασφάλεια ιστοσελίδων, εφαρμογών Διαδικτύου και υπηρεσιών Διαδικτύου. Βασίζεται δηλαδή στις αρχές που διέπουν την ασφάλεια εφαρμογών οι οποίες όμως προσανατολίζονται στο Διαδίκτυο και στα συστήματα που λειτουργούν σε αυτό.

1.2 Περιγραφή του υπό μελέτη προβλήματος

Το πρόβλημα που μελετάται στην εργασία είναι το πώς μπορούμε να σχεδιάσουμε και να υλοποιήσουμε μια εφαρμογή Διαδικτύου, η οποία να προστατεύεται από επιθέσεις που είναι δυνατόν να πραγματοποιηθούν εναντίον τέτοιου είδους εφαρμογών. Πιο συγκεκριμένα, παρουσιάζεται ο τρόπος με τον οποίο το πλαίσιο .NET [1] λειτουργεί για να διαχειριστεί θέματα ασφάλειας, τι προβλήματα μπορεί να επιλύσει και με ποια εργαλεία. Αναλύονται επίσης τα μέτρα προστασίας που μπορούμε να πάρουμε προκειμένου να ασφαλίσουμε μια διαδικτυακή εφαρμογή πάντα εντός πλαισίου που καθορίζει το .NET.

Για την καλύτερη μελέτη του προβλήματος της ασφάλειας διαδικτυακών εφαρμογών με τη χρήση του πλαισίου .NET, υλοποιήθηκε μια διαδικτυακή εφαρμογή παρακολούθησης κινηματογραφικών ταινιών. Το λειτουργικό σύστημα πάνω στο οποίο αναπτύχθηκε η εφαρμογή ήταν το Windows 8. Τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής ήταν το .NET Framework όπως αναφέρθηκε, μέσω του περιβάλλοντος ανάπτυξης (Integrated Development Environment) Visual Studio 2012 [2] που προσφέρει η Microsoft. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε και η οποία τρέχει στον εξυπηρετητή (server side) είναι η C#[3] ενώ ο server στον οποίο εγκαταστάθηκε η εφαρμογή ήταν ο Internet Information Services (IIS) [4]. Επιπλέον χρησιμοποιήθηκε το πρωτόκολλο Transport Layer Security (TSL) [5] για ασφάλεια στην επικοινωνία με χρήση κρυπτογραφίας μεταξύ του server και των περιηγητών (browsers). Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το μοντέλο ανάπτυξης διαδικτυακών εφαρμογών Model/View/Controller (MVC) [6].

1.3 Στόχοι της εργασίας

Οι στόχοι της εργασίας περιλαμβάνουν τα ακόλουθα:

1. Θεωρητική μελέτη των απαιτήσεων ασφάλειας που αφορούν τις διαδικτυακές εφαρμογές.
2. Θεωρητική μελέτη του πλαισίου .NET από τη σκοπιά της ασφάλειας Διαδικτυακών εφαρμογών.
3. Μελέτη των βιβλιοθηκών που παρέχονται από το πλαίσιο .NET και που σχετίζονται με την ασφάλεια διαδικτυακών εφαρμογών.
4. Μελέτη του τρόπου με τον οποίο οι βιβλιοθήκες ασφαλείας του πλαισίου .NET χρησιμοποιούνται για την προστασία των διαδικτυακών εφαρμογών.
5. Χρήση των εργαλείων που παρέχονται από το .NET Framework για την υλοποίηση μιας ασφαλής Διαδικτυακής εφαρμογής.
6. Εγκατάσταση της εφαρμογής σε εργαστηριακό περιβάλλον έτσι ώστε να μπορεί να ελεγχθεί και να αξιολογηθεί η ασφάλεια της.

1.4 Παραδοτέα της εργασίας

Μαζί με το κείμενο της εργασίας παρέχεται και cd το οποίο περιέχει:

i. Κατάλογο «MoviesOnlineProject» ο οποίος περιλαμβάνει τον κώδικα που δημιουργήθηκε από το Visual Studio (project) για την υλοποίηση της εφαρμογής. Στον υποκατάλογο «MoviesOnlineProject\MoviesOnline\App_Data», υπάρχουν τα αρχεία των βάσεων δεδομένων που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής.
ii. Κατάλογο «SecurityGuard-SourceCode» με τον πηγαίο κώδικα του πακέτου διαχείρισης ταυτοτήτων που χρησιμοποιήθηκε στην εφαρμογή.
iii. Κατάλογο «Certificates» με τα πιστοποιητικά και τα ιδιωτικά κλειδιά τους που χρησιμοποιήθηκαν για τη λειτουργία της εφαρμογής.
iv. Αρχείο που περιέχει το παρόν κείμενο σε ηλεκτρονική μορφή.

1.5 Δομή της εργασίας

Το έντυπο κείμενο της εργασίας αποτελείται από 10 Κεφάλαια. Στο 1^{ο} Κεφάλαιο αναλύεται το πρόβλημα της ασφάλειας των διαδικτυακών εφαρμογών, αναλύεται ο σκοπός και οι στόχοι της εργασίας, παρουσιάζεται η δομή της και το πλάνο υλοποίησής της. Στο 2^{ο} Κεφάλαιο παρουσιάζονται οι απαραίτητες τεχνολογίες και δίνονται οι απαιτούμενοι ορισμοί που πρέπει να είναι γνωστά για την κατανόηση της εργασίας. Στο 3^{ο} Κεφάλαιο παρουσιάζονται οι απειλές κατά της ασφάλειας των διαδικτυακών εφαρμογών και οι θεμελιώδεις απαιτήσεις ασφάλειας που πρέπει αυτές να ικανοποιούν. Παρουσιάζονται επίσης οι υπηρεσίες ασφάλειας και πώς αυτές θα πρέπει να εφαρμόζονται ανά επίπεδο για τη θωράκιση των διαδικτυακών εφαρμογών. Στο 4^{ο} Κεφάλαιο περιγράφονται οι τεχνολογίες ασφάλειας που προσφέρει το πλαίσιο .NET για την προστασία των διαδικτυακών εφαρμογών από τις απειλές που αναλύθηκαν στο Κεφάλαιο 3. Στο 5^{ο} Κεφάλαιο καθορίζονται οι λειτουργικές απαιτήσεις της εφαρμογής όπως και οι απαιτήσεις ασφάλειας που θα πρέπει να ικανοποιεί. Στο 6^{ο} Κεφάλαιο αναλύονται οι τρόποι με τους οποίους υλοποιήθηκαν οι λειτουργικές απαιτήσεις της εφαρμογής, ενώ στο 7^{ο}, αναλύονται οι τρόποι υλοποίησης των απαιτήσεων ασφάλειας χρησιμοποιώντας τα εργαλεία που παρουσιάστηκαν στα Κεφάλαια 3 και 4. Στο 8^{ο} Κεφάλαιο γίνεται η παρουσίαση της εγκατάστασης της εφαρμογής στον IIS server. Στο 9^{ο} Κεφάλαιο αναφέρονται τα συμπεράσματα που εξήχθησαν από την παρούσα διατριβή και στο 10^{ο} παρατίθεται η βιβλιογραφία που χρησιμοποιήθηκε για την εκπόνηση της διατριβής.

1.6 Πλάνο υλοποίησης της Διατριβής

Για την ολοκλήρωση της διατριβής ακολουθήθηκε το πλάνο που φαίνεται στον Πίνακα 1.

Παραδοτέο	Περιγραφή	Ημερομηνία παράδοσης
Π1	Πρόταση εκπόνησης διατριβής	4/06/2014
Π2 (Κεφάλαια 2, 3, 4)	Θεωρητική μελέτη ασφάλειας διαδικτυακών εφαρμογών και τεχνολογιών που προσφέρονται από το .NET	30/07/2014
Π3.1 (Κεφάλαια 5,6) Π3.2	Ανάπτυξη εφαρμογής. Υλοποίηση λειτουργικών απαιτήσεων	30/08/2014

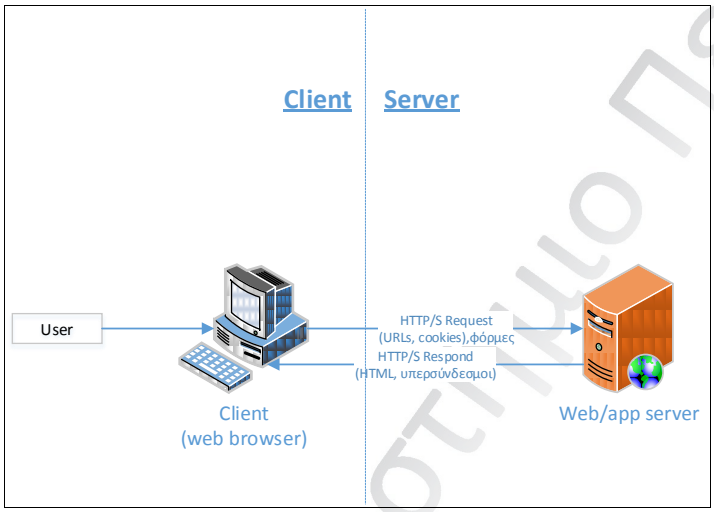
(1η έκδοση της εφαρμογής)		
Π4.1 **(Κεφάλαιο 7)** **Π4.2** **(2η έκδοση της εφαρμογής)**	Ανάλυση επιπλέον μέτρων προστασίας εφαρμογής. Υλοποίηση απαιτήσεων ασφαλείας.	30/09/2014
Π5.1 **(Κεφάλαιο 8)** **Π5.2** **(Τελική έκδοση εφαρμογής)**	Δημιουργία εργαστηριακού περιβάλλοντος. Εγκατάσταση εφαρμογής στον IIS Server.	30/10/2014
Π6	Ολοκληρωμένη Έκδοση διατριβής	30/11/2014
Π7	Διορθωμένη Έκδοση διατριβής	20/12/2014
Π8	Τελική Έκδοση διατριβής	30/1/2015

Πίνακας 1. Πλάνο υλοποίησης Μεταπτυχιακής διατριβής

Κεφάλαιο 2°

2 Αρχιτεκτονικές Διαδικτυακών Εφαρμογών

Το Διαδίκτυο στηρίζεται στην αρχιτεκτονική πελάτη/εξυπηρετητή (client/server). Κάθε διαδικτυακή εφαρμογή λειτουργεί με βάση αυτό το πρότυπο και καθορίζει ότι τα συνεργαζόμενα κομμάτια της εφαρμογής μπορούν να κατηγοριοποιηθούν ως client ή ως server. Η client εφαρμογή ζητά υπηρεσίες και δεδομένα από το server και η εφαρμογή server απαντάει στις αιτήσεις του client. Στην περίπτωση που ένας client επικοινωνεί με το web server χρησιμοποιώντας κάποιο πρωτόκολλο (π.χ. HTTP) κάνοντας αίτηση για δεδομένα ή υπηρεσία, ο web server λαμβάνει την αίτηση του web browser και φροντίζει να την εξυπηρετήσει στέλνοντας τα δεδομένα (πχ στέλνοντας ένα HTML αρχείο στον browser) ή τρέχοντας κάποιο πρόγραμμα στο server και αποστέλλοντας τα αποτελέσματα κάποιας επεξεργασίας (πχ η άντληση δεδομένων από μία βάση δεδομένων) [7].



Εικόνα 1. Αρχιτεκτονική client/server

Για την ευκολότερη κατανόηση των τεχνολογιών που χρησιμοποιούνται κατά την εκτέλεση μιας Διαδικτυακής εφαρμογής θα διαχωριστούν αυτές σε τεχνολογίες που χρησιμοποιούνται κατά την επικοινωνία μεταξύ clients και servers, τεχνολογίες που χρησιμοποιούνται στην πλευρά του client (client side) και τεχνολογίες που χρησιμοποιούνται στην πλευρά του server (server side).

2.1 Τεχνολογίες επικοινωνίας μεταξύ clients και servers

2.1.1 Πρωτόκολλο HTTP

Το HTTP [8] είναι το βασικό πρωτόκολλο επικοινωνίας που χρησιμοποιείται για την πρόσβαση σε όλες τις διαδικτυακές εφαρμογές. Το πρωτόκολλο χρησιμοποιεί μηνύματα που ανταλλάσσονται μεταξύ clients και servers. Ο client στέλνει ένα μήνυμα αιτήματος (request) και στη συνέχεια ο server απαντάει με ένα μήνυμα απόκρισης (response).

2.1.2 Uniform Resource Locators (URLs)

Ένα URL είναι ένα μοναδικό αναγνωριστικό για έναν πόρο Ιστού μέσω του οποίου μπορεί να ανακτηθεί ο συγκεκριμένος πόρος. Η μορφή του είναι η παρακάτω:

πρωτόκολλο://όνομα_Host[:θύρα]/[διαδρομή/]αρχείο[?παράμετρος=τιμή]

2.1.3 Cookie

Είναι ένα μικρό κομμάτι δεδομένων που αποστέλλεται από έναν ιστότοπο και αποθηκεύεται στον browser του χρήστη για όσο αυτός βρίσκεται στον ιστότοπο. Κάθε φορά που ο χρήστης φορτώνει κάποια ιστοσελίδα του ιστότοπου, ο browser επιστρέφει το cookie στον server για να τον ενημερώσει για την προηγούμενη δραστηριότητά του σχετικά με τον ιστότοπο (π.χ. Ποίο είναι το όνομα χρήστη, ποιες σελίδες έχει επισκεφθεί πιο συχνά, τι προϊόντα είχε στο καλάθι αγορών του κατά την τελευταία συνεδρία κτλ.) Αποτελεί έναν αξιόπιστο μηχανισμό με τον οποίο οι ιστότοποι ‘’θυμούνται’’ πληροφορίες κατάστασης.

2.1.4 Secure Socket Layer (SSL) Πρωτόκολλο

Πρωτόκολλο που χρησιμοποιεί συνδυασμό συμμετρικής και ασύμμετρης κρυπτογραφίας. Χρησιμοποιείται για τη δημιουργία κρυπτογραφημένου καναλιού μεταξύ δύο συσκευών συνδεδεμένων μέσω δικτύου.

2.2 Τεχνολογίες Client Side

Μια διαδικτυακή εφαρμογή λαμβάνει δεδομένα εισόδου από τους χρήστες και με αυτά αποκρίνεται με το ανάλογο αποτέλεσμα. Το αποτέλεσμα αυτό είναι ορατό από τη διεπαφή χρήστη στην πλευρά του client, δηλαδή τον web browser. Οι τεχνολογίες που χρησιμοποιούνται από την πλευρά του client αναλύονται στις επόμενες παραγράφους.

2.2.1 HyperText Markup Language (HTML)

Μια από τις σημαντικότερες τεχνολογίες που χρησιμοποιούνται για τη δημιουργία διεπαφών ιστού είναι η γλώσσα HTML. Είναι μια γλώσσα που βασίζεται σε ετικέτες (tags) οι οποίες περιγράφουν τη δομή των εγγράφων που εμφανίζουν οι browsers.

2.2.2 Υπερσύνδεσμοι

Ένα μεγάλο μέρος της επικοινωνίας ενός client με τον server καθοδηγείται με την επιλογή από τον χρήστη κάποιου υπερσυνδέσμου. Συνήθως οι υπερσύνδεσμοι περιέχουν προκαθορισμένες παραμέτρους αίτησης ιστοσελίδας, δηλαδή δεδομένα που δεν εισάγονται από τον χρήστη κάθε φορά αλλά υποβάλλονται προς τον server γιατί αυτός τα τοποθέτησε στο URL του υπερσυνδέσμου.

2.2.3 Φόρμες επικοινωνίας HTML

Στις περισσότερες διαδικτυακές εφαρμογές απαιτείται ευελιξία στη συγκέντρωση δεδομένων εισόδου από τον χρήστη. Οι φόρμες HTML είναι ο πιο συνηθισμένος τρόπος που χρησιμοποιείται για να επιτρέψει στους χρήστες να εισάγουν δεδομένα μέσω του browser.

2.2.4 JavaScript

Η JavaScript είναι μια απλή γλώσσα προγραμματισμού που χρησιμοποιείται εύκολα για την βελτίωση των διεπαφών ιστού με τρόπους που δεν είναι εφικτοί χρησιμοποιώντας μόνο HTML.

2.2.5 Asynchronous JavaScript and XML (AJAX)

Μια πολύ σημαντική εξέλιξη στη χρήση της JavaScript είναι η τεχνική AJAX που χρησιμοποιείται για τη περαιτέρω βελτίωση των διεπαφών ιστού. Η τεχνική αυτή περιλαμβάνει την έκδοση δυναμικών αιτημάτων HTTP μέσα από μια σελίδα HTML με σκοπό την ανταλλαγή δεδομένων με τον server και την ανανέωση της τρέχουσας ιστοσελίδας χωρίς την εξ' ολοκλήρου επαναφόρτωσή της.

2.3 Τεχνολογίες Server Side

Το περιεχόμενο που παρουσιάζουν οι διαδικτυακές εφαρμογές στους χρήστες δημιουργείται δυναμικά. Αυτό σημαίνει ότι όταν ένας χρήστης ζητάει κάποια ιστοσελίδα, η απόκριση από τον server δημιουργείται απευθείας τη συγκεκριμένη στιγμή και κάθε χρήστης μπορεί να λάβει περιεχόμενο προσανατολισμένο αποκλειστικά σε αυτόν. Το δυναμικό περιεχόμενο δημιουργείται με την εκτέλεση κάποιου κώδικα στον server.

Οι διαδικτυακές εφαρμογές χρησιμοποιούν ένα μεγάλο εύρος τεχνολογιών στην πλευρά του server για να μπορέσουν να προσφέρουν την λειτουργικότητά τους. Μερικές από αυτές που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής της παρούσας εργασίας αναλύονται στις παρακάτω παραγράφους.

2.3.1 Γλώσσα προγραμματισμού server side

Για τη δυναμική δημιουργία του περιεχομένου που προσφέρουν οι διαδικτυακές εφαρμογές χρησιμοποιούνται γλώσσες προγραμματισμού όπως οι PHP, ASP.NET, Java, ColdFusion, Ruby, Perl, Python, οι οποίες εκτελούνται στους web ή application servers.

2.3.2 Web server

Ένας web server είναι ο υπολογιστής ο οποίος όταν λάβει ένα αίτημα HTTP, αποκρίνεται αποστέλλοντας ένα αρχείο HTML ή μια εικόνα. Όταν επεξεργάζεται το αίτημα, εκτός από την επιλογή να αποστείλει το αρχείο HTML ή μια εικόνα, μπορεί να απαντήσει στέλνοντας μια ανακατεύθυνση ή να αναθέσει την δημιουργία δυναμικού περιεχομένου σε κάποιο πρόγραμμα όπως π.χ. JavaServer Pages (JSP), servlets, ASPs ή κάποια άλλη τεχνολογία server side. Η μόνη λειτουργικότητα δηλαδή που έχει είναι η παροχή περιβάλλοντος στο οποίο εκτελείται μια τεχνολογία server side η οποία δημιουργεί τις αποκρίσεις που πρόκειται αυτός να στείλει.

2.3.3 Application server

Ο application server είναι ο τύπος εξυπηρετητή που προσφέρει επιχειρησιακή λογική σε εφαρμογές clients. Ενώ ο web server ασχολείται κυρίως με την αποστολή αρχείων HTML για προβολή στους χρήστες (clients), ο application server παρέχει πρόσβαση σε λειτουργίες που έχουν να κάνουν με τη επιχειρησιακή λογική. Οι πληροφορίες δηλαδή που ανταλλάσσονται μεταξύ ενός application server και του client δεν είναι μόνο για σκοπούς προβολής HTML και γενικότερα markup. Σημειώνεται ότι σε κάποιες περιπτώσεις ένας application server μπορεί να περιέχει έναν web server με αποτέλεσμα η διαφοροποίησή τους να μην είναι διακριτή.

2.3.4 Αρχιτεκτονική Model-View-Controller (MVC)

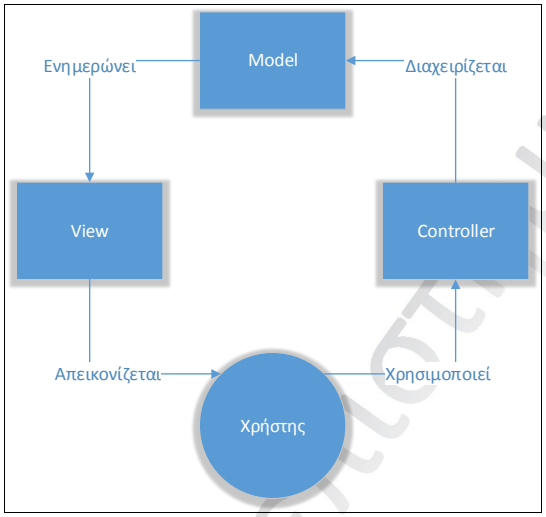
Το Model–view–controller είναι ένα μοντέλο αρχιτεκτονικής λογισμικού το οποίο χρησιμοποιείται για την δημιουργία εφαρμογών. Παρέχει έναν τρόπο ανάπτυξης λογισμικού που βασίζεται στο διαχωρισμό των εννοιών μοντέλου δεδομένων (model), εμφάνισης (view) και λειτουργιών (controller). Ο σαφής διαχωρισμός αυτών των εννοιών προσθέτει μια μικρή πολυπλοκότητα στις εφαρμογές που αναπτύσσονται με αυτήν την αρχιτεκτονική, αλλά το κέρδος που αποκομίζεται ισοσταθμίζει αυτήν την πολυπλοκότητα.

Με την αρχιτεκτονική MVC διαχωρίζεται η διεπαφή χρήστη μιας εφαρμογής σε τρία συστατικά:

- *Model*: Αποτελείται συνήθως από ένα σύνολο κλάσεων που περιγράφει τα δεδομένα με τα οποία δουλεύει η εφαρμογή καθώς επίσης και τους κανόνες που χρησιμοποιούνται για την χρήση ή την τροποποίηση αυτών.
- *View*: Καθορίζει το πώς θα φαίνεται η διεπαφή στον χρήστη.
- *Controller*: Αποτελείται από ένα σύνολο κλάσεων που διαχειρίζονται την επικοινωνία με τον χρήστη και τη συνολική ροή της εφαρμογής.

Αξίζει να σημειωθεί ότι η αρχιτεκτονική MVC προσφέρει μια λύση για την ανάπτυξη διεπαφής χρήστη και όχι για το πώς θα διαχειριστούν τα υπόλοιπα θέματα που αφορούν μια εφαρμογή όπως για παράδειγμα ο τρόπος πρόσβασης στα δεδομένα, η αλληλεπίδραση υπηρεσιών κ.α.

Το πώς πρέπει να συνεργάζονται τα τρία συστατικά παρουσιάζεται στην παρακάτω εικόνα:

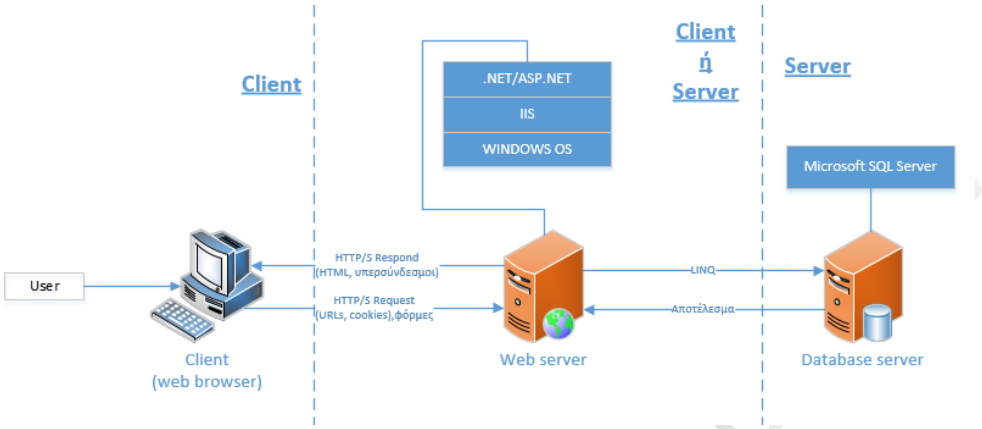


Εικόνα 2. Αλληλεπίδραση στοιχείων αρχιτεκτονικής MVC

2.4 Τεχνολογίες Ανάπτυξης Διαδικτυακής Εφαρμογής .NET

2.4.1 Αρχιτεκτονική τριών επιπέδων

Η γενική αρχιτεκτονική που χρησιμοποιείται για τη σχεδίαση διαδικτυακών εφαρμογών είναι η client/server και πιο συγκεκριμένα αυτή των πολλαπλών βαθμίδων (multi-tier) [7]. Στην Εικόνα 3 φαίνεται ένα παράδειγμα αρχιτεκτονικής σχεδίασης διαδικτυακής εφαρμογής τριών βαθμίδων [9]. Η συγκεκριμένη αρχιτεκτονική χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής της παρούσας εργασίας και είναι προσανατολισμένη στο .NET Framework.



Εικόνα 3. Αρχιτεκτονική τριών (3) επιπέδων διαδικτυακών εφαρμογών .NET

Οι έννοιες που πρέπει να είναι γνωστές για την κατανόηση της παραπάνω αρχιτεκτονικής και κατ' επέκταση της εργασίας, αναλύονται στις επόμενες παραγράφους.

2.4.2 Internet Information Services (IIS) Server

Ο IIS server είναι ένα εξάρτημα του λειτουργικού συστήματος Windows ή Windows Server το οποίο παρέχει λειτουργίες web server και application server. Χρησιμοποιείται για τη φιλοξενία ιστότοπων, υπηρεσιών ή εφαρμογών Διαδικτύου και παρέχει μια πλατφόρμα μέσω της οποίας μπορούν να διαμοιραστούν πληροφορίες μεταξύ χρηστών σε οποιοδήποτε δίκτυο.

Οι πληροφορίες για την ρύθμιση του IIS server είναι αποθηκευμένες σε Extensible Markup Language (XML) αρχεία. Τα δύο αρχεία που ελέγχουν τις ρυθμίσεις του είναι τα applicationhost.config και web.config. Μέσω του web.config ελέγχεται η διαμόρφωσή του σε επίπεδο εφαρμογής ενώ μέσω του applicationhost.config ελέγχεται αποκλειστικά αυτός. Οι ρυθμίσεις διαμόρφωσηςhς κληρονομούνται, άρα οι ρυθμίσεις που καθορίζονται στο web.config μπορούν να παρακάμψουν τις ρυθμίσεις του υψηλότερου επιπέδου, δηλαδή του αρχείου applicationhost.config. Το αρχείο applicationhost.config βρίσκεται στο κατάλογο %windir%\system32\inetsrv\config ενώ τα web.config στους υποκαταλόγους της εφαρμογής.

2.4.3 .NET Framework

Το .NET Framework είναι ένα πλαίσιο εργασίας το οποίο περιλαμβάνει ένα σύνολο τεχνολογιών που περιγράφονται παρακάτω:

Γλώσσες προγραμματισμού .NET: Οι γλώσσες που χρησιμοποιούνται στο πλαίσιο .NET είναι οι C#, Visual Basic .NET (VB .NET) οι οποίες περιλαμβάνουν τις Jscript .NET (έκδοση server-side της JavaScript), J# και C++.

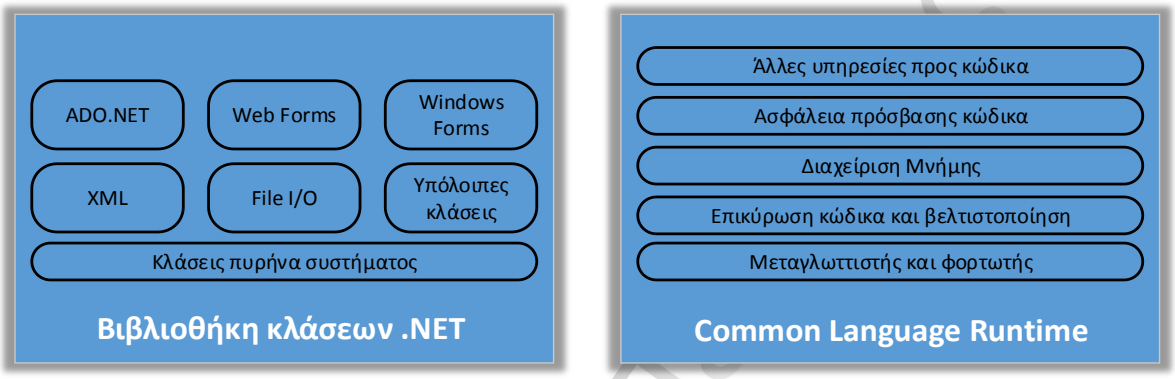
Common Language Runtime (CLR): Η μηχανή που εκτελεί όλα τα προγράμματα που χρησιμοποιούν το .NET, το οποίο παρέχει αυτόματα υπηρεσίες στις εφαρμογές αυτές όπως έλεγχο ασφάλειας, διαχείριση μνήμης και βελτιστοποίηση.

Βιβλιοθήκη κλάσεων .NET: Στη βιβλιοθήκη αυτή συλλέγονται χιλιάδες έτοιμα κομμάτια κώδικα που παρέχουν λειτουργικότητα προς χρήση από τις εφαρμογές που αναπτύσσονται. Τα χαρακτηριστικά αυτά οργανώνονται σε ομάδες ανάλογα με την τεχνολογία τους (π.χ. ADO.NET, τεχνολογία για την δημιουργία εφαρμογών που χρησιμοποιούν βάσεις δεδομένων).

ASP.NET: Η μηχανή πάνω στην οποία εκτελούνται οι εφαρμογές και οι υπηρεσίες ιστού. Χρησιμοποιεί σχεδόν όλα τα χαρακτηριστικά της βιβλιοθήκης κλάσεων του .NET Framework καθώς επίσης και ένα σύνολο από ειδικές υπηρεσίες προσανατολισμένες σε εφαρμογές ιστού.

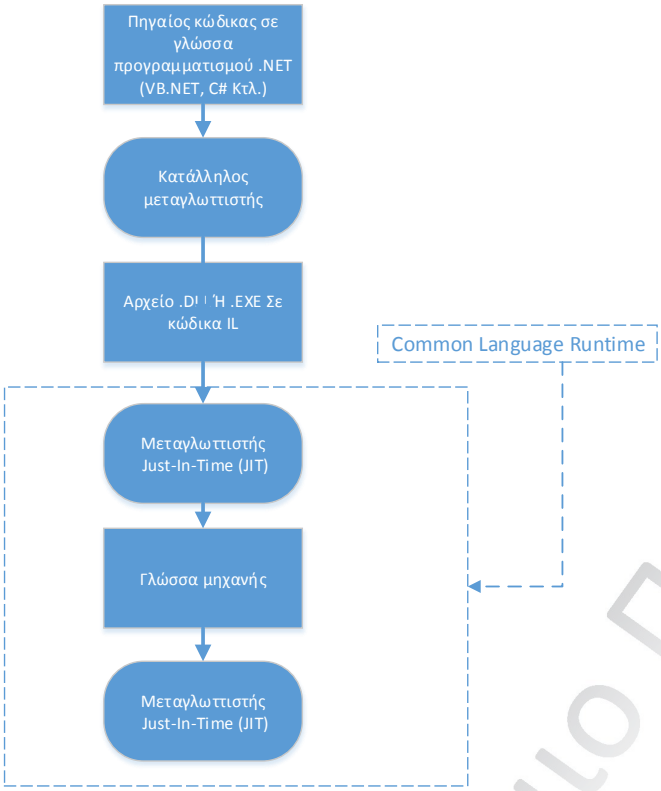
Visual Studio: Ένα προαιρετικό εργαλείο ανάπτυξης εφαρμογών που περιέχει πλούσια γκάμα λειτουργιών παραγωγής εφαρμογών και εντοπισμού σφαλμάτων.

Στην παρακάτω εικόνα φαίνονται τα δύο θεμελιώδη κομμάτια του .NET Framework.



Εικόνα 4. Βιβλιοθήκη κλάσεων .NET και CLR

Οι γλώσσες προγραμματισμού που χρησιμοποιούνται στο .NET μεταγλωττίζονται σε μια άλλη γλώσσα χαμηλότερου επιπέδου και στη συνέχεια εκτελείται ο κώδικας από το CLR. Η γλώσσα αυτή ονομάζεται Ενδιάμεση Γλώσσα (Microsoft Intermediate Language – MSIL ή IL) και είναι η μόνη που μπορεί να χρησιμοποιήσει το CLR. Στο παρακάτω σχεδιάγραμμα φαίνεται ο τρόπος με τον οποίο οι γλώσσες .NET μεταγλωττίζονται σε IL. Οποιοδήποτε αρχείο *.exe ή *.dll που χτίζεται με γλώσσα .NET περιέχει κώδικα IL. Αυτά είναι και τα αρχεία που αναπτύσσονται σε άλλους υπολογιστές ώστε να μπορεί να εκτελείται η εφαρμογή από αυτούς, ανεξάρτητα από τον πηγαίο κώδικά της.



Εικόνα 5. Μεταγλώττιση στο .NET

2.4.4 ASP.NET

Το ASP.NET είναι το πλαίσιο εργασίας που χρησιμοποιείται από τους προγραμματιστές για την υλοποίησης εφαρμογών ιστού και δυναμικών ιστοσελίδων. Είναι χτισμένο πάνω στο CLR του .NET και επιτρέπει τη συγγραφή κώδικα σε οποιαδήποτε γλώσσα προγραμματισμού .NET. Στο ASP.NET μπορούν να χρησιμοποιηθούν στοιχεία ελέγχου όπως π.χ. κουμπιά, ετικέτες, κτλ., υπό μορφή αντικειμένων της γλώσσας προγραμματισμού. Αυτό σημαίνει ότι κατά τη φάση της μεταγλώττισης της εφαρμογής, τα αντικείμενα αυτά θα δημιουργήσουν αυτόματα τον κώδικα HTML που απαιτείται.

Μια εφαρμογή ASP.NET αποτελείται από ένα συνδυασμό αρχείων, ιστοσελίδων και εκτελέσιμου κώδικα που μπορεί να προσπελαστεί από ένα εικονικό κατάλογο ενός server. Άρα ο κατάλογος αυτός αποτελεί τη βασική δομή που οριοθετεί μια διαδικτυακή εφαρμογή.

Οι κυριότεροι τύποι αρχείων που μπορούν περιέχονται σε ASP.NET εφαρμογές περιγράφονται παρακάτω.

- Αρχεία με κατάληξη .aspx : Ιστοσελίδα ASP.NET. Τα αρχεία αυτά περιέχουν τη διεπαφή χρήστη της ιστοσελίδας και τον κώδικα της εφαρμογής που εκτελείται στο υπόβαθρο. Οι χρήστες αποστέλλουν αιτήσεις στον server για να λάβουν κάποια από αυτές και να ξεκινήσει η εφαρμογή.
- Αρχείο global.asax : Είναι το καθολικό αρχείο της εφαρμογής. Μπορεί να χρησιμοποιηθεί για να καθοριστούν καθολικές μεταβλητές οι οποίες μπορούν να προσπελαστούν από οποιοδήποτε αρχείο της εφαρμογής.

- Αρχεία με κατάληξη .cs : Λέγονται και αρχεία code-behind, αφού περιέχουν τον κώδικα της γλώσσας προγραμματισμού .NET. Η χρήση αυτών των αρχείων επιτρέπει το διαχωρισμού του κώδικα που αφορά την υλοποίηση της διεπαφής χρήστη από τον υπόλοιπο κώδικα μιας ιστοσελίδας.
- Αρχείο web.config : Αρχείο διαμόρφωσης της εφαρμογής, σε XML μορφή. Σε αυτό περιέχονται οι ρυθμίσεις για την ασφάλεια της εφαρμογής, τη διαχείριση μνήμης και κατάστασης.

Δομή καταλόγων μιας εφαρμογής ASP.NET

Κάθε εφαρμογή ASP.NET θα πρέπει να χαρακτηρίζεται από μια καλά οργανωμένη δομή καταλόγων. Εκτός από τη δομή καταλόγων που δημιουργεί ο προγραμματιστής, το ASP.NET χρησιμοποιεί κάποιους ειδικούς υποκαταλόγους μέσα στη δομή της εφαρμογής που περιγράφονται παρακάτω.

- Κατάλογος Bin : Περιέχει όλα τα μεταγλωττισμένα .NET στοιχεία (DLLs) που χρησιμοποιεί η εφαρμογή. Για παράδειγμα, αν αναπτυχθεί κάποιο εξάρτημα βάσης δεδομένων, αυτό θα τοποθετηθεί σε αυτόν τον φάκελο. Το ASP.NET θα το εντοπίσει αυτόματα και θα μπορεί να το χρησιμοποιήσει οποιαδήποτε ιστοσελίδα της εφαρμογής.
- Κατάλογος App_Data : Ο κατάλογος αυτός χρησιμοποιείται συνήθως για την αποθήκευση δεδομένων όπως αρχεία βάσης δεδομένων SQL Server και XML αρχεία.

Σε μια τυπική εφαρμογή δεν είναι απαραίτητο να υπάρχουν όλοι. Στις περισσότερες περιπτώσεις που γίνεται χρήση του Visual Studio, αυτό προτείνει την δημιουργία τους εφόσον απαιτηθεί.

Ρυθμίσεις ASP.NET

Το σύστημα ρυθμίσεων του ASP.NET είναι δομημένο με τρόπο που επιτρέπει μεγάλη επεκτασιμότητα. Οι πληροφορίες ρυθμίσεων αποθηκεύονται σε αρχεία μορφής XML, τα οποία επεξεργάζονται με οποιονδήποτε επεξεργαστή κειμένου. Επίσης, πολλά αρχεία ρυθμίσεων με όνομα web.config μπορούν να υπάρχουν σε πολλούς υποκαταλόγους εντός μιας εφαρμογής ιστού. Το καθένα από αυτά, εφαρμόζει τις ρυθμίσεις που περιγράφονται σε αυτό, στα στοιχεία (αρχεία και υποκατάλογοι) του καταλόγου που βρίσκεται.

Τα αρχεία ρύθμισης που βρίσκονται σε καταλόγους «παιδιά» μπορούν να:

- Παρέχουν επιπρόσθετες ρυθμίσεις αυτών που έχουν κληρονομηθεί από το αρχείο ρύθμισης «γονέα».
- Παρακάμψουν ή να τροποποιήσουν τις ρυθμίσεις που έχουν καθοριστεί σε κατάλογο «γονέα».

Το αρχείο machine.config που βρίσκεται στον κατάλογο systemroot\Microsoft.NET\Framework\versionNumber\CONFIG\ είναι το αρχείο ρυθμίσεων που βρίσκεται στην υψηλότερη θέση στο ιεραρχικό μοντέλο που περιγράφηκε και παρέχει και τις ρυθμίσεις για το σύνολο των λειτουργιών του ASP.NET.

Κατά τη διάρκεια εκτέλεσης μιας εφαρμογής, το ASP.NET χρησιμοποιεί τις πληροφορίες ρυθμίσεων που υπάρχουν στα αρχεία web.config σύμφωνα με την ιεραρχική δομή του εικονικού καταλόγου της εφαρμογής, για να υπολογίσει το σύνολο των ρυθμίσεων που πρέπει να εφαρμόσει σε κάθε αίτημα χρήστη.

Το ASP.NET μπορεί και εντοπίζει αυτόματα αλλαγές στα αρχεία ρυθμίσεων, τις οποίες και εφαρμόζει χωρίς να απαιτείται επανεκκίνηση του server.

Για την προστασία των αρχείων ρυθμίσεων από εξωτερική πρόσβαση, αρκεί να ρυθμιστεί ο IIS με τρόπο που να απαγορεύει την απευθείας περιήγηση σε αυτά. Σφάλμα HTTP 403 (forbidden) θα επιστραφεί εφόσον γίνει προσπάθεια περιήγησης σε αρχείο ρυθμίσεων.

2.4.5 Language Integrated Query (LINQ)

Τα ερωτήματα που γίνονται σε βάση δεδομένων μέσω του πλαισίου Entity Framework είναι γραμμένα σε γλώσσα LINQ. Η συγκεκριμένη γλώσσα είναι στενά συνδεδεμένη με το .NET Framework και αποτελεί μια ισχυρά τυποποιημένη γλώσσα ερωτημάτων βάσεων δεδομένων. Λέγοντας ότι είναι ισχυρά τυποποιημένη εννοείται ότι τα ερωτήματα καθορίζονται χρησιμοποιώντας τις κλάσεις που υπάρχουν στο μοντέλο δεδομένων της εφαρμογής [10].

2.4.6 Microsoft SQL Server

Αποτελεί το σύστημα διαχείρισης της σχεσιακής βάσης δεδομένων που έχει αναπτυχθεί από τη Microsoft. Σκοπός του συστήματος είναι η αποθήκευση δεδομένων και η ανάκτησή τους εφόσον απαιτηθεί από άλλες εφαρμογές. Μέσω των εργαλείων που παρέχονται από το Visual Studio μπορεί να εκτελεστεί οποιαδήποτε εργασία σχεδίασης βάσης δεδομένων στον server.

2.4.7 ASP.NET MVC Framework

Το ASP.NET MVC Framework είναι το πλαίσιο εργασίας που εφαρμόζει την αρχιτεκτονική MVC στο ASP.NET για την ανάπτυξη εφαρμογών ιστού [11].

Το ASP.NET μπορεί να διαχωριστεί σε δύο συστατικά :

1. Τα οπτικά στοιχεία που αποτελούν την διεπαφή χρήστη ή όπως ονομάζονται από τη Microsoft, Web Forms. Αυτά ανήκουν στο χώρο ονομάτων System.Web.UI.* της βιβλιοθήκης κλάσεων του .NET.
2. Τα μη οπτικά στοιχεία της εφαρμογής που τρέχουν στο υπόβαθρο . Αυτά ανήκουν στο χώρο ονομάτων System.Web.* της βιβλιοθήκης κλάσεων του .NET.

Το ASP.NET MVC αποτελεί το οπτικό στοιχείο της εφαρμογής αντί των Web Forms και οι κλάσεις του ανήκουν στο χώρο ονομάτων System.Web.Mvc της βιβλιοθήκης κλάσεων του .NET.

Σύμφωνα με το μοντέλο MVC, τα συστατικά που απαρτίζουν το ASP.NET MVC περιγράφονται παρακάτω.

Model: Οι κλάσεις που αντιπροσωπεύουν το πεδίο της εφαρμογής που αναπτύσσεται. Τα αντικείμενα των κλάσεων περιέχουν τα δεδομένα που βρίσκονται αποθηκευμένα στη βάση δεδομένων της εφαρμογής και τον κώδικα που χρησιμοποιείται για τη διαχείριση των δεδομένων αυτών. Αποτελεί δηλαδή ένα είδος επιπέδου πρόσβασης δεδομένων στο οποίο μπορεί να χρησιμοποιηθεί το Entity Framework για την εκπλήρωση των λειτουργιών του.

View : Αποτελεί το πρότυπο με το οποίο δημιουργείται κώδικας HTML δυναμικά.

Controller : Οι κλάσεις που διαχειρίζονται τη σύνδεση μεταξύ View και Model. Ο κώδικας που υπάρχει στις κλάσεις αυτές διαχειρίζεται τα δεδομένα εισόδου των χρηστών, επικοινωνεί με τις κλάσεις του Model και αποφασίζει ποιο αρχείο View θα απεικονιστεί στο χρήστη.

Δομή ASP.NET MVC εφαρμογής

Όταν δημιουργείται μια εφαρμογή ASP.NET MVC με το Visual Studio, στον κατάλογο της εφαρμογής δημιουργούνται οι παρακάτω προκαθορισμένοι κατάλογοι [12].

Όνομα Καταλόγου	Σκοπός
/Controllers	Τοποθετούνται οι κλάσεις Controller που διαχειρίζονται τα αιτήματα URL
/Models	Τοποθετούνται οι κλάσεις που αντιπροσωπεύουν και διαχειρίζονται τα δεδομένα της εφαρμογής
/Views	Τοποθετούνται πρότυπα αρχεία διεπαφής που παρουσιάζουν

	περιεχόμενο δημιουργώντας κώδικα HTML.
/Scripts	Τοποθετούνται αρχεία σεναρίων Javascript
/Images	Τοποθετούνται αρχεία εικόνων που χρησιμοποιεί η εφαρμογή
/Content	Τοποθετούνται αρχεία κώδικα CSS ή αρχεία περιεχομένου του ιστοτόπου, εκτός από τα σενάρια και τις εικόνες.
/Filters	Τοποθετούνται αρχεία κώδικα φιλτραρίσματος, ένα προκεχωρημένο χαρακτηριστικό που παρέχει το ASP.NET MVC
/App_Data	Τοποθετούνται αρχεία δεδομένων τα οποία χρησιμοποιούνται για ανάγνωση και εγγραφή
/App_Start	Τοποθετούνται αρχεία που περιέχουν κώδικα διαμόρφωσης της εφαρμογής

Πίνακας 2. Δομή καταλόγων ASP.NET MVC εφαρμογών

Οι εφαρμογές ASP.NET MVC βασίζονται κυρίως σε συμβάσεις. Με αυτόν τον τρόπο οι προγραμματιστές δεν χρειάζεται να εκτελέσουν ρυθμίσεις στην εφαρμογή που μπορούν να συμπεραθούν λόγω σύμβασης. Ένα παράδειγμα τέτοιας σύμβασης είναι η ονοματολογία των καταλόγων και η δομή τους. Όταν για παράδειγμα χρειάζεται να παρουσιαστεί κάποιο πρότυπο αρχείο View, η διαδρομή θέσης του συγκεκριμένου αρχείου μπορεί να παραληφθεί όταν το αρχείο αναφέρεται μέσω μιας κλάσης Controller. Το ASP.NET MVC θα ψάξει να βρει το αρχείο στον κατάλογο \Views\[Όνομα Controller]\.

Σύμφωνα με τα παραπάνω, από τις πιο σημαντικές συμβάσεις που χρησιμοποιεί το ASP.NET MVC είναι ότι:

- Το όνομα κάθε κλάσης Controller έχει την κατάληξη Controller, π.χ. HomeControler, και βρίσκεται στον κατάλογο Controllers.
- Υπάρχει ένας κατάλογος Views για όλες τις όψεις της εφαρμογής.
- Τα Views που χρησιμοποιούν οι Controllers υπάρχουν σε υποκαταλόγους του καταλόγου Views οι οποίοι έχουν το ίδιο όνομα με τον Controller, χωρίς όμως την κατάληξη Controller. Για παράδειγμα, οι όψεις που χρησιμοποιεί ο HomeController θα βρίσκονται στη διαδρομή /Views/Home.

2.4.8 Razor View Engine

Ο μηχανισμός οπτικοποίησης Razor είναι μια σύνταξη γλώσσας προγραμματισμού που βασίζεται στην C# και χρησιμοποιείται για τη δημιουργία δυναμικών ιστοσελίδων. Ο σκοπός της είναι να παρέχει στον προγραμματιστή μια βελτιστοποιημένη σύνταξη για την παραγωγή κειμένου HTML με όσο το δυνατόν λιγότερες μεταβάσεις μεταξύ HTML και κώδικα. Ένα τμήμα Razor σύνταξης ξεκινάει με τον χαρακτήρα ‘’at’’ (@) ο οποίος σηματοδοτεί την έναρξη σύνταξης κώδικα εντός του αρχείου HTML [11].

2.4.9 Visual Studio

Το Visual Studio [13] είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών, υπηρεσιών και ιστοσελίδων υλοποιημένο από τη Microsoft. Περιλαμβάνει επεξεργαστή κώδικα με λειτουργία IntelliSense, ένα χαρακτηριστικό αυτόματης συμπλήρωσης κώδικα, που διευκολύνει τη συγγραφή κώδικα. Μπορεί να υποστηρίξει γλώσσες προγραμματισμού όπως C, C++, VB.NET, C#, F#, καθώς επίσης και XML, HTML/XHTML, JavaScript και CSS.

Επιπλέον, στο Visual Studio ενσωματώνονται εργαλεία σχεδίασης φορμών για την δημιουργία γραφικού περιβάλλοντος διεπαφής για εφαρμογές και εργαλεία σχεδίασης σχημάτων βάσεων δεδομένων. Έχει τη δυνατότητα να αναβαθμιστεί με επιπρόσθετα εργαλεία που βελτιώνουν τη λειτουργικότητά του σε πολλά επίπεδα, όπως για παράδειγμα επεξεργαστές κώδικα για άλλες γλώσσες προγραμματισμού εκτός από τις προκαθορισμένες.

2.4.10 Entity Framework

Είναι το σύνολο των τεχνολογιών που παρέχει το .NET Framework με τις οποίες υποστηρίζεται η ανάπτυξη εφαρμογών που προσανατολίζονται στη χρήση αποθηκευμένων δεδομένων. Με τη χρήση του συγκεκριμένου Framework μπορεί να δημιουργηθεί ένα μοντέλο δεδομένων γράφοντας κώδικα ή χρησιμοποιώντας εργαλεία σχεδίασης όπως πλαίσια και γραμμές. Οι δύο αυτές προσεγγίσεις είναι χρήσιμες στη δημιουργία καινούριας βάσης δεδομένων ή τροποποίησης κάποιας υπάρχουσας [14].

Κεφάλαιο 3°

3 Μοντέλο Ασφάλειας Διαδικτυακών Εφαρμογών

Η Ασφάλεια Πληροφοριών περιλαμβάνει την προστασία αγαθών. Άρα για να προστατέψουμε τα αγαθά μας και στην περίπτωση των διαδικτυακών εφαρμογών πρέπει να στηριχθούμε στις αρχές της αυθεντικοποίησης, εξουσιοδότησης, καταγραφής, εμπιστευτικότητας, ακεραιότητας και διαθεσιμότητας.

Ένας ολοένα αυξανόμενος αριθμός επιθέσεων στοχεύει τις διαδικτυακές εφαρμογές. Συνήθως οι επιθέσεις αυτές εισέρχονται στο περιβάλλον της εφαρμογής χρησιμοποιώντας το πρωτόκολλο HTTP. Η τυφλή εμπιστοσύνη στα firewalls και στην προστασία των hosts δεν επαρκεί αν υλοποιηθούν ανεξάρτητα το ένα από το άλλο. Η ασφάλιση μιας εφαρμογής θα πρέπει να περιλαμβάνει την εφαρμογή μέτρων ασφάλειας σε τρία επίπεδα:

- Επίπεδο δικτύου
- Επίπεδο συστήματος
- Επίπεδο εφαρμογής

Το ασφαλές δίκτυο και η ασφαλής υποδομή της πλατφόρμας λειτουργίας του host είναι απαραίτητα. Επιπλέον, μια διαδικτυακή εφαρμογή πρέπει να έχει σχεδιαστεί και υλοποιηθεί χρησιμοποιώντας κάποιες κατευθυντήριες γραμμές και ακολουθώντας αρχές ασφάλειας που έχουν αποδειχθεί αποτελεσματικές.

Το να γνωρίζουμε τη συνηθισμένη μεθοδολογία που ακολουθούν οι επιτιθέμενοι και τους στόχους που θέτουν όταν επιτίθενται σε μια εφαρμογή θα μας κάνει πιο αποτελεσματικούς όταν θα εφαρμόσουμε τα αντίμετρα που θα απαιτούνται για την ασφάλιση της εφαρμογής μας.

Για την αναγνώριση των πιθανών απειλών μιας εφαρμογής θα πρέπει να προσεγγίσουμε το θέμα βασιζόμενοι στους στόχους που προσπαθεί ο επιτιθέμενος να πετύχει. Για την κατηγοριοποίηση των απειλών θα χρησιμοποιηθεί το μοντέλο STRIDE που χρησιμοποιείται από την Microsoft [15] και θα αναλυθεί στη συνέχεια του Κεφαλαίου.

3.1 Βασικές Αρχές Ασφάλειας Διαδικτυακών Εφαρμογών

3.1.1 Αγαθά (Assets)

Ένα αγαθό είναι κάποιος πόρος αξίας όπως για παράδειγμα τα δεδομένα που υπάρχουν σε μια βάση δεδομένων ή στο σύστημα αρχείων ή οι υπολογιστικοί πόροι που χρησιμοποιούνται για το χειρισμό των δεδομένων [16].

3.1.2 Απειλή (Threat)

Απειλή είναι οποιοδήποτε πιθανό περιστατικό, κακόβουλο ή μη, το οποίο μπορεί να βλάψει κάποιο αγαθό.

3.1.3 Επίθεση (attack)

Η εκδήλωση μιας απειλής είναι η επίθεση. Αυτή ορίζεται ως η προσπάθεια ενός δυνητικού εισβολέα που εκμεταλλεύεται τις αδυναμίες των εφαρμογών να τις καταστρέψει, εκθέσει, αλλοιώσει,

υποκλέψει ή αποκτήσει σε αυτές μη εξουσιοδοτημένη πρόσβαση ή να τις χρησιμοποιήσει χωρίς εξουσιοδότηση.

3.1.4 Αδυναμία (Vulnerability)

Λέγοντας αδυναμία εννοείται μια αδυναμία του συστήματός ή της εφαρμογής που μπορεί να κάνει μια απειλή εφικτή. Αδυναμίες μπορεί να υπάρχουν λόγω φτωχού σχεδιασμού, λόγω σφαλμάτων διαμόρφωσης ή από ακατάλληλες τεχνικές χρήσης κώδικα. Η αδύναμη επικύρωση δεδομένων εισόδου είναι παράδειγμα αδυναμίας επιπέδου εφαρμογής.

3.1.5 Ευπάθεια (Exploit)

Η ευπάθεια είναι η πράξη που εκμεταλλεύεται μια αδυναμία για να βλάψει κάποιο αγαθό. Παράδειγμα ευπάθειας είναι η αποστολή κακόβουλων δεδομένων εισόδου σε μια εφαρμογή η πλημμύρισμα ενός δικτύου με μηνύματα προσπαθώντας να επιτευχθεί άρνηση υπηρεσιών από την εφαρμογή.

3.1.6 Αντίμετρα (Countermeasures)

Τα αντίμετρα είναι τα μέτρα εκείνα που λαμβάνονται με σκοπό να προστατέψουν την εφαρμογή και να αντιμετωπίσουν οτιδήποτε απειλεί τα αγαθά της. Είναι δηλαδή ο τρόπος με τον οποίο διαφυλάττουμε την εφαρμογή από κάποια συγκεκριμένη απειλή ελαττώνοντας ή μηδενίζοντας οποιεσδήποτε απώλειες μπορεί να επιφέρει η απειλή.

3.2 Κατηγοριοποίηση Απειλών Ασφάλειας κατά STRIDE

Οι απειλές που αντιμετωπίζουν οι διαδικτυακές εφαρμογές μπορούν να κατηγοριοποιηθούν ανάλογα με τους στόχους και τους σκοπούς κάθε επίθεσης. Η κατηγοριοποίηση αυτή βοηθάει πάρα πολύ στη οργάνωση μιας σωστής και αποτελεσματικής στρατηγικής ασφάλειας για την αντιμετώπιση των απειλών. Το ακρωνύμιο STRIDE χρησιμοποιείται από το προσωπικό της Microsoft για να κατηγοριοποιήσουν τους διαφορετικούς τύπους απειλών [15]. Τα αρχικά STRIDE αντιστοιχούν στις παρακάτω λέξεις:

- Spoofing (Εξαπάτηση): Είναι η προσπάθεια πρόσβασης σε ένα σύστημα χρησιμοποιώντας ψεύτικη ταυτότητα. Αυτό μπορεί να επιτευχθεί είτε χρησιμοποιώντας κλεμμένα διαπιστευτήρια χρήστη (credentials) είτε χρησιμοποιώντας ψεύτικη διεύθυνση IP. Αφότου επιτευχθεί η πρόσβαση, κλιμάκωση δικαιωμάτων και κατάχρηση της εφαρμογής είναι τα επόμενα βήματα του επιτιθέμενου.
- Tampering (Αλλοίωση): Είναι η μη εξουσιοδοτημένη τροποποίηση δεδομένων όταν αυτά μεταφέρονται εντός κάποιου δικτύου μεταξύ δύο υπολογιστών.
- Repudiation (Απάρνηση): Είναι η δυνατότητα που δίνεται στους χρήστες (νόμιμους ή μη) να αρνηθούν ότι εκτέλεσαν κάποιες πράξεις ή συναλλαγές. Χωρίς επαρκή παρακολούθηση και καταγραφή επιθέσεις τέτοιου είδους είναι πολύ δύσκολο να αποδειχθούν.
- Information disclosure (Αποκάλυψη πληροφοριών): Είναι η μη επιθυμητή αποκάλυψη ιδιωτικών δεδομένων. Παράδειγμα αυτής της απειλής είναι όταν ένας χρήστης μπορεί να δει τα περιεχόμενα ενός πίνακα ή κάποιο αρχείο, για τα οποία δεν είναι εξουσιοδοτημένος.
- Denial of service (Άρνηση παροχής υπηρεσίας): Είναι η διαδικασία με την οποία ένα σύστημα ή εφαρμογή καθίσταται μη διαθέσιμο στους χρήστες. Τέτοιου είδους επιθέσεις μπορούν να επιτευχθούν με βομβαρδισμό αιτημάτων προς έναν server έτσι ώστε να καταναλωθούν όλοι οι πόροι της εφαρμογής.

- Elevation of privileges (Κλιμάκωση δικαιωμάτων): Είναι η περίπτωση όπου ένας χρήστης με περιορισμένα δικαιώματα πρόσβασης αναλαμβάνει την ταυτότητα ενός προνομιακού χρήστη για να αποκτήσει πρόσβαση σε πόρους της εφαρμογής που δεν θα μπορούσε κανονικά.

3.3 Απαιτήσεις Ασφάλειας Διαδικτυακών Εφαρμογών

Οι απαιτήσεις ασφάλειας των διαδικτυακών εφαρμογών που προκύπτουν ύστερα από την προηγούμενη κατηγοριοποίηση είναι οι παρακάτω.

3.3.1 Αυθεντικοποίηση (Authentication)

Όταν μιλάμε για αυθεντικοποίηση είναι σαν να κάνουμε την ερώτηση «Ποιος είσαι;». Εννοούμε δηλαδή τη διαδικασία της μοναδικής ταυτοποίησης των πελατών (clients) που χρησιμοποιούν την εφαρμογή μας. Clients της εφαρμογής μπορεί να είναι άνθρωποι, υπολογιστές, άλλες υπηρεσίες ή διεργασίες.

3.3.2 Εξουσιοδότηση (Authorization)

Όταν μιλάμε για εξουσιοδότηση είναι σαν να κάνουμε την ερώτηση «Τι επιτρέπεται να κάνεις;». Είναι η διαδικασία ελέγχου των πόρων και των λειτουργιών της εφαρμογής που επιτρέπεται να προσπελάσει κάποιος αυθεντικοποιημένος client. Οι πόροι περιλαμβάνουν τα αρχεία της εφαρμογής, βάσεις δεδομένων που χρησιμοποιεί η εφαρμογή, τους πίνακες της βάσης δεδομένων κα. Οι λειτουργίες περιλαμβάνουν εκτελέσεις συναλλαγών όπως αγορά προϊόντων μεταφορά χρημάτων από ένα λογαριασμό σε άλλο ή αλλαγή των στοιχείων πιστωτικής κάρτας.

3.3.3 Καταγραφή (Auditing)

Η αποτελεσματική παρακολούθηση και καταγραφή είναι σημαντικά για την μη αποποίηση (non-repudiation). Η ιδιότητα της μη αποποίησης εγγυάται ότι ένας χρήστης της εφαρμογής δεν μπορεί να αρνηθεί ότι εκτέλεσε μια λειτουργία ή μια συναλλαγή. Για παράδειγμα σε μια εφαρμογή ηλεκτρονικού καταστήματος, ένας χρήστης δεν μπορεί να αρνηθεί ότι παρήγγειλε ένα πουκάμισο συγκεκριμένου χρώματος.

3.3.4 Εμπιστευτικότητα (Confidentiality)

Η εμπιστευτικότητα πολλές φορές αναφέρεται και ως ιδιωτικότητα. Είναι η διαδικασία με την οποία εξασφαλίζεται ότι τα δεδομένα παραμένουν εμπιστευτικά και ότι δεν μπορούν να διαβαστούν από μη εξουσιοδοτημένους χρήστες ή ωτακουστές (eavesdroppers) που παρακολουθούν την κίνηση μηνυμάτων στο δίκτυο. Συνήθως χρησιμοποιείται η κρυπτογράφηση για την επίτευξη της εμπιστευτικότητας.

3.3.5 Ακεραιότητα (Availability)

Έχοντας πετύχει ακεραιότητα στα δεδομένα εξασφαλίζουμε ότι αυτά είναι προστατευμένα από εκούσια ή ακούσια τροποποίηση. Είναι πολύ σημαντική ιδιότητα που πρέπει να χαρακτηρίζει μια εφαρμογή διαδικτύου αφού τα δεδομένα της διασχίζουν το Διαδίκτυο. Ο τρόπος με τον οποίο επιτυγχάνεται για τα δεδομένα κίνησης είναι με τεχνικές κατακερματισμού (hashing) και με κώδικες αυθεντικοποίησης μηνυμάτων (message authentication code) [17].

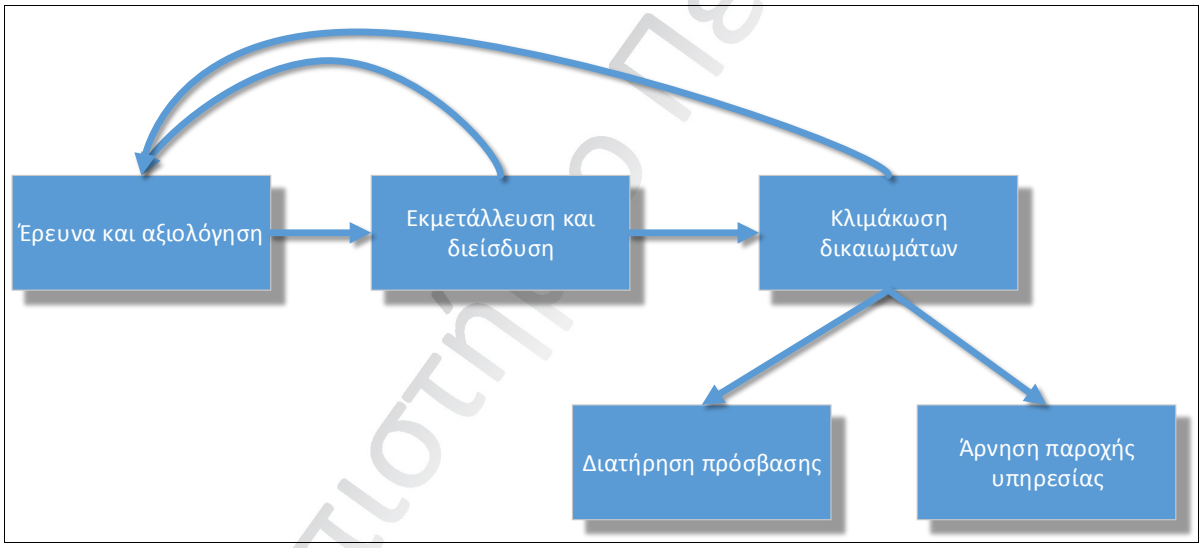
3.3.6 Διαθεσιμότητα (Availability)

Μια διαδικτυακή εφαρμογή θα πρέπει να είναι πάντα διαθέσιμη στους νόμιμους χρήστες της. Πολλές φορές ο σκοπός πολλών επιτιθέμενων είναι να καταστρέψουν μια εφαρμογή ή να την κατακλύσουν με μηνύματα έτσι ώστε άλλοι χρήστες να μην μπορούν να την προσπελάσουν.

3.4 Μεθοδολογία επίθεσης

Γνωρίζοντας την προσέγγιση που ακολουθούν οι επιτιθέμενοι για να στοχοποιήσουν εφαρμογές ιστού θα βοηθήσει στη λήψη καλύτερων μέτρων άμυνας [18]. Τα βασικά βήματα της μεθοδολογίας που ακολουθούν οι επιτιθέμενοι είναι τα παρακάτω:

- Έρευνα και αξιολόγηση (Survey and assess)
- Εκμετάλλευση και διείσδυση (Exploit and penetrate)
- Κλιμάκωση δικαιωμάτων (Escalate privileges)
- Διατήρηση πρόσβασης (Maintain access)
- Άρνηση παροχής υπηρεσίας (Deny service)



Εικόνα 6. Βασικά βήματα μεθοδολογίας επιτιθέμενων

3.4.1 Έρευνα και αξιολόγηση

Η διερεύνηση και η αξιολόγηση των χαρακτηριστικών του πιθανού στόχου εκτελούνται διαδοχικά. Αυτό είναι το πρώτο βήμα που κάνει ένας επιτιθέμενος [19]. Τα χαρακτηριστικά μπορεί να περιλαμβάνουν τις παρεχόμενες υπηρεσίες από τον στόχο, τα πρωτόκολλα τα οποία χρησιμοποιεί καθώς επίσης και τις πιθανές ευπάθειες ή τα πιθανά σημεία εισόδου στην εφαρμογή. Οι πληροφορίες που λαμβάνονται από αυτό το βήμα χρησιμοποιούνται στη συνέχεια για τη σχεδίαση της αρχικής επίθεσης.

3.4.2 Εκμετάλλευση και διείσδυση

Το δεύτερο βήμα είναι αυτό της εκμετάλλευσης και διείσδυσης. Αν θεωρήσουμε ότι το δίκτυο και ο host της εφαρμογής είναι πλήρως ασφαλισμένοι, τότε η εφαρμογή αποτελεί το επόμενο επίπεδο

επίθεσης. Για έναν επιτιθέμενο, ο ευκολότερος τρόπος για να διεισδύσει σε μια εφαρμογή είναι μέσω των ίδιων εισόδων που χρησιμοποιούν οι μη κακόβουλοι χρήστες.

3.4.3 Κλιμάκωση δικαιωμάτων

Αφότου ένας επιτιθέμενος καταφέρει να διεισδύσει σε μια εφαρμογή ή ένα δίκτυο, το επόμενο βήμα που ακολουθεί είναι να προσπαθήσει να αυξήσει τα δικαιώματα πρόσβασης που διαθέτει. Συγκεκριμένα ψάχνει για δικαιώματα διαχειριστή τα οποία παρέχονται από τους λογαριασμούς που ανήκουν σε γκρουπ διαχειριστών.

3.4.4 Διατήρηση πρόσβασης

Παράλληλα με τη κλιμάκωση δικαιωμάτων, ο επιτιθέμενος συνήθως προσπαθεί να διευκολύνει μελλοντικές προσπάθειες πρόσβασης στην εφαρμογή και προσπαθεί επίσης να καλύψει τα ίχνη του. Η συνηθισμένη προσέγγιση που ακολουθείται για μελλοντική πρόσβαση είναι η χρήση υπαρχόντων λογαριασμών χρηστών οι οποίοι στερούνται ισχυρής προστασίας. Για την κάλυψη των ιχνών τους συνήθως καθαρίζουν τα αρχεία καταγραφής. Συνεπώς, τα αρχεία καταγραφής θεωρούνται πρωτεύον στόχος για τους επιτιθέμενους.

3.4.5 Άρνηση παροχής υπηρεσίας

Στις περιπτώσεις που δεν μπορούν να αποκτήσουν πρόσβαση στους πόρους της εφαρμογής, συνήθως εξαπολύουν επιθέσεις άρνησης παροχής υπηρεσιών έτσι ώστε να μην μπορεί να χρησιμοποιήσει κανείς την εφαρμογή. Την επιλογή αυτή ακολουθούν επιτιθέμενοι που βρίσκονται εξωτερικά του δικτύου.

3.5 Αντίμετρα στις κατηγορίες απειλών STRIDE

Σύμφωνα με το μοντέλο STRIDE, σε κάθε κατηγορία απειλής αντιστοιχεί και ένα σύνολο από αντίμετρα που θα πρέπει να χρησιμοποιούνται για την ελάττωση της πιθανότητας επίθεσης. Η αντιστοιχία φαίνεται στον παρακάτω πίνακα.

Απειλή	Αντίμετρα
Spoofing user identity	Χρήση ισχυρής αυθεντικοποίησης. Οι κωδικοί πρόσβασης δεν θα πρέπει να αποθηκεύονται με τη μορφή απλού κειμένου. Τα διαπιστευτήρια δεν θα πρέπει να μεταδίδονται με τη μορφή απλού κειμένου. Η προστασία των cookies αυθεντικοποίησης θα πρέπει να γίνεται με πρωτόκολλο SSL.
Tampering	Στα δεδομένα θα πρέπει πάντα να εφαρμόζονται τεχνικές κατακερματισμού και να χρησιμοποιούνται ψηφιακές υπογραφές. Χρήση ισχυρής εξουσιοδότησης. Ασφάλιση των ζεύξεων επικοινωνίας με πρωτόκολλα που παρέχουν ακεραιότητα μηνυμάτων.
Repudiation	Δημιουργία ασφαλών μηχανισμών καταγραφής και ελέγχου. Χρήση ψηφιακών υπογραφών.
Information	Ισχυρή εξουσιοδότηση και κρυπτογράφηση.

disclosure	Ασφάλιση των ζεύξεων επικοινωνίας με πρωτόκολλα που παρέχουν εμπιστευτικότητα μηνυμάτων. Δεν θα πρέπει να γίνεται η αποθήκευση κωδικών πρόσβασης υπό μορφή απλού κειμένου.
Denial of service	Χρήση τεχνικών ρύθμισης του εύρους ζώνης και των πόρων της εφαρμογής. Φιλτράρισμα και επικύρωση των δεδομένων εισόδου.
Elevation of privilege	Θα πρέπει να ακολουθηθεί η αρχή των ελαχίστων προνομίων.

Πίνακας 3. Αντιστοίχιση αντιμέτρων στις απειλές σύμφωνα με το μοντέλο STRIDE

3.6 Υπηρεσίες Ασφάλειας Διαδικτυακών Εφαρμογών

Οι παρακάτω γενικές οδηγίες συνίσταται να ακολουθούνται κατά τη διάρκεια σχεδίασης και υλοποίησης μιας εφαρμογής ιστού [20]. Αυτές μπορούν να εφαρμοστούν ανεξάρτητα από την τεχνολογία υλοποίησης ή τη λειτουργικότητα της εφαρμογής.

3.6.1 Τμηματοποίηση

Συνίσταται να γίνεται διαχωρισμός των περιοχών πιθανής επίθεσης σε τμήματα. Το ερώτημα που θα πρέπει να τίθεται για να πετύχουμε κατακερματισμό είναι: Αν κάποιος επιτιθέμενος καταφέρει να διεισδύσει στην εφαρμογή σε ποιους πόρους θα μπορεί να έχει πρόσβαση; Παραδείγματα κατακερματισμού είναι τα firewalls και η χρήση λογαριασμών με όσο το δυνατόν πιο περιορισμένα προνόμια.

3.6.2 Χρήση ελαχίστων προνομίων

Εκτελώντας διεργασίες με τη χρήση λογαριασμών ελαχίστων προνομιών και δικαιωμάτων πρόσβασης, ελαττώνονται δραστικά οι δυνατότητες που θα έχει κάποιος επιτιθέμενος σε περίπτωση που αυτός καταφέρει να διεισδύσει στην εφαρμογή και να τρέξει κάποιο κώδικα.

3.6.3 Εφαρμογή προστασίας εις βάθος

Ποτέ δε θα πρέπει να βασιζόμαστε σε ένα μόνο επίπεδο εφαρμογής μέτρων ασφαλείας.

3.6.4 Μη εμπιστοσύνη δεδομένων εισόδου από τον χρήστη

Τα πεδία εισόδου δεδομένων από το χρήστη που περιλαμβάνει μια εφαρμογή είναι το πρωτεύον όπλο που θα χρησιμοποιήσει ένας επιτιθέμενος που την έχει θέσει ως στόχο. Το σκεπτικό που θα πρέπει να ακολουθείται είναι ότι όλα τα δεδομένα εισόδου θεωρούνται ως κακόβουλα εκτός αν αποδειχθεί το αντίθετο. Στη συνέχεια θα πρέπει να εφαρμόζεται προστασία εις βάθος για την επικύρωση των δεδομένων, προσέχοντας πάρα πολύ στα σημεία όπου διασχίζεται κάποιο όριο εμπιστοσύνης της εφαρμογής.

3.6.5 Έλεγχος στην αρχική είσοδο

Συνίσταται να γίνεται αυθεντικοποίηση και εξουσιοδότηση των καλούντων όσο το δυνατόν νωρίτερα.

3.6.6 Αποτυχία με ασφάλεια

Αν αποτύχει η εφαρμογή δεν θα πρέπει να επιτρέπεται πρόσβαση στα ευαίσθητα δεδομένα. Συνίσταται να επιστρέφονται φιλικά μηνύματα προς το χρήστη τα οποία όμως δεν εκθέτουν εσωτερικές λεπτομέρειες του συστήματος ή της εφαρμογής. Επίσης δεν θα πρέπει να περιέχονται οποιεσδήποτε λεπτομέρειες που πιθανόν να μπορούν να βοηθήσουν έναν επιτιθέμενο να εκμεταλλευτεί ευπάθειες της εφαρμογής.

3.6.7 Διασφάλιση του πιο αδύναμου κρίκου

Οπουδήποτε υπάρχει αδύναμος κρίκος στην αλυσίδα ασφάλειας στα επίπεδα δικτύου, host ή εφαρμογής είναι μια ευκαιρία για παραβίαση ασφάλειας.

3.6.8 Δημιουργία ασφαλών προεπιλογών

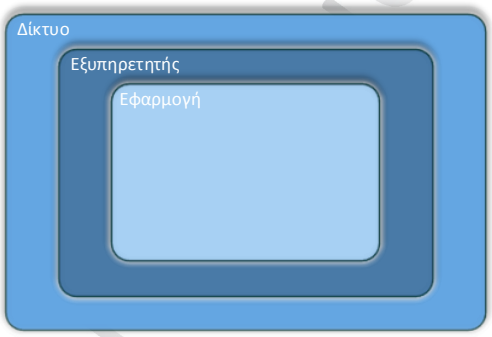
Συνίσταται πάντα να χρησιμοποιούνται όσο πιο ασφαλείς προεπιλογές γίνεται. Π.χ. ένας προεπιλεγμένος λογαριασμός χρήστη θα πρέπει να είναι ρυθμισμένος με όσο το δυνατόν λιγότερα προνόμια ή όταν συμβαίνει κάποιο λάθος στην εφαρμογή θα πρέπει η προεπιλογή να καθορίζει ότι ποτέ δεν θα στέλνονται μηνύματα λάθους που μπορεί να περιέχουν ευαίσθητες πληροφορίες ικανές να χρησιμοποιηθούν εναντίον της εφαρμογής.

3.6.9 Ελάττωση της επιφάνειας επίθεσης

Αν δεν χρησιμοποιείται κάποια λειτουργία, τότε θα πρέπει να αφαιρεθεί ή να απενεργοποιηθεί. Με αυτόν τον τρόπο ελαττώνεται η επιφάνεια επίθεσης κατά της εφαρμογής.

3.7 Εφαρμογή Μέτρων Ασφάλειας ανά Επίπεδο

Η σχεδίαση και η ανάπτυξη του λογισμικού μιας διαδικτυακής εφαρμογής πρέπει να υποστηρίζεται από ένα ασφαλές δίκτυο, ασφαλή συστήματα και οπωσδήποτε ασφαλείς ρυθμίσεις της εφαρμογής στο σύστημα που πρόκειται αυτή να αναπτυχθεί. Τα τρία αυτά επίπεδα φαίνονται στην παρακάτω εικόνα και αναλύονται στις επόμενες παραγράφους.



Εικόνα 7. Επίπεδα εφαρμογής αρχών ασφάλειας στις διαδικτυακές εφαρμογές

3.7.1 Επίπεδο Δικτύου

Μια ασφαλής εφαρμογή Ιστού βασίζεται σε μια ασφαλή υποδομή δικτύου. Ένα δίκτυο αποτελείται από δρομολογητές, τείχη ασφαλείας (firewalls) και μεταγωγείς (switches). Ένα ασφαλές δίκτυο πρέπει να προστατεύεται από επιθέσεις που βασίζονται στα πρωτόκολλα TCP/IP αλλά θα πρέπει

να υλοποιεί αντίμετρα όπως ισχυρούς κωδικούς πρόσβασης και ασφαλείς διεπαφές διαχείρισης δικτύου. Είναι επίσης υπεύθυνο για την διασφάλιση της ακεραιότητας των μηνυμάτων που διαχειρίζεται και προωθεί. Ο παρακάτω πίνακας περιγράφει τις ιδιότητες ασφάλειας κάθε συστατικού που περιέχεται σε ένα δίκτυο.

Συστατικό	Ιδιότητες ασφάλειας
Router	Οι δρομολογητές αποτελούν το εξωτερικό περίγραμμα του δικτύου. Μεταφέρουν πακέτα δεδομένων στις θύρες (ports) και με τα πρωτόκολλα που χρειάζεται η κάθε εφαρμογή. Οι ευπάθειες που σχετίζονται με τα πρωτόκολλα TCP/IP θα πρέπει να μπλοκάρονται σε αυτό το περίγραμμα.
Firewall	Το firewall μπλοκάρει τα πρωτόκολλα και τις θύρες που δεν χρησιμοποιούνται από την εφαρμογή. Επιπρόσθετα, επιβάλλουν μια ασφαλή κίνηση στο δίκτυο παρέχοντας φιλτράρισμα κακόβουλων επικοινωνιών σχετικά με κάθε εφαρμογή.
Switch	Οι μεταγωγείς χρησιμοποιούνται για το διαχωρισμό τμημάτων του δικτύου.

Πίνακας 4. Απαιτήσεις ασφάλειας συστατικών δικτύου

3.7.2 Επίπεδο Συστήματος

Για να ασφαλίσουμε τα συστήματα που θα χρησιμοποιηθούν από την εφαρμογή γίνεται μια διάσπαση των ρυθμίσεων διαμόρφωσης που τους αφορούν σε κατηγορίες. Με αυτόν τον τρόπο δημιουργούμε ένα πλαίσιο εργασίας σύμφωνα με το οποίο γίνεται πιο εύκολο να επικεντρωθούμε σε μια συγκεκριμένη κατηγορία και να ασχοληθούμε με τα θέματα ασφάλειας που την αφορούν. Στον παρακάτω πίνακα περιγράφονται οι κατηγορίες που αφορούν τη διαμόρφωση των συστημάτων.

Κατηγορία	Περιγραφή
Αναβαθμίσεις	Όταν ανακαλύπτονται νέες ευπάθειες ο κώδικας που χρησιμοποιείται για την εκμετάλλευσή τους αναρτάται συνήθως στο Διαδίκτυο εντός των πρώτων ωρών μιας επιτυχημένης επίθεσης. Η αναβάθμιση του λογισμικού των συστημάτων αποτελεί το πρώτο βήμα που θα πρέπει να γίνεται για τη διασφάλισή του.
Υπηρεσίες	Το σύνολο των υπηρεσιών που θα παρέχει ένα σύστημα καθορίζεται από το ρόλο του και τις εφαρμογές που φιλοξενεί. Απενεργοποιώντας τις αχρησιμοποίητες υπηρεσίες ελαττώνεται γρήγορα-γρήγορα η πιθανότητα επίθεσης.
Πρωτόκολλα	Απενεργοποιώντας τα πρωτόκολλα δικτύου που δεν χρησιμοποιούνται ελαττώνονται και οι δίοδοι που πιθανόν να χρησιμοποιούσε κάποιος επιτιθέμενος.
Λογαριασμοί	Ο αριθμός των λογαριασμών στους οποίους έχει πρόσβαση ένας εξυπηρετητής θα πρέπει να περιορίζεται στο ελάχιστο απαιτούμενο. Επιπλέον θα πρέπει να θεσπίζονται κατάλληλες πολιτικές ασφάλειας λογαριασμών όπως επιβολή χρήσης ισχυρών κωδικών πρόσβασης.
Αυθεντικοποίηση	Η γνωστή ιδιότητα του «Ποιος είσαι;» σχετιζόμενη με την εφαρμογή.
Εξουσιοδότηση	Πώς η εφαρμογή παρέχει έλεγχο πρόσβασης στις λειτουργίες και τους πόρους της.
Αρχεία και κατάλογοι	Θα πρέπει να διασφαλίζονται με περιορισμένα δικαιώματα τα οποία να επιτρέπουν πρόσβαση μόνο στις απαραίτητες υπηρεσίες και λογαριασμούς χρηστών.
Shares	Ο διαμοιρασμός αρχείων θα πρέπει να περιορίζεται στο ελάχιστο πρακτικό.

Θύρες	Οι ανοιχτές θύρες σε έναν σύστημα θα πρέπει να είναι γνωστές και να παρακολουθούνται σε τακτική βάση.
Παρακολούθηση και καταγραφή	Η παρακολούθηση της κίνησης είναι ζωτικής σημασίας για την αναγνώριση εισβολέων ή επιθέσεων. Με την καταγραφή παρέχονται πολύ σημαντικές πληροφορίες για τον καθορισμό του τρόπου με τον οποίο επιτεύχθηκε μια εισβολή ή επίθεση στο σύστημα.
Registry	Πολλές ρυθμίσεις που σχετίζονται με την ασφάλεια διατηρούνται στην registry. Αυτή θα πρέπει να ασφαλίζεται εφαρμόζοντας χρησιμοποιώντας τις λειτουργίες του λειτουργικού συστήματος. Θα πρέπει επίσης να απαγορεύεται απομακρυσμένη διαχείρισή της.

Πίνακας 5. Κατηγορίες διαμόρφωσης συστημάτων

3.7.3 Επίπεδο Εφαρμογής

Για να διασφαλίσουμε μια εφαρμογή σε αυτό το επίπεδο πρέπει να οργανώσουμε τα θέματα ασφαλείας σε κατηγορίες, έτσι ώστε να μπορέσουμε να τα αντιμετωπίσουμε συστηματικά. Οι κατηγορίες που περιγράφονται στον παρακάτω πίνακα αντιπροσωπεύουν τις περιοχές ασφάλειας όπου συναντώνται τα περισσότερα σφάλματα στις διαδικτυακές εφαρμογές. Η κατηγοριοποίηση διευκολύνει στην επικέντρωση στα σημεία κλειδιά και στις επιλογές υλοποίησης της εφαρμογής που επηρεάζουν την ασφάλεια της εφαρμογής.

Κατηγορία	Περιγραφή
Επικύρωση δεδομένων εισόδου	Πώς μπορεί κάποιος να γνωρίζει ότι τα δεδομένα που λαμβάνει η εφαρμογή είναι επικυρωμένα και ασφαλή; Η κατηγορία αυτή αναφέρεται στο πώς η εφαρμογή φιλτράρει ή απορρίπτει τα δεδομένα εισόδου προτού τα επεξεργαστεί περαιτέρω.
Αυθεντικοποίηση	Η γνωστή ιδιότητα του «Ποιος είσαι;» σχετιζόμενη με την εφαρμογή.
Εξουσιοδότηση	Πώς η εφαρμογή παρέχει έλεγχο πρόσβασης στις λειτουργίες και τους πόρους της.
Διαχείριση διαμορφώσεων	Η κατηγορία αυτή έχει να κάνει με το πώς η εφαρμογή διαχειρίζεται λειτουργικά θέματα όπως με ποια βάση δεδομένων συνδέεται, πώς γίνεται η διαχείριση της εφαρμογής και πώς ασφαλίζονται οι όποιες ρυθμίσεις της.
Ευαίσθητα δεδομένα	Αναφέρεται στο πώς η εφαρμογή χειρίζεται τα δεδομένα που πρέπει να προστατευτούν είτε στη μνήμη είτε κατά τη μεταφορά τους.
Διαχείριση συνόδων	Με τον όρο σύνοδος εννοείται μια σειρά από σχετιζόμενες αλληλεπιδράσεις μεταξύ ενός χρήστη και της εφαρμογής. Η διαχείριση των συνόδων αναφέρεται στο πώς η εφαρμογή χειρίζεται και προστατεύει αυτές τις αλληλεπιδράσεις.
Κρυπτογραφία	Λέγοντας κρυπτογραφία αναφερόμαστε στο πώς η εφαρμογή επιβάλει τις ιδιότητες της εμπιστευτικότητας και της ακεραιότητας χρησιμοποιώντας μεθόδους κρυπτογραφίας.
Χειρισμός παραμέτρων	Πεδία από φόρμες, ορίσματα και τιμές των cookies συχνά χρησιμοποιούνται ως παράμετροι της εφαρμογής. Ο χειρισμός των παραμέτρων έχει να κάνει με το πώς η εφαρμογή διασφαλίζει τη μη αλλοίωσή τους και πώς τις επεξεργάζεται.
Διαχείριση εξαιρέσεων	Όταν μια κλήση μεθόδου της εφαρμογής αποτυγχάνει, πώς αντιδρά η εφαρμογή; Πόσα στοιχεία για αυτήν την αποτυχία αποκαλύπτονται; Επιστρέφεται κάποιο φιλικό μήνυμα σφάλματος στον χρήστη;

Παρακολούθηση και καταγραφή	Πώς παρακολουθούνται και καταχωρούνται γεγονότα σχετιζόμενα με την ασφάλεια της εφαρμογής.

Πίνακας 6. Κατηγορίες ευπαθειών διαδικτυακών εφαρμογών

3.8 Απειλές και αντίμετρα στο επίπεδο δικτύου

Ένας επιτιθέμενος μπορεί εύκολα να εκμεταλλευτεί συσκευές δικτύου που δεν έχουν διαμορφωθεί κατάλληλα. Οι πιο συνηθισμένες ευπάθειες που συναντώνται στο επίπεδο δικτύου είναι: ασθενείς προκαθορισμένες ρυθμίσεις εγκατάστασης, ασθενής έλεγχος πρόσβαση και συσκευές που στερούνται τις πιο πρόσφατες ενημερώσεις λογισμικού. Παρακάτω περιγράφονται κάποιες από αυτές.

3.8.1 Συλλογή πληροφοριών (Information Gathering)

Οι επιτιθέμενοι συνήθως ξεκινάνε τη συλλογή πληροφοριών σχετικών με ένα δίκτυο ελέγχοντας τις θύρες εισόδου. Αφού αναγνωρίσουν τις ανοιχτές χρησιμοποιούν τεχνικές banner grabbing και enumeration για να αναγνωρίσουν τους τύπους των συσκευών που υπάρχουν στο δίκτυο, το λειτουργικό σύστημα και τις εκδόσεις των εφαρμογών. Εφοδιασμένοι με αυτές τις πληροφορίες, μπορούν να επιτεθούν εναντίον γνωστών αδυναμιών που μπορεί να μην έχουν αναβαθμιστεί.

Τα αντίμετρα που πρέπει να παρθούν για την απαγόρευση της συλλογής πληροφοριών είναι:

- Διαμόρφωση των δρομολογητών έτσι ώστε να απαγορεύουν απαντήσεις σε αιτήματα που θέλουν να δημιουργήσουν αποτύπωμα του δικτύου.
- Διαμόρφωση του λειτουργικού συστήματος πάνω στο οποίο τρέχουν τα προγράμματα του δικτύου (π.χ. firewalls) έτσι ώστε να απαγορεύουν την αποτύπωση απενεργοποιώντας τα μη χρησιμοποιούμενα πρωτόκολλα και τις θύρες που δεν χρειάζονται.

3.8.2 Υποκλοπή (Sniffing)

Sniffing ή eavesdropping είναι η πράξη κατά την οποία κάποιος παρακολουθεί την κίνηση στο δίκτυο έτσι ώστε να ανακαλύψει πιθανούς κωδικούς πρόσβασης που μεταδίδονται σε απλό κείμενο ή πληροφορίες διαμόρφωσης οποιουδήποτε λογισμικού. Αυτό επιτυγχάνεται πολύ εύκολα με ένα πρόγραμμα υποκλοπής πακέτων (packet sniffer). Επίσης, είναι πολλές φορές εφικτή η αποκρυπτογράφηση κρυπτογραφημένων πακέτων που έχουν κρυπτογραφηθεί με ελαφριάς μορφής αλγόριθμους κρυπτογράφησης. Για να επιτευχθούν αυτά θα πρέπει οπωσδήποτε ο packet sniffer να βρίσκεται στο ενδιάμεσο της επικοινωνίας μεταξύ client και server.

Αντίμετρα που μπορούν να ληφθούν για την απαγόρευση υποκλοπών είναι:

- Κατάλληλη τμηματοποίηση του δικτύου και ισχυρή φυσική ασφάλεια.
- Πλήρης κρυπτογράφηση της επικοινωνίας, συμπεριλαμβανομένου των διαπιστευτηρίων πιστοποίησης. Τα πρωτόκολλα SSL και IPSec (Internet Protocol Security) είναι παραδείγματα τέτοιων λύσεων κρυπτογράφησης [17].

3.8.3 Πλαστογράφηση (Spoofing)

Η πλαστογράφηση είναι ένας τρόπος για να κρύψει κάποιος την αληθινή του ταυτότητα στο δίκτυο. Για να δημιουργήσει ένας επιτιθέμενος μια πλαστογραφημένη ταυτότητα, χρησιμοποιεί μια ψεύτικη διεύθυνση πηγής η οποία δεν ταυτίζεται με την πραγματική διεύθυνση από την οποία προέρχεται το πακέτο δεδομένων που αποστέλλεται. Η απειλή αυτή χρησιμοποιείται είτε για να κρύψει την προέλευση της επίθεσης, είτε για να παρακάμψει τις λίστες ελέγχου πρόσβασης του δικτύου

(Access Control Lists - ACLs) που περιορίζουν την πρόσβαση στο σύστημα ανάλογα με τη διεύθυνση προέλευσης του κάθε πακέτου.

Αντίμετρα για την προστασία από πλαστογράφηση περιλαμβάνουν:

- Φιλτράρισμα εισερχομένων πακέτων που φαίνεται ότι προέρχονται από μια εσωτερική διεύθυνση IP.
- Φιλτράρισμα εξερχομένων πακέτων που φαίνεται ότι προέρχονται από μη έγκυρη τοπική διεύθυνση IP.

3.8.4 Υφαρπαγή συνόδου (Session Hijacking)

Η υφαρπαγή συνόδου είναι γνωστή και ως επίθεση man in the middle. Με την απειλή αυτή εξαπατάται ένας server ή ένας client και θεωρεί τον «απέναντι» με τον οποίο επικοινωνεί ως νόμιμο, ενώ δεν είναι. Στην πραγματικότητα επικοινωνεί με κάποιον επιτιθέμενο ο οποίος χειρίζεται το δίκτυο με τρόπο που φαίνεται ότι είναι ο επιθυμητός «απέναντι».

Αντίμετρα για την αποτροπή session hijacking περιλαμβάνουν:

- Χρήση κρυπτογραφημένης επικοινωνίας στη σύνοδο
- Χρήση κρυπτογραφημένων διαύλων επικοινωνίας
- Διαρκής ενημέρωση σχετικά με αναβαθμίσεις επιπέδου πλατφόρμας που αφορούν διορθώσεις ευπαθειών TCP/IP.

3.8.5 Άρνηση παροχής υπηρεσίας (Denial of Service)

Η απειλή άρνησης παροχής υπηρεσίας απαγορεύει στους χρήστες την πρόσβαση σε έναν server ή μια υπηρεσία. Η επίθεση SYN flood είναι ένα τέτοιο παράδειγμα στο επίπεδο δικτύου. Ο σκοπός της είναι να στείλει μεγαλύτερο αριθμό αιτήσεων στον server απ' ότι μπορεί αυτός να ικανοποιήσει.

Αντίμετρα για την αποτροπή denial of service περιλαμβάνουν:

- Εφαρμογή των πιο πρόσφατων πακέτων υπηρεσιών
- Χρήση συστημάτων εντοπισμού διεισδύσεων δικτύου (Intrusion Detection Systems - IDS) τα οποία εντοπίζουν και ανταποκρίνονται αυτόματα σε επιθέσεις SYN flood.

3.9 Απειλές και αντίμετρα στο επίπεδο συστήματος

Οι απειλές κατά των συστημάτων έχουν ως στόχο το λογισμικό του συστήματος πάνω στο οποίο τρέχει η εφαρμογή όπως για παράδειγμα το λειτουργικό σύστημα, τον IIS, το .NET Framework ή τον SQL Server, ανάλογα με το ρόλο του εκάστοτε εξυπηρετητή. Οι κυριότερες από αυτές τις απειλές παρουσιάζονται στις παρακάτω παραγράφους.

3.9.1 Ιοί, Δούρειοι Ίπποι και Σκουλήκια (Viruses, Trojan Horses,Worms)

Ο ιός είναι ένα πρόγραμμα που έχει σχεδιαστεί για να εκτελεί κακόβουλες ενέργειες και να προκαλεί δυσλειτουργίες στο λογισμικό του συστήματος.

Ο Δούρειος Ίππος μοιάζει με τον ιό εκτός από το ότι ο κακόβουλος κώδικας περιέχεται μέσα σε κάποιο άλλο εκτελέσιμο πρόγραμμα ή αρχείο δεδομένων τα οποία φαίνονται ακίνδυνα.

Ένα Σκουλήκι είναι παρόμοιο με ένα Δούρειο Ίππο εκτός από το ότι αυτό-αναπαράγεται από έναν server σε άλλο. Είναι δύσκολο να εντοπιστεί γιατί συνήθως δεν δημιουργούν ορατά αρχεία. Συχνά εντοπίζονται όταν αρχίζουν να καταναλώνουν πόρους του συστήματος και το καθιστούν αργό ή προκαλούν παύση εκτέλεσης άλλων προγραμμάτων.

Αντίμετρα που μπορούν να χρησιμοποιηθούν εναντίον τους είναι:

- Διατήρηση ενημερωμένου λειτουργικού συστήματος με τα πιο πρόσφατα πακέτα αναβάθμισης.
- Μπλοκάρισμα όλων των μη χρησιμοποιούμενων θυρών στο τείχος προστασίας και στον host.
- Απενεργοποίηση οποιασδήποτε λειτουργικότητας που δεν χρησιμοποιείται (πρωτόκολλα και υπηρεσίες).
- Αλλαγή των προκαθορισμένων ρυθμίσεων διαμόρφωσης και κατά συνέπεια ισχυροποίησή τους.

3.9.2 Αποτύπωση (Footprinting)

Οι επιτιθέμενοι χρησιμοποιούν την αποτύπωση για να συλλέξουν πολύτιμες πληροφορίες σχετικά με το σύστημα και να προετοιμαστούν κατάλληλα για πιο ουσιαστικές επιθέσεις. Παράδειγμα απειλής αποτύπωσης είναι η σάρωση θυρών. Οι τύποι της πληροφορίας που μπορεί να αποκαλυφθεί από footprinting είναι πληροφορίες λογαριασμών, εκδόσεις λειτουργικού συστήματος και άλλων τύπων λογισμικού, ονόματα server και πληροφορίες σχετικά με το σχήμα της βάσης δεδομένων.

Αντίμετρα για την αποτροπή footprinting είναι τα παρακάτω:

- Απενεργοποίηση των πρωτοκόλλων που δεν χρησιμοποιούνται.
- Κλείδωμα των θυρών με κατάλληλες ρυθμίσεις στο firewall του host.
- Χρήση φίλτρων για τα πρωτόκολλα TCP/IP και IPSec για την επίτευξη άμυνας εις βάθους.
- Χρήση συστήματος IDS το οποίο μπορεί να ρυθμιστεί έτσι ώστε να αναγνωρίζει πρότυπα footprinting και να απορρίπτει ύποπτη κίνηση.

3.9.3 Διάρρηξη κωδικών πρόσβασης (Password Cracking)

Αν ο επιτιθέμενος δεν μπορεί να συνδεθεί ανώνυμα με έναν server, τότε θα προσπαθήσει να πετύχει μια αυθεντικοποιημένη σύνδεση με αυτόν. Για να το καταφέρει θα πρέπει να γνωρίζει τον κατάλληλο συνδυασμό ονόματος χρήστη και κωδικού πρόσβασης. Όταν χρησιμοποιούνται από τον host προκαθορισμένα ονόματα χρηστών (λογαριασμοί) δίνεται στον επιτιθέμενο σαφές πλεονέκτημα, αφού το μόνο που έχει να κάνει είναι να «σπάσει» τον κωδικό πρόσβασης του λογαριασμού. Ακόμα πιο εύκολο για τον επιτιθέμενο είναι να χρησιμοποιούνται λογαριασμοί χωρίς κωδικούς πρόσβασης.

Αντίμετρα για την αντιμετώπιση αυτής της απειλής είναι:

- Χρήση ισχυρών κωδικών πρόσβασης για όλους τους λογαριασμούς χρηστών.
- Εφαρμογή πολιτικής κλειδώματος λογαριασμού για να μειωθεί ο αριθμός των προσπαθειών για να μαντέψει κάποιος τον κωδικό πρόσβασης.
- Δεν θα πρέπει να χρησιμοποιούνται οι προκαθορισμένοι λογαριασμοί. Θα πρέπει να γίνεται μετονομασία των συγκεκριμένων λογαριασμών.
- Θα πρέπει να γίνεται έλεγχος και καταγραφή των αποτυχημένων προσπαθειών login για να μπορούν να ανιχνευθούν προσπάθειες σπασίματος κωδικών πρόσβασης.

3.9.4 Denial of Service

Η απειλή αυτή μπορεί να επιτευχθεί με πολλούς τρόπους εναντίον πολλών στόχων εντός της υποδομής του host. Για παράδειγμα, ένας επιτιθέμενος μπορεί να χρησιμοποιήσει επίθεση brute force για να διακόψει τις υπηρεσίες που παρέχονται από μια εφαρμογή.

Αντίμετρα που μπορούν να χρησιμοποιηθούν εναντίον τέτοιου είδους απειλών είναι:

- Διαμόρφωση εφαρμογών, υπηρεσιών και λειτουργικού συστήματος έχοντας κατά νου τη συγκεκριμένη απειλή.
- Ενημέρωση του συστήματος πάντα με τις πιο πρόσφατες αναβαθμίσεις.
- Εξασφάλιση ότι οι πολιτικές κλειδώματος λογαριασμών δεν μπορούν να χρησιμοποιηθούν από τους επιτιθέμενους για να κλειδώσουν γνωστούς και ευρέως χρησιμοποιούμενους λογαριασμούς υπηρεσιών.
- Εξασφάλιση ότι η εφαρμογή που τρέχει στον host είναι ικανή να διαχειριστεί μεγάλο όγκο κίνησης και υλοποίηση κατωφλίων για να διαχειριστούν ασυνήθιστα υψηλή κίνηση.
- Χρήση IDS για εντοπισμό πιθανών επιθέσεων denial of service.

3.9.5 Αυθαίρετη εκτέλεση κώδικα (Arbitrary Code Execution)

Αν κάποιος επιτιθέμενος καταφέρει να εκτελέσει κακόβουλο κώδικα στον server τότε θα θέσει σε κίνδυνο τους πόρους του ή θα μπορέσει να ξεκινήσει καινούριες επιθέσεις εναντίον των υπόλοιπων συστημάτων του. Ο κίνδυνος αυξάνει στην περίπτωση που η διεργασία του server υπό την οποία τρέχει ο κακόβουλος κώδικας, έχει πολλά προνόμια. Αδυναμίες όπως μη σωστή ρύθμιση του server χωρίς τις πιο πρόσφατες αναβαθμίσεις μπορεί να επιφέρει τέτοιου είδους απειλές.

Για την αποφυγή arbitrary code execution μπορούν να παρθούν τα παρακάτω αντίμετρα:

- Ρύθμιση του IIS έτσι ώστε να απορρίπτονται urls που περιέχουν "../" και απαγόρευση πρόσβασης σε σημαντικά αρχεία.
- Κλείδωμα εντολών και εργαλείων συστήματος με χρήση περιοριστικών ACLs.
- Διαρκής ενημέρωση server με τις πιο πρόσφατες αναβαθμίσεις

3.9.6 Μη εξουσιοδοτημένη πρόσβαση (Unauthorized Access)

Ανεπαρκής έλεγχος πρόσβασης θα επιτρέψει σε κάποιον μη εξουσιοδοτημένο χρήστη να εκτελέσει λειτουργίες, ή να πάρει πληροφορίες που δεν του επιτρέπεται κανονικά. Συνηθισμένες αδυναμίες που επιτρέπουν μη εξουσιοδοτημένη πρόσβαση είναι ελλιπείς ρυθμίσεις ελέγχου πρόσβασης στον server και ασθενής μηχανισμός απόδοσης δικαιωμάτων στα αρχεία του συστήματος.

Για την αποφυγή αυτής της απειλής μπορούν να παρθούν τα παρακάτω αντίμετρα:

- Διαμόρφωση δικαιωμάτων πρόσβασης με ασφάλεια.
- Κλείδωμα καταλόγων και αρχείων με περιορισμένα δικαιώματα.
- Χρήση των μηχανισμών που παρέχει το .NET Framework εντός των εφαρμογών ASP.NET.

3.10 Κατηγοριοποίηση αδυναμιών στο επίπεδο εφαρμογής και αντίμετρα

Ένας τρόπος που βοηθάει στην ανάλυση των απειλών επιπέδου εφαρμογής είναι η οργάνωσή τους ανά κατηγορία αδυναμίας. Μια τέτοια κατηγοριοποίηση φαίνεται στον παρακάτω πίνακα.

Κατηγορία Αδυναμίας	Απειλές
Επικύρωση δεδομένων εισόδου	Buffer overflow, cross-site scripting, SQL injection, canonicalization
Αυθεντικοποίηση	brute force, network eavesdropping, dictionary attacks, cookie replay, κλοπή διαπιστευτηρίων
Εξουσιοδότηση	Κλιμάκωση δικαιωμάτων, αποκάλυψη εμπιστευτικών δεδομένων, αλλοίωση δεδομένων, παραπλάνηση (luring)
Διαχείριση διαμορφώσεων	Μη εξουσιοδοτημένη πρόσβαση στις διεπαφές διαχείρισης ή σε άλλα σημεία διαμόρφωσης, ανάκτηση δεδομένων ρυθμίσεων σε μορφή απλού κειμένου, έλλειψη ατομικής ευθύνης, διεργασίες και λογαριασμοί υπηρεσιών με περισσότερα προνόμια από τα απαιτούμενα.
Ευαίσθητα δεδομένα	Πρόσβαση στις αποθήκες ευαίσθητων δεδομένων, network eavesdropping, αλλοίωση δεδομένων
Διαχείριση συνόδων	Υφαρπαγή συνεδρίας, session replay, επίθεση man in the middle
Κρυπτογραφία	Αδύναμος μηχανισμός δημιουργίας ή διαχείρισης κλειδιών, χρήση συνηθισμένων τεχνικών κρυπτογράφησης
Χειρισμός παραμέτρων	Κακόβουλος χειρισμός των: Query string, πεδίων φόρμας, cookie και επικεφαλίδων HTTP
Διαχείριση εξαιρέσεων	Αποκάλυψη πληροφοριών, άρνηση παροχής υπηρεσίας
Παρακολούθηση και καταγραφή	Άρνηση ενός χρήστη ότι εκτέλεσε μια εργασία, κάλυψη ιχνών επιτιθέμενου, εκμετάλλευση εφαρμογής χωρίς να αφήνονται ίχνη

Πίνακας 7. Απειλές ανά κατηγορία αδυναμίας στο επίπεδο εφαρμογής.

3.10.1 Επικύρωση δεδομένων εισόδου (Input Validation)

Η επικύρωση δεδομένων αποτελεί κρίσιμη αδυναμία ασφάλειας εφόσον κάποιος επιτιθέμενος ανακαλύψει ότι μια εφαρμογή κάνει αβάσιμες υποθέσεις σχετικά με τον τύπο, το μήκος, τη μορφή ή το εύρος των δεδομένων εισόδου που δέχεται. Ο επιτιθέμενος μπορεί να εισάγει κατάλληλα δεδομένα που μπορούν να απειλήσουν την εφαρμογή.

Όταν τα σημεία εισόδου σε επίπεδο δικτύου και συστήματος είναι πλήρως ασφαλισμένα, τα κοινά σημεία διεπαφής που είναι εκτεθειμένα από την εφαρμογή αποτελούν την μοναδική πηγή επίθεσης. Η εισαγωγή δεδομένων στην εφαρμογή είναι ένας τρόπος για να δοκιμαστεί το σύστημα κάτω από την εφαρμογή και να εκτελεστεί κάποιος κώδικας για λογαριασμό του επιτιθέμενου.

Οι αδυναμίες επικύρωσης δεδομένων μιας εφαρμογής μπορεί να οδηγήσουν στις επιθέσεις εξετάζονται στις παρακάτω υποπαραγράφους.

Υπερχείλιση ενταμιευτών (Buffer Overflows)

Η υπερχείλιση ενταμιευτών μπορεί να οδηγήσει σε ευπάθειες άρνησης παροχής υπηρεσιών ή έγχυση κώδικα. Ο κώδικας με τον οποίο έχουν υλοποιηθεί οι βιβλιοθήκες του .NET και απαιτούν τη χρήση CLR για να τρέξει δεν είναι επιδεκτικός σε τέτοιου είδους προβλήματα λόγω του ότι τα καθορισμένα όρια των πινάκων (arrays) που χρησιμοποιούνται από κάποιο κομμάτι κώδικα ελέγχονται αυτόματα όταν τα στοιχεία του πίνακα πρόκειται να χρησιμοποιηθούν. Όταν όμως δεν

χρησιμοποιείται τέτοιου είδους κώδικας σε μια εφαρμογή, υπερχείλιση ενταμιευτών μπορεί εύκολα να συμβεί.

Για την αποφυγή buffer overflow μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Εκτέλεση εξονυχιστικού ελέγχου επικύρωσης δεδομένων εισόδου, κάτι που αποτελεί την πρώτη γραμμή άμυνας κατά του buffer overflow. Παρά το γεγονός ότι μπορεί να υπάρχει κάποιο ψεγάδι στην εφαρμογή που επιτρέπει στα αναμενόμενα δεδομένα να ξεπεράσουν τα όρια κάποιου ενταμιευτή, τα απροσδόκητα δεδομένα εισόδου θα αποτελέσουν το κύριο λόγω ύπαρξης αυτής της ευπάθειας. Οπότε θα πρέπει να περιορίζεται η είσοδος δεδομένων ανάλογα με τον τύπο, το μήκος, τη μορφή και το εύρος τους.

Cross-Site Scripting

Γράφεται επίσης ως XSS και μπορεί να προκαλέσει την εκτέλεση αυθαίρετου κώδικα στον browser κάποιου χρήστη κατά τη διάρκεια που αυτός είναι συνδεδεμένος σε έναν έμπιστο ιστότοπο. Σε αυτήν την περίπτωση ο επιτιθέμενος στοχεύει τους χρήστες της εφαρμογής και όχι την εφαρμογή καθεαυτή, όμως την χρησιμοποιεί ως όχημα για την επίθεση.

Επειδή ο κακόβουλος κώδικας προέρχεται από έμπιστο ιστότοπο, ο browser του χρήστη δεν έχει κάποιο τρόπο να τον αναγνωρίσει ως κακόβουλο. Συνηθισμένος στόχος τέτοιων επιθέσεων είναι τα cookies αυθεντικοποίησης ενός χρήστη, από τη στιγμή που αυτά είναι αποθηκευμένα στον υπολογιστή του και ο κώδικας του επιτιθέμενου μπορεί να έχει πρόσβαση σε αυτά.

Για να ξεκινήσει μια τέτοια επίθεση, ο επιτιθέμενος πρέπει να πείσει τον χρήστη να επιλέξει έναν υπερσύνδεσμο προσεκτικά τροποποιημένο, ο οποίος δείχνει προς μια κακόβουλη σελίδα η οποία με τη σειρά της επιστρέφει τα μη επικυρωμένα δεδομένα στον browser, εντός του ρεύματος εξόδου HTML. Παρακάτω φαίνεται ένα παράδειγμα κακόβουλου συνδέσμου:

www.yourwebapplication.com/logon.aspx?username=<script>alert('hacker code')</script>

Αν η εφαρμογή πάρει το query string username=<script>alert('hacker code')</script> αποτύχει να το επικυρώσει σωστά και το επιστρέψει στον browser, τότε ο κώδικας <script>alert('hacker code')</script> θα εκτελεστεί στον browser. Με χρήση του κατάλληλου script, ένας επιτιθέμενος μπορεί εύκολα να αποσπάσει το cookie αυθεντικοποίησης ενός χρήστη, να το στείλει στον δικό του ιστότοπο και στη συνέχεια να το χρησιμοποιήσει για να επικοινωνήσει με τον αντίστοιχο ιστότοπο ως αυθεντικοποιημένος χρήστης.

Για την αποφυγή XSS μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Εκτέλεση εξονυχιστικής επικύρωσης δεδομένων. Η εφαρμογή θα πρέπει να εξασφαλίζει ότι δεδομένα εισόδου από query strings, πεδία φορμών και cookies είναι έγκυρα για αυτήν. Όλα τα δεδομένα εισόδου θα πρέπει να θεωρούνται ως πιθανά κακόβουλα και να φιλτράρονται.
- Χρήση των μεθόδων HTMLEncode και URLEncode της κλάσης WebUtility του χώρου ονομάτων System.Net για κωδικοποίηση δεδομένων που μπορεί να περιλαμβάνουν δεδομένα χρήστη. Με τον τρόπο αυτό μετατρέπονται τα εκτελέσιμα scripts σε αβλαβή κώδικα HTML.

SQL Injection

Μια απειλή SQL injection εκμεταλλεύεται αδυναμίες επικύρωσης δεδομένων εισόδου με σκοπό να εκτελέσει αυθαίρετες εντολές στη βάση δεδομένων μιας εφαρμογής. Αυτό συμβαίνει στις περιπτώσεις που η εφαρμογή χρησιμοποιεί δεδομένα εισόδου για τη κατασκευή δυναμικών δηλώσεων SQL για λήψη δεδομένων από τη βάση δεδομένων. Επίσης μπορεί να συμβεί όταν ο

κώδικας της εφαρμογής χρησιμοποιεί αποθηκευμένες διαδικασίες βάσης δεδομένων στις οποίες εισάγονται αφιλτράριστα δεδομένα εισόδου. Το πρόβλημα μεγαλώνει όταν η εφαρμογή χρησιμοποιεί υπέρ-εξουσιοδοτημένο λογαριασμό για σύνδεση με τη βάση δεδομένων. Τότε, είναι εφικτή η χρήση του server της βάσης δεδομένων για εκτέλεση εντολών λειτουργικού συστήματος οι οποίες μπορούν να θέσουν σε κίνδυνο κι άλλες υπηρεσίες, ενώ παράλληλα μπορεί να κινδυνεύσουν και τα δεδομένα της βάσης.

Για την αποφυγή SQL injection μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Εκτέλεση εξονυχιστικής επικύρωσης δεδομένων. Η εφαρμογή θα πρέπει να εξασφαλίζει ότι τα δεδομένα που χρησιμοποιούνται για εκτέλεση ερωτημάτων είναι επικυρωμένα και ασφαλή πριν τα στείλει στη βάση.
- Χρήση παραμετροποιημένων αποθηκευμένων διαδικασιών για πρόσβαση στη βάση με σκοπό την εξασφάλιση ότι τα στοιχεία εισόδου που υπάρχουν στη διαδικασία δεν χρησιμοποιούνται ως εκτελέσιμες δηλώσεις.
- Χρήση λογαριασμών με όσο το δυνατόν λιγότερα προνόμια για σύνδεση της εφαρμογής στη βάση δεδομένων.

3.10.2 Αυθεντικοποίηση (Authentication)

Ανάλογα με τις απαιτήσεις μιας εφαρμογής, υπάρχουν αρκετοί μηχανισμοί αυθεντικοποίησης από τους οποίους μπορεί να επιλέξει κάποιος. Αν δεν επιλεγεί ο κατάλληλος για την κάθε περίπτωση ή δεν υλοποιηθεί όπως πρέπει, ένας επιτιθέμενος μπορεί να εκμεταλλευτεί τις αδυναμίες που θα προκύψουν για να αποκτήσει πρόσβαση στο σύστημα.

Οι κυριότερες απειλές που μπορούν να εκμεταλλευθούν αδυναμίες αυθεντικοποίησης μιας εφαρμογής μπορεί να οδηγήσουν στις επιθέσεις που εξετάζονται στις παρακάτω υποπαραγράφους.

Υποκλοπή δικτύου (Network Eavesdropping)

Αν τα διαπιστευτήρια αυθεντικοποίησης μεταδοθούν από τον client στον server σε μορφή απλού κειμένου, ένας επιτιθέμενος εξοπλισμένος με ένα απλό λογισμικό παρακολούθησης δικτύου εγκατεστημένο στο ίδιο δίκτυο με τον client ή το server μπορεί να δεί την κίνηση του δικτύου και να αποκτήσει τα διαπιστευτήρια.

Για την αποφυγή υποκλοπής δικτύου μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Χρήση μηχανισμών αυθεντικοποίησης που δεν μεταδίδουν κωδικούς πρόσβασης σε δίκτυο όπως π.χ. πρωτόκολλο Kerberos ή μηχανισμός αυθεντικοποίησης των Windows.
- Εξασφάλιση ότι οι κωδικοί πρόσβασης είναι κρυπτογραφημένοι (αν πρέπει να μεταδοθούν μέσω ενός δικτύου) ή ότι χρησιμοποιείται κρυπτογραφημένο κανάλι επικοινωνίας όπως π.χ. SSL.

Επιθέσεις Brute Force

Οι επιθέσεις αυτές βασίζονται στη διαθέσιμη υπολογιστική δύναμη που κατέχει ένας επιτιθέμενος και τη χρησιμοποιεί για να βρει κωδικούς πρόσβασης ή άλλες μυστικά δεδομένα που είναι κατακερματισμένα (hashed) ή είναι κρυπτογραφημένα. Για την ελάττωση του συγκεκριμένου ρίσκου αρκεί να χρησιμοποιούνται ισχυροί κωδικοί πρόσβασης. Επιπλέον, μαζί με τους κατακερματισμένους κωδικούς πρόσβασης μπορεί να χρησιμοποιηθούν τιμές salt [21]. Αυτός ο τρόπος καθυστερεί τον επιτιθέμενο να φτάσει στο στόχο του και επιτρέπει επαρκή χρόνο έτσι ώστε να ενεργοποιηθούν τα αντίμετρα.

Επιθέσεις λεξικού (Dictionary Attacks)

Η επίθεση αυτή χρησιμοποιείται για να αποκτήσει ο επιτιθέμενος κωδικούς πρόσβασης. Τα περισσότερα συστήματα δεν αποθηκεύουν τους κωδικούς πρόσβασης σε μορφή απλού κειμένου ούτε κρυπτογραφημένους. Αποφεύγουν την κρυπτογράφηση γιατί αν το κλειδί κρυπτογράφησης βρεθεί από κάποιον, τότε θα βρεθούν και θα πρέπει να ακυρωθούν όλοι οι κωδικοί πρόσβασης που είναι αποθηκευμένοι στο σύστημα.

Τα περισσότερα συστήματα που υλοποιούν αποθήκες λογαριασμών χρηστών δεν αποθηκεύουν τους κωδικούς πρόσβασης αλλά συνόψεις αυτών. Οι χρήστες πιστοποιούνται με υπολογισμό της σύνοψης του κωδικού που έδωσε ο χρήστης και σύγκρισή της με την αποθηκευμένη. Αν ένας επιτιθέμενος καταφέρει να αποκτήσει αυτή τη λίστα με τις συνόψεις, εκτελώντας μια brute force επίθεση θα καταφέρει να βρει και τους κωδικούς.

Για την επίθεση λεξικού χρησιμοποιείται ένα πρόγραμμα το οποίο διαπερνά μία-μία τις λέξεις ενός λεξικού (ή ακόμα και πολλών λεξικών σε διαφορετικές γλώσσες) και υπολογίζει τη σύνοψη καθεμιάς. Η σύνοψη αυτή συγκρίνεται με την αποθηκευμένη. Αυτό σημαίνει ότι ασθενείς κωδικοί πρόσβασης όπως η λέξη password θα βρεθεί πολύ γρήγορα. Πιο δυνατοί κωδικοί όπως p455w0rD είναι λιγότερο πιθανό να ανακαλυφθούν.

Αξίζει να σημειωθεί ότι από τη στιγμή που ο επιτιθέμενος αποκτήσει τη λίστα με τις συνόψεις, μπορεί να εκτελέσει την επίθεση λεξικού εκτός δικτύου, χωρίς να αλληλεπιδράσει με την εφαρμογή.

Για την αποφυγή επιθέσεων λεξικού μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Χρήση σύνθετων κωδικών πρόσβασης και όχι συνηθισμένων λέξεων. Μίξη κεφαλαίων γραμμάτων, πεζών, αριθμών και ειδικών χαρακτήρων.
- Οι συνόψεις που είναι αποθηκευμένες στη βάση δεδομένων πρέπει να είναι μη αναστρέψιμες. Χρήση τιμής salt μαζί με τη σύνοψη κάθε κωδικού πρόσβασης.

3.10.3 Επιθέσεις Cookie Replay

Από την επίθεση αυτή, ο επιτιθέμενος αποκτά το cookie αυθεντικοποίησης ενός χρήστη χρησιμοποιώντας το κατάλληλο λογισμικό παρακολούθησης και το ξαναστέλνει στην εφαρμογή για να αποκτήσει δικαιώματα χρησιμοποιώντας διαπιστευτήρια άλλου χρήστη.

Για την αποφυγή cookie replay attacks μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Χρήση κρυπτογραφημένου καναλιού επικοινωνίας με SSL, οποτεδήποτε μεταδίδεται cookie αυθεντικοποίησης.
- Καθορισμός σχετικά μικρού χρονικού ορίου που λήγει η ισχύς του cookie. Εφόσον λήξει, απαιτείται εκ νέου πιστοποίηση του χρήστη. Αν και το συγκεκριμένο αντίμετρο δεν εμποδίζει όλες τις επιθέσεις επανάληψης cookie, ελαττώνει το χρονικό διάστημα που μπορεί ο επιτιθέμενος να μεταδώσει το cookie χωρίς να απαιτηθεί νέα πιστοποίηση

Κλοπή διαπιστευτηρίων (Credential Theft)

Το ιστορικό που διατηρεί ο browser και η κρυφή του μνήμη, πολλές φορές αποθηκεύουν τα δεδομένα σύνδεσης των χρηστών για μελλοντική χρήση. Αν ο υπολογιστής χρησιμοποιηθεί από κάποιον άλλο χρήστη -εκτός από αυτόν που είχε συνδεθεί – τότε αυτός θα μπορεί να συνδεθεί στον ιστότοπο που είχε συνδεθεί ο προηγούμενος χρήστης χρησιμοποιώντας τα στοιχεία του τελευταίου.

Για την αποφυγή κλοπής διαπιστευτηρίων μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Επιβολή χρήσης ισχυρών κωδικών πρόσβασης.
- Οι κωδικοί πρόσβασης πρέπει να αποθηκεύονται σε μορφή σύνοψης μαζί με salt μιας διαδρομής.
- Επιβάλλεται κλείδωμα λογαριασμών για εκείνους στους οποίους γίνεται ένας καθορισμένος αριθμός αποτυχημένων προσπαθειών πρόσβασης.

- Για να αντιμετωπιστεί η δυνατότητα που έχουν οι browsers να αποθηκεύουν κωδικούς πρόσβασης, θα πρέπει να υλοποιείται στην εφαρμογή λειτουργία η οποία θα αφήνει τον χρήστη να αποφασίσει αν θα επιτρέπει την αποθήκευση διαπιστευτηρίων στον browser.

3.10.4 Εξουσιοδότηση (Authorization)

Η εξουσιοδότηση ενός χρήστη για πρόσβαση σε κάποιο συγκεκριμένο πόρο ή υπηρεσία της εφαρμογής επιτρέπεται ή απαγορεύεται ανάλογα με την ταυτότητά του και την ιδιότητά του να είναι μέλος κάποιου ρόλου. Οι απειλές που εκμεταλλεύονται αδυναμίες εξουσιοδότησης μιας εφαρμογής μπορεί να οδηγήσουν στις επιθέσεις που εξετάζονται στις παρακάτω υποπαραγράφους.

Elevation of Privilege

Όταν σχεδιάζεται ένα μοντέλο εξουσιοδότησης για μια εφαρμογή θα πρέπει να λαμβάνεται υπ' όψη η περίπτωση που ένας επιτιθέμενος θα προσπαθήσει να ανυψώσει τα δικαιώματά του σε αυτά ενός ισχυρού λογαριασμού της εφαρμογής. Με αυτόν τον τρόπο θα μπορέσει να αποκτήσει πλήρη έλεγχο της εφαρμογής ή του τοπικού συστήματος

Το μοναδικό αντίμετρο που μπορεί να ληφθεί για την αντιμετώπιση αυτής της απειλής είναι να χρησιμοποιούνται διεργασίες, υπηρεσίες και λογαριασμοί χρηστών με όσο το δυνατόν λιγότερα προνόμια.

Αποκάλυψη εμπιστευτικών πληροφοριών (Disclosure of Confidential Data)

Η αποκάλυψη εμπιστευτικών πληροφοριών προκύπτει όταν ευαίσθητα δεδομένα μπορούν να εμφανιστούν σε μη εξουσιοδοτημένους χρήστες. Στα εμπιστευτικά δεδομένα συγκαταλέγονται συγκεκριμένα δεδομένα μιας εφαρμογής όπως αριθμοί πιστωτικών καρτών, στοιχεία υπαλλήλων, λογιστικές εγγραφές, καθώς επίσης δεδομένα διαμόρφωσης όπως διαπιστευτήρια λογαριασμών χρηστών και συμβολοσειρές σύνδεσης με βάσεις δεδομένων. Για την αποτροπή αυτής της απειλής, τα παραπάνω θα πρέπει να τους παρέχεται προστασία σε κατάλληλες αποθήκες δεδομένων όπως βάσεις δεδομένων και αρχεία διαμόρφωσης, και κατά τη διάρκεια μετάδοσής τους μέσω του δικτύου. Μόνο οι αυθεντικοποιημένοι και εξουσιοδοτημένοι χρήστες θα πρέπει να έχουν πρόσβαση σε δεδομένα που προορίζονται για αυτούς. Η πρόσβαση σε δεδομένα διαμόρφωσης συστήματος θα πρέπει να επιτρέπεται μόνο στους διαχειριστές των συστημάτων αυτών.

Για την αποφυγή της παραπάνω απειλής μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Θα πρέπει να εκτελείται έλεγχος ρόλου πριν επιτραπεί η πρόσβαση ενός χρήστη σε λειτουργίες που μπορεί να αποκαλύψουν ευαίσθητα δεδομένα.
- Χρήση ισχυρών ACLs για να σφαλιστούν οι πόροι των λειτουργικών συστημάτων.
- Χρήση κρυπτογράφησης για την αποθήκευση ευαίσθητων δεδομένων σε αρχεία διαμόρφωσης και βάσεις δεδομένων.

Data Tampering

Για την αποφυγή της απειλής αλλοίωσης δεδομένων μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Χρήση ισχυρών διαδικασιών ελέγχου πρόσβασης έτσι ώστε να προστατευτούν τα δεδομένα που υπάρχουν στις αποθήκες της εφαρμογής και να εξασφαλιστεί ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση και να τροποποιήσουν τα δεδομένα αυτά.

- Χρήση ασφάλειας που βασίζεται σε ρόλους για να επιτευχθεί διαφοροποίηση μεταξύ των χρηστών που μπορούν απλά να δουν κάποια δεδομένα και αυτών που μπορούν να τα τροποποιήσουν.

Επιθέσεις παραπλάνησης (Luring Attacks)

Οι επιθέσεις αυτές προκύπτουν όταν μια οντότητα με περιορισμένα προνόμια καταφέρει να κάνει μια άλλη με περισσότερα προνόμια να εκτελέσει μια ενέργεια για λογαριασμό της.

Για την αντιμετώπιση της απειλής θα πρέπει να περιοριστεί η πρόσβαση σε έμπιστο κώδικα με τον κατάλληλο μηχανισμό εξουσιοδότησης. Το .NET Framework παρέχει έναν τέτοιο μηχανισμό με την ονομασία Code Access Security [22].

3.10.5 Διαχείριση διαμορφώσεων (Configuration Management)

Πολλές εφαρμογές υποστηρίζουν διεπαφές διαχείρισης διαμορφώσεων με την κατάλληλη λειτουργικότητα, που επιτρέπουν στους διαχειριστές να αλλάζουν παραμέτρους διαμόρφωσης, να ανανεώνουν το περιεχόμενο της εφαρμογής τους και γενικά να μπορούν να εκτελούν συντήρηση ρουτίνας. Οι κυριότερες απειλές που εκμεταλλεύονται αδυναμίες διαχείρισης διαμορφώσεων μπορεί να οδηγήσουν στις επιθέσεις που εξετάζονται στις παρακάτω υποπαραγράφους.

Μη εξουσιοδοτημένη πρόσβαση στις διεπαφές διαχειριστή

Συνήθως οι διεπαφές διαχείρισης παρέχονται μέσω επιπρόσθετων ιστοσελίδων ή ξεχωριστών εφαρμογών Διαδικτύου. Οι διεπαφές αυτές θα πρέπει να είναι διαθέσιμες στους κατάλληλα εξουσιοδοτημένους χρήστες. Κακόβουλοι χρήστες που καταφέρνουν να αποκτήσουν πρόσβασης σε μια λειτουργία διαχείρισης διαμόρφωσης μπορούν να τροποποιήσουν τα δεδομένα διαμόρφωσης και πιθανότατα να καταστρέψουν τον ιστότοπο, να αποκτήσουν πρόσβαση στις βάσεις δεδομένων ή και να καταστήσουν την εφαρμογή ανενεργή.

Για την αποφυγή απειλών τέτοιου είδους μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Ελαχιστοποίηση των διεπαφών διαχειριστή.
- Χρήση ισχυρών μηχανισμών αυθεντικοποίησης όπως για παράδειγμα με χρήση πιστοποιητικών.
- Χρήση ισχυρών μηχανισμών εξουσιοδότησης.
- Σε περίπτωση που είναι απαραίτητη απομακρυσμένη διαχείριση της εφαρμογής, θα πρέπει να χρησιμοποιούνται κρυπτογραφημένα κανάλια επικοινωνίας λόγω της ευαίσθητης φύσης των δεδομένων που μεταφέρονται μέσω των διεπαφών διαχείρισης. Επιπλέον, για περαιτέρω ασφάλεια, μπορούν να χρησιμοποιηθούν πολιτικές IPSec για να περιοριστεί η απομακρυσμένη διαχείριση σε υπολογιστές που βρίσκονται στο εσωτερικό δίκτυο.

Μη εξουσιοδοτημένη πρόσβαση στις αποθήκες διαμόρφωσης

Λόγω της ευαίσθητης φύσης των δεδομένων που διατηρούνται στις αποθήκες διαμόρφωσης (αρχεία, βάσεις δεδομένων), θα πρέπει να εξασφαλίζεται σε αυτές επαρκής ασφάλεια.

Για την προστασία τους μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Διαμόρφωση περιορισμένων ACLs σε αρχεία διαμόρφωσης κειμένου όπως τα machine.config και web.config.

- Διατήρηση των αποθηκών δεδομένων διαμόρφωσης εκτός του χώρου της εφαρμογής. Αυτό αποτρέπει την περίπτωση λήψης των στοιχείων διαμόρφωσης του server με σκοπό την κακόβουλη εκμετάλλευσή του.

Ανάκτηση μυστικών στοιχείων διαμόρφωσης υπό μορφή απλού κειμένου

Ως μηχανισμός άμυνας εις βάθος, θα πρέπει να κρυπτογραφούνται τα ευαίσθητα δεδομένα όπως οι κωδικοί πρόσβασης και οι συμβολοσειρές σύνδεσης με βάσεις δεδομένων όταν αυτά βρίσκονται στα αρχεία διαμόρφωσης ώστε να αποτρέπεται η λήψη τους από εξωτερικούς εισβολείς ή και ακόμα από κακόβουλους διαχειριστές.

Έλλειψη ατομικής υπευθυνότητας

Η έλλειψη ελέγχου και καταγραφής των αλλαγών που γίνονται στις πληροφορίες διαμόρφωσης μιας εφαρμογής, δεν επιτρέπει τον προσδιορισμό των αλλαγών που έγιναν και του ποιος έκανε αυτές τις αλλαγές. Όταν μια προβληματική αλλαγή συμβαίνει είτε από ένα διαχειριστή είτε από κάποιον κακόβουλο χρήστη, πρώτα πρέπει να γίνουν οι κατάλληλες ενέργειες ώστε να διορθωθεί αυτή η αλλαγή και στη συνέχεια πρέπει να εφαρμοστούν προληπτικά μέτρα για την αποτροπή εμφάνισης παρόμοιων αλλαγών.

Υπέρ-προνομιούχοι λογαριασμοί εφαρμογής και υπηρεσιών

Όταν στους λογαριασμούς εφαρμογής και υπηρεσιών χορηγούνται δικαιώματα να αλλάζουν δεδομένα διαμόρφωσης, μπορεί να χειραγωγηθούν από κάποιον επιτιθέμενο. Το ρίσκο που προκύπτει από αυτήν την απειλή μπορεί να ελαττωθεί αν υιοθετηθεί πολιτική χρήσης λογαριασμών με όσο το δυνατόν λιγότερα προνόμια. Μεγάλη προσοχή απαιτείται όταν χορηγείται στους λογαριασμούς ικανότητα να τροποποιούν τα ατομικά τους στοιχεία διαμόρφωσης, εκτός βέβαια αν απαιτείται ρητώς από την εφαρμογή.

3.10.6 Ευαίσθητα δεδομένα

Τα ευαίσθητα δεδομένα είναι υποκείμενα πολλών απειλών που προσπαθούν να αποκτήσουν πρόσβαση σε αυτά και να τα τροποποιήσουν. Οι κυριότερες απειλές που εκμεταλλεύονται ευπάθειες που αφορούν τα ευαίσθητα δεδομένα μιας εφαρμογής εξετάζονται στις παρακάτω υποπαραγράφους.

Πρόσβαση σε αποθηκευμένα ευαίσθητα δεδομένα

Τα ευαίσθητα δεδομένα πρέπει να ασφαλίζονται όταν είναι αποθηκευμένα, έτσι ώστε να εμποδίζεται ένας επιτιθέμενος να αποκτήσει πρόσβαση σε αυτά.

Για την προστασία τους μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Χρήση περιορισμένων ACLs στις αποθήκες των δεδομένων.
- Αποθήκευση των δεδομένων με κρυπτογράφηση.
- Χρήση μηχανισμών εξουσιοδότησης που βασίζεται σε ρόλους για να εξασφαλιστεί ότι μόνο χρήστες που διαθέτουν το απαιτούμενο επίπεδο εξουσιοδότησης μπορούν να έχουν πρόσβαση στα ευαίσθητα δεδομένα της εφαρμογής. Με αυτόν τον μηχανισμό επιτυγχάνεται διαφοροποίηση μεταξύ των χρηστών που μπορούν απλά να δουν κάποια δεδομένα και αυτών που μπορούν να τα τροποποιήσουν.

Network Eavesdropping

Τα δεδομένα μιας διαδικτυακής εφαρμογής ταξιδεύουν μέσω του δικτύου υπό μορφή απλού κειμένου, μπορούν να απειληθούν από ωτακουστές δικτύου, να υποκλαπούν και στη συνέχεια να τροποποιηθούν.

Για την αποφυγή απειλών network eavesdropping στο επίπεδο δικτύου μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Κρυπτογράφηση των δεδομένων κατά την αποστολή τους.
- Χρήση κρυπτογραφημένου καναλιού επικοινωνίας όπως είναι το SSL.

Αλλοίωση δεδομένων

Ένα από τα σημαντικότερα αντίμετρα που μπορεί να ληφθεί για την αντιμετώπιση απειλών αλλοίωσης δεδομένων είναι η προστασία τους με χρήση ανθεκτικών στην αλλοίωση πρωτοκόλλων όπως η χρήση σύνοψης κωδικού αυθεντικοποίησης μηνύματος (hashed message authentication codes - HMACs).

Ένας HMACs παρέχει ακεραιότητα μηνύματος με τον παρακάτω τρόπο:

- ✓ Ο αποστολέας χρησιμοποιεί ένα διαμοιραζόμενο κλειδί για να δημιουργήσει μια σύνοψη μηνύματος ανάλογα με τα δεδομένα του.
- ✓ Ο αποστολέας εκπέμπει τη σύνοψη μαζί με τα δεδομένα του μηνύματος.
- ✓ Ο παραλήπτης χρησιμοποιεί το κλειδί και υπολογίζει ξανά τη σύνοψη των δεδομένων του μηνύματος που έλαβε. Στη συνέχεια τη συγκρίνει με αυτή που έλαβε από τον αποστολέα και εφόσον είναι ίδιες, τότε τα δεδομένα του μηνύματος αποκλείεται να έχουν αλλοιωθεί.

3.10.7 Διαχείριση συνόδου (Session Management)

Η διαδικασία διαχείρισης συνόδου μιας διαδικτυακής εφαρμογής αποτελεί ευθύνη του επιπέδου εφαρμογής και είναι πολύ σημαντική για την συνολική ασφάλειά της. Οι κυριότερες απειλές που εκμεταλλεύονται ευπάθειες διαχείρισης συνόδου εξετάζονται στις παρακάτω υποπαραγράφους.

Υφαρπαγή συνόδου

Απειλή υφαρπαγής συνόδου συμβαίνει όταν ένας επιτιθέμενος χρησιμοποιεί λογισμικό παρακολούθησης δικτύου με σκοπό να υποκλέψει κάποιο τεκμήριο αυθεντικοποίησης (συνήθως cookie) που αντιπροσωπεύει τη σύνοδο ενός χρήστη μιας εφαρμογής, με την εφαρμογή. Έχοντας καταλάβει το cookie, ο επιτιθέμενος μπορεί να ξεγελάσει την εφαρμογή και να αποκτήσει πρόσβαση σε αυτή με την ταυτότητα του εξαπατημένου χρήστη.

Για την αποφυγή απειλών υφαρπαγής συνόδου μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Χρήση πρωτοκόλλου SSL για δημιουργία ασφαλούς καναλιού επικοινωνίας και μετάδοση του cookie μέσω σύνδεσης HTTPS.
- Υλοποίησης λειτουργικότητας αποσύνδεσης που επιτρέπει σε έναν χρήστη να τερματίζει πρώτα μια σύνδεση που απαιτεί αυθεντικοποίηση, πριν ξεκινήσει επόμενη.
- Εξασφάλιση ότι υπάρχει χρονικό όριο λήξης του cookie εφόσον δεν χρησιμοποιείται SSL. Αν και αυτό δεν αποτρέπει την υφαρπαγή συνόδου, ελαττώνει το χρονικό διάστημα που έχει στη διάθεσή του ο επιτιθέμενος.

Επαναληπτική εκτέλεση συνόδου (Session Replay)

Η συγκεκριμένη απειλή προκύπτει όταν ένας επιτιθέμενος υποκλέψει ένα τεκμήριο αυθεντικοποίησης ενός χρήστη μιας εφαρμογής και το υποβάλλει μετέπειτα με σκοπό να παρακάμψει το μηχανισμό αυθεντικοποίησης της εφαρμογής.

Για την αποφυγή απειλών επαναληπτικής εκτέλεσης συνόδου μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Επανάληψη αυθεντικοποίησης χρήστη όταν πρόκειται να εκτελεστούν κρίσιμες ενέργειες.
- Λήξη συνόδου με κατάλληλο τρόπο (συμπεριλαμβανομένου όλων των cookies και των τεκμηρίων συνόδου).
- Δημιουργία επιλογής “do not remember me” έτσι ώστε να μην αποθηκεύονται δεδομένα συνόδου στην πλευρά του client.

Επίθεση ενδιάμεσου (Man in the Middle Attack)

Η επίθεση ενδιάμεσου συμβαίνει όταν ένας επιτιθέμενος υποκλέπτει ένα μήνυμα του αποστολέα προς τον επιθυμητό του παραλήπτη, στη συνέχεια το τροποποιεί και το στέλνει στον παραλήπτη που έπρεπε να είχε πάει από την αρχή. Ο παραλήπτης το διαβάζει, εκτελεί τις ενέργειες που πρέπει βάσει αυτού και στέλνει απάντηση στον αρχικό αποστολέα. Όπως και πριν, ο επιτιθέμενος υποκλέπτει και αυτό το μήνυμα, το τροποποιεί και το στέλνει στον αρχικό αποστολέα. Έτσι, κανένας από τους δύο δεν έχει καταλάβει ότι δέχτηκε επίθεση.

Οποιοδήποτε αίτημα - μήνυμα που αποστέλλεται μέσω δικτύου όπως η επικοινωνία client - server, είναι ευάλωτο σε τέτοιου είδους επιθέσεις.

Για την αποφυγή απειλών man in the middle μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Χρήση κρυπτογραφίας. Με την κρυπτογράφηση των δεδομένων πριν τη μετάδοσή τους, ένας επιτιθέμενος δεν μπορεί να τα διαβάζει ή να τα τροποποιήσει, παρόλο που μπορεί να τα υποκλέψει. Ακόμα κι αν προσπαθήσει να τα τροποποιήσει στα τυφλά, ο παραλήπτης του μηνύματος δεν θα μπορέσει να τα αποκρυπτογραφήσει και συνεπώς θα καταλάβει ότι έχουν αλλοιωθεί.
- Χρήση HMACs. Στην περίπτωση που ο επιτιθέμενος καταφέρει να τροποποιήσει το μήνυμα, δεν θα επιτευχθεί σωστός επανυπολογισμός του HMAC στον παραλήπτη, άρα τα δεδομένα θα απορριφθούν ως άκυρα.

3.10.8 Κρυπτογραφία

Οι περισσότερες εφαρμογές χρησιμοποιούν την κρυπτογραφία για να προστατέψουν τα δεδομένα τους και να τα διατηρήσουν αναλλοίωτα και απροσπέλαστα. Οι κυριότερες απειλές που εκμεταλλεύονται ευπάθειες κρυπτογραφίας εξετάζονται στις παρακάτω υποπαραγράφους.

Αδύναμος μηχανισμός δημιουργίας ή διαχείρισης κλειδιών

Ένας επιτιθέμενος μπορεί να αποκρυπτογραφήσει τα κρυπτογραφημένα δεδομένα που υποκλέπτει, αν καταφέρει να αποκτήσει πρόσβαση στο κλειδί κρυπτογράφησής τους ή αν μπορεί να το βρει με κάποιο τρόπο. Ένα κλειδί μπορεί να βρεθεί αν δεν διαχειρίζεται κατάλληλα ή αν δημιουργήθηκε με μη τυχαίο τρόπο.

Για την αποφυγή της παραπάνω απειλής, μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Χρήση πολιτικής κρυπτογράφησης που περιλαμβάνει ασφαλή διαχείριση των κλειδιών.
- Χρήση ισχυρών συναρτήσεων παραγωγής κλειδιών με τυχαίο τρόπο και διαφύλαξη των κλειδιών σε περιορισμένες τοποθεσίες, όπως για παράδειγμα ένα κλειδί της registry το οποίο ασφαλίζεται με περιοριστική ACL.

- Κρυπτογράφηση του κρυπτογραφημένου κλειδιού για δύο στάδια ασφάλειας.
- Λήξη των κλειδιών σε τακτά χρονικά διαστήματα.

Ασθενής κρυπτογράφηση

Ένας αλγόριθμος κρυπτογράφησης δεν θα παρέχει καμία ασφάλεια αν η κρυπτογράφηση παραβιαστεί ή αν είναι ευάλωτος σε επιθέσεις brute force. Οι αλγόριθμοι που δεν είναι ευρέως γνωστοί είναι ιδιαίτερα ευάλωτοι, εφόσον δεν έχουν δοκιμαστεί. Αντίθετα, οι πολύ γνωστοί που έχουν δημοσιευθεί και δοκιμαστεί για αρκετά χρόνια είναι πιο ανθεκτικοί.

Για την αποφυγή των επιθέσεων που προκύπτουν από ασθενή κρυπτογράφηση, μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Δεν θα πρέπει να αναπτύσσονται προσωπικοί αλγόριθμοι κρυπτογράφησης.
- Θα πρέπει να χρησιμοποιούνται οι αλγόριθμοι που παρέχονται από την πλατφόρμα του συστήματος που βρίσκεται η εφαρμογή.
- Διαρκής ενημέρωση σχετικά με αλγόριθμους κρυπτογράφησης που «σπάσανε» και με ποιο τρόπο.

Έλεγχος πλαστογράφησης (Checksum Spoofing)

Δεν θα πρέπει κάποιος να βασίζεται σε συνόψεις μηνυμάτων που αποστέλλονται μέσω δικτύων, για να υπάρχει ακεραιότητα στο μήνυμα που αποστέλλεται. Οι αλγόριθμοι Secure Hash Algorithm (SHA1) και Message Digest (MD5) μπορούν να υποκλαπούν και συνεπώς να αλλοιωθεί το περιεχόμενο του μηνύματος.

Έστω ότι κάποιος στείλει το μήνυμα με τον κώδικα MAC:

```
Plaintext: Place 10 orders.
Hash: T0mUNdEQh13IO9oTcaP4FYDX6pU=
```

Έστω ότι ένας επιτιθέμενος είχε υποκλέψει το μήνυμα, το τροποποιήσει και καταφέρει να υπολογίσει το καινούριο MAC το μήνυμα θα γινόταν :

```
Plaintext: Place 100 orders.
Hash: oEDuJpv/ZtIU7BXDDNv17EAHeAU=
```

Όταν ο παραλήπτης επεξεργαστεί το μήνυμα και επαναϋπολογίσει τη σύνοψη, δεν θα υπάρχει αναντιστοιχία και άρα το μήνυμα θα θεωρηθεί ακέραιο ενώ δεν είναι.

Για την αντιμετώπιση αυτής της απειλής πρέπει να χρησιμοποιείται ο αλγόριθμος Message Authentication Code Triple Data Encryption Standard (MACTripleDES). Ο αλγόριθμος υπολογίζει ένα MAC για το μήνυμα και ένα HMAC. Οι δύο κωδικοί αυτοί χρησιμοποιούν ένα κλειδί και παράγουν ένα άθροισμα ελέγχου. Με αυτόν τον αλγόριθμο, ένας επιτιθέμενος θα πρέπει να ξέρει το κλειδί δημιουργίας αθροίσματος ελέγχου για να τροποποιήσει ένα μήνυμα και να επαναϋπολογίσει το νέο άθροισμα που θα έπρεπε να έχει το τροποποιημένο μήνυμα.

3.10.9 Τροποποίηση παραμέτρων (Parameter Manipulation)

Οι επιθέσεις τροποποίησης παραμέτρων συμβαίνουν όταν τροποποιούνται τα δεδομένα παραμετροποίησης που στέλνονται από τον client στην εφαρμογή. Τα δεδομένα αυτά μπορεί να υπάρχουν σε query strings, σε πεδία φορμών της εφαρμογής, σε cookies ή σε επικεφαλίδες HTTP.

Οι κυριότερες απειλές που εκμεταλλεύονται αδυναμίες τροποποίησης παραμέτρων εξετάζονται στις παρακάτω υποπαραγράφους.

Τροποποίηση Query String

Οποιοσδήποτε χρήστης μπορεί εύκολα να τροποποιήσει τις τιμές του query string που μεταδίδονται μέσω μηνύματος HTTP GET από τον client στον server της εφαρμογής, γιατί αυτές φαίνονται στη μπάρα διευθύνσεων URL του φυλλομετρητή. Εφαρμογές που χρησιμοποιούν query strings για να λάβουν αποφάσεις ασφάλειας ή για να μεταδώσουν ευαίσθητα δεδομένα είναι ευπαθείς σε τέτοιες απειλές.

Για την αποφυγή των επιθέσεων που προκύπτουν από ασθενή κρυπτογράφηση, μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Αποφυγή χρήσης παραμέτρων query strings που περιέχουν ευαίσθητα δεδομένα ή μπορούν να επηρεάσουν την ασφάλεια του server. Αντί αυτού, καλό είναι να χρησιμοποιείται κάποιο αναγνωριστικό συνόδου για να αναγνωριστεί ο χρήστης και για να αποθηκευτούν τα ευαίσθητα δεδομένα που θα χρειαστούν για τη σύνοδο στον server.
- Επιλογή μεθόδου αποστολής HTTP POST αντί για GET για την υποβολή φόρμας δεδομένων από τον client στον server.
- Κρυπτογράφηση των παραμέτρων που μεταδίδονται μέσω του query string.

Τροποποίηση πεδίων φόρμας

Τα δεδομένα των πεδίων μιας φόρμας, ορατά ή όχι, στέλνονται στον server υπό μορφή απλού κειμένου με μέθοδο HTTP POST. Τα πεδία αυτά μπορούν να τροποποιηθούν εύκολα και η επικύρωσή τους στη μεριά του client μπορεί εύκολα να παρακαμφθεί.

Για την αντιμετώπιση επιθέσεων τροποποίησης φόρμας δεδομένων, αντί της χρήσης κρυφών πεδίων, καλό είναι να χρησιμοποιείται αναγνωριστικό συνόδου για διατήρηση της κατάστασης της συνόδου στην πλευρά του server.

Τροποποίηση Cookie

Τα cookies είναι επίσης εύκολο να τροποποιηθούν στη μεριά του client, είτε αυτά είναι μόνιμα αποθηκευμένα στη μνήμη του περιηγητή είτε όχι. Υπάρχουν πολλά εργαλεία στη διάθεση ενός επιτιθέμενου για να μπορέσει αυτός να τροποποιήσει τα περιεχόμενα ενός cookie. Η επίθεση τροποποίησης cookie συνήθως χρησιμοποιείται για να αποκτήσει κάποιος μη εξουσιοδοτημένος χρήστης πρόσβαση σε έναν ιστότοπο.

Με το πρωτόκολλο SSL μπορεί να προστατευτούν τα cookies που μεταδίδονται μέσω δικτύου, αλλά δεν μπορεί να τα προστατέψει όταν αυτά βρίσκονται στην μεριά του client. Για την αντιμετώπιση αυτού του προβλήματος τα cookies θα πρέπει να κρυπτογραφούνται και να χρησιμοποιείται και HMAC.

Τροποποίηση επικεφαλίδας HTTP

Οι επικεφαλίδες HTTP χρησιμοποιούνται για να περάσουν διάφορες πληροφορίες μεταξύ client και server. Οι clients δημιουργούν επικεφαλίδες αιτημάτων ενώ οι servers επικεφαλίδες αποκρίσεων. Όταν μια εφαρμογή χρησιμοποιεί τις επικεφαλίδες HTTP για να λάβει κρίσιμες αποφάσεις είναι ευάλωτη σε τέτοιες επιθέσεις τροποποίησης επικεφαλίδων HTTP, κάτι που θα πρέπει να αποφεύγεται.

3.10.10 Διαχείριση εξαιρέσεων

Οι εξαιρέσεις που επιτρέπεται να γνωστοποιηθούν στον client μπορούν να αποκαλύψουν τις λεπτομέρειες υλοποίησης της εφαρμογής οι οποίες μπορεί να μην είναι χρήσιμες στους απλούς χρήστες αλλά είναι στους επιτιθέμενους. Οι εφαρμογές που δεν υλοποιούν μηχανισμούς διαχείρισης εξαιρέσεων ή του υλοποιούν αναποτελεσματικά είναι ευάλωτες σε επιθέσεις άρνησης παροχής υπηρεσιών, όπως αναλύεται και παρακάτω.

Αποκάλυψη λεπτομερειών υλοποίησης εφαρμογής

Ένα από τα πιο σημαντικά χαρακτηριστικά που περιλαμβάνεται στο .NET Framework είναι οι πλούσιες λεπτομέρειες εξαιρέσεων που βοηθούν στο έπακρο τη δουλειά των προγραμματιστών. Αν αυτές οι πληροφορίες πέσουν στα χέρια κάποιου επιτιθέμενου μπορούν να τον βοηθήσουν για την κακόβουλη εκμετάλλευση της εφαρμογής και για την μελλοντική σχεδίαση άλλου είδους επιθέσεων. Μερικές συνηθισμένες πληροφορίες που παρέχονται από τις εξαιρέσεις είναι η έκδοση του λειτουργικού συστήματος του host, ονόματα των servers που χρησιμοποιούνται από την εφαρμογή, συμβολοσειρές εντολών SQL και συμβολοσειρές σύνδεσης με βάσεις δεδομένων.

Για την αποφυγή αποκάλυψης πληροφοριών υλοποίησης εφαρμογής στον client μέσω εξαιρέσεων, μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Χρήση μηχανισμών διαχείρισης εξαιρέσεων στον κώδικα της εφαρμογής.
- Διαχείριση και καταγραφή των εξαιρέσεων που επιτρέπεται να γνωστοποιηθούν στον client.
- Σε περιπτώσεις σφάλματος της εφαρμογής θα πρέπει να επιστρέφονται πολύ γενικά μηνύματα στον client.

Άρνηση παροχής υπηρεσίας

Οι επιτιθέμενοι συνήθως δοκιμάζουν μια εφαρμογή Διαδικτύου στέλνοντάς της, εκούσια, κακοσχηματισμένα δεδομένα εισόδου. Ο στόχος τους είναι διπλός: είτε να προκαλέσουν εξαιρέσεις που θα αποκαλύψουν χρήσιμες για αυτούς πληροφορίες της εφαρμογής, είτε να καταστρέψουν την διεργασία της εφαρμογής. Αυτά θα μπορέσουν να τα καταφέρουν εφόσον οι εξαιρέσεις που προκύπτουν δεν πιάνονται έγκαιρα και δεν διαχειρίζονται κατάλληλα.

Για την αποφυγή άρνησης παροχής υπηρεσιών στο επίπεδο της εφαρμογής, μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Λεπτομερής επικύρωση όλων των δεδομένων εισόδου προς τον server.
- Χρήση μηχανισμών διαχείρισης εξαιρέσεων στον κώδικα της εφαρμογής.

3.10.11 Παρακολούθηση και καταγραφή ιστορικού

Η παρακολούθηση της κίνησης μιας εφαρμογής και η καταγραφή του ιστορικού της κίνησης θα πρέπει να χρησιμοποιείται για τον εντοπισμό ύποπτης δραστηριότητας χρηστών με σκοπό την πρόληψη επερχόμενης επίθεσης. Βοηθάει επίσης στην αντιμετώπιση απειλών τύπου απάρνησης. Αυτό συμβαίνει γιατί είναι δύσκολο για ένα χρήστη της εφαρμογής να αρνηθεί ότι εκτέλεσε κάποια λειτουργία αν υπάρχουν συγχρονισμένες καταγραφές σε πολλούς servers που καταδεικνύουν ότι εκτέλεσε τις λειτουργίες αυτές.

Οι κυριότερες απειλές που εκμεταλλεύονται ευπάθειες παρακολούθησης και καταγραφής ιστορικού αρχείου εξετάζονται στις παρακάτω υποπαραγράφους.

Άρνηση χρήστη ότι εκτέλεσε μια ενέργεια

Απαιτούνται αμυντικοί μηχανισμοί που να διασφαλίζουν ότι παρακολουθείται και καταγράφεται η δραστηριότητα οποιουδήποτε χρήστη της εφαρμογής.

Για την αποφυγή απάρνησης στο επίπεδο της εφαρμογής, μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Παρακολούθηση και καταγραφή δραστηριότητας στους servers που χρησιμοποιεί η εφαρμογή.
- Καταγραφή των πιο σημαντικών γεγονότων όπως σύνδεση και αποσύνδεση στην εφαρμογή.
- Δεν πρέπει να χρησιμοποιούνται διαμοιραζόμενοι λογαριασμοί χρηστών γιατί η πραγματική ταυτότητα του χρήστη που εκτέλεσε τις ενέργειες δεν μπορεί να εντοπιστεί

Εκμετάλλευση εφαρμογής χωρίς να αφήνονται ίχνη από τους επιτιθέμενους

Απαιτείται παρακολούθηση του συστήματος και της εφαρμογής έτσι ώστε να διασφαλίζεται ότι οποιαδήποτε ύποπτη δραστηριότητα συμβεί, θα εντοπιστεί.

Για τον εντοπισμό ύποπτης δραστηριότητας μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Καταγραφή των σημαντικών λειτουργιών σε επίπεδο εφαρμογής.
- Παρακολούθηση και καταγραφή στο επίπεδο της πλατφόρμας υλοποίησης της εφαρμογής συμβάντων σύνδεσης και αποσύνδεσης, πρόσβασης στα αρχεία του συστήματος είτε επιτυχημένη είτε αποτυχημένη.
- Δημιουργία αντιγράφων ασφάλειας των αρχείων καταγραφής συμβάντων και ανάλυσή τους σε τακτά χρονικά διαστήματα για εντοπισμό ύποπτης δραστηριότητας.

Κάλυψη ιχνών επίθεσης

Τα αρχεία καταγραφής θα πρέπει να προστατεύονται κατάλληλα για να μην μπορούν οι επιτιθέμενοι να καλύψουν τα ίχνη τους.

Για την αποτροπή τέτοιου είδους επιθέσεων μπορούν να ληφθούν τα παρακάτω αντίμετρα:

- Ασφάλιση αρχείων καταγραφής με ACLs.
- Επανατοποθέτηση των αρχείων καταγραφής του συστήματος σε άλλη θέση από την προκαθορισμένη.

3.11 Σύνοψη αντιμέτρων επιπέδου εφαρμογής

Στον Πίνακας 8 που ακολουθεί, παρουσιάζονται συνολικά τα αντίμετρα που θα πρέπει να υλοποιούνται για την αντιμετώπιση των απειλών που υπάρχουν σε κάθε αδυναμία του επιπέδου εφαρμογής.

Κατηγορία Αδυναμίας	Απειλές	Αντίμετρα
Επικύρωση δεδομένων εισόδου	Buffer overflow, cross-site scripting, SQL injection, canonicalization	Εξονυχιστικοί έλεγχοι (φιλτράρισμα) δεδομένων εισόδου Μηχανισμοί μετατροπής scripts σε αβλαβή
Αυθεντικοποίηση	brute force, network eavesdropping, dictionary attacks, cookie replay, κλοπή	Κρυπτογράφηση και χρήση σύνθετων κωδικών πρόσβασης,

	διαπιστευτηρίων	SSL, hashing, cookie timeout
Εξουσιοδότηση	Κλιμάκωση δικαιωμάτων, αποκάλυψη εμπιστευτικών δεδομένων, αλλοίωση δεδομένων, παραπλάνηση (luring)	Διεργασίες least privileged, role authorization, κρυπτογράφηση ευαίσθητων δεδομένων
Διαχείριση διαμορφώσεων	Μη εξουσιοδοτημένη πρόσβαση στις διεπαφές διαχείρισης ή σε άλλα σημεία διαμόρφωσης, ανάκτηση δεδομένων ρυθμίσεων σε μορφή απλού κειμένου, έλλειψη ατομικής ευθύνης, διεργασίες και λογαριασμοί υπηρεσιών με περισσότερα προνόμια από τα απαιτούμενα.	Ελαχιστοποίηση διεπαφών, χρήση μηχανισμών αυθεντικοποίησης – εξουσιοδότησης, SSL, IPSEC, κρυπτογράφηση
Ευαίσθητα δεδομένα	Πρόσβαση στις αποθήκες ευαίσθητων δεδομένων, network eavesdropping, αλλοίωση δεδομένων	Κρυπτογράφηση, μηχανισμοί εξουσιοδότησης, SSL, HMACS
Διαχείριση συνόδων	Υφαρπαγή συνεδρίας, session replay, επίθεση man in the middle	SSL, cookie timeout, επανάληψη αυθεντικοποίησης, λήξη συνόδου κατάλληλα, κρυπτογραφία, HMACS
Κρυπτογραφία	Αδύναμος μηχανισμός δημιουργίας ή διαχείρισης κλειδιών, χρήση συνηθισμένων τεχνικών κρυπτογράφησης	Χρήση ισχυρών συναρτήσεων, λήξη κλειδιών, κρυπτογράφηση 2 σταδίων, χρήση αποδεδειγμένα αξιόπιστων αλγορίθμων
Χειρισμός παραμέτρων	Κακόβουλος χειρισμός των: Query string, πεδίων φόρμας, cookie και επικεφαλίδων HTTP	Χρήση αναγνωριστικών συνόδου, HTTP POST, κρυπτογράφηση,
Διαχείριση εξαιρέσεων	Αποκάλυψη πληροφοριών, άρνηση παροχής υπηρεσίας	Διαχείριση και καταγραφή των εξαιρέσεων, γενικά μηνύματα σφαλμάτων στον client, επικύρωση δεδομένων εισόδου
Παρακολούθηση και καταγραφή	Άρνηση ενός χρήστη ότι εκτέλεσε μια εργασία, κάλυψη ιχνών επιτιθέμενου, εκμετάλλευση εφαρμογής χωρίς να αφήνονται ίχνη	Παρακολούθηση και καταγραφή δραστηριότητας, όχι διαμοιραζόμενοι λογαριασμοί χρηστών, αντίγραφα ασφαλείας, προστασία αρχείων καταγραφής

Πίνακας 8. Σύνοψη αντιμέτρων ανάλογα με τις απειλές στο επίπεδο εφαρμογής

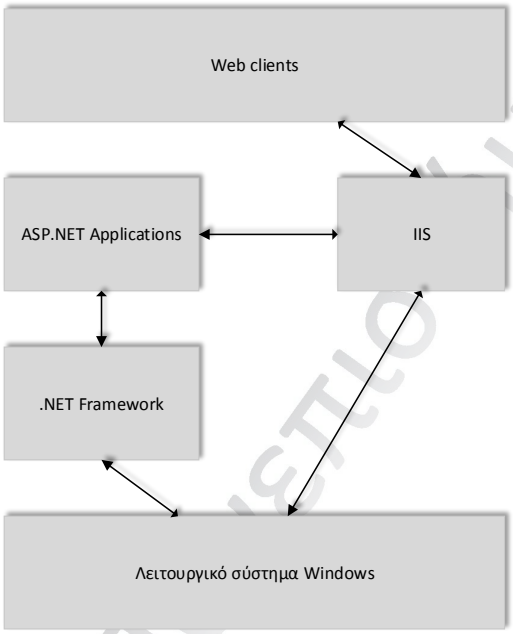
Κεφάλαιο 4°

4 Μελέτη τεχνολογιών ασφαλείας .NET Framework

Το ASP.NET παρέχει ένα πολυεπίπεδο μοντέλο ασφάλειας για την προστασία των διαδικτυακών εφαρμογών. Λόγω των πολλαπλών επιπέδων στα οποία μπορούν να υλοποιηθούν τεχνολογίες ασφάλειας, το μοντέλο αυτό φαίνεται πολύπλοκο. Πολλές φορές όμως η αποτελεσματικότητα των μέτρων ασφάλειας μιας εφαρμογής δεν εξαρτάται από τον κώδικα με τον οποίο γράφεται η εφαρμογή αλλά με το αν εφαρμόζονται πολιτικές ασφάλειας στα σημεία που απαιτείται. Ο σκοπός του Κεφαλαίου είναι να διαχωριστούν τα υποσυστήματα ασφάλειας του .NET έτσι ώστε να γίνει πιο εύκολη η διαδικασία χρήσης των υποσυστημάτων αυτών για την ανάπτυξη της διαδικτυακής εφαρμογής της παρούσας εργασίας.

4.1 Μοντέλο ασφάλειας ASP.NET

Οι αιτήσεις clients προς έναν εξυπηρετητή ιστού, φιλτράρονται πρώτα από τον IIS. Στη συνέχεια, εφόσον η αίτηση ικανοποιεί κάποιες προϋποθέσεις, προωθείται στο ASP.NET. Στο παρακάτω σχεδιάγραμμα απεικονίζεται η αλληλεπίδραση των clients, του IIS και του ASP.NET όταν αιτείται μια ιστοσελίδα από έναν server.



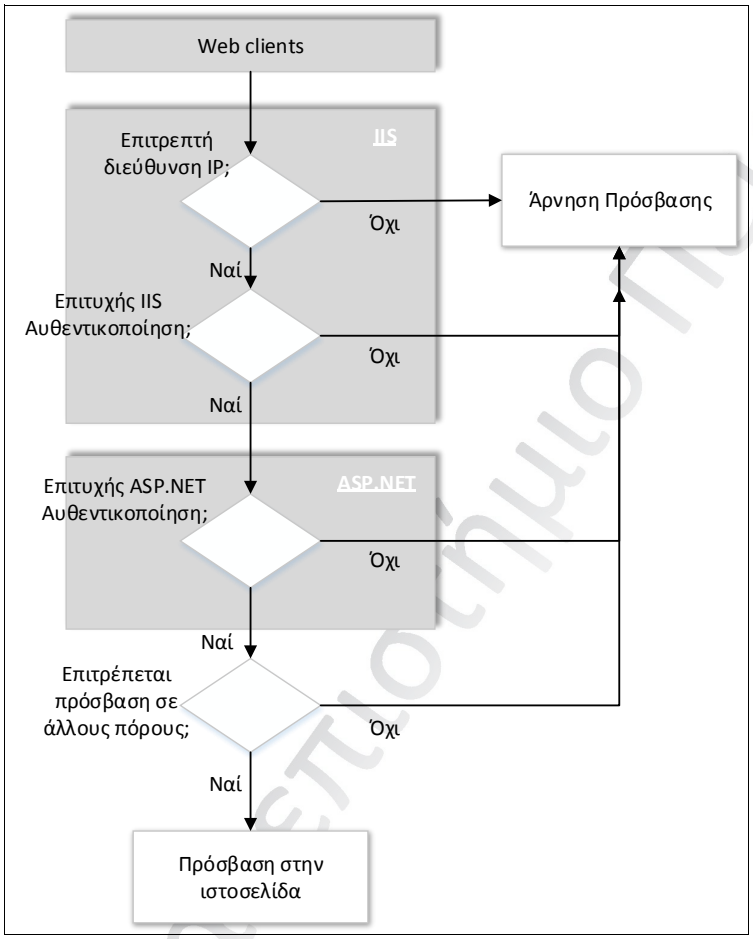
Εικόνα 8. Αλληλεπίδραση IIS, ASP.NET και clients κατά την αίτηση ιστοσελίδας.

Στην αλυσίδα που δημιουργείται από αυτήν την αλληλεπίδραση μπορούμε να εφαρμόσουμε τους επιθυμητούς μηχανισμούς ασφάλειας.

Λεπτομερέστερα, τα βήματα που ακολουθούνται για την ικανοποίηση ενός αιτήματος ιστοσελίδας είναι τα παρακάτω:

1. Ο IIS προσπαθεί να αυθεντικοποιήσει τον χρήστη που απέστειλε το αίτημα. Σε γενικές γραμμές επιτρέπει όλα τα αιτήματα ανώνυμων χρηστών, εκτός αν έχουν τροποποιηθεί οι προκαθορισμένες ρυθμίσεις του.

2. Εφόσον ο χρήστης αυθεντικοποιηθεί, το αίτημα μεταφέρεται στο ASP.NET μαζί με κάποιες επιπλέον πληροφορίες για τον χρήστη. Το ASP.NET μπορεί να χρησιμοποιήσει τη δικιά του υπηρεσία αυθεντικοποίησης, ανάλογα με τις ρυθμίσεις που υπάρχουν στο αρχείο web.config και την ιστοσελίδα που ζητήθηκε από τον client.
3. Όταν και το ASP.NET αυθεντικοποιήσει τον χρήστη, τότε επιτρέπει την πρόσβαση στην ιστοσελίδα που αιτήθηκε.
4. Στην περίπτωση που ο κώδικας απαιτεί χρήση επιπλέον πόρων (π.χ. πρόσβαση σε κάποια βάση δεδομένων ή άνοιγμα κάποιου αρχείου) το λειτουργικό σύστημα θα εκτελέσει τους δικούς του ελέγχους ασφάλειας.



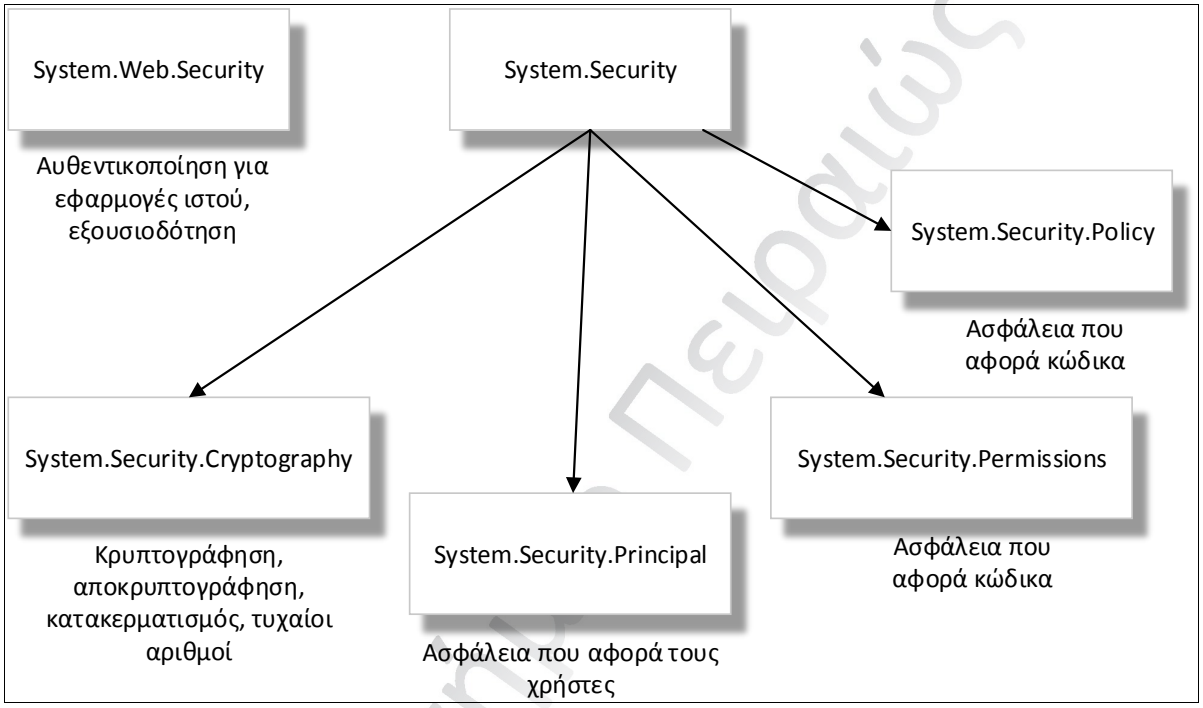
Εικόνα 9. Έλεγχος αυθεντικοποίησης αιτήματος client.

4.2 Κλάσεις ασφάλειας του .NET

Η βιβλιοθήκη κλάσεων του .NET περιλαμβάνει τους παρακάτω χώρους ονομάτων που χρησιμοποιούνται για την υλοποίηση μηχανισμών ασφάλειας σε εφαρμογές, είτε αυτές είναι διαδικτυακές είτε όχι. Οι χώροι ονομάτων που χρησιμοποιούνται για τον προγραμματισμό λειτουργιών ασφαλείας είναι οι παρακάτω :

- System.Security

- System.Web.Security
- System.Security.Cryptography
- System.Security.Principal
- System.Security.Policy
- System.Security.Permissions



Εικόνα 10. Χώροι ονομάτων ασφάλειας .NET

4.2.1 Χώρος ονομάτων System.Web.Security

Ο χώρος ονομάτων System.Web.Security περιέχει τις κλάσεις που χρησιμοποιούνται για την υλοποίηση των μηχανισμών αυθεντικοποίησης και εξουσιοδότησης χρηστών στις διαδικτυακές εφαρμογές. Οι κλάσεις που χρησιμοποιούνται περισσότερο είναι οι παρακάτω:

- FormsAuthentication: Παρέχει στατικές μεθόδους για την αυθεντικοποίηση χρηστών με φόρμες και για τη διαχείριση του εισιτηρίου αυθεντικοποίησης.
- FormsIdentity: Χρησιμοποιείται για την ενθυλάκωση της ταυτότητας ενός αυθεντικοποιημένου χρήστη μέσω φόρμας.
- Membership: Χρησιμοποιείται για την επικύρωση των διαπιστευτηρίων των χρηστών της εφαρμογής και για τη διαχείριση των ρυθμίσεων των λογαριασμών τους (κωδικοί πρόσβασης, διευθύνσεις email κτλ.).
- SqlMembershipProvider: Χρησιμοποιείται για την αποθήκευση των δεδομένων των χρηστών σε βάση δεδομένων στον Microsoft SQL Server.
- Roles: Η συγκεκριμένη κλάση επιτρέπει τη διαχείριση των ρυθμίσεων εξουσιοδότησης με βάση ομάδες χρηστών σε καθεμιά από τις οποίες έχει ανατεθεί κάποιος ρόλος.

- SqlRoleProvider: Χρησιμοποιείται για την αποθήκευση των ρόλων των χρηστών σε βάση δεδομένων στον Microsoft SQL Server.

Οι Membership και Roles αποτελούν δύο από τις πιο σημαντικές κλάσεις καθώς χρησιμοποιούνται για την υλοποίηση των βασικών λειτουργιών αυθεντικοποίησης και εξουσιοδότησης μιας εφαρμογής. Συνεργάζονται με κλάσεις γνωστές ως providers, οι οποίες έχουν πρόσβαση στις αντίστοιχες αποθήκες δεδομένων της εφαρμογής και μπορούν να ανακτήσουν πληροφορίες σχετικά με τους χρήστες και τους ρόλους τους. Οι πληροφορίες που αφορούν χρήστες και ρόλους μπορούν να αποθηκευθούν σε βάση δεδομένων Microsoft SQL χρησιμοποιώντας τις κλάσεις SqlMembershipProvider και SqlRoleProvider. Οι provider κλάσεις ανήκουν και αυτές στο χώρο ονομάτων System.Web.Security.

4.3 Membership Class

Όπως αναφέρθηκε στην προηγούμενη παράγραφο, η κλάση Membership χρησιμοποιείται για επικύρωση των διαπιστευτηρίων ενός χρήστη μιας εφαρμογής και για τη διαχείριση των ρυθμίσεων των λογαριασμών τους. Μπορεί επίσης να συνεργαστεί με την κλάση FormsAuthentication με σκοπό τη δημιουργία ενός πλήρους συστήματος αυθεντικοποίησης χρηστών για μια εφαρμογή διαδικτύου. Συγκεκριμένα, οι δυνατότητες που προσφέρει είναι:

- Δημιουργία νέων χρηστών.
- Αποθήκευση πληροφοριών χρηστών της εφαρμογής (όνομα, κωδικός πρόσβασης, διεύθυνση email και οποιαδήποτε επιπλέον δεδομένα σχετικά με τους χρήστες απαιτεί η εφαρμογή).
- Αυθεντικοποίηση χρηστών που επισκέπτονται την εφαρμογή.
- Διαχείριση κωδικών πρόσβασης. Δημιουργία, αλλαγή, ανάκτηση και επαναφορά τους. Επιπλέον μπορεί να γίνει η κατάλληλη ρύθμιση έτσι ώστε να απαιτείται η απάντηση σε μια μυστική ερώτηση σε περίπτωση που αιτηθεί από έναν χρήστη επαναφορά η ανάκτηση του κωδικού πρόσβασης, στην περίπτωση που αυτός το έχει ξεχάσει.

Η Membership μπορεί επίσης να συνεργαστεί με την κλάση Roles που αναλύεται στην παράγραφο 4.4 με σκοπό την παροχή σε μια εφαρμογή διαδικτύου ενός συστήματος εξουσιοδότησης χρηστών βασισμένο σε ρόλους. Αλληλεπιδρά με υλοποιήσεις της κλάσης MemberhipProvider για να μπορέσει να επικοινωνήσει με κάποια πηγή δεδομένων. Το .NET Framework περιλαμβάνει την κλάση SqlMembershipProvider μέσω της οποίας αποθηκεύονται δεδομένα χρηστών σε μια βάση δεδομένων Microsoft SQL Server και την κλάση ActiveDirectoryMembershipProvider για αποθήκευση των δεδομένων σε Active Directory.

Από τις προεπιλεγμένες ρυθμίσεις των ASP.NET εφαρμογών οι κλάση Membership χρησιμοποιείται με provider τον SqlMembershipProvider και καθορίζεται στο αρχείο διαμόρφωσης machine.config με το όνομα AspNetSqlProvider.

Στους παρακάτω πίνακες παρουσιάζονται οι ιδιότητες, οι μέθοδοι και τα γεγονότα που περιλαμβάνει η κλάση Membership. Με έντονο γκρι φαίνονται τα στοιχεία που έχουν χρησιμοποιηθεί κατά την υλοποίηση της εφαρμογής της παρούσας διατριβής.

PROPERTIES	
ApplicationName	Gets or sets the name of the application.
EnablePasswordReset	Gets a value indicating whether the current membership provider is configured to allow users to reset their passwords.
EnablePasswordRetrieval	Gets a value indicating whether the current membership provider is configured to allow users to

	retrieve their passwords.
HashAlgorithmType	The identifier of the algorithm used to hash passwords.
MaxInvalidPasswordAttempts	Gets the number of invalid password or password-answer attempts allowed before the membership user is locked out.
MinRequiredNonAlphanumericCharacters	Gets the minimum number of special characters that must be present in a valid password.
MinRequiredPasswordLength	Gets the minimum length required for a password.
PasswordAttemptWindow	Gets the time window between which consecutive failed attempts to provide a valid password or password answer are tracked.
PasswordStrengthRegularExpression	Gets the regular expression used to evaluate a password.
Provider	Gets a reference to the default membership provider for the application.
Providers	Gets a collection of the membership providers for the ASP.NET application.
RequiresQuestionAndAnswer	Gets a value indicating whether the default membership provider requires the user to answer a password question for password reset and retrieval.
UserIsOnlineTimeWindow	Specifies the number of minutes after the last-activity date/time stamp for a user during which the user is considered online.

Πίνακας 9. Ιδιότητες κλάσης Membership

METHODS	
CreateUser(String, String, String)	Adds a new user with a specified e-mail address to the data store.
CreateUser(String, String)	Adds a new user to the data store.
CreateUser(String, String, String, String, String, Boolean, MembershipCreateStatus)	Adds a new user with specified property values to the data store and returns a status parameter indicating that the user was successfully created or the reason the user creation failed.
CreateUser(String, String, String, String, String, Boolean, Object, MembershipCreateStatus)	Adds a new user with specified property values and a unique identifier to the data store and returns a status parameter indicating that the user was successfully created or the reason the user creation failed.
DeleteUser(String)	Deletes a user and any related user data from the database.
DeleteUser(String, Boolean)	Deletes a user from the database.
FindUsersByEmail(String)	Gets a collection of membership users where the e-mail address contains the specified e-mail address to match.
FindUsersByEmail(String, Int32, Int32, Int32)	Gets a collection of membership users, in a page of data, where the e-mail address contains the specified e-

	mail address to match.
FindUsersByName(String)	Gets a collection of membership users where the user name contains the specified user name to match.
FindUsersByName(String, Int32, Int32, Int32)	Gets a collection of membership users, in a page of data, where the user name contains the specified user name to match.
GeneratePassword	Generates a random password of the specified length.
GetAllUsers()	Gets a collection of all the users in the database.
GetAllUsers(Int32, Int32, Int32)	Gets a collection of all the users in the database in pages of data.
GetNumberOfUsersOnline	Gets the number of users currently accessing an application.
GetUser()	Gets the information from the data source and updates the last-activity date/time stamp for the current logged-on membership user.
GetUser(Boolean)	Gets the information from the data source for the current logged-on membership user. Updates the last-activity date/time stamp for the current logged-on membership user, if specified.
GetUser(Object)	Gets the information from the data source for the membership user associated with the specified unique identifier.
GetUser(String)	Gets the information from the data source for the specified membership user.
GetUser(Object, Boolean)	Gets the information from the data source for the membership user associated with the specified unique identifier. Updates the last-activity date/time stamp for the user, if specified.
GetUser(String, Boolean)	Gets the information from the data source for the specified membership user. Updates the last-activity date/time stamp for the user, if specified.
GetUserNameByEmail	Gets a user name where the e-mail address for the user matches the specified e-mail address.
UpdateUser	Updates the database with the information for the specified user.
ValidateUser	Verifies that the supplied user name and password are valid.

Πίνακας 10. Μέθοδοι κλάσης Membership

EVENTS	
ValidatingPassword	Occurs when a user is created, a password is changed, or a password is reset.

Πίνακας 11. Γεγονότα κλάσης Membership

4.4 Roles Class

Η κλάση Roles βοηθάει στη διαχείριση εξουσιοδότησης των χρηστών μιας εφαρμογής. Με αυτήν καθορίζονται οι πόροι στους οποίους επιτρέπεται να έχει πρόσβαση ο κάθε χρήστης της εφαρμογής. Ο συγκεκριμένος τρόπος διαχείρισης των χρηστών αντιμετωπίζει ομάδες χρηστών ως μια μονάδα αναθέτοντας σε αυτήν την ομάδα κάποιο ρόλο. Με την ανάθεση ρόλων επιτυγχάνεται έλεγχος πρόσβασης στα επιθυμητά κομμάτια της εφαρμογής ανάλογα με το ρόλο του κάθε χρήστη (επιπλέον, έλεγχος πρόσβασης μπορεί να επιτευχθεί και ανάλογα με το όνομα του κάθε χρήστη).

Για την ενεργοποίηση του συγκεκριμένου συστήματος διαχείρισης ρόλων, χρησιμοποιείται το στοιχείο <roleManager> του τμήματος <system.web> που βρίσκεται στο αρχείο web.config της εφαρμογής.

Οι ιδιότητες και οι μέθοδοι της κλάσης Roles φαίνονται στους παρακάτω πίνακες.

PROPERTIES	
ApplicationName	Gets or sets the name of the application to store and retrieve role information for.
CacheRolesInCookie	Gets a value indicating whether the current user's roles are cached in a cookie.
CookieName	Gets the name of the cookie where role names are cached.
CookiePath	Gets the path for the cached role names cookie.
CookieProtectionValue	Gets a value that indicates how role names cached in a cookie are protected.
CookieRequireSSL	Gets a value indicating whether the role names cookie requires SSL in order to be returned to the server.
CookieSlidingExpiration	Indicates whether the role names cookie expiration date and time will be reset periodically.
CookieTimeout	Gets the number of minutes before the roles cookie expires.
CreatePersistentCookie	Gets a value indicating whether the role-names cookie is session-based or persistent.
Domain	Gets the value of the domain of the role-names cookie.
Enabled	Gets or sets a value indicating whether role management is enabled for the current Web application.
MaxCachedResults	Gets the maximum number of role names to be cached for a user.
Provider	Gets the default role provider for the application.
Providers	Gets a collection of the role providers for the ASP.NET application.

Πίνακας 12. Ιδιότητες κλάσης Roles

METHODS	
AddUsersToRole	Adds the specified users to the specified role.
AddUsersToRoles	Adds the specified users to the specified roles.
AddUserToRole	Adds the specified user to the specified role.
AddUserToRoles	Adds the specified user to the specified roles.
CreateRole	Adds a new role to the data source.

DeleteCookie	Deletes the cookie where role names are cached.
DeleteRole(String)	Removes a role from the data source.
DeleteRole(String, Boolean)	Removes a role from the data source.
FindUsersInRole	Gets a list of users in a specified role where the user name contains the specified user name to match.
GetAllRoles	Gets a list of all the roles for the application.
GetRolesForUser()	Gets a list of the roles that the currently logged-on user is in.
GetRolesForUser(String)	Gets a list of the roles that a user is in.
GetUsersInRole	Gets a list of users in the specified role.
IsUserInRole(String)	Gets a value indicating whether the currently logged-on user is in the specified role. The API is only intended to be called within the context of an ASP.NET request thread, and in that sanctioned use case it is thread-safe.
IsUserInRole(String, String)	Gets a value indicating whether the specified user is in the specified role. The API is only intended to be called within the context of an ASP.NET request thread, and in that sanctioned use case it is thread-safe.
RemoveUsersFromRole	Removes the specified users from the specified role.
RemoveUsersFromRoles	Removes the specified user names from the specified roles.
RoleExists	Gets a value indicating whether the specified role name already exists in the role data source.

Πίνακας 13. Μέθοδοι κλάσης Roles

Κανόνες εξουσιοδότησης μπορούν να καθοριστούν είτε στο αρχείο διαμόρφωσης web.config της εφαρμογής είτε προγραμματιστικά στον κώδικα υλοποίησης της εφαρμογής χρησιμοποιώντας ρόλους που έχουν δημιουργηθεί για τους σκοπούς της εφαρμογής. Για παράδειγμα, στο παρακάτω τμήμα <authorization> ενός αρχείου web.config, απαιτείται από τους χρήστες να έχουν συνδεθεί με τα διαπιστευτήριά τους (απαγορεύοντας ανώνυμους χρήστες), ενώ στη συνέχεια επιτρέπεται η πρόσβαση μόνο σε χρήστες που έχουν ρόλο Administrators.

```
<authorization>
  <deny users="?" />
  <allow roles="Administrators" />
  <deny users="*" />
</authorization>
```

Εικόνα 11. Παράδειγμα τμήματος authorization αρχείου web.config

Οι πληροφορίες που αφορούν ρόλους μπορούν να αποθηκευτούν σε βάση δεδομένων Microsoft SQL Server χρησιμοποιώντας την κλάση SqlRoleProvider.

4.5 Σύνοψη Τεχνολογιών Ασφάλειας .NET Framework

Τις τεχνολογίες που αναλύθηκαν στις παραγράφους 4.1 έως 4.4 συμπληρώνουν επιπλέον υπηρεσίες που παρέχει το πλαίσιο .NET. Παρακάτω παρουσιάζεται μια σύνοψη των τεχνολογιών

που παρέχονται από το πλαίσιο σε συνάρτηση με το είδος της αδυναμίας που μπορούν να αντιμετωπίσουν, όπως κατηγοριοποιήθηκαν στην παράγραφο 3.10.

- **Αυθεντικοποίηση**
  - ☑ None
  - ☑ Windows
  - ☑ Forms
  - ☑ OpenID/OAuth

- **Εξουσιοδότηση**
  - ☑ Αρχείων
  - ☑ URL
  - ☑ Βασισμένη σε χρήστες
  - ☑ Βασισμένη σε ρόλους

- **Επικύρωση Δεδομένων**
  - ☑ Client side
  - ☑ Server side
  - ☑ Μηχανισμός Request Validation
  - ☑ Βιβλιοθήκη AntiXSS

- **Διαχείριση Διαμορφώσεων**
  - ☑ Χρήση κατάλληλων μηχανισμών αυθεντικοποίησης – εξουσιοδότησης
  - ☑ Μηχανισμός Προστατευμένης Διαμόρφωσης (Protected Configuration)

- **Ευαίσθητα Δεδομένα**
  - ☑ Μηχανισμοί Κρυπτογράφησης
  - ☑ Χρήση Κρυπτογραφημένου καναλιού SSL
  - ☑ Κλάση RequireHttpsAttribute
  - ☑ ValidateAntiForgeryTokenAttribute – Μέθοδος AntiForgeryToken

- **Διαχείριση Συνόδων**
  - ☑ System.Web.SessionState
  - ☑ InProc
  - ☑ StateServer
  - ☑ SQLServer
  - ☑ Custom
  - ☑ Off

- **Κρυπτογραφία**
  - ☑ Χώροςονομάτων System.Security.Cryptography

- **Χειρισμός Παραμέτρων**
  - ☑ Επιβολή χρήσης μεθόδου HTTP POST
  - ☑ Ρύθμιση httpOnlyCookies

- **Διαχείριση Εξαιρέσεων**
  - ☑ Ρυθμιση Custom errors modes (on – off - RemoteOnly)

- **Παρακολούθηση και Καταγραφή**
  - ☑ Καταγραφέας συμβάντων των Windows
  - ☑ EventLog κλάση
  - ☑ System.Web.Mail και System.Net.Mail
  - ☑ Tracing
  - ☑ ELMAH

Στη συνέχεια του Κεφαλαίου περιγράφονται οι παραπάνω τεχνολογίες.

4.6 Αυθεντικοποίηση με χρήση φόρμας (Forms Authentication)

Για την αυθεντικοποίηση χρηστών μέσω φόρμας χρησιμοποιείται η κλάση FormsAuthentication. Η αυθεντικοποίηση με χρήση φόρμας καθιστά εφικτή την επικύρωση χρηστών και κωδικών πρόσβασης σε εφαρμογές διαδικτύου, οι οποίες δεν απαιτούν χρήση του μηχανισμού αυθεντικοποίησης των Windows. Οι πληροφορίες των χρηστών αποθηκεύονται σε εξωτερική πηγή δεδομένων όπως για παράδειγμα μια βάση δεδομένων με αντικείμενα της κλάσης Membership ή ακόμα και σε κάποιο αρχείο διαμόρφωσης της εφαρμογής. Από τη στιγμή που κάποιος χρήστης έχει αυθεντικοποιηθεί, διατηρείται ένα εισιτήριο αυθεντικοποίησής του ως μέρος ενός cookie ή στο URL ως query string, με σκοπό να μην απαιτείται κάθε φορά που γίνεται ένα αίτημα HTTP να εισάγονται τα στοιχεία αυθεντικοποίησης από τον χρήστη.

Στον παρακάτω πίνακα παρουσιάζονται οι ιδιότητες και οι μέθοδοι της κλάσης FormsAuthentication.

Properties	Description
CookieDomain	Gets the value of the domain of the forms-authentication cookie.
CookieMode	Gets a value that indicates whether the application is configured for cookieless forms authentication.
CookiesSupported	Gets a value that indicates whether the application is configured to support cookieless forms authentication.
DefaultUrl	Gets the URL that the FormsAuthentication class will redirect to if no redirect URL is specified.
EnableCrossAppRedirects	Gets a value indicating whether authenticated users can be redirected to URLs in other Web applications.
FormsCookieName	Gets the name of the cookie used to store the forms-authentication ticket.
FormsCookiePath	Gets the path for the forms-authentication cookie.
IsEnabled	Gets a value that indicates whether forms authentication is enabled.

LoginUrl	Gets the URL for the login page that the FormsAuthentication class will redirect to.
RequireSSL	Gets a value indicating whether the forms-authentication cookie requires SSL in order to be returned to the server.
SlidingExpiration	Gets a value indicating whether sliding expiration is enabled.
TicketCompatibilityMode	Gets a value that indicates whether to use Coordinated Universal Time (UTC) or local time for the ticket expiration date.
Timeout	Gets the amount of time before an authentication ticket expires.

Πίνακας 14. Ιδιότητες κλάσης FormsAuthentication

METHODS	Description
Decrypt	Creates a FormsAuthenticationTicket object based on the encrypted forms-authentication ticket passed to the method.
EnableFormsAuthentication	Enables forms authentication.
Encrypt	Creates a string containing an encrypted forms-authentication ticket suitable for use in an HTTP cookie.
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object.)
GetAuthCookie(String, Boolean)	Creates an authentication cookie for a given user name. This does not set the cookie as part of the outgoing response, so that an application can have more control over how the cookie is issued.
GetAuthCookie(String, Boolean, String)	Creates an authentication cookie for a given user name. This does not set the cookie as part of the outgoing response.
GetHashCode	Serves as the default hash function. (Inherited from Object.)
GetRedirectUrl	Returns the redirect URL for the original request that caused the redirect to the login page.
GetType	Gets the Type of the current instance. (Inherited from Object.)
HashPasswordForStoringInConfigFile	**Obsolete.** Produces a hash password suitable for storing in a configuration file based on the specified password and hash algorithm.
Initialize	Initializes the FormsAuthentication object based on the configuration settings for the application.
RedirectFromLoginPage(String, Boolean)	Redirects an authenticated user back to the originally requested URL or the default URL.
RedirectFromLoginPage(String, Boolean, String)	Redirects an authenticated user back to the originally requested URL or the default URL using the specified cookie path for the forms-authentication cookie.
RedirectToLoginPage()	Redirects the browser to the login URL.
RedirectToLoginPage(String)	Redirects the browser to the login URL with the specified

	query string.
RenewTicketIfOld	Conditionally updates the issue date and time and expiration date and time for a FormsAuthenticationTicket.
SetAuthCookie(String, Boolean)	Creates an authentication ticket for the supplied user name and adds it to the cookies collection of the response, or to the URL if you are using cookieless authentication.
SetAuthCookie(String, Boolean, String)	Creates an authentication ticket for the supplied user name and adds it to the cookies collection of the response, using the supplied cookie path, or using the URL if you are using cookieless authentication.
SignOut	Removes the forms-authentication ticket from the browser.

Πίνακας 15. Μέθοδοι κλάσης FormsAuthentication

Για την ενεργοποίηση του μηχανισμού Forms Authentication αρκεί η ρύθμιση του χαρακτηριστικού mode=”Forms” του στοιχείου authentication στο αρχείο ρυθμίσεων της εφαρμογής web.config.Η συγκεκριμένη ρύθμιση δεν θα λειτουργήσει εφόσον τοποθετηθεί σε αρχείο διαμόρφωσης που βρίσκεται σε κάποιο υποκατάλογο της εφαρμογής άρα πρέπει να τοποθετηθεί στο αρχείο διαμόρφωσης που βρίσκεται στον αρχικό κατάλογο της εφαρμογής.

Ένα παράδειγμα ρύθμισης που απαιτεί κάθε αίτημα για σύνδεση με την εφαρμογή να συνοδεύεται και με το εισιτήριο αυθεντικοποίησης (είτε μέσω cookie είτε μέσω query string) φαίνεται παρακάτω:

```
<authentication mode="Forms">
      <forms loginUrl="~/SGAccount/Login" timeout="30"/>
    </authentication>
```

Σημειώνεται ότι στην περίπτωση που κάποιος αυθεντικοποιημένος χρήστης κλείσει τον περιηγητή του ή λήξει ο χρόνος συνεδρίας του cookie σύμφωνα με τον χρόνο σε λεπτά που έχει καθοριστεί στο χαρακτηριστικό timeout, το εισιτήριο αυτό ακυρώνεται και ο χρήστης θα πρέπει να εισάγει ξανά τα διαπιστευτήριά του για να του εκδοθεί νέο εισιτήριο.

Ο μηχανισμός Forms Authentication μπορεί να εφαρμοστεί σε οποιοδήποτε είδος εφαρμογής ASP.NET άρα και σε εφαρμογές που υλοποιούνται με την αρχιτεκτονική MVC.

4.6.1 Ροή γεγονότων μηχανισμού Forms Authentication σε εφαρμογή MVC

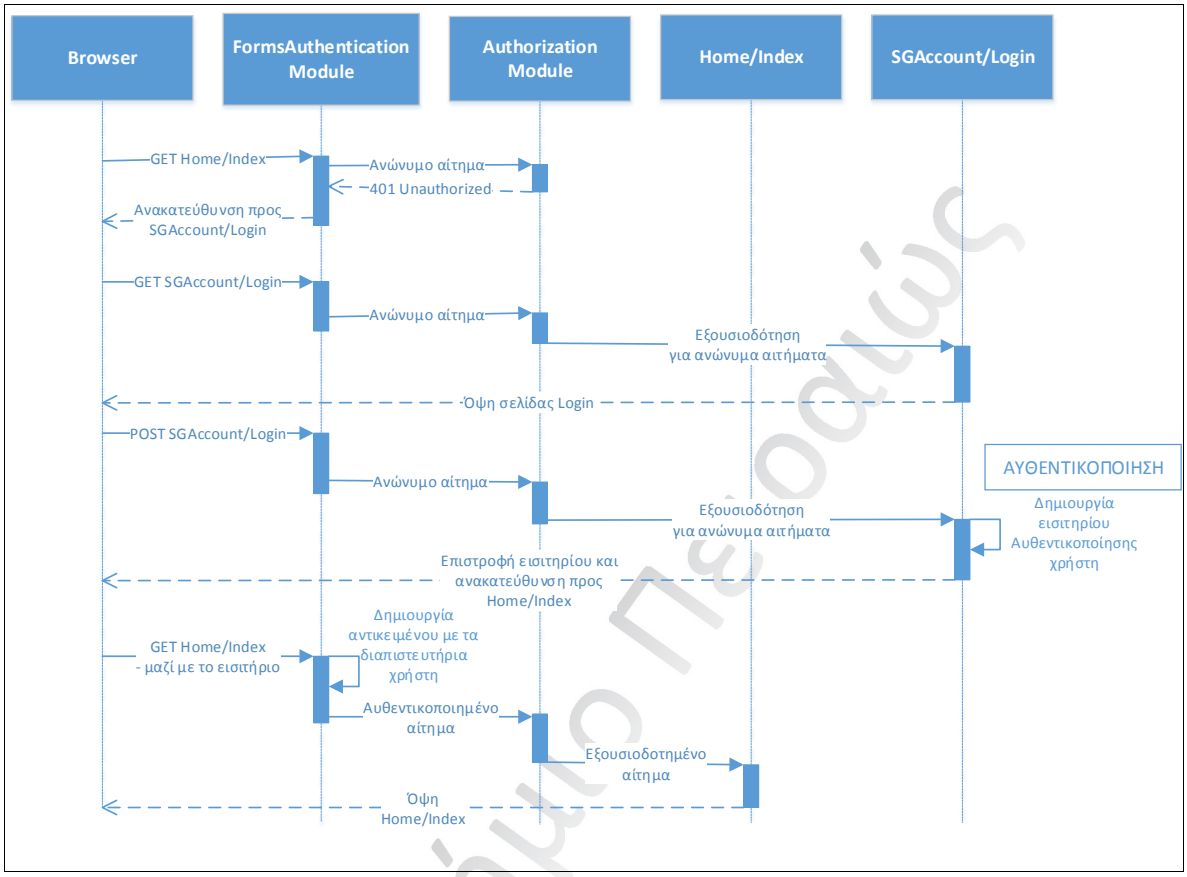
Εφόσον έχει γίνει ρύθμιση σύμφωνα με την προηγούμενη παράγραφο για χρήση του μηχανισμού αυθεντικοποίησης μέσω φόρμας για μια εφαρμογή υλοποιημένη με το ASP.NET MVC Framework, τα γεγονότα που συμβαίνουν κατά χρονολογική σειρά είναι τα παρακάτω:

1. Ένας χρήστης της εφαρμογής ζητάει μια σελίδα της, έστω την Home/Index.
2. Το αίτημα περνάει αρχικά από τον IIS Server ο οποίος προσπαθεί να αυθεντικοποιήσει τον χρήστη (επίπεδο server). Από τις προκαθορισμένες ρυθμίσεις του, εφόσον είναι ανώνυμος, θα δημιουργήσει ένα τεκμήριο (token) που θα τον αντιπροσωπεύει και θα το περάσει στο ASP.NET (επίπεδο εφαρμογής).
3. Το ASP.NET με τη σειρά του θα προσπαθήσει να τον αυθεντικοποιήσει σύμφωνα με τις ρυθμίσεις του αρχείου web.config της εφαρμογής. Για το γεγονός αυτό, το υπεύθυνο τμήμα κώδικα ονομάζεται FormsAuthenticationModule.

4. Το FormsAuthenticationModule δημιουργεί ένα αντικείμενο που αντιπροσωπεύει το συγκεκριμένο χρήστη και το μεταβιβάζει στο Authorization Module, το τμήμα του κώδικα που είναι υπεύθυνο για τη διαχείριση της εξουσιοδότησης χρήσης των πόρων της εφαρμογής. Στην περίπτωση που στη συγκεκριμένη ιστοσελίδα Home/Index δεν μπορούν να έχουν πρόσβαση οι ανώνυμοι χρήστες, τότε το Authorization Module επιστρέφει σφάλμα με κωδικό 401 στο Forms Authentication Module και δεν επιτρέπει την πρόσβαση του ανώνυμου χρήστη στην ιστοσελίδα αυτή.
5. Το Forms Authentication Module βλέποντας το συγκεκριμένο κωδικό σφάλματος, απαντάει στον χρήστη στέλνοντάς του την ιστοσελίδα για εισαγωγή διαπιστευτηρίων η οποία καθορίζεται από το χαρακτηριστικό loginUrl, στην προκειμένη περίπτωση SGAccount/Login.
6. Ο χρήστης παρέχει τα απαιτούμενα στοιχεία της φόρμας (όνομα χρήστη και κωδικό πρόσβασης) και υποβάλλει τη φόρμα.
7. Ο Controller της φόρμας όταν τα λάβει, τα επικυρώνει σύμφωνα με την βάση δεδομένων των χρηστών της εφαρμογής, δημιουργεί το εισιτήριο πιστοποίησης του χρήστη και το αποστέλλει στον browser του χρήστη.
8. Σε κάθε μετέπειτα αίτημα του χρήστη, το εισιτήριο αυτό μεταδίδεται ταυτόχρονα και έτσι αναγνωρίζεται κάθε φορά ο αυθεντικοποιημένος πλέον χρήστης.

Οι αλληλεπιδράσεις αυτές θα πρέπει να εκτελούνται χρησιμοποιώντας το πρωτόκολλο SSL για την κρυπτογράφηση και περαιτέρω προστασία των μηνυμάτων που περιλαμβάνουν τα διαπιστευτήρια του χρήστη καθώς επίσης και το εισιτήριο αυθεντικοποίησης. Αν δεν χρησιμοποιηθεί το συγκεκριμένο πρωτόκολλο κρυπτογράφησης, τα μηνύματα θα αποστέλλονται υπό μορφή απλού κειμένου και εύκολα κάποιος επιτιθέμενος θα μπορεί να τα υποκλέψει, να τα αλλοιώσει ή να τα χρησιμοποιήσει προς όφελός του.

Στο παρακάτω διάγραμμα ροής φαίνονται οι αλληλεπιδράσεις που αναλύθηκαν παραπάνω για την αυθεντικοποίηση ενός χρήστη με το μηχανισμό Forms Authentication, για μια εφαρμογή υλοποιημένη με το ASP.NET MVC Framework [23].



Εικόνα 12. Διάγραμμα ροής μηχανισμού Forms Authentication

Ο συγκεκριμένος μηχανισμός αυθεντικοποίησης χρησιμοποιείται πιο συχνά στις εφαρμογές διαδικτύου. Τα πρότυπα εφαρμογών που παρέχονται από το Visual Studio χρησιμοποιούν τις απαιτούμενες ρυθμίσεις για χρήση αυτού του μηχανισμού ενώ παράλληλα παρέχουν και έτοιμες υλοποιήσεις φορμών για σύνδεση (Login) και εγγραφή (Registration) χρηστών. Στις υλοποιήσεις εγγραφής χρηστών περιλαμβάνεται λειτουργία επικύρωσης κωδικών πρόσβασης χρήστη και έλεγχος για ελάχιστο μήκος έξι χαρακτήρων κωδικού. Οι συγκεκριμένοι Controllers είναι ρυθμισμένοι έτσι ώστε να χρησιμοποιούν βάση δεδομένων χρηστών SQL Express (χρησιμοποιώντας Membership και Roles providers) η οποία έχει το προκαθορισμένο όνομα ASPNETDB.MDF και αποθηκεύεται στον φάκελο της εφαρμογής /App_Data.

4.7 Εξουσιοδότηση σε εφαρμογές ASP.NET

Αρκετά συχνά απαιτείται σύνδεση ενός αυθεντικοποιημένου χρήστη με την εφαρμογή για να του επιτραπεί η πρόσβαση σε πόρους της. Σε ορισμένες περιπτώσεις απαιτείται επιπλέον περιορισμός πρόσβασης σε περιεχόμενο της εφαρμογής των αυθεντικοποιημένων χρηστών ή χρηστών που ανήκουν σε κάποιο ρόλο. Ανάλογα με τον τρόπο υλοποίησης της εφαρμογή, είτε απλή εφαρμογή ASP.NET είτε βασισμένη στην αρχιτεκτονική MVC, μπορεί να χρησιμοποιηθεί και ο κατάλληλος μηχανισμός που αναλύεται στις επόμενες παραγράφους.

4.7.1 Απλές εφαρμογές ASP.NET

Ο έλεγχος εξουσιοδότησης χρηστών σε απλές εφαρμογές ASP.NET γίνεται δηλώνοντας ρητώς ότι επιτρέπεται ή απαγορεύεται η πρόσβαση σε κάποιο κατάλογο του συστήματος αρχείου της εφαρμογής ανάλογα με το όνομα του χρήστη ή κάποιο συγκεκριμένο ρόλο. Αυτό επιτυγχάνεται δημιουργώντας ένα τμήμα <authorization> στο αρχείο διαμόρφωσης web.config του φακέλου για τον οποίο θέλουμε να επιτρέψουμε ή να απαγορέψουμε την πρόσβαση. Ένα τέτοιο παράδειγμα φαίνεται στην Εικόνα 11. Εντός του τμήματος <authorization> καθορίζονται ονόματα χρηστών και ρόλοι στα στοιχεία allow ή deny, για χορήγηση ή απαγόρευση πρόσβασης αντίστοιχα. Οι εξουσιοδοτήσεις που περιγράφονται σε αρχείο ρυθμίσεων ενός καταλόγου αφορούν και τους υποκαταλόγους του. Για να αναιρεθεί κάποια δήλωση δηλαδή, θα πρέπει να καθοριστεί νέος κανόνας εντός του επιθυμητού υποκαταλόγου.

Εντός των στοιχείων allow και deny χρησιμοποιούνται τα χαρακτηριστικά που περιγράφονται στον παρακάτω πίνακα:

Χαρακτηριστικό	Περιγραφή
Users	Καθορίζει τους λογαριασμούς χρηστών για τους οποίους ισχύει ο κανόνας Οι ανώνυμοι χρήστες δηλώνονται χρησιμοποιώντας ερωτηματικό (?). Μπορούν να καθοριστούν όλοι οι αυθεντικοποιημένοι χρήστες τοποθετώντας αστερίσκο (*).
Roles	Καθορίζει το όνομα του ρόλου για τον οποίο ισχύει ο κανόνας εξουσιοδότησης
Verbs	Καθορίζει το είδος των αιτημάτων HTTP (GET, HEAD, POST) για τα οποία θα ισχύει ο κανόνας. Εφόσον δεν δηλωθεί το συγκεκριμένο χαρακτηριστικό, τότε εννοείται ότι ο κανόνας ισχύει για όλα τα είδη.

Πίνακας 16. Χαρακτηριστικά πεδία στοιχείων allow και deny για έλεγχο εξουσιοδότησης

4.7.2 Εφαρμογές ASP.NET MVC Framework

Στις περιπτώσεις εφαρμογών διαδικτύου που έχουν υλοποιηθεί χρησιμοποιώντας ASP.NET MVC Framework, για να απαγορευθεί η πρόσβαση σε κάποιο πόρο ή ιστοσελίδα της εφαρμογής, πρέπει να περιοριστεί η πρόσβαση στη μέθοδο κάποιας κλάσης Controller που είναι υπεύθυνη για την δημιουργία της όψης. Αυτό συμβαίνει χρησιμοποιώντας την κλάση AuthorizeAttribute που παρέχει το Framework.

Όταν μαρκάρεται μια μέθοδος με το χαρακτηριστικό Authorize, η πρόσβαση σε αυτή περιορίζεται μόνο στους χρήστες που είναι αυθεντικοποιημένοι[24]. Αν μια κλάση Controller είναι μαρκαρισμένη με το Authorize, όλες οι μέθοδοι που υπάρχουν στην κλάση περιορίζονται επίσης μόνο στους αυθεντικοποιημένους χρήστες. Σε αυτήν την περίπτωση μπορεί να χρησιμοποιηθεί η κλάση AllowAnonumousAttribute και να μαρκαριστεί κάποια μέθοδος του συγκεκριμένου Controller για την οποία είναι επιτρέπεται η πρόσβασης και σε χρήστες που δεν είναι αυθεντικοποιημένοι.

Στους παρακάτω πίνακες παρουσιάζονται οι ιδιότητες και κάποιες μέθοδοι της κλάσης AuthorizeAttribute.

Ιδιότητα	Περιγραφή
AllowMultiple	Gets or sets a value that indicates whether more than one instance of the filter attribute can be specified
Order	Gets or sets the order in which the action filters are executed.
Roles	Gets or sets the user roles that are authorized to access the controller or action method.

TypeId	Gets the unique identifier for this attribute
Users	Gets or sets the users that are authorized to access the controller or action method.

Πίνακας 17. Ιδιότητες κλάσης AuthorizeAttribute

Name	Description
AuthorizeCore	When overridden, provides an entry point for custom authorization checks.
HandleUnauthorizedRequest	Processes HTTP requests that fail authorization.
OnAuthorization	Called when a process requests authorization.

Πίνακας 18. Μέθοδοι της κλάσης AuthorizeAttribute

Το παρακάτω παράδειγμα κώδικα C# δείχνει έναν απλό Controller (AccountController) που υλοποιεί λειτουργίες Login,Logoff και Registration. Εδώ εφαρμόζονται τα χαρακτηριστικά Authorize στην κλάση για να απαγορευθεί η πρόσβαση στις μεθόδους της σε όλους τους μη αυθεντικοποιημένους χρήστες, αλλά παράλληλα να επιτρέπεται η πρόσβαση στη μέθοδο Register έτσι ώστε να μπορεί να εγγραφεί οποιοσδήποτε χρήστης.

```
[Authorize]
public class AccountController : Controller
{
    public AccountController () { . . . }

    [AllowAnonymous]
    public ActionResult Register() { . . . }

    public ActionResult Manage() { . . . }

    public ActionResult LogOff() { . . . }
. . .
}
```

Εικόνα 13. Παράδειγμα εξουσιοδότησης με χρήση της κλάσης AuthorizeAttribute και AllowAnonymousAttribute

Το χαρακτηριστικό Authorize επιτρέπει να καθοριστούν κανόνες εξουσιοδότησης σε προκαθορισμένους ρόλους χρηστών ή ακόμα και σε μεμονωμένους χρήστες, χρησιμοποιώντας τις ιδιότητες Roles και Users. Με αυτό τον τρόπο αυξάνεται ο βαθμός ελέγχου εξουσιοδότησης για πρόσβαση στις ιστοσελίδες ή στους πόρους της εφαρμογής.

Στα παρακάτω παραδείγματα κώδικα C# φαίνεται ο τρόπος υλοποίησης μιας κλάσης Controller η οποία είναι διαθέσιμη μόνο σε χρήστες που είναι στο ρόλο του διαχειριστή ή μόνο σε συγκεκριμένους χρήστες.

```
[Authorize(Roles="Administrators")]
public class ExampleController : Controller
{
    . . .
}
```

Εικόνα 14. Παράδειγμα εξουσιοδότησης μόνο σε χρήστες ρόλου Administrator

```
[Authorize(Users="Alice,Bob")]
public class ExampleController : Controller
{
    . . .
}
```

Εικόνα 15. Παράδειγμα εξουσιοδότησης μόνο σε συγκεκριμένους χρήστες

Όταν ένας μη αυθεντικοποιημένος χρήστης προσπαθήσει να προσπελάσει μια μέθοδο Controller που έχει χαρακτηριστεί με το [Authorize], το ASP.NET MVC Framework θα επιστρέψει σφάλμα HTTP με κώδικα 401. Εφόσον η εφαρμογή έχει ρυθμιστεί να χρησιμοποιεί το μηχανισμό αυθεντικοποίησης Forms Authentication, η ιστοσελίδα που θα εμφανιστεί στο χρήστη θα είναι η Login.

Οι κλάσεις AuthorizeAttribute και AllowAnonymousAttribute είναι φίλτρα, που σημαίνει ότι εκτελούνται πριν εκτελεστεί ο κώδικας που ακολουθεί μετά από τη δήλωσή τους. Μια καλή προσέγγιση ασφάλειας είναι να τοποθετούνται οι έλεγχοι ασφάλειας όσο πιο κοντά γίνεται στον πόρο που πρόκειται να ασφαλιστεί. Παράλληλα μπορούν να υπάρχουν έλεγχοι ασφάλειας σε υψηλότερο επίπεδο της διαδικασίας πρόσβασης αλλά τελικά το επιθυμητό είναι να ασφαλιστεί ο συγκεκριμένος πόρος, στην περίπτωση εφαρμογών MVC ο Controller και οι μέθοδοι που περιέχει. Αυτός είναι και ο σκοπός των κλάσεων - φίλτρων.

4.8 Επικύρωση Δεδομένων Εισόδου

Η επικύρωση δεδομένων εισόδου είναι μια διαδικασία η οποία πρέπει να εκτελείται στην πλευρά του client αλλά και στην πλευρά touserver. Στην πλευρά του client δίνει άμεσα στον χρήστη αναπληροφόρηση που αφορά τα δεδομένα που έχει εισάγει σε φόρμες, αναμενόμενη λειτουργία στις σύγχρονες διαδικτυακές εφαρμογές. Η επικύρωση στην πλευρά του server θα πρέπει να εκτελείται γιατί ποτέ δεν πρέπει να θεωρούνται εκ των προτέρων αξιόπιστα τα δεδομένα που προέρχονται από ένα δίκτυο.

Το ASP.NET παρέχει την λειτουργία Request Validation σύμφωνα με την οποία ένα αίτημα HTTP εξετάζεται για το αν περιέχει πιθανό επικίνδυνο περιεχόμενο, δηλαδή κώδικα HTML ή JavaScript στο σώμα, στην επικεφαλίδα, στο query string ή στα cookies τους. Το ASP.NET εκτελεί αυτόν τον έλεγχο γιατί τέτοιος κώδικας στα URL query strings και στα cookies, ή οι υποβαλλόμενες τιμές μιας φόρμας μπορεί να έχουν προστεθεί κακόβουλα [25]. Η λειτουργία παρέχεται από τις προκαθορισμένες ρυθμίσεις του πλαισίου.

Η επικύρωση δεδομένων στις εφαρμογές που χρησιμοποιούν το ASP.NET MVC αφορούν την επικύρωση των τιμών που περιγράφονται από τις κλάσεις που αποτελούν το Μοντέλο. Γι 'αυτό το σκοπό χρησιμοποιούνται κλάσεις χαρακτηριστικών Data Annotations που βρίσκονται στο χώρο ονομάτων System.ComponentModel.DataAnnotations. Οι πιο συνηθισμένες είναι :

Κλάση	Περιγραφή
RequiredAttribute	Specifies that a data field value is required.
StringLengthAttribute	Specifies the minimum and maximum length of characters that are allowed in a data field.
RangeAttribute	Specifies the numeric range constraints for the value of a data field.
RegularExpressionAttribute	Specifies that a data field value in ASP.NET Dynamic Data must match the specified regular expression.

DataTypeAttribute	Specifies the name of an additional type to associate with a data field.

Πίνακας 19. Κλάσεις χαρακτηριστικών DataAnnotations του ASP.NET

Επιπλέον, το ASP.NET MVC Framework προσθέτει δύο ακόμα χαρακτηριστικά που μπορούν να χρησιμοποιηθούν για επικύρωση δεδομένων. Ανήκουν δηλαδή στον χώρο ονομάτων System.Web.Mvc.

Κλάσεις	Περιγραφή
CompareAttribute	Provides an attribute that compares two properties of a model.
RemoteAttribute	Provides an attribute that uses the jQuery validation plug-in remote validator.

Πίνακας 20.Κλάσεις χαρακτηριστικών του ASP.NET MVC

Τα χαρακτηριστικά αυτά εφαρμόζονται στις ιδιότητες των κλάσεων του μοντέλου και παρέχουν επικύρωση τόσο στην πλευρά του server όσο και στου client. Από τις προκαθορισμένες ρυθμίσεις του ASP.NET στο τμήμα <appSettings> του αρχείου web.config της εφαρμογής έχουν προστεθεί οι παρακάτω δηλώσεις που επιτρέπουν την επικύρωση στην πλευρά του client :

```
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="true" />
```

Οι συγκεκριμένες ρυθμίσεις αποτελούν ιδιότητες της κλάσης HtmlHelper του χώρου ονομάτων System.Web.Mvc.

Ένα επιπλέον επίπεδο ασφάλειας στην επικύρωση δεδομένων προσφέρει η βιβλιοθήκη AntiXSS [26], η οποία παρέχει μεθόδους κωδικοποίησης δεδομένων εισόδου όπως HTML, XML, CSS και JavaScript. Η βιβλιοθήκη, εφόσον εγκατασταθεί, παρακάμπτει την κλάση HttpEncoder που χρησιμοποιείται προκαθορισμένα από το ASP.NET. Για να γίνει αυτό αρκεί να τοποθετηθεί το χαρακτηριστικό encoderType=”System.Web.Security.AntiXss.AntiXssEncoder” στο στοιχείο <httpRuntime> του αρχείου web.config της εφαρμογής.

4.9 Διαχείριση Διαμορφώσεων

Για να μην μπορούν να παρακαμφθούν ρυθμίσεις που βρίσκονται στο αρχείο machine.config από οποιαδήποτε εφαρμογή χρησιμοποιείται το στοιχείο <location>. Εντός του στοιχείου τοποθετείται το τμήμα της διαμόρφωσης που είναι επιθυμητό να μην παρακαμφθεί και τίθεται το χαρακτηριστικό allowOverride=”false” του <location> όπως φαίνεται στο παρακάτω παράδειγμα:

```
<location path="" allowOverride="false">
      <system.web>
        <!--....ρυθμίσεις.....-->
      </system.web>
    </location>
```

Σύμφωνα με την παραπάνω δήλωση σε ένα αρχείο machine.config, οι ρυθμίσεις που αναγράφονται εντός του τμήματος <system.web> δεν μπορούν να παρακαμφθούν από ρυθμίσεις στο επίπεδο εφαρμογής. Αφήνοντας το χαρακτηριστικό path κενό, υποδεικνύεται ότι οι ρυθμίσεις αφορούν τη συγκεκριμένη μηχανή, ενώ το χαρακτηριστικό allowOverride=”false” εξασφαλίζει ότι οι ρυθμίσεις στο web.config οποιασδήποτε εφαρμογής δεν μπορούν να παρακάμψουν τις συγκεκριμένες ρυθμίσεις. Οποιαδήποτε προσπάθεια για παράκαμψή τους θα προκαλέσει δημιουργία εξαίρεσης, ακόμα κι αν οι ρυθμίσεις των αρχείων συμπίπτουν.

Όταν αποθηκεύονται ευαίσθητες πληροφορίες μιας εφαρμογής σε ένα αρχείο ρυθμίσεων τότε αυτές θα πρέπει να κρυπτογραφούνται. Αυτό επιτυγχάνεται με τη λειτουργία Προστατευμένης Διαμόρφωσης (Protected Configuration) [27] που παρέχει το .NET Framework. Παράδειγμα πληροφοριών που μπορεί να αποθηκεύονται σε αρχεία διαμόρφωσης είναι κλειδιά κρυπτογράφησης και συμβολοσειρές σύνδεσης (connection strings) με βάση δεδομένων. Η διαχείριση της λειτουργίας αυτής γίνεται με το εργαλείο ASP.NET IIS Registration (Aspnet_regiis.exe) ή μέσω των κλάσεων του System.Configuration χώρου ονομάτων [28].

Η κρυπτογράφηση και αποκρυπτογράφηση περιεχομένου αρχείων ρυθμίσεων web.config εκτελείται χρησιμοποιώντας την κλάση ProtectedConfigationProvider του System.Configuration. Οι δύο κλάσεις που υλοποιούνται για το σκοπό αυτό στο .NET Framework είναι οι:

- DpapiProtectedConfigurationProvider : Χρήση του Windows Data Protection API (DPAPI)
- RsaProtectedConfigurationProvider : Χρήση του αλγόριθμου RSA

Το AP.NET αποκρυπτογραφεί το περιεχόμενο του αρχείου web.config όταν το επεξεργάζεται. Συνεπώς, για να χρησιμοποιηθούν οι κρυπτογραφημένες ρυθμίσεις ή τιμές από το πλαίσιο δεν απαιτούνται επιπρόσθετα βήματα για την αποκρυπτογράφησή τους.

4.10 Ευαίσθητα Δεδομένα

Τα ευαίσθητα δεδομένα μιας εφαρμογής περιλαμβάνουν τις λεπτομέρειες διαμόρφωσης της εφαρμογής και τα ιδιαίτερα δεδομένα της εφαρμογής. Για την προστασία τους όταν αυτά είναι αποθηκευμένα, μπορούν να χρησιμοποιηθούν διαδικασίες κρυπτογράφησής τους όπως αναλύεται στις παραγράφους 4.9 και 4.12.

Για τη διαφύλαξή των ευαίσθητων δεδομένων κατά τη μετάδοσή τους από τον server στον client, πρέπει αυτά να προστατεύονται με χρήση πρωτοκόλλου SSL. Το ASP.NET MVC διαθέτει την κλάση RequireHttpsAttribute ως ένα επιπλέον μέτρο για να εξασφαλίσει ότι θα χρησιμοποιηθεί σύνδεση SSL για τη μετάδοση τους.

Ένας ακόμα μηχανισμός που παρέχει το ASP.NET MVC για την προστασία δεδομένων που μεταδίδονται μέσω φόρμας από απειλές πλαστογράφησης αίτησης δεδομένων μεταξύ ιστοτόπων (cross-site request forgery - CSFR) είναι η μέθοδος AntiForgeryToken(). Για τη χρήση αυτής της υπηρεσίας αρκεί να γίνει κλήση της μεθόδου εντός της επιθυμητής φόρμας και παράλληλα να προστεθεί το χαρακτηριστικό ValidateAntiForgeryTokenAttribute στην αντίστοιχη μέθοδο Controller.

4.11 Διαχείριση Συνόδου

Η διαχείριση κατάστασης συνόδου που παρέχει το ASP.NET επιτρέπει την αποθήκευση οποιουδήποτε είδους πληροφορίας στην μνήμη του server. Η συγκεκριμένες πληροφορίες προστατεύονται αφού δεν μεταδίδονται στον client και είναι μοναδικά συνδεδεμένες με κάποια σύνοδο. Κάθε client που αποκτά πρόσβαση σε μια εφαρμογή ASP.NET αποκτά και διαφορετική σύνοδο (SessionID) και διατηρούνται για αυτόν ξεχωριστή συλλογή πληροφοριών [29].

Ο χώρος ονομάτων System.Web.SessionState [30] παρέχει τις διεπαφές και τις κλάσεις που επιτρέπουν την αποθήκευση δεδομένων στον server, που αφορούν έναν client μιας εφαρμογής Ιστού. Αυτά τα δεδομένα προσδίδουν στον client την εμφάνιση μιας μόνιμης σύνδεσης με την εφαρμογή.

Για τη διατήρηση της κάθε συνόδου του client με την εφαρμογή, πρέπει αυτός να παρουσιάζει σε κάθε αίτημά του προς τον server και τον κατάλληλο κωδικό συνόδου. Αυτό μπορεί να το κάνει με δύο τρόπους:

1. Χρησιμοποιώντας cookies. Σε αυτήν την περίπτωση, η οποία είναι προκαθορισμένα ενεργοποιημένη, το SessionID μεταδίδεται μέσω ενός cookie με την, συνήθως

χρησιμοποιούμενη, ονομασία ASP.NET SessionId. Ο κωδικός αυτός δημιουργείται αυτόματα από το ASP.NET.

2. Χρησιμοποιώντας τροποποιημένο URL. Σε αυτήν την περίπτωση, το SessionID μεταδίδεται μέσω ειδικά τροποποιημένου URL. Η λειτουργία αυτή εφαρμόζεται όταν κάποιος client δεν υποστηρίζει cookies.

Η ρυθμίσεις για τη διαχείριση συνόδου γίνεται μέσω του αρχείου web.config της εφαρμογής και συγκεκριμένα του στοιχείου <sessionState> που βρίσκεται εντός του στοιχείου <system.web>. Το ASP.NET παρέχει τους παρακάτω τρόπους λειτουργίας για τη διαχείριση της κατάστασης συνόδου μιας εφαρμογής.

- InProc: Ο προκαθορισμένος τρόπος σύμφωνα με τον οποίο η κατάσταση συνόδου αποθηκεύεται στην μνήμη του server.
- StateServer: Με αυτόν τον τρόπο η κατάσταση συνόδου αποθηκεύεται σε ξεχωριστή διεργασία που ονομάζεται ASP.NET state service και διατηρείται σε περίπτωση επανεκκίνησης της εφαρμογής.
- SQLServer: Με αυτόν τον τρόπο η κατάσταση συνόδου αποθηκεύεται σε βάση δεδομένων του SQL Server και διατηρείται σε περίπτωση επανεκκίνησης της εφαρμογής και διατηρείται σε περίπτωση επανεκκίνησης της εφαρμογής.
- Custom: Με αυτόν τον τρόπο καθορίζεται κάποιος διαφορετικός από τους προηγούμενους πάροχος αποθήκευσης της κατάστασης συνόδου.
- Off: Απενεργοποιείται η διατήρηση κατάστασης συνόδου.

Οι παραπάνω τρόποι λειτουργίας επιλέγονται από το χαρακτηριστικό mode της ετικέτας sessionState.

Σημαντικό χαρακτηριστικό ρύθμισης της διαχείρισης συνόδου που παρέχει το ASP.NET είναι το timeout. Με αυτό καθορίζεται ο αριθμός των λεπτών που θα περιμένει το ASP.NET για να απορρίψει μια σύνοδο αν δεν λάβει κάποιο αίτημα από τον client της συγκεκριμένης συνόδου.

4.12 Κρυπτογραφία

Ο χώρος ονομάτων System.Security.Cryptography παρέχει υπηρεσίες κρυπτογράφησης όπως ασφαλή κωδικοποίηση/αποκωδικοποίηση δεδομένων, κατακερματισμό, γεννήτρια τυχαίων αριθμών και αυθεντικοποίηση μηνυμάτων [31].

4.13 Τροποποίηση Παραμέτρων

Για να αποφευχθεί οποιαδήποτε τροποποίηση των παραμέτρων της εφαρμογής που μπορεί να μεταδίδονται στο Διαδίκτυο μέσω query strings, cookies και πεδία φορμών αυτά πρέπει να εξασφαλιστεί ότι μεταδίδονται χρησιμοποιώντας την μέθοδο HTTP POST. Στο ASP.NET MVC αυτό επιτυγχάνεται χαρακτηρίζοντας τις επιθυμητές μεθόδους Controller με το χαρακτηριστικό HttpPostAttribute.

Για την προστασία των cookies από τροποποίηση το ASP.NET παρέχει ρύθμιση στο αρχείο web.config της εφαρμογής με το στοιχείο <httpCookies>. Όταν στο χαρακτηριστικό httpOnlyCookies του συγκεκριμένου στοιχείου δοθεί η τιμή true, τότε τα cookies που χρησιμοποιεί η εφαρμογή μπορούν να χρησιμοποιηθούν μόνο από σενάρια που εκτελούνται από τον server. Σενάρια που εκτελούνται στον client δεν μπορούν να τα τροποποιήσουν ούτε να τα διαβάσουν.

4.14 Διαχείριση Εξαιρέσεων

Κατά την υλοποίηση μιας διαδικτυακής εφαρμογής δεν θα πρέπει να γνωστοποιούνται λεπτομέρειες πιθανόν σφαλμάτων της (εξαιρέσεων) στους clients. Για το λόγο αυτό, το ASP.NET παρέχει ρύθμιση με το στοιχείο <customErrors> στο αρχείο web.config. Με το στοιχείο αυτό καθορίζεται ποια μηνύματα σφαλμάτων και με ποιο τρόπο θα επιστρέφονται στους χρήστες της εφαρμογής, όταν συμβαίνει κάποια πρόβλημα στην εφαρμογή που δημιουργεί εξαίρεση.

Το στοιχείο τοποθετείται εντός του στοιχείου <system.web> και περιλαμβάνει τα χαρακτηριστικά mode και defaultRedirect.

Το χαρακτηριστικό mode καθορίζει αν η λειτουργία είναι ενεργοποιημένη και μπορεί να πάρει μία από τις παρακάτω επιλογές:

- On: Η λειτουργία είναι ενεργοποιημένη, που σημαίνει ότι σε περίπτωση σφάλματος δεν θα εμφανιστούν λεπτομέρειες αλλά κάποιο τυποποιημένο μήνυμα.
- Off: Η λειτουργία είναι απενεργοποιημένη και σε περίπτωση σφάλματος θα εμφανιστούν στον client όλες οι πληροφορίες γι' αυτό.
- RemoteOnly: Είναι η προκαθορισμένη ρύθμιση και καθορίζει ότι η λειτουργία ισχύει για απομακρυσμένους clients. Εφόσον ο client που χρησιμοποιεί την εφαρμογή δεν βρίσκεται στον τοπικό υπολογιστή δεν θα εμφανιστούν λεπτομέρειες του σφάλματος. Στους clients που βρίσκονται στον τοπικό υπολογιστή εμφανίζονται όλες οι λεπτομέρειες.

Το χαρακτηριστικό defaultRedirect καθορίζει την προκαθορισμένη ιστοσελίδα που θα εμφανιστεί στους clients για τους οποίους είναι ενεργοποιημένη η λειτουργία, σε περίπτωση οποιουδήποτε σφάλματος της εφαρμογής. Αν δεν έχει καθοριστεί τέτοια ιστοσελίδα παρουσιάζεται ένα τυποποιημένο μήνυμα από το ASP.NET.

Επιπλέον, εντός του στοιχείου <customErrors> μπορεί να τοποθετηθεί το στοιχείο <error> το οποίο χρησιμοποιείται για να καθορίσει διαφορετικές ιστοσελίδες παρουσίασης ανάλογα με το σφάλμα. Το στοιχείο μπορεί να χρησιμοποιηθεί μία ή περισσότερες φορές. Ένα παράδειγμα διαμόρφωσης φαίνεται παρακάτω.

```
<configuration>
        . . .
        <system.web>
                . . .
                <customErrors mode="On" defaultRedirect="YourErrorPage.htm">
                <error statusCode="404" redirect="YourNotFoundPage.htm"/>
                <error statusCode="500" redirect="YourInternalErrorPage.htm"/>
                </customErrors>
                . . .
        </system.web>
        . . .
</configuration>
```

4.15 Παρακολούθηση και καταγραφή ιστορικού

Το .NET Framework παρέχει γκάμα εργαλείων για τη παρακολούθηση και την καταγραφή του ιστορικού και των σφαλμάτων των εφαρμογών. Ο καταγραφέας συμβάντων των Windows (Event Viewer) είναι το συνηθέστερο εργαλείο που χρησιμοποιείται και το .NET παρέχει την κλάση EventLog του χώρου ονομάτων System.Diagnostics για την αλληλεπίδραση με αυτόν. Σημειώνεται ότι εφόσον συμβεί μια καταγραφή, αυτή χρησιμοποιείται για την ενημέρωση του διαχειριστή της εφαρμογής και όχι του χρήστη. Η ενημέρωση του χρήστη για το πρόβλημα που προέκυψε θα πρέπει να εκτελείται σύμφωνα με τα αναγραφόμενα της παραγράφου 4.14. Η κλάση EventLog

χρησιμοποιείται επίσης για την ανάκτηση των καταχωρήσεων που έχουν γίνει στον καταγραφέα, σε συνδυασμό με υλοποίηση κατάλληλης ιστοσελίδας προβολής των καταχωρήσεων.

Μια επίσης συνηθισμένη προσέγγιση για την καταγραφή μη επειγουσών σφαλμάτων είναι η αποστολή μηνυμάτων ηλεκτρονικού ταχυδρομείου σε κάποιο λογαριασμό που παρακολουθείται τακτικά. Το .NET περιέχει κλάσεις στους χώρους ονομάτων System.Web.Mail και System.Net.Mail για αυτό το σκοπό. Για την ενεργοποίηση αυτής της λειτουργίας χρησιμοποιείται το στοιχείο <mailSettings> εντός του στοιχείου <system.net> στο αρχείο web.config της εφαρμογής. Ένα παράδειγμα ρυθμίσεων φαίνεται παρακάτω:

```
<system.net>
    <mailSettings>
      <smtp from="some-email@gmail.com">
        <network enableSsl="true"
              host="smtp.gmail.com"
              port="587"
              userName="some-email@gmail.com"
              password="valid-password" />
      </smtp>
    </mailSettings>
  </system.net>
```

Το ASP.NET παρέχει τη λειτουργία ιχνογράφησης (tracing), η οποία επιτρέπει την προβολή διαγνωστικών πληροφοριών που σχετίζονται με κάθε αίτημα ιστοσελίδας της εφαρμογής. Όταν η λειτουργία είναι ενεργοποιημένη οι πληροφορίες αυτές εμφανίζονται στο τέλος κάθε ιστοσελίδας ή όλες μαζί σε έναν ξεχωριστό οπτικοποιητή (trace.axd).Για το που θα εμφανίζονται οι συγκεκριμένες πληροφορίες είναι θέμα ρυθμίσεων στο αρχείο web.config της εφαρμογής ή στην κάθε ιστοσελίδα της.

Η λειτουργία για όλη την εφαρμογή ενεργοποιείται θέτοντας το χαρακτηριστικό enable=”true” της ετικέτας <trace> του αρχείου web.config. Η συγκεκριμένη ρύθμιση που αφορά όλες τις ιστοσελίδες της εφαρμογής είναι χρήσιμη καθώς δεν απαιτούνται αλλαγές σε κάθε ιστοσελίδα για την απενεργοποίηση ή την ενεργοποίησή της. Η υπηρεσία αυτή συγκεντρώνει τις πληροφιρίες ιχνογράφησης κάθε αιτήματος προς την εφαρμογή μέχρι κάποιο μέγιστο αριθμό αιτημάτων που μπορεί να καθοριστεί από τις ρυθμίσεις. Παράδειγμα ρύθμισης στο επίπεδο εφαρμογής φαίνεται παρακάτω:

```
<system.web>
      <trace enabled="true"
       pageOutput="false"
       requestLimit="40"
       localOnly="false"/>
</system.web>
```

Το ELMAH (Error Logging Modules and Handlers) [32] είναι μια ευκολία καταγραφής σφαλμάτων που συνδέεται πολύ εύκολα με εφαρμογές ASP.NET ως πακέτο NuGet και προσφέρει τις παρακάτω δυνατότητες:

- Καταγραφή των εξαιρέσεων που δεν διαχειρίζονται από τον κώδικα της εφαρμογής.
- Οπτική απεικόνιση σε ιστοσελίδα των καταγεγραμμένων σφαλμάτων.
- Οπτική απεικόνιση σε ιστοσελίδα των λεπτομερειών κάθε σφάλματος που έχει καταγραφεί.
- Κοινοποίηση με email κάθε σφάλματος που προκύπτει, τη στιγμή που αυτό προκύπτει.

Κεφάλαιο 5°

5 Ανάπτυξη Διαδικτυακής εφαρμογής

Η εφαρμογή που αναπτύχθηκε είναι ένα διαδικτυακό ηλεκτρονικό κατάστημα προβολής κινηματογραφικών ταινιών με τίτλο «Online Movies». Οι χρήστες μπορούν να δουν τις διαθέσιμες ταινίες που παρέχει η εφαρμογή, να επιλέξουν κάποια και να την παρακολουθήσουν.

Για τις απαιτήσεις της εργασίας θεωρείται ότι εφόσον κάποιος χρήστης διαθέτει λογαριασμό (username και password) για την είσοδο στην εφαρμογή τότε έχει πληρώσει και το αντίστοιχο ποσό που απαιτείται από κάθε ταινία για να την παρακολουθήσει.

Στο Κεφάλαιο αυτό περιγράφονται οι λειτουργικές απαιτήσεις που ικανοποιεί η εφαρμογή, οι απαιτήσεις ασφάλειας και οι περιπτώσεις χρήσης της.

5.1 Λειτουργικές Απαιτήσεις εφαρμογής

1. Στην εφαρμογή θα μπορεί οποιοσδήποτε να δει τις διαθέσιμες ταινίες του ηλεκτρονικού καταστήματος.
2. Κάθε ταινία περιγράφεται από τα παρακάτω χαρακτηριστικά: Τίτλος ταινίας, είδος, τιμή και βαθμός αξιολόγησης.
3. Οι χρήστες που έχουν δημιουργήσει λογαριασμό έχουν την δυνατότητα να παρακολουθήσουν εικονικά την ταινία που θα επιλέξουν. (Θεωρείται ότι έχει πληρωθεί το αντίστοιχο ποσό της κάθε ταινίας.)
4. Για να συνδεθεί κάποιος στην εφαρμογή (login) πρέπει να δώσει το όνομα χρήστη και τον κωδικό πρόσβασής του.
5. Για τους χρήστες που δεν έχουν δημιουργήσει λογαριασμό παρέχεται μόνο μια λίστα με τις διαθέσιμες ταινίες και οι λεπτομέρειες της κάθε ταινίας.
6. Όλοι οι χρήστες της εφαρμογής μπορούν να ψάξουν για την ταινία που επιθυμούν ανάλογα με το είδος της ή με λέξεις που υπάρχουν στον τίτλο της.
7. Η διαχείριση των ταινιών που διαθέτει το κατάστημα γίνεται από κάποιο χρήστη της εφαρμογής που έχει αναλάβει το ρόλο του Store Administrator. Μπορεί να υπάρχουν παραπάνω από ένας Store Administrators.
8. Στους Store Administrators δίνεται η δυνατότητα να προσθέσουν ταινίες στη λίστα με τις διαθέσιμες του καταστήματος. Αυτοί μπορούν επίσης να διαγράψουν μια ταινία ή να διορθώσουν τα στοιχεία της.

5.2 Απαιτήσεις Ασφάλειας εφαρμογής

1. Για την δημιουργία λογαριασμού (registration) στην εφαρμογή πρέπει να δοθούν τα παρακάτω στοιχεία: όνομα χρήστη, διεύθυνση e-mail, κωδικό πρόσβασης κρυφή ερώτηση και κρυφή απάντηση (για την περίπτωση που κάποιος ξεχάσει τον κωδικό πρόσβασής του).
2. Οι συνδεδεμένοι χρήστες έχουν την δυνατότητα να αλλάξουν τον κωδικό πρόσβασης εφόσον το επιθυμούν.
3. Υπάρχει ένας χρήστης με ρόλο Administrator ο οποίος μπορεί να διαχειρίζεται όλους τους υπόλοιπους χρήστες της εφαρμογής. Μπορεί να αναθέτει ρόλο Store Administrator σε όποιον χρήστη επιθυμεί καθώς επίσης μπορεί να δημιουργεί χρήστες, να διορθώνει τα στοιχεία των υπαρχόντων χρηστών ή ακόμα και να τους διαγράφει.

4. Ο Administrator μπορεί να προσθέτει, να διορθώνει και να διαγράφει ρόλους. Για παράδειγμα μπορεί να προσθέσει κάποιο νέο ρόλο χρηστών οι οποίοι θα μπορούν να βλέπουν δωρεάν κάποιες συγκεκριμένες ταινίες. Επιπλέον διαθέτει όλες τις δυνατότητες που διαθέτει και ο Store Administrator.
5. Αν κάποιος χρήστης εισάγει λανθασμένο κωδικό πρόσβασης πέντε φορές τότε ο λογαριασμός του κλειδώνεται και δεν μπορεί να εγγραφεί στην εφαρμογή. Ο Administrator σε αυτήν την περίπτωση μπορεί να ξεκλειδώσει τον λογαριασμό του.
6. Αν κάποιος χρήστης ξεχάσει τον κωδικό πρόσβασης μπορεί να του αποσταλεί καινούριος μέσω email. Για το σκοπό αυτό θα πρέπει να απαντήσει την μυστική ερώτηση που του γίνεται με την μυστική απάντηση, στοιχεία που είχε δώσει με την αρχική εγγραφή του στην εφαρμογή. Μόλις λάβει το email με τον καινούριο κωδικό πρόσβασης, αφού τον χρησιμοποιήσει για την επόμενη είσοδό του στην εφαρμογή, θα πρέπει να τον αλλάξει άμεσα.
7. Η βάση δεδομένων για τις ταινίες μπορεί να είναι διαφορετική από τη βάση δεδομένων με τα στοιχεία των χρηστών.
8. Το πρωτόκολλο επικοινωνίας μεταξύ clients και του server πρέπει να είναι SSL.

5.3 Περιπτώσεις χρήσης

Οι περιπτώσεις χρήσεις της εφαρμογής διαχωρίζονται ανάλογα με το ποιος είναι ο χρήστης που ενεργοποιεί τις λειτουργίες της εφαρμογής. Οι περιπτώσεις χρήσης που παρουσιάζονται παρακάτω είναι ανάλογα με το αν ο χρήστης είναι:

- Μη αυθεντικοποιημένος (Unauthenticated)
- Αυθεντικοποιημένος (Authenticated)
- Διαχειριστής καταστήματος(Store Administrator)
- Γενικός διαχειριστής εφαρμογής (Administrator)

5.3.1 Μη αυθεντικοποιημένος χρήστης

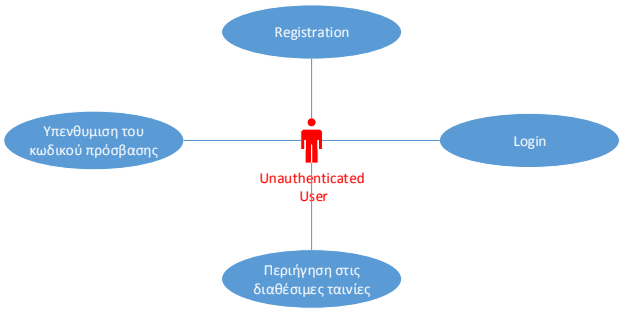
Ένας μη αυθεντικοποιημένος χρήστης μπορεί να δημιουργήσει λογαριασμό στην εφαρμογή, να συνδεθεί με την εφαρμογή και περιηγηθεί στις ταινίες που αυτή διαθέτει.

Για να κάνει registration πρέπει να δώσει τα στοιχεία: όνομα χρήστη, κωδικός πρόσβασης, διεύθυνση e-mail, μυστική ερώτηση και μυστική απάντηση.

Για να συνδεθεί αρκεί να δώσει το όνομα χρήστη και τον κωδικό πρόσβασης.

Για να περιηγηθεί στις ταινίες επιλέγει τους αντίστοιχους υπερσυνδέσμους που υπάρχουν στην εφαρμογή.

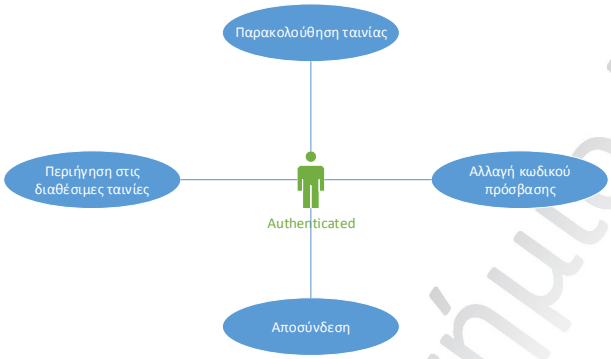
Για να γίνει υπενθύμιση του κωδικού πρόσβασης, θα πρέπει να δοθεί η μυστική απάντηση στην μυστική ερώτηση του συγκεκριμένου χρήστη και κατόπιν να αποσταλεί το μήνυμα με τον καινούριο κωδικό πρόσβασης (ο οποίος θα πρέπει άμεσα να αλλαχθεί).



Εικόνα 16. Περιπτώσεις χρήσης μη αυθεντικοποιημένου χρήστη

5.3.2 Αυθεντικοποιημένος χρήστης

Αυθεντικοποιημένος χρήστης είναι αυτός που έχει συνδεθεί (login) στην εφαρμογή επιτυχώς. Αυτός μπορεί να περιηγηθεί στις διαθέσιμες ταινίες, να παρακολουθήσει όποια επιθυμεί, να αλλάξει τον κωδικό του πρόσβασης και να αποσυνδεθεί από την εφαρμογή.



Εικόνα 17.Περιπτώσεις χρήσης αυθεντικοποιημένου χρήστη

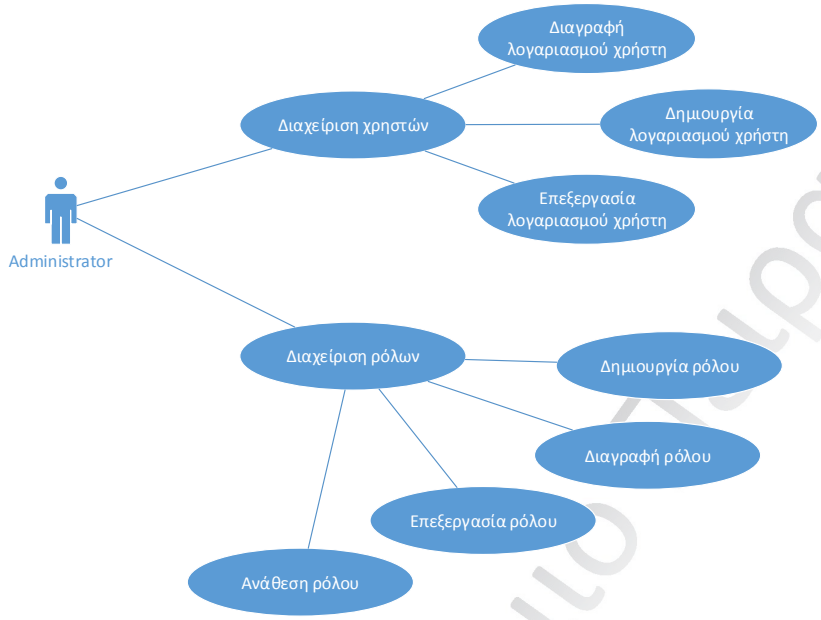
5.3.3 Διαχειριστής καταστήματος

Ο διαχειριστής του καταστήματος είναι ένας αυθεντικοποιημένος χρήστης. Άρα οι λειτουργίες της παραγράφου 5.3.2 μπορούν να εκτελεσθούν και από αυτόν. Επιπλέον μπορεί να δημιουργήσει ταινίες στη λίστα με τις διαθέσιμες, να διαγράψει όποια επιθυμεί η να διορθώσει κάποια στοιχεία. Τα στοιχεία τα οποία μπορεί να επεξεργαστεί για τις ταινίες είναι τα: τίτλος ταινίας, είδος ταινίας, τιμή παρακολούθησης και βαθμός αξιολόγησης.

Εικόνα 18. Περιπτώσεις χρήσης διαχειριστή καταστήματος

5.3.4 Γενικός διαχειριστής εφαρμογής

Ο γενικός διαχειριστής της εφαρμογής μπορεί να εκτελέσει τις ίδιες λειτουργίες με τον Store Administrator και επιπλέον μπορεί να: δημιουργήσει νέους λογαριασμούς χρηστών, διαγράψει λογαριασμούς χρηστών, τροποποιήσει στοιχεία χρηστών (εκτός από τον κωδικό πρόσβασης), δημιουργήσει καινούριους ρόλους, να διαγράψει ρόλους, αναθέσει ρόλους σε χρήστες.



Εικόνα 19. Περιπτώσεις χρήσης γενικού διαχειριστή

Κεφάλαιο 6°

6 Υλοποίηση εφαρμογής

Το περιβάλλον που χρησιμοποιήθηκε για την ανάπτυξη της διαδικτυακής εφαρμογής Online Movies ήταν το Microsoft Visual Studio 2012 Ultimate (VS12) σε λειτουργικό σύστημα Windows 8 Professional. Ο ενσωματωμένος server που περιλαμβάνεται με το IDE VS12 για την ανάπτυξη και δοκιμή ιστοτόπων είναι ο ASP.NET Development Web Server επίσης γνωστός ως Cassini. Επειδή ο συγκεκριμένος server δεν υποστηρίζει συνδέσεις SSL, απαραίτητο συστατικό για ασφαλείς εφαρμογές διαδικτύου, το VS12 διαθέτει ως εναλλακτική επιλογή τη χρήση του IIS Express server τον οποίο και χρησιμοποιήσαμε αντί του Cassini. Για τις βάσεις δεδομένων της εφαρμογής χρησιμοποιήθηκε ο Microsoft SQL Server 2012 Express Edition.

Στο κεφάλαιο αυτό γίνεται μια παρουσίαση του τρόπου υλοποίησης της εφαρμογής που αφορά τις λειτουργικές της απαιτήσεις.

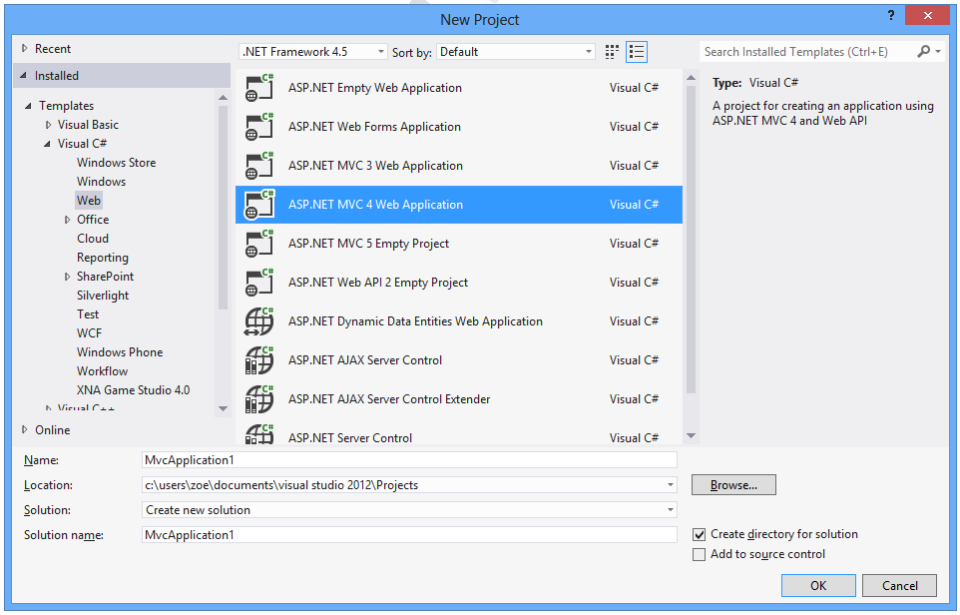
6.1 Προαπαιτούμενα

Για την ανάπτυξη της εφαρμογής απαιτήθηκαν τα παρακάτω:

- Εγκατάσταση του IDE VS12 στο σύστημα.
- Εγκατάσταση του Microsoft SQL Server 2012 Express Edition από το [33]. Το VS12 διαθέτει τα εργαλεία για την αλληλεπίδραση με αυτόν.

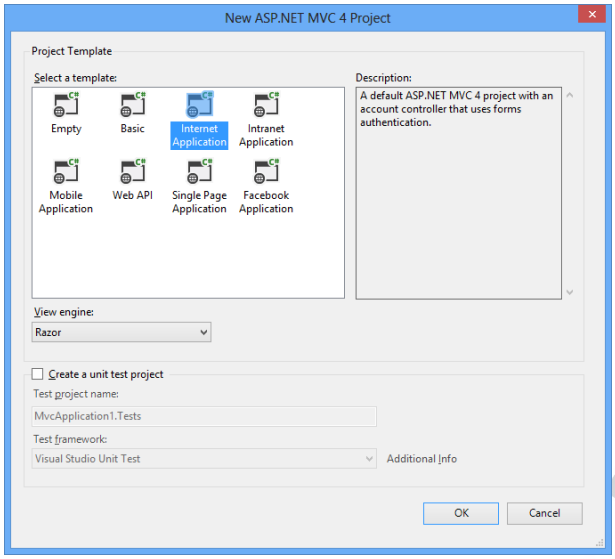
6.2 Δημιουργία του Project

Έχοντας ξεκινήσει το VS12 δημιουργήθηκε αρχικά το project της εφαρμογής, δηλαδή μια καινούρια εφαρμογή ASP.NET MVC 4:



Εικόνα 20. Δημιουργία του project

Χρησιμοποιήθηκε το παρεχόμενο πρότυπο από το VS12 για διαδικτυακή εφαρμογή. Στο συγκεκριμένο πρότυπο παρέχεται ένας Controller διαχείρισης λογαριασμών της εφαρμογής (AccountController) με τα αντίστοιχα Views ο οποίος χρησιμοποιεί forms authentication και Razor View Engine.



Εικόνα 21. Επιλογή template Internet Application

6.3 Δημιουργία του μοντέλου δεδομένων

Για τη δημιουργία της βάσης δεδομένων της εφαρμογής που αφορά τις διαθέσιμες ταινίες χρησιμοποιήθηκε το Entity Framework της Microsoft. Η τεχνική με την οποία δημιουργήθηκε η βάση δεδομένων που περιλαμβάνει τον πίνακα Movie ήταν η Code First [14]. Δημιουργήθηκε δηλαδή αρχικά το αρχείο στο οποίο περιλαμβάνεται ο κώδικας C# για τη δημιουργία των απαιτούμενων πινάκων της βάσης δεδομένων. Το αρχείο αυτό τοποθετήθηκε στο φάκελο Models του project της εφαρμογής:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;
using System.ComponentModel.DataAnnotations;

namespace MoviesOnline.Models
{
    public class Movie
    {
        public int ID { get; set; }

        [Required]
        public string Title { get; set; }

        [DataType(DataType.Date)]
        public DateTime ReleaseDate { get; set; }

        [Required]
        public string Genre { get; set; }

        [Range(1, 100)]
        [DataType(DataType.Currency)]
        public decimal Price { get; set; }

        [StringLength(5)]
        public string Rating { get; set; }

        public string photoPath { get; set; }
    }

    public class MovieDBContext : DbContext
    {
        public DbSet<Movie> Movies { get; set; }
    }
}
```

Εικόνα 22. Κώδικας κλάσης Movie

Η κλάση Movie αντιπροσωπεύει μία ταινία στη βάση δεδομένων της εφαρμογής. Κάθε εγγραφή ταινίας περιλαμβάνει τα πεδία:

- ID τύπου int. Το πρωτεύον κλειδί της εγγραφής.
- Title τύπου string. Ο τίτλος της ταινίας.
- ReleaseDate τύπου DateTime. Η ημερομηνία κυκλοφορίας της ταινία.
- Genre τύπου string. Το είδος της ταινίας.
- Price τύπου decimal. Η τιμή που κοστίζει η παρακολούθηση της ταινίας.
- Rating τύπου string. Ο βαθμός αξιολόγησης της ταινίας.
- photoPath τύπου string. Η διαδρομή στην οποία είναι αποθηκευμένη η εικόνα του εξωφύλλου της ταινίας.

Όπως φαίνεται και στην Εικόνα 22, το αρχείο περιλαμβάνει επιπλέον μία κλάση, την MovieDBContext. Η κλάση αυτή αντιπροσωπεύει το σύνολο των εγγραφών που υπάρχουν στον πίνακα Movies της βάσης δεδομένων της εφαρμογής. Μέσω αυτής γίνεται η διαχείριση όλων των ταινιών που υπάρχουν στη βάση δεδομένων.

6.4 Σύνδεση της εφαρμογής με τη βάση δεδομένων

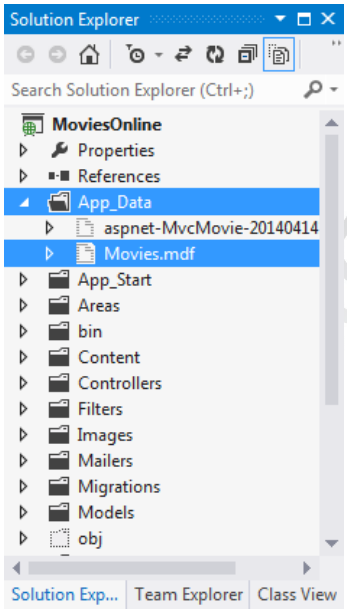
Η κλάση MovieDBContext είναι επίσης υπεύθυνη για τη σύνδεση της εφαρμογής με τη βάση δεδομένων. Προσφέρει την αντιστοίχιση των αντικειμένων της κλάσης Movie στις εγγραφές της βάσης δεδομένων. Για να καθοριστεί με ποια βάση δεδομένων θα συνδεθεί η εφαρμογή προστέθηκε η παρακάτω πληροφορία σύνδεσης στο αρχείο web.config της εφαρμογής:

```
<add name="MovieDBContext"
connectionString="Data Source=(LocalDB)\v11.0;
                  AttachDbFilename=|DataDirectory|Movies.mdf;
providerName="System.Data.SqlClient"  Integrated Security=True" />
```

Εικόνα 23. String σύνδεσης με τη βάση δεδομένων Movies

Η βάση δεδομένων υλοποιείται στον Microsoft SQL Server 2012 Express Edition που είναι εγκατεστημένος στο σύστημα ανάπτυξης της εφαρμογής και συγκεκριμένα σε mode λειτουργίας LocalDb, όπως υποδηλώνεται από το τμήμα Data Source=(LocalDb)\v11.0. Το συγκεκριμένο mode απευθύνεται σε προγραμματιστές που βρίσκονται στη φάση της ανάπτυξης εφαρμογών και επιταχύνει τη δημιουργία και τη σύνδεση με βάσεις δεδομένων. Κατά τη φάση της εγκατάστασης της εφαρμογής για κανονική χρήση θα χρησιμοποιηθεί ο Microsoft SQL Server Express 2012.

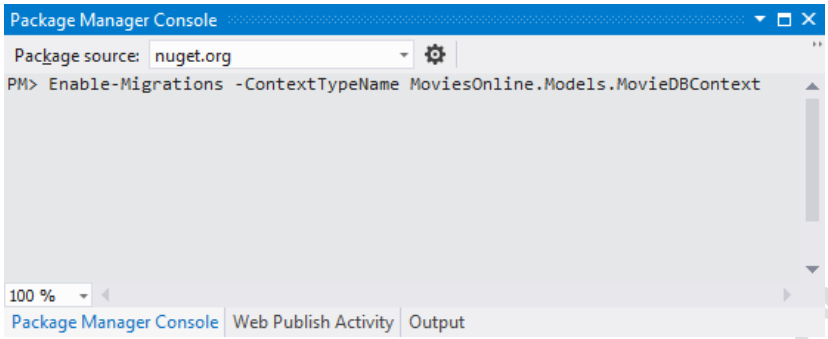
Το αρχείο που περιλαμβάνει τη βάση δεδομένων Movies αποθηκεύεται στον φάκελο App_Data του project με όνομα Movies.mdf και συνήθως είναι κρυμμένο:



Εικόνα 24. Θέση αρχείου βάσης δεδομένων Movies.mdf

Επιπλέον χρησιμοποιήθηκε η λειτουργία Code First Migrations που παρέχεται από το Entity Framework του .NET. Το συγκεκριμένο χαρακτηριστικό επιτρέπει την αλλαγή του μοντέλου δεδομένων όπως αυτό καθορίζεται από τις αντίστοιχες κλάσεις μοντέλων, χωρίς να χρειάζεται να διαγράφεται όλη η βάση δεδομένων με τα περιεχόμενά της.

Για την ενεργοποίηση της λειτουργίας δόθηκε η εντολή Enable-Migrations στον Package Manager Console του VS12:



Εικόνα 25. Εντολή ενεργοποίησης Code First Migrations

Με την εντολή αυτή δημιουργήθηκε ένας φάκελος Migrations στο project και εντός αυτού η κλάση Configuration στο αρχείο Configuration.cs. Στην κλάση Configuration που φαίνεται στην Εικόνα 26 περιλαμβάνεται η μέθοδος Seed, η οποία χρησιμοποιήθηκε για την γρήγορη εισαγωγή δεδομένων στην βάση Movies.

Στη συνέχεια δόθηκε η εντολή add - migration Initial στον Package Manager Console, η οποία δημιούργησε τον απαιτούμενο κώδικα για τη δημιουργία της βάσης δεδομένων, φτιάχνοντας δηλαδή την κλάση Initial εντός του φακέλου Migrations. Ο κώδικας που δημιουργήθηκε φαίνεται στην Εικόνα 27. Σε αυτήν την κλάση, η μέθοδος Up δημιουργεί τον πίνακα Movies που αντιστοιχεί στο μοντέλο που δημιουργήθηκε, ενώ η Down τον σβήνει.

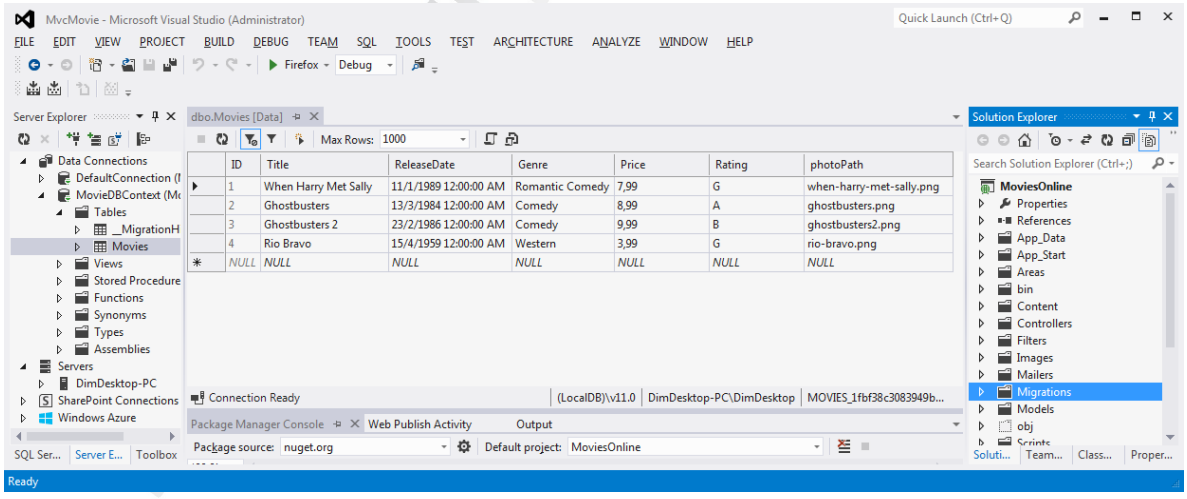
Τέλος, για την εκτέλεση του ανωτέρω κώδικα, δηλαδή της δημιουργίας της βάσης δεδομένων και της φόρτωσής της με τα δεδομένα, δόθηκε η εντολή update-database στον Package Manager Console. Με την εντολή αυτή εκτελείται πρώτα η μέθοδος Up της κλάσης Initial και μετά η Seed της κλάσης Configuration. Το τελικό αποτέλεσμα είναι ότι από τη μια μεριά, η βάση δεδομένων θα δημιουργείται πάντα κατόπιν οποιασδήποτε τροποποίησής της, μέσω του κώδικα που αναφέρθηκε, ενώ από την άλλη θα περιέχει κάποια στοιχεία έτσι ώστε αυτά να μην εισάγονται ξανά. Ελέγχοντας τα στοιχεία που περιέχει η βάση δεδομένων μέσω του VS12 βλέπουμε ότι πράγματι έχουν εισαχθεί, όπως φαίνεται στην Εικόνα 28.

```
  protected override void Seed(MoviesOnline.Models.MovieDBContext
context)
        {
            //  This method will be called after migrating to the latest
version.
            //  use the DbSet<T>.AddOrUpdate() helper extension method
            //  to avoid creating duplicate seed data.

            context.Movies.AddOrUpdate(i => i.Title,
        new Movie
        {
            Title = "When Harry Met Sally",
            ReleaseDate = DateTime.Parse("1989-1-11"),
            Genre = "Romantic Comedy",
            Rating = "G",
            Price = 7.99M,
            photoPath = "when-harry-met-sally.png"
        },
         new Movie
         {
             Title = "Ghostbusters ",
             ReleaseDate = DateTime.Parse("1984-3-13"),
             Genre = "Comedy",
             Rating = "A",
             Price = 8.99M,
             photoPath = "ghostbusters.png"
         },
         new Movie
         {
             Title = "Ghostbusters 2",
             ReleaseDate = DateTime.Parse("1986-2-23"),
             Genre = "Comedy",
             Rating = "B",
             Price = 9.99M,
             photoPath = "ghostbusters2.png"
         },
       new Movie
       {
           Title = "Rio Bravo",
           ReleaseDate = DateTime.Parse("1959-4-15"),
           Genre = "Western",
           Rating = "G",
           Price = 3.99M,
           photoPath = "rio-bravo.png"
       }
   );
        }
    }
```

Εικόνα 26. Μέθοδος Seed κλάσης Configuration για εισαγωγή δεδομένων στη βάση

```
namespace MoviesOnline.Migrations
{
    using System;
    using System.Data.Entity.Migrations;

    public partial class Initial : DbMigration
    {
        public override void Up()
        {
            CreateTable(
                "dbo.Movies",
                c => new
                    {
                        ID = c.Int(nullable: false, identity: true),
                        Title = c.String(),
                        ReleaseDate = c.DateTime(nullable: false),
                        Genre = c.String(),
                        Price = c.Decimal(nullable: false, precision: 18,
scale: 2),
                    })
                .PrimaryKey(t => t.ID);

        }

        public override void Down()
        {
            DropTable("dbo.Movies");
        }
    }
}
```

Εικόνα 27. Κλάση δημιουργίας της βάσης δεδομένων, Initial



Εικόνα 28. Επιβεβαίωση εισαχθέντων στοιχείων στη βάση δεδομένων Movies

6.5 Πρόσβαση στα δεδομένα της βάσης με τον MoviesController

Για την πρόσβαση στα δεδομένα που αφορούν τις ταινίες υλοποιήθηκε ο παρακάτω Controller:

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MoviesOnline.Models;

namespace MoviesOnline.Controllers
{
    [RequireHttps]
    [Authorize(Roles = "Administrator, Store Administrator")]
    public class MoviesController : Controller
    {
        private MovieDBContext db = new MovieDBContext();

        // GET: /Movies/Watchnow
        [OverrideAuthorization]
        public ActionResult Watchnow(int id = 0)
        {…}
        // GET: /Movies/Details/5
        [AllowAnonymous]
        public ActionResult Details(int id = 0)
        {…}
        // GET: /Movies/Create
        public ActionResult Create()
        {…}
        // POST: /Movies/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Movie movie)
        {…}
        // GET: /Movies/Edit/5
        public ActionResult Edit(int id = 0)
        {…}
        // POST: /Movies/Edit/5
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Movie movie)
        {…}
        // GET: /Movies/Delete/5
        public ActionResult Delete(int id = 0)
        {…}
        // POST: /Movies/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {…}
        // GET: /Movies/SearchIndex/
        [AllowAnonymous]
        public ActionResult SearchIndex(string movieGenre, string
searchString)
        {…}
    }
}
```

Εικόνα 29. MoviesController

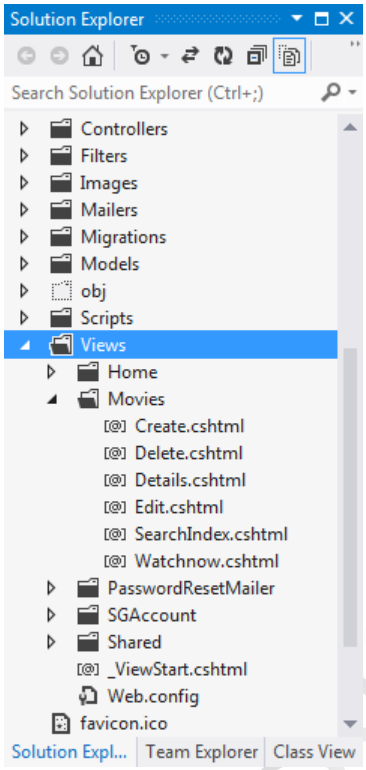
Μέσω αυτής της κλάσης μπορούν να γίνουν οι παρακάτω ενέργειες:

- Δημιουργία μια ταινίας (Μέθοδοι Create)

- Διαγραφή ταινίας από τη βάση δεδομένων (Μέθοδοι Delete)
- Επεξεργασία των στοιχείων μιας ταινίας (Μέθοδοι Edit)
- Αναζήτηση και προβολή υπαρχόντων ταινιών (Μέθοδος SearchIndex)
- Παρακολούθηση ταινίας (Μέθοδος WatchNow)
- Προβολή των λεπτομερειών μιας ταινίας (Μέθοδος Details)

6.6 Δημιουργία Views του MoviesController

Για κάθε μέθοδο του MoviesController δημιουργήθηκε και το αντίστοιχο αρχείο στον φάκελο Views και εντός του υποφακέλου Movies:

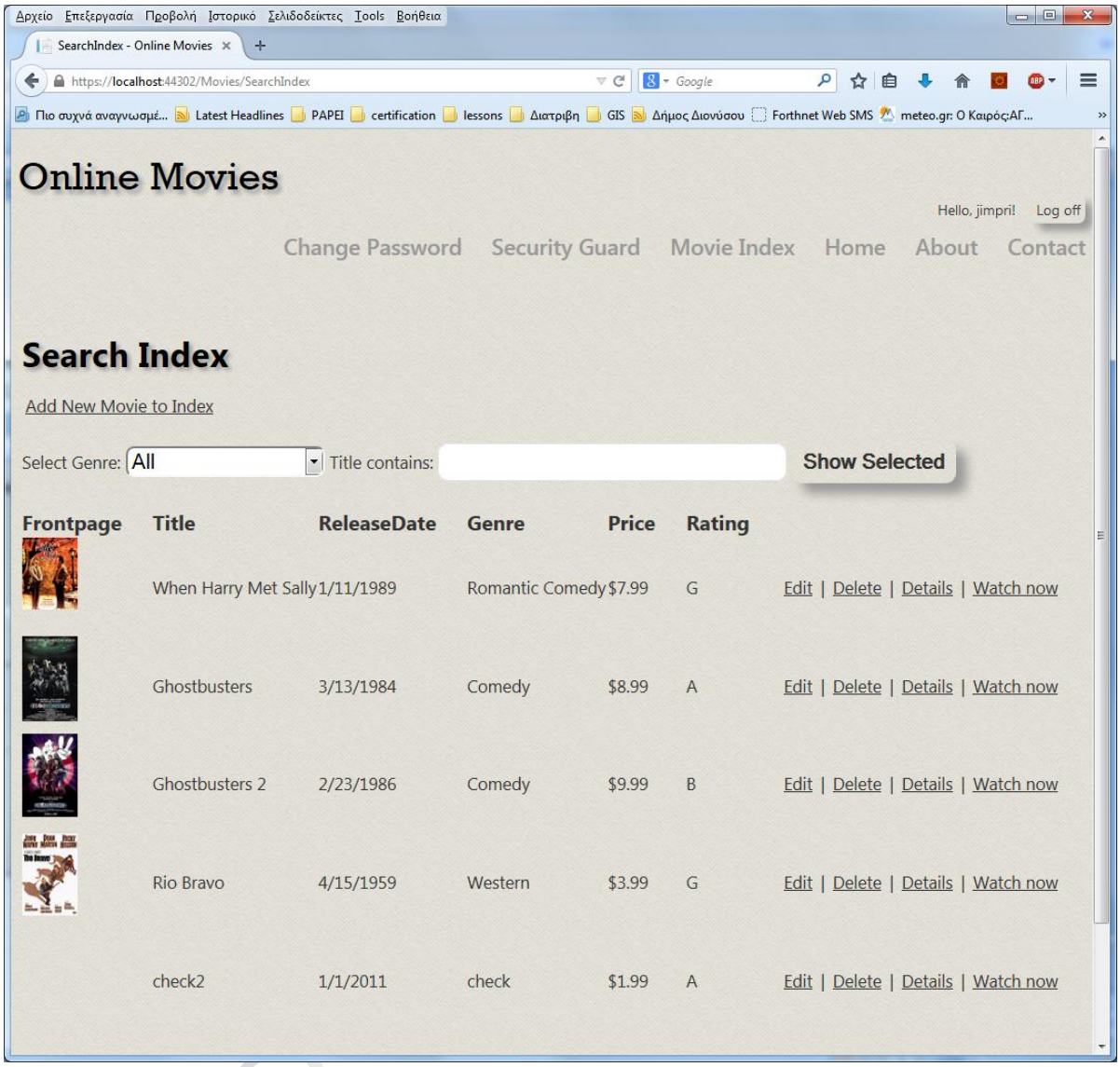


Εικόνα 30. MoviesController Views

Εκτελώντας την εφαρμογή, για να δούμε ποιες είναι οι διαθέσιμες ταινίες που υπάρχουν στο κατάστημα και άρα στη βάση δεδομένων χρησιμοποιούμε το url https://localhost:44302/Movies/SearchIndex που παρέχεται επιλέγοντας “Movie Index” από το μενού της εφαρμογής.

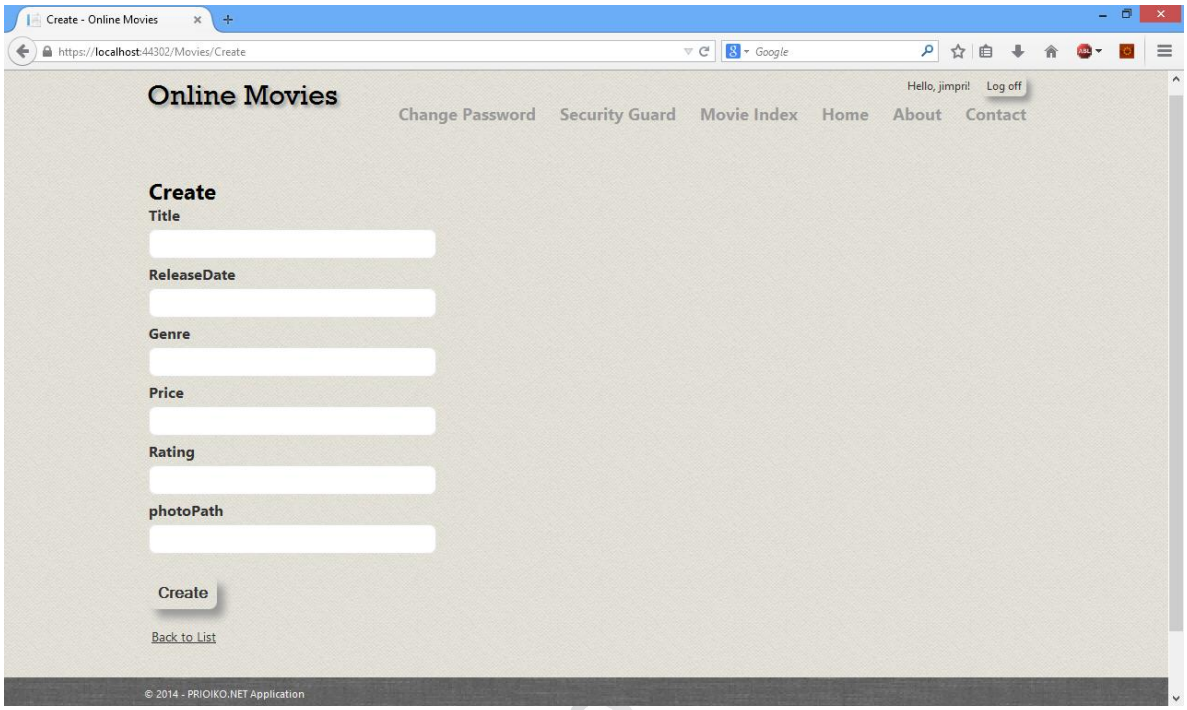
Εικόνα 31. Μενού εφαρμογής Online Movies

Στη σελίδα αυτή παρέχεται και η δυνατότητα αναζήτησης ταινιών ανάλογα με το είδος ταινίας και τον τίτλο.



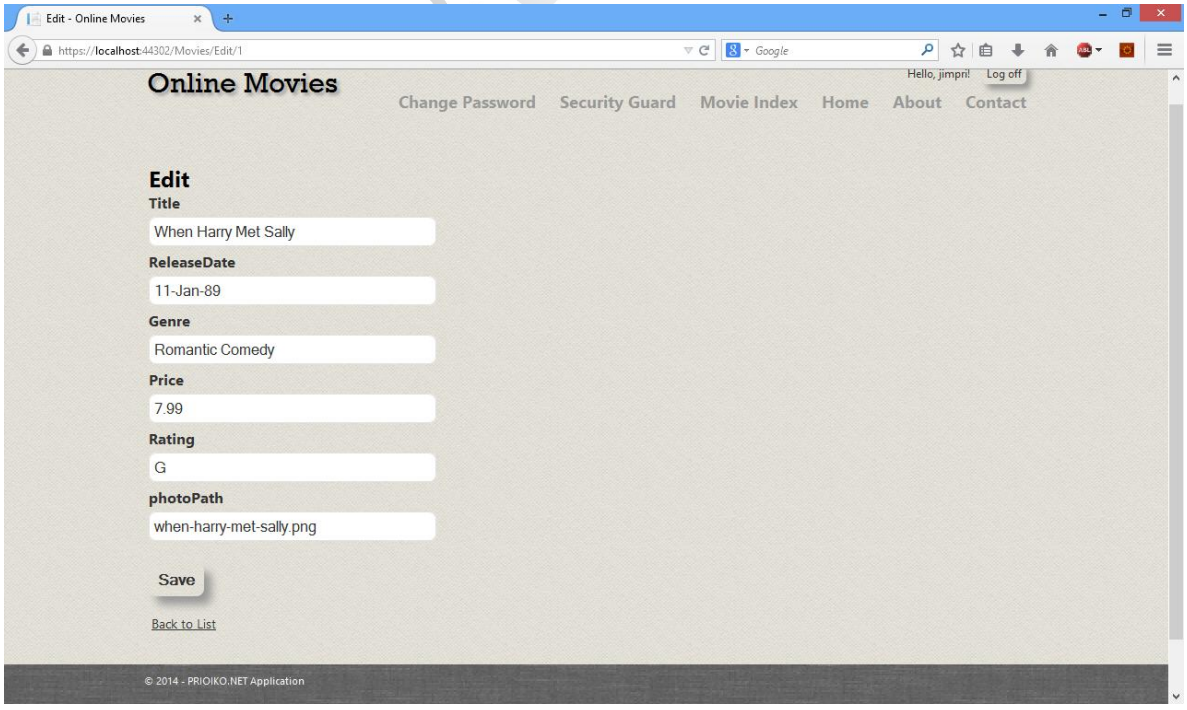
Εικόνα 32. Σελίδα προβολή λίστας ταινιών και αναζήτηση ταινίας, Search Index

Για την δημιουργία νέας ταινίας επιλέγεται “Add New Movie to Index” από τη σελίδα “Search Index” και η σελίδα που εμφανίζεται είναι η παρακάτω, με url https://localhost:44302/Movies/Create.



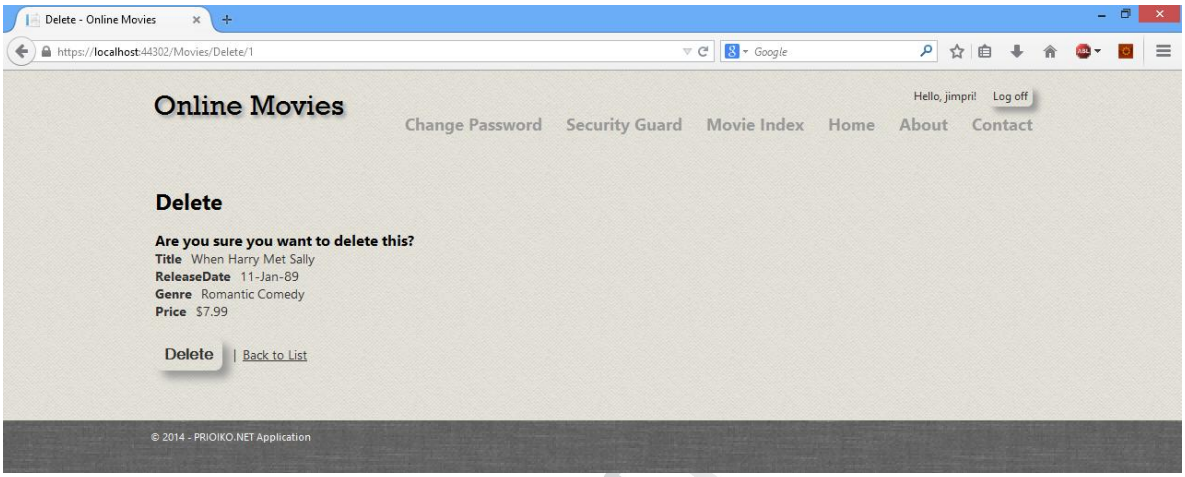
Εικόνα 33. Σελίδα δημιουργίας ταινίας, Create

Για την τροποποίηση των στοιχείων μιας ταινίας επιλέγεται “Edit” από τη σελίδα “Search Index” και η σελίδα που εμφανίζεται είναι η παρακάτω, με url “https://localhost:44302/Movies/Edit/1”. Ο αριθμός 1 στο url αντιστοιχεί στο ID της ταινίας:



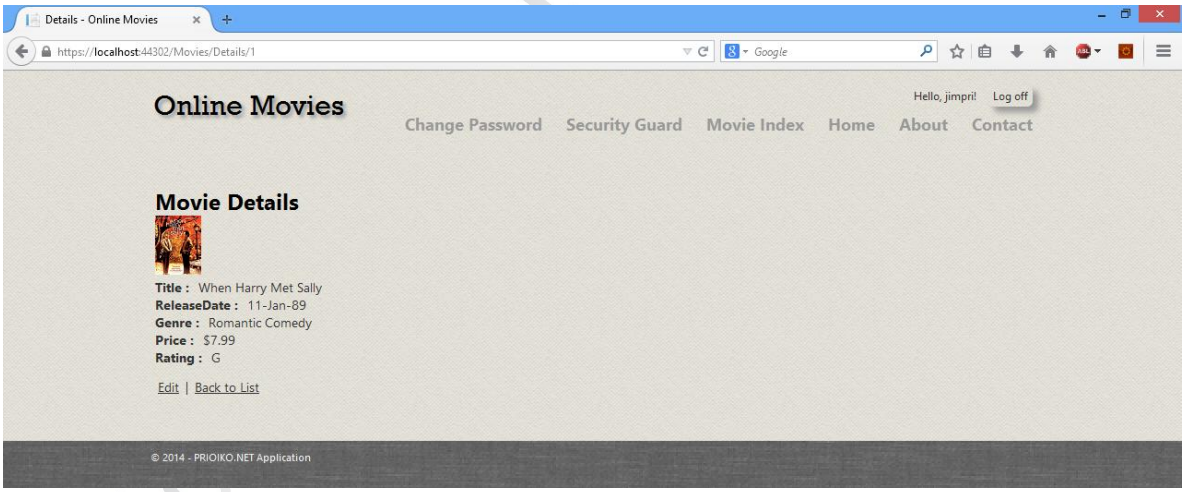
Εικόνα 34. Σελίδα τροποποίησης στοιχείων ταινίας, Edit

Για την διαγραφή μιας ταινίας από τη βάση επιλέγεται “Delete” από τη σελίδα “Search Index” και η σελίδα που εμφανίζεται είναι η παρακάτω, με url “https://localhost:44302/Movies/Delete/1”. Ο αριθμός 1 στο url αντιστοιχεί στο ID της ταινίας:



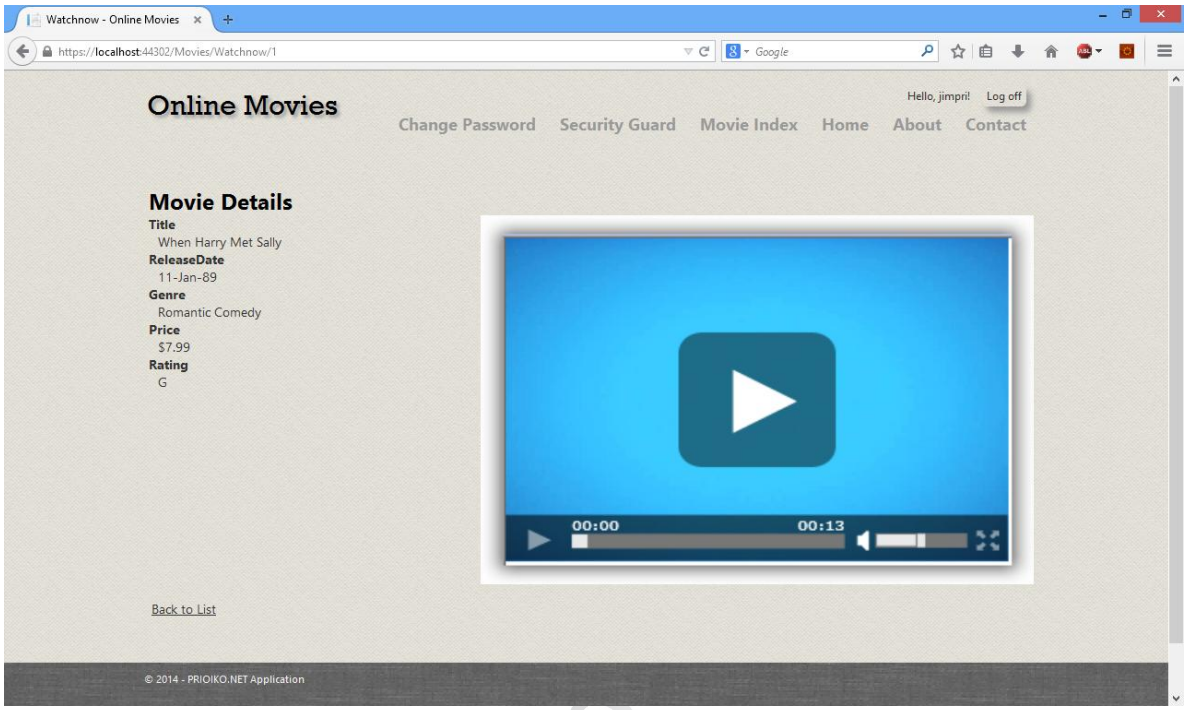
Εικόνα 35.Σελίδα διαγραφής ταινίας, Delete

Για την προβολή των χαρακτηριστικών μιας ταινίας επιλέγεται “Details” από τη σελίδα “Search Index” και η σελίδα που εμφανίζεται είναι η παρακάτω, με url “https://localhost:44302/Movies/Details/1”. Ο αριθμός 1 στο url αντιστοιχεί στο ID της ταινίας:



Εικόνα 36. Σελίδα προβολής χαρακτηριστικών ταινίας, Details

Τέλος, για την παρακολούθηση μιας ταινίας επιλέγεται “WatchNow” από τη σελίδα “Search Index” και η σελίδα που εμφανίζεται είναι η παρακάτω, με url “https://localhost:44302/Movies/Watchnow/1”. Ο αριθμός 1 στο url αντιστοιχεί στο ID της ταινίας:



Εικόνα 37.Σελίδα παρακολούθησης ταινίας, WatchNow

Κεφάλαιο 7°

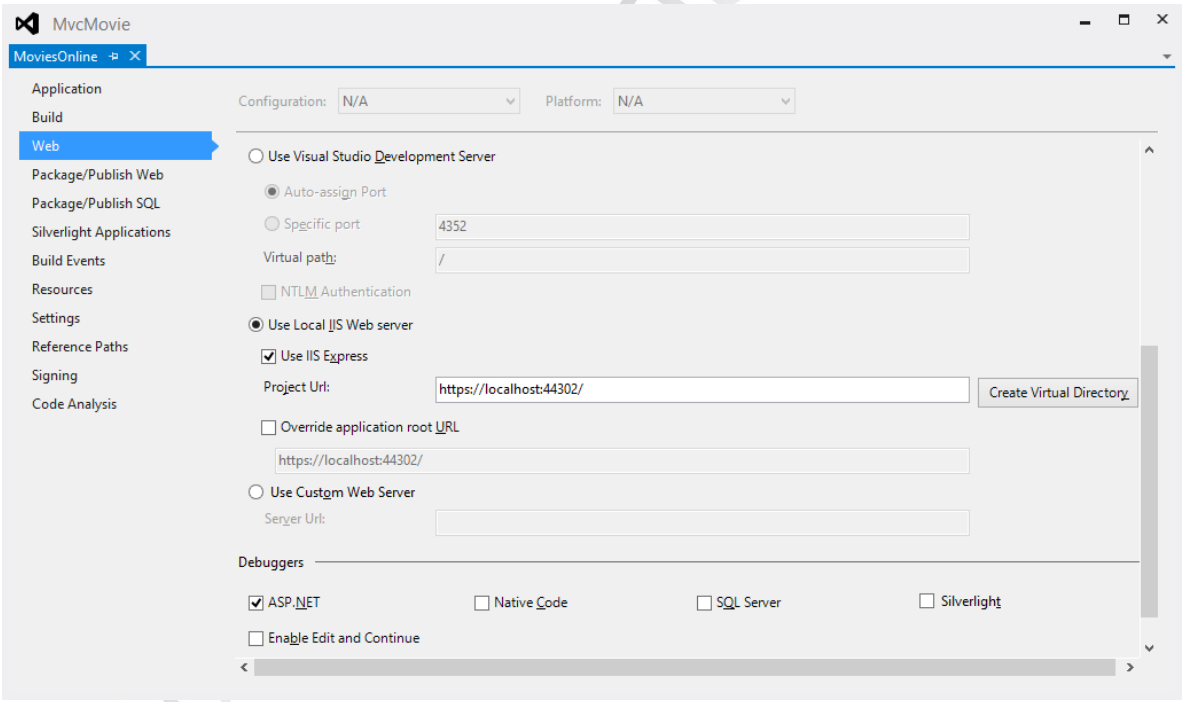
7 Υλοποίηση μέτρων ασφαλείας της εφαρμογής

Στο Κεφάλαιο παρουσιάζεται αρχικά ο τρόπος με τον οποίο ενεργοποιήθηκε η χρήση του πρωτοκόλλου SSL, απαραίτητο αντίμετρο κατά αρκετών απειλών και στη συνέχεια τα μέτρα ασφαλείας που έχουν υλοποιηθεί σε επίπεδο εφαρμογής, σε κάθε κατηγορία αδυναμιών σύμφωνα με την παράγραφο 3.10.

7.1 Υλοποίηση SSL

Για να θωρακιστεί η εφαρμογή από απειλές κατά της αυθεντικοποίησης χρηστών, των ευαίσθητων δεδομένων, της διαχείρισης συνόδου και της τροποποίησης παραμέτρων υλοποιήθηκε πρωτόκολλο SSL για την επικοινωνία client και server.

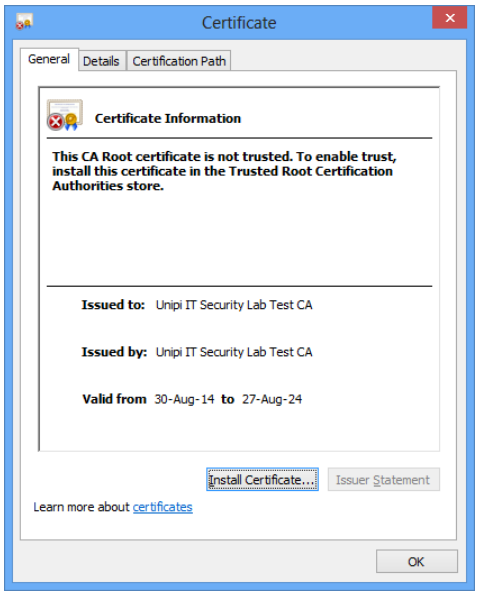
Ο server που υποστηρίζει το πρωτόκολλο είναι ο IIS Express στη φάση της ανάπτυξης της εφαρμογής. Η ρύθμισή του έγινε μέσω του VS12 στη θύρα 44302.



Εικόνα 38. Ρύθμιση για χρήση IIS Express

7.1.1 Δημιουργία Αρχής Πιστοποίησης και πιστοποιητικού server

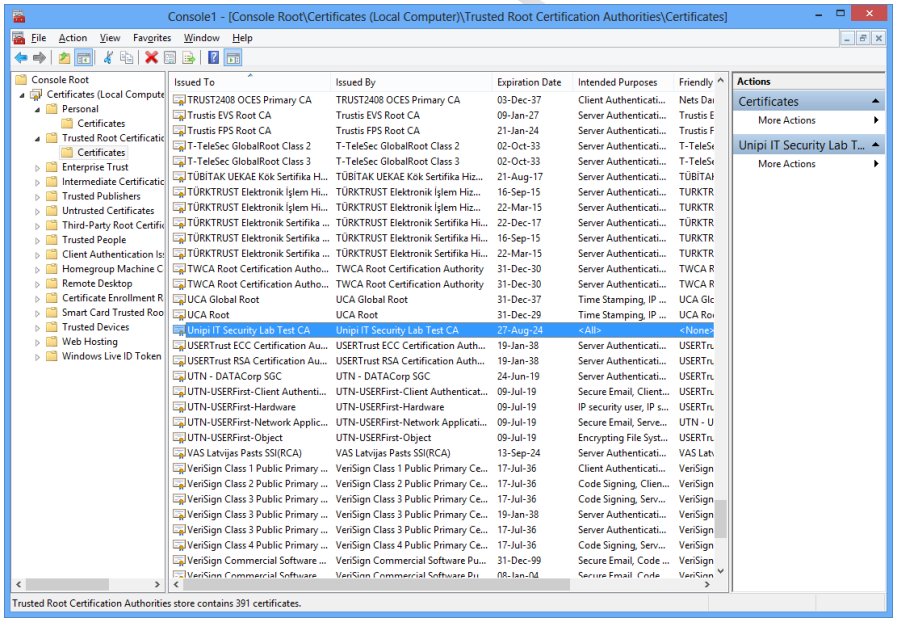
Με την χρήση του εργαλείου OpenSSL δημιουργήθηκε Αρχή Πιστοποίησης με όνομα Unipi IT Security Lab Test CA με αυτο-υπογεγραμμένο Πιστοποιητικό όπως φαίνεται στην Εικόνα 39 [34].



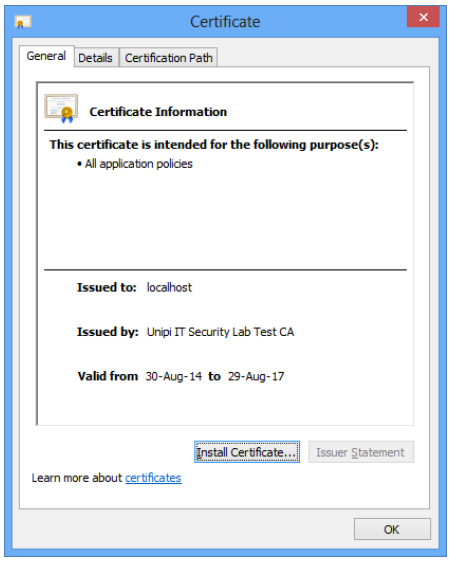
Εικόνα 39. Πιστοποιητικό Αρχής Πιστοποίησης

Έγινε εισαγωγή του πιστοποιητικού στα έμπιστα πιστοποιητικά του συστήματος (Εικόνα 40), με σκοπό την εξομοίωση μιας πραγματικής Αρχής Πιστοποίησης.

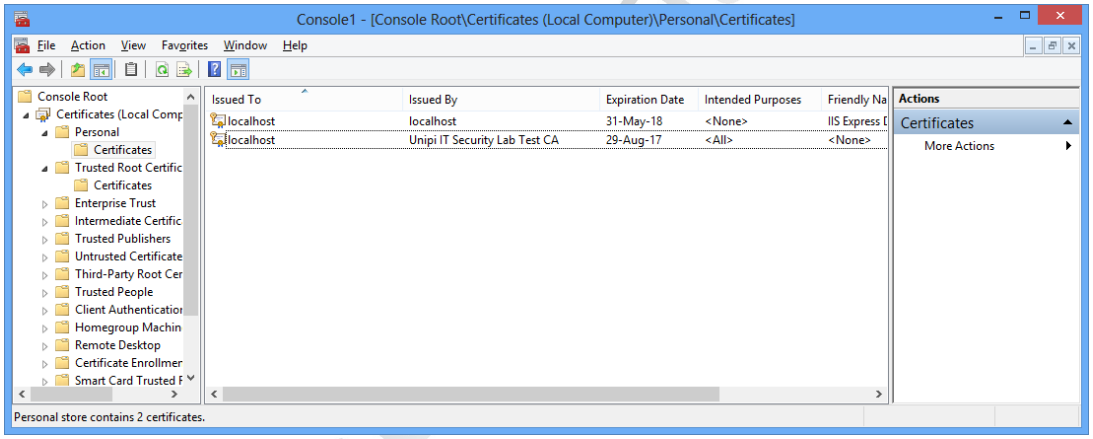
Στη συνέχεια δημιουργήσαμε το πιστοποιητικό για τον server με όνομα localhost, υπογεγραμμένο από την Αρχή Πιστοποίησης Unipi IT Security Lab Test CA (Εικόνα 41) και έγινε εισαγωγή του στο σύστημα (Εικόνα 42).



Εικόνα 40. Εισαγωγή πιστοποιητικού Αρχής Πιστοποίησης στα έμπιστα πιστοποιητικά



Εικόνα 41.Πιστοποιητικό server (localhost) υπογεγραμμένο από την Αρχή Πιστοποίησης



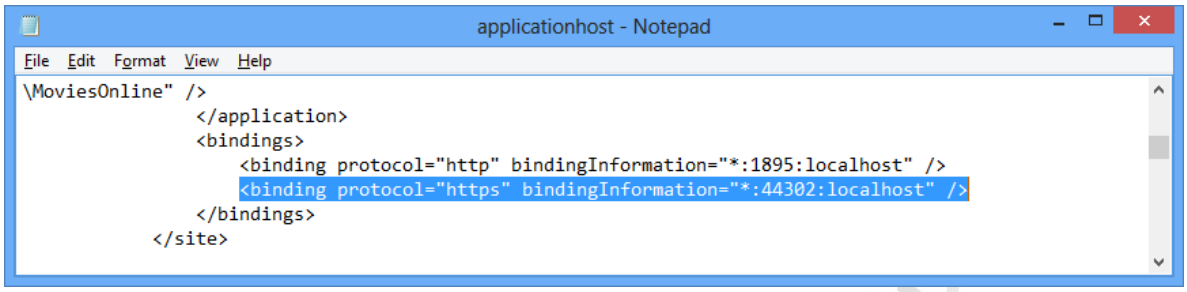
Εικόνα 42. Εγκατάσταση πιστοποιητικού server (localhost) στο σύστημα

7.1.2 Ρύθμιση για χρήση IIS Express Server στη θύρα 44302.

Εκτελέστηκε εντολή για url reservation στη θύρα 44302 από διεπαφή γραμμής εντολών με δικαιώματα administrator (Εικόνα 43).

Εικόνα 43.Url reservation στη θύρα 44302

Έγινε ρύθμιση για τον IIS Express μέσω του αρχείου applicationHost.config στο στοιχείο binding έτσι ώστε να τρέχει την εφαρμογή όταν γίνει αίτημα σε αυτή τη θύρα(Εικόνα 44).



Εικόνα 44. Ρυθμίσεις binding για τον IIS Express

Τέλος, χρησιμοποιήθηκε η εντολή :

```
netsh firewall add portopening TCP 44302 IISExpressWeb enable ALL
```

σε γραμμή εντολών, έτσι ώστε να επιτρέπεται από το firewall του λειτουργικού συστήματος οποιαδήποτε επικοινωνία με τον IIS Express στην θύρα 44302.

7.1.3 Εγκατάσταση πιστοποιητικού στον server

Για να εγκαταστήσουμε το πιστοποιητικό στο server εκτελέσαμε την παρακάτω εντολή σε κονσόλα:

netsh http add sslcert ipport=0.0.0.0:44302 appid={214124cd-d05b-4309-9af9-9caa44b2b74a} certhash=fe910a5bf28242704c1918f9d1ddf4bd6aabe42a

Για την τιμή certhash χρησιμοποιήθηκε το αποτύπωμα (thumbprint) του πιστοποιητικού του server που δημιουργήθηκε στην Εικόνα 41.

7.1.4 Ρυθμίσεις στον κώδικα της εφαρμογής για χρήση SSL

Σε όλες τις κλάσεις Controllers της εφαρμογής, εφαρμόστηκε μαρκάρισμα με την κλάση RequireHttpsAttribute, όπως φαίνεται παρακάτω:

```
[RequireHttps]
    public class SGAccountController : BaseController
    {  . . .  }
```

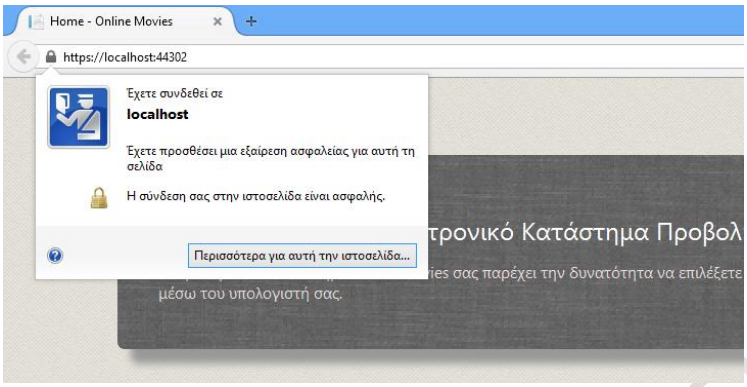
Αυτό σημαίνει ότι οποτεδήποτε ζητηθεί μια ιστοσελίδα, αν ο Controller που τη χειρίζεται χαρακτηρίζεται από το [RequireHttps] τότε το αίτημα για την ιστοσελίδα και η απάντηση από το server θα πρέπει να αποσταλούν χρησιμοποιώντας πρωτόκολλο SSL.

Οι κλάσεις της εφαρμογής που έχουν χαρακτηριστεί με [RequireHttps] είναι οι:

- SGAccountController
- HomeController
- MoviesController
- DashboardController
- MembershipController
- RoleController

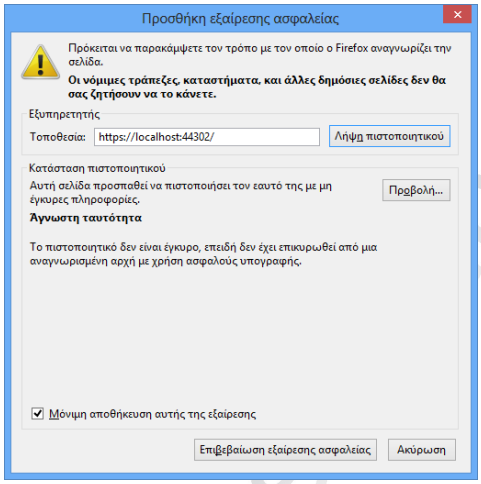
7.1.5 Δοκιμή λειτουργίας εφαρμογής με SSL και χρήση πιστοποιητικών

Μετά από τις προηγούμενες ρυθμίσεις, κατά την εκτέλεση της εφαρμογής παρατηρείται ότι χρησιμοποιείται το πρωτόκολλο SSL για την επικοινωνία του client με τον server και χρησιμοποιείται το πιστοποιητικό που δημιουργήθηκε στο προηγούμενο βήμα.



Εικόνα 45. Επιβεβαίωση λειτουργίας εφαρμογής με SSL

Όπως φαίνεται από την Εικόνα 45, προστέθηκε εξαίρεση ασφαλείας για την εφαρμογή γιατί η Αρχή Πιστοποίησης δεν είναι αναγνωρισμένη Αρχή με χρήση ασφαλούς υπογραφής αφού δημιουργήθηκε για τους σκοπούς της εργασίας.



Εικόνα 46. Προσθήκη εξαίρεσης ασφαλείας στον Mozilla Firefox για την εφαρμογή

7.2 Διαχείριση χρηστών, ρόλων και αυθεντικοποίηση

Για τη διαχείριση των χρηστών και των ρόλων της εφαρμογής χρησιμοποιήθηκε το πακέτο εφαρμογής Security Guard από το [35]. Το συγκεκριμένο πακέτο χρησιμοποιεί τις παρεχόμενες βιβλιοθήκες από το ASP.NET Membership, Roles, Forms Authentication και προσφέρει τον αντίστοιχο μηχανισμό αυθεντικοποίησης (Forms Authentication) που προσφέρει και το πρότυπο δημιουργίας εφαρμογής Internet που παρέχει το VS12.

Οι κύριες λειτουργίες που παρέχονται από το πακέτο είναι:

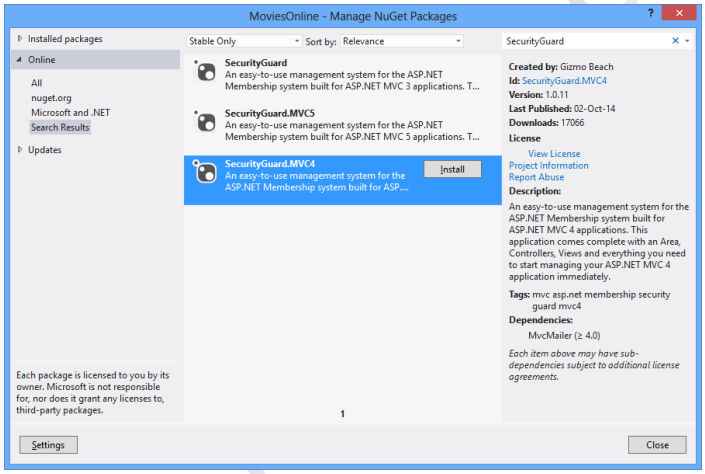
- Login (Σύνδεση)
- LogOff (Αποσύνδεση)

- Register (Εγγραφή)
- Change Password (Αλλαγή κωδικού πρόσβασης)
- Forgot Password (Υπενθύμιση κωδικού πρόσβασης)

Προσφέρει επίσης γραφικό περιβάλλον διεπαφής για τις παρακάτω λειτουργίες διαχείρισης χρηστών και ρόλων:

- Δημιουργία χρηστών
- Προβολή χρηστών ανάλογα με το όνομα χρήστη
- Προβολή χρηστών ανάλογα με το email χρήστη
- Διόρθωση στοιχείων χρήστη
- Ξεκλείδωμα λογαριασμού χρήστη
- Διαγραφή λογαριασμού χρήστη
- Προβολή ρόλων που έχουν αποδοθεί σε κάποιον χρήστη
- Απόδοση ρόλων σε κάποιο χρήστη
- Ανάκληση ρόλων από χρήστη
- Δημιουργία νέων ρόλων
- Διαγραφή ρόλου

Για την εγκατάσταση του πακέτου στο project εκτελέστηκε το εργαλείο διαχείρισης πακέτων NuGet που παρέχει το VS12 (NuGet Package Manager). Έγινε αναζήτηση με τον όρο “Security Guard” και στη συνέχεια έγινε η εγκατάστασή του στο project.



Εικόνα 47. Εύρεση και εγκατάσταση πακέτου Security Guard μέσω VS12

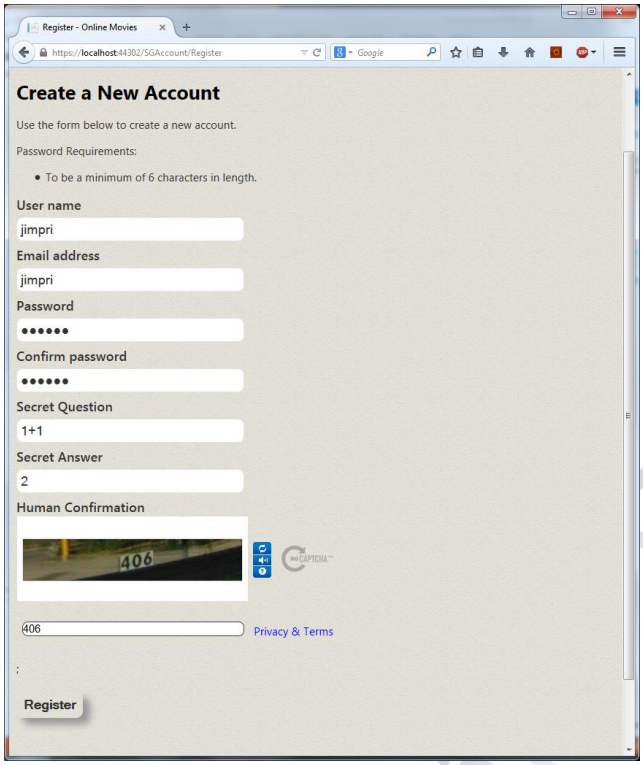
7.2.1 Δημιουργία χρήστη Administrator

Για να μπορέσει κάποιος χρήστης να χρησιμοποιήσει το γραφικό περιβάλλον διεπαφής του SecurityGuard θα πρέπει να έχει ρόλο Administrator. Έτσι, Για τη διαχείριση της βάσης δεδομένων των χρηστών και των ρόλων, δημιουργήθηκε ένας χρήστης με τα παρακάτω στοιχεία :

- UserName : jimpri
- password : 123456.

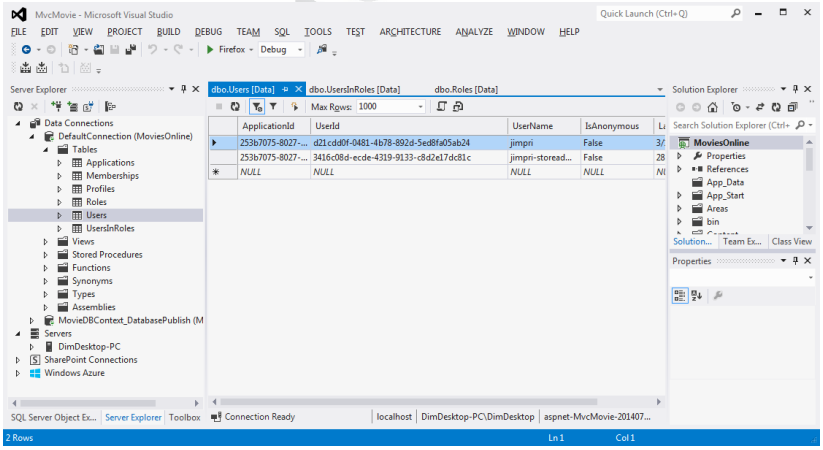
Επίσης δημιουργήθηκε και ένας ρόλος με όνομα Administrator και αποδόθηκε στον χρήστη jimpri έτσι ώστε αυτός να μπορεί να εκτελέσει τις λειτουργίες που παρέχονται από το SecurityGuard.

Για την υλοποίηση των παραπάνω, εκτελέστηκε πρώτα η εφαρμογή και έγινε εγγραφή του χρήστη jimpri χρησιμοποιώντας τη διαδικασία εγγραφής μέσω της ιστοσελίδας /SGAccount/Register όπως φαίνεται στην Εικόνα 48.



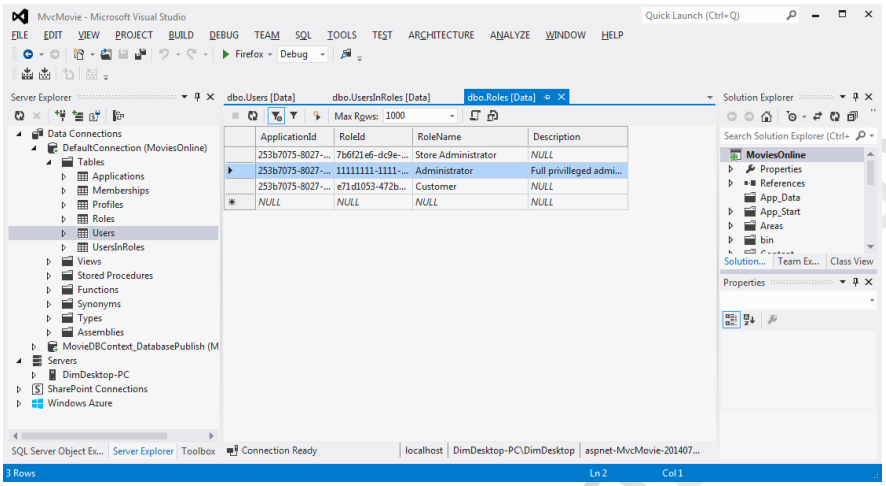
Εικόνα 48. Δημιουργία χρήστη με όνομα jimpri στην εφαρμογή Online Movies

Στη συνέχεια χρησιμοποιήθηκε ο Server Explorer του VS12 για την επιβεβαίωση εγγραφής του χρήστη στη βάση δεδομένων και για την επεξεργασία της βάσης δεδομένων που περιέχει τους χρήστες και τους ρόλους.



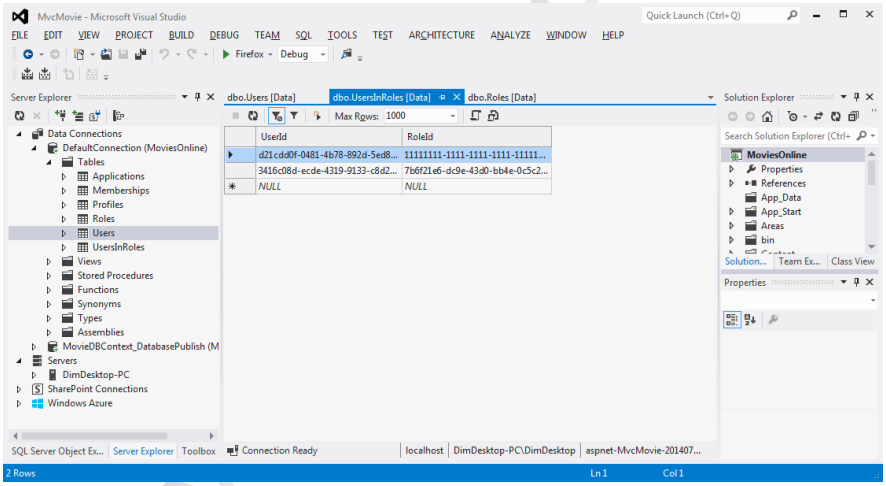
Εικόνα 49. Επιβεβαίωση εγγραφής χρήστη jimpri στη βάση δεδομένων χρηστών - ρόλων

Δημιουργήθηκε ο ρόλος Administrator στον πίνακα Roles με απευθείας πρόσθεση νέας εγγραφής στον πίνακα Roles μέσω του VS12 (Εικόνα 50).



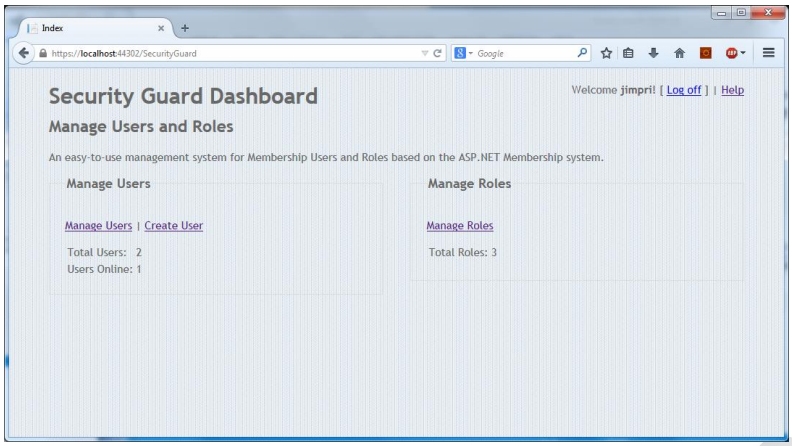
Εικόνα 50. Δημιουργία ρόλου Administrator στη βάση δεδομένων χρηστών και ρόλων

Ο ρόλος Administrator ανατέθηκε στον jimpri δημιουργώντας αντίστοιχη εγγραφή στον πίνακα UsersInRoles (Εικόνα 51).



Εικόνα 51. Ανάθεση ρόλου Administrator στον χρήστη jimpri

Με αυτόν τον τρόπο υλοποιήθηκε η δυνατότητα του jimpri να μπορεί να χρησιμοποιήσει το γραφικό περιβάλλον που παρέχει το πακέτο SecurityGuard για τη διαχείριση των χρηστών και των ρόλων της εφαρμογής. Κάνοντας Login με τα στοιχεία του χρήστη jimpri και επιλέγοντας τον υπερσύνδεσμο Security Guard από το μενού της εφαρμογής, εμφανίζεται η παρακάτω εικόνα για έλεγχο χρηστών και ρόλων:



Εικόνα 52. Γραφικό περιβάλλον διεπαφής για διαχείριση χρηστών και ρόλων

7.3 Επικύρωση δεδομένων εισόδου

Για να γίνεται επικύρωση των δεδομένων που εισάγουν οι χρήστες πριν αυτά εισαχθούν στις βάσεις δεδομένων της εφαρμογής, χρησιμοποιήθηκε η κλάση System.ComponentModel.DataAnnotations. Στα αρχεία που περιγράφουν τα μοντέλα της εφαρμογής προστέθηκε η παρακάτω δήλωση στην αρχή του κώδικα :

```
using System.ComponentModel.DataAnnotations;
```

Τα παρακάτω χαρακτηριστικά της κλάσης DataAnnotations (Required, DataType, Range, StringLength) χρησιμοποιήθηκαν για τα πεδία του πίνακα Movies της βάσης δεδομένων που περιγράφονται από το μοντέλο:

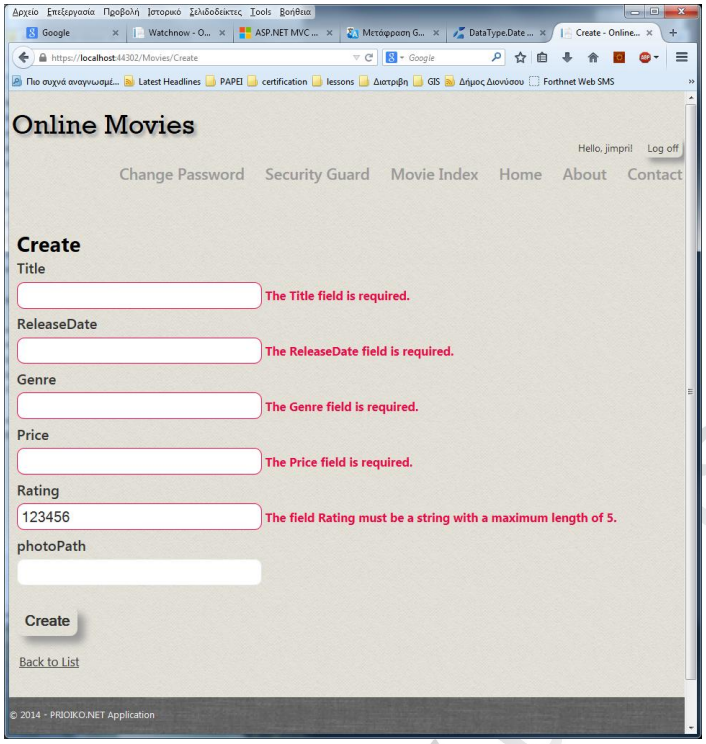
```
        [Required]
        public string Title { get; set; }

        [DataType(DataType.Date)]
        public DateTime ReleaseDate { get; set; }

        [Required]
        public string Genre { get; set; }

        [Range(1, 100)]
        [DataType(DataType.Currency)]
        public decimal Price { get; set; }

        [StringLength(5)]
        public string Rating { get; set; }
```

Εικόνα 53. Data Annotations στο αρχείο Movie.cs

Σύμφωνα με τον παραπάνω κώδικα, σε περίπτωση που κάποιος χρήστης προσπαθήσει να δημιουργήσει μια ταινία χωρίς να δώσει κάποια στοιχεία ή δώσει λανθασμένα στοιχεία στα πεδία φόρμας, θα εμφανιστούν αντίστοιχα μηνύματα σφάλματος και δεν θα εγγραφούν δεδομένα στη βάση.

Στην Εικόνα 54 φαίνεται ένα παράδειγμα στο οποίο έγινε προσπάθεια να εισαχθεί μια κενή εγγραφή ταινίας στη βάση δεδομένων. Δίπλα από κάθε πεδίο εμφανίστηκε το αντίστοιχο μήνυμα

σφάλματος με κόκκινα γράμματα όταν επιλέχθηκε Create στη φόρμα. Σημειώνεται ότι η λειτουργία επικύρωσης δεδομένων είσόδου εκτελείται στην πλευρά του client αλλά και του server, αφού έχουν εφαρμοσθεί οι ρυθμίσεις που αναφέρθηκαν στην παράγραφο 4.8.



Εικόνα 54. Παράδειγμα λανθασμένης εισαγωγής στοιχείων δημιουργίας ταινίας

Στο φάκελο που περιλαμβάνει τα μοντέλα της διαχείρισης χρηστών Areas\SecurityGuard\Models χρησιμοποιήθηκαν τα παρακάτω Data Annotations (Required, DataType, StringLength):

```
public class ForgotPasswordViewModel

    {
        [Required(ErrorMessage = "Email is required.")]
        public string Email { get; set; }

        [Display(Name = "Secret Question")]
        public string SecretQuestion { get; set; }

        [Display(Name = "Secret Answer")]
        public string SecretAnswer { get; set; }

        public bool Checked { get; set; }

        public bool RequireSecretQuestionAndAnswer { get; set; }
    }
```

Εικόνα 55.Data Annotations στο αρχείο ForgotPasswordViewModel.cs

```
public class ChangePasswordViewModel
    {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Current password")]
        public string OldPassword { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2}
characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
        [System.Web.Mvc.Compare("NewPassword", ErrorMessage = "The new
password and confirmation password do not match.")]
        public string ConfirmPassword { get; set; }
    }
```

Εικόνα 56.Data Annotations στο αρχείο ChangePasswordViewModel.cs

```
public class RegisterViewModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [DataType(DataType.EmailAddress)]
        [Display(Name = "Email address")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2}
characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [System.Web.Mvc.Compare("Password", ErrorMessage = "The password
and confirmation password do not match.")]
        public string ConfirmPassword { get; set; }

        [Display(Name = "Secret Question")]
        public string SecretQuestion { get; set; }

        [Display(Name = "Secret Answer")]
        public string SecretAnswer { get; set; }

        public bool Approve { get; set; }

        public bool RequireSecretQuestionAndAnswer { get; set; }
    }
```

Εικόνα 57. Data Annotations στο αρχείο RegisterViewModel.cs

```
public class RoleViewModel
    {
        [Required(ErrorMessage="RoleName is required.")]
        public string RoleName { get; set; }
        public string Description { get; set; }
    }
```

Εικόνα 58. Data Annotations στο αρχείο RoleViewModel.cs

7.4 Αυθεντικοποίηση

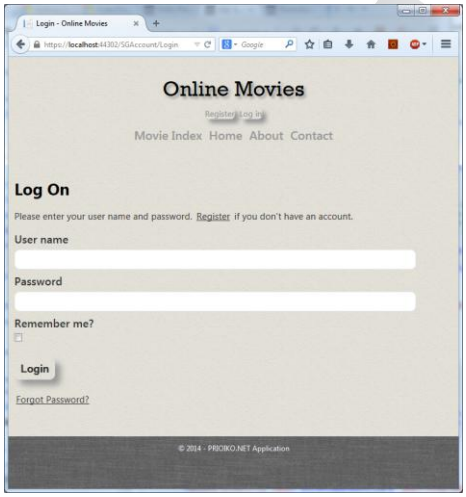
Ο μηχανισμός αυθεντικοποίησης χρηστών που χρησιμοποιείται για την εφαρμογή είναι ο Forms Authentication. Η ρύθμισή του φαίνεται στο αρχείο web.config της εφαρμογής στην Εικόνα 59.

```
<authentication mode="Forms">
      <forms loginUrl="~/SGAccount/Login"
             timeout="300"
             requireSSL="true"/>
    </authentication>
```

Εικόνα 59. Ρύθμιση μηχανισμού αυθεντικοποίησης εφαρμογής

Σύμφωνα με τις ρυθμίσεις, η φόρμα στην οποία θα ανακατευθυνθεί κάποιος χρήστης σε περίπτωση που απαιτείται αυθεντικοποίησή του είναι η SGAccount/Login (Εικόνα 60), ενώ για το εισιτήριο αυθεντικοποίησης έχει καθοριστεί χρόνος λήξης του 300 λεπτά. Μετά από αυτόν τον χρόνο θα πρέπει να γίνει πάλι αυθεντικοποίηση του χρήστη. Για την αποστολή του εισιτηρίου από τον browser του χρήστη στον server απαιτείται σύνδεση SSL.

7.4.1 Φόρμα σύνδεσης χρήστη (Login)



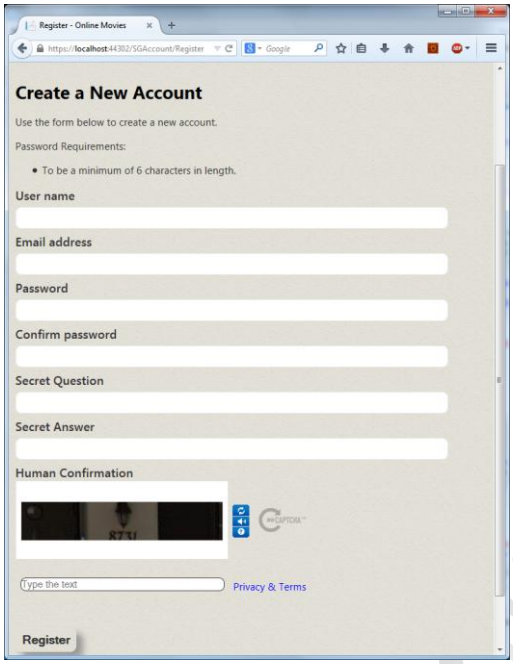
Εικόνα 60. Φόρμα αυθεντικοποίησης Login

Από τη φόρμα φαίνεται η επιλογή που έχει ο χρήστης κατά τη διαδικασία της αυθεντικοποίησης (επιλογή remember me) να διατηρήσει το εισιτήριο αυθεντικοποίησης στη μνήμη του browser (persistent cookie), έτσι ώστε ακόμα και αν αυτός κλείσει τον browser και τον ξανανοίξει, να μπορεί

να το χρησιμοποιήσει για να παραμείνει αυθεντικοποιημένος από την εφαρμογή χωρίς να χρειαστεί να εισάγει ξανά το όνομα χρήστη και τον κωδικό πρόσβασης.

7.4.2 Φόρμα εγγραφής χρήστη στην εφαρμογή (Register)

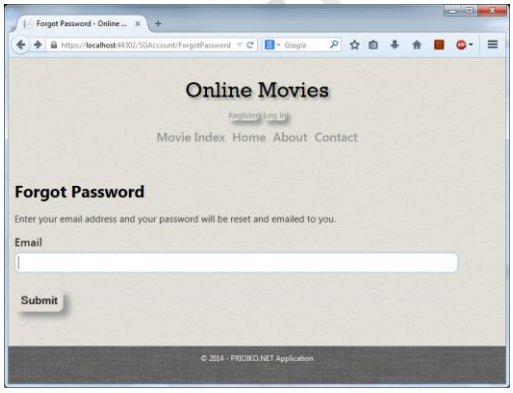
Για να εγγραφεί κάποιος χρήστης στην εφαρμογή και να μπορεί επομένως να αυθεντικοποιηθεί, υλοποιήθηκε η φόρμα που φαίνεται στην Εικόνα 61.



Εικόνα 61. Φόρμα εγγραφής Registration

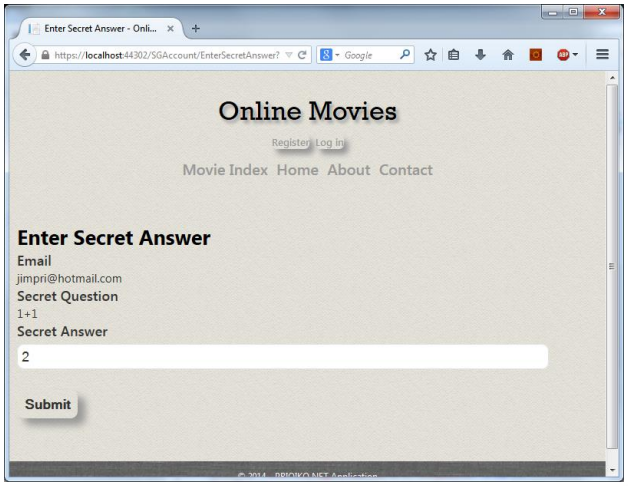
Ο κωδικός πρόσβασης που εισάγει ο χρήστης πρέπει να εισαχθεί δύο φορές στην φόρμα έτσι ώστε να αποκλειστεί το ενδεχόμενο λανθασμένης πληκτρολόγησης από τον χρήστη.

Τα πεδία μυστικής ερώτησης και απάντησης απαιτούνται για την περίπτωση που κάποιος χρήστης ξεχάσει τον κωδικό πρόσβασής του και επιθυμεί να αποκτήσει νέο. Αυτό που θα πρέπει να κάνει είναι να επιλέξει Forgot Password από τη φόρμα Login (Εικόνα 60) και να εισάγει την ηλεκτρονική του διεύθυνση στην ιστοσελίδα που θα εμφανιστεί (Εικόνα 62).



Εικόνα 62. Ιστοσελίδα υπενθύμισης κωδικού πρόσβασης

Στη συνέχεια η εφαρμογή θα ρωτήσει την μυστική ερώτηση που είχε καθορίσει αυτός κατά τη διαδικασία εγγραφής στην οποία θα πρέπει να απαντήσει με την μυστική απάντηση (Εικόνα 63) έτσι ώστε να αποσταλεί νέος κωδικός πρόσβασης.



Εικόνα 63. Ιστοσελίδα μυστικής ερώτησης και απάντησης

Οι ρυθμίσεις του server μέσω του οποίου γίνεται η αποστολή των μηνυμάτων νέων κωδικών πρόσβασης γίνονται στο αρχείο web.config της εφαρμογής και εντός του στοιχείου <system.net> όπως φαίνεται στην Εικόνα 64.

```
<mailSettings>
      <!-- Method#1: Configure smtp server credentials -->
      <smtp from="jimpri2008@hotmail.com">
        <network   enableSsl="true"
                   host="smtp.live.com"
                   port="587"
                   userName="jimpri2008@hotmail.com"
                   password="*********"/>
      </smtp>
    </mailSettings>
```

Εικόνα 64. Ρυθμίσεις σύνδεσης με server αποστολής email αλλαγής κωδικού πρόσβασης

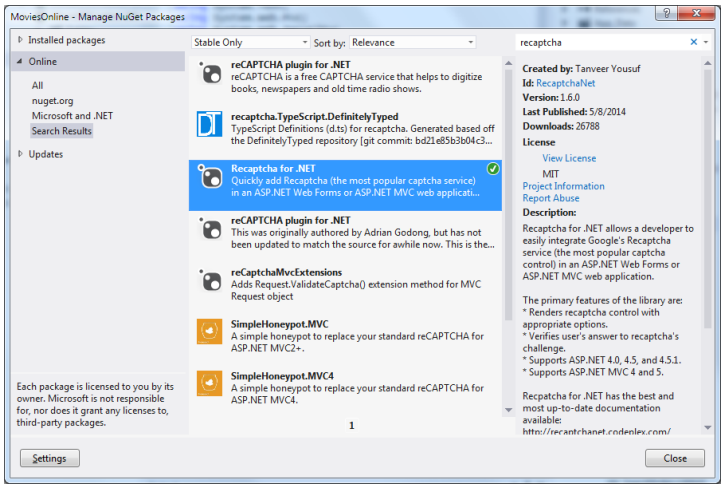
Για τις ανάγκες της εργασίας χρησιμοποιήθηκε η ηλεκτρονική διεύθυνση jimpri2008@hotmail.com μέσω της οποίας αποστέλλονταν τα συγκεκριμένα μηνύματα.

7.4.3 Έλεγχος εγγραφής χρηστών με CAPTCHA

Οποιαδήποτε εφαρμογή επιτρέπει σε ανθρώπους να εγγραφούν όπως η Online Movies είναι πιθανό να λάβει πλημμύρα από ψεύτικους λογαριασμούς χρηστών, οι οποίοι δημιουργούνται από αυτόματα προγράμματα (bots) [36].

Για να εξασφαλιστεί ότι οι λογαριασμοί χρηστών δεν δημιουργούνται από τέτοιου είδους προγράμματα χρησιμοποιούνται έλεγχοι CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart). Ο πιο συνηθισμένος έλεγχος CAPTCHA είναι αυτός που παρουσιάζονται στην οθόνη μερικά παραμορφωμένα γράμματα ή αριθμοί και ο χρήστης πρέπει να πληκτρολογήσει αυτό που βλέπει. Η παραμόρφωση αυτή δυσκολεύει τα bots στην αναγνώρισή τους, όχι όμως τους ανθρώπους.

Για την υλοποίηση τέτοιου είδους ελέγχου στην εφαρμογή έγινε εγκατάσταση της βιβλιοθήκης Recaptcha.Web [37] στο project, μέσω της λειτουργίας Manage NuGet Packages του VS12 όπως φαίνεται στην Εικόνα 65.



Εικόνα 65. Εγκατάσταση βιβλιοθήκης Recaptcha.Web

Για να εμφανιστεί ο έλεγχος CAPTCHA στην φόρμα Register, χρησιμοποιήθηκε η παρακάτω δήλωση στην αρχή του αρχείου Register.cshtml:

```
@using Recaptcha.Web.Mvc
```

Στο ίδιο αρχείο συμπληρώθηκε ο επισημασμένος με κίτρινο χρώμα κώδικας που φαίνεται στην Εικόνα 66.

```
<div class="editor-label">
        @Html.Label("Human Confirmation")
</div>
<div>
        @Html.Recaptcha(theme:Recaptcha.Web.RecaptchaTheme.Clean)
</div>
<p>
        <input type="submit" value="Register" />
</p>
```

Εικόνα 66. Κώδικας για εμφάνιση ελέγχου CAPTCHA στην ιστοσελίδα Register

Ο κώδικας μέσω του οποίου γίνεται ο έλεγχος των στοιχείων που εισήγαγε ο χρήστης με αυτά που εμφανίζονται στη φόρμα εκτελείται στον SGAccountController και συγκεκριμένα στη μέθοδο Register. Στην κλάση αυτή τοποθετήθηκε ο επισημασμένος με κίτρινο χρώμα κώδικας που φαίνεται στην Εικόνα 67.

```
public virtual ActionResult Register(viewModels.RegisterViewModel model)
        {
            if (ModelState.IsValid)
            {
                //captcha validation
                RecaptchaVerificationHelper recaptchaHelper =
                   this.GetRecaptchaVerificationHelper();

                if (String.IsNullOrEmpty(recaptchaHelper.Response))
                {
                    ModelState.AddModelError("", "Captcha answer cannot be
                    empty.");
                    return View(model);
                }

                //The 'GEM' Ignore validation callback which cause an error
                /* This gem is called ServerCertificateValidationCallBack.
                 * This CallBack will return false in case the Server
                   Certificate
                 * is not complying with the policies (in our case,
                 * it is by default not trusted as it is self-signed),
                 * so all we need to do is ensure that it always returns
                   true and thus
                 * ignoring the security check
                 */
                ServicePointManager.ServerCertificateValidationCallback +=
                (sender, certificate, chain, sslPolicyErrors) => true;

                RecaptchaVerificationResult recaptchaResult =
                 recaptchaHelper.VerifyRecaptchaResponse();
                 if (recaptchaResult !=
                 RecaptchaVerificationResult.Success)
                {
                    ModelState.AddModelError("", "Incorrect captcha
                     answer.");
                    return View(model);
                }

                    . . . . . .

            }

            return RedirectToAction("Register");
        }
```

Εικόνα 67. Κώδικας ελέγχου στοιχείων CAPTCHA

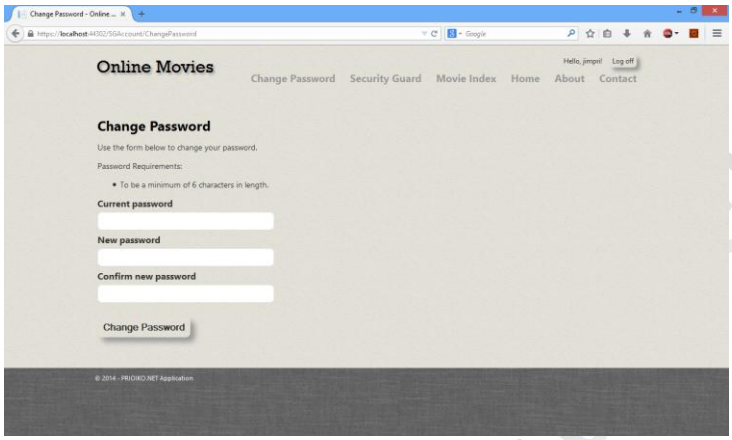
Για χρήση της συγκεκριμένης βιβλιοθήκης, απαιτείται η λήψη ζεύγους κλειδιών από την Google κατόπιν εγγραφής στην ιστοσελίδα. Τα κλειδιά αυτά τοποθετήθηκαν στο αρχείο διαμόρφωσης της εφαρμογής web.config, στο στοιχείο <appSettings>:

```
<add key="recaptchaPublicKey"
                    value="6LdCA_gSAAAAADXxDpMY_yioGc9Q6N-qUDnHWCXx"/>
<add key="recaptchaPrivateKey"
                    value="6LdCA_gSAAAAANUlunMUfLQZZDR7iJhnDOz2U6pS"/>
```

Εικόνα 68. Ρύθμιση για χρήση κλειδιών λειτουργίας ελέγχου CAPTCHA

7.4.4 Διαδικασία αλλαγής κωδικού πρόσβασης

Για τις περιπτώσεις που κάποιος χρήστης απαιτεί αλλαγή του κωδικού πρόσβασης για τον λογαριασμό του, υπάρχει η δυνατότητα να εκτελέσει την αλλαγή μέσω της ιστοσελίδας SGAccount/ChangePassword. Εφόσον κάποιος χρήστης έχει αυθεντικοποιηθεί, η δυνατότητα αυτή εμφανίζεται στο μενού της εφαρμογής. Επιλέγοντας Change Password από το μενού εμφανίζεται η ιστοσελίδα που φαίνεται στην Εικόνα 69. Ο χρήστης πληκτρολογεί τον παλιό κωδικό πρόσβασης (current) και δύο φορές τον καινούριο (new) για να συμβεί η αλλαγή. Ο περιορισμός για τουλάχιστον έξι γράμματα στους κωδικούς πρόσβασης ισχύει όπως και στη διαδικασία Registration λόγω των Data Annotations που περιγράφηκαν και φαίνονται στην Εικόνα 57.



Εικόνα 69.Ιστοσελίδα αλλαγής κωδικού πρόσβασης χρήστη

7.5 Εξουσιοδότηση

Για την εφαρμογή των απαιτήσεων ασφάλειας της εφαρμογής έχουν υλοποιηθεί κανόνες εξουσιοδότησης στα αρχεία-κλάσεις SGAccountController, MoviesController, MembershipController και RoleController σύμφωνα με την παράγραφο 4.7.2.

7.5.1 Εφαρμογή κανόνων στην κλάση SGAccountController

Στην κλάση SGAccountController έχουν μαρκαριστεί με το χαρακτηριστικό [Authorize] μόνο οι μέθοδοι ChangePassword, που χρησιμοποιούνται στην περίπτωση που κάποιος αυθεντικοποιημένος χρήστης επιθυμεί να αλλάξει κωδικό πρόσβασης στην εφαρμογή.

```
[Authorize]
public virtual ActionResult ChangePassword()
{    . . .    }

[Authorize]
[HttpPost]
[ValidateAntiForgeryToken()]
public virtual ActionResult ChangePassword(ChangePasswordViewModel model)
{
    . . .
}
```

Πίνακας 21. Ρυθμίσεις εξουσιοδότησης μεθόδων ChangePassword κλάσης SGAccountController

7.5.2 Εφαρμογή κανόνων στην κλάση MoviesController

Η κλάση MoviesController έχει μαρκαριστεί με το χαρακτηριστικό [Authorize(Roles=”Administrator, Store Administrator”)], που σημαίνει ότι όλες οι μέθοδοι που υπάρχουν σε αυτήν μπορούν να προσπελαστούν από χρήστες με τους δύο αυτούς ρόλους. Μέσω αυτής της κλάσης υπενθυμίζεται ότι εκτελούνται λειτουργίες εισαγωγής, διαγραφής και τροποποίησης ταινιών, λειτουργίες που πρέπει να μπορούν να εκτελούν μόνο συγκεκριμένοι χρήστες.

Η μέθοδος Watchnow έχει μαρκαριστεί με το χαρακτηριστικό [OverrideAuthorization], που σημαίνει ότι αναιρείται ο κανόνας εξουσιοδότησης για αυτήν, αλλά ισχύει κανόνας αυθεντικοποίησης. Οι χρήστες δηλαδή που είναι αυθεντικοποιημένοι (οποιουδήποτε ρόλου) μπορούν να παρακολουθήσουν την επιθυμητή ταινία.

Οι μέθοδοι SearchIndex και Details έχουν μαρκαριστεί με το χαρακτηριστικό [AllowAnonymous], που σημαίνει ότι οποιοσδήποτε χρήστης, είτε αυθεντικοποιημένος είτε όχι, μπορεί να δει ποιες ταινίες υπάρχουν στο ηλεκτρονικό κατάστημα.

```
    [RequireHttps]
    [Authorize(Roles = "Administrator, Store Administrator")]
    public class MoviesController : Controller
    {
        private MovieDBContext db = new MovieDBContext();

        // GET: /Movies/Watchnow
        [OverrideAuthorization]
        public ActionResult Watchnow(int id = 0)
        {       }


        // GET: /Movies/Details/5
        [AllowAnonymous]
        public ActionResult Details(int id = 0)
        {       }


        // GET: /Movies/SearchIndex/
        [AllowAnonymous]
        public ActionResult SearchIndex(string movieGenre, string
searchString)
        {       }
    }
```

Εικόνα 70. Ρυθμίσεις εξουσιοδότησης κλάσης MoviesController

7.5.3 Εφαρμογή κανόνων στην κλάση MembershipController

Η κλάση MembershipController έχει μαρκαριστεί με το χαρακτηριστικό [Authorize(Roles=”Administrator)], που σημαίνει ότι όλες οι μέθοδοι που υπάρχουν σε αυτήν μπορούν να προσπελαστούν από χρήστες με ρόλο Administrator.

```
    [Authorize(Roles = "Administrator")]
    [RequireHttps]
    public partial class MembershipController : BaseController
    {     …     }
```

Εικόνα 71. Ρυθμίσεις εξουσιοδότησης κλάσης MembershipController

Η κλάση αυτή χρησιμοποιείται για :

1. Προβολή όλων των χρηστών της εφαρμογής
2. Δημιουργία νέων χρηστών
3. Διαγραφή χρηστών

4. Ενημέρωση των στοιχείων χρηστών
5. Ξεκλείδωμα λογαριασμών χρηστών
6. Ανάθεση ρόλων σε χρήστες
7. Ανάκληση ρόλων από χρήστες

7.5.4 Εφαρμογή κανόνων στην κλάση RoleController

Η κλάση RoleController έχει μαρκαριστεί με το χαρακτηριστικό [Authorize(Roles="Administrator)], που σημαίνει ότι όλες οι μέθοδοι που υπάρχουν σε αυτήν μπορούν να προσπελαστούν από χρήστες με ρόλο Administrator.

```
[Authorize(Roles="Administrator")]
[RequireHttps]
public partial class RoleController : BaseController
{     …     }
```

Εικόνα 72. Ρυθμίσεις εξουσιοδότησης κλάσης RoleController

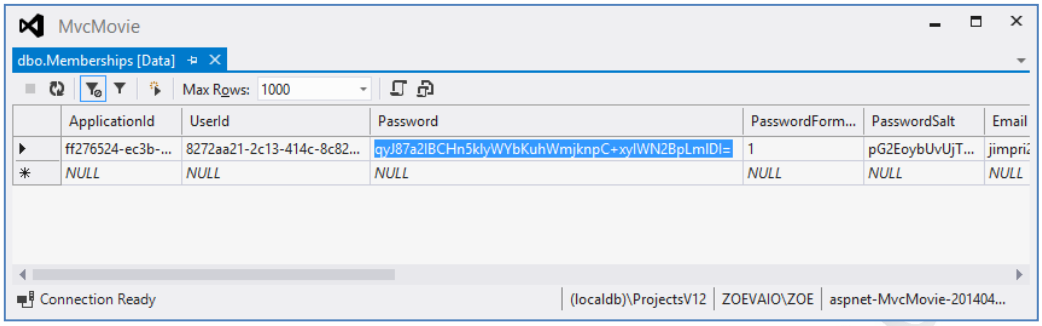
Η κλάση αυτή χρησιμοποιείται για :

1. Προβολή όλων των διαθέσιμων ρόλων της εφαρμογής
2. Δημιουργία νέων ρόλων
3. Διαγραφή ρόλων
4. Εύρεση χρηστών που ανήκουν σε συγκεκριμένους ρόλους

7.6 Ευαίσθητα δεδομένα

Τα ευαίσθητα δεδομένα της εφαρμογής είναι οι πληροφορίες χρηστών και ταινιών που αποθηκεύονται στις βάσεις δεδομένων που χρησιμοποιεί η εφαρμογή, τα connection strings με τις βάσεις δεδομένων που υπάρχουν στο αρχείο ρυθμίσεων web.config της εφαρμογής καθώς επίσης και τα cookies αυθεντικοποίησης και συνόδου.

- Για τα δεδομένα αυτά έχουν καθοριστεί οι παρακάτω κανόνες εξουσιοδότησης, όπως παρουσιάστηκαν στην παράγραφο 7.5:
- Για τις πληροφορίες των χρηστών, μόνο οι Administrators έχουν πρόσβαση, ενώ ο κάθε χρήστης έχει τη δυνατότητα να αλλάξει τον κωδικό του πρόσβασης.
- Τις πληροφορίες που αφορούν τις ταινίες, μόνο οι Administrators και οι Store Administrators μπορούν να τις τροποποιήσουν.
- Οι κωδικοί πρόσβασης που καθορίζει ο κάθε χρήστης για το λογαριασμό του δεν αποθηκεύονται στη βάση δεδομένων με τη μορφή που τους εισάγει αυτός στις φόρμες της εφαρμογής. Εφαρμόζεται πρώτα συνάρτηση κατακερματισμού σε αυτούς και στη συνέχεια αποθηκεύεται η τιμή που προκύπτει από τη συνάρτηση κατακερματισμού. Με αυτόν τον τρόπο, ο μόνος που γνωρίζει τον κωδικό πρόσβασης είναι ο ίδιος ο χρήστης και κανένας άλλος (Εικόνα 73).



Εικόνα 73. Αποθήκευση τιμής κατακερματισμού κωδικού πρόσβασης στη βάση δεδομένων χρηστών

Επιπλέον, κατά τη μετάδοση των δεδομένων, έχει υλοποιηθεί ο μηχανισμός κρυπτογράφησης SSL που αναλύθηκε στην παράγραφο 7.8. Με αυτόν το μηχανισμό επικοινωνίας μεταξύ server και client, τα ευαίσθητα αυτά δεδομένα προστατεύονται από υποκλοπή, αλλοίωση και τροποποίηση.

7.7 Διαχείριση συνόδου

Για τη διαχείριση της κατάστασης συνόδου χρησιμοποιείται η προκαθορισμένη ρύθμιση του ASP.NET, “In-Process”, όπως φαίνεται από τις ρυθμίσεις στο αρχείο web.config, στο τμήμα <system.web> και στο χαρακτηριστικό sessionState.

```
<sessionState mode="InProc" customProvider="DefaultSessionProvider">
```

Σε αυτή τη λειτουργία διαχείρισης κατάστασης συνόδου, τα δεδομένα της συνόδου αποθηκεύονται σε μεταβλητές οι οποίες διατηρούνται στη μνήμη του server για το χρονικό διάστημα που έχει καθοριστεί από το χαρακτηριστικό timeout του στοιχείου <authentication>, δηλαδή 300 λεπτά για τη συγκεκριμένη εφαρμογή (Εικόνα 59) ή εωσότου τερματιστεί η σύνοδος με επιλογή Logout από την εφαρμογή.

Τα δεδομένα αυθεντικοποίησης των χρηστών αποθηκεύονται σε cookies όπως αναφέρθηκε στην παράγραφο 4.6 και για να αποσταλούν στον server απαιτείται χρήση πρωτοκόλλου επικοινωνίας SSL, όπως φαίνεται και από τις ρυθμίσεις της εφαρμογής στην Εικόνα 59.

Για να αποφευχθούν επιθέσεις ενδιαμέσου, υφαρπαγής συνόδου και επαναληπτικής εκτέλεσης συνόδου έχει υλοποιηθεί μηχανισμός επαλήθευσης κλείδας (token verification). Για την υλοποίηση του μηχανισμού, χρησιμοποιήθηκε η μέθοδος AntiforgeryToken() της κλάσης HtmlHelpers του χώρου ονομάτων System.Web.Mvc και η κλάση ValidateAntiforgeryTokenAttribute.

Στα αρχεία που αφορούν τις όψεις της εφαρμογής και περιλαμβάνουν φόρμες εισαγωγής στοιχείων από τους χρήστες, έχει χρησιμοποιηθεί η μέθοδος AntiforgeryToken() σύμφωνα με το παράδειγμα της Εικόνα 74, ενώ η μέθοδος του Controller που αντιστοιχεί στη συγκεκριμένη όψη έχει μαρκαριστεί από την κλάση ValidateAntiforgeryTokenAttribute όπως φαίνεται για παράδειγμα στην Εικόνα 75.

```
@using (Html.BeginForm((string)ViewBag.FormAction, "SGAccount"))
{
    @Html.AntiForgeryToken()

    @Html.ValidationSummary(true)
    <div>
        <fieldset>
            . . .
        </fieldset>
    </div>
}
```

Εικόνα 74. Χρήση μεθόδου AntiforgeryToken() στην όψη Register.cshtml

```
        [HttpPost]
        [ValidateAntiForgeryToken()]
        public virtual ActionResult Register(viewModels.RegisterViewModel model)
        {     . . .  }
```

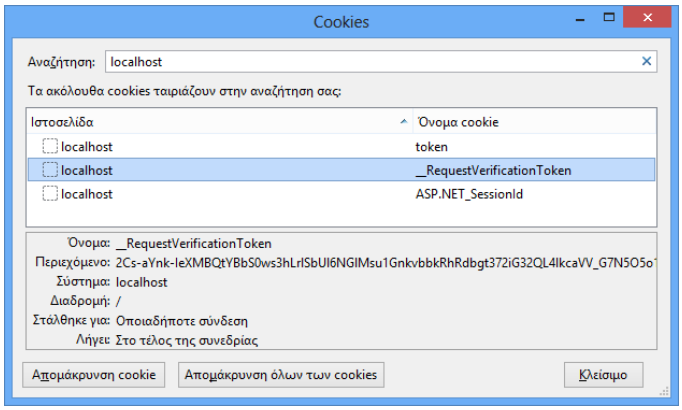
Εικόνα 75. Χρήση χαρακτηριστικού ValidateAntiForgeryToken στη μέθοδο Register του SGAccountController

Με αυτόν τον τρόπο η αποστολή προς τον client αυτών των ιστοσελίδων περιλαμβάνουν μια κρυπτογραφημένη τιμή ως κρυφό πεδίο με όνομα RequestVerificationToken εντός της φόρμας. Παράδειγμα του κώδικα html που αποστέλλεται όταν αιτηθεί η ιστοσελίδα SGAccount/Register φαίνεται στην Εικόνα 76.

```
<form action="/SGAccount/Register" method="post"><input
name="__RequestVerificationToken" type="hidden"
value="Ldvh6L1rhsPyuhFHmgIgaodw_Es2D6O5w-
7RBZW4nT7cUPEk6l1Dt1CbOGmhVVKK7hgugzTgc9EWpFSjybPA0ijuZfDDcrVR8v6r_UUuP
sc1" />    <div>
        <fieldset>
                . . .
        </fieldset>
    </div>
</form>
```

Εικόνα 76. Πηγαίος κώδικας ιστοσελίδας SGAccount/Register

Παράλληλα στέλνεται και cookie στον browser του χρήστη με το ίδιο όνομα, _RequestVerificationToken, το οποίο περιλαμβάνει επίσης μια τιμή, όπως φαίνεται στην Εικόνα 77.



Εικόνα 77. Παράδειγμα _RequestVerificationToken cookie

Όταν η φόρμα αποσταλεί στον server, ο αντίστοιχος Controller θα ελέγξει τις δύο αυτές τιμές και πρέπει να ταιριάζουν για να συνεχιστεί η εκτέλεση της απαιτούμενης λειτουργίας. Αν δεν ταιριάζουν, η σύνοδος δεν θα συνεχιστεί.

7.8 Κρυπτογραφία

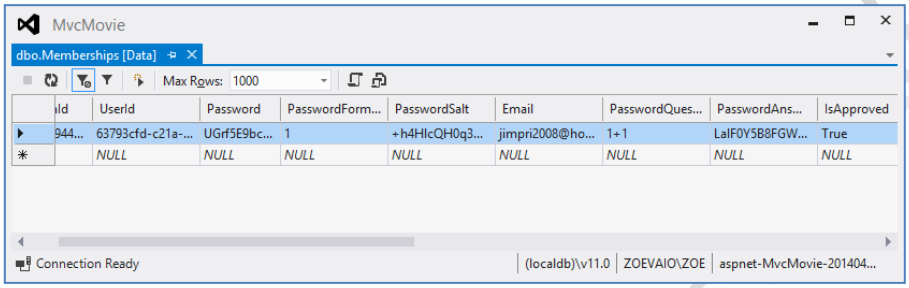
Ενεργοποιήθηκε ο μηχανισμός ασύμμετρης κρυπτογράφησης που χρησιμοποιείται από το πρωτόκολλο SSL, όπως αναφέρθηκε στην παράγραφο 7.1. Επιπλέον, οι κωδικοί πρόσβασης των χρηστών της εφαρμογής, αποθηκεύονται στη βάση δεδομένων αφού εφαρμοστούν πάνω τους μηχανισμοί κατακερματισμού που δημιουργούν μη αναστρέψιμες τιμές.

Μέσω της κλάσης SqlMembershipProvider που διαχειρίζεται τη βάση δεδομένων των χρηστών, έχει γίνει ρύθμιση για αποθήκευση των κωδικών πρόσβασης υπό μορφή “Hashed”. Η ρύθμιση αυτή φαίνεται στο αρχείο ρυθμίσεων machine.config του συστήματος (Εικόνα 78).

```
<membership>
            <providers>
                    <add name="AspNetSqlMembershipProvider"
                    type="System.Web.Security.SqlMembershipProvider, System.Web,
                    Version=4.0.0.0, Culture=neutral,
                    PublicKeyToken=b03f5f7f11d50a3a"
                    connectionStringName="LocalSqlServer"
                    enablePasswordRetrieval="false"
                    enablePasswordReset="true"
                    requiresQuestionAndAnswer="true"
                    applicationName="/"
                    requiresUniqueEmail="false"
                    passwordFormat="Hashed"
                    maxInvalidPasswordAttempts="5"
                    minRequiredPasswordLength="7"
                    minRequiredNonalphanumericCharacters="1"
                    passwordAttemptWindow="10"
                    passwordStrengthRegularExpression=""/>
            </providers>
</membership>
```

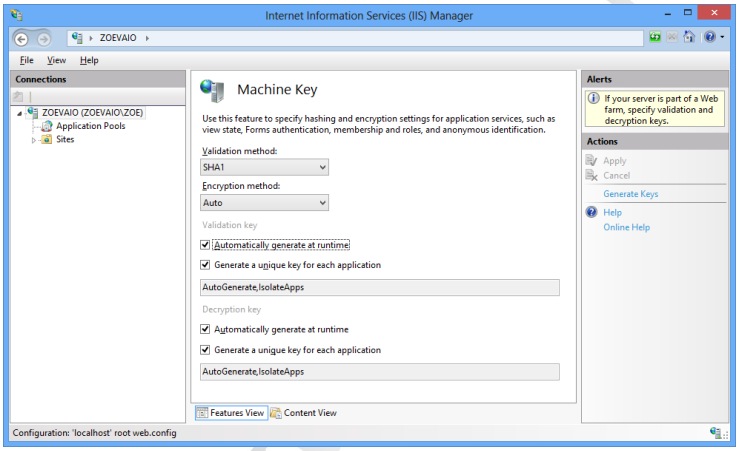
Εικόνα 78. Ρύθμιση SqlMemberhipProvider στο αρχείο machine.config

Με αυτήν την ρύθμιση, στα πεδία Password και PasswordAnswer του πίνακα Membership αποθηκεύονται τιμές κατακερματισμού που έχουν προκύψει από συνδυασμό του κωδικού πρόσβασης του χρήστη και μιας τυχαίας τιμής salt που δημιουργείται αυτόματα από το .NET Framework. Από κατακερματισμό του ζευγαριού τιμών κωδικού πρόσβασης/απάντηση στη μυστική ερώτηση που δίνει ο χρήστης και αυτήν την τιμή salt, δημιουργείται η τιμή PasswordSalt και αποθηκεύεται στον πίνακα Membership στο ομώνυμο πεδίο. Επίσης, από κατακερματισμό του κωδικού πρόσβασης που δίνει ο χρήστης σε συνδυασμό με την τιμή salt, δημιουργείται η τιμή Password και με αντίστοιχο τρόπο η τιμή PasswordAnswer (Εικόνα 79).



Εικόνα 79. Πίνακας Membership της βάσης δεδομένων χρηστών

Σημειώνεται ότι ο προκαθορισμένος αλγόριθμος κατακερματισμού που χρησιμοποιείται είναι ο SHA1, κατόπιν ρύθμισής του στο αρχείο machine.config του IIS server.

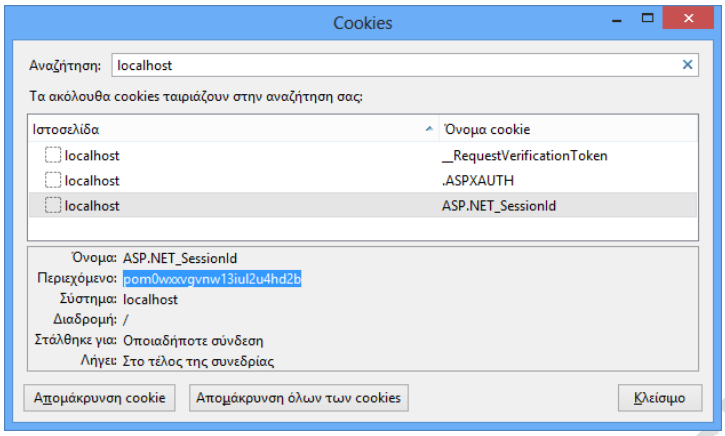


Εικόνα 80. Ρύθμιση αλγορίθμου κρυπτογράφησης SHA1 στον IIS

7.9 Χειρισμός παραμέτρων

Η εφαρμογή δεν χρησιμοποιεί την παράμετρο query string για την αποστολή ευαίσθητων δεδομένων ή δεδομένων που μπορούν να επηρεάσουν τη λογική ασφαλείας του server. Χρησιμοποιεί cookie αναγνώρισης συνόδου για την ταυτοποίηση του client και την αποθήκευση των ευαίσθητων δεδομένων του χρήστη που αποστέλλονται με το cookie αυθεντικοποίησης. Η λειτουργία αυτή καθορίζεται από τη ρύθμιση του στοιχείου <sessionState> στο αρχείο web.config της εφαρμογής όπως αναφέρθηκε στην παράγραφο 7.7. Από τη στιγμή δηλαδή που κάποιος χρήστης κάνει Login, εκδίδεται κωδικός συνόδου για αυτόν και σε κάθε μετέπειτα αίτημά του προς τον server, αποστέλλει cookie με όνομα ASP.NET_SessionId. Το cookie αυτό χρησιμοποιείται από τη μεριά του server για να διατηρηθεί η κατάσταση συνόδου που σχετίζεται με τον εκάστοτε χρήστη, για κάθε αίτημά του προς τον server. Στην Εικόνα 81 φαίνεται το ASP.NET_SessionId cookie που έχει εκδοθεί για τον χρήστη jimpri μετά την εκτέλεση Login στην εφαρμογή. Επιπλέον έχει υλοποιηθεί και η ρύθμιση httpOnlyCookies που περιγράφεται στην παράγραφο 4.13.

```
<httpCookies httpOnlyCookies="true" requireSSL="true"/>
```



Εικόνα 81. ASP.NET_SessionId cookie του χρήστη jimpri

Επιπλέον, για κάθε φόρμα της εφαρμογής μέσω της οποίας αποστέλλονται δεδομένα στον server χρησιμοποιείται μέθοδος HTTP POST. Για την υλοποίηση αυτού του περιορισμού έγινε μαρκάρισμα των μεθόδων των κλάσεων Controller της εφαρμογής που διαχειρίζονται φόρμες αποστολής δεδομένων με το χαρακτηριστικό [HttpPost].

7.10 Διαχείριση εξαιρέσεων

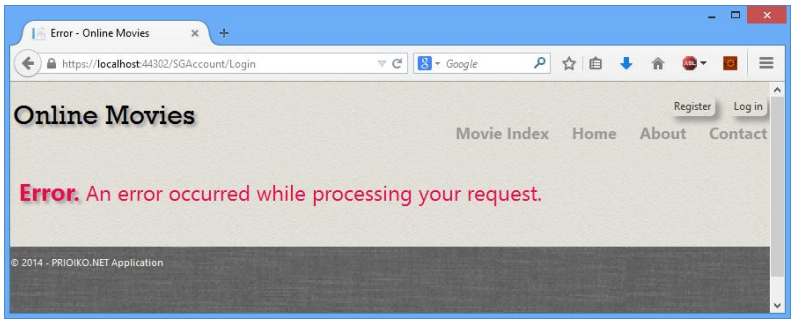
Για την αποφυγή αποκάλυψης πληροφοριών σχετικά με τον τρόπο υλοποίησης της εφαρμογής εφαρμόστηκε ρύθμιση στο αρχείο web.config της εφαρμογής έτσι ώστε σε περίπτωση που συμβεί κάποιο σφάλμα σε αυτή, να μην γνωστοποιείται το είδος του. Συγκεκριμένα χρησιμοποιήθηκε το στοιχείο <customErrors> εντός του στοιχείου <system.web> για την ενεργοποίηση λειτουργίας κατά την οποία σε περίπτωση λάθους, να επιστρέφεται στον χρήστη ένα φιλικό μήνυμα χωρίς να αποκαλύπτονται λεπτομέρειες στην εφαρμογή. Η ρύθμιση είναι της μορφής :

```
<customErrors mode="On"></customErrors>
```

Επιπλέον, για όλη την εφαρμογή έχει καθοριστεί ότι σε περίπτωση σφάλματος εξαίρεσης, θα παρουσιάζεται στον χρήστη η προκαθορισμένη όψη Error.cshtml που βρίσκεται στον κατάλογο /Views/Shared. Αυτό πραγματοποιήθηκε προσθέτοντας στο αρχείο FilterConfig.cs που βρίσκεται εντός του φακέλου App_Start η δήλωση :

```
filters.Add(new HandleErrorAttribute());
```

Αυτό που συμβαίνει δηλαδή όταν ένας χρήστης αποστέλλει ένα αίτημα στον server για το οποίο δημιουργείται σφάλμα εξαίρεσης, είναι ότι το ASP.NET MVC ψάχνει το αρχείο Error.cshtml στον τρέχοντα κατάλογο που επεξεργάζεται το αίτημα, για να απαντήσει στο αίτημα. Αν δεν το βρει, ψάχνει εν συνεχεία στον κατάλογο Views/Shared, οπότε και επιστρέφει την αντίστοιχη ιστοσελίδα (Εικόνα 82).



Εικόνα 82. Ιστοσελίδα σφάλματος εξαίρεσης

7.11 Παρακολούθηση και καταγραφή ιστορικού

Για την καταγραφή του ιστορικού αιτημάτων και σφαλμάτων της εφαρμογής εφαρμόσθηκαν οι ρυθμίσεις για tracing σε επίπεδο εφαρμογής που αναλύθηκαν στην παράγραφο 4.15. Στην Εικόνα 83 φαίνεται η καταγραφή κάποιων αιτημάτων προς την εφαρμογή.

Εικόνα 83. Στιγμιότυπο καταγραφής ιστορικού αιτημάτων προς την εφαρμογή

Κεφάλαιο 8°

8 Εγκατάσταση εφαρμογής στον IIS server

Στο κεφάλαιο αυτό περιγράφεται η διαδικασία που ακολουθήθηκε για την εγκατάσταση της εφαρμογής στον IIS server στον τοπικό υπολογιστή.

Κατά την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε ο IIS Express server μέσω του Visual Studio, ο οποίος συμπεριφέρεται παρόμοια αλλά όχι ακριβώς όπως ο IIS. Είναι πολύ πιθανό λοιπόν, η εφαρμογή να λειτουργεί σωστά όταν αναπτύσσεται και δοκιμάζεται μέσω του Visual Studio ενώ να παρουσιάζει προβλήματα όταν εγκατασταθεί στον IIS. Γι' αυτό το λόγο πρέπει να δοκιμαστεί σε περιβάλλον ανάπτυξης που εξομοιώνει περιβάλλον κανονικής λειτουργίας της εφαρμογής.

Στις παρακάτω παραγράφους αναλύονται οι διαδικασίες που εκτελέστηκαν για να εγκατασταθεί η εφαρμογή στον IIS, όπως ακριβώς θα γίνουν σε περίπτωση πραγματικής εγκατάστασης σε περιβάλλον λειτουργίας. Ακολουθώντας αυτήν την μέθοδο εξασφαλίζεται η ορθότητα τόσο των διαδικασιών εγκατάστασης όσο και της σωστής λειτουργίας της εφαρμογής.

8.1 Τροποποίηση εγκατάστασης βάσεων δεδομένων στον SQL Server Express

Κατά την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε ο Microsoft SQL Server Express LocalDb, ο οποίος δεν μπορεί να λειτουργήσει με τον IIS. Επομένως, οι βάσεις δεδομένων της εφαρμογής εγκαταστάθηκαν στον Microsoft SQL Server Express αλλάζοντας τα connection string στο αρχείο web.config της εφαρμογής έτσι ώστε να χρησιμοποιούνται αυτά που αναφέρονται στις βάσεις δεδομένων στον SQLEXPRESS και όχι στον LocalDb server, όπως φαίνεται στην Εικόνα 84 .

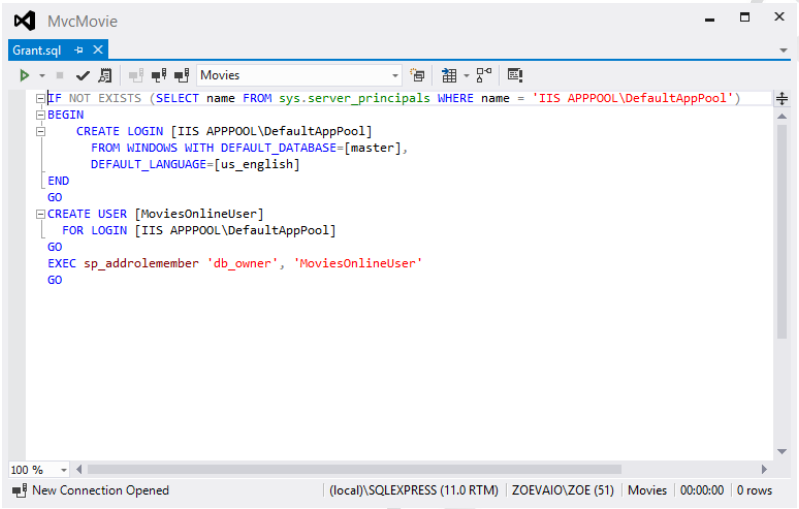
```
<!--<add name ="DefaultConnection"
connectionString="Data
Source=(localdb)\v11.0;AttachDbFilename=|DataDirectory|aspnet-MvcMovie-
20140722001856.mdf;
Integrated Security=True;Connect Timeout=30;Encrypt=False;
TrustServerCertificate=False" providerName="System.Data.SqlClient" />

<add name ="MovieDBContext"
connectionString="Data
Source=(localdb)\v11.0;AttachDbFilename=|DataDirectory|Movies.mdf;
Integrated Security=True;Connect Timeout=30;Encrypt=False;
TrustServerCertificate=False" providerName="System.Data.SqlClient" />-->

<add name="DefaultConnection"
providerName="System.Data.SqlClient"
connectionString="Data Source=.\SQLEXPRESS;
Initial Catalog=aspnet-MvcMovie-20140722001856;Integrated Security=True;" />

<add name="MovieDBContext"
providerName="System.Data.SqlClient"
connectionString="Data Source=.\SQLEXPRESS;
Initial Catalog=Movies;Integrated Security=True"/>
```

Εικόνα 84.Connection string για χρήση του SQLEXPRESS server

Η βάση δεδομένων Movies στον SQLEXPRESS server δημιουργήθηκε τρέχοντας την εντολή update-database στον Package Manager Console του Visual Studio, με τον τρόπο που αναλύθηκε στην παράγραφο 6.4.

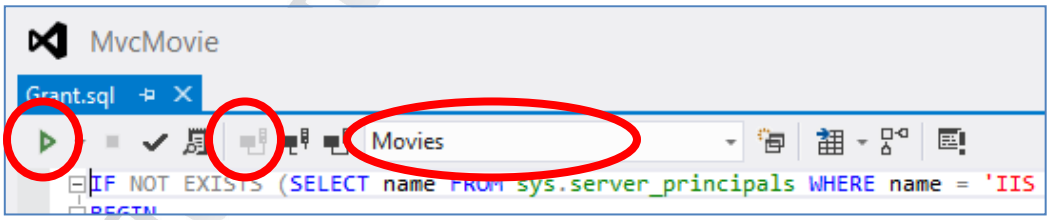
Για την βάση δεδομένων asp-net-MvcMovie-20140722001856 έπρεπε να δημιουργηθεί πάλι ένας χρήστης και να του αποδοθεί ρόλος Administrator για να μπορεί να διαχειρίζεται όλους τους χρήστες μέσω του γραφικού περιβάλλοντος διεπαφής, οπότε εκτελέστηκε η εφαρμογή μέσω του Visual Studio και ακολουθήθηκε η διαδικασία που περιγράφεται στην παράγραφο 7.2.1 για τη δημιουργία του και την απόδοση του ρόλου. Η εφαρμογή δηλαδή σε αυτήν την φάση εκτελέστηκε μέσω του IIS Express χρησιμοποιώντας όμως τον SQL Server Express.

Όταν η εφαρμογή θα εκτελείται μέσω του IIS, η πρόσβαση στις βάσεις δεδομένων θα γίνεται χρησιμοποιώντας τα διαπιστευτήρια του IIS. Επομένως θα πρέπει να αποδοθούν δικαιώματα πρόσβασης στις βάσεις δεδομένων στον IIS (db_owner)[38]. Αυτό επιτεύχθηκε με το παρακάτω SQL script Grant.sql της που δημιουργήθηκε στο project της εφαρμογής και εκτελέστηκε για τη βάση δεδομένων Movies και aspnet-MvcMovie-20140722001856.



Εικόνα 85.Script απόδοσης στον IIS δικαιωμάτων πρόσβασης στις βάσεις δεδομένων Movies και aspnet-MvcMovie-20140722001856 του SQL Express

Για την εκτέλεση του script έγινε σύνδεση με τον server και επιλογή του κουμπιού εκτέλεσης του script για τις δύο βάσεις δεδομένων διαδοχικά.

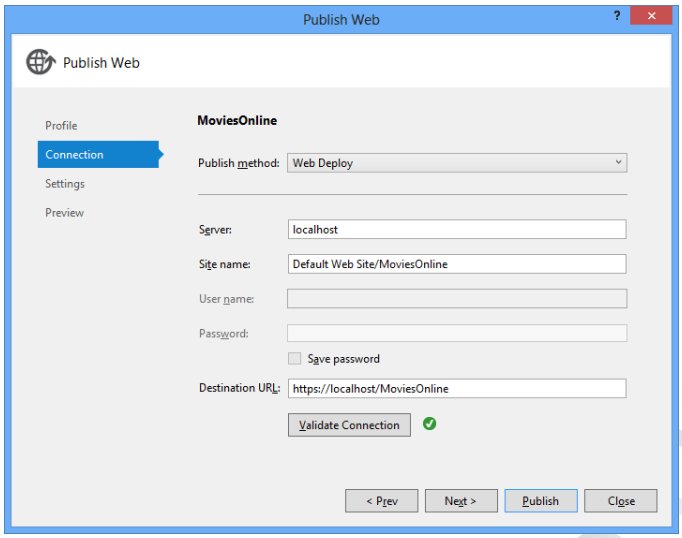


Εικόνα 86. Σύνδεση με SQL Express και εκτέλεση του script Grant.sql για την βάση δεδομένων Movies

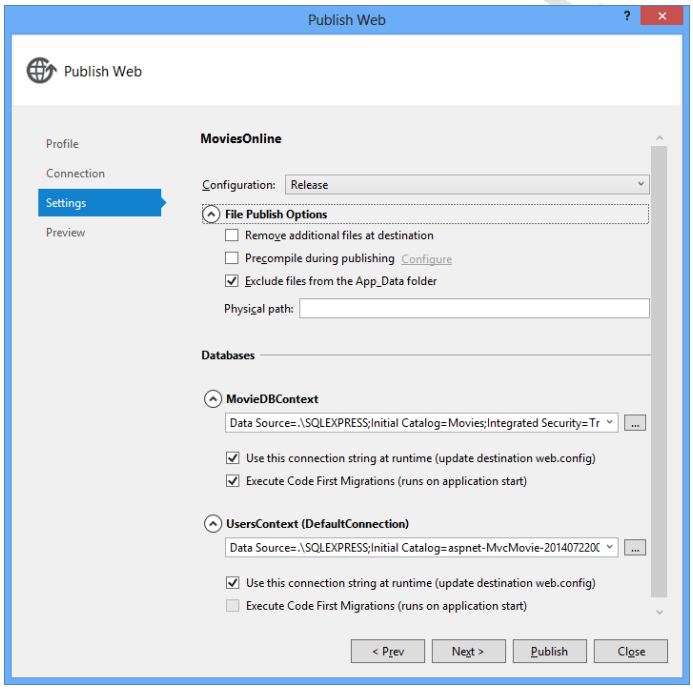
8.2 Ρυθμίσεις προφίλ εγκατάστασης εφαρμογής στον IIS μέσω Visual Studio

Για την εγκατάσταση της εφαρμογής στον IIS χρησιμοποιήθηκε η διαδικασία δημοσίευσης (Publish) που παρέχει το Visual Studio, κατά την οποία δημιουργείται ένα προφίλ δημοσίευσης και έπειτα με ένα κλικ εγκαθίσταται η εφαρμογή στον IIS.

Επειδή η εφαρμογή θα εγκατασταθεί στον IIS που βρίσκεται στον τοπικό υπολογιστή, η λειτουργία του Visual Studio θα πρέπει να γίνεται με δικαιώματα Administrator. Επιλέγοντας με δεξί κλικ από τον Solution Explorer το project MoviesOnline, την επιλογή Publish, δημιουργήθηκε το προφίλ δημοσίευσης της εφαρμογής με όνομα MoviesOnline και τις ρυθμίσεις που φαίνονται στις παρακάτω εικόνες.



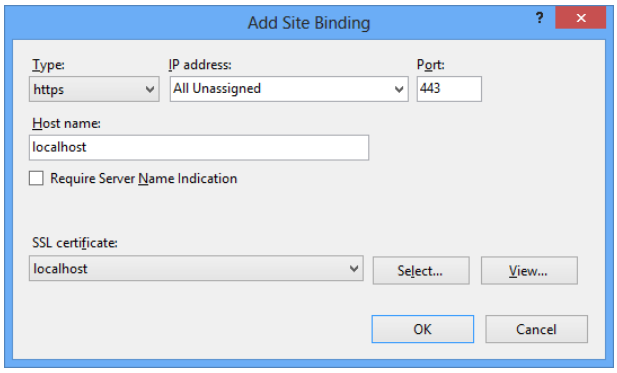
Εικόνα 87. Ρυθμίσεις σύνδεσης για τη δημοσίευση της εφαρμογής στον IIS



Εικόνα 88. Ρυθμίσεις προφίλ δημοσίευσης για σύνδεση με τις βάσεις δεδομένων

Αφού ολοκληρώθηκαν οι ρυθμίσεις και επιλέχθηκε Publish, η εφαρμογή ξεκίνησε να λειτουργεί μέσω IIS αλλά επέστρεφε σφάλμα HTTP 503 διότι έπρεπε να ρυθμιστεί το «δέσιμο» (binding) της εφαρμογής με το URL που δόθηκε στη ρύθμιση που φαίνεται στην Εικόνα 87, στο πεδίο Destination

URL. Για τη ρύθμιση αυτή εκτελέστηκε ο IIS Manager και δημιουργήθηκε το binding που φαίνεται στην Εικόνα 89 για το Default Web Site του IIS. Όπως φαίνεται στην εικόνα, επιλέχθηκε πρωτόκολλο https στην προκαθορισμένη θύρα 443 και εισήχθη το πιστοποιητικό του server που είχε δημιουργηθεί και εισαχθεί στην λίστα των έμπιστων πιστοποιητικών του συστήματος.



Εικόνα 89. Δημιουργία binding στον IIS για την εφαρμογή

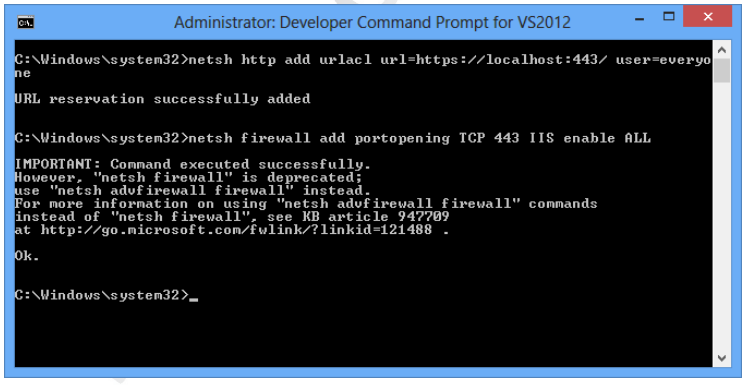
8.3 Επιπλέον ρυθμίσεις συστήματος

Εκτός από τις ρυθμίσεις που αναλύθηκαν στις παραγράφους 8.1 και 8.2 για να μπορέσει να λειτουργήσει η εφαρμογή προστέθηκε Url reservation έτσι ώστε να επιτρέπεται από το Λειτουργικό Σύστημα αιτήματα HTTP προς το url που αντιστοιχεί στην εφαρμογή. Η εντολή που εκτελέστηκε σε περιβάλλον γραμμής εντολών του Λειτουργικού Συστήματος, με δικαιώματα administrator, ήταν η :

```
netsh http add urlacl url=https://localhost:443/ user=everyone
```

Επιπλέον, δημιουργήθηκε κανόνας πρόσβασης στο Τείχος Προστασίας του Λειτουργικού Συστήματος έτσι ώστε να επιτρέπεται σε αιτήματα που έρχονται από άλλο υπολογιστή του δικτύου να αποστέλλονται στον IIS στην θύρα 443, με την εντολή :

```
netsh firewall add portopening TCP 443 IIS enable ALL
```



Εικόνα 90. Εκτέλεση εντολών url reservation και δημιουργίας κανόνα πρόσβασης στο Τείχος Προστασίας

Ύστερα από τις παραπάνω ρυθμίσεις, η εφαρμογή εκτελείται χωρίς πρόβλημα στο https://localhost/MoviesOnline που σημαίνει ότι αν ακολουθηθεί η διαδικασία που αναλύθηκε στο

παρόν Κεφάλαιο σε κάποιο άλλο σύστημα, σε κανονικό server παραγωγής, η εφαρμογή θα εγκατασταθεί χωρίς πρόβλημα.

Κεφάλαιο 9°

9 Συμπεράσματα

Οι διαδικτυακές εφαρμογές σήμερα, αρκετά χρόνια μετά την ευρεία χρήση τους, είναι «γεμάτες» από ευπάθειες που πρέπει να αντιμετωπίσουν. Η κατανόηση των απειλών ασφάλειας που αυτές αντιμετωπίζουν βρίσκεται σε διαρκή εξέλιξη. Όσον αφορά το κοντινό μέλλον, δεν υπάρχουν κάποιες ενδείξεις ότι οι απειλές αυτές πρόκειται να εξαλειφθούν αλλά θα προσεγγίζουν τα «σπίτια» των διαδικτυακών εφαρμογών πάντα μέσω πρωτοκόλλου HTTP. Αυτό σημαίνει ότι τα παραδοσιακά «οχυρά» των υπολογιστών όπως το Τείχος Προστασίας που παρέχουν τα Λειτουργικά Συστήματα και οι υπόλοιποι αμυντικοί μηχανισμοί των συστημάτων, δεν επαρκούν για προστασία από αυτές τις απειλές όταν χρησιμοποιούνται μεμονωμένα.

Η προστασία μιας διαδικτυακής εφαρμογής περιλαμβάνει την εφαρμογή μέτρων ασφάλειας σε τρία επίπεδα: δικτύου, συστήματος και εφαρμογής. Η ύπαρξη ασφαλούς δικτύου μετάδοσης πληροφοριών είναι ουσιώδης απαίτηση, όπως επίσης και μια ασφαλής υποδομή πλατφόρμας στην οποία «πατάει» μια εφαρμογή. Πιο σημαντικό απ' όλα όμως είναι η ίδια η εφαρμογή να έχει σχεδιαστεί και υλοποιηθεί έχοντας ως βάση τετριμμένες αρχές ασφάλειας και πατώντας πάνω σε κατευθυντήριες γραμμές που οδηγούν προς τη θωράκισή της.

Για να θωρακιστεί μια διαδικτυακή εφαρμογή όσο το δυνατόν καλύτερα, ένας προγραμματιστής, εκτός από τους μηχανισμούς ασφάλειας που μπορεί να έχει στη διάθεσή του, θα πρέπει να γνωρίζει και τους μηχανισμούς επίθεσης που πιθανόν να χρησιμοποιήσει ένας υποψήφιος επιτιθέμενος. Μόνο έτσι μπορεί να είναι σίγουρος ότι έχει καταβάλει κάθε δυνατή προσπάθεια για να αντιμετωπίσει τυχόν απειλές ασφάλειας που μπορεί να παρουσιαστούν στην εφαρμογή του.

Παράγοντας που παίζει σημαντικό ρόλο στην αποτελεσματική υλοποίηση μέτρων ασφαλείας διαδικτυακών εφαρμογών είναι η τήρηση μεθοδολογίας ανάλυσης και μοντελοποίησης των πιθανών απειλών. Η κατηγοριοποίηση των απειλών αυτών, ανάλογα με τους στόχους και τους σκοπούς κάθε επίθεσης, βοηθάει πάρα πολύ στη οργάνωση μιας σωστής και αποτελεσματικής στρατηγικής ασφάλειας για την αντιμετώπισή τους.

Οι τεχνολογίες ασφάλειας που παρέχει το .NET Framework περιέχουν όλα τα απαραίτητα εργαλεία για την υλοποίηση μηχανισμών ασφάλειας διαδικτυακών εφαρμογών. Ο χώρος ονομάτων System.Web.Security περιέχει τις κλάσεις που χρησιμοποιούνται για την υλοποίηση των μηχανισμών αυθεντικοποίησης και εξουσιοδότησης χρηστών, απαραίτητα συστατικά για αντιμετώπιση απειλών όπως οι brute force attack, network eavesdropping, dictionary attack, cookie replay και κλοπή διαπιστευτηρίων.

Επίσης σημαντικός μηχανισμός ασφάλειας που παρέχει το .NET Framework είναι μέσω της κλάσης System.ComponentModel.DataAnnotation,για να γίνεται επικύρωση των δεδομένων που εισάγουν οι χρήστες στην εφαρμογή. Ως γνωστόν, οι περισσότερες διαδικτυακές εφαρμογές αντιμετωπίζουν το παρακάτω βασικό πρόβλημα ασφάλειας:

Οι χρήστες μπορούν να υποβάλουν οποιαδήποτε δεδομένα προς αυτές μέσω φορμών ή παραμέτρων όπως cookies και query strings.

Άρα από κάθε άποψη, οποιαδήποτε αλληλεπίδραση χρήστη με μια εφαρμογή διαδικτύου θα πρέπει να θεωρείται κακόβουλη εκτός και αν αποδειχθεί το αντίθετο. Αν αυτή η κατάσταση δεν αντιμετωπιστεί σωστά τότε οι εφαρμογές γίνονται ευάλωτες στις κρίσιμες απειλές, cross-site scripting και SQL injection.

Χρησιμοποιώντας τις βιβλιοθήκες ασφάλειας του .NET Framework, έχουν υλοποιηθεί μηχανισμοί ασφάλειας όπως στην περίπτωση της παρούσας διατριβής, το SecurityGuard, σύστημα διαχείρισης χρηστών. Οι μηχανισμοί αυτοί, καθ' όσον είναι ανοιχτού κώδικα, μπορούν εύκολα να

χρησιμοποιηθούν και να τροποποιηθούν κατάλληλα προς όφελος της εφαρμογής με μικρό κόστος χρόνου αλλά με μεγάλη αξιοπιστία.

Το Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών που παρέχει τα απαιτούμενα εργαλεία για την υποστήριξη του έργου των προγραμματιστών. Η συμμετοχή του στα στάδια ανάπτυξης λογισμικού ήταν διαρκής κατά τη διάρκεια εκπόνησης της διατριβής. Με τη βοήθειά του εκτελέστηκαν οι περισσότερες διαδικασίες που απαιτήθηκαν όπως η συγγραφή του κώδικα της εφαρμογής, η δημιουργία των βάσεων δεδομένων και η αλληλεπίδραση με αυτές, η ρύθμιση του server εφαρμογής τόσο κατά τη φάση της ανάπτυξης όσο και κατά τη φάση της εγκατάστασης της εφαρμογής και στην τελική φάση η εγκατάσταση της εφαρμογής σε server που εξομοιώνει περιβάλλον παραγωγής. Η χρήση του κατά τη διάρκεια ανάπτυξης εφαρμογών επιβάλλεται να είναι συνεχής, κάτι που σημαίνει ότι οι προγραμματιστές πρέπει οπωσδήποτε να είναι εξοικειωμένοι με αυτό και να γνωρίζουν τα εργαλεία που παρέχει, έτσι ώστε να μπορούν να τα χρησιμοποιούν αποτελεσματικά για την εκτέλεση του έργου τους. Χωρίς τη συμβολή του η ανάπτυξη εφαρμογών θα ήταν εξαιρετικά δύσκολη και χρονοβόρα διαδικασία.

Η προσπάθεια για θωράκιση μιας διαδικτυακής εφαρμογής δεν πρέπει να περιορίζεται μόνο στα θέματα που πρέπει να ληφθούν υπ' όψη κατά τη διαδικασία ανάπτυξής της. Ακόμα και μετά από το στάδιο εγκατάστασης της εφαρμογής πρέπει να γίνεται εντοπισμός ευπαθειών αλλά και παρακολούθηση της αποτελεσματικότητας των εφαρμοσμένων μέτρων προστασίας μιας εφαρμογής. Για το σκοπό αυτό έχουν αναπτυχθεί εργαλεία εντοπισμού και ανάλυσης ευπαθειών όπως τα Open Vas [39], Burp Suite [40], Retina [41], Nexpose [42], MBSA – Microsoft Baselive Security Analyser [43], Core Impact [44], Nessus [45], Nmap [46], QualysGuard [47], GFI LanGuard [48] κ.α. Ανάλογα με το εργαλείο και τις λειτουργίες του που χρησιμοποιούνται, μπορούν να εντοπιστούν και αντίστοιχες ευπάθειες. Το γεγονός αυτό βοηθάει στην υλοποίηση επιπλέον αντίμετρων ακόμα και με ανασχεδιασμό του κώδικα συγγραφής της εφαρμογής. Εκτελώντας συχνούς ελέγχους, είναι εφικτό μια εφαρμογή να παραμένει όσο το δυνατόν πιο ασφαλής.

Συνοψίζοντας, το .NET παρέχει πλούσια γκάμα εργαλείων για ανάπτυξη ασφαλών διαδικτυακών εφαρμογών. Είναι σίγουρο ότι για έναν προγραμματιστή απαιτούνται πολλές ώρες μελέτης των εργαλείων, δοκιμής τους και εξάσκησης με αυτά, αλλά εφόσον γίνουν «κτήμα» του, η υλοποίηση ασφαλούς εφαρμογής γίνεται πολύ εύκολα και αξιόπιστα. Αν όλα τα παραπάνω συνδυαστούν και με χρήση εργαλείων εντοπισμού ευπαθειών, τότε είναι σίγουρο ότι η εφαρμογή θα καταλήξει στο επιθυμητό επίπεδο ασφάλειας.

Κεφάλαιο 10ο

10 Βιβλιογραφικές Πηγές

[1] «Overview of the .NET Framework,» MIcrosoft, [Ηλεκτρονικό]. Available: http://msdn.microsoft.com/en-us/library/zw4w595w%28v=vs.110%29.aspx. [Πρόσβαση 24 12 2014].

[2] «Visual Studio Features Overview,» Microsoft, 5 June 2014. [Ηλεκτρονικό]. Available: http://www.visualstudio.com/explore/features-overview-vs. [Πρόσβαση 24 12 2014].

[3] «Introduction to the C# Language and the .NET Framework,» Microsoft, [Ηλεκτρονικό]. Available: http://msdn.microsoft.com/en-us/library/z1zx9t92.aspx. [Πρόσβαση 24 December 2014].

[4] K. Schaefer, J. Cochran, S. Forsyth, D. Glendenning και B. Perkins, Proffesional Microsoft IIS 8, Indianapolis, Indiana: John Wiley & Sons, Inc., 2013.

[5] T. Dierks, «The Transport Layer Security (TLS) Protocol Version 1.1,» Network Working Group , 25 April 2006. [Ηλεκτρονικό]. Available: http://www.ietf.org/rfc/rfc4346.txt. [Πρόσβαση 24 December 2014].

[6] A. Leff και J. T. Rayfield, «Web-application development using the Model/View/Controller design pattern,» σε *Fifth IEEE International Enterprise Distributed Object Computing Conference*, Seattle, Washington, 2001.

[7] L. S. Shklar και R. Rosen, Web Application Architecture. Principles, protocols and practices, West Sussex PO19 8SQ, England: John Wiley & Sons Ltd, 2003.

[8] T. Berners-Lee, J. C. Mogul και L. Masinter, «RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1,» June 1999. [Ηλεκτρονικό]. Available: http://tools.ietf.org/html/rfc2616. [Πρόσβαση 23 December 2014].

[9] C. Kambal, «3-Tier Architecture,» [Ηλεκτρονικό]. Available: http://channukambalyal.tripod.com/NTierArchitecture.pdf. [Πρόσβαση 25 December 2014].

[10] «What is Entity Framework?,» [Ηλεκτρονικό]. Available: http://www.entityframeworktutorial.net/what-is-entityframework.aspx. [Πρόσβαση 7 January 2015].

[11] J. Chadwick, T. Snyder και H. Panda, Programming ASP.NET MVC4, United States of America: O'REILLY Media, Inc, 2012.

[12] J. Galloway, P. Haack, B. Wilson και S. K. Allen, Professional ASP.NET MVC 4, Indianapolis: John Wiley & Sons, 2012.

[13] B. Johnson, Professional Visual Studio2012, Indianapolis: John Wiley & Sons, Inc., 2013.

[14] J. Lerman και R. Miller, Programming Entity Framework: Code First, Sebastopol: O'Reilly Media, 2012.

[15] A. Mackman, S. Vasireddy, M. Dunner, R. Escamilla, A. Murukan και J. Meier, Improving Web Application Security. Threats and Countermeasures, Microsoft Corporation, 2003.

[16] Σ. Κάτσικας, Ασφάλεια Υπολογιστών, Πάτρα: Ελληνικό Ανοικτό Πανεπιστήμιο, 2011.

[17] W. Stallings, Cryptography and Network Security Principles and Practice 5th Edition, Prentice Hall, 2011.

[18] J. Scambray, V. Liu και C. Sima, Hacking Exposed Web Applications 3rd Edition, McGraw-Hill Companies.

[19] D. Stuttard και M. Pinto, The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws, Indianapolis: Wiley Publishing, Inc., 2008.

[20] J. Meier. [Ηλεκτρονικό]. Available: http://blogs.msdn.com/b/jmeier/archive/2008/04/07/security-principles.aspx.

[21] R. Morris και K. Thompson, «Password security: a case history,» *Communications of the ACM,* pp. 594-597, 03 04 1978.

[22] Microsoft, «Code Access Security,» Microsoft, 12 October 2014. [Ηλεκτρονικό]. Available: http://msdn.microsoft.com/en-us/library/930b76w0%28v=vs.90%29.aspx. [Πρόσβαση 12 October 2014].

[23] B. Lakshmiraghavan, Pro ASP.NET Web API Security, Apress, 2012.

[24] B. Dorrans, Beginning ASP.NET Security, Chichester, West Sussex: John Wiley & Sons Ltd, 2010.

[25] «ASP.NET Request Validation,» OWASP, 26 November 2014. [Ηλεκτρονικό]. Available: https://www.owasp.org/index.php/ASP.NET_Request_Validation. [Πρόσβαση 23 December 2014].

[26] «Microsoft Web Protection Library - AntiXSS,» Microsoft, [Ηλεκτρονικό]. Available: https://wpl.codeplex.com/. [Πρόσβαση 23 December 2014].

[27] «Walkthrough: Encrypting Configuration Information Using Protected Configuration,» Microsoft Developer Network, [Ηλεκτρονικό]. Available: http://msdn.microsoft.com/en-us/library/dtkwfdky%28v=vs.140%29.aspx. [Πρόσβαση 23 December 2014].

[28] «ASP.NET IIS Registration Tool (Aspnet_regiis.exe),» Microsoft Developer Network, [Ηλεκτρονικό]. Available: http://msdn.microsoft.com/en-us/library/k6h9cz8h%28v=vs.140%29.aspx. [Πρόσβαση 24 December 2014].

[29] M. MacDonald, Beginning ASP.NET 2.0 in C# 2005: From Novice to Professional, New York: Apress, 2006.

[30] «System.Web.SessionState Namespace,» Microsoft Developer Network, [Ηλεκτρονικό]. Available: http://msdn.microsoft.com/en-us/library/system.web.sessionstate%28v=vs.110%29.aspx. [Πρόσβαση 23 December 2014].

[31] «System.Security.Cryptography Namespace,» Microsoft Developer Network, [Ηλεκτρονικό]. Available: http://msdn.microsoft.com/en-us/library/system.security.cryptography%28v=vs.110%29.aspx. [Πρόσβαση 24 December 2014].

[32] «Error Logging Modules and Handlers for ASP.NET,» [Ηλεκτρονικό]. Available: https://code.google.com/p/elmah/. [Πρόσβαση 24 December 2014].

[33] Microsoft, «Download Microsoft® SQL Server® 2012 Express from Official Microsoft Download Center,» Microsoft, 2 October 2014. [Ηλεκτρονικό]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=29062. [Πρόσβαση 2 October 2014].

[34] Π. Κοτζανικολάου, Ασφάλεια Πληροφοριών, Σημειώσεις μαθήματος "Ασφάλεια Πληροφοριών" ΠΜΣ Πληροφορικής, Πειραιάς: Πανεπιστήμιο Πειραιώς, 2013.

[35] W. King, «SecurityGuard - NuGet package for ASP.NET Membership,» MVC CENTRAL, 25 Αύγουστος 2011. [Ηλεκτρονικό]. Available: http://www.mvccentral.net/Story/Details/tools/kahanu/securityguard-nuget-package-for-asp-net-membership. [Πρόσβαση 29 Νοέμβριος 2014].

[36] «Using a CAPTCHA to Prevent Bots from Using Your ASP.NET Web Razor) Site,» Microsoft ASP.NET, [Ηλεκτρονικό]. Available: http://www.asp.net/web-pages/overview/security/using-a-catpcha-to-prevent-automated-programs-%28bots%29-from-using-your-aspnet-web-site. [Πρόσβαση 23 December 2014].

[37] «Recaptcha for .NET,» CodePlex. Project Hosting for Open Source Software, [Ηλεκτρονικό]. Available: http://recaptchanet.codeplex.com/. [Πρόσβαση 23 December 2014].

[38] T. Hunt, OWASP Top 10 for .NET developers, 2011.

[39] « OpenVAS - Open Vulnerability Assessment System,» OpenVAS, 8 December 2014. [Ηλεκτρονικό]. Available: http://www.openvas.org/. [Πρόσβαση 24 December 2014].

[40] «Burp Suite,» PortSwigger Ltd, 4 March 2014. [Ηλεκτρονικό]. Available: http://portswigger.net/burp/. [Πρόσβαση 24 December 2014].

[41] «Retina Network Security Scanner,» BeyondTrust, [Ηλεκτρονικό]. Available: http://www.beyondtrust.com/Products/RetinaNetworkSecurityScanner/. [Πρόσβαση 24 December 2014].

[42] «Vulnerability Management & Risk Management Software,» Rapid7, [Ηλεκτρονικό]. Available: http://www.rapid7.com/products/nexpose/. [Πρόσβαση 24 December 2014].

[43] «Microsoft Baseline Security Analyzer – MBSA,» Microsoft, [Ηλεκτρονικό]. Available: http://technet.microsoft.com/en-us/security/cc184924.aspx. [Πρόσβαση 24 December 2014].

[44] «Penetration Testing with Core Impact Pro,» Core Security, [Ηλεκτρονικό]. Available: http://www.coresecurity.com/core-impact-pro. [Πρόσβαση 24 December 2014].

[45] «Nessus Vulnerability Scanner,» Tenable Network Security , [Ηλεκτρονικό]. Available: http://www.tenable.com/products/nessus/nessus-vulnerability-scanner. [Πρόσβαση 24 December 2014].

[46] Nmap, «Nmap - Free Security Scanner For Network Exploration & Security Audits,» Gordon Lyon, [Ηλεκτρονικό]. Available: http://nmap.org/. [Πρόσβαση 24 December 2014].

[47] «Qualys Enterprise for Global Organizations,» Qualys, Inc, [Ηλεκτρονικό]. Available: https://www.qualys.com/enterprises/qualysguard/. [Πρόσβαση 24 December 2014].

[48] GFI, «GFI - Network Vulnerability Scanner,» GFI LanGuard, [Ηλεκτρονικό]. Available: http://www.gfi.com/products-and-solutions/network-security-solutions/gfi-languard. [Πρόσβαση 24 December 2014].

[49] B. Walther και B. Hope, Web Security Testing Cookbook:Systematic Techniques to Find Problems Fast, O'Reilly, 2008.

[50] J. Andress, The Basics of Information Security, Syngress, Elsevier, 2011.

[51] «ASP.NET MVC 4,» Microsoft, 24 December 2014. [Ηλεκτρονικό]. Available: http://www.asp.net/mvc/mvc4. [Πρόσβαση 24 December 2014].