security practices

 Summary of popular security practices to implement in web development.

Recall

- Why are security practices important?
- What are some best practices?

Overview

With the trend toward using web-based applications for … well, basically everything, more attention is being placed on `cybersecurity`, a term we've come to know since the very early 1990s and the advent of the web. Because we are using web applications for so many things and passing so much sensitive information around via so many different types of online channels, we should next be obliged to also take a hard stance at protecting and securing that information.

To date, no web technology has proven itself invulnerable beyond all doubt. New threats pop up every single day that require at least some change or improvement in implementing countermeasures and general web-focused security. To improve the overall quality of web applications, developers should abide by these rules.

11 Tips Develops Should Remember to Protect and Secure Information

Maintain Security During Web App Development

Before you run out and hire a team of security consultants, realize that you can maintain security in your web applications during the actual development of those tools.

Require Injection Protection and Input Validation (User Input is not your friend)

A good rule of thumb is to consider all input to be hostile until proven otherwise. Input validation is done so that only properly-formed data passes through the workflow in a web application. This prevents bad or possibly corrupted data from being processed and possibly triggering the malfunction of downstream components.

Some types of input validation are as follows:

- Data type validation (ensures that parameters are of the correct type: numeric, text, et cetera).
- Data format validation (ensures data meets the proper format guidelines for schemas such as JSON or XML).
- Data value validation (ensures parameters meet expectations for accepted value ranges or lengths).

The basic thing to keep in mind is that you want to validate inputs with both a `syntactical` as well as a `semantic` approach. Syntactic validation should enforce correct syntax of information (SSN, birth date, currency or whole numbers) while semantic validation should enforce the correctness of their values within a very specific business context (end date is greater than the start date, low price is less than high price).

Encrypt Data

Encryption itself does not prevent interference in transmit of the data but obfuscates the intelligible content to those who are not authorized to access it.

Not only is encryption the most common form of protecting sensitive information across transit, but it can also be used to secure data “at rest” such as information that is stored in databases or other storage devices.

When using Web Services and APIs you should not only implement an authentication plan for entities accessing them, but the data across those services should be encrypted in some fashion. An open, unsecured web service is a hacker’s best friend (and they have shown increasingly smarter algorithms that can find these services rather painlessly).

Use Exception Management

Another development-focused security measure is proper exception management. You would never want to display anything more than just a generic error message in case of a failure. Including the actual system messages verbatim does not do the end-user any good, and instead works as valuable clues for potentially threatening entities.

When developing, consider that there are generally only three possible outcomes from a security standpoint:

1. Allow the operation.
2. Reject the operation.
3. Handle an exception.

Usually, in the case of an exception or error, you will revert to rejecting the operation. An application that fails securely will prevent operations from unintentionally being allowed. For example, if an ATM failed you would prefer it to display a simple, friendly message to the user (not spill money out onto the ground).

Apply Authentication, Role Management, and Access Control

Implementing effective account management practices such as strong password enforcement, secure password recovery mechanisms and multi-factor authentication are some strong steps to take when building a web application. You can even force re-authentication for users when accessing more sensitive features.

When designing a web application, one very basic goal should be to give each and every user as little privileges as possible for them to get what they need from the system. Using this principle of minimal privilege, you will vastly reduce the chance of an intruder performing operations that could crash the application or even the entire platform in some cases (thus adversely affecting other applications running on that same platform or system).

Other considerations for authentication and access control include things such as password expiration, account lock-outs where applicable, and of course SSL to prevent passwords and other account-related information being sent in plain view.

Hosting/Service-Focused Measures

Equally important as development-focused security mechanisms, proper configuration management at the service level is necessary to keep your web applications safe.

Avoid Security Misconfigurations

Given the endless amount of options that contemporary web server management software provides, this also means that there are endless ways to really muck things up:

- Not protecting files/directories from being served
- Not removing default, temporary, or guest accounts from the webserver
- Unnecessarily having ports open on the webserver
- Using old/defunct software libraries
- Using outdated security level protocols
- Allowing digital certificates to expire

Have a well-documented process for not only setting up new websites but also for setting up the web servers and the software used to serve those websites.

The modular nature of web server features allows for more granular control over resources and security. Although, this can make your applications less secure if you are not careful when using them. Be extremely cautious and careful when managing more high-risk security options and features.

Implement HTTPS (and redirect all HTTP traffic to HTTPS)

We had discussed encryption previously with development-focused approaches. Encryption at the service level is also extremely helpful (and sometimes necessary) preventative measure that can be taken to safeguard information. This is typically done by using HTTPS (SSL or Secure Sockets Layer).

SSL is a technology used to establish an encrypted link between a web server and a browser. This ensures that the information passed between the browser and the webserver remains private. SSL is used by millions of websites and is the industry standard for protecting online transactions.

In addition, blanket use of SSL is advised not only because it simply will then protect your entire website, but also because many issues can crop up with resources like stylesheets, JavaScript or other files if they aren’t referenced via HTTPS over an SSL.

Include Auditing and Logging

We are also concerned with auditing and logging at the server level. Thankfully, much of this is built into the content serving software applications such as IIS (Internet Information Services) and is readily accessible should you need to review various activity-related information.

Not only are logs often the only record that suspicious activity is taking place, but they also provide individual accountability by tracking a user’s actions.

Different from Error Logging, Activity or Audit Logging should not require really much setup at all since it is generally built into the webserver software. Be sure to leverage it to spot unwanted activities, track end user’s actions, and to review application errors not caught at code-level.

In extremely rare cases, logs may be needed in legal proceedings. As I am sure you well know, in these cases the handling of the log data is critical.

Use Rigorous QA and Testing

If your situation at all allows you to, utilizing a third-party service that specializes in penetration testing or vulnerability scanning as an addition to your own testing efforts is a great idea. Many of these specialized services are very affordable.

It is better to be overly cautious when possible, and not rely on only your own in-house quality assurance process to uncover every little hole in every little web application you are using. Adding another layer of testing to catch a few holes here and there that were perhaps not identified by other means of testing is never a bad thing.

To make security upgrades and routine testing efforts go more smoothly, have a well-defined and easily replicable process in place, as well as a thorough inventory of all web applications and where they exist. Nothing is more frustrating than trying to fix security bugs with a specific code library, but to only then have no idea which web applications are even using it!

Your web applications should also be free of any vulnerabilities or breaches that would fail any PCI or HIPAA guidelines. To be certain of this, you should be diligent in all these areas with your approach and design. Whenever possible, you should consult with a party that specializes in adherence to these guidelines so that you can be fully confident that you have everything in place to not only thwart attacks but to simply follow the rules put forth by governing agencies as well.

Be Proactive

Threats are constantly evolving.

You should have a well-defined blueprint for a security plan for all your sensitive web applications. This means prioritizing your more high-risk applications. It can be easier to identify if you have an inventory or repository of all the web applications that your business uses or provides to its end users.

As security threats evolve, so should your approach and plan for handling them. Increasingly sophisticated adversaries and ever-expanding soft spots as we turn to web applications to solve more and more of even our most tenable business needs is a concern that requires a full-time effort.

The current reality is that while you cannot exactly expect to avert all attacks, you should certainly aim to meet the challenge by building your own intel as a force multiplier. Get your leadership fully engaged and make sure you have ample resources applied to build an active defense to detect and respond to emerging security risks and hazards.

The web security landscape is changing constantly, and so must your strategy to traverse it.

Web Application Firewalls

A web application firewall (WAF) is designed to secure web applications from **application-layer attacks**. It offers robust protection against the most critical web application vulnerabilities, such as cross-site scripting, injection attacks, cross-site forgery, broken authentication, among others.

You can think of WAF as a shield between the web application and the client. It constantly monitors and inspects the HTTP traffic going in and out of web applications. If the traffic is found to be safe, WAF allows it to pass through. On the contrary, malicious traffic is blocked from web apps to prevent threats and attacks.

The web application firewall uses a set of rules, also known as policies, to differentiate between safe and malicious traffic. These policies are customizable and can be tailored to meet the unique needs of your web application.

Web application firewalls can be configured in multiple ways. The two most common types of WAFs are:

1. Hardware-based WAFs
2. Cloud-based WAFs

Both have their advantages and disadvantages. So choosing the appropriate option for yourself is a matter of understanding your unique business needs and making the decision accordingly.

Encrypt Sensitive Date in Transit

Data security is crucial for web applications. For example, when someone shares confidential information on your application, like personal details or bank credentials,

they expect that information to be safely delivered and stored on your web server. That’s where TLS steps in to help.

Transport layer security (TLS) encrypts the communication between client and server via HTTPS protocol. As a result of this encryption, your web application remains protected against data breaches. In addition, TLS also authenticates the parties exchanging the information to prevent any unauthorized data disclosure and modification.

TLS protocol has become a standard security practice in recent years. It is also helpful from the SEO standpoint since Google uses a secure connection as a ranking signal.

To implement TLS on your website, you need to buy a TLS certificate from a certificate authority. Then, install it on your origin server. One can recognize TLS encryption by the padlock icon that appears right before the URL in the address bar. Besides, if the URL begins with “HTTPS”, it’s also a sign that your browser is connected via TLS.

Utilize Pen Testing

Pen testing works on one principle: Hack your web app before hackers do.

It may sound outlandish at first, but it’s not. Here’s why.

If you can find vulnerabilities in your web application and take security measures to fix them, your chances of getting hacked in the future will drastically reduce.

That’s the idea behind penetration testing, popularly known as pen test or pen-testing. It’s a preventive measure to reduce, if not eliminate, cyber attacks.

In this **cybersecurity exercise**, cybersecurity experts, with permission, attempt to find and exploit vulnerabilities in your system.

They use different penetration tools like Nmap, Wireshark, Metasploit, etc., for this purpose. This simulated attack intends to test the effectiveness of your existing security policies and identify unknown vulnerabilities that hackers could exploit. It also discovers loopholes that have the potential to lead to data theft. Thus, the test reports help you identify vulnerabilities before hackers, helping you update your security solutions and patch vulnerabilities in time.

Include Security Practices in the Design and Development Phases

The majority of security incidents are caused due to defects in the design and code of the software. That's why integrating security practices in the application design and development phase is crucial.

When it comes to the design phase, some of the best security practices include performing threat analysis, implementing design principles like server-side validation to mitigate risks, and building a security test plan.

For secure coding, developers should be educated about the OWASP Top 10 vulnerabilities and the OWASP secure coding practices they can adopt to prevent those vulnerabilities. Developers should also make a habit of scanning their code to catch security vulnerabilities early in the development phase. They can integrate security tools into the DevOps pipeline to find any vulnerability that may have sneaked into their code. This will allow them to revise their code quickly and nip the problem in the bud.

OWASP has also worked actively to identify the best coding security practices that can be integrated into the software development lifecycle to mitigate the most common software vulnerabilities.

Cyber Security Framework

The last element on our best cyber security practices list is employing a cyber security framework. A cyber security framework is a set of standards, guidelines, and practices that an organization can follow to manage its cyber security risks. The framework aims to reduce the company's exposure to cyberattacks and identify the most prone areas to these attacks.

There are different types of cyber security frameworks. Some popular ones that dominate the market include the NIST cybersecurity framework, CIS, and ISO/IEC 27001. When it comes to choosing a cyber security framework for your organization, adopt the one that can protect the most vital areas of your business. You can also look at existing security standards prevalent in your industry for inspiration.