

CS 378 - Network Security and Privacy

Spring 2012

Project #1

Due: 2:00pm CST, March 6, 2012

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 **March 6**. 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 + 5 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/cs378/cs378_boxes.tar.bz2

Getting Started

1. Download both the VMware Player and the project tarball. Decompress the tarball and run the virtual machine 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`.
4. Go into the desktop environment with `startx`.
5. To start a browser, type `iceweasel &`.

Should you need any other packages to do your work (*e.g.*, `emacs`), you can install it with the command `apt-get` (*e.g.*, `apt-get install emacs`).

The vm is configured to use NAT for networking. From the virtual machine, 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 onto the virtual machine using `scp`. Alternatively, you can fetch files directly from the web on the vm using `wget`.

The source code of the KeyRing website is available at `/var/keyring/www`

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://keymail.org/sendmail.php> (open this URL from within the virtual machine 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`.

KeyCorp

KeyCorp Security, Inc. is all about the keys. Our state-of-the-art KeyRing service allows a group of friends to share their public keys with each other using a central repository.

The KeyRing server is located at this URL:

`http://keyring.org`

You can create a new account by registering on the main page. You can share your public key by typing it into the input box and saving the key. When you edit your key, the server strips out all characters from the key except digits and lower-case letters from 'a' to 'f'. The server also ensures that the key is no longer than 256 characters.

You can view anybody's public key by submitting their username.

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

Create a malicious HTML page that should work as follows. Suppose the victim has logged into the KeyRing server, and, while still logged in, visits your HTML page. Your page should overwrite the victim's public key stored on the KeyRing server with a hex-encoded 256-character random value.

Important: The victim should not see the URL or the content of the malicious HTML page. Assuming that the victim clicks on a link from a page located at URL X to the malicious page located at URL Y , the victim should see only a page on the KeyCorp website, **not** the address or content of the Y page. (It is Ok if the browser displays Y for a fraction of a second before it finishes fetching the KeyCorp page.)

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

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

`http://keyring.org/users.php?user=EVILMAGIC`

When the logged in victim visits this URL, the victim's KeyRing 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://keyring.org/users.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://keyring.org/users.php?user=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 lines 20-21 of `users.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 13 of `users.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 name and key 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, your attack should, without redirection, result in a page that looks exactly like the page `http://keyring.org/users.php?user=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.

This attack requires a client-side script in addition to a malicious URL. You can use your UT webspace or any other webspace available to you to host that script.

Tip: You are allowed to hardcode the string `victim` wherever you want. You cannot, however, hardcode the value of the key; it should be retrieved from the database. You will probably need to understand and exploit the manner in which the value of the key 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 “Cannot find that user” warning.

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://keyring.org/index.php`. 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 KeyCorp.

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://keyring.org/index.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 `jyang` and the project name is `proj1`. Make sure you include a file for each attack:

- Your malicious HTML page (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.