Report for login at 2024-06-09

Summary

Enumeration

Binary Information

File Name	Path	Format	Bit
login	app/testFile/login	ELF	32-bit

Security of the Binary

Basic Security Features			
Linked	**Stripped**	**RELRO**	**Canary**
dynamically linked	no	partial	no

Advanced Security Mechanisms		
NX	**PIE**	**RPath**
no	no	no

Security Meta-Information		
RunPath	**Symbols**	**Fortify Source**
no	yes	no

Strings

- Enter admin password:
- pass
- Correct Password!
- Incorrect Password!
- Successfully logged in as Admin (authorised=%d) :)
- Failed to log in as Admin (authorised=%d) :(
- login.c
- .note.gnu.build-id

Vulnerable Functions

- gets
- printf

Libraries

- linux-gate.so.1
- libc.so.6
- /lib/ld-linux.so.2

Assembly Code

```
xor ebp, ebp
pop esi
mov ecx, esp
and esp, 0xfffffff0
push eax
push esp
push edx
call 0x80490b3
add ebx, 0x2f70
lea eax, [ebx - 0x2d40]
push eax
lea eax, [ebx - 0x2da0]
push eax
push ecx
push esi
mov eax, 0x8049192
push eax
call 0x8049070
hlt
mov ebx, dword ptr [esp]
ret
nop
nop
nop
nop
```

```
nop
ret
nop
nop
nop
nop
nop
nop
nop
nop
mov ebx, dword ptr [esp]
ret
nop
nop
nop
nop
nop
nop
mov eax, 0x804c028
cmp eax, 0x804c028
je 0x8049110
mov eax, 0
test eax, eax
je 0x8049110
push ebp
mov ebp, esp
sub esp, 0x14
push 0x804c028
call eax
add esp, 0x10
leave
ret
lea esi, [esi]
nop
ret
lea esi, [esi]
lea esi, [esi]
```

```
nop
mov eax, 0x804c028
sub eax, 0x804c028
mov edx, eax
shr eax, 0x1f
sar edx, 2
add eax, edx
sar eax, 1
je 0x8049158
mov edx, 0
test edx, edx
je 0x8049158
push ebp
mov ebp, esp
sub esp, 0x10
push eax
push 0x804c028
call edx
add esp, 0x10
leave
ret
lea esi, [esi]
ret
lea esi, [esi]
cmp byte ptr [0x804c028], 0
jne 0x8049180
push ebp
mov ebp, esp
sub esp, 8
call 0x80490e0
mov byte ptr [0x804c028], 1
leave
ret
lea esi, [esi]
ret
lea esi, [esi]
lea esi, [esi]
```

```
nop
jmp 0x8049120
lea ecx, [esp + 4]
and esp, 0xfffffff0
push dword ptr [ecx - 4]
push ebp
mov ebp, esp
push ebx
push ecx
sub esp, 0x10
call 0x80490d0
add ebx, 0x2e57
mov dword ptr [ebp - 0xc], 0
sub esp, 0xc
lea eax, [ebx - 0x1ff8]
push eax
call 0x8049060
add esp, 0x10
sub esp, 0xc
lea eax, [ebp - 0x12]
push eax
call 0x8049050
add esp, 0x10
sub esp, 8
lea eax, [ebx - 0x1fe1]
push eax
lea eax, [ebp - 0x12]
push eax
call 0x8049030
add esp, 0x10
test eax, eax
jne 0x804920c
sub esp, 0xc
lea eax, [ebx - 0x1fdc]
push eax
call 0x8049060
add esp, 0x10
```

```
mov dword ptr [ebp - 0xc], 1
jmp 0x804921e
sub esp, 0xc
lea eax, [ebx - 0x1fca]
push eax
call 0x8049060
add esp, 0x10
cmp dword ptr [ebp - 0xc], 0
je 0x804923b
sub esp, 8
push dword ptr [ebp - 0xc]
lea eax, [ebx - 0x1fb4]
push eax
call 0x8049040
add esp, 0x10
jmp 0x8049250
sub esp, 8
push dword ptr [ebp - 0xc]
lea eax, [ebx - 0x1f80]
push eax
call 0x8049040
add esp, 0x10
mov eax, 0
lea esp, [ebp - 8]
pop ecx
pop ebx
pop ebp
lea esp, [ecx - 4]
ret
nop
push ebp
call 0x80492c1
add ebp, 0x2d9a
push edi
push esi
push ebx
sub esp, 0xc
```

```
mov ebx, ebp
mov edi, dword ptr [esp + 0x28]
call 0x8049000
lea ebx, [ebp - 0xf0]
lea eax, [ebp - 0xf4]
sub ebx, eax
sar ebx, 2
je 0x80492b5
xor esi, esi
lea esi, [esi]
sub esp, 4
push edi
push dword ptr [esp + 0x2c]
push dword ptr [esp + 0x2c]
call dword ptr [ebp + esi*4 - 0xf4]
add esi, 1
add esp, 0x10
cmp ebx, esi
jne 0x8049298
add esp, 0xc
pop ebx
pop esi
pop edi
pop ebp
ret
lea esi, [esi]
ret
mov ebp, dword ptr [esp]
ret
```

Code Analysis

Pseudo C Code

main.c

```
/* WARNING: Function: __x86.get_pc_thunk.bx replaced with
injection: get_pc_thunk_bx */

undefined4 main(void)

{
  int comparisonResult;
  char enteredPassword[6];
  int isAuthenticated;
  undefined *stackPointer;

  stackPointer = &stack0x00000004;
  isAuthenticated = 0;
  puts("Enter admin password: ");
  gets(enteredPassword);
  comparisonResult = strcmp(enteredPassword, "pass");
  if (comparisonResult == 0) {
    puts("Correct Password!");
    isAuthenticated = 1;
  }
  else {
    puts("Incorrect Password!");
  }
  if (isAuthenticated == 0) {
    printf("Failed to log in as Admin (authorised=%d) :
(\n", 0);
  }
  else {
    printf("Successfully logged in as Admin
(authorised=%d) :)\n", isAuthenticated);
```

```
  }
  return 0;
}
```

Description

This code is a simple password authentication program written in C. It prompts the user to enter the admin password via the console. The entered password is then compared with the hardcoded string "pass" using the strcmp function. If the entered password matches the hardcoded string, the program outputs "Correct Password!" and sets the isAuthenticated variable to 1. If the entered password does not match, the program outputs "Incorrect Password!". Finally, the program checks the value of isAuthenticated: if it is 0, it prints a message indicating a failed login attempt, otherwise, it prints a message indicating a successful login as Admin. The main function then returns 0 to indicate the program has finished executing.

_start.c

```
/* WARNING: Function: __i686.get_pc_thunk.bx replaced with
injection: get_pc_thunk_bx */

void processEntry _start(undefined4 initialize_param,
undefined4 auxiliary_param)

{
  undefined stack_buffer[4];

  __libc_start_main(main, auxiliary_param,
&stack_buffer[0], __libc_csu_init, __libc_csu_fini,
initialize_param, stack_buffer)
  ;
  do {
                    /* WARNING: Do nothing block with
infinite loop */
```

```
  } while( true );
}
```

Description

This code is an entry point for a program, typical for a C runtime library startup routine. The function `processEntry _start` calls `__libc_start_main`, which initializes the C runtime, executes the `main` function of the program, sets up the initialization (`__libc_csu_init`) and finalization (`__libc_csu_fini`) routines. The parameters `initialize_param` and `auxiliary_param` are passed in at the start, and `stack_buffer` is used during the call to `__libc_start_main`. After the setup is complete, the code enters an infinite loop doing nothing, likely to keep the program running indefinitely.

_init.c

```
/* WARNING: Function: __x86.get_pc_thunk.bx replaced with
injection: get_pc_thunk_bx */

int initialize(EVP_PKEY_CTX *context)

{
  undefined *gmonStartPointer;

  gmonStartPointer = PTR___gmon_start___0804bffc;
  if (PTR___gmon_start___0804bffc != (undefined *)0x0) {
    gmonStartPointer = (undefined *)(*(code
*)PTR___gmon_start___0804bffc)();
  }
  return (int)gmonStartPointer;
}
```

Description

the code initializes a function named initialize that takes a single argument of type evp_pkey_ctx pointer it defines a local variable gmonstartpointer of type undefined pointer it checks if the external reference ptr__gmon_start__0804bffc is not null if it is not null it calls the function pointed to by ptr__gmon_start__0804bffc and updates gmonstartpointer the function returns the integer conversion of the gmonstartpointer the purpose is likely to initialize or trigger some internal monitoring or profiling mechanism

ChatGPT Analysis

Exploit

Fuzzing

Exploit success with these input :

- pass
- login.c

Buffer Overflow

Format String

Credits

This report was generated using automated tools and the expert analysis of security researchers.