Software Security

Daniel Bosk

Department of Information and Communication Systems,
Mid Sweden University, SE-851 70 Sundsvall

14th May 2018

1 Introduction
- Security and Reliability
- Changes

2 Broken Abstractions
- File System Paths
- Character Encoding
- Integer Overflows
- Data and Code

3 Memory Management
- Memory Structure
- Overruns
- Type Confusion

4 Malware
- Background
- Malware Types

- As long as our computer is offline, used only by ourselves, and we don't add any accessories (e.g. USB devices [Sch14]), then we don't have any problems.
- Problems start to occur when other users start using our software (in some way), then input to our programs isn't necessarily what we expect.

- As long as our computer is offline, used only by ourselves, and we don't add any accessories (e.g. USB devices [Sch14]), then we don't have any problems.
- Problems start to occur when other users start using our software (in some way), then input to our programs isn't necessarily what we expect.

Software reliability This concerns software quality in the sense of accidental failures, i.e. the assumption that input is benign.

Software security This concerns software quality in the sense of intentional failures, i.e. the assumption that input is malign.

Software reliability This concerns software quality in the sense of accidental failures, i.e. the assumption that input is benign.

Software security This concerns software quality in the sense of intentional failures, i.e. the assumption that input is malign.

- Change is one of the dangers to security.
- There are systems which are designed to be secure, and actually are secure, but then ...
- upgrades are needed, or not needed but wanted.
- This might come in the form of updating a component or utilizing the system in an environment it wasn't designed for.

1 Introduction
- Security and Reliability
- Changes

2 Broken Abstractions
- File System Paths
- Character Encoding
- Integer Overflows
- Data and Code

3 Memory Management
- Memory Structure
- Overruns
- Type Confusion

4 Malware
- Background
- Malware Types

```python
#!/bin/env python3
import sys, os

JAIL_PATH = os.environ["HOME"]

def jailed_open(filename):
    return open(JAIL_PATH + "/" + filename)

def main(argv):
    f = jailed_open(argv[1])

    print("\\begin{verbatim}")
    for line in f.readlines():
        print(line.strip())
    print("\\end{verbatim}\n")

if __name__ == "__main__":
```

Example (./jail.py ../../etc/passwd)

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```

The Problem: Abstraction of paths

- We had `JAIL_PATH = os.environ["HOME"]`.
- We let `filename = "../../etc/passwd"`.
- Thus the file we open is `JAIL_PATH + "/" + filename` which results in /home/dbosk/../../etc/passwd .
- Hence we actually read /etc/passwd.

- Fine, we ban the string `"../"`.
- Then what about `"..\%c0\%af.."`?

- All character representations in the computer comes in the form of different encodings, e.g. UTF-8 encoding.
- The decoders might be programmed differently, some takes into account the errors in different encoders to compensate – and this can be exploited.
- Where the encoding and decoding is done can also be exploited.

UTF-8

```
1  char buf[128];
2
3  void
4  combine( char *s1, size_t len1, char *s2, size_t len2)
5  {
6    if ( len1 + len2 + 1 <= sizeof(buf) ) {
7      strncpy( buf, s1, len1 );
8      strncat( buf, s2, len2 );
9    }
10 }
```

The Problem: Abstraction of integers

- Let `len2` be very long, say $2^{32} - 1$, i.e. `len2 = 0xffffffff`.
- Now we have

$$\begin{aligned} \texttt{len1} + \texttt{len2} + 1 \pmod{2^{32}} &= \texttt{len1} + 2^{32} - 1 + 1 \pmod{2^{32}} \\ &= \texttt{len1} \pmod{2^{32}} \\ &< \texttt{sizeof(buf)}. \end{aligned}$$

- Thus we pass the test, although we shouldn't.

Note

This is worse if we use *signed* integers . . .

Example (echo.sh "-E test\ning")

```
1 #!/bin/sh
2 /bin/echo -e ${1}
```

```
test\ning
```

Example (echofix.sh "-E test\ning")

```
1 #!/bin/sh
2 /bin/echo -e "${1}"
```

```
-E test
ing
```

- The login(1) and rlogin(1) composition bug was found in Linux and AIX systems which didn't check the syntax of the username.
- The syntax of login(1) is `login [-p] [-h host] [[-f] user]`.
- The syntax of rlogin(1) is `rlogin [-l user] machine`.
- rlogin(1) connects to the machine and runs `login user machine`.
- However, the user could be chosen to be "-froot".

```
1 cat ${1} | mail ${2}
```

- What happens with the address `"foo@bar.org | rm -Rf /"`?

```
1  $sql = "SELECT * FROM client WHERE name = '$name'"
```

- Insert the name `Eve' OR 1=1--`.
- This will get a totally different meaning.

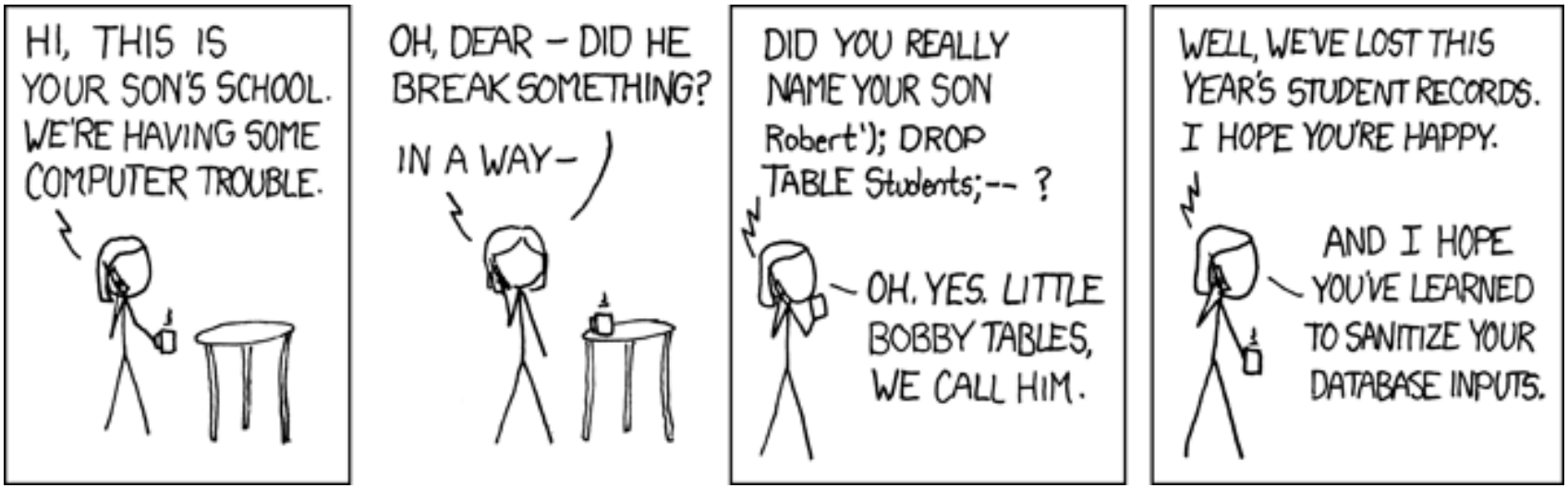


Figure: XKCD's Exploits of a Mom. Image: [XKC].

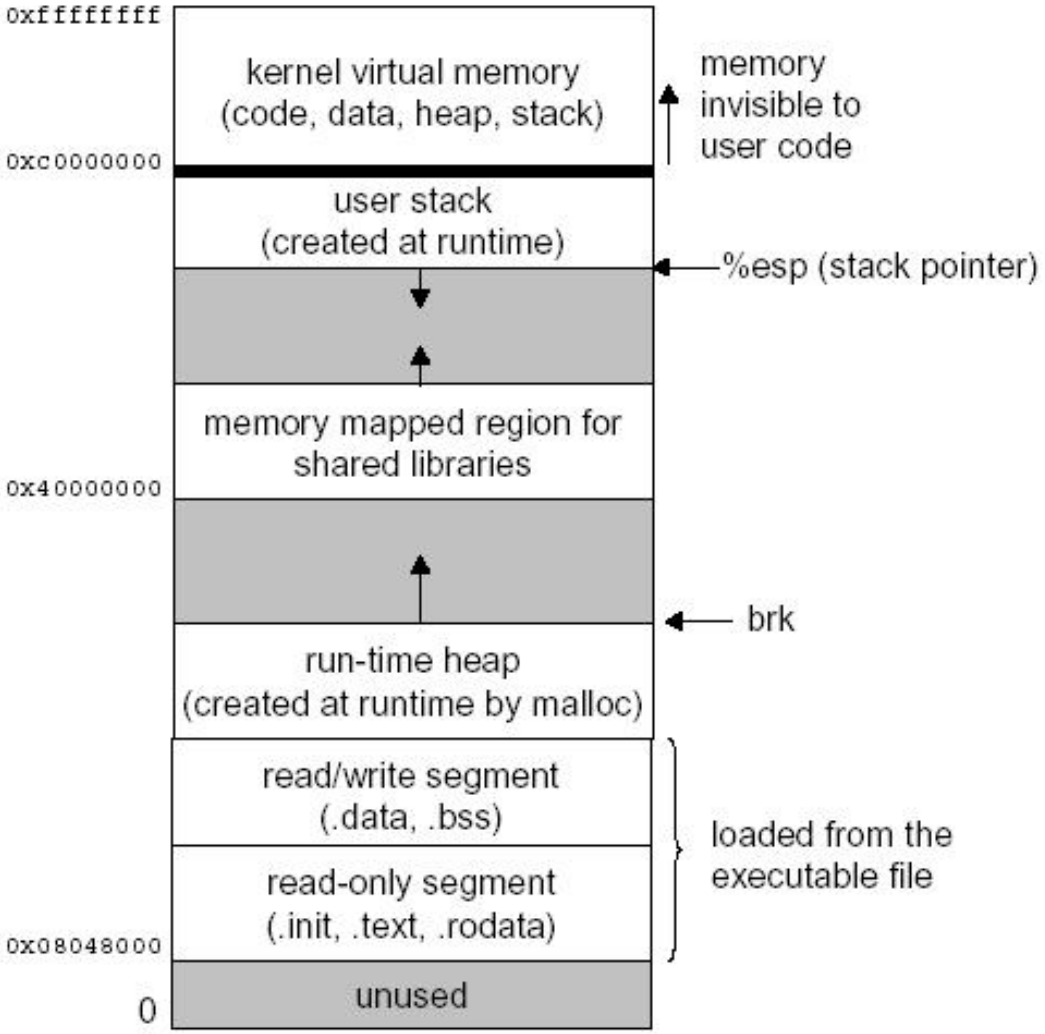
1 Introduction
- Security and Reliability
- Changes

2 Broken Abstractions
- File System Paths
- Character Encoding
- Integer Overflows
- Data and Code

3 Memory Management
- Memory Structure
- Overruns
- Type Confusion

4 Malware
- Background
- Malware Types

Memory Structure



- Buffer overruns
  - Stack overruns
  - Heap overruns
- All variables in a program use storage from either the stack or heap.

```
1  int
2  login( void )
3  {
4    char correct_password[] = "swordfish";
5    char user_password[16] = {0};
6
7    printf( "user password: ");
8    fscanf( "\%s", user_password );
9
10   if ( !strcmp( correct_password, user_password ) )
11     return 0;
12   return 1;
13 }
```

- There are some problems in object-oriented languages too.
- Trick the system to point to a different memory location.
- Thus a write using one type actually modifies something believed to be of another type somewhere else.

1 Introduction
- Security and Reliability
- Changes

2 Broken Abstractions
- File System Paths
- Character Encoding
- Integer Overflows
- Data and Code

3 Memory Management
- Memory Structure
- Overruns
- Type Confusion

4 Malware
- Background
- Malware Types

- Comes from *malicious software* and means software with a malicious intent.
- In the early days they were mostly experiments or pranks.
- Today they are mostly used for special purposes:
  - steal personal, financial or business information,
  - cripple competition,
  - etc.

- There are many types of malware.
- Their classification depends on the largest threat vector.

Computer Virus A form of malware which has self-replicating code. It *infects* other programs by inserting itself into their program code, and in turn when these programs are run the virus payload is run to replicate even further.

Worm A form of malware which replicates itself, not by infection, but by copying itself to different disks, via networks, or even emailing itself automatically to everyone in the user's contact list.

Trojan Horse A form of malware which acts as a legitimate program but has hidden features which are malicious, e.g. a utility program which steals your login credentials in the background or simply acts as a backdoor. Usually used in combination of social engineering.

Malware Types

Rootkit A piece of software designed to provide access that would otherwise be restricted. It also keeps well-hidden and is notoriously difficult to detect and remove. Usually this comes from modifying the operating system.

Spyware This software simply tries to gather information about a target without their knowledge. Usually the collected information is sent to a third party. Keylogging falls under this category.

Adware This is simply a type of malware that presents advertisements to the user of the infected system. Obviously staying undetected is not an option, so making itself difficult to remove is the strategy of choice.

Scareware This is a type of malware that uses social engineering to trick users to buy unwanted software, e.g. fake antivirus software.

Ransomware This is a type of malware that restricts the users access to the system. A common technique is to encrypt all the user's files. Then the user is presented with the option of buying the decryption key for bitcoins.
They typically propagate as trojans.

[Sch14] David Schneider. "USB Flash Drives Are More Dangerous Than You Think". In: *IEEE Spectrum* (Aug. 2014). URL: http://spectrum.ieee.org/tech-talk/computing/embedded-systems/usb-flash-drives-are-more-dangerous-than-you-think.

[XKC] XKCD. *Exploits of a Mom*. URL: http://xkcd.com/327/.