Erich Viebrock

ECPE 178

Lab #9: Vulnerability Management

1. **Inet addr:**

   10.0.2.15

2. **Try to identify at least two risk areas where a malicious SQL query can be injected.**

   a. Under the quick item search, a potential threat could be a SQL search of ‘OR 1=1 OR’, which would return all entries in the database. If there is any information an administrator would want to hide from users, this would obviously be a problem.

   b. Under “Login/Register”, a potential threat could be a SQL injection of ‘OR 1=1 OR’ in the email address entry. By doing so, an attacker may be able to breach the site with a back end account.

3. **Which user has Badstore.net authenticated you as?**

   Test User.

4. **After the above SQL injection, which user has BadStore.net authenticated you as?**

   Test User.

5. **Take a screen shot of the Badstore home page showing the above user login privilege.**

6. **What is the security implication that someone could use a SQL injection above to login Badstore?**

   If a “Test User” possesses any additional abilities than a typical user, this could pose a threat.

7. **What is the response from the web site after SQL injection?**

   UserID and Password not found!

   Use your browser’s Back button and try again.

8. **Please list the SQL query below.**

   SELECT itemnum, sdesc, ldesc, price FROM itemdb WHERE “ IN (itemnum,sdesc,ldesc)

9. **Based on this SQL query, what value can be injected to list the entire item database?**

   I stopped at the previous question, and started this question a couple days later. However, for some reason, I am no longer able to login as the “Test User” anymore. When I try to sign in with the email address ‘OR 1=1 OR’, with the password field left blank, I still remain an unregistered user. I’ve tried erasing the browser history and cache, but still have had no luck. Being I cannot sign in as the Test User, I cannot perform the SQL inject to return all the entries in the database, but I figure it would be something along the lines of ‘OR 1=1 OR’.

10. **What is the range of item numbers after you performed the SQL injection?**

    1-xxx, with no missing numbers. The entry should return all values in the database, including hidden entries.

11. **What is the risk for a web application when it shows its SQL query to users?**

    It provides a clue to users for how to return all entries in a database.

12. **What should an application developer do to mitigate the risk mentioned above?**

    Do not display the SQL query to users if an incorrect search is performed.

13. **What is the response from the web site after SQL injection?**

    No items matched your search criteria: Please change your search criteria.

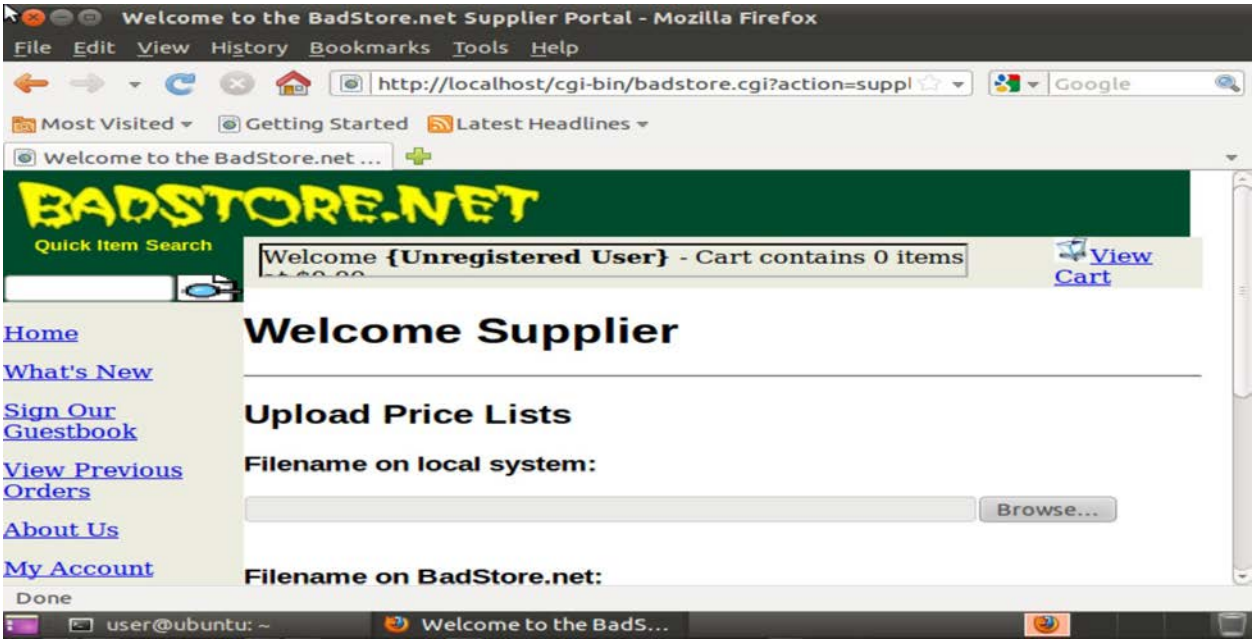
14. **Explain the differences between the original search query and the modified search query. Also explain how the modification can fix the SQL injection flaw.**

    The modified query is no longer hard-coded to reveal the SQL injection parameters. This no longer allows a typical user access to this information.

15. What injection string can you authenticate you as a supplier?

'OR 1=1 OR'

16. Paste the screenshot that the Badstore site authenticates you as a supplier.



17. Original Query:

```
### Connect to the SQL Database ###
my $dbh = DBI->connect("DBI:mysql:database=badstoredb;host=localhost", "root", "123456",
{'RaiseError' => 1})
        or die "Cannot connect: " . $DBI::errstr;
### Prepare, Evaluate and Execute SQL Query ###
my $sth = $dbh->prepare("SELECT * FROM userdb WHERE email='$email' AND passwd='$passwd'
");
eval {
        $sth->execute();
        1;
} or do {
        print "Location: /cgi-bin/badstore.cgi?action=supplierlogin\n\n";
};
```

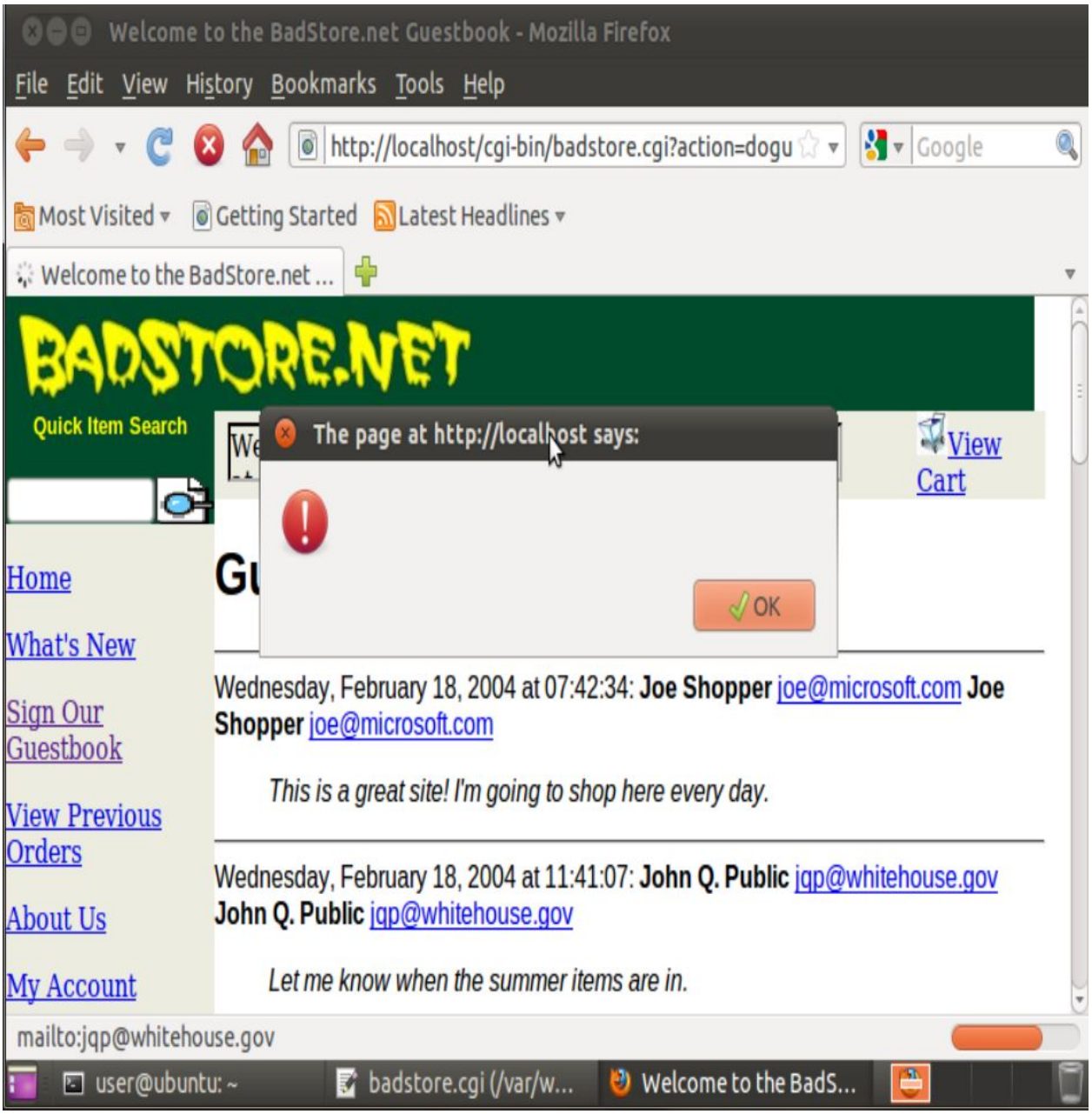
18. Modified Query:

```
my $sth = $dbh->prepare("SELECT * FROM userdb WHERE email=? AND passwd=? ");
eval {
        $sth->execute($email, $passwd);
        1;
} or do {
        print "Location: /cgi-bin/badstore.cgi?action=supplierlogin\n\n";
};
```

19. What is the response from the web site after the SQL injection?

UserID and Password not found!

20. Paste a screen shot of the web site with the alert window.



Waited a couple minutes, browser never got any farther than the screenshot.

21. What is the security impact of the XSS attack above?

It allows users to add Javascript to the guestbook, which can do just about anything to any user on the guestbook page. In this case, sends each user a notification window of their shopping cart.

22. **In additional to deleting malicious input in the database, Please describe what else can a web master do to prevent such an XSS attack.**

    Disallow users to enter text like `<script>` and `</script>`, and any syntax from other scripting languages.

23. **Does the sanitizing code delete the malicious script? What has happened after you enter the malicious script?**

    Yes, sanitizing the code deleted the malicious script. I was able to send a guestbook entry, but no text was visible. This leads me to assume the `<script>` … `</script>` section was deleted by the Perl script.

24. **Does the sanitizing code delete the malicious script? What has happened after you enter the malicious script?**

    No, this method of sanitizing the code did not work for me. I was still able to submit the `<script>` … `</script>` command, undetected. Here's what I changed the .cgi file to:

```
        ###Convert script character into HTML safe characters, for example < should be changed
to &lt; and > should be changed to &gt.
        $comments=~ s/</\&lt\;/isg;
        $comments=~ s/>/\&gt\;/isg;
```

    And yes, I tried the `<script>` … `</script>` command after saving the .cgi file.

25. **Compare the two different ways of sanitizing the user comments.**

    The second method is much more effective than the first. The first command blocks all Javascript attacks, whereas the second blocks all Javascript attacks and any other attacks that use the '<' or '>' characters. Of course, users may become annoyed they won't be able to use them to compare things, but who cares about the users' feelings. No one said network security was a place to make friends.