ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Διδακτικής της Τεχνολογίας και Ψηφιακών Συστημάτων

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

"Δικτυοκεντρικά Συστήματα"

" Αξιολόγηση επιπέδου ασφαλείας, εντοπισμός προβλημάτων και πλεονεκτημάτων ασφάλειας των τεχνολογιών ανάπτυξης της Microsoft"

Εκπόνηση: **Παπακωνσταντίνου Κωνσταντίνος, *ΜΕ/0583*,**
kpapakonst@yahoo.gr
Επιβλέπων Καθηγητής: **Χρήστος Ξενάκης**

Αθήνα 2009

1. Εισαγωγή 6
2. Microsoft Sql Server 7
2.1 Εισαγωγή 7
2.2 Ιστορικά στοιχεία.....7
2.3 Μηχανισμός ασφάλειας.....8
2.3.1 Αυθεντικοποίηση (επίπεδο server).....8
2.3.2 Εξουσιοδότηση (επίπεδο server) 11
2.3.3 Εξουσιοδότηση (επίπεδο βάσης δεδομένων).....13
2.4 Πρακτικές μεγιστοποίησης ασφάλειας 20
2.4.1 Χρήση Windows authentication 21
2.4.2 Συχνός έλεγχος των δικαιωμάτων χρηστών και groups22
2.4.3 Χρήση «ισχυρού» κωδικού για το login sa 22
2.4.4 Χρήση stored procedures για υλοποίηση μηχανισμού ασφάλειας.....23
2.4.5 Απενεργοποίηση των Sql Mail δυνατοτήτων 23
2.4.6 Περιορισμός των δικαιωμάτων των Sql Server services24
2.4.7 Έλεγχος των συνδέσεων 26
2.4.8 Χρήση του ασφαλέστερου συστήματος αρχείων.....27
2.4.9 Εγκατάσταση των τελευταίων service packs 27
2.4.10 Κατάλληλες ρυθμίσεις στο firewall 28
2.4.11 Φυσική και λογική απομόνωση του Sql Server.....28
2.5 Διαπιστωμένα προβλήματα ασφάλειας και «έξυπνες» στρατηγικές επίθεσης 29
2.5.1 Ανάλυση (parsing) παραμέτρων σε Extended stored procedures (Sql Server 2000).....29
2.5.2 Επαναχρησιμοποίηση μιας cached σύνδεσης (Sql Server 2000) 31
2.5.3 Text formatting μέθοδοι του Sql Server που περιέχουν μη ελεγχόμενα buffers (Sql Server 2000).....32
2.5.4 Μη ελεγχόμενα buffers στη function απομακρυσμένων πηγών δεδομένων του Sql Server (Sql Server 2000). 34

2.5.5 Log files με ευαίσθητες πληροφορίες κατά την εγκατάσταση του Sql Server 2000 35
2.5.6 Αδυναμία ασφάλειας στη διαδικασία μαζικής εισαγωγής δεδομένων (Sql Server 2000) 38
2.5.7 Αδυναμία ασφάλειας σχετικά με την αποθήκευση των πληροφοριών του Sql Server service λογαριασμού (Sql Server 2000) 39
2.5.8 Αδυναμίες ασφάλειας στις Sql Server 2000 Utilities 39
2.5.9 Αδυναμία ασφάλειας κατά τη διαδικασία αυθεντικοποίησης στον Sql Server 2000 42
2.5.10 Αδυναμία ασφάλειας στις προγραμματιζόμενες εργασίες του Sql Server 2000 42
2.5.11 Sql Injection 43
3. Internet Information Server (IIS) 48
3.1 Εισαγωγή 48
3.2 Ιστορικά στοιχεία 48
3.3 Μηχανισμός ασφάλειας 49
3.3.1 Μηχανισμός αυθεντικοποίησης 49
3.3.2 Δικαιώματα χρήσης πόρων των web sites 57
3.3.3 Βελτιωμένος έλεγχος αξιοπιστίας δεδομένων 60
3.3.4 Σύνθετοι μηχανισμοί logging 60
3.3.5 Απομόνωση applications 61
3.4 Πρακτικές μεγιστοποίησης ασφάλειας 62
3.4.1 Ενημέρωση του φιλοξενούντος λειτουργικού με τις τελευταίες ενημερώσεις 62
3.4.2 Απενεργοποίηση πρόσβασης στο "Default Web site".... 62
3.4.3 Ενεργοποίηση logging μηχανισμού 63
3.4.4 Κατάλληλη χρήση partitions και directories 65
3.4.5 Επιλογή κατάλληλου hosting συστήματος 67
3.4.6 Ενεργοποίηση Rapid-Fail προστασίας 67
3.4.7 Σύνθετες ρυθμίσεις για την ανάλυση των URL requests 68

3.4.8 Κατάλληλη επιλογή δικαιωμάτων πρόσβασης 69
3.4.9 Χρήση του SSL πρωτοκόλλου 70
3.4.10 «Έξυπνη» ονοματολογία των web αρχείων και directories..... 72
3.5 Διαπιστωμένα προβλήματα ασφάλειας και «έξυπνες» στρατηγικές επίθεσης 72
3.5.1 Αδυναμία ασφάλειας που επιτρέπει εκτέλεση κακόβουλα σχεδιασμένων Active Server Pages..... 72
3.5.2 Αδυναμία ασφάλειας στον handler των WebDAV XML μηνυμάτων 74
3.5.3 Αδυναμία ασφάλειας σχετικά με λανθασμένη αναγνώριση της προέλευσης ενός request..... 75
4. .NET Framework 78
4.1 Εισαγωγή 78
4.2 Μηχανισμός ασφάλειας..... 79
4.2.1 ASP.NET Αυθεντικοποίηση..... 79
4.2.2 ASP.NET εξουσιοδότηση..... 80
4.2.3 Ασφάλεια πρόσβασης κώδικα 84
4.2.4 Κρυπτογραφικές Υπηρεσίες 91
4.2.5 Ασφάλεια βασιζόμενη σε ρόλους 100
4.3 Πρακτικές μεγιστοποίησης ασφάλειας 101
4.3.1 Προστασία δεδομένων 101
4.3.2 Προστασία πρόσβασης σε μεθόδους (Δηλωτική ασφάλεια) 102
4.3.3 Interfaces και LinkDemands 105
4.3.4 Virtual internal override μέθοδοι 106
4.3.5 Wrapper κώδικας 107
4.3.6 Unmanaged κώδικας 108
4.3.7 Σύμβαση Ονοματολογίας για μεθόδους unmanaged κώδικα 109
4.3.8 Έλεγχος δεδομένων εισόδου από χρήστες 110
4.3.9 Κίνδυνοι από το .NET Remoting 112

4.3.10 Serialization 113
4.3.11 Επιβολή permissions..... 114
4.3.12 Επικίνδυνα Permissions..... 116
4.4 Αδυναμίες ασφάλειας 117
4.4.1 Ανεπιθύμητο client-side scripting 117
4.4.2 Έκθεση ευαίσθητων πληροφοριών..... 118
5. Πηγές 120

1. Εισαγωγή

Τα τελευταία χρόνια, η ολοένα και μεγαλύτερη διείσδυση της επιστήμης της πληροφορικής στον τομέα του εμπορίου αλλά και σε τομείς όπως η επικοινωνία και η παροχή υπηρεσιών έχει καταστήσει την ασφάλεια των δημόσια προσβάσιμων πληροφοριακών συστημάτων παράγοντα πρωταρχικής σημασίας.

Στην παρούσα πτυχιακή εργασία θα μελετηθούν όσον αφορά την ασφάλεια τα τρία βασικότερα εργαλεία που παρέχει η Microsoft για τη δημιουργία ολοκληρωμένων εφαρμογών, δηλαδή ο Sql Server, ο IIS και το .NET Framework. Συγκεκριμένα, θα πραγματοποιηθεί μια επισκόπηση του μηχανισμού ασφάλειας του κάθε εργαλείου, θα μελετηθούν κάποιες πρακτικές που μπορούν να αυξήσουν το επίπεδο ασφάλειας, ενώ θα αναλυθούν και κάποια από τα προβλήματα ασφάλειας που έχουν προκύψει σε αυτά τα εργαλεία.

2. Microsoft Sql Server

2.1 *Εισαγωγή*

Ο Sql Server αποτελεί ένα Σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων (Σ.Δ.Β.Δ) της Microsoft που προορίζεται αποκλειστικά για Windows συστήματα. Αποτελεί ένα πολύ ισχυρό πακέτο, το οποίο έχει διεισδύσει δυναμικά τα τελευταία χρόνια στην αγορά των Σ.Δ.Β.Δ.

Στα επόμενα κεφάλαια, θα παρουσιαστεί ο μηχανισμός ασφαλείας και επίσης θα επιχειρηθεί μία αξιολόγηση του επιπέδου ασφαλείας του Sql Server.

2.2 *Ιστορικά στοιχεία*

Οι «ρίζες» του Sql Server βρίσκονται στον Sybase Sql Server, που αρχικά αναπτυσσόταν μόνο για συστήματα Unix από την εταιρεία Sybase. Το 1988, οι εταιρείες Sybase, Microsoft και Ashton-Tate ολοκλήρωσαν από κοινού την ανάπτυξη του Sql Server 1.0 για το λειτουργικό σύστημα OS/2. Στη συνέχεια η απόσυρση της εταιρείας Ashton-Tate από την ανάπτυξη του προϊόντος καθώς και η μεταφορά του Sql Server στην πλατφόρμα των Windows NT, έδωσαν στη Microsoft των πρώτο λόγο στη συνεργασία της με τη Sybase.

Η Microsoft και η Sybase εμπορεύονταν και υποστήριζαν από κοινού το προϊόν μέχρι την έκδοση 4.21. Το 1993 η συμφωνία από κοινού ανάπτυξης του προϊόντος μεταξύ της Microsoft και της Sybase έληξε και οι δύο εταιρείες διαχώρισαν τους δρόμους τους, συνεχίζοντας να αναπτύσσουν κάθε μία το δικό της πλέον προϊόν. Η πρώτη έκδοση του Sql Server χωρίς τη συμμετοχή της Sybase ήταν η έκδοση 6.0 η οποία ήταν σχεδιασμένη αποκλειστικά για τα Windows NT, ενώ οι εκδόσεις του Microsoft Sql Server μέχρι και το 1994 περιείχαν, βάσει

νομικής απόφασης, ενδείξεις πνευματικών δικαιωμάτων της Sybase, ως ένδειξη της προέλευσης τους.

Η έκδοση 7.0 του Sql Server ήταν το πρώτο πραγματικά γραφικά διαχειριζόμενο σύστημα διαχείρισης βάσεων δεδομένων, ενώ η έκδοση 2000 του Sql Server ήταν το πρώτο σύστημα διαχείρισης βάσεων δεδομένων για την IA64 αρχιτεκτονική της Intel.

Η τελευταία έκδοση του Microsoft Sql Server είναι η 2008, η οποία δόθηκε στην αγορά τον Αύγουστο του 2008.

2.3 *Μηχανισμός ασφάλειας*

Ο μηχανισμός ασφάλειας του Sql Server επεκτείνεται σε δύο λογικά επίπεδα. Το πρώτο επίπεδο είναι το επίπεδο του server, ενώ το δεύτερο αναφέρεται σε μία συγκεκριμένη βάση δεδομένων του Sql Server.

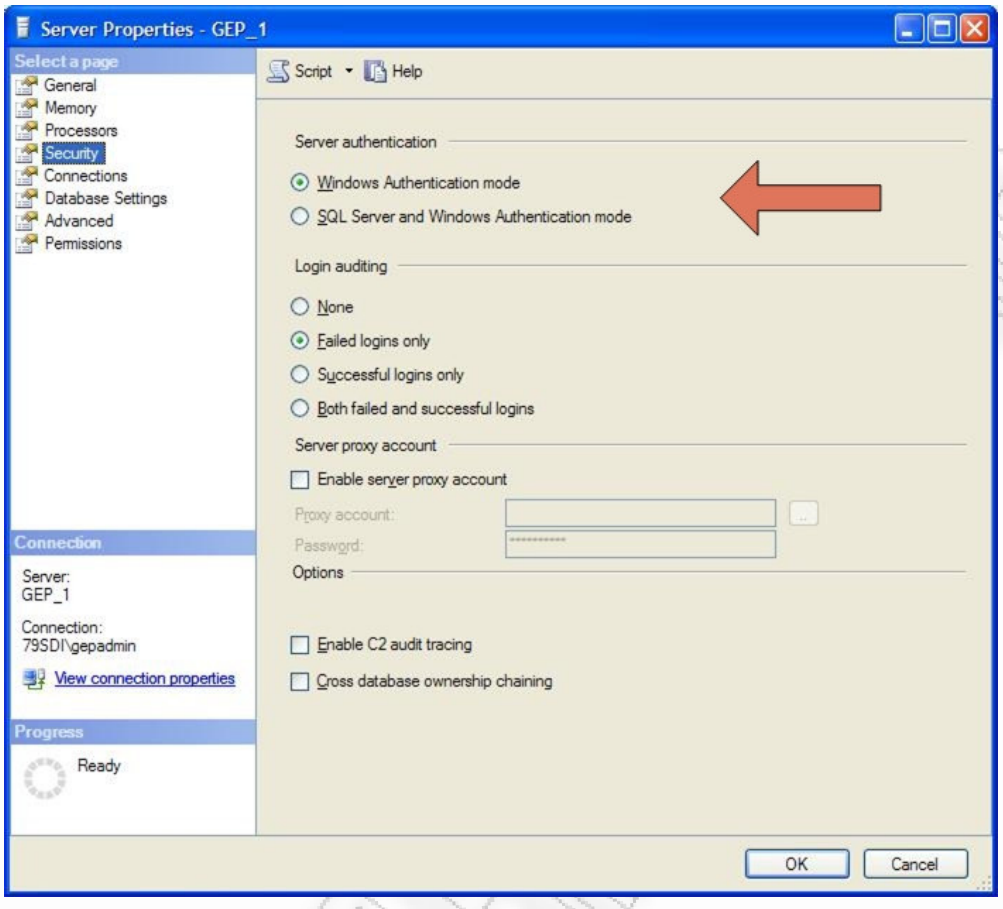
2.3.1 Αυθεντικοποίηση (επίπεδο server)

Η αυθεντικοποίηση πραγματοποιείται στο επίπεδο του Sql Server.

Υποστηρίζονται δύο τρόποι αυθεντικοποίησης:

Windows Authentication Mode (Αυθεντικοποίηση Windows): Με τον τρόπο αυτό αυθεντικοποίησης δε χρειάζεται να δοθούν όνομα χρήστη και κωδικός για τη σύνδεση στον Sql Server. Άντ' αυτού η σύνδεση στον Sql Server πραγματοποιείται διαφανώς μέσω του Windows NT/2000 λογαριασμού (ή του group στο οποίο ανήκει ένας λογαριασμός), που χρησιμοποιήθηκε για τη σύνδεση στο λειτουργικό σύστημα των Windows του συστήματος από το οποίο επιχειρείται η σύνδεση στον Sql Server. Ανάλογα με το αν έχει δοθεί ή όχι δικαίωμα πρόσβασης στον Sql Server για το λογαριασμό (ή το group στο οποίο ανήκει ο λογαριασμός) που επιχειρεί να αυθεντικοποιηθεί, του επιτρέπεται ή όχι η σύνδεση στον Sql Server.

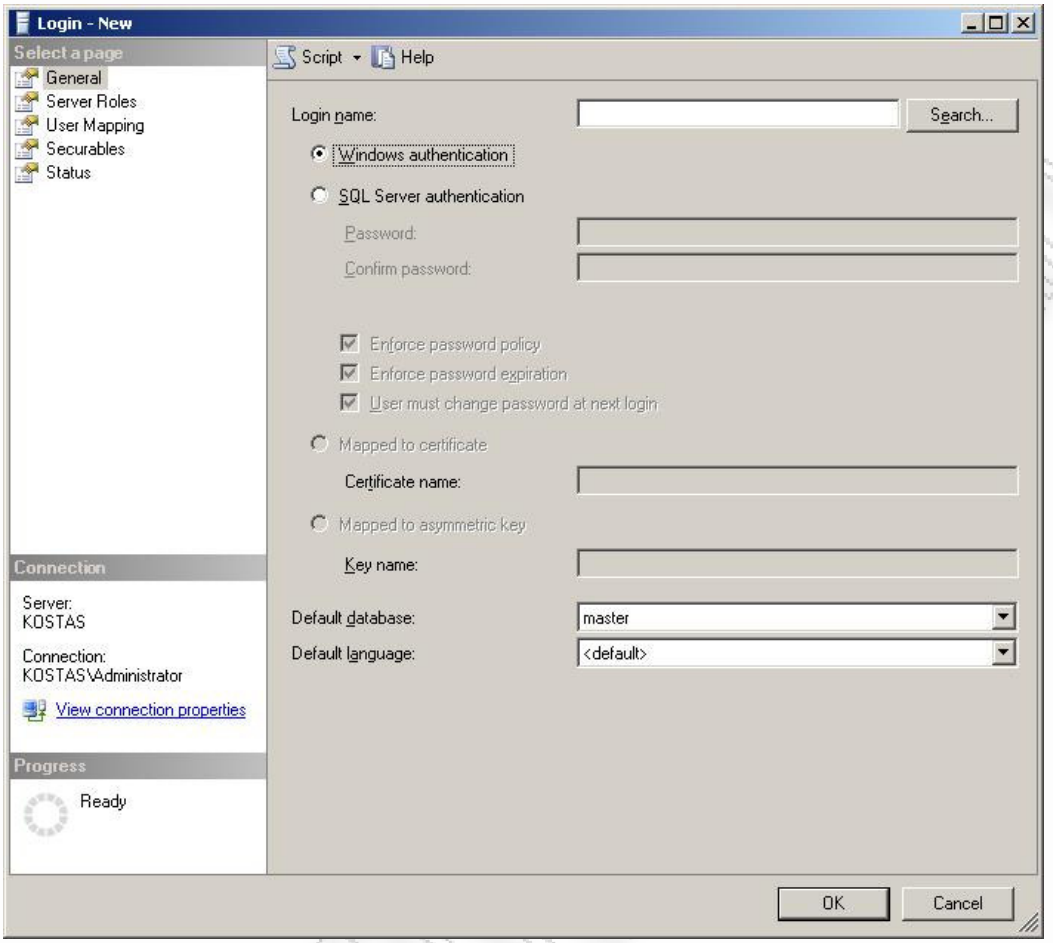
Mixed Mode Authentication (Μεικτή αυθεντικοποίηση): Με αυτό τον τρόπο αυθεντικοποίησης, οι χρήστες μπορούν να συνδεθούν είτε χρησιμοποιώντας τη Windows authentication είτε με την αυθεντικοποίηση που παρέχει ο ίδιος ο Sql Server. Πριν τη χρησιμοποίηση του συγκεκριμένου τρόπου αυθεντικοποίησης πρέπει να έχουν δημιουργηθεί έγκυροι Sql Server λογαριασμοί και κωδικοί. Αυτοί οι λογαριασμοί δε σχετίζονται με τους Windows NT/2000 λογαριασμούς και χειρίζονται από τον ίδιο τον Sql Server. Κατά τη διαδικασία εγκατάστασης του Sql Server δίνεται η δυνατότητα να δημιουργηθεί ο default Sql Server λογαριασμός με όνομα χρήστη sa και κωδικό της επιλογής του χρήστη που πραγματοποιεί την εγκατάσταση. Εφόσον είναι ενεργοποιημένος αυτός ο τρόπος αυθεντικοποίησης, τότε κατά την προσπάθεια σύνδεσης ελέγχεται αν το παρεχόμενο όνομα χρήστη και κωδικός ανήκουν σε κάποιον από τους Sql Server λογαριασμούς που έχουν δημιουργηθεί. Εάν όχι, τα παρεχόμενα αυτά διαπιστευτήρια ελέγχονται με τη βοήθεια του λειτουργικού συστήματος για το αν είναι έγκυρα απέναντι στη Windows authentication.



Εικόνα 1 Οθόνη καθορισμού του τρόπου λειτουργίας της αυθεντικοποίησης στον Sql Server 2005, δηλαδή επιλογή μεταξύ "Windows Authentication mode" και "SQL Server and Windows Authentication mode (Mixed mode)"

Συμπερασματικά, για τη σύνδεση στον Sql Server απαιτείται ένα έγκυρο login. Ένα έγκυρο login μπορεί να είναι:

- Ένας Windows NT/2000 λογαριασμός στον οποίο έχει δοθεί δικαίωμα πρόσβασης στον Sql Server είτε άμεσα μέσω του ονόματος του λογαριασμού είτε μέσω κάποιου από τα groups στα οποία ανήκει.
- Ένας Sql Server λογαριασμός ο οποίος διαχειρίζεται από τον ίδιο τον Sql Server (έχει ισχύ μόνο όταν είναι ενεργή η Mixed Mode Authentication)



Εικόνα 2 Οθόνη δημιουργίας ενός νέου login

Η Windows authentication είναι αυτή που προτείνεται, καθώς είναι περισσότερο ασφαλής και δεν απαιτεί τη διακίνηση ονομάτων χρηστών και κωδικών μέσω του δικτύου. Η Mixed Mode Authentication πρέπει να αποφεύγεται, εκτός των περιπτώσεων που τα περιβάλλοντα των clients που χρησιμοποιούν τον Sql Server δεν είναι Windows NT/2000 ή της περίπτωσης που χρειάζεται συμβατότητα με παλαιότερες εφαρμογές.

2.3.2 Εξουσιοδότηση (επίπεδο server)

Από τη στιγμή που ένας χρήστης αυθεντικοποιηθεί επιτυχώς, οι ενέργειες που έχει δικαίωμα να εκτελέσει στο επίπεδο του server καθορίζονται από τις ρυθμίσεις που έχουν γίνει στο σχήμα εξουσιοδότησης του Sql Server. Το σχήμα εξουσιοδότησης του Sql

Server καθορίζεται αποκλειστικά από τη συμμετοχή ή όχι του αυθεντικοποιημένου χρήστη στους προκαθορισμένους server ρόλους (Fixed Server Roles):

Οι προκαθορισμένοι server ρόλοι είναι ρόλοι που εφαρμόζονται στο επίπεδο του server. Σε αυτούς τους μπορούν να προστεθούν logins, έτσι ώστε να αποκτήσουν τα συσχετισμένα με το συγκεκριμένο ρόλο δικαιώματα. Οι προκαθορισμένοι server ρόλοι δεν μπορούν να τροποποιηθούν, ενώ επίσης δεν επιτρέπεται η δημιουργία νέων server ρόλων. Παρακάτω παρατίθενται οι προκαθορισμένοι server ρόλοι και τα συσχετισμένα με αυτούς δικαιώματα.

Server ρόλος	Περιγραφή
sysadmin	Μπορεί να πραγματοποιήσει οποιαδήποτε ενέργεια στον Sql Server.
serveradmin	Μπορεί να διαχειριστεί τις ρυθμίσεις του server ή και να σταματήσει το server.
setupadmin	Μπορεί να διαχειριστεί τους linked servers[1] και να εκτελέσει stored procedures του συστήματος.
securityadmin	Μπορεί να διαχειρίζεται τα logins, και τα δικαιώματα δημιουργίας βάσεων δεδομένων, επίσης να διαβάζει τα error logs και να αλλάζει τα κωδικούς των logins και των user accounts (2.3.3).
processadmin	Μπορεί να διαχειρίζεται διεργασίες (processes) που τρέχουν για λογαριασμό του Sql Server.
dbcreator	Μπορεί να δημιουργεί, να τροποποιεί και να διαγράφει βάσεις δεδομένων.

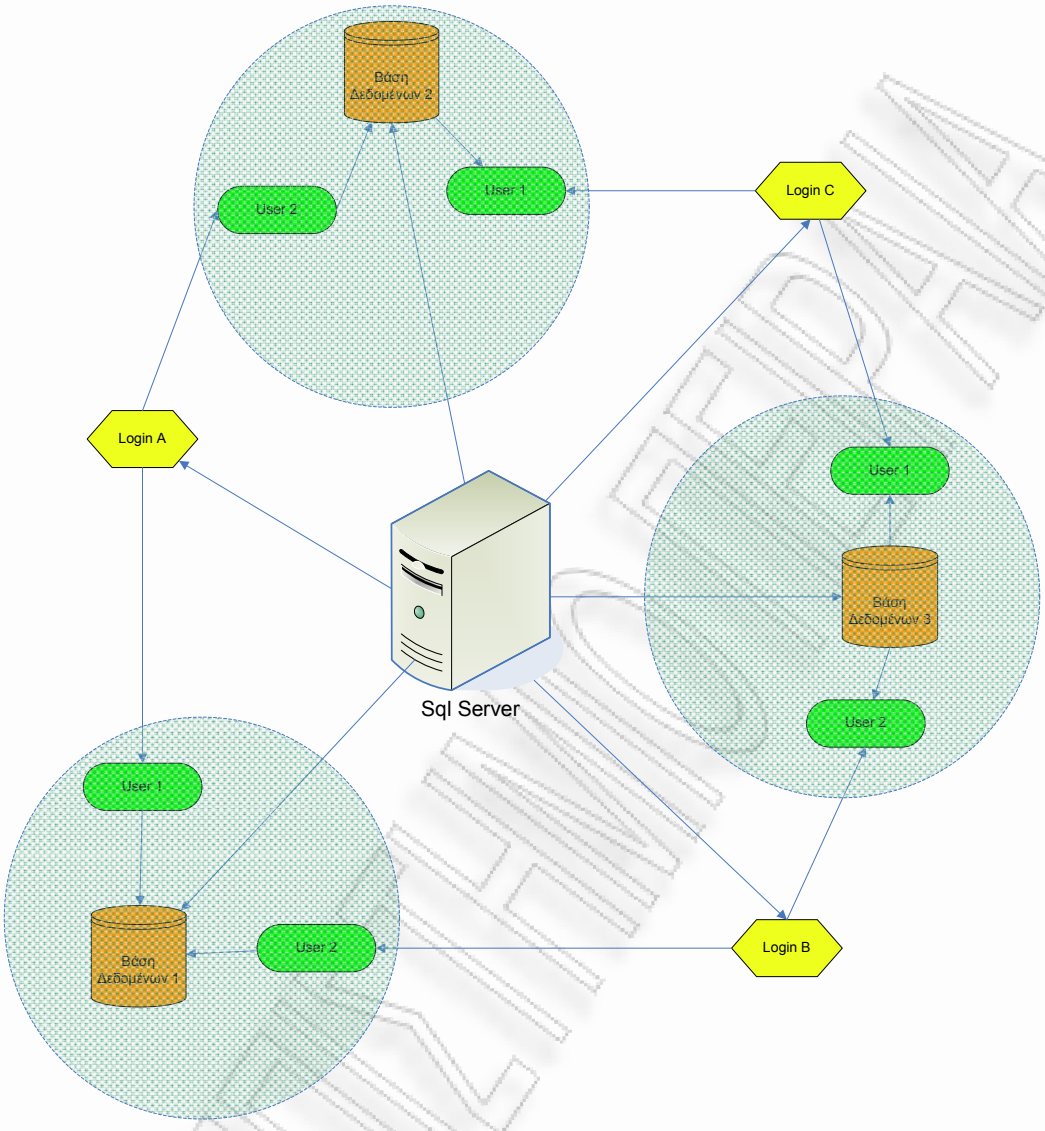
[1] Linked server – Λειτουργία του Sql Server που επιτρέπει την πρόσβαση σε εξωτερικές πηγές δεδομένων με τον ίδιο τρόπο που γίνεται η πρόσβαση σε μία Sql Server βάση δεδομένων. Επιτρέπει τη σύνδεση ακόμη και με βάσεις δεδομένων διαφορετικών Σ.Δ.Β.Δ. (όπως π.χ. Oracle, DB2 κλπ)

diskadmin	Μπορεί να χειρίζεται τα αρχεία του δίσκου που χρησιμοποιεί ο Sql Server.
bulkadmin	Μπορεί να εκτελεί BULK INSERT εντολές, δηλαδή εντολές μαζικής εισαγωγής δεδομένων από εξωτερικές πηγές.

2.3.3 Εξουσιοδότηση (επίπεδο βάσης δεδομένων)

Η εξουσιοδότηση ενός αυθεντικοποιημένου χρήστη στο επίπεδο μιας συγκεκριμένης βάσης δεδομένων γίνεται με τη βοήθεια των user accounts, των προκαθορισμένων ρόλων βάσης δεδομένων και των ρόλων εφαρμογής:

User account: Για την πρόσβαση σε μία βάση δεδομένων απαιτείται ένα έγκυρο user account. Τα user accounts είναι ειδικά για μία και μόνο βάση δεδομένων. Όλα τα δικαιώματα (permissions) καθώς και τα ιδιοκτησίες (ownerships) των αντικειμένων στη βάση δεδομένων ελέγχονται από το user account. Το default user account που δημιουργείται αυτόματα με τη δημιουργία μιας νέας βάσης δεδομένων είναι το dbo, το οποίο έχει πλήρη πρόσβαση σε όλα τα αντικείμενα αυτής της βάσης δεδομένων. Τα logins του Sql Server συσχετίζονται με τα user accounts. Ένα login μπορεί να έχει συσχετιζόμενα με αυτό user accounts σε διαφορετικές βάσεις δεδομένων, αλλά μόνο ένα user account για κάθε βάση δεδομένων.



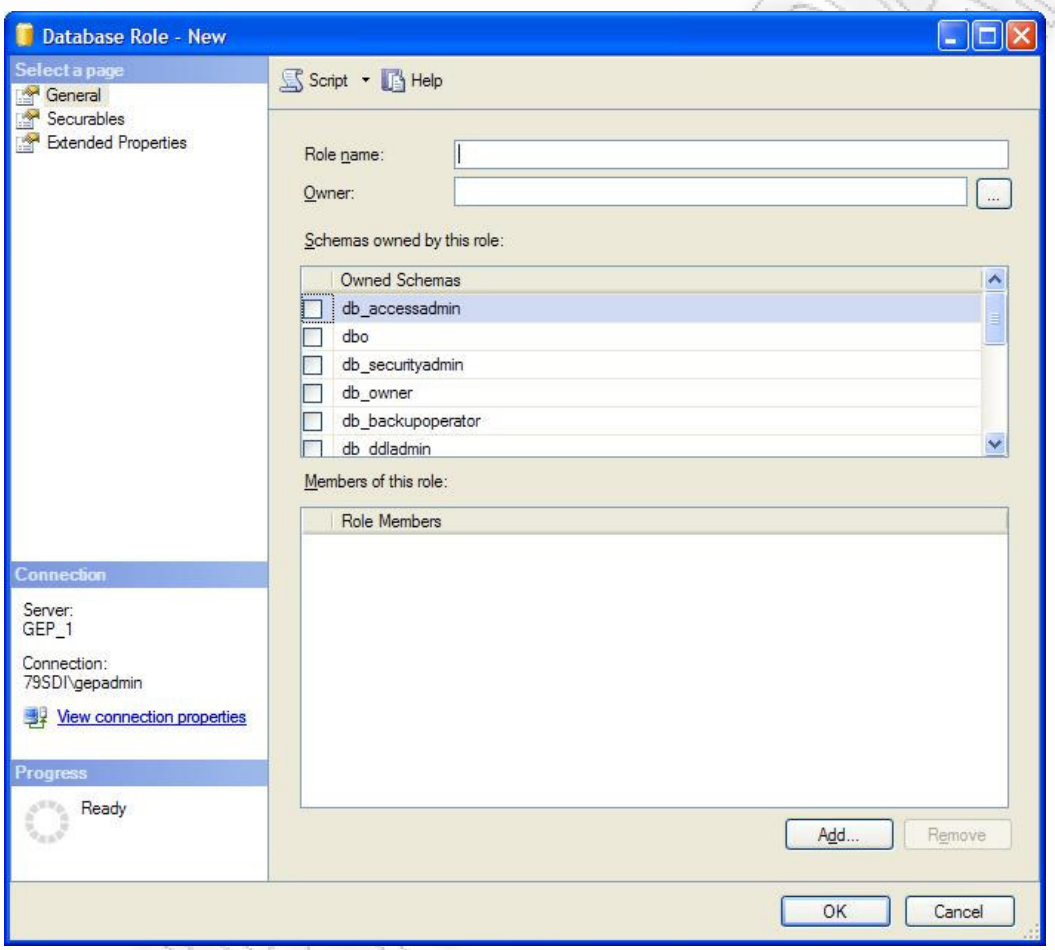
Εικόνα 3 Παράδειγμα σχήματος ασφαλείας του Sql Server

Προκαθορισμένοι ρόλοι βάσης δεδομένων (Fixed database roles): Κάθε βάση δεδομένων έχει ένα σύνολο προκαθορισμένων ρόλων, σε κάθε έναν από τους οποίους μπορούν να προστεθούν user accounts. Αυτοί οι προκαθορισμένοι ρόλοι είναι μοναδικοί για μία βάση δεδομένων. Παρά το ότι οι προκαθορισμένοι ρόλοι βάσης δεδομένων δεν μπορούν να τροποποιηθούν υπάρχει δυνατότητα δημιουργίας νέων ρόλων βάσης δεδομένων. Στη συνέχεια παρατίθενται οι προκαθορισμένοι ρόλοι βάσης δεδομένων του Sql Server:

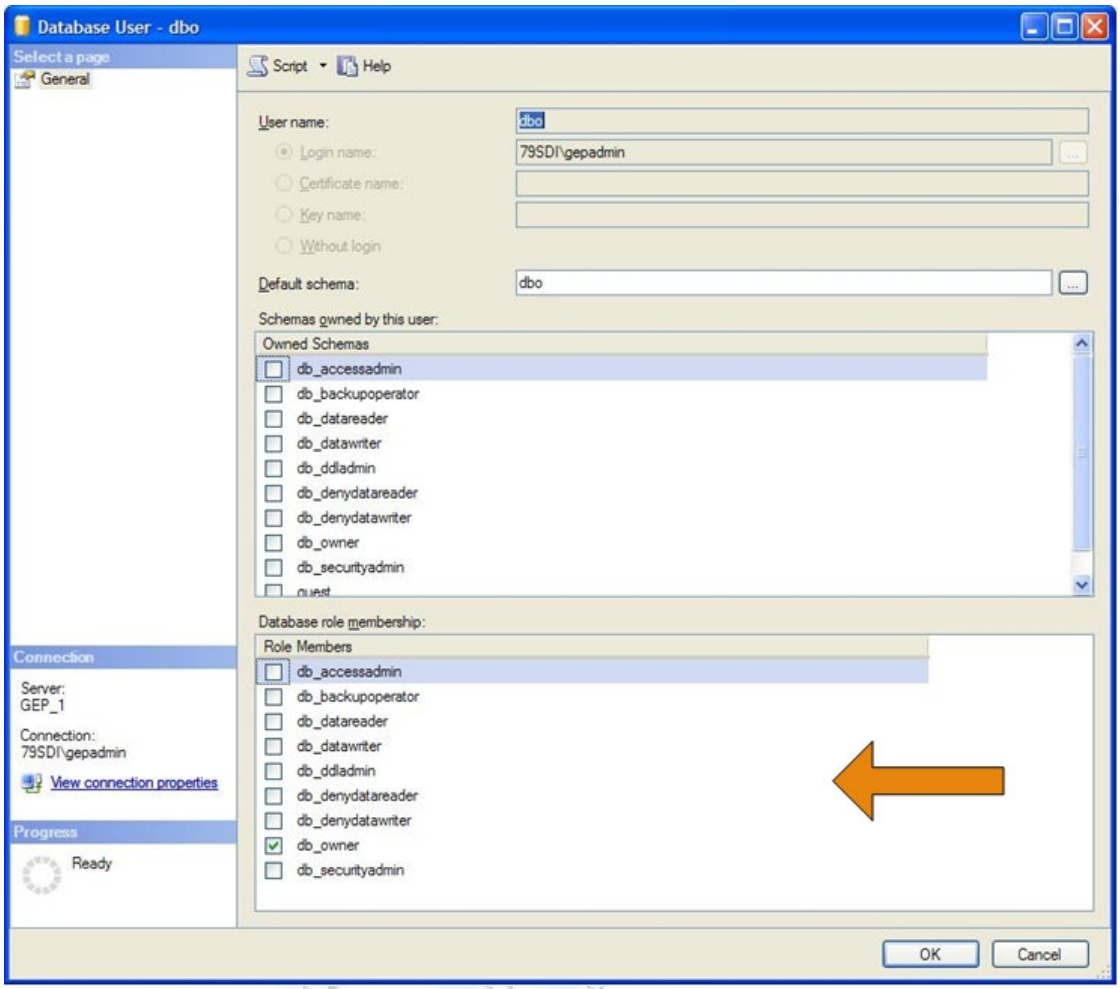
Database ρόλος	Περιγραφή
db_owner	Έχει όλα τα δικαιώματα για τη συγκεκριμένη βάση δεδομένων.
db_accessadmin	Μπορεί να διαχειριστεί την πρόσβαση Windows λογαριασμών, Windows groups και Sql Server λογαριασμών στη συγκεκριμένη βάση δεδομένων.
db_securityadmin	Μπορεί να διαχειρίζεται όλα τα δικαιώματα, τους ρόλους και τις συμμετοχές σε κάθε ρόλο
db_ddladmin	Μπορεί να εκτελέσει DDL (Data Definition Language) εντολές δηλαδή π.χ. CREATE, ALTER και DROP πάνω στα αντικείμενα της βάσης δεδομένων, αλλά δεν μπορεί να διαχειριστεί δικαιώματα και ρόλους ούτε και να έχει πρόσβαση στα δεδομένα της βάσης δεδομένων.
db_backupoperator	Μπορεί να εκτελεί εντολές σχετικές με το backup της βάσης δεδομένων. Δεν έχει πρόσβαση στα αντικείμενα της βάσης δεδομένων ούτε και στα δικαιώματα και τους ρόλους.
db_datareader	Μπορεί να έχει πρόσβαση ανάγνωσης σε όλα τα δεδομένα οποιουδήποτε πίνακα χρήστη[2] στη βάση δεδομένων.
db_datawriter	Μπορεί να τροποποιεί όλα τα δεδομένα σε οποιοδήποτε πίνακα χρήστη στη βάση δεδομένων.
db_denydatareader	Δεν έχει δικαίωμα ανάγνωσης από κανένα

[2] Εκτός από τους πίνακες μιας βάσης δεδομένων που χρησιμοποιούνται για την αποθήκευση των δεδομένων (πίνακες χρήστη) υπάρχουν και άλλοι πίνακες που εξυπηρετούν λειτουργίες της ίδιας της βάσης δεδομένων (πίνακες συστήματος)

	πίνακα χρήστη στη βάση δεδομένων.
db_denydatawriter	Δεν έχει δικαίωμα τροποποίησης των δεδομένων σε κανένα πίνακα χρήστη στη βάση δεδομένων.



Εικόνα 4 Οθόνη δημιουργίας νέου database ρόλου



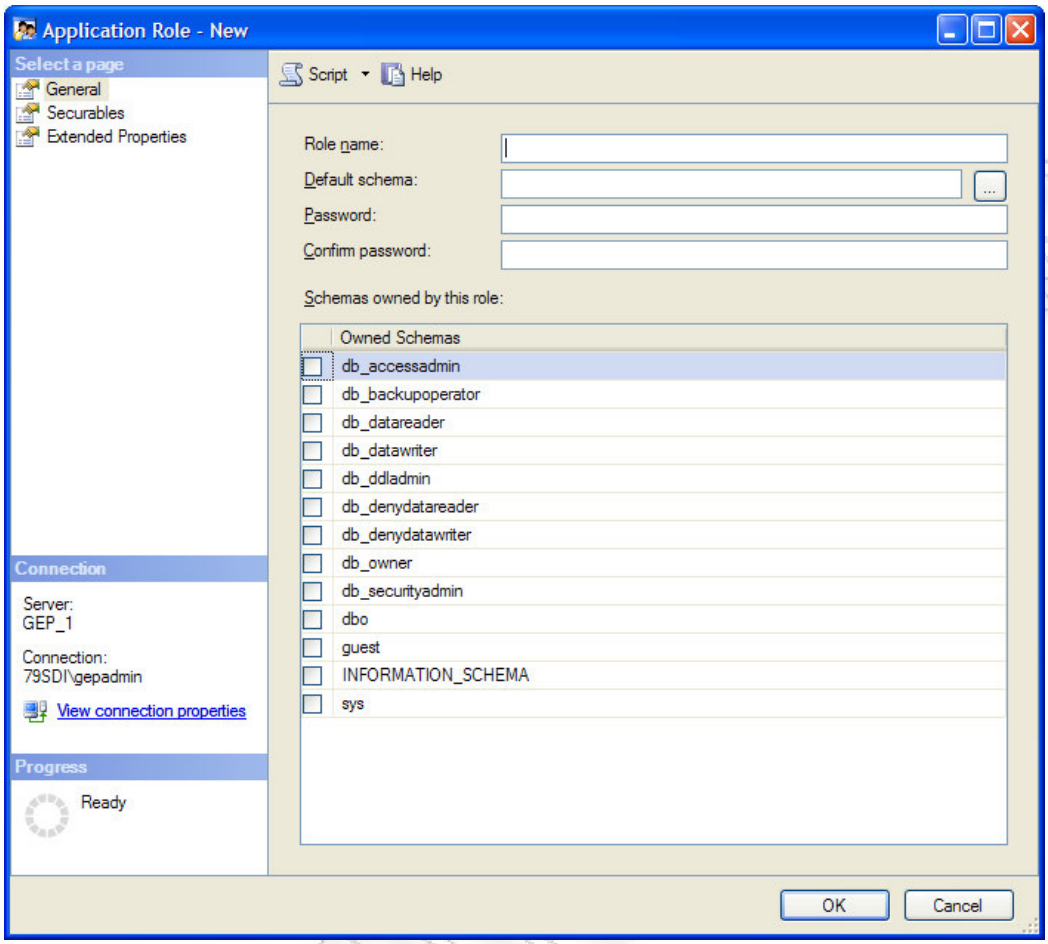
Εικόνα 5 Οθόνη ιδιοτήτων ενός user account μιας βάσης δεδομένων στον Sql Server 2005 στην οποία μπορούν να οριστούν οι database ρόλοι στους οποίους ανήκει

Ρόλοι εφαρμογής (Application roles): Οι ρόλοι εφαρμογής αποτελούν ένα ακόμη τρόπο για την υλοποίηση της ανάθεσης των δικαιωμάτων πρόσβασης. Διαφέρουν σε πολύ μεγάλο βαθμό από τους ρόλους server και βάσης δεδομένων. Μετά τη δημιουργία ενός ρόλου εφαρμογής και την ανάθεση δικαιωμάτων σε αυτόν, η εφαρμογή πρέπει να ενεργοποιήσει το ρόλο αυτό σε χρόνο εκτέλεσης για να αποκτήσει τα δικαιώματα που σχετίζονται με αυτόν. Οι ρόλοι εφαρμογής απλοποιούν τη δουλειά των διαχειριστών βάσεων δεδομένων, αφού πλέον δεν έχουν να ανησυχούν για τη διαχείριση των δικαιωμάτων σε επίπεδο χρήστη αλλά μόνο σε επίπεδο

εφαρμογής. Το μόνο που χρειάζεται να κάνουν είναι να δημιουργήσουν έναν ρόλο εφαρμογής και να αναθέσουν δικαιώματα σε αυτόν. Η εφαρμογή που συνδέεται στη βάση δεδομένων ενεργοποιεί το ρόλο εφαρμογής και κληρονομεί τα δικαιώματα που σχετίζονται με το ρόλο αυτό. Τα κύρια χαρακτηριστικά των ρόλων εφαρμογής είναι τα εξής:

- Δεν υπάρχουν ενσωματωμένοι / προκαθορισμένοι ρόλοι εφαρμογής
- Οι ρόλοι εφαρμογής δεν έχουν μέλη
- Οι ρόλοι εφαρμογής ενεργοποιούνται σε χρόνο εκτέλεσης από την εφαρμογή παρέχοντας τον κωδικό με το οποίο έχει δημιουργηθεί ο χρησιμοποιούμενος ρόλος εφαρμογής.
- Οι ρόλοι εφαρμογής επικαλύπτουν τα στάνταρ δικαιώματα χρήσης. Για παράδειγμα, μετά την ενεργοποίηση του ρόλου εφαρμογής, η εφαρμογή χάνει όλα τα δικαιώματα που είναι σχετισμένα με το login/user που χρησιμοποιήθηκε για την σύνδεση στον Sql Server και αποκτά τα δικαιώματα που σχετίζονται με τον ρόλο εφαρμογής.
- Οι ρόλοι εφαρμογής είναι ειδικοί για μία βάση δεδομένων. Μετά την ενεργοποίηση του ρόλου εφαρμογής για μία βάση δεδομένων, εάν η εφαρμογή επιθυμεί να εκτελέσει μια δοσοληψία που απαιτεί σύνδεση σε άλλη βάση δεδομένων (cross-database transaction), τότε αυτή η βάση δεδομένων πρέπει να έχει ενεργοποιημένο ένα guest user account[3].

[3] Ειδικό user account με περιορισμένα δικαιώματα μέσω του οποίου ένα επιτυχώς αυθεντικοποιημένο login μπορεί να έχει πρόσβαση σε οποιαδήποτε βάση δεδομένων του ίδιου Sql Server με την οποία δε συσχετίζεται μέσω κάποιου άλλου user account.



Εικόνα 6 Οθόνη δημιουργίας ενός νέου application ρόλου

Η ανάθεση / αφαίρεση δικαιωμάτων σε user accounts, ρόλους βάσης δεδομένων και ρόλους εφαρμογής γίνεται μέσω των ακόλουθων T-SQL εντολών που χρησιμοποιούνται για τη διαχείριση των δικαιωμάτων σε επίπεδο users και ρόλων:

- GRANT: Παραχωρεί το συγκεκριμένο δικαίωμα (π.χ. SELECT, DELETE κλπ) στο συγκεκριμένο user account ή ρόλο στη συγκεκριμένη βάση δεδομένων. Π.χ. η εντολή:

```
GRANT SELECT ON authors TO Alex
```

δίνει δικαίωμα στο user account Alex να πραγματοποιεί SELECT εντολές στον πίνακα authors.

- REVOKE: Αφαιρεί ένα προηγουμένως παραχωρημένο δικαίωμα ή άρει την απαγόρευση για ένα αφαιρεμένο δικαίωμα από ένα user account ή ρόλο στη συγκεκριμένη βάση δεδομένων. Π.χ. η εντολή:

```
REVOKE SELECT ON authors TO Alex
```

αφαιρεί το δικαίωμα SELECT από τον user account Alex στον πίνακα authors.

- DENY: Απαγορεύει τη χρήση ενός συγκεκριμένου δικαιώματος σε ένα συγκεκριμένο user account στη συγκεκριμένη βάση δεδομένων. Για παράδειγμα η εντολή:

```
DENY SELECT ON authors TO Alex
```

δεν επιτρέπει στο user account Alex να εκτελεί SELECT στον πίνακα authors.

Αξίζει να σημειωθεί ότι η διαχείριση των δικαιωμάτων μπορεί να γίνει ακόμη και σε πολύ χαμηλό επίπεδο όπως είναι τα columns των πινάκων μίας βάσης δεδομένων.

Παρ' όλα αυτά στον Sql Server δεν υπάρχει δυνατότητα διαχείρισης δικαιωμάτων σε επίπεδο γραμμών (rows). Αυτό σημαίνει ότι δεν υπάρχει δυνατότητα π.χ. να δοθεί δικαίωμα SELECT μιας συγκεκριμένης row στον User1 και να αποκλειστεί το δικαίωμα SELECT σε μια άλλη γραμμή στον User2. Αυτή η πτυχή ασφάλειας μπορεί να υλοποιηθεί χρησιμοποιώντας αποτελεσματικά views και stored procedures.

2.4 *Πρακτικές μεγιστοποίησης ασφάλειας*

Στον Sql Server το επίπεδο ασφάλειας είναι σε μεγάλο βαθμό διαμορφώσιμο, δηλαδή εξαρτάται από τις ρυθμίσεις ασφαλείας που έχουν επιλεγεί από το διαχειριστή του. Είναι επομένως εξαιρετικά σημαντικό να αναλυθούν κάποιοι ενδιαφέροντες παράγοντες διαμόρφωσης της ασφάλειας στον Sql Server.

2.4.1 Χρήση Windows authentication

Όπως αναφέρθηκε και προηγουμένως, ο Sql Server υποστηρίζει δύο τρόπους αυθεντικοποίησης: τη Windows Authentication και τη Mixed Mode Authentication. Η Windows Authentication θεωρείται περισσότερο ασφαλής από την Mixed Mode Authentication και επομένως είναι προτιμότερο να εφαρμόζεται στην πράξη. Με τη χρήση Windows authentication αποφεύγεται η διακίνηση κωδικών καθαρού κειμένου στο δίκτυο. Επίσης, δεν υπάρχει ανάγκη αποθήκευσης κωδικών στα configuration αρχεία που χρησιμοποιούνται από τις εφαρμογές που χρησιμοποιούν τον Sql Server, όπως αναγκαστικά πρέπει να γίνει όταν χρησιμοποιείται η Mixed mode authentication. Η αποθήκευση κωδικών καθαρού κειμένου σε configuration αρχεία μπορεί να μην ενέχει άμεσους κινδύνους ασφάλειας, αλλά γίνεται εξαιρετικά επικίνδυνη όταν για παράδειγμα κάποιος επιτιθέμενος αποκτήσει πρόσβαση, με κάποιο τρόπο, στο configuration αρχείο μιας εφαρμογής.

Παρακάτω παρουσιάζονται δύο αποσπάσματα από configuration αρχεία μιας τυχαίας εφαρμογής που χρησιμοποιεί τον Sql Server. Στην πρώτη περίπτωση χρησιμοποιείται σύνδεση σε Sql Server που χρησιμοποιεί Windows authentication, ενώ στη δεύτερη σε Sql Server με Mixed mode authentication:

```
Server=myServerName\theInstanceName;Database=myDataBase;Truste
d_Connection=True;
```

(Sql Server με Windows Authentication)

```
Data Source=myServerAddress;Initial Catalog=myDataBase;User
Id=myUsername;Password=myPassword;
```

(Sql Server με Mixed Mode Authentication)

2.4.2 Συχνός έλεγχος των δικαιωμάτων χρηστών και groups

Είναι εξαιρετικά σημαντικό να γίνεται τακτικός έλεγχος των δικαιωμάτων που είναι ανατεθειμένα σε χρήστες και groups, έτσι ώστε να εξασφαλίζεται ότι κανένας χρήστης (login ή user account) και κανένα group δεν έχει περισσότερα δικαιώματα από αυτά που πραγματικά χρειάζεται.

Ένα παράδειγμα της συγκεκριμένης περίπτωσης παρουσιάζεται όταν ένας χρήστης αλλάζει επιχειρησιακό ρόλο μέσα σε μία εταιρεία που χρησιμοποιεί μια κοινή βάση δεδομένων. Αν δηλαδή ένας χρήστης μετακινηθεί, μέσα σε μια εταιρεία, από ένα τμήμα σε ένα άλλο τα δικαιώματα πρόσβασής του στη βάση δεδομένων πρέπει να τροποποιηθούν κατάλληλα αμέσως με την αλλαγή του τμήματος, διαφορετικά θα έχει τα δικαιώματα πρόσβασης που αντιστοιχούν στο προηγούμενο τμήμα και όχι στο καινούριο. Κάτι τέτοιο σαφώς εγκυμονεί κινδύνους ασφάλειας όσον αφορά τη βάση δεδομένων της εταιρείας.

2.4.3 Χρήση «ισχυρού» κωδικού[4] για το login sa

Η χρήση Mixed Mode Authentication προϋποθέτει την ύπαρξη του ειδικού login sa (system administrator) που έχει πλήρη δικαιώματα σε όλο το server. Η επιλογή ενός μη ισχυρού κωδικού για το login sa μπορεί να προκαλέσει προβλήματα ασφάλειας. Δεν είναι λίγα τα παραδείγματα στα οποία servers έχουν δεχθεί επιθέσεις επειδή δεν είχε καθοριστεί καθόλου κωδικός για το login sa ή είχε καθοριστεί ένας πολύ «ασθενής» κωδικός.

[4] Ισχύς κωδικού – Μέτρο αποτελεσματικότητας ενός κωδικού όσον αφορά την ανθεκτικότητά του σε επιθέσεις συνεχών προσπαθειών για την ανάκτησή του

2.4.4 Χρήση stored procedures για υλοποίηση μηχανισμού ασφάλειας

Όταν μία εφαρμογή απαιτεί πρόσβαση σε μία βάση δεδομένων, τότε αντί της προσέγγισης να δημιουργηθεί ένα user account που να έχει πλήρη απευθείας πρόσβαση σε όλους τους πίνακες και views που χρειάζεται η εφαρμογή είναι προτιμότερο να δημιουργηθούν stored procedures που να πραγματοποιούν τις επιθυμητές από την εφαρμογή λειτουργίες και να δοθούν τα απαραίτητα δικαιώματα σε αυτές και μόνο σε αυτές τις stored procedures. Με τον τρόπο αυτό δημιουργείται ένα απόλυτα ελεγχόμενο σχήμα ασφάλειας, το οποίο περιορίζει σημαντικά τις πιθανότητες επιθέσεων στην περίπτωση δημόσια προσβάσιμων εφαρμογών.

2.4.5 Απενεργοποίηση των Sql Mail δυνατοτήτων

Ο Sql Server παρέχει τη δυνατότητα αποστολής και λήψης e-mails μέσω μιας ενσωματωμένης λειτουργίας που ονομάζεται Sql Mail.
Η ενεργοποίηση των Sql Mail δυνατοτήτων δίνει μια ακόμη δυνατότητα επιθέσεων με τη μορφή λήψης trojans[5], ιών ή με τη μορφή επίθεσης της μορφής "denial of service"[6]. Από μόνες τους οι Sql Mail δυνατότητες είναι ακίνδυνες αλλά μπορούν να δώσουν μια νέα διέξοδο για επιθέσεις, οπότε στην περίπτωση που δε πρόκειται να χρησιμοποιηθούν είναι προτιμότερη η απενεργοποίησή τους.

---

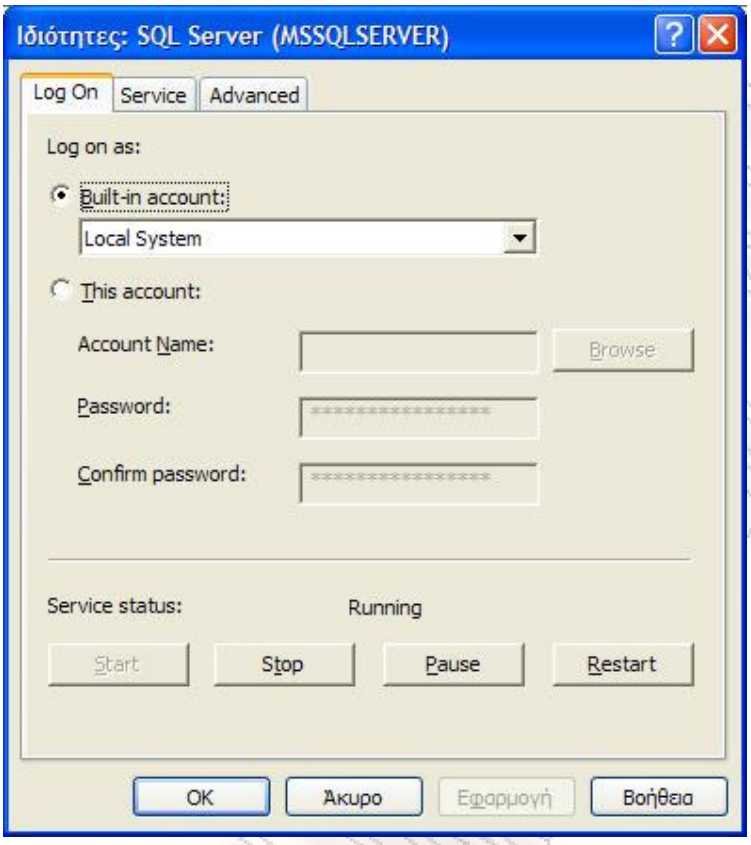
[5] Παραπλανητικό λογισμικό που εμφανίζεται να πραγματοποιεί μια επιθυμητή λειτουργία, αλλά στην πραγματικότητα έχει τελικό σκοπό τη μη εξουσιοδοτημένη πρόσβαση από κάποιον επιτιθέμενο στο σύστημα στο οποίο εκτελείται.

[6] Επίθεση της οποίας σκοπός είναι να καταστήσει ένα σύστημα μη διαθέσιμο για τους χρήστες για τους οποίους προορίζεται.

2.4.6 Περιορισμός των δικαιωμάτων των Sql Server services

Τόσο ο Sql Server όσο και ο Sql Server agent[7] εκτελούνται ως Windows services. Κάθε ένα από αυτά τα services πρέπει να σχετισθεί με ένα Windows λογαριασμό από τον οποίο θα κληρονομεί το σύνολο των δικαιωμάτων του. Ο Sql Server επιτρέπει στο login sa και σε κάποιες περιπτώσεις και σε άλλα logins να έχουν πρόσβαση σε λειτουργίες του λειτουργικού συστήματος. Η πρόσβαση σε αυτές τις λειτουργίες γίνεται με χρήση των δικαιωμάτων που κατέχει ο Windows λογαριασμός που σχετίζεται με το service του Sql Server. Σε μία ενδεχόμενη επιτυχημένη επίθεση, αυτές οι λειτουργίες θα μπορούσαν να χρησιμοποιηθούν έτσι ώστε να κατευθυνθεί η επίθεση σε οποιοδήποτε πόρο (resource) έχει πρόσβαση ο λογαριασμός που σχετίζεται με το service του Sql Server. Για το λόγο αυτό είναι σημαντικό να δίνονται μόνο τα απαραίτητα δικαιώματα στα services του Sql Server.

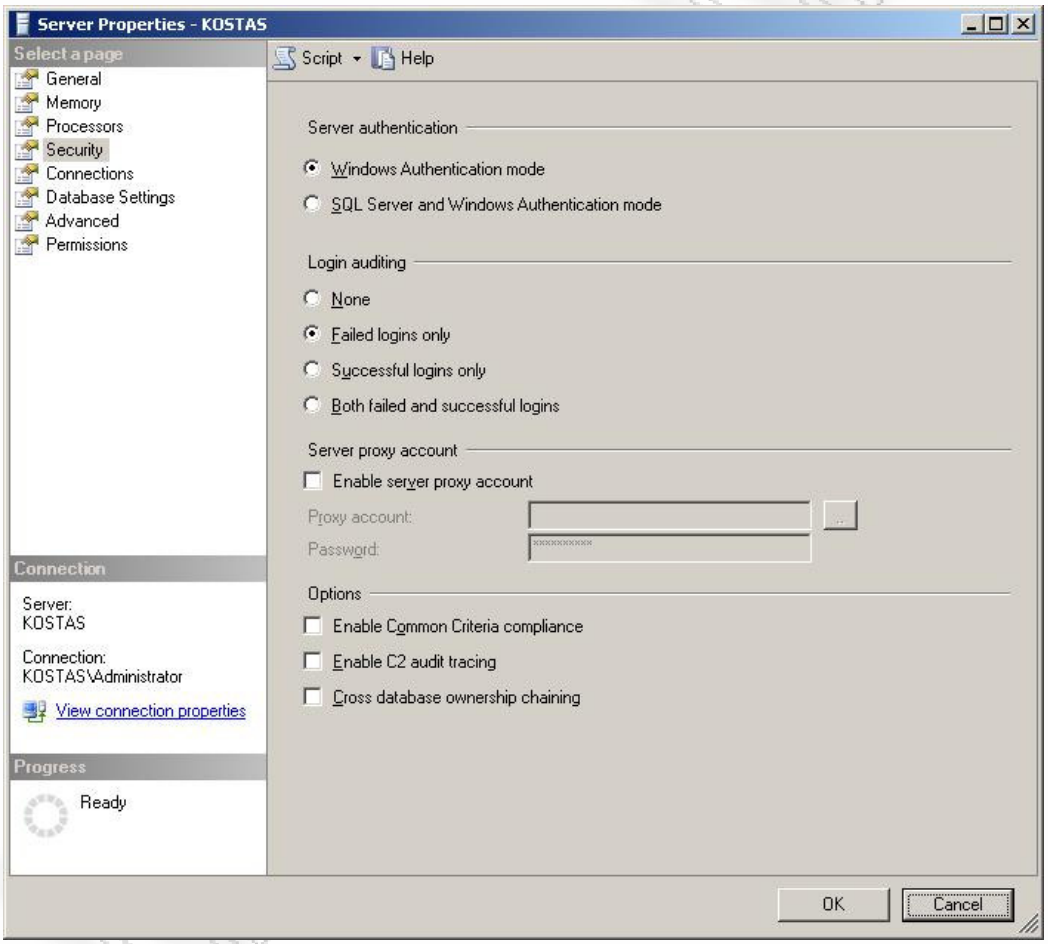
[7] Διεργασία του Sql Server η οποία εκτελεί εργασίες, «παρακολουθεί» τη λειτουργία του Sql Server και επιτρέπει την αυτοματοποίηση κάποιων διαχειριστικών εργασιών.



Εικόνα 7 Καθορισμός του Windows λογαριασμού που σχετίζεται με το SQL Server service μέσω του SQL Server Configuration Manager

2.4.7 Έλεγχος των συνδέσεων

Ο Sql Server έχει τη δυνατότητα να καταγράφει σε log files τις ενέργειες που πραγματοποιούνται κατά τη διάρκεια κάθε σύνδεσης σε αυτόν, έτσι ώστε να είναι δυνατή η αξιολόγηση τους από το διαχειριστή του συστήματος. Στο ελάχιστο, θα πρέπει οπωσδήποτε να καταγράφονται στα log files οι αποτυχημένες προσπάθειες σύνδεσης στον Sql Server και να ελέγχονται τακτικά. Οπότε αυτό είναι δυνατό θα πρέπει τα log files αυτά να μεταφέρονται σε διαφορετικό σκληρό δίσκο από τα αρχεία δεδομένων των βάσεων δεδομένων.



Εικόνα 8 Οθόνη που παρέχει τη δυνατότητα ρύθμισης των πληροφοριών που θα καταγράφονται στα log files σχετικά με τις συνδέσεις των χρηστών του Sql Server

2.4.8 Χρήση του ασφαλέστερου συστήματος αρχείων

Το σύστημα αρχείων NTFS θεωρείται ως το καταλληλότερο σύστημα αρχείων για τις εγκαταστάσεις του Sql Server. Είναι σταθερότερο, ανακτάται ευκολότερα από το σύστημα FAT και παρέχει δυνατότητες ασφάλειας όπως ACLs[8] αρχείων και καταλόγων και κρυπτογράφησης αρχείων (EFS[9]). Κατά τη διάρκεια της εγκατάστασης του Sql Server, εφόσον ανιχνευθεί η ύπαρξη NTFS συστήματος τίθενται αυτόματα τα κατάλληλα ACLs σε registry keys και αρχεία. Αυτές οι προκαθορισμένες ρυθμίσεις δεν συνίσταται να τροποποιούνται. Με το EFS τα αρχεία των βάσεων δεδομένων του Sql Server κρυπτογραφούνται κάτω από το Windows λογαριασμό έχει σχετισθεί με το service του Sql Server. Αν χρειαστεί για κάποιο λόγο να αλλάξει ο λογαριασμός αυτός, πρέπει πρώτα να αποκρυπτογραφηθούν τα αρχεία με τον παλιό λογαριασμό και να επανακρυπτογραφηθούν κάτω από τον καινούριο λογαριασμό.

2.4.9 Εγκατάσταση των τελευταίων service packs

Τα τελευταία service packs και security updates που προσφέρονται δωρεάν από τη Microsoft πρέπει να εγκαθίστανται όσο το δυνατό γρηγορότερα από την ημερομηνία που δίνονται στη δημοσιότητα καθώς πολλές φορές περιέχουν λύσεις και διορθώσεις σε σημαντικές αδυναμίες ασφάλειας του Sql Server.

---

[8] Access Control List – Λίστα δικαιωμάτων (π.χ. δικαίωμα ανάγνωσης, τροποποίησης) που σχετίζονται με ένα αντικείμενο (π.χ. αρχείο ή κατάλογο) του NTFS συστήματος αρχείων.

[9] Encrypting File System – Τεχνολογία του συστήματος αρχείων NTFS που επιτρέπει τη «διαφανή» κρυπτογράφηση αρχείων με σκοπό την προστασία των δεδομένων των αρχείων από επιτιθέμενους με φυσική πρόσβαση στο σκληρό δίσκο που φιλοξενεί τα αρχεία.

2.4.10 Κατάλληλες ρυθμίσεις στο firewall

Οι προκαθορισμένες θύρες που χρησιμοποιεί ο Sql Server είναι η 1433 για TCP συνδέσεις και η 1434 για UDP συνδέσεις. Το firewall του συστήματος που φιλοξενεί τον Sql Server, πρέπει να είναι ρυθμισμένο έτσι ώστε να μπλοκάρει πακέτα που κατευθύνονται προς τις θύρες αυτές, εάν αυτά τα πακέτα δεν προέρχονται από servers που έχουν το δικαίωμα επικοινωνίας με τον Sql Server. Τυχόν πρόσθετες θύρες που έχουν συσχετιστεί με named instances[10] του Sql Server πρέπει επίσης να μπλοκαριστούν.

2.4.11 Φυσική και λογική απομόνωση του Sql Server

Πέρα από τους παραμετροποιήσιμους μηχανισμούς ασφάλειας του Sql Server, είναι σημαντική η φυσική προστασία του. Η φυσική πρόσβαση στο σύστημα που φιλοξενεί τον Sql Server πρέπει να επιτρέπεται μόνο σε συγκεκριμένους χρήστες και επίσης το σύστημα αυτό να προστατεύεται π.χ. σε ένα κλειδωμένο server room. Στην ιδανική περίπτωση, το δωμάτιο αυτό θα πρέπει να είναι εφοδιασμένο με μηχανισμό ανίχνευσης πλημμύρας και ανίχνευσης / κατάσβεσης φωτιάς.

Οι βάσεις δεδομένων θα πρέπει να εγκαθίστανται στην ασφαλέστερη ζώνη ενός εταιρικού δικτύου και δεν πρέπει ποτέ να συνδέονται απευθείας με το Internet. Επίσης, είναι σημαντικό το συχνό backup των βάσεων δεδομένων και η αποθήκευση αντιγράφων σε σημεία περιορισμένης πρόσβασης.

---

[10] Παράλληλη εγκατάσταση Sql Server στο ίδιο σύστημα με την κύρια εγκατάσταση που χρησιμοποιεί διαφορετικές θύρες επικοινωνίας από την κύρια εγκατάσταση του Sql Server

2.5 *Διαπιστωμένα προβλήματα ασφάλειας και «έξυπνες» στρατηγικές επίθεσης*

Στο παρόν κεφάλαιο, θα αναλυθούν κάποια προβλήματα ασφαλείας που έχουν διαπιστωθεί στις διάφορες εκδόσεις του Sql Server και οφείλονται σε παραλείψεις της ομάδας ανάπτυξης με σκοπό να εντοπισθούν τα αδύνατά σημεία του Sql Server όσον αφορά την ασφάλεια. Επίσης θα παρουσιαστούν κάποιες «έξυπνες» μορφές επίθεσης που μπορεί να δεχθεί ο Sql Server που όμως δεν ευθύνονται σε αυτόν καθαυτόν.

2.5.1 Ανάλυση (parsing) παραμέτρων σε Extended stored procedures (Sql Server 2000)

Οι Extended Stored Procedures (XPs) είναι μια βιβλιοθήκη (dll) η οποία μπορεί να εγκατασταθεί στον Sql Server με σκοπό να του παρέχει εκτεταμένη λειτουργικότητα. Στο API[11] που παρέχεται από τον Sql Sever για το χειρισμό των παραμέτρων εισόδου των XPs, μία μέθοδος, η srv_paraminfo(), περιέχει μία αδυναμία η οποία μπορεί να οδηγήσει σε κατάσταση υπερχείλισης buffer. Το API είναι σχεδιασμένο, έτσι ώστε να εντοπίζει τη n-ιοστή παράμετρο σε ένα string και να την τοποθετεί σε ένα buffer που παρέχεται από την XP. Από σχεδιασμού, το API δεν παρέχει κάποιο τρόπο «ενημέρωσης» της XP για το μήκος του buffer που χρειάζεται, αντίθετα η XP θεωρείται ότι έχει εξασφαλίσει ότι το buffer θα είναι αρκετά μεγάλο, έτσι ώστε να αποθηκεύσει την παράμετρο. Παρ' όλα αυτά, τον έλεγχο αυτόν δεν τον πραγματοποιούν όλες οι XPs που παρέχονται by default στον Sql Server. Ένας επιτιθέμενος που έδωσε μια επαρκώς μεγάλη παράμετρο σε μία τέτοια XP, θα μπορούσε να προκαλέσει υπερχείλιση buffer μέσα στην srv_paraminfo(), με

[11] Application Programming Interface – Σύνολο ρουτινών, δομών δεδομένων, κλάσεων ή/και πρωτοκόλλων παρεχόμενες από βιβλιοθήκες ή/και λειτουργικά συστήματα με σκοπό να υποστηρίξουν την ανάπτυξη εφαρμογών.

αποτέλεσμα είτε να προκαλέσει αποτυχία (denial of service) στον Sql Server είτε να προκαλέσει εκτέλεση κώδικα της επιλογής του.
Υπάρχουν δύο σενάρια στα οποία ένας επιτιθέμενος θα μπορούσε να εκμεταλλευτεί αυτή την αδυναμία ασφάλειας:

- Θα μπορούσε να επιτεθεί σε έναν Sql Server, απευθείας, συνδεόμενος σε αυτόν και καλώντας μία XP. Παρ' όλα αυτά θα μπορούσε να πραγματοποιήσει κάτι τέτοιο μόνο εάν είχε προηγουμένως αυθεντικοποιηθεί επιτυχώς στον Sql Server.
- Εναλλακτικά, θα μπορούσε να προσπαθήσει να επιτεθεί σε έναν Sql Server που χρησιμοποιείται από μία web εφαρμογή παρέχοντας προσεκτικά επιλεγμένες εισόδους σε κάποιο σημείο που η web εφαρμογή απαιτεί είσοδο από το χρήστη. Παρ' όλα αυτά οι XPs χρησιμοποιούνται σπάνια από web εφαρμογές. Ακόμα και αν μία web εφαρμογή χρησιμοποιεί μία XP, θα χρειαζόταν ο επιτιθέμενος να έχει λεπτομερή γνώση του σχεδιασμού της εφαρμογής, ώστε να την τροφοδοτήσει με παραμέτρους τέτοιες που θα περάσουν σε μία XP, και στη συνέχεια στη srv_paraminfo με τρόπο τέτοιο που να γίνει εκμετάλλευση της συγκεκριμένης αδυναμίας ασφάλειας.

Ως αποτέλεσμα αυτών των περιορισμών, προκύπτει ότι αυτή η αδυναμία ασφάλειας θα ήταν περισσότερο χρήσιμη σε έναν επιτιθέμενο ο οποίος έχει ήδη καταφέρει να αποκτήσει πρόσβαση σε έναν web server και να έχει στη διάθεσή του ένα login στον back-end Sql Server. Εάν ο επιτιθέμενος παρ' όλα αυτά καταφέρει να τρέξει κώδικα της επιλογής του στον server, αυτός θα έτρεχε στο περιβάλλον ασφάλειας του Sql Server service λογαριασμού. Εάν ακολουθηθούν οι προτεινόμενες στρατηγικές, οι οποίες περιγράφηκαν και στο προηγούμενο κεφάλαιο, τότε ο λογαριασμός αυτός θα έχει μόνο κανονικά δικαιώματα απλού χρήστη στο σύστημα. Σε αυτή την περίπτωση, ο κώδικας του επιτιθέμενου θα μπορούσε να πραγματοποιήσει οποιαδήποτε επιθυμητή ενέργεια

απέναντι στη βάση δεδομένων, αλλά δε θα μπορούσε να κερδίσει διαχειριστικό έλεγχο στο σύστημα.

Η Microsoft παρέχει ένα patch διόρθωσης για την αδυναμία αυτή. Το patch αυτό λειτουργεί τροποποιώντας όλες τις default XPs, έτσι ώστε να εξασφαλίζουν ένα buffer σωστού μεγέθους πριν καλέσουν την srv_paraminfo(). Η προσέγγιση αυτή προτιμήθηκε από την επιλογή να γίνουν μετατροπές στην srv_paraminfo(), γιατί αυτή η ενέργεια θα προκαλούσε προβλήματα συμβατότητας προς τα πίσω.

Η αδυναμία αυτή αφορά τον Sql Server 2000 και ανακοινώθηκε από τη Microsoft την 1/12/2000, ημερομηνία κατά την οποία δόθηκε στη δημοσιότητα και το patch διόρθωσης.

2.5.2 Επαναχρησιμοποίηση μιας cached σύνδεσης (Sql Server 2000)

Όταν τερματίζεται μια σύνδεση ενός client με τον Sql Server, παραμένει cached για ένα μικρό χρονικό διάστημα για λόγους απόδοσης. Μία από τις sql query μεθόδους (μέθοδοι υποβολής ερωτημάτων σε Β.Δ. του Sql Server) περιέχει μία αδυναμία η οποία έχει ως αποτέλεσμα να είναι δυνατό να επαναχρησιμοποιηθεί μία cached σύνδεση που ανήκε σε ένα sa login.

Εκμεταλλευόμενος αυτή την αδυναμία ένας επιτιθέμενος θα μπορούσε να εκτελέσει ένα query σε administrative περιβάλλον ασφάλειας (security context). Αυτό θα μπορούσε να του δώσει τη δυνατότητα να πραγματοποιήσει οποιαδήποτε επιθυμητή ενέργεια στη βάση δεδομένων. Επιπλέον, θα μπορούσε να του δώσει τη δυνατότητα να τρέξει extended stored procedures και επομένως να τρέξει κώδικα της επιλογής του που θα του δώσει τον έλεγχο του server.

Παρ' όλα αυτά, αυτή η αδυναμία επηρεάζει μόνο servers που έχουν ρυθμιστεί να χρησιμοποιούν τη Mixed Mode Authentication.Οι servers που έχουν ρυθμιστεί να χρησιμοποιούν τη Windows authentication δεν είναι ευάλωτοι σε αυτή την αδυναμία ασφάλειας.

Επιπλέον, οι τερματιζόμενες συνδέσεις γίνονται cached μόνο για ένα μικρό χρονικό διάστημα. Ο επιτιθέμενος θα χρειαζόταν να προγραμματίσει την επίθεσή του έτσι ώστε να συμβεί κατά το διάστημα στο οποίο η σύνδεση του sa login είναι ακόμη στην cache.
Τέλος, η query μέθοδος που παρουσιάζει τη συγκεκριμένη αδυναμία μπορεί να εκτελεσθεί μόνο από αυθεντικοποιημένο χρήστη. Αυτό όχι μόνο περιορίζει τον αριθμό των χρηστών οι οποίοι θα μπορούσαν να εκμεταλλευτούν την αδυναμία αυτή, αλλά ταυτόχρονα επιτρέπει την καταγραφή των ενεργειών του χρήστη που επιχείρησε να εκμεταλλευτεί τη συγκεκριμένη αδυναμία ασφάλειας.
Η αδυναμία αυτή αφορά τον Sql Server 2000 και ανακοινώθηκε από τη Microsoft την 12/6/2001, ημερομηνία κατά την οποία δόθηκε στη δημοσιότητα και το patch διόρθωσης.

2.5.3 Text formatting μέθοδοι του Sql Server που περιέχουν μη ελεγχόμενα buffers (Sql Server 2000).

Ο Sql Server 2000 παρέχει ένα πλήθος από μεθόδους που καθιστούν εφικτό τα ερωτήματα στη βάση δεδομένων να δημιουργούν μηνύματα κειμένου. Σε κάποιες περιπτώσεις οι μέθοδοι αυτές δημιουργούν ένα μήνυμα κειμένου και το αποθηκεύουν σε μία μεταβλητή, ενώ άλλες παρουσιάζουν απευθείας το μήνυμα. Έχουν εντοπιστεί δύο αδυναμίες ασφάλειας σχετικά με αυτές τις μεθόδους.
Η πρώτη αδυναμία προκύπτει από το γεγονός ότι πολλές από αυτές τις μεθόδους δεν εξασφαλίζουν επαρκώς ότι το παραχθέν κείμενο θα χωράει στο buffer στο οποίο παρέχεται για να το αποθηκεύει. Ως αποτέλεσμα, θα μπορούσε να συμβεί υπερχείλιση buffer κάτι το οποίο θα μπορούσε να χρησιμοποιηθεί είτε για την εκτέλεση κώδικα στο περιβάλλον ασφάλειας του Sql Server service είτε ώστε να προκαλέσει αποτυχία στο Sql Server service. Όπως έχει αναφερθεί και προηγουμένως, ο Sql Server μπορεί να ρυθμιστεί έτσι ώστε να τρέχει σε ένα από τα πολλά περιβάλλοντα ασφαλείας, και by default τρέχει ως απλός χρήστης. Τα ακριβή δικαιώματα που θα μπορούσε

να αποκτήσει ο επιτιθέμενος εξαρτώνται άμεσα από το περιβάλλον ασφάλειας στο οποίο τρέχει το Sql Server service.

Η δεύτερη αδυναμία προκύπτει εξαιτίας μιας αδυναμίας που υπάρχει στις C runtime[12] μεθόδους διαμόρφωσης κειμένου που καλούν οι μέθοδοι του Sql Server όταν αυτός είναι εγκατεστημένος σε συστήματα Windows NT 4.0, Windows 2000 ή Windows XP. Παρά το γεγονός ότι οι αδυναμίες στις μεθόδους διαμόρφωσης κειμένου θα μπορούσαν να χρησιμοποιηθούν για την εκτέλεση κώδικα επιλογής του επιτιθέμενου, αυτό δεν ισχύει στην πράξη. Εξαιτίας του συγκεκριμένου τρόπου με τον οποίο λειτουργεί αυτή η αδυναμία, ο C runtime κώδικας θα υπερχειλίζει πάντα με τις ίδιες τιμές, ανεξάρτητα από τις εισόδους του επιτιθέμενου. Ως αποτέλεσμα, αυτή η αδυναμία θα μπορούσε να χρησιμοποιηθεί μόνο για επίθεση τύπου denial of service.

Ένας επιτιθέμενος θα μπορούσε να εκμεταλλευτεί αυτές τις δύο αδυναμίες με αρκετούς τρόπους. Ο πιο απλός τρόπος θα ήταν να εκτελέσει ένα ερώτημα στη βάση δεδομένων που καλεί μία από τις προβληματικές μεθόδους. Εναλλακτικά, αν ένα web site ή κάποιο front-end μιας βάσης δεδομένων δέχεται και εκτελεί τυχαία ερωτήματα, θα μπορούσε να είναι δυνατό για τον επιτιθέμενο να παρέχει εισόδους που θα μπορούσαν να προκαλέσουν ένα ερώτημα στη βάση δεδομένων το οποίο να καλεί κάποια από αυτές τις «προβληματικές» μεθόδους με τις κατάλληλες παραμέτρους.

Παρ' όλα αυτά υπάρχουν κάποιοι αντισταθμιστικοί παράγοντες όσον αφορά τις επιπτώσεις των δύο αυτών αδυναμιών:

---

[12] Το C Runtime είναι ένα σύνολο εκτελέσιμων και αρχείων που παρέχουν υποστήριξη για εφαρμογές που έχουν γραφεί στη γλώσσα προγραμματισμού C. Όλες οι Windows πλατφόρμες περιέχουν runtime για τη C, όπως επίσης και για άλλες γλώσσες. Το πρόβλημα παρουσιάζεται εξαιτίας μίας αδυναμίας διαμόρφωσης string που περιέχεται σε μία από τις functions του C runtime που βρίσκεται στα Windows NT 4.0., στα Windows 2000 και στα Windows XP.

- Το αποτέλεσμα της εκμετάλλευσης της πρώτης αδυναμίας ασφάλειας εξαρτάται από τα δικαιώματα του λογαριασμού που έχει σχετισθεί με το Sql Server service. Εάν έχουν ακολουθηθεί οι προτεινόμενες πρακτικές για το περιβάλλον ασφάλειας του Sql Server service, τότε το εύρος των προβλημάτων που μπορεί να προκαλέσει ένας επιτιθέμενος περιορίζεται σημαντικά.
- Η δεύτερη αδυναμία θα μπορούσε να χρησιμοποιηθεί μόνο για επιθέσεις denial of service. Δε μπορεί να χρησιμοποιηθεί για εκτέλεση κώδικα στο σύστημα.
- Η δεύτερη αδυναμία μπορεί να εκμεταλλευθεί μόνο εναντίον Sql Server που είναι εγκατεστημένος σε Windows 2000, Windows NT 4.0 και Windows XP.

Η συγκεκριμένη αδυναμία ασφάλειας, που αφορά τον Sql Server 2000, ανακοινώθηκε στις 20/12/2001 οπότε και δόθηκαν στη δημοσιότητα τα δύο patches διόρθωσης.

2.5.4 Μη ελεγχόμενα buffers στη function απομακρυσμένων πηγών δεδομένων του Sql Server (Sql Server 2000).

Ένα από τα χαρακτηριστικά της Sql στον Sql Server 2000 είναι η δυνατότητα σύνδεσης με απομακρυσμένες πηγές δεδομένων. Μία δυνατότητα αυτής της λειτουργίας είναι η χρήσης ad hoc συνδέσεων για σύνδεση με μια απομακρυσμένη πηγή δεδομένων χωρίς εγκατάσταση linked server. Αυτό είναι δυνατό μέσω της χρήσης OLE DB[13] παροχέων, οι οποίες επιτρέπουν παροχή δεδομένων σε χαμηλό επίπεδο. Αυτή η δυνατότητα γίνεται εφικτή χρησιμοποιώντας το όνομα του OLE DB παροχέα απευθείας σε ένα ερώτημα στη βάση δεδομένων.

[13] Object Linking and Embedding, Database – API σχεδιασμένο από τη Microsoft μέσω του οποίου είναι δυνατή η πρόσβαση σε διαφορετικούς τύπους δεδομένων μέσω ενός ενοποιημένου τρόπου.

Ο χειρισμός των ονομάτων των OLE DB παροχέων στις ad-hoc συνδέσεις περιέχει ένα μη ελεγχόμενο buffer. Η υπερχείλιση του buffer μπορεί να προκαλέσει αποτυχία στο Sql Server service ή την εκτέλεση κώδικα στο περιβάλλον ασφάλειας του Sql Server. Τα ακριβή δικαιώματα που θα μπορούσε να κερδίσει ένας επιτιθέμενος εξαρτώνται από το συγκεκριμένο περιβάλλον ασφάλειας στο οποίο τρέχει το service του Sql Server.

Ένας επιτιθέμενος θα μπορούσε να εκμεταλλευτεί αυτή την αδυναμία με δυο τρόπους. Θα μπορούσε να προσπαθήσει να φορτώσει και να εκτελέσει ένα ερώτημα στη βάση δεδομένων που καλεί μία από τις προβληματικές μεθόδους. Εναλλακτικά, αν ένα web-site ή ένα front-end μιας βάσης δεδομένων είναι ρυθμισμένο να εκτελεί τυχαία ερωτήματα, θα ήταν δυνατό για τον επιτιθέμενο να παρέχει εισόδους που θα μπορούσαν να αναγκάσουν το ερώτημα να καλέσει την «προβληματική» μέθοδο με τις κατάλληλες παραμέτρους.

Αντισταθμιστικός παράγοντας στους κινδύνους από αυτή την αδυναμία ασφάλειας είναι το γεγονός ότι και οι δύο τρόποι εκμετάλλευσης αυτής της αδυναμίας μπορούν εύκολα να αποσοβηθούν. Συγκεκριμένα, μη έμπιστοι χρήστες δεν πρέπει να έχουν τη δυνατότητα να φορτώνουν και να εκτελούν ερωτήματα της επιλογής τους σε έναν database server και επιπλέον τα δημόσια προσβάσιμα ερωτήματα συνήθως ελέγχουν τις εισόδους τους πριν να ξεκινήσει η επεξεργασία τους.

Η συγκεκριμένη αδυναμία αφορά τον Sql Server 2000, ανακοινώθηκε στις 20/2/2002, οπότε δόθηκε στη δημοσιότητα και το patch διόρθωσης.

2.5.5 Log files με ευαίσθητες πληροφορίες κατά την εγκατάσταση του Sql Server 2000

Κατά την εγκατάσταση του Sql Server 2000 ή ενός από τα service packs για τον Sql Server 2000, οι πληροφορίες που παρέχονται από το χρήστη κατά τη διαδικασία εγκατάστασης συγκεντρώνονται και

αποθηκεύονται σε ένα αρχείο που ονομάζεται setup.iss. Το αρχείο setup.iss μπορεί στη συνέχεια να χρησιμοποιηθεί για να αυτοματοποιήσει τη διαδικασία εγκατάστασης σε άλλα Sql Server συστήματα. Ο Sql Server 2000 επίσης περιλαμβάνει τη δυνατότητα να καταγράψει τις πληροφορίες για μία εγκατάσταση στο αρχείο setup.iss, χωρίς στην πραγματικότητα να πραγματοποιήσει την εγκατάσταση. Ο διαχειριστής Sql Server κατά τη διάρκεια της εγκατάστασης πέρα από τις υπόλοιπες πληροφορίες πρέπει να παρέχει και έναν κωδικό κάτω από τις παρακάτω συνθήκες:

- Εάν ο Sql Server ρυθμιστεί κατά την εγκατάσταση να λειτουργεί με Mixed Mode Authentication, πρέπει να δημιουργηθεί ένας κωδικός για τον Sql Server administrator (sa login).
- Είτε στη Mixed Mode Authentication είτε στη Windows authentication, μπορεί προαιρετικά να δοθεί ένα όνομα χρήστη και ένας κωδικός για το Sql Server service.

Και στις δύο περιπτώσεις, οι κωδικοί που θα εισαχθούν αποθηκεύονται στο αρχείο setup.iss. Επιπρόσθετα, κατά τη διάρκεια της εγκατάστασης δημιουργείται ένα log file που περιέχει τα αποτελέσματα της εγκατάστασης. Το log file αυτό περιέχει και αυτό τους κωδικούς που αποθηκεύονται στο setup.iss αρχείο.

Στη συγκεκριμένη περίπτωση προκύπτει μια αδυναμία ασφάλειας εξαιτίας δύο παραγόντων:

- Τα προαναφερθέντα αρχεία παραμένουν στο server ακόμη και μετά το τέλος της εγκατάστασης και βρίσκονται (εκτός από το αρχείο setup.iss) σε καταλόγους που δεν προστατεύονται από περιορισμένα δικαιώματα.
- Οι κωδικοί αποθηκεύονται στα προαναφερθέντα αρχεία χρησιμοποιώντας κρυπτογραφία χαμηλής προστασίας. Ένας επιτιθέμενος που απέκτησε με κάποιο τρόπο αυτά τα αρχεία

μπορεί να τα υποβάλει σε μία cracking[14] επίθεση με σκοπό να ανακτήσει τους κωδικούς του sa login ή/και του Sql Server service λογαριασμού.

Αντισταθμιστικοί παράγοντες σε αυτή την αδυναμία ασφάλειας είναι οι εξής:

- Την αδυναμία αυτή μπορεί να εκμεταλλευθεί ένας επιτιθέμενος που είχε τη δυνατότητα να συνδεθεί απομακρυσμένα με το σύστημα του Sql server. Παρ' όλα αυτά οι προτεινόμενες πρακτικές από τη Microsoft συνιστούν να μην είναι δυνατή η απομακρυσμένη σύνδεση σε servers που φιλοξενούν τον Sql Server.
- Η αδυναμία σχετικά με τον κωδικό του sa login επηρεάζει μόνο τους servers που έχουν ρυθμιστεί να λειτουργούν με Mixed Mode Authentication. Οι servers που χρησιμοποιούν τη Windows Authentication θα έχουν θέσει σε κίνδυνο τις πληροφορίες πρόσβασης μόνο εάν έχει επιλεχθεί όνομα Windows λογαριασμού και κωδικός για το Sql Server service.
- Οι κωδικοί που αποθηκεύονται στο αρχείο setup.iss και στα log files είναι αυτά που παρέχονται κατά τη διάρκεια της εγκατάστασης και δεν ενημερώνονται όταν γίνονται αλλαγές σε αυτούς τους κωδικούς μετά το πέρας της εγκατάστασης. Ως αποτέλεσμα, εάν ο διαχειριστής αλλάξει τον κωδικό, οι πληροφορίες που παρέχονται στο αρχείο setup.iss δε θα παρέχουν τις σωστές πληροφορίες για την πρόσβαση στον Sql Server.
- Το αρχείο setup.iss αποθηκεύεται σε έναν κατάλογο στο οποίο επιτρέπεται η πρόσβαση μόνο σε administrators και το χρήστη που εγκαθιστά τον Sql Server.

---

[14] Διαδικασία επίθεσης κατά την οποία γίνονται επαναλαμβανόμενες προσπάθειες για την εύρεση του σωστού κωδικού πάνω σε δεδομένα που είτε είναι αποθηκευμένα είτε μεταδίδονται.

- Αν τα αρχεία setup.iss, και τα ini και log files που περιέχουν τους κωδικούς διαγραφούν, τότε τα passwords που δόθηκαν κατά τη διάρκεια της εγκατάστασης δεν μπορούν να ανακτηθούν.

Η αδυναμία αυτή ασφάλειας αφορά τον Sql Server 2000. Το patch διόρθωσης τιτλοφορείται με την ονομασία KillPwd και απομακρύνει τους κινδύνους που προκύπτουν από αυτή. Η τελευταία έκδοση του KillPwd δόθηκε στη δημοσιότητα στις 14/6/2005.

2.5.6 Αδυναμία ασφάλειας στη διαδικασία μαζικής εισαγωγής δεδομένων (Sql Server 2000)

Η διαδικασία μαζικής εισαγωγής δεδομένων σε Sql Server πίνακες περιέχει μια αδυναμία μη ελεγχόμενου buffer. Ένας επιτιθέμενος εκμεταλλευόμενος επιτυχώς αυτή την αδυναμία μπορεί να αποκτήσει σημαντικό έλεγχο της βάσης δεδομένων και πιθανότατα του ίδιου του server.

Αντισταθμιστικοί παράγοντες στους ελλοχεύοντες κινδύνους αυτής της αδυναμίας είναι:

- Η προϋπόθεση ο επιτιθέμενος να έχει ήδη σημαντικά δικαιώματα πρόσβασης στο server, αφού μόνο οι Bulk admins και οι administrators έχουν τη δυνατότητα να φορτώνουν και να εκτελούν ερωτήματα που ενεργοποιούν την «προβληματική» όσον αφορά το συγκεκριμένο πρόβλημα ασφάλειας αυτή διαδικασία.
- Όπως και σε πολλά από τα προαναφερθέντα προβλήματα ασφάλειας το εύρος εκμετάλλευσης της συγκεκριμένης αδυναμίας εξαρτάται από το περιβάλλον ασφάλειας στο οποίο έχει ρυθμιστεί να εκτελείται το service του Sql Server.

Η αδυναμία αυτή ασφάλειας αφορά τον Sql Server 2000 και ανακοινώθηκε από την εταιρεία παραγωγής στις 10/7/2002.

2.5.7 Αδυναμία ασφάλειας σχετικά με την αποθήκευση των πληροφοριών του Sql Server service λογαριασμού (Sql Server 2000)

Οι πληροφορίες για τον Sql Server service λογαριασμό αποθηκεύονται σε ένα Registry key του συστήματος στο οποίο είναι εγκατεστημένος ο Sql Server. Η αδυναμία ασφάλειας έγκειται στο γεγονός ότι τα δικαιώματα πρόσβασης σε αυτό το registry key δεν είναι τα κατάλληλα. Ένας επιτιθέμενος που εκμεταλλεύθηκε επιτυχώς αυτή την αδυναμία θα μπορούσε να αποκτήσει περισσότερα δικαιώματα στο σύστημα από αυτά τα οποία του έχουν δοθεί από το διαχειριστή.

- Στη συγκεκριμένη περίπτωση αντισταθμιστικοί παράγοντες είναι η προϋπόθεση ο επιτιθέμενος να έχει τη δυνατότητα να φορτώνει και να εκτελεί ερωτήματα στη βάση δεδομένων. Ακολουθώντας τις προτεινόμενες τακτικές και περιορίζοντας αυτή τη δυνατότητα στους administrators, οι χρήστες μπορούν να αντισταθμίσουν την απειλή που θέτει αυτή η αδυναμία ασφάλειας.
- Η επιτυχής εκμετάλλευση αυτής της αδυναμίας επίσης απαιτεί έναν sysadmin ή κάποιον που κατέχει τα xp_regwrite execute δικαιώματα.

Η συγκεκριμένη αδυναμία ασφάλειας αφορά τον Sql Server 2000 και αντιμετωπίζεται από τη Microsoft με patch διόρθωσης.

2.5.8 Αδυναμίες ασφάλειας στις Sql Server 2000 Utilities

Η πρώτη αδυναμία ασφάλειας που εντοπίζεται στις Sql Server 2000 Utilities είναι μια αδυναμία υπερχείλισης buffer που συμβαίνει σε δύο από τους Database Consistency Checkers (DBCCs) που αποτελούν μέρος του Sql Server 2000. Τα DBCCs είναι command line utilities που επιτρέπουν να εκτελεστούν διαδικασίες συντήρησης αλλά και άλλες διαδικασίες στον Sql Server. Παρά το γεγονός ότι πολλές από αυτές μπορούν να εκτελεστούν μόνο από sysadmins, κάποιες άλλες

είναι εκτελέσιμες και από μέλη των db_owner και db_ddladmin ρόλων επίσης. Η εμφάνιση της αδυναμίας αυτής οφείλεται στην ύπαρξη μη ελεγχόμενου buffer στο κομμάτι του κώδικα των δύο αυτών stored procedures που ελέγχει τις παραμέτρους εισόδου. Εάν κάποιος επιτιθέμενος χρησιμοποιήσει μια κατάλληλη παράμετρο είναι δυνατό να προκαλέσει υπερχείλιση στο buffer κάποιας από αυτές τις δύο stored procedures με αποτέλεσμα είτε την αποτυχία του Sql Server είτε, στη σοβαρότερη περίπτωση, τη δυνατότητα στον επιτιθέμενο να εκτελέσει κώδικα της επιλογής του στο περιβάλλον ασφάλειας του Sql Server Service, και επομένως δυνητικά τη δυνατότητα να αποκτήσει πλήρη πρόσβαση σε όλες τις βάσεις δεδομένων του server. Παρά το γεγονός ότι η συγκεκριμένη αδυναμία ασφάλειας είναι εκμεταλλεύσιμη μόνο από users που είναι μέλη των db_owner και db_ddladmin, που σημαίνει ότι συνήθως πρόκειται για έμπιστους χρήστες για μία συγκεκριμένη βάση δεδομένων, τους επιτρέπει να αποκτήσουν ενδεχόμενα πρόσβαση σε όλες τις βάσεις δεδομένων ενός Sql Server, πράγμα που ενέχει κινδύνους ασφάλειας, καθώς εμπιστοσύνη για μία συγκεκριμένη βάση δεδομένων δεν συνεπάγεται εμπιστοσύνη και για όλες τις υπόλοιπες βάσεις δεδομένων του Sql Server.

Οι παράγοντες που αντισταθμίζουν τους κινδύνους αυτής της αδυναμίας ασφάλειας είναι οι εξής:

- Τόσο ο db_owner όσο και ο db_ddladmin ρόλος εμπεριέχουν σημαντικά δικαιώματα και συνήθως ανατίθενται σε έμπιστους χρήστες.
- Η εκμετάλλευση και αυτής της αδυναμίας εξαρτάται από τα δικαιώματα του Sql Server λογαριασμού.

Η δεύτερη αδυναμία εντοπίζεται σε μια Sql injection (2.5.11) αδυναμία που συμβαίνει σε δύο stored procedures που χρησιμοποιούνται για τη δημιουργία αντιγράφων (replication) βάσεων δεδομένων. Η μία από αυτές μπορεί να εκτελεστεί μόνο από

users που ανήκουν στον db_owner ρόλο, ενώ η άλλη, εξαιτίας μίας παράλειψης στα δικαιώματα πρόσβασης, μπορεί να εκτελεσθεί και από έναν απομακρυσμένο χρήστη. Η εκμετάλλευση αυτής της αδυναμίας θα μπορούσε να δώσει στον επιτιθέμενο τη δυνατότητα να εκτελεί εντολές SQL ή και εντολές του λειτουργικού συστήματος στο server, άλλα ταυτόχρονα υπόκειται και σε σημαντικούς αντισταθμιστικούς παράγοντες:

- Η εκμετάλλευση αυτής της αδυναμίας απαιτεί ως ελάχιστη προϋπόθεση τη δυνατότητα του επιτιθέμενου να συνδεθεί απομακρυσμένα στο server. Παρ' όλα αυτά οι προτεινόμενες πρακτικές αποθαρρύνουν την ανάθεση τέτοιων δικαιωμάτων σε μη έμπιστους χρήστες.
- Απλά η δυνατότητα να εκτελεί κάποιος επιτιθέμενος τις συγκεκριμένες stored procedures δεν επαρκεί έτσι ώστε να εκμεταλλευθεί τη συγκεκριμένη αδυναμία ασφάλειας. Χρειάζεται επίσης ο διαχειριστής να έχει προηγουμένως ενεργοποιήσει το Sql Server Agent Proxy account[15] το οποίο by default είναι απενεργοποιημένο. Ακόμα και όταν ο Sql Server Agent Proxy λογαριασμός είναι ενεργοποιημένος έχει by default μόνο τα δικαιώματα που σχετίζονται με έναν domain user.

Οι αδυναμίες αυτές ανακοινώθηκαν από τη Microsoft στις 24/7/2002, οπότε δόθηκαν στη δημοσιότητα και τα patches διόρθωσης.

[15] Ο Sql Server Agent Proxy account είναι ένα user account ειδικού σκοπού που χρησιμοποιείται από τον SQL server agent και το xp_cmdshell όταν εκτελούνται εργασίες ή εντολές για χρήστες που δεν είναι μέλη του sysadmin ρόλου. Ο λογαριασμός αυτός είναι απενεργοποιημένος by default και μπορεί να ενεργοποιηθεί μόνο από έναν διαχειριστή. Ακόμα και όταν ενεργοποιηθεί τρέχει με τα δικαιώματα με τα οποία τον έχει ρυθμίσει να τρέχει ο διαχειριστής.

2.5.9 Αδυναμία ασφάλειας κατά τη διαδικασία αυθεντικοποίησης στον Sql Server 2000

Η αδυναμία ασφάλειας αυτή εντοπίζεται σε μια πιθανή υπερχείλιση buffer σε ένα κομμάτι κώδικα του Sql Server που σχετίζεται με την αυθεντικοποίηση χρηστών. Αποστέλλοντας μια προσεκτικά σχεδιασμένη αίτηση αυθεντικοποίησης στον Sql Server, ένας επιτιθέμενος θα μπορούσε είτε να προκαλέσει αποτυχία στο server είτε να αποκτήσει τη δυνατότητα να υπερχειλίσει τη μνήμη στο server δίνοντας του τη δυνατότητα να τρέξει κώδικα στον server στο περιβάλλον ασφάλειας του Sql Server service. Επιπλέον δεν είναι απαραίτητο για το χρήστη να αυθεντικοποιηθεί επιτυχώς στο server ή να εκτελέσει συγκεκριμένες εντολές σε αυτόν έτσι ώστε να εκμεταλλευτεί την αδυναμία αυτή ασφάλειας.

Αντισταθμιστικοί παράγοντες στους κινδύνους που εισάγει αυτή η αδυναμία ασφάλειας είναι οι εξής:

- Εάν το port του Sql Server (1433) είναι μπλοκαρισμένο από το firewall, αυτή η αδυναμία δεν μπορεί να γίνει εκμεταλλεύσιμη χρησιμοποιώντας ως μέσο το Internet.
- Η εκμετάλλευση αυτής της αδυναμίας θα μπορούσε να επιτρέψει στον επιτιθέμενο να κλιμακώσει τα δικαιώματα του στο επίπεδο των δικαιωμάτων του Sql Server service. By default, το service τρέχει με τα περιορισμένα δικαιώματα του domain user και όχι του system user.

Η συγκεκριμένη αδυναμία ασφάλειας ανακοινώθηκε από την εταιρεία παραγωγής στις 2/10/2002 και αφορά τον Sql Server 2000.

2.5.10 Αδυναμία ασφάλειας στις προγραμματιζόμενες εργασίες του Sql Server 2000

Ο Sql Server επιτρέπει σε μη εξουσιοδοτημένους χρήστες να δημιουργήσουν προγραμματισμένες εργασίες που θα εκτελεσθούν από τον Sql Server Agent. By default, ο Sql Server Agent εκτελεί μόνο τις εργασίες που επιτρέπονται από τα δικαιώματα χρήσης του

χρήστη που προγραμμάτισε τη συγκεκριμένη εργασία. Παρ' όλα αυτά όταν ένα από τα βήματα μιας εργασίας απαιτεί να δημιουργηθεί ένα output αρχείο, ο Sql Server Agent δημιουργεί αυτό το αρχείο χρησιμοποιώντας τα δικά του δικαιώματα αντί για τα δικαιώματα του χρήστη που δημιούργησε την εργασία. Αυτό δημιουργεί μια κατάσταση στην οποία ένας μη εξουσιοδοτημένος χρήστης μπορεί να προγραμματίσει μία εργασία η οποία να δημιουργεί ένα αρχείο που περιέχει εντολές λειτουργικού συστήματος που να πραγματοποιούν κάποιες επιθυμητές ενέργειες σε καταλόγους του συστήματος ή απλά να επικαλύψουν αρχεία συστήματος με σκοπό να προκληθεί πρόβλημα στην ομαλή λειτουργία του συστήματος.
Η συγκεκριμένη αδυναμία ασφάλειας ανακοινώθηκε από την εταιρεία παραγωγής στις 2/10/2002 και αφορά τον Sql Server 2000.

2.5.11 Sql Injection

Ο όρος Sql Injection (Sql «εγχείρηση») περιγράφει την εμφώλευση SQL κώδικα στο σημείο μιας εφαρμογής στο οποίο ο προγραμματιστής δεν προόριζε για αυτό το σκοπό. Στην πραγματικότητα τα περισσότερα προβλήματα λόγω Sql Injection δεν είναι λάθη που οφείλονται στο ίδιο το σύστημα διαχείρισης Β.Δ. αλλά σε κακή υλοποίηση σε προγραμματιστικό επίπεδο του ελέγχου της εγκυρότητας της εισόδου των χρηστών.
Τα προβλήματα Sql Injection παρουσιάζονται σε εφαρμογές που χρησιμοποιούν τεχνικές «χτισίματος» string για την εκτέλεση SQL κώδικα. Για παράδειγμα, σε μία web σελίδα αναζήτησης, ο προγραμματιστής μπορεί να χρησιμοποιήσει τον ακόλουθο κώδικα για να εκτελέσει ένα ερώτημα (παράδειγμα σε γλώσσα VBScript/ASP):

```
Set myRecordset = myConnection.execute("SELECT * FROM myTable
WHERE someText ='" & request.form("inputdata") & "'")
```

Ο λόγος για τον οποίο η παραπάνω γραμμή κώδικα είναι πιθανό να προκαλέσει ένα πρόβλημα SQL injection είναι ο πιθανά κακός

έλεγχος εγκυρότητας. Θεωρείται λανθασμένα αυτονόητο ότι ο χρήστης δεν έχει εισάγει κάτι κακόβουλο στη φόρμα αναζήτησης. Για παράδειγμα, είναι εντυπωσιακό τι θα συνέβαινε εάν ο χρήστης εισήγε το παρακάτω κείμενο στη φόρμα αναζήτησης:

```
master..xp_cmdshell 'net user test testpass /ADD' –
```

Τότε το παραχθέν string που θα αποθηκευόταν στη μεταβλητή myRecordset θα ήταν ένα έγκυρο SQL ερώτημα που θα στελνόταν στον SQL server, ο οποίος θα εξυπηρετούσε τον ακόλουθο ερώτημα:

SELECT * FROM myTable WHERE someText ='' exec master..xp_cmdshell 'net user test testpass /ADD'--'

Το πρώτο μονό εισαγωγικό που εισήχθη από το χρήστη κλείνει το string (κενό string) και ο SQL Server προχωρά στην επόμενη sql εντολή που προσθέτει έναν νέο χρήστη στη βάση δεδομένων των τοπικών accounts. Εάν η web εφαρμογή αυτή έτρεχε ως 'sa' και το Sql Server service τρέχει με επαρκή δικαιώματα αυτές οι δύο ουσιαστικά εντολές θα εκτελούνταν κανονικά. Επίσης, να σημειωθεί ο τελεστής σχολίου της SQL (--) που χρησιμοποιείται ώστε να αναγκάσει τον SQL Server να αγνοήσει το τελευταίο μονό εισαγωγικό που παράγει ο κώδικας της εφαρμογής.
Τα μονά εισαγωγικά δεν είναι το μόνο πρόβλημα. Μπορεί να παρουσιαστεί σελίδα αναζήτησης στην οποία η είσοδος του χρήστη να είναι ένα νούμερο και όχι ένα string. Εάν ο χρήστης αντί για νούμερο τοποθετήσει SQL κώδικα στην είσοδο και ο προγραμματιστής δεν ελέγξει τον τύπο των δεδομένων της εισόδου, τότε ο SQL Server πιθανότατα θα εκτελέσει το παραγόμενο ερώτημα. Οι κίνδυνοι που προκύπτουν από μια επίθεση SQL Injection ποικίλουν. Το μέγεθος των κινδύνων αυτών βρίσκεται σε άμεση συνάρτηση με τα δικαιώματα χρήσης του Sql Server από την εφαρμογή.

Μερικά από τα πιθανά προβλήματα που μπορούν να προκύψουν από επιτυχείς SQL Injection επιθέσεις ανάλογα με τα δικαιώματα χρήσης του SQL Server από την εφαρμογή είναι τα εξής:

- Όταν η εφαρμογή έχει πρόσβαση στον SQL Server χρησιμοποιώντας 'sa' δικαιώματα: Πλήρη πρόσβαση στον SQL Server με δυνατότητα εκτέλεσης εντολών του λειτουργικού συστήματος στο περιβάλλον ασφάλειας του SQL Server service λογαριασμού χρησιμοποιώντας την xp_cmdshell extended stored procedure. Δυνατότητα ανάγνωσης, εγγραφής και διαχείρισης όλων των δεδομένων των Β.Δ. του SQL Server.
- Όταν η εφαρμογή έχει πρόσβαση στον SQL Server χρησιμοποιώντας 'db_owner' δικαιώματα: Δυνατότητα ανάγνωσης / εγγραφής σε όλα τα δεδομένα της επηρεαζόμενης βάσης δεδομένων. Δυνατότητα διαγραφής πινάκων, δημιουργίας νέων αντικειμένων και γενικά πλήρους ελέγχου της επηρεαζόμενης βάσης δεδομένων.
- Όταν η εφαρμογή έχει πρόσβαση στον SQL Server χρησιμοποιώντας κανονικά user δικαιώματα (προτείνεται): Δυνατότητα πρόσβασης σε όλα τα αντικείμενα της βάσης δεδομένων στα οποία έχει δικαίωμα πρόσβαση ο συγκεκριμένος λογαριασμός. Στην καλύτερη περίπτωση αυτό σημαίνει μόνο δυνατότητα εκτέλεσης κάποιων stored procedures. Στη χειρότερη περίπτωση, σημαίνει δυνατότητα ανάγνωσης / εγγραφής σε όλους τους πίνακες και views της βάσης δεδομένων. Εδώ, γίνεται κατανοητό πόσο σημαντικές μπορεί να είναι οι stored procedures στην ασφάλεια. Εάν επιτραπεί στους χρήστες να διαχειρίζονται δεδομένα της βάσης μόνο με χρήση κάποιων συγκεκριμένων stored procedures και αποκλειστεί η απευθείας πρόσβαση σε πίνακες, αυτόματα περιορίζεται σημαντικότατα το εύρος των δυνατών ενεργειών των επιτιθέμενων.

Υπάρχουν αρκετοί τρόποι για την προγραμματιστική αντιμετώπιση των SQL injection επιθέσεων (π.χ. isnumeric έλεγχοι, αντικατάσταση μονών εισαγωγικών κλπ). Παρ' όλα αυτά ο αποδοτικότερος τρόπος όσον αφορά τον Sql Server (αλλά και άλλα Σ.Δ.Β.Δ) είναι η χρήση παραμετροποιημένων ερωτημάτων. Η μη χρήση παραμετροποιημένων ερωτημάτων για πρόσβαση σε δεδομένα αποτελεί το ιδανικότερο έδαφος για αδυναμίες ασφάλειας. Ακόμη όμως και όταν χρησιμοποιούνται παραμετροποιημένα ερωτήματα πρέπει να ελέγχεται ότι τυχούσες κλήσεις EXEC ή sp_execute[16] δεν αναπαράγουν το πρόβλημα SQL injection. Παρακάτω παρουσιάζεται ένα παράδειγμα με το σωστό τρόπο για την πρόσβαση σε δεδομένα με χρήση παραμετροποιημένου ερωτήματος:

```
SqlDataReader rdr = null;
SqlConnection con = null;
SqlCommand cmd = null;

string ConnectionString = "server=yourserver;Integrated Security=SSPI;
database=northwind";
con = new SqlConnection(ConnectionString);
con.Open();

// Set up a command with the given query and associate
// this with the current connection.
string CommandText = "SELECT FirstName, LastName" +
" FROM Employees" +
" WHERE (LastName LIKE @Find)";
cmd = new SqlCommand(CommandText);
cmd.Connection = con;

// Add LastName to the above defined paramter @Find
cmd.Parameters.Add(
new SqlParameter(
"@Find", // The name of the parameter to map
System.Data.SqlDbType.NVarChar, // SqlDbType values
20, // The width of the parameter
"LastName")); // The name of the source column

// Fill the parameter with the value retrieved
// from the text field
```

[16] Stored procedure που επιτρέπει την εκτέλεση του SQL κώδικα που δέχεται στις παραμέτρους της

```
cmd.Parameters["@Find"].Value = txtFind.Text;

// Execute the query
rdr = cmd.ExecuteReader();
```

3. Internet Information Server (IIS)

3.1 *Εισαγωγή*

Ο Internet Information Server, γνωστότερος με τη σύντμηση IIS, είναι ένα σύνολο βασιζόμενων στο Internet services για Windows server. Αποτελεί το δεύτερο δημοφιλέστερο web server, πίσω από τον Apache HTTP Server. Τον Οκτώβριο του 2007, σύμφωνα με έρευνα της Netcraft, το 37,13% όλων web sites και το 38,23% των ενεργών web sites λειτουργούν με IIS. Τα υποστηριζόμενα πρωτόκολλα από τον IIS είναι τα FTP, SMTP, NNTP, και HTTP/HTTPS.

Στα επόμενα κεφάλαια θα επιχειρηθεί μια αξιολόγηση του επιπέδου ασφάλειας των τριών τελευταίων εκδόσεων του IIS, δηλαδή της 5.1, 6.0 και 7.0.

3.2 *Ιστορικά στοιχεία*

Ο IIS αρχικά βγήκε στην αγορά ως ένα πρόσθετο σύνολο βασιζόμενων στο Internet υπηρεσιών για τα Windows NT 3.51. Η έκδοση 2.0 του IIS πρόσθεσε υποστήριξη για τα Windows NT 4.0, ενώ ο IIS 3.0 εισήγε το περιβάλλον των Active Server Pages[17].

Ο IIS παρέχεται σαν προαιρετικό στοιχείο εγκατάστασης (window component) των Microsoft Windows. Οι τρέχουσες εκδόσεις του είναι η 7.0 για τα Windows Vista, η 6.0 για τα Windows Server 2003 και η 5.1 για τα Windows XP Professional. Από την έκδοση 6.0 του IIS και μετά παρέχεται υποστήριξη για το IPv6 πρωτόκολλο.

[17] Η πρώτη χρονικά server-side scripting τεχνολογία της Microsoft, για δυναμικές web σελίδες

3.3 *Μηχανισμός ασφάλειας*

3.3.1 Μηχανισμός αυθεντικοποίησης

Ο IIS προσφέρει πέντε διαφορετικούς τρόπους αυθεντικοποίησης οι οποίοι μπορούν να χρησιμοποιηθούν είτε ο καθένας μόνος του είτε συνδυαζόμενοι. Η επιλογή του συνδυασμού των τρόπων αυθεντικοποίησης μπορεί να γίνει στο επίπεδο του web site, αλλά και ακόμη σε επίπεδο καταλόγων και αρχείων του web site. Οι λεπτομέρειες κάθε τρόπου αυθεντικοποίησης παρουσιάζονται παρακάτω:

Anonymous Authentication: Η anonymous authentication επιτρέπει στους χρήστες πρόσβαση σε επιλεγμένες περιοχές ενός web site χωρίς να απαιτεί διαπιστευτήρια εισόδου. Χρησιμοποιεί έναν συγκεκριμένο Windows λογαριασμό, τα δικαιώματα του οποίου κληρονομεί όποιος αυθεντικοποιείται μέσω της anonymous authentication. Στις προκαθορισμένες ρυθμίσεις, αυτός ο λογαριασμός είναι ο IUSR_<ΌνομαΣυστήματος>, όπου το <ΌνομαΣυστήματος> είναι το όνομα του συστήματος στο οποίο τρέχει ο IIS.

Εφόσον είναι ενεργοποιημένη η anonymous authentication, όταν γίνει request για ένα αρχείο στον IIS, ο IIS ελέγχει τα αντίστοιχα δικαιώματα για να επιβεβαιώσει ότι ο λογαριασμός που έχει οριστεί να χειρίζεται τις ανώνυμες συνδέσεις επιτρέπεται να έχει πρόσβαση σε αυτό το αρχείο. Εάν η πρόσβαση επιτρέπεται, τότε το αρχείο παρέχεται στο χρήστη. Εάν η πρόσβαση δεν επιτρέπεται, ο IIS επιστρέφει ένα 401.3 (άρνηση πρόσβασης – access denied) μήνυμα στον client.

Ο λογαριασμός που θα χρησιμοποιείται από την anonymous authentication μπορεί να τροποποιηθεί στον IIS Manager, είτε στο επίπεδο ολόκληρου του web server είτε στο επίπεδο συγκεκριμένων καταλόγων και αρχείων.

Όταν είναι ενεργοποιημένη η anonymous authentication, ο IIS προσπαθεί να αυθεντικοποιήσει το χρήστη πρώτα με την anonymous authentication ακόμα και αν είναι ενεργοποιημένες πρόσθετες μέθοδοι αυθεντικοποίησης. Εάν ο ανώνυμος λογαριασμός δεν έχει δικαίωμα πρόσβασης σε ένα συγκεκριμένο αρχείο ή πόρο, ο Web server δε δημιουργεί ανώνυμη σύνδεση για αυτόν τον πόρο.

Integrated Windows Authentication: Χρησιμοποιεί το πρωτόκολλο Kerberos 5, όταν αυτό είναι εφικτό, διαφορετικά χρησιμοποιεί το πρωτόκολλο NT Challenge/Response (επίσης γνωστό ως NTLM) για τη διαδικασία αυθεντικοποίησης. Συγκεκριμένα το NTLM χρησιμοποιείται στις εξής μόνο περιπτώσεις:

- Ο client επιχειρεί να αυθεντικοποιηθεί μέσω IP διεύθυνσης.
- Ο client επιχειρεί να αυθεντικοποιηθεί σε ένα server που ανήκει σε διαφορετικό Active Directory[18] forest, ή δεν ανήκει σε κάποιο domain.
- Δεν υπάρχει καθόλου Active Directory domain (συχνά η περίπτωση αυτή αναφέρεται ως «workgroup» ή «peer-to-peer»).
- Όταν κάποιο firewall εμποδίζει την πρόσβαση στις θύρες που απαιτούνται από το Kerberos.

Ας δούμε αναλυτικότερα τα δύο χρησιμοποιούμενα πρωτόκολλα.

Kerberos: Το πρωτόκολλο Kerberos αναπτύχθηκε το 1980 από το Massachusetts Institute of Technology. Χρησιμοποιεί μία τρίτη έμπιστη οντότητα που ονομάζεται κέντρο διανομής κλειδιών (Key Distribution Center – KDC) που αποτελείται από δύο λογικές μονάδες: Έναν server αυθεντικοποίησης και έναν server έκδοσης tickets (Ticketing Server). Το Kerberos πρωτόκολλο λειτουργεί με

[18] Τεχνολογία της Microsoft που παρέχει ένα σύνολο υπηρεσιών δικτύου που περιλαμβάνουν LDAP-like υπηρεσίες καταλόγων, Kerberos αυθεντικοποίηση, DNS ονοματολογία κ.α.

την έννοια των tickets που εξυπηρετούν στο να αποδεικνύουν την ταυτότητα του κάθε χρήστη.

Ο KDC διατηρεί μία βάση δεδομένων από ιδιωτικά κλειδιά. Κάθε οντότητα στο δίκτυο διατηρεί ένα ιδιωτικό κλειδί γνωστό μόνο στον εαυτό της και στο KDC. Η γνώση αυτού του κλειδιού αποδεικνύει την ταυτότητα μιας οντότητας. Για την επικοινωνία μεταξύ δύο οντοτήτων ο KDC δημιουργεί ένα κλειδί συνεδρίας το οποίο χρησιμοποιούν για την επικοινωνία τους.

Η όλη διαδικασία της αυθεντικοποίησης με χρήση του Kerberos πρωτοκόλλου έχει ως εξής: Ο client αυθεντικοποιεί τον εαυτό του στον server αυθεντικοποίησης και λαμβάνει ένα ticket. Στη συνέχεια επικοινωνεί με τον Ticketing Server και χρησιμοποιώντας το ticket που έλαβε αποδεικνύει την ταυτότητά του και ζητά να του παρασχεθεί ένας πόρος / υπηρεσία. Εάν ο client έχει δικαίωμα για αυτή την υπηρεσία, τότε ο Ticketing Server στέλνει ένα ticket στον client. Στη συνέχεια ο client επικοινωνεί με τον server που παρέχει τον πόρο / υπηρεσία που επιθυμεί και χρησιμοποιώντας αυτό το ticket, αποδεικνύει ότι έχει εγκριθεί ώστε να λάβει τον πόρο / υπηρεσία.

Network Challenge/Response: Το πρωτόκολλο αυτό χρησιμοποιεί μια αλληλουχία challenge-response (πρόκληση-απάντηση) που απαιτεί τη μετάδοση τριών μηνυμάτων μεταξύ του client (που επιθυμεί να αυθεντικοποιηθεί) και του server (που αυθεντικοποιεί τον client):

- Ο client αρχικά στέλνει στο server ένα μήνυμα Τύπου 1 που περιέχει ένα σύνολο από flags που αντιστοιχούν σε χαρακτηριστικά που υποστηρίζονται ή απαιτούνται.
- Ο server απαντά με ένα μήνυμα Τύπου 2 που περιέχει ένα παρόμοιο σύνολο από flags που υποστηρίζονται ή απαιτούνται από το server. Με τον τρόπο αυτό επιτυγχάνεται μία συμφωνία στις παραμέτρους αυθεντικοποίησης μεταξύ του server και του client. Τέλος, το μήνυμα Τύπου 2 περιέχει επίσης ένα τυχαίο challenge μήκους 8 bytes.

- Τελικά, ο client χρησιμοποιεί το challenge που έλαβε από το μήνυμα Τύπου 2 και τα διαπιστευτήρια του χρήστη για να υπολογίσει το response. Οι μέθοδοι υπολογισμού διαφέρουν ανάλογα με τις παραμέτρους αυθεντικοποίησης που αναφέρθηκαν προηγουμένως, αλλά στη γενική περίπτωση χρησιμοποιούν MD4/MD5 hashing αλγορίθμους και DES κρυπτογράφηση για να υπολογίσουν το response. Ο client στέλνει το response στο server σε ένα μήνυμα Τύπου 3.

Όταν χρησιμοποιείται η Windows Authentication, αρχικά δε ζητείται κωδικός από το χρήστη. Ο client περνά διαφανώς το τρέχον όνομα χρήστη και κωδικό του χρήστη. Εάν η διαφανής αυτή αυθεντικοποίηση αποτύχει, ο χρήστης θα λάβει μία ένδειξη όπου πρέπει να συμπληρώσει έναν έγκυρο Windows λογαριασμό και κωδικό.

Όλες οι εκδόσεις του Internet Explorer μετά την έκδοση 2 μπορούν να υποστηρίξουν αυτό τον τρόπο αυθεντικοποίησης. Εξαιτίας του γεγονότος ότι αυτός ο τύπος αυθεντικοποίησης δε λειτουργεί με HTTP proxies ή με firewalls που χρησιμοποιούν HTTP proxies, είναι καταλληλότερος για χρήση σε intranet εφαρμογές. Η Integrated Windows Authentication, όπως και η Anonymous Authentication, είναι προεπιλεγμένες στις default ρυθμίσεις.

Digest Authentication: Η Digest authentication μεταδίδει ασφαλώς μία MD5 hash τιμή αντί του clear-text κωδικού, κρατώντας με τον τρόπο αυτό τους κωδικούς εμπιστευτικούς. Η MD5 είναι μια ευρέως χρησιμοποιούμενη μέθοδος hash με hash τιμή μήκους 128 bits. Υπάρχουν ωστόσο κάποια προαπαιτούμενα για τη χρήση αυτού του τρόπου αυθεντικοποίησης. Συγκεκριμένα:

- Οι browsers των clients πρέπει να είναι όλοι Internet Explorer έκδοσης 5 ή μεγαλύτερης.

- Οι χρήστες πρέπει να έχουν ένα έγκυρο λογαριασμό στο Active Directory ή έναν τοπικό Windows λογαριασμό στο σύστημα του IIS.
- Οι κωδικοί των λογαριασμών στο Active Directory (εφόσον αυτό χρησιμοποιείται) πρέπει να είναι αποθηκευμένοι ως καθαρό κείμενο (clear text).

Basic Authentication: Αποτελεί τον ευρύτερα χρησιμοποιούμενο τρόπο αυθεντικοποίησης για τον IIS και πρακτικά, όλοι οι Web browsers τον υποστηρίζουν. Με τη χρήση της Basic Authentication, τα διαπιστευτήρια των χρηστών μεταδίδονται ως καθαρό κείμενο και ελέγχονται απέναντι στους λογαριασμούς του domain του IIS ή αν δεν υπάρχει domain απέναντι στους λογαριασμούς των τοπικών χρηστών του συστήματος του IIS. Για ένα site που είναι δημόσια προσβάσιμο από χρήστες διαφόρων λειτουργικών συστημάτων και browsers, αυτή είναι πιθανότατα η καλύτερη επιλογή όσον αφορά την υλοποίηση της αυθεντικοποίησης. Παρ' όλα οι κωδικοί μεταδίδονται χωρίς χρήση κάποιας κρυπτογράφησης.

.NET Passport Authentication: Η Windows .NET Passport Authentication είναι ένα σύστημα αυθεντικοποίησης που σχεδιάστηκε για να επωφελήσει τα Web sites ηλεκτρονικού εμπορίου έτσι ώστε να διευκολύνει τη διαδικασία πρόσβασης σε αυτά και της πραγματοποίησης online αγορών. Επιλέγοντας αυτό τον τρόπο αυθεντικοποίησης ο server αυθεντικοποιεί τους λογαριασμούς χρησιμοποιώντας έναν .NET Passport Server αντί να χρησιμοποιεί τη βάση δεδομένων τοπικών χρηστών ή το Active Directory. Από τη στιγμή που το .NET passport χρησιμοποιεί καθιερωμένες τεχνολογίες όπως cookies, SSL, JavaScript, είναι συμβατό με τον Internet Explorer έκδοσης 4 και πάνω, με το Netscape Navigator έκδοσης 4 και πάνω ακόμη και με μερικούς Unix browsers.

Εικόνα 9 Επιλογή τρόπου αυθεντικοποίησης στον IIS 6.0

Όταν ο browser ενός client πραγματοποιεί ένα request πάντα, την πρώτη φορά, θεωρεί ότι επιτρέπεται η Anonymous Authentication. Επομένως, δεν στέλνει καθόλου διαπιστευτήρια. Εάν ο server δεν υποστηρίζει την anonymous authentication για το ζητούμενο πόρο ή εάν ο anonymous user λογαριασμός δεν έχει δικαιώματα στον πόρο που ζητείται, ο IIS απαντά με το μήνυμα λάθους «Access Denied» και στέλνει μία λίστα με τους τύπους αυθεντικοποίησης που υποστηρίζονται χρησιμοποιώντας ένα από τα ακόλουθα σενάρια:

- Εάν η Windows Authentication είναι η μόνη υποστηριζόμενη μέθοδος (ή εάν η Anonymous Authentication απέτυχε), τότε ο

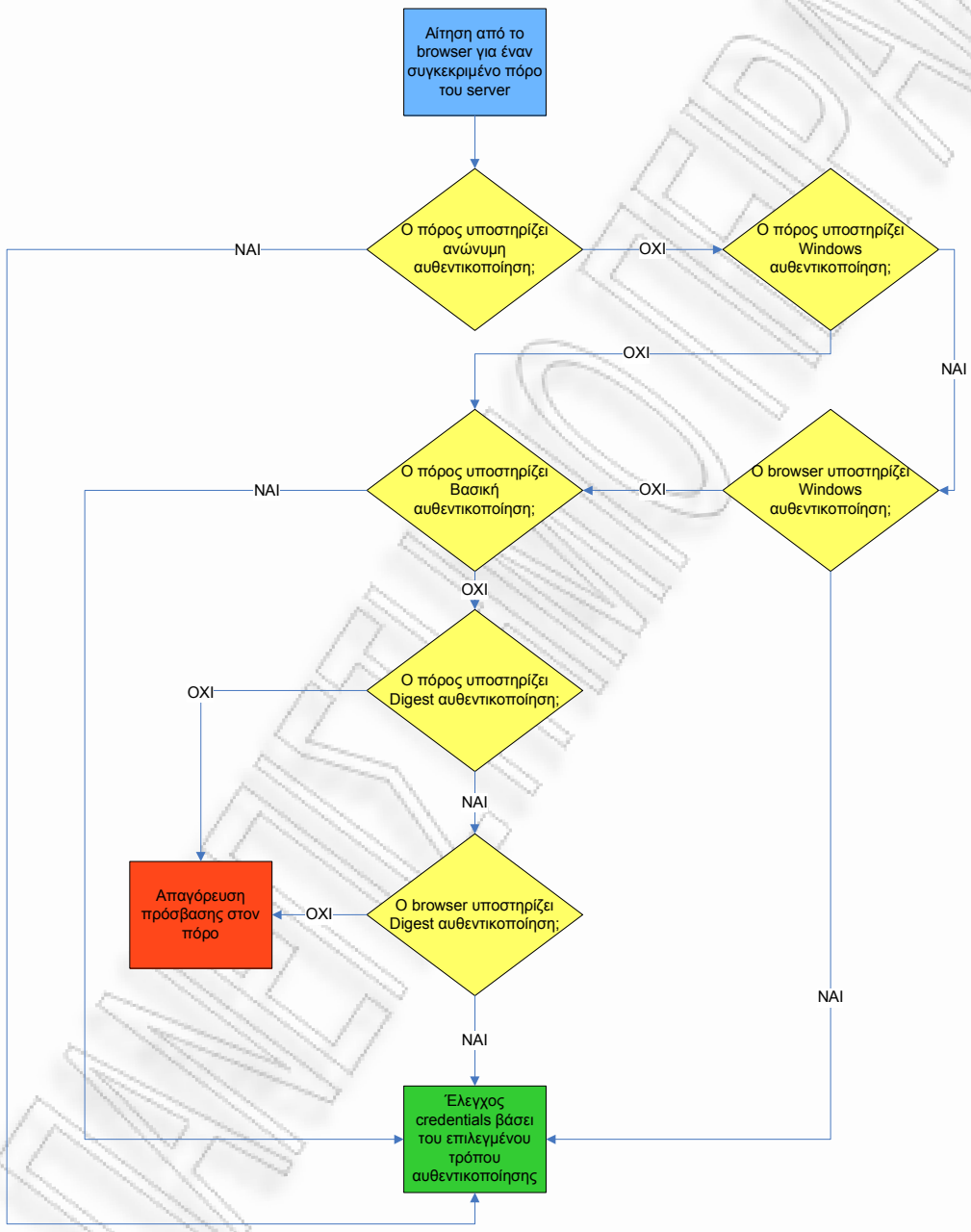
browser πρέπει να υποστηρίζει αυτή τη μέθοδο αυθεντικοποίησης ώστε να επικοινωνήσει με το server. Εάν η Windows authentication αποτύχει ή ο browser δεν την υποστηρίζει, τότε η πρόσβαση στο ζητούμενο πόρο δεν επιτρέπεται.

- Εάν η Basic authentication είναι η μόνη υποστηριζόμενη μέθοδος (ή εάν η Anonymous Authentication απέτυχε), τότε στον browser του client εμφανίζεται ένα παράθυρο διαλόγου στο οποίο ο χρήστης πρέπει να συμπληρώσει τα διαπιστευτήρια του. Αφού ο χρήστης τα εισάγει, τότε αυτά στέλνονται στο server. Γίνονται μέχρι 3 προσπάθειες ελέγχου εγκυρότητας των διαπιστευτηρίων. Εάν όλες αυτές αποτύχουν, δεν επιτρέπεται η πρόσβαση στο ζητούμενο πόρο.
- Εάν υποστηρίζονται τόσο η Basic authentication όσο και η Windows authentication, ο browser αποφασίζει ποια μέθοδος θα χρησιμοποιηθεί. Εάν ο browser υποστηρίζει τη Windows authentication χρησιμοποιεί αυτή τη μέθοδο. Εάν η Windows authentication δεν υποστηρίζεται από τον browser, τότε χρησιμοποιούνται με τη σειρά η Basic και η Digest authentication (εφόσον αυτή υποστηρίζεται και από client και από server).

Παρ΄ όλα αυτά υπάρχουν κάποιες εξαιρέσεις όσον αφορά αυτή τη συμπεριφορά:

Όταν ο browser επιτυγχάνει σύνδεση με ένα Web site χρησιμοποιώντας τη Basic ή της Windows Authentication, τότε δεν επιστρέφει στην Anonymous Authentication κατά τη διάρκεια αυτής της συνεδρίας με το server. Εάν για παράδειγμα μετά την επιτυχή αυθεντικοποίηση είτε με τη Basic είτε με τη Windows authentication επιχειρηθεί στην ίδια συνεδρία σύνδεση σε μία σελίδα στην οποία επιτρέπεται μόνο η Anonymous Authentication, τότε ο server απαντά με μήνυμα άρνησης πρόσβασης.

Όταν ο Internet Explorer έχει εγκαθιδρύσει μία σύνδεση με το server με κάποια μέθοδο αυθεντικοποίησης διαφορετική από την anonymous authentication, τότε αυτόματα περνά τα διαπιστευτήρια για κάθε νέο request κατά τη διάρκεια αυτού της συνεδρίας.



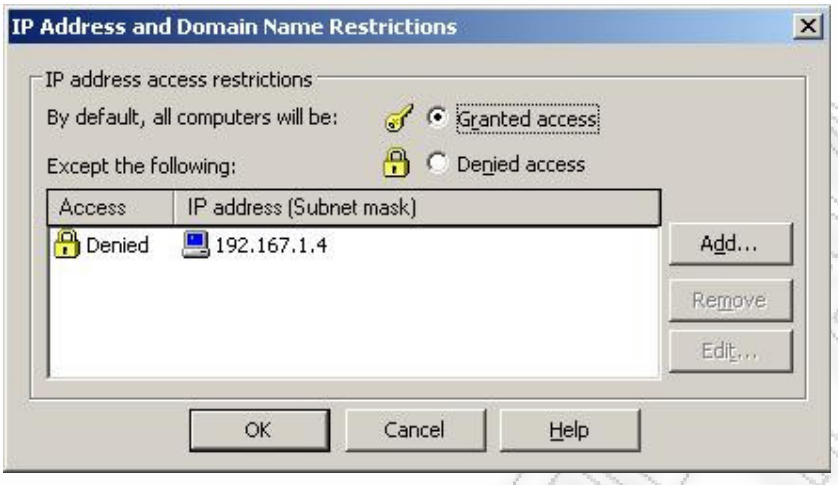
Εικόνα 10 Διαδικασία αυθεντικοποίησης στον IIS 6.0

3.3.2 Δικαιώματα χρήσης πόρων των web sites

Στην περίπτωση που κάποιες περιοχές ενός web site πρέπει να είναι ελεύθερης πρόσβασής ενώ κάποιες άλλες πρέπει να είναι προστατευμένες, τότε πρέπει να επιτρέπεται η ανώνυμη πρόσβαση αλλά ταυτόχρονα να είναι καθορισμένα κατάλληλα δικαιώματα σε συγκεκριμένα αρχεία και καταλόγους που πρέπει να προστατευθούν. Η δυνατότητα αυτή είναι διαθέσιμη μόνο σε NTFS partitions, καθώς δεν μπορούν να καθοριστούν δικαιώματα σε FAT partitions.

Όταν χρησιμοποιείται η Anonymous Authentication, ο IIS θα χρησιμοποιήσει το λογαριασμό του ανώνυμου χρήστη ώστε αρχικά να προσπαθήσει να διαβάσει το αρχείο που ζητήθηκε και να το δώσει πίσω στον client. Για το ελεύθερα προσβάσιμο Web περιεχόμενο που δεν έχει κανένα περιορισμό πρόσβασης, η IIS διεργασία έχει τα δικαιώματα να αποκτήσει πρόσβαση σε αυτό μέσω του ανώνυμου λογαριασμού (IUSR_<ΟνομαΣυστήματος> στις default ρυθμίσεις). Παρ' όλα αυτά αν ο IIS αποτύχει να αποκτήσει πρόσβαση σε ένα αρχείο λόγω κάποιου περιορισμού πρόσβασης, θα αναζητήσει αυθεντικοποίηση από το χρήστη μέσω κάποιου από τους τρόπους αυθεντικοποίησης που έχουν επιλεγεί για το συγκεκριμένο περιεχόμενο και θα επιχειρήσει την πρόσβαση χρησιμοποιώντας αυτά τα στοιχεία που θα συγκεντρώσει από το χρήστη (ή τον browser του χρήστη). Επομένως, εάν πρέπει να αποκλειστεί η πρόσβαση σε συγκεκριμένα αρχεία και καταλόγους για συγκεκριμένους χρήστες, απλά πρέπει να προστεθούν τα δικαιώματα για τα συγκεκριμένα αρχεία ή καταλόγους στους επιθυμητούς χρήστες και να αποκλειστεί η πρόσβαση για τον ανώνυμο λογαριασμό χρήστη.

Ένας άλλος τρόπος για θωράκιση ενός web site είναι ο περιορισμός του ποιος μπορεί να έχει πρόσβαση στο site βάσει IP διεύθυνσης, εύρους IP διευθύνσεων ή ονόματος του domain στο οποίο ανήκει. Η λειτουργία αυτή προαπαιτεί να γνωρίζει κάποιος ποιος θα συνδέεται με το web site του, αλλά είναι εξαιρετικά χρήσιμη για web sites σχεδιασμένα για πελάτες ή προμηθευτές.



Εικόνα 11 Καθορισμός της γενικής συμπεριφοράς απέναντι με όλες τις αιτήσεις πρόσβασης (επίτρεψη ή απαγόρευση πρόσβασης – granted access ή denied access) και επίσης καθορισμός της λίστας εξαιρέσεων

Υπάρχει επιλογή με την οποία επιτρέπεται ή απαγορεύεται η πρόσβαση σε όλους και επίσης δυνατότητα χειρισμού μιας λίστας εξαιρέσεων από αυτή τη γενική επιλογή. Οι εγγραφές της λίστας εξαιρέσεων μπορούν να οριστούν ως ξεχωριστές IP διευθύνσεις, διευθύνσεις δικτύου ή ονόματα domain. Οι περιορισμοί σε επίπεδο ξεχωριστών IP διευθύνσεων μπορεί να είναι χρήσιμη για οικιακούς χρήστες, εφόσον έχουν μία στατική IP διεύθυνση που τους έχει ανατεθεί από τον πάροχο της σύνδεσής τους. Παρ' όλα αυτά είναι περισσότερο συνηθισμένο για τους οικιακούς και τους φορητούς υπολογιστές να χρησιμοποιούνται δυναμικά ανατιθέμενες IP διευθύνσεις. Μερικές φορές μια εταιρεία χρησιμοποιεί firewall ή κάποιο proxy server που κρύβει όλες τις εσωτερικές IP διευθύνσεις και χρησιμοποιεί μόνο μία διεύθυνση ως πηγή όλων των εσωτερικά δημιουργημένων requests. Στην περίπτωση που υπήρχε η ανάγκη να επιτρέπονται οι συνδέσεις μόνο από μία συγκεκριμένη εταιρεία και να αποτρέπονται από οπουδήποτε αλλού, η προσέγγιση θα ήταν να προστιθόταν η IP διεύθυνση του firewall ή του proxy server της εταιρείας στη λίστα εξαιρέσεων. Ομάδες υπολογιστών με στατικές διευθύνσεις ή δυναμικές διευθύνσεις που πάντα βρίσκονται μέσα σε

ένα συγκεκριμένο εύρος μπορούν επίσης να οριστούν χρησιμοποιώντας μια διεύθυνση δικτύου και ένα subnet mask. Οι περιορισμοί με χρήση του domain name είναι χρήσιμες μόνο ως η τελευταία λύση, όταν δεν είναι γνωστές οι IP διευθύνσεις των client συστημάτων που θα επικοινωνούν με τον Web server.

Εικόνα 12 Οθόνη δημιουργίας νέας καταχώρησης στη λίστα εξαιρέσεων βάσει IP διεύθυνσης (Single computer), εύρους IP διευθύνσεων (Group of computers) ή ονόματος domain (domain name)

Παρά το γεγονός ότι υπάρχει τρόπος ένα σύστημα μπορεί να «προσποιηθεί» ότι έχει μια συγκεκριμένη IP διεύθυνση, κάτι που σημαίνει ότι οι IP διευθύνσεις δεν πρέπει να χρησιμοποιούνται ως το μοναδικό μέτρο για την προστασία ευαίσθητου περιεχομένου, οι περιορισμοί βάσει IP διεύθυνσης σε συνδυασμό με περιορισμούς αυθεντικοποίησης μπορούν να προσφέρουν ένα ικανοποιητικό επίπεδο ασφάλειας στο web site.

3.3.3 Βελτιωμένος έλεγχος αξιοπιστίας δεδομένων

Ένα καινούριο χαρακτηριστικό που ενσωματώθηκε στο σχεδιασμό του IIS 6.0 είναι ο kernel-mode[19] HTTP driver, HTTP.sys. Είναι σχεδιασμένος όχι μόνο ώστε να μεγιστοποιεί την απόδοση του web server, αλλά επίσης για να ενδυναμώνει σημαντικά την ασφάλεια του. Το HTTP.sys λειτουργεί ως μία «πύλη» για τα requests των χρηστών προς τον web server. Αρχικά, αναλύει το request και στη συνέχεια το αποστέλλει στις κατάλληλες user-level[20] διεργασίες. Ο περιορισμός των διεργασιών του IIS στο user-mode τις αποτρέπει από πρόσβαση σε κρίσιμους πόρους στον πυρήνα του συστήματος. Με τον τρόπο αυτό περιορίζεται σημαντικά ο κίνδυνος από πιθανούς επιτιθέμενους που σκοπεύουν να αποκτήσει εκτεταμένη πρόσβαση σε έναν server.

Ο kernel-mode driver ενσωματώνει αρκετούς μηχανισμούς ασφάλειας. Αυτοί οι μηχανισμοί περιλαμβάνουν προστασία από ενδεχόμενες υπερχειλίσεις buffers, βελτιωμένους logging μηχανισμούς και εκτεταμένη URL ανάλυση έτσι ώστε να ελέγχεται επαρκώς η εγκυρότητα των requests των χρηστών.

3.3.4 Σύνθετοι μηχανισμοί logging

Το εκτεταμένο logging είναι μία από τις κύριες προϋποθέσεις για την επιτυχή ανίχνευση και αντιμετώπιση ενός κρούσματος ασφάλειας. Η Microsoft έχει αναγνωρίσει την ανάγκη αυτή και έχει υλοποιήσει έναν εκτεταμένο και αξιόπιστο logging μηχανισμό στο HTTP.sys. Το HTTP.sys καταγράφει πληροφορίες σε ένα log file πριν στείλει το

[19] Kernel Mode: Ονομάζεται έτσι το λογικό επίπεδο στο οποίο τρέχουν κάποιες διεργασίες ενός λειτουργικού συστήματος οι οποίες έχουν πλήρη πρόσβαση σε οποιοδήποτε πόρο. Ο πυρήνας του λειτουργικού συστήματος τρέχει σε αυτό το επίπεδο.

[20] User Level: Σε αντιδιαστολή με τις διεργασίες που τρέχουν στο kernel mode, όλες οι υπόλοιπες διεργασίες τρέχουν στο user level και έχουν περιορισμένη πρόσβαση σε πόρους και δικό τους χώρο στη μνήμη.

request σε συγκεκριμένες διεργασίες. Το γεγονός αυτό εξασφαλίζει ότι μια περίπτωση σφάλματος καταγράφεται στα logs ακόμη και αν το σφάλμα αυτό προκάλεσε τον τερματισμό μιας διεργασίας. Μια καταχώρηση του log file αποτελείται από την ημερομηνία και την ώρα στην οποία συνέβη το σφάλμα, τις IP διευθύνσεις και θύρες της πηγής και του προορισμού, την έκδοση του πρωτοκόλλου, το HTTP verb[21], το URL, την κατάσταση του πρωτοκόλλου, το ID του web site και την HTTP.sys reason phrase. Η reason phrase παρέχει λεπτομερείς πληροφορίες για το λόγο για το οποίο συνέβη το λάθος, για το αν μπορεί να αποδοθεί σε μία κατάσταση timeout ή σε μια σύνδεση που εγκαταλείφθηκε από το application pool (3.3.5) λόγω του απρόσμενου τερματισμού της διεργασίας που είχε αναλάβει την εξυπηρέτηση ενός request.

3.3.5 Απομόνωση applications

Στις παλιότερες εκδόσεις του IIS (έκδοση 5.0 και νωρίτερα), το κόστος στην απόδοση από την απομόνωση των web εφαρμογές σε ξεχωριστές λογικές μονάδες καθιστούσε μη πρακτική τη χρησιμοποίηση του χαρακτηριστικού αυτού. Έτσι, συχνά η αποτυχία μιας web εφαρμογής είχε αλυσιδωτή επίδραση στις υπόλοιπες εφαρμογές που βρίσκονταν στον ίδιο web server. Οι βελτιώσεις στην απόδοση του συγκεκριμένου μηχανισμού συνοδεύτηκαν από αλλαγές στο σχεδιασμό της αρχιτεκτονικής της επεξεργασίας των requests. Από την έκδοση 6.0 του IIS και μετά είναι δυνατή πλέον η απομόνωση web εφαρμογών σε αυτοδιαχειριζόμενες μονάδες που ονομάζονται application pool. Κάθε application pool εξυπηρετείται από μία ή περισσότερες ανεξάρτητες διεργασίές. Αυτό επιτρέπει τον εντοπισμό της προέλευσης μιας αποτυχίας και εμποδίζει μια διεργασία που αντιμετωπίζει πρόβλημα να επηρεάσει τις υπόλοιπες.

[21] HTTP Verb: Το HTTP πρωτόκολλο ορίζει 8 μεθόδους οι οποίες αποκαλούνται verbs

Το χαρακτηριστικό αυτό ενισχύει σημαντικά την αξιοπιστία του server και επομένως των εφαρμογών που φιλοξενούνται σε αυτόν.

3.4 *Πρακτικές μεγιστοποίησης ασφάλειας*

Στον IIS, το επίπεδο ασφάλειας εξαρτάται σε μεγάλο βαθμό από τις επιπρόσθετες στρατηγικές ασφάλειας που θα εφαρμοστούν.

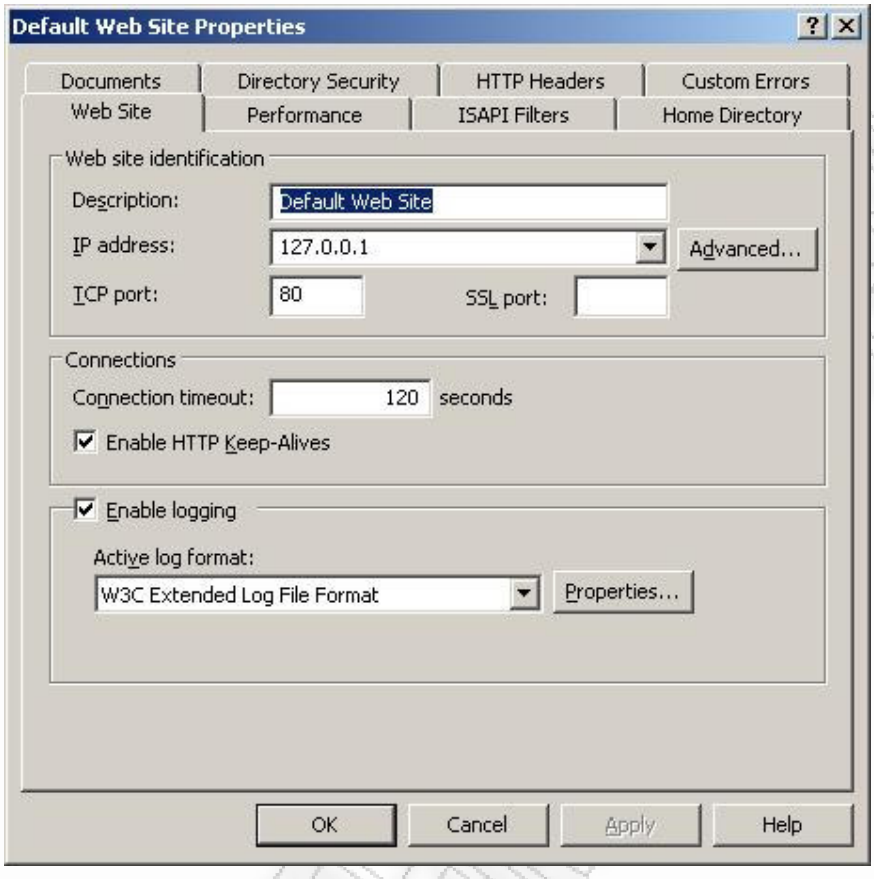
3.4.1 Ενημέρωση του φιλοξενούντος λειτουργικού με τις τελευταίες ενημερώσεις

Ο IIS ως στοιχείο του λειτουργικού συστήματος εξαρτά σημαντικό κομμάτι της ασφάλειας του στο κατά πόσο το λειτουργικό σύστημα στο οποίο είναι εγκατεστημένος (που στην περίπτωση αυτή θα είναι κάποια έκδοση των Microsoft Windows) είναι ενημερωμένο με τις τελευταίες διορθώσεις ασφάλειας και service packs. Επομένως, είναι σημαντικό το φιλοξενούν λειτουργικό σύστημα να ενημερώνεται εγκαίρως με τις πιο πρόσφατες ενημερώσεις.

3.4.2 Απενεργοποίηση πρόσβασης στο "Default Web site"

Η εγκατάσταση του IIS, by default, δημιουργεί ένα "default web site", το οποίο αν δε χρησιμοποιείται είναι προτιμότερο να απενεργοποιηθεί γιατί δημιουργεί μία ακόμη δίοδο επιθέσεων.

Η απενεργοποίηση του default web site μπορεί να γίνει μέσω του «Internet Services Manager» (ο οποίος βρίσκεται στο Control Panel -> Administrative Tools). Στην οθόνη των ιδιοτήτων του "default web site", το default web site πρέπει να ανατεθεί στην localhost διεύθυνση (η οποία είναι η 127.0.0.1). Αυτό γίνεται εάν στο πεδίο "IP address" (στο tab «Web site") διαγραφεί το «All Unassigned» και εισαχθεί το 127.0.0.1.



Εικόνα 13 Οθόνη ιδιοτήτων του Default Web site. Στην παραπάνω οθόνη έχει ρυθμιστεί η πρόσβαση να επιτρέπεται μόνο μέσα από τον ίδιο το server.

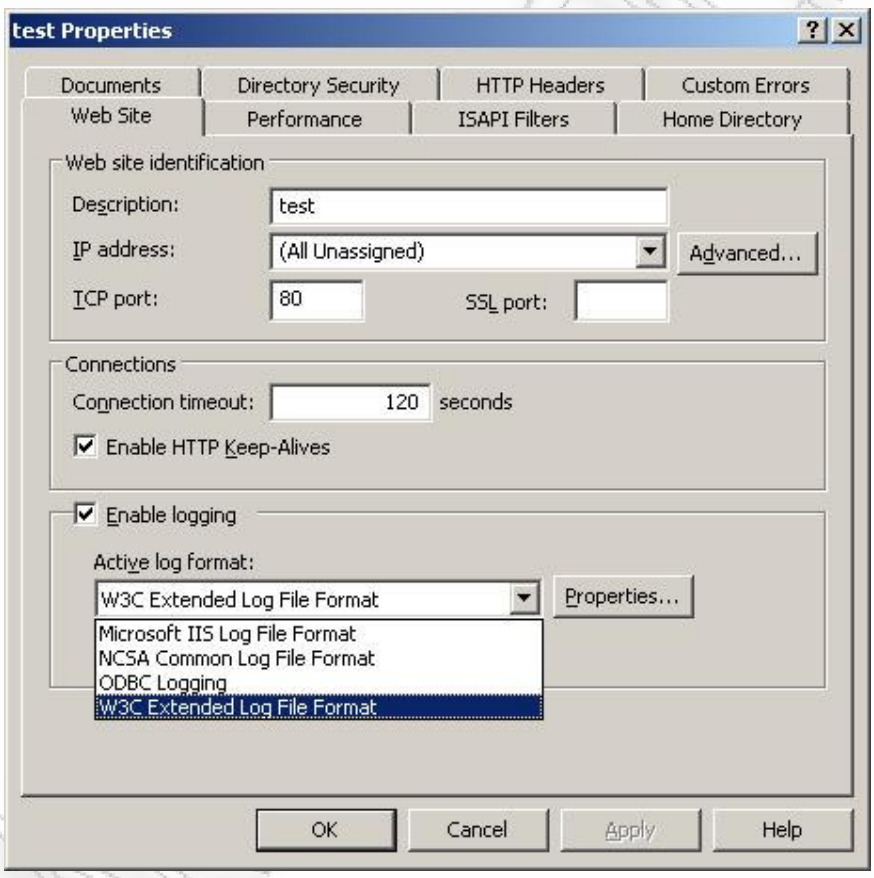
Η παραπάνω επιλογή ρυθμίζει το «default web site», έτσι ώστε να είναι προσβάσιμο μόνο από τον Web Browser που τρέχει στον ίδιο το server και από πουθενά αλλού στο δίκτυο. Η απενεργοποίηση του «default web site» είναι προτιμότερη από τη διαγραφή του, γιατί μπορεί να αποδειχθεί χρήσιμο στη συνέχεια.

3.4.3 Ενεργοποίηση logging μηχανισμού

Για τον καλύτερο έλεγχο του server είναι σημαντική η ύπαρξη logging μηχανισμού που να καταγράφει χρήσιμες πληροφορίες όσον αφορά τα requests που φτάνουν στο server. Οι πληροφορίες αυτές μπορούν να αξιοποιηθούν και να χρησιμοποιηθούν για την ανίχνευση επιθέσεων ή αδυναμιών ασφάλειας μιας web εφαρμογής. Οι εκδόσεις του IIS από την 6 και μετά επιτρέπουν τη ρύθμιση ανεξάρτητου logging μηχανισμού για κάθε web site που υπάρχει στο server.

Στην οθόνη ιδιοτήτων ενός web site υπάρχει δυνατότητα επιλογής της μορφής των log αρχείων που θα παράγονται για το συγκεκριμένο web site. Συγκεκριμένα για τα log αρχεία υποστηρίζονται τα εξής formats:

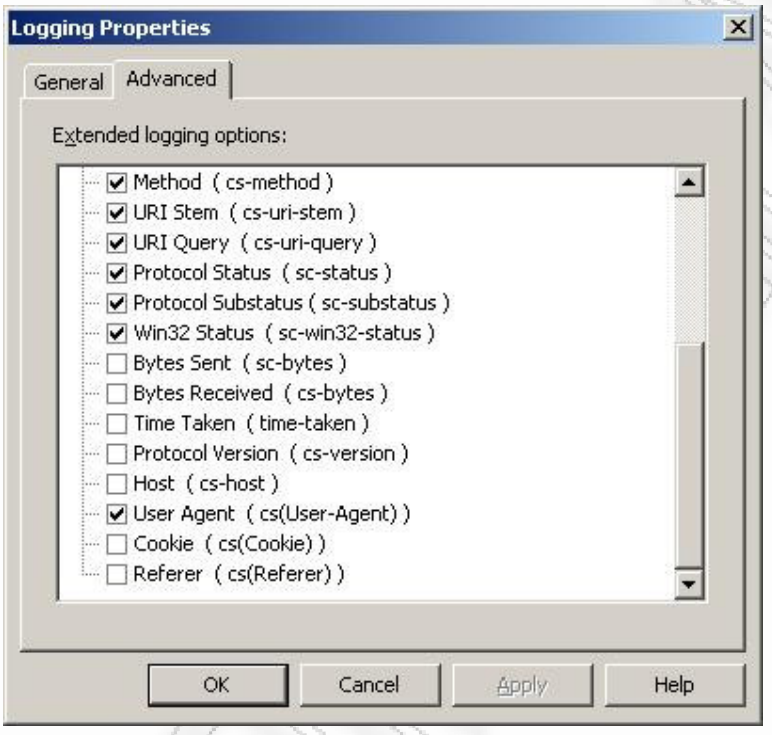
- Microsoft IIS Log File Format
- NCSA Common Log File Format
- ODBC Logging (εγγραφή των log σε πίνακα κάποιας βάσης δεδομένων)
- W3C Extended Log File Format



Εικόνα 14 Οθόνη ιδιοτήτων ενός web site. Επιλογή του format των log αρχείων

By default, για κάθε νέο δημιουργηθέν web site, ο IIS ενεργοποιεί ένα αρκετά εκτεταμένο logging χρησιμοποιώντας το format W3C Extended Log File. Έχοντας επιλεγμένο αυτό το format, μπορούν στις ιδιότητές του, να οριστούν η περίοδος ανά την οποία θα παράγεται καινούριο log αρχείο, αλλά και το ποια ακριβώς στοιχεία

θα καταγράφονται στο log αρχείο. Παρά το γεγονός ότι την προκαθορισμένη λειτουργία είναι προεπιλεγμένα τα σημαντικότερα στοιχεία, προτείνεται να επιλέγονται όλες οι διαθέσιμες επιλογές όσον αφορά τα στοιχεία καταγραφής στο log αρχείο.



Εικόνα 15 Οθόνη ρύθμισης στοιχείων καταγραφής στο log αρχείο για το format W3C Extended Log File

3.4.4 Κατάλληλη χρήση partitions και directories

Για τις web εφαρμογές που φιλοξενούνται από τον IIS είναι προτιμότερο να αποθηκεύονται τα αρχεία των web σελίδων σε ξεχωριστό partition του σκληρού δίσκου. Δηλαδή, το λειτουργικό σύστημα και τα αρχεία της web εφαρμογής είναι προτιμότερο να βρίσκονται σε διαφορετικά partitions. Η τοποθέτηση κάθε λογικού γκρουπ αρχείων σε ξεχωριστά partitions (ξεχωριστά logical drives) μειώνει τον κίνδυνο επιθέσεων. Για παράδειγμα, εάν οι web σελίδες και τα Web scripts τοποθετηθούν στον ίδιο κατάλογο, ένας επιτιθέμενος που αποκτά πρόσβαση στην περιοχή των web σελίδων μπορεί να αποκτήσει δυνητικά πρόσβαση και στα scripts.

Το partition αποθήκευσης των Web σελίδων μπορεί να είναι το ίδιο partition με αυτό στο οποίο αποθηκεύονται τα log files. Στο partition αυτό είναι σημαντικό να μην αποθηκεύονται αρχεία του λειτουργικού συστήματος και αν είναι δυνατόν καμία γενικώς εφαρμογή. Αυτή η προσέγγιση προτείνεται για το λόγο ότι ο IIS έχει ένα ιστορικό προβλημάτων σχετικά με μη εξουσιοδοτημένη πρόσβαση σε αρχεία εκτός του καταλόγου των Web pages και γιατί υπάρχουν ακόμα αρκετές καταχωρήσεις όπως για παράδειγμα η:

```
../scripts/../../winnt/system32/cmd.exe+/c+dir+\
```

σε logs και οι οποίες αντιστοιχούν σε προσπάθειες επιτιθέμενων να εκτελέσουν απομακρυσμένα εντολές του λειτουργικού συστήματος. Όταν το λειτουργικό σύστημα και οι web σελίδες είναι σε διαφορετικό partition, ο server είναι προστατευμένος από αυτού του είδους τις επιθέσεις για το λόγο ότι ο πιθανός επιτιθέμενος δεν μπορεί να ανακατευθύνει την αίτησή του μεταξύ διαφορετικών drives.

Είναι επίσης εξαιρετικά σημαντικό, το partition των Web σελίδων να είναι τύπου NTFS και επίσης οι επιλογές ασφάλειας του να είναι οι κατάλληλες. Στο partition που είναι αποθηκευμένες οι Web σελίδες θα πρέπει να μην επιτρέπεται η πρόσβαση στο group Everyone, ενώ θα πρέπει να επιτρέπεται η πρόσβαση μόνο στα groups Administrators, SYSTEM και Authenticated Users. Όσον αφορά τα δικαιώματα, τα groups Administrators και SYSTEM θα πρέπει να έχουν τα μέγιστα δικαιώματα, δηλαδή να είναι ενεργοποιημένη η επιλογή «Full Control», ενώ το group Authenticated Users θα πρέπει να έχει ενεργοποιημένο μόνο το δικαίωμα χρήσης «List Folder Contents». Στο κατάλογο τον Web scripts θα πρέπει να είναι ενεργοποιημένο για το group των Authenticated Users επίσης το δικαίωμα «Read and Execute», ενώ στον κατάλογο των log files θα πρέπει να ενεργοποιηθεί το δικαίωμα «Read» μόνο για τους Administrators.

3.4.5 Επιλογή κατάλληλου hosting συστήματος

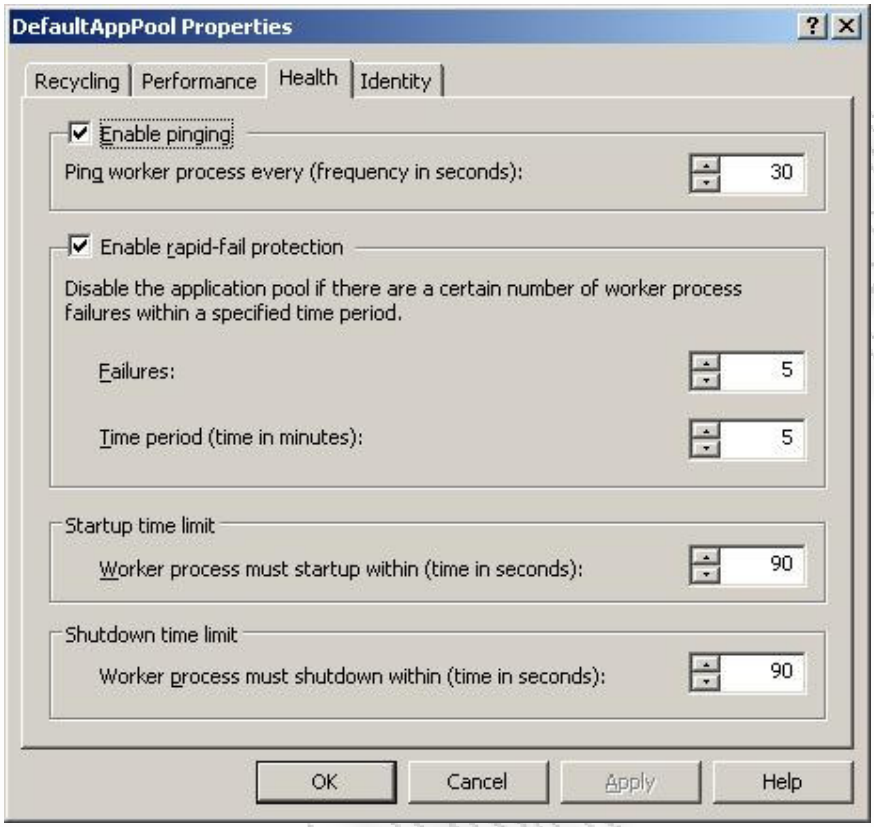
Το σύστημα που φιλοξενεί τον IIS δεν πρέπει ταυτόχρονα να φιλοξενεί και το domain controller ενός εσωτερικού δικτύου. Επίσης, θα πρέπει η κάρτα δικτύου που δέχεται τα HTTP requests από το Internet να μην είναι συνδεδεμένη σε κάποιο εσωτερικό δίκτυο. Αν υπάρχει εσωτερικό δίκτυο, τότε αυτό πρέπει να εξυπηρετείται ιδανικά από διαφορετική κάρτα δικτύου.

3.4.6 Ενεργοποίηση Rapid-Fail προστασίας

Ο IIS μπορεί να ρυθμιστεί έτσι ώστε να σταματά ή να επανεκκινεί αυτόματα διεργασίες (Application pools) των οποίων οι εφαρμογές έχουν αποτύχει επανειλημμένα μέσα σε ένα συγκεκριμένο χρονικό διάστημα. Με τον τρόπο αυτό ο server αποκτά ένα επιπλέον μέτρο προστασίας από επανειλημμένες αποτυχίες, οι οποίες είναι πολύ πιθανό να αποτελούν ένδειξη επίθεσης. Αυτό το χαρακτηριστικό ονομάζεται Rapid-Fail προστασία.

Η Rapid Fail προστασία μπορεί να ενεργοποιηθεί από τον IIS manager ως εξής:

1. Στον IIS manager, expand του local computer
2. Expand του «Application Pools»
3. Επιλογή του Application pool στο οποίο θέλουμε να ενεργοποιήσουμε το χαρακτηριστικό της Rapid Fail προστασίας
4. Επιλογή των properties αυτού
5. Στο Health tab, ενεργοποίηση της επιλογής «Enable rapid-fail protection»
6. Στο Failures box, εισαγωγή του αριθμού των αποτυχιών στις worker διεργασίες στον οποίο πρέπει να γίνει τερματισμός της διεργασίας
7. Στο Time Period box, εισαγωγή του αριθμού των λεπτών για τα οποία μετρώνται οι αποτυχίες στις διεργασίες.



Εικόνα 16 Οθόνη ρύθμισης rapid-fail προστασίας

3.4.7 Σύνθετες ρυθμίσεις για την ανάλυση των URL requests

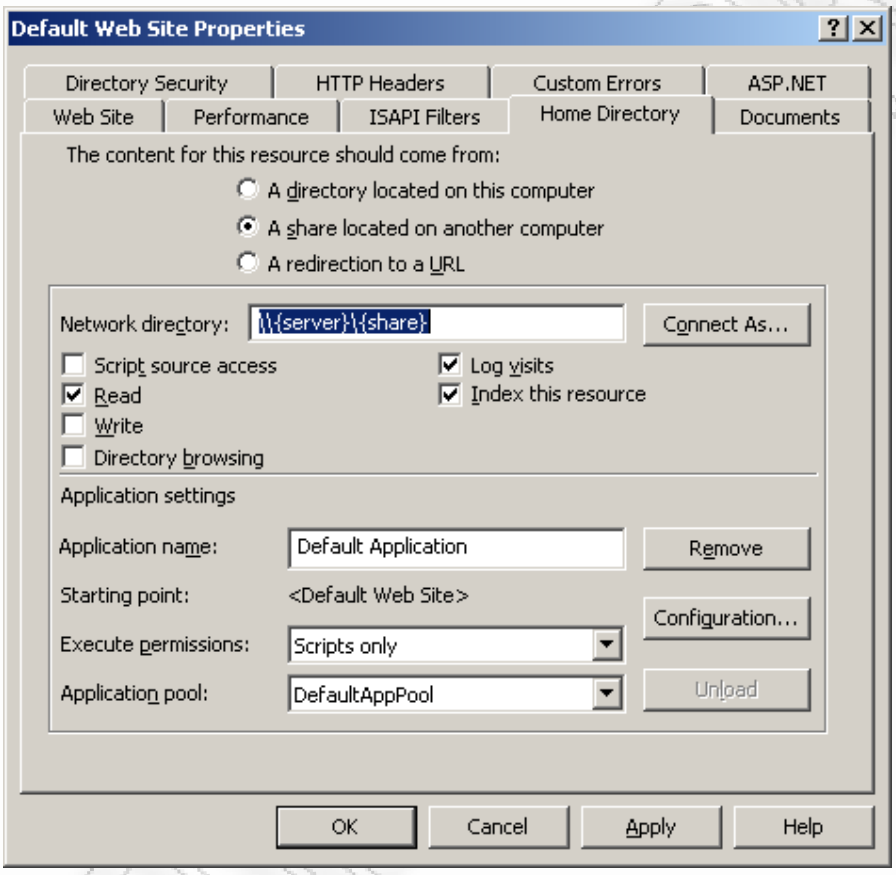
Για να προλάβει την εκμετάλλευση μιας ενδεχόμενης αδυναμίας υπερχείλισης buffer που μπορεί να προκύψει στο μέλλον, η Microsoft κατέφυγε στην αρχή ασφάλειας της «άμυνας σε βάθος». Η τακτική αυτή επιτυγχάνεται με την ενσωμάτωση συγκεκριμένων δυνατοτήτων ανάλυσης URL στο ρεπερτόριο των χαρακτηριστικών του HTTP.sys. Αυτές οι δυνατότητες μπορούν να ρυθμιστούν περαιτέρω, κατά την κρίση του διαχειριστή, με την κατάλληλη τροποποίηση καταχωρήσεων της registry και συγκεκριμένα στο path: HKLM\System\CurrentControlSet\Services\HTTP\Parameters.

3.4.8 Κατάλληλη επιλογή δικαιωμάτων πρόσβασης

Στην κατηγορία «Home directory» των ιδιοτήτων του Web site συναντώνται οι εξής επιλογές που αφορούν τα δικαιώματα πρόσβασης:

- «Script source access»: Επιτρέπει την πρόσβαση στον κώδικα των δυναμικών σελίδων του web site μέσω HTTP. Η συγκεκριμένη επιλογή είναι σχεδόν πάντοτε απαραίτητο να είναι απενεργοποιημένη.
- «Read»: Επιτρέπει την ανάγνωση των αρχείων του web site. Αν το web site ή ένας υποκατάλογος του web site αποτελείται αποκλειστικά από δυναμικές σελίδες, η επιλογή αυτή μπορεί να απενεργοποιηθεί.
- «Write»: Επιτρέπει την εγγραφή αρχείων μέσω HTTP. Εάν ο server είναι προσβάσιμος μέσω Web, αυτή η επιλογή θα πρέπει να είναι απενεργοποιημένη, εκτός κάποιων ειδικών περιπτώσεων.
- «Directory browsing»: Επιτρέπει την πλοήγηση στη δομή των καταλόγων, δηλαδή όλα τα ονόματα των αρχείων και των καταλόγων είναι ορατά στους χρήστες. Είναι προτιμότερο η επιλογή αυτή να είναι απενεργοποιημένη, καθώς δίνει ακόμη μία διέξοδο σε πιθανούς επιτιθέμενους.
- «Execute Permissions»: Η επιλογή αυτή καθορίζει τα δικαιώματα εκτέλεσης. Η επιλογή «Scripts and Executables» επιτρέπει την εκτέλεση εφαρμογών από τον IIS (συμπεριλαμβανομένου αρχείων EXE και DLL). Η επιλογή αυτή, εφόσον είναι ενεργοποιημένη, είναι πολύ πιθανό να γίνει εκμεταλλεύσιμη από πιθανούς επιτιθέμενους, γιατί αυτές οι εφαρμογές εφόσον περιέχουν bugs μπορούν να προκαλέσουν πρόσφορο έδαφος για επιθέσεις. Η επιλογή «Scripts only» επιτρέπει την εκτέλεση μόνο scripts, ενώ η επιλογή «None» δεν επιτρέπει ούτε την εκτέλεση scripts ούτε εκτελέσιμων. Προτείνεται, η επιλογή «Scripts only» να είναι ενεργοποιημένη

για τους υποκαταλόγους εκείνους που περιέχουν δυναμικές σελίδες, ενώ η επιλογή «None» να είναι ενεργοποιημένη για τα directories με στατικά αρχεία (όπως π.χ. ένας κατάλογος που περιέχει μόνο εικόνες). Η επιλογή «Scripts and Executables» πρέπει να αποφεύγεται όπου αυτό είναι εφικτό.



Εικόνα 17 Οθόνη επιλογής ρυθμίσεων δικαιωμάτων για έναν συγκεκριμένο κατάλογο ενός web site

3.4.9 Χρήση του SSL πρωτοκόλλου

Η χρήση του SSL πρωτοκόλλου από τον IIS ενεργοποιείται και απενεργοποιείται με χρήση του Internet Service Manager. Το SSL παρέχει μια «χειραψία ασφαλείας» που χρησιμοποιείται για την εκκίνηση μιας TCP/IP σύνδεσης. Αυτή η «χειραψία» έχει ως αποτέλεσμα τη συμφωνία μεταξύ client και server για το επίπεδο ασφάλειας που θα χρησιμοποιήσουν. Από εκείνη τη στιγμή και μετά, ο μόνος ρόλος του SSL είναι η κρυπτογράφηση και

αποκρυπτογράφηση των δεδομένων που μεταδίδονται μεταξύ client και server μέσω του Internet.

Για την ενεργοποίηση της SSL ασφάλειας σε μια web εφαρμογή του IIS, πρέπει να ακολουθηθούν τα εξής βήματα:

1. Δημιουργία ενός αρχείου ζεύγους κλειδιών και ενός sign request αρχείου.
2. Αίτηση για πιστοποίηση από μια αρχή πιστοποίησης. Αυτή τη στιγμή, η μόνη γνωστότερη αρχή πιστοποίησης είναι η Verisign (http://www.verisign.com)
3. Εγκατάσταση της πιστοποίησης στο server.
4. Ενεργοποίηση της SSL ασφάλειας στον επιθυμητό κατάλογο ενός website.

Τα ακόλουθα σημεία είναι σημαντικά όταν ενεργοποιηθεί η SSL ασφάλεια:

- Η χρήση του SSL μπορεί να ρυθμιστεί ώστε να είναι ενεργοποιημένη σε όλες τις σελίδες ενός web site ή σε συγκεκριμένους υποκαταλόγους.
- Αφού ενεργοποιηθεί και ρυθμιστεί κατάλληλα η SSL ασφάλεια, μόνο οι clients που έχουν ενεργοποιημένο το SSL μπορούν να επικοινωνήσουν με τα SSL-ενεργοποιημένα web directories.
- Για τα URLs που δείχνουν σε αρχεία που βρίσκονται σε directories που είναι SSL-ενεργοποιημένα, πρέπει να χρησιμοποιείται το πρόθεμα https:// αντί του http://.

Η χρήση του SSL δεν αφορά την αυθεντικοποίηση. Εξασφαλίζει μόνο ότι όταν αποστέλλεται κάτι μέσω του Internet, κανένας δεν μπορεί να «κρυφακούσει» τα δεδομένα που ανταλλάσσονται. Επίσης, δεν εξασφαλίζει ότι οι πληροφορίες που ανταλλάσσονται είναι αυθεντικές, όπως π.χ. στην περίπτωση των πιστωτικών καρτών.

3.4.10 «Έξυπνη» ονοματολογία των web αρχείων και directories

Η ονοματολογία των web αρχείων και των καταλόγων μπορεί να ακολουθήσει κάποιες έξυπνες στρατηγικές, έτσι ώστε να αποτρέψει συνηθισμένες στρατηγικές που ακολουθούν επιτιθέμενοι σε web sites:

- Χρήση ενός ασυνήθιστου ονόματος για τον κατάλογο των scripts. Ονόματα όπως «scripts», «cgi-bin», «exchange» και «bin» είναι εξαιρετικά συνηθισμένα και τα αυτόματα εργαλεία επίθεσης αναζητούν για αυτά.
- Μετονομασία της επέκτασης όλων των αρχείων scripts με κάτι ασυνήθιστο., π.χ. μετονομασία του αρχείου myscript.asp σε myscript.dum. Πληροφοριακά, η μετονομασία όλων των .asp αρχείων σε .html λειτουργεί κανονικά, χωρίς να απαιτεί περαιτέρω ρύθμιση στην απεικόνιση των επεκτάσεων των αρχείων στον IIS.
- Compile των scripts σε dll αρχεία. Πέρα από το γεγονός ότι με το compile των scripts ο κώδικάς τους προστατεύεται από αντίστροφη ανάλυση (reverse engineering), υπάρχει και μεγάλο κέρδος στην απόδοση του συστήματος, καθώς ο compiled κώδικας ειδικά για το ASP.NET είναι έως και 20 φορές γρηγορότερος.

3.5 *Διαπιστωμένα προβλήματα ασφάλειας και «έξυπνες» στρατηγικές επίθεσης*

3.5.1 Αδυναμία ασφάλειας που επιτρέπει εκτέλεση κακόβουλα σχεδιασμένων Active Server Pages

Η συγκεκριμένη αδυναμία ασφάλειας του IIS επιτρέπει στον επιτιθέμενο, εφόσον την εκμεταλλευτεί σωστά μέχρι και την απόκτηση πλήρους ελέγχου του συστήματος. Ο επιτιθέμενος μπορεί

στη συνέχεια να εγκαταστήσει προγράμματα, να δει, να αλλάξει ή να διαγράψει δεδομένα ή να δημιουργήσει νέους λογαριασμούς με πλήρη δικαιώματα.

Για την πραγματοποίηση του τύπου αυτού της επίθεσης, ο επιτιθέμενος πρέπει να έχει στην κατοχή του έναν έγκυρο συνδυασμό λογαριασμού / κωδικού ή ο server να έχει ρυθμιστεί έτσι ώστε να επιτρέπει ακόμη και σε ανώνυμους χρήστες το ανέβασμα αρχείων όπως .ASP σελίδων σε web εφαρμογές. Ένας επιτιθέμενος μπορεί να σχεδιάσει και στη συνέχεια να ανεβάσει στο server μια προσεκτικά σχεδιασμένη asp σελίδα που να προκαλέσει υπερχείλιση buffer στον IIS και στη συνέχεια να εκτελέσει κώδικα της επιλογής του.

Αντισταθμιστικοί παράγοντες για τους κινδύνους από αυτήν την αδυναμία ασφάλειας είναι τα εξής:

- Στον IIS 5.1, οι ASP ενεργές εφαρμογές, by default, τρέχουν στην "Pooled Out Of Process" εφαρμογή, το οποίο σημαίνει ότι τρέχουν στο DLLHOST.exe, που τρέχει στο περιβάλλον των περιορισμένων δικαιωμάτων του λογαριασμού IWAM_<Όνομα Συστήματος>.
- Στον IIS 6.0, by default, η ASP δεν είναι ενεργοποιημένη. Παρ' όλα αυτά ακόμη και όταν είναι ενεργοποιημένη, τρέχει στο περιβάλλον της W3WP.exe διεργασίας η οποία τρέχει με τα περιορισμένα δικαιώματα χρήσης του NetworkService λογαριασμού.

Η συγκεκριμένη αδυναμία ασφάλειας ανακοινώθηκε από την εταιρεία παραγωγής στις 11/7/2006, οπότε δόθηκε στη δημοσιότητα και το patch διόρθωσης που αφορά τις εκδόσεις 5.1 και 6.0 του IIS.

3.5.2 Αδυναμία ασφάλειας στον handler των WebDAV[22] XML μηνυμάτων

Η συγκεκριμένη αδυναμία ασφάλειας επιτρέπει στον επιτιθέμενο που την εκμεταλλεύθηκε επιτυχώς να καταναλώσει όλη τη διαθέσιμη μνήμη και CPU χρόνο του συστήματος. Αυτή η συμπεριφορά μπορεί να προκαλέσει «denial of service», ενώ για την ανάκτηση της λειτουργίας του IIS service απαιτείται επανεκκίνηση του. Ο server παραμένει ευάλωτος στη συγκεκριμένη αδυναμία ασφάλειας ακόμη και μετά την επανεκκίνηση του service.

Ο επιτιθέμενος μπορεί να προκαλέσει "denial of service" σε έναν server που τρέχει τον IIS και WebDav αποστέλλοντας ένα προσεκτικά σχεδιασμένο WebDAV request. Αιτία της αδυναμίας αυτής είναι ο τρόπος λειτουργίας του XML Parser της Microsoft, MSXML πριν την έκδοση 3.0. Το WebDAV και άλλες εφαρμογές που χρησιμοποιούν τον Microsoft XML Parser, δεν μπορούν να περιορίσουν τον αριθμό των attributes τις οποίες ο Microsoft XML Parser θα προσπαθούσε να επεξεργαστεί στα ληφθέντα documents.

Αντισταθμιστικοί παράγοντες στον κίνδυνο που εισάγει αυτή η αδυναμία ασφάλειας είναι:

- το γεγονός ότι μπορεί να γίνει εκμεταλλεύσιμη μόνο εάν ο επιτιθέμενος έχει τη δυνατότητα να εγκαταστήσει ένα Web session με τον server-στόχο.
- το γεγονός ότι σε καμία από τις τρεις εκδόσεις του IIS που εξετάζονται (5.1, 6.0, 7.0) το WebDAV δεν είναι ενεργοποιημένο by default.

Η συγκεκριμένη αδυναμία ασφάλειας επηρεάζει τις εκδόσεις 5.1 και 6.0 του IIS. Ανακοινώθηκε από τη Microsoft στις 12/10/2004, οπότε δόθηκε στη δημοσιότητα και το patch διόρθωσης.

---

[22] Σύνολο επεκτάσεων για το πρωτόκολλο HTTP που επιτρέπει σε χρήστες συνεργαζόμενοι να επεξεργάζονται και να διαχειρίζονται αρχεία μέσω κάποιου Web Server (π.χ. IIS)

3.5.3 Αδυναμία ασφάλειας σχετικά με λανθασμένη αναγνώριση της προέλευσης ενός request

Η συγκεκριμένη αδυναμία ασφάλειας αφορά τη δυνατότητα «εξαπάτησης» του IIS στην περίπτωση που η μεταβλητή συστήματος «SERVER_NAME» έχει την τιμή «localhost». Στη συγκεκριμένη περίπτωση, με χρήση ενός κατάλληλα σχεδιασμένου HTTP request, είναι δυνατό να «νομίσει» ο IIS ότι το συγκεκριμένο request προέρχεται από έναν client που βρίσκεται στο ίδιο μηχάνημα με τον ίδιο, ενώ στην πραγματικότητα ο client αυτός να είναι απομακρυσμένος (remote). Από την αδυναμία αυτή μπορεί να επηρεαστούν web εφαρμογές και web services που βασίζουν την ασφάλειά τους στην εγκυρότητα αυτής της server μεταβλητής.

Στη μεταβλητή «SERVER_NAME» μπορεί να αποκτηθεί πρόσβαση προγραμματιστικά μέσω του:

```
request.servervariables("SERVER_NAME")
```

στην ASP και του:

```
HttpContext.Current.Request.ServerVariables("SERVER_NAME")
```

στο .NET, ενώ άλλες γλώσσες παρέχουν άλλες μεθόδους για την πρόσβαση σε αυτή τη server μεταβλητή. Όλοι οι τρόποι πρόσβασης είναι το ίδιο ευάλωτοι στη συγκεκριμένη αδυναμία.

Εάν το HTTP request προέρχεται από απομακρυσμένο client, τότε η μεταβλητή «SERVER_NAME» επιστρέφει την IP διεύθυνση του web server. Εάν το HTTP request προέρχεται από την ίδια IP με τον web server (όπως το request που προέρχεται από έναν αυθεντικοποιημένο χρήστη που κάνει browse σε μία web εφαρμογή από το ίδιο μηχάνημα στο οποίο αυτή φιλοξενείται), τότε το «request.servervariables("SERVER_NAME") επιστρέφει «localhost». Αυτό το γεγονός χρησιμοποιείται ως «απόδειξη» για τις web εφαρμογές και τα web services ότι το πρόσωπο που κάνει browse τον web server στην πραγματικότητα κάνει browse από τον ίδιο το web server. Οι web εφαρμογές και τα web services μπορεί για

παράδειγμα να χρησιμοποιούν αυτή την απόδειξη για να παρουσιάζουν το διαχειριστικό μέρος ενός web site.

Η τεχνική περιγραφή της συγκεκριμένης server μεταβλητής είναι «Το host name του server, το DNS ψευδώνυμο, ή η IP διεύθυνση όπως θα εμφανιζόταν σε αυτοαναφορικά URLs». Επομένως, πολλές φορές η συγκεκριμένη μεταβλητή χρησιμοποιείται για τον προσδιορισμό της IP διεύθυνσης του web server σε κώδικες εφαρμογών. Το γεγονός αυτό ανοίγει δρόμους για επιθέσεις που μπορεί να περιλαμβάνουν υποκλοπή cookies, ανακατεύθυνση δεδομένων και άλλα σχετικά με τα URLs προβλήματα.

Μια διαδικασία που θα μπορούσε να ακολουθηθεί έτσι ώστε να διαπιστωθεί στην πράξη η συγκεκριμένη αδυναμία ασφάλειας είναι η εξής:

[1]. Δημιουργία μιας ASP σελίδας με όνομα test.asp στο root του IIS και προσθήκη του ακόλουθου κώδικα:

```
<% response.write request.servervariables("SERVER_NAME")
%>
```

[2]. Προσπάθεια πρόσβασης στη συγκεκριμένη σελίδα από έναν απομακρυσμένο server μέσω telnet με χρησιμοποίηση του ακόλουθου HTTP request:

```
GET /test.asp HTTP/1.0
```

[3]. Η απάντηση είναι η IP διεύθυνση του web server, όπως ακριβώς θα περίμενε κανείς.

[4]. Προσπάθεια πρόσβασης της σελίδας μέσα από τον ίδιο τον web server με χρήση του HTTP request:

```
GET /test.asp HTTP/1.0
```

[5]. Η απάντηση είναι «localhost» όπως θα περίμενε κανείς

[6]. Προσπάθεια πρόσβασης στην ίδια σελίδα από έναν απομακρυσμένο server πάλι μέσω telnet. Αυτή τη φορά, χρήστη του HTTP request:

GET http://localhost/test.asp HTTP/1.0

[7]. Η απάντηση είναι «localhost». Ο IIS μόλις έχει εξαπατηθεί και «νομίζει» ότι το συγκεκριμένο request ήρθε από έναν χρήστη που έκανε browse από τον ίδιο το web server.

Η συγκεκριμένη αδυναμία ασφάλειας αφορά την έκδοση 6.0 του IIS.

4. .NET Framework

4.1 *Εισαγωγή*

Το .NET Framework είναι ένα στοιχείο λογισμικού που είναι κομμάτι των λειτουργικών συστημάτων Microsoft Windows. Διαθέτει μια μεγάλη γκάμα έτοιμων λύσεων για συνηθισμένες προγραμματιστικές απαιτήσεις και χειρίζεται την εκτέλεση εφαρμογών που έχουν γραφτεί ειδικά για το Framework.

Οι έτοιμες λύσεις που σχηματίζουν τη Base Class βιβλιοθήκη καλύπτουν ένα μεγάλο εύρος προγραμματιστικών αναγκών σε τομείς που περιλαμβάνουν: user interface, πρόσβαση σε δεδομένα, συνδεσιμότητα με βάσεις δεδομένων, κρυπτογραφία, ανάπτυξη web εφαρμογών, αριθμητικούς αλγόριθμους και επικοινωνίες δικτύου. Η Base Class βιβλιοθήκη χρησιμοποιείται από τους προγραμματιστές που συνδυάζοντάς την με το δικό τους κώδικα μπορούν να αναπτύσσουν εφαρμογές.

Οι εφαρμογές που γράφονται στο .NET Framework εκτελούνται σε ένα περιβάλλον που χειρίζεται τις απαιτήσεις τους σε χρόνο εκτέλεσης (runtime). Αυτό το runtime περιβάλλον, που είναι επίσης μέρος του .NET Framework είναι γνωστό ως Common Language Runtime (CLR). Το CLR παίζει το ρόλο μιας εικονικής μηχανής, έτσι ώστε οι προγραμματιστές να μην ασχολούνται καθόλου με τη διαχείριση του υλικού, αφού το ρόλο αυτό τον αναλαμβάνει το ίδιο το CLR. Το CLR παρέχει επίσης άλλες σημαντικές υπηρεσίες όπως μηχανισμούς ασφάλειας, διαχείριση μνήμης και χειρισμό εξαιρέσεων (exceptions). Η Base Class βιβλιοθήκη μαζί με το CLR συνθέτουν το .NET Framework.

Το ASP.NET αποτελεί κομμάτι του .NET Framework που επικεντρώνεται σε Web εφαρμογές και επιτρέπει στους προγραμματιστές να δημιουργήσουν δυναμικές web εφαρμογές.

Το .NET Framework περιλαμβάνεται στα Windows Server 2003, στα Windows Server 2008 και στα Windows Vista και μπορεί να εγκατασταθεί στις περισσότερες από τις παλιότερες εκδόσεις των Windows. Η τρέχουσα έκδοσή του είναι η 3.5.

4.2 *Μηχανισμός ασφάλειας*

Το .NET Framework παρέχει ένα πλήθος μηχανισμών έτσι ώστε να προστατεύει πόρους και κώδικα από μη εξουσιοδοτημένες ενέργειες.

4.2.1 ASP.NET Αυθεντικοποίηση

Ως αυθεντικοποίηση νοείται η διαδικασία συλλογής διαπιστευτηρίων όπως είναι το όνομα και ο κωδικός ενός χρήστη και ο έλεγχος της εγκυρότητας αυτών των διαπιστευτηρίων ενώπιον μιας αρχής. Εάν τα διαπιστευτήρια είναι έγκυρα, η οντότητα που έστειλε τα διαπιστευτήρια θεωρείται αυθεντικοποιημένη.

Το ASP.NET υλοποιεί την αυθεντικοποίηση μέσω παρόχων αυθεντικοποίησης, που είναι οι λογισμικές μονάδες (modules) που είναι υπεύθυνες για την αυθεντικοποίηση των διαπιστευτηρίων του αιτούμενου. Το ASP.NET υποστηρίζει τους εξής παρόχους αυθεντικοποίησης:

- Forms authentication: Ένα σύστημα στο οποίο τα requests που δεν έχουν ήδη αυθεντικοποιηθεί ανακατευθύνονται (redirect) σε μία HTML φόρμα. Εάν η εφαρμογή, μέσω της φόρμας αυτής, αυθεντικοποιήσει επιτυχώς το request, το σύστημα παράγει ένα ticket το οποίο θεωρείται ως το αποδεικτικό ότι ο συγκεκριμένος χρήστης έχει αυθεντικοποιηθεί επιτυχώς. Τις περισσότερες φορές αυτό το ticket εμπεριέχεται μέσα σε ένα cookie. Υποστηρίζεται όμως και Forms authentication χωρίς χρήση cookies στην οποία παράγεται ένα κλειδί για ανάκτηση της αυθεντικοποιημένης οντότητας και το οποίο

χρησιμοποιείται ως παράμετρος στα urls των επόμενων requests.

- Passport authentication: Κεντρικοποιημένη υπηρεσία αυθεντικοποίησης που παρέχεται από τη Microsoft και προσφέρει ένα μοναδικό λογαριασμό που είναι έγκυρος για την πρόσβαση σε προστατευόμενες υπηρεσίες διαφορετικών sites που υποστηρίζουν αυτόν τον τρόπο αυθεντικοποίησης.
- Windows authentication: Το ASP.NET χρησιμοποιεί Windows authentication σε συνδυασμό με την αυθεντικοποίηση του IIS. Όταν η αυθεντικοποίηση του IIS ολοκληρωθεί, το ASP.NET χρησιμοποιεί την αυθεντικοποιημένη από τον IIS οντότητα για να εγκρίνει ή όχι την πρόσβαση.

Η επιλογή του τρόπου αυθεντικοποίησης τον οποίο θα χρησιμοποιεί μια ASP.NET εφαρμογή μπορεί να γίνει είτε μέσω των configuration αρχείων της εφαρμογής είτε προγραμματιστικά, με χρήση κώδικα.

4.2.2 ASP.NET εξουσιοδότηση

Σκοπός της διαδικασίας της εξουσιοδότησης είναι να αποφασίσει εάν πρέπει να δοθεί σε μια οντότητα ο αιτούμενος τύπος πρόσβασης σε έναν συγκεκριμένο πόρο. Υπάρχουν δύο επίπεδα όσον αφορά την εξουσιοδότηση πρόσβασης:

- Εξουσιοδότηση αρχείου: Πραγματοποιείται από το στοιχείο FileAuthorizationModule, που είναι ενεργό μόνο όταν χρησιμοποιείται η Windows authentication. Πραγματοποιεί ένα access control list (ACL) έλεγχο στον αιτούμενο πόρο για να αποφασίσει εάν ένας χρήστης δικαιούται να έχει πρόσβαση.
- Εξουσιοδότηση URL: Πραγματοποιείται από το στοιχείο URLAuthorizationModule, που αντιστοιχεί χρήστες και ρόλους σε τμήματα του URL namespace. Το module αυτό υποστηρίζει τόσο θετικές όσο και αρνητικές δηλώσεις εξουσιοδότησης, που σημαίνει ότι το module μπορεί να χρησιμοποιηθεί έτσι ώστε να επιτρέπει ή να αρνείται επιλεκτικά την πρόσβαση σε

συγκεκριμένα κομμάτια του URL namespace για συγκεκριμένα σύνολα, χρήστες ή ρόλους.

Το URLAuthorizationModule είναι διαθέσιμο για χρήση κάθε στιγμή. Απαιτεί μόνο την τοποθέτηση λίστας χρηστών ή/και ρόλων στα στοιχεία <allow> ή <deny> του <authorization> στοιχείου του configuration αρχείου μιας ASP.NET εφαρμογής.

Η πρόσβαση σε ένα συγκεκριμένο κατάλογο μπορεί να ελεγχθεί αν τοποθετηθεί στον κατάλογο αυτό ένα configuration αρχείο το οποίο να περιέχει ένα <authorization> tag. Οι ρυθμίσεις που τίθενται όσον αφορά την εξουσιοδότηση για αυτό τον κατάλογο εφαρμόζονται αυτόματα και σε όλους τους υποκαταλόγους του, εάν τα configuration αρχεία των υποκαταλόγων δεν τις υπερκαλύπτουν. Η γενική σύνταξη για το <authorization> tag είναι η εξής:

```
<[element] [users] [roles] [verbs]/>
```

Το attribute element είναι υποχρεωτικό. Είτε το attribute users είτε το attribute roles πρέπει να περιλαμβάνεται υποχρεωτικά, ενώ το attribute verbs είναι προαιρετικό.

Τα επιτρεπόμενα elements είναι τα <allow> και <deny> που δίνουν και αποκλείουν την πρόσβαση αντίστοιχα. Τα attributes του authorization tag περιγράφονται στον παρακάτω πίνακα:

Attribute	Περιγραφή
Roles	Ορίζει έναν ή περισσότερους ρόλους-στόχους στους οποίους θα εφαρμοστούν οι ρυθμίσεις πρόσβασης του συγκεκριμένου κανόνα.
Users	Ορίζει ένα ή περισσότερους χρήστες-στόχους στους οποίους θα εφαρμοστούν οι ρυθμίσεις πρόσβασης του συγκεκριμένου κανόνα.
Verbs	Ορίζει τα HTTP verbs στα οποία εφαρμόζεται αυτός ο κανόνας, όπως GET, HEAD και POST

Επιπρόσθετα με το ονόματα των οντοτήτων, υπάρχουν δύο ειδικές οντότητες:

- * : Αναφέρεται σε όλες τις οντότητες
- ? : Αναφέρεται στην ανώνυμη οντότητα

Στο παρακάτω παράδειγμα επιτρέπεται η πρόσβαση στον χρήστη Kim και στα μέλη του ρόλου Admins, ενώ απαγορεύεται η πρόσβαση στον χρήστη John και σε όλους τους ανώνυμους χρήστες:

```
<authorization>
    <allow users="Kim"/>
    <allow roles="Admins"/>
    <deny users="John"/>
    <deny users="?"/>
</authorization>
```

Τόσο το attribute users όσο και το roles μπορούν να περιέχουν πολλαπλές οντότητες χρησιμοποιώντας το κόμμα ως διαχωριστικό. Για παράδειγμα:

```
<allow users="John, Kim, contoso\Jane"/>
```

Στο ακόλουθο παράδειγμα επιτρέπεται σε όλους η χρήση GET, αλλά μόνο ο χρήστης Kim μπορεί να χρησιμοποιήσει το POST.

```
<authorization>
    <allow verb="GET" users="*"/>
    <allow verb="POST" users="Kim"/>
    <deny verb="POST" users="*"/>
</authorization>
```

Οι ακόλουθοι κανόνες είναι αυτοί που εφαρμόζονται από το .NET Framework κατά τη διαδικασία εξουσιοδότησης:

- Οι δηλώσεις που βρίσκονται σε configuration αρχεία σε χαμηλότερα επίπεδα καταλόγου έχουν προτεραιότητα απέναντι στις δηλώσεις που βρίσκονται σε υψηλότερα επίπεδα καταλόγου. Το Framework αποφασίζει ποιος κανόνας έχει προτεραιότητα κατασκευάζοντας μια λίστα όλων των κανόνων που αφορούν ένα συγκεκριμένο URL, με τους πιο πρόσφατους (πλησιέστερους στην ιεραρχία) κανόνες να είναι στην κορυφή αυτής της λίστας.
- Δεδομένου ενός συνόλου κανόνων για ένα URL, το .NET Framework ξεκινά από την κορυφή της λίστας και ελέγχει τους κανόνες μέχρις ότου βρεθεί το πρώτο ταίριασμα. Να σημειωθεί ότι η προκαθορισμένη ρύθμιση για το ASP.NET είναι ένας <allow users=”*”> κανόνας, που εξουσιοδοτεί όλους ανεξαιρέτως τους χρήστες. Εάν δεν υπάρξει κάποιο ταίριασμα μετά τη διάτρεξη όλης της λίστας το request εξουσιοδοτείται επιτυχώς. Εάν υπάρξει ταίριασμα και αυτό είναι ένα στοιχείο <deny>, τότε επιστρέφεται ο κωδικός κατάστασης 401 (Access Denied) του πρωτοκόλλου HTTP. Οι ASP.NET εφαρμογές μπορούν εύκολα χρησιμοποιώντας ένα στοιχείο <deny users=”*”> στο κορυφαίο επίπεδο να αποτρέψουν την ανεξέλεγκτη πρόσβαση.

Υποστηρίζεται επίσης το <location> tag το οποίο μπορεί να χρησιμοποιηθεί για τον προσδιορισμό ενός συγκεκριμένου αρχείου ή καταλόγου στο οποίο εφαρμόζονται οι ρυθμίσεις αυτού του tag. Για παράδειγμα:

```
<location path="subdir1">
     <system.web>
          <authorization>
```

```
                <allow users ="*" />
            </authorization>
        </system.web>
</location>
```

Με τον τρόπο αυτό, ο κανόνας εφαρμόζεται μόνο στον κατάλογο subdir1.

4.2.3 Ασφάλεια πρόσβασης κώδικα

Η ασφάλεια πρόσβασης κώδικα (Code Access Security) αποτελεί ένα μηχανισμό με σκοπό να ελέγχεται η πρόσβαση που έχει ο κώδικας σε προστατευμένους πόρους και λειτουργίες.

Η ασφάλεια πρόσβασης κώδικα συντίθεται από τα ακόλουθα στοιχεία:

- *code*
- *permissions*
- *permission requests*
- *demands*
- *link demands*
- *permission sets*
- *code groups*
- *evidence*
- *policy*

Ολόκληρος ο managed κώδικας[23] υπόκειται στην ασφάλεια πρόσβασης κώδικα. Όταν φορτώνεται μια assembly, της παρέχεται ένα σύνολο από δικαιώματα πρόσβασης κώδικα που καθορίζει σε

[23] Ο κώδικας που τρέχει υπό τη διαχείριση του CLR

ποιους τύπους πόρων μπορεί να έχει πρόσβαση και ποιους τύπους λειτουργιών μπορεί να εκτελέσει.

- Ένα *permission* αντιπροσωπεύει δικαίωμα πρόσβασης σε έναν προστατευόμενο πόρο ή τη δυνατότητα εκτέλεσης μιας προστατευόμενης λειτουργίας. Το .NET Framework παρέχει πολλές permission κλάσεις όπως η FileIOPermission (για ενέργειες σε αρχεία και καταλόγους), και UIPermission (για δικαιώματα σχετικά με χρήση user interfaces και clipboard).
- Τα *permission requests* είναι ένα χαρακτηριστικό της ασφάλειας πρόσβασης κώδικα που δίνει στον κώδικα τη δυνατότητα να γνωρίζει ο ίδιος τα περί ασφάλειας. Αυτά τα requests καθιστούν εφικτά τα εξής:
  - Απαιτούν τα ελάχιστα δικαιώματα που πρέπει να λάβει ο κώδικας για να τρέξει.
  - Εξασφαλίζουν ότι ο κώδικας δε θα λάβει περισσότερα δικαιώματα από αυτά που πραγματικά χρειάζεται.

  Για παράδειγμα:

```
[assembly:FileIOPermissionAttribute
 (SecurityAction.RequestMinimum, Write="C:\\test.tmp")]
[assembly:PermissionSet
 (SecurityAction.RequestOptional, Unrestricted=false)]
(SecurityAction.RequestRefused)]
```

  Στο παραπάνω παράδειγμα, ο κώδικας που περιέχει την παραπάνω δήλωση δε θα εκτελεστεί αν δε λάβει δικαίωμα να γράψει το C:\test.tmp. Εάν ο κώδικας κάποτε αντιμετωπίσει πολιτική ασφάλειας τέτοια που δεν του παραχωρεί αυτό το permission, παράγεται ένα PolicyException και ο κώδικας δεν εκτελείται. Με τον τρόπο αυτό ο προγραμματιστής μπορεί να είναι σίγουρος ότι ο κώδικάς του θα λάβει αυτό το δικαίωμα και δε χρειάζεται να ανησυχεί για πιθανά λάθη που μπορεί να

προκύψουν εξαιτίας λιγότερων permissions από τα απαιτούμενα.

Επίσης, με το keyword RequestMinimum δηλώνεται ότι δε χρειάζονται επιπλέον permissions. Χωρίς αυτό, ο κώδικας θα λάμβανε τα permissions που θα αποφάσιζε η πολιτική ασφάλειας για αυτόν. Παρά το γεγονός ότι τα επιπλέον permissions δεν προκαλούν κινδύνους, στην περίπτωση που υπάρχει ένα bug, το να υπάρχουν λιγότερα permissions ίσως βοηθήσει στο κλείσιμο της «τρύπας». Γενικά, η κτήση permissions που δε χρειάζεται ο κώδικας μπορεί να οδηγήσει σε προβλήματα ασφάλειας.

- Το *Demand* καθορίζει τη διάτρεξη του stack εκτέλεσης από την ασφάλεια κώδικα πρόσβασης κατά το χρόνο εκτέλεσης. Όλα τα στοιχεία του stack πρέπει να έχουν το απαραίτητο permission ή evidence για να επιτραπεί η πρόσβαση σε κάποιο προστατευόμενο πόρο. Το Demand πραγματοποιείται σε κάθε κλήση καθώς κάθε φορά το stack μπορεί να αποτελείται από διαφορετικά assemblies. Εάν μία μέθοδος καλείται συνεχώς, ο έλεγχος αυτός θα συμβαίνει κάθε φορά. Το Demand είναι ανθεκτικό απέναντι σε luring attacks[24], καθώς ενδεχόμενη προσπάθεια μη εξουσιοδοτημένου κώδικα να διεισδύσει θα ανιχνευθεί. Ένα παράδειγμα μιας μεθόδου προστατευμένης με Demand στη γλώσσα C# είναι το ακόλουθο:

```
[CustomPermissionAttribute(SecurityAction.Demand, Unrestricted
= true)]

public static string ReadData()

{
```

[24] Luring attack – Επίθεση σκόπος της οποίας είναι η εκμετάλλευση ενός στοιχείου με ισχυρότερα δικαιώματα από τον επιτιθέμενο με τελικό στόχο την πραγματοποιήση κάποιας ενέργειας εκ μέρους του.

```
    //Read from a custom resource.
}
```

- Το Link Demand πραγματοποιείται στο just-in-time (JIT) compile χρόνο και ελέγχει μόνο τον άμεσο καλούντα. Δεν ελέγχει ολόκληρο το stack εκτέλεσης όπως το Demand. Αφού πραγματοποιηθεί αυτός ο έλεγχος στον compile χρόνο, δεν επαναλαμβάνεται ο ίδιος έλεγχος όσες φορές και να γίνει η ίδια κλήση. Με τον τρόπο αυτό δεν παρέχεται προστασία από luring attacks. Με το Link Demand, το interface είναι ασφαλές αλλά οποιοσδήποτε κώδικας περάσει τον έλεγχο μπορεί ενδεχομένως να παραβιάσει την ασφάλεια επιτρέποντας κακόβουλο κώδικα να καλέσει τον αρχικό κώδικα χρησιμοποιώντας τον εξουσιοδοτημένο κώδικα. Επομένως, η χρήση του Link Demand δε συνίσταται εκτός αν όλες οι πιθανές αδυναμίες μπορούν να αποφευχθούν. Ένα παράδειγμα μιας μεθόδου προστατευμένης με Link Demand στη γλώσσα C# είναι το ακόλουθο:

```
[CustomPermissionAttribute(SecurityAction.LinkDemand)]
public static string ReadData()
{
    // Access a custom resource.
}
```

- Το permission set είναι ένα σύνολο από permissions. Για παράδειγμα μπορούν το FileIOPermission και το UIPermission να προστεθούν σε ένα νέο permission set. Το permission set μπορεί να περιλαμβάνει έναν οποιοδήποτε αριθμό από permissions. Τα permission sets FullTrust, LocalIntranet,

Internet, Execution και Nothing είναι κάποια από τα προϋπάρχοντα permission sets στο .NET Framework. Το FullTrust permission set έχει όλα τα υπάρχοντα permissions, ενώ το Nothing δεν περιέχει κανένα απολύτως permission.

- Το code group αποτελεί μια λογική ομαδοποίηση του κώδικα βάσει μιας συγκεκριμένης συνθήκης που καθορίζει τη συμμετοχή ή όχι στην ομάδα αυτή. Για παράδειγμα, ο κώδικας από το http://www.somewebsite.com/ μπορεί να ανήκει σε ένα code group, ενώ ο κώδικας μιας συγκεκριμένης assembly μπορεί να ανήκει σε ένα άλλο code group. Υπάρχουν προκαθορισμένα code groups όπως το My_Computer_Zone, το LocalIntranet_Zone, το Internet_Zone κλπ. Όπως και με τα permission sets, είναι δυνατό να δημιουργηθούν νέα code groups για την ικανοποίηση των εκάστοτε αναγκών βάσει του evidence που παρέχεται από το .NET Framework.
- Το evidence μπορεί να είναι οποιαδήποτε πληροφορία σχετίζεται με μία assembly και χρησιμοποιείται από το .NET Framework για να αναγνωρίσει μία assembly.

  Οι διαφορετικοί τύποι των evidences είναι:

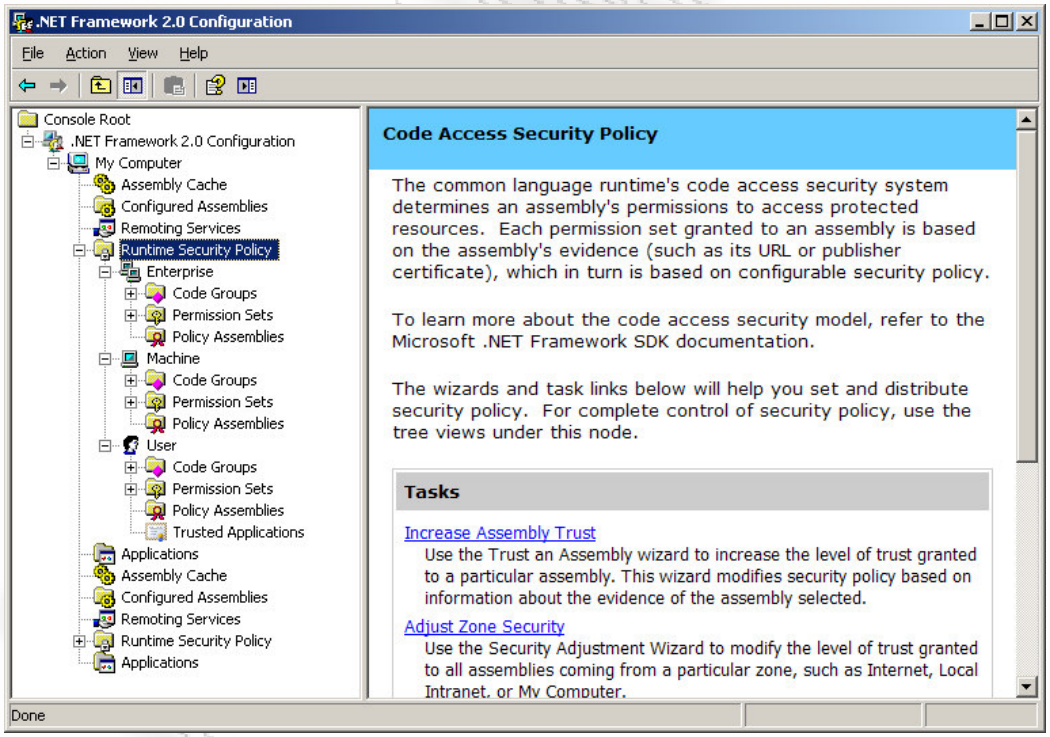
  - Application directory – Ο κατάλογος στον οποίο βρίσκεται η assembly
  - Publisher – Η ψηφιακή υπογραφή του «εκδότη» της assembly (απαιτεί η assembly να είναι ψηφιακά υπογεγραμμένη)
  - URL – το πλήρες URL από το οποίο η βιβλιοθήκη μεταφορτώθηκε
  - Site – το hostname της URL
  - Zone – οι καθορισμένες ζώνες ασφάλειας (από τον Internet Explorer)
  - Hash – ένα κρυπτογραφικό hash της assembly, που αντιπροσωπεύει μια συγκεκριμένη έκδοση

- Strong Name – το πιστοποιητικό X.509 που αναγνωρίζει μοναδικά έναν «εκδότη»

Ο προγραμματιστής μπορεί επίσης να δημιουργήσει το δικό του evidence.

- Η security policy είναι το ρυθμιζόμενο σύνολο κανόνων που ακολουθεί το CLR όταν αποφασίζει τα permissions που θα αναθέσει στον κώδικα. Υπάρχουν τέσσερα policy επίπεδα:
  - Enterprise – policy για την οικογένεια των μηχανών που ανήκουν σε μια εγκατάσταση Active Directory
  - Machine – policy για το τρέχον σύστημα
  - User – policy για τον συνδεδεμένο χρήστη
  - AppDomain – policy για το domain της εκτελεσθείσας εφαρμογής

  Η διαχείριση των τριών πρώτων policies μπορεί να γίνει από το .NET Configuration Tool.



Εικόνα 18 Οθόνη διαχείρισης των policy επιπέδων - .NET Configuration Tool

Το AppDomain policy διαχειρίζεται μέσω κώδικα για το τρέχον domain της εφαρμογής.

Η ασφάλεια κώδικα πρόσβασης παρουσιάζει το evidence μιας assembly σε κάθε policy και στη συνέχεια λαμβάνει την τομή των permissions(που είναι τα κοινά permissions μεταξύ όλων των παραχθέντων permission sets) ως τα permissions που τελικά ανατίθενται στην assembly.

By default, τα policies Enterprise, User και AppDomain δίνουν πλήρη εμπιστοσύνη (επιτρέπουν σε όλες τις assemblies να έχουν όλα τα permissions), ενώ το Machine policy είναι πιο περιοριστικό. Από τη στιγμή που λαμβάνεται υπόψη η τομή των permissions, στην default λειτουργία, το τελικό permission set αποφασίζεται από το Machine policy.

Εκτός της απόφασης του permission set για μία assembly, η ασφάλεια πρόσβασης κώδικα εκτελεί τις ακόλουθες λειτουργίες:

- Δίνει τη δυνατότητα στους διαχειριστές των συστημάτων να καθορίσουν τακτικές ασφαλείας συσχετίζοντας permission sets με code groups.
- Δίνει τη δυνατότητα στον κώδικα να ζητήσει (μέσω των permission requests) τα δικαιώματα που χρειάζεται για να τρέξει, όπως επίσης και τα δικαιώματα που θα ήταν χρήσιμο να έχει, και καθορίζει ποια permissions δεν πρέπει να έχει ποτέ ο κώδικας.
- Δίνει την δυνατότητα στον κώδικα να απαιτεί αυτοί που τον καλούν να έχουν συγκεκριμένα δικαιώματα.
- Δίνει τη δυνατότητα στον κώδικα να απαιτεί αυτοί που τον καλούν να κατέχουν μια ψηφιακή υπογραφή, επιτρέποντας έτσι να τον καλούν μόνο όσοι ανήκουν σε ένα συγκεκριμένο οργανισμό ή τοποθεσία.
- Επιβάλλει περιορισμούς στον κώδικα κατά το χρόνο εκτέλεσης συγκρίνοντας τα παραχωρημένα permissions κάθε καλούντος στο stack εκτέλεσης με τα permissions που πρέπει να έχουν οι καλούντες.

4.2.4 Κρυπτογραφικές Υπηρεσίες

Τα δημόσια δίκτυα όπως το Internet δεν αποτελούν μέσο ασφαλούς επικοινωνίας μεταξύ οντοτήτων. Η επικοινωνία μέσω τέτοιων δικτύων είναι ευάλωτη σε ανάγνωση ή ακόμη και τροποποίηση από τρίτες μη εξουσιοδοτημένες οντότητες. Συμπληρωματικά με την κρυπτογράφηση αρχείων και την κρυπτογράφηση σε έναν τοπικό δίσκο, η κρυπτογραφία βοηθάει στη δημιουργία κρυπτογραφημένων καναλιών επικοινωνίας πάνω από κανάλια που σε διαφορετική περίπτωση θα ήταν ανασφαλή, παρέχοντας εμπιστευτικότητα, ακεραιότητα δεδομένων και αυθεντικοποίηση.

Το .NET Framework παρέχει τέσσερις τρόπους κρυπτογράφησης οι οποίοι μπορούν να χρησιμοποιηθούν ανεξάρτητα ή σε συνδυασμό έτσι ώστε να σχηματίσουν ένα κρυπτογραφικό σχήμα ασφάλειας. Οι τρόποι αυτοί είναι οι εξής:

Κρυπτογράφηση ιδιωτικού κλειδιού: Οι αλγόριθμοι ιδιωτικού κλειδιού χρησιμοποιούν ένα μοναδικό κλειδί για την κρυπτογράφηση και αποκρυπτογράφηση των δεδομένων. Η επιτυχία του αλγορίθμου εξαρτάται από την προστασία πρόσβασης στο κλειδί από μη εξουσιοδοτημένες οντότητες, καθώς οποιοσδήποτε έχει το κλειδί μπορεί να το χρησιμοποιήσει ώστε να αποκρυπτογραφήσει δεδομένα. Η κρυπτογράφηση ιδιωτικού κλειδιού αναφέρεται επίσης ως συμμετρική κρυπτογράφηση καθώς το ίδιο κλειδί χρησιμοποιείται τόσο για την κρυπτογράφηση όσο και την αποκρυπτογράφηση. Οι αλγόριθμοι ιδιωτικού κλειδιού είναι εξαιρετικά γρήγοροι (συγκριτικά με τους αλγόριθμους δημοσίου κλειδιού) και είναι κατάλληλοι για την εκτέλεση κρυπτογραφικών μετασχηματισμών σε μεγάλους όγκους δεδομένων.

Στην τυπική περίπτωση, οι αλγόριθμοι ιδιωτικού κλειδιού χρησιμοποιούνται ώστε να κρυπτογραφούν ένα μπλοκ δεδομένων κάθε φορά. Οι διάφορες παραλλαγές του αλγόριθμου (RC2, DES, TripleDES και Rijndael) μετασχηματίζουν κρυπτογραφικά ένα μπλοκ δεδομένων n bytes σε ένα μπλοκ κρυπτογραφημένων bytes. Εάν

πρέπει να κρυπτογραφηθεί μια αλληλουχία από bytes, τότε αυτό πρέπει να γίνει μπλοκ προς μπλοκ. Επειδή το n είναι μικρό (n = 8 bytes για τους RC2, DES και TripleDES, n = 16, n = 24 ή n = 32 για τον Rijndael), τιμές μεγαλύτερες του n πρέπει να κρυπτογραφηθούν χρησιμοποιώντας ένα μπλοκ κάθε φορά.

Οι κλάσεις ιδιωτικού κλειδιού που παρέχονται από την Base Class βιβλιοθήκη του .NET Framework χρησιμοποιούν μια λειτουργία αλυσίδωσης που ονομάζεται αλυσίδα μπλοκ κρυπτογράφησης (Cipher Block Chaining – CBC), που χρησιμοποιεί ένα κλειδί και ένα διάνυσμα αρχικοποίησης (Initialization Vector – IV) για να πραγματοποιήσει κρυπτογραφικούς μετασχηματισμούς στα δεδομένα. Για ένα δοσμένο ιδιωτικό κλειδί k, η κρυπτογράφηση που δε χρησιμοποιεί διάνυσμα αρχικοποίησης θα κρυπτογραφήσει το ίδιο μπλοκ καθαρού κειμένου με το ίδιο μπλοκ κρυπτογραφημένου κειμένου. Για την αντιμετώπιση αυτού του προβλήματος, οι πληροφορίες από το προηγούμενο μπλοκ αναμειγνύονται στη διαδικασία κρυπτογράφησης του επόμενου μπλοκ. Με τον τρόπο αυτό, η έξοδος από την κρυπτογράφηση δύο πανομοιότυπων μπλοκ είναι διαφορετική. Επειδή η τεχνική αυτή χρησιμοποιεί το προηγούμενο μπλοκ για να κρυπτογραφήσει το επόμενο μπλοκ, το διάνυσμα αρχικοποίησης χρησιμοποιείται για την κρυπτογράφηση του πρώτου μπλοκ δεδομένων. Με την χρησιμοποίηση αυτού του συστήματος είναι αδύνατο να χρησιμοποιήσει κάποιος π.χ. συνήθη header μηνύματα για να δημιουργήσει αντίστροφα ένα κλειδί.

Ένας τρόπος να αξιοποιήσει κάποιος τα κρυπτογραφημένα δεδομένα είναι να εκτελέσει μια εκτεταμένη αναζήτηση όλων των πιθανών κλειδιών. Παρ' όλα αυτά βάσει του μεγέθους πάντα του κλειδιού που χρησιμοποιήθηκε για την κρυπτογράφηση, αυτός ο τρόπος αναζήτησης είναι εξαιρετικά χρονοβόρος ακόμη και για ισχυρότατους υπολογιστές και επομένως ανεφάρμοστος. Τα μεγαλύτερα μεγέθη κλειδιών είναι δυσκολότερα να αποκρυπτογραφηθούν. Παρά το γεγονός ότι η κρυπτογράφηση θεωρητικά δεν αποκλείει τη

δυνατότητα για κάποιον να ανακτήσει τα κρυπτογραφημένα δεδομένα, ανεβάζει απαγορευτικά το κόστος να πραγματοποιηθεί κάτι τέτοιο. Εάν για παράδειγμα χρειάζονται τρεις μήνες για την εκτέλεση μιας εκτεταμένης αναζήτησης για την ανάκτηση δεδομένων που έχουν πρακτική σημασία μόνο για λίγες μέρες, τότε δεν έχει νόημα η αναζήτηση αυτή.

Το μειονέκτημα της κρυπτογράφησης ιδιωτικού κλειδιού είναι ότι προϋποθέτει ότι οι δύο οντότητες έχουν συμφωνήσει ένα κλειδί και ένα διάνυσμα αρχικοποίησης και έχουν επικοινωνήσει τις τιμές τους. Επίσης, το κλειδί πρέπει να κρατηθεί κρυφό από μη εξουσιοδοτημένους χρήστες. Εξαιτίας αυτών των προβλημάτων, η κρυπτογράφηση ιδιωτικού κλειδιού χρησιμοποιείται σε συνδυασμό με την κρυπτογράφηση δημοσίου κλειδιού, έτσι ώστε να ανταλλαχθούν με ασφάλεια οι τιμές του κλειδιού και του διανύσματος αρχικοποίησης.

Το .NET Framework παρέχει τις ακόλουθες κλάσεις που υλοποιούν αλγορίθμους κρυπτογράφησης ιδιωτικού κλειδιού:

- DESCryptoServiceProvider
- RC2CryptoServiceProvider
- RijndaelManaged
- TripleDESCryptoServiceProvider

Κρυπτογράφηση δημόσιου κλειδιού: Η κρυπτογράφηση δημοσίου κλειδιού χρησιμοποιεί ένα ιδιωτικό κλειδί που πρέπει να κρατηθεί κρυφό από μη εξουσιοδοτημένους χρήστες και ένα δημόσιο κλειδί το οποίο μπορεί να είναι γνωστό σε οποιονδήποτε. Το δημόσιο και το ιδιωτικό κλειδί συνδέονται μαθηματικά. Τα δεδομένα που έχουν κρυπτογραφηθεί με το δημόσιο κλειδί μπορούν να αποκρυπτογραφηθούν μόνο με το ιδιωτικό κλειδί, ενώ τα δεδομένα που έχουν υπογραφεί με το ιδιωτικό κλειδί μπορούν να αυθεντικοποιηθούν μόνο με το δημόσιο κλειδί. Το δημόσιο κλειδί μπορεί να είναι γνωστό στον καθένα. Χρησιμοποιείται για την

κρυπτογράφηση των δεδομένων που αποστέλλονται στον κάτοχο του ιδιωτικού κλειδιού. Και τα δυο κλειδιά είναι μοναδικά για μια συνεδρία επικοινωνίας. Οι κρυπτογραφικοί αλγόριθμοι δημοσίου κλειδιού είναι επίσης γνωστοί ως ασυμμετρικοί αλγόριθμοι επειδή άλλο κλειδί απαιτείται για την κρυπτογράφηση των δεδομένων και άλλο για την αποκρυπτογράφηση των δεδομένων.

Οι κρυπτογραφικοί αλγόριθμοι δημοσίου κλειδιού χρησιμοποιούν ένα σταθερό μέγεθος buffer ενώ οι αλγόριθμοι ιδιωτικού κλειδιού χρησιμοποιούν buffer μεταβλητού μεγέθους. Οι αλγόριθμοι δημοσίου κλειδιού δεν μπορούν να χρησιμοποιηθούν για την αλυσίδωση δεδομένων σε streams όπως γίνεται με τους αλγόριθμους ιδιωτικού κλειδιού επειδή μόνο μικρές ποσότητες δεδομένων μπορούν να κρυπτογραφηθούν. Επομένως, οι ασυμμετρικές λειτουργίες δε χρησιμοποιούν το ίδιο μοντέλο streaming με τις συμμετρικές λειτουργίες.

Δύο οντότητες (έστω Alice και Bob) μπορούν να χρησιμοποιήσουν την κρυπτογράφηση δημοσίου κλειδιού ως εξής: Αρχικά, η Alice δημιουργεί ένα ζεύγος δημόσιου / ιδιωτικού κλειδιού. Εάν ο Bob θέλει να στείλει στην Alice ένα κρυπτογραφημένο μήνυμα, τη ρωτάει για το δημόσιο κλειδί της. Η Alice στέλνει στον Bob το δημόσιο κλειδί της ακόμη και μέσω ενός ανασφαλούς δικτύου και ο Bob χρησιμοποιεί αυτό το κλειδί για την κρυπτογράφηση ενός μηνύματος. (Εάν ο Bob έλαβε το κλειδί της Alice μέσω ενός ανασφαλούς καναλιού, όπως ένα δημόσιο δίκτυο, ο Bob πρέπει να εξακριβώσει με την Alice ότι έχει τη σωστή έκδοση του δημόσιου κλειδιού της). Ο Bob στέλνει το κρυπτογραφημένο μήνυμα στην Alice και αυτή το αποκρυπτογραφεί χρησιμοποιώντας το ιδιωτικό κλειδί της.

Κατά τη μετάδοση του δημόσιου κλειδιού της Alice παρ' όλα αυτά μία μη εξουσιοδοτημένη οντότητα μπορεί να διαβάσει το δημόσιο κλειδί ή και να διαβάσει τα κρυπτογραφημένα δεδομένα που στέλνονται από τον Bob. Παρ' όλα αυτά η οντότητα αυτή δεν μπορεί να

αποκρυπτογραφήσει τα δεδομένα αυτά με το δημόσιο κλειδί. Το μήνυμα μπορεί να αποκρυπτογραφηθεί μόνο με το ιδιωτικό κλειδί της Alice το οποίο όμως δεν μεταδίδεται. Η Alice δε χρησιμοποιεί το ιδιωτικό κλειδί της για την κρυπτογράφηση της απάντησής της προς τον Bob, καθώς οποιοσδήποτε που κατέχει το δημόσιο κλειδί θα μπορούσε να αποκρυπτογραφήσει το μήνυμα. Εάν η Alice θέλει να στείλει ένα μήνυμα στον Bob, ρωτά τον Bob για το δημόσιο κλειδί του και κρυπτογραφεί το μήνυμα χρησιμοποιώντας αυτό το δημόσιο κλειδί. Ο Bob τότε αποκρυπτογραφεί το μήνυμα χρησιμοποιώντας το δικό του ιδιωτικό κλειδί.

Σε σενάριο πραγματικού κόσμου, η Alice και ο Bob χρησιμοποιούν κρυπτογράφηση δημόσιου κλειδιού (ασύμμετρη) για τη μεταφορά ενός μυστικού (συμμετρικού) κλειδιού και κρυπτογράφηση ιδιωτικού κλειδιού για το υπόλοιπο της συνεδρίας τους.

Η κρυπτογράφηση δημοσίου κλειδιού έχει πολύ μεγαλύτερο εύρος τιμών για το κλειδί και επομένως είναι λιγότερο ευάλωτη σε επιθέσεις που επιχειρούν να δοκιμάσουν κάθε πιθανό κλειδί. Το δημόσιο κλειδί είναι εύκολο να μεταδοθεί επειδή δε χρειάζεται να προστατευθεί. Οι αλγόριθμοι δημοσίου κλειδιού μπορούν να χρησιμοποιηθούν για τη δημιουργία ψηφιακών υπογραφών έτσι ώστε να πιστοποιείται η ταυτότητα του αποστολέα των δεδομένων. Παρ' όλα αυτά οι αλγόριθμοι δημοσίου κλειδιού είναι εξαιρετικά αργοί (σε σύγκριση με τους αλγόριθμους ιδιωτικού κλειδιού) και δεν είναι σχεδιασμένοι για την κρυπτογράφηση μεγάλων όγκων δεδομένων. Οι αλγόριθμοι δημοσίου κλειδιού είναι χρήσιμοι μόνο για τη μεταφορά πολύ μικρών όγκων δεδομένων. Στην πράξη, η κρυπτογράφηση δημοσίου κλειδιού χρησιμοποιείται για την κρυπτογράφηση ενός κλειδιού και ενός διανύσματος αρχικοποίησης που θα χρησιμοποιηθούν από έναν αλγόριθμο ιδιωτικού κλειδιού. Αφού το κλειδί και το διάνυσμα αρχικοποίησης μεταδοθούν, η κρυπτογράφηση ιδιωτικού κλειδιού χρησιμοποιείται για το υπόλοιπο της συνεδρίας.

Το .NET Framework παρέχει τις ακόλουθες κλάσεις για την υλοποίηση των αλγορίθμων δημοσίου κλειδιού:

- DSACryptoServiceProvider
- RSACryptoServiceProvider

Ψηφιακές Υπογραφές: Οι αλγόριθμοι δημοσίου κλειδιού μπορούν επίσης να χρησιμοποιηθούν για να σχηματίσουν ψηφιακές υπογραφές. Οι ψηφιακές υπογραφές αυθεντικοποιούν την ταυτότητα ενός αποστολέα (από τη στιγμή που θεωρείται έμπιστο το ληφθέν δημόσιο κλειδί) και βοηθούν στην διατήρηση της ακεραιότητας των δεδομένων. Χρησιμοποιώντας ένα δημόσιο κλειδί που έχει παραχθεί από την Alice, ο Bob μπορεί να πιστοποιήσει ότι όντως η Alice έστειλε αυτά τα δεδομένα συγκρίνοντας την ψηφιακή υπογραφή με τα δεδομένα που έλαβε και το δημόσιο κλειδί της Alice.

Για να χρησιμοποιηθεί η κρυπτογράφηση δημοσίου κλειδιού για την ψηφιακή υπογραφή ενός μηνύματος, η Alice αρχικά εφαρμόζει έναν αλγόριθμο hash στο μήνυμα, έτσι ώστε να πραγματοποιηθεί κωδικοποίηση του μηνύματος. Η έξοδος του hash αλγόριθμου αποτελεί μία συμπαγή και μοναδική αναπαράσταση των δεδομένων. Η Alice στη συνέχεια κρυπτογραφεί το μήνυμα χρησιμοποιώντας το δικό της ιδιωτικό κλειδί έτσι ώστε να δημιουργήσει την προσωπική της υπογραφή. Όταν λάβει το κρυπτογραφημένο μήνυμα και υπογραφή, ο Bob αποκρυπτογραφεί την υπογραφή χρησιμοποιώντας το δημόσιο κλειδί της Alice έτσι ώστε να ανακτήσει την κωδικοποίηση του μηνύματος και εφαρμόζει τον ίδιο hash αλγόριθμο που χρησιμοποίησε και η Alice στο ληφθέν μήνυμα. Εάν η έξοδος του αλγόριθμου είναι ακριβώς ίδια με την κωδικοποίηση του μηνύματος που έλαβε από την Alice, ο Bob είναι σίγουρος ότι το μήνυμα ήρθε από τον κάτοχο του ιδιωτικού κλειδιού και ότι τα δεδομένα δεν τροποποιήθηκαν. Εάν ο Bob είναι σίγουρος ότι η Alice και μόνο είναι κάτοχος του ιδιωτικού κλειδιού, τότε γνωρίζει ότι το μήνυμα προέρχεται από την Alice.

Να σημειωθεί ότι η υπογραφή μπορεί να επικυρωθεί από οποιονδήποτε επειδή το δημόσιο κλειδί του αποστολέα μπορεί να είναι ευρέως γνωστό και στην πράξη περιλαμβάνεται στη μορφή της ψηφιακής υπογραφής. Η μέθοδος αυτή δεν παρέχει προστασία για το ίδιο το μήνυμα. Για να προστατευθεί το ίδιο το μήνυμα, πρέπει και αυτό να κρυπτογραφηθεί.

Το .NET Framework παρέχει τις ακόλουθες κλάσεις για την υλοποίηση αλγόριθμων ψηφιακής υπογραφής:

- DSACryptoServiceProvider
- RSACryptoServiceProvider

Τιμές Hash: Οι αλγόριθμοι hash αντιστοιχίζουν δυαδικές τιμές τυχαίου μεγέθους σε μικρές δυαδικές τιμές σταθερού μεγέθους, γνωστές ως τιμές hash. Μία τιμή hash αποτελεί μοναδική και εξαιρετικά συμπαγή αριθμητική αναπαράσταση ενός κομματιού δεδομένων. Εάν εφαρμοστεί hash σε μια παράγραφο καθαρού κειμένου και στη συνέχεια τροποποιηθεί έστω και ένα μόνο γράμμα της παραγράφου, η ακόλουθη εφαρμογή hash θα παράγει μια διαφορετική τιμή. Είναι υπολογιστικά αδύνατο να ευρεθούν δύο διαφορετικές είσοδοι που να επιστρέφουν την ίδια τιμή όταν εφαρμοστεί hash σε αυτές.

Δύο οντότητες (έστω Alice και Bob) θα μπορούσαν να χρησιμοποιήσουν μια hash μέθοδο με τον ακόλουθο τρόπο για να εξασφαλίσουν την ακεραιότητα των δεδομένων. Εάν η Alice γράψει ένα μήνυμα για τον Bob και δημιουργήσει ένα hash αυτού του μηνύματος, ο Bob μπορεί να εφαρμόσει hash στο μήνυμα αυτό και να συγκρίνει το δικό του hash με το αρχικό hash. Αν οι δύο τιμές hash ταυτίζονται, τότε το μήνυμα δεν έχει τροποποιηθεί. Αν όμως οι τιμές δεν είναι ίδιες, το μήνυμα έχει τροποποιηθεί. Για να λειτουργήσει σωστά αυτή η μέθοδος, η Alice πρέπει να διατηρήσει κρυφή την αρχική hash τιμή από όλους εκτός από τον Bob.

Το .NET Framework παρέχει τις ακόλουθες κλάσεις που υλοποιούν αλγορίθμους hash:

- HMACSHA1
- MACTripleDES
- MD5CryptoServiceProvider
- SHA1Managed
- SHA256Managed
- SHA384Managed
- SHA512Managed

Παραγωγή τυχαίων αριθμών: Η παραγωγή τυχαίων αριθμών είναι ενσωματωμένη σε πολλές κρυπτογραφικές λειτουργίες. Για παράδειγμα, τα κρυπτογραφικά κλειδιά πρέπει να είναι όσο περισσότερο τυχαία γίνεται, έτσι ώστε να μην είναι εφικτή η αναπαραγωγή τους. Οι παραγωγείς κρυπτογραφικών τυχαίων αριθμών πρέπει να δημιουργούν αριθμούς που είναι υπολογιστικά ανέφικτο να αναπαραχθούν με μία πιθανότητα μεγαλύτερη από 0,05, που σημαίνει ότι οποιαδήποτε μέθοδος που προβλέπει το επόμενο bit δεν πρέπει να λειτουργεί καλύτερα από την τυχαία πρόβλεψη. Οι κλάσεις στο .NET Framework χρησιμοποιούν παραγωγείς τυχαίων αριθμών για να δημιουργήσουν κρυπτογραφικά κλειδιά.
Η RNGCryptoServiceProvider κλάση αποτελεί μια υλοποίηση ενός αλγορίθμου παραγωγής τυχαίων αριθμών.

Κληρονομικότητα αντικειμένων: Το σύστημα ασφάλειας του .NET Framework υλοποιεί ένα επεκτάσιμο σχήμα κληρονομικότητας. Η ιεραρχία έχει ως εξής:

- Κλάση που περιγράφει τον τύπο του αλγορίθμου, π.χ. SymmetricAlgorithm ή HashAlgorithm. Το επίπεδο αυτό είναι αφηρημένο (abstract).

- Κλάση που κληρονομεί από μια κλάση τύπου αλγόριθμου, για παράδειγμα RC2 ή SHA1. Το επίπεδο αυτό είναι αφηρημένο (abstract)
- Υλοποίηση ενός αλγορίθμου με μία κλάση που κληρονομεί από μια κλάση αλγορίθμου, για παράδειγμα RC2CryptoServiceProvider ή SHA1Managed. Το επίπεδο αυτό είναι πλήρως υλοποιημένο. Χρησιμοποιώντας αυτή τη δομή των κληρονομούμενων κλάσεων, είναι εύκολο για κάποιον προγραμματιστή να προσθέσει το δικό του αλγόριθμο ή μια νέα υλοποίηση ενός υπάρχοντος αλγορίθμου. Για παράδειγμα, για δημιουργία μιας νέα υλοποίησης ενός συγκεκριμένου αλγόριθμου, θα πρέπει να δημιουργηθεί μία νέα μη αφηρημένη παραγόμενη κλάση για αυτό τον αλγόριθμο.

Υλοποίηση για streams: Το CLR χρησιμοποιεί έναν σχεδιασμό που προσανατολίζεται στα streams για την υλοποίηση συμμετρικών και hash αλγορίθμων. Ο πυρήνας αυτού του σχεδιασμού είναι η CryptoStream κλάση που κληρονομεί από την κλάση Stream. Τα κρυπτογραφικά αντικείμενα που βασίζονται σε streams υποστηρίζουν όλα ένα μοναδικό interface (CryptoStream) για την επεξεργασία του κομματιού της μεταφοράς δεδομένων του αντικειμένου. Επειδή όλα τα αντικείμενα είναι βασισμένα σε ένα συγκεκριμένο interface, είναι δυνατό να αλυσιδωθούν πολλά αντικείμενα (όπως ένα hash αντικείμενο ακολουθούμενο από ένα αντικείμενο κρυπτογράφησης), και να πραγματοποιηθούν πολλαπλές λειτουργίες στα δεδομένα χωρίς να χρειάζεται ενδιάμεση αποθήκευση για αυτά. Το μοντέλο streaming επίσης επιτρέπει τη δημιουργία αντικειμένων από μικρότερα αντικείμενα. Για παράδειγμα, μια συνδυασμένη κρυπτογράφηση και ένας hash αλγόριθμος μπορούν να θεωρηθούν ως ένα μοναδικό stream αντικείμενο ακόμη και αν αυτό το αντικείμενο έχει δημιουργηθεί από ένα σύνολο stream αντικειμένων.

4.2.5 Ασφάλεια βασιζόμενη σε ρόλους

Οι ρόλοι εφαρμόζονται συχνά στον οικονομικό ή επιχειρησιακό τομέα για την επιβολή της επιθυμητής πολιτικής. Για παράδειγμα, μία εφαρμογή του μηχανισμού των ρόλων στον τραπεζικό τομέα, είναι η επιβολή περιορισμών στο ύψος των συναλλαγών που επιτρέπονται βάσει του αν ο χρήστης που διεκπεραιώνει τη συναλλαγή είναι μέλος ενός συγκεκριμένου ρόλου. Για παράδειγμα οι υπάλληλοι μπορούν να έχουν εξουσιοδότηση να διεκπεραιώνουν συναλλαγές που είναι μικρότερες από ένα συγκεκριμένο κατώφλι, οι διευθυντές μπορεί να έχουν ένα υψηλότερο όριο, ενώ οι αντιπρόεδροι μπορεί να έχουν ένα ακόμη υψηλότερο όριο (ή καθόλου όριο). Η ασφάλεια βασιζόμενη στους ρόλους μπορεί επίσης να χρησιμοποιηθεί σε περιπτώσεις που χρειάζονται πολλαπλές εγκρίσεις για να πραγματοποιηθεί μια πράξη. Τέτοια περίπτωση μπορεί να αποτελεί ένα σύστημα αγοράς στο οποίο ένας εργαζόμενος έχει τη δυνατότητα να πραγματοποιήσει μία αίτηση αγοράς, αλλά μόνο ο διευθυντής αγορών μπορεί να μετατρέψει αυτή την αίτηση σε παραγγελία που να μπορεί να σταλεί στον προμηθευτή.

Η βασιζόμενη σε ρόλους ασφάλεια του .NET Framework υποστηρίζει εξουσιοδότηση συλλέγοντας πληροφορίες για μία οντότητα οι οποίες κατασκευάζονται από μια ανατεθειμένη ταυτότητα η οποία είναι διαθέσιμη στο τρέχον thread. Αυτή η ταυτότητα (και η οντότητα η οποία ορίζει) μπορεί να βασίζεται σε ένα Windows λογαριασμό ή να είναι μια ξεχωριστή ταυτότητα που δε σχετίζεται με Windows λογαριασμό. Οι εφαρμογές του .NET Framework πραγματοποιούν τις αποφάσεις εξουσιοδότησης βασιζόμενες στην ταυτότητα της οντότητας ή στο αν ανήκει ή όχι σε ένα ρόλο. Ρόλος είναι ένα ονομασμένο σύνολο οντοτήτων που έχουν τα ίδια δικαιώματα ασφάλειας. Μία οντότητα μπορεί να ανήκει σε έναν ή περισσότερους ρόλους. Επομένως, οι εφαρμογές μπορούν να χρησιμοποιήσουν την τεχνική των ρόλων για να αποφασίσουν εάν μια οντότητα είναι εξουσιοδοτημένη να πραγματοποιήσει μια λειτουργία.

Με σκοπό την ευκολία χρήσης και τη συνέπεια με την ασφάλεια πρόσβασης κώδικα, η βασιζόμενη σε ρόλους ασφάλεια του .NET Framework παρέχει τα PrincipalPermission αντικείμενα που επιτρέπουν στο CLR να πραγματοποιεί τη διαδικασία της εξουσιοδότησης με έναν τρόπο παρόμοιο με τον οποίο ελέγχει η ασφάλεια κώδικα πρόσβασης. Η κλάση PrincipalPermission αντιπροσωπεύει την ταυτότητα ή το ρόλο με τον οποίο πρέπει να ταιριάζει η οντότητα και είναι συμβατή τόσο με δηλωμένους όσο και με αυτόματους ελέγχους ασφάλειας. Επίσης οι πληροφορίες ταυτότητας μίας οντότητας είναι απευθείας διαθέσιμες, ενώ μπορούν να γίνονται έλεγχοι ρόλων και ταυτοτήτων οποτεδήποτε χρειάζονται.

4.3 *Πρακτικές μεγιστοποίησης ασφάλειας*

Καθώς το .NET Framework είναι ένα προγραμματιστικό εργαλείο, η ασφάλεια των παραγόμενων μέσω αυτού εφαρμογών εξαρτάται σε μεγάλο βαθμό από τον ίδιο τον προγραμματιστή. Παρακάτω, παρουσιάζονται συγκεκριμένες τεχνικές και στρατηγικές οι οποίες έχουν ως αποτέλεσμα ασφαλέστερες εφαρμογές.

4.3.1 Προστασία δεδομένων

Οι εφαρμογές που χειρίζονται ευαίσθητα δεδομένα ή παίρνουν κάθε είδους αποφάσεις ασφάλειας πρέπει να κρατούν τα δεδομένα υπό το δικό τους έλεγχο και δεν πρέπει να επιτρέπουν σε άλλο ενδεχόμενα κακόβουλο κώδικα να έχει απευθείας πρόσβαση σε αυτά τα δεδομένα. Ο καλύτερος τρόπος να κρατηθούν αυτά τα δεδομένα προστατευμένα είναι ως private ή internal (πρόσβαση μόνο από την ίδια assembly) μεταβλητές στις αντίστοιχες κλάσεις. Παρ' όλα αυτά ακόμη και αυτού του τύπου οι μεταβλητές είναι προσβάσιμες στις εξής περιπτώσεις

- Όταν χρησιμοποιείται reflection[25], ένας κώδικας με ισχυρά δικαιώματα που έχει αναφορά σε ένα αντικείμενο μπορεί να διαβάσει και να γράψει private και internal μέλη του αντικειμένου.
- Όταν μια κλάση υποστηρίζει serialization, ένας κώδικας με ισχυρά permissions μπορεί να διαβάσει και να γράψει private μέλη στην περίπτωση που έχει πρόσβαση στα αντίστοιχα δεδομένα στη serialized μορφή του αντικειμένου
- Κατά το debugging, ακόμη και οι private μεταβλητές μπορούν να διαβαστούν

Σε άλλες περιπτώσεις, οι μεταβλητές μπορούν να προστατευθούν ως protected με την πρόσβαση να περιορίζεται στην ίδια την κλάση και τις παραγόμενες από αυτή κλάσεις. Παρ' όλα αυτά πρέπει να λαμβάνονται οι παρακάτω επιπρόσθετες προφυλάξεις:

- Έλεγχος του ποιος κώδικας επιτρέπεται να κληρονομεί τη συγκεκριμένη κλάση περιορίζοντάς τον στην ίδια assembly ή χρησιμοποιώντας την ασφάλεια πρόσβασης κώδικα έτσι ώστε να απαιτείται κάποια ταυτότητα ή permission για να κληρονομήσει κάποιος από τη συγκεκριμένη κλάση
- Εξασφάλιση ότι όλες οι παραγόμενες κλάσεις υλοποιούν την ίδια προστασία ή είναι sealed[26].

4.3.2 Προστασία πρόσβασης σε μεθόδους (Δηλωτική ασφάλεια)

Κάποιες μέθοδοι μπορεί να μην είναι κατάλληλες ώστε να κληθούν από μη έμπιστο κώδικα. Τέτοιες μέθοδοι εκθέτουν πολλούς κινδύνους: η μέθοδος μπορεί να εκθέτει προς τα έξω κάποιες

[25] Διαδικασία κατά την οποία μια εφαρμογή μπορεί να επισκοπήσει και να τροποποιήσει την ίδια της τη δομή και συμπεριφορά σε χρόνο εκτέλεσης.

[26] Sealed Class – Κλάση που δεν επιτρέπει σε κανένα να κληρονομήσει τον εαυτό της

εμπιστευτικές πληροφορίες, μπορεί να μην πραγματοποιεί έλεγχο λαθών ή παραπλανητικών πληροφοριών στις παραμέτρους της ή με προσεκτικά επιλεγμένες παραμέτρους μπορεί να λειτουργήσει με μη αναμενόμενο τρόπο και να κάνει κάτι επιζήμιο.

Στις περισσότερες περιπτώσεις, πρέπει να περιορίζονται οι μέθοδοι που δεν προορίζονται για δημόσια χρήση, αλλά πρέπει να είναι public. Για παράδειγμα, μπορεί να υπάρχει ένα interface που χρειάζεται να κληθεί από DLLs της ίδιας εφαρμογής, και για αυτό το λόγο πρέπει να είναι public, αλλά δεν είναι επιθυμητό να εκτεθεί δημόσια για να προστατέψει τους πελάτες από το να το χρησιμοποιήσουν ή να αποτρέψει το γεγονός κακόβουλος κώδικας να εκμεταλλευθεί αυτό το σημείο εισόδου για το στοιχείο αυτό. Ένας ακόμη συνήθης λόγος για τον περιορισμό μιας μεθόδου που δεν προορίζεται για δημόσια χρήση (παρ' όλα αυτά πρέπει να είναι public) είναι για να μη χρειαστεί να τεκμηριωθεί και να υποστηριχθεί. Η χρήση των ακόλουθων δηλώσεων βοηθάει στην προστασία κλάσεων και μεθόδων (συμπεριλαμβανομένων properties και events) από το να χρησιμοποιηθούν από μερικώς έμπιστο κώδικα. Εφαρμόζοντας αυτούς τους περιορισμούς σε μία κλάση, η προστασία θα εφαρμοστεί σε όλες τις μεθόδους, τα properties και τα events. Παρ' όλα αυτά να σημειωθεί ότι η πρόσβαση στα πεδία (μεταβλητές) μιας κλάσης δεν επηρεάζεται από αυτού του τύπου τη δηλωτική ασφάλεια.

- Για τις strong-named assemblies, η δηλωτική ασφάλεια εφαρμόζεται αυτόματα σε όλες τις δημόσια προσβάσιμες μεθόδους, properties και events έτσι ώστε να περιορίσει τη χρήση τους μόνο στους πλήρως έμπιστους καλούντες, εκτός και αν η assembly έχει επιλέξει να λειτουργεί διαφορετικά,

χρησιμοποιώντας το attribute AllowPartiallyTrustedCallers[27]. Επομένως, η προστασία αυτών των κλάσεων ώστε να αποκλείουν τους μη έμπιστους καλούντες χρειάζεται μόνο για τις unsigned assemblies ή assemblies με αυτό το attribute στην τιμή true.

- Για public non-sealed κλάσεις:

```
[System.Security.Permissions.PermissionSetAttribute(System.Security.
  Permissions.SecurityAction.InheritanceDemand, Name="FullTrust")]
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.LinkDemand,
Name="FullTrust")]
public class CanDeriveFromMe
```

- Για public sealed κλάσεις:

```
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.LinkDemand,
Name="FullTrust")]
public sealed class CannotDeriveFromMe
```

- Για public abstract κλάσεις:

```
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.InheritanceDemand,
Name="FullTrust")]
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.LinkDemand,
Name="FullTrust")]
public abstract class CannotCreateInstanceOfMe_CanCastToMe
```

- Για public virtual μεθόδους:

```
class Base {
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.InheritanceDemand,
  Name="FullTrust")]
[System.Security.Permissions.PermissionSetAttribute(
  System.Security.Permissions.SecurityAction.LinkDemand,
  Name="FullTrust")]
public override void CanOverrideOrCallMe() { ... }
```

- Για public abstract μεθόδους:

```
class Base {
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.InheritanceDemand,
Name="FullTrust")]
```

[27] Επιτρέπει σε strong-named assemblies να κληθούν από μερικώς έμπιστο κώδικα. Χωρίς αυτή τη δήλωση μόνο πλήρως έμπιστοι καλούντες μπορούν να χρησιμοποιήσουν αυτές τις assemblies.

```
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.LinkDemand,
  Name="FullTrust")]
public override void CanOverrideMe() { ... }
```

- Για public override μεθόδους όπου αρχική κλάση δεν απαιτεί πλήρη εμπιστοσύνη:

```
class Derived {
[System.Security.Permissions.PermissionSetAttribute
(System.Security.Permissions.SecurityAction.Demand,
Name="FullTrust")]
public override void CanOverrideOrCallMe() { ... }
```

- Για public override μεθόδους όπου η αρχική κλάση απαιτεί πλήρη εμπιστοσύνη:

```
class Derived {
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.LinkDemand,
  Name="FullTrust")]
public override void CanOverrideOrCallMe() { ... }
```

Για public interfaces:

```
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.InheritanceDemand,
  Name="FullTrust")]
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.LinkDemand,
  Name="FullTrust")]
public interface CanCastToMe
```

- Για public interfaces:

```
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.InheritanceDemand,
  Name="FullTrust")]
[System.Security.Permissions.PermissionSetAttribute
  (System.Security.Permissions.SecurityAction.LinkDemand,
  Name="FullTrust")]
public interface CanCastToMe
```

4.3.3 Interfaces και LinkDemands

Εάν μια virtual μέθοδος, property ή event με LinkDemand κάνει override μια base class μέθοδο, η base class μέθοδος πρέπει να έχει το ίδιο LinkDemand για την overridden μέθοδο έτσι ώστε να είναι ασφαλής. Ένας ενδεχόμενα κακόβουλος κώδικας θα μπορούσε να

πραγματοποιήσει cast στο base type και να καλέσει τη μέθοδο της base class. Επίσης, να σημειωθεί ότι το LinkDemand μπορεί να προστεθεί σε assemblies που δεν έχουν το attribute AllowPartiallyTrustedCallers attribute στην τιμή true.
Αποτελεί καλή πρακτική να προστατεύονται οι υλοποιήσεις των μεθόδων με LinkDemands όταν οι interface μέθοδοι έχουν επίσης LinkDemands.
Να σημειωθούν τα παρακάτω όσον αφορά τη χρήση LinkDemands με interfaces:

- Το AllowPartiallyTrustedCallers attribute μπορεί να επηρεάσει interfaces.
- Τα LinkDemands μπορούν να τοποθετηθούν σε interfaces έτσι ώστε να αποκλειστούν συγκεκριμένα interfaces από μερικώς έμπιστη χρήση κώδικα, όπως γίνεται με τη χρήση του attribute AllowPartiallyTrustedCallers.
- Εάν έχει οριστεί ένα interface σε μία assembly που δεν περιέχει το attribute AllowPartiallyTrustedCallers, δίνεται με τον τρόπο αυτό η δυνατότητα να υλοποιηθεί το interface αυτό σε μία μερικώς έμπιστη κλάση.
- Εάν προστεθεί ένα LinkDemand σε μία public μέθοδο μιας κλάσης που υλοποιεί μία interface μέθοδο, το LinkDemand δε θα εφαρμοστεί εάν γίνει cast στο interface και στη συνέχεια κληθεί η μέθοδος. Σε αυτή την περίπτωση, επειδή η «σύνδεση» γίνεται με το interface, εφαρμόζεται μόνο το LinkDemand του interface.

4.3.4 Virtual internal override μέθοδοι

Υπάρχει ένα λεπτό σημείο του συστήματος προσβασιμότητας στο οποίο πρέπει να δοθεί προσοχή όταν πρέπει να παραχθεί κώδικας που δε θα είναι διαθέσιμος σε άλλες assemblies. Μία μέθοδος που δηλώνεται ως virtual και internal μπορεί να κάνει override μέρος της υπερκλάσης και μπορεί να χρησιμοποιηθεί μόνο από την ίδια

assembly, αφού δηλώνεται ως internal. Παρ' όλα αυτά η δυνατότητα για overriding καθορίζεται από τη λέξη-κλειδί virtual κάτι που σημαίνει ότι η μέθοδος μπορεί να γίνει overridden από μια άλλη assembly με την προϋπόθεση ότι ο κώδικας στον οποίο γίνεται το override έχει πρόσβαση στην ίδια την κλάση. Εάν η πιθανότητα ενός override ενέχει ένα πρόβλημα ασφάλειας, προτείνεται η δηλωτική ασφάλεια ή η αφαίρεση της λέξης-κλειδί virtual όταν αυτό δεν είναι αυτό απαραίτητο.

4.3.5 Wrapper κώδικας[28]

Ο wrapper κώδικας ειδικά στην περίπτωση που το wrapper έχει υψηλότερη εμπιστοσύνη από τον κώδικα που το χρησιμοποιεί μπορεί να προσθέσει ένα σύνολο κινδύνων ασφάλειας. Εάν τα δικαιώματα του καλούντος δεν συμπεριληφθούν στον κατάλληλο έλεγχο ασφάλειας δημιουργούνται δυνατότητες εκμετάλλευσης αυτών των κινδύνων ασφάλειας.

Ποτέ δεν πρέπει να δίνεται η δυνατότητα στο wrapper κώδικα να κάνει κάτι το οποίο δεν μπορεί να το κάνει από μόνος του ο καλών. Κι αυτό γιατί εισάγει έναν ιδιαίτερο κίνδυνο εάν χρησιμοποιείται ο απλός έλεγχος ασφάλειας (LinkDemand) (και όχι η πλήρης διάτρεξη του stack εκτέλεσης - Demand). Όταν γίνεται έλεγχος ενός μόνο επιπέδου του stack εκτέλεσης (LinkDemand), η ένθεση του wrapper κώδικα μεταξύ του πραγματικού καλούντος και του API στοιχείου μπορεί εύκολα να προκαλέσει επιτυχία στον έλεγχο ασφάλειας τη στιγμή που έπρεπε να αποτύχει.

---

[28] Wrapper κώδικας – Κώδικας που παίζει το ρόλο ενός επιπέδου μεταξύ δύο προγραμματιστικών interfaces το οποίο επιτρέπει τη μεταξύ τους επικοινωνία η οποία θα ήταν αδύνατη χωρις αυτόν.

4.3.6 Unmanaged κώδικας[29]

Είναι πολλές οι φορές που χρειάζεται κάποια κλήση σε unmanaged κώδικα (για παράδειγμα native code APIs, όπως το Win32). Επειδή αυτό σημαίνει ότι απαιτείται να γίνει υπέρβαση των ορίων ασφάλειας του managed κώδικα, χρειάζεται επιπλέον προσοχή. Εάν ο κώδικας είναι ουδέτερος όσον αφορά την ασφάλεια, τόσο ο κώδικας που κάνει την κλήση στον unmanaged κώδικα όσο και ο κώδικας που τον καλεί πρέπει να έχουν το permission για unmanaged κώδικα (SecurityPermission.UnmanagedCode).

Παρ' όλα αυτά, τις περισσότερες φορές δεν είναι λογικό να απαιτούνται από τον καλούντα τόσο ισχυρά permissions. Σε αυτές τις περιπτώσεις ο έμπιστος κώδικας μπορεί να αποτελέσει το μέσο μεταξύ του καλούντα και του unmanaged κώδικα όπως συμβαίνει και με το wrapper κώδικα. Εάν η λειτουργία του unmanaged κώδικα που καλείται από τον έμπιστο κώδικα δεν ενέχει κινδύνους ασφάλειας, μπορεί να εκτεθεί απευθείας. Στην αντίθετη περίπτωση, χρειάζεται πρώτα ένας κατάλληλος έλεγχος ασφάλειας (Demand).

Όταν υπάρχει κώδικας που καλεί unmanaged κώδικα αλλά δεν είναι επιθυμητό οι καλούντες να έχουν αυτή τη δυνατότητα πρέπει ο κώδικας να διεκδικεί το δικαίωμα αυτό. Αυτό σημαίνει ότι πρέπει να απαιτείται ένα κατάλληλο permission από τους καλούντες και στη συνέχεια να χρησιμοποιείται ο unmanaged κώδικας για την εκτέλεση αυτού που επιτρέπει το permission και τίποτα παραπάνω. Σε μερικές περιπτώσεις (για παράδειγμα, στην επιστροφή της ώρας της μέρας), ο unmanaged κώδικας μπορεί να εκτεθεί απευθείας στους καλούντες χωρίς κανένα έλεγχο ασφαλείας.

Εξαιτίας του ότι οποιοσδήποτε managed κώδικας που περιέχει μια «διαδρομή» προς τον unmanaged κώδικα είναι ένας ενδεχόμενος στόχος επιθέσεων, η απόφαση ποιος unmanaged κώδικας μπορεί να

[29] Unmanaged κώδικας – Οποιοσδήποτε κώδικας τρέχει έξω από το πλαίσιο του CLR

χρησιμοποιηθεί και πώς, απαιτεί επιπλέον προσοχή. Γενικά, δεν πρέπει να εκτίθεται απευθείας unmanaged κώδικας σε μερικώς έμπιστους καλούντες. Υπάρχουν δύο βασικοί παράγοντες για την αξιολόγηση της χρήσης unmanaged κώδικα σε βιβλιοθήκες που μπορούν να κληθούν από μερικώς έμπιστο κώδικα.

- Λειτουργικότητα: Κρίση του αν το unmanaged API παρέχει ασφαλή λειτουργικότητα που δεν περιλαμβάνει ενδεχόμενα επικίνδυνες λειτουργίες να εκτελεστούν κατά την κλήση του. Η ασφάλεια πρόσβασης κώδικα χρησιμοποιεί permissions για τον έλεγχο της πρόσβασης σε πόρους, οπότε πρέπει ο προγραμματιστής να αναλογιστεί εάν το API χρησιμοποιεί αρχεία, user interface, threading ή εκθέτει προστατευμένες πληροφορίες. Αν συμβαίνει κάτι από αυτά το wrapping στο managed κώδικα πρέπει να απαιτεί τα απαραίτητα permissions πριν να επιτρέψει την κλήση του unmanaged κώδικα.
- Έλεγχος παραμέτρων: Μία συνήθης επίθεση περνά μη αναμενόμενες τιμές για τις παραμέτρους στις εκτεθειμένες API μεθόδους του unmanaged κώδικα σε μια προσπάθεια να τις αναγκάσουν να λειτουργήσουν εκτός προδιαγραφών. Οι υπερχειλίσεις buffers είναι ένα συνηθισμένο παράδειγμα αυτού του τύπου των επιθέσεων. Με τον τρόπο αυτό ακόμα και αν το API του unmanaged κώδικα είναι λειτουργικά ασφαλές για μερικώς έμπιστους καλούντες (μετά από τα απαραίτητα demands) ο managed κώδικας πρέπει να ελέγχει εξαντλητικά την εγκυρότητα των παραμέτρων έτσι ώστε να εξασφαλίσει ότι δεν είναι δυνατές μη προβλεπόμενες κλήσεις από κακόβουλο κώδικα χρησιμοποιώντας το επίπεδο του wrapper του managed κώδικα.

4.3.7 Σύμβαση Ονοματολογίας για μεθόδους unmanaged κώδικα

Μία χρήσιμη και ταυτόχρονα προτεινόμενη σύμβαση έχει καθιερωθεί για την ονοματολογία των μεθόδων που πραγματοποιούν κλήσεις σε

unmanaged κώδικα. Όλες οι μέθοδοι unmanaged κώδικα χωρίζονται σε τρεις κατηγορίες: safe, native και unsafe. Αυτές οι λέξεις κλειδιά μπορούν να χρησιμοποιηθούν ως ονόματα κλάσεων στις οποίες ορίζονται τα σημεία εισόδου για τον unmanaged κώδικα. Στον κώδικα αυτές οι λέξεις κλειδιά μπορούν να προστεθούν στο όνομα κλάσης: για παράδειγμα, Safe.GetTimeOfDay ή Native.Xyz ή Unsafe.DangerousAPI. Κάθε μία από αυτές τις κατηγορίες δίνει ένα ισχυρό μήνυμα στους προγραμματιστές να τις χρησιμοποιούν όπως περιγράφεται στον παρακάτω πίνακα.

Λέξη κλειδί	Θεωρήσεις ασφάλειας
safe	Εντελώς ακίνδυνος κώδικας για οποιοδήποτε κώδικα (ακόμη και κακόβουλο). Μπορεί να χρησιμοποιηθεί όπως ο υπόλοιπος managed κώδικας. Παράδειγμα: get time of day
native	Ουδέτερη ασφάλεια, που σημαίνει unmanaged κώδικας που απαιτεί permission unmanaged κώδικα για να κληθεί. Η ασφάλεια ελέγχεται, οπότε ο μη εξουσιοδοτημένος καλών αποτρέπεται
unsafe	Ενδεχόμενα επικίνδυνο σημείο κλήσης unmanaged κώδικα. Οι προγραμματιστές πρέπει να έχουν τη μεγαλύτερη προσοχή όταν χρησιμοποιούν unsafe κώδικα, εξασφαλίζοντας ότι έχουν ληφθεί όλες οι προφυλάξεις για να αποτραπεί μια αδυναμία ασφάλειας.

4.3.8 Έλεγχος δεδομένων εισόδου από χρήστες

Τα δεδομένα εισόδου των χρηστών που μπορεί να είναι οποιασδήποτε μορφής (δεδομένα από ένα web request ή URL,

είσοδος σε πεδία μιας φόρμας κλπ) μπορούν να επηρεάσουν αρνητικά τον κώδικα καθώς συχνά αυτά τα δεδομένα χρησιμοποιούνται απευθείας ως παράμετροι για κλήση άλλου κώδικα. Η είσοδος από τους χρήστες δυσκολεύει την προστασία ασφάλειας επειδή δεν υπάρχει stack για να ανιχνευθεί η παρουσία ενδεχόμενα μη έμπιστων δεδομένων.

Τα bugs που σχετίζονται με την είσοδο των χρηστών είναι συνήθως τα δυσκολότερα ανιχνεύσιμα bugs ασφάλειας καθώς ακόμη και αν βρίσκονται σε κώδικα που με την πρώτη ματιά δε σχετίζεται με την ασφάλεια, αποτελούν μία «πύλη» για το πέρασμα κακόβουλα σχεδιασμένων δεδομένων σε άλλο κώδικα. Για την ανίχνευση αυτών των λαθών, πρέπει να γίνει αντιληπτό όλο το εύρος των πιθανών τιμών μιας εισόδου από το χρήστή και να εξεταστεί αν ο κώδικας που δέχεται την είσοδο του χρήστη μπορεί να χειριστεί όλες αυτές τις περιπτώσεις.

Κάποιες στρατηγικές για την αποφυγή παραλείψεων όσον αφορά το χειρισμό των δεδομένων εισόδου των χρηστών είναι οι εξής:

- Όλα τα δεδομένα από το χρήστη στην απάντηση του server τρέχουν στο περιβάλλον ασφάλειας του server. Εάν μία web εφαρμογή παίρνει π.χ. την είσοδο του χρήστη και τα περιλαμβάνει στην επιστρεφόμενη σελίδα, θα μπορούσε για παράδειγμα ο χρήστης να περιλάβει ένα <script> tag στην επιστρεφόμενη σελίδα με όλες τις συνεπαγόμενες συνέπειες. Επομένως, η απευθείας συμπερίληψη των δεδομένων των χρηστών στην επιστρεφόμενη σελίδα θα πρέπει να αποφεύγεται
- Αξιολόγηση όλων των δυνατών μορφών που μπορεί να έχουν τα url requests. Αυτές ενδεικτικά μπορεί να περιλαμβάνουν:
    - ..\, υπερβολικά μεγάλα paths.
    - Χρήση wildcards (*)
    - Επέκταση token (%token%)
    - Παράξενοι σχηματισμοί paths με ειδική σημασία

    - Εναλλακτικά NTFS ονόματα streams, για παράδειγμα, filename::$DATA.
    - Εναλλακτικές μορφές ονομάτων αρχείων, για παράδειγμα longfilename και longfi~1.
- Εάν πρόκειται για δεδομένα που μεταδίδονται μέσω web, πρέπει να ληφθούν υπόψη οι πολλές μορφές των escape characters που επιτρέπονται, συμπεριλαμβανομένων:
    - Hex escapes (%nn)
    - Unicode escapes (%nnn)
    - Overlong UTF-8 escapes (%nn%nn)
    - Διπλά escapes (%nn γίνεται %mmnn, όπου %mm είναι το escape του '%').
- Να λαμβάνεται υπόψη ότι τα user names μπορεί να έχουν περισσότερες από μία κανονικές μορφές. Π.χ. στα Microsoft Windows 2000, μπορεί να χρησιμοποιηθεί είτε η μορφή REDMON\username είτε η μορφή *username*@redmond.microsoft.com.

4.3.9 Κίνδυνοι από το .NET Remoting

Το .NET Remoting επιτρέπει «διαφανείς» κλήσεις μεταξύ application domains, διεργασιών ή συστημάτων. Παρ' όλα αυτά η διάτρεξη του stack εκτέλεσης από την ασφάλεια κώδικα πρόσβασης δεν μπορεί να περάσει από τα όρια των διεργασιών ή των συστημάτων (εφαρμόζεται παρ' όλα αυτά μεταξύ application domains της ίδιας διεργασίας).

Κάθε κλάση που είναι remotable (κληρονομεί δηλαδή από μία MarshalByRefObject κλάση) πρέπει να λαμβάνει ειδικά μέτρα για την ασφάλειας. Είτε ο κώδικας πρέπει να χρησιμοποιείται μόνο σε κλειστά περιβάλλοντα ασφάλειας όπου ο καλών κώδικας μπορεί να θεωρηθεί έμπιστος είτε οι remoting κλήσεις πρέπει να σχεδιαστούν έτσι ώστε να μην εκθέτουν εξωτερικά προστατευμένο κώδικα που μπορεί να χρησιμοποιηθεί κακοβούλως.

Επίσης δεν πρέπει ποτέ να εκτίθενται μέθοδοι, properties ή events που προστατεύονται από τη δηλωτική LinkDemand και InheritanceDemand ασφάλεια. Με το remoting, αυτοί οι έλεγχοι δεν εφαρμόζονται. Άλλοι έλεγχοι ασφάλειας όπως Demand, Assert κλπ λειτουργούν μεταξύ διαφορετικών application domains μιας διεργασίας αλλά όχι μεταξύ διαφορετικών διεργασιών ή μηχανών.

4.3.10 Serialization

Το serialization ενός αντικειμένου μπορεί να επιτρέψει σε άλλο κώδικα να δει ή να τροποποιήσει τα δεδομένα του αντικειμένου που σε διαφορετική περίπτωση θα ήταν μη προσβάσιμα. Για το λόγο αυτό χρειάζεται ένα ειδικό permission για τον κώδικα που χρησιμοποιεί serialization, το SecurityPermission.SerializationFormatter. Στην προκαθορισμένη λειτουργία, αυτό το permission δε δίνεται σε κώδικα κατεβασμένο από το Internet ή από Intranet. Μόνο κώδικας από το τοπικό σύστημα λαμβάνει αυτό το permission.

Κανονικά όλα τα πεδία ενός στιγμιότυπου ενός αντικειμένου γίνονται serialize, που σημαίνει ότι αυτά τα δεδομένα θα αναπαρίστανται στα serialized δεδομένα του στιγμιότυπου του αντικειμένου. Επομένως, είναι δυνατό κάποιος κώδικας να αναλύσει το format των serialized δεδομένων και να αποφασίσει ποιες είναι οι τιμές των δεδομένων, ανεξάρτητα από την προσβασιμότητα των μελών. Ομοίως, κατά την deserialize αποσπώνται δεδομένα από τη serialized αναπαράσταση και το state του αντικειμένου τίθεται κατευθείαν χωρίς να λαμβάνονται υπόψη οι κανόνες προσβασιμότητας.

Κάθε αντικείμενο που θα μπορούσε να περιέχει ευαίσθητα όσον αφορά την ασφάλεια δεδομένα θα πρέπει να γίνεται non-serializable εάν είναι δυνατόν. Εάν το αντικείμενο πρέπει να είναι serializable, πρέπει ορισμένα πεδία που περιέχουν ευαίσθητα δεδομένα να μην είναι serializable. Αν αυτό δεν μπορεί να γίνει, πρέπει να εξασφαλιστεί ότι αυτά τα δεδομένα δεν πρέπει να εκτεθούν σε

κώδικα που έχει permission να κάνει serialize και ότι δεν είναι δυνατό κακόβουλος κώδικας να πάρει αυτό το permission.

Το interface ISerializable προορίζεται για χρήση μόνο από τη serialization δομή. Παρ' όλα αυτά αν δεν ληφθούν μέτρα προστασίας, μπορεί ενδεχόμενα να εκθέσει ευαίσθητα δεδομένα. Εάν παρέχεται custom serialization με την υλοποίηση του Iserializable, πρέπει να εξασφαλιστεί ότι έχουν ληφθεί οι ακόλουθες προφυλάξεις:

- Η μέθοδος GetObjectData πρέπει να έχει ασφαλιστεί αποκλειστικά είτε απαιτώντας το SecurityPermission.SerializationFormatter permission είτε εξασφαλίζοντας το ότι δεν εκτίθενται ευαίσθητα δεδομένα με την έξοδο της μεθόδου. Για παράδειγμα:

```
[SecurityPermissionAttribute(SecurityAction.Demand,SerializationForma
tter
=true)]
public override void GetObjectData(SerializationInfo info,
StreamingContext context)
```

- Ο ειδικός constructor που χρησιμοποιείται για το serialization πρέπει επίσης να κάνει εκτενή έλεγχο του input και πρέπει να είναι είτε protected είτε private για να αποφευχθεί χρήση από κακόβουλο κώδικα. Πρέπει να εφαρμόζει τους ίδιους ελέγχους ασφάλειας και permissions που απαιτούνται για να αποκτηθεί ένα στιγμιότυπο αυτής της κλάσης με οποιονδήποτε άλλο τρόπο (άμεση δημιουργία ή έμμεση δημιουργία μέσω κάποιου είδους factory).

4.3.11 Επιβολή permissions

Μία από τις αρχές της ασφάλειας πρόσβασης κώδικα είναι ότι μόνο ο έμπιστος κώδικας λαμβάνει υψηλή εμπιστοσύνη (πολλά ισχυρά permissions) και ο λιγότερο έμπιστος κώδικας λαμβάνει μόνο περιορισμένη ή και καθόλου εμπιστοσύνη. Η προκαθορισμένη τακτική για το .NET Framework χρησιμοποιεί ζώνες για την ανάθεση permissions. Ακολουθεί μια απλουστευμένη περιγραφή της προκαθορισμένης τακτικής ασφάλειας:

- Ζώνη Local Machine (για παράδειγμα, c:\app.exe): Λαμβάνει πλήρη εμπιστοσύνη. Η υπόθεση είναι ότι οι χρήστες πρέπει να τοποθετούν στο μηχάνημά τους μόνο κώδικα που εμπιστεύονται και ότι οι περισσότεροι χρήστες δεν επιθυμούν να χωρίσουν το σκληρό τους δίσκο σε διαφορετικές περιοχές εμπιστοσύνης. Ο κώδικας αυτός έχει τα δικαιώματα να κάνει οτιδήποτε και η ασφάλεια managed κώδικα δεν εφαρμόζεται, οπότε δεν υπάρχει τρόπος άμυνας εάν κακόβουλος κώδικας βρεθεί σε αυτή τη ζώνη.
- Ζώνη Internet (για παράδειγμα, http://www.microsoft.com/): Ο κώδικας αυτής της ζώνης λαμβάνει ένα πολύ περιορισμένο σύνολο από permissions που γενικά θεωρείται ασφαλές να ανατίθενται ακόμη και σε κακόβουλο κώδικα. Γενικά, ο κώδικας αυτής της ζώνης δεν μπορεί να θεωρηθεί έμπιστος όπότε η ασφάλεια managed κώδικα δεν είναι εφαρμόσιμη, οπότε μπορεί να εκτελεστεί ασφαλώς μόνο με ένα μικρό σύνολο από permissions που δεν είναι δυνατό να προκαλέσουν κινδύνους όπως:
    - WebPermission. Πρόσβαση στο ίδιο server site από το οποίο προέρχεται
    - FileDialogPermission. Πρόσβαση μόνο σε αρχεία που διαλέγει συγκεκριμένα ο χρήστης
    - IsolatedStorageFilePermission. Μόνιμη αποθήκευση, απομονωμένη από το web site
    - UIPermission. Μπορεί να γράψει σε ένα ασφαλές UI παράθυρο
- Ζώνη Intranet (για παράδειγμα, \\UNC\share): Ο κώδικας λαμβάνει ένα ελαφρά ισχυρότερο σύνολο από Internet permissions αλλά και πάλι μη ισχυρά permissions:
    - FileIOPermission. Πρόσβαση ανάγνωσης μόνο σε αρχεία του καταλόγου από το οποίο προέρχεται.

- WebPermission. Πρόσβαση στο server από τον οποίο προέρχεται
- DNSPermission. Επιτρέπει τα DNS ονόματα να αναλύονται σε IP διευθύνσεις
- FileDialogPermission. Πρόσβαση μόνο σε αρχεία που διαλέγει συγκεκριμένα ο χρήστης
- IsolatedStorageFilePermission. Μόνιμη αποθήκευση, απομονωμένη από το web site
- UIPermission. Μπορεί να χρησιμοποιήσει ελεύθερα τα δικά του παράθυρα

- Ζώνη Restricted: Λαμβάνει μόνο τα ελάχιστα δικαιώματα για να εκτελεστεί.

Παρ' όλα αυτά η προκαθορισμένη αυτή ρύθμιση ασφάλειας είναι αδύνατο να καλύψει όλες τις διαφορετικές ανάγκες που μπορούν να παρουσιαστούν στις διάφορες περιπτώσεις και χρειάζεται αξιολόγηση των permissions που χρειάζεται ο κώδικας καθώς και το από πού μπορεί να προέρχεται ο επιτιθέμενος κώδικας.

4.3.12 Επικίνδυνα Permissions

Σε πολλές από τις λειτουργίες του .NET Framework που μπορούν να προστατευθούν από permissions υπάρχει η δυνατότητα να δοθούν permissions που να επιτρέψουν την παράκαμψη του συστήματος ασφάλειας. Αυτά τα permissions πρέπει να δίνονται μόνο σε έμπιστο κώδικα και ακόμη και στην περίπτωση αυτή μόνο όταν είναι απαραίτητο. Στις περισσότερες περιπτώσεις δεν υπάρχει άμυνα απέναντι σε κακόβουλο κώδικα αν έχει λάβει κάποιο από αυτά τα permissions.

Τα επικίνδυνα permissions περιλαμβάνουν:

- SecurityPermission
  - UnmanagedCode: Επιτρέπει σε managed κώδικα να καλέσει unmanaged κώδικα κάτι που συχνά είναι επικίνδυνο.

- SkipVerification: Χωρίς verification ο κώδικας με αυτό το permission μπορεί να κάνει οτιδήποτε
- ControlEvidence: Η παραγωγή ενός evidence επιτρέπει την «εξαπάτηση» του συστήματος ασφάλειας.
- ControlPolicy: Η δυνατότητα τροποποίησης της τακτικής ασφάλειας μπορεί να απενεργοποιήσει τον έλεγχο ασφάλειας
- SerializationFormatter: Η χρήση serialization μπορεί να παρακάμψει τον έλεγχο προσβασιμότητας, όπως αναφέρθηκε σε προηγούμενο κεφάλαιο.
- ControPrincipal: Η δυνατότητα να τίθεται το τρέχον principal μπορεί να «εξαπατήσει» την ασφάλεια βασιζόμενη σε ρόλους.
- ControlThread: Η δυνατότητα χειρισμού threads είναι επικίνδυνη εξαιτίας του security state που σχετίζεται με τα threads.

- ReflectionPermission
  - MemberAccess. Εξουδετερώνει τους μηχανισμούς προσβασιμότητας.

4.4 *Αδυναμίες ασφάλειας*

4.4.1 Ανεπιθύμητο client-side scripting

Η συγκεκριμένη αδυναμία ασφάλειας είναι μία cross-site scripting αδυναμία που επιτρέπει σε κάποιον επιτιθέμενο να ενθέσει client side κώδικα στο browser του χρήστη. Ο κώδικας αυτός θα μπορούσε να παραποιήσει το περιεχόμενο μιας web σελίδας, να εκθέσει ευαίσθητες πληροφορίες ή να πραγματοποιήσει οποιαδήποτε ενέργεια θα μπορούσε να πραγματοποιήσει και ο χρήστης στο web site. Η προσπάθεια εκμετάλλευσης αυτής της αδυναμίας ασφάλειας απαιτεί τη συμμετοχή του χρήστη. «Υπεύθυνα» για αυτή την αδυναμία ασφάλειας είναι τα ASP.NET είναι τα server controls του

ASP.NET που έχουν ρυθμιστεί με το property AutoPostBack στο "true".

Αντισταθμιστικός παράγοντας σε αυτή την αδυναμία ασφάλειας είναι το γεγονός ότι by default τα server controls έχουν το property AutoPostBack στο "false".

Η αδυναμία ασφάλειας αυτή αφορά την έκδοση 2.0 του .NET Framework και διορθώνεται με την εγκατάσταση μιας ενημέρωσης ασφάλειας, η οποία δόθηκε στη δημοσιότητα στις 10/10/2006.

4.4.2 Έκθεση ευαίσθητων πληροφοριών

Αυτή η αδυναμία ασφάλειας θα μπορούσε να επιτρέψει σε έναν επιτιθέμενο να παρακάμψει την ASP.NET ασφάλεια και να αποκτήσει μη εξουσιοδοτημένη πρόσβαση στους καταλόγους μιας εφαρμογής χρησιμοποιώντας το όνομα των καταλόγων αυτών. Παρ' όλα αυτά η αδυναμία αυτή δεν επιτρέπει σε έναν επιτιθέμενο να εκτελέσει κώδικα της επιλογής του ή να κλιμακώσει άμεσα τα δικαιώματά του, αλλά θα μπορούσε να εκθέσει πολύτιμες πληροφορίες που μπορεί να ήταν χρήσιμες στον επιτιθέμενο έτσι ώστε να αποκτήσει μεγαλύτερο έλεγχο του επηρεαζόμενου συστήματος.

Αντισταθμιστικοί παράγοντες σε αυτή την αδυναμία ασφάλειας είναι οι εξής:

Το browsing στους καταλόγους δεν είναι ενεργοποιημένο by default. Ένας επιτιθέμενος θα έπρεπε να μαντέψει ή να ξέρει τα ονόματα των αρχείων που θα προσπαθούσε να ανακτήσει ή να δει.

Οι καταλήξεις των αρχείων που χρησιμοποιούνται από τις ASP.NET εφαρμογές αντιστοιχίζονται στον aspnet_isapi.dll System.Web.HttpForbiddenHandler και ως αποτέλεσμα τέτοιου είδους αρχεία δεν μπορούν να ανακτηθούν ή να ανοιχθούν απομακρυσμένα με χρήση αυτής της αδυναμίας ασφάλειας. Η πλήρης λίστα των καταλήξεων των αρχείων που προστατεύονται και επομένως δεν είναι ευάλωτα σε αυτή την αδυναμία ασφάλειας είναι η εξής: *.asax, *.ascx, *.master, *.skin, *.browser, *.sitemap,

*.config (but not *.exe.config or *.dll.config), *.cs, *.csproj, *.vb, *.vbproj, *.webinfo, *.licx, *.resx, *.resources, *.mdb, *.vjsproj, *.java, *.dd, *.jsl, *.ldb, *.ad, *.ldd, *.sd, *.cd, *.adprototype, *.lddprototype, *.sdm, *.sdmDocument, *.mdf, *.ldf, *.exclude, *.refresh.

Η αδυναμία ασφάλειας αυτή αφορά την έκδοση 2.0 του .NET Framework και διορθώνεται με την εγκατάσταση μιας ενημέρωσης ασφάλειας, η οποία δόθηκε στη δημοσιότητα στις 11/7/2006.

5. Πηγές

[1]. Microsoft Development Network (MSDN) (http://www.msdn.com)

[2]. Microsoft TechNet (http://technet.microsoft.com)

[3]. Wikipedia (http://www.wikipedia.org)

[4]. Sql Server Magazine (http://www.sqlmag.com)

[5]. CERT (http://www.cert.org)

[6]. CVE (Common Vulnerabilities and Exposures) (http://www.cve.mitre.org)

[7]. "SQL Server Security", Chip Andrews, David Litchfield, Bill Grindlay, εκδ. McGraw-Hill Professional, 2003

[8]. "Beginning SQL Server 2005 Administration", Dan Wood, Chris Leiter, Paul Turley, εκδ. John Wiley and Sons, 2006

[9]. "Professional SQL Server 2005 Administation", Brian Knight, Ketan Patel, Snyder, εκδ. Wiley-India, 2006

[10]. "Which database is most secure? Oracle vs. Microsoft", David Litchfield, 2006

[11]. MSDN Webcast: Security Best Practices : Hardening Your SQL Server (http://whitepapers.techrepublic.com.com/abstract.aspx?docid=154918)

[12]. SqlSecurity (http://www.sqlsecurity.com)

[13]. Sql Server Central (http://www.sqlservercentral.com/articles/Security/updatedsqlinjection/2065/)

[14]. MsSqlTips (http:// www.mssqltips.com)

[15]. WindowSecurity (http://www.windowsecurity.com/articles/Secure_SQL_Server.html)

[16]. Sql Server Performance (http://www.sql-server-performance.com/articles/dba/sql_security_p1.aspx)
[17]. NetworkWorld (http://www.networkworld.com/community/node/42290)
[18]. Cgi Security (http://www.cgisecurity.com/mssql-security.html)
[19]. Search System Channel (http://searchsystemschannel.techtarget.com/tip/0,289483,sid99_gci1305471,00.html)
[20]. http://vyaskn.tripod.com/sql_server_security_best_practices.htm
[21]. SecurityFocus (http://www.securityfocus.com/infocus/1765)
[22]. IIS Training (http://www.iistraining.com)
[23]. TheRegister (http://www.theregister.co.uk/2002/04/11/eight_new_iis_security_holes/)
[24]. SoOperTutorials (http://www.soopertutorials.com/tutorial-tags/iis-security-threats)
[25]. "Professional IIS 7", Ken Schaefer, Jeff Cochran, Scott Forsyth, Rob Baugh, Mike Everest, Dennis Glendenning, εκδ. John Wiley and Sons, 2008
[26]. "IIS Security", Marty Jost, Michael Cobb, εκδ. McGraw-Hill Professional, 2002
[27]. "CYA securing IIS 6.0", Chun Hai Cheah, Ken Schaefer, Chris Peiris, εκδ. Syngress, 2004
[28]. "Mastering Windows Server 2003", Mark Minasi, Christa Anderson, Michele Beveridge, C. A. Callahan, Lisa Justice, εκδ. SYBEX, 2003
[29]. ServerWatch (www.serverwatch.com/news/article.php/1400491)

[30]. Symantec (http://www.symantec.com/security_response/vulnerability.jsp?bid=6068)

[31]. IIS Workstation (http://www.iisworkstation.com/2008/06/iis-6-authentication-vs-authorization.html)

[32]. Informit (http://www.informit.com/articles/article.aspx?p=101750&seqNum=4)

[33]. Host01 (http://www.host01.com/article/Net/00020007/0561316360564690.htm)

[34]. Developer.Com (http://www.developer.com/security/article.php/3483866)

[35]. DevHood (http:// www.devhood.com/training_modules/dist-c/Security.NET/?module_id=5)

[36]. Embarcadero Developer Network (http://conferences.embarcadero.com/article/32273)

[37]. KnowDotNet (http:// www.knowdotnet.com/articles/declarativesecurity.html)

[38]. AppSecInc (Application Security Inc) (http://www.appsecinc.com/products/appdetective/mssql)

[39]. AspAlliance (http://aspalliance.com/1506_Understanding_Code_Access_Security_in_NET.2)

[40]. CodeProject (http://www.codeproject.com/KB/security/UB_CAS_NET.aspx)

[41]. "Secure Coding Practices For Microsoft .NET Applications", White Paper, Amit Klein, Directory of Security and Research, Sanctum Inc.

[42]. ".NET Framework Security", Brian A. LaMacchia, Sebastian Lange, Matthew Lyons, Rudi Martin, Kevin T. Price, εκδ. Addison-Wesley, 2002

[43]. "Microsoft .NET Framework Security", Surbhi Malhotra, Roopendra Sandhu, NIIT (Corporation), εκδ. Premier Press, 2002

[44]. "Hacking the code: ASP.NET web application security", Mark Burnett, James C. Foster εκδ. Syngress, 2004

[45]. Database Journal (http://www.databasejournal.com/features/mssql/article.php/3481751/SQL-Server-2005-Security---Part-2-Authorization.htm)