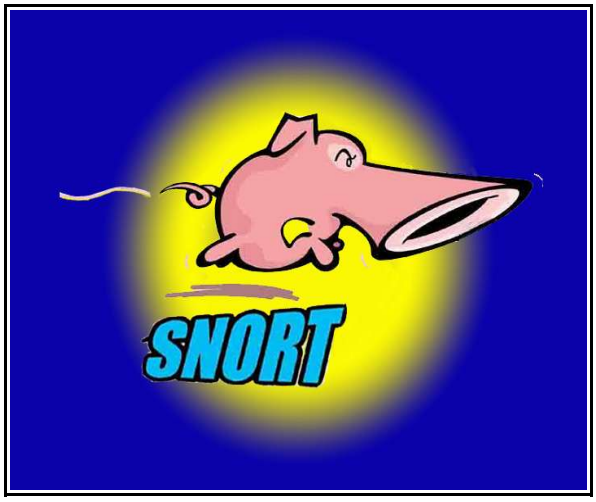
Κεφάλαιο 5

Το NIDS SNORT



Περιεχόμενα

ΕΙΣΑΓΩΓΗ

Στο Κεφάλαιο 2 παρουσιάστηκαν τα Intrusion Detection Systems (IDSs) και αναφέρθηκε η χρησιμότητά τους, ο τρόπος με τον οποίο κατηγοριοποιούνται και ο τρόπος εφαρμογής και χρήσης τους.

Η ανάγνωση του Κεφαλαίου 2 θεωρείται απαραίτητη για την καλύτερη κατανόηση των όσων αναφέρονται σε αυτό το Κεφάλαιο στο οποίο θα γίνει η παρουσίαση του NIDS Snort, ενός πολύ διαδεδομένου εργαλείου για την Ανίχνευση Δικτυακών Επιθέσεων.

Ο λόγος που αυτή η παρουσίαση του Snort γίνεται στο 5ο Κεφάλαιο, είναι για να γίνει πιο εύκολα κατανοητή, η περιγραφή της εφαρμογής που έχει γραφτεί σαν plug-in στο Snort και παρουσιάζεται στο 6ο Κεφάλαιο, η οποία υλοποιεί την Τεχνική Ανίχνευσης του Icmp Echo Spoofing όπως αυτή περιγράφηκε στο Κεφάλαιο 4.

Γενική Περιγραφή του Snort

Το Snort είναι ένα ΄lightweight΄ Network Intrusion Detection System (NIDS), καθώς και ένα εργαλείο ανάλυσης πακέτων το οποίο στηρίζεται στην βιβλιοθήκη libpcap.

Ο δημιουργός του Snort είναι ο Martin Roesch ο οποίος ξεκίνησε την ανάπτυξη του κώδικά σε γλώσσα προγραμματισμού C, ενώ σήμερα ένας μεγάλος αριθμός ατόμων έχει εμπλακεί σε αυτήν την διαδικασία, με στόχο την προσθήκη νέων λειτουργιών στο Snort και την βελτίωση των δυνατοτήτων του.
Σημαντικό ρόλο για το γεγονός αυτό παίζει ότι το Snort είναι ένα Open Source λογισμικό, το οποίο διατίθεται κάτω από την GNU General Public License (GPL) που καθιστά ελεύθερη την χρήση και ανάπτυξη του κώδικά του από τον καθένα.
Ο όρος ΄lightweight΄ έχει δύο έννοιες. Η μία έχει να κάνει με την εφαρμογή του σε σχετικά μικρά δίκτυα, ενώ η δεύτερη έχει να κάνει με το μικρό μέγεθος του και την ευκολία εφαρμογής και χρήσης του.

Μερικές από τις προδιαγραφές του προγράμματος Snort αναφέρονται στον παρακάτω πίνακα:

Προδιαγραφές του Snort	
Μικρό σε μέγεθος	Το μέγεθος του πακέτου Snort με τον πηγαίο κώδικά του δεν ξεπερνάει τα 2MB.
Υποστηρίζεται από πολλές πλατφόρμες.	Το Snort είναι συμβατό με διάφορες πλατφόρμες όπως : Linux, Solaris, Windows, BSD, IRIX, MacOS X, TRU-64, HP-UX κ.α.
Γρήγορο	Το Snort έχει μεγάλη πιθανότητα ανίχνευσης μίας επίθεσης, που μπορεί να συμβαίνει σε ένα δίκτυο με ταχύτητα μετάδοσης δεδομένων στα 100Mbps.

Αρκετά Ρυθμίσιμο	Ο χρήστης μπορεί να ρυθμίσει με μεγάλη ευελιξία τις επιθέσεις που θα ανιχνεύει το Snort, καθώς και τον τρόπο με τον οποίο θα παρουσιάζει τα αποτελέσματά του.
Ελεύθερο Λογισμικό	Είναι ένα Open Source λογισμικό και διατίθεται κάτω από την GNU-GPL άδεια χρήσης, που επιτρέπει την χρήση του σε οποιοδήποτε περιβάλλον και την ελεύθερη ανάπτυξη του κώδικά του.

Πίνακας 5-1: Προδιαγραφές του προγράμματος Snort

Παρόλο που το Snort είναι γνωστό σαν ένα NIDS, στην ουσία δεν είναι μόνο αυτό.
Το Snort χαρακτηρίζεται από τρεις καταστάσεις λειτουργίας (modes), που συνήθως εκτελούνται σε συνδυασμό μεταξύ τους.
Τα modes στα οποία εκτελείται το Snort, ορίζονται με τον ορισμό παραμέτρων από τον χρήστη κατά την εκτέλεση του προγράμματος.
Τα τρία modes λειτουργίας του Snort είναι:

1. Sniffer Mode

Σε αυτό το mode λειτουργίας το Snort έχει την ικανότητα να διαβάζει τα πακέτα καθώς αυτά περνάνε από το δίκτυο, να τα αποκωδικοποιεί και να τα εμφανίζει στην οθόνη σε φιλική προς τον χρήστη μορφή.
Ο χρήστης με διάφορα BPF (Berkley Packet Filter) φίλτρα που μπορεί να χρησιμοποιήσει, έχει την δυνατότητα να ορίσει το είδος των πακέτων που θα εμφανίζονται όσο αναφορά το πρωτόκολλο, τον αποστολέα, τον παραλήπτη και διάφορα άλλα χαρακτηριστικά ενός πακέτου.
Η λειτουργία αυτή του Snort είναι παρόμοια με αυτή του γνωστού εργαλείου tcpdump, το οποίο διατίθεται κυρίως με τα περισσότερα Λειτουργικά Συστήματα της οικογένειας του Unix.

Για παράδειγμα η εντολή `snort -dve` θα εκτελέσει το Snort ώστε να διαβάζει και να εμφανίζει στην οθόνη πλήρη πληροφορία για όλα τα πακέτα που περνάνε από το δίκτυο.
Στο παρακάτω σχήμα παρουσιάζεται ένα πακέτο που έχει καταγραφεί από το Snort μετά την εκτέλεση της παραπάνω εντολής.

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

03/11-11:06:46.387141 0:D0:B7:9E:CF:83 -> 0:50:4:44:52:17 type:0x800 len:0x66
192.168.0.9:1151 -> 192.168.100.25:22 TCP TTL:128 TOS:0x0 ID:8344 IpLen:20 DgmLen:88 DF
***AP*** Seq: 0x2E5A428E  Ack: 0x53D61FCA  Win: 0xF608  TcpLen: 20
EB 7D 11 CA 76 B5 D2 7B 5A 72 AF 93 00 C1 C8 A2   .}..v..{Zr......
D4 EF 39 85 4A A0 C0 B2 9A 12 AB F8 27 47 0C E5   ..9.J.......'G..
1C 20 9A 1D 42 B6 0F 0C 69 EB 40 E8 F0 95 90 D7   . ..B...i.@.....

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

Σχήμα 5-1: Δείγμα ενός πακέτου που καταγράφηκε από το Snort

Το δείγμα αυτό αποτελεί ένα πακέτο που προέρχεται από ένα telnet session.
Στην 1η γραμμή του δείγματος στο **Σχήμα 5-1** υπάρχει η ημερομηνία και ώρα που καταγράφηκε το πακέτο (`03/11-11:06:46.387141`) και άλλες πληροφορίες που αφορούν το Data Link Layer του OSI, όπως οι MAC διευθύνσεις του αποστολέα (`0:D0:B7:9E:CF:83`) και του παραλήπτη (`0:50:4:44:52:17`) του πακέτου και τα πεδία Type (`0x800`) και Length (`0x66`) του Ethernet frame.

Η 2η γραμμή του δείγματος περιέχει πληροφορίες για τα πεδία του IP header του πακέτου, όπως τις IP διευθύνσεις του αποστολέα (`192.168.0.9`) και του παραλήπτη (`192.168.100.25`) με τις πόρτες πηγής (`1151`) και προορισμού (`22`) του πακέτου, τα πεδία Protocol Type (`TCP`) , Time To Live (`128`), Type Of Service (`0x0`), Identification Number (`8344`), Header Length (`20`), Datagram Total Length (`88`) και τα Flags (`DF`) του IP header.

Στην 3η γραμμή του δείγματος υπάρχουν οι πληροφορίες για τον TCP header του πακέτου, όπως τα TCP Flags (`AP`), το Sequence Number (`0x2E5A428E`), το Acknowledgement Number (`0x53D61FCA`), το Window (`0xF608`), και το TCP Header Length (`20`).
Μέρος της πληροφορίας του TCP header είναι και οι πόρτες πηγής και προορισμού του πακέτου οι οποίες για λόγους συνοχής εμφανίζονται στην 2η γραμμή δίπλα από τις αντίστοιχες IP διευθύνσεις.

Στο τελευταίο κομμάτι του δείγματος εμφανίζεται στην δεξιά στήλη το payload του πακέτου σε δεκαεξαδική μορφή και στην αριστερή στήλη το payload σε ASCII μορφή που το καθιστά κατανοητό στο διάβασμα.

2. Packet Logger Mode

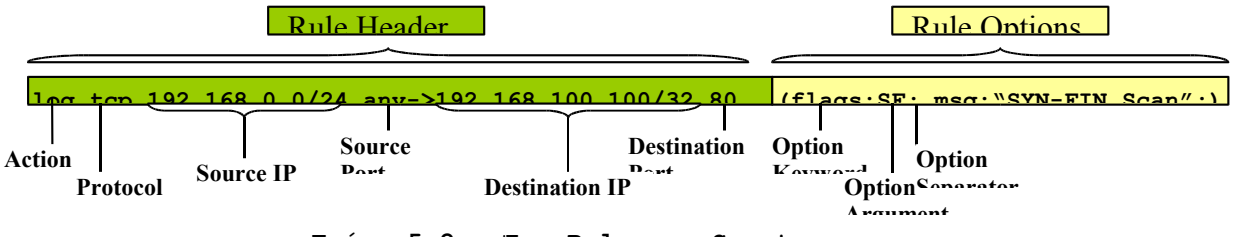
Σε αυτό το mode λειτουργίας το Snort αποθηκεύει στο δίσκο τα πακέτα που διαβάζει από το δίκτυο, αντί απλά να τα εμφανίζει στην οθόνη. Η διαδικασία αυτή είναι αρκετά σημαντική στην περίπτωση που απαιτείται τα πακέτα αυτά να εξεταστούν με λεπτομέρεια σε επόμενο στάδιο.
Το Snort μπορεί να αποθηκεύσει τα πακέτα αυτά σε διάφορα formats, ανάλογα με τις ανάγκες του χρήστη. Για παράδειγμα μπορεί να αποθηκεύσει τα πακέτα σε binary μορφή (tcpdump format), με την οποία μπορούν να χρησιμοποιηθούν σαν είσοδο σε διάφορα άλλα προγράμματα ανάλυσης πακέτων και πρωτοκόλλων, σε ASCII μορφή ώστε να είναι δυνατή η ανάγνωσή τους, σε XML μορφή ή και να οργανωθούν σε βάσεις δεδομένων.
Για να λειτουργήσει σε αυτό το mode το Snort θα πρέπει να εκτελεστεί με την παράμετρο `-l`. Δηλαδή `snort -l <logfile>` όπου το `logfile` είναι το αρχείο στο οποίο θα καταγραφούν τα πακέτα.

3. NIDS Mode

Αυτή είναι η κύρια λειτουργία του Snort και ο λόγος για τον οποίο παρουσιάζεται σε αυτό το Κεφάλαιο. Όπως αναφέρθηκε προηγουμένως, το Snort είναι ένα IDS το οποίο ενεργεί σε επίπεδο δικτύου, δηλαδή τα γεγονότα που παρακολουθεί και εξετάζει για την εμφάνιση μίας πιθανής επίθεσης, αφορούν την δραστηριότητα που παρατηρείται σε ένα δίκτυο.
Το Snort έχει την ικανότητα να ανιχνεύει ένα μεγάλο φάσμα από γνωστές δικτυακές επιθέσεις, όπως portscans, buffer overflows, OS fingerprints κ.α.
Η τεχνική που χρησιμοποιεί το Snort για την διαδικασία αυτή είναι κατά κύριο λόγο η *Misuse Detection* με την χρήση των *Signatures*. Παρόλα αυτά το Snort συνδυάζει στην λειτουργία της ανάλυσης των γεγονότων για την ανίχνευση πιθανών επιθέσεων και κάποιες από τις μεθόδους του *Protocol Anomaly Detection* και του *Anomaly Detection*.

Snort Signatures

Τα *Signatures* του Snort ονομάζονται και *Rules,* καθώς είναι κανόνες οι οποίοι περιγράφουν τα χαρακτηριστικά ενός πακέτου που μπορεί να είναι μέρος μίας γνωστής επίθεσης, καθώς και την ενέργεια που θα εκτελεστεί κατά τον εντοπισμό του.
Κάθε πακέτο που εντοπίζεται από το Snort, ελέγχεται για το αν έχει τα ίδια χαρακτηριστικά με αυτά που περιγράφονται από κάποιο *Rule*.
Τα *Rules* του Snort μπορούν να γραφτούν σε απλή περιγραφική γλώσσα σε ASCII μορφή και κάθε ένα από αυτά αποτελείται από δύο λογικά μέρη, τον *Rule Header* και τα *Rule Options*.
Στο Σχήμα που ακολουθεί περιγράφεται ένα *Rule*.



Ο *Rule Header* περιέχει τις εξής πληροφορίες:

- ***Action:*** Είναι η ενέργεια που θα εκτελέσει το Snort όταν ταιριάξει κάποιο πακέτο με ένα *Rule*. Η ενέργεια αυτή έχει να κάνει με την αντίδραση (Response) του Snort κατά την ανίχνευση μίας πιθανής επίθεσης.

  Το *Action* μπορεί να είναι ένα από τα πέντε:

  - ***Alert,*** το οποίο θα δημιουργήσει ένα *alert* για το γεγονός που εντόπισε και στη συνέχεια θα καταγράψει το πακέτο. Τα *alerts* είναι ο τρόπος με τον οποίο το Snort επισημαίνει το γεγονός της ανίχνευσης μίας επίθεσης και θα παρουσιαστούν παρακάτω.
  - ***Log,*** το οποίο θα καταγράψει το πακέτο στον δίσκο.
  - ***Pass,*** το οποίο θα αγνοήσει το πακέτο.
  - ***Activate,*** το οποίο θα προκαλέσει ένα *alert* και στη συνέχεια θα ενεργοποιήσει ένα *dynamic Rule*.
  - ***Dynamic,*** το οποίο θα περιμένει μέχρι να ενεργοποιηθεί από ένα *activate Rule* και στη συνέχεια θα ενεργήσει σαν ένα *log Rule*.

Ο χρήστης έχει την δυνατότητα να ορίσει και δικούς του τύπους από *Actions*.

- ***Protocol:*** Είναι το είδος του πρωτοκόλλου στο οποίο ανήκει το πακέτο που θα εξεταστεί. Το πρωτόκολλο μπορεί να είναι ip, tcp, icmp, udp.

- ***Source IP:*** Είναι η IP διεύθυνση αποστολέα που βρίσκεται στον IP header του πακέτου, σε συνδυασμό με την μάσκα του δικτύου (netmask) στο οποίο μπορεί να ανήκει, εκφρασμένη με CIDR τρόπο γραφής. Με τον CIDR τρόπο γραφής γίνεται δυνατό να ορισθεί μία ομάδα (block) από συνεχείς IP διευθύνσεις.

Στο συγκεκριμένο παράδειγμα η IP διεύθυνση αποστολέα μπορεί να ανήκει σε μία από τις 198.168.0.0 έως 198.168.0.255.
Στο πεδίο αυτό μπορεί να χρησιμοποιηθεί και το λεκτικό *any* που θα σημαίνει οποιαδήποτε IP διεύθυνση.

- ***Source Port:*** Είναι η πόρτα αποστολής που έχει νόημα στα tcp και udp πακέτα. Στο συγκεκριμένο παράδειγμα με το λεκτικό *any* εννοείται οποιαδήποτε πόρτα.

- ***Destination IP:*** Είναι η IP διεύθυνση του παραλήπτη του πακέτου. Ο τρόπος γραφής της είναι ο ίδιος με αυτόν που ισχύει για την Source IP και στο συγκεκριμένο παράδειγμα η IP διεύθυνση του παραλήπτη με την netmask που έχει ορισθεί, αντιπροσωπεύει έναν μόνο αριθμό τον 198.168.100.100.

- ***Destination Port:*** Είναι η πόρτα προορισμού του πακέτου.

Τα *Rule Options* περιέχουν πληροφορία που αναφέρεται στα χαρακτηριστικά για τα οποία θα ελεγχθεί το πακέτο. Επίσης στα *Rule Options* μπορούν να ορισθούν και κάποιες επιπρόσθετες ενέργειες που θα εκτελεστούν για κάποιο πακέτο που θα ταιριάξει με το *Rule*. Το κάθε option που περιέχεται στο *Rule Options* τμήμα του *Rule,* αποτελείται από τα *Option Keyword* και τα *Option Arguments*.

Option Keyword: Είναι το λεκτικό που υποδηλώνει το όνομα-είδος του option.

Option Argument: Είναι οι παράμετροι που δέχεται το option σε σχέση με τις οποίες θα ελεγχθεί το πακέτο. Τα *Option Arguments* για κάθε option, πρέπει να διαχωρίζονται με τον χαρακτήρα ΄:΄ από το αντίστοιχο *Option Keyword*.

Option Separator: Είναι ο χαρακτήρας διαχωρισμού ΄;΄ μεταξύ δύο options.

Το συγκεκριμένο *Rule* έχει δύο options. Το πρώτο είναι το option με το όνομα *flags,* το οποίο υποδηλώνει ότι θα ελεγχθούν τα flags του TCP header του πακέτου. Τα *arguments* αυτού του option είναι τα SF, το οποίο δηλώνει ότι θα ελεγχθεί αν ο TCP header του πακέτου έχει ενεργοποιημένα τα flags Syn και Fin.
Το δεύτερο option έχει το όνομα *msg,* το οποίο δηλώνει ότι αν ταιριάξει κάποιο πακέτο με αυτό το *Rule,* τότε μαζί με το *alert* (ή το πακέτο που θα καταγραφεί) θα τυπωθεί και κάποιο μήνυμα, το οποίο είναι αυτό που ακολουθεί μέσα στα ΄΄ ΄΄ και το οποίο αποτελεί το *Option Argument* αυτού του option.

Το Snort διανέμεται με πάνω από 2500 έτοιμα *Signatures,* για χρήση τους στην ανίχνευση γνωστών επιθέσεων, ενώ για την δημιουργία νέων *Rules* προσφέρει μία μεγάλη γκάμα από options που μπορεί ο χρήστης να χρησιμοποιήσει, τα οποία του δίνουν την ευελιξία να εκτελεί λεπτομερής και σε βάθος περιγραφή των χαρακτηριστικών του κάθε πακέτου, για το οποίο θέλει να γίνει έλεγχος για τον εντοπισμό μίας επίθεσης.

Όταν το Snort ανιχνεύσει μία επίθεση έχει την δυνατότητα να γνωστοποιήσει τα αποτελέσματά του με διάφορους τρόπους, με την μορφή *alerts*.
Κάποιοι από αυτούς είναι, σε πραγματικό χρόνο με την χρήση αναδυόμενων παραθύρων στην οθόνη, σε ASCII μορφή στην κονσόλα, να τα αποθηκεύσει σε αρχεία σε ASCII μορφή για μετέπειτα ανάγνωση ή και να τα οργανώσει σε βάσεις δεδομένων όπως MySQL, PostgreSQL, Oracle, unixODBC, κ.α.

Ένα *alert* μπορεί να προσφέρει λίγες πληροφορίες για μία επίθεση, όπως το είδος της, τον επιτιθέμενο και τον στόχο, μέχρι και αναλυτική περιγραφή του πακέτου που οδήγησε την ανίχνευση της επίθεσης. Στο παρακάτω σχήμα παρουσιάζεται ένα *alert* του Snort.

```
[**] [1:553:4] POLICY FTP anonymous login attempt [**]
[Classification: Misc activity] [Priority: 3]
03/11-12:23:37.280737 192.168.0.9:1245 -> 192.168.100.25:21
TCP TTL:128 TOS:0x0 ID:13318 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0x74603DEF  Ack: 0x7B16BDCC  Win: 0xFAB2  TcpLen: 20
```

Σχήμα 5-3: Ένα alert του Snort

Το *alert* αυτό δημιουργήθηκε από το Snort καθώς εντόπισε μία προσπάθεια για σύνδεση ενός χρήστη σαν anonymous, στον ftp server με την IP διεύθυνση 192.168.100.25.
Κάποιοι ftp servers επιτρέπουν την χρήση τους από χρήστες που δεν έχουν έναν εξουσιοδοτημένο λογαριασμό, αν αυτοί συνδεθούν με το username anonymous και δώσουν για password την mail διεύθυνσή τους.
Αυτό όμως μπορεί να δημιουργήσει τρύπες ασφάλειας σε έναν ftp server και για αυτό κάποιοι δεν επιτρέπουν την χρήση του κοινού λογαριασμού anonymous.
Στην περίπτωση που ο συγκεκριμένος ftp server δεν επιτρέπει σε χρήστες χωρίς κάποιο εξουσιοδοτημένο προσωπικό λογαριασμό να συνδεθούν σε αυτόν, τότε αυτό το *alert* μπορεί να είναι μία ένδειξη για προσπάθεια παραβίασης του συστήματος.
Στο δείγμα του *alert* που παρουσιάζεται στο **Σχήμα 5-3**, αναφέρεται η ώρα που ανιχνεύτηκε το γεγονός καθώς και το είδος της επίθεσης που εντοπίστηκε, ενώ ακολουθεί περιγραφή των χαρακτηριστικών του πακέτου που οδήγησε στην δημιουργία του *alert*.

Για να εκτελεστεί το Snort σε NIDS Mode θα πρέπει να δοθεί η παράμετρος `-c` , τιμή της οποίας θα είναι το *configuration* αρχείο του Snort (συνήθως το `snort.conf`), το οποίο στην ουσία περιέχει όλες τις ρυθμίσεις που θα καθορίσουν τον τρόπο εκτέλεσης και λειτουργίας του Snort.

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

Η Δομή Του Snort

Το Snort αποτελείται από τέσσερα υποσυστήματα λειτουργίας. Κάθε πακέτο που επεξεργάζεται το Snort, θα περάσει από κάθε ένα από αυτά τα υποσυστήματα τα οποία περιγράφονται παρακάτω:

A. Decode Engine

B. Detection Engine
C. Output Engine
D. Plug-ins

A. Decode Engine

Το υποσύστημα αυτό του Snort είναι το πρώτο που λαμβάνει μέρος στην επεξεργασία των πακέτων που εντοπίζει το Snort.
Η εργασία του *Decode Engine* ξεκινάει με το διάβασμα (sniffing) των πακέτων από το δίκτυο με την χρήση της βιβλιοθήκης libpcap, η οποία δίνει την δυνατότητα της μεταφοράς των πακέτων από επίπεδο πυρήνα του συστήματος σε επίπεδο χρήστη, από όπου θα μπορέσει το Snort να τα επεξεργαστεί.
Στην συνέχεια το Snort εκτελεί την αποκωδικοποίησή του κάθε πακέτου και την οργάνωση των πληροφοριών που το αφορούν σε μία δομή δεδομένων, την `struct Packet`.

Το *Decode Engine* του Snort υλοποιείται σχεδόν αποκλειστικά στο αρχείο `decode.c,` το οποίο περιέχει συναρτήσεις που εξάγουν την απαιτούμενη πληροφορία για το κάθε πακέτο, ξεκινώντας από τα πρωτόκολλα του Data Link Layer του OSI ανεβαίνοντας μέχρι και τα πρωτόκολλα του Application Layer. Για κάθε πρωτόκολλο υπάρχει και η αντίστοιχη συνάρτηση που θα ενεργήσει στο πακέτο.

Την παρούσα στιγμή το Snort διατίθεται με την έκδοση 1.9.1 και τα πρωτόκολλα που έχει την δυνατότητα να επεξεργάζεται είναι :

- ICMP, TCP, UDP
- IP, ARP
- Ethernet, FDDI, T/R, SLIP, PPP, ISDN, Raw
-

Η έκδοση 2.0 του Snort θα υποστηρίζει και την νέα έκδοση του IP το IPv6.

Μετά την φάση της αποκωδικοποίησης, οι πληροφορίες που αφορούν το κάθε πακέτο θα έχουν αποθηκευτεί στην `struct Packet` την οποία το Snort θα χρησιμοποιεί για την παραπέρα επεξεργασία του κάθε πακέτου.
Στο επόμενο στάδιο το κάθε πακέτο προωθείται στο *Detection Engine*.

B. Detection Engine

Το *Detection Engine* είναι αυτό που εξετάζει το κάθε πακέτο και το ελέγχει σε σχέση με τα *Rules* του Snort ώστε να αποφανθεί αν αυτό ανήκει σε μία επίθεση.

Το *Detection Engine* του Snort υλοποιείται στο αρχείο `rules.c` και η λειτουργία του χωρίζεται σε δύο φάσεις.

Η πρώτη φάση εκτελείται κατά την εκκίνηση του Snort και έχει σαν στόχο να διαβάσει τα *Rules* και να τα οργανώσει με τέτοιο τρόπο ώστε να είναι δυνατή η παραπέρα επεξεργασία τους. Τα *Rules* του Snort όπως αναφέρθηκε παραπάνω είναι γραμμένα σε ASCII μορφή μοιρασμένα σε διάφορα text αρχεία.

Κατά την εκκίνηση του Snort, το *Detection Engine* μέσω της συνάρτησης `ParseRulesFile()` θα διαβάσει τα *Rules* από το κάθε αρχείο που παίρνει σαν παράμετρο.
Στην ουσία η συνάρτηση αυτή θα διαβάσει το αρχείο γραμμή προς γραμμή, θα απαλείψει τα σχόλια και τα κενά που μπορεί να υπάρχουν και για κάθε *Rule* που θα εντοπίσει θα καλέσει την συνάρτηση `ParseRule()`.
Η συνάρτηση αυτή θα προετοιμάσει κατάλληλα κάποιο *Rule*, ώστε με κλήσεις που θα κάνει σε διάφορες άλλες συναρτήσεις, να το προσθέσει σε μία δυναμική συνδεδεμένη λίστα.

Οι δυναμικές συνδεδεμένες λίστες είναι ο τρόπος με τον οποίο το Snort αποθηκεύει και οργανώνει τα *Rules,* ώστε να μπορεί στην συνέχεια με διάφορες αναζητήσεις στις λίστες αυτές να εκτελέσει την διαδικασία της ανίχνευσης.

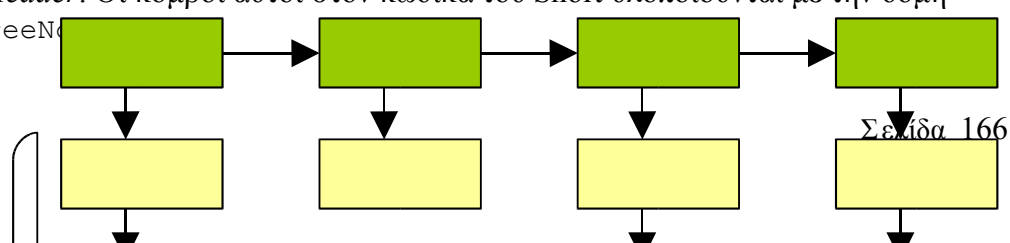
Το Snort διατηρεί πέντε ξεχωριστές αλυσίδες (*chains*) από *Rules,* που στην ουσία η κάθε μία αντιστοιχεί και στην ενέργεια (*Action*) που μπορεί να οριστεί στον *Rule Header*.
Οι αλυσίδες αυτές είναι οι *Alert*, *Log*, *Pass*, *Activation*, *Dynamic*.
Έτσι ανάλογα με το *Action* που ορίζεται για κάποιο *Rule* στον *Rule Header*, αυτό θα καταχωρηθεί στο αντίστοιχο *Chain*.

Κάθε *Chain* διατηρεί τρεις ξεχωριστές δυναμικές συνδεδεμένες λίστες, οπού κάθε μία από αυτές αντιστοιχεί και σε κάθε ένα από τα πρωτόκολλα TCP, UDP, ICMP.
Έτσι κάθε *Rule* θα αποθηκευτεί σε μία από αυτές τις τρεις λίστες, ανάλογα με το πρωτόκολλο (*Protocol*) το οποίο ορίζεται στον *Rule Header*.

Κάθε μία από αυτές τις λίστες είναι μία τρισδιάστατη λίστα, της οποίας η $1^{η}$ και $2^{η}$ διάσταση, αποτελούνται από κόμβους στους οποίους αποθηκεύονται αντίστοιχα ο *Rule Header* και τα *Rule Options* του κάθε *Rule*, ενώ η $3^{η}$ διάσταση αποτελείται από συνδεδεμένες λίστες που περιέχουν τις συναρτήσεις που θα εκτελεστούν για την εξέταση του πακέτου.
Στο **Σχήμα 5-4** παρουσιάζεται η μορφή που μπορεί να έχει μία από τις λίστες που αναφέρθηκε παραπάνω.

Rule Nodes

Οι πράσινοι κόμβοι στο σχήμα αποτελούν τους κόμβους που περιέχουν την πληροφορία που αφορά τον *Rule Header*. Οι κόμβοι αυτοί στον κώδικα του Snort υλοποιούνται με την δομή `struct RuleTreeN`



Σχήμα 5-4: Η Λίστα με τα Rules

Σε κάθε τέτοια δομή θα καταχωρηθεί η πληροφορία που αναγράφεται στον *Rule Header* καθώς και κάποιοι δείκτες (pointers) οι οποίοι χρησιμοποιούνται για την διασύνδεση των διαφόρων κόμβων μεταξύ τους. Υπάρχει ένας pointer ο οποίος δείχνει στον επόμενο *Rule Node*, ένας που δείχνει στον 1ο *Rule Node*, ένας pointer που δείχνει στον 1ο *Options Node* της 2ης διάστασης και ένας pointer ο οποίος δείχνει στην αρχή της λίστας της 3ης διάστασης που περιέχει τις συναρτήσεις που θα χρησιμοποιηθούν για τον έλεγχο του πακέτου όσο αναφορά τον *Rule Header*.

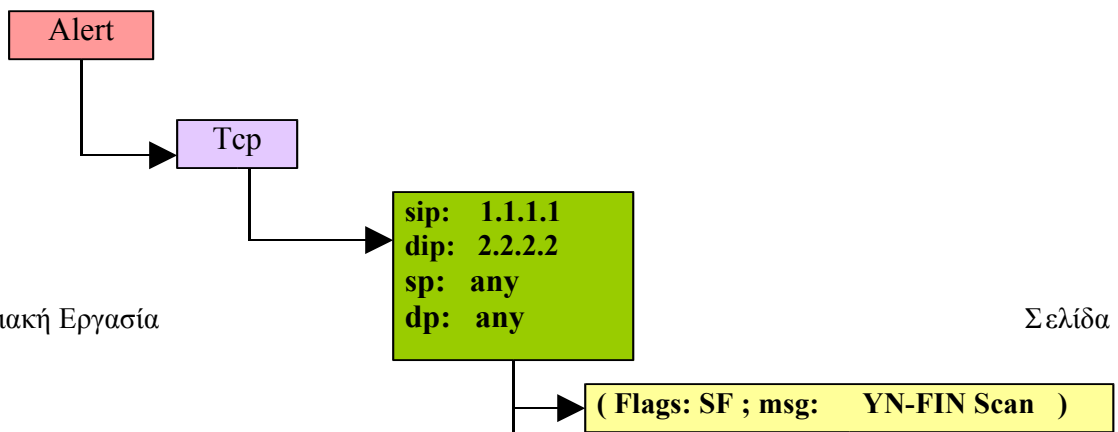
Οι κίτρινοι κόμβοι στο σχήμα αποτελούν τους κόμβους που περιέχουν την πληροφορία που αφορά τα *Rule Options*. Οι κόμβοι αυτοί στον κώδικα του Snort υλοποιούνται με την δομή `struct OptTreeNode`.
Σε κάθε τέτοια δομή θα καταχωρηθεί η πληροφορία που αναγράφεται στα *Rule Options* καθώς και κάποιοι δείκτες (pointers) οι οποίοι χρησιμοποιούνται για την διασύνδεση των διαφόρων κόμβων μεταξύ τους. Υπάρχει ένας pointer ο οποίος δείχνει στον επόμενο *Options Node*, ένας που δείχνει στον *Rule Node* κάτω από τον οποίο συνδέεται η λίστα στην οποία ανήκει αυτός ο κόμβος και ένας pointer που δείχνει στην αρχή της λίστας της 3ης διάστασης που περιέχει τις συναρτήσεις που θα χρησιμοποιηθούν για τον έλεγχο του πακέτου όσο αναφορά τα *Rule Options*.
Οι συναρτήσεις αυτές μπορεί να είναι μέρος των *Detection Plug-ins* του Snort που θα παρουσιαστούν πιο μετά.

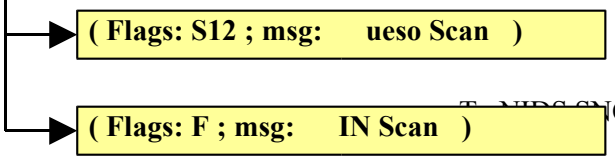
Για να γίνει καλύτερα κατανοητό πώς οργανώνονται τα *Rules* σε μία λίστα δίνεται ένα παράδειγμα.

```
Alert tcp 1.1.1.1 any -> 2.2.2.2 any ( Flags:SF; msg: "SYN-FIN Scan";)

Alert tcp 1.1.1.1 any -> 2.2.2.2 any ( Flags:S12; msg:   ueso Scan)

Alert tcp 1.1.1.1 any -> 2.2.2.2 any ( Flags:F; msg:   IN Scan)
```

Σχήμα 5-5: Τρία Rules Του Snort

Στο **Σχήμα 5-5**, παρουσιάζονται τρία *Rules* τα οποία ελέγχουν τα flags του TCP Header των πακέτων για να εντοπίσουν διάφορα Portscans που μπορεί να γίνονται από την IP διεύθυνση 1.1.1.1 στην 2.2.2.2 .
Τα *Rules* αυτά θα καταχωρηθούν στο *Alert Chain,* στην συνδεδεμένη λίστα που αφορά το TCP πρωτόκολλο, με τον τρόπο που φαίνεται στο **Σχήμα 5-6**.





Σχήμα 5-6: Ένα δείγμα του Alert Chain

Η δεύτερη φάση λειτουργίας του *Detection Engine* εκτελείται κάθε φορά που το Snort εντοπίζει ένα πακέτο και αφού αυτό περάσει από το *Decode Engine*. Σε αυτή την φάση λειτουργίας το *Detection Engine* θα ελέγξει το πακέτο σε σχέση με τα καταχωρημένα *Rules,* με στόχο να εντοπίσει αν αυτό ανήκει σε κάποια επίθεση, όπως τα *Rules* υποδεικνύουν.
Ο τρόπος της οργάνωσης των *Rules* που παρουσιάστηκε παραπάνω εξυπηρετεί στην βελτιστοποίηση της ταχύτητας με την οποία γίνεται η αναζήτηση στις λίστες και ο έλεγχος του κάθε πακέτου σε σχέση με τα *Rules* προς την ανίχνευση μίας επίθεσης.
Το *Detection Engine* ξεκινάει την αναζήτηση ελέγχοντας με την σειρά τα *Rules* που είναι αποθηκευμένα σε κάθε *Chain*.
Η σειρά με την οποία ελέγχονται τα *Chains* είναι

Alert -> Log -> Pass ->Activation ->Dynamic.

Αν κατά την αναζήτηση δεν βρεθεί κάποιο *Rule* που να ταιριάζει με τα χαρακτηριστικά του πακέτου, τότε το Snort θα αγνοήσει αυτό το πακέτο, το οποίο θα συνεχίσει την πορεία του στο δίκτυο.
Αν κατά την διάρκεια της αναζήτησης βρεθεί κάποιο *Rule* που ταιριάζει στο πακέτο που ελέγχεται, τότε η αναζήτηση σταματά και δεν συνεχίζεται στα επόμενα *Chains*.
Αυτό που μένει πλέον είναι να εκτελεστεί η ενέργεια (*Action*) που έχει οριστεί στο *Rule*.

Αφού το πακέτο ελεγχθεί ως προς τα *Rules* από το *Detection Engine,* θα προωθηθεί στο επόμενο υποσύστημα, το *Output Engine*.

C. Output Engine

Το *Output Engine* υλοποιείται στο αρχείο `log.c` του κώδικα του Snort και είναι αυτό που θα εκτελέσει την διαδικασία της γνωστοποίησης των αποτελεσμάτων του Snort.
Το *Output Engine* με διάφορες συναρτήσεις στο αρχείο `log.c,` υλοποιεί τις διαδικασίες του *Log* και του *Alert*, δηλαδή της αποθήκευσης ενός πακέτου σε διάφορα formats και την επισήμανση των *alerts* που παράγει το Snort.
Το *Output Engine* για να εκτελέσει τις λειτουργίες του, δεν δέχεται είσοδο μόνο από το *Detection Engine* άλλα και από άλλα υποσυστήματα του Snort.
Οι συναρτήσεις του αρχείου `log.c` θεωρούνται πλέον ξεπερασμένες και έχουν αντικατασταθεί από διάφορα *Output Plug-ins* τα οποία επιτελούν το μεγαλύτερο μέρος της παρουσίασης των αποτελεσμάτων του Snort. Τα *Output Plug-ins* θα παρουσιαστούν στην συνέχεια.

D. Plug-ins

Τα *plug-ins* είναι κομμάτια κώδικα, με την λογική ξεχωριστών υποπρογραμμάτων που ενσωματώνονται στον υπόλοιπο κώδικα του Snort. Τα *plug-ins* επιτρέπουν την επέκταση των δυνατοτήτων του Snort χωρίς να χρειάζεται αλλαγή το κύριο σώμα του κώδικά του. Το γεγονός ότι το Snort είναι ένα Open Source λογισμικό, επιτρέπει στον οποιονδήποτε, εκμεταλλευόμενος τις δυνατότητες του Snort στον χειρισμό και την επεξεργασία των πακέτων, να δημιουργήσει *plug-ins,* που μπορεί να εκτελούν από απλές λειτουργίες υπολογισμού στατιστικών, μέχρι και πολύπλοκες τεχνικές ανίχνευσης επιθέσεων.
Το Snort αυτή τη στιγμή υποστηρίζει την δημιουργία και ενσωμάτωση στον κώδικά του τριών ειδών *plug-ins*: τα ***preprocessor plug-ins***, τα ***detection plug-ins*** και τα ***output plug-ins***.
Ο χειρισμός των *plug-ins* και η συνεργασία τους με το υπόλοιπο πρόγραμμα του Snort, επιτυγχάνεται μέσα από το αρχείο `plugbase.c` του κώδικα του Snort.

D.1 Preprocessor Plug-ins

Οι *preprocessors* είναι *plug-ins* του Snort που εκτελούνται μετά το *Decode Engine* και πριν το *Detection Engine*. Οι *preprocessors* 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 αφορούν τα πακέτα που επεξεργάζεται το Snort, είτε με την κατάλληλη προετοιμασία των πακέτων πριν αυτά προωθηθούν στο *Detection Engine*.
Τα πηγαία αρχεία που περιέχουν τον κώδικα του κάθε *preprocessor,* ξεκινάνε με το λεκτικό `spp_` και ακολουθεί το όνομα του *preprocessor*.
Μερικοί από τους πιο σημαντικούς *preprocessors* είναι ο `spp_portscan` ο οποίος ανιχνεύει τα portscans, ο `spp_frag2` ο οποίος εκτελεί την επανασυγκρότηση τυχόν κατακερματισμένων IP πακέτων, ο `spp_stream4` ο οποίος ελέγχει την ορθότητα των TCP πακέτων όσο αναφορά την συμμετοχή τους σε ένα συγκεκριμένο stream και ανιχνεύει επιθέσεις που στηρίζονται στην μη φυσιολογική χρήση τους.

D.2 Detection Plug-ins

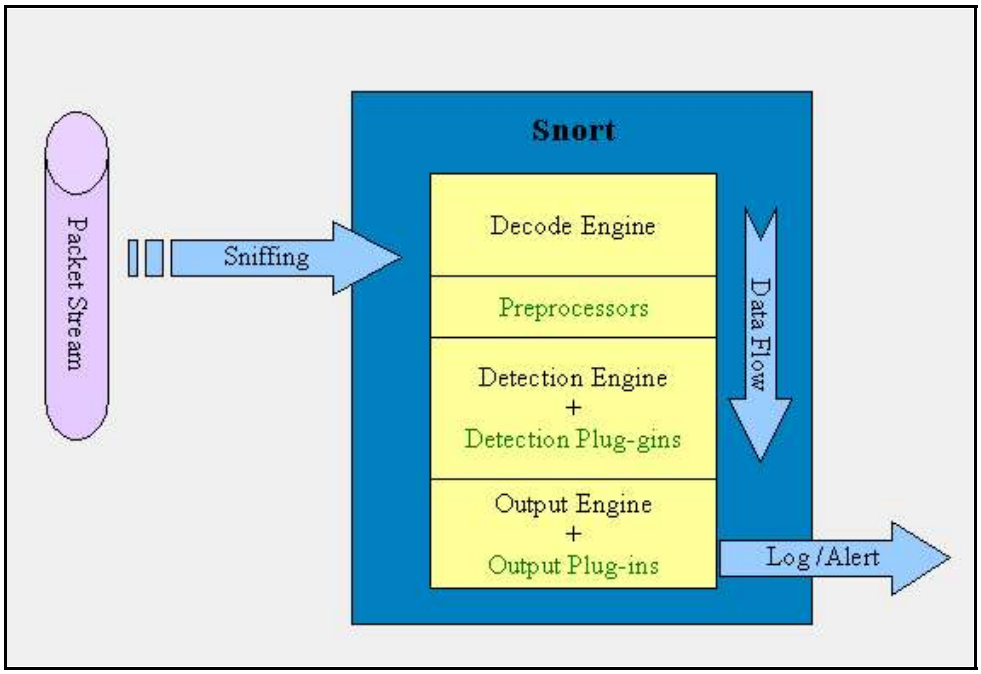
Τα *Detection plug-ins* αυξάνουν την λειτουργικότητα του *Detection Engine* του Snort και εκτελούν διάφορες εργασίες ελέγχου ενός πακέτου σε σχέση με τα *Rules*. Ένα *Detection plug-in* μπορεί να εκτελεστεί αρκετές φορές για κάθε πακέτο και τα πηγαία αρχεία που τα αποτελούν ξεκινάνε με το λεκτικό *sp_* και ακολουθούνται από το όνομα του *plug-in*.
Τα *Detection Plug-ins* με τον καιρό αντικατέστησαν αρκετές από τις συναρτήσεις του αρχείου `rules.c` και μερικά από αυτά είναι το `sp_ip_id_check` το οποίο ελέγχει το πεδίο id του IP header, το `sp_icmp_type_check` το οποίο ελέγχει το πεδίο type του Icmp header κ.α.

D.3 Output Plug-ins

Τα *Output Plug-ins* αυξάνουν την λειτουργικότητα του *Output Engine* του Snort, εκτελώντας εργασίες που έχουν να κάνουν με το *Alert* και το *Log*.
Στην ουσία τα *Output Plug-ins* έχουν αντικαταστήσει ένα μεγάλο μέρος των συναρτήσεων του αρχείου `log.c` του Snort, το οποίο παλιότερα εκτελούσε εξ 'ολοκλήρου τις λειτουργίες του *Output Engine*.
Τα *Output plug-ins* ξεκινάνε με το λεκτικό `spo_` και ακολουθούνται από το όνομα του *plug-in*.
Μερικά γνωστά *plug-ins* τέτοιου είδους είναι το `spo_alert_full` που καταγράφει σε ASCII

μορφή κάθε *alert* που παράγει το Snort, με μεγάλη λεπτομέρεια όσο αναφορά το πακέτο που οδήγησε στην δημιουργία του, το `spo_alert_fast` το οποίο καταγράφει τα *alerts* με πιο συνοπτικό τρόπο και το `spo_log_tcpdump` το οποίο αποθηκεύει σε binary μορφή τα πακέτα που περνάνε από το δίκτυο.

Το **Σχήμα 5-7** που ακολουθεί παρουσιάζει την δομή του Snort, όσο αναφορά τα υποσυστήματα από τα οποία αποτελείται και αναπαριστά την διαδρομή που ακολουθεί κάθε πακέτο κατά την επεξεργασία του από το Snort.



Σχήμα 5-7: Το Engine του Snort

ΕΠΙΛΟΓΟΣ

Η συνεχής ανάπτυξη και βελτίωση του Snort το έχουν καταστήσει ένα από τα πιο δημοφιλή και αποτελεσματικά IDSs σήμερα.
Σημαντικό ρόλο σε αυτό έχει παίξει ο μηχανισμός με τον οποίο το Snort ανιχνεύει πιθανές επιθέσεις, καθώς τα *Signatures* που χρησιμοποιεί ανανεώνονται διαρκώς ώστε να καλύπτουν νέες επιθέσεις, μέσα σε μικρό χρονικό διάστημα από την στιγμή που αυτές θα εμφανιστούν.
Σε αυτό βοηθάει επίσης η ευκολία και η ευελιξία που παρουσιάζει η συγγραφή νέων *Rules* από τον καθένα, αρκεί κάποιος να γνωρίζει τα χαρακτηριστικά μίας επίθεσης όσο αναφορά τα πακέτα που την υλοποιούν και να μπορεί να τα περιγράψει σε απλή ASCII μορφή.

Ένα μεγάλο μέρος της επιτυχίας του Snort, οφείλεται και στο γεγονός ότι είναι ένα Open Source λογισμικό, κάτι που έχει οδηγήσει στην ευρεία χρήση του και στην συμμετοχή πολλών ενδιαφερόμενων στο κομμάτι της ανάπτυξης του κώδικά του.
Το Snort με τον μηχανισμό των *Plug-ins* προσφέρει την δυνατότητα σε κάποιον που θέλει να χρησιμοποιήσει την ήδη έτοιμη υποδομή του προγράμματος, να προσθέσει κομμάτια κώδικα σε αυτό, τα οποία σαν ξεχωριστά υποπρογράμματα (modules), μπορούν να συνεργάζονται με το υπόλοιπο Snort, αυξάνοντας την λειτουργικότητά του, προσθέτοντας νέες δυνατότητες σε αυτό.

Στο επόμενο Κεφάλαιο παρουσιάζεται ένα τέτοιο *plug-in*, το οποίο ανήκει στην κατηγορία των *preprocessors* του Snort και το οποίο έχει υλοποιηθεί στα πλαίσια αυτής της πτυχιακής εργασίας.

Περισσότερες πληροφορίες για το Snort μπορεί κάποιος να βρει στο δικτυακό του τόπο http://www.snort.org, όπου επίσης διατίθενται τα αντίστοιχα εκτελέσιμα αρχεία του προγράμματος καθώς και ο πηγαίος κώδικάς του.