GDB and x86 assembly

Note: **Feel free to come by office hours held by any of the staff.** Don't hesitate to ask for help! Our office hours exist to help you. Please visit us if you have any questions or doubts about the material.

A few useful GDB commands

For OS X users: `lldb` uses different commands. You will be expected to know `gdb`.

- run (r)
- break (b) ⟨ `func` | ***addr* | `line` ⟩: add a breakpoint at the specified spot
- step (s): continue to next line, next (n): next line, skip function calls
- stepi (si), nexti (ni): same, but at the instruction level
- continue (c): until next breakpoint
- ⟨enter⟩: repeat previous command
- print (p) [/*f*] ⟨ `var` | `$register` ⟩: print the specified value (in format *f*)
- list (l) [`line`]: show source code around the current line or `line`
- layout split: splits the GDB interface into source, assembly, and commands sections.
- disassemble (disas) [`func`]: show the assembly for the current context, or `func`
- x/*n*x[b|w] *addr*: print *n* bytes (b) or 4-byte words (w) of memory as hex (x)
  (If displaying bytes, keep in mind that x86 is *little-endian*!)

Intro to x86 assembly

32-bit x86 prefixes its registers with *e-* (eax, ebp, esp...). x86–64 uses *r-* (rax, rbp, rsp...).

In AT&T syntax, the suffixes *-b*, *-i*, *-l*, and *-q* clarify if the instruction operates on bytes, 16-bit words, 32-bit words, or 64-bit words. Source is on the left, destination on the right.

There are 8 general-purpose registers: EAX, EBX, ECX, EDX, ESI, EDI, ESP, and EBP. The registers EBP (base pointer) and ESP (stack pointer) are usually used to delimit the current function's *stack frame*.

The stack grows down (towards lower addresses), by decrementing ESP (`subl $0x18, %esp`) or using the shortcut `push`: `pushl %ebp` (decrement ESP by 4 and copy EBP there).

Correspondingly, `popl %ebp` puts the memory (ESP,ESP+4) into EBP and increments ESP.

The usual *function prologue* is

```
push %ebp       // save the top of the previous frame
mov %esp %ebp   // start new frame by moving EBP down to ESP
sub X %esp      // X = size of local variables
```

And the corresponding exit is

```
add X %esp      // * (sometimes ‘mov %ebp %esp‘)
pop %ebp        // *
ret             // pops return address from stack, goes there
// * sometimes these two lines are replaced with the leave instruction.
```

Conversely to `ret`, `call addr` pushes EIP (the instruction pointer, that is, the address of the *next* instruction) onto the stack as a saved return address before jumping to *addr*.

A more thorough overview of 32-bit x86 can be found at https://www.cs.virginia.edu/~evans/cs216/guides/x86.html

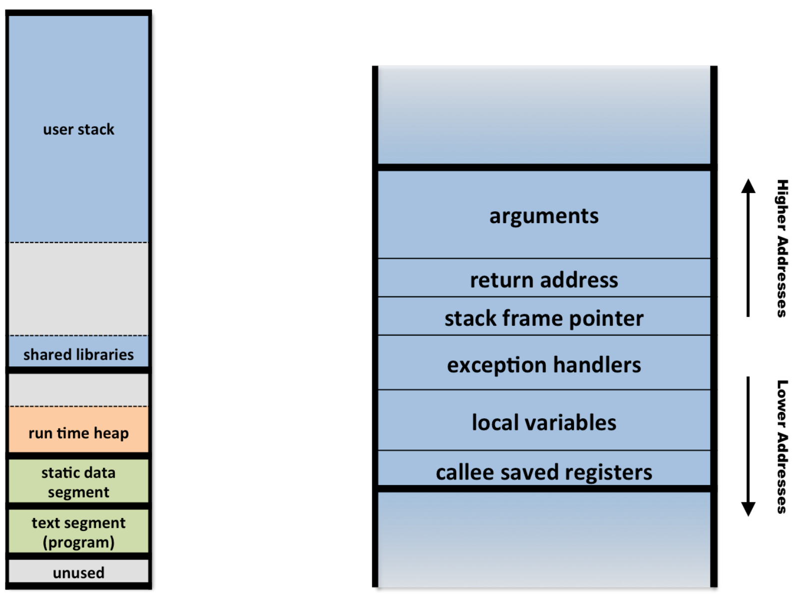

Figure 1: Left: memory layout for 32-bit Linux. The stack (left, at top) grows downward. Right: the contents of one frame on the stack.

Security Principles

We discussed the following security principles in lecture (*or in the lecture notes*, which you are responsible for reading):

A. Security is economics

B. Least privilege

C. Know your threat model

D. Defense in depth

E. Consider human factors

F. Design in security from the start

G. Ensure complete mediation

H. Division of trust

I. Consider Shannon's Maxim

Identify the principle(s) relevant to each of the following scenarios:

1. New cars often come with a valet key. This key is intended to be used by valet drivers who park your car for you. The key opens the door and turns on the ignition, but it does not open the trunk or the glove compartment.

2. Many home owners leave a house key under the floor mat in front of their door.

3. It is not worth it to use a $400,000 bike lock to protect a $100 bike.

4. Warranties on cell phones do not cover accidental damage, which includes liquid damage. Unfortunately for cell phone companies, many consumers who accidentally damage their phones with liquid will wait for it to dry, then take it in to the store, claiming that "it broke by itself". To combat this threat, many companies have begun to include on the product a small sticker that turns red (and stays red) when it gets wet.

5. Social security numbers were not originally designed as a secret identifier. Nowadays, they are often easily obtainable or guessable.

6. Even if you use a password on your laptop lockscreen, there is software which lets a skilled attacker with specialized equipment to bypass it.

7. Shamir's secret sharing scheme allows us to split a "secret" between multiple people, so that all of them have to collaborate in order to recover the secret.

8. DRM encryption is often effective, until someone can reverse-engineer the decryption algorithm.

9. Banks often make you answer your security questions over the phone. Answers to these questions are "low entropy", meaning that they are easy to guess. Some security conscious people instead use a random password as the answer to the security question.[1] However attackers can sometimes convince the phone representative by claiming "I just put in some nonsense for that question".

10. Apartments often come with a set of two keys, both of which are intended for those who live in the building: one to open the front gate and another for the individual's apartment door. The key to the front gate will not open the apartment door, and likewise, the key to the apartment door will not open the front gate.

11. Often times at bars, an employee will wait outside the only entrance to the bar, enforcing that people who want to enter the bar form a single-file line. Then, the employee checks each individual's ID to verify if they are 21 before allowing them entry into the bar.

Solution: (Note that there may be principles that apply other than those listed below.)

1. Principle of least privilege. They do not need to access your trunk or your glove box, so you don't give them the access to do so.
2. Shannon's Maxim. The security of your home depends on the belief that most criminals don't know where your key is. With a modicum of effort, criminals could find your key and open the lock.
3. Security is economics. It is more expensive to buy $400 bike lock than to simply buy a new bike to replace it.
4. There are probably two most relevant factors. "Consider human factors": people will always try to lie and you must account for that when creating a system. More

[1] Q: "What is your dog's maiden name?". A: "60ba6b1c881c6b87"

importantly, "Design in security from the start": it's prudent to try to add ways to detect something when creating the phone rather than trying to determine water damage after-the-fact.

5. Design security in from the start. Social security numbers were not designed to be authenticators, so security was not designed in from the start. The number is based on geographic region, a sequential group number, and a sequential serial number. They have since been repurposed as authenticators.

6. Know your threat model: most petty thieves do not have access to this software. (The software referenced is pcileech. The corresponding hardware is on my wishlist. -Keyhan Vakil)

7. Division of trust: require everyone to come together to produce the secret, preventing one person from using the secret alone.

8. Shannon's Maxim. You must assume the attacker knows the system, so DRM encryption is not effective.

9. Consider human factors. The phone rep is inclined to believe the attacker is not malicious (social engineering).

10. Defense in depth. In order for someone to break into your apartment, they would have needed the key to both the front gate (or bypass entry somehow) and the apartment door. Additionally, this could potentially have to do something to do with the principle of least privilege. Someone who has legitimate need of access to just the front gate (e.g., someone delivering packages and mail) may have the key or code to just the front gate.

11. Complete mediation. There is a single access point through which everyone who wishes to enter the bar must be verified to be 21 before obtaining access.