Celah Keamanan pada Framework Java
“ Spring Web MVC ”

Cakra Adipura Wicaksana
23214322
Program Studi Magister Teknik Elektro
Sekolah Teknik Elektro dan Informatika (STEI)
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung, Jawa Barat, Indonesia
adipurapunya@gmail.com
adipura@students.itb.ac.id

Abstrak— **Saat ini banyak framework yang *free* untuk mengembangkan aplikasi berbasis bahasa pemrograman java seperti framework spring, framework database hibernate dan ebatis, framework struts, framework spring web mvc, dan framework lainnya. Pada makalah ini akan dibahas hanya tentang framework spring web mvc saja, yaitu celah keamanan apa saja yang ada pada framework ini. Sebelum membahas celah keamanannya, makalah ini juga akan menjelaskan secara singkat tentang penggunaan spring MVC. Ada dua celah keamanan yang akan dibahas pada makalah ini, yaitu memasukkan Spring Data MVC ke textfield yang tidak bisa diubah (*Spring MVC Data Submission to Non-Editable Fields*) dan Injeksi pada Spring MCV Model dan View (*Spring MVC ModelView Injection*). Dari kedua celah keamanan tersebut, akan dijelaskan satu per satu dengan menggunakan contoh-contoh yang telah ada sebelumnya.**

Kata Kunci— **Spring MVC; Web MVC; Framework Java; Celah Keamanan Spring; Java Spring; Java MVC.**

I. PENDAHULUAN

1.1 Latar Belakang

Java merupakan suatu bahasa pemrograman yang kian diminati saat ini. Java diminati karena *free* alias gratis dan banyak depelover-depelover yang menggunakan java sebagai *tools* untuk membuat suatu aplikasi. *Client-client* seperti perusahaan besar banyak yang menggunakan java sebagai toolsnya untuk mengurangi biaya pembuatan aplikasi karena tidak perlu untuk membayar biaya apapun.

Untuk membuat suatu aplikasi biasanya para developer menggunakan suatu kerangka kerja atau framework untuk mempermudah dan mempercepat proses pembuatan aplikasi sehingga para programmer hanya fokus pada logic dan goal dari aplikasi yang akan dibuat. Framework-framework java yang biasa digunakan oleh para programmer yang umum adalah Spring sebagai logic nya, Spring MVC sebagai antarmukanya, dan Hibernate atau eBatis sebagai framework untuk berhubungan dengan database.

Jika ditinjau dari segi keamanan, masing – masing framework memiliki kelebihan dan kekurangan tersendiri. Pada Makalah ini, tidak akan dijelaskan framework secara keseluruhan, penulis hanya akan menjelaskan framework spring MVC saja mengenai apa saja celah-celah atau kerentanan yang mungkin terjadi pada framework spring MVC.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijabarkan sebelumnya, berikut adalah permasalahan yang akan dibahas, yaitu :

1. Apa berapa celah keamanan atau kerentanan yang terdapat pada framework spring MVC ?
2. Kerentanan seperti apa yang terdapat pada framework spring MVC ?

II. LANDASAN TEORI

2.1 Definisi Framework

Framework adalah kerangka kerja. Framework juga dapat diartikan sebagai kumpulan script (terutama class dan function) yang dapat membantudeveloper/programmer dalam menangani berbagai masalah-masalah dalam pemrograman seperti koneksi ke database, pemanggilan variabel, file, dan lain-lain sehingga developer lebih fokus dan lebih cepat membangun aplikasi. Bisa juga dikatakan bahwa framework adalah komponen pemrorgaman yang siap re-use kapan saja sehingga programmer tidak harus membuat skrip yang sama untuk tugas yang sama [6].

2.2 Jenis-Jenis Framework Java

Berikut adalah framework-framework java yang sering digunakan [6] :

1. Untuk Business Logic
   Commons Chain of Responsiblity, Spring, dan XWork
2. Untuk Data Access
   Cayenne, Enterprise Java Beans, Hibernate, iBATIS, JDBC, Object Relational Bridge
3. Untuk View
   Freemarker, iText (PDF), JasperReports, Velocity, XSLT

2.3 Framework Spring

Spring adalah framework pengembangan aplikasi ysang dikembangkan oleh Rod Johnson, muncul karena spesifikasi EJB yang memaksakan pengembangan komponen harus mengikuti aturan EJB agar dapat berjalan dalam aplikasi server SJAS dan Jboss. Dengan Spring, pengembangan komponen dapat dilakukan dengan teknik pemrograman yang lebih sederhana [6].

Berikut adalah keuntungan dari framework Spring [6]:

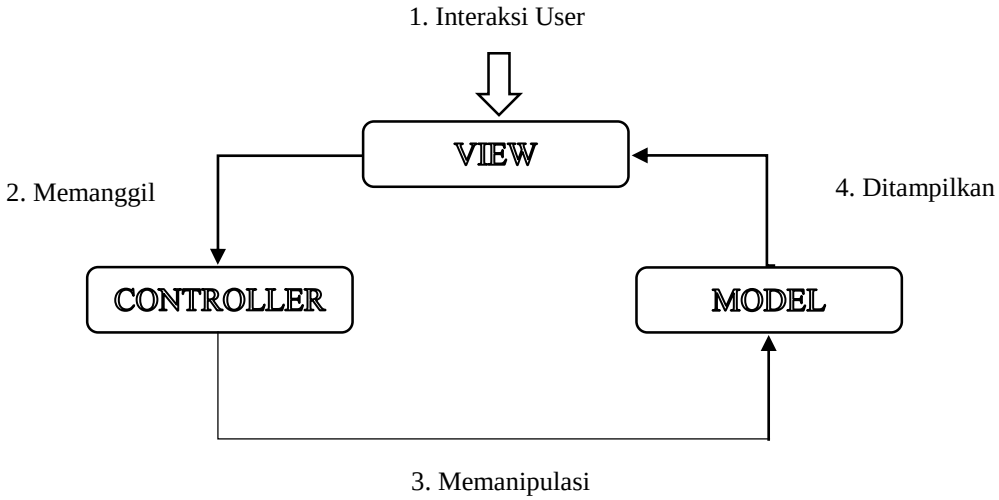
1. IoC
   Sebagai sebuah framework, spring menawarkan loosely coupling dengan teknik yang dinamakan IoC. jadi dengan menggunakan IoC, objek memberikan dependenciesnya saat pembuatan dengan menggunakan entityluar yang mengkoordinir setiap objek di sistem. Untuk lebih jelasnya akandijelaskan pada bagian Dependency Injection.
2. AOP
   AOP merupakan salah satu paradigma pemrograman dengan tujuan untuk meningkatkan modularitas dengan memfokuskan pada pemisahan- pemisahan modul dengan tujuan-tujuan khusus yang biasa disebut dengan "*Crosscutting Concerns*".
3. Bersifat *Container*
   Spring juga merupakan sebuah Container yang mengatur daur hidup dan konfigurasi dari objek. Dalam spring kita dapat mendeklarasi bagaimana setiap objek tersebut seharusnya

dibuat, bagaimana seharusnya dikonfigurasikan dan bagaimana objek tersebut dapat berasosiasi dengan yang lain.

4. *Lightweight Container*
   Beberapa container aplikasi seperti, EJB Container memaksa kita mengikuti suatu aturan (EJB-spec) untuk membuat komponen interface atau model. EJB merupakan standar pengembangan komponen dalam java yg berjalan disisi server dengan suatu kontrak terhadap aplikasi server seperti SJAS dan JBoss.

2.4 MVC

MVC adalah singkatan dari Model, View, Controller, yang merupakan sebuah arsitektur untuk membuat sebuah program. Arsitektur ini menekankan kepada pembagian dari komponen-komponen program menjadi tiga bagian utama, yaitu Model, View, dan Controller [6].

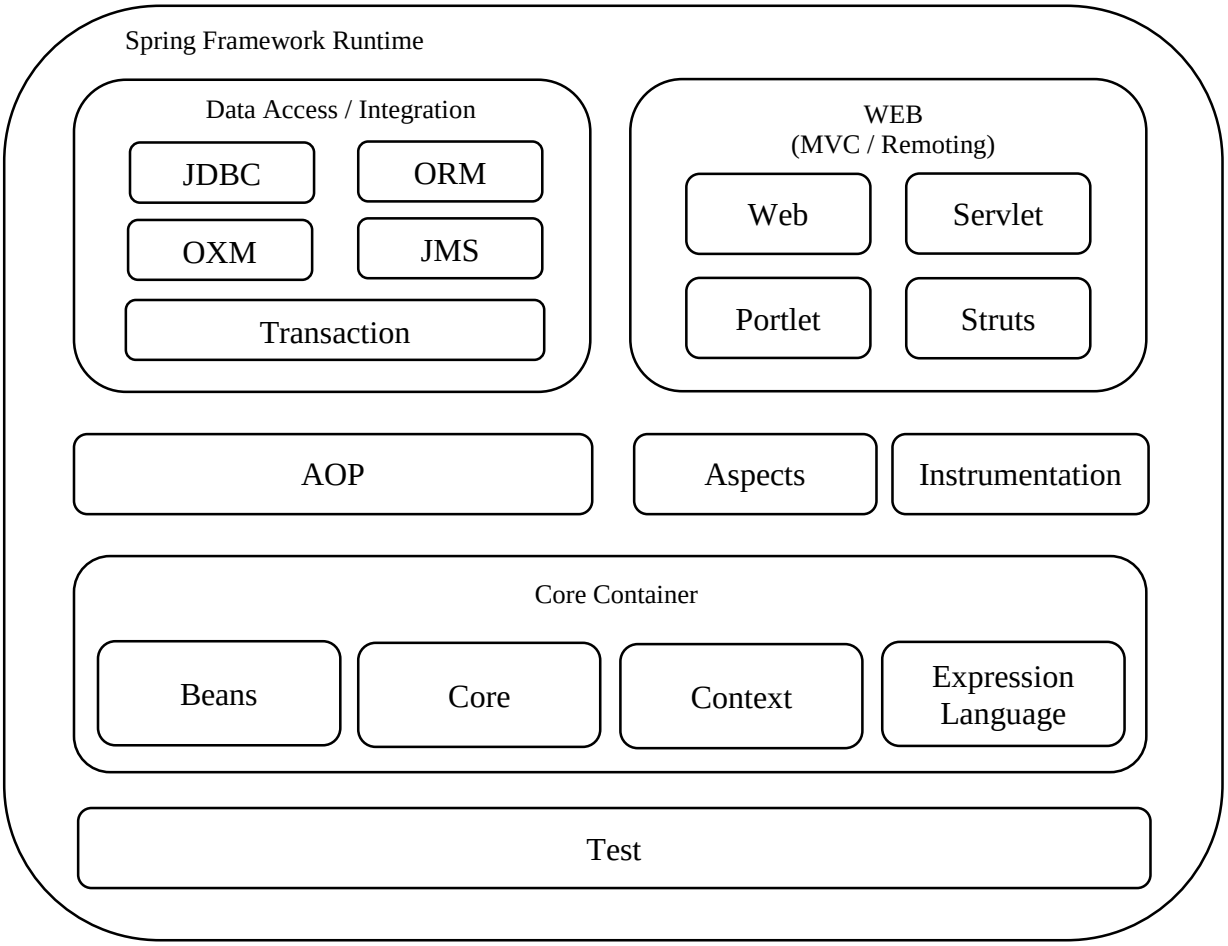


Gambar 2.1 Arsitektur MVC

Interaksi user dengan program digambarkan dengan arah panah besar yang menuju View. Kemudian View memanggil Controller. Selanjutnya Controller akan membuat atau memanipulasi Model. Model ini akan diberikan kepada View untuk ditampilkan kepada user.Dengan demikian tugas View adalah menangani tampilan program dan interaksiantara user dengan program. Controller melakukan koordinasi antara View dan Model, sedangkan Model adalah bagian yang bekerja di belakang layar untuk memenuhi permintaan user dalam sebuah interaksi. Tujuan dari pembagian program ke dalam tiga bagian besar ini adalah untuk memisahkan fokus perhatian, tanggung jawab, dan logic ke dalam bagian masing-masing. View hanya fokus kepada tampilan dan menangani interaksi dengan user. Model hanya fokus kepada manipulasi objek-objek non-visual dan logic di dalamnya untuk memenuhi skenario sebuah proses bisnis, sedangkan Controller menerima input dari View, membuat dan memanipulasi Model, lalu memberikan Model kepada View untuk ditampilkan ke user[6].

2.5 Spring Web MVC

Spring Web MVC ini merupakan bagian dari framework spring atau Spring Web MVC ini adalah modul dari framework Spring. Berikut ini adalah diagram overview dari Spring sehingga kita bisa mengetahui posisi dari Spring Web MVC ini sebenarnya



Gambar 2.2 Diagram dari Spring Framework[8]

III. METODE PENELITIAN

3.1 Jenis Penelitian

Penelitian ini termasuk jenis penelitian studi literatur dengan mencari referensi teori yang relefan dengan kasus atau permasalahan yang ditemukan. Referensi teori yang diperoleh dengan jalan penelitian studi literatur dijadikan sebagai fondasi dasar.

3.2 Metode Pengumpulan Data

Jenis data yang digunakan penulis dalam penelitian ini adalah data sekunder, yaitu data yang diperoleh dari jurnal, buku dokumentasi, dan internet.

3.2.1 Studi Literatur

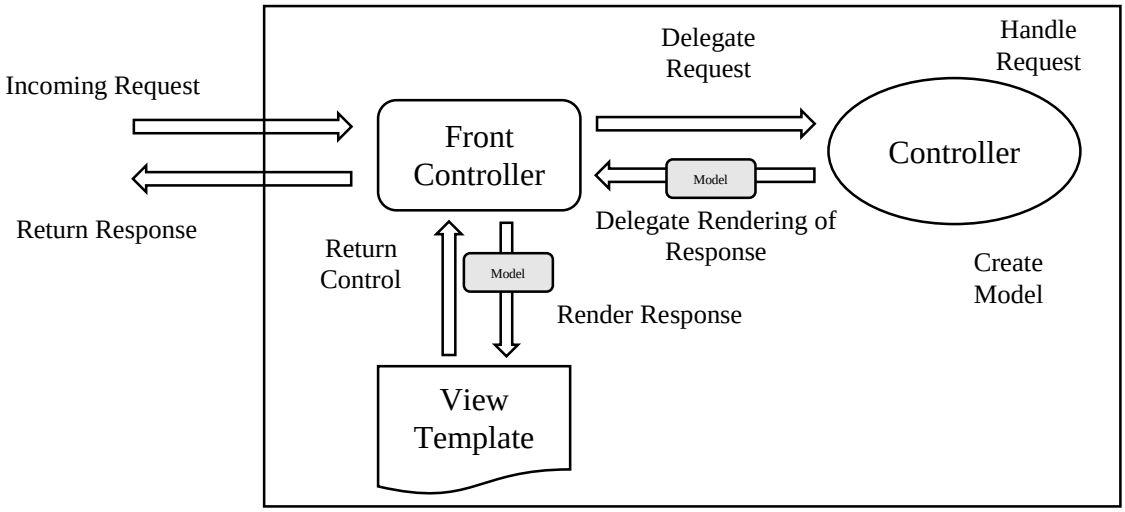
Studi literatur adalah cara yang dipakai untuk menghimpun data-data atau sumber-sumber yang berhubungan dengan topik yang diangkat dalam suatu penelitian. Studi literatur bisa didapat dari berbagai sumber, jurnal, buku dokumentasi, internet, dan pustaka

IV. PEMBAHASAN

Celah-Celah Keamanan Pada Framework Java Spring MVC :

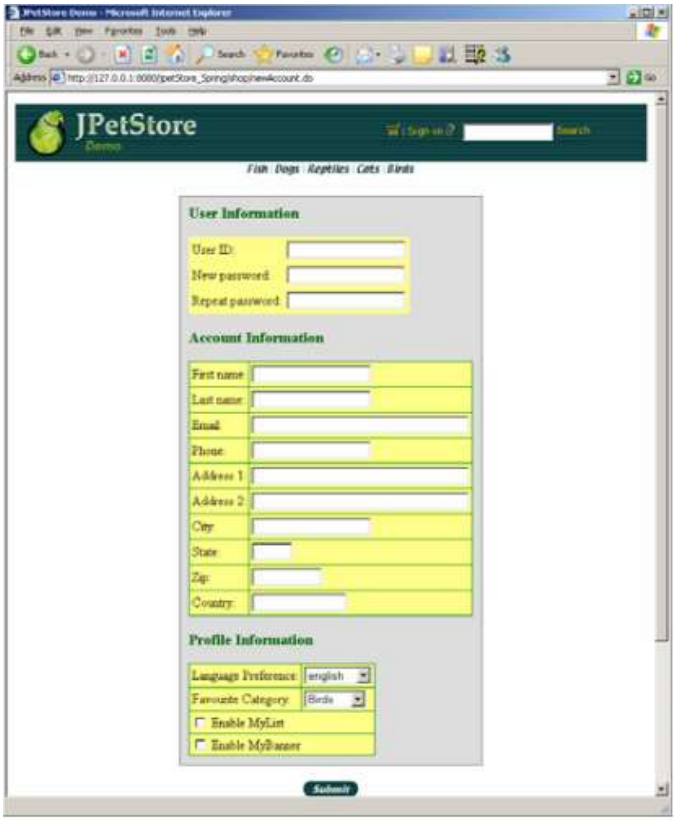
1. *Spring MVC Data Submission to Non-Editable Fields*

Spring MVC adalah sebuah pola pengimplementasian dari Model View dan Controller. Berikut adalah diagram alir proses request dengan contoh Servlet Engine nya Tomcat.



Gambar 4.1 Gambar request processing Workflow [1]

Pada Spring ini berarti ada sebuah binding otomatis dari user berupa data yang disubmit ke dalam POJO Class yang mengandung bisnis logic. Contohnya pada web jPetStore Application berikut "*new user form*" dengan link http://127.0.0.1:8080/jpetStore_Spring/shop/newAccount.do :



Gambar 4.2 Gambar jPetStore Application ketika user baru dibuat [1]

Pada gambar 4.2 ketika user mengklik tombol submit, Spring MVC secara otomatis mengisi nilai dari field-field form ke objek perwakilan yang ada pada server side. Prosesnya kan ditunjukkan pada script di bawah ini :

```
<font color="darkgreen"><h3>User Information</h3></font>
<table border="0" cellpadding="3" cellspacing="1" bgcolor="#FFFF88">
<tr bgcolor="#FFFF88"><td>
User ID:</td><td>
<c:if test="${accountForm.newAccount}">
    <spring:bind path="accountForm.account.username">
        <input type="text" name="<c:out value="${status.expression}"/>" value="<c:out value="${status.value}"/>"/>
    </spring:bind>
</c:if>
....
<br /><center>
<input border="0" type="image" src="../images/button_submit.gif" name="submit" value="Save Account Information" />
</center>
    protected ModelAndView onSubmit(
            HttpServletRequest request, HttpServletResponse response, Object command, BindException errors)
            throws Exception {

        AccountForm accountForm = (AccountForm) command;
        try {
            if (accountForm.isNewAccount()) {
                this.petStore.insertAccount(accountForm.getAccount());
public class AccountForm implements Serializable {

    private Account account;

    private boolean newAccount;

    private String repeatedPassword;
public class Account implements Serializable {

  /* Private Fields */

  private String username;
  private String password;
  private String email;
  private String firstName;
  private String lastName;
  private String status;
  private String address1;
  private String address2;
  private String city;
  private String state;
  private String zip;
  private String country;
  private String phone;
  private String favouriteCategoryId;
  private String languagePreference;
  private boolean listOption;
  private boolean bannerOption;
  private String bannerName;
```

Gambar 4.3 Tampilan Script Proses Pengisian Nilai Field ke Server Side [1]

Data pada objek yang telah dibuat dari *AccountForm.Account* dihuni oleh dynamic setter seperti pada script di bawah ini :

```
*Account.java   AccountForm.java   *petstore-servlet...

public void setUsername(String username) {
    this.username = username;
    }
```

Gambar 4.4 Tampilan Script Dynamic Setter [1]

Sebagai contoh lain, untuk menambahkan field baru pada form, para programmer perlu untuk melakukan hal-hal berikut :

- Menambahkan fieldnya itu ke halaman jsp :

```
<spring:bind path="accountForm.account.firstNameXXX">
    <input type="text" name="<c:out value="${status.expression}"/>" value="<c:out value="${status.value}"/>"/>
</spring:bind>
```

Gambar 4.5 Proses Penambahan Binding firstNameXXX[1]

- Menambahkan getter dan setter untuk field tersebut pada class targer.

```
public String getFirstNameXXX() { return firstNameXXX; }
public void setFirstNameXXX(String firstNameXXX) { this.firstNameXXX = firstNameXXX; }
```

Gambar 4.6 Proses Penambahan Getter dan Setter firstNameXXX[1]

Setelah proses pengisian field, Spring MVC secara otomatis akan memanggil setter yang bersangkutan dan mengisinya dengan nilai yang telah disubmit pada HTTP field form accountForm.account.firstNameXXX sebagai suatu variabel perintah ke dalam suatu fungsi onSubmit sebagai berikut :

```
protected ModelAndView onSubmit(
        HttpServletRequest request, HttpServletResponse response, Object command, BindException errors)
        throws Exception {

    AccountForm accountForm = (AccountForm) command;
    try {
        if (accountForm.isNewAccount()) {
            this.petStore.insertAccount(accountForm.getAccount());
```

Gambar 4.7 Script Fungsi OnSubmit[1]

Spring MVC Data Submission to Non-Editable Fields memiliki kerentanan yang dibuat melalui fitur "*auto binding*" karena dalam banyak kasus tidak semua field diekspos oleh pemetaan dari POJO Class.

Karena dimungkinkan bagi penyerang untuk mengirimkan data ke semua field pada Class yang digunakan untuk mem-bind data yang diterima jadi kerentanannya adalah sebagai berikut :

- Ada perbedaan antara field pada web (misalnya pada JSP) dengan field setter pada Class Backend.
- Para developer tidak menyadari bahwa semua field-field pada form bisa diisi oleh data yang bisa membahayakan.
- Manipulasi dari extra field ini (yang dikontrol oleh penyerang) memungkinkan pengelakan dari bisnis logic aplikasi.

2. *Spring MVC ModelView Injection*

Pada kerentanan Spring MVC yang kedua ini dapat memungkinkan penyerang tidak hanya mem-bypass proses bisnis logic saja, tetapi dapat mendownload semua aplikasi untuk membongkarnya. Ketentanan ini tercipta oleh suatu alur yang digunakan oleh Spring MVC untuk menentukan View mana saja yang harus diberikan kepada user ketika user melakukan proses request. Supaya bisa dieksploitasi atau dimanipulasi, arsitektur aplikasi baik dari desain nya atau dari kesalahan-kesalahan yang ada memungkinkan data user untuk digunakan sebagai *View Name*.

Spring MVC Model View

Spring MVC dibangun dari *Front Controller* untuk diimplementasikan sebagai servlet dasar yang mengirimkan request dari *Controller* (dipetakan pada file web.xml) :

```
<web-app>
<servlet>
  <servlet-name>MyFrontController</servlet-name>
  <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <load-on-startup>2</load-on-startup>
</servlet>
<!—all requests will be dispatched through Spring →
<servlet-mapping>
  <servlet-name>MyFrontController</servlet-name>
  <url-pattern>*.*</url-pattern>
  </servlet-mapping>
</web-app>
```

Gambar 4.8 Konfigurasi File web.xml [1]

Ketika request dari user disampaikan ke *Front Controller*, berikut ini adalah aksi-aksi yang terjadi :

1. WebApplicationContext akan dicari dan di-bound dalam request sebagai sebuah atribut sehingga dapat dibuat tersedia dalam keseluruhan proses.
2. Variabel-variabel local di-bound untuk menjadikannya global.
3. Resolver di-bound untuk dimungkinkan diubah ke bentuk view based.
4. Jika Resolver tersebut diperiksa maka akan dikemas menjadi MultipartHttpServletRequest.
5. Semua Handler Mapping yang terdefinisi pada file Spring Configuration akan dicari untuk menemukan handler yang sesuai dengan permintaan. Semua handler akan dicoba sampai Handler Mapping yang sesuai ditemukan.

```
<!— map all requests to a single controller →
<bean
class=org.springframework.web.servlet.handler.SimpleUrlHandlerMapping”>
      <property name =”mapping”>
            <props>
                  <prop key=”/*”>MyController</prop>
            <props>
      </property>
</bean>
```

Gambar 4.9 File Spring Configuration [1]

6. View resolver adalah entitas yang bertanggung jawab untuk menemukan view yang akan dimuat pada respon.

Berikut adalah contoh dari View Resolver dan Controller untuk menyoroti permasalahannya :

```
<bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
            <property name="viewClass"
value="org.springframework.web.servlet.view.JstlView"/>
            <property name="prefix" value="/WEB-INF/jsp/"/>
            <property name="suffix" value=".jsp"/>
</bean>
```

Gambar 4.10 Contoh sederhana View Resolver dan Controller [1]

Resolver ini muncul untuk menyelesaikan semua request ke halaman jsp pada direktori WEB-INF/jsp. Jadi sebuah request dari form http://my-domain/resource akan mencoba memuat tampilan dari /WEB-INF/jsp/resource.jsp.

Contoh Kerentanan :

Berikut diberikan sebuah Controller

```
Class MyController implements Controller {
public ModelAndView handleRequest(HttpServletRequest request,
                                  HttpServletResponse response) throws
Exception {
    String page = request.getParameter("page");
    if (page!=null)
        return new ModelAndView(page);
    return new ModelAndView(this.successView, "cart", cart);

}
```

Gambar 4.11 Contoh Script Controller pada Spring Web MVC [1]

Controller tersebut akan muncul berdasarkan konfigurasi yang diperoleh dari tampilan form /WEBINF/jsp/<page>.jsp

Perhatikan UrlBasedViewResolver berikut ini :

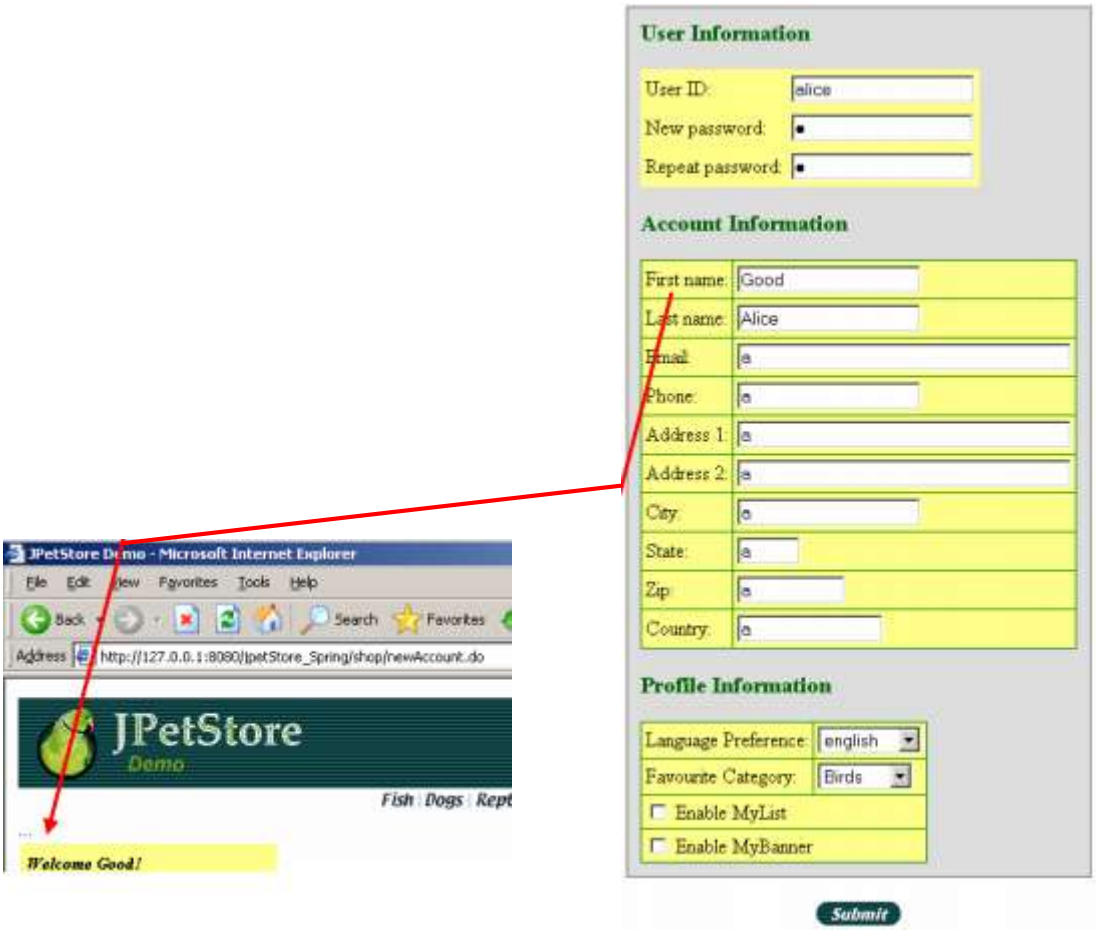
```
protected View createView(String viewName, Locale locale) throws Exception {
// If this resolver is not supposed to handle the given view,
  // return null to pass on to the next resolver in the chain.
  if (!canHandle(viewName, locale)) {
        return null;
  }
  // Check for special "redirect:" prefix.
  if (viewName.startsWith(REDIRECT_URL_PREFIX)) {
        String redirectUrl = viewName.substring(REDIRECT_URL_PREFIX.length());
return new RedirectView(redirectUrl, isRedirectContextRelative(),
isRedirectHttp10Compatible());
  }
  // Check for special "forward:" prefix.
  if (viewName.startsWith(FORWARD_URL_PREFIX)) {
        String forwardUrl = viewName.substring(FORWARD_URL_PREFIX.length());
        return new InternalResourceView(forwardUrl);
  }
  // Else fall back to superclass implementation: calling loadView.
  return super.createView(viewName, locale);
}
```

Gambar 4.12 Contoh Script UrlBasedViewResolver pada Spring Web MVC [1]

Metode ini dipanggil untuk menampilkan atau membuat suatu view. Hal yang menarik adalah script yang telah ditandai di mana jika ada awalan khusus berupa pra-pended maka akan diteruskan ke internal view resolver. Jika pengguna membuat suatu request dengan http://my-domain/forward:/WEB-INF/web.xml , pengguna akan diberikan akses ke sebuah resource yang di mana pengguna tidak boleh untuk mendapatkannya.

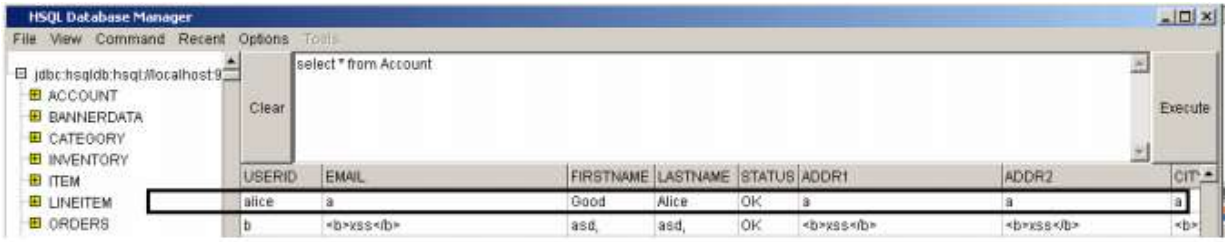
Contoh Studi Kasus : “Meraih kepemilikan sebuah akun”

Masih di aplikasi web yang sama, yaitu jPetStore Application



Gambar 4.13 Membuat Akun dan Halamn ketika akun berhasil dibuat [1]

Setelah akun dengan nama alice terbuat maka didatabase akan tersimpan seperti berikut :



Gambar 4.14 Tampilan data telah berhasil disimpan ke database [1]

Kemudian Alice akan memulai untuk berbelanja dan pada akun tersebut akan history belanja yang sudah pernah dilakukan sebelumnya.

Gambar 4.15 Tampilan Akun Alice yang sedang berbelanja dan history belanja [1]

Setelah itu kita logout akun Alice dan membuat user baru dengan nama “Evil Eve”.

Gambar 4.16 Proses Pembuatan Akun baru dengan nama“Evil Eve” [1]

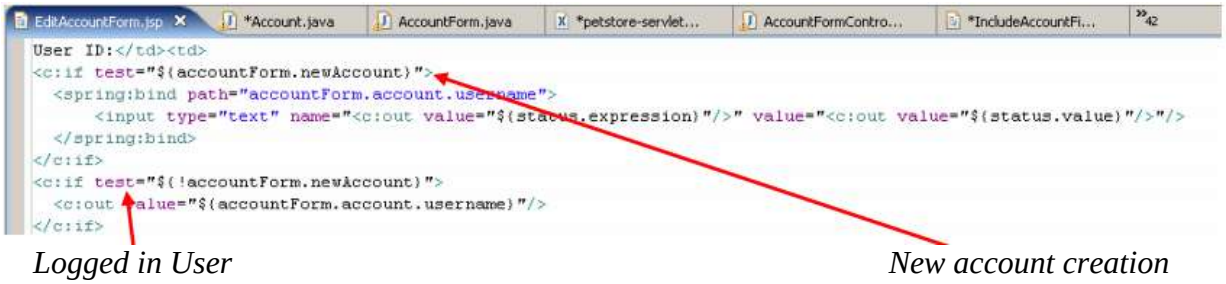
Setelah itu makan user Evil Eve akan berhasil dibuat

Gambar 4.17 Pembuatan Akun baru “Evil Eve” berhasil dibuat [1]

User evil eve akan menuju home page nya dan memilih “My Account“ page

Gambar 4.18 Halaman “My Account Page” Akun baru “Evil Eve” [1]

Potongan kode script berikut menunjukan bagaimana proses ini diimplementasikan :

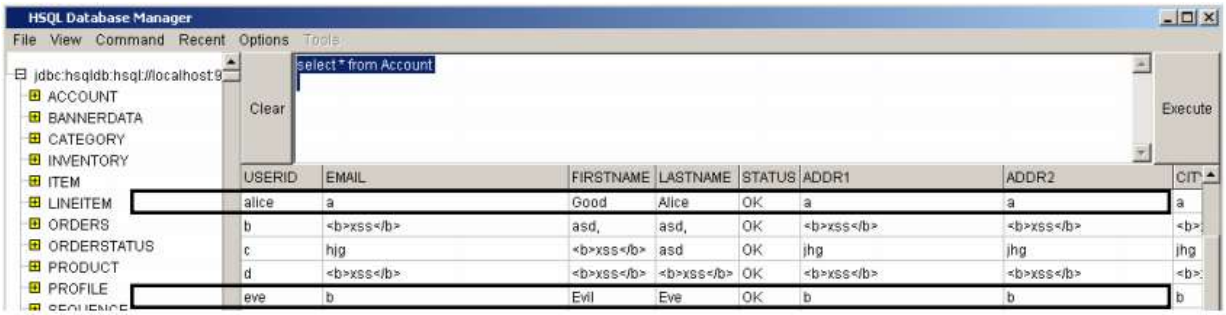


Gambar 4.19 Potongan Script dari file jsp, yaitu EditAccoutForm.jsp [1]

Hal ini berarti bahwa satu-satunya perubahan yang dibuat adalah "*Client Side Data Validation*" yang dengan mudah bisa dilewati dengan menggunakan proxy.

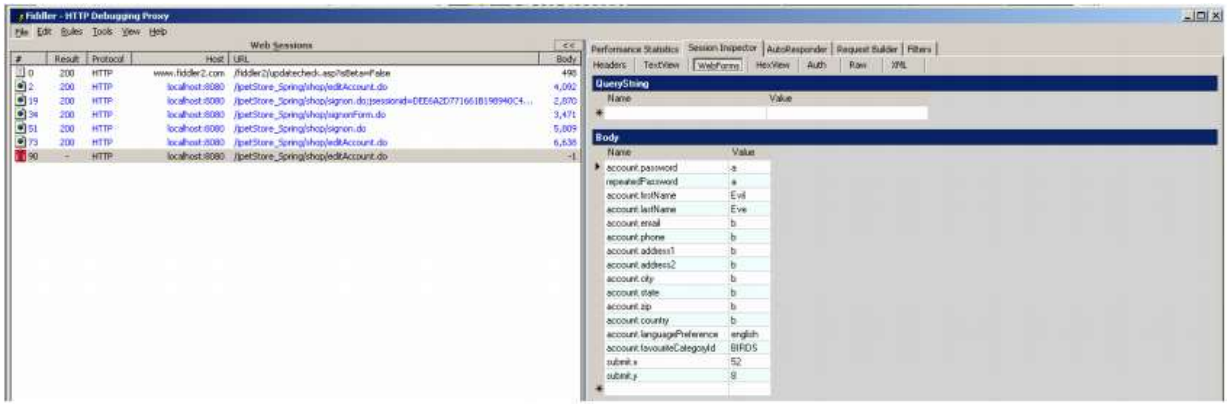
Proses Exploitasi :

Sebelum melakukan proses exploitasi, kita lihat di database terlebih dahulu untuk akun "Alice" dan "Evil Eve".



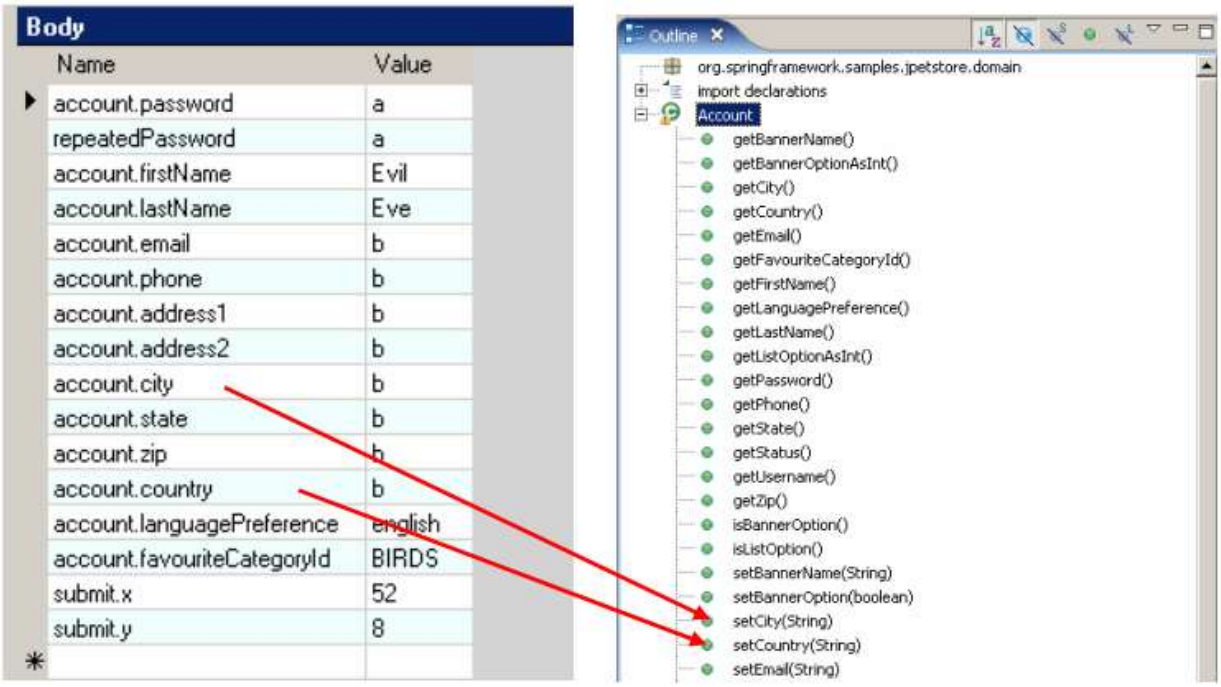
Gambar 4.20 Data akun "Alice" dan "Evil Eve" pada database [1]

Sekarang kita gunakan web proxy untuk mencegat request yang dikirim. Pada contoh di bawah ini digunakan fiddler sebagai toolsnya.



Gambar 4.21 Tampilan Tool Web Proxy Fiddler [1]

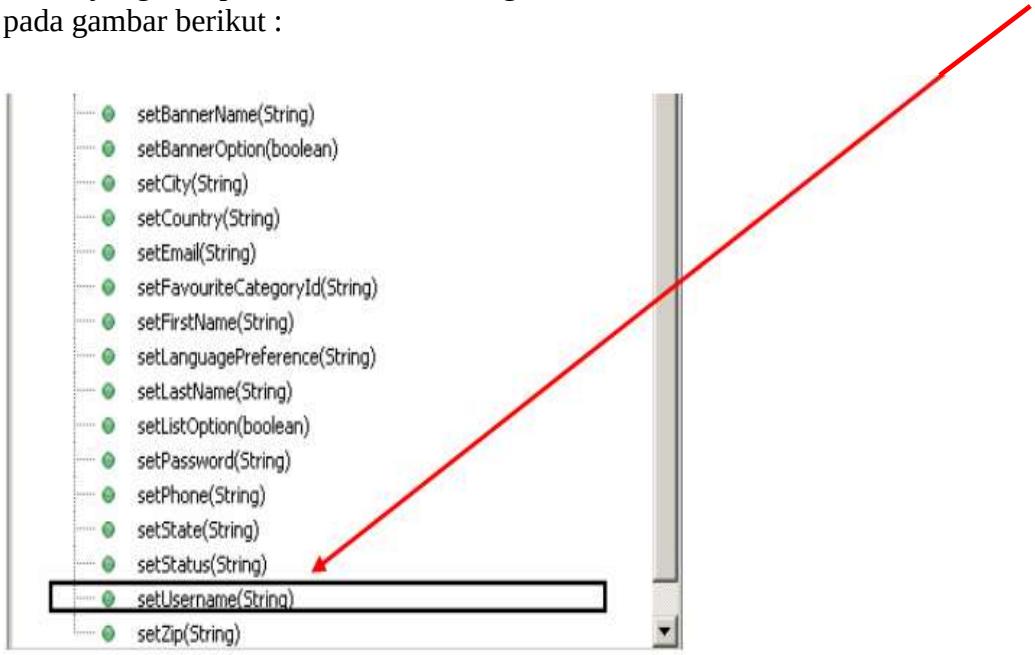
Berikut adalah pemetaan langsung antara kolom-kolom pada field dan java field name.



Gambar 4.22 Pemetaan Field-Field [1]

Satu-satunya yang terjadi adalah konversi dari account.email Account's setEmail() (Menambahkan set dan memanfaatkan nama pertama).

Setter yang ada pada Class akun mengidentifikasi keberadaan metode setUserName() seperti pada gambar berikut :

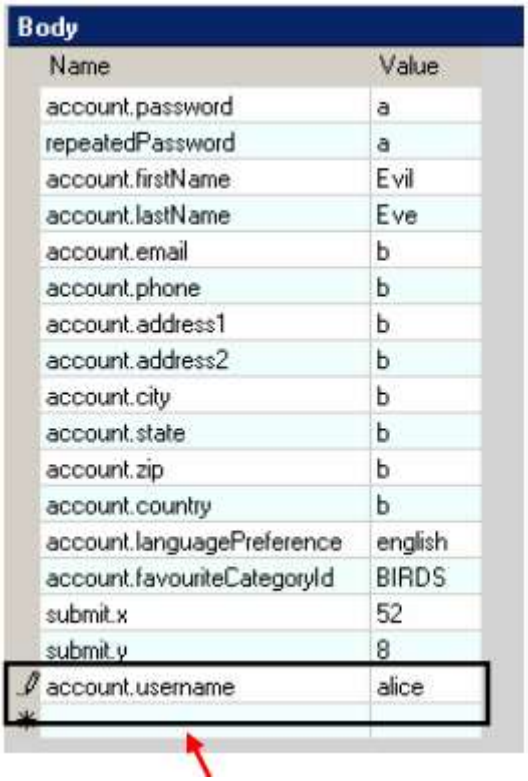


Gambar 4.23 Field-Field pada Java form[1]

Ini adalah metode yang digunakan untuk mem-bind atau mengikat nilai yang telah disediakan oleh field account.username yang HTTP ke dalam internal username field. Pada contoh ini eve sedang mengubah user detailnya di mana seharusnya field ini tidak bisa diubah. Karena auto binding HTMLform data ke dalam internal Class, hal ini sangat memungkinkan untuk mensumbmit data ke semua setter yang tersedia. Untuk mensubmit data ke setUsername (), hal

yang perlu untuk dilakukan adalah menambahkan form HTML untuk pengisian data ke form submission data : account.username = {data}.

Hal ini dapat dengan mudah dicegat dengan menggunakan web proxy seperti Fiddler.

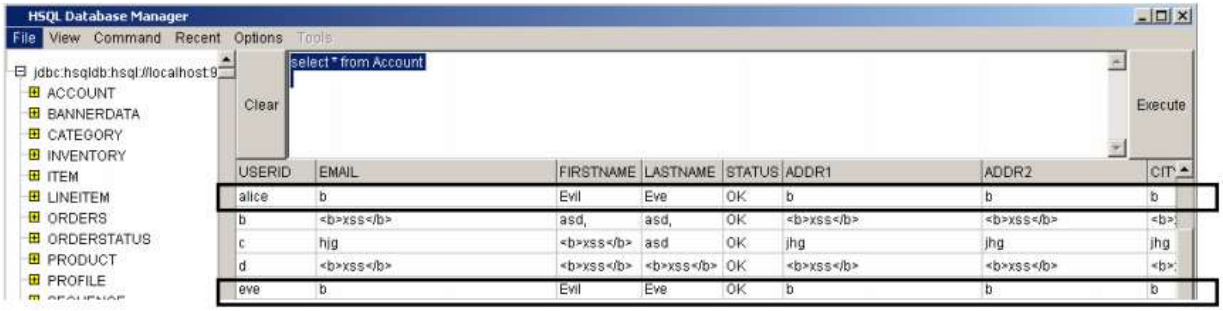


Gambar 4.24 Field-Field pada web proxy fiddler [1]

Tambahkan field extra seperti pada gambar 4.24 dan biarkan request berjalan sampai selesai.

Gambar 4.25 web proxy fiddler ketika dijalankan [1]

Tampilan pada database berikut menegaskan bahwa telah terjadi perubahan-perubahan pada semua field kecuali field username :



Gambar 4.26 Perubahan Pada database [1]

Jika kita melakukan login menggunakan akun dari Alice maka data Account Information dari Alice akan berubah menjadi Evil Eve.

V. PENUTUP

5.1 Kesimpulan

Dari pembahasan yang telah dilakukan sebelumnya, berikut ini adalah beberapa point yang bisa disimpulkan :

1. Ada dua celah keamanan yang diperoleh pada Framework Spring MVC data, yaitu *Spring MVC Data Submission to Non-Editable Fields* dan *Spring MVC ModelView Injection*.
2. Kerentanan yang terdapat pada Framework Spring MVC java sebenarnya ada ketika pada field – field pada form yang dibuat. Form form tersebut memiliki celah sehingga para developer harus waspada akan masukkan sesuatu yang berbahaya pada form. Kerentanan yang kedua adalah melalui view. Dengan View ini, pera penyerang bisa masuk dan mengakses sumber daya yang seharusnya tidak bisa dimasuki.

5.2 Saran

1. Jangan pernah membiarkan user untuk meresolve view. Validasi tidak bisa digunakan untuk mencegahnya karena penyerang menggunakan name view yang valid untuk mem-bypass bisnis proses.

DAFTAR PUSTAKA

[1] Ryan Berg and Dinis Cruz (2008). "*Two Security Vulnerabilities in The Spring Framework's MVC*". Advance Research Team Technical Advisory.

[2] Viega J, Mutdosch T, W Edward (2000). "*Statically Scanning Java Code : Finding Security Vulnerabilities"*. IEEE Software, (7) : .http://ieeexplore.ieee.org/ielx5/52/6156700/06156713.pdf (diakses tanggal 1 Mei 2016).

[3] Dandan Zang, Zhiqiang Wei, Yongquan Yang (2013). "*Research on Lightweight MVC Framework Based on Spring MVC and Mybatis*". 2013 Sixth International Symposium on Computational Intelligence and Design, (4) : . http://ieeexplore.ieee.org/ielx7/6804196/6804763/06805007.pdf (diakses tanggal 1 Mei 2016)

[4] *Spring MVC Known Vulnerabilities and Issues*. http://support.springsource.com/security/spring-mvc .

[5] *Remote code vulnerability in Spring Framework for Java*. http://www.infosecurity-magazine.com/news/remote-code-vulnerability-in-spring-framework-for/

[6] Basari Khoirul, M dan MS Sutanto Himawan. (2013). "*MVC Framework Java : Spring*". Makalah pada Prodi Teknik Informatika Universitas Komputer Indonesia, Bandung.

[7] https://fendiyuniawanaditya.wordpress.com/2010/01/15/jenis-jenis-framework-java/

[8] http://docs.spring.io/spring-framework/docs/3.0.x/reference/overview.html