

Business Logic Security

Ilia Alshanetsky

@iliaa

whois: Ilia Alshanetsky

- ❖ PHP Core Developer since 2001
 - ❖ Release Master of 4.3, 5.1 and 5.2
- ❖ Author of “Guide to PHP Security”
- ❖ Author / Co-Author of many PHP extensions
- ❖ CIO @ Centah Inc.
- ❖ Occasional Photographer ;-)



The Usual Suspects

- ❖ Cross-Site Scripting (XSS)
- ❖ Cross-Site Request Forgery (CSRF)
- ❖ Code Injection
- ❖ SQL Injection
- ❖ Authentication Issues & Session Management
- ❖ Insecure Cryptographic Storage
- ❖ Insufficient Transport Layer Protection
- ❖ Unvalidated Redirects

The Usual Suspects

- ❖ Cross-Site Scripting (XSS)
- ❖ Cross-Site Request Forgery (CSRF)
- ❖ Code Injection
- ❖ SQL Injection
- ❖ Authentication Issues & Session Management
- ❖ Insecure Cryptographic Storage
- ❖ Insufficient Transport Layer Protection
- ❖ Unvalidated Redirects

**Common
Topics
for
Conversation,
but not Today**





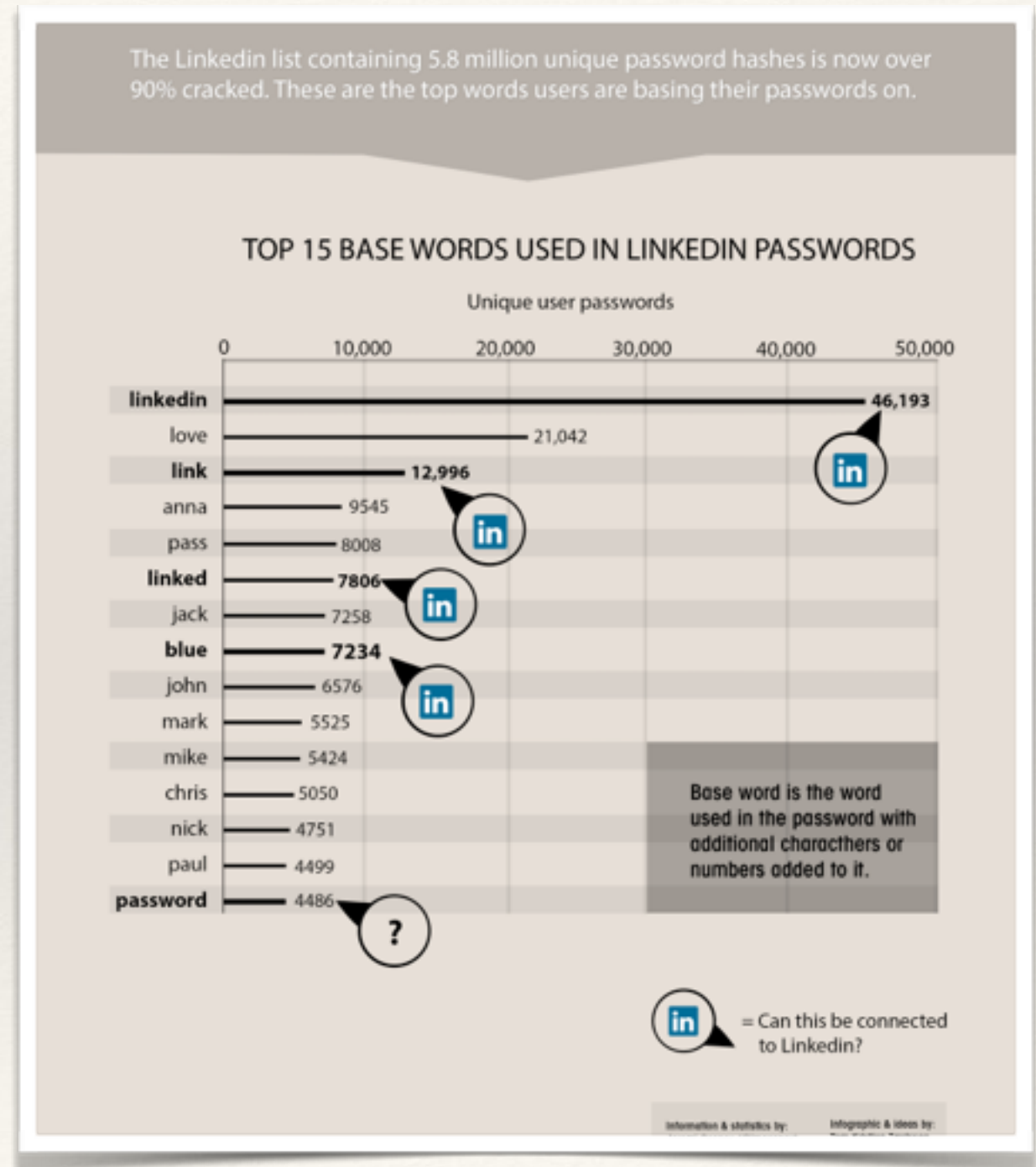
Authentication

Require Strong Passwords

- ❖ Require password length of 8 characters
- ❖ Enforce Password Complexity (3 of 4 rules):
 - ❖ At least one upper-case letter
 - ❖ At least one lower-case letter
 - ❖ At least one number
 - ❖ At least one special (non-alphanumeric) character

But even that is weak...

- ❖ Rainbow Tables
- ❖ GPU optimized hash guessing
- ❖ AWS ;-)



Secure Password Hashes

```
$password = "@foo1Bar#";
```

```
$passwd = crypt($password,  
    '$2y' . // BlowFish base  
    '$10$' . // cryptographic complexity  
    bin2hex(fread(fopen("/dev/urandom", "r"), 32)) // random bytes  
    . '$'  
);
```

This will generate a password hash 60 bytes long

```
if ($passwd === crypt($password, $passwd)) {  
    // password ok  
} else {  
    // password check failed  
}
```

PHP 5.5+ Makes This Simpler

```
$hash = password_hash($password,  
    PASSWORD_BCRYPT,  
    ['cost' => 10]  
);  
  
if (password_verify($password, $hash)) {  
    // password ok  
} else {  
    // password check failed  
}
```

Web Brute Force Attacks

- ❖ Limit the number of sequential unsuccessful attempts to 3 - 5
- ❖ After that implement one or more of the following:
 - ❖ Lockout future attempts for 10-15 minutes
 - ❖ Require entry of CAPTCHA for all further attempts
 - ❖ Require multi-factor authentication
 - ❖ SMS if you have phone number
 - ❖ E-mail if you don't
 - ❖ Security Questions

Web Brute Force Attacks

- ❖ Implement blocks for multiple failed authentication attempts from the same IP address
- ❖ Don't use the standard "login" and "password" form field names
- ❖ Re-authorize attempts when login is successful from an unknown IP address and / or Browser.
- ❖ If possible randomly generate the field names for authentication forms

Unpredictable Field Names

```
<?php
// secret key for encoding form fields
$_SESSION['__form_key'] = $secret =
    bin2hex(openssl_random_pseudo_bytes(16));
?>
<form>
Login: <input type="text"
name="<?= hash_hmac('md5', 'login', $secret); ?>" />
<br />Password: <input type="password"
name="<?= hash_hmac('md5', 'password', $secret); ?>" />
</form>
```

Processing

```
$secret = $_SESSION['__form_key'];  
$input = array();  
  
foreach ($field_names as $v) {  
    $hashed_name = hash_hmac('md5', $v, $secret);  
  
    if (isset($_POST[$hashed_name])) {  
        $input[$v] = $_POST[$hashed_name];  
    }  
}
```

Post Authentication Paranoia

- ❖ Ensure Session Expiry Times are enforced at 24 - 30 mins
- ❖ Idle time logout after 10 mins of in-activity (JavaScript)
- ❖ For long-term session require re-authentication for key actions
 - ❖ Profile Changes
 - ❖ E-Commerce activities
- ❖ Prevent duplicate logins

Click Jacking

- ❖ Make sure you have **X-Frame-Options** header (with **DENY** or **SAMEORIGIN** values)
- ❖ Avoid GET method to make requests (yes, this includes Ajax)

Transport Security

- ❖ Use HTTP-Strict-Transport-Policy to direct browser to use HTTPS
 - ❖ Does not work in IE, yet...
- ❖ Redirect to separate sub-domain after HTTP > HTTPS redirect and restrict cookies to that domain.

Apache:

Header always set Strict-Transport-Security
"max-age=31536000; includeSubDomains"

Nginx:

add_header Strict-Transport-Security "max-age=31536000;
includeSubDomains";



Session Security

Basic Protections

- ❖ Only use cookies

```
ini_set("session.use_only_cookies", true);
```

- ❖ Ensure session ID integrity

```
ini_set("session.entropy_file", "/dev/urandom");  
ini_set("session.entropy_length", "32");  
ini_set("session.hash_bits_per_character", 6);
```

- ❖ Use HTTPOnly cookies for session storage

```
ini_set("session.cookie_httponly", true);
```

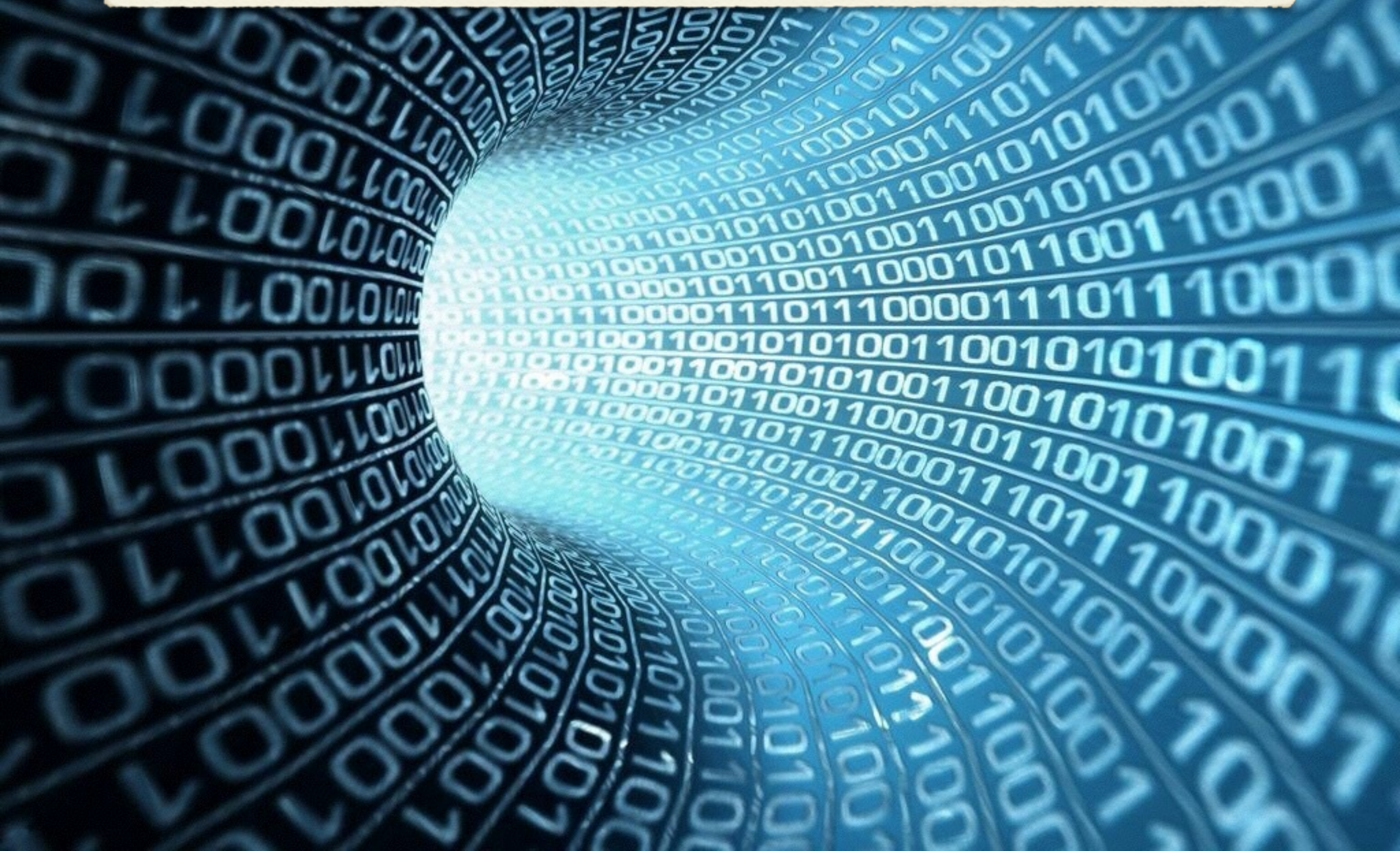
- ❖ Set Secure session bit (when using SSL/TLS)

```
ini_set("session.cookie_secure", true);
```

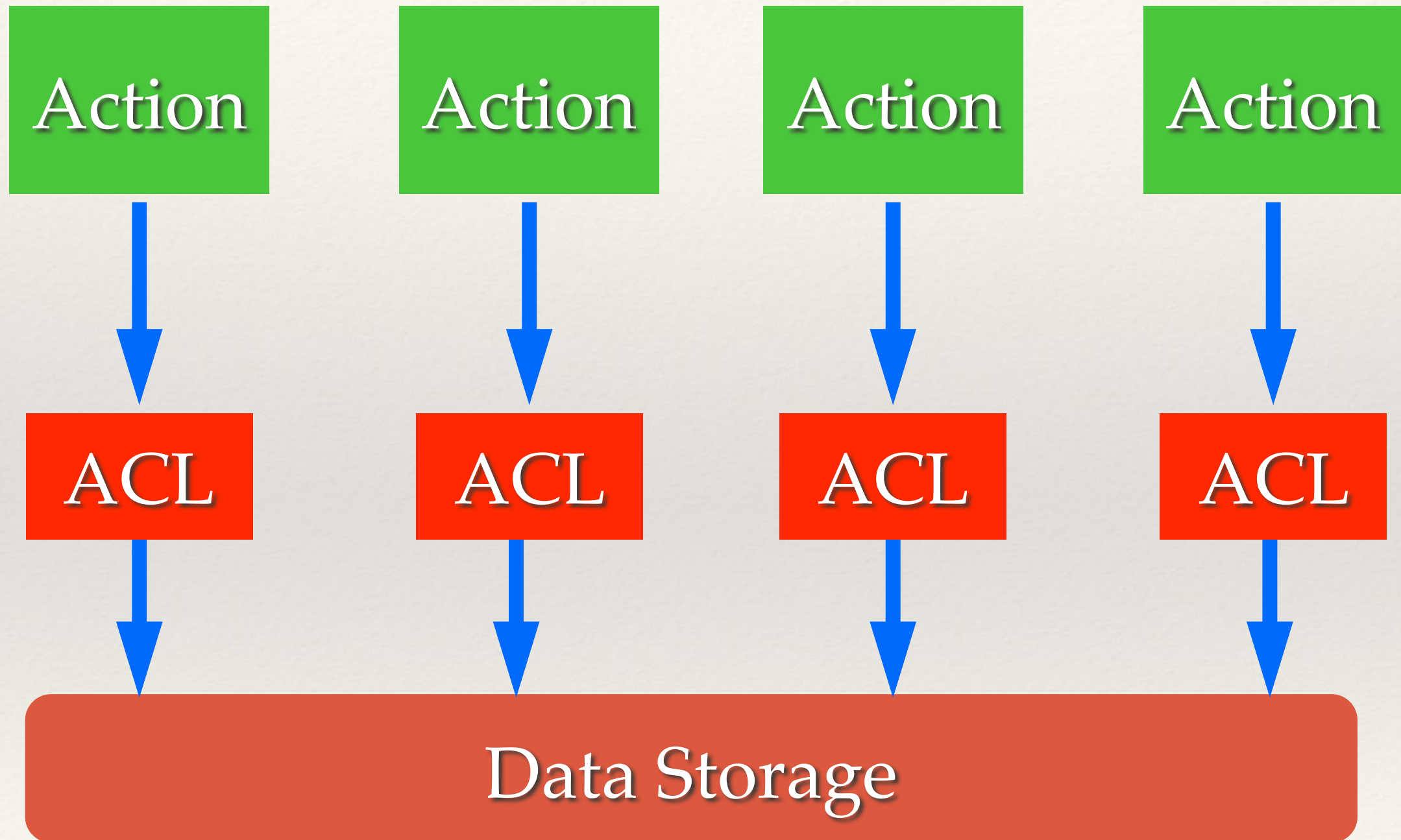
Avoid Session Fixation

```
ini_set("session.name", "unique name");  
  
session_start();  
  
if (empty($_SESSION['__validated'])) {  
    session_regenerate_id(true);  
    $_SESSION['__validated'] = 1;  
}
```

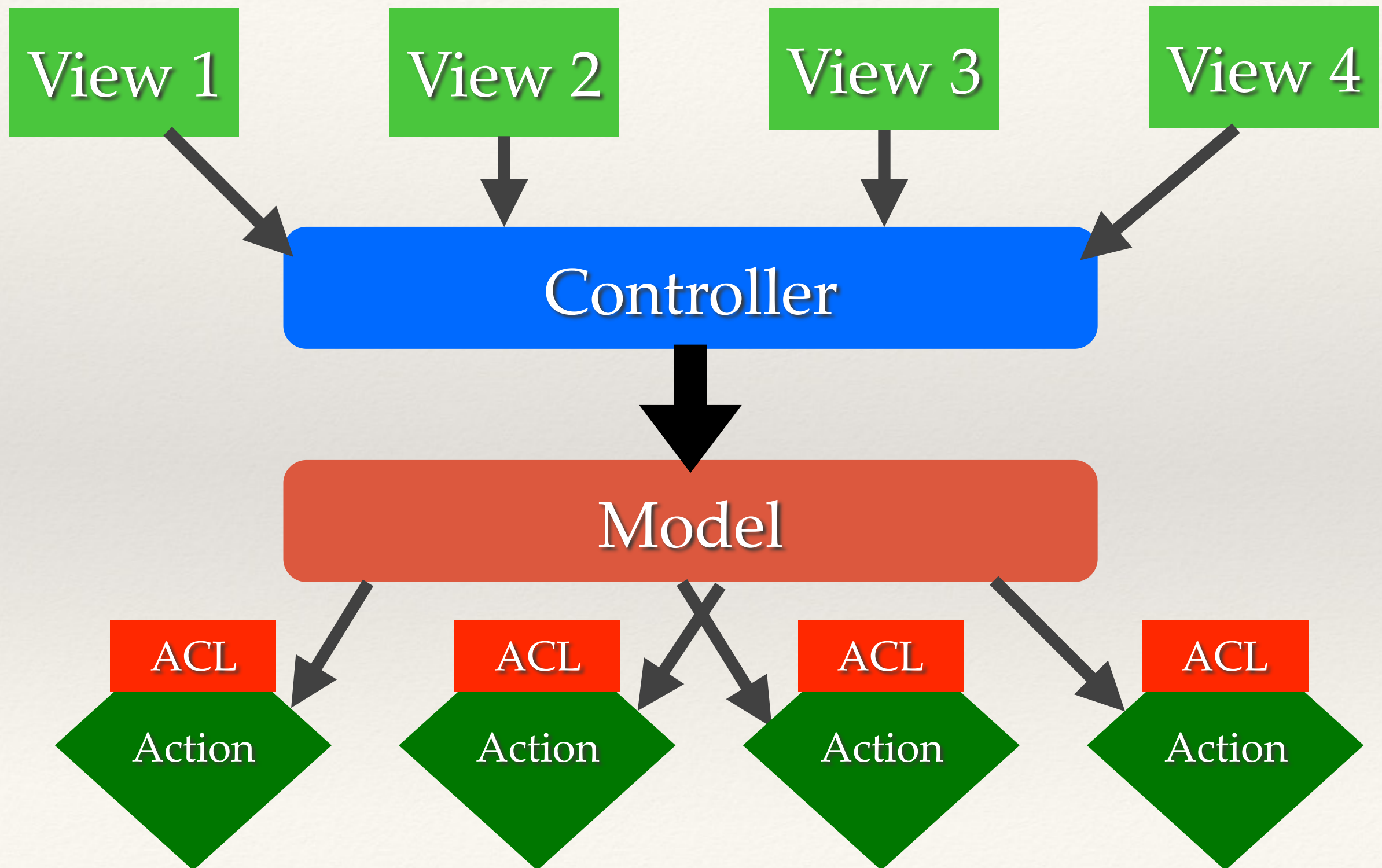

Data Access Management



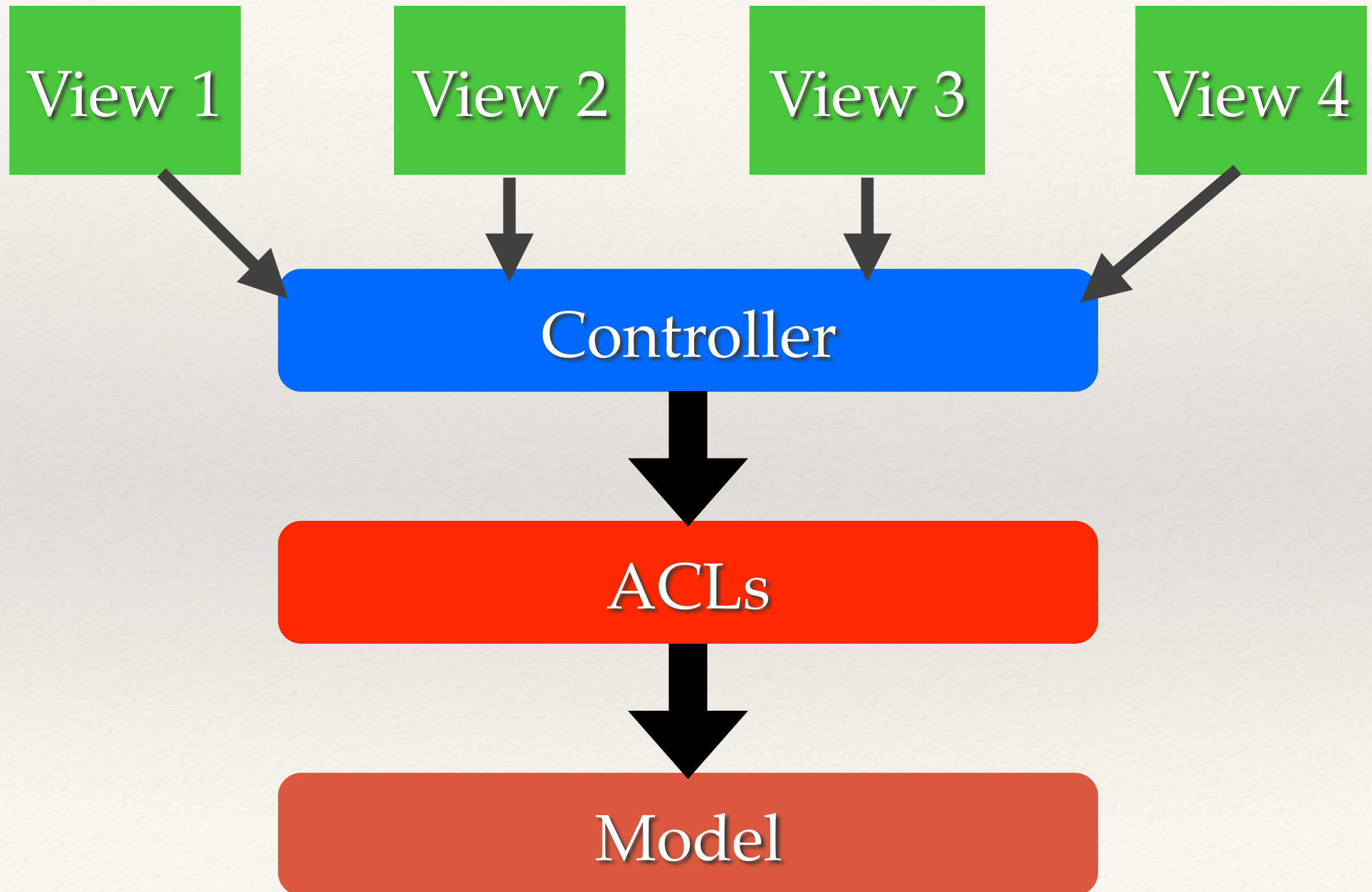
Typical Situation (pre-MVC)



Typical Situation (Post-MVC)



Ideal Approach



```
class DataModel {
    private $aclRules = array();

    public function __construct() {
        $this->aclRules['user_id'] = $_SESSION['user_id'];

        switch ($_SESSION['role']) {
            case 'admin':
                break;
            case 'user':
                $this->aclRules['public'] = 1;
                break;
            case 'editor':
                $this->aclRules['category'] = $_SESSION['category'];
                break;
        }
    }

    public function ActionName(array $params) {
        $input = array_replace_recursive($params, $this->aclRules);
        $this->runAction($input);
    }
}
```




Audit Trail

Why?

- ❖ Makes tracking down user activity easier when there is a security issue...
- ❖ All kinds of uses for debugging purposes
- ❖ Allows for pattern analysis for “unusual” activity detection
- ❖ Creates a “revert” path, versioning on the cheap

How?

- ❖ Should be done at the lowest level possible to avoid creating a possibility of un-audit-able actions.
- ❖ **Inside a Model**
- ❖ **Inside Database (via triggers)**

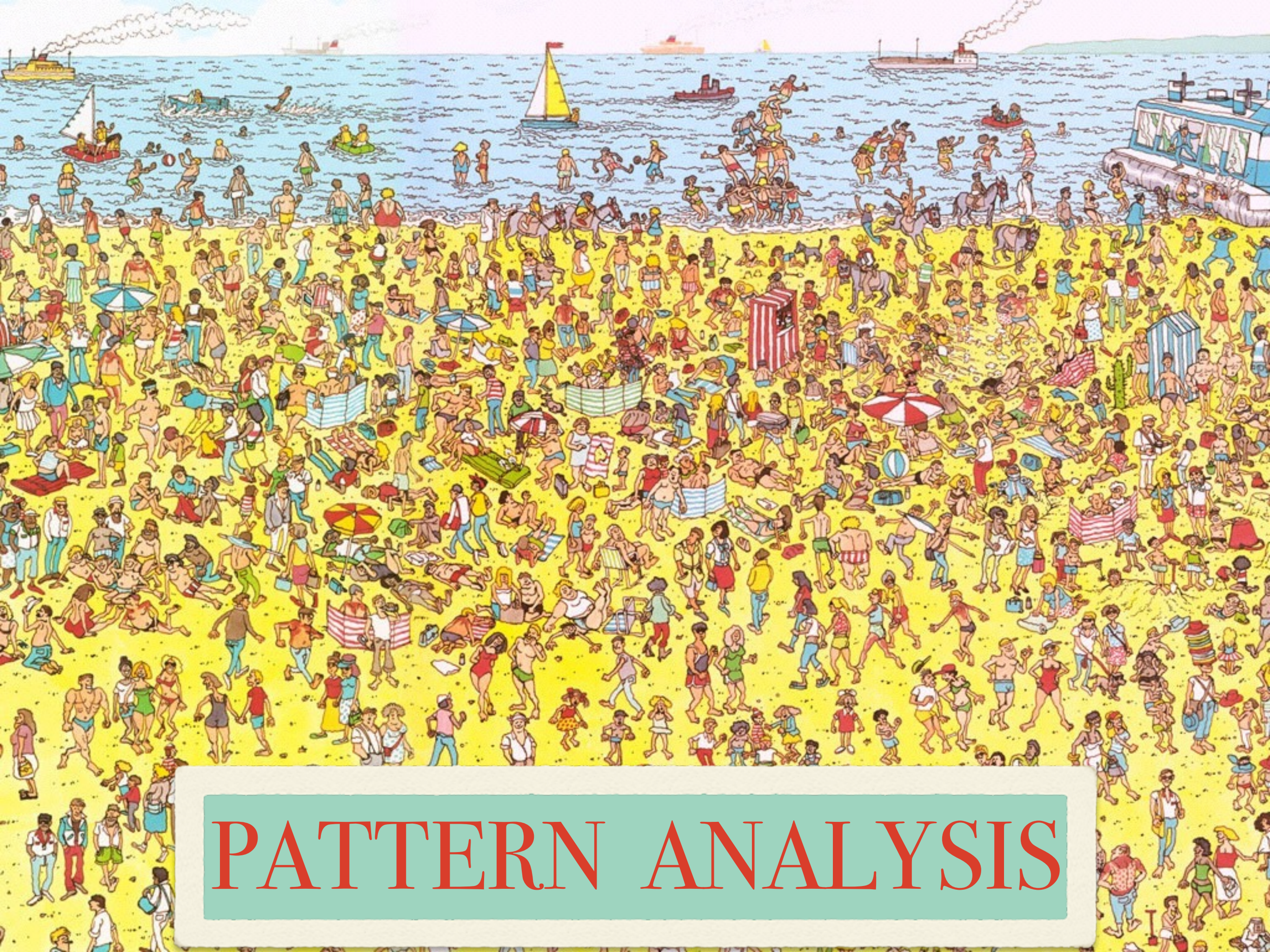
```
class DataModel {
    private function __save() {
        $current = $this->fetch($this->id);
        $changes = array_diff_assoc($this->input, $current);

        $this->pdo->beginTransaction();

        if (($return_val = parent::save())) {
            $this->log(array(
                'user_id'    => $_SESSION['user_id'],
                'when'       => microtime(1),
                'what'       => get_class($this),
                'record'     => $this->id,
                'changes'    => serialize($changes)
            ));

            $this->pdo->commit();
        } else {
            $this->pdo->rollback();
        }

        return $return_val;
    }
}
```

PATTERN ANALYSIS

What does it mean?

- ❖ The best application vulnerabilities are the ones no one knows about.
- ❖ But even those usually require some “trial & error” to get to
- ❖ Reviewing audit trails and access logs often can let you spot something “unusual” before even knowing what it is...

Patterns to Look For

- ❖ Unusually high number of request per session
- ❖ Atypical access pattern (late at night, different browser / IP combinations)
- ❖ Frequent accesses to same page within very short span of time, especially so if it is a data modification page.

ONE SHOULD NOT SIMPLY IGNORE



LOW LEVEL INPUT VALIDATION

Application should verify it's own inputs

Even at a model
level application
should verify
input for validity



```

class DataModel {
    private $input_config = array(
        'active' => array(
            'filter' => FILTER_VALIDATE_BOOLEAN,
            'flags' => FILTER_REQUIRE_SCALAR),
        'login' => array(
            'filter' => FILTER_VALIDATE_REGEXP,
            'flags' => FILTER_REQUIRE_SCALAR,
            'options' => array('regexp' => '!^[A-Za-z0-9_]+$!')),
        'id' => array(
            'filter' => FILTER_VALIDATE_INT,
            'flags' => FILTER_REQUIRE_SCALAR,
            'options' => array('min_range' => 1)),
        'email' => array(
            'filter' => FILTER_VALIDATE_EMAIL,
            'flags' => FILTER_REQUIRE_SCALAR),
        'blog' => array(
            'filter' => FILTER_VALIDATE_URL,
            'flags' => FILTER_REQUIRE_SCALAR)
    );

    public function save() {
        if (!filter_var_array($this->input, $this->input_config)) {
            throw new validationException('Invalid input');
        }
        // proceed as normal
    }
}

```


A close-up, high-contrast photograph of a man's face, likely from a horror movie. His face is covered in blood and gore, particularly around the eyes, nose, and mouth. He has a wide-eyed, open-mouthed expression, showing his teeth. The background is dark and blurry, suggesting an industrial or outdoor setting.

Remote URL Access

Things to Consider

- ❖ Whenever possible use the API URL sitting behind HTTPs
- ❖ Ensure that Peer and Domain verification is enabled
- ❖ If you are using cURL know what your settings mean...

Native PHP

```
$url = 'https://en.wikipedia.org/w/api.php ...';

$context = array(
    'ssl' => array(
        'verify_peer'    => TRUE,
        // wget http://curl.haxx.se/ca/cacert.pem
        'cafile'          => '/usr/share/ssl/cacert.pem',
        'verify_depth'    => 5,
        'CN_match'        => 'en.wikipedia.org'
    ),
    'http' => array(
        'user_agent' => 'My App',
        'ignore_errors' => TRUE
    )
);

file_get_contents($url, NULL, stream_context_create($context));
```

With cURL

```
$curlh = curl_init($url);
```

```
curl_setopt($curlh, CURLOPT_RETURNTRANSFER, TRUE);
```

```
curl_setopt($curlh, CURLOPT_CAINFO, '/usr/share/ssl/cert-bundle.crt');
```

```
$data = curl_exec($curlh);
```

- ❖ Do not set CURLOPT_SSL_VERIFYPEER to FALSE
- ❖ Do not set CURLOPT_SSL_VERIFYHOST to FALSE or 1



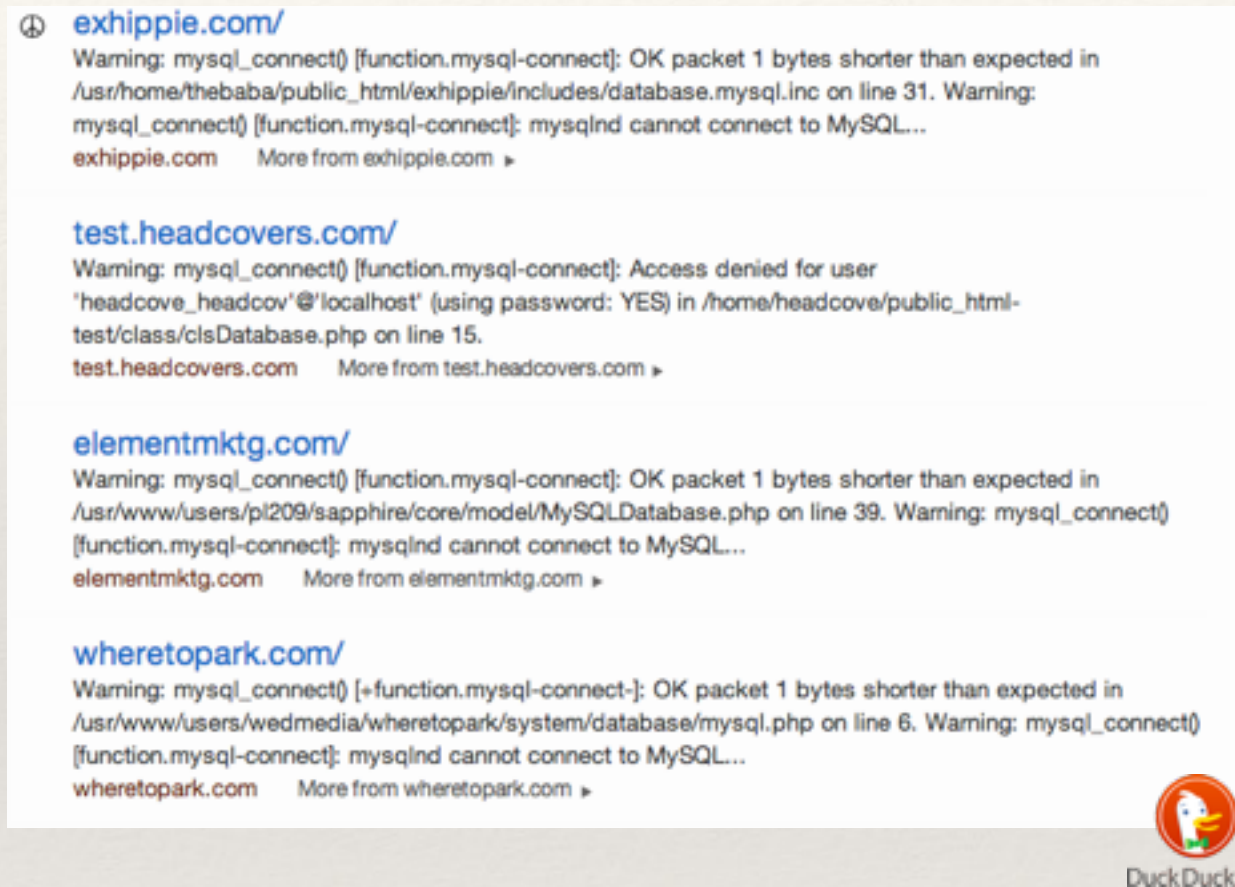
errors?

ERROR HANDLING

How to Handle Them?

- ❖ Log all errors
- ❖ Logging should not have dependencies
 - ❖ Disk is a good target
 - ❖ So is syslog
- ❖ There are no “trivial” errors

Disable Displaying of Errors!



```
ini_set("display_errors", false);
```

THANK YOU FOR
LISTENING

Ilia Alshanetsky

<http://ilia.ws>

@iliaa