Greetings of the day!!

“Shielding your Digital Assets: Web Application Security and its significance”

SUPRA

Digitalization vs Digitization

Converting Analog information into digital
Moving existing processes into digital tech

India becomes 7th target country suffering from web application attacks

- 70 Indian government, private websites face international cyber attacks

Orchestrated by the hacktivist group DragonForce Malaysia, the series of attacks targeted the Indian embassy in Israel, National Institute of Agriculture Extension Management, and e-portal of the Indian Council of Agriculture Research, among other online platforms.

- All India Institute Of Medical Science (AIIMS), was hit by a ransomware attack.

- 'Hacktivist Indonesia' claims to attack 12,000 Indian govt websites.

- 1100+ Security bugs in Indian Government Websites and Servers compromised
- Low-priority bugs = 471, with open-redirect, CRLF injection, cache storage, server-version disclosure, server-status, and PHP info-files leaking
- Medium-priority bugs = 495, most prevalent --> Zip-backup files (sensitive data such as DB Credentials, Backend Code, and more), PHP-backup files, reflected XSS, common CVEs.
- High-priority bugs = 87 detected include Firebase data exposure, credential file disclosure, server config files, arbitrary code execution, local and remote file inclusion bugs, and SQL injection (injection).
- Critical-priority bugs = 38, with remote-code execution, OS Command Injections, and other CVE-IDs being the most frequent. It provided me with complete access to the server.

- **What is Web Application?**

Stored on remote server and delivered over the internet through browser interface.

Browsers examples: Google Chrome, Apple Safari, Microsoft Edge, Firefox, Samsung Internet, Opera, Brave, Vivaldi, DuckDuckgo…….

- **How many web browsers exist?**

- **What was the first web browser?**

- Is a Web Browser a Web Application?

- What is the difference between web browser, web application and search engine?

- Web Application Security - Key Web Technologies?

  Browsers, HTML & CSS, Web Development Frameworks, Programming Languages, Protocols, API, Data formats, Client (or Client-side), Server (or Server-side)

- **Browsers:** Browsers request information and then they show us in the way we can understand. Think of them as the interpreters of the web.

- **HTML & CSS:** HTML makes sure web browsers know what to show once they receive the request. CSS - Cascading Style Sheets describe how HTML elements are to be displayed on the screen. Allows you to create Text effects, page transitions, image hover effects and more.

- **Web development frameworks: allows to go beyond simple or mundane tasks.**

Example: Angular -designed specifically for developing dynamic web applications. Easily create front-end based applications without needing to use other plugins.

Ruby on Rails: server-side website technology that makes app development much easier and faster. Popular websites written with Ruby include Basecamp, Ask.fm, GitHub, 500px

Django - most popular frameworks written in Python and follows MVC architecture.

- **Programming Languages:**

  Javascript – used by all web browsers, Meteor, and lots of other frameworks.

  CoffeeScript – a “dialect” of JavaScript. It is viewed as simpler but it converts back into JavaScript.

  Python – used by the Django framework as well as in the majority of mathematical calculations.

  PHP – used by WordPress to create WYSIWYG editors. It’s also used by Facebook, Wikipedia, and other major sites.

  Java – used by Android and a lot of desktop applications.

- **Protocols:** instructions on how to pass information back and forth.

  HTTP - protocol requests the website from the server and then receives a response with the HTML, CSS, and JavaScript of the website.

  DDP - Uses WebSockets to create a consistent connection between the client and the server. As a result of that, you get website updates in real-time without having to refresh the browser.

  REST - Used mostly for API’s, this protocol has standard methods like GET, POST, and PUT that let information be exchanged between applications.

- **API:** Application Programming Interface allows other developers to use some of the app's functionality without sharing the code. The endpoints are exposed by the developers while the API can control access with an API key. Examples of well-made APIs are those created by Facebook, Twitter, and Google for their web services.

- **Data formats:** Data is stored in a structure called a data format.
  - JSON – JavaScript Object Notation is a syntax for storing and exchanging data (just like XML). It is currently becoming the most popular data format out there.
  - XML – Predominantly used by Microsoft systems, it used to be the most popular data format
  - CSV – is data formatted by commas; for example Excel data

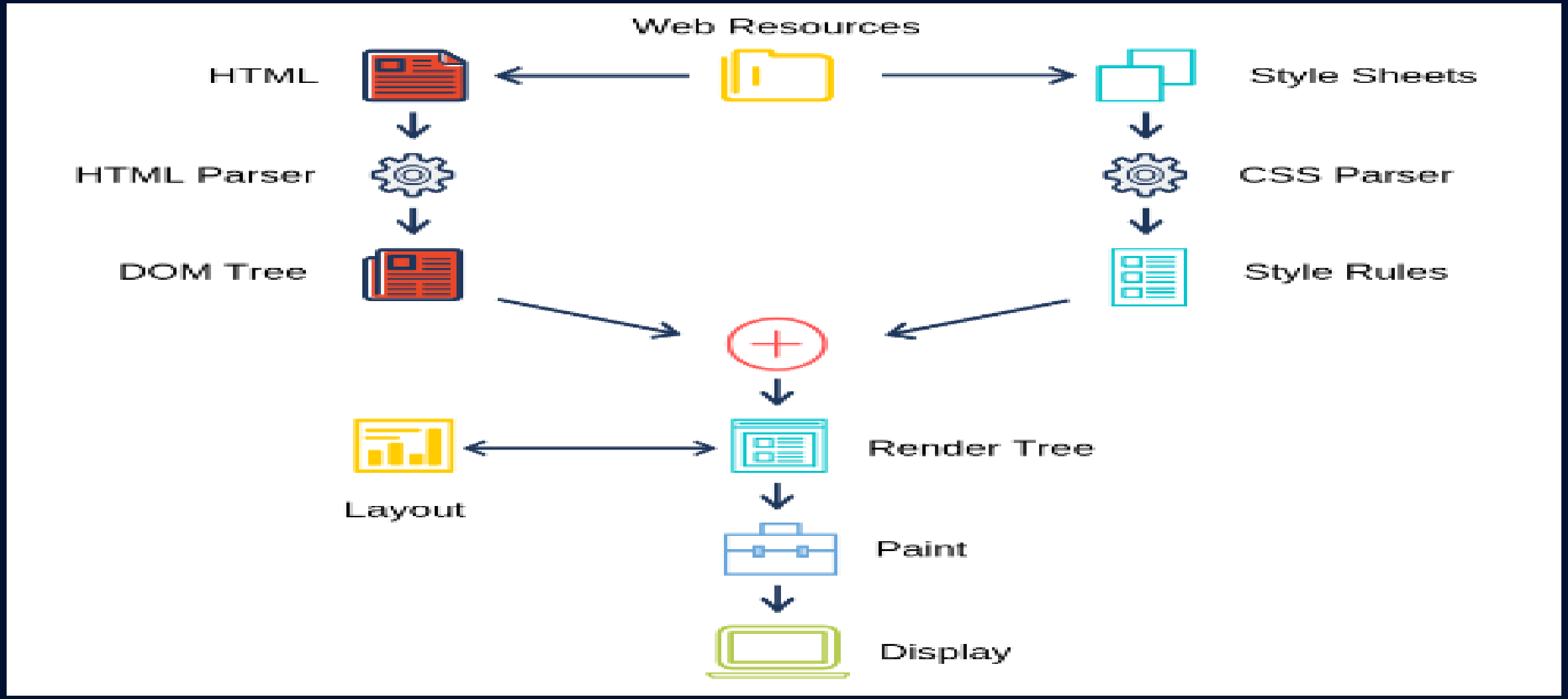
- **Client-side or Client:**
  - Each user of an application is called a client. Clients can be computers, mobile devices, tablets etc. Usually, multiple clients are interacting with the same app stored on a server.

- **Server-side or Server:**
  - The application code is usually stored on the server. The clients make requests to the servers. The servers then respond to those requests after gathering the requested information.

- **How does Web Browser work?**

5 steps:

- Handling user input
- Sending a URL request
- Preparing a renderer process
- Committing navigation
- Rendering page


Web Resources
HTML
Style Sheets
HTML Parser
CSS Parser
DOM Tree
Style Rules
Render Tree
Layout
Paint
Display

- Rendering engine subsystem processes data from the network layer and displays web content on the screen.
- It can process HTML, XML and Image files. Using extensions can process other file types too.
- Render engines are usually written in C++
- Chrome and Opera both use Blink, Firefox uses Gecko, Internet Explorer uses Trident, Edge uses EdgeHTML, Safari uses WebKit.
- Rendering engines parse web resources. For example, a HTML parser converts a HTML template into an object called the DOM tree.

- Rendering engine subsystem processes data from the network layer and displays web content on the screen.
- By default, it can process HTML, XML and Image files. However, it can be extended to accommodate various data types via extensions.
- Many render engines are available, and are usually written in C++.

Examples include:

- Chrome and Opera both use Blink
- Firefox uses Gecko
- Internet Explorer uses Trident
- Edge uses EdgeHTML
- Safari uses WebKit

With rendering engines, web resources are parsed. For example, a HTML parser converts a HTML template into an object called the DOM tree.

Exploiting dynamic rendering engines to take control of web apps

Leveraging weaknesses in Rendertron and other headless renderers

- Dynamic rendering is a technique used to serve prerendered web site pages to crawlers (e.g., Google search engine, Slack or Twitter bots, etc.)
- PWAs (progressive web applications) and SPAs (single page applications) that do most of the work in the client's browser and use JavaScript to generate the contents of the page on the fly.
- This technology has many advantages and can be effective at creating a sleek UI and UX, but at the same time, it is not SEO friendly, because crawlers and bots are not developed to render or understand JavaScript.
- common ways to help bots get valid HTML content is to open a requested page in a headless browser instance on the server, such as Puppeteer or Playwright, get the resulting HTML, strip parts that are not intended to be crawled and return it. This approach is called dynamic rendering and is promoted by Google as one of the possible ways to serve content.

Security Hacks:

- Nagpur unit making military weapons hit by hackers — ‘data up for sale’ -- Hacker group BlackCat or ALPHV

- Transparent Tribe, a persistent threat group that originated in 2013 in Pakistan, has been targeting Indian government and military entities.

  The Pakistan-based group (dubbed as APT36) is using a malicious file titled "Revision of Officers posting policy" to lure the Indian Army into compromising their systems.

- The personal website of Minister of State (Home) G Kishan Reddy -- kishanreddy.com -- was targeted by self-claimed Pakistani hackers. The website was hacked on Independence Day and the hackers had put messages related to Free Kashmir Pakistan, and also warned the Government of India.

OWASP ZAP	Burp Suite	Metasploit
W3af	Sqlmap	OWASP
Netsparker	Nmap	Nikto
Veracode	SonarQube	Checkmarx
Qualys	Rapid7	Security AppScan
Invicti	Dynamic application sec...	Nessus
Fortify Software	Skipfish	Synopsys

Top 10 Web Application vulnerabilities

1. Broken access control

2. Cryptographic failures

3. Injection

4. Insecure design

5. Security misconfiguration

6. Vulnerable and outdated components

7. Identification and authentication failures

8. Software and data integrity failures

9. Security logging and monitoring failures

10. Server-side request forgery (SSRF)

Broken access control:

Access control limits what users can access, restricting them to resources within their assigned permissions. Failure leads to unauthorized information disclosure, modification, or data destruction.

Access control vulnerabilities include:

- Violating the principle of least privilege by giving all users access to resources intended for specific roles, users, or permissions groups.
- Bypassing access control checks through URL modification, internal application state modification, or HTML page modification. An API attack tool can also be used to bypass access control checks.
- Missing API access controls for PUT, POST, or DELETE.
- Elevation of privilege, where an attacker has access as a user without logging in or can perform admin-level functions from a lower-privilege user account.

Protection methods include:

- ✓ Limit access to resources with access control.
- ✓ Access control works in a trusted server environment where data cannot be modified by the attacker. Access should be denied by default, unless the item is a public resource.
- ✓ Reuse access control mechanisms throughout the application
- ✓ Enforce application business limits
- ✓ Disable directory listings for web servers
- ✓ Rate-limit API and controller access

Cryptographic failures:

A broad symptom of a breakdown or deficiency in cryptography, which can lead to system compromise or sensitive data exposure.

Cryptographic failures include:

- Unenforced encryption in the browser where HTTP security headers are missing.
- Invalid server certificate trust chain.
- Data transmitted in clear text over browser protocols such as FTP, SMTP, and HTTP.
- Deploying weak cryptographic algorithms and protocols; ignoring weak cryptographic algorithms in legacy code.

Prevention:

Preventing cryptographic failure depends on application functionality and the type of data used.

- ✓ Classify data that an application has processed, transmitted, or stored.
- ✓ Store only the data needed, then discard once the action is completed.
- ✓ Encrypt all data during transmission and at rest.
- ✓ Avoid using legacy protocols to transport sensitive data.

Do you reckon SQL Injection, XSS are SAME or DIFFERENT?

Injection:

-source-code review
-includes cross-site scripting, SQL injection, and XML injection among many others

Applications are vulnerable to injection when:

- User-entered data is accepted without validation, sanitization, or filtering.
- Hostile data is used to extract sensitive information.

Prevention

- ✓ Keeping queries and commands separate from data is critical to preventing attempts at injection:
- ✓ Ensure that escape syntax includes special characters for the interpreter being used.
- ✓ Use query controls to prevent unexpected input from performing unauthorized actions.
- ✓ Use prepared statements with parameterized queries through a safe API, separate from the interpreter.

Insecure design:

-Insecure design differs from insecure implementation. A failure to accurately assess business risk associated with the software or system under development leads to insufficient levels of security.

Prevention

Using a secure design methodology that evaluates threats and ensures code is designed and tested against known attack methods. Prevention includes methods that foster a secure development culture:

- ✓ Partner with application security professions to help evaluate and design controls around privacy and security. Use a secure development lifecycle.
- ✓ Components should be ready to use and design patterns that are secure.
- ✓ Apply threat modeling against access controls, key data flows, business logic, and critical authentication.

Security misconfiguration:

-Security misconfigurations can be caused by an array of inappropriately configured controls as well as other factors which contribute to application vulnerability.

Common misconfigurations:

- Misconfigured permissions for cloud services.
- Enabling unnecessary features, which may lead to needless opened ports, services, or incorrectly elevated privileges.
- Unchanged default account login credentials.

Prevention

Prevention begins with a thorough security configuration process that is repeatable across systems and preferably automated:

- ✓ Establish a repeatable security hardening process, ideally through automation, to ensure new environments are secured appropriately with every deployment.
- ✓ Use only what is needed. Uninstall or remove unneeded features and components.
- ✓ Deploy an automated process to review security settings across environments.

Vulnerable and outdated components:

-Unpatched and legacy components that remain in production well after vulnerabilities are discovered and disclosed can be a major risk.
-Applications can be vulnerable when they aren't running the latest software version.
-If it's unclear which library or component version is being used, the application may be vulnerable.
-Components that aren't scanned for vulnerabilities may also be at risk.

Prevention:

Establishing a patch-management process can help alleviate the potential for attack by closing vulnerabilities before they become an issue.

- ✓ Removal of unused or unnecessary libraries, components, frameworks, documentation, and files.
- ✓ Continual monitoring and inventory of server-side and client-side components.
- ✓ Use of only official libraries and sources through secure links.
- ✓ Monitoring for unsupported libraries and components that are no longer maintained or have reached end of life.

Identification and authentication failures:

-user identity, authentication, and session information aren’t confirmed before the user is permitted to access systems and data. Allowing weak passwords; using weakly hashed, plain-text password data stores; and allowing bots, which can perform automated attacks such brute-force and credential stuffing.

Prevention:

- ✓ Providing secure password storage and retrieval.
- ✓ Implementing multi-factor authentication.
- ✓ Avoiding deployment using default credentials, especially for administrative accounts.
- ✓ Limit exposure to account enumeration.

Software and data integrity failures:

-Trusting data and software updates without checking their integrity. Attackers have used the software supply chain to issue malware through seemingly legitimate software updates. Many systems use automated software update features that do not verify the integrity of updates.

Prevention:

- ✓ Using digital signatures or other verification methods to digitally sign software updates to ensure they've come from expected sources and have arrived intact.
- ✓ Verifying that third-party libraries and other dependencies originate from legitimate sources.
- ✓ Verifying that third-party resources contain no vulnerabilities by using automated security tools designed for the software supply chain.

Security logging and monitoring failures:

-Security monitoring and logs are essential to detect and mitigate an active breach.

Failures happen when:

- Logging doesn’t keep track of transactions with high value, login attempts, and failed login attempts.
- Errors and warnings generate unclear, inadequate, or no log entries.
- APIs and applications aren’t monitored for suspicious activities.
- Security logs are only available locally.
- Applications that can neither detect nor issue timely alerts for attacks in progress.

Prevention:

Ensure security controls are implemented where appropriate. Security controls should include the following:

- ✓ Login, access control, and server-side validation failures should be logged with user context to ensure malicious and suspicious activity can be preserved long enough to allow for analysis.
- ✓ Logs should be generated in an appropriate format for log management tools to read.
- ✓ Enable monitoring and alerting for suspicious activities.
- ✓ Adopt an incident response and mitigation plan.

Server-side request forgery (SSRF):

-These flaws happen when web applications fetch user-requested remote sources without verifying the destination first.
-Specific requests can be sent to the application through the unexpected source.
-Applications commonly fetch URLs to enable easier task-switching for end-users, often keeping them in the application while providing access to another feature through the fetched URL. Ever-increasing cloud architecture complexity means SSRF is occurring at a higher frequency.

Prevention:

- ✓ Occurs at the network and application levels.
- ✓ Protect networks by using network segmentation to separate remote resources.
- ✓ Block other, nonessential traffic with “deny-by-default” policies.
- ✓ Login, access control, and server-side validation failures should be logged with user context to ensure malicious and suspicious activity can be preserved long enough to allow for analysis.

SSRF contd…:

Application protection methods should include:

- ✓ Data input sanitization, validation, and filtering.
- ✓ Disabling HTTP redirection at the server level
- ✓ Ensuring server responses received conform to expected results. Raw responses from the server should never be sent to the client.

Questions?

Thank you!!!

You can reach me on

PrasannaGuntur@supratech.xyz

Ph: +91 99028 49444