2012

Θανάσης Λιούτας

Ψηφιακά Συστήματα
E/01076
thanoslioutas@yahoo.gr

Πάτσιας Μιχαήλ

Ψηφιακά Συστήματα
E/01125
mikepatsias13@hotmail.com

[ΜΕΤΑ-ΕΠΕΞΕΡΓΑΣΙΑ ΣΥΝΑΓΕΡΜΩΝ IDS ΣΥΣΤΗΜΑΤΩΝ]

[Η πτυχιακή αυτή εργασία πραγματεύεται τη μελέτη και υλοποίηση αλγορίθμων για την μετα-επεξεργασία των συναγερμών που δημιουργούν Intrusion Detection Systems.]

Πίνακας Περιεχομένων

Εκφώνηση

Τα συστήματα αναγνώρισης εισβολών αποτελούν βασικό εργαλείο στην αντιμετώπιση ποικίλλων απειλών και επιθέσεων σε ένα δίκτυο. Η ποιότητα των συναγερμών όμως που παράγουν δεν είναι η επιθυμητή. Σημαντική περιοχή έρευνας αποτελεί η μετα-επεξεργασία αυτών των συναγερμών, με σκοπό την αύξηση της ποιότητας τους. Η πτυχιακή αυτή αφορά τον σχεδιασμό και την υλοποίηση (σε γλώσσα JAVA) πλατφόρμας η οποία θα επιτρέπει στο χρήστη να σχεδιάζει λύσεις μετα-επεξεργασίας των συναγερμών αυτών χρησιμοποιώντας είτε έτοιμα εργαλεία που θα του παρέχει η εφαρμογή είτε εργαλεία που θα μπορεί να κατασκευάσει μόνος του. Η εφαρμογή θα λειτουργεί σε παραθυρικό περιβάλλον. Η πλατφόρμα θα πρέπει να προσφέρει δυνατότητες λήψης δεδομένων από εξωτερικές πηγές, επεξεργασίας των δεδομένων βάση συγκεκριμένων μεθόδων και εκτίμησης της ποιότητας των αποτελεσμάτων βάση συγκεκριμένων κριτηρίων. Όλα αυτά θα γίνονται από τον χρήστη με συνδυασμό των κατάλληλων components. Έτσι ο χρήστης μπορεί να σχεδιάσει μία απλή λύση μετα-επεξεργασίας συναγερμών βάση των εργαλείων της πλατφόρμας αρκετά εύκολα και να υλοποιήσει μία πιο πολύπλοκη λύση απλά γράφοντας επιπλέον τον απαραίτητο κώδικα μόνο για τις μεθόδους που δεν περιλαμβάνονται στην πλατφόρμα. Ο υποψήφιος για την ανάληψη της πτυχιακής θα πρέπει να έχει εμπειρία ανάπτυξης εφαρμογών σε JAVA καθώς και σχετική αντίληψη των βασικών εννοιών της ασφάλειας υπολογιστικών συστημάτων.

Κεφάλαιο 1. Ασφάλεια Υπολογιστών Συστημάτων

Η ασφάλεια των υπολογιστών συστημάτων είναι ένας κλάδος της τεχνολογίας γνωστή και ως «ασφάλεια πληροφοριών» η οποία εφαρμόζεται σε υπολογιστές και δίκτυα. Ο στόχος της ασφάλειας των υπολογιστικών συστημάτων περιλαμβάνει την προστασία των πληροφοριών και της ιδιοκτησίας από κλοπή, δωροδοκία ή την φυσική καταστροφή, επιτρέποντας παράλληλα οι πληροφορίες αυτές και οι αντίστοιχες ιδιοκτησίες, να είναι προσιτές και προσβάσιμες σε συγκεκριμένους-εξουσιοδοτημένους χρήστες.

Ο όρος «ασφάλεια υπολογιστικών συστημάτων», περιλαμβάνει τις συλλογικές διαδικασίες και τους μηχανισμούς με τους οποίους οι ευαίσθητες και πολύτιμες πληροφορίες και οι διάφορες υπηρεσίες, προστατεύονται από την δημοσιοποίησή τους, από την αλλοίωση τους ή την καταστροφή τους από αναρμόδιες-μη εξουσιοδοτημένες δραστηριότητες ή άτομα αντίστοιχα.

1.1 Αρχιτεκτονική συστημάτων ασφαλείας

Η ασφάλεια των υπολογιστικών συστημάτων βασίζεται στην λογική. Δεδομένου ότι η ασφάλεια δεν είναι απαραιτήτως ο αρχικός στόχος των περισσότερων εφαρμογών, ο σχεδιασμός ενός προγράμματος με κύριο στόχο την ασφάλεια επιβάλλει συχνά περιορισμούς στο αντίστοιχο πρόγραμμα.

Υπάρχουν 4 προσεγγίσεις για την ασφάλεια των υπολογιστικών συστημάτων (μερικές φορές ο σωστός συνδυασμός τους είναι αποδεκτός):

1. Εμπιστοσύνη στο λογισμικό για να τηρηθεί μια πολιτική ασφαλείας αλλά το λογισμικό δεν είναι αξιόπιστο (αυτό οδηγεί σε αβεβαιότητα της ασφάλειας υπολογιστικών συστημάτων).
2. Εμπιστοσύνη στο λογισμικό για να τηρηθεί μια πολιτική ασφαλείας και το λογισμικό επικυρώνεται ως αξιόπιστο (παραδείγματος χάριν από την χρονοβόρα ανάλυση κλάδων και πορειών).
3. Καμία εμπιστοσύνη σε κανένα λογισμικό αλλά επιβολή μιας πολιτική ασφαλείας με μηχανισμούς που δεν είναι αξιόπιστοι (πάλι αυτό οδηγεί σε αβεβαιότητα της ασφάλειας υπολογιστικών συστημάτων).
4. Καμία εμπιστοσύνη σε κανένα λογισμικό αλλά επιβολή μιας πολιτική ασφαλείας με αξιόπιστους μηχανισμούς.

Πολλά συστήματα έχουν οδηγηθεί όχι σκόπιμα στην πρώτη προσέγγιση. Δεδομένου ότι η δεύτερη προσέγγιση απαιτεί αρκετά χρήματα η χρήση της είναι πολύ περιορισμένη. Όπως και η πρώτη προσέγγιση έτσι και η τρίτη οδηγούν σε αποτυχία των συστημάτων ασφαλείας. Επειδή η τελευταία μέθοδος για την ασφάλεια των υπολογιστικών συστημάτων βασίζεται κυρίως σε μηχανισμούς υλικού και αποφεύγει τον περιορισμό της «ελευθερίας» των χρηστών, είναι πρακτικότερη.

Υπάρχουν διάφορες στρατηγικές και τεχνικές που χρησιμοποιούνται για να σχεδιάσουν τα συστήματα ασφαλείας. Εντούτοις υπάρχουν ελάχιστες, αν όχι καθόλου, αποτελεσματικές στρατηγικές για να ενισχύσουν την ασφάλεια ενός συστήματος μετά από τον σχέδιο του. Μια τεχνική επιβάλλει την αρχή των «ελαχίστων προνομίων», όπου μια οντότητα έχει μόνο τα προνόμια που απαιτούνται για τη λειτουργία του. Με αυτό τον τρόπο ακόμα κι αν ένας εισβολέας αποκτήσει πρόσβαση σε ένα μέρος του συστήματος, αυτού του είδους η ασφάλεια εξασφαλίζει ότι είναι εξίσου δύσκολο για αυτόν να έχει πρόσβαση στο υπόλοιπο του συστήματος.

Επιπλέον, με την διάσπαση του συστήματος σε μικρότερα συστατικά μέρη, η πολυπλοκότητα των επιμέρους αυτών συστατικών μειώνεται, παρέχοντας την δυνατότητα σε διάφορες τεχνικές-μεθόδους να αξιολογήσουν την ορθότητα κρίσιμων υποσυστημάτων του λογισμικού.

Κατά τον σχεδιασμό της ασφάλειας ενός συστήματος πρέπει να χρησιμοποιήσει η "σε βάθος άμυνα" (defense in dept) , όπου περισσότερα από ένα υποσυστήματα θα πρέπει να παραβιαστούν για να παραβιάσουν την ακεραιότητα του συστήματος και των πληροφοριών που διαθέτει. Η «σε βάθος άμυνα» λειτουργεί σωστά όταν η παραβίαση της ασφάλειας σε ένα επίπεδο δεν υπομονεύει την ακεραιότητα της ασφάλειας του επόμενου επιπέδου. Επίσης, η αρχή του «cascading» δεν συνεπάγεται πως τα διάφορα “χαμηλά εμπόδια” σε κάθε επίπεδο αποτελούν ένα ενιαίο “υψηλό εμπόδιο”. Έτσι η αρχή του «cascading» δεν παρέχει την ασφάλεια ενός ενιαίου ισχυρότερου μηχανισμού.

Τέλος, η ασφάλεια των υπολογιστικών συστημάτων δεν θα πρέπει να λαμβάνεται πλήρως υπόψη ή καθόλου. Οι σχεδιαστές και οι χειριστές των συστημάτων πρέπει να υποθέσουν ότι οι παραβιάσεις της ασφαλείας είναι αναπόφευκτες. Επιπλέον τα συστήματα ελέγχου πρέπει πάντα να είναι ενεργά ακόμα και όταν εμφανίζεται κάποια παραβίαση στο σύστημα, έτσι ώστε όταν εμφανίζεται αυτή η παραβίαση της ασφάλειας, να μπορεί να καθοριστεί ο μηχανισμός της και το μέγεθος της από τα συστήματα ελέγχου.

1.2 Απειλές σε συστήματα ασφαλείας

Από τη φύση τους, τα διάφορα συστήματα ηλεκτρονικών υπολογιστών συγκεντρώνουν μια σειρά ευπαθειών. Υπάρχουν ανθρώπινες ευπάθειες οι οποίες μπορούν σκόπιμα ή όχι να διακινδυνεύσουν το αντίστοιχο σύστημα προστασίας πληροφοριών. Οι ευπάθειες υλικού μοιράζονται μεταξύ του υπολογιστή, των εγκαταστάσεων επικοινωνίας, και των απομακρυσμένων μονάδων και των κονσόλων. Υπάρχουν ευπάθειες στο λογισμικό σε όλα τα επίπεδα του λειτουργικού συστήματος των υπολογιστικών συστημάτων και υπάρχουν

ευπάθειες στην οργάνωση του συστήματος προστασίας (π.χ., στον έλεγχο πρόσβασης, στην αναγνώριση χρηστών και την επικύρωσή τους, κ.λπ.). Η σοβαρότητα αυτών των ευπαθειών εξαρτάται από την ευαισθησία των πληροφοριών που χειρίζονται, την κατηγορία των χρηστών, τις υπολογιστικές δυνατότητες που είναι διαθέσιμες στον χρήστη, το λειτουργικό περιβάλλον, την παραμετροποίηση του συστήματος, και τις ικανότητες των πιθανών επιτιθεμένων στο σύστημα.

Αυτά τα σημεία ευπάθειας υπάρχουν και στα εταιρικά περιβάλλοντα που χειρίζονται τις διάφορες ιδιόκτητες πληροφορίες και στις κυβερνητικές εγκαταστάσεις που επεξεργάζονται απόρρητα στοιχεία.

1.3 Τύποι ευπαθειών των συστημάτων ασφαλείας

Ο σχεδιασμός ενός ασφαλούς συστήματος πρέπει να παρέχει προστασία ενάντια σε διάφορους τύπους ευπαθειών. Αυτοί εμπίπτουν σε τρεις σημαντικές κατηγορίες: τυχαίες κοινοποιήσεις, σκόπιμες διεισδύσεις, και φυσική επίθεση.

Τυχαία κοινοποίηση (Accidental Disclosure).

Μια αποτυχία των συστατικών, του εξοπλισμού, του λογισμικού ή του υποσυστήματος, ίσως να έχει ως συνέπεια την έκθεση ευαίσθητων πληροφοριών ή την παραβίαση οποιουδήποτε στοιχείου του συστήματος. Οι τυχαίες κοινοποιήσεις είναι συχνά αποτέλεσμα αποτυχιών του υλικού ή του λογισμικού.

Τέτοιες αποτυχίες μπορούν να περιλάβουν τη σύζευξη των πληροφοριών από έναν χρήστη (ή το λογισμικό ενός υπολογιστή) με άλλες πληροφορίες από άλλο χρήστη (clobbering of information), καθιστώντας τα αρχεία ή τα προγράμματα ακατάλληλα προς χρήση, την καταπάτηση ή την καταστρατήγηση των μέτρων ασφάλειας, ή την απρόβλεπτη αλλαγή της ασφάλειας των χρηστών, των αρχείων, ή των τερματικών σταθμών. Οι τυχαίες-λανθασμένες κοινοποιήσεις μπορούν επίσης να εμφανιστούν από ανάρμοστες ενέργειες του συστήματος ή λόγω της λανθασμένης συντήρησής τους.

Σκόπιμη διείσδυση (Deliberate Penetration).

Μια σκόπιμη και συγκεκαλυμμένη προσπάθεια αποσκοπεί στο:

- Να λάβει πληροφορίες που περιλαμβάνονται στο σύστημα
- Να αναγκάσει το σύστημα να λειτουργεί προς όφελος του “εισβολέα”
- Να καταστήσει το σύστημα αναξιόπιστο ή ακατάλληλο για χρήση από τον νόμιμο χρήστη του.

Οι σκόπιμες προσπάθειες να διαπεραστούν τα συστήματα ασφαλείας μπορούν είτε να είναι ενεργές είτε παθητικές. Οι *παθητικές* μέθοδοι περιλαμβάνουν την παγίδευση των καλωδίων και τον έλεγχο των μαγνητικών εκπομπών. Η *ενεργός* διήθηση είναι μια

προσπάθεια εισαγωγής στο σύστημα έτσι ώστε να ληφθούν στοιχεία από τα αρχεία ή για να παρεμποδιστούν διάφορα αρχεία ή ακόμα και το σύστημα.

***Ενεργός διήθηση (*Active Infiltration)**

Μια μέθοδος για να πραγματοποιηθεί η ενεργός διήθηση είναι από έναν εξουσιοδοτημένο χρήστη να διαπεράσει τα στοιχεία του συστήματος για το οποίο δεν έχει καμία έγκριση. Το πρόβλημα του σχεδιασμού είναι η παρεμπόδιση της πρόσβασης στα αρχεία από κάποιον που γνωρίζει τους μηχανισμούς ελέγχου πρόσβασης και που έχει τη γνώση και την επιθυμία να διαμορφώσει τα παραπάνω στις ανάγκες του. Παραδείγματος χάριν, εάν οι κωδικοί πρόσβασης σε ένα σύστημα είναι όλοι τετραψήφιοι αριθμοί, ένας χρήστης μπορεί να επιλέξει οποιοδήποτε τετραψήφιο αριθμό, και έπειτα, αποκτώντας πρόσβαση σε κάποιο αρχείο, να αρχίσει να επεξεργάζεται το περιεχόμενο του.

Μια άλλη κατηγορία τεχνικών ενεργής διήθησης περιλαμβάνει την εκμετάλλευση των σημείων εισόδων trap-door[1] του συστήματος που παρακάμπτουν τα σημεία ελέγχου και επιτρέπουν την άμεση πρόσβαση στα αρχεία. Τα σημεία εισόδων trap-door συχνά δημιουργούνται σκόπιμα κατά τη διάρκεια του σχεδιασμού και ανάπτυξης για να απλοποιήσουν εξουσιοδοτημένες αλλαγές του προγράμματος από τους νόμιμους προγραμματιστές του συστήματος με την προϋπόθεση του κλεισίματος της trap-door πριν από την κάθε τους χρήση. Επιπλέον τα σημεία trap-door μπορούν να δημιουργηθούν από έναν προγραμματιστή συστημάτων προκειμένου να εκτελεί ευκολότερα ελέγχους εσωτερικής ασφαλείας. Παρόλα αυτά, τα σημεία αυτά συνήθως δεν δημιουργούνται σκόπιμα αλλά λόγω του ελλιπούς σχεδιασμού του συστήματος. Για παράδειγμα είναι δυνατό να βρεθεί ένας ασυνήθιστος συνδυασμός μεταβλητών ελέγχου συστημάτων που θα δημιουργήσει μια πορεία εισόδων που θα διαπερνά μερικά ή και όλα τα μέτρα προστασίας.

Ένας άλλος πιθανός τρόπος ενεργής διήθησης είναι η χρήση ενός πρόσθετου τερματικού που "δένεται" παράνομα στο σύστημα επικοινωνιών. Ένα τέτοιο τερματικό μπορεί να χρησιμοποιηθεί για να παρεμποδιστούν πληροφορίες που ρέουν μεταξύ ενός νόμιμου τερματικού και του κεντρικού επεξεργαστή. Παραδείγματος χάριν, ένα εξουσιοδοτημένο «sign-off» ενός χρήστη μπορεί να παρεμποδιστεί και να ακυρωθεί και έπειτα το παράνομο τερματικό μπορεί να αναλάβει την αλληλεπίδραση με τον επεξεργαστή. Ή ακόμη ένα παράνομο τερματικό μπορεί να διατηρεί επικοινωνία με τον σύστημα κατά τη διάρκεια των περιόδων όπου ο νόμιμος χρήστης είναι ανενεργός αλλά ακόμα διατηρεί ανοικτό κανάλι με το σύστημα.

Παθητική «ανατροπή» (Passive Subversion)

Στην παθητική ανατροπή, εφαρμόζονται διάφορα μέσα προκειμένου να ελέγχουν πληροφορίες που εδρεύουν μέσα στο σύστημα ή πληροφορίες που διαβιβάζονται στα κανάλια επικοινωνίας χωρίς οποιαδήποτε προσπάθεια να παρεμποδιστεί η λειτουργία του συστήματος. Η προφανέστερη μέθοδος παθητικής διήθησης είναι η παγίδευση καλωδίων. Εάν οι επικοινωνίες μεταξύ απομακρυσμένων τερματικών και του κεντρικού επεξεργαστή

πραγματοποιούνται με μη προστατευμένα κυκλώματα, η μέθοδος για την παγίδευση των καλωδίων είναι παρόμοια με την μέθοδο παρακολούθησης ενός τηλεφωνικού κυκλώματος. Είναι επίσης δυνατή η παρακολούθηση των ηλεκτρομαγνητικών εκπομπών που ακτινοβολούνται από τα μεγάλα ηλεκτρονικά κυκλώματα που χαρακτηρίζουν μεγάλος μέρος των υπολογιστικών συστημάτων. Η ενέργεια που εκπέμπεται με αυτήν την μορφή επικοινωνίας μπορεί να καταγραφεί χωρίς να είναι απαραίτητη η φυσική πρόσβαση στο σύστημα ή σε οποιαδήποτε από τα διάφορα μέρη του ή στα κανάλια επικοινωνίας.

1.4 Μηχανισμοί υλικού που προστατεύουν υπολογιστές και δεδομένα

Ασφάλεια βασισμένη στο υλικό ή βοηθητική ασφάλεια υπολογιστών αποτελεί μια εναλλακτική λύση για την ασφάλεια υπολογιστών που βασίζεται μόνο σε διάφορα λογισμικά. Οι συσκευές όπως τα dongles μπορούν να θεωρηθούν ασφαλέστερες λόγω της φυσικής πρόσβασης που απαιτείται προκειμένου να διαμορφωθούν.

Ενώ πολλές λύσεις ασφάλειας που βασίζονται στο λογισμικό κρυπτογραφούν τα στοιχεία για να αποτρέψουν τα στοιχεία από την κλοπή, ένα κακόβουλο πρόγραμμα ή ένας χάκερ μπορεί να αλλοιώσει τα στοιχεία προκειμένου να τα καταστήσει μη επανεκτήσιμα ή ακατάλληλα προς χρήση. Ομοίως, τα μερικά κρυπτογραφημένα λειτουργικά συστήματα μπορούν να αλλοιωθούν από ένα κακόβουλο πρόγραμμα ή έναν χάκερ, και έτσι το αντίστοιχο σύστημα να είναι πλέον ακατάλληλο προς χρήση. Οι βασισμένες στο υλικό λύσεις ασφάλειας μπορούν να αποτρέψουν την ανάγνωση και την επεξεργασία σε διάφορα δεδομένα και ως εκ τούτου προσφέρουν πολύ ισχυρή προστασία ενάντια σε αναρμόδια πρόσβαση.

Ασφάλεια των εργασιών βασισμένη στο υλικό: Μια συσκευή επιτρέπει σε έναν χρήστη να συνδεθεί, αποσυνδεθεί και να θέσει διαφορετικά επίπεδα προνομίων με χειρωνακτικές ενέργειες. Η συσκευή χρησιμοποιεί τη βιομετρική τεχνολογία για να αποτρέψει τους κακόβουλους χρήστες από το να συνδεθούν, αποσυνδεθούν, και να μεταβάλλουν τα διάφορα προνόμια των χρηστών. Η τρέχουσα κατάσταση ενός χρήστη της συσκευής ελέγχεται και από έναν υπολογιστή και από τους “ελεγκτές” των περιφερειακών συσκευών όπως οι σκληροί δίσκοι. Η παράνομη πρόσβαση από έναν κακόβουλο χρήστη ή ένα κακόβουλο πρόγραμμα διακόπτεται λαμβάνοντας υπόψη την τρέχουσα κατάσταση ενός χρήστη από το σκληρό δίσκο και τα DVD, καθιστώντας έτσι την παράνομη πρόσβαση στα δεδομένα αδύνατη. Η ασφάλεια των συστημάτων που βασίζεται στο υλικό είναι αποδοτικότερη. Αυτές οι συσκευές προστατεύουν τα προνόμια και την ακεραιότητα των λειτουργικών συστημάτων. Επομένως, ένα απολύτως ασφαλές σύστημα μπορεί να δημιουργηθεί χρησιμοποιώντας έναν συνδυασμό συστημάτων ασφαλείας που αποτελείται από συσκευές ελέγχου και διάφορες πολιτικές διοίκησης των συστημάτων.

Κεφάλαιο 2. Συστήματα ανίχνευσης παρεισφρήσεων

Ένα σύστημα ανίχνευσης παρείσφρησης (IDS) είναι μια συσκευή (ή εφαρμογή) που ελέγχει τις δραστηριότητες των δικτύων ή/και των συστημάτων για τις κακόβουλες δραστηριότητες ή τις παραβιάσεις και συντάσσουν αναφορές σε έναν κεντρικό υπολογιστή. Η ανίχνευση παρεισφρήσεων είναι η διαδικασία λοιπόν με την οποία συλλέγονται δεδομένα από ένα δίκτυο υπολογιστών και αναλύονται προκειμένου να εντοπισθούν πιθανών κακόβουλα γεγονότα, όπως είναι παραβιάσεις ή επικείμενες απειλές της ασφάλειας του δικτύου υπολογιστών ή των τυποποιημένων πρακτικών ασφάλειας.

Το **Snort** είναι ένα ελεύθερο και ανοικτό σύστημα πρόληψης παρεισφρήσεων για δίκτυα υπολογιστών *(network intrusion prevention system – NIPS)* και ένα σύστημα ανίχνευσης παρεισφρήσεων *(network intrusion detection system-NIDS)*, ικανό να αναλύει σε πραγματικό χρόνο την κυκλοφορία δεδομένων σε ένα δίκτυο IP. Το Snort δημιουργήθηκε από τον Martin Roesch και τώρα αναπτύσσεται από την Sourcefire, ιδρυτής της οποία είναι ο Roesch και η CTO. Το Snort αναλύει πρωτόκολλα, αναζητά, αντιστοιχεί περιεχόμενο και χρησιμοποιείται συνήθως για να εμποδίσει ενεργές επιθέσεις ή να ανιχνεύσει επικείμενες επιθέσεις. Παρόλα αυτά το λογισμικό αυτό χρησιμοποιείται συνήθως για λόγους πρόληψης παρεισφρήσεων. Στην συνέχεια αυτού του report υπάρχει μια πιο λεπτομερής ανάλυση στο Snort.

Η ανίχνευση παρείσφρησης είναι ένα σύνολο τεχνικών και μεθόδων που χρησιμοποιούνται για να ανιχνεύσουν ύποπτες δραστηριότητες σε ένα δίκτυο. Τα συστήματα ανίχνευσης παρείσφρησης ταξινομούνται σε δύο κατηγορίες:

1. Στα συστήματα *signature-based intrusion detection*
2. Στα συστήματα *anomaly detection*

Οι εισβολείς έχουν signatures, όπως οι ιοί των υπολογιστών, οι οποίοι μπορούν να ανιχνευθούν από τα διάφορα συστήματα χρησιμοποιώντας διάφορα λογισμικά. Προσπαθούν να εντοπίσουν πακέτα δεδομένων που περιέχουν οποιεσδήποτε γνωστές μη εξουσιοδοτημένες και “φιλικές” signatures ή ανωμαλίες σχετικές με τα διάφορα πρωτόκολλα του διαδικτύου. Βασισμένα σε ένα σύνολο υπογραφών(signatures) και κανόνων, το σύστημα ανίχνευσης είναι σε θέση να βρει και να καταγράψει ύποπτες δραστηριότητες και έπειτα να “αφυπνίσει” τους αρμόδιους φορείς με τα κατάλληλα μηνύματα. Τα προγράμματα της κατηγορίας *anomaly-based intrusion detection* εξετάζουν συνήθως τις ανωμαλίες που παρουσιάζουν τα πακέτα στα πρωτόκολλα του διαδικτύου. Σε μερικές περιπτώσεις αυτές οι μέθοδοι παράγουν καλύτερα αποτελέσματα έναντι των αποτελεσμάτων που παράγουν τα συστήματα της *signature-based intrusion detection*. Συνήθως ένα σύστημα ανίχνευσης παρεισφρήσεων συλλέγει δεδομένα από το δίκτυο και εφαρμόζει σε αυτά τους κανόνες του ή ανιχνεύει τις ανωμαλίες που βρίσκονται σε αυτά τα δεδομένα. Το Snort είναι πρώτιστα ένα σύστημα ανίχνευσης παρεισφρήσεων βασισμένο στους κανόνες, εντούτοις διαθέτει *input plug-ins* προκειμένου να ανιχνεύει ανωμαλίες στις επιγραφές των πρωτοκόλλων.

2.1 Εισαγωγή στην ανίχνευση παρεισφρήσεων και το Snort

Το Snort κάνει την χρήση κανόνων που είναι αποθηκευμένοι σε αρχεία κειμένων που μπορούν να τροποποιηθούν από έναν συντάκτη κειμένων. Οι κανόνες αυτοί ομαδοποιούνται σε διάφορες κατηγορίες. Οι κανόνες της κάθε κατηγορίας αποθηκεύονται σε ξεχωριστά αρχεία. Αυτά τα αρχεία έπειτα συμπεριλαμβάνονται σε ένα κύριο αρχείο, το snort.conf. Το Snort διαβάζει αυτούς τους κανόνες κατά την εκκίνηση του και διαμορφώνει τις εσωτερικές δομές δεδομένων ή "τις αλυσίδες" προκειμένου στην συνέχεια να εφαρμόσει αυτούς τους κανόνες στα δεδομένα που θα εξετάσει.

Η εύρεση των υπογραφών και η χρήση τους στους κανόνες είναι μια απαιτητική εργασία, δεδομένου ότι όσο περισσότεροι κανόνες χρησιμοποιούνται, τόσο περισσότερη επεξεργαστική ισχύς απαιτείται για να εξεταστούν τα δεδομένα που θα συλλέξει το σύστημα, σε πραγματικό χρόνο. Είναι σημαντικό να εφαρμοστούν τόσες υπογραφές όσες κάποιος χρειάζεται αλλά με όσο το δυνατόν λιγότερους κανόνες. Το Snort έρχεται με ένα πλούσιο σύνολο προκαθορισμένων κανόνων για να ανιχνεύσει την δραστηριότητα παρεισφρήσεων ενώ δίνεται παράλληλα η δυνατότητα να προστεθούν και άλλοι κανόνες. Υπάρχει επίσης και η δυνατότητα να διαγραφούν – απενεργοποιηθούν μερικοί από τους ενσωματωμένους κανόνες για να αποφευχθούν λανθασμένες ειδοποιήσεις.

2.2 Πολιτική των IDS

Προτού να εγκατασταθεί το σύστημα ανίχνευσης παρεισφρήσεων στο δίκτυό, θα πρέπει να υπάρχει μια πολιτική για την ανίχνευση των επιθέσεων και μια πολιτική για τα μέτρα που θα πρέπει να ληφθούν όταν εντοπιστεί μία επίθεση. Μια πολιτική είναι απαραίτητο να «υπαγορεύσει» τους κανόνες του IDS και πώς αυτού θα εφαρμοστούν. Μία ορθή και γενικευμένη πολιτική των IDS θα πρέπει να περιέχει τουλάχιστον τα ακόλουθα:

1. Ποιος θα ελέγχει το IDS; Ανάλογα με το IDS, οι ειδοποιήσεις διαφέρουν για τις ίδιες επιθέσεις σε ένα σύστημα. Αυτές οι ειδοποιήσεις μπορεί να σε μορφή απλών αρχείων κειμένων, ή μπορεί να είναι περισσότερο περίπλοκες ή ενσωματωμένες σε συστήματα διαχείρισης δικτύων όπως σε κάποια βάση δεδομένων OpenView ή MySQ. Παρόλα αυτά απαιτείται κάποιος για να ελέγχει την δραστηριότητα των εισβολέων και πρέπει να καθοριστεί ένα αρμόδιο πρόσωπο για αυτήν την δραστηριότητα. Επίσης οι επιθέσεις μπορούν να ελεγχθούν σε πραγματικό χρόνο χρησιμοποιώντας pop-up windows ή web interfaces. Σε αυτήν την περίπτωση οι χρήστες πρέπει να έχουν τη γνώση για την ερμηνεία και την αντιμετώπιση αυτών των μηνυμάτων.

2. Ποίος θα συντηρεί το σύστημα; Όπως με όλα τα συστήματα, έτσι και με τα IDS πρέπει να καθιερωθούν περιοδικοί έλεγχοι στο συστήματα από τους αρμόδιους φορείς.

3. Ποιος θα χειρίζεται τις ειδοποιήσεις και πώς; Επίσης σε περίπτωση που δεν υπάρχει κίνδυνος επιθέσεων δεν είναι απαραίτητη η χρήση ενός IDS.
4. Ποια θα είναι η διαδικασία κλιμάκωσης (επίπεδο 1, επίπεδο 2 και ούτω καθεξής); Η διαδικασία κλιμάκωσης είναι βασική στρατηγική για την αντιμετώπιση μιας επίθεσης. Η πολιτική αυτή επίσης πρέπει να περιγράφει σαφώς ποιες από τις ειδοποιήσεις θα πρέπει να μεταφερθούν σε ανώτατους διαχειριστές

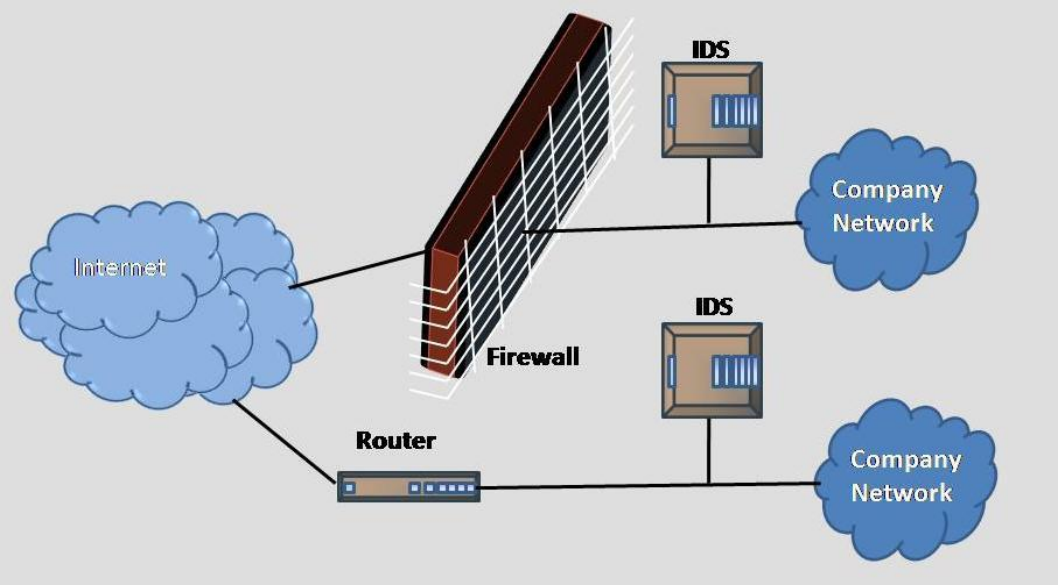
5. Υποβολή αναφορών. Οι εκθέσεις-αναφορές θα πρέπει να δημιουργούνται ανά τακτά χρονικά διαστήματα (κάθε μέρα, εβδομάδα κλπ) προκειμένου να υπάρχει μια πλήρης εικόνα των κινδύνων του συστήματος.

6. Αναπροσαρμογές των υπογραφών (*Signature updates*). Οι χάκερ δημιουργούν συνεχώς νέους τύπους επιθέσεων. Αυτές οι επιθέσεις ανιχνεύονται από τα IDS μόνο εάν το σύστημα γνωρίζει τις υπογραφές των νέων αυτών επιθέσεων. Λόγω της συχνής μεταβολής της φύσης των επιθέσεων, πρέπει να ενημερώνεται συχνά το σύστημα ανίχνευσης παραφράσεων με νέες υπογραφές και κανόνες.

2.3 Θέση ενός IDS στην τοπολογία ενός δικτύου

Ανάλογα με την τοπολογία ενός δικτύου, μπορεί κανείς να τοποθετήσει ένα σύστημα ανίχνευσης παρεισφρήσεων σε μία ή περισσότερες θέσεις. Εξαρτάται επίσης από το ποιους τύπους παρεισφρήσεων θέλει κανείς να ανιχνεύσει: εσωτερικές, εξωτερικές ή και τα δύο είδη. Παραδείγματος χάριν, εάν θέλετε να ανιχνεύσετε μόνο τις εξωτερικές δραστηριότητες παρείσφρησης, και έχετε μόνο έναν δρομολογητή που συνδέει με το σύστημα με το διαδίκτυο, τότε η καλύτερη θέση για ένα IDS μπορεί να είναι ακριβώς “μέσα” στο δρομολογητή ή το firewall. Εάν έχετε πολλαπλάσιες συνδέσεις διαδικτύου, τότε ίσως η καλύτερη λύση για να τοποθετηθεί ένα IDS να είναι σε κάθε σημείου εισόδου στο διαδίκτυο.

Εντούτοις ίσως να χρειάζεται να ανιχνεύονται οι εσωτερικές επιθέσεις του συστήματος επομένως ίσως να είναι απαραίτητο να τοποθετηθεί ένα IDS σε κάθε τομέα του δικτύου. Άλλες φορές πάλι ίσως δεν είναι απαραίτητο (ούτε και να πρέπει) να τοποθετηθεί ένα σύστημα ανίχνευσης παρεισφρήσεων σε κάθε δικτυακό τομέα αλλά στις ευαίσθητες περιοχές του. Αξίζει να σημειωθεί ότι όσο περισσότερα IDS υπάρχουν τόσο περισσότερη εργασία χρειάζεται καθώς και το κόστος συντήρησης αυξάνεται.



2.4 Ορολογία συστημάτων παρεισφρήσεων

Προτού να προχωρήσουμε στις λεπτομέρειες της ανίχνευσης παρεισφρήσεων και του Snort, πρέπει να αναλύσουμε μερικούς από τους βασικούς ορισμούς που χρησιμοποιούνται συχνά στα IDS (Intrusion Detection Systems).

IDS (Intrusion Detection Systems)

Το σύστημα ανίχνευσης παρεισφρήσεων ή IDS είναι ένα λογισμικό, υλικό ή συνδυασμός και των δυο που χρησιμοποιείται για να ανιχνεύσει την δραστηριότητα εισβολέων. Το Snort είναι μια ανοικτή εφαρμογή IDS διαθέσιμη στο ευρύ κοινό. Ένα IDS μπορεί να έχει διαφορετικές ικανότητες ανάλογα με το πόσο σύνθετο είναι αυτό και τα συστατικά του μέρη. Οι συσκευές IDS που είναι ένας συνδυασμός υλικού και λογισμικού είναι διαθέσιμες από πολλές εταιρίες. Όπως αναφέρθηκε και νωρίτερα, ένα IDS μπορεί να χρησιμοποιεί signatures, anomaly-based techniques ή και τα δύο.

Δικτυακό (Network) IDS ή NIDS

Το NIDS είναι συστήματα ανίχνευσης παρεισφρήσεων που «συλλαμβάνει» πακέτα δεδομένων που μεταδίδονται μέσα στα δίκτυα (ενσύρματα, ασύρματα) και τα αντιστοιχεί με μια βάση δεδομένων των υπογραφών τους. Ανάλογα με το εάν ένα πακέτο αντιστοιχίζεται με μια υπογραφή εισβολέων (intruder signature), τότε ειδοποιούνται τα κατάλληλα στελέχη. Διαφορετικά το πακέτο καταγράφεται σε ένα αρχείο ή μια βάση δεδομένων. Μια σημαντική χρήση του Snort είναι και η λειτουργία του ως NIDS.

Host IDS ή HIDS

Τα συστήματα Τα Host-based intrusion detection ή HIDS εγκαθίστανται ως “πράκτορες” σε έναν host. Αυτά τα συστήματα ανίχνευσης παρεισφρήσεων μπορούν να εξετάσουν τα αρχεία των συστημάτων και των εφαρμογών για να ανιχνεύσουν οποιαδήποτε δραστηριότητα εισβολέων. Μερικά από αυτά τα συστήματα είναι αντιδραστικά, που σημαίνει ότι ειδοποιούν για οποιαδήποτε δραστηριότητα τους μόνο όταν κάτι έχει συμβεί. Μερικά HIDS είναι δυναμικά: μπορούν ελέγχουν την κυκλοφορία των δικτύων και των πακέτων που μεταδίδονται στον κόμβο στον οποίο το HIDS είναι εγκατεστημένο και να ειδοποιεί με αυτό τον τρόπο σε πραγματικό χρόνο.

Υπογραφές (Signatures)

Η υπογραφή είναι το σχέδιο(pattern) που αναζητείται μέσα σε ένα πακέτο δεδομένων. Μια υπογραφή χρησιμοποιείται προκειμένου να ανιχνευτούν διάφοροι τύποι επιθέσεων. Παραδείγματος χάριν, η παρουσία scripts/iisadmin” σε ένα πακέτο δεδομένων που κατευθύνεται προς κάποιο κεντρικό υπολογιστή ενός δικτύου μπορεί να δείξει κάποιες από τις δραστηριότητες των εισβολέων.

Οι υπογραφές μπορούν να είναι παρούσες σε διαφορετικά μέρη ενός πακέτου δεδομένων ανάλογα με την φύση της επίθεσης. Παραδείγματος χάριν, μπορείτε να βρείτε τις υπογραφές στην διεύθυνση IP, στο πρωτόκολλο TCP UDP. Συνήθως τα IDS βασίζονται στις υπογραφές για να αναγνωρίσουν την δραστηριότητα των εισβολέων-επιθέσεων. Μερικά IDS από διάφορους προμηθευτές ενημερώνονται και προσθέτουν αυτόματα νέες υπογραφές όταν ένας νέος τύπος επίθεσης ανακαλύπτεται. Σε άλλα IDS, όπως το Snort, η ενημέρωση δεν γίνεται αυτόματα αλλά από τον διαχειριστή του συστήματος

Ειδοποιήσεις (Alerts)

Οι ειδοποιήσεις είναι οποιοδήποτε είδος ανακοίνωσης προς τους χρήστες και τους αρμόδιους φορείς για διάφορες επιθέσεις που αναγνωρίζονται σε ένα σύστημα. Όταν ένα IDS ανιχνεύει μία επίθεση, τότε αυτό ενημερώσει τον διαχειριστή ασφάλειας με αυτές τις ειδοποιήσεις. Αυτές οι ειδοποιήσεις αποθηκεύονται επίσης σε αρχεία συμβάντων (log) ή σε βάσεις δεδομένων όπου μπορούν να επεξεργαστούν αργότερα από τους ειδικούς ασφάλειας. Το Snort μπορεί να παραγάγει τέτοιου είδους ειδοποιήσεις ενώ έχει παράλληλα την δυνατότητα να τις στείλει σε πολλαπλούς προορισμούς. Για παράδειγμα, είναι δυνατό να καταγραφούν τα alerts σε μια βάση δεδομένων ενώ παράλληλα να σταλούν και emails με αυτές τις ειδοποιήσεις.

Συμβάντα (Logs)

Τα log messages σώζονται συνήθως σε κάποιο αρχείο. Εξ ορισμού το Snort αποθηκεύει αυτά τα μηνύματα στον κατάλογο του `/var/log/snort`. Εντούτοις, η θέση που αποθηκεύονται αυτά τα μηνύματα μπορεί να αλλάξει κατά την εκκίνηση του Snort, από την γραμμή εντολών του. Τα log messages μπορούν να αποθηκευτούν σε μορφή κειμένου είτε

σε δυαδική μορφή. Τα δυαδικά αρχεία μπορούν έπειτα να επεξεργαστούν χρησιμοποιώντας το λογισμικό «tcpdump».
Ένα καινούργιο λογισμικό αποκαλούμενο Barnyard είναι επίσης διαθέσιμο προκειμένου να αναλύει τα δυαδικά αρχεία που παράγει το Snort. Τέλος να σημειωθεί ότι τα αρχεία των log messages σε δυαδική μορφή είναι γενικώς αποδοτικότερα.

False Alarms

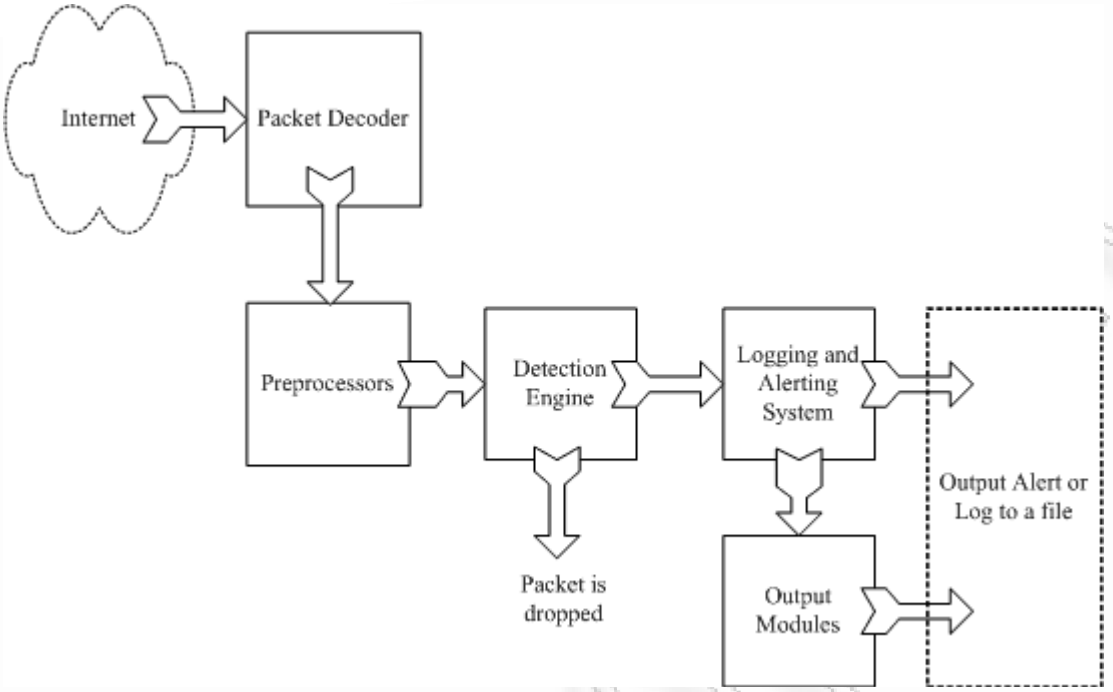
Τα False Alarms παράγονται λόγω μιας λανθασμένης ένδειξης επιθέσεων στο σύστημα. Παραδείγματος χάριν, εσωτερικοί hosts μπορεί μερικές φορές να μεταδώσουν μηνύματα τα οποία να "αφυπνίσουν" το σύστημα χωρίς λόγο. Μερικοί δρομολογητές, οι Linksys, παράγουν πολλά alerts UPnP λανθασμένα. Για να αποφύγετε τις ψεύτικες αυτές ειδοποιήσεις, πρέπει να τροποποιήσετε και να συντονίστε τους διαφορετικούς κανόνες προεπιλογής. Σε μερικές περιπτώσεις ίσως να χρειάζεται να απενεργοποιηθούν εντελώς κάποιοι από αυτούς τους κανόνες προκειμένου να αποφευχθούν αυτού του είδους οι συναγερμοί.

2.5 Συστατικά μέρη του Snort

Η λειτουργία του Snort διαιρείται σε 5 μέρη . Σε αυτές τις κατηγορίες υπάρχουν ενέργειες που λειτουργούν όλες μαζί προκειμένου να ανιχνεύουν επιθέσεις στο δίκτυο. Τα μέρη από τα οποία απαρτίζεται το Snort είναι τα ακόλουθα:

- Αποκωδικοποιητής πακέτων
- Preprocessors
- Μηχανή ανίχνευσης
- Σύστημα καταγραφής και ειδοποιήσεων
- Output Modules

Το παρακάτω σχήμα επιδεικνύει πώς αυτά τα συστατικά μέρη του Snort κατανέμονται. Τα εισερχόμενα πακέτα δεδομένων από το διαδίκτυο εισάγονται αρχικά στον αποκωδικοποιητή πακέτων και στην συνέχεια το κάθε πακέτο δεδομένων επεξεργάζεται από το κάθε κομμάτι του Snort όπως φαίνεται παρακάτω.



Στην συνέχεια παρουσιάζεται μια συνοπτική εισαγωγή στα 5 μέρη του Snort:

1. ***Αποκωδικοποιητής πακέτων***

   Ο αποκωδικοποιητής πακέτων παίρνει τα πακέτα από τους διαφορετικούς τύπους δικτύων (ενσύρματα και ασύρματα) και τα προετοιμάζει έτσι ώστε να μπορούν να επεξεργαστούν ή να σταλούν στη μηχανή ανίχνευσης.

2. ***Preprocessors***

   Preprocessors είναι "συστατικά" που μπορούν να χρησιμοποιηθούν με το Snort για να τροποποιήσουν τα διάφορα πακέτα δεδομένων προτού η μηχανή ανίχνευσης αναλάβει δράση για να ανιχνεύσει τυχόν επιθέσεις. Επιπλέον μερικοί από τους preprocessors σε ένα IDS έχουν και αυτοί την δυνατότητα να ειδοποιήσουν τους διαχειριστές του συστήματος για τυχόν επιθέσεις. Οι Preprocessors είναι πολύ σημαντικοί για οποιοδήποτε IDS προκειμένου να προετοιμάσει τα πακέτα δεδομένων που προορίζονται για ανάλυση από τις μηχανές ανίχνευσης. Οι χάκερς χρησιμοποιούν διάφορες τεχνικές για να "ξεγελάσουν» ένα IDS. Για παράδειγμα, σε κάποιο σύστημα ίσως υπάρχει ένας κανόνας για την ανίχνευση της υπογραφής, π.χ. μπορεί να έχει κανείς δημιουργήσει έναν κανόνα για την υπογραφή *"scripts/iisadmin"*. Εάν λοιπόν ο προηγούμενος κανόνας συγκρίνει τις δύο υπογραφές τότε αν ο εισβολέας χρησιμοποιήσει τις υπογραφές:

   - `"scripts/./iisadmin"`
   - `"scripts/examples/../iisadmin"`
   - `"scripts\iisadmin"`
   - `"scripts/.\iisadmin"`

Μπορεί εύκολα να παρακάμψει ένα μέρος της ασφάλειας του συστήματος. Για να περιπλέξουν ακόμη περισσότερο την κατάσταση, οι χάκερς μπορούν επίσης να παρεμβάλουν στο web Uniform Resource Identifier (URI) δεκαεξαδικούς χαρακτήρες ή Unicode χαρακτήρες που είναι απόλυτα εξουσιοδοτημένοι από τους web servers. Αξίζει να σημειωθεί πως οι web servers αναγνωρίζουν τις παραπάνω υπογραφές και έτσι παράγουν ως αποτέλεσμα την επιθυμητή υπογραφή. Δηλαδή την "scripts/iisadmin". Εντούτοις εάν το IDS ψάχνει για ακριβή αντιστοιχία μεταξύ των υπογραφών τότε δεν θα είναι ικανό να εντοπίσει αυτές τις αλλαγές επομένως ούτε και τις αντίστοιχες επιθέσεις.

Οι Preprocessors χρησιμοποιούνται επίσης για την επανένωση πακέτων. Όταν ένα μεγάλο πακέτο δεδομένων μεταφέρεται σε έναν άλλο υπολογιστή μέσω ενός δικτύου, το πακέτο αυτό συνήθως διασπάται σε μικρότερα. Παραδείγματος χάριν, το μέγιστο μήκος ενός οποιοιδήποτε πακέτου σε ένα δίκτυο Ethernet είναι συνήθως 1500 bytes. Αυτή η τιμή ελέγχεται από Transfer Unit (MTU). Αυτό σημαίνει ότι αν σταλεί ένα αρχείο μεγαλύτερο των 1500 bytes, αυτό θα χωριστεί σε πολλαπλάσια πακέτα δεδομένων έτσι ώστε κάθε πακέτο να έχει μέγεθος μικρότερο ή ίσο με 1500bytes. Έτσι το σύστημα το οποίο λαμβάνει αυτά τα πακέτα, τα αναδιαμορφώνει σε ένα ενιαίο πακέτο.

Σε ένα IDS, προτού να γίνει οποιοσδήποτε έλεγχος πρέπει αυτά τα διασπασμένα πακέτα να επανενωθούν έτσι ώστε να είναι δυνατή η ανάλυσή τους. Για παράδειγμα πολλές φορές ίσως, ένα κομμάτι της υπογραφής βρίσκεται σε ένα πακέτο δεδομένων και η υπόλοιπη σε ένα άλλο. Έτσι λοιπόν για να ανιχνευτεί αυτή η υπογραφή σωστά πρέπει πρώτα να συνδυαστούν σωστά όλα τα τμήματα των πακέτων. Οι χάκερς χρησιμοποιούν συχνά την μέθοδο της διάσπασης των πακέτων προκειμένου να εισχωρήσουν πέρα από τα IDS.

Οι Preprocessors στο Snort έχουν την δυνατότητα να επανασυνδέσουν πακέτα δεδομένων, να αποκωδικοποιήσουν το HTTP URI και ούτω καθεξής. Αυτές οι λειτουργίες αποτελούν πολύ σημαντικό μέρος του συστήματος ανίχνευσης παρεισφρήσεων.

3. ***Μηχανή ανίχνευσης (Detection Engine)***

Η μηχανή ανίχνευσης είναι το σημαντικότερο μέρος του Snort. Η ευθύνη της είναι να ανιχνεύει εάν υπάρχει οποιαδήποτε εισβολή σε κάποιο από τα πακέτα που εξετάζει. Η μηχανή ανίχνευσης χρησιμοποιεί τους κανόνες του Snort για τον λόγο αυτό. Οι κανόνες διαβάζονται σε εσωτερικές δομές δεδομένων και έπειτα σύμφωνα με αυτούς εξετάζονται τα πακέτα δεδομένων. Εάν ένα το πακέτο ανταποκρίνεται στις απαιτήσεις κάποιου κανόνα, τότε η κατάλληλη ενέργεια λαμβάνεται για το πακέτο αυτό διαφορετικά "αφήνεται ελεύθερο". Μία από αυτές τις κατάλληλες ενέργειες είναι και η ειδοποίηση του διαχειριστή του συστήματος.

Η μηχανή ανίχνευσης κρίσιμο κομμάτι του Snort. Ανάλογα με το πόσο ισχυρό είναι το υπολογιστικό σύστημα στο οποίο είναι εγκατεστημένο το Snort και ανάλογα με το

πόσοι κανόνες έχουν καθοριστεί, διαφέρει και ο χρόνος επεξεργασίας των διαφόρων πακέτων που δέχεται το Snort για έλεγχο. Εάν η κυκλοφορία σε ένα δίκτυο είναι πάρα πολύ υψηλή όταν το Snort εργάζεται σε NIDS mode, τότε ο έλεγχος για τυχόν επιθέσεις ίσως να καθυστερήσει σημαντικά. Το "φορτίο" στη μηχανή ανίχνευσης εξαρτάται από τους ακόλουθους παράγοντες:

- ✓ Αριθμός κανόνων
- ✓ Επεξεργαστική ισχύς του συστήματος που είναι εγκατεστημένο στο Snort
- ✓ Ταχύτητα των διαύλων του συστήματος του Snort
- ✓ Το φορτίο του δικτύου

Κατά το σχεδιασμό ενός συστήματος ανίχνευσης παρεισφρήσεων, όλοι οι παραπάνω παράγοντες θα πρέπει να λαμβάνονται υπόψη. Να σημειωθεί επίσης ότι το σύστημα ανίχνευσης μπορεί να διασπάσει ένα πακέτο σε άλλα μικρότερα και να εφαρμόσει τους κανόνες στα διασπασμένα πλέον μέρη. Αυτά τα μέρη μπορεί να είναι:

- ✓ Η IP του πακέτου.
- ✓ Τα πρωτόκολλα TCP,UDP, ICMP ή σε άλλα πρωτόκολλα.
- ✓ Οι "επικεφαλίδες» της επιγραφής του DNS, του FTP, της SNMP και του SMTP.

Η μηχανή ανίχνευσης του Snort λειτουργεί διαφορετικά για τις διάφορες εκδόσεις του. Σε όλες τις εκδόσεις 1.x, η μηχανή ανίχνευσης σταματά την περαιτέρω επεξεργασία ενός πακέτου όταν αντιστοιχείται με έναν κανόνα. Ανάλογα με τον κανόνα, η μηχανή ανίχνευσης παίρνει τη κατάλληλη ενέργεια. Δηλαδή είτε καταγράφει το πακέτο είτε ειδοποιεί τους αρμόδιους φορείς. Αυτό σημαίνει ότι εάν ένα πακέτο ταιριάζει με τα κριτήρια παραπάνω από ενός κανόνα, τότε μόνο ο πρώτος κανόνας εφαρμόζεται στο πακέτο χωρίς να γίνεται έρευνα άλλων αντιστοιχιών με άλλους κανόνες. Αυτή η μέθοδος παρουσιάζει ένα πρόβλημα. Ένας κανόνας χαμηλής προτεραιότητας "παράγει μια χαμηλής επικινδυνότητας ειδοποίηση", ακόμα κι αν στην συνέχεια βρεθεί κάποιος κανόνας υψηλότερης επικινδυνότητας.

Αυτό το πρόβλημα αποκαθίσταται στο Snort στην έκδοση 2 όπου όλοι οι κανόνες αντιστοιχούνται ενάντια σε ένα πακέτο πριν γίνει οποιαδήποτε ειδοποίηση ή καταγραφή. Η μηχανή ανίχνευσης στην έκδοση 2.0 του Snort έχει δημιουργηθεί εξολοκλήρου από την αρχή έτσι ώστε να είναι πολύ γρηγορότερη στην ανίχνευση των απειλών από τις προηγούμενες εκδόσεις.

4. ***Σύστημα καταγραφής και ειδοποιήσεων***

Ανάλογα με το τι εντοπίζει το σύστημα ανίχνευσης μέσα σε ένα πακέτο, το πακέτο ίσως να καταγραφεί, ή εξαιτίας αυτού να ειδοποιηθεί ο διαχειριστής του συστήματος. Οι καταγραφές για τα συμβάντα που αντιμετωπίζει το Snort αποθηκεύονται σε μορφή απλών αρχείων κειμένων τη, αρχείων τύπου tcpdump-style ή σε κάποια άλλη μορφή. Όλα αυτά τα αρχεία αποθηκεύονται εξ ορισμού στον κατάλογο `/var/log/`. Βέβαια

είναι δυνατή η αλλαγή αυτής της τοποθεσίας μέσω της γραμμής εντολών του Snort και της χρήσης της εντολής – l.

5. ***Output modules***

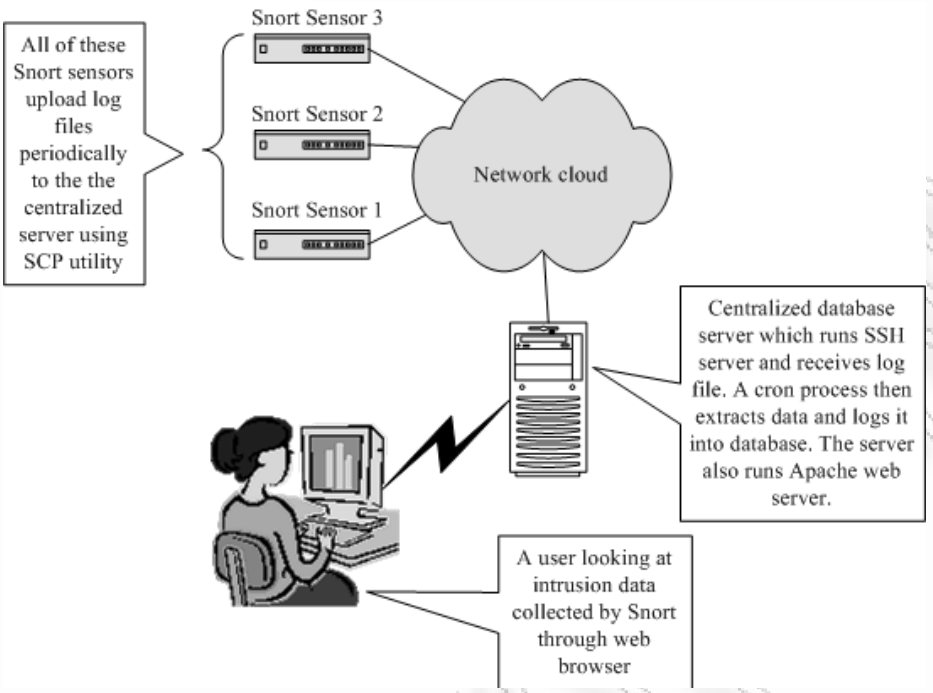
Τα **Output Modules** μπορούν να κάνουν διαφορετικές διαδικασίες ανάλογα με το επιθυμεί κάποιος να αποθηκεύει τις "εξόδους" του Snort. Δηλαδή τις προειδοποιήσεις και τις καταγραφές των απειλών. Στην πραγματικότητα ελέγχουν τον τύπο των εξόδων αυτών. Ανάλογα με την παραμετροποίηση, τα output modules μπορούν να κάνουν πράγματα όπως:

- ✓ Απλή καταγραφή στα αρχεία /var/log/snort/alerts ή σε κάποιο άλλο αρχείο
- ✓ Αποστολή των SNMP traps
- ✓ Αποστολή των μηνυμάτων στο syslog
- ✓ Καταγραφή σε μια βάση δεδομένων όπως την MySQL ή την Oracle.
- ✓ Παραγωγή αρχείων τύπου XML
- ✓ Τροποποίηση των ρυθμίσεων στους δρομολογητές και τα firewalls
- ✓ Αποστόλη μηνυμάτων με την μορφή pop-up windows σε windows-based συστήματα
- ✓ Αποστολή ειδοποιήσεων σε μορφή e-mail
- ✓ Αποστολή ειδοποιήσεων σε web-interfaces

2.6 Υποστηριζόμενες Πλατφόρμες

Το Snort υποστηρίζεται από ένα μεγάλο πλήθος λειτουργικών συστημάτων. Την τρέχουσα στιγμή το Snort είναι διαθέσιμο για τα ακόλουθα συστήματα:

- ✓ Linux
- ✓ OpenBSD
- ✓ FreeBSD
- ✓ NetBSD
- ✓ Solaris (both Sparc and i386)
- ✓ HP-UX
- ✓ AIX
- ✓ IRIX
- ✓ MacOS
- ✓ Windows



Μια πολύπλοκη αλλά σταθερή και αξιόπιστη εγκατάστη 3 Snort για την ανίχνευση παρεισφρήσεων σε ένα δίκτυο.

Εγκατάσταση του Snort σε linux

Παρακάτω περιγράφεται η διαδικασία εγκατάστασης του Snort σε υπολογιστικό σύστημα με λειτουργικό σύστημα Linux. Αναφέρονται επίσης τα πακέτα που θα πρέπει να εγκατασταθούν πριν να γίνει οποιαδήποτε εγκατάσταση του Snort.

Προαπαιτούμενα πακέτα

- ✓ *Libpcap0.8-dev*
- ✓ *libmysqlclient15-dev*
- ✓ *mysql-client-5.0*
- ✓ *mysql-server-5.0*
- ✓ *bison*
- ✓ *flex*
- ✓ *apache2*
- ✓ *libapache2-mod-php5*
- ✓ *php5-gd*
- ✓ *php5-mysql*
- ✓ *libphp-adodb*
- ✓ *php-pear*
- ✓ *libc6-dev*
- ✓ *g++*
- ✓ *gcc*

Τα παραπάνω πακέτα είναι απαραίτητα για μία σωστή και πλήρως λειτουργική εγκατάσταση του Snort σε περιβάλλον Linux.

Απόκτηση πακέτων Snort, εγκατάσταση και παραμετροποίηση

```
# cd /root
# mkdir snorttmp
# cd /root/snorttmp
# wget http://www.snort.org/dl/current/snort-2.8.0.tar.gz
# tar -xzvf /root/snorttmp/snort-2.8.0.tar.gz
# wget http://www.snort.org/pub-
bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.4.tar.gz
# tar -xzvf /root/snorttmp/snort-2.8.0/snortrules-pr-2.4.tar.gz
# wget http://downloads.sourceforge.net/secureideas/base-
1.3.8.tar.gz?modtime=1183896336&big_mirror=0
# tar -xzvf /root/snorttmp/base-1.3.8.tar.gz
# wget
http://downloads.sourceforge.net/adodb/adodb502a.tgz?modtime=1191343792&big_
mirror=0
# tar -xzvf /root/snorttmp/adodb502a.tgz
```

Με την παραπάνω διαδικασία έχουμε αποκτήσει το Snort καθώς και πρόσθετο λογισμικό το οποίο όμως είναι απαραίτητο για την ορθή εγκατάσταση του Snort. Στην συνέχεια παρουσιάζεται η εγκατάσταση και η παραμετροποίηση του.

```
# cd /root/snorttmp/pcre-7.4# ./configure
# make
# make install
# cd /root/snorttmp/snort-2.8.0
# ./configure -enable-dynamicplugin --with-mysql
# make
# make install
```

Πρέπει επιπλέον να δημιουργηθούν οι παρακάτω φάκελοι για την σωστή παραμετροποίηση του Snort.

```
# mkdir /etc/snort /etc/snort/rules /var/log/snort
# cd /root/snorttmp/snort-2.8.0/rules
# cp * /etc/snort/rules
# cd /root/snorttmp/snort-2.8.0/etc
# cp * /etc/snort/
# cp /usr/local/lib/libpcre.so.0 /usr/lib
```

Επιπλέον στο /etc/snort/snort.conf πρέπει να γίνουν οι εξής αλλαγές

```
"var HOME_NET any" σε "var HOME_NET XXX.XXX.XXX.XXX/XXX" (ip και subnetmask
του αντιστοιχου δικτύου)
"var EXTERNAL_NET any" σε "var EXTERNAL_NET !$HOME_NET
"var RULE_PATE ../rules" σε "var RULE_PATH /etc/snort/rules"
```

Έπειτα από αυτές τις ενέργειες το Snort είναι πλέον έτοιμο να ξεκινήσει να αναλύει τα διάφορα πακέτα και ανάλογα με τα αποτελέσματα των ελέγχων να εκτελεί και τις κατάλληλες ενέργειες. Για να ξεκινήσει το Snort αρκεί η παρακάτω εντολή:

```
#snort -v
```

Τα αποτελέσματα που εμφανίζει στην οθόνη θα είναι παρόμοια με τα παρακάτω:

```
-*> Snort! <*-
Version 1.6.2.2
By Martin Roesch (roesch@clark.net, www.clark.net/~roesch)
07/11-20:33:59.675507 192.168.1.10:3436 -> 192.168.1.1:53
UDP TTL:64 TOS:0x0 ID:22707
Len: 36

07/11-20:33:59.916825 192.168.1.1.436 -> 192.168.1.10:3436
UDP TTL:57 TOS:0x0 ID:1177
Len: 91

07/11-20:33:59.917642 192.168.1.10:3437 -> 192.168.1.153
UDP TTL:64 TOS:0x0 ID:22708
Len: 47

07/11-20:34:00.078208 192.168.1.1:53 -> 192.168.1.10:3437
UDP TTL:57 TOS:0x0 ID:1367
Len: 109
```

Βλέποντας λοιπόν αποτελέσματα σαν και αυτά στην οθόνη καταλαβαίνει κανείς πως το Snort λειτουργεί. Παρόλα αυτά στην συνέχεια δίνονται και κάποιοι άλλοι τρόποι για την εξακρίβωση της σωστής λειτουργίας του Snort.

2.7 Δοκιμή του Snort

Αφού ξεκινήσει να εκτελείται το Snort, πρέπει να ελέγξουμε εάν πραγματικά ελέγχει τα διάφορα πακέτα δεδομένων που διαβιβάζονται στα interfaces τα οποία το Snort παρακολουθεί.

Εάν η εκίνηση του Snort έγινε από την γραμμή εντολών με την εντολή "`-A console`" τότε στην οθόνη θα εμφανιστούν άμεσα ειδοποιήσεις ασφαλείας. Εντούτοις, εάν η εκίνηση έγινε σε daemon mode και δεν χρησιμοποιήθηκε η γραμμή εντολών, τότε οι ειδοποιήσεις θα στο αρχείο /var/log/snort/alert.

Οι ειδοποιήσεις που θα εμφανιστούν στην οθόνη θα πρέπει να μοιάζουν στην ακόλουθη:

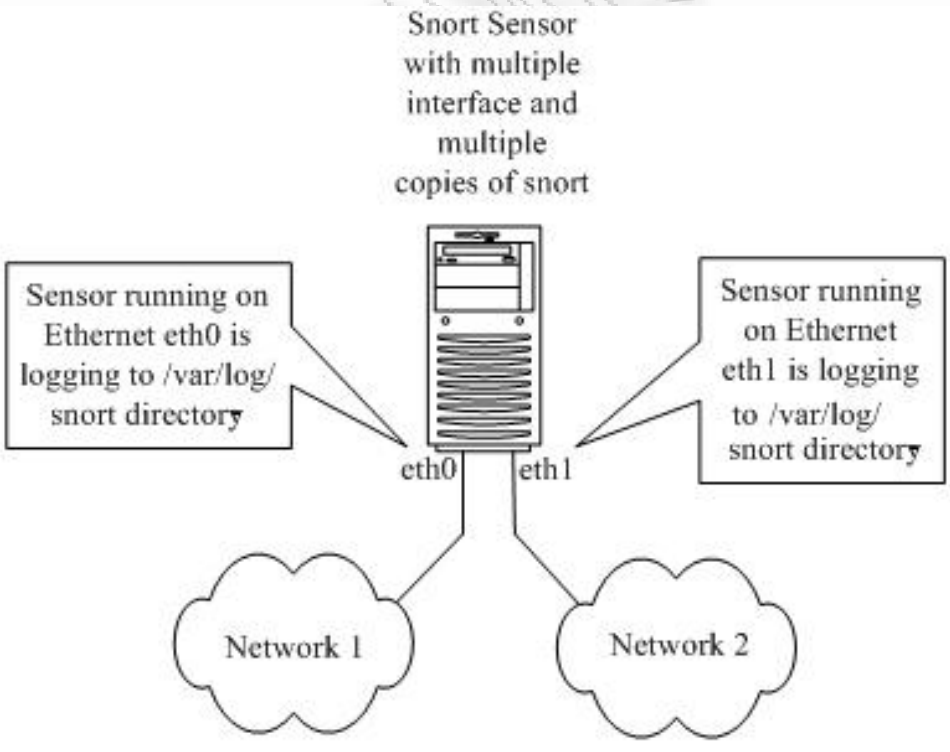
```
11/19-18:51:04.560952 [**] [1:498:3] ATTACK RESPONSES id
check returned root [**] [Classification: Potentially Bad
Traffic] [Priority: 2] {ICMP} 10.100.1.105 -> 255.255.255.255
```

2.8 Εκτέλεση του Snort σε πολλαπλά network interfaces

Όταν το Snort εκτελείται με τις προκαθορισμένες ρυθμίσεις τότε αυτό ελέγχει τα δεδομένα ενός μόνο διαδικτυακού interface.. Χρησιμοποιώντας στην γραμμή εντολών την επιλογή –i <interface_name> μπορεί κανείς να ορίσει το interface το οποίο θέλει να ελέγχει ο Snort. Δίνεται όμως και η δυνατότητα του ελέγχου πολλαπλών interfaces μέσω μιας εγκατάστασης του Snort. Για παράδειγμα, οι ακόλουθες δύο εντολές αναγκάζουν το Snort να ελέγχει δύο διεπαφές(interfaces) δικτύων eth0 και eth1 σε έναν υπολογιστή βασισμένο σε Linux.

```
/opt/snort/bin/snort -c /opt/snort/etc/snort.conf -i eth0 -l /
var/log/snort0
/opt/snort/bin/snort -c /opt/snort/etc/snort.conf -i eth1 -l /
var/log/snort1
```

Να σημειωθεί ότι με τις παραπάνω εντολές δημιουργούνται δύο κατάλογοι: */log/snort0 και /var/log/snort1*, έτσι ώστε να καταγράφονται ξεχωριστά τα συμβάντα από τα δύο interfaces. Επιπλέον πρέπει αυτοί οι κατάλογοι να έχουν δημιουργηθεί πριν την εκκίνηση του Snort.



2.9 Κανόνες με το Snort

Στην συνέχεια παρουσιάζονται κάποια βασικά παραδείγματα κανόνων που χρησιμοποιούνται στο Snort καθώς επίσης και ο τρόπος γραφής νέων κανόνων.

Παράδειγμα 1.1

Παρακάτω παρουσιάζεται ο πρώτος κανόνας. Δεν υπάρχει λόγος το Snort να ελέγχει για έναν τέτοιο κανόνα καθώς όχι μόνο δεν προσφέρει κάποια ειδοποίηση αλλά και καταστρέφει το συστήματα. Στην συνέχεια αναλύεται γιατί συμβαίνει αυτό:

```
alert ip any any -> any any (msg: "IP Packet detected";)
```

Για να χρησιμοποιηθεί αυτός ο κανόνας πρέπει να προστεθεί στο τέλος του αρχείου snort.conf την πρώτη φορά αφού γίνει εγκατάσταση. Αφού γίνει αυτό το Snort θα παραγάγει συνεχώς ειδοποιήσεις για κάθε πακέτο IP που ελέγχει είτε πρόκειται για κακόβουλα δεδομένα είτε όχι. Επομένως αυτό θα καταλήξει αναπόφευκτα στην “κατάληψη” ολόκληρου του ελεύθερου χώρου στο σκληρό δίσκο. Συνήθως αυτή η εντολή χρησιμοποιείται προκειμένου να διαπιστωθεί εάν όντως το Snort λειτουργεί σωστά.

Παράδειγμα 1.2

Ο παρακάτω κανόνας ειδοποιεί το σύστημα μόνο για τα πακέτα που έχουν την IP 192.168.1.113 ως παραλήπτη.

```
alert icmp any any -> 192.168.1.113/32 any \ (msg: "Ping with
TTL=100"; ttl:100;)
```

Δομή ενός κανόνα

Η γενική δομή ενός κανόνα που αναγνωρίζει το Snort είναι της μορφής που φαίνεται στο παρακάτω σχήμα.

Action	Protocol	Address	Port	Direction	Address	Port

Στην συνέχεια αναλύεται κάθε ένα από τα τμήματα ενός κανόνα που φαίνονται στο παραπάνω σχήμα.

- ***Action***
  Καθορίζει τον τύπο της ενέργειας που θα γίνει όταν ένα πακέτο πληροί τις προϋποθέσεις του συγκεκριμένου κανόνα. Ένα τυπικό όρισμα σαν action συνήθως ειδοποιεί το σύστημα ή καταγράφει το συμβάν.
- ***Protocol***
  Χρησιμοποιείται για να ορίσει σε ποιο τύπο πρωτοκόλλου που θα έχουν τα

δεδομένα που θα εξετάσει το Snort, θα εφαρμοστεί ο συγκεκριμένος κανόνας. Για παράδειγμα μερικά από τα ορίσματα είναι τα: IP, ICMP, UDP κ.α..

- ***Address***

  Καθορίζει την IP προέλευσης και την IP προορισμού. Σαν όρισμα μπορεί να είναι ένας μόνο host, περισσότεροι από ένας ή ακόμα και ένα δίκτυο υπολογιστών. Να σημειωθεί πως υπάρχουν δύο πεδία αυτού του τύπου. Το ένα, όπως αναφέρθηκε και προηγουμένως, είναι για την διεύθυνση προέλευσης και το άλλο για τον προορισμό. Το ποιο πεδίο από τα δύο θα είναι προορισμού και ποιο προέλευσης καθορίζεται από το πεδίο *Direction.* Για παράδειγμα εάν το πεδίο *Direction* είναι "->", τότε η διεύθυνση στα αριστερά είναι αυτή της προέλευσης και αντίστοιχα αυτή στα δεξιά της αποστολής.

- ***Port***

  Στην περίπτωση που το πρωτόκολλο είναι TCP ή UDP, τότε το πεδίο αυτό καθορίζει την port προορισμού και προέλευσης του πακέτου το οποίο θα εξεταστεί σύμφωνα με τον συγκεκριμένο κανόνα. Στην περίπτωση που το πεδίο port είναι διαφορετικό από TCP ή UDP, τότε ο καθορισμός στον κανόνα κάποιας πόρτας δεν έχει κανένα νόημα.

Κεφάλαιο 3. Θεωρία

Η πλατφόρμα του λογισμικού που προτείνετε σκοπεύει να ειναι η γενική λύση διεξαγόμενων πειραμάτων στον χώρο επεξεργασίας αλληλογραφίας για συναγερμούς IDS. Οι ερευνητές στο συγκεκριμένο χώρο προτείνουν διάφορες λύσεις με σκοπό να αυξήσουν ποιότητα των προκειπτόμενων alert-set ,και τείνουν να διεξάγουν σχετικά πειράματα έτσι ώστε να αποδείξουν την αποτελεσματικότητα των μεθόδων τους.

Αυτά τα πειράματα όμοια ή πανομοιότυπα μέρη.Αυτά τα μέρη αναπτίσσονται από από τον κάθε ερευνητή αυστηρά, ενώ θα μπορούσαν να αναπτύσσονται μια φορά και να χρησιμοποιούνται από όλους.Αφενός , τα αποτελέσματα που λαμβάνουμαι είναι από διαφορετικά πειράματα δεν είναι κατάλληλα για ευθύς σύγκριση , αφού έχουν παραχθεί κάτω από διαφορετικές περιστάσεις. Επιπλέον, μπορεί να είναι χρήσιμο για τους ερευνητές να ξαναχρησιμοποιήσουνε αναπτυσσόμενα στοιχεία από άλλους σαν μέροι των μεθόδων τους.

Ο σκοπός για να αναπτύξουμε την πλατφόρμα του λογισμικού είναι:

1. να παρέχουν στους ερευνητές με όλα "τα έτοιμα προς χρησιμοποίηση" στοιχεία που μπορεί να είναι χρήσιμα σε αυτούς και να τουσ επιτρέψει να συγκεντρωθούν στη μέθοδο ανάπτυξης του πυρήνα που πρότειναν
2. να παρέχουν ένα πλαίσιο το οποίο θα δημιουργήσει μια ομοιομορφία σχετικά με τα αποτελέσματα πειραμάτων ασχέτως τους παραμέτρους και περιστάσεις του κάθε πειράματος και έτσι να δώσει την δυνατότητα στους ερευνητές να παράγουν συγκρίσιμα αποτελέσματα
3. να δώσει την δυνατότητα στους ερευνητές να ξαναχρησιμοποιήσουν στοιχεία που έχουν αναπτυχθεί από άλλους.

Στην διαδικασία διεξαγωγής πειράματος σχετικά με τη διαδικασία συναγερμών IDS προειδοποίησης αλληλογραφιας , υπάρχουν μερικές καθιερωμένες εργασίες οι οποίες ο ανοικοδομητής πρέπει να διεξάγει :

- πρέπει να αποκτήσει πρόσβαση στον μνχανισμό αποθήκευσης (αρχεία κειμένων, xml αρχεία, βάσεις δεδομένων) του IDS και την εξαγωγή ακατέργαστων δεδομένων
- πρέπει να διαβάσει τα ακατέργαστα δεδομένα και να τα μετατρέψει σε φόρμες επεξεργασίας αλληλογραφίας
- πρέπει να πέρει μέτρα να αφερεί περιττές πληροφορίες όπως διπλότυπες ή άχρηστες προειδοποιήσεις

Η προτεινόμενη πλατφόρμα λογισμικού σχεδιάσθηκε με τέτοιο τρόπο να συμπεριλαμβάνει έτοιμα προς χρησιμοποίηση στοιχεία για αυτές της σύνηθες εργασίες, τουλάχιστον για τα πιο δημοφιλοί IDSs.

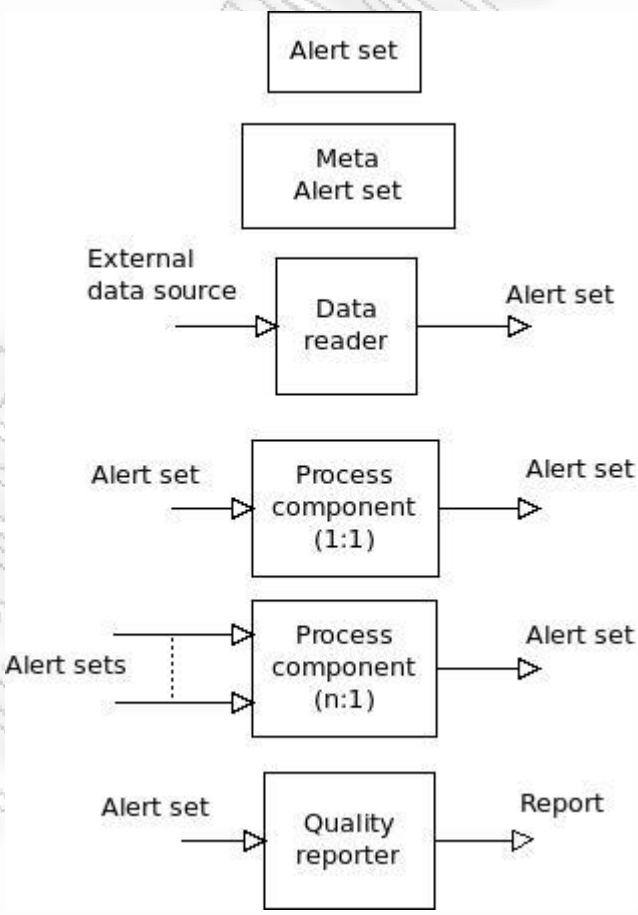
Εκτός από τα στοιχεία τα οποία χρησιμοποιούνται να προετοιμάσουν πληροφορίες για το πείραμα , υπάρχουν και κάποιες άλλες καθιερωμένες διαδικασίες ,οι οποίες πρέπει να πάρουν μέρος στο τέλος κάθε πειράματος.Το μέτρο ποιότητας που χρησιμοποιήτε σχετικά με το παραγόμενο αποτέλεσμα διαφέρει μεταξύ των ερευνητών.Θα ήταν χρήσιμο να

χρησιμοποιηθούν καθιερωμένα στοιχεία ,τα οποία βασίζονται σε συγκεκριμένα μέτρα, με σκοπό να κάνουμε συγκρίσιμα μέτρα ποιότητας και να μπορούμε μαζικά να καταλήξουμε σε μεθόδους αποτελεσματικότητας.

Όπως συμβαίνει σε κάθε τομέα έρευνας ,οι προτεινόμενοι μέθοδοι μπορούν να χρησιμοποιηθούν ,με ή χωρίς μετατροπές, σε μεταγενέστερες έρευνες.Η προτεινόμενη πλατφόρμα θα στηρίξει εξαγωγή και εισαγωγή στοιχείων.Ο σκοπός είναι να διευκολύνει την παροκία ερευνητών IDS ,να συνεργαστούν και να παράγουν καλύτερα αποτελέσματα μακροπρόθεσμα. Πρέπει να υπέρχει ένα μέρος αποθήκευσης στοιχείων στο διαδίκτυο υπογεγραμένα από τους ερευνητές.Τα στοιχεία σε αυτό το μέρος αποθήκευσης με σχετικά μεταδεδομένα,τα οποία θα δίνουν την δυνατότητα έρευνας μεταξύ τους.Οι ερενυητές έτσι θα μπορούν να βρούν στοιχεία που ταιριάζουν στις ανάγκες τους και να τα χρησιμοποιούμε ,με την άδεια του ιδιοκτήτη,για να βελτιώσουν την έρευνα τους.

ΣΧΕΔΙΟ ΣΥΣΤΗΜΑΤΟΣ

Τα στοιχεία του πυρήνα στη βάση είναι τα Data readers, Process components και Quality Reporters και απεικονίζονται στο ακόλουθο διάγραμμα.Το κυρίως στοιχείο πληροφορίας είναι το Alert set το οποίο χρησιμοποιειται για ανταλλαγή πληροφοριών μεταξύ των προαναφερόμενων στοιχείων.



Το Alert set είναι μια συγκεκριμένη φόρμα πληροφοριών που περιλαμβάνει τομείς από όλες τις απαιτούμενες πληροφορίες για τον κάθε συναγερμό προερχόμενη από το IDS ,όπως χρόνος, IP αποστολέα, IP προορισμου, είδος συναγερμού κτλ.Αυτή η φόρμα καθορίζεται αυστηρά και οι χρήστες πρέπει να την ακολουθούν για να εκπληρώσουν την απαίτηση συμβατότητας μεταξύ των στοιχείων που αναπτύχθηκαν από τους διαφορετικούς χρήστες.

Τα Data readers είναι στοιχεία υπεύθυνα για την είσοδο στο σύστημα. Χρησιμοποιούνται για να διαβάσουν πληροφορίες από εξωτερικές πηγές, έτσι για καθε IDS χρησιμοποιείται διαφορετικό Data reader που θα πρέπει να εφαρμοστεί.Η πλατφόρμα θα σταλεί με ένα έτοιμο Data reader for Snort .Η διαμόρφωση αυτού του είδους στοιχείων θα πρεπει να αναφέρονται στο αρχείο καταχώρησης ,συμπεριλαμβάνοντας συναγερμούς ή λεπτομέρειες για πρόσβαση στη σχετικη βάση δεδομένων.

Τα Process components είναι το κύριο δοχείο μεθόδων.Ο χρήστης μπορεί να συνδυάσει διαφορετικά υπάρχοντα Process components ή να δημιουργήσει καινούργια που αντανακλούνε την λογική των μεθόδων που προτείνει.Τα Process components συνήθως θα λειτουργήσουν με μία εισαγωγή. Μπορούν επίσης να λειτουργήσουν με πολλαπλά Alert set εισαγωγών, για να αντανακλουν μεθόδους που θα χρησιμοποιούν παραπάνω του ενός IDS αισθητήρων.Μπορούν ελεύθερα να συνδεθούν με οποιαδήποτε τρόπο που προτιμάει ο χρήστης με την προυπόθεση οι απαιτήσεις εισαγωγής και εξαγωγής να εκπληρώνονται.

Οι Quality Reporters είναι στοιχεία που δέχονται ως εισαγωγή Alert set και παραγουν ένα μέτρο ποιότητας.Για παράδειγμα ,ένα Quality Reporter μπορεί να υπολογίσει λανθασμένα θετικό ποσοστό ή πυκνότητα προειδοπειήσεων.Μερικά Quality Reporters μπορούν να χρησιμοποιηθούν με σκοπό να καταβάλουνε την χρησιμότητα της μεθόδου που αναπτύσεται.Μερικά από αυτά μπορεί να απαιτούν επιπλέον πληροφορίες όπως μία λίστα από αληθινές επιθέσεις που πηραν μέρος.

Ο χρήστης μπορεί να βασιστεί σε ένα μοντέλο αθέτησης πληροφοριών(Alert set) και να δημιουργήσει τα δικά του μοντέλα σε μία μορφή από Meta alert sets.Αυτά τα μοντέλα μπορούν να συμπεριλαμβάνουν meta data που παράγονται από κάθε συναγερμό, τις οποίες ο ανοικοδομητής τις θεωρεί σημαντικές.Θα πρέπει να συμπεριλάβουν όλες τις βασικές πληροφορίες για τα Alert sets για συμβατικούς λόγους, ενώ καινούργια Quality Reporters μπορεί να απαιτούνται για να βοηθήσουνε τα τεχνητά δεδομένα.

3.1 Φίλτρο NRA (Neighboring Related Alerts)

Το πρώτο συστατικό του φίλτρου είναι το NRA, το οποίο είναι βασισμένο στην παρατήρηση ότι TPs εμφανίζεται σε batches με ομοιότητες στις IP διευθύνσεις προέλευσης και προορισμού. Το NRA διαμορφώνεται από δύο παραμέτρους, δηλαδή την t0 και την n0. Η παράμετρος t0 καθορίζει το μέγεθος του χρονικού παραθύρου που χρησιμοποιείται για να μετρηθούν οι γείτονες. Η παράμετρος n0 χρησιμοποιείται ως threshold για την μετατροπή του αριθμού των γειτόνων σε belief.

Η ιδέα είναι να μετρηθεί για κάθε ειδοποίηση a (i) ο αριθμός των ειδοποιήσεων που βρίσκονται στο χρονικό παράθυρο [ti − t0, ti + t0] και έχουν την ίδια IP στους τομείς της IP προέλευσης και προορισμού (οι IPs προέλευσης ή προορισμού μπορεί να είναι ίδιες ή

μπορεί η IP προέλευσης του ενός να είναι ίδια με την IP προορισμού της άλλης ειδοποίησης). Το NRA υπολογίζει το ποσοστό αυτού του αριθμού στον προκαθορισμένο μεταβλητό n0, το οποίο χρησιμοποιείται threshold έτσι ώστε να χαρακτιριστεί αληθής ή ψευδής μια ειδοποιήση.

Το σύνολο που περιέχει όλες τις διευθύνσεις IP (προέλευσης ή προορισμός) των a(i) ειδοποιήσεων συμβολίζεται με το $\{ip_i\}$. Για κάθε ειδοποίηση $\alpha(i)$ ο αριθμός nra_i υπολογίζετα ως εξής:

$$nra_i = \sum_{j=0}^{n} c_{ij}$$

Όπου

$$c_{ij} = \begin{cases} 1, & \text{if } \{ip_i\} \cap \{ip_j\} \neq \varnothing \text{ and } |t_i - t_j| < t_0 \\ 0, & \text{otherwise} \end{cases}$$

Κατόπιν, η εξακρίβωση για το αν μια a (i) είναι ένα TP, βασίζεται στην παράμετρο n0:

$$b_i^{nra} = \frac{max(nra_i, n_0)}{n_0}$$

Το αποτέλεσμα του NRA είναι μία παράταξη $b_i^{nra}, i \in (0, n)$, η οποία περιέχει τις εξακριβώσεις για το αν οι ειδοποιήσεις είναι TPs (το b_i^{nra} είναι μεταξύ 0 και 1).

Τα πρώτα μας πειράματα με το NRA αποκάλυψαν ότι ενώ αποδίδει καλά γενικά, δεν μπορεί να αναγνωρίσει τα TPs που συσχετίζονται με IP sweep attacks. Η απόδοση του NRA μπορεί να ενισχυθεί με τη χρήση μιας παραλλαγής του συστατικού που αποκαλείται NRAsweep. Αυτό το συστατικό είναι βασισμένο στην ίδια λογική με το NRA , αλλά μαζί με την IP

προέλευσης και προορισμού, χρησιμοποιεί και τα υποδίκτυα τους (τα πρώτα τρία μέρη της διεύθυνσης IP). Παραδείγματος χάριν το υποδίκτυο της διεύθυνσης 195.251.123.76 είναι:

$$sub(195.251.123.76) = 195.251.123$$

Για κάθε συγκεκριμένη ειδοποίηση και για ένα ορισμένο χρονικό παράθυρο το τελευταίο συστατικό, μετρά τις ειδοποιήσεις πού έχουν ίδιες IP διευθύνσεις και ίδια υποδίκτυα στις IP διευθύνσεις. Το συστατικό NRAsweep λειτουργεί ως εξής:

$$nra_i^{sweep} = \sum_{j=0}^{n} c_{ij}$$

Όπου $c_{ij} = 1$ εάν μια από τις ακόλουθες υποθέσεις ισχύει:

$$sip_i \equiv sip_j \text{ and } sub(dip_i) \equiv sub(dip_j) \text{ and } |t_i - t_j| < t_0$$
$$sip_i \equiv dip_j \text{ and } sub(dip_i) \equiv sub(sip_j) \text{ and } |t_i - t_j| < t_0$$
$$dip_i \equiv sip_j \text{ and } sub(sip_i) \equiv sub(dip_j) \text{ and } |t_i - t_j| < t_0$$
$$dip_i \equiv dip_j \text{ and } sub(sip_i) \equiv sub(sip_j) \text{ and } |t_i - t_j| < t_0$$

διαφορετικά $c_{ij} = 0$. Τότε, η εξακρίβωση ότι το a(`i`) είναι ένα TP υπολογίζεται βασιζόμενο στην παράμετρο `n0` :

$$b_i^{nra_{sweep}} = \frac{max(nra_i^{sweep}, n_0)}{n_0}$$

The maximum of the two beliefs $b_i^{nra}, b_i^{nra_{sweep}}$ is selected as the final one.
If $b_i^{nra} >, b_i^{nra_{sweep}}$ then b_i^{nra} is the result.
If $b_i^{nra} <, b_{ij}^{nra_{sweep}}$ then $b_i^{nra_{sweep}}$ is the result.

3.2 Φίλτρο HAF (High Alert Frequency)

Το HAF είναι το δεύτερο συστατικό του φίλτρου που είναι βασισμένο στην παρατήρηση ότι τα TPs χαρακτηρίζονται από υψηλές υπογραφές σχετικές με την συχνότητα (signature-related frequency), η οποία είναι σημαντικά υψηλότερη από την μέση υπογραφή σχετική με την συχνότητα της υπογραφής τους. Το HAF διαμορφώνεται από μια ενιαία παράμετρο, η οποία χρησιμοποιείται ως “πρότυπο” για να καθοριστεί εάν μια ειδοποίηση είναι αληθινή ή ψεύτικη.

Για να λειτουργεί σωστά το HAF, μια μέση συχνότητα πρέπει να υπολογιστεί για κάθε υπογραφή. Αυτό είναι ένα μετρικό που παρουσιάζει τη διανομή της συγκεκριμένης υπογραφής γενίκοτερα. Κατόπιν μια στιγμιαία υπογραφή σχετική με την συχνότητα πρέπει να υπολογιστεί για κάθε ειδοποίηση κατά την διάρκεια της εμφάνισης της. Αυτό είναι ένα μετρικό της διανομής της συγκεκριμένης υπογραφής την στιγμή της εμφάνισης της ειδοποιήσης. Οι πληροφορίες για την εγκυρότητα της ειδοποίηση συμπεριλαμβάνονται στο ratio υπογραφής για την συχνότητα.

Η μέση συχνότητα μπορεί να υπολογιστεί εκ των προτέρων ή ταυτόχρονα με την εκτέλεση του HAF. Στην τελευταία περίπτωση το HAF ίσως χρειαστεί διάφορες ειδοποιήσεις (για να υπολογιστεί η μέση συχνότητα)να αγνοηθούν στην αρχή μιας ενεργής περιόδου.

Μια μέση συχνότητα για κάθε υπογραφή πρέπει να υπολογιστεί ως $saf_m, m \in (0, s)$, όπου s+1 είναι ο αριθμός των διαφορετικών υπογραφών. Κατά αυτόν τον τρόπο κάθε ειδοποίηση μπορεί να αντιστοιχιστεί στο saf_m σύμφωνα με το δικό του sid_i.

Για κάθε ειδοποιήση υπολογίζεται η μόνιμη υπογραφή σχετική με την συχνότητα. Πρώτα υπολογίζεται ο ελάχιστος χρόνος διαφόρας; Χρόνος διαφοράς είναι ο χρονός μεταξύ μιας ειδοποίησης και της αμέσως επόμενης ειδοποιήσης με την ίδια υπογραφή.

$$mtd_i = \begin{cases} min(|t_i - t_j|, & \text{if } \exists j \rightarrow sid_j = sid_i, j \in (0, i-1) \cup (i+1, n) \\ \infty, & otherwise \end{cases}$$

Ύστερα, η μόνιμη signature-related frequency μπορεί να υπολογιστεί ως εξής

$$f_i = \frac{1}{1 + mtd_i}$$

Στον ελάχιστο χρόνο διαφοράς προσθέτουμε και το 1 (αντί για mtd_i έχουμε 1+mtd_i) έτσι ώστε να αποφύγουμε τυχόν διαιρέσεις με το 0. Η συχνότητα κάθε ειδοποίησης ομαλοποιείται από την αντίστοιχη μέση συχνότητα υπογραφών:

$$nf_i = \frac{f_i}{sfa_{sid_i}}$$

Προκειμένου να εξακριβωθεί το αν μια επιφυλακή είναι ένα TP, η ομαλοποιημένη συχνότητα ελέγχεται σε σχέση με την "παράμετρο κατώτατων ορίων" (threshold parameter).

$$b_i^{haf} = \frac{min(nf_i, \ell)}{l}$$

Το αποτέλεσμα του HAF είναι μια διάταξη όπου περιέχονται «εξακριβώσεις» για το εάν οι ειδοποιήσεις είναι TPs(το b_i^{haf} έχει τιμή είτε 1 είτε 0)

3.3 Φίλτρο UFP (Usual False Positives)

Το τελευταίο από τα τρία συστατικά είναι το UFP, το οποίο είναι βασισμένο στην πιθανότητα ότι μια επιφυλακή είναι ένα FP, λαμβάνοντας υπόψη την υπογραφή της. Αυτή η πιθανότητα μπορεί να εξαχθεί εύκολα από μια επίθεση σε ελεύθερη περίοδο (attack-free period). Η ιδέα πίσω από το UFP είναι να υπολογιστούν οι συχνότητες για κάθε υπογραφή s_m σε μια attack-free period ($f_{s_m}^{af}$).

Για την attack-free period το $f_{s_m}^{af}$ υπολογίζεται για όλες τις υπογραφές.

$$f_{s_m}^{af} = \frac{\sum_{j=0}^{n} k_{js_m}}{n}$$

Όπου

$$k_{js_m} = \begin{cases} 1, & \text{if } sid_j = s_m \\ 0, & \text{if } sid_j \neq s_m \end{cases}$$

και n+1 είναι ο αριθμός όλων των ειδοποιήσεων στην attack-free period.

Αυτές οι συχνότητες μπορούν να συνδεθούν με τις ειδοποιήσεις σύμφωνα με την υπογραφή των τελευταίων:

$$f_i^{af} = f_{s_{sid_i}}^{af}$$

Το επόμενο βήμα είναι να υπολογιστεί η τρέχουσα υπογραφή σχετική με την συχνότητα για κάθε ειδοποιήση για την πραγματική εξεταστική περίοδο(f_i^c). Μια επιλογή πρέπει να γίνει για το χρονικό διάστημα που θα χρησιμοποιηθεί για αυτόν τον υπολογισμό. Παραδείγματος χάριν, η μόνιμη σχετική με την υπογραφή συχνότητα μπορεί να υπολογιστεί σε περίοδο μιας ώρας ή σε περίοδο μιας ημέρας.

Αυτό πρέπει να επιλεχτεί λαμβάνοντας υπόψη την συχνότητα των ειδοποιήσεων και την ανοχή-δυνατότητα του συστήματος να μετρά την χρονική διαφορά μεταξύ των διαφόρων ειδοποιήσεων και των πραγματικών επιθέσεων. Η μόνιμη υπογραφή-σχετική με την συχνότητα για κάθε ειδοποιήση υπολογίζεται χρησιμοποιώντας την ίδια μεθοδολογία με την οποία η συχνότητα της attack-free period υπολογίζεται, με τη διαφορά ότι χρησιμοποιείται ένα συγκεκριμένο χρονικό διάστημα.

Για κάθε ειδοποιήση δύο συχνότητες είναι διαθέσιμες f_i^{af} και f_i^c. Η σχέση μεταξύ αυτών των δύο συχνοτήτων είναι ενδεικτική της φύσης της ειδοποιήσης.

Εάν η υπογραφή μιας ειδοποιήσης είναι πιθανότερο να οδηγήσει σε ένα FP, τότε η f_i^{af} θα πρέπει να μεγαλύτερη από την f_i^c.

Εάν η υπογραφή μιας ειδοποιήσης είναι πιθανότερο να οδηγήσει σε ένα TP, τότε η f_i^c θα πρέπει να μεγαλύτερη από την f_i^{af}.

Η αναλογία αυτών των δύο συχνοτήτων και του ορίου ℓ χρησιμοποιούνται για να εξακριβωθεί έαν μια ειδοποιήση είναι εάν TP

.
$$b_i^{ufp} = \frac{min(\frac{f_i^c}{f_i^{af}}, \ell)}{\ell}$$

Το αποτέλεσμα του συστατικού UFP είναι μια παράταξη $b_i^{ufp}, i \in (0, n),$ η οποία περιέχει την εξακρίβωση για το αν οι ειδοποιήσεις είναι TPs (b_i^{ufp} παίρνει τις τιμές 0 ή 1).

Κεφάλαιο 4. Υλοποίηση

Alert.java

Το αρχείο **Alert.java** αποτελεί ένα αντικείμενο στο οποίο θα αποθηκεύεται κάθε μια γραμμή που παράγεται από το Snort.

Ας υποθέσουμε ότι το Snort παράγει 3 alerts:

```
11/10-18:30:59.814220  [**] [1:3441:4] FTP PORT bounce attempt [**] [Classification: Misc Attack] [Priority: 2] {TCP} 206.48.44.18:1054 -> 172.16.112.100:21
11/10-18:30:59.815452  [**] [1:3441:4] FTP PORT bounce attempt [**] [Classification: Misc Attack] [Priority: 2] {TCP} 206.48.44.18:1054 -> 172.16.112.100:21
11/10-18:54:49.479979  [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.113.105:3719
```

Για κάθε μια από τις παραπάνω γραμμές μας ενδιαφέρουν

Α) Ο χρόνος (ημερομηνία και ώρα) του alert

Β) Η υπογραφή – signature

Γ) Το source IP Address

Δ) Το destination IP Address

Για κάθε μια γραμμή λοιπόν του παραπάνω αρχείου θα δημιουργείται ένα αντικείμενο τύπου Alert ώστε να τοποθετείται στη μνήμη και να μπορούμε προγραμματιστικά να αποκτούμε πρόσβαση στις πληροφορίες που μας ενδιαφέρουν:

```
package javaids;

public class Alert {

    // Οι μεταβλητές
    Time t;
    String signature;
    String src;
    String dest;

    // Ο Constructor
    Alert(String date, String sig, String s, String d) {
        t = new Time(date);
        signature = new String(sig);
        src = new String(s);
        dest = new String(d);
    }
```

```
    /*
     * Επιστρεφει το Snort Signature
     */
    String getSign() {
        return this.signature;
    }

    /**
     * @return Επιστρέφει το source IP Address
     */
    String subS() {
        int index = src.lastIndexOf('.');
        return src.substring(0, index);
    }

    /**
     * @return Επιστρέφει το destination IP Address
     */
    String subD() {
        int index = dest.lastIndexOf('.');
        return dest.substring(0, index);
    }

    /**
     * @return Επιστρέφει ένα αντικείμενο τύπου Time που
     * περιέχει την ακριβή ώρα που συνέβη το alert
     */
    Time getTime() {
        return t;
    }
}
```

Time.java

Όπως μπορεί να παρατηρήσει κανείς, το αρχείο Alert χρησιμοποιεί μια μεταβλητή τύπου Time. Αντικείμενο τέτοιου τύπου δεν υπάρχει στην Java και υλοποιήθηκε για τον σκοπό της εργασίας αυτής.

Το Snort παράγει όπως αναφέραμε alerts με την παρακάτω μορφή:

11/10-18:30:59.814220 [**] [1:3441:4] FTP PORT bounce attempt [**] [Classification: Misc Attack] [Priority: 2] {TCP} 206.48.44.18:1054 -> 172.16.112.100:21

Το πρώτο κομμάτι που είναι μαρκαρισμένο με έντονα γράμματα αποτελεί την ημερομηνία και ώρα. Η μορφή που χρησιμοποιεί το Snort είναι:

- Οι δύο πρώτοι χαρακτήρες είναι ο μήνας

- Μετά την κάθετο / ακολουθεί η ημέρα του μήνα

- Κατόπιν ακολουθεί η ώρα (ώρα , λεπτό , δευτερόλεπτο)

- Τέλος μετά την τελεία τα milliseconds του δευτερολέπτου που δημιουργήθηκε το alert

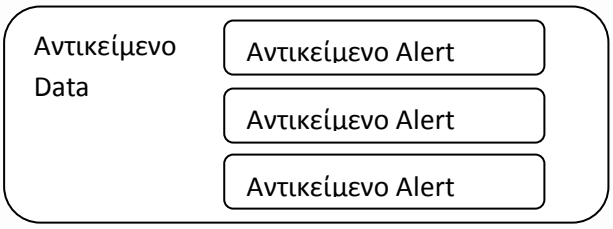
Το παρακάτω αρχείο λοιπόν, αναλαμβάνει την «αποκρυπτογράφηση» της ώρας. Περιέχει επίσης μια μέθοδο που υλοποιήθηκε για να βρίσκει την χρονική διαφορά ανάμεσα σε δύο alerts, κάτι που θα φανεί πολύ χρήσιμο στη συνέχεια της εργασίας.

```
package javaids;

class Time {

    // Οι μεταβλητές
    int date;
    int month;
    int hour;
    int min;
    int sec;
    int millis;

    // Ο Constructor. Παίρνει ένα string ως όρισμα
    // και αμέσως το αποκωδικοποιεί και τοποθετεί
    // αντίστοιχες τιμές στις μεταβλητές.
    Time(String d) {
        String temp;
        int index;

        month = Integer.parseInt(d.substring(0, 2));
        date = Integer.parseInt(d.substring(3, 5));
        hour = Integer.parseInt(d.substring(6, 8));
        min = Integer.parseInt(d.substring(9, 11));
        sec = Integer.parseInt(d.substring(12, 14));
```

```
        millis = Integer.parseInt(d.substring(15));

//System.out.println(month+""+date+""+hour+""+min+""+sec+""+millis);

    }

    /**
     * Βρίσκει τη χρονική διαφορά ανάμεσα σε δύο αντικείμενα τύπου
Time
     * Ανάμεσα σε δύο alerts του Snort δηλαδή. Επιστρέφει το
αποτέλεσμα
     * σε μορφή 'δευτερόλεπτα,milliseconds'
     */
    static float difference(Time t1, Time t2) {
        float ans = (t1.hour - t2.hour) * 3600 + (t1.min - t2.min) *
60 + (t1.sec - t2.sec) + (t1.millis - t2.millis) / 1000000;
        if (ans < 0) {
            ans *= -1;
        }

        return ans;
    }

    /*
     * Εκτυπώνει στην οθόνη το μήνα , ημέρα , ώρα , λεπτό ,
δευτερόλεπτο και
     * milisecond που συνέβη το alert
     */
    void print() {
        System.out.println(month + " " + date + " " + hour + " " +
min + " " + sec + " " + millis);
    }
}
```

Data.java

Τώρα που κάθε γραμμή – alert τοποθετείται σε ένα αντικείμενο, θα χρειαστεί ένα νέο αντικείμενο που να περιέχει **όλες τις γραμμές**, δηλαδή όλα τα αντικείμενα τύπου Alert. Το αντικείμενο αυτό είναι το Data.java.



Το παραπάνω αντικείμενο Data θα περιέχει δηλαδή 3 αντικείμενα Alert – ένα για κάθε γραμμή που παράγει το Snort.

Η υλοποίηση είναι η παρακάτω και ουσιαστικά χρησιμοποιείται ένα Java Vector που ουσιαστικά είναι ένας πίνακας μεταβλητού μεγέθους. Έχει δημιουργηθεί και μια μέθοδος (add) η οποία παίρνει ως παραμέτρους την ώρα, την υπογραφή, το src ip και το des ip και δημιουργεί ένα νέο αντικείμενο τύπου Alert και το προσθέτει στον Vector.

Επίσης υπάρχουν τέσσερεις ακόμα Vectors με τιμές Float που χρησιμοποιούνται από τα φίλτρα.

```
package javaids;

import java.util.Vector;

public class Data {

    // Ένας Vector (πίνακας μεταβλητού μεγέθους) που θα περιέχει όλα
    // τα αντικείμενα τύπου Alert
    static Vector<Alert> alertArr = new Vector();
    // Τέσσερεις Vectors: bnra, bhaf, bufp, belief που περιέχουν
    // τιμές τύπου Float. Χρησιμοποιούνται από τους αλγόριθμους -
    // φίλτρα
    static Vector<Float> bnra, bhaf, bufp, belief;


    /*
     * Η μέθοδος αυτή δέχεται ώς όρισμα τα στοιχεία μιας γραμμής
     * Alert και προσθέτει αντίστοιχο αντικείμενο στον Vector με
     * τα alerts.
     */
    static void add(String time, String sig, String src, String dest)
{
        alertArr.addElement(new Alert(time, sig, src, dest));
    }
}
```

LogReader.java

Ο τρόπος αποθήκευσης των δεδομένων στην μνήμη έχει παρουσιαστεί. Σειρά έχει η υλοποίηση ενός νέου αντικειμένου Java που θα διαβάζει το αρχείο που παράγει το Snort και θα τοποθετεί όλα τα δεδομένα στη μνήμη για μετέπειτα επεξεργασία. Το αρχείο αυτό είναι το LogReader.

Περιέχει μεθόδους για validation – επικύρωση ότι το αρχείο που του ζητάμε να διαβάσει είναι όντως ένα αρχείο Snort και όχι κάτι άσχετο και μια μέθοδο που διαβάζει το αρχείο και τοποθετεί τα δεδομένα στη μνήμη.

```
package javaids;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.logging.Level;
import java.util.logging.Logger;

public class LogReader {

    // Οι μεταβλητές του αντικειμένου
    private File file;
    private BufferedReader br;
    private FileReader fr;
    private int CHK = 18;

    LogReader() {
    }

    // Με τη μέθοδο αυτή ορίζουμε το αρχείο που περιέχει τα
    // alerts και θα αναγνωστεί
    void setFile(String input) {
        try {
            this.file = new File(input);
        } catch (Exception ex) {
            System.out.println("Exception in Setting File.");
        }
    }

    // Η μέθοδος αυτή επιβεβαιώνει ότι το αρχείο περιέχει πράγματι
    // alerts του Snort
    boolean validateFile() {
        try {
            fr = new FileReader(file);
            br = new BufferedReader(fr);

            if (CHK == br.readLine().split(" ").length) {
                return true;
            }

        } catch (Exception ex) {
```

```
            System.out.println("Exception while Validating File.");
        } finally {
            try {
                fr.close();
                br.close();
            } catch (IOException ex) {

Logger.getLogger(LogReader.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
        return false;
    }

    // Η μέθοδος αυτή διαβάζει γραμμή γραμμή το αρχείο και τοποθετεί
    // τα στοιχεία του σε αντίστοιχες μεταβλητές
    void parseFile() {
        try {
            fr = new FileReader(file);
            br = new BufferedReader(fr);
            String[] temparr;

            while (true) {
                String str = br.readLine();
                if (str == null) {
                    return;
                }
                temparr = str.split(" ");
                // new Time(temparr[0]);
                // System.out.println(temparr[0]+" "+temparr[3]+"
"+temparr[temparr.length-3]+" "+temparr[temparr.length-1]);
                Data.add(temparr[0], temparr[3],
temparr[temparr.length - 3], temparr[temparr.length - 1]);
            }
        } catch (Exception ex) {
            System.out.println("Error in Parsing File.");
        }

    }

    /**
     * Η μέθοδος αυτή επιστρέφει ένα HashMap με όλα τα Snort
     * Signatures. Χρησιμοποιείται από το φίλτρο UFP !
     */
    HashMap retSigMap() {
        HashMap<String, Integer> hm = new HashMap();
        try {
            fr = new FileReader(file);
            br = new BufferedReader(fr);
            String[] temparr;

            while (true) {
                String str = br.readLine();
                if (str == null) {
                    return hm;
                }
                temparr = str.split(" ");

                Integer temp = hm.get(temparr[3]);
                if (temp != null) {
                    hm.put(temparr[3], temp + 1);
```

```
                } else {
                    hm.put(temparr[3], 1);
                }

            }


        } catch (Exception ex) {
            System.out.println("Error in Parsing File.");
        }
        return null;
    }

    // Βοηθητική μέθοδος
    private void parseLine(String str) {
        //   System.out.println(str);
        String[] s = str.split(" ");

        int len = s.length - 1;
        System.out.println(s[0] + " " + s[3] + " " + s[len - 2] + " "
+ s[len]);
    }
}
```

HAF.java

Στο κεφάλαιο 3.2 παρουσιάστηκε η θεωρία του φίλτρου HAF. Συνοπτικά μας αναφέρει ότι «Υπάρχει λοιπόν μια παράμετρος ℓ η οποία καθορίζει ένα όριο. Πάνω από το όριο αυτό το alert είναι true – positive, ενώ κάτω από αυτό θα θεωρείται false-positive. Για να λειτουργήσει το HAF θα πρέπει να υπολογιστεί η μέση συχνότητα για κάθε υπογραφή. Όταν συμβεί μια επίθεση τότε θα υπολογίζεται η συχνότητα εμφάνισης της υπογραφής αυτής κατά τη διάρκεια της επίθεσης. Αν η συχνότητα εμφάνισης της υπογραφής είναι «φυσιολογική» η υπογραφή αυτή μπορεί με ασφάλεια να αγνοηθεί. Αν η συχνότητα διαφέρει κατά πολύ από τη μέση συχνότητα εμφάνισης τότε ίσως είναι σημαντική και δεν πρέπει να αγνοηθεί.»

Οι μαθηματικοί υπολογισμοί που ζητούνται από τη θεωρία έχουν υλοποιηθεί στο παρακάτω αρχείο. Η τιμή ℓ πρέπει να περαστεί ως παράμετρος στο αρχείο αυτό.

```
package javaids;

import java.util.HashMap;
import java.util.Vector;

public class HAF {

    // Οι μεταβλητές της κλάσσης για την υλοποίηση του φιλτρου HAF
    // Πίνακας με τα alerts (alert-vector) : av
    Vector<Alert> av;

    // Η παράμετρος λ
    int param_l;

    // Ένα HashMap με τις συχνότητες εμφάνισης
    HashMap<String, Integer> freqm;

    // Ο Constructor
    HAF() {
    }

    // Τοποθετούμε το πίνακα με τα Alerts
    void setAlertData(Vector<Alert> v) {
        this.av = v;
    }

    // Η μέθοδος αυτή ορίζει την παράμετρο λ
    void setParameterL(int l) {
        this.param_l = l;
    }

    // Η μέθοδος αυτή υπολογίζει τη συχνότητα
    // εμφάνισης των alerts !
    void calFreq() {
        this.freqm = new HashMap();
        String s;
        for (int i = 0; i < av.size(); i++) {
```

```
            s = av.get(i).getSign();
            Integer I = freqm.get(s);
            if (I == null) {
                freqm.put(s, 1);
            } else {
                freqm.put(s, I + 1);
            }
        }
    }

    /**
     * Η μέθοδος αυτή
     * 1. Αρχικοποιεί το πίνακα (Vector) Data.bhaf
     * 2. Υπολογίζει τη συχνότητα εμφάνισης
     * 3. Χρησιμοποιεί τα στοιχεία αυτά για να εντοπίσει
     *    κατά πόσο κάθε alert είναι True-Positive ή
     *    False-Positive
     */
    void compute() {
        Data.bhaf = new Vector();
        calFreq();
        int size = this.av.size();
        Alert alert, temp;

        float val, val1, val2, nf, b;
        for (int i = 0; i < size; i++) {
            alert = av.get(i);
            val1 = Common.INF;
            val2 = Common.INF;

            if (i > 0) {
                for (int j = i - 1; j >= 0; j--) {
                    temp = av.get(j);
                    if (alert.getSign().equals(temp.getSign())) {
                        //System.out.print(j+" ");
           val1 = Time.difference(alert.getTime(), temp.getTime());
                        break;
                    }
                }
            }
            if (i < size - 1) {
                for (int j = i + 1; j < size; j++) {
                    temp = av.get(j);
                    if (alert.getSign().equals(temp.getSign())) {
            val2 = Time.difference(alert.getTime(), temp.getTime());
                        break;
                    }
                }
            }

            val = Math.min(val1, val2);
            nf = (1 / (1 + val)) / freqm.get(alert.getSign());
            b = (Math.min(nf, this.param_1)) / this.param_1;

            //if(b>0)
            // System.out.println(b);
            Data.bhaf.add(i, b);

        }
    }
}
```

NRA.java

Στο κεφάλαιο 3.1 παρουσιάστηκε η θεωρία του φίλτρου NRA. Συνοπτικά μας αναφέρει ότι πρέπει κανείς να μετρά για κάθε Alert όλα τα υπόλοιπα Alerts στο χρονικό παράθυρο `[ti - t0, ti + t0]` όπου τυγχάνει το source και destination IP να ταυτίζονται. Ο ρυθμός εμφάνισης τέτοιων Alerts τοποθετείται σε μια μεταβλητή και που χρησιμοποιείται ως το όριο πάνω ή κάτω από το οποίο ένας Alert χαρακτηρίζεται True Positive ή False Positive.

```
package javaids;

import java.util.Vector;

public class NRA {

    // 4 αριθμητικές μεταβλητές
    private int config_n0;
    private int config_t0;
    private int sconfig_n0;
    private int sconfig_t0;
    // Ο πίνακας με τα Snort Alerts
    private Vector<Alert> av;

    // Ορίζει την τιμή της n0
    void set_n0(int n0) {
        this.config_n0 = n0;
    }

    // Ορίζει την τιμή της t0
    void set_t0(int t0) {
        this.config_t0 = t0;
    }

    // Ορίζει την τιμή της sn0
    void set_sn0(int n0) {
        this.sconfig_n0 = n0;
    }

    // Ορίζει την τιμή της st0
    void set_st0(int t0) {
        this.sconfig_t0 = t0;
    }

    void setAlertData(Vector<Alert> v) {
        this.av = v;
    }

    // Κάνει τον υπολογισμό του φίλτρου NRA
    void compute() {

        Data.bnra = new Vector();

        boolean flag = true, flag1 = true;
        // Κοιτάζει ένα - ένα τα Alerts
        for (int i = 0; i < av.size(); i++) {
```

```
            // Για κάθε Alert αρχικοποιεί τις μεταβλητές
            int temp = 0;
            int temp1 = 0;
            flag = true;
            flag1 = true;

            if (i > 0) {
                // Για κάθε alert στη θέση i-1 .. και μέχρι το 0
                for (int j = i - 1; j >= 0; j--) {
                    Alert a1 = av.get(i);
                    Alert a2 = av.get(j);

                    // Αν η χρονική διαφορά ανάμεσα στα 2 alerts είναι
μικρότερη από το t0...
                    if (flag && Time.difference(a1.getTime(), a2.getTime())
< this.config_t0) {
                        // Και τα destination IP Address ή τα source IP
Address ή το destination του ενός
                        // είναι το source του άλλου
                        if (a1.dest.equals(a2.dest) ||
a1.src.equals(a2.src) ||
                                a1.dest.equals(a2.src) ||
a1.src.equals(a2.dest)) {
                            // προσθέτουμε 1 στο temp
                            temp++;
                        }
                        // Διαφορετικά δε χρειάζεται να συνεχιστεί το loop
                    } else {
                        flag = false;
                    }

                    // Αυτός ο έλεγχος κοιτά αν ΚΑΙ το source αλλά και το
                    // destination είναι τα ίδια (με όλες τις παραλλαγές)
                    if (flag1 && Time.difference(a1.getTime(),
a2.getTime()) < this.config_t0) {
                        if ((a1.src.equals(a2.src) &&
a1.subD().equals(a2.subD())) ||
                                (a1.src.equals(a2.dest) &&
a1.subD().equals(a2.subS())) ||
                                (a2.src.equals(a1.dest) &&
a2.subD().equals(a1.subS())) ||
                                (a1.dest.equals(a2.dest) &&
a1.subS().equals(a2.subS())));
                            temp1++;
                    } else {
                        flag1 = false;
                    }

                    // Αν κανείς από τους δύο ελέγχους δεν έπιασε κάτι
                    // .. κάνε break
                    if (!flag1 || !flag) {
                        break;
                    }
                }
            }


            // Αρχικοποιούμε ξανά σε true
            flag = true;
            flag1 = true;
```

```
                // Αν δεν έχουμε φτάσει στο τέλος των Alerts
                // Κάνε τους ίδιους ελέγχους, αλλά με το i+1
                // αντί για το i-1
                // Αυτή τη φορά αντί να χρησιμοποιηθούν όμως οι τιμές
                // των config_t0 χρησιμοποιείται το sconfig_t0
                if (i < av.size() - 1) {
                    for (int j = i + 1; j < av.size(); j++) {
                        Alert a1 = av.get(i);
                        Alert a2 = av.get(j);
                        if (flag && Time.difference(a1.getTime(), a2.getTime())
< this.sconfig_t0) {
                            //System.out.println("aa");
                            if (a1.dest.equals(a2.dest) ||
a1.src.equals(a2.src) ||
                                    a1.dest.equals(a2.src) ||
a1.src.equals(a2.dest)) {
                                temp++;
                            }
                        } else {
                            flag = false;
                        }

                        if (flag1 && Time.difference(a1.getTime(),
a2.getTime()) < this.sconfig_t0) {
                            if ((a1.src.equals(a2.src) &&
a1.subD().equals(a2.subD())) ||
                                    (a1.src.equals(a2.dest) &&
a1.subD().equals(a2.subS())) ||
                                    (a2.src.equals(a1.dest) &&
a2.subD().equals(a1.subS())) ||
                                    (a1.dest.equals(a2.dest) &&
a1.subS().equals(a2.subS())));
                            temp1++;
                        } else {
                            flag1 = false;
                        }

                        if (!flag || !flag1) {
                            break;
                        }

                    }
                }

                // Διαιρούμε με το αντίστοιχο n0
                // Για να υπολογίζουμε το belief
                float b = Math.max(temp, this.config_n0) / this.config_n0;
                b = 1 - 1 / b;     // η τιμή του belief
                float b1 = Math.max(temp1, this.sconfig_n0) / this.sconfig_n0;
                b1 = 1 - 1 / b1;

                Data.bnra.add(i, Math.max(b, b1));
                //System.out.print(exp+" "+i);
                //System.out.println(b + " "+ b1);
            }

        }
}
```

UFP.java

Στο κεφάλαιο 3.3 παρουσιάστηκε η θεωρία του φίλτρου UFP. Συνοπτικά μας αναφέρει ότι πρέπει να μετρηθεί η συχνότητα εμφάνισης υπογραφών από Snort Alerts σε περίοδο που δεν συμβαίνει κάποια επίθεση και να συγκριθεί με τη συχνότητα εμφάνισης των υπογραφών. Το εάν ένα Alert είναι True-Positive ή False-Positive ορίζεται από την τιμή του

$$b_i^{ufp} = \frac{min(\frac{f_i^c}{f_i^{af}}, \ell)}{\ell}$$

Το παρακάτω αρχείο υλοποιεί τη θεωρία του φίλτρου και προσφέρει μεθόδους τόσο για τον υπολογισμό των συχνοτήτων (δύο διαφορετικά αρχεία πρέπει να χρησιμοποιηθούν) όσο και για τον υπολογισμό του φίλτρου.

```
package javaids;

import java.util.HashMap;
import java.util.Vector;

public class UFP {

    // Μεταβλητές για το φίλτρο UFP
    Vector<Alert> av;
    // Η παράμετρος λ
    int param_l;
    // Μεταβλητή για τη συχνότητα εμφάνισης υπογραφής
    HashMap<String, Integer> freqm;
    // Μεταβλητή για τη συχνότητα εμφάνισης υπογραφής
    // σε περίοδο χωρίς επιθέσεις (No attack)
    HashMap<String, Integer> noafreqm;

    // Ο Constructor παίρνει ως παράμετρο το όνομα του αρχείου
    // με τα Alerts
    UFP(String FileName) {

        // Διαβάζουμε το αρχείο επειδή το φίλτρο αυτό
        // θέλει ένα signature map από μια περίοδο
        // που δεν γίνονται επιθέσεις. Το τοποθετεί
        // στο No-Attack Frequency map (noafreqm)
        LogReader lg = new LogReader();
        lg.setFile(FileName);
        if (lg.validateFile()) {
            noafreqm = lg.retSigMap();
        } else {
            System.out.println("Invalid File");
        }
    }

    // ορίζει τα δεδομένα που θα περάσουν το φίλτρο
    void setAlertData(Vector<Alert> v) {
        this.av = v;
    }
```

```
    // ορίζει την παράμετρο λ
    void setParameterL(int l) {
        this.param_l = l;
    }

    // Η μέθοδος αυτή υπολογίζει συνολική συχνότητα εμφάνισης των
signatures
    void calFreq() {
        this.freqm = new HashMap();
        String s;
        // Για κάθε alert στο αρχείο
        for (int i = 0; i < av.size(); i++) {
            // Ανακτούμε το signature
            s = av.get(i).getSign();
            // Εντοπίζουμε στο HashMap πόσες φορές έχει εμφανιστεί
            Integer I = freqm.get(s);
            // Αν δεν έχει εμφανιστεί τοποθετούμε την τιμή 1
            if (I == null) {
                freqm.put(s, 1);
                // Αλλιώς προσθέτουμε 1 στον αριθμό των εμφανίσεων
            } else {
                freqm.put(s, I + 1);
            }
        }
    }

    // Η μέθοδος αυτή υπολογίζει
    void compute() {
        Data.bufp = new Vector();
        // καλεί την προηγούμενη μέθοδο για υπολογισμό
        // συχνότητας εμφάνισης
        calFreq();
        int size = av.size();
        float b;
        // Για κάθε Snort Alert..
        for (int i = 0; i < size; i++) {
            String sig = av.get(i).getSign();
            // Κοιτάμε τη συχνότητα εμφάνισης
            // σε περίοδο χωρίς επιθέσεις
            Integer i1 = noafreqm.get(sig);
            // και σε περίοδο ΜΕ επιθέσεις
            Integer i2 = freqm.get(sig);

            if (i1 == null) {
                b = 1;
            } else {
                b = (Math.min((i2 / i1), this.param_l));
                b = b / this.param_l;
                //System.out.print(i2/i1 +" B "+b+" A ");
            }
            Data.bufp.add(i, b);
            //System.out.println(b);
        }
    }
}
```

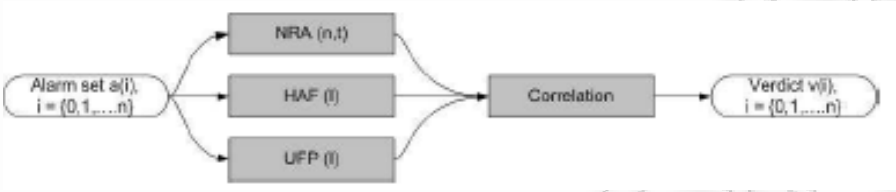
Main.java

Η υλοποίηση των κύριων αντικειμένων και αλγορίθμων έχει παρουσιαστεί. Το μόνο που απομένει είναι το αρχείο το οποίο θα αρχικοποιήσει την εφαρμογή, θα ορίσει την παράμετρο ℓ και θα ορίσει και την τοποθεσία του αρχείου με τα Snort alerts στον σκληρό δίσκο. Την δουλειά αυτή την κάνει το παρακάτω αρχείο:

```
package javaids;

public class Main {

    public static void main(String[] args) {
        // Arguments:
        // @1 - Το πρώτο αρχείο με Alert περιέχει τα στοιχεία προς
        // φιλτράρισμα
        // @2 - Το δεύτερο αρχείο δεν περιέχει Alert επιθέσεων και
        //      δίνεται για τον υπολογισμό του UFP φίλτρου !
        IDSFilter ids = new
IDSFilter("C:\\Snort\\log\\week2_alerts\\alert.ids",
"C:\\Snort\\log\\week1_alerts\\alert.ids");
        ids.setThreshold((float) 1f);
        ids.maxC();

        // Αν θέλουμε να εκτελέσουμε ΜΟΝΟ το HAF ή μόνο το NRA ή μόνο
        // το UFP, τότε κάνουμε σχόλια την γραμμή ids.maxC();
        // και αφαιρούμε τα σχόλια από μια από τις παρακάτω
        //ids.justHAF();
        //ids.justNRA();
        //ids.justUFP();

        // Αν θέλουμε έξτρα πληροφορίες, αφαιρούμε το σχόλιο από
        // την παρακάτω
        ids.print();

        ids.printStat();
    }
}
```

IDSFilter.java

Η υλοποίηση των αντικειμένων και των φίλτρων έχει παρουσιαστεί. Αυτό που απομένει είναι το Correlation. Δηλαδή ο συσχετισμός όλων των αποτελεσμάτων των τριών φίλτρων και η έκδοση του τελικού αποτελέσματος.



Ο μηχανισμός αυτός που αναλαμβάνει το Correlation έχει υλοποιηθεί στο αρχείο IDSFilter.java

```
package javaids;

import java.util.Vector;

public class IDSFilter {

    // Το threshold
    float thres;
    // Ο συνολικός αριθμός από Alerts που θα φιλτραριστούν
    int totalert;
    // Ο αριθμός των True-Positive Alerts
    int truealert;

    // Ο Constructor δέχεται 2 αρχεία ως όρισμα
    // το 1ο είναι προς φιλτράρισμα
    // το 2ο περιέχει στατιστικά κάτω από συνθήκες μη-επίθεσης
    IDSFilter(String FileName, String FileName1) {

        // Αρχικοποιούμε μεταβλητές και διαβάζουμε το αρχείο
        Data.belief = new Vector();
        LogReader lg = new LogReader();
        lg.setFile(FileName);
        if (lg.validateFile()) {
            lg.parseFile();
        }
        totalert = Data.alertArr.size();

        // Αρχικοποιούμε το φίλτρο HAF !
        HAF haf = new HAF();
        haf.setAlertData(Data.alertArr);
        haf.setParameterL(10); // λ = 10
        haf.calFreq();
        haf.compute();

        // Αρχικοποιούμε το φίλτρο NRA !
        NRA nra = new NRA();
        nra.setAlertData(Data.alertArr);
        nra.set_n0(20);   // n0 = 20
        nra.set_t0(30);   // t0 = 30
        nra.set_sn0(10);  // sn0 = 100
        nra.set_st0(200); // st0 = 200
```

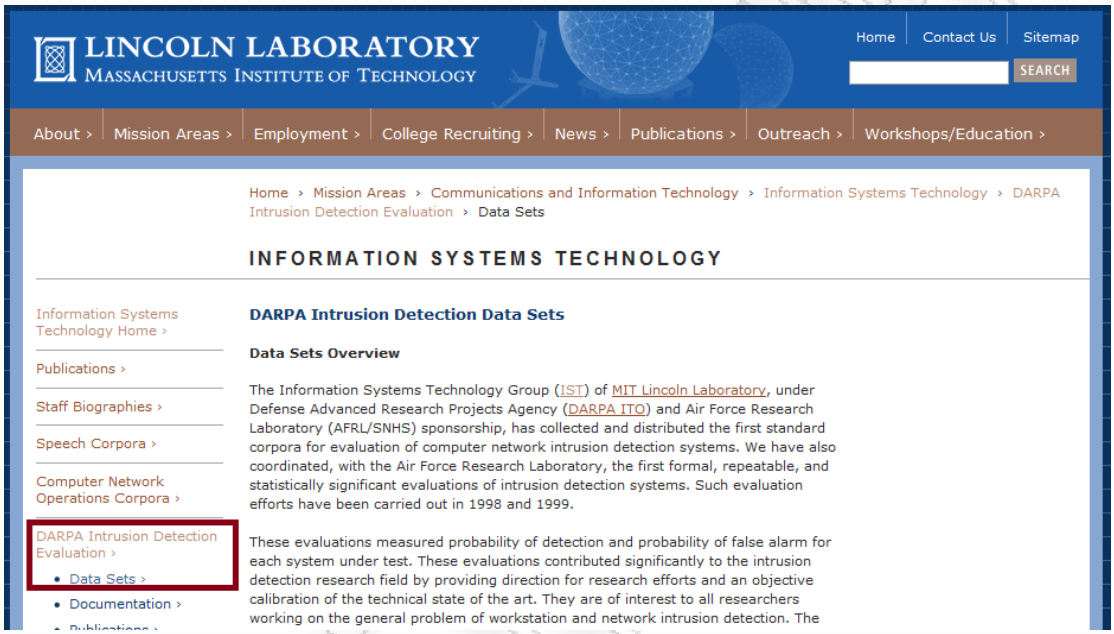
```
            nra.compute();

            // Αρχικοποιούμε το φίλτρο UFP
            // με το αρχείο για στατιστική χρήση
            UFP ufp = new UFP(FileName1);
            ufp.setAlertData(Data.alertArr);
            ufp.setParameterL(100); // λ = 100
            ufp.compute();

        }

        // Κάνει Correlation χρησιμοποιώντας τη Μέγιστη τιμή
        void maxC() {
            for (int i = 0; i < Data.bhaf.size(); i++) {
                // Βρίσκουμε τη μέγιστη τιμή ανάμεσα στα 3 φίλτρα
                float val = Math.max(Math.max(Data.bhaf.get(i),
Data.bnra.get(i)), Data.bufp.get(i));
                // Αν η τιμή αυτή είναι μεγαλύτερη από κάποιο threshold..
                if (val >= this.thres) {
                    truealert++; // προσθέτουμε 1 στα True-Positive
                }
                Data.belief.add(i, val);
            }
        }

        // Αν θέλουμε να τρέξουμε μονό το HAF
        void justHAF() {
            for (int i = 0; i < Data.bhaf.size(); i++) {
                // Βρίσκουμε τη μέγιστη τιμή ανάμεσα στα 3 φίλτρα
                float val = Data.bhaf.get(i);
                // Αν η τιμή αυτή είναι μεγαλύτερη από κάποιο threshold..
                if (val >= this.thres) {
                    truealert++; // προσθέτουμε 1 στα True-Positive
                }
                Data.belief.add(i, val);
            }
        }

        // Αν θέλουμε να τρέξουμε μονο το NRA
        void justNRA() {
            for (int i = 0; i < Data.bnra.size(); i++) {
                // Βρίσκουμε τη μέγιστη τιμή ανάμεσα στα 3 φίλτρα
                float val = Data.bnra.get(i);
                // Αν η τιμή αυτή είναι μεγαλύτερη από κάποιο threshold..
                if (val >= this.thres) {
                    truealert++; // προσθέτουμε 1 στα True-Positive
                }
                Data.belief.add(i, val);
            }
        }

        // Αν θέλουμε να τρέξουμε μονο το UFP
        void justUFP() {
            for (int i = 0; i < Data.bufp.size(); i++) {
                // Βρίσκουμε τη μέγιστη τιμή ανάμεσα στα 3 φίλτρα
                float val = Data.bufp.get(i);
                // Αν η τιμή αυτή είναι μεγαλύτερη από κάποιο threshold..
                if (val >= this.thres) {
                    truealert++; // προσθέτουμε 1 στα True-Positive
                }
                Data.belief.add(i, val);
```

```
        }
    }

    // Κάνει Correlation χρησιμοποιώντας την Ελάχιστη τιμή
    void minC() {
        for (int i = 0; i < Data.bhaf.size(); i++) {
            float val = Math.min(Math.min(Data.bhaf.get(i),
Data.bnra.get(i)), Data.bufp.get(i));
            if (val >= this.thres) {
                truealert++; // προσθέτουμε 1 στα True-Positive
            }
            Data.belief.add(i, val);
        }
    }

    void avgC() {
        for (int i = 0; i < Data.bhaf.size(); i++) {
            float val = (Data.bhaf.get(i) + Data.bnra.get(i) +
Data.bufp.get(i)) / 3;
            if (val >= this.thres) {
                truealert++;
            }

            Data.belief.add(i, val);
        }
    }

    // Εκτυπώνει τα δεδομένα του πινακα belief
    void print() {
        for (int i = 0; i < Data.belief.size(); i++) {
            System.out.println(Data.belief.get(i));
        }
    }

    // Ορίζει τη τιμή του Threshold
    void setThreshold(float val) {
        this.thres = val;
    }

    // Εκτυπώνει στατιστικά στοιχεία
    void printStat() {
        System.out.println("Total alerts detected : " + totalert);
        System.out.println("True alerts detected : " + truealert);
        System.out.println(100 - (((float) truealert / totalert) *
100) + " % of Redection of Alerts.");
    }
}
```

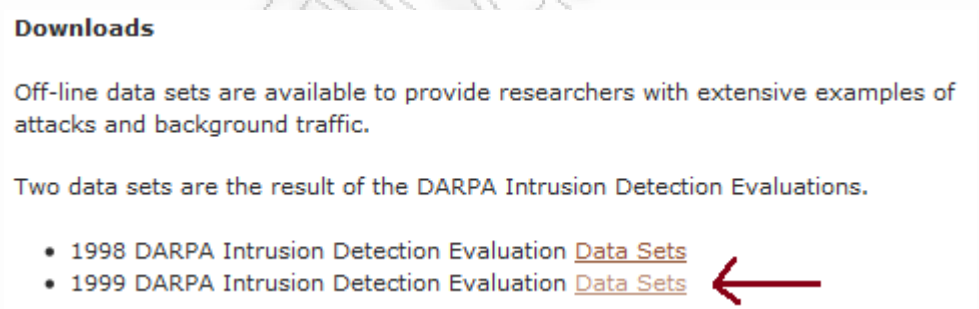
Κεφάλαιο 5. Μετρήσεις

5.1 Κατέβασμα DATA-SETS

Από τη σελίδα:
http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html
επιλέγουμε στη θέση DARPA Intrusion Detection Evaluation την επιλογή 'Data Sets'.

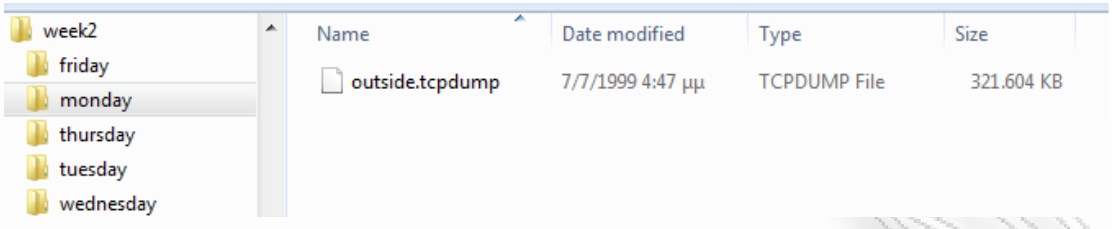


Και στη σελίδα αυτή επιλέγουμε στο Download το Data Set του 1999



Στη σελίδα που μεταφερόμαστε περιέχει λινκ για τη δεύτερη εβδομάδα (που μας ενδιαφέρει), αλλά και για άλλες εβδομάδες. Αρχικά μας ενδιαφέρουν δεδομένα της δεύτερης εβδομάδας, αλλά για το φίλτρο UFP χρειαζόμαστε και δεδομένα από περιόδους χωρίς επιθέσεις (για την εξαγωγή στατιστικών στοιχείων).

Και κατεβάζουμε για κάθε ημέρα της εβδομάδας το αρχείο outside.tcpdump.



Τα αποσυμπιέζουμε και τα τοποθετούμε στο φάκελο

`C:\week2` όπως φαίνεται στην παραπάνω εικόνα. Συνολικά απαιτούν 1,35 GBytes αποθηκευτικού χώρου.

Επίσης κατεβάζουμε το αντίστοιχο αρχείο από την Week 1 – Monday, και τον τοποθετούμε στο φάκελο C:\week1 όπως φαίνεται στην παρακάτω εικόνα.

Ο λόγος που χρειαζόμαστε και το αρχείο της 1ης εβδομάδας είναι ότι περιέχει logs από Attack Free περίοδο που είναι απαραίτητα για το φίλτρο UFP.

5.2 Εγκατάσταση του Snort

Τα αρχεία των data-set αποτελούν tcpdumps. Περιέχουν δηλαδή καταγεγραμμένη όλη την κίνηση του δικτύου κατά την εβδομάδα αυτή. Ο κώδικας που υλοποιήθηκε όμως πρέπει να διαβάσει αρχεία με Snort Alerts. Για να δημιουργηθούν τα Snort Alerts χρειάζεται να ακολουθηθεί μια διαδικασία.

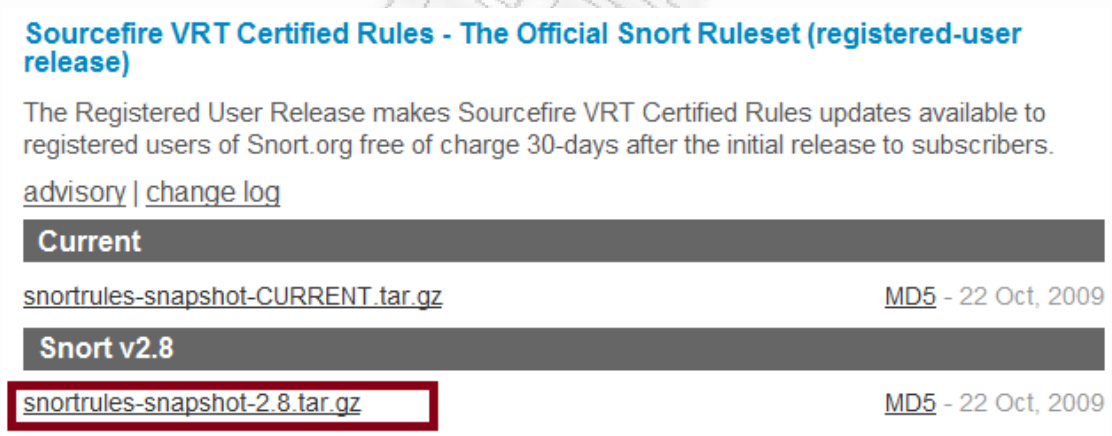
Βήμα 1°: Πρώτα από όλα κάνουμε εγκατάσταση του Snort στο φάκελο C:\Snort

Το κατεβάζουμε από τη σελίδα http://www.snort.org

Βήμα 2°: Κατόπιν κατεβάζουμε τα Snort rules.

Πρέπει να κάνουμε register ένα νέο account στη σελίδα του Snort και να κατεβάσουμε την έκδοση για τους registered users.

Η τρέχουσα έκδοση είναι η 2.8:



Βήμα 3°: Στη συνέχεια αποσυμπιέζουμε τα ανωτέρο Snort-Rules στο φάκελο C:\Snort

Βήμα 4°: Τοποθετούμε στο φάκελο C:\Snort το αρχείο **snort.config**

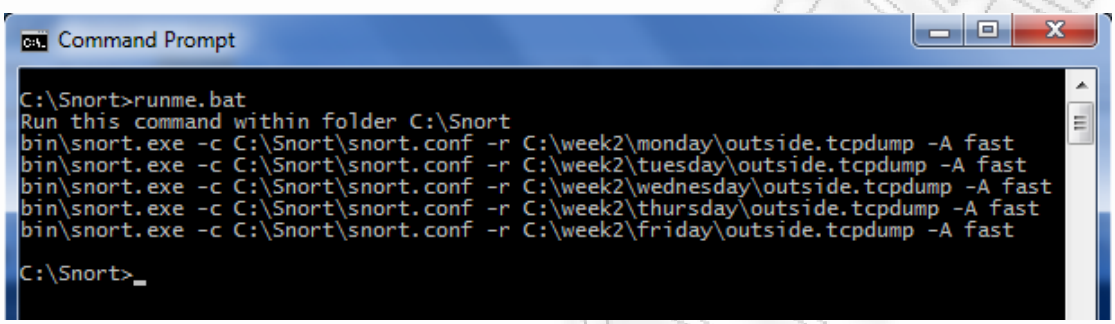
Το αρχείο αυτό περιέχει ορισμένες επιλογές σχετικά με την τοποθεσία που έχουν τοποθετηθεί τα Snort Rules όπως και άλλες επιλογές. Επειδή το Snort μπορεί να τρέχει τόσο σε Windows όσο και σε Linux οι επιλογές αυτές διαφέρουν. Το αρχείο αυτό μπορεί να βρεθεί ως Appendix στη συνέχεια της εργασίας, αλλά και σε μορφή αρχείου στο συνοδευτικό DVD.

5.3 Δημιουργία των Snort Alerts

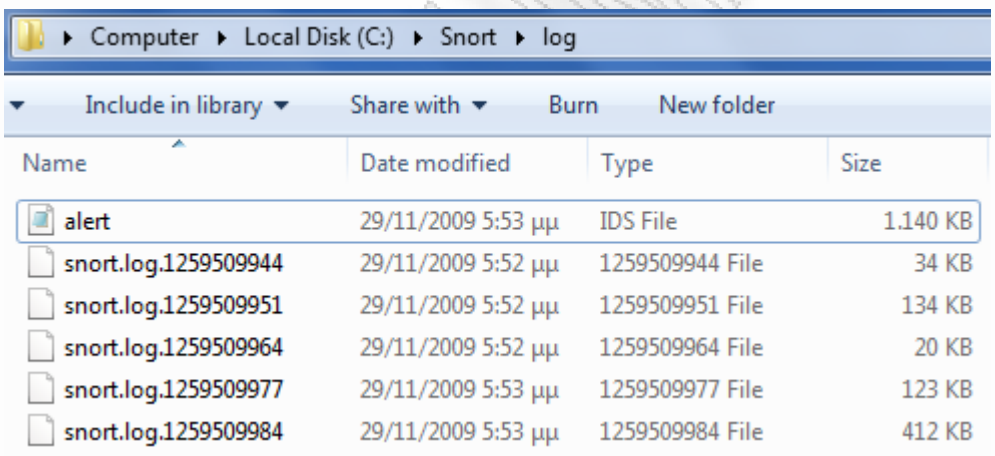
Το περιβάλλον εργασίας είναι έτοιμο. Εκτελούμε λοιπόν διαδοχικά την εντολή

```
bin\snort.exe -c C:\Snort\snort.conf -r αρχείο_tcpdump
```

μέσα στον φάκελο `C:\Snort`



Το Snort κάνει ανάγνωση στα συνολικά 1.35 GByte δεδομένων και χρησιμοποιώντας τους κανόνες (snort-rules) εντοπίζει πιθανά Alerts και τα αποθηκεύει στο φάκελο C:\Snort\log όπως ακριβώς φαίνεται και στην παρακάτω εικόνα.



Οι επιλογές στην εντολή snort.exe είναι οι παρακάτω

Με –c ορίζεται η θέση του configuration file

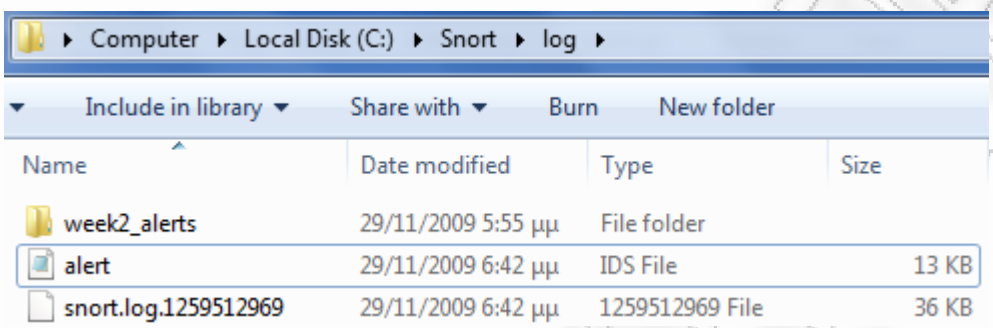
Με –r δίδεται η εντολή στο snort να διαβάσει το αρχείο tcpdump

Με –A fast δίδεται η εντολή να γίνει ταχεία δημιουργία των Alerts

Τα παραπάνω δεδομένα τοποθετήθηκαν στον φάκελο **Week2_Alerts**

Στη συνέχεια θα πρέπει να δημιουργηθούν Snort Alerts από περιόδους χωρίς επιθέσεις. Θα χρησιμοποιήσουμε λοιπόν το outside.tcpdump της 1ης εβδομάδας, ημέρας Δευτέρας για να δημιουργήσουμε το free-of-attack-alert-file.

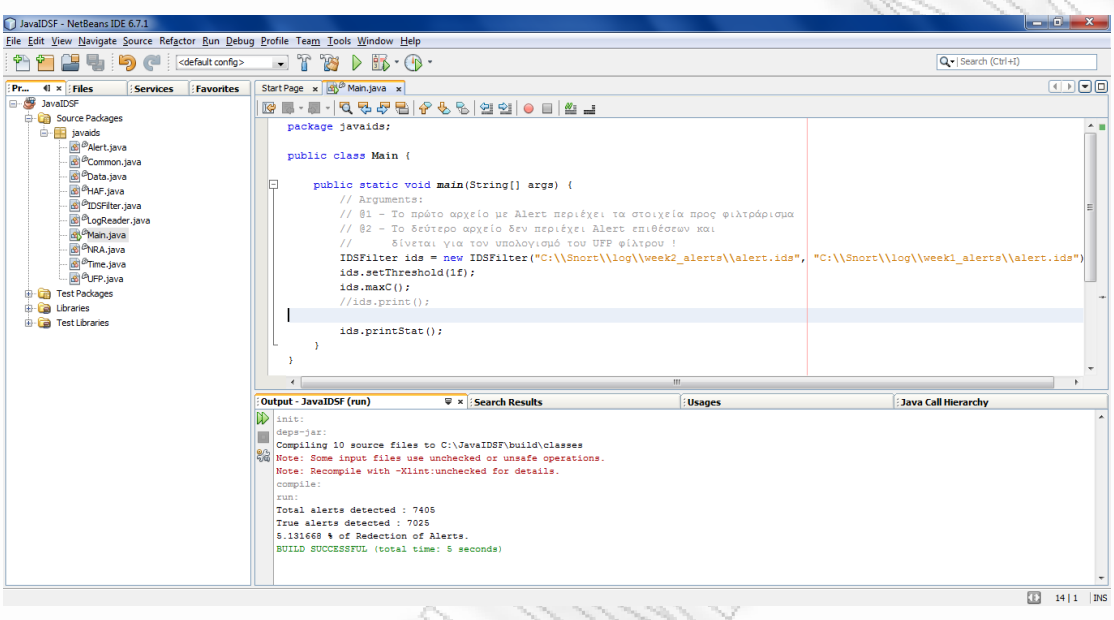
Δημιουργείται λοιπόν ένα νέο αρχείο που αυτή τη φορά δεν περιέχει επιθέσεις.



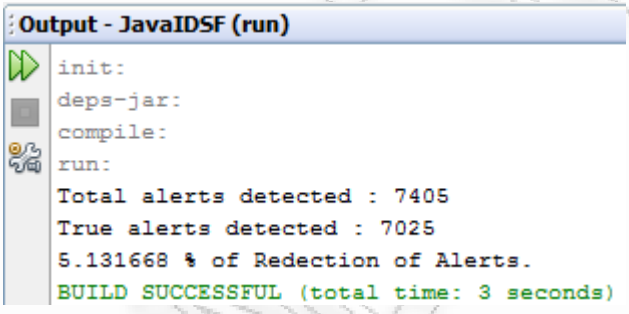
Το τοποθετούμε στο φάκελο week1_alerts.

5.4 Εκτέλεση εφαρμογής

Μέσα από το NetBeans ανοίγουμε το Project και πατάμε F9 (Run). Τα αρχεία Java γίνονται compile και η εφαρμογή εκτελείται εμφανίζοντας τα αποτελέσματα στην οθόνη.



Με τις τρέχουσες ρυθμίσεις τα 3 φίλτρα επιτυγχάνουν μείωση 5,13 % στον αριθμό των Alerts.



Αν θέλουμε να τροποποιήσουμε ορισμένες από τις παραμέτρους όπως την τιμή του λ στο φίλτρο UFP ή HAF ή τις τιμές n0,t0,sn0,st0 στο φίλτρο NRA αυτό γίνεται από το αρχείο IDSFilter.java.

Αν θέλουμε να εκτελέσουμε μόνο 1 από τα φίλτρα για να μελετήσουμε πως συμπεριφέρεται με τις αντίστοιχες τιμές τότε θα πρέπει να τροποποιήσουμε το αρχείο Main.java που περιέχει αναλυτικές οδηγίες.

Εκτέλεση μόνο του φίλτρου HAF

Τιμή λ	Threshold	Ποσοστό μείωσης Alert
10	0.00001	30.911545 %
7	0.000001	1.4584732 %
10	0.000002	2.2147217 %

Εκτέλεση μόνο του φίλτρου NRA

Τιμές μεταβλητών	Threshold	Ποσοστό μείωσης Alert
n0=10, t0=30, sn0=10, st0=200	0.99	43.281567 %
-/-	0.98	36.529373 %
-/-	0.94	20.877785 %
-/-	0.92	8.413231 %
-/-	0.00001	5.374748 %

Εκτέλεση μόνο του φίλτρου UFP

Τιμή λ	Threshold	Ποσοστό μείωσης Alert
1000	1	5.131668 %
100	1	5.131668 %
10	1	1.4989853 %

Κεφάλαιο 6. Συμπεράσματα

Τα φίλτρα που υλοποιήθηκαν στα πλαίσια της πτυχιακής αυτής εργασίας και περιγράφονται αναλυτικά στο «*Reducing false positives in intrusion detection systems*» που δημοσιεύτηκε στο Computers & Security Volume 29, Issue 1, February 2010, Σελίδες 35-44 βασίζονται στην στατιστική παρατήρηση συμπεριφορών είτε σε attack free περιόδους είτε σε μικρά μεσοδιαστήματα καθώς και στη μέση συχνότητα εμφάνισης μιας υπογραφής.

Το κάθε component ξεχωριστά μπορεί να αποδώσει είτε μεγαλύτερη είτε μικρότερη μείωση στον αριθμό των False Alerts αναλόγως των παραμέτρων που θα χρησιμοποιηθούν. Συνδυάζοντας τα τρία ξεχωριστά components το βέλτιστο αποτέλεσμα μπορεί να επιτευχθεί μέσω μιας επιμερισμένης απαλλαγής από False Alerts. Το κάθε component ξεχωριστά να αφαιρεί μόνο αυτά τα Alerts τα οποία με ισχυρή βεβαιότητα (confidence) γνωρίζει ότι δεν αντιστοιχεί σε πραγματικό Alerts.

Μέσα από πειραματισμούς στις τιμές των παραμέτρων του κάθε component μπορεί κανείς να εξάγει ορισμένα συμπεράσματα:

Α) Το φίλτρο **HAF** μπορεί να έχει από πολύ μικρή απόδοση της τάξης του 1-3% έως αρκετά μεγάλη και να προσεγγίσει το 30%. Μεγαλύτερη απόδοση δεν είναι εφικτή καθώς αν οι τιμές των παραμέτρων είναι μεγάλες, τότε η απόδοση του φίλτρου εκτοξεύεται στο 98%. Το ποσοστό αυτό της μείωσης φαίνεται να είναι εξωπραγματικό και λανθασμένο. Επειδή το φίλτρο βασίζεται στη συχνότητα εμφάνισης, αν το παράθυρο που παρατηρούμε είναι μεγάλο, τότε ακόμα και πραγματικά Alerts θα παραγράφονται.

Β) Μέσα από πειραματισμούς με το **NRA**, ενώ το φίλτρο αποδίδει καλά, δεν μπορεί να αναγνωρίσει τα TPs που συσχετίζονται με IP sweep attacks. Η απόδοση του NRA μπορεί να ενισχυθεί με τη χρήση μιας παραλλαγής του συστατικού που αποκαλείται NRAsweep. Αυτό το συστατικό είναι βασισμένο στην ίδια λογική με το NRA , αλλά μαζί με την IP προέλευσης και προορισμού, χρησιμοποιεί και τα υποδίκτυά τους (τα πρώτα τρία μέρη της διεύθυνσης IP).

Επιπλέον το **NRA** είναι το πιο ελαστικό component καθώς αναλόγως των παραμέτρων του, μπορεί να αποκλείσει διαφορετικό ποσοστό Alerts.

Γ) Το φίλτρο **UFP** μπορεί να πετύχει μείωση των Alerts από ένα μικρό ποσοστό μέχρι 5,2%. Όσο μεγάλη και να είναι η περίοδος attack-free που θα χρησιμοποιηθεί τόσο καλύτερο ποσοστό μείωσης μπορεί να επιτευχθεί. Παρόλα αυτά υπάρχει σχετικό άνω όριο στο ποσοστό μείωσης.

Ο μεγαλύτερος προβληματισμός αφορά το κομμάτι **Correlation**. Στο κεφάλαιο 4 έχει παρουσιαστεί το αρχείο IDSFilter.java το οποίο υλοποιεί 3 διαφορετικές μεθόδους για το Correlation. Η μια βασίζεται στο average (στο μέσο όρο των τριών components) η άλλη στην min (ελάχιστη τιμή) και η άλλη στην max (μέγιστη τιμή).

Οι τιμές όμως που επιστρέφει το κάθε component διαφέρουν μεταξύ του σε τάξεις μεγέθους.

Η HAF επιστρέφει πολύ μικρές τιμές της τάξης του 0.0000007

Η NRA επιστρέφει τιμές από 0 μέχρι 0.99

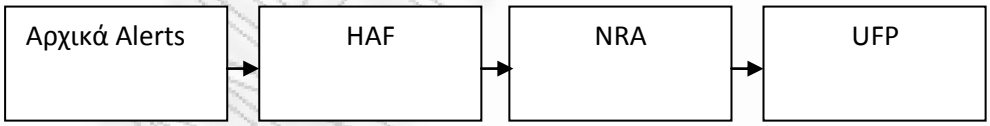
Η UFP επιστρέφει τιμές από 0.01 μέχρι 1

```
HAF:7.142857E-6 NRA:0.9166667 UFP:1.0
HAF:4.761905E-6 NRA:0.9166667 UFP:1.0
HAF:7.142857E-6 NRA:0.9166667 UFP:1.0
HAF:7.142857E-6 NRA:0.9166667 UFP:1.0
HAF:4.761905E-6 NRA:0.9166667 UFP:1.0
HAF:7.142857E-6 NRA:0.9166667 UFP:1.0
HAF:7.142857E-6 NRA:0.9166667 UFP:1.0
HAF:7.142857E-6 NRA:0.9166667 UFP:1.0
```

Αυτό σημαίνει ότι το Correlation που βασίζεται στην μέγιστη ή ελάχιστη τιμή δεν είναι λειτουργικό. Το Correlation που βασίζεται στην μέση τιμή και αυτό δεν είναι λειτουργικό καθώς η τιμή του HAF αγνοείται μπροστά στις τιμές του NRA και του UFP (τι επίδραση έχει η τιμή 0.000000007 όταν βγαίνει ο μέσος όρος της με την τιμή 1;).

Συμπερασματικά για το βέλτιστο correlation θα πρέπει είτε κάθε component να τρέχει σειριακά μετά το άλλο ή να υπάρξει κάποιος συντελεστής βαρύτητας στο αποτέλεσμα του κάθε component έτσι ώστε το κάθε ένα να παίζει το ρόλο του.

1η περίπτωση (Σειριακή χρήση με ξεχωριστά thresholds):



2η περίπτωση (Παράλληλο correlation με συντελεστή βαρύτητας):

Δίνοντας έναν συντελεστή βαρύτητας επί 50000 στις τιμές του HAF και κάνοντας correlation με τη μέση τιμή των 3ων components και με threshold την τιμή 0.9 έχουμε μια μείωση της τάξης του 39%.

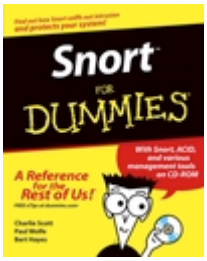
6.1 Μελλοντική δουλειά

Η κατάλληλη μελέτη και παραμετροποίηση του correlation engine όσον αφορά τους συντελεστές βαρύτητας και τις αποδεκτές τιμές των παραμέτρων του κάθε επιμέρους component θα μπορούσε να αποτελέσει έναυσμα για μια μελλοντική πτυχιακή εργασία. Μια τέτοια μελέτη θα απαιτούσε πολλές μετρήσεις σε διαφορετικά data sets.

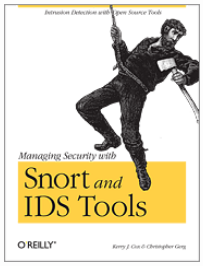
Μάλιστα ένας αλγόριθμος θα μπορούσε να δημιουργηθεί που να εκτελεί διαδοχικά simulations με διαφορετικές τιμές σε κάθε component και να προτείνει μέσα από γραφικό περιβάλλον στον χρήστη να επιλέξει το ποσοστό μείωσης των Alerts που επιθυμεί. Έτσι αν ο χρήστης επέλεγε ότι επιθυμεί 98% μείωση των Alerts οι κατάλληλες τιμές να τοποθετιόντουσαν στα components αλλά και στις τιμές threshold ώστε μόνο το 2% των πιο πιθανών Alerts να φιλτράρονταν και να εμφανιζόντουσαν στον χρήστη.

Μια ακόμη πιθανή μελλοντική επέκταση της εφαρμογής που δημιουργήθηκε θα μπορούσε να είναι η εκτέλεση της σε δεδομένα που δημιουργούνται σε πραγματικό χρόνο και όχι σε στατικά datasets.

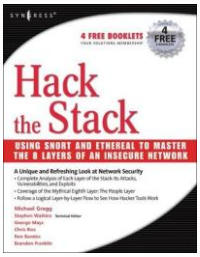
Κεφάλαιο 7. Βιβλιογραφία και πηγές από το Ιντερνετ



Snort For Dummies



(O'Reilly) Managing Security with Snort and IDS Tools



Hack the Stack - Using Snort and Ethereal to Master The 8 Layers of An Insecure Network

Snort :: HomePage
http://www.snort.org/

Snort :: Snort Rules
http://www.snort.org/snort-rules/

Introduction to Snort
http://www.seren.net/documentation/unix%20utilities/Snort.pdf

Snort tutorial
http://luctus.es/wp-content/uploads/2010/03/Snort.ppt

Data Sets from DARPA Intrusion Detection Evaluation.
http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html

Investigating the problem of IDS false alarms: An experimental study using Snort
http://www.springerlink.com/content/e490276x0h804504/

Συστήματα Ανίχνευσης Παρεισφρήσεων IDS
http://digilib.lib.unipi.gr/dspace/bitstream/unipi/2462/3/Michailidis.pdf

ISLAB HACK: Βασικές Έννοιες & Προγραμματισμός του Snort 2.0
http://www.islab.demokritos.gr/gr/html/Snort2_dpritsos/Snort2&Snort_Preprocessors.pdf

SECURITY CONTROLS FOR COMPUTER SYSTEMS: Report of Defense Science Board Task Force on Computer Security
http://cryptome.org/sccs.htm

Συστήματα Ανίχνευσης & Ειδοποίησης Κακόβουλων Ενεργειών Σε Περιβάλλοντα Προσωπικών Δικτύων
http://artemis.cslab.ntua.gr/Dienst/UI/1.0/Download/artemis.ntua.ece/PD2009-0082

Improvement on rules matching algorithm of snort based on dynamic adjustment, Anti-counterfeiting, Security and Identification, 2008. ASID 2008. 2nd International Conference.

Reducing false positives in intrusion detection systems. Georgios P. Spathoulas, and Sokratis K. Katsikas.

TCPDUMP/LIBPCAP public repository
http://www.tcpdump.org

Netbeans Integrated Development Environment (IDE)
http://www.netbeans.org

APPENDIX: Δείγμα του αρχείου με Alerts που χρησιμοποιήθηκε

03/01-15:28:08.917799 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.200.75.201:80 -> 172.16.117.132:6243
03/01-15:28:39.598591 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.200.75.201:80 -> 172.16.117.132:6743
03/01-15:28:52.742581 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.200.75.201:80 -> 172.16.117.132:7211
03/01-15:40:27.458100 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.168.8.10:80 -> 172.16.117.132:9450
03/01-15:42:55.874494 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 131.84.1.34:80 -> 172.16.117.132:10250
03/01-16:07:59.787194 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 209.113.183.114:80 -> 172.16.115.5:23786
03/01-16:08:37.548968 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.115.87:24628
03/01-16:11:31.346180 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.117.103:27245
03/01-16:12:55.995506 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.115.87:29213
03/01-16:19:26.306554 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 209.1.224.190:80 -> 172.16.117.111:7732
03/01-16:19:35.442940 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.117.132:7733
03/01-16:26:14.287577 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 196.32.32.19:80 -> 172.16.117.103:13830
03/01-16:35:13.315298 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.114.169:21141
03/01-16:37:48.681924 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 206.253.217.8:80 -> 172.16.115.87:26099
03/01-16:39:42.706144 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.117.111:29335
03/01-16:44:49.434000 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 209.1.224.190:80 -> 172.16.115.5:5203
03/01-16:52:25.239639 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.115.87:14574
03/01-16:58:08.309574 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.168.8.10:80 -> 172.16.115.5:24827
03/01-17:00:19.362106 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.112.194:29229
03/01-17:10:38.039029 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.115.87:19848
03/01-17:12:33.117554 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.200.75.201:80 -> 172.16.116.194:23622
03/01-17:20:27.450206 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.168.8.10:80 -> 172.16.113.105:5941
03/01-17:39:42.868893 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 161.188.250.169:80 -> 172.16.112.207:26027
03/01-17:48:59.515244 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 208.134.246.202:80 -> 172.16.113.204:10510
03/01-17:51:29.572525 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.200.75.201:80 -> 172.16.117.103:12624
03/01-17:51:55.787306 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.200.75.201:80 -> 172.16.117.103:13744
03/01-17:55:57.008931 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 206.253.217.8:80 -> 172.16.113.204:20559
03/01-17:59:43.591540 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.168.8.10:80 -> 172.16.114.148:24713
03/01-18:33:13.307610 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.114.148:4111

03/01-19:29:31.427605 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 209.1.224.190:80 -> 172.16.115.87:10370
03/01-19:37:37.603416 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.25.71.200:80 -> 172.16.112.207:16960
03/01-19:38:48.352915 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.25.71.200:80 -> 172.16.112.207:18007
03/01-19:39:12.372691 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.25.71.200:80 -> 172.16.112.207:18613
03/01-19:43:07.522857 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 209.67.29.11:80 -> 172.16.113.84:23653
03/01-19:48:00.455786 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.112.207:28825
03/01-20:05:06.275129 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.113.105:18006
03/01-20:07:19.641062 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.200.75.201:80 -> 172.16.113.105:20701
03/01-20:19:52.092673 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 198.3.98.99:80 -> 172.16.113.105:32085
03/01-20:21:05.429964 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.200.75.201:80 -> 172.16.113.105:1346
03/01-20:22:09.066850 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 137.245.85.134:80 -> 172.16.117.132:3726
03/01-20:22:56.725579 [**] [1:1200:10] ATTACK-RESPONSES Invalid URL [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 207.200.75.201:80 -> 172.16.117.52:4190
03/01-20:43:25.346082 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 131.84.1.68:80 -> 172.16.117.52:25252
03/01-20:43:34.287238 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 131.84.1.68:80 -> 172.16.117.52:25316
03/01-20:44:00.345814 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 131.84.1.68:80 -> 172.16.117.52:25386
03/01-20:48:03.542729 [**] [1:1292:9] ATTACK-RESPONSES directory listing [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 172.16.112.100:23 -> 196.227.33.189:6504
03/01-20:48:04.423718 [**] [1:1292:9] ATTACK-RESPONSES directory listing [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 172.16.112.100:23 -> 196.227.33.189:6504
03/01-21:01:17.650297 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 209.185.151.135:80 -> 172.16.117.132:8235
03/01-21:02:17.141226 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 205.175.220.62:80 -> 172.16.112.207:8885
03/01-21:04:11.951764 [**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 205.175.220.62:80 -> 172.16.112.207:11013

APPENDIX: Snort.config

```
#--------------------------------------------------
#   http://www.snort.org     Snort 2.8.5.1 Ruleset
#     Contact: snort-sigs@lists.sourceforge.net
#--------------------------------------------------
# $Id$
#
##################################################
# This file contains a sample snort configuration.
# You can take the following steps to create your own custom configuration:
#
#  1) Set the variables for your network
#  2) Configure dynamic loaded libraries
#  3) Configure preprocessors
#  4) Configure output plugins
#  5) Add any runtime config directives
#  6) Customize your rule set
#
##################################################
# Step #1: Set the network variables:
#
# You must change the following variables to reflect your local network. The
# variable is currently setup for an RFC 1918 address space.
#
# You can specify it explicitly as:
#
# var HOME_NET 10.1.1.0/24
#
# if Snort is built with IPv6 support enabled (--enable-ipv6), use:
#
# ipvar HOME_NET 10.1.1.0/24
#
# or use global variable $<interfacename>_ADDRESS which will be always
# initialized to IP address and netmask of the network interface which you run
# snort at.  Under Windows, this must be specified as
# $(<interfacename>_ADDRESS), such as:
# $(\Device\Packet_{12345678-90AB-CDEF-1234567890AB}_ADDRESS)
#
# var HOME_NET $eth0_ADDRESS
#
# You can specify lists of IP addresses for HOME_NET
# by separating the IPs with commas like this:
#
# var HOME_NET [10.1.1.0/24,192.168.1.0/24]
#
# MAKE SURE YOU DON'T PLACE ANY SPACES IN YOUR LIST!
#
# or you can specify the variable to be any IP address
# like this:

var HOME_NET any

# Set up the external network addresses as well.  A good start may be "any"
var EXTERNAL_NET any

# Configure your server lists.  This allows snort to only look for attacks to
# systems that have a service up.  Why look for HTTP attacks if you are not
# running a web server?  This allows quick filtering based on IP addresses
```

```
# These configurations MUST follow the same configuration scheme as defined
# above for $HOME_NET.

# List of DNS servers on your network
var DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
var SMTP_SERVERS $HOME_NET

# List of web servers on your network
var HTTP_SERVERS $HOME_NET

# List of sql servers on your network
var SQL_SERVERS $HOME_NET

# List of telnet servers on your network
var TELNET_SERVERS $HOME_NET

# List of telnet servers on your network
var FTP_SERVERS $HOME_NET

# List of snmp servers on your network
var SNMP_SERVERS $HOME_NET

# Configure your service ports.  This allows snort to look for attacks destined
# to a specific application only on the ports that application runs on.  For
# example, if you run a web server on port 8180, set your HTTP_PORTS variable
# like this:
#
# portvar HTTP_PORTS 8180
#
# Ports you run web servers on
portvar HTTP_PORTS 80

# NOTE:  If you wish to define multiple HTTP ports, use the portvar
# syntax to represent lists of ports and port ranges.  Examples:
## portvar HTTP_PORTS [80,8080]
## portvar HTTP_PORTS [80,8000:8080]
# And only include the rule that uses $HTTP_PORTS once.
#
# The pre-2.8.0 approach of redefining the variable to a different port and
# including the rules file twice is obsolete.  See README.variables for more
# details.

# Ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# Ports you might see oracle attacks on
portvar ORACLE_PORTS 1521

# Ports for FTP servers
portvar FTP_PORTS 21

# other variables
#
# AIM servers.  AOL has a habit of adding new AIM servers, so instead of
# modifying the signatures when they do, we add them to this list of servers.
var AIM_SERVERS
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205
.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.0/24]

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
```

```
# such as:  c:\snort\rules
var RULE_PATH rules
var PREPROC_RULE_PATH c:\Snort\preproc_rules

# Configure the snort decoder
# ============================
#
# Snort's decoder will alert on lots of things such as header
# truncation or options of unusual length or infrequently used tcp options
#
#
# Stop generic decode events:
#
 config disable_decode_alerts
#
# Stop Alerts on experimental TCP options
#
 config disable_tcpopt_experimental_alerts
#
# Stop Alerts on obsolete TCP options
#
 config disable_tcpopt_obsolete_alerts
#
# Stop Alerts on T/TCP alerts
#
# In snort 2.0.1 and above, this only alerts when a TCP option is detected
# that shows T/TCP being actively used on the network.  If this is normal
# behavior for your network, disable the next option.
#
 config disable_tcpopt_ttcp_alerts
#
# Stop Alerts on all other TCPOption type events:
#
 config disable_tcpopt_alerts
#
# Stop Alerts on invalid ip options
#
config disable_ipopt_alerts
#
# Alert if value in length field (IP, TCP, UDP) is greater than the
# actual length of the captured portion of the packet that the length
# is supposed to represent:
#
 config enable_decode_oversized_alerts
#
# Same as above, but drop packet if in Inline mode -
# enable_decode_oversized_alerts must be enabled for this to work:
#
 config enable_decode_oversized_drops
#

# Configure the detection engine
# ===============================
#
# Use a different pattern matcher in case you have a machine with very limited
# resources:
#
 config detection: search-method lowmem

# Configure Inline Resets
# ========================
#
# If running an iptables firewall with snort in InlineMode() we can now
```

```
# perform resets via a physical device. We grab the indev from iptables
# and use this for the interface on which to send resets. This config
# option takes an argument for the src mac address you want to use in the
# reset packet.  This way the bridge can remain stealthy. If the src mac
# option is not set we use the mac address of the indev device. If we
# don't set this option we will default to sending resets via raw socket,
# which needs an ipaddress to be assigned to the int.
#
# config layer2resets: 00:06:76:DD:5F:E3

##################################################
# Step #2: Configure dynamic loaded libraries
#
# If snort was configured to use dynamically loaded libraries,
# those libraries can be loaded here.
#
# Each of the following configuration options can be done via
# the command line as well.
#
# Load all dynamic preprocessors from the install path
# (same as command line option --dynamic-preprocessor-lib-dir)
#
#dynamicpreprocessor directory c:\snort\lib\snort_dynamicpreprocessor\
#
# Load a specific dynamic preprocessor library from the install path
# (same as command line option --dynamic-preprocessor-lib)
#
# dynamicpreprocessor file c:\snort\lib\snort_dynamicpreprocessor\libdynamicexample.so
#
# Load a dynamic engine from the install path
# (same as command line option --dynamic-engine-lib)
#
#dynamicengine c:\snort\lib\snort_dynamicengine\sf_engine.dll
#
# Load all dynamic rules libraries from the install path
# (same as command line option --dynamic-detection-lib-dir)
#
# dynamicdetection directory /usr/local/lib/snort_dynamicrule/
#
# Load a specific dynamic rule library from the install path
# (same as command line option --dynamic-detection-lib)
#
# dynamicdetection file /usr/local/lib/snort_dynamicrule/libdynamicexamplerule.so
#

dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_dcerpc.dll
dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_dns.dll
dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_ftptelnet.dll
dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_smtp.dll
dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_ssh.dll


##############################################
#########
# Step #3: Configure preprocessors
#
# General configuration for preprocessors is of
# the form
# preprocessor <name_of_processor>: <configuration_options>

# frag3: Target-based IP defragmentation
# --------------------------------------
#
```

```
# Frag3 is a brand new IP defragmentation preprocessor that is capable of
# performing "target-based" processing of IP fragments.  Check out the
# README.frag3 file in the doc directory for more background and configuration
# information.
#
# Frag3 configuration is a two step process, a global initialization phase
# followed by the definition of a set of defragmentation engines.
#
# Global configuration defines the number of fragmented packets that Snort can
# track at the same time and gives you options regarding the memory cap for the
# subsystem or, optionally, allows you to preallocate all the memory for the
# entire frag3 system.
#
# frag3_global options:
#   max_frags: Maximum number of frag trackers that may be active at once.
#           Default value is 8192.
#   memcap: Maximum amount of memory that frag3 may access at any given time.
#         Default value is 4MB.
#   prealloc_frags: Maximum number of individual fragments that may be processed
#                at once.  This is instead of the memcap system, uses static
#                allocation to increase performance.  No default value.  Each
#                preallocated fragment typically eats ~1550 bytes.  However,
#                the exact amount is determined by the snaplen, and this can
#                go as high as 64K so beware!
#
# Target-based behavior is attached to an engine as a "policy" for handling
# overlaps and retransmissions as enumerated in the Paxson paper.  There are
# currently five policy types available: "BSD", "BSD-right", "First", "Linux"
# and "Last".  Engines can be bound to standard Snort CIDR blocks or
# IP lists.
#
# frag3_engine options:
#   timeout: Amount of time a fragmented packet may be active before expiring.
#          Default value is 60 seconds.
#   ttl_limit: Limit of delta allowable for TTLs of packets in the fragments.
#           Based on the initial received fragment TTL.
#   min_ttl: Minimum acceptable TTL for a fragment, frags with TTLs below this
#          value will be discarded.  Default value is 0.
#   detect_anomalies: Activates frag3's anomaly detection mechanisms.
#   policy: Target-based policy to assign to this engine.  Default is BSD.
#   bind_to: IP address set to bind this engine to.  Default is all hosts.
#
# Frag3 configuration example:
#preprocessor frag3_global: max_frags 65536, prealloc_frags 65536
#preprocessor frag3_engine: policy linux \
#                   bind_to [10.1.1.12/32,10.1.1.13/32] \
#                   detect_anomalies
#preprocessor frag3_engine: policy first \
#                   bind_to 10.2.1.0/24 \
#                   detect_anomalies
#preprocessor frag3_engine: policy last \
#                   bind_to 10.3.1.0/24
#preprocessor frag3_engine: policy bsd

#preprocessor frag3_global: max_frags 65536
#preprocessor frag3_engine: policy first detect_anomalies overlap_limit 10

# stream5: Target Based stateful inspection/stream reassembly for Snort
# ---------------------------------------------------------------------
# Stream5 is a target-based stream engine for Snort.  It handles both
# TCP and UDP connection tracking as well as TCP reassembly.
#
# See README.stream5 for details on the configuration options.
```

```
#
# Example config
preprocessor stream5_global: max_tcp 8192, track_tcp yes,                    track_udp yes
preprocessor stream5_tcp: policy first, use_static_footprint_sizes
 preprocessor stream5_udp: ignore_any_rules


# Performance Statistics
# ----------------------
# Documentation for this is provided in the Snort Manual.  You should read it.
# It is included in the release distribution as doc/snort_manual.pdf
#
# preprocessor perfmonitor: time 300 file /var/snort/snort.stats pktcnt 10000

##################################################################
# Step #6: Customize your rule set
#
# Up to date snort rules are available at http://www.snort.org
#
# The snort web site has documentation about how to write your own custom snort
# rules.

#========================================
# Include all relevant rulesets here
#
# The following rulesets are disabled by default:
#
#   web-attacks, backdoor, shellcode, policy, porn, info, icmp-info, virus,
#   chat, multimedia, and p2p
#
# These rules are either site policy specific or require tuning in order to not
# generate false positive alerts in most enviornments.
#
# Please read the specific include file for more information and
# README.alert_order for how rule ordering affects how alerts are triggered.
#========================================

# Rules and include files
include C:\Snort\etc\classification.config
include $RULE_PATH\bad-traffic.rules

include $RULE_PATH\scan.rules
include $RULE_PATH\finger.rules
include $RULE_PATH\ftp.rules
include $RULE_PATH\telnet.rules
include $RULE_PATH\smtp.rules
include $RULE_PATH\rpc.rules
include $RULE_PATH\dos.rules
include $RULE_PATH\ddos.rules
include $RULE_PATH\dns.rules
include $RULE_PATH\tftp.rules

include $RULE_PATH\sql.rules

include $RULE_PATH\icmp.rules


include $RULE_PATH\attack-responses.rules
```

APPENDIX: Τιμές με μόνο το HAF φίλτρο

λ = 10

0.0125
0.0125
3.0058192E-7
3.0058192E-7
1.0748989E-6
2.2686025E-5
2.2686025E-5
6.9300067E-6
6.9300067E-6
7.28295E-7
2.841361E-7
1.6232449E-6
9.921816E-7
3.748126E-6
1.6232449E-6
3.748126E-6
7.509312E-7
1.2742587E-6
7.509312E-7
5.2075197E-6
2.3707918E-5
2.3707918E-5
4.290004E-6
1.954232E-6
1.4530659E-6
5.288767E-7
1.7521727E-6
1.596424E-5
1.596424E-5
6.348843E-7
1.5532774E-5
1.5532774E-5
5.256518E-6
5.256518E-6
9.009009E-4
9.009009E-4
5.3946164E-6
5.3946164E-6
1.1745774E-6
1.0561558E-6
8.273214E-7
1.0561558E-6
8.273214E-7
7.0382885E-6
8.869022E-7
7.0382885E-6
3.7158145E-6
3.7158145E-6
1.161818E-6
5.387931E-5
5.387931E-5
8.0294194E-7
9.863489E-7
8.0294194E-7

9.863489E-7
8.62069E-7
6.952169E-7
6.4766425E-7
5.739474E-7
3.4046957E-7
3.8698494E-7
2.2806057E-6
2.2806057E-6
2.2830216E-7
2.2830216E-7
2.2806057E-6
2.2806057E-6
0.0125
0.0125
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
4.761905E-6
4.761905E-6
4.761905E-6
4.761905E-6
4.761905E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6

7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
3.5714286E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6

7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
2.061558E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6

4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
7.142857E-6
4.761905E-6
7.142857E-6

Για κάθε μια από τις γραμμές του αρχείου με τα Snort alerts δημιουργείται μια τιμή. Η τιμή αυτή αναλόγως με αυτό που θα αποφασίσει ο χρήστης θα οδηγήσει στο συμπέρασμα αν το alert είναι True-Positive ή False-Positive.

APPENDIX: Τιμές με μόνο το NRA φίλτρο

n0 = 20 t0 = 30 sn0 = 100 st0 = 200

0.9285714
0.9285714
0.9285714

0.9285714
0.9285714
0.9285714
0.9285714
0.9285714
0.9285714
0.9285714
0.9230769
0.9230769
0.9230769
0.9230769
0.9230769
0.9230769
0.9230769
0.9230769
0.9230769
0.9230769
0.9166667
0.9166667
0.9166667
0.9166667
0.9166667
0.9166667
0.9166667
0.9166667
0.9166667
0.9166667
0.9090909
0.9090909
0.9090909
0.9090909
0.9090909
0.9090909
0.9090909
0.9090909
0.9090909
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.8888889
0.8888889
0.8888889
0.8888889
0.8888889
0.8888889
0.8888889
0.8888889
0.8888889
0.875
0.875
0.875
0.875
0.875
0.875

0.875
0.875
0.875
0.875
0.85714287
0.85714287
0.85714287
0.85714287
0.85714287
0.85714287
0.85714287
0.85714287
0.85714287
0.85714287
0.8333333
0.8333333
0.8333333
0.8333333
0.8333333
0.8333333
0.8333333
0.8333333
0.8333333
0.8333333
0.8
0.8
0.8
0.8
0.8
0.8
0.8
0.8
0.8
0.8
0.75
0.75
0.75
0.75
0.75
0.75
0.75
0.75
0.75
0.6666666
0.6666666
0.6666666
0.6666666
0.6666666
0.6666666
0.6666666
0.6666666
0.6666666
0.6666666
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.5

0.5
0.5
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.94736844
0.94736844
0.94736844
0.94736844
0.94736844
0.94736844
0.94736844
0.94736844
0.94736844
0.94736844
0.95
0.95
0.95
0.95
0.95
0.95
0.95
0.95
0.95
0.95
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096
0.95238096

APPENDIX: Τιμές με μόνο το UFP φίλτρο

λ = 100

1.0
1.0
0.05
0.05
0.05
0.05
0.05
0.04
0.04
0.04
0.05
0.04
0.12
0.05
0.04
0.05
0.05
0.04
0.05
0.04
0.04
0.04
0.04
0.04
0.04
0.05
0.05
0.05
0.05
0.04
0.04
0.04
0.05
0.05
0.04
0.04
0.04
0.04
0.04
0.04
0.05
0.04
0.05
0.04
0.05
0.04
0.05
0.05
0.05
0.05
0.05
0.04
0.05
0.04
0.05
0.05

0.05
0.04
0.05
0.05
0.04
0.05
0.05
0.05
0.05
0.05
0.05
1.0
1.0
1.0

1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0