Security & PayPal

by
Peter Lavin

November 20, 2004

Introduction

PayPal is a convenient way of accepting payment over the internet, especially for small to medium-sized businesses. In an earlier article, I described how to integrate a third party shopping cart with PayPal. Reading that article, while not essential, will be helpful in following this discussion. You may find it at http://www.softcoded.com/pdfs/paypal1.pdf. Readers of that article raised a number questions related to security.

In general, problems may occur if a merchant's order is not cross-reference with a notice from PayPal. An inattentive merchant might well ship goods when payment has not been made or made in an incorrect amount. A small retailer is likely to notice any discrepancies but as business increases so do the chances of mishandled orders.

Having procedures in place to lower or eliminate the risk of this happening is a much more realistic solution. This article will deal with how this can be done.

Define the Problem

The security concerns raised are not uncommon. Any form on the internet is open to being "high jacked" regardless of whether values are exposed or hidden. A local copy of that form can be made and variables changed. If there is no way of cross checking these variables then an order might easily be processed at discounted prices.

Let's have a look at some of the HTML code that might appear in a form submitted to PayPal so that we know exactly what we are talking about. For the sake of simplicity we will only show the parts of the form that have a bearing on our discussion. Assume that the code below has been dynamically created – though what is said here applies equally to a static page.

```
<form name= "order" action="https://www.paypal.com/cgi-bin/webscr"
method="post">

<!—other parts of form here-->
…
<input type="hidden" name="amount" value="1000.00" />
<input type="hidden" name="return"
value="http://www.ecommerce.com/notify.php?status=T&orderid=001” />
<br />
<input type="hidden" name="cancel_return"
value=http://www.ecommerce/notify.php?status=F” /><br />
…
</form>
```

Of particular interest are the “return” and the “amount” variables. The meaning of “amount” is fairly straightforward and, as defined by PayPal “return” is, “An internet URL where your customer will be returned after completing payment”. By making a copy of this form someone could change the value of the amount field, or they could bypass PayPal and go directly to the return URL.

PayPal sends out a confirmation email when any money is deposited into a client’s account. It is a relatively simple matter for a smaller merchant to manually check any order to ensure that prices are correct and that payment has, in fact, been made. If this is done conscientiously then any “high jacking” attempts will be readily spotted.

However, a merchant might want the return URL to act as a trigger for the creation of an order or at least the removal of goods from inventory. If this is the case and a buyer has decided to skip payment and go directly to the URL shown in the “return” field then problems may arise. The obvious solution would be to remove the return URL altogether as this is an optional field and is not required when submitting to PayPal. You need not return your customer to your site after payment and could simply await notification from PayPal. This is certainly one solution, but apart from it being poor retail practice there are a number of good reasons why you might want your customers to return to your site. (You may also encrypt your form but this option only applies to “Buy Now Buttons”.)

One alternative to exposing your return URL in the form submitted to PayPal is to set up the “Auto Return for Website Payments” within your PayPal profile. In this way you could achieve greater security by hiding your return URL. But even if you choose to hide the return URL within your PayPal profile it would be a fairly easy matter to determine it by placing a small order for inexpensive goods. Additionally, especially when creating a dynamic URL, you may have no choice but to expose it in your form.

There must be a way of automating this process and handling these kinds of security problems.

Instant Payment Notification (IPN)

IPN is PayPal’s answer to the questions raised above. Order processing may be automated for shopping carts, “Buy Now Buttons” – in fact any of the payment solutions offered by PayPal may be “seamlessly integrated” with back-end operations.

While various kinds of online transactions can be handled by IPN, here we will concentrate on payments. IPN is instant because you can know right away when

money has been transferred into your account. In your merchant profile you can specify an URL to receive notification immediately upon payment. To quote from the PayPal Integration guide:

"Included in this payment notification will be all of your customer's payment information (e.g. customer name, payment amount) as well as a piece of encrypted code. When your server receives a notification it will then post the information, including the encrypted code, back to the secure PayPal URL. PayPal will authenticate the transaction and send confirmation of its validity back to your server."

However this does not mean that you can happily process an order simply because the encrypted code is verified by PayPal. Further checks still need to be made. You might ask aren't we right back where we started? The answer is "No" because verification ensures that notification really did originate from PayPal and because we have access to the actual variables passed into the PayPal site we have a chance to check them programmatically and do away with a manual check.

For all the details about IPN you can see the "Integration Guide" referenced in the Resources section of this article. The important thing to note is that IPN allows you to verify online that a payment has been received at PayPal and in what amount. You need not await an email and verify it by hand.

Implementation

In any but the simplest scenario, IPN will probably be used in conjunction with a database. Typically the process may be something like the following: the invoice number passed in will be used to look up invoice items and calculate an invoice total. In turn this total will be matched with the value sent by PayPal. Checks will also be made to ensure that the money is credited to the correct account.

Creation of the code necessary to use IPN is not onerous because PayPal provides software development kits using PHP, Perl, ColdFusion, and ASP. See the Resources section for a link to these files. For PHP, a couple of different solutions are presented; one using "fsockopen" and the other "Curl". The snippet of code below is based on PayPal's toolkit. It will give you a sense of what's required and alert you to one small frustration.

```
//loop and capture values posted from PayPal
foreach($_POST as $key => $value){
  $value=urlencode(stripslashes($value));
  $data.="$key=$value&";
}
//must add this command before sending back
```

```
$data.="cmd=_notify-validate";
//assume code here to create variables for values that need verifying
etc
//ignore others
…
//now send back to paypal for confirmation
$c=curl_init('https://www.paypal.com/cgi-bin/webscr');
//test url below
//$c=curl_init("https://www.sandbox.paypal.com/cgi-bin/webscr");
curl_setopt($c, CURLOPT_POST,1);
curl_setopt($c, CURLOPT_POSTFIELDS, $data);
curl_setopt($c, CURLOPT_SSL_VERIFYPEER,FALSE);
//Start ob to prevent curl_exec from displaying
ob_start();
curl_exec($c);
//get contents of output buffer
$status=ob_get_contents();
//strip CR
$status=rtrim($status);
//only one value returned
if (strcmp($status,"VERIFIED")==0){ …
      //check amount etc here against database values
}
```

Note the code in bold above. I found it necessary to right trim the value returned by PayPal but that is the only glitch I encountered. Using libCurl is a nice, tidy way to handle IPN and most servers that support PHP come with the CURL package installed.

Testing

You might have noticed that a test URL has been commented out in the above code. Joining the PayPal Developers Network allows you to test your code in a “sandbox” and work out the wrinkles before going live. The sandbox allows you to simulate almost every situation that you might encounter in real life. Buyer and seller accounts may be set up. By posting to the sandbox you can see exactly what would happen on the PayPal site. You can check that form variables are submitted properly and see what your customers would see. As is the case with real transactions, email notifications are sent out to your “buyers” and “sellers”.

Most importantly, you can check that IPN works without having to actually purchase goods. The sandbox is a valuable tool that makes a developer’s life simpler and much less stressful when you do actually “go live”.

Conclusion

For the small merchant who manually checks all orders and waits for email confirmation from PayPal before shipping, security problems like the ones

described here are not an issue. However, if you decide to automate order processing then precautions must be taken. Use of IPN and, in most cases a database, will give you adequate protection. IPN allows for online confirmation of receipt of payment into your account. Payment amounts and other sensitive information can be verified against database records. In this way the online retailer can get on with business and not have to waste time scrutinizing orders.

Special thanks to Torstein Sorlid who raised and helped clarify some of the issues dealt with in this article.

Resources

PayPal Tool Kit
https://www.paypal.com/row/cgi-bin/webscr?cmd=p/pdn/software_dev_kit

PayPal Integration Guide
https://www.paypal.com/en_US/pdf/integration_guide.pdf

Previous Article about PayPal
http://www.softcoded.com/pdfs/paypal1.pdf

Adding PayPal to a Third Party Shopping Cart
https://www.paypal.com/cgi-bin/webscr?cmd=p/pdn/howto_checkout-outside -

About the Author

Peter Lavin runs a Web Design/Development firm in Toronto, Canada. For more information visit http://www.softcoded.com/. He may be reached at peterlavin@sympatico.ca.