PHP: Best Mailing Practices

Wez Furlong
<wez@messagesystems.com>

About the author

Or, "why is this guy talking to me about this?"

- Day-job is lead architect on the fastest MTA on Earth
- Some of our customers are large scale senders and receivers of mail
- I also know a little bit about PHP

What we will cover

- Constructing mail
- Sending mail
- Reliability, Performance
- Bounce processing
- Responsibility

Constructing mail (rfc2822)

```
From: “Wez Furlong” <wez@omniti.com>

To: “Me” <wez@php.net>

Subject: a simple message

Date: Sun, 10 Sep 2006 07:39:06 -0400
  (EDT)


Here’s the body of the message.
```

Constructing mail (alexandria)

```
include ‘OmniTI/Mail/Mailer.php’;

$mail = new OmniTI_Mail_Mailer;

$mail->setBodyText(‘Here’s the body of the
  message.’);

$mail->setFrom(‘wez@omniti.com’,
               ‘Wez Furlong’);

$mail->addRecipient(‘wez@php.net’, ‘Me’);

$mail->setSubject(‘a simple message’);

$mail->send();  # uses mail() by default
```

Constructing mail (ZF)

```
require_once ‘Zend/Mail.php’;

$mail = new Zend_Mail();

$mail->setBodyText(‘Here’s the body of the
  message.’);

$mail->setFrom(‘wez@omniti.com’,
               ‘Wez Furlong’);

$mail->addTo(‘wez@php.net’, ‘Me’);

$mail->setSubject(‘a simple message’);

$mail->send();  # uses mail() by default
```

Constructing Mail (eZ)

```
require_once ‘example_autoload.php’;

$mail = new ezcMailComposer();

$mail->from = new ezcMailAddress(‘wez@omniti.com’,
                 Wez Furlong’));

$mail->addTo(new ezcMailAddress(‘wez@php.net’,
   ‘Me’));

$mail->subject = “a simple message”;

$mail->plainText = “Here’s the body of the message”;

$tr = new ezcMailMtaTransport();

$tr->send($mail);
```

Performance

mail() forks a process on each call

this has performance implications for

- batch sending
- busy servers

You can avoid these issues by using SMTP

Sending over SMTP (ZF)

```
require_once 'Zend/Mail/Transport/Smtp.php';

$tr = new Zend_Mail_Transport_Smtp(
                'mail.example.com');

Zend_Mail::setDefaultTransport($tr);



$mail = new Zend_Mail;

…

$mail->send();
```

Sending over SMTP (eZ)

```
require_once ‘example_autoload.php’;

$mail = new ezcMailComposer();

$mail->from = new ezcMailAddress(‘wez@omniti.com’,
                 Wez Furlong’));

$mail->addTo(new ezcMailAddress(‘wez@php.net’,
   ‘Me’));

$mail->subject = “a simple message”;

$mail->plainText = “Here’s the body of the message”;

$tr = new ezcMailSMTPTransport(“mail.example.com”);

$tr->send($mail);
```

Batch sending over SMTP

```
require_once 'Zend/Mail/Transport/Smtp.php';

$tr = new Zend_Mail_Transport_Smtp(
                 'mail.example.com');

Zend_Mail::setDefaultTransport($tr);

$tr->connect();

foreach ($batch as $item) {

  $mail = new Zend_Mail;

  …

  $mail->send();

}

$tr->disconnect();
```

Batch Sending over SMTP (eZ)

```
require_once ‘example_autoload.php’;

$mail = new ezcMailComposer();

$mail->from = new ezcMailAddress(‘wez@omniti.com’,
                 Wez Furlong’));

$mail->addTo(new ezcMailAddress(‘wez@php.net’, ‘Me’));

$mail->subject = “a simple message”;

$mail->plainText = “Here’s the body of the message”;

$tr = new ezcMailSMTPTransport(“mail.example.com”);

$tr->keepConnection();

$tr->send($mail);
```

Batch Sending over SMTP (alexandria)

```
include ‘OmniTI/Mail/Mailer.php’;
$t = new OmniTI_Mail_Transport_SMTP_Injector;
OmniTI_Mail_Mailer::setTransport($t);


$mail = new OmniTI_Mail_Mailer();
while (...) {
 $mail->clearRecipientList();
 $mail->setFrom(‘wez@omniti.com’, ‘Wez Furlong’);
 $mail->addRecipient(‘wez@php.net’, ‘Me’));
 $mail->setSubject(“a simple message”);
 $mail->setBodyText(“Here’s the body of the message”);
```

Batch Sending over SMTP (alexandria)

```
  // continuing from previous...
  process_results($mail->send());
}


do {
  $st = $t->getAsyncStatus(true);
  if ($st === false) break; // all done
  process_results($st);
} while (true);
```

Batch Sending over SMTP (alexandria)

```
function process_results($st){
  if (is_array($st)) foreach ($st as $resp) {
    // $resp is array(
    //   0 => from
    //   1 => to
    //   2 => smtp status: something like
    //       “250 ok” or “550 you stink”,
    //       or “451 try later”
    //  );
  }
}
```

Batch sending from a web page

- If you have a page that sends notification mail to a bunch of people in response to a submission
- Try to avoid sending one mail per recipient right there:
  - multi-recipient mail (no good for VERP)
  - delegate mail sending to a background process

Reliability

Using mail() (on unix) submits to a local MTA to handle the harder part of getting the mail where it needs to go

Switching to SMTP can improve performance

Connectivity and load issues on the SMTP host can introduce reliability issues from your app

Best to use SMTP to submit to a local MTA, or other local process that will generate/ send.

Responsible sending

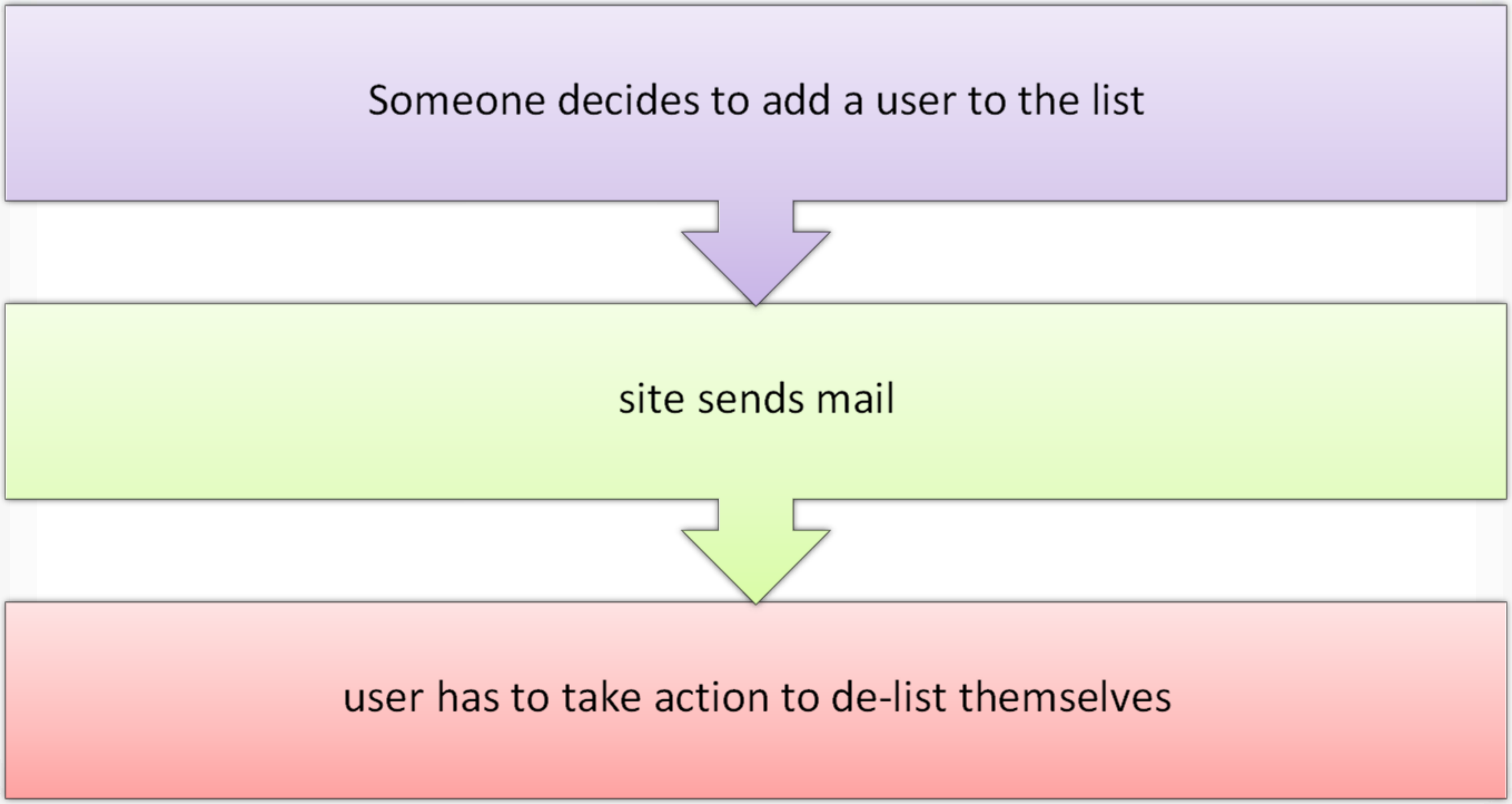
- Goals:
  - Provide useful service to those that want it
  - Avoid being mis-categorized as a spammer

Responsible sending

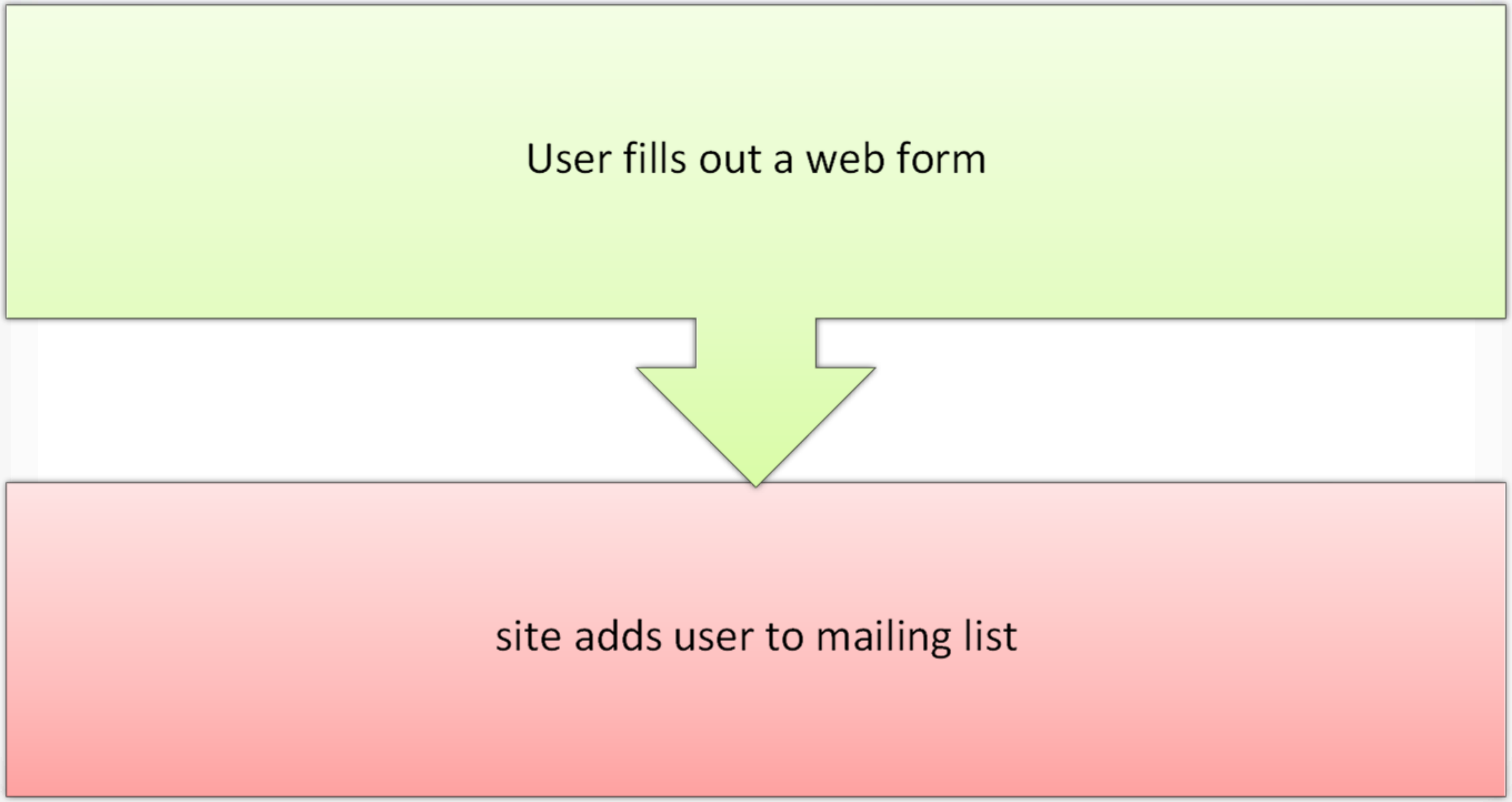
General Requirements:

- Avoid sending mail unless you know someone wants it
- Paying attention to bounces and acting on them in a timely manner

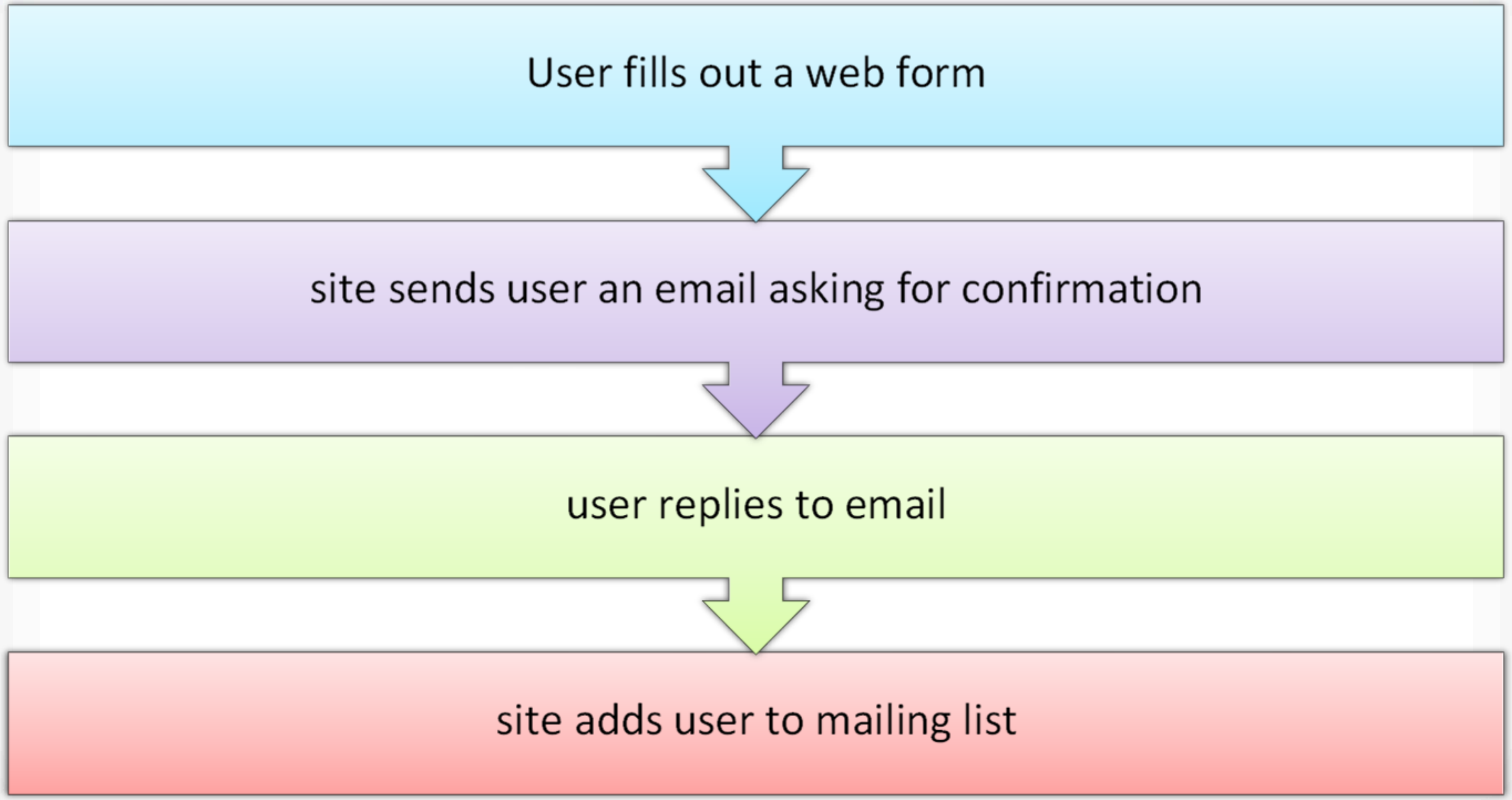
opt-out


Someone decides to add a user to the list
site sends mail
user has to take action to de-list themselves

opt-in



Double or confirmed opt-in


User fills out a web form
site sends user an email asking for confirmation
user replies to email
site adds user to mailing list

Abuse issues

- Double opt-in is clearly much more mailbox friendly
- In implementing it, you should build in throttling behavior so that the confirm mails are not sent in bulk

Bounces

- What happens when mail could not be delivered?
  - 5xx
  - Mail Delivery Notification (MDN)
  - Delivery Status Report (DSN)

Mail Envelope

To: and From: headers are what’s written on the “letter”

The envelope recipient and sender are what’s written on the packaging

MTAs route based on the envelope

The envelope can change during transit; aliasing, forwarding and so on

Return Path

- Return Path is another name for the envelope sender
- It’s the address to which bounces should be sent
- It’s usually the same as the From: address
- But can be changed in transit using schemes like SRS

Bounce processing

- If you send mail and it fails, it’s important to determine why
- Typical reasons:
  - user doesn’t exist
  - user over quota
  - site/user regards your mail as spam

Bounce processing

- Sending mail to non-existent mailboxes is a spammy behavior
  - you might find that your other mail is penalized as a result
- If your mail is being classed as spam you should contact the user or their site to find out why.
  - maybe the user couldn't find out how to stop getting your mail and resorted to a block on their side

Bounce processing

- If your mail is not getting through due to transient errors (like being over quota) you might want to re-try your mail later.
  - Would the user benefit from getting it later?
  - Would the user get swamped by a flood of mail from you once they're accepting mail again?
  - Would that cause them to block you as a spammer?

Variable Envelope Return Paths

- Bounce processing is made difficult because there isn't widespread adoption of DSN standards, making it hard to determine which address bounced
- VERP is a mechanism for creating a per recipient bounce address; if you receive mail to such a VERP address it must be a bounce

VERP

```
Return-Path: <bounces-wez=omniti.com@example.com>

From: "my list" <my-list@example.com>

To: wez@omniti.com

Subject: a message
```

Users replies go to <my-list@example.com>

Bounces go to <bounces-wez=omniti.com@example.com>

When they arrive you know that you were trying to send to wez@omniti.com

You can encode whatever you want into your VERP

- mailing ID, user ID, other tracking information

Sending using VERP

```
eZ: $mail->returnPath = new ezcMailAddress('bounces-
   wez=omniti.com@example.com');

others: $mail->setReturnPath('bounces-
   wez=omniti.com@example.com');
```

if you're using mail(), passing –f will set the return path.

if you're talking SMTP directly, the MAIL FROM: that you send to the server is the return path.

Sending to a third party on behalf of a user

User adds a 3rdParty address to a web form

↓

site sends email to 3rdParty because User is trusted

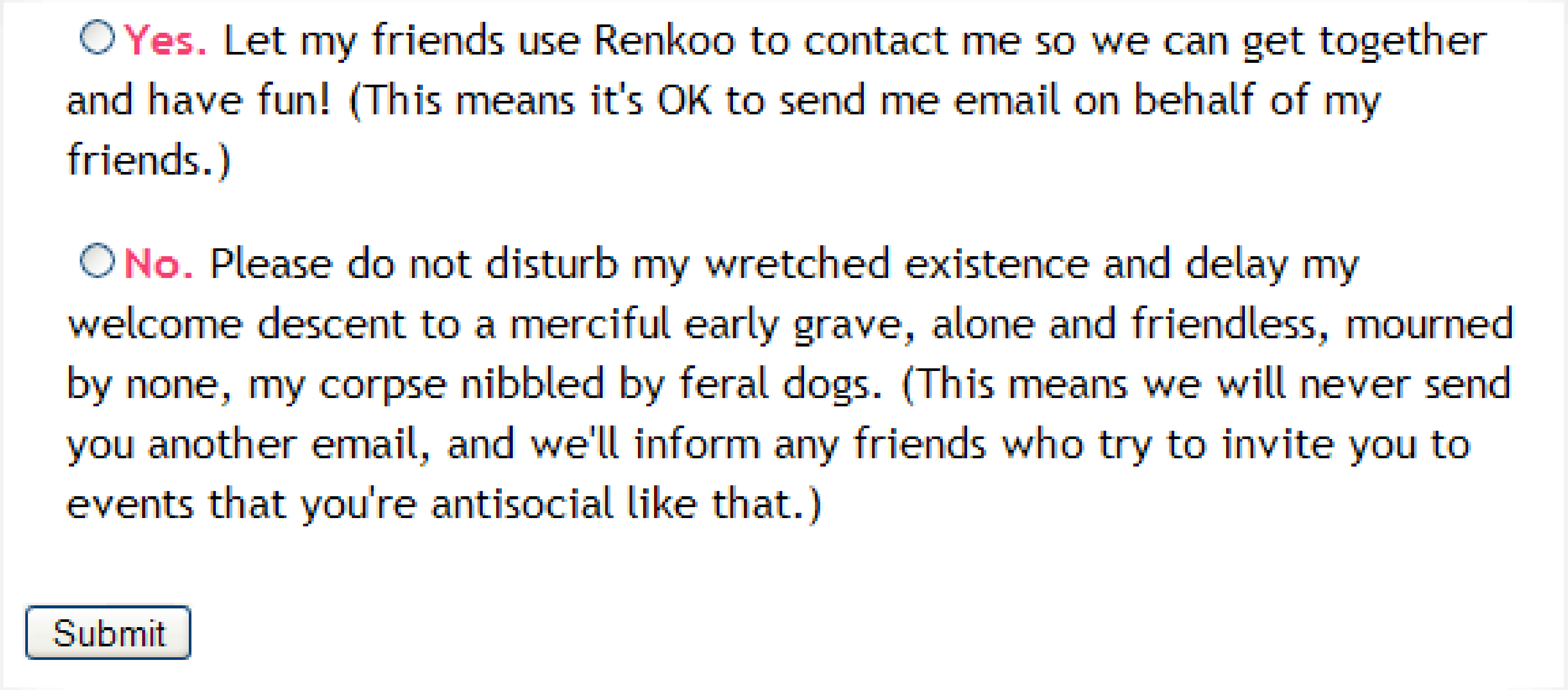
↓

3rdParty gets email that maybe they didn’t want.
Does this sound like opt-out?

Sending to a third party on behalf of a user

- If your users turn into abusers, it’s your responsibility to do something about it.
- It may be their fault but it came from your network, so you get the blame, and possibly blacklisted.
- You need to take steps to reduce the impact of such an occurrence, so it doesn’t affect your business.

Renkoo: opt-out



○ **Yes.** Let my friends use Renkoo to contact me so we can get together and have fun! (This means it's OK to send me email on behalf of my friends.)

○ **No.** Please do not disturb my wretched existence and delay my welcome descent to a merciful early grave, alone and friendless, mourned by none, my corpse nibbled by feral dogs. (This means we will never send you another email, and we'll inform any friends who try to invite you to events that you're antisocial like that.)

Sending to a third party on behalf of a user

- Allow 3rdParty to opt-out
- Throttle sending to 3rdParties that are not also users
- Throttle sending from users to an unusually high number of 3rdParties

Joe Jobs, Phishing

- Joe Job:
  - Spam forged to look like it came from you, designed to get you blacklisted.
- Phishing:
  - Mail forged to look like it came from you, designed to extract user credentials. Your users are the people that will get hurt.

Mitigating Risks of forged mail

- Adopting standards like DKIM and Sender ID allow you publish information about what constitutes valid mail from your site.
- While not a silver bullet, they help.

Domain Keys Identified Mail

- DKIM uses cryptography to sign portions of each message that you send. The public key is published in your DNS so that the receiver can validate the signature.
- An invalid signature is likely a forgery
- A missing signature doesn't mean anything
  - unless you have communicated that your policy as "all my mail is dkim signed"
  - a mechanism for this a draft IETF standard

Sender ID

- Sender ID (and SPF) publish information about legitimate sending hosts and your sending policy in DNS.
- A receiver can look up the information and compare it against fields in the message to determine if it was sent from your site.

Authentication, Reputation

- DKIM and Sender ID allow the receiver to determine if the sender is who they say they are.
- They don’t assert that the sender is, or is not, a spammer or a fraud.

Summing up

- It’s easy to send mail from a web app
- That doesn’t mean that you shouldn’t think a bit harder about it
- How you send, and how you let others send through your site affect your ability to send mail
- This affects your business

Questions?

- These slides are available from my blog and slideshare.net
- My blog http://netevil.org/
- Alexandria http://labs.omniti.com