Πανεπιστήμιο Πειραιώς

Τμήμα Ψηφιακών Συστημάτων

Π.Μ.Σ. «Ασφάλεια Ψηφιακών Συστημάτων»

Μεταπτυχιακή Διπλωματική Εργασία

ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ & ΑΝΙΧΝΕΥΣΗΣ ΕΙΣΒΟΛΩΝ

ΔΙΑΧΕΙΡΙΣΗ LOGS

Ρούσσος Βασίλειος - MTE1067 - vroussos@freemail.gr

Επιβλέπων Καθηγητής: Δρ. Κωνσταντίνος Λαμπρινουδάκης

Πειραιάς 2012

Αφιερώνεται στην
οικογένεια μου, γιατί
χάρη στις προσπάθειες
τους έχω φτάσει εδώ
που είμαι σήμερα.

Ρούσσος Βασίλειος

Ευχαριστίες

Ευχαριστώ θερμά τον επιβλέποντα καθηγητή μου Δρ. Κωνσταντίνο Λαμπρινουδάκη για την εμπιστοσύνη του ως προς το πρόσωπο μου, παρέχοντας μου τη δυνατότητα να ασχοληθώ σε βάθος με το γνωστικό αντικείμενο της Ανίχνευσης Εισβολών. Η ουσιαστική επιστημονική βοήθεια και υποστήριξη που μου παρείχε σε όλη τη διάρκεια εκπόνησης της παρούσας Διπλωματικής Εργασίας, υπήρξε καθοριστική για την περάτωση της.

ΠΡΟΛΟΓΟΣ

Στον σημερινό «ψηφιακό» κόσμο, της δικτύωσης και της διασύνδεσης, η συνεχής και αλματώδης ανάπτυξη της τεχνολογίας έχει παράλληλα συνοδευτεί και από την αντίστοιχη αύξηση των δικτυακών επιθέσεων. Η τεχνολογία δε μένει ποτέ στάσιμη, ούτε ο τρόπος με τον οποίο οι άνθρωποι «καταναλώνουν» τις πληροφορίες που εμπεριέχονται σε αυτήν. Αποτέλεσμα, οι ολοένα αυξανόμενες απαιτήσεις που σχετίζονται με θέματα ασφάλειας να έχουν επιφέρει στην επιφάνεια σύγχρονους προβληματισμούς σχετικά με τα δικτυωμένα συστήματα. Όσο, τα δίκτυα υπολογιστών χρησιμοποιούνται από όλο και μεγαλύτερα τμήματα του πληθυσμού παγκοσμίως, τόσο η Ανίχνευση Εισβολών θα αναδεικνύεται σε ένα δημοφιλές και διαρκώς εξελισσόμενο επιστημονικό πεδίο εφαρμογής. Τα Συστήματα Ανίχνευσης Εισβολών (*Intrusion Detection Systems - IDS*), αποτελούν μια τεχνολογία που έχει γνωρίσει σημαντική διάδοση τα τελευταία χρόνια, καθιστώντας την ασφάλεια των πληροφοριών μία πρωταρχική και αδιαπραγμάτευτη προτεραιότητα.

Στα πλαίσια της παρούσας διπλωματικής εργασίας, μελετάται και αναλύεται εκτενώς ο τρόπος «αντίληψης» και αντιμετώπισης των Συστημάτων Ανίχνευσης Εισβολών, σε περίπτωση παραβίασης της ασφάλειας ενός υπολογιστικού συστήματος και κατ' επέκταση ενός δικτύου, από ανεπιθύμητες επιθέσεις. Κεντρικός άξονας, ωστόσο, και κύριο πεδίο αναφοράς της συγκεκριμένης εργασίας είναι η διαχείριση των συναγερμών (*alerts*) και των αρχείων καταγραφής (*log files*) που προέρχονται από τα Συστήματα Ανίχνευσης Εισβολών. Αναφέροντας και αναπτύσσοντας τους τύπους των IDSs, καθώς και τους μηχανισμούς ανίχνευσης και ανάλυσης των επιθέσεων, συγκρίνονται μεταξύ τους τα δύο πιο ευρέως διαδεδομένα/χρησιμοποιούμενα IDS. Ειδικότερα, εξετάζονται τα οφέλη και τα μειονεκτήματα χρήσης των Συστημάτων Ανίχνευσης Δικτυακών Εισβολών (*Network-Intrusion Detection Systems*), καθώς και των Συστημάτων Ανίχνευσης Εισβολών βάσει του Host (*Host-Intrusion Detection Systems*).

Μια λογική επέκταση των συστημάτων firewalls (*πρώτη γραμμή άμυνας ενός δικτύου*), η οποία έχει αρχίσει να αναπτύσσεται την τελευταία δεκαετία, είναι τα Συστήματα Ανίχνευσης Δικτυακών Εισβολών ή αλλιώς εν συντομία "NIDS". Καθώς, λοιπόν, τα δίκτυα επιδεικνύουν σήμερα μία εξαιρετική ποικιλομορφία, εξαιτίας του

τεράστιου εύρους των λειτουργιών που καλούνται να επιτελέσουν, ο βασικός στόχος και σημείο ενδιαφέροντος αυτής της διπλωματικής εργασίας, σε συνδυασμό με την διαχείριση των logs, είναι η *μελέτη*, ο *σχεδιασμός* και η *ενσωμάτωση* ενός Συστήματος Ανίχνευσης Εισβολών Δικτύου.

Ένα από τα πλέον δημοφιλή, διαδεδομένα και αποτελεσματικά Συστήματα Ανίχνευσης Δικτυακών Εισβολών είναι το Snort, το οποίο διατίθεται δωρεάν μαζί με τον πηγαίο του κώδικα (*source code*). Παρά την δωρεάν του διάθεση, αποτελεί στις μέρες μας ένα από τα αποτελεσματικότερα συστήματα NIDS και χρησιμοποιείται ως σημείο αναφοράς σε αυτή τη διπλωματική εργασία. Παρόλο, που υπάρχουν και άλλα πακέτα λογισμικών αντίστοιχα του Snort, οι λόγοι που καταλήξαμε σε αυτήν την επιλογή, να ασχοληθούμε δηλαδή με τον χώρο του Ελεύθερου Λογισμικού, αφήνοντας το ευρύ «φάσμα» των ακριβών εμπορικών προϊόντων, αναλύονται λεπτομερειακά στην παρούσα εργασία. Ωστόσο, σημαντικό ρόλο στην επιλογή μας, έπαιξε ο μηχανισμός με τον οποίο το Snort ανιχνεύει και εντοπίζει τις επιθέσεις από κακόβουλους χρήστες, καθώς και τα αντίστοιχα δίκτυα που τις προκαλούν. Η συνεχής βελτίωση και ανάπτυξη του, οφείλεται στο γεγονός ότι οποιοσδήποτε χρήστης, έχοντας την απαραίτητη γνώση και εξειδίκευση πάνω σε θέματα ασφάλειας πληροφοριακών συστημάτων, μπορεί να το προσαρμόσει στις δικές του ανάγκες και να θέσει τους δικούς του κανόνες (*rules*) για το τι θέλει να κάνει το Snort για εκείνον. Τα βασικά ερωτήματα που καθορίζουν και την δομή της παρούσας εργασίας, είναι να γίνει κατανοητός ο τρόπος με τον οποίο ένα IDS ανιχνεύει, δημιουργεί και διαχειρίζεται τους παραγόμενους συναγερμούς και τα log files. Συγκεκριμένα, τι θέλουμε και τι πρέπει να παράγουμε για πετύχουμε τον ανωτέρω σκοπό.

Τέλος, η συγκεκριμένη Διπλωματική Εργασία φιλοδοξεί με την εισαγωγή, τη δημιουργία και την σε βάθος ανάλυση των κανόνων και της λειτουργίας του Snort, να συμβάλλει στο μέγιστο δυνατό βαθμό, σε μία πιο ολοκληρωμένη και αποτελεσματική προσέγγιση ως προς την *ασφάλεια* και τη *διασφάλιση* ενός δικτύου από πιθανές επιθέσεις. Βάζοντας, παράλληλα, τα θεμέλια για μια πλήρη προστασία με την κατασκευή και το «στήσιμο» ενός αποδοτικού, καινοτόμου και ευέλικτου Συστήματος Ανίχνευσης Εισβολών Δικτύου.

ΠΕΡΙΕΧΟΜΕΝΑ

Ευρετήριο Εικόνων

Ευρετήριο Σχημάτων

Ευρετήριο Πινάκων

1 Εισαγωγή

Σε μία εποχή όπου η παγκόσμια ηλεκτρονική συνδεσιμότητα, των ιών και των χάκερ, της ηλεκτρονικής απάτης και παρακολούθησης βρίσκεται στο απόγειο της, η φράση Ασφάλεια των Πληροφοριών είναι ένα από τα πιο καίρια και σύγχρονα ζητήματα στο χώρο της τεχνολογίας και της πληροφορικής. Η έννοια της κατά τη διάρκεια της τελευταίας δεκαετίας έχει επεκταθεί σημαντικά και ο όρος αυτός πλέον δεν αφορά μόνο στην προστασία της πληροφορίας των διαφόρων εταιριών και κυβερνήσεων, αλλά συμπεριλαμβάνει πλέον και τον «απλό» καταναλωτή. Οι πολύ γρήγορα εξελισσόμενες απειλές και η συνεχώς αυξανόμενη εξάρτηση τόσο του δημόσιου όσο και του ιδιωτικού τομέα από τα πληροφοριακά συστήματα προκαλούν άμεσα τις προτεραιότητες μας και τις διαδικασίες που χρησιμοποιούμε με το να καθιστά κρίσιμη και αναγκαία προϋπόθεση το να προστατεύουμε και να διασφαλίζουμε την πληροφορία. Η ασφάλεια των πληροφοριών σημαίνει προστασία των πληροφοριών και των συστημάτων πληροφοριών από μη εξουσιοδοτημένη πρόσβαση, χρήση, διακοπή, αλλαγή, αποκάλυψη, καταστροφή ή την καταγραφή. Ανεξαρτήτως από το τι επιτάσσουν οι διάφορες κανονιστικές απαιτήσεις και τα πρότυπα, η αποτελεσματική προστασία της στον ευρύτερο χώρο της διαχείρισης της ψηφιακής πληροφορίας και της πληροφορικής είναι συνυφασμένη με το συνεχή έλεγχο και τη διαδικασία προσδιορισμού, ελαχιστοποίησης και πρόληψης των κινδύνων ασφάλειας μέσα από μια δομημένη και επαναλαμβανόμενη διαδικασία αξιολόγησης και παρακολούθησης των κινδύνων. Ειδικοί που ασχολούνται με θέματα ασφάλειας της πληροφορίας, σε οποιαδήποτε μορφή και αν αυτή υφίσταται προσπαθούν μέσω τεχνικών αναλύσεων και έρευνας να εντοπίσουν μεθόδους παράκαμψης των ήδη υπαρχόντων μέτρων ασφαλείας και αδυναμίες σε εφαρμογές και συστήματα, αναπτύσσοντας νέα μέτρα προστασίας για να ενισχύσουν την ασφάλεια των συστημάτων ενάντια σε κάθε είδους κακόβουλης επίθεσης. Τα «δίκτυα» αποτέλεσαν ένα εξαιρετικό μέσο για την παράδοση των σχετιζόμενων με την ασφάλεια πληροφοριών στα χέρια των κατάλληλων ανθρώπων. Η αυξημένη πληροφόρηση σημαίνει επίσης αυξημένη ευθύνη, μια αυξημένη ευθύνη που καθιστά την αναγκαιότητα της ασφάλειας σίγουρα απαραίτητη και όχι απλά σημαντική.

2 Η Αναγκαιότητα της Ασφάλειας

Η Ασφάλεια Πληροφοριών είναι μία συνεχής και επαναλαμβανόμενη διαδικασία που στηρίζεται στη συνεχή διαχείριση των σχετικών κινδύνων. Η αποτελεσματικότητα της είναι αποτέλεσμα του τρόπου υλοποίησης των τεχνικών και διαχειριστικών δικλείδων ασφαλείας και της συνεχούς διαδικασίας ελέγχου,παρακολούθησης και επιθεώρησης.

Σύμφωνα με τη θεωρία υπάρχει διαφοροποίηση ανάμεσα σε αυτό που ονομάζουμε συνεχή έλεγχο και επιθεώρηση και σε αυτό που ονομάζουμε συνεχή παρακολούθηση.

- Συνεχής έλεγχος και επιθεώρηση (*continuous audit*) είναι η μέθοδος που χρησιμοποιείται για τη συνεχή αξιολόγηση των κινδύνων αλλά και την αποτελεσματικότητα των υφιστάμενων δικλείδων ασφαλείας.
- Συνεχής παρακολούθηση (*continuous monitoring*) είναι μία διαδικασία διαχείρισης που ελέγχει κατά πόσον οι πολιτικές διαδικασίες και οι επιχειρηματικές διεργασίες λειτουργούν αποτελεσματικά σε συνεχή βάση.

Και τα δύο παραπάνω αποτελούν αναπόσπαστο μέρος της διεργασίας προστασίας της αξιοπιστίας των πληροφοριών (*information assurance*). Η διασφάλιση της αξιοπιστίας των πληροφοριών αφορά στη διαχείριση των κινδύνων που σχετίζονται με τη χρήση, την επεξεργασία, την αποθήκευση και διαβίβαση των πληροφοριών, καθώς και των συστημάτων και διαδικασιών που χρησιμοποιούνται για τους σκοπούς αυτούς.

2.1 Υπηρεσίες Ασφάλειας

Τα Πληροφοριακά Συστήματα στη δυνατότητα τους να «υποστηρίξουν» τα διάφορα χαρακτηριστικά της Ασφάλειας Πληροφοριών είναι λιγότερο ή περισσότερο ευάλωτα σε διάφορους τύπους κινδύνων. Για τον λόγο αυτό και η προσέγγιση της ασφάλειας της πληροφορίας ξεκινάει από την ανάλυση των αναγκών και των σχετικών κινδύνων που παρουσιάζονται και εμφανίζονται σε κάθε περίπτωση. Η μεγαλύτερη πρόκληση στο χώρο της ασφάλειας μ’ αυτήν την ξέφρενη εξέλιξη της τεχνολογίας που ολοένα μοιάζει να δημιουργεί νέα προβλήματα στην ασφάλεια, οφείλεται στην απαίτηση για άμεση εκμετάλλευση των τεχνολογιών αιχμής με σκοπό την αντιμετώπιση των νέων προβλημάτων που συνεχώς αναδύονται. Βασικός στόχος λοιπόν της ασφάλειας

πληροφοριών, παραμένει και είναι η διαφύλαξη της εμπιστευτικότητας, της ακεραιότητας και της διαθεσιμότητας όλων των συστατικών της μερών.

Η έννοια της Ασφάλειας Πληροφοριών προσδιορίζεται από τρεις βασικές συνιστώσες

- **Εμπιστευτικότητα (*Confidentiality*)**

Ως εμπιστευτικότητα ορίζουμε τη διασφάλιση και την απόκρυψη εμπιστευτικών πληροφοριών από μη εξουσιοδοτημένους χρήστες, δηλαδή η πληροφορία πρέπει να είναι διαθέσιμη μόνο σε όσους είναι εξουσιοδοτημένοι. Ευαίσθητα προσωπικά στοιχεία πρέπει να προστατεύονται από παράνομη πρόσβαση. Αυτός ο τύπος ασφάλειας περιλαμβάνει τόσο την προστασία του συνόλου της πληροφορίας όσο και μέρους της, το οποίο μπορεί να οδηγήσει στην αποκάλυψη άλλων σημαντικών πληροφοριών. Για να είναι σε θέση κάποιος οργανισμός ή επιχείρηση να υποστηρίξει την εμπιστευτικότητα θα πρέπει μπορεί να ελέγχει την πρόσβαση στην αποθηκευμένη πληροφορία και να διασφαλίζει ότι κατά τη μεταφορά της, δεν θα πέσει σε «λάθος χέρια». Η εμπιστευτικότητα των δεδομένων επιτυγχάνεται με την επιβολή ειδικών μηχανισμών και συγκεκριμένων διαδικασιών.

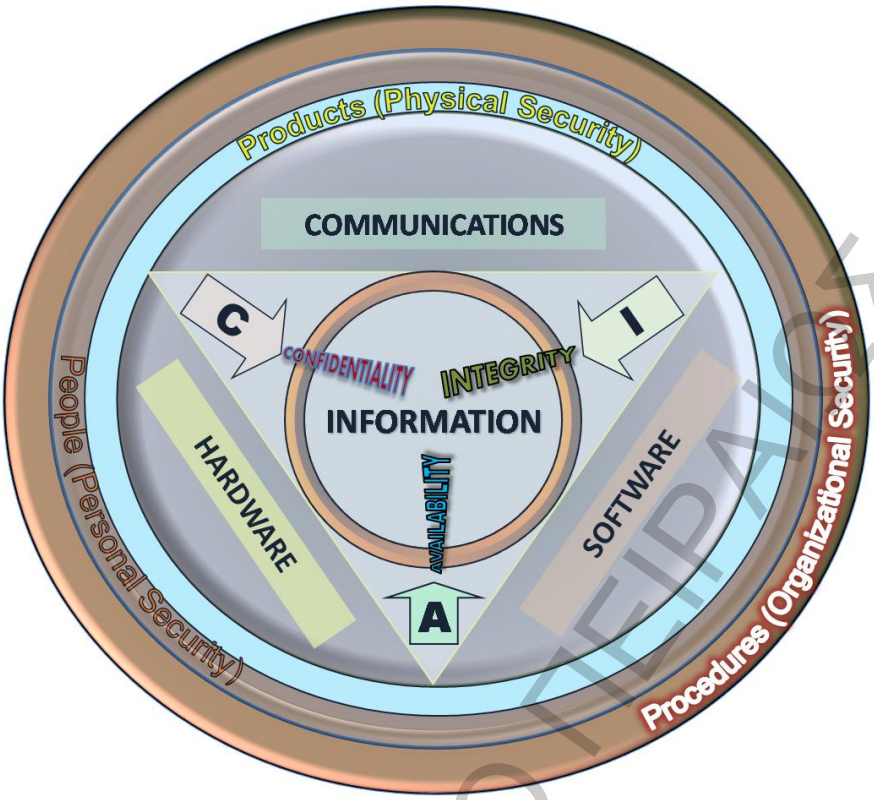
- **Ακεραιότητα (*Integrity*)**

Ακεραιότητα στην ασφάλεια πληροφοριών θεωρούμε τη προστασία και την εξασφάλιση της πληρότητας, ακρίβειας και εγκυρότητας της πληροφορίας, από το σβήσιμο της ή την αλλοίωσή της με οποιονδήποτε τρόπο χωρίς την άδεια του ιδιοκτήτη της. Πιο απλά, η μεταφορά της πληροφορίας μέσω ενός δικτύου μοιάζει με την αποστολή ενός «πακέτου» με το ταχυδρομείο. Το πακέτο μπορεί να διασχίσει διάφορες περιοχές και δίκτυα, έμπιστα ή μη, μέχρι να φτάσει στον τελικό αποδέκτη. Είναι πιθανόν λοιπόν τα δεδομένα να υποκλαπούν ή να αλλάξουν κατά τη μεταφορά, και γι' αυτό υπάρχουν μηχανισμοί και διαδικασίες που εξασφαλίζουν την προστασία της πληροφορίας και των δεδομένων από πιθανή τροποποίηση, που προέρχεται από μη εξουσιοδοτημένους χρήστες. Η ανάγκη για ακρίβεια της πληροφορίας σε ένα περιβάλλον που βασίζεται σ' αυτή, είναι αδιαπραγμάτευτη και η πρωταρχική μέριμνα των αρμοδίων είναι να διασφαλίσουν ότι τίποτα δεν προστίθεται ή αφαιρείται από αυτή, εκτός και αν είναι ηθελημένο ή εξουσιοδοτημένο από το νόμιμο χρήστη της.

- **Διαθεσιμότητα (*Availability*)**

Η διαθεσιμότητα αφορά την προστασία των συστημάτων και κατ' επέκταση τις υπηρεσίες που προσφέρονται μέσω αυτών οι οποίες πρέπει να είναι πάντα διαθέσιμες για χρήση έτσι ώστε να μην υποβαθμιστεί η δυνατότητα παροχής τους. Εξασφαλίζεται έτσι η συνεχής και γρήγορη διανομή των πληροφοριών, καθώς και η συνεχής και γρήγορη απόκριση των πληροφοριακών συστημάτων. Αυτό επιτυγχάνεται με διαδικασίες και μηχανισμούς, που αυξάνουν τη διαθεσιμότητα των πληροφοριακών πόρων και προβλέπουν ανάκτηση των πληροφοριών σε περιπτώσεις όπου τα πληροφοριακά συστήματα υποστούν σοβαρές βλάβες.

Για παράδειγμα, εάν ζητηθεί μια συγκεκριμένη υπηρεσία από νόμιμο χρήστη, αυτή θα πρέπει να του δοθεί άμεσα, σε αντίθετη περίπτωση αυτό ισοδυναμεί με την απώλεια της πληροφορίας που βρίσκεται στο συγκεκριμένο σύστημα. Η συνηθέστερη αιτία μη διαθεσιμότητας είναι η επίθεση *Denial of Service (DoS)* αλλά μπορεί να προκληθεί και από διακοπή διαφόρων υποσυστημάτων ή από προβλήματα του δικτύου και του παρόχου. Στενά συνδεδεμένοι με τη διαθεσιμότητα και πολύ σημαντικοί είναι οι δείκτες της ανταπόκρισης και της αξιοπιστίας. Η ανταπόκριση μετρά την ταχύτητα με την οποία ένα σύστημα ανακάμπτει μετά από μία απρόσμενη διακοπή και η αξιοπιστία μας «δείχνει» ότι το σύστημα λειτουργεί κατά τα αναμενόμενα.



Εικόνα 1: Χαρακτηριστικά της Ασφάλειας των Πληροφοριών

Εκτός από τις παραπάνω αναφερθείσες βασικές συνιστώσες της Ασφάλειας Πληροφοριών υπάρχουν και άλλα συστατικά και συνιστώσες που υπάρχουν, αφορούν και εμπεριέχονται στις υπηρεσίες της Ασφάλειας, αυτές είναι: **1)** ο Έλεγχος (*control*), **2)** η Καταγραφή (*audit*) και **3)** η Σταθερότητα (*consistency*).

1) Έλεγχος (*Control*)

Σημαντικό βήμα για την ασφάλεια αποτελεί ο συνεχής έλεγχος επάρκειας και η επιθεώρηση τήρησης των όσων αφορούν στην προστασία των κρίσιμων πληροφοριών. Ο έλεγχος πρόσβασης στο σύστημα γίνεται για τον εντοπισμό των παράνομων χρηστών και κατ΄ επέκταση παράνομου λογισμικού που μπορεί να δημιουργήσουν μεγάλα προβλήματα στο εκάστοτε πληροφοριακό σύστημα. Η αποτελεσματικότητα ενός προγράμματος ελέγχου είναι μεγαλύτερη όταν έχει συγκεκριμένη κατεύθυνση. Το ίδιο ισχύει και στην περίπτωση του ελέγχου τήρησης των μέτρων προστασίας που αφορούν στις κρίσιμες *πληροφορίες*. Σε ένα καλά σχεδιασμένο και αποτελεσματικό σύστημα ελέγχου, το οποίο με τη χρήση αρχείων

χρηστών, πράξεων και εφαρμογών που χρησιμοποιήθηκαν σε συγκεκριμένες και μη χρονικές στιγμές θα πρέπει να είναι δυνατός ο εντοπισμός προβλημάτων και εν συνεχεία η ανασύσταση του συστήματος, σε κάθε εξέλιξη της τεχνολογίας.

2) Καταγραφή (*Audit*)

Ο διαχειριστής (*administrator*) ενός δικτύου θα πρέπει πάντοτε να ελέγχει εκτός από τους μη νόμιμους χρήστες, δηλαδή τους χρήστες που δεν έχουν άδεια πρόσβασης σε ένα σύστημα και τους νόμιμους χρήστες που μπορεί με κάποιο λάθος τους να προκαλέσουν σκόπιμα ή μη κάποιο πρόβλημα εντός του συστήματος. Σε αυτές τις περιπτώσεις πάντοτε πρέπει να καθορίσουμε τι έχει προκληθεί, από ποιόν και τι επηρεάστηκε στο σύστημα μας. Η λύση για να επιτύχουμε όλα τα παραπάνω είναι να κάνουμε χρήση αρχείων καταγραφής, το λεγόμενο *auditing*. Με τη φράση *χρήση αρχείων καταγραφής*, εννοούμε την καταγραφή (*audit*) της οποιασδήποτε δραστηριότητας στο σύστημα μας, για να το καθιστά ικανό να μας δώσει πληροφορίες για το ποιος έκανε τι και πότε.

3) Σταθερότητα (*Consistency*)

Σταθερότητα ενός συστήματος είναι η διασφάλιση της ορθότητας των δεδομένων και των προγραμμάτων που χρησιμοποιούνται σε ένα πληροφοριακό σύστημα. Σε περίπτωση που υλικό μέρος του συστήματος ή το ίδιο το λογισμικό αρχίσει να «συμπεριφέρεται» ιδιόρρυθμα, εξαιτίας κάποιας μετατροπής, αναβάθμισης ή ακόμη και από εξωγενείς παράγοντες - χρήστες, τότε λέμε ότι η σταθερότητα του συστήματος παύει να υφίσταται, με ενδεχόμενη δυσλειτουργία του και με αποτελέσματα μη αναμενόμενα που τις περισσότερες φορές τείνουν προς την ολική φθορά του.

Αν και όλες οι παραπάνω μορφές ασφάλειας είναι εξίσου απαραίτητες και σημαντικές, πολλοί *οργανισμοί* δίνουν διαφορετική προτεραιότητα στη καθεμία της μορφή διότι αντιμετωπίζουν διαφορετικού είδους απειλές. Για παράδειγμα, σε ένα ακαδημαϊκό περιβάλλον όπως είναι το πανεπιστήμιο, πιο σημαντικά θεωρούνται η *διαθεσιμότητα* και η *ακεραιότητα* της πληροφορίας, ενώ σε ένα περιβάλλον που σχετίζεται με την εθνική ασφάλεια και επεξεργάζεται απόρρητες πληροφορίες, η

εμπιστευτικότητα βρίσκεται σε προτεραιότητα και είναι σημαντικότερη από τη *διαθεσιμότητα*.

2.2 Διαχείριση Ασφάλειας Πληροφοριών

Όπως προείπαμε και παραπάνω η ασφάλεια πληροφοριών είναι μία συνεχής και επαναλαμβανόμενη διαδικασία, με κεντρικό σημείο τη διαχείριση των σχετικών κινδύνων. Η διαχείριση οποιασδήποτε μορφής κινδύνων περιλαμβάνει στάδια προσδιορισμού και αξιολόγησής τους και στη συνέχεια προτάσεις για την υλοποίηση των κατάλληλων δικλείδων ασφαλείας, προκειμένου να επιτευχθεί η ελαχιστοποίηση των κινδύνων. Ο κίνδυνος που αφορά στην ασφάλεια πληροφοριών, είτε χρησιμοποιούνται ποσοτικά είτε ποιοτικά κριτήρια προσδιορίζεται από το γινόμενο της πιθανότητας εκδήλωσης μιας απειλής επί την επίδραση της απειλής αυτής.

Κίνδυνος = (Πιθανότητα εκδήλωσης της Απειλής) x (Επίδραση της Απειλής)

Απειλή (Threat) ασφάλειας πληροφοριών ορίζουμε την πιθανότητα εκμετάλλευσης μιας αδυναμίας ασφάλειας των πληροφοριακών συστημάτων ή των διαδικασιών διαχείρισης της ασφάλειας πληροφοριών από μία συγκεκριμένη εστία απειλών, ικανή να υποκινήσει την εκδήλωση μιας αδυναμίας ασφάλειας και να εκμεταλλευθεί το αποτέλεσμα της. Σύμφωνα με τον παραπάνω ορισμό η απειλή ασφάλειας είναι αποτέλεσμα της εκμετάλλευσης των εκάστοτε κενών - αδυναμιών ασφάλειας. Πιο απλά θα μπορούσαμε να πούμε ότι είναι ένα σύνολο γεγονότων, η εκδήλωση των οποίων μειώνει τη δυνατότητα ενός «οργανισμού» να πετύχει τους στόχους του, όπως είναι η μη εξουσιοδοτημένη πρόσβαση ή η μη εξουσιοδοτημένη τροποποίηση δεδομένων.

Οι κυριότερες κατηγορίες απειλών ασφάλειας των πληροφοριών είναι οι ακόλουθες:

1. Απώλεια Εμπιστευτικότητας των πληροφοριών που διακινούνται, καθώς και των πληροφοριών που αφορούν στα προσωπικά δεδομένα.
2. Απώλεια Ακεραιότητας των πληροφοριών κατά τη διακίνηση τους, με πιθανό αποτέλεσμα τη μη εξουσιοδοτημένη τροποποίησή τους.

3. Απώλεια Διαθεσιμότητας λόγω ακούσιας ή ηθελημένης δυσλειτουργίας των δικτύων επικοινωνίας ή λόγω ηθελημένης ενέργειας, με σκοπό τη διακοπή παροχής των συγκεκριμένων υπηρεσιών. Αποτέλεσμα μιας τέτοιας απειλής είναι η πιθανή αδυναμία λειτουργίας των παρεχόμενων υπηρεσιών, καθώς και η αδυναμία αναζήτησης δεδομένων.
4. Επανάληψη μίας συγκεκριμένης διενέργειας, χρησιμοποιώντας τα ίδια ακριβώς δεδομένα, όπως είναι οι διπλές καταχωρήσεις ή οι διπλές εγγραφές.
5. Άρνηση κάποιου από τα συναλλασσόμενα μέρη ότι διενέργησε ή συμμετείχε σε μία «συναλλαγή».
6. Εξαπάτηση μέσω πλαστοπροσωπίας.
7. Απάτη μέσω εκμετάλλευσης διαφόρων αδυναμιών ασφάλειας σε ένα συγκεκριμένο σύστημα, με αποτέλεσμα την εξαπάτηση ενός οργανισμού από ή μη άτομα του περιβάλλοντος του.

Οι απειλές ασφάλειας πληροφοριών είναι αποτέλεσμα της εκδήλωσης διαφόρων υφιστάμενων αδυναμιών ασφάλειας (*vulnerabilities*) σε τεχνικό και οργανωτικό επίπεδο. Οι απειλές αυτές υφίστανται σε κάθε «δραστηριότητα» και η εκδήλωση τους εξαρτάται από την ύπαρξη και την αποτελεσματικότητα των κατάλληλων δικλείδων ασφαλείας, οι οποίες περιορίζουν την πιθανότητα κάποιας αδυναμίας να εκδηλωθεί και με τη σειρά της να εκδηλώσει την αντίστοιχη απειλή περιβάλλον. Κάθε απειλή είναι συνυφασμένη με μία ή και περισσότερες αδυναμίες ασφάλειας. Οι αδυναμίες αυτές δεν εκδηλώνονται από μόνες τους, αλλά υποκινούνται από εστίες απειλών ικανές να υποκινήσουν την εκδήλωση των αδυναμιών, με αποτέλεσμα την πραγματοποίηση των απειλών ασφάλειας. Οι εστίες απειλών μπορεί να είναι ο ανθρώπινος παράγοντας, κάποια τεχνολογική δυσλειτουργία είτε φυσικά φαινόμενα και διάφορες περιβαλλοντικές απειλές. Οι εστίες αυτές υποκινούν την εκδήλωση των αδυναμιών ασφάλειας μέσω ενός συνόλου ενεργειών, πράξεων, δράσεων και μηχανισμών. Κάθε αδύνατο σημείο που εκδηλώνεται τυχαία ή εκούσια έχει ως αποτέλεσμα την παραβίαση της *Εμπιστευτικότητας*, της *Ακεραιότητας* και της *Διαθεσιμότητας* των πληροφοριών.

Ως **Αδυναμία Ασφάλειας Πληροφοριών (*information security vulnerability*)** ορίζεται η οποιαδήποτε ανεπάρκεια, ευπάθεια ή αδύναμο σημείο:

- Των υφιστάμενων διαδικασιών διαχείρισης ασφάλειας.
- Των υφιστάμενων διαδικασιών ασφάλειας που ακολουθεί ένα πληροφοριακό σύστημα.
- Του σχεδιασμού και των εσωτερικών δικλείδων ασφαλείας του πληροφοριακού συστήματος.
- Της αποτελεσματικότητας των υφιστάμενων τεχνικών και οργανωτικών δικλείδων ασφαλείας.
- Της εναρμόνισης με νομικές και θεσμικές απαιτήσεις.

2.2.1 Διασφάλιση ενός Συστήματος

Για τη διασφάλιση ενός συστήματος ακολουθούμε τα εξής βήματα:

❖ **Αξιολόγηση, Προσδιορισμός και Διαχείριση των Κινδύνων και των Απαιτήσεων**

Για τη σωστή λειτουργία και διασφάλιση ενός συστήματος είναι απαραίτητο σαν πρώτο βήμα να καθορίσουμε το αντικείμενο της ασφάλειας, τι ακριβώς δηλαδή επιδιώκουμε και θέλουμε να προστατεύσουμε. Να γίνει η πλήρης αναγνώριση των απειλών και εν συνεχεία ο σωστός υπολογισμός του κόστους. Το πρώτο βήμα αφορά στη διενέργεια Αξιολόγησης Κινδύνων, προσαρμοσμένη στην απώλεια των κρίσιμων πληροφοριών.

Η συγκεκριμένη διεργασία χρειάζεται να περιέχει τα ακόλουθα:

- ✓ Οριοθέτηση κρίσιμων πληροφοριών.
- ✓ Προσδιορισμός των υφιστάμενων μέτρων προστασίας.
- ✓ Προσδιορισμός κινδύνου και επιπέδου αποδοχής κινδύνου.
- ✓ Προσδιορισμός μέτρων προστασίας.

Τα προσδοκώμενα αποτελέσματα από τη διενέργεια της προσαρμοσμένης αξιολόγησης κινδύνων είναι:

- ✓ Προσδιορισμός των κρίσιμων πληροφοριών.
- ✓ Προσδιορισμός προσβάσεων.

- ✓ Εσωτερική ροή πληροφοριών.
- ✓ Ροή πληροφοριών από το εσωτερικό δίκτυο προς τρίτα μέρη και συνεργάτες.
- ✓ Κίνδυνοι και μέτρα προστασίας κρίσιμων πληροφοριών.

Το αποτέλεσμα της παραπάνω διεργασίας οδηγεί στη χάραξη στρατηγικής και θέτει απαιτήσεις και προτεραιότητες που αφορούν στην προστασία των πληροφοριών.

Καμία πολιτική ασφάλειας δεν έχει ουσιαστικό αντίκρισμα, εάν δεν συνοδεύεται από διαχείριση των κινδύνων σε όλο το φάσμα της «επιχειρηματικής» δραστηριότητας. Η διαχείριση συνίσταται στην αναγνώριση των κινδύνων, στην εκτίμηση της κρισιμότητας τους, στη μείωση των επιπτώσεων με τη χρήση τεχνολογιών ασφάλειας και στην κάλυψη με άλλα μέσα των κινδύνων που παραμένουν ορατοί. Η μείωση των κινδύνων επιτυγχάνεται με την υιοθέτηση βέλτιστων πρακτικών στην προστασία της τεχνολογικής υποδομής, όπως με κανόνες για τη φυσική ασφάλεια της εγκατάστασης, προγράμματα antivirus, ισχυρούς μηχανισμούς ταυτοποίησης, συστήματα προστασίας από παράνομη πρόσβαση, κρυπτογράφηση, αντίγραφα ασφαλείας και με τη διαχείριση έκτακτων περιστατικών.

❖ **Ανάλυση του κόστους απολαβής**

Μετά την ολοκλήρωση της αξιολόγησης και του προσδιορισμού των κινδύνων και των απαιτήσεων μας για ασφάλεια, το επόμενο βήμα είναι η ανάλυση του κόστους απολαβής. Με πιο απλά λόγια, ανάλυση του κόστους απολαβής είναι η αντιστοίχηση κάθε προστατευμένου ή απροστάτευτου αντικειμένου - προϊόντος με κάποιο κόστος. Έτσι, για τα «αντικείμενα» που χρήζουν προστασίας, είναι το κόστος που θα πρέπει να καταβάλουμε για την προστασία τους, ενώ σε αντίθετη περίπτωση η αντιστοίχηση του κόστους είναι η επιβάρυνση οποιασδήποτε ζημιάς εξαιτίας της επιλογής μας για *μηδενική προστασία*. Η επεξεργασία των στοιχείων που προκύπτουν από το σύνολο αυτών των διεργασιών, μας δίνει το αποτέλεσμα που καθορίζει το βαθμό εφαρμογής λύσεων για την προστασία των δεδομένων μας.

❖ Πολιτική Ασφάλειας

Στο τελευταίο βήμα για τη διασφάλιση ενός συστήματος ανήκει η αποδοχή μιας κοινής πολιτικής ασφάλειας. Η πολιτική ασφαλείας ενός δικτύου για να είναι αποτελεσματική πρέπει να είναι ακριβής, ξεκάθαρη και συμβατή με το υπόλοιπο πληροφοριακό σύστημα και να απαντά σε βασικά ερωτήματα, όπως:

- ✓ Ποια είναι τα κρισιμότερα και πιο ευαίσθητα συστήματα.
- ✓ Ποιες πληροφορίες είναι εμπιστευτικές και πρέπει να προστατευτούν.
- ✓ Πως θα επιτευχθεί αυτό.
- ✓ Ποιο σύστημα ταυτοποίησης θα χρησιμοποιηθεί.
- ✓ Ποιος είναι υπεύθυνος για την εγκατάσταση και διαμόρφωση της υποδομής.
- ✓ Υπάρχουν εναλλακτικά σχέδια σε περίπτωση καταστροφής.

Οι μηχανισμοί και τα συστήματα σε υλικό και λογισμικό που χρησιμοποιούνται για τους σκοπούς της Πολιτικής Ασφάλειας Δικτύου περιλαμβάνουν ενδεικτικά τα αναχώματα ασφάλειας (*firewalls*), τα συστήματα ανίχνευσης και προστασίας εισβολών (*IDS/IPS*), τις λίστες ελέγχου πρόσβασης (*access control lists*), τα ιδεατά ιδιωτικά δίκτυα (*virtual private networks*) και τέλος τα ιδεατά τοπικά δίκτυα (*virtual LANs*). Η πολιτική ασφάλειας βοηθά στην αποτελεσματική *διαχείριση του υφιστάμενου κινδύνου* και καλύπτει κάθε επιμέρους σύστημα, εξωτερικές και εσωτερικές απειλές, ανθρώπινους και μηχανικούς παράγοντες, διοικητικές και μη ευθύνες. Είναι αυτή που διασφαλίζει σε πρωταρχικό στάδιο την *ακεραιότητα* και *εμπιστευτικότητα* των πληροφοριών και καθορίζει εν τέλει το γενικό πλαίσιο του τι προσπαθούμε να προστατέψουμε και γιατί.

Συνεπώς, η πολιτική ασφάλειας ενός δικτύου είναι αυτή που δίνει τις γενικές γραμμές και κατευθύνσεις που θα πρέπει να ακολουθηθούν, «υπαγορεύοντας» το σκεπτικό λειτουργίας του εκάστοτε δικτύου και τη θέση όλων των χρηστών αλλά και υπηρεσιών του.

2.3 Ασφάλεια της Πληροφορίας: Μία διαρκής αναθεώρηση

Μετά τον προσδιορισμό των απαιτήσεων της ασφάλειας, ο κάθε οργανισμός ή επιχείρηση πρέπει να εντοπίζει τις τεχνολογίες που θα τους ικανοποιήσουν. Ένα ικανοποιητικό πλαίσιο ασφάλειας υλοποιείται με τη χρήση τεχνολογιών ταυτοποίησης και εξουσιοδότησης, σε συνδυασμό με μία ενιαία πλατφόρμα εφαρμογών, ισχυρούς servers και σύγχρονα συστήματα παρακολούθησης της κυκλοφορίας των δικτύων επικοινωνίας. Ο συνδυασμός αυτός επιτρέπει τον έλεγχο πρόσβασης στις εφαρμογές και στα δεδομένα, με ταυτόχρονη μείωση του κινδύνου από πιθανές επιθέσεις στο εκάστοτε δίκτυο.

Η υλοποίηση της ασφάλειας είναι μια δυναμική διαδικασία και διαρκής αναθεώρηση. Η τεχνολογία αλλάζει διαρκώς, το ίδιο και οι κίνδυνοι. Η εδραίωση εταιρικής κουλτούρας προσανατολισμένης στη διαχείριση του κινδύνου και στην επίτευξη της ασφάλειας δεν επιτυγχάνεται από τη μια μέρα στην άλλη. Απαιτείται η υιοθέτηση ενός αποτελεσματικού συστήματος παρακολούθησης και επαναπροσδιορισμού των υφιστάμενων μέτρων προστασίας, ώστε αυτά να συνεχίσουν να λειτουργούν ομαλά, σωστά και χωρίς κανένα απολύτως πρόβλημα. Αποτέλεσμα όλων αυτών καθιστά αναγκαίο, σημαντικό και απαραίτητο για την υλοποίηση λύσεων ασφάλειας στα δίκτυα και υπολογιστικά συστήματα, τη δημιουργία ενός Συστήματος «Διαχείρισης» και Ανίχνευσης Εισβολών.

"Απέναντι στους έμπειρους στην επίθεση, ο εχθρός δεν ξέρει πως να αμυνθεί. Απέναντι στους επιδέξιους στην άμυνα ο εχθρός δεν ξέρει πως να επιτεθεί"

–Sun Tzu

3 Συστήματα Ανίχνευσης Εισβολών (IDS)

3.1 Εισαγωγή

Τα Συστήματα Ανίχνευσης Εισβολών (*Intrusion Detection Systems*) ή εν συντομία «IDS» όπως συνήθως ονομάζονται στον ευρύτερο χώρο της Ασφάλειας Δικτύων και Συστημάτων, αντιπροσωπεύουν ένα προωθημένο σκεπτικό στο οποίο εμπλέκονται πολλές τεχνολογίες. Ένα μεγάλο ερώτημα που απασχολούσε πάντα τους υπευθύνους ασφαλείας είναι το πως θα αντιληφθεί η επιχείρηση ή ένας οργανισμός αν κάποιος εισέβαλε στο δίκτυο τους. Για να διαπιστωθεί τι ακριβώς συμβαίνει σε αυτές τις περιπτώσεις, χρειάζεται η άμεση εγκατάσταση ενός *Συστήματος Ανίχνευσης Εισβολών*. Τα συστήματα αυτά αναπτύχθηκαν προκειμένου να παρέχουν έγκαιρη προειδοποίηση για τις εισβολές παρακολουθώντας τις εξελίξεις των γεγονότων, ώστε αφενός να ληφθούν τα κατάλληλα αμυντικά μέτρα κατά του επιτιθέμενου, αφετέρου να εξαχθούν χρήσιμα συμπεράσματα από την επίθεση και να περιοριστούν οι ζημιές. Μπορούμε να τα ορίσουμε σαν μία διαδικασία ανίχνευσης λανθασμένης ή ανώμαλης δραστηριότητας εντός του δικτύου, ψάχνοντας για σήματα εισβολής. Ο κύριος σκοπός των συστημάτων αυτών είναι να προστατεύουν ένα δίκτυο από παράνομες επιθέσεις και από προσπάθειες παραβίασης του. Τα συστήματα ανίχνευσης εισβολών αποτελούν ουσιαστικά μηχανές εποπτείας και ελέγχου, όταν εμφανίζεται μία εισβολή, είναι απαραίτητη κάποια απόκριση. Εάν η προσπάθεια εισβολής ανιχνευτεί προτού η επίθεση ολοκληρωθεί, το σύστημα μπορεί να λάβει μέτρα για να αποτρέψει την επιτυχία της επίθεσης. Σε αντίθετη περίπτωση πρέπει να αντιμετωπιστεί η εισβολή. Λόγω των περιορισμών τους, τα συστήματα αυτά απαιτούν η παρακολούθηση από τους διαχειριστές (*administrators*) ασφάλειας να είναι αποτελεσματική. Στα πλαίσια των αρμοδιοτήτων των διαχειριστών συστημάτων περιλαμβάνεται και η προστασία των υπολογιστικών συστημάτων από ποικίλες επιθέσεις. Ένα IDS εκτελείται μόνιμα στο σύστημα και συγκεκριμένα στο παρασκήνιο

(*background*), προειδοποιώντας τον διαχειριστή σε περίπτωση που ανιχνεύσει κάποια ύποπτη κίνηση.

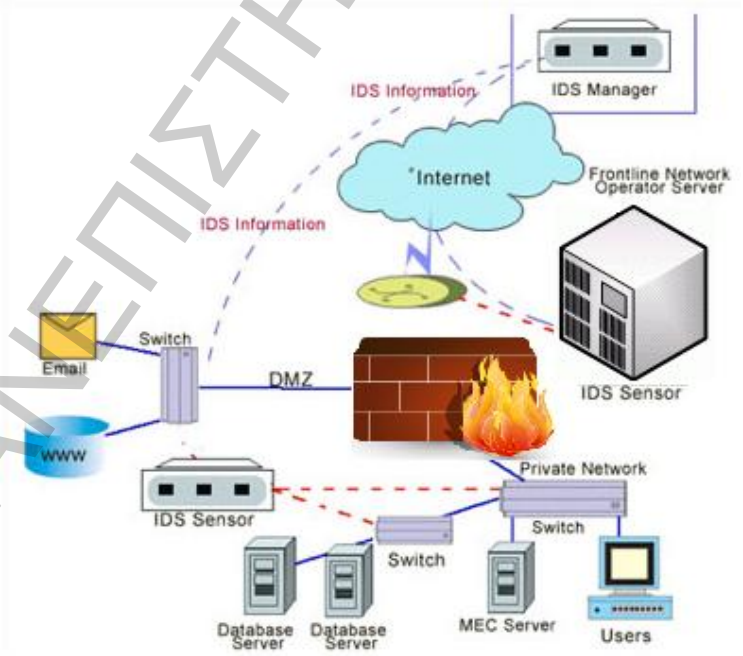
Τα συστήματα πρέπει να παρακολουθούνται με σκοπό την ανίχνευση και καταγραφή όλων των προσπαθειών για επιτυχή αλλά και ανεπιτυχή παραβίαση της ασφάλειας. Καθημερινά τα πληροφοριακά συστήματα «απειλούνται» από καινούριες ευπάθειες με αποτέλεσμα να υπάρχει μεγάλη δραστηριότητα *εισβολών*, όπου οι διαχειριστές ασφάλειας ψάχνουν για πράγματα που δεν έχουν δει προηγουμένως ή για ενδείξεις ότι υφίστανται «επίθεση». Ένα IDS (*Intrusion Detection System*) αποτελεί στα χέρια του διαχειριστή ασφαλείας ένα κατάλληλο εργαλείο που του επιτρέπει να ανιχνεύει άμεσα και να αντιδρά γρήγορα σε μια επίθεση που δέχεται το δίκτυο του. Τα σύγχρονα IDSs βασίζονται στο να προειδοποιούν τους διαχειριστές ενός συστήματος για την παρουσία μιας επίθεσης, γεγονός που τους καθιστά ενεργό κομμάτι των *συστημάτων ανίχνευσης εισβολών*.

3.2 Ο στόχος ενός Συστήματος Ανίχνευσης Εισβολών

Ο στόχος ενός Συστήματος Ανίχνευσης Εισβολών είναι ο έγκαιρος εντοπισμός μιας προσπάθειας παραβίασης και η άμεση άρνηση πρόσβασης στον τομέα εκείνο του δικτύου. Είναι η διάκριση μεταξύ *παράνομης* και *νόμιμης* συμπεριφοράς, περιλαμβάνοντας την αναγνώριση ασυνήθιστων μοτίβων ή δραστηριοτήτων που είναι ήδη γνωστό ότι συνδέονται με απόπειρες εισβολής. Επομένως, ένα σύστημα ανίχνευσης εισβολών μπορεί να οριστεί σαν ένα σύστημα ταξινόμησης το οποίο μπορεί και αναλύει τις συμπεριφορές του συστήματος ή των γεγονότων ασφαλείας και αναγνωρίζει κάθε είδους κακόβουλες συμπεριφορές και επιθέσεις. Αυτό που μπορούμε να πούμε με βεβαιότητα είναι ότι στην πραγματικότητα τα Συστήματα Ανίχνευσης Εισβολών έχουν γίνει τόσο σημαντικά για την ασφάλεια των δικτύων όσο σημαντικά είναι και τα **firewalls** (*πύρινο τείχος*). Αυτοί οι μηχανισμοί, όπως τα firewalls είναι αδιαμφισβήτητα αναγκαίοι και αποτελούν αναπόσπαστο μέρος της υποδομής ασφαλείας, εμπεριέχουν όμως από τη φύση τους ένα βασικό μειονέκτημα, ότι τίθενται σε ισχύ αφού η επίθεση έχει ήδη αρχίσει να πραγματοποιείται.

Όπως φαίνεται και από την ονομασία τους αυτά τα συστήματα ανίχνευσης εισβολών έχουν σαν κύριο στόχο την ανίχνευση επιθέσεων που προέρχονται από το εσωτερικό

του προστατευόμενου δικτύου και που έχουν διαπεράσει την *πρώτη γραμμή άμυνας* του, η οποία κατά κανόνα απαρτίζεται από τα firewalls. Τα συστήματα αυτά σε περιπτώσεις απειλών εφαρμόζουν μια τακτική αντίδρασης. Ενισχύουμε όσο μπορούμε την πρώτη γραμμή άμυνας ώστε να μην επιτρέπεται η παράνομη πρόσβαση στο εσωτερικό μας δίκτυο, αλλά αν αυτό αποτύχει, τότε «καλείται» η *δεύτερη γραμμή άμυνας*. Η δεύτερη γραμμή άμυνας ενός συστήματος είναι ένας ανιχνευτικός μηχανισμός εισβολών, αν η εισβολή ανιχνευθεί σχετικά γρήγορα, ο εισβολέας μπορεί να εντοπιστεί και να «αποβληθεί» από το σύστημα πριν προκαλέσει κάποιου είδους ζημιά ή πριν εκτεθούν σε κίνδυνο τα δεδομένα. Ακόμα και αν η ανίχνευση δεν είναι έγκαιρη για να προλάβει τον εισβολέα, όσο συντομότερα ανιχνευθεί η εισβολή τόσο λιγότερη ζημιά θα γίνει, με αποτέλεσμα να επιτευχθεί ανάκαμψη γρηγορότερα. Στο χώρο της ασφάλειας όμως, οφείλουμε να είμαστε όσο το δυνατόν πιο προνοητικοί γίνεται, για αυτό και είναι αναγκαία η εγκατάσταση και η χρήση εργαλείων που να μπορούν να μας ειδοποιήσουν για μία επικείμενη επίθεση προτού εκείνη κλιμακωθεί, ενδεχομένως ακόμη και να διαθέτουν τη δυνατότητα να προβούν σε ενέργειες προκειμένου να αποφευχθεί η περάτωση τους.



Σχήμα 1: Μια Τυπική Εγκατάσταση Ενός Συστήματος Ανίχνευσης Επιθέσεων (IDS)

Τα συστήματα ανίχνευσης εισβολών (IDS) εξυπηρετούν τρεις βασικές λειτουργίες ασφαλείας: να *παρακολουθούν*, να *ανιχνεύουν* και να *ανταποκρίνονται* σε μη εξουσιοδοτημένη δραστηριότητα διείσδυσης από κατόχους εμπιστευτικών πληροφοριών. Στην πιο απλή του μορφή του ένα σύστημα IDS αναλύει τα αρχεία καταγραφής (*log files*) και ελέγχου (*audit*) του συστήματος και προσπαθεί να εντοπίσει ίχνη από επιθέσεις εισβολής, που γνωρίζει ή που έχουν πραγματοποιηθεί. Μια *εισβολή* μπορεί να θεωρηθεί ως ένα σύνολο ενεργειών με σκοπό να παραβιαστεί η *εμπιστευτικότητα*, η *ακεραιότητα* και η *διαθεσιμότητα* οποιουδήποτε πόρου ενός συστήματος.

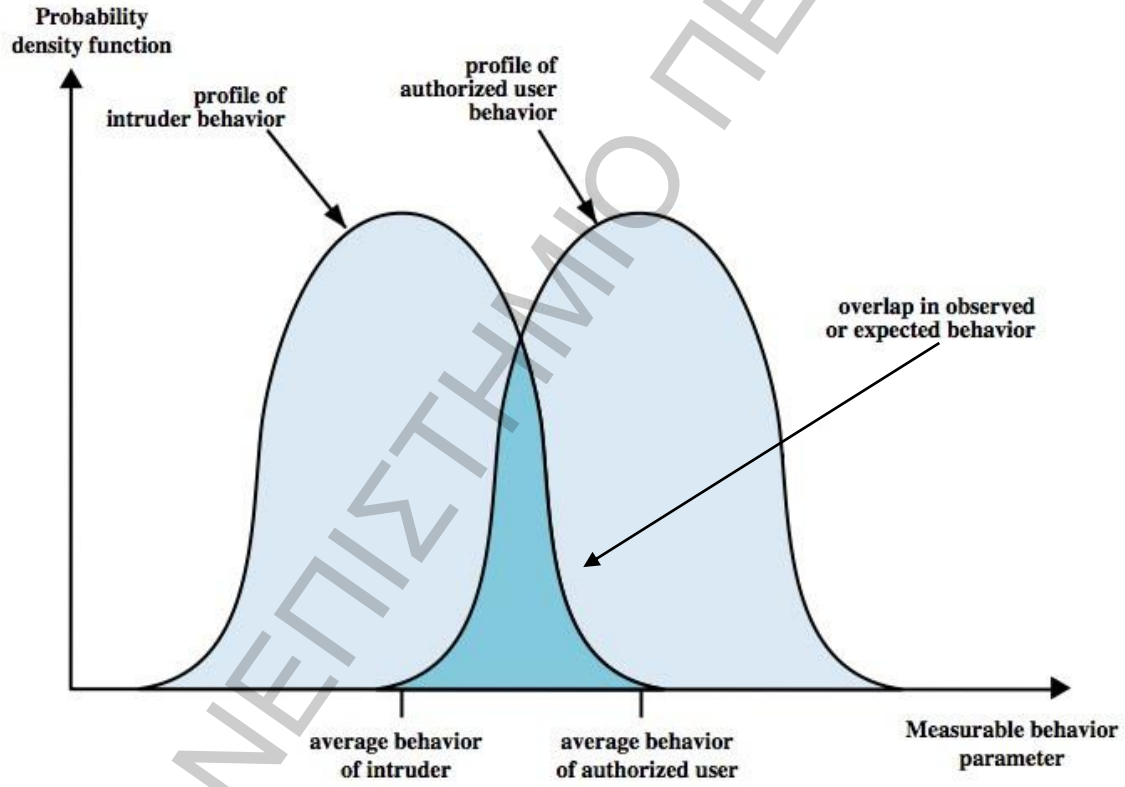
Ο στόχος ενός εισβολέα

Το σκεπτικό και ο στόχος ενός εισβολέα είναι να αποκτήσει πρόσβαση σε ένα υπολογιστικό σύστημα ή να μεγαλώσει την εμβέλεια των «προνομίων πρόσβασης» σε αυτό. Ως επί το πλείστον, αυτό απαιτεί από τον εκάστοτε εισβολέα (*attacker*) να αποκτήσει πληροφορίες οι οποίες είναι προστατευμένες και να εξασκήσει προνόμια που αντιστοιχούν σε ένα νόμιμο χρήστη. Η ανίχνευση εισβολών βασίζεται στην υπόθεση ότι η συμπεριφορά ενός εισβολέα διαφέρει από εκείνη ενός νόμιμου χρήστη με τρόπους που μπορούν να *αυξηθούν ποσοτικά*. Τις περισσότερες φορές, δεν υπάρχει μια ξεκάθαρη και ακριβής διάκριση μεταξύ της επίθεσης ενός εισβολέα και της κανονικής χρήσης πόρων από έναν εξουσιοδοτημένο – νόμιμο χρήστη. Αντιθέτως, υπάρχουν κάποια κοινά σημεία. Η *Εικόνα 2*, δείχνει τη φύση της αποστολής που πρέπει να αντιμετωπίσει ένας σχεδιαστής ενός Συστήματος Ανίχνευσης Εισβολών (*IDS*). Αν και η τυπική συμπεριφορά ενός εισβολέα διαφέρει από την τυπική συμπεριφορά ενός εξουσιοδοτημένου χρήστη, υπάρχει μια επικάλυψη σε αυτές τις «συμπεριφορές».

Σε μελέτη που είχε γίνει από τον **James P. Anderson**, διατυπώθηκε η θεώρηση ότι θα μπορούσε κανείς με εύλογη σιγουριά να ξεχωρίσει ένα μεταμφιεσμένο εισβολέα από ένα νόμιμο χρήστη. Τα «μοτίβα» συμπεριφοράς των νόμιμων - εξουσιοδοτημένων χρηστών μπορούν να επαληθευτούν με ανάκληση του ιστορικού και να ανιχνευθούν περιπτώσεις σημαντικής απόκλισης από αυτά. Σύμφωνα με τον J. Anderson, η διαδικασία ανίχνευσης ενός «νόμιμου» χρήστη που ενεργεί με μη εξουσιοδοτημένο

τρόπο είναι αρκετά δύσκολη, επειδή η διάκριση μεταξύ αντικανονικής και κανονικής συμπεριφοράς είναι σχεδόν πάντα μικρή. Επίσης, κατέληξε ότι τέτοιου είδους παραβιάσεις θα ήταν μη ανιχνεύσιμες αν γινόταν μόνο έρευνα για ανώμαλη συμπεριφορά. Η παράνομη συμπεριφορά θα μπορούσε παρόλα αυτά να είναι ανιχνεύσιμη από μία κατηγορία συνθηκών που υποδηλώνουν μη εξουσιοδοτημένη χρήση.

Οι αναφερθείσες παρατηρήσεις, έγιναν από τον *James P. Anderson* [Ande80] πριν από 22 χρόνια και συγκεκριμένα τον Φεβρουάριο του 1980 και παραμένουν μέχρι σήμερα αληθείς.



Εικόνα 2: Απεικόνιση του προφίλ συμπεριφοράς των εισβολέων με το αντίστοιχο των εξουσιοδοτημένων χρηστών

Οι δύο κατηγορίες εισβολέων σε ένα IDS που μπορούν να καταλύσουν την ασφάλεια ενός συστήματος:

- **Εξωτερικοί εισβολείς :** Οι περισσότεροι ειδικοί στην ασφάλεια, θεωρούν την κατηγορία αυτή των εισβολέων ως τη μεγαλύτερη απειλή για την ασφάλεια των συστημάτων.
- **Εσωτερικοί εισβολείς :** Έρευνες στις Ηνωμένες Πολιτείες της Αμερικής (Η.Π.Α.) έδειξαν ότι το 70% των εισβολών και των επιθέσεων σε ένα πληροφοριακό σύστημα προέρχεται από το εσωτερικό μιας επιχείρησης ή ενός οργανισμού με συχνά απρόβλεπτες επιθέσεις. Πολλοί λένε πως αυτό είναι κάτι το φυσιολογικό, αφού οι νόμιμοι εσωτερικοί χρήστες του συστήματος γνωρίζουν καλύτερα τους τρόπους που επιλέγονται για την προστασία του και μπορούν να εκμεταλλευθούν τη δομή, τους πόρους και τα πολύτιμα δεδομένα του, πολύ πιο εύκολα από κάποιον εξωτερικό χρήστη.

Τα περισσότερα συστήματα ανίχνευσης επιθέσεων (IDS) βασίζουν τη λειτουργία τους στην ανάλυση των ελέγχων ορθότητας του εκάστοτε λειτουργικού τους συστήματος. Ένα IDS μπορεί να ασκεί τον δικό του έλεγχο, συγκεντρώνοντας ένα σύνολο στατιστικών που καταγράφουν και σκιαγραφούν το προφίλ της χρήσης του συστήματος. Αυτά τα στατιστικά στοιχεία ενημερώνονται συνεχώς ώστε να αντανακλούν τη τρέχουσα κατάσταση του συστήματος, εξάγοντας τα από μία μεγάλη ποικιλία πηγών, με την σημαντικότερη και πιο αποδοτική να είναι η *διαχείριση καταχωρίσεων καταγραφής logs*.

Ανεξαρτήτως από τον εκάστοτε μηχανισμό στον οποίο βασίζεται και στηρίζεται ένα IDS πρέπει απαραιτήτως να ανταποκρίνεται στις ακόλουθες απαιτήσεις:

1. Πρέπει να εκτελείται συνεχώς στο background, με ή χωρίς την επιτήρηση του διαχειριστή, ώστε να αυξάνεται η αξιοπιστία και η λειτουργία του συστήματος.
2. Πρέπει να είναι ανθεκτικό, ώστε έπειτα από οποιαδήποτε δυσλειτουργία ή αδυναμία του συστήματος να είναι σε θέση να αντεπεξέλθει άμεσα χωρίς να χρειάζεται να ξανά «καλέσει» την βάση δεδομένων (*database*) του.
3. Πρέπει να καταναλώνει αμελητέες ποσότητες υπολογιστικών πόρων, ώστε να μην επιβαρύνει το σύστημα και να ελέγχει τις λειτουργίες του σε συνεχή βάση.

4. Πρέπει να είναι προσαρμοσμένο για τις ανάγκες του εκάστοτε συστήματος που το περιλαμβάνει και χρησιμοποιείται.
5. Πρέπει να μην γίνεται «εύκολα» η παραβίαση του από κάποιον κακόβουλο χρήστη. Πιο συγκεκριμένα, ένα IDS σχετίζεται άμεσα με δύο τύπους λαθών που μπορεί να του συμβούν κατά τη διάρκεια της λειτουργίας του και διακρίνονται σε *ψευδή θετικά* και *αρνητικά*.

- **False Positive**

Το ψευδές θετικό (*false positive*) είναι ένα είδος λάθους που συμβαίνει όταν το σύστημα χαρακτηρίζει μια νόμιμη πράξη σαν μια πιθανή εισβολή χωρίς να υφίσταται επίθεση.

- **False Negative**

Σε αντίθεση με το ψευδές θετικό, ένα ψευδές αρνητικό (*false negative*) λάθος δημιουργείται όταν έχει πραγματοποιηθεί μια εισβολή και το σύστημα επιτρέπει την εξέλιξη της θεωρώντας την σαν μια νόμιμη πράξη. Το αποτέλεσμα του συγκεκριμένου τύπου λάθους είναι να δημιουργεί μια ψευδαίσθηση ασφάλειας που το καθιστά επικίνδυνο και σοβαρότερο από ένα ψευδές θετικό.

3.3 Γιατί Συστήματα Ανίχνευσης Εισβολών (IDS)

Υπάρχουν αναρίθμητοι λόγοι για τους οποίους μία επιχείρηση ή ένας οργανισμός χρειάζεται να εγκαταστήσει και να θέσει σε λειτουργία ένα *σύστημα ανίχνευσης εισβολών*. Δεν θα πρέπει ο εκάστοτε οργανισμός να μένει στην επιθυμία της εγκατάστασης αλλά στην υλοποίηση. Ο λόγος είναι σαφής, τα IDSs επιτρέπουν σε οργανισμούς να προστατέψουν τα συστήματα τους από απειλές και προβλήματα που δημιουργούνται λόγω της αυξημένης ζήτησης για *δικτύωση*. Τέτοια συστήματα πρέπει να είναι μέρος κάθε υπολογιστικού συστήματος.

Σκεπτόμενοι λοιπόν τις σημερινές απαιτήσεις και απειλές που εμπεριέχονται σε κάθε δίκτυο, αναλύουμε παρακάτω τους λόγους εισαγωγής και χρησιμοποίησης ενός συστήματος ανίχνευσης εισβολών σε οποιονδήποτε οργανισμό ή επιχείρηση.

Λόγοι εισαγωγής και ανάπτυξης ενός IDS

1. Εντοπισμός και πρόληψη προβλημάτων

Όλα τα IDSs *προβλέπουν* και *εντοπίζουν* ένα πρόβλημα με τρόπο όπου τις περισσότερες φορές η επισήμανση και η ειδοποίηση για τις προσπάθειες εισβολής σε ένα σύστημα γίνεται σε πρώιμο στάδιο στον διαχειριστή του, με αποτέλεσμα να ληφθούν τα κατάλληλα μέτρα για την αντιμετώπιση της πριν επέλθει κάποια σημαντική ζημιά ή απώλεια.

2. Ανίχνευση επιθέσεων που δεν ανιχνεύονται από άλλα μέσα ασφάλειας

Ένα IDS μπορεί να παρακολουθεί το εσωτερικό δίκτυο για συμπεριφορές που παρεκκλίνουν από τις πολιτικές ασφάλειας δικτύων του εκάστοτε οργανισμού. Είναι σχεδιασμένο για να ελέγχει και να αντιμετωπίζει καταχρήσεις, παραβιάσεις και ζητήματα δικαιωμάτων των εσωτερικών και εξωτερικών χρηστών ενός δικτύου, όπου πολλές από τις ενέργειες τους είναι αδύνατον να εντοπιστούν με άλλα μέτρα και προγράμματα ασφαλείας.

3. Καταγραφή και τεκμηρίωση των υπαρχόντων απειλών

Τα συστήματα ανίχνευσης εισβολών μπορούν να «διακρίνουν» και να εμφανίσουν ότι ένα πληροφοριακό σύστημα αντιμετωπίζει κάποιου είδους απειλές ασφαλείας, πριν κάποια από αυτές δημιουργήσει σημαντικά προβλήματα στο σύστημα. Μία τέτοια τεκμηρίωση είναι ποικιλοτρόπως χρήσιμη, διότι:

- ✓ Εντοπίζει και προσδιορίζει σε μεγάλο βαθμό τα αντίμετρα που πρέπει να εφαρμοσθούν ενάντια σε συγκεκριμένες απειλές και βοηθά στον προσδιορισμό των μέτρων ασφαλείας που είναι πιο κατάλληλα για το σύστημα.
- ✓ Επισημαίνει στον εκάστοτε οργανισμό ή εταιρία για κατανομή πόρων στα συστήματα ασφάλειας του.
- ✓ Στα διάφορα τμήματα του συστήματος βοηθά στην αποτελεσματική κατανομή των πόρων ασφαλείας, ανάλογα με τις απειλές που το καθένα αντιμετωπίζει και την αξία του μέσα στον οργανισμό.

4. Αντιμετώπιση των προσπαθειών ελέγχου και ανίχνευσης

Το πρώτο στάδιο για μία επιτυχημένη επίθεση σε ένα υπολογιστικό σύστημα είναι να ανιχνευτεί το εκάστοτε σύστημα από κάποιον κακόβουλο χρήστη για να διαπιστωθεί η διαμόρφωση του και οι προσφερόμενες από αυτό υπηρεσίες. Πολλά από τα μέτρα ασφάλειας, όπως τα firewalls και ο έλεγχος πρόσβασης (*access control*) εστιάζονται στην αντιμετώπιση αυτού του πρώτου σταδίου επίθεσης. Όμως τα συστήματα ανίχνευσης εισβολών μπορούν να ανιχνεύσουν άμεσα τις προσπάθειες ελέγχου και ανίχνευσης και να αντιμετωπίσουν πρώιμες επιθέσεις όπως σαρώσεις δικτύων (*network scans*). Αυτές τις προσπάθειες ανίχνευσης τα IDSs μπορούν να τις μπλοκάρουν, ενημερώνοντας άμεσα τους διαχειριστές (*administrators*) για λήψη μέτρων. Σε τέτοια ενδεχόμενα αυτού του είδους ανταπόκριση και αντιμετώπιση θωρακίζει το σύστημα και αποθαρρύνει κάθε επίδοξο εισβολέα.

5. Αναγκαιότητα ύπαρξης και χρησιμοποίησης ευπαθών υπηρεσιών

Πολλοί οργανισμοί διατηρούν και έχουν από την πλευρά της ασφάλειας υπηρεσίες που είναι ευπαθείς και επισφαλείς σε κινδύνους, λόγω του ότι θεωρούνται πιο εύχρηστες και παραγωγικές από τους χρήστες τους. Για παράδειγμα, υπάρχουν υπηρεσίες που η μεταφορά των αρχείων τους γίνεται μη κρυπτογραφημένα, ωστόσο μία ασφαλέστερη αντίστοιχη υπηρεσία που εφαρμόζει ένα ασφαλές *πρωτόκολλο μεταφοράς αρχείων*, θεωρείται από πολλούς χρήστες σημαντικά πιο δύσχρηστη και άρα λιγότερο παραγωγική. Τα IDSs μπορούν να ελέγχουν αυτές τις ευπαθείς υπηρεσίες, εντοπίζοντας και ενημερώνοντας τους διαχειριστές για περιστατικά όπου αυτές δημιουργούν και προξενούν αυξημένους και επιβλαβείς κινδύνους.

6. Κάλυψη και θωράκιση των παλαιών πληροφοριακών συστημάτων

Με τη χρήση των συστημάτων ανίχνευσης εισβολών μπορούμε να προστατεύσουμε τα παλιά πληροφοριακά συστήματα. Συνήθως αυτού του είδους τα συστήματα δεν υποστηρίζονται πλέον από τους κατασκευαστές τους και η διατήρηση και η ενεργοποίηση τους είναι απαραίτητη, λόγω του ότι είναι πολύ πιο ευάλωτα και επικίνδυνα σε επιθέσεις απ' ότι τα καινούργια συστήματα.

7. Αξιολόγηση ενεργειών των διαχειριστών και των χρηστών του συστήματος

Σε πολλές περιπτώσεις οι λειτουργίες και οι μηχανισμοί ασφάλειας που παρέχονται από το εκάστοτε πληροφοριακό σύστημα είναι δυνατόν να μην χρησιμοποιούνται αποτελεσματικά και σωστά από τους διαχειριστές και τους χρήστες του, με αποτέλεσμα να δημιουργούνται σοβαρά και σημαντικά προβλήματα. Για την αντιμετώπιση του συγκεκριμένου προβλήματος, τα συστήματα ανίχνευσης εισβολών είναι ικανά να επισημαίνουν και να παρέχουν σχετικές δυνατότητες βελτίωσης.

8. Ποιοτικός έλεγχος διαχείρισης για τον σχεδιασμό ασφάλειας

Είναι σύνηθες φαινόμενο για έναν οργανισμό, η υλοποίηση του σχεδίου ασφαλείας του να παρουσιάσει κάποια στιγμή και σε ανύποπτο χρόνο επιμέρους ατέλειες. Τα συστήματα ανίχνευσης εισβολών μπορούν να ενεργήσουν ως ποιοτικός έλεγχος για τον ασφαλή σχεδιασμό και έλεγχο των δικτύων, εμφανίζοντας τυχόν ατέλειες και βοηθώντας με τον τρόπο αυτό στη διόρθωση τους, πριν κάποια από αυτές γίνει αντικείμενο εκμετάλλευσης.

9. Επιβολή και έλεγχος των πολιτικών ασφαλείας

Ο έλεγχος συνέπειας που ισχύει στα πλαίσια ενός οργανισμού μεταξύ πολιτικής ασφάλειας και κανόνων πρόσβασης πολλές φορές δεν αποδίδεται πιστά στους κανόνες πρόσβασης που έχουν θεσπιστεί από τους διαχειριστές (*administrators*) του. Τα συστήματα ελέγχου εισβολών μπορούν μέσω αρχείων καταγραφών που υπάρχουν και τηρούνται από αυτά, να εντοπίσουν τις συγκεκριμένες «ασυνέπειες» και να τις διορθώσουν άμεσα.

10. Παροχή και απόδοση πολύτιμων πληροφοριών από επιτυχείς επιθέσεις

Τα συστήματα ανίχνευσης εισβολών παρέχουν χρήσιμες πληροφορίες για εισβολές που συνέβησαν, προσφέροντας βελτιωμένη διάγνωση, επαναφορά και διόρθωση των αιτιολογικών παραγόντων. Με τον τρόπο αυτό βοηθούν και συνεισφέρουν στην αποτίμηση του μεγέθους της ζημιάς και στη διαμόρφωση για ανάκαμψη και σχεδιασμό, εφαρμόζοντας προληπτικά μέτρα για μελλοντική αποφυγή αντίστοιχων περιστατικών επίθεσης.

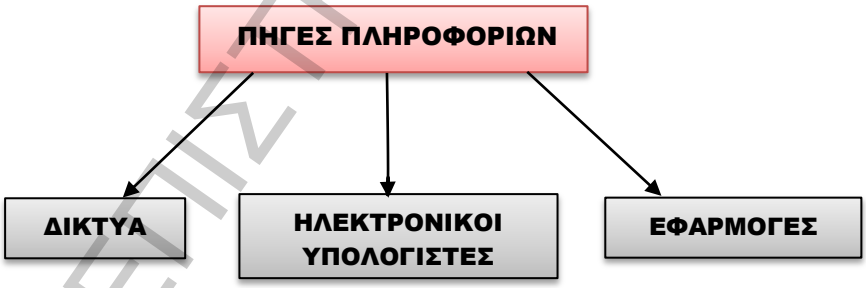
4 Πρότυπο IDS μοντέλο και Τεχνικές υλοποίησης

Για να λειτουργήσει επιτυχημένα και αποτελεσματικά ένα σύστημα ανίχνευσης εισβολών (IDS) απαιτείται να καθορισθούν οι παρακάτω γενικές παράμετροι και τεχνικές υλοποίησης του.

❖ Προέλευση πληροφοριών

Όπως αναφέραμε και παραπάνω, τα συστήματα ανίχνευσης εισβολών με ειδικά αναπτυγμένους αλγόριθμους παρακολουθούν και αναλύουν σε συνεχή βάση συμβάντα και πληροφορίες που ανταλλάσσονται και λαμβάνουν χώρα μέσα στο δίκτυο ενός πληροφοριακού συστήματος, προκειμένου να εντοπιστούν προσπάθειες εισβολής και παραβίασης της ασφάλειας του συστήματος του. Αποτέλεσμα αυτών είναι ο απαραίτητος και «αναπόφευκτος» ορισμός των συμβάντων που θα παρακολουθούνται και θα επεξεργάζονται μαζί με τα αντίστοιχα συστήματα από τα οποία θα αντλούνται και θα ανασύρονται οι πληροφορίες.

Οι τρεις συνολικά κατηγορίες πηγών των πληροφοριών



❖ Ανάλυση πληροφοριών

Ένα IDS έχοντας εντοπίσει και συλλέξει όλες τις απαραίτητες πληροφορίες από τις παραπάνω αναφερθείσες πηγές, με μια επεξεργασία ελέγχου τις αξιολογεί άμεσα για να συμπεράνει αν τα συμβάντα που καταγράφηκαν διερμηνεύουν κάποιου είδους επίθεση.

Οι εξελίξεις τα τελευταία χρόνια στον τομέα της Ανίχνευσης Εισβολών (*Intrusion Detection*) περιορίζονται σε τρεις ευρείες κατηγορίες ανάλυσης των πληροφοριών και πιο συγκεκριμένα στην:

1. Ανίχνευση ανωμαλιών (*Anomaly Detection*)
2. Ανίχνευση κατάχρησης (*Misuse Detection*)
3. Ανίχνευση βασισμένη σε προδιαγραφές (*Specification based Detection*)

❖ **Ανταπόκριση συστήματος**

Πολύ σημαντική και ακέραια παράμετρος είναι αυτή της ανταπόκρισης του συστήματος, για το πως δηλαδή θα αντιδράσει το σύστημα όταν διαπιστώσει ότι κάποια προσπάθεια εισβολής είναι σε εξέλιξη ή ότι έχει ήδη πραγματοποιηθεί η επίθεση. Στην αντίδραση του συστήματος διακρίνονται δύο κατευθύνσεις, η *ενεργός* και η *παθητική* αντίδραση. Η πρώτη θέτει - ορίζει ότι το ίδιο το IDS θα προσπαθήσει να μπλοκάρει και να ανακόψει την επίθεση, ενώ η δεύτερη ενημερώνει τους διαχειριστές του συστήματος σε περίπτωση ανίχνευσης εισβολής.

4.1 Μοντέλα και Τεχνολογίες Ανίχνευσης Εισβολών

4.1.1 Common Intrusion Detection Framework (CIDF)

Ανεξαρτήτως του μοντέλου που χρησιμοποιεί και υλοποιεί το κάθε σύστημα, υπάρχουν κάποιες γενικές προδιαγραφές με τις οποίες πρέπει να ευθυγραμμίζεται. Οι προδιαγραφές αυτές προέρχονται από ένα πλαίσιο γνωστό ως *Common Intrusion Detection Framework* (*CIDF*). Πιο συγκεκριμένα, όπως παρουσιάζεται και στο *Σχήμα 2* το εν λόγω πλαίσιο προδιαγράφει την ύπαρξη *τεσσάρων λειτουργικών κομματιών* για ένα Σύστημα Ανίχνευσης Εισβολών (IDS) τα οποία επικοινωνούν μεταξύ τους με ανταλλαγή μηνυμάτων.

1. Γεννήτριες συμβάντων (*Event Generators*)

Οι *γεννήτριες συμβάντων* είναι εξειδικευμένα κομμάτια που έχουν σαν στόχο τη συλλογή πληροφοριών μέσα από το γενικότερο πληροφοριακό περιβάλλον το οποίο πλαισιώνει ένα σύστημα ελέγχου εισβολών.

2. Αναλύτριες συμβάντων (*Event Analyzers*)

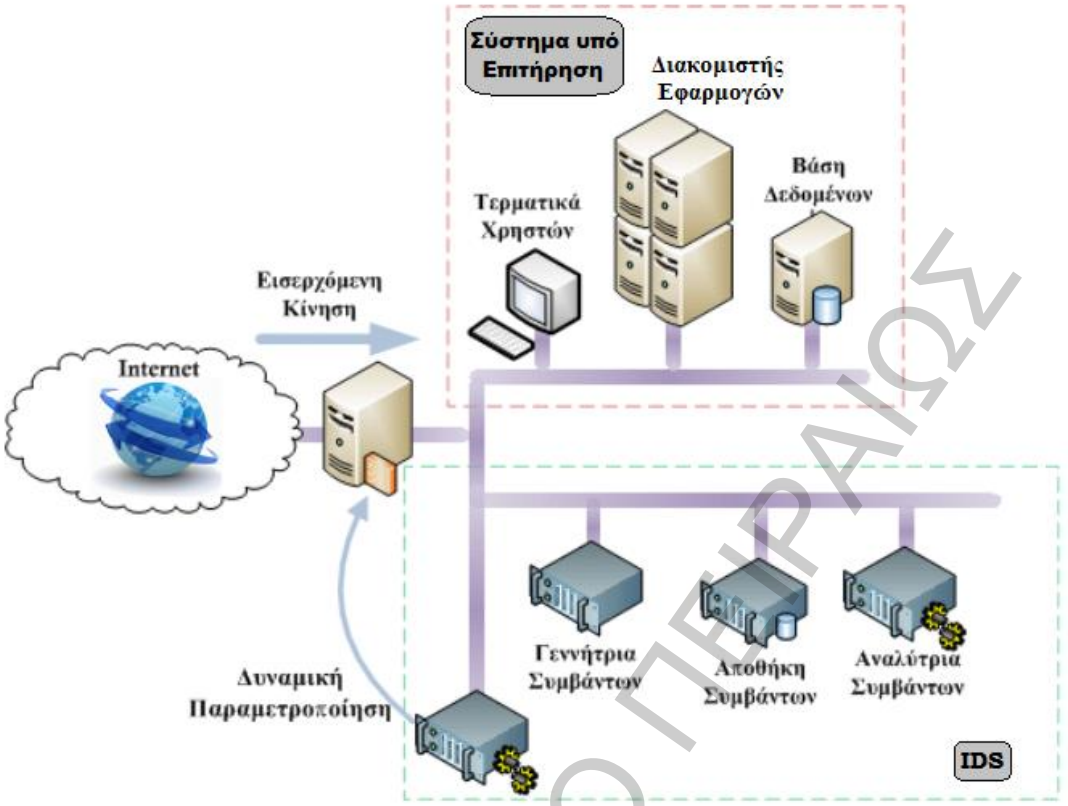
Οι *αναλύτριες συμβάντων* έχουν σαν στόχο πρώτον την ανάλυση της πληροφορίας η οποία συλλέγεται από τις γεννήτριες συμβάντων και δεύτερον τη σύνθεση ενός λογικού συμπερασμού σε σχέση με τη φύση του συμβάντος. Αποτελούνται από μονάδες που «καταφεύγουν» σε στατιστικές μεθόδους προκειμένου να καταλήξουν σε ένα αποτέλεσμα σε σχέση με ένα παρατηρηθέν συμβάν και από μονάδες οι οποίες λειτουργούν με βάση την ύπαρξη προτύπων μεταξύ των συμβάντων και των προκαθορισμένων λιστών που προέρχονται από χαρακτηριστικές υπογραφές επιθέσεων (*attack signatures*).

3. Μονάδες απόκρισης (*Response Units*)

Οι *μονάδες απόκρισης* έχουν σαν κύριο μέλημα τους την εφαρμογή των αποφάσεων που επιτάσσουν οι αναλύτριες συμβάντων (*event analyzers*), σε σχέση με το πως θα χειριστεί το σύστημα τα εισερχόμενα συμβάντα.

4. Αποθήκες συμβάντων (*Event Databases*)

Οι *αποθήκες συμβάντων*, όπως φαίνεται και από τον τίτλο τους έχουν σαν κύριο στόχο την αποθήκευση των συμβάντων.



Σχήμα 2: Γενικό μοντέλο ενός Συστήματος Ανίχνευσης Εισβολών, σύμφωνα με το πλαίσιο «*Common Intrusion Detection Framework*»

4.1.2 Μηχανισμοί Ανίχνευσης και Ανάλυσης Επιθέσεων

Όπως αναφέρθηκε και παραπάνω, στον τομέα της Ανίχνευσης Εισβολών υπάρχουν τρεις βασικοί μηχανισμοί ανάλυσης των πληροφοριών, η ανίχνευση ανωμαλιών (*anomaly detection*), η ανίχνευση κατάχρησης (*misuse detection*) και η ανίχνευση βασισμένη σε προδιαγραφές (*specification based detection*). Στις παραγράφους που ακολουθούν γίνεται μια περαιτέρω ανάλυση και εμβάθυνση των συγκεκριμένων αυτών μοντέλων.

❖ **Συστήματα Ανίχνευσης Ανωμαλιών**

Η Ανίχνευση Ανωμαλιών (*Anomaly Detection*) βασίζεται και στηρίζεται στη γενική υπόθεση ότι ένα πληροφοριακό σύστημα έχει κάποια συγκεκριμένη συμπεριφορά όταν βρίσκεται υπό φυσιολογικές συνθήκες, η οποία διαφοροποιείται όταν το σύστημα βρίσκεται σε κατάσταση επίθεσης. Με πιο απλά λόγια, η ανίχνευση ανωμαλιών επιχειρεί να εντοπίσει συμπεριφορές συστημάτων μη προβλεπόμενης

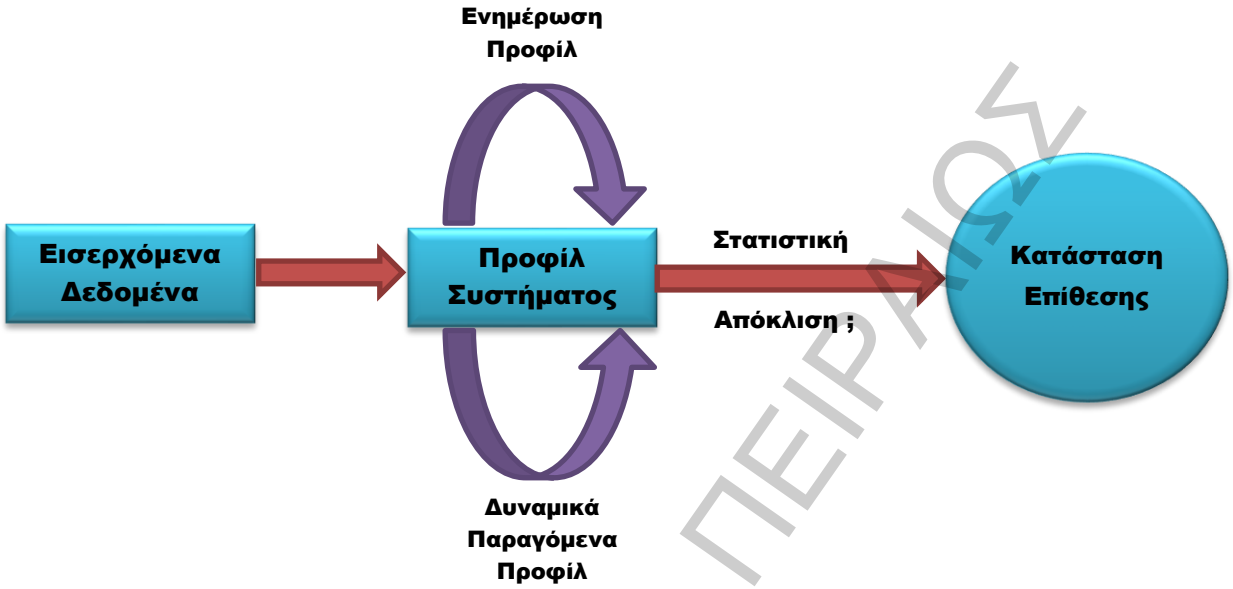
δραστηριότητας, που συνήθως αποκλίνουν από τη κανονική και φυσιολογική τους λειτουργία. Εστιάζει στην κατασκευή προτύπων σχετικά με τη χρήση του συστήματος που παρακολουθείται, συνδυάζοντας διαφορετικές μετρικές και παρατηρώντας πιθανές σημαντικές αποκλίσεις. Ο αριθμός των διεργασιών που δημιουργούνται και εκτελούνται, ο αριθμός των συνδέσεων στο εκάστοτε δίκτυο και η συχνότητα εισαγωγής, είναι μερικές μόνο από τις μετρικές που χρησιμοποιούνται. Η δημιουργία του συγκεκριμένου μοντέλου θα πρέπει να περιέχει και να συγκρίνει τις νεοσυλλεχθείσες με τις προηγούμενες μετρικές, οι οποίες βρίσκονται υπό συνθήκες κανονικής χρήσης του συστήματος. Τα κατώφλια (*threshold*) που καθορίζονται κατά τη διάρκεια της ρύθμισης ενός IDS, χρησιμοποιούνται από το σύστημα για να χαρακτηρίσουν τυχόν δραστηριότητες που περνούν τα παραπάνω κατώφλια σαν εισβολές.

Παραδείγματα μη συνηθισμένων – ανώμαλων ενεργειών

- ✓ Αυξημένος αριθμός συνδέσεων στο σύστημα
- ✓ Αυξημένος αριθμός αποτυχημένων προσπαθειών εισόδου
- ✓ Απομακρυσμένες συνδέσεις σε αδέσμευτες θύρες προορισμού
- ✓ Αυξημένη κατανάλωση υπολογιστικών πόρων του εκάστοτε συστήματος

Ένα τυπικό σύστημα ανίχνευσης ανωμαλιών απεικονίζεται στο *Σχήμα 3*. Λόγω του ότι, οι εισβολές είναι υποσύνολο μίας ανώμαλης δραστηριότητας, είναι σύνηθες φαινόμενο για τα συστήματα ανίχνευσης ανωμαλιών να χαρακτηρίζουν την ανώμαλη συμπεριφορά ως εισβολή όταν δεν είναι (*ψευδές θετικό*) και να μην ειδοποιούν στην περίπτωση μιας εισβολής (*ψευδές αρνητικό*), επειδή το σύστημα δεν θεωρεί το συγκεκριμένο γεγονός ως κάποιου είδους ανωμαλία. Επομένως, καταλαβαίνουμε άμεσα ότι τα συγκεκριμένα συστήματα έχουν μερικές ενδογενείς, λόγω του μοντέλου, δυσκολίες. Για παράδειγμα, η αποδοτικότητα του συστήματος εξαρτάται από τον αριθμό των παρακολουθούμενων μετρικών και τη συχνότητα με την οποία ανανεώνονται. Καθώς ο αριθμός των μετρικών και η συχνότητα παρακολούθησης αυξάνονται, αυξάνεται και η ακρίβεια του συστήματος. Πολύ βασική και άκρας σημασίας είναι η σχέση μεταξύ ακρίβειας μοντέλου και απόδοσης συστήματος, η οποία είναι αντιστρόφως ανάλογη, καθώς απαιτείται ένας αυξανόμενος αριθμός από

πόρους του συστήματος για να αξιοποιηθεί, προκειμένου να παραμείνει η ακρίβεια του συστήματος στα ίδια ή και μεγαλύτερα επίπεδα.



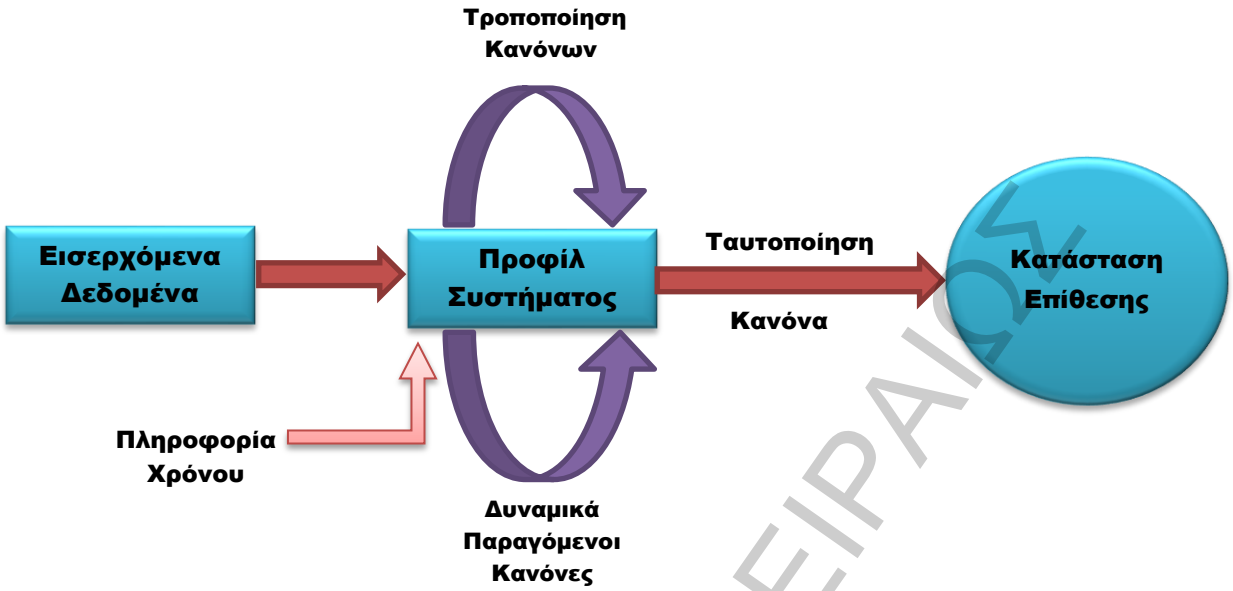
Σχήμα 3: Τυπικό Σύστημα Ανίχνευσης Ανωμαλιών

Ένα ακόμη σημαντικό πρόβλημα που δημιουργείται στα συστήματα ανίχνευσης ανωμαλιών, είναι ο καθορισμός του κατωφλιού (*threshold*) που καθορίζει το σημείο στο οποίο τελειώνει η ομαλή λειτουργία και αρχίζει η εισβολή. Παρόλα αυτά, επειδή τα προαναφερόμενα συστήματα δεν επιχειρούν να κατατάξουν συγκεκριμένες επιθέσεις με κάποιο τρόπο, υπερέχουν στο να εντοπίζουν νέες επιθέσεις χωρίς επιπλέον προγραμματισμό και να μπορούν με τον καιρό να προσαρμοστούν στις αλλαγές που εφαρμόζουν οι χρήστες τους.

❖ **Συστήματα Ανίχνευσης Κατάχρησης**

Όπως αναφέρθηκε και προηγουμένως, η δεύτερη κατηγορία συστημάτων ανίχνευσης εισβολών είναι τα Συστήματα Ανίχνευσης Κατάχρησης (*Misuse Detection Systems*), τα οποία ψάχνουν και προσπαθούν να εντοπίσουν συγκεκριμένα πρότυπα χρήσης ή ακόμη και ακολουθίες γεγονότων - συμβάντων που είναι γνωστό από προηγούμενες απόπειρες επίθεσης ότι αποτελούν και εντάσσονται σε διαδικασίες εισβολής. Εξαιτίας αυτού του χαρακτηριστικού, το συγκεκριμένο μοντέλο μπορεί να χαρακτηριστεί ως μία τεχνική βασισμένη στην εκ των προτέρων γνώση (*knowledge-based technique*) και

είναι εφαρμόσιμο μόνο όταν υπάρχει ή μπορεί να κατασκευαστεί μία υπογραφή (*signature*) που να περιγράφει την επίθεση. Επιπλέον, τα συστήματα ανίχνευσης κατάχρησης έχουν τόσο λανθασμένες θετικές (*false positive*) όσο και αρνητικές (*false negative*) προειδοποιήσεις, όπως τα προαναφερόμενα συστήματα ανίχνευσης ανωμαλιών. Η ακρίβεια ενός τέτοιου συστήματος ανίχνευσης εξαρτάται πρωτίστως από την ακρίβεια των υπογραφών (*signatures*), οι οποίες πρέπει να είναι ακριβείς και σαφείς. Συνεπώς, όταν ένας εισβολέας επιτίθεται με άγνωστα για το σύστημα πρότυπα, δηλαδή απουσία υπογραφής για τη συγκεκριμένη εισβολή, προκύπτει μια λανθασμένη αρνητική προειδοποίηση. Όταν όμως ένα συμβάν ταιριάζει με μια υπογραφή επίθεσης αλλά δεν αποτελεί απειλή, προκύπτει μια λανθασμένη θετική προειδοποίηση. Τα συστήματα ανίχνευσης κατάχρησης, ενεργοποιούν πολύ λίγες λανθασμένες θετικές προειδοποιήσεις, αλλά έχουν τη δυνατότητα να παράγουν μεγάλο αριθμό λανθασμένων αρνητικών. Παρόλα αυτά, υπάρχει μια αντιστρόφως ανάλογη σχέση μεταξύ των υπογραφών αυτών των δύο τύπων συμβάντων. Όπως όλα τα συστήματα, έτσι και τα συστήματα ανίχνευσης κατάχρησης έχουν προτερήματα και μειονεκτήματα. Παράγουν μειωμένο αριθμό λανθασμένων θετικών προειδοποιήσεων όταν υπάρχουν συγκεκριμένες υπογραφές επιθέσεων, αλλά αυτό αναπόφευκτα οδηγεί στην αύξηση των λανθασμένων αρνητικών προειδοποιήσεων που παράγει το σύστημα λόγω της αντιστρόφως ανάλογης σχέσης, που περιγράφηκε παραπάνω. Ένα σύστημα βασισμένο στο μοντέλο ανίχνευσης κατάχρησης, απεικονίζεται στο *Σχήμα 4*.



Σχήμα 4: Τυπικό Σύστημα Ανίχνευσης Κατάχρησης

Τα Συστήματα Ανίχνευσης Κατάχρησης (*Misuse Detection Systems*), έχουν τη δυνατότητα να ανιχνεύουν εγκαίρως συγκεκριμένου τύπου επιθέσεις, καθώς και τα εργαλεία που χρησιμοποιήθηκαν σ' αυτές, ώστε να θωρακιστεί πλήρως ένα σύστημα. Η συγκεκριμένη τεχνική ανίχνευσης επειδή βασίζεται σε γνωστές επιθέσεις, υπάρχουν στοιχεία για αυτές που αποθηκεύονται σε μια βάση δεδομένων (*database*) των παραπάνω συστημάτων και για να είναι πλήρως αποτελεσματική θα πρέπει απαραιτήτως να γίνεται τακτική ενημέρωση της βάσης αυτής με στοιχεία που αφορούν νέες επιθέσεις.

Εκτός όμως από την άμεση εξάρτηση τους με την ύπαρξη των υπογραφών επίθεσης, φέρουν και άλλες δύο πολύ σημαντικές αρνητικές συνέπειες - επιπτώσεις. Πρώτον, όταν οι υπογραφές στα συστήματα αυτά είναι πολύ συγκεκριμένες, δεν μπορούν να αντιμετωπίσουν διαφορετικές ποικιλίες της ίδιας επίθεσης. Δηλαδή, για κάθε επίθεση που το σύστημα χρειάζεται να είναι σε επιφυλακή, πρέπει να φέρει την απαραίτητη υπογραφή. Στην πραγματικότητα, αυτή είναι μία κοινή τακτική μεταξύ εισβολέων όταν προσπαθούν να διαπεράσουν ένα τέτοιου τύπου σύστημα ανίχνευσης εισβολών. Στη συγκεκριμένη περίπτωση το μόνο που χρειάζεται να γίνει, είναι να αλλαχθεί ελαφρώς η υπογραφή της επίθεσης, ώστε το σύστημα να «παραπλανηθεί», κάτι το

οποίο είναι εύκολο με τα σημερινά εργαλεία. Δεύτερον, υπάρχει ένα βασικό κενό ασφαλείας (*zero-day exploit*), όπου το σύστημα είναι εκτεθειμένο *από* την ώρα που μια νέα και προηγουμένως άγνωστη απειλή σημειώνεται, *έως* και την ώρα που θα δημιουργηθεί μία υπογραφή για να καλύψει τη συγκεκριμένη επίθεση.

Σύγκριση Ανίχνευσης Ανωμαλιών με Ανίχνευση Κατάχρησης

Ανίχνευση Ανωμαλιών (*Anomaly Detection*)	Ανίχνευση Κατάχρησης (*Misuse Detection*)
✓ Μπορεί να αποτελέσει πηγή πληροφοριών για ανιχνευτές υπογραφών	✓ Λιγότερο επιρρεπή σε ψευδείς ειδοποιήσεις
✓ Μπορεί να ανιχνεύσει νέες καινοτόμες τεχνικές επιθέσεων	✓ Δεν χρειάζονται εκτεταμένες περίοδοι μάθησης (*training periods*)
✗ Χρειάζονται εκτεταμένες περίοδοι μάθησης (*training periods*)	✗ Ίσως να μη μπορέσει να ανιχνεύσει παραλλαγές γνωστών επιθέσεων
✗ Επιρρεπή σε ψευδείς ειδοποιήσεις	✗ Πρέπει να ενημερώνεται με καινούργιες υπογραφές (*signatures*) νέων επιθέσεων

❖ **Συστήματα Ανίχνευσης Βάσει Προδιαγραφών**

Στην τρίτη και τελευταία κατηγορία των συστημάτων ανίχνευσης εισβολών (IDS) βρίσκεται η Ανίχνευση Βασισμένη σε Προδιαγραφές (*Specification based Detection*). Η συγκεκριμένη τεχνική ανίχνευσης έχει παρόμοια «αντιμετώπιση» με την ανίχνευση ανωμαλιών, αλλά αντί να εστιάζει στη δραστηριότητα του χρήστη, επικεντρώνεται στην αναμενόμενη συμπεριφορά του συστήματος. Όπως όλα τα προαναφερθέντα συστήματα ανίχνευσης επιθέσεων, έτσι και τα συστήματα ανίχνευσης βάσει προδιαγραφών (*Specification-based Detection*) μπορούν να παρουσιάσουν λανθασμένες αρνητικές προειδοποιήσεις. Για να επιτευχθεί πλήρως η ανίχνευση βασισμένη σε προδιαγραφές, ο αναλυτής πρέπει να καθορίσει τη συμπεριφορά του συστήματος για κάθε πιθανή κατάσταση και να κατασκευάσει ένα προφίλ για το σύστημα παρακολούθησης. Στη συνέχεια, αφού κατασκευαστεί το προφίλ, κάθε ενέργεια του συστήματος συγκρίνεται με αυτό και κάθε ανομοιότητα ή απόκλιση του χαρακτηρίζεται ως εισβολή.

❖ Υβριδική Ανίχνευση

Πολλές φορές τα Συστήματα Ανίχνευσης Εισβολών, χρησιμοποιούν ένα συνδυασμό των παραπάνω μοντέλων. Αυτός ο συνδυασμός δύο ή τριών διαφορετικών τύπων μοντέλων, καλείται υβριδικό μοντέλο. Η Υβριδική Ανίχνευση (*Hybrid Detection*), όπως ήδη αναφέρθηκε συνδυάζει τις παραπάνω προσεγγίσεις ανίχνευσης και είναι βασισμένη στην κανονική συμπεριφορά του συστήματος και στην παρεμβατική συμπεριφορά των εισβολέων. Οι προσεγγίσεις που βασίζονται στους κανόνες της υβριδικής ανίχνευσης, μπορεί να είναι σε θέση να αναγνωρίσουν συμβάντα και αλληλουχίες συμβάντων τα οποία υπό συνθήκες θα αποκαλύψουν τη διείσδυση. Στην πράξη, το υβριδικό μοντέλο μπορεί να παρουσιάζει ένα συνδυασμό των παραπάνω προσεγγίσεων - μοντέλων, έτσι ώστε να είναι αποτελεσματικό ενάντια σε μια ευρεία γκάμα επιθέσεων.

Αν και τις περισσότερες φορές ο συνδυασμός των πολλαπλών τεχνολογιών - μοντέλων ανίχνευσης εισβολών παράγει ένα πολύ ισχυρότερο σύστημα, τα υβριδικά συστήματα δεν είναι πάντοτε αποτελεσματικά. Τα διαφορετικά μοντέλα ανίχνευσης εισβολών εξετάζουν την κυκλοφορία των δικτύων και ψάχνουν για δραστηριότητα που υποδεικνύει «μη κανονική» συμπεριφορά με πολλούς και διαφορετικούς τρόπους. Η σημαντικότερη πρόκληση στην οικοδόμηση ενός υβριδικού συστήματος για αποτελεσματικότερη και αποδοτικότερη Ανίχνευση Εισβολών (*Intrusion Detection*), είναι το να «παίρνει» και να αξιοποιεί στο μέγιστο βαθμό αυτές τις παραπάνω διαφορετικές τεχνολογίες.

Ένα παράδειγμα χρήσης των παραπάνω μοντέλων ανίχνευσης εισβολών, είναι η δημιουργία του *Πίνακα 1* που δείχνει διάφορα κριτήρια τα οποία εξετάστηκαν και ελέγχθηκαν για το σύστημα ανίχνευσης εισβολών του **Ινστιτούτου Ερευνών του Stanford** (*Stanford Research Institute, SRI*).

Measure	Model	Type of Intrusion Detected
Login and Session Activity		
Login frequency by day and time	Mean and standard deviation	Intruders may be likely to log in during off-hours.
Frequency of login at different locations	Mean and standard deviation	Intruders may log in from a location that a particular user rarely or never uses.
Time since last login	Operational	Break-in on a "dead" account.
Elapsed time per session	Mean and standard deviation	Significant deviations might indicate masquerader.
Quantity of output to location	Mean and standard deviation	Excessive amounts of data transmitted to remote locations could signify leakage of sensitive data.
Session resource utilization	Mean and standard deviation	Unusual processor or I/O levels could signal an intruder.
Password failures at login	Operational	Attempted break-in by password guessing.
Failures to login from specified terminals	Operational	Attempted break-in.
Command or Program Execution Activity		
Execution frequency	Mean and standard deviation	May detect intruders, who are likely to use different commands, or a successful penetration by a legitimate user, who has gained access to privileged commands.
Program resource utilization	Mean and standard deviation	An abnormal value might suggest injection of a virus or Trojan horse, which performs side-effects that increase I/O or processor utilization.
Execution denials	Operational model	May detect penetration attempt by individual user who seeks higher privileges.
File Access Activity		
Read, write, create, delete frequency	Mean and standard deviation	Abnormalities for read and write access for individual users may signify masquerading or browsing.
Records read, written	Mean and standard deviation	Abnormality could signify an attempt to obtain sensitive data by inference and aggregation.
Failure count for read, write, create, delete	Operational	May detect users who persistently attempt to access unauthorized files.

Πίνακας 1: **Όροι και προϋποθέσεις χρησιμοποίησης στην Ανίχνευση Εισβολών (*Intrusion Detection*)**

4.2 Τεχνικές Υλοποίησης Μοντέλων

Στη συγκεκριμένη ενότητα, είναι σκόπιμο να εξετάσουμε τις διάφορες τεχνικές που μπορούν να χρησιμοποιηθούν προκειμένου να υλοποιηθούν τα μοντέλα ανίχνευσης που αναφέραμε προηγουμένως. Μία από τις πρώτες τεχνικές που «βρήκαν» άμεση εφαρμογή στην ανίχνευση εισβολών και συγκεκριμένα στην υλοποίηση του μοντέλου της ανίχνευσης ανωμαλιών (*anomaly detection*), ήταν η στατιστική ανάλυση των δεδομένων που συγκεντρώνονταν από το δίκτυο. Υπάρχουν δύο βασικοί τύποι συστημάτων που υιοθετούν τη στατιστική ανάλυση δεδομένων. Ο πρώτος εστιάζει στην ανάλυση του προφίλ ενός χρήστη, ενώ ο δεύτερος στηρίζεται στην καταγραφή και ανάλυση διαφόρων μεγεθών και τη σύγκριση των τιμών τους με κάποια προκαθορισμένη τιμή.

- Η ανάλυση του προφίλ ενός χρήστη βασίζεται στη δημιουργία προτύπων συμπεριφοράς που προκύπτουν μέσα από τη χρήση του συστήματος. Το βασικό χαρακτηριστικό αυτής της μεθόδου είναι ότι δεν απαιτείται εκ των προτέρων γνώση από ένα συγκεκριμένο χρήστη, μιας και συστήματα τέτοιου τύπου προσαρμόζονται αυτόματα στις συνήθειες του εκάστοτε χρήστη, προκειμένου να διαγνώσουν μια επικείμενη επίθεση. Παρόλο που το τελευταίο αυτό χαρακτηριστικό αποτελεί μεγάλο πλεονέκτημα για τα συγκεκριμένα συστήματα, αποτελεί ταυτόχρονα και το μεγάλο τους μειονέκτημα, μιας και ένας εκ των επιτιθέμενων μπορεί να «εκπαιδεύσει» μεθοδικά το σύστημα ώστε να αποδέχεται μια σειρά κακόβουλων ενεργειών ως φυσιολογικών.
- Συστήματα τα οποία λειτουργούν καταγράφοντας διάφορα μεγέθη, όπως τη χρήση μνήμης ή τη χρήση του επεξεργαστή (*CPU*) και συγκρίνουν τις τιμές τους με κάποιο προκαθορισμένο κατώφλι (*threshold*), επηρεάζονται άμεσα από τρεις παραμέτρους. Συγκεκριμένα, από τα μεγέθη τα οποία πρέπει να καταγραφούν, από το πόσο συχνά το σύστημα αναλύει τα καταγεγραμμένα μεγέθη και τέλος από το προκαθορισμένο κατώφλι λειτουργίας, το οποίο αποτελεί και την κύρια παράμετρο που επηρεάζει την ακριβή λειτουργία του συστήματος.

Προχωρώντας, θα πρέπει ενδεικτικά να εξετάσουμε ακόμη μερικές τεχνικές, οι οποίες αφορούν το μοντέλο ανίχνευσης κατάχρησης (*misuse detection*). Δύο από τις πιο δημοφιλείς τεχνικές είναι η σύγκριση - ταίριασμα των προτύπων (*pattern matching*) που αντιπροσωπεύουν γνωστές επιθέσεις με τα καταγεγραμμένα αρχεία χρήσης του συστήματος και η ανάλυση και η σύγκριση των βημάτων μιας επίθεσης, με τις αντίστοιχες εναλλαγές καταστάσεων που προκαλούν τα βήματα αυτά στο σύστημα (*State Transition Analysis*). Και οι δύο αυτές τεχνικές κατατάσσονται σε μια κλάση γνωστή ως τεχνικές γραφημάτων και αποδίδουν καλά στην ανίχνευση γνωστών επιθέσεων, δηλαδή επιθέσεις για τις οποίες υπάρχουν γνωστές υπογραφές. Επιπρόσθετα, όσον αφορά στη πρώτη περίπτωση, το συνολικό μέγεθος των υπογραφών αλλά και η πολυπλοκότητά τους, μπορεί να επηρεάσουν άμεσα την ικανότητα του συστήματος να λειτουργεί σε πραγματικό χρόνο (*real-time*). Ενώ για τα συστήματα τα οποία χρησιμοποιούν τη δεύτερη τεχνική, παρουσιάζεται μια περιορισμένη δυνατότητα στο να ανιχνεύουν μερικώς άγνωστες επιθέσεις, επειδή δεν στηρίζονται στη σύγκριση προτύπων από κάποιο καταγεγραμμένο αρχείο χρήσης αλλά στις εναλλαγές των καταστάσεων εξαιτίας των επιθέσεων αυτών. Το πρόβλημα στη συγκεκριμένη περίπτωση είναι πως η χρησιμότητα της τεχνικής αυτής περιορίζεται σε αυτές τις επιθέσεις, που μπορούν να περιγραφούν ως ακολουθίες εναλλαγών για το σύστημα. Συστήματα αυτού του είδους έχουν επιδείξει μια δυσκολία στο να λειτουργήσουν σε πραγματικό χρόνο, εξαιτίας του βαθμού επεξεργασίας που απαιτείται.

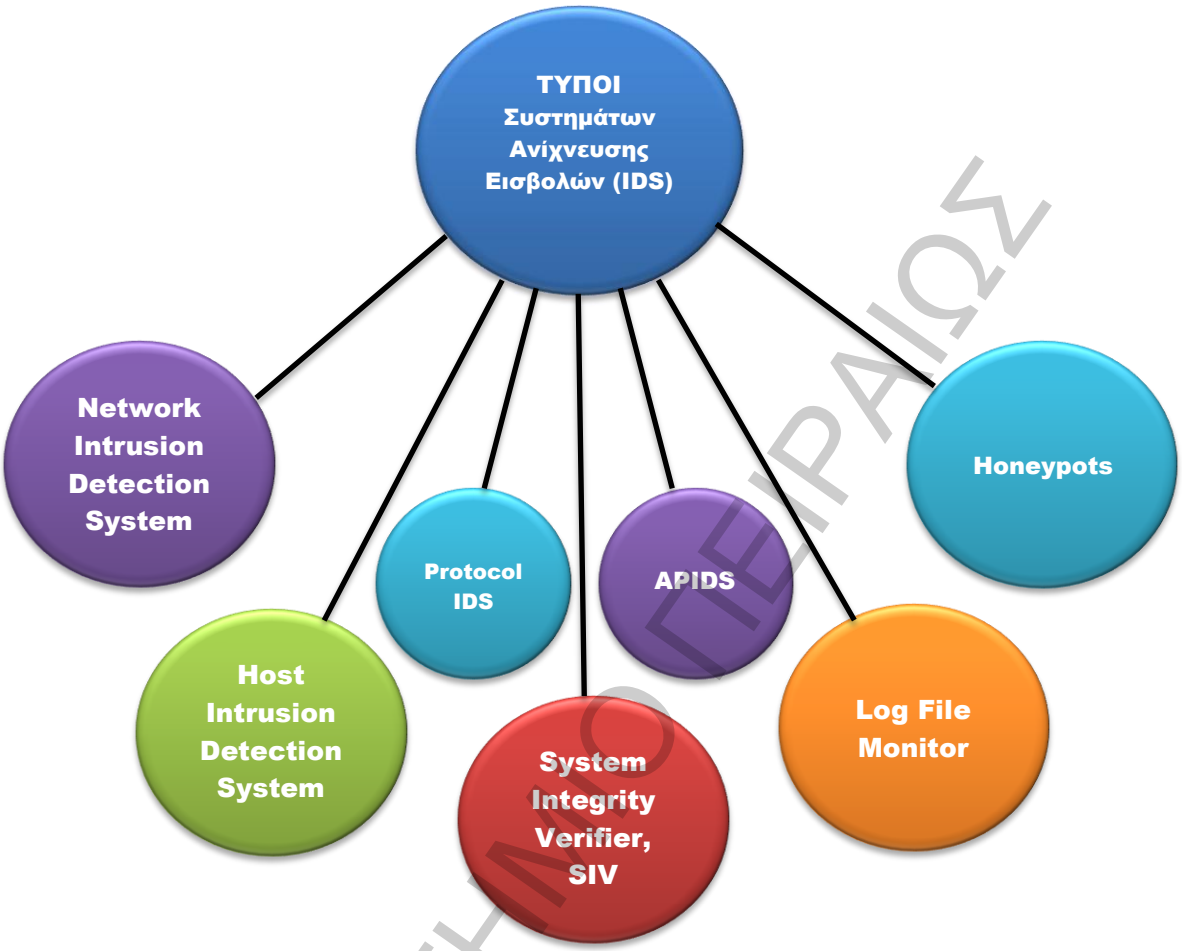
Τέλος, πρέπει να αναφέρουμε πως τεχνικές που έχουν τη βάση τους στο πεδίο της τεχνητής νοημοσύνης, όπως είναι για παράδειγμα τα νευρωνικά δίκτυα και οι γενετικοί αλγόριθμοι, έχουν από νωρίς υιοθετηθεί από το πεδίο της ανίχνευσης εισβολών. Το ζήτημα της ανίχνευσης εισβολών εναντίον πληροφοριακών συστημάτων, κάθε άλλο παρά εύκολο μπορεί να χαρακτηριστεί, μιας και όλες οι μέθοδοι και τα μοντέλα που εξετάστηκαν παραπάνω, έχουν τόσο πλεονεκτήματα όσο και μειονεκτήματα. Συνεπώς, για να επιτευχθεί η ανίχνευση εισβολών με επιτυχή τρόπο, θα πρέπει κάθε φορά να επιλέγεται η σωστή προσέγγιση σύμφωνα με το περιβάλλον που θα αναπτυχθεί ένα σύστημα ανίχνευσης εισβολών (IDS). Για παράδειγμα, σε περιβάλλοντα όπου επικρατούν επιθέσεις που προέρχονται από το

εσωτερικό ενός οργανισμού, τα συστήματα ανίχνευσης κατάχρησης (*Misuse Detection Systems*), είναι πιο κατάλληλα, καθώς σε μια τέτοια περίπτωση τα συστήματα ανίχνευσης ανωμαλιών (*Anomaly Detection Systems*) θα ήταν μη αποδοτικά, γιατί οι επιτιθέμενοι θα μπορούσαν να «εκπαιδεύσουν» τα συστήματα ανίχνευσης ανωμαλιών να θεωρούν ως φυσιολογική μια παρεισφρητική συμπεριφορά. Η αποδοτικότητα ενός IDS μπορεί να επηρεαστεί από τα πρότυπα χρήσης που οφείλονται στους προστατευμένους χρήστες του συστήματος. Με άλλα λόγια, περιβάλλοντα στα οποία οι χρήστες τείνουν να χρησιμοποιούν συγκεκριμένες εντολές και με συγκεκριμένη σειρά, χρειάζονται συστήματα ανίχνευσης ανωμαλιών, ενώ για περιβάλλοντα όπου οι χρήστες δεν έχουν σταθερές ακολουθίες χρήσης εντολών ή που κάνουν μη συχνή χρήση του προστατευμένου συστήματος, τα συστήματα ανίχνευσης κατάχρησης είναι καλύτερη και αποδοτικότερη επιλογή. Συμπερασματικά, τα Συστήματα Ανίχνευσης Εισβολών δεν είναι αλάνθαστα και για να έχουμε ένα όσο το δυνατόν πιο ασφαλές σύστημα, που θα είναι προστατευμένο από διάφορες πηγές απειλών, θα έπρεπε να ενσωματωθούν και να υιοθετηθούν περισσότερα του ενός μοντέλα προστασίας από τον κάθε ενδιαφερόμενο οργανισμό ή επιχείρηση.

5 Τύποι Συστημάτων Ανίχνευσης Εισβολών

Τα Συστήματα Ανίχνευσης Εισβολών (IDS), όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο χρησιμοποιούνται για την ανάλυση της κυκλοφορίας ενός δικτύου, αναζητώντας ενδείξεις επιθέσεων. Ωστόσο, με διαρκώς αυξανόμενο ρυθμό, τα συγκεκριμένα συστήματα χρησιμοποιούνται για την εξέταση των αρχείων καταγραφής και την ανάλυση των χαρακτηριστικών των αρχείων, στην προσπάθεια τους να εξακριβώσουν εάν τα αρχεία αυτά έχουν παραβιαστεί. Επίσης, σε πολλές περιπτώσεις τα συστήματα ανίχνευσης εισβολών χρησιμοποιούνται και σαν «δόλωμα», το λεγόμενο **honeypot**. Ένας αρκετά διαδεδομένος τρόπος ταξινόμησης των IDSs είναι με βάση την *πηγή* των πληροφοριών - δεδομένων που επεξεργάζονται. Επιπλέον, μπορούν να «χαρακτηριστούν» και από τον εκάστοτε *μηχανισμό ανάλυσης* των δεδομένων που χρησιμοποιείται για την ανίχνευση των επιθέσεων, καθώς και από το *μηχανισμό αντιμετώπισης* που ενεργοποιείται μετά από τη δημιουργία των συναγερμών (*alerts*). Υπάρχουν, λοιπόν, IDSs που χρησιμοποιούν πληροφορίες που προέρχονται από την παρακολούθηση της δικτυακής κίνησης, αναλύοντας πακέτα ώστε να βρεθεί αν ένα σύστημα γίνετε στόχος κάποιας επίθεσης, καθώς και IDSs που αναλύουν πληροφορίες που εξάγονται από εφαρμογές λογισμικού ή ακόμη και από το ίδιο το λειτουργικό σύστημα. Σε αυτό το κεφάλαιο θα επικεντρωθούμε κυρίως στα συστήματα Ανίχνευσης Εισβολών σε Δίκτυα (*NIDS*), επειδή σε συνδυασμό με ένα firewall, αποτελούν τη βασική γραμμή άμυνας ενός δικτύου.

Κατηγορίες ταξινόμησης των διαφόρων τύπων IDS



5.1 Network - Intrusion Detection Systems (NIDS)

Τα Συστήματα Ανίχνευσης Εισβολών Δικτύου, παρακολουθούν και αναλύουν την κίνηση των πακέτων που διακινούνται σε ένα δίκτυο, εξακριβώνοντας εάν κάποιος εισβολέας προσπαθεί να διεισδύσει σε κάποιον υπολογιστή του δικτύου, με σκοπό τη κατάλυση του. Η πλειονότητα των συγκεκριμένων διεισδύσεων αφορά επιθέσεις άρνησης εξυπηρέτησης, τη λεγόμενη και «δημοφιλή» πλέον επίθεση Denial-of-Service (*DoS*). Τα συστήματα ανίχνευσης επιθέσεων σε δίκτυα (*Network - Intrusion Detection Systems*), είναι συστήματα τα οποία εκτελούν μία συγκεκριμένη λειτουργία σε ένα δίκτυο, χρησιμοποιώντας δεδομένα από την κίνηση του (*network traffic analysis*), σε συνδυασμό με δεδομένα ελέγχων ορθότητας από τους hosts του εκάστοτε δικτύου. Ένα σύστημα NIDS παρέχει μεγάλη ευελιξία στη χρήση του και πολλές δυνατότητες

ελέγχου της διερχόμενης κίνησης. Αρχικά, διαβάζει τα πακέτα του δικτύου (*packet-sniffing*), στη συνέχεια φιλτράρει τα πακέτα βάσει των απαιτήσεων και των ελέγχων που ορίζει ο κάθε διαχειριστής δικτύου και σε τελική φάση ένα NIDS εμφανίζει γραφήματα και αναπαραστάσεις από τα αποτελέσματα, χρησιμοποιώντας στατιστικές μεθόδους. Μία τυπική διάταξη ενός Network-Based IDS, παρουσιάζεται στο *Σχήμα 5*.

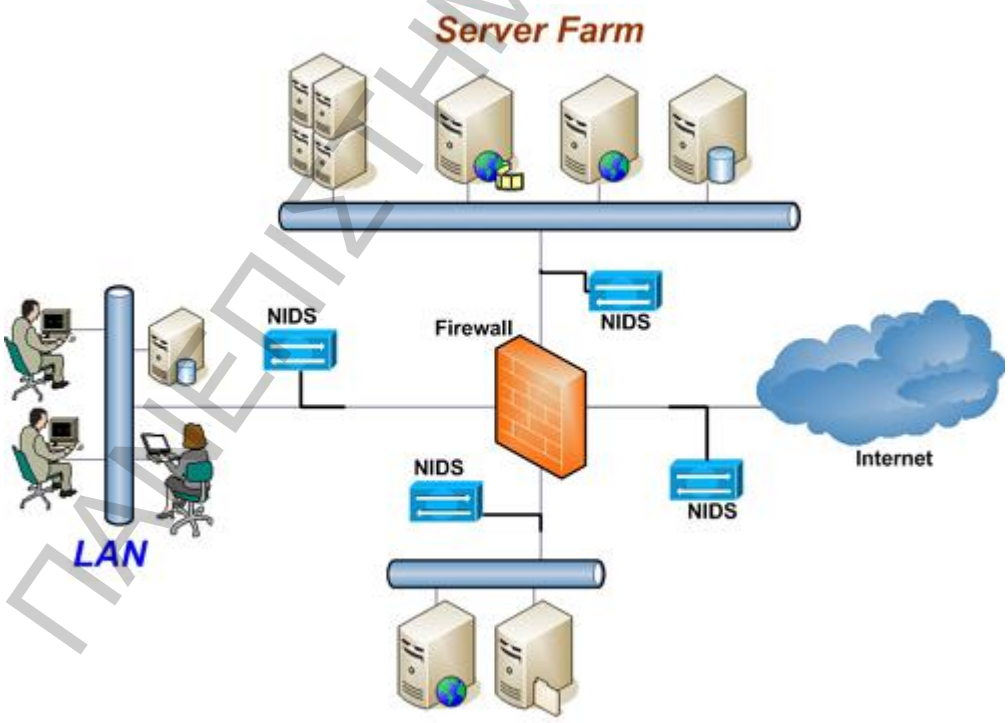
Τα συστήματα NIDS, έχουν αποθηκευμένα σε μία βάση δεδομένων (*database*) διάφορες μορφές γνωστών κακόβουλων επιθέσεων, τα οποία έχουν προκύψει από προηγούμενες επιθέσεις, με στόχο την αποτροπή των μελλοντικών. Συγκρίνοντας τα περιεχόμενα της βάσης τους, με τα περιεχόμενα των TCP/IP πακέτων που διακινούνται στο δίκτυο, προσπαθούν να προβλέψουν σε πραγματικό χρόνο (*real-time*) δικτυακές επιθέσεις που γίνονται σε διάφορα συστήματα του δικτύου. Για να έχουμε τα αναμενόμενα αποτελέσματα, η βάση δεδομένων που χρησιμοποιούν τα συστήματα αυτά είναι βασικό να αναβαθμίζεται σε καθημερινή βάση, με νέες υπογραφές επιθέσεων. Τα παραπάνω συστήματα, δεν περιορίζονται μόνο στην παρακολούθηση της εισερχόμενη κίνησης, αλλά συλλέγουν και επεξεργάζονται πολύτιμες πληροφορίες για επιθέσεις που εξελίσσονται μέσω και της εξερχόμενης κίνησης. Σκοπός τους δεν είναι να αναλάβουν κάποια δράση εάν προβλέψουν μία πιθανή επίθεση, παρά μόνο όταν ανιχνεύσουν μία «ύποπτη» λειτουργία να την αναφέρουν στον διαχειριστή του δικτύου και εν συνεχεία να διακόψουν τη συγκεκριμένη σύνδεση. Ο λόγος που δεν ενεργούν είναι γιατί κάτι τέτοιο πέραν από το γεγονός ότι συμβαίνουν λάθη που μπορεί να οδηγήσουν σε λάθος ενέργειες, υπάρχει και η περίπτωση της άρνησης των υπηρεσιών που μπορεί να οφείλεται σε λανθασμένη ενέργεια των NIDS, η οποία τις περισσότερες φορές παρακινείται και διενεργείται από κάποιον εισβολέα.

Ένα παράδειγμα συστήματος NIDS και πλέον από τα πιο διαδεδομένα Συστήματα Ανίχνευσης Εισβολών Δικτύου, είναι το **Snort**.

Τα 10 πιο διαδεδομένα NIDS

	Network-Based IDS
1	*Snort*
2	*Shadow*
3	*Dragon*
4	*PortSentry*
5	*Scanlogd*
6	*RealSecure*
7	*NFR*
8	*Bro-IDs*
9	*Prelude-IDS*
10	*NetProwler*

Η επιλογή να βρίσκεται το **Snort** στην πρώτη θέση της λίστας με τα δέκα πιο διαδεδομένα συστήματα ανίχνευσης εισβολών δικτύου έγινε, καθώς είναι ένα από τα πιο προηγμένα και δημοφιλή NIDS ανοιχτού κώδικα (*open source*) με εκατομμύρια χρήστες σε όλο τον κόσμο να το χρησιμοποιούν. Η ανάλυση της λειτουργίας του, θα μας απασχολήσει σε παρακάτω κεφάλαιο.



Σχήμα 5: Τυπική Διάταξη ενός Network - Intrusion Detection System (*NIDS*)

Τέλος, είναι συνηθισμένο για ένα NIDS να «επικοινωνεί» και με άλλα συστήματα ασφαλείας. Για παράδειγμα, όταν ένα σύστημα ανίχνευσης εισβολών δικτύου εντοπίσει μια διεύθυνση IP που την θεωρεί ύποπτη, ενημερώνει άμεσα το firewall έτσι ώστε η συγκεκριμένη διεύθυνση να εισαχθεί σε μία απαγορευμένη λίστα με IP addresses, για να μην έχει πλέον το δικαίωμα να συνδεθεί με το εκάστοτε δίκτυο.

Πολλοί ειδικοί στο χώρο της ασφάλειας δικτύων και υπολογιστικών συστημάτων, χαρακτηρίζουν τα συστήματα ανίχνευσης εισβολών σε δίκτυα ως μία λογική επέκταση των συστημάτων firewalls, η οποία έχει αρχίσει τα τελευταία χρόνια να αναπτύσσεται με ταχείς ρυθμούς. Η διαφορά ενός συστήματος ανίχνευσης εισβολών δικτύου από ένα firewall είναι ότι το NIDS, εκτός από το να ελέγχει τις επικεφαλίδες των διερχόμενων πακέτων σε πραγματικό χρόνο με βάση γνωστές υπογραφές κακόβουλων πακέτων, μπορεί να εξετάζει και να αποκωδικοποιεί τα περιεχόμενα αυτών των πακέτων ώστε να ανιχνεύσει συγκεκριμένες ανωμαλίες.

Network-Based IDS vs Firewall

Τα συστήματα ανίχνευσης εισβολών σε δίκτυα είναι σαφώς πιο περίπλοκα από τα firewalls. Ένα σύστημα ανίχνευσης εισβολών δικτύου, όπως ήδη αναφέρθηκε είναι μια ανεξάρτητη πλατφόρμα που εντοπίζει διαφόρων τύπων εισβολές, εξετάζοντας την κίνηση του δικτύου και παρατηρώντας ταυτόχρονα πολλούς δικτυακούς πόρους. Εν αντιθέσει με το firewall, ένα NIDS δεν «κάθεται» ανάμεσα στους κλάδους ενός δικτύου. Είναι σχεδιασμένο ώστε να λειτουργεί αθόρυβα μέσα σε ένα *collision domain* και σε περίπτωση εξουδετέρωσης του, δεν παρενοχλείται η ροή της κυκλοφορίας του δικτύου.

Ενώ ένα firewall λειτουργεί σαν συνοριακός φρουρός, που σημαίνει ότι όλη η κυκλοφορία πρέπει να διέλθει από αυτό για να μπορέσει να μεταφερθεί από έναν κλάδο ενός δικτύου σ' έναν άλλο. Εάν εκδηλωθεί επίθεση στο firewall και διακοπεί η παροχή των υπηρεσιών του, τότε δεν μπορεί να διακινήσει την κυκλοφορία του δικτύου. Επειδή αυτή η κατάσταση διακόπτει όλες τις επικοινωνίες, δεν επιτρέπει στον εισβολέα να εξουδετερώσει το firewall και να χρησιμοποιήσει αυτή την «ευκαιρία» για να εκκινήσει μία επίθεση εναντίον οποιουδήποτε συστήματος του εσωτερικού δικτύου.

Λειτουργία ενός τυπικού συστήματος NIDS

Ένα σύστημα NIDS «συλλαμβάνει» και καταγράφει όλη την διερχόμενη κυκλοφορία του δικτύου, όπως ακριβώς λειτουργεί και ένα εργαλείο ανάλυσης. Όπως ήδη αναφέρθηκε παραπάνω, αφού διαβαστούν όλες οι πληροφορίες στην μνήμη, το σύστημα συγκρίνει τα πακέτα με «μοτίβα» γνωστών ύποπτων επιθέσεων. Για παράδειγμα, εάν ένα NIDS παρατηρήσει ότι ένας συγκεκριμένος υπολογιστής στέλνει κατ' επανάληψη πακέτα SYN σε έναν άλλο υπολογιστή χωρίς να επιχειρήσει ποτέ να ολοκληρώσει την σύνδεση, αμέσως το NIDS θα χαρακτηρίσει τη συγκεκριμένη δραστηριότητα σαν επίθεση κατακλυσμού σημάτων SYN, γνωστή και ως **Syn-Flood Attack**, κάνοντας τις κατάλληλες αποτρεπτικές ενέργειες. Τα τελευταίας γενιάς συστήματα ανίχνευσης επιθέσεων δικτύου, έχουν στη βάση δεδομένων τους γύρω στα διακόσια και πλέον μοτίβα ύποπτων επιθέσεων. Τα αποτρεπτικά μέτρα που λαμβάνονται από τα NIDS, εξαρτώνται από το ίδιο το σύστημα και από τον τρόπο με τον οποίο θα διαμορφωθεί. Όλα τα συστήματα NIDS έχουν την δυνατότητα να καταγράφουν ύποπτα συμβάντα (*logs*), πολλά από αυτά μάλιστα μπορούν να αποθηκεύουν σε «αδιαμόρφωτη» μορφή όλη την κυκλοφορία του εκάστοτε δικτύου, έτσι ώστε στη συνέχεια να μπορεί να γίνει διεξοδική ανάλυση των συγκεκριμένων «πακέτων» από τον διαχειριστή του συστήματος. Η λειτουργία όμως, της *αναζήτησης* μέσα στα δεδομένα ενός πακέτου έχει ένα βασικό μειονέκτημα. Είναι χρονοβόρα και απαιτητική, χρησιμοποιώντας πολλούς πόρους συστήματος. Σε περίπτωση που η διερχόμενη κίνηση είναι μεγαλύτερη από αυτή που μπορεί να επεξεργαστεί ένα NIDS, επέρχεται υπερφόρτωση του συστήματος, γεγονός που συνεπάγεται καθυστερήσεις και μεγαλύτερο ποσοστό απώλειας των πακέτων λόγω έλλειψης χώρου. Ένα NIDS παρέχει ευελιξία ως προς την υλοποίηση και τις λειτουργίες του. Καθώς όμως, οι ταχύτητες μεταφοράς δεδομένων σε ένα δίκτυο αυξάνονται με ρυθμό μεγαλύτερο από ότι αυξάνεται η ταχύτητα ενός NIDS, είναι λογικό να υπάρχει το ενδεχόμενο ορισμένα «επικίνδυνα» πακέτα να καταφέρουν να διαφύγουν από τον έλεγχο. Μια λύση στο πρόβλημα της υπερφόρτωσης μπορεί να προσφέρει η *χρήση υλικού*, καθώς μπορεί να παρέχει μεγαλύτερους ρυθμούς επεξεργασίας δεδομένων.

Τέλος, υπάρχουν πολλά συστήματα NIDS που προσπαθούν να παρεμποδίσουν την «ύποπτη» επικοινωνία διακόπτοντας τελείως την σύνδεση, τα οποία μπορούν να

επικοινωνούν με ένα firewall ή ένα router για να τροποποιούν τους *κανόνες φιλτραρίσματος*, μπλοκάροντας το εκάστοτε επιτιθέμενο σύστημα.

Ένα NIDS αποτελείται τυπικά από τρία βασικά κομμάτια: **1)** τον αισθητήρα, **2)** τη κονσόλα ελέγχου και **3)** μία κεντρική μηχανή καταγραφής και αποθήκευσης των *Logs*. Ο αισθητήρας (*sensor*), είναι υπεύθυνος για τη συλλογή και την ανάλυση της κυκλοφορίας ενός δικτύου, ανιχνεύοντας αλλαγές στο περιβάλλον του δικτύου. Ενώ, η κονσόλα ελέγχου (*console*) παράγει αναφορές και λειτουργεί ως παρατηρητής των ειδοποιήσεων που προέρχονται από τον αισθητήρα. Επειδή τα NIDS καταγράφουν όλη την διερχόμενη κυκλοφορία, απαιτούνται τεράστιες ποσότητες αποθηκευτικού χώρου στις βάσεις δεδομένων τους. Είναι εξαιρετικά απαιτητικά όσον αφορά στην κατανάλωση πόρων και συνήθως οι κατασκευαστές τους συνιστούν, να «τρέχει» ο αισθητήρας (*sensor*) σε έναν υπολογιστή δεσμευμένο αποκλειστικά γι' αυτή την εργασία ενώ οι απαιτήσεις για τον υπολογιστή στον οποίο θα τρέχει η κονσόλα (*console*) είναι περίπου ίδιες με αυτές του αισθητήρα, εκτός από το γεγονός ότι θα πρέπει να «δεσμευτεί» επαρκής χώρος στον δίσκο για την αποθήκευση ενός αντιγράφου της βάσης δεδομένων του κάθε αισθητήρα.

5.2 Host - Intrusion Detection Systems (HIDS)

Μέχρι τώρα επικεντρωθήκαμε κυρίως σε συστήματα ανίχνευσης εισβολών που τρέχουν σε έναν server δεσμευμένο αποκλειστικά για αυτή την εργασία, παρακολουθώντας όλη την κυκλοφορία του δικτύου. Τα παραπάνω αναφερθέντα συστήματα χρησιμοποιούνται για τον έλεγχο της κυκλοφορίας μέσα σε ένα ολόκληρο *collision domain*. Ωστόσο, υπάρχουν IDSs τα οποία είναι ειδικά σχεδιασμένα ώστε να προστατεύουν έναν μεμονωμένο υπολογιστή. Αυτού του είδους τα IDSs παρέχουν πιο ακριβή πληροφορία για την ύπαρξη κάποιας επίθεσης και αυτό γιατί παρακολουθούν μέρος ή όλες τις καταστάσεις στις οποίες μεταβαίνει ένα υπολογιστικό σύστημα σε πραγματικό χρόνο (*real-time*).

Παρόμοια με ένα σύστημα NIDS, ένα Σύστημα Ανίχνευσης Εισβολών βάσει του Host (*Host Intrusion Detection System, HIDS*) αναλύει την κυκλοφορία του δικτύου που στέλνεται ή λαμβάνεται από μία συγκεκριμένη «μηχανή». Τα περισσότερα εμπορικά διαθέσιμα NIDS σήμερα περιλαμβάνουν λειτουργίες συστημάτων HIDS και για τον

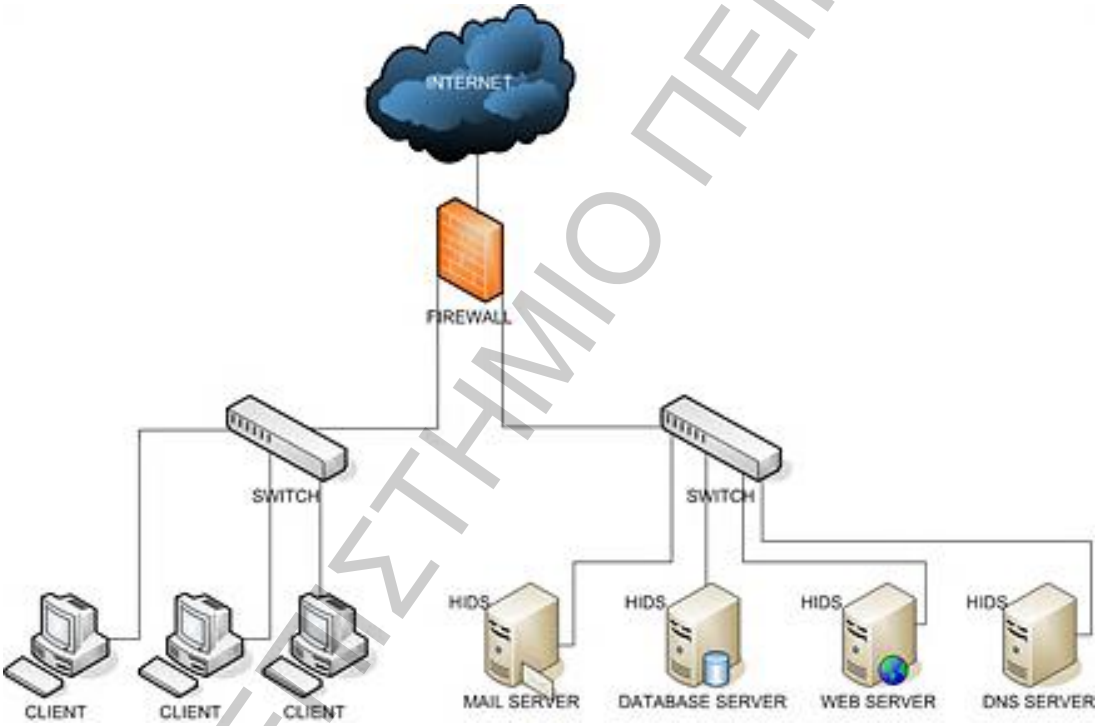
λόγο αυτό, όπως θα δούμε και παρακάτω αποκαλούνται και «υβριδικά» συστήματα ανίχνευσης εισβολών.

Ένα σύστημα HIDS είναι ένα σύστημα ανίχνευσης εισβολών, όπου για την εκάστοτε ανίχνευση χρησιμοποιούνται δεδομένα ελέγχων ορθότητας ενός και μόνου host. Αναλυτικότερα, παρακολουθεί και αναλύει τα επιμέρους συστατικά ενός υπολογιστικού συστήματος. Ο εντοπισμός μίας επίθεσης γίνεται εξετάζοντας τις κλήσεις του συστήματος, τα log files και τις αλλαγές που δημιουργούνται στα αρχεία του, αντιθέτως ένα σύστημα NIDS αναλύει όλα τα πακέτα του δικτύου που λαμβάνει. Τα HIDS, δεν παρακολουθούν δικτυακά πακέτα αλλά ανιχνεύουν ποια εφαρμογή χρησιμοποιεί ένα συγκεκριμένο πόρο του συστήματος και ποιος πόρος είναι αυτός. Για παράδειγμα, μπορούν να εντοπίσουν μια εφαρμογή που ξαφνικά προσπαθεί να προσπελάσει την βάση δεδομένων (*database*) ενός λειτουργικού συστήματος που αποθηκεύονται οι πληροφορίες και οι κωδικοί των χρηστών του. Επιπλέον, ένα σύστημα HIDS καταγράφει τα log files από τις διάφορες καταστάσεις ενός υπολογιστή και είναι σε θέση να τα συγκρίνει με τις τωρινές, με σκοπό να εντοπίσει μη εξουσιοδοτημένες αλλαγές ή ύποπτες δραστηριότητες. Παρακολουθεί τα πάντα που συμβαίνουν στο εσωτερικό ενός υπολογιστή και «αποφασίζει» με βάση την εκάστοτε πολιτική ασφαλείας που έχει καταγεγραμμένη.

Θα μπορούσε κάλλιστα, κάποιος ειδικός πάνω σε θέματα ασφάλειας δικτύου και υπολογιστικών συστημάτων να χαρακτηρίσει τη λειτουργία των συστημάτων HIDS παρόμοια με αυτή της ανίχνευσης ιών. Το λογισμικό τρέχει σαν μία διεργασία παρασκηνίου στο σύστημα για να προστατεύει, προσπαθώντας να ανιχνεύσει ύποπτη δραστηριότητα. Στις ύποπτες δραστηριότητες μπορεί να περιλαμβάνονται οι προσπάθειες εισαγωγής άγνωστων εντολών μέσω μιας αίτησης *http* ή οι απόπειρες τροποποίησης του συστήματος αρχείων. Όταν ανιχνευτεί μία ύποπτη δραστηριότητα, το HIDS προσπαθεί να τερματίσει την σύνοδο από την οποία εκκίνησε η επίθεση και στέλνει αυτόματα έναν "συναγερμό" - ειδοποίηση στον διαχειριστή του συστήματος.

Η αρχή λειτουργίας ενός συστήματος HIDS βασίζεται στο γεγονός πως ένας εισβολέας συνήθως αφήνει ψηφιακά ίχνη από τις δραστηριότητες που εκτελεί. Στην πραγματικότητα, οι περισσότεροι εισβολείς σε ένα σύστημα επιθυμούν να αποκτήσουν τον πλήρη έλεγχο έτσι ώστε να είναι σε θέση να εκτελέσουν ότι *κώδικα*

και *εφαρμογές* αυτοί επιθυμούν. Μόνο οι «άπειροι» εισβολείς θα αμελούσαν να καλύψουν τα ίχνη τους, μη εξαλείφοντας τα αρχεία καταγραφής και τις ύποπτες διεργασίες. Για τον λόγο αυτό πολλοί ειδήμονες στην ασφάλεια συνιστούν στους επόπτες συστημάτων να προωθούν όλες τις καταχωρίσεις για τα αρχεία καταγραφής σε ένα απομακρυσμένο σύστημα. Έτσι, εάν ένας εισβολέας παραβιάσει τον προστατευόμενο υπολογιστή, τουλάχιστον δεν θα μπορέσει να αλλάξει τα αρχεία καταγραφής. Η ίδια βασική αρχή θα πρέπει να ισχύει και για όλα τα Συστήματα Ανίχνευσης Εισβολών.



Σχήμα 6: Τυπική Διάταξη ενός Host - Intrusion Detection System (*HIDS*)

Σε γενικές γραμμές, ένα σύστημα HIDS χρησιμοποιεί μια βάση δεδομένων που περιέχει διάφορους κρίσιμους πόρους του συστήματος που πρέπει να παρακολουθεί (π.χ. *event logs*) και ελέγχει τις ειδικές περιοχές της μνήμης ώστε να διασφαλίζει πως δεν έχουν αλλοιωθεί. Για κάθε ένα πόρο το σύστημα καταγράφει και αναλύει όλες τις ιδιότητες που έχει ξεχωριστά και δημιουργεί συνήθως με χρήση συναρτήσεων κατακερματισμού (*hash functions*) μια ειδική τιμή, την οποία και αποθηκεύει σε μια μία ασφαλή βάση δεδομένων για την μετέπειτα επεξεργασίας της.

Ένα HIDS παρακολουθεί σε ένα σύστημα

- ✓ Την ακεραιότητα των αρχείων του
- ✓ Τις εισερχόμενες και τις εξερχόμενες συνδέσεις
- ✓ Την λειτουργία σύνδεσης (*login*) και αποσύνδεσης (*logout*) των χρηστών του
- ✓ Τις λειτουργίες των χρηστών του για μη συνηθισμένη συμπεριφορά

Για πιο λεπτομερή ανάλυση και καταγραφή των δεδομένων σχετικά με τα πακέτα και τις διευθύνσεις IP του εκάστοτε εισβολέα, ένα σύστημα HIDS χρησιμοποιεί μια εναλλακτική μέθοδο εφαρμογής που αφορά στην υλοποίηση κάποιων τυπικών στοιχείων των συστημάτων NIDS. Αυτά τα δεδομένα δεν θα μπορούσε να τα συλλέξει μόνο από το επίπεδο παρακολούθησης των εφαρμογών ενός μεμονωμένου υπολογιστή. Έτσι, με τη συγκεκριμένη υλοποίηση το σύστημα μπορεί και συγκεντρώνει περισσότερα στοιχεία - λεπτομέρειες για τον επιτιθέμενο και τον τρόπο εισβολής του.

Κατά τη διάρκεια της εγκατάστασης ενός συστήματος HIDS, ακολουθείται μια διαδικασία αρχικοποίησης των τιμών όλων των πόρων του συστήματος που θα παρακολουθούνται. Αυτό οδηγεί άμεσα στην ενημέρωση της βάσης δεδομένων (*database*) ώστε να υπάρχει πάντα η τρέχουσα κατάσταση του συστήματος. Η δημιουργία της παραπάνω βάσης είναι συνήθως μια αργή διαδικασία, καθώς όλοι οι πόροι του συστήματος πρέπει να ελεγχθούν και οι τιμές να κατακερματιστούν (*fragmentation*) και να κρυπτογραφηθούν (*encryption*).

Τα σύγχρονα υπολογιστικά συστήματα διαθέτουν πλέον αρκετούς πόρους και τα συστήματα HIDS πρέπει να τους παρακολουθούν σε συνεχή βάση, με αποτέλεσμα οι

συχνές αλλαγές στις τιμές της βάσης δεδομένων να είναι αναπόφευκτες. Για να αντιμετωπιστεί το συγκεκριμένο πρόβλημα, ένα HIDS είναι σε θέση να παρέχει και άλλες περαιτέρω τεχνικές εντοπισμού, όπως για παράδειγμα είναι η καταγραφή και η ενημέρωση των αλλαγών στις ιδιότητες ενός αρχείου. Μετά τη δημιουργία της συγκεκριμένης βάσης, το σύστημα παρακολουθεί όλους τους πόρους του εκάστοτε συστήματος και σε περίπτωση εντοπισμού κάτι ύποπτου το «αναφέρει» άμεσα στον διαχειριστή ασφαλείας μέσω των αρχείων καταγραφής συμβάντων.

Ένα σύστημα HIDS έχει κατασκευαστεί με τέτοιο τρόπο ώστε να προστατεύει επαρκώς και να μην επιτρέπει εύκολα να τροποποιηθεί η βάση δεδομένων με τις τιμές των event logs και των πόρων του. Εάν κάποιος εισβολέας καταφέρει να τροποποιήσει τα συγκεκριμένα αρχεία τότε το σύστημα δεν θα είναι σε θέση να εντοπίσει την επίθεση, αφού δεν θα καταλάβει την αλλοίωση της κατάστασής του σε σχέση με την τιμή της βάσης δεδομένων του και ο εισβολέας θα αποκτήσει τον πλήρη έλεγχο του συστήματος. Για να αποφευχθεί το παραπάνω σενάριο θα πρέπει ο εκάστοτε διαχειριστής (*administrator*) ασφαλείας να λαμβάνει επιπρόσθετα μέτρα ώστε να εξασφαλίζει την ακεραιότητα του HIDS συστήματος. Ως μέτρο προστασίας της βάσης δεδομένων του HIDS συνήθως επιλέγεται η κρυπτογράφηση της βάσης αλλά και η λήψη αντιγράφου ασφαλείας ώστε να υπάρχει ένα ιστορικό των αλλαγών της βάσης.
Τέλος, υπάρχει η δυνατότητα τα event logs να αποστέλλονται σε ειδικές «συσκευές» εκτός δικτύου μέσω ειδικών συνδέσεων που επιτρέπουν την μονόδρομη και όχι την αμφίδρομη επικοινωνία. Με τον τρόπο αυτό τα αρχεία θα αποθηκεύονται κάπου που ο διαχειριστής μπορεί να τα επεξεργαστεί τοπικά και δεν θα υπάρχει δυνατότητα ο εκάστοτε εισβολέας να αποκτήσει απομακρυσμένα πρόσβαση σε αυτά.

Αδυναμίες - Ευπάθειες Συστημάτων HIDS

Τα συστήματα ανίχνευσης εισβολών που χρησιμοποιούνται για την προστασία υπολογιστών έχουν αρκετές αδυναμίες, οι οποίες καθιστούν αυτή την λύση μη πρακτική για πολλά περιβάλλοντα. Κατ' αρχήν, τα περισσότερα HIDS μπορούν να παρακολουθούν μόνο συγκεκριμένους τύπους υπολογιστών. Για παράδειγμα, μπορεί να προστατεύουν μόνο *web servers*. Εάν στον εκάστοτε server τρέχουν πολλαπλές υπηρεσίες, ένα HIDS μπορεί να μην καταφέρει να ανιχνεύσει μία εισβολή. Αν και τα περισσότερα συστήματα HIDS παρακολουθούν τις βασικότερες λειτουργίες ενός server, όπως είναι οι αλλαγές στα δικαιώματα πρόσβασης των χρηστών, ένας εισβολέας μπορεί να βρει κάποιο τρόπο για να απενεργοποιήσει ένα HIDS πριν επιχειρήσει να κάνει οποιεσδήποτε αλλαγές στο προστατευόμενο σύστημα. Εάν το HIDS απενεργοποιηθεί, ο κάθε εισβολέας είναι ελεύθερος να κάνει οτιδήποτε θέλει στο σύστημα.

Ένα άλλο πρόβλημα σχετίζεται με το γεγονός ότι τα συστήματα HIDS τρέχουν απλώς σαν μία διεργασία παρασκηνίου και δεν έχουν πρόσβαση στις «ζωτικές» λειτουργίες επικοινωνίας του προστατευόμενου συστήματος. Αυτό σημαίνει ότι το HIDS δεν μπορεί να μπλοκάρει τις επιθέσεις που έχουν σαν στόχο τον ίδιο τον σωρό πρωτοκόλλων. Για παράδειγμα, σε μία επίθεση κατακλυσμού σταγονιδίων (*teardrop attack*) χρειάζονται 20 ή περισσότερα προβληματικά πακέτα για να καταρρεύσει ένας server με τα Windows στον οποίο δεν έχουν εγκατασταθεί οι πρόσφατες ενημερώσεις - διορθώσεις ασφαλείας. Αν και ο χρόνος αυτός είναι αρκετός για να υπάρξει αντίδραση και να ληφθούν τα κατάλληλα αντίμετρα, ένα HIDS θα ήταν εντελώς «αβοήθητο» σε αυτή την περίπτωση, επειδή δεν θα έβλεπε ποτέ τη συγκεκριμένη κυκλοφορία.

Συνοψίζοντας, θα μπορούσαμε να ισχυριστούμε ότι είναι κατά κάποιο τρόπο αντιφατικό να «τρέχει» ένα λογισμικό ανίχνευσης εισβολών στο ίδιο το σύστημα που θέλουμε να προστατέψουμε. Εάν ένας εισβολέας καταφέρει να διεισδύσει στον συγκεκριμένο υπολογιστή, μπορεί κάλλιστα να παραβιάσει και το ίδιο το IDS. Μία τέτοια παραβίαση θα σήμαινε ότι ο επιτιθέμενος κατάφερε να καταρρίψει και την τελευταία γραμμή άμυνας μας.

Αποτελεσματικότητα - Αποδοτικότητα Συστημάτων HIDS

Εκτός από τις ήδη αναφερθείσες ευπάθειες που αντιμετωπίζουν τα συστήματα ανίχνευσης εισβολών με βάση την προστασία μεμονωμένων υπολογιστών, σε πολλές των περιπτώσεων είναι πλήρως αποτελεσματικά. Σαν παράδειγμα, υποθέτουμε ότι ο web server που θέλουμε να προστατέψουμε βρίσκεται στην αποστρατικοποιημένη ζώνη (*demilitarized zone, DMZ*) του δικτύου μας. Η αποστρατικοποιημένη ζώνη «βρίσκεται» μεν πίσω από το firewall, αλλά είναι ένας απομονωμένος τομέας του δικτύου ο οποίος περιλαμβάνει μόνο τον web server. Το firewall είναι διαμορφωμένο ώστε να επιτρέπει μόνο την διέλευση της κυκλοφορίας του HTTP έως τον web server. Σε αυτή την διαμόρφωση, ένα σύστημα HIDS είναι αρκετό για την προστασία του web server επειδή το μεγαλύτερο μέρος της προστασίας παρέχεται από το firewall. Το firewall είναι αυτό που πρέπει να διασφαλίσει ότι η μοναδική κυκλοφορία που θα φτάνει στον web server είναι οι αιτήσεις HTTP. Αυτό σημαίνει άμεσα, ότι δεν χρειάζεται καμία περαιτέρω ανησυχία για τυχόν επιθέσεις σε άλλες υπηρεσίες που τρέχουν στον web server.

Στην παραπάνω περίπτωση, το μόνο που χρειάζεται να διασφαλίσει ένα HIDS είναι ότι δεν περιλαμβάνονται ύποπτες αιτήσεις για την προσπέλαση αρχείων, scripts ή εφαρμογές Java στις αιτήσεις HTTP. Αν και αυτή δεν είναι καθόλου μικρή ευθύνη, περιορίζει σημαντικά τα είδη των επιθέσεων που καλείται να αντιμετωπίσει το HIDS.
Τα συστήματα HIDS, μπορούν επίσης να αποδειχθούν εξαιρετικά χρήσιμα σε περιβάλλοντα δικτύων στα οποία κάθε σύστημα συνδέεται σ' ένα switch, δηλαδή αποτελεί έναν ξεχωριστό κλάδο στο δίκτυο. Μία τέτοια διαμόρφωση παρουσιάζεται στο *Σχήμα 6*. Όπως διαφαίνεται, ο κάθε υπολογιστής του δικτύου συνδέεται απευθείας σε ένα switch. Ουσιαστικά, αυτή η διαμόρφωση σημαίνει ότι κάθε σταθμός του δικτύου έχει το δικό του collision domain, το switch απομονώνει όλη την κυκλοφορία που απευθύνεται σε έναν συγκεκριμένο σταθμό, έτσι ώστε να την βλέπουν μόνο τα εμπλεκόμενα συστήματα.

Επειδή το switch απομονώνει τις συνόδους επικοινωνίας μεταξύ των σταθμών του δικτύου, ένα βασιζόμενο στο δίκτυο IDS δεν μπορεί να παρακολουθεί όλη την διερχόμενη κυκλοφορία του δικτύου. Στην περίπτωση, που ένας «σταθμός» εκκινήσει μία επίθεση εναντίον του web server που τρέχει σε έναν *Intranet Server* του δικτύου,

τότε το NIDS δεν θα αντιληφθεί την επίθεση και κατά συνέπεια δεν θα λάβει τα κατάλληλα αντίμετρα. Αυτό σημαίνει, ότι η επίθεση δεν θα εμφανίζεται καθόλου στα αρχεία καταγραφής του NIDS και η εκδήλωση της θα περάσει απόλυτα απαρατήρητη. Ένα σύστημα HIDS είναι σε πολύ καλύτερη θέση να προστατέψει τον web server αυτού του δικτύου, καθώς τρέχει στο σύστημα το οποίο προστατεύει και δεν επηρεάζεται από την απομόνωση της κυκλοφορίας που οφείλεται στο switch. Με τον τρόπο αυτό, θα παρακολουθεί όλη την κυκλοφορία που βλέπει ο web server και κατά συνέπεια θα μπορεί να προστατέψει το σύστημα από βασιζόμενες στο HTTP επιθέσεις.

Κλείνοντας, θα ήταν καλό να τονίσουμε πως οι περισσότεροι κατασκευαστές προϊόντων switch δίνουν την δυνατότητα να διαμορφωθεί μία από τις θύρες του switch σαν θύρα παρακολούθησης (*monitoring port*). Με τον τρόπο αυτό το εκάστοτε switch θα μπορεί να στέλνει ένα αντίγραφο όλης της διερχόμενης κυκλοφορίας σε οποιοδήποτε σύστημα συνδέεται σε αυτή την θύρα. Εάν χρησιμοποιήσουμε ένα σύστημα NIDS σ' ένα περιβάλλον με switches, τότε πρέπει να συνδεθεί στην συγκεκριμένη θύρα παρακολούθησης για να διασφαλιστεί ότι το IDS θα μπορεί να ελέγχει όλη την διερχόμενη κυκλοφορία.

5.3 NIDS VS HIDS

Όπως ήδη αναφέρθηκε σε προηγούμενη ενότητα, δεν υπάρχει κάποια «οριστική» απάντηση για το αν ένα σύστημα NIDS είναι αποτελεσματικότερο από κάποιο HIDS ή αντίστροφα. Η χρησιμοποίηση των δύο αυτών τύπων IDSs θα πρέπει να γίνεται αν όχι πάντα, τις περισσότερες φορές συγχρόνως. Αναλυτικότερα, ένα σύστημα ανίχνευσης εισβολών δικτύου θα πρέπει να περιλαμβάνεται - χρησιμοποιείται σε ένα δίκτυο και αντίστοιχα ένα HIDS στους εκάστοτε servers/workstations.

Όπως όλα τα Συστήματα Ανίχνευσης Εισβολών (IDS), έτσι και τα συστήματα NIDS και HIDS χαρακτηρίζονται από κάποια *πλεονεκτήματα* στον τρόπο ανίχνευσης επιθέσεων και από ορισμένες *αδυναμίες* τα οποία τα καθιστούν μη αποτελεσματικά στην λειτουργία τους.

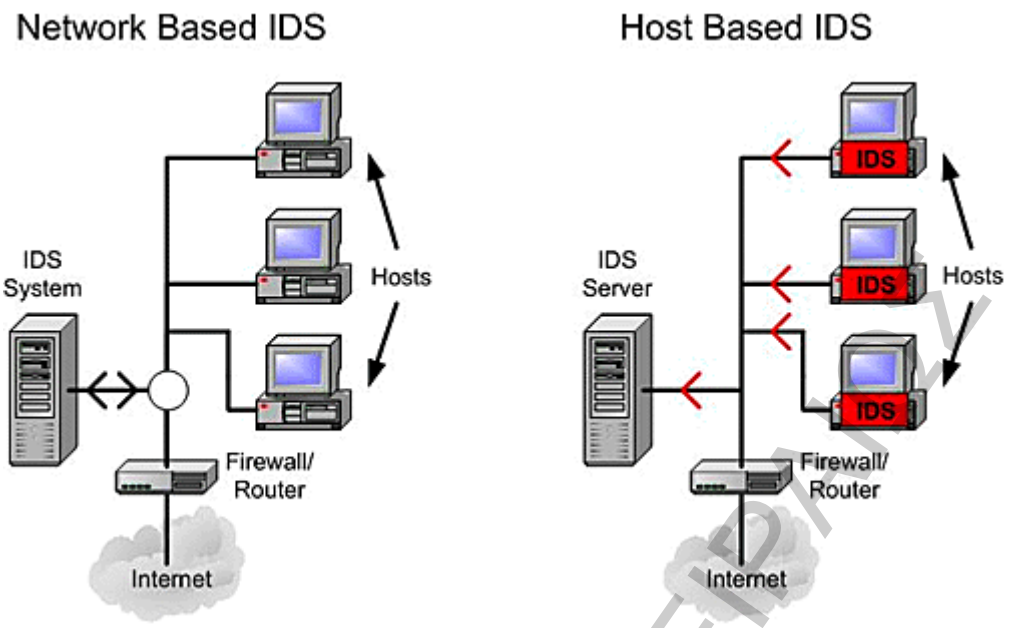
Για την καλύτερη κατανόηση των δύο αυτών τύπων Συστημάτων Ανίχνευσης Εισβολών, δημιουργήθηκαν και παρουσιάζονται αναλυτικά δύο συγκριτικοί πίνακες μεταξύ των Συστημάτων Ανίχνευσης Εισβολών Δικτύου και αυτών με βάση τον Host.

Network - Intrusion Detection Systems (NIDS)

Πλεονεκτήματα	Μειονεκτήματα
✓ Ικανά να παρακολουθούν πολλαπλά λειτουργικά συστήματα	✘ Παύει να λειτουργεί εάν ο όγκος των πακέτων υπερβεί τις «δυνατότητες» του
✓ Δεν έχουν επίπτωση στην ταχύτητα του εκάστοτε δικτύου	✘ Αντιμετώπιση δυσκολίας στο χειρισμό επιθέσεων κρυπτογράφησης
✓ Δεν μπορούν να ανιχνευθούν εύκολα από εισβολείς (*attackers*)	✘ Switched δίκτυα
✓ Ικανά να παρακολουθούν την κίνηση σε πολλούς υπολογιστές	✘ Δυσκολία στην ικανότητα ανίχνευσης πολλαπλών τύπων επιθέσεων

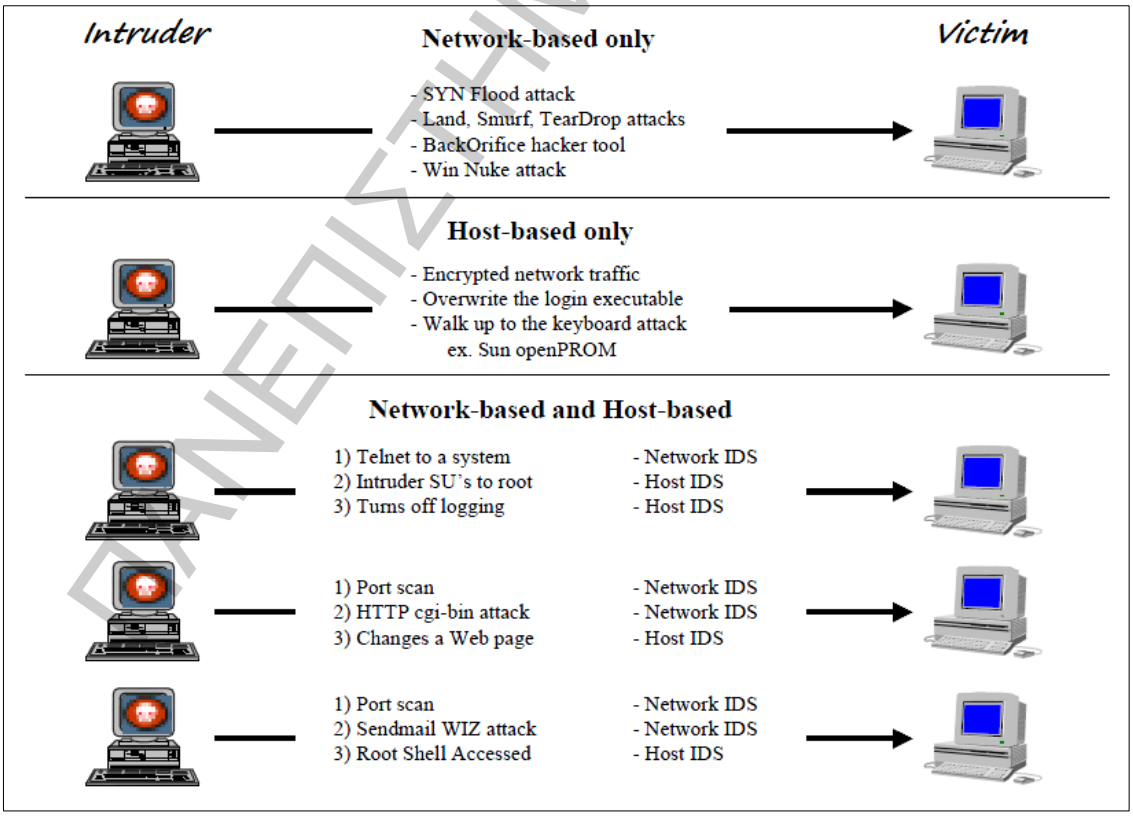
Host - Intrusion Detection Systems (HIDS)

Πλεονεκτήματα	Μειονεκτήματα
✓ Ικανά να προσδιορίσουν αν η επίθεση είναι επιτυχής	✘ Η διαχείριση πολλαπλών συστημάτων είναι πιο δύσκολη
✓ Δεν επηρεάζονται από την κρυπτογράφηση	✘ Καταλαμβάνουν αρκετούς πόρους του υπολογιστή
✓ Δεν επηρεάζονται από switches	✘ Δεν έχουν ορατότητα της κίνησης όλου του δικτύου
✓ Ακριβής ανίχνευση λόγω της πληθώρας δεδομένων που μπορούν να συλλέξουν	✘ Είναι τα ίδια ευάλωτα σε επιθέσεις



Εικόνα 3: **Network-Based IDS vs Host-Based IDS**

Ο ακόλουθος πίνακας, παρουσιάζει ορισμένα «σενάρια» επίθεσης σε ένα ανεξάρτητο σύστημα NIDS και HIDS αντίστοιχα, καθώς και τα οφέλη από την εφαρμογή και των δύο σε ένα δίκτυο συγχρόνως.



Πίνακας 2: **Σύγκριση και εφαρμογή των συστημάτων NIDS και HIDS**

5.3.1 ΣΥΜΠΕΡΑΣΜΑΤΑ

Σε κάθε περίπτωση, θα ήταν άτοπο να μπούμε στο δίλημμα να επιλέξουμε μεταξύ των δύο παραπάνω αναφερθέντων Συστημάτων Ανίχνευσης Εισβολών. Ο λόγος είναι σαφής, οι συγκεκριμένοι δύο τύποι συστημάτων ανίχνευσης εισβολών διαφέρουν σημαντικά μεταξύ τους αλλά συμπληρώνουν ο ένας τον άλλο, με το κάθε σύστημα να έχει τα δικά του δυνατά και αδύνατα σημεία.

Κατά συνέπεια, μια ολοκληρωμένη λύση για ένα δίκτυο αποτελεί η ορθή χρήση και εφαρμογή των δύο αυτών συστημάτων, ώστε να έχουμε σαν αποτέλεσμα την καλύτερη δυνατή επίβλεψη - προστασία του. Θα ήταν επωφελές για ένα IDS να ενσωματώσει πλήρως τα παραπάνω συστήματα για να ολοκληρώσει την άμυνα του. Ένα πραγματικά ασφαλές περιβάλλον για να παρέχει ένα σύστημα ισχυρό και με βάσεις που θα έχουν σαν σύνολο τη *παρακολούθηση*, την *ανταπόκριση* και την *ανίχνευση* επιθέσεων, απαιτείται τόσο ένα σύστημα ανίχνευσης εισβολών δικτύου όσο και ένα σύστημα ανίχνευσης εισβολών που χρησιμοποιούνται για την προστασία μεμονωμένων υπολογιστών.

Ως συμπέρασμα, θα πρέπει να γίνει κατανοητό ότι κανένα σύστημα από μόνο του δεν είναι ικανό να παρέχει επαρκή προστασία σε ένα υπολογιστικό περιβάλλον, αλλά θα πρέπει να υποστηρίζεται από έμπειρους διαχειριστές - επόπτες και σαφή πολιτική ασφαλείας που θα καλύπτει λεπτομερώς όλα τα πιθανά και «απίθανα» σενάρια που μπορούν να συμβούν.

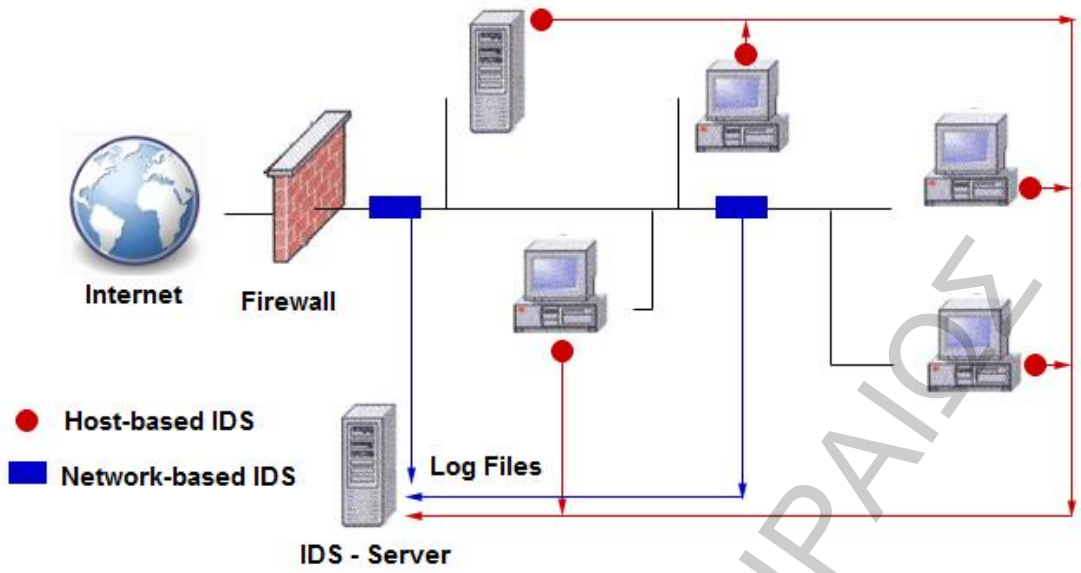
5.4 Hybrid IDS

Η συγκεκριμένη κατηγορία των Συστημάτων Ανίχνευσης Εισβολών (IDS) είναι κατά κάποιο τρόπο υβριδική των δύο προηγούμενων, εξετάζοντας τη συμπεριφορά εφαρμογών και αναλύοντας τα application log files. Τα υβριδικά IDS, χρησιμοποιούν συνδυασμό από τις παραπάνω «πηγές» για την επίτευξη και την εξασφάλιση καλύτερων αποτελεσμάτων με την υψηλότερη κάλυψη σε ότι αφορά την ανίχνευση εισβολών. Λειτουργούν συνήθως παρόμοια με τα NIDS παρακολουθώντας τα διακινούμενα πακέτα και συγκρίνοντας τα με τις καταχωρημένες υπογραφές επιθέσεων. Η διαφορά τους είναι, πως η παραπάνω διαδικασία χρησιμοποιείται μόνο για τα πακέτα που υπάρχουν σε έναν συγκεκριμένο κόμβο του δικτύου στον οποίο «τρέχει» το υβριδικό σύστημα ανίχνευσης εισβολών.

Ένα υβριδικό IDS αποτελείται από τρεις βασικές συνιστώσες

- Κεντρικό Σύστημα Διαχείρισης
- Network-Based IDS (*NIDS*)
- Host-Based IDS (*HIDS*)

Συμπερασματικά, τα υβριδικά IDS (*Hybrid IDS*) απαιτούν πολύ λιγότερους πόρους σε σχέση με τα NIDS και HIDS διορθώνοντας ή καταργώντας πολλά από τα προβλήματα τους. Ένα βασικό παράδειγμα είναι η σωστή λειτουργία σε συστήματα με κρυπτογραφημένες επικοινωνίες και σε switched δίκτυα, όπου υπάρχει μεγάλη δικτυακή κίνηση πακέτων. Το παραπάνω πρόβλημα γινόταν εντονότερο με τη χρησιμοποίηση των εικονικών ιδιωτικών δικτύων, τα λεγόμενα VPNs (*virtual private networks*).



Σχήμα 7: **Hybrid IDS**

5.5 Log File Monitor - LFM

Τα εργαλεία παρακολούθησης των αρχείων καταγραφής (*Log File Monitor, LFM*), «διαβάζουν» τα αρχεία καταγραφής συμβάντων που παράγονται από τις υπηρεσίες του δικτύου, αναζητώντας για νέες μορφές – μοτίβα επιθέσεων. Τα αρχεία καταγραφής συμβάντων (*event logs*) είναι μία πολύτιμη πηγή πληροφοριών, αλλά λόγω του μεγάλου όγκου δεδομένων που περιέχουν μόνο μία αυτοματοποιημένη προσέγγιση στο θέμα του ελέγχου τους μπορεί να είναι επιτυχής. Πριν γίνει η επιλογή για το κατάλληλο λογισμικό παρακολούθησης δικτύου, όταν δημιουργείται ένα πλάνο για τον έλεγχο των αρχείων καταγραφής συμβάντων είναι απαραίτητο να ακολουθηθούν τα τρία παρακάτω βήματα.

1. Καθορισμός πληροφοριών λήψης του δικτύου.
2. Εντοπισμός των αρχείων καταγραφής συμβάντων που περιέχουν αυτές τις πληροφορίες.
3. Καθορισμός καταχωρίσεων των αρχείων καταγραφής που θα προκαλούν την ενεργοποίηση ενός συναγερμού (alert).

Τα εργαλεία παρακολούθησης των αρχείων καταγραφής συμβάντων (*LFM*), επιχειρούν να ανιχνεύσουν εισβολές αναλύοντας τα αρχεία καταγραφής συμβάντων του συστήματος. Παρακολουθούν και καταγράφουν τα log files που δημιουργούνται από τις υπηρεσίες δικτύου, με τρόπο παρόμοιο όπως τα NIDS και ψάχνουν για πρότυπα (*patterns*) στα αρχεία καταγραφής που «δείχνουν» ότι κάποιος εισβολέας επιτίθεται. Για παράδειγμα, ένα βασικό LFM μπορεί να κάνει αναζήτηση (*search*) για ένα **Apache access.log** αρχείο, με το χαρακτηριστικό αίτημα / cgi-bin /. Παρακάτω φαίνεται ένα αρχείο καταγραφής σάρωσης CGI σε έναν web server.

Log Generated - /cgi-bin/

```
==========================================================
127.0.0.1 - - [14/Jun/2012:09:48:14 -0400] [ccc] "HEAD /cgi-bin/test-cgi HTTP/1.0" 404 0
127.0.0.1 - - [14/Jun/2012:09:48:14 -0400] [ccc] "HEAD /cgi-bin/nph-test-cgi HTTP/1.0" 404 0
127.0.0.1 - - [14/Jun/2012:09:48:14 -0400] [ccc] "HEAD /cgi-bin/phf HTTP/1.0" 404 0
127.0.0.1 - - [14/Jun/2012:09:48:14 -0400] [ccc] "HEAD /cgi-bin/phf.pp HTTP/1.0" 404 0
127.0.0.1 - - [14/Jun/2012:09:48:14 -0400] [ccc] "HEAD /cgi-bin/phf.cgi HTTP/1.0" 404 0
127.0.0.1 - - [14/Jun/2012:09:48:14 -0400] [ccc] "HEAD /cgi-bin/websendmail HTTP/1.0" 404 0

==========================================================
```

Ένα δύσκολο ερώτημα που «απασχολεί» και αφορά τα εργαλεία παρακολούθησης των αρχείων καταγραφής, είναι ποιά συμβάντα θα ενεργοποιούν τους συναγερμούς (*alerts*). Συνήθως οι διαχειριστές των LFM, έχουν σαν *σημείο εκκίνησης* τον εντοπισμό οποιασδήποτε προσπάθειας αναμετάδοσης, συγκρίνοντας τη διεύθυνση IP του υπολογιστή που επιχειρεί την αναμετάδοση έναντι μιας λίστας διευθύνσεων IP από έγκυρα συστήματα αναμετάδοσης. Επιπρόσθετα, μπορούν να αναζητούν τη συγκεκριμένη εντολή αναμετάδοσης στα αρχεία καταγραφής. Ένας βασικός τρόπος για να γίνεται η επαλήθευση της ορθής λειτουργίας ενός Log File Monitor που ήδη χρησιμοποιείται, είναι η δημιουργία ενός περιστατικού παρόμοιο με αυτά που είναι άξια εντοπισμού. Με την συγκεκριμένη τακτική μπορούμε να διασφαλίσουμε ότι το LFM που χρησιμοποιείται θα ανταποκριθεί με τον αναμενόμενο τρόπο εάν συμβούν πραγματικά παρόμοια περιστατικά.

Εργαλεία παρακολούθησης των αρχείων καταγραφής συμβάντων

- Το **Logcheck**, είναι ένα τυπικό LFM λογισμικό, που βοηθά και επιλύει τα προβλήματα επιτόπου από παραβιάσεις ασφαλείας στα αρχεία καταγραφής, στέλνοντας αυτομάτως τα αποτελέσματα μέσω e-mail στον διαχειριστή του εκάστοτε log file monitor. Πολλά HIDS περιλαμβάνουν ήδη δικά τους log-checking λογισμικά, όμοια με το Logcheck.

- Το **Logsurfer**, είναι ένα εργαλείο καταγραφής ελέγχου παρόμοιο με το Logcheck, αλλά με την πρόσθετη δυνατότητα να χειρίζεται επιπρόσθετα πολλαπλά μηνύματα και δυναμικούς κανόνες (*rule set*). Το Logsurfer είναι ένα πολύ γρήγορο, ευέλικτο και τεκμηριωμένο εργαλείο LFM γραμμένο σε γλώσσα προγραμματισμού "C". Λειτουργεί και διαβάζει οποιαδήποτε αρχεία κειμένου και μπορεί να «τρέχει» κατά διαστήματα ή συνεχώς, με χρονικά όρια τα όρια των πόρων του εκάστοτε συστήματος.

- Το **Swatch** (*Simple Log WATCHer*) είναι ένα πρόγραμμα αρχείων καταγραφής γεγονότων, που αφορά τα λειτουργικά συστήματα UNIX. Είναι γραμμένο σε γλώσσα προγραμματισμού Perl και παρακολουθεί όλα τα ενεργά μηνύματα του συστήματος, όπως αυτά καταγράφονται σε ένα αρχείο καταγραφής μέσω του βοηθητικού προγράμματος UNIX syslog. Το Swatch έχει σχεδιαστεί για να ενημερώνει τους διαχειριστές (*administrators*) του συστήματος από πιθανή υπερφόρτωση, λόγω μεγάλης ποσότητας των δεδομένων καταγραφής. Παρακολουθεί τα αρχεία καταγραφής, φιλτράρει τα ανεπιθύμητα στοιχεία και λαμβάνει μία ή περισσότερες ενέργειες *simple user-specified*, βασισμένες πάνω σε γνωστά πρότυπα αρχείων καταγραφής. Ενώ τα περισσότερα LFM καταγραφής και ανάλυσης σαρώνουν τα logs περιοδικά, το Swatch κάνει σαρώσεις σε πραγματικό χρόνο (*real-time*), όπου συγχρόνως δημιουργεί και τις κατάλληλες ειδοποιήσεις προς τους διαχειριστές για σοβαρά προβλήματα του συστήματος.

The Swatch Startup Script

```
----------------------------------------------------------------------
#!/bin/sh
# Simple Log Watcher Program

case "$1" in
'start')
            /usr/bin/swatch --daemon --config-file=/etc/swatch.conf --
tail-file=/var/log/auth.log --pid-file=/var/run/swatch.pid
            ;;
'stop')
            PID=`cat /var/run/swatch.pid`
            kill $PID
            ;;
*)
            echo "Usage: $0 { start | stop }
            ;;
esac
exit 0
----------------------------------------------------------------------
```

Ένα τυπικό παράδειγμα, ενός Log File Monitor θα μπορούσε να είναι ένα πρόγραμμα ανάλυσης για log files ενός HTTP server, το οποίο ψάχνει για εισβολείς που προσπαθούν να εκμεταλλευθούν γνωστά κενά ασφαλείας, όπως είναι η επίθεση - phf, (*PHF Attack*). Ακολούθως, παρουσιάζεται ένα δείγμα από μια ενέργεια ρύθμισης παραμέτρων του Swatch.

Sample Swatch configuration script

```
------------------------------
watchfor /[dD]enied|/DEN.*ED/
echo bold
bell 3
mail
exec "/etc/call_pager 5512345 08"
------------------------------
```

Στο συγκεκριμένο παράδειγμα, το Swatch αναζητά μια γραμμή που ταιριάζει με ένα μοτίβο που περιέχει τις λέξεις *denied* ή *Denied*, ή κάτι που ξεκινά με το "DEN" και τελειώνει με "ED". Όταν βρίσκει μια γραμμή που περιέχει μία από αυτές τις τρεις συμβολοσειρές αναζήτησης, τονίζει τη συγκεκριμένη γραμμή με έντονους χαρακτήρες και συγχρόνως ακούγεται ένας ήχος επί 3 (*bell 3*). Τέλος, εκτελεί το / etc / call_pager με τις παραπάνω δεδομένες επιλογές.

The Swatch Command

```
------------------------------------------------------------
/usr/bin/swatch --daemon --config-file=/etc/swatch.conf --tail-
file=/var/log/auth.log --pid-file=/var/run/swatch.pid
------------------------------------------------------------
```

Η χρήση ενός εργαλείου παρακολούθησης των αρχείων καταγραφής συμβάντων (LFM) σε συνδυασμό με ένα σύστημα ανίχνευσης εισβολών σε δίκτυα (NIDS), αποτελεί μία δυναμική και ιδιαίτερα ισχυρή προσέγγιση για την ασφάλεια ενός δικτύου.

6 Περιορισμοί & Παραποίηση Δεδομένων των Συστημάτων NIDS

Με βάση όσων αναφέρθηκαν μέχρι τώρα, τα συστήματα NIDS έχουν έναν αξιόπιστο και λειτουργικό μηχανισμό ασφαλείας, με την δημοτικότητα τους να αυξάνεται ανοδικά και σταθερά. Όμως, όπως όλα τα συστήματα ανίχνευσης εισβολών έτσι και αυτά έχουν μειονεκτήματα και συγκεκριμένους *περιορισμούς*.

Σε παλαιότερη έρευνα, που είχε γίνει από την **Secure Networks Inc.** σε πολλά συστήματα ανίχνευσης εισβολών δικτύου, αποκαλύφθηκαν πολλά τρωτά σημεία των συστημάτων NIDS. Τα αποτελέσματα των δοκιμών, έδειξαν κενά ασφαλείας σε διάφορα συστήματα NIDS τα οποία δίνουν σε έναν εισβολέα τη δυνατότητα να εκκινήσει μία επίθεση η οποία θα μπορούσε να περάσει εντελώς απαρατήρητη.
Η συγκεκριμένη έρευνα επικεντρώθηκε σε δύο βασικούς τομείς προβλημάτων: *Πρώτον*, στην δυνατότητα των συστημάτων NIDS να εντοπίζουν τα παραποιημένα δεδομένα και *δεύτερον*, στις επιθέσεις οι οποίες έχουν σαν στόχο το ίδιο το σύστημα NIDS. Το συμπέρασμα ήταν, ότι τα συστήματα ανίχνευσης εισβολών που παρακολουθούν την κυκλοφορία του δικτύου δεν είναι πάντοτε σε θέση να ανιχνεύουν αξιόπιστα όλες τις επιθέσεις.

Ουσιαστικά, κανένα από τα συστήματα ανίχνευσης εισβολών δικτύου που εξετάστηκαν δεν επανασυναρμολογούσε τα πακέτα IP με τον ίδιο ακριβώς τρόπο όπως τα συστήματα που επικοινωνούν μέσω IP. Το αποτέλεσμα ήταν, να υπάρχουν ορισμένες διαφορές μεταξύ του τι αντιλαμβάνεται το NIDS και του τι μπορεί να χειριστεί το σύστημα που λαμβάνει. Ένα από τα βασικά προβλήματα που αποκάλυψε η παραπάνω έρευνα, ήταν ότι ορισμένα NIDS δεν έλεγχαν το πεδίο *checksum* της κεφαλίδας των πακέτων IP. Ο έλεγχος αυτός γίνεται υποχρεωτικά από το σύστημα και η αλλαγή της τιμής αυτού του πεδίου θα υποχρέωνε το NIDS να καταγράψει διαφορετικό μέγεθος δεδομένων από αυτό που μπορεί να χειριστεί το σύστημα που λαμβάνει.

Αποκωδικοποίηση της κεφαλίδας ενός πακέτου IP

```
-----------------------------------------------------------------------

Packet Number : 13          3:52:02 PM
Length : 66 bytes
ether: ==================== Ethernet Datalink Layer ====================
       Station: Skylar -----> This_Workstation
       Type: 0x0800 (IP)
   ip: ====================== Internet Protocol ======================
       Station:10.1.1.100 ---->10.1.1.25
       Protocol: TCP
       Version: 4
       Header Length (32 bit words): 5
       Precedence: Routine
              Normal Delay, Normal Throughput, Normal Reliability
       Total length: 48
       Identification: 21249
       Fragmentation not allowed, Last fragment
       Fragment Offset: 0
       Time to Live: 128 seconds
   ==> Checksum: 0x9148(Valid)
  tcp: ----------------- Transmission Control Protocol -----------------
       Source Port: 258
       Destination Port: 1027
       Sequence Number: 417610
       Acknowledgement Number: 898472
       Data Offset (32-bit words): 5
       Window: 8510
       Control Bits: Acknowledgement Field is Valid (ACK)
              Push Function Requested (PSH)
       Checksum: 0x5DB5(Valid)
       Urgent Pointer: 0

-----------------------------------------------------------------------
```

Η έρευνα ανέφερε σαν παράδειγμα την επίθεση phf-cgi attack. Ενα NIDS επιχειρεί να ανιχνεύσει αυτή την επίθεση αναζητώντας το αλφαριθμητικό "*phf*" στο τμήμα δεδομένων όλων των αιτήσεων HTTP. Εάν, ανιχνεύσει αυτή την ακολουθία χαρακτήρων, το NIDS υποθέτει ότι λαμβάνει χώρα μία τέτοια επίθεση. Ένας «πονηρός» εισβολέας θα μπορούσε να στείλει μία σειρά πακέτων με έναν χαρακτήρα στο κάθε ένα για να παράγει το αλφαριθμητικό "*phoof*". Ο εισβολέας θα μπορούσε κατόπιν να τροποποιήσει την τιμή του πεδίου *checksum*, έτσι ώστε κάθε πακέτο που περιέχει το γράμμα "*o*" να έχει άκυρη τιμή σ' αυτό το πεδίο. Σαν αποτέλεσμα, αν και το σύστημα το οποίο ελέγχει την τιμή του πεδίου *checksum* βλέπει μόνο το αλφαριθμητικό *phf*, το NIDS το οποίο δεν ελέγχει αυτό το πεδίο θεωρεί ότι αυτό που μεταδόθηκε είναι το αλφαριθμητικό *phoof*.

Αν και αυτή η διαφορά στον τρόπο με τον οποίο αντιμετωπίζεται η κυκλοφορία του δικτύου είναι ένα έγκυρο ζήτημα το οποίο θα πρέπει να μας απασχολεί πάντοτε, δεν είναι ανυπέρβλητο. Μετά από την συγκεκριμένη έρευνα, τα περισσότερα «πακέτα» NIDS που παρουσίασαν αυτό το πρόβλημα, σε επόμενες νεότερες εκδόσεις τους το διόρθωσαν. Τέτοιου είδους προβλήματα είναι συνήθη σε οποιαδήποτε τεχνολογία και

σε οποιοδήποτε στάδιο της, για αυτό και δεν μπορεί να θεωρηθεί ότι η ασφάλεια των δικτύων είναι ένας στάσιμος και αμετάβλητος τομέας.

Δεν υπάρχει κανένας λόγος, να είναι οποιοδήποτε NIDS απευθείας προσπελάσιμο από κάθε υπολογιστή του δικτύου. Για την εξέταση της κυκλοφορίας του δικτύου δεν απαιτείται μόνο μία έγκυρη διεύθυνση IP. Τα μόνα συστήματα που χρειάζονται επικοινωνία μεταξύ τους είναι:

- Ο αισθητήρας (*sensor*)
- Η κονσόλα ελέγχου (*console*)
- Το firewall ή το router
- Ένα σύστημα το οποίο λειτουργεί σαν DNS server

Ο διαχωρισμός της επικοινωνίας που απαιτείται για την λειτουργία ενός NIDS από τις κανονικές επικοινωνίες του δικτύου, μπορεί να επιτευχθεί εύκολα χρησιμοποιώντας ένα ξεχωριστό ιδιωτικό δίκτυο με ιδιωτικό χώρο όλων των διευθύνσεων IP. Στην πραγματικότητα, η επικοινωνία των υποσυστημάτων του NIDS θα μπορούσε να γίνεται μέσω του κανονικού δικτύου, εν αγνοία των υπόλοιπων συστημάτων, εφόσον απενεργοποιηθεί η δυνατότητα δρομολόγησης. Αν και ο αισθητήρας (*sensor*) χρειάζεται πολλά πρωτόκολλα IP και κατά συνέπεια μία διεύθυνση IP στο κύριο δίκτυο, η διεύθυνση αυτή δεν χρειάζεται να είναι έγκυρη. Τις περισσότερες φορές, οι υπολογιστές στους οποίους τρέχει ο αισθητήρας και η κονσόλα ενός NIDS χρησιμοποιούν τον χώρο διευθύνσεων ενός υποδικτύου. Αν και μπορούν να επικοινωνούν μεταξύ τους και με τον DNS server, δεν μπορούν να επικοινωνούν με οποιοδήποτε άλλο σύστημα χρησιμοποιεί μία διεύθυνση από το υποδίκτυο. Αυτό οφείλεται στο γεγονός ότι δεν υπάρχουν συσκευές οι οποίες να δρομολογούν την κυκλοφορία μεταξύ αυτών των δυο τομέων του δικτύου. Επίσης, το NIDS δεν μπορεί να στέλνει ή να λαμβάνει δεδομένα από συστήματα τα οποία βρίσκονται έξω από το firewall. Μία πολύ βασική ερώτηση είναι, τι συμβαίνει όταν ο αισθητήρας του NIDS προσπαθεί να παρακολουθήσει όλη την κυκλοφορία του δικτύου. Όπως αναφέραμε, ο αισθητήρας ενός NIDS καταγράφει όλη την κυκλοφορία που διακινείται μέσα σε ένα δίκτυο, και όχι μόνο την κυκλοφορία του δικού του υποδικτύου. Αυτό σημαίνει ότι είναι απόλυτα «ικανός» να παρακολουθεί όλη την κυκλοφορία που διακινείται στο τοπικό δίκτυο, συμπεριλαμβανομένων και των επικοινωνιών μεταξύ των υπολογιστών

του υποδικτύου και του internet. Μπορεί κατόπιν να αναφέρει όλα τα ευρήματα του στην κονσόλα, μέσω του υποδικτύου.

Ο αισθητήρας και η κονσόλα ενός NIDS είναι ευπρόσβλητα σε «εσωτερικές» επιθέσεις. Εάν κάποιος χρήστης από ένα δίκτυο ανακαλύψει την διεύθυνση IP του NIDS, είναι απλό θέμα να αλλάξει ή να παραποιήσει την τοπική διεύθυνση για να προσπελάσει απευθείας τον αισθητήρα και την κονσόλα του NIDS σε ένα υποδίκτυο. Η κατάσταση αυτή χαρακτηρίζεται με τον όρο "**security through obscurity**", τα συστήματα παραμένουν ασφαλή μόνο για όσο χρόνο δεν γνωρίζει κανείς πού είναι κρυμμένα. Ωστόσο, καθιστώντας αυτά τα συστήματα εντελώς απροσπέλαστα από το Internet περιορίζεται δραματικά το εύρος των πιθανών σημείων επίθεσης και απλοποιείται η διαδικασία αποκάλυψης του εισβολέα.

Για την σωστή αντιμετώπιση των εσωτερικών επιθέσεων σε ένα σύστημα NIDS, πρέπει να επιλεγεί ένα NIDS το οποίο δεν απαιτεί το πρωτόκολλο IP. Για παράδειγμα, συστήματα ανίχνευσης εισβολών δικτύου που υποστηρίζουν την παρακολούθηση δικτύων από έναν υπολογιστή στον οποίο δεν έχει συσχετιστεί το πρωτόκολλο IP με το παρακολουθούμενο δίκτυο. Χωρίς διεύθυνση IP, το σύστημα είναι απροσπέλαστο - "άτρωτο" σε οποιαδήποτε μορφή βασιζόμενης στην IP επίθεση. Αυτό σημαίνει επίσης ότι θα πρέπει να γίνουν ειδικές προβλέψεις και ρυθμίσεις στην κονσόλα παρακολούθησης. Συγκεκριμένα, θα πρέπει είτε να λειτουργεί η κονσόλα του NIDS στο ίδιο σύστημα με τον αισθητήρα, είτε να γίνει εγκατάσταση μιας δεύτερης κάρτα δικτύου στο σύστημα με τον αισθητήρα έτσι ώστε να μπορεί να επικοινωνεί με την κονσόλα μέσω ενός ιδιωτικού υποδικτύου.

Ένας εισβολέας, για να αποκτήσει πρόσβαση στον αισθητήρα ή στην κονσόλα του NIDS θα πρέπει να περάσει από το firewall και να παραβιάσει τον DNS server. Εάν το NIDS δεν είναι απευθείας προσπελάσιμο, προφανώς δεν μπορεί να δέχεται απευθείας επιθέσεις. Όπως ισχύει και στην περίπτωση των firewalls, ο αισθητήρας ενός NIDS που χρησιμοποιεί το πρωτόκολλο IP για να επικοινωνήσει με το δίκτυο θα πρέπει να θωρακιστεί όσο το δυνατόν περισσότερο πριν χρησιμοποιηθεί.

Συγκεκριμένα, θα πρέπει να διασφαλιστεί ότι έχουν εγκατασταθεί οι πιο πρόσφατες διορθώσεις - ενημερώσεις για την ασφάλεια, καθώς και ότι το σύστημα δεν τρέχει οποιεσδήποτε μη απαραίτητες υπηρεσίες. Ένα καλά θωρακισμένο σύστημα είναι

πολύ πιο ανθεκτικό στις επιθέσεις και κατά συνέπεια αποτελεί πολύ καλύτερη πλατφόρμα για την εκτέλεση λειτουργιών ασφάλειας - παρακολούθησης.

6.1 Αντίμετρα των Συστημάτων NIDS

Μαζί με τις λειτουργίες καταγραφής και προειδοποίησης, ένα NIDS έχει στην διάθεση του δύο επιπλέον βασικές μεθόδους αποτροπής επιθέσεων: **1)** Τη διακοπή συνόδου (*session disruption*) και **2)** Την τροποποίηση των κανόνων φιλτραρίσματος (*filter rule manipulation*). Αυτές οι μέθοδοι αποτροπής διαφέρουν ανάλογα με το προϊόν, αλλά σε αυτή την ενότητα θα εξετάσουμε σε γενικό επίπεδο τα ισχυρά και αδύνατα τους σημεία.

❖ **Διακοπή συνόδου (*session disruption*)**

Η διακοπή συνόδου (*session disruption*), είναι το ευκολότερο αντίμετρο NIDS προς υλοποίηση. Αν και υπάρχουν ορισμένες παραλλαγές στην υλοποίηση του, στην πιο βασική της μορφή η διακοπή συνόδου εκτελείται από το NIDS, το οποίο διακόπτει την σύνοδο επικοινωνίας και στα δύο άκρα όταν ανιχνεύσει μία επίθεση. Μία τέτοια ενέργεια μπορεί να μην εμποδίζει τον εισβολέα να εκκινήσει επόμενες επιθέσεις, αλλά δεν θα του επιτρέψει να προκαλέσει περαιτέρω ζημιά κατά την τρέχουσα σύνοδο.

Για παράδειγμα, ο αισθητήρας του NIDS εντοπίζει έναν εισβολέα ο οποίος προσπαθεί να στείλει μία ακολουθία χαρακτήρων κατά την διάρκεια μιας συνόδου FTP. Εάν υλοποιηθεί σωστά, αυτή η επίθεση θα μπορούσε να παρέχει στον εισβολέα πρόσβαση μέσω του FTP σε ορισμένα παλαιότερα συστήματα. Αυτό το επίπεδο πρόσβασης παραχωρείται χωρίς να απαιτείται κάποια πιστοποίηση μέσω κωδικού πρόσβασης και ο εισβολέας θα μπορούσε να διαβάσει ή να γράψει σε οποιοδήποτε αρχείο του συστήματος.

Εάν ενεργοποιηθεί το αντίμετρο διακοπής συνόδου, ο αισθητήρας του NIDS θα εντοπίσει κατ' αρχήν και θα καταγράψει αυτή την πιθανή επίθεση και κατόπιν θα στείλει παραποιημένα πακέτα ACK/FIN στα δύο άκρα της συνόδου για να κλείσει τη σύνδεση. Ο αισθητήρας του NIDS κάνει αυτή την ενέργεια προσποιούμενος ότι είναι το σύστημα που βρίσκεται στο άλλο άκρο της σύνδεσης. Για παράδειγμα, θα έστελνε

ένα πακέτο ACK/FIN στον εισβολέα χρησιμοποιώντας την διεύθυνση προέλευσης IP, τους αριθμούς θύρας και τους αριθμούς ακολουθίας του FTP server. Ουσιαστικά αυτή η ενέργεια θα έκλεινε την σύνοδο επικοινωνίας, εμποδίζοντας τον εισβολέα να προσπελάσει το σύστημα αρχείων. Ανάλογα με το είδος του αισθητήρα NIDS, ο αισθητήρας μπορεί να προσπαθούσε να μπλοκάρει όλες τις επικοινωνίες από το επιτιθέμενο σύστημα, για ένα καθοριζόμενο από τον διαχειριστή χρονικό διάστημα.

Αν και η διακοπή συνόδου (*session disruption*) είναι ένα ισχυρό αντίμετρο, έχει τους περιορισμούς της. Για παράδειγμα, επιθέσεις όπως είναι η επίθεση κατακλυσμού σταγονιδίων (*teardrop attack*), η διακοπή συνόδου δεν είναι σε θέση να την μπλοκάρει. Αν και το NIDS έχει αρκετό χρόνο για να αντιδράσει στην συγκεκριμένη επίθεση μέσω FTP, δεν θα μπορούσε όμως να αντιδράσει αρκετά γρήγορα για να προστατέψει έναν υπολογιστή από μία επίθεση κατακλυσμού σταγονιδίων, αφού αρκεί ένα και μόνο *επιτιθέμενο πακέτο IP* για να καταρρεύσει το σύστημα.

❖ **Τροποποίηση κανόνων φιλτραρίσματος (*filter rule manipulation*)**

Οι αισθητήρες ορισμένων NIDS έχουν την δυνατότητα να τροποποιούν τους κανόνες φιλτραρίσματος ενός firewall ή ενός δρομολογητή (*router*), με στόχο την αποτροπή συνεχιζόμενων επιθέσεων. Το μέτρο αυτό εμποδίζει τη περαιτέρω κυκλοφορία από το επιτιθέμενο σύστημα στον στόχο του. Το NIDS προσθέτει ένα νέο φίλτρο στο εκάστοτε firewall, το οποίο μπλοκάρει όλη την εισερχόμενη κυκλοφορία που προέρχεται από την ύποπτη διεύθυνση IP. Αν και η τροποποίηση των κανόνων φιλτραρίσματος είναι μία ισχυρή καινοτομία, έχει και αυτή τους περιορισμούς της. Παρακάτω, θα αναδείξουμε τα πλεονεκτήματα και τα μειονεκτήματα της.

Η τροποποίηση των κανόνων φιλτραρίσματος, μπορεί να προκαλέσει πρόβλημα εάν ο εισβολέας παραποιήσει την διεύθυνση ενός server ή ενός site στο ίντερνετ, με το οποίο χρειάζεται να επικοινωνεί ο εκάστοτε οργανισμός ή επιχείρηση και εκκινήσει από εκεί την επίθεση του. Το NIDS θα αντιληφθεί την επίθεση και θα εφαρμόσει έναν κανόνα φιλτραρίσματος στον αντίστοιχο server ή site για να απαγορεύσει την πρόσβαση από την διεύθυνση προέλευσης του εισβολέα η οποία έχει παραποιηθεί και κατά συνέπεια θα καταστήσει αδύνατη την επικοινωνία του οργανισμού με το συγκεκριμένο site.

Η δυνατότητα τροποποίησης των κανόνων φιλτραρίσματος σε πραγματικό χρόνο (*real-time*), ενώ το σύστημα βρίσκεται σε λειτουργία θα μπορούσε να χρησιμοποιηθεί από κάποιον για την έναρξη μιας επίθεσης άρνησης εξυπηρέτησης (*DoS - Denial of Service*). Εάν ο εισβολέας τροποποιήσει σκόπιμα την διεύθυνση προέλευσης IP για να προκαλέσει αλλαγές στους κανόνες φιλτραρίσματος, το αντίστοιχο firewall θα σταματήσει τη διέλευση της κυκλοφορίας, λόγω υπερχείλισης. Κατά τη διάρκεια που αλλάζουν οι κανόνες φιλτραρίσματος, ακόμη και οι ενεργές σύνοδοι επικοινωνίας, μπορεί επίσης να τερματιστούν.

Πλεονεκτήματα

Από την πλευρά των πλεονεκτημάτων, η τροποποίηση των κανόνων φιλτραρίσματος μπορεί να αποτρέψει μία επίθεση προκαλώντας πολύ λιγότερη κυκλοφορία στο δίκτυο σε σύγκριση με τη διακοπή συνόδου (*session disruption*). Μόλις ένα NIDS τροποποιήσει τους κανόνες φιλτραρίσματος (*filter rule manipulation*), η σχετιζόμενη με την επίθεση κυκλοφορία σταματά. Με τη διακοπή συνόδου, το NIDS έπρεπε να προσπαθεί διαρκώς για να κλείσει κάθε σύνοδο που σχετίζεται με μία επίθεση. Για παράδειγμα, ένας «επίμονος» εισβολέας θα μπορούσε να αυξήσει σημαντικά την κυκλοφορία στο εκάστοτε δίκτυο.

Μειονεκτήματα

Υπάρχουν περιπτώσεις, όπου η τροποποίηση των κανόνων φιλτραρίσματος δεν είναι πάντα πλήρως αποτελεσματική. Ένα βασικό πρόβλημα είναι, τι γίνεται εάν η διεύθυνση προέλευσης IP του επιτιθέμενου συστήματος ανήκει στο δίκτυο που προστατεύει το firewall. Σε αυτή την περίπτωση η τροποποίηση των κανόνων φιλτραρίσματος δεν θα έχει καμία απολύτως επίδραση. Επειδή η σχετιζόμενη με την επίθεση κυκλοφορία δεν διέρχεται ποτέ από το firewall, δεν υπόκεινται στους κανόνες φιλτραρίσματος. Καταλαβαίνουμε λοιπόν, ότι η αλλαγή των κανόνων φιλτραρίσματος της κυκλοφορίας δεν πρόκειται να επηρεάσει την επίθεση.

Επίσης, ένας εισβολέας μπορεί να χρησιμοποιήσει μία παραποιημένη διεύθυνση IP αντί για της πραγματικής. Ακόμη κι αν το firewall αρχίσει να μπλοκάρει την αρχική επίθεση, το μόνο που χρειάζεται να κάνει ο εισβολέας είναι να επιλέξει μία διαφορετική και παραποιημένη διεύθυνση για να παρακάμψει τον νέο κανόνα.

Αντίθετα, με τη διακοπή συνόδου ένα NIDS «αντιδρά» βασιζόμενο στην *υπογραφή* των επιθέσεων και όχι στην διεύθυνση προέλευσης IP. Αυτό σημαίνει ότι σαν αντίμετρο, η διακοπή συνόδου μπορεί να απωθεί συνεχώς μία επίθεση ενώ η τροποποίηση των κανόνων φιλτραρίσματος όχι. Ένα NIDS θα μπορούσε να κάνει διαδοχικές αλλαγές στους κανόνες, επιχειρώντας να μπλοκάρει όλες τις παραποιημένες διευθύνσεις όταν τις ανιχνεύει. Εάν ο εισβολέας αλλάζει γρήγορα την διεύθυνση προέλευσης IP, το NIDS δεν θα καταφέρει ποτέ να τον «προλάβει». Απαιτείται κάποιο χρονικό διάστημα, συνήθως από 10 έως 20 δευτερόλεπτα μέχρι το NIDS και το firewall να ολοκληρώσουν την αλλαγή στους κανόνες φιλτραρίσματος.

Τις περισσότερες φορές, η τροποποίηση των κανόνων φιλτραρίσματος (*filter rule manipulation*) χρησιμοποιείται για επιθέσεις οι οποίες θεωρούνται εξαιρετικά καταστροφικές. Για παράδειγμα, παλαιότερα συστήματα που δεν είχαν εγκατεστημένες τις κατάλληλες διορθώσεις ήταν ευάλωτα σε επιθέσεις τύπου "*Ping Of Death - POD*". Μία μορφή επίθεσης η οποία προκαλεί την κατάρρευση του σωρού πρωτοκόλλων IP στο σύστημα - στόχο, στέλνοντας του ένα πακέτο ICMP (*Internet Control Message Protocol*) με πολύ μεγάλο μέγεθος. Σε ένα δίκτυο που περιλαμβάνει παλαιότερα συστήματα, θα πρέπει απαραιτήτως να έχουν εγκατασταθεί οι απαιτούμενες διορθώσεις, ώστε η τροποποίηση των κανόνων φιλτραρίσματος να είναι σε θέση να μπλοκάρει κάθε είδους επιθέσεις. Αν και οι συχνές αλλαγές των κανόνων θα μπορούσαν ενδεχομένως να προκαλέσουν μία κατάσταση άρνησης εξυπηρέτησης, το να επιτρέπεται οποιαδήποτε διέλευση κυκλοφορίας σε ένα δίκτυο είναι σίγουρο ότι θα διακόψει κάθε επικοινωνία που βασίζεται στις IP διευθύνσεις.

Δεν είναι συμβατά όλα τα NIDS με όλα τα firewalls και όλους τους δρομολογητές (*router*) που κυκλοφορούν στην αγορά. Συνεπώς, ενώ η διακοπή συνόδου μπορεί να χρησιμοποιείται από οποιοδήποτε NIDS την υποστηρίζει, η τροποποίηση των κανόνων φιλτραρίσματος μπορεί να χρησιμοποιείται μόνο εάν υπάρχει ένα συμβατό σύστημα firewall.

6.2 NIDS Consolidation: Ενοποίηση Δεδομένων

Σε μία προσπάθεια όχι μόνο να ξεπεραστούν οι περιορισμοί των παραδοσιακών IDS, αλλά επίσης να αναπτυχθούν νέες και πιο ενεργητικές μορφές άμυνας, οι πρόσφατες έρευνες που αφορούν τα συστήματα NIDS οδεύουν προς τη σύντηξη ή αλλιώς ενοποίηση των δεδομένων.

Συνδυάζοντας τις πληροφορίες των πακέτων που μεταδίδονται από τους servers, μαζί με πληροφορίες από άλλες πηγές, τα συστήματα NIDS μπορούν να βρίσκουν με πολύ μεγαλύτερη ακρίβεια δεδομένα για μία επίθεση.

Πηγές Δεδομένων

- **Μηνύματα Συστήματος**

Αν και συνήθως τα περισσότερα δεδομένα - πληροφορίες που σχετίζονται με ένα σύστημα καταχωρούνται στα αρχεία καταγραφής, αυτό δεν ισχύει πάντα είτε λόγω προβλημάτων στην διαμόρφωση είτε επειδή το λειτουργικό σύστημα δεν διαθέτει την παραπάνω δυνατότητα. Το σύστημα NIDS χρησιμοποιεί τα μηνύματα συστήματος για να δημιουργήσει μία συνολική εικόνα ολόκληρου του δικτύου η οποία επιτρέπει τον συνδυασμό - ενοποίηση των δεδομένων για την εύρεση πληροφοριών και ανάλυση μοτίβων από την κατάσταση του εκάστοτε δικτύου.

- **Αρχεία Καταγραφής**

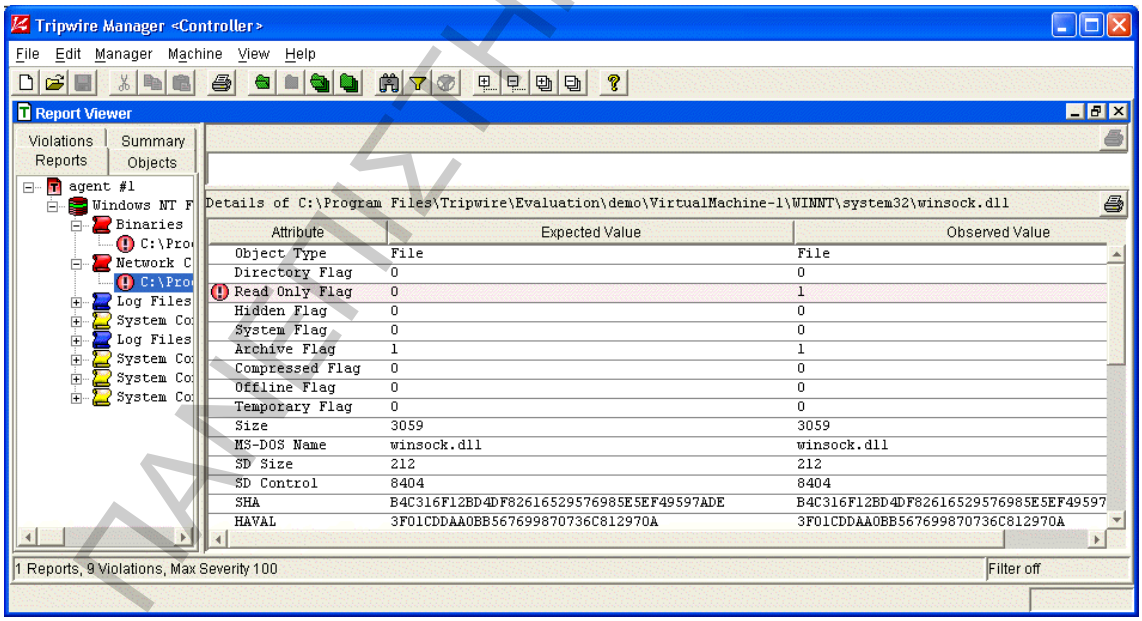
Τα περισσότερα λειτουργικά συστήματα μπορούν να διαμορφωθούν έτσι ώστε να καταγράφουν με κάθε λεπτομέρεια την κατάσταση τους ανά πάσα στιγμή, μαζί με συγκεκριμένα στοιχεία από κάθε *συστατικό* του εκάστοτε λειτουργικού συστήματος. Ένα τέτοιο παράδειγμα είναι ένας Mail Server, ο οποίος καταγράφει όχι μόνο τον χρόνο άφιξης των μηνυμάτων αλλά και τη διεύθυνση IP του server από τον οποίο προήλθαν. Ενα NIDS, θα μπορούσε να χρησιμοποιήσει αυτές τις πληροφορίες για να «ανιχνεύσει» την διαδρομή που ακολούθησε ένα κακόβουλο - παραπλανητικό e-mail και να ενημερώσει όλους τους mail servers ενός δικτύου ώστε να μπλοκάρουν οποιαδήποτε μηνύματα προέρχονται από τον συγκεκριμένο server.

- **Simple Network Management Protocol (SNMP)**

Το πρωτόκολλο SNMP, επιτρέπει στις συσκευές του δικτύου να επικοινωνούν με ένα κεντρικό σύστημα παρακολούθησης και να αναφέρουν όχι μόνο ποια δεδομένα μεταφέρονται, αλλά και το πώς λειτουργούν. Για παράδειγμα, ένας δρομολογητής ο οποίος ενημερώνει το σύστημα παρακολούθησης του δικτύου, αναφέροντας τον όγκο της κυκλοφορίας που διέρχεται ανά δευτερόλεπτο από ένα συγκεκριμένο σύστημα επικοινωνίας. Ένα σύστημα NIDS μπορεί να χρησιμοποιήσει αυτό το στοιχείο για να εξακριβώσει εάν κάποιος εισβολέας επιχειρεί μία επίθεση άρνησης εξυπηρέτησης.

- **Σύστημα Αρχείων**

Η χρήση κάποιου εργαλείου για τη δημιουργία και σύγκριση υπογραφών αρχείων μπορεί να προσθέσει μία ισχυρή τελευταία γραμμή άμυνας σε ένα δίκτυο. Αν και ένα τέτοιο εργαλείο δεν πρόκειται να εμποδίσει έναν εισβολέα να διεισδύσει σε ένα σύστημα μπορεί τουλάχιστον να ειδοποιήσει τους διαχειριστές του εκάστοτε συστήματος εάν τροποποιηθούν τα αρχεία. Ένα παράδειγμα, συστήματος σύγκρισης υπογραφών αρχείων είναι η χρήση του Tripwire που φαίνεται στην *Εικόνα 4*.



Εικόνα 4: Tripwire Manager - File Access Permission Change

- **Καταγραφή Εντολών**

Αρκετά λειτουργικά συστήματα δεν είναι σχεδιασμένα - διαμορφωμένα ώστε να καταγράφουν κάθε εντολή που εισάγουν οι χρήστες. Η ενοποίηση δεδομένων NIDS είναι ειδικά σχεδιασμένη ώστε να αντιπαρέρχεται τον παραπάνω περιορισμό, αποκαλύπτοντας μοτίβα τα οποία μπορεί να μην είναι εμφανή στα αρχεία καταγραφής του συστήματος. Ένα τέτοιο παράδειγμα θα μπορούσε να είναι μία εντολή σχεδιασμένη ώστε να διαγράφει εμπιστευτικές πληροφορίες ενός οργανισμού. Η συγκεκριμένη ενέργεια, αν και ιδιαίτερα καταστροφική δεν παραβιάζει και δεν επηρεάζει την ακεραιότητα του συστήματος.

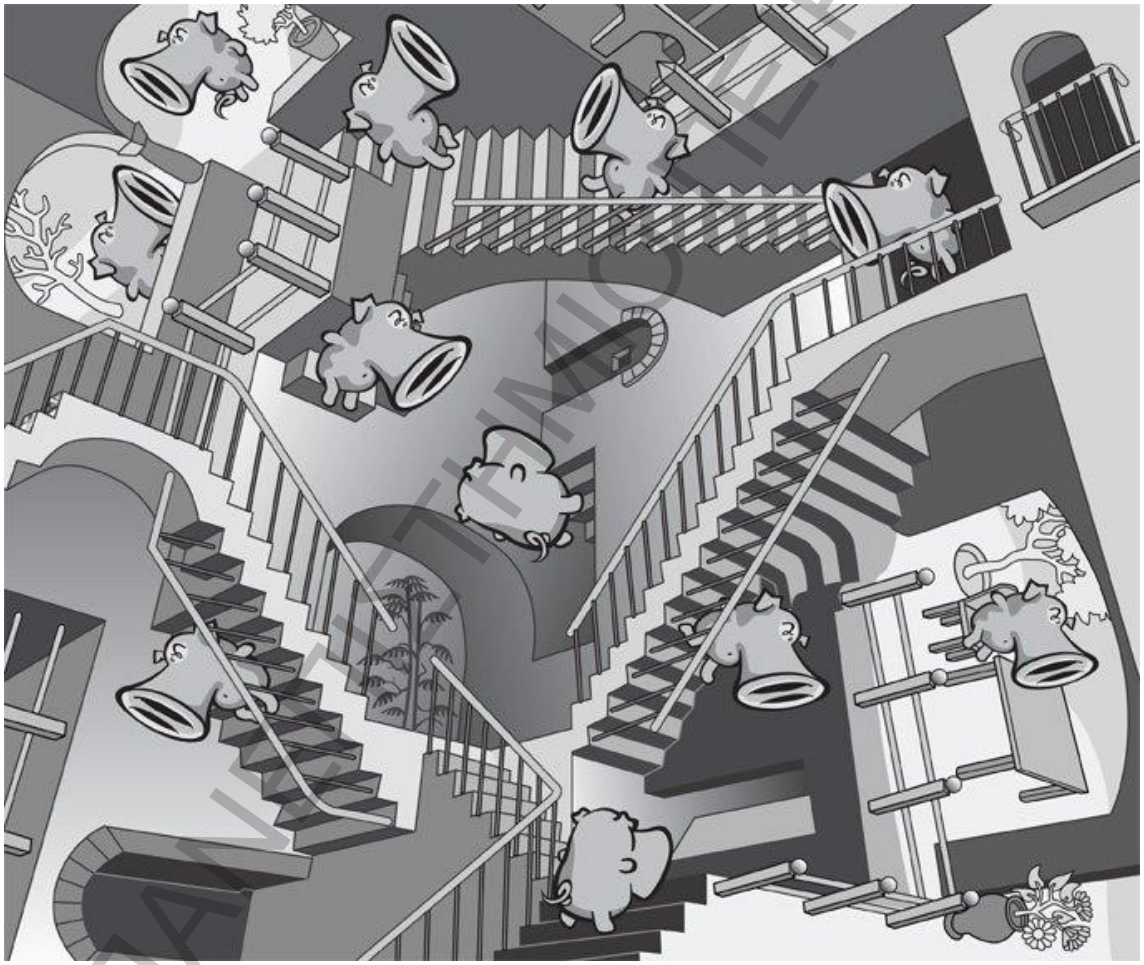
- **Συμπεριφορά των χρηστών**

Συμπληρωματικά ως προς την παρακολούθηση των εντολών που εισάγουν οι χρήστες, η κανονική συμπεριφορά των χρηστών δημιουργεί με τον χρόνο τα δικά της «μοτίβα». Αναλύοντας συνεχώς την δραστηριότητα των χρηστών έναντι του προφίλ κάθε λογαριασμού, τα συστήματα NIDS μπορούν να εξακριβώσουν εάν έχει παραβιαστεί ο λογαριασμός ενός χρήστη, αποτρέποντας οποιεσδήποτε περαιτέρω απόπειρες διείσδυσης στα εκάστοτε συστήματα.

Συμπερασματικά, αν και το σκεπτικό της ανάλυσης όλων των δεδομένων που σχετίζονται με τους χρήστες, την συμπεριφορά τους, τα συστήματα και το δίκτυο φαίνεται απλό, στην πραγματικότητα η ενοποίηση δεδομένων NIDS είναι αρκετά πολύπλοκη. Βασίζεται σε εξειδικευμένους μαθηματικούς τύπους και απαιτεί σημαντικότατους πόρους υποστήριξης για να λειτουργήσει αποτελεσματικά. Ακόμη και τότε όμως, είναι εξαιρετικά υποκειμενική και πειραματική. Ωστόσο, η ενοποίηση δεδομένων NIDS έχει φέρει ένα είδος επανάστασης στην ασφάλεια, μέσω μιας συνεργατικής προσέγγισης με κοινούς τρόπους αντίδρασης όλων των δικτύων που επηρεάζουν μία επίθεση.

NETWORK – INTRUSION DETECTION SYSTEMS

Snort: Ένα Δημοφιλές Σύστημα Ανίχνευσης Δικτυακών Επιθέσεων



" Όποιος επιθυμεί να προβλέψει το μέλλον πρέπει να συμβουλευτεί το παρελθόν, γιατί τα γεγονότα πάντα μοιάζουν με εκείνα που έχουν ήδη συντελεστεί "

–Niccolò Machiavelli

7 Snort: Ένα Δημοφιλές Σύστημα Ανίχνευσης Εισβολών Δικτύου

7.1 Εισαγωγή

Στον τομέα των Συστημάτων Ανίχνευσης Δικτυακών Επιθέσεων, το **Snort** αποτελεί ένα από τα πλέον διαδεδομένα, αποτελεσματικά και αξιόπιστα συστήματα. Πρόκειται για ένα σύστημα ανίχνευσης και αποτροπής - παρεμπόδισης εισβολών (*IDS/IPS*), όπου αναπτύχθηκε από την **Sourcefire** και διατίθεται δωρεάν υπό την **GNU General Public License** μέσω της επίσημης ιστοσελίδας του Snort - www.snort.org. Δημιουργός του είναι ο **Martin Roesch**, ο οποίος ξεκίνησε την ανάπτυξη του κώδικα σε γλώσσα προγραμματισμού "C", αξιοποιώντας μια σειρά από κανόνες (π.χ. *υπογραφές, πρωτόκολλα*) με σκοπό την προστασία του εκάστοτε δικτύου υπολογιστών. Θεωρείται, ένα από τα καλύτερα NIDS προγράμματα ανοιχτού κώδικα (*open source*), ικανό να πραγματοποιεί ανάλυση κίνησης όλου του δικτύου (*traffic analysis*), να καταγράφει και να αποθηκεύει σε πραγματικό χρόνο (*real-time*) κάθε πακέτο IP που εισέρχεται σε αυτό και τέλος να «αξιολογεί» την επικινδυνότητα του. Η ανάπτυξη και η υλοποίηση του έχει γίνει κατά τέτοιο τρόπο, ώστε να λειτουργεί σε μία μεγάλη ποικιλία λειτουργικών συστημάτων (π.χ. *Windows, Linux, MacOS, Solaris*).

Πολλοί αναλυτές και ειδικοί στο χώρο της Ασφάλειας Δικτύων και Συστημάτων, χαρακτηρίζουν το Snort ως ένα «ιδιαίτερο» lightweight IDS. Ο χαρακτηρισμός «ιδιαίτερο», βασίστηκε στο γεγονός της ελεύθερης διάθεσης του πηγαίου κώδικα, όπου αποτέλεσε κίνητρο και πρόκληση για ένα μεγάλο αριθμό ανθρώπων, με αποτέλεσμα τη σταδιακή αλλά σχετικά γρήγορη δημιουργία μίας μεγάλης κοινότητας από άτομα που ασχολούνται με το Snort. Ο στόχος της συγκεκριμένης κοινότητας, είναι να γίνει το Snort ακόμα πιο ευέλικτο, γρήγορο και αποτελεσματικό, με τη προσθήκη νέων λειτουργιών και τη βελτίωση των δυνατοτήτων του. Ο όρος *lightweight* έχει να κάνει κυρίως με το μικρό μέγεθος του «πακέτου» του Snort και την

ευκολία εφαρμογής - χρήσης του, το οποίο μαζί με τον πηγαίο του κώδικα δεν ξεπερνάει τα 6 MB.

7.2 Ο σκοπός του Snort

Όπως προαναφέρθηκε, το Snort είναι μία εφαρμογή ανοιχτού κώδικα (*open source*), κάτι που σημαίνει ότι εκτός από τη δωρεάν διάθεση της, οποιοσδήποτε θέλει μπορεί να δει και να επεξεργαστεί τον κώδικα της. Το παραπάνω, δίνει τη δυνατότητα στον καθένα να ρυθμίσει τον μεγάλο αριθμό υπογραφών (*signatures*) που έχει ήδη το Snort σύμφωνα με τις δικές του ανάγκες, ενώ επίσης μπορεί να φτιάξει και να προσθέσει τις δικές του «υπογραφές». Συνήθως, οι υπογραφές των επιθέσεων είναι δημόσια διαθέσιμες από διάφορους προγραμματιστές. Σε αυτό το σημείο, πρέπει να αναφερθεί ότι παλαιότερες εκδόσεις του Snort δεν χειρίζονταν καλά τον κατακερματισμό των πακέτων και χρησιμοποιούνταν σε δίκτυα μικρού μεγέθους και με μικρό σχετικά bandwidth. Πλέον, μέσα από διάφορες εκδόσεις και πολλές εξελίξεις στη μορφή του, το Snort έχει φτάσει στην έκδοση 2.9.3, η οποία έχει αλλάξει ριζικά τον μηχανισμό εντοπισμού υποστηρίζοντας ταχύτητες σύνδεσης - μετάδοσης σε δίκτυα της τάξης των Gigabit bandwidth. Στην *Εικόνα 5*, βλέπουμε μία «λίστα» μιας προσπάθειας σάρωσης θυρών.

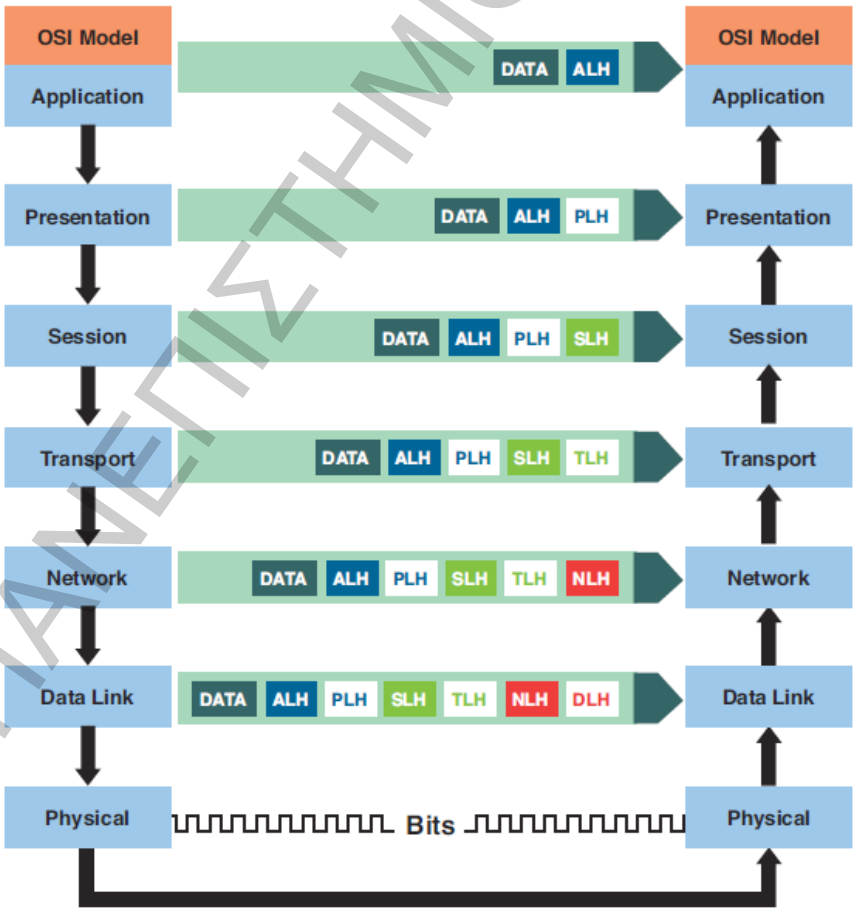
```
[**] spp_portscan: PORTSCAN DETECTED from 192.168.1.10 [**]
05/22-18:48:53.681227
[**] spp_portscan: portscan status from 192.168.1.10: 4 connections across
     1 hosts: TCP(0), UDP(4) [**]
05/22-18:49:14.180505
[**] spp_portscan: End of portscan from 192.168.1.10 [**]
05/22-18:49:34.180236
```

Εικόνα 5: Port Scan

Η αποτελεσματικότητα του Snort κινείται - βρίσκεται σε πολύ υψηλά επίπεδα, συγκρίσιμα και σε αρκετές περιπτώσεις ανώτερα από εκείνα των αντίστοιχων ακριβών εμπορικών προϊόντων. Οι βελτιώσεις που έχουν γίνει στις τελευταίες εκδόσεις του, έχει εκτοξεύσει το Snort στις πρώτες θέσεις των IDSs της αγοράς.

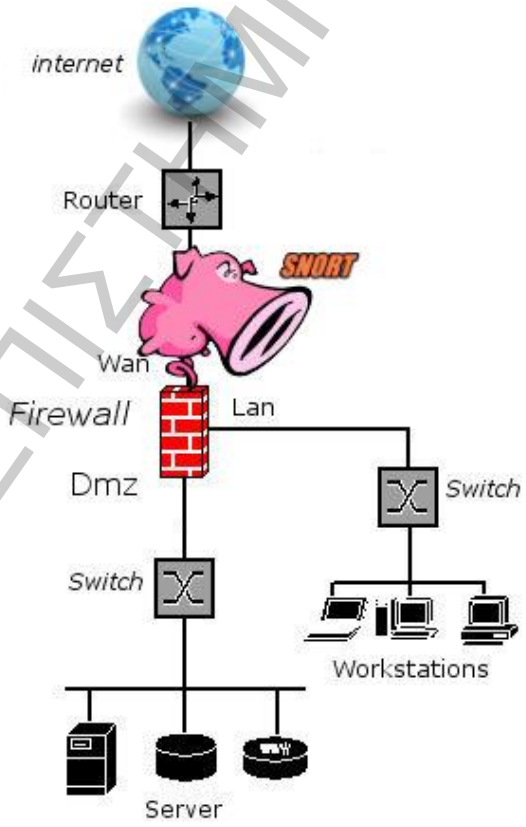
Αναλυτικότερα, το Snort μπορεί να «εκτελέσει» και να πραγματοποιήσει ανάλυση πρωτοκόλλων, εύρεση και ταίριασμα περιεχομένων από επίπεδο δικτύου μέχρι

επίπεδο εφαρμογής του μοντέλου του OSI, εξετάζοντας IP πακέτα για την ανίχνευση ενός μεγάλου εύρους δικτυακών επιθέσεων συστήματος και δικτύων. Επιθέσεις, όπως η υπερχείλιση καταχωρητών (*buffer overflows*), η κρυφή σάρωση πορτών (*stealth portscans*), οι CGI τύπου επιθέσεις (*CGI attacks*), οι διερευνήσεις πρωτοκόλλου SMB (*SMB probes*), οι επιθέσεις τύπου fingerprinting αναγνώρισης του λειτουργικού συστήματος (*OS fingerprinting attempts*) και πολλές άλλες, μπορούν να ανιχνευτούν και να αποτραπούν ενεργά. Οι ειδοποιήσεις για τις παραπάνω επιθέσεις γίνονται σε πραγματικό χρόνο (*real-time*) και εκτός από την καταγραφή και την αποθήκευση τους σε κάποιο alert file, το Snort έχει την δυνατότητα να ειδοποιεί - ενημερώνει με ένα κατάλληλο μήνυμα τον διαχειριστή (*administrator*) του εκάστοτε δικτύου μέσα από διαφορετικά προγράμματα, όπως το **Syslog**, αρχεία καθορισμένα από τον χρήστη ή οποιοδήποτε άλλο Unix socket. Τα πακέτα καταγράφονται στο δίσκο είτε σε ένα TCPDump Binary File, είτε σε μορφή text στα directories που δημιουργούνται με βάση την IP διεύθυνση της πηγής.



Εικόνα 6: **Μοντέλο Αναφοράς OSI**

Το Snort για να περιγράφει την κίνηση την οποία θα συλλέγει και θα εξετάζει, χρησιμοποιεί μία ευέλικτη γλώσσα προγραμματισμού καθοδηγούμενη από κανόνες, η οποία συνδυάζει όλα τα πλεονεκτήματα των μεθόδων ανίχνευσης που βασίζονται και στηρίζονται στις υπογραφές (*signatures*). Υπάρχουν ακόμη και πρόσθετα εργαλεία τα οποία επεξεργάζονται τα δεδομένα που αποθηκεύει το Snort και τα αποθηκεύουν σε οποιαδήποτε επιθυμητή μορφή βάσεων δεδομένων (π.χ. *XML*). Άλλα εργαλεία εκτελούν επιπλέον «εξωτερικές» δουλειές μέσω του Snort, όπως για παράδειγμα είναι η ανίχνευση portscan και η στατιστική ανίχνευση ανωμαλιών των διαφόρων πακέτων. Στο *Σχήμα 8*, απεικονίζεται ένα απλό παράδειγμα τοποθέτησης του Snort σε ένα δίκτυο. Σε αυτό το παράδειγμα, η κίνηση προς το διαδίκτυο «φεύγει» από το εσωτερικό δίκτυο μέσω του firewall προς τον εξωτερικό δρομολογητή (*router*). Το σύστημα που λειτουργεί το Snort, τοποθετείται ανάμεσα στον εξωτερικό δρομολογητή και στο firewall για να παρακολουθεί τόσο την εισερχόμενη όσο και την εξερχόμενη κίνηση του δικτύου.



Σχήμα 8: Snort

Όταν το Snort εκτελείται διαβάζει και ελέγχει όλους τους ενεργοποιημένους κανόνες (*rules*). Στην συνέχεια, οι κανόνες αυτοί έρχονται στον επιλογέα κανόνων που τους κατατάσσει σε ομάδες. Αυτή η επεξεργασία γίνεται πριν από οποιαδήποτε επεξεργασία πακέτων ή ροής δεδομένων. Αφού οι κανόνες του Snort έχουν πλέον χωριστεί σε ομάδες, κάθε εισερχόμενο ή εξερχόμενο πακέτο ελέγχεται με την αντίστοιχη ομάδα κανόνων ανάλογα με τις παραμέτρους του εκάστοτε πακέτου. Για παράδειγμα, υποθέτουμε ότι το Snort χρησιμοποιεί 2000 κανόνες, αυτοί οι κανόνες χωρίζονται σε ομάδες ανάλογα με το πρωτόκολλο μεταφοράς και εφαρμογής. Έτσι, 800 από αυτούς τους κανόνες μπορεί να μπουν στην ομάδα των HTTP Client κανόνων και 150 στην ομάδα των HTTP Server κανόνων. Όταν το Snort προχωρήσει στη διαδικασία επεξεργασίας των πακέτων, κάθε πακέτο περνά από τον διαχειριστή των ομάδων και ανάλογα με τις παραμέτρους του, ελέγχεται από την κατάλληλη ομάδα.

7.3 Χαρακτηριστικά εισαγωγής και ανάπτυξης του Snort

Όπως αναφέρθηκε στην προηγούμενη ενότητα, το Snort είναι ένα σύστημα προστασίας του εκάστοτε δικτύου, που του επιτρέπει να «ακροάζεται» και να αναλύει την οποιαδήποτε κίνηση μέσα σε αυτό για πιθανές επιθέσεις. Παρακάτω ακολουθεί μία λίστα που συγκεντρώνει και αναλύει τους λόγους, καθώς και τα κύρια χαρακτηριστικά που μας οδήγησαν στην επιλογή του Snort έναντι των αντίστοιχων ανταγωνιστικών εμπορικών συστημάτων.

Λόγοι επιλογής

1. Ελεύθερο λογισμικό

Το Snort είναι ένα ελεύθερο λογισμικό, που σημαίνει ότι μπορεί να χρησιμοποιηθεί δωρεάν. Αν και για να αποκτηθεί η πρόσβαση στους πιο πρόσφατους κανόνες (*rules*), θα πρέπει να καταβάλλουμε ένα ποσό ανάλογα με τη χρήση και το πλήθος των συστημάτων που θα χρησιμοποιηθούν, η διαφορά του σε σχέση με τα υπόλοιπα αντίστοιχα NIDS είναι ότι διατίθεται δωρεάν ο πηγαίος του κώδικας. Το γεγονός αυτό, δεν «δείχνει» το παραμικρό πρόβλημα σε σχέση με την ποιότητα των υπηρεσιών που μπορεί να προσφέρει.

2. Υποστήριξη πολλών λειτουργικών συστημάτων

Το Snort, είναι σχεδιασμένο και συμβατό έτσι ώστε να λειτουργεί σε πολλά και διαφορετικά λειτουργικά συστήματα. Μπορεί να λειτουργήσει τόσο σε πλατφόρμες όπως το **Linux**, το **Solaris** και το **MacOS** όσο και σε διανομές βασισμένες στα **Windows**.

3. Τακτική αναβάθμιση

Η αναβάθμιση και η «συντήρηση» του Snort γίνεται ανά τακτά χρονικά διαστήματα, με αποτέλεσμα οι διορθώσεις και οι προσθήκες των χαρακτηριστικών του να είναι άμεσα διαθέσιμες για την ομαλή λειτουργία του. Ακόμη, ο κάθε εγγεγραμμένος χρήστης στην κοινότητα του Snort μπορεί να κατεβάσει όλες τις νέες υπογραφές (*signatures*) επιθέσεων, μέσω της επίσημης ιστοσελίδας του.

4. Εύχρηστος και ευέλικτος τρόπος διαμόρφωσης

Ο εκάστοτε χρήστης, έχει τη δυνατότητα να ρυθμίζει και να διαμορφώνει με μεγάλη ευελιξία τις επιθέσεις που θα θέλει να ανιχνεύει το Snort για αυτόν, καθώς και τον τρόπο με τον οποίο θα παρουσιάζει τα αποτελέσματα του. Επιπλέον, ο τρόπος καθορισμού των κανόνων (*rules*) με βάση τους οποίους «συμπεριφέρεται» το σύστημα στα πακέτα που κυκλοφορούν σε ένα δίκτυο, είναι απλός και εύχρηστος. Είναι τόσο ρυθμίσιμο, που όλα τα στοιχεία που το απαρτίζουν (π.χ. *αρχεία διαμόρφωσης*), μαζί με τους κανόνες που αναφέραμε και χρησιμοποιεί είναι διαθέσιμα ώστε να προσαρμόζονται με βάση τις προτιμήσεις και ανάγκες των χρηστών του εκάστοτε δικτύου.

5. Ανάλυση και άμεση ειδοποίηση σε πραγματικό χρόνο

Η ικανότητα του Snort να αναλύει όλη τη διακινούμενη κίνηση, καταγράφοντας (*logging*) σε πραγματικό χρόνο τα IP πακέτα που εισέρχονται και κυκλοφορούν σε ένα δίκτυο, το καθιστά ένα από τα πλέον πολύτιμα και σημαντικά NIDS ανίχνευσης μεγάλου εύρους επιθέσεων (π.χ. *DoS/DDoS, portscans, buffer overflow attacks*). Επιπλέον, σε περίπτωση ύποπτης δραστηριότητας το Snort παρέχει στις τελευταίες του εκδόσεις άμεση ειδοποίηση των διαχειριστών των εκάστοτε δικτύων, η οποία μπορεί να πραγματοποιηθεί με πολλούς και ποικίλους τρόπους (π.χ. *e-mail*).

6. Φιλικό και γρήγορο περιβάλλον

Το κύριο χαρακτηριστικό που κάνει το Snort να μην καθυστερεί - κολλάει και να ξεχωρίζει από τα υπόλοιπα του είδους του, είναι η ελάχιστη «απαίτηση» που έχει σε κατανάλωση υπολογιστικών πόρων για την λειτουργία του. Επίσης, το περιβάλλον χρήσης του χαρακτηρίζεται από ένα ιδιαίτερα εύχρηστο και φιλικό προγραμματιστικό interface.

7. Μεγάλη υποστηριζόμενη κοινότητα

Το σημαντικό σε ένα υπολογιστικό σύστημα, είναι να βρίσκεται και να υπάρχει πίσω από αυτό μία καλά οργανωμένη κοινότητα (*community*) που θα το στηρίζει, θα το ενημερώνει και γενικώς θα το βελτιώνει με διάφορες νέες καινοτόμες ιδέες. Πλέον, το Snort αποτελεί ένα από τα πιο γνωστά και διαδεδομένα προγράμματα NIDS, χρησιμοποιούμενο από πολλές χιλιάδες χρηστών και με μία μεγάλη κοινότητα να το (υπο)στηρίζει. Αποτέλεσμα αυτών, είναι να υπάρχει ένα πολύ μεγάλο υλικό τεκμηρίωσης όσον αφορά την λειτουργία και τις «δυνατότητες» του.

7.4 Καταστάσεις λειτουργίας

Πολλοί άνθρωποι που βρίσκονται - «ανήκουν» στον χώρο της Εφαρμοσμένης Πληροφορικής και κυρίως άτομα από τον χώρο της Ασφάλειας Δικτύων και Συστημάτων, γνωρίζουν το Snort σαν ένα σύστημα ανίχνευσης εισβολών δικτύου (*NIDS*). Όμως στην ουσία δεν είναι μόνο αυτό, εκτός από την λειτουργία του σαν NIDS, το Snort είναι μία single-threaded εφαρμογή η οποία μπορεί να ρυθμιστεί έτσι ώστε να λειτουργεί με τέσσερις διαφορετικούς τρόπους.

Συνεπώς, το Snort χαρακτηρίζεται από τέσσερις διαφορετικές καταστάσεις λειτουργίας (*modes*), που τις περισσότερες φορές εκτελούνται σε συνδυασμό μεταξύ τους και ορίζονται από τον εκάστοτε χρήστη κατά την εκτέλεση του προγράμματος με τον ορισμό παραμέτρων (*parameter*).

<u>Τα modes λειτουργίας του Snort</u>

1) *Packet Sniffer Mode*

Όταν επιλέγεται αυτός ο τρόπος λειτουργίας, το Snort έχει την ικανότητα να διαβάζει τα εισερχόμενα πακέτα καθώς αυτά περνάνε από το εκάστοτε δίκτυο, να τα αποκωδικοποιεί και να τα εμφανίζει στην οθόνη με τις επικεφαλίδες και τα δεδομένα τους. Ο χρήστης, με τα διάφορα φίλτρα (π.χ. *Berkeley Packet Filter*) που διαθέτει και μπορεί να χρησιμοποιήσει, έχει την δυνατότητα να ορίσει το είδος των πακέτων που θα εμφανίζονται για αυτόν, όπως για παράδειγμα είναι το πρωτόκολλο, ο αποστολέας ή ο παραλήπτης ενός πακέτου. Η συγκεκριμένη λειτουργία του Snort είναι παρόμοια με αυτή του γνωστού εργαλείου tcpdump.

Για παράδειγμα, σε περίπτωση που θέλουμε το Snort να εκτελέσει και να εμφανίσει στην οθόνη όλα τα δεδομένα των πακέτων που περνάνε από το δίκτυο μας, καθώς και τις επικεφαλίδες τους, χρησιμοποιούμε την εντολή `./snort -vde`. Ενώ, αν θέλουμε μόνο να εμφανιστούν οι επικεφαλίδες των πακέτων TCP/IP στην οθόνη (*sniffer mode*), γράφουμε την εντολή `./snort -v`. Ως συνέπεια των παραπάνω, με την πρώτη εντολή έχουμε πλήρη πληροφορία των πακέτων, ενώ με την δεύτερη απλά εμφάνιση των εκάστοτε επικεφαλίδων των TCP/IP πακέτων.

Στην *Εικόνα 7*, παρουσιάζεται ένα δείγμα πακέτου που καταγράφηκε από το Snort με τη χρήση των παραπάνω εντολών.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

11/09-11:12:02.954779 10.1.1.6:1032 -> 10.1.1.8:23
TCP TTL:128 TOS:0x0 ID:31237 IpLen:20 DgmLen:59 DF
***AP*** Seq: 0x16B6DA  Ack: 0x1AF156C2  Win: 0x2217  TcpLen: 20
FF FC 23 FF FC 27 FF FC 24 FF FA 18 00 41 4E 53  ..#..'..$....ANS
49 FF F0                                         I..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Εικόνα 7: Παράδειγμα καταγραφής ενός πακέτου από το Snort

Η *Εικόνα 7*, μας δείχνει και αποτελεί ένα παράδειγμα καταγραφής ενός πακέτου που προέρχεται από ένα Telnet session. Στην <u>1η γραμμή</u>, βλέπουμε την ημερομηνία και την ώρα (11/09-11:12:02.954779) καταγραφής του συγκεκριμένου πακέτου, καθώς

και πληροφορίες για τα πεδία του IP header, όπως οι διευθύνσεις IP του αποστολέα (`10.1.1.6`) και του παραλήπτη (`10.1.1.8`), μαζί με τις πόρτες πηγής (`1032`) και προορισμού (`23`) του πακέτου.

Στη 2η γραμμή, εμφανίζονται - φαίνονται επίσης τα πεδία Transmission Control Protocol (`TCP`), το Time To Live (`TTL:128`), το Type Of Service (`TOS:0x0`), ο Identification Number (`ID:31237`), το IpLength (`20`), το Datagram Length (`DgmLen:59`) και τέλος τα DF Flags του IP header.

Εκτός, από τις πόρτες πηγής και προορισμού του πακέτου οι οποίες είναι μέρος της πληροφορίας του TCP header και εμφανίζονται στην 1η γραμμή δίπλα από τις αντίστοιχες IP διευθύνσεις, στην 3η γραμμή του παραπάνω δείγματος, βλέπουμε πως υπάρχουν επιπλέον σχετικές πληροφορίες που αφορούν τον TCP header του πακέτου. Αυτές είναι, τα TCP Flags (`AP`), το Sequence Number (`Seq:0x16B6DA`), το Acknowledgement Number (`Ack:0x1AF156C2`), το Window (`Win:0x2217`) και το TCP Header Length (`TcpLen:20`).

Τέλος, στο τελευταίο κομμάτι της παραπάνω καταγραφής εμφανίζεται στην δεξιά στήλη το payload του πακέτου σε δεκαεξαδική μορφή και στην αριστερή στήλη το payload σε μορφή απλού κειμένου ASCII που το καθιστά περισσότερο ευανάγνωστο.

Ένα ενιαίο ICMP echo στη διεύθυνση loopback `// root# ping -c 1 127.0.0.1`

==

```
11/20-10:57:18.991441 127.0.0.1 -> 127.0.0.1
ICMP TTL:64 TOS:0x0 ID:6482 IpLen:20 DgmLen:84
Type:8  Code:0  ID:3656   Seq:0  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

11/20-10:57:18.991448 127.0.0.1 -> 127.0.0.1
ICMP TTL:64 TOS:0x0 ID:6483 IpLen:20 DgmLen:84
Type:0  Code:0  ID:3656  Seq:0  ECHO REPLY
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

^C*** Caught Int-Signal

===============================================================================

Snort received 2 packets
    Analyzed: 2(100.000%)
    Dropped: 0(0.000%)
    Outstanding: 0(0.000%)
===============================================================================
Breakdown by protocol:
    TCP: 0          (0.000%)
    UDP: 0          (0.000%)
   ICMP: 2          (100.000%)
    ARP: 0          (0.000%)
  EAPOL: 0          (0.000%)
   IPv6: 0          (0.000%)
ETHLOOP: 0          (0.000%)
    IPX: 0          (0.000%)
   FRAG: 0          (0.000%)
  OTHER: 0          (0.000%)
DISCARD: 0          (0.000%)
===============================================================================
Action Stats:
ALERTS: 0
LOGGED: 0
PASSED: 0
===============================================================================
Snort exiting
```

Συγκεντρωτικά, θα μπορούσαμε να πούμε πως το Snort εκτός από την ευρέως γνωστή λειτουργία του *Detection Mode* που εμπεριέχεται σε ένα NIDS και την οποία θα δούμε αναλυτικότερα παρακάτω, μπορεί και να δουλέψει σαν ένας Sniffer (*sniffer mode*) που διαβάζει και καταγράφει (*logging*) όλα τα πακέτα που περνάνε από τη NIC (*Network Interface Controller*) στην οποία «ακούει» και τυπώνει στην οθόνη τις εκάστοτε επικεφαλίδες (*headers*) ή τα δεδομένα τους ή και τα δύο μαζί.

2) *Packet Logger Mode*

Με τη λειτουργία καταγραφής (*packet logger mode*), το Snort έχει τη δυνατότητα να <u>καταγράφει</u> και να <u>αποθηκεύει</u> στο δίσκο όλα τα εισερχόμενα πακέτα που διαβάζει από ένα δίκτυο. Η συγκεκριμένη διαδικασία είναι πολύ σημαντική, διότι αντί απλά να εμφανίζει τα πακέτα στην οθόνη, τα «συλλέγει» στο δίσκο έτσι ώστε να μπορεί να γίνει λεπτομερής ανάλυση αυτών οποιαδήποτε στιγμή χρειαστεί στο μέλλον. Το

αρχείο καταγραφής που δημιουργεί το Snort μπορεί να αποθηκευτεί σε διάφορα formats, ανάλογα με τις ανάγκες του εκάστοτε χρήστη. Για παράδειγμα, μπορεί να αποθηκεύσει τα πακέτα σε δυαδική (*binary*) μορφή, με την οποία το αρχείο έχει μικρότερο μέγεθος και μπορεί να διαβαστεί τόσο από το ίδιο το Snort όσο και από άλλα προγράμματα ανάλυσης πακέτων και πρωτοκόλλων που «υποστηρίζουν» τη συγκεκριμένη μορφή (π.χ. *tcpdump, ethereal*). Επίσης, τα πακέτα μπορούν να αποθηκευτούν και σε μορφές όπως είναι η XML και η Flat ASCII μορφή ώστε να είναι δυνατή η ανάγνωση τους ή να οργανωθούν σε βάσεις δεδομένων.

Για να λειτουργήσει το Snort σε Packet Logger mode, θα πρέπει να εκτελεστεί μέσα από αυτό η παράμετρος `-l`. Αναλυτικότερα, χρησιμοποιούμε την εντολή `./snort -dev -l ./log`, όπου `log` είναι το αρχείο καταγραφής (*log file*) στο οποίο θα καταγραφούν όλα τα πακέτα. Φυσικά, αυτό προϋποθέτει ότι έχουμε ένα directory με το όνομα log στο τρέχον directory. Σε περίπτωση που δεν υπάρχει αυτή η προϋπόθεση, τότε το Snort θα βγάλει άμεσα ένα μήνυμα σφάλματος - λάθους. Όταν το Snort «τρέχει» σε αυτή την κατάσταση λειτουργίας (*Packet logger Mode*), υπάρχει μία λεπτομερής καταγραφή όπου συλλέγει και αποθηκεύει κάθε πακέτο που βλέπει και το τοποθετεί σε διαφορετικούς φακέλους ανάλογα με την IP διεύθυνση του. Με τον τρόπο αυτό, η καταγραφή των πακέτων γίνεται σε μία εύκολα αναγνώσιμη μορφή για τον εκάστοτε χρήστη.

Επίσης, υπάρχει και ένας άλλος πιο γρήγορος τρόπος καταγραφής των πακέτων όπου η καταγραφή γίνεται σε ένα ενιαίο αρχείο σε μορφή tcpdump. Το αποτέλεσμα, είναι να έχουμε λίγες πληροφορίες για το πότε και που προσήλθε το πακέτο αλλά χωρίς καμία αποκωδικοποίηση.

Αναφορά των λεπτομερειών ενός πακέτου

```
------------------------------------------------------------------

        --== Initializing Snort ==--
Initializing Output Plugins!
Verifying Preprocessor Configurations!
TCPDUMP file reading mode.
Reading network traffic from "tests/snort.log.1164036694" file.
snaplen = 1514
...edited...
11/20-10:31:38.528360 127.0.0.1 -> 127.0.0.1
ICMP TTL:64 TOS:0x0 ID:5688 IpLen:20 DgmLen:84
Type:8  Code:0  ID:49479   Seq:0  ECHO
0x0000: 02 00 00 00 45 00 00 54 16 38 00 00 40 01 66 6F
....E..T.8..@.fo
0x0010: 7F 00 00 01 7F 00 00 01 08 00 2C 09 C1 47 00 00
..........,..G..
0x0020: 45 61 CA 5A 00 08 0F E8 08 09 0A 0B 0C 0D 0E 0F
Ea.Z............
0x0030: 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
................
0x0040: 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-
./
0x0050: 30 31 32 33 34 35 36 37                         01234567

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

11/20-10:31:38.528362 127.0.0.1 -> 127.0.0.1
ICMP TTL:64 TOS:0x0 ID:5689 IpLen:20 DgmLen:84
Type:0  Code:0  ID:49479  Seq:0  ECHO REPLY
0x0000: 02 00 00 00 45 00 00 54 16 39 00 00 40 01 66 6E
....E..T.9..@.fn
0x0010: 7F 00 00 01 7F 00 00 01 00 00 34 09 C1 47 00 00
..........4..G..
0x0020: 45 61 CA 5A 00 08 0F E8 08 09 0A 0B 0C 0D 0E 0F
Ea.Z............
0x0030: 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
................
0x0040: 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-
./
0x0050: 30 31 32 33 34 35 36 37                         01234567

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
------------------------------------------------------------------
```

Το παραπάνω αρχείο μπορεί να διαβαστεί από οποιονδήποτε **Libpcap-based sniffer**, όπως είναι για παράδειγμα το ήδη αναφερόμενο Tcpdump. Στο συγκεκριμένο αρχείο, βλέπουμε μία πλήρη και λεπτομερή αναφορά όλων των στοιχείων ενός πακέτου αρχικοποιώντας το Snort.

Μια άλλη επίσης σημαντική περίπτωση, είναι αυτή του οικιακού δικτύου (*home network*). Για να συνδεθεί ένας χρήστης στο οικιακό του δίκτυο, θα πρέπει απαραιτήτως να «ενημερώσει» το Snort για το ποιο δίκτυο είναι το δικό του. Αυτό γίνεται πολύ εύκολα με την εντολή `./snort -dev -l ./log -h 192.168.1.0/24`. Αυτός ο κανόνας «λέει» στο Snort να εμφανίσει το data link και τις επικεφαλίδες TCP/IP, καθώς και τα δεδομένα εφαρμογής που βρίσκονται στο directory `./log`. Εν συνεχεία, θα «συνδέσει» τα πακέτα που έχουν σχέση με το δίκτυο κλάσης `C 192.168.1.0`. Όλα τα εισερχόμενα πακέτα θα πρέπει να καταγράφονται σε

υποκαταλόγους του εκάστοτε log directory, με τα ονόματα του καταλόγου που βασίζονται στην διεύθυνση του απομακρυσμένου host (`non-192.168.1`).

Κλείνοντας, είναι σημαντικό να σημειωθεί ότι μία από τις βασικότερες και χρησιμότερες εντολές που αφορούν τα πακέτα ICMP είναι η εντολή `./snort -dvr packet.log icmp`. Η συγκεκριμένη εντολή, «βοηθά» τον εκάστοτε χρήστη να δει μέσω του Snort μόνο τα πακέτα ICMP από το αρχείο καταγραφής (*log file*), καθορίζοντας απλά ένα BPF φίλτρο στη γραμμή εντολών του. Το Packet logger Mode, δεν είναι απαραίτητο να λειτουργεί υποχρεωτικά μόνο του και ανεξάρτητα από τα άλλα modes. Συνήθως, λειτουργεί παράλληλα με το προαναφερόμενο Packet Sniffer Mode ή με το Detection Mode που θα αναφερθεί εκτενέστερα στην επόμενη παράγραφο.

3) *Detection Mode*

Η κύρια και πιο «ενδιαφέρουσα» μέθοδος λειτουργίας του Snort, είναι η ειδική κατηγορία Ανίχνευσης Εισβολών Δικτύου (*Network Intrusion Detection*). Όπως ήδη αναφέρθηκε, το Snort είναι ένα σύστημα ανίχνευσης εισβολών το οποίο ενεργεί σε επίπεδο δικτύου, καθώς έχει την δυνατότητα να παρακολουθεί και να αναλύει την κίνηση όλων των πακέτων σε ένα δίκτυο. Συνεπώς, τοποθετώντας κοντά σε αυτό μία δικτυακή συσκευή όπως ένα hub ή ένα switch, τα «γεγονότα» που παρακολουθεί και εξετάζει για την εμφάνιση μίας πιθανής επίθεσης, αφορούν την δραστηριότητα που παρατηρείται σε ένα δίκτυο.

Για να ενεργοποιηθεί το Network Intrusion Detection System (*NIDS*) mode, καλούμε την εντολή `./snort -dev -l ./log -h 192.168.1.0/24 -c snort.conf`, όπου `snort.conf` είναι το όνομα του αρχείου ρυθμίσεων (*configuration file*) του Snort μας. Με τον τρόπο αυτό, θα εφαρμοστούν για κάθε πακέτο οι κανόνες που διαμορφώνονται στο αρχείο `snort.conf`, αποφασίζοντας έτσι εάν πρέπει να ληφθεί οποιαδήποτε ενέργεια με βάση τον εκάστοτε κανόνα που εμπεριέχεται στο αρχείο. Σε περίπτωση που δεν καθοριστεί ένα output directory, θα προκαθοριστεί αυτομάτως το default directory `/var/log/snort`. Αναλυτικότερα, στην αρχή διαβάζεται το κατάλληλο αρχείο ρυθμίσεων στο οποίο υπάρχουν οι πληροφορίες για τα αρχεία κανόνων που θα χρησιμοποιηθούν (π.χ. *προεπεξεργαστές*). Αφού ολοκληρωθεί η

αρχικοποίηση (*initialization*) και γίνει ο διαχωρισμός των κανόνων στις κατάλληλες ομάδες, το σύστημα είναι έτοιμο να δεχτεί τα πακέτα. Αρχικά, κάθε πακέτο που λαμβάνει αποκωδικοποιείται περνώντας εφόσον υπάρχουν από τους προεπεξεργαστές (*preprocessors*). Εάν, το πακέτο δεν απορριφθεί τότε εξετάζεται για πιθανή επαλήθευση των κανόνων. Σε περίπτωση που επαληθευτούν, τότε το πακέτο καταγράφεται και αποθηκεύονται όλες οι πληροφορίες σχετικά με τους κανόνες που ενεργοποιήθηκαν. Ενώ, αν το πακέτο είναι «καθαρό» και δεν ενεργοποιηθεί ούτε ένας κανόνας τότε το Snort σταματά αυτομάτως και δεν εκτελείται καμία περαιτέρω ενέργεια. Το πιο σημαντικό στην όλη διεργασία, είναι πως η μορφή με την οποία καταγράφονται τα πακέτα αλλά και η λεπτομέρεια των πληροφοριών για τους κανόνες που ενεργοποιήθηκαν επιλέγονται ανάλογα με τις απαιτήσεις του εκάστοτε χρήστη.

Ενεργοποίηση Snort σε λειτουργία IDS

```
generic:/usr/local/snort # bin/snort -c
tests/snortconf.test
-i lo0 -l tests/
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Var 'lo0_ADDRESS' defined, value len = 19 chars, value =
127.0.0.0/255.0.0.0
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file tests/snortconf.test

+++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
1 Snort rules read...
1 Option Chains linked into 1 Chain Headers
0 Dynamic rules
+++++++++++++++++++++++++++++++++++++++++++++++++++

Tagged Packet Limit: 256

+----------------------[thresholding-config]----------------------
| memory-cap : 1048576 bytes
+----------------------[thresholding-global]----------------------
| none
+----------------------[thresholding-local]-----------------------
| none
+----------------------[suppression]-----------------------------
| none
------------------------------------------------------------------
Rule application order: ->activation->dynamic->pass->drop->alert->log
Log directory = tests/
Verifying Preprocessor Configurations!
0 out of 512 flowbits in use.
...edited...
^C*** Caught Int-Signal
==================================================================

Snort received 2 packets
    Analyzed: 2(100.000%)
    Dropped: 0(0.000%)
    Outstanding: 0(0.000%)
==================================================================
Breakdown by protocol:
    TCP: 0          (0.000%)
    UDP: 0          (0.000%)
   ICMP: 2          (100.000%)
    ARP: 0          (0.000%)
  EAPOL: 0          (0.000%)
   IPv6: 0          (0.000%)
ETHLOOP: 0          (0.000%)
    IPX: 0          (0.000%)
   FRAG: 0          (0.000%)
  OTHER: 0          (0.000%)
DISCARD: 0          (0.000%)
Action Stats:
ALERTS: 5
LOGGED: 5
PASSED: 0
==================================================================
Snort exiting
```

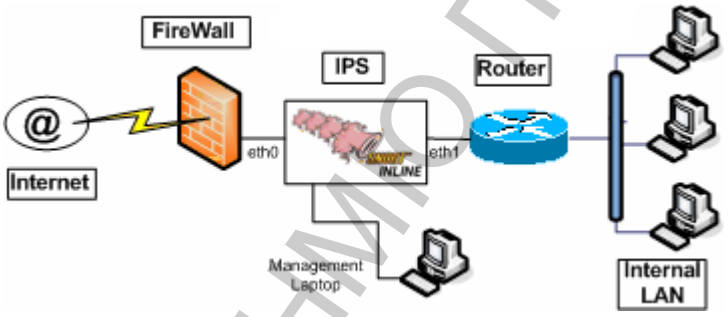
Όπως φαίνεται και στο παραπάνω στιγμιότυπο, χρησιμοποιούμε το switch –c για να ενεργοποιηθεί το Snort σε IDS λειτουργία στέλνοντας εν συνεχεία ένα «μη εμφανίσιμο» ICMP πακέτο. Παρατηρούμε, πως στην αρχικοποίηση του output το Snort αναγνωρίζει τον κανόνα (*single rule*) που περιέχεται στο αρχείο μας `/snortconf.test`.

Σημαντικό κομμάτι του Snort, είναι η τεχνική που χρησιμοποιεί για την ανίχνευση των δικτυακών επιθέσεων. Η βασική μέθοδος - διαδικασία ανίχνευσης των εισβολών γίνεται κυρίως μέσω των κανόνων. Αναλυτικότερα, το Snort χρησιμοποιεί συγκεκριμένα πρότυπα που έχουν οριστεί σαν υπογραφές (*signatures*) γνωστών επιθέσεων, ταιριάζοντας τα με ορισμένα χαρακτηριστικά μιας συγκεκριμένης «επικοινωνίας». Τα *πρότυπα εισβολών* μπορεί να είναι οποιοδήποτε χαρακτηριστικό πακέτο, συνθήκες, διάταξη ή «σχέσεις» μεταξύ των γεγονότων που οδηγούν σε μία εισβολή ή άλλη κακή χρήση. Συνήθως, τα παραπάνω πρότυπα - υπογραφές ονομάζονται και κανόνες (*rules*), που πολλές φορές χρησιμοποιούνται σαν «συνώνυμες» λέξεις, καθώς είναι κανόνες οι οποίοι περιγράφουν τα χαρακτηριστικά ενός πακέτου που μπορεί να είναι μέρος μίας γνωστής επίθεσης, όπως και την ενέργεια που θα εκτελεστεί κατά τον εντοπισμό του. Με τον τρόπο αυτό, κάθε πακέτο που εντοπίζεται από το Snort ελέγχεται για το αν έχει τα ίδια χαρακτηριστικά με αυτά που περιγράφονται από κάποιον συγκεκριμένο κανόνα και προσπαθεί να βρει αντιστοιχίες με βάση τις υπογραφές - κανόνες που έχει ορίσει ο εκάστοτε χρήστης, εκτελώντας στη συνέχεια τις αντίστοιχες λειτουργίες.

Παρόλα αυτά, υπάρχουν πολλές «διαθέσιμες» λειτουργίες ανίχνευσης πιθανών επιθέσεων που το Snort συνδυάζει στην λειτουργία ανάλυσης των γεγονότων του. Για παράδειγμα, η μέθοδος της *Statistical Anomaly Detection* και του *Protocol Verification* είναι κάποιες από τις τεχνικές που το Snort στις τελευταίες του εκδόσεις συνδυάζει και χρησιμοποιεί για την ανίχνευση του. Οι συγκεκριμένοι μηχανισμοί, υλοποιούνται κατά κύριο λόγο από τον νέο μηχανισμό επεξεργασίας των κανόνων του και από τους προεπεξεργαστές (*preprocessors*) οι οποίοι εξηγούνται αναλυτικά σε παρακάτω ενότητα.

4) *Prevention Mode/ Inline Mode*

Κλείνοντας, η τελευταία αλλά εξίσου σημαντική με τις προαναφερόμενες καταστάσεις λειτουργίας του Snort, είναι η λειτουργία *Inline* ή αλλιώς *Prevention Mode*. Το Snort ανήκει στην κατηγορία συστημάτων IDS/IPS, συνδυάζοντας χαρακτηριστικά τόσο από IDS (*Intrusion Detection System*) όσο και από IPS (*Intrusion Prevention System*). Τα IPS αποτελούν μία επέκταση των Συστημάτων Ανίχνευσης Εισβολών (*IDS*), επειδή εκτός από την ανάλυση της κίνησης σε επίπεδο δικτύου ή συστήματος, μπορούν επιπλέον να αντιδράσουν άμεσα σε πραγματικό χρόνο (*real-time*) και να εμποδίσουν ή να σταματήσουν μία εισβολή όταν αυτή εντοπιστεί. Συγκεκριμένα, ένα Intrusion Prevention System τοποθετείται **Inline** απορρίπτοντας επιθέσεις πριν καν εκείνες φτάσουν στον προορισμό τους.

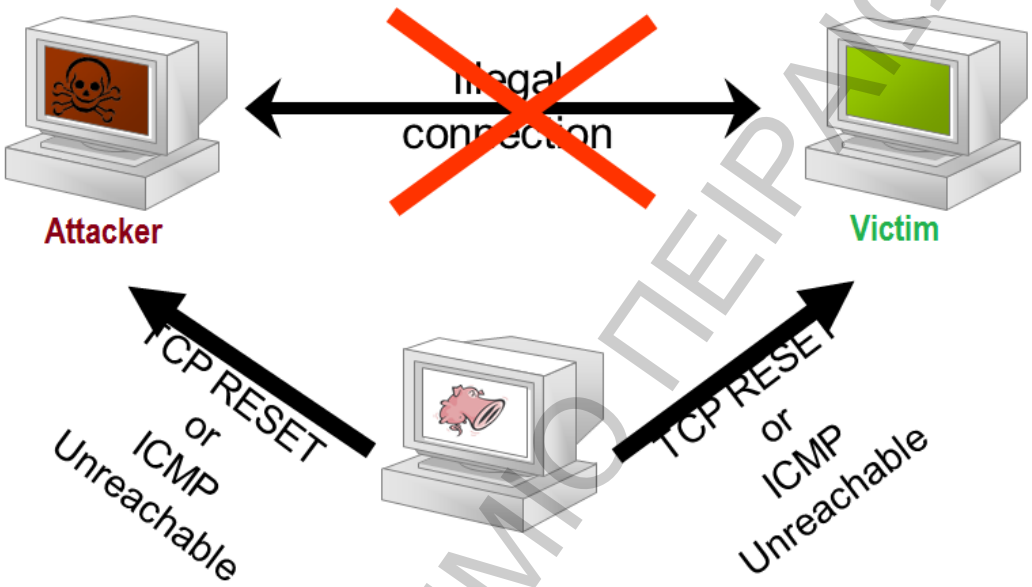


Σχήμα 9: Snort-Inline

Η λειτουργία Snort-inline είναι μια τροποποιημένη έκδοση του Snort που δέχεται πακέτα IP από **iptables** και **IPFW** μέσω της βιβλιοθήκης libipq (*linux*) ή sockets (*FreeBSD*), αντί της libpcap.

Στη συνέχεια, για την αποτροπή των απειλών ενός δικτύου χρησιμοποιεί νέες μορφές κανόνων (*drop, reject, sdrop*) «βοηθώντας» με αυτόν τον τρόπο τα iptables / IPFW, για το αν ένα πακέτο θα πρέπει να περάσει ή να απορριφθεί με βάση τους κανόνες του Snort (*Snort-rules*). Υπάρχουν, τρεις βασικοί τύποι κανόνων που μπορεί να χρησιμοποιηθούν κατά την εκτέλεση του Snort με Snort Inline.

1. Απορρίπτοντας τα επικίνδυνα πακέτα.
2. Τερματίζοντας μια σύνδεση στέλνοντας TCP reset ή ICMP Unreachable error messages στη μία ή και στις δύο πλευρές της σύνδεσης, στη περίπτωση που το πρωτόκολλο είναι TCP ή UDP αντίστοιχα.
3. Αποκλείοντας όλες τις συνδέσεις από κάποια συγκεκριμένη IP.



Εικόνα 8: **Reject rule type**

Snort με υποστήριξη MySQL

Όπως ήδη έχει γίνει αναφορά, το Snort έχει τη δυνατότητα να αποθηκεύει όλα τα «γεγονότα/συμβάντα» σε ένα δυαδικό αρχείο (*binary file*). Όταν όμως ο όγκος των δεδομένων είναι πολύ μεγάλος είναι σχεδόν επιτακτική η ανάγκη για οργάνωση τους, κάνοντας χρήση μίας βάσης δεδομένων που θα τα περιέχει. Για τον σκοπό αυτό, το Snort υποστηρίζει την απευθείας εγγραφή του «ρεύματος» εξόδου σε μία MySQL βάση δεδομένων.

Για την ενεργοποίηση της Inline λειτουργίας, θα πρέπει να προστεθεί ένα configure όρισμα και να γίνει πλήρης εγκατάσταση της εφαρμογής iptables που επιτρέπει στον εκάστοτε χρήστη να διαχειριστεί τους πίνακες που παρέχονται από το Linux kernel firewall. Για την εγκατάσταση του iptables, κατεβάζουμε τον πηγαίο κώδικα της τρέχουσας έκδοσης του και αποσυμπιέζουμε το αρχείο στον κατάλογο `/usr/src/`. Στη

συνέχεια, μεταβαίνουμε στον κατάλογο που αποθηκεύτηκαν τα αρχεία του πηγαίου κώδικα και εκτελούμε τις εντολές **./configure --enable-libipq** και **make && make install**. Βλέποντας, την αλλαγή που επιφέρει το iptables στην εγκατάσταση του Snort, η εκτέλεση του configure αλλάζει προσθέτοντας την παράμετρο " --enable-inline " και γίνεται **./configure --enable-dynamicplugin --with-mysql --enable-inline**. Τέλος, μένει να γίνει η μεταγλώττιση του κώδικα με τον ίδιο τρόπο όπως αναφέρθηκε και προηγουμένως.

Ως συμπέρασμα, το Snort χρησιμοποιείται για την ανάλυση/καταγραφή πακέτων, καθώς και για την ανίχνευση οποιασδήποτε προσπάθειας εισβολής σε ένα δίκτυο συνδυάζοντας «λειτουργίες» τόσο ανίχνευσης (*detection*) όσο και πρόληψης (*prevention*). Όταν αναλύει πακέτα, το Snort εμφανίζει απλώς τα πακέτα στην κονσόλα, ενώ με τη λειτουργία καταγραφής καταγράφει τα πακέτα στον σκληρό δίσκο. Κλείνοντας, όπως θα δούμε και στην επόμενη ενότητα οι λειτουργίες καταγραφής και προειδοποίησης του NIDS, οι οποίες εντοπίζουν την ύποπτη κυκλοφορία και αντιδρούν κατάλληλα, βασίζονται στην διαμόρφωση των κανόνων/υπογραφών και των μεθόδων αντίδρασης του Snort.

7.5 Κανόνες και Μέθοδοι Αντίδρασης του Snort

7.5.1 Εισαγωγή

Το πιο σημαντικό και καθοριστικό «κομμάτι» της ρύθμισης του Snort είναι η *δημιουργία*, η *σύγκριση* και η *αποτελεσματικότητα* των κανόνων και των υπογραφών του. Οι κανόνες (*rules*) και οι υπογραφές (*signatures*) χρησιμοποιούνται σαν ισοδύναμες έννοιες, καθώς «περιγράφουν» τα χαρακτηριστικά ενός πακέτου που μπορεί να είναι μέρος μίας γνωστής επίθεσης - εισβολής.

Το Snort βασίζεται σε συγκεκριμένους κανόνες για να αναγνωρίσει μία κακόβουλη δραστηριότητα ελέγχοντας υπογραφές, πρωτόκολλα και «ύποπτες» καταγραφές. Η ενημέρωση τους γίνεται από τον διαχειριστή (*administrator*) του συστήματος, υποδηλώνοντας όχι μόνο τι θέλουμε να ψάξει το σύστημα στα εισερχόμενα πακέτα αλλά επίσης και τι ενέργεια θα εκτελέσει σε περίπτωση που βρεθεί κάποιο πακέτο που να ικανοποιεί όλες τις συνθήκες του εκάστοτε κανόνα. Οι τελευταίες διανομές του Snort περιλαμβάνουν χιλιάδες έτοιμες υπογραφές (*signatures*) για χρήση στην ανίχνευση γνωστών μοτίβων επιθέσεων, οι οποίες ανανεώνονται διαρκώς ώστε να (ανα)καλύπτουν νέες επιθέσεις μέσα σε μικρό χρονικό διάστημα από την στιγμή που αυτές θα εμφανιστούν.

Παράλληλα, για την δημιουργία νέων - καινούργιων κανόνων το Snort προσφέρει μία μεγάλη γκάμα από *επιλογές συγγραφής*, με το πλεονέκτημα της προσαρμογής και της χρησιμοποίησης αυτών στις ανάγκες του εκάστοτε χρήστη. Οι συγκεκριμένες επιλογές (*options*), διακρίνονται από την ευκολία και την ευελιξία που «παρουσιάζουν» στη συγγραφή καινούργιων κανόνων, καθώς μπορούν να δημιουργηθούν από οποιονδήποτε γνωρίζει τα χαρακτηριστικά μιας επίθεσης όσον αφορά τα πακέτα που την υλοποιούν. Αποτέλεσμα των παραπάνω, είναι το σύστημα να εκτελεί λεπτομερής και σε βάθος περιγραφή των χαρακτηριστικών του κάθε πακέτου, ελέγχοντας κάθε φορά για τον εντοπισμό οποιασδήποτε μορφής επίθεσης. Αν όχι όλοι, οι περισσότεροι κανόνες έχουν γραφτεί με στόχο την *ανίχνευση* και την *απαγόρευση* των πακέτων που περιέχουν υπογραφές εισβολής (*intrusion signatures*). Συνήθως, τέτοιου είδους «μολυσμένα» πακέτα αποτελούν μέρος μίας προσπάθειας που έχει ως στόχο να πλήξει την ακεραιότητα του συστήματος. Αντιθέτως, για τα πακέτα που πληρούν

συγκεκριμένες προϋποθέσεις στα δεδομένα αλλά και στην επικεφαλίδα τους (*packet header*), υπάρχουν κανόνες οι οποίοι *επιτρέπουν* και *καθορίζουν* τη διέλευση τους.

7.5.2 Κανόνες

Οι κανόνες του Snort, δεν είναι τίποτα περισσότερο από καταχωρήσεις σε μορφή απλού κειμένου οι οποίες καθορίζουν τα χαρακτηριστικά των υπογραφών των επιθέσεων και επιλογές οι οποίες υπαγορεύουν τον τρόπο αντίδρασης του συστήματος όταν εντοπίζει τα συγκεκριμένα χαρακτηριστικά. Σε κάθε νέα έκδοση του Snort συμπεριλαμβάνονται συγκεκριμένοι κανόνες, ενώ ανά περιόδους γίνονται διαθέσιμα καινούργια ή ενημερωμένα αρχεία κανόνων ώστε να είναι εύκολος ο «συγχρονισμός» του Snort απέναντι σε νέες μορφές επίθεσης. Επιπλέον, ο κάθε χρήστης έχει τη δυνατότητα να γράφει νέους κανόνες, επιτρέποντας έτσι την προσαρμογή του Snort στις *απαιτήσεις* και στο *είδος* κίνησης που αναμένεται να υπάρχει στο εκάστοτε δίκτυο. Ενώ, κάνοντας ορισμένες τροποποιήσεις στο αρχείο ρυθμίσεων του Snort, μπορεί να ελέγχει και να επιλέγει τα αρχεία κανόνων που θα χρησιμοποιούνται στο σύστημα του.

Το Snort, χρησιμοποιεί μια απλή περιγραφική γλώσσα κανόνων η οποία προσφέρει πληθώρα επιλογών που την κάνουν αρκετά ευέλικτη και ισχυρή. Οι κανόνες του Snort βρίσκονται κατηγοριοποιημένοι σε διάφορα αρχεία με την κατάληξη ".rules", όπου το κάθε αρχείο από αυτά συνήθως περιλαμβάνει κανόνες που περιέχουν ένα συγκεκριμένο τύπο επίθεσης. Για παράδειγμα, το αρχείο "smtp.rules" περιέχει κανόνες που είναι γραμμένοι με σκοπό - στόχο τα πακέτα που σχετίζονται με εφαρμογές smtp.

Οι περισσότεροι κανόνες γράφονται σε μία και μόνο γραμμή. Σε παλαιότερες εκδόσεις του Snort και κυρίως πριν από την έκδοση 1.8 κάτι τέτοιο ήταν απαραίτητο. Εντούτοις, στις νεότερες εκδόσεις είναι εφικτό και συχνά προβλεπόμενο να έχουμε κανόνες που τείνουν να καταλαμβάνουν περισσότερο από μία γραμμή. Ένας τυπικός κανόνας απεικονίζεται ως εξής:

```
alert tcp any any -> 192.168.1.0/24 111 \
    (content:"|00 01 86 a5|"; msg:"mountd access";)
```

Οι κανόνες διαβάζονται με την σειρά που βρίσκονται στο κάθε αρχείο και τοποθετούνται σε μια εσωτερική δομή δεδομένων (*internal data structure*). Η συγκεκριμένη διαδικασία συμβαίνει κατά την εκκίνηση του Snort, αυτό σημαίνει αυτομάτως πως σε κάθε τροποποίηση των εκάστοτε κανόνων θα πρέπει να γίνεται επανεκκίνηση για να εισαχθούν οι νέες αλλαγές. Βάσει προεπιλογής, η σειρά με την οποία «αξιολογεί» τους κανόνες το Snort εμφανίζεται ακολούθως:

1. Ειδοποίηση κανόνων
2. Πέρασμα κανόνων
3. Σύνδεση κανόνων

Όπως φαντάζει λογικό, κατά τη διάρκεια της δημιουργίας των κανόνων, η εισαγωγή των ίδιων διευθύνσεων IP ή πρωτοκόλλων σε κάθε κανόνα μπορεί να καταστεί ιδιαίτερα κουραστική. Το Snort μας επιτρέπει να ορίσουμε μεταβλητές και τιμές οι οποίες μπορούν να χρησιμοποιούνται κατ' επανάληψη, σε πολλαπλούς κανόνες. Η απαιτούμενη σύνταξη είναι:

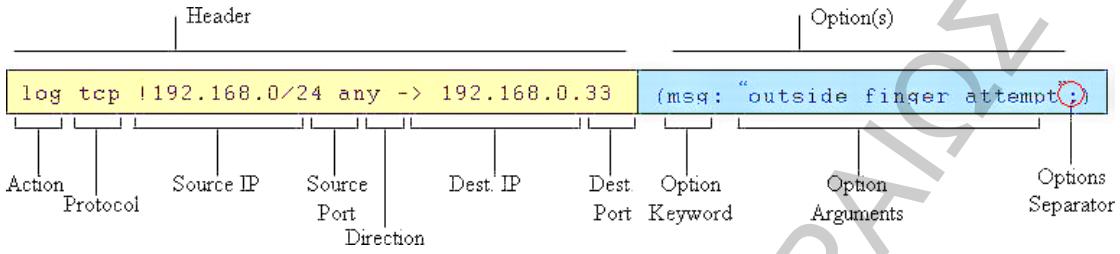
```
var <όνομα μεταβλητής> <τιμή μεταβλητής>
```

Συγκεκριμένα, υποθέτουμε ότι έχουμε πολλά ιδιωτικά δίκτυα βασιζόμενα στο πρωτόκολλο IP και θέλουμε το Snort να μας ειδοποιεί όταν παρατηρεί οποιεσδήποτε αιτήσεις σύνδεσης μέσω TCP (*Transmission Control Protocol*) προς οποιοδήποτε από τα εσωτερικά μας δίκτυα. Αντί, λοιπόν, να ορίσουμε έναν κανόνα για την διεύθυνση IP κάθε δικτύου, μπορούμε κάλλιστα να ορίσουμε μία μεταβλητή η οποία θα περιλαμβάνει όλες τις διευθύνσεις IP του εσωτερικού μας δικτύου, ως εξής:

```
var INT [172.16.0.0/12,192.168.1.0/24,192.168.2.0/24]
```

Στη συνέχεια, αφού οριστεί η συγκεκριμένη μεταβλητή, μπορούμε να την χρησιμοποιήσουμε σε έναν κανόνα ο οποίος θα μας ειδοποιεί όταν παρουσιάζονται οποιεσδήποτε απόπειρες υλοποίησης μιας σύνδεσης μέσω TCP με οποιοδήποτε από τα εσωτερικά μας δίκτυα.

Οι κανόνες του Snort αποτελούνται από δύο λογικές ενότητες - μέρη, **1)** την Επικεφαλίδα του κανόνα (*Rule Header*) και **2)** τις Ρυθμίσεις/Επιλογές του κανόνα (*Rule Options*). Στην *Εικόνα 9* που ακολουθεί περιγράφονται αναλυτικά τα δύο τμήματα ενός τυπικού κανόνα του Snort (*single rule*).



Εικόνα 9: Δείγμα ενός Κανόνα του Snort

Το πρώτο τμήμα του κανόνα, έως την αριστερή παρένθεση αποκαλείται «κεφαλίδα» του κανόνα (*rule header*). Στο τμήμα της κεφαλίδας, περιέχονται πληροφορίες για τις συνθήκες και την ενέργεια που θα εκτελείται όταν αναγνωρίζεται η υπογραφή μίας συγκεκριμένης επίθεσης, το εμπλεκόμενο πρωτόκολλο, οι διευθύνσεις IP προέλευσης/προορισμού μαζί με τις κατάλληλες netmasks και τέλος οι διευθύνσεις θυρών προέλευσης και προορισμού. Το δεύτερο τμήμα του κανόνα, αποκαλείται ενότητα επιλογών (*option section*). Εδώ, μπορούμε να καθορίσουμε ποια επιπλέον χαρακτηριστικά των πακέτων θα ελέγχει το Snort πριν κάνει οποιαδήποτε ενέργεια, καθώς επίσης και το συγκεκριμένο προειδοποιητικό μήνυμα που θα εμφανίζει.

Αξίζει να σημειωθεί, ότι το δεύτερο τμήμα των επιλογών (*rule options*) δεν είναι πάντοτε αναγκαίο για τη δήλωση ενός κανόνα. Ωστόσο, χρησιμοποιείται για να δίνεται ένας πιο ακριβής προσδιορισμός των πακέτων που θέλουμε να καταγράψουμε ή να απορρίψουμε. Επίσης, όλα τα προσδιοριστικά στοιχεία ενός κανόνα πρέπει να ισχύουν πάντοτε για ένα πακέτο ώστε να εκτελεστεί η εκάστοτε ενέργεια του κανόνα.

ΕΝΕΡΓΕΙΕΣ

Το πρώτο στοιχείο σε κάθε κανόνα είναι η ενέργεια που θα εκτελεστεί (*rule action*). Οι ενέργειες ενός κανόνα ορίζουν τι «θα κάνει» το Snort όταν ανιχνεύει ένα πακέτο το οποίο ταιριάζει με τις προδιαγραφές ενός κανόνα. Αναλυτικά, η επικεφαλίδα κάθε κανόνα περιέχει τις πληροφορίες που καθορίζουν από που έρχεται ένα «ύποπτο» πακέτο, που πηγαίνει, τι είδους είναι και τι πρέπει να κάνουμε στην περίπτωση που μας έρθει ένα πακέτο με τα συγκεκριμένα χαρακτηριστικά του κανόνα. Το Snort υποστηρίζει πέντε προεπιλεγμένες ενέργειες (*default actions*):

- **Alert (συναγερμός)** - Παράγει ένα προειδοποιητικό μήνυμα και καταγράφει το πακέτο, χρησιμοποιώντας την καθοριζόμενη από τον χρήστη μέθοδο.
- **Log (καταγραφή)** - Καταγράφει μόνο το πακέτο.
- **Pass (διέλευση)** - Επιτρέπει την διέλευση του πακέτου χωρίς να κάνει καμία ενέργεια.
- **Activate (ενεργοποίηση)** - Παράγει ένα προειδοποιητικό μήνυμα και κατόπιν ενεργοποιεί έναν «δυναμικό» (*Dynamic*) κανόνα.
- **Dynamic (δυναμικός κανόνας)** - Ένας ειδικός τύπος κανόνα καταγραφής ο οποίος ενεργοποιείται από έναν άλλο κανόνα και όχι από ένα πακέτο, ο οποίος με την σειρά του, αναφέρεται σαν κανόνας ενεργοποίησης (*activate rule*).

Ένα από τα χρησιμότερα και πιο «ευέλικτα» χαρακτηριστικά του Snort, είναι η δυνατότητα προσαρμογής της λειτουργίας του. Με άλλα λόγια, μπορούμε και μας επιτρέπεται να δημιουργήσουμε τις δικές μας ενέργειες κανόνων, γνωστές σαν τύποι κανόνων (*rule types*) και να τις χρησιμοποιήσουμε στο Snort όπως θα χρησιμοποιούσαμε οποιαδήποτε άλλη ενέργεια. Το ακόλουθο παράδειγμα κανόνα καταγράφει τις πληροφορίες σε μία βάση δεδομένων:

```
ruletype dbalert
{
     type output
     output database: mysql, user=vroussos dbname=dbsnort host=localhost
}
```

Ρυθμίσεις Συναγερμών

Στην αρχική του διαμόρφωση, το Snort καταγράφει τις «ανευρέσεις» σε μορφή ASCII, όπου μία ανεύρεση [hit] θεωρείται ότι συμβαίνει όταν το Snort ανιχνεύει ένα πακέτο το οποίο ταιριάζει με τις συνθήκες που έχουμε ορίσει σε μία ή περισσότερες ομάδες κανόνων. Όταν οριστεί ένας συναγερμός, εμφανίζει το προειδοποιητικό μήνυμα μαζί με την κεφαλίδα του «ύποπτου» πακέτου. Στον πίνακα που ακολουθεί, αναφέρονται οι έξι τύποι συναγερμών που υποστηρίζει το Snort.

Επιλογή	Περιγραφή
`-A fast`	Γρήγορη κατάσταση συναγερμού. Περιλαμβάνει μία ένδειξη χρόνου, το μήνυμα, τις διευθύνσεις IP προέλευσης/προορισμού και τους αριθμούς θυρών (*ports*).
`-A full`	Πλήρης κατάσταση συναγερμού. Είναι η προεπιλεγμένη (*default*) κατάσταση συναγερμού που χρησιμοποιείται αυτόματα αν δεν καθοριστεί μια λειτουργία, εμφανίζοντας όλες τις πληροφορίες.
`-A unsock`	Αποστέλλει τους συναγερμούς σε ένα socket του UNIX, έτσι ώστε να μπορούν να καταγραφούν από ένα άλλο πρόγραμμα που τρέχει στην ίδια μηχανή.
`-A none`	Δεν παράγονται συναγερμοί.
`-A console`	Στέλνει «γρήγορα» προειδοποιήσεις στην κονσόλα - οθόνη.
`-A cmg`	Δημιουργεί cmg-ειδοποιήσεις.

Πίνακας 3: Λειτουργίες Συναγερμών

Από τον παραπάνω πίνακα, διακρίνεται εύκολα ότι όλοι οι ανωτέρω τύποι συναγερμών είναι προσβάσιμοι με το switch -A της γραμμής εντολών (*command line*). Εκτός, από την καταγραφή των πακέτων ως προεπιλογή αποκωδικοποίησης τους σε μορφή ASCII, μπορούν να καταγραφούν και σε ένα δυαδικό αρχείο (*binary log file*) μέσω του switch -b της γραμμής εντολών. Για παράδειγμα, με την εντολή `./snort -c snort.conf -b -A fast -l /var/log/snort`, κάνουμε χρήση της γρήγορης κατάστασης συναγερμού (*fast alerting mode*) και αποθηκεύουμε σε δυαδική μορφή (*binary file*) όλα τα logs στον προεπιλεγμένο κατάλογο `/var/log/snort`.

Στο σημείο αυτό θα πρέπει να αναφερθεί, πως τα log messages αποθηκεύονται «εξ ορισμού» σε κάποιο αρχείο. Το Snort έχει σαν προεπιλογή αποθήκευσης των logs τον

κατάλογο `/var/log/snort`, ωστόσο η θέση που αποθηκεύονται τα συγκεκριμένα μηνύματα μπορεί να αλλάξει κατά την εκκίνηση από τη γραμμή εντολών του.

Τέλος, θα πρέπει να είμαστε ιδιαίτερα προσεκτικοί με τις αρχικές ρυθμίσεις για τους συναγερμούς. Η βασικότερη αδυναμία οποιουδήποτε συστήματος NIDS είναι ο τεράστιος αριθμός ψεύτικων συναγερμών. Με άλλα λόγια, ένα σύστημα θα θεωρεί ότι δέχεται συνεχώς επιθέσεις από παντού, όταν στην πραγματικότητα ο κύριος «υπαίτιος» είναι η *αναποτελεσματική υλοποίηση* πολλών πρωτοκόλλων δικτύωσης.

ΠΡΩΤΟΚΟΛΛΑ

Το επόμενο πεδίο στην επικεφαλίδα ενός κανόνα είναι το πρωτόκολλο (*protocol*). Το Snort «υποστηρίζει» και αναλύει πακέτα για ύποπτη συμπεριφορά από τέσσερα πρωτόκολλα: TCP (*Transmission Control Protocol*), UDP (*User Datagram Protocol*), ICMP (*Internet Control Message Protocol*) και IP (*Internet Protocol*). Το TCP εξετάζει τις αιτήσεις σύνδεσης, κάτι το οποίο χρησιμοποιείται κυρίως από τους crackers για κατανεμημένες επιθέσεις άρνησης εξυπηρέτησης (*DDoS attack*). Το UDP μπορεί επίσης να χρησιμοποιηθεί για την εκκίνηση μιας επίθεσης DDoS, αλλά αυτό μπλοκάρεται - διακόπτεται σχετικά εύκολα. Το ICMP, ο λεγόμενος «αγγελιοφόρος» των πρωτοκόλλων TCP/IP, μπορεί να χρησιμοποιηθεί για την *παράνομη* διέλευση πληροφοριών από firewalls. Τέλος, οι πληροφορίες του IP είναι το πιο «εύκολο θύμα» διάφορων κακόβουλων επιθέσεων, εκτός κι αν χρησιμοποιείται κάποια ασφαλής παραλλαγή του IP, όπως είναι για παράδειγμα το πρωτόκολλο IPSec (*IP Security*).

ΔΙΕΥΘΥΝΣΕΙΣ IP

Μετά το πρωτόκολλο βρίσκεται το τμήμα που περιέχει τις πληροφορίες για τις διευθύνσεις IP και τις θύρες ενός κανόνα. Οι διευθύνσεις IP ακολουθούν το «σχήμα ονομασίας» **Classless Inter-Domain Routing (*CIDR*)** βάσει του οποίου η διεύθυνση IP ακολουθείται από έναν χαρακτήρα " / " και τον αριθμό των ψηφίων που χρησιμοποιούνται για το εκάστοτε υποδίκτυο. Για παράδειγμα, βάσει του συστήματος CIDR, το δίκτυο `192.168.1.0` με διεύθυνση υποδικτύου `255.255.255.0` αναπαρίσταται ως εξής: `192.168.1.0/24`

Επιπλέον, μπορούμε να χρησιμοποιήσουμε μεταβλητές για την αναπαράσταση διευθύνσεων IP, λιστών πολλαπλών διευθύνσεων, οι οποίες χωρίζονται μεταξύ τους με κόμματα "`[192.168.1.0/24,10.1.1.0/24]`", καθώς και την δεσμευμένη λέξη *any*, που δηλώνει ότι ένας κανόνας ισχύει για οποιαδήποτε διεύθυνση IP.

Στις διευθύνσεις, μπορεί να εφαρμοστεί και ένας τελεστής άρνησης (*negation operator*) ο οποίος συμβολίζεται με "!", δηλώνοντας στο Snort ότι όλες οι διευθύνσεις ταιριάζουν εκτός από την αναγραφόμενη. Για παράδειγμα, στην ακόλουθη κεφαλίδα κανόνα (*rule header*) η διεύθυνση αποστολέα είναι ρυθμισμένη με έναν τελεστή άρνησης έτσι ώστε να μην δημιουργούνται συναγερμοί - ειδοποιήσεις σε κινήσεις που προέρχονται εντός του τοπικού μας δικτύου παρά μόνο εκτός του δικτύου `192.168.1.0/24`.

```
alert tcp !192.168.1.0/24 any -> 192.168.1.0/24 111 \
    (content:"|00 01 86 a5|"; msg:"external mountd access";)
```

<u>ΑΡΙΘΜΟΙ ΘΥΡΩΝ</u>

Όμοια με τις διευθύνσεις IP, στην περίπτωση που θέλουμε να δηλώσουμε ότι ένας κανόνας ισχύει για όλες τις θύρες μπορούμε να χρησιμοποιήσουμε την δεσμευμένη λέξη *any* και για τους αριθμούς των θυρών. Οι αριθμοί θυρών, μπορούν να καθορίζονται σαν περιοχές τιμών ή εξαιρώντας περιοχές τιμών (*negation*). Μία περιοχή τιμών που αντιπροσωπεύει αριθμούς θυρών <u>καθορίζεται</u> χρησιμοποιώντας την άνω-κάτω τελεία ":" για τον διαχωρισμό των δύο τιμών που αντιστοιχούν στην αρχή και το τέλος της εκάστοτε περιοχής. Για παράδειγμα, το `1:1024` αντιπροσωπεύει όλες τις θύρες με αριθμούς μεταξύ `1` και `1024`. Για την εξαίρεση ορισμένων αριθμών θυρών χρησιμοποιείται ο χαρακτήρας του θαυμαστικού "!" . Ο χαρακτήρας αυτός μπορεί επιπλέον να συνδυαστεί με μία περιοχή τιμών για την εξαίρεση μιας περιοχής διευθύνσεων. Επίσης, υπάρχουν και στατικές θύρες (*static ports*) οι οποίες «υποδεικνύονται» από έναν ενιαίο - συγκεκριμένο αριθμό θύρας, όπως για παράδειγμα είναι η τιμή `111` για portmapper, `23` για telnet ή `80` για http.

Όταν δηλώνουμε μία ομάδα θυρών μπορούμε να παραλείψουμε το ένα άκρο του ορισμού δηλώνοντας, κατ' αυτόν τον τρόπο, ότι ενδιαφερόμαστε για όσες

διευθύνσεις είναι μικρότερες ή μεγαλύτερες από μία άλλη. Αναλυτικά, μια έκφραση της μορφής "`500:`" ισχύει για θύρες μεγαλύτερες ή ίσες με `500`, ενώ σε αντίθετη περίπτωση η μορφή "`:1024`" ισχύει για θύρες μικρότερες ή ίσες με `1024`.

```
log tcp any :1024 -> 192.168.1.0/24 500:
```

ΚΑΤΕΥΘΥΝΣΗ

Η κατεύθυνση της κυκλοφορίας ενός δικτύου υποδεικνύεται με τον χαρακτήρα του δεξιού βέλους (`->`). Ο συγκεκριμένος τελεστής βρίσκεται ανάμεσα στα δύο ζεύγη διεύθυνσης IP και θύρας, συμβολίζοντας την κατεύθυνση της κίνησης που μας ενδιαφέρει. Οτιδήποτε εμφανίζεται στα αριστερά του βέλους είναι η *προέλευση*, ενώ οτιδήποτε υπάρχει στα δεξιά του βέλους είναι ο *προορισμός*. Επιπλέον, υπάρχει και ένας ενδείκτης κατεύθυνσης που ονομάζεται αμφίδρομος τελεστής (`<>`), ο οποίος «υποδεικνύει» ότι θα παρακολουθείται οποιοσδήποτε συνδυασμός προέλευσης/προορισμού. Σε αυτή την περίπτωση, το Snort μπορεί να παρακολουθεί την συνομιλία που διαμείβεται μεταξύ δύο υπολογιστών και σε συνδυασμό με την σχολαστική και σε βάθος ανάλυση των δεδομένων των αρχείων καταγραφής που παράγονται, αυξάνουμε στο μέγιστο βαθμό την πιθανότητα εντοπισμού κακόβουλων επιθέσεων.

Παρακάτω, παρουσιάζεται ένα παράδειγμα χρήσης ενός αμφίδρομου τελεστή κατεύθυνσης (*bidirectional operator*), καταγράφοντας και τις δύο πλευρές μίας telnet επικοινωνίας.

```
log tcp !192.168.1.0/24 any <> 192.168.1.0/24 23
```

ΕΠΙΛΟΓΕΣ ΚΑΝΟΝΩΝ

Η γρήγορη διάδοση του Snort, οφείλεται σε μεγάλο βαθμό στις πολλές και διαφορετικές επιλογές/ρυθμίσεις που διαθέτει. Οι επιλογές των κανόνων αποτελούν την «καρδιά» του μηχανισμού ανίχνευσης εισβολών του Snort, συνδυάζοντας την ευκολία στη χρήση, τη «διορατικότητα» και την ευελιξία. Όλες οι ρυθμίσεις σε έναν κανόνα διαχωρίζονται μεταξύ τους με τον χαρακτήρα του ερωτηματικού ";", ενώ οι λέξεις-κλειδιά (*keywords*) των επιλογών του κανόνα, διαχωρίζονται από τα ορίσματα

τους μέσω του χαρακτήρα της άνω-κάτω τελείας ":". Υπάρχουν τέσσερις κύριες κατηγορίες των επιλογών ενός κανόνα (*rule options*).

- **Γενικές**

Παρέχουν πληροφορίες σχετικά με τον εκάστοτε κανόνα, αλλά χωρίς να έχουν την οποιαδήποτε επίδραση κατά τη διάρκεια της ανίχνευσης.

- **Περιεχομένου**

Οι επιλογές του συγκεκριμένου τύπου, καθορίζουν όλα τα στοιχεία - δεδομένα που θα ανιχνευθούν στο εσωτερικό ενός πακέτου και μπορούν να αλληλοεξαρτώνται μεταξύ τους.

- **Μη-περιεχομένου**

Εξετάζουν τις «πληροφορίες» που δεν βρίσκονται στο περιεχόμενο ενός πακέτου.

- **Ολοκληρωμένης Ανίχνευσης**

Προσδιορίζουν την εκτέλεση συγκεκριμένων γεγονότων που συμβαίνουν μόλις επαληθευτεί ένας κανόνας.

Οι παραπάνω επιλογές, μπορούν να *συνδυάζονται* και να *προσαρμόζονται* έτσι ώστε να μπορούν να χειρίζονται όλους τους τύπους κυκλοφορίας όπως ορίζονται από τις κεφαλίδες των κανόνων. Ο ακόλουθος πίνακας, παρουσιάζει ορισμένες από τις προκαθορισμένες επιλογές του Snort.

ΔΕΣΜΕΥΜΕΝΗ ΛΕΞΗ	ΠΕΡΙΓΡΑΦΗ
`msg`	Ορίζει το κείμενο του προειδοποιητικού μηνύματος για έναν συναγερμό (*alert*).
`content`	Ορίζει το «μοτίβο» που θα αναζητά το Snort μέσα στο πακέτο.
`uricontent`	Καθορίζει το μοτίβο URI (Uniform Resource Identifier, ενιαίο αναγνωριστικό εντοπισμού πόρων) που θα αναζητά το Snort.
`ttl`	Εξετάζει την τιμή του IP πακέτου στο πεδίο TTL (*time-to-live*).
`stateless`	Ανεξαρτήτως της κατάστασης, καθορίζει ότι το πακέτο θα ελέγχεται έναντι του κανόνα.
`logto`	Προσδιορίζει την θέση αποθήκευσης ενός εναλλακτικού αρχείου καταγραφής, (π.χ. `logto:"filename";`).
`session`	Καταγράφει τις πληροφορίες *επιπέδου εφαρμογής* που περιλαμβάνει ένα πακέτο.

Πίνακας 4: **Rule Options**

Μία από τις σημαντικότερες και χρησιμότερες επιλογές των κανόνων του Snort, είναι η προαναφερθείσα «λέξη-κλειδί» **content**. Η συγκεκριμένη επιλογή, αποτελεί ένα από τα βασικότερα χαρακτηριστικά του Snort, καθώς επιτρέπει στον εκάστοτε χρήστη να ορίζει κανόνες οι οποίοι αναζητούν - ψάχνουν για οποιοδήποτε συγκεκριμένο αλφαριθμητικό/ακολουθία χαρακτήρων στο εσωτερικό των πακέτων. Ένας κανόνας περιέχοντας τη δεσμευμένη λέξη *content*, ελέγχει - ταιριάζει αρχικά τα δεδομένα του πακέτου με τον κατάλληλο αλγόριθμο και σε περίπτωση που βρεθεί η «επιθυμητή» αντιστοιχία με το περιεχόμενο του, εκτελούνται οι υπόλοιπες παράμετροι καθώς και ο έλεγχος διαχωρισμού μεταξύ κεφαλαίων και μικρών γραμμάτων.

Το όρισμα της παραπάνω ρύθμισης, δίνεται με τη μορφή απλής συμβολοσειράς, με δυαδικά δεδομένα ως δεκαεξαδικοί αριθμοί (*bytecode*) ή με συνδυασμό και των δύο.

- **Μορφή**

```
content:[!]"<content string>";
```

◆ **Παραδείγματα content**

```
1. alert tcp any any -> any 139 (content:"|5c 00|P|00|I|00|P|00|E|00  5c|";)
2. alert tcp any any -> any 80 (content:!"GET";)
```

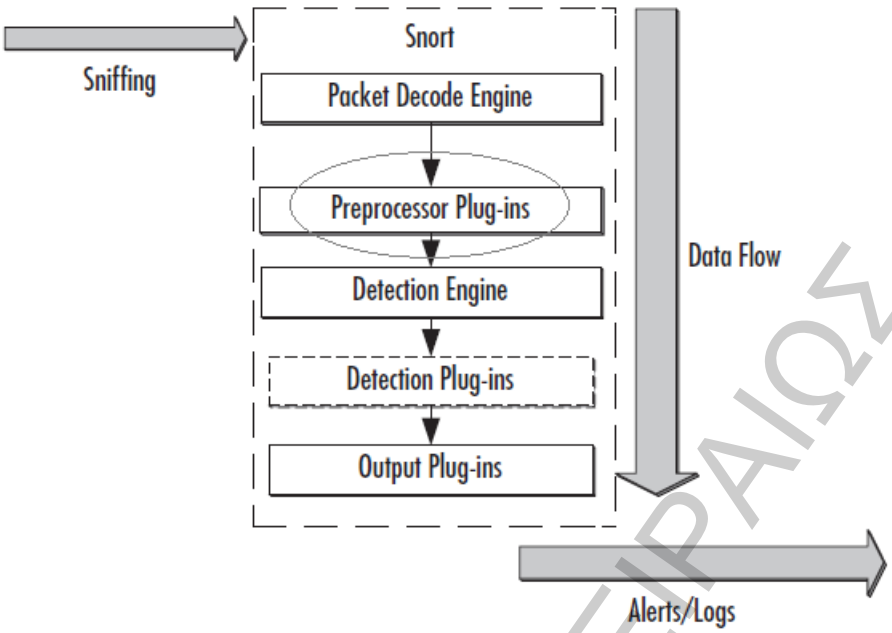
Στα ανωτέρω παραδείγματα, παρουσιάζεται η χρήση του μικτού κειμένου και των δεκαεξαδικών δεδομένων σε έναν κανόνα Snort, καθώς και ο τελεστής άρνησης πριν από το όρισμα του content, γεγονός που σημαίνει αυτομάτως ότι ο κανόνας θα ισχύει όταν δεν θα υπάρχει το συγκεκριμένο περιεχόμενο μέσα στο πακέτο. Συγκεκριμένα, όπως εμφανίζεται και στο πρώτο παράδειγμα, το κομμάτι που αντιστοιχεί σε δεκαεξαδική μορφή ορίζεται στην αρχή και στο τέλος με τον χαρακτήρα " | ", ενώ το εσωτερικό κάθε χαρακτήρα γράφεται με δύο δεκαεξαδικά ψηφία, διαχωρίζοντας τα από τον επόμενο με ένα κενό. Επιπλέον, μπορούν να «συνυπάρχουν» στον ίδιο κανόνα περισσότερες από μία εντολές *content*.

Τέλος, αποτελεί άξιο αναφοράς το γεγονός πως τα πιο κοινά worms και επιθέσεις DDoS περιλαμβάνουν μία γνωστή, αλλά και σταθερή σειρά χαρακτήρων την οποία χρησιμοποιούν για να προκαλέσουν μία υπερχείλιση buffer (*buffer overflow*) ή για να ενεργοποιήσουν ένα κώδικα σε «μη-ασφαλείς» - εκτεθειμένους servers. Ελέγχοντας λοιπόν όλα τα πακέτα για αυτά τα γνωστά μοτίβα επιθέσεων, είμαστε σε θέση να γνωρίζουμε έγκαιρα εάν δεχόμαστε επίθεση από τα πιο κοινά εργαλεία των παραπάνω κατηγοριών. Η εντολή *content*, μπορεί να συνδυάσει μία ή περισσότερες εντολές τροποποίησης οι οποίες επηρεάζουν τον τρόπο με τον οποίο θα γίνει η αναζήτηση κάθε περιεχομένου.

7.6 Προεπεξεργαστές

Οι προεπεξεργαοτές (*preprocessors*), είναι το επόμενο στάδιο εξέλιξης μετά από την χρήση προσαρμοσμένων επιλογών κανόνων. Εισήχθησαν για πρώτη φορά στην έκδοση 1.5 του Snort και ουσιαστικά είναι πρόσθετα *plug-ins* τα οποία δίνουν στους χρήστες τη δυνατότητα να ορίζουν εξειδικευμένα μοτίβα επιθέσεων και τρόπους αντίδρασης. Τα *plug-ins* είναι κομμάτια κώδικα, με τη λογική ξεχωριστών υποπρογραμμάτων που ενσωματώνονται στον υπόλοιπο κώδικα του Snort, επιτρέποντας με αυτό τον τρόπο, την επέκταση των δυνατοτήτων του χωρίς να χρειάζεται καμία περαιτέρω αλλαγή στο «κύριο σώμα» του κώδικά του. Επιπλέον, οι εκάστοτε χρήστες μπορούν να ενεργοποιούν αυτά τα πρόσθετα *plug-ins* στο αρχείο κανόνων για να εκμεταλλεύονται και να επεκτείνουν την λειτουργικότητα τους.

Όπως φαίνεται και στο ακόλουθο *Σχήμα 10*, ο κώδικας των προεπεξεργαστών εκτελείται μετά από την αποκωδικοποίηση του πακέτου (*packet decode engine*) και πριν από την εφαρμογή της μηχανής ανίχνευσης (*detection engine*), παρουσιάζοντας παράλληλα την διαδρομή που ακολουθεί κάθε πακέτο κατά την επεξεργασία του από το Snort. Οι προεπεξεργαστές, καλούνται προς εκτέλεση μία μόνο φορά για κάθε πακέτο και μέσω αυτών είναι δυνατόν να υλοποιηθούν εργασίες που έχουν να κάνουν είτε με την εξαγωγή διαφόρων στατιστικών που σχετίζονται και αφορούν τα πακέτα που επεξεργάζεται το Snort για ύποπτη δραστηριότητα, είτε με την κατάλληλη προετοιμασία τους πριν αυτά προωθηθούν στη μηχανή αναζήτησης. Στο σημείο αυτό πρέπει να σημειωθεί, ότι για μερικές θεμελιώδεις λειτουργίες της μηχανής αναζήτησης του Snort, οι προεπεξεργαστές είναι πλήρως απαραίτητοι.



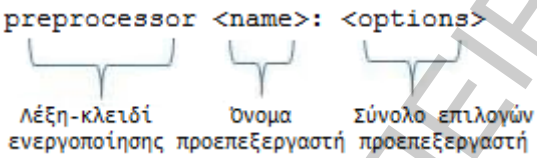
Σχήμα 10: Ροή Δεδομένων

Ως συμπληρωματική μηχανή και «προέκταση» της βασικής λειτουργίας του Snort, ορισμένα είδη επιθέσεων δεν θα μπορούσαν να εντοπιστούν - ανιχνευτούν από το εκάστοτε σύστημα χωρίς την επιπλέον επεξεργασία των προεπεξεργαστών. Αναλυτικότερα, οι προεπεξεργαστές συμβάλλουν στη βελτίωση του εντοπισμού κάποιων επιθέσεων που δεν έχουν γίνει ακόμη «κανόνες». Όπως το λέει και η ίδια η λέξη, ένα *plug-in* είναι ένα πρόσθετο πρόγραμμα, που έχει τη δυνατότητα να ενεργοποιείται και να απενεργοποιείται κατά βούληση από τον διαχειριστή (*administrator*) ενός συστήματος. Οι προεπεξεργαστές, αναπτύσσονται σαν *plug-ins*, δίνοντας στο Snort ευελιξία και επεκτασιμότητα. Η ικανότητα, λοιπόν, του Snort να δέχεται τέτοιου είδους προγράμματα δεν δίνει μονό μεγάλη «πιθανότητα» επέκτασης του ίδιου του Snort αλλά και την δυνατότητα προσαρμογής – ρύθμισης ανάλογα με τις ανάγκες του περιβάλλοντος δικτύου που θα χρησιμοποιηθεί. Για παράδειγμα, σε περίπτωση που δεν επιθυμούμε να ελέγχεται για ένα συγκεκριμένο χρονικό διάστημα το δίκτυο μας από σαρώσεις θυρών (*port scans*), μπορούμε να απενεργοποιήσουμε το εν λόγω *plug-in* χωρίς να επηρεαστεί καθόλου το υπόλοιπο σύστημα.

Διαμόρφωση προεπεξεργαστών

Οι προεπεξεργαστές (*preprocessors*), ενεργοποιούνται και φορτώνονται μέσω της δεσμευμένης λέξης *preprocessor*. Παράλληλα, οι παράμετροι που τους «αφορούν» ρυθμίζονται και διαμορφώνονται στο αρχείο snort config (*snort.conf*), επιλέγοντας ποιους προεπεξεργαστές και ποιες από τις υπολειτουργίες τους θέλουμε να ενεργοποιήσουμε.

Η γενική σύνταξη που καθιστά ενεργό έναν προεπεξεργαστή, αφού προηγουμένως γίνει compiled με το υπόλοιπο Snort, είναι η εξής:



Το Snort υποστηρίζει τη λειτουργία πολλών και διαφόρων προεπεξεργαστών, ενώ μπορούμε επίσης να δημιουργούμε και τους δικούς μας ανάλογα με τις εκάστοτε ανάγκες. Ακολουθούν, ορισμένα παραδείγματα δημοφιλών προεπεξεργαστών:

- **Stream5**

Ο προεπεξεργαστής Stream5 και «διάδοχος» του Stream4, παρέχει τη δυνατότητα ανασυγκρότησης μιας ροής δεδομένων TCP όσο και UDP με βάση τον αποδέκτη, απορρίπτοντας πακέτα από ροές που δεν έχουν δημιουργηθεί μέσω 3-way handshake. Είναι σε θέση, να παρακολουθεί και να ελέγχει για μεγάλους αριθμούς σε πακέτα TCP και UDP, επιθέσεις που δεν βασίζονται σε πληροφορίες κατάστασης και περιπτώσεις κατακερματισμού των πακέτων σε κομμάτια με μικρότερο από το κανονικό μέγεθος, τα οποία χρησιμοποιούνται συνήθως για να κρύψουν την δραστηριότητα ενός επιτιθέμενου.

Ρύθμιση Stream5 UDP

```
preprocessor stream5_udp: [timeout <number secs>], [ignore_any_rules]
```

- **HTTP Inspect**

Ο προεπεξεργαστής HTTP Inpect είναι υπεύθυνος για την ανίχνευση «παράξενης» κίνησης HTTP, μετατρέποντας τις διευθύνσεις URI από την μορφή Unicode σε μορφή ASCII, ώστε να μπορούν να διαβαστούν και να ερμηνευτούν σωστά από την μηχανή ανίχνευσης (*detection engine*). Με την συγκεκριμένη διαδικασία της κανονικοποίησης, το Snort δύναται - μπορεί να αναγνωρίσει μια ασαφή και μη ξεκάθαρη συλλογή χαρακτήρων.

Πρέπει να τονιστεί, ότι ένας HTTP Inspect είναι ένας γενικός αποκωδικοποιητής HTTP για εφαρμογές χρήστη, όπου χωρίς την χρήση του μία επίθεση είναι εύκολο να «μεταμφιεστεί» έτσι ώστε να μην ταιριάζει με τις υπάρχουσες υπογραφές ανίχνευσης και ο εκάστοτε web server να την θεωρήσει ως ένα έγκυρο URL.

- **sfPortscan**

Η λειτουργική μονάδα sfPortscan, έχει αναπτυχθεί από την Sourcefire και είναι σχεδιασμένη για να ανιχνεύει τις απόπειρες «αναγνώρισης» δικτύων από crackers και συγκεκριμένα την πρώτη φάση σε μια δικτυακή επίθεση.

Στην φάση της αναγνώρισης, ένας εισβολέας προσπαθεί να καθορίσει ποιοί τύποι δικτυακών πρωτοκόλλων ή υπηρεσίες υποστηρίζονται από έναν host. Η ανωτέρω φάση, «υποθέτει» ότι ο επιτιθέμενος δεν έχει προηγούμενη γνώση για το τι πρωτόκολλα ή υπηρεσίες υποστηρίζονται από τον στόχο του, με αποτέλεσμα τα περισσότερα ερωτήματα (*queries*) που στέλνει να του έρχονται με αρνητική απάντηση. Αναλυτικά, όταν πρόκειται για μία νόμιμη δικτυακή επικοινωνία, οι αρνητικές απαντήσεις από τους hosts είναι πολύ σπάνιο φαινόμενο, πόσο μάλλον όταν υπάρχουν πολλαπλές αρνητικές απαντήσεις μέσα σε ένα δεδομένο χρονικό διάστημα. Πρωταρχικός στόχος, λοιπόν, για την ανίχνευση του sfPortscan είναι να ανιχνεύουν αυτές οι αρνητικές απαντήσεις.

Ένα από τα πιο κοινά και γνωστότερα εργαλεία σάρωσης θυρών (*portscanning*) που χρησιμοποιείται στις μέρες μας, είναι το Nmap. Το Nmap περιλαμβάνει και υποστηρίζει πολλές, αν όχι όλες, τις τρέχουσες - γνωστές τεχνικές portscanning. Κατά συνέπεια, το sfPortscan έχει σχεδιαστεί με τέτοιο τρόπο ώστε να είναι σε θέση να

ανιχνεύει τους διαφορετικούς τύπους και τεχνικές των σαρώσεων που παράγει το Nmap.

Το sfPortscan, είναι «συμβατό» με τους ακόλουθους τύπους σαρώσεων Nmap:

- ✓ TCP portscan
- ✓ UDP portscan
- ✓ IP portscan

◆ **RPC Decode**

Ο σκοπός του προεπεξεργαστή RPC decode είναι να κανονικοποιήσει - «ομαλοποιήσει» τις πολλαπλές κατακερματισμένες εγγραφές σε μια ενιαία και μεμονωμένη εγγραφή, ώστε να είναι δυνατή σε κάθε περίπτωση η αναγνώριση οποιασδήποτε υπογραφής (π.χ. *κακόβουλης εγγραφής*) από την μηχανή ανίχνευσης. Συγκεκριμένα, αυτό επιτυγχάνεται με την *κανονικοποίηση* του πακέτου στην ενδιάμεση, προσωρινή, μνήμη του.

Για παράδειγμα, στην περίπτωση που ο προεπεξεργαστής Stream5 είναι ενεργοποιημένος, θα επεξεργάζεται μόνο την κίνηση από πλευράς client. Από προεπιλογή, ο προεπεξεργαστής RPC decode τρέχει στις θύρες `111` και `32771`. Για να ενεργοποιηθεί ο προεπεξεργαστής rpc_decode θα πρέπει να γραφτεί η παρακάτω εντολή στο Snort.conf:

Μορφή

```
preprocessor rpc_decode: \
    <ports> [ alert_fragments ] \
    [no_alert_multiple_requests] \
    [no_alert_large_fragments] \
    [no_alert_incomplete]
```

Επιλογή	Περιγραφή
`alert_fragments`	Ειδοποίηση για οποιαδήποτε κατακερματισμένη RPC εγγραφή.
`no_alert_multiple_requests`	Δεν ειδοποιεί, όταν υπάρχουν πολλαπλές εγγραφές σε ένα πακέτο.
`no_alert_large_fragments`	Δεν ειδοποιεί, όταν το άθροισμα των κατακερματισμένων εγγραφών υπερβαίνει το ένα πακέτο.
`no_alert_incomplete`	Δεν ειδοποιεί, όταν μία απλή εγγραφή ενιαίου τμήματος υπερβαίνει το μέγεθος ενός πακέτου.

Πίνακας 5: **Ανάλυση περιεχομένου του προεπεξεργαστή RPC Decode**

Ένας από τους πιο γνωστούς, ξεχωριστούς και ευρέως διαδεδομένους προεπεξεργαστές, είναι ο HTTP Inspect. Πολλές περιπτώσεις που βασίζονται στο *pattern matching* δεν θα μπορούσαν να υλοποιηθούν χωρίς την δυνατότητα κανονικοποίησης που δίνεται από τον προεπεξεργαστή HTTP Inspect. Όπως ήδη έχει αναφερθεί, αυτός ο ισχυρός προεπεξεργαστής δέχεται οποιοδήποτε αναγνωριστικό URI και το μετατρέπει σε στάνταρ ASCII μορφή πριν «διαβαστεί» από το Snort. Η συγκεκριμένη διαδικασία, είναι πολύ σημαντική για τον λόγο ότι οι εισβολείς χρησιμοποιούν διάφορες τεχνικές μορφοποίησης για να κρύψουν τις αιτήσεις που κάνουν για τις ισχυρές εντολές ενός συστήματος, οι οποίες βρίσκονται εκτός της κανονικής δομής καταλόγων ενός Web server. Πρέπει να σημειωθεί, ότι η κωδικοποίηση των δεδομένων HTTP, είναι μία από τις πιο γνωστές μεθόδους που χρησιμοποιούν οι crackers για να κρύψουν μια επίθεση από ένα NIDS.

Συνεπώς, επειδή στην αρχική διαμόρφωση του Snort οι περισσότεροι κανόνες βασίζονται στην μορφή ASCII για την αναγνώριση αυτών των ασυνήθιστων και μη επιτρεπόμενων αιτήσεων, όταν οι αιτήσεις αυτές είναι σε μορφή Unicode αγνοούνται.

Ακολουθεί, ένα παράδειγμα σύνταξης του HTTP Inspect και της σύνταξης των προεπεξεργαστών γενικότερα:

```
preprocessor http_inspect: \
    global \
    iis_unicode_map <map_filename> \
    codemap <integer> \
    [detect_anomalous_servers] \
    [proxy_alert] \
    [max_gzip_mem <num>] \
    [compress_depth <num>] [decompress_depth <num>] \
    [memcap <num>] \
    disabled
```

7.7 Οδηγίες Μορφοποίησης

Οι λειτουργικές μονάδες μορφοποίησης (*formatting modules*), ή όπως συνηθέστερα αποκαλούνται και περιγράφονται ως λειτουργικές μονάδες εξόδου (*output modules*), μας επιτρέπουν να καθορίσουμε την μορφή και την παρουσίαση των μηνυμάτων προειδοποίησης και καταχωρίσεων στα αρχεία καταγραφής (*log files*). Συγκεκριμένα, τα *output modules/plug-ins* αυξάνουν την λειτουργικότητα του Snort, εκτελώντας εργασίες οι οποίες έχουν να κάνουν με το *alerting* και το *logging*. Για παράδειγμα, όταν τα δεδομένα που έχουν ήδη περάσει από τους προεπεξεργαστές και την μηχανή ανίχνευσης ταιριάζουν με κάποιον κανόνα, τότε αυτομάτως εκδίδεται μια ειδοποίηση (*alert*) για το συγκεκριμένο συμβάν. Πρέπει να σημειωθεί, πως η μορφή των οδηγιών στο αρχείο config είναι παρόμοια με εκείνη ενός προεπεξεργαστή. Το Snort παρέχει πολλούς και διαφορετικούς τρόπους για την αποτύπωση των δεδομένων, καθώς υπάρχουν επιπρόσθετα προγράμματα που έχουν γραφτεί για να τον συγκεκριμένο σκοπό. Στην ουσία, ένα *output plug-in*, δίνει τη δυνατότητα στο Snort να προβάλλει και να μεταφέρει τις ειδοποιήσεις σε διάφορες μορφές και μηχανισμούς, όπως είναι για παράδειγμα η αποστολή των εκάστοτε «μηνυμάτων» στο **syslog**.

Επίσης, πρέπει να αναφερθεί ότι υπάρχει σχετικό template μέσα στο πακέτο του Snort για οποιονδήποτε χρήστη θέλει να γράψει τη δική του «αποτύπωση» εξόδου. Το Snort, δεν μας περιορίζει στη χρησιμοποίηση ενός και μόνου output module, αλλά

μπορούμε να ενεργοποιήσουμε μία διαδικασία η οποία αποκαλείται συσσώρευση οδηγιών, ορίζοντας πολλαπλές «οδηγίες» μορφοποίησης με σειριακό τρόπο στο αρχείο ρυθμίσεων του. Οι σημαντικότερες και πιο συχνά χρησιμοποιούμενες «οδηγίες» μορφοποίησης, είναι οι ακόλουθες:

`alert_syslog`	Καθορίζει πώς θα χειρίζεται τους «συναγερμούς» το syslog , δίνοντας στους χρήστες μεγαλύτερη ευελιξία στην καταγραφή των ειδοποιήσεων (*logging alerts*).
`alert_fast`	Εμφανίζει ένα «απλό» προειδοποιητικό μήνυμα μιας γραμμής σε ένα καθορισμένο αρχείο εξόδου. Πρόκειται για την ταχύτερη μέθοδο ειδοποίησης, καθώς καταγράφει ένα αρχείο κάθε φορά χωρίς να εμφανίζει τις κεφαλίδες των πακέτων στο αντίστοιχο αρχείο εξόδου.
`alert_full`	Εμφανίζει ένα πλήρες προειδοποιητικό μήνυμα με όλες τις πληροφορίες για το περιεχόμενο του κάθε πακέτου, συμπεριλαμβανομένων και των κεφαλίδων του.
`alert_unixsock`	«Δημιουργεί» μια επικοινωνία με ένα Unix domain socket το οποίο τρέχει στο ίδιο σύστημα, στέλνοντας σε αυτό όλα τα προειδοποιητικά μηνύματα.
`log_tcpdump`	Καταγράφει και μορφοποιεί τα προειδοποιητικά μηνύματα σε ένα tcpdump αρχείο σύμφωνα με την διαμόρφωση που χρησιμοποιεί το tcpdump. Χρησιμοποιείται συνήθως για περαιτέρω ανάλυση και επεξεργασία των πληροφοριών των πακέτων
`csv`	Στέλνει τα μηνύματα σε ένα αρχείο μορφής CSV (comma-separated values, τιμές διαχωρισμένες με κόμματα), διευκολύνοντας έτσι την εισαγωγή των των στοιχείων σε μια βάση δεδομένων.
`log_null`	Δυνατότητα δημιουργίας κανόνων που θα ειδοποιούν σε προκαθορισμένους τύπους δικτυακής κυκλοφορίας, χωρίς όμως να γίνονται καταχωρήσεις των «προκληθέντων» πακέτων στο αρχείο καταγραφής.

Το ακόλουθο παράδειγμα χρησιμοποιεί ένα από τα παραπάνω output modules:

```
output alert_syslog: log_auth log_alert
```

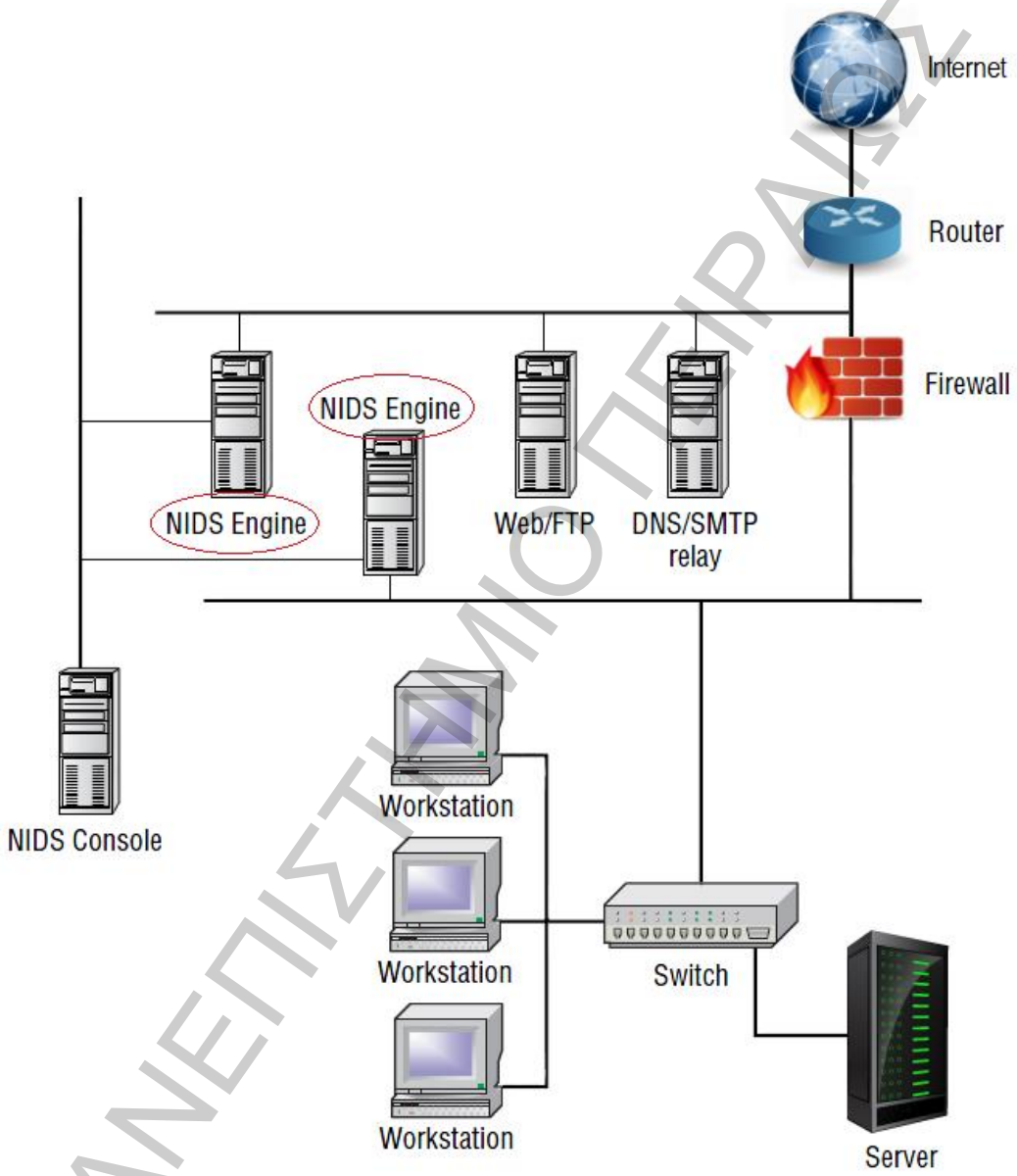
Στο συγκεκριμένο παράδειγμα, το `output` είναι η λέξη-κλειδί (*keyword*), το `alert_syslog` είναι η οδηγία μορφοποίησης (*<name>*) και τα `log_auth` και `log_alert` είναι τα συστατικά - επιλογές (*<options>*) του syslog. Με αυτόν τον τρόπο, μπορούμε να καθορίσουμε ότι τα προειδοποιητικά μηνύματα των συναγερμών όχι μόνο θα περνιούνται στο syslog, πράγμα το οποίο είναι η προεπιλεγμένη μας κατάσταση, αλλά επίσης «δείχνουμε» στο syslog πως θα χειρίζεται αυτά τα μηνύματα όταν βρεθούν υπό την επίβλεψη του.

7.8 Τοποθέτηση Αισθητήρων Ανίχνευσης Εισβολών

Ο βασικός σκοπός και «στόχος» των Συστημάτων Ανίχνευσης Εισβολών Δικτύου (*NIDS*), είναι η δημιουργία δικτύων με NIDS αισθητήρες (*NIDS sensors*) καθένας από τους οποίους θα είναι «υπεύθυνος» για ένα συγκεκριμένο κομμάτι μέσα στο δίκτυο. Έχοντας, λοιπόν, αναλύσει τη συνδεσμολογία του εκάστοτε δικτύου θα πρέπει να μελετηθούν με ακρίβεια και με «στρατηγικό» τρόπο τα διαφορετικά σημεία τοποθέτησης των NIDS αισθητήρων. Τα σημεία που χρήζουν προστασίας και εκτιμώνται - κρίνονται αναγκαία προς παρακολούθηση, είναι τα σημεία όπου διαφορετικά δικτυακά μέρη ενώνονται μεταξύ τους. Με αυτόν τον τρόπο, επιτυγχάνεται καλύτερη και αξιόπιστη ανίχνευση των επιθέσεων, καθώς παρακολουθείται η κίνηση από και προς όλες τις συσκευές του δικτύου και παράλληλα ελαχιστοποιούμε την απώλεια πακέτων που περνάνε από τον κάθε αισθητήρα - ανιχνευτή.

Συνεπώς, για να αποφασιστεί ποια είναι η καλύτερη θέση για ένα NIDS, θα πρέπει να προσδιορίσουμε ποια συστήματα θέλουμε να προστατευτούν και από ποιες «πηγές». Το συγκεκριμένο ζήτημα, θα πρέπει να αποσαφηνίζεται πολύ προσεκτικά και εμπεριστατωμένα, καθώς δεν είναι λίγες οι φορές που ένα σύστημα ανίχνευσης εισβολών σε δίκτυα χρειάζεται περισσότερους από έναν αισθητήρες NIDS. Συγκεντρωτικά, η επιλογή του σημείου που θα τοποθετηθεί ένας NIDS αισθητήρας σε ένα δίκτυο, πρέπει να γίνεται αφού ληφθούν υπόψιν διάφορα κριτήρια και καθοριστεί η «φύση» των αποτελεσμάτων που θέλουμε να παραχθούν από το

εκάστοτε NIDS. Συνήθως, οι πιθανές θέσεις ενός αισθητήρα NIDS είναι πίσω ή μπροστά από το firewall, ή σε θέσεις που θα παρακολουθεί ένα συγκεκριμένο υποδίκτυο. Ένας πιθανός τρόπος χρήσης, παρουσιάζεται στο *Σχήμα 11*.



Σχήμα 11: Χρήση ενός NIDS με δύο αισθητήρες

Στην παραπάνω σχηματική διαμόρφωση, παρακολουθείται τόσο η αποστρατικοποιημένη ζώνη (*DMZ, demilitarized zone*), όσο και η σύνδεση του firewall στο εσωτερικό δίκτυο. Το ανωτέρω, μας επιτρέπει να ελέγχουμε όλη την εισερχόμενη κυκλοφορία από το Internet. Επιπλέον, μας επιτρέπει να «ενδυναμώσουμε» το υπάρχον firewall, καθώς και οι δύο αισθητήρες του NIDS τρέχουν χωρίς να έχει συσχετιστεί το πρωτόκολλο IP με τον δημόσιο κλάδο του δικτύου. Το πρωτόκολλο IP τρέχει μόνο σε μία κάρτα δικτύου η οποία συνδέει τους αισθητήρες με την κονσόλα (*console*). Με αυτόν τον τρόπο, οι αισθητήρες του NIDS είναι εντελώς «αόρατοι» και δεν αναγνωρίζονται από τα συστήματα που βρίσκονται στον δημόσιο κλάδο του δικτύου.

Ωστόσο, η συγκεκριμένη διαμόρφωση έχει ορισμένους περιορισμούς και αυτό διότι, δεν θα μπορεί να γίνεται πλήρης παρακολούθηση των επιθέσεων που προέρχονται από το Internet και έχουν σαν στόχο το firewall. Στην περίπτωση που θέλουμε να παρακολουθήσουμε μόνο την σχετιζόμενη με το Internet κυκλοφορία, πρέπει να τοποθετηθεί ένας «ισχυρός» server, τρέχοντας όλες τις λειτουργίες του NIDS «έξω» από το firewall. Επειδή, λοιπόν, το πρωτόκολλο IP δεν είναι απαραίτητο στο σύστημα στο οποίο τρέχει το NIDS, το προαναφερόμενο σύστημα θα είναι ασφαλές από κάθε είδους επιθέσεις.

Ενας άλλος περιορισμός της διαμόρφωσης που παρουσιάζεται στο *Σχήμα 11*, είναι ότι δεν μας επιτρέπει να παρακολουθούμε οποιαδήποτε κυκλοφορία διακινείται μεταξύ δύο συγκεκριμένων σταθμών του εσωτερικού μας δικτύου. Εάν, ο στόχος μας είναι να παρακολουθούμε σε συνεχή βάση όλη την κυκλοφορία του δικτύου, θα πρέπει να μεταφερθεί ο αισθητήρας NIDS για το εσωτερικό δίκτυο σε ξεχωριστή θύρα του switch και να διαμορφωθεί ακολούθως σαν θύρα παρακολούθησης. Με τον συγκεκριμένο τρόπο, θα μας επιτρέπεται να βλέπουμε όλη την κυκλοφορία που διακινείται στο εκάστοτε firewall.

Ως συμπέρασμα των παραπάνω, για να «κλειδώσει» ιδανικά ένα δίκτυο στο μέγιστο δυνατό βαθμό, θα πρέπει να συνδυαστούν οι ακόλουθες λύσεις:

- ✓ Τοποθέτηση ενός αισθητήρα NIDS «έξω» από το firewall.
- ✓ Τοποθέτηση ενός δεύτερου αισθητήρα στην θύρα παρακολούθησης του switch.
- ✓ Διαμόρφωση των δύο ανωτέρω αισθητήρων, ώστε να επικοινωνούν με την κονσόλα (*console*) μέσω ενός ιδιωτικού υποδικτύου.

Η συγκεκριμένη «προσέγγιση», μας επιτρέπει να παρακολουθούμε όλη την εσωτερική κυκλοφορία ενός δικτύου, διατηρώντας τον έλεγχο από μία *κεντρική κονσόλα* και αποκτώντας παράλληλα πλεονεκτήματα - οφέλη όπως η καταγραφή των πακέτων σε «ακατέργαστη» μορφή και η δυναμική τροποποίηση των κανόνων των φίλτρων. Κατά συνέπεια, αφού καθοριστούν οι περιοχές παρακολούθησης και ανίχνευσης, μπορεί να επιλεγεί εν συνεχεία ο αριθμός των απαραίτητων αισθητήρων για το NIDS, καθώς επίσης και ο κατάλληλος υλικός εξοπλισμός.

Κλείνοντας, θα πρέπει να αναφερθεί ότι πλέον τα περισσότερα δίκτυα αν όχι όλα, βασίζουν και στηρίζουν όλη την υποδομή τους σε switches έναντι των hubs που χρησιμοποιούνταν παλαιότερα. Η αλλαγή αυτή φέρει ως αποτέλεσμα την αισθητή αύξηση της ταχύτητας και της αποτελεσματικότητας σε ένα δίκτυο. Πολλοί ειδικοί στο χώρο της Ασφάλειας Συστημάτων, αποδίδουν στα switches τον χαρακτηρισμό ”έξυπνα hubs”, καθώς όταν ένα πακέτο διέρχεται μέσω ενός switch, αναλύεται ώστε να καθοριστεί μέσω της MAC address σε ποιον υπολογιστή - παραλήπτη θα σταλθεί. Αντίθετα, κάθε πακέτο δεδομένων που περνάει από ένα hub αποστέλλεται σε όλα τα ports που είναι «συνδεδεμένα» πάνω του, με αποτέλεσμα να δημιουργείται μεγάλη σπατάλη πόρων κατά τη μετάδοση του.

7.9 Διαμόρφωση και Ρύθμιση Παραμέτρων του Snort

Μία από τις πρώτες και καθοριστικές εργασίες μετά την εγκατάσταση ή την ενημέρωση του Snort, είναι η διαμόρφωση των μεταβλητών που θα «αντιπροσωπεύουν» το εκάστοτε δίκτυο και τις διευθύνσεις IP που μπορεί να θέλουμε να εξαιρέσουμε από τους ελέγχους μας. Παραδείγματα τέτοιων μεταβλητών είναι τα RULES_PATH, MY_NET, MY_PORTS και DNS. Οι τιμές των ανωτέρω μεταβλητών, μπορούν να είναι μία λίστα δικτύων ή απλώς η δεσμευμένη λέξη *any*. Για παράδειγμα, τοποθετώντας τις διευθύνσεις IP όλων των DNS servers στην μεταβλητή DNS, μπορούμε να αποφύγουμε μία σημαντική αιτία ψεύτικων συναγερμών. Το Snort, ορίζει τους ακόλουθους τρεις τύπους μεταβλητών:

- var
- portvar
- ipvar

Θα πρέπει να σημειωθεί, ότι ο τύπος '*ipvar*' ενεργοποιείται μόνο όταν υποστηρίζεται το πρωτόκολλο IPv6, σε αντίθετη περίπτωση χρησιμοποιείται ο τύπος '*var*'. Το ακόλουθο παράδειγμα παρουσιάζει διάφορες βασικές μεταβλητές, ορίζοντας τους τύπους *var*, *portvar* και *ipvar* που ήδη προαναφέρθηκαν.

```
var RULES_PATH rules/
portvar MY_PORTS [22,80,1024:1050]
ipvar MY_NET [192.168.1.0/24,10.1.1.0/24]
alert tcp any any -> $MY_NET $MY_PORTS (flags:S; msg:"SYN packet";)
include $RULE_PATH/example.rules
```

Αρχικά, βλέπουμε τη γραμμή όπου ορίζεται το «μονοπάτι» των κανόνων και ακολούθως δηλώνονται οι θύρες και η IP διεύθυνση του τοπικού μας δικτύου σε μορφή CIDR, "`ipvar MY_NET [192.168.1.0/24,10.1.1.0/24]`". Επίσης, μπορούμε να αλλάξουμε τη ρύθμιση της μεταβλητής MY_NET σε "`ipvar MY_NET any`".

Τροποποίηση Μεταβλητών

ΣΥΝΤΑΞΗ ΜΕΤΑΒΛΗΤΗΣ	ΠΕΡΙΓΡΑΦΗ
`var`	Ορίζει μια μεταβλητή.
`$(var) ή $var`	Αντικαθιστά με τα περιεχόμενα της μεταβλητής var.
`$(var:-default)`	Σε περίπτωση που είναι απροσδιόριστη η μεταβλητή var, τα περιεχόμενα της αντικαθιστώνται βάσει «προεπιλογής».
`$(var:?message)`	Αντικαθιστά με το περιεχόμενο της μεταβλητής var ή εκτυπώνει το εκάστοτε μήνυμα σφάλματος.

Πίνακας 6: Διαμόρφωση μεταβλητών του Snort

Όπως περιγράφεται αναλυτικά και στον παραπάνω πίνακα, τα ονόματα των μεταβλητών μπορούν να τροποποιηθούν με διάφορους τρόπους. Το παράδειγμα που ακολουθεί, αποτελεί μία τροποποιημένη μεταβλητή κάνοντας χρήση των ανωτέρω τελεστών:

```
ipvar MY_NET 192.168.1.0/24
log tcp any any -> $(MY_NET:?MY_NET is undefined!) 23
```

Ένα από τα κρισιμότερα και καίρια κομμάτια της διαμόρφωσης του Snort, είναι η εισαγωγή και η ρύθμιση των προεπεξεργαστών (*preprocessors*). Οι προεπεξεργαστές είναι ένα θέμα που απασχολούσε σχεδόν πάντα τους εκάστοτε χρήστες, καθώς είναι οι κύριοι «υπεύθυνοι» για την απόδοση και τη λήψη των συναγερμών του Snort. Όσο περισσότεροι προεπεξεργαστές ενεργοποιηθούν, τόσο περισσότεροι συναγερμοί θα λαμβάνονται. Οι ακόλουθοι προεπεξεργαστές, ανιχνεύουν τον «ασυνήθιστο» κατακερματισμό των πακέτων, επιτρέπουν την καλύτερη παρακολούθηση των συνδέσεων, εκτελούν τις απαιτούμενες μετατροπές από Unicode σε ASCII και ψάχνουν για απόπειρες σάρωσης θυρών, με τη σάρωση θυρών να ορίζεται σαν τρεις διαδοχικές αιτήσεις που γίνονται μέσα σε ένα χρονικό διάστημα λίγων δευτερολέπτων. Επιπλέον, πρέπει να τονίσουμε ότι η μονάδα για το Unicode είναι μία νεότερη έκδοση του HTTP Inspect.

```
preprocessor frag3
preprocessor stream5: detect_scans detect_state_problems
preprocessor http_inspect: 80 8080
```

```
preprocessor sfportscan: 0.0.0.0/0 3 4 /var/log/snort/sfportscan.log
```

Στη συνέχεια, αφού οριστούν οι προεπεξεργαστές, ακολουθεί το «κομμάτι» της δημιουργίας ενός *output module* για την αποστολή των προειδοποιητικών μηνυμάτων/συναγερμών σε μία βάση δεδομένων:

```
output database: alert, mysql, user=vroussos password=rousso$02
dbname=snort host=localhost
```

Καθορισμός των κανόνων

Σε αυτό το σημείο, θα πρέπει να επισημανθεί και να αναπτυχθεί η «ουσία» του Snort που δεν είναι τίποτα περισσότερο από τον καθορισμό των κανόνων. Στην πραγματικότητα, ο καθορισμός των κανόνων είναι μία σειρά από εντολές *include* οι οποίες «δείχνουν» σε έναν κατάλογο στον οποίο περιέχονται τα διάφορα αρχεία κειμένου που απαρτίζουν κάθε κανόνα. Το Snort, παρέχει και διατίθεται με μία εκτενή συλλογή χρήσιμων κανόνων για την ανίχνευση των συνηθέστερων μορφών επίθεσης σε δίκτυα:

```
include /etc/snort/exploit.rules
include /etc/snort/scan.rules
include /etc/snort/backdoor.rules
include /etc/snort/ftp.rules
include /etc/snort/virus.rules
include /etc/snort/blacklist.rules
include /etc/snort/ddos.rules
include /etc/snort/icmp-info.rules
include /etc/snort/telnet.rules
include /etc/snort/dos.rules
include /etc/snort/finger.rules
include /etc/snort/voip.rules
include /etc/snort/mysql.rules
include /etc/snort/dns.rules
include /etc/snort/netbios.rules
include /etc/snort/policy.rules
```

Μετά τη διαμόρφωση των κανόνων, θα πρέπει να γίνει ο προσδιορισμός των συστημάτων επικοινωνίας (*interfaces*) που θέλουμε να ελέγχουμε για τυχόν επιθέσεις. Επίσης, το Snort παρέχει τη δυνατότητα τροποποίησης του αρχείου προσδιορισμού του παραλήπτη - αποδέκτη, προσθέτοντας μία ή περισσότερες διευθύνσεις e-mail οι οποίες θα λαμβάνουν τα προειδοποιητικά μηνύματα των συναγερμών του Snort. Ο έλεγχος των εκάστοτε συναγερμών (*alerts*), γίνεται συνήθως από το αρχείο `/var/log/messages`, την βάση δεδομένων ή την θυρίδα εισερχομένων μηνυμάτων (*inbox*).

7.9.1 Παράδειγμα Συναγερμού

Ένας από τους πιο κοινούς και συνηθισμένους συναγερμούς που «παράγει» το Snort, είναι αυτός που ενεργοποιείται όταν εντοπίζεται μία απόπειρα σάρωσης θυρών. Η σάρωση θυρών (*portscan*), είναι απλώς ένας γρήγορος τρόπος ανίχνευσης των εκάστοτε συστημάτων ενός δικτύου από κάποια άλλη οντότητα μέσω του Internet, τυπικά είναι μία απόπειρα εύρεσης των υπηρεσιών που τρέχουν σε έναν υπολογιστή. Αν και η συχνή εμφάνιση συναγερμών θα μπορούσε να σημαίνει ότι ένας χρήστης είναι «στόχος» προσπαθειών ανίχνευσης ενός εισβολέα, στην πραγματικότητα θα μπορούσε επίσης να υποδεικνύει και μία ακίνδυνη δραστηριότητα. Ακολουθούν ορισμένα από τα μηνύματα προειδοποίησης που παράγει το Snort καθώς ανιχνεύει μία προσπάθεια σάρωσης θυρών:

```
Oct 25 19:35:58 snort[16227]: SCAN synscan portscan:
 102.17.46.3:7543 -> 12.16.1.101:21
Oct 25 19:35:58 snort[16227]: SCAN synscan portscan:
 102.17.46.3:24500 -> 12.16.1.102:21
Oct 25 19:35:58 snort[16227]: SCAN synscan portscan:
 102.17.46.3:26344 -> 12.16.1.103:21
Oct 25 19:35:58 snort[16227]: SCAN synscan portscan:
 102.17.46.3:2556 -> 12.16.1.104:21
Oct 25 19:35:58 snort[16227]: SCAN synscan portscan:
 102.17.46.3:8745 -> 12.16.1.105:21
Oct 25 19:35:58 snort[16227]: SCAN synscan portscan:
 102.17.46.3:33 -> 12.16.1.106:21
Oct 25 19:35:58 snort[16227]: SCAN synscan portscan:
 102.17.46.3:5555 -> 12.16.1.107:21
```

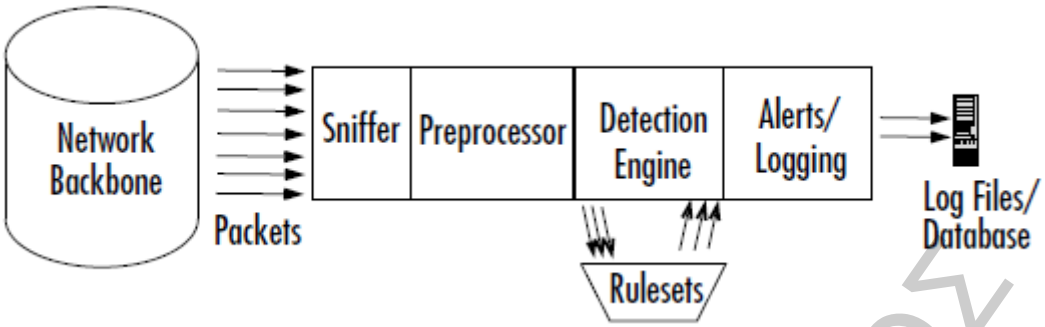
```
Oct 25 19:35:58 snort[16227]: SCAN synscan portscan:
 102.17.46.3:21 -> 12.16.1.108:21
```

Ένα στοιχείο το οποίο θεωρείται εξαιρετικά θετικό και χρήσιμο στο προηγούμενο παράδειγμα, είναι ότι το Snort κατάφερε να εντοπίσει μία αρκετά «έξυπνη» επίθεση, η οποία θα περνούσε απαρατήρητη μόλις λίγα χρόνια πριν. Σε αυτή την επίθεση ο εισβολέας αλλάζει την θύρα του μετά από κάθε μεμονωμένο πακέτο. Επιπρόσθετα, παρατηρείται ότι στην πραγματικότητα διερευνάται μία θύρα την φορά, σε μία ευρεία γκάμα διευθύνσεων IP. Συγκεκριμένα, αυτού του είδους η προσπάθεια παράκαμψης «απευθύνεται» σε ένα όχι ιδιαίτερα εξελιγμένο - προηγμένο NIDS, το οποίο μπορεί να μην αντιληφθεί ότι όλες αυτές οι αιτήσεις είναι μέρος της ίδιας προσπάθειας ανίχνευσης, κάτι το οποίο γίνεται άμεσα εμφανές και αντιληπτό όταν οι πληροφορίες συγκεντρώνονται και παρουσιάζονται με τον «τρόπο» του Snort.

8 Σχεδίαση και Υλοποίηση ενός συστήματος NIDS

Στο συγκεκριμένο κεφάλαιο, θα εξετάσουμε και θα παρουσιάσουμε αναλυτικά *τι είναι* ένα Σύστημα Ανίχνευσης Εισβολών (*Intrusion Detection System*), καθώς και πως μπορούμε να παρακολουθούμε σε συνεχή βάση το δικό μας δίκτυο από κακόβουλες και μη ενέργειες, διαχειρίζοντας τα εκάστοτε αρχεία καταγραφής (*log files*). Για την διαδικασία του σχεδιασμού και της υλοποίησης ενός Συστήματος Ανίχνευσης Δικτυακών Επιθέσεων (*NIDS*), θα ασχοληθούμε σε βάθος με το πολύ γνωστό και ευρέως χρησιμοποιούμενο **Snort**. Ένα έργο λογισμικού με ελεύθερα διαθέσιμο τον πηγαίο του κώδικα, το οποίο είναι από τα δημοφιλέστερα συστήματα NIDS που υπάρχουν σήμερα, συγκρινόμενο ακόμη και με τα αντίστοιχα πολυδάπανα εμπορικά προϊόντα. Στην πραγματικότητα, η ανίχνευση και η «αποφυγή» των επιθέσεων σε δίκτυα είναι μία συνεχής πρόκληση για οποιοδήποτε NIDS.

Καθώς, λοιπόν, το Snort είναι ένα από τα πλέον διαδεδομένα συστήματα ανίχνευσης εισβολών δικτύου, η ανάλυση της αρχιτεκτονικής δομής του μπορεί να μας δώσει σημαντικές πληροφορίες για το πως κατασκευάζεται ένα ασφαλές και αποδοτικό NIDS, ώστε να κατανοήσουμε πλήρως την λειτουργία και τον τρόπο που μας παρέχει τα αποτελέσματα του. Συγκεκριμένα, το Snort αποτελείται από επιμέρους τμήματα - συστατικά μέρη, όπως για παράδειγμα είναι οι προεπεξεργαστές (*preprocessors*), η μηχανή ανίχνευσης (*detection engine*) και τα πακέτα λογισμικού ειδοποίησης (*alert plug-ins*). Στο ακόλουθο σχήμα, παρουσιάζεται η δομή του Snort όσον αφορά τα υποσυστήματα από τα οποία αποτελείται, αναπαριστώντας την διαδρομή που ακολουθεί κάθε πακέτο δεδομένων κατά την επεξεργασία του από το Snort.



Σχήμα 12: Διαδρομή των πακέτων που ελέγχονται από το Snort

Στόχος του παρόντος κειμένου είναι η εγκατάσταση, η ρύθμιση και η διαμόρφωση του Snort σε έναν υπολογιστή του τοπικού μας δικτύου σύμφωνα - ανάλογα με τις πραγματικές μας «ανάγκες», όπου με την βοήθεια μιας βάσης δεδομένων όπως είναι η MySQL και με την χρήση ενός Web Analyzer όπως είναι το Base (*Basic Analysis and Security Engine*), να παρακολουθούμε ανά πάσα στιγμή το δίκτυο μας.

Ένα Σύστημα Ανίχνευσης Εισβολών (*IDS*), εντοπίζει κυρίως τους «χειρισμούς» ανεπιθύμητων ηλεκτρονικών συστημάτων. Αποτελείται από διάφορες συνιστώσες, όπως είναι οι αισθητήρες (*sensors*) και η κονσόλα (*console*) για την παρακολούθηση των παραγόμενων προειδοποιήσεων (*alerts*), διαθέτοντας επιπλέον μία κεντρική μηχανή καταγραφής των logs. Θα πρέπει να τονιστεί, ότι τα logs αποθηκεύονται συνήθως σε ένα προεπιλεγμένο αρχείο ή μία βάση δεδομένων που χρησιμοποιεί το εκάστοτε σύστημα κανόνων για την δημιουργία καταχωρήσεων των συμβάντων που ελήφθησαν. Τα IDSs χωρίζονται σε διάφορες κατηγορίες ανάλογα με την μεθοδολογία που χρησιμοποιούν για να παράγουν τις ειδοποιήσεις - συναγερμούς. Μια τέτοια κατηγορία είναι και τα συστήματα ανίχνευσης εισβολών σε δίκτυα (*NIDS*). Οι αισθητήρες (*sensors*) ενός τέτοιου συστήματος «διαβάζουν» όλο το σύνολο της κίνησης ενός δικτύου και αναλύουν το περιεχόμενο των πακέτων για την ανίχνευση κακόβουλης - παράνομης δραστηριότητας. Ένα σύστημα NIDS, είναι μια ανεξάρτητη πλατφόρμα στην οποία επισημαίνονται οι ενέργειες από την «εξέταση» της κίνησης του εκάστοτε δικτύου. Το Snort, είναι ένα τέτοιο σύστημα το οποίο αποτελεί και το κύριο αντικείμενο του πειραματικού μας σχεδιασμού.

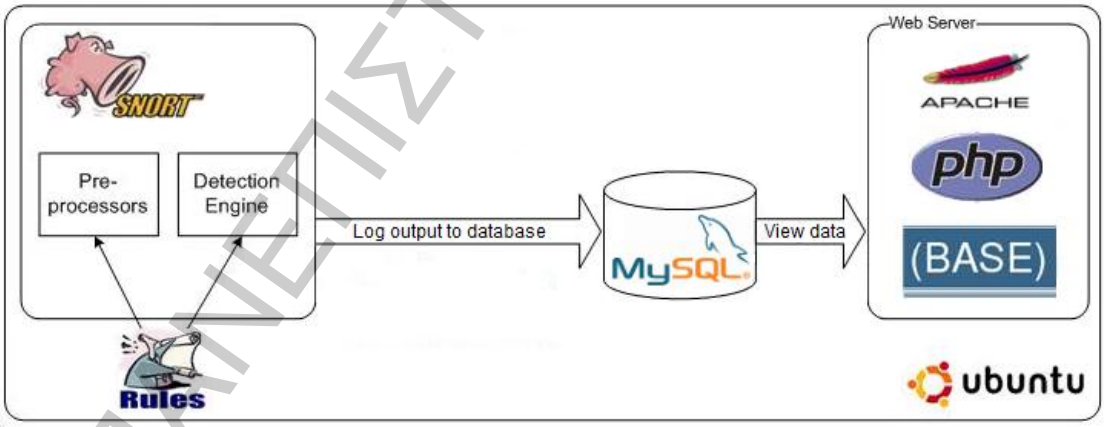
Όπως ήδη έχει γίνει αναφορά, το Snort είναι «ικανό» να αναλύει την κυκλοφορία των IP δικτύων, καταγράφοντας τα εκάστοτε πακέτα σε πραγματικό χρόνο (*real-time*). Για

τους σκοπούς της υλοποίησης μας θα επικεντρωθούμε στην έκδοση του Snort για Linux λειτουργικά συστήματα. Υπάρχουν, επίσης, και άλλες δημοφιλείς πλατφόρμες που υποστηρίζουν το Snort, όπως είναι για παράδειγμα τα Windows.

Το Linux έχει το πλεονέκτημα ότι είναι πολύ καλά τεκμηριωμένο και υποστηρίζεται από την κοινότητα των «οπαδών» της ελεύθερης διάθεσης του πηγαίου κώδικα. Αντίθετα, απαιτείται πολύ καλή γνώση των εντολών του Unix και της σύνταξης γενικότερα των αρχείων διαμόρφωσης. Επιπλέον, θα πρέπει να σημειωθεί ότι ανεξαρτήτως λειτουργικού συστήματος η λήψη και ο έλεγχος των πακέτων που διέρχονται σε ένα δίκτυο, απαιτεί σημαντική επεξεργαστική ισχύ.

Για τη δυνατότητα, λοιπόν, παρακολούθησης και διαχείρισης των αρχείων καταγραφής μέσω Web manager στο πειραματικό μας περιβάλλον, θα χρειαστούμε το Snort, την MySQL, τον Apache Web Server, την PHP, την βιβλιοθήκη ADOdb (*Database Abstraction Library*), καθώς και κάποια έξτρα *tarballs* για τη δυνατότητα δημιουργίας Image Graphs. Ακολούθως, παρουσιάζεται σχηματικά η δομή του συστήματος μας, μαζί με τα επιπλέον *components* που χρησιμοποιήθηκαν για την υλοποίηση του.

Σχηματική αποτύπωση της αρχιτεκτονικής του πειραματικού περιβάλλοντος



Όπως περιγράφηκε και στο *Κεφάλαιο 7*, το Snort έχει τη δυνατότητα να αποθηκεύει τα logs σε ένα δυαδικό αρχείο (*binary file*). Όταν όμως ο όγκος των δεδομένων είναι πολύ μεγάλος και συγκεκριμένα στο αρχικό στάδιο της βελτιστοποίησης του συστήματος ανίχνευσης, το Snort δεν μπορεί να χειριστεί τις τεράστιες ποσότητες πληροφοριών που παράγει, με αποτέλεσμα να είναι σχεδόν επιτακτική η ανάγκη για

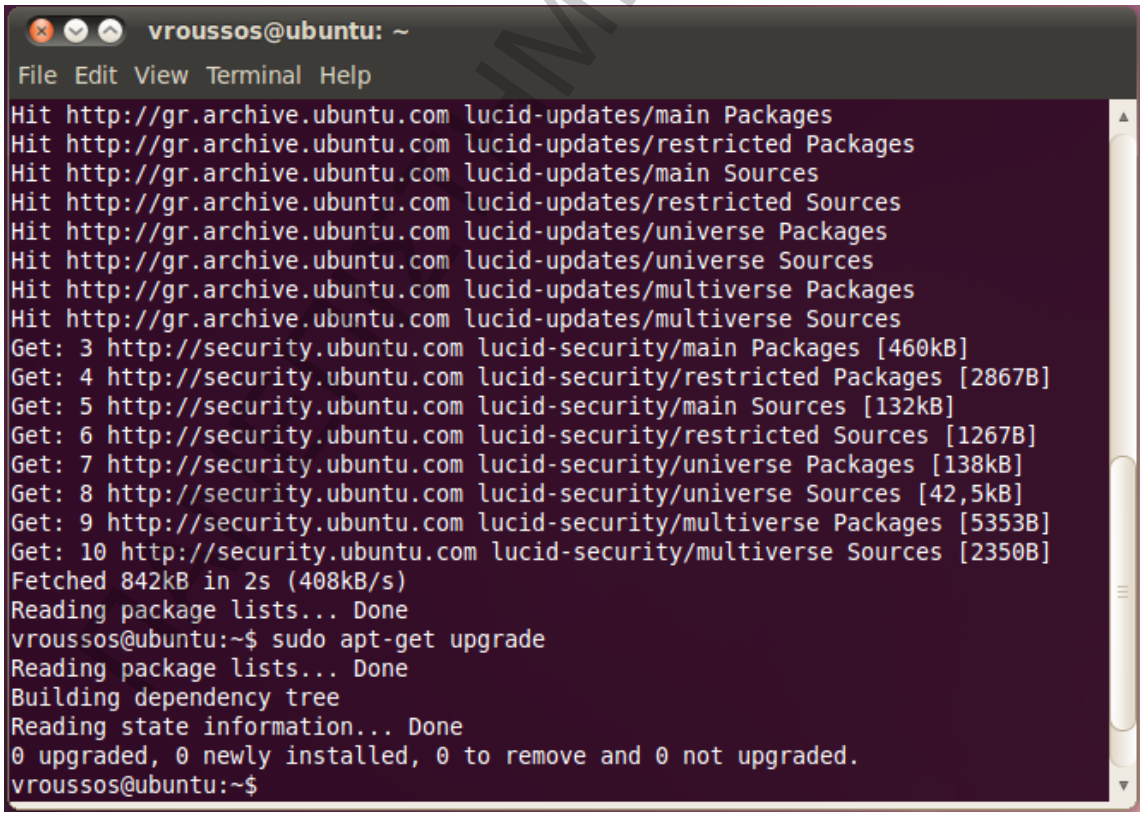
οργάνωση τους. Αποτέλεσμα των ανωτέρω, είναι η κατασκευή και η χρήση μίας βάσης δεδομένων, κάνοντας ευκολότερη την προσπέλαση των αποτελεσμάτων του Snort. Επίσης, ορισμένες επιθέσεις αναγνωρίζονται μόνο μετά από την σχολαστική ανάλυση των αρχείων καταγραφής που παράγει το Snort. Για αυτόν το σκοπό, θα ασχοληθούμε με την απευθείας εγγραφή του «ρεύματος εξόδου» σε μία MySQL βάση δεδομένων, διατηρώντας ένα αναλυτικό ιστορικό επιθέσεων.

Πριν προχωρήσουμε με την υλοποίηση του συστήματος μας, είναι απαραίτητο για τη σωστή του λειτουργία να εγκατασταθούν ορισμένα πακέτα λογισμικού:

- ✓ Για **Debian** ή **Ubuntu**, θα χρησιμοποιήσουμε τον apt.
- ✓ Αν έχουμε **Gentoo**, θα χρησιμοποιήσουμε τον portage.
- ✓ Σε **Red Hat**, **Fedora** ή **CentOS**, θα χρησιμοποιήσουμε το yum/yust.
- ✓ Αν έχουμε **Arch**, θα χρησιμοποιήσουμε τον pacman.

Ενημέρωση Λειτουργικού Συστήματος

```
sudo apt-get update
sudo apt-get upgrade
```



Εγκατάσταση Πακέτων Λογισμικού

```
sudo apt-get install nmap
sudo apt-get install nbtscan
sudo apt-get install php5
sudo apt-get install php5-mysql
sudo apt-get install php5-gd
sudo apt-get install libpcap0.8-dev
sudo apt-get install libpcre3-dev
sudo apt-get install g++
sudo apt-get install bison
sudo apt-get install flex
sudo apt-get install libpcap-ruby
sudo apt-get install autoconf
sudo apt-get install libtool
```

Επιπρόσθετα, εγκαθιστούμε τον mysql server και client για τη δημιουργία και χρησιμοποίηση της MySQL βάσης.

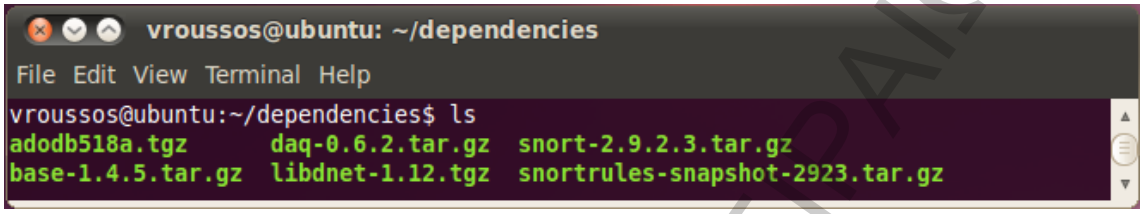
```
sudo apt-get install mysql-server
sudo apt-get install libmysqlclient-dev
```

Βεβαιωνόμαστε ότι έχουμε ενεργοποιήσει την υποστήριξη της MySQL στον Apache Web Server και είναι ενεργό το module στο αρχείο διαμόρφωσης του, `etc/apache2/apache2.conf`.

Σε αυτό το σημείο, θα πρέπει να κατεβάσουμε από το επίσημο site του Snort (www.snort.org) τον πηγαίο κώδικα της τρέχουσας έκδοσης του και έπειτα να τον αποσυμπιέσουμε στον υπολογιστή που θα υλοποιηθεί το σύστημα μας για την ανίχνευση των δικτυακών επιθέσεων και κατ' επέκταση της διαχείρισης των αρχείων καταγραφής.

```
sudo tar zxvf snort-2.9.2.3.tar.gz
cd snort-2.9.2.3
sudo ./configure -enable-dynamicplugin --with-mysql
sudo make
sudo make install
sudo mkdir /etc/snort
sudo mkdir /var/log/snort
sudo groupadd snort
sudo useradd -g snort snort
sudo chown snort:snort /var/log/snort
```

Στην περίπτωση, που θέλουμε να χρησιμοποιήσουμε **Gentoo** ή **Arch Linux** θα πρέπει απαραιτήτως να «χτίσουμε» το Snort με υποστήριξη της MySQL βάσης δεδομένων. Αν χρησιμοποιούμε Debian-based διανομές, θα πρέπει να κάνουμε μεταγλώττιση του κώδικα (*compile*) και να φτιάξουμε το αντίστοιχο `.deb` πακέτο που θα το εγκαταστήσουμε με την βοήθεια του `apt`. Το ακόλουθο στιγμιότυπο, παρουσιάζει τα *dependencies* που απαιτούνται για την υλοποίηση του Snort και του BASE.



Για τη λειτουργία ενός συστήματος NIDS, τον σημαντικότερο - σπουδαιότερο ρόλο έχει το αρχείο που περιέχει τους κανόνες (*snortrules*) με τους οποίους το Snort αποφασίζει ποια πακέτα είναι ύποπτα και ποια όχι. Συνεπώς, οι κανόνες/υπογραφές είναι η «καρδιά» της λειτουργίας του Snort, καθώς χωρίς τη χρησιμοποίηση τους δεν μπορεί να υπάρξει οποιοδήποτε είδος ανίχνευσης, με αποτέλεσμα το εκάστοτε δίκτυο να είναι απροστάτευτο και ευάλωτο σε κάθε είδους εισβολή. Τους επίσημους κανόνες, μπορούμε να τους κατεβάσουμε κάνοντας μία δωρεάν εγγραφή στον ιστοχώρο του Snort. Για να έχει κάποιος απευθείας πρόσβαση στους κανόνες και να μην περιμένει 30 ημέρες από την πρώτη δημοσίευση τους, θα πρέπει να εγγραφεί πληρώνοντας ένα χρηματικό ποσό ετησίως. Αφού, κατεβάσουμε το αρχείο `snortrules-snapshot-version.tar.gz` που περιέχει τους ενημερωμένους κανόνες (*Sourcefire VRT Certified Rules*), το αποσυμπιέζουμε και αποθηκεύουμε τα περιεχόμενα του στον φάκελο rules, `/etc/snort/rules`.

```
mv snortrules-snapshot-2923.tar.gz /etc/snort/rules
cd /etc/snort/rules
tar zxvf snortrules-snapshot-2923.tar.gz
```

Δημιουργία MySQL Βάσης Δεδομένων

Για να χρησιμοποιήσουμε μία MySQL βάση στην υλοποίηση του συστήματος μας, θα πρέπει αρχικά να κατασκευάσουμε το «σχήμα» της βάσης δεδομένων και εν συνεχεία έναν χρήστη με τα κατάλληλα δικαιώματα πρόσβασης σε αυτήν.

Συνδεόμαστε, λοιπόν, ως **root** χρήστης στον mysql server και δημιουργούμε τους πίνακες που θα χρησιμοποιεί η βάση δεδομένων για την αποθήκευση των ειδοποιήσεων του Snort, καθώς και τους χρήστες που θα έχουν πρόσβαση στη συγκεκριμένη βάση.

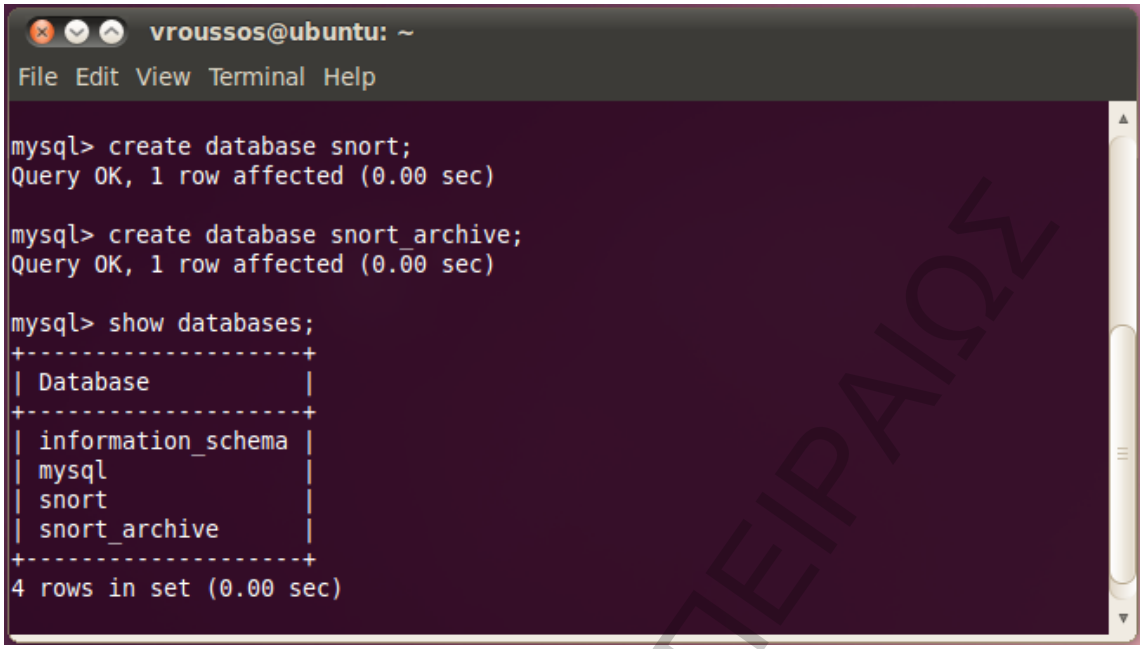
```
mysql -u root -p
create database snort;
grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort@localhost;
SET PASSWORD FOR snort@localhost=PASSWORD('mysnort'); // "mysnort" ο κωδικός χρήστη
exit
```

Όπως φαίνεται και από τον ανωτέρω κώδικα, δημιουργήσαμε μία βάση με όνομα "snort" και έναν χρήστη που θα την διαχειρίζεται, παραχωρώντας του πλήρη δικαιώματα.

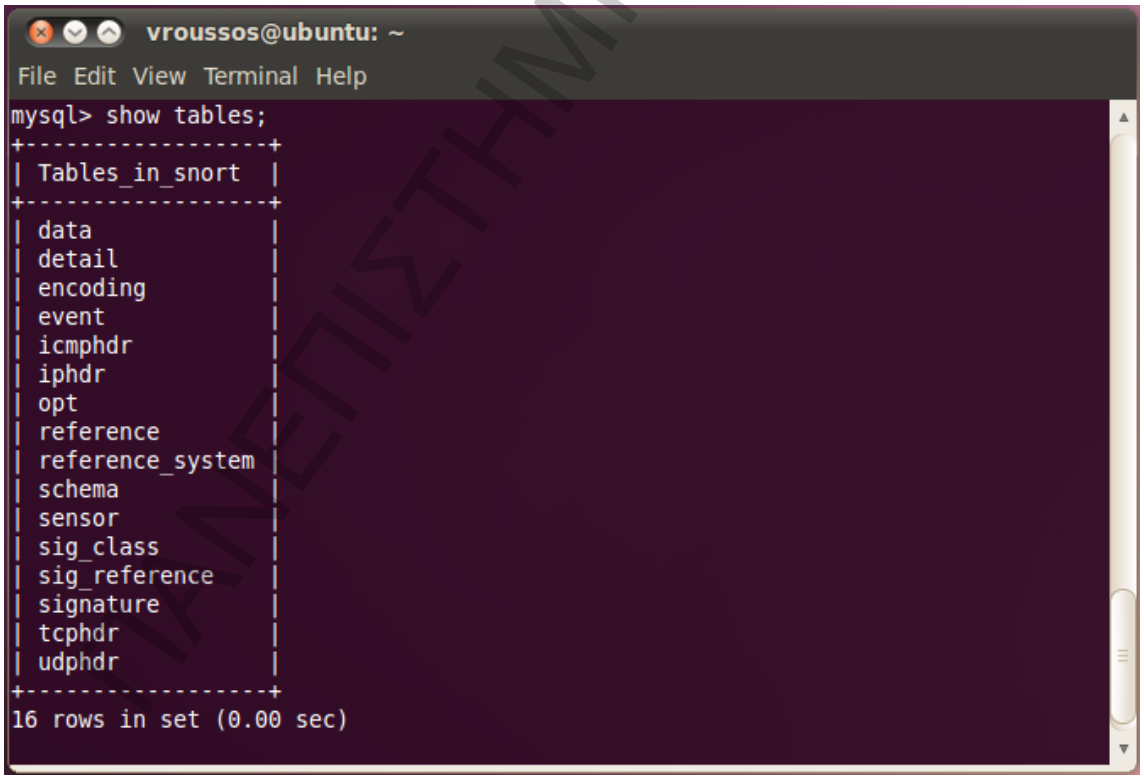
Κατασκευή βασικής δομής της βάσης δεδομένων

```
mysql -D snort -u snort -p < ~/vroussos/snort-2.9.2.3/schemas/create_mysql

mysql -u root -p
SHOW DATABASES;
use snort;
SHOW TABLES;
```

- Έλεγχος της βάσης δεδομένων



- Έλεγχος των πινάκων της βάσης δεδομένων



Αρχείο Διαμόρφωσης του Snort

Το βασικότερο και πιο σημαντικό «κομμάτι» όσον αφορά τη σωστή λειτουργία του Snort και κατ' επέκταση ολόκληρου του συστήματος μας, είναι η κατάλληλη τροποποίηση του αρχείου ρυθμίσεων του Snort, `snort.conf`. Καθώς, ο σκοπός της παρούσας διπλωματικής εργασίας είναι να κατανοήσουμε σε βάθος την λειτουργία ενός IDS και τον τρόπο με τον οποίο ανιχνεύει, δημιουργεί και διαχειρίζεται τους παραγόμενους συναγερμούς (*alerts*) και αντίστοιχα τα log files, κατασκευάσαμε σύμφωνα με τις ανάγκες και τις απαιτήσεις του δικτύου μας ορισμένους κανόνες (*rules*), τους οποίους στη συνέχεια τους προσαρμόσαμε κατάλληλα στο αρχείο διαμόρφωσης του Snort. Στο σημείο αυτό θα πρέπει να αναφερθεί, ότι ένα από τα ισχυρά πλεονεκτήματα του Snort, είναι ότι ελέγχει κάθε πακέτο για όσες «ύποπτες» συνθήκες έχουμε καθορίσει στους κανόνες του.

vroussos.rules

```
# Incoming connection on port 80

alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"Connection on port
8080"; sid:1231211;)

# Spam e-mail

alert tcp $HOME_NET any -> $MAIL_SERVER 20 (msg:"SPAM e-mail";
content:"Subject:"; pcre:"/spam/i"; sid:1231212;)

# Incoming Ping
```

- Windows OS

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING
Windows"; itype:8; content:"qrstuvwxyzefghij"; depth:16;
sid:12312131;)
```

- Linux/Unix OS

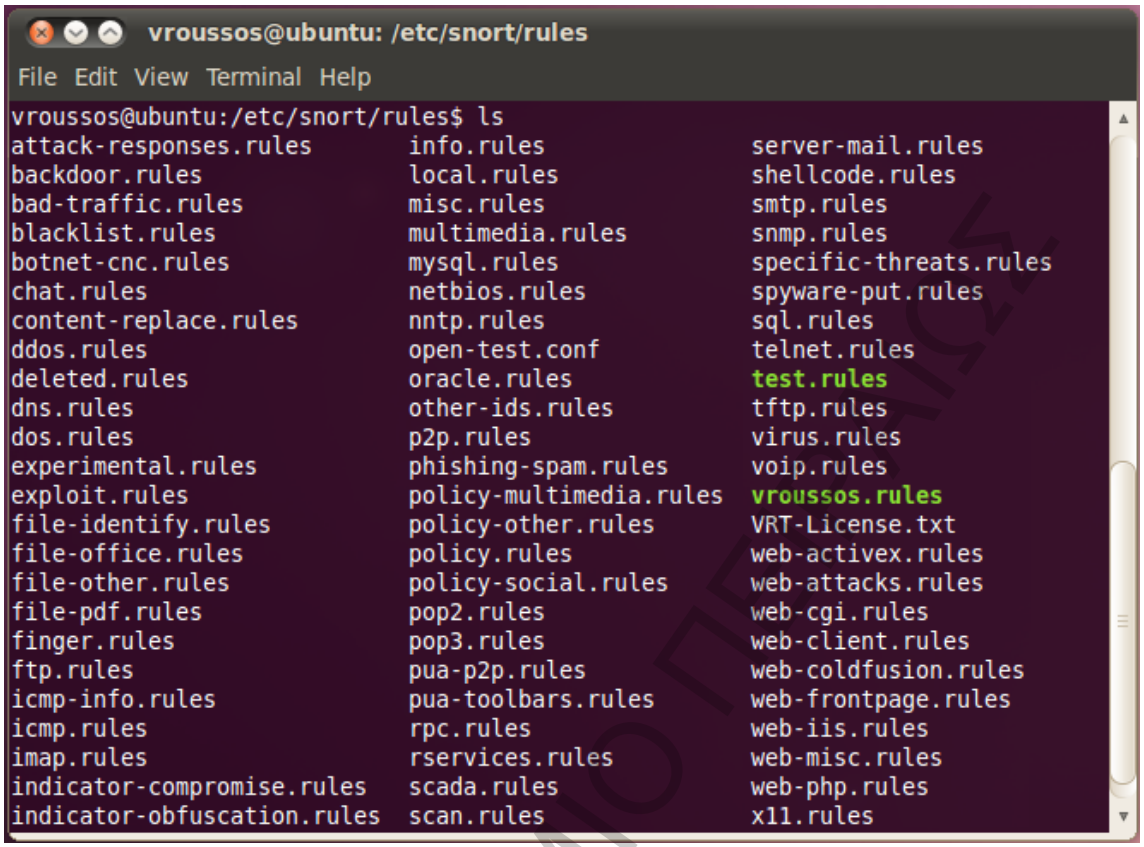
```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING Unix";
itype:8; content:"|10 11 12 13 14 15 16 17 18 19 1S 1T 1U 1V 1W 1X|";
depth:32; sid:12312132;)
```

```
# General bad-traffic

alert tcp $EXTERNAL_NET any <> $HOME_NET 1 (msg:"BAD-TRAFFIC tcp port
1 traffic"; sid:371; rev:8;)

alert udp $EXTERNAL_NET any <> $HOME_NET 1 (msg:"BAD-TRAFFIC udp port
1 traffic"; sid:372; rev:10;)

# Buffer overflow

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Oversized message
buffer overflow attempt"; content:"|2A|"; depth:1;
byte_test:2,>,1024,3,relative; classtype:attempted-user;)

# Alert! Someone open facebook

alert tcp any any -> any any (content:"www.facebook.com"; msg:"Someone
visiting facebook website.."; sid:1231213;)

# Alert! Worm

alert tcp any any -> any 1001 (content:"|58 75 5c 5c 7f|"; msg:"Alert!
Worm"; sid:1231214;)

# Port Scanning
```

Τέλος, ο παρακάτω κανόνας με τη χρήση του προεπεξεργαστή sfPortscan ανιχνεύει οποιαδήποτε σάρωση θύρας (*port scan*) γίνεται στον host που τρέχει το Snort.

```
preprocessor flow: stats_interval 0 hash 2
preprocessor sfportscan:\
    proto { all } \
    scan_type { all } \
    sense_level { low } \
    logfile { sfportscan.log }
```

Κανόνες του εσωτερικού συστήματος ανίχνευσης εισβολών



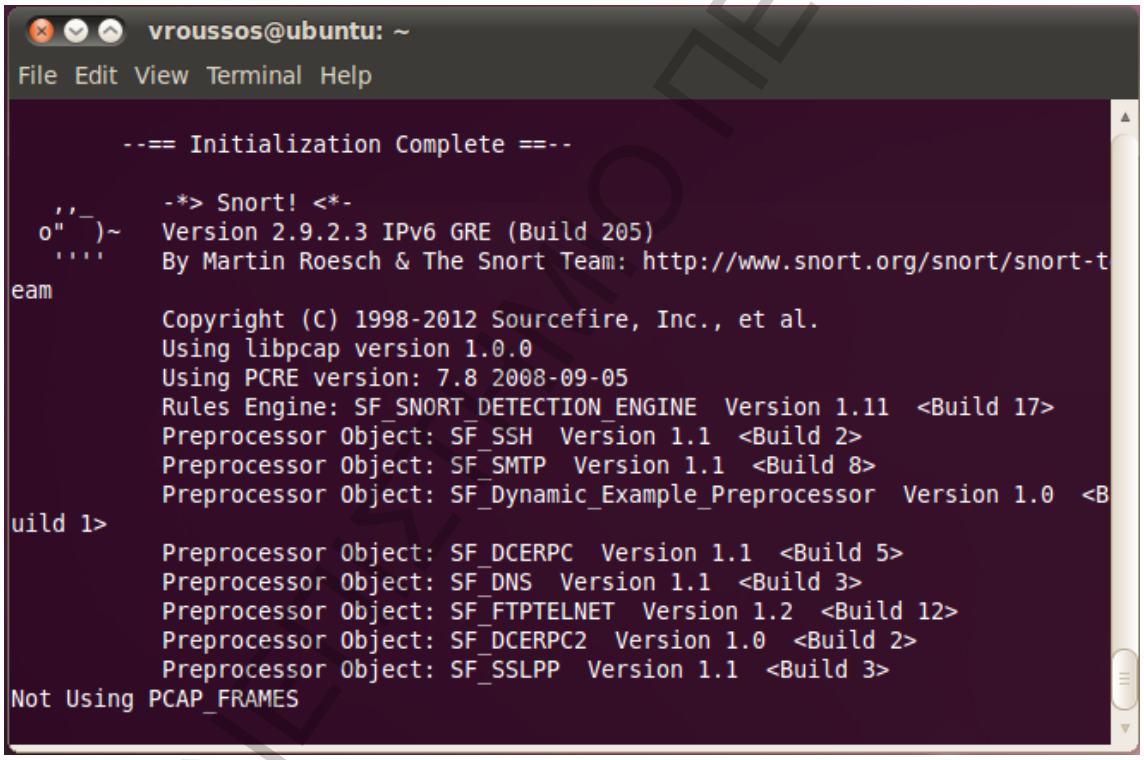
Ολοκληρώνοντας με τη δημιουργία των κανόνων, το επόμενο «θεμέλιο» βήμα στην υλοποίηση του συστήματος μας, είναι η παραμετροποίηση του Snort. Αναλυτικότερα, θα πρέπει να καθορίσουμε τη *συμπεριφορά ελέγχου* του δικτύου, τις τοποθεσίες των αναγκαίων αρχείων για την σωστή του λειτουργία και να δώσουμε συγκεκριμένες πληροφορίες τοπολογίας, ώστε το Snort να μας παρέχει σωστότερα και ασφαλέστερα αποτελέσματα.

Αρχικά, ορίζουμε τις τοποθεσίες των αρχείων δίνοντας την πλήρη διαδρομή τους. Μεγάλη σημασία έχει ο ορισμός της μεταβλητής **var HOME_NET**, όπου προσδιορίζουμε το εύρος του τοπικού δικτύου που θέλουμε να παρακολουθεί το Snort. Επιπλέον, η προσθήκη της γραμμής: "`output database: log, mysql, user=snort password=mysnort dbname=snort host=localhost`", έχει ιδιαίτερη σπουδαιότητα καθώς παίρνει όλες τις παραμέτρους ώστε το Snort να «τοποθετεί» τα αρχεία καταγραφής στην βάση δεδομένων που καθορίσαμε στην αρχή της υλοποίησης.

Για να εκτελεστεί το Snort σε λειτουργία NIDS θα πρέπει να δοθεί η παράμετρος -c, τιμή της οποίας είναι το αρχείο διαμόρφωσης του Snort, περιέχοντας όλες τις ρυθμίσεις που καθορίσαμε για τον τρόπο εκτέλεσης και λειτουργίας του συστήματος μας.

```
snort -c /etc/snort/snort.conf -i eth0
```

Αναλυτικά, το πρώτο όρισμα είναι το «μονοπάτι» όπου βρίσκεται το αρχείο διαμόρφωσης και το δεύτερο η διασύνδεση από όπου το πρόγραμμα παρακολουθεί την εκάστοτε κίνηση του δικτύου. Στη δική μας περίπτωση αυτή η διεπαφή είναι το `eth0`.

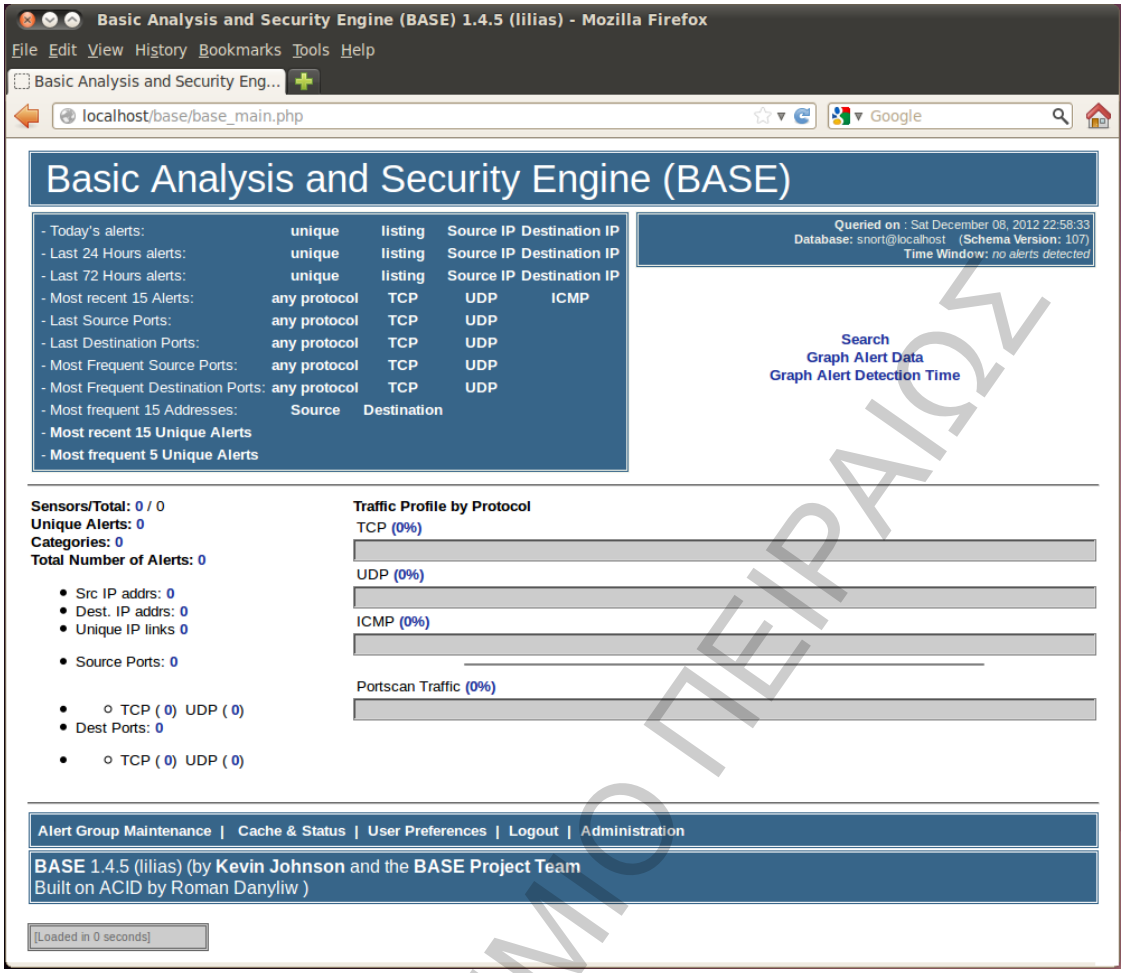


Εικόνα 10: **Στιγμιότυπο της λειτουργίας ελέγχου του Snort**

Basic Analysis and Security Engine (BASE)

Σε αυτό σημείο, έχουμε ένα λειτουργικό περιβάλλον ανίχνευσης δικτυακών επιθέσεων (*NIDS*), το οποίο είναι σε θέση να *ανιχνεύσει* και να *αποθηκεύσει* τα δικτυακά δεδομένα σε μία MySQL βάση δεδομένων. Κάθε φορά, που θα θέλουμε να δούμε στο μέλλον τα δεδομένα που έχουν συγκεντρωθεί στη βάση που δημιουργήσαμε, θα πρέπει να συνδεόμαστε με τον mysqlclient στη βάση snort και εκτελώντας τις κατάλληλες εντολές θα έχουμε τη δυνατότητα να εξάγουμε και να επεξεργαζόμαστε τις πληροφορίες που επιθυμούμε ανά πάσα στιγμή.

Επιπλέον, για την περαιτέρω ανάλυση των επιθέσεων, εγκαταστήσαμε στο σύστημα μας ένα γραφικό περιβάλλον χρήσης που έχει την δυνατότητα να επεξεργάζεται τα δεδομένα που υπάρχουν αποθηκευμένα στην MySQL βάση, προβάλλοντας τα σύμφωνα με τα κριτήρια προστασίας του δικτύου μας. Για τον ανωτέρω σκοπό, επιλέξαμε το BASE (*Basic Analysis and Security Engine*). Το BASE, είναι μία «μηχανή» ανάλυσης και διαχείρισης των logs που είναι αποθηκευμένα σε μία βάση δεδομένων, όπως είναι στην περίπτωση μας η MySQL. Η αρχική σελίδα του BASE, όπως φαίνεται και στην *Εικόνα 11*, παρέχει γρήγορη πρόσβαση σε πολλές και διαφορετικές πληροφορίες σχετικά με τα δεδομένα που αποθηκεύει το Snort στη βάση δεδομένων. Συγκεκριμένα, κάτω δεξιά της σελίδας παρουσιάζεται ποιο ποσοστό από το σύνολο των ληφθέντων πακέτων ανήκει σε κάθε ένα από τα τρία πρωτόκολλα TCP, UDP και ICMP. Ειδική κατηγορία αποτελεί το Portscan Traffic, όπου δίνεται το ποσοστό των πακέτων που ανεξαρτήτως πρωτοκόλλου αποτελούν «απόπειρα» σάρωσης πολλών θυρών ενός μηχανήματος. Πάνω αριστερά, υπάρχουν σύνδεσμοι για τις λίστες καταγραφής των πακέτων που ελήφθησαν τις τελευταίες 24 ή 72 ώρες, για τα πιο συχνά είδη ειδοποιήσεων - συναγερμών, καθώς και για τις πιο συχνές θύρες προορισμού/προέλευσης που έχουν εμφανιστεί. Κάτω αριστερά, δίνονται ο συνολικός αριθμός των ειδοποιήσεων (*alerts*), των IP προέλευσης/προορισμού, των θυρών προέλευσης, καθώς και σύνδεσμοι για την απευθείας προσπέλαση των ανωτέρω δεδομένων.



Εικόνα 11: Κονσόλα διαχείρισης του BASE

Apache2

Το BASE, είναι μία *εφαρμογή ιστού* βασιζόμενη στην γλώσσα προγραμματισμού PHP. Αυτομάτως, αυτό σημαίνει ότι θα χρειαστούμε επιπλέον και έναν web server με υποστήριξη php. Επιλέξαμε, λοιπόν, να εγκαταστήσουμε τον server ανοιχτού κώδικα Apache2, `/apt-get install apache2`.

Παρατηρούμε, ότι έχει δημιουργηθεί ένα νέος κατάλογος στη θέση /var/www/. Το root directory του Apache Web Server, από προεπιλογή είναι το /var/www/, σε αντίθετη περίπτωση ορίζουμε εμείς τη θέση που θέλουμε να βρίσκεται. Στο συγκεκριμένο σημείο, περιέχεται η αρχική σελίδα του Apache και εκεί θα πρέπει να τοποθετηθούν τα αρχεία οποιασδήποτε εφαρμογής ιστού θέλουμε να εκτελεστούν. Επιπλέον, όλα τα αρχεία ρυθμίσεων για τον Apache βρίσκονται στον κατάλογο `/etc/apache2/`.

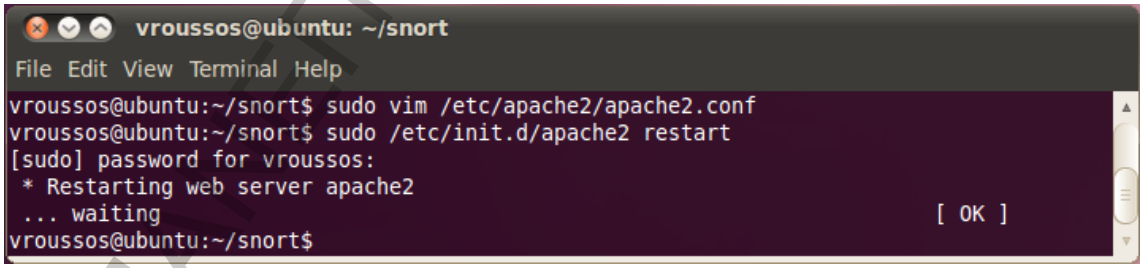
Ρύθμιση του BASE

Το επόμενο βήμα, είναι να κατεβάσουμε τον κώδικα του BASE από τη διεύθυνση http://sourceforge.net/projects/secureideas/ και να τον αποσυμπιέσουμε στο root directory του Apache. Επίσης, χρησιμοποιήθηκε και μία «διασύνδεση» της PHP που ονομάζεται ADOdb για την επικοινωνία με τη βάση δεδομένων που κατασκευάσαμε. Τον κώδικα της βιβλιοθήκης ADOdb, τον βρίσκουμε στην διεύθυνση http://sourceforge.net/projects/adodb/files/.

```
cd ~/vroussos
tar zxvf adodb518a.tgz
tar zxvf base-1.4.5.tar.gz
sudo mv adodb5 /var/www
sudo mv base-1.4.5 /var/www
```

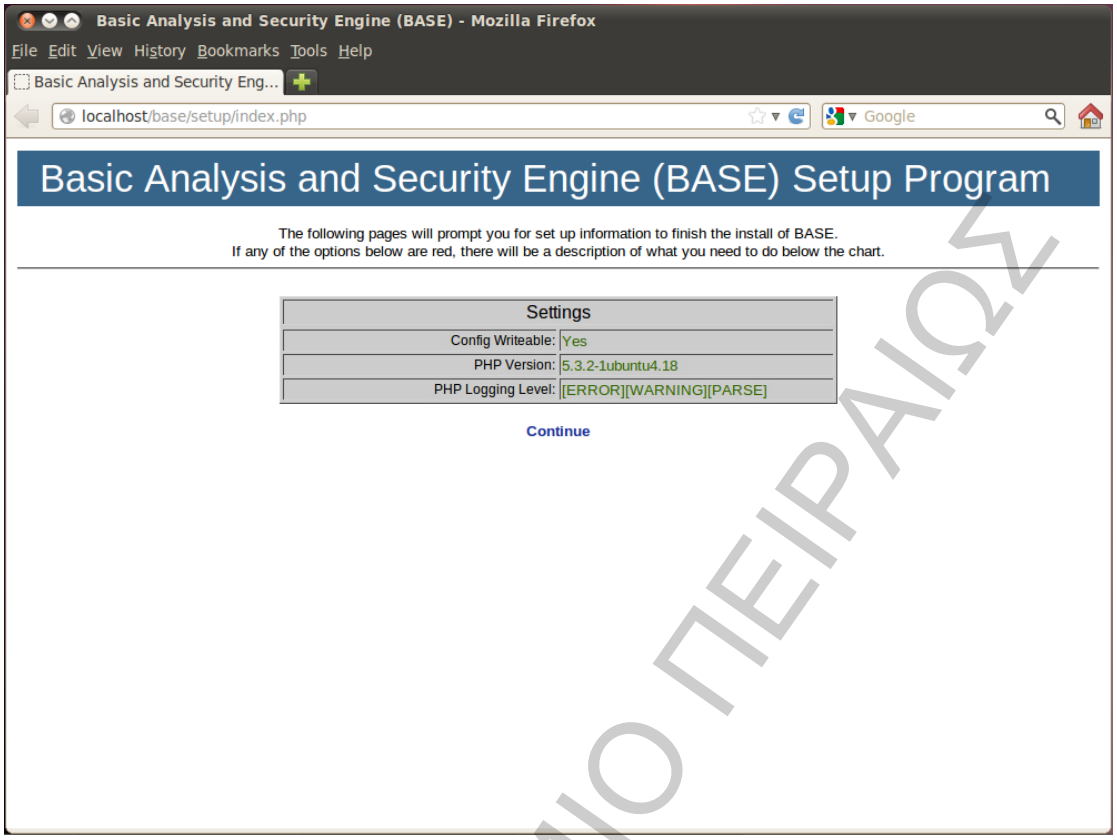
Στη συνέχεια, κάνουμε μετονομασία τον φάκελο για καλύτερο χειρισμό, `mv base-1.4.5 base` και αλλάζουμε τα «δικαιώματα» του γιατί θα μας χρειαστεί, `chmod a+w base`.

Κλείνοντας, κάνουμε επανεκκίνηση τον apache, `/etc/init.d/apache2 restart` και «αφήνουμε» το snort να τρέχει στο παρασκήνιο. Βεβαιωνόμαστε ότι το σύστημα μας λειτουργεί κανονικά με όλες τις παραμέτρους που του ορίσαμε για το τοπικό μας δίκτυο και δημιουργούμε ορισμένους δικούς μας πίνακες για τις εσωτερικές λειτουργίες του BASE.

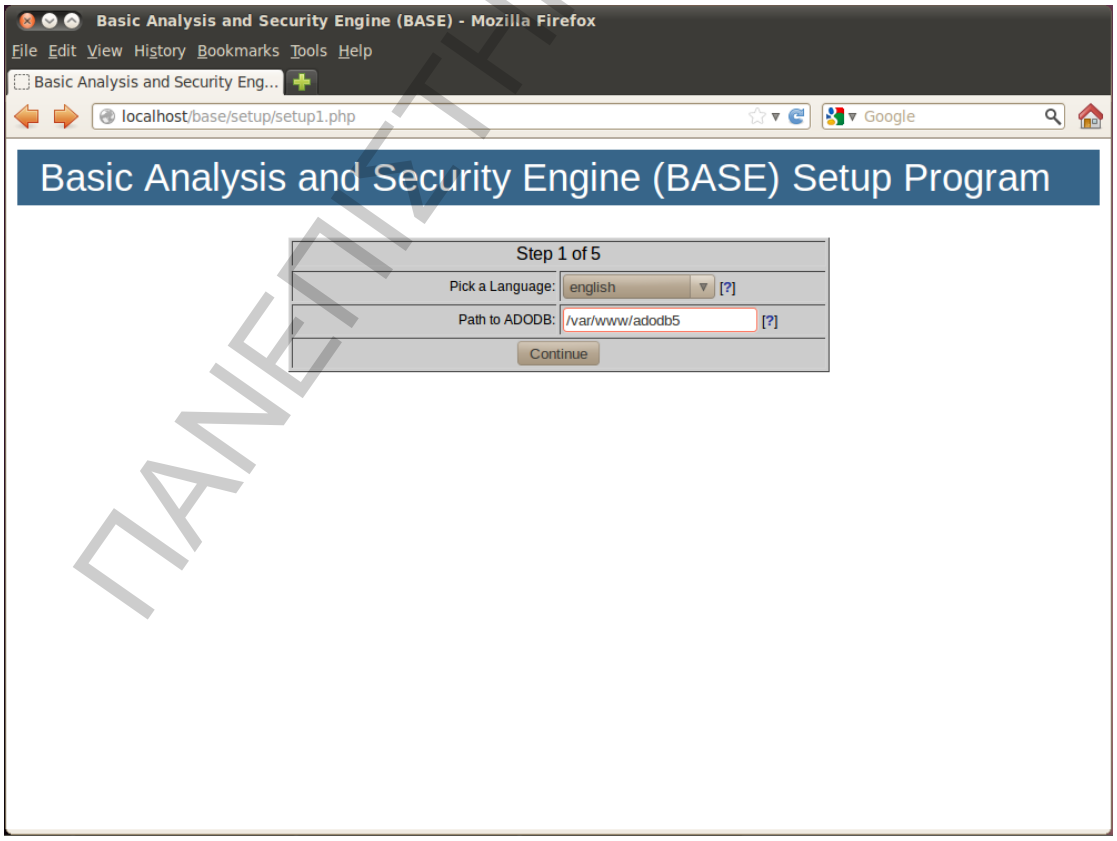


Για να επιτύχουμε το τελευταίο μέρος της υλοποίησης μας, ανοίγουμε έναν browser και εισάγουμε στην μπάρα διευθύνσεων του την διεύθυνση του τοπικού μας «κεντρικού» υπολογιστή, http://localhost/base, ακολουθώντας τα βήματα του BASE για την δημιουργία των πινάκων.

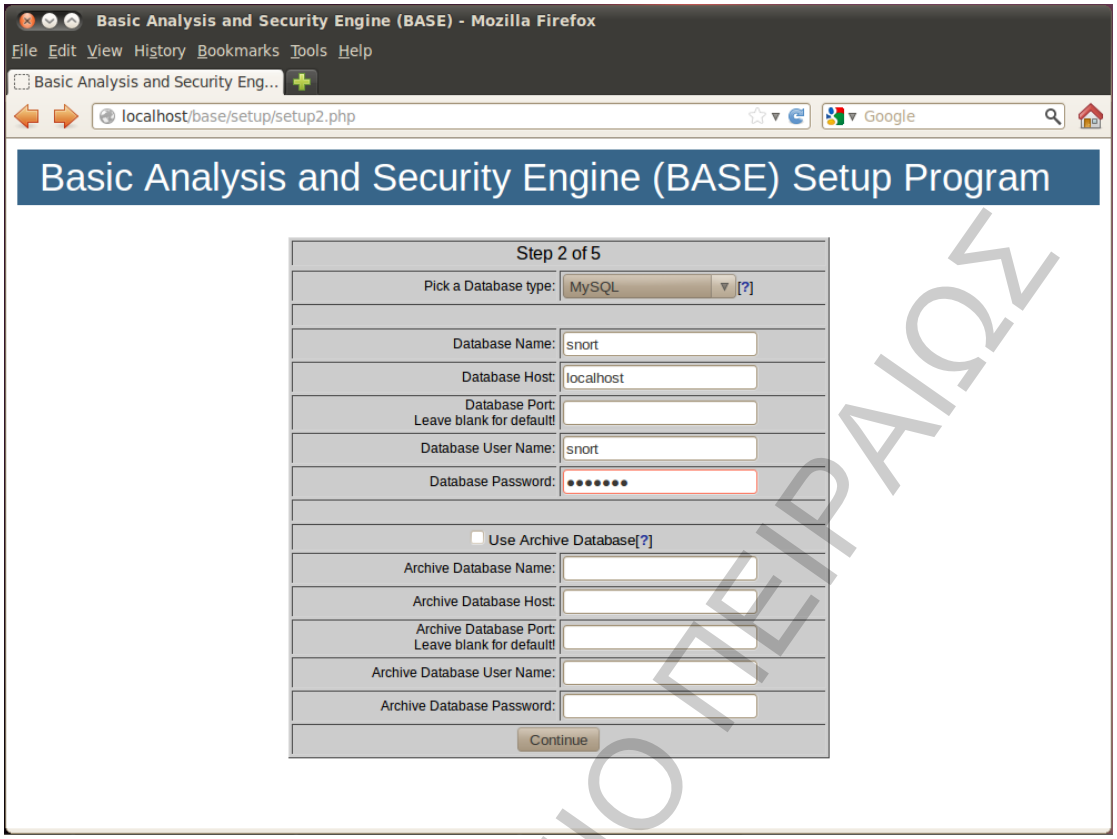
Διαμόρφωση του BASE



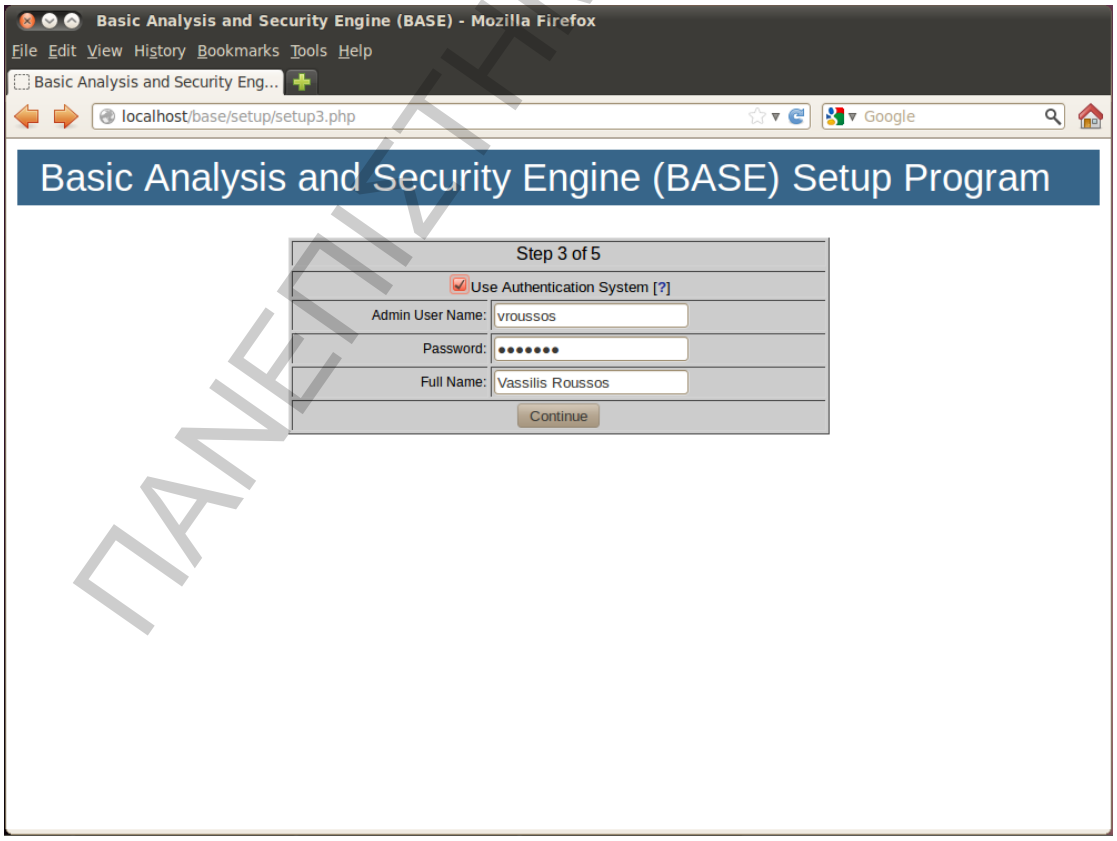
Βήμα #1



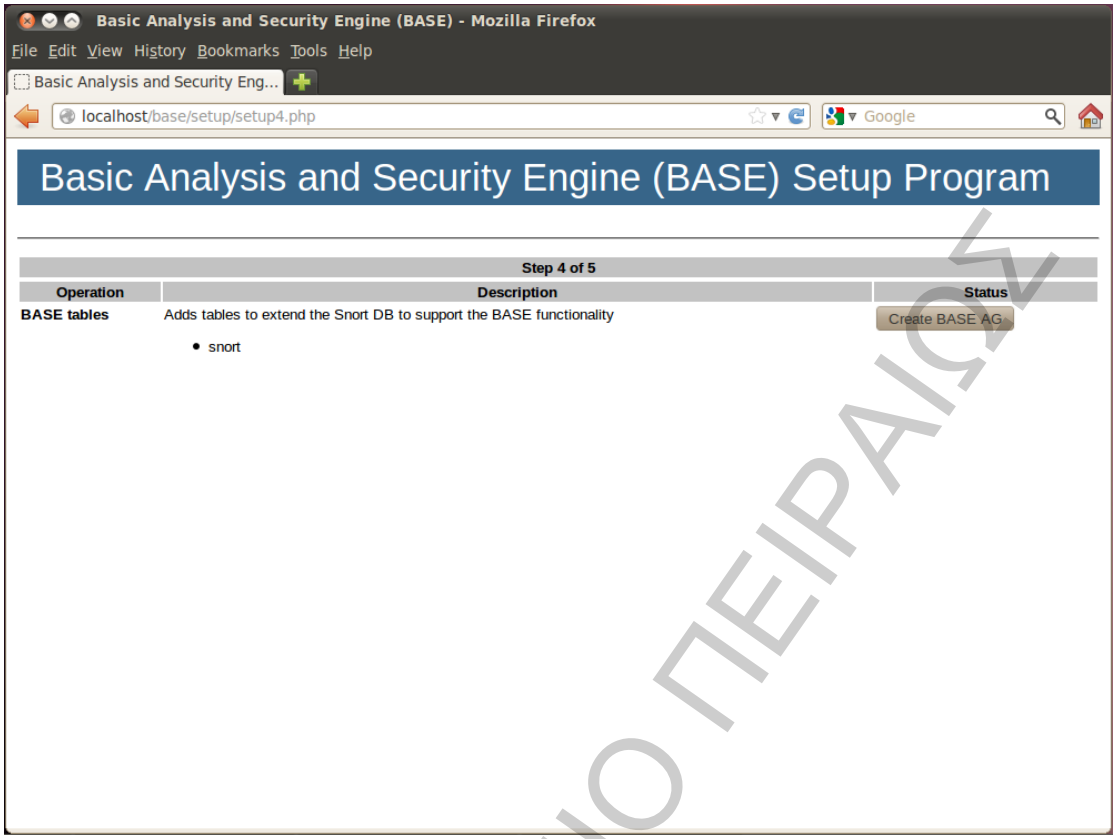
Βήμα #2: Παράμετροι σύνδεσης με τη βάση δεδομένων

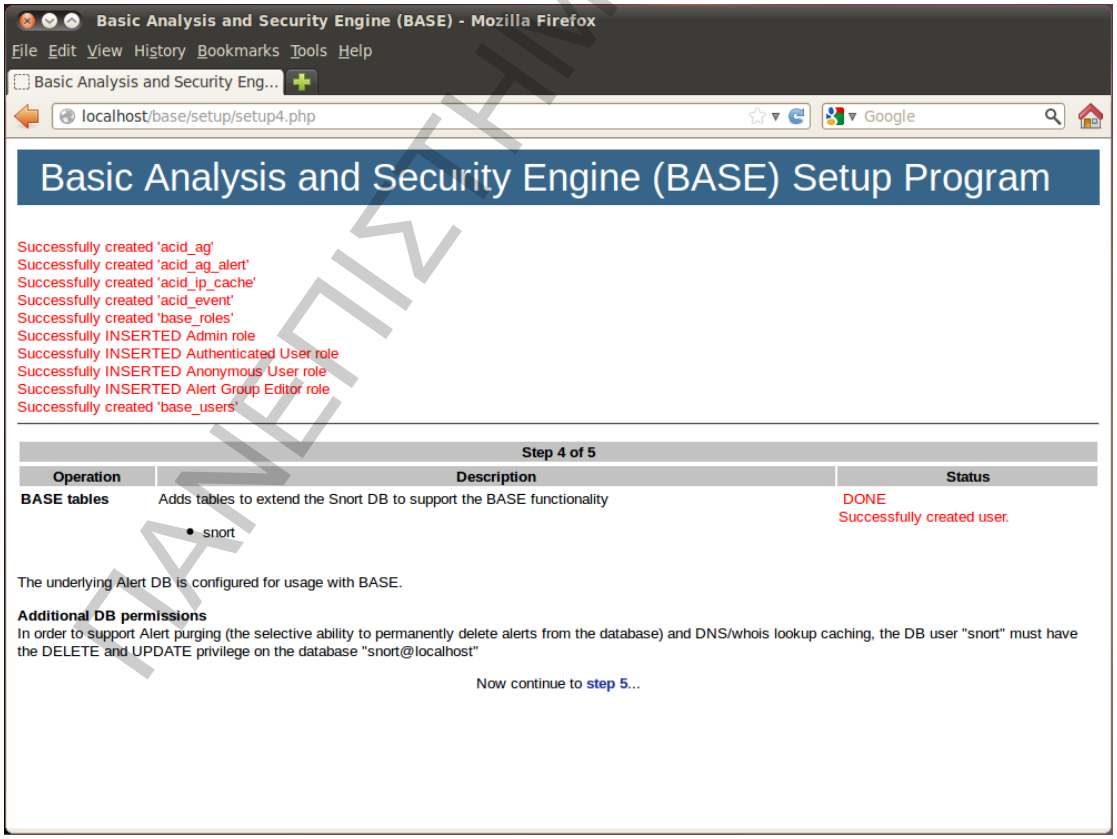


Βήμα #3: Σύστημα Ελέγχου Ταυτότητας

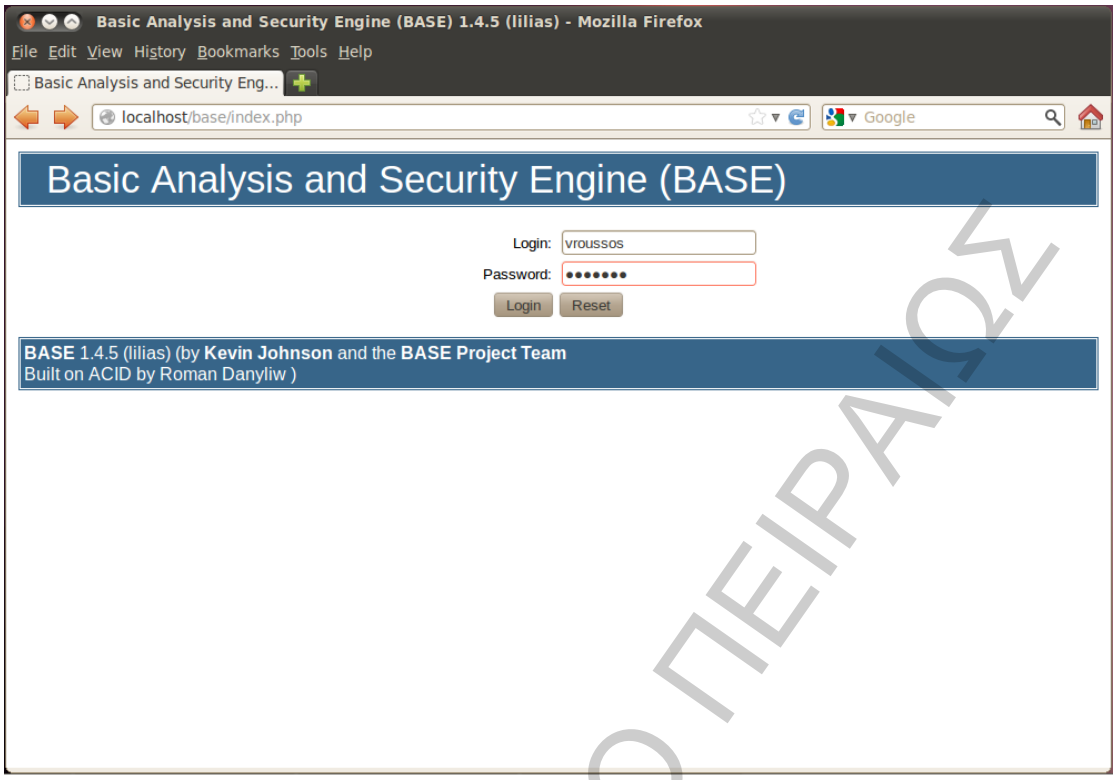


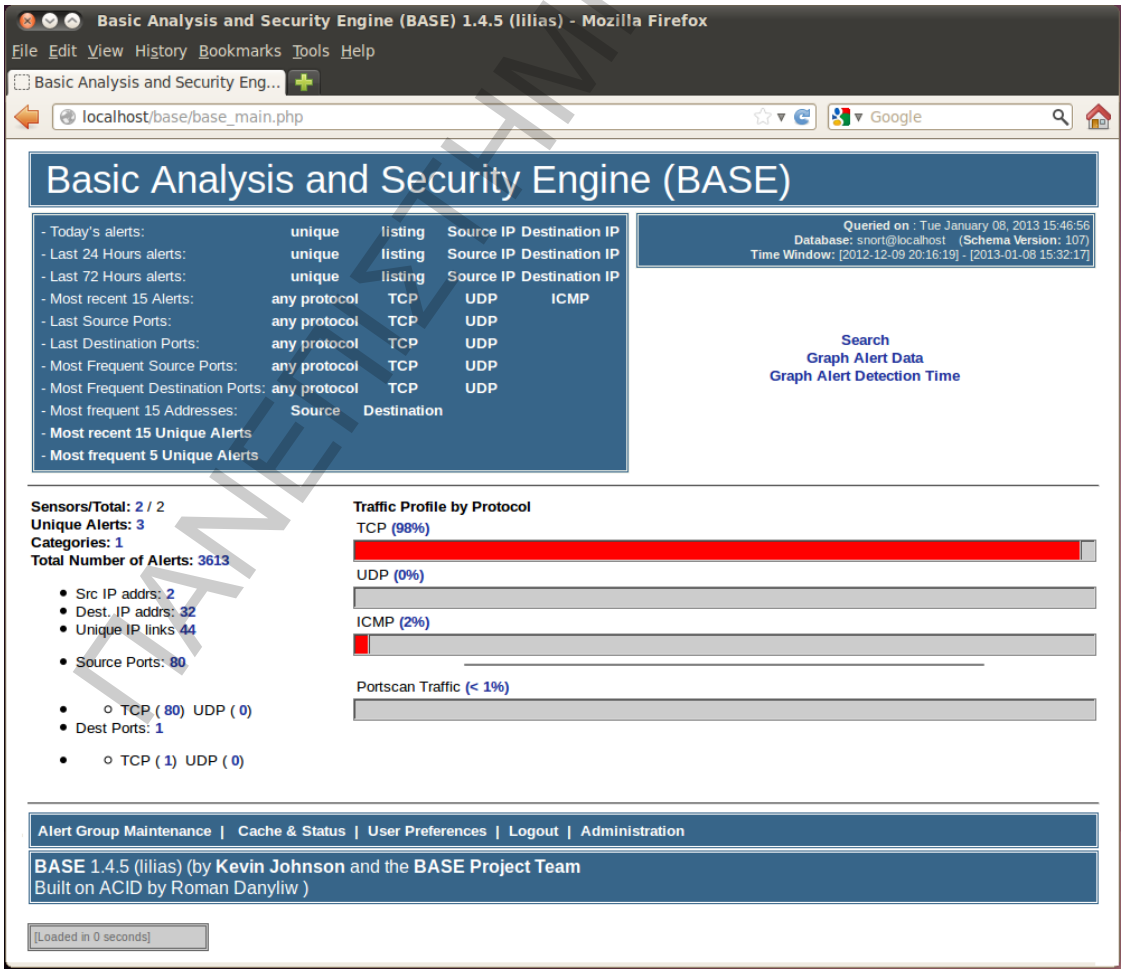
Βήμα #4: Δημιουργία BASE AG





Παραγόμενα αποτελέσματα - Ειδοποιήσεις παρακολούθησης

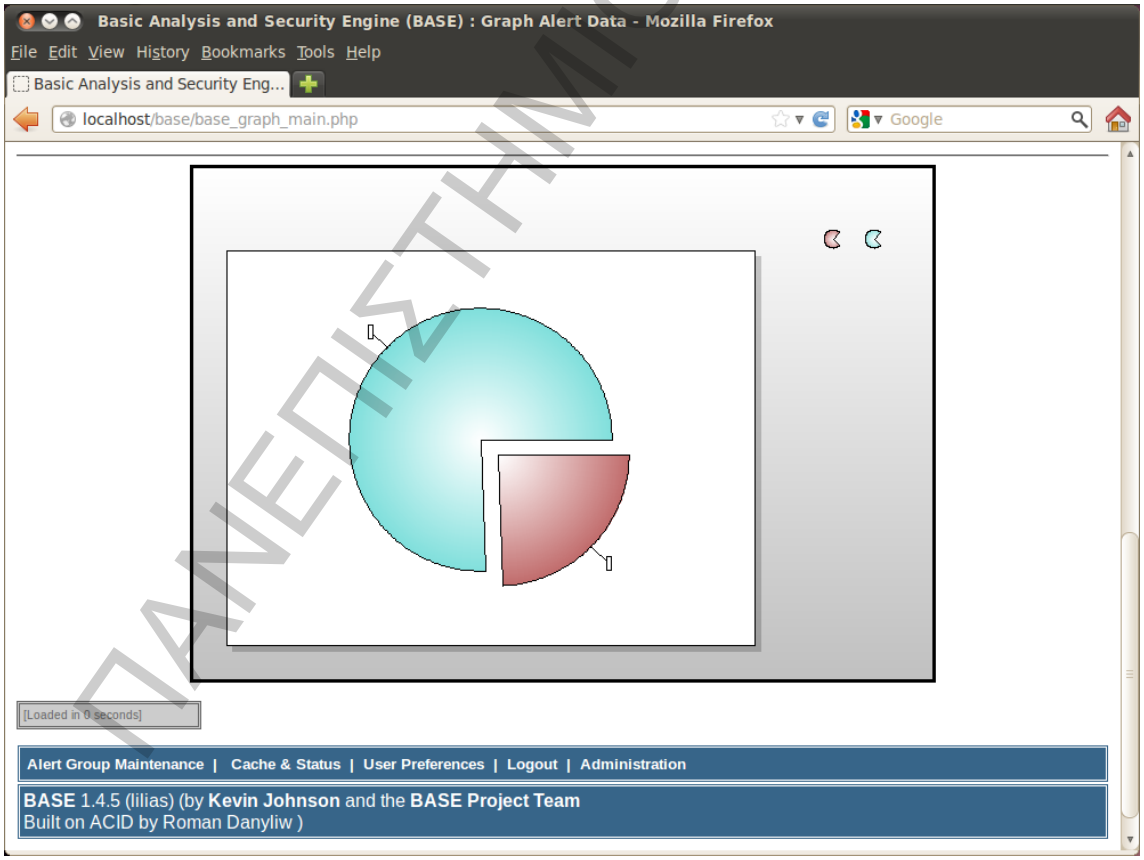


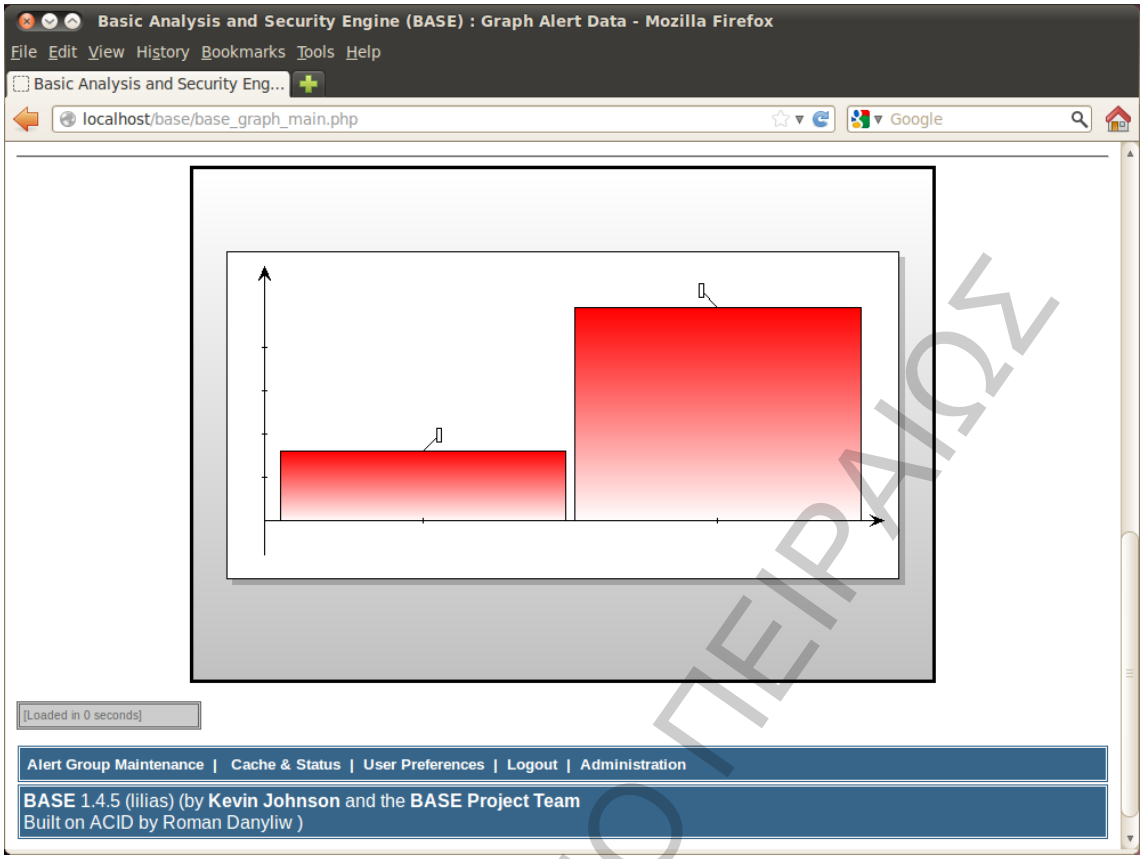


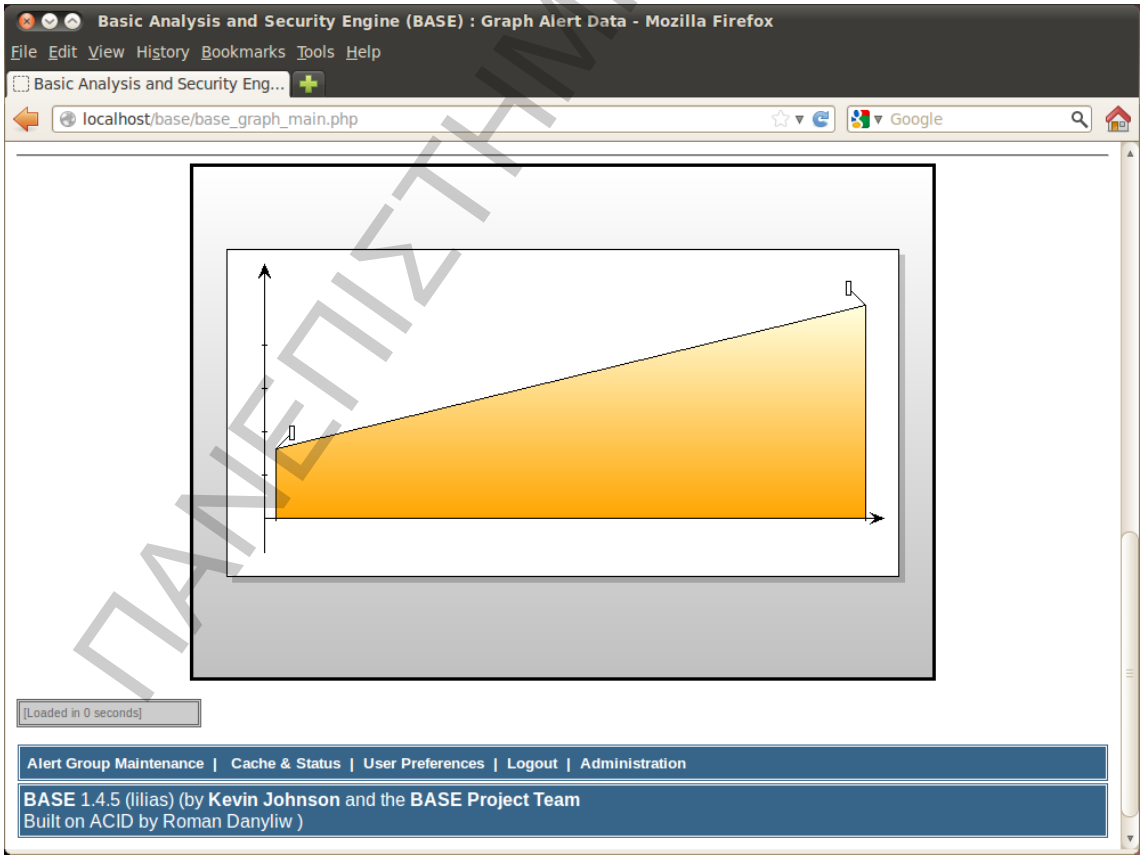
Διαγράμματα – Γραφήματα

Ο σύνδεσμος *Graph Alert Data* στην αρχική σελίδα των μετρήσεων μας, οδηγεί σε μία άλλη διεπαφή του Base από την οποία μπορούμε να εξάγουμε συγκριτικά διαγράμματα από συγκεκριμένες χρονικές περιόδους. Ενδεικτικά, μερικά από τα χρησιμότερα - σημαντικότερα διαγράμματα είναι η κατανομή του αριθμού των ειδοποιήσεων ανά ώρα, ημέρα ή μήνα, ο αριθμός ειδοποιήσεων για κάθε μία διεύθυνση IP προέλευσης/προορισμού, καθώς και ο αριθμός ειδοποιήσεων για κάθε χώρα προέλευσης/προορισμού σε παγκόσμιο χάρτη με προβολή του ποσοστού επί των συνολικών ειδοποιήσεων (*alerts*) που δέχτηκε το σύστημα μας. Αξιοσημείωτη είναι και η δυνατότητα επιλογής της χρονικής περιόδου από την οποία θέλουμε να εξάγονται τα γραφήματα μας. Στα ακόλουθα γραφήματα, παρουσιάζεται μία συγκεκριμένη IP προέλευσης σε σχέση με τον αριθμό των συνολικών ειδοποιήσεων.

Src. IP address vs. Number of Alerts







9 ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΡΟΒΛΗΜΑΤΙΣΜΟΙ

Στη σημερινή ψηφιακή πραγματικότητα, η «επανάσταση» της πληροφορίας σε συνδυασμό με την εκθετική αύξηση του διαδικτύου και την επέκταση των υπολογιστικών συστημάτων, έχει επιφέρει στην επιφάνεια το πρόβλημα της προστασίας του χρήστη και αντίστοιχα του ελέγχου της πληροφορίας.

Το Διαδίκτυο (*Internet*), τα τελευταία χρόνια, αποτέλεσε και συνεχίζει να αποτελεί ένα τεχνολογικό επίτευγμα, προσφέροντας στον εκάστοτε χρήστη δυνατότητες πρόσβασης και δημοσίευσης της πληροφορίας με «πολυπληθείς» τρόπους. Παράλληλα, λόγω της προαναφερόμενης ανάπτυξης του διαδικτύου και της τεχνολογίας γενικότερα, παρατηρείται το φαινόμενο της αύξησης όλο και πιο πολύπλοκων, εφευρετικών και επιζήμιων επιθέσεων που σαν μοναδικό στόχο έχουν τα δικτυωμένα συστήματα και τις συνεχώς αναπτυσσόμενες «δικτυακές» υπηρεσίες - εφαρμογές που εμπεριέχονται και προσφέρονται μέσα από αυτά. Με δεδομένη, αυτή την αυξανόμενη τάση για δικτύωση, τα «κλασσικά» μέτρα και οι ήδη υπάρχοντες μηχανισμοί ασφαλείας φαίνεται να μην επαρκούν ως προς την αναμενόμενη προστασία των συστημάτων, καθώς γεγονότα - επιθέσεις που συνέβησαν στο παρελθόν, «μαρτυρούν» πως η πρόληψη από μόνη της όσον αφορά την παραβίαση της ασφάλειας ενός δικτύου, είναι ανεπαρκής.

Η τεχνολογία, δεν εξελίσσεται μόνο προς όφελος του χρήστη αλλά και προς όφελος των διαφόρων επίδοξων επιτιθέμενων στα διάφορα δίκτυα. Οι εισβολείς, άλλαξαν τους στόχους τους και πλέον δεν επιτίθενται μόνο στα λειτουργικά συστήματα αλλά και στις εφαρμογές που διενεργούν σε αυτά. Όσο, λοιπόν, εξελίσσεται η τεχνολογία, τόσο θα αυξάνεται αυτομάτως η ανάγκη για συνεχή παρακολούθηση και ανάλυση των ανωτέρω επιθέσεων, δημιουργώντας ταυτόχρονα τις κατάλληλες «συνθήκες» προστασίας της *εμπιστευτικότητας*, της *ακεραιότητας* και της *διαθεσιμότητας* των εκάστοτε πληροφοριών. Άξιο αναφοράς αποτελεί το γεγονός, πως η αυθεντικοποίηση ενός μηνύματος είναι άρρηκτα συνδεδεμένη με την έννοια της ακεραιότητας (*integrity*), καθώς χωρίς ακεραιότητα δεν υφίσταται η έννοια της αυθεντικοποίησης (*message authentication*).

Κατά συνέπεια, προκειμένου να διασφαλιστεί η έγκαιρη ανίχνευση και η αποτελεσματική αντιμετώπιση των κακόβουλων χρηστών, με την πάροδο του χρόνου «γεννήθηκε» η ανάγκη δημιουργίας μίας συνεχούς αναπτυσσόμενης και εναλλακτικής μεθόδου προστασίας, η οποία καλείται στον ευρύτερο χώρο της Ασφάλειας Δικτύων ως Ανίχνευση Εισβολής (*Intrusion Detection*). Τα τωρινά δεδομένα, υποδεικνύουν πως η ύπαρξη και η δυνατότητα ενός Συστήματος Ανίχνευσης Εισβολών (*Intrusion Detection System - IDS*) να ανταπεξέρχεται σε καταστάσεις και κινήσεις αυξημένης επικινδυνότητας, θα γίνεται ολοένα και εντονότερη και κατ' επέκταση απαραίτητη. Πλέον, τα Συστήματα Ανίχνευσης Εισβολών θεωρούνται μία απαραίτητη προσθήκη στην πολιτική ασφαλείας κάθε δικτύου, κατέχοντας και αποτελώντας αντίστοιχα έναν πρωταγωνιστικό ρόλο και μία ολοκληρωμένη λύση για την πλήρη προστασία των δικτυακών επιθέσεων. Ο συνδυασμός των αποτελεσμάτων τους, με αυτά των ήδη γνωστών μηχανισμών ασφαλείας (π.χ. *firewall*), μπορεί να οδηγήσει στον σχηματισμό μιας πιο σφαιρικής και ολοκληρωμένης εικόνας των κινδύνων που προκύπτουν από τις διάφορες δικτυακές απειλές, συντελώντας με αυτόν τον τρόπο στον σχεδιασμό πιο αποτελεσματικών μέτρων ασφαλείας ενός δικτύου.

Υπάρχουν διάφοροι τύποι Συστημάτων Ανίχνευσης Εισβολών. Η κατηγοριοποίηση των IDSs, προκύπτει από τον διαχωρισμό τους, σύμφωνα με τον τρόπο που το κάθε ένα προσεγγίζει συγκεκριμένα πρότυπα και λειτουργίες. Την τελευταία δεκαετία, τα ευρέως χρησιμοποιούμενα IDSs, είναι αυτά που λειτουργούν σε επίπεδο δικτύου (*NIDS*). Τα Συστήματα Ανίχνευσης Δικτυακών Εισβολών (*NIDS*), βρίσκονται υπό συνεχή εξέλιξη που κυρίως έχει να κάνει με την βελτίωση της αποδοτικότητας και της αξιοπιστίας τους, καθώς δημιουργήθηκαν για να παρακολουθούν και να προστατεύουν σε συνεχή βάση δίκτυα από εξελιγμένες απειλές.

Η προτεινόμενη πρακτική χρήση ενός IDS σε ένα δίκτυο, περιλαμβάνει υψηλές παραμέτρους ασφαλείας σε «ζωτικά» σημεία του δικτύου, παρέχοντας πλήρη αυτοματοποίηση, καθώς και βέλτιστες πρακτικές ευελιξίας και παραγωγικότητας των λειτουργιών του. Σε κάθε περίπτωση, η αποτελεσματική, συνεχής και αδιάλειπτη σε βάθος λειτουργία των NIDS, δίνει την δυνατότητα για έναν ουσιαστικότερο έλεγχο αλλά και γνώση των αδυναμιών του εκάστοτε δικτύου. Αξιοσημείωτο, επίσης, είναι πως ένα Σύστημα Ανίχνευσης Δικτυακών Εισβολών από μόνο του δεν επαρκεί για την

ολοκληρωμένη προστασία και επίβλεψη ενός δικτύου από επιθέσεις - εισβολές, ενώ η σωστή εφαρμογή των κλασσικών μηχανισμών ασφάλειας πρέπει να θεωρείται δεδομένη. Σήμερα, διατίθενται πολλά και διαφορετικά NIDS, υλοποιημένα τόσο σε hardware/software όσο και με την μορφή εμπορικών ή open source εφαρμογών, ενώ το κριτήριο επιλογής ενός τέτοιου συστήματος, εξαρτάται από τους στόχους και τις ανάγκες προστασίας του κάθε δικτύου. Στην παρούσα διπλωματική εργασία, παρουσιάζεται το Snort (*Open Source*), καθώς θεωρείται στον τομέα της Ασφάλειας των Πληροφοριακών Συστημάτων ένα από τα πιο πετυχημένα, αποτελεσματικά και ευέλικτα εργαλεία που υπάρχουν για την Ανίχνευση των Δικτυακών Επιθέσεων.

Ευελιξία σημαίνει επιλογές. Ένα σύστημα ανίχνευσης δικτυακών επιθέσεων - απειλών σαν το Snort, μπορεί αναμφίβολα να προστατέψει ένα δίκτυο, καθώς διαθέτει και παρέχει σημαντικά στοιχεία «τεχνογνωσίας» ως προς την λειτουργία των επίδοξων εισβολέων. Η σχολαστική και εκτεταμένη τήρηση των πρακτικών κανόνων καταγραφής και διαχείρισης των logs ενός δικτύου, είναι το «κλειδί» όχι μόνο για τον χαρακτηρισμό ενός αξιόπιστου και δυναμικού NIDS, αλλά και για τον επιτυχημένο εντοπισμό ενάντια σε ένα ευρύ φάσμα επιθέσεων. Θα πρέπει, επίσης, να αναφερθεί, ότι το Snort εκτός από την χρήση του ως ένα Σύστημα Ανίχνευσης Δικτυακών Εισβολών (*Network-Intrusion Detection System – NIDS*), μπορεί να λειτουργήσει και ως Σύστημα Πρόληψης Δικτυακών Εισβολών (*Network-Intrusion Prevention System – NIPS*). Σε αυτή την περίπτωση, το Snort έχει την δυνατότητα να σταματήσει και εν συνεχεία να μπλοκάρει όποιες συνδέσεις θεωρεί επιθέσεις πριν καν αυτές φτάσουν στον προορισμό τους, αυξάνοντας με αυτόν τον τρόπο την συνολική ασφάλεια και προστασία ενός δικτύου. Συνεπώς, με την εισαγωγή του Snort σε ένα δίκτυο, γίνεται αυτομάτως εφικτή η αναγνώριση όλων των εξωτερικών επιθέσεων προς το συγκεκριμένο δίκτυο, προσφέροντας παράλληλα και την δυνατότητα της πρόληψης. Το βασικό χαρακτηριστικό και κύριο πλεονέκτημα του, είναι η ικανότητα να παρέχει μία δυναμική διασύνδεση με άλλες εσωτερικές ή εξωτερικές εφαρμογές. Με τα «πρόσθετα» εργαλεία που μπορεί να δεχτεί και να προσαρμοστούν στους κανόνες λειτουργίας του, διαχειρίζεται σε αναλυτικό επίπεδο την απόδοση και την αποτελεσματικότητα του εκάστοτε συστήματος, προσφέροντας μετρήσιμη ωφέλεια σε όλους του τομείς της ασφάλειας ενός δικτύου.

Κλείνοντας, θα πρέπει να τονίσουμε και να μας προβληματίσει το γεγονός, ότι η αποτελεσματικότητα των συστημάτων όπως είναι το Snort δεν εξαρτάται μόνο από το βάθος άσκησης ελέγχου, αλλά και από την ικανότητα να ελέγχουν σε πραγματικό χρόνο τον όγκο και την ποικιλία των δεδομένων, αποφασίζοντας σύμφωνα με τους δοθέντες κανόνες αν η χρήση του εύρους από συγκεκριμένες εφαρμογές είναι λογική και κατ' επέκταση επιτρεπόμενη. Οι κανόνες, θα πρέπει πάντοτε να προσαρμόζονται σύμφωνα με τα δεδομένα της εκάστοτε πολιτικής ασφάλειας και τη δραστηριότητα του δικτύου, αυξάνοντας συγχρόνως την παραγωγικότητα, εντείνοντας την ασφάλεια και παρουσιάζοντας στον διαχειριστή του δικτύου (*network administrator*) μια γενική και αξιόπιστη εικόνα για τη χρηστικότητα των πόρων του. Καθώς, η λειτουργία ενός NIDS είναι να αναλαμβάνει, να βελτιστοποιεί και να διεξαγάγει μέρος των λειτουργιών της προστασίας ενός δικτύου, σε καμία περίπτωση δεν θα πρέπει να του «ανατίθενται» λειτουργίες που εκτελούνται από άλλους μηχανισμούς ασφάλειας. Η ασφάλεια των δικτύων είναι απαίτηση αδιαπραγμάτευτη. Όσο, λοιπόν, οι «προκλήσεις» στον τομέα της Ασφάλειας Πληροφοριακών Συστημάτων πολλαπλασιάζονται καθημερινά, τόσο οι ανεπιθύμητες εισβολές θα αποτελούν ένα από τα σημαντικότερα ζητήματα στο μεταβαλλόμενο χώρο των δικτύων. Ασφαλή δικτυακά περιβάλλοντα μπορούν να επιτευχθούν μόνο αν η ασφάλεια προσεγγισθεί σαν μια *πολυσύνθετη έννοια* και όχι απλά σαν ένα *άθροισμα ενεργειών*, διαφυλάσσοντας και ασφαλίζοντας σε κάθε περίπτωση την ορθή «ισορροπία» ενός δικτύου.

10 ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Information Security: Principles and Practice - *Mark Stamp*, 2011
2. Recent Trends in Intrusion Detection System & Network Monitoring - *M. Sadiq Ali Khan, S. M. Aqil Burney*, 2012
3. Intrusion Detection Systems - *Pawel Skrobanek*, 2011
4. Cisco Systems Inc, http://www.cisco.com/en/US/docs/security/
5. Computer Security: Protecting Digital Resources - *Robert C Newman*, 2009
6. Evasions in Intrusion Prevention/Detection Systems - *Abhishek Singh, Scott Lambert, Tanmay A. Ganacharya, Jeff Williams*, 2010
7. Ασφάλεια Δικτύων Υπολογιστών - *Γκρίτζαλης Στ., Κάτσικας Σ., Γκρίτζαλης Δ.*, 2003
8. Network Security Essentials: Applications and Standards (4th Edition) - *William Stallings*, 2010
9. Network Warrior - *Gary A. Donahue*, 2011
10. Korea University - Professor Huy Kang Kim, http://ocw.korea.edu/ocw/college-of-engineering/network-security/NetworkSecurity1104.pdf
11. Network Intrusion Detection (3rd Edition) - *Stephen Northcutt, Judy Novak*
12. Intrusion Detection And Prevention System: CGI Attacks - *Tejinder Aulakh*, 2009
13. James P. Anderson, Computer Security Threat Monitoring and Surveillance - *James P. Anderson Co., Fort Washington, PA* (Apr. 1980)
14. Ασφάλεια Δικτύων - *Κάτσικας Σωκράτης, ΕΑΠ*, Πάτρα 2001
15. Wikipedia, http://en.wikipedia.org/
16. Intrusion Signatures and Analysis - *Mark Cooper, Stephen Northcutt, Matt Fearnow, Karen Frederick*
17. ENISA, http://www.enisa.europa.eu/
18. A Comparative Study on the Currently Existing Intrusion Detection Systems - *Ahmed M., Pal R., Hossain M., Hasan K., Bikas A.N.*, 2009
19. A Multilevel Secure Constrained Intrusion Detection System Prototype - *Kah Kin Ang*, 2012

20. A Hybrid IDS for Detecting Intrusions Based on Classification of Features and Complex Relations - *Aravind Kumar, International Journal of Engineering Research and Applications (www.ijera.com)*, 2011
21. Design of Intrusion Detection System Based on a New Pattern Matching Algorithm - *Zhang Hu*, 2009
22. How to Cheat at Securing Linux - *James Stanger*, 2008
23. Tripwire Inc, http://www.tripwire.com/it-resources/
24. Berkeley Packet Filter (BPF) syntax, http://biot.com/capstats/bpf.html
25. Extrusion Detection: Security Monitoring for Internal Intrusions - *Richard Bejtlich*, 2006
26. Applied Security Visualization - *Raffael Marty*, 2009
27. Improving snort performance under linux - *Salah K., Kahtani A*, 2009
28. Snort :: Home Page, http://www.snort.org/
29. Sourcefire | Network Security Solutions, http://www.sourcefire.com/
30. Managing Security with Snort & IDS Tools - *Kerry J. Cox, Christopher Gerg*
31. Nessus, Snort, & Ethereal Power Tools: Customizing Open Source Security Applications - *Neil Archibald, Gilbert Ramirez, Noam Rathaus, Josh Burke, Brian Caswell, Renaud Deraison*, 2006
32. Intrusion Detection with Snort - *Jack Koziol, TechRepublic*
33. Martin Roesch, http://securitysauce.blogspot.gr/
34. Hack the Stack: Using Snort and Ethereal to Master The 8 Layers of An Insecure Network - *Michael Gregg, Stephen Watkins, George Mays, Chris Ries, Ron Bandes, Brandon Franklin*, 2007
35. TCPDUMP/LIBPCAP, http://www.tcpdump.org/#documentation
36. Intrusion Detection With Mondrian and Snort - *Gerhard Jauk*, 2009
37. Intrusion Prevention and Active Response: Deploying Network and Host IPS - *Michael Rash, Angela Orebaugh, Graham Clark, Becky Pinkard, Jake Babbin*, 2006
38. Snort IDS and IPS Toolkit - *Andrew Baker, Joel Esler, Raven Alder*, 2007
39. Layered Approach Using Conditional Random Fields for Intrusion Detection - *Kapil Kumar Gupta, Baikunth Nath, Ramamohanarao Kotagiri*, 2010 (vol. 7 no. 1)

40. Framework of Intrusion Detection System via Snort Application on Campus Network Environment - *Ismail M.N., Ismail, M.T*, 2009

41. Intrusion Detection Systems (Advances in Information Security) - *Roberto Pietro, Luigi V. Mancini*, 2010

42. Practical Intrusion Analysis: Prevention and Detection for the Twenty-First Century - *Ryan Trost*, 2010

43. Packet Storm, http://packetstormsecurity.com/files/tags/exploit/

ΠΑΡΑΡΤΗΜΑ Α

```
#--------------------------------------------------
#   VRT Rule Packages Snort.conf
#
#   For more information visit us at:
#     http://www.snort.org                    Snort Website
#     http://vrt-sourcefire.blogspot.com/     Sourcefire VRT Blog
#
#     Mailing list Contact:      snort-sigs@lists.sourceforge.net
#     False Positive reports:    fp@sourcefire.com
#     Snort bugs:                bugs@snort.org
#
#     Compatible with Snort Versions:
#     VERSIONS : 2.9.2.3
#
#     Snort build options:
#     OPTIONS : --enable-ipv6 --enable-gre --enable-mpls --enable-
targetbased --enable-decoder-preprocessor-rules --enable-ppm --
enable-perfprofiling --enable-zlib --enable-active-response --
enable-normalizer --enable-reload --enable-react --enable-flexresp3
#
#     Additional information:
#     This configuration file enables active response, to run snort
in
#     test mode -T you are required to supply an interface -i
<interface>
#     or test mode will fail to fully validate the configuration
and
#     exit with a FATAL error
#--------------------------------------------------

######################################################
# This file contains a sample snort configuration.
# You should take the following steps to create your own custom
configuration:
#
#  1) Set the network variables.
#  2) Configure the decoder
#  3) Configure the base detection engine
#  4) Configure dynamic loaded libraries
#  5) Configure preprocessors
#  6) Configure output plugins
#  7) Customize your rule set
#  8) Customize preprocessor and decoder rule set
#  9) Customize shared object rule set
######################################################

######################################################
# Step #1: Set the network variables.  For more information, see
README.variables
######################################################

# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.1.0/24

# Set up the external network addresses. Leave as "any" in most
situations
```

```
ipvar EXTERNAL_NET any

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

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

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

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

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

# List of ssh servers on your network
ipvar SSH_SERVERS $HOME_NET

# List of ftp servers on your network
ipvar FTP_SERVERS $HOME_NET

# List of sip servers on your network
ipvar SIP_SERVERS $HOME_NET

# List of ports you run web servers on
portvar HTTP_PORTS
[80,81,311,591,593,901,1220,1414,1741,1830,2301,2381,2809,3128,3702
,4343,4848,5250,7001,7145,7510,7777,7779,8000,8008,8014,8028,8080,8
088,8090,8118,8123,8180,8181,8243,8280,8800,8888,8899,9000,9080,909
0,9091,9443,9999,11371,55555]

# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1024:

# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22

# List of ports you run ftp servers on
portvar FTP_PORTS [21,2100,3535]

# List of ports you run SIP servers on
portvar SIP_PORTS [5060,5061,5600]

# List of file data ports for file inspection
portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]

# List of GTP ports for GTP preprocessor
portvar GTP_PORTS [2123,2152,3386]

# other variables, these should not be modified
ipvar AIM_SERVERS
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.20
```

```
0.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,
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 /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative
to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG
89986
# Set the absolute path appropriately
var WHITE_LIST_PATH ../rules
var BLACK_LIST_PATH ../rules

##################################################
# Step #2: Configure the decoder.  For more information, see
README.decode
##################################################

# 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
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 th
elength of the packet
# config enable_decode_oversized_alerts

# Same as above, but drop packet if in Inline mode (requires
enable_decode_oversized_alerts)
# config enable_decode_oversized_drops

# Configure IP / TCP checksum mode
config checksum_mode: all

# Configure maximum number of flowbit references.  For more
information, see README.flowbits
# config flowbits_size: 64
```

```
# Configure ports to ignore
# config ignore_ports: tcp 21 6667:6671 1356
# config ignore_ports: udp 1:17 53

# Configure active response for non inline operation. For more
information, see REAMDE.active
# config response: eth0 attempts 2

# Configure DAQ related options for inline operation. For more
information, see README.daq
#
# config daq: <type>
# config daq_dir: <dir>
# config daq_mode: <mode>
# config daq_var: <var>
#
# <type> ::= pcap | afpacket | dump | nfq | ipq | ipfw
# <mode> ::= read-file | passive | inline
# <var> ::= arbitrary <name>=<value passed to DAQ
# <dir> ::= path as to where to look for DAQ module so's

# Configure specific UID and GID to run snort as after dropping
privs. For more information see snort -h command line options
#
# config set_gid:
# config set_uid:

# Configure default snaplen. Snort defaults to MTU of in use
interface. For more information see README
#
# config snaplen:
#

# Configure default bpf_file to use for filtering what traffic
reaches snort. For more information see snort -h command line
options (-F)
#
# config bpf_file:
#

# Configure default log directory for snort to log to.  For more
information see snort -h command line options (-l)
#
# config logdir:


##################################################
# Step #3: Configure the base detection engine.  For more
information, see  README.decode
##################################################

# Configure PCRE match limitations
config pcre_match_limit: 3500
config pcre_match_limit_recursion: 1500
# Configure the detection engine  See the Snort Manual, Configuring
Snort - Includes - Config
config detection: search-method ac-split search-optimize max-
pattern-len 20
```

```
# Configure the event queue.  For more information, see
README.event_queue
config event_queue: max_queue 8 log 3 order_events content_length
##################################################
## Configure GTP if it is to be used.
## For more information, see README.GTP
##################################################

# config enable_gtp

##################################################
# Per packet and rule latency enforcement
# For more information see README.ppm
##################################################

# Per Packet latency configuration
#config ppm: max-pkt-time 250, \
#   fastpath-expensive-packets, \
#   pkt-log

# Per Rule latency configuration
#config ppm: max-rule-time 200, \
#   threshold 3, \
#   suspend-expensive-rules, \
#   suspend-timeout 20, \
#   rule-log alert

##################################################
# Configure Perf Profiling for debugging
# For more information see README.PerfProfiling
##################################################

#config profile_rules: print all, sort avg_ticks
#config profile_preprocs: print all, sort avg_ticks

##################################################
# Configure protocol aware flushing
# For more information see README.stream5
##################################################
config paf_max: 16000

##################################################
# Step #4: Configure dynamic loaded libraries.
# For more information, see Snort Manual, Configuring Snort -
Dynamic Modules
##################################################

# path to dynamic preprocessor libraries
dynamicpreprocessor directory
/usr/local/lib/snort_dynamicpreprocessor/

# path to base preprocessor engine
dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so
# path to dynamic rules libraries
dynamicdetection directory /usr/local/lib/snort_dynamicrules
```

```
##################################################
# Step #5: Configure preprocessors
# For more information, see the Snort Manual, Configuring Snort -
Preprocessors
##################################################
# GTP Control Channle Preprocessor. For more information, see
README.GTP
# preprocessor gtp: ports { 2123 3386 2152 }

# Inline packet normalization. For more information, see
README.normalize
# Does nothing in IDS mode
preprocessor normalize_ip4
preprocessor normalize_tcp: ips ecn stream
preprocessor normalize_icmp4
preprocessor normalize_ip6
preprocessor normalize_icmp6

# Target-based IP defragmentation.  For more inforation, see
README.frag3
preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy windows detect_anomalies
overlap_limit 10 min_fragment_length 100 timeout 180

# Target-Based stateful inspection/stream reassembly.  For more
inforation, see README.stream5
preprocessor stream5_global: track_tcp yes, \
   track_udp yes, \
   track_icmp no, \
   max_tcp 262144, \
   max_udp 131072, \
   max_active_responses 2, \
   min_response_seconds 5
preprocessor stream5_tcp: policy windows, detect_anomalies,
require_3whs 180, \
   overlap_limit 10, small_segments 3 bytes 150, timeout 180, \
    ports client 21 22 23 25 42 53 79 109 110 111 113 119 135 136
137 139 143 \
        161 445 513 514 587 593 691 1433 1521 1741 2100 3306 6070
6665 6666 6667 6668 6669 \
        7000 8181 32770 32771 32772 32773 32774 32775 32776 32777
32778 32779, \
    ports both 80 81 311 443 465 563 591 593 636 901 989 992 993
994 995 1220 1414 1830 2301 2381 2809 3128 3702 4343 4848 5250 7907
7001 7145 7510 7802 7777 7779 \
        7801 7900 7901 7902 7903 7904 7905 7906 7908 7909 7910 7911
7912 7913 7914 7915 7916 \
        7917 7918 7919 7920 8000 8008 8014 8028 8080 8088 8090 8118
8123 8180 8243 8280 8800 8888 8899 9000 9080 9090 9091 9443 9999
11371 55555
preprocessor stream5_udp: timeout 180

# performance statistics.  For more information, see the Snort
Manual, Configuring Snort - Preprocessors - Performance Monitor
# preprocessor perfmonitor: time 300 file /var/snort/snort.stats
pktcnt 10000
```

```
# HTTP normalization and anomaly detection.  For more information,
see README.http_inspect
preprocessor http_inspect: global iis_unicode_map unicode.map 1252
compress_depth 65535 decompress_depth 65535
preprocessor http_inspect_server: server default \
    http_methods { GET POST PUT SEARCH MKCOL COPY MOVE LOCK UNLOCK
NOTIFY POLL BCOPY BDELETE BMOVE LINK UNLINK OPTIONS HEAD DELETE
TRACE TRACK CONNECT SOURCE SUBSCRIBE UNSUBSCRIBE PROPFIND PROPPATCH
BPROPFIND BPROPPATCH RPC_CONNECT PROXY_SUCCESS BITS_POST CCM_POST
SMS_POST RPC_IN_DATA RPC_OUT_DATA RPC_ECHO_DATA } \
    chunk_length 500000 \
    server_flow_depth 0 \
    client_flow_depth 0 \
    post_depth 65495 \
    oversize_dir_length 500 \
    max_header_length 750 \
    max_headers 100 \
    max_spaces 200 \
    small_chunk_length { 10 5 } \
    ports { 80 81 311 591 593 901 1220 1414 1741 1830 2301 2381
2809 3128 3702 4343 4848 5250 7001 7145 7510 7777 7779 8000 8008
8014 8028 8080 8088 8090 8118 8123 8180 8181 8243 8280 8800 8888
8899 9000 9080 9090 9091 9443 9999 11371 55555 } \
    non_rfc_char { 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 } \
    enable_cookie \
    extended_response_inspection \
    inspect_gzip \
    normalize_utf \
    unlimited_decompress \
    normalize_javascript \
    apache_whitespace no \
    ascii no \
    bare_byte no \
    directory no \
    double_decode no \
    iis_backslash no \
    iis_delimiter no \
    iis_unicode no \
    multi_slash no \
    utf_8 no \
    u_encode yes \
    webroot no

# ONC-RPC normalization and anomaly detection.  For more
information, see the Snort Manual, Configuring Snort -
Preprocessors - RPC Decode
preprocessor rpc_decode: 111 32770 32771 32772 32773 32774 32775
32776 32777 32778 32779 no_alert_multiple_requests
no_alert_large_fragments no_alert_incomplete

# Back Orifice detection.
preprocessor bo

# FTP / Telnet normalization and anomaly detection.  For more
information, see README.ftptelnet
preprocessor ftp_telnet: global inspection_type stateful
encrypted_traffic no check_encrypted
preprocessor ftp_telnet_protocol: telnet \
```

```
    ayt_attack_thresh 20 \
    normalize ports { 23 } \
    detect_anomalies
preprocessor ftp_telnet_protocol: ftp server default \
    def_max_param_len 100 \
    ports { 21 2100 3535 } \
    telnet_cmds yes \
    ignore_telnet_erase_cmds yes \
    ftp_cmds { ABOR ACCT ADAT ALLO APPE AUTH CCC CDUP } \
    ftp_cmds { CEL CLNT CMD CONF CWD DELE ENC EPRT } \
    ftp_cmds { EPSV ESTA ESTP FEAT HELP LANG LIST LPRT } \
    ftp_cmds { LPSV MACB MAIL MDTM MIC MKD MLSD MLST } \
    ftp_cmds { MODE NLST NOOP OPTS PASS PASV PBSZ PORT } \
    ftp_cmds { PROT PWD QUIT REIN REST RETR RMD RNFR } \
    ftp_cmds { RNTO SDUP SITE SIZE SMNT STAT STOR STOU } \
    ftp_cmds { STRU SYST TEST TYPE USER XCUP XCRC XCWD } \
    ftp_cmds { XMAS XMD5 XMKD XPWD XRCP XRMD XRSQ XSEM } \
    ftp_cmds { XSEN XSHA1 XSHA256 } \
    alt_max_param_len 0 { ABOR CCC CDUP ESTA FEAT LPSV NOOP PASV
PWD QUIT REIN STOU SYST XCUP XPWD } \
    alt_max_param_len 200 { ALLO APPE CMD HELP NLST RETR RNFR STOR
STOU XMKD } \
    alt_max_param_len 256 { CWD RNTO } \
    alt_max_param_len 400 { PORT } \
    alt_max_param_len 512 { SIZE } \
    chk_str_fmt { ACCT ADAT ALLO APPE AUTH CEL CLNT CMD } \
    chk_str_fmt { CONF CWD DELE ENC EPRT EPSV ESTP HELP } \
    chk_str_fmt { LANG LIST LPRT MACB MAIL MDTM MIC MKD } \
    chk_str_fmt { MLSD MLST MODE NLST OPTS PASS PBSZ PORT } \
    chk_str_fmt { PROT REST RETR RMD RNFR RNTO SDUP SITE } \
    chk_str_fmt { SIZE SMNT STAT STOR STRU TEST TYPE USER } \
    chk_str_fmt { XCRC XCWD XMAS XMD5 XMKD XRCP XRMD XRSQ } \
    chk_str_fmt { XSEM XSEN XSHA1 XSHA256 } \
    cmd_validity ALLO < int [ char R int ] > \
    cmd_validity EPSV < [ { char 12 | char A char L char L } ] > \
    cmd_validity MACB < string > \
    cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ] string >
\
    cmd_validity MODE < char ASBCZ > \
    cmd_validity PORT < host_port > \
    cmd_validity PROT < char CSEP > \
    cmd_validity STRU < char FRPO [ string ] > \
    cmd_validity TYPE < { char AE [ char NTC ] | char I | char L [
number ] } >
preprocessor ftp_telnet_protocol: ftp client default \
    max_resp_len 256 \
    bounce yes \
    ignore_telnet_erase_cmds yes \
    telnet_cmds yes


# SMTP normalization and anomaly detection.  For more information,
see README.SMTP
preprocessor smtp: ports { 25 465 587 691 } \
    inspection_type stateful \
    b64_decode_depth 0 \
    qp_decode_depth 0 \
    bitenc_decode_depth 0 \
```

```
    uu_decode_depth 0 \
    log_mailfrom \
    log_rcptto \
    log_filename \
    log_email_hdrs \
    normalize_cmds \
    normalize_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO EMAL
ESAM ESND ESOM ETRN EVFY } \
    normalize_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT
RCPT RSET SAML SEND SOML } \
    normalize_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-
ADAT X-DRCP X-ERCP X-EXCH50 } \
    normalize_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50
XGEN XLICENSE XQUE XSTA XTRN XUSR } \
    max_command_line_len 512 \
    max_header_line_len 1000 \
    max_response_line_len 512 \
    alt_max_command_line_len 260 { MAIL } \
    alt_max_command_line_len 300 { RCPT } \
    alt_max_command_line_len 500 { HELP HELO ETRN EHLO } \
    alt_max_command_line_len 255 { EXPN VRFY ATRN SIZE BDAT DEBUG
EMAL ESAM ESND ESOM EVFY IDENT NOOP RSET } \
    alt_max_command_line_len 246 { SEND SAML SOML AUTH TURN ETRN
DATA RSET QUIT ONEX QUEU STARTTLS TICK TIME TURNME VERB X-EXPS X-
LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN XLICENSE XQUE XSTA XTRN
XUSR } \
    valid_cmds { ATRN AUTH BDAT CHUNKING DATA DEBUG EHLO EMAL ESAM
ESND ESOM ETRN EVFY } \
    valid_cmds { EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT
RSET SAML SEND SOML } \
    valid_cmds { STARTTLS TICK TIME TURN TURNME VERB VRFY X-ADAT X-
DRCP X-ERCP X-EXCH50 } \
    valid_cmds { X-EXPS X-LINK2STATE XADR XAUTH XCIR XEXCH50 XGEN
XLICENSE XQUE XSTA XTRN XUSR } \
    xlink2state { enabled }

# Portscan detection.  For more information, see README.sfportscan
# preprocessor sfportscan: proto  { all } memcap { 10000000 }
sense_level { low }

# ARP spoof detection.  For more information, see the Snort Manual
- Configuring Snort - Preprocessors - ARP Spoof Preprocessor
# preprocessor arpspoof
# preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00



# SSH anomaly detection.  For more information, see README.ssh
preprocessor ssh: server_ports { 22 } \
                  autodetect \
                  max_client_bytes 19600 \
                  max_encrypted_packets 20 \
                  max_server_version_len 100 \
                  enable_respoverflow enable_ssh1crc32 \
                  enable_srvoverflow enable_protomismatch
```

```
# SMB / DCE-RPC normalization and anomaly detection.  For more
information, see README.dcerpc2
preprocessor dcerpc2: memcap 102400, events [co ]
preprocessor dcerpc2_server: default, policy WinXP, \
    detect [smb [139,445], tcp 135, udp 135, rpc-over-http-server
593], \
    autodetect [tcp 1025:, udp 1025:, rpc-over-http-server 1025:],
\
    smb_max_chain 3, smb_invalid_shares ["C$", "D$", "ADMIN$"]

# DNS anomaly detection.  For more information, see README.dns
preprocessor dns: ports { 53 } enable_rdata_overflow

# SSL anomaly detection and traffic bypass.  For more information,
see README.ssl
preprocessor ssl: ports { 443 465 563 636 989 992 993 994 995 7801
7802 7900 7901 7902 7903 7904 7905 7906 7907 7908 7909 7910 7911
7912 7913 7914 7915 7916 7917 7918 7919 7920 }, trustservers,
noinspect_encrypted

# SDF sensitive data preprocessor.  For more information see
README.sensitive_data
preprocessor sensitive_data: alert_threshold 25

# SIP Session Initiation Protocol preprocessor.  For more
information see README.sip
preprocessor sip: max_sessions 40000, \
   ports { 5060 5061 5600 }, \
   methods { invite \
             cancel \
             ack \
             bye \
             register \
             options \
             refer \
             subscribe \
             update \
             join \
             info \
             message \
             notify \
             benotify \
             do \
             qauth \
             sprack \
             publish \
             service \
             unsubscribe \
             prack }, \
   max_uri_len 512, \
   max_call_id_len 80, \
   max_requestName_len 20, \
   max_from_len 256, \
   max_to_len 256, \
   max_via_len 1024, \
   max_contact_len 512, \
   max_content_len 2048
```

```
# IMAP preprocessor.  For more information see README.imap
preprocessor imap: \
   ports { 143 } \
   b64_decode_depth 0 \
   qp_decode_depth 0 \
   bitenc_decode_depth 0 \
   uu_decode_depth 0

# POP preprocessor. For more information see README.pop
preprocessor pop: \
   ports { 110 } \
   b64_decode_depth 0 \
   qp_decode_depth 0 \
   bitenc_decode_depth 0 \
   uu_decode_depth 0

# Modbus preprocessor. For more information see README.modbus
preprocessor modbus: ports { 502 }

# DNP3 preprocessor. For more information see README.dnp3
preprocessor dnp3: ports { 20000 } \
   memcap 262144 \
   check_crc

# Reputation preprocessor. For more information see
README.reputation
preprocessor reputation: \
   memcap 500, \
   priority whitelist, \
   nested_ip inner, \
   whitelist $WHITE_LIST_PATH/white_list.rules, \
   blacklist $BLACK_LIST_PATH/black_list.rules

##################################################
# Step #6: Configure output plugins
# For more information, see Snort Manual, Configuring Snort -
Output Modules
##################################################

# unified2
# Recommended for most installs
# output unified2: filename merged.log, limit 128, nostamp,
mpls_event_types, vlan_event_types

# Additional configuration for specific types of installs
# output alert_unified2: filename snort.alert, limit 128, nostamp
# output log_unified2: filename snort.log, limit 128, nostamp
output unified2: filename snort.log, limit 128

# syslog
# output alert_syslog: LOG_AUTH LOG_ALERT

# pcap
# output log_tcpdump: tcpdump.log

# database
output database: log, mysql, user=snort password=mysnort
dbname=snort host=localhost
```

```
# prelude
# output alert_prelude

# metadata reference data.  do not modify these lines
include classification.config
include reference.config


##################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#
# NOTE: All categories are enabled in this conf file
##################################################


# site specific rules
include $RULE_PATH/local.rules

include $RULE_PATH/attack-responses.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/blacklist.rules
include $RULE_PATH/botnet-cnc.rules
include $RULE_PATH/chat.rules
include $RULE_PATH/content-replace.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/file-identify.rules
include $RULE_PATH/file-office.rules
include $RULE_PATH/file-other.rules
include $RULE_PATH/file-pdf.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/icmp.rules
include $RULE_PATH/icmp-info.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/indicator-compromise.rules
include $RULE_PATH/indicator-obfuscation.rules
include $RULE_PATH/info.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/multimedia.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/nntp.rules
include $RULE_PATH/oracle.rules
include $RULE_PATH/other-ids.rules
include $RULE_PATH/p2p.rules
include $RULE_PATH/phishing-spam.rules
include $RULE_PATH/policy.rules
include $RULE_PATH/policy-multimedia.rules
include $RULE_PATH/policy-other.rules
include $RULE_PATH/policy-social.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules
include $RULE_PATH/pua-p2p.rules
```

```
include $RULE_PATH/pua-toolbars.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
include $RULE_PATH/scada.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/server-mail.rules
include $RULE_PATH/shellcode.rules
include $RULE_PATH/smtp.rules
include $RULE_PATH/snmp.rules
include $RULE_PATH/specific-threats.rules
include $RULE_PATH/spyware-put.rules
include $RULE_PATH/sql.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/test.rules
include $RULE_PATH/tftp.rules
include $RULE_PATH/virus.rules
include $RULE_PATH/voip.rules
include $RULE_PATH/vroussos.rules
include $RULE_PATH/web-activex.rules
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-client.rules
include $RULE_PATH/web-coldfusion.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-php.rules
include $RULE_PATH/x11.rules

##################################################
# Step #8: Customize your preprocessor and decoder alerts
# For more information, see README.decoder_preproc_rules
##################################################

# decoder and preprocessor event rules
# include $PREPROC_RULE_PATH/preprocessor.rules
# include $PREPROC_RULE_PATH/decoder.rules
# include $PREPROC_RULE_PATH/sensitive-data.rules

##################################################
# Step #9: Customize your Shared Object Snort Rules
# For more information, see http://vrt-
sourcefire.blogspot.com/2009/01/using-vrt-certified-shared-object-
rules.html
##################################################

# dynamic library rules
# include $SO_RULE_PATH/bad-traffic.rules
# include $SO_RULE_PATH/chat.rules
# include $SO_RULE_PATH/dos.rules
# include $SO_RULE_PATH/exploit.rules
# include $SO_RULE_PATH/icmp.rules
# include $SO_RULE_PATH/imap.rules
# include $SO_RULE_PATH/misc.rules
# include $SO_RULE_PATH/multimedia.rules
# include $SO_RULE_PATH/netbios.rules
# include $SO_RULE_PATH/nntp.rules
# include $SO_RULE_PATH/p2p.rules
```

```
# include $SO_RULE_PATH/smtp.rules
# include $SO_RULE_PATH/snmp.rules
# include $SO_RULE_PATH/specific-threats.rules
# include $SO_RULE_PATH/web-activex.rules
# include $SO_RULE_PATH/web-client.rules
# include $SO_RULE_PATH/web-iis.rules
# include $SO_RULE_PATH/web-misc.rules

# Event thresholding or suppression commands. See threshold.conf
include threshold.conf
```