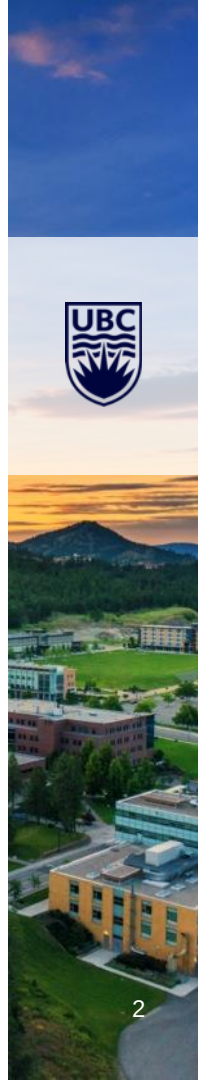
Security

I respectfully acknowledge that UBC Okanagan is situated on the traditional, unceded and ancestral territory of the Syilx Okanagan Nation

Learning Objectives

At the end of this class, students will be able to:

- Describe a wide range of security principles and practices
- Describe best practices for authentication systems and data storage
- Describe and apply key cryptography, SSL, and certificates
- Describe how to proactively protect their sites against common attacks



1. Security Principles
2. Authentication
3. Cryptography
4. Hypertext Transfer Protocol Secure (HTTP)
5. Security Best Practices
6. Common Threat Vectors

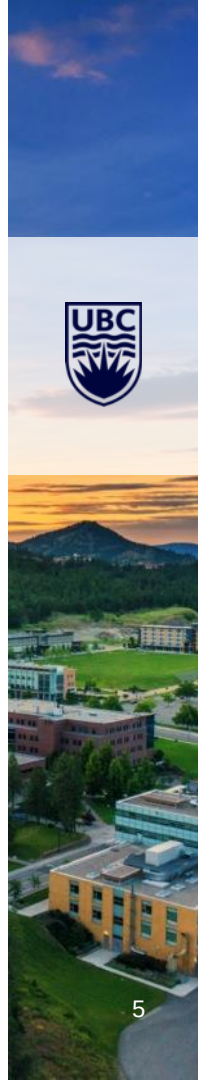
1. Security Principles

Security Overview

The right way of addressing security is right from the beginning and all along the way so that you can plan for a secure system and hopefully have one in the end.

Not only is a malicious hacker on a tropical island a threat but so too is a sloppy programmer, a disgruntled manager, or a naive secretary.

Since websites are an application of networks and computer systems, you must draw from those disciplines to learn many foundational security ideas.

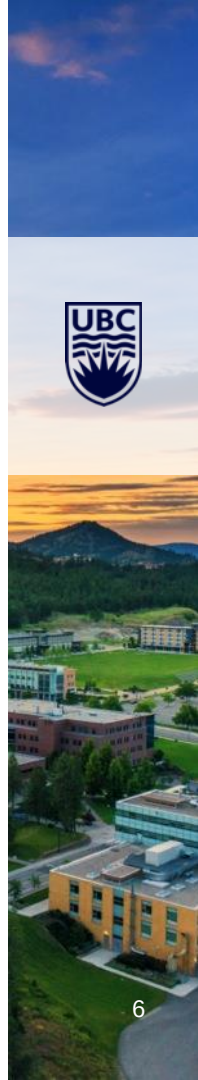


Information Security

The CIA Triad

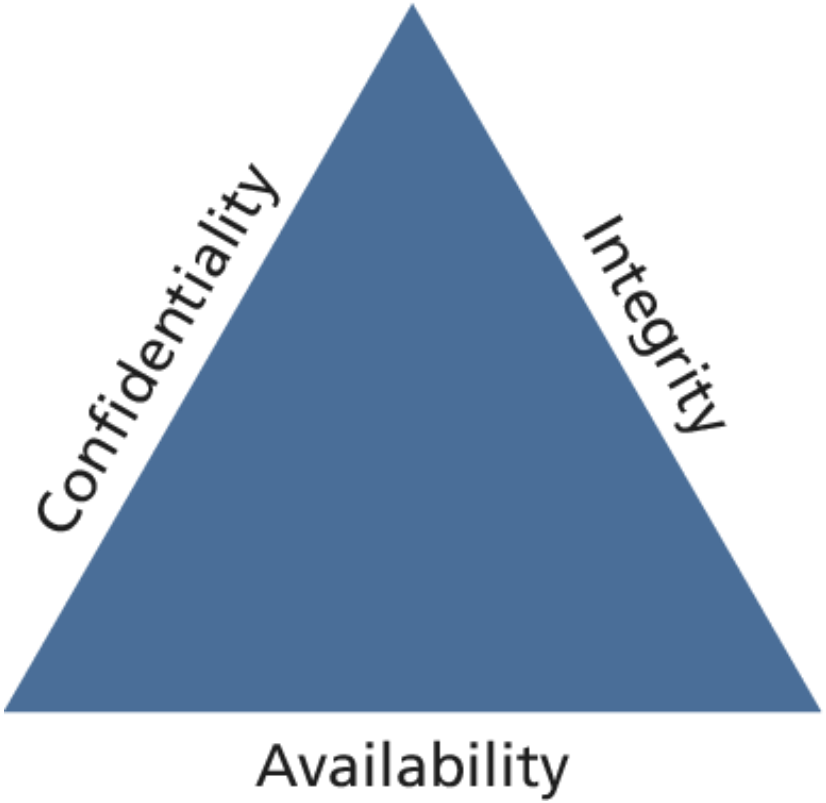
Information security is the holistic practice of protecting information from unauthorized users. Computer/IT security is just one aspect of this holistic thinking, which addresses the role computers and networks play.

The other is **information assurance**, which ensures that data is not lost when issues do arise.



Information Security

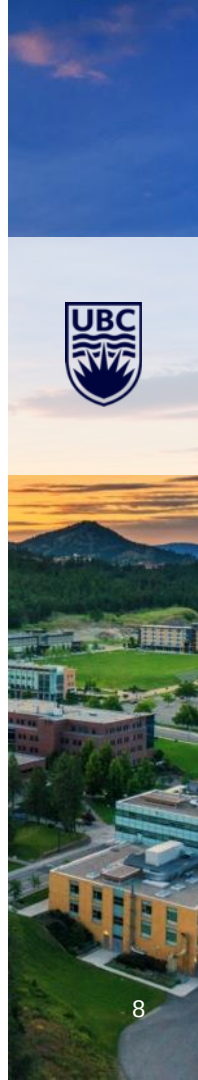
The CIA Triad



Information Security

The CIA Triad

- **Confidentiality** is the principle of maintaining privacy for the data you are storing, transmitting, etc.
  - This is the concept most often thought of when security is brought up.
- **Integrity** is the principle of ensuring that data is accurate and correct.
  - This can include preventing unauthorized access and modification, but also extends to disaster preparedness and recovery.
- **Availability** is the principle of making information available when needed to authorized people.
  - It is essential to making the other two elements relevant, since without it, it's easy to have a confidential and integral system (a locked box).



Risk Assessment and Management

Risk assessment uses the concepts of

- *actors*,
- *impacts*,
- *threats*, and
- *vulnerabilities*

to determine where to invest in defensive countermeasures.

Actors

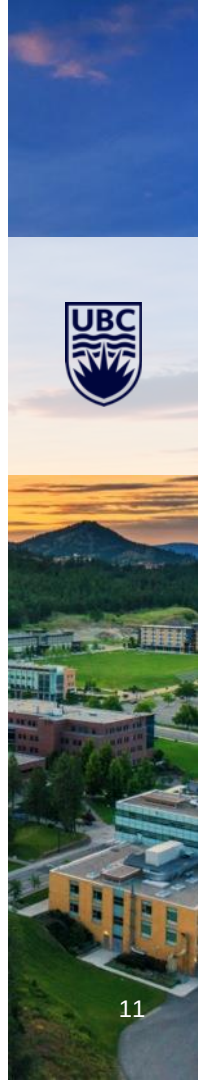
Risk Assessment

The term “actors” refers to the people who are attempting to access your system. They can be categorized as:

- Internal actors
- External actors
- Partner actors

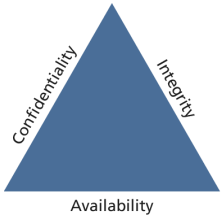
Actors

- **Internal actors** are the people who work for the organization. Although they account for a small percentage of the attacks, they are especially dangerous due to their internal knowledge of the systems.
- **External actors** are the people outside of the organization. It turns out that more than three quarters of external actors are affiliated with organized crime or nation states. Most common source of attacks.
- **Partner actors** are affiliated with an organization that you partner or work with. Quite often partners are granted some access to each other's systems (to place orders, for example). If your partner is somehow compromised, there is a chance your data is at risk as well



Impact

What systems were infiltrated and what data was stolen or lost?



- A ***loss of availability*** prevents users from accessing some or all of the systems.

  This might manifest as a denial of service attack, or a SQL injection attack (described later), where the payload removes the entire user database, preventing logins from registered users.

- A ***loss of confidentiality*** includes the disclosure of confidential information to a (often malicious) third party.

  This could manifest as a cross-site script attack where data is stolen right off your screen or a full-fledged database theft where credit cards and passwords are taken.

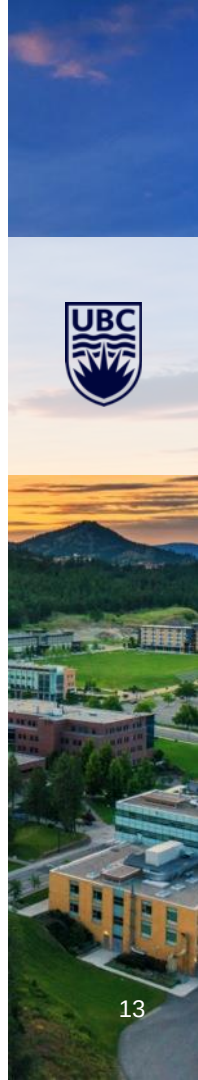
- A ***loss of integrity*** changes your data or prevents you from having correct data.

  This might manifest as an attacker hijacking a user session, perhaps placing fake orders or changing a user's home address.

Threats

A **threat** refers to a particular path that a hacker could use to exploit a vulnerability and gain unauthorized access to your system.

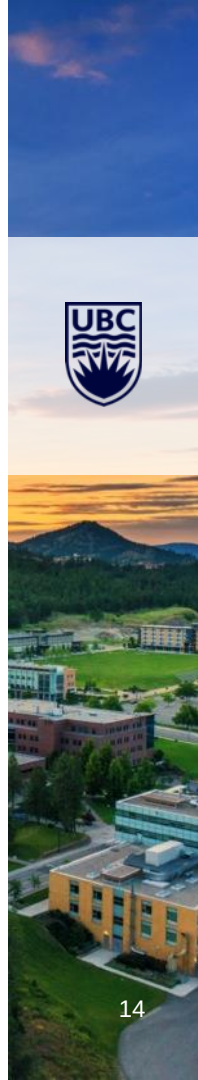
A flood destroying your data center is a threat just as much as malicious SQL injections, buffer overflows, denial of service, and cross-site scripting attacks.



Threats

Categorize threats with **Stride**

- **S**poofing – The attacker uses someone else's information to access the system.
- **T**ampering – The attacker modifies some data in nonauthorized ways.
- **R**epudiation – The attacker removes all trace of their attack, so that they cannot be held accountable for other damages done.
- **I**nformation disclosure – The attacker accesses data they should not be able to.
- **D**enial of service – The attacker prevents real users from accessing the systems.
- **E**levation of privilege – The attacker increases their privileges on the system thereby getting access to things they are not authorized to do.



Vulnerabilities

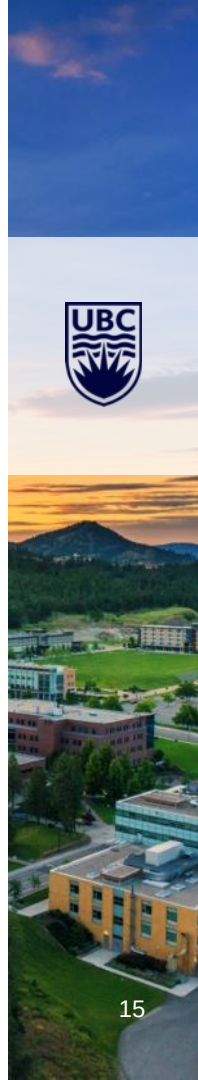
The holes in your armor

Once vulnerabilities are identified, they can be assessed for risk.

Some vulnerabilities are not fixed because they are unlikely to be exploited, while others are low risk because the consequences of an exploit are not critical.

The top five classes of web vulnerability are:

1. Injection
2. Broken authentication and session management
3. Cross-site scripting
4. Insecure direct object references
5. Security misconfiguration

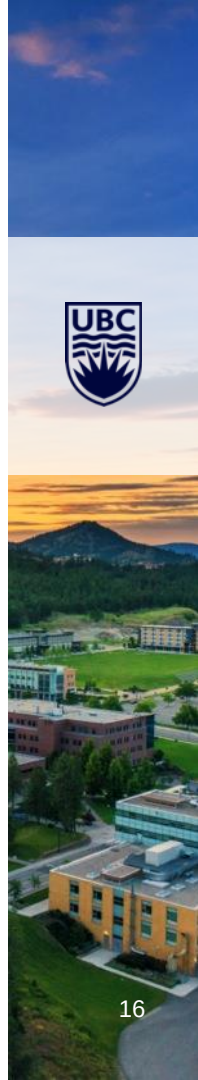


Assessing Risk

Putting it all together

For our purposes, it will suffice to summarize that in risk assessment you would begin by identifying the actors, vulnerabilities, and threats to your information systems.

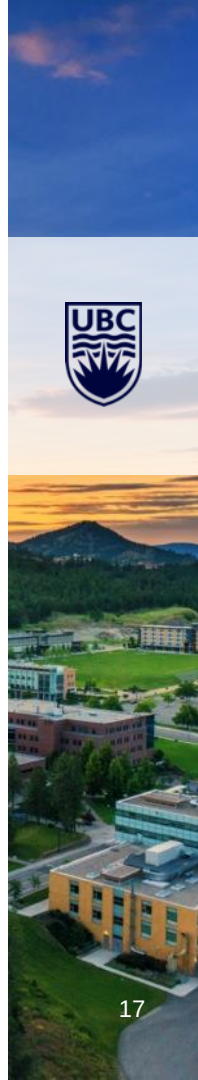
The probability of an attack, the skill of the actor, and the impact of a successful penetration are all factors in determining where to focus your security efforts.



Policies

One part of your defenses

- **Usage policy** defines what systems users are permitted to use, and under what situations. A company may, for example, prohibit social networking while at work. Usage policies are often designed to reduce risk by removing some attack vector.
- **Authentication policy** controls how users are granted access to the systems. Ex. A badge access is needed, biometric ID or password including password policies.
- **Legal policies** define a wide range of things including data retention and backup policies as well as accessibility requirements (like having all public communication well organized for the blind).



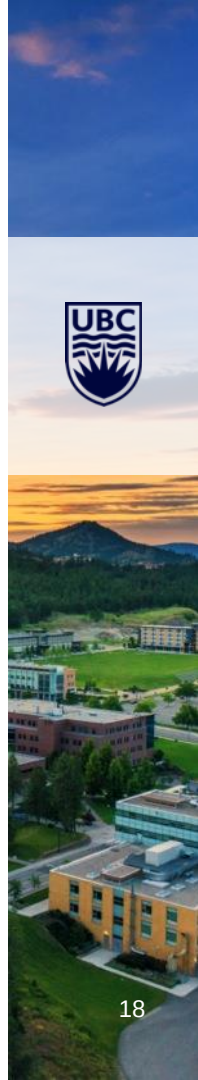
Policies

Password policies can stipulate

- the characteristics of acceptable passwords, and
- expiration of passwords.

Ironically, draconian password policies introduce new attack vectors, nullifying the purpose of the policy at the first place.

Where authentication is critical, *two-factor authentication* should be applied in place of password policies that do not increase security.



Security Principles

Social Engineering

Social engineering refers to the techniques used to manipulate people into doing something, normally by appealing to their baser instincts.

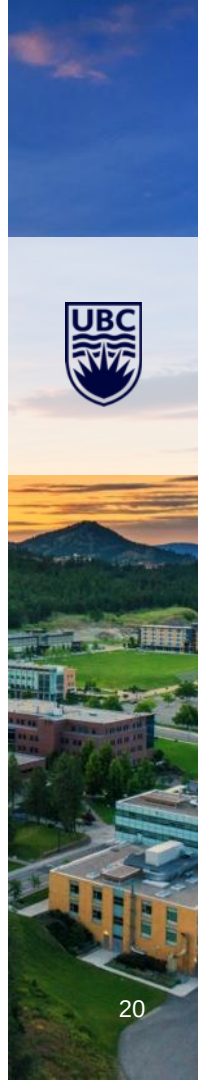
- Phishing Scams. Ex. email with fake login link
- Security Theater

Security Theater

Security theater is when visible security measures are put in place without too much concern as to how effective they are at improving actual security.

This is often done in 404 pages where a stern warning might read:

> *Your IP address is XX.XX.XX.XX. This unauthorized access attempt has been logged. Any illegal activity will be reported to the authorities.*



2. Authentication

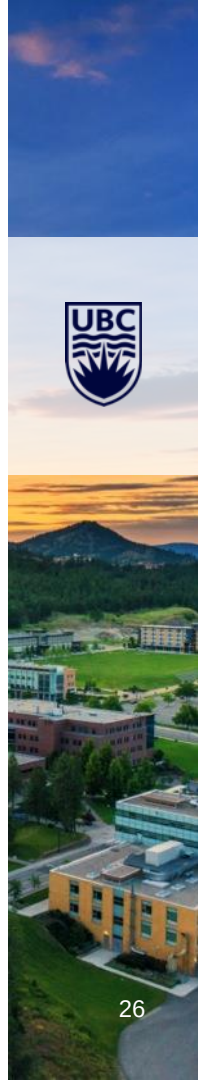
Authentication

Authentication is the process by which you decide that someone is who they say they are and therefore permitted to access the requested resources.

Whether

- getting entrance to an airport,
- getting past the bouncer at the bar, or
- logging into your web application,

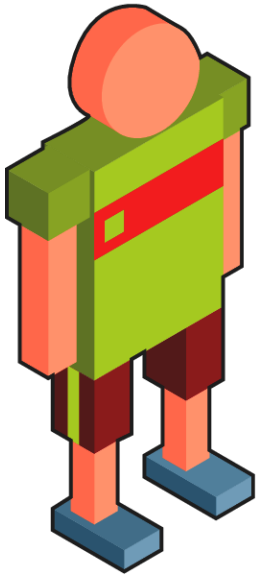
you have already seen authentication in action.



Authentication Factors

The things you can ask someone for

Authentication factors (token) are the things you can ask someone for in an effort to validate that they are who they claim to be.



What you know (Knowledge)

Passwords, PIN, security questions, ...

What you have (Ownership)

Access card, cell phone, cryptographic FOB, ...

What you are (Inherence)

Retinas, fingerprints, DNA, walking gait, ...

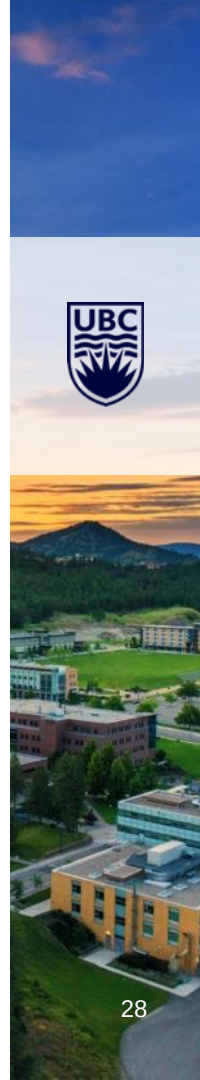
Single Factor Authentication

How many factors do you need?

Single-factor authentication is the weakest and most common category of authentication system where you ask for only one of the three factors.

- Know a password
- Posses an access card
- Fingerprint access on your mobile phone

When better authentication confidence is required, more than one authentication factor should be considered



Multifactor Authentication

More than one.

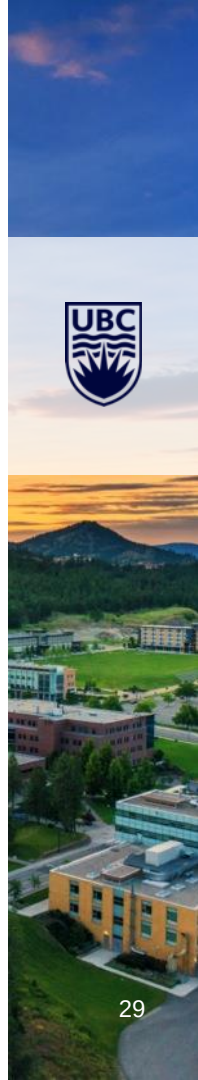
Multifactor authentication is where two distinct factors of authentication must pass before you are granted access.

The way we all access an ATM machine is an example of two-factor authentication:

- you must have both *the knowledge factor* (PIN) and
- the *ownership factor*(card)

Multifactor authentication is becoming prevalent in consumer products as well:

- your cell phone is used as the *ownership factor* alongside
- your password as a *knowledge factor*.

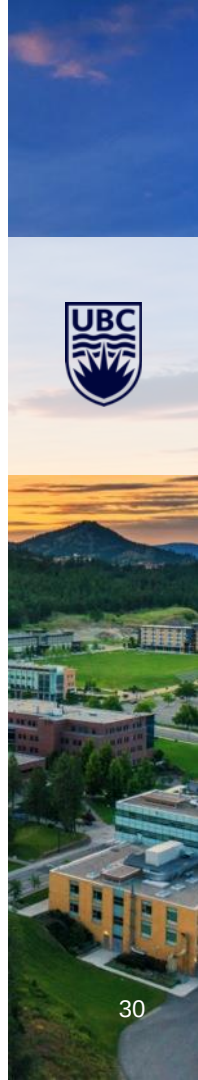


Third Party Authentication

Let someone else worry about it

Many popular services allow you to use their system to authenticate the user and provide you with enough data to manage your application.

Third-party authentication schemes like OpenID and OAuth are popular with developers and are used under the hood by many major websites including Amazon, Facebook, Microsoft, and Twitter.



Authorization

Not the same as authentication

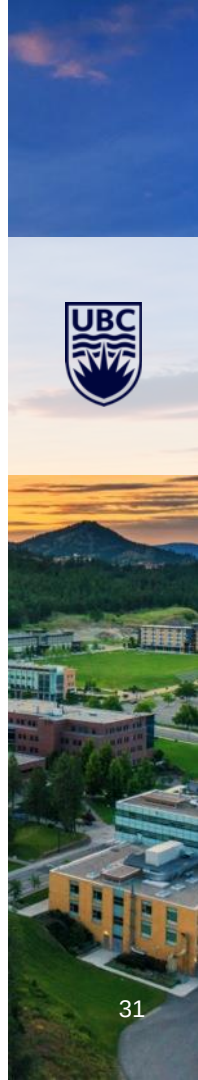
Authorization defines what rights and privileges a user has once they are authenticated.

- Authentication identifies that someone is who they say they are and ***grants*** them access

vs

- Authorization ***defines*** what the user with access can (and cannot) do.

The **principle of least privilege** is a helpful rule of thumb that tells you to give users and software only the privileges required to accomplish their work.

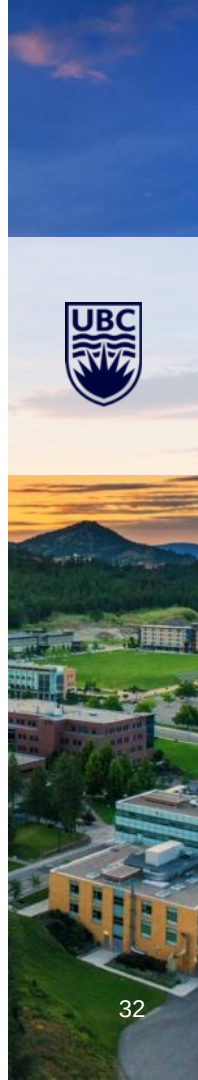


Authorization

Not the same as authentication

Some examples in web development where proper authorization increases security include:

- Using a separate database user for read and write privileges on a database
- Providing each user an account where they can access their own files securely
- Setting permissions correctly so as to not expose files to unauthorized users
- Ensuring Apache is not running as the root account (i.e. the account that can access everything)



3. Cryptography

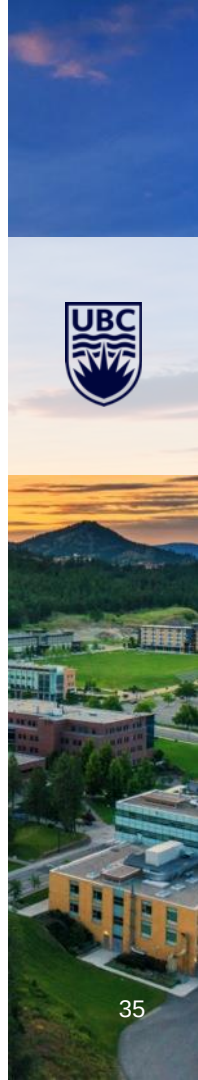
Cryptography

Secret Messages

Being able to send a secure message has been an important tool in warfare and affairs of state for centuries.

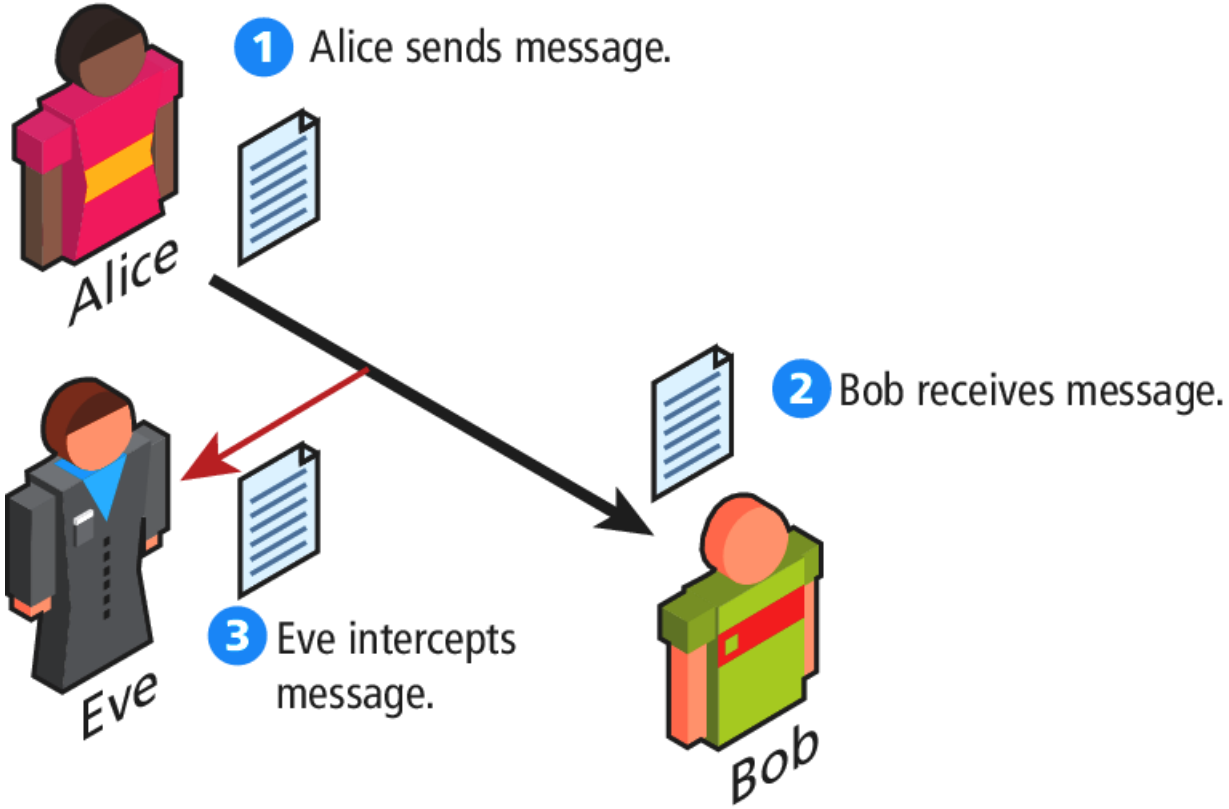
At a basic level we are trying to get a message from one actor (we will call her **Alice**), to another (**Bob**), without an eavesdropper (**Eve**) intercepting the message.

Since a single packet of data is routed through any number of intermediate locations on its way to the destination, getting your data (and passwords) is as simple as reading the data during one of the hops unless you use cryptography.



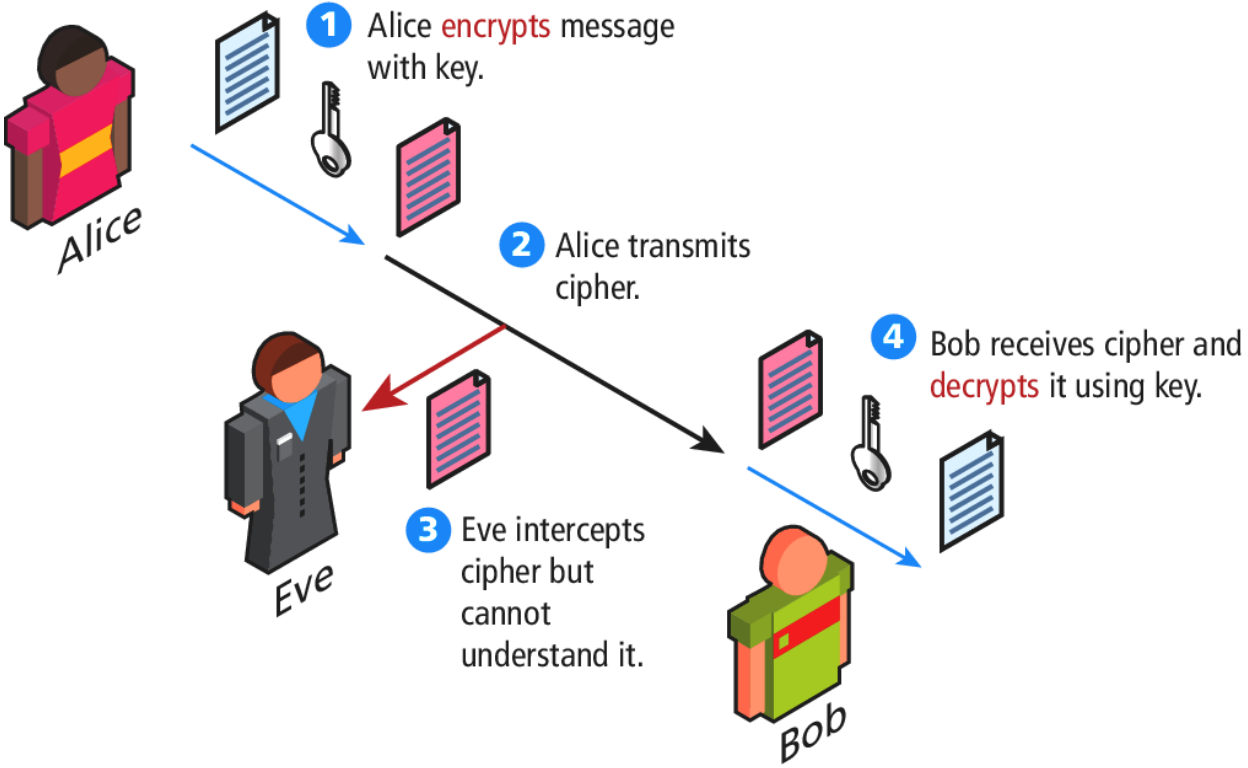
Cryptography

The problem



Cryptography

The goal



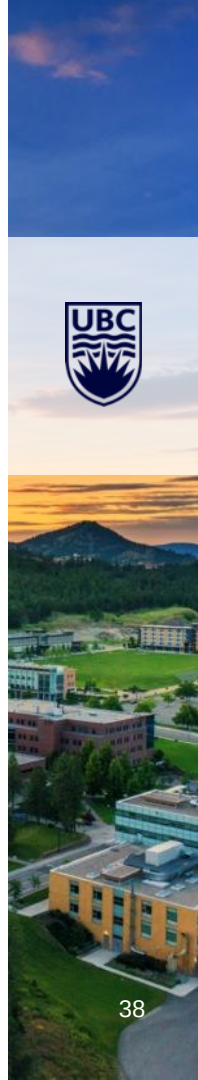
Cryptography

Some key terms

A **cipher** is a message that is scrambled so that it cannot easily be read, unless one has some secret **key**.

The **key** can be a number, a phrase, or a page from a book.

What is important in both ancient and modern cryptography is to keep the key a secret between the sender and the receiver.

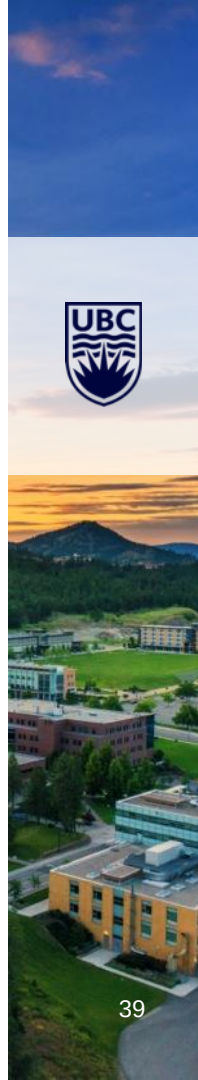


Cryptography

Substitution ciphers

A **substitution cipher** is one where each character of the original message is replaced with another character according to the encryption algorithm and key.

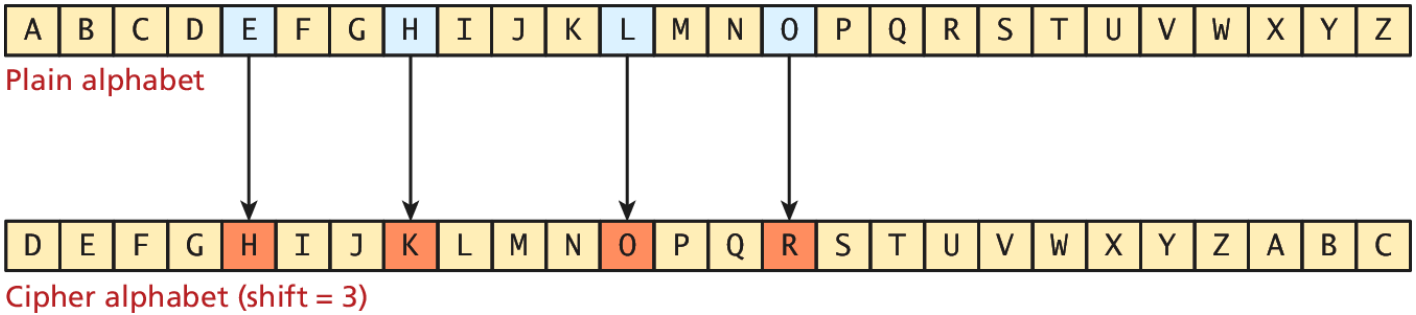
- Caesar
- Vigenère
- One Time Pad
- Modern Block Ciphers



Caesar

Substitution ciphers

The **Caesar cipher**, named for and used by the Roman Emperor, is a substitution cipher where every letter of a message is replaced with another letter, by shifting the alphabet over an agreed number (from 1 to 25).

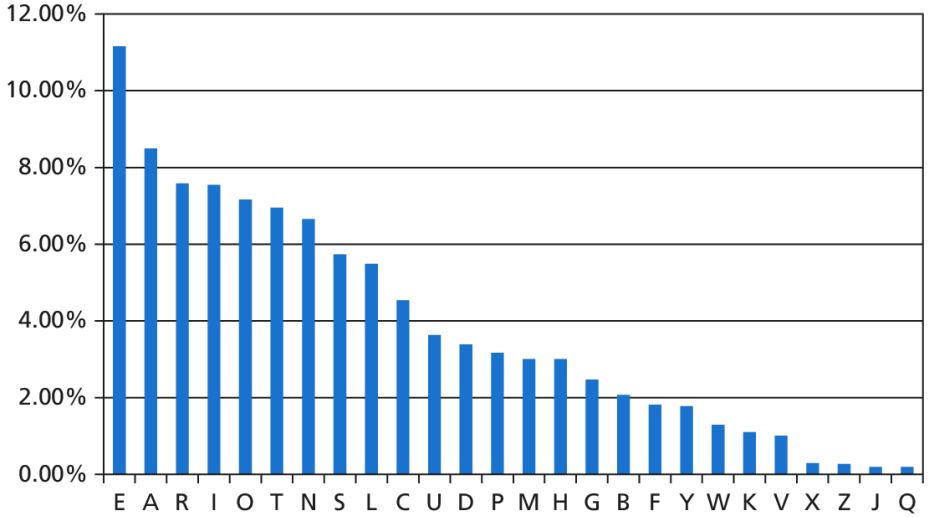


The message HELLO, for example, becomes KHOOR when a shift value of 3 is used. To decode, go the other direction

The Problem with Weak Ciphers

Letter distribution is not flat

The frequency of letters (and sets of two and three letters) is well known



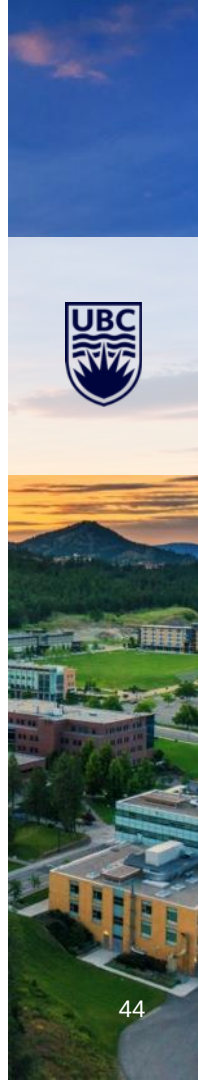
If you noticed the letter J occurring most frequently, it might well be the letter E

The Problem with Weak Ciphers

Letter distribution is not flat

Any good cipher must therefore try to make the resulting cipher text letter distribution relatively flat so as to remove any trace of the telltale pattern of letter distributions.

Simply swapping one letter for another does not do that, necessitating other techniques.

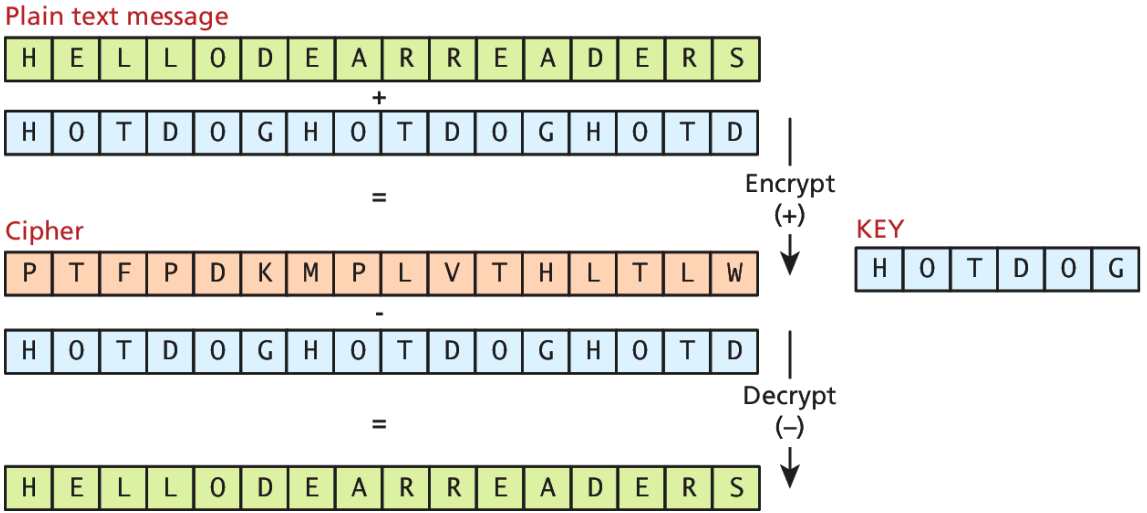


Vigenère

Early attempt to flatten letter distribution of ciphers

The **Vigenère cipher**, named for the sixteenth-century cryptographer, uses a keyword to encode a message.

The key phrase is written below the message and the letters are added together to form the cipher text as illustrated



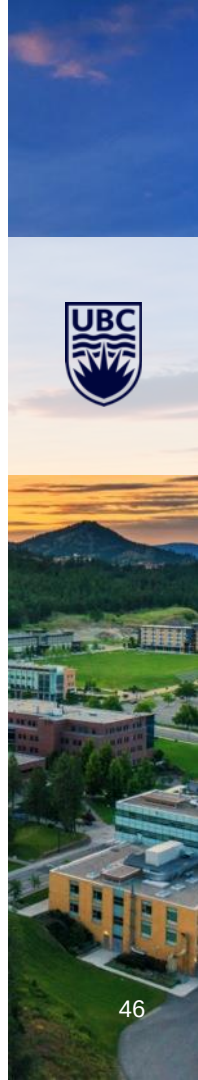
One Time Pad

Vigenere with an infinitely long key

The **one-time pad** refers to a (theoretically) perfect technique of cryptography where Alice and Bob both have identical copies of a very long sheet of numbers, randomly created.

Claude Shannon famously proved that the one-time pad is impossible to crack.

However, it is impractical to implement on a large scale and remains a theoretical benchmark that is rarely applied in practice.



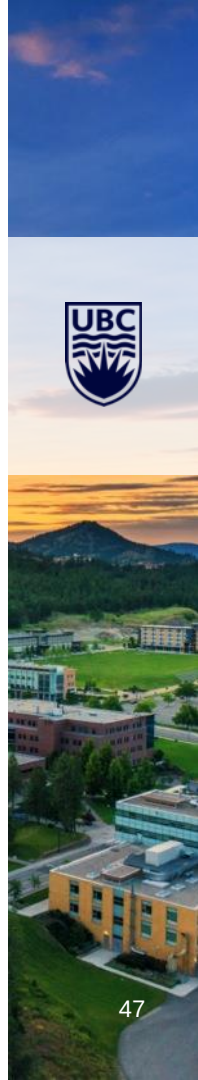
Modern Block Ciphers

Ciphers in the computer age

Block ciphers encrypt and decrypt messages using an iterative replacing of a message with another scrambled message using 64 or 128 bits at a time (the block).

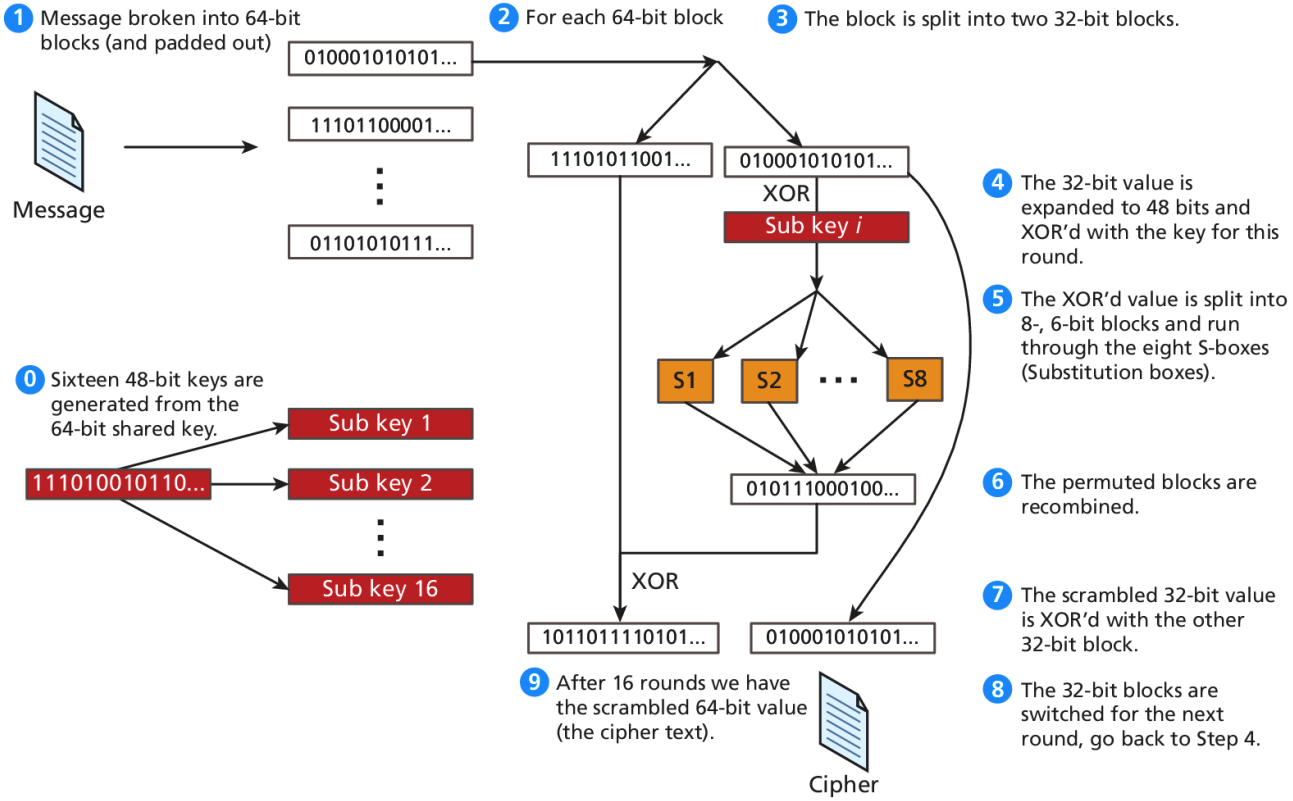
- The Data Encryption Standard (DES) and its replacement,
- The Advanced Encryption Standard (AES)

Are two-block ciphers still used in web encryption today



DES illustration

A modern block cipher



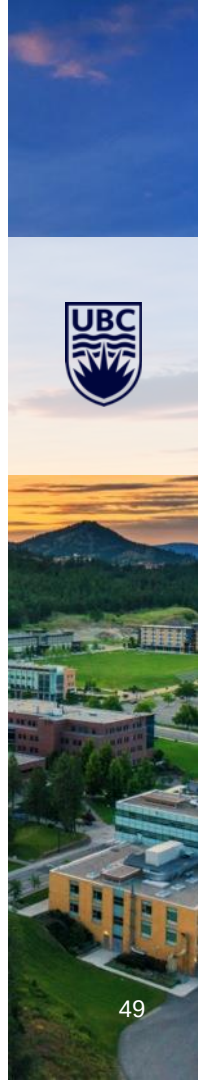
Symmetric Key Problem

How to exchange the key?

All of the ciphers we have covered thus far use the same key to encode and decode, so we call them **symmetric ciphers**.

The problem is that we have to have a shared private key. How?

- Over the phone?
- In an email?
- Through the regular mail?
- In person?



Public Key Cryptography

Solves the problem of key exchange

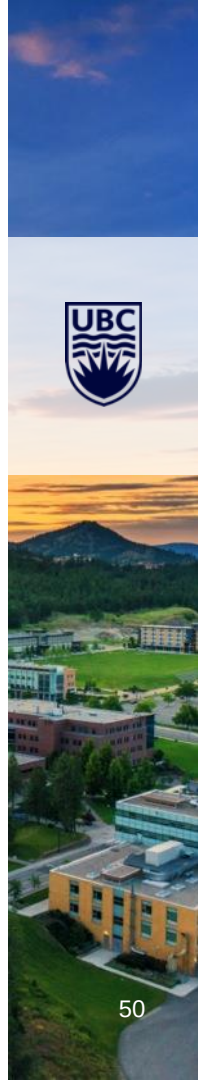
Public key cryptography (or **asymmetric cryptography**) solves the problem of the secret key by using two distinct keys:

- a public one, widely distributed
- another one, kept private

Algorithms like the **Diffie-Hellman Key Exchange,** published in 1976, provide the basis for secure communication on the WWW.

They allow a shared secret to be created out in the open, despite the presence of an eavesdropper.

RSA (Ron Rivest, Adi Shamir, and Leonard Adleman) is a practical public-key cryptosystems and is widely used for secure data transmission

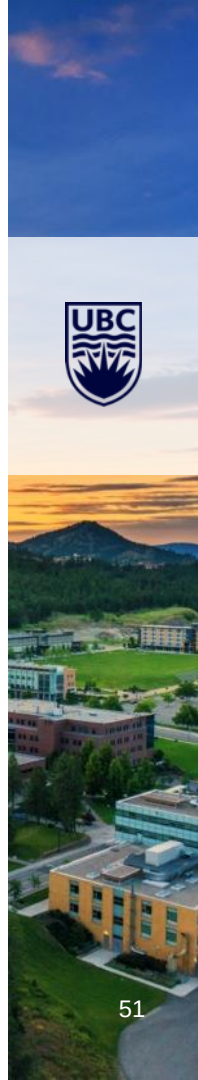


Digital Signatures

Confirming the sender is authentic

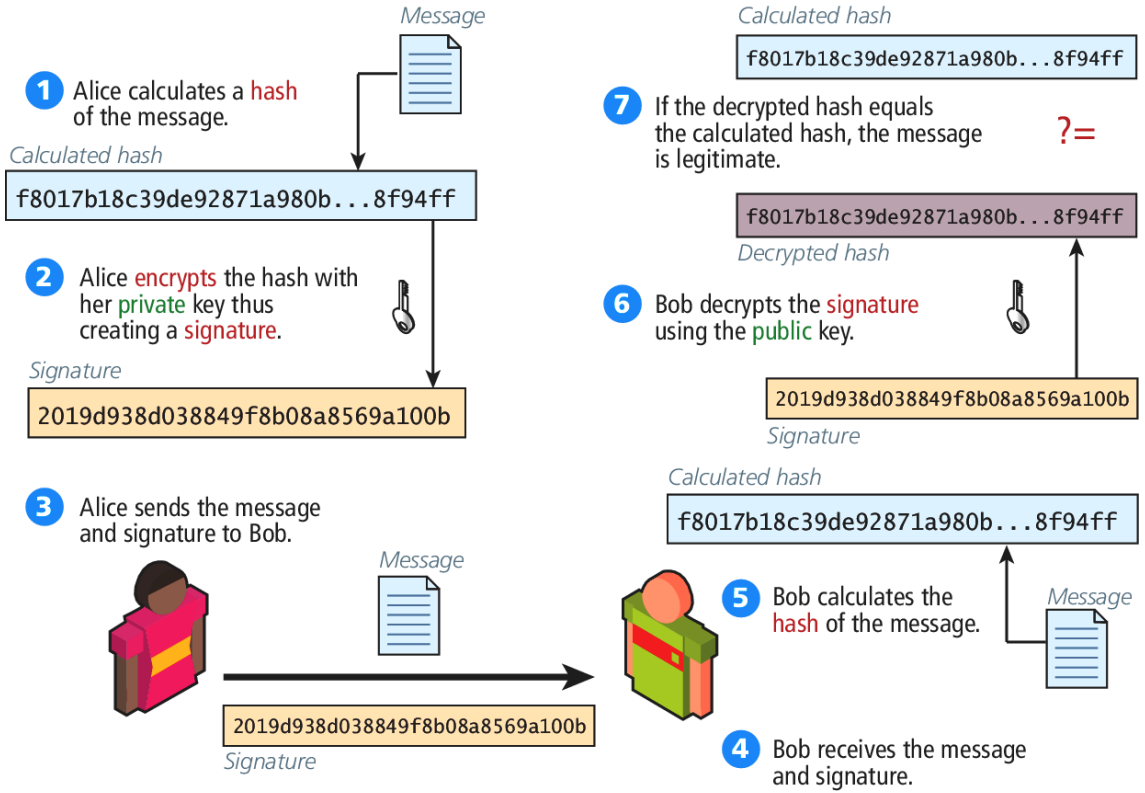
A **digital signature** is a mathematically secure way of validating that a particular digital document

- Was created by the person claiming to create it (authenticity),
- Was not modified in transit (integrity),
- Cannot be denied (non-repudiation).
- The process of signing a digital document can be as simple as encrypting a **hash** of the transmitted message.



Digital Signatures

Confirming the sender is authentic



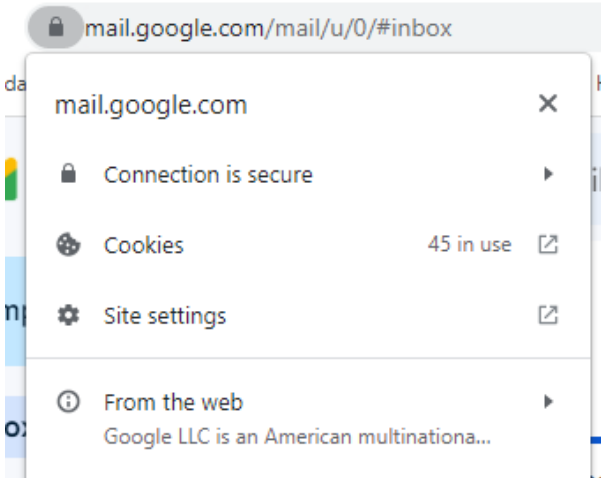
4. Hypertext Transfer Protocol Secure (HTTP)

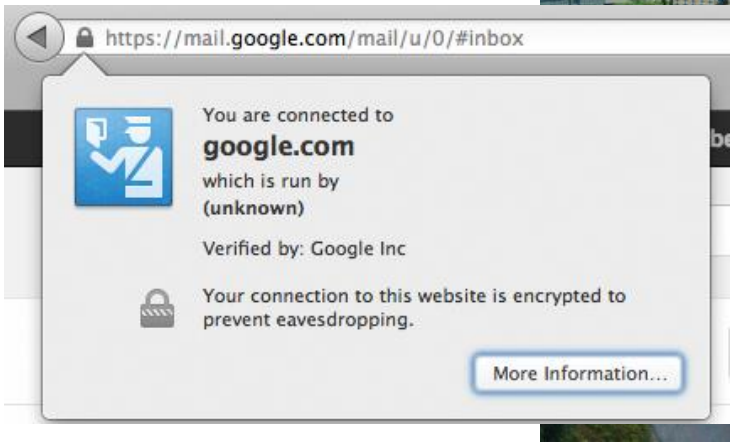
HTTPS

Secure HTTP

HTTPS is the HTTP protocol running on top of the Transport Layer Security (TLS).

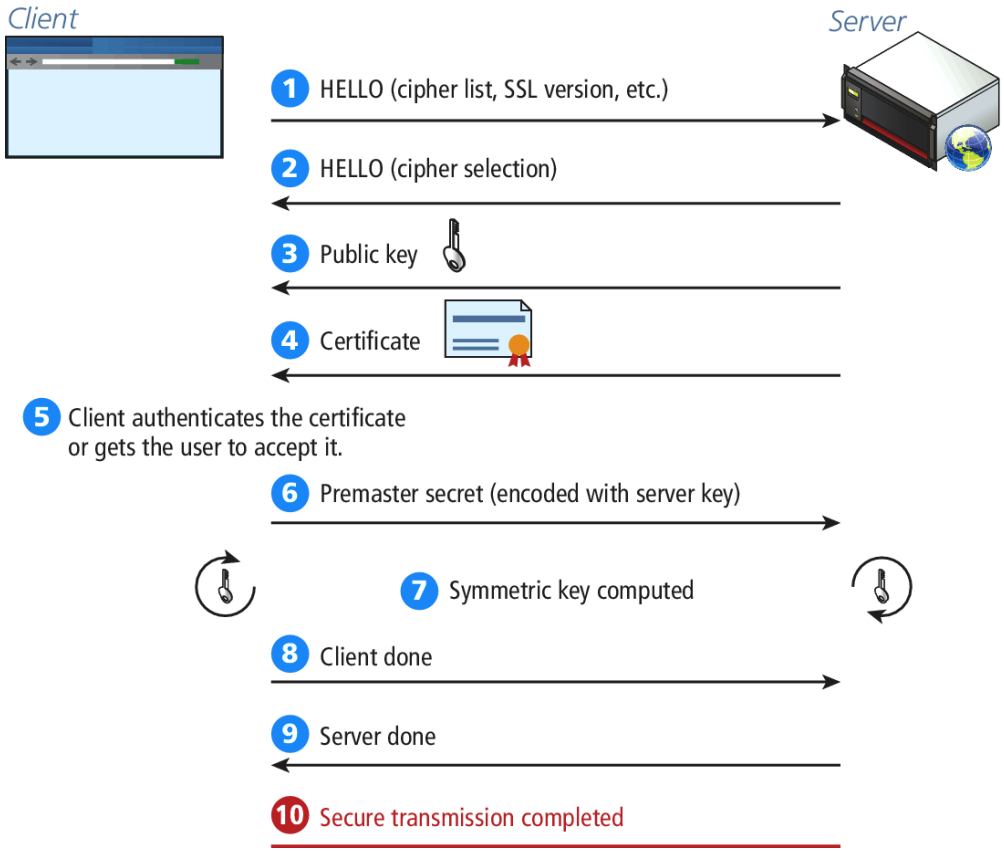
It’s easy to see from a client’s perspective that a site is secured by the little padlock icons in the URL bar used by most modern browsers





HTTPS

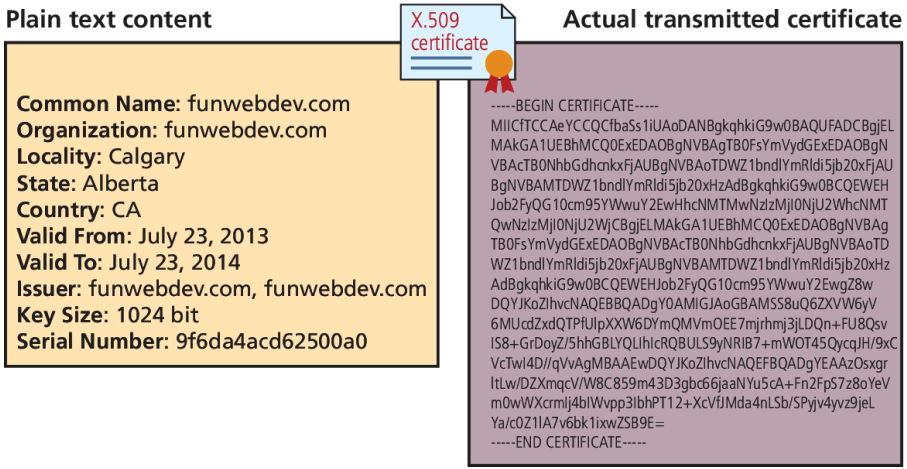
Secure Handshakes



HTTPS

Certificates

The certificate that is transmitted during the handshake is actually an X.509 certificate, which contains many details including the algorithms used, the domain it was issued for, and some public key information.



HTTPS

Certificate Authorities

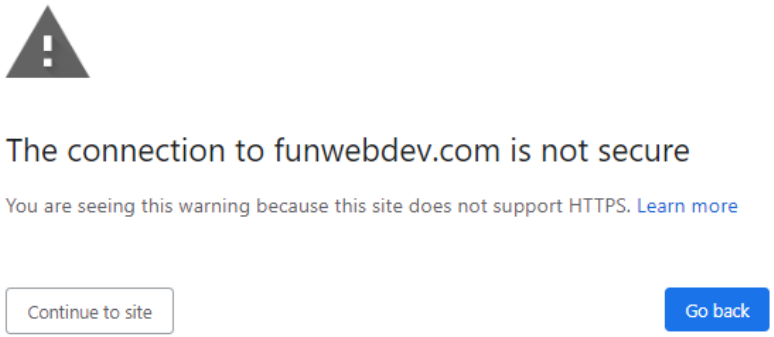
A **Certificate Authority** (CA) allows users to place their trust in the certificate since a trusted, independent third party signs it.

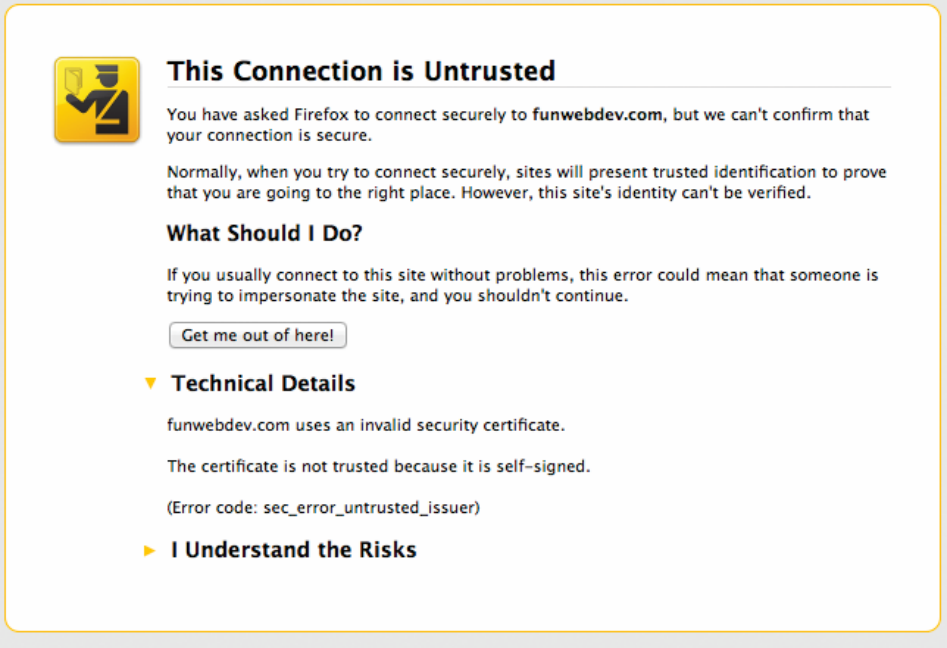
The CA's primary role is to validate that the requestor of the certificate is who they claim to be, and issue and sign the certificate containing the public keys so that anyone seeing them can trust they are genuine.

HTTPS

Self-Signed Certificates

Self-signed certificates provide the same level of encryption, but the validity of the server is not confirmed.





5. Security Best Practices

Best Practices

Some good things to do

It’s now time to discuss some practical things you can do to harden your system against attacks

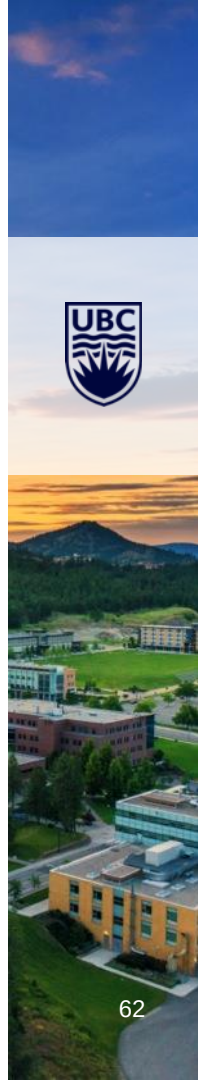
- Secure Data Storage
- Monitor Your Systems
- Audit and Attack Thyself

Secure Data Storage

Not if, but when

Even household names like Sony, Citigroup, Target and GE Money have had their systems breached and data stolen.

If even globally active companies can be impacted, you must ask yourself: when (not if) you are breached, what data will the attacker have access to?



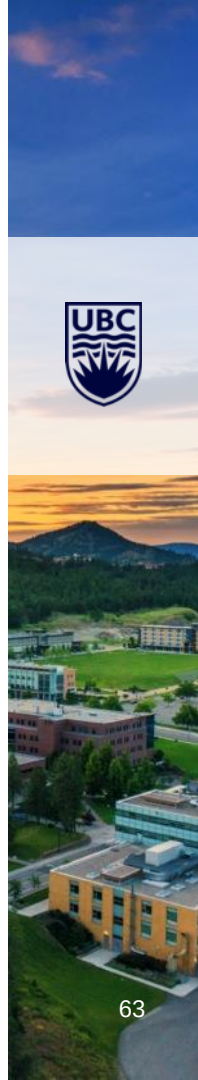
Data Storage

Plain Text Password Storage

Anyone who can see the database can see all the passwords

- Issue of confidentiality:
  - Passwords in plain text are subject to disclosure
- Issue of integrity
  - Passwords in plain text can be stolen and used to authenticate another user or changed to something else

UserID (int)	Username (varchar)	Password (varchar)
1	ricardo	password
2	randy	password



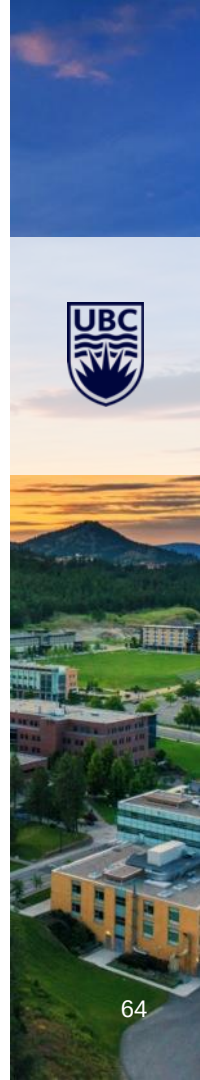
Data Storage

Naïve credential storage example

```
//Insert the user with the password
function insertUser($username,$password){
   $link = mysqli_connect("localhost", "my_user", "my_password",
                          "Login");
   $sql = "INSERT INTO Users(Username,Password)
           VALUES('$username','$password')");
   mysqli_query($link, $sql);            //execute the query
}

//Check if the credentials match a user in the system
function validateUser($username,$password){
   $link = mysqli_connect("localhost", "my_user", "my_password",
                          "Login");
   $sql = "SELECT UserID FROM Users WHERE Username='$username' AND
           Password='$password'";
   $result = mysqli_query($link, $sql);   //execute the query
   if($row = mysqli_fetch_assoc($result)){
      return true;   //record found, return true.
   }

   return false; //record not found matching credentials, return false
}
```



Data Storage

How can we make it better

Two techniques that improve the integrity of your data.

- Secure Hash
- Salted Secure Hash

Data Storage

Secure Hash

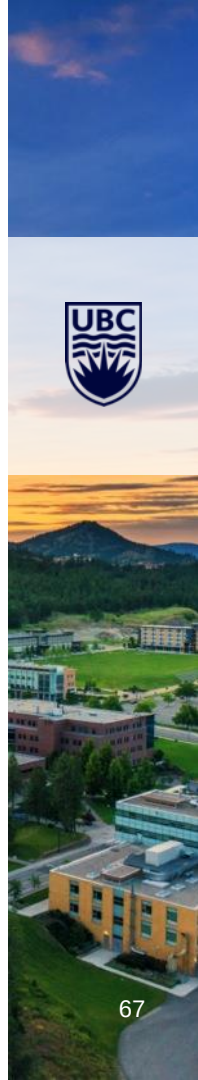
UserID (int)	Username (varchar)	Password (varchar)
1	ricardo	5f4dcc3b5aa765d61d8327deb882cf99
2	randy	5f4dcc3b5aa765d61d8327deb882cf99

TABLE 16.3 Users Table with MD5 Hash Applied to Password Field

```
//Insert the user with the password being hashed by MD5 first.
function insertUser($username,$password){
 $link = mysqli_connect("localhost", "my_user", "my_password",
                        "Login");
 $sql = "INSERT INTO Users(Username,Password)
         VALUES('$username',MD5('$password'))");
 mysqli_query($link, $sql); //execute the query
}

//Check if the credentials match a user in the system with MD5 hash
function validateUser($username,$password){
 $link = mysqli_connect("localhost", "my_user", "my_password",
                        "Login");
 $sql = "SELECT UserID FROM Users WHERE Username='$username' AND
         Password=MD5('$password')";

 $result = mysqli_query($link, $sql);    //execute the query
 if($row = mysqli_fetch_assoc($result)){
   return true;  //record found, return true.
 }
 return false;  //record not found matching credentials, return false
}
```



Data Storage

Secure Hash

UserID (int)	Username (varchar)	Password (varchar)
1	ricardo	5f4dcc3b5aa765d61d8327deb882cf99
2	randy	5f4dcc3b5aa765d61d8327deb882cf99

TABLE 16.3 Users Table with MD5 Hash Applied to Password Field

A simple Google search for the string stored in our newly defined table:

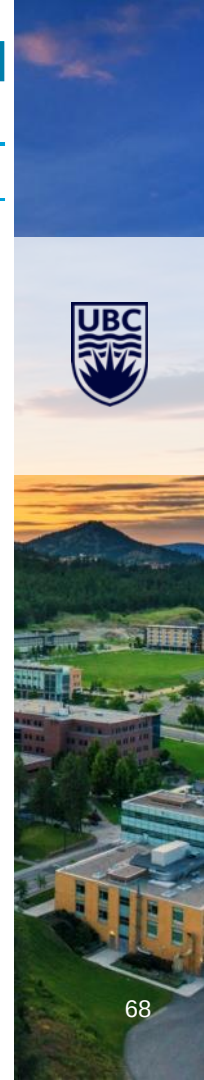
`5f4dcc3b5aa765d61d8327deb882cf99` brings up dozens of results which tell you that that string is indeed the MD5 digest for *password*.

The technique of adding some noise (unique) to each password is called **salting** the password and makes your passwords more secure.

UserID (int)	Username (varchar)	Password (varchar)	Salt
1	ricardo	edee24c1f2f1a1fda2375828fbeb6933	12345a
2	randy	ffc7764973435b9a2222a49d488c68e4	54321a

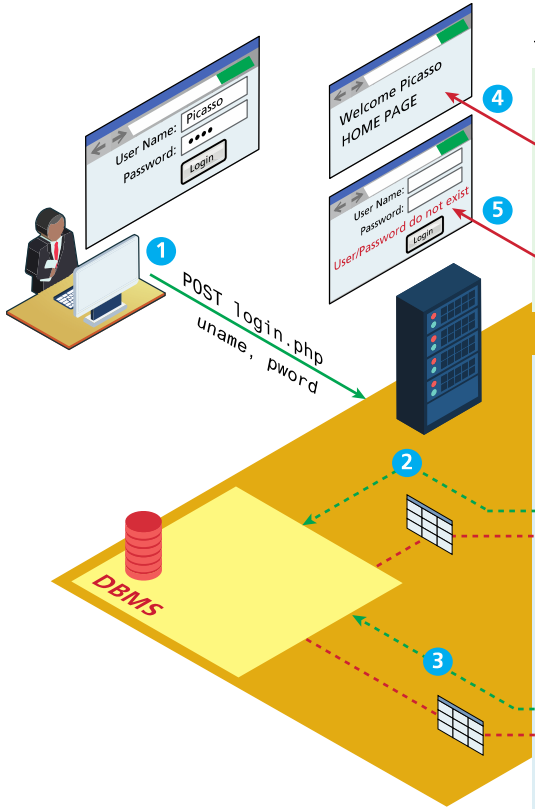
```
MD5("password12345a");   // edee24c1f2f1a1fda2375828fbeb6933

MD5("password54321a");   // ffc7764973435b9a2222a49d488c68e4
```



Security Best Practices

Data Storage



login.php

```
session_start();
if (isset($_POST['uname']) ) {
   if (validateUser($_POST['uname'],$_POST['pword']) {
      $_SESSION['user']=$_POST['uname'];
      echo HomeScreen();
   }
   else
      echo LoginFormErrorPage();
}
```

```
function validateUser($username,$password) {
   $pdo = new PDO(DBCONN_STRING,DBUSERNAME,DBPASS);
   $sql = "SELECT Salt FROM Users WHERE Username=?";
   $statement = $pdo->prepare($sql);

   $statement->execute(array($username));
   $salt = $statement->fetchColumn();

   $saltSql = "SELECT UserID FROM Users
               WHERE Username=? AND Password=?";

   $params = array($username, md5($password . $salt));
   $statment = $pdo->prepare($saltSql);
   $statement->execute($params);
   if ($statement->rowCount())
      return true;
   }
   return false;
}
```

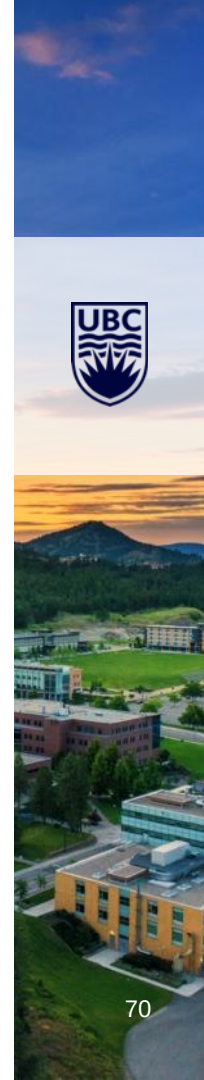
Data Storage

Secure Salted Hash

`mcrypt_create_iv` deprecated in php 7.1.0 and replaced with `random_bytes()`

```
function generateRandomSalt(){
 return base64_encode(mcrypt_create_iv(12), MCRYPT_DEV_URANDOM));
}
//Insert the user with the password salt generated, stored, and
//password hashed
function insertUser($username,$password){
 $link = mysqli_connect("localhost", "my_user", "my_password",
                        "Login");
 $salt = generateRandomSalt();
 $sql = "INSERT INTO Users(Username,Password,Salt)
         VALUES('$username',MD5('$password$salt'), '$salt')");
  mysqli_query($link, $sql); //execute the query
}

//Check if the credentials match a user in the system with MD5 hash
//using salt
function validateUser($username,$password){
 $link = mysqli_connect("localhost", "my_user", "my_password",
                        "Login");
 $sql = "SELECT Salt FROM Users WHERE Username='$username'";
 $result = mysqli_query($link, $sql);    //execute the query
 if($row = mysqli_fetch_assoc($result)){
   //username exists, build second query with salt
   $salt = $row['Salt'];
   $saltSql = "SELECT UserID FROM Users WHERE Username='$username'
               AND Password=MD5('$password$salt')";";
   $finalResult = mysqli_query($link, $saltSql);
   if($finalrow = mysqli_fetch_assoc($finalResult))
      return true;  //record found, return true.
 }
 return false; //record not found matching credentials, return false
}
```



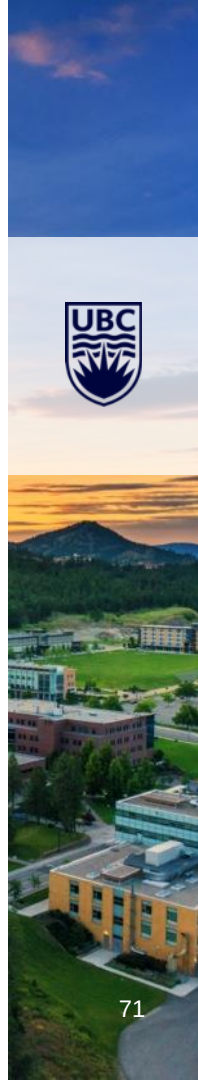
Monitor Your Systems

Systems Monitors

One of the best ways to mitigate damage is to detect an attack as quickly as possible, rather than let an attacker take their time in exploiting your system once inside.

We can detect intrusion

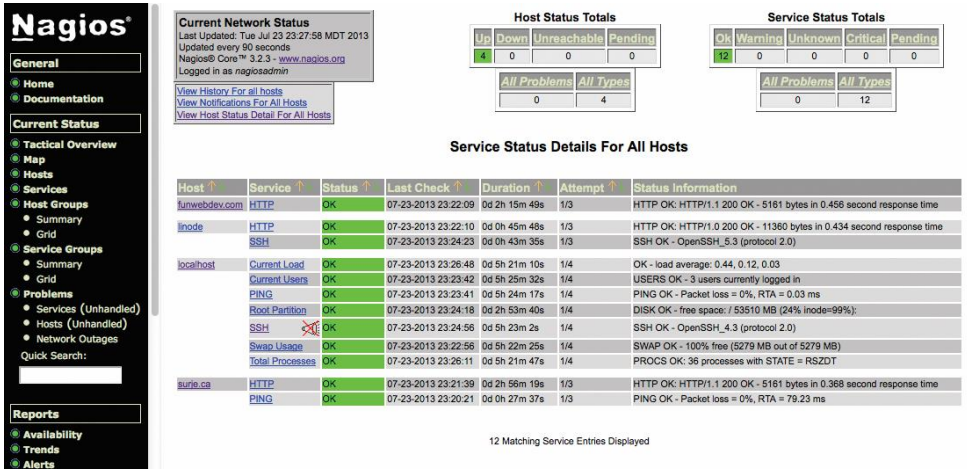
- directly by watching login attempts, and
- indirectly by watching for suspicious behavior like a web server going down.



Monitor Your Systems

There are tools that allow you to pre-configure a system to check in on all your sites and servers periodically.

You see the status and history of your devices, and sends out notifications by email as per your preferences.



Monitor Your Systems

Access Monitors

It is not uncommon for there are thousands of attempted login attempts being performed all day long

```
Jul 23 23:35:04 funwebdev sshd[19595]: Invalid user randy from
    68.182.20.18
Jul 23 23:35:04 funwebdev sshd[19596]: Failed password for invalid
    user randy from 68.182.20.18 port 34741 ssh2
```

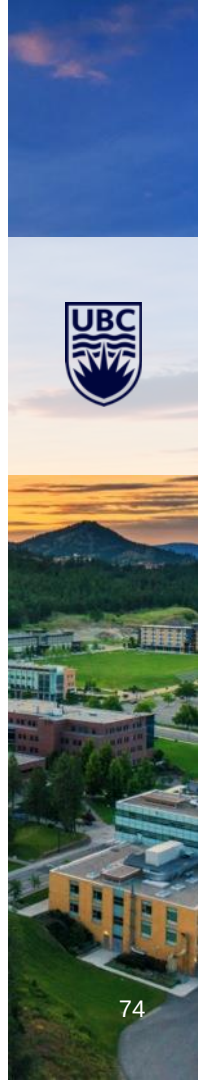
Audit and Attack Thyself

Attacking the systems you own or are authorized to attack in order to find vulnerabilities is a great way to detect holes in your system and patch them before someone else does.

It should be part of all the aspects of testing, including the deployment tests, but also unit testing done by developers. This way SQL injection, for example, is automatically performed with each unit test, and vulnerabilities are immediately found and fixed.

There are a number of companies that you can hire to test your servers and report on what they've found.

You can perform your own analysis using open-source attack tools such as w3af, which provide a framework to test your system including SQL injections, XSS, bad credentials, and more.



6. Common Threat Vectors

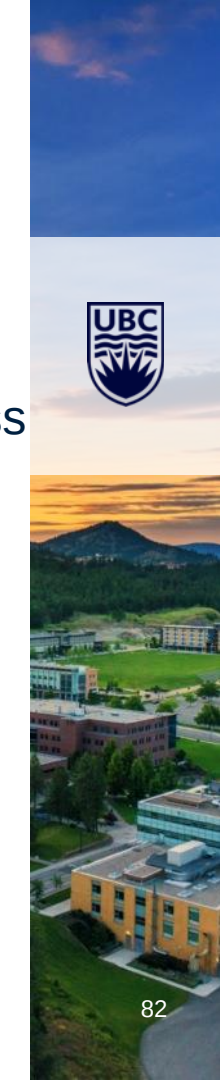
Brute-Force Attack

An intruder simply tries repeatedly guessing the password. For instance an automated script might try looping through words in the dictionary or use combinations of words, numbers, and symbols.

Automated intrusion blocking may provide protection by blocking the IP address of the script but might not be enough (possible to hide IP addresses).

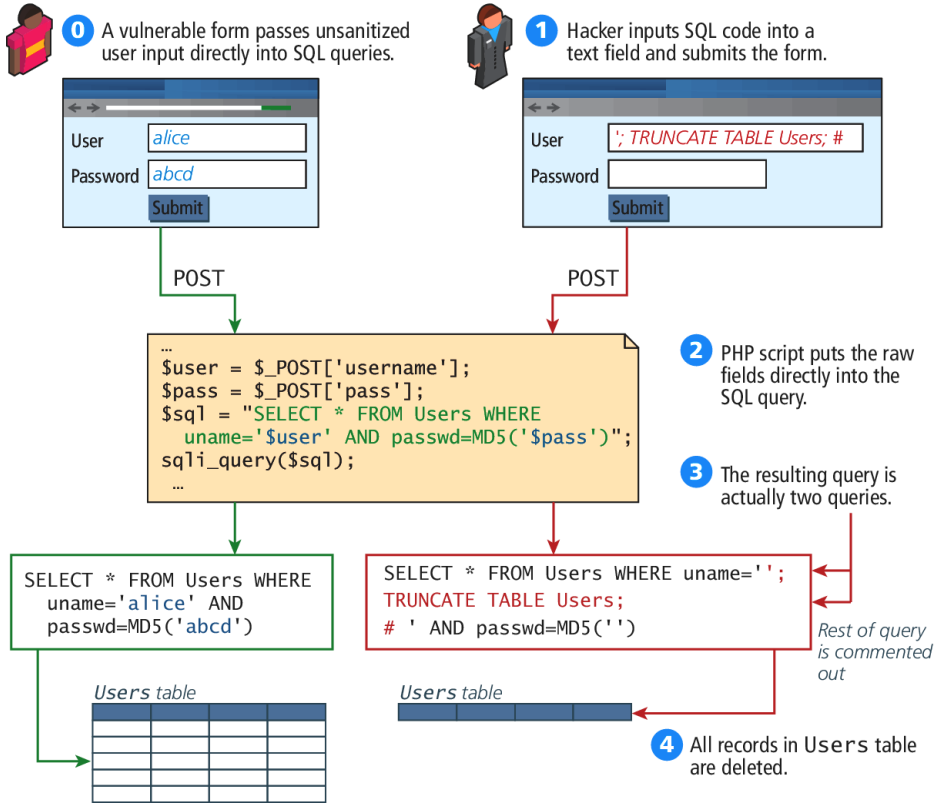
Possible solution is to throttle login attempts

- Lock user account after set number of incorrect guesses
- Add time delay between login attempts
    - first two or three login attempts might have no delays, but login attempts four through seven have a delay of 5 seconds, attempts after the seventh are delayed 10 minutes, etc.
- CAPTCHA



SQL Injection

Using user input fields for evil.



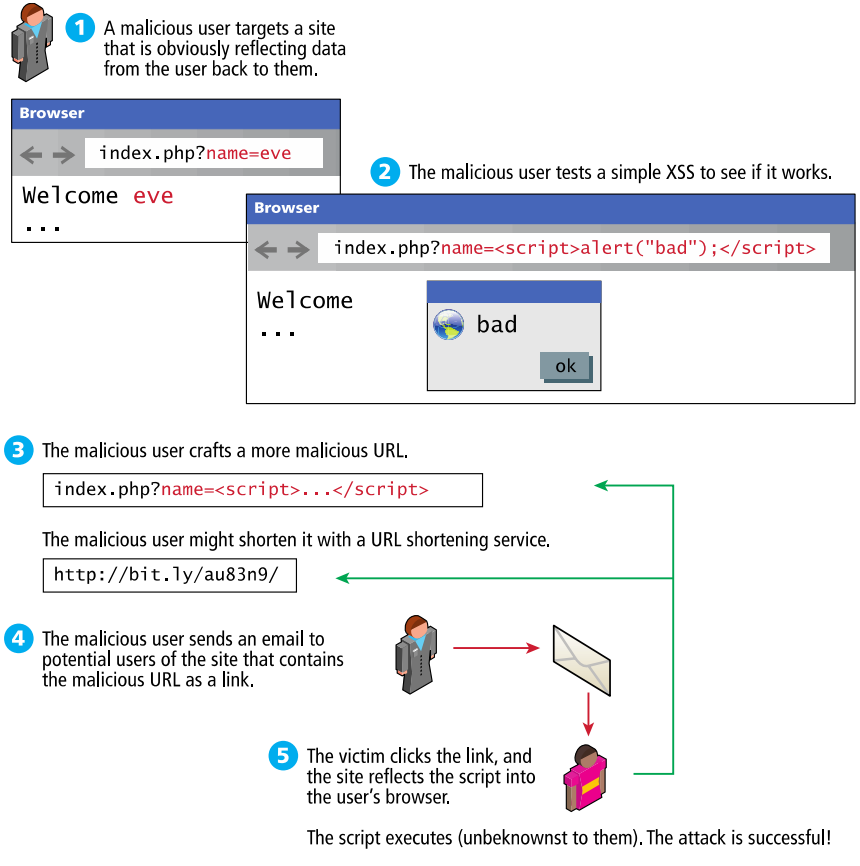
Possible solutions:

- Sanitize user input by applying sanitization functions (ex. mysqli_real_escape_string() method or the quote() method) and using prepared statements.

- Apply least possible privileges by assigning users and applications the privileges they need to complete their work, but no more.
  Ex. 3 types of database users:
  - One with read only privileges for non authenticated users
  - One with write privileges
  - Admin that can add, drop, truncate tables

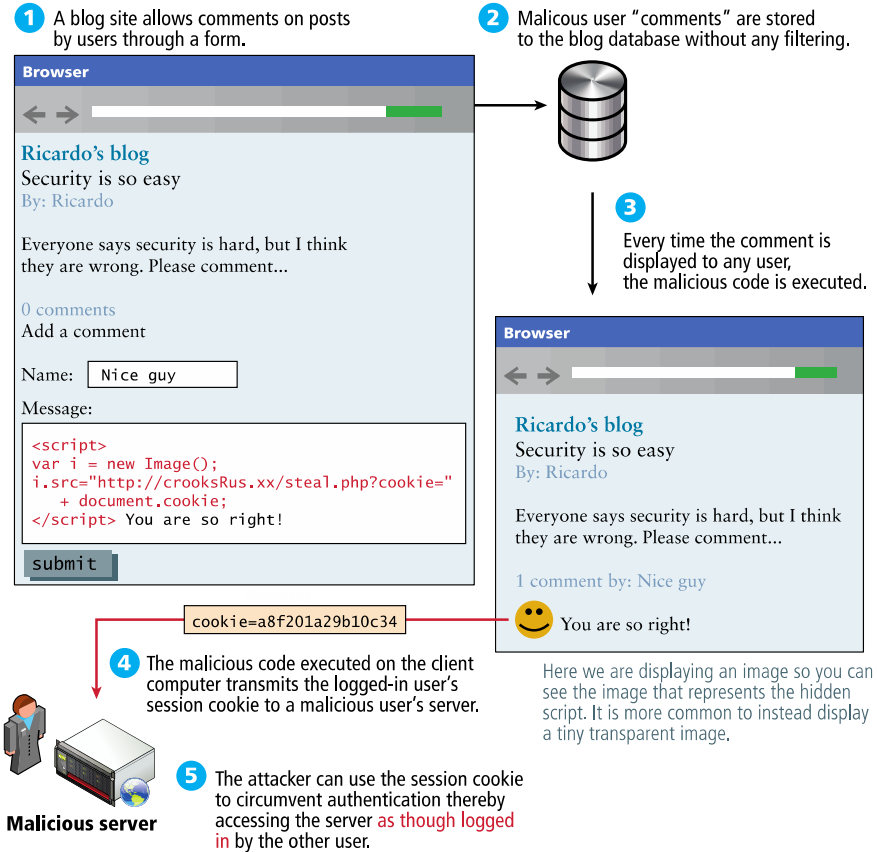
Common Threat Vectors

Cross-Site Scripting (XSS)



Common Threat Vectors

Stored Cross-Site Scripting (XSS)



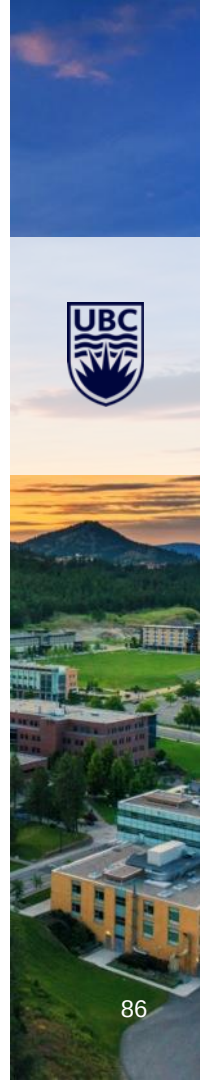
Thwarting XSS

Filter User Input

- strip_tags() removes all the HTML tags

But hackers are sophisticated, so libraries such as HTMLPurifier or HTML sanitizer from Google allows you to easily remove a wide range of dangerous characters that could be used as part of an XSS attack

```
$user= $_POST['uname'];
$purifier = new HTMLPurifier();
$clean_user = $purifier->purify($user);
```

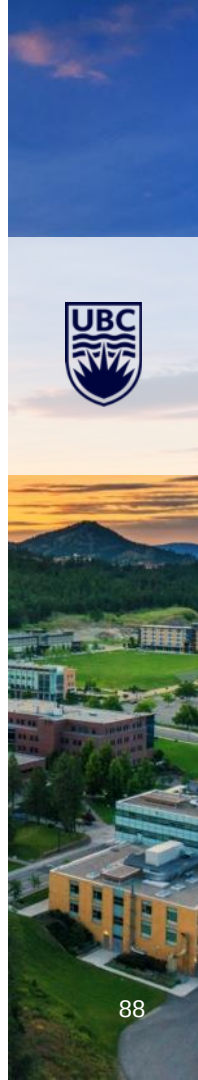


Common Threat Vectors

Insecure Direct Object Reference

An **insecure direct object reference** is a fancy name for when some internal value is exposed to the user, and attackers can then manipulate these internal keys to gain access to things they should not have access to.

For instance, if a user can determine that his or her uploaded photos are stored sequentially as **/images/99/1.jpg**, **/images/99/2.jpg**, . . . , they might try to access images of other users by requesting **/images/101/1.jpg**.



Insecure Direct Object Reference

One strategy for protecting your site against this threat is to obfuscate URLs to use hash values rather than sequential names. For example:

Rather than store images as

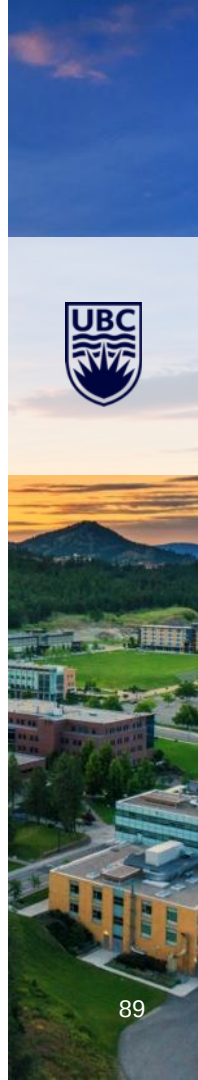
- 1.jpg, 2.jpg . . .

Generate URLs like

- 9a76eb01c5de4362098.jpg

Possible solution:

- Using a one way hash rather than sequential names.

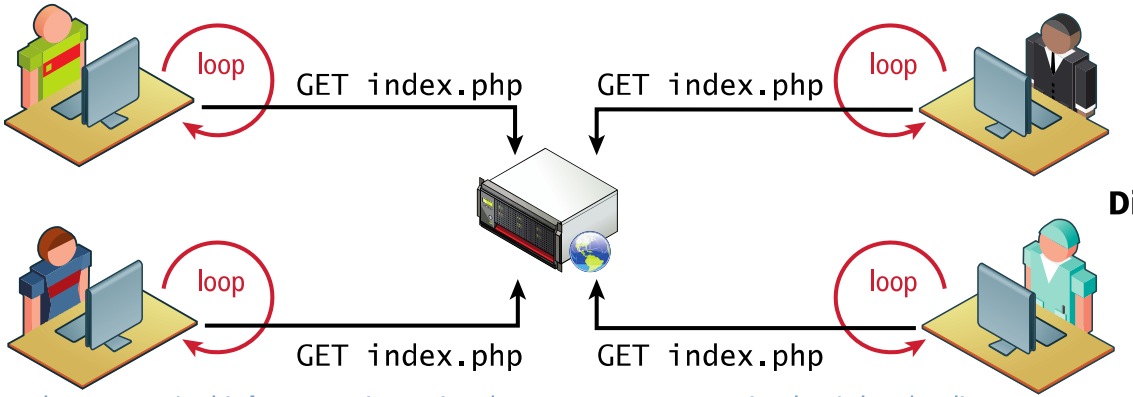


Common Threat Vectors

Denial of Service

Denial of service

This computer is running a program or script
that is repeatedly requesting a page from the server.



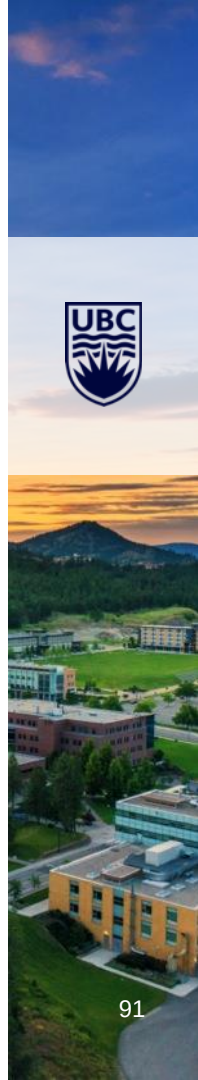
Distributed DOS

Each computer in this **bot army** is running the same program or script that is bombarding
the server with requests. These users are probably unaware that this is happening.

DDoS

Interestingly, defense against this type of attack is similar to preparation for a huge surge of traffic, that is, caching dynamic pages whenever possible, and ensuring you have the bandwidth needed to respond.

Unfortunately, these attacks are very difficult to counter, as illustrated by attack on the spamhaus servers, which generated 300Gbps worth of requests

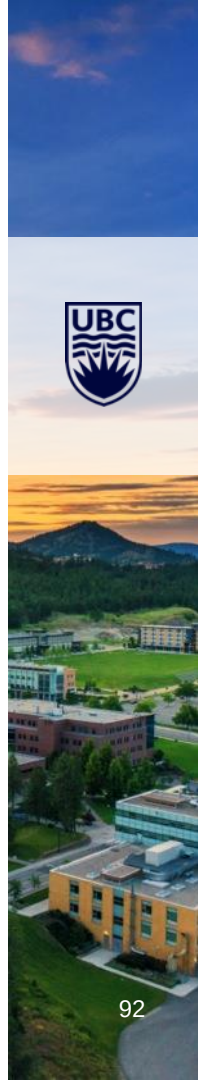


Security Misconfiguration

That's a wide net

There are many systems that can be badly configured:

- Out-of-date Software:
  - Solution is to update software as quickly as possible
- Open Mail Relays
- More Input Attacks
  - Virtual Open Mail Relay
  - Arbitrary Program Execution



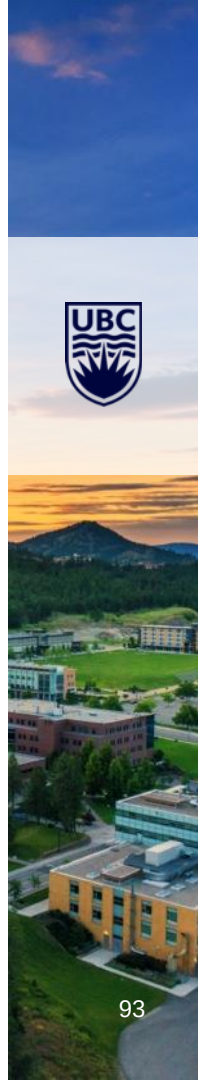
Open Mail Relays

Letting someone send mail as you

An **open mail relay** refers to any mail server that allows someone to route email through without authentication.

Open relays are troublesome since spammers can use your server to send their messages rather than use their own servers.

This means that the spam messages are sent as if the originating IP address was your own web server! If that spam is flagged at a spam agency like spamhaus, your mail server's IP address will be blacklisted, and then many mail providers will block legitimate email from you.



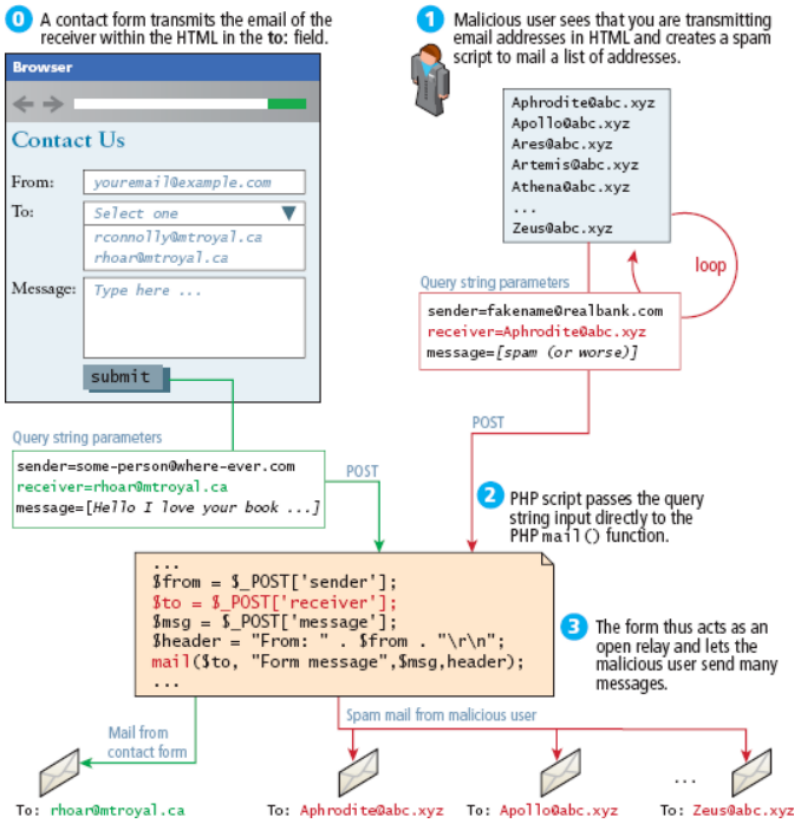
Virtual Open Mail Relay

Consider a website that uses an HTML form to allow users contact website admin or others.

If form allows you select recipient from a dropdown menu, it could expose mail server as virtual open mail relay.

Contact form can be abused as attacker could send to any email they want.

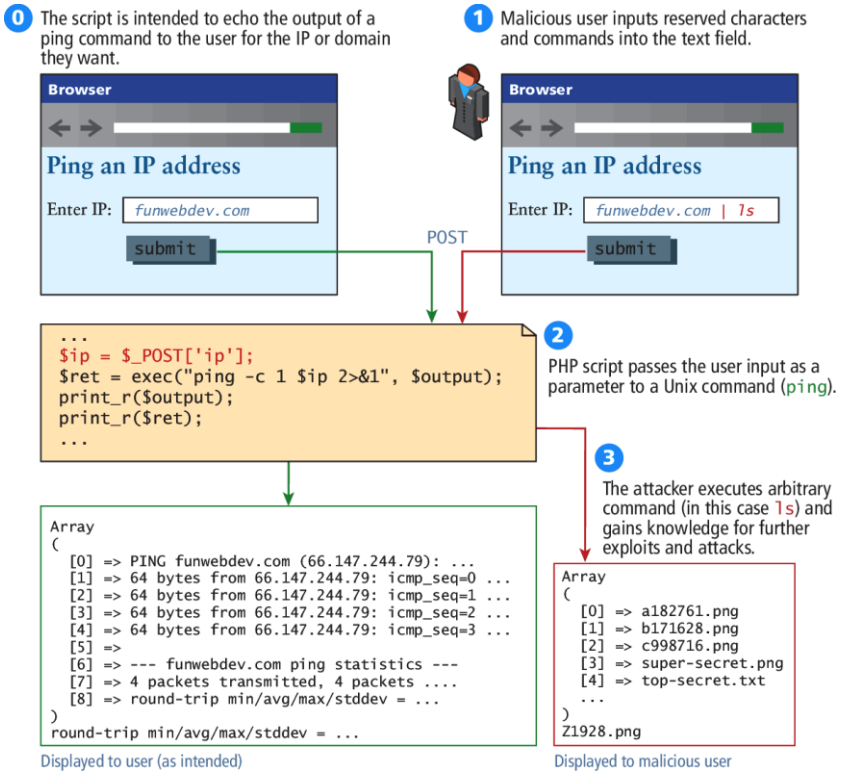
Solution is to use integer that corresponds to ID in user table requiring database to lookup validity of recipient.



Arbitrary Program Execution

Let them run any program they want

Solution:
Sanitize input with escapeshellarg() and be mindful of how user input is being passed to the shell.



Summary

- Described a wide range of security principles and practices
- Described best practices for authentication systems and data storage
- Described and applied key cryptography, SSL, and certificates
- Described how to proactively protect their sites against common attacks

Acknowledgement

- These slides were adapted from the lecture slides of Dr. Fazackerley

Reference

- Fundamentals of Web Development By Randy Connolly and Ricardo Hoar, 2nd Edition, ISBN: 0134481267