

# Is Drupal secure?

A high-level perspective on web vulnerabilities, Drupal's solutions, and how to maintain site security

# Thinking Securely

“Security is a **process**, not an event.”

Security is cooperative:  
**It's everyone's responsibility.**

Security involves the **whole software stack**,  
not just Drupal (or any other application).

**Finding problems** is a good thing.

# The Top 10 Web Security Issues\*

...and how Drupal addresses them

\*According to The Open Web Application Security Project, 2007

**PROBLEM**

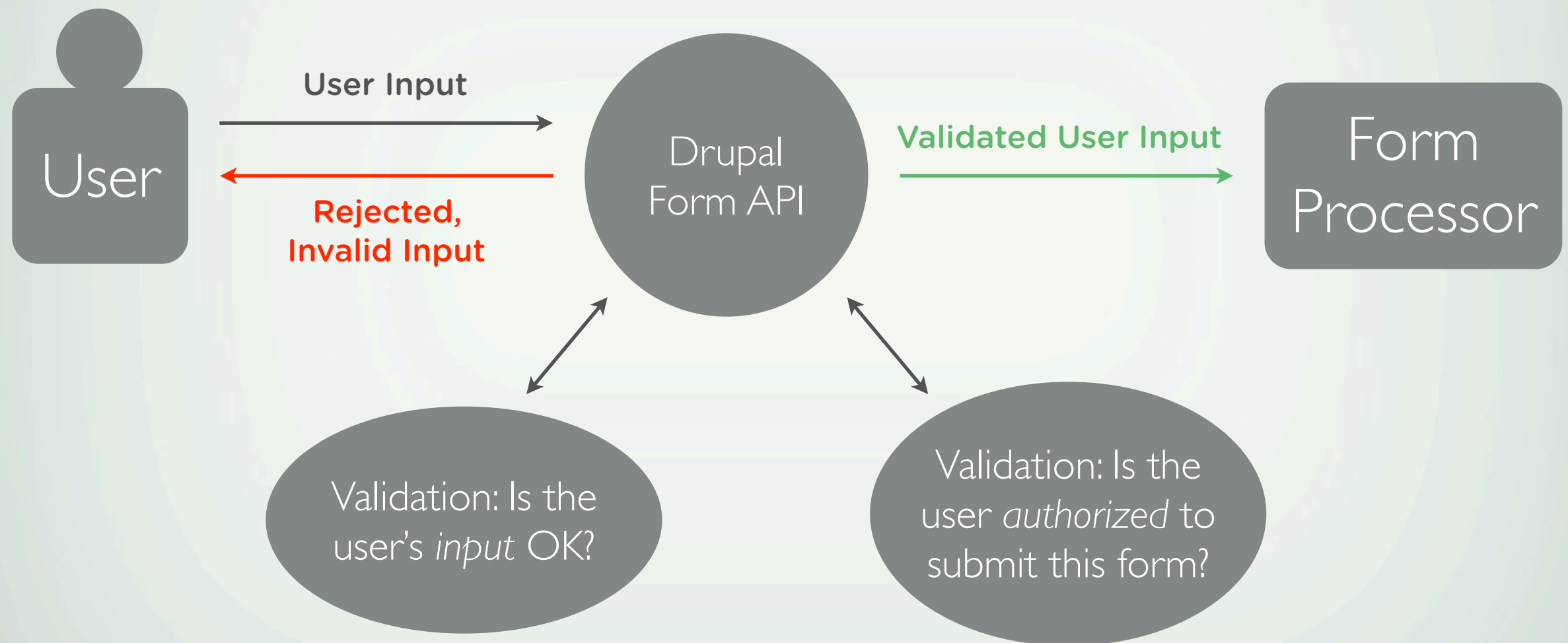
# Unvalidated input

When users submit information to sites, their input must be checked for validity.

**Note:** This issue made the top 10 in 2004 but not 2007.

How Drupal prevents...

# Unvalidated input

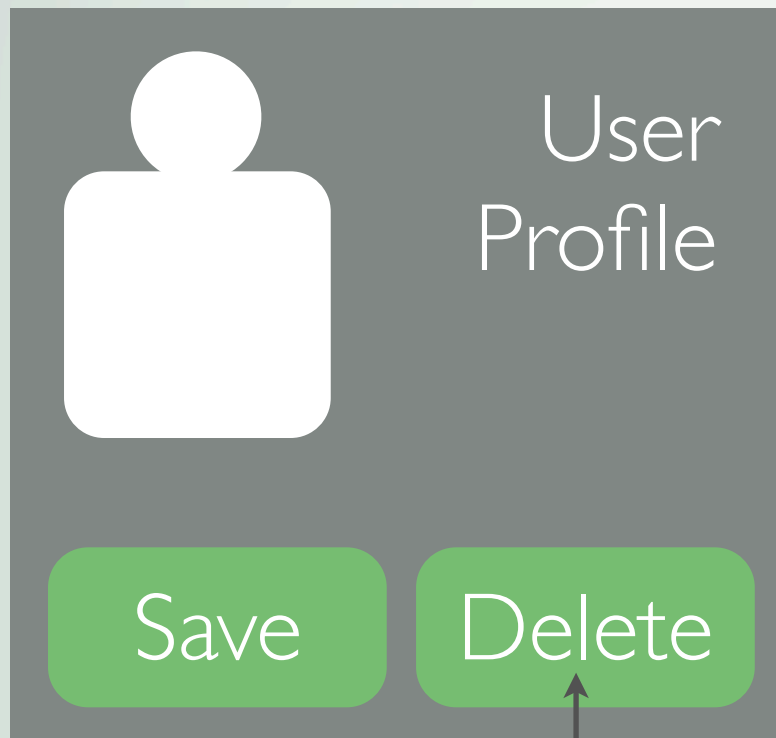


**PROBLEM**

# #10 Failure to Restrict URL Access

“Frequently, an application only protects sensitive functionality by preventing the display of links or URLs to unauthorized users. Attackers can use this weakness to access and perform unauthorized operations by accessing those URLs directly.”

# #10 Failure to Restrict URL Access



- 2 Enter a similar URL on a site that isn't yours:

`http://example.com/user/10/delete`

- 1 See where “delete” normally links.

- 3 What happens?

How Drupal prevents...

## #10 Failure to Restrict URL Access

- ▶ Drupal uses an integrated URL/access control system. Every URL in the system must have access control configured, even if that access is “allow everyone.”
- ▶ Drupal’s menu system associates link display with access, so direct URL entry rarely works if a link is not already visible.



**PROBLEM**

## #9 Insecure Communications

“Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications.”

# #9 Insecure Communications

http://example.com/user

↑

Insecure

Username

Administrator

Password

.....

Login

How Drupal prevents...

## #9 Insecure Communications

- ▶ As a PHP-based system, Drupal can use Apache's widely-trusted SSL support.
- ▶ If only part of the site is behind SSL, administrators can install modules to make certain URLs available only through a secure connection. Login, ecommerce, and administration page URLs often have this sort of security configured.

**PROBLEM**

# #8 Insecure Cryptographic Storage

“Web applications rarely use cryptographic functions properly to protect data and credentials. Attackers use weakly protected data to conduct identity theft and other crimes, such as credit card fraud.”

# #8 Insecure Cryptographic Storage



User table

Username	Password
Administrator	unguessable
Editor	1234
David	drupal

Encrypted  
or hashed?



How Drupal prevents...

## #8 Insecure Cryptographic Storage

- ▶ Passwords are stored using a one-way hash. Even if someone downloads the site database, recovering usable passwords is difficult.
- ▶ Drupal provides a randomly generated private key for every installation. Modules can use this key to use reversible encryption for sensitive data like credit-card numbers.
- ▶ Commerce modules for Drupal minimize *any* retention of sensitive data, even in encrypted form.

**PROBLEM**

# #7 Broken Authentication

“Account credentials and session tokens are often not properly protected. Attackers compromise passwords, keys, or authentication tokens to assume other users’ identities.”





# #7 Broken Authentication

## Site Cookie

Name	Value
SESS_1234	username=editor
administrator	TRUE
password	unguessable

Can users change their own access to the site?



Are authentication cookies broadcasting secrets to the world?



How Drupal prevents...

# #7 Broken Authentication

- ▶ Authentication cookies are not modifiable by site users. This prevents users from masquerading as more powerful users.
- ▶ User sessions (and related cookies) are completely destroyed and recreated on login and logout.
- ▶ User name, ID, and password are only managed on the server side, not in the user's cookie. Passwords are never emailed.
- ▶ Session cookies are named uniquely for each Drupal installation and strongly restricted by domain, limiting cross-site snooping.

**PROBLEM**

## #6 Information Leakage

“Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data, or conduct more serious attacks.”



# #6 Information Leakage

<http://example.com/broken-site>

## Error

Cannot connect to database.  
(mysql://rootuser:unguessable@localhost:/content)

Password shown  
to visitors.

How Drupal prevents...

# #6 Information Leakage

- ▶ Administrators can configure Drupal (and even PHP) to privately log errors, intercepting them before they ever reach users.
- ▶ Drupal (unlike some PHP applications) never displays password information when experiencing database connection issues.
- ▶ Drupal ships with a .htaccess (Apache web server configuration) file preventing many forms of snooping.
- ▶ It is not possible to read original user passwords.



**PROBLEM**

# #5 Cross-Site Request Forgery

“A CSRF attack forces a logged-on victim’s browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim’s browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks.”

# #5 Cross-Site Request Forgery

- 1 Log into bank site.
- 2 Visit a web forum.
- 3 An “image” on the forum makes a request on the bank site.

<http://example.com/node/10>

## My forum post

This is a really interesting forum post.

?

```

```



How Drupal prevents...

## #5 Cross-Site Request Forgery

- ▶ If a site allows users to load any content off external servers, the site can be used to originate attacks. This is configurable either way in Drupal.
- ▶ Drupal filters out scripting variations of this attack, leaving only simpler (GET-type) ones.
- ▶ The simpler CSRF attacks fail when attacking Drupal because the Form API isolates state-changing operations behind POST requests.
- ▶ The Form API also requires loading forms prior to submission, making CSRF attacks much harder.

**PROBLEM**

# #4 Insecure Direct Object Reference

“A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter. Attackers can manipulate those references to access other objects without authorization.”



# #4 Insecure Direct Object Reference

<http://example.com/show.php?page=/etc/group>

**Welcome to /group**

```
nobody:*:-2:nogroup:*:-1:wheel:*:0:rootdaemon:*:1:rootkmem:*:  
2:rootsys:*:3:roottty:*:4:root
```

System data  
shown to visitors.

How Drupal prevents...

## #4 Insecure Direct Object Reference

- ▶ Drupal's menu and form APIs encourage validating and sanitizing data submitted from users.
- ▶ When object references are passed through the Form API, Drupal core protects the values from tampering by site users.
- ▶ Drupal and PHP provide file and session APIs that allow convenient and secure object reference passing.

**PROBLEM**

## #3 Malicious File Execution

“Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework which accepts filenames or files from users.”

# #3 Malicious File Execution

<http://example.com/show.php?page=../reinstall-site.php>

**Site content deleted**

Ready to run the installer. [Reinstall site >](#)

↑  
User tries to run  
arbitrary files.

How Drupal prevents...

## #3 Malicious File Execution

- ▶ PHP has a configurable base directory for inclusions. Using this option limits possible attacks to only the Drupal directories.
- ▶ Drupal modules generally offer no entry point except through Drupal's secure URL/menu handler. So, while users may be able to load arbitrary PHP files, the "attacks" will have no effect.
- ▶ Prevention of "insecure direct object reference" attacks also helps here.

**PROBLEM**

## #2 Injection Flaws

“Injection flaws, particularly SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker’s hostile data tricks the interpreter into executing unintended commands or changing data.”

# #2 Injection Flaws

http://example.com/user

Username  
Administrator" OR uid=1 OR "1"="1

Password  
.....

Login

1 Carefully construct a “username” that changes the SQL.

```
SELECT uid FROM users  
WHERE  
name="Administrator"  
OR uid=1  
OR "1"="1"  
AND password="kjsdkjds"
```

2 See how it affects the actual query run.

3 Use administrator privileges.



How Drupal prevents...

## #2 Injection Flaws

- ▶ Drupal provides a database API with built-in SQL injection attack prevention. Properly used, it is not possible to inject arbitrary SQL.
- ▶ Drupal 7's new database API makes writing insecure database code even more difficult.
- ▶ Drupal provides a set of functions to process URLs and SQL arguments, making security an easy choice for developers.



**PROBLEM**

# #1 Cross-Site Scripting

“XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim’s browser which can hijack user sessions, deface web sites, possibly introduce worms, etc.”

# #1 Cross-Site Scripting

- 1 Visit a web forum.
- 2 Embedded script performs actions as you.

<http://example.com/node/10>

## My forum post

This is a really interesting forum post.

```
<script>  
fetch_url('http://example.com/user/20/delete')  
</script>
```

How Drupal prevents...

# #1 Cross-Site Scripting

- ▶ Drupal has a system of input filters that remove potential XSS exploits from user input.
- ▶ The Form API verifies that a user loaded a form before submitting it. This verification makes effective XSS against Drupal sites considerably more difficult.

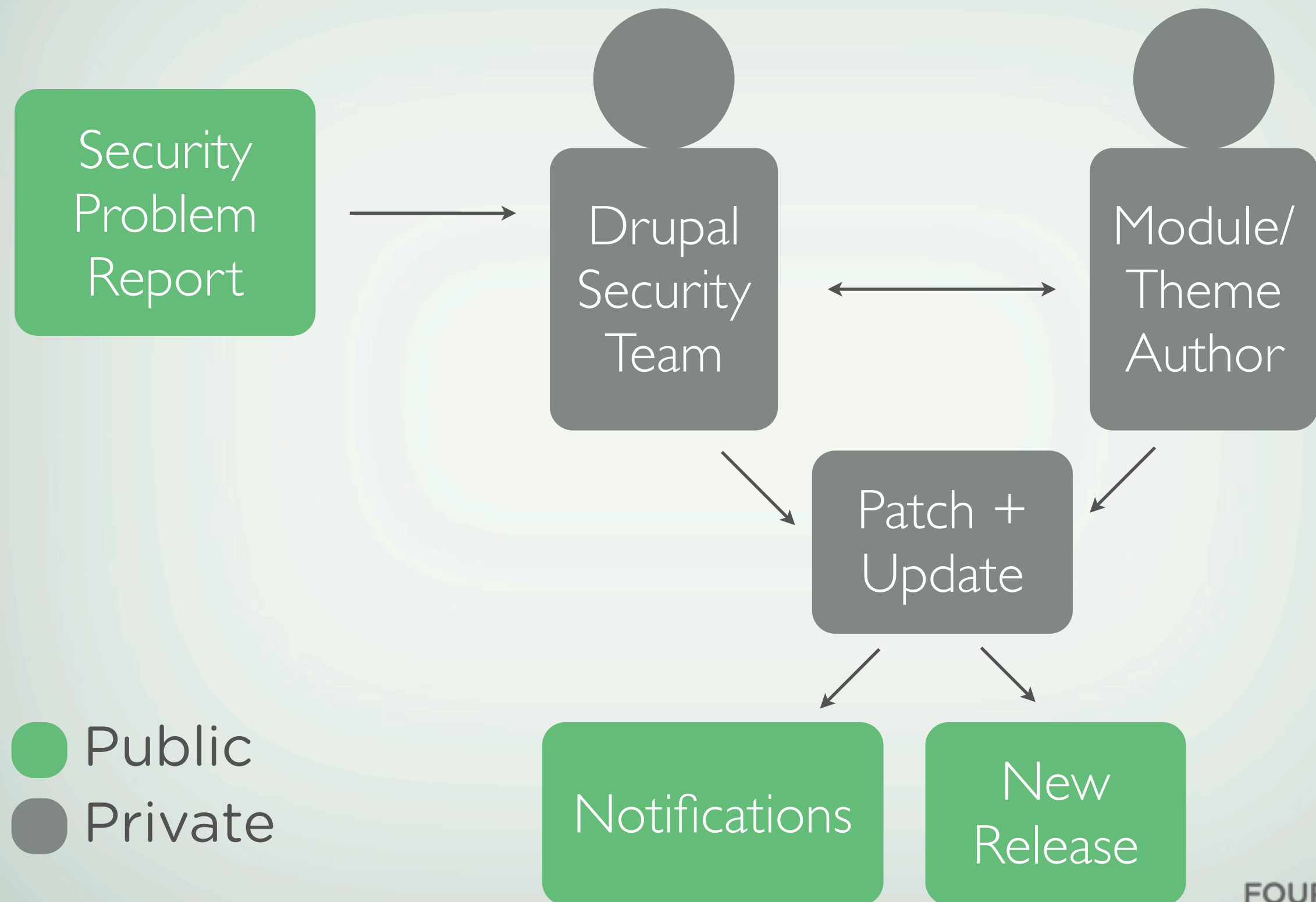
# Fixing + finding problems

How the Drupal project finds, fixes, and notifies users about security problems

# Who's checking Drupal?

- ▶ Hundreds of contributors
- ▶ Thousands of users
- ▶ Security researchers
  - ▶ There's glory in finding problems
- ▶ Government and corporate certification organizations

# Handling problem reports



# Thinking in terms of “tainted” data

- ▶ The #1 and #2 security flaws (injection and XSS) represent the vast majority of Drupal security issues in the past year.
- ▶ Both are the results of inadequately processing user-submitted data prior to use.
- ▶ “Tainting” is one way to model use of user-provided data.
- ▶ In 2008, Drupal received core-wide automated analysis of its handling of user-submitted “tainted” data. (Thanks Barry Jaspan.)



# How data tainting works





# Enterprise best practices

Working with Drupal's security team  
to keep your project secure

# Enterprise best practices

- ▶ Subscribe to Drupal **security notification lists**.
- ▶ Have a **testing environment** ready to evaluate updates for deployment. (Drupal core updates typically come out on Wednesdays.)
- ▶ If you modify Drupal core, have a **vendor branch** management strategy for keeping your changes and still being able to upgrade.
- ▶ Have **code reviews** for your own work and all but the most popular contributed modules.
- ▶ If your team is new to Drupal, **find a vendor** to review your code, configuration, and choice of modules.

we make BIG websites

FOUR  
KITCHENS



All content in this presentation, except where noted otherwise, is [Creative Commons Attribution-ShareAlike 3.0 licensed](#) and copyright 2009 Four Kitchen Studios, LLC.