COMP 307

Principles of Web Development

Lecture 20

Unit 4 – Security

Security 1

Announcements

- 2 lectures left
  - Counting this lecture
  - Security 1 – theory
  - Security 2 – a practical example
- Midterm – November 23 – online, during class
- Office hours extended during lecture time
- Survey will be posted next class
  - Note: course evaluations also available

Class Outline

- Attack vectors
- Security techniques
  - File permissions
  - Browser support (cache & cookies)
  - Obfuscation
  - Encryption
  - Tickets

Why is security problematic?

- With enough computers, time and money, any security measure can be compromised.
  - AES-256 is safe from brute force attacks, but there are smarter ways to find the keys on your disk
  - With bad luck, bad-guy might break in quickly
- Solution?
  - We cannot stop bad-guys 
  - So, make it hard for bad-guys 

Why is security problematic?

- For example:
  - Anything you want to run on someone's browser can be seen by the user of the browser (inspect DOM)
  - Fishing attacks can trick people into revealing information (like passwords)
  - A cache attack can be used to get a copy of a key (virus, worm, mole)
  - People use personal information as their passwords
  - Install a key-stroke monitor
  - Find someone's laptop unattended to install bots or copy keys

Attack Vectors

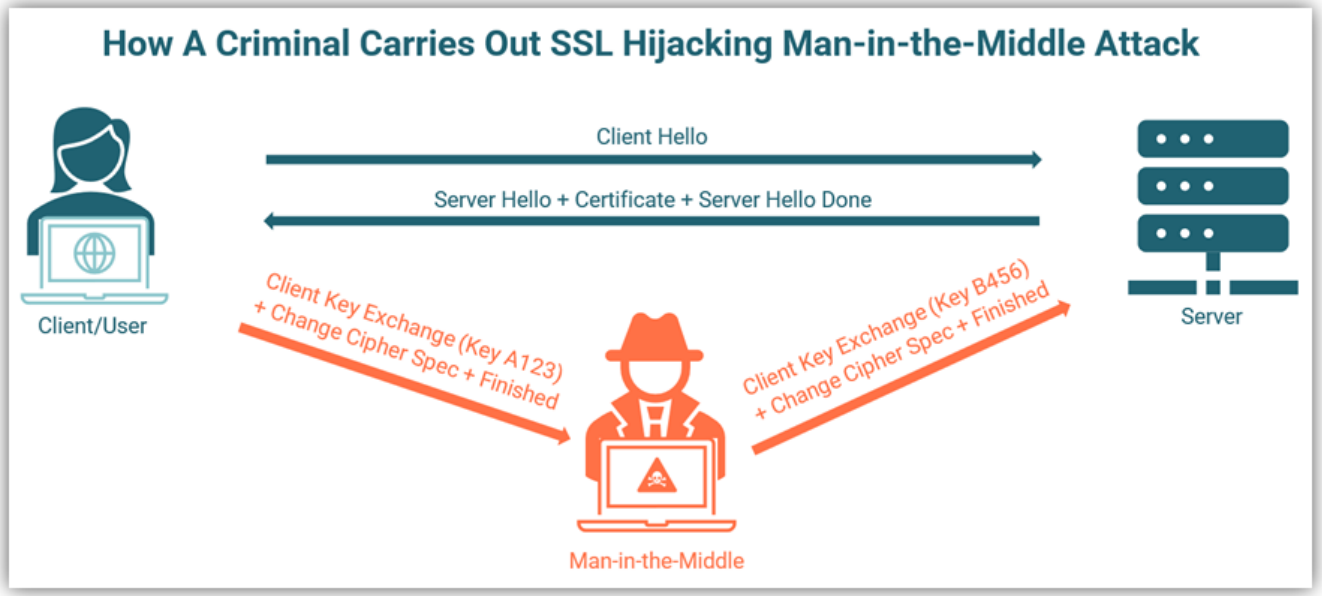
Security 1

Contents

Attack Vectors

- "Bad Guys" – generic term referring to hackers, industrial espionage, political agents, and the dark web.
  - Curious students could sometimes find themselves unintentionally here
- "Vector" – a technique used to compromise a server or client application.
  - Man-in-the-middle
  - Denial-of-service
  - Privacy issues: Absence, Adwords, Mmap

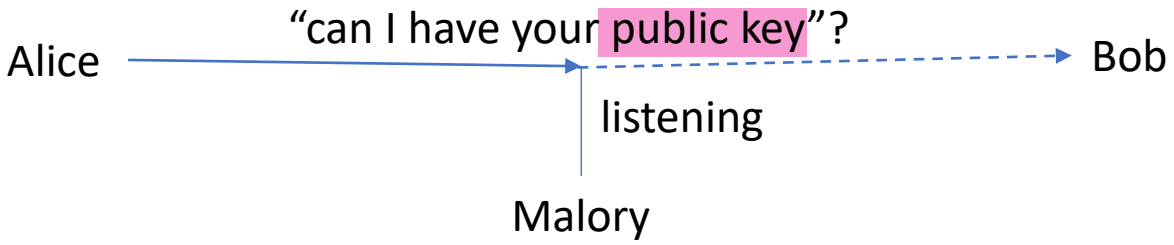
Man-in-the-Middle



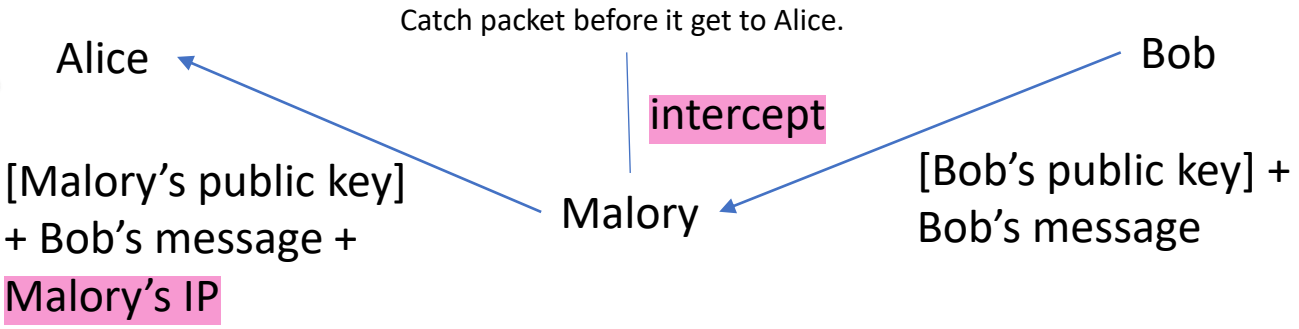
- Is a cyberattack where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other, as the attacker has inserted themselves between the two parties.

Man-in-the-Middle Attack

Step 1



Step 2



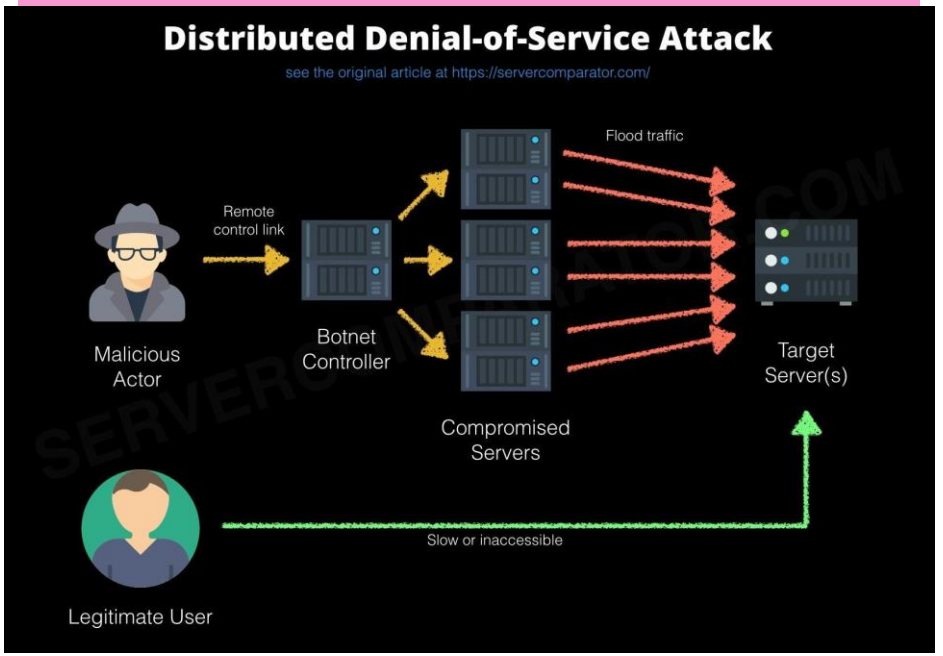
Step 3

- Alice now uses Malory's IP to send messages to Bob.
- Malory records & modifies packet and sends to Bob.
- Repeat….

Man-in-the-Middle Defense

- Regular authentication tests
  - At regular timed intervals
  - When accessing a new resource
- Alice or Bob have software to verify that the packet's IP and/or public key was not modified
- Forensic analysis with Wireshark

Denial-of-Service



- A bad-buy uses multiple servers to send false API requests to a target server.
- The volume of fake requests keeps the target server busy filling up its packet queue and waists valuable CPU time processing fake requests.
- Valid user requests cannot get on the queue

Denial-of-Service Attacks

- DoS attack
  - Denial-of-service attack
  - A bad-guy uses a single application to flood a server with fake requests.
- DDoS attack
  - Distributed denial-of-service attack
  - The bad-guy uses many applications to flood a server with fake requests
  - These “many applications” could include installing bots on random people’s computers or random organization’s servers to all send these fake requests at an appointed date-time

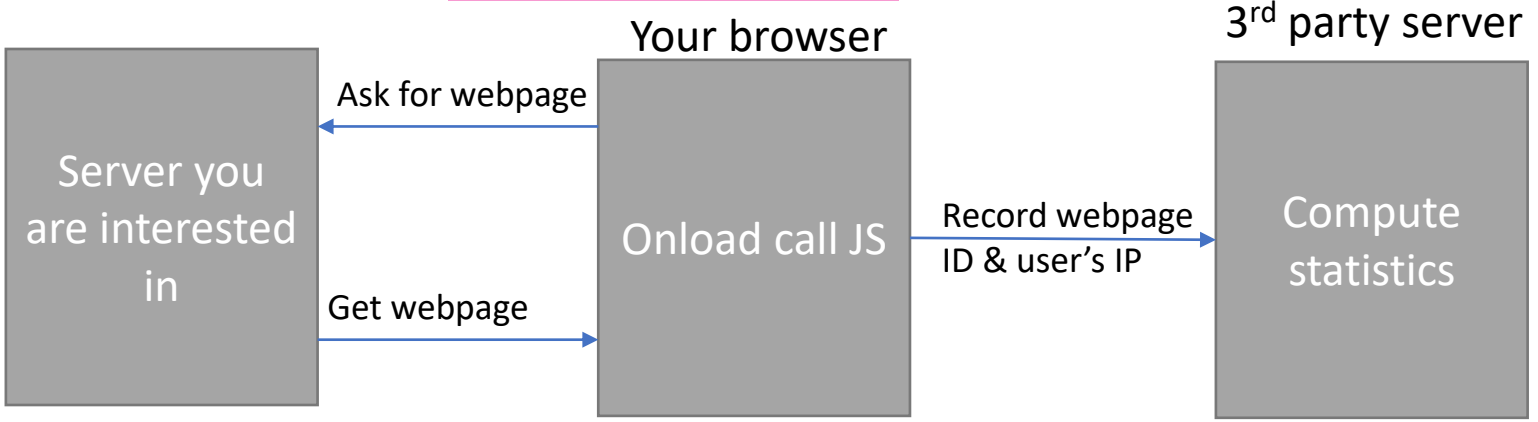
Denial-of-Service Defense

- Generally, hard to defend. Good way to attack.
- DoS attack
  - Since a single application, then a single IP. Find that IP and block it.
  - Normally, bad-guy ssh'd multiple times to make it hard to find their original IP location, but you can find the apps IP location and block.
- DDoS attack
  - This attack uses many IP addresses. Some of these IP addresses can be from valid users. This form of attack often uses thousands of IP addresses.
  - We will still need to block each IP address and someone clean valid user's computers.
    - Or, the target can change their IP address

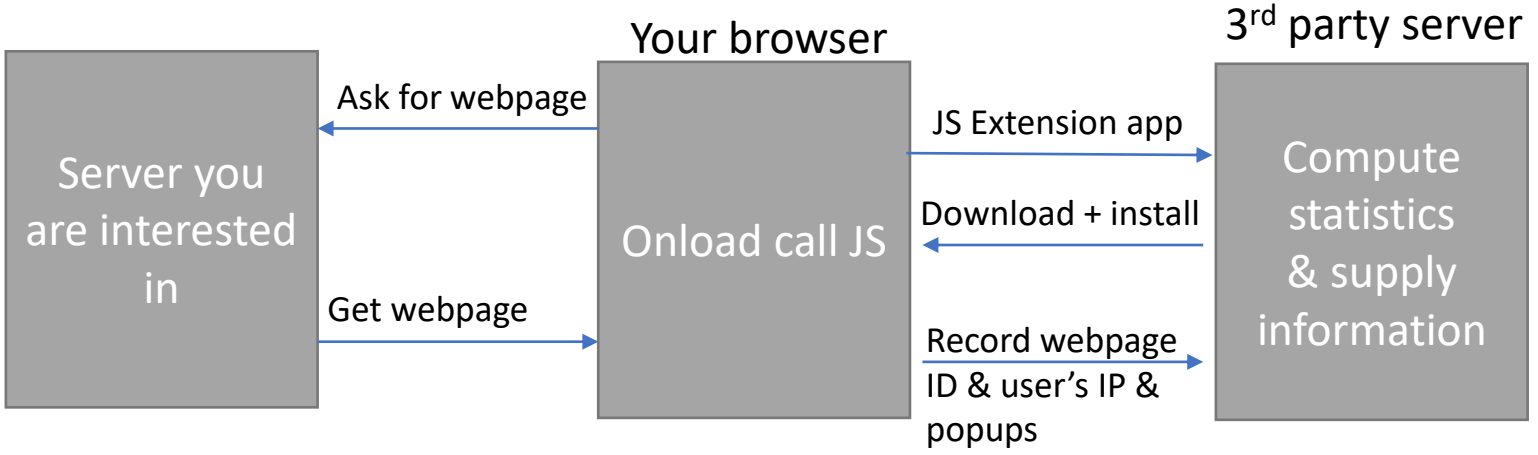
Absence – Adwords - Mmap

Privacy Issues

Type 1: Logging



Type 2: Browser extensions



Question

- How confident are you that Bootstrap's libraries (extensions) are not recording information about your website?
- What about the other tools you are using?
- How confident are you that they are telling the truth when they write on their websites that they are not recording information?
  - What does sharing info with partners mean?

Privacy Issues

- Maybe you want this
  - News media updates
  - Notifications
  - Suggestions
- Maybe you don't want this tracking
  - Turn off JS… (problematic maybe)
  - Go to browser settings and delete extensions
  - Install a virus checker with browser extension features
  - Install an Ad blocker extension… (problematic maybe)

COMP 307
Principles of Web Development

Security Techniques

Security 1

Contents

What are security techniques

- Theoretical ways one can defend the website.
- These are common strategies.
- Basic support exists in some operating systems and servers.
- Some support must be built by the developer or acquired through a library or tool.

A SOCS Example

Server File Security

- The SOCS web security centers around
  - Basic built-in UNIX file security
    - Login username / password
    - chmod
  - Resource identification checks
    - The server asks for you to input your password when you access a new resources – a popup.
    - This is protection against man-in-the-middle
    - This is protection from someone using your account/laptop
  - Encryption using public/private keys
    - SSH uses public/private keys
    - All packet communication is encrypted
    - Casual Wireshark peeking is hard to do
  - Apache path restrictions

A SOCS Example

Server File Security

- The SOCS web security centers around
  - UNIX file security (chmod & login)
  - Resource identification checks
  - Encryption using public/private keys
  - **Apache path restrictions**
    - Website's $HOME = ~$USER/public_html
      - Clients **cannot** cd .. above this point
    - Programs = ~$USER/public_html/cgi-bin
      - Software **can** cd .. above this point, because software's permission is based on the owner's ID who compiled the program which uses $USER path
    - $USER must be chmod +x for public, but –r and –w
    - public_html must be chmod +x and +r for public, but –w
    - cgi-bin must be chmoded +x and +r for public, but –w

Browser Cache

Security 1

Contents

Why is this connected to security?

- Browser memory is a mechanism for your website to remember information locally on the client's computer instead of the server.
- It is connected to security because of what we can store locally. In other words, the information does not need to be transmitted using a packet.
  - No packet, then no bad-guys.
  - Unless bad-guy gets access to your computer… ! (virus)

What is Browser Cache?

- There are two ways of saving data in a browser:
  - Browser cache (web storage)
    - window.localStorage - stores data with no expiration date
    - window.sessionStorage - stores data for one session (data is lost when the browser tab is closed)
  - Cookies

https://www.w3schools.com/html/html5_webstorage.asp

- Web storage (value pairs)
  - The storage limit is at least 5MB and information is never transferred to the server.
  - Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

Example

```
<!DOCTYPE html>
<html>
<body>

<div id="result"></div>

<script>
  // Check browser support
  if (typeof(Storage) !== "undefined") {
    // Store
    localStorage.setItem("lastname", "Smith");

    // Retrieve
    document.getElementById("result").innerHTML =
          localStorage.getItem("lastname");
  } else {
    document.getElementById("result").innerHTML =
          "Sorry, your browser does not support Web Storage...";
  }
</script>

</body>
</html>
```

```
localStorage.removeItem("lastname");
```

Browser Cookies

Security 1

Contents

What are Browser Cookies?

- Cookies are data, stored in small text files (depends on browser), on your computer.
- Cookies were invented to solve the problem "how to remember information about the user":

https://www.w3schools.com/js/js_cookies.asp

  - When a user visits a web page, his/her name can be stored in a cookie. (username, password) (scroll location)
  - Next time the user visits the page, the cookie "remembers" his/her name.
- Property: `document.cookie`

```
document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";
document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";
```

COMP 307
Principles of Web Development

Contents

Attack vectors
Security techniques

Example

```
<!DOCTYPE html>
<html>
<head>
<script>
function setCookie(cname,cvalue,exdays) {
  const d = new Date();
  d.setTime(d.getTime() + (exdays*24*60*60*1000));
  let expires = "expires=" + d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}

function getCookie(cname) {
  let name = cname + "=";
  let decodedCookie = decodeURIComponent(document.cookie);
  let ca = decodedCookie.split(';');
  for(let i = 0; i < ca.length; i++) {
    let c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return "";
}

function checkCookie() {
  let user = getCookie("username");
  if (user != "") {
    alert("Welcome again " + user);
  } else {
     user = prompt("Please enter your name:","");
     if (user != "" && user != null) {
       setCookie("username", user, 30);
     }
  }
}
</script>
</head>

<body onload="checkCookie()"></body>

</html>
```

Security through obfuscation

Security 1

Contents

What is obfuscation?

- Hiding information
  - The less information the bad-guy has the harder it is for them
  - Hide server-side architecture using routing
    - Directory
    - Software, libraries, tools
    - Databases and files
  - Hide browser DOM/Inspect information using obfuscation
    - HTML, CSS and JS code

The less people know about the architecture of your website, the fewer ways to attack your website.

Good or Bad?

- Benefits
  - Harder for bad-guy to figure out how the website is doing the things they see it doing.
- Drawbacks
  - Sometimes requires more processing time on the server and/or browser, impacting the user's experience and the number of connections the server can handle in a short time.

Obfuscation through Routing

- <a href="path">
  - The path reveals a lot about your website:
    - /dir1/dir2/program.extension
    - Public information:
      - Directory structure
      - Name of the file
      - File extension shows programming language or technology (SQL)
- Since an OS has bugs and since Web Servers/technologies have bugs and since programming languages have weaknesses, making that information public gives bad-guys an advantage.

Obfuscation through Routing

- Definition:
  - Replacing public server information with TAGS
  - The TAGS are processed by a program that "routes" the "request" (the TAG is a request) to the actual destination program.
  - This routing takes place on the server-side and is not visible to the user.
    - The user only sees the packet with the webpage returned
    - The user only sees the GET request with the TAGS
  - Server needs to spend extra time parsing the TAG and then calling the correct software

Example

- www.cs.mcgill.ca/~jvybihal

```
// --------- ROUTING WEBPAGE BODY ----------

if (sizeof($_GET)==0 || $_GET["Page"]=="Home") {
        // HOME PAGE
        display("matter/home_matter.txt");
} else if ($_GET["Page"]=="About") {
        // INFO PAGE
        display("matter/about_matter.txt");
} else if ($_GET["Page"]=="Research") {
        // RESEARCH
        display("matter/research_matter.txt");
} else if ($_GET["Page"]=="Publications") {
        // PUBLICATIONS
        display("matter/publications_matter.txt");
} else if ($_GET["Page"]=="Courses") {
        // COURSES
        display("matter/courses_matter.txt");
} else if ($_GET["Page"]=="Hall") {
        // HALL OF FAME
        display("matter/hall.txt");
} else {
        // ERROR PAGE
        echo "404: Invalid Page!";
}
```

tag

routing

Public information

index.php?Page=About

Routing Frameworks

- https://www.slimframework.com/docs/v4/objects/routing.html
  - PHP
- https://www.freecodecamp.org/news/express-explained-with-examples-installation-routing-middleware-and-more/
  - Node JS

What is obfuscating code?

- The problem with HTML, CSS and JS is that it is text and readable using the browser's inspect feature.
- Obfuscating code refers to making the browser code hard to read.

Before obfuscation

```
<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Objects</h2>

<p>Creating an object:</p>

<p id="demo"></p>

<script>

let person = {

  firstName : "John",

  lastName  : "Doe",

  age     : 50,

  eyeColor  : "blue"

};

document.getElementById("demo").innerHTML = person.firstName + " " + person.lastName;

</script>

</body>

</html>
```

Obfuscation – level 1

```
<!DOCTYPE html><html><body><h2>JavaScript Objects</h2><p>Creating an
object:</p><p id="demo"></p><script>let person = {firstName:
"John",lastName:"Doe",age:50,eyeColor:"blue"};document.getElementById("demo").
innerHTML = person.firstName + " " + person.lastName;</script></body></html>
```

Same code as before, but without proper software indenting.

Remove:

- Extra spaces
- Carriage returns
- Add extra random code which is meaningless

Notice that it is hard to read.

This can be created by-hand or using a simple C program.

Obfuscation – level 2

```
<!DOCTYPE html><html><body><h2>JavaScript Objects</h2><p>Creating an object:</p><p
id="p1"></p><script>let p2 = {d1: "John",d2:"Doe",d3:50,d4:
"blue"};document.getElementById("p1").innerHTML = p2.d1 + " " +
p1.d2;</script></body></html>
```

Same code as before but replaced variable names with counted-names.

Remove:

- Extra spaces
- Carriage returns
- Change variable names

Notice that it is hard to read.

This is easiest done using a program that “compiles” the code.

Its most powerful with CSS since it is already hard to read. Also useful in JS.

Obfuscating Architecture

Architecture 1:

- Code is already pre-obfuscated on server.
- Developers have the “original code” in their development directory.
- Code is “compiled” before moved to the public_html or HTDOCS folder. (ex. Bootstrap’s *.min files)

Architecture 2:

- Deployed website has the “original code”
- A compiler function exists in the router program
- Packet payload is compiled before returned to the browser

```
String x = “bob”;
printf(“%s”, compile(x));
```

“compilers”

- https://github.com/javascript-obfuscator/javascript-obfuscator
- https://promon.co/security-news/javascript-obfuscation/

About Encryption

Security 1

Contents

What is encryption?

- Definition
  - Converting plain text to an unreadable form.
- Two forms exist:
  - Data-based encryption
    - When you encrypt the contents of the payload in a packet
    - In this case only the payload is encrypted, the packet header is in plain text
  - Network routing encryption
    - When the entire packet is encrypted
    - However, this is not entirely possible, so the entire packet is often placed in the payload of another packet. "Your" entire packet is encrypted, only certain information is present in the "encasing" packet, like IP addresses (unless you go through a network hiding service).
  - Service provider (indirection)
    - Uses network routing encryption through a server. Hiding the user.
    - User sends request to service. Service sends request to destination. Destination sends answer to service. Service forwards to user.

Types of encryption

- Hashing
  - One directional. Once encrypted it cannot be decrypted.
  - Used for user/service identification.
- 1-key encryption
  - Two-way encryption. Message can be encrypted and decrypted.
  - It uses 1-key, like Caesar's cipher
  - Examples: AES, DES
- 2-key encryption
  - Two-way encryption. Message can be encrypted and decrypted.
  - It uses 2-keys, like Public/Private key ciphers.
  - Examples: TLS, SSL, SSH

Example

- 1-key encryption
  - Alice and Bob share a single key.
    - Alice texts Bob, or sends an email to Bob, or sends a packet to Bob.
    - This is the first time they spoke, so the key is in plain text.
    - After,
      - `Packet(payload) = encrypt(msg, Storage.shared_key)`
      - `Msg = decrypt(packet(payload), Storage.shared_key)`
- 2-key encryption
  - Generate (or get from authority) reciprocal-keys:
    - Alice does this = Key1 and Key2 (keep Key2 private)
    - Bob does this = Key3 and Key4 (keep Key4 private)
  - Alice gives Key1 to the world, Bob does same with Key3
  - Communication method:
    - World to Alice: `packet(payload) = encrypt(msg, Key1)`
    - Alice: `msg = decrypt(packet(payload), Key2)`

Single key encryption

- Simple example: Caesar's cipher
  - Message = “BOB”
  - Key = 2
  - Cipher = Message + Key = “DQD”
  - Message = Cipher – Key = “BOB”
- Strong example: AES
  - Hard to break using brute force
  - Length of key makes it stronger but takes longer to run
    - 128, 192, 256 bits are common key lengths

Two key encryption

- Simple example: modulo cipher
  - Note:
    - 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

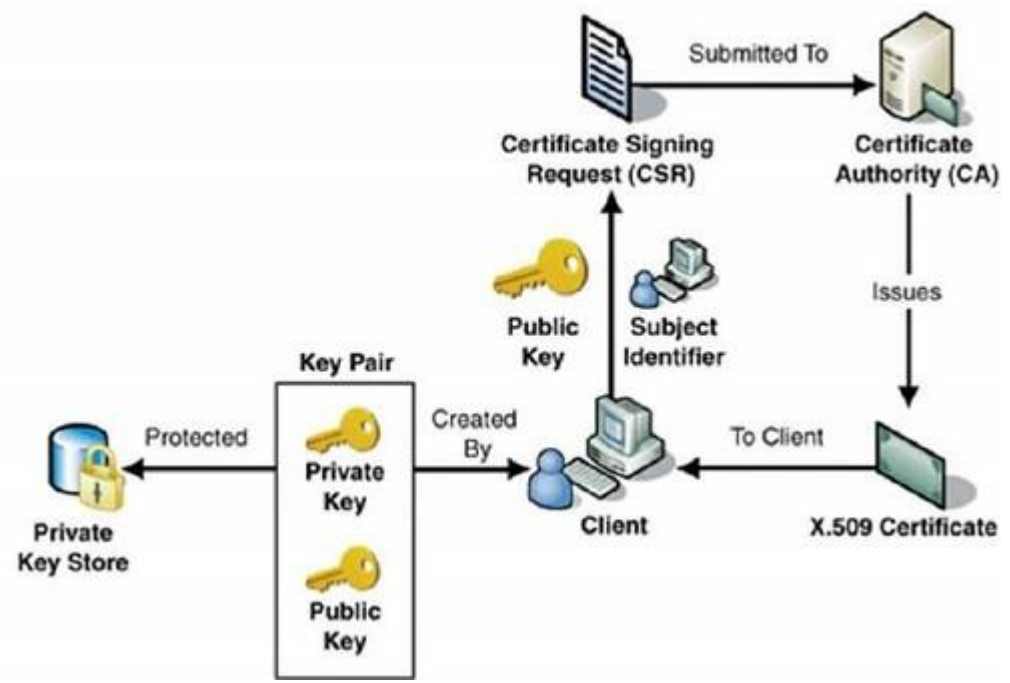
      7 (L → S)

      19 (S → L)
    - Supporting number is 26, in this example.
  - Message = “BOB”
  - Modulo = 100;
  - Key_encrypt = 25
  - Key_dycrypt = 75
  - Message_as_code = ASCII – ‘A’ = 2 15 2
  - Cipher = Message_as_code + Key_encrypt = 27 40 27
  - Message_as_code = (Cipher + Key_dycrypt) % Modulo = 2 15 2
  - Message = ‘A’ + Message_as_code = “BOB”

Technology

- Certificate authority:
  - An official institution that gives out reciprocal keys (public/private keys) via a “certificate”
  - They make sure that no two users have the same certificate
- Certificate
  - File = proof_path + public_key + private_key
    - The proof_path proves that the certificate was created by an authority
    - The reciprocal keys are the public and private keys
- There are public tools for creating your own reciprocal keys – but it is not guaranteed to be unique

Certificate Authority



In the above diagram, the client generated the keys locally and is asking the authority to certify the uniqueness.

Alternatively, the client can ask the authority to generate the key and return that to the client, which my definition would be unique.

x.509 Certificate Example

Version Number	
Serial Number	
Signature Algorithm ID	
Issuer Name	
Validity Period	Not Before
	Not After
Subject Name	
Subject Public Key Info Public Key Algorithm Subject Public Key	
Issuer Unique Identifier (optional)	
Subject Unique Identifier (optional)	
Extensions (optional)	
Certificate Signature Algorithm	
Certificate Signature	

(Important to not confuse the difference between a HASH that uses public/private keys, and message encryption that uses public/private keys. – details next class)

Public key & private key are in the certificate.

Recursive hash ID number used to validate authenticity of certificate.

Using the certificate

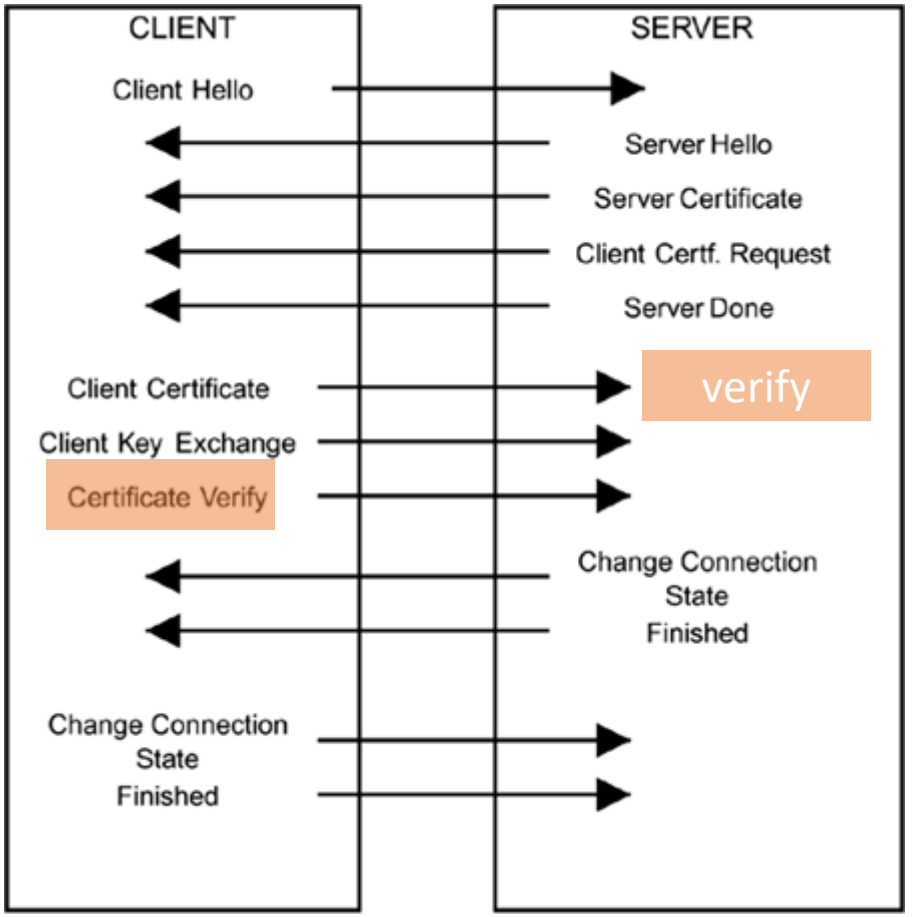
- Want to do two things
  - Encrypt a message
  - Make sure the messages comes from the real user (not man-in-the-middle)
- Steps:
  - Payload = cipher + ID
    - Cipher = encrypt(msg, public_key_bob)
    - ID = encrypt(hash(msg), private_key_Alice)
  - Alice sends payload to Bob, payload = cipher + ID
  - Bob wants to validate:
    - Message = decrypt(cipher, private_key_bob)
    - “digest_1” = decrypt(ID, public_key_Alice)
    - “digest_2” = hash(Message)
    - If the two digests agree, then it came from the correct person

Network Security

- Does network security exist by default?
  - When you use your browser and there is a “lock” symbol, then you know network security exists, otherwise it is public.
  - Without the lock symbol, the website is not using the certificates. Its algorithms do not use the certificates, even when they exist on your computer.

COMP 307
Principles of Web Development

Network Security
(the handshake)



The lock browser symbol is important, because:

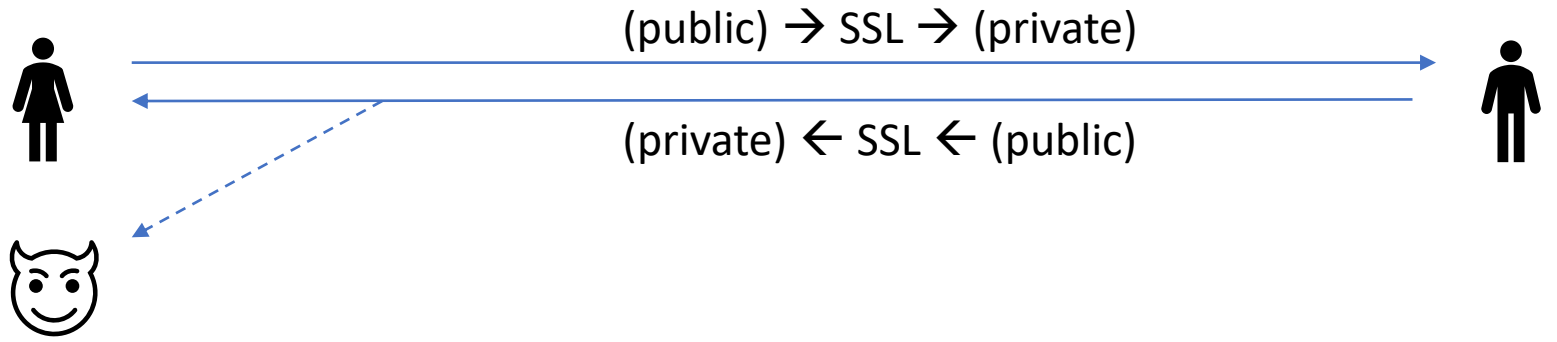
(a) The request for the certificate can be denied because the website is not implementing certificates.
(b) The SSL connection you want between the browser and server can even be denied because the server is not implementing certificates.

As students of McGill you all have your own keys, and with them you see all the information from server.

Contents

Attack vectors
Security techniques

Network Security



If on same network?
(fellow student)

User stores the public key of all the servers and their own private key.

Website or server stores the public key of every user. And its own private key.

Msg = decrypt(payload, public_key)
Payload = encrypt(msg.private_key)

Payload = encrypt(msg, private_key)
Msg = decrypt(payload, public_key)

Prepare for Next Class

- Assignments
  - How is the project going?
- Lab this week
  - Lab F (last lab)
- Do on your own
  - Check to see what information your libraries/extensions are recording.