ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη κατανεμημένου συστήματος ανίχνευσης επιθέσεων σε δίκτυα υπολογιστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΓΙΩΡΓΟΥ Ο. ΑΝΔΡΟΥΛΙΔΑΚΗ

Επιβλέπων : Βασίλειος Μάγκλαρης
Καθηγητής ΕΜΠ

Αθήνα, Ιούνιος 2003

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη κατανεμημένου συστήματος ανίχνευσης επιθέσεων σε δίκτυα υπολογιστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΓΙΩΡΓΟΥ Ο. ΑΝΔΡΟΥΛΙΔΑΚΗ

Επιβλέπων : Βασίλειος Μάγκλαρης
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τον Ιούνιο 2003.

…….....	……..	…….
Βασίλειος Μάγκλαρης	Ευστάθιος Συκάς	Ανδρέας Σταφυλοπάτης
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ

Αθήνα, Ιούνιος 2003

……

ΓΕΩΡΓΙΟΣ Ο. ΑΝΔΡΟΥΛΙΔΑΚΗΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών ΕΜΠ

Περίληψη

Οι επιθέσεις σε δίκτυα υπολογιστών στις μέρες μας έχουν γίνει πολύ συχνές. Από τις πρώτες εμφανίσεις τους, συστήματα που ανιχνεύουν τέτοιες επιθέσεις, γνωστά ως Συστήματα Ανίχνευσης Επιθέσεων (IDS) έχουν αναπτυχθεί για να επιλύσουν το πρόβλημα αυτό. Σε μικρά δίκτυα ένα μεμονωμένο IDS είναι επαρκές για να ανιχνεύσει αυτές τις επιθέσεις. Αντίθετα, σε μεγάλα δίκτυα, όπου ο αριθμός των πακέτων που ρέουν μέσα στο δίκτυο είναι τεράστιος, ένα μεμονωμένο IDS δεν είναι αρκετό. Σε αυτές τις περιπτώσεις, ένα κατανεμημένο σύστημα ανίχνευσης επιθέσεων με πολλαπλούς ανιχνευτές είναι η λύση στην ανίχνευση τέτοιων επιθέσεων.

Ο σκοπός αυτής της διπλωματικής εργασίας ήταν η ανάπτυξη ενός κατανεμημένου συστήματος ανίχνευσης επιθέσεων. Το σύστημα που κατασκευάστηκε αποτελείται από πολλούς ανιχνευτές διασκορπισμένους στο δίκτυο και κεντρικούς κόμβους που συλλέγουν τις πληροφορίες που στέλνουν οι ανιχνευτές. Ο κάθε ανιχνευτής είναι υπεύθυνος για να επιβλέπει ένα συγκεκριμένο τμήμα του δικτύου. Μόλις ανιχνευτεί μία επίθεση, ο ανιχνευτής ειδοποιεί τους κεντρικούς κόμβους στέλνοντας ένα μήνυμα που περιγράφει την επίθεση. Το πρωτόκολλο IDMEF επιλέχθηκε να είναι αυτό που θα περιγράφει τις πληροφορίες της επίθεσης.

Για να προστατευθεί ο κεντρικός κόμβος από την λήψη ψεύτικων μηνυμάτων επίθεσης, έγινε επέκταση του πρωτοκόλλου IDMEF ώστε να υποστηρίζει ψηφιακές υπογραφές. Για να μειώσουμε την κίνηση στο δίκτυο αλλά και συγχρόνως να λαμβάνουν τις πληροφορίες επίθεσης περισσότεροι από ένας κεντρικοί κόμβοι χρησιμοποιήσαμε την τεχνολογία IP Multicast για την μεταφορά των μηνυμάτων.

Για την υλοποίηση του κατανεμημένου συστήματος ανίχνευσης επιθέσεων, επελέγη το Snort IDS ως το λογισμικό του ανιχνευτή. Πιο συγκεκριμένα, κατασκευάστηκε ένα καινούριο Snort output plugin το οποίο έχει τη δυνατότητα να χρησιμοποιεί το πρωτόκολλο IDMEF για να στέλνει μηνύματα από τον ανιχνευτή στους κεντρικούς κόμβους.

Abstract

Attacks on computer networks have become very common nowadays. Since their first appearance, systems that detect such attacks, known as Intrusion Detection Systems have been developed to solve the problem. In small-scale networks a single IDS is sufficient to detect attacks and probably respond to them. On the other hand, in large-scale networks, where the number of packets across the network is enormous, a single IDS is not enough. In such cases, a Distributed Intrusion Detection System with multiple sensors is the solution for detecting these attacks.

The scope of this thesis was the development of a Distributed Intrusion Detection System. The system that was developed comprises a number of IDS sensors scattered around the network and central IDS nodes that collect the information sent by the sensors. Each IDS sensor is responsible for monitoring a certain part of the network; once an attack is detected, the sensor must notify the central IDS nodes with a message that describes the attack. The Intrusion Detection Message Exchange Format (IDMEF) protocol was chosen to represent the attack information.

To protect the central IDS nodes from receiving fake IDMEF messages, the IDMEF protocol was extended to support digital signatures. In order to reduce network traffic and allow attack information to be delivered to more than one central IDS node, we used IP Multicast for the transportation of the messages.

For the implementation of the Distributed Intrusion Detection System, Snort IDS was chosen as the IDS sensor's software. Specifically, a new Snort output plugin was created which is capable of using the IDMEF protocol to send alerts from the sensor to the central IDS nodes.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κ. Βασίλη Μάγκλαρη για τη δυνατότητα που μου έδωσε να ασχοληθώ με την ασφάλεια δικτύων και να εκπονήσω στον τομέα αυτό τη διπλωματική μου εργασία. Στη συνέχεια θα ήθελα να ευχαριστήσω τους υποψήφιους διδάκτορες Βασίλη Χατζηγιαννάκη και Γιώργο Κουτέπα για την ουσιαστική συμβολή τους στην αποπεράτωση της εργασίας αυτής. Τέλος, θα ήθελα να ευχαριστήσω και τα υπόλοιπα μέλη του εργαστηρίου NETMODE για την υποστήριξη και τη βοήθειά τους. Η "οικογενειακή ατμόσφαιρα" που επικρατεί είναι κάτι που δύσκολα θα βρει κανείς σε άλλο εργαστήριο.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1. Εισαγωγή – Επιθέσεις σε δίκτυα υπολογιστών

Η ανάπτυξη των δικτύων υπολογιστών προήλθε από την ανάγκη των διαφόρων εταιριών και οργανισμών να καταστούν διαθέσιμα όλα τα προγράμματα και τα δεδομένα που κρατούνταν μέχρι τότε σε μεμονωμένους υπολογιστές, σε οποιονδήποτε στο δίκτυο, ανεξάρτητα από τη φυσική θέση του πόρου (resource) και του χρήστη.

Στον τομέα των επιχειρήσεων, τα δίκτυα υπολογιστών προσφέρουν ένα ισχυρό επικοινωνιακό μέσο για εργαζόμενους που είναι μεταξύ τους πολύ απομακρυσμένοι. Με τη χρήση ενός δικτύου, είναι εύκολο για δύο ή περισσότερους ανθρώπους, που βρίσκονται σε μεγάλη απόσταση μεταξύ τους να γράψουν μία αναφορά μαζί. Όταν ένας εργαζόμενος πραγματοποιεί μία αλλαγή σε ένα on-line κείμενο οι άλλοι μπορούν να δουν αμέσως την αλλαγή αυτή αντί να περιμένουν αρκετές μέρες για μία επιστολή.

Στις αρχές της δεκαετίας του ΄90, με την εξάπλωση του Διαδικτύου σε ολόκληρο τον κόσμο, τα δίκτυα υπολογιστών άρχισαν να παρέχουν υπηρεσίες σε ιδιώτες, στα σπίτια τους. Οι υπηρεσίες αυτές περιλαμβάνουν την πρόσβαση σε απομακρυσμένες πληροφορίες, την επικοινωνία πρόσωπο με πρόσωπο και τη διασκέδαση με αλληλεπίδραση. Η πρόσβαση σε απομακρυσμένες πληροφορίες είναι πολυμορφική. Μία περιοχή στην οποία ήδη συμβαίνει είναι η πρόσβαση σε τραπεζικούς οργανισμούς. Πολλοί άνθρωποι πληρώνουν τα χρέη τους, διαχειρίζονται τους τραπεζικούς τους λογαριασμούς και τις επενδύσεις τους ηλεκτρονικά. Οι αγορές από το σπίτι (home shopping) έχουν γίνει δημοφιλείς, με τη δυνατότητα να ανατρέχει κανείς σε on-line καταλόγους χιλιάδων εταιριών.

Μία άλλη μεγάλη καινοτομία στα δίκτυα υπολογιστών αποτελεί ο Παγκόσμιος Ιστός (World Wide Web) που περιέχει πληροφορίες για τις τέχνες, τις επιχειρήσεις, την κυβέρνηση, την υγεία, την ιστορία, τη διασκέδαση, τις επιστήμες, τα ταξίδια και ένα σωρό άλλα θέματα. Επίσης το ηλεκτρονικό ταχυδρομείο (email) χρησιμοποιείται ήδη από εκατομμύρια ανθρώπων και μεταφέρει εκτός του κειμένου, εικόνα και ήχο. Μία άλλη πολύ σημαντική καινοτομία αποτελεί η αποκαλούμενη τηλεδιάσκεψη (video-conference) μεταξύ διεσπαρμένων, σε μεγάλες γεωγραφικές περιοχές, ανθρώπων. Οι διασκέψεις αυτές χρησιμοποιούνται σήμερα για τη

λειτουργία σχολείων από απόσταση (e-learning), τη λήψη ιατρικών διαγνώσεων από απομακρυσμένους ειδικούς γιατρούς και πολυάριθμες άλλες εφαρμογές.

Στις πρώτες δεκαετίες της ύπαρξης τους τα δίκτυα υπολογιστών χρησιμοποιήθηκαν πρωταρχικά από πανεπιστημιακούς ερευνητές για την αποστολή ηλεκτρονικού ταχυδρομείου και από υπαλλήλους οργανισμών για την από κοινού χρήση εκτυπωτών. Υπό τις συνθήκες αυτές η ασφάλεια δεν συγκέντρωνε την προσοχή. Αλλά σήμερα, καθώς εκατομμύρια πολιτών χρησιμοποιούν τα δίκτυα για τις τραπεζικές συναλλαγές τους και τις αγορές τους, η ασφάλεια δικτύου εμφανίζεται απειλητικά στον ορίζοντα ως ένα ενδεχόμενο μαζικό πρόβλημα.

Τα περισσότερα προβλήματα ασφάλειας προκαλούνται εσκεμμένα από κακόβουλους ανθρώπους που προσπαθούν να αποσπάσουν κάποιο κέρδος ή να βλάψουν κάποιον. Χαρακτηριστικό παράδειγμα αποτελεί η βιομηχανική κατασκοπία, όπου μια εταιρία θα ήθελε να γνωρίζει τα μελλοντικά σχέδια των ανταγωνιστικών εταιριών και αν ήταν δυνατόν να αχρηστέψει τις υπηρεσίες που προσφέρουν μέσω του Διαδικτύου έτσι ώστε να τις βλάψει οικονομικά και να κερδίσει η ίδια μερίδιο από τους πελάτες αυτών των εταιριών.

Είναι ολοφάνερο ότι το να κάνει κάποιος ένα δίκτυο ασφαλές είναι πολύ πιο απαιτητικό από το να το προστατεύει από λάθη προγραμματισμού. Απαιτεί το να ξεγελάσει κανείς τους συχνά ευφυείς, αφοσιωμένους και μερικές φορές καλά χρηματοδοτημένους αντιπάλους. Είναι επίσης ολοφάνερο, ότι τα μέτρα που θα σταματήσουν τους περιστασιακούς αντιπάλους θα ασκήσουν λίγη επίδραση στους σοβαρούς.

Για το λόγο αυτό, επειδή η δημιουργία ενός ασφαλούς δικτύου φαντάζει ουτοπική, θα πρέπει να υπάρχει πρόληψη για την έγκαιρη ανίχνευση των επιθέσεων από κακόβουλους χρήστες του δικτύου. Συμπεραίνουμε λοιπόν πως το δίκτυο μας θα πρέπει να είναι εφοδιασμένο με ένα εργαλείο ανίχνευσης επιθέσεων. Το εργαλείο αυτό θα πρέπει να είναι σε θέση να ειδοποιεί τον διαχειριστή ασφάλειας του δικτύου όταν γίνεται κάποια επίθεση, προκειμένου αυτός να προσπαθήσει να την αντιμετωπίσει προστατεύοντας το στόχο της επίθεσης. Τέτοια εργαλεία υπάρχουν, ονομάζονται Συστήματα Ανίχνευσης Επιθέσεων και θα μιλήσουμε αναλυτικά για αυτά στα επόμενα κεφάλαια.

2. Συστήματα Ανίχνευσης Επιθέσεων (IDS)

2.1. Εισαγωγή στα Συστήματα Ανίχνευσης Επιθέσεων

Η Ανίχνευση Επιθέσεων (Intrusion Detection) αποτελεί μια προσέγγιση για την παροχή μιας αίσθησης ασφάλειας στους υπάρχοντες υπολογιστές και δίκτυα, επιτρέποντας παράλληλα σε αυτά να λειτουργούν με μη περιορισμένο τρόπο. Ο στόχος της ανίχνευσης επιθέσεων είναι να προσδιοριστεί, κατά προτίμηση σε πραγματικό χρόνο, η κακή χρήση και η κατάχρηση των συστημάτων ηλεκτρονικών υπολογιστών τόσο από τα ίδια τα εσωτερικά μέλη των συστημάτων όσο και από εξωτερικούς χρήστες. Το πρόβλημα αυτό γίνεται μια πρόκληση καθώς η αυξανόμενη δικτύωση των ηλεκτρονικών υπολογιστών δίνει μεγαλύτερη πρόσβαση στους εξωτερικούς χρήστες και διευκολύνει τους εισβολείς να αποφεύγουν την αναγνώρισή τους. Τα Συστήματα Ανίχνευσης Επιθέσεων (Intrusion Detection Systems) βασίζονται στο γεγονός ότι η συμπεριφορά του εισβολέα θα είναι διαφορετική από αυτήν κάποιου κανονικού χρήστη και στο ότι οι ανάρμοστες πράξεις είναι άμεσα ανιχνεύσιμες.

Η συμβατική προσέγγιση για την ασφάλεια ενός υπολογιστικού συστήματος ή δικτύου περιλαμβάνει τη δημιουργία μιας προστατευτικής ασπίδας γύρω από αυτό. Η ασπίδα αυτή θα πρέπει να αποτρέπει την ροή πληροφοριών από προστατευμένες περιοχές του δικτύου προς τον εξωτερικό κόσμο. Τεχνικές ελέγχου πρόσβασης μπορεί να χρησιμοποιηθούν για το σχεδιασμό τέτοιων ασφαλών συστημάτων. Ωστόσο υπάρχουν κάποιοι περιορισμοί στο σχεδιασμό ασφαλών υπολογιστικών συστημάτων και δικτύων. Ο βασικότερος από αυτούς αναφέρεται στο ότι είναι πολύ δύσκολο αν όχι ακατόρθωτο να σχεδιάσει κανείς ένα σύστημα που θα είναι εύχρηστο και συγχρόνως ασφαλές. Επίσης, δεν μπορεί κανείς να αποκλείσει από ένα θεωρητικά ασφαλές σύστημα κάποιο λάθος στο configuration από τον διαχειριστή το οποίο θα οδηγήσει σε πρόβλημα ασφάλειας.

Για το λόγο αυτό, στα μέσα της δεκαετίας του ‘80 μία εναλλακτική προσέγγιση που ονομαζόταν ανίχνευση εισβολής (intrusion detection) έκανε την εμφάνισή της. Η νέα αυτή προσέγγιση της ασφάλειας, δεν είχε σκοπό να αλλάξει την υπάρχουσα υποδομή των πιθανά ανασφαλών συστημάτων με καινούρια συστήματα που θα ήταν ασφαλή, γιατί κάτι τέτοιο δεν θα ήταν ούτε εφικτό αλλά ούτε και οικονομικό. Τα συστήματα ανίχνευσης επιθέσεων θα ήταν βασισμένα σε μια

τεχνολογία που θα επέτρεπε να ανιχνεύει επιθέσεις σε υπολογιστές και σε δίκτυα, κατά προτίμηση σε πραγματικό χρόνο, και να ειδοποιούν για αυτές το διαχειριστή ασφαλείας. Η προσέγγιση αυτή με το πέρασμα του χρόνου κέρδισε όλο και περισσότερο έδαφος στο χώρο της ασφάλειας με αποτέλεσμα ένας μεγάλος αριθμός από πρωτότυπα IDS να έχουν δημιουργηθεί σήμερα σε πολλά ερευνητικά κέντρα και μερικά από αυτά να έχουν εγκατασταθεί σε παραγωγικά συστήματα.

2.2. Ταξινόμηση Συστημάτων Ανίχνευσης Επιθέσεων

Υπάρχουν δύο κύριες ταξινομήσεις των συστημάτων ανίχνευσης επιθέσεων. Ο πρώτος διαιρεί τις τεχνικές ανίχνευσης επιθέσεων σε δύο κύριους τύπους: ανίχνευση ανωμαλίας (anomaly detection) και κακής χρήσης (misuse detection). Το πρότυπο ανίχνευσης ανωμαλίας χρησιμοποιεί ένα σύνολο στατιστικών στοιχείων που διαμορφώνουν τη συμπεριφορά μιας οντότητας. Οντότητα μπορεί να είναι ένας χρήστης, μια ομάδα χρηστών ή ένας υπολογιστής. Το προφίλ μιας οντότητας χρηστών, μπορεί να περιλάβει πληροφορίες όπως η μέση διάρκεια των συνόδων του Telnet και FTP, το ποσό των bytes που μεταδίδονται και προς τις δύο κατευθύνσεις, τις ώρες της ημέρας ή τα τερματικά από τα οποία συνδέεται ο χρήστης. Το προφίλ ενός υπολογιστή μπορεί να περιλαμβάνει τη μέση χρησιμοποίηση της CPU, το μέσο αριθμό συνδεδεμένων χρηστών, κ.α. Ένα IDS (Intrusion Detection System) ελέγχει τη λειτουργία ενός υπολογιστικού συστήματος και συγκρίνει συνεχώς το προφίλ μιας τρέχουσας συνόδου ενός χρήστη, με το προφίλ που είναι αποθηκευμένη στη βάση δεδομένων του. Σε περίπτωση που ανιχνεύσει μια μεγάλη απόκλιση από την κανονική συμπεριφορά στέλνει μία ειδοποίηση στο διαχειριστή ασφάλειας των υπολογιστικών συστημάτων. Το μέγεθος μιας μεγάλης απόκλισης ορίζεται ως ένα κατώτατο όριο που τίθεται από το IDS ή τον διαχειριστή ασφάλειας των συστημάτων. Συνήθως τα αποθηκευμένα προφίλ ενημερώνονται συνεχώς προκειμένου να απεικονιστούν οι αλλαγές στη συμπεριφορά των χρηστών ή του συστήματος. Δεδομένου ότι αυτό το πρότυπο λειτουργεί με βάση την ανίχνευση συνόδων που διαφέρουν σημαντικά από τις συνηθισμένες συνόδους ενός χρήστη, καλείται πρότυπο ανίχνευσης ανωμαλίας.

Το πρότυπο ανίχνευσης κακής χρήσης, λειτουργεί με βάση την ανίχνευση ενός συνόλου γνωστών επιθέσεων που έχουν αποθηκευτεί στη βάση δεδομένων του

συστήματος. Η γνώση των επιθέσεων κωδικοποιείται ως ένα σύνολο από “υπογραφές επιθέσεων”, οι οποίες είναι ουσιαστικά ακολουθίες από χαρακτήρες, που εμφανίζονται κάθε φορά που πραγματοποιείται μια επίθεση. Ο τρόπος που μια γνωστή επίθεση αντιπροσωπεύεται στο σύστημα είναι ένα σημαντικό χαρακτηριστικό της λειτουργίας του. Ο τρόπος που αυτό το πρότυπο λειτουργεί είναι παρόμοιος με αυτόν ενός anti-virus προγράμματος. Η εφαρμογή ενός τέτοιου IDS περιλαμβάνει συνήθως ένα έμπειρο σύστημα που εκτελεί τη σύγκριση με κανόνες αποθηκευμένους σε μια βάση δεδομένων. Μια προφανής δυσκολία σε αυτή την αρχιτεκτονική είναι η ανάγκη για τη σταθερή ενημέρωση της βάσης με καινούριες υπογραφές επιθέσεων, καθώς νέες μέθοδοι επιθέσεων γίνονται γνωστές. Δεδομένου ότι το πρότυπο αυτό λειτουργεί με την έρευνα για δείγματα που είναι αντιπροσωπευτικά διαφόρων επιθέσεων, αναφέρεται στη βιβλιογραφία ως πρότυπο ανίχνευσης κακής χρήσης.

Η δεύτερη ταξινόμηση είναι βασισμένη στο εάν το IDS ελέγχει τη δραστηριότητα σε ένα συγκεκριμένο υπολογιστή ή σε ένα δίκτυο υπολογιστών. Τα πρώτα συστήματα ανίχνευσης επιθέσεων συνήθιζαν να εξετάζουν στοιχεία σε ένα μεμονωμένο υπολογιστή και να παράγουν τα συμπεράσματά τους βασισμένα στα τοπικά αρχεία του συγκεκριμένου υπολογιστή. Συνεπώς, δεν θα μπορούσαν να ανιχνεύσουν τις επιθέσεις που στοχεύουν σε πολλούς υπολογιστές σε ένα δίκτυο. Επιπλέον, τα IDS αυτά στηρίζονται σε μεγάλο ποσοστό στα logs αρχεία που παρέχονται από το λειτουργικό σύστημα του υπολογιστή, το οποίο τα καθιστά αρχιτεκτονικά εξαρτώμενα και πιο ευάλωτα σε επιθέσεις DoS (Denial of Service) ενάντια σε αυτά, δεδομένου ότι ένας εισβολέας μπορεί να κατορθώσει να καθυστερήσει το μηχανισμό καταγραφής ή ακόμα και να τον σταματήσει τελείως. Τα συστήματα αυτά είναι γνωστά ως host-based IDS.

Μια πιο αποδοτική λύση για ανίχνευση των επιθέσεων παρέχεται από τα IDS που ελέγχουν παθητικά το δίκτυο, εξετάζοντας τα πακέτα που ρέουν σε αυτό, για ύποπτη δραστηριότητα. Δεδομένου ότι στηρίζονται στο πρωτόκολλο TCP/IP, είναι ανεξάρτητα από την αρχιτεκτονική του λειτουργικού συστήματος και μπορούν να ελέγξουν τα δίκτυα υπολογιστών αρκετά φυσικά. Αν συνυπολογίσουμε και τη σύγχρονη τάση προς τη σύνδεση των υπολογιστών μέσω δικτύων, σχεδόν κάθε επίθεση περιλαμβάνει χρήση του δικτύου. Επομένως αυτή η κατηγορία των IDS που είναι γνωστή ως network-based αποτελεί σήμερα μία αναγκαιότητα για την ασφάλεια του δικτύου.

3. Βασικά Πρωτόκολλα και Τεχνολογίες

3.1. Η γλώσσα XML

3.1.1. Περιγραφή

Η XML (eXtensible Markup Language) όπως και η HTML αποτελεί ένα υποσύνολο της SGML (Standard Generalized Markup Language) και χρησιμοποιείται για τη παρουσίαση δομημένων δεδομένων με μια μορφή κειμένου. Καταφέρνει να διατηρήσει όλη τη περιγραφική δύναμη της SGML αφαιρώντας όλη σχεδόν την πολυπλοκότητα όπως και η HTML αλλά δεν χρησιμοποιείται για να μεταφέρει τη πληροφορία παρουσίασης του κειμένου αλλά τη συντακτική δομή του.

Ένα κείμενο XML μπορεί προαιρετικά να έχει μια περιγραφή της γραμματικής που χρησιμοποιεί. Ο μηχανισμός περιγραφής της γραμματικής καλείται Document Type Definition (DTD). Το DTD περιγράφει τα στοιχεία (elements) που επιτρέπονται στο κείμενο καθώς και τη δομή αυτών των στοιχείων. Ένα κείμενο που ακολουθεί την περιγραφή του DTD καλείται καλά δομημένο (well formed) ενώ προαιρετικά μπορεί να καλείται και έγκυρο (valid). Ένα έγκυρο κείμενο πρέπει να περιέχει ένα DTD και να ακολουθεί τη γραμματική αυτού.

Τα XML κείμενα δεν περιέχουν πληροφορία ως προς το πώς τα δεδομένα που βρίσκονται στο κείμενο θα αναλυθούν και θα παρουσιαστούν. Αυτή τη γραφική ανάλυση μπορούμε να την επιτύχουμε χρησιμοποιώντας XSL (eXtensible Style Language) φύλλα. Γενικά αυτά τα φύλλα χρησιμοποιούνται για να μετατρέψουν ένα κείμενο XML σε κάποια άλλη δομή (format). Έτσι μπορούμε για παράδειγμα να μετατρέψουμε ένα κείμενο XML σε HTML ώστε να το δούμε μέσα από έναν browser ή σε κείμενο του Word και να μπορούμε να το προσθέσουμε σε κάποιο άλλο κείμενο. Το σημαντικό είναι ότι το ίδιο XML κείμενο μπορούμε με τη χρήση διαφορετικών φύλλων XSL να το μετατρέψουμε σε οποιαδήποτε δομή επιθυμούμε.

3.1.2. Πλεονεκτήματα της XML

Η XML είναι ιδιαίτερα χρήσιμη σε αρκετές περιπτώσεις. Ειδικά σε περιβάλλοντα όπου η εφαρμογή πελάτης χρησιμοποιεί web-based τεχνικές για την διαχείριση και εμφάνιση των κειμένων αποτελεί πολύ χρήσιμο εργαλείο.

- **Ανάλυση και μετασχηματισμός**.

Όπως προαναφέρθηκε, με τη χρήση των XSL φύλλων μπορούμε να μετασχηματίσουμε ένα κείμενο XML σε οποιαδήποτε άλλη αναπαράσταση. Παράλληλα μπορούμε να χρησιμοποιήσουμε περισσότερα από ένα τέτοια φύλλα στο ίδιο κείμενο αποκτώντας έτσι ένα μεγάλο πλήθος επιλογών. Για παράδειγμα μπορούμε να μετατρέψουμε μια πληροφορία διαχείρισης είτε σε ένα HTML κείμενο για να το δούμε σε ένα browser είτε σε ένα MOF (Managed Object Format).

Η χρήση των XSL φύλλων μπορεί να πραγματοποιηθεί τόσο στον πελάτη όσο και στον εξυπηρετητή. Το να γίνεται η ανάλυση της πληροφορίας στον πελάτη έχει το πλεονέκτημα ότι μπορούν να χρησιμοποιηθούν διαφορετικά στυλ μετασχηματισμού χωρίς να επιστρέφουμε συνέχεια στον εξυπηρετητή για νέες πληροφορίες. Έτσι για παράδειγμα θα μπορούσαμε να σχεδιάσουμε μια ιστοσελίδα που θα ρωτάει τον χρήστη ποιο στυλ ανάλυσης επιθυμεί και αφού γίνει η επιλογή τότε χωρίς να γίνεται νέα χρήση του εξυπηρετητή να εφαρμόζεται το XSL στυλ στο ήδη υπάρχον XML κείμενο.

Αυτή η ιδιότητα μπορεί να γίνει εκμεταλλεύσιμη από μια εφαρμογή διαχείρισης σε αρκετές περιπτώσεις. Διαφορετικά XSL φύλλα μπορούν να υλοποιηθούν ώστε να καλύπτουν κάθε περίπτωση και σενάριο. Δημιουργώντας XSL φύλλα μπορούμε την ίδια πληροφορία να την εκμεταλλευτούν διαχειριστές που χρησιμοποιούν MOF ή διαχειριστές που δουλεύουν με το Excel και θέλουν τα δεδομένα να είναι οργανωμένα σε κελιά. Επιπλέον μπορούμε να επιλέξουμε από πολλούς διαφορετικούς τρόπους αναπαράστασης των δεδομένων είτε σε μια ιστοσελίδα ή με τη μορφή ενός διαγράμματος.

Τέλος εφ’ όσον η XSL μπορεί να παράγει και διάφορα scripts είναι δυνατό να παράγουμε και εφαρμογές συνδυάζοντας την HTML με scripts. Έτσι θα μπορούσαμε αν είχαμε ένα κείμενο XML που περιγράφει έναν δίσκο να δημιουργήσουμε μια σελίδα με ένα button το οποίο όταν το πατούσαμε να φόρμαρε το δίσκο.

- **Ετερογενείς εφαρμογές.**

Η XML είναι ιδιαίτερα βολική όσον αφορά τη χρήση της σε ετερογενή περιβάλλοντα. Εφ' όσον αποτελεί ένα «ανοιχτό» προϊόν έχουν υλοποιηθεί parsers για όλες τις πλατφόρμες και προγραμματιστικές γλώσσες. Έτσι υπάρχουν υλοποιήσεις σε Unix και C++ όσο και σε Java. Επιπλέον η δημιουργία ενός parser αποτελεί μια σχετικά απλή διαδικασία αφού το μόνο που απαιτείται από το προγραμματιστικό περιβάλλον είναι να κατανοεί ASCII χαρακτήρες κάτι που μπορούν να κάνουν όλα σχεδόν τα συστήματα.

Το γεγονός αυτό τοποθετεί την XML στην κορυφή των προτιμήσεων όσον αφορά την μεταφορά δεδομένων σε ετερογενείς πλατφόρμες. Σε ένα σύστημα διαχείρισης θα μπορούσαμε να μεταφέρουμε δεδομένα σε διάφορες διαχειριστικές πλατφόρμες που συνεργάζονται μεταξύ τους. Αυτές οι πλατφόρμες δεν θα υποχρεούνται να χρησιμοποιούν π.χ. ίδιο λειτουργικό σύστημα αφού θα μπορούν όλες να καταλάβουν και να επεξεργαστούν τα XML κείμενα που λαμβάνουν.

- **Η XML ως πηγή διαχειριζόμενων πληροφοριών**

Η XML μπορεί να αποτελέσει και μια άριστη επιλογή για την παρουσίαση δεδομένων διαχείρισης σε συσκευές που δεν χρησιμοποιούν κάποιο άλλο τρόπο παραγωγής αυτών των δεδομένων. Το μόνο που χρειάζεται είναι να υλοποιηθεί ένα λεξικό (DTD) που περιγράφει τη συσκευή και με τη χρήση XSL φύλλων μπορούμε να εκμεταλλευτούμε το κείμενο που θα παραχθεί σε οποιαδήποτε εφαρμογή διαχείρισης.

- **Έλεγχος εγκυρότητας**

Τα κείμενα XML μπορούν προαιρετικά να ελεγχθούν για την εγκυρότητά τους. Αυτή η ενέργεια πραγματώνεται ελέγχοντας αν το κείμενο ακολουθεί την γραμματική ενός DTD. Το DTD είτε θα περιέχεται μέσα στο XML κείμενο ή θα παρέχεται μέσω ενός link. Η χρήση ενός απομακρυσμένου DTD έχει αρκετές ενδιαφέρουσες επιλογές για τα συστήματα διαχείρισης. Χρησιμοποιώντας αυτόν τον τρόπο ελέγχου εγκυρότητας μπορούμε να προσδιορίσουμε το κατά πόσο ένα κείμενο που περιέχει δεδομένα διαχείρισης ακολουθεί τα πρότυπα που έχει ορίσει ένα κεντρικό σύστημα ώστε να επιλέξουμε αν θα το απορρίψουμε ή όχι.

Σε ένα τέτοιο περιβάλλον κεντρικού ελέγχου η συσκευή που τίθεται υπό διαχείριση δεν είναι απαραίτητο να παρέχει το DTD αλλά μόνο ένα link σε αυτό. Αυτή η περίπτωση είναι αρκετά ωφέλιμη για συσκευές με περιορισμένες δυνατότητες. Ακόμη μας δίνει την δυνατότητα να ελέγξουμε το περιεχόμενο ενός κειμένου που προέρχεται από κάποιο server που είναι αναξιόπιστος.

3.2. Πρωτόκολλο IDMEF

Το πρωτόκολλο IDMEF (Intrusion Detection Message Exchange Format) αποτελεί τη "γλώσσα" την οποία χρησιμοποιούν τα συστήματα ανίχνευσης επιθέσεων για να επικοινωνήσουν μεταξύ τους και να ανταλλάξουν πληροφορίες που αφορούν δικτυακές επιθέσεις. Το πρωτόκολλο αυτό έχει αναπτυχθεί από το IETF (Internet Engineering Task Force) και αυτή τη στιγμή αποτελεί ένα Internet-Draft που αναμένεται σύντομα να γίνει RFC. Ο σκοπός δημιουργίας αυτού του πρωτοκόλλου είναι να καθοριστεί το "format" και να οριστούν επακριβώς οι διαδικασίες ανταλλαγής πληροφοριών που αφορούν περιστατικά ασφαλείας ανάμεσα σε Intrusion Detection Systems, σε συστήματα που αντιμετωπίζουν τέτοια περιστατικά καθώς και σε συστήματα διαχείρισης δικτύων που αλληλεπιδρούν με αυτά.

Η ανάπτυξη αυτού του "format" βοηθάει στη σωστή συνεργασία μεταξύ εμπορικών συστημάτων, συστημάτων ανοικτού κώδικα και ερευνητικών συστημάτων, τα οποία θα μπορέσουν να επικοινωνήσουν μεταξύ τους, ώστε να δώσουν στο χρήστη τη δυνατότητα να συνδυάσει τα συστήματα αυτά με αποτέλεσμα την καλύτερη υλοποίηση ενός συστήματος για την ανίχνευση επιθέσεων.

Το πιο προφανές μέρος για την υλοποίηση του πρωτοκόλλου IDMEF είναι το κανάλι δεδομένων μεταξύ του συστήματος ανίχνευσης επιθέσεων (sensor) και του συστήματος στο οποίο στέλνονται οι ειδοποιήσεις για επιθέσεις (manager).
Εκτός από τα παραπάνω, το πρωτόκολλο IDMEF μπορεί να είναι χρήσιμο και στις παρακάτω περιπτώσεις:

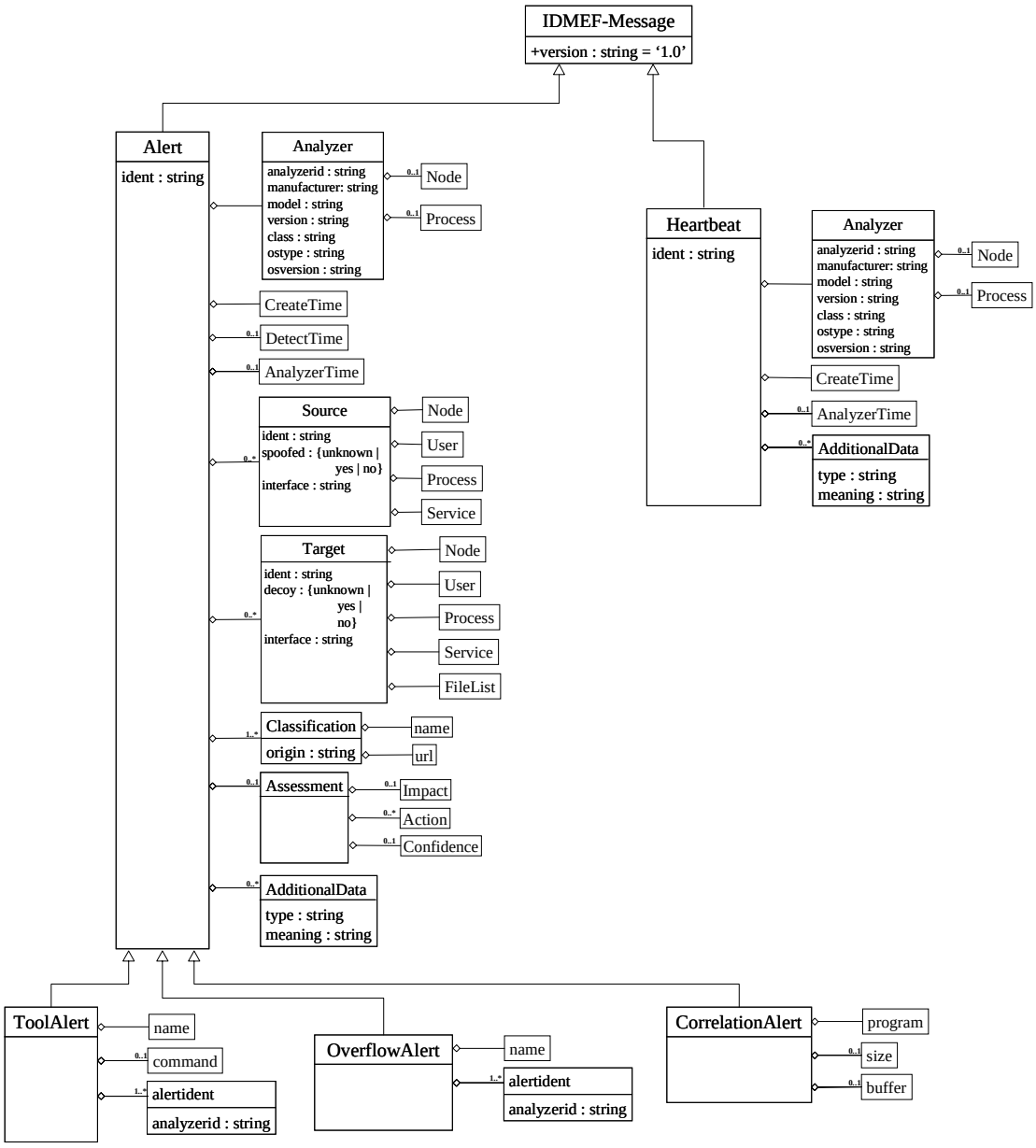
- Σε μία βάση δεδομένων που αποθηκεύει τα στοιχεία επιθέσεων που προέρχονται από διαφορετικά IDS, ώστε να έχουν την ίδια μορφή και να είναι εύκολη η ανάλυση τους.
- Σε ένα σύστημα συσχέτισης περιστατικών ασφάλειας το οποίο δέχεται πληροφορίες από πολλαπλά IDS.

- Σε ένα γραφικό περιβάλλον το οποίο θα εμφανίζει τις πληροφορίες δικτυακών επιθέσεων από διαφορετικά IDS σε μία οθόνη.
- Σε μεταφορά δεδομένων που αφορούν δικτυακές επιθέσεις ανάμεσα σε διαφορετικές ομάδες ανθρώπων (απλούς χρήστες, κατασκευαστές, ομάδες αντιμετώπισης περιστατικών ασφαλείας).

Το IDMEF data model είναι μία object-oriented αναπαράσταση των πληροφοριών ενός Alert μηνύματος που στέλνεται στους IDS managers από τους IDS sensors. Στόχος αυτού του μοντέλου είναι να παρέχει μία πρότυπη αναπαράσταση των επιθέσεων και να επιτρέψει τόσο σε απλά όσο και σε πολύπλοκα Alerts να περιγραφούν.

Δύο υλοποιήσεις του IDMEF είχαν αρχικά προταθεί. Η μία υλοποίηση χρησιμοποιεί το SMI (Structure of Management Information) για να περιγράψει μία SNMP MIB. Η άλλη υλοποίηση χρησιμοποιεί ένα DTD (Document Type Definition) για να περιγράψει ένα XML Document. Τελικά, το Φεβρουάριο του 2000 αποφασίστηκε από το IDWG (Intrusion Detection Working Group) ότι η δεύτερη υλοποίηση ικανοποιούσε περισσότερο τις απαιτήσεις της κοινότητας. Αναλυτικά η περιγραφή του IDMEF DTD υπάρχει στο Παράρτημα Β.

Παρακάτω φαίνεται το IDMEF data model σε UML αναπαράσταση.



Όπως φαίνεται και από το παραπάνω διάγραμμα η αρχική κλάση ενός IDMEF μηνύματος ονομάζεται IDMEF-Message. Κάθε τύπος μηνύματος είναι μία υποκλάση αυτής της αρχικής κλάσης. Υπάρχουν δύο τύποι μηνύματος: Τα Alerts και τα Heartbeats. Τα Alerts είναι μηνύματα που δημιουργούνται όταν ο sensor ανιχνεύσει ένα ύποπτο γεγονός (επίθεση) και το στέλνει στον manager. Σε αυτό το μήνυμα περιέχονται πληροφορίες σχετικά με την επίθεση, όπως η πηγή της επίθεσης, ο στόχος, η ακριβής ώρα καθώς και το είδος της επίθεσης. Τα Heartbeats είναι

μηνύματα που περιγράφουν την παρούσα κατάσταση των sensors στους managers. Τα μηνύματα αυτά στέλνονται σε περιοδικά διαστήματα π.χ. κάθε δέκα λεπτά ή κάθε μία ώρα. Η παραλαβή ενός τέτοιου μηνύματος από τον manager δηλώνει ότι ο sensor που το έστειλε λειτουργεί κανονικά. Η έλλειψη αυτών των μηνυμάτων υποδηλώνει πρόβλημα στη λειτουργία του sensor ή του δικτύου μέσω του οποίου στέλνεται το μήνυμα στο manager.

3.3. Κρυπτογραφία - Ψηφιακές Υπογραφές

3.3.1. Εισαγωγή

Η κρυπτογραφία έχει μακριά και γραφική ιστορία που αρχίζει περίπου πριν 2500 χρόνια. Στην εποχή μας παίζει ιδιαίτερα σημαντικό ρόλο στον τομέα των δικτύων υπολογιστών, καθώς εκατομμύρια ανθρώπων χρησιμοποιούν τα δίκτυα για τις τραπεζικές συναλλαγές τους και τις αγορές διαφόρων προϊόντων.

Η διαδικασία με την οποία μετατρέπουμε ένα μήνυμα έτσι ώστε να κρύψουμε το πραγματικό του περιεχόμενο ονομάζεται κρυπτογράφηση (encryption) και το αποτέλεσμα ονομάζεται κρυπτογράφημα (ciphertext). Η αντίστροφη διαδικασία της μετατροπής του κρυπτογραφήματος σε απλό κείμενο (plaintext) ονομάζεται αποκρυπτογράφηση (decryption).

Ένας κρυπτογραφικός αλγόριθμος (cipher) είναι μία μαθηματική συνάρτηση που χρησιμοποιείται για την κρυπτογράφηση και την αποκρυπτογράφηση. Γενικά συνήθως υπάρχουν δύο συναρτήσεις: μία για την κρυπτογράφηση και μία για τη αποκρυπτογράφηση. Εάν η ασφάλεια ενός αλγορίθμου βασίζεται στην απόκρυψη της λειτουργίας του, τότε αυτός ονομάζεται περιορισμένος αλγόριθμος (restricted algorithm). Οι περιορισμένοι αλγόριθμοι έχουν ιστορικό ενδιαφέρον, αλλά στις μέρες μας είναι ξεπερασμένοι. Μία μεγάλη ομάδα ανθρώπων ή μια ομάδα που αλλάζει συνεχώς άτομα δεν μπορεί να τους χρησιμοποιήσει διότι κάθε φορά που φεύγει ένα άτομο από την ομάδα, η υπόλοιπη ομάδα πρέπει να αλλάξει τον αλγόριθμο. Η μοντέρνα κρυπτογραφία λύνει αυτό το πρόβλημα με την χρήση ενός κλειδιού. Τόσο η διαδικασία της κρυπτογράφησης όσο και της αποκρυπτογράφησης κάνουν χρήση του κλειδιού. Έτσι συμβολίζουμε τη διαδικασία της κρυπτογράφησης ως :

$$E_K(M) = C$$

και την διαδικασία της αποκρυπτογράφησης:

$$D_K(C) = M,$$

όπου με M συμβολίζουμε το απλό κείμενο (plaintext),

C συμβολίζουμε το κρυπτογράφημα (ciphertext),

K συμβολίζουμε το κλειδί.

Μερικοί αλγόριθμοι χρησιμοποιούν διαφορετικό κλειδί για την κρυπτογράφηση και την αποκρυπτογράφηση. Έτσι οι παραπάνω εξισώσεις μετατρέπονται όπως φαίνεται παρακάτω:

$$E_{K1}(M) = C$$
$$D_{K2}(C) = M$$

Η ασφάλεια των παραπάνω αλγορίθμων βασίζεται στο κλειδί (ή στα κλειδιά) και όχι στις λεπτομέρειες του αλγορίθμου. Αυτό σημαίνει ότι ο αλγόριθμος μπορεί να δημοσιευτεί και να αναλυθεί. Όσο κάποιος δεν γνωρίζει το κλειδί, δεν μπορεί να διαβάσει τα μηνύματα που κρυπτογραφούνται με αυτό. Υπάρχουν δύο είδη αλγορίθμων κρυπτογράφησης που βασίζονται σε κλειδί: Οι συμμετρικοί αλγόριθμοι και οι αλγόριθμοι δημόσιου κλειδιού.

Οι συμμετρικοί αλγόριθμοι, οι οποίοι πολλές φορές ονομάζονται και συμβατικοί, είναι αλγόριθμοι όπου το κλειδί κρυπτογράφησης μπορεί να υπολογιστεί από το κλειδί αποκρυπτογράφησης και το αντίστροφο. Στους περισσότερους από αυτούς τους αλγορίθμους το κλειδί κρυπτογράφησης και αποκρυπτογράφησης είναι το ίδιο. Οι αλγόριθμοι αυτοί που ονομάζονται και αλγόριθμοι μυστικού κλειδιού (secret-key algorithms) απαιτούν ότι ο αποστολέας και ο παραλήπτης θα έχουν συμφωνήσει σε ένα κλειδί πριν αρχίσουν την ασφαλή επικοινωνία τους. Η ασφάλεια ενός συμμετρικού αλγορίθμου βασίζεται στο κλειδί. Αν το κλειδί μαθευτεί τότε οποιοσδήποτε θα μπορεί να κρυπτογραφεί και να αποκρυπτογραφεί μηνύματα.

Οι αλγόριθμοι δημόσιου κλειδιού, που ονομάζονται επίσης και ασύμμετροι αλγόριθμοι, είναι σχεδιασμένοι για να χρησιμοποιούν διαφορετικό κλειδί στην διαδικασία της κρυπτογράφησης από το κλειδί που χρησιμοποιείται στην διαδικασία της αποκρυπτογράφησης. Επιπρόσθετα το κλειδί της αποκρυπτογράφησης δεν μπορεί να υπολογιστεί (τουλάχιστον σε λογικά όρια χρόνου) από το κλειδί της κρυπτογράφησης. Οι αλγόριθμοι αυτοί ονομάζονται δημόσιου κλειδιού γιατί το κλειδί της κρυπτογράφησης μπορεί να δημοσιευτεί. Οποιοδήποτε άτομο με χρήση του δημόσιου κλειδιού μπορεί να κρυπτογραφήσει ένα μήνυμα, αλλά μόνο το άτομο

που διαθέτει το αντίστοιχο κλειδί αποκρυπτογράφησης μπορεί να αποκρυπτογραφήσει το μήνυμα. Σε αυτά τα συστήματα, το κλειδί κρυπτογράφησης ονομάζεται και δημόσιο κλειδί (public key), ενώ το κλειδί αποκρυπτογράφησης ονομάζεται και ιδιωτικό κλειδί (private key).

3.3.2. Κρυπτογραφία Δημόσιου Κλειδιού

Το 1976 δύο ερευνητές στο Stanford University, οι Diffie και Hellman πρότειναν έναν ριζικά καινούριο τύπο κρυπτοσυστήματος όπου τα κλειδιά κρυπτογράφησης και αποκρυπτογράφησης ήταν διαφορετικά και το κλειδί αποκρυπτογράφησης δεν μπορούσε να προκύψει από το κλειδί κρυπτογράφησης. Στην πρότασή τους, ο αλγόριθμος κρυπτογράφησης E και ο αλγόριθμος αποκρυπτογράφησης D έπρεπε να ικανοποιούν τις ακόλουθες τρεις απαιτήσεις. Αυτές οι απαιτήσεις μπορούν να διατυπωθούν με απλό τρόπο ως εξής:

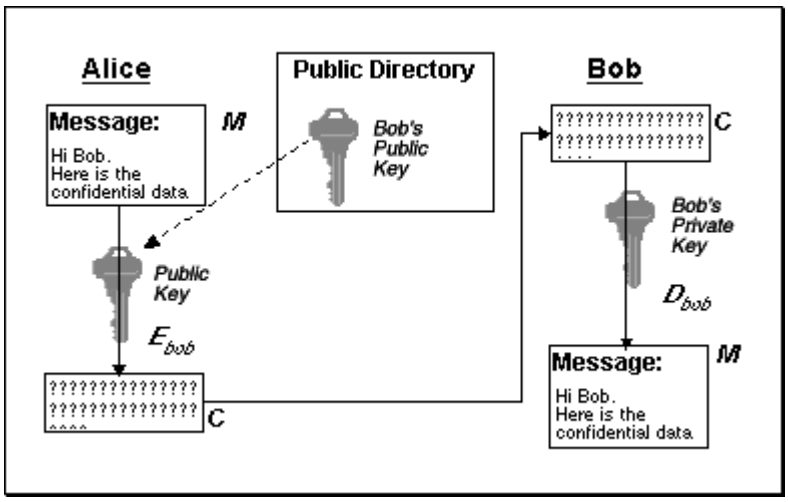
- D(E(P)) = P
- Είναι υπερβολικά δύσκολη η παραγωγή του D από το E.
- Ο αλγόριθμος E δεν μπορεί να σπάσει με επίθεση επιλεγμένου κειμένου.

Η πρώτη απαίτηση λέει ότι αν εφαρμόσουμε τον αλγόριθμο D σε κρυπτογράφημα, το E(P), θα ξαναπάρουμε το πρωτότυπο κείμενο P. Η δεύτερη απαίτηση μας εξασφαλίζει ότι από τον δημόσιο αλγόριθμο E δεν μπορούμε να παράγουμε τον αλγόριθμο D. Η τρίτη απαίτηση χρειάζεται επειδή οι παρείσακτοι μπορούν να πειραματίζονται με τον αλγόριθμο όσο θέλουν. Κάτω από αυτές τις συνθήκες δεν υπάρχει λόγος να μην δημοσιοποιηθεί το κλειδί κρυπτογράφησης.

Η μέθοδος λειτουργεί ως εξής: Ένα άτομο, ας πούμε η Alice, που επιθυμεί να λάβει μυστικά μηνύματα, επινοεί αρχικά δύο αλγορίθμους, τον E_A και τον D_A, οι οποίοι ικανοποιούν τις παραπάνω απαιτήσεις. Ο αλγόριθμος και το κλειδί κρυπτογράφησης, E_A, στη συνέχεια δημοσιοποιούνται και για το λόγο αυτό η μέθοδος αυτή παίρνει το όνομα κρυπτογραφία δημόσιου κλειδιού (public key cryptography), σε αντιδιαστολή με την παραδοσιακή κρυπτογραφία μυστικού κλειδιού.

Ας δούμε τώρα πως μπορούμε να λύσουμε το πρόβλημα της εγκατάστασης ενός ασφαλούς διαύλου ανάμεσα σε δύο άτομα, την Alice και τον Bob. Τόσο το κλειδί κρυπτογράφησης της Alice, E_A, όσο και το κλειδί κρυπτογράφησης του Bob,

E_B, υποτίθεται ότι βρίσκονται σε αρχείο αναγνώσιμο από το κοινό. Τώρα η Alice θέλει να στείλει ένα απόρρητο μήνυμα M στον Bob. Για το σκοπό αυτό η Alice παίρνει το δημόσιο κλειδί του Bob, υπολογίζει το $E_B(M)$ και το στέλνει στον Bob. Ο Bob το αποκρυπτογραφεί στην συνέχεια, εφαρμόζοντας το ιδιωτικό κλειδί D_B, δηλαδή υπολογίζει το $D_B(E_B(M)) = M$. Κανείς άλλος δεν μπορεί να διαβάσει το κρυπτογραφημένο μήνυμα $E_B(M)$ επειδή το σύστημα κρυπτογράφησης θεωρείται ισχυρό και επειδή είναι πάρα πολύ δύσκολο να παραχθεί το κλειδί αποκρυπτογράφησης D_B από το δημόσια γνωστό κλειδί E_B. Τα όσα περιγράψαμε παραπάνω φαίνονται σχηματικά στην παρακάτω εικόνα:



3.3.3. Αλγόριθμος RSA

Επειδή στην κρυπτογραφία με δημόσιο κλειδί υπάρχουν πολλά ενδεχόμενα πλεονεκτήματα, πολλοί ερευνητές δουλεύουν σκληρά και ήδη έχουν δημοσιευθεί μερικοί αλγόριθμοι. Μία καλή μέθοδος ανακαλύφθηκε από μία ομάδα στο MIT. Η μέθοδος αυτή είναι γνωστή από τα αρχικά των ονομάτων των τριών που την ανακάλυψαν και ονομάζεται RSA (Rivest, Shamir, Adleman). Η μέθοδός τους βασίζεται σε αρχές της θεωρίας αριθμών. Παρακάτω αναφέρουμε συνοπτικά το πως χρησιμοποιείται η μέθοδος:

1. Επιλέγουμε δύο μεγάλους πρώτους αριθμούς, p και q (συνήθως μεγαλύτερους από 10^{100}).
2. Υπολογίζουμε το $n = p \bullet q$ και $z = (p-1) \bullet (q-1)$.
3. Επιλέγουμε έναν πρώτο αριθμό ως προς το z και τον ονομάζουμε d.
4. Βρίσκουμε έναν αριθμό e, έτσι ώστε $e \bullet d = 1 \bmod z$

Έχοντας υπολογίσει εκ των προτέρων αυτές τις παραμέτρους, είμαστε έτοιμοι να ξεκινήσουμε την κρυπτογράφηση. Διαιρούμε το κείμενο σε μπλοκ, έτσι ώστε κάθε μήνυμα κειμένου P να πέφτει στο διάστημα $0 \leq P < n$. Αυτό μπορεί να γίνει με ομαδοποίηση του κειμένου σε μπλοκ των k bits, όπου το k είναι ο μεγαλύτερος ακέραιος για τον οποίο η σχέση $2^k < n$ είναι αληθής.

Για να κρυπτογραφήσουμε ένα μήνυμα P, υπολογίζουμε το $C = P^e \pmod n$. Για να αποκρυπτογραφήσουμε το C, υπολογίζουμε το $P = C^d \pmod n$. Μπορεί να αποδειχθεί ότι για όλα τα P στην καθορισμένη περιοχή, οι λειτουργίες κρυπτογράφησης και αποκρυπτογράφησης είναι αντίστροφες. Για την κρυπτογράφηση χρειαζόμαστε τα e και n, ενώ για την αποκρυπτογράφηση τα d και n. Επομένως, το δημόσιο κλειδί αποτελείται από το ζευγάρι (e, n) και το ιδιωτικό κλειδί από το ζευγάρι (d, n).

Η ασφάλεια της μεθόδου βασίζεται στη δυσκολία της παραγοντοποίησης μεγάλων αριθμών. Εάν ο κρυπταναλυτής μπορούσε να παραγοντοποιήσει το δημόσια γνωστό n, στη συνέχεια θα μπορούσε να βρει τα p και q και από αυτά το z. Εάν διαθέτει τα z και e, τότε μπορεί να βρει το d με τη βοήθεια του αλγόριθμου του Ευκλείδη. Ευτυχώς οι μαθηματικοί προσπαθούν να παραγοντοποιήσουν μεγάλους αριθμούς εδώ και 300 τουλάχιστον χρόνια, αλλά οι συσσωρευμένες ενδείξεις συνηγορούν ότι πρόκειται για υπερβολικά δύσκολο πρόβλημα.

3.3.4. One-way Hash Functions - Αλγόριθμος MD5

Μία μονόδρομη συνάρτηση ανάδευσης (One-way Hash Function) παίρνει ένα αυθαίρετα μακρύ κομμάτι κειμένου και υπολογίζει από αυτό ένα συρμό από bits σταθερού μήκους. Αυτή η συνάρτηση ανάδευσης, που συχνά αποκαλείται και σύνοψη μηνύματος (Message Digest), παρουσιάζει τρεις σημαντικές ιδιότητες.

1. Δεδομένου του P, είναι εύκολο να υπολογιστεί το MD(P).
2. Δεδομένου του MD(P), είναι πρακτικά αδύνατο να υπολογιστεί το P.
3. Κανείς δεν μπορεί να δημιουργήσει δύο μηνύματα που να έχουν την ίδια σύνοψη μηνύματος.

Για να ικανοποιηθεί το κριτήριο 3, ο συρμός ανάδευσης θα πρέπει να έχει μήκος τουλάχιστον 128 bit, προτιμότερο δε να είναι και μεγαλύτερος.

Ο υπολογισμός της σύνοψης μηνύματος από ένα κομμάτι κειμένου είναι πολύ ταχύτερος από την κρυπτογράφηση αυτού του κειμένου με τη χρήση ενός αλγόριθμου δημοσίου κλειδιού, έτσι η σύνοψη μηνυμάτων μπορεί να χρησιμοποιηθεί για να επιταχυνθούν οι αλγόριθμοι ψηφιακής υπογραφής όπως θα δούμε παρακάτω.

Έχει προταθεί μία ποικιλία από συναρτήσεις σύνοψης μηνυμάτων. Αυτές που χρησιμοποιούνται περισσότερο είναι οι MD5 (Rivest 1992) και SHA (NIST 1993). Το MD5 είναι η πέμπτη σε μία σειρά από συναρτήσεις ανάδευσης που σχεδιάστηκαν από τον Ron Rivest. Ο αλγόριθμος λειτουργεί αναμειγνύοντας τα bits με αρκετά περίπλοκο τρόπο ώστε κάθε bit εξόδου να επηρεάζεται από κάθε bit εισόδου. Σε συντομία, αρχίζει να συμπληρώνει το μήνυμα μέχρι το μήκος των 448 bits (modulo 512). Μετά, το μήκος του αρχικού μηνύματος προσαρτάται ως ένας ακέραιος των 64 bit ώστε να δώσει μία είσοδο συνολικού μήκους που είναι πολλαπλάσιο των 512 bits. Το τελευταίο βήμα προ του υπολογισμού είναι η αρχικοποίηση ενός καταχωρητή των 128 bit με μια σταθερή τιμή. Τώρα αρχίζει ο υπολογισμός. Ο κάθε γύρος παίρνει ένα μπλοκ εισόδου των 512 bits και τα ανακατεύει με τον καταχωρητή των 128 bits. Σε κάθε μπλοκ εισόδου εκτελούνται τέσσερις γύροι. Η διαδικασία συνεχίζεται μέχρι να καταναλωθούν όλα τα μπλοκ εισόδου. Στο τέλος, τα περιεχόμενα του καταχωρητή των 128 bits σχηματίζουν τη σύνοψη μηνύματος.

3.3.5. Ψηφιακές Υπογραφές

Η αυθεντικότητα πολλών νομικών, οικονομικών και άλλων εγγράφων καθορίζεται από την παρουσία ή την απουσία μιας εξουσιοδοτημένης, χειρόγραφης υπογραφής. Τα φωτοαντίγραφα δεν μετρούν. Για να αντικατασταθεί η φυσική μεταφορά εγγράφων (από χαρτί και μελάνι) από συστήματα ανταλλαγής μηνυμάτων μέσω υπολογιστών, πρέπει να βρεθεί μία λύση αυτών των προβλημάτων.

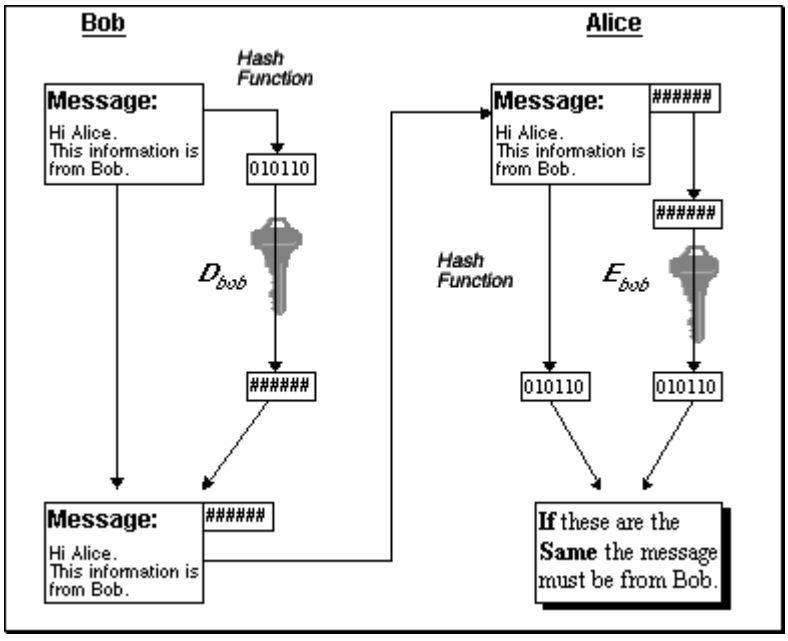
Το πρόβλημα της επινόησης ενός υποκατάστατου των χειρόγραφων υπογραφών είναι δύσκολο. Βασικά, αυτό που χρειάζεται είναι ένα σύστημα όπου το ένα μέρος να μπορεί να στείλει ένα “υπογεγραμμένο” μήνυμα στο άλλο μέρος με τέτοιο τρόπο ώστε:

1. Ο αποδέκτης να μπορεί να επιβεβαιώσει την ταυτότητα που ισχυρίζεται ότι έχει ο αποστολέας.
2. Ο αποστολέας να μην μπορεί αργότερα να αποκηρύξει τα περιεχόμενα του μηνύματος.

3. Ο αποδέκτης να μην μπορεί να έχει κατασκευάσει το μήνυμα μόνος του.

Ευτυχώς η κρυπτογραφία δημόσιου κλειδιού μπορεί να συμβάλλει σημαντικά και εδώ. Υποθέτουμε ότι οι αλγόριθμοι κρυπτογράφησης και αποκρυπτογράφησης δημόσιου κλειδιού έχουν την ιδιότητα ότι E(D(P)) = P επιπλέον της συνήθους ιδιότητας ότι D(E(P)) = P. Ο αλγόριθμος RSA έχει αυτή την ιδιότητα, έτσι η υπόθεση δεν είναι παράλογη.

Οι συνόψεις μηνυμάτων, όπως αναφέραμε και προηγουμένως, χρησιμοποιούνται στις ψηφιακές υπογραφές με χρήση των κρυπτοσυστημάτων δημόσιου κλειδιού. Για να στείλει ένα άτομο, έστω ο Bob, ένα ψηφιακά υπογεγραμμένο μήνυμα, σε ένα άλλο άτομο, την Alice, ακολουθείται η παρακάτω διαδικασία. Ο Bob υπολογίζει πρώτα τη σύνοψη μηνύματος MD(P) του κειμένου P που θέλει να στείλει. Στη συνέχεια, υπογράφει τη σύνοψη μηνύματος και στέλνει τόσο την υπογεγραμμένη σύνοψη $D_B(MD(P))$ όσο και το μήνυμα P (χωρίς κρυπτογράφηση) στην Alice. Η Alice, όταν πάρει το μήνυμα, υπολογίζει τη σύνοψη του μηνύματος χρησιμοποιώντας την ίδια Hash Function με αυτή που χρησιμοποίησε και ο Bob. Ταυτόχρονα, χρησιμοποιώντας το δημόσιο κλειδί του Bob, E_B, υπολογίζει τη σύνοψη που έστειλε ο Bob, δηλαδή $E_B(D_B(MD(P)) = MD(P)$. Αν οι δύο συνόψεις μηνύματος είναι οι ίδιες, τότε το μήνυμα είναι όντως από τον Bob και το περιεχόμενό του δεν έχει υποστεί καμία αλλαγή. Σχηματικά η διαδικασία που περιγράψαμε φαίνεται στην παρακάτω εικόνα:



3.4. IP Multicast

Το IP Multicast είναι μια τεχνολογία που έχει σαν στόχο να μειώσει την κυκλοφορία πακέτων στο δίκτυο και ταυτόχρονα να παραδώσει ένα ενιαίο ρεύμα πληροφοριών σε πολλούς πελάτες. Στις εφαρμογές που εκμεταλλεύονται την τεχνολογία αυτή περιλαμβάνονται η τηλεδιάσκεψη, οι εταιρικές επικοινωνίες, η τηλεκπαίδευση και άλλες όπως η μετάδοση της κατάστασης των χρηματιστηριακών μετοχών και των ειδήσεων.

Το Multicast μεταδίδει πληροφορία από την πηγή σε πολλαπλούς δέκτες χωρίς προσθήκη οποιουδήποτε πρόσθετου φορτίου στην πηγή ή τους δέκτες χρησιμοποιώντας λιγότερο bandwidth από οποιαδήποτε ανταγωνιστική τεχνολογία. Τα πολλαπλής διανομής πακέτα αντιγράφονται στο δίκτυο από τους ενδιάμεσους δρομολογητές όπου αυτό χρειάζεται, με συνέπεια την αποδοτικότερη παράδοση τους. Όλες οι εναλλακτικές λύσεις απαιτούν από την πηγή να στείλει περισσότερα από ένα αντίγραφα των πακέτων. Εάν υπάρχουν χιλιάδες δέκτες ή οι ενδιάμεσες γραμμές έχουν συμφόρηση, ακόμη και οι εφαρμογές χαμηλού bandwidth ωφελούνται από τη χρησιμοποίηση της τεχνολογίας αυτής.

Οι εφαρμογές υψηλού bandwidth, όπως το βίντεο MPEG, μπορούν να απαιτήσουν μια μεγάλη μερίδα του διαθέσιμου εύρους ζώνης δικτύων για ένα ενιαίο ρεύμα. Σε αυτές τις εφαρμογές, ο μόνος τρόπος να στείλει κανείς πληροφορία σε περισσότερους από έναν δέκτες ταυτόχρονα είναι η χρησιμοποίηση του Multicast.

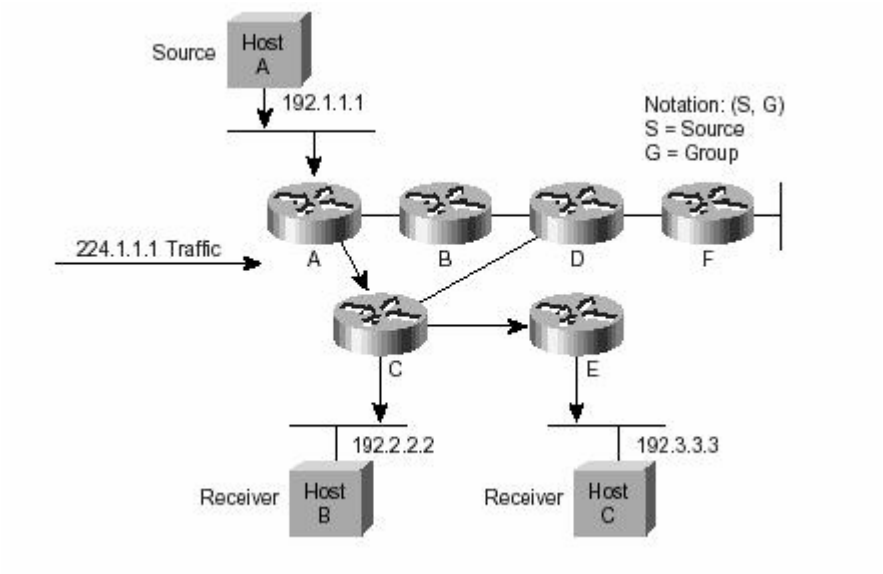
Το Multicast είναι βασισμένο στην έννοια μιας ομάδας. Μια ομάδα δεκτών εκφράζει ένα ενδιαφέρον για τη λήψη μιας ιδιαίτερης ροής στοιχείων. Αυτή η ομάδα δεν έχει φυσικά ή γεωγραφικά όρια, οι δέκτες μπορούν να βρίσκονται οπουδήποτε στο διαδίκτυο. Αυτοί που ενδιαφέρονται για τη λήψη των στοιχείων που μεταδίδονται σε μια συγκεκριμένη ομάδα πρέπει να προσχωρήσουν στην ομάδα αυτή χρησιμοποιώντας το πρωτόκολλο IGMP (Internet Group Management Protocol).

Μια Multicast ομάδα ορίζεται από μια διεύθυνση. Οι διευθύνσεις αυτές είναι ειδικές και αποτελούν το σύνολο των παραληπτών που επιθυμούν να λάβουν την πληροφορία που στέλνεται σε αυτήν την ομάδα. Υπεύθυνη για την ανάθεση των Multicast διευθύνσεων είναι η αρχή IANA (Internet Assigned Numbers Authority). Η IANA έχει δεσμεύσει το διάστημα των IP διευθύνσεων: 224.0.0.0 - 239.255.255.255 για χρησιμοποίηση από εφαρμογές Multicast. Ειδικότερα, οι διευθύνσεις από 224.0.0.0 μέχρι 224.0.0.255 μπορούν να χρησιμοποιηθούν μόνο τοπικά. Πακέτα με

διευθύνσεις σε αυτό το διάστημα δεν προωθούνται ποτέ από ένα δρομολογητή και ο χρόνος ζωής τους (TTL) είναι πάντα ίσος με 1. Χρησιμοποιούνται συνήθως από δικτυακά πρωτόκολλα για μεταβίβαση πληροφοριών, για παράδειγμα με την 224.0.0.2 επικοινωνεί κανείς με όλους τους δρομολογητές στο ίδιο υποδίκτυο. Το διάστημα 224.0.1.0 - 238.255.255.255 είναι διευθύνσεις που μπορούν να χρησιμοποιηθούν σε παγκόσμια κλίμακα. Μερικές από αυτές είναι δεσμευμένες για χρησιμοποίηση από δικτυακά πρωτόκολλα, για παράδειγμα η 224.0.1.1 χρησιμοποιείται από το πρωτόκολλο NTP (Network Time Protocol). Το διάστημα 239.0.0.0 - 239.255.255.255 είναι διευθύνσεις περιορισμένου εύρους που μπορούν να χρησιμοποιηθούν από μία τοπική ομάδα ή από κάποιον οργανισμό.

Όπως αναφέραμε προηγουμένως, για τη σύνδεση ενός υπολογιστή σε μια multicast ομάδα χρησιμοποιείται το Internet Group Management Protocol. Οι υπολογιστές που θέλουν να ακούσουν μια μετάδοση Multicast, στέλνουν πακέτα IGMP στον τοπικό τους δρομολογητή. Οι δρομολογητές που υποστηρίζουν την τεχνολογία αυτή ακούν για IGMP μηνύματα και περιοδικά κάνουν ερωτήσεις για να μάθουν ποιες ομάδες είναι ενεργές και ποιες όχι στο υποδίκτυο τους. Υπάρχουν δυο εκδόσεις του πρωτοκόλλου IGMP. Στη δεύτερη έκδοση οι υπολογιστές που αποχωρούν από το group στέλνουν ένα αντίστοιχο μήνυμα ειδοποιώντας το δρομολογητή και τότε αυτός ελέγχει αν έχει μείνει κανείς στο group προκειμένου να συνεχίσει τη μετάδοση. Με τη διαδικασία αυτή μειώνεται η κίνηση σε σχέση με την πρώτη έκδοση που δεν υποστηρίζει μηνύματα αποχώρησης.

Οι δρομολογητές προκειμένου να μεταδώσουν τις ροές των Multicast πακέτων δημιουργούν δέντρα. Κορυφή του δέντρου μπορεί να θεωρηθεί η πηγή της Multicast μετάδοσης και φύλλα οι δέκτες. Τα δέντρα φτιάχνονται με βάση τον αλγόριθμο Shortest Path Tree και δεν περιέχουν βρόχους. Τα μέλη μιας Multicast ομάδας μπορούν να μπουν και να βγουν σε οποιαδήποτε χρονική στιγμή, οπότε τα δέντρα πρέπει να ενημερώνονται δυναμικά. Όταν όλοι οι δέκτες σε ένα κλαδί αποχωρήσουν από την ομάδα, τότε το κλαδί καταργείται.



Στο παραπάνω δίκτυο ως πηγή εκπομπής θεωρούμε τον Host A, ενώ ως δέκτες των Multicast πακέτων τους Hosts B και C. Κάθε Multicast πακέτο στέλνεται μία φορά στον δρομολογητή A. Ο δρομολογητής A αναλαμβάνει να το προωθήσει στους κατάλληλους δρομολογητές οι οποίοι έχουν πελάτες στο συγκεκριμένου Multicast group. Με αυτό το σκεπτικό ο δρομολογητής A προωθεί το πακέτο στον δρομολογητή C. Με τη σειρά του ο δρομολογητής C προωθεί το πακέτο στον Host B ο οποίος είναι τελικός παραλήπτης ενώ συγχρόνως αντιγράφει το πακέτο και στον δρομολογητή E. Τέλος ο δρομολογητής E προωθεί το πακέτο στον τελικό παραλήπτη Host C.

4. Θεωρητική προσέγγιση του συστήματος

4.1. Περιγραφή προβλήματος

Με βάση όσα γράφτηκαν στα προηγούμενα κεφάλαια, έχουμε διαπιστώσει ότι οι δικτυακές επιθέσεις είναι καθημερινό φαινόμενο. Οι επιθέσεις αυτές συχνά αφορούν είτε μεμονωμένους υπολογιστές είτε ολόκληρα δίκτυα υπολογιστών. Στόχος τους είναι να πλήξουν συγκεκριμένες υπηρεσίες όπως είναι το web, το email, το ftp, ενώ μερικές φορές στόχος επιθέσεων αποτελεί και ολόκληρη η δικτυακή υποδομή μεγάλων εμπορικών δικτύων. Για τις πρώτες, συχνά οι επιτιθέμενοι χρησιμοποιούν “security vulnerabilities” των ίδιων των υπηρεσιών ή του λειτουργικού συστήματος στο οποίο αυτές τρέχουν. Οι δεύτερες επιθέσεις που είναι γνωστές με τον όρο Denial of Service Attacks έχουν ως σκοπό να πλημμυρίσουν τις γραμμές των δικτύων αυτών με άχρηστα πακέτα, ώστε να αποτρέψουν τη ροή των κανονικών πακέτων.

Οι παραπάνω επιθέσεις τη σημερινή εποχή έχουν γίνει πολύπλοκες και είναι δύσκολο να ανιχνευτούν από ένα μεμονωμένο σύστημα ανίχνευσης επιθέσεων (IDS). Συχνά ένας κακόβουλος χρήστης χρησιμοποιεί περισσότερους από έναν υπολογιστή για να πραγματοποιήσει μία επίθεση. Ένα απλό σύστημα ανίχνευσης επιθέσεων ίσως να μην θεωρήσει ύποπτη την κίνηση που προέρχεται από μία πηγή. Αν όμως έχουμε τη συνολική εικόνα του δικτύου, μία ροή πακέτων που φαινομενικά ήταν ακίνδυνη ως μεμονωμένη ροή, αν την εξέταζε κάποιος προσεκτικά σαν συνδυασμό μαζί με άλλες ροές πακέτων θα την κατέτασσε στις ύποπτες.

Σε αυτό το πρόβλημα έρχεται να προστεθεί και η μεγάλη ανάπτυξη των δικτύων όπου συχνά αυτά αποτελούνται από μερικές χιλιάδες υπολογιστές και εκατοντάδες υποδίκτυα. Σε τέτοια δίκτυα η ανίχνευση επιθέσεων είναι πολύ δύσκολη και σε μερικές περιπτώσεις αδύνατη με τη χρήση απλών συστημάτων ανίχνευσης επιθέσεων.

4.2. Λύση προβλήματος

Με την παρούσα διπλωματική εργασία προτείνουμε έναν τρόπο για την επίλυση του προβλήματος που περιγράψαμε στην προηγούμενη παράγραφο. Η λύση που προτείνουμε περιλαμβάνει ένα κατανεμημένο δίκτυο από συστήματα ανίχνευσης επιθέσεων (IDS Sensors) διασκορπισμένα σε όλο το δίκτυο που θέλουμε να

ελέγχουμε, τα οποία θα στέλνουν τις πληροφορίες που συλλέγουν σχετικά με επιθέσεις σε ένα κεντρικό κόμβο (Central IDS Node) ο οποίος θα πραγματοποιεί την ανάλυση των περιστατικών ασφαλείας κάνοντας χρήση των πληροφοριών από όλους τους IDS Sensors.

Με τον τρόπο αυτό οι διαχειριστές ασφαλείας του δικτύου θα μπορούν να ανιχνεύουν κακόβουλους χρήστες που επιτίθενται σε ολόκληρο το δίκτυο και όχι σε ένα κομμάτι αυτού. Επίσης, με βάση τη συνάθροιση των πληροφοριών που μαζεύει ο κεντρικός κόμβος θα μπορεί να εντοπίσει τις πραγματικές πηγές των επιθέσεων και να πάρει μέτρα για να τις αντιμετωπίσει.

Σε ένα τέτοιο σύστημα θέτονται ερωτήματα τόσο για την ασφάλεια των πληροφοριών που διακινούνται από τους IDS Sensors προς τον κεντρικό κόμβο όσο και για την επιβάρυνση του δικτύου που δέχεται επίθεση με πρόσθετα πακέτα που δημιουργούνται από τους IDS sensors και περιέχουν τις πληροφορίες για την επίθεση. Παρακάτω θα περιγράψουμε αναλυτικά την αρχιτεκτονική του κατανεμημένου συστήματος ανίχνευσης επιθέσεων και θα εξηγήσουμε πως επιλύσαμε τους πιο πάνω προβληματισμούς.

4.3. Αρχιτεκτονική του συστήματος

Ο βασικός σκοπός του κατανεμημένου συστήματος ανίχνευσης επιθέσεων είναι να δημιουργήσουμε ένα δίκτυο από IDS Sensors καθένας από τους οποίους θα είναι υπεύθυνος για κάποιο κομμάτι του δικτύου και θα αναφέρει τις επιθέσεις που ανιχνεύει σε κάποιο κεντρικό κόμβο. Με τους πολλαπλούς IDS sensors τοποθετημένους σε διαφορετικά σημεία του δικτύου επιτυγχάνουμε καλύτερη ανίχνευση των επιθέσεων γιατί ελαχιστοποιούμε την απώλεια πακέτων που περνάνε από κάθε ανιχνευτή. Σε διαφορετική περίπτωση θα έπρεπε να τοποθετήσουμε το μεμονωμένο σύστημα ανίχνευσης στην περίμετρο του δικτύου μας, π.χ. στον κεντρικό δρομολογητή ή στο firewall για να ελέγξουμε όλα τα πακέτα που κινούνται από και προς το δίκτυό μας.

Επειδή ο IDS Sensor και ο κεντρικός κόμβος που συλλέγει τις πληροφορίες είναι διαφορετικές οντότητες, πρέπει να βρούμε ένα τρόπο επικοινωνίας μεταξύ τους για να ανταλλάσσουν τις πληροφορίες σχετικά με τις επιθέσεις. Για το σκοπό αυτό επιλέξαμε το πρωτόκολλο IDMEF (Intrusion Detection Message Exchange Format),

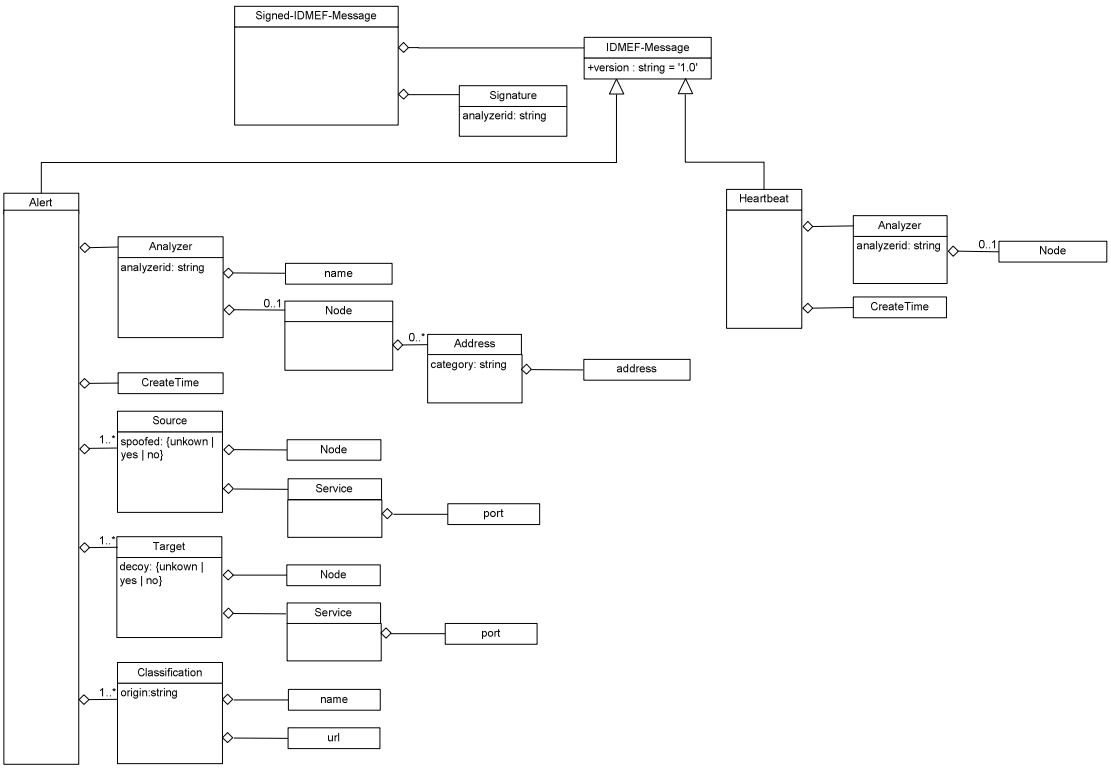
για το οποίο έχουμε κάνει λόγο σε προηγούμενο κεφάλαιο. Το πρωτόκολλο IDMEF εκτός από τα μηνύματα τα οποία περιγράφουν κάποια επίθεση (Alert messages) υποστηρίζει και έναν άλλο τύπο μηνυμάτων τα οποία στέλνει ο Sensor στο κεντρικό κόμβο για να δηλώσει την κατάσταση της λειτουργίας του (Heartbeat messages). Τα μηνύματα αυτά θα τα χρησιμοποιήσουμε στο σύστημα μας για να γνωρίζουμε κάθε στιγμή ποιος από τους IDS Sensors έχει πρόβλημα. Τα μηνύματα αυτά στέλνονται περιοδικά από κάθε Sensor στον κεντρικό IDS κόμβο. Διακοπή των μηνυμάτων αυτών από κάποιο Sensor υποδηλώνει ότι ο συγκεκριμένος Sensor έχει τεθεί εκτός λειτουργίας.

Η ανταλλαγή των μηνυμάτων αυτών θα πρέπει να γίνεται με ασφαλή τρόπο. Συγκεκριμένα υπάρχουν τρία προβλήματα που πρέπει να επιλυθούν:

- Εξασφάλιση της ταυτότητας του αποστολέα του μηνύματος
- Διασφάλιση της ακεραιότητας του περιεχομένου του μηνύματος
- Αποτροπή επανάληψης ενός μηνύματος από κάποιον υποκλοπέα

Τα παραπάνω προβλήματα λύθηκαν με την επέκταση του πρωτοκόλλου IDMEF ώστε να περιλαμβάνει μαζί με το μήνυμα και την ψηφιακή του υπογραφή. Από αυτά που έχουμε αναφέρει σχετικά με τις ψηφιακές υπογραφές εύκολα μπορούμε να διαπιστώσουμε την επίλυση των δύο πρώτων προβλημάτων. Η εξάλειψη του τελευταίου προβλήματος προκύπτει από το γεγονός ότι στις πληροφορίες ενός IDMEF μηνύματος υπάρχει και η ακριβής ώρα δημιουργίας του, γεγονός που καθιστά την επανάληψή του αδύνατη από κάποιον που υπέκλεψε το μήνυμα.

Παρακάτω φαίνεται το UML διάγραμμα των μηνυμάτων που ανταλλάσσει το κατανεμημένο σύστημά μας:



Όπως μπορούμε να δούμε, έχουμε προσθέσει ένα νέο αρχικό tag στο XML IDMEF μήνυμα. Το νέο αυτό tag ονομάζεται <Signed-IDMEF-Message> και περιέχει δύο στοιχεία: ένα στοιχείο τύπου <IDMEF-Message> και ένα στοιχείο <Signature>. Το πρώτο αποτελεί ένα κανονικό IDMEF-Message όπως αυτό ορίζεται στο Internet-Draft του IETF. Το δεύτερο στοιχείο αποτελεί την ψηφιακή υπογραφή του μηνύματος σε δεκαεξαδική μορφή. Το γνώρισμα analyzerid φανερώνει το Sensor ο οποίος υπέγραψε το μήνυμα.

Παραδείγματα από Signed-IDMEF-Messages τύπου Alert και Heartbeat φαίνονται παρακάτω:

```
<Signed-IDMEF-Message>
 <IDMEF-Message version="1.0">
  <Alert>
   <Analyzer analyzerid="sensor1">
    <Node>
     <name>eleni.netmode.ece.ntua.gr</name>
     <Address category="ipv4-addr">
      <address>147.102.13.11</address>
     </Address>
    </Node>
   </Analyzer>
   <CreateTime ntpstamp="0xc270abec.0xee2ac322">
    2003-05-17T12:39:08Z
   </CreateTime>
   <Source spoofed="unknown">
    <Node>
     <Address category="ipv4-addr">
      <address>219.94.88.197</address>
     </Address>
    </Node>
    <Service>
     <port>1264</port>
    </Service>
   </Source>
   <Target decoy="unknown">
    <Node>
     <Address category="ipv4-addr">
      <address>147.102.13.1</address>
     </Address>
    </Node>
    <Service>
     <port>23</port>
    </Service>
   </Target>
   <Classification origin="vendor-specific">
    <name>SCAN SYN FIN</name>
    <url>http://www.whitehats.com/info/IDS198</url>
   </Classification>
  </Alert>
 </IDMEF-Message>
 <Signature analyzerid="sensor1">
509c9cbbd0a321d30040e2b47133d5d2ea8060ba82185b43656399aa4fa2cf18d232ddd51
dad167c1b3f2a7b3c3ae22adfeeb28f602fe2a9fd6df7f2f40979777aa3af03b123565b21
25f9349e7e98dc0098b9e278d46ec457054f71bab7dc6e80af445868f0f34a25780439b9f
68536270643342e809c629fb74867f86f4de5
 </Signature>
</Signed-IDMEF-Message>
```

Στο παραπάνω παράδειγμα ενός Alert μηνύματος μπορούμε να δούμε ότι ο υπολογιστής με IP διεύθυνση 219.94.88.197 έστειλε ένα πακέτο TCP προς τον υπολογιστή του δικτύου μας με IP διεύθυνση 147.102.13.1 στη θύρα 23. Το πακέτο αυτό είχε ενεργοποιημένα τα flags SYN+FIN στον TCP header, γεγονός που το καθιστά ύποπτο και πιθανώς αποτελεί μέρος ενός port scanning που διέπραξε κάποιος κακόβουλος χρήστης.

```
<Signed-IDMEF-Message>
 <IDMEF-Message version="1.0">
  <Heartbeat>
   <Analyzer analyzerid="sensor1">
    <Node>
     <name>eleni.netmode.ece.ntua.gr</name>
     <Address category="ipv4-addr">
      <address>147.102.13.11</address>
     </Address>
    </Node>
   </Analyzer>
   <CreateTime ntpstamp="0xc26ce925.0xe3db6e50">
    2003-05-14T16:11:17Z
   </CreateTime>
  </Heartbeat>
 </IDMEF-Message>
 <Signature analyzerid="sensor1">
bc0918f645fbda156150490e9a4ff9e5d0642e63d72e7b8566ed3b04f1edb65098b30afab
7b2026e2e718fd3c10fa5a9c518f71ae35e448fc9a2c22f161cb2b82cad7870dc500ddb85
59d84699bd576d592b2da97de1268aa52af6ba7414f66d5a4db3c7f1267f9483a83a9263b
afefdb6359f3c26cf0ffb23e9bcc0ce2626c7
 </Signature>
</Signed-IDMEF-Message>
```

Το παραπάνω Heartbeat message αποτελεί ένδειξη ότι ο IDS Sensor που είναι γνωστός στο κατανεμημένο δίκτυο με το όνομα “sensor1” τη δεδομένη χρονική στιγμή που εμφανίζεται στο πιο πάνω μήνυμα, λειτουργούσε κανονικά.

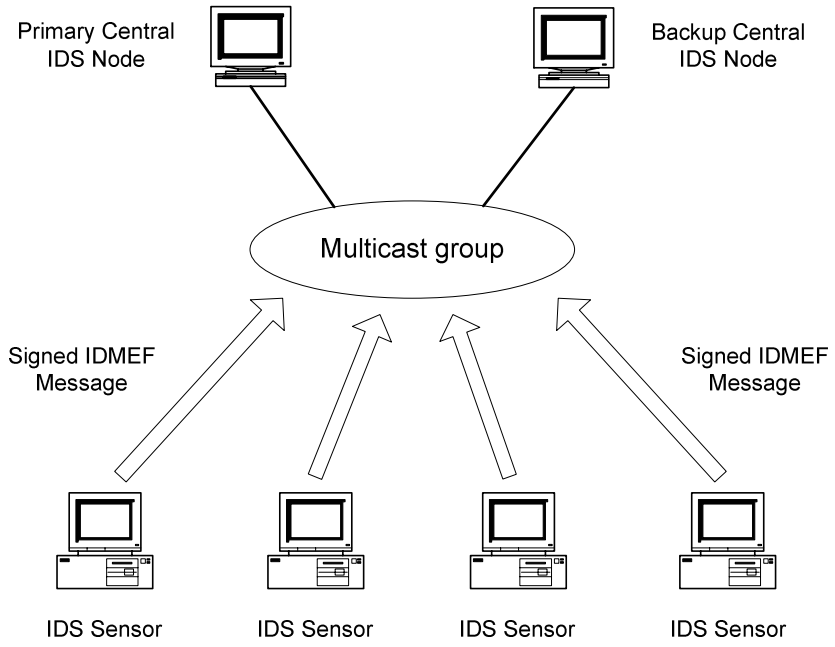
Εδώ πρέπει να σημειώσουμε ότι εκτός από την μεταφορά των πληροφοριών της επίθεσης από τους IDS sensors προς τον κεντρικό κόμβο, για λόγους ασφάλειας κρατούνται και τοπικά logs σε κάθε sensor με τα πακέτα που αποτελούν μέρος επιθέσεων. Αυτό κρίνεται σκόπιμο στην περίπτωση που θέλουμε αργότερα να διερευνήσουμε σε βάθος κάποιο περιστατικό ασφάλειας ή να αποδείξουμε ότι δεχτήκαμε επίθεση. Για αυτό το σκοπό τα logs που κρατούνται τοπικά στο δίσκο κάθε IDS sensor υπογράφονται ψηφιακά.

Όπως αναφέραμε και προηγουμένως, σε ένα δίκτυο που δέχεται επίθεση τα πακέτα που στέλνουν οι Sensors πρέπει να επιβαρύνουν όσο το δυνατόν λιγότερο το δίκτυο. Για το λόγο αυτό επιλέξαμε να χρησιμοποιήσουμε το IP Multicast για τη μεταφορά αυτών των μηνυμάτων. Με τον τρόπο αυτό πετυχαίνουμε τη λιγότερη δυνατή κίνηση στο δίκτυο και συγχρόνως επειδή ένα multicast πακέτο μπορεί να έχει περισσότερους από έναν αποδέκτες μας δίνεται η δυνατότητα να έχουμε περισσότερα από ένα Central IDS Nodes.

Έτσι, στην αρχιτεκτονική του κατανεμημένου συστήματος ανίχνευσης επιθέσεων έχουμε προβλέψει και την λειτουργία ενός δεύτερου Backup Central IDS

Node το οποίο λαμβάνει και αυτό τα μηνύματα που στέλνουν οι Sensors και είναι έτοιμο να αντικαταστήσει το Primary Central IDS Node σε περίπτωση που παρουσιαστεί σε αυτό κάποιο πρόβλημα, π.χ. γίνει το ίδιο θύμα επίθεσης.

Παρακάτω φαίνεται η δομή του κατανεμημένου δικτύου:



Στα πλαίσια αυτής της διπλωματικής εργασίας ο κεντρικός κόμβος που συλλέγει τα μηνύματα, σχεδιάστηκε ώστε απλά να αποθηκεύει τα μηνύματα σε μία βάση δεδομένων. Μελλοντικός στόχος θα ήταν να μπορεί να αναλύει τα στοιχεία που μαζεύει και να λαμβάνει μέτρα ώστε να αντιμετωπίζει σε πραγματικό χρόνο τις επιθέσεις.

5. Υλοποίηση του συστήματος

5.1. Εισαγωγή στο Snort

Το Snort είναι ένας libpcap-based packet sniffer που χρησιμοποιείται ευρέως ως ένα σύστημα ανίχνευσης επιθέσεων σε δίκτυα υπολογιστών. Έχει την δυνατότητα να πραγματοποιεί rule-based logging και να ανιχνεύει επιθέσεις όπως DoS/DDoS Attacks, Portscans, buffer overflow attacks, HTTP/DNS/SMTP/POP3/IMAP attacks που βασίζονται σε κανόνες που ικανοποιούν συγκεκριμένα IP πακέτα. Οι ειδοποιήσεις για τις επιθέσεις γίνονται σε πραγματικό χρόνο και εκτός από την καταγραφή τους σε κάποιο alert file, το Snort έχει την δυνατότητα να στέλνει τις ειδοποιήσεις στο SYSLOG, να παράγει SNMP traps, ακόμη και να στέλνει winpopup μηνύματα σε υπολογιστές με λειτουργικό σύστημα Windows με τη χρήση του πρωτοκόλλου SMB. Τα πακέτα καταγράφονται στο δίσκο είτε σε tcpdump binary μορφή, είτε σε text μορφή σε directories που δημιουργούνται με βάση την IP διεύθυνση της πηγής.

5.2. Επέκταση του Snort

Το Snort αποτελεί το λογισμικό που θα χρησιμοποιήσουμε στους IDS Sensors. Για το σκοπό αυτό επεκτείναμε τον κώδικά του, ώστε να υλοποιεί τις απαιτήσεις που εξηγήσαμε στο προηγούμενο κεφάλαιο. Πιο συγκεκριμένα, επεκτείναμε το plugin που καταγράφει σε logs τα πακέτα των επιθέσεων ώστε να δημιουργεί για κάθε τέτοιο log αρχείο και ένα καινούριο αρχείο που θα περιέχει την ψηφιακή του υπογραφή. Επιπρόσθετα, για την παραγωγή και την αποστολή των IDMEF μηνυμάτων στο κεντρικό IDS node κατασκευάσαμε ένα καινούριο output plugin. Το plugin αυτό θα είναι υπεύθυνο για την δημιουργία των Alert μηνυμάτων αλλά και των Heartbeat μηνυμάτων. Για τη δημιουργία και αποστολή των δεύτερων, το plugin κατά την αρχικοποίηση του θα δημιουργεί μία νέα διεργασία η οποία θα ελέγχει αν η διεργασία του Snort λειτουργεί κανονικά και θα στέλνει σε περιοδικά διαστήματα μηνύματα στον κεντρικό κόμβο που υποδηλώνουν την ορθή λειτουργία του Sensor. Αν η διεργασία του Snort πεθάνει τότε θα σταματάει αμέσως και η αποστολή των μηνυμάτων τύπου Heartbeat σκοτώνοντας τη νέα διεργασία που δημιούργησε το plugin.

5.3. Ψηφιακή υπογραφή στα logs του Snort

Η ψηφιακή υπογραφή στα logs του Snort έχει σκοπό να μας εξασφαλίσει την ακεραιότητα τους σε περίπτωση που θελήσουμε να τα χρησιμοποιήσουμε ως ψηφιακά πειστήρια για να αποδείξουμε την επίθεση που δέχτηκε κάποιο τμήμα του δικτύου μας.

Για το σκοπό αυτό παράγουμε ένα ζεύγος RSA κλειδιών μήκους 1024 bits για να το χρησιμοποιήσουμε κατά την υπογραφή των logs του Snort που κρατούνται τοπικά στο δίσκο, αλλά και για την υπογραφή των IDMEF μηνυμάτων που στέλνονται στο κεντρικό κόμβο. Για το σκοπό αυτό χρησιμοποιήθηκε το εργαλείο openssl.

Παραγωγή ιδιωτικού RSA κλειδιού:

```
openssl genrsa -out key.pem -rand filename.seed 1024
```

Παραγωγή δημόσιου RSA κλειδιού:

```
openssl rsa -in key.pem -pubout -out pubkey.pem
```

Τα κλειδιά είναι αποθηκευμένα σε PEM μορφή και φαίνονται παρακάτω:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDELLYF2t2gZS+B/GFY8YnuUdvhIfkvPSwB/i5DN0tkJgYjVhGk
2emck7zgisYm1maCeipsXnVdIW8DsunCby2r8QVhg24W+UCAVHcU2767bFxcRP72
78eE4QXQ5VWgh8qD3LCBGa5e8t0fQ/Z6dAT9txochBUorfUktyp3HsnPAQIDAQAB
AoGASvort6EnA56owa2CJ0ppxURUHX+2lW9z2m8jj10a23QM9IR4yeDfgJCWp9ti
UIWxOD0o6bJQeP/GRkujPETx1xJAe7LIdwuifiUD4TO6zBF4gtsSBBrLkA9hv/TD
1zmGWx1nbi/KWtOFG8tTT1VzI/bMcOjMMAeyRu1wgKRXvWECQQDh2HOJNoB6oQlI
UBjED6KUqPClGS37QRyCr8DbwqbMWIQxH/Jvn8SPP24WecIhzl4em/XeO+7PaYny
S/e636S3AkEA3l4XEiGTLqSZEN+OorjUMyWZYBKZe3OzKZg3N5FhmSErX/1Y3Od7
pxIdQ5KGB3TpkkZPT/H95beGfqP5k5QiBwJBAJbAhw6WRNiWWb3dSUrGLP4Sbt+l
mR1X7pIFgzSuD+zjQAoRiyCbgvcjnfZ+u+kY+ZZxAIOhed0VU7FaqDgjRK0CQHXD
tswvY9dL7JR3JR8Ya6k6vGOY9hffSp7/Yh7Xq6yBWItZjPuSy/XLLhxfOQQx8pE5
Ez/vlNtXuq7PAJob1XkCQBcAEkJ9dCMvbn/HuK7M+cr1xPFVxCNQd56rEVMN3X9G
+BTIVoRBTgqyBnB/J7lkqFzDSHJA4qZo2bBkNat/3Ys=
-----END RSA PRIVATE KEY-----

-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGPIb3DQEBAQUAA4GNABCBiQKBgQDELLYF2t2gZS+B/FGY8YnuUdvh
```

```
ICkvPSwB/i5DNOtkJgYjVhGk2emck7zgisYm1maCeipsXnWdIW8DsunCby2r8QVh
g24W+UCAVHcU2767bFxcRP7273eE4QXP5VWgh8qD3LCBGa5e8tOfQ/Z6dAT9txoc
hBUoifUktyp3HsnPAQIDAQAB
-----END PUBLIC KEY-----
```

Για την επέκταση αυτή στο Snort, μεταβάλλαμε το αρχείο spo_log_ascii.c το οποίο αποτελεί το output plugin που είναι υπεύθυνο για την καταγραφή σε text μορφή των χαρακτηριστικών ενός IP πακέτου.
Συγκεκριμένα, προσθέσαμε στη συνάρτηση LogAsciiInit() τις παρακάτω γραμμές κώδικα:

```
/* Read RSA private key from file */

if((kfp = fopen(args, "r")) == NULL)
{
   FatalError("ERROR: fopen(%s) key file: %s\n",
                args, strerror(errno));
}
else
{
   x = PEM_read_RSAPrivateKey(kfp, &x, NULL, NULL);
   fclose(kfp);

   /* Allocate space for the signature */
   sig = (unsigned char *) malloc(RSA_size(x)*sizeof(unsigned char));
}
```

Όπως φαίνεται και παραπάνω, στο τμήμα αυτό του κώδικα διαβάζουμε το ιδιωτικό κλειδί από το αρχείο και το αποθηκεύουμε σε δομή τύπου RSA. Αυτό γίνεται με τη βοήθεια της συνάρτησης PEM_read_RSAPrivateKey() η οποία επιστρέφει δείκτη σε δομή RSA. Το πρότυπο της συνάρτησης φαίνεται παρακάτω:

```
RSA *PEM_read_RSAPrivateKey(FILE *fp, RSA **x,
                                    pem_password_cb *cb, void *u);
```

Επίσης, προσθέσαμε στη συνάρτηση LogAscii() τις παρακάτω γραμμές κώδικα:

```
/* Compute the signature of the logfile and write it in logfile.sig */

if((lfp = fopen(log_file, "rb")) == NULL)
{
   FatalError("ERROR: fopen(%s) log file: %s\n",
                           log_file, strerror(errno));
}
else
{
   /* Compute the MD5 checksum */

   MD5_Init(&c);
   while ((bytes = fread(buf, sizeof(unsigned char), BUF_SIZE, lfp)))
      MD5_Update(&c, buf, bytes);

   MD5_Final(md, &c);

   fclose(lfp);

   /* Sign Message Digest */

   if (siglen = RSA_private_encrypt(MD5_DIGEST_LENGTH, md, sig, x,
   RSA_PKCS1_PADDING))
   {
      /* Set .sig suffix for the signature file */

      sigfilename = (char *) malloc(sizeof(char) * (strlen(log_file)+5));

      sprintf(sigfilename, "%s.sig", log_file);

      /* Write the signature to file */

      if((sfp = fopen(sigfilename, "wb")) == NULL)
      {
             FatalError("ERROR: fopen(%s) signature file: %s\n",
                     sigfilename, strerror(errno));
      }
      else
      {
             fwrite(sig, sizeof(unsigned char), siglen, sfp);
             fclose(sfp);
      }
      free(sigfilename);
```

```
    }
    else
    {
        FatalError("ERROR: Cannot sign: %s\n", strerror(errno));
    }
}
```

Όπως βλέπουμε στον παραπάνω κώδικα, αρχικά ανοίγουμε το log file για διάβασμα και υπολογίζουμε με διαδοχικά βήματα το MD5 checksum του αρχείου. Τα πρότυπα των συναρτήσεων που χρησιμοποιήσαμε για τον υπολογισμό του MD5 checksum φαίνονται παρακάτω:

```
void MD5_Init(MD5_CTX *c);
void MD5_Update(MD5_CTX *c, const void *data, unsigned long len);
void MD5_Final(unsigned char *md, MD5_CTX *c);
```

Στη συνέχεια, κρυπτογραφούμε με το ιδιωτικό κλειδί το MD5 checksum μήκους 128 bit που προέκυψε προηγουμένως και παράγεται η ψηφιακή υπογραφή, την οποία και αποθηκεύουμε σε αρχείο που έχει το ίδιο όνομα με το log file, προσθέτοντας απλά στο τέλος την κατάληξη .sig. Το πρότυπο της συνάρτησης που χρησιμοποιήσαμε για τη δημιουργία της ψηφιακής υπογραφής φαίνεται παρακάτω:

```
int RSA_private_encrypt(int flen, unsigned char *from,
    unsigned char *to, RSA *rsa,int padding);
```

Εδώ πρέπει να σημειώσουμε ότι για να γίνει σωστά compile το αρχείο προσθέσαμε στη μεταβλητή LIBS του Makefile στον κατάλογο snort/src/ την βιβλιοθήκη crypto που περιέχει τις συναρτήσεις για τον υπολογισμό του MD5 checksum και τις συναρτήσεις που υλοποιούν τις ψηφιακές υπογραφές με χρήση του αλγόριθμου RSA.

Για να διαπιστώσουμε την ορθή λειτουργία των ψηφιακών υπογραφών στα logs του Snort υλοποιήσαμε το πρόγραμμα verify.c (βλέπε Παράρτημα Β) που ελέγχει αν το MD5 checksum του συγκεκριμένου log file είναι ίδιο με αυτό που προκύπτει από την ψηφιακή υπογραφή που υπάρχει για αυτό το αρχείο. Αν τα δύο

MD5 checksums διαφέρουν, τότε το ένα από τα δύο αρχεία (είτε το log file είτε το sig file) έχει τροποποιηθεί από κάποιον τρίτο.

5.4. Δημιουργία του IDMEF output-plugin

Ένα output plugin στο Snort αποτελείται από δύο αρχεία, ένα αρχείο .c και ένα αρχείο .h. Στη δική μας περίπτωση τα αρχεία αυτά έχουν ονόματα spo_alert_idmef.c και spo_alert_idmef.h. Για να λειτουργήσει το plugin θα πρέπει να το δηλώσουμε στη λίστα με τα output plugins που ενεργοποιούνται κατά την αρχικοποίηση του Snort όταν αυτό ξεκινά. Αρχικά, προσθέτουμε στο αρχείο plugbase.c που βρίσκεται στον κατάλογο snort/src/ την παρακάτω γραμμή:

```
#include "output-plugins/spo_alert_idmef.h"
```

Με την γραμμή αυτή ουσιαστικά κάνουμε include στο αρχείο plugbase.c το πρότυπο της συνάρτησης AlertIdmefSetup().

Στη συνέχεια στο ίδιο αρχείο και συγκεκριμένα στη συνάρτηση InitOutputPlugins() στην οποία δηλώνονται τα output plugins που θα χρησιμοποιήσει το Snort προσθέτουμε και την παρακάτω γραμμή:

```
AlertIdmefSetup();
```

Οι κύριες συναρτήσεις από τις οποίες αποτελείται ένα output-plugin στο snort είναι καθορισμένες και κάθε μία επιτελεί μία ξεχωριστή λειτουργία. Για το IDMEF plugin που κατασκευάσαμε, οι κύριες συναρτήσεις φαίνονται παρακάτω:

```
void AlertIdmefSetup()
```

Η συνάρτηση AlertIdmefSetup() κάνει register το plugin στη λίστα με τα output plugins του snort.

```
void AlertIdmefInit(u_char *)
```

Η συνάρτηση AlertIdmefInit() καλεί τη συνάρτηση ParseAlertIdmefArgs() η οποία επεξεργάζεται τις παραμέτρους του plugin και αρχικοποιεί βασικές δομές και μεταβλητές του plugin.

```
SpoAlertIdmefData *ParseAlertIdmefArgs(char *)
```

Η συνάρτηση ParseAlertIdmefArgs() διαβάζει τα arguments του plugin και τα αποθηκεύει στην δομή SpoAlertIdmefData που έχουμε κατασκευάσει.

```
void AlertIdmef(Packet *, char *, void *, Event *)
```

Η συνάρτηση AlertIdmef() είναι το κύριο κομμάτι του plugin και είναι αυτή που καλείται κάθε φορά που προκύπτει κάποιο alert.

```
void AlertIdmefCleanExit(int, void *)
```

Η συνάρτηση AlertIdmefCleanExit() καλείται κατά τον τερματισμό του snort και έχει ως σκοπό να ελευθερώσει τον χώρο μνήμης που είχε δεσμεύσει δυναμικά το plugin.

```
void AlertIdmefRestart(int, void *)
```

Η συνάρτηση AlertIdmefRestart() καλείται κατά την επανεκκίνηση του snort και έχει ως σκοπό να ελευθερώσει τον χώρο μνήμης που είχε δεσμεύσει δυναμικά το plugin.

Εκτός από τις παραπάνω συναρτήσεις που είναι απαραίτητες για την λειτουργία του plugin, κατασκευάσαμε και μία σειρά άλλων βοηθητικών συναρτήσεων οι οποίες συντέλεσαν στην ορθή δόμηση του plugin.

Οι συναρτήσεις αυτές φαίνονται παρακάτω:

```
xmlNodePtr BuildAnalyzer()
```

Η συνάρτηση αυτή κατασκευάζει τον κόμβο Analyzer του IDMEF μηνύματος.

```
xmlNodePtr BuildSource(Packet *p)
```

Η συνάρτηση αυτή κατασκευάζει τον κόμβο Source του IDMEF μηνύματος.

```
xmlNodePtr BuildTarget(Packet *p)
```

Η συνάρτηση αυτή κατασκευάζει τον κόμβο Target του IDMEF μηνύματος.

```
void BuildClassification(char *, xmlNodePtr)
```

Η συνάρτηση αυτή κατασκευάζει τον κόμβο Classification του IDMEF μηνύματος.

```
void SignMessage(char *, char *)
```

Η συνάρτηση αυτή παίρνει σαν πρώτη παράμετρο κάποιο κείμενο και επιστρέφει στην δεύτερη παράμετρο την ψηφιακή του υπογραφή.

```
void HeartBeatProcess()
```

Η συνάρτηση αυτή υλοποιεί την διεργασία που είναι υπεύθυνη για την κατασκευή και αποστολή των heartbeats.

```
int sig_chld()
```

Η συνάρτηση αυτή χειρίζεται το σήμα SIGCHLD που εγείρεται όταν πεθάνει η διεργασία παιδί (στην περίπτωσή μας είναι η διεργασία που υλοποιεί το Heartbeat).

Για τη δημιουργία των IDMEF μηνυμάτων χρησιμοποιήσαμε τη βιβλιοθήκη libidmef. Η βιβλιοθήκη αυτή χρησιμοποιεί τις συναρτήσεις της libxml2 για κατασκευή XML μηνυμάτων.
Οι συναρτήσεις που χρησιμοποιήσαμε από τη συγκεκριμένη βιβλιοθήκη φαίνονται αναλυτικά παρακάτω:

```
Όνομα συνάρτησης: newIDMEF_Message
Πρότυπο: xmlNodePtr newIDMEF_Message(xmlNodePtr, ...);
Παράμετροι:

    ATTRIBUTES
    xmlNodePtr version: προαιρετικό
```

SUB-ELEMENTS
xmlNodePtr Alert   : μηδέν ή περισσότερα
xmlNodePtr Heartbeat: μηδέν ή περισσότερα

Επιστρέφει ένα δείκτη xmlNodePtr στο root element

Όνομα συνάρτησης: newAlert
Πρότυπο: xmlNodePtr newAlert(xmlNodePtr, ...);
Παράμετροι:

ATTRIBUTES
xmlNodePtr ident : προαιρετικό

SUB-ELEMENTS
xmlNodePtr Analyzer   : ακριβώς ένα
xmlNodePtr CreateTime   : ακριβώς ένα
xmlNodePtr Source   : μηδέν ή περισσότερα
xmlNodePtr Target   : μηδέν ή περισσότερα
xmlNodePtr Classification : ένα ή περισσότερα
xmlNodePtr AdditionalData : μηδέν ή περισσότερα

Επιστρέφει ένα δείκτη xmlNodePtr στο Alert node που δημιουργήθηκε.

Όνομα συνάρτησης: newHeartbeat
Πρότυπο: xmlNodePtr newHeartbeat(xmlNodePtr, ...);
Παράμετροι:

ATTRIBUTES
xmlNodePtr ident: προαιρετικό

SUB-ELEMENTS
xmlNodePtr Analyzer   : ακριβώς ένα
xmlNodePtr CreateTime   : ακριβώς ένα
xmlNodePtr AdditionalData : μηδέν ή περισσότερα

Επιστρέφει ένα δείκτη xmlNodePtr στο Heartbeat node που δημιουργήθηκε.

Όνομα συνάρτησης: newAnalyzer
Πρότυπο: xmlNodePtr newAnalyzer(xmlNodePtr, ...);
Παράμετροι:

ATTRIBUTES

xmlNodePtr analyzerid : προαιρετικό

SUB-ELEMENTS
xmlNodePtr Node   : zero or one

Επιστρέφει ένα δείκτη xmlNodePtr στο Analyzer node που δημιουργήθηκε.

Όνομα συνάρτησης: newClassification
Πρότυπο: xmlNodePtr newClassification(xmlNodePtr, ...);
Παράμετροι:

ATTRIBUTES
xmlNodePtr origin: υποχρεωτικό

SUB-ELEMENTS
xmlNodePtr name: ακριβώς ένα
xmlNodePtr url : ακριβώς ένα

Επιστρέφει ένα δείκτη xmlNodePtr στο Classification node που δημιουργήθηκε.

Όνομα συνάρτησης: newSource
Πρότυπο: xmlNodePtr newSource(xmlNodePtr, ...);
Παράμετροι:

ATTRIBUTES
xmlNodePtr ident : προαιρετικό
xmlNodePtr spoofed: προαιρετικό

SUB-ELEMENTS
xmlNodePtr Node   : μηδέν ή ένα
xmlNodePtr Service: μηδέν ή ένα

Επιστρέφει ένα δείκτη xmlNodePtr στο Source node που δημιουργήθηκε.

Όνομα συνάρτησης: newTarget
Πρότυπο: xmlNodePtr newTarget(xmlNodePtr, ...);
Παράμετροι:

ATTRIBUTES
xmlNodePtr ident: προαιρετικό
xmlNodePtr decoy: προαιρετικό

SUB-ELEMENTS

```
    xmlNodePtr Node    : μηδέν ή ένα
    xmlNodePtr Service : μηδέν ή ένα

Επιστρέφει ένα δείκτη xmlNodePtr στο Target node που δημιουργήθηκε.


Όνομα συνάρτησης: newCreateTime
Πρότυπο: xmlNodePtr newCreateTime(struct timeval *);
Παράμετροι:

    struct timeval *

Επιστρέφει ένα δείκτη xmlNodePtr στο CreateTime node που δημιουργήθηκε.


Όνομα συνάρτησης: newNode
Πρότυπο: xmlNodePtr newNode(xmlNodePtr, ...);
Παράμετροι:

    ATTRIBUTES
    xmlNodePtr ident   : προαιρετικό
    xmlNodePtr category: προαιρετικό

    SUB-ELEMENTS
    xmlNodePtr name    : μηδέν ή ένα
    xmlNodePtr Address : μηδέν ή περισσότερα

Επιστρέφει ένα δείκτη xmlNodePtr στο Node node που δημιουργήθηκε.


Όνομα συνάρτησης: newAddress
Πρότυπο: xmlNodePtr newAddress(xmlNodePtr, ...);
Παράμετροι:

    ATTRIBUTES
    xmlNodePtr ident    : προαιρετικό
    xmlNodePtr category : προαιρετικό

    SUB-ELEMENTS
    xmlNodePtr address: exactly one
    xmlNodePtr netmask: zero or one

 Επιστρέφει ένα δείκτη xmlNodePtr στο Address node που δημιουργήθηκε.


Όνομα συνάρτησης: newService
```

Πρότυπο: xmlNodePtr newService(xmlNodePtr, ...);
Παράμετροι:

    ATTRIBUTES
    xmlNodePtr ident: προαιρετικό

    SUB-ELEMENTS
    xmlNodePtr port : μηδέν ή ένα

 Επιστρέφει ένα δείκτη xmlNodePtr στο Service node που δημιουργήθηκε.

Όνομα συνάρτησης: newSimpleElement
Πρότυπο: xmlNodePtr newSimpleElement(char *, char *);
Παράμετροι:

   char *name : το όνομα του νέου στοιχείου
   char *value: η τιμή του νέου στοιχείου

Επιστρέφει ένα δείκτη xmlNodePtr στο νέο κόμβο που δημιουργήθηκε.

Όνομα συνάρτησης: newAttribute
Πρότυπο: xmlNodePtr newAttribute(char *, char *);
Παράμετροι:

    char *name : το όνομα του γνωρίσματος
    char *value: η τιμή του γνωρίσματος

Επιστρέφει ένα δείκτη xmlNodePtr στο νέο κόμβο που δημιουργήθηκε.

Όνομα συνάρτησης: addElement
Πρότυπο: xmlNodePtr addElement(xmlNodePtr, xmlNodePtr);
Παράμετροι:

     xmlNodePtr parent: δείκτης στον κόμβο-γονέα
     xmlNodePtr child : δείκτης στον κόμβο-παιδί

Επιστρέφει δείκτη στο κόμβο παιδί που δημιούργησε

Όνομα συνάρτησης: setAttribute
Πρότυπο: xmlNodePtr setAttribute(xmlNodePtr, xmlNodePtr);
Παράμετροι:
      xmlNodePtr element : δείκτης xmlNodePtr στο στοιχείο

```
      xmlNodePtr attrib  : δείκτης xmlNodePtr στο γνώρισμα

Επιστρέφει ένα δείκτη στο στοιχείο, αφού ορίσει το νέο γνώρισμα


Όνομα συνάρτησης: globalsInit
Πρότυπο: int globalsInit(char *);
Παράμετροι:

       char * :  Το DTD που περιγράφει το XML κείμενο

Αρχικοποιεί τις μεταβλητές και τις δομές που χρησιμοποιούνται για την
κατασκευή των IDMEF messages. Επιστρέφει 1 όταν επιτύχει, διαφορετικά 0.


Όνομα συνάρτησης: createCurrentDoc
Πρότυπο: int createCurrentDoc(const char *);
Παράμετροι:

       char * :  Η έκδοση της γλώσσας XML

Δημιουργεί ένα καινούριο XML Document και το θέτει ως το τρέχον. Επιστρέφει
1 όταν επιτύχει, διαφορετικά 0.


Όνομα συνάρτησης: validateCurrentDoc
Πρότυπο: int validateCurrentDoc();

Κάνει validate το XML Document που δημιουργήθηκε με το DTD που του έχουμε
ορίσει. Επιστρέφει 1 όταν επιτύχει, διαφορετικά 0.


Όνομα συνάρτησης: clearCurrentDoc
Πρότυπο: void clearCurrentDoc();

Ελευθερώνει το χώρο του XML Document μαζί με όλες τις δομές που αυτό
κατείχε.
```

Αφού δημιουργήσουμε το IDMEF μήνυμα, κατασκευάζουμε την ψηφιακή του υπογραφή, μετατρέπουμε την υπογραφή στη δεκαεξαδική της μορφή, την ενσωματώνουμε στο μήνυμα δημιουργώντας έτσι ένα Signed-IDMEF Message και το στέλνουμε στο multicast socket που έχουμε δημιουργήσει.

Για τη δημιουργία της ψηφιακής υπογραφής ακολουθείται η εξής διαδικασία:
Αρχικά υπολογίζεται το MD5 checksum του μηνύματος με τη βοήθεια της παρακάτω συνάρτησης:

```
unsigned char *MD5(const unsigned char *d, unsigned long n,
                   unsigned char *md);
```

Στη συνέχεια υπογράφουμε το MD5 checksum με τη βοήθεια της συνάρτησης RSA_sign(), το πρότυπο της οποίας φαίνεται παρακάτω:

```
int RSA_sign(int type, unsigned char *m, unsigned int m_len,
    unsigned char *sigret, unsigned int *siglen, RSA *rsa);
```

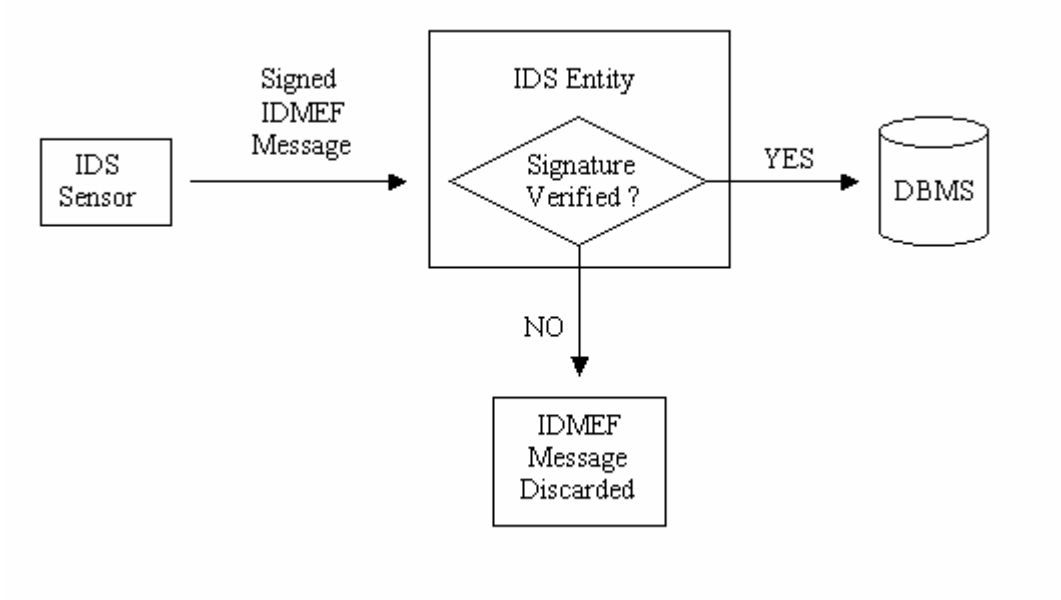
Εδώ πρέπει να σημειώσουμε ότι για να γίνει σωστά compile το αρχείο προσθέσαμε στη μεταβλητή LIBS του Makefile στον κατάλογο snort/src/ τη βιβλιοθήκη idmef, που περιέχει τις συναρτήσεις για την κατασκευή των IDMEF μηνυμάτων καθώς και τη βιβλιοθήκη xml2 που απαιτείται από την libidmef.

5.5. Σύνδεση με οντότητα IDS

Για την υλοποίηση του κεντρικού κόμβου στον οποίο στέλνουν τα μηνύματα οι Sensors στηριχθήκαμε στην Οντότητα IDS (IDS Entity) που έχει υλοποιηθεί στα πλαίσια της διπλωματικής εργασίας του Βασίλη Χατζηγιαννάκη. Για να μπορέσει να συνεργαστεί η Οντότητα IDS με τους IDS Sensors του δικτύου μας, επεκτείναμε το XML Parsing module της οντότητας ώστε να υποστηρίζει επαλήθευση της ψηφιακής υπογραφής των μηνυμάτων.

Πιο συγκεκριμένα, έχουμε τοποθετήσει όλα τα δημόσια κλειδιά στην Οντότητα IDS και κάθε φορά που φτάνει ένα Signed-IDMEF-Message στην Οντότητα IDS, αφού δούμε από το γνώρισμα analyzerid από ποιον Sensor προέρχεται, επιλέγουμε το αντίστοιχο δημόσιο κλειδί για να επαληθεύσουμε την υπογραφή που φέρει το μήνυμα. Αν η επαλήθευση επιτύχει το μήνυμα καταχωρείται στη βάση δεδομένων. Σε αντίθετη περίπτωση το μήνυμα σημαίνει ότι έχει υπογραφεί με διαφορετικό ιδιωτικό κλειδί από αυτό που έχει ο Sensor από τον οποίο προέρχεται ή έχει αλλαχτεί το περιεχόμενο του μηνύματος κατά την διαδρομή του προς τον

κεντρικό κόμβο. Στην περίπτωση αυτή το μήνυμα αγνοείται, ενώ ταυτόχρονα ειδοποιείται ο διαχειριστής της Οντότητας IDS με μήνυμα στην κονσόλα.
Σχηματικά η διαδικασία που περιγράψαμε φαίνεται παρακάτω:



Η γλώσσα που χρησιμοποιήθηκε για την υλοποίηση της επαλήθευσης της ψηφιακής υπογραφής είναι η Java, αφού ολόκληρη η εφαρμογή της Οντότητας IDS είναι γραμμένη στη γλώσσα Java. Πιο συγκεκριμένα έγινε επέκταση του αρχείου Xml2Doc.java το οποίο παρατίθεται ολόκληρο στο Παράρτημα Β.

6. Εγκατάσταση και χρήση του συστήματος

Σε αυτό το κεφάλαιο της διπλωματικής εργασίας, εξηγούμε πως θα εγκαταστήσουμε το τροποποιημένο Snort σε ένα IDS Sensor. Η εγκατάσταση και λειτουργία του Snort όπως το επεκτείναμε στα πλαίσια αυτής της διπλωματικής εργασίας έχει πραγματοποιηθεί με επιτυχία στα παρακάτω λειτουργικά συστήματα:

- Linux 2.4.18
- Solaris 9
- FreeBSD 4.7
- OpenBSD 3.1

Αρχικά κάνουμε download τον πηγαίο κώδικα του Snort. Η έκδοση του Snort στην οποία εργαστήκαμε είναι η 1.9.1. Έτσι από τη διεύθυνση http://www.snort.org/dl/snort-1.9.1.tar.gz κατεβάζουμε τον πηγαίο κώδικα σε συμπιεσμένη μορφή. Για την επέκταση του Snort έχουμε χρησιμοποιήσει την βιβλιοθήκη libidmef την οποία και κάνουμε download από την διεύθυνση: http://www.silicondefense.com/idwg/libidmef/libidmef-0.7.2.tar.gz

Ξεκινάμε την εγκατάσταση της βιβλιοθήκης libidmef:

```
tar –zxvf libidmef-0.7.2.tar.gz
cd libidmef-0.7.2
./configure
make
make install
```

Η βιβλιοθήκη libidmef απαιτεί την ύπαρξη της βιβλιοθήκης libxml2 στο σύστημά μας. Αν δεν την διαθέτουμε μπορούμε να την κατεβάσουμε από τη διεύθυνση: http://www.xmlsoft.org/.

Αφού ολοκληρωθεί η εγκατάσταση της βιβλιοθήκης libidmef μπορούμε να συνεχίσουμε με την εγκατάσταση του Snort.

```
tar –zxvf snort-1.9.1.tar.gz
cd snort-1.9.1
./configure
```

Σε αυτό το σημείο πριν κάνουμε compile θα πρέπει να αντικαταστήσουμε τα αρχεία που αλλάξαμε με την καινούρια έκδοσή τους, καθώς και να βάλουμε στα κατάλληλα directories και τα νέα αρχεία που δημιουργήσαμε. Για αυτό το σκοπό έχουμε δημιουργήσει ένα συμπιεσμένο αρχείο με όνομα snort-idmef.tar.gz το οποίο περιέχει τα αρχεία αυτά ώστε να διευκολύνουμε την εγκατάσταση.
Το συμπιεσμένο αρχείο περιέχει τα εξής:

```
src/output-plugins/
src/output-plugins/spo_log_ascii.c
src/output-plugins/spo_alert_idmef.c
src/output-plugins/spo_alert_idmef.h
src/preprocessors/
src/preprocessors/spp_portscan.c
src/detect.c
src/plugbase.c
```

Μέσα στο directory του Snort κάνουμε extract το αρχείο αυτό.

```
tar –zxvf snort-idmef.tar.gz
```

Αφού εγκατασταθούν τα αρχεία στα κατάλληλα σημεία, πρέπει να αλλάξουμε τα Makefiles πριν το compilation. Στο directory src/output-plugins στο αρχείο Makefile, αλλάζουμε την μεταβλητή libspo_a_SOURCES ώστε να περιλαμβάνει και τα αρχεία spo_alert_idmef.c, spo_alert_idmef.h που δημιουργήσαμε. Επίσης, προσθέτουμε στη μεταβλητή libspo_a_OBJECTS το αρχείο spo_alert_idmef.o και στη μεταβλητή INCLUDES το εξής: -I/usr/local/ssl/include `xml2-config --cflags`
Το /usr/local/ssl/include είναι το directory των include αρχείων της βιβλιοθήκης libcrypto (σε κάθε λειτουργικό σύστημα το directory αυτό είναι διαφορετικό). Το xml2-config είναι το πρόγραμμα που βρίσκει και προσθέτει αυτόματα το path των include αρχείων για την βιβλιοθήκη libxml2.
Στο directory src/, στο αρχείο Makefile προσθέτουμε στη μεταβλητή LIBS τα εξής:

```
-lcrypto -lidmef -lm -lxml2
```

Ειδικά για το OpenBSD, χρειάζεται να προσθέσουμε και το –lz (χρήση της βιβλιοθήκης libz).

Στη συνέχεια είμαστε έτοιμοι να κάνουμε compile:

```
make
make install
```

Το Snort είναι πλέον εγκατεστημένο στο σύστημά μας. Τώρα δημιουργούμε το ζεύγος των RSA κλειδιών με την βοήθεια του εργαλείου openssl όπως περιγράψαμε στο προηγούμενο κεφάλαιο και τα τοποθετούμε στο directory keys που δημιουργούμε μέσα στο directory του Snort. Στη συνέχεια πηγαίνουμε στο directory etc/ του Snort και προσθέτουμε τις παρακάτω γραμμές στο αρχείο snort.conf.

```
output log_ascii: /home/gandr/snort/keys/key.pem
output alert_fast: /var/log/snort/alert

ruletype send-idmef
{
  type alert
  output alert_fast: /var/log/snort/alert
  output alert_idmef: address=230.0.0.1 port=4446 ttl=1 analyzerid=sensor1
key=/home/gandr/snort/keys/key.pem dtd=/home/gandr/snort/idmef.dtd
interval=300
}

config order: send-idmef alert pass log
```

Όπως βλέπουμε παραπάνω έχουμε υποθέσει ότι το Snort είναι εγκατεστημένο στο directory /home/gandr/snort/. Ορίζουμε ένα καινούριο ruletype το οποίο θα είναι αυτό που εκτός από την απλή καταγραφή των alerts σε αρχείο θα στέλνει και ένα IDMEF μήνυμα στον κεντρικό κόμβο κάθε φορά που ανιχνεύεται κάποια επίθεση. Το plugin που δημιουργήσαμε μπορεί να διαμορφωθεί από τις επτά παραμέτρους που εμφανίζονται παρακάτω:

1. Την IP multicast address
2. Το multicast port
3. Το Time-To-Live του πακέτου το οποίο εξαρτάται από την δικτυακή απόσταση του Sensor από το Central IDS Node.
4. Το όνομα του Sensor με το οποίο θα είναι γνωστός στο κατανεμημένο δίκτυο μας.

5. Το ιδιωτικό κλειδί με το οποίο υπογράφουμε τα logs και τα IDMEF μηνύματα.
6. Το DTD με το οποίο κάνουμε validate τα IDMEF μηνύματα που δημιουργούμε.
7. Η συχνότητα με την οποία ο συγκεκριμένος Sensor θα στέλνει Heartbeat μηνύματα στον κεντρικό κόμβο.

Η τελευταία γραμμή που προσθέσαμε στο αρχείο, προσδιορίζει την σειρά με την οποία γίνονται trigger οι κανόνες (rules) του Snort. Ορίζουμε στο configuration του Snort να είναι ενεργοποιημένοι πρώτα οι κανόνες του τύπου send-idmef που δημιουργήσαμε.

Στη συνέχεια, πηγαίνουμε στο directory rules/ του Snort όπου βρίσκονται τα αρχεία με τους κανόνες και ορίζουμε τον τύπο send-idmef στους κανόνες για τους οποίους επιθυμούμε να στέλνονται μηνύματα IDMEF στον κεντρικό κόμβο. Τώρα είμαστε πλέον έτοιμοι να ξεκινήσουμε το Snort. Οι παράμετροι με τους οποίους τρέχουμε το Snort φαίνονται παρακάτω:

```
snort -c /home/gandr/snort/etc/snort.conf -D -i hme0
```

Η παράμετρος –c ορίζει το configuration file που θα διαβάσει το Snort κατά την εκκίνηση του, η παράμετρος –D χρειάζεται για να ξεκινήσει το Snort σε daemon mode, ενώ η παράμετρος –i ορίζει το interface το οποίο θα ακούει το Snort.

Παράρτημα Α : Βιβλιογραφία

[1] Andrew S. Tanenbaum, “Δίκτυα Υπολογιστών”, Τρίτη Έκδοση,
ISBN: 960-7510-70-4

[2] Biswanath Mukherjee, L.Todd Heberlein and Karl N. Levitt, “Network Intrusion Detection”, IEEE Network - May/June 1994

[3] Bruce Schneier, “Applied Cryptography: Protocols, Algorithms and Source Code in C”, John Wiley & Sons, ISBN: 0-471-11709-9

[4] D. Curry and H. Debar, "Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition", Internet Draft, November 2002.

[5] Libidmef library - http://www.silicondefense.com/idwg/libidmef/

[6] Marcelo Medina, “A Layered Framework for Placement of Distributed Intrusion Detection Devices”, 21st National Information Systems Security Conference, 1998

[7] Openssl project - http://www.openssl.org

[8] Q. Zhang and R. Janakiraman, “Indra: A Distributed Approach to Network Intrusion Detection and Prevention”, Washington University Technical Report # WUCS-01-30, 2001.

[9] R.L.Rivest, A.Shamir and L.M.Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, Communications of the ACM, Feb 1978

[10] R.Rivest, “The MD5 Message-Digest Algorithm”, RFC 1321, Apr 1992

[11] Snort project - http://www.snort.org

[12] Stephen Northcutt, Judy Nocvac, “Network Intrusion Detection An Analyst’s Handbook, Second Edition”, ISBN: 0-7357-1008-2

[13] W. Richard Stevens, “TCP/IP Illustrated, Volume 1 - The protocols”, Addison Wesley, ISBN: 0-201-63346-9

[14] W. Richard Stevens, “UNIX Network Programming, Volume 1 – Networking APIs: Sockets and XTI”, Prentice Hall, ISBN: 0-13-490012-X

[15] Yarochkin Fyodor, “Snortnet – A Distributed Intrusion Detection System”, Kyrgyz Russian Slavin University, Bishkek, Kyrgyzstan, June 2000.

[16] Βασίλης Χατζηγιαννάκης, “Υλοποίηση συστήματος επικοινωνίας και διαχείρισης για κατανεμημένο δίκτυο αντιμετώπισης επιθέσεων απάρνησης υπηρεσίας”, Διπλωματική εργασία, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών ΕΜΠ, Οκτώβριος 2002.

Παράρτημα Β : Κώδικας

Λίστα αρχείων:

1. spo_log_ascii.c
2. verify.c
3. spo_alert_idmef.h
4. spo_alert_idmef.c
5. Xml2Doc.java
6. idmef.dtd

Αρχείο spo_log_ascii.c

```
/*
** Copyright (C) 1998-2002 Martin Roesch <roesch@sourcefire.com>
**           (C) 2002 Sourcefire, Inc.
**           (C) 2003 Androulidakis Georgios <gandr@netmode.ntua.gr>
**
** Author(s):   Martin Roesch <roesch@sourcefire.com>
**              Andrew R. Baker <andrewb@sourcefire.com>
**              Androulidakis Georgios <gandr@netmode.ntua.gr>
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
** GNU General Public License for more details.
**
** You should have received a copy of the GNU General Public License
** along with this program; if not, write to the Free Software
** Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307,
USA.
*/
/* $Id: */

/* spo_log_ascii
 *
 * Purpose:
 *
 * This output module provides the default packet logging funtionality
 *
 * Arguments:
 *
 * None.
 *
 * Effect:
 *
 * None.
 *
 * Comments:
 *
 */

#ifdef HAVE_CONFIG_H
#include "config.h"
#endif
#include <sys/types.h>
#include <string.h>
#ifdef HAVE_STRINGS_H
#include <strings.h>
#endif
#include <errno.h>
#include <sys/stat.h>
#ifndef WIN32
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#endif /* ! WIN32 */

#include "plugbase.h"
#include "parser.h"
#include "debug.h"
#include "decode.h"
#include "event.h"
#include "log.h"
```

```
#include "util.h"

#include "snort.h"

/* includes for signing log files */
#include <openssl/md5.h>
#include <openssl/pem.h>
#include <openssl/rsa.h>

#define BUF_SIZE 1024            /* How many bytes to read in each
MD5_Update */

FILE *lfp, *sfp, *kfp;
unsigned char buf[BUF_SIZE], md[MD5_DIGEST_LENGTH], *sig;
MD5_CTX c;
RSA *x;                          /* RSA Private key structure */
int siglen, bytes;
char *sigfilename;
char log_file[STD_BUF+1]; /* name of log file (moved up) */

/* external globals from rules.c */
extern char *file_name;
extern int file_line;
extern OptTreeNode *otn_tmp;

/* internal functions */
void LogAsciiInit(u_char *args);
void LogAscii(Packet *p, char *msg, void *arg, Event *event);
void LogAsciiCleanExit(int signal, void *arg);
void LogAsciiRestart(int signal, void *arg);
char *IcmpFileName(Packet * p);
int OpenLogFile(int mode, Packet * p);


#define DUMP              1
#define BOGUS             2
#define NON_IP            3
#define ARP               4
#define GENERIC_LOG   5

/* XXX fix this */
static FILE *log_ptr;

void LogAsciiSetup()
{
    /* link the preprocessor keyword to the init function in
       the preproc list */
    RegisterOutputPlugin("log_ascii", NT_OUTPUT_LOG, LogAsciiInit);

    DEBUG_WRAP(DebugMessage(DEBUG_PLUGIN, "Output: LogAscii is setup\n"););
}

void LogAsciiInit(u_char *args)
{
    DEBUG_WRAP(DebugMessage(DEBUG_PLUGIN, "Output: Ascii logging
initialized\n"););

    pv.log_plugin_active = 1;

    /* Read RSA private key from file */

    if((kfp = fopen(args, "r")) == NULL)
    {
      FatalError("ERROR: fopen(%s) key file: %s\n",
                       args, strerror(errno));
    }
    else
    {
```

```
       x = PEM_read_RSAPrivateKey(kfp, &x, NULL, NULL);
       fclose(kfp);

       /* Allocate space for the signature */
       sig = (unsigned char *) malloc(RSA_size(x)*sizeof(unsigned char));
    }

    /* Set the preprocessor function into the function list */
    AddFuncToOutputList(LogAscii, NT_OUTPUT_LOG, NULL);
    AddFuncToCleanExitList(LogAsciiCleanExit, NULL);
    AddFuncToRestartList(LogAsciiRestart, NULL);
}



void LogAscii(Packet *p, char *msg, void *arg, Event *event)
{
    DEBUG_WRAP(DebugMessage(DEBUG_LOG, "LogPkt started\n"););
    if(p)
    {
        if(p->iph)
            OpenLogFile(0, p);
        else if(p->ah)
            OpenLogFile(ARP, p);
        else
            OpenLogFile(NON_IP, p);
    }
    else
        OpenLogFile(GENERIC_LOG, p);

    if(msg)
    {
        fwrite("[**] ", 5, 1, log_ptr);
        fwrite(msg, strlen(msg), 1, log_ptr);
        fwrite(" [**]\n", 6, 1, log_ptr);
    }
    if(p)
    {
        if(p->iph)
            PrintIPPkt(log_ptr, p->iph->ip_proto, p);
        else if(p->ah)
            PrintArpHeader(log_ptr, p);
    }
    fclose(log_ptr);

    /* Compute the signature of the logfile and write it in logfile.sig */

    if((lfp = fopen(log_file, "rb")) == NULL)
    {
       FatalError("ERROR: fopen(%s) log file: %s\n",
                        log_file, strerror(errno));
    }
    else
    {

       /* Compute the MD5 checksum */

       MD5_Init(&c);
       while ((bytes = fread(buf, sizeof(unsigned char), BUF_SIZE, lfp)))
         MD5_Update(&c, buf, bytes);

       MD5_Final(md, &c);

       fclose(lfp);

       /* Sign Message Digest */
```

```
       if (siglen = RSA_private_encrypt(MD5_DIGEST_LENGTH, md, sig, x,
RSA_PKCS1_PADDING))
       {
          /* Set .sig suffix for the signature file */

          sigfilename = (char *) malloc(sizeof(char) * (strlen(log_file)+5));

          sprintf(sigfilename, "%s.sig", log_file);

          /* Write the signature to file */

          if((sfp = fopen(sigfilename, "wb")) == NULL)
          {
              FatalError("ERROR: fopen(%s) signature file: %s\n",
                        sigfilename, strerror(errno));
          }
          else
          {
              fwrite(sig, sizeof(unsigned char), siglen, sfp);
              fclose(sfp);
          }
          free(sigfilename);

       }
       else
       {
           FatalError("ERROR: Cannot sign: %s\n", strerror(errno));
       }
   }
}


void LogAsciiCleanExit(int signal, void *arg)
{
    free(sig);
    return;
}

void LogAsciiRestart(int signal, void *arg)
{
    free(sig);
    return;
}

static char *logfile[] =
        { "", "PACKET_FRAG", "PACKET_BOGUS", "PACKET_NONIP", "ARP", "log" };

/*
 * Function: OpenLogFile()
 *
 * Purpose: Create the log directory and file to put the packet log into.
 *          This function sucks, I've got to find a better way to do this
 *          this stuff.
 *
 * Arguments: None.
 *
 * Returns: 0 on success, exits on error
 */
int OpenLogFile(int mode, Packet * p)
{
    char log_path[STD_BUF+1]; /* path to log file */
    char proto[5];      /* logged packet protocol */
    char suffix[5];     /* filename suffix */
#ifdef WIN32
    strcpy(suffix,".ids");
#else
    suffix[0] = '\0';
#endif
```

```
    /* zero out our buffers */
    bzero((char *) log_path, STD_BUF + 1);
    bzero((char *) log_file, STD_BUF + 1);
    bzero((char *) proto, 5);

    if (mode == GENERIC_LOG || mode == DUMP || mode == BOGUS ||
            mode == NON_IP || mode == ARP)
    {
        snprintf(log_file, STD_BUF, "%s%s/%s",
                chrootdir == NULL ? "" : chrootdir, pv.log_dir,
logfile[mode]);

        if((log_ptr = fopen(log_file, "a")) == NULL)
        {
            FatalError("ERROR: OpenLogFile() => fopen(%s) log file: %s\n",
                    log_file, strerror(errno));
        }
        return 0;
    }

    if(otn_tmp != NULL)
    {
        if(otn_tmp->logto != NULL)
        {
            snprintf(log_file, STD_BUF, "%s%s/%s",
                    chrootdir == NULL ? "" : chrootdir,  pv.log_dir,
                                             otn_tmp->logto);

            if((log_ptr = fopen(log_file, "a")) == NULL)
            {
                FatalError("ERROR: OpenLogFile() => fopen(%s) log file:
%s\n",
                           log_file, strerror(errno));
            }
            return 0;
        }
    }
    /* figure out which way this packet is headed in relation to the homenet
*/
    if((p->iph->ip_dst.s_addr & pv.netmask) == pv.homenet)
    {
        if((p->iph->ip_src.s_addr & pv.netmask) != pv.homenet)
        {
            snprintf(log_path, STD_BUF, "%s%s/%s",
                    chrootdir == NULL ? "" : chrootdir, pv.log_dir,
                                             inet_ntoa(p->iph->ip_src));
        }
        else
        {
            if(p->sp >= p->dp)
            {
                snprintf(log_path, STD_BUF, "%s%s/%s",
                        chrootdir == NULL ? "" : chrootdir,  pv.log_dir,
                                                 inet_ntoa(p->iph->ip_src));
            }
            else
            {
                snprintf(log_path, STD_BUF, "%s%s/%s",
                        chrootdir == NULL ? "" : chrootdir, pv.log_dir,
                                                 inet_ntoa(p->iph->ip_dst));
            }
        }
    }
    else
    {
        if((p->iph->ip_src.s_addr & pv.netmask) == pv.homenet)
        {
```

```
            snprintf(log_path, STD_BUF, "%s%s/%s",
                    chrootdir == NULL ? "" : chrootdir,  pv.log_dir,
                                            inet_ntoa(p->iph->ip_dst));
        }
        else
        {
            if(p->sp >= p->dp)
            {
                snprintf(log_path, STD_BUF, "%s%s/%s",
                        chrootdir == NULL ? "" : chrootdir, pv.log_dir,
                                            inet_ntoa(p->iph->ip_src));
            }
            else
            {
                snprintf(log_path, STD_BUF, "%s%s/%s",
                        chrootdir == NULL ? "" : chrootdir, pv.log_dir,
                                            inet_ntoa(p->iph->ip_dst));
            }
        }
    }

    DEBUG_WRAP(DebugMessage(DEBUG_FLOW, "Creating directory: %s\n",
log_path););

    /* build the log directory */
    if(mkdir(log_path, S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH))
    {

        if(errno != EEXIST)
        {
            FatalError("ERROR: OpenLogFile() => mkdir(%s) log directory:
%s\n",
                    log_path, strerror(errno));
        }
    }

    DEBUG_WRAP(DebugMessage(DEBUG_FLOW, "Directory Created!\n"););

    /* build the log filename */
    if(p->iph->ip_proto == IPPROTO_TCP ||
            p->iph->ip_proto == IPPROTO_UDP)
    {
        if(p->frag_flag)
        {
            snprintf(log_file, STD_BUF, "%s/IP_FRAG%s", log_path, suffix);
        }
        else
        {
            if(p->sp >= p->dp)
            {
#ifdef WIN32
                snprintf(log_file, STD_BUF, "%s/%s_%d-%d%s", log_path,
                        protocol_names[p->iph->ip_proto], p->sp, p->dp,
suffix);
#else
                snprintf(log_file, STD_BUF, "%s/%s:%d-%d%s", log_path,
                        protocol_names[p->iph->ip_proto], p->sp, p->dp,
suffix);
#endif
            }
            else
            {
#ifdef WIN32
                snprintf(log_file, STD_BUF, "%s/%s_%d-%d%s", log_path,
                        protocol_names[p->iph->ip_proto], p->dp, p->sp,
suffix);
#else
                snprintf(log_file, STD_BUF, "%s/%s:%d-%d%s", log_path,
```

```
                        protocol_names[p->iph->ip_proto], p->dp, p->sp,
suffix);
#endif
            }
        }
    }
    else
    {
        if(p->frag_flag)
        {
            snprintf(log_file, STD_BUF, "%s/IP_FRAG%s", log_path, suffix);
        }
        else
        {
            if(p->iph->ip_proto == IPPROTO_ICMP)
            {
                snprintf(log_file, STD_BUF, "%s/%s_%s%s", log_path, "ICMP",
                         IcmpFileName(p), suffix);
            }
            else
            {
                snprintf(log_file, STD_BUF, "%s/PROTO%d%s", log_path,
                         p->iph->ip_proto, suffix);
            }
        }
    }

    DEBUG_WRAP(DebugMessage(DEBUG_FLOW, "Opening file: %s\n", log_file););

    /* finally open the log file */
    if((log_ptr = fopen(log_file, "a")) == NULL)
    {
        FatalError("ERROR: OpenLogFile() => fopen(%s) log file: %s\n",
                log_file, strerror(errno));
    }

    DEBUG_WRAP(DebugMessage(DEBUG_FLOW, "File opened...\n"););

    return 0;
}



/****************************************************************************
*
 *
 * Function: IcmpFileName(Packet *p)
 *
 * Purpose: Set the filename of an ICMP output log according to its type
 *
 * Arguments: p => Packet data struct
 *
 * Returns: the name of the file to set
 *

****************************************************************************/
char *IcmpFileName(Packet * p)
{
    if(p->icmph == NULL)
    {
        return "ICMP_TRUNC";
    }

    switch(p->icmph->type)
    {
        case ICMP_ECHOREPLY:
            return "ECHO_REPLY";
```

```
case ICMP_DEST_UNREACH:
    switch(p->icmph->code)
    {
        case ICMP_NET_UNREACH:
            return "NET_UNRCH";

        case ICMP_HOST_UNREACH:
            return "HST_UNRCH";

        case ICMP_PROT_UNREACH:
            return "PROTO_UNRCH";

        case ICMP_PORT_UNREACH:
            return "PORT_UNRCH";

        case ICMP_FRAG_NEEDED:
            return "UNRCH_FRAG_NEEDED";

        case ICMP_SR_FAILED:
            return "UNRCH_SOURCE_ROUTE_FAILED";

        case ICMP_NET_UNKNOWN:
            return "UNRCH_NETWORK_UNKNOWN";

        case ICMP_HOST_UNKNOWN:
            return "UNRCH_HOST_UNKNOWN";

        case ICMP_HOST_ISOLATED:
            return "UNRCH_HOST_ISOLATED";

        case ICMP_PKT_FILTERED_NET:
            return "UNRCH_PKT_FILTERED_NET";

        case ICMP_PKT_FILTERED_HOST:
            return "UNRCH_PKT_FILTERED_HOST";

        case ICMP_NET_UNR_TOS:
            return "UNRCH_NET_UNR_TOS";

        case ICMP_HOST_UNR_TOS:
            return "UNRCH_HOST_UNR_TOS";

        case ICMP_PKT_FILTERED:
            return "UNRCH_PACKET_FILT";

        case ICMP_PREC_VIOLATION:
            return "UNRCH_PREC_VIOL";

        case ICMP_PREC_CUTOFF:
            return "UNRCH_PREC_CUTOFF";

        default:
            return "UNKNOWN";

    }

case ICMP_SOURCE_QUENCH:
    return "SRC_QUENCH";

case ICMP_REDIRECT:
    return "REDIRECT";

case ICMP_ECHO:
    return "ECHO";

case ICMP_TIME_EXCEEDED:
    return "TTL_EXCEED";
```

```
            case ICMP_PARAMETERPROB:
                return "PARAM_PROB";

            case ICMP_TIMESTAMP:
                return "TIMESTAMP";

            case ICMP_TIMESTAMPREPLY:
                return "TIMESTAMP_RPL";

            case ICMP_INFO_REQUEST:
                return "INFO_REQ";

            case ICMP_INFO_REPLY:
                return "INFO_RPL";

            case ICMP_ADDRESS:
                return "ADDR";

            case ICMP_ADDRESSREPLY:
                return "ADDR_RPL";

            default:
                return "UNKNOWN";
        }
    }
```

Αρχείο verify.c

```
/*
** Copyright (C) 2003 Androulidakis Georgios <gandr@netmode.ntua.gr>
**
** Author(s):   Androulidakis Georgios <gandr@netmode.ntua.gr>
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
** GNU General Public License for more details.
**
** You should have received a copy of the GNU General Public License
** along with this program; if not, write to the Free Software
** Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307,
USA.
*/

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<openssl/md5.h>
#include<openssl/pem.h>
#include<openssl/rsa.h>
#include<openssl/err.h>

#define BUF_SIZE 1024
#define SIG_LENGTH 128
#define KEYFILE "/home/gandr/snort/keys/pubkey.pem"

FILE *fp, *kfp, *sfp;
unsigned char buf[BUF_SIZE], md[MD5_DIGEST_LENGTH],
mdsig[MD5_DIGEST_LENGTH], sig[SIG_LENGTH];
MD5_CTX c;
RSA *x;
int i, bytes;
unsigned long e;
char *s, *sigfilename;

int main(int argc, char **argv)
{

       if (argc != 2)
       {
              printf("Usage: %s <filename>\n", argv[0]);
              exit(0);
       }

       if ((fp = fopen(argv[1], "rb")) == NULL)
       {
              printf("Cannot open file %s\n", argv[1]);
              exit(0);
       }

       MD5_Init(&c);
       while ((bytes = fread(buf, sizeof(unsigned char), BUF_SIZE, fp)))
         MD5_Update(&c, buf, bytes);
       MD5_Final(md, &c);
       fclose(fp);

       printf("MD5 checksum from file: ");
       for (i = 0; i < MD5_DIGEST_LENGTH; i++)
         printf("%02x", md[i]);
```

```
	sigfilename = (char *) malloc(sizeof(char) * (strlen(argv[1])+5));
	sprintf(sigfilename, "%s.sig", argv[1]);

	if ((sfp = fopen(sigfilename, "rb")) == NULL)
	{
		printf("Cannot open sig file %s\n", sigfilename);
		exit(0);
	}
	bytes = fread(sig, sizeof(unsigned char), SIG_LENGTH, sfp);
	fclose(sfp);

	if ((kfp = fopen(KEYFILE, "r")) == NULL)
	{
		printf("Cannot open key file %s\n", KEYFILE);
		exit(0);
	}

	x = PEM_read_RSA_PUBKEY(kfp, NULL, NULL, NULL);
	fclose(kfp);

	if ((i = RSA_public_decrypt(SIG_LENGTH, sig, mdsig, x,
RSA_PKCS1_PADDING)) > 0)
	{
		printf("\nMD5 checksum from sig : ");

		for (i=0; i < MD5_DIGEST_LENGTH; i++)
		  printf("%02x", mdsig[i]);
	}
	else
	{
		 e = ERR_get_error();
		 s = ERR_error_string(e, NULL);
		 printf("ERROR: %s\n", s);
	}

	return 0;
}
```

Αρχείο spo_alert_idmef.h

```
/*
** Copyright (C) 2003 Androulidakis Georgios <gandr@netmode.ntua.gr>
**
** Author(s):   Androulidakis Georgios <gandr@netmode.ntua.gr>
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
** GNU General Public License for more details.
**
** You should have received a copy of the GNU General Public License
** along with this program; if not, write to the Free Software
** Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307,
USA.
*/

#ifndef __SPO_ALERT_IDMEF_H__
#define __SPO_ALERT_IDMEF_H__


void AlertIdmefSetup();


#endif  /* __SPO_ALERT_IDMEF_H__ */
```

Αρχείο spo_alert_idmef.c

```
/*
** Copyright (C) 2003 Androulidakis Georgios <gandr@netmode.ntua.gr>
**
** Author(s):   Androulidakis Georgios <gandr@netmode.ntua.gr>
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
** GNU General Public License for more details.
**
** You should have received a copy of the GNU General Public License
** along with this program; if not, write to the Free Software
** Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307,
USA.
*/

/* spo_alert_idmef
 *
 * Purpose:  output plugin for idmef alerting
 *
 * Effect:
 *
 * Alerts are sent to the multicast group defined by the args using IDMEF
 *
 */

#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <signal.h>
#include <fcntl.h>

#include "decode.h"
#include "event.h"
#include "rules.h"
#include "plugbase.h"
#include "parser.h"
#include "debug.h"
#include "util.h"
#include "generators.h"
#include "signature.h"
#include "snort.h"

#include <libidmef/idmefxml.h>
#include <openssl/md5.h>
#include <openssl/pem.h>
#include <openssl/rsa.h>
```

```
#define XML_VERSION           "1.0"
#define IDMEF_VERSION         "1.0"
#define MAX_MESSAGE_SIZE    65536
#define CHECK_INTERVAL        10

#define IDMEF_LOG

/*
 * Includes from other sources:  spp_portscan.c
 */

/* Definitions for scan types */
struct spp_timeval
{
    time_t tv_sec;
    time_t tv_usec;
};

typedef enum _scanType
{
    sNONE = 0, sUDP = 1, sSYN = 2, sSYNFIN = 4, sFIN = 8, sNULL = 16,
    sXMAS = 32, sFULLXMAS = 64, sRESERVEDBITS = 128, sVECNA = 256, sNOACK =
512,
 sNMAPID = 1024,
    sSPAU = 2048, sINVALIDACK = 4096
} ScanType;

/* Structures for keeping track of connection information. */
typedef struct _connectionInfo
{
    ScanType scanType;
    u_short sport;
    u_short dport;
    struct spp_timeval timestamp;
    char tcpFlags[9];       /* Eight flags and a NULL */
    u_char *packetData;
    struct _connectionInfo *prevNode;
    struct _connectionInfo *nextNode;
}              ConnectionInfo;

typedef struct _destinationInfo
{
    struct in_addr daddr;
    int numberOfConnections;
    ConnectionInfo *connectionsList;
    struct _destinationInfo *prevNode;
    struct _destinationInfo *nextNode;
}               DestinationInfo;

typedef struct _sourceInfo
{
    struct in_addr saddr;
    int numberOfConnections;
    int numberOfDestinations;
    int numberOfTCPConnections;
    int numberOfUDPConnections;
    int totalNumberOfTCPConnections;
    int totalNumberOfUDPConnections;
    int totalNumberOfDestinations;

    struct spp_timeval firstPacketTime;
    struct spp_timeval lastPacketTime;
    int reportStealth;

    int stealthScanUsed;
    int scanDetected;
    struct spp_timeval reportTime;  /* last time we reported on this
                     * source's activities */
```

```
    DestinationInfo *destinationsList;
    u_int32_t event_id;
    struct _sourceInfo *prevNode;
    struct _sourceInfo *nextNode;
}           SourceInfo;



typedef struct _SpoAlertIdmefData
{
             char *address;
             char *port;
             char *ttl;
             char *analyzerid;
             char *key;
             char *dtd;
             char *interval;

} SpoAlertIdmefData;

void AlertIdmefInit(u_char *);
void AlertIdmef(Packet *, char *, void *, Event *);
SpoAlertIdmefData *ParseAlertIdmefArgs(char *);
void AlertIdmefCleanExit(int, void *);
void AlertIdmefRestart(int, void *);

xmlNodePtr BuildAnalyzer();
xmlNodePtr BuildSource(Packet *p);
xmlNodePtr BuildTarget(Packet *p);
void BuildClassification(char *, xmlNodePtr);
void SignMessage(char *, char *);
void HeartBeatProcess();
int sig_chld();

/* external globals from rules.c */
extern char *file_name;
extern int file_line;
extern OptTreeNode *otn_tmp;
extern SourceInfo *livePortScanSource;

FILE *fp;
SpoAlertIdmefData *data;
struct hostent *h;
struct sockaddr_in localAddr;
struct sockaddr_in remoteAddr;
FILE *kfp;
RSA *x;       /* RSA Private Key */
unsigned char *sig;
int pi[2];    /* pipe */
int sd;              /* socket */
unsigned char ttl = 1;  /* TTL value for multicast packet */
int pid;
int idmef_enabled = 0; /* Flag used by preprocessors */

/*
 * Function: SetupAlertIdmef()
 *
 * Purpose: Registers the output plugin keyword and initialization
 *          function into the output plugin list.  This is the function that
 *          gets called from InitOutputPlugins() in plugbase.c.
 *
 * Arguments: None.
 *
 * Returns: void function
 *
 */
void AlertIdmefSetup()
{
```

```
    /* link the preprocessor keyword to the init function in
       the preproc list */
    RegisterOutputPlugin("alert_idmef", NT_OUTPUT_ALERT, AlertIdmefInit);

    DEBUG_WRAP(DebugMessage(DEBUG_PLUGIN, "Output plugin: AlertIdmef is
setup...\n"););
}


/*
 * Function: AlertIdmefInit(u_char *)
 *
 * Purpose: Calls the argument parsing function, performs final setup on
data
 *          structs, links the preproc function into the function list.
 *
 * Arguments: args => ptr to argument string
 *
 * Returns: void function
 *
 */
void AlertIdmefInit(u_char *args)
{

    DEBUG_WRAP(DebugMessage(DEBUG_PLUGIN, "Output: AlertIdmef
Initialized\n"););

    pv.alert_plugin_active = 1;

    idmef_enabled = 1;

    /* parse the argument list from the rules file */
    data = ParseAlertIdmefArgs(args);

    /* Build remote address structure */

    h = gethostbyname(data->address);

    if (h == NULL)
    {
       FatalError("Unknown host: %s\n", data->address);
    }

    remoteAddr.sin_family = h->h_addrtype;
    memcpy((char *) &remoteAddr.sin_addr.s_addr, h->h_addr_list[0],
              h->h_length);
    remoteAddr.sin_port = htons(atoi(data->port));

    /* Check if remote address is multicast */

    if (!IN_MULTICAST(ntohl(remoteAddr.sin_addr.s_addr)))
    {
       FatalError("Given Address %s is not multicast\n",
inet_ntoa(remoteAddr.sin_addr));
    }

    /* Set TTL for multicast packet */

    if (data->ttl != NULL)
       ttl = (unsigned char) atoi(data->ttl);

    /* Read RSA private key from file */

    if((kfp = fopen(data->key, "r")) == NULL)
    {
      FatalError("ERROR: fopen(%s) key file: %s\n",
                      data->key, strerror(errno));
    }
```

```
    else
    {
      x = PEM_read_RSAPrivateKey(kfp, &x, NULL, NULL);
      fclose(kfp);

      /* Allocate space for the signature */
      sig = (unsigned char *) malloc(RSA_size(x)*sizeof(unsigned char));
    }

    /* Initialize the global variables */
    globalsInit(data->dtd);

    /* Creation of HeartBeat Process */

    if (pipe(pi) < 0)
       FatalError("Cannot create pipe\n");

    if ((pid = fork()) == -1)
    {
       FatalError("Cannot create new process\n");
    }
    else if (pid == 0) /* Heartbeat process */
    {
       HeartBeatProcess();
    }
    else /* parent process */
    {
       signal(SIGCHLD, sig_chld);
       close(pi[0]);

       /* Build local address structure */

       localAddr.sin_family = AF_INET;
       localAddr.sin_addr.s_addr = htonl(INADDR_ANY);
       localAddr.sin_port = htons(0);

       /* Create socket */

       if ((sd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
              FatalError("Cannot open socket\n");

       /* Bind socket */

       if (bind(sd, (struct sockaddr *) &localAddr, sizeof(localAddr)) < 0)
              FatalError("Cannot bind socket\n");

       /* Set TTL */

       if (setsockopt(sd, IPPROTO_IP, IP_MULTICAST_TTL, &ttl, sizeof(ttl)) <
0)
              FatalError("Cannot set TTL\n");


       DEBUG_WRAP(DebugMessage(DEBUG_PLUGIN, "Linking IdmefAlert functions to
call lists...\n"););

       /* Set the preprocessor function into the function list */
       AddFuncToOutputList(AlertIdmef, NT_OUTPUT_ALERT, data);
       AddFuncToCleanExitList(AlertIdmefCleanExit, data);
       AddFuncToRestartList(AlertIdmefRestart, data);

   }

}


/*
 * Function: ParseAlertIdmefArgs(char *)
```

```
 *
 * Purpose: Process the preprocessor arguements from the rules file and
 *          initialize the preprocessor's data struct.  This function
doesn't
 *          have to exist if it makes sense to parse the args in the init
 *          function.
 *
 * Arguments: args => argument list
 *
 * Returns: void function
 *
 */

SpoAlertIdmefData *ParseAlertIdmefArgs(char *args)
{
    SpoAlertIdmefData *data;
    char *xarg, *tmp;

    DEBUG_WRAP(DebugMessage(DEBUG_PLUGIN, "ParseAlertIdmefArgs: %s\n",
args););

    if (args == NULL)
    {
       FatalError("You must supply arguments for alert_idmef plugin\n");
    }

    data = (SpoAlertIdmefData *) malloc(sizeof(SpoAlertIdmefData));

    xarg = strtok(args," =");

    while (xarg != NULL)
    {
      tmp = strtok(NULL, " ");

       if(!strncasecmp(xarg,"address",7))
       {
              data->address = tmp;
       }
       if(!strncasecmp(xarg,"port",4))
       {
              data->port = tmp;
       }
       if(!strncasecmp(xarg,"ttl",3))
       {
              data->ttl = tmp;
       }
       if(!strncasecmp(xarg,"analyzerid",10))
       {
              data->analyzerid = tmp;
       }
       if(!strncasecmp(xarg,"key",3))
       {
              data->key = tmp;
       }
       if(!strncasecmp(xarg,"dtd",3))
       {
              data->dtd = tmp;
       }
       if(!strncasecmp(xarg,"interval",8))
       {
              data->interval = tmp;
       }

       xarg = strtok(NULL, " =");
   }

   return data;
}
```

```
/****************************************************************************
*
 *
 * Function: AlertIdmef(Packet *, char *)
 *
 * Purpose: Send the current alert using IDMEF
 *
 * Arguments: p => pointer to the packet data struct
 *            msg => the message to print in the alert
 *
 * Returns: void function
 *

****************************************************************************/
void AlertIdmef(Packet *p, char *msg, void *arg, Event *event)
{
       xmlNodePtr theMessage, theAlert, theAnalyzer, theSource, theTarget,
theClassification;
       xmlDocPtr doc = 0;
       char *text=0, *tmsg=0, *smsg=0, *s;
       int text_length = 0, sl=0;


#ifdef IDMEF_LOG
  fp = fopen("/var/log/snort/idmef.log", "a");
#endif

       /* Check args before building the IDMEF Message */

       if ((msg == NULL) || (event == NULL))
              return;

       if ((p == NULL) && (event->sig_generator != GENERATOR_SPP_PORTSCAN))
              return;

       /* Create the current document structure, passing in the current
version of XML (1.0)  */

       if (!createCurrentDoc(XML_VERSION))
       {
              FatalError("createCurrnetDoc failed\n");
       }

       /* Create the Analyzer Node */

       if((theAnalyzer = BuildAnalyzer()) == NULL)
       {
              FatalError("Problem building Analyzer node\n");
       }

       theAlert =
              newAlert(
                     theAnalyzer,
                     newCreateTime(NULL),
                     NULL
               );


       /* Build Portscan Message */

       if (event->sig_generator == GENERATOR_SPP_PORTSCAN)
       {

              /* Create the Source Node */

              theSource =
                  newSource(
```

```
                        newAttribute("spoofed","unknown"),
                        newNode(
                                newAddress(
                                newAttribute("category","ipv4-addr"),
                                newSimpleElement("address",
inet_ntoa(livePortScanSource->saddr)),
                                NULL
                                ),
                        NULL
                        ),
                NULL
                );

                addElement(theAlert, theSource);


                /* Add Targets */
                {
                        DestinationInfo *d = livePortScanSource-
>destinationsList;

                        while (d)
                        {
                                /* Create the Target Node */
                                theTarget =
                                   newTarget(
                                       newAttribute("spoofed","unknown"),
                                       newNode(
                                          newAddress(
                                          newAttribute("category","ipv4-addr"),
                                          newSimpleElement("address", inet_ntoa(d-
>daddr)),
                                          NULL
                                          ),
                                     NULL
                                     ),
                                  NULL
                                 );

                                addElement(theAlert, theTarget);

                                d = d->nextNode;
                        }
                }

                /* Create the Classification Node */

                theClassification =
                        newClassification(
                                newAttribute("origin","vendor-specific"),
                                newSimpleElement("name",msg),
                                newSimpleElement("url","No URL available"),
                                NULL
                        );

                addElement(theAlert, theClassification);

        }
        else /* Simple Packet */
        {

                /* Create the Source Node */

                if((theSource = BuildSource(p)) == NULL)
                {
                        FatalError("Problem building Source node\n");
                }
                else
```

```
		{
			addElement(theAlert, theSource);
		}


		/* Create the Target Node */

		if((theTarget = BuildTarget(p)) == NULL)
		{
			FatalError("Problem building Target node\n");
		}
		else
		{
			addElement(theAlert, theTarget);
		}


		/* Create the Classification Node(s) */

		BuildClassification(msg, theAlert);

	}

	/* Now build the IDMEF Message */

	theMessage = newIDMEF_Message(
		newAttribute("version", IDMEF_VERSION),
		theAlert,
		NULL
		);

	if(theMessage == NULL)
	{
		FatalError("Problem building IDMEF-Message node\n");
	}

	/* IDMEF Message is ready */

	if (!validateCurrentDoc())
		FatalError("IDMEF XML Message does not match DTD\n");

	doc = getCurrentDoc();
	xmlDocDumpMemory(doc, (xmlChar**)&text, &text_length);

	/* Grab only the message */
	tmsg = strstr(text, "<IDMEF-Message");

	/* Build the signed Message */

	smsg = (char *) malloc(MAX_MESSAGE_SIZE * sizeof(char));

	sprintf(smsg, "%s", "<Signed-IDMEF-Message>");
	strcat(smsg, tmsg);
	strcat(smsg, "<Signature analyzerid=\"");
	strcat(smsg, data->analyzerid);
	strcat(smsg, "\">");

	/* Add the signature */
	sl = 2 * RSA_size(x)+ 1;
	s = (char *) malloc(sl * sizeof(char));

	SignMessage(tmsg, s);
	strcat(smsg, s);

	strcat(smsg, "</Signature>");
	strcat(smsg, "</Signed-IDMEF-Message>\n");

#ifdef IDMEF_LOG
```

```
	fprintf(fp, "%s\n", smsg);
	fflush(fp);
	fclose(fp);
#endif

	/* Send IDMEF Message */

	if (sendto(sd, smsg, strlen(smsg), 0, (struct sockaddr *) &remoteAddr,
sizeof(remoteAddr)) < 0)
	{
		close(sd);
		FatalError("Cannot send IDMEF Message");
	}

	/* Cleanup */
	clearCurrentDoc();
	free(s);
	free(smsg);
}


/*
 * Function: BuildAnalyzer()
 *
 * Purpose: Constructs the Analyzer node for the IDMEF message.
 *
 * Returns: An xmlNodePtr to the Analyzer node.
 */

xmlNodePtr BuildAnalyzer()
{
	char hostname[256];
	int hostlen = 256;
	struct hostent *h;
	xmlNodePtr theAnalyzer, theNode, theAddress;

	theAnalyzer = newAnalyzer(
		newAttribute("analyzerid",data->analyzerid),
		NULL
		);

	theNode = newNode(NULL);
	theAddress = newAddress(
			newAttribute("category","ipv4-addr"),
			NULL
		       );

	if (gethostname(hostname, hostlen))
		FatalError("Cannot get hostname");

	h = gethostbyname(hostname);

	if (strlen(h->h_name) > 0)
		addElement(theNode,newSimpleElement("name", h->h_name));
	else
		addElement(theNode,newSimpleElement("name", hostname));

	addElement(theAddress, newSimpleElement("address",
GetIP(pv.interfaces[0])));

	if(theAddress->children != NULL)
		addElement(theNode, theAddress);
	else
		xmlFreeNode(theAddress);

	if(theNode->children != NULL)
		addElement(theAnalyzer, theNode);
	else
```

```
              xmlFreeNode(theNode);

       return theAnalyzer;

}


/*
 * Function: BuildSource(Packet *)
 *
 * Purpose: Constructs the Source node for the IDMEF message.
 *
 * Arguments: p => The current packet
 *
 * Returns: An xmlNodePtr to the Source node.
 */

xmlNodePtr BuildSource(Packet *p)
{
       xmlNodePtr theSource, theAddress, theNode;

       theSource = newSource(newAttribute("spoofed", "unknown"), NULL);

       theNode = newNode(NULL);

       theAddress =  newAddress(
                    newAttribute("category","ipv4-addr"),
                    newSimpleElement("address",inet_ntoa(p->iph->ip_src)),
                    NULL
             );

       addElement(theNode,theAddress);

       addElement(theSource,theNode);

       /* We need to add a Service element to include port information if
this was a TCP or UDP packet */

       if((p->iph->ip_proto == IPPROTO_TCP) || (p->iph->ip_proto ==
IPPROTO_UDP))
       {
              char sp[6];

              sprintf(sp, "%d", p->sp);

              addElement(theSource,
                 newService(
                      newSimpleElement("port", sp),
                      NULL)
                 );
       }

       return theSource;
}



/*
 * Function: BuildTarget(Packet *)
 *
 * Purpose: Constructs the Target node for the IDMEF message.
 *
 * Arguments: p => The current packet
 *
 * Returns: An xmlNodePtr to the Target node.
 */

xmlNodePtr BuildTarget(Packet *p)
```

```
{
       xmlNodePtr theTarget, theAddress, theNode;

       theTarget = newTarget(
              newAttribute("decoy", "unknown"),
              NULL
              );

       theNode = newNode(NULL);

       theAddress =  newAddress(
                    newAttribute("category","ipv4-addr"),
                    newSimpleElement("address",inet_ntoa(p->iph->ip_dst)),
                    NULL
             );


       addElement(theNode,theAddress);

       addElement(theTarget,theNode);

       /* We need to add a Service element to include port information if
this was a TCP or UDP packet */

       if((p->iph->ip_proto == IPPROTO_TCP) || (p->iph->ip_proto ==
IPPROTO_UDP))
       {
              char dp[6];

              sprintf(dp, "%d", p->dp);

              addElement(theTarget,
                 newService(
                      newSimpleElement("port", dp),
                      NULL)
                 );
       }

       return theTarget;
}


/*
 * Function: BuildClassification(char *, xmlNodePtr)
 *
 * Purpose: Constructs the Classification node(s) for the IDMEF message.
 *
 * Arguments: msg => The message to print in the alert
 *
 * Returns: Nothing.
 */

void BuildClassification(char *msg, xmlNodePtr theAlert)
{
       xmlNodePtr theClassification;
       ReferenceNode *refNode;
       char url[256];

       if ((otn_tmp == NULL) || ((refNode = otn_tmp->sigInfo.refs) == NULL))
       {
              theClassification =
                     newClassification(
                            newAttribute("origin","vendor-specific"),
                            newSimpleElement("name",msg),
                            newSimpleElement("url","No URL available"),
                            NULL
                     );
```

```
                addElement(theAlert, theClassification);

        }
        else
        {
                while(refNode)
                {
                        if (!strcasecmp(refNode->system->name, "bugtraq"))
                        {
                                sprintf(url,
        "http://www.securityfocus.com/bid/%s", refNode->id);

                                theClassification =
                                        newClassification(
                                                newAttribute("origin","bugtraqid"),
                                                newSimpleElement("name",msg),
                                                newSimpleElement("url",url),
                                                NULL
                                        );

                                addElement(theAlert, theClassification);
                        }
                        else if (!strcasecmp(refNode->system->name, "cve"))
                        {
                                sprintf(url, "http://cve.mitre.org/cgi-
bin/cvename.cgi?name=%s", refNode->id);
                                theClassification =
                                        newClassification(
                                                newAttribute("origin","cve"),
                                                newSimpleElement("name",msg),
                                                newSimpleElement("url",url),
                                                NULL
                                        );

                                addElement(theAlert, theClassification);
                        }
                        else if (!strcasecmp(refNode->system->name,
"arachnids"))
                        {
                                sprintf(url,
        "http://www.whitehats.com/info/IDS%s", refNode->id);
                                theClassification =
                                        newClassification(
                                                newAttribute("origin","vendor-
specific"),
                                                newSimpleElement("name",msg),
                                                newSimpleElement("url",url),
                                                NULL
                                        );

                                addElement(theAlert, theClassification);
                        }
                        else if (!strcasecmp(refNode->system->name, "MCAFEE"))
                        {
                                sprintf(url,
        "http://vil.nai.com/vil/content/v_%s", refNode->id);
                                theClassification =
                                        newClassification(
                                                newAttribute("origin","vendor-
specific"),
                                                newSimpleElement("name",msg),
                                                newSimpleElement("url",url),
                                                NULL
                                        );

                                addElement(theAlert, theClassification);
                        }
                        else if (!strcasecmp(refNode->system->name, "nessus"))
```

```
                        {
                                sprintf(url,
        "http://cgi.nessus.org/plugins/dump.php3?id=%s", refNode->id);
                                theClassification =
                                        newClassification(
                                                newAttribute("origin","vendor-
specific"),
                                                newSimpleElement("name",msg),
                                                newSimpleElement("url",url),
                                                NULL
                                        );

                                addElement(theAlert, theClassification);
                        }
                        else if (!strcasecmp(refNode->system->name, "url"))
                        {
                                sprintf(url, "http://%s", refNode->id);
                                theClassification =
                                        newClassification(
                                                newAttribute("origin","vendor-
specific"),
                                                newSimpleElement("name",msg),
                                                newSimpleElement("url",url),
                                                NULL
                                        );

                                addElement(theAlert, theClassification);
                        }

                        refNode = refNode->next;
                }
        }

}


/*
 * Function: SignMessage(char *, char *)
 *
 * Purpose: Signs the message and returns the signaturee.
 *
 * Arguments: msg  => The message to sign
 *                                      sign => The signature
 *
 * Returns: Nothing.
 */

void SignMessage(char *msg, char *sign)
{
  unsigned char md[MD5_DIGEST_LENGTH];
  int i, sighexsize;
  char *hexsig = 0;
       unsigned int siglen;


  /* Compute the MD5 checksum */

  MD5(msg, strlen(msg)-1, md);

  /* Sign Message Digest */

  if (RSA_sign(NID_md5, md, MD5_DIGEST_LENGTH, sig, &siglen, x))
  {
     char hn[3] = {0};

     sighexsize = 2 * siglen * sizeof(char) + 1;
     hexsig = (char *) malloc(sighexsize * sizeof(char));
     hexsig[0] = '\0';
```

```
      for (i=0; i < siglen; i++)
      {
        snprintf(hn, 3, "%02x", sig[i]);
        strcat(hexsig, hn);
      }

      /* Return the hex-coded signature */
      strncpy(sign, hexsig, sighexsize);

      free(hexsig);

    }
    else
    {
      FatalError("Cannot sign: %s\n", strerror(errno));
    }

}


/*
 * Function: HeartBeatProcess()
 *
 * Purpose: Creation of the HeartBeat IDMEF Message
 *
 * Arguments: None
 *
 * Returns: void function
 *
 */

void HeartBeatProcess()
{
       xmlDocPtr doc = 0;
       xmlNodePtr theMessage, theAnalyzer;
       char *text = 0, *tmsg = 0, *smsg = 0, *s = 0;
       int sl = 0, text_length = 0, i = 0;
       char buf;

       close(pi[1]);

       /* Build local address structure */

       localAddr.sin_family = AF_INET;
       localAddr.sin_addr.s_addr = htonl(INADDR_ANY);
       localAddr.sin_port = htons(0);

       /* Create socket */

       if ((sd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
              FatalError("Cannot open socket\n");

       /* Bind socket */

       if (bind(sd, (struct sockaddr *) &localAddr, sizeof(localAddr)) < 0)
              FatalError("Cannot bind socket\n");

       /* Set TTL */

       if (setsockopt(sd, IPPROTO_IP, IP_MULTICAST_TTL, &ttl, sizeof(ttl)) <
0)
              FatalError("Cannot set TTL\n");


       /* Make the following read()'s in non-blocking mode */
       fcntl(pi[0], F_SETFL, fcntl(pi[0], F_GETFL) | O_NONBLOCK);
```

```
        /* Check if snort process has died */
        if (read(pi[0], &buf, 1) == 0)
        {
                close(pi[0]);
                FatalError("Snort process died !!!\n");
        }

        while(1)
        {

#ifdef IDMEF_LOG
                fp = fopen("/var/log/snort/idmef.log", "a");
#endif

                /* Create the current document structure, passing in the
current version of XML (1.0)  */

                if (!createCurrentDoc(XML_VERSION))
                {
                        FatalError("createCurrnetDoc failed\n");
                }

                /* Create the Analyzer Node */

                if((theAnalyzer = BuildAnalyzer()) == NULL)
                {
                        FatalError("Problem building Analyzer node\n");
                }

                /* Now build the IDMEF Message */

                theMessage = newIDMEF_Message(
                        newAttribute("version", IDMEF_VERSION),
                        newHeartbeat(
                                theAnalyzer,
                                newCreateTime(NULL),
                                NULL
                                ),
                        NULL
                        );

                if(theMessage == NULL)
                {
                        FatalError("Problem building IDMEF-Message node\n");
                }


                if (!validateCurrentDoc())
                        FatalError("IDMEF XML Message does not match DTD\n");

                text = NULL;
                doc = getCurrentDoc();
                xmlDocDumpMemory(doc, (xmlChar**)&text, &text_length);

                /* Grab only the message */
                tmsg = strstr(text, "<IDMEF-Message");

                /* Build the signed Message */

                smsg = (char *) malloc(MAX_MESSAGE_SIZE * sizeof(char));

                sprintf(smsg, "%s", "<Signed-IDMEF-Message>");
                strcat(smsg, tmsg);
                strcat(smsg, "<Signature analyzerid=\"");
                strcat(smsg, data->analyzerid);
                strcat(smsg, "\">");

                /* Add the signature */
```

```
              sl = 2 * RSA_size(x)+ 1;
              s = (char *) malloc(sl * sizeof(char));

              SignMessage(tmsg, s);
              strcat(smsg, s);

              strcat(smsg, "</Signature>");
              strcat(smsg, "</Signed-IDMEF-Message>\n");

#ifdef IDMEF_LOG
              fprintf(fp, "%s\n", smsg);
              fflush(fp);
              fclose(fp);
#endif

              /* Send IDMEF Message */

              if (sendto(sd, smsg, strlen(smsg), 0, (struct sockaddr *)
&remoteAddr, sizeof(remoteAddr)) < 0)
              {
                     close(sd);
                     FatalError("Cannot send IDMEF Message");
              }

              /* Cleanup */
              clearCurrentDoc();
              free(s);
              free(smsg);

              for (i=0; i< (atoi(data->interval) / CHECK_INTERVAL); i++)
              {
                     /* Check every X seconds if snort process has died */
                     if (read(pi[0], &buf, 1) == 0)
                     {
                            close(pi[0]);
                            close(sd);
                            FatalError("Snort process died! Terminating
HeartBeat process...\n");
                     }
                     sleep(CHECK_INTERVAL);
              }

       }

}


/*
 * Function: sig_chld()
 *
 * Purpose: Handling of the signal SIGCHLD produced when child process dies.
 *
 * Arguments: None
 *
 * Returns: Nothing.
 *
 */

int sig_chld()
{
       close(pi[1]);
       close(sd);
       FatalError("HeartBeat process Died! Terminating Snort process...\n");
       exit(0);
}


void AlertIdmefCleanExit(int signal, void *arg)
```

```
{
    SpoAlertIdmefData *data = (SpoAlertIdmefData *)arg;

    DEBUG_WRAP(DebugMessage(DEBUG_PLUGIN, "SpoAlertIdmefCleanExitFunc\n"););

    free(sig);
    free(data);
}

void AlertIdmefRestart(int signal, void *arg)
{
    SpoAlertIdmefData *data = (SpoAlertIdmefData *)arg;

    DEBUG_WRAP(DebugMessage(DEBUG_PLUGIN, "SpoAlertIdmefRestartFunc\n"););

    free(sig);
    free(data);
}
```

Αρχείο Xml2Doc.java

```
/**    Implements the class needed
         for parsing XML documents.
       @author  Vasilis Hatzigiannakis
       @author  Georgios Androulidakis

       */

package Ids;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.text.ParsePosition;
import java.text.SimpleDateFormat;
import java.security.*;
import java.security.spec.*;
import org.apache.soap.encoding.Hex;

import org.w3c.dom.Node;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Element;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.xml.sax.InputSource ;
import org.w3c.dom.NamedNodeMap;



public class Xml2Doc {

  private Alert alert=null;
  private Heartbeat heartbeat=null;

  Alert doc2Alert(Document d)
  {
      NodeList rootList = d.getChildNodes();
      Node rootNode = rootList.item(0);

      Node curNode;
      NodeList curList;


      String ident="NAN";
      String dateString="NAN";
      String timeString="NAN";
      int dateInt=0;
      int timeInt=0;


      String analyzerIdent=null;
      String analyzerNodeIdent=null;
      String analyzerNodeCategory=null;
      String analyzerNodeLocation=null;
      String analyzerNodeName=null;
      String analyzerNodeAddressIdent=null;
      String analyzerNodeAddress=null;
      String entityIdent=null;
      String entityNodeIdent=null;
      String entityNodeCategory=null;
      String entityNodeLocation=null;
      String entityNodeName=null;
      String entityNodeAddressIdent=null;
```

```
      String entityNodeAddress=null;

      String[] sourceIdent=null;
      String[] sourceSpoofed=null;
      String[] sourceInterface=null;
      String[] sourceNodeIdent=null;
      String[] sourceNodeAddressIdent=null;
      String[] sourceNodeAddress=null;
      String[] targetIdent=null;
      String[] targetDecoy=null;
      String[] targetInterface=null;
      int[] targetPort=null;
      String[] targetNodeIdent=null;
      String[] targetNodeCategory=null;
      String[] targetNodeAddressIdent=null;
      String[] targetNodeAddress=null;

      String[] classification=null;
      String[] adData = null;

      NamedNodeMap map = rootNode.getAttributes();
      if (map.getLength()>0)
      {
        curNode = map.item(0);
        ident = curNode.getNodeValue();
       System.out.println("Alert ident:"+ident);
      }
      NodeList alertList = rootNode.getChildNodes();


      //get createTime
       curList = d.getElementsByTagName("CreateTime");
       curNode = curList.item(0);
       curList = curNode.getChildNodes();
       curNode = curList.item(0);
       dateString = curNode.getNodeValue();
       try
       {
              SimpleDateFormat formatter = new SimpleDateFormat ("yyyy.MM.dd
G 'at' hh:mm:ss a zzz");
              ParsePosition pos = new ParsePosition(0);
              Date currTime = formatter.parse(dateString, pos);
              System.out.println("CreateTime: " + dateString );
              Calendar cal = formatter.getCalendar();
              int nYear=cal.get( Calendar.YEAR );
              int nMonth=cal.get( Calendar.MONTH )+1;
              int nDay=cal.get( Calendar.DAY_OF_MONTH );

              dateInt = nDay +100*nMonth+ 10000*nYear;
              dateString = Integer.toString(nYear)+"-
"+Integer.toString(nMonth)+"-"+Integer.toString(nDay);
              System.out.println("Date: "+dateString);

              int nHour=cal.get( Calendar.HOUR_OF_DAY );
              int nMin=cal.get( Calendar.MINUTE );
              int nSec=cal.get( Calendar.SECOND );

              timeInt = nSec + 100*nMin + 10000*nHour;
              timeString =
Integer.toString(nHour)+":"+Integer.toString(nMin)+":"+Integer.toString(nSec
);
              System.out.println("Time: "+timeString);

        }
        catch(Exception e)
        {
              System.out.println(e.getMessage());
        }
```

```
       //get Entity
       curList = d.getElementsByTagName("Entity");
        int Esize = curList.getLength();
       String id="";
       String type="";
       if(Esize>0)
       {
              NodeList entityList = curList;

              Node entityNode = entityList.item(0);     //source i
              map = entityNode.getAttributes();
              System.out.println("MapLength: "+map.getLength());
              for(int j=0;j<map.getLength();j++)  //Set attributes for entity
              {
                     curNode=map.item(j);
                     id = curNode.getNodeValue();
                     type = curNode.getNodeName();

                     if (type.equals("ident"))
                     { entityIdent = id; System.out.println(" Attribute
"+type+":"+entityIdent);}

              }
              if( entityNode.hasChildNodes() )
              {
                System.out.println("entity hasChildnodes");
                Node node = entityNode.getFirstChild();
                map = node.getAttributes();
                for(int j=0;j<map.getLength();j++)  //Set attributes for
entity
                {
                     curNode=map.item(j);
                     id = curNode.getNodeValue();
                     type = curNode.getNodeName();

                     if (type.equals("ident"))
                     { entityNodeIdent = id; System.out.println(" Attribute
"+type+":"+entityNodeIdent);}

                     if (type.equals("category"))
                     { entityNodeCategory = id; System.out.println("
Attribute  "+type+":"+entityNodeCategory);}

                     if (type.equals("location"))
                     { entityNodeLocation = id; System.out.println("
Attribute  "+type+":"+entityNodeLocation);}

                     if (type.equals("name"))
                     { entityNodeName = id; System.out.println(" Attribute
"+type+":"+entityNodeName);}

                }


                 if ( node.hasChildNodes() )
                 {
                     System.out.println(" Node hasChildnodes");
                     curList = node.getChildNodes();
                     int nodeSize = curList.getLength();
                     for(int j=0;j<nodeSize;j++)
                      {
                        curNode = curList.item(j);
                        if(curNode.getNodeName().equals("Address"))
                        {
                            entityNodeAddress  =
curNode.getFirstChild().getNodeValue();
```

```
                            System.out.println("entityNodeAddress:" +
entityNodeAddress);

                            map = curNode.getAttributes();
                            if(map.getLength()>0)  //  get address ident
                            {
                                    entityNodeAddressIdent =
       map.item(0).getNodeValue();
                                    System.out.println("entityNodeAddressIdent:
"+ entityNodeAddressIdent);
                             }
                         }
                      }
                   }
             }
       }


     //get analyzer
     curList = d.getElementsByTagName("Analyzer");
     int Asize = curList.getLength();
     id="";
     type="";
     if(Asize>0)
     {
             NodeList analyzerList = curList;

             Node analyzerNode = analyzerList.item(0);
             map = analyzerNode.getAttributes();
             System.out.println("MapLength: "+map.getLength());
             for(int j=0;j<map.getLength();j++)  //Set attributes for
analyzer
             {
                    curNode=map.item(j);
                    id = curNode.getNodeValue();
                    type = curNode.getNodeName();

                    if (type.equals("ident"))
                    { analyzerIdent = id; System.out.println(" Attribute
"+type+":"+analyzerIdent);}

             }
             if( analyzerNode.hasChildNodes() )
             {
                    System.out.println("analyzer hasChildnodes");
                    Node node = analyzerNode.getFirstChild();
                    map = node.getAttributes();
                    for(int j=0;j<map.getLength();j++)  //Set attributes for
analyzer
                    {
                            curNode=map.item(j);
                            id = curNode.getNodeValue();
                            type = curNode.getNodeName();

                            if (type.equals("ident"))
                            { analyzerNodeIdent = id; System.out.println("
Attribute  "+type+":"+analyzerNodeIdent);}

                            if (type.equals("category"))
                            { analyzerNodeCategory = id; System.out.println("
Attribute  "+type+":"+analyzerNodeCategory);}

                            if (type.equals("location"))
                            { analyzerNodeLocation = id; System.out.println("
Attribute  "+type+":"+analyzerNodeLocation);}

                            if (type.equals("name"))
```

```
                        { analyzerNodeName = id; System.out.println("
Attribute  "+type+":"+analyzerNodeName);}

                    }


                    if ( node.hasChildNodes() )
                    {
                           System.out.println(" Node hasChildnodes");
                           curList = node.getChildNodes();
                           int nodeSize = curList.getLength();
                           for(int j=0;j<nodeSize;j++)
                           {
                              curNode = curList.item(j);
                              if(curNode.getNodeName().equals("Address"))
                              {
                                  analyzerNodeAddress  =
curNode.getFirstChild().getNodeValue();

                                  System.out.println("analyzerNodeAddress:" +
analyzerNodeAddress);

                                  map = curNode.getAttributes();
                                  if(map.getLength()>0)
                                  {

                                     analyzerNodeAddressIdent =
map.item(0).getNodeValue();

                           System.out.println("analyzerNodeAddressIdent : " +
                           analyzerNodeAddressIdent);
                                  }
                              }
                        }
                }
            }
    }


     //get source(s)
     curList = d.getElementsByTagName("Source");
     int Ssize = curList.getLength();
     id="";
     type="";
     if(Ssize>0)
     {
        NodeList sourceList = curList;
        System.out.println("SourceLIst size: "+Ssize);

        sourceIdent= new String[Ssize];
        sourceSpoofed = new String[Ssize];
        sourceInterface=new String[Ssize];
        sourceNodeIdent=new String[Ssize];
        sourceNodeAddressIdent= new String[Ssize];
        sourceNodeAddress=new String[Ssize];

        for(int i=0;i<Ssize;i++)
        {
           sourceIdent[i]="NAN";
           sourceSpoofed [i]="NAN";
           sourceInterface[i]="NAN";
           sourceNodeIdent[i]="NAN";
           sourceNodeAddressIdent[i]="NAN";
           sourceNodeAddress[i]="NAN";
        }


        for(int i=0;i<Ssize;i++)
```

```
        {
              System.out.println("loop no:"+i);
              Node sourceNode = sourceList.item(i);     //source i
              map = sourceNode.getAttributes();
              System.out.println("MapLength: "+map.getLength());
              for(int j=0;j<map.getLength();j++)
              {
                     curNode=map.item(j);
                     id = curNode.getNodeValue();
                     type = curNode.getNodeName();

                     if (type.equals("ident"))
                     { sourceIdent[i] = id; System.out.println(i+" Attribute
"+type+":"+sourceIdent[i]);}
                     else if (type.equals("spoofed"))
                     {  sourceSpoofed[i] = id; System.out.println(i+"
Attribute  "+type+":"+sourceSpoofed[i]);}
                     else if (type.equals("interface"))
                     { sourceInterface[i] = id; System.out.println(i+"
Attribute  "+type+":"+sourceInterface[i]);}

              }
              if( sourceNode.hasChildNodes() )
              {
                 System.out.println("Source hasChildnodes");

                 Node node = sourceNode.getFirstChild();
                 System.out.println("found : "+node.getNodeName());
                 if ( node.hasChildNodes() )
                 {
                     System.out.println(" NOde hasChildnodes");
                     curList = node.getChildNodes();
                     int nodeSize = curList.getLength();
                     for(int j=0;j<nodeSize;j++)
                     {
                         curNode = curList.item(j);
                         if(curNode.getNodeName().equals("Address"))
                         {
                                sourceNodeAddress[i]  =
curNode.getFirstChild().getNodeValue();
                                System.out.println("sourceNodeAddress:" +
sourceNodeAddress[i]);

                                map = curNode.getAttributes();
                                if(map.getLength()>0)
                                {
                                   sourceNodeAddressIdent[i] =
map.item(0).getNodeValue();
                                   System.out.println("sourceNodeAddressIdent:
"+ sourceNodeAddressIdent[i]);
                                }
                         }
                     }
                 }
              }
           }
        }
      }

     //get target(s)
     curList = d.getElementsByTagName("Target");
     int size = curList.getLength();
     id="";
     type="";
     if(size>0)
     {
       NodeList sourceList = curList;
       System.out.println("SourceLIst size: "+size);
```

```
        targetIdent= new String[size];
        targetDecoy = new String[size];
        targetInterface=new String[size];
        targetPort = new int[size];
        targetNodeIdent=new String[size];
        targetNodeCategory= new String[size];
        targetNodeAddressIdent= new String[size];
        targetNodeAddress=new String[size];

        for(int i=0;i<size;i++)
        {
          targetIdent[i]="NAN";
          targetDecoy [i]="NAN";
          targetInterface[i]="NAN";
          targetPort[i]=0;
          targetNodeIdent[i]="NAN";
          targetNodeCategory[i]="NAN";
          targetNodeAddressIdent[i]="NAN";
          targetNodeAddress[i]="NAN";
        }


        for(int i=0;i<size;i++)
        {
               System.out.println("loop no:"+i);
               Node targetNode = sourceList.item(i);
               map = targetNode.getAttributes();
               System.out.println("MapLength: "+map.getLength());
               for(int j=0;j<map.getLength();j++)
               {
                      curNode=map.item(j);
                      id = curNode.getNodeValue();
                      type = curNode.getNodeName();

                      if (type.equals("ident"))
                      { targetIdent[i] = id; System.out.println(i+" Attribute
"+type+":"+targetIdent[i]);}
                      else if (type.equals("decoy"))
                      {  targetDecoy[i] = id; System.out.println(i+" Attribute
"+type+":"+targetDecoy[i]);}
                      else if (type.equals("interface"))
                      { targetInterface[i] = id; System.out.println(i+"
Attribute  "+type+":"+ targetInterface[i]);}
                      else if(type.equals("port"))
                      { targetPort[i] = (new Integer(id)).intValue();
System.out.println(i+" Attribute  "+type+":"+ targetPort[i]);}

               }
               if( targetNode.hasChildNodes() )
               {
                      System.out.println("Target hasChildnodes");
                      Node node = targetNode.getFirstChild();
                      System.out.println("found : "+node.getNodeName());
                      if ( node.hasChildNodes() )
                      {
                             System.out.println(" Node hasChildnodes");
                             curList = node.getChildNodes();
                             int nodeSize = curList.getLength();
                             for(int j=0;j<nodeSize;j++)
                             {
                                curNode = curList.item(j);
                                if(curNode.getNodeName().equals("Category"))
                                {
                                   targetNodeCategory[i]  =
curNode.getFirstChild().getNodeValue();
                                   System.out.println("targetNodeCategory:" +
targetNodeCategory[i]);
                                }
```

```
                                if(curNode.getNodeName().equals("Address"))
                                {
                                   targetNodeAddress[i]  =
curNode.getFirstChild().getNodeValue();
                                   System.out.println("targetNodeAddress:" +
targetNodeAddress[i]);

                                   map = curNode.getAttributes();
                                   if(map.getLength()>0)
                                   {
                                      targetNodeAddressIdent[i] =
map.item(0).getNodeValue();

System.out.println("targetNodeAddressIdent : " + targetNodeAddressIdent[i]);
                                    }
                                }
                             }
                          }
                    }
             }
      }
```

```
    //get classification
    curList = d.getElementsByTagName("Classification");
    size = curList.getLength();
    if(size>0)
    {
       NodeList classificationList = curList;
       classification = new String[size];
       for(int i=0;i<size;i++)
       {
             curNode = classificationList.item(i);
             curList = curNode.getChildNodes();
             curNode = curList.item(0);
             classification[i] = curNode.getNodeValue();
             System.out.println("Classification: " +classification[i]);
       }
    }


    //get additional data
    curList = d.getElementsByTagName("AdditionalData");
    size = curList.getLength();
    if(size>0)
    {
       NodeList adDataList = curList;
       adData= new String[size];
       for(int i=0;i<size;i++)
       {
             curNode = adDataList.item(i);
             curList = curNode.getChildNodes();
             curNode = curList.item(0);
             adData[i] = curNode.getNodeValue();
             System.out.println("AdditionalData: " +adData[i]);
       }
    }

    //alert from analyzer
    Alert al1 = new
Alert(ident,dateInt,timeInt,analyzerIdent,analyzerNodeIdent,analyzerNodeCate
gory,analyzerNodeLocation,analyzerNodeName,analyzerNodeAddressIdent,analyzer
NodeAddress,classification,adData,sourceIdent,sourceSpoofed,sourceInterface,
sourceNodeIdent,sourceNodeAddressIdent,sourceNodeAddress,targetIdent,targetD
```

```
ecoy,targetInterface,targetPort,targetNodeIdent,targetNodeCategory,
targetNodeAddressIdent,targetNodeAddress);

    //alert from entity
    Alert al2 = new
Alert(ident,dateInt,timeInt,classification,adData,entityIdent,entityNodeIden
t,entityNodeCategory,entityNodeLocation,entityNodeName,entityNodeAddressIden
t,entityNodeAddress,sourceIdent,sourceSpoofed,sourceInterface,sourceNodeIden
t,sourceNodeAddressIdent,sourceNodeAddress,targetIdent,targetDecoy,targetInt
erface,targetPort,targetNodeIdent,targetNodeCategory,
targetNodeAddressIdent,targetNodeAddress);

    if(Asize>0)
       return al1;
    else
       return al2;
  }


  Alert doc2AlertIDMEF(Document d)
  {
      NodeList rootList = d.getChildNodes();
      Node rootNode = rootList.item(0);
      Node curNode;
      NodeList curList;


      String ident="NAN";
      String dateString="NAN";
      String timeString="NAN";
      int dateInt=0;
      int timeInt=0;


      String analyzerIdent=null;
      String analyzerNodeIdent=null;
      String analyzerNodeCategory=null;
      String analyzerNodeLocation=null;
      String analyzerNodeName=null;
      String analyzerNodeAddressIdent=null;
      String analyzerNodeAddress=null;

      String[] sourceIdent=null;
      String[] sourceSpoofed=null;
      String[] sourceInterface=null;
      String[] sourceNodeIdent=null;
      String[] sourceNodeAddressIdent=null;
      String[] sourceNodeAddress=null;
      String[] targetIdent=null;
      String[] targetDecoy=null;
      String[] targetInterface=null;
      int[] targetPort=null;
      String[] targetNodeIdent=null;
      String[] targetNodeCategory=null;
      String[] targetNodeAddressIdent=null;
      String[] targetNodeAddress=null;

      String[] classification=null;
      String[] adData = null;


      NamedNodeMap map = rootNode.getAttributes();
      if (map.getLength()>0)
      {
        curNode = map.item(0);
        ident = curNode.getNodeValue();
        System.out.println("Alert ident:"+ident);
      }
```

```
      NodeList alertList = rootNode.getChildNodes();


      //get createTime
      curList = d.getElementsByTagName("CreateTime");
      curNode = curList.item(0);
      curList = curNode.getChildNodes();
      curNode = curList.item(0);
      dateString = curNode.getNodeValue();
      try
      {
          SimpleDateFormat formatter = new SimpleDateFormat ("yyyy-MM-
dd'T'HH:mm:ss'Z'");
          ParsePosition pos = new ParsePosition(0);
          Date currTime = formatter.parse(dateString, pos);
          System.out.println("CreateTime: " + dateString );
          Calendar cal = formatter.getCalendar();
          int nYear=cal.get( Calendar.YEAR );
          int nMonth=cal.get( Calendar.MONTH )+1;
          int nDay=cal.get( Calendar.DAY_OF_MONTH );

          dateInt = nDay +100*nMonth+ 10000*nYear;
          dateString = Integer.toString(nYear)+"-
"+Integer.toString(nMonth)+"-"+Integer.toString(nDay);
          System.out.println("Date: "+dateString);

          int nHour=cal.get( Calendar.HOUR_OF_DAY );
          int nMin=cal.get( Calendar.MINUTE );
          int nSec=cal.get( Calendar.SECOND );

          timeInt = nSec + 100*nMin + 10000*nHour;
          timeString =
Integer.toString(nHour)+":"+Integer.toString(nMin)+":"+Integer.toString(nSec
);
          System.out.println("Time: "+timeString);

     }
     catch(Exception e)
     {
          System.out.println(e.getMessage());
     }


     //get analyzer
     curList = d.getElementsByTagName("Analyzer");
     int Asize = curList.getLength();
     String id="";
     String type="";
     if(Asize>0)
     {
             NodeList analyzerList = curList;

             Node analyzerNode = analyzerList.item(0);
             map = analyzerNode.getAttributes();
             System.out.println("MapLength: "+map.getLength());
             for(int j=0;j<map.getLength();j++)  //Set attributes for
analyzer
             {
                    curNode=map.item(j);
                    id = curNode.getNodeValue();
                    type = curNode.getNodeName();

                    if (type.equals("analyzerid"))
                    { analyzerIdent = id; System.out.println(" Attribute
"+type+":"+analyzerIdent);}

             }
             if( analyzerNode.hasChildNodes() )
```

```
            {
                    System.out.println("analyzer hasChildnodes");
                    Node node = analyzerNode.getFirstChild();
                    map = node.getAttributes();
                    for(int j=0;j<map.getLength();j++)
                    {
                           curNode=map.item(j);
                           id = curNode.getNodeValue();
                           type = curNode.getNodeName();

                           if (type.equals("ident"))
                           { analyzerNodeIdent = id; System.out.println("
Attribute  "+type+":"+analyzerNodeIdent);}

                           if (type.equals("category"))
                           { analyzerNodeCategory = id; System.out.println("
Attribute  "+type+":"+analyzerNodeCategory);}

                           if (type.equals("location"))
                           { analyzerNodeLocation = id; System.out.println("
Attribute  "+type+":"+analyzerNodeLocation);}

                           if (type.equals("name"))
                           { analyzerNodeName = id; System.out.println("
Attribute  "+type+":"+analyzerNodeName);}

                    }


                    if ( node.hasChildNodes() )
                    {
                           System.out.println(" Node hasChildnodes");
                           curList = node.getChildNodes();
                           int nodeSize = curList.getLength();
                           for(int j=0;j<nodeSize;j++)
                           {
                              curNode = curList.item(j);
                              if(curNode.getNodeName().equals("Address"))
                              {

                                  map = curNode.getAttributes();
                                  if(map.getLength()>0)
                                  {
                                     analyzerNodeAddressIdent =
map.item(0).getNodeValue();
                               System.out.println("analyzerNodeAddressCategory : "+
analyzerNodeAddressIdent );
                                   }
                                   if ( curNode.hasChildNodes() )
                                   {
                                      System.out.println(" Address
hasChildnodes");
                                      curList = curNode.getChildNodes();
                                      nodeSize = curList.getLength();
                                      for( j=0;j<nodeSize;j++)
                                      {
                                         curNode = curList.item(j);

                                   if(curNode.getNodeName().equals("address"))
                                         {
                                            analyzerNodeAddress  =
                    curNode.getFirstChild().getNodeValue();

                    System.out.println("analyzerNodeAddress:"
 + analyzerNodeAddress);

                                         }
                                     }
```

```
                                    }
                                }
                            }
                        }
                    }

            }


            //get source(s)
            curList = d.getElementsByTagName("Source");
            int Ssize = curList.getLength();
            id="";
            type="";
            if(Ssize>0)
            {
               NodeList sourceList = curList;
               System.out.println("SourceLIst size: "+Ssize);

               sourceIdent= new String[Ssize];
               sourceSpoofed = new String[Ssize];
               sourceInterface=new String[Ssize];
               sourceNodeIdent=new String[Ssize];
               sourceNodeAddressIdent= new String[Ssize];
               sourceNodeAddress=new String[Ssize];

               for(int i=0;i<Ssize;i++)
               {
                  sourceIdent[i]="NAN";
                  sourceSpoofed [i]="NAN";
                  sourceInterface[i]="NAN";
                  sourceNodeIdent[i]="NAN";
                  sourceNodeAddressIdent[i]="NAN";
                  sourceNodeAddress[i]="NAN";
               }


               for(int i=0;i<Ssize;i++)
               {
                     System.out.println("loop no:"+i);
                     Node sourceNode = sourceList.item(i);     //source i
                     map = sourceNode.getAttributes();
                     System.out.println("MapLength: "+map.getLength());
                     for(int j=0;j<map.getLength();j++)
                     {
                            curNode=map.item(j);
                            id = curNode.getNodeValue();
                            type = curNode.getNodeName();

                            if (type.equals("ident"))
                            { sourceIdent[i] = id; System.out.println(i+" Attribute
"+type+":"+sourceIdent[i]);}
                            else if (type.equals("spoofed"))
                            {  sourceSpoofed[i] = id; System.out.println(i+"
Attribute  "+type+":"+sourceSpoofed[i]);}
                            else if (type.equals("interface"))
                            { sourceInterface[i] = id; System.out.println(i+"
Attribute  "+type+":"+sourceInterface[i]);}

                     }
                     if( sourceNode.hasChildNodes() )
                     {
                            System.out.println("Source hasChildnodes");
                            Node node = sourceNode.getFirstChild();
                            System.out.println("found : "+node.getNodeName());
                            if ( node.hasChildNodes() )
                            {
                               System.out.println(" Node hasChildnodes");
```

```
                        curList = node.getChildNodes();
                        int nodeSize = curList.getLength();
                        for(int j=0;j<nodeSize;j++)
                        {
                           curNode = curList.item(j);
                           if(curNode.getNodeName().equals("Address"))
                           {
                                map = curNode.getAttributes();
                                if(map.getLength()>0)
                                {
                                    sourceNodeAddressIdent[i] =
map.item(0).getNodeValue();

       System.out.println("sourceNodeAddressCategory: "+
sourceNodeAddressIdent[i]);
                                }

                                if ( curNode.hasChildNodes() )
                                {
                                   System.out.println(" Address
hasChildnodes");
                                   curList = curNode.getChildNodes();
                                   nodeSize = curList.getLength();
                                   for( j=0;j<nodeSize;j++)
                                   {
                                       curNode = curList.item(j);

if(curNode.getNodeName().equals("address"))
                                           {
                                              sourceNodeAddress[i]  =
curNode.getFirstChild().getNodeValue();

       System.out.println("sourceNodeAddress:" + sourceNodeAddress[i]);

                                           }
                                    }
                                }

                             }
                          }
                      }
                 }
            }
     }

     //get target(s)
     curList = d.getElementsByTagName("Target");
     int size = curList.getLength();
     id="";
     type="";
     if(size>0)
     {
       NodeList sourceList = curList;
       System.out.println("SourceLIst size: "+size);

       targetIdent= new String[size];
       targetDecoy = new String[size];
       targetInterface=new String[size];
       targetPort = new int[size];
       targetNodeIdent=new String[size];
       targetNodeCategory= new String[size];
       targetNodeAddressIdent= new String[size];
       targetNodeAddress=new String[size];

       for(int i=0;i<size;i++)
       {
         targetIdent[i]="NAN";
         targetDecoy [i]="NAN";
```

```
          targetInterface[i]="NAN";
          targetPort[i]=0;
          targetNodeIdent[i]="NAN";
          targetNodeCategory[i]="NAN";
          targetNodeAddressIdent[i]="NAN";
          targetNodeAddress[i]="NAN";
        }


        for(int i=0;i<size;i++)
        {
              System.out.println("loop no:"+i);
              Node targetNode = sourceList.item(i);
              map = targetNode.getAttributes();
              System.out.println("MapLength: "+map.getLength());
              for(int j=0;j<map.getLength();j++)
              {
                  curNode=map.item(j);
                  id = curNode.getNodeValue();
                  type = curNode.getNodeName();

                  if (type.equals("ident"))
                  { targetIdent[i] = id; System.out.println(i+" Attribute
"+type+":"+targetIdent[i]);}
                  else if (type.equals("decoy"))
                  {  targetDecoy[i] = id; System.out.println(i+" Attribute
"+type+":"+targetDecoy[i]);}
                  else if (type.equals("interface"))
                   { targetInterface[i] = id; System.out.println(i+" Attribute
"+type+":"+ targetInterface[i]);}

                   }
                   if( targetNode.hasChildNodes() )
                   {
                      System.out.println("Target hasChildnodes");

                       Node node = targetNode.getFirstChild();
                       System.out.println("found : "+node.getNodeName());
                       if ( node.hasChildNodes() )
                       {
                           System.out.println(" Node hasChildnodes");
                           curList = node.getChildNodes();
                           int nodeSize = curList.getLength();
                           for(int j=0;j<nodeSize;j++)
                           {
                               curNode = curList.item(j);
                               if(curNode.getNodeName().equals("Category"))
                               {
                                 targetNodeCategory[i]  =
curNode.getFirstChild().getNodeValue();
                                   System.out.println("targetNodeCategory:" +
targetNodeCategory[i]);
                               }

                               if(curNode.getNodeName().equals("Address"))
                               {

                                  map = curNode.getAttributes();
                                  if(map.getLength()>0)
                                  {
                                     targetNodeAddressIdent[i] =
map.item(0).getNodeValue();
                                    System.out.println("targetNodeAddressIdent:
"+ targetNodeAddressIdent[i]);
                                  }

                                  if ( curNode.hasChildNodes() )
                                  {
```

```
                                    System.out.println(" Address
hasChildnodes");
                                    curList = curNode.getChildNodes();
                                    nodeSize = curList.getLength();
                                    for( j=0;j<nodeSize;j++)
                                    {
                                        curNode = curList.item(j);

       if(curNode.getNodeName().equals("address"))
                                        {
                                           targetNodeAddress[i]  =
curNode.getFirstChild().getNodeValue();

       System.out.println("targetNodeAddress:" + targetNodeAddress[i]);

                                         }
                                     }
                                  }
                               }
                           }
                  }

                  node = targetNode.getLastChild();
                  if (node.getNodeName() == "Service")
                  {
                  if ( node.hasChildNodes() )
                  {
                     System.out.println(" Service hasChildnodes");
                     curList = node.getChildNodes();
                     int nodeSize = curList.getLength();

                     for(int j=0;j<nodeSize;j++)
                     {
                         curNode = curList.item(j);

                         if(curNode.getNodeName().equals("port"))
                         {
                                id = curNode.getFirstChild().getNodeValue();
                                                targetPort[i] = (new
Integer(id)).intValue();
                                System.out.println("targetPort:" +
targetPort[i]);

                          }
                       }
                    }
                 }
              }
         }
    }



    //get classification
    curList = d.getElementsByTagName("Classification");
    size = curList.getLength();
    if(size>0)
    {
       NodeList classificationList = curList;
       classification = new String[size];
       for(int i=0;i<size;i++)
       {
              curNode = classificationList.item(i);

              if ( curNode.hasChildNodes() )
              {
                     System.out.println("Classification hasChildnodes");
```

```
                    curList = curNode.getChildNodes();

                    curNode = curList.item(0);
                    if(curNode.getNodeName().equals("name"))
                    {
                          classification[i]  =
curNode.getFirstChild().getNodeValue();
                          System.out.println("Classification: "
+classification[i]);
                    }
               }

        }
    }


    //get additional data
    curList = d.getElementsByTagName("AdditionalData");
    size = curList.getLength();
    if(size>0)
    {
       NodeList adDataList = curList;
       adData= new String[size];
       for(int i=0;i<size;i++)
       {
             curNode = adDataList.item(i);
             curList = curNode.getChildNodes();
             curNode = curList.item(0);
             adData[i] = curNode.getNodeValue();
             System.out.println("AdditionalData: " +adData[i]);
       }
     }

     Alert al = new
Alert(ident,dateInt,timeInt,analyzerIdent,analyzerNodeIdent,analyzerNodeCate
gory,analyzerNodeLocation,analyzerNodeName,analyzerNodeAddressIdent,analyzer
NodeAddress,classification,adData,sourceIdent,sourceSpoofed,sourceInterface,
sourceNodeIdent,sourceNodeAddressIdent,sourceNodeAddress,targetIdent,targetD
ecoy,targetInterface,targetPort,targetNodeIdent,targetNodeCategory,
targetNodeAddressIdent,targetNodeAddress);

      return al;
    }


    Heartbeat doc2Heartbeat(Document d)
    {

      String ident="NAN";
      String dateString="NAN";
      String timeString="NAN";
      String[] adData = null;

      NodeList rootList = d.getChildNodes();
      Node rootNode = rootList.item(0);
      NodeList heartbeatList = rootNode.getChildNodes();
      Node curNode;
      NodeList curList;

      //get heartbeat ident

      NamedNodeMap map = rootNode.getAttributes();
      if (map.getLength()>0)
      {
        curNode = map.item(0);
        ident = curNode.getNodeValue();
       System.out.println("Heartbeat ident: "+ident);
      }
```

```
       //get createTime
        curList = d.getElementsByTagName("CreateTime");
        curNode = curList.item(0);
        curList = curNode.getChildNodes();
        curNode = curList.item(0);
        dateString = curNode.getNodeValue();
        try
        {
               SimpleDateFormat formatter = new SimpleDateFormat ("yyyy.MM.dd
G 'at' hh:mm:ss a zzz");
               ParsePosition pos = new ParsePosition(0);
               Date currTime = formatter.parse(dateString, pos);
               System.out.println("CreateTime: " + dateString );
               Calendar cal = formatter.getCalendar();
               int nYear=cal.get( Calendar.YEAR );

               int nMonth=cal.get( Calendar.MONTH )+1;
               int nDay=cal.get( Calendar.DAY_OF_MONTH );
               dateString = Integer.toString(nYear)+"-
"+Integer.toString(nMonth)+"-"+Integer.toString(nDay);
               System.out.println("Date: "+dateString);

               int nHour=cal.get( Calendar.HOUR_OF_DAY );
               int nMin=cal.get( Calendar.MINUTE );
               int nSec=cal.get( Calendar.SECOND );
               timeString =
Integer.toString(nHour)+":"+Integer.toString(nMin)+":"+Integer.toString(nSec
);
               System.out.println("Time: "+timeString);

      }
      catch(Exception e)
      {
               System.out.println(e.getMessage());
      }

      //get additional data
      curList = d.getElementsByTagName("AdditionalData");
      int size = curList.getLength();
      if(size>0)
      {
         NodeList adDataList = curList;
         adData = new String[size];
         for(int i=0;i<size;i++)
         {
               curNode = adDataList.item(i);
               curList = curNode.getChildNodes();
               curNode = curList.item(0);
               adData[i] = curNode.getNodeValue();
               System.out.println("AdditionalData: " +adData[i]);
         }
      }

        Heartbeat h = new Heartbeat(ident,dateString,timeString,adData);
        return h ;
    }


     Heartbeat doc2HeartbeatIDMEF(Document d)
     {

       String ident="NAN";
       String dateString="NAN";
       String timeString="NAN";
       String[] adData = null;

       NodeList rootList = d.getChildNodes();
```

```
       Node rootNode = rootList.item(0);
       NodeList heartbeatList = rootNode.getChildNodes();
       Node curNode;
       NodeList curList;

       //get heartbeat ident


        NodeList nl = d.getElementsByTagName("address");
        ident = nl.item(0).getFirstChild().getNodeValue();
        System.out.println("Heartbeat ident: "+ident);

        //get createTime
        curList = d.getElementsByTagName("CreateTime");
        curNode = curList.item(0);
        curList = curNode.getChildNodes();
        curNode = curList.item(0);
        dateString = curNode.getNodeValue();
        try
        {
               SimpleDateFormat formatter = new SimpleDateFormat ("yyyy-MM-
dd'T'HH:mm:ss'Z'");
               ParsePosition pos = new ParsePosition(0);
               Date currTime = formatter.parse(dateString, pos);
               System.out.println("CreateTime: " + dateString );
               Calendar cal = formatter.getCalendar();
               int nYear=cal.get( Calendar.YEAR );
               int nMonth=cal.get( Calendar.MONTH )+1;
               int nDay=cal.get( Calendar.DAY_OF_MONTH );
               dateString = Integer.toString(nYear)+"-
"+Integer.toString(nMonth)+"-"+Integer.toString(nDay);
               System.out.println("Date: "+dateString);

               int nHour=cal.get( Calendar.HOUR_OF_DAY );
               int nMin=cal.get( Calendar.MINUTE );
               int nSec=cal.get( Calendar.SECOND );
               timeString =
Integer.toString(nHour)+":"+Integer.toString(nMin)+":"+Integer.toString(nSec
);
               System.out.println("Time: "+timeString);

      }
      catch(Exception e)
      {
               System.out.println(e.getMessage());
      }

      //get additional data
      curList = d.getElementsByTagName("AdditionalData");
      int size = curList.getLength();
      if(size>0)
      {
         NodeList adDataList = curList;
         adData = new String[size];
         for(int i=0;i<size;i++)
         {
               curNode = adDataList.item(i);
               curList = curNode.getChildNodes();
               curNode = curList.item(0);
               adData[i] = curNode.getNodeValue();
               System.out.println("AdditionalData: " +adData[i]);
         }
      }

        Heartbeat h = new Heartbeat(ident,dateString,timeString,adData);
        return h ;
    }
```

```
   public Alert returnAlert()
   {
       return alert;
   }

   public Heartbeat returnHeartbeat()
   {
        return heartbeat;
   }

   public Xml2Doc(String message)
   {
      try
      {

        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        //System.out.println(message);
        Document doc  = db.parse( new InputSource(new
StringReader(message)));

        NodeList rootList = doc.getChildNodes();
        Node rootNode = rootList.item(0);

        if (rootNode.getNodeName().compareTo("Alert")==0)
              alert = doc2Alert(doc);
        else if (rootNode.getNodeName().compareTo("Heartbeat")==0)
              heartbeat = doc2Heartbeat(doc);
        else if (rootNode.getNodeName().compareTo("Signed-IDMEF-Message")==0)
        {

              /* Get the IDMEF Message */
              int startmsg = message.indexOf("<IDMEF-Message");
              int endmsg = message.lastIndexOf("</IDMEF-Message>");
              String idmefmsg = message.substring(startmsg, endmsg+16);
              byte[] idmefbytes = idmefmsg.getBytes();
              // System.out.println(idmefmsg);

              /* Grab the Signature */
              NodeList nl = doc.getElementsByTagName("Signature");
              String s = nl.item(0).getFirstChild().getNodeValue();

              Hex h = new Hex();
              byte[] sigbytes = h.decode(s);

              /* Get the analyzerid */
              NamedNodeMap aid = nl.item(0).getAttributes();
              Node anid = aid.item(0);

              String aidName = anid.getNodeName();
              String aidValue = anid.getNodeValue();

              /* Build the key filename & path */
              String publicKeyFilename = "/home/gandr/";
              publicKeyFilename += aidValue;
              publicKeyFilename += ".der";

              ByteArrayOutputStream baos = new ByteArrayOutputStream();

              // Load the public key bytes
              try
              {
                 FileInputStream fis = new
FileInputStream(publicKeyFilename);

                 int theByte = 0;
                 while ((theByte = fis.read()) != -1)
                 {
```

```
                    baos.write(theByte);
                }
                fis.close();
            }
            catch (FileNotFoundException e)
            {
                    System.out.println(e.getMessage());
            }

            byte[] keyBytes = baos.toByteArray();
            baos.close();

            try
            {
               // Turn the encoded key into a real RSA public key.
               // Public keys are encoded in X.509.
               X509EncodedKeySpec keySpec = new
X509EncodedKeySpec(keyBytes);
               KeyFactory keyFactory = KeyFactory.getInstance("RSA");
               PublicKey publicKey = keyFactory.generatePublic(keySpec);

               Signature sig = Signature.getInstance("MD5withRSA");
               sig.initVerify(publicKey);
               sig.update(idmefbytes);
               boolean valid = sig.verify(sigbytes);
               if (valid == true)
               {
                   System.out.println("Signature is valid");

                   /* IDMEF Message is verified so continue */

                   NodeList nl2 = doc.getElementsByTagName("Alert");

                   if (nl2.getLength() == 1)  /* Alert */
                   {
                          int start = message.indexOf("<Alert");
                          int end = message.lastIndexOf("</Alert>");
                          String msg = message.substring(start, end+8);
            //            System.out.println(msg);
                          DocumentBuilderFactory dbf2 =
DocumentBuilderFactory.newInstance();
                          DocumentBuilder db2 = dbf2.newDocumentBuilder();
                          Document doc2 = db2.parse( new InputSource(new
StringReader(msg)));
                          alert = doc2AlertIDMEF(doc2);
                   }
                   else /* HeartBeat */
                   {
                          int start = message.indexOf("<Heartbeat");
                          int end = message.lastIndexOf("</Heartbeat>");
                          String msg = message.substring(start, end+12);
            //            System.out.println(msg);
                          DocumentBuilderFactory dbf2 =
DocumentBuilderFactory.newInstance();
                          DocumentBuilder db2 = dbf2.newDocumentBuilder();
                          Document doc2 = db.parse( new InputSource(new
StringReader(msg)));
                          heartbeat = doc2HeartbeatIDMEF(doc2);
                   }

               }
               else
               {
                   System.out.println("Signature is invalid");
               }
            }
            catch (NoSuchAlgorithmException e)
            {
```

```
                        System.out.println(e.getMessage());
                }
                catch (InvalidKeySpecException e)
                {
                        System.out.println(e.getMessage());
                }

           }

      }
      catch (Exception e)
          {e.printStackTrace();}

   }


}
```

Αρχείο idmef.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ************************************************************
 ************************************************************
 *** Intrusion Detection Message Exchange Format (IDMEF) XML DTD ***
 ***                Version 1.0, 8 November 2002                ***
 ***                                                            ***
 *** The use and extension of the IDMEF XML DTD are described in ***
 *** RFC XXXX, "Intrusion Detection Message Exchange Format Data ***
 *** Model and Extensible Markup Language (XML) Document Type    ***
 *** Definition," D. Curry and H. Debar.                         ***
 ************************************************************
 ************************************************************ -->

<!-- ============================================================
 ============================================================
 === SECTION 1. Attribute list declarations.
 ============================================================
 ============================================================ -->

<!--
 | Attributes of the IDMEF element.  In general, the fixed values of
 | these attributes will change each time a new version of the DTD
 | is released.
 -->
<!ENTITY % attlist.idmef                "
    version             CDATA                   #FIXED    '1.0'
  ">

<!--
 | Attributes of all elements.  These are the "XML" attributes that
 | every element should have.  Space handling, language, and name
 | space.
 -->
<!ENTITY % attlist.global               "
    xmlns:idmef         CDATA                   #FIXED
        'urn:iana:xml:ns:idmef'
    xmlns               CDATA                   #FIXED
        'urn:iana:xml:ns:idmef'
    xml:space           (default | preserve)    'default'
    xml:lang            NMTOKEN                 #IMPLIED
  ">

<!-- ============================================================
 ============================================================
 === SECTION 2. Attribute value declarations.  Enumerated values for
 ===            many of the element-specific attribute lists.
 ============================================================
 ============================================================ -->

<!--
 | Values for the Action.category attribute.
 -->
<!ENTITY % attvals.actioncat            "
    ( block-installed | notification-sent | taken-offline | other )
  ">

<!--
 | Values for the Address.category attribute.
 -->
<!ENTITY % attvals.addrcat              "
    ( unknown | atm | e-mail | lotus-notes | mac | sna | vm |
      ipv4-addr | ipv4-addr-hex | ipv4-net | ipv4-net-mask |
      ipv6-addr | ipv6-addr-hex | ipv6-net | ipv6-net-mask )
  ">
```

```
<!--
 | Values for the AdditionalData.type attribute.
 -->
<!ENTITY % attvals.adtype               "
    ( boolean | byte | character | date-time | integer | ntpstamp |
      portlist | real | string | xml )
  ">

<!--
 | Values for the Impact.completion attribute.
 -->
<!ENTITY % attvals.completion           "
    ( failed | succeeded )
  ">

<!--
 | Values for the File.category attribute.
 -->
<!ENTITY % attvals.filecat              "
    ( current | original )
  ">

<!--
 | Values for the Id.type attribute.
 -->
<!ENTITY % attvals.idtype               "
    ( current-user | original-user | target-user | user-privs |
      current-group | group-privs | other-privs )
  ">

<!--
 | Values for the Impact.type attribute.
 -->
<!ENTITY % attvals.impacttype           "
    ( admin | dos | file | recon | user | other )
  ">

<!--
 | Values for the Linkage.category attribute.
 -->
<!ENTITY % attvals.linkcat              "
    ( hard-link | mount-point | reparse-point | shortcut | stream |
      symbolic-link )
  ">

<!--
 | Values for the Node.category attribute.
 -->
<!ENTITY % attvals.nodecat              "
    ( unknown | ads | afs | coda | dfs | dns | hosts | kerberos |
      nds | nis | nisplus | nt | wfw )
  ">

<!--
 | Values for the Classification.origin attribute.
 -->
<!ENTITY % attvals.origin               "
    ( unknown | bugtraqid | cve | vendor-specific )
  ">

<!--
 | Values for the Confidence.rating attribute.
 -->
<!ENTITY % attvals.rating               "
    ( low | medium | high | numeric )
  ">
```

```
<!--
 | Values for the Impact.severity attribute.
 -->
<!ENTITY % attvals.severity             "
    ( low | medium | high )
  ">

<!--
 | Values for the User.category attribute.
 -->
<!ENTITY % attvals.usercat              "
    ( unknown | application | os-device )
  ">

<!--
 | Values for yes/no attributes such as Source.spoofed and
 | Target.decoy.
 -->
<!ENTITY % attvals.yesno                "
    ( unknown | yes | no )
  ">

<!-- ====================================================================
 ====================================================================
 === SECTION 3. Top-level element declarations.  The IDMEF-Message
 ===            element and the types of messages it can include.
 ====================================================================
 ==================================================================== -->

<!ELEMENT IDMEF-Message                 (
    (Alert | Heartbeat)*
  )>
<!ATTLIST IDMEF-Message
    %attlist.global;
    %attlist.idmef;
  >

<!ELEMENT Alert                         (
    Analyzer, CreateTime, DetectTime?, AnalyzerTime?, Source*,
    Target*, Classification+, Assessment?, (ToolAlert |
    OverflowAlert | CorrelationAlert)?, AdditionalData*
  )>
<!ATTLIST Alert
    ident               CDATA                   '0'
    %attlist.global;
  >

<!ELEMENT Heartbeat                     (
    Analyzer, CreateTime, AnalyzerTime?, AdditionalData*
  )>
<!ATTLIST Heartbeat
    ident               CDATA                   '0'
    %attlist.global;
  >

<!-- ====================================================================
 ====================================================================
 === SECTION 4. Subclasses of the Alert element that provide more
 ===            data for specific types of alerts.
 ====================================================================
 ==================================================================== -->

<!ELEMENT CorrelationAlert              (
    name, alertident+
  )>
<!ATTLIST CorrelationAlert
    %attlist.global;
  >
```

```
<!ELEMENT OverflowAlert                 (
    program, size?, buffer?
  )>
<!ATTLIST OverflowAlert
    %attlist.global;
  >

<!ELEMENT ToolAlert                     (
    name, command?, alertident+
  )>
<!ATTLIST ToolAlert
    %attlist.global;
  >

<!-- ====================================================================
 ====================================================================
 === SECTION 5.  The AdditionalData element.  This element allows an
 ===             alert to include additional information that cannot
 ===             be encoded elsewhere in the data model.
 ====================================================================
 ==================================================================== -->

<!ELEMENT AdditionalData            ANY >
<!ATTLIST AdditionalData
    type                %attvals.adtype;        'string'
    meaning             CDATA                   #IMPLIED
    %attlist.global;
  >

<!-- ====================================================================
 ====================================================================
 === SECTION 6. Elements related to identifying entities - analyzers
 ===            (the senders of these messages), sources (of
 ===            attacks), and targets (of attacks).
 ====================================================================
 ==================================================================== -->

<!ELEMENT Analyzer                      (
    Node?, Process?
  )>
<!ATTLIST Analyzer
    analyzerid          CDATA                   '0'
    manufacturer        CDATA                   #IMPLIED
    model               CDATA                   #IMPLIED
    version             CDATA                   #IMPLIED
    class               CDATA                   #IMPLIED
    ostype              CDATA                   #IMPLIED
    osversion           CDATA                   #IMPLIED
    %attlist.global;
  >

<!ELEMENT Source                        (
    Node?, User?, Process?, Service?
  )>
<!ATTLIST Source
    ident               CDATA                   '0'
    spoofed             %attvals.yesno;         'unknown'
    interface           CDATA                   #IMPLIED
    %attlist.global;
  >

<!ELEMENT Target                        (
    Node?, User?, Process?, Service?, FileList?
  )>
<!ATTLIST Target
    ident               CDATA                   '0'
    decoy               %attvals.yesno;         'unknown'
```

```
    interface           CDATA                   #IMPLIED
    %attlist.global;
  >

<!-- ====================================================================
 ====================================================================
 === SECTION 7. Support elements used for providing detailed info
 ===            about entities - addresses, names, etc.
 ====================================================================
 ================================================================== -->

<!ELEMENT Address                       (
    address, netmask?
  )>
<!ATTLIST Address
    ident               CDATA                   '0'
    category            %attvals.addrcat;       'unknown'
    vlan-name           CDATA                   #IMPLIED
    vlan-num            CDATA                   #IMPLIED
    %attlist.global;
  >

<!ELEMENT Assessment                    (
    Impact?, Action*, Confidence?
  )>
<!ATTLIST Assessment
    %attlist.global;
  >

<!ELEMENT Classification                (
    name, url
  )>
<!ATTLIST Classification
    origin              %attvals.origin;        'unknown'
    %attlist.global;
  >

<!ELEMENT File                          (
    name, path, create-time?, modify-time?, access-time?,
    data-size?, disk-size?, FileAccess*, Linkage*, Inode?
  )>
<!ATTLIST File
    ident               CDATA                   '0'
    category            %attvals.filecat;       #REQUIRED
    fstype              CDATA                   #REQUIRED
    %attlist.global;
  >

<!ELEMENT FileAccess                    (
    UserId, permission+
  )>
<!ATTLIST FileAccess
    %attlist.global;
  >

<!ELEMENT FileList                      (
    File+
  )>
<!ATTLIST FileList
    %attlist.global;
  >

<!ELEMENT Inode                         (
    change-time?, (number, major-device, minor-device)?,
    (c-major-device, c-minor-device)?
  )>
<!ATTLIST Inode
    %attlist.global;
```

```
  >

<!ELEMENT Linkage                       (
    (name, path) | File
  )>
<!ATTLIST Linkage
    category            %attvals.linkcat;       #REQUIRED
    %attlist.global;
  >

<!ELEMENT Node                          (
    location?, (name | Address), Address*
  )>
<!ATTLIST Node
    ident               CDATA                   '0'
    category            %attvals.nodecat;       'unknown'
    %attlist.global;
  >

<!ELEMENT Process                       (
    name, pid?, path?, arg*, env*
  )>
<!ATTLIST Process
    ident               CDATA                   '0'
    %attlist.global;
  >

<!ELEMENT Service                       (
    (((name, port?) | (port, name?)) | portlist), protocol?,
    SNMPService?, WebService?
  )>
<!ATTLIST Service
    ident               CDATA                   '0'
    %attlist.global;
  >

<!ELEMENT SNMPService                   (
    oid?, community?, command?
  )>
<!ATTLIST SNMPService
    %attlist.global;
  >

<!ELEMENT User                          (
    UserId+
  )>
<!ATTLIST User
    ident               CDATA                   '0'
    category            %attvals.usercat;       'unknown'
    %attlist.global;
  >

<!ELEMENT UserId                        (
    (name, number?) | (number, name?)
  )>
<!ATTLIST UserId
    ident               CDATA                   '0'
    type                %attvals.idtype;        'original-user'
    %attlist.global;
  >

<!ELEMENT WebService                    (
    url, cgi?, http-method?, arg*
  )>
<!ATTLIST WebService
    %attlist.global;
  >
```

```
<!-- ====================================================================
 ====================================================================
 === SECTION 8. Simple elements with sub-elements or attributes of a
 ===            special nature.
 ====================================================================
 ==================================================================== -->

<!ELEMENT Action              (#PCDATA) >
<!ATTLIST Action
    category            %attvals.actioncat;     'other'
    %attlist.global;
  >

<!ELEMENT AnalyzerTime        (#PCDATA) >
<!ATTLIST AnalyzerTime
    ntpstamp            CDATA                   #REQUIRED
    %attlist.global;
  >

<!ELEMENT Confidence          (#PCDATA) >
<!ATTLIST Confidence
    rating              %attvals.rating;        'numeric'
    %attlist.global;
  >

<!ELEMENT CreateTime          (#PCDATA) >
<!ATTLIST CreateTime
    ntpstamp            CDATA                   #REQUIRED
    %attlist.global;
  >

<!ELEMENT DetectTime          (#PCDATA) >
<!ATTLIST DetectTime
    ntpstamp            CDATA                   #REQUIRED
    %attlist.global;

  >

<!ELEMENT Impact              (#PCDATA) >
<!ATTLIST Impact
    severity            %attvals.severity;      #IMPLIED
    completion          %attvals.completion;    #IMPLIED
    type                %attvals.impacttype;    'other'
    %attlist.global;
  >

<!ELEMENT alertident          (#PCDATA) >
<!ATTLIST alertident
    analyzerid          CDATA                   #IMPLIED
    %attlist.global;
  >

<!-- ====================================================================
 ====================================================================
 === SECTION 9. Simple elements with no sub-elements and no special
 ===            attributes.
 ====================================================================
 ==================================================================== -->

<!ELEMENT access-time         (#PCDATA) >
<!ATTLIST access-time
    %attlist.global;
  >

<!ELEMENT address             (#PCDATA) >
<!ATTLIST address
    %attlist.global;
  >
```

```
<!ELEMENT arg                (#PCDATA) >
<!ATTLIST arg
    %attlist.global;
  >

<!ELEMENT buffer             (#PCDATA) >
<!ATTLIST buffer
    %attlist.global;
  >

<!ELEMENT c-major-device     (#PCDATA) >
<!ATTLIST c-major-device
    %attlist.global;
  >

<!ELEMENT c-minor-device     (#PCDATA) >
<!ATTLIST c-minor-device
    %attlist.global;
  >

<!ELEMENT cgi                (#PCDATA) >
<!ATTLIST cgi
    %attlist.global;
  >

<!ELEMENT change-time        (#PCDATA) >
<!ATTLIST change-time
    %attlist.global;
  >

<!ELEMENT command            (#PCDATA) >
<!ATTLIST command
    %attlist.global;
  >

<!ELEMENT community          (#PCDATA) >
<!ATTLIST community
    %attlist.global;
  >

<!ELEMENT create-time        (#PCDATA) >
<!ATTLIST create-time
    %attlist.global;
  >

<!ELEMENT data-size          (#PCDATA) >
<!ATTLIST data-size
    %attlist.global;
  >

<!ELEMENT disk-size          (#PCDATA) >
<!ATTLIST disk-size
    %attlist.global;
  >

<!ELEMENT env                (#PCDATA) >
<!ATTLIST env
    %attlist.global;
  >

<!ELEMENT http-method        (#PCDATA) >
<!ATTLIST http-method
    %attlist.global;
  >

<!ELEMENT location           (#PCDATA) >
<!ATTLIST location
```

```
    %attlist.global;
  >

<!ELEMENT major-device        (#PCDATA) >
<!ATTLIST major-device
    %attlist.global;
  >

<!ELEMENT minor-device        (#PCDATA) >
<!ATTLIST minor-device
    %attlist.global;
  >

<!ELEMENT modify-time         (#PCDATA) >
<!ATTLIST modify-time
    %attlist.global;
  >

<!ELEMENT name                (#PCDATA) >
<!ATTLIST name
    %attlist.global;
  >

<!ELEMENT netmask             (#PCDATA) >
<!ATTLIST netmask
    %attlist.global;
  >

<!ELEMENT number              (#PCDATA) >
<!ATTLIST number
    %attlist.global;
  >

<!ELEMENT oid                 (#PCDATA) >
<!ATTLIST oid
    %attlist.global;
  >

<!ELEMENT path                (#PCDATA) >
<!ATTLIST path
    %attlist.global;
  >

<!ELEMENT permission          (#PCDATA) >
<!ATTLIST permission
    %attlist.global;
  >

<!ELEMENT pid                 (#PCDATA) >
<!ATTLIST pid
    %attlist.global;
  >

<!ELEMENT port                (#PCDATA) >
<!ATTLIST port
    %attlist.global;
  >

<!ELEMENT portlist            (#PCDATA) >
<!ATTLIST portlist
    %attlist.global;
  >

<!ELEMENT program             (#PCDATA) >
<!ATTLIST program
    %attlist.global;
  >
```

```
<!ELEMENT protocol            (#PCDATA) >
<!ATTLIST protocol
    %attlist.global;
  >

<!ELEMENT size                (#PCDATA) >
<!ATTLIST size
    %attlist.global;
  >

<!ELEMENT url                 (#PCDATA) >
<!ATTLIST url
    %attlist.global;
  >
```