Network Programming

Program Security

Note: **This class lecture will be recorded!**
If you do not consent to this recording, please do not ask questions via your video, audio or public chat; send your question to the instructor using the private chat.

Lisa Frye, Instructor
frye@Kutztown.edu
Kutztown University

Secure Program

- What is meant by the statement “Program x is secure”?
- Software Quality
- Program Faults

Which one is more Secure?

- 100 faults were discovered and fixed
- 20 faults were discovered and fixed

IEEE Terminology

- IEEE has a standard for software engineering terminology (610.12, 1990)

- ***Quality*** – The degree to which a system, component, or process meets specified requirements
- ***Failure*** – The inability of a system or component to perform its required functions within specified performance requirements
- ***Fault*** – An incorrect step, process, or data definition in a computer program

Software Quality Control

- Software Quality Assurance (SQA)
- Secure Software Development
  - Secure SDLC
- Software Reverse Engineering
  - What is it?
  - Benefits?
- Whose responsibility is to ensure software is secure in a corporate setting?

OWASP

- https://www.owasp.org/
- https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

- 2019 Top Five
- 2017
- 2013

Program Patches

- ***Penetrate and Patch*** paradigm
- Program patches often led to a less secure program.
- Why?

- Pressure to correct quickly
- Nonobvious side effects
- Performance consequences

Program Flaws

- Requirements vs. actual behavior
- ***Program security flaw***

- Vulnerabilities and Flaws Views
  - Cause
  - Effect

- Patching and Analyzing program behavior are not always the best approach

Taxonomy of Program Flaws

- Inadvertent human errors
- Malicious, intentionally introduced flaws
  - Malicious
  - Non-Malicious
    - Covert channel
    - Other

❖ What types of program flaws do you think cause more damage?

Inadvertent Flaws

- Validation error (incomplete or inconsistent)
- Domain error
- Serialization and aliasing
- Inadequate identification and authentication
- Boundary condition violation
- Other exploitable logic errors

Buffer Overflow

- Buffer
- Buffer Overflow

- char buf[10];
  buf[10] = ‘A’;
- char *buf;

Buffer Overflow Example

- 8-byte string (empty) and 2-byte integer (3)

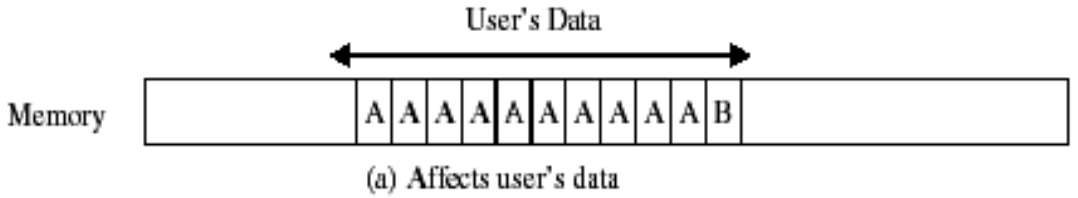
A	A	A	A	A	A	A	A	B	B
0	0	0	0	0	0	0	0	0	3

A	A	A	A	A	A	A	A	B	B
'e'	'x'	'c'	'e'	's'	's'	'i'	'v'	'e'	0

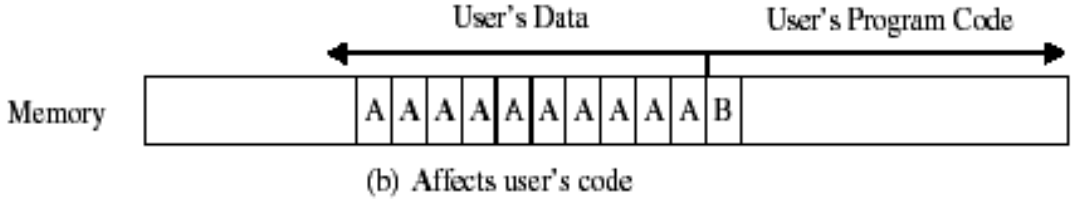
Another Buffer Overflow Example

```
char sample[10];
for (i=0; i<10; i++) sample[i] = ‘A’;
sample[10] = ‘B’;
```

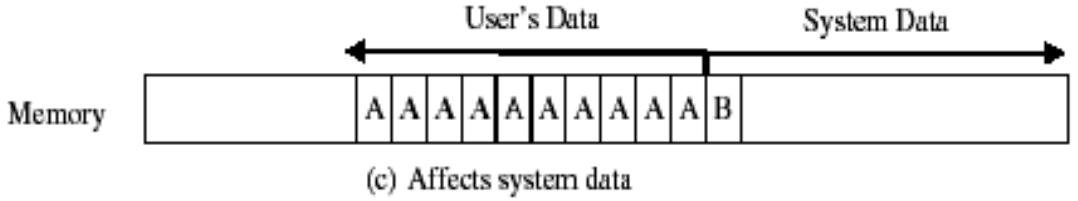
- Where does the last assignment (‘B’) go?



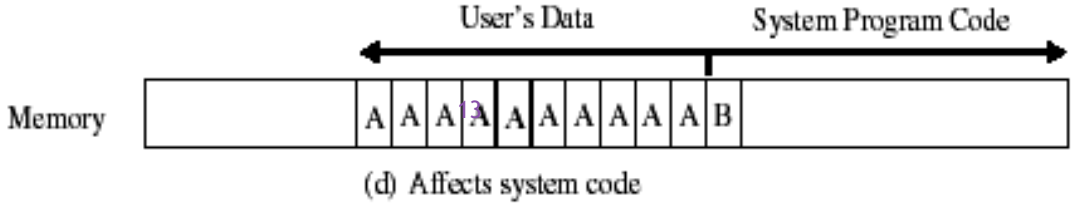
(a) Affects user's data



(b) Affects user's code



(c) Affects system data



(d) Affects system code

From: “Security in Computing”, chapter 3, by Pfleeger and Pfleeger

Stack-Based Buffer Overflow Example

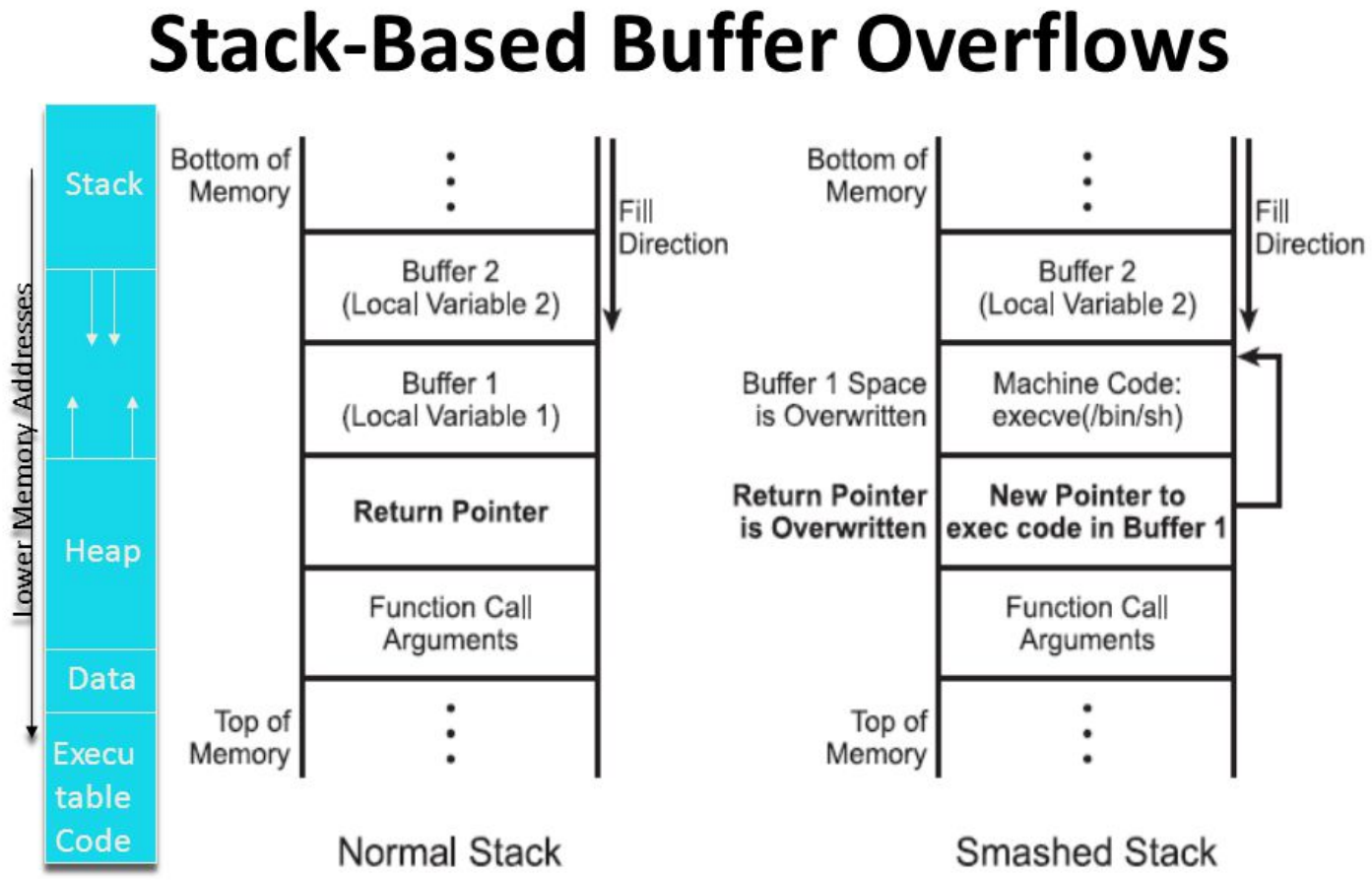


EXHIBIT 10.2 A normal stack and a stack with a buffer overflow.

https://www.mkdynamics.net/current_projects/computer_security/Basic_Linux_exploits.html

C String Functions

- strcpy
- Strncpy

Common C string functions to use and avoid	
Don’t use these functions	Use these instead
strcat	strlcat
strcpy	strlcpy
strncat	strlcat
strncpy	strlcpy
sprintf	snprintf (see note) or asprintf
vsprintf	vsnprintf (see note) or vasprintf
gets	fgets (see note) or use Core Foundation or Foundation APIs

Preventing & Detecting Buffer Overflow

- Address Space Layout Randomization (ASLR)
- No stack or heap execution

- Test your Programs!!!

Buffer Underflow

- What is Buffer Underflow
- Is this a problem (is it dangerous)?
- Two Types
  - Short Write
  - Short Read
- CVE – Common Vulnerabilities and Exposures

CVE – Common Vulnerabilities and Exposures

- https://cve.mitre.org
- CVE Details
  - https://www.cvedetails.com/index.php
- OpenSLL Vulnerabilities
  - https://www.cvedetails.com/vulnerability-list.php?vendor_id=217&product_id=0&version_id=0&page=1&hasexp=0&opdos=0&opec=0&opov=0&opcsrf=0&opgpriv=0&opsqli=0&opxss=0&opdirt=0&opmemc=0&ophttprs=0&opbyp=0&opfileinc=0&opginf=0&cvssscoremin=0&cvssscoremax=0&year=0&month=0&cweid=0&order=3&trc=85&sha=d709ee3c0dc47c3827b5990023842398148d082b

Underflow Error

- What is an underflow error?

- http://hmarco.org/bugs/CVE-2015-8370-Grub2-authentication-bypass.html

Unchecked Data Values

- http://www.somesite.com/page/userinput&param=2004Jun1
- Possible values for param
  - 2004Jun1
  - 1800Jan1
  - 2015Feb30
  - 2000Min5

- What would be the results of these input values?

Security Implications - Example

- Things, Inc. - very large, international vendor of consumer products: Objects
- The company was ready to sell its Objects through a web site
- To Things developed a complete price list of its Objects
- For example, a customer on the web could choose to buy 20 of part number 555A Objects. If the price of one such part were $10, the web server would correctly compute the price of the 20 parts to be $200.
- Then the customer could decide whether to have the Objects shipped by boat, by ground transportation, or sent electronically.
- http://www.things.com/order/final&custID=101&part=555A &qy=20&price=10&ship=boat&shipcost=5&total=205

From “Security in Computing”, chapter 3, by Pfleeger and Pfleeger

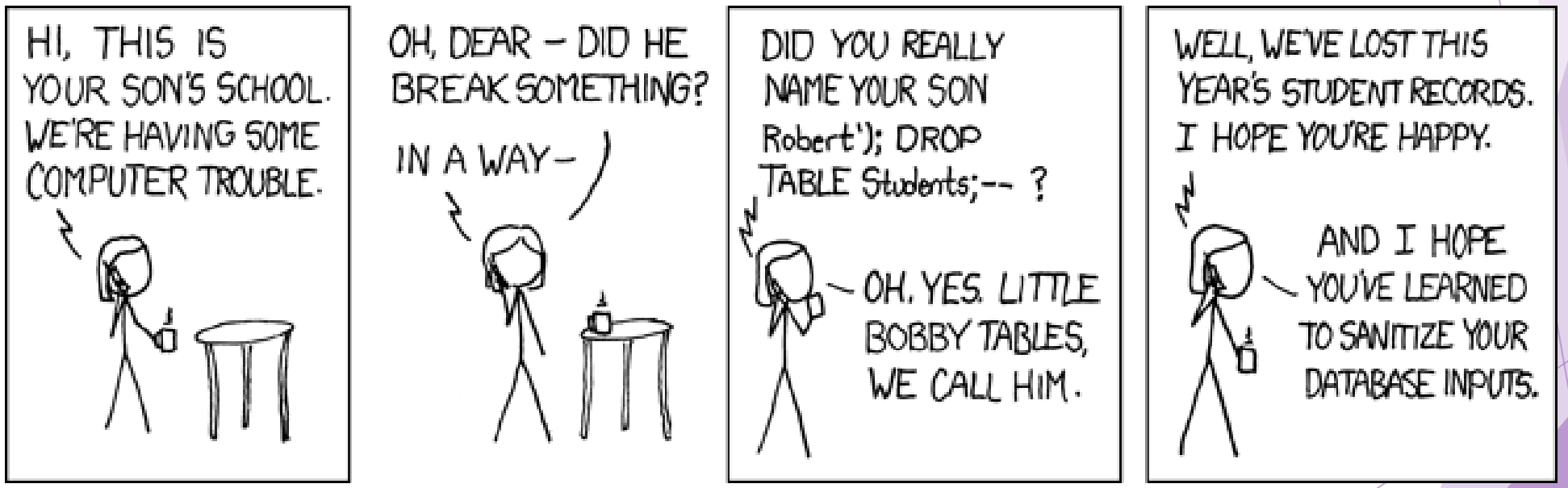
Security Flaw in Example

- What is the security flaw in the Things, Inc. example?

- http://www.things.com/order/final&custID=101&part=555A&qy=20&price=1&ship=boat&shipcost=5&total

SQL Injection

- What is SQL Injection?



http://xkcd.com/327

SQL Injection Example 1

- userID = getRequestString("UserId");
  SQL = "SELECT * FROM Users WHERE UserID = " + userID

- User enters ‘frye’

- User enters ‘asdf or 1=1’

SQL Injection Example 2

- productID = getRequestString(“ProductID”);

  SQL = “SELECT * from Products Where ProductID = “ + productID;

- User enters ‘abc’

- User enters ‘abc; DROP TABLE Products’

SQL Injection Example 3

```
$sql = "INSERT INTO Customers " .
          "(Email, Passwd, FirstName, LastName, Address1, “ .
          “Address2, City, State, ZipCode, PhoneNumber) " .
      "VALUES ('" . $_POST['email'] . "', '" .
                    $_POST['password'] . "', '" .
                    $_POST['fname'] . "', '" .
                    $_POST['lname'] . "', '" .
                    $_POST['street'] . "', '" .
                    $_POST['street2'] . "', '" .
                    $_POST['city'] . "', '" .
                    $_POST['state'] . "', " .
                    $_POST['zip'] . ", '" .
                  $phone . "')";
$result = mysql_query($sql);
```

Example 3 Data Input

- INSERT INTO Customers (Email, Passwd, FirstName, LastName, Address1, Address2, City, State, ZipCode, PhoneNumber) VALUES ('I',‘Got', You','Good',':)',",':)); TRUNCATE TABLE Customers;\"',':)',':)',")

- What was the result?

Design SQL Injection Prevention Code

- userID = getRequestString("UserId");
  SQL = "SELECT * FROM Users WHERE UserID = " + userID

- productID = getRequestString(“ProductID”);

  SQL = “SELECT * from Products Where ProductID = “ + productID;

Prevent SQL Injection

- Blacklist
- SQL Parameters
  - Values added to SQL query at execution
  - @
- userID = getRequestString("UserId");
  SQL = "SELECT * FROM Users WHERE userID = @0";
  db.Execute(SQL, userID);

SQL Parameters Example

```
txtNam = getRequestString("CustomerName");
txtAdd = getRequestString("Address");
txtCit = getRequestString("City");
txtSQL = "INSERT INTO Customers
        (CustomerName,Address,City) Values(@0,@1,@2)";
db.Execute(txtSQL,txtNam,txtAdd,txtCit);
```

From W3 Schools (http://www.w3schools.com/sql/sql_injection.asp)

Other Program Threats

- Trapdoor
- Salami Attack
- Covert Channels

Controls Against Program Threats

- Peer reviews
- Encapsulation and Information Hiding
- Independent Testing
- Configuration Management