SOFTWARE SECURITY - I

Pawan Bhandari

```
#include<stdio.h>
#include<string.h>
int main(int argc, char *argv[])
{
    if(1 < argc)
    {
        printf("Incorrect number of input
parametrs\n");
        return 1;
    }

    const char passwd[10] = “passwd”;
    char input[16];

    while(1)
    {
        printf("Enter password: ");
        scanf("%s",input);
        if(0 == strcmp(passwd,input))
            break;
    }
    printf("Voila!\n");
    return 0;
}
```

Any

Issue

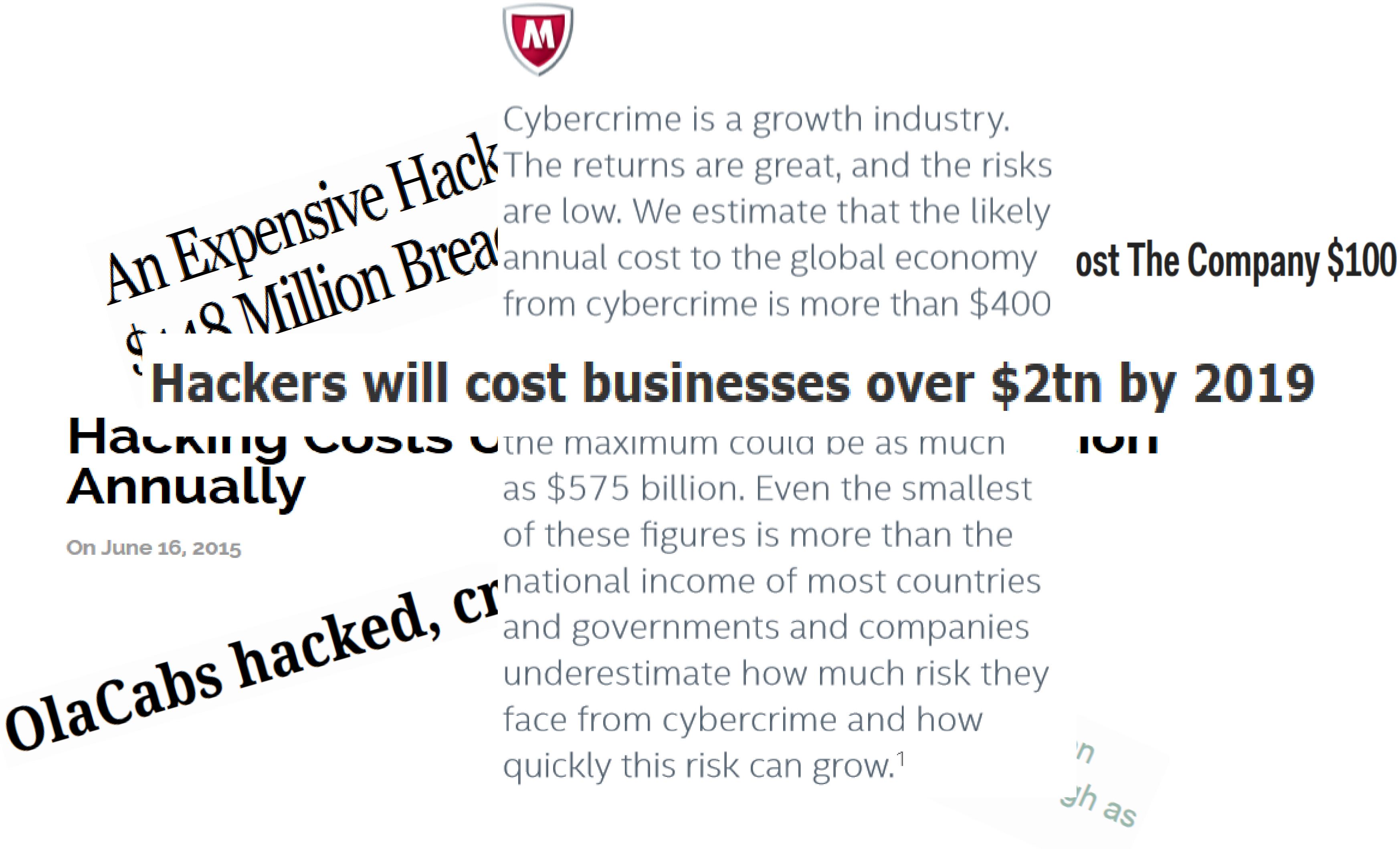
Here?

Agenda

- Introduction
- Programming Language
- Buffer Overflow
- Data Issues
- Let’s Defense
- Web Vulnerabilities
- Future Pointers

INTRODUCTION

What is at Stake



An Expensive Hack

$ 48 Million Brea

Cybercrime is a growth industry. The returns are great, and the risks are low. We estimate that the likely annual cost to the global economy from cybercrime is more than $400

ost The Company $100

Hackers will cost businesses over $2tn by 2019

Hacking Costs C ion

Annually

On June 16, 2015

the maximum could be as much as $575 billion. Even the smallest of these figures is more than the national income of most countries and governments and companies underestimate how much risk they face from cybercrime and how quickly this risk can grow.[1]

OlaCabs hacked, cr

n

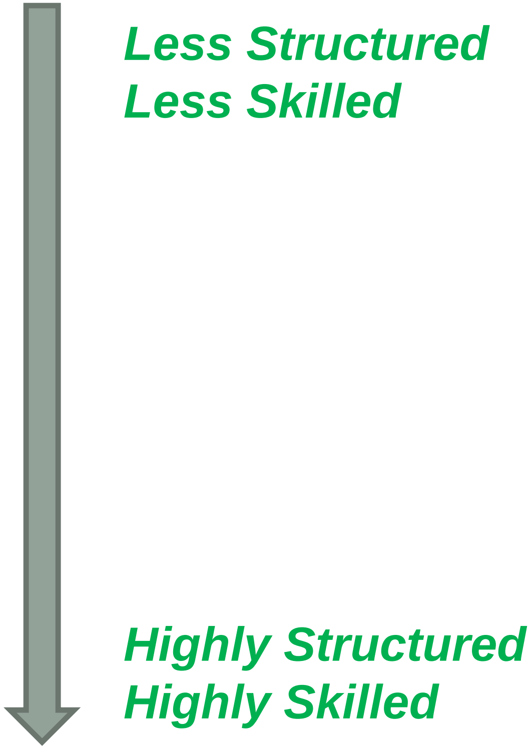
gh as

Stakeholders

- People
  - Passwords, Bank/Credit Card details, Personal Information etc.
- Researchers
  - Chrysler, GM
- Banks, Stock Markets, Financial Organization
  - JP Morgan Chase
- Small/Large Business
  - OlaCabs, Ebay, Sony, LinkedIn, NASA, ISRO and many more…..
- Countries
  - India, USA, UK, Japan, Pakistan, Iran, S. Korea, and every single country

Attackers

- Trusted Insiders
- Hackers/Hactivist
- Terrorist and Extremist Groups
- Industrial Spies and Organized Crime Groups
- Nation States



Less Structured
Less Skilled

Highly Structured
Highly Skilled

Our Goal

- Understand how the attack works and defend against them

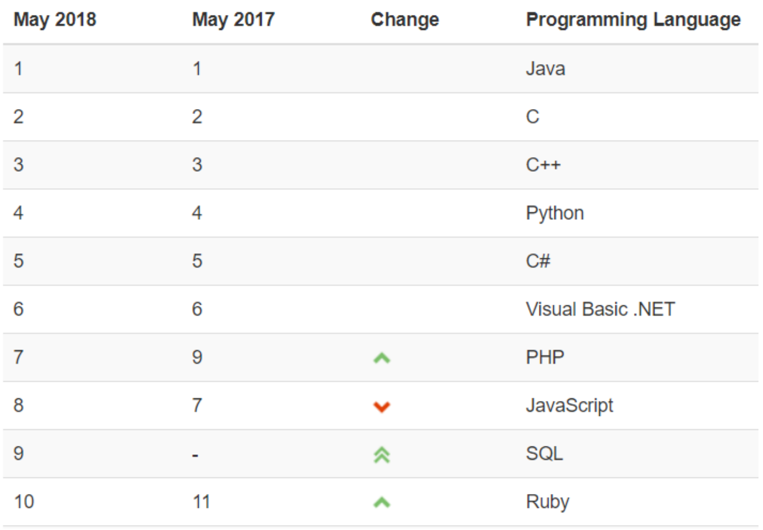
- This require knowledge about
  - Operating System
  - Architecture
  - Compiler

Analyzing security requires complete system view

PROGRAMMING LANGUAGE

Why C/C++

- C/C++ are in top 3
- Used in many critical systems



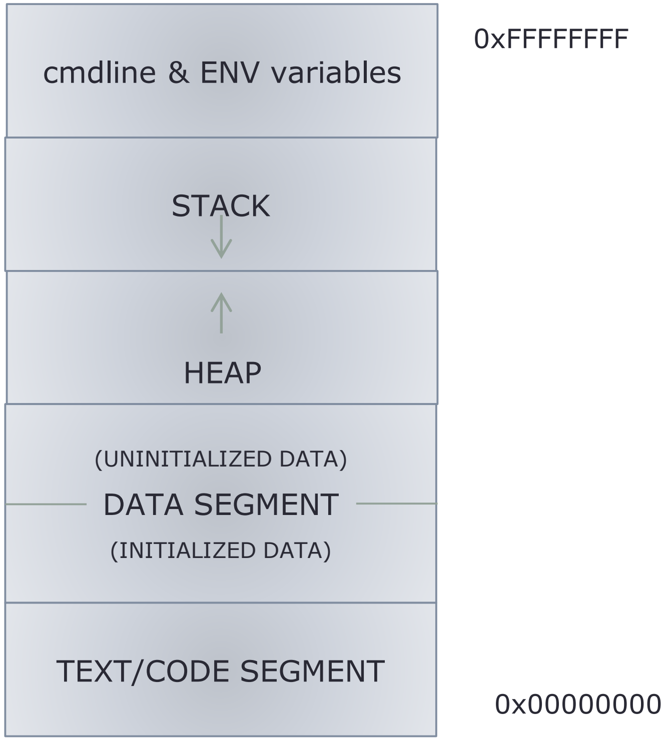
May 2018	May 2017	Change	Programming Language
1	1		Java
2	2		C
3	3		C++
4	4		Python
5	5		C#
6	6		Visual Basic .NET
7	9	▲	PHP
8	7	▼	JavaScript
9	-	▲▲	SQL
10	11	▲	Ruby

Why C/C++ is Important

- Major OS (kernel) and related utilities
  - shell
- High-performance servers
  - Apache serve
  - Microsoft SQL server
- Many embedded systems
  - Mars rover

BUFFER OVERFLOW

Memory layout of x86



Buffer Overflow

- Buffer is contiguous memory associated with a variable e.g. `*char*` array in C

- Buffer Overflow is any access of a buffer outside of allocated bounds (before or after)
  - over-read
  - over-write

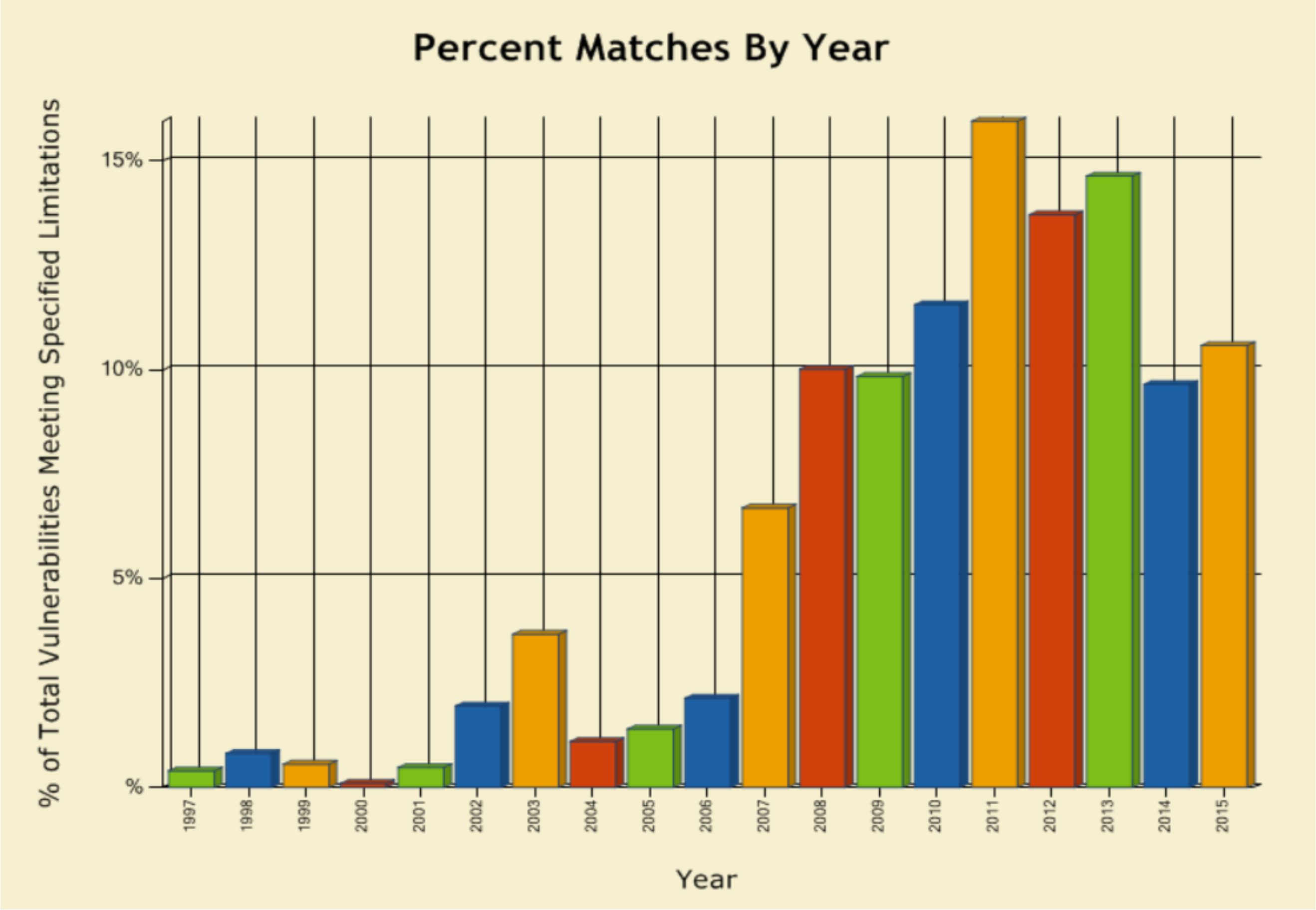
But why are we even talking of "Buffer Overflow"?

Buffer Overflow contd.

- Buffer overflow is a `BUG`.
- Normally this bug will cause system to crash (in fact its good if crash)
- It can have significant security impacts on low-level programming language.
- Attacker can use these conditions to:
  - Steal private information
  - Corrupt important (valuable) information
  - Execute his/her own code (aka Code Injection)

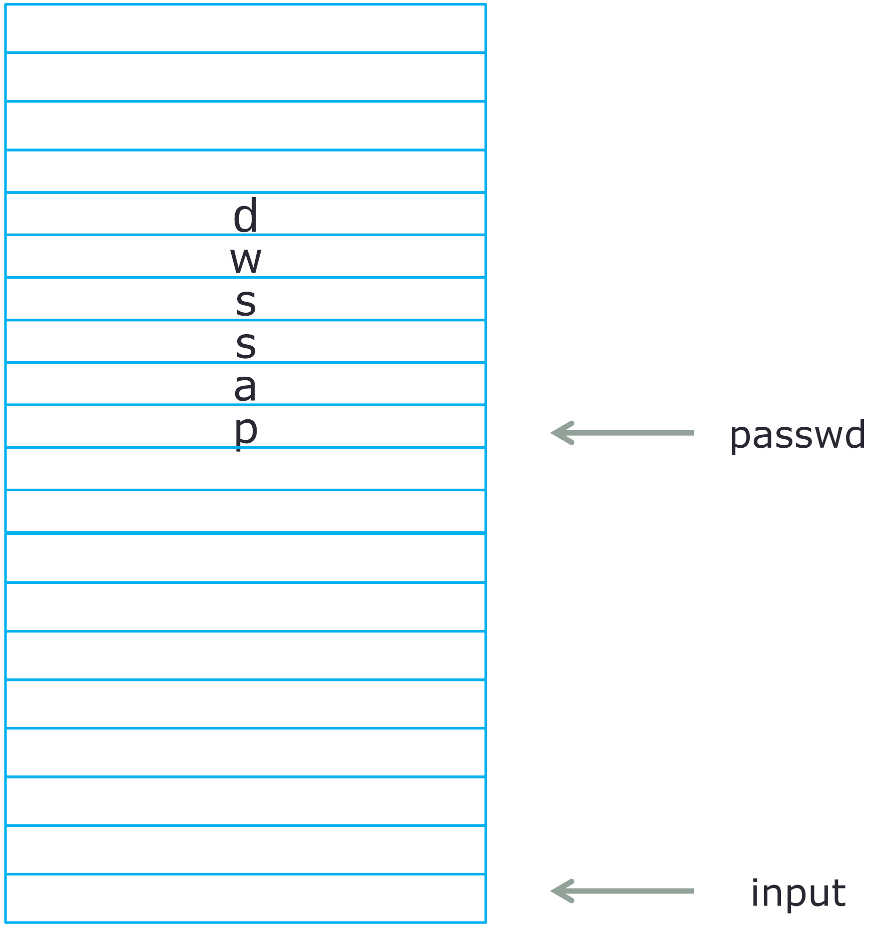
MOST COMMON FORM OF SECURITY VULNERABILITY

Buffer Overflow Trends



Program - Revisit

```
int main(int argc, char *argv[])
{
    if(1 < argc)
    {
        printf("Incorrect number of input parametrs\n");
        return 1;
    }
    const char passwd[10] = “passwd";
    char input[16];
    while(1)
    {
        printf("Enter password: ");
        scanf("%s",&input);
        if(0 == strcmp(passwd,input))
            break;
    }
    printf("Voila!\n");
    return 0;
}
```

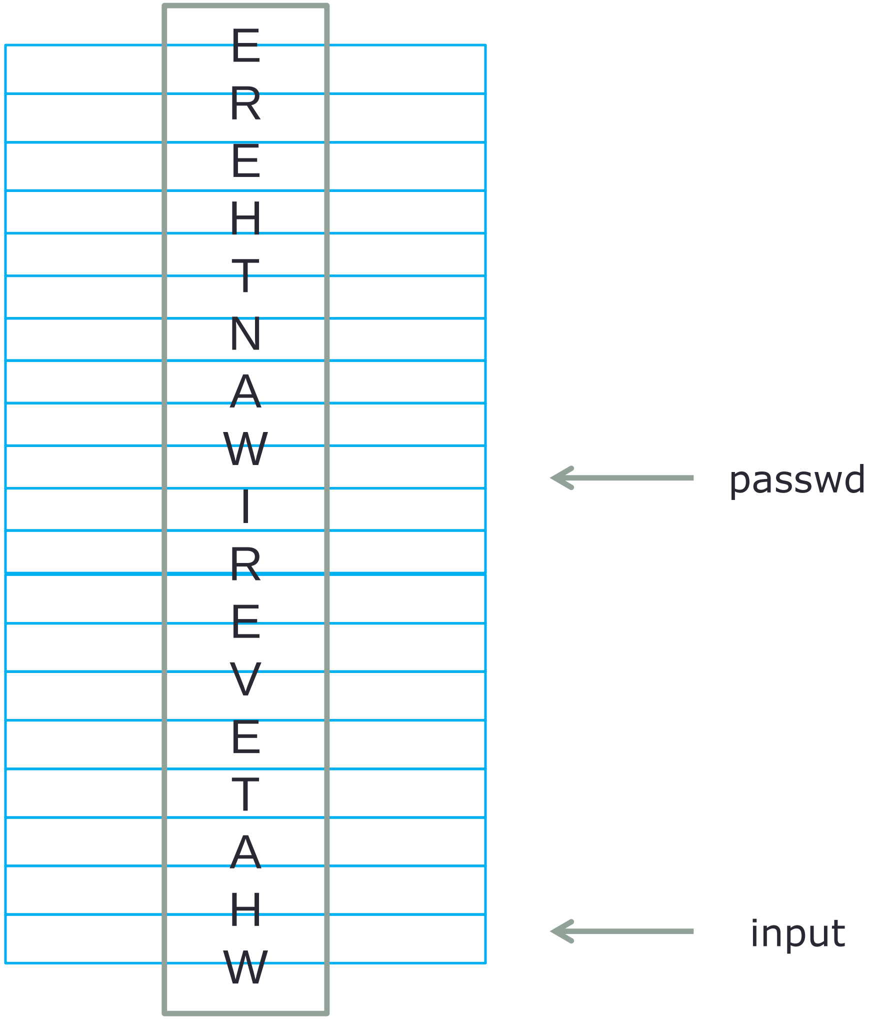


Program – Revisit contd.

after

scanf(“%s”,&input);

And “passwd” is overwritten



E
R
E
H
T
N
A
W
I
R
E
V
E
T
A
H
W

passwd

input

DATA ISSUES

Potential Inputs for Buffer Overflow

- Text Input
- File Input
- Environment Variables
- Packets (Messages)

Plain Text File Input

- PAM (Pluggable Authentication Module) service in Linux is used for authentication

- If integrated with LDAP (Lightweight Directory Access Protocol), “*pam_ldap.conf*” file is required

- WOW!!!!, username and password to LDAP server is in plain text. Easy target to hack

Other Sources of Buffer Overflow

- Buffer Overflow (Read/Write Overflow)
- Heap Overflow
- Integer Overflow
- Stale Memory

LET’S DEFENSE

Behind the Scene – Attacker Perspective

- Attacker gets the control of the data
- Usage of such data permits access to memory area
- Now, attacker has all the control of the system

Secure Development Life Cycle

- Different than regular Software Development Life Cycle

- Security is required at every step of development:
  - Requirements (Architectural Risk Analysis, Security Requirement)
  - Design (Security Oriented Design)
  - Implementation (Secure Coding, Code Reviews)
  - Testing/ Quality Assurance (Security Tests, Penetration Testing)

Security Requirements

- Security policies and goals:
  - E.g. one person’s data (email/account) shall not be seen (or modified) by other users

- Required mechanism to enforce them
  - E.g. users needs to be authenticated using password
  - E.g. Based on the role, user can access the data
  - Authentication, Authorization are the keys

Security Oriented Design

- Design Software which has:
  - Prevention (Authentication using passwords, Firewall)
  - Detection (Monitoring, Logging operational activities)
  - Mitigation (Authorization, Data encryption)
  - Recovery (Back-up)

Secure Coding

- My view - Secure Coding is just a common sense with system knowledge

- The intention is to leave no loop-holes in the system:
  - Unattended data
  - Dangling pointers
  - Unclean memory
  - Unused/Temporary files
  - Open sockets

Principles of Secure Coding

- Never trust user
- Never ignore compiler warnings
- Validate input & data sanity
- Return oriented programming
- Use NULL after free
- Data encryption
- Have defense & recovery mechanism
- Don’t try adventurous coding (avoid complex code)
- Use safer APIs (good coding practices)

Safe `C` APIs

- Need of safe `C` APIs
  - The standard `C` APIs are decades old, thus doesn’t address today’s security issues
  - e.g. strcpy(dest,src) doesn’t check buffer boundaries
- Strong Points
  - Input validation to avoid buffer overflows
  - Have return values
- Challenges
  - Slow performance
  - More code is required to handle return codes

Safe `C` string APIs list

Standard `C` API	Safe `C` API	Safer API
strcpy	strncpy	strlcpy
strcat	strncat	strlcat
sprintf	snprintf	
gets	fgets	

Standard `C` API	Safe `C` API (Microsoft Versions)
strcpy(dest, src)	strcpy_s(dest, size, src)
strcat(dest, src)	strcat_s(dest, size, src)
strlen(str)	strnlen_s(str, maxcnt)

Secure Testing

- Secure testing is not functional testing

- Functional test ensures that legitimate users can use the system conveniently and expectedly with basic checks and safety measures

- Secure testing is from attacker's perspective, which ensures that software is secure

e.g. functional testing ensures that user with non-admin privileges cannot access system files, however secure testing ensures that a non-admin users shall not get admin privileges

Secure Testing Tools

- Static Analysis
  - Coverity, klocwork, Fortify, Lint

- Dynamic Analysis
  - Valgrind
  - Netspark (Web)

- Fuzz Testing (aka Fuzzing)

- Penetration testing

Reality = ytilaeR

- Most of the people are not even aware of it
  - Those who are aware, don’t accept till it happens
  - Lack of resources

- Need time/ resources to create secure software
  - High competition
  - Short time to market
  - Everyone is looking for fast solution

Are we done yet?

WEB VULNERABILITIES

SQL Injection

SELECT Age FROM Users WHERE Name=‘Dee’;

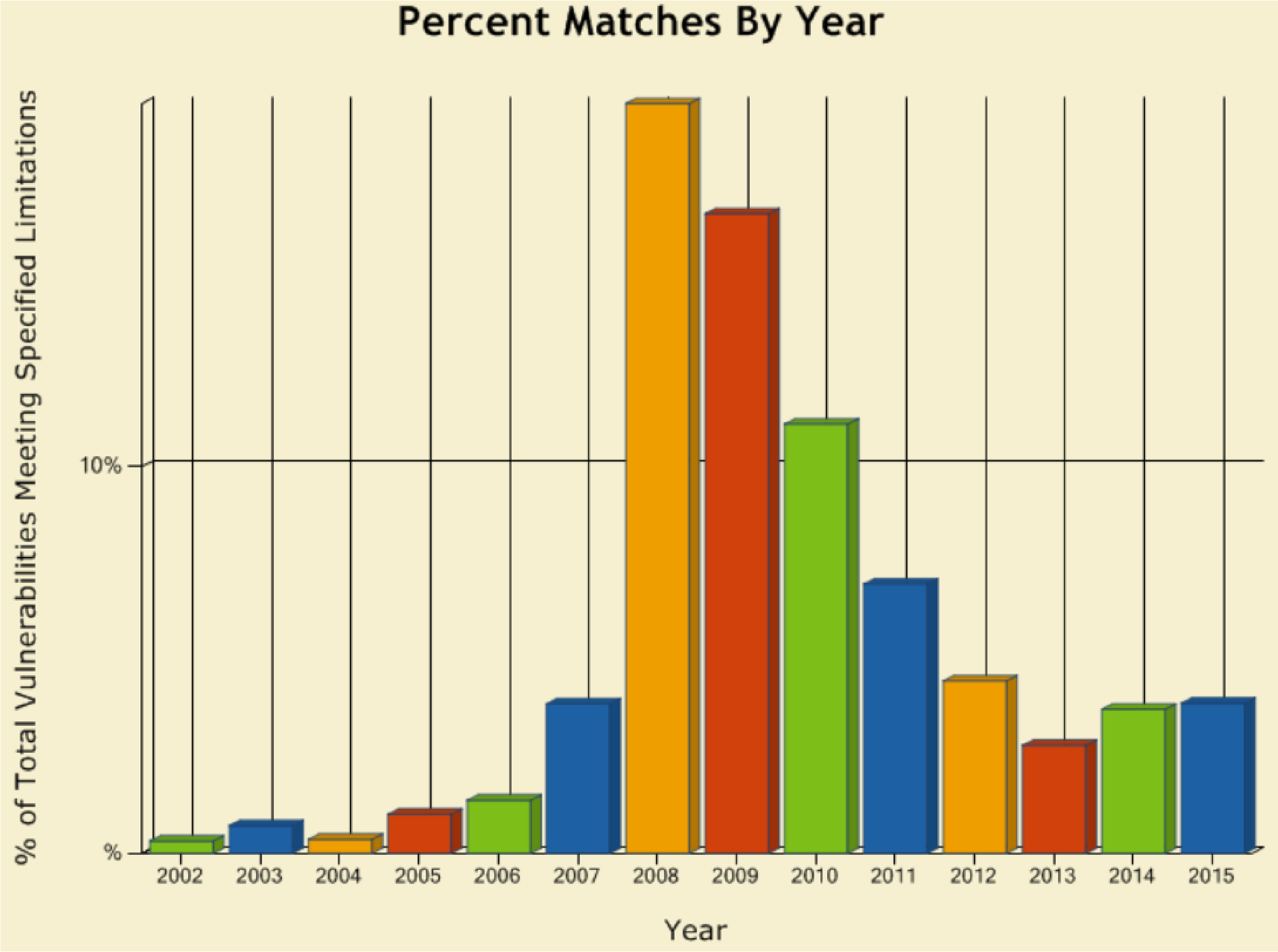
Server side Login Code – PHP

$result = mysql_query(“select * from Users where(name=‘$user’ and password=‘$pass’);”);

What if user input as ***frank’ OR 1=1); --***

$result = mysql_query(“select * from Users where(name=‘frank’ OR 1=1); --’ and password=‘whocares’);”);

SQL Injection Trends



Weak Areas

- URL parameters
  - http://tgt.com/buy?item=1&price=5.00
- Default user & password
  - E.g. root, admin
- Hidden files & directories

FUTURE POINTERS

Future Learning

- HTTP
- Session Cookies
- CSRF (Cross-Site Request Forgery)
- XSS (Cross-Site Scripting)

References

- http://www.cse.scu.edu/~tschwarz/coen152_05/Lectures/BufferOverflow.html
- https://crypto.stanford.edu/cs155/papers/cowan-vulnerability.pdf
- http://heartbleed.com/

Thank You