Republic of the Philippines
Nueva Ecija University of Science and Technology
Nueva Ecija, Philippines
ISO 9001:2015 Certified

College of Information and Communications Technology

Learning Module in ITWS-04 Web Systems Vulnerabilities

By:

Alexander S. Cochanco, MSIT

Angelito I. Cunanan, Jr., MSIT

TABLE OF CONTENTS

UNIT I. INTRODUCTION TO WEB VULNERABILITIES

Learning Objectives

At the end of the unit, the student will be able to:

1. Define and have an understanding of the terms threat, vulnerability, attack and web application vulnerability.
2. Identify the different kinds of attacks on web applications.
3. Define the good habits of a security-conscious developer.

THREATS, VULNERABILITIES AND ATTACKS

A ***threat*** is any potential event that could harm an asset, malicious or otherwise. In other words, any bad thing that can happen to your assets, is a threat.

A ***vulnerability*** is a weakness which allows for an attack. This may be due to poor design, configuration errors or improper and insecure coding techniques. Low input validation is an example of a weakness in an application layer which can lead to input attacks.

An ***attack*** is an action exploiting a vulnerability or making a threat. Examples of attacks include sending malicious input to an app, or flooding a network to attempt to deny service.

To sum up, a threat is a future occurrence that can adversely affect an asset, while the vulnerability in your system is exploited by a successful attack.

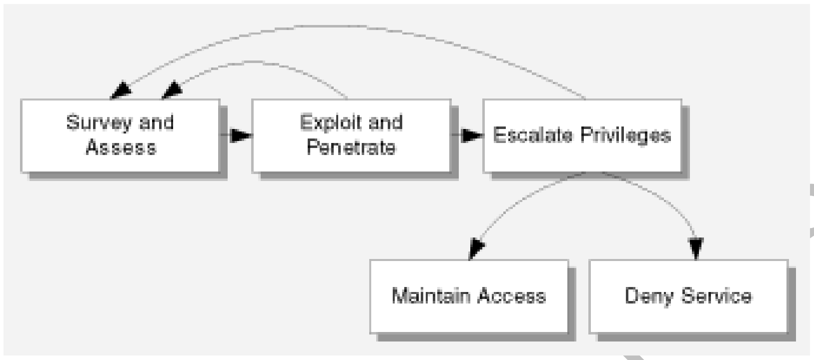
WHAT IS A WEB APPLICATION VULNERABILITY?

A ***web application vulnerability*** is a weakness or misconfiguration in a website or web application code that enables an attacker to gain some level of control of the site, and possibly the hosting server. Most vulnerabilities are exploited through automated means, such as vulnerability scanners and botnets.

Web application vulnerabilities involve a system flaw or weakness in a web-based application. They have been around for years, largely due to not validating or sanitizing form inputs, misconfigured web servers, and application design flaws, and they can be exploited to compromise the application's security. These vulnerabilities are not the same as other common types of vulnerabilities, such as network or asset. They arise because web applications need to interact with multiple users across multiple networks, and that level of accessibility is easily taken advantage of by hackers.

ANATOMY OF AN ATTACK

- Survey and Assess
- Exploit and Penetrate
- Escalate Privileges
- Maintain Access
- Deny Service



Survey and Assess

Surveying and assessing of the future target are performed in parallel. The first step normally taken by an intruder is to survey the possible target to define and assess its characteristics.

These characteristics can include its supported services and protocols along with possible vulnerabilities as well as entry points. To plan an initial attack, the attacker uses the information gathered in the survey and assess phase.

Exploit and Penetrate

Having assessed the potential target, the next move is to exploit and penetrate. If the network and host are completely protected, then the next platform for attack will be your application.

The easiest way for an attacker to get into an application is through the same entrance that legitimate users use, for example, through the logon page of the application or a page that does not require authentication.

Escalate Privileges

After attackers managed to enter an application or network by injecting code into the application or creating an authenticating session with the operating system, They will immediately try to escalate privileges. In particular, they are looking for administrative rights that are offered by accounts that are members of the Administrators group. They 're just searching for the high degree of rights the local network account provides.

A primary protection against privilege escalation attacks is the use of least privileged service accounts in the application.

Maintain Access

When an intruder has obtained access to a network, he takes steps to encourage future access and cover his or her tracks.

Popular approaches to encouraging potential access and making them easier include planting of backdoor programs or the use of an established account lacking strong security. Usually, covering tracks includes clearing logs, and hiding tools.

Log files should be secured, and should be periodically examined. Analysis of the log file will also show the early signs of an attempted break-in before the harm is done.

Deny Service

Attackers who are unable to get access also launch a denial-of-service attack to discourage anyone from using the device. For other attackers, their target from the beginning is the denial of service to the application.

WHAT KINDS OF ATTACKS ARE WEB APPLICATIONS VULNERABLE TO?

There are three possible scenarios where web applications are vulnerable to attacks: (a) ***when users provide information***, (b) ***when information is provided to users*** and (c) ***in other cases***.

When Users Provide Information

One of the most common types of web applications allows users to enter information. Later, this information may be stored and recovered. Right now, however, we are concerned simply with the data, imagined to be harmless, that people type in.

Human Attacks

Humans are capable of using any technology either in a helpful or harmful way. While you are generally not legally responsible for the actions of people who use your online applications, being a good netizen requires you to take some level of responsibility for them. Moreover, in practical terms, dealing with malicious users can consume a significant amount of resources, and their actions can do real damage to the reputation of the site that you have worked so hard to create.

Most of the following behaviors could be considered as annoyances rather than attacks, because they do not involve an actual breach of the security of the application. However, these disruptions are still violations of the policy and social contract, and to the extent that they can be discouraged by the programmer, they are still violations of the social contract.

- ***Abuse of Storage***: With the popularity of weblogging and message board systems, a lot of sites allow their users to keep a journal or post photos. Sites like these may attract abusers who want to store, without fear that it can be traced back to their own servers, not journal entries or photos but rather illegal or inflammatory content. Or abusers may simply want free storage space for large quantities of data that they would otherwise have to pay for.

- ***Sock Puppets***: Any site that solicits user opinions or feedback is vulnerable to the excellently named Sock Puppet Attack, where one physical user registers under either a misleading alias or even a number of different aliases in order to sway opinion or stuff a ballot. Posters of fake reviews on Amazon.com are engaging in sock puppetry; so are quarrelsome participants on message boards who create multiple accounts and use them to create the illusion of wide-ranging support for a particular opinion. A single puppeteer can orchestrate multiple conversations via different accounts. While this sort of attack is more effective when automated, even a single puppeteer can degrade the signal-to-noise ratio on an otherwise interesting comment thread.

- ***Defamation***: Related to sock puppetry is the attacker's use of your application to post damaging things about other people and organizations. Posting by an anonymous user is usually no problem; the poster's anonymity degrades the probability of its being believed, and anyway it can be removed upon discovery. But an actionable posting under your own name, even if it is removed as soon as it is noticed, may mean that you will have to prove in court that you were not the author of the message. This situation has progressed far enough so that many lists are now posting legal disclaimers and warnings for potential abusers right up front on their lists.

- ***Griefers, trolls and pranksters***: While possibly not quite as serious as the malicious liars described previously, the class of users commonly known as griefers or trolls or pranksters are more annoying by a factor of 10, and can quickly take the fun out of participating in a virtual community. Griefers are users who enjoy attacking others. The bullies you find as a new user in any online role-playing game are griefers, who, hiding behind the anonymity of a screen name, can be savagely malicious. Trolls, on the other hand, enjoy being attacked as much as attacking. They make outrageous assertions and post wild ideas just to get your attention, even if it's negative. Pranksters might insert HTML or JavaScript instructions into what should have been plaintext, in order to distort page appearance; or they might pretend to be someone else; or they might figure out some other way to distract from what had been intended to be serious business. These users destroy a community by forcing attention away from ideas and onto the personalities of the posters

Automated Attacks

Attacks in this class exploit the power of computers to amplify human effort. These scripted attacks, or robots, slow down services, fill up error logs, saturate bandwidth, and attract other malicious users by advertising that the site has been compromised. They are particularly dangerous because of their efficiency.

- ***Worms and viruses***: Probably the most prominent form of automated attack, and certainly the most notorious, is the worm, or virus, a small program that installs itself onto your computer without your knowledge, possibly by attachment to an email message, or by inclusion into a downloaded application. There is a small technical difference between the two; a worm is capable of existing by itself, whereas a virus must piggyback onto an executable or document file. The primary purpose of a worm or a virus is to duplicate itself by spreading to other machines. A secondary purpose is to wreak havoc on its host machine, deleting or modifying files, opening up backdoors (which outsiders might use to, for example, forward spam via your machine), or popping up messages of various sorts. A worm or virus can spread itself throughout the Internet within minutes if it uses a widespread vulnerability.

- ***Spam***: Spam is the sending of unsolicited (and often unwelcome) messages in huge quantities. It is an automated attack of a different sort, because it gives the appearance of being normal, albeit excessive, usage. It doesn't take long for users to be trained to recognize spam (or at least most spam); it takes servers (which carry out the hard work of transfer) quite a bit longer. But spam causes both to suffer from an unwelcome burden of service.

- ***Automated user input***: Other kinds of attacks automate the providing of input (supposedly from users) in various settings.

    - An organization running Internet portal services might decide to attract users by offering free services like email accounts or offsite storage. Such services are extremely attractive both to legitimate users and to abusers, who could, for example, use free email accounts to generate spam.

    - Political or public interest organizations might create a web application where users are allowed to express their preferences for candidates and issues for an upcoming election. The organization intends to let users' expressed preferences guide public opinion about which candidates are doing better than others, and which issues are of more interest to the public. Such online polls are a natural target for a malicious organization or individual, who might create an automated attack to cast tens or hundreds of thousands of votes for or against a particular candidate or issue. Such ballot stuffing would create an inaccurate picture of the public's true opinions.

    - An organization might create a website to promote interest in a new and expensive product, an automobile, a piece of electronic equipment, or almost anything. It might decide to create interest in the new product by setting up a sweepstakes, where one of the new products will be given away to a person chosen by random from among all those who register. Someone might create a robotic or automated attack that could register 10,000 times, thus increasing the chances of winning from, say, one in 100,000 (0.001%) to 10,000 in 110,000 (9.99%).

    - It is not at all unusual for certain kinds of web applications to provide the capability for users to leave comments or messages on a discussion board or in a guestbook. Stuffing content in these kinds of situations might seem innocuous, since that input seems not to be tied to actual or potential value. But in fact, messages containing little or nothing besides links to a website have become a serious problem recently, for they can inflate hugely that website's search engine rankings, which have all-too-obvious value. Even without this financial angle, automated bulk responses are an abuse of a system that exists otherwise for the common good.

- A similar potential vulnerability exists on any website where registration is required, even when no free services are offered. It may seem that there is little point in an attack that registers 10,000 fictitious names for membership in an organization, but one can't generalize that such abuse is harmless. It might, for example, prevent others from legitimate registration, or it might inflate the perceived power of the organization by misrepresenting its number of members. A competitor could attempt to influence an organization by providing bogus demographic data on a large scale, or by flooding the sales team with bogus requests for contact.

When Information is Provided to Users

It might seem that the creators of any web application whose business is to provide information to users would be happy when such information is actually provided. But given the uses to which such information can sometimes be put, giving out information is not always a pleasure, especially when it winds up being given to automated processes.

- ***Harvesting email addresses***: It's commonplace for websites to include an email address. Businesses may choose to offer users the possibility of contact by email rather than a form, thinking (probably correctly) that email is more flexible than a form. Individuals and organizations of various kinds will provide email addresses precisely because they want users to be able to communicate directly with key personnel. Such websites are open targets for automated harvesting of email addresses. Compiled lists of such addresses are marketed to spammers and other bulk emailers, and email messages generated from such stolen lists constitute a significant portion of Internet traffic.

- ***Flooding an email address***: Often a website displays only a specially crafted email address designed for nothing but receiving user emails, typically something like info@mycompany.com or contact@something.org. In this case, harvesting is less likely than simple flooding of a single email address. A quick examination of server email logs shows just how high a percentage of email messages to such addresses consists of spammers' offers of cheap mortgages, sexual paraphernalia, Nigerian bank accounts, and so forth.

- ***Screen scraping***: Enterprise websites are often used to make proprietary or special information available to all employees of the enterprise, who may be widely scattered geographically or otherwise unable to receive the information individually. Automated attacks might engage in what is known as screen scraping, simply pulling all information off the screen and then analyzing what has been captured for items of interest to the attacker: business plans and product information, for instance. Alternatively, attackers might be interested in using screen scraping not so much for the obvious content of a website page as for the information obliquely contained in URIs and filenames. Such information can be analyzed for insight into the structure and organization of an enterprise's web applications, preparatory to launching a more intensive attack in the future.

- ***Improper archiving***: Search robots are not often thought of as automated abusers, but when enterprise websites contain time-limited information, pricing, special offers, or subscription content, their archiving of that content can't be considered proper. They could be making outdated information available as if it were current, or presenting special prices to a wider audience than was intended, or providing information free that others have had to pay for.

In Other Cases

Malicious attacks on web applications sometimes aren't even interested in receiving or sending data. Rather, they may attempt to disrupt the normal operation of a site at the network level.

- ***Denial of Service***: Even a simple request to display an image in a browser could, if it were repeated enough times in succession, create so much traffic on a website that legitimate activity would be slowed to a crawl. Repeated, parallel requests for a large image could cause your server to exceed its transfer budget. In an extreme case, where such requests hog CPU cycles and bandwidth completely, legitimate activity could even be halted completely, a condition known as Denial of Service (DoS).

- ***DNS attacks***: The Domain Name System (DNS), which resolves domain names into the numerical IP addresses used in TCP/IP networking, can sometimes be spoofed into providing erroneous information. If an attacker is able to exploit a vulnerability in the DNS servers for your domain, she may be able to substitute for your IP address her own, thus routing any requests for your application to her server.

FIVE GOOD HABITS OF A SECURITY-CONSCIOUS DEVELOPER

Given all of these types of attacks and the stakes involved in building a web application, you'll rarely (if ever) meet a developer who will publicly say, "Security isn't important." In fact, you'll likely hear the opposite, communicated in strident tones, that security is extremely important. However, in most cases, security is often treated as an afterthought.

Think about any of the projects you've been on lately and you'll agree that this is an honest statement. If you're a typical PHP developer working on a typical project, what are the three things you leave for last?

Without pausing to reflect, you can probably just reel them off: usability, documentation, and security.

This isn't some kind of moral failing, we assure you. It also doesn't mean that you're a bad developer. What it does mean is that you're used to working with a certain workflow. You gather your requirements; you analyze those requirements; create prototypes; and build models, views, and controllers. You do your unit testing as you go, integration testing as components are completed and get bolted together, and so on.

The last things you're thinking about are security concerns. Why stop to sanitize user-submitted data when all you're trying to do right now is establish a data connection? Why can't you just "fix" all that security stuff at the end with one big code review? It's natural to think this way if you view security as yet another component or feature of the software and not as a fundamental aspect of the entire package.

Well, if you labor under the impression that somehow security is a separate process or feature, then you're in for a rude awakening. It's been decades (if ever) since any programmer could safely assume that their software might be used in strictly controlled environments: by a known group of users, with known intentions, with limited capabilities for interacting with and sharing the software, and with little or no need for privacy, among other factors.

In today's world, we are becoming increasingly interconnected and mobile. Web applications in particular are no longer being accessed by stodgy web browsers available only on desktop or laptop computers. They're being hit by mobile devices and behind-the-scenes APIs. They're often being mashed up and remixed or have their data transformed in interesting ways.

For these and many other reasons, developers need to take on a few habits—habits that will make them into more security-conscious developers. Here are five habits that will get you started:

- Nothing is 100% secure.
- Never trust user input.
- Defense in depth is the only defense.
- Simpler is easier to secure.
- Peer review is critical to security.

There are other habits, no doubt, but these will get you started.

Nothing Is 100% Secure

There's an old joke in computer security circles that the only truly secure computer is one that's disconnected from all power and communication lines, and locked in a safe at the bottom of a reinforced bunker surrounded by armed guards. Of course, what you've got then is an unusable computer, so what's the point, really?

It's the nature of the work we do: nothing we can ever do, no effort, no tricks, nothing can make your application 100% secure. Protect against tainted user input, and someone will try to sneak a buffer overflow attack past you. Protect against both of those, and they're trying SQL injection. Or trying to upload corrupt or virus-filled files. Or just running a denial of service attack on you. Or spoofing someone's trusted identity. Or just calling up your receptionist and using social engineering approaches to getting the password. Or just walking up to an unsecured physical location and doing their worst right there.

Why bring this up? Not to discourage or disillusion you, or make you walk away from the entire security idea entirely. It's to make you realize that security isn't some monolithic thing that you have to take care of—it's lots and lots of little things. You do your best to cover as many bases as you can, but at some point, you have to understand that some sneaky person somewhere will try something you haven't thought of, or invent a new attack, and then you have to respond.

At the end of the day, that's what security is – a mindset. Just start with your expectations in the right place, and you'll do fine.

Never Trust User Input

Most of the users who will encounter your web application won't be malicious at all. They will use your application just as you intended, clicking on links, filling out forms, and uploading documents at your behest.

A certain percentage of your user base, however, can be categorized as "unknowing" or even "ignorant." That last term is certainly not a delicate way of putting it, but you know exactly what we're talking about. This category describes a large group of people who do things without much forethought, ranging from the innocuous (trying to put a date into a string field) to the merely curious ("What happens if I change some elements in the URL, will that change what shows up on the screen?") to the possibly fatal (at least to your application, like uploading a resume that's 400 MB in size).

Then, of course, there are those who are actively malicious, the ones who are trying to break your forms, inject destructive SQL commands, or pass along a virus-filled Word document. Unfortunately for you, high enough levels of "stupidity" or "ignorance" are indistinguishable from "malice" or "evil." In other words, how do you know that someone is deliberately trying to upload a bad file?

You can't know, not really. Your best bet in the long run? Never trust user input. Always assume that they're out to get you, and then take steps to keep bad things from happening. At the very least, here's what your web application should be guarding against:

- Always check to make sure that any URLs or query strings are sanitized, especially if URL segments have significant meaning to the backend controllers and models (for example, if /category/3 passes an ID of 3 to a database query). In this instance, you can make sure that last URL segment is an integer and that it's less than 7 digits long, for example.

- Always sanitize each form element, including hidden elements. Don't just do this kind of thing on the front end, as it's easy to spoof up a form and then post it to your server. Check for field length and expected data types. Remove HTML tags.

- It's a good idea to accept form posts only from your own domain. You can easily do this by creating a server-side token that you check on the form action side. If the tokens match, then the POST data originated on your server.

- If you're allowing users to upload files, severely limit file types and file sizes. • If user input is being used to run queries (even if it's a simple SELECT), sanitize using mysql_escape_string() or something similar.

Defense in Depth Is the Only Defense

There will almost never be a scenario in which a single line of defense will be enough. Even if you only allow users to submit forms after they log in to a control panel, always sanitize form input. If they want to edit their own profile or change their password, ask them to enter their current password one more time. Don't just sanitize uploaded files, but store them using encryption so that they won't be useful unless decrypted. Don't just track user activity in the control panel with a cookie, but write to a log file, too, and report anything overly suspicious right away.

Having layered defenses is much easier to implement (and so much harder to defeat) than a single strong point. This is classic military defensive strategy —create many obstacles and delays to stop or slow an attacker or keep them from reaching anything of value. Although in our context we're not actually trying to hurt or kill anyone, what we are interested in is redundancy and independent layers. Anyone trying to penetrate one layer or overcome some kind of defensive barrier (authentication system, encryption, and so on) would only be faced with another layer.

This idea of defense in depth forces a development team to really think about their application architecture. It becomes clear, for example, that applying piecemeal sanitization to user input forms will probably just amount to a lot of code that is hard to maintain and use. However, having a single class or function that cleans user input and using that every time you process a form makes the code useful and used in actual development.

Simpler Is Easier to Secure

If you've been a developer for any amount of time, then you've probably run into lots of code that just makes your head hurt to look at. Convoluted syntax, lots of classes, a great deal of includes or requires, and any other techniques might make it hard for you to decipher exactly what is happening in the code. Small pieces that are joined together in smart, modular ways, where code is reused across different systems, are easier to secure than a bunch of mishmash code with HTML, PHP, and SQL queries all thrown into the same pot.

The same thing goes for security. If you can look at a piece of code and figure out what it does in a minute, it's a lot easier to secure it than if it takes you half an hour to figure out what it does. Furthermore, if you have a single function you can reuse anywhere in your application, then it's easier to secure that function than to try to secure every single time you use bare code.

Another pain point is when developers don't understand (or know about) the core native functions of PHP. Rewriting native functions (and thus reinventing the wheel) will almost always result in code that is less secure (or harder to secure).

Peer Review Is Critical to Security

Your security is almost always improved when reviewed by others. You can say to yourself that you will just keep everything hidden or confusing, and thus no one will be able to figure out how to bypass what you're doing, but there will come a day when a very smart someone will make your life intolerable.

A simple peer review process at regular intervals can keep bad things from happening to you and your application. Simple reminders to secure against cross-site scripting or suggestions for encryption approaches will almost always make your code more secure. Suggestions at the architectural level (get rid of all the repetition, use a single authentication function or class to handle that instead) will also make your application easier to maintain and probably more efficient and effective.

UNIT II. WEB VULNERABILITIES AND COUNTERMEASURES

Learning Objectives

At the end of the unit, the student will be able to:

1. Define different web vulnerabilities in web applications
2. Define and implement different countermeasures to prevent attacks on web applications

PHISHING

Phishing is a form of identity theft in which a scammer uses an authentic-looking email from a legitimate business to trick recipients into giving out sensitive personal information, such as a credit card, bank account, Social Security numbers or other sensitive personal information. The spoofed email message urges the recipient to click on a link to update their personal profile or carry out some transaction. The link then takes the victim to a fake website where any personal or financial information entered is routed directly to the scammer.

Various Types of Phishing Attacks

- ***Spear Phishing*** - Spear phishing is one of the common types of phishing attacks that are done by sending an email to a particular targeted individual. An attacker generally steals the user's information from social media sites like Linked-in, Facebook, etc. They use fake accounts to send emails that seem to be genuine to receivers. For example, attackers used to target employees of the finance department by sending them fake emails. In which, an attacker can try to impersonate as the victim's manager and can ask to transfer large amounts of money in a bank account.
- ***Whale Phishing*** - It is a form of phishing attack that is used to achieve big targets. Whale phishing is a technique to trick organizations and companies for stealing their confidential data. This type of scam generally happens to board members of the company. Attackers can simply target them, as it only requires the company's email id to deceive them.
- ***Deceptive Phishing*** - Nowadays, it is one of the most common types of phishing attacks. Deceptive phishing emails involves threatening messages to scare users by creating urgency. Attackers such as PayTM scammers send emails to customers and ask them to click on a link to rectify a mistake in their account. As the user clicks on the link, it takes them to a fake webpage which might look similar to that of an actual PayTM login page. Here user enters the login credential details, and this information can be further used for illegal activities.
- ***Pharming*** - It is a type of Phishing attack that hackers use to steal sensitive or personal information from the users on the internet. In this attack, the hacker uses malicious code injected into the user's computer system or the server that misdirects users to fraudulent websites without their consent.
- ***Dropbox Phishing*** - Some phishers do not use **'baiting'** to deceive their targets. Instead, they send attack emails to individuals or companies. They generally use common popular sites like Dropbox to target the users. For example, cybercriminals may start the attack by creating a fake sign-in Dropbox page which seems similar to

the original Dropbox site. By doing so, phishers try to confuse the victim and trick them to submit their personal information.

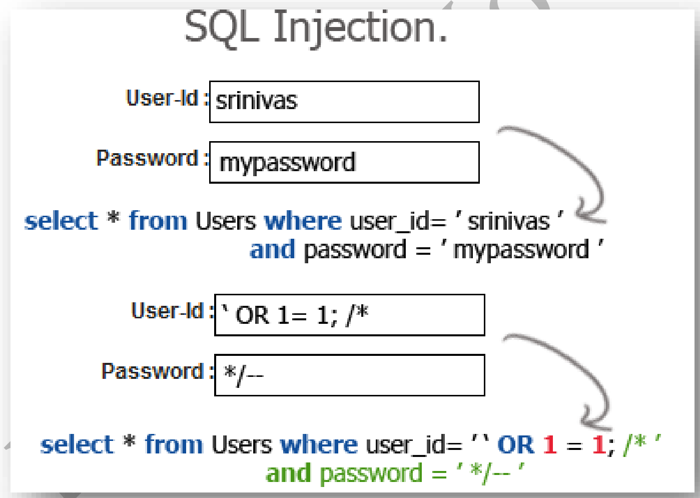
How to Prevent Phishing

- Lock down your browser with pop-up and phishing blockers
- Use multi-factor authentication where possible.
- Configure your email providers spam filter for maximum effectiveness.

SQL INJECTION

SQL injection is a code injection technique used to target data-driven applications, where malicious SQL statements are inserted into a data entry field for execution (e.g. dumping the contents of the database to the attacker).

SQL injection attacks allow attackers to spoof identities, exploit existing data, trigger repudiation problems such as voiding transactions or changing balances, allow full disclosure of all data on the network, kill or otherwise make data inaccessible, and become database server administrators.



User-Id : srinivas

Password : mypassword

```
select * from Users where user_id= ' srinivas '
          and password = ' mypassword '
```

User-Id : ' OR 1= 1; /*

Password : */--

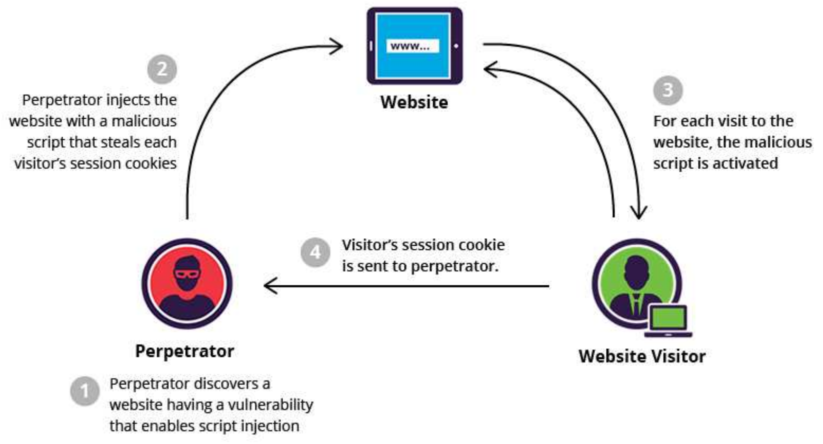
```
select * from Users where user_id= ' ' OR 1 = 1; /* '
          and password = ' */-- '
```

How to Prevent SQL Injection

- Parameterized Statements
- Object Relational Mapping
- Escaping Inputs
- Sanitizing Inputs

CROSS-SITE SCRIPTING

Cross-site scripting (XSS) is a type of vulnerability which is usually found in web applications. XSS helps attackers to inject scripts client-side to web pages accessed by other users


Website
www...
2
Perpetrator injects the website with a malicious script that steals each visitor's session cookies
3
For each visit to the website, the malicious script is activated
4
Visitor's session cookie is sent to perpetrator.
Perpetrator
Website Visitor
1
Perpetrator discovers a website having a vulnerability that enables script injection

When cybercriminals use cross-site scripting, they inject code on a site via form fields or other areas of user inputs in order to target website users. When the user's browser executes this code, attackers can hijack user sessions, covertly track session data, or even display spam content on an otherwise legitimate site.

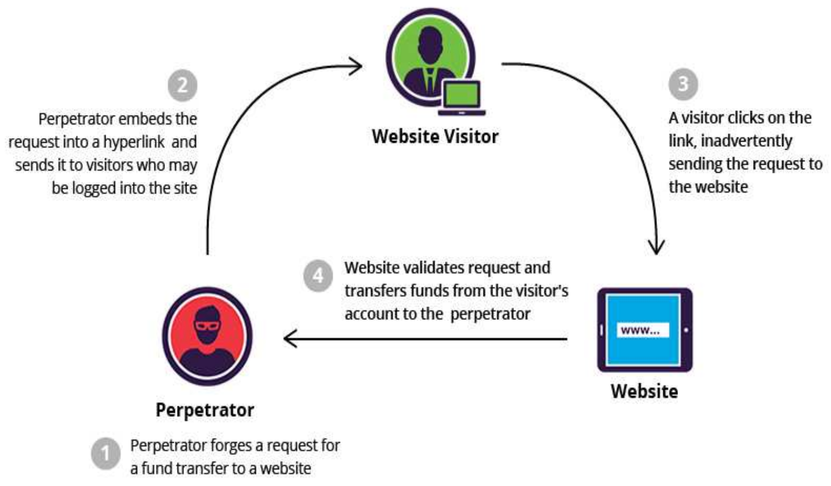
How to Prevent Cross-Site Scripting

- Keep Software Updated
- Sanitize Input Fields
- Use Client and Server-Side Form Validation
- Use a Web Application Firewall

CROSS-SITE REQUEST FORGERY

Cross-Site Request Forgery (CSRF) is an attack that causes an end-user to perform unauthorized actions on a web application that they are currently authenticated to.

CSRF attacks primarily target state-changing requests, not data theft, because the attacker has no means of knowing the answer to the request fabricated.

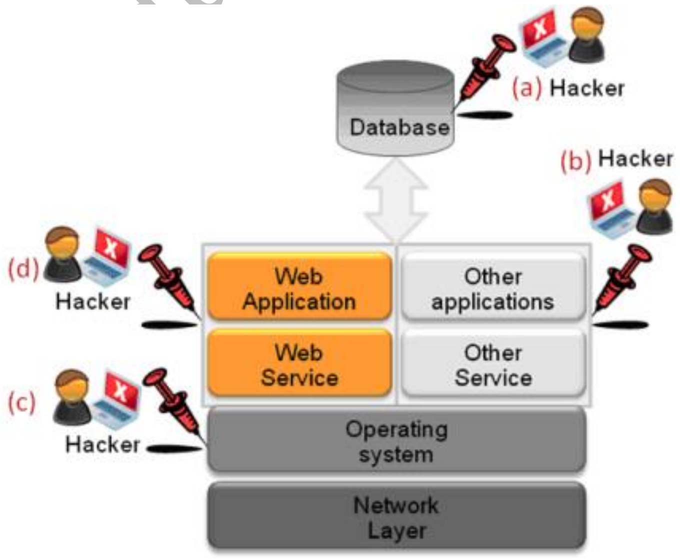

2
Perpetrator embeds the request into a hyperlink and sends it to visitors who may be logged into the site
Website Visitor
3
A visitor clicks on the link, inadvertently sending the request to the website
4
Website validates request and transfers funds from the visitor's account to the perpetrator
www...
Website
Perpetrator
1
Perpetrator forges a request for a fund transfer to a website

How to Prevent Cross-Site Request Forgery

- REST (Representation State Transfer)
- Anti-Forgery Tokens
- Ensure Cookies are sent with the SameSite Cookie Attribute
- Include Addition Authentication for Sensitive Actions

REMOTE CODE EXECUTION

Remote Code Execution, also known as *code injection,* is used in computer security to describe an attacker's ability to execute any commands of the attacker's choice on a target machine or in a target process.



How to Prevent Remote Code Execution

- Validate User Input
- Use Established Software
- Use the Principle of Least Privilege
- Configure server to reject URLs with ".. /" to prevent traversal of resource path.
- Lock down system commands and utilities with restricted ACL (Access Control Lists).
- Keep updated with patches and updates to ensure timely patching of newly found buffer overflows.

TEMPORARY FILE ABUSE

Many applications and utilities could never even run without temporary files, which typically provide accessible behind-the-scenes workspace. We list here just a few examples of the practical roles temporary files fulfill:

- Interim versions of files being manipulated by applications like word processors or graphics programs.

- Temporary database query caches, providing accessibility to previously selected data without requiring another database access. While not normally used for transactions involving a local database, they are a regular feature of applications that make queries to remote databases or XML-based web services.

- Temporary storage for files in the process of being transferred. These are the files named by PHP's superglobal $_FILES['userfile']['tmp_name'] variable.

- System files being used to store session properties (or other temporary data) between HTTP requests. For session properties, these are the files named for the session ID (typically something like sess_7483ae44d51fe21353afb671d13f7199).

- Interim storage for data being passed either to other applications or libraries that expect file-based input, or to later instances of the same application (like messages in a mail queue).

As this brief list suggests, temporary files are perfectly capable of containing some of the most private information on your computer.

So everything from bits and pieces to a complete file may be floating around on your server (or on a search engine server), available to anybody who has access, whether that access is legitimate or not.

One obvious risk is therefore that your private data could be exposed to the public or (very likely worse) to a prowler looking for it.

In most cases, an exploit would require that the attacker have shell or FTP access to the locations of your temporary files. But if such an attacker were to get in, a file named 2020_Confidential_Sales_Strategies.tmp would probably be of great interest to him, especially if he worked for your employer's biggest competitor. Similarly, a file named something like sess_95971078f4822605e7a18c612054f658 could be interesting to someone looking to hijack a session containing a user's login to a shopping site.

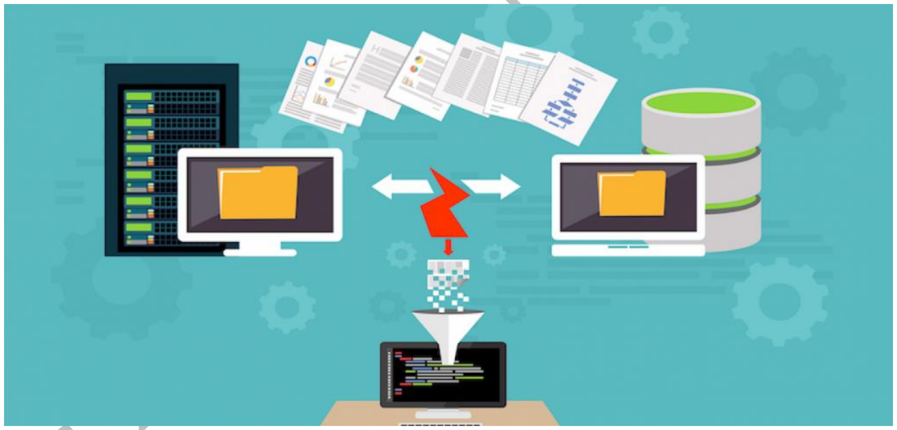
However, exposure of private data may be possible even without such access. If a prowler were to observe that a $_GET variable is being used to allow access to, for example, the output from a spellchecking program (with a URI something like http://bad.example.com/spellcheck.php?tmp_file=spellcheck46), it might be very illuminating for him to enter into his browser a URI like this: http://bad.example.com/spellcheck.php?tmp_file=spellcheck45. The chances seem very good that he would be able to read the file that was previously checked.

How to Prevent Temporary File Abuse

- Regularly check temporary files and remove those not needed anymore

SESSION HIJACKING

Session hijacking occurs when a session token is sent to a client browser from the Web server following the successful authentication of a client logon. A session hijacking attack works when it compromises the token by either confiscating or guessing what an authentic token session will be, thus acquiring unauthorized access to the Web server.



How to Prevent Session Hijacking

- Install an SSL Certificate
- Install a Security Plugin
- Update your Website

PASSWORD CRACKING

Password cracking refers to various measures used to discover computer passwords. This is usually accomplished by recovering passwords from data stored in, or transported from, a computer system. Password cracking is done by either repeatedly guessing the password, usually through a computer algorithm in which the computer tries numerous combinations until the password is successfully discovered.

Password cracking can be done for several reasons, but the most malicious reason is in order to gain unauthorized access to a computer without the computer owner's awareness. This results in cybercrime such as stealing passwords for the purpose of accessing banking information.

Other, nonmalicious, reasons for password cracking occur when someone has misplaced or forgotten a password. Another example of nonmalicious password cracking may take place if a system administrator is conducting tests on password strength as a form of security so that hackers cannot easily access protected systems.

How to Prevent Password Cracking

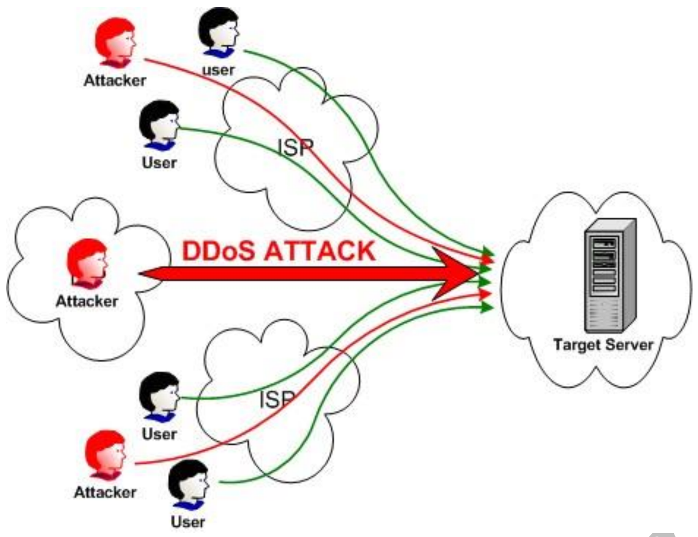
- Create password policies
- Use strong passwords for all account types.
- Apply lockout policies to end-user accounts to limit the number of retry attempts that can be used in guessing user passwords.
- Do not use default account names, and rename standard accounts such as the administrator's account and the anonymous Internet user account used by many Web applications.
- Track and review failed logins for patterns of password hacking attempts.

DENIAL-OF-SERVICE ATTACK

A ***Denial-of-Service (DoS) Attack*** is an attempt to render a machine or network resource unavailable to its intended users, such as interrupting or suspending services temporarily or indefinitely from a host connected to the Internet.

Denial of service is usually achieved by overwhelming the targeted computer or resource with superfluous requests in an effort to overwhelm systems and prevent the fulfillment of any or all valid requests.

A ***Distributed Denial-of - Service (DDoS)*** is where more than one, often thousands, of unique IP addresses are the source of the attack. It is similar to a group of people crowding the doorway or entrance to a shop or company, and not allowing legitimate parties to access the shop or company, disrupting normal business.



How to Prevent Denial-of-Service Attack

- Develop a Denial-of-Service Response Plan
- Secure Network Infrastructure
- Practice Basic Network Security
- Maintain Strong Network Architecture
- Leverage the Cloud
- Understand the Warning Signs
- Consider DDoS-as-a-Service
- CAPTCHAs

SPYWARE, VIRUS, TROJANS AND WORMS

Spyware describes software with malicious behavior that aims to gather information about a person or organization and send such information to another entity in a way that harms the user; for example, by violating their privacy or endangering their device's security. This behavior may be present in malware as well as in legitimate software. Websites may also engage in spyware behaviors like web tracking.

A ***computer virus*** is a type of malicious code or program written to alter the way a computer operates and is designed to spread from one computer to another. A virus operates by inserting or attaching itself to a legitimate program or document that supports macros in order to execute its code. In the process, a virus has the potential to cause unexpected or damaging effects, such as harming the system software by corrupting or destroying data.

A ***trojan horse*** is a program appearing to be something safe, but is performing malicious tasks, such as giving access to your computer or sending personal information to other computers. Trojan horses are one of the most common methods a computer criminal uses to infect your computer and collect personal information from your computer.

A ***computer worm*** is a type of malware that spreads copies of itself from computer to computer. It can replicate itself without any human interaction and does not need to attach itself to a software program in order to cause damage. Worms can be transmitted via software vulnerabilities.

How to Prevent Spyware, Virus, Trojans and Worms

- Stay updated with the latest operating system service packs and software patches.
- Block all unnecessary ports at the firewall and host.
- Disable unused features including protocols and services;
- Strengthen weak or default configuration settings.

SPOOFING

Spoofing, in general, is a fraudulent or malicious practice in which communication is sent from an unknown source disguised as a source known to the receiver. Spoofing is most prevalent in communication mechanisms that lack a high level of security.

Email spoofing is one of the best-known spoofs. Since core SMTP fails to offer authentication, it is simple to forge and impersonate emails. Spoofed emails may request personal information and may appear to be from a known sender. Such emails request the recipient to reply with an account number for verification. The email spoofer then uses this account number for identity theft purposes, such as accessing the victim's bank account, changing contact details and so on.

The attacker (or spoofer) knows that if the recipient receives a spoofed email that appears to be from a known source, it is likely to be opened and acted upon. So, a spoofed email may also contain additional threats like Trojans or other viruses. These programs can cause significant computer damage by triggering unexpected activities, remote access, deletion of files and more.

How to Prevent Spoofing

- Using strong authentication.
- Do not store secrets in plaintext (for example, passwords).
- Do not transfer the credentials over the wire in plaintext.
- Keep authentication cookies safe with Secure Sockets Layer (SSL).

TAMPERING

Tampering is the unauthorized alteration of data, for instance as it flows between two computers over a network.

How to Prevent Tampering

- Using data *hashing* and signing.
- Using *digital signatures*.
- Using strong authorization.
- Using tamper-resistant protocols across communication links.

- Secure communication links with protocols that provide message confidentiality and integrity.

FULL PATH DISCLOSURE

Full Path Disclosure (FPD) is the reveal of a vulnerable script's full operating path. The FPD bug is executed by injecting unexpected characters into some webpage parameters. The script does not expect the injected character and returns an error message which includes error information, as well as the targeted script's operating path.

How to Prevent Full Path Disclosure

- Turn off error reporting within your server
- Use regular expressions to disable error messages

LOCAL AND REMOTE FILE INCLUSION

Arbitrary File Inclusion allows an attacker to include a file, normally via a web server script. The vulnerability occurs without proper testing due to the use of user-supplied data without proper validation. This can result in anything as trivial as outputting the file contents, or more severe events like: (1) execution of codes in the web server, (2) execution of codes in the client-side which can lead to other attacks such as cross-site scripting, (3) denial-of-service, and (4) data theft/manipulation.

Local File Inclusion (LFI) is the method of including files, that are already locally present on the server, through the exploiting or attacking of vulnerable inclusion procedures implemented within the web system.

Remote File Inclusion (RFI) is the most common form of vulnerability found on websites. This helps an intruder to access a remote file, normally through a Web server script. The weakness occurs when there is no proper testing of user-supplied data.

How to Prevent Local and Remote File Inclusion

- Never use arbitrary input data in a literal file include request
- Use a filter to thoroughly scrub input parameters against possible file inclusions
- Build a dynamic whitelist

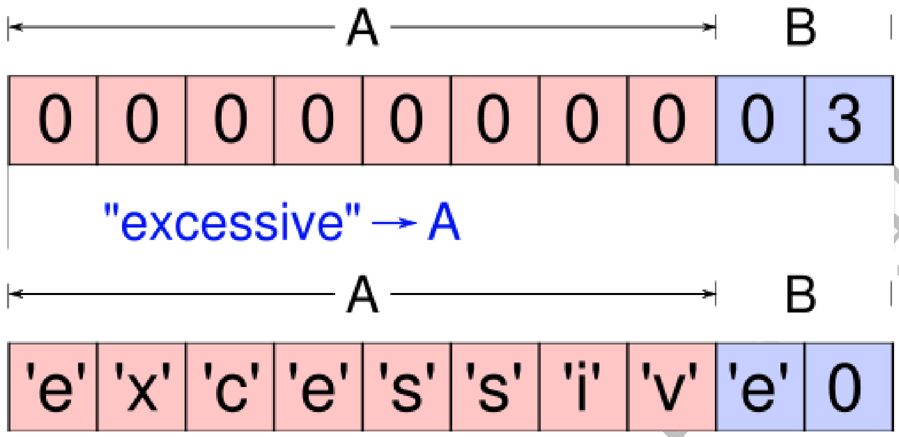
MEMORY CORRUPTION AND BUFFER OVERFLOW

Within a computer system, ***memory corruption*** happens when the contents of a memory location are inadvertently changed due to programming errors; this is called violating memory safety.

Later in that program, when the corrupted memory content is used, it leads either to system crash or to odd and weird program behavior.

When a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold, a ***buffer overflow*** occurs.

Because buffers are created to contain a finite amount of data, the extra information-which has to go somewhere-can overflow into adjacent buffers, corrupting or overwriting the valid data contained therein.



How to Prevent Buffer Overflow and Memory Corruption

- Input Validation
- Handle Strings Safely
- Maintaining Code Quality

DATA BREACH

A ***Data Breach*** is the deliberate or accidental release into an untrusted environment of sensitive information. Other words for this phenomenon include *Unintentional Information Disclosure, Data Leak and even Data Spill.*

A ***Data Breach*** is a security event where important, safe or confidential information is copied, distributed, accessed, stolen or used by an unauthorized person to do so.

Data breaches may include financial information such as *credit card or bank numbers, personal health information (PHI), Personally Identifiable Information (PII), corporate trade secrets or intellectual property.*

How to Prevent Data Breach

- Strong password policy
- Two Factor Authentication (2FA) and Multifactor Authentication
- Physical Security Practices
- Monitoring User Activity
- Endpoint Security

REFERENCES

What are Web Application Vulnerabilities? (https://www.rapid7.com/fundamentals/web-application-vulnerabilities)

What is a Website Vulnerability and How Can it be Exploited?
(https://www.sitelock.com/blog/what-is-a-website-vulnerability)

Types and Countermeasures Against Phishing Attacks
(https://www.mailxaminer.com/blog/countermeasures-against-phishing-attacks)

How to Prevent Phishing
(https://www.wikihow.com/Prevent-Phishing)

Hacksplaining – Web Vulnerabilities
(https://www.hacksplaining.com)

How to Prevent Cross-Site Scripting Attacks
(https://www.sitelock.com/blog/prevent-cross-site-scripting-attacks)

How to Protect Your Website from Remote Code Execution
(https://www.liquidweb.com/kb/how-to-protect-your-website-from-remote-code-execution)

Tactics to prevent DDos Attacks and Keep Your Website Safe
(https://phoenixnap.com/blog/prevent-ddos-attacks)

UNIT III. PRACTICING SECURE OPERATIONS

Learning Objectives

At the end of the unit, the student will be able to:

1. Describe different procedures to secure web operations.

SECURITY FOR COOKIES

Session cookies are often seen as one of the biggest security and privacy issues with HTTP, but often they need to be used to maintain state of the art in modern web applications. By default, it is unsafe and vulnerable to being intercepted by an authorized party. Cookies typically store session identifiers that may provide full access to an account, so if a cookie is intercepted, a session may be hijacked by someone who is not the actual user, but who pretends to be the user.

For this reason, it is very important to set the necessary settings to make cookies safer, and this can be achieved by paying attention to two things below:

1. ***HttpOnly Flag***

   The first flag that we need to set up is the ***HttpOnly flag***. By default, when there is no restriction in place, cookies can be transferred not only via HTTP, but any JavaScript files loaded on a page can also access the cookies. This capability can be dangerous because it makes the page vulnerable to cross-site scripting (XSS) attacks.

   The only way to restrict this is by setting the ***HttpOnly flag***, which means that the only way to send cookies is via an HTTP connection, not directly through other means (i.e., JavaScript).

2. ***Secure Flag***

   The second flag that we need to pay attention to is the ***Secure flag***. This flag highlights the second issue that default cookies are always sent to both HTTP and HTTPS requests. A malicious attacker who cannot see encrypted traffic with an HTTPS connection can easily switch to an HTTP connection and access the same cookie because it is not encrypted.

   We therefore need to set the Secure flag to ensure that the cookie is encrypted when it is created.

These settings should be managed within the application code, by right. However, due to a lack of awareness among developers, server administrators may have to force the settings on the respective web servers by following one the procedure provided below.

Implementation Procedure in Apache Web Server:

- Ensure you have **mod_headers.so** enabled in Apache HTTP server
- Add following entry in **httpd.conf**
  Header edit Set-Cookie ^(.)$ $1;HttpOnly;Secure;SameSite=None*
- Restart Apache HTTP server to test

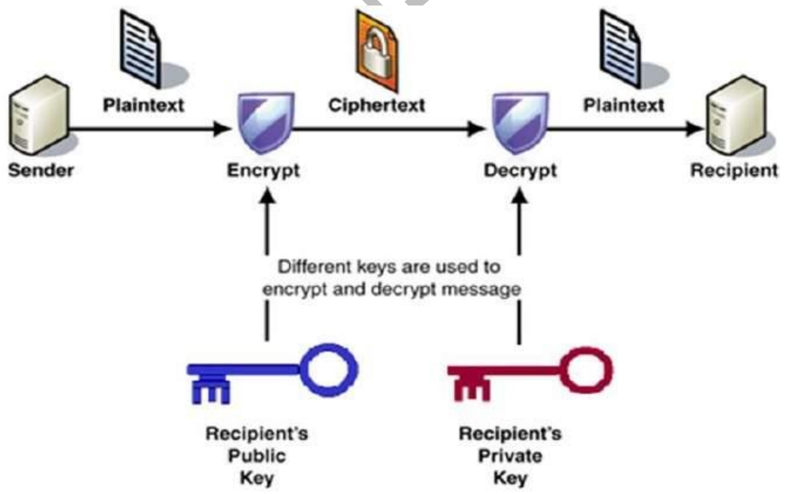
Note: Header edit is not compatible with servers lower than 2.2.4 version.

- You can use the following to set the HttpOnly and Secure flag in lower than 2.2.4 version.
  Header set Set-Cookie HttpOnly;Secure;SameSite=None

PUBLIC KEY ENCRYPTION

Public key encryption, or public key cryptography, is a method of encrypting data with two different keys and making one of the keys, the public key, available for use by anyone. The other key is known to be the private key. Data encrypted with the public key can only be decrypted using the private key, and data encrypted with the private key can only be decrypted using the public key.

Public key encryption is also referred to as asymmetric encryption. It is widely used, particularly for TLS/SSL, which makes HTTPS possible.



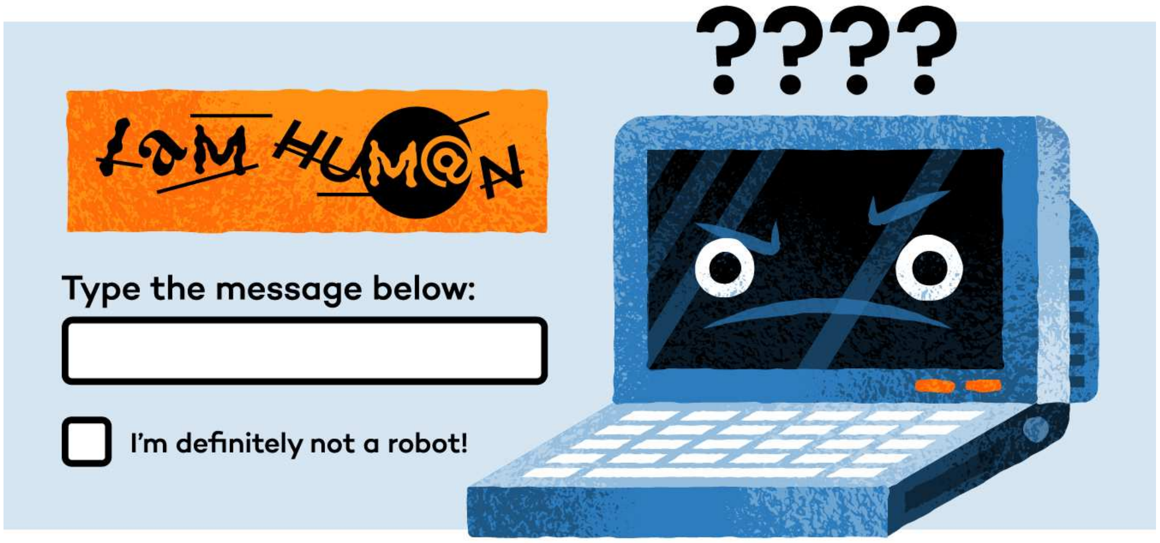
The most important features of the public key encryption scheme are:

- For encryption and decryption, different keys are used. This is a property that sets this scheme apart from the symmetric encryption scheme.
- Each receiver has a unique decryption key, generally referred to as its private key.
- Receiver needs to publish the encryption key referred to as its public key.

- Some assurance of the authenticity of the public key is needed in this scheme in order to avoid being spoofed by the adversary as the receiver. Generally, this type of cryptosystem involves a trusted third party that certifies that a particular public key belongs only to a specific person or entity.
- Encryption algorithm is complex enough to prevent an attacker from deducting plain text from the cipher text and the (public) encryption key.
- Although private and public keys are mathematically linked, it is not possible to calculate the private key from the public key. In fact, the smart part of any public-key cryptosystem is in the design of a relationship between two keys.

CAPTCHAs

CAPTCHA stands for *Completely Automated Public Turing Test to tell Computers and Humans Apart*. In other words, CAPTCHA determines whether the user is a real robot or a spam robot. CAPTCHAs stretch or manipulate letters and numbers, and rely on human capacity to determine which symbols they are.



CAPTCHAs were invented to block spamming software from posting comments on pages or buying excess items at once. The most common form of CAPTCHA is an image with a number of distorted letters. It is also common to choose from a variety of images where you need to choose a common theme.

The internet and computers are made up of a unique coding language. Computers find it difficult to understand languages because of the strange and intricate rules human languages takes on and the slang that humans use.

CAPTCHAs are used by varieties of websites to verify that the user is not a robot. Gmail and Yahoo used CAPTCHAs in their registration forms to prevent spammers from using bots to create a plethora of spam email accounts.

CAPTCHA has a variety of applications for keeping websites and users secure. These include but are not limited to:

- Protecting email addresses from scammers
- Protect website registrations
- Protects online polling
- Protects against email worms/junk mail
- Prevents dictionary attacks
- Prevents content spamming on blogs

If a website needs proper protection from scammers, it's recommended to use a CAPTCHA. There are a few extra measures to take when using CAPTCHA code:

- ***Secure Images***: images should be distorted randomly when presented to the user. With minor distortions, the image is vulnerable to automated attacks.
- ***Unique CAPTCHAs***: If every site used similar CAPTCHA codes, hackers could catch on and create bots that would bypass this test. That's why it is important to change the type of CAPTCHAs every so often and avoid common mathematical equations such as 1+1.
- ***Script Security***: In addition to making sure your images are unreadable by computers, you should ensure that there are no easy ways around the script level.
- ***Accessibility***: CAPTCHAs need to be accessible for every user. In this regard, CAPTCHAs cannot be based solely on reading text or choosing images. It's important that users have the opportunity to opt for an audio CAPTCHA if needed.

USER AUTHENTICATION, AUTHORIZATION AND LOGGING

Authentication is the process of identifying users who request access to a system, network or device. Access control often determines the identity of the user according to username and password. Other authentication technologies, such as biometrics and authentication applications, are also used to authenticate user identity.

User authentication is a method that prevents unauthorized users from accessing sensitive information. For example, User A only has access to the relevant information and is unable to view the sensitive information of User B.

Cyber criminals can gain access to the system and steal information if user authentication is not secure.

Cyber criminals are always improving their attacks. As a result, security teams are faced with many authentication-related challenges. That is why companies are starting to implement more sophisticated incident response strategies, including authentication as part of the process. The list below examines some common authentication methods used to secure modern systems.

- ***Password-based Authentication***

  Passwords are the most common authentication methods. Passwords may be in the form of a string of letters, numbers, or special characters. You need to create strong passwords that include a combination of all possible options to protect yourself.

However, passwords are prone to phishing attacks and poor hygiene, which weakens their effectiveness. An average person has about 25 different online accounts, but only 54% of users use different passwords across their accounts.

- ***Multi-factor Authentication***

Multi-Factor Authentication (MFA) is an authentication method that requires two or more independent means of identifying the user. Examples include smartphone-generated codes, CAPTCHA tests, fingerprints, or facial recognition.

MFA authentication methods and technologies increase user confidence by adding multiple layers of security. MFA may be a good defense against most account hacks, but it has its own pitfalls. People may lose their phones or SIM cards and may not be able to generate an authentication code.

- ***Certificate-based Authentication***

Certificate-based authentication technologies identify users, machines or devices using digital certificates. An electronic document based on the idea of a driver's license or passport is a digital certificate.

The certificate contains the digital identity of the user, including the public key, and the digital signature of the Certification Authority. Digital certificates prove the ownership of the public key and are issued only by the certification authority.

Users provide their digital certificates when they log in to a server. The server shall verify the credibility of the digital signature and the certificate authority. The server uses cryptography to confirm that the user has the correct private key associated with the certificate.

- ***Biometric Authentication***

Biometric authentication is a security process that relies on an individual's unique biological characteristics. Key advantages of using biometric authentication technologies are as follows:

- The biological characteristics can be easily compared to the authorized features stored in the database.
- Biometric authentication can control physical access when installed on doors and doors.
- You can add biometrics to your multi-factor authentication process.

Biometric authentication technologies are used by consumers, governments and private companies, including airports, military bases and national borders. Common biometric authentication methods include: (a) facial recognition, (b) fingerprint scanners, (c) voice identification and (d) eye scanners.

- ***Token-based Authentication***

  Token-based authentication technologies allow users to enter their credentials once and receive a unique encrypted string of random characters in exchange. You can then use the token to access protected systems instead of re-entering your credentials. The digital token shows that you already have access permission.

Authorization is a process by which the server determines whether the client has permission to use a resource or to access a file.

Authorization is usually coupled with authentication so that the server has some idea of who the client is asking for access.

Logging is the process of recording the date and time when an authenticated user enters the system and lists all the processes that the user has done or interacted with the system. Logging is helpful in identifying the actors for every event that happened within the application.

Created log files are commonly located in text files that are placed away from the database folder to ensure it will be available in cases where the database has been breached and access to its directory has been blocked by the hacker.

USING SECURITY CERTIFICATES

The ***Web Security Certificate*** is a validation and encryption tool, part of the HTTPS protocol, which secures and encrypts data back and forth between the server and the client browser. It is issued by a trusted certification authority (CA) which verifies the identity of the website owner. The certificate then assures the user that the website to which it is linked is legitimate and that the connection is secure and secure.

Website security certificates act as a means of ensuring the identity of individual websites and of giving responsibility to their owners for the privacy and security of all their users and visitors.

Websites that use security certificates are identifiable by means of the HTTPS address starter and the padlock icon that appears in the address bar, with a different location depending on the browser being used. When the padlock icon is clicked, it shows the details of the certificate, such as the identity of the website owner, the issuing CA, and the encryption and connection mechanisms used.

Security certificates are therefore part of HTTPS for two purposes: to validate that the user is actually connected to the site they think they are connected to, and to establish a means of encrypting communications between the user and the website. The current standard for website security certificates is SSL certificates.

PREVENTING DATA LOSS

Data Loss Prevention (DLP) is the practice of detecting and preventing data breaches, exfiltration or unintended destruction of sensitive data. Organizations are using DLP to protect and secure their data and comply with the regulations.

The term DLP refers to the organization's defense against both data loss and data leak prevention. Data loss refers to an event in which important data is lost to an enterprise, such as a ransomware attack. Data loss prevention focuses on preventing the illicit transfer of data beyond organizational boundaries.

DLP is typically used by organizations to:

- Protect personally identifiable information (PII) and comply with applicable regulations.
- Protect Intellectual Property that is critical to the organization
- Achieve Visibility of data in large organizations
- Secure mobile workforce and enforce security in the Bring Your Own Device (BYOD) environment
- Secure data for remote cloud systems

Causes of Data Leaks

Three common causes of data leakage are:

- ***Insider threats*** **-** a malicious insider or an attacker who compromised a privileged user account, misused their permissions and attempted to move data outside the organization.

- ***Extrusion by Attackers*** **-** many cyberattacks have sensitive data as their target. Attackers enter the security perimeter using techniques such as phishing, malware or code injection, and gain access to sensitive data.

- ***Unintentional or negligent data exposure*** **-** many data leaks occur as a result of employees losing sensitive data to the public, providing open access to data on the Internet, or failing to restrict access through organizational policies.

Data Leakage Prevention

You can use standard security tools to protect against data loss and leakage. For example, the Intrusion Detection System (IDS) can alert attackers to attempts to access sensitive data. Antivirus software may prevent attackers from compromising sensitive systems. Firewalls may block access from any unauthorized party to systems that store sensitive data.