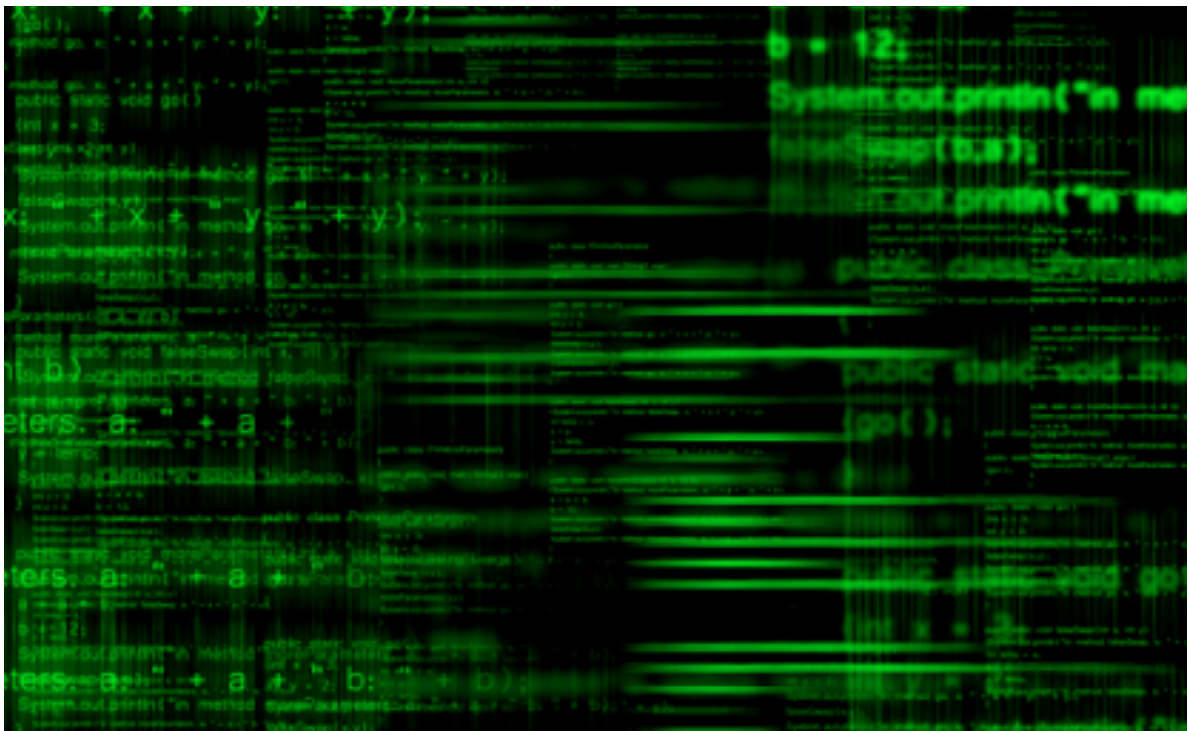


Οι επιθέσεις από κακόβουλους χρήστες είναι πολλών ειδών. Πολλές φορές εκμεταλλεύονται ευπάθειες των συστημάτων ή των προγραμμάτων που χρησιμοποιούνται σε ιστοσελίδες και εφαρμογές. Μια απ' αυτές είναι ή επίθεση υπερχείλισης ή γνωστή στα αγγλικά με τον όρο : Buffer Overflow όπου χρήστες εκμεταλλεύονται συστήματα, φόρμες, ιστοσελίδες και εφαρμογές οι οποίες δεν πραγματοποιούν έλεγχο όριων στις εκάστοτε μεταβλητές ή ακόμη και οι συναρτήσεις δεν χρησιμοποιούνται με τρόπο ορθό και προνοητικό ούτως ώστε να αποφεύγονται προβλήματα που προκύπτουν από υπερχείλιση καταχωρητή. Σε αυτή τη φάση αν ο κακόβουλος χρήστης εισάγει παραπάνω από τους επιτρεπτούς χαρακτήρες οι οποίοι μπορούν να χωρέσουν σε μια μεταβλητή. Αυτό έχει ως αποτέλεσμα την καταστροφή δυνητικά του προγράμματος μας, ή των μεταβλητών που δεσμεύονται για άλλη διεργασία. Ένας άλλος τρόπος επίθεσης είναι και η : SQL injection μέσα από την οποία ο κακόβουλος χρήστης εκμεταλλεύεται σχεδιαστικές αδυναμίες του συστήματος και μπορεί μέσα από κατάλληλες εντολές στην γλώσσα SQL να καταστρέψει, να τροποποιήσει, εισβάλλει στο σύστημα.

- **Επίθεση υπερχείλισης καταχωρητή**
(Buffer overflow attack)
- **Επίθεση “έγχυσης” SQL**
(SQL injection attack)



Ερώτημα 1)

a)

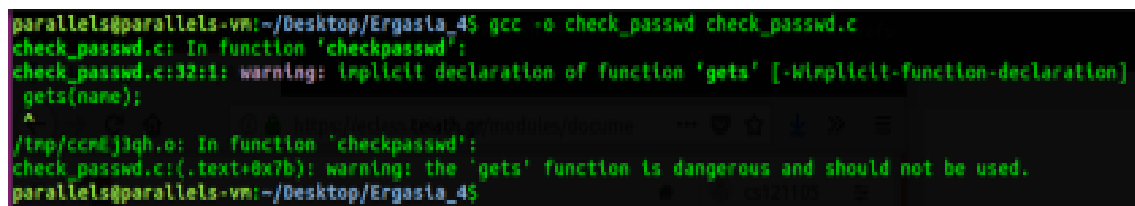
Από τον κώδικα που μας δίνεται βάσει του προγράμματος: **check_passwd.c**

Παρατηρούμε αμέσως πως ο τρόπος που είναι δομημένος κώδικας παρουσιάζει ευπάθεια καθώς η συνάρτηση : `gets(name)` & `gets(passwd)` δεν ελέγχει πόσοι χαρακτήρες θα εισαχθούν – αποθηκευτούν στην μεταβλητή.

```
while ((attempts!=3)&&(correct==0))
{
    printf("Enter login:");
    gets(name);
    printf("Enter password:");
    gets(passwd);
```

Αν ο χρήστης βάλει παραπάνω χαρακτήρες τότε πιθανόν οι περισσότεροι χαρακτήρες θα καταλάμβαναν θέσεις μνήμης στις διπλανές θέσεις πράγμα το οποίο μπορεί να οδηγήσει στο να καταστρέψει δυνητικά μεταβλητές του προγράμματος μας.

Κάνοντας : `compile` το πρόγραμμα παίρνουμε από το σύστημα `warning` το οποίο μας λέει πως το κάθε παρακάτω πρόγραμμα χρησιμοποιεί συναντήσεις `gets` οι οποίες χαρακτηρίζονται ως μη ασφαλείς συναντήσεις.



```
parallels@parallels-vm:~/Desktop/Ergasia_4$ gcc -o check_passwd check_passwd.c
check_passwd.c: In function 'checkpasswd':
check_passwd.c:32:1: warning: implicit declaration of function 'gets' [-Wimplicit-function-declaration]
  gets(name);
  ^
/tmp/ccm5j1qh.o: In function 'checkpasswd':
check_passwd.c:(.text+0x7b): warning: the 'gets' function is dangerous and should not be used.
parallels@parallels-vm:~/Desktop/Ergasia_4$
```

Εικόνα 1

Πληκτρολογώντας το : `username` και το : `password` του χρήστη μπαίνουμε κανονικά στο “σύστημά” μας. Το πρόγραμμα σ’ αυτή τη φάση δεν παρουσιάζει κανένα θέμα.

```
parallels@parallels-vm:~/Desktop/Ergasia_4$ ./check_passwd
Enter login:admin
Enter password:secret
Password correct - Login approved
parallels@parallels-vm:~/Desktop/Ergasia_4$
```

Εικόνα 2

Αν δώσουμε όμως παραπάνω χαρακτήρες θα δούμε ότι αυτή η παραπανίσιοι χαρακτήρες προκαλούν Buffer Overflow. Στην εικόνα παρακάτω παρουσιάζονται οι προσπάθειες που γίνονται με παραπάνω χαρακτήρες καθώς και το μήνυμα που το σύστημα εμφανίζει στην περίπτωση του Buffer Overflow.

```
parallels@parallels-vm:~/Desktop/Ergasia_4$ ./check_passwd
Enter login:adminstratoradmin
Enter password:secretsecretsecret
Enter login:hoooooooooooooooooooo
Enter password:hoooooooooooooooooooooooooooooooooooo
Enter login:hoooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
Enter password:hoooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
*** stack smashing detected ***: ./check_passwd terminated
Aborted (core dumped)
parallels@parallels-vm:~/Desktop/Ergasia_4$
```

Εικόνα 3

Το σύστημα σ' αυτή τη φάση εμφανίζει μήνυμα : stack smashing detected το οποίο σημαίνει πως έχουμε Buffer Overflow και μπορούμε κατά αυτό τον τρόπο να καταστρέψουμε δυνητικά μεταβλητές του προγράμματος. Το πρόβλημα αυτό διορθώνεται με τη χρήση της συνάρτησης : fgets η οποία παίρνει σαν όρισμα την αρχική μεταβλητή που στην περίπτωσή μας ήταν το : name ή το : password, το μήκος των χαρακτήρων που θα εισαχθούν ως δεύτερο όρισμα της συνάρτησης αυτής, και τέλος το : stdin Κάνοντας κατ' αυτό τον τρόπο έλεγχο ωρίων της μεταβλητής, όπου ο χρήστης δεν μπορεί να εισάγει πάνω από 16 χαρακτήρες καθώς το σύστημα θα εμφανίσει μήνυμα λάθους.

```
printf("Enter login:");
fgets(name,16,stdin);
printf("Enter password:");
```

```
gets(passwd,16,stdin);
```

b) Με την χρήση της συνάρτησης `strcpy(s1,s2)` δεν γίνεται έλεγχος για το αν η συμβολοσειρά `s1` μπορεί πραγματικά να αποθηκευτεί στην δεύτερη συμβολοσειρά `s2`. Στη θέση της προηγούμενης συνάντησης για περισσότερη ασφάλεια μπαίνει η συνάρτηση `strncpy(s1,s2,50)` Η οποία δέχεται μια ακόμη παράμετρο που είναι το μήκος της συμβολοσειράς που θα αποθηκευτεί στην δεύτερη συμβολοσειρά.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main()
5  {
6      char source[] = "username12"; // username12 to source[]
7      char destination[7]; // Destination is 8 bytes
8      strcpy(destination, source); // Copy source to destination
9
10     return 0;
11 }
```

Εικόνα 4

c)

```
while ((c=getchar()) !='\n');
    str[i++]=c;
```

Στο συγκεκριμένο πρόγραμμα, μέσα στην επαναληπτική δομή : `while` η συνάρτηση `getchar()`, παίρνει το αποτέλεσμα και το αποθηκεύει στην μεταβλητή : `c` χωρίς να ελεγχει για το μέγεθος της. Αντί αυτού δίνουμε ένα δεύτερο όρισμα δηλαδή όσο οι χαρακτήρες που δέχεται η μεταβλητή : `c` είναι συγκεκριμένου μεγέθους τότε η μεταβλητή : `c` παίρνει τους χαρακτήρες που διαβάζει η συνάρτηση `getchar()`.

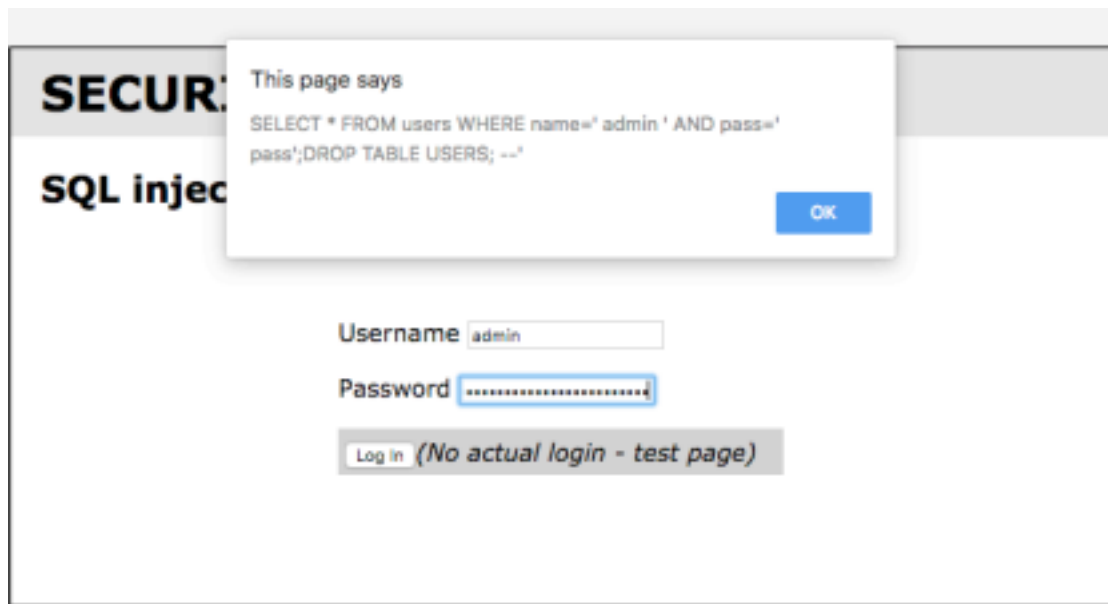
```
while ((i<50)&& (c=getchar()) !='\n'))
    str[i++]=c;
```

Με αυτό τον τρόπο οριοθετούμε το μέγεθος των χαρακτήρων που θα αποθηκευτεί στην μεταβλητή του παραπάνω προγράμματος.

Ερώτημα 2)

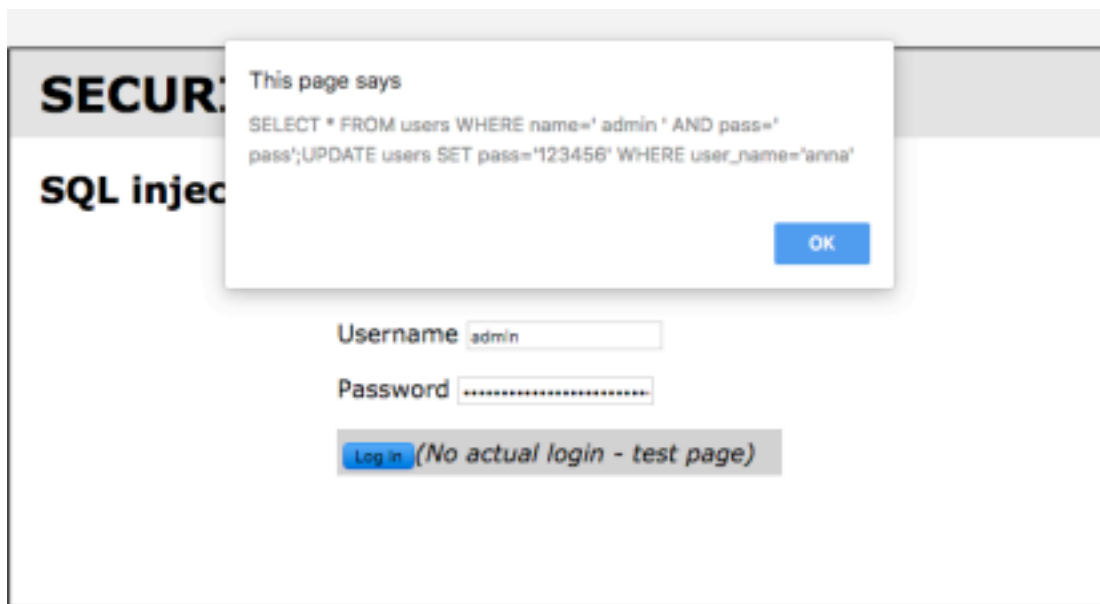
a) Από τη φόρμα, που μας δίνεται στο ερώτημα 4 για να δοκιμάσουμε τα SQL injection κάνουμε της παρακάτω δοκιμές :
Στο πεδίο : Username βάζουμε τη λέξη : admin και στο πεδίο : Password pass'; DROP TABLE users; --

Συμπεραίνουμε πως είναι μια φόρμα ευπαθή στα SQL injection.



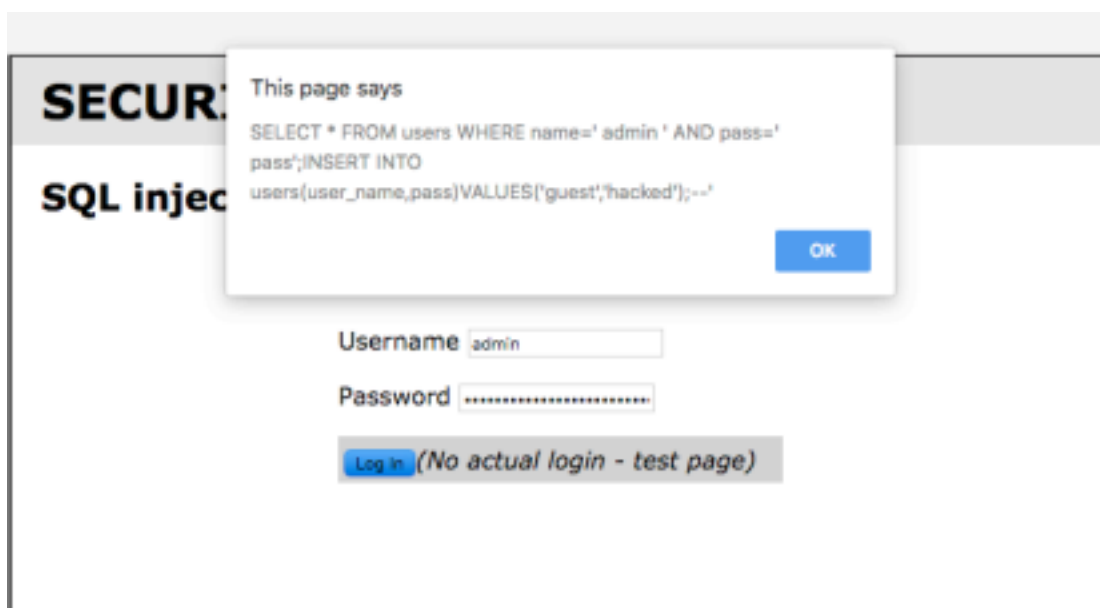
Εικόνα 5

b) Στο πεδίο : Username βάζουμε τη λέξη : admin και στο πεδίο : Password 'pass';UPDATE users SET pass='123456' WHERE user_name='anna'



Εικόνα 6

Στο πεδίο : Username βάζουμε τη λέξη : admin και στο πεδίο : Password pass';INSERT INTO users(user_name,pass)VALUES('guest','hacked');-- c)



Εικόνα 7

Συμπέρασμα : Η συγκεκριμένη φόρμα που μας δόθηκε από την τέταρτη άσκηση εργαστηρίου είναι ευπαθή στα SQL injection. Από τις παραπάνω ενέργειες διαπιστώνουμε πως μπορούμε να διαγράψουμε ολόκληρο πίνακα, να τροποποιήσουμε πίνακας χρηστών προσθέτοντας νέο χρήστη ή τροποποιώντας το password ήδη υπάρχοντα χρήστη κλπ.

Πηγες που χρησιμοποιήθηκαν :

https://www.hackingtutorials.org/exploit-tutorials/buffer-overflow-explained_basics/

Άμπελ Μπάσα

Αθήνα Μάιος 2018