

## Redmine - Defect #19577

### Open redirect vulnerability with back\_url param

2015-04-09 12:50 - Jan from Planio [www.plan.io](http://www.plan.io)

<b>Status:</b>	Closed	<b>Start date:</b>	
<b>Priority:</b>	Normal	<b>Due date:</b>	
<b>Assignee:</b>	Jean-Philippe Lang	<b>% Done:</b>	0%
<b>Category:</b>	Security	<b>Estimated time:</b>	0.00 hour
<b>Target version:</b>	2.6.7	<b>Affected version:</b>	
<b>Resolution:</b>	Fixed		

**Description**

### Summary

The valid\_back\_url? method used e.g. after a login to redirect the user back to where they came from doesn't properly validate passed URLs resulting in an open redirect vulnerability which can be used for phishing and other attacks.

After the redirect to the untrusted site, phishers may then steal the user's credentials and then use these credentials to access the legitimate web site. Because the server name in the modified link is identical to the original site, phishing attempts have a more trustworthy appearance.

### Description

When redirecting the user back after a successful login, redirect\_back\_or\_default is trying to validate the passed URL to ensure that the target of the HTTP 302 redirect is valid.

However, the valid\_back\_url? method used to validate the URL doesn't take some cases into account which can result in a redirect to an arbitrarily chosen host.

Example exploit:

```
http : // redmine. example. com/login? back_url=@attacker. com
```

This URL results in a redirect to

```
http : // redmine. example. com @attacker. com
```

which results in a request to `http : // attacker. com` with `redmine. example. com` passed as a basic auth user.

### Credits

This issue was discovered by Yassine ABOUKIR of <http://yassineaboukir.com/>. The patch was developed by Holger Just of Planio.

### Solution

The attached patch fixes this vulnerability. It adapts the valid\_back\_url? method to a method called validate\_back\_url which returns the validated and cleaned up URL which can be used by the redirect method.

The patch cleanly applies against the current trunk as well as previous Redmine versions (including 2.5, 2.6, and 3.0)

---

## Associated revisions

### Revision 14560 - 2015-09-13 16:35 - Jean-Philippe Lang

Open redirect vulnerability (#19577).

Patch by Holger Just.

### Revision 14562 - 2015-09-13 16:43 - Jean-Philippe Lang

Merged r14560 and r14561 (#19577).

### Revision 14563 - 2015-09-13 16:44 - Jean-Philippe Lang

Merged r14560 and r14561 (#19577).

### Revision 14564 - 2015-09-13 16:45 - Jean-Philippe Lang

Merged r14560 and r14561 (#19577).

---

## History

### #1 - 2015-04-09 12:52 - Jan from Planio [www.plan.io](http://www.plan.io)

- Description updated

(I am submitting this in separate steps changing the description since my first attempts were blocked by redmine.org's spam filter?)

### #2 - 2015-04-09 12:52 - Jan from Planio [www.plan.io](http://www.plan.io)

- Description updated

### #3 - 2015-04-09 12:53 - Jan from Planio [www.plan.io](http://www.plan.io)

- File 0001-Prevent-open-redirect-to-arbitrary-hosts.patch added

### #4 - 2015-04-09 12:53 - Jan from Planio [www.plan.io](http://www.plan.io)

- Description updated

### #5 - 2015-04-12 23:38 - Jan from Planio [www.plan.io](http://www.plan.io)

- File 0001-Prevent-open-redirect-to-arbitrary-hosts.patch added

Here's an updated version that also prevents something like ?back\_url=/attacker.com

### #6 - 2015-04-12 23:38 - Jan from Planio [www.plan.io](http://www.plan.io)

- File deleted (0001-Prevent-open-redirect-to-arbitrary-hosts.patch)

### #8 - 2015-09-13 16:39 - Jean-Philippe Lang

- Project changed from 2 to Redmine

- Subject changed from Open redirect vulnerability to Open redirect vulnerability with back\_url param

- Category set to Security

- Status changed from New to Resolved

- Assignee set to Jean-Philippe Lang

- Target version set to 2.6.7

- Private changed from No to Yes

- Resolution set to Fixed

### #9 - 2015-09-13 16:45 - Jean-Philippe Lang

- Status changed from Resolved to Closed

Fix committed in 3.1, 3.0 and 2.6 branches. Thanks.

### #11 - 2015-09-16 19:22 - Jean-Philippe Lang

FTR, this vulnerability was introduced in [r13213](#), which was supposed to fix another issue. This is not good :(

Rather than trying to parse the `back_url` and check if it's OK, another option would be (in addition or even instead of the current check) to sign the `back_url` param with a secret using `ActiveSupport::MessageVerifier`.  
What do you think?

**#12 - 2015-09-16 19:43 - Jan from Planio [www.plan.io](http://www.plan.io)**

Jean-Philippe Lang wrote:

What do you think?

Hmm, interesting thought. Similar to how OpenID/OAuth are making sure that `back_urls` are valid. It would certainly make sure the URLs are all coming from "within" Redmine. I am pretty sure that Holger would have an opinion on this, but he's currently on holidays :) I'll ask him next week and post any feedback if you'd like one more opinion.

**#13 - 2015-09-16 20:45 - Jean-Philippe Lang**

Redmine may also redirect to the referer:

- no check is done in the current implementation (might be exploited)
- redirect to the referer would not be supported with the solution proposed in [#19577-11](#)

**#14 - 2015-09-22 16:21 - Jan from Planio [www.plan.io](http://www.plan.io)**

Jan from Planio [www.plan.io](http://www.plan.io) wrote:

I'll ask Holger next week and post any feedback if you'd like one more opinion.

As promised, here's Holger's opinion on this for your consideration:

While it is certainly possible to sign the URL using ActiveSupport's `MessageVerifier` class, doing so would significantly complicate matters for external users of the interface (e.g. external services integrating into Redmine who want to force a login, OAuth or single sign-on login provider, ...). Also, it would hide the URL verification behind an opaque layer of cryptographic "magic" which isn't guaranteed to solve the issue but makes verification much harder, for devs as well as end users.

Instead, in order to provide an easier verification solution, I'd propose to define the protocol for these URLs more strictly which would make verification easier for us.

One way for this is to only allow full URLs with the correct protocol and hostname as configured in the setting which could then easily be verified using this Regexp (with proper escaping added):

```
\A#{Setting.protocol}://#{Setting.host_name}(/|\z)
```

Such a full URL will be passed through all Rails layers unchanged on redirects. Then, you just need to adapt all places where the back URL is generated to generate a full URL instead, but this needs roughly the same effort as when using signed URLs.

**#15 - 2015-09-23 21:13 - Jean-Philippe Lang**

Thanks for sharing your thoughts.

While it is certainly possible to sign the URL using ActiveSupport's `MessageVerifier` class, doing so would significantly complicate matters for external users of the interface (e.g. external services integrating into Redmine who want to force a login, OAuth or single sign-on login provider, ...).

I'm not sure to get your point. Do you have a specific use case where this would be problematic while the current implementation is not?

Also, it would hide the URL verification behind an opaque layer of cryptographic "magic" which isn't guaranteed to solve the issue

If we sign URLs generated by Redmine only, how could it not solve the issue? Do you have a particular example in mind?

but makes verification much harder, for devs as well as end users.

That's a concern, but we can leave the `back_url` param as is (clear text) and add a `back_url_token` param that would hold the HMAC for verification.

Instead, in order to provide an easier verification solution, I'd propose to define the protocol for these URLs more strictly which would make verification easier for us.

One way for this is to only allow full URLs with the correct protocol and hostname as configured in the setting which could then easily be verified using this Regexp (with proper escaping added):

```
\A#{Setting.protocol}://#{Setting.host_name}(/|\z)
```

Such a full URL will be passed through all Rails layers unchanged on redirects. Then, you just need to adapt all places where the back URL is generated to generate a full URL instead, but this needs roughly the same effort as when using signed URLs.

I see a few problems with this solution:

- Generating full URLs with protocol is a hassle, we've got a few tickets about the generated links in email notifications especially from people who run Redmine in a sub-uri
- Redirects to back urls would not work until an admin properly sets the protocol and host name
- This would not work when accessing a Redmine instance with different (eg. internal/external) host names, or different protocols.

And checking a param against a regexp is IMHO much open for attacks than signing generated values. This old Firefox vulnerability is a good example. CRLF injection in a back url param would not work in Redmine but still, see how the malicious URL would perfectly match the regexp above: [https://bugzilla.mozilla.org/show\\_bug.cgi?id=655389](https://bugzilla.mozilla.org/show_bug.cgi?id=655389)

#### #16 - 2015-09-25 19:34 - Jan from Planio [www.plan.io](http://www.plan.io)

Jean-Philippe Lang wrote:

Thanks for sharing your thoughts.

While it is certainly possible to sign the URL using ActiveSupport's `MessageVerifier` class, doing so would significantly complicate matters for external users of the interface (e.g. external services integrating into Redmine who want to force a login, OAuth or single sign-on login provider, ...).

I'm not sure to get your point. Do you have a specific use case where this would be problematic while the current implementation is not?

I guess it would be for cases where external software needs to redirect to content in Redmine but explicitly wants the user to sign in for it. Currently, this is possible by just composing the URL "from the outside". I agree though, that this case is rather exotic and probably not something that Redmine should worry too much about. If external apps need this, there will certainly also be a way to achieve it *with* a signed `back_url` param. It could also be solved by having a config param in `configuration.yml` that would disable this feature.

Also, it would hide the URL verification behind an opaque layer of cryptographic "magic" which isn't guaranteed to solve the issue

If we sign URLs generated by Redmine only, how could it not solve the issue? Do you have a particular example in mind?

I guess that Holger's argument was that if someone somehow manages to get a malicious URL into Redmine to get it signed and then redirected, and we would rely on signed `back_url` s *only* for verification, we might still be vulnerable to malicious redirects if there's no additional validity check based on regex for example. I agree that this risk is somehow theoretical, though.

but makes verification much harder, for devs as well as end users.

That's a concern, but we can leave the `back_url` param as is (clear text) and add a `back_url_token` param that would hold the HMAC for verification.

Yes, I think it would be great if the `back_url` would remain "visible" for end users.

Instead, in order to provide an easier verification solution, I'd propose to define the protocol for these URLs more strictly which would

make verification easier for us.

One way for this is to only allow full URLs with the correct protocol and hostname as configured in the setting which could then easily be verified using this Regex (with proper escaping added):

[...]

Such a full URL will be passed through all Rails layers unchanged on redirects. Then, you just need to adapt all places where the back URL is generated to generate a full URL instead, but this needs roughly the same effort as when using signed URLs.

I see a few problems with this solution:

- Generating full URLs with protocol is a hassle, we've got a few tickets about the generated links in email notifications especially from people who run Redmine in a sub-uri
- Redirects to back urls would not work until an admin properly sets the protocol and host name
- This would not work when accessing a Redmine instance with different (eg. internal/external) host names, or different protocols.

And checking a param against a regexp is IMHO much open for attacks than signing generated values. This old Firefox vulnerability is a good example. CRLF injection in a back url param would not work in Redmine but still, see how the malicious URL would perfectly match the regexp above:

[https://bugzilla.mozilla.org/show\\_bug.cgi?id=655389](https://bugzilla.mozilla.org/show_bug.cgi?id=655389)

All valid points, I agree.

Jean-Philippe, do you know any other open source projects or sites, solving this issue using signed params? To be honest, I have never seen it "in the wild" which might also be the reason I'm still on the fence regarding this.

#### #17 - 2015-09-25 21:52 - Jean-Philippe Lang

Jan from Planio [www.plan.io](http://www.plan.io) wrote:

Jean-Philippe, do you know any other open source projects or sites, solving this issue using signed params? To be honest, I have never seen it "in the wild" which might also be the reason I'm still on the fence regarding this.

No. After a quick search, I found this guy from Google that suggests this solution:

Consider signing your redirects. If your website does have a genuine need to provide URL redirects, you can properly hash the destination URL and then include that cryptographic signature as another parameter when doing the redirect. That allows your own site to do URL redirection without opening your URL redirector to the general public.

<http://googlewebmastercentral.blogspot.fr/2009/01/open-redirect-urls-is-your-site-being.html>

Another option would be to use full URLs in all back\_url params as proposed by Holger but use a simpler approach for verification, without using protocol and hostname settings, but #root\_url instead. Rails generates the root url for us based on the current request:

```
back_url.starts_with?(root_url)
```

#### #20 - 2015-12-08 06:39 - Jan from Planio [www.plan.io](http://www.plan.io)

- Private changed from Yes to No

#### #21 - 2015-12-08 06:39 - Jan from Planio [www.plan.io](http://www.plan.io)

This has been fixed and released, therefore I'm opening up this issue.

## Files

---

0001-Prevent-open-redirect-to-arbitrary-hosts.patch	3.69 KB	2015-04-12	Jan from Planio <a href="http://www.plan.io">www.plan.io</a>
---	---------	------------	--