Τεχνικές διασφάλισης

- Εκτεταμένες δοκιμές, καλές πρακτικές ανάπτυξης λογισμικού
  - Στην πράξη, δεν έχει δώσει αποτελέσματα
- Ανάλυση του συστήματος για εντοπισμό πιθανών αδυναμιών
  - Ας γνωρίζουμε εμείς, πριν το μάθουν οι κακοί
  - Δημιουργία «Ομάδας τίγρη»

- Πρόληψη ή ανίχνευση των προσπαθειών εκμετάλλευσης των αδυναμιών

Αδυναμίες συστημάτων

- Η ασφάλεια των συστημάτων διακυβεύεται από:
  - Προγραμματιστικά σφάλματα σε μεμονωμένες διεργασίες
    - » Υπερχείλιση ενδιάμεσης μνήμης
    - » Συνθήκες ανταγωνισμού
    - » Δούρειοι ίπποι
  - Απρόσμενες αλληλεπιδράσεις μεταξύ προγραμμάτων
    - » Εσφαλμένα δικαιώματα αρχείων
    - » Δομή και περιεχόμενο αρχείων διαμόρφωσης

Η άμυνα έναντι των επιθέσεων

	Ανάλυση (στατική μέθοδος)	Ανίχνευση-επιβολή πολιτικών (δυναμική μέθοδος)
Προγραμματιστικά σφάλματα	Ανάλυση πρωτογενούς κώδικα για ανίχνευση σφαλμάτων	Παρακολούθηση συμπεριφοράς και επιβολή ασφαλών προτύπων για τα προγράμματα
Σφάλματα ολοκλήρωσης	Ανάλυση διαμόρφωσης συστήματος για εντοπισμό αδυναμιών	Ανίχνευση προσπαθειών για εκμετάλλευση αδυναμιών

Προσεγγίσεις στην ανίχνευση εισβολών

- Ανίχνευση αδόκιμων τρόπων χρήσης
  - Κωδικοποίηση των αδόκιμων τρόπων χρήσης και ανίχνευση εμφανίσεών των
- Ανίχνευση μη φυσιολογικών συμπεριφορών
  - Μαθαίνουμε τη «φυσιολογική συμπεριφορά» και ανιχνεύουμε παρεκκλίσεις
- Ανίχνευση βάσει προδιαγραφών
  - Καθορίζεται η προτιθέμενη συμπεριφορά των προγραμμάτων έναντι της κωδικοποίησης των αδόκιμων τρόπων χρήσης

Κανάλια διαρροής

- Με αποθήκευση δεδομένων
  - Π.χ. τιμή ενός σημαφόρου
- Με χρονισμό
  - Ο χρόνος διεκπεραίωσης μιας εργασίας διαρρέει πληροφορία
- Με οποιουδήποτε άλλου είδους συμπεριφορά
  - αυξομείωση μνήμης ή μεγέθους αρχείων, κλειδώματα σε πόρους, χρήση ΚΜΕ

Ασφάλεια στον προγραμματισμό

- Τα λάθη
  - «Η συγγραφή κώδικα δεν έχει σχέση με την ασφάλεια»
  - «Το μόνο που θέλω είναι να τελειώνω με το πρόγραμμά μου»
  - «Μεθαύριο το παραδίδουμε στον πελάτη – τι μου λες για ασφάλεια τώρα;»
- Τα αποτελέσματα (όταν αναδειχθούν τα προβλήματα)
  - Οι προγραμματιστές σταματάνε τη δουλειά τους και ασχολούνται με την επιδιόρθωση του προβλήματος
  - Ειδοποιούνται οι πελάτες/χρήστες
  - Αποστέλλεται η νέα έκδοση
  - Εγκαθίσταται η νέα έκδοση (με πιθανή συνδρομή τεχνικών της εταιρείας)
  - Ζημιά στην εικόνα της εταιρείας
    - » Ειδικά αν πρόκειται για τράπεζες και παρόμοιους οργανισμούς

Ασφάλεια στον προγραμματισμό

- Απαραίτητο συστατικί της διαδικασίας διασφάλισης ποιότητας του λογισμικού
  - Γνώση από τους προγραμματιστές όλων των πιθανών ευπαθειών ασφάλειας της γλώσσας και του περιβάλλοντος προγραμματισμού
  - Επιθεωρήσεις ασφαλείας για τον κώδικα
    - » Από προγράμματα ανίχνευσης ευπαθειών
    - » Από εξειδικευμένους προγραμματιστές

Παράδειγμα προγράμματος ελέγχου ευπαθειών

- Πρόγραμμα:

```
#include <stdio.h>
#include <string.h>

int main(void) {
char s1[100], s2[80];

puts("Enter a string: ");
gets(s1);
strcpy(s2, s1);
printf(s2);
return 0;
}
```

- Ανάλυση ευπαθειών

```
its4 probs.c
probs.c:8:(Urgent) gets
The input buffer can almost always be overflowed.
Use fgets(buf,size,stdin) instead.
----------------
probs.c:10:(Urgent) printf
Non-constant format strings can often be attacked.
Use a constant format string.
----------------
probs.c:9:(Very Risky) strcpy
This function is high risk for buffer overflows
Use strncpy instead.
----------------
```

Ευπάθειες στη C/C++

- Τα περισσότερα προβλήματα ασφάλειας έχουν εντοπισθεί σε κώδικα που γράφηκε σε C
  - Η ευρεία χρήση της γλώσσας βοηθάει σ' αυτό
    - » Λειτουργικά συστήματα
    - » Βάσεις δεδομένων
    - » Παραθυρικά συστήματα
  - Αλλά και η «εμπιστοσύνη» που δείχνει στον προγραμματιστή

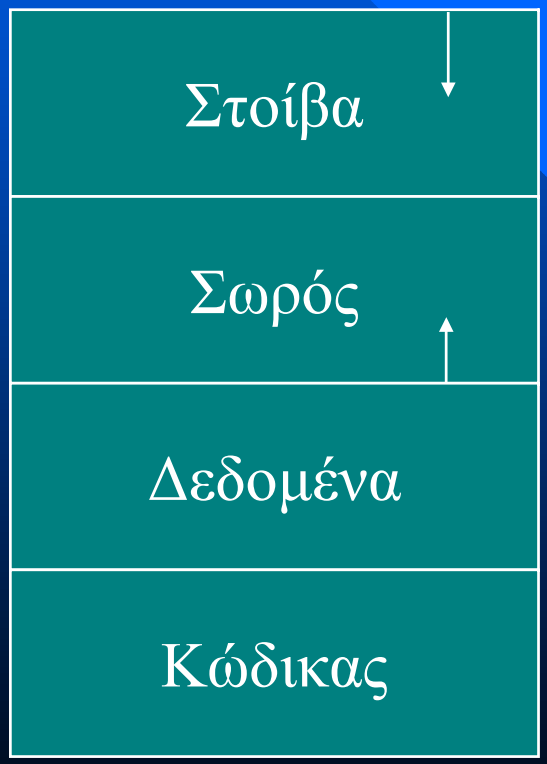
Υπερχείλιση ενδιάμεσης μνήμης

- Η χαρά του εισβολέα!
- Προϋποθέσεις εμφάνισης:
  - Λήψη δεδομένων από μη αξιόπιστη πηγή
    - » Πληκτρολόγιο, δίκτυο, αρχείο, διαδιεργασιακή επικοινωνία κ.λπ.
  - Αποθήκευση των δεδομένων σε ενδιάμεση μνήμη περιορισμένου μεγέθους
  - … χωρίς κατάλληλο έλεγχο αν τα δεδομένα όντως χωράνε στη μνήμη
- Αποτελέσματα:
  - Αλλοίωση τιμών μεταβλητών
  - Αλλοίωση τιμών αποθηκευμένων καταχωρητών
  - Αλλοίωση διευθύνσεων επιστροφής και πιθανή εκτέλεση αυθαίρετου κώδικα

… με αντίστοιχες επιπτώσεις στην ασφάλεια

Υπερχείλιση ενδιάμεσης μνήμης

- Δομή μνήμης διεργασίας




```
int status = 0; int tbl[5000];
void f(int a, char *b, int c[1000]) {
    int i, j; double sum; …; return; }
int main(int argc, char *argv[]) {
    int x = 7, y = 2; int t[2000];
    …
    f(x + y, "str", &t[999]);
}
```

Παράμετροι main (argc, argv)
Διεύθυνση επιστροφής από main
Τοπικές μεταβλητές main
Παράμετροι f (a, b, c)
Διεύθυνση επιστροφής από f
Τοπικές μεταβλητές f

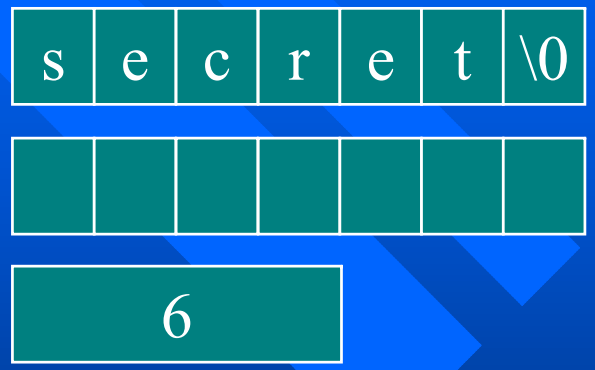
Υπερχείλιση ενδιάμεσης μνήμης για τοπικές μεταβλητές

```
int testPassword(void) {
int password_correct = 0, attempts = 0;
char buf[16];

while ((attempts < 3) && (password_correct == 0)) {
        printf("Enter password: ");
        gets(buf);
        if (strcmp(buf, "secret") == 0)
                password_correct = 1;
        else
                attempts++;
}
if (password_correct) return 1;
else return 0;
}
```

- Ο χρήστης εισάγει: AI SIXTIR KAI ESY KAI TO PASSWORD
- Η συνάρτηση επιστρέφει 1!

Υπερχείλιση ενδιάμεσης μνήμης για τοπικές μεταβλητές

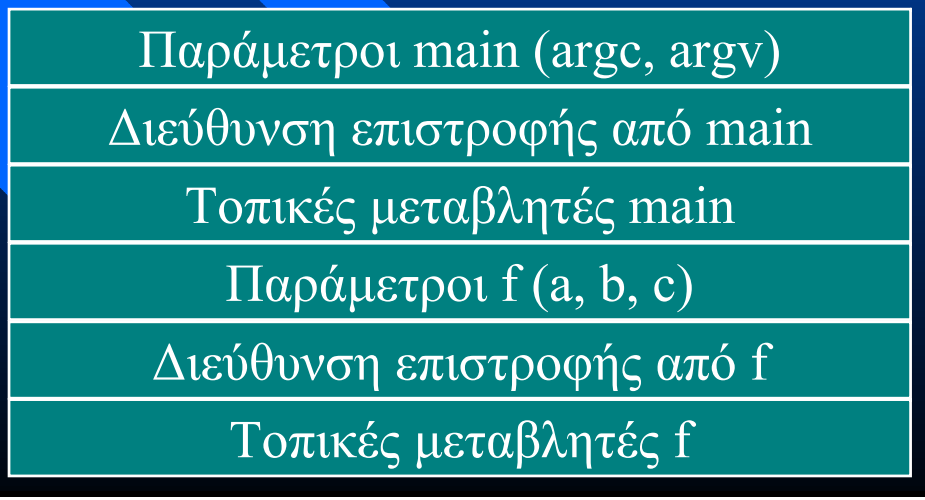
```
int testPassword(void) {
    char thePass[] = "secret";
    char userPass[7];
    int passlen = strlen(thePass);

    printf("Enter password:");
    gets(userPass);
    if (strncmp(thePass, userPass, passlen) == 0)
        return 1;
    else
        return 0;
}
```



- Ο χρήστης εισάγει: aaaaaaaaaaaaaaaa
- Η συνάρτηση επιστρέφει 1!

Υπερχείλιση τοπικής μνήμης για διευθύνσεις επιστροφής

- Όταν εκτελείται η εντολή return μιας συνάρτησης:
  - Καθαρίζεται η στοίβα από τυχόν τοπικές μεταβλητές
  - Εξάγεται η διεύθυνση επιστροφής από τη στοίβα και ο έλεγχος του προγράμματος μεταφέρεται εκεί
- Τι συμβαίνει αν η διεύθυνση επιστροφής έχει υπερκαλυφθεί με δεδομένα;
  - Καλή περίπτωση: το πρόγραμμα καταρρέει
  - Κακή περίπτωση: πρόβλημα στην ασφάλεια



Υπερχείλιση τοπικής μνήμης για διευθύνσεις επιστροφής

- Η καλή περίπτωση
  - Οι τιμές των δεδομένων που θα γραφούν στη στοίβα είναι τυχαίες και η διεύθυνση που θα σχηματιστεί αντιστοιχεί:
    - » Σε άκυρες διευθύνσεις μνήμης
    - » Σε έγκυρες διευθύνσεις που περιέχουν τυχαία bytes
- Η κακή περίπτωση:
  - Οι τιμές των δεδομένων που θα γραφούν στη στοίβα είναι τυχαίες και η διεύθυνση που θα σχηματιστεί αντιστοιχεί:
    - » Σε κώδικα που παρείχε ο εισβολέας ως τμήμα των δεδομένων
    - » Σε επιλεγμένες συναρτήσεις βιβλιοθήκης του συστήματος με κατάλληλες παραμέτρους
      - Π.χ. exec("sh", "/bin/sh", NULL);

Υπερχείλιση τοπικής μνήμης για διευθύνσεις επιστροφής

```
7000    int status = 0; int tbl[5000];
1000    void f(int a, char *b, int c[1000]) {
                int i, j; double sum; …; return; }
1100    int main(int argc, char *argv[]) {
Stack           int x = 7, y = 2; int t[2000];
…               …
1180            f(x + y, "str", &t[999]);
1194            return;
        }
1900    int exec(char *path, char *argv0, …);
```

Συνήθεις ύποπτοι για υπερχείλιση μνήμης

Συνάρτηση	Αντικαταστάτης
gets(myStr);	fgets(myStr, 80, stdin);
scanf("%s", mystr);	scanf("%80s", mystr);
sprintf(myStr, "%d%s%f", …);	snprintf(myStr, 80, "%d%s%f", …); sprintf(myStr, "%5d%.65s%8.2f", …);
strcpy(prayItFits, bigOne);	strncpy(prayItFits, bigOne, 80);
strcat(smallStr, possiblyBig);	strncat(smallStr, possiblyBig, 80 – strlen(smallStr));
while ((c = getchar()) != '\n') str[idx++] = c;	while ((idx < 80) && ((c = getchar()) != '\n')) str[idx++] = c;
char s[256]; getcwd(s, PATH_MAX);	char s[PATH_MAX + 1]; getcwd(s, PATH_MAX);

Υπερχειλίσεις μνήμης με λειτουργίες σωρού

```
printf("Enter array size: ");
scanf("%d", &numElements);
numBytes = numElements * sizeof(int);
if ((myArray = malloc(numBytes)) == NULL) {
    perror("Out of memory");
    exit(1);
}
for (i = 0; i < numElements; i++) myArray[numElements – 1] = 0;
```

numElements	numBytes
1	4
2	8
2147483647	4 (αν είμαστε τυχεροί 4294967292. Σωστό: 8589934588)

Παράλειψη προσδιορισμού μορφοποίησης στην printf

- Όταν η printf/fprintf κ.λπ. τυπώνει μία συμβολοσειρά και μόνο, πρέπει να καθορίζεται ως προσδιοριστής μορφοποίησης το "%s"
  - fgets(theString, 1024, stdin);
  - fprintf(myfile, theString);
- Τι θα συμβεί αν η συμβολοσειρά περιέχει %s, %d κ.λπ.;
  - Οι τιμές τους θα εξαχθούν από θέσεις μνήμης στη στοίβα (τοπικές παράμετροι, διευθύνσεις επιστροφής κ.λπ.)

Συνθήκες ανταγωνισμού

- Περιπτώσεις όπου η υλοποίηση ατομικών λειτουργιών με χρήση μη ατομικών λειτουργιών έχει επιπτώσεις στην ασφάλεια
- Παράδειγμα – πρόγραμμα εκτύπωσης (euid = 0 /* root */):

```
if(access(theFile, R_OK) == 0) {
    /* User has read permission */
    if((fd = open(theFile, O_RDONLY)) < 0)    {
            perror(theFile);
            exit(-1);
    }
    /* print file */
} else fprintf(stderr, "%s: permission dennied\n");
```

Συνθήκες ανταγωνισμού

- Παλιά υλοποίηση του mkdir:

```
mknod(path, S_IFDIR, dev);
/* Υπάρχει ο κατάλογος, ιδιοκτησία του root και είναι κενός */
chown(path, uid, gid);  /* Αλλαγή ιδιοκτήτη*/
chdir(path);
link(path, ".");        /* Δημιουργία καταχώρησης . */
link(parent, "..");     /* Δημιουργία καταχώρησης .. */}
```

- Ο «κακός» εκτελεί

```
while true
do
nice -39 mkdir foo &
rm -rf foo; ln /etc/passwd foo
rm -fr foo &
ls -l /etc/passwd
done
```

- … με στόχο η γραμμή «`rm -rf foo; ln /etc/passwd foo`» να εκτελεστεί *μετά* τη mknod και *πριν* την chown
- Λύση: κλήση συστήματος με ατομική υλοποίηση της δημιουργίας καταλόγου (mkdir)

Συνθήκες ανταγωνισμού

- Ο έλεγχος δικαιωμάτων έχει διαχωριστεί από το άνοιγμα αρχείου – σίγουρα το αρχείο δεν έχει αλλάξει;

```
while(1) lpr harmless

if(access(theFile, R_OK) == 0) {
   /* User has read permission */
      if((fd = open(theFile, O_RDONLY)) < 0)    {
      perror(theFile);
           exit(-1);
      }
      /* print the file */
}
else perror(theFile);
```

```
while(1) {
     creat("harmless", 0644);
     unlink("harmless");
     symlink("/etc/shadow", "harmless");
}
```

Συνθήκες ανταγωνισμού

- Αντιμετώπιση:
  - χρήση faccess όπου παρέχεται
    - » Δυστυχώς όχι σε πολλά συστήματα π.χ. Linux, OpenBSD, Solaris and AIX
  - Χρήση lstat για αποκλεισμό συμβολικών συνδέσμων
    - » Βοηθάει αλλά δεν αντιμετωπίζει τους «κανονικούς» συνδέσμους – περιορίζει το πρόβλημα στα πλαίσια ενός συστήματος αρχείων
    - » Περιορίζει τη λειτουργικότητα
  - Χρήση setegid() and seteuid()
    - » Το πρόγραμμα αποκτά την ενεργό ταυτότητα του χρήστη για να ανοίξει το αρχείο και κατόπιν επανέρχεται στην πρότερη ενεργό ταυτότητα (πρέπει να είναι ορισμένο το *_POSIX_SAVED_IDS* στο σύστημα) [επόμενη διαφάνεια]
  - fork() και μεταβίβαση περιγραφέα αρχείου
    - » Δημιουργούμε μία θυγατρική διεργασία με τα προνόμια του πραγματικού χρήστη. Αυτή ανοίγει το αρχείο και στέλνει τον *περιγραφέα αρχείου* στη γονική διεργασία

Συνθήκες ανταγωνισμού

```
uid_t euid, ruid;
gid_t egid, rgid;

euid = geteuid(); /* Αποθήκευση τρέχουσας ενεργού ταυτότητας χρήστη */
egid = getegid(); /* Αποθήκευση τρέχουσας ενεργού ταυτότητας ομάδας */
ruid = getuid(); /* Αποθήκευση τρέχουσας πραγματικής ταυτότητας χρήστη */
rgid = getgid(); /* Αποθήκευση τρέχουσας πραγματικής ταυτότητας ομάδας */

if(setegid(rgid) < 0) /* Η πραγματική ταυτότητα χρήστη τίθεται ως ενεργός */
    exit(1);
if(seteuid(ruid) < 0) /* Η πραγματική ταυτότητα χρήστη τίθεται ως ενεργός */
     exit(1);

open("...", ...);

if(setegid(egid) < 0) /* Αποκατάσταση αρχικής ενεργού ταυτότητας ομάδας */
     exit(1);
if(seteuid(euid) < 0) /* Αποκατάσταση αρχικής ενεργού ταυτότητας χρήστη */
     exit(1);
```

Προσωρινά αρχεία

- Ιδιαίτερα ύποπτα για διαρροή πληροφοριών αλλά και άλλα προβλήματα στην ασφάλεια
- Τα προσωρινά αρχεία δημιουργούνται σε καταλόγους που συνήθως είναι εγγράψιμοι απ' όλους
  - /tmp, /var/tmp
- Παλαιότερα ήταν δυνατόν να διαγραφούν από οποιονδήποτε
  - Αυτό αντιμετωπίσθηκε υποστηρίζοντας τη σημασιολογία του bit δικαιωμάτων t για καταλόγους:
  - chmod +t /tmp → drwxrwxrwt … /tmp
  - Μπορούμε να δημιουργήσουμε οποιοδήποτε αρχείο αλλά να σβήσουμε *μόνο* τα δικά μας

Προσωρινά αρχεία

- Δημιουργία αρχείου με:

```
char *tmp_name;
int tmpfd;
tmp_name = tmpnam(NULL);
if( (tmpfd = open(tmp_name, O_RDWR | O_CREAT | 0600)) < 0)
    exit(1);
unlink(tmp_name); /* Αδύνατη η αναφορά στο αρχείο, αυτόματη διαγραφή του
    όταν τερματίσει το πρόγραμμα */
```

- Πρόβλημα:
  - Το tmp_name δεν υπάρχει *όταν τερματίζει η tmpnam* – τι συμβαίνει όταν εκτελείται η open;

Προσωρινά αρχεία

- Λύση1: Χρήση mkstemp – Δημιουργία και *ταυτόχρονο άνοιγμα* αρχείου με ασφαλή τρόπο – *ΑΜΕΣΩΣ ΜΕΤΑ* αλλαγή δικαιωμάτων πρόσβασης
  - » fd = mkstemp("/var/tmp/atempFile.XXXXXX");
  - » fchmod(fd, 0600);
  - *Και αν προλάβει ο «κακός» να ανοίξει το αρχείο πριν το fchmod;*
- Λύση 2: αξιοποίηση umask για προληπτικό ορισμό δικαιωμάτων
  - » umask(066);
  - » fd = mkstemp("/var/tmp/atempFile.XXXXXX");
- Λύση 3: αξιοποίηση O_EXCL στην open
  - » tmp_name = tmpnam(NULL);
  - » if((tmpfd = open(tmp_name, O_RDWR | O_CREAT | O_EXCL | 0600)) < 0)
  - » exit(1);
  - » unlink(tmp_name); /* Αδύνατη η αναφορά στο αρχείο, αυτόματη διαγραφή του όταν τερματίσει το πρόγραμμα */

Εμπιστοσύνη στις μεταβλητές περιβάλλοντος

- Κάθε πρόγραμμα έχει μεταβλητές περιβάλλοντος που περιέχουν συγκεκριμένες πληροφορίες
  - Π.χ. το όνομα του χρήστη, τη διαδρομή αναζήτησης εντολών, διαδρομή αναζήτησης βιβλιοθηκών κ.λπ.
- Οι μεταβλητές είναι προσπελάσιμες μέσω της μεταβλητής environ
  - environ[0] = "USER=thomas";
  - environ[1] = "PATH=/usr/bin:/sbin:/usr/local/bin";
  - environ[2] = "LD_LIBRARY_PATH=/oracle/lib:/usr/lib";
- Τα προγράμματα (ειδικότερα τα πιο προνομιούχα) *ΔΕΝ* πρέπει να βασίζονται στις τιμές αυτές, καθώς μπορούν να τροποποιηθούν από τους χρήστες
  - USER=root; export USER

Εμπιστοσύνη στις μεταβλητές περιβάλλοντος

- Λύσεις:
  - Χρησιμοποιούμε κλήσεις συστήματος όπου είναι διαθέσιμες ως εναλλακτική
    - » getpwuid(getuid())->pw_name αντί getenv("USER");
  - Φροντίζουμε να έχουμε μόνο ασφαλή στοιχεία στο PATH όταν χρησιμοποιούμε τις execlp, execvp από προνομιούχα προγράμματα/χρήστες
    - » SUPATH
  - Το σύστημα πρέπει να αγνοεί επανορισμούς κλήσεων συστήματος (και των σχετικών συναρτήσεων βιβλιοθήκης) που εμφανίζονται σε «μη ασφαλείς» βιβλιοθήκες
    - » Π.χ. αγνοούμε την getuid() που φαίνεται να ορίζεται σε μία βιβλιοθήκη στον κατάλογο /home/users/snoopy/libs

Περιορισμός πόρων

- Περιορίζοντας τους πόρους μιας διεργασίας μπορεί να αποφευχθούν ορισμένες επιθέσεις τύπου άρνησης παροχής υπηρεσίας
  - Π.χ. εξάντληση της μνήμης, των καταχωρήσεων του πίνακα διεργασιών, του πλήθους των ανοικτών αρχείων
- Περιορίζοντας το μέγεθος του αρχείου αποτύπωσης μνήμης (προτιμότερα: μηδενίζοντάς το) προλαβαίνουμε τυχόν διαρροή πληροφοριών
  - passwd → pid = 2078
  - kill –ABRT 2078 → stop and core dump
  - strings core | grep root → extract root entry

Ασφάλεια δικτύων

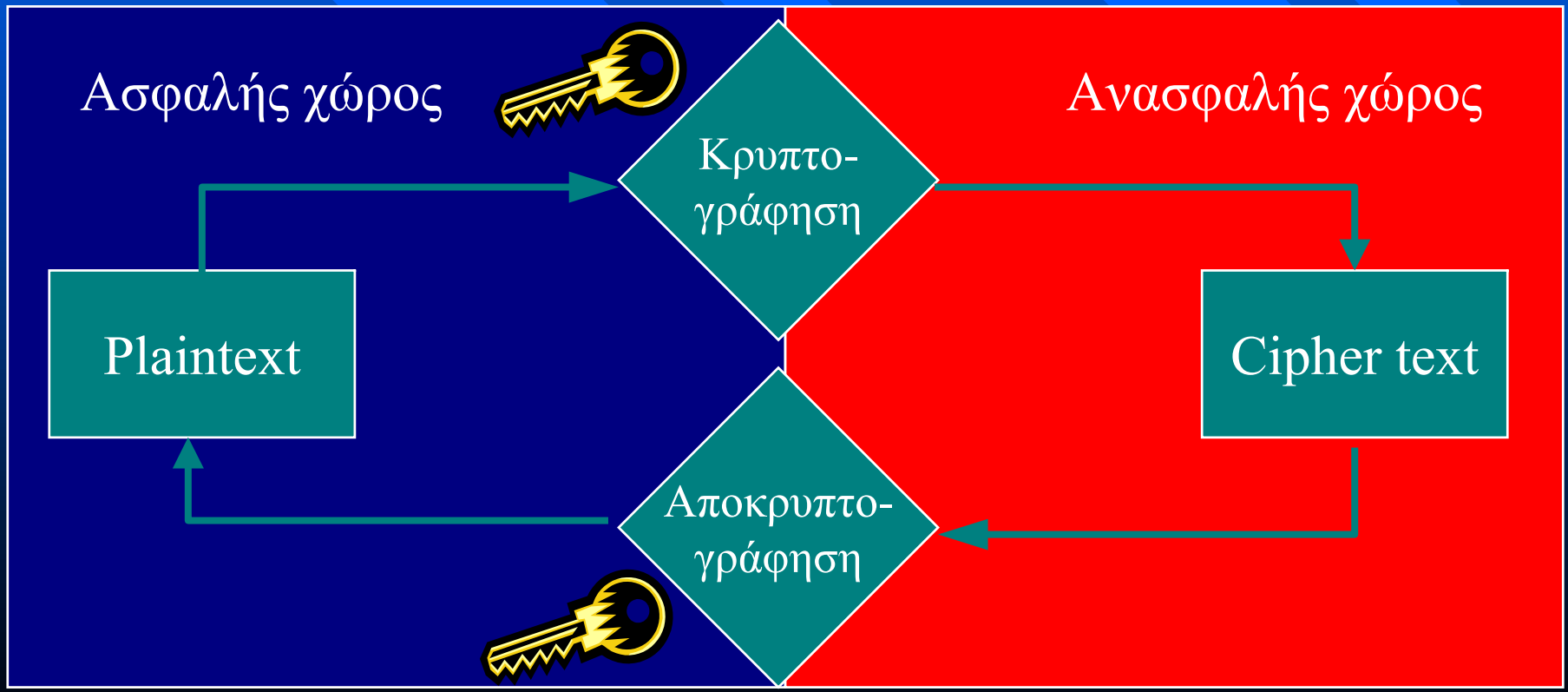
- Ασφάλεια επικοινωνιών
  - Ιδιωτικότητα
  - Αυθεντικότητα
  - Ακεραιότητα
  - Καταλογισμός και αδυναμία αποκήρυξης
- Προστασία πόρων

Κρυπτογραφία

- *Αρχικά:* Δεδομένα σε μη κρυπτογραφημένη μορφή (plaintext, clear text, *red*)
- *Κρυπτογράφηση*: απεικόνιση των δεδομένων σε μία ακατάληπτη μορφή (cipher text, *black*) με χρήση ενός *κλειδιού*
- *Κλειδί*: μία ακολουθία από bits
- *Αποκρυπτογράφηση*: η απεικόνιση της ακατάληπτης μορφής των δεδομένων στην αρχική, με χρήση ενός κλειδιού
- Τα κλειδιά μπορούν να είναι κρυπτογραφημένα ή μη

Γενική ιδέα

- Τα δεδομένα κρυπτογραφούνται για να αποθηκευθούν σε χώρο χαμηλής ασφάλειας ή για να μεταδοθούν από ανασφαλές κανάλι



Γενικές παρατηρήσεις

- Η κρυπτογραφία δεν λύνει προβλήματα: απλά τους αλλάζει μορφή
  - Προστασία δεδομένων → Διαχείριση κλειδιών
- Για τη διαφύλαξη ενός μυστικού χρειάζεται ένα άλλο μυστικό

Καταλληλότητα κρυπτογραφίας

- Ιδεώδης για διασφάλιση επικοινωνιακών καναλιών
- Καλή για μακρόχρονα αποθηκευόμενα δεδομένα
- Δεν συνηθίζεται για ηλεκτρονική αλληλογραφία
- Μάλλον ακατάλληλη για ενεργές βάσεις δεδομένων

Συμμετρική κρυπτογραφία

- Το ίδιο κλειδί κρυπτογραφεί και αποκρυπτογραφεί τα δεδομένα
  - Dec(Enc(D, k), k) = D
  - Παράδειγμα:

```
char *encrypt(s, k) {
    char *res = malloc(strlen(s) + 1), *w = res, rk = k;
    while (*s) {
            *(w++) = *(s++) ^ *(rk++);
            if (!*rk) rk = k;
    }
    *w = '\0';
    return res;
}
```

Ασύμμετρη κρυπτογραφία

- *Δύο* κλειδιά, το κλειδί κρυπτογράφησης (δημόσιο κλειδί) και το κλειδί αποκρυπτογράφησης (ιδιωτικό κλειδί)
- Δεν υπάρχει υπολογιστικά εφικτός τρόπος υπολογισμού του ενός κλειδιού από το άλλο

Λειτουργία ασύμμετρης κρυπτογραφίας (1)

(1) Αίτηση για K_{Δ}^{B}

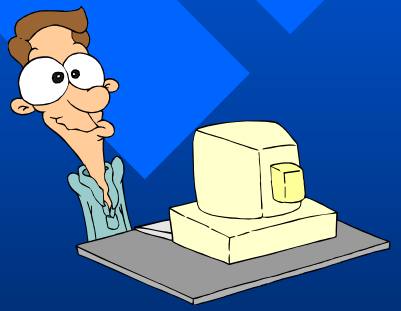
(2) Αποστολή K_{Δ}^{B}

(3) Αποστολή Κωδ(Δ, K_{Δ}^{B})

(4) Αίτηση για K_{Δ}^{A}

(5) Αποστολή K_{Δ}^{A}

(6) Αποστολή Κωδ(Δ', K_{Δ}^{B})



K_{I}^{A}

K_{I}^{B}

Λειτουργία ασύμμετρης κρυπτογραφίας (2)

Λειτουργία ασύμμετρης κρυπτογραφίας (3)

Λειτουργία ασύμμετρης κρυπτογραφίας (4)

Προστασία πόρων

- Μη προστατευμένο δίκτυο
  - Κάθε υπολογιστής στο Internet μπορεί να επικοινωνήσει με κάθε υπολογιστή του εταιρικού δικτύου
  - Απαιτείται η προστασία του κάθε υπολογιστή ξεχωριστά
  - Αν *ένας* υπολογιστής είναι ευάλωτος, η ασφάλεια είναι προβληματική
  - Μη διαχειρίσιμο σύστημα

Προστασία πόρων (2)

Προστατεύοντας το δίκτυο

- Εισαγωγή firewalls
  - Συστήματα ταυτοποίησης-εξουσιοδότησης που τοποθετούνται στα όρια του προστατευόμενου δικτύου
    - » Αποτελεσματικός έλεγχος πρόσβασης
    - » Φιλτράρισμα των ευάλωτων πρωτοκόλλων
    - » Παρακολούθηση της λειτουργίας του δικτύου
    - » Απλοποιημένη διαχείριση

Firewalls

- Σκοπός: η προστασία του δικτύου που βρίσκεται πίσω από το firewall
- Έλεγχος δικτυακής κυκλοφορίας και περιορισμός των επιτρεπόμενων ροών
- Συστατικά:
  - Δρομολογητές με φιλτράρισμα πακέτων
  - Υπολογιστές με ρόλο αντιπροσώπευσης υπηρεσιών
  - Υπολογιστές με ρόλο *προμαχώνα*

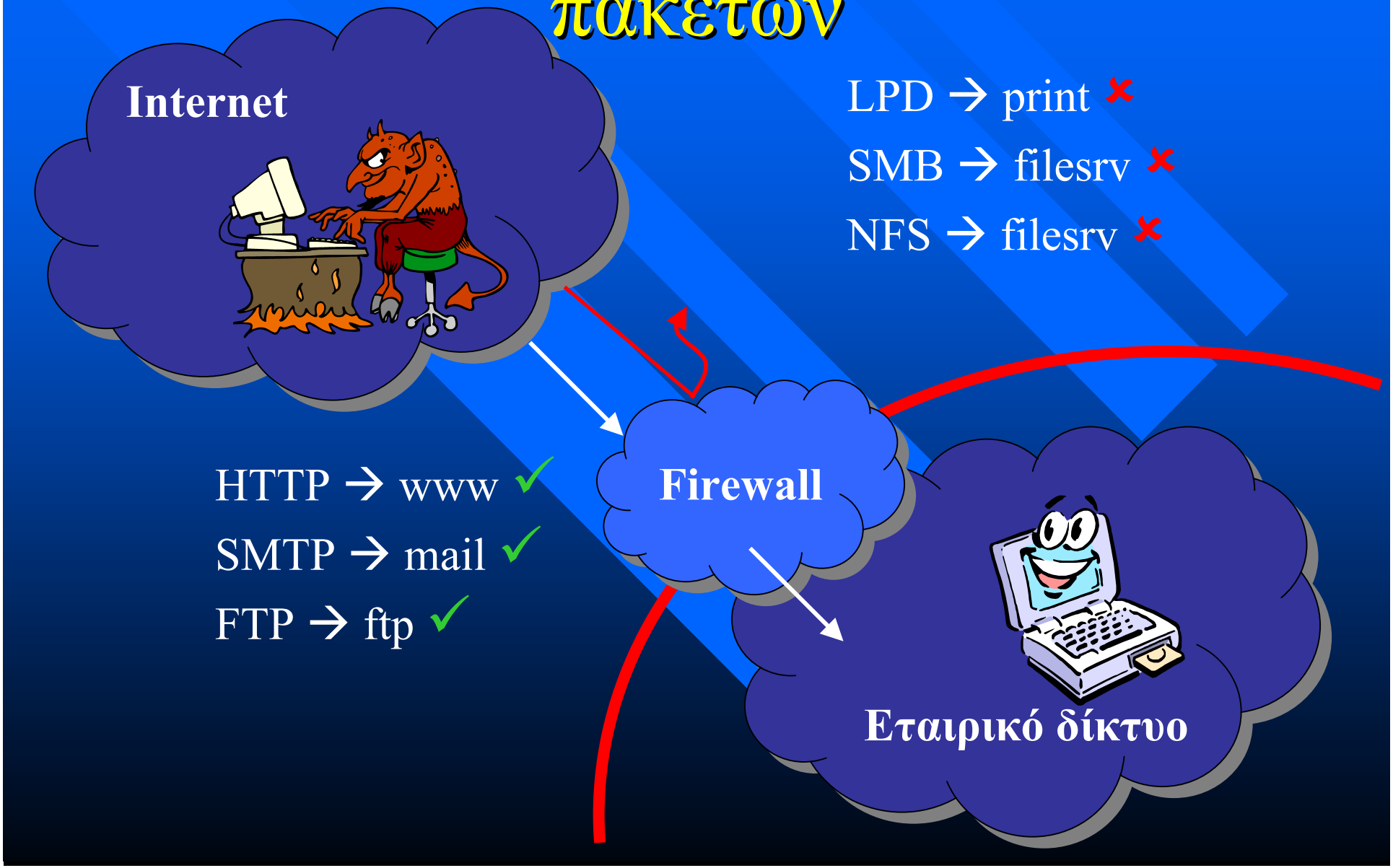
Γενικές αρχές για Firewalls

- Όλη η δικτυακή κυκλοφορία διέρχεται μέσω του firewall
- Μόνο η κυκλοφορία που επιτρέπεται από την πολιτική ασφάλειας διεκπεραιώνεται τελικά
- *Το ίδιο το firewall είναι άτρωτο*
- Δύο σχεδιασμοί:
  - επιτρέπονται όσα δεν απαγορεύονται
  - απαγορεύονται όσα δεν επιτρέπονται

Τα Firewalls ΔΕΝ είναι πανάκεια

- Δεν αντιμετωπίζουν πλήρως τα ζητήματα του περιεχομένου
- Δεν παρέχουν καμία προστασία έναντι επιθέσεων εκ των έσω ή σε κανάλια διαρροής
- Είναι δυνατόν να αποτελέσουν σημεία συμφόρησης
- Αν καταρρεύσουν, το δίκτυο είναι εκτεθειμένο

Firewalls – Φιλτράρισμα πακέτων

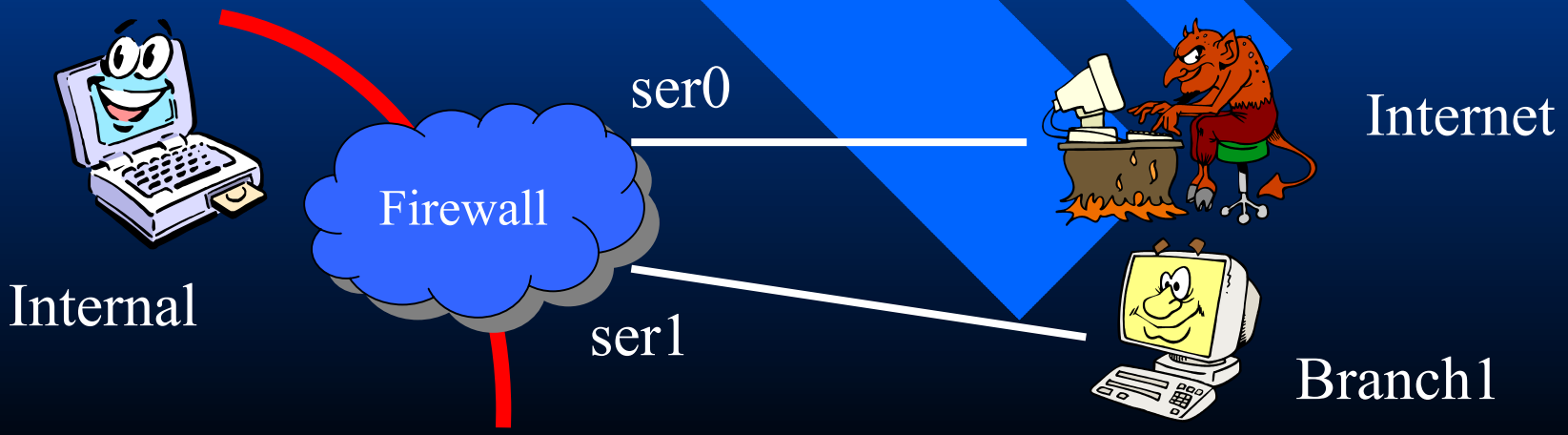


Φιλτράρισμα πακέτων – Οι κανόνες

- Φυσική συσκευή λήψης του πακέτου
- Διεύθυνση αφετηρίας (IP, Port)
- Διεύθυνση προορισμού (IP, Port)
- Πρωτόκολλο ανώτερου επιπέδου (TCP/UDP)
- Δικτυακές ενδείξεις (π.χ. αίτηση εγκαθίδρυσης σύνδεσης)
- Τελική απόφαση (επιτρέπεται, απαγορεύεται)

Παράδειγμα κανόνων

Phydev	Src	Dest	Proto	Opt	Action
ser0	branch1/*	*/*	*	*	Deny
*	*/*	mailsrv/25	TCP	*	Allow
*	*/*	nsrv/143	TCP UDP	*	Allow
ser1	branch1/*	orasrv/1525	TCP	*	Allow



Φιλτράρισμα πακέτων

Υπέρ	Κατά
✓ Απλό στην υλοποίηση ✓ Εξαιρετικές επιδόσεις	✗ Απουσία καταγραφής ✗ Δύσκολο στη ρύθμιση ✗ Απουσία ευελιξίας και επεκτασιμότητας ✗ Μπορεί να παρακαμφθεί μέσω «tunneling»

Δυναμικό φιλτράρισμα πακέτων

- Παράλληλα με τους κανόνες πρόσβασης, εξετάζουμε τους κανόνες του πρωτοκόλλου
  - Όχι πακέτα δεδομένων χωρίς εγκαθίδρυση σύνδεσης
  - Όχι πακέτα δεδομένων μετά την καταστροφή της σύνδεσης
  - Όχι απαντήσεις σε ερωτήσεις που δεν έγιναν
- Εξαιρετικός υποψήφιος για επιθέσεις με στόχο την εξάντληση των πόρων

Αρχιτεκτονική με φιλτράρισμα πακέτων

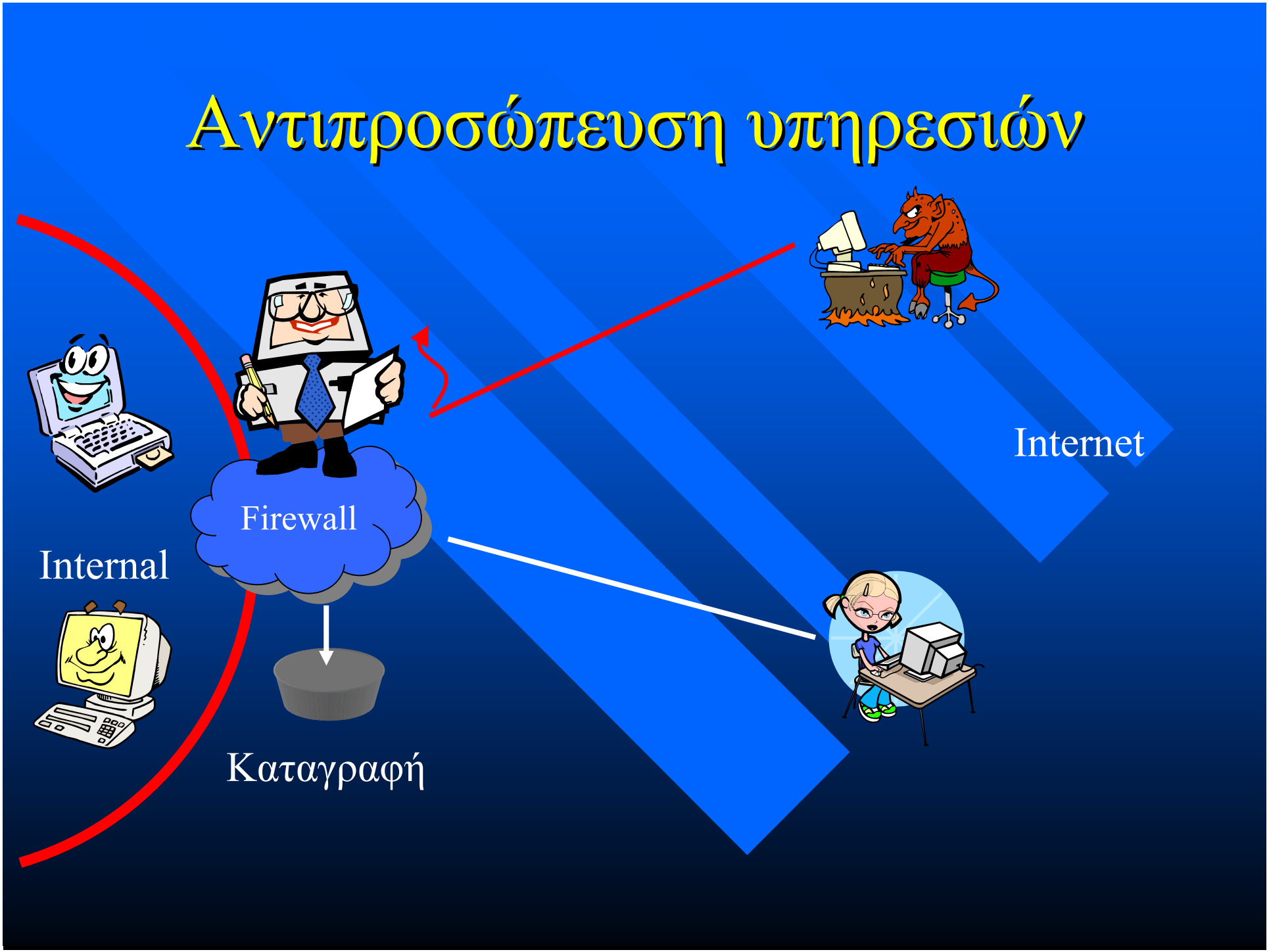
Παράδειγμα κανόνων

```
iptables -s 10.30.19.0/24 –i ! eth0 –j DENY –l
iptables -s 195.134.65.128/26 –i ! eth2 –j DENY –l

iptables -A forward -s 10.30.19.0/24 -i eth0 –o eth2 -j good-dmz
iptables -A forward -s 10.30.19.0/24 -i eth0 –o eth1 -j good-bad
iptables -A forward -s 195.134.65.128/26 -i eth2 –o eth1 -j dmz-bad
iptables -A forward -s 195.134.65.128/26 -i eth2 –o eth0 -j dmz-good
iptables -A forward -i eth0 -j bad-dmz
iptables -A forward -i eth2 -j bad-good
iptables -A forward -j DENY -l

iptables -A good-dmz -p tcp -d 195.134.65.183 smtp -j ACCEPT
iptables -A good-dmz -p tcp -d 195.134.65.183 pop3 -j ACCEPT
iptables -A good-dmz -j DENY -l
```

Παράδειγμα κανόνων - συνέχεια

```
iptables -A dmz-good -p tcp ! -y -s 195.134.65.183 smtp -j ACCEPT
iptables -A dmz-good -p tcp ! -y -s 195.134.65.183 pop3 -j ACCEPT
iptables -A dmz-good -j DENY –l

iptables -A bad-dmz -p tcp -d 195.134.65.183 smtp -j ACCEPT
iptables -A bad-dmz -p tcp ! -y --sport 25 -d 195.134.65.183 -j
   ACCEPT
iptables -A bad-dmz -j DENY

iptables -A dmz-bad -p tcp -s 195.134.65.183 ––dport smtp -j
   ACCEPT
iptables -A dmz-bad -p tcp ! –y -s 195.134.65.183 smtp -j ACCEPT
iptables -A dmz-bad -j DENY –l

iptables -A good-bad -p tcp --dport www -j MASQ
iptables –A good-bad -j DENY –l

iptables –A bad-good -j DENY
```

Αντιπροσώπευση υπηρεσιών



Ζητήματα αντιπροσώπευσης υπηρεσιών

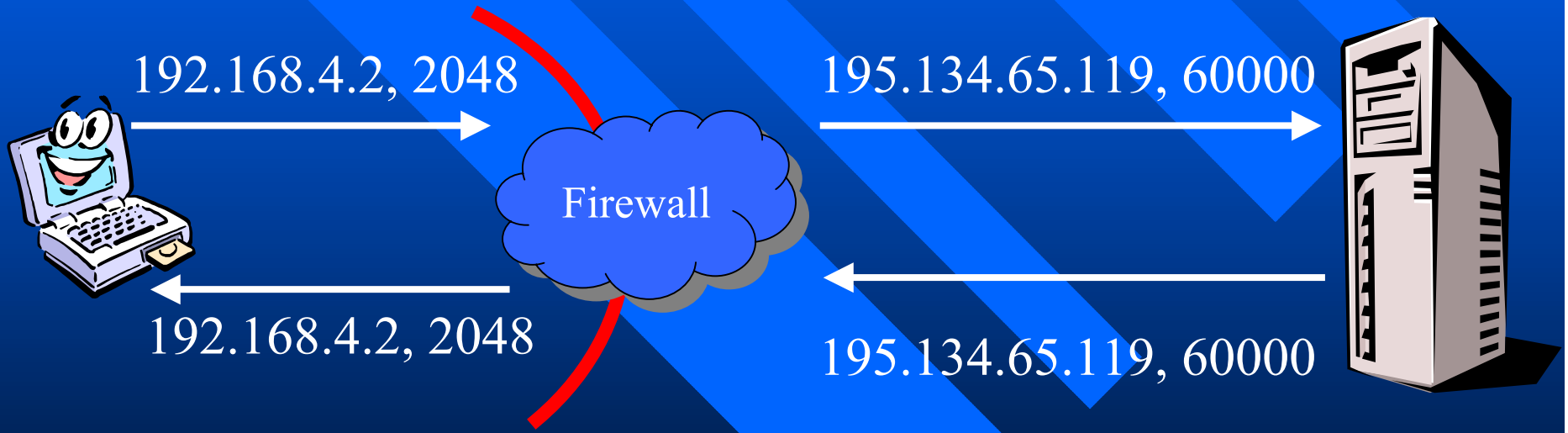
Υπέρ	Κατά
✓ Απλό στην υλοποίηση, χαμηλού κόστους ✓ Δυνατότητα καταγραφής υπόπτων αιτήσεων ✓ Το εσωτερικό δίκτυο είναι αόρατο	✗ Πιθανή εισαγωγή σημείων συμφόρησης ✗ Ευελιξία ✗ Μερικές υπηρεσίες δεν είναι εύκολο να αντιπροσωπευθούν ✗ Κάθε υπηρεσία είναι «διπλή»

Φιλτράρισμα με αντιπροσώπευση

- Φιλτράρισμα ώστε να επιτρέπεται η σύνδεση από το Internet μόνο προς τον εξυπηρέτη αντιπροσώπευσης
- Ο εξυπηρέτης αντιπροσώπευσης προωθεί τις αιτήσεις στους «πραγματικούς» εξυπηρέτες
  - Σε μερικές περιπτώσεις μπορεί να επιτρέπεται απευθείας πρόσβαση στους εσωτερικούς εξυπηρέτες (απόδοση, «ακίνδυνες» υπηρεσίες)

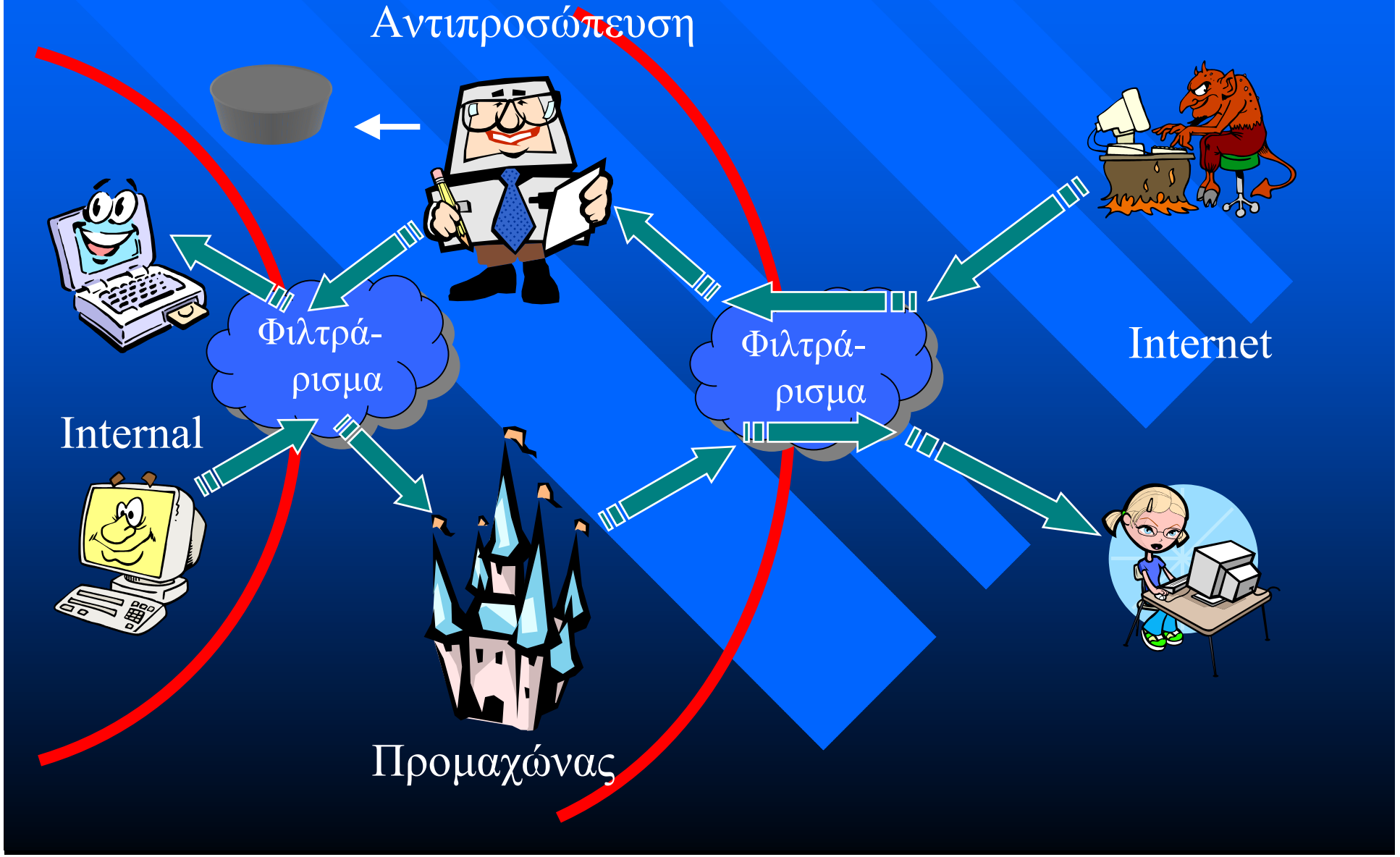
Πρόσβαση από εσωτερικούς χρήστες σε εξωτερικές υπηρεσίες

- Απ' ευθείας πρόσβαση
- Με *μετάφραση διευθύνσεων δικτύου*



- Με χρήση αντιπροσώπευσης
- Με *υπολογιστές-προμαχώνες*

Γενική αρχιτεκτονική firewalls



«Περιτυλίγματα» Υπηρεσιών

- *Περιτύλιγμα:* ένα πρόγραμμα που χρησιμοποιείται για να ελέγχει την πρόσβαση σε ένα άλλο πρόγραμμα
- Συνήθως απλό και εύκολο να ελεγχθεί ως προς την ορθότητά του
- Αναπτύσσεται-συντηρείται ανεξάρτητα από το ελεγχόμενο πρόγραμμα
- Ένα «περιτύλιγμα» ελέγχει την πρόσβαση σε πολλά προγράμματα

«Περιτυλίγματα»: Στόχοι

- Καταγραφή συνδέσεων
- Διενέργεια ελέγχων
  - Διεύθυνση σύνδεσης
  - Ονοματολογία συνδεόμενου υπολογιστή
  - Εξέταση της ταυτότητας του συνδεόμενου χρήστη
- Π.χ. tcpd
  - Καλείται *πριν* από τη σχετική υπηρεσία, μέσω διαμόρφωσης του αρχείου καθορισμού εκκίνησης προγραμμάτων εξυπηρέτησης inetd.conf
  - Αφού διενεργήσει τους ελέγχους επιτυχώς, παραδίδει τον έλεγχο στο πρόγραμμα παροχής υπηρεσίας
  - Αν οι έλεγχοι αποτύχουν, υπάρχει η δυνατότητα

Περιτυλίγματα υπηρεσιών - Παράδειγμα

- hosts.allow

```
imapd: .mycom.com .affliatecom.com LOCAL
in.telnetd: .mycom.com .affliatecom.com
in.telnetd: 195.170.21.138
sshd: 60.70.80. 10.
ALL: .mycom.com .affliatecom.com
```

- hosts.deny

```
ALL: UNKNOWN: (/usr/sbin/safe_finger -l @%a | \
                /usr/ucb/mail -s %H-%d-%h root)&
ALL: 143.233.160.99: (/usr/sbin/safe_finger -l @%a | \
                /usr/ucb/mail -s %H-%d-%h root)&
ALL: PARANOID
ALL: .hackers.org .rivals.com 195.134.79.73
imapd: ALL
```