Buffer Overflow Lab

Computer Security

Set-UID and gdb

- Execution of set-uid programs not allowed by gdb
- gdb will have trouble with an app that is setuid or one that attempts to fork a setuid program such as sendmail as a child process using system() or its relatives like popen() and execl() .
- “sh: privileges disabled because of outstanding IPC access to task” and the program or child process won't execute.

Steps to debug

- Compile using:
  `gcc prog.c –o prog` **`-g`**
- Run program in gdb using:
  `gdb OR gdb prog`
- You will get a prompt that looks like this:
  `(gdb)`
- If you didn't specify a program to run, load it in by using:
  `(gdb) file prog`

gdb

- gdb has an interactive shell and has a command history (up and down arrow keys), can auto complete words (with the TAB key) and some other features
- If you need help with a particular command, use “help”
  `(gdb) help <command>`

Basic gdb commands

- To run the program, use:
  `(gdb) run OR (gdb) r`
- To set breakpoints, use:
  `(gdb) break prog.c:10 OR`
  `(gdb) b prog.c:10`

  WHERE
  prog.c – filename to break at
  10 – Line number
- To break at a particular function, use:
  `(gdb) break printf`

Basic gdb commands

- Continue, step and next:

```
(gdb) continue
Moves to next breakpoint
(gdb) step
Execute next line of code
(gdb) next
Steps over functions as if they
were 1 line of code
```

Basic gdb commands

- Print registers

```
(gdb) info registers
Displays values of all
available registers
(gdb) print $<register_name>
Print value of register
e.G
(gdb) print $esp
Print value of stack pointer
```

Basic gdb commands

- Print and x:

```
(gdb) print var_name
Prints the value of the variable

(gdb) print/x var_name
Prints the value in hexadecimal

(gdb) x 0xaddress
Prints the value at the address

(gdb) x/n 0xaddress
Prints values from address to
address+(n words)
```

Basic gdb commands

- (gdb) print *var_name
  Prints the value at the address in var_name

  (gdb) print &var_name
  Prints the address of var_name

Basic gdb commands

- Watch:

```
(gdb) watch var_name
Watch the variable, and interrupt
when there's a change in the value
```

- Backtrace:

```
(gdb) backtrace
Produce a stack trace of the
function calls that led to a
segfault
```

Stack Frame

```
DrawLine() {
        ...
        ...
   DrawSquare() {
        ...
        ...
        ...
   }
```