

Security Checkpoints

Table of Contents

Basic Security	2
XSS Vulnerability	3
Iframe/ HTML Injection	4
For IFrame Injection/ clickjacking:.....	4
Malicious File Upload.....	5
Session Hijack.....	6
Account Lockout Issue.....	8
Concurrent Login	8
OTP Bombarding/ Form Bombarding	9
AES (Advanced Encryption Standard) Encryption	11
CSRF (Cross Site Request Forgery) Implementation.....	13
Outdated JQuery	14
Outdated Bootstrap	14
Improper Input Validation	15
OTP Bypass.....	16
SQL Injection.....	17
References.....	17

Basic Security

Solution: Use Spring security built-in functions/ methods

Spring Security is a framework which provides various security features like: authentication, authorization to create secure Java Enterprise Applications.

```
protected void configure(HttpSecurity http) throws Exception {
    http.addFilterBefore(new CaptchaAuthenticationFilter("/login", "/login"), UsernamePasswordAuthenticationFilter.class);
    http.addFilterAfter(new SessionCookieFilter(), UsernamePasswordAuthenticationFilter.class);
    http.authorizeRequests()
        .antMatchers("/UrbanGIS", "/SaveAgriRegister/**", "/SaveUrbanRegister", "/distfromcommissionary", "/getDistCodeByDistName", "/ulbtypefromcommi",
            "/tablepagination", "/ScienceTechnology", "/RoadsBuilding", "/Revenue", "/RevenueP", "/PortTransport", "/Labour", "/Industries", "/Navlak",
            "/Home", "/Health", "/Ayush", "/Ayurveda", "/Homeopathic", "/Yunani", "/Energy", "/Education", "/Climate", "/RandB", "/Forest", "/Rural", "/F",
            "/SocialEmpowerment", "/Sports", "/Tribal", "/UrbanAdmin", "/Urban", "/Water", "/WomenChild", "/Agriculture", "/Tourism", "/FoodSafety", "/",
            "/GUDM", "/ISR", "/Road", "/Warehousing", "/GEDA", "/Indextb", "/Pavitrayatradham", "/GRIDE", "/TCGL", "/GUJSAIL", "/Mining", "/Medical", "/I",
            "/Settlement_Commissioner", "/GETCO", "/GIFTPower", "/GreenEnergyCorridor", "/GSPC", "/GroundWater", "/MinorIrrigation", "/Stamp", "/MSME",
            "/Petrochemicals", "/REMC", "/UjjwalDISCOM", "/FoodCorporation", "/Dholera", "/DholeraAirport", "/GeologyMines", "/Transport", "/Commerci",
            "/GIDB", "/GIDC", "/GPCPIRDA", "/GRIDE", "/GUJSAIL", "/Indextb", "/Mandalbecharaji", "/SeedCorporation", "/Environment",
            "/Pavitrayatradham", "/TCGL", "/DET", "/COT", "/GSRTC", "/GujaratMaritimeBoard", "/GSRDC", "/GEC", "/GPCL", "/Home", "/FoodCivil", "/Seconda",
            "/RandB", "/LivelihoodMission", "/Panchayat", "/RuralDevelopment", "/ISR", "/ArchaeologySurveyofGujarat", "/Culture",
            "/GujaratSportsAuthority", "/GUDM", "/GujaratMetroRailCorporation", "/Kalpasar", "/SSNNL", "/WaterResources", "/WaterSupply", "/Cooperat",
            "/ICDS", "/AgroIndustries", "/AnimalHusbandary", "/DirectorofAgriculture", "/DirectorofSugar", "/Horticulture", "/GWSSB", "/Bandobast",
            "/login**", "/captcha", "/register", "/webjars/**", "/*.js", "/*.css", "/*.jpg", "/*.png", "/*.jpeg")
        .permitAll().anyRequest().authenticated().and().formLogin().loginPage("/login")
        .usernameParameter("username").successHandler(this).failureHandler(this).permitAll().and().logout()
        .logoutSuccessUrl("/logout1").permitAll().and().exceptionHandling().accessDeniedPage("/logout1").and()
}
```

XSS Vulnerability

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.

XSS Vulnerability

- Reflected XSS, where the malicious script comes from the current HTTP request.
- Stored XSS, where the malicious script comes from the website's database.
- DOM-based XSS, where the vulnerability exists in client-side code rather than server-side code.

07/06/2023

2

Solution:

```
application-... application-... application-... AdminControl... SecurityConf... X CaptchaAuth... »_1
"/Mandalbecharaji", "/SeedCorporation", "/Pavitrayatradham", "/TCGL", "/DET", "/COT", "/GSRTC",
"/GujaratMaritimeBoard", "/GSRDC", "/GEC", "/GPCL", "/RANDB", "/LivelihoodMission",
"/Panchayat", "/RuralDevelopment", "/ISR", "/ArchaeologySurveyofGujarat",
"/CommissionerateofCulture", "/GujaratSportsAuthority", "/GUDM", "/GujaratMetroRailCorporation",
"/Kalpasar", "/SSNNL", "/WaterResources", "/WaterSupply", "/RegistrarofCooperativeSocieties",
"/ICDS", "/AgroIndustries", "/AnimalHusbandary", "/CommissionerofFisheries", "/ApexAuthoritySIR",
"/DirectorofAgriculture", "/DirectorofSugar", "/Horticulture", "/GWSSB", "/login**", "/captcha",
"/register", "/SIR", "/webjars/**", "/*/*.*.js", "/*/*.*.*", "/*/*.*.css", "/*/*.*.jpg", "/*/*.*.png", "/nocfishf",
"/*/*.*.jpeg", "/GETCOLogin", "/Applicant", "/getcoformlogin", "/landform", "/landformpage", "/agriculturereform",
"/ifp", "/ViewApplication", "/signup", "/nocUpdate/**", "/nocfishingformreport/**", "/asiform", "/asinoc", "/asif",
"/gedaregisterdata", "/GEDALogin", "/saveregistrationdetails", "/registerwater/1",
"/testqrcodeata/**", "/zingaqrcodeata/**", "/saltpanqrcodeata/**", "/QRcodeValidationwater/**", "/QRcodeVal",
"/testqrcodeata_admin/**", "/View_Certificate_eSarkar/**",
"/registerwater/**", "/registerkalpsar/1", "/registerkalpsar/**", "/forgotpassword/**", "/forgetVerify/**", "/sav",
"/saveregistration", "/GEDAfile", "/downloadqrcodefile/**", "/updatePassword", "/revenueloginvalidation", "/cal",
.permitAll().anyRequest().authenticated().and().formLogin().loginPage("/login")
.usernameParameter("username").successHandler(this).failureHandler(this).permitAll().and().logout()
.logoutSuccessUrl("/logout1").permitAll().and().exceptionHandling().accessDeniedPage("/logout1").and()
.headers().xssProtection().block(false).and().contentTypeOptions().and().cacheControl().and()
.contentSecurityPolicy("form-action 'self' https://iora.gujarat.gov.in https://staging2.gujarat.gov.in:9080/ro",
.frameOptions().sameOrigin().and()
// .csrf().disable()
.sessionManagement().maximumSessions(1)
.expiredUrl("/login?invalid-session=true").maxSessionsPreventsLogin(true);
```

Iframe/ HTML Injection

An iFrame injection XSS is a common cross-site scripting attack that combines malicious JavaScript with an iframe that loads a legitimate page in an effort to steal data from an unsuspecting user. This attack is usually only successful when combined with social engineering

- Iframe Injection
- HTML injection
- Malicious File Upload
- Session Hijacking

07/06/2023

3

Solution:

```
permitAll().anyRequest().authenticated().and().formLogin().loginPage("/login")
usernameParameter("username").successHandler(this).failureHandler(this).permitAll().and().logout
logoutSuccessUrl("/logout1").permitAll().and().exceptionHandling().accessDeniedPage("/logout1")
headers().xssProtection().block(false).and().contentTypeOptions().and().cacheControl().and()
ContentSecurityPolicy("form-action 'self' https://iora.gujarat.gov.in https://staging2.gujarat.
frameOptions().sameOrigin().and()
.contentSecurityPolicy("form-action 'self' https://iora.gujarat.gov.in http
"form-action 'self' https://iora.gujarat.gov.in https://staging2.gujarat.gov.in:9080/row/gati
080/row/gatishakti/application https://demoegov.gidcgujarat.org:4433/;").and()
:9080/row/gatishakti/application https://demoegov.gidcgujarat.org:4433/;" "script-src 'self'")
Syntax error on token "'script-src 'self'", delete this token
Press F2 for focus
```

For Iframe Injection/ clickjacking:

Clickjacking is an attack that fools users into thinking they are clicking on one thing when they are actually clicking on another

```
.frameOptions().sameOrigin().and()
```

'script-src' 'self' will allow scripts only from script-src folder and disallow all external java scripts. Apply the same for style.

Malicious File Upload

File upload vulnerabilities are when a web server allows users to upload files to its file system without sufficiently validating things like their name, type, contents, or size.

Solution:

```
MultipartFile signature_file = req.getFile("signature_file");
if (signature_file.isEmpty()) {
} else {

    List<String> allowedFileTypes = Arrays.asList("pdf");
    try {
        String fileExtension = FilenameUtils.getExtension(signature_file.getOriginalFilename());
        if (!allowedFileTypes.contains(fileExtension.toLowerCase())) {
            return "error: You have uploaded unsupported File type.\nPlease upload only pdf files";
        }
        long timeadd = System.currentTimeMillis();
        signature_file.transferTo(Paths.get(myDir.getAbsolutePath(),
            timeadd + "_" + signature_file.getOriginalFilename()));
        savebuform.setSignature(timeadd + "_" + signature_file.getOriginalFilename());
    } catch (Exception e) {
        return "Error: " + e.getMessage();
    }
}

MultipartFile scrutinyfees_file = req.getFile("scrutinyfees_file");
```

Note:

Apply Client side basic validation like, checking file type extension, file size etc.

At server side, also check the signature of well known files like doc/ docs, xls, pdf etc.

Session Hijack

The Session Hijacking attack consists of the exploitation of the web session control mechanism, which is normally managed for a session token. The Session Hijacking attack compromises the session token by stealing or predicting a valid session token to gain unauthorized access to the Web Server.



Suppose, you have logged-in the portal.

JSESSIONID is generated.

You can debug the page and find it.

COPY this JSESSIONID.

Open the same page in other browser.

PASTE this JSESSIONID.

Now, you can access the portal without login.

That is Session hijack.

Solution:

Use SessionCookieFilter.

Assign SAME_SITE_ATTRIBUTE_VALUES to HTTPOnly; Secure; SameSite=Strict;

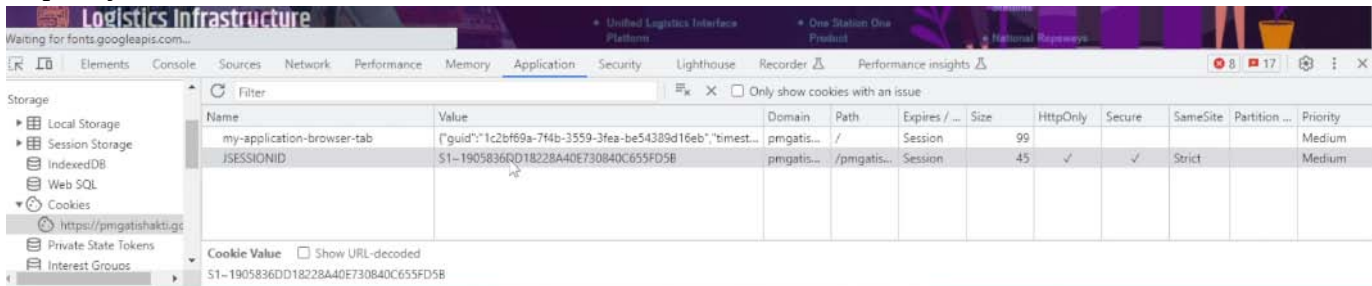
```
protected void configure(HttpSecurity http) throws Exception {
    http.addFilterBefore(new CaptchaAuthenticationFilter("/login", "/login"), UsernamePasswordAuthenticationFilter.class);
    http.addFilterAfter(new SessionCookieFilter(), UsernamePasswordAuthenticationFilter.class);
    http.authorizeRequests()
        .antMatchers("/UrbanGIS", "/SaveAgriRegister/**", "/SaveUrbanRegister", "/distfromcommissionary", "/getDistCodeByDistName", "/ulbtypefromcommi...

@Configuration
public class SessionCookieFilter extends GenericFilterBean {

    private final List<String> PATHS_TO_IGNORE_SETTING_SAMESITE = Arrays.asList("resources");
    private final String SESSION_PATH_ATTRIBUTE = ";Path=";
    private final String ROOT_CONTEXT = "/";
    private final String SAME_SITE_ATTRIBUTE_VALUES = ";HttpOnly;Secure;SameSite=Strict";

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse resp = (HttpServletResponse) response;
        String requestUrl = req.getRequestURL().toString();
        boolean isResourceRequest = requestUrl != null ? StringUtils.isNotBlank(PATHS_TO_IGNORE_SETTING_SAMESITE.stream().filter(s -> requestUrl.
        if (!isResourceRequest) {
            Cookie[] cookies = ((HttpServletRequest) request).getCookies();
            if (cookies != null && cookies.length > 0) {
                List<Cookie> cookieList = Arrays.asList(cookies);
                for (Cookie cookie : cookieList) {
                    String contextPath = request.getServletContext() != null && StringUtils.isNotBlank(request.getServletContext().getContextPath()
                    resp.setHeader(HttpHeaders.SET_COOKIE, cookie.getName() + "=" + cookie.getValue() + SESSION_PATH_ATTRIBUTE + contextPath + SAME_
                    // System.out.println(resp.getHeader(SAME_SITE_ATTRIBUTE_VALUES));
                }
            }
        }
        chain.doFilter(request, response);
    }
}
```

After applying this class, once you debug the Application cookies, you will observe that HttpOnly and Secure will be **tick marked** and SameSite is attributed as **Strict**.



Account Lockout Issue

Account lockout allows you to lock an account after repeated failed login attempts. The Account lockout duration policy setting determines the number of minutes that a locked-out account remains locked out before automatically becoming unlocked. Session automatically times out if user remains idle and exceeds timeout limit.

Solution: Session gets expired after 300 seconds of idle time.

Application.property

```
server.servlet.session.timeout=300s  
server.servlet.session.cookie.secure=true
```

Concurrent Login

Concurrent Users means the number of Authorized Users that are simultaneously logged in to the Software at any single point in time.

Solution:

SecurityConfig.java

```
.sessionManagement().maximumSessions(1)  
.expiredUrl("/login?invalid-session=true").maxSessionsPreventsLogin(true);
```


OTP Bombarding/ Form Bombarding

OTP Bombarding is an attack where a large number of OTP are clicked/ sent in a very short period of time by a single user to overload SMS API gateway and hence disrupt the normal working of a system.



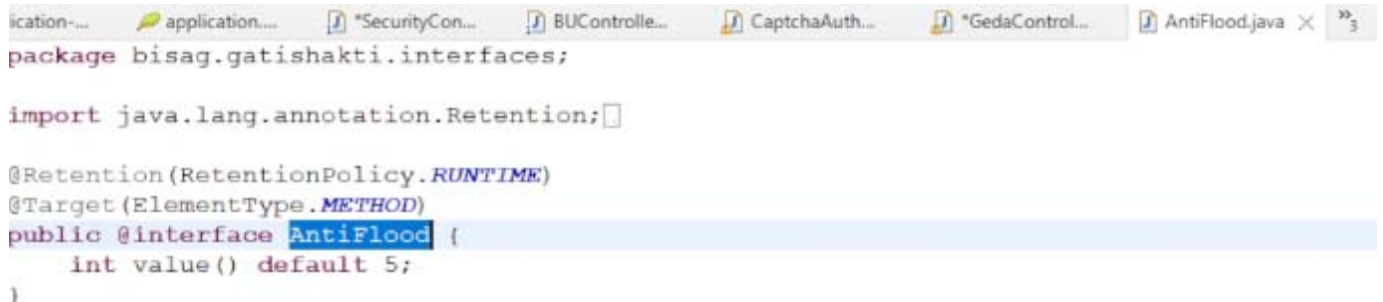
```
    }

    @AntiFlood(value=2000)

    @RequestMapping(value = "/prvwpermissionletterstu", method = RequestMethod.POST)
    public String prvwpermissionletterstu(Model model, Principal principal, HttpServletRequest request,
        Authentication auth) throws Exception, IOException {

        String[] aaa = request.getParameterValues("1a");
        String[] copyto = request.getParameterValues("1c");
        String poffname = request.getParameter("p_offname");
        String pdate = request.getParameter("p_date");
        String pdate1 = request.getParameter("p_date1");
        String encid = request.getParameter("makeid_enc");
        model.addAttribute("encid", encid);
    }
```

It will stop any request for 2000 milliseconds



```
package bisag.gatishakti.interfaces;

import java.lang.annotation.Retention;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface AntiFlood {
    int value() default 5;
}
```

You can also apply custom logic to generate OTP for limited times.

```

String page = obj.getString("pagetitle");

if(users !=null) {
if(bcript.matches(obj.getString("password"), users.getPassword())){
if(users!=null && users.getStateName() !=null?users.getStateName().equals("Gujarat"):
    System.out.println(users.getUsername() + " Database otp is---> " + users.getOtp());
    System.out.println(users.getUsername() + " new otp is---> " + otp);
    System.err.println(users.getTargeturl() + "/////" + page);
if(users.getTargeturl().contains(page.substring(9)) || page.equals("/Gujarat/login'
if (users.getDate() != null) {
    long diffMinutes = TimeUnit.MILLISECONDS.toMinutes(
        srf.parse(srf.format(System.currentTimeMillis())).getTime() - users.getDate().getTime());
    diffMinutes = Math.abs(diffMinutes);
    System.err.println(users.getOtpcount() + "..." + diffMinutes);
    if (users.getOtpcount() >= 2 && diffMinutes < 24*60) {
        return "Sorry! You can generate OTP 3 times only within 24 hrs, if have time left";
    } else {
        if (diffMinutes < 24*60) {
            flag = true;
            userDao.updateOPT_withouttime(users.getUsername(), otp, flag);
        }
    }
}
}

```

AES (Advanced Encryption Standard) Encryption

The Advanced Encryption Standard (AES) is a symmetric block cipher. AES is essential for government computer security, cyber security and electronic data protection.

```
application-... AdminControl... Users.java SecurityConf... application... identityConten... SessionCooki... login.html 22
<script th:inline="javascript">
    $('#loginbutton').on('click', function (e) {
        var aesUtil = new AesUtil(128, 1000);
        $('#loginform input[name="password"]').val(aesUtil.encrypt($('#loginform input[name="password"]').val(),
        $('#loginform input[name="username"]').val(aesUtil.encrypt($('#loginform input[name="username"]').val(),
        $('#loginform').submit();
    });
    $( document ).ready(function() {
        var error = [[${error}]];
        if(error!=null) alert(error);
    });
    $(".otp").click(function (){
        var token = [[${_csrf.token}]];
        var aesUtil = new AesUtil(128, 1000);
        /* alert(token); */
        $.ajax({
            url:"otp/?_csrf="+ token,
            method:"POST",
            contentType: 'application/json',
            data:JSON.stringify({"username":aesUtil.encrypt($('.username').val(),[[${_string_to}]]),"password":aesUt
            success:function(data)
            {
                alert(data);
            }
        });
    });
</script>
<script th:src="@{/js/crypto-js.min.js}"></script>
<script th:src="@{/js/AesUtil.js}"></script>
```

LoginController:

```
@PostMapping("/otp/")
@ResponseBody
public String otp(@RequestBody Object loginForm) throws ParseException {
    String otp= new DecimalFormat("000000").format(new Random().nextInt(999999));
    JSONObject obj=new JSONObject(new Gson().toJson(loginForm));
    Users users = userDao.findFirstByUsername(decode(obj.getString("username")));

    public String decode(String input) {
        String decryptedPassword = new String(java.util.Base64.getDecoder().decode(input));
        System.out.println(decryptedPassword+"decryptedPassword");
        AesUtil aesUtil = new AesUtil(128, 1000);
        String inputdecoded=aesUtil.decrypt(decryptedPassword.split("::")[1], decryptedPassword.split("
        return inputdecoded;
    }
}
```

Use the same key which is used for encoding.

```
ssword.split("::")[1], decryptedPassword.split("::")[0], "1234567891234567", decryptedPassword.split("::")[2]);
```

Add a class: AESUtil.java

```
package bisag.gatishakti.utills;

import java.io.UnsupportedEncodingException;

public class AesUtil {
    private final int keySize;
    private final int iterationCount;
    private final Cipher cipher;
    public String salt;
    public String iv;

    public AesUtil(int keySize, int iterationCount) {
        this.keySize = keySize;
        this.iterationCount = iterationCount;
        this.salt=randomHex();
        this.iv=randomHex();
        try {
            cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        }
        catch (NoSuchAlgorithmException | NoSuchPaddingException e) {
            throw fail(e);
        }
    }

    public String encrypt(String salt,String iv, String passphrase,String input) throws NoSuchPaddingException,
        InvalidAlgorithmParameterException, InvalidKeyException,
        BadPaddingException, IllegalBlockSizeException {
```

CSRF (Cross Site Request Forgery) Implementation

Cross-Site Request Forgery (CSRF) is a type of attack that occurs when a malicious web site, email, blog, instant message, or program causes a user's web browser to perform an unwanted action on a trusted site when the user is authenticated.

Solution:

SecurityConfig.java

```
.csrf().disable();
```

Enable csrf()

Once it is enabled, you have to pass token for each request.

```
$.ajax({
  url: "otp/?_csrf="+ token,
  method: "POST",
  contentType: 'application/json',
  data: JSON.stringify({"username":aes
  success: function(data)
```

Each ajax call should use token. If any of the ajax calls is sent without token, request will be terminated.

Store csrf token in hidden variable

```
<input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}"/>
```

In ajax call, you can also pass token as shown below.

```
//$("#AddData").submit();
//window.open('noc?id='+enc,"_blank");
$.ajax({
  url: "getgeomdataforofrm",
  method: "POST",
  async: false,
  data: { _csrf:$('#token').val(), data: JSON.stringify(data)},
  success: function(j) {
    $('#dataModal_incident').modal('show');
    var json = JSON.parse(j)
```

Another method to pass token in form

```
application.... identifyConten... SessionCooki... login.html LoginControl... map3.html 24

    <select id="intersectsLayers">
    </select>
</div>
<div>
<button class="btn-primary btn-skype mt-2" onclick="IntersectsReportbylname()" type
<form th:action="@{/intersectsLayersreport/{token}(token=${_csrf.token})}"
      method="POST" id="intersectsLayersreport">
    <input type="hidden" name="wktforintersectsLayers" id="wktforintersects
    <input type="hidden" name="intersectedLayersname" id="intersectedlayers
</form>
```

If you do now want to define the token above way, you can also store as a hidden input parameter.

Outdated JQuery

Outdated Bootstrap

Improper Input Validation

Solution:

At server side, define validations in entity class.

```
mmonGroup.js  main.js  idenfyConten...  panel.js  switchlayers.js  map.html  MultiCriteri...  User.java  15

@Entity
@Table(name = "login_ngdr", uniqueConstraints = { @UniqueConstraint(columnNames = "user_id") })
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int user_id;
    @NotNull
    @NotBlank(message = "* Name field is required!!")
    @Size(min = 2,max = 15, message = "* Name must be between 2-15 characters!!")
    private String username;
    @NotNull
    @Pattern(regexp="(?!.*\\d)(?!.*)(?!.*[A-Z]).{6,}",message="Must contain at least one number and c
    private String password;
    private boolean enabled;
    private String captcha;

    @Column(name = "account_non_locked")
    private boolean accountNonLocked;
```

At Controller, use Binding result to address the response.

```
application...  SessionCooki...  login.html  LoginControl...  map3.html  Users.java  GedaControl...  26

// @AntiFlood(value=2000)
@PostMapping(value = "/savewindform")
public String savewindform(@ModelAttribute("windform") WindEntity wind, BindingResult result,
    WindMakedetails windmake1, MultipartHttpServletRequest req, Principal principal, Model model,
    RedirectAttributes attributes) throws Exception {

    File myDir = new File(folderpath);
    if (!myDir.exists()) {
        myDir.mkdirs();
    }

    if(result.hasErrors())
    {
    }
}
```

Upto 54:30 minutes

OTP Bypass

Solution:

SecurityConfig.java

```
user = Arrays.stream(allowedUrls).anyMatch(url -> url.equals(fullurl[4]))
DAO.findFirstByUsernameAndStatenameAndEnabled(username, appliactioncontext.getApplicationName())
DAO.findFirstByUsernameAndStatenameAndEnabledAndOtp(username, appliactioncontext.getApplicationName())

@Override
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
    String fullurl[] = request.getRequestURL().toString().split("/");
    String otp = request.getParameter("otp");
    System.err.println(username + "... " + appliactioncontext.getApplicationName().substring(0, 10));

    //AB// Add NEW SSO IN ALLOWEDURL TO GET IT WORKING //
    String[] allowedUrls = { "FORESTSTATUSAPI1", "FORESTSTATUSAPI", "CRZSTATUSAPI", "CRZSTATUSAPI" };
    Users user = Arrays.stream(allowedUrls).anyMatch(url -> url.equals(fullurl[4]))
    ? userDAO.findFirstByUsernameAndStatenameAndEnabled(username, appliactioncontext.getApplicationName())
    : userDAO.findFirstByUsernameAndStatenameAndEnabledAndOtp(username, appliactioncontext.getApplicationName(), otp);

    UserBuilder builder = null;
    if (user != null) {
        builder = org.springframework.security.core.userdetails.User.withUsername(username);
        builder.password(user.getPassword());
        builder.roles(user.getAuthorities().stream().map(auth -> auth.getAuthority()).collect(Collectors.toList()));
    }
    return builder.build();
}
```

In NGDR, it is implemented in User_Details.java file.

SQL Injection

SQL injection is a code injection technique that might destroy your database. SQL injection is one of the most common web hacking techniques. SQL injection is the placement of malicious code in SQL statements, via web page input.

Solution: Use prepared statement for any database operation.

```
public void insert(PersonEntity personEntity) {  
    String query = "INSERT INTO persons(id, name) VALUES( ?, ?)";  
  
    PreparedStatement preparedStatement = connection.prepareStatement(query);  
    preparedStatement.setInt(1, personEntity.getId());  
    preparedStatement.setString(2, personEntity.getName());  
    preparedStatement.executeUpdate();  
}
```

References:

<https://portswigger.net/>

<https://owasp.org/>

4.0 Application Security Test Summary:

Sr. No.	Vulnerability Details
1.	<p>Unencrypted Login Request i.e. Insufficient Transport layer Protection: It was observed that User credentials i.e. User Id & Password value is unencrypted & travel as clear text. Vulnerable URL: http://10.247.22.132/UrbanPortal/auth/login_check Refer Annexure 'A" Sr. No.-1</p>
2.	<p>AUTOCOMPLETE Enabled (Remember credential): AUTOCOMPLETE attribute is not disabled in HTML FORM/INPUT element containing user id & password type input. It was observed that the user ID & Password of all the registered users is stored in the login form (with Auto complete Enabled) which may be retrieved & misuse by any user having the physical access to machine. Vulnerable URL: http://10.247.22.132/UrbanPortal/auth/login_check Refer Annexure 'A" Sr. No.-2</p>
3.	<p>No Password Policy : The Password policy is not implemented .The password for admin user supplied is set as admin.Could not be verified as change password is not implemented. Password Policy needs to be defined keeping in view the Password complexity requirements such as composition, length, Expiry time & password history requirements. Vulnerable URL: http://10.247.22.132/UrbanPortal/auth/login_check</p>
4.	<p>Brute Force Attack is possible: Application has no provision to stop the machine based attacks. User ID is not locked for any number of invalid login attempts. It may be handled by using Captcha after a certain no. of invalid attempts to protect the application from brute force type of attacks. Vulnerable URL: http://10.247.22.132/UrbanPortal/auth/login_check</p>
5.	<p>Session Management - Not Implemented : Session token Updation after authentication, new session for every user & session expiry with logout & browser close button are not implemented. The Sessions must have a limited lifetime and expire after a certain period of time based on business and usability requirements balanced with security considerations. The application should be able to measure the period of inactivity for a session and expire it, destroying the session and overwriting the session cookie based on Logout or Browser Close button. Vulnerable URL: http://10.247.22.132/UrbanPortal/admin/gisModule</p>

STQC-IT, Delhi Application Security Test Report

Application Security Test Report ID	Date	Page No.
STQC-IT Delhi/DeitY/NeGD/Urban Portal/STR-01	23/02/2016	7 of 13

6.	<p>SQL Injection: Sanitation of hazardous characters was not performed correctly on user input. The manual test results confirm vulnerability because the response contains SQL Server errors. This suggests that the test managed to penetrate the application and reach the SQL query itself, by injecting hazardous characters.</p> <p>Vulnerable URL: http://10.247.22.132/UrbanPortal/auth/login_check?targetUrl=</p> <p>Refer Annexure 'A' Sr. No.-3</p> <p>For complete list and more details please refer the AppScan report enclosed.</p>
7.	<p>Cross-Site Scripting: Sanitation of hazardous characters was not performed correctly on user input. The manual & automated test results confirm vulnerability because through manual testing we are able to successfully embedded a script in the response, which will be executed when the page loads in the user's browser.</p> <p>Vulnerable URL: http://10.247.22.132/UrbanPortal/admin/misModule?msg=Data+Saved</p> <p>Refer Annexure 'A' Sr. No.-4</p>
8.	<p>Forced Browsing & Broken Authentication:</p> <p>The functionalities & authenticated pages which need to be accessed through application flow are directly accessible by providing their URI in the address bar i.e. directly going to a particular page through change in its object id from URL. It is observed for most of the pages.</p> <p>The Forward & Back button can also be used to access the function directly bypassing the application flow.</p> <p>Vulnerable URL: http://10.247.22.132/UrbanPortal/admin/gisModule</p>
9.	<p>Buffer Overflow: The web server or application server are configured in an insecure way as no validation on input at server side as it is possible to supply any number of characters in the user ID & Password field which resulted into buffer overflow.</p> <p>The application shows "500 internal server error" when 1000 characters are applied to the field 'Username' & 'Password'.</p> <p>Vulnerable URL:</p> <p>http://10.247.22.132/UrbanPortal/auth/login_check</p> <p>Refer Annexure 'A' Sr. No.-5</p>

10.	Web Application Source Code Disclosure Pattern Found: The response contains source code of script files, which may expose sensitive information about the site and the application logic.It is possible to retrieve the source code of server-side scripts, which may expose the application logic. Vulnerable URL: http://10.247.22.132/docs/jndi-resources-howto.html		
	Refer Annexure 'A" Sr. No.-6 For complete list and more details please refer the AppScan report enclosed.		
STQC-IT, Delhi Application Security Test Report			
Application Security Test Report ID		Date	Page No.
STQC-IT Delhi/DeitY/NeGD/Urban Portal/STR-01		23/02/2016	8 of 13

11.	Missing "HttpOnly"&"Secure" Attribute in Session Cookie: It is observed that that a session cookie is used without the "HttpOnly"& “Secure”attribute. It is recommended to add the 'HttpOnly' attribute to all session cookies. Vulnerable URL: http://192.168.1.5/dotproject/index.php Refer Annexure ‘A” Sr. No.-7		
12.	Application Error message/ Exception handling : Application is unable to handle the exceptions/ Runtime error as it displays the error message “HTTP/1.1 500 Internal Server Error”revealing the details of platform info & database structure which may be misused by the attackers. Application error messages reveal the server details. Vulnerable URL: http://10.247.22.132/UrbanPortal/admin/getLatLonList Refer Annexure ‘A" Sr. No.- 8 For complete list and more details please refer the AppScan report enclosed.		
13.	Logging of Authentication Events is not configured: Enablment of the logging of authentication events such as logins, logouts, password resets etc. are not configured in the application.		

Note: For further details, please refer the detailed security scan report (IBM Rational APPSCAN Report) provided separately for these URLs.

Remarks:

1. The web application allows the use of default account using user id & password as admin/admin.
2. SSL is not implemented.
3. Change Password & Forgot Password feature are not available.
4. Session timeout is not implemented
5. Logging of authentication events such as Logins, logouts, password changes, password resets etc. required to be logged by the application are not configured.
6. Secure Storage of Credentials at server level could not be verified.
7. Privilege escalation could not be verified as only single user credentials are shared with STQC.

Reference: Application Security Test Report 23022016_GLIS.pdf