

# How to Break Odoo's Security

*(or how to prevent it :-))*

**Olivier DONY** · Platform & Security

security@odoo.com - @odony

# Odoo Security Team

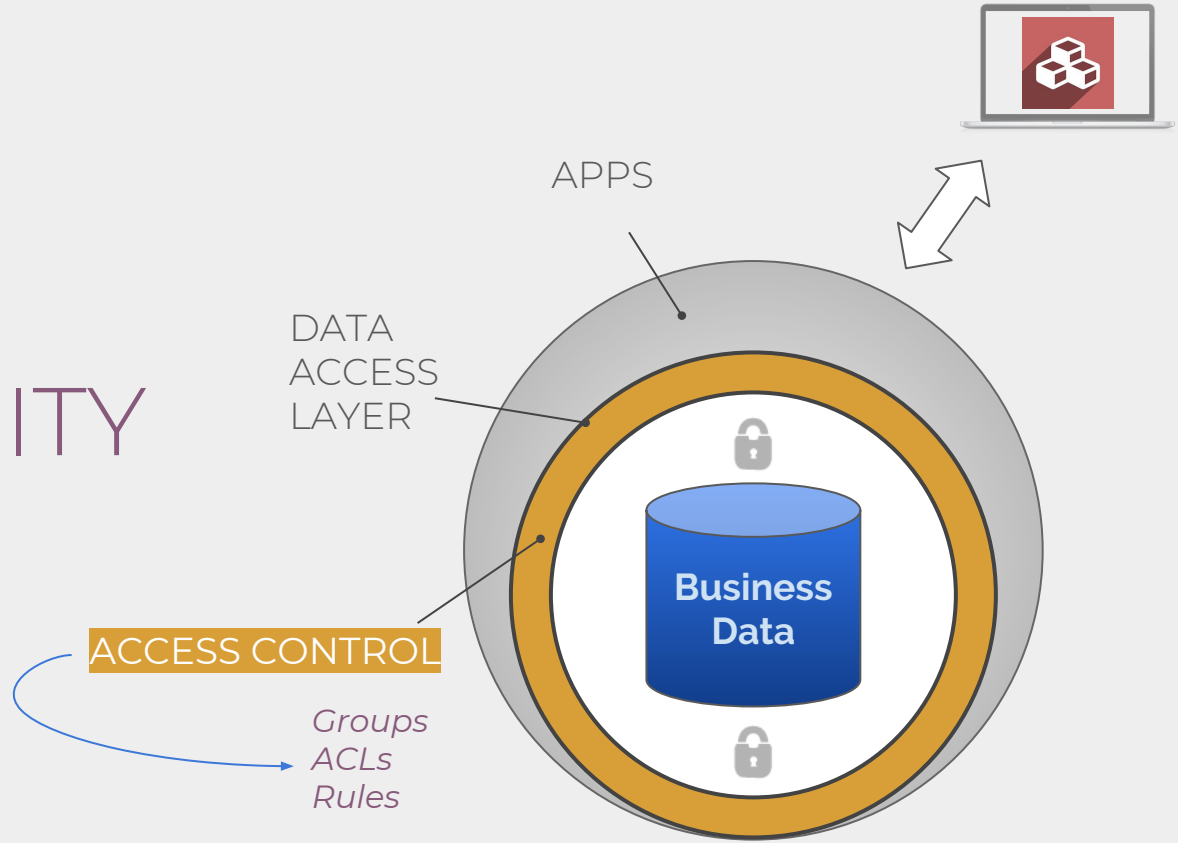


- ★ Audit / Review source code
- ★ Analyze customer audits
- ★ Raise awareness
- ★ Monitor [security@odoo.com](mailto:security@odoo.com)
- ★ Responsible Disclosure Process
- ★ Security Advisories

[odoo.com/security-report](https://odoo.com/security-report)

- ★ Now an Official CVE Numbering Authority (CNA)! 🧐

# THE SECURITY MODEL



# Multi-Level Access Control

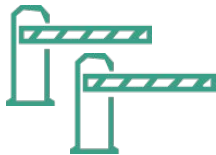


Per-field @groups

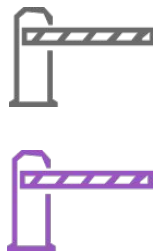


data

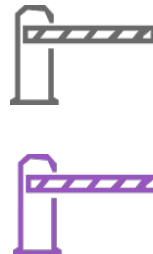
ir.rule  
(global)



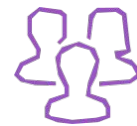
ir.rule  
(groups)



ir.model.access  
(groups)



GroupA



GroupB



User

User wants to  
access data

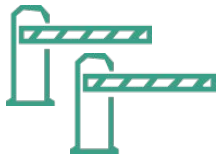
# Multi-Level Access Control



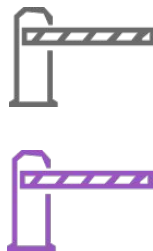
Per-field @groups



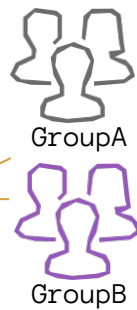
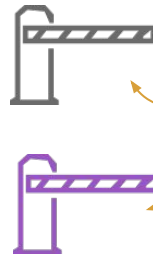
ir.rule  
(global)



ir.rule  
(groups)



ir.model.access  
(groups)



User wants to  
access data

# Multi-Level Access Control



Per-field @groups

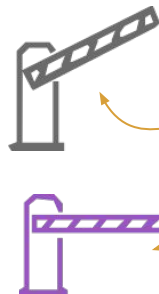
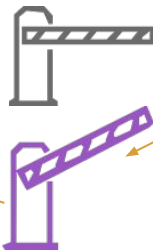
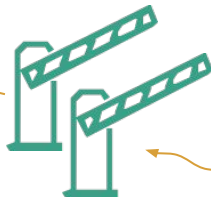
ir.rule  
(global)

ir.rule  
(groups)

ir.model.access  
(groups)



data



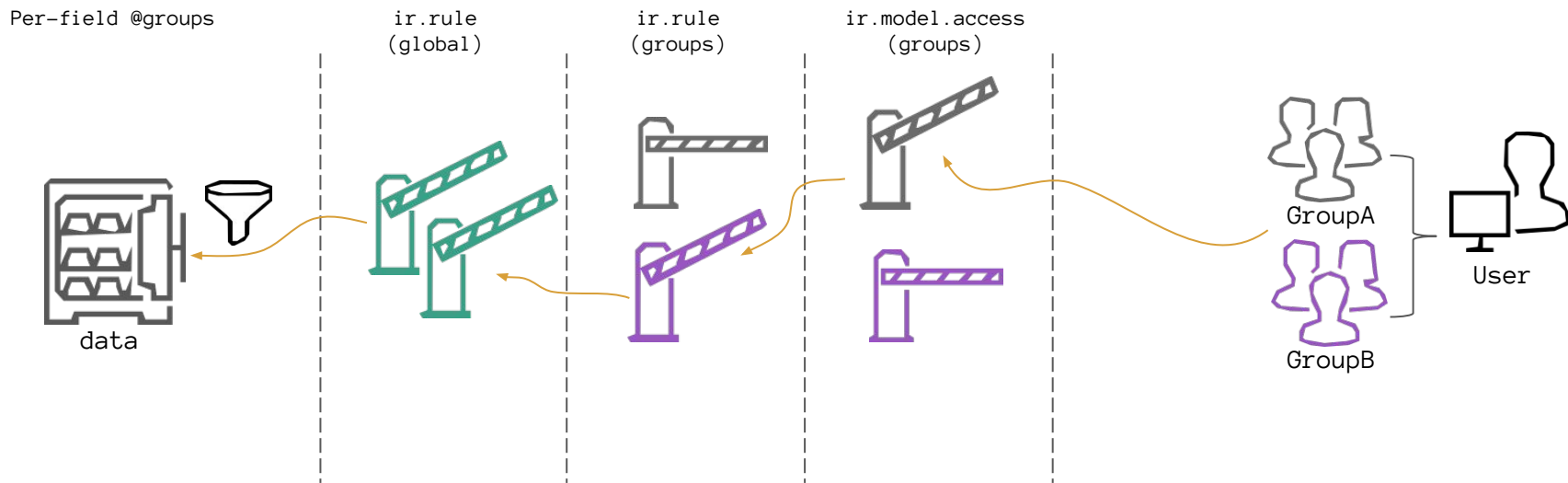
GroupA

GroupB



User

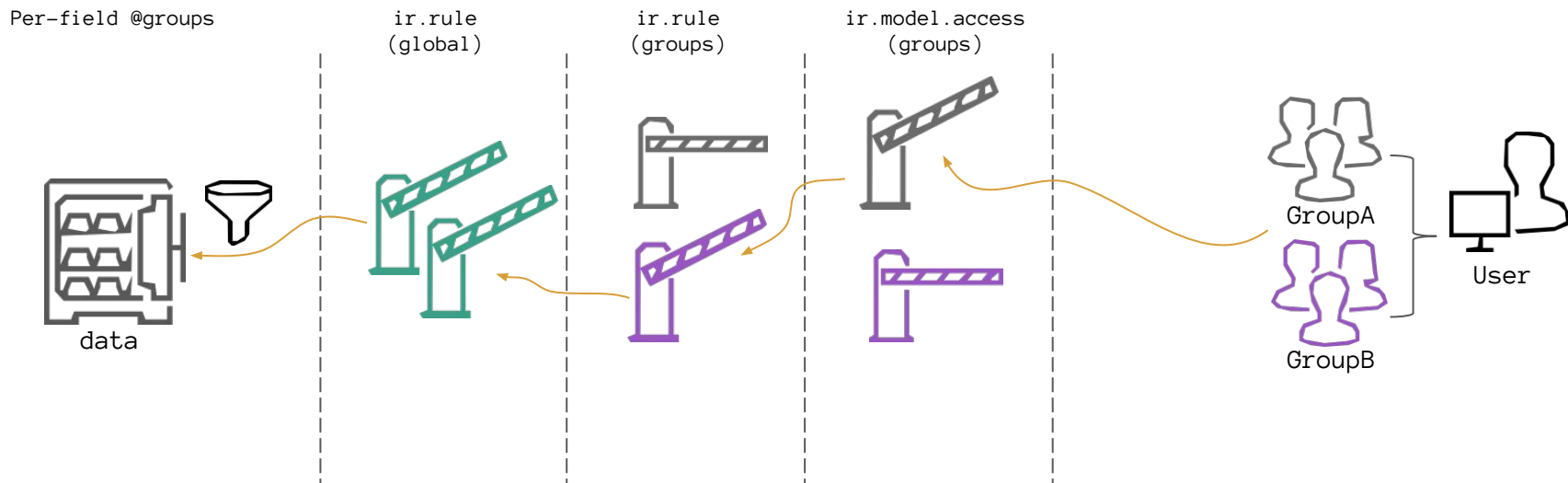
# Multi-Level Access Control



Each layer controls separately the 4 CRUD operations

☐ Create   ☐ Read   ☐ Update   ☐ Delete

# Multi-Level Access Control



These layers controls separately the 4 **CRUD** operations

☐ Create ☐ Read ☐ Update ☐ Delete

a.k.a **CRWU**

Create  
Read  
Write  
Unlink



# Correct ACLs are **key** for securing an Odoo App

fleet/  
controllers/  
data/  
models/  
views/  
security/  
ir.model.access.csv  
security.xml

fleet.vehicle

fleet.contract

model	group	C	R	W	U
fleet.vehicle	base.group_user	0	1	0	0
fleet.vehicle	fleet.group_manager	1	1	1	1
fleet.contract	fleet.group_manager	1	1	1	1



```
<record model="res.groups" id="group_manager">  
  <field name="name">Fleet Manager</field>  
</record>
```



```
<record model="ir.rule" id="rule_fleet_user">  
  <field name="model_id" ref="model_fleet_vehicle"/>  
  <field name="groups" eval="[(4, ref('base.group_user'))]"/>  
  <field name="domain_force">  
    [('driver_id', '=', user.id)]  
  </field>  
</record>
```

Injection  
(SQL, Code, ...)

Broken Auth  
(Sessions/Cred)

Access Control  
(Wrong checks)

Cross-Site  
Request  
Forgery  
(CSRF/XSRF)

Information  
Leaks

Security  
Misconfiguration

Cross-site  
scripting (XSS)



# Common vulnerabilities? Covered.



*So, how do we break in?*

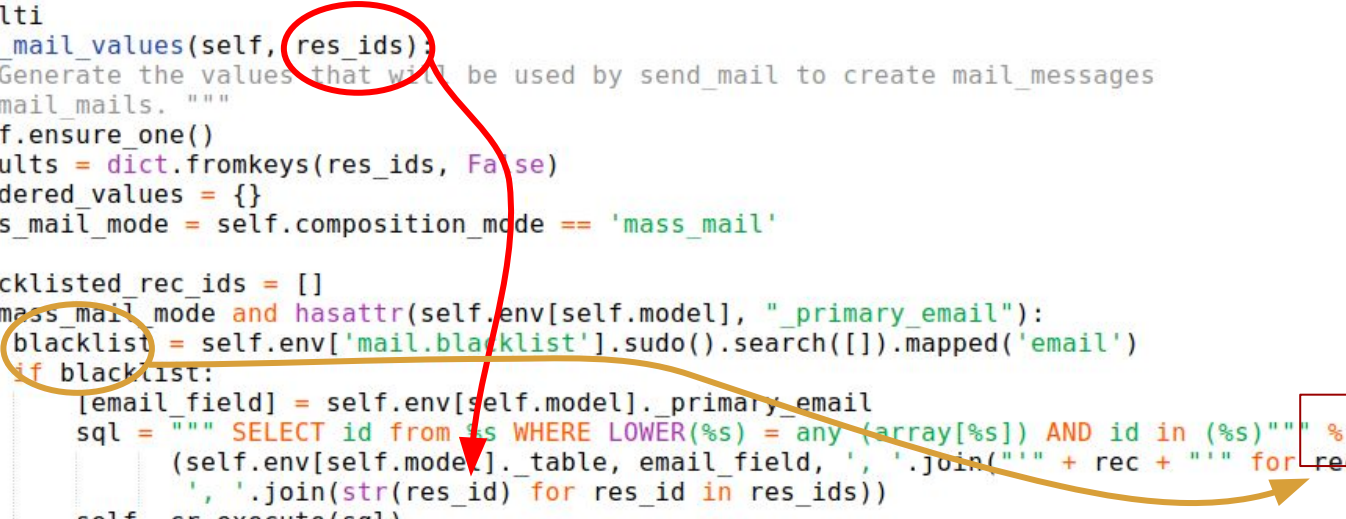
# Breaks security? (1)

```
264 @api.multi
265 def get_mail_values(self, res_ids):
266     """Generate the values that will be used by send_mail to create mail_messages
267     or mail_mails. """
268     self.ensure_one()
269     results = dict.fromkeys(res_ids, False)
270     rendered_values = {}
271     mass_mail_mode = self.composition_mode == 'mass_mail'
272
273     blacklisted_rec_ids = []
274     if mass_mail_mode and hasattr(self.env[self.model], "_primary_email"):
275         blacklist = self.env['mail.blacklist'].sudo().search([]).mapped('email')
276         if blacklist:
277             [email_field] = self.env[self.model]._primary_email
278             sql = """ SELECT id from %s WHERE LOWER(%s) = any (array[%s]) AND id in (%s)""" % \
279                 (self.env[self.model]._table, email_field, ', '.join("'" + rec + "'" for rec in blacklist),
280                  ', '.join(str(res_id) for res_id in res_ids))
281             self._cr.execute(sql)
282             blacklisted_rec_ids = [rec[0] for rec in self._cr.fetchall()]
283
284     # (...)
```



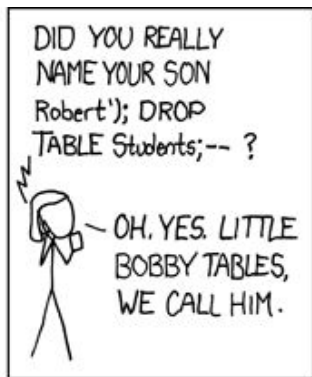
# Breaks security. (1)

```
264 @api.multi
265 def get_mail_values(self, res_ids):
266     """Generate the values that will be used by send_mail to create mail_messages
267     or mail_mails. """
268     self.ensure_one()
269     results = dict.fromkeys(res_ids, False)
270     rendered_values = {}
271     mass_mail_mode = self.composition_mode == 'mass_mail'
272
273     blacklisted_rec_ids = []
274     if mass_mail_mode and hasattr(self.env[self.model], "_primary_email"):
275         blacklist = self.env['mail.blacklist'].sudo().search([]).mapped('email')
276         if blacklist:
277             [email_field] = self.env[self.model]._primary_email
278             sql = """ SELECT id from %s WHERE LOWER(%s) = any (array[%s]) AND id in (%s)""" % \
279                 (self.env[self.model]._table, email_field, ', '.join("'" + rec + "'" for rec in blacklist),
280                  ', '.join(str(res_id) for res_id in res_ids))
281             self._cr.execute(sql)
282             blacklisted_rec_ids = [rec[0] for rec in self._cr.fetchall()]
283
284     # (...)
```



**SQL Injection!**

# Bobby Tables...



# Breaks security? (2)



```
24 @api.model_create_multi
25 def create(self, values):
26     """ To avoid crash during import due to unique email, return the existing records if any """
27     sql = '''SELECT id, email FROM mail_blacklist
28           WHERE LOWER(email) = any (array[%s])
29           ''' % (' , '.join(['%s'] * len(values)))
30     params = [value['email'].lower() for value in values]
31     self._cr.execute(sql, params)
32     records = self._cr.fetchall()
33
34     bl_ids = bl_emails = []
35     if records:
36         bl_ids, bl_emails = list(izip(*records))
37     non_blacklisted_records = [value for value in values if value['email'] not in bl_emails]
38
39     results = super(MailBlackList, self).create(non_blacklisted_records)
40     return self.env['mail_blacklist'].browse(bl_ids) | results
41
```

← Ugly but not injectable

**NOPE, but close!**

# Breaks security? (3)

```
6 class PhotoFolderShare(models.Model):
7     _name = 'photo.folder.share'
8
9     name = fields.Char()
10    folder_id = fields.Many2one('photo.folder', required=True)
11    access_token = fields.Char(default=lambda x: str(uuid.uuid4()))
12    url = fields.Char(compute='_compute_url')
13
14    @api.multi
15    @api.onchange('access_token')
16    def _compute_url(self):
17        base_url = self.env["ir.config_parameter"].sudo().get_param('photo.folder.share.base_url')
18        for record in self:
19            record.url = "{}/photos/share/{}/{}".format(base_url, record.folder_id.id, record.access_token)
20
```

model	group	C	R	W	U
photo.photo	base.group_user	✓	✓	✓	✗
photo.photo	base.group_portal 	✗	✓	✗	✗
photo.photo	photos.group_manager	✓	✓	✓	✓
photo.folder	base.group_user	✗	✓	✗	✗
photo.folder	base.group_portal 	✗	✓	✗	✗
photo.folder	photos.group_manager	✓	✓	✓	✓
photo.share	base.group_user	✓	✓	✓	✓
photo.share	base.group_portal	✗	✓	✗	✗

```
90 @http.route(['/photos/download/<int:share_id>/<access_token>'], type='http', auth='public')
91 def share_download(self, share_id=None, access_token=None, **kwargs):
92     env = request.env
93     share = env['photo.folder.share'].sudo().browse(share_id)
94     if access_token == share.access_token:
95         return self._make_zip(share.name, share.folder_id.photos)
96     return request.not_found()
```



# Breaks security? (3)

```
6 class PhotoFolderShare(models.Model):
7     _name = 'photo.folder.share'
8
9     name = fields.Char()
10    folder_id = fields.Many2one('photo.folder', required=True)
11    access_token = fields.Char(default=lambda x: str(uuid.uuid4()))
12    url = fields.Char(compute='_compute_url')
13
14    @api.multi
15    @api.onchange('access_token')
16    def _compute_url(self):
17        base_url = self.env["ir.config_parameter"].sudo().get_param('photo.folder.share.base_url')
18        for record in self:
19            record.url = "{}/photos/share/{}/{}".format(base_url, record.folder_id, record.access_token)
20
```


model	group	C	R	W	U
photo.photo	base.group_user	✓	✓	✓	✗
photo.photo	base.group_portal	✗	✓	✗	✗
photo.photo	photos.group_manager	✓	✓	✓	✓
photo.folder	base.group_user	✗	✓	✗	✗
photo.folder	base.group_portal	✗	✓	✗	✗
photo.folder	photos.group_manager	✓	✓	✓	✓
photo.share	base.group_user	✓	✓	✓	✓
photo.share	base.group_portal	✗	✓	✗	✗

```
90 @http.route(["/photos/download/<int:share_id>/<access_token>"], type='http', auth='public')
91 def share_download(self, share_id=None, access_token=None, **kwargs):
92     env = request.env
93     share = env['photo.folder.share'].sudo().browse(share_id)
94     if access_token == share.access_token:
95         return self._make_zip(share.name, share.folder_id.photos)
96     return request.not_found()
```


**ACCESS  
CONTROL!**

# Breaks security? (3)

```
6 class PhotoFolderShare(models.Model):
7     _name = 'photo.folder.share'
8
9     name = fields.Char()
10    folder_id = fields.Many2one('photo.folder', required=True)
11    access_token = fields.Char(default=lambda self: str(uuid.uuid4()), groups="base.group_user")
12    url = fields.Char(compute=_compute_url)
13
14    @api.multi
15    @api.onchange('access_token')
16    def _compute_url(self):
17        base_url = self.env["ir.config_parameter"].sudo().get_param("web.base.url")
18        for record in self:
19            record.url = "{}{/photos/share/{}/{}".format(base_url, record.id, record.access_token)
20
```



```
90 @http.route(["/photos/download/<int:share_id>/<access_token>"], type='http', auth='public')
91 def share_download(self, share_id=None, access_token=None, **kwargs):
92     env = request.env
93     share = env['photo.folder.share'].sudo().browse(share_id)
94     if consteq(access_token, share.access_token):
95         return self._make_zip(share.name, share.folder_id.photos)
96     return request.not_found()
```



# Breaks security? (4)

 / Your Details

## Your Details

**Your Name**

Joel Willis

**Email**

joel.willis63@example.com

**Phone**

(683)-556-5104

**Company Name**

YourCompany

Confirm

# Breaks security? (4)

```
351 <template id="portal_my_details">
352   <t t-call="portal.portal_layout">
353     <h3>Your Details</h3>
354     <form action="/my/account" method="post">
355       <div class="row o_portal_details">
356         <div class="row">
357           <div class="col-lg-12">
358             <div class="form-group col-xl-6">
359               <label class="col-form-label" for="name">Your Name</label>
360               <input type="text" name="name" class="form-control" t-att-value="partner.name" />
361             </div>
362             <div class="form-group col-xl-6">
363               <label class="col-form-label" for="email">Email</label>
364               <input type="email" name="email" class="form-control" t-att-value="partner.email" />
365             </div>
366             <div class="form-group col-xl-6">
367               <label class="col-form-label" for="phone">Phone</label>
368               <input type="tel" name="phone" class="form-control" t-att-value="partner.phone" />
369             </div>
370             <div class="form-group col-xl-6">
371               <label class="col-form-label label-optional"
372                 for="company_name">Company Name</label>
373               <input type="text" name="company_name" class="form-control"
374                 t-att-value="partner.company_name"/>
375             </div>
376             <button type="submit" class="btn btn-primary float-right mb32 ">
377               Confirm
378             </button>
379           </div>
380         </div>
381       </div>
382     </form>
383   </t>
384 </template>
```

```
150
151 @route(['/my/account'], type='http', auth='user', website=True)
152 def account(self, redirect=None, **post):
153     partner = request.env.user.partner_id
154     if post:
155         partner.sudo().write(post)
156         return request.redirect('/my/account')
157
158 values = self._prepare_portal_layout_values()
159 values['partner'] = request.env.user.partner_id
160 response = request.render("portal.portal_my_details", values)
161 response.headers['X-Frame-Options'] = 'DENY'
162 return response
```

# Breaks security? (4)

```
351 <template id="portal_my_details">
352   <t t-call="portal.portal_layout">
353     <h3>Your Details</h3>
354     <form action="/my/account" method="post">
355       <input type="hidden" name="csrf_token" t-att-value="request.csrf_token()"/>
356       <div class="row o_portal_details">
357         <div class="row">
358           <div class="col-lg-12">
359             <div class="form-group col-xl-6">
360               <label class="col-form-label" for="name">Your Name</label>
361               <input type="text" name="name" class="form-control" t-att-value="partner.name" />
362             </div>
363             <div class="form-group col-xl-6">
364               <label class="col-form-label" for="email">Email</label>
365               <input type="email" name="email" class="form-control" t-att-value="partner.email" />
366             </div>
367             <div class="form-group col-xl-6">
368               <label class="col-form-label" for="phone">Phone</label>
369               <input type="tel" name="phone" class="form-control" t-att-value="partner.phone" />
370             </div>
371             <div class="form-group col-xl-6">
372               <label class="col-form-label label-optional"
373                 for="company_name">Company Name</label>
374               <input type="text" name="company_name" class="form-control"
375                 t-att-value="partner.company_name"/>
376             </div>
```

```
150
151 @route(['/my/account'], type='http', auth='user', website=True)
152 def account(self, redirect=None, **post):
153     partner = request.env.user.partner_id
154     if post:
155         partner.sudo().write(post)
156         return request.redirect('/my/account')
157
158 values = self._prepare_portal_layout_values()
159 values['partner'] = request.env.user.partner_id
160 response = request.render("portal.portal_my_details", values)
161 response.headers['X-Frame-Options'] = 'DENY'
162 return response
```



# Breaks security? (4)

```
150
151 @route(['/my/account'], type='http', auth='user', website=True)
152 def account(self, redirect=None, **post):
153     partner = request.env.user.partner_id
154     if post:
155         partner.sudo().write(post)
156     return request.redirect('/my/account')
157
158 values = self._prepare_portal_layout_values()
159 values['partner'] = request.env.user.partner_id
160 response = request.render("portal.portal_my_details", values)
161 response.headers['X-Frame-Options'] = 'DENY'
162 return response
```

Accepts arbitrary fields, not just those in the form: unsafe **sudo**!!

**ACCESS  
CONTROL!**

# Breaks security? (5)

```
98 @http.route('/sale/message_history', type='json', auth='public', website=True)
99 def order_history_search(self, order_id, access_token=False, domain=None):
100     env = request.env
101     order = env['sale.order'].sudo().browse(order_id)
102     if not consteq(order.access_token, access_token):
103         raise Forbidden()
104     force_domain = [('model', '=', 'sale.order'), ('res_id', '=', order_id)]
105     domain = (domain or []) + force_domain
106     Msg_sudo = env['mail.message'].sudo()
107     result = Msg_sudo.search(domain)
108     return result
```

Unsafe domain concat

Could receive **partial** domain: ['|', ('body', 'ilike', 'password'), '&']

**Result:** ['|', ('body', 'ilike', 'password'),  
&', ('model', '=', 'sale.order'),  
( 'res\_id', '=', res\_id)]

**DATA LEAK!**

# Breaks security? (5)

```
8 @http.route('/sale/message_history', type='json', auth='public', website=True)
9 def order_history_search(self, order_id, access_token=False, domain=None):
10     env = request.env
11     order = env['sale.order'].sudo().browse(order_id)
12     if not consteq(order.access_token, access_token):
13         raise Forbidden()
14     force_domain = [('model', '=', 'sale.order'), ('res_id', '=', order_id)]
15     domain = expression.AND([domain or [], force_domain])
16     Msg_sudo = env['mail.message'].sudo()
17     result = Msg_sudo.search(domain)
18     return result
```



Safe domain  
combination



# There's more...



<https://www.odoo.com/r/dbN>



<https://www.odoo.com/r/h3s>

# Merge Security Checklist

- New fields/models? -> ACLs to add? Sensitive fields?
- New methods? -> Private by default?
- **sudo**? -> Double-check scope - record leaks - args
- **t-raw**? -> Remove it quickly, unless it's a sanitized `Html` field
- **getattr**? -> Find an alternative, there should be one...
- **(safe)\_eval**? -> Triple-check! Not for parsing data, right?
- raw SQL? -> No %, **concat** or **format()**, right? Check again!



*And now imagine that you're an attacker...*



# Thank you.

Security concern? ➤ [security@odoo.com](mailto:security@odoo.com)



#odooexperience

Photos credits:

<https://www.flickr.com/photos/129153735@N02/>

<https://www.flickr.com/photos/fallentomato/>

[https://www.flickr.com/photos/popatito\\_feo/](https://www.flickr.com/photos/popatito_feo/)

<https://www.flickr.com/photos/medevac71/>