Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων
Π.Μ.Σ. " Ψηφιακές Επικοινωνίες & Δίκτυα "

# Διπλωματική Εργασία

## Μελέτη των Μηχανισμών Ασφαλείας στο Λειτουργικό Σύστημα Unix

Παντελής Σπυρίδων
ΑΜ: ΜΨΕ1806

Επιβλέπων Καθηγητής
Μηλιώνης Απόστολος

Πειραιάς, Σεπτέμβριος 2021

# ***Ευχαριστίες***

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον Αναπληρωτή Καθηγητή κ. Δρ. Μηλιώνη Απόστολο για την δυνατότητα που μου έδωσε να πραγματοποιήσω την διπλωματική μου εργασία και για το πολύτιμο χρόνο που διέθεσε για την περάτωση της παρούσας εργασίας. Οι σημαντικές υποδείξεις και συμβουλές του με κατεύθυναν σ' ένα σωστό τρόπο σκέψης πάνω απ' όλα και μου προσέφεραν σημαντικά εφόδια για την μετέπειτα ζωή μου.

Τέλος, θέλω να εκφράσω ένα τεράστιο ευχαριστώ στην οικογένεια μου, για την στήριξη και την εμπιστοσύνη που μου έδειξε όλα αυτά τα χρόνια των σπουδών μου. Πέραν όμως από την πολύτιμη αυτή στήριξη, μου έδωσαν όλα τα εφόδια ώστε να γίνω ένας σωστός Άνθρωπος και αυτό είναι κάτι που δεν μαθαίνεται, αλλά μεταδίδεται.

**Πίνακας περιεχομένων**

## Πίνακας Εικόνων

# ΚΕΦΑΛΑΙΟ 1 ΛΟΓΙΣΜΙΚΟ ΣΥΣΤΗΜΑΤΟΣ

## 1.1. Η ανάπτυξη του λογισμικού συστήματος: μια ιστορική αναδρομή

Το λογισμικό (software) περιλαμβάνει το σύνολο προγραμμάτων (ένα σύνολο εντολών που καθοδηγεί λεπτομερώς έναν υπολογιστή ώστε να εκτελέσει συγκεκριμένες εργασίες, οι οποίες είτε εκτελούνται αυτόματα, είτε έπειτα από αίτηση κάποιου χρήστη), διαδικασιών και ρουτινών που σχετίζονται με τη λειτουργία ενός συστήματος υπολογιστή. Ο όρος επινοήθηκε για τη διαφοροποίηση αυτών των οδηγιών από το υλικό - δηλαδή, τα φυσικά στοιχεία ενός συστήματος υπολογιστή. Το υλικό (hardware) μαζί με το λογισμικό, αποτελούν ένα ολοκληρωμένο υπολογιστικό σύστημα (Ceruzzi, 2011). Το λογισμικό ως σύνολο προγραμματισμένων οδηγιών αποθηκευμένων στη μνήμη των ψηφιακών υπολογιστών, για εκτέλεση από τον επεξεργαστή, αποτελεί μια πρόσφατη εξέλιξη στην ανθρώπινη ιστορία η οποία είναι θεμελιώδης για την εποχή της πληροφορίας (Mindell, 2018).

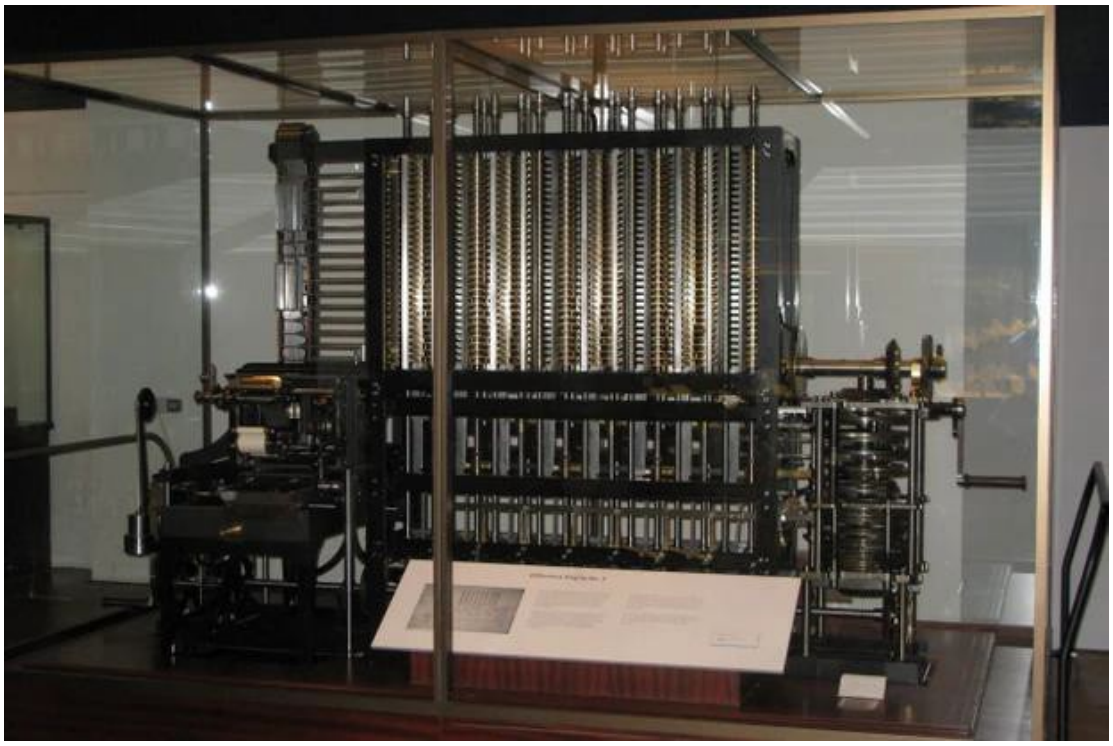
Η μαθηματικός Ada Lovelace (Augusta Ada King, Countess of Lovelace), είναι ένα όνομα που έχει συνδεθεί στενά με τις πρώτες εξελίξεις στον υπολογιστή. Αυτό οφείλεται στη συνεργασία της με τον πολυμαθή Charles Babbage το 1833 όταν ενδιαφέρθηκε για ένα μοντέλο που ο Babbage είχε κατασκευάσει από μια μηχανική συσκευή για τον υπολογισμό τιμών τετραγωνικών συναρτήσεων, της μηχανής διαφορών (Hammerman & Russell, 2015). Παράλληλα η Lovelace ενδιαφέρθηκε για τις ιδέες του σε μια άλλη μηχανή, την γνωστή ως «Αναλυτική Μηχανή» η οποία μπορούσε να προγραμματιστεί με τη χρήση διάτρητων καρτών για να «διαβάσει» οδηγίες και δεδομένα για την επίλυση μαθηματικών προβλημάτων (Essinger, 2014).

Ο Babbage άρχισε να εργάζεται στην Αναλυτική Μηχανή του στα μέσα της δεκαετίας του 1830, με την ιδέα να δημιουργήσει μια νέα μηχανή υπολογισμού που θα μπορούσε να «τρώει τη ουρά της», δηλαδή εννοούσε ότι μπορούσε να τροποποιήσει τους υπολογισμούς της κατά την διάρκεια την εκτέλεσης των λειτουργιών της. Αυτό ήταν δυνατό να συμβεί με την παύση κατά τη διάρκεια ενός υπολογισμού και την χρήση των τιμών που ο Babbage, είχε ήδη αποφασίσει να επιλέξει μεταξύ δύο πιθανών επόμενων βημάτων (Fuegi & Francis, 2013)

Ο Babbage, απαριθμούσε τις βασικές λειτουργίες που ένα τέτοιο μηχάνημα, με αρκετά μεγάλη μνήμη, θα χρειαζόταν αν εκτελούσε «το σύνολο των εξελίξεων και των λειτουργιών της ανάλυσης», με άλλα λόγια οποιονδήποτε υπολογισμό που θα μπορούσε να γίνει αντιληπτός τότε. Σήμερα γνωρίζουμε, ότι οι βασικές λειτουργίες που σε εκείνη την περίοδο περιέγραψε ο Babbage, είναι αυτές που αποτελούν απαραίτητη προϋπόθεση για την εκτέλεση οποιουδήποτε υπολογισμού από ένα σύγχρονο μηχάνημα. Αυτό σημαίνει ότι η Αναλυτική Μηχανή θα ήταν, με σύγχρονους όρους, ένας υπολογιστής γενικής χρήσης, μια έννοια που προσδιορίστηκε για πρώτη φορά από τον Alan Turing στη συνέχεια, τη δεκαετία του 1930 (Ord-Smith, 2011).

Η Αναλυτική Μηχανή δεν κατασκευάστηκε ποτέ, αλλά πολλές πτυχές του σχεδιασμού της καταγράφηκαν με άψογη λεπτομέρεια στα σχέδια και τη μηχανική σημειογραφία του Babbage. Έπρεπε να προγραμματιστεί με κάρτες διάτρησης, παρόμοιες με αυτές που χρησιμοποιούνται στους αργαλειούς ύφανσης που σχεδίασε ο Joseph Marie Jacquard. Ξεχωριστές τράπουλες από κάρτες αποτελούσαν αυτό που θα αποκαλούσαμε σήμερα «πρόγραμμα» και έδιναν τις αρχικές τιμές για τους υπολογισμούς. Ένας σύνθετος μηχανισμός επέτρεπε στο μηχάνημα να επαναλάβει την εκτέλεση ενός συνόλου καρτών (μιας τράπουλας), ώστε να εκτελέσει βρόχο (Brattka, 2016).

Το υλικό περιλάμβανε πολλούς νέους και περίπλοκους μηχανισμούς και σχεδιάστηκε σε μαζική κλίμακα. Η κεντρική μονάδα επεξεργασίας, την οποία ο Babbage ονόμασε Μύλο ( Mill ) θα είχε ύψος δεκαπέντε πόδια, η μνήμη ( store ) που θα μπορούσε να κρατήσει εκατό 50-ψήφους αριθμούς θα έχει μήκος είκοσι πόδια ( 6m ) ενώ άλλα στοιχεία περιλάμβαναν έναν εκτυπωτή, μια κάρτα διάτρησης και έναν σχεδιαστή γραφικών. Ο Babbage εκτιμούσε ότι θα χρειαζόντουσαν τρία λεπτά για να πολλαπλασιαστούν δύο 20ψήφιοι αριθμοί (Babbage, 1830).



**Εικόνα 1 Μια σύγχρονη έκδοση της Αναλυτικής Μηχανής του Babbage**

Lovelace και του Babbage παρέμειναν μόνο θεωρητικές, ο Alan Turing είναι ο πρώτος ο οποίος παρουσίασε μια θεωρία για το λογισμικό το 1935, η οποία οδήγησε στους δύο ακαδημαϊκούς τομείς της επιστήμης των υπολογιστών και της μηχανικής λογισμικού. Ο Alan Turing (1912–1954) συχνά αναφέρεται ως ο «Ιδρυτής της Πληροφορικής» (Bowen, 2017). Έχει επίσης ονομαστεί «Πατέρας της Πληροφορικής» (Daylight, 2015).

Πριν από τον Δεύτερο Παγκόσμιο Πόλεμο, ο Turing έθεσε τα θεωρητικά θεμέλια για μια καθολική μηχανή που θα διαμόρφωνε έναν υπολογιστή στη γενικότερη μορφή του. Κατά τη διάρκεια του πολέμου, η συμβολή του Turing ήταν καθοριστική για την ανάπτυξη των υπολογιστικών συσκευών. Λέγεται μάλιστα ότι η συνεισφορά του Turing συντόμευσε τον πόλεμο έως και δύο χρόνια αποκωδικοποιώντας κρυπτογραφημένα εχθρικά μηνύματα που πιστεύεται γενικά ότι είναι απαραβίαστα, και ειδικότερα «έσπασε» τον μυστικό κωδικό Enigma των Γερμανών (Copeland, 2012).

Το 1936 ο Turing δημοσίευσε το έργο του «On Computable Numbers, with an Application to the Entscheidungs problem», στο οποίο υποστηρίζει ότι υπάρχουν

ορισμένα μαθηματικά προβλήματα τα οποία μπορούν να επιλυθούν μέσω μιας σταθερά καθορισμένης εκ των προτέρων διαδικασίας, την οποία χαρακτήριζε ως διεργασία και η οποία μπορεί να εκτελεστεί από αυτόματη μηχανή (Bowen et al., 2018).

Ο Turing υποστήριξε περαιτέρω, την δυνατότητα κατασκευής μιας μηχανής γενικής χρήσης («Turing Machine»), η οποία, μέσω του κατάλληλου προγραμματισμού θα ήταν σε θέση να εκτελέσει το έργο οποιασδήποτε μηχανής η οποία θα είχε κατασκευαστεί για την επίλυση ειδικών προβλημάτων. Η ιδέα του Turing για την δημιουργία μια μηχανής γενικής χρήσης αποτέλεσε τη θεωρητική βάση για τους ηλεκτρονικούς υπολογιστές, που εμφανίστηκαν αργότερα τη δεκαετία του 1940 (Bonhams, 2017).

Εξελίσσοντας ακόμα περισσότερο την ιδέα του, ο Turing, δημιουργεί και τα πρώτα προγράμματα τα οποία ήταν φτιαγμένα με τέτοιο τρόπο ώστε να μπορούν να διαβαστούν από μια μηχανή. Ειδικότερα, ο Turing «έγραφε» προγράμματα σε μορφή διάτρητων ταινιών, τις οποίες στη συνέχεια «διάβαζε» σειριακά μια μηχανή, η οποία σύμφωνα με αυτό που διάβαζε, εκτελούσε μια συγκεκριμένη ενέργεια (Bullynck et al., 2015).

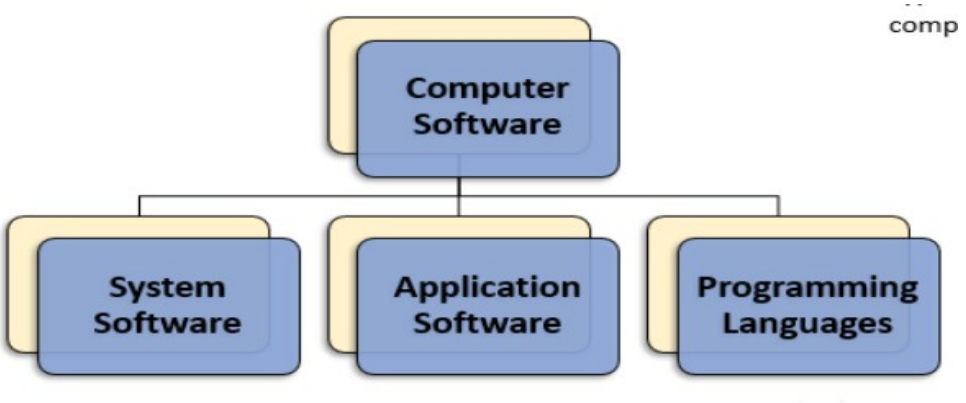
Η πρώτη αυτή νοητή μηχανή την οποία ο Turing δημιούργησε το 1937 είναι ο πρόδρομος του σημερινού σύγχρονου υπολογιστή, ενώ οι διάτρητες ταινίες που δημιούργησε, είναι τα σύγχρονα προγράμματα. Αν και η εξέλιξη της επιστήμης των υπολογιστών έκτοτε υπήρξε τεράστια ως σήμερα, η βασική αρχή παραμένει η ίδια με το νοητικό πείραμα του Touring στην ουσία δηλαδή ίδια με την «Τεχνητή Νοημοσύνη» (Campagna et al., 2017).

Ο Turing στη συνέχεια εργάστηκε στο Εθνικό Εργαστήριο Φυσικής όπου δημιούργησε τα σχέδια για έναν από τους πρώτους υπολογιστές με αποθηκευμένα προγράμματα, τον Automatic Computing Engine (ACE). Ειδικότερα, το 1945, ο Turing σχεδίασε το επαναστατικό του σχέδιο για μια ηλεκτρονική υπολογιστική μηχανή - την αυτόματη μηχανή υπολογισμού του («ACE»). Ένα πιλοτικό μοντέλο του ACE έτρεξε το πρώτο του πρόγραμμα το 1950 και η έκδοση παραγωγής, η «DEUCE», έγινε ο ακρογωνιαίος λίθος της νεοσύστατης βρετανικής βιομηχανίας υπολογιστών. Ο πρώτος «προσωπικός» υπολογιστής βασίστηκε στην ανακάλυψη ACE του Turing. Τέλος, το 1948 ο Turing μεταβαίνει στο Πανεπιστήμιο του Manchester όπου εργάστηκε πάνω

στη δημιουργία και εξέλιξη ενός πλέον από τους γνωστότερους πρώτους ψηφιακούς προγραμματίσιμους ηλεκτρονικούς υπολογιστές τον Colossus Mark I σε συνεργασία με τον Thomas Flowers (Cooper & van Leeuwen, 2013).

### 1.2. Λογισμικό συστήματος και κατηγορίες λογισμικού συστήματος

Το λογισμικό μπορεί να διακριθεί σε τρεις μεγάλες κατηγορίες στο Λογισμικό Εφαρμογών (Application Software) στο Λογισμικό Συστήματος (System Software) και στις γλώσσες προγραμματισμού (Programming Languages)




**Εικόνα 2 Υπολογιστής και κατηγορίες λογισμικού**

Στο Λογισμικό Εφαρμογών εντάσσονται τα προγράμματα που ο χρήστης χρησιμοποιεί στον υπολογιστή κάθε φορά που θέλει να εκτελέσει μια συγκεκριμένη εργασία όπως η σύνταξη μιας επιστολής, η ακρόαση μουσικής ή η προβολή οποιουδήποτε βίντεο. Για όλες αυτές τις απαιτήσεις είναι απαραίτητο ένα ειδικό λογισμικό, για κάθε τύπο. Το λογισμικό αυτό που έχει σχεδιαστεί για τον συγκεκριμένο σκοπό είναι γνωστό ως λογισμικό εφαρμογής. Το λογισμικό εφαρμογής ανάλογα με την δραστηριότητα που ο αναπτύσσει ο χρήστης, επιλέγει και το αντίστοιχο πρόγραμμα. Κάποια παραδείγματα λογισμικού εφαρμογών είναι τα προγράμματα αυτοματισμού γραφείου όπως εφαρμογές επεξεργασίας κειμένου, φυλλομετρητές που

επιτρέπουν την περιήγηση στο διαδίκτυο και την χρήση δικτυακών εφαρμογών, εφαρμογές υπολογιστικού φύλλου, εφαρμογές διαχείρισης βάσεων δεδομένων, γλώσσες προγραμματισμού, εφαρμογές σχεδίασης, προγράμματα σχεδίασης, επεξεργασίας φωτογραφίας, βίντεο, εικόνων, ήχου και πολυμέσων, εγκυκλοπαίδειες & λεξικά, ηλεκτρονικά παιχνίδια (Campbell-Kelly & Aspray, 2011)

Η διαφορά μεταξύ λογισμικού συστήματος και λογισμικού εφαρμογής είναι η διαφορά στη διεπαφή χρήστη. Στο λογισμικό συστήματος, δεν υπάρχει διεπαφή χρήστη, ενώ στο λογισμικό εφαρμογών το περιβάλλον εργασίας χρήστη υπάρχει για κάθε λογισμικό, έτσι ώστε οι χρήστες να μπορούν εύκολα να χρησιμοποιούν το λογισμικό. Ο χρήστης δεν μπορεί να δει το λογισμικό συστήματος σαν ένα λειτουργικό σύστημα και δεν μπορεί να λειτουργήσει σε λογισμικό συστήματος, αλλά σε μια εφαρμογή, οι χρήστες λογισμικού μπορούν να δουν το λογισμικό εφαρμογής χρησιμοποιώντας μια γραφική διεπαφή χρήστη και μπορούν επίσης να εργαστούν στο λογισμικό της εφαρμογής. Ο χρήστης έχει επίσης την επιλογή να δημιουργήσει το δικό του λογισμικό και να χρησιμοποιήσει το λογισμικό για προσωπική του χρήση (Silverstone & Haddon, 2012).

Υπάρχουν τα πρότυπα τα οποία μπορούν να χρησιμοποιηθούν από τον χρήστη για τη δημιουργία προγραμμάτων που γράφονται από τον χρήστη. Το λογισμικό εφαρμογών μπορεί να ομαδοποιηθεί, και αυτό το πακέτο είναι γνωστό ως σουίτα εφαρμογών. Ένα παράδειγμα μιας σουίτας εφαρμογών είναι το Microsoft Office. Το λογισμικό επεξεργασίας κειμένου έχει σχεδιαστεί συνδυάζοντας διάφορα μικρά προγράμματα για να δημιουργηθεί ένα μόνο πρόγραμμα που μπορεί να χρησιμοποιηθεί για τη σύνταξη κειμένου, τη δημιουργία υπολογιστικού φύλλου ή τη δημιουργία παρουσιάσεων. Ο άλλος τύπος λογισμικού εφαρμογής είναι ο Mozilla Firefox, ο οποίος είναι εξερευνητής Διαδικτύου (internet explorer). Αυτά τα είδη λογισμικού εφαρμογών μπορούν να χρησιμοποιηθούν για την αναζήτηση οποιουδήποτε άρθρου, κειμένου στον Ιστό και για αλληλεπίδραση με τον έξω κόσμο (Kurose & Ross, 2015).

Αντίστοιχα, στο Λογισμικό Συστήματος εντάσσονται τα προγράμματα τα οποία είναι απαραίτητα για τον έλεγχο της λειτουργίας του υπολογιστή και τη δημιουργία και την εκτέλεση των προγραμμάτων εφαρμογών. Το λογισμικό

συστήματος είναι ειδικότερα, ένας τύπος λογισμικού υπολογιστών που έχει σχεδιαστεί για τη λειτουργία των τμημάτων υλικού του υπολογιστή και των προγραμμάτων εφαρμογών. Είναι η πλατφόρμα που παρέχει την βάση στο σύστημα υπολογιστών όπου μπορούν να εκτελεστούν άλλα προγράμματα υπολογιστών. Το λογισμικό του συστήματος λειτουργεί ως μεσαίο επίπεδο μεταξύ των εφαρμογών του χρήστη και του υλικού (Russell & Norvig, 2017).

Ο άλλος σκοπός του λογισμικού συστήματος είναι να μεταφράσει τις εισόδους που λαμβάνονται από άλλες πηγές και να τις μετατρέψει σε γλώσσα έτσι ώστε το μηχάνημα να τις αντιληφθεί. Το BIOS είναι για παράδειγμα ένας άλλος τύπος λογισμικού συστήματος που λειτουργεί όταν ξεκινά το σύστημα του υπολογιστή και χρησιμοποιείται για τη διαχείριση των δεδομένων μεταξύ των συσκευών υλικού (προσαρμογέας βίντεο, ποντίκι, πληκτρολόγιο και εκτυπωτής) και του λειτουργικού συστήματος. Το λογισμικό του συστήματος παρέχει τη λειτουργικότητα για τον χρήστη ώστε να μπορεί χρησιμοποιεί το υλικό απευθείας χρησιμοποιώντας το πρόγραμμα οδήγησης συσκευών. Η εκκίνηση (boot) είναι το πρόγραμμα λογισμικού συστήματος που φορτώνει το λειτουργικό σύστημα στην κύρια μνήμη του υπολογιστή ή μπορεί να φορτώσει σε μνήμη τυχαίας προσπέλασης (RAM) (Brass, 2008).

Το βασικότερο λογισμικό της κατηγορίας αυτής είναι το Λειτουργικό Σύστημα (Operating System, OS). Επίσης, στην κατηγορία αυτή εντάσσονται βοηθητικά προγράμματα (utilities) όπως προγράμματα ελέγχου και διαμόρφωσης του σκληρού δίσκου, ελέγχου και επιδιόρθωσης δυσλειτουργιών του υπολογιστή, ανάλυσης της κίνησης δεδομένων σε ένα δίκτυο υπολογιστών κ.ά., μεταφραστές (compilers) δηλαδή προγράμματα που διαβάζουν κώδικα γραμμένο σε μια γλώσσα προγραμματισμού (την πηγαία γλώσσα) και τον μεταφράζουν σε ισοδύναμο κώδικα σε μια άλλη γλώσσα προγραμματισμού (τη γλώσσα στόχο), διερμηνείς (interpreters) δηλαδή προγράμματα που εκτελούν ή ερμηνεύουν εντολές σε κάποια γλώσσα προγραμματισμού κλπ. (Russell & Norvig, 2017).

Η γλώσσα προγραμματισμού είναι η τρίτη κατηγορία λογισμικού υπολογιστών που χρησιμοποιείται από τους προγραμματιστές για να γράψουν τα προγράμματα, τα σενάρια και τις οδηγίες που μπορούν να εκτελεστούν από έναν υπολογιστή. Το άλλο όνομα της γλώσσας προγραμματισμού είναι μια γλώσσα

υπολογιστή που μπορεί να χρησιμοποιηθεί για τη δημιουργία ορισμένων κοινών προτύπων. Η γλώσσα προγραμματισμού μπορεί να θεωρηθεί ως η βάση που μπορεί να χρησιμοποιηθεί για την κατασκευή προγραμμάτων υπολογιστών και λειτουργικού συστήματος. Τα παραδείγματα γλωσσών προγραμματισμού είναι JAVA, C, C ++ και άλλες γλώσσες (Aho et al., 2013).

Υπάρχει πάντα κάποια ομοιότητα μεταξύ των γλωσσών προγραμματισμού, η μόνη διαφορά είναι η σύνταξη της γλώσσας προγραμματισμού που τις κάνει διαφορετικές. Ο προγραμματιστής χρησιμοποιεί τη σύνταξη και τους κανόνες της γλώσσας προγραμματισμού για τη σύνταξη των προγραμμάτων τους. Μόλις ο πηγαίος κώδικας γράφεται από έναν προγραμματιστή στο IDE (Ολοκληρωμένο Περιβάλλον Ανάπτυξης) καταρτίζει στη συνέχεια τον κώδικα σε γλώσσα μηχανής που μπορεί να γίνει κατανοητός από τον υπολογιστή. Η χρήση της γλώσσας προγραμματισμού γίνεται στην ανάπτυξη ιστότοπων, εφαρμογών και πολλών άλλων προγραμμάτων (Russell & Norvig, 2017).

Η γλώσσα προγραμματισμού μπορεί να χωριστεί σε δύο βασικά στοιχεία σύνταξης και σημασιολογίας. Η γλώσσα προγραμματισμού διέπεται από ακολουθίες λειτουργιών έτσι ώστε να επιτυγχάνεται το επιθυμητό αποτέλεσμα. Η γλώσσα προγραμματισμού είναι επίσης γνωστή ως γλώσσα υψηλού επιπέδου, καθώς τα προγράμματα που γράφει ένας προγραμματιστής είναι ευανάγνωστα και κατανοητά. Οι γλώσσες προγραμματισμού JAVA, C, C ++ θεωρούνται ως γλώσσες υψηλού επιπέδου. Η άλλη κατηγορία μιας γλώσσας προγραμματισμού είναι μια γλώσσα χαμηλού επιπέδου.

Το χαμηλό επίπεδο γλώσσας περιλαμβάνει τη γλώσσα μηχανής καθώς και τη γλώσσα συναρμολόγησης. Η γλώσσα συναρμολόγησης περιέχει μια λίστα με οδηγίες που δεν είναι ευανάγνωστες και κατανοητές. Η γλώσσα μηχανής περιέχει δυαδικούς κωδικούς που μπορούν να διαβαστούν απευθείας από την CPU και δεν είναι σε μορφή αναγνώσιμη από τον άνθρωπο. Το χαμηλό επίπεδο γλώσσας μπορεί να γίνει κατανοητό απευθείας από το υλικό του υπολογιστή (Knuth, 2014).

# ΚΕΦΑΛΑΙΟ 2 ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ (OPERATING SYSTEM)

## 2.1. Τι είναι το λειτουργικό σύστημα

Λειτουργικό σύστημα ορίζουμε το λογισμικό που εκτελείται σε κατάσταση πυρήνα και εκτελεί δύο λειτουργίες: παρέχει στους προγραμματιστές ένα σαφές σύνολο πόρων υλικού, για την χρήση των προγραμμάτων τους, και είναι υπεύθυνο για την σωστή διαχείριση αυτών των πόρων. Ποιο αναλυτικά το λειτουργικό σύστημα:

- Συντονίζει και ελέγχει τις διαδικασίες εισόδου, εξόδου και επεξεργασίας
- Διαχειρίζεται την ΚΜΕ (Κεντρική Μονάδα Επεξεργασίας) και την μνήμη του υπολογιστή
- Φροντίζει για τον ορθό διαμοιρασμό των πόρων (υλικό και λογισμικό) ανάλογα με την ζήτηση και την χρήση τους
- Ενεργοποιεί, απενεργοποιεί και διαχειρίζεται διάφορα προγράμματα (πχ τους μεταφραστές)
- Δημιουργεί ένα φιλικό περιβάλλον επικοινωνίας ανάμεσα στον χρήστη και τον Ηλεκτρονικό Υπολογιστή.

Οι σύγχρονοι υπολογιστές αποτελούνται από επεξεργαστές, μνήμες, χρονόμετρα (timers), δίσκους, διασυνδέσεις δικτύων και μία ευρεία ποικιλία από άλλες συσκευές. Βασική λειτουργία των σύγχρονων υπολογιστών είναι η ταυτόχρονη εκτέλεση πολλών προγραμμάτων. Το λειτουργικό σύστημα σχεδιάστηκε ώστε να παρέχει συστηματική και ελεγχόμενη κατανομή των πόρων που ζητείται από το εκάστοτε πρόγραμμα.


Ο συντονισμός των πόρων του συστήματος γίνεται με πολύπλεξη (multiplexing) είτε χρονική είτε χωρική. Όταν εκτελείται χρονική πολύπλεξη αυτό σημαίνει ότι διαφορετικά προγράμματα ή χρήστες χρησιμοποιούν τους πόρους εναλλάξ. Σε αυτήν την περίπτωση το λειτουργικό σύστημα είναι αρμόδιο ώστε να ορίζει τον χρόνο που κάθε πρόγραμμα θα χρησιμοποιεί τον εκάστοτε πόρο με τρόπο δίκαιο ως προς τα προς εκτέλεση προγράμματα. Όταν εκτελείται χωρική πολύπλεξη αυτό σημαίνει ότι κάθε πρόγραμμα παραλαμβάνει ένα μέρος του πόρου που ζητείται. Και εδώ το λειτουργικό σύστημα καλείται να αποφασίσει ποιο κομμάτι του πόρου θα δοθεί σε ποια εφαρμογή.

Τα λειτουργικά συστήματα χωρίζονται σε τρεις μεγάλες κατηγορίες: i) Λειτουργικά Συστήματα για Προσωπικούς Υπολογιστές, ii) Λειτουργικά Συστήματα για Μεγάλους Υπολογιστές, iii) Λειτουργικά Συστήματα για φορητές συσκευές. Τα βασικότερα λειτουργικά συστήματα ανά κατηγορία, τα οποία αξίζει να αναφέρουμε, είναι:

I. Λειτουργικά Συστήματα για Προσωπικούς υπολογιστές
   a. Windows
   b. Linux
   c. Mac OS X
II. Λειτουργικά Συστήματα για Μεγάλους Υπολογιστές
   a. UNIX
   b. Aix
   c. Solaris
III. Λειτουργικά Συστήματα για Φορητές Συσκευές
   b. iOS

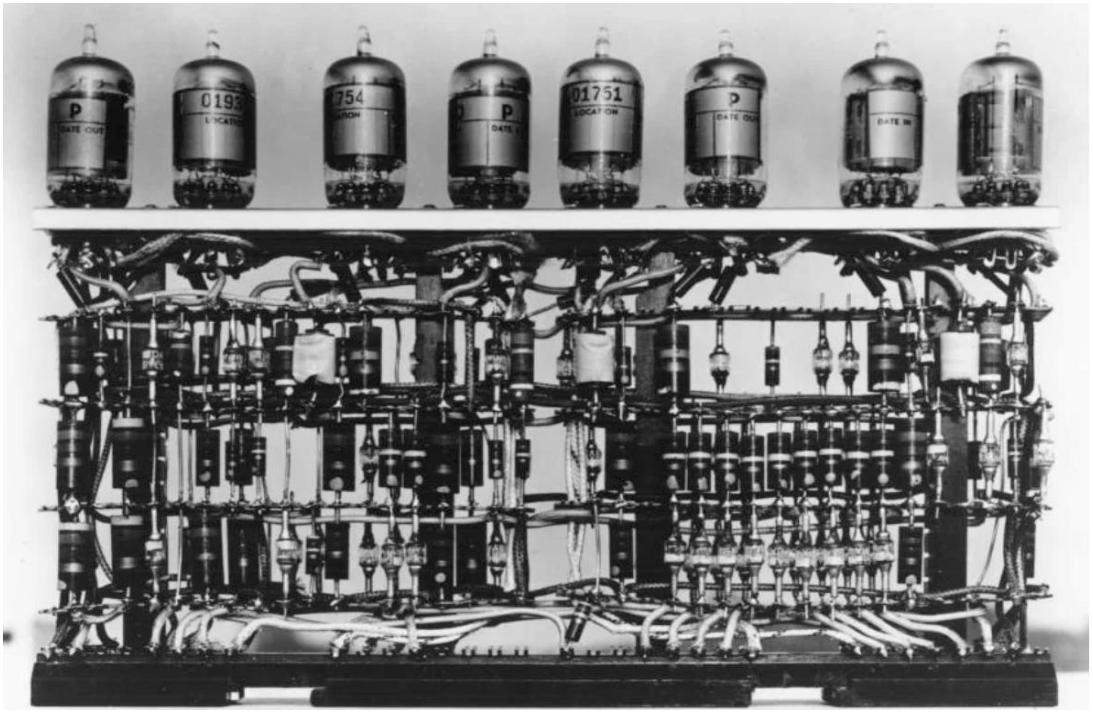
c. Windows Phone
d. Blackberry OS6
e. Firefox OS
f. Symbian

Στο επόμενο κεφάλαιο θα προχωρήσουμε σε μία σύντομη ιστορική αναδρομή επισημαίνοντας τα σημαντικότερα ορόσημα στην εξέλιξη των λειτουργικών συστημάτων τα οποία είναι αλληλένδετα με την εξέλιξη της αρχιτεκτονικής των υπολογιστών.

## 2.2. Ιστορική αναδρομή των λειτουργικών συστημάτων

### 2.2.1. Πρώτη γενιά (1945-1955): λυχνίες κενού

Μετά τις αποτυχημένες προσπάθειες του Babbage χρειάστηκε να φτάσουμε στην περίοδο του Β Παγκοσμίου πολέμου για να σημειώσουμε πρόοδο στις υπολογιστικές μηχανές. Ο καθηγητής John Atanassof και ο μεταπτυχιακός φοιτητής του Cliford Berry χρησιμοποίησαν 300 λυχνίες κενού και δημιούργησαν τον πρώτο ψηφιακό υπολογιστή στην ιστορία. Την ίδια περίοδο ο Konrad Zuse δημιούργησε, χρησιμοποιώντας μηχανικούς ηλεκτρονόμους (relays), τον υπολογιστή Z3. Την ίδια περίοδο κατασκευάστηκαν ο Κολοσσός (Colossus) από μία ομάδα στο Bletchley Park της Αγγλίας, ο Mark I από τον Howard Aiken στο Harvard και ο γνωστός σε όλους μας ENIAC από τον William Mauchley και τον μεταπτυχιακό σπουδαστή J. Presper Eckert στο πανεπιστήμιο της Πενσυλβάνιας.



**Εικόνα 3 Υπολογιστής με λυχνίες κενού**

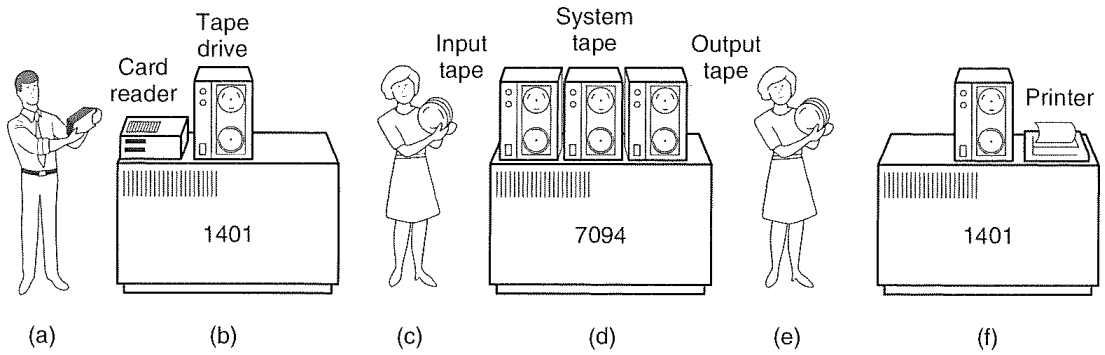
Όλοι οι παραπάνω υπολογιστές αν και διέπονται από διαφορετικά χαρακτηριστικά, έχουν ένα κοινό σημείο: όλοι χρειάζονταν δευτερόλεπτα για να εκτελέσουν ακόμη και τις πιο απλές αριθμητικές πράξεις. Όλος ο προγραμματισμός γινόταν με καλωδίωση των πινάκων συνδέσεων (plugboards), οι γλώσσες προγραμματισμού ήταν άγνωστες και τα λειτουργικά συστήματα υπήρχαν μόνο στην σφαίρα της φαντασίας.

### 2.2.2. Δεύτερη γενιά (1955-1965): τρανζίστορ και συστήματα δέσμης

Με την χρήση των τρανζίστορ έχουμε πλέον υπολογιστές αρκετά αξιόπιστους ώστε να προχωρήσει και η πώληση σε πελάτες. Οι υπολογιστές αυτοί που πλέον ονομάζονται μεγάλα υπολογιστικά συστήματα (main-frames) για να εκτελέσουν μία εργασία (job), ενός προγράμματος ή ενός συνόλου προγραμμάτων, ο προγραμματιστής αφού έγραφε το πρόγραμμά σε χαρτί, χρησιμοποιώντας Fortran ή συμβολική γλώσσα, το τρυπούσε σε κάρτες. Οι κάρτες αυτές μεταφέρονταν στην αίθουσα εισαγωγής ώστε να προχωρήσει η εκτέλεση του προγράμματος. Με το πέρας του προγράμματος τα

αποτελέσματα συλλέγονταν από τον εκτυπωτή και μεταφέρονταν στην αίθουσα αποτελεσμάτων. Η όλη διαδικασία ήταν αρκετά χρονοβόρα και σε συνδυασμό με το υψηλό κόστος των μηχανημάτων οδήγησαν σε εύρεση περαιτέρω λύσεων και στο γνωστό ως σύστημα δέσμης (batch system).

Βασική λειτουργία ήταν να συγκεντρωθούν οι εργασίες στην αίθουσα αναμονής και να γραφτούν σε μία μαγνητική ταινία με έναν χαμηλού κόστους υπολογιστή όπως ο IMB 1401. Στην συνέχεια, η ταινία μεταφερόταν σε άλλον υπολογιστή για την εκτέλεσή της, όπως ο IBM 7094. Τα αποτελέσματα επέστρεφαν στον IBM 1401 για εκτέλεση. Η όλη διαδικασία σηματοδοτούνταν από κάρτες ελέγχου. Οι κάρτες αυτές είναι οι πρόδρομοι των σύγχρονων κελυφών ( shells ) διερμηνευτών γραμμής ( command-line interpreters )




**Εικόνα 4 Ένα από τα πρώτα συστήματα δέσμης**

Οι μεγάλοι υπολογιστές δεύτερης γενιάς χρησιμοποιήθηκαν για επίλυση επιστημονικών προβλημάτων όπως η επίλυση των μερικών διαφορικών εξισώσεων που προκύπτουν αρκετά συχνά σε τεχνικές εφαρμογές. Προγραμματίζονταν κυρίως σε Fortran και σε συμβολική γλώσσα ( assembly ). Λειτουργικά συστήματα το είδους ήταν το FMS ( Fortran Monitor System ) και το IBSYS.

### 2.2.3. Τρίτη γενιά (1965-1980): ολοκληρωμένα κυκλώματα και πολυπρογραμματισμός

Το πρώτο σημαντικό βήμα έγινε από την IBM η οποία παρουσίασε την σειρά System/360. Η σειρά αποτελούνταν από μηχανήματα παρόμοια με το 1401 έως και μεγαλύτερα από το 7094. Βασικό χαρακτηριστικό της σειράς ήταν η παρόμοια αρχιτεκτονική και το ίδιο σύνολο εντολών με αποτέλεσμα τα προγράμματα που γράφονταν για το ένα μηχάνημα να μπορεί, θεωρητικά, να εκτελεστεί και στα υπόλοιπα. Η σειρά 360 ήταν η πρώτη σειρά που χρησιμοποίησε ολοκληρωμένα κυκλώματα ( Integrated Circuits – IC ). Μαζί με την εν λόγω οικογένεια υπολογιστών δημιουργήθηκε και το πρώτο λειτουργικό σύστημα OS/360. Ωστόσο μία σειρά από διαδοχικά μη επιλύσημα σφάλματα οδήγησαν σε περαιτέρω εξέλιξη και στην δημιουργία του πρώτου συστήματος χρονομερισμού το CTSS ( Compatible Time Sharing System ) το οποίο αναπτύχθηκε στο M.I.T σε έναν 7094.

Η επιτυχία του CTSS οδήγησε το M.I.T, τα Bell Labs, και την General Electric στην δημιουργία ενός συστήματος το οποίο θα μπορούσε να υποστηρίζει ταυτόχρονα πολλούς χρήστες. Το σύστημα αυτό ονομάστηκε MULTICS ( MULTiplexed Information and Computing Services ). Παρόλο που το MULTICS δεν σημείωσε εμπορική επιτυχία είχε μεγάλη επιρροή σε μεταγενέστερα λειτουργικά συστήματα όπως ήταν το Corbato ( 1972 ), Vyssotsky ( 1965 ), Daley and Dennis (1968), Organick ( 1972 ) και Saltzer ( 1974 ).

Μία σημαντική εξέλιξη κατά την Τρίτη γενιά ήταν η ανάπτυξη των μίνι υπολογιστών με αρχή τον DEC PDP-1 το 1961. Βασικό χαρακτηριστικό των εν λόγω υπολογιστών ήταν ότι είχαν μόνο 4K λέξεις με 18bit. Σε έναν PDP-7 μίνι υπολογιστή ο επιστήμονας των Bell Labs Ken Thomson έγραψε μία απλοποιημένη έκδοση του MULTICS που αργότερα εξελίχθηκε στο λειτουργικό σύστημα UNIX. Επειδή ωστόσο, ο πηγαίος κώδικας του UNIX ήταν ελεύθερος οδήγησε στην ανάπτυξη περαιτέρω εκδόσεων όπως το System V, AT & T και BSD. Για να είναι δυνατή η εκτέλεση προγραμμάτων σε κάθε μηχάνημα που εκτελεί μία κάποια έκδοση του UNIX η IEEE ανέπτυξε ένα πρότυπο το οποίο ονόμασε POSIX.

**Εικόνα 5 DEC PDP-1**

Αργότερα και συγκεκριμένα το 1987, ο συγγραφέας του UNIX ανέπτυξε μια απλοποιημένη έκδοση, το MINIX, για ακαδημαϊκούς σκοπούς. Το εν λόγω λειτουργικό σύστημα ήταν ίδιο με το UNIX και περιλάμβανε υποστήριξη POSIX. Η ανάγκη ωστόσο για μια ελεύθερή έκδοση του MINIX, αντί της επι πληρωμής που υπήρχε μέχρι εκείνη την στιγμή οδήγησε τον Φιλανδό Linus Torvalds να γράψει το γνωστό σε όλους μας LINUX. Όλα τα παραπάνω σχετικά με το UNIX θα τα αναφέρουμε αναλυτικά σε μετέπειτα κεφάλαια.

### 2.2.4. Τέταρτη γενιά ( 1980 - σήμερα)

Η τέταρτη γενιά των υπολογιστών, χαρακτηρίζεται από την χρήση κυκλωμάτων LSI ( Large Scale Integration – Ολοκλήρωση Μεγάλης Κλίμακας ). Αυτό οδήγησε στην δημιουργία πλέον προσωπικών υπολογιστών, παρόμοιων με τους PDP, και στην ραγδαία ανάπτυξη των λειτουργικών συστημάτων.

Το 1974 και για τις ανάγκες της INTEL ο Garry Kildall, ιδρυτής της Digital Research, έγραψε ένα λειτουργικό σύστημα βασισμένο σε δίσκο, το CP/M ( Control Program for Microcomputers – Πρόγραμμα Ελέγχου Μικροϋπολογιστών ).

Στην αρχές της δεκαετίας του 1980, ο Gates και για τις ανάγκες της συνεργασίας του με την IBM εξαγόραση από την εταιρεία Seattle Computer Products το λειτουργικό σύστημα DOS ( Disk Operating System – Λειτουργικό Σύστημα Δίσκου ) το οποίο ενσωμάτωσε με τον δικό του διερμηνέα, BASIC, δημιουργώντας το DOS/BASIC το οποίο αργότερα και κατόπιν τροποποιήσεων μετονομάστηκε σε MS-DOS ( Microsoft Disk Operating System). Η Microsoft πλέον όντας ο μεγαλύτερος κολοσσός στην σχεδίαση και παραγωγή λειτουργικών συστημάτων προχώρησε σε ανάπτυξη αρκετών εκδόσεων λειτουργικού συστήματος όπως τα Windows 95, Windows 98, Windows NT, Windows Me, Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8 και Windows 10.

Στον αντίποδα, ο άλλος σημαντικός ανταγωνιστής των Windows αποτελεί το UNIX. Το UNIX είναι ισχυρότερο σε σταθμούς εργασίας και τους διακομιστές δικτύων επιχειρήσεων, ωστόσο η παρουσία του επεκτείνεται και σε υπολογιστές γραφείου. Άλλο ένα παράγωγο του UNIX αποτελεί το FreeBSD προερχόμενο από το BSD πρόγραμμα Berkeley. Μια τροποποιημένη έκδοση του FreeBSD χρησιμοποιείται από τους υπολογιστές MACINTOSH. Επίσης, οι σταθμοί εργασίας με υψηλής απόδοσης chip RISC χρησιμοποιούν ως πρότυπο το UNIX.

# ΚΕΦΑΛΑΙΟ 3 ΚΑΤΗΓΟΡΙΕΣ ΛΕΙΤΟΥΡΓΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Κάθε κομμάτι λογισμικού στον υπολογιστή δημιουργείται κατασκευάζοντας πηγαίο κώδικα. Ο πηγαίος κώδικας ουσιαστικά αποτελεί το τεχνικό σχέδιο το οποίο λέει στο πρόγραμμα πως να λειτουργεί. Προτού οι δημιουργοί κυκλοφορήσουν το λογισμικό τους στην αγορά πρέπει να αποφασίσουν αν ο πηγαίος κώδικας του προϊόντος θα είναι ανοιχτός ( **open source** ) ή κλειστός ( **closed source** ). Ας δούμε τι σημαίνουν οι δύο αυτοί όροι και ποιες είναι οι σημαντικότερες διαφορές τους.

## 3.1. Λογισμικό Ανοιχτού Κώδικά ( OSS )

Το λογισμικό ανοιχτού κώδικα ( **OSS** ) διανέμεται βάσει συμφωνίας αδειοδότησης που επιτρέπει την κοινή χρήση, προβολή και τροποποίηση του πηγαίου κώδικα από άλλους χρήστες και οργανισμούς. Ή πιο απλά το λογισμικό ανοιχτού κώδικα είναι διαθέσιμο στο ευρύ κοινό για χρήση και τροποποίηση από τον αρχικό του σχεδιασμό, δωρεάν ή σε πολύ χαμηλό κόστος. Αυτό πρακτικά σημαίνει ότι ένα κομμάτι λογισμικού μπορεί να εξελιχθεί και να διορθωθεί από άλλους προγραμματιστές οπουδήποτε στον κόσμο. Στην ιδανική περίπτωση, αυτό σημαίνει ότι το λογισμικό βελτιώνεται με την πάροδο του χρόνου, αλλά συχνά μπορεί να πάρει πολλές ενδιαφέρουσες τροπές με όλη αυτή την εξέλιξη και μπορεί να αλλάξει εντελώς μορφή και σχήμα. Μερικά από τα σημαντικότερα λογισμικά ανοιχτού κώδικα είναι τα: Firefox, OpenOffice, Gimp, Alfresco, Android, Zimbra, Thunderbird, MySQL, Mailman, Moodle, TeX, Samba, Perl, PHP, KDE.

## 3.2. Λογισμικό Κλειστού Κώδικα ( CSS )

Το λογισμικό κλειστού κώδικα ( **CSS** ) μπορεί να οριστεί ως ιδιόκτητο λογισμικό που διανέμεται βάσει συμφωνίας αδειοδότησης σε εξουσιοδοτημένους χρήστες με ιδιωτικούς περιορισμούς τροποποίησης, αντιγραφής και αναδημοσίευσης. Ή πιο απλά, ο πηγαίος κώδικας δεν μοιράζεται με το κοινό για να μπορεί κάποιος να

τον δει ή να τον αλλάξει. Η τιμή του λογισμικού κλειστού κώδικα είναι υψηλή και οι χρήστες πρέπει να διαθέτουν έγκυρη και πιστοποιημένη άδεια χρήσης του λογισμικού. Όποιος εκδίδει μια εξουσιοδοτημένη άδεια, έτσι θέτει επίσης πολλούς περιορισμούς στους χρήστες με βάση τη χρηστικότητα και την τροποποίηση του λογισμικού. Η επιλογή κλειστού κώδικα είναι στην πραγματικότητα το είδος της ρύθμισης που θα περιμένατε από τις περισσότερες επιχειρήσεις, οι οποίες προστατεύουν το προϊόν τους και επιθυμούν να διατηρήσουν τον έλεγχο της μάρκας τους και της εμπειρίας χρήστη που προσφέρεται στους πελάτες τους. Μερικά από τα σημαντικότερα λογισμικά κλειστού κώδικα είναι τα: Skype, Google Earth, Java, Adobe Flash, Virtual Box, Adobe Reader, Microsoft Office, Microsoft Windows, WinRAR, Mac OS, Adobe Flash Player.

### 3.3. Διαφορές μεταξύ Λειτουργικών Κλειστού και Ανοιχτού Κώδικα

Για καλύτερη κατανόηση των ιδιοτήτων του λογισμικού ανοιχτού κώδικα και του λογισμικού κλειστού κώδικα, θα προχωρήσουμε σε μια σύγκριση πέντε βασικών πτυχών, και πιο συγκεκριμένα για κάθε κατηγορία θα μελετήσουμε: την τιμολογιακή πολιτική, την ασφάλεια, την ποιότητα υποστήριξης, την διαθεσιμότητα πηγαίου κώδικα και την χρηστικότητα.

- **Τιμολογιακή Πολιτική**:

  - Τα λογισμικά ανοιχτού κώδικα συχνά αναφέρονται ως λογισμικά χωρίς κόστος. Μπορεί ωστόσο να έχουν κόστος για πρόσθετες υπηρεσίες όπως βοήθεια ή κάποια πρόσθετη λειτουργικότητα. Έτσι ενδέχεται ενώ ο χρήστης έχει στην κατοχή του ένα δωρεάν λογισμικό ανοιχτό κώδικα να χρειαστεί εν τέλει να πληρώσει.

  - Τα λογισμικά κλειστού κώδικα ως επί το πλείστων είναι λογισμικά επί πληρωμή. Το κόστος διαφέρει ανάλογα με την πολυπλοκότητα του λογισμικού. Το τελικό προϊόν χαρακτηρίζεται από πλήρη υποστήριξη, λειτουργικότητα και καινοτομία. Πολλές εταιρείες προσφέρουν και

κάποιο δοκιμαστικό δωρεάν προϊόν προτού προχωρήσουν στην πώληση του προϊόντος.

- **Ασφάλεια**

  - Το ζήτημα της ασφάλειας αποτελεί ίσως το πιο αμφιλεγόμενο κομμάτι της εν λόγω σύγκρισης. Καθώς κάθε είδος έχει υπέρ και κατά. Στα λογισμικά ανοιχτού κώδικα, ο κώδικας μπορεί να προβληθεί, να κοινοποιηθεί και να τροποποιηθεί από την κοινότητα, πράγμα που σημαίνει ότι ο καθένας μπορεί να διορθώσει, να αναβαθμίσει και να δοκιμάσει τον κατεστραμμένο κώδικα. Τα σφάλματα διορθώνονται γρήγορα και ο κώδικας ελέγχεται διεξοδικά μετά από κάθε κυκλοφορία. Ωστόσο, λόγω της διαθεσιμότητας, ο πηγαίος κώδικας είναι ανοικτός για τους χάκερς ώστε να τον μελετήσουν και να εξασκηθούν πάνω του.

  - Αντίθετα, το λογισμικό κλειστού κώδικα μπορεί να διορθωθεί μόνο από τον προμηθευτή. Εάν κάτι πάει στραβά με το λογισμικό, ο χρήστης στέλνει ένα αίτημα και περιμένει την απάντηση από την ομάδα υποστήριξης. Η επίλυση του προβλήματος μπορεί να διαρκέσει πολύ περισσότερο από ό, τι σε σύγκριση με τα λογισμικά ανοιχτού κώδικα.

- **Ποιότητα Υποστήριξης**

  - Για το λογισμικό ανοιχτού κώδικα, οι μόνες επιλογές υποστήριξης είναι τα διάφορα φόρουμ, χρήσιμα άρθρα και σε κάποιες περιπτώσεις η χρήση επί πληρωμή κάποιου ειδικού. Γίνεται εύκολα αντιληπτό ότι η ανταπόκριση από τις παραπάνω επιλογές δεν θα βρίσκεται σε υψηλό επίπεδο.

  - Για το λογισμικό κλειστού κώδικα, η επιλογή είναι τα κανάλια επικοινωνίας ( τηλέφωνο, email, online chat ) με την τεχνική υποστήριξη του προϊόντος. Η απάντηση στο οποιοδήποτε ζήτημα έρχεται άμεσα και είναι καλά οργανωμένη και τεκμηριωμένη.

- **Διαθεσιμότητα Πηγαίου Κώδικα**

  - Το λογισμικό ανοιχτού κώδικα παρέχει τη δυνατότητα αλλαγής του πηγαίου κώδικα χωρίς περιορισμούς. Οι μεμονωμένοι χρήστες μπορούν να αναπτύξουν αυτό που θέλουν και να επωφεληθούν από την καινοτομία που αναπτύχθηκε από άλλους εντός της κοινότητας χρηστών. Καθώς ο πηγαίος κώδικας είναι εύκολα προσβάσιμος, επιτρέπει στους προγραμματιστές λογισμικού να βελτιώσουν τα ήδη υπάρχοντα προγράμματα.

  - Το λογισμικό κλειστού κώδικα είναι πιο περιορισμένο από το λογισμικό ανοιχτού κώδικα επειδή ο πηγαίος κώδικας δεν μπορεί να αλλάξει ή να προβληθεί. Ωστόσο, αυτός ο περιορισμός είναι και ο λόγος που τα εν λόγω λογισμικά προσφέρουν ασφάλεια και αξιοπιστία.

- **Ευχρηστία**

  - Όσο αφορά τα λογισμικά ανοιχτού κώδικα η ευχρηστία δεν είναι το δυνατό τους σημείο. Οι οδηγοί χρήστη γράφονται για τους προγραμματιστές/συγγραφείς και όχι για τους απλούς καθημερινούς χρήστες. Επίσης, τις περισσότερες φορές αυτοί οι οδηγοί δεν συμβαδίζουν με την δομή και τα πρότυπα του λογισμικού.

  - Για τη χρήση λογισμικού κλειστού κώδικα, η χρηστικότητα είναι ένα από τα πλεονεκτήματα τους. Οι οδηγοί χρήσης είναι συνήθως τεκμηριωμένοι, καλογραμμένοι και περιέχουν λεπτομερείς οδηγίες.

Σε επόμενο κεφάλαιο θα συγκρίνουμε ένα λειτουργικό κλειστού και ένα λειτουργικό ανοιχτού κώδικα ως προς την ασφάλειά τους.

# ΚΕΦΑΛΑΙΟ 4 ΑΣΦΑΛΕΙΑ ΛΕΙΤΟΥΡΓΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

## 4.1. Περιβάλλον Ασφάλειας

Τα λειτουργικά συστήματα πλέον αποτελούν την καρδιά κάθε συσκευής που χρησιμοποιούμε. Είτε πρόκειται για κάποιον προσωπικό υπολογιστή, είτε για μεγάλα συστήματα εταιρειών. Πολλοί χρήστες διατηρούν στους υπολογιστές τους προσωπικά στοιχεία, απόρρητα έγγραφα και χιλιάδες άλλα δεδομένα. Οι σημερινοί σκληροί δίσκοι είναι γεμάτοι φωτογραφίες, βίντεο και ταινίες. Η προστασία αυτών των πληροφοριών από μη εξουσιοδοτημένη χρήση είναι βασικό ζήτημα σε όλα τα λειτουργικά συστήματα.

Η ασφάλεια των λειτουργικών συστημάτων έχει αλλάξει ριζικά τις τελευταίες δεκαετίες. Μέχρι την αρχή της δεκαετίας του 1990, οι ηλεκτρονικοί υπολογιστές χρησιμοποιούνταν ως επί το πλείστων από εταιρείες, πανεπιστήμια και μεγάλους οργανισμούς. Η έννοια του δικτύου ακόμα δεν ήταν διαδεδομένη και έτσι κάθε μηχάνημα ήταν αυτόνομο. Η ασφάλεια του συστήματος επικεντρώθηκε στην μη εμπλοκή των χρηστών μεταξύ τους. Αναπτύχθηκαν πολύπλοκοι μηχανισμοί προκειμένου να επιτευχθεί αυτός ο στόχος. Οι μηχανισμοί αυτοί σε συνδυασμό με το λειτουργικό σύστημα ήταν κυρίως απλοί και έκαναν αυτήν την μία λειτουργία για την οποία δημιουργήθηκαν.

Η έλευση των προσωπικών υπολογιστών και του διαδικτύου άλλαξε εντελώς το σκηνικό. Πλέον ο κίνδυνος κάποιος άλλος χρήστης να κατασκοπεύσει αρχεία χρηστών σε έναν υπολογιστή εξαλείφθηκε. Προστέθηκαν ωστόσο άλλα σημαντικότερα προβλήματα ασφαλείας. Πλέον ιοί, σκουλήκια ( worms ) και άλλες ψηφιακές απειλές προσβάλλουν τα σημερινά συστήματα μέσω διαδικτύου. Βασικό πλεονέκτημα όλων αυτών των απειλών εκτός από το διαδίκτυο έπαιξε και ο υπερβολικά πλέον διογκωμένος, και γεμάτος σφάλματα, κώδικας των λειτουργικών συστημάτων.

Στην συνέχεια του εν λόγω κεφαλαίου, θα αναλύσουμε τους μηχανισμούς προστασίας και συνοπτικά τα διαθέσιμα μοντέλα που βοηθούν στην ασφάλεια ενός συστήματος. Θα χρειαστεί να αναλύσουμε την φύση των απειλών, την φύση των εισβολέων και την εκούσια απώλεια δεδομένων.

### 4.1.1. Απειλές Λειτουργικών Συστημάτων

Τα συστήματα υπολογιστών έχουν 4 βασικούς πυλώνες για τους οποίους υπάρχουν και ισάριθμες απειλές. Ο πρώτος πυλώνας αφορά την **εμπιστευτικότητα των δεδομένων** ( data confidentiality ). Με απλά λόγια η πρόσβαση σε δεδομένα τα οποία έχουν χαρακτηριστεί από τον διαχειριστή του συστήματος ως εμπιστευτικά θα πρέπει να γίνεται μόνο από εξουσιοδοτημένα άτομα. Ο διαχειριστής ορίζει τα προνόμια για κάθε χρήστη και το λειτουργικό σύστημα θα πρέπει να φροντίσει για την εφαρμογή του συγκεκριμένου κανόνα.

Ο δεύτερος πυλώνας αφορά την **ακεραιότητα των δεδομένων** ( data integrity ). Πρακτικά αυτό σημαίνει ότι η τροποποίηση, η αφαίρεση και η προσθήκη δεδομένων θα πρέπει να γίνεται μόνο από εξουσιοδοτημένους χρήστες καθώς και από τον διαχειριστή του συστήματος. Το σύστημα θα πρέπει να φροντίζει ώστε τα δεδομένα να διατηρούνται άθικτα.

Ο τρίτος πυλώνας έχει να κάνει με την **διαθεσιμότητα του συστήματος** ( system availability ). Είναι σημαντικό να μην μπορεί κανένας να διαταράσσει το σύστημα ώστε να το οδηγήσει σε **σημείο καμπής** ( bottleneck ) και να το αχρηστεύσει. Επιθέσεις αυτού του είδους ονομάζονται επιθέσεις **άρνησης εξυπηρέτησης** και πλέον είναι όλο και πιο συχνές. Για παράδειγμα, αν έχουμε έναν υπολογιστή ο οποίος είναι ταυτόχρονα και διακομιστής διαδικτύου, η αποστολή υπεράριθμών αιτήσεων σε αυτόν θα μπορούσε να τον αχρηστεύσει καθώς θα κατανάλωνε όλη την διαθέσιμη υπολογιστική ισχύ.

Τελευταίος πυλώνας ασφάλειας, και πιο πρόσφατος, έχει να κάνει με τις εξωτερικές απειλές του συστήματος μέσω του διαδικτύου. Εξωτερικοί εισβολείς μπορούν να εισχωρήσουν σε έναν οικιακό υπολογιστή, με χρήση ιών και άλλων αθέμητων μέσων, και είτε να προχωρήσουν σε αφαίρεση προσωπικών δεδομένων οποιασδήποτε μορφής είτε να μετατρέψουν τον υπολογιστή σε μηχάνημα **ζόμπι**. Συχνά τα ζόμπι χρησιμοποιούνται για την αποστολή μαζικής ανεπιθύμητης αλληλογραφίας ( spam ) ώστε να μην μπορεί να εντοπιστεί ο εγκέφαλος της επίθεσης.

### 4.1.2. Εισβολείς στα Λειτουργικά Συστήματα

Με βάση την ορολογία της ασφάλειας τα άτομα που είτε προσπαθούν είτε καταφέρνουν να αποκτήσουν πρόσβαση σε συστήματα ή κομμάτια του συστήματος για τα οποία δεν έχουν καμία δικαιοδοσία ονομάζονται **εισβολείς** ( intruders ) ή **εχθροί** ( adversaries ). Οι εισβολείς με βάση τον τρόπο δράσης τους χωρίζονται σε δύο μεγάλες κατηγορίες, τους παθητικούς και τους ενεργητικούς. Όταν ένας εισβολέας δρα παθητικά αυτό συνήθως σημαίνει ότι έχει αποκτήσει πρόσβαση και διαβάζει αρχεία για τα οποία δεν έχει τα κατάλληλα δικαιώματα. Οι ενεργητικοί εισβολείς από την άλλη προχωράνε σε τροποποιήσεις, οποιασδήποτε μορφής, των δεδομένων. Όταν σχεδιάζεται η ασφάλεια ενός λειτουργικού συστήματος πρέπει να ληφθεί υπόψιν και ποιο είδος εισβολέα θα αποτρέψει. Οι πιο σύνηθες κατηγορίες από την σημαντικότερη έως την λιγότερο σημαντική είναι:

i. Εμπορική ή στρατιωτική κατασκοπία. Ίσως η πιο σοβαρή εισβολή καθώς τις περισσότερες φορές αφορά προσπάθειες χρηματοδοτούμενες από ανταγωνιστές ή ξένα κράτη με σκοπό την απόσπαση χρήσιμων πληροφοριών. Η εν λόγω προσπάθεια εισβολής συχνά υποβοηθιέται με παγίδευση τηλεφωνικών συσκευών ή από τοποθέτηση κατευθυντικών κεραιών.

ii. Εισβολή με σκοπό το οικονομικό όφελος. Στόχοι σε αυτές τις περιπτώσεις είναι συνήθως μεγάλες τράπεζες. Κατά καιρούς έχουν χρησιμοποιηθεί ποικίλές μέθοδοι, όπως μεταφορές χρημάτων από λογαριασμούς που δεν έχουν χρησιμοποιηθεί για χρόνια, μπλοκάρισμα εγγραφών στην τράπεζα με σκοπό τον εκβιασμό και την απόσπαση χρηματικού ποσού, παρακράτηση λεπτών του ευρώ από τις τρέχουσες συναλλαγές και πολλές άλλες μέθοδοι.

iii. Εισβολή με σκοπό την ανάδειξη κενών στην ασφάλεια. Αυτό το είδος εισβολής συναντάται κυρίως σε φοιτητές ή προγραμματιστές σε εταιρείες λογισμικού οι οποίοι, κατέχοντας σημαντικές ικανότητες, αφιερώνουν μεγάλο μέρος του χρόνου τους με σκοπό να σπάσουν την ασφάλεια του τοπικού συστήματος υπολογιστών. Σκοπός είναι η ανάδειξη των ικανοτήτων και η επιδιόρθωση του κενού ασφαλείας το οποίο εντοπίστηκε.

iv. Εισβολή σε δεδομένα από χρήστες που τις περισσότερες φορές δεν έχουν τεχνικές γνώσεις. Πολλοί χρήστες διαθέτουν προσωπικούς υπολογιστές οι οποίοι συνδέονται σε ένα κοινόχρηστο διακομιστή αρχείων. Τα κοινόχρηστα αυτά αρχεία πολλές φορές πέφτουν θύμα αδιάκριτων χρηστών που θέλουν να τα διαβάσουν, χωρίς να είναι δικά τους, αν δεν υπάρχει κάποιος περιορισμός προς αυτούς. Πολλά πχ συστήματα UNIX επιτρέπουν εξ ορισμού την ανάγνωση νέων αρχείων από όλους.

v. Τελευταία κατηγορία, η οποία είναι και η πιο πρόσφατη, είναι η εισβολή από ιούς, μέσω διαδικτύου. Οι ιοί συνήθως είναι τμήματα κώδικα τα οποία κατά την εκτέλεση τους αναπαράγουν αντίγραφα του εαυτού τους κάνοντας κάποια ζημιά ή αλλοιώνοντας τον πηγαίο κώδικα του λειτουργικού συστήματος στο οποίο έχουν εισχωρήσει.

### 4.1.3. Εκούσια Απώλεια Δεδομένων

Απώλεια δεδομένων δεν προκύπτει μόνο από εισβολείς ή από κακόβουλα λογισμικά. Μπορεί να προκύψει και από κάποιό ατύχημα. Οι πιο συχνές απώλειες δεδομένων οφειλόμενες σε ατυχήματα, είναι:

1. Περιβαλλοντικές καταστροφές: φωτιές, πλημμύρες, σεισμοί, πόλεμοι ή ακόμα και ζημιά, κυρίως στα παλαιότερα συστήματα, από κάποιο τρωκτικό.

2. Σφάλματα στο υλικολογισμικό του μηχανήματος: δυσλειτουργίες της CPU, μη αναγνώσιμοι δίσκοι, σφάλματα που οφείλονται σε κακογραμμένα προγράμματα ή ακόμα και σφάλματα στις τηλεπικοινωνίες.

3. Ανθρώπινα λάθη: Εκτέλεση λάθος προγράμματος, εσφαλμένη καταχώρηση δεδομένων, απώλεια δίσκου ή κάτι παρόμοιο.

Η πλειοψηφία των παραπάνω προβλημάτων αντιμετωπίζεται εύκολα, διατηρώντας αντίγραφα ασφαλείας είτε τοπικά είτε πλέον σε κάποιον εικονικό υπολογιστή νέφους (

cloud computing ). Θα αναλύσουμε περαιτέρω βασικούς μηχανισμούς ασφαλείας και αντιμετώπισης, αποκλειστικά για το UNIX, σε επόμενο κεφάλαιο. Είναι ωστόσο σημαντικό να αναφερθούμε στο κομμάτι της κρυπτογραφίας καθώς σε αυτό βασίζονται οι περισσότεροι μηχανισμοί ασφαλείας.

## 4.2. Βασικές Αρχές Κρυπτογραφίας

Η κρυπτογραφία παίζει σημαντικό ρόλο στην ασφάλεια των λειτουργικών συστημάτων. Θα προχωρήσουμε σε μία σύντομη περιγραφή της έννοιας και των βασικών χαρακτηριστικών. Σκοπός της κρυπτογραφίας είναι να παραλάβει ένα μήνυμα ή αρχείο, το οποίο ονομάζεται **απλό κείμενο** ( plaintext ) και να το μετατρέψει σε **κρυπτοκείμενο** ( cipher-text ). Με αυτόν τον τρόπο η πρόσβαση στο εν λόγω μήνυμα, ή αρχείο, επιτρέπεται μόνο σε άτομα που μπορούν να εκτελέσουν την αντίστροφη διαδικασία. Για όλους τους υπόλοιπους το κρυπτοκείμενο είναι απλά ακατανόητα σύμβολα.

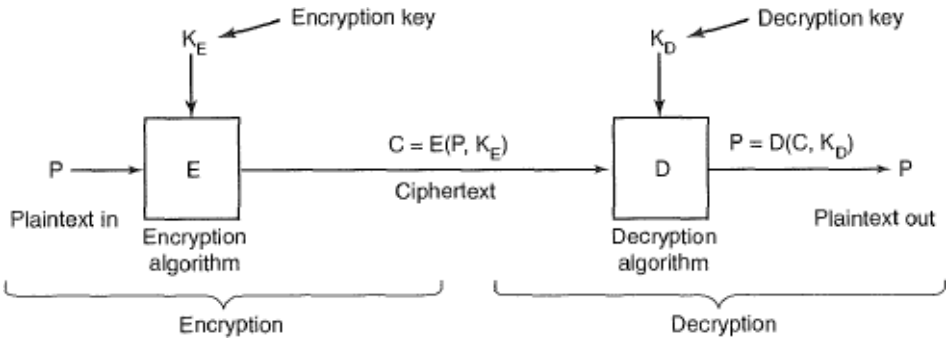
Οι αλγόριθμοι κρυπτογραφίας έχουν παραμέτρους που ονομάζονται **κλειδιά** ( keys ). Ο ορισμός της κρυπτογράφησης δηλώνει ότι το κρυπτοκείμενο παράγεται με την χρήση του αλγορίθμου κρυπτογράφησης E, με παραμέτρους το απλό κείμενο P και το μυστικό κλειδί κρυπτογράφησης $K_E$, δηλαδή προκύπτει η συνάρτηση κρυπτογράφησης:

$$C = E\left( P, K_E \right)$$

Παρόμοια αν D είναι ο αλγόριθμος αποκρυπτογράφησης και $K_D$ είναι το μυστικό κλειδί αποκρυπτογράφησης τότε προκύπτει η συνάρτηση αποκρυπτογράφησης:

$$P = D\left( C, K_D \right)$$

Ο Ολλανδός κρυπτογράφος Auguste Kerckoffs, τον 19ο αιώνα διατύπωσε την ιδέα ότι όλοι οι αλγόριθμοι κρυπτογράφησης και αποκρυπτογράφησης πρέπει να είναι δημόσιοι και η μυστικότητα να βρίσκεται στο κλειδί. Η ιδέα αυτή ονομάζεται **Αρχή του Kerckhoff** ( Kerckhoffs' Principle ).




**Εικόνα 6 Δημιουργία Κρυπτοκειμένου από απλό κείμενο**

### 4.2.1. Κρυπτογραφία Μυστικού και Δημοσίου κλειδιού

Πολλά συστήματα κρυπτογράφησης έχουν την ιδιότητα αν τους δοθεί το κλειδί κρυπτογράφησης μπορούν να αναπαράγουν το κλειδί αποκρυπτογράφησης, και αντίστροφα. Αυτή η διαδικασία ονομάζεται **κρυπτογραφία μυστικού κλειδιού** ( secret key cryptography ) ή **κρυπτογραφία συμμετρικού κλειδιού**. Τα εν λόγω συστήματα, είναι αποδοτικά επειδή η ποσότητα υπολογισμών που απαιτείται για να κρυπτογραφηθεί ή να αποκρυπτογραφηθεί ένα μήνυμα, είναι διαχειρίσιμη. Ωστόσο, έχουν ένα βασικό μειονέκτημά: πρέπει το κοινό μυστικό κλειδί να είναι γνωστό τόσο στον αποστολέα όσο και στον παραλήπτη.

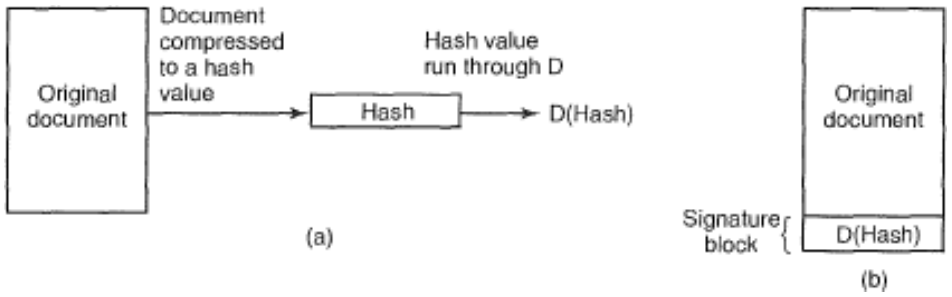
Το 1976, οι Diffie-Hellman πρότειναν την **κρυπτογραφία δημοσίου κλειδιο**ύ ( public key cryptography ). Στο σύστημα αυτό τα κλειδιά κρυπτογράφησης και αποκρυπτογράφησης είναι διαφορετικά. Έτσι αν δοθεί ένα καλά επιλεγμένο κλειδί κρυπτογράφησης, είναι αδύνατον να εντοπιστεί το κλειδί αποκρυπτογράφησης. Ουσιαστικά και οι δύο πλευρές που επικοινωνούν διαλέγουν ένα ζεύγος κλειδιών ( δημόσιο και ιδιωτικό ) και κοινοποιούν μόνο το δημόσιο κλειδί. Το δημόσιο κλειδί

είναι το κλειδί κρυπτογράφησης και το ιδιωτικό κλειδί είναι το κλειδί αποκρυπτογράφησης. Το μειονέκτημα της παραπάνω μεθόδου ωστόσο εντοπίζεται στην ταχύτητα, η κρυπτογραφία δημοσίου κλειδιού είναι χίλιες φορές πιο αργή από την συμμετρική κρυπτογραφία.

### 4.2.2. Ψηφιακές Υπογραφές

Πολλά έγγραφα όπως και το ηλεκτρονικό ταχυδρομείο για να μεταδοθούν θα πρέπει να κρυπτογραφηθούν. Η πιο διαδεδομένη μέθοδος είναι να περάσει το έγγραφο από έναν κρυπτογραφικό αλγόριθμο κατακερματισμού, ο οποίος συνήθως αντιστρέφεται πολύ δύσκολα. Οι δημοφιλέστερες συναρτήσεις κατακερματισμού ( hashing functions ) είναι οι **MD5** ( Message Digest 5 ) και η **SHA-1** ( Secure Hash Algorithm )

Επόμενο βήμα, είναι να εφαρμοστεί η κρυπτογραφία δημοσίου κλειδιού. Ο ιδιοκτήτης του εγγράφου χρησιμοποιεί το ιδιωτικό κλειδί το **μπλοκ υπογραφής D** ( $D_{Κατακερματισμού}$ ). Το μπλοκ υπογραφής προσαρμόζεται στο έγγραφο πριν την αποστολή.




**Εικόνα 7 a)Υπολογισμός του D, b) Τι λαμβάνει ο παραλήπτης**

Από την πλευρά του, ο παραλήπτης του εγγράφου πρέπει πρώτα να ελέγξει την ψηφιακή υπογραφή. Προκειμένου να το κάνει αυτό, γίνεται χρήση του δημοσίου κλειδιού του αποστολέα στο μπλοκ υπογραφής D. Πρακτικά γίνεται κρυπτογράφηση του αποκρυπτογραφημένου κατακερματισμού με σκοπό να παραχθεί πάλι ο κατακερματισμός. Αν το αποτέλεσμα δεν ταιριάζει με το μπλοκ υπογραφής, τότε το έγγραφο, το μπλοκ υπογραφής ή και τα δύο έχουν υποστεί κάποια αλλοίωση.

Όπως γίνεται εύκολα αντιληπτό από το παραπάνω, απαραίτητη προϋπόθεση για την χρήση της ψηφιακής υπογραφής είναι η κοινοποίηση του δημοσίου κλειδιού του αποστολέα. Προκειμένου να εξασφαλιστεί ότι πάντοτε ο παραλήπτης θα έχει το δημόσιο κλειδί του αποστολέα προσαρμόζεται από τον αποστολέα ένα **πιστοποιητικό** ( certificate ) το οποίο περιέχει το όνομα χρήστη και το δημόσιο κλειδί του αποστολέα και είναι ψηφιακά υπογεγραμμένο από κάποιον τρίτο ο οποίος έχει κοινοποιήσει στον παραλήπτη το δικό του δημόσιο κλειδί.

Η πιο έμπιστη πλατφόρμα που παράγει τέτοια πιστοποιητικά είναι η Αρχή Πιστοποίησης ή αλλιώς **CA** ( Certification Authority ). Για να μπορέσει ένας χρήστης να ελέγξει τα πιστοποιητικά της CA χρειάζεται το δημόσιο κλειδί της υπηρεσίας, το οποίο το λαμβάνει μέσω της **Υποδομής Δημοσίου Κλειδιού** ή αλλιώς **PKI** ( Public Key Infrastructure ).

Όλη η διαδικασία της κρυπτογράφησης βασίζεται στα κλειδιά. Είναι πολύ σημαντικό να υπάρχει ένας μηχανισμός όπου να γίνεται αποθήκευση των κλειδιών με ασφάλεια. Η επικρατέστερη πρόταση είναι να χρησιμοποιούνται **Υπομονάδες Έμπιστης Πλατφόρμας** ή αλλιώς **TPM** ( Trusted Platform Modules ). Πρόκειται για έναν μικροεπεξεργαστή με μνήμη, για την αποθήκευση των κλειδιών. Το TPM εκτελεί διαδικασίες κρυπτογράφησης και αποκρυπτογράφηση πολύ πιο γρήγορα. Η Microsoft όντας από τις πρώτες εταιρείες που πίστεψε στο TPM ανέπτυξε διάφορες τεχνολογίες βασιζόμενες σε αυτό, όπως οι Palladium, NGSCB, και BitLocker. Η Microsoft πιστεύει, ότι το TPM μπορεί να λειτουργήσει αποτρεπτικά ώστε να εμποδίζει την εκτέλεση μη εξουσιοδοτημένου λογισμικού.

# ΚΕΦΑΛΑΙΟ 5 ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ UNIX

Βασικός σκοπός της εν λόγω μελέτης είναι η ασφάλεια του Unix OS. Ωστόσο, προτού μελετήσουμε αυτό το κεφάλαιο θα πρέπει να δούμε τι σημαίνει Unix, ποιες είναι οι βασικές αρχές λειτουργίας του, ποια η δομή του, καθώς και να αναφερθούμε εν τάχει στην εξελικτική του πορεία.

## 5.1. Επισκόπηση

Το **Unix** ή **UNIX** είναι λειτουργικό σύστημα Ηλεκτρονικών Υπολογιστών, το οποίο αναπτύχθηκε κατά τις δεκαετίες του 1960 και του 1970 από ομάδα εργαζομένων των εργαστηρίων Μπελ (*Bell Labs*) της εταιρείας AT&T, στην οποία συμμετείχαν, μεταξύ άλλων, οι Κεν Τόμσον (*Ken Thompson*), Ντένις Ρίτσι (*Dennis Ritchie*) και Ντάγκλας Μακιλρόι (*Douglas McIlroy*).

Τα Unix OS κατασκευάστηκαν ώστε να χρησιμοποιούνται ευρέως τόσο σε σταθμούς εργασίας, προσωπικούς υπολογιστές όσο και σε εξυπηρετητές ( servers ). Άλλωστε το Unix OS σε συνδυασμό με το μοντέλο πελάτη – εξυπηρετητή ( Client – Server ) έπαιξε καθοριστικό ρόλο στην δημιουργία και στην εξελικτική πορεία των δικτύων αντί για ξεχωριστούς υπολογιστές. Αρχικά, το Unix OS διανεμήθηκε σε κρατικά και ακαδημαϊκά ιδρύματα με σκοπό να προσαρμοστούν σε έναν σημαντικά μεγάλο αριθμό υπολογιστών. Αυτός είναι και ο λόγος που το Unix έχει και την έννοια του ανοιχτού συστήματος ( Open Source ).

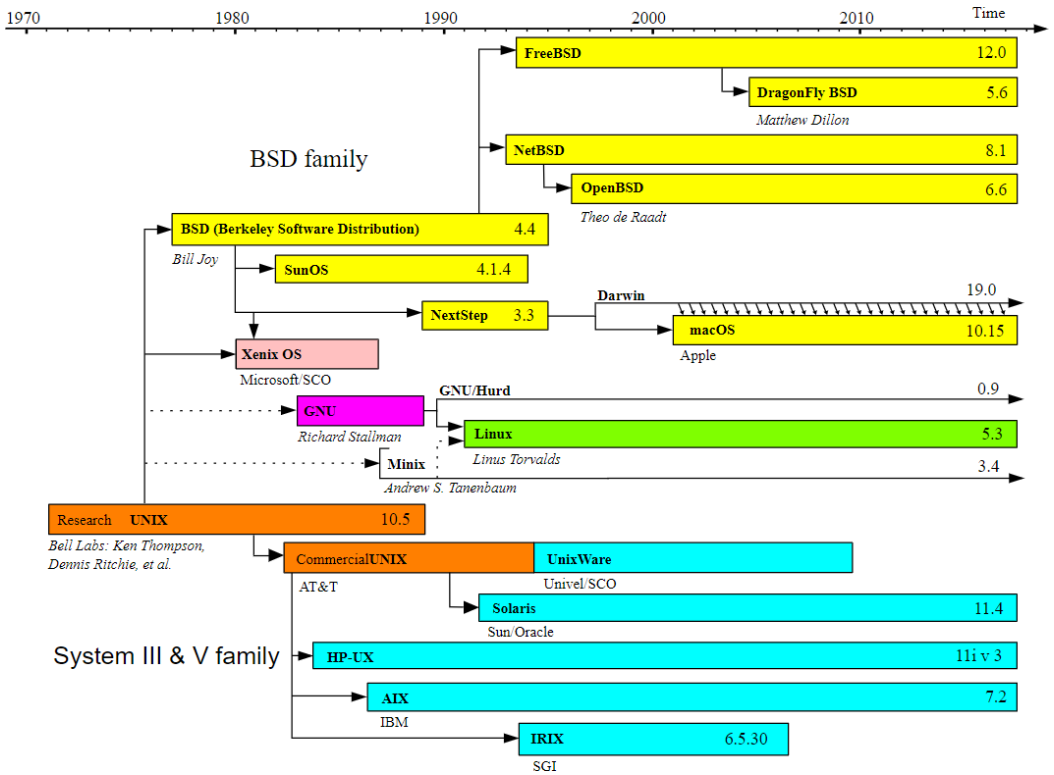
Όπως αναφέραμε και παραπάνω, 2.1.3 , το Unix σχεδιάστηκε ώστε να υποστηρίζει ταυτόχρονη χρήση από πολλούς χρήστες ( multiusers ) και πλήρη συμβατότητα σε πολλές πλατφόρμες. Τα κύρια χαρακτηριστικά του Unix τα οποία είναι γνωστά και ως η **φιλοσοφία Unix** είναι τα εξής:

- Αποθήκευση δεδομένων με χρήση απλού κειμένου
- Σύστημα αρχείων το οποίο διέπετέ από ιεραρχική δομή

- Αντιμετώπιση συσκευών και κάποιων μορφών διαδιεργασιακής επικοινωνίας ως αρχεία
- Χρήση μικρών εφαρμογών, οι οποίες συνδυάζονται με έναν διερμηνέα γραμμής εντολών με χρήση σωληνώσεων ( pipes )

Στο Unix OS, όλα τα παραπάνω εργαλεία συνεργάζονται με το κύριο πρόγραμμα ελέγχου, τον πυρήνα. Ο πυρήνας πέρα των βασικών λειτουργιών, εκκίνηση/τερματισμό προγραμμάτων – και λειτουργίες " χαμηλού επιπέδου ", φροντίζει για την σωστή διανομή του υλικού στα προγράμματα ώστε να αποφευχθούν συγκρούσεις λόγω ταυτόχρονης πρόσβασης στον ίδιο πόρο ή συσκευή. Για τον σκοπό αυτό ο πυρήνας έχει ειδικά δικαιώματα πάνω στο σύστημα ( χώρος πυρήνα ).

## 5.2. Unix – Ιστορική αναδρομή




**Εικόνα 8 UNIX Timeline**

### 5.2.1. Multics: Το πρωτότυπο του Unix

Το 1960 το MIT, η AT&T στα εργαστήρια Μπελ και η General Electric εργάζονταν πάνω σε ένα φορητό πειραματικό λειτουργικό σύστημα που λεγόταν Multics (Multiplexed Information and Computing Service), το οποίο είχε σχεδιαστεί ώστε να τρέχει στη σειρά υπολογιστών mainframe GE-645. Βασική αρχή λειτουργίας του, καθώς και καινοτομία της εποχής, ήταν η δυνατότητα απενεργοποίησης συσκευών ή πόρων του συστήματος χωρίς να επηρεάζονται οι χρήστες ή άλλα κομμάτια του συστήματος.

Το Multics, σχεδιάστηκε έχοντας ως γνώμονα την στρατιωτική ασφάλεια ώστε να είναι ταυτόχρονα ανθεκτικό τόσο σε εξωτερικές επιθέσεις όσο και σε επιθέσεις μεταξύ των χρηστών του συστήματος. Επομένως, λόγω του σχεδιασμού του, μυστικές, εμπιστευτικές και μη ταξινομημένες πληροφορίες θα μπορούσαν να συνυπάρχουν στον ίδιο υπολογιστή. Το σύστημα Multics σχεδιάστηκε για να αποτρέψει πληροφορίες που είχαν ταξινομηθεί σε ένα επίπεδο από το να βρεθούν στα χέρια κάποιου που δεν είχε πιστοποίηση να δει αυτές τις πληροφορίες. Τα Multics παρείχαν τελικά ένα επίπεδο ασφάλειας και υπηρεσίες που εξακολουθούν να είναι απαράμιλλες σε πολλά από τα σημερινά συστήματα υπολογιστών -συμπεριλαμβανομένου, ίσως, του Unix.

Ωστόσο, το 1969 το project του Multics δεν είχε προχωρήσει όπως περίμεναν όλοι έχοντας ως αποτέλεσμα η εταιρεία AT & T να αποχωρήσει από την συνεργασία.

Την ίδια χρονιά ο Ken Thomson, ένας από τους ερευνητές που εργαζόταν στο project Multics, χρησιμοποίησε έναν PDP-7 υπολογιστή και ανέπτυξε ένα δικό του λειτουργικό το οποίο ονομάστηκε Unix. Σε αντίθεση με το Multics που έκανε πολλά πράγματα ταυτόχρονα, το Unix προσπάθησε να κάνει ένα πράγμα καλά: να τρέχει προγράμματα. Η ασφάλεια συστήματος δεν ήταν κομμάτι του πρωταρχικού στόχου.

### 5.2.2. Η γέννηση του Unix

Καθώς οι επιστήμονες της AT & T πρόσθεσαν χαρακτηριστικά στο σύστημά τους καθ' όλη τη δεκαετία του 1970, το Unix εξελίχθηκε σε όνειρο κάθε προγραμματιστή. Το σύστημα βασίστηκε σε συμπαγή προγράμματα, τα οποία ονομάζονται εργαλεία, καθένα από τα οποία εκτελούσε μια μόνο λειτουργία. Συνδυάζοντας εργαλεία, οι προγραμματιστές θα μπορούσαν να κάνουν περίπλοκα πράγματα. Το Unix μιμήθηκε τον τρόπο που σκέφτονταν οι προγραμματιστές. Για να αποκτήσουν την πλήρη λειτουργικότητα του συστήματος, οι χρήστες χρειάζονταν πρόσβαση σε όλα αυτά τα εργαλεία - και σε πολλές περιπτώσεις, στον πηγαίο κώδικα για τα εργαλεία επίσης. Έτσι, όπως το σύστημα εξελίχθηκε, σχεδόν όλοι με πρόσβαση στις μηχανές βοήθησαν στη δημιουργία νέων εργαλείων και στον εντοπισμό σφαλμάτων των ήδη υπάρχοντών.

Το 1973, ο Thompson ξαναέγραψε το μεγαλύτερο μέρος του Unix στην πρόσφατα εφευρεμένη γλώσσα προγραμματισμού του Ritchie γνωστή σε όλους ως C. Η C σχεδιάστηκε για να είναι μια απλή, φορητή γλώσσα. Προγράμματα γραμμένα στη C θα μπορούσαν να μετακινηθούν εύκολα από το ένα είδος υπολογιστή στον άλλο, όπως συνέβαινε με προγράμματα γραμμένα σε άλλες γλώσσες υψηλού επιπέδου όπως η FORTRAN, ωστόσο έτρεχαν σχεδόν τόσο γρήγορα όσο τα προγράμματα που γράφονταν απευθείας στη μητρική γλώσσα του υπολογιστή.

Τουλάχιστον, αυτή ήταν η θεωρία. Στην πράξη, κάθε διαφορετικό είδος υπολογιστή στα Bell Labs είχε το δικό του λειτουργικό σύστημα. Τα προγράμματα σε C που γράφονταν στο PDP-11 μπορούσαν να εκτελεστούν στα υπόλοιπα μηχανήματα του εργαστηρίου, αλλά δεν λειτουργούσαν πάντα σωστά, γιατί κάθε λειτουργικό σύστημα εκτελούσε είσοδο και έξοδο με ελαφρώς διαφορετικούς τρόπους. Ο Mike Lesk ανέπτυξε μια «φορητή βιβλιοθήκη I/O» για να ξεπεράσει ορισμένες από τις ασυμβατότητες των συστημάτων, ωστόσο αρκετά έμειναν αξεπέραστα. Στη συνέχεια, το 1977, η ομάδα συνειδητοποίησε ότι ίσως ήταν πιο εύκολο να μεταφερθεί ολόκληρο το λειτουργικό σύστημα Unix αντί να προσπαθεί να μεταφέρει όλες τις βιβλιοθήκες.

Η πρώτη μεταφορά ολόκληρου του Unix ήταν στον Interdata 8/32 της AT & T, ένας μικροϋπολογιστής παρόμοιος με τον PDP-11. Το 1978, το λειτουργικό σύστημα

μεταφέρθηκε στον νέο μικροϋπολογιστή VAX της Digital. Το Unix εξακολουθούσε να είναι ένα πολύ πειραματικό λειτουργικό σύστημα. Παρ 'όλα αυτά, είχε γίνει δημοφιλές λειτουργικό σύστημα σε πολλά πανεπιστήμια και ήταν διαθέσιμο ήδη στο εμπόριο από πολλές εταιρείες.

Το 1984 η AT & T κυκλοφόρησε την πρώτη εμπορική έκδοση Unix, το System III. Καθώς δεν είχε την αποδοχή που περίμενε σύντομα προχώρησε σε πιο βελτιωμένες εκδόσεις, System IV και System V.

### 5.2.3. Berkeley Unix

Το πανεπιστήμιο της Καλιφόρνιας, στο Berkeley, ήταν από τα πρώτα που απέκτησαν την έκτη έκδοση του Unix. Έχοντας στην διάθεσή τους όλον τον πηγαίο κώδικα προχώρησε σε τροποποιήσεις του συστήματος. Αποτέλεσμα αυτού ήταν να κυκλοφορήσει σύντομα την έκδοση **1BSD** ( First Berkeley Software Distribution ) προοριζόμενη για τον PDP-11. Γρήγορα ακολούθησαν και άλλες εκδόσεις ( 2BSD, 3BSD ) με την σημαντικότερη να είναι η 4BSD για τον VAX.

Η εν λόγω έκδοση, εισήγαγε μία σειρά βελτιώσεων όπως η χρήση εικονικής μνήμης και σελιδοποίησης, η διαφορετική υλοποίηση του συστήματος αρχείων – κάνοντας το σύστημα ταχύτερο – καθώς επίσης και η πιο αξιόπιστη χρήση σημάτων. Σημαντικότερη όμως προσθήκη είναι η δικτύωση που παρουσιάστηκε για πρώτη φορά, με αποτέλεσμα να καθιερωθεί το πρωτόκολλο δικτύου ( **TCP/IP** ) στον κόσμο του Unix και αργότερα σε όλο το διαδίκτυο όπου έχουμε διακομιστές με Unix OS.

Το Berkeley πρόσθεσε επίσης αρκετά βοηθητικά προγράμματα όπως ένας διορθωτής κειμένου ( vi ), ένα νέο κέλυφος ( csh ), μεταγλωττιστές Pascal και Lisp και πολλά άλλα. Συνέπεια όλων των παραπάνω είναι το Berkeley Unix να καθιερωθεί σε όλων τον ακαδημαϊκό και αμυντικό χώρο.

### 5.2.4. Πρότυπο UNIX

Στα τέλη της δεκαετίας του 1980, κυκλοφορούσαν δύο κύριες εκδόσεις του UNIX: η 4.3BSD και η System V Release 3. Οι διαφορετικές αυτές εκδόσεις σε συνδυασμό με την δυνατότητα των κατασκευαστών να προσθέτουν τις δικές τους βελτιώσεις καθιστούσε αδύνατη την συγγραφή και διανομή λογισμικών για το Unix καθώς εμφανίζονταν ασυμβατότητες στις διαφορετικές εκδόσεις. Γρήγορα γεννήθηκε η ανάγκη για προτυποποίηση του Unix.

Η πρώτη προσπάθεια έγινε από την IEEE όπου και γεννήθηκε το πρότυπο POSIX ( Portable Operating System ). Η κατάληξη IX προστέθηκε για να φέρει το όνομα UNIX. Μετά από πολλές προσπάθειες παρήχθη το γνωστό πλέον πρότυπό **1003.1**. Το πρότυπο αυτό ορίζει ένα σύνολο από διαδικασίες βιβλιοθήκης τις οποίες θα πρέπει πλέον να παρέχει κάθε έκδοση UNIX. Το πρότυπο 1003.1 γράφτηκε με τέτοιο τρόπο ώστε να είναι κατανοητό τόσο από αυτούς που ασχολούνται με την υλοποίηση λειτουργικών συστημάτων όσο και από τους συγγραφείς λογισμικού.

### 5.2.5. MINIX

Το βασικότερο μειονέκτημα των μέχρι τώρα εκδόσεων ήταν η πολυπλοκότητα και το μέγεθος τους, κάτι εντελώς αντίθετο με την αρχική φιλοσοφία του UNIX. Επίσης, ο πηγαίος κώδικας δεν ήταν διαθέσιμος και όποτε ήταν διαθέσιμος δεν μπορούσε να γίνει κατανοητός. Αποτέλεσμα αυτού ήταν η συγγραφή μίας αρκετά μικρότερης και πλήρως κατανοητής έκδοσης UNIX, από τον καθηγητή Andrew Tanenbaum, η οποία ονομάστηκε MINIX.

Το MINIX, βασίστηκε στην σχεδίαση μικροπυρήνα ( microkernel ). Σε σύγκριση με τα μονολιθικά συστήματα οι μικροπυρήνες κατανοούνται και συντηρούνται εύκολα καθώς έχουν αρθρωτή δομή. Το MINIX σύντομα έγινε αντικείμενο λατρείας και καθιερώθηκε ως πρότυπο για όλες τις μετέπειτα εκδόσεις του UNIX.

### 5.2.6. LINUX

Γρήγορα εμφανίστηκε η ανάγκη για ένα πλήρες σύστημα με λειτουργίες που δεν υπήρχαν στο MINIX. Έτσι Το 1991, ένας Φινλανδός φοιτητής πληροφορικής με το όνομα Linus Torvalds αποφάσισε να δημιουργήσει μια δωρεάν έκδοση του λειτουργικού συστήματος Unix που θα ήταν καταλληλότερη για καθημερινή χρήση. Ξεκινώντας με το σύνολο κωδικών Minix, ο Torvalds επαναπροσδιορίζει μόνο τον πυρήνα και το σύστημα αρχείων κομμάτι-κομμάτι έως ότου είχε ένα νέο σύστημα που δεν είχε κανένα πρωτότυπο κώδικα του Tanenbaum μέσα σε αυτό. Ο Torvalds ονόμασε το σύστημα που προέκυψε "Linux". Συνδυάζοντας το σύστημά του με άλλα ελεύθερα διαθέσιμα εργαλεία, κυρίως τον μεταγλωττιστή C και τον επεξεργαστή που αναπτύχθηκαν από το Free Software Foundation’s GNU project και τον X Consortium’s window server, ο Torvalds μπόρεσε να δημιουργήσει ένα ολόκληρο λειτουργικό σύστημα. Οι εργασίες στο Linux συνεχίζονται μέχρι σήμερα από πλήθος συντελεστών.

## 5.3. Ασφάλεια και UNIX

Το UNIX εν τη γενέσει του δεν σχεδιάστηκε για να είναι ασφαλές. Σχεδιάστηκε με τα απαραίτητα χαρακτηριστικά για να κάνει την υπηρεσία ασφαλή. Το Unix χαρακτηρίζεται ως πολυχρηστικό ( multi-user ) σύστημα και ως σύστημα πολλαπλών εργασιών ( multi-tasking ). Πολυχρηστικό σημαίνει ότι το λειτουργικό σύστημα επιτρέπει σε πολλούς χρήστες να χρησιμοποιούν ταυτόχρονα τον ίδιο υπολογιστή. Σύστημα πολλαπλών εργασιών, σημαίνει ότι ο κάθε χρήστης μπορεί να εκτελέσει ταυτόχρονα πάρα πολλά προγράμματα.

Μία φυσική λειτουργία ενός τέτοιου λειτουργικού συστήματος είναι να αποτρέπει διαφορετικούς χρήστες ή προγράμματα που χρησιμοποιούν τον ίδιο υπολογιστή να παρεμβάλουν μεταξύ τους. Χωρίς να υπάρχει αυτή η ασφάλεια τότε θα μπορούσε π.χ., ένα πρόγραμμα γραμμένο από έναν φοιτητή, με λάθη στον κώδικα, να επηρεάσει άλλα προγράμματα, να διαγράψει αρχεία ή ακόμα και να κρασάρει ολόκληρο το σύστημα. Για την αποτροπή λοιπόν μίας τέτοιας περίπτωσης, μία μορφή ασφάλειας πάντοτε υπήρχε στην φιλοσοφία του Unix.

Αλλά η ασφάλεια του UNIX παρέχει κάτι περισσότερο από μία απλή προστασία μνήμης. Το UNIX έχει εξελιγμένη ασφάλεια έτσι ώστε το λειτουργικό σύστημα να ελέγχει τους τρόπους πρόσβασης των χρηστών στα αρχεία, να τροποποιεί τις βάσεις δεδομένων του συστήματος και να χρησιμοποιεί τους πόρους του συστήματος κατά το δοκούν.

Στα επόμενα κεφάλαια θα αναλύσουμε τους βασικούς μηχανισμούς ασφαλείας του UNIX.

### 5.3.1. Λογαριασμοί Χρηστών

Κάθε σύστημα UNIX περιλαμβάνει έναν λογαριασμό διαχειριστή ( root account ) , που είναι ο μόνος λογαριασμός που μπορεί να εκτελεί άμεσα καθήκοντα διαχειριστή. Όλοι οι άλλοι λογαριασμοί στο σύστημα είναι “φτωχοί”. Αυτό σημαίνει

ότι αυτοί οι λογαριασμοί δεν έχουν δικαιώματα πέρα από την πρόσβαση σε αρχεία που έχουν επισημανθεί με το κατάλληλο δικαιώματα, καθώς και τη δυνατότητα εκκίνησης υπηρεσιών δικτύου.

Κάθε χρήστης πρέπει να έχει έναν μοναδικό λογαριασμό στο σύστημα. Επίσης οι υπηρεσίες δικτύου μπορεί επίσης να έχουν τους δικούς τους ξεχωριστούς λογαριασμούς, προκειμένου να έχουν πρόσβαση στα αρχεία του συστήματος που απαιτούνται για την λειτουργία τους. Εγκεκριμένα βοηθητικά προγράμματα επιτρέπουν στους χρήστες να αποκτήσουν προσωρινά δικαιώματα διαχειριστή όταν είναι απαραίτητο, έτσι ώστε οι διαχειριστές να μπορούν να διαχειρίζονται το σύστημα με τους δικούς τους λογαριασμούς χρηστών στην περίπτωση που αυτό απαιτηθεί.

Για λόγους ευκολίας, οι λογαριασμοί μπορεί να είναι μέλη ενός ή περισσοτέρων ομάδων. Αυτό είναι πρακτικό, καθώς αν δοθεί για παράδειγμα πρόσβαση σε έναν πόρο για μία συγκεκριμένη ομάδα χρηστών, αυτό πρακτικά σημαίνει πως όλοι οι χρήστες της ομάδας έχουν την δυνατότητα να αποκτήσουν πρόσβαση στον εν λόγω πόρο.

Τα συστήματα UNIX χρησιμοποιούν μηχανισμούς έλεγχος ταυτότητας με δυνατότητα σύνδεσης ( Pluggable Authentication Modules ), έτσι ώστε να διευκολύνεται η διαχείριση της πρόσβασης από τους χρήστες. Για κάθε προσπάθεια εισόδου ή αλλαγή κωδικού πρόσβασης, η σχετική υπηρεσία καλεί τους μηχανισμούς PAM με τη σειρά. Ορισμένες μονάδες υποστηρίζουν πηγές ελέγχου ταυτότητας, όπως τοπικά αποθηκευμένα αρχεία δεδομένων ή χρήση εξωτερικών υπηρεσιών καταλόγου ( LDAP ).

### 5.3.2. Δικαιώματα αρχείων

Κάθε αρχείο και κατάλογος στο UNIX χαρακτηρίζεται από τρία σύνολα δικαιωμάτων τα οποία καθορίζουν πως και από ποιον θα είναι προσπελάσιμα:

- Δικαιώματα για τον ιδιοκτήτη. Δηλαδή για τον συγκεκριμένο λογαριασμό ο οποίος είναι υπεύθυνος για το συγκεκριμένο αρχείο ή κατάλογο.
- Δικαιώματα για τα γκρουπ χρηστών που χρησιμοποιούν το αρχείο ή τον κατάλογο.

- Δικαιώματα για όλους τους άλλους λογαριασμούς.

Κάθε σύνολο δικαιωμάτων μπορεί να έχει ή να μην έχει κάποια από τα παρακάτω δικαιώματα:

- Ανάγνωση ( read )
- Εγγραφή ( write )
- Εκτέλεση ( execute )

Ένας χρήστης μπορεί να εκτελέσει ένα πρόγραμμα αρχείου μόνο αν ανήκει σε ένα σύνολο που χαρακτηρίζεται από την άδεια εκτέλεσης. Για καταλόγους, η άδεια εκτέλεσης υποδεικνύει ότι οι χρήστες στο σχετικό σύνολο ενδέχεται να δουν τα αρχεία μέσα σε αυτόν τον κατάλογο, αν και δεν μπορούν πραγματικά να διαβάσουν, να γράψουν ή να εκτελέσουν οποιοδήποτε αρχείο εκτός εάν τους το επιτρέπουν τα δικαιώματα χρήστη που κατέχουν. Τα εν λόγω εκτελέσιμα αρχεία εκτελούνται αυτόματα με τα προνόμια του κατόχου του λογαριασμού και όχι με τα προνόμια του λογαριασμού που τα ενεργοποιεί.

Η πλειοψηφία των αρχείων σε ένα σύστημα UNIX ανήκει στον λογαριασμό διαχειριστή και έχουν δικαιώματα που περιορίζουν ή αποκλείουν την πρόσβαση από όλους τους άλλους λογαριασμούς. Αντίθετα, ο λογαριασμός διαχειριστή έχει πλήρη πρόσβαση σε όλα τα αρχεία του συστήματος ανεξαρτήτως των δικαιωμάτων κάθε αρχείου.

### 5.3.3. Κρυπτογράφηση χώρου αποθήκευσης

Τα περισσότερα UNIX συστήματα υποστηρίζουν κρυπτογράφηση για τον χώρο αποθήκευσης. Πρακτικά δημιουργούνται από το σύστημα ένα ή περισσότερα τμήματα του δίσκου τα οποία κρυπτογραφούνται. Κάθε τμήμα είναι ένα αρχείο το οποίο περιέχει άλλα αρχεία και καταλόγους. Για να μπορέσει κάποιος χρήστης να λάβει πρόσβαση στο εν λόγω τμήμα του δίσκου θα πρέπει να καταχωρήσει το κατάλληλο κλειδί

ασφαλείας. Τα περιεχόμενα του κρυπτογραφημένου τμήματος δεν μπορούν να χρησιμοποιηθούν αν δεν υπάρχει πρόσβαση σε αυτόν. Αξίζει ωστόσο να σημειωθεί ότι η κρυπτογράφηση ολόκληρου του δίσκου ή τμήματος του οδηγεί σε μείωση της απόδοσης του συστήματος, γεγονός μη αποδεκτό σε συστήματα που εκτελούνται απαιτητικές εφαρμογές.

### 5.3.4. Ασφαλής απομακρυσμένη πρόσβαση

Κάθε UNIX σύστημα είναι εξοπλισμένο με μία έκδοση του προτύπου openSSH. Μία SSH υπηρεσία χρησιμοποιεί, εξ ορισμού, ισχυρή κρυπτογράφηση και προσφέρει τις εξής λειτουργίες/εργαλεία:

- Απομακρυσμένη πρόσβαση σε γραμμή εντολών
- Απομακρυσμένη εκτέλεση εντολών
- Απομακρυσμένη πρόσβαση σε γραφικό περιβάλλον
- Μεταφορά αρχείων

Επιπλέον, οι δυνατότητες του SSH προτύπου επιτρέπουν στον χρήστη να δημιουργήσει μία σήραγγα συνδέσεων με άλλες υπηρεσίες. Μια υπηρεσία η οποία χρησιμοποιεί σύνδεση μέσω σήραγγας SSH επωφελείται από τα ίδια χαρακτηριστικά ασφαλείας και συμπίεσης δεδομένων με αυτά του SSH προτύπου. Αυτό επιτρέπει την προστασία όλων των επικοινωνιών μεταξύ UNIX συστημάτων, ακόμη και όταν η κυκλοφορία περνάει ανοιχτά ασύρματα δίκτυα ή το δημόσιο Διαδίκτυο.

Το λογισμικό SSH δεν κρυπτογραφεί μόνο τη σύνδεση μεταξύ συστημάτων, αλλά επίσης χρησιμοποιεί ένα σύστημα κλειδιών ασφαλείας το οποίο προσφέρει αμοιβαία πιστοποίηση μεταξύ των συστημάτων. Κάθε βοηθητικό πρόγραμμα - πελάτης SSH ελέγχει αυτόματα την ταυτότητα οποιουδήποτε απομακρυσμένου συστήματος στο οποίο συνδέεται, επαληθεύοντας το κλειδί ασφαλείας. Ομοίως, οι χρήστες ενδέχεται να αποκτήσουν πρόσβαση στα συστήματα καταχωρώντας το κλειδί ασφαλείας, αντί να πληκτρολογούν δυνητικά σπάσιμο κωδικούς πρόσβασης.

### 5.3.5. Διαχείριση Λογισμικού

Η πλειοψηφία των διανομών UNIX ενσωματώνει λειτουργίες διαχείρισης λογισμικού που βασίζονται σε πακέτα αρχείων και σετ από ειδικές προσχεδιασμένες ιστοσελίδες, γνωστές ως αποθετήρια ( repositories ) ή κανάλια ( channels ). Τα εργαλεία διαχείρισης πακέτων κατασκευάζουν ή ενημερώνουν αντίγραφα λογισμικού από αυτά πακέτα και εκτελούν τυχόν άλλες εργασίες που απαιτούν τα εν λόγω πακέτα. Τα αποθετήρια επιτρέπουν στα εργαλεία διαχείρισης να παρέχουν αυτόματα τη σωστή έκδοση κάθε προϊόντος λογισμικού, με τη λήψη των απαιτούμενων πακέτων απευθείας από το σχετικό αποθετήριο. Τα περισσότερα συστήματα χρησιμοποιούν επίσης αθροίσματα ελέγχου και δοκιμές ψηφιακής υπογραφής για να διασφαλιστεί ότι αυτά τα πακέτα είναι αυθεντικά και δεν περιέχουν σφάλματα.

Επιπλέον, τα εργαλεία διαχείρισης πακέτων μπορούν να εντοπίσουν ξεπερασμένες εκδόσεις του λογισμικό ελέγχοντας τα πακέτα λογισμικού που είναι εγκατεστημένα σε ένα σύστημα και συγκρίνοντάς τα με τα πακέτα στα αποθετήρια. Εκτελώντας απλώς την ρουτίνα ενημέρωσης το λειτουργικό μπορεί να είναι βέβαιο ότι δεν θα υπάρχουν κενά ασφαλείας στο σύστημα.

### 5.3.6. Δοκιμή ακεραιότητας κεντρικού υπολογιστή

Για να επαληθεύσει ότι το σύστημα δεν έχει παραβιαστεί ή παραποιηθεί, το UNIX, χρησιμοποιεί ρουτίνες ελέγχου ακεραιότητας. Οι ρουτίνες αυτές αφού λάβουν ένα πλήρες αντίγραφο του συστήματος, το συγκρίνουν με ένα άθροισμα ελέγχου ( chechsum ) από προηγούμενο έλεγχο. Διανομές UNIX, όπως οι Solaris και FreeBSD, περιλαμβάνουν βοηθητικά προγράμματα ελέγχου ακεραιότητας συστήματος για αυτόν τον σκοπό. Δεδομένου ότι οι διαμορφώσεις του συστήματος ποικίλλουν, οι διαχειριστές πρέπει να προβούν στις κατάλληλες ρυθμίσεις του προγράμματος ελέγχου ακεραιότητας, ώστε να αποκλειστούν οι ειδικοί κατάλογοι και τα αρχεία, τα οποία εξ ορισμού αλλάζουν σε ένα σύστημα, πριν δημιουργήσουν ένα αρχικό άθροισμα ελέγχου

βάση δεδομένων για αυτό το σύστημα. Ο έλεγχος ακεραιότητας μπορεί στη συνέχεια να συγκρίνει το αθροίσματα ελέγχου για κάθε αρχείο συγκρινόμενα με αυτά της βάσης δεδομένων και να αναφέρουν οποιαδήποτε αλλοίωση.

### 5.3.7. Επαναφορά συστήματος

Το Unix παρέχει την δυνατότητα να γίνει ολική επαναφορά των αρχείων λογισμικού που συμπεριλαμβάνεται στην διανομή του λειτουργικού. Προκειμένου να γίνει πλήρης επαναφορά ενός συστήματος που επηρεάστηκε από κάποιο ατύχημα ή υπέστη κάποια παραβίαση ασφαλείας θα πρέπει η ρουτίνα επαναφοράς να έχει πρόσβαση στα αντίγραφα των αρχείων, στις ρυθμίσεις και στα αρχεία συστήματος. Τα αρχεία αυτά λοιπόν απαιτούν έναν ξεχωριστό μηχανισμό δημιουργίας αντιγράφων.

Δεδομένου ότι ανά πάσα στιγμή μπορεί ένα σύστημα να υποστεί κάποιας μορφής αλλοίωση, όλα τα αποτελεσματικά εφεδρικά συστήματα παρέχουν τη δυνατότητα επαναφοράς εκδόσεων των αρχείων από προηγούμενα σημεία στο χρόνο. Ανά τακτά χρονικά διαστήματα λαμβάνονται αυτόματα από το σύστημα αντίγραφα ασφαλείας των σημαντικών αρχείων του λειτουργικού, δημιουργώντας έτσι μία χρονική σειρά αντιγράφων, τα οποία είναι διαθέσιμα προς χρήση από την ρουτίνα επαναφοράς δεδομένων.

### 5.3.8. Έλεγχοι κατανομής πόρων

Δεδομένου ότι το UNIX είναι ένα πολυχρηστικό σύστημα, όπως έχουμε αναλύσει παραπάνω, είναι απαραίτητη η επιβολή ορίων στους πόρους του συστήματος για να διασφαλιστεί ότι κανένας χρήστης δεν μπορεί, είτε τυχαία είτε σκόπιμα, να προκαλέσει την αποσταθεροποίηση του συστήματος χρησιμοποιώντας όλους τους διαθέσιμους πόρους. Δεδομένου ότι οι κατανομές πόρων διαφέρουν σημαντικά μεταξύ των συστημάτων και των διαθέσιμων εφαρμογών, οι διαχειριστές πρέπει να καθορίσουν τα κατάλληλα όρια για το σύστημα ξεχωριστά.

Οι διαχειριστές των συστημάτων έχουν την δυνατότητα είτε να προχωρήσουν σε περιορισμούς που αφορούν τον κάθε πόρο του συστήματος ξεχωριστά είτε να περιορίσουν μεμονωμένες διεργασίες.

Το λογικό σύστημα PAM, που αναφερθήκαμε παραπάνω, έχει την δυνατότητα επιβολής περιορισμών συγκεκριμένου πόρου για ολόκληρες περιόδους λειτουργίας χρήστη. Οι περιορισμοί που επιβάλλει το εν λόγω σύστημα ενδέχεται να παρακάμπτονται, αλλά παρέχουν κάποια άμυνα έναντι των τυχαίων προβλημάτων που μπορεί να προκύψουν.

### 5.3.9. Παρακολούθηση και έλεγχος

Οι κατασκευαστές του UNIX προκειμένου να ενισχύσουν την ασφάλεια τους έχουν ενσωματώσει στο λειτουργικό, διεργασίες ( syslog και klogd ) οι οποίες λαμβάνουν αναφορές ασφαλείας και λειτουργίας από πολλά διαφορετικά στοιχεία του συστήματος. Μετά τον έλεγχο και την ανάλυση αυτών των αναφορών, πολλές UNIX εκδόσεις λειτουργικού, αποστέλλουν τα αποτελέσματα στον διαχειριστή του συστήματος ( root account ) μέσω email και της SMTP υπηρεσίας.

Η παραπάνω λειτουργία είναι πολύ σημαντική και στον τομέα της δικτύωσης όπου το UNIX παίζει σημαντικό ρόλο. Πολλά κομμάτια των δικτύων, όπως πχ δρομολογητές, υποστηρίζουν τα syslog και SNMP πρωτόκολλα. Έτσι δίνουν την δυνατότητα στον διαχειριστή του συστήματος/δικτύου να λαμβάνει αναφορές ασφαλείας τόσο για το σύστημα όσο και για το εμπλεκόμενο δίκτυο, ταυτόχρονα.

### 5.3.10. Το τείχος προστασίας του συστήματος

Η ρουτίνα ελέγχου δικτύων ( netfilter ) που περιλαμβάνεται στο πυρήνα του UNIX φιλτράρει και μπλοκάρει εισερχόμενες και εξερχόμενες συνδέσεις δικτύου σύμφωνα με ένα σύνολο κανόνων που έχουν οριστεί από τον διαχειριστή. Διάφορες διανομές UNIX έχουν προρυθμισμένο τείχος προστασίας βάση προεπιλεγμένων

κανόνων, και παρέχουν βοηθητικά εργαλεία διαχείρισης και αλλαγής των ρυθμίσεων αυτών.

Εκδόσεις όπως το Fedora, το Red Hat και το SUSE ενεργοποιούν αυτόματα το τείχος προστασίας και παρέχουν τα δικά τους βοηθητικά προγράμματα γραφικής παρακολούθησης. Σε άλλες εκδόσεις όπως το Debian και το Ubuntu χρειάζεται παραμετροποίηση από τους διαχειριστές. Οι τρέχουσες εκδόσεις του Ubuntu περιλαμβάνουν ένα βοηθητικό πρόγραμμα γραμμής εντολών που ονομάζεται ufw και χρησιμεύει στην παραμετροποίηση του τείχους προστασίας.

Οι διανομές UNIX που ενεργοποιούν το τείχος προστασίας από προεπιλογή χρησιμοποιούν την ρουτίνα ελέγχου δικτύων ( netfilter ) η οποία αποκλείει τις συνδέσεις από άλλα συστήματα. Οποιαδήποτε προσπάθεια απόκτησης πρόσβασης σε μία υπηρεσία της οποίας η θύρα ( port ) πρόσβασης είναι μπλοκαρισμένη από την εν λόγω ρουτίνα απλά αποτυγχάνει. Αυτό σημαίνει ότι κανένα άλλο απομακρυσμένο σύστημα δεν μπορεί να συνδεθεί σε μία εγκατεστημένη υπηρεσία, εκτός εάν ο διαχειριστής επιλέξει συγκεκριμένα να ξεμπλοκάρει τη σχετική θύρα.

### 5.3.11. Απομόνωση εφαρμογών

Όλες σχεδόν οι UNIX εκδόσεις λειτουργικών συστημάτων παρέχουν διάφορες μεθόδους περιορισμού της ικανότητας ενός προγράμματος να επηρεάσει είτε άλλα προγράμματα είτε το ίδιο το σύστημα του κεντρικού υπολογιστή. Οι σημαντικότερες μέθοδοι είναι:

- Ο υποχρεωτικός έλεγχος πρόσβασης (MAC), όπου συμπληρώνει τις βασικές ρουτίνες ασφαλείας ενός UNIX συστήματος, επιβάλλοντας απόλυτα όρια που δεν μπορούν να παρακαμφθούν από κανένα πρόγραμμα ή λογαριασμό.

- Η εικονικοποίηση ( Virtualization ) σάς επιτρέπει να εκχωρήσετε ένα περιορισμένο σύνολο πόρων υλικού σε μια εικονική μηχανή, το οποίο μπορεί να παρακολουθείται και να υποστηρίζεται από ξεχωριστές ρουτίνες στο σύστημα του κεντρικού υπολογιστή.

- Οι εγκαταστάσεις UNIX Container , όπως το Docker, εκτελούν ρουτίνες εντός ενός νέο-δημιουργημένου καταλόγου συστήματος και τις διαχωρίζουν από τις ρουτίνες του βασικού συστήματος.

- Το βοηθητικό πρόγραμμα chroot εκτελεί προγράμματα εντός ενός καθορισμένου καταλόγου εργασίας και αποτρέπει την πρόσβαση σε οποιονδήποτε άλλο κατάλογο σε αυτό το σύστημα.

Οι διαχειριστές τέτοιων συστημάτων μπορούν να φιλοξενήσουν λειτουργικά συστήματα σε κατάλληλα εικονικά περιβάλλοντα για συγκεκριμένες εργασίες και περιορίσουν αυτούς τους επισκέπτες πολύ περισσότερο από όσο θα ήταν δυνατό σε ένα σύστημα πολλαπλών χρήσεων. Κάθε τέτοιο σύστημα μπορεί να περιλαμβάνει πολύ λιγότερο λογισμικό και αυτό απλοποιεί επίσης κάθε διαχειριστική εργασία, συμπεριλαμβανομένων των ρυθμίσεων MAC. Ούτε το MAC πρωτόκολλο ούτε η διαδικασία της εικονικοποίησης εμποδίζουν την ύπαρξη μεμονωμένων εφαρμογών ή υπηρεσιών από το να παραβιαστούν, να δυσλειτουργήσουν ή να παραμετροποιηθούν λανθασμένα. Μπορούν όμως να αποτρέψουν την κλιμάκωση αυτών των προβλημάτων.

Σε σύγχρονες Linux εκδόσεις λογισμικού η ρουτίνα SELinux μπορεί να παρέχει τις υπηρεσίες του πρωτοκόλλου MAC, για την επιβολή μιας πολιτικής που καθορίζει την επιτρεπόμενη πρόσβαση σε προγράμματα ή λογαριασμούς του συστήματος. Η ρουτίνα SELinux δημιουργήθηκε στην πραγματικότητα από την NSA για την κάλυψη των αναγκών των κρατικών υπηρεσιών που χειρίζονται ταξινομημένα δεδομένα και επιτρέπει στους διαχειριστές να αναπτύσσουν εξαιρετικά λεπτομερείς και ακριβείς ρυθμίσεις ασφαλείας που επηρεάζουν την λειτουργικότητα ολόκληρου του συστήματος. Πολλοί προγραμματιστές και διαχειριστές θεωρούν την ρουτίνα SELinux πολύ υψηλό βάρος ώστε να εφαρμοστεί και να διαχειριστεί πλήρως.

Τα συστήματα Linux της Fedora και της Red Hat Enterprise περιλαμβάνουν αυτόματα μια περιορισμένη έκδοση του SELinux που περιορίζει πολλές τυπικές υπηρεσίες δικτύου, χωρίς να επηρεάζει τους χρήστες ή τα προγράμματα που εκτελούνται. Αυτές οι διανομές παρέχουν επίσης μερικά απλά εργαλεία διαχείρισης για την προσαρμογή της προεπιλεγμένης πολιτικής και αντιμετώπισης προβλημάτων του SELinux, αλλά όχι εργαλεία που θα βοηθήσουν στην ανάπτυξη νέων πολιτικών. Το Debian παρέχει την ρουτίνα SELinux, αλλά η υποστήριξη είναι περιορισμένη.

Το Ubuntu και το SUSE δεν ενεργοποιούν την ρουτίνα SELinux από προεπιλογή. Αντ 'αυτού, παρέχουν την ρουτίνα AppArmor . Η παραμετροποίηση της ρουτίνας

AppArmor είναι πολύ απλούστερη από το SELinux, αλλά προσφέρει περιορισμένες δυνατότητες.

Υπάρχουν αρκετές λύσεις ανοιχτού κώδικα για την εκτέλεση πλήρως λειτουργικών συστημάτων σε ένα εικονικό περιβάλλον. Μακράν τα πιο δημοφιλή είναι τα Xen και KVM. Το Xen σάς δίνει τη δυνατότητα να ρυθμίσετε ένα σύστημα για να λειτουργεί ως κεντρικός υπολογιστής για πολλαπλά εικονικά περιβάλλοντα, τα οποία όλα ελέγχονται από έναν μόνο υπερεπόπτη ( hypervisor ). Οι τρέχουσες linux διανομές σε μηχανές που περιλαμβάνουν CPU με υποστήριξη εικονικοποίησης μπορούν να εκτελούνται σε όλο και πιο απλά και ευέλικτα KVM.

Τα σύγχρονα συστήματα Linux περιλαμβάνουν υποστήριξη για κοντέινερ και παρέχουν εργαλεία που επιτρέπουν στον χρήστη την εύκολη χρήση και εγκατάσταση αυτών των εφαρμογών. Το Docker βασίζεται σε μια υπηρεσία παρασκηνίου που διαχειρίζεται τα κοντέινερ στο σύστημα κεντρικού υπολογιστή, και αυτό μπορεί να υποστηρίξει μεγάλο αριθμό κοντέινερ σε έναν μόνο κεντρικό υπολογιστή.

Η παλαιότερη εγκατάσταση chroot είναι παγκοσμίως διαθέσιμη, αλλά σχεδιάστηκε αρχικά ώστε να συμβάλλει στην εγγραφή λογισμικών και όχι για ασφάλεια, και μπορεί να παρακαμφθεί. Οι προγραμματιστές χρησιμοποιούν αυτήν τη δυνατότητα ώστε να μπορούν να κατασκευάζουν και να δοκιμάζουν λογισμικά σε καθαρό περιβάλλον. Ιστορικά, οι διαχειριστές χρησιμοποίησαν επίσης το chroot για να εκτελέσουν δυνητικά μη ασφαλή υπηρεσίες δικτύου όπως διακομιστές FTP σε ειδικά σχεδιασμένα περιβάλλοντα.

### 5.3.12. Ιοί και κακόβουλα λογισμικά

Όλες οι παραπάνω λειτουργίες ασφαλείας του UNIX συνδυάζονται ώστε να δημιουργήσουν ισχυρή άμυνα ενάντια σε κακόβουλα λογισμικά:

- Το λογισμικό διανέμεται σε μορφή πακέτων αντί προγραμμάτων.
- Ένα πρόγραμμα που βρίσκεται στον υπολογιστή, δεν μπορεί να εκτελεστεί μέχρι να χαρακτηριστεί ως εκτελέσιμο.

- Από προεπιλογή, εφαρμογές όπως η σουίτα OpenOffice.org και το πρόγραμμα ηλεκτρονικού ταχυδρομείου δεν πραγματοποιούν εκτέλεση χωρίς έγκριση προγραμμάτων ενσωματωμένα σε μηνύματα ηλεκτρονικού ταχυδρομείου ή έγγραφα.

- Τα προγράμματα περιήγησης ιστού απαιτούν από τον χρήστη να παρέχει την έγκριση του για την εγκατάσταση προσθηκών.

- Οι ευπάθειες λογισμικού μπορούν να κλείσουν γρήγορα από προμηθευτές οι οποίοι μπορούν να παρέχουν ενημερωμένα πακέτα στα διάφορα αποθετήρια.

Παρόλο που ένας ιός θα μπορούσε να γραφτεί ώστε να προσβάλλει τα τρέχοντα συστήματα που μοιάζουν με UNIX, δεν υπάρχει μέχρι στιγμής αποτελεσματικό κακόβουλο λογισμικό. Δεδομένου ότι ένα πιθανό μελλοντικό κακόβουλο θα χρειαζόταν τη συγκατάθεση ενός χρήστη στο σύστημα προκειμένου να εγκατασταθεί, μειώνεται έτσι σημαντικά η πιθανότητα οποιοδήποτε τέτοιο λογισμικό να εξαπλωθεί στα δίκτυα.

## 5.4. Βελτίωση ασφαλείας του Unix

Δυστυχώς, αυτοί οι μηχανισμοί δεν βοηθάνε πολύ όταν τα συστήματα είναι διαμορφωμένα εσφαλμένα, όταν χρησιμοποιούνται απρόσεκτα ή όταν το λειτουργικό είναι γεμάτο σφάλματα. Σχεδόν όλα τα κενά ασφαλείας που έχουν βρεθεί μέσα στο UNIX με τα χρόνια έχουν προκύψει από αυτού του είδους τα προβλήματα και όχι από ελλείψεις στον εγγενή σχεδιασμό του συστήματος.

### 5.4.1. Προσδοκίες

Το μεγαλύτερο πρόβλημα με την βελτίωση της ασφάλειας του UNIX είναι αναμφισβήτητα μια από τις προσδοκίες. Πολλοί χρήστες έχουν μεγαλώσει και έχουν συνηθίσει το UNIX να λειτουργεί με ένα συγκεκριμένο τρόπο. Η εμπειρία τους με το UNIX, είτε σε ακαδημαϊκό είτε σε ερευνητικό επίπεδο, ήταν πάντα ότι έχουν πρόσβαση στους περισσότερους καταλόγους και στις περισσότερες εντολές του συστήματός. Οι χρήστες έχουν συνηθίσει να κάνουν, από προεπιλογή, τα αρχεία τους αναγνώσιμα από όλους. Επίσης οι χρήστες συχνά συνηθίζουν να μπορούν να κατασκευάζουν και να εγκαθιστούν το δικό τους λογισμικό, συχνά απαιτώντας προνόμια συστήματος για να το κάνουν. Η τάση στις "δωρεάν" εκδόσεις του UNIX για συστήματα προσωπικών υπολογιστών έχει ενισχύσει αυτές τις προσδοκίες.

Δυστυχώς, όλες αυτές οι προσδοκίες είναι αντίθετες με τις ορθές πρακτικές ασφάλειας στον χώρο των επιχειρήσεων. Για μεγαλύτερη ασφάλεια, οι διαχειριστές συστήματος πρέπει συχνά να περιορίζουν την πρόσβαση σε αρχεία και εντολές που δεν είναι αυστηρά απαραίτητα ώστε οι χρήστες να κάνουν τη δουλειά τους. Έτσι, κάποιος που χρειάζεται email και έναν επεξεργαστή κειμένου για τη δουλειά του δεν πρέπει επίσης να περιμένει ότι θα είναι σε θέση να προχωρήσει σε εκτέλεση των διαγνωστικών προγραμμάτων δικτύου και του μεταγλωττιστή C. Επίσης, για υψηλή ασφάλεια, οι χρήστες δεν θα πρέπει να είναι σε θέση να εγκαταστήσουν λογισμικό που δεν έχει εξεταστεί και εγκριθεί από εκπαιδευμένο και εξουσιοδοτημένο άτομο.

Η παράδοση της ανοικτής πρόσβασης είναι ισχυρή και είναι μία από τους λόγους που το UNIX ήταν και είναι ελκυστικό για τόσους πολλούς ανθρώπους.

Ορισμένοι χρήστες υποστηρίζουν ότι ο περιορισμός αυτού του είδους της πρόσβασης θα έκανε τα συστήματα κάτι διαφορετικό από το UNIX. Αν και αυτά τα επιχειρήματα μπορεί να είναι έγκυρα, σε περιπτώσεις όπου απαιτείται ισχυρή ασφάλεια, μπορεί να χρειαστούν περιοριστικά μέτρα.

Ταυτόχρονα, οι διαχειριστές μπορούν να ενισχύσουν την ασφάλεια του συστήματος εφαρμόζοντας κάποιες γενικές αρχές ασφάλειας. Για παράδειγμα, αντί να αφαιρεθούν όλοι οι μεταγλωττιστές και οι βιβλιοθήκες από κάθε μηχάνημα, τα εργαλεία μπορούν να προστατευτούν έτσι ώστε μόνο χρήστες οι οποίοι ανήκουν σε μία συγκεκριμένη ομάδα χρηστών, μπορούν να έχουν πρόσβαση σε αυτά. Παρόμοιες μέθοδοι μπορούν να χρησιμοποιηθούν και με άλλες κατηγορίες εργαλείων, όπως πχ το λογισμικό παρακολούθησης δικτύων.

Επιπλέον, η αλλαγή της θεμελιώδους άποψης σχετικά με την πρόσβαση στα δεδομένα του συστήματος μπορεί να είναι επωφελές: από αναγνώσιμο ως προεπιλογή σε μη αναγνώσιμο ως προεπιλογή. Για παράδειγμα, τα αρχεία και οι κατάλογοι χρηστών πρέπει να προστατεύονται ενάντια στην πρόσβαση για ανάγνωση αντί να είναι παγκοσμίως αναγνώσιμα από προεπιλογή. Ρυθμίζοντας σωστά την πρόσβαση στα αρχεία του συστήματος και χρησιμοποιώντας κρυφούς κωδικούς μπορεί να βελτιωθεί η συνολική ασφάλεια του UNIX.

Η περισσότερο κρίσιμη πτυχή στην ενίσχυση της ασφάλειας του UNIX είναι οι ίδιοι οι χρήστες να αλλάξουν τις προσδοκίες τους. Ο καλύτερος τρόπος για την επίτευξη αυτού του στόχου είναι μέσω της εκπαίδευσης και του κινήτρου. Πολλοί χρήστες άρχισαν να χρησιμοποιούν το UNIX σε ένα περιβάλλον λιγότερο απειλητικό από ό, τι αυτό που αντιμετωπίζουμε σήμερα. Εκπαιδεύοντας χρήστες σχετικά με τους κινδύνους και πώς η συνεργασία τους μπορεί να βοηθήσει στην αποτροπή αυτών, η ασφάλεια του συστήματος αυξάνεται. Με σωστό τρόπο παρακινώντας τους χρήστες να συμμετέχουν σε καλές πρακτικές ασφάλειας, τους καθιστούν μέρος του μηχανισμού ασφαλείας.

### 5.4.2. Unix και ποιότητα εξυπηρέτησης

Μεγάλο μέρος του UNIX λειτουργικού και τα βοηθητικά προγράμματα που οι άνθρωποι θεωρούν δεδομένα γράφτηκαν από φοιτητές που συμμετείχαν σε ερευνητικά έργα, ή ως γρήγορες υλοποιήσεις από προγραμματιστές λογισμικού μέσα σε ερευνητικά εργαστήρια. Αυτά τα προγράμματα δεν σχεδιάστηκαν και δεν δοκιμάστηκαν επίσημα: συνενώθηκαν και διορθώθηκαν γρήγορα και χωρίς μελέτη. Το αποτέλεσμα είναι μια μεγάλη συλλογή εργαλείων που συνήθως λειτουργούν, αλλά μερικές φορές αποτυγχάνουν με απροσδόκητους και θεαματικούς τρόπους. Τα βοηθητικά προγράμματα δεν ήταν τα μόνα πράγματα που έγραψαν σπουδαστές. Μεγάλο κομμάτι του BSD UNIX, συμπεριλαμβανομένου του κώδικα δικτύωσης, γράφτηκε από σπουδαστές σε ερευνητικά έργα. Αυτές οι προσπάθειες αγνοούσαν μερικές φορές τα υπάρχοντα πρότυπα και τις υπάρχουσες συμβάσεις.

Ωστόσο, υπάρχουν δύο παράγοντες που δεν μπορούν να βελτιωθούν απλώς εντοπίζοντας και διορθώνοντας τα σφάλματα στον ελλαττωματικό κώδικα. Το πρώτο είναι ότι οι σχεδιαστές λογισμικού δεν μαθαίνουν από τα λάθη του παρελθόντος. Για παράδειγμα, η υπέρβαση του buffer (κυρίως προκύπτει από buffers σταθερού μήκους και συναρτήσεις που δεν ελέγχουν τις μεταβλητές τους) έχουν αναγνωριστεί ως κυρίαρχα προβλήματα του UNIX για μεγάλο χρονικό διάστημα. Ωστόσο, ακόμα το λογισμικό συνεχίζει να περιέχει τέτοια σφάλματα και τα νέα λογισμικά γράφονται χωρίς να λαμβάνουν υπόψιν αυτά τα προβλήματα.

Ένα σοβαρότερο πρόβλημα από οποιοδήποτε άλλο ελάττωμα είναι το γεγονός ότι λίγοι, αν υπάρχουν, προμηθευτές κατασκευάζουν οργανωμένα προγράμματα εκτελώντας δοκιμές στο λογισμικό που παρέχουν. Αν και πολλοί κατασκευαστές δοκιμάζουν τα λογισμικά τους ώστε να βεβαιωθούν ότι είναι σύμφωνα με τα "πρότυπα" της βιομηχανίας, λίγοι προφανώς προχωράνε σε δοκιμές του παραγόμενου λογισμικού ώστε να δουν την συμπεριφορά του σε απροσδόκητα δεδομένα ή συνθήκες. Με ποσοστό έως και 40% τα βοηθητικά προγράμματα σε ορισμένα μηχανήματα περιέχουν σφάλματα στον κώδικά τους.

Όσο οι χρήστες απαιτούν αυστηρή συμμόρφωση της συμπεριφοράς στις υπάρχουσες εκδόσεις των προγραμμάτων, καθώς και όσο η ποιότητα του λογισμικού

δεν αποτελεί βασικό κριτήριο αγοράς από τους ίδιους χρήστες, οι προμηθευτές πιθανότατα θα κάνουν πολύ λίγα για συστηματική δοκιμή και διόρθωση της ασφάλειας του λογισμικού τους. Επίσημα πρότυπα, όπως τα ANSI C και POSIX διαιωνίζουν και επισημοποιούν αυτές τις αδυναμίες.

### 5.4.3. Πρόσθετες Λειτουργικότητες

Μια άλλη όψη του προβλήματος έχει να κάνει με τις "βελτιώσεις" που προσθέτει ο κάθε κατασκευαστής. Αντί να γίνει μια ενιαία και απλή διεπαφή για τη διαχείριση του συστήματος σε όλες τις πλατφόρμες, ο κάθε κατασκευαστής δημιουργούσε ένα νέο σύνολο εντολών και λειτουργιών. Αποτέλεσμα αυτού είναι να υπάρχουν πλέον εκατοντάδες (ίσως και χιλιάδες) νέες εντολές, επιλογές, κελύφη, δικαιώματα και ρυθμίσεις που ο διαχειριστής ενός ετερογενούς υπολογιστικού περιβάλλοντος πρέπει να καταλαβαίνει και να θυμάμαι. Επιπλέον, πολλές από τις εντολές και τις επιλογές είναι παρόμοιες μεταξύ τους, αλλά έχουν διαφορετική σημασία αναλόγως του περιβάλλοντος όπου χρησιμοποιούνται. Αυτή η πολυπλοκότητα περιπλέκει περαιτέρω την ανάπτυξη εργαλείων που προορίζονται ώστε να παρέχουν υποστήριξη και έλεγχο μεταξύ των διαφορετικών πλατφορμών. Το UNIX σχεδιάστηκε ως ένα τυποποιημένο λειτουργικό σύστημα, και έχει πλέον καταλήξει ως ένα από τα πιο μη τυποποιημένα συστήματα για διαχείριση.

### 5.4.4. Ασφάλεια στο Unix και Διαδίκτυο

Μια τελευταία επίδραση στην ασφάλεια του UNIX περιλαμβάνει τον τρόπο με τον οποίο έχουν προστεθεί νέες λειτουργίες ανά χρόνια. Το UNIX χαρακτηρίζεται για την ευελιξία του και την επαναχρησιμοποίηση χαρακτηριστικών · Συνεπώς, νέες λειτουργίες συνεχώς χτίζονταν πάνω από UNIX πλατφόρμες και τελικά ενσωματωνόταν σε εκδόσεις που κυκλοφόρησαν. Δυστυχώς, η προσθήκη νέων δυνατοτήτων γίνεται συχνά χωρίς την κατανόηση των υποθέσεων που έγιναν με τους

υποκείμενους μηχανισμούς και χωρίς ανησυχία για την πρόσθετη πολυπλοκότητα που παρουσιάζεται στους χειριστές και τους συντηρητές συστήματος. Η εφαρμογή των ίδιων χαρακτηριστικών και του ίδιου κώδικα σε ετερογενή υπολογιστικά περιβάλλοντα μπορεί επίσης να οδηγήσει σε προβλήματα.

Σαν μία ειδική περίπτωση, σκεφτείτε πόσο τα μεγάλης κλίμακας δίκτυα υπολογιστών, όπως για παράδειγμα το Διαδίκτυο, έχει αλλάξει δραματικά τους βασικούς κανόνες ασφαλείας σε σχέση με εκείνουν που υπήρχαν όταν αναπτύχθηκε το UNIX. Το UNIX αναπτύχθηκε αρχικά σε ένα περιβάλλον όπου οι υπολογιστές δεν συνδέονταν μεταξύ τους εκτός των ορίων του μικρού δωματίου ή του ερευνητικού εργαστηρίου στο οποίο ανήκαν. Τα δίκτυα σήμερα διασυνδέουν δεκάδες χιλιάδες μηχανές, και εκατομμύρια χρήστες, σε κάθε ήπειρο του κόσμου. Για αυτόν ακριβώς τον λόγο ο καθένας από εμάς αντιμετωπίζει απευθείας ζητήματα ασφαλείας.

Ωστόσο η ασφάλεια του UNIX διαμορφώθηκε βασιζόμενη στην παλαιότερη, πιο περιορισμένη άποψη των δικτύων και όχι από τις πιο πρόσφατες εμπειρίες μας. Για παράδειγμα, η έννοια του ελέγχου των αναγνωριστικών χρήστη και των αναγνωριστικών ομάδας καθώς και η πρόσβαση σε αρχεία αναπτύχθηκε σε μια εποχή που το φυσικό μηχάνημα, το οποίο ήταν εξοπλισμένο με UNIX λειτουργικό, ήταν στημένο σε ένα ασφαλές περιβάλλον. Μέσα σε ένα κλειστό εργαστήριο με μηχανήματα συνδεόμενα σε ένα τοπικό δίκτυο, χρησιμοποιώντας μόνο σχετικά αργούς υπολογιστές, αυτός ο σχεδιασμός λειτουργούσε καλά. Τώρα όμως, με την ραγδαία αύξηση των σταθμών εργασίας και με UNIX μηχανές στα διεθνή δίκτυα, αυτός ο σχεδιασμός, με τις υποθέσεις του σχετικά με την περιορισμένη πρόσβαση στο δίκτυο, οδηγεί σε μεγάλες αδυναμίες στην ασφάλεια.

Οι προγραμματιστές του UNIX όμως δεν ασχολήθηκαν με όλα αυτά τα ζητήματα ασφαλείας. Η σουίτα πρωτοκόλλου IP στην οποία βασίζεται το Διαδίκτυο, αναπτύχθηκε έξω από UNIX αρχικά, και αναπτύχθηκε χωρίς επαρκή ανησυχία για την αυθεντικοποίηση και την εμπιστευτικότητα.

**Βιβλιογραφικές αναφορές**


*Linux and UNIX Security Features*. (2016). FIELD NOTES. https://www.stuartellis.name/articles/unix-security-features/

Aho, A. V., University, C., Lam, M. S., University, S., Sethi, R., Avaya, Ullman, J. D., & University, S. (2007). *Compilers Second Edition Principles, Techniques, & Tools* (E. E. M. Hirsch, A. E. M. Goldstein, P. E. K. Harutunian, A. M. E. J. Holcomb, C. D. J. C. Wells, D. A. M. M. Groth, M. P. B. Tidd, S. M. M. M. Brown, M. A. S. Milmore, S. A. Support/, T. S. J. Vetere, & S. M. B. C. Melville (Eds.); Second). Greg Tobin.

Andrew, T. S. (2008). *Modern Operating Systems* (3rd ed.). Pearson Prentice Hall.

Babbage, C. (2008). *Reflections On the Decline of Science in England, and Some of Its Causes* (D. W. Anonymous Voluntee (Ed.)). Project Gutenberg's. https://www.gutenberg.org/files/1216/1216-h/1216-h.htm

Bowen, J., Trickett, T., Green, J. B. A., & Lomas, A. (2018). Turing's Genius – Defining an apt microcosm. In *Electronic Visualisation and the Arts* (p. 8). Electronic Visualisation and the Arts (EVA 2018) (EVA). https://doi.org/10.14236/ewic/EVA2018.31

Brass, P. (2008). *Advanced Data Structures*. Cambridge University Press. https://doi.org/https://doi.org/10.1017/CBO9780511800191

Bullynck, M., Daylight, E.G., De Mol, L. (2015). Why did computer science make a hero out of Turing? *Communications of the ACM*, *58*(3), 37–39. https://doi.org/10.1145/2658985

Campbell-Kelly Martin, Aspray William, Ensmenger Nathan, Y. R. J. (2013). *A History of the Information Machine* (Third Edit). Westview Press.

Cooper S. Barry, V. L. j. (2013). *Alan Turing: His Work and Impact*. Elsevier Science.

David, M. A. (2008). *Digital Apollo: Human and Machine in Spaceflight*. MIT Press.

Daylight, E. G. (2015). Towards a Historical Notion of 'Turing—the Father of Computer Science.' *History and Philosophy of Logic*, *36*(3), 205–228. https://doi.org/10.1080/01445340.2015.1082050

Donald, K. E. (1998). *The art of computer programming* (2nd ed.). Addison-Wesley Pub (Sd).

Eric, R. S. (2003). *The Art of Unix Programming*. Addison-Wesley Profrssional.

Fuegi John, F. J. (2003). Lovelace & Babbage and the Creation of the 1843 'Notes.' *IEEE Annals of the History of Computing*, *25*(4), 16–26. https://doi.org/10.1109/mahc.2003.1253887

Grampp F. T., M. H. R. (1984). UNIX Operating System Security. *AT&T Bell Laboratories Technical Journal*, *63*(8).

Hammerman, R., & Russell, A. L. (2015). *Ada's Legacy: Cultures of Computing from the Victorian to the Digital Age*. Association for Computing Machinery. https://doi.org/10.1145/2809523

J., O.-S. R. (1965). Irascible Genius (Charles Babbage). *The Computer Journal*, *7*(4), 277. https://doi.org/10.1093/comjnl/7.4.277

Jack, C. (2012). Alan Turing: The codebreaker who saved 'millions of lives'. *BBC News*. https://www.bbc.com/news/technology-18419691

James, E. (2014). *Ada's Algorithm: How Lord Byron's Daughter Ada Lovelace Launched the Digital Age*. Melville House.

Jennifer, K. (n.d.). *Open Source Software as Lead User's Make or Buy Decision: A Study of Open and Closed Source Quality*.

Jonathan P. Bowen. (2017). Alan Turing: Founder of computer science. *Engineering Trustworthy Software Systems, Second International School*, 15.

Kurose James, R. K. (2015). *No TitleComputer Networking: A Top-Down Approach* (6th ed.). Pearson.

Nitesh, D. (2003). *HackNotes™ Linux and Unix Security Portable Reference*. The McGraw-Hill/Osborne Companies.

Paul, C. E. (2003). *A History of Modern Computing* (2nd ed.). MIT Press.

R. Campagna, S. Cuomo, F. Giannino, G. S. G. T. (2018). A Semi-Automatic Numerical Algorithm for Turing Patterns Formation in a Reaction-Diffusion Model. *IEEE Access*, *6*, 4720–4724. https://doi.org/10.1109/ACCESS.2017.2780324

Raghunathan Srinivasan, Prasad Ashutosh, Mishra k. Birendra, C. H. (2995). Open Source Versus Closed Source: Software Quality in Monopoly and Competitive Markets. *MAN, AND CYBERNETICS*, 17.

Russell Stuart, N. P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall.

Silverstone Roger, H. L. (1996). Design and the domestication of information and communication technologies: technical change and everyday life. In

*Communication by Design: The Politics of Information and Communication Technologies* (pp. 44–74). Oxford University Press. http://eprints.lse.ac.uk/id/eprint/64821

Simson Garfinkel, Gene Spafford, A. S. (2011). *Practical Unix and Internet Security* (3rd ed.). O'Reilly Media Publication.

V, B. (2016). Computability and Analysis, a Historical Approach. In *Pursuit of the Universal* (Vol. 9709, p. 12). Springer. https://doi.org/10.1007/978-3-319-40189-8_5