Protection

Disclaimer: some slides are adopted from book authors' slides with permission

Examples of OS Protection

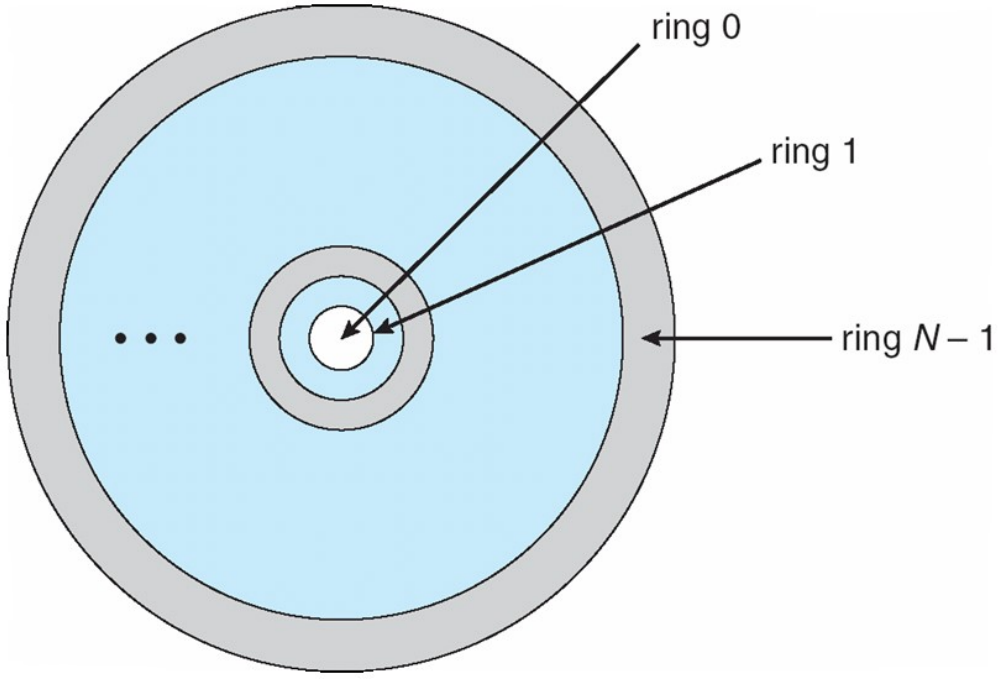
- Memory protection
  - Between user processes
  - Between user and kernel
- File protection
  - Prevent unauthorized accesses to files
- Privileged instructions
  - Page table updates
  - Cache/TLB updates

Principles of Protection

- Principle of least privilege
  - Programs and users should be given just enough privileges to perform their tasks
  - Limit the damage if the entity has a bug or abused

Protection Domains

- Let D_i and D_j be any two domain rings
- If $j < I \Rightarrow D_i \subseteq D_j$
- Kernel mode vs. user mode



Access Control Matrix

- **Domains** in rows
  - Domain: a user or a group of users
- **Resources** in columns
  - File, device, …

E.g., User D1 can read F1 or F3

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Method 1: Access Control List

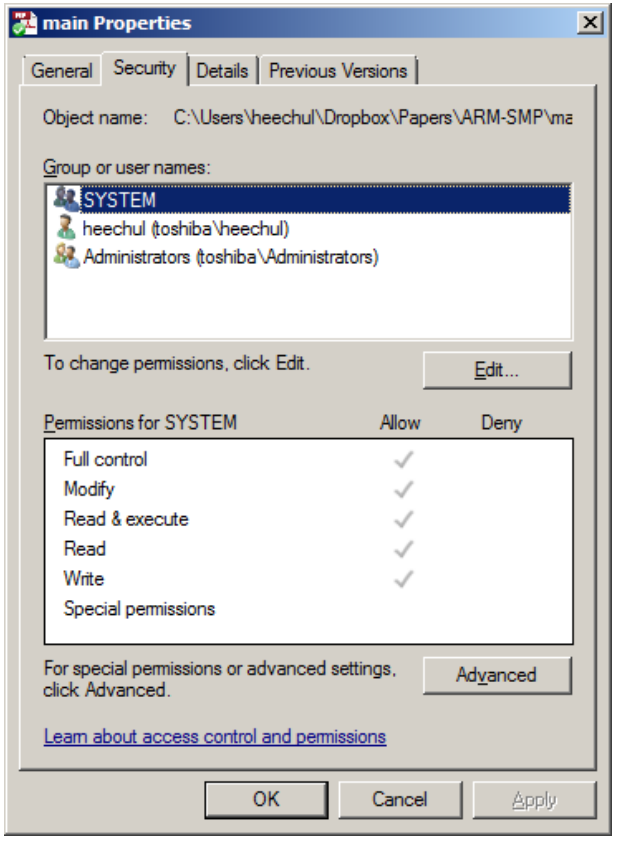
- Each **object** stores users and their permissions

-rw-rw-r-- heechul heechul 38077 Apr 23 15:16 **main.tex**

owner

group

world



Method 2: Capability List

- Each **domain** tracks which objects can access
  - Page table: each process (domain) tracks all pages (objects) it can access

Summary

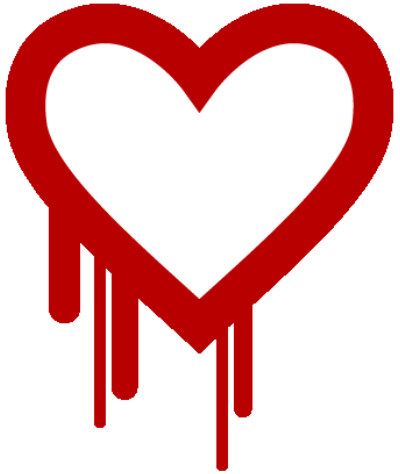
- Protection
  - Prevent unintended/unauthorized accesses
- Protection domains
  - Class hierarchy: *root* can to everything a normal *user* can do + alpha
- Access control matrix
  - Domains (Users) ← → Resources (Objects)
  - Resource oriented: Access control list
  - Domain oriented: Capability list

Security

Today

- Security basics

- Some recent security bugs
  - Heartbleed bug (OpenSSL)
  - Goto fail bug (Apple SSL)
  - Shellshock bug (Bash)



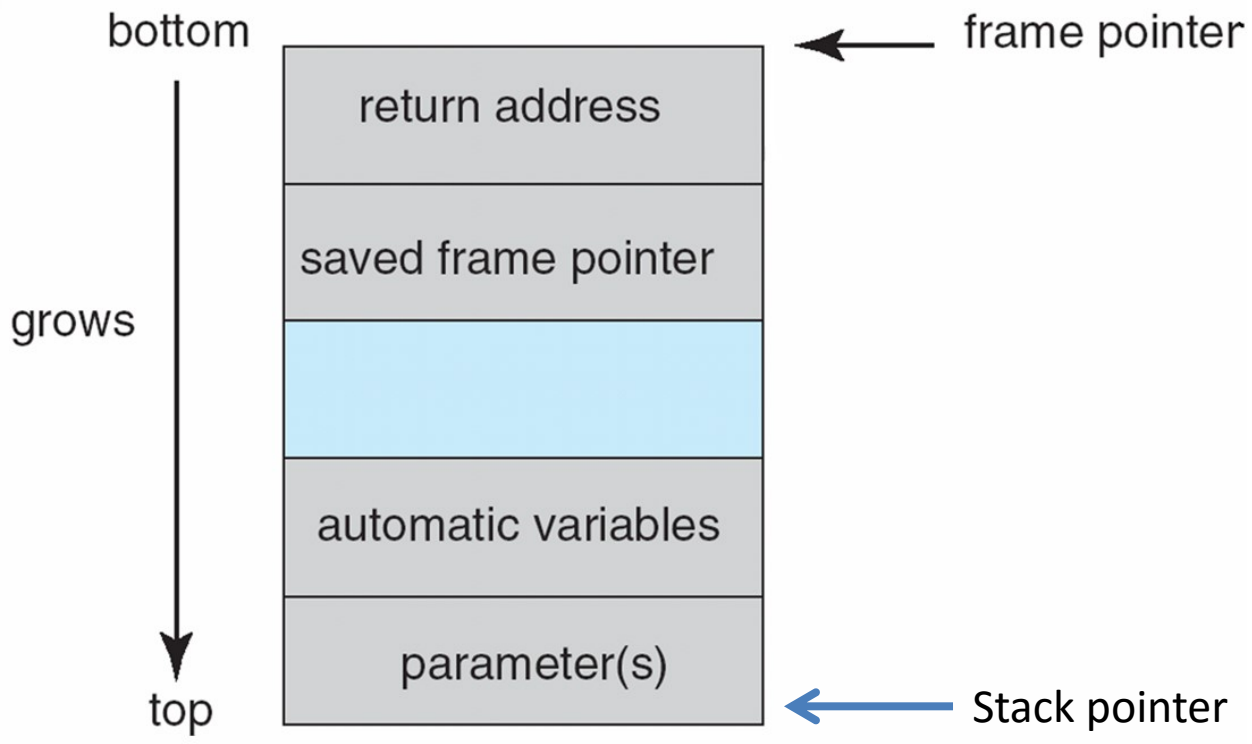
Security

- System **secure** if resources used and accessed as intended under all circumstances
  - Unachievable
- **Intruders** (**crackers**) attempt to breach security
- **Threat** is potential security violation
- **Attack** is attempt to breach security

Program Threats

- **Stack and Buffer Overflow**
  - Exploits a bug in a program (overflow either the stack or memory buffers)
  - Failure to check bounds on inputs, arguments
  - Write past arguments on the stack into the return address on stack
  - When routine returns from call, returns to hacked address
    - Pointed to code loaded onto stack that executes malicious code
  - Unauthorized user or privilege escalation

Stack Frame Layout



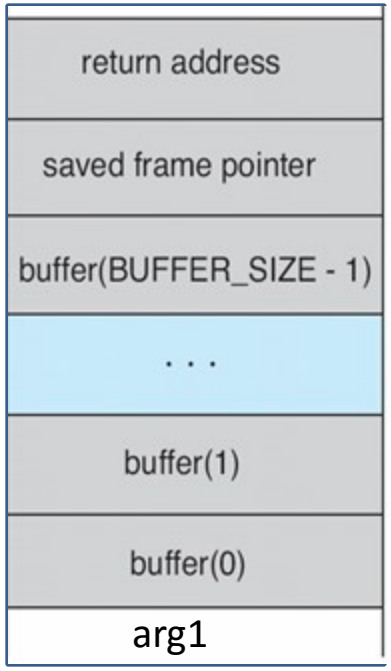
Code with Buffer Overflow

```
#define BUFFER_SIZE 256
int process_args(char *arg1)
{
    char buffer[BUFFER SIZE];
    strcpy(buffer,arg1);
    ...
}

int main(int argc, char *argv[])
{
    process_args(argv[1]);
    ...
}
```

- What is wrong in this code?

Code with Buffer Overflow

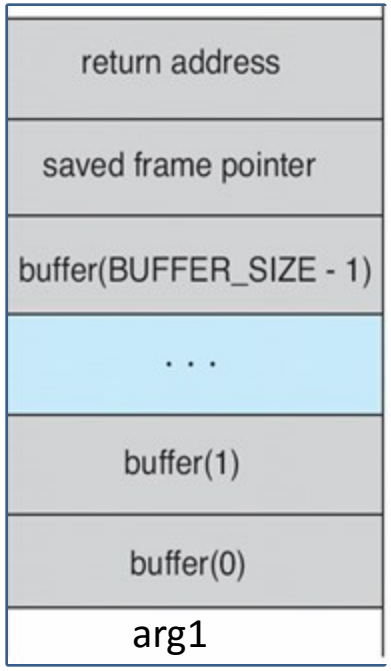
```
#define BUFFER_SIZE 256
int process_args(char *arg1)
{
    char buffer[BUFFER SIZE];
    strcpy(buffer,arg1);
    ...
}

int main(int argc, char *argv[])
{
    process_args(argv[1]);
    ...
}
```



- Stack layout after calling *process_arg()*

Code with Buffer Overflow

```
#define BUFFER_SIZE 256
int process_args(char *arg1)
{
    char buffer[BUFFER SIZE];
    strcpy(buffer,arg1);
    ...
}

int main(int argc, char *argv[])
{
    process_args(argv[1]);
    ...
}
```



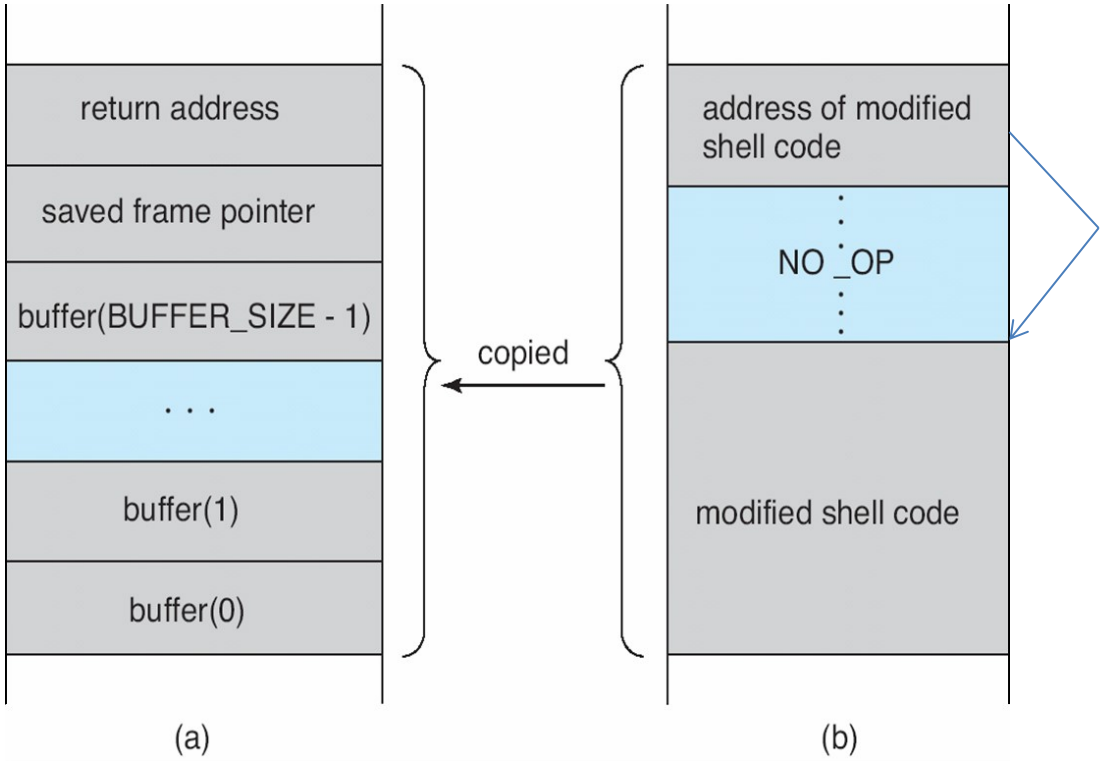
- Do you remember *strcpy()* in C?

Let's Get the Shell

- Steps
  - Compile the code you want to illegitimately execute
  - 'Carefully' modify the binary
  - Pass the modified binary as string to the *process_arg()*

```
#include <stdio.h>
int main(int argc, char *argv[])
{
        execvp(``/bin/sh'',``/bin/sh'', NULL);
        return 0;
}
```

The Attack: Buffer Overflow



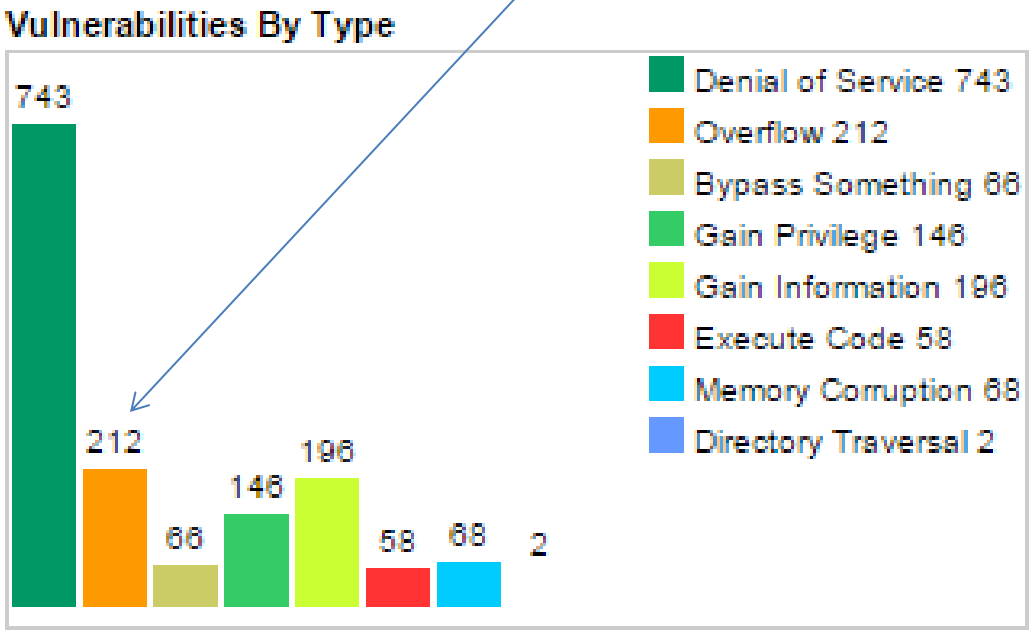
Before

After executing
strcpy(buffer, ***arg1****)*

the crafted string containing the illegitimate code

Linux Kernel Buffer Overflow Bugs

212 reported buffer overflow bugs in Linux



Source: http://www.cvedetails.com/vulnerability-list/vendor_id-33/product_id-47/cvssscoremin-9/cvssscoremax-/Linux-Linux-Kernel.html

Linux Kernel Buffer Overflow Bugs

6	CVE-2010-2521 119	DoS Exec Code Overflow	2010-09-07	2012-03-19	10.0	None	Remote	Low	Not required	Complete	Complete Complete

Multiple buffer overflows in fs/nfsd/nfs4xdr.c in the XDR implementation in the NFS server in the Linux kernel before 2.6.34-rc6 allow remote attackers to cause a denial of service (panic) or possibly execute arbitrary code via a crafted NFSv4 compound WRITE request, related to the read_buf and nfsd4_decode_compound functions.

9	CVE-2009-0065 119	Overflow	2009-01-07	2012-03-19	10.0	Admin	Remote	Low	Not required	Complete	Complete Complete

Buffer overflow in net/sctp/sm_statefuns.c in the Stream Control Transmission Protocol (sctp) implementation in the Linux kernel before 2.6.28-git8 allows remote attackers to have an unknown impact via an FWD-TSN (aka FORWARD-TSN) chunk with a large stream ID.

10	CVE-2008-5134 119	Overflow	2008-11-18	2012-03-19	10.0	None	Remote	Low	Not required	Complete	Complete Complete

Buffer overflow in the lbs_process_bss function in drivers/net/wireless/libertas/scan.c in the libertas subsystem in the Linux kernel before 2.6.27.5 allows remote attackers to have an unknown impact via an "invalid beacon/probe response."

11	CVE-2008-3915 119	Overflow	2008-09-10	2012-03-19	9.3	None	Remote	Medium	Not required	Complete	Complete Complete

Buffer overflow in nfsd in the Linux kernel before 2.6.26.4, when NFSv4 is enabled, allows remote attackers to have an unknown impact via vectors related to decoding an NFSv4 acl.

12	CVE-2008-3496 119	Overflow	2008-08-06	2012-03-19	10.0	None	Remote	Low	Not required	Complete	Complete Complete

Buffer overflow in format descriptor parsing in the uvc_parse_format function in drivers/media/video/uvc/uvc_driver.c in uvcvideo in the video4linux (V4L) implementation in the Linux kernel before 2.6.26.1 has unknown impact and attack vectors.

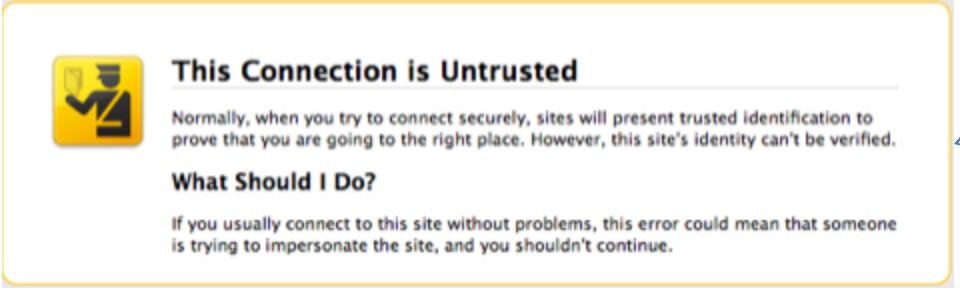
13	CVE-2008-1673 119	DoS Exec Code Overflow	2008-06-09	2012-11-26	10.0	None	Remote	Low	Not required	Complete	Complete Complete

Goto Fail Bug

iOS 7.0.6
Data Security
Available for: iPhone 4 and later, iPod touch (5th generation), iPad 2 and later

Impact: An attacker with a privileged network position may capture or modify data in sessions protected by SSL/TLS

Description: Secure Transport *failed to validate the authenticity of the connection*. This issue was addressed by restoring missing validation steps.



Goto Fail Bug

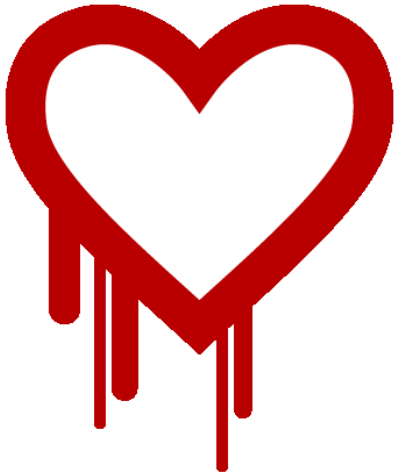
```
    err = 0
    . . .
    hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
    hashOut.length = SSL_SHA1_DIGEST_LEN;
    if ((err = SSLFreeBuffer(&hashCtx)) != 0)
     goto fail;
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
     goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
     goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
     goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
     goto fail;
     goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
     goto fail;

    err = sslRawVerify(...); // This code must be executed
    . . .
fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    Return err;
```

MISTAKE! THIS LINE SHOULD NOT BE HERE

KU THE UNIVERSITY OF KANSAS

Heartbleed Bug

- Synopsis
  - Due to a bug in OpenSSL (popular s/w for encrypted communication), web server's internal memory can be dumped remotely



Heartbleed Bug

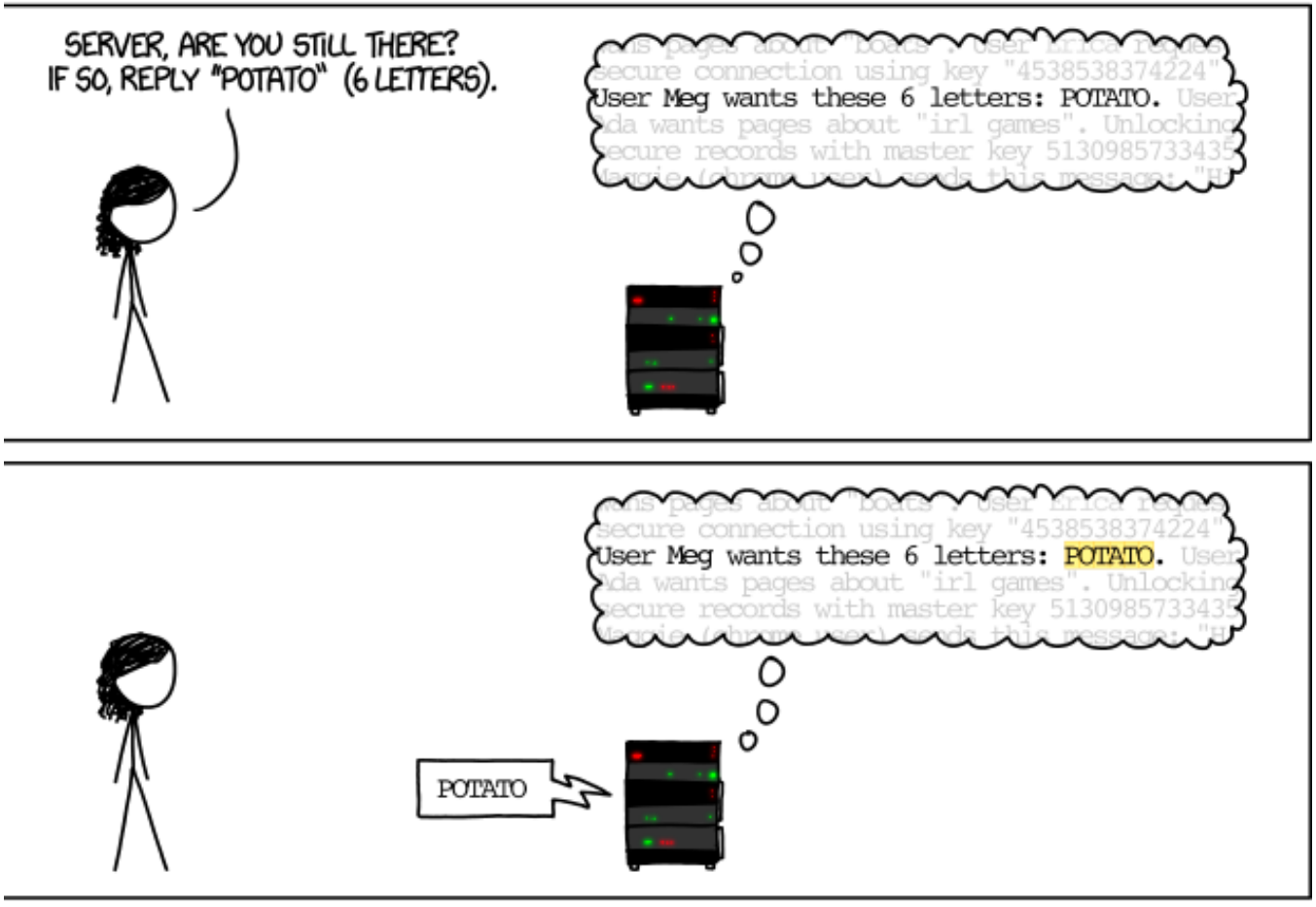


Image source: xkcd.com

Heartbleed Bug

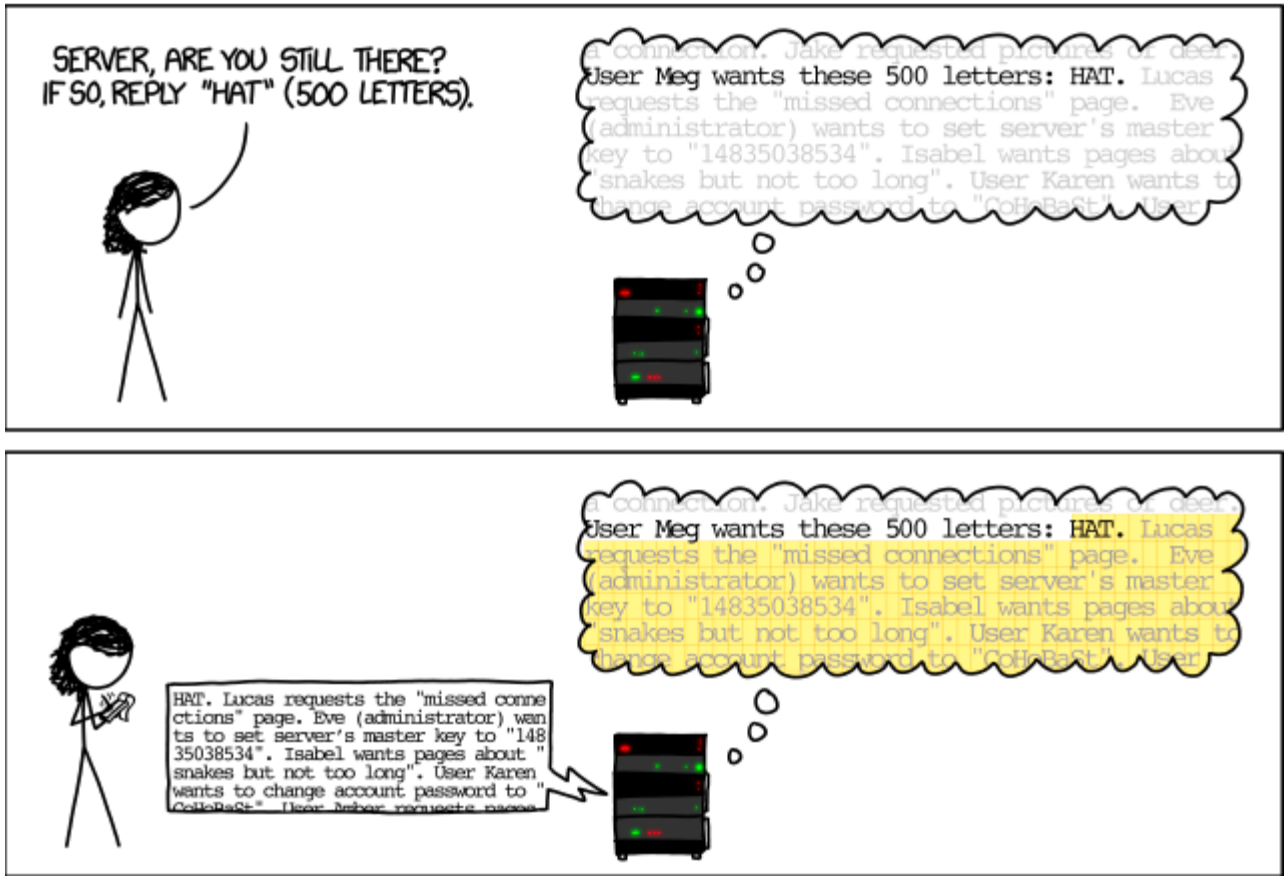


Image source: xkcd.com

Heartbleed Bug

```
struct {
   HeartbeatMessageType type;
   uint16 payload_length;
   opaque payload[HeartbeatMessage.payload_length];
   opaque padding[padding_length];
} HeartbeatMessage
```

Heartbeat
req. message

```
int tls1_process_heartbeat(SSL *s)
{
   ...
   /* Read type and payload length first */
   hbtype = *p++;
   n2s(p, payload); // payload = recv_packet.payload_length
   pl = p;
   ...
   if (hbtype == TLS1_HB_REQUEST) {
      ...
      buffer = OPENSSL_malloc(1 + 2 + payload + padding);
      bp = buffer;
      memcpy(bp, pl, payload);
      r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
                  ...
```

Heartbeat
Response function

Shellshock Bug

- Synopsis
  - You can *remotely execute arbitrary programs* on a server running a web server by simply sending a specially crafted http request.
  - Example

```
curl -H "User-Agent: () { :; }; /bin/eject" http://example.com/
```

- The problem
  - Fail to check the validity of a function definition before executing it

For detailed explanation: security.stackexchange.com