Ασφαλής προγραμματισμός σε γλώσσα C

Πάτροκλος Αργυρούδης Δημήτρης Γλυνός

census
IT security research, development and services
{argp, dimitris}@census.gr

Συνέδριο Δημιουργών ΕΛ/ΛΑΚ
19-20 Ιουνίου 2009

Εισαγωγή

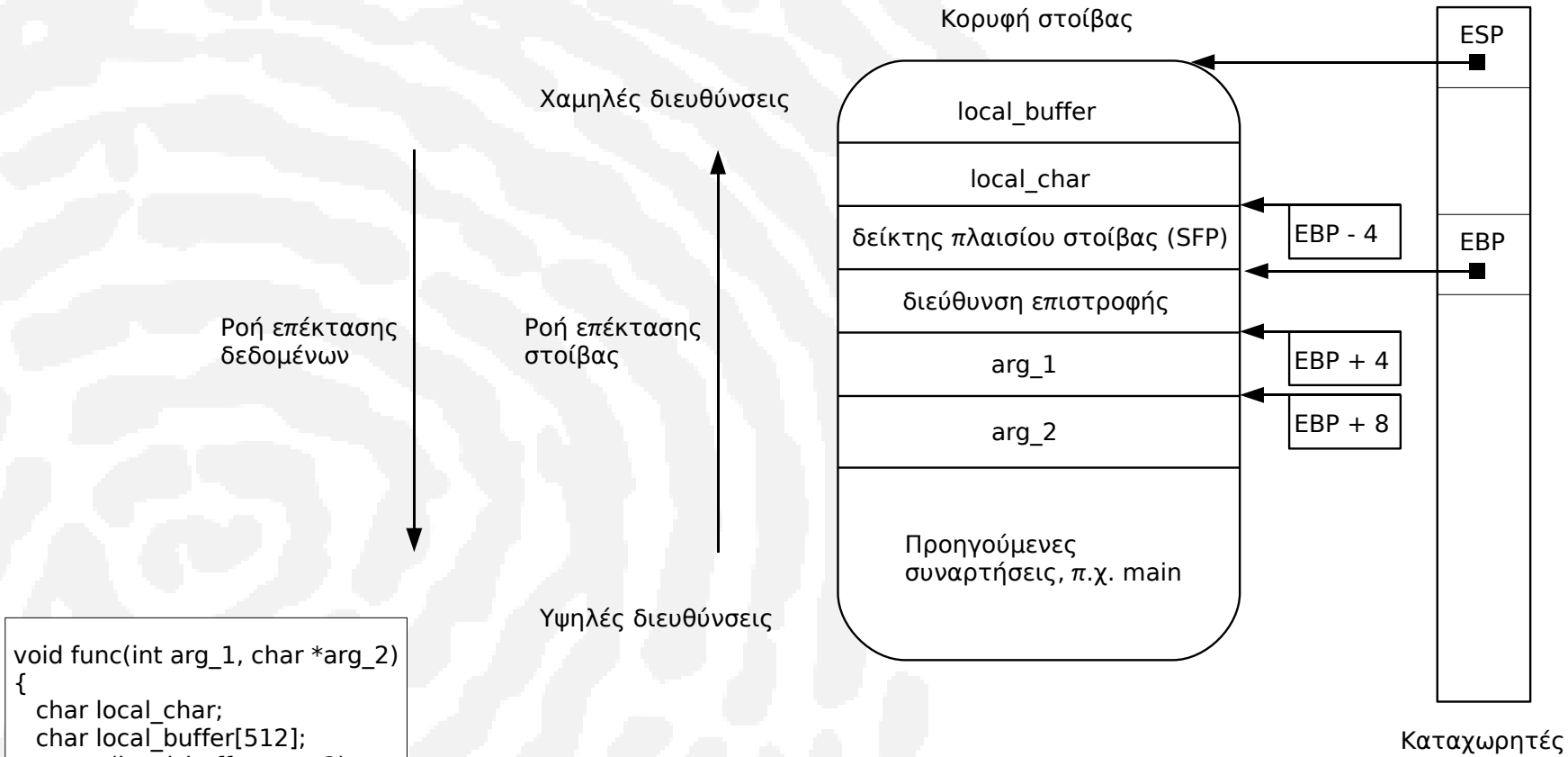
Κενά ασφάλειας στη γλώσσα C

- Η γλώσσα C έχει σχεδιαστεί με βάση τη φιλοσοφία της παροχής ελευθερίας στον προγραμματιστή κατά τη διαδικασία ανάπτυξης λογισμικού
  - Αυθαίρετες μετατροπές μεταξύ τύπων δεδομένων
  - Πλήρης έλεγχος κατά τη δέσμευση και αποδέσμευση μνήμης
  - Αναφορές σε αυθαίρετες θέσεις μνήμης
  - Χρήση αυθαίρετων δεικτών σε πίνακες
  - Εκτέλεση κώδικα από αυθαίρετες θέσεις μνήμης
- Η λανθασμένη χρήση αυτής της ελευθερίας οδηγεί συχνά σε προγραμματιστικά σφάλματα
  - Από τα οποία προκύπτουν κενά ασφάλειας

Κατηγορίες σφαλμάτων

Υπερχείλιση μνήμης στη στοίβα

Πολύ γνωστό και κατανοητό πρόβλημα, υπερχείλιση τοπικών μεταβλητών από αντιγραφή δεδομένων χωρίς έλεγχο ορίων




```
void func(int arg_1, char *arg_2)
{
  char local_char;
  char local_buffer[512];
  strcpy(local_buffer, arg_2);
}
```

Υπερχείλιση μνήμης στο σωρό

- Επικάλυψη δυναμικών δομών
  > Κενά ασφάλειας: εξαρτώμενα από την εφαρμογή
  - Δείκτες συναρτήσεων
  - Σημαντικά δεδομένα για τη λογική της εφαρμογής, π.χ. σημαίες ταυτοποίησης (authentication flags)
- Επικάλυψη δομών διαχείρισης του σωρού
  > Κενά ασφάλειας: εξαρτώμενα από την υλοποίηση του σωρού
  - glibc: διπλά συνδεδεμένη λίστα για τη διαχείριση της μνήμης που δεσμεύει (και αποδεσμεύει) ένα πρόγραμμα
  - Αλλοίωση των δομών της λίστας οδηγεί σε αυθαίρετη επικάλυψη μνήμης κατά τη διαγραφή κόμβων από τη λίστα

Παρά-ένα λάθη (off-by-one)

- Λάθη κατά τα οποία ο υπολογισμός μεγέθους μίας περιοχής μνήμης είναι ανακριβής για ένα byte
- Συνήθεις αιτίες
  - Τα strings στη C απαιτούν ένα byte επιπλέον για να τερματιστούν
  - Παρανόηση του τρόπου με τον οποίο λειτουργούν οι πίνακες
- Παρουσιάζονται στη στοίβα και στο σωρό
- Η εκμετάλλευσή τους (σε i386 και στη στοίβα) εξαρτάται από το αν η προβληματική περιοχή μνήμης βρίσκεται ακριβώς μετά από τον αποθηκευμένο δείκτη πλαισίου (SFP)

Παράλειψη προσδιοριστή μορφής (format string)

- Παράλειψη του προσδιοριστή μορφής ``%s'' σε συναρτήσεις της οικογένειας printf(3) (αλλά και σε άλλες, όπως π.χ. syslog(3))
- Η προβληματική συνάρτηση αναζητά τους παραλειπόμενους προσδιοριστές στη στοίβα
- Επιτρέπει την αυθαίρετη διαμόρφωση του προσδιοριστή μορφής
- Μπορεί να έχει ως αποτέλεσμα
  - τη διαρροή δεδομένων από τη στοίβα (διευθύνσεις, τιμές μεταβλητών, κτλ),
  - την αυθαίρετη επικάλυψη μνήμης

Υπερχείλιση ακέραιων μεταβλητών

- Οι ακέραιες μεταβλητές έχουν τη δυνατότητα να αποθηκεύσουν τιμές συγκεκριμένου εύρους
  - -INT_MAX-1 <= int <= INT_MAX, INT_MAX == 2147483647
  - 0 <= unsigned int <= UINT_MAX, UINT_MAX == 4294967295
- Αν ξεπεραστεί η μέγιστη τιμή, π.χ. ως αποτέλεσμα κάποιας πρόσθεσης ή πολλαπλασιασμού, έχουμε αναδίπλωση
- Γενικά δεν είναι εκμεταλλεύσιμο σφάλμα, δεν υπάρχει άμεση επικάλυψη μνήμης από την υπερχείλιση
- Συχνά όμως ο προβληματικός ακέραιος χρησιμοποιείται ως θέση στοιχείου σε πίνακα ή κατά τη δυναμική δέσμευση μνήμης
  - Υπερχείλιση ή/και αυθαίρετη επικάλυψη μνήμης

Σφάλματα προσήμου (signedness)

- Στα σφάλματα προσήμου μία μεταβλητή δίχως πρόσημο (unsigned) ερμηνεύεται σαν να έχει πρόσημο (signed)
  - Και το αντίστροφο
- Παρουσιάζονται σε περιπτώσεις όπου ακέραιοι με πρόσημο
  - χρησιμοποιούνται σε συγκρίσεις,
  - χρησιμοποιούνται σε αριθμητικές πράξεις,
  - συγκρίνονται με ακέραιους δίχως πρόσημο,
  - υπερχειλίζουν, αναδιπλώνονται και αλλάζουν πρόσημο
- Η εκμετάλλευσή τους από κακόβουλους χρήστες δεν είναι πάντα απλή
  - Η τιμή -1 όταν ερμηνευτεί δίχως πρόσημο είναι 4294967295 (~4GB)

Άλλες κατηγορίες σφαλμάτων

- Διαρροή ευαίσθητων δεδομένων
  - Συνθηματικών, κλειδιών κρυπτογράφησης, εσωτερικών δομών και διαδικασιών της εφαρμογής
  - Μπορεί να επιτρέψει την παρά πέρα εκμετάλλευση σφαλμάτων
- Συνθήκες ανταγωνισμού
  - Η υλοποίηση μίας εννοιολογικά αδιαίρετης διαδικασίας διασπάται σε τμήματα επιτρέποντας εξωτερικές παρεμβάσεις
  - Κάποιο αντικείμενο του συστήματος (π.χ. πόροι, προνόμια) αλλάζει κατάσταση μεταξύ του ελέγχου και της χρήσης του
  - Προσωρινά αρχεία

Επικίνδυνες συναρτήσεις

Επικίνδυνες συναρτήσεις της βασικής βιβλιοθήκης

- gets(3), scanf(3), sprintf(3), strcpy(3), strcat(3), ...
- char *strncpy(char *dest, const char *src, size_t n);
  - Προτρέπει σε παρά-ένα λάθη τον προγραμματιστή που δεν έχει υπολογίσει προσεκτικά τη τιμή της παραμέτρου n
  - Δεν τερματίζει αυτόματα το dest με NULL με αποτέλεσμα να έχει διαφορετικό μέγεθος από αυτό που πιστεύει ο προγραμματιστης
- Λανθασμένη χρήση του τελεστή sizeof
  - buffer[sizeof(buffer)] = NULL;
  - if(strlen(str_input) > sizeof(buffer))

Σφάλματα σε σύγχρονο λογισμικό

Mac OS X 10.3.9: bsd/kern/sysv_sem.c

CVE-2005-0971: Σφάλμα προσήμου στη semop(2)

```
int
semop(struct proc *p, struct semop_args *uap, register_t *retval)
{
        int semid = uap->semid;
        int nsops = uap->nsops;
        struct sembuf sops[MAX_SOPS];
        . . .
        if (nsops > MAX_SOPS) {
        . . .
        if ((eval = copyin(uap->sops, &sops,
                nsops * sizeof(struct sembuf))) != 0) {
        . . .
```

Διόρθωση (patch)

```
-       if (nsops > MAX_SOPS) {
+        if (nsops < 0 || nsops > MAX_SOPS) {
```

FreeBSD 7.1: kern/kern_time.c

CVE-2009-1041: Σφάλμα προσήμου στην itimer_find()

```
static struct itimer *
itimer_find(struct proc *p, int timerid)
{
        struct itimer *it;
        . . .
        if ((p->p_itimers == NULL) || (timerid >= TIMER_MAX) ||
            (it = p->p_itimers->its_timers[timerid]) == NULL) {
                return (NULL);
         }
        . . .
```

Διόρθωση (patch)

```
-       if ((p->p_itimers == NULL) || (timerid >= TIMER_MAX) ||
+       if ((p->p_itimers == NULL) ||
+           (timerid < 0) || (timerid >= TIMER_MAX) ||
```

WU-FTPD 2.6.2: realpath.c

CVE-2003-0466: Παρά-ένα λάθος στην fb_realpath()

```
char *
fb_realpath(const char *path, char *resolved)
{
        struct stat sb;
        int fd, n, rootd, serrno;
        char *p, *q, wbuf[MAXPATHLEN];
        . . .
        if (resolved[0] == '/' && resolved[1] == NULL)
                rootd = 1;
        else
                rootd = 0;
        if (*wbuf) {
                if (strlen(resolved) + strlen(wbuf) + rootd + 1 > MAXPATHLEN) {
                        errno = ENAMETOOLONG;
                        . . .
                }
                if (rootd == 0)
                        (void) strcat(resolved, "/");
                (void) strcat(resolved, wbuf);
        }
```

Διόρθωση (patch)

```
-           if (strlen(resolved) + strlen(wbuf) + rootd + 1 > MAXPATHLEN) {
+           if (strlen(resolved) + strlen(wbuf) + !rootd + 1 > MAXPATHLEN) {
```

OpenBSD 2.8: libexec/ftpd/ftpd.c

CVE-2001-0053: Παρά-ένα λάθος στη replydirname()

```
replydirname(const char *name, const char *message)
{
        . . .
        char npath[MAXPATHLEN];
        int i;
        . . .
        for (i = 0; *name != NULL && i < sizeof(npath) - 1;
             i++, name++) {
              npath[i] = *name;
              if (*name == '"')
                    npath[++i] = '"';
        }
        npath[i] = NULL;
```

Διόρθωση (patch)

```
+       p = npath;
+       ep = &npath[sizeof(npath) - 1];
```

OpenSolaris 10: uts/common/os/aio.c

CVE-2009-0132: Υπερχείλιση ακέραιας μεταβλητής στη aiosuspend()

```
aiosuspend(void *aiocb, int nent, struct timespec *timout, int flag, . . .
{
        . . .
        aiop = curproc->p_aio;
        if (aiop == NULL || nent <= 0)
                return (EINVAL);
        . . .
        if (model == DATAMODEL_NATIVE)
                ssize = (sizeof (aiocb_t *) * nent);
#ifdef _SYSCALL32_IMPL
        else
                ssize = (sizeof (caddr32_t) * nent);
#endif /* _SYSCALL32_IMPL */
        cbplist = kmem_alloc(ssize, KM_NOSLEEP);
```

Διόρθωση (patch)

```
-           if (aiop == NULL || nent <= 0)
+            if (aiop == NULL || nent <= 0 || nent > _AIO_LISTIO_MAX)
```

FreeBSD 6.0: net80211/ieee80211_ioctl.c

CVE-2006-0380: Διαρροή δεδομένων στην ieee80211_ioctl_getchanlist()

```
static int
ieee80211_ioctl_getchanlist(struct ieee80211com *ic,
        struct ieee80211req *ireq)
{
        if (sizeof(ic->ic_chan_active) > ireq->i_len)
                ireq->i_len = sizeof(ic->ic_chan_active);
        return copyout(&ic->ic_chan_active, ireq->i_data, ireq->i_len);
}
```

Διόρθωση (patch)

```
-       if (sizeof(ic->ic_chan_active) > ireq->i_len)
+        if (sizeof(ic->ic_chan_active) < ireq->i_len)
```

Θωράκιση λογισμικού

Κατά τη μεταγλώττιση (1)

Χρήση του GNU C Compiler (gcc), της βιβλιοθήκης GNU libc και του GNU linker (ld)

- gcc -D_FORTIFY_SOURCE=2 -O2
  - Ανίχνευση (απλών) σφαλμάτων υπερχείλισης μνήμης
  - Αντικατάσταση συναρτήσεων από εναλλακτικές με ελέγχους (π.χ. sprintf => __sprintf_chk)
- gcc -Wformat=2
  - Έλεγχος προσδιοριστών μορφής (format strings)
- gcc -fstack-protector ή -fstack-protector-all
  - SSP: Προστασία δεδομένων της στοίβας με canary
  - Αλλαγή των θέσεων των τοπικών μεταβλητών στη στοίβα

Κατά τη μεταγλώττιση (2)

- gcc -pie -fpie
  - Δημιουργία εκτελέσιμων που ο κώδικας τους μπορεί να φορτωθεί σε τυχαία διεύθυνση μνήμης (position independent executables)
- ld -z relro -z now ή gcc -Wl,-z,relro,-z,now
  - Προστασία πεδίων GOT και PLT σε ELF εκτελέσιμα

Κατά την εκτέλεση (1)

- Address Space Layout Randomisation
  - Τυχαιοποίηση των διευθύνσεων των σελίδων μνήμης:
    - της στοίβας
    - του σωρού
    - αυτών που δεσμεύονται με mmap(2) (βιβλιοθήκες, malloc(3), κοινή μνήμη κ.α.)
    - του κώδικα μιας εφαρμογής (σε position independent executables)
  - Τυχαιοποίηση της κορυφής της στοίβας (top of stack)
  - Στο Linux η τυχαιοποίηση παραμετροποιείται μέσω του αρχείου /proc/sys/kernel/randomize_va_space ως εξής:
    - 2: τυχαιοποίηση παντού (stack, top of stack, heap, mmap, brk)
    - 1: τυχαιοποίηση παντού εκτός από τη διεύθυνση του σωρού (brk)
    - 0: καμμία τυχαιοποίηση

Κατά την εκτέλεση (2)

- Μη εκτελέσιμες σελίδες μνήμης
  - Υλοποίηση στο hardware
    - AMD NX (No eXecute) bit
    - Intel XD (eXecute Disable) bit
  - Υλοποίηση με λογισμικό
    - ExecShield
    - W^X
- Θωράκιση του περιβάλλοντος εκτέλεσης της εφαρμογής
  - παροχή τυχαίων canaries από το loader και τη libc
  - πολιτική μεταγλώττισης εφαρμογών και βιβλιοθηκών του λειτ. συστήματος με γνώμονα την ασφάλεια
  - περιορισμός του περιβάλλοντος εκτέλεσης της εφαρμογής (Solaris zones, BSD jails, Linux containers κλπ.)
  - θωράκιση του πυρήνα (CONFIG_CC_STACKPROTECTOR=Y στο Linux)

Συμπεράσματα

- Η ανάπτυξη λογισμικού σε C απαιτεί μεγάλη προσοχή και καλή γνώση των ``σκοτεινών'' σημείων εξαιτίας των ελευθεριών που παρέχει η γλώσσα
- Προσοχή στο input/output, στη διαχείριση προσωρινής μνήμης και στις πράξεις με ακέραιους
- Αξιοποίηση των βιβλιοθηκών δυναμικής σύνδεσης
  - Στατικά συνδεδεμένα προγράμματα μένουν απροστάτευτα κατά τις αναβαθμίσεις του συστήματος
- Χρήση θωρακισμένου λογισμικού
- Τακτικός έλεγχος πηγαίου κώδικα από τρίτους

Παραπομπές

Phantasmal Phantasmagoria

The malloc maleficarum: glibc malloc exploitation techniques
MallocMaleficarum.txt, 2005

klog

The frame pointer overwrite
Phrack, Volume 9, Issue 55, 1999

blexim

Basic integer overflows
Phrack, Volume 0x0b, Issue 0x3c, 2002

David A. Wheeler

Secure Programming for Linux and Unix HOWTO
Secure-Programs-HOWTO, v3.010, 2003

Mark Dowd, John McDonald and Justin Schuh

The art of software security assessment
Addison-Wesley, 2006

Debian Wiki

Debian hardening options
http://wiki.debian.org/Hardening, 2009