1

PayPal Hack

#Add A Buy Now Button

Use PayPal to add a Buy Now button to your PHP web application

PayPal's "Buy Now" buttons are ideal for ecommerce impulse sales. They are easy for the customer to use and there's no lengthy check-out procedure. One click and you're on PayPal's secured site. They are also very easy to install and for this reason sometimes get ignored when it comes to tracking and securing sales. This hack shows how to track and secure purchases made using "Buy Now" buttons. We will take you through the steps of creating a "Buy Now" button, modifying it in order to create a database record of the purchase and, finally, we'll show how to secure this purchase using Instant Payment Notification (IPN).

This hack requires PHP 5 and MySQL version 4.1.3 or higher, along with the mysqli extension.

Creating a "Buy Now" Button

We needn't worry about manually creating the HTML form for a "Buy Now" button. We only need go to our PayPal account, choose "Merchant Tools" and click on the "Buy Now Buttons" link. Within seconds we have the HTML form to embed in our website. Our "Buy Now" button looks like Figure 1-1.

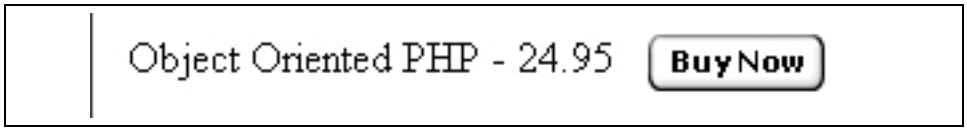

Figure 1-1. The Buy Now button

As it is, this form has a couple of shortcomings. It's not geared towards tracking purchases in a database and, like any form on the internet, it is subject to highjacking. A local copy of the form can be made and values can be changed. If there is no way of cross-checking these values then an order might easily be processed at discounted prices.

This is not a major problem for a low volume retailer who manually tracks a small number of orders – in these circumstances discrepancies will be quickly spotted. However, we will need something a little more secure if we want to automate the purchase process.

The *buynow.html* file in the code section below contains the HTML for our "Buy Now" button. It differs in one major respect from the code generated by PayPal - the "action" property of the form doesn't point to the PayPal site but to the file, *presubmit.php*. This will allow information to be added to a database before submitting purchases to PayPal.

Tracking the Sale

Briefly stated, the code in *presubmit.php* inserts a record into an orders table and into a related order items table. The order id is then retrieved and added to the query string constructed from the values posted to this page. Finally, this query string is forwarded to the PayPal site as shown in Figure 1-2.

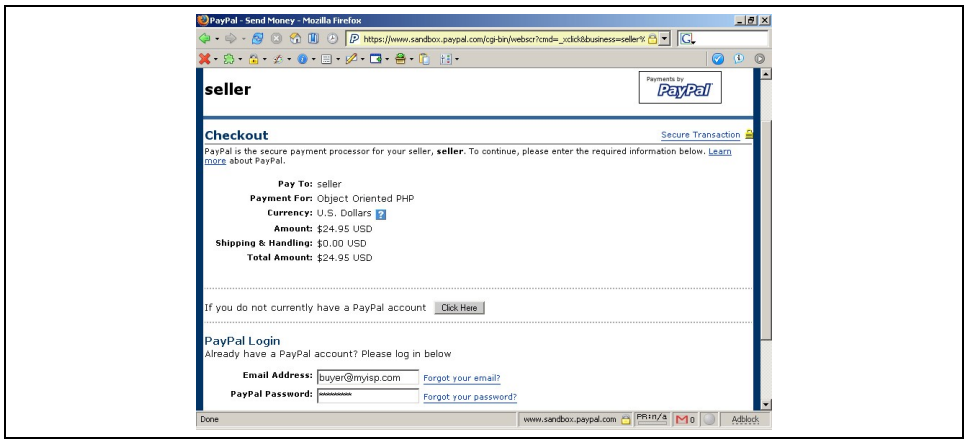

Figure 1-2. The PayPal checkout page

Creating a database entry and passing the order number along to PayPal will assist in tracking and securing the purchase.

Since we've accessed our database using the relatively new object-oriented (OO) interface to the mysqli extension a few comments are in order. Even with no experience of OO programming it is easy to understand the code that inserts a record into the orders table. Since this table has an "auto_increment" field we are able to retrieve the order id after insertion so as to later identify our order.

The way a record is inserted into the order items table is not quite so readily understood. To create a record in the order items table we use a prepared statement – a capability of MySQL 4.1 that is supported by the mysqli extension. Prepared statements are commonly used to insert multiple records into a database and are much more efficient than a series of individual SQL statements. However, we've used a prepared statement here because data passed to a prepared statement does not have to be escaped first. Prepared statements automatically escape data.

Verifying the Sale

Securing a payment means ensuring that the payment is made to the correct account, in the correct amount and is not a duplicate of an earlier transaction. IPN allows us to do this programmatically by identifying an URL that will receive notification of payment.

When a purchase is made at our site the sequence of events is as follows. Clicking the "Buy Now" button invokes the script to create a database record and then forwards the buyer to the PayPal site. After PayPal receives payment, a hidden post is made to our IPN URL. This post contains encrypted code and information about the payment. To ensure that this post is not specious and did in fact originate at PayPal, we must return this post to PayPal. PayPal will then respond, verifying that the post originated with them.

verifypurchase.php contains the code that confirms the source of the post and validates the data. It can be summarized as follows. The PayPal post is captured and resubmitted adding the name/value pair "cmd=_notify-validate" as required by PayPal. If this resubmitted post is verified as originating from PayPal, we confirm the details of the purchase. The price for the specific item is retrieved from the database and compared to the posted value. We then make sure that the transaction id is not a duplicate of an earlier purchase and, by verifying the receiver email, we ensure that the payment has been made into the correct account.

A few comments on less obvious features of the code. Using the CURL package is not the only way to handle the resubmission of posted values to PayPal but it certainly does make it easy. When using IPN, the names of some of the values we retrieve are different from the ones we originally posted. The "receiver_email" is synonymous with the "business" email posted from our form. "mc_gross" holds the payment amount and, in our case, its value should be equal to the price of the purchased item. On the other hand, "item_number" and "custom" have not changed at all. Again, we access the database using the OO interface of mysqli. Depending upon server settings, the port and socket settings may or may not be required when creating a connection object. Notice also that we can specify the database when creating the connection. Also, since we are using PHP 5, objects are automatically passed by reference and there is no need to adjust our syntax when objects are function parameters.

The Code

Save this code as *buynow.html*:

```
<html>
<head>
<title>Buy Now Button</title>
</head>
<body>
<!-- alter action of form-->
<form action="presubmit.php" method="post">
<label>Purchase: Object-Oriented PHP for 24.95</label>
<input type="hidden" name="cmd" value="_xclick" />
<input type="hidden" name="business" value="seller@myisp.com" />
<input type="hidden" name="item_name" value="Object-Oriented PHP" />
<input type="hidden" name="item_number" value="673498" />
<input type="hidden" name="amount" value="24.95" />
<input type="hidden" name="no_note" value="1" />
<input type="hidden" name="currency_code" value="USD" />
```

```
<input type="image" src="https://www.paypal.com/en_US/i/btn/x-click-but23.gif"
border="0" name="submit" alt="" />
</form>
</body>
</html>
```

Save this code as *connection.php*:

```
<?php
  $hostname = "localhost";
  $databasename = "books";
  $username = "username";
  $password = "password";
?>
```

Save this code as *presubmit.php*:

```
<?php
  include "connection.php";
  //create a connection using mysqli
  $con = new mysqli($hostname, $username, $password, $databasename,
     3306, "/var/lib/mysql/mysql.sock");
  //create an order id
  $strsql = "INSERT INTO tblorders SET orderdate = CURDATE()";
  $con->query($strsql);
  //retrieve insertid - property not method
  $id = $con->insert_id;
  $item_number = $_POST['item_number'];
  //now add order item to db
  $strsql = "INSERT INTO tblorderitems SET orderid = ?, ".
    "inventorynumber = ?";
  //use statement even though only one insert
  $stmt = $con->stmt_init();
  $stmt->prepare($strsql);
  //bind integer and string values
  $stmt->bind_param('ii', $id, $item_number);
  $stmt->execute();
  //resubmit
  $querystring = "?";
  //loop for posted values
  foreach($_POST as $key => $value)
  {
    $value = urlencode(stripslashes($value));
    $querystring .= "$key=$value&";
  }
  //update querystring with order id
  //use "custom" not "on0" for order id value
  $querystring .= "custom=$id";
  header('location:https://www.paypal.com/cgi-bin/webscr'.$querystring);
  exit();
?>
```

Save this code as *verifypurchase.php*:

```
<?php
include "connection.php";
////////////////////////////////////////////////////////////
function check_txnid($con, $txnid)
{
```

```
  $valid_txnid = false;
  //get result set
  $strsql = "SELECT * FROM tblorders ".
    " WHERE txnid = '$txnid'";
  $rs = $con->query($strsql);
  if($rs->num_rows == 0)
  {
    $valid_txnid = true;
  }
  return $valid_txnid;
}
////////////////////////////////////////////////////////////
function check_price($con, $price, $inventoryid)
{
  $valid_price = false;
  //get result set
  $strsql = "SELECT listprice FROM tblbooks ".
    " WHERE inventorynumber = '$inventoryid'";
  $rs = $con->query($strsql);
  $row = $rs->fetch_array();
  $num = (float)$row[0];
  if($num == $price)
  {
    $valid_price = true;
  }
  return $valid_price;
}
////////////////////////////////////////////////////////////
function check_email($email)
{
  $valid_email = false;
  //compare to paypal merchant email
  if($email == "seller@myisp.com" )
  {
    $valid_email = true;
  }
  return $valid_email;
}
////////////////////////////////////////////////////////////
function do_post($data)
{
  //now send back to paypal
  $c = curl_init('https://www.paypal.com/cgi-bin/webscr');
  curl_setopt($c, CURLOPT_POST,1);
  curl_setopt($c, CURLOPT_POSTFIELDS, $data);
  curl_setopt($c, CURLOPT_SSL_VERIFYPEER,FALSE);
  curl_setopt($c, CURLOPT_RETURNTRANSFER, 1);
  $status = curl_exec($c);
  curl_close($c);
  return $status;
}
////////////////////////////////////////////////////////////
//loop for posted values
$data = "";
foreach($_POST as $key => $value)
{
  $value = urlencode(stripslashes($value));
```

```
  $data .= "$key=$value&";
}
//must add this before returning to paypal
$data .= "cmd=_notify-validate";
$status = do_post($data);
//strip CR
$status = rtrim($status);
$payment_status = $_POST['payment_status'];
//get transaction id
$txn_id = $_POST['txn_id'];
if ($status == "VERIFIED" && $payment_status == "Completed")
{
  //need these variables
  $price = $_POST['mc_gross'];
  //get order number
  $orderid = $_POST['custom'];
  $inventoryid = $_POST['item_number'];
  //merchant's email i.e. paypal account
  //equals business in paynow.html
  $receiver_email = $_POST['receiver_email'];
  //create a mysqli connection
  $con = new mysqli($hostname, $username, $password, $databasename, 3306,
     "/var/lib/mysql/mysql.sock");
  //check merchant email, price & not recycled txn id
  //no need to change syntax to pass object by reference
  $valid_txnid = check_txnid($con, $txn_id);
  $valid_price = check_price($con, $price, $inventoryid);
  $valid_email = check_email($receiver_email);
  //if all checks write record
  if($valid_price && $valid_email && $valid_txnid)
  {
    //update database with txn id
    $strsql = "UPDATE tblorders SET txnid = '$txn_id' ".
      "WHERE orderid = $orderid";
    $con->query($strsql);
    $message ="Successful, transaction id: $txn_id\n";
  }
  else
  {
    //unsuccessful transaction
    $message ="Unsuccessful, transaction id: $txn_id\n";
  }
}
else if($status == "INVALID")
{
  //notify suspicious transaction
  $message ="Suspicious IPN with transaction id: $txn_id";
}
else
{
  //deal with other types
  $message ="Incomplete purchase with transaction id: $txn_id";

}
mail ("notify@myisp.com", "PayPal", $message);
?>
```

Running the Hack

First you will need a PayPal account. Create one by going to the PayPal home page and signing up for a business account.

Then you need to alter the files to your specifications. Your *buynow.html* file will of course reflect the product you are selling. You will also need to change the email addresses in both the *buynow.html* file and the *verifypurchase.php* file. Replace "seller@myisp.com" with the email address associated with your PayPal account. This is important because it identifies the account that will receive payment. Change "notify@myisp.com" to the appropriate address for receiving confirmation of payment. You may not need a payment confirmation at all or you may want to replace it with code to write a log file, especially in the case of a failed payment. Change the *connection.php* file to reflect values appropriate to your MySQL server. No changes are required for the *presubmit.php* file unless you change the database structure.

You will doubtless create a database suited to your specific business needs but, if you wish to test this code as is, here are the SQL statements that will create the minimum required database structure:

```
 CREATE TABLE `tblbooks` (
  `inventorynumber` int(11) NOT NULL auto_increment,
  `title` varchar(150) NOT NULL default '',
  `author` varchar(100) NOT NULL default '',
  `cost` float(6,2) NOT NULL default '0.00',
  `listprice` float(7,2) NOT NULL default '0.00',
  `publicationdate` varchar(4) default NULL,
  `publisher` varchar(4) NOT NULL default '',
  PRIMARY KEY  (`inventorynumber`),
  KEY `authidx` (`author`),
  KEY `titleidx` (`title`),
) ENGINE=MyISAM DEFAULT CHARSET=latin1

CREATE TABLE `tblorders` (
  `orderid` int(11) NOT NULL auto_increment,
  `customerid` int(11) default NULL,
  `orderdate` date default NULL,
  `txnid` varchar(17) default NULL,
  PRIMARY KEY  (`orderid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1

CREATE TABLE `tblorderitems` (
  `orderid` int(11) NOT NULL default '0',
  `inventorynumber` int(11) NOT NULL default '0',
  PRIMARY KEY  (`orderid`,`inventorynumber`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

Next, upload the files to your server ensuring that the *connection.php*, *buynow.html* and *presubmit.php* files are all in the same directory. You can put the *verifypurchase.php* file in the same directory as well but it's probably better off in its own directory. If you do put it in a separate directory be sure to change the include path for the *connection.php* file.

Go to your PayPal account, turn on IPN and enter the fully qualified URL for the *verifypurchase.php* file. To make a purchase point your browser at *buynow.php*. You will know that everything is working when you click on the "Buy Now" button, are taken to

the PayPal site and, when payment is complete, you then receive an email containing the transaction id.

Hacking the Hack

One size never fits all. In this particular case the price of the individual item purchased is identical to the total price. However, in many cases, shipping charges may need to be added and different currencies taken into account. Such changes can easily be accommodated by adjusting the “check_price” function.

Use of mysqli is not a requirement although it is apparent how prepared statements might be a real advantage especially when processing a shopping cart rather than a single item.

Signing up for a PayPal developer account makes sense for someone who regularly develops PayPal applications. It certainly is an advantage to use the PayPal sandbox to test applications before going live. This is especially important with an application like this one which depends upon a hidden post. And finally , “How do you debug a web page that you never see?” – by emailing the variable that holds the data you reposted to PayPal. Email this value and you should see something like the following:

```
mc_gross=24.95&address_status=confirmed&payer_id=TYWM55XFZCN8S
&tax=0.00&address_street=36+Main+Street
&payment_date=16%3A00%3A32+Aug+11%2C+2005+PDT
&payment_status=Completed&charset=windows-1252&address_zip=12345
&first_name=Peter&mc_fee=0.82&address_country_code=US
&address_name=Peter+Buyer&notify_version=1.9&custom=20
&payer_status=unverified&business=seller%40isp.com
&address_country=United+States&address_city=Toledo&quantity=1
&verify_sign=ACUe-E7Hjxmeel8FjYAtjnx-yjHAAVhtx75Yq6UdimmRaeJhnewr0ugZ
&payer_email=buyer%40myisp.com&txn_id=1E044782YK461110T
&payment_type=instant&last_name=Buyer&address_state=OH
&receiver_email=seller%40isp.com&payment_fee=0.82
&receiver_id=JEFVKNSSDLTBL&txn_type=web_accept
&item_name=Object+Oriented+PHP&mc_currency=USD
&item_number=06734980548&test_ipn=1&payment_gross=24.95
&shipping=0.00&cmd=_notify-validate
```

Do this and you can confirm all the name/value pairs posted.

About the Author

Peter Lavin runs a Web Design/Development firm in Toronto, Canada. He has been published in a number of magazines and online sites, including UnixReview.com, php| architect and International PHP Magazine. He is a contributor to the recently published O'Reilly book, PHP Hacks and is also the author of Object Oriented PHP, published by No Starch Press.