

---

# **Implementation and Management of Systems Security**

**158.738**

A/Prof. Julian Jang-Jaccard  
Massey University

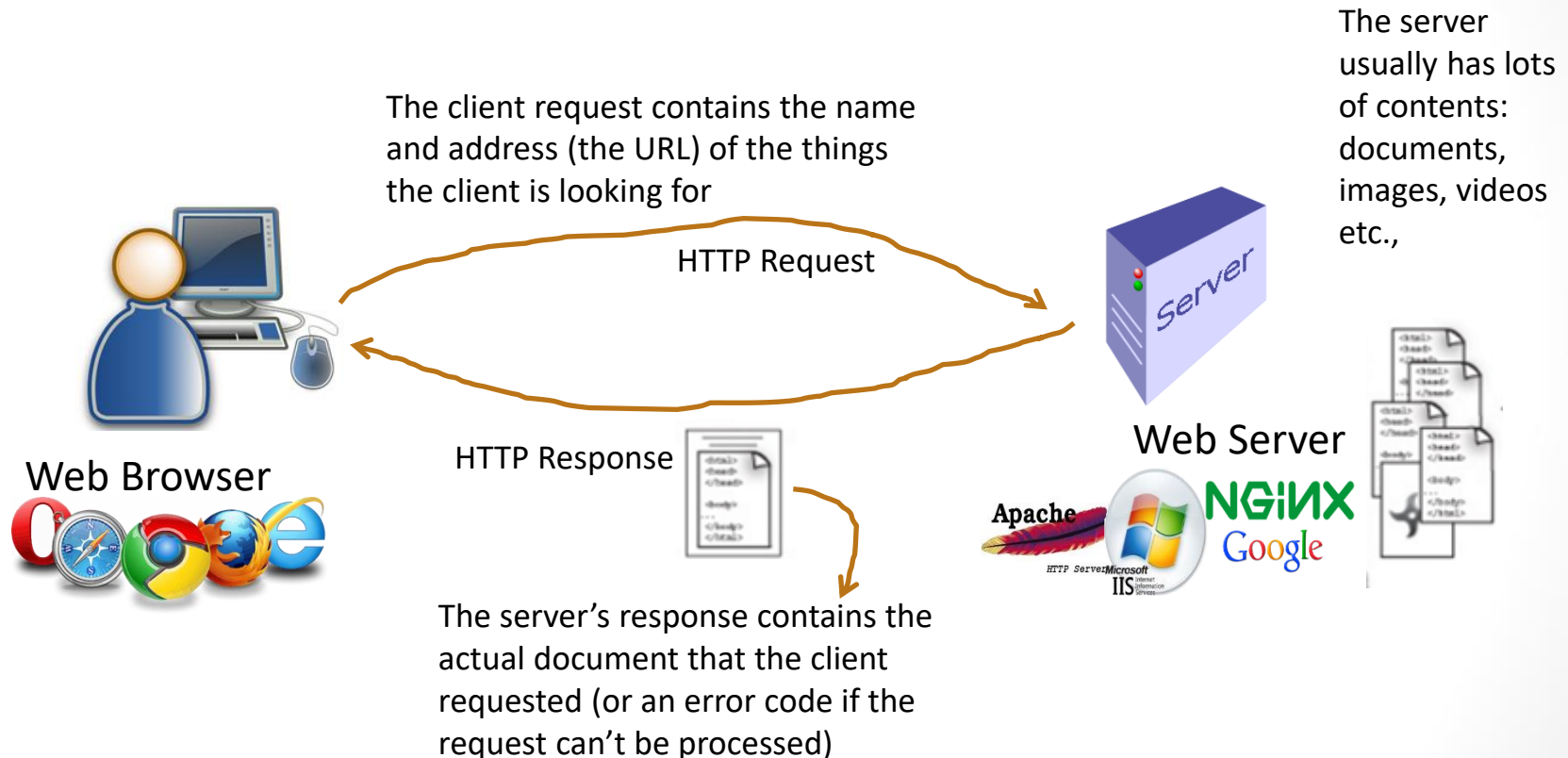
---

---

# **INTERNET SECURITY**

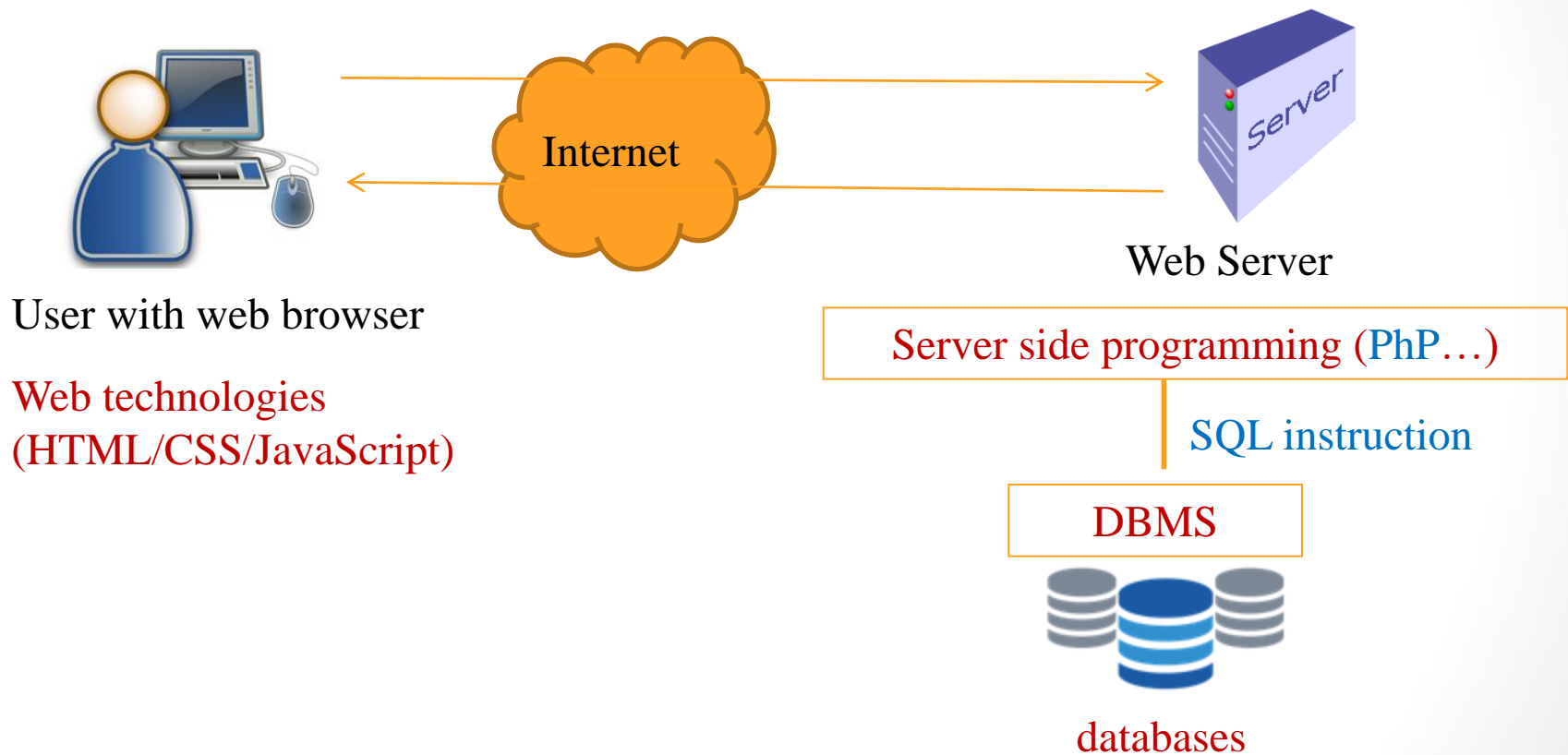
---

# How the Internet Works I



# How the Internet Works II

---



# Dynamic Content

---

- Content (web page) is generated “on-the-fly” and changes regularly
  - Content contains “server-side” code, allows the server to generate unique content when the page is loaded
  - PHP, ASP, JSP or other language is used to pull content from a database
  - Example: upcoming events on a homepage pulling from a calendar and changing each day
-

# Internet Vulnerabilities

---

- Web Browsers
  - Mulvertising
  - Drive-by Downloads
  - Cookies
-

# Browser Vulnerabilities

---

- Scripting Code

- “automatically” download a script or a set of instructions to add more user interactive experience
- JavaScript embedded on HTML documents
- Defense: limit capabilities (e.g., sandboxing, same origin)



# Browser Vulnerabilities

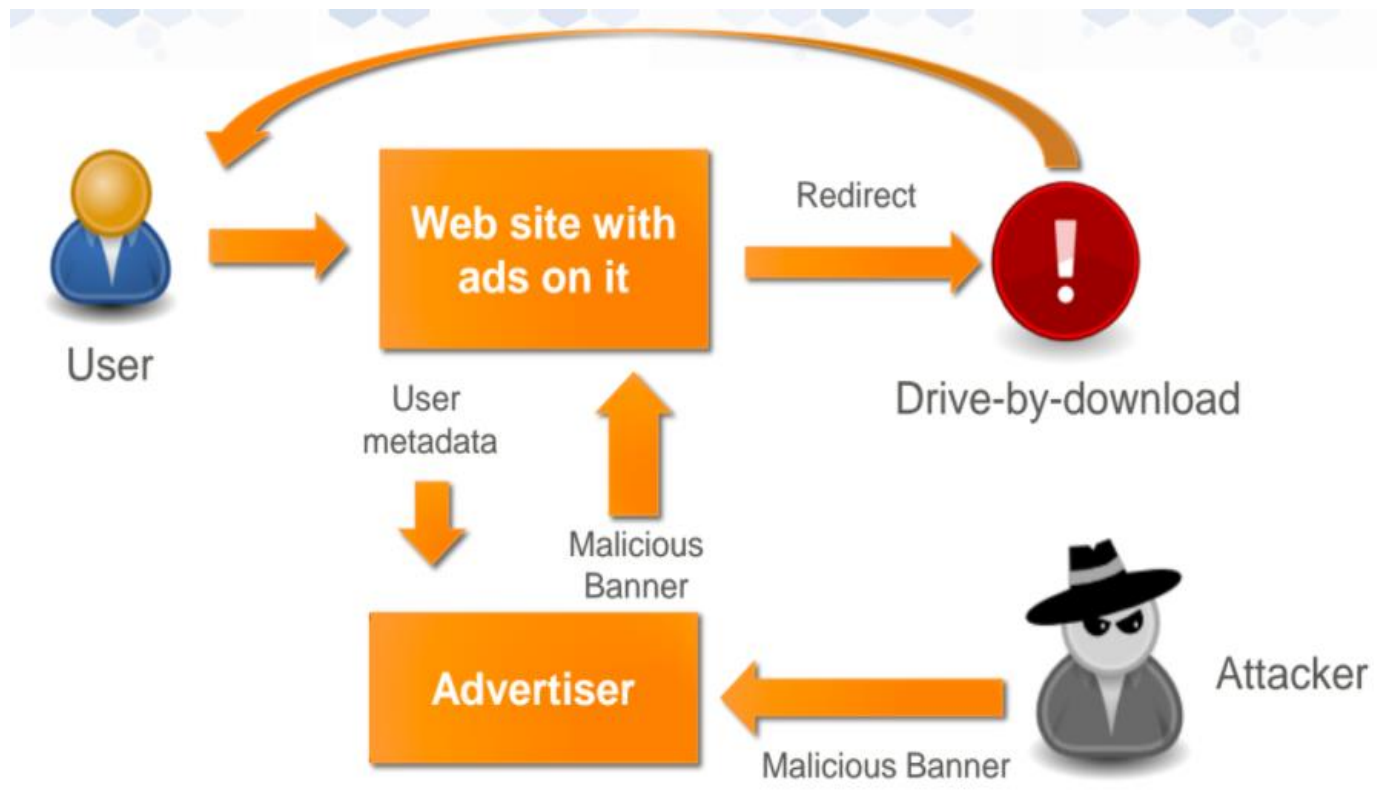
---

Name	Description	Location	Browser support	Examples
Extension	Written in JavaScript and has wider access to privileges	Part of web browser	Only works with a specific browser	Download selective links on webpage, display specific fonts
Add-ons	Adds functionality to browser itself	Part of web browser	Only works with a specific browser	Dictionary and language packs
Plug-ins	Links to external program	Outside of web browser	Compatible with many different browsers	Audio, video, PDF file display



# Mulvertising

- Infect a mainstream website through third-party advertising networks



# Drive-by Downloads

---

- Infect the website directly just from view the website
  - Attackers implant malicious code in the web server
- Websites with popular content
  - Games: 60% of websites contain executable content, one-third contain at least one malicious executable
  - Celebrities, adult content, everything except news
- Many infectious sites exist only for a short time, behave non-deterministically, change often



# Cookies

---

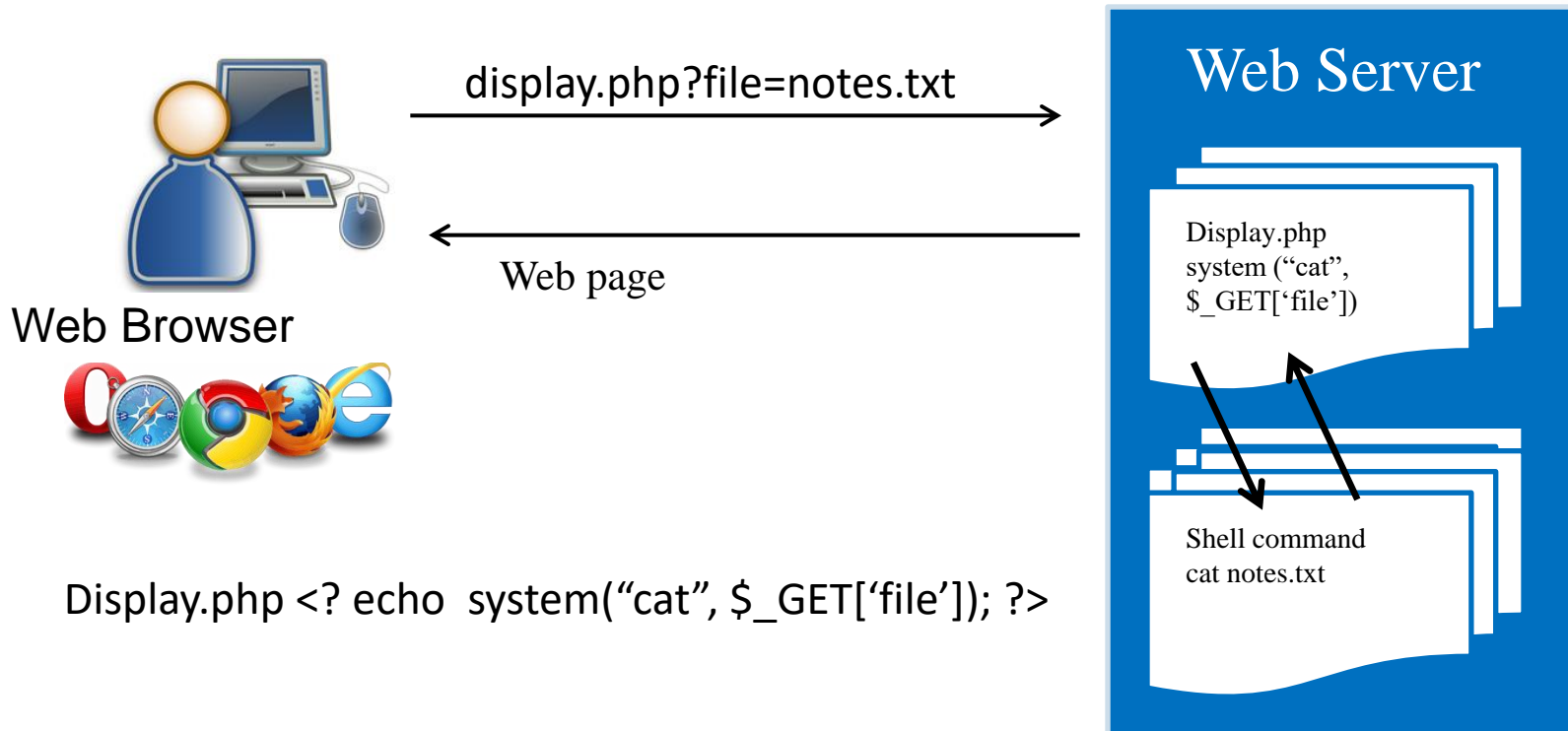
- HTML does not have a mechanism to track users if they have previously visited certain websites.
  - The web server stores user-specific information through a cookie
  - A cookie can contain a variety of information
    - User's preferences when visiting a website
    - Personally identifiable information (name, email address, work address, etc.,)
-

# Web Attack Techniques

---

- Command Injection
  - SQL Injection
  - Cross-site Scripting (XSS)
-

# Command Injection



# Command Injection

---

- Which one of the following URIs is an attack URI?
  - a. `http://www.example.net/display.php?get=rm`
  - b. `http://www.example.net/display.php?file=rm -rf /;`
  - c. `http://www.example.net/display.php?file=notes.txt; rm -rf /;`
  - d. `http://www.example.net/display.php?file=`

# Command Injection

---

- Which one of the following URIs is an attack URI?
  - a. `http://www.example.net/display.php?get=rm`
  - b. `http://www.example.net/display.php?file=rm -rf /;`
  - c. `http://www.example.net/display.php?file=notes.txt; rm -rf /;`
  - d. `http://www.example.net/display.php?file=`

# SQL Injection

---

- SQL: A query language for database
    - E.g., SELECT, INSERT, UPDATE, DELETE etc.,
  - More info
    - E.g., <http://en.wikipedia.org/wiki/SQL>
  - One of the most exploited vulnerabilities on the web. Cause of massive data theft
    - 24% of all data stolen in 2010
    - 89% of all data stolen in 2009
  - Like command injection, caused when attacker controlled data interpreted as a (SQL) command
-



# SQL Injection

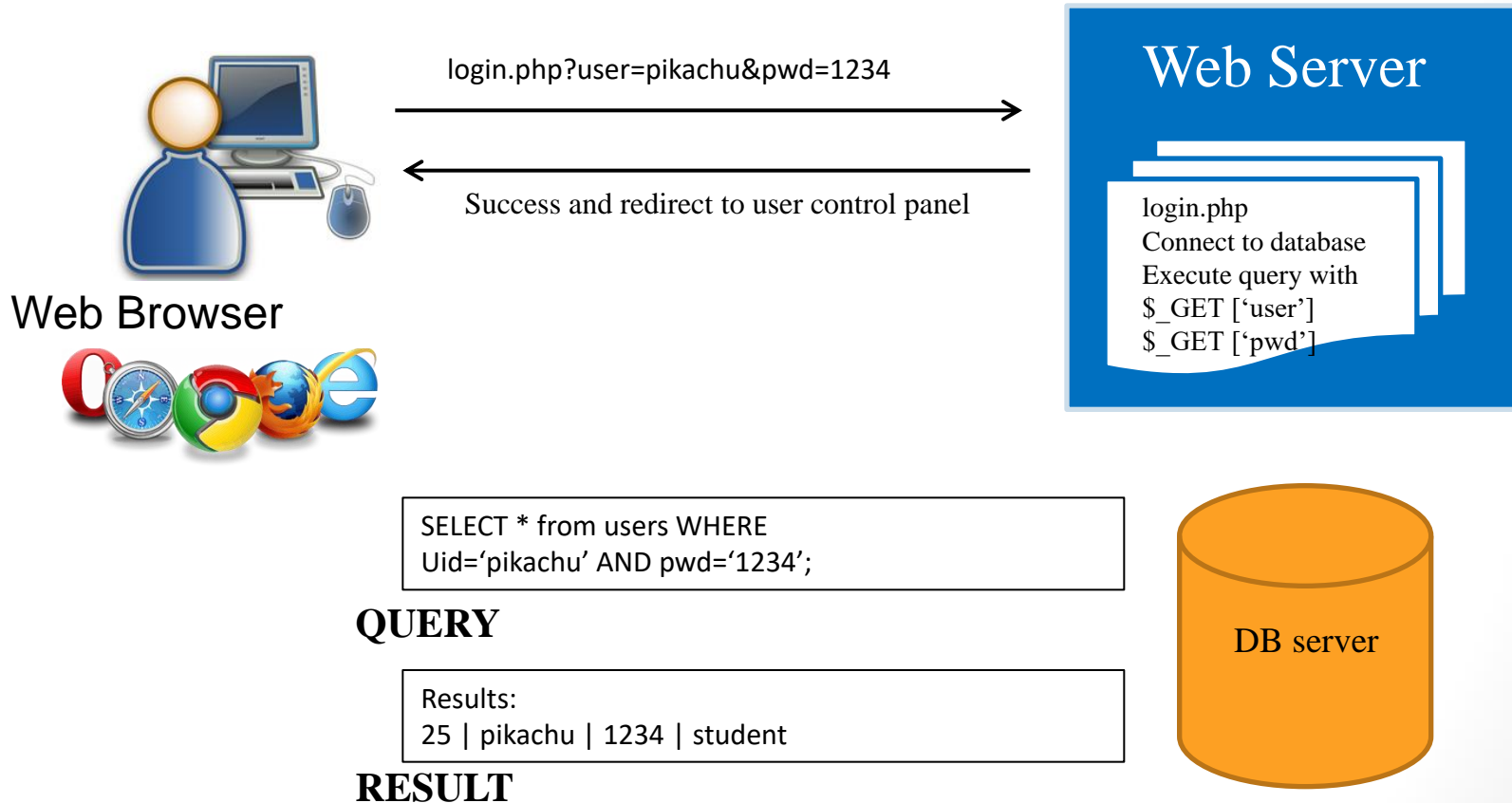
---

- Consider a web page that logs in a user by seeing if a user exists with the given username and password.

```
$result = pg_query("SELECT * from users WHERE  
                        uid = '$_GET['user']' AND  
                        pwd = '$_GET['pwd']'");  
  
);  
  
if (pg_query_num($result) > 0) {  
    echo "Success";  
    user_control_panel_redirect();  
}
```

- It sees if results exist and if so logs the user in and redirects them to their user control panel.
-

# SQL Injection



# SQL Injection

---

- Q: Which one of the following queries will log you in as admin?
- Hints: The SQL language supports comments via '--' characters
  - a. `http://www.example.net/login.php?user=admin&pwd='`
  - b. `http://www.example.net/login.php?user=admin--&pwd=foo`
  - c. `http://www.example.net/login.php?user=admin'--&pwd=f`

# SQL Injection

---

- Q: Which one of the following queries will log you in as admin?
- Hints: The SQL language supports comments via '--' characters
  - a. `http://www.example.net/login.php?user=admin&pwd='`
  - b. `http://www.example.net/login.php?user=admin--&pwd=foo`
  - c. `http://www.example.net/login.php?user=admin'--&pwd=f`

```
pg_query("SELECT * from users WHERE  
uid = 'admin'--' AND pwd = 'f';");
```

```
pg_query("SELECT * from users WHERE  
uid = 'admin';");
```



# SQL Injection

---

- Q: Under the same premise as before, which URI can delete the users table in the database?
  - a. `www.example.net/login.php?user=;DROP TABLE users;--`
  - b. `www.example.net/login.php?user=admin'; DROP TABLE users;--'`  
`AND pwd='f';`
  - c. `www.example.net/login.php?user=admin; DROP TABLE users; --`  
`AND pwd=f`
  - d. It is not possible. (None of the above)

# SQL Injection

---

- Q: Under the same premise as before, which URI can delete the users table in the database?
  - a. `www.example.net/login.php?user=;DROP TABLE users;--`
  - b. `www.example.net/login.php?user=admin'; DROP TABLE users;--' AND pwd='f';`
  - c. `www.example.net/login.php?user=admin; DROP TABLE users; -- AND pwd=f`
  - d. It is not possible. (None of the above)

```
pg_query("SELECT * from users WHERE  
uid = 'admin'; DROP TABLE users;--' AND  
pwd = 'f';");
```

```
pg_query("SELECT * from users WHERE uid = 'admin';  
DROP TABLE users;");
```



# Input Validation

- Whitelisting: Only allow known-good values

```
<?  
if(!preg_match("/^[a-z0-9A-Z.]*$/", $_GET['file'])) {  
    echo "The file should be alphanumeric."  
    return;  
}  
echo system("cat ".$_GET['file']);  
?>
```

GETINPUT	PASSES?
notes.txt	Yes
notes.txt; rm -rf /;	No
security notes.txt	No

# Input Escaping

---

<?

```
echo system("cat ".escapeshellarg($_GET['file']));
```

?>

- *escapeshellarg()* adds single quotes around a string and quotes/escapes any existing single quotes allowing you to pass a string directly to a shell function and having it be treated as a single safe argument :--<http://www.php.net/manual/en/function.escapeshellarg.php>

GETINPUT	Command Executed
notes.txt	cat 'notes.txt'
notes.txt; rm -rf /;	cat 'notes.txt rm -rf /;'
mary o'donnel	cat 'mary o\'\'donnel'



# SQL Injection

---

- Given that our web application employs the input validation mechanism for usernames, which of the following URIs would still allow you to login as admin?

```
pg_query("SELECT * from users WHERE  
uid = '$_GET['user']' AND  
pwd = '$_GET['pwd']'");
```

- <http://www.example.net/login.php?user=admin&pwd=admin>
- <http://www.example.net/login.php?user=admin&pwd=' OR 1=1;--'>
- <http://www.example.net/login.php?user=admin'--&pwd=f>
- <http://www.example.net/login.php?user=admin&pwd='-->

# SQL Injection

- Given that our web application employs the input validation which of the following can be used to login as admin?

```
pg_query("SELECT * from users WHERE uid = 'admin' AND  
pwd = ' OR 1 = 1;--'");
```

1=1 is true everywhere. This returns all the rows in the table, and thus number of results is greater than zero.

```
pg_query("SELECT  
uid = '$_GET['user']'" AND  
pwd = '$_GET['pwd']'");
```

- <http://www.example.net/login.php?user=admin&pwd=admin>
- <http://www.example.net/login.php?user=admin&pwd=' OR 1=1;--'>
- <http://www.example.net/login.php?user=admin'--&pwd=f>
- <http://www.example.net/login.php?user=admin&pwd='-->

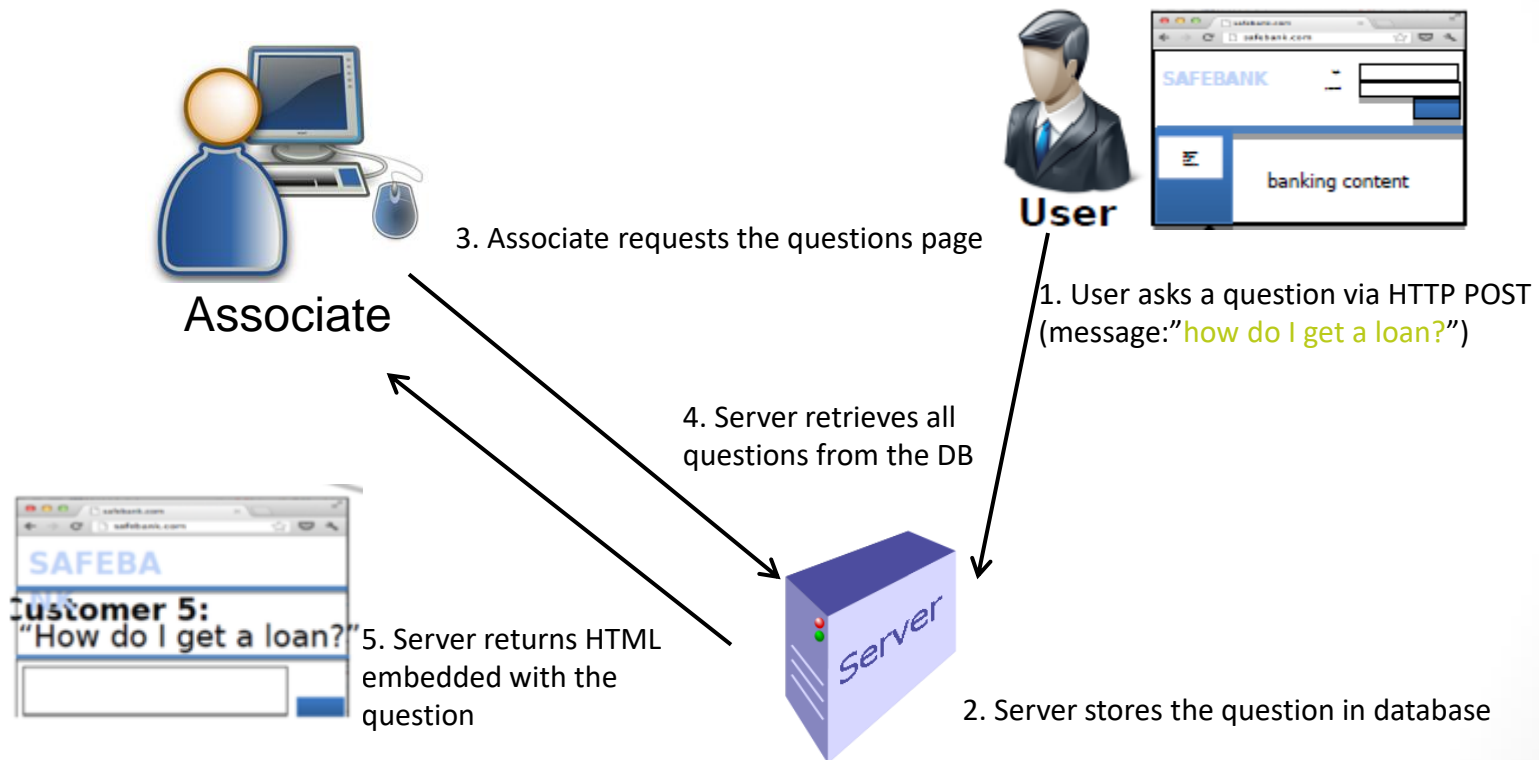
# Cross Site Scripting

---

- Vulnerability in web application that enables attackers to inject malicious scripts into web pages viewed by other users.
- Types
  - Type 2: The attack vector is stored at the server
  - Type 1: Reflected: The vulnerability is in the server-side
  - Type 0: DOM based: The vulnerability is in the client side only

# XSS: Type2: setting the scene

Consider a form on a website that allows a user to chat with a customer service associate



```
PHP code: <? echo "<div class='question'>$question</div>";?>
HTML code: <div class="question">How do I get a loan?</div>
```

# Cross Site Scripting

---

- Look at the following code fragments. Which one of these could possibly be a command that could be used to perform a XSS injection?
  - a. `' ; system('rm -rf /');`
  - b. `rm -rf /`
  - c. `DROP TABLE QUESTIONS;`
  - d. `<script>doEvil()</script>`

# Cross Site Scripting

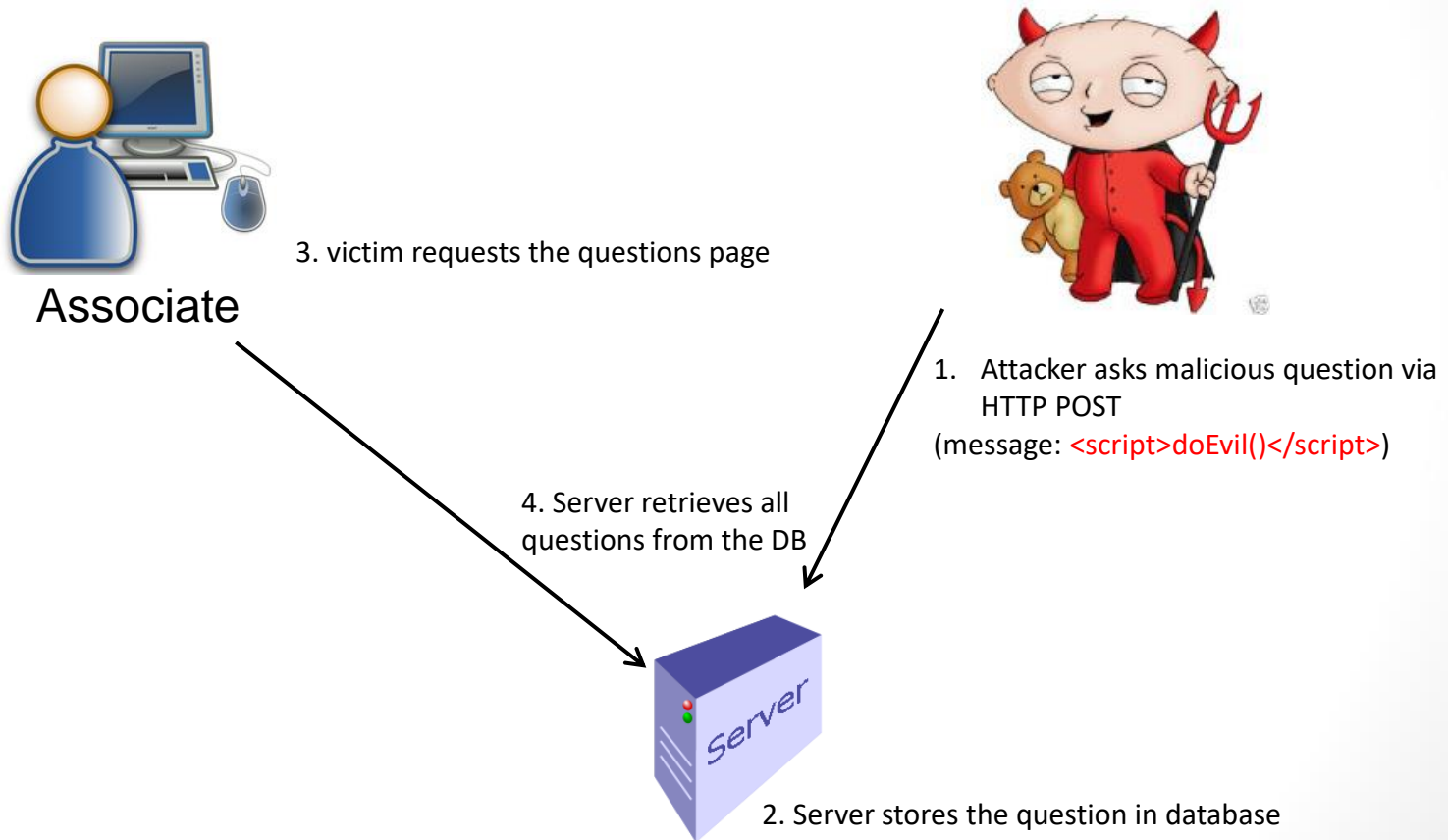
---

- Look at the following code fragments. Which one of these could possibly be a command that could be used to perform a XSS injection?
  - a. `' ; system('rm -rf /');`
  - b. `rm -rf /`
  - c. `DROP TABLE QUESTIONS;`
  - d. `<script>doEvil()</script>`

```
<html><body>
...
<div class='question'>
<script>doEvil()</script>
</div>
...
</body></html>
```

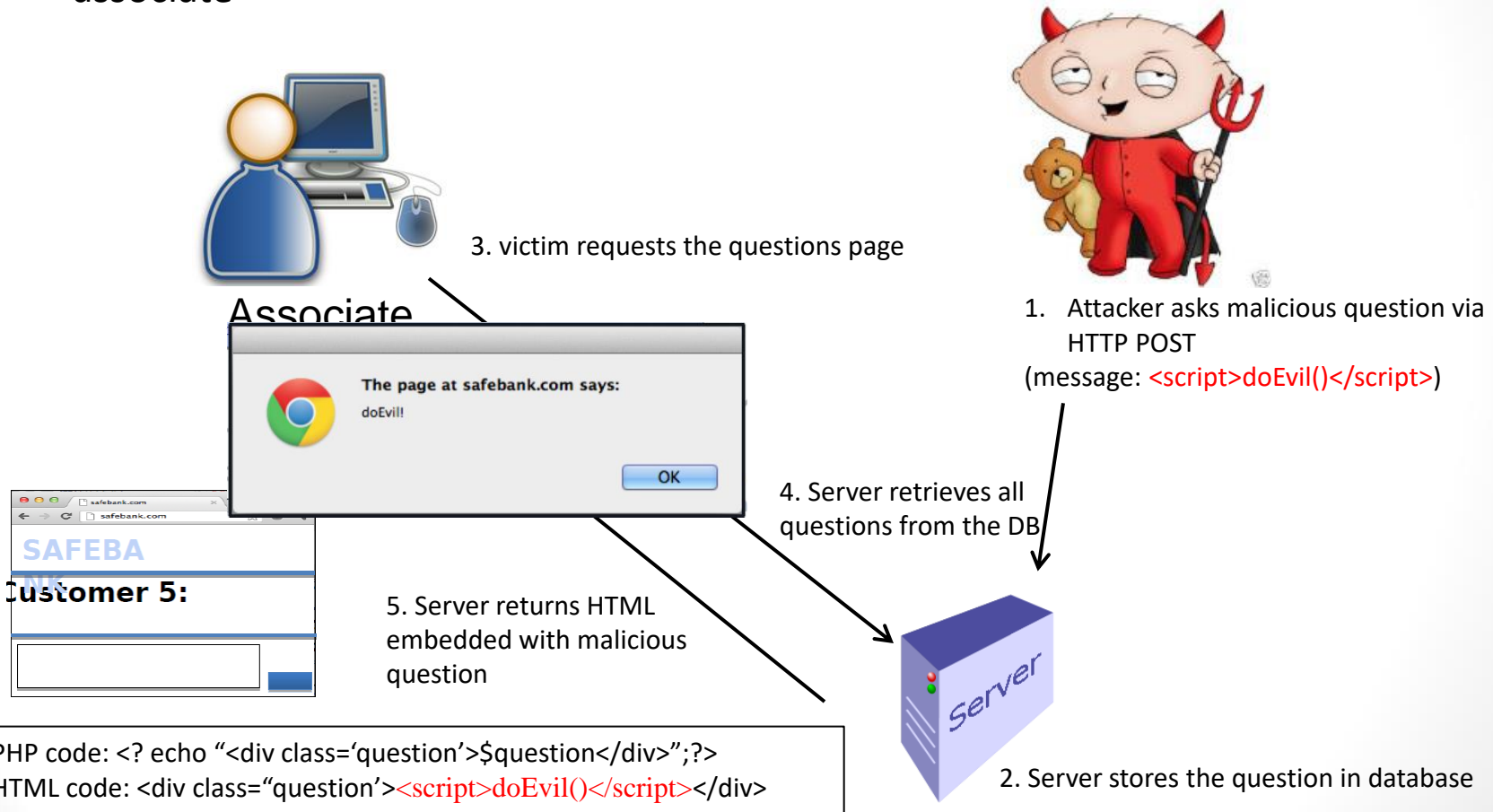
# XSS: Type2: attack (1)

Consider a form on a website that allows a user to chat with a customer service associate



# XSS: Type2: attack (2)

Consider a form on a website that allows a user to chat with a customer service associate





---

---

**END**

---