IT security

Monday 11th March
Course day #5

Theme B: Software and system security

Case: The Code Red buffer overflow attack

Niels Christian Juul (ncjuul@ruc.dk)
Niels Jørgensen (nielsj@ruc.dk)

Literature

Stallings & Brown

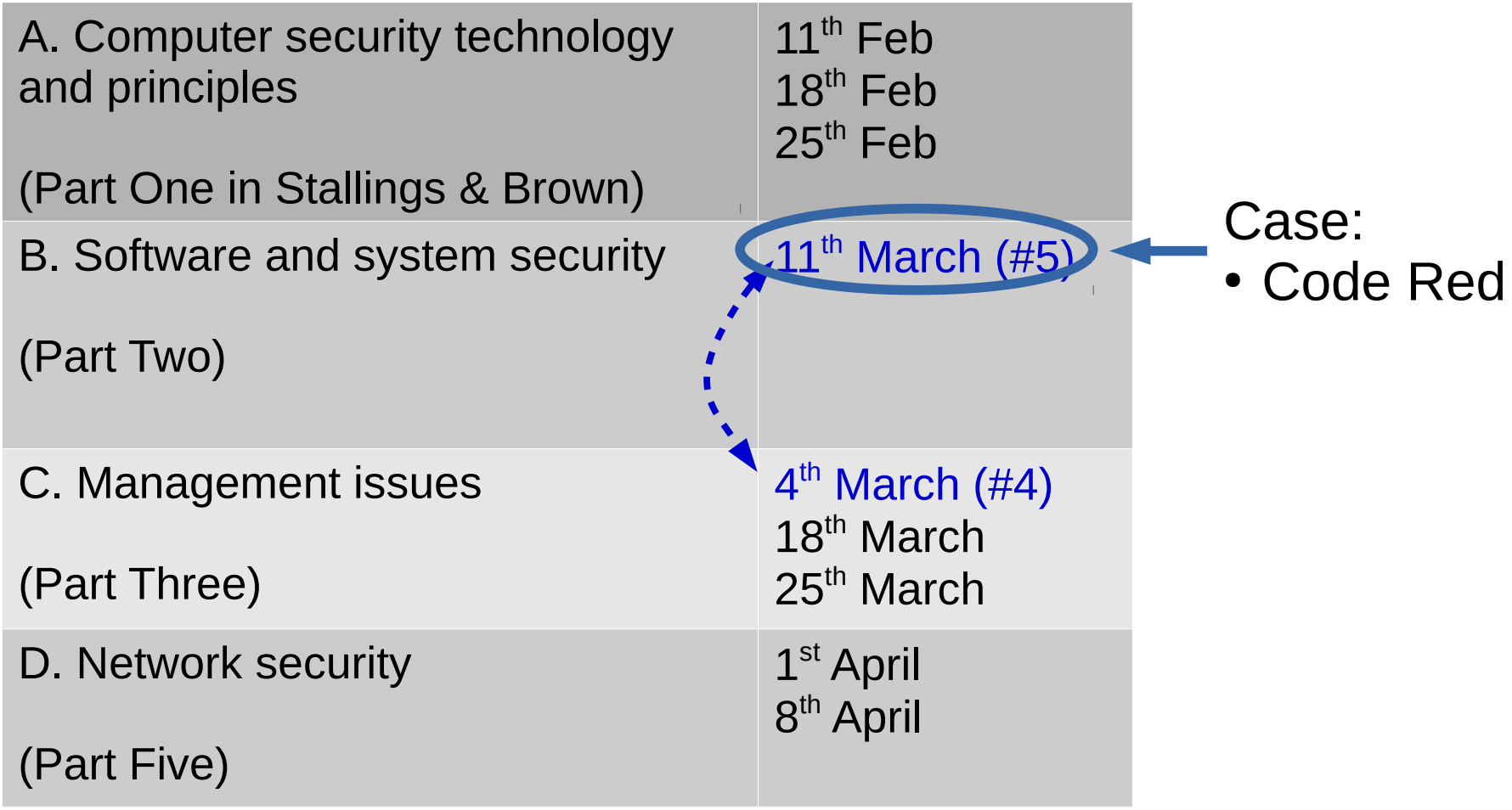
- Chapter 10:
  - buffer overflow: 341-347, 364-367
- Chapter 13:
  - cloud computing: 445-451, 457-464
  - IoT: 466-473

Additional mandatory literature:

- *“Code red"*, Communications of the ACM, December 2001.
- *"Webcam hack shows vulnerability of connected devices",* Engineering & Technology December 2016.

Four themes (A-D)

A. Computer security technology and principles (Part One in Stallings & Brown)	11th Feb 18th Feb 25th Feb
B. Software and system security (Part Two)	11th March (#5)
C. Management issues (Part Three)	4th March (#4) 18th March 25th March
D. Network security (Part Five)	1st April 8th April



Exam questions for theme B (see list on moodle)

Q7: What is a buffer overflow?

Q8: What are the main preventive measures an enterprise can take against buffer overflows and against threats to cloud and IoT security?

Plan for today

Buffer overflows: introduction

Code Red (Anton)

Buffer overflows: how they work

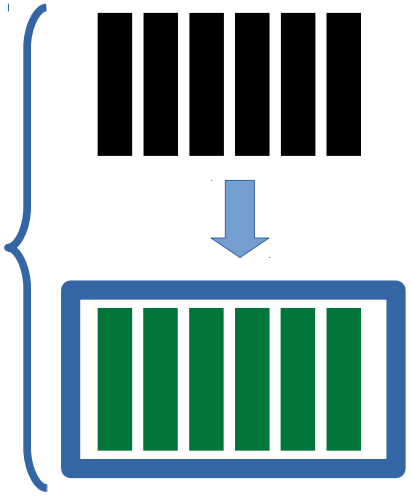
Buffer overflows: prevention

Cloud security

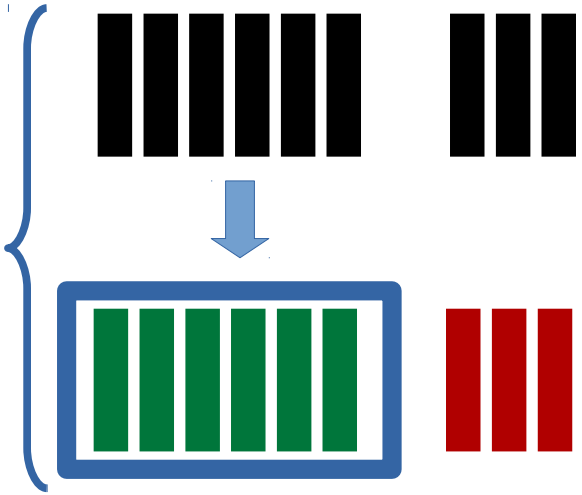
IoT security

Definition of a buffer overflow

Intended use



Overflow



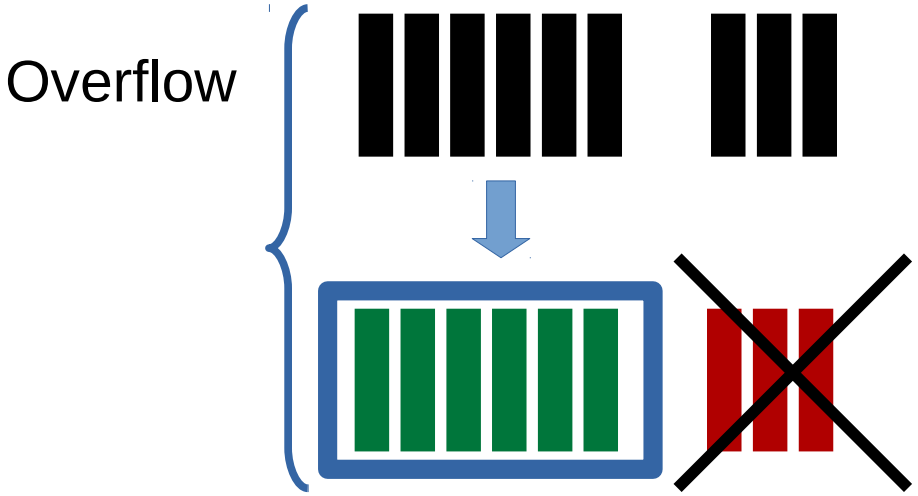
".. more input can be placed into a buffer .. than the capacity allocated, overwriting other information"

(Stallings & Brown, p 343)

Prevention

A program must reject attempts to write more data to a buffer (an array) than the capacity allocated.

Rejection may be by array bounds checking, as in Java



Exploiting buffer overflows

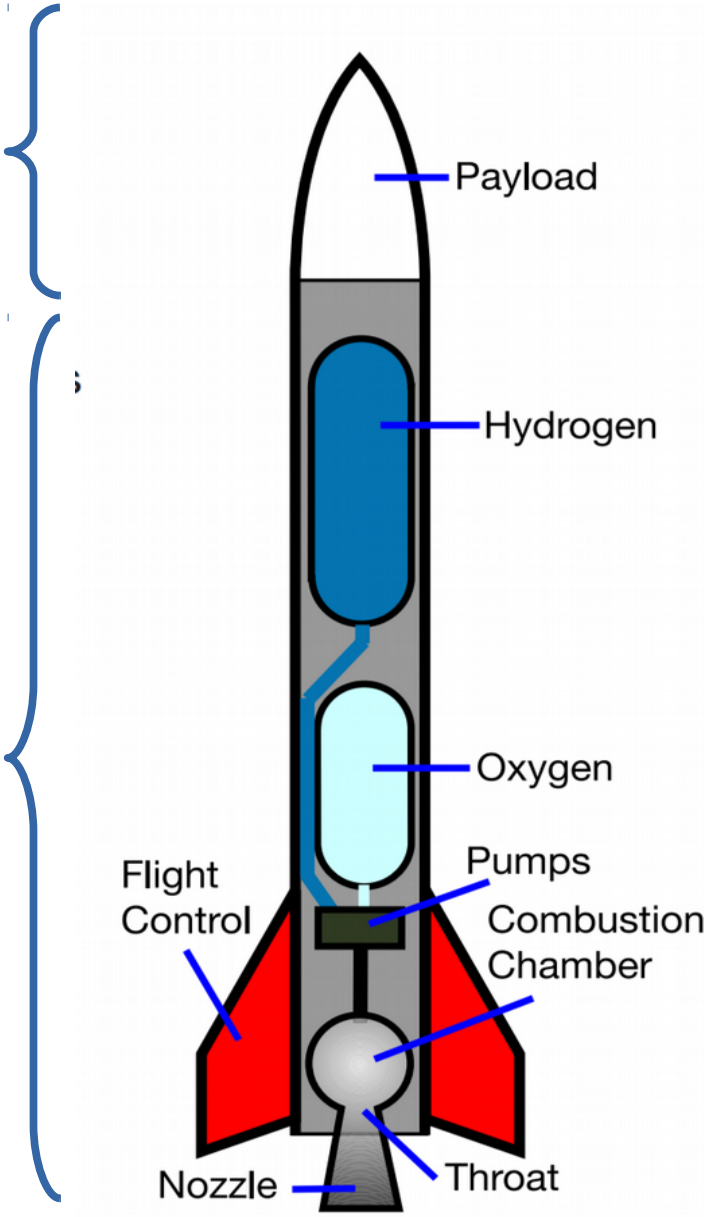
Exploiting a buffer overflow provides a propagation method.

An exploit may propagate different paylods
- malicious data..
- malicious programs..

Exploiting buffer overflows to run mailicious programs is perhaps the most common type of exploit
- Morris worm
- Code Red
- ..

A malware's payload
- *it's impact on the computer*

A malware's propagation method
- *how it spreads to other computers*



Plan for today

Buffer overflows: introduction

Code Red (Anton)

Buffer overflows: how they work

Buffer overflows: prevention

Cloud security

IoT security

Code Red buffer overflow

GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNN
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801
%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3
%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0

Plan for today

Buffer overflows: introduction

Code Red (Anton)

Buffer overflows: how they work

Buffer overflows: prevention

Cloud security

IoT security

Buffer overflows in C
Stallings & Brown, Figure 10.1, p 344

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

Suppose we are debugging the program

- therefore, we have inserted the println statement

Step 1: What is the C program supposed to do?
Step 2: In what ways is the C program not doing what it is supposed to?
Step 3: Understand the C program
Step 4: Why is the C program not doing what it is supposed to?

Step 1: What is the C program supposed to do?

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

The program compares two strings (arrays of characters)
- str1 is assigned by the program (eg., “START”)
- str2 is supplied by the user

If the strings are equal, the variable *valid* is assigned the value TRUE
- otherwise it is assigned the value FALSE
- note that TRUE is 1 and FALSE is 0

The program is similar to a program that compares two password hashes:
- one hash retrived from the password file
- another hash resulting from hashing a user-supplied password

My variant of buffer1 (on moodle)

```
#include <stdio.h>        // defines printf
#include <string.h>       // defines strncmp

void next_tag(char[]);    // template ("declaration") of function next_tag()

int main(int argc, char * argv[]) {
  int valid = 0;
  char str1[8];
  char str2[8];

  next_tag(str1);
  gets(str2);
  if (strncmp(str1, str2, 8) == 0) valid = 1;
  printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}

void next_tag(char c[]) {
  c[0] = 'S';
  c[1] = 'T';
  c[2] = 'A';
  c[3] = 'R';
  c[4] = 'T';
  c[5] = '\0';
}
```

My variant defines next_tag()

The function definition uses the formal parameter c to assign values to the actual parameter str1

Step 1: What is the C program supposed to do?

test case	str1 (input / final)	str2 (input / final)	valid (final)
SB#1	START / START	START / START	1
(NJ)	START / START	STOP / STOP	0

In these two test cases, the program does what it is supposed to.

Step 2: In what ways is the C program not doing what it is supposed to?

test case	str1 (input / final)	str2 (input / final)	valid (final)
SB#2	START / TVALUE	EVILINPUTVALUE / EVILINPUTVALUE	0
SB#3	START / BADINPUT	BADINPUTBADINPUT / BADINPUTBADINPUT	1

In test case SB#2:
- part of str2 overflows into str1

In test case SB#3
- part of str2 overflows into srt1
- the string comparison succeeds, because the first eight characters are equal

Step 3: Understand the C program

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

char str1[8]:	declares an array of characters with length 8
char str2[8]:	same
next_tag(str1):	calls a user-defined function, and passes a pointer to the array str1
gets(str1):	calls the system-defined function gets() for reading a string input by the user
strncmp(.., .., 8)	compares the first eight characters of the two strings, returns 0 if equal

Step 3: Understand the C program (cont.)

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

printf(format, arguments...)

prints format, where:

- ordinary characters are merely printed
  - eg., the ordinary characters “buffer1: str1(“ are merely printed
- conversions specifications convert and print the arguments
  - eg., the conversion spec. %s converts and prints str1 (in this case, the conversion is trivial)

printf(“str1(%.3s)”, str1) would print “str1(STA)” (the first three characters of str1)

Exercise

test case	**str1 (input / final)**	**str2 (input / final)**	**valid (final)**
ex1	START / ?	STARTXX / ?	?
ex2	START / ?	STARTXXXX / ?	?
ex3	START / ?	XXXXXXXXXX / ?	?

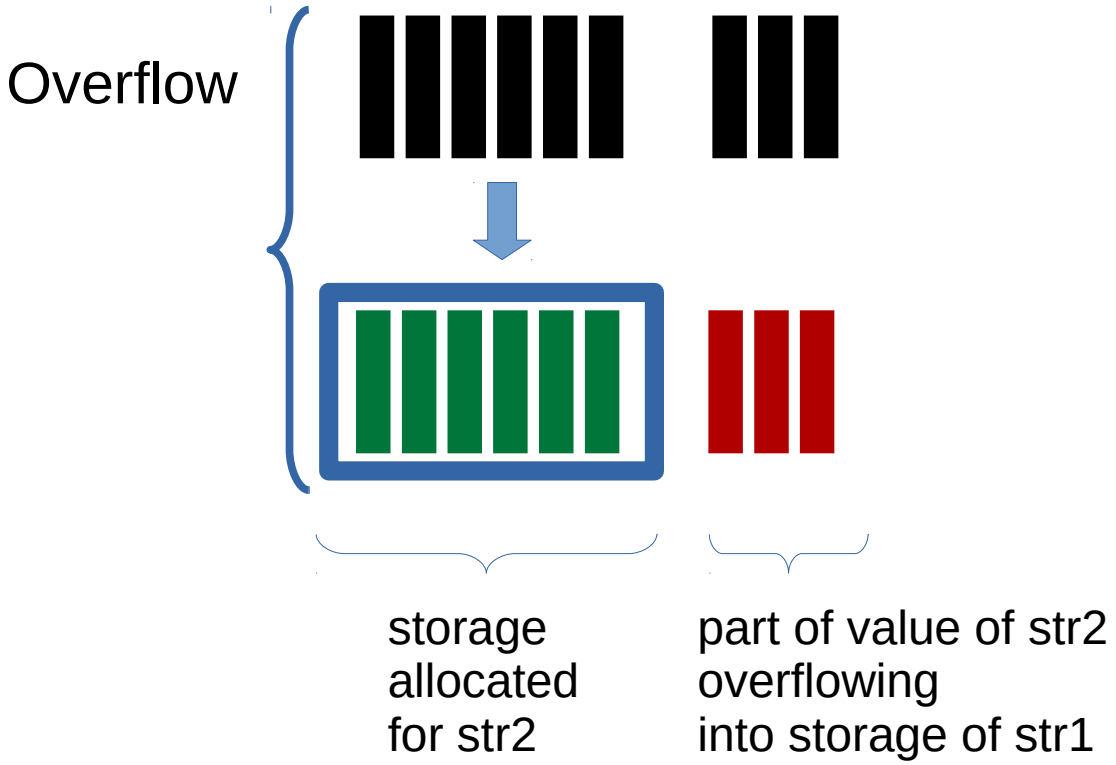
Substitute the question marks with the three final values (as printed)

Exercise solution

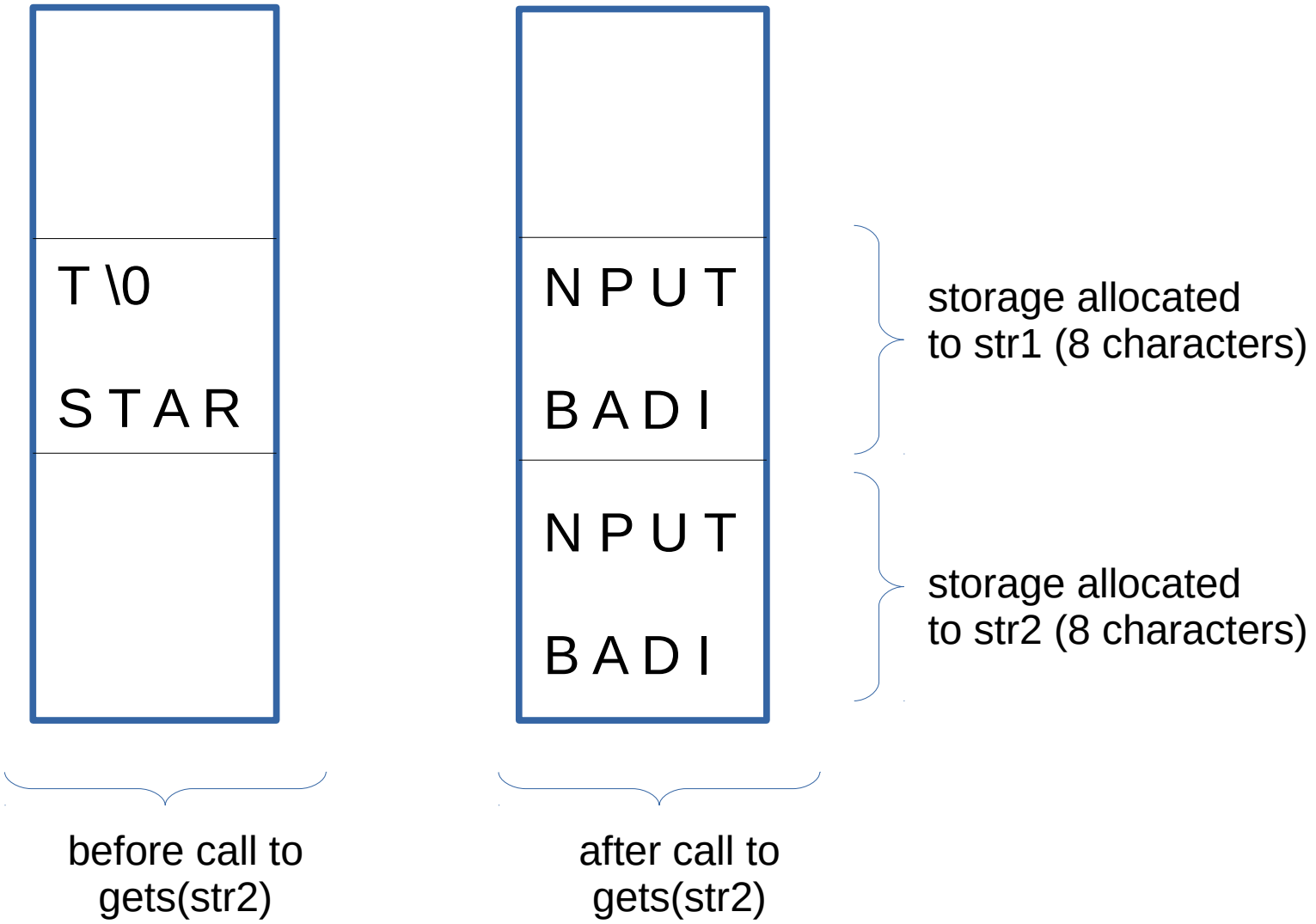
test case	**str1 (input / final)**	**str2 (input / final)**	**valid (final)**
ex1	START / START	STARTXX / STARTXX	0
ex2	START / X	STARTXXXX / STARTXXXX	0
ex3	START / XX	XXXXXXXXXX / XXXXXXXXXX	0

Step 4: Why is the C program not doing what it is supposed to?

Because the value of str2 overflows into str1



Step 4: Why is the C program not doing what it is supposed to? (cont.)



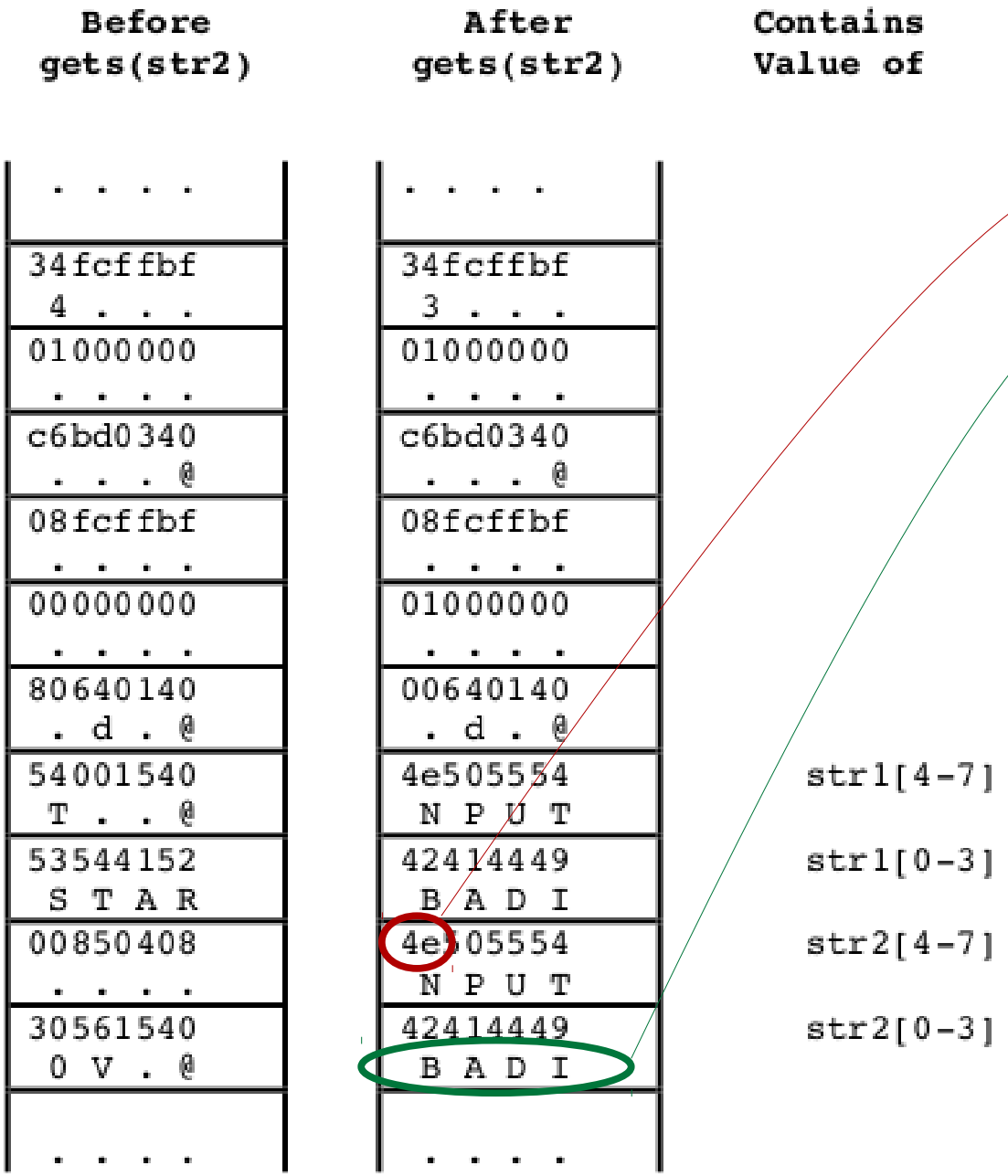
Step 4: details of Fig. 10.2

Before gets(str2)	After gets(str2)	Contains Value of
. . . .	. . . .	
34fcffbf 4 . . .	34fcffbf 3 . . .	
01000000	01000000	
c6bd0340 . . . @	c6bd0340 . . . @	
08fcffbf	08fcffbf	
00000000	01000000	
80640140 . d . @	00640140 . d . @	
54001540 T . . @	4e505554 N P U T	str1[4-7]
53544152 S T A R	42414449 B A D I	str1[0-3]
00850408	4e505554 N P U T	str2[4-7]
30561540 0 V . @	42414449 B A D I	str2[0-3]
. . . .	. . . .	

42:
- hexadecimal number
- uses 16 symbols
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f,

42_{16}
= 4*16 + 2 (in decimal)
= 66 (in decimal)
= ASCII value of ‘B’

Exercise

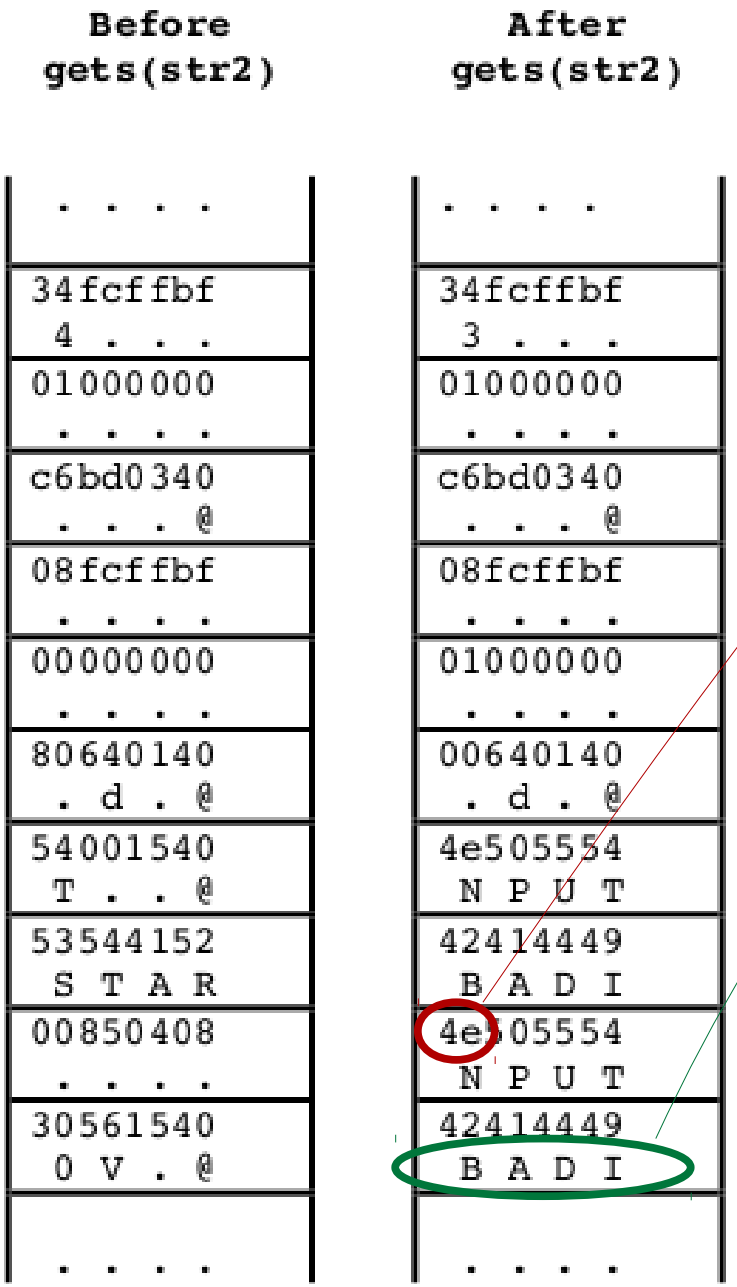


What does 4e mean?

How many bits in a memory word?

- such as B A D I
- with a separate memory address

Exercise solution



What does 4e mean?

4e is in hexadecimal notation

$4e_{16}$
*= 4*16 + 14 in decimal*
= 78 in decimal
= ASCII value of 'N'

str1[4-7]

str1[0-3]

str2[4-7]

str2[0-3]

How many bits in a memory word?

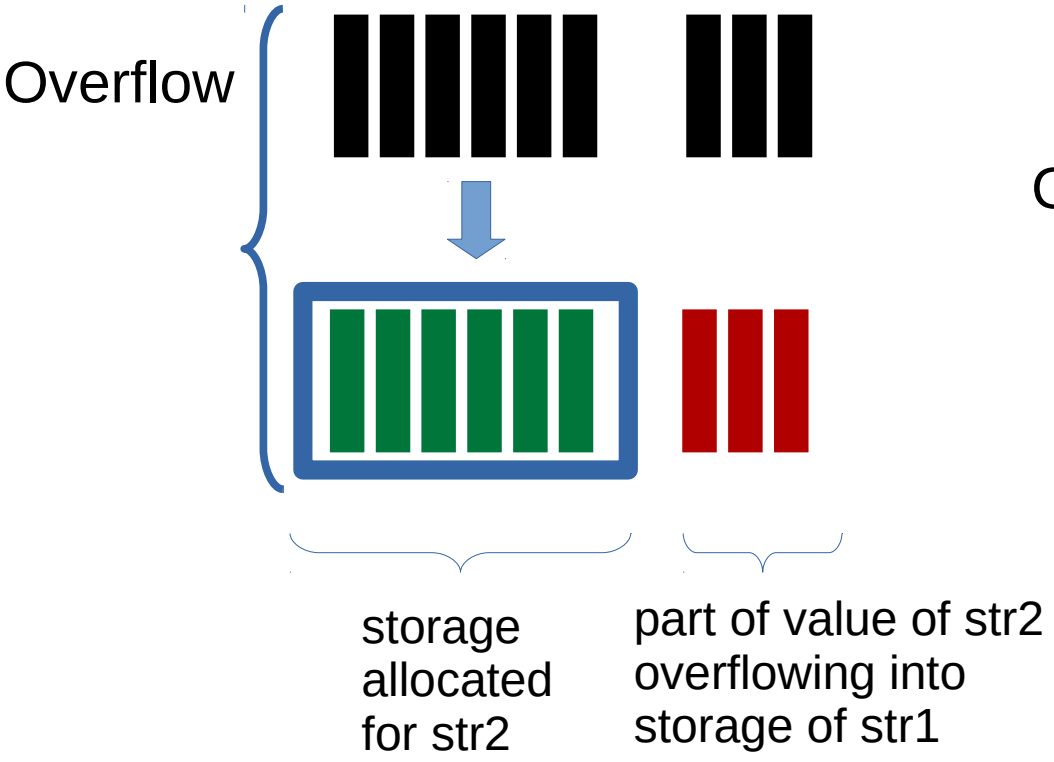
- such as B A D I
- with a separate memory address

One word = eight hexadecimals

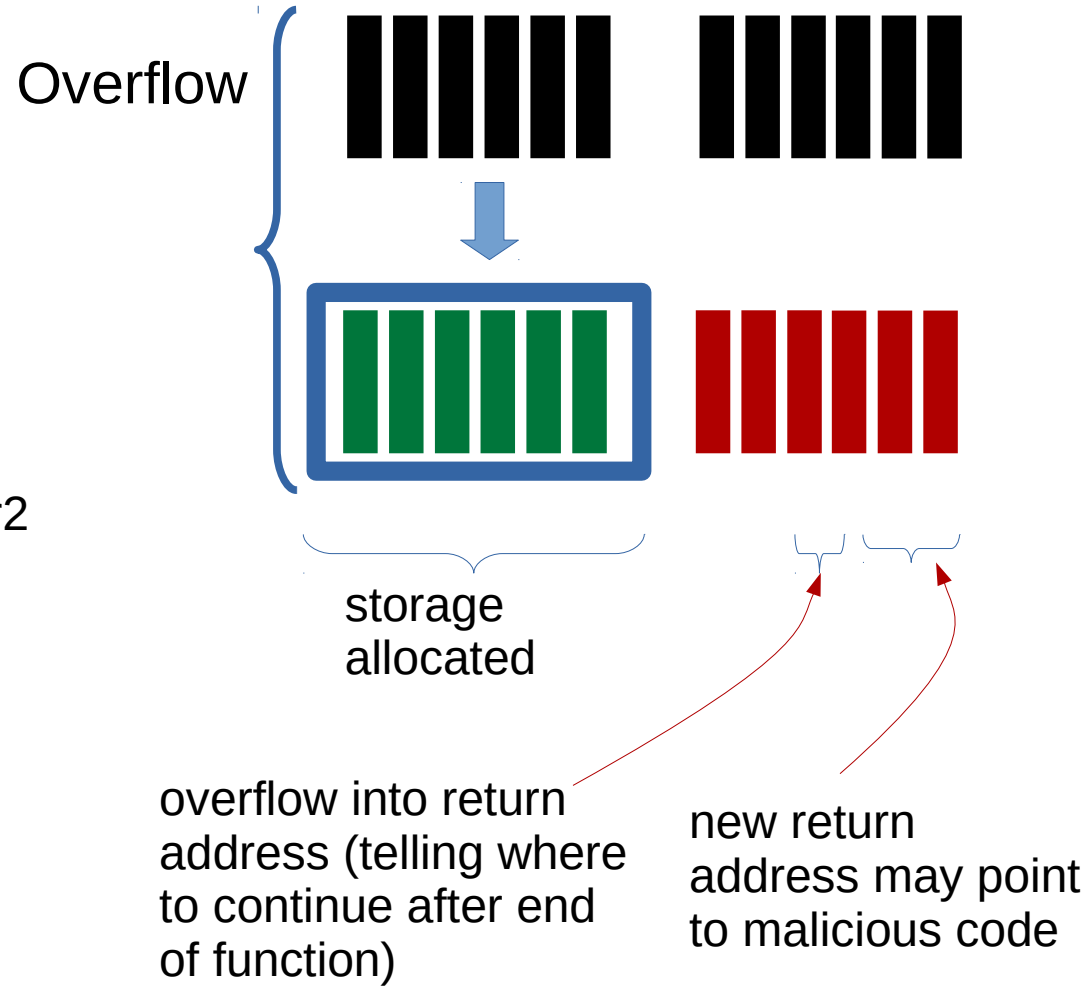
One hexadecimal ~ 4 bits
Thus, eight hexadecimals ~ 32 bits

Buffer overflows with malicious code

The program buffer1 contains a buffer overflow which flows into data storage



Buffer overflows may contain malicious code as well



Plan for today

Buffer overflows: introduction

Code Red (Anton)

Buffer overflows: how they work

Buffer overflows: prevention

Cloud security

IoT security

Any internet service must prevent buffer overflows

Servers
- web
- file servers
- “telephone books”
- ..

Clients

Exercise

What are the major approaches to prevent buffer overflows?

Exercise solution

What are the major approaches to prevention of buffer overflows?

1) Choose a safe programming language
- a language that checks array bounds
- eg., Java

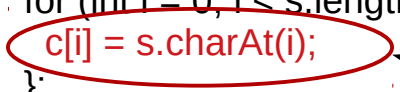
2) Hardening of existing programs written in eg., C
- by inspecting and rewriting the programs

3) Guidelines for safe use of C
- do not use unsafe system-defined functions, such as gets()
- programmer should always check that sufficient storage is allocated
- ..

4) Remove library of unsafe system-defined functions

Java has array bounds checking

```
import java.util.Scanner;

public class Buffer1 {

  public static void main(String[] args) {
    int valid = 0;
    char[] str1 = new char[8];
    char[] str2 = new char[8];
    next_tag(str1);
    gets(str2);

    String s1 = new String(str1);
    String s2 = new String(str2);
    if (s1.equals(s2)) valid = 1;
    System.out.println("buffer1: "
                       + "str1(" + s1 + "), "
                       + "str2(" + s2 + "), "
                       + "valid(" + valid + ")");
  }

  static void next_tag(char[] c) {
    c[0] = 'S';
    c[1] = 'T';
    c[2] = 'A';
    c[3] = 'R';
    c[4] = 'T';
  }

  static void gets(char[] c) {
    Scanner scanner = new Scanner(System. in);
    String s = scanner.nextLine();
    for (int i = 0; i < s.length(); i++) {
      c[i] = s.charAt(i);
    };
  }
}
```



The execution of

c[i] = s.charAt(i)

aborts if i is 8 or higher,
because Java checks i against
the array bounds 0 and 7

Exercise

Would it be a good preventive measure against buffer overflows to stop using languages such as C?

Exercise solutions

Would it be a good preventive measure against buffer overflows to stop using languages such as C?

There are also advantages to the use of C

- much more efficient
  - uses less time and space
- more flexible
- most operating systems written in C
  - Unix
  - Mac
  - Windows

Plan for today

Buffer overflows: introduction

Code Red (Anton)

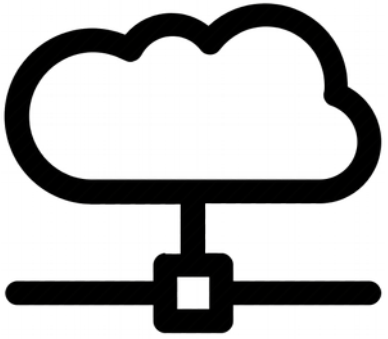
Buffer overflows: how they work

Buffer overflows: prevention

Cloud security

IoT security

Cloud and IoT security



Both cloud and IoT security
- new areas
- with well-known technical solutions
  - encryption (storage, transmission)
  - authentication

Chapter focuses on general concepts
- helps us identify what are main challenges?
- few details, dilemmas, real-life problems

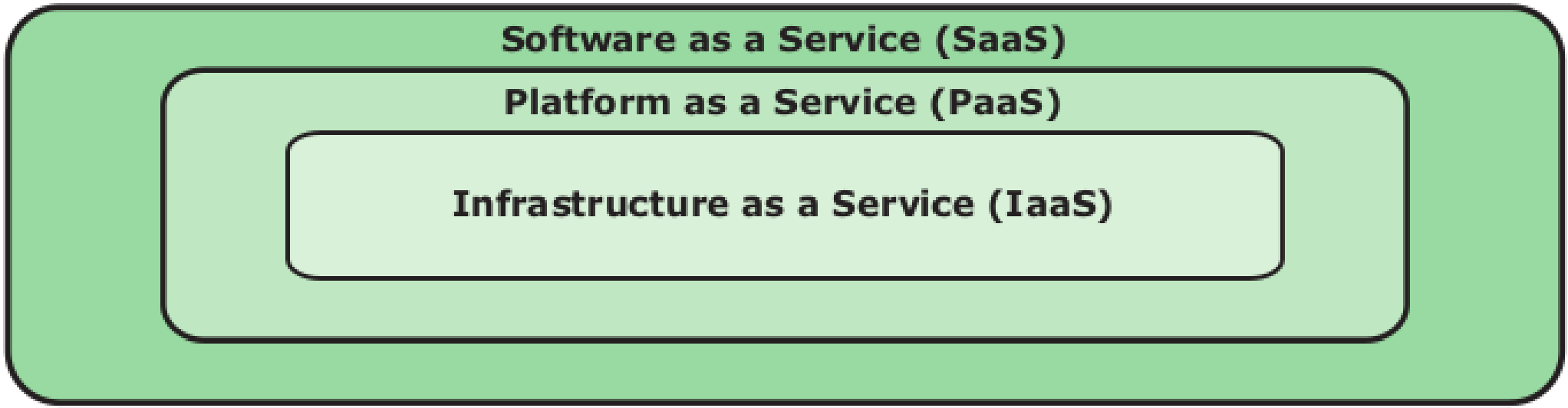
Example: should RUC use cloud services?

RUC's IT systems include:
- mail
- moodle (information about courses)
- PCs of employees (including text processing)

RUC's IT department:
- maintain servers and software for mail, moodle, ..
- provide service for employees' PCs

Approximately 40 people

Service models (types of services)

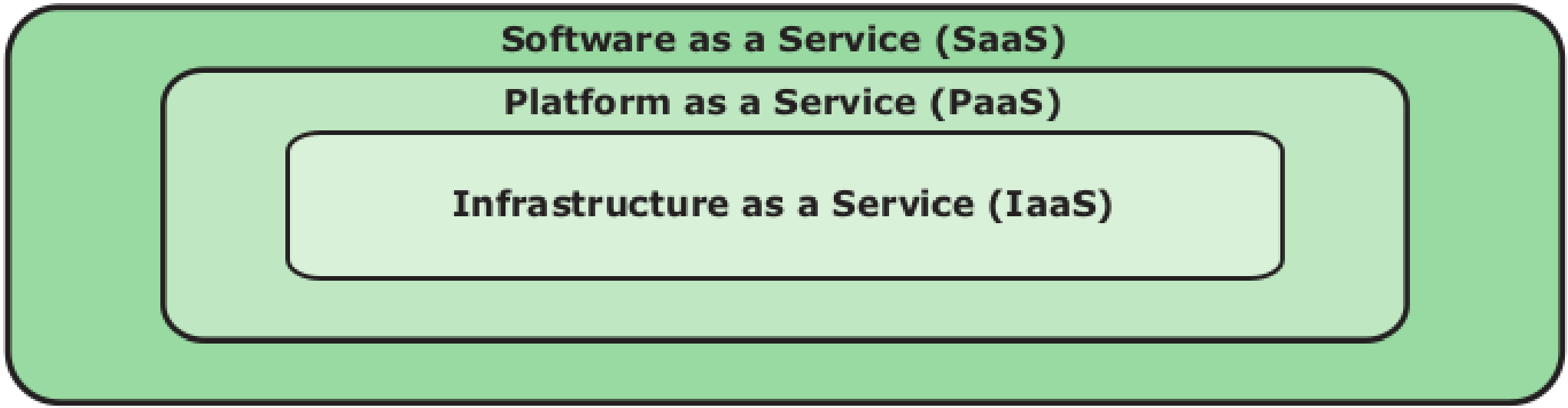


SaaS
- client may do word processing, email in cloud

PaaS
- client may run specific applications in cloud

IaaS
- client gets access to machines (processing, storage) in cloud

Exercise: what types of services may be relevant to RUC?

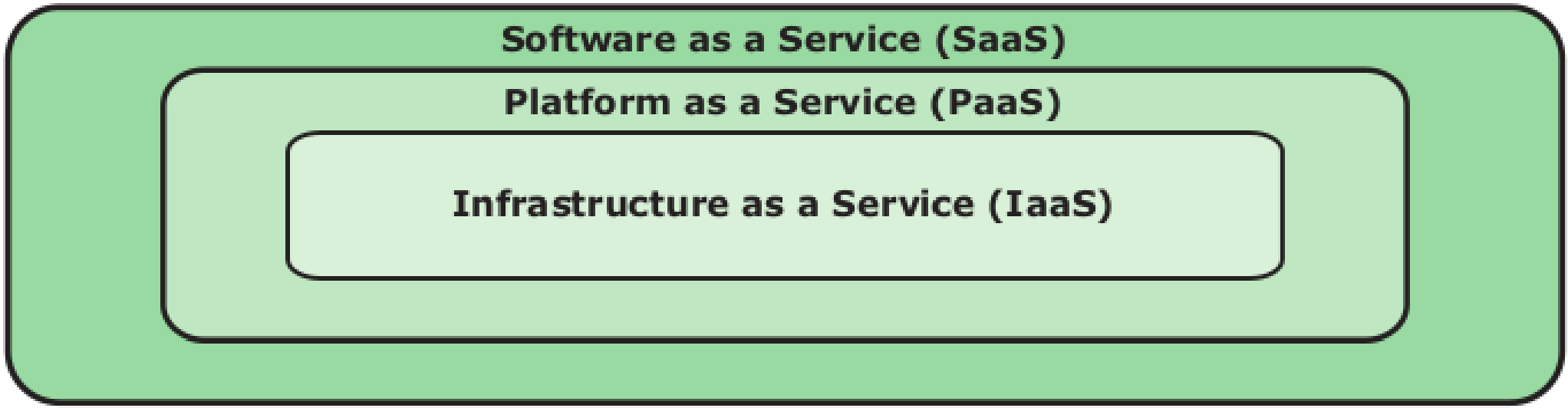


SaaS
- client may do word processing, email in cloud

PaaS
- client may run specific applications in cloud

IaaS
- client gets access to machines (processing, storage) in cloud

Exercise solution

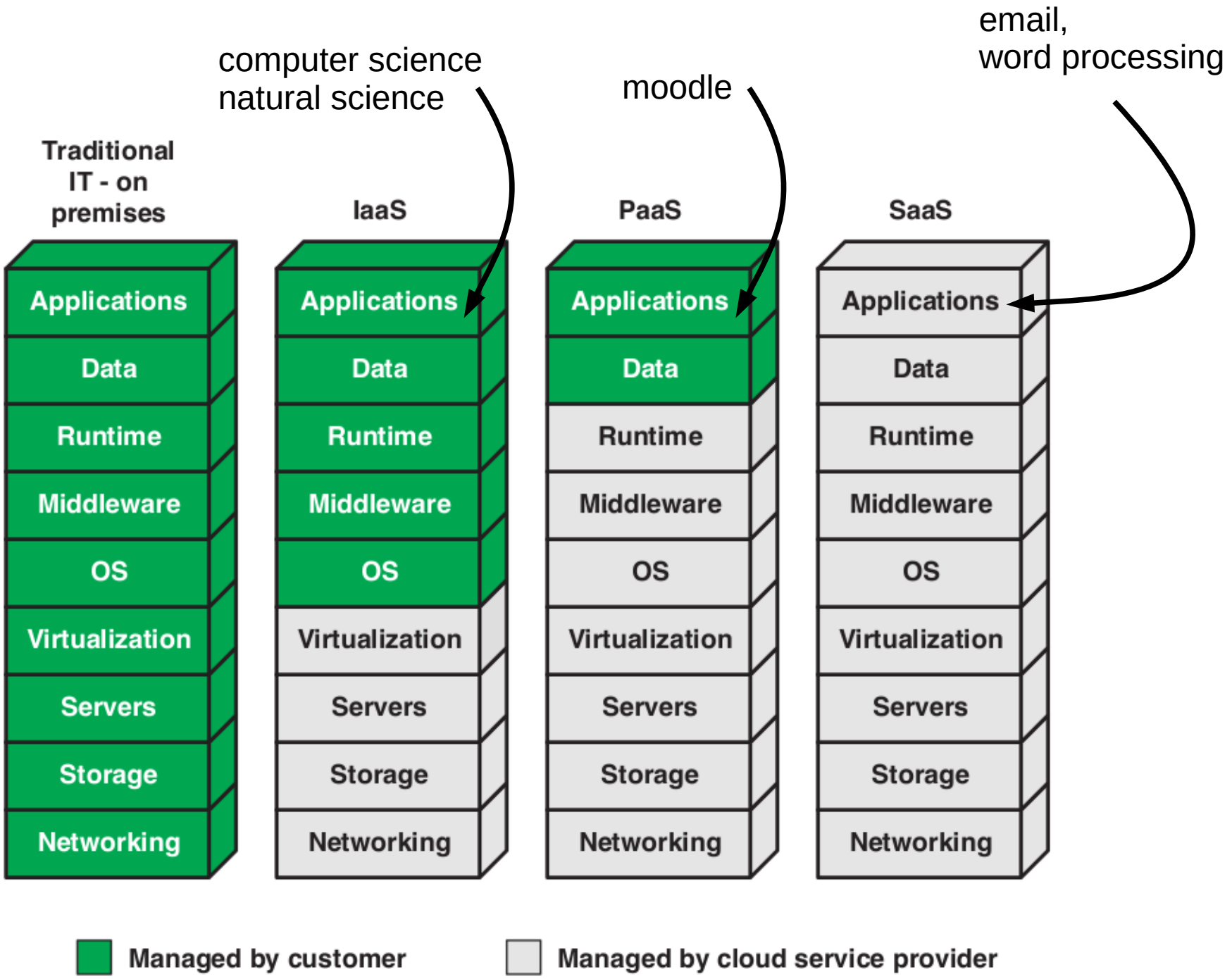


SaaS
- client may do word processing, email in cloud
- relevant: most RUC employees use word processing and email

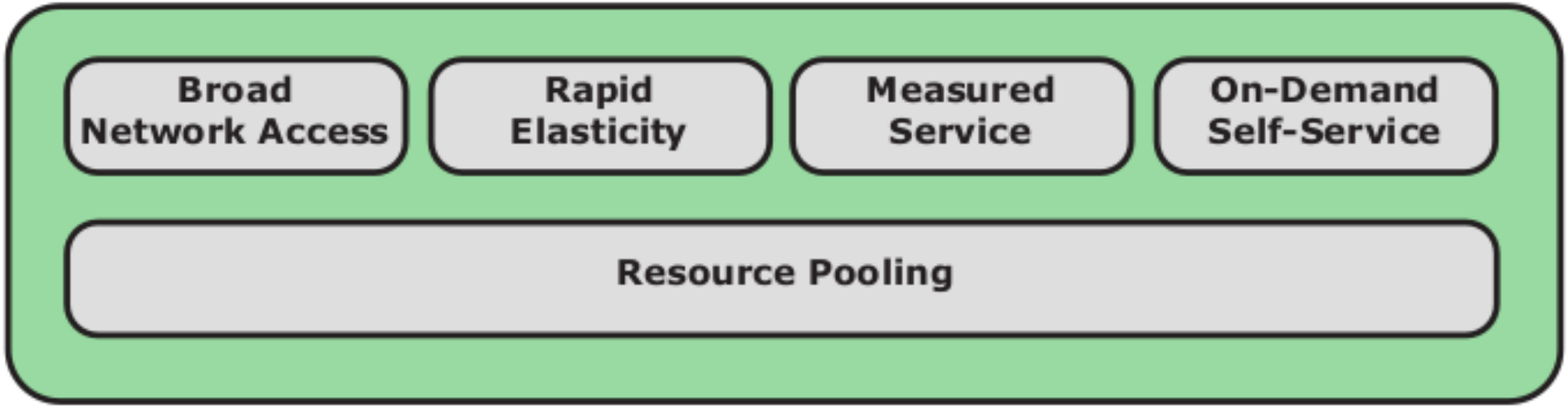
PaaS
- client may run specific applications in cloud
- relevant: RUC runs specific applications such as moodle

IaaS
- client gets access to machines (processing, storage) in cloud
- not highly relevant: only employees at computer sc. and natural sc. develop own applications

Service models



Characteristics (benefits) of cloud computing



Broad network access
- RUC employees, students may access resources from laptops, mobiles,..

Rapid elasticity
- moodle scale-up at start of semester, ..

Measured service
- (in-house service can be measured/metered just as well)

On-demand self-service, resource pooling
- (this is how rapid elasticity is achieved)

Cloud security

“.. while the enterprise loses a substantial amount of control over resources, services, and applications, it must maintain accountability for security and privacy policies” (Stallings & Brown, p 457)

CSA categories of cloud security

Stallings & Brown, p 461-462

A category of cloud security is a type of service that may be offered by a cloud service provider (CSP)

Let’s run through the categories, with the RUC example in mind.

CSA categories of cloud security

Identity and access management
- CSP must provide access control to word processing and email resourcers

Data loss prevention
- CSP must secure that documents and emails are not lost
- CSC may encrypt documents, emails to prevent leakage

Web security (security of web browsing)
- CSP may provide added protection against, eg. web viruses
- eg., in Adobe Flash Player

E-mail security
- similarly, CSP may provided added protection against email threasts
- eg., spam, phishing

Security assessmes
- third party audits

CSA categories of cloud security (cont.)

Exercise: please explain the following categories (possibly by example)

Intrusion management

Security information and event management

Encryption

Business continuity and disaster recovery

Network security

Exercise solution (partial)

Intrusion management
- i. detection, prevention, response
- same as in enterprises own network
- CSP may be more competent/professional

Security information and event management
- monitoring and reporting intrusion and other events

Encryption
- if provided by CSP, entails complex key management because clients must be able to decrypt

Business continuity and disaster recovery
- CSP may provide better backup
- eg., multiple locations

Network security
- firewalls, protection against (D)DoS, ..
- again CSP may be more comptent/professional

Plan for today

Buffer overflows: introduction

Code Red (Anton)

Buffer overflows: how they work

Buffer overflows: prevention

Cloud security

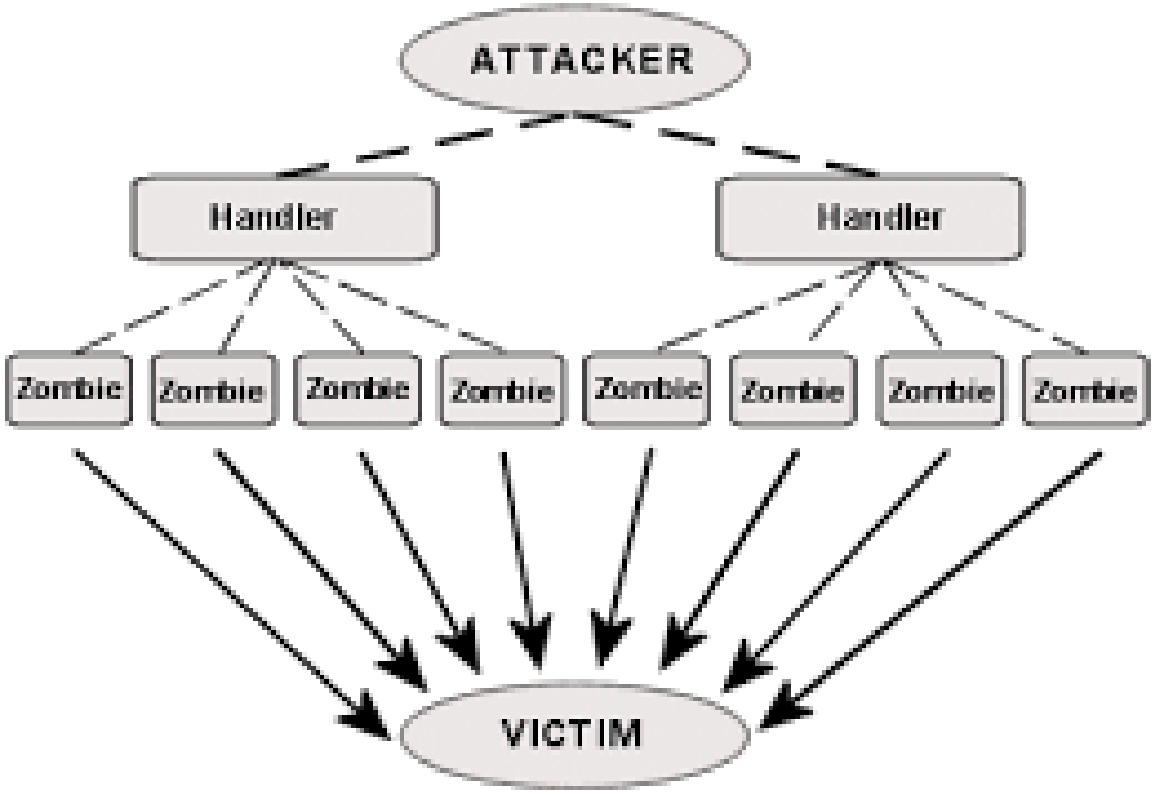
IoT security

IoT security

“IoT is perhaps the most complex and undeveloped area of network security.”

Stallings & Brown p 471

The October 2016 webcam hack: a DDoS attack



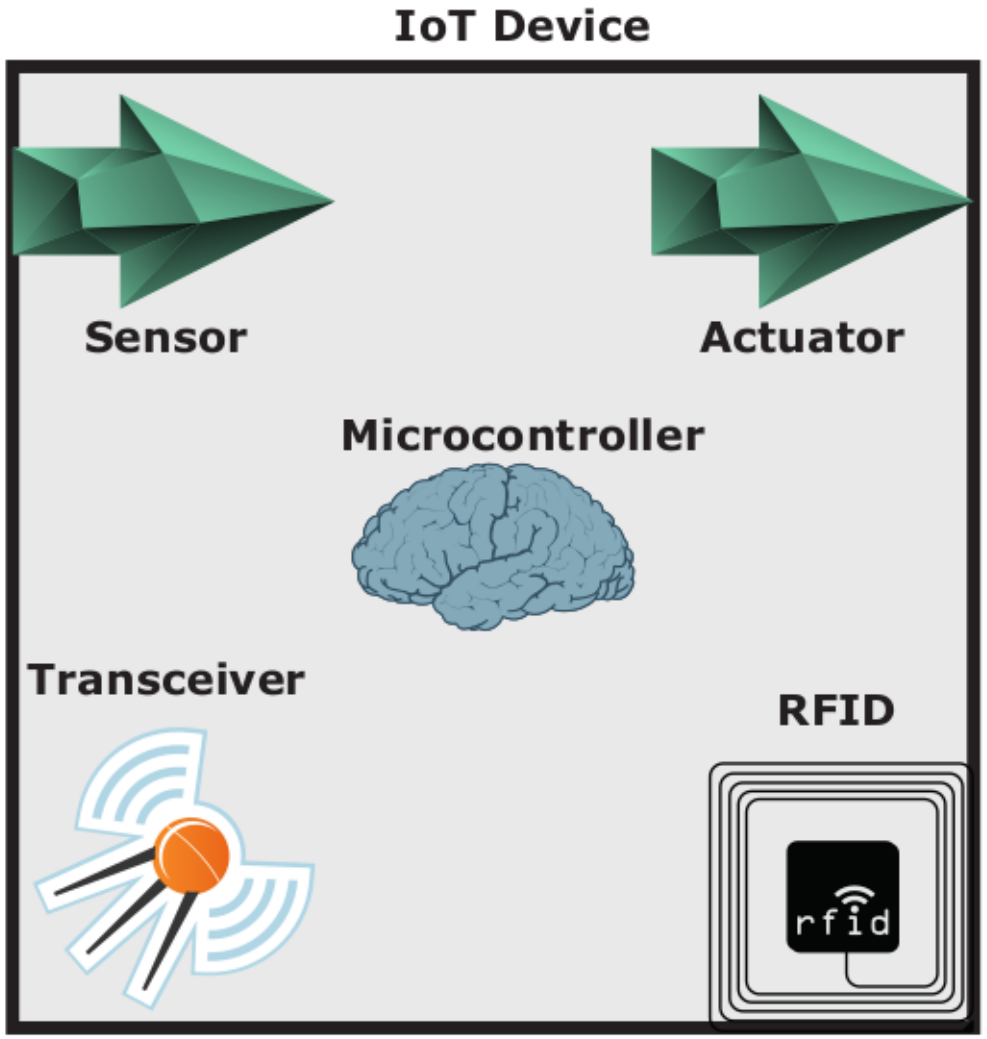
The hack targetted DYN
- a Internet/DNS service provider (for Twitter and many others)

The hack compromised
- webcams (IoT devices)
- not ordinary computers

Malicious software: Mirai
- attacks linux-based cameras
- worm: scans for more cameras

Mirai exploited..
- default usernames/passwords
- some devices can not be updated
- so can not change the default usernames/passwords

IoT devices



Webcam as en IoT example

- sensor
  - the camera
- actuator
  - moving camera angle
- microcontroller
  - may pause camera
- transceiver (internet connection)
  - WIFI
  - IP adress

We will assume webcam has
- unprotected logon
- permits upload/execute programs

How to protect against unprotected IoT devices?

(1) Your own IoT devices

- use IoT devices where you can change the password
- and where software can be updated
- put IoT devices behind a gateway
- this protects your privacy etc.

(2) Your neighbor's IoT devices

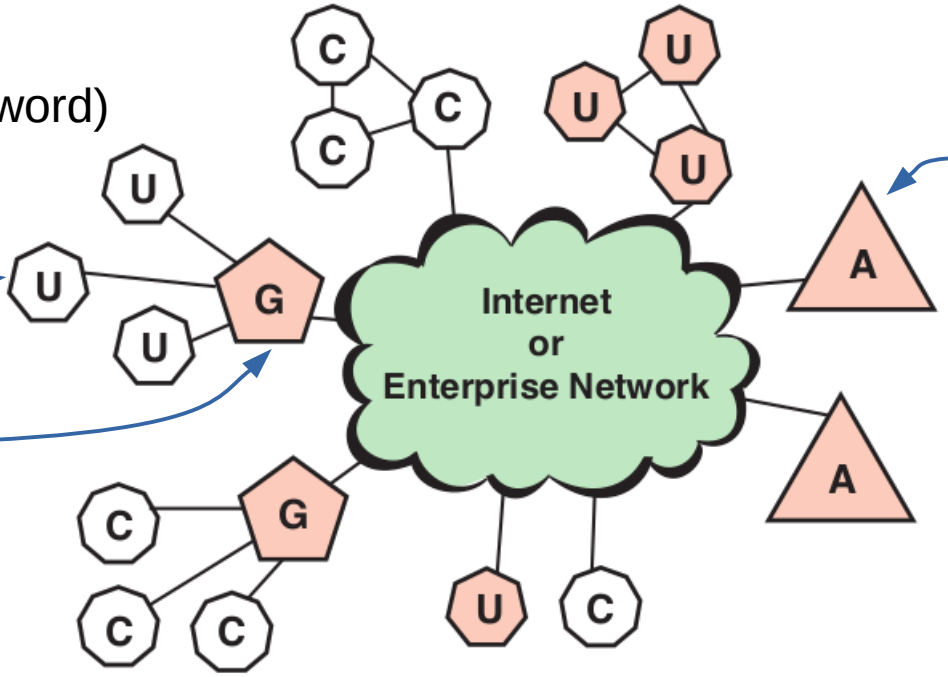
> *"Even if we purchase better designed products, our neighbours who don't will have their devices hacked and used against the rest of us... The only approach to solving this is .. the designers of these products take their responsibilities seriously".*
>
> ("Webcam hack shows vulnerability of connected devices", p.10).

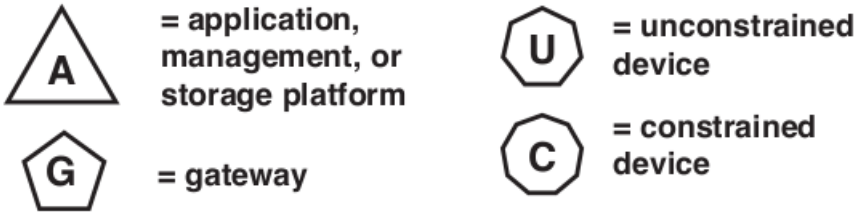
IoT network

camera assumed to be
- "unconstrained" (general cpu capabilities)
- no shading = no security (eg. default username/password)

A gateway
- controlling access to video cameras

Database
- stores videos

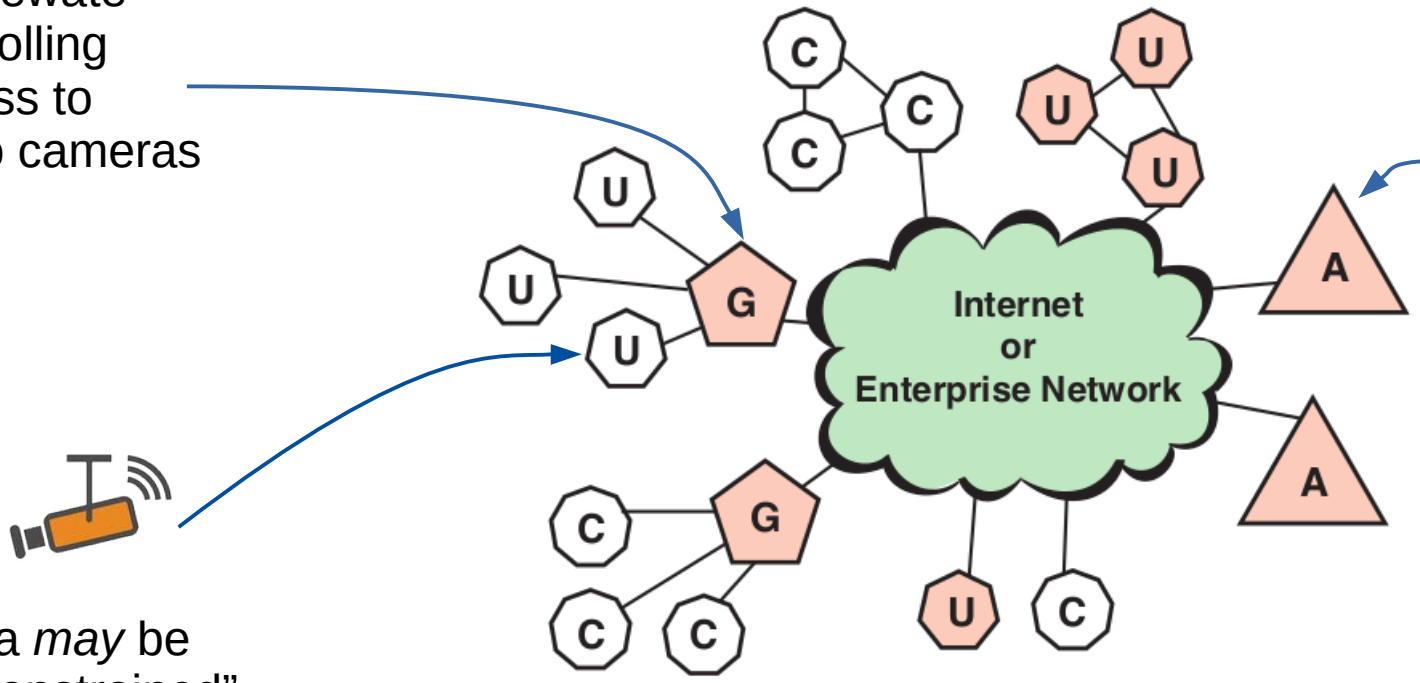




shading = includes security features

IoT network

A gatewate controlling access to video cameras



Database
- storing videos

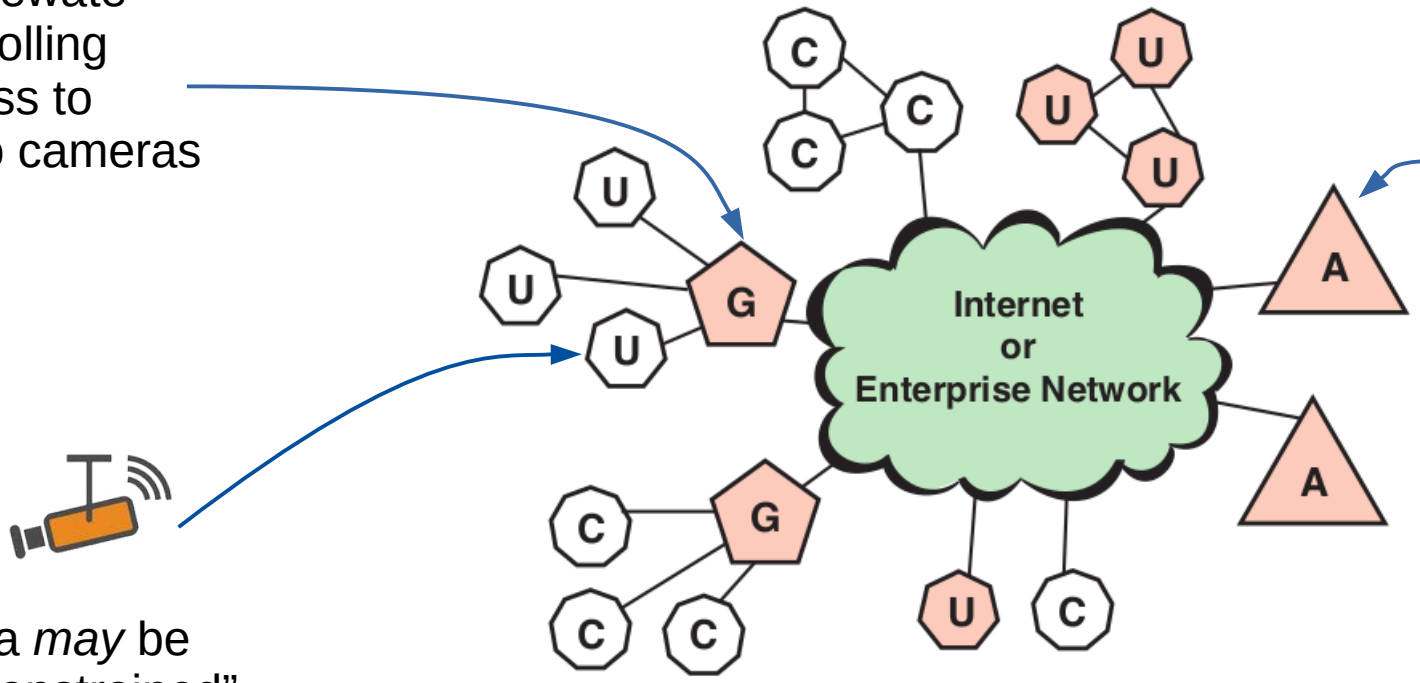
camera *may* be
- "unconstrained" (general cpu capabilities)
- no security (default username/password)

What functions would we like the gateway to provide?
(assuming cameras are unconstrained and have no security)

- see p 472-473 on ITU-T's list of gateway security functions

IoT network

A gatewate controlling access to video cameras



Database
- storing videos

camera *may* be
- "unconstrained" (general cpu capabilities)
- no security (default username/password)

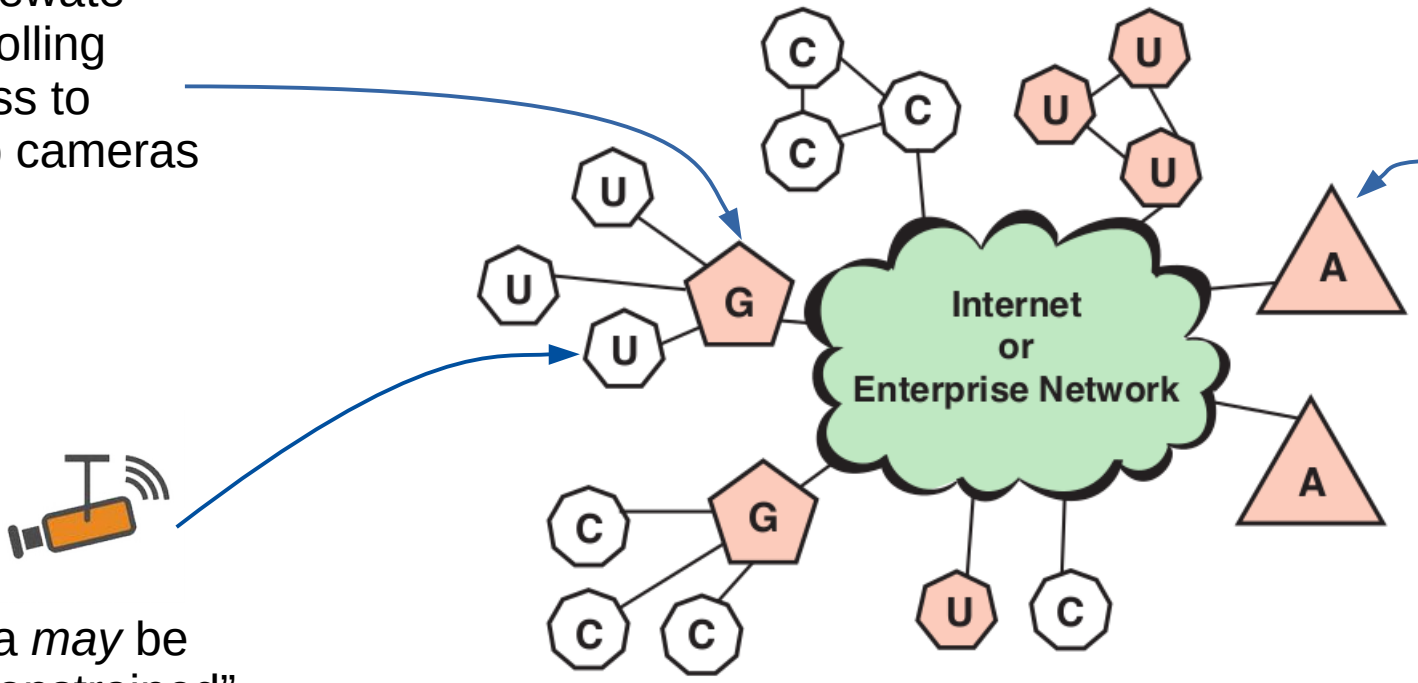
What functions would we like the gateway to provide?
(assuming cameras are unconstrained and have no security)

- identification (logging): who has accessed the cameras?
- access control #1 (authenticate users who access camera)
- access control #2 (authenticate camera) (not highly relevant)

Exercise: please complete the list

Exercise solution

A gatewate controlling access to video cameras



Database
- storing videos

camera *may* be
- "unconstrained" (general cpu capabilities)
- no security (default username/password)

Exercise: what functions would we like the gateway to provide? (assuming camera is unconstrained and without own security)

- identification (logging): who has accessed the cameras?
- access control #1 (authenticate users who access camera)
- access control #2 (authenticate camera) (not highly relevant)
- encrypt data in transmission (we can't encrypt data in the cam.)
- support self-diagnosis (eg., no recording from camera)
- support firm/soft-ware update
- auto configuration: gateway puts cameras in "pause mode"