ΚΕΦΑΛΑΙΟ 6

Πρόγραμμα Ανίχνευσης του *Icmp Echo spoofing* (`spp_icmpspoof`)

Περιεχόμενα

ΜΕΡΟΣ Α: Γενική Περιγραφή του Προγράμματος

Το πρόγραμμα αυτό έχει γραφτεί σαν ένα plug-in στο Snort το οποίο παρουσιάστηκε στο Κεφάλαιο 5, με σκοπό να αυξήσει την λειτουργικότητά του, προσθέτοντας σε αυτό την δυνατότητα ανίχνευσης μιας ακόμα κατηγορίας ύποπτων πακέτων που μπορεί να χρησιμοποιηθούν για μία δικτυακή επίθεση. Το plug-in αυτό ανήκει στους preprocessors του Snort, δηλαδή στα κομμάτια του κώδικα που εκτελούνται μετά το *Decode Engine* και πριν το *Detection Engine*, είναι γραμμένο σε γλώσσα προγραμματισμού C όπως και ο υπόλοιπος κώδικας του Snort και έχει το όνομα ***spp_icmpspoof***.

Στόχος του συγκεκριμένου προγράμματος είναι να υλοποιήσει την *Τεχνική Ανίχνευσης του Icmp Echo spoofing* όπως αυτή περιγράφηκε λεπτομερώς στο Κεφάλαιο 4. Το πρόγραμμα αυτό εκτελείται σαν μία επέκταση του Snort, το οποίο μπορεί να είναι εγκατεστημένο στην συσκευή *Sensor* όπως αυτή αναφέρθηκε στο Κεφάλαιο 4. Για ευκολότερη αντίληψη της λειτουργίας του προγράμματος, επιβάλλεται η ανάγνωση και κατανόηση των διαδικασιών που περιγράφηκαν στο Κεφάλαιο 4, καθώς και της περιγραφής της δομής του Snort που έγινε στο Κεφάλαιο 5.

Ο `icmpspoof` preprocessor εκμεταλλεύεται την ικανότητα του Snort να αποκωδικοποιεί και να αναλύει τα πακέτα που ανήκουν στο traffic ενός δικτύου, και στην συνέχεια επεξεργάζεται τα πακέτα που τον ενδιαφέρουν, με σκοπό να ανιχνεύσει τα *Icmp Echo Request* και *Icmp Echo Reply* πακέτα που εισέρχονται και εξέρχονται από το δίκτυο που προστατεύει το Snort και τα οποία έχουν ψεύτικη διεύθυνση αποστολέα.
Το αποτέλεσμα του preprocessor είναι η επισήμανση του γεγονότος, με την καταγραφή σε ένα αρχείο, του πακέτου που οδήγησε στην ανίχνευσή του *spoofing*, ακολουθούμενο από ένα σενάριο που δημιουργείται δυναμικά και περιγράφει τον τρόπο με τον οποίο το *spoofing* υλοποιήθηκε, όσο αναφορά την θέση του κάθε συστήματος που πήρε μέρος στην διαδικασία σε σχέση με το προστατευόμενο δίκτυο.
Έτσι με την χρήση του συγκεκριμένου preprocessor είναι δυνατή η ανίχνευση των *spoofed Icmp Echo Request / Reply* πακέτων:

- ☞ Τα οποία στέλνονται σε κάποιο από τα προστατευόμενα δίκτυα
- ☞ Η ψεύτικη διεύθυνση αποστολέα που έχουν, ανήκει σε κάποιο από τα προστατευόμενα δίκτυα.

◆ Ενσωμάτωση του Προγράμματος στο Snort

Το συγκεκριμένο πρόγραμμα ανήκει στην κατηγορία των preprocessor plug-ins του Snort.
Το πρόγραμμα αυτό αποτελείται από δύο αρχεία. Το `spp_icmpspoof.c` το οποίο περιέχει τον πηγαίο κώδικα που υλοποιεί την λειτουργία του preprocessor και το `spp_icmpspoof.h` το οποίο περιέχει τους ορισμούς κάποιων συναρτήσεων του προγράμματος.

Για να μπορέσει ο συγκεκριμένος preprocessor να ενσωματωθεί στον υπόλοιπο κώδικα του Snort έγιναν οι εξής ενέργειες :

1. Προστέθηκε η δήλωση `#include spp_icmpspoof.h` στο αρχείο `plugbase.h` του Snort, στο σημείο που βρίσκονται και οι υπόλοιπες `#include` δηλώσεις που αφορούν τους preprocessors του Snort.

2. Στο αρχείο `plugbase.c` του Snort προστέθηκε μέσα στην συνάρτηση `InitPreprocessors()` η κλήση της συνάρτησης `SetupIcmpSpoof()` η οποία ανήκει στον `spp_icmpspoof` preprocessor. Με τον τρόπο αυτό θα γίνουν οι απαραίτητες αρχικοποιήσεις του preprocessor όταν ξεκινήσει η εκτέλεση του Snort.

3. Στο αρχείο `Makefile.am` του Snort και στο κομμάτι με τον τίτλο "`snort_SOURCES`", προστέθηκαν τα ονόματα των δύο αρχείων του preprocessor (`spp_icmpspoof.c` και `spp_icmpspoof.h`) έτσι ώστε να συμπεριληφθεί και ο κώδικάς τους κατά την μεταγλώττιση (compilation) του Snort.

4. Εκτελέστηκε η μεταγλώττιση (compilation) του Snort με την εντολή `automake`.

Μετά από αυτά τα βήματα ο νέος preprocessor αποτελεί μέρος του κώδικα του Snort.

◆ Χρήση – Παράμετροι Εκτέλεσης του Προγράμματος

Για την εκτέλεση του preprocessor μπορούν να οριστούν παράμετροι που θα καθορίσουν την λειτουργία του ανάλογα με τις ανάγκες του εκάστοτε χρήστη και του δικτύου που έχει σκοπό να προστατέψει.
Οι παράμετροι αυτοί ορίζονται ανεξάρτητα από τις παραμέτρους με τις οποίες θα εκτελεστεί το Snort και αναγράφονται στο configuration αρχείο του Snort (συνήθως το `snort.conf`) στο σημείο που θα δηλωθεί και το όνομα του preprocessor, ώστε να ενεργοποιηθεί και αυτός με την εκτέλεση του Snort. Η δήλωση του preprocessor και των παραμέτρων του έχει την μορφή :

```
preprocessor icmpspoof: <Protected Nets(s)> <Timeout Value> <Log Dir/File>
```

Ο συγκεκριμένος preprocessor, όπως φαίνεται και στην παραπάνω δήλωση, μπορεί να δεχτεί μέχρι και τρεις παραμέτρους, οι οποίες πρέπει να διαχωρίζονται με έναν κενό χαρακτήρα μεταξύ τους. Το λεκτικό `icmpspoof` είναι το όνομα (keyword) του preprocessor και πρέπει να ακολουθείται από τα ¨:¨ πριν από την δήλωση των παραμέτρων του. Στη συνέχεια περιγράφεται η χρήση και η λειτουργία των παραμέτρων αυτών:

1η Παράμετρος: **Προστατευόμενα Δίκτυα (*Protected Net(s)*).**

Με αυτή την παράμετρο ορίζονται τα δίκτυα τα οποία επιθυμεί ο χρήστης να ελέγχονται για την διέλευση τυχόν *spoofed Icmp Echo* πακέτων. Ο χρήστης μπορεί να ορίσει ένα ή περισσότερα δίκτυα με CIDR τρόπο γραφής. Σε περίπτωση που δηλωθούν περισσότερα του ενός δίκτυα, τότε αυτά πρέπει να διαχωρίζονται με ένα κόμμα (",") μεταξύ τους (χωρίς κενούς χαρακτήρες).

Επίσης σαν τιμή σε αυτήν την παράμετρο μπορεί να χρησιμοποιηθεί και η μεταβλητή `$HOME_NET` του Snort.
Η παράμετρος αυτή είναι υποχρεωτική.

2η Παράμετρος: **Τιμή Χρόνου Παραμονής των Πακέτων (*Timeout value*).**

Η παράμετρος αυτή ορίζει τον χρόνο σε δευτερόλεπτα που θεωρείται αρκετός να αναμένεται μία απάντηση (για παράδειγμα ένα *Icmp Echo Reply*) για ένα *Icmp Echo Request* πακέτο. Ο καθορισμός αυτής της παραμέτρου παίζει σημαντικό ρόλο στην απόδοση και την αξιοπιστία του preprocessor. Αν δοθεί μία μεγάλη τιμή σε αυτή, τότε θα αυξηθεί άσκοπα ο χρόνος εκτέλεσης του preprocessor όπως επίσης και το μέγεθος της μνήμης που χρησιμοποιεί. Αντίθετα αν δοθεί μία αρκετά μικρή τιμή σε αυτή, τότε είναι πιθανό να μειωθεί η αξιοπιστία του preprocessor όσο αναφορά την ανίχνευση των spoofed πακέτων. Ο ορισμός αυτής της παραμέτρου από τον χρήστη δεν είναι υποχρεωτικός και αν δεν δοθεί κάποια τιμή τότε θα χρησιμοποιηθεί η εξ ‘ορισμού τιμή των 3 δευτερολέπτων.

3η Παράμετρος: **Κατάλογος και Αρχείο Καταγραφής (*Log Dir /File*).**

Με την παράμετρο αυτή ορίζεται το αρχείο στο οποίο θα καταγράφονται τα γεγονότα που ανιχνεύτηκαν.
Ο κατάλογος και η πλήρης διαδρομή που οδηγεί σε αυτόν πρέπει ήδη να υπάρχουν, ενώ το αρχείο θα δημιουργηθεί κατά την εκτέλεση του preprocessor.
Αυτή η παράμετρος είναι επίσης προαιρετική και αν δεν δοθεί κάποια τιμή σε αυτή τότε τα γεγονότα θα καταγράφονται στο εξ’ ορισμού αρχείο `EchoSpoofs` το οποίο θα δημιουργηθεί κατά την εκτέλεση του preprocessor μέσα στον κατάλογο που χρησιμοποιεί το Snort (συνήθως στον `/var/log/snort/`) για να καταγράφει τα δικά του αποτελέσματα.

> **Σημείωση :**
>
> Αν δοθεί μόνο η 1η παράμετρος τότε οι άλλες δύο θα πάρουν τις εξ’ ορισμού τιμές τους.
> Αν δεν δοθεί η 2η παράμετρος και δηλωθεί η 3η τότε πρέπει να δοθεί στην 2η παράμετρο η τιμή 0 για να χρησιμοποιηθεί η εξ’ ορισμού τιμή της.

Για να γίνει καλύτερα κατανοητή η χρήση των παραμέτρων δίνονται μερικά παραδείγματα :

- `preprocessor icmpspoof: 192.100.100.0/24 4 /tmp/spoofs.log`

  Όλες οι παράμετροι είναι ορισμένες από τον χρήστη. Το προστατευόμενο δίκτυο είναι το 192.100.100.0 με μάσκα 24 (δηλαδή τα συστήματα με IP διεύθυνση από 192.100.100.0 έως 192.100.100.255). Για κάθε *Icmp Echo Request* πακέτο που εντοπίζεται, θα αναμένεται κάποια απάντηση σε αυτό για διάστημα τεσσάρων δευτερολέπτων, ενώ τα αποτελέσματα του preprocessor θα καταγράφονται στο αρχείο `/tmp/spoofs.log`.

- `preprocessor icmpspoof: 192.100.100.0/24`

  Η 2η και οι 3η παράμετρος θα πάρουν τις εξ’ ορισμού τιμές τους.

- `preprocessor icmpspoof: 192.100.100.0/24 4`

Η 3η παράμετρος θα πάρει την εξ' ορισμού τιμή της.

- `preprocessor icmpspoof: 192.100.100.0/24 0 /tmp/spoofs.log`

  Η 2η παράμετρος θα πάρει την εξ' ορισμού τιμή της.

- `preprocessor icmpspoof: 192.100.100.0/24,192.110.100.1/32 4 /tmp/spoofs.log`

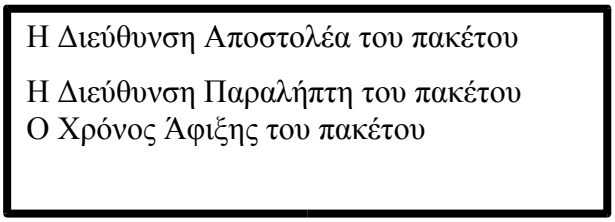
  Όλες οι παράμετροι είναι ορισμένες από τον χρήστη και έχουν δηλωθεί δύο δίκτυα τα οποία θα ελέγχονται για τυχόν διέλευση *spoofed Icmp Echo* πακέτων.

◆ Η Λειτουργία του Προγράμματος

Ο preprocessor εκτελείται κάθε φορά που το Snort επεξεργάζεται ένα πακέτο. Η λειτουργία του στηρίζεται στη αντιστοίχηση των εισερχόμενων και εξερχόμενων από το προστατευόμενο δίκτυο *Icmp Echo Request* πακέτων, με άλλα *Icmp* πακέτα που εντοπίζονται και υποτίθεται ότι σχετίζονται με κάποιο *Icmp Echo Request*. Για την επίτευξη του στόχου του ο preprocessor κρατάει ένα είδος ιστορικού των *Icmp Echo Request* πακέτων που εισέρχονται και εξέρχονται από τα δίκτυα που προστατεύει.

Η διαδικασία αυτή υλοποιείται με την χρήση δύο δυναμικών συνδεδεμένων λιστών, μία για τα εισερχόμενα και μία για τα εξερχόμενα από το προστατευόμενο δίκτυο *Icmp Echo Request* πακέτα. Οι λίστες αυτές ονομάζονται *Inbound* και *Outbound Lists* και σε αυτές δεν καταχωρούνται ολόκληρα τα *Icmp Echo Request* πακέτα, αλλά μόνο οι πληροφορίες που θεωρούνται απαραίτητες για την επεξεργασία τους.

Οι πληροφορίες που πρέπει να αποθηκευτούν για το κάθε *Icmp Echo Request* πακέτο είναι :



Σχήμα Α-1: Πληροφορία του Icmp Echo Request πακέτου προς αποθήκευση

Οι κύριες ενέργειες που εκτελεί το πρόγραμμα και αποτελούν την λειτουργία του είναι :

1. **Εισαγωγή** των *Echo Requests* στις *Inbound/Outbound Λίστες*
2. **Αναζήτηση** στις *Inbound/Outbound Λίστες* - **Ανίχνευση** του *spoofing*.
3. **Καθαρισμός** των *Inbound/Outbound Λιστών* από τα πακέτα που θεωρούνται παλιά.
4. **Επισήμανση** – **Καταγραφή** του *spoofing* (εάν αυτό υφίσταται) .

Η ενέργεια **1** εκτελείται όταν εντοπιστεί ένα *Icmp Echo Request* πακέτο, ενώ οι ενέργειες **2** και **4** εκτελούνται για τα *Icmp* πακέτα που εντοπίζονται και πρέπει να ελεγχθούν για το αν έχουν λόγο ύπαρξης, με την έννοια ότι έχει προηγηθεί κάποιο αποθηκευμένο *Icmp Echo Request* πακέτο που οδήγησε στην δημιουργία τους. Η ενέργεια **3** εκτελείται μετά από κάθε εκτέλεση της ενέργειας **1** ή της ενέργειας **2**.
Στη συνέχεια περιγράφονται αναλυτικά οι παραπάνω λειτουργίες:

1. Εισαγωγή Στις *Inbound/Outbound Λίστες*

Η εισαγωγή πραγματοποιείται όταν ένα *Icmp Echo Request* πακέτο γίνει αντιληπτό.
Ανάλογα με την προέλευση και τον προορισμό του πακέτου, αυτό θα καταχωρηθεί στην κατάλληλη λίστα, αφού πρώτα γίνουν οι απαραίτητοι έλεγχοι για το αν αυτό τηρεί τις παρακάτω προϋποθέσεις :

a) Το πακέτο που εντοπίζεται πρέπει να είναι ένα ολόκληρο πακέτο και όχι κάποιο fragment του.
Χάρη στις δυνατότητες του Snort τα πακέτα που δεν είναι ολόκληρα, αλλά έχουν κατακερματιστεί, επανασυγκροτούνται με την χρήση άλλου preprocessor (`spp_frag2`) ο οποίος θα δώσει σαν αποτέλεσμα το ολοκληρωμένο πακέτο. Παρόλα αυτά ο κάθε preprocessor θα πάρει και όλα τα fragments που έχουν καταφθάσει πριν την επανασυγκρότηση του πακέτου και αυτά πρέπει να αγνοηθούν, όπως και γίνεται.
Ο λόγος που ο `icmpspoof` preprocessor πρέπει να επεξεργάζεται μόνο ολόκληρα πακέτα, είναι γιατί αν το πακέτο έχει κατακερματιστεί, τότε το κάθε fragment του θα περιέχει μόνο ένα μέρος της πληροφορίας του πακέτου που απαιτείται για την λειτουργία του preprocessor, κάτι που θα οδηγούσε σε λανθασμένα συμπεράσματα.

b) Το πακέτο που εντοπίζεται δεν πρέπει να έχει διεύθυνση αποστολέα και παραλήπτη, είτε που και οι δύο δεν ανήκουν σε κάποιο(α) από τα προστατευόμενα δίκτυα, είτε που και οι δύο ανήκουν σε κάποιο(α) από τα προστατευόμενα δίκτυα.
Το 1^{ο} μέρος της προϋπόθεσης πρέπει να ισχύει έτσι ώστε ο preprocessor να μην επεξεργάζεται πακέτα που δεν ενδιαφέρουν τα δίκτυα που προστατεύει, ενώ το 2^{ο} μέρος της προϋπόθεσης πρέπει να ισχύει έτσι ώστε ο preprocessor να μην επεξεργάζεται τα πακέτα που ανήκουν στο εσωτερικό traffic ενός προστατευόμενου δικτύου ή πακέτα που ανταλλάζουν δύο προστατευόμενα δίκτυα μεταξύ τους. Αυτό πρέπει να ισχύει καθώς ο `icmpspoof` preprocessor έχει σαν στόχο να ανιχνεύσει μόνο τα spoofed *Icmp Echo* πακέτα που εισέρχονται ή εξέρχονται από τα προστατευόμενα δίκτυα και έχουν σχέση με κάποιο σύστημα που δεν ανήκει σε αυτά.

Για την εισαγωγή του πακέτου σε κάποια λίστα ισχύουν οι παρακάτω κανόνες :

- Η καταχώρησή του γίνεται πάντα στο τέλος της λίστας. Έτσι ο τελευταίος κόμβος της κάθε λίστας αφορά το τελευταίο *Icmp Echo Request* που καταγράφηκε και θα έχει τον μεγαλύτερο χρόνο από τους υπόλοιπους.

- Δεν επιτρέπονται διπλοεγγραφές στις λίστες . Αυτό έχει την έννοια ότι δεν επιτρέπεται η εισαγωγή ενός *Echo Request,* εάν στην ίδια λίστα υπάρχει κάποιο ήδη καταχωρημένο, με την ίδια διεύθυνση αποστολέα και την ίδια διεύθυνση παραλήπτη. Σε αυτήν την περίπτωση ο ήδη καταχωρημένος κόμβος θα μεταφερθεί στο τέλος της λίστας έχοντας τον νέο χρόνο που εντοπίστηκε το πακέτο.

2. Ανίχνευση του *Spoofing*

Η ανίχνευση πραγματοποιείται με την προσπάθεια αντιστοίχησης των καταχωρημένων *Icmp Echo Request* πακέτων με άλλα *Icmp* πακέτα που γίνονται αντιληπτά και μπορεί να έχουν κάποια σχέση με αυτά.
Τα *Icmp* πακέτα που ελέγχονται και ο έλεγχος που γίνεται για το καθένα είναι :

- ***Τα Icmp Echo Reply***
  για τα οποία ελέγχεται αν υπάρχουν καταχωρημένα τα αντίστοιχα *Echo Request* και

  - ***Τα Icmp Destination Unreachable (codes 0/1 και 13)***
    που περιέχουν ένα

    - ***Icmp Echo Request***
      το οποίο ελέγχεται για το αν είναι καταχωρημένο σε κάποια λίστα

    - ***Icmp Echo Reply***
      για το οποίο ελέγχεται αν υπάρχει καταχωρημένο το αντίστοιχο *Echo Request*

Αν δεν πετύχει η αντιστοίχηση τότε η εμφάνιση αυτού του πακέτου θεωρείται ύποπτη και θα επισημανθεί ώστε να καταγραφεί το γεγονός του *spoofing*.
Αν πετύχει η αντιστοίχηση τότε το πακέτο θεωρείται νόμιμο και δεν υφίσταται περίπτωση *spoofing*.

Πριν ελεγχθεί κάποιο από τα παραπάνω πακέτα θα πρέπει να τηρεί τις ίδιες προϋποθέσεις με αυτές που ισχύουν για την εισαγωγή των *Icmp Echo Request* πακέτων, που αναφέρθηκαν παραπάνω.

Για τα *Icmp Destination Unreachable* πακέτα η προϋπόθεση **b** που αφορά τις IP διευθύνσεις, ισχύει για το *Icmp* πακέτο που περιέχουν στα data τους, ενώ για το ίδιο το *Unreachable* ισχύει μόνο το ένα μέρος της προϋπόθεσης αυτό που καμία από τις διευθύνσεις δεν ανήκουν σε κάποιο (α) από τα προστατευόμενα δίκτυα.
Το άλλο μέρος της προϋπόθεσης **b** (αυτό που και οι δύο διευθύνσεις ανήκουν σε κάποιο(α) από τα προστατευόμενα δίκτυο(α)) δεν ισχύει για τα συγκεκριμένα πακέτα, γιατί έτσι θα αγνοούνταν τα πακέτα που στέλνονται από κάποιο router ή firewall που ανήκει σε κάποιο από τα προστατευόμενα δίκτυα, προς κάποιο προστατευόμενο δίκτυο.

Επίσης ένα *Icmp Destination Unreachable (codes 0/1)* που περιέχει ένα *Icmp Echo Reply* πακέτο θα προκαλέσει την καταγραφή ενός γεγονότος *spoofing*, έστω και αν έχει βρεθεί καταχωρημένο σε κάποια λίστα το αντίστοιχο *Icmp Echo Request* που προκάλεσε την δημιουργία του *Echo Reply* που περιέχει το *Unreachable*.
Αυτό συμβαίνει στις περιπτώσεις που αναφέρθηκαν στο ΜΕΡΟΣ Β του Κεφαλαίου 4 όταν ισχύει ο *Γενικός Κανόνας*, κατά τον οποίο δεν είναι δυνατόν κάποιος host να στέλνει ένα *Reply* σε κάποιον που υποτίθεται ότι του έστειλε ένα *Request* και αυτός (ή το δίκτυό του) να μην υπάρχει.

3. Καθαρισμός Των *Inbound/Outbound Λιστών*

Ο καθαρισμός των λιστών από πακέτα που θεωρούνται παλιά έχει σαν κριτήριο τον χρόνο που έχει κάθε καταχωρημένο πακέτο. Πραγματοποιείται ώστε να απομακρυνθούν πακέτα που έχουν ελεγχθεί ή για τα οποία δεν έχει εμφανιστεί κάποια απάντηση που να είναι μέσα στα χρονικά όρια που έχουν καθοριστεί με την παράμετρο *Timeout* που εκτελείται ο preprocessor.

Τα καταχωρημένα πακέτα ελέγχονται για τον χρόνο τους πριν από κάθε νέα εισαγωγή και πριν από κάθε αναζήτηση για ανίχνευση και επισημαίνονται αυτά που πρέπει να απομακρυνθούν αν έχουν χρόνο μικρότερο από την διαφορά του χρόνου του πακέτου που μόλις εντοπίστηκε πλην την *Timeout* παράμετρο.
Ο καθαρισμός της λίστας θα πραγματοποιηθεί μετά το πέρας της εισαγωγής ή της ανίχνευσης.

Με τον καθαρισμό επιτυγχάνεται η βελτιστοποίηση της ταχύτητας του προγράμματος και η οικονομία του ποσού της μνήμης που χρησιμοποιεί, καθώς στις λίστες παραμένουν τα πακέτα που πραγματικά έχουν κάποιο ενδιαφέρον για να ελεγχθούν, με την έννοια ότι είναι πιθανό να καταφθάσει κάποιο πακέτο που να έχει σχέση με αυτά.
Επίσης με τον καθαρισμό αποφεύγονται τυχόν λανθασμένα συμπεράσματα που μπορεί να εξαχθούν από συμπτωματικές συσχετίσεις που μπορεί να προκύψουν μεταξύ νέων πακέτων που εντοπίζονται με *Icmp Echo Request* πακέτα που έχουν καταχωρηθεί σε αρκετά προηγούμενο στάδιο και στην ουσία δεν έχουν κάποια σχέση μεταξύ τους.

Κατά την διαγραφή των κόμβων από τις λίστες αυτοί μεταφέρονται σε μία άλλη λίστα η οποία ονομάζεται *Λίστα Ανακύκλωσης*. Αυτό συμβαίνει έτσι ώστε αυτοί οι κόμβοι να μπορούν να χρησιμοποιηθούν στο μέλλον για μία νέα εισαγωγή και να μην χρειάζεται να δεσμεύονται και να αποδεσμεύονται συνεχώς μικρά κομμάτια μνήμης με τους κινδύνους που αυτό εγκυμονεί.
Η *Λίστα Ανακύκλωσης* είναι επίσης μία δυναμική συνδεδεμένη λίστα με κάποιο όμως προκαθορισμένο ανώτατο όριο του αριθμού των κόμβων που μπορούν να καταχωρηθούν σε αυτήν.

4. **Καταγραφή *Spoofing***

Η διαδικασία αυτή είναι που καταγράφει το γεγονός του *spoofing* όταν αυτό συμβαίνει.

Το γεγονός έχει επισημανθεί από την διαδικασία της *Ανίχνευσης του spoofing* (Λειτουργία 2) και οι πληροφορίες που το αφορούν θα καταγραφούν στο αρχείο που έχει δοθεί σαν παράμετρος κατά την εκτέλεση του preprocessor.

Σε αυτό το αρχείο θα καταγραφεί το *Icmp* πακέτο που οδήγησε στην *Ανίχνευση* του *spoofing* καθώς και ένα σενάριο που περιγράφει τον τρόπο με τον οποίο το *spoofing* πραγματοποιήθηκε .

Το σενάριο αυτό δημιουργείται δυναμικά και λαμβάνει υπόψη του την λίστα που ελέγχθηκε και στην οποία δεν βρέθηκε το ζητούμενο *Icmp Echo Request* πακέτο, καθώς επίσης και το είδος του *Icmp* πακέτου που χρησιμοποιήθηκε για να γίνει η ανίχνευση. Το αποτέλεσμα είναι η επισήμανση και η καταγραφή του *spoofing,* αναφέροντας την IP διεύθυνση του παραλήπτη του *spoofed* πακέτου καθώς και την ψεύτικη IP διεύθυνση που χρησιμοποιήθηκε για την αποστολή του πακέτου. Επίσης αναφέρει αν το *spoofed* πακέτο προήλθε από κάποιο σύστημα που ανήκει στα προστατευόμενα δίκτυα ή αν προήλθε από κάποιο σύστημα που δεν ανήκει σε αυτά, δηλώνοντας την θέση του κάθε συστήματος που πήρε μέρος στο *spoofing* σε σχέση με το προστατευόμενο δίκτυο.

ΜΕΡΟΣ Β: Περιγραφή των Συνδεδεμένων Λιστών του Προγράμματος

Στις επόμενες σελίδες περιγράφονται οι δυναμικές λίστες που χρησιμοποιούνται στο πρόγραμμα και οι δομές δεδομένων οι οποίες υλοποιούν τους κόμβους τους. Οι λίστες αυτές είναι οι *Inbound / Outbound Λίστες* στις οποίες αποθηκεύονται τα εισερχόμενα και εξερχόμενα *Icmp Echo Request* πακέτα αντίστοιχα, καθώς και η *Λίστα Ανακύκλωσης* στην οποία μεταφέρονται οι κόμβοι των *Inbound / Outbound Λιστών* που χρειάστηκε να διαγραφούν.

Κατά την άφιξη των πακέτων στον preprocessor γίνεται ένας αρχικός έλεγχος για το αν τα *Icmp* πακέτα τηρούν τις προϋποθέσεις **a** και **b** που αναφέρθηκαν στο ΜΕΡΟΣ Α του Κεφαλαίου. Τα πακέτα που δεν ανήκουν στα *Icmp* πακέτα που ενδιαφέρουν τον preprocessor και/ή δεν τηρούν τις προϋποθέσεις **a** και **b,** αγνοούνται από τον preprocessor και δεν υποβάλλονται σε κάποια επεξεργασία.

Στη συνέχεια για κάθε ένα από τα πακέτα που απαιτείται να επεξεργαστούν από τον preprocessor, αποθηκεύεται η απαραίτητη μόνο πληροφορία που τα αφορά και όχι ολόκληρο το πακέτο, σε μία δομή την **`DtgInfo`.** Η δομή αυτή δεν υλοποιεί κόμβο κάποιας λίστας που χειρίζεται ο preprocessor αλλά παρόλα αυτά κατέχει σημαντικό ρόλο για την λειτουργία του προγράμματος, καθώς αυτή χρησιμοποιείται για την αναπαράσταση του πακέτου, κατέχοντας την ελάχιστη πληροφορία που απαιτείται για αυτό.

Η Δομή `DtgInfo`

	`Type`	**`DtgSrc`**	**`DtgDst`**	**`DtgT`**
`DtgInfo`	**Τα πεδία Code και Type του Icmp πακέτου**	**Διεύθυνση αποστολέα του πακέτου**	**Διεύθυνση παραλήπτη του πακέτου**	**Χρόνος άφιξης του πακέτου.**

Σχήμα Β-1: Η δομή `DtgInfo`.

Η δομή **`DtgInfo`** αποτελείται από τέσσερα πεδία όπως φαίνεται και στο παραπάνω σχήμα, στα οποία θα καταχωρηθεί η ελάχιστη πληροφορία που απαιτείται για το κάθε πακέτο που πρέπει να επεξεργαστεί ο preprocessor.

Ο παρακάτω πίνακας εξηγεί τα πεδία της δομής **`DtgInfo`** και την πληροφορία που αυτά περιέχουν.

Στην 1η στήλη του πίνακα αναφέρεται το όνομα του πεδίου της δομής, ενώ στην 2η στήλη αναφέρεται η πληροφορία που αποθηκεύεται σε αυτό το πεδίο.

<table>
<tr><th>Πεδίο</th><th>Πληροφορία</th></tr>
<tr><td>Type</td><td>Σε αυτό το πεδίο αποθηκεύεται ένας ακέραιος αριθμός που αντιπροσωπεύει το είδος του Icmp πακέτου (Type και Code).
Το πεδίο αυτό μπορεί να έχει τις εξής τιμές :

<table>
<tr><th>Τιμή</th><th>Είδος πακέτου</th></tr>
<tr><td>8</td><td>Icmp Echo Request πακέτο</td></tr>
<tr><td>0</td><td>Icmp Echo Reply πακέτο</td></tr>
<tr><td>3018</td><td>Icmp Destination Unreachable-Host/Net Unreachable που περιέχει ένα Icmp Echo Request πακέτο</td></tr>
<tr><td>3010</td><td>Icmp Destination Unreachable-Host/Net Unreachable που περιέχει ένα Icmp Echo Reply πακέτο</td></tr>
<tr><td>3138</td><td>Icmp Destination Unreachable-Administrative Prohibited που περιέχει ένα Icmp Echo Request πακέτο.</td></tr>
<tr><td>3130</td><td>Icmp Destination Unreachable-Administrative Prohibited που περιέχει ένα Icmp Echo Reply πακέτο</td></tr>
</table></td></tr>
<tr><td>DtgSrc</td><td>Σε αυτό το πεδίο καταχωρείται η IP διεύθυνση του αποστολέα του πακέτου και έχει τον τύπο <code>struct in_addr</code> ο οποίος είναι κατάλληλος ώστε να αποθηκεύει IP διευθύνσεις.</td></tr>
<tr><td>DtgDst</td><td>Σε αυτό το πεδίο καταχωρείται η IP διεύθυνση του παραλήπτη του πακέτου και έχει τον τύπο <code>struct in_addr</code> ο οποίος είναι κατάλληλος για να αποθηκεύει IP διευθύνσεις.</td></tr>
<tr><td>DtgT</td><td>Σε αυτό το πεδίο καταχωρείται ο χρόνος που κατέφθασε το πακέτο. Ο χρόνος εκφράζεται σε δευτερόλεπτα, και είναι το πλήθος των δευτερολέπτων που έχουν περάσει από την 1/1/1970 μέχρι σήμερα (τοπική ώρα του συστήματος).
Το πεδίο αυτό έχει τύπο <code>TIME_T</code> που είναι κατάλληλος για να αποθηκεύει τέτοιες τιμές.</td></tr>
</table>

Πίνακας Β-1: Τα πεδία της δομής `DtgInfo`

Σημείωση

Για τα Unreachable πακέτα οι IP διευθύνσεις που αποθηκεύονται αφορούν τα Icmp πακέτα που περιέχουν στα data τους και όχι τα ίδια τα Unreachable πακέτα.

Οι *Inbound/Outbound Λίστες*

Τα *Icmp Echo Request* πακέτα καταχωρούνται σε δύο όμοιες, δύο διαστάσεων, διπλής κατεύθυνσης, κυκλικές, δυναμικές συνδεδεμένες λίστες.
Στην μία από αυτές καταχωρούνται τα εισερχόμενα *Echo Requests*, δηλαδή αυτά που προέρχονται από ένα μη προστατευόμενο δίκτυο και προορίζονται προς ένα προστατευόμενο δίκτυο (Inbound traffic), ενώ στην άλλη καταχωρούνται τα εξερχόμενα *Echo Requests*, δηλαδή αυτά που προέρχονται από κάποιο προστατευόμενο δίκτυο και προορίζονται προς ένα μη προστατευόμενο δίκτυο (Outbound traffic).

Χαρακτηριστικά των *Inbound/Outbound Λιστών*

Η **1η** διάσταση της κάθε λίστας αποτελείται από κόμβους που αποθηκεύουν την απαιτούμενη πληροφορία για την πηγή του πακέτου, τον αποστολέα .

Η **2η** διάσταση της κάθε λίστας αποτελείται από κόμβους που κρατάνε την απαιτούμενη πληροφορία για τον προορισμό του πακέτου, τον παραλήπτη, καθώς και τον χρόνο τον οποίο κατέφθασε το πακέτο.

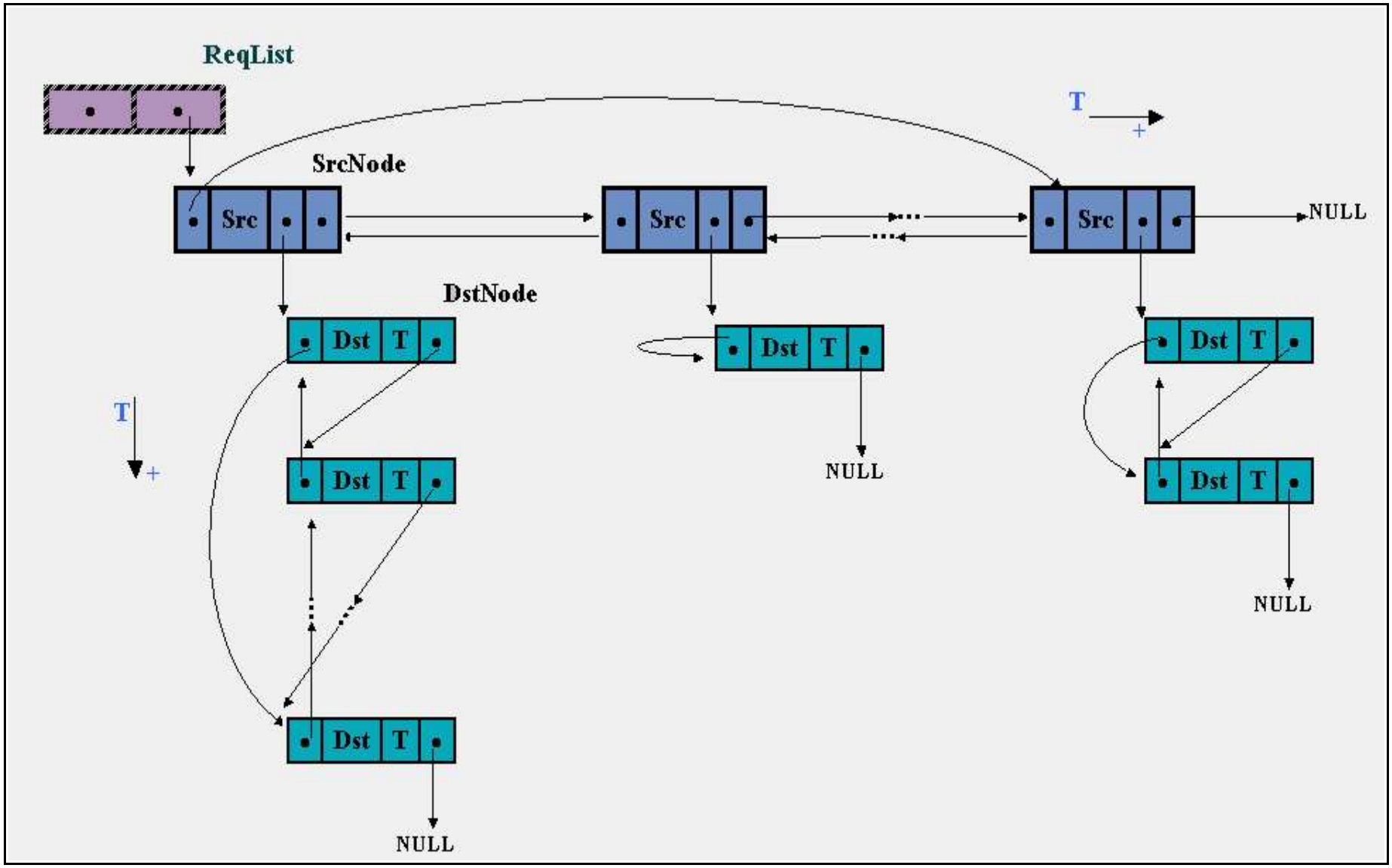
Ο κάθε κόμβος της κάθε διάστασης της λίστας δείχνει στον επόμενο και τον προηγούμενο κόμβο της ίδιας διάστασης.
Ο δείκτης προς τον επόμενο κόμβο, του τελευταίου κόμβου της κάθε διάστασης δείχνει στο NULL.

Ο δείκτης προς τον προηγούμενο κόμβο, του πρώτου κόμβου της κάθε διάστασης, δείχνει στον τελευταίο κόμβο της διάστασης αυτής, εκτός εάν είναι ο ίδιος τελευταίος (δηλαδή είναι ο μοναδικός κόμβος που υπάρχει) οπότε και δείχνει στον εαυτό του.

Ο κάθε κόμβος που αποθηκεύει την πληροφορία για τον αποστολέα του πακέτου έχει και έναν pointer που δείχνει στην αρχή της $2^{ης}$ διάστασης της λίστας, που βρίσκονται οι κόμβοι με όλους τους παραλήπτες αυτού του αποστολέα.

Κάθε αποστολέας ενός *Icmp Echo Request* πακέτου μπορεί να υπάρχει στην $1^{η}$ διάσταση της λίστας μόνο μία φορά, ενώ κάθε παραλήπτης ενός *Icmp Echo Request*, μπορεί να υπάρχει μόνο μία φορά στην $2^{η}$ διάσταση της λίστας που αφορά τον ίδιο αποστολέα.

Στο παρακάτω σχήμα είναι δυνατόν να εντοπιστούν τα χαρακτηριστικά που περιγράφηκαν παραπάνω, καθώς φαίνεται η γενική μορφή που μπορεί να έχει κάποια από τις *Inbound / Outbound Λίστες* μία δεδομένη χρονική στιγμή.



Σχήμα Β-2: Η μορφή των Inbound/Outbound Λιστών

Οι Δομές που Υλοποιούν τους Κόμβους των *Inbound/Outbound Λιστών*

Στη συνέχεια περιγράφονται οι δομές που υλοποιούν τους κόμβους των *Inbound/Outbound Λιστών*:

Η Δομή `RequestList`

	`InReqs`	`OutReqs`
`RequestList`	**Pointer που δείχνει στον 1ο SrcNode της Inbound Λίστας.**	**Pointer που δείχνει στον 1ο SrcNode της Outbound Λίστας**

Σχήμα Β-3: Η δομή `RequestList`

Η δομή αυτή στην ουσία αποτελεί το σημείο εκκίνησης των δύο λιστών. Αποτελείται από 2 πεδία τα οποία είναι pointers που ο καθένας δείχνει στην αρχή της κάθε λίστας.

Ο παρακάτω πίνακας εξηγεί τα πεδία της δομής **`RequestList`** και την πληροφορία που περιέχουν.

Πεδίο	Πληροφορία
`InReqs`	Ένας pointer ο οποίος δείχνει στον 1ο κόμβο της λίστας στην οποία καταχωρούνται τα εισερχόμενα *Icmp Echo Request* πακέτα – την *Inbound Λίστα*
`OutReqs`	Ένας pointer ο οποίος δείχνει στον 1ο κόμβο της λίστας στην οποία καταχωρούνται τα εξερχόμενα *Icmp Echo Request* πακέτα – την *Outbound Λίστα*

Πίνακας Β-2: Τα πεδία της δομής `RequestList`

Οι κόμβοι στους οποίους δείχνουν οι δύο αυτοί pointers είναι κόμβοι-αποστολέων (**`SrcNode`**).

Η Δομή `SrcNode` - Κόμβος Αποστολέα

	`PrevSN`	`Src`	`FirstDN`	`NextSN`
`SrcNode`	**Pointer που δείχνει στον Προηγούμενο `SrcNode`**	**Διεύθυνση αποστολέα του Icmp Echo Request**	**Pointer που δείχνει στον πρώτο `DstNode`**	**Pointer που δείχνει στον Επόμενο `SrcNode`**

Σχήμα Β-4: Η δομή `SrcNode`

Η δομή αυτή είναι η δομή που κρατάει την απαιτούμενη πληροφορία για τον αποστολέα του *Icmp Echo Request* πακέτου. Αποτελεί τον κόμβο-αποστολέα.

Ο παρακάτω πίνακας εξηγεί τα πεδία της δομής **`SrcNode`** και την πληροφορία που περιέχουν.

Πεδίο	Πληροφορία

PrevSN	Pointer που δείχνει προς τον προηγούμενο `SrcNode` κόμβο της λίστας.
Src	Είναι η IP διεύθυνση του αποστολέα του πακέτου
FirstDN	Ένας pointer που δείχνει στον **1ο** κόμβο της **2η** διάστασης της λίστας, που κρατάει την απαραίτητη πληροφορία για τον παραλήπτη του *Icmp Echo Request* πακέτου.
NextSN	Pointer που δείχνει προς τον επόμενο `SrcNode` κόμβο της λίστας

Σχήμα Β-3: Τα πεδία της δομής `SrcNode`

Σημείωση

Το πεδίο `PrevSn` του 1ου κόμβου της λίστας δείχνει στον τελευταίο κόμβο της λίστας, εκτός αν είναι ο ίδιος τελευταίος (και μοναδικός) οπότε και δείχνει στον εαυτό του.

Δομή `DstNode` - Κόμβος Παραλήπτη

	PrevDN	**Dst**	**T**	**NextDN**
DstNode	**Pointer που δείχνει στον Προηγούμενο DstNode**	**Διεύθυνση παραλήπτη του Icmp Echo Request**	**Χρόνος άφιξης του πακέτου.**	**Pointer που δείχνει στον Επόμενο DstNode**

Σχήμα Β-5: Η δομή `DstNode`

Η δομή αυτή είναι η δομή που κρατάει την απαιτούμενη πληροφορία για τον προορισμό του *Icmp Echo Request* πακέτου. Αποτελεί τον κόμβο-παραλήπτη.

Ο παρακάτω πίνακας εξηγεί τα πεδία της δομής **`DstNode`** και την πληροφορία που περιέχουν.

Πεδίο	Πληροφορία
PrevDN	Pointer που δείχνει προς τον προηγούμενο **`DstNode`** κόμβο της λίστας.
Dst	Είναι η IP διεύθυνση του παραλήπτη του *Icmp Echo Request* πακέτου
T	Είναι ο χρόνος στον οποίο ανιχνεύτηκε το πακέτο
NextDN	Pointer που δείχνει προς τον επόμενο **`DstNode`** κόμβο της λίστας

Πίνακας Β-4: Τα πεδία της δομής `DstNode`

Σημείωση

Το πεδίο `PrevDn` του 1ου κόμβου της 2ης διάστασης της λίστας , δείχνει στον τελευταίο κόμβο της ίδιας διάστασης, εκτός αν είναι ο ίδιος τελευταίος (και μοναδικός) οπότε και δείχνει στον εαυτό του.

Οι *Λίστες Ανακύκλωσης*

Οι κόμβοι που έπρεπε να διαγραφούν από τις *Inbound/Outbound Λίστες* μεταφέρονται στις *Λίστες Ανακύκλωσης*. Η κάθε μία από αυτές τις συνδεδεμένες λίστες είναι μονής διάστασης, διπλής κατεύθυνσης, δυναμική λίστα.
Σε κάθε μία από αυτές τις λίστες αποθηκεύεται ένας αριθμός κόμβων-αποστολέων και κόμβων-παραληπτών αντίστοιχα, που χρειάστηκε να διαγραφούν από τις *Inbound/Outbound Λίστες* που καταχωρούν τα εισερχόμενα και εξερχόμενα από το προστατευόμενο δίκτυο *Icmp Echo Request* πακέτα.

Η χρησιμότητα των *Λιστών Ανακύκλωσης*, είναι ότι με την μεταφορά των κόμβων σε αυτές κατά την διαγραφή τους, αυτοί είναι διαθέσιμοι για μελλοντική χρήση για νέα εισαγωγή τους σε μία από τις *Inbound/Outbound Λίστες*. Με αυτόν τον τρόπο αποφεύγεται η συνεχής δέσμευση και αποδέσμευση μικρών τμημάτων μνήμης που έχει τον κίνδυνο να κατακερματιστεί (fragmented) η μνήμη και να μην είναι δυνατή η περαιτέρω χρήση της.
Το μέγιστο όριο του πλήθους των κόμβων, που μπορεί κάθε μία από τις *Λίστες Ανακύκλωσης* να αποθηκεύει ορίζεται μέσα στο πρόγραμμα.

Η δομή που χρησιμοποιείται για την υλοποίηση των *Λιστών Ανακύκλωσης* παρουσιάζεται παρακάτω:

	SumSN	**TrSN**	**SumDn**	**TrDN**
TrashBin	**Το πλήθος των κόμβων SrcNode**	**Pointer που δείχνει στον πρώτο κόμβο SrcNode της λίστας**	**Το πλήθος των κόμβων DstNode**	**Pointer που δείχνει στον πρώτο κόμβο DstNode της λίστας**

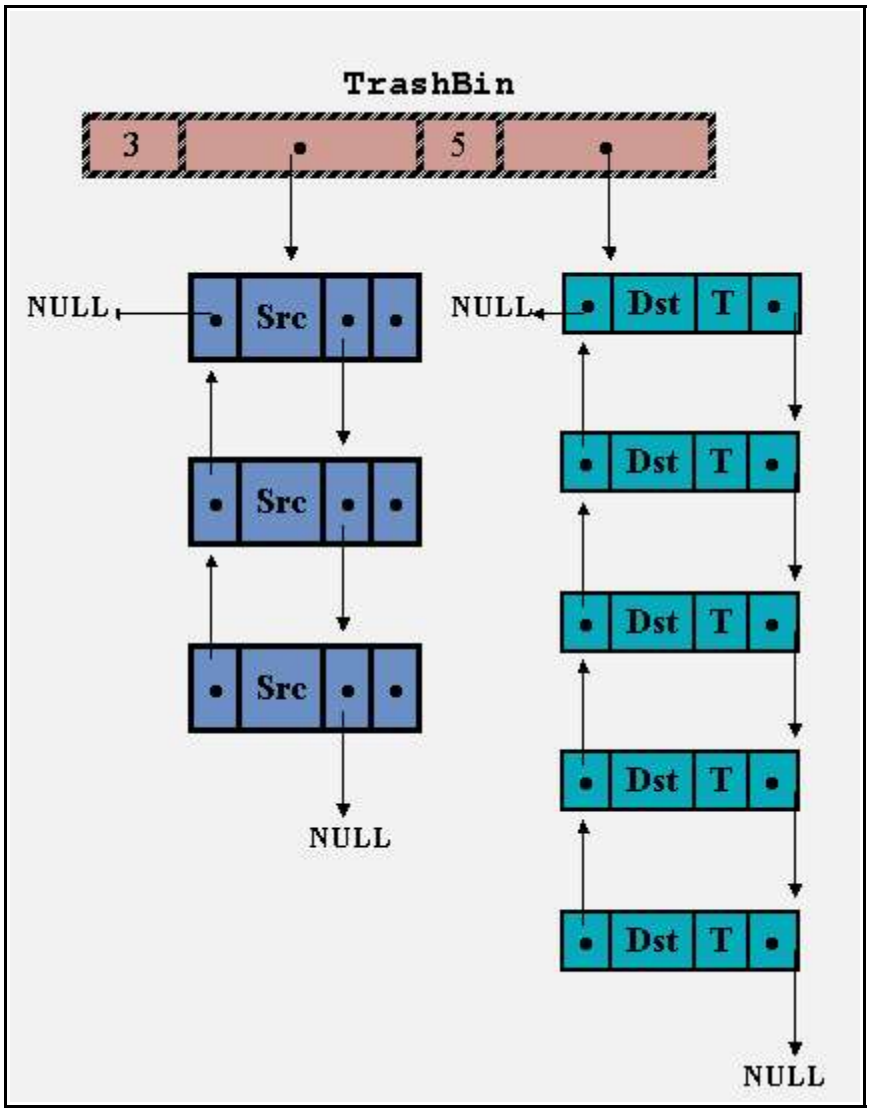
Σχήμα Β-6: Η δομή `TrashBin`

Ο παρακάτω πίνακας εξηγεί τα πεδία αυτής της δομής και της πληροφορίας που αποθηκεύεται σε αυτά.

Πεδίο	Πληροφορία
`SumSN`	Είναι το πλήθος των **`SrcNode`** κόμβων που βρίσκονται στο **`TrashBin`**. Δηλαδή των διαγραμμένων κόμβων-αποστολέα
`TrSN`	Ένας pointer που δείχνει στον 1ο **`SrcNode`** που βρίσκεται στο **`TrashBin`**.
`SumDn`	Είναι το πλήθος των **`DstNode`** κόμβων που βρίσκονται στο **`TrashBin`**. Δηλαδή των διαγραμμένων κόμβων-παραλήπτη.
`TrDN`	Ένας pointer που δείχνει στον 1ο **`DstNode`** που βρίσκεται στο **`TrashBin`**

Πίνακας Β-5: Τα πεδία της δομής `TrashBin`

Η *Λίστα Ανακύκλωσης* μετά την διαγραφή τριών κόμβων-αποστολέων (**`SrcNode`**) και πέντε κόμβων-παραληπτών (**`DstNode`**) από τις *Inbound/Outbound Λίστες* έχει την παρακάτω μορφή:



Σχήμα Β-7: Η μορφή των Λιστών Ανακύκλωσης

ΜΕΡΟΣ C: Περιγραφή των Συναρτήσεων του Προγράμματος

Στις επόμενες σελίδες περιγράφονται οι συναρτήσεις από τις οποίες αποτελείται το πρόγραμμα.

Οι συναρτήσεις `SetupIcmpSpoof()`, `IcmpSpoofInit()`, `ParseIcmpSpoofArgs()` και `IcmpSpoofFunction()`, είναι υποχρεωτικές συναρτήσεις που πρέπει να υλοποιεί ο κάθε preprocessor (με διαφορετικά ονόματα για τον καθένα), καθώς αυτές είναι που καλούνται από το Snort, για να μπορεί ο preprocessor να συνεργαστεί μαζί του. Η υλοποίηση όμως αυτών των συναρτήσεων μπορεί να περιλαμβάνει και κώδικα που εξυπηρετεί τις λειτουργίες του εκάστοτε preprocessor.
Οι συναρτήσεις `InsertRequest()`, `CheckResponse()`, `InsertNode_SN()`, `InsertNode_DN()`, `EjectCurNode_SN()`, `EjectCurNode_DN()`, `SearchNode_SN()`, `SearchNode_DN()`, `Prepare_To_Clean_SN()`, `Prepare_To_Clean_DN()`, `Trash_SN()`, `Trash_DN()` χρησιμοποιούνται για τον χειρισμό και τις λειτουργίες των λιστών, δηλαδή για την εισαγωγή , αναζήτηση και διαγραφή σε αυτές.
Παρακάτω γίνεται αναλυτική περιγραφή όλων αυτών των συναρτήσεων, καθώς και μερικών ακόμα γενικού σκοπού συναρτήσεων που υλοποιούνται στο πρόγραμμα.

```
void SetupIcmpSpoof()
```

Σκοπός /Περιγραφή
Η συνάρτηση αυτή εκτελείται κατά την εκκίνηση του Snort και καλείται στο αρχείο `plugbase.c` του Snort μέσα από την συνάρτηση `InitPreprocessors()`.
Στην ουσία με αυτόν τον τρόπο γίνεται δυνατό να μπορέσει να ενσωματωθεί ο κώδικας του preprocessor με τον υπόλοιπο κώδικα του Snort κατά την εκτέλεση.
Επίσης αυτή η συνάρτηση δηλώνει το όνομα (keyword) του preprocessor στο Snort και καλεί την `IcmpSpoofInit()`.

Καλείται
Από την συνάρτηση του Snort `InitPreprocessors()` (source αρχείο `plugbase.c`).

Καλεί
Την συνάρτηση `IcmpSpoofInit()`.

Παράμετροι
Δεν δέχεται παραμέτρους.

Μεταβλητές
Δεν έχει κάποιες τοπικές μεταβλητές.

Επιστρέφει
Δεν επιστρέφει κάποια τιμή.

```
void IcmpSpoofInit (u_char *args)
```

Σκοπός / Περιγραφή
Η συνάρτηση αυτή θα εκτελεστεί κατά την εκκίνηση του Snort και καλείται από την `SetupIcmpSpoof()`. Αρχικοποιεί τις μεταβλητές , τους pointers και τις διάφορες δομές του preprocessor .
Επίσης καλεί την `ParseIcmpSpoofArgs()` και με την κλήση της συνάρτησης `AddFuncToPreprocList()` του Snort, η οποία ορίζεται στο αρχείο `rules.c`, προσθέτει την συνάρτηση `IcmpSpoofFunction()` σε μία δυναμική λίστα (η οποία ορίζεται στο αρχείο `rules.c` του Snort) από την οποία θα καλείται κάθε φορά που θα επεξεργάζεται κάποιο πακέτο το Snort. Σε αυτή την λίστα είναι καταχωρημένες και οι αντίστοιχες συναρτήσεις των υπόλοιπων preprocessors.

Καλείται
Από την συνάρτηση `SetupIcmpSpoof()`.

Καλεί
Την συνάρτηση `ParseIcmpSpoofArgs()`.

Παράμετροι

Δέχεται μία παράμετρο :

Παράμετρος	Λειτουργία
`char *args`	Ένας pointer σε `char`. Η χρήση αυτής της παραμέτρου καθιστά δυνατό μέσα από αυτή την συνάρτηση να κληθεί η `ParseIcmpSpoofArgs()` η οποία κάνει χρήση αυτής της παραμέτρου.

Μεταβλητές
Δεν έχει κάποιες τοπικές μεταβλητές.

Επιστρέφει
Δεν επιστρέφει κάποια τιμή.

```
void ParseIcmpSpoofArgs (char *args)
```

Σκοπός / Περιγραφή
Η συνάρτηση αυτή θα εκτελεστεί μία φορά κατά την εκκίνηση του Snort και όταν θα κληθεί από την `IcmpSpoofInit()` .
Μέσω αυτής της συνάρτησης θα γίνει η επεξεργασία των διαφόρων παραμέτρων που δίνονται από τον χρήστη για την εκτέλεση του preprocessor (συνήθως στο αρχείο `snort.conf`).
Η συνάρτηση κάνει έλεγχο για την ορθότητα των παραμέτρων και στην ουσία αυτή είναι που θα περάσει τις ρυθμίσεις του χρήστη μέσα στο πρόγραμμα.
Ο χρήστης μπορεί να δώσει μέχρι τρεις παραμέτρους με τον κενό χαρακτήρα ως διαχωριστικό μεταξύ τους.
Οι παράμετροι αυτοί παρουσιάζονται στην συνέχεια:

1η . Protected Net(s). (Υποχρεωτική)

Η παράμετρος αυτή είναι και η μόνη υποχρεωτική παράμετρος που πρέπει να δοθεί για να λειτουργήσει σωστά ο preprocessor. Μέσω αυτής δίνεται η δυνατότητα να οριστεί το προστατευόμενο δίκτυο, δηλαδή το δίκτυο του οποίου επιθυμείται να ελέγχεται, το εισερχόμενο και εξερχόμενο traffic για την ανίχνευση spoofed *Icmp Echo* πακέτων. Αυτή η παράμετρος μπορεί να δοθεί με τέτοιο τρόπο ώστε να ορίζονται περισσότερα του ενός προστατευόμενα δίκτυα, είτε διαχωρίζοντάς τα με ένα κόμμα (¨,¨) χωρίς κενά μεταξύ τους , είτε διαχωρίζοντάς τα με κόμμα χωρίς κενά μεταξύ τους και τοποθετώντας τα όλα μέσα σε αγκύλες ([ ]).
Ο τελευταίος τρόπος παρέχει την δυνατότητα να υπάρχει συμβατότητα με την μεταβλητή `&HOME_NET` που χρησιμοποιείται από το Snort μέσα στο αρχείο `snort.conf`, για ευκολία ρύθμισής του. Έτσι σαν τιμή σε αυτήν την παράμετρο μπορεί να δοθεί η μεταβλητή του Snort `$HOME_NET`.

2η . Timeout (Προαιρετική)

Η παράμετρος αυτή δεν είναι υποχρεωτική καθώς αν δεν δοθεί τότε θα χρησιμοποιηθεί κάποια default τιμή (Timeout = 3).
Με αυτήν την παράμετρο ορίζεται ο χρόνος σε δευτερόλεπτα, που θα κρατείται ένα *Icmp Echo Request* αποθηκευμένο σε κάποια λίστα, ώστε να χρησιμοποιηθεί στην διαδικασία ανίχνευσης του *spoofing*. Στην ουσία ορίζεται το χρονικό διάστημα μέσα στο οποίο είναι λογικό να αναμένεται κάποια απάντηση για ένα αποθηκευμένο *Icmp Echo Request* πακέτο.
Αν αυτή η τιμή είναι πολύ μεγάλη, τότε θα καθυστερεί περισσότερο η εκτέλεση του preprocessor και θα χρησιμοποιείται μεγαλύτερο ποσό μνήμης. Αυτό γιατί οι λίστες που κρατούν στην μνήμη τα *Echo Request* πακέτα, θα είναι μεγαλύτερες και θα χρειάζεται περισσότερος χρόνος για να γίνονται σε αυτές οι διαδικασίες της εισαγωγής , αναζήτησης και διαγραφής.
Αν αντίθετα η τιμή αυτής της παραμέτρου είναι πολύ μικρή, τότε υπάρχει κίνδυνος να μην ανιχνεύονται κάποια spoofed πακέτα.

3η . Log Dir/File (Προαιρετική)

Η παράμετρος αυτή επίσης δεν είναι υποχρεωτική και αν δεν δοθεί τότε θα χρησιμοποιηθεί κάποια default τιμή.
Η χρήση αυτής της παραμέτρου είναι να ορισθεί ο κατάλογος και το αρχείο στο οποίο θα καταχωρεί ο preprocessor τα alerts που παράγει για τα spoofed πακέτα που εντοπίζει.
Ο κατάλογος πρέπει να δοθεί με την πλήρη διαδρομή στην οποία βρίσκεται και πρέπει να υπάρχει.
Το αρχείο δεν είναι απαραίτητο να υπάρχει καθώς θα το δημιουργήσει ο preprocessor. Αν δεν υπάρχει ο κατάλογος ή για κάποιο άλλο λόγο δεν μπορεί να δημιουργηθεί το αρχείο, τότε το πρόγραμμα θα σταματήσει την εκτέλεσή του καθώς και την εκτέλεση του Snort.
Αν δεν δοθεί τιμή σε αυτή την παράμετρο, τότε τα alerts που παράγονται θα καταγράφονται σε ένα αρχείο με το όνομα `EchoSpoofs`, μέσα στον κατάλογο που καταγράφει τα logs και το ίδιο Snort . Ο κατάλογος αυτός, είναι αυτός που έχει δοθεί σαν τιμή στην μεταβλητή –l για την εκτέλεση του Snort, και αν δεν έχει δοθεί η παράμετρος –l θα είναι ο default κατάλογος καταγραφής του Snort ο `/var/log/snort`.

Καλείται
Από την συνάρτηση `IcmpSpoofInit()`.

Καλεί
Καλεί τις συναρτήσεις `mSplit()`,`mstring.h` , `SplitProtectedNets()` .

Οι δύο πρώτες ανήκουν στο Snort.

Παράμετροι

Δέχεται μία παράμετρο:

Παράμετρος	Λειτουργία
`char **args`	Ένας pointer σε `char` ο οποίος δείχνει σε ένα string το οποίο περιέχει τις παραμέτρους που δόθηκαν από τον χρήστη για την εκτέλεση του preprocessor στο αρχείο `snort.conf`.

Μεταβλητές

Οι τοπικές μεταβλητές της συνάρτησης είναι :

Μεταβλητή	Λειτουργία
`char **toks`	Ένας πίνακας από pointers σε `char`, στον οποίο θα αποθηκευτούν οι παράμετροι με τις οποίες έχει ρυθμιστεί να εκτελεστεί ο preprocessor στο αρχείο `snort.conf`. Ο κάθε pointer θα δείχνει σε ένα string που θα αντιστοιχεί και σε μία παράμετρο. Την τιμή της θα πάρει μετά την εκτέλεση της `mSplit()`.
`int numToks`	Ένας `integer` στον οποίο θα αποθηκευτεί το πλήθος των παραμέτρων του preprocessor. Την τιμή της θα πάρει μετά την εκτέλεση της `mSplit()`.
`Struct ProtectedNets *pr_Net`	Ένας pointer που δείχνει σε δομές τύπου `ProtectedNets` και θα χρησιμοποιηθεί για να είναι δυνατή η πλοήγηση στα μέλη της δυναμικής λίστας που θα περιέχει τα προστατευόμενα δίκτυα που έχουν δοθεί στο `snort.conf`.
`char *LogFile`	Ένας `char` pointer ο οποίος θα δείξει στο string που θα περιέχει τον κατάλογο / αρχείο που θα καταγραφούν τα αποτελέσματα.

Επιστρέφει
Δεν επιστρέφει κάποια τιμή.

```
void IcmpSpoofFunction (Packet *p)
```

Σκοπός / Περιγραφή
Η συνάρτηση αυτή εκτελείται κάθε φορά που κάποιο πακέτο εισέρχεται στο Snort και αφού περάσει την *Decode Engine*.
Μέσα σε αυτήν την συνάρτηση υλοποιείται όλη η λειτουργία του preprocessor.
Πριν δεχτεί κάποιο πακέτο για επεξεργασία ελέγχει ότι το αυτό είναι ένα ολόκληρο πακέτο και όχι κάποιο fragment ενός πακέτου, καθώς επίσης και ότι το συγκεκριμένο πακέτο είναι *Icmp*.

Επίσης για να δεχτεί κάποιο πακέτο προς επεξεργασία ελέγχει αν αυτό προέρχεται ή έχει προορισμό κάποιο από τα προστατευόμενα δίκτυα. Αυτό γίνεται έτσι ώστε ο preprocessor να μην επεξεργάζεται πακέτα που δεν έχουν σχέση με κάποιο από τα προστατευόμενα δίκτυα, σε περίπτωση που αυτά τα πακέτα περνάνε από τον ο `Sensor` στον οποίο εκτελείται το Snort.
Στην περίπτωση που το πακέτο πληρεί τις προϋποθέσεις θα κληθεί συνάρτηση `Get_Dtg_Data()` για να πάρει από το πακέτο τις απαιτούμενες πληροφορίες και στην συνέχεια θα αρχικοποιήσει κατάλληλα τους pointers να δείχνουν στην απαιτούμενη *Inbound/Outbound Λίστα*.
Στην συνέχεια ανάλογα αν για το πακέτο πρέπει να γίνει εισαγωγή σε κάποια λίστα ή αν πρέπει να γίνει έλεγχος για ανίχνευση του *spoofing*, θα κληθούν οι συναρτήσεις `InsertRequest()` ή `CheckResponse()` αντίστοιχα.
Αν κληθεί η `CheckResponse()` θα γίνει έλεγχος για τον αν πρέπει στη συνέχεια να κληθεί η `AlertSpoofs()` ή οποία θα καταγράψει κάποιο alert.
Στη συνέχεια θα κληθούν οι συναρτήσεις `TrashSN()` και `TrashDN()` για να διαγραφούν από την *Inbound/Outbound Λίστα* που χρησιμοποιήθηκε, τα *Echo Requests* που έχει λήξει ο χρόνος παραμονής τους στην λίστα και δεν παρουσιάζουν πλέον ενδιαφέρον. Τέλος θα επαναφερθούν οι pointers που δείχνουν στην αρχή της τρέχουσας λίστας στην σωστή θέση.

Παράμετροι

Παράμετρος	Λειτουργία
`packet` **`*p`**	Είναι ένας pointer ο οποίος δείχνει στην δομή `Packet` του Snort, η οποία περιέχει όλες τις πληροφορίες που αφορούν κάθε πακέτο που εξέρχεται από την *Decode Engine* του Snort.

Μεταβλητές

Οι τοπικές μεταβλητές της συνάρτησης είναι :

Μεταβλητή	Λειτουργία
`int` **`action`**	Ένας `integer` που θα δηλώνει τι είδους ενέργεια θα γίνει για το συγκεκριμένο πακέτο. Δηλαδή αν θα γίνει εισαγωγή σε κάποια από τις *Inbound/Outbound Λίστες* ή αν θα γίνει έλεγχος σε κάποια από αυτές για ανίχνευση του *spoofing*. Την τιμή του θα τη πάρει με την εκτέλεση της `Get_Dtg_Data()`.
`Int` **`fix_Curlist`**	Ένας `integer` που δηλώνει την λίστα που χρησιμοποιήθηκε για το συγκεκριμένο πακέτο. Την τιμή του θα τη πάρει με την εκτέλεση της `Get_Dtg_Data()`.

`Int` **`alertaction`**	Ένας integer που δηλώνει αν μετά την εκτέλεση της `CheckResponse()` βρέθηκε ή όχι το αντίστοιχο πακέτο για το οποίο έγινε η αναζήτηση και κατά συνέπεια αν πρέπει να παραχθεί κάποιο alert . Την τιμή του θα την πάρει με την εκτέλεση της `CheckResponse()`.
`long` **`int limit`**	Αυτή η μεταβλητή θα κρατήσει το αποτέλεσμα του υπολογισμού του μικρότερου χρόνου που μπορεί να έχει ένα πακέτο στις *Inbound/Outbound Λίστες* για να θεωρείται ικανό να παραμείνει σε αυτές.
`time_t` **`arrvtime`**	Αυτή η μεταβλητή αποθηκεύει τον τρέχον χρόνο στον οποίο έφτασε κάποιο πακέτο. Η τιμή αυτής της μεταβλητής θα είναι ο χρόνος του συστήματος σε δευτερόλεπτα που έχουν περάσει από την 01/01/1970. Την τιμή της θα τη πάρει με την εκτέλεση της `Get_Dtg_Data()`.

Καλείται
Καλείται από την συνάρτηση `ParsePreprocessor()` του Snort (source αρχείο `rules.c`), η οποία ελέγχει την λίστα με τα ονόματα (keywords) όλων των preprocessors και εκτελεί την αντίστοιχη συνάρτηση του κάθε preprocessor για κάθε πακέτο.
Το Keyword του συγκεκριμένου preprocessor σε αυτή την λίστα έχει εισαχθεί μετά την εκτέλεση της `IcmpSpoofInit()`.

Καλεί
Καλεί τις συναρτήσεις `Get_Dtg_Data()`, `InsertRequest()`, `CheckResponse()`, `AlertSpoofs()`, `TrashSN()`, `TrashDN()`.

Επιστρέφει
Δεν επιστρέφει κάτι.

```
void Get_Dtg_Data(Packet *p, int *action, int *fix_Curlist,time_t *captime)
```

Σκοπός / Περιγραφή
Η συνάρτηση αυτή υπολογίζει τον χρόνο άφιξης του πακέτου, ελέγχει το *Icmp* πακέτο και ανάλογα με τα χαρακτηριστικά του (τα πεδία *Type* και *Code*, την πηγή και τον προορισμό του) δίνει τιμές στα κατάλληλα πεδία της δομής **`DtgInfo`** η οποία κρατάει από το πακέτο την απαραίτητη μόνο για τον preprocessor πληροφορία και στη συνέχεια ορίζει την ενέργεια που θα γίνει για αυτό το πακέτο και σε ποια λίστα.

Παράμετροι
Δέχεται για παραμέτρους τον pointer προς τη δομή `Packet` όπως την δημιούργησε η *Decode Engine* του Snort, pointers στις μεταβλητές `action`, `fix_Curlist`, `arrvtime` οι οποίες έχουν οριστεί στην `IcmpSpoofFunction()`. Με αυτόν τον τρόπο (by reference) αλλάζει τις τιμές των μεταβλητών που έχουν ορισθεί σε άλλη συνάρτηση.

Μεταβλητές
Δεν έχει τοπικές μεταβλητές.

Καλείται
Από την συνάρτηση `IcmpSpoofFunction()`

Καλεί
Την συνάρτηση `IpIsMynet()`.

Επιστρέφει
Δεν επιστρέφει κάτι.

```
void SplitProtectedNets (char *netargs)
```

Σκοπός / Περιγραφή
Η συνάρτηση αυτή δημιουργεί την λίστα η οποία θα περιέχει τα προστατευόμενα δίκτυα που δόθηκαν σαν παράμετροι από τον χρήστη. Η IP διεύθυνση του κάθε δικτύου είναι το αποτέλεσμα της εφαρμογής, της μάσκας στην IP διεύθυνση, που δόθηκε από τον χρήστη σε CIDR τρόπο γραφής.

Παράμετροι
Η μοναδική παράμετρος που δέχεται είναι ένας *char* pointer ο οποίος δείχνει στο string που δημιουργήθηκε από την `mSplit()` κατά την κλήση της μέσα στην `ParseIcmpSpoofArgs()`, και περιέχει τα προστατευόμενα δίκτυα που δόθηκαν από τον χρήστη.

Μεταβλητές

Οι τοπικές μεταβλητές είναι :

Μεταβλητή	Λειτουργία
`struct ProtectedNets` **`*newPrNet,*curPrNet`**	Είναι δύο pointers από τους οποίους ο 1ος χρησιμοποιείται για την δημιουργία νέων κόμβων-αντικειμένων `ProtectedNets` που θα σχηματίσουν την λίστα με τα προστατευόμενα δίκτυα, ενώ ο 2ος χρησιμοποιείται για την πλοήγηση στην λίστα που δημιουργείται.
`char` **`**net_toks`**	Ένας πίνακας από pointers σε `char` στον οποίο θα αποθηκευτούν τα προστατευόμενα δίκτυα που δόθηκαν από τον χρήστη για την περαιτέρω επεξεργασία τους. Ο κάθε pointer θα δείχνει σε ένα string που θα αντιστοιχεί και σε ένα δίκτυο. Την τιμή της θα πάρει μετά την εκτέλεση της `mSplit()`.
`int` **`net_numToks`**	Ένας `integer` στον οποίο θα αποθηκευτεί το πλήθος των προστατευόμενων δικτύων που δόθηκαν. Την τιμή της θα πάρει μετά την εκτέλεση της `mSplit()`.
`int` **`given_nets`**	Ένας προσωρινός, βοηθητικός `Integer`.

Καλείται
Από την συνάρτηση `ParseIcmpSpoofArgs()`.

Καλεί
Τις συναρτήσεις `mSplit()`, `GenProtectedNets()`.

Επιστρέφει
Δεν επιστρέφει κάτι.

```
struct ProtectedNets GenProtectedNets (char *net_tok)
```

Σκοπός / Περιγραφή
Η συνάρτηση αυτή δέχεται ένα string το οποίο έχει προκύψει μετά την εκτέλεση της `SplitProtectedNets()` και περιέχει την IP διεύθυνση και την μάσκα ενός προστατευόμενου δικτύου σε CIDR τρόπο γραφής. Η συνάρτηση θα υπολογίζει την IP διεύθυνση αφού εφαρμόσει την μάσκα.

Παράμετροι
Δέχεται σαν παράμετρο ένα `char pointer` ο οποίος δείχνει σε ένα string το οποίο περιέχει μία IP διεύθυνση και μία μάσκα σε CIDR notation.

Μεταβλητές

Οι τοπικές μεταβλητές είναι :

Μεταβλητή	Λειτουργία
`struct ProtectedNets` **`PrNet`**	Είναι η δομή στην οποία θα αποθηκευτεί η επεξεργασμένη IP διεύθυνση.
`char` **`**toks`**	Ένας πίνακας από pointers σε `char` στον οποίο θα αποθηκευτούν η IP διεύθυνση και η μάσκα αφού διαχωριστούν μέσα από το string στο οποίο δείχνει ο pointer `*net_tok`. Ο κάθε pointer θα δείχνει σε ένα string που θα αντιστοιχεί και σε ένα δίκτυο ή σε μία μάσκα. Την τιμή της θα πάρει μετά την εκτέλεση της `mSplit()`.
`int` **`numToks`**	Ένας `integer` ο οποίος θα πάρει την τιμή 2 αφού διαχωριστεί η IP διεύθυνση από την μάσκα. Την τιμή του θα την πάρει μετά την εκτέλεση της `mSplit()`.

Καλείται
Από την συνάρτηση `SplitProtectedNets()`

Καλεί
Την συνάρτηση `mSplit()`.

Επιστρέφει
Επιστρέφει μία δομή τύπου `ProtectedNets` η οποία θα περιέχει την επεξεργασμένη IP διεύθυνση και την μάσκα του δικτύου.

```
int IpIsMynet (struct in_addr ipaddr)
```

Σκοπός / Περιγραφή
Η συνάρτηση ελέγχει αν μία IP διεύθυνση ανήκει σε ένα από τα προστατευόμενα δίκτυα . Εκτελείται έτσι ώστε να ελέγξει αν το πακέτο που εντοπίστηκε έχει διεύθυνση αποστολέα ή παραλήπτη, μία IP διεύθυνση που ανήκει στα προστατευόμενα δίκτυα που έδωσε ο χρήστης.

Παράμετροι
Δέχεται σαν παράμετρο μία δομή τύπου `in_addr` κατάλληλη για να αποθηκεύει μια IP διεύθυνση.

Μεταβλητές
Οι τοπικές μεταβλητές είναι :

Μεταβλητή	Λειτουργία
`struct ProtectedNets *curPrNet`	Ένας pointer σε δομή `ProtectedNets`, ο οποίος χρησιμοποιείται για πλοήγηση στην λίστα με τα προστατευόμενα δίκτυα.

Καλείται
Από τις συναρτήσεις `IcmpSpoofFunction()`, `Get_Dtg_Data()`.

Καλεί
Δεν καλεί κάποια συνάρτηση.

Επιστρέφει
Επιστρέφει 0 αν η δοσμένη IP διεύθυνση δεν ανήκει σε κάποιο από τα προστατευόμενα δίκτυα και 1 αν ανήκει.

```
void AlertSpoofs (Packet *p, int alert , int CurrentList,
                  time_t *alerttime )
```

Σκοπός / Περιγραφή
Η συνάρτηση αυτή είναι υπεύθυνη ώστε να γράψει το σωστό alert μέσα στο αρχείο καταγραφής των alerts που όρισε ο χρήστης.

Παράμετροι

Δέχεται σαν παραμέτρους :

Παράμετρος	Λειτουργία

`Packet` ***p**	Ένας pointer προς τη δομή `Packet` την οποία δημιούργησε η *Decode Engine* του Snort
`int` **alert**	Έναν `integer` που δηλώνει αν βρέθηκε ή όχι το πακέτο για το οποίο έγινε αναζήτηση μέσω της `CheckResponse()`, σε μία από τις *Inbound/Outbound Λίστες* και κατά συνέπεια αν πρέπει να παραχθεί κάποιο alert. Παίρνει την τιμή που θα έχει επιστρέψει η `CheckResponse()` και μπορεί να είναι 0 ή 1.
`int` **CurrentList**	Δηλώνει σε ποια λίστα έγινε η αναζήτηση για το συγκεκριμένο πακέτο. Παίρνει την τιμή που θα έχει δημιουργήσει η `Get_Dtg_Data()` και μπορεί να είναι 1 ή 2 .
`time_t` **alerttime**	Είναι ο χρόνος καταγραφής του πακέτου.

Μεταβλητές

Οι τοπικές μεταβλητές είναι :

Μεταβλητή	Λειτουργία
`char` ***** **Scenario**	Ένας `char` pointer που δείχνει σε ένα string το οποίο θα περιέχει το σενάριο που θα τυπωθεί.

Καλείται
Από την συνάρτηση `IcmpSpoofFunction()` .

Καλεί
Δεν καλεί κάποια συνάρτηση.

Επιστρέφει
Δεν επιστρέφει κάτι.

```
void InsertRequest (long int limit)
```

Σκοπός / Περιγραφή
Η συνάρτηση αυτή είναι υπεύθυνη ώστε να κάνει εισαγωγή σε κάποια από τις *Inbound/Outbound Λίστες* (και στο σωστό σημείο της), τις απαραίτητες πληροφορίες για ένα *Icmp Echo Request* πακέτο. Επίσης κατά την διαδικασία της εισαγωγής γίνεται έλεγχος για τον καθαρισμό της λίστας, δηλαδή προετοιμασία για διαγραφή των *Echo Request* πακέτων που έχει λήξει ο χρόνος παραμονής τους στην λίστα και δεν θεωρούνται πλέον υποψήφια για να ελεγχθούν.

Αν δεν υπάρχει κάποιος κόμβος στην λίστα τότε θα κάνει εισαγωγή στην αρχή.

Σε αντίθετη περίπτωση η εισαγωγή γίνεται στο τέλος της λίστας. Με αυτόν τον τρόπο ο τελευταίος αποστολέας ενός *Icmp Echo Request* πακέτου θα αποθηκευτεί στο τέλος της λίστας μαζί με όλους τους παραλήπτες του, στους οποίους έχει στείλει τέτοια πακέτα.

Πριν από κάθε εισαγωγή αρχικά ελέγχεται αν έχει λήξει ο χρόνος παραμονής στην λίστα του τελευταίου αποθηκευμένου αποστολέα και αν ισχύει κάτι τέτοιο τότε προετοιμάζεται η λίστα για ολική διαγραφή των κόμβων της και γίνεται εισαγωγή στην αρχή, αλλιώς ξεκινώντας από την αρχή της λίστας και περνώντας από κάθε κόμβο της σειριακά, ελέγχεται αν υπάρχει ήδη ο αποστολέας αποθηκευμένος .
Αρχικά για όσους αποθηκευμένους αποστολείς έχει λήξει ο χρόνος παραμονής τους στην λίστα , θα προετοιμαστούν για διαγραφή όπως και όλοι οι παραλήπτες τους και η αναζήτηση θα συνεχιστεί στους αποστολείς που έχουν απομείνει.
Σε περίπτωση που δεν βρεθεί ο αποστολέας αποθηκευμένος στην λίστα τότε θα γίνει νέα εισαγωγή στο τέλος της λίστας.
Αν βρεθεί ο αποστολέας, τότε ξεκινώντας από την αρχή της λίστας με τους παραλήπτες του συγκεκριμένου αποστολέα ($2^{η}$ διάσταση), γίνεται αναζήτηση για το αν υπάρχει ήδη αποθηκευμένος ο παραλήπτης του πακέτου. Αρχικά για όσους αποθηκευμένους παραλήπτες έχει λήξει ο χρόνος παραμονής τους στην λίστα , θα προετοιμαστούν για διαγραφή και η αναζήτηση θα συνεχιστεί στους παραλήπτες που έχουν απομείνει.
Αν βρεθεί ίδιος παραλήπτης στην λίστα τότε αυτός θα πάρει τον νέο χρόνο καταγραφής του πακέτου, θα μεταφερθεί στο τέλος της $2^{ης}$ διάστασης της λίστας και ο αποστολέας του μαζί με όλους τους παραλήπτες του θα μεταφερθούν στο τέλος της $1^{ης}$ διάστασής της λίστας. Με αυτόν τον τρόπο το τελευταίο *Icmp Echo Request* που εντοπίστηκε θα είναι αποθηκευμένο στο τέλος της λίστας.
Αν δεν βρεθεί ίδιος παραλήπτης τότε θα γίνει νέα εισαγωγή στο τέλος της λίστας.

Παράμετροι

Δέχεται σαν παραμέτρους :

Παράμετρος	Λειτουργία
`long int` **`limit`**	Αυτή η μεταβλητή θα κρατάει το αποτέλεσμα του υπολογισμού του μικρότερου χρόνου που μπορεί να έχει ένα πακέτο στις λίστες για να θεωρείται άξιο προς έλεγχο.

Μεταβλητές

Οι τοπικές μεταβλητές είναι :

Μεταβλητή	Λειτουργία
`int` **`FlagSN, FlagDN`**	Είναι δύο `integers` οι οποίοι θα παίρνουν την τιμή 0 ή 1 δηλώνοντας αν έχει βρεθεί ή όχι, ο αποστολέας (`FlagSN`) ή ο παραλήπτης (`FlagDN`), μετά την διαδικασία αναζήτησης στις λίστες.

Καλείται
Από την συνάρτηση `IcmpSpoofFunction()` .

Καλεί
Τις συναρτήσεις : `InsertNode_SN()`, `InsertNode_DN()`, `EjectCurNode_SN()`, `Prepare_To_Clean_SN()`, `SearchNode_SN()`.

Επιστρέφει
Δεν επιστρέφει κάτι.

```
int CheckResponse (long int limit)
```

Σκοπός / Περιγραφή
Η συνάρτηση αυτή είναι υπεύθυνη για τον έλεγχο της ανίχνευσης του *spoofing*, εκτελώντας την αναζήτηση σε μία από τις *Inbound/Outbound Λίστες,* για να εντοπίσει αν υπάρχει το αντίστοιχο αποθηκευμένο *Icmp Echo Request* πακέτο, με κάποιο *Icmp Echo Reply* ή *Icmp Destination Unreachable* που κατέφθασε.
Επίσης κατά την διαδικασία της αναζήτησης γίνεται και έλεγχος για τον καθαρισμό της λίστας, δηλαδή προετοιμασία για διαγραφή των *Echo Request* πακέτων που έχει λήξει ο χρόνος παραμονής τους στην λίστα και δεν θεωρούνται πλέον υποψήφια για να ελεγχθούν.
Στην ουσία αυτή είναι η συνάρτηση που εντοπίζει το *spoofing* όταν συμβαίνει, καθώς δεν μπορεί να αντιστοιχήσει τα πακέτα που καταφθάνουν με τα *Icmp Echo Request* πακέτα που είναι αποθηκευμένα.

Η συνάρτηση αυτή έχει σχεδόν την ίδια δομή και λειτουργία με την `InsertRequest()`, με την διαφορά ότι αν δεν βρει το αντίστοιχο πακέτο με αυτό που ψάχνει, τότε θα επισημάνει την ανάγκη δημιουργίας ενός alert, αλλιώς τον κόμβο που εντόπισε θα τον αφήσει ως έχει και δεν θα κάνει τίποτα, εκτός από το να προετοιμάσει την λίστα για την διαγραφή των κόμβων που έχει λήξει ο χρόνος παραμονής τους.

Παράμετροι

Δέχεται σαν παραμέτρους :

Παράμετρος	Λειτουργία
`long int` **`limit`**	Αυτή η μεταβλητή θα κρατάει το αποτέλεσμα του υπολογισμού, του μικρότερου χρόνου που μπορεί να έχει ένα πακέτο στις λίστες, για να μπορεί να παραμείνει σε αυτές.

Μεταβλητές

Οι τοπικές μεταβλητές είναι :

Μεταβλητή	Λειτουργία
`Int` **`FlagSN, FlagDN`**	Είναι δύο `integers` οι οποίοι θα παίρνουν την τιμή 0 ή 1 δηλώνοντας αν έχει βρεθεί ή όχι ο αποστολέας (`FlagSN`) ή ο παραλήπτης (`FlagDN`) , μετά την διαδικασία αναζήτησης στις λίστες.

Καλείται
Από την συνάρτηση `IcmpSpoofFunction()`.

Καλεί
Τις συναρτήσεις : `EjectCurNode_SN()`,`Prepare_To_Clean_SN()`, `SearchNode_SN()`.

Επιστρέφει
Επιστρέφει μία ακέραια τιμή, την τοπική μεταβλητή alert, η οποία παίρνει τις τιμές 1 ή 0 και δηλώνει αν το αντίστοιχο πακέτο βρέθηκε ή όχι στη λίστα και κατά συνέπεια αν απαιτείται να δημιουργηθεί κάποιο alert.

```
void InsertNode_SN ()
```

Σκοπός / Περιγραφή
Η συνάρτηση αυτή υλοποιεί την εισαγωγή ενός κόμβου-αποστολέα σε μία από τις *Inbound/Outbound Λίστες*, κατά την άφιξη ενός *Icmp Echo Request* πακέτου.

Η εισαγωγή μπορεί να γίνει με τρεις τρόπους :

1 . Αλλαγή θέσης ενός αποστολέα στην λίστα .

Για να συμβεί αυτό πρέπει ο νέος αποστολέας να είναι ήδη καταχωρημένος στην λίστα από προηγούμενο *Echo Request* που έστειλε.
Σε αυτήν την περίπτωση απλά θα μεταφερθεί στο τέλος της λίστας μαζί με όλους τους παραλήπτες στους οποίους έχει στείλει *Echo Request* πακέτα.

2. Χρησιμοποιώντας τον τελευταίο κόμβο από τους ανακυκλώσιμους.

Αυτή η περίπτωση ισχύει όταν ο νέος αποστολέας δεν είναι ήδη καταχωρημένος στην λίστα και απαιτείται να δημιουργηθεί ένας νέος κόμβος.
Σε αυτήν την περίπτωση ο νέος κόμβος θα είναι ο τελευταίος από τους κόμβους που βρίσκονται στην *Λίστα Ανακύκλωσης*.
Ο νέος κόμβος θα κρατήσει την πληροφορία για τον νέο αποστολέα και θα τοποθετηθεί στο τέλος της λίστας.

3. Δημιουργία νέου κόμβου.

Αυτή η περίπτωση επίσης ισχύει όταν ο νέος αποστολέας δεν είναι ήδη καταχωρημένος στην λίστα και απαιτείται να δημιουργηθεί ένας νέος κόμβος, αλλά δεν υπάρχει κάποιος διαθέσιμος στην *Λίστα Ανακύκλωσης*.
Σε αυτή την περίπτωση για την δημιουργία του νέου κόμβου θα δεσμευτεί νέο κομμάτι μνήμης από τον σωρό (heap).
Ο νέος κόμβος θα κρατήσει την πληροφορία για τον νέο αποστολέα και θα τοποθετηθεί στο τέλος της λίστας.

Παράμετροι
Δεν δέχεται παραμέτρους .

Μεταβλητές
Δεν έχει τοπικές μεταβλητές.

Καλείται
Από την συνάρτηση `InsertRequest()`.

Καλεί
Τις συναρτήσεις : `InsertNode_DN()`.

Επιστρέφει
Δεν επιστρέφει κάτι.

```
void InsertNode_DN ()
```

Σκοπός / Περιγραφή
Η συνάρτηση αυτή υλοποιεί την εισαγωγή ενός κόμβου-παραλήπτη σε μία από τις *Inbound/Outbound Λίστες*, κατά την άφιξη ενός *Icmp Echo Request* πακέτου.
Η εισαγωγή μπορεί να γίνει με τρεις τρόπους :

1 . Αλλαγή θέσης ενός παραλήπτη στην λίστα .

Για να συμβεί αυτό πρέπει ο νέος παραλήπτης να είναι ήδη καταχωρημένος στην λίστα με τους παραλήπτες του συγκεκριμένου αποστολέα (1η διάσταση) από προηγούμενο *Icmp Echo Request* που δέχτηκε από αυτόν τον αποστολέα.
Σε αυτήν την περίπτωση απλά θα μεταφερθεί στο τέλος της λίστας των παραληπτών του συγκεκριμένου αποστολέα και θα αλλάξει η τιμή του πεδίου του χρόνου, ώστε να περιέχει τον νέο χρόνο.

2. Χρησιμοποιώντας τον τελευταίο κόμβο από τους ανακυκλώσιμους.

Αυτή η περίπτωση ισχύει όταν ο νέος παραλήπτης δεν είναι ήδη καταχωρημένος στην λίστα παραληπτών για τον συγκεκριμένο αποστολέα και απαιτείται να δημιουργηθεί ένας νέος κόμβος.
Σε αυτήν την περίπτωση ο νέος κόμβος θα είναι ο τελευταίος από τους κόμβους που βρίσκονται στην *Λίστα Ανακύκλωσης*.
Ο νέος κόμβος θα κρατήσει την πληροφορία για τον νέο παραλήπτη, θα πάρει την τρέχον ώρα στο πεδίο του χρόνου και θα τοποθετηθεί στο τέλος της λίστας.

3. Δημιουργία νέου κόμβου.

Αυτή η περίπτωση επίσης ισχύει όταν ο νέος παραλήπτης δεν είναι ήδη καταχωρημένος στην λίστα παραληπτών για τον συγκεκριμένο αποστολέα και απαιτείται να δημιουργηθεί ένας νέος κόμβος, αλλά δεν υπάρχει κάποιος διαθέσιμος στην *Λίστα Ανακύκλωσης*.
Σε αυτή την περίπτωση για την δημιουργία του νέου κόμβου θα δεσμευτεί νέο κομμάτι μνήμης από τον σωρό (heap).
Ο νέος κόμβος θα κρατήσει την πληροφορία για τον νέο παραλήπτη, θα πάρει την τρέχον ώρα στο πεδίο του χρόνου και θα τοποθετηθεί στο τέλος της λίστας.

Παράμετροι
Δεν δέχεται παραμέτρους .

Μεταβλητές
Δεν έχει τοπικές μεταβλητές.

Καλείται
Από τις συναρτήσεις `InsertNode_SN()`,`InsertRequest()`.

Καλεί
Δεν καλεί κάποια συνάρτηση.

Επιστρέφει
Δεν επιστρέφει κάτι.

`void` **`SearchNode_SN ()`**`,` `void` **`SearchNode_DN ()`**

Σκοπός / Περιγραφή
Η συναρτήσεις αυτές υλοποιούν την διαδικασία της αναζήτησης σε μία από τις *Inbound/Outbound Λίστες* με στόχο την εύρεση ίδιου αποστολέα ή παραλήπτη αντίστοιχα, με αυτόν που αναφέρεται στο πακέτο, προς εισαγωγή ή προς έλεγχο για *spoofing*.
Επίσης κατά την αναζήτηση που κάνουν, υλοποιούν και την διαδικασία ελέγχου του χρόνου παραμονής και επισημαίνουν ποιοι αποστολείς οι παραλήπτες πρέπει να διαγραφούν από την λίστα καθώς έχει λήξει ο χρόνος παραμονής τους.

Παράμετροι
Δεν δέχονται παραμέτρους .

Μεταβλητές
Δεν έχουν τοπικές μεταβλητές.

Καλείται
Από τις συναρτήσεις `InsertRequest()`, `CheckResponse()`.

Καλεί
Δεν καλούν κάποια συνάρτηση.

Επιστρέφει
Δεν επιστρέφουν κάτι.

`void` **`EjectCurNode_SN ()`**`,` `void` **`EjectCurNode_DN ()`**

Σκοπός / Περιγραφή
Η συναρτήσεις αυτές αποδεσμεύουν από την λίστα έναν κόμβο-αποστολέα ή κόμβο-παραλήπτη αντίστοιχα, για να μεταφερθεί στο τέλος της λίστας, με την εκτέλεση της `InsertNode_SN()` ή της `InsertNode_DN()`, όταν καταφθάσουν *Icmp Echo Request* πακέτα που τους ανανεώνουν

με νέες πληροφορίες. Ο κόμβος που θα μεταφερθεί στο τέλος έχει την ίδια διεύθυνση αποστολέα ή παραλήπτη με αυτή του νέου πακέτου.

Παράμετροι
Δεν δέχεται παραμέτρους .

Μεταβλητές
Δεν έχει τοπικές μεταβλητές.

Καλείται
Από την συνάρτηση `InsertRequest()`.

Καλεί
Δεν καλεί κάποια συνάρτηση.

Επιστρέφει
Δεν επιστρέφει κάτι.

`void` **`Prepare_To_Clean_SN ()`** , `void` **`Prepare_To_Clean_DN ()`**

Σκοπός / Περιγραφή
Οι συναρτήσεις αυτές παίρνουν υπόψη τις επισημάνσεις της `SearchNode_SN()` ή `SearchNode_DN()` και προετοιμάζουν τις *Inbound/Outbound Λίστες* ώστε να αποδεσμευτούν ένας ή περισσότεροι κόμβοι-αποστολέων ή κόμβοι-παραληπτών αντίστοιχα, για να διαγραφούν από τις `Trash_SN()`, `Trash_DN()`.

Παράμετροι
Δεν δέχονται παραμέτρους .

Μεταβλητές
Δεν έχουν τοπικές μεταβλητές.

Καλείται
Από την συνάρτηση `InsertRequest()`.

Καλεί
Δεν καλούν κάποια συνάρτηση.

Επιστρέφει
Δεν επιστρέφουν κάτι.

`void` **`Trash_SN()`**, `void` **`Trash_DN()`**

Σκοπός / Περιγραφή
Οι συναρτήσεις αυτές λαμβάνουν υπόψη τους τα αποτελέσματα των `Prepare_To_Clean_SN()`, `Prepare_To_Clean_DN()` και εκτελούν την διαγραφή των κόμβων-αποστολέων και των κόμβων-παραληπτών αντίστοιχα από τις *Inbound/Outbound Λίστες*, για τους οποίους έχει λήξει ο χρόνος παραμονής τους στην λίστα.
Η διαγραφή έχει σαν αποτέλεσμα οι κόμβοι, να μεταφέρονται στην αντίστοιχη *Λίστα Ανακύκλωσης,* εφόσον ο μετρητής που κρατάει τον αριθμό των ανακυκλωμένων κόμβων (maxSN , maxDN) δεν έχει φτάσει την μέγιστη τιμή του.
Εάν συμβεί κάτι τέτοιο, τότε οι κόμβοι αυτοί διαγράφονται ολοκληρωτικά αποδεσμεύοντας το τμήμα του σωρού (heap) της μνήμης που είχε δεσμευτεί για αυτούς.

Παράμετροι
Δεν δέχονται παραμέτρους .

Μεταβλητές

Οι τοπικές μεταβλητές για αυτές τις συναρτήσεις είναι:

Μεταβλητή	Λειτουργία
`int` **`maxSN`** (Για την `TrashSN`)	Προκαθορισμένος αριθμός που ορίζει το μέγιστο πλήθος των κόμβων-αποστολέων που μπορούν να υπάρχουν στην *Λίστα Ανακύκλωσης*.
`int` **`maxDN`** (Για την `TrasηDN`)	Προκαθορισμένος αριθμός που ορίζει το μέγιστο πλήθος των κόμβων-παραληπτών που μπορούν να υπάρχουν στην *Λίστα Ανακύκλωσης*.

Καλείται
Από την συνάρτηση `IcmpSpoofFunction()`.
Η `Trash_DN()` καλείται και από την `Trash_SN()`.

Καλεί
Δεν καλούν κάποια συνάρτηση.
Η `Trash_SN()` καλεί την `Trash_DN()`.

Επιστρέφει
Δεν επιστρέφουν κάτι.