Access Control

Unit -3

What is Access control?

Access control is a security technique that can be used to regulate who or what can view or use resources in a computing environment. There are two main types of access control: physical and logical. Physical access control limits access to campuses, buildings, rooms and physical IT assets. Logical access limits connections to computer networks, system files and data.

Access is the flow of information between a subject and a resource. A subject is an active entity that requests access to a resource or the data within a resource.

E.g.: user, program, process etc.

A resource is an entity that contains the information. E.g.: Computer, Database, File, Program, Printer etc.

Different Models Of Access Control

Different access control models are:

1. **Attribute-based Access Control (ABAC):** In this model, access is **granted or declined** by evaluating a set of rules, policies, and relationships using the attributes of users, systems and environmental conditions.
2. **Discretionary Access Control (DAC):** In DAC the owner of data determines **who can access specific resources.**

3. **History-Based Access Control (HBAC):** In this model, access is **granted or declined by evaluating the history of activities** of the inquiring party that includes behaviour, the time between requests and content of requests.

5. **Mandatory Access Control (MAC):** A control model in which access rights are regulated by a central authority based on multiple levels of security. Security Enhanced Linux is implemented using MAC on the Linux operating system.

6. **Organization-Based Access control (OrBAC) :** This model allows the policy designer to define a security policy independently of the implementation.
7. **Role-Based Access Control (RBAC) :** RBAC allows access based on the job title. RBAC eliminates discretion on a large scale when providing access to objects. For example, there should not be permissions for human resources specialist to create network accounts.
8. **Rule-Based Access Control (RAC) :** RAC method is largely context based. For example, this would be only allowing students to use the labs during a certain time of day.

Implementation Of Access Control

Implementation of access control:

1. **Administrative access control:**

a. Administrative access control **sets the access control policies and procedures for the whole organization,** defines the implementation requirements of both physical and technical access control, and what the consequences of non-compliance will be.

b. Examples are supervisory structure, staff and contractor controls, information classification, and training, auditing, and testing.

2. Physical access control:

a. Physical access control is critical to an organizations security and applies to the access or restriction of access to a place such as property, building or room.

b. Examples are fences, gates, doors, turnstiles, etc., **using locks, badges, bio-metrics** (facial recognition, fingerprints), video surveillance cameras, security guards, motion detectors, mantrap doors, etc., to allow access to certain areas.

3 .Technical or logical access control :

a. Technical or logical access control limits connections to computer networks, system files, and data.

b. It enforces restrictions on applications, protocols, operating systems, encryptions mechanisms, etc.

c. Examples are access control lists, intrusion detection systems, and antivirus software.

Access control components

- Access Cards
- Card readers
- Access control keypads
- Electronic lock hardware
- Access control field panels
- Access control server computer

Issues in Access control

- Appropriate role based access
- Poor password management
- Poor user education

Unix OS

Characteristics of Unix:

- Memory management
- Processor management
- Device management
- File management
- Security

Features of Unix:

- portable

- Multi-user
- Multi-tasking
- Networking
- Organized file system
- Device independence
- Utility

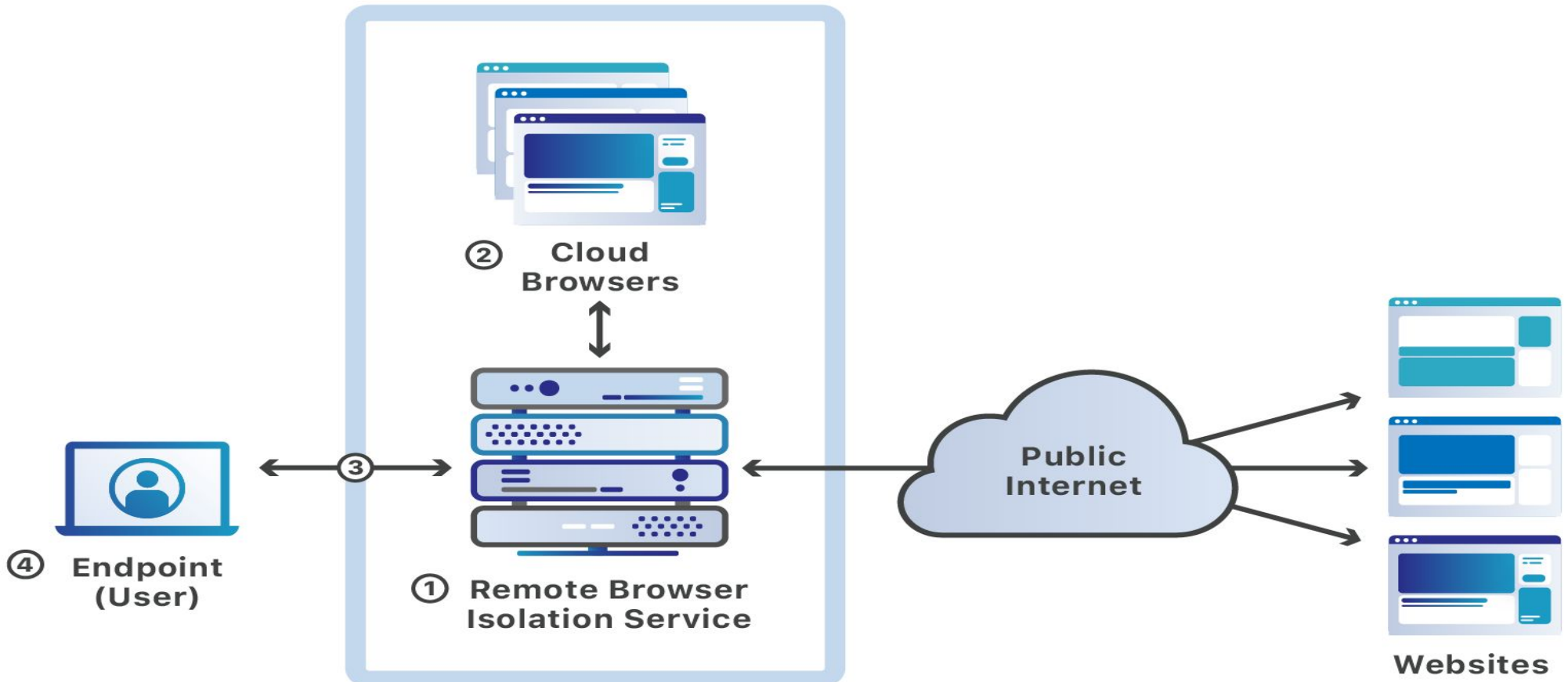
Difference between unix and windows:

Unix o.s	Windows o.s
• 1.It is an open source.	• 1.It is a close source.
• 2.It has very high security system.	• 2.It has low security system.
• 3.It is a command based O.S.	• 3.It is a not command based O.S.
• 4.The file system is arranged in hierichial manner. • 5.unix is not user friendly.	• 4.The file system is arranged in a parallel manner • 5 . It is user friendly

Browser Isolation

- Browser Isolation (also known as Web Isolation) is a technology that contains web browsing activity inside an isolated environment, like a sandbox or virtual machine, in order to protect computers from any malware the user may encounter.
- This isolation may occur locally on the computer or remotely on a server.
- Browser Isolation technology provides malware protection for day-to-day browsing by eliminating the opportunity for malware to access the end user's device.

Contin. From Word doc.


② Cloud Browsers
③
④ Endpoint (User)
① Remote Browser Isolation Service
Public Internet
Websites

CLOUDFLARE

Web security

- In general, 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.
- **web security gateway** protects organizations against online threats by monitoring and filtering internet traffic in real time and blocking traffic deemed to be suspicious, malicious, or outside of policy.
- Web Security functions as a web security gateway, enabling access to benign websites and blocking access to inappropriate sites.

Web security Goals:

The ultimate goal of cybersecurity is to protect the information from being stolen or compromised. To achieve this we look at 3 fundamental goals of cybersecurity.

- 1. Protecting the Confidentiality of data
- 2. Preserving the Integrity of data
- 3. Restricting the Availability of data only to authorized users

Advantages of web security

- Block access to sites that contain malware, phishing and other threats.
- Prevent users from visiting certain categories of websites that are inappropriate for business use.
- Help to ensure that files downloaded from the web are free of threats and malware.
- Block compromised devices from communicating with attackers using the web.
- Protect data from exfiltration attacks.
- Improve understanding of employee use of the web.
- Simplify administration by managing web security from a single, cloud-based console and by consistently applying security policies throughout the organization.
- Combine email and web security in a single, easy-to-use solution.

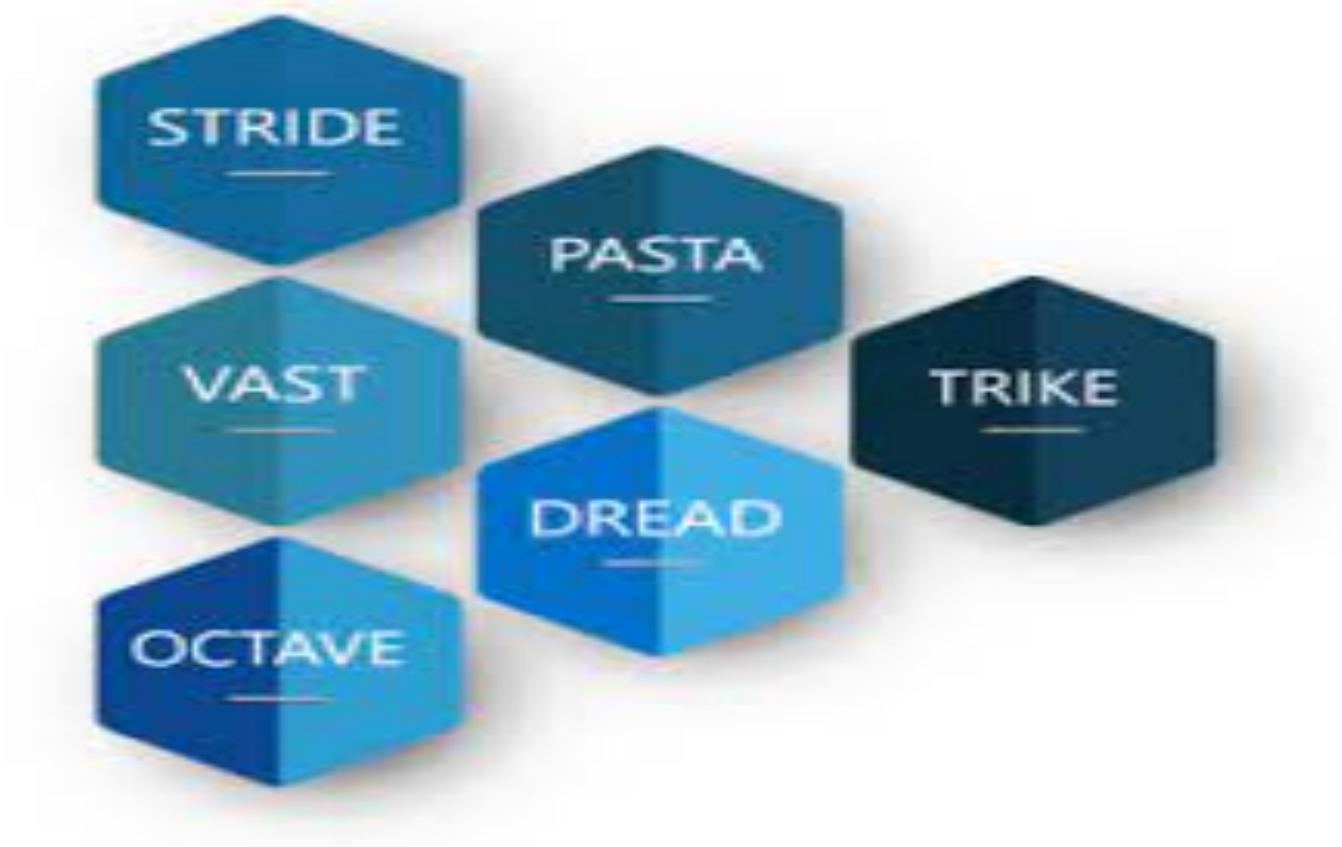
web security threats?

- Web security threats are vulnerabilities within websites and applications, or attacks launched by malicious actors.
- Web security threats are designed to breach an organizations security defenses, enabling hackers and cyber criminals to control systems, access data and steal valuable resources.
- Common web security threats include malware, **ransomware**, cross-site scripting (XSS), SQL injection, **phishing**, denial of service and many others.

Definition of threat modeling

- Threat modeling is a structured process with these objectives: identify security requirements, pinpoint security threats and potential vulnerabilities, quantify threat and vulnerability criticality, and prioritize remediation methods. Threat modeling methods create these artifacts:
- An abstraction of the system
- Profiles of potential attackers, including their goals and methods
- A catalog of threats that could arise

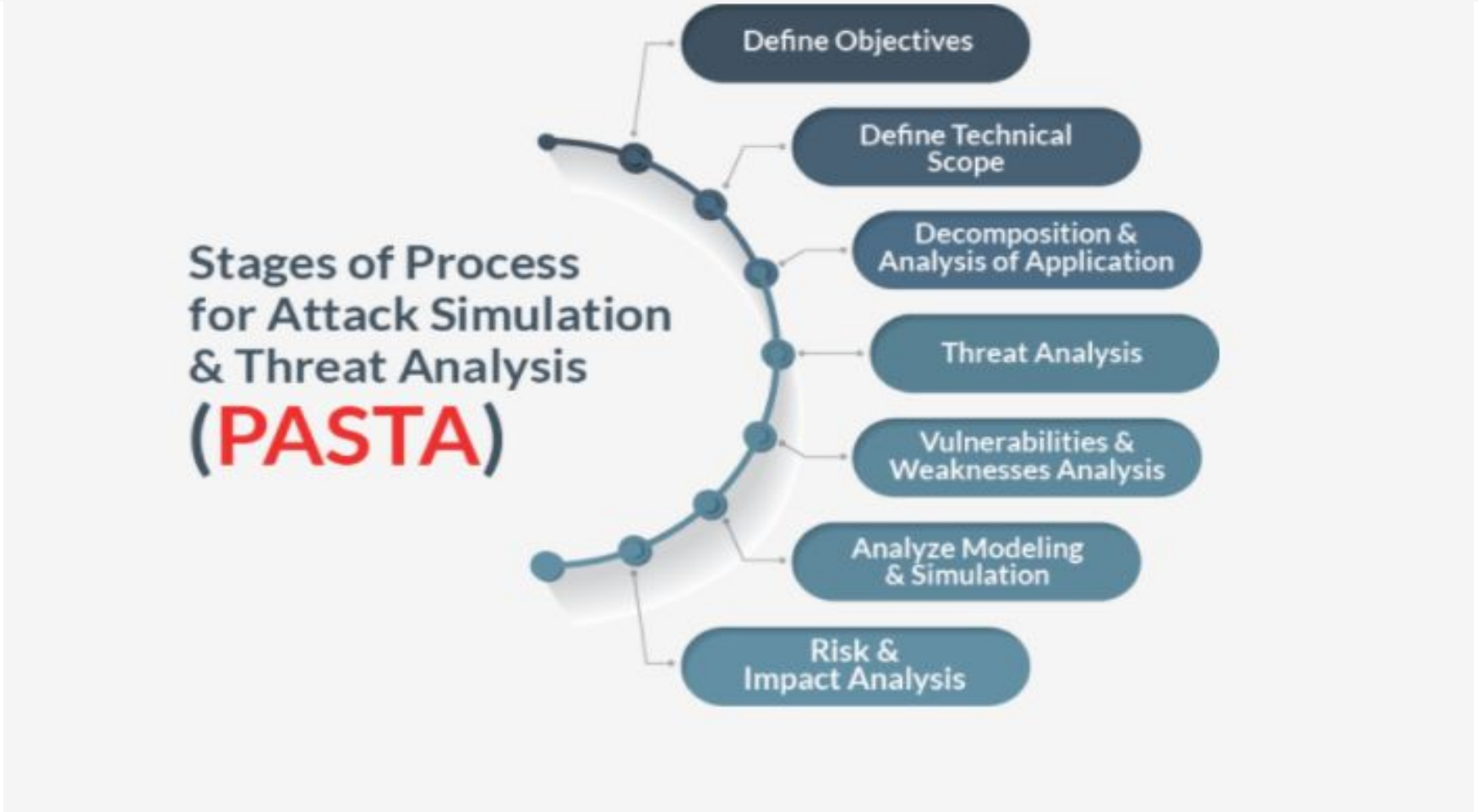
Threat models



	Type of Threat	What Was Violated	How Was It Violated?
S	Spoofing	Authentication	Impersonating something or someone known and trusted.
T	Tampering	Integrity	Modifying data on disk, memory, network, etc.,
R	Repudiation	Non-repudiation	Claim to not be responsible for an action
I	Information Disclosure	Confidentiality	Providing information to someone who is not authorized
D	Denial of Service (DoS)	Availability	Denying or obstructing access to resources required to provide service
E	Elevation of Privilege	Authorization	Allowing access to someone without proper authorization

Stages of Process for Attack Simulation and Threat Analysis (PASTA)

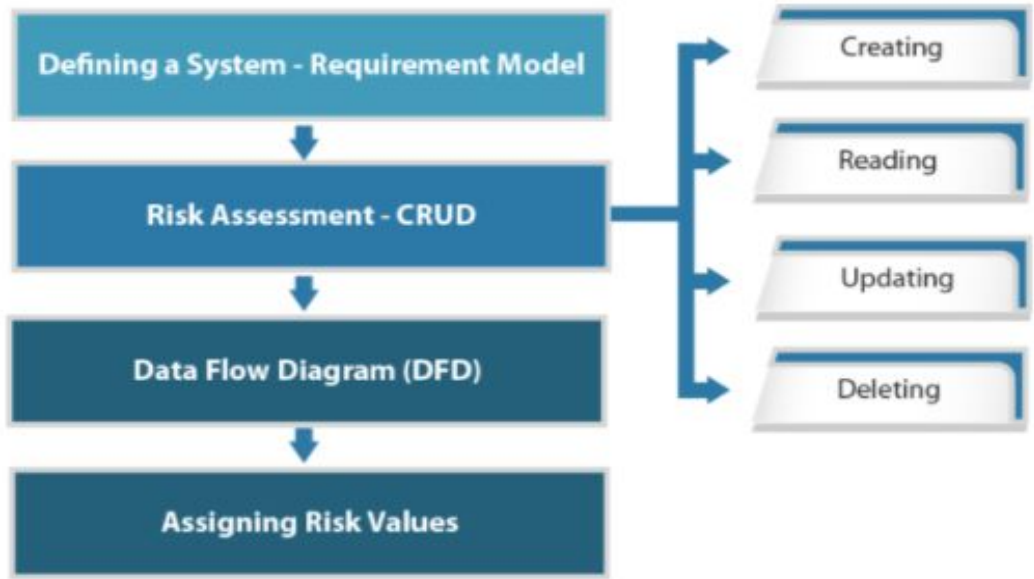
PASTA is a seven-step methodology to create a process for simulating attacks to IT applications, analyzing the threats, their origin, the risks they pose to an organization, and how to mitigate them. The objective of this model is to identify the threat, enumerate them, and assign a score. By following this method, the organization can determine the appropriate countermeasures that must be deployed to mitigate the risk.



TRIKE

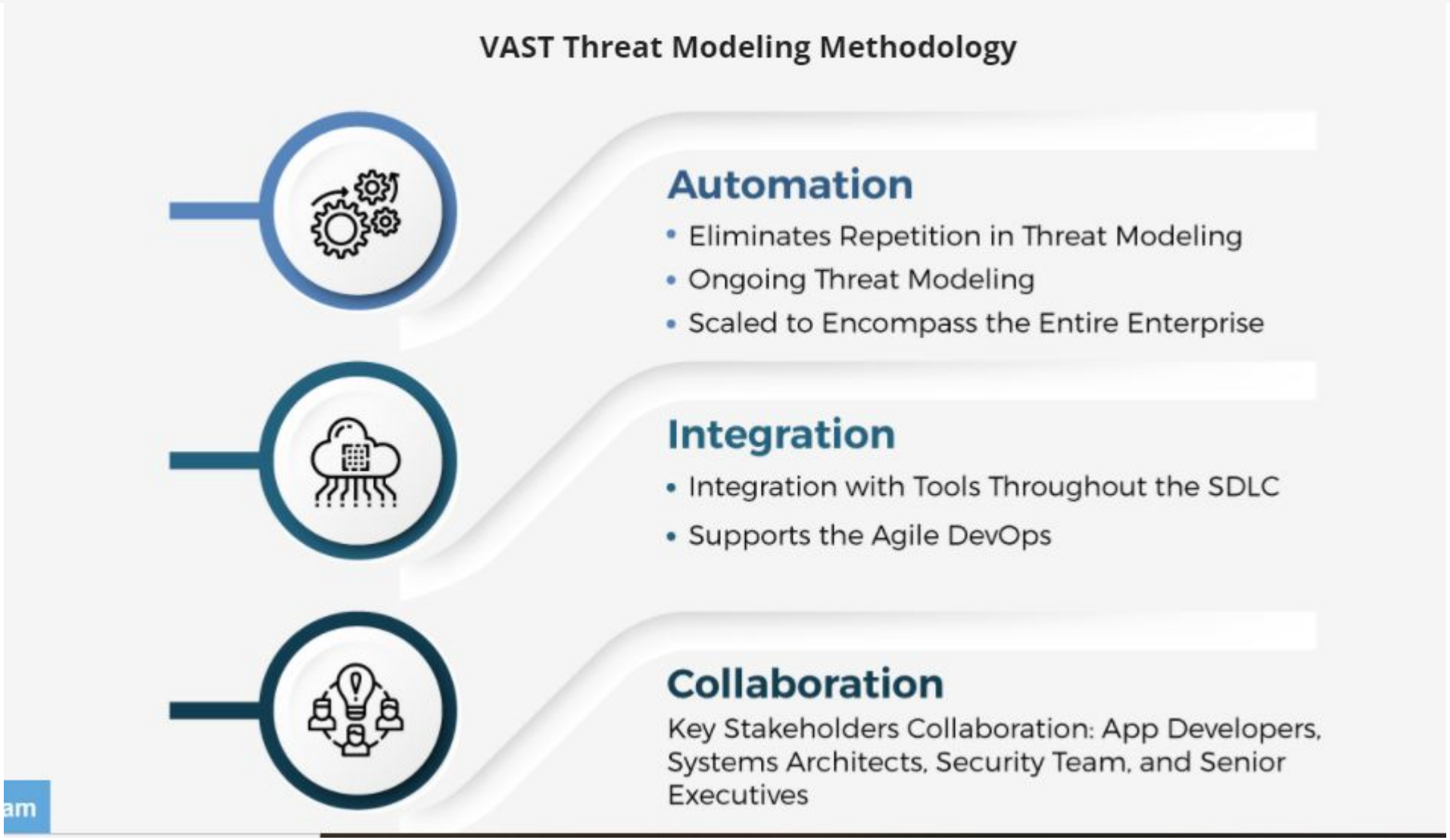
TRIKE is an open-source threat modeling methodology that is used when security auditing from a risk management perspective. TRIKE threat modeling is a fusion of two models namely – Requirement Model and Implementations Model. The requirement model is the base of TRIKE modeling that explains the security characteristics of an IT system and assigns acceptable levels of risk to each asset. This model also enables coordination among different security teams and stakeholders by providing a conceptual framework. After this comes the implementation model. In this model, a Data Flow Diagram (DFD) is created to illustrate the flow of data and the user performed actions within a system. In this model, threats are analyzed to enumerate and assign a risk value. Based on this, security controls or preventive measures defined to address the threats as per the priority and assigned risks.

TRIKE Methodology



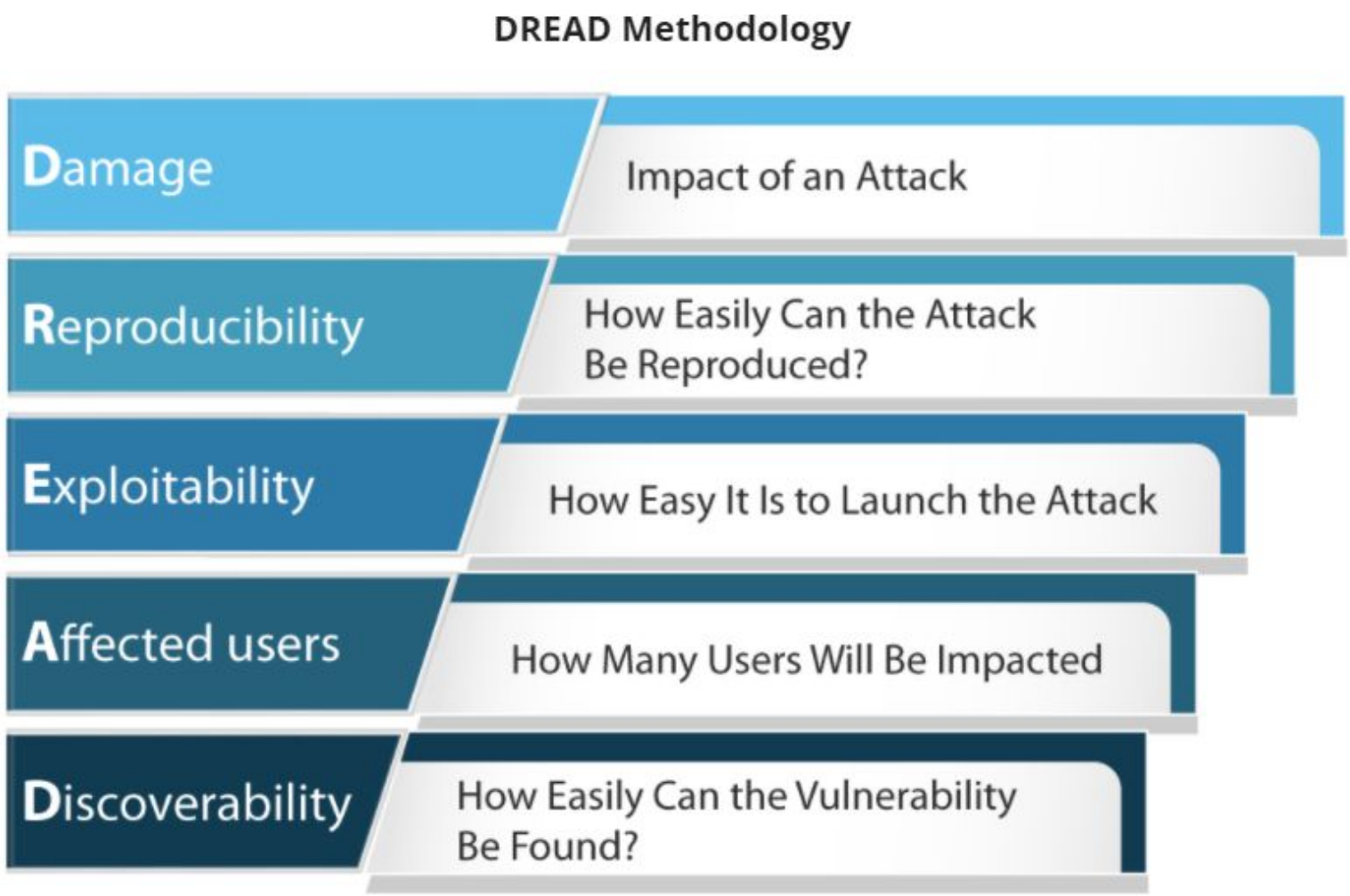
VAST

VAST (Visual, Agile, and Simple Threat) methodology is based on automated threat modeling that covers the software development life cycle across the organization with proper integration with tools and collaboration with all key stakeholders like developers, architects, security professionals, and leaders across the organization.



DREAD

DREAD methodology is used to assess, analyze, and find the probability of risk by rating the threats as described in the image below.



OCTAVE

OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) is an approach to identify, assess, and manage risks to IT assets. This process identifies the critical components of information security and the threats that could affect their confidentiality, integrity, and availability. This helps them understand what information is at risk and design a protection strategy to reduce or eliminate the risks to IT assets.

Objectives of the OCTAVE Model

Threat Modeling Tools

Microsoft Threat Modelling Tool

Microsoft's Threat Modelling Tool is free and allows software architects to identify and mitigate most likely security issues at an early stage when they are comparatively easy and cost-effective to fix. Thus, reducing the total cost of development.

ThreatModeler

ThreatModeler is an automated threat modeling tool that secures and scales the enterprise software development life cycle (SDLC). It helps identify, predict, and define threats on the attack surface to make proactive security measures and reduce the overall risk.

securiCAD Professional

securiCAD Professional helps create virtual models of existing and future IT environments. The attack simulations on a virtual model provides detailed insights about the security posture of the organization. This facilitates prioritizing security mitigations and compare different design alternatives. securiCAD works with intruder's mindset and to find the most likely attack paths in your IT systems.

IriusRisk

Iriusrisk is a threat modeling tool with architectural design and questionnaires defined by an expert system that explains the technical architecture, the features, and the security context of the application. The model has major components and a list of the potential security risks and vulnerabilities and provides specific recommended preventive measures.

SD Elements

SD Elements is a software security requirement management platform that allows automation of the security process. It reduces the risks and its cost by following the best practices, easily tracking the status security part of the project and fast security and compliance processes in line with Agile and DevOps methods.

Tutamen

The Tutamen Threat Modeling tool is designed to enable security during architecture, to minimize the cost of resolving flaws. The automation reduces human error and inconsistencies with a single input of variables. Tutamen is a living threat model that changes accordingly with the design.

OWASP Threat Dragon

OWASP Threat Dragon is an open-source threat modeling tool (both web application and desktop) that is used to create threat model diagrams, record the most likely threats, and decide the action to mitigate said threats. Threat Dragon is both an online threat modeling web application and a desktop application. It includes system diagramming as well as a rule engine to auto-generate threats and their mitigations.

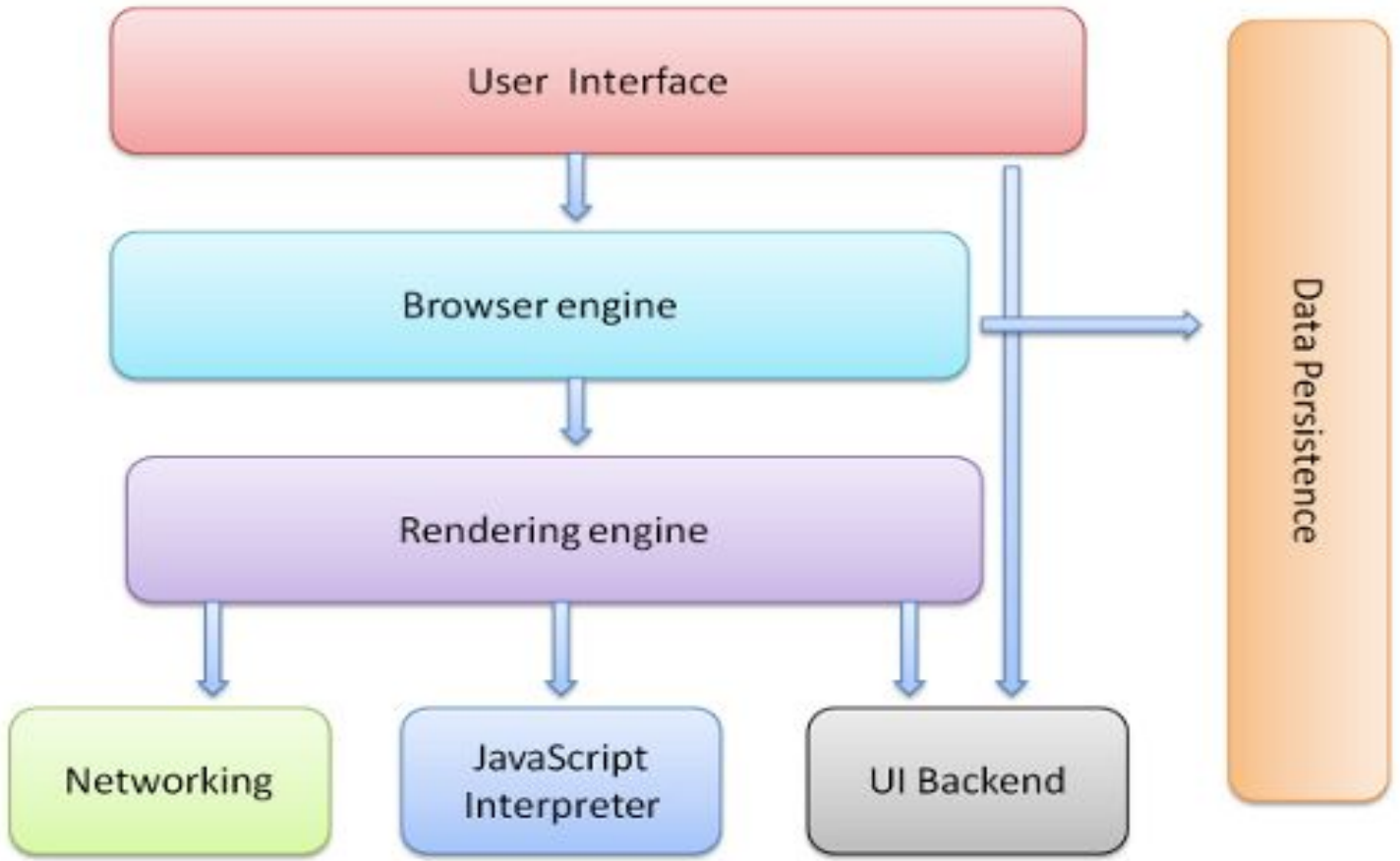
Thus, threat modeling delivers more value if executed more frequently and consistently. When threat modeling is conducted on a consistent basis throughout an organization, secure architecture patterns start emerging that can be documented and leveraged by security and development teams.

Rendering

- Rendering or image synthesis is the process of generating a photorealistic or non-photorealistic image from a 2D or 3D model by means of a computer program. The resulting image is referred to as the render.

Understanding the architecture of a web browser

A web browser is a software application that enables a user to access and display web pages or other online content through its graphical user interface. Refer to the image below to understand the key components involved in building a web browser.



- **Browser Engine:** It is a core component of every web browser. The browser engine functions as an intermediary or a bridge between the user interface and the rendering engine. It queries and handles the rendering engine as per the inputs received from the user interface.

- **Rendering Engine:** As the name suggests, this component is responsible for rendering a specific web page requested by the user on their screen. It interprets HTML and XML documents along with images that are styled or formatted using CSS, and a final layout is generated, which is displayed on the user interface.

Note: Every browser has its own unique rendering engine. Rendering engines might also differ for different browser versions. The list below mentions browser engines used by a few common browsers:

1. Google Chrome and Opera v.15+: **Blink**
2. Internet Explorer: **Trident**
3. Mozilla Firefox: **Gecko**
4. Chrome for iOS and Safari: **WebKit**

Role of Rendering Engine:

Once a user requests a particular document, the rendering engine starts fetching the content of the requested document. This is done via the networking layer. The rendering engine starts receiving the content of that specific document in chunks of 8 KBs from the networking layer. After this, the basic flow of the rendering engine begins.

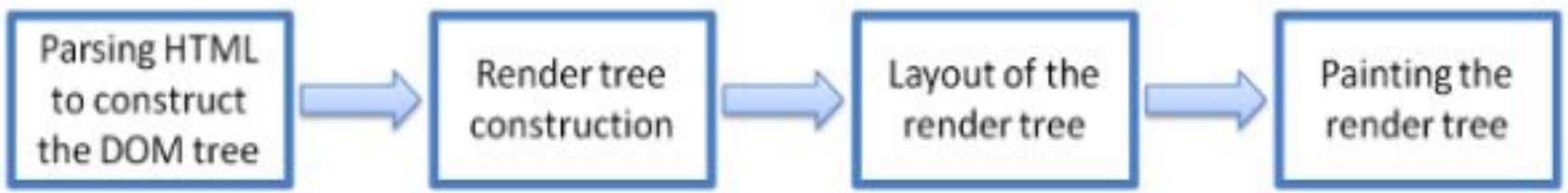


Figure : Rendering engine basic flow

The four basic steps include:

1. The requested HTML page is parsed in chunks, including the external CSS files and in style elements, by the rendering engine. The HTML elements are then converted into DOM nodes to form a **"content tree" or "DOM tree."**

2. Simultaneously, the browser also creates a **render tree.** This tree includes both the styling information as well as the visual instructions that define the order in which the elements will be displayed. The render tree ensures that the content is displayed in the desired order.

3. Further, the render tree goes through the **layout process.** When a render tree is created, the position or size values are not assigned. The entire process of calculating values for evaluating the desired position is called a layout process. In this process, every node is assigned the exact coordinates. This ensures that every node appears at an accurate position on the screen.

4. The final step is to paint the screen, wherein the render tree is traversed, and the renderer's **paint()** method is invoked, which paints each node on the screen using the UI backend layer.

Security Interface

- The Security Interface framework is a set of Objective-C classes that provide user interface elements for programs that implement security features such as authorization, access to digital certificates, and access to items in keychains.
- a **user interface** (**UI**) is the space where interactions between humans and machines occur.
- The goal of this interaction is to allow effective operation and control of the machine from the human end, while the machine simultaneously feeds back information that aids the operators' decision-making process.
- User interfaces are composed of one or more layers, including a **human-machine interface** (**HMI**) that interfaces machines with physical input hardware such as keyboards, mice, or game pads, and output hardware such as computer monitors, speakers, and printers.

12/11

- **Composite user interfaces** (**CUIs**) are UIs that interact with two or more senses. The most common CUI is a *graphical user interface* (GUI), which is composed of a tactile UI and a visual UI capable of displaying graphics

https://en.wikipedia.org/wiki/User_interface

What is Web server?

- A server that will only be used for hosting websites is known as a web server
- Websites are hosted on web servers. Web servers are themselves computers running an operating system; connected to the back-end database, running various applications.
- Any vulnerability in the applications, Database, Operating system or in the network will lead to an attack on the web server.

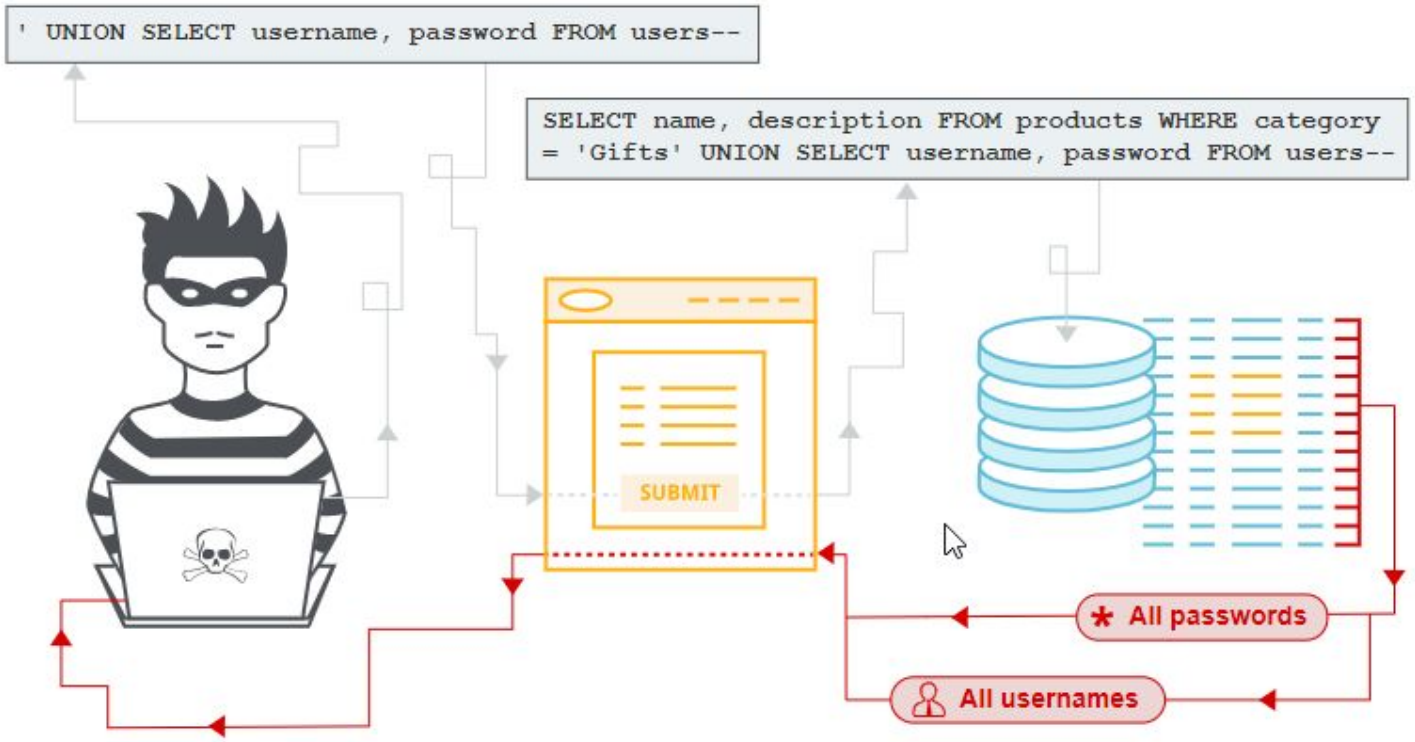
Web server threats:

- **DOS attack:**
- An attacker may cause a denial of service attack by sending numerous service request packets overwhelming the servicing capability of the web server, or he may try to exploit a programming error in the application causing a DOS attack.
- E.g. buffer overflow attack, SYN flooding, HTTP get Request Flooding, Ping of death.

- **Website Defacement:**

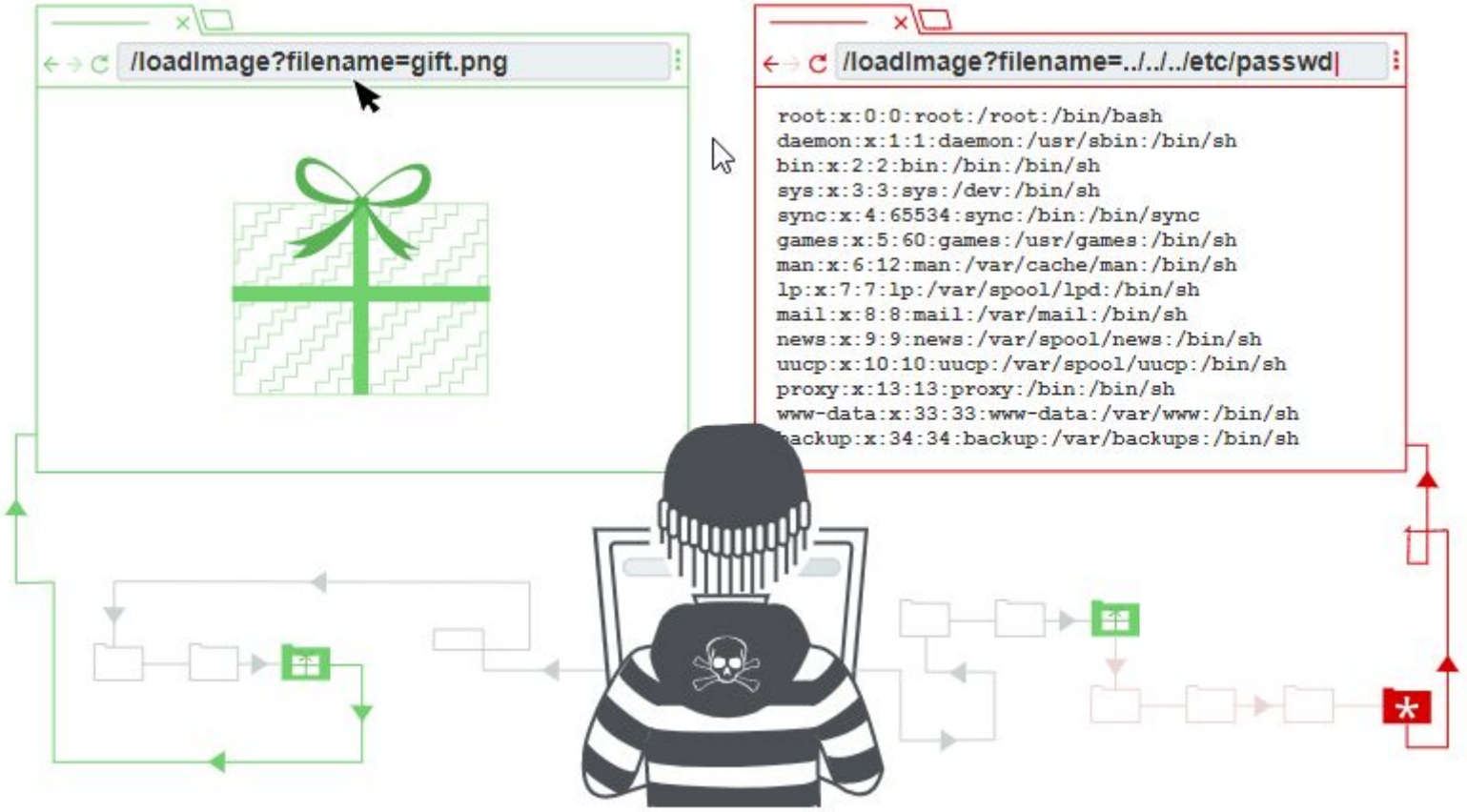
- SQL injection attacks are used to deface the website. When an attacker finds out that input fields are not sanitized properly, he can add SQL strings to maliciously craft a query which is executed by the web browser. He may store malicious/unrelated data in the database; when the website is requested, it will show irrelevant data on the website, thus displaying a defaced website.

Example : SQL Injection



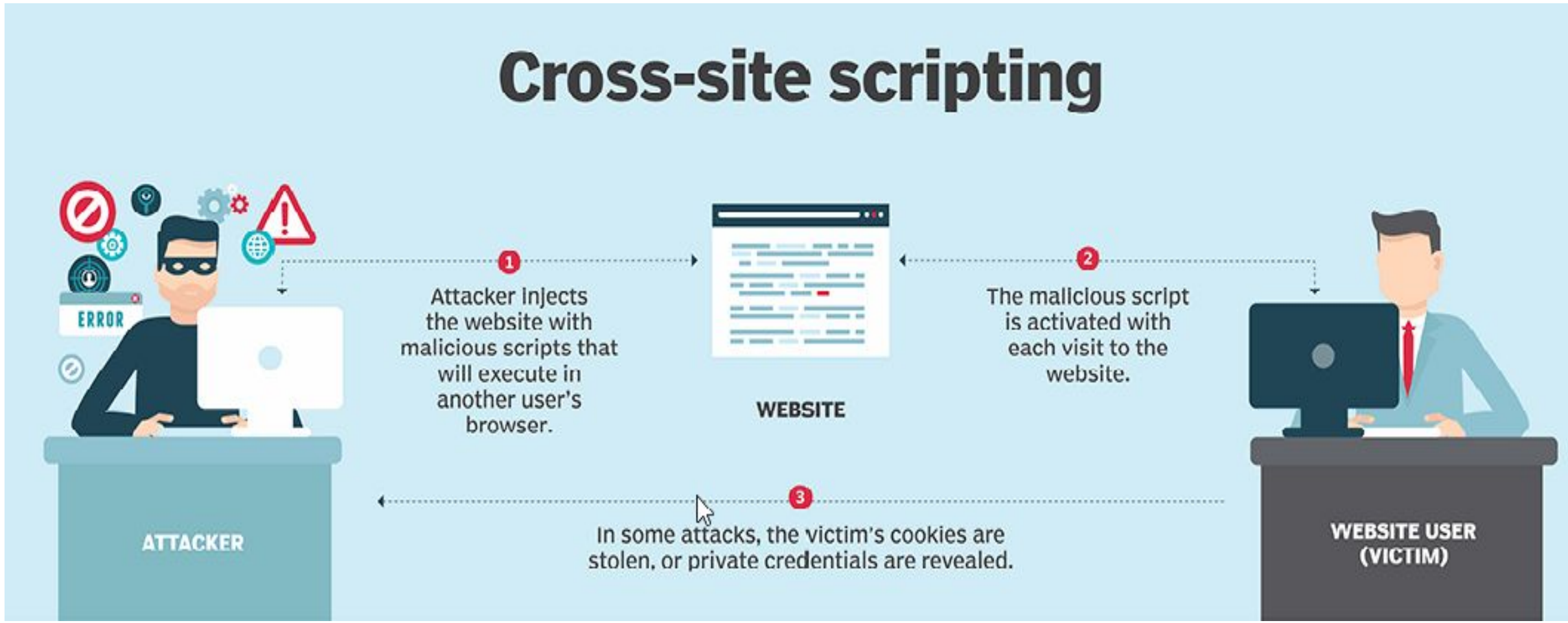
- **Parameter tempering:**
- Parameter tampering is a form of Web-based hacking event (called an attack) in which certain parameters in the Uniform Resource Locator (URL) or Web page form field data entered by a user are changed without that user's authorization.
- **Misconfiguration attacks:**
- If unnecessary services are enabled or default configuration files are used, verbose/error information is not masked; an attacker can compromise the web server through various attacks like password cracking, Error-based SQL injection, Command Injection, etc.

- **Directory Traversal:**
- This is vulnerability where an attacker is able to access beyond the web root directory from the application. If he is able to access beyond web root directory, he might execute OS commands and get sensitive information or access restricted directories.

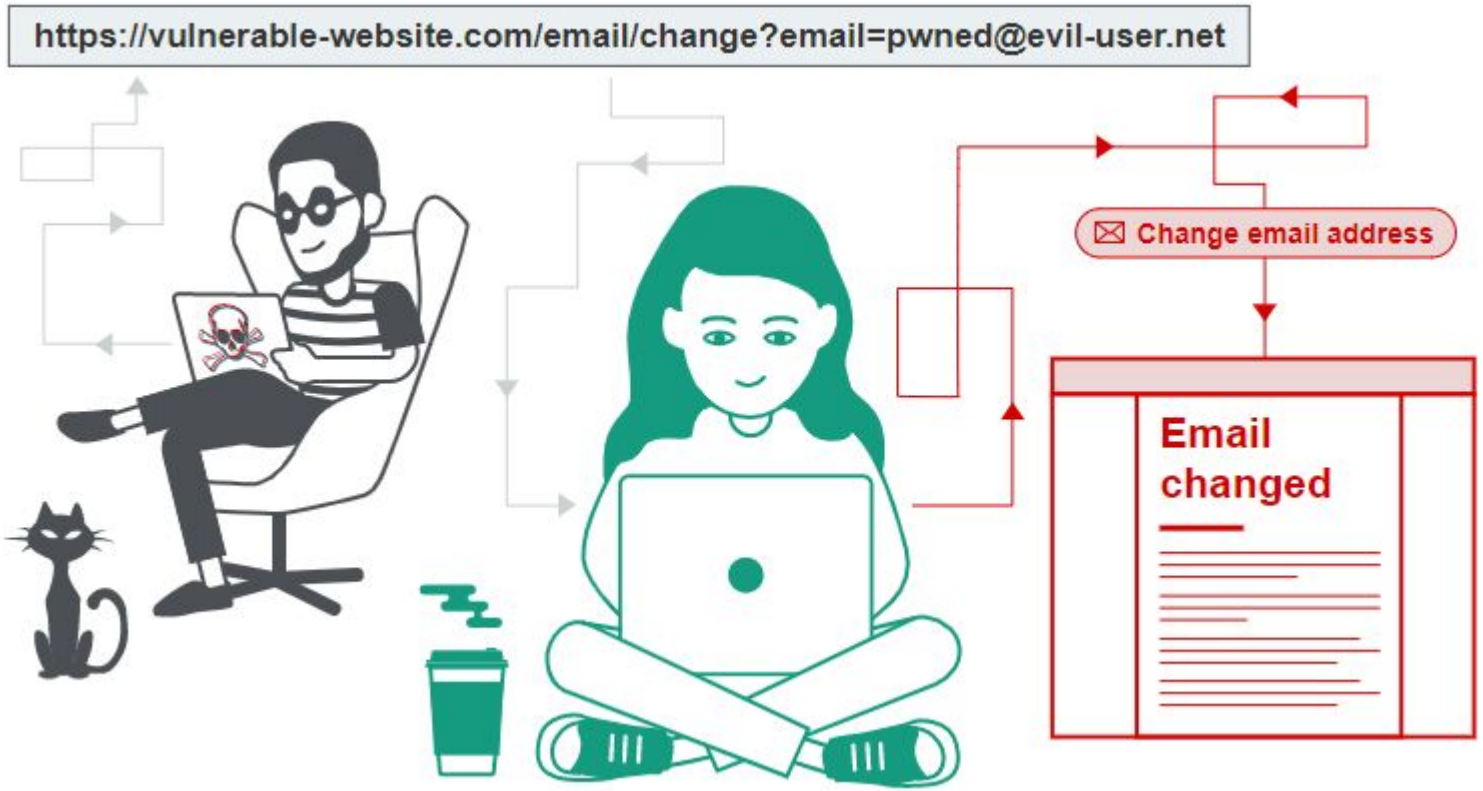


- **Phishing Attack:**
- An attacker may redirect the victim to malicious websites by sending him/her a malicious link by email which looks authentic, but redirects him/her to malicious web page thereby stealing their data.
- **cross-site scripting (XSS):**
- Cross-site scripting (XSS) is a type of injection attack in which a threat actor inserts data, such as a malicious script, into content from trusted websites. The malicious code is then included with dynamic content delivered to a victim's browser.
- There are a lot of other web application attacks which can lead to a web server attack- Parameter form tampering, Cookie tampering, unvalidated inputs, SQL injection, Buffer overflow attacks.

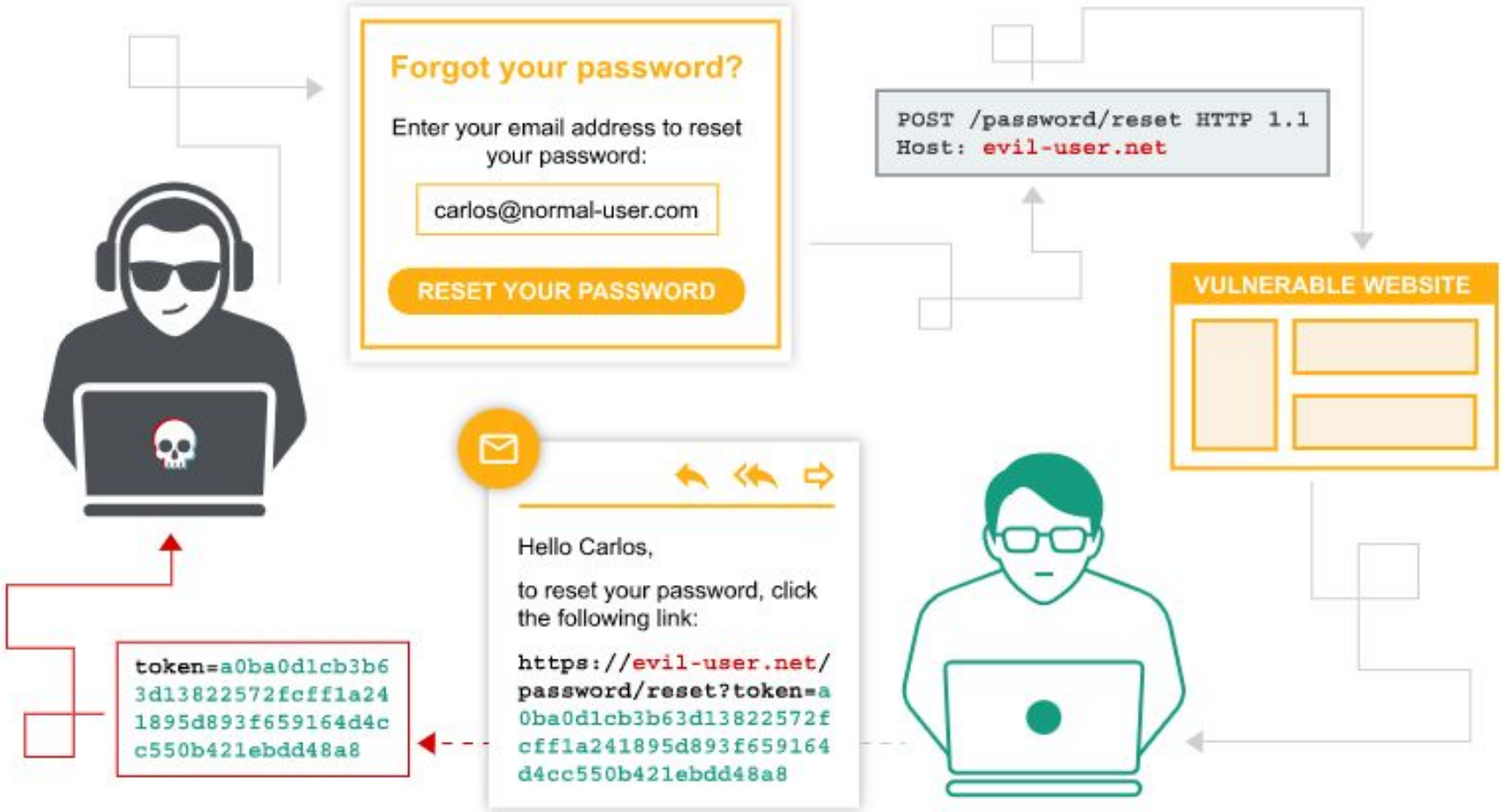
Cross-site scripting



- **Cross-site request forgery(CSRF):**
- Cross-site request forgery (also known as CSRF) is a web security vulnerability that allows an attacker to induce users to perform actions that they do not intend to perform. It allows an attacker to partly circumvent the same origin policy, which is designed to prevent different websites from interfering with each other.



Authentication vulnerabilities:



Cross site request forgery

- Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

- An attacker can use XSS to send a malicious script to an unsuspecting user. The end user’s browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. For more details on the different types of XSS flaws

- Cross-Site Scripting (XSS) attacks occur when:

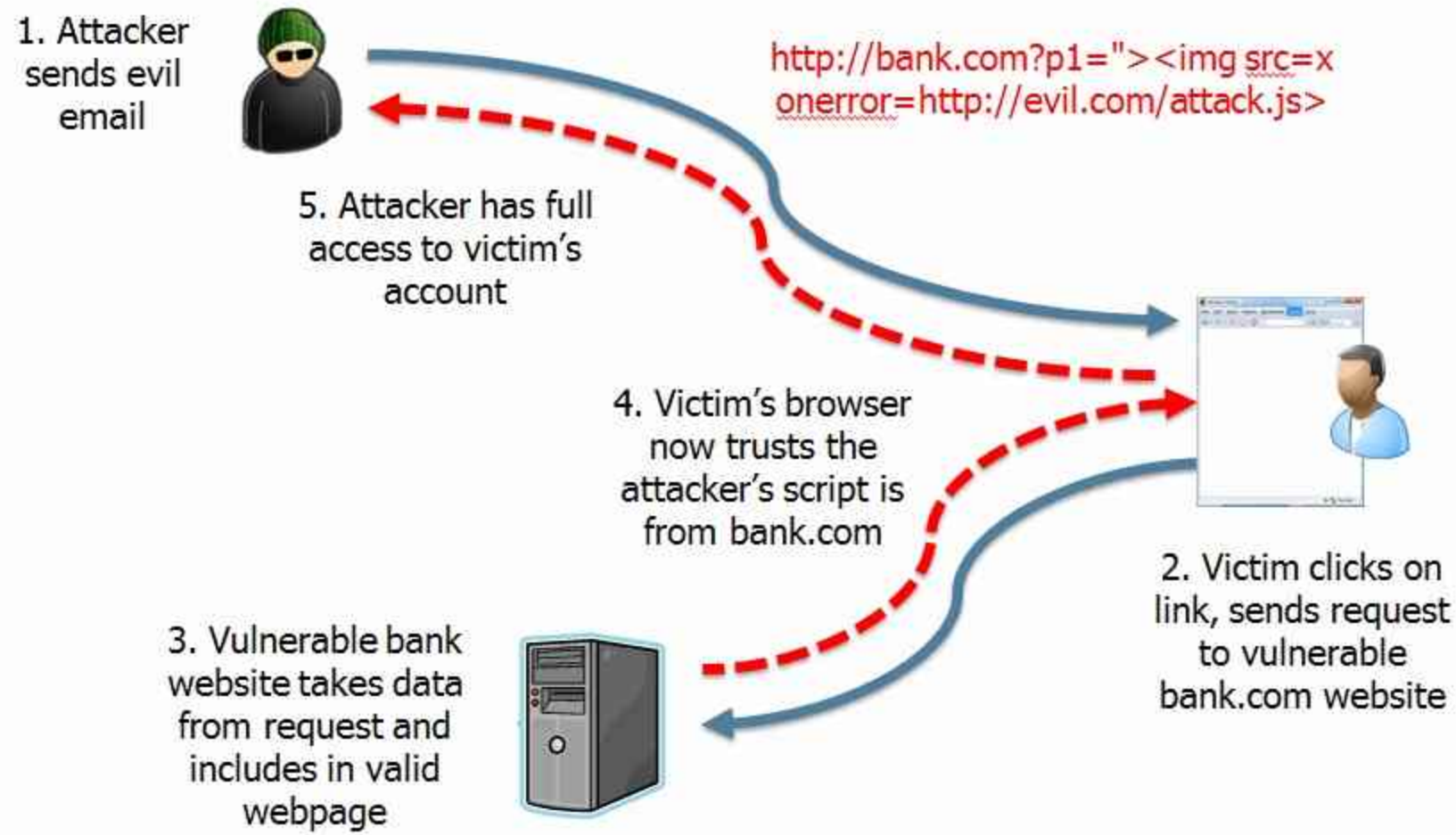
1. Data enters a Web application through an untrusted source, most frequently a web request.
2. The data is included in dynamic content that is sent to a web user without being validated for malicious content.

- The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash, or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data, like cookies or other session information, to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

- **Stored XSS Attacks**
- Stored attacks are those where the injected script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc. The victim then retrieves the malicious script from the server when it requests the stored information. Stored XSS is also sometimes referred to as Persistent or Type-I XSS.
- **Blind Cross-site Scripting**
- Blind Cross-site Scripting is a form of persistent XSS. It generally occurs when the attacker’s payload saved on the server and reflected back to the victim from the backend application. For example in feedback forms, an attacker can submit the malicious payload using the form, and once the backend user/admin of the application will open the attacker’s submitted form via the backend application, the attacker’s payload will get executed. Blind Cross-site Scripting is hard to confirm in the real-world scenario but one of the best tools for this is XSS Hunter.

- **Reflected XSS Attacks**
- Reflected attacks are those where the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other website. When a user is tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious site, the injected code travels to the vulnerable web site, which reflects the attack back to the user’s browser. The browser then executes the code because it came from a “trusted” server. Reflected XSS is also sometimes referred to as Non-Persistent or Type-II XSS.

How Does Reflected XSS Work?



AGENDA

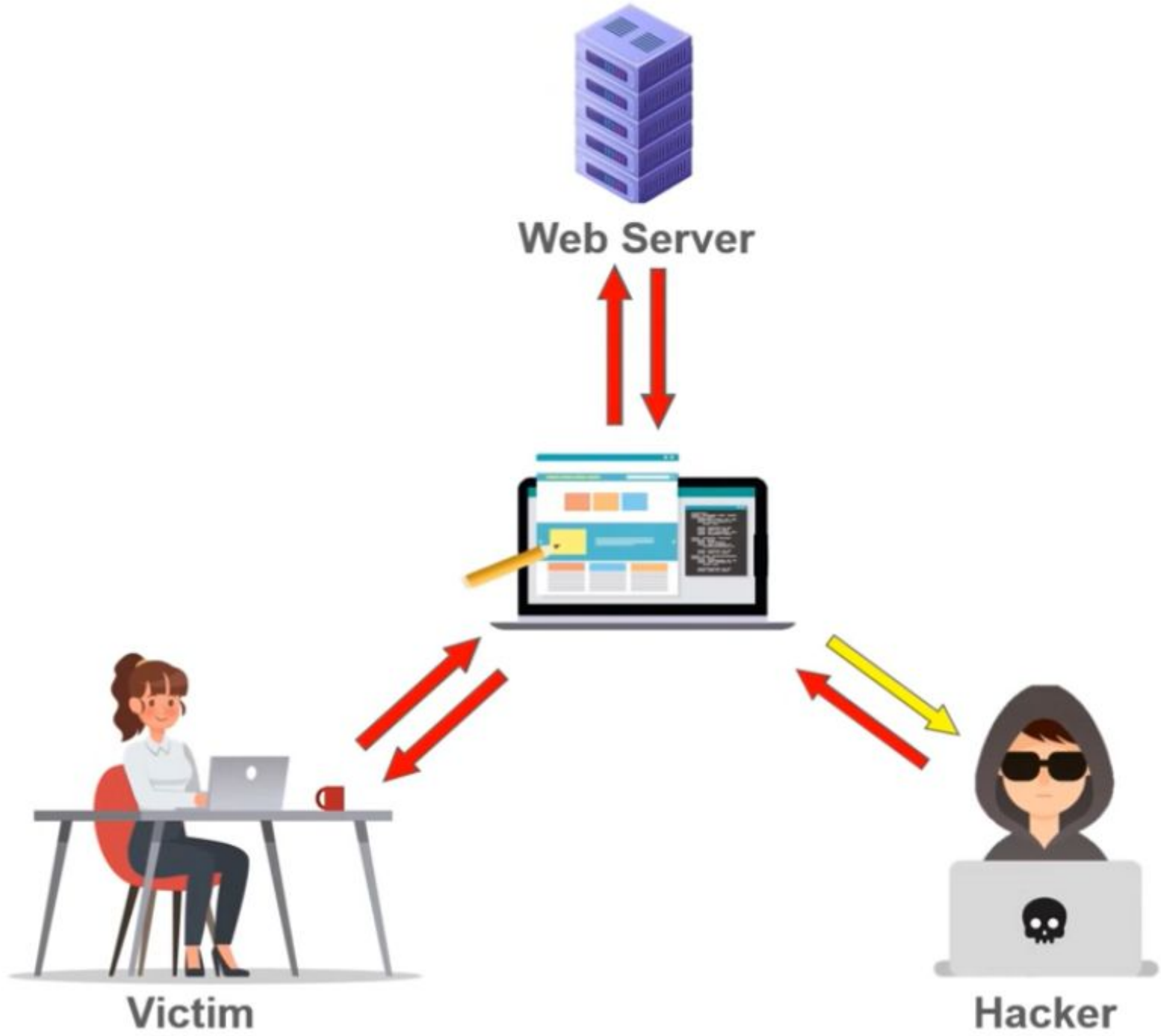
WHAT IS CROSS SITE SCRIPTING?
HOW CROSS SITE SCRIPTING WORKS?
TYPES OF CROSS SITE SCRIPTING
HOW TO USE CROSS SITE SCRIPTING?
HOW TO PREVENT CROSS SITE SCRIPTING?

WHAT IS CROSS SITE SCRIPTING?

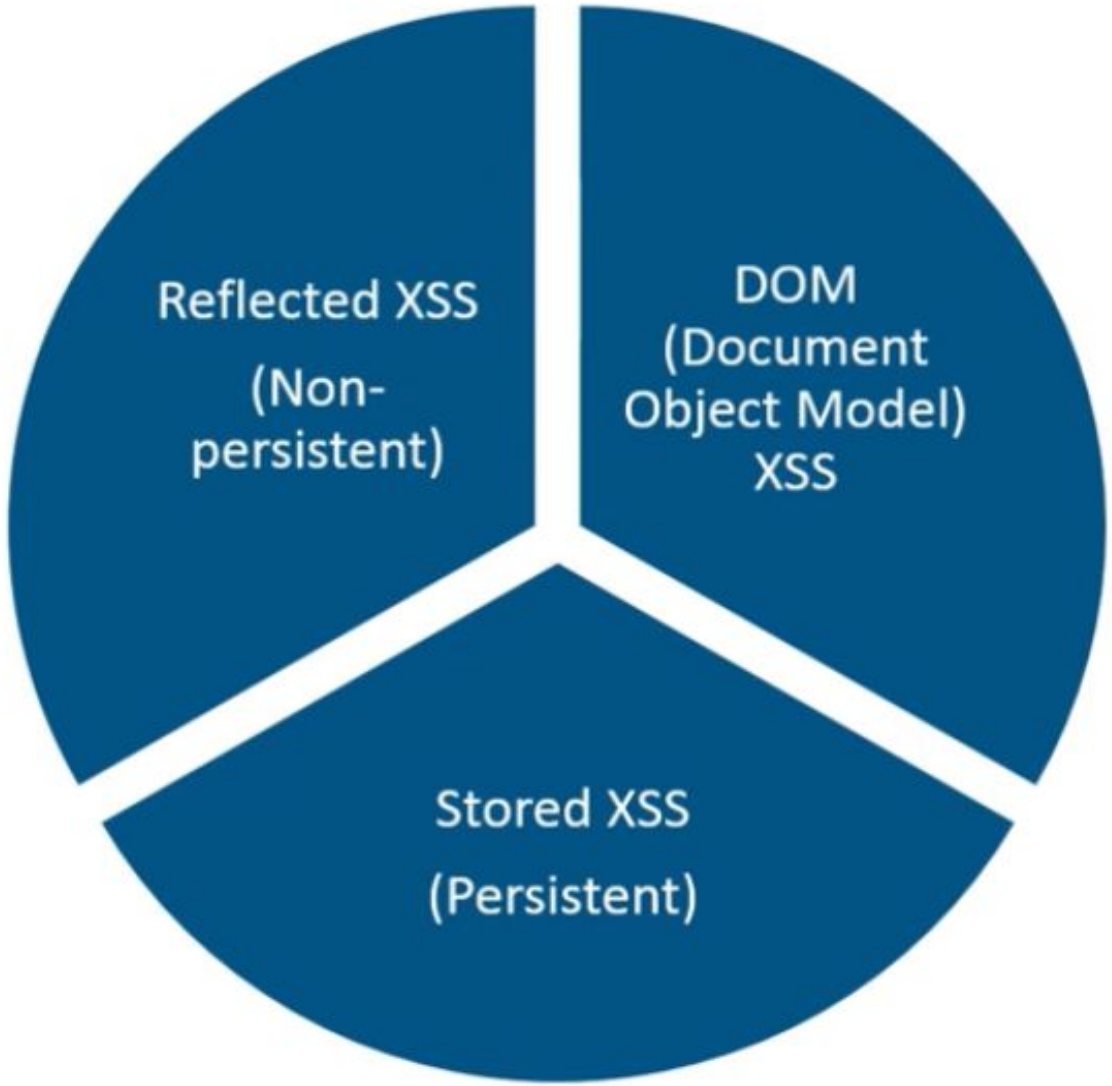
Cross Site Scripting (XSS) is a Code Injection attack executed on the client-side of a Web Application.

- Attacker injects malicious script through the web browser
- The malicious script is executed when the victim visits the web page or web server
- Steals Cookies, Session tokens and other sensitive information
- Modify the contents of the Website

HOW CROSS SITE SCRIPTING WORKS?



TYPES OF CROSS SITE SCRIPTING



HOW TO USE CROSS SITE SCRIPTING?

Reflected XSS (Non-persistent)

- Script is executed on the victim side
- Script is not stored on the server

Stored XSS (Persistent)

- Script is stored and executed on the server
- Executed every time the malicious site is requested

DOM (Document Object Model) XSS

- Client side attack. Script is not sent to the server
- Legitimate Server script is executed followed by Malicious script

HOW TO PREVENT CROSS SITE SCRIPTING?

- User input Escaping
- Consider all input as a threat
- Data Validation
- Sanitize data
- Encode output
- Use right response headers
- Use Content Security Policy

Cross-site Request Forgery

CSRF(Cross-site request forgery)

Problem | Overview

- CSRF is an OWASP Top 10 vulnerability but it's not as well understood as many others
- Many struggle with how to validate it
- Customers have difficulty explaining to management why it's important to fix

Basics | Description

"Cross-site Request Forgery is a vulnerability in a website that allows attackers to force victims to perform security-sensitive actions on that site without their knowledge."

Basics | Description

- What do we mean by “sensitive actions”?
- How do attackers “force” victims to perform them?
- And how do the victims not know it’s happening?

Basics | Description

1. The target is a sensitive operation in the application, e.g. UpdateSalary.aspx, that's able to be tricked into executing.

2. Victims can be forced to execute this action through any method that gets them to load a resource automatically, e.g. img tag, script tag, onload form submit, etc. Note: credentials go with all requests!

3. These happen unknowingly because the actions are performed by the victim's browser, not by the victim explicitly.

Basics | Trust Abuse

Both XSS and CSRF are possible due to abused trust relationships:

- In XSS the browser will run malicious JavaScript because it was served from a site (origin) it trusts

- In CSRF the server will perform a sensitive action because it was sent by a client that it trusts