CS 361S - Network Security and Privacy
Spring 2014

Project #1

Due: 11:00am CST, February 27, 2014

Submission instructions

Follow the instructions in the project description.
If you are submitting late, please indicate how many late days you are using.

Collaboration policy

This assignment can be done individually or in two-person teams. Any cheating (*e.g.*, submitting another person's work as your own, or permitting your work to be copied) will automatically result in a failing grade. The Computer Science department code of conduct can be found at `http://www.cs.utexas.edu/undergraduate-program/code-conduct`.

Late submission policy

This project is due at the **beginning of class** on **February 27**. All late submissions will be subject to the following policy.

You start the semester with a credit of 3 late days. For the purpose of counting late days, a "day" is 24 hours starting at 2pm on the assignment's due date. Partial days are rounded up to the next full day. You are free to divide your late days among the take-home assignments (3 homeworks and 2 projects) any way you want: submit three assignments 1 day late, submit one assignment 3 days late, *etc.* After your 3 days are used up, no late submissions will be accepted and you will automatically receive 0 points for each late assignment.

Project #1 (50 points + 7 bonus points)

The objective of this project is to give you hands-on experience implementing attacks against vulnerable Web applications. You will do this project on a virtual machine using VMware Player.

You will need:

- The VMware Player:
  `http://www.vmware.com/products/player/`
- The project image:
  `http://www.cs.utexas.edu/~shmat/courses/cs361s/cs361s-proj1.tgz`

Getting Started

1. Download both the VMware Player and the project tarball. Uncompress the tarball and run the vm with VMware Player.
2. If VMware Player asks whether you moved or copied the image, say that you copied it.
3. The machine has two accounts: root/root and user/user. You will do your work as user, but feel free to explore as root.

The vm is configured to use NAT for networking. From the vm, you can type `ifconfig` as root to see the IP address of the vm. It should be listed in the field `inet addr:` under `eth0`.

The vm also has an SSH server. You can SSH into the vm from your machine, using the IP address produced by `ifconfig` (see above) as the destination. You can also use this to transfer files into the vm using `scp`. Alternatively, inside the vm, you can fetch files directly from the Web using `wget`.

Some attacks will require an email to be sent to `user` on the system. You will need a server-side script to automatically email information captured by your client-side JavaScript. We have provided this script for you at `http://hackmail.org/sendmail.php` (open this URL from within the vm for more instructions) and use that URL in your attack scripts to send emails. Any mail the user receives will appear in `/var/mail/user`.

Interacting with the vm

Using SSHFS / X Tunneling (recommended)

Probably the least painful way to interact with the vm is through local networking.

1. Install the `sshfs` package on your *host* machine and create an empty `netfs` directory.

2. Run `sshfs root@[vm IP address]:/ netfs` and enter the vm's root password. You can now access the vm's filesystem via the `netfs` directory with your host machine's applications.

3. Log in to the vm via `ssh -Y user@[vm IP address]`. You can now run graphical applications such as iceweasel within the vm.

Good old-fashioned logging in (slow and annoying)

If you're determined to do this the painful way and interact with the VMware Player window:

1. You may find it beneficial to install vmware-tools in the vm. VMware Player will prompt you about this.

2. Go into the desktop environment with `startx`.

3. To start a Web browser, type `iceweasel &`.

4. You can change the display resolution in the virtual machine with xrandr, *e.g.*:
   `xrandr -s 19`

5. If you want to install other packages / a friendlier desktop environment, you can use `apt-get`.

UT Payroll

UT Payroll is all about making sure people get paid while doing the least amount of work possible. To that end, they've created a Web application that lets you set up your direct deposit information, replacing the six hardcopy forms they had previously. Because they find themselves frequently looking up the name attached to a UT EID, they have included that functionality as well.

The UT Payroll server is located at this URL (accessible only from within the vm):

`http://payroll.utexas.edu`

You can create a new account by registering on the main page. You can then save your account number and routing number (this should be obvious, but please please please do not save your *actual* banking information, or use your actual UT Direct password when registering).

You can view any registered user's name by looking up their UT EID on the right.

The source code of the Payroll website is available within the vm in `/var/payroll/www`

Attack #1: Cross-site request forgery (10 points + 2 bonus points)

Create a malicious HTML page that should work as follows. Suppose the victim has logged into the UT Payroll server, and, while still logged in, visits your HTML page. Your page should overwrite the victim's account number and routing number stored on the UT Payroll server with your own values: 3133731337 and 1000000001 respectively.

Important: The victim should be redirected to the UT Payroll website immediately. In particular, he should not see the URL or the content of the malicious HTML page. (It is Ok if the browser displays your malicious page for a fraction of a second before it finishes fetching the UT Payroll page.)

Bonus (2 points)

Instead of redirecting the victim as described in the previous paragraph, make the attack transparent to the victim. In this case, the victim should see only the URL and content of your malicious HTML page. For example, the victim is browsing his favorite forum and sees your link promising a cute picture of a kitten. He clicks your link, sees the kitten, nods appreciatively, then closes the tab, unaware that his data at UT Payroll has been modified.

Attack #2: Cookie theft (15 points + 5 bonus points)

A user named `victim` has logged into the UT Payroll server. Create a URL that looks like this (with `EVILMAGIC` replaced by your exploit):

`http://payroll.utexas.edu/account.php?eid=EVILMAGIC`

When the logged in victim visits this URL, the victim's UT Payroll cookie should get sent by email to `user`.

The user should notice no difference in the behavior or appearance of the web page compared to simply typing a username into the text box on `http://payroll.utexas.edu/account.php` and hitting Enter. The source of the page can be arbitrarily different, but it should look and feel exactly the same.

Important: While you can technically satisfy the wording of the problem by redirecting the user to `http://payroll.utexas.edu/account.php?eid=victim` after stealing the cookie, this is not what we are looking for. You **must** exploit the way that the username variable is used in the PHP script.

In particular, your attack code must:

- Pull the victim's record from the database using the SQL query on line 30 of `account.php` (therefore, SQL must not barf on being given a query constructed from the username part of your URL).

- Result in the correct username (`victim`) being displayed in the input field on the user page. Thus, when the PHP code spits back the username you gave it on line 100 of `account.php`, it must somehow render as `victim`. It should be **exactly** that string—you cannot have more text hidden beyond the whitespace in the input box.

- Display the user's EID and name in the area below. Your code should also somehow ensure that even though the username you supplied is a long and ugly string, it should render as `victim` in this part of the page as well.

To summarize, you attack should, without redirection, result in a page that looks exactly like the page `http://payroll.utexas.edu/account.php?eid=victim`

The HTML source will be different, and so will the address bar (it will be your malicious URL) but the content of the page should look and behave the same.

If you choose to use a client-side script in addition to a malicious URL, you can use your UT webspace or any other webspace available to you (such as `http://pastebin.com`) to host that script.

Tip: You are allowed to hardcode the string `victim` wherever you want. You cannot, however, hardcode the value of the name; it should be retrieved from the database. You will probably need to understand and exploit the manner in which the value of the name is encoded into the HTML page and how the JavaScript retrieves it.

Partial credit: If you are not able to email the cookie, at least display it in a pop-up alert. If you are not able to make the page look exactly the same, make it look approximately the same. At the very least, try to make sure that your URL does not result in the "This UT EID is not registered" warning. While some point will be given for simply sending the email / alerting the cookie, the majority of the points for this question are for cleaning up the display after the email has been sent.

Bonus (5 points)

The team with the **shortest** URL that implements the full attack #2 gets 5 bonus points.

Attack #3: Password theft (10 points)

Create a malicious HTML page that should work as follows. Assume your victim is not logged in. Upon visiting your page, the victim should be redirected to `http://payroll.utexas.edu/`. When the victim enters a username and password and hits "Log in", an email should be sent to `user` containing the username and password entered by the victim.

Important: Assuming a valid username/password pair was entered, the next page should look as if the victim did indeed log into UT Payroll.

Attack#4: SQL injection (15 points)

Create an HTML page that the tester will open in his browser. The tester will not be logged in. The HTML page should have a form with a single text field and a submit button (note: the form should not ask the tester for a password). The tester will type a username into the text field and submit the form. You can assume the username submitted by the tester is already associated with a registered account.

As a result, the tester should be logged in as the user whose username he submitted. The browser's location bar should be `http://payroll.utexas.edu/account.php`, and the page should function exactly as if the correct username and password were entered on the real site.

Deliverables

You will submit your project with `turnin`. The grader is `ojensen` and the project name is `proj1`. Make sure you include a file for each attack:

- Your malicious HTML page(s) (the entire page, not just the URL) implementing attack #1.
- Text file with your malicious URL implementing attack #2.
- Your malicious HTML page implementing attack #3.
- Your malicious HTML page implementing attack #4.

Your submission should also include a file `SUBMISSION`. The first line should state how many late days were used (if any). Then give the following on a single line, one for each student:

- your UTEID
- your UTCS username
- your real name

You may want to include a `README` file with comments about your experiences or suggestions for improvement.