UNIVERSITY OF MARYLAND
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ENEE 459C
Computer Security
Instructor: Charalampos Papamanthou

Homework 5

Out: 11/25/15 Due: 12/11/15

Instructions

1. Strictly adhere to the University of Maryland Code of Academic Integrity.
2. **Type** and submit your solutions as a pdf document at Canvas. Include your full name in the solutions document. Name the solutions document as x-hw5.pdf, where x is your last name.
3. No extensions will be given. Everything will be due at 11.59pm on 12/11/15.

Problem 1 (20 points)

Suppose a new computer virus, called H1NQ, just got released. Mike has a new malware-detection program, QSniffer, that is 95% accurate at detecting H1NQ. That is, if a computer is infected with H1NQ, then QSniffer will correctly detect this fact 95% of the time, and if a computer is not infected, then QSniffer will correctly detect this fact 95% of the time. It turns out that the H1NQ virus will only infect any given computer with a probability of 1%. Nevertheless, you are nervous and run QSniffer on your computer, and it unfortunately says that your computer is infected with H1NQ. What is the probability that your computer really is infected?

Problem 2 (20 points)

The TCP three-way handshake works as follows: When a client wants to establish a TCP connection with a server, it first sends a SYN message with sequence number x. Then the server replies with a SYN-ACK message that has a fresh sequence number y along with $x + 1$, and finally the client replies with an ACK message that contains the sequence number $y + 1$.

In class we talked about the SYN-flooding attack, which works as follows: A malicious client could send a very large number of SYN messages (say M such messages) to the same server from M different spoofed IPs. When the server receives SYN message i, it would allocate space in memory for this connection i, i.e., it would store the fresh sequence number y_i that it chooses for the respective SYN-ACK message i as long as the respective client IP and port ($i \in \{1, \ldots, M\}$). With SYN flooding, a server's resources will become exhausted since eventually all of its resources will be allocated for these half-open connections.

One of the proposed solutions to this problem is an approach called "SYN cookies". SYN cookies work by constructing a sequence number y for the SYN-ACK message that encodes information about the connection: Namely, when a server receives a SYN message, it replies with the appropriate SYN-ACK message (with the sequence number constructed as below) and then discards the state of the connection. If the client responds with the appropriate ACK message (i.e., with sequence

number $y + 1$), then the server can reconstruct the state of the connection from its encoding in the sequence number. The SYN cookie has the following components:

- (5 bits) A timestamp t that equals $current\_time$ modulo 32 (where $current\_time$ is in seconds);
- (3 bits) The maximum segment size m that the server would have stored in the SYN queue entry;
- (24 bits) A $\mathsf{MAC}_k(.)$ of (i) the server IP address and port number for the connection; (ii) the client IP address and port number; and (iii) the value t. Note that only the server knows secret key k.

1. Explain why it is important that the sequence numbers x and y are chosen at random.
2. Even if x and y are indeed chosen at random, what might be a problem when the connection is not over SSL?
3. Explain why SYN cookies help to mitigate SYN flooding attacks.
4. When SYN cookies are used, explain what the server needs to do when he receives an ACK message from the client. Recall that without SYN cookies, the server just needs to decrease the sequence number by one and compare it to the locally stored sequence number.
5. Explain why the cookie needs to contain the counter t.
6. Explain why the cookie needs to contain the client IP address.
7. What could an attacker do if he could forge the MAC used in a SYN cookie, namely if he could compute arbitrary MACs without having access to the secret key k?

Problem 3 (20 points)

In this assignment you are going to write a program (in the programming language of your preference) that implements Merkle hash trees (originally introduced by Ralph Merkle in this paper), as we discussed in class. Suppose n is a power of 2. Your program should implement the following functions:

1. $T \leftarrow \mathsf{setup}(f_1, f_2, \ldots, f_n)$. On input n files $f_1, f_2, \ldots, f_n$ (1 KB each) it outputs the Merkle hash tree T. Recall that a Merkle hash tree is a binary tree such that for each internal node v (that has children u and w) we store a hash $h_v = \mathsf{HASH}(h_u, h_w)$. If v is a leaf corresponding to file f_i we set $h_v = f_i$. You can use an existing implementation of SHA-2 for the implementation of HASH.
2. $\pi_i \leftarrow \mathsf{prove}(i, T)$. On input an index i and the Merkle hash tree T, it outputs a proof π_i for file f_i. Recall the proof is a collection of hashes along the path from i to the root r.
3. $\{\texttt{ACCEPT}, \texttt{REJECT}\} \leftarrow \mathsf{verify}(f_i, \pi_i, h_r)$. On input a file f_i, a proof π_i and the hash of the root h_r, it outputs either `ACCEPT` or `REJECT`.
4. $T' \leftarrow \mathsf{update}(f_i', T)$. On input an updated file f_i', it outputs the updated Merkle hash tree T'.

Run experiments and plot the time it takes to execute the algorithms $\mathsf{setup}()$, $\mathsf{prove}()$, $\mathsf{verify}()$ and $\mathsf{update}()$ for $n = 2^{10}, 2^{11}, 2^{12}, \ldots, 2^{20}$.

Problem 4 ***Web Tracking*** (20 points—*taken from SEED labs* `http://www.cis.syr.edu/~wedu/seed/`)

4 Lab Overview

Behavioral targeting is a type of online advertising where ads are displayed based on the users web-browsing behavior. The user leaves a trail of digital foot prints moving from one website to the other. Behavioral targeting anonymously monitors and tracks the sites visited by a user. When a user surfs internet, the pages they visit, the searches they make, location of the user browsing from, device used for browsing and many other inputs are used by the tracking sites to collect data. A user profile is created from the data and data-mined for an online behavioral pattern of the user. As a result when users return to a specific site or a network of sites, the created user profiles are helpful in reaching the targeted audience to advertise. The targeted ads will fetch more user interest, the publisher (or seller) can charge a premium for these ads over random advertising or ads based on the context of a site.

4 Lab Environment

You need to use our provided virtual machine image for this lab. The name of the VM image that supports this lab is called `SEEDUbuntu12.04.zip`, which is built in June 2014. If you happen to have an older version of our pre-built VM image, you need to download the most recent version, as the older version does not support this lab. Go to our SEED web page (`http://www.cis.syr.edu/~wedu/seed/`) to get the VM image.

4.1 Environment Configuration

In this lab, we need three things, are of which are already installed in the provided VM image: (1) the Firefox web browser, (2) the Apache web server, and (3) the `Elgg` web application. For the browser, we need to use the `LiveHTTPHeaders` extension for Firefox to inspect the HTTP requests and responses. The pre-built ubuntu VM image provided to you has already installed the Firefox web browser with the required extensions.

Starting the Apache Server. The Apache web server is also included in the pre-built ubuntu image. However, the web server is not started by default. You need to first start the web server using the following command:

```
% sudo service apache2 start
```

The `Elgg` Web Application. We use an open-source web application called `Elgg` in this lab. `Elgg` is a web-based social-networking application. It is already set up in the pre-built Ubuntu VM image.

Configuring DNS. We have configured the following URL needed for this lab. To access the URL , the Apache server needs to be started first:

URL	Description	Directory
http://www.wtlabelgg.com	Elgg web site	/var/www/webtracking/elgg
http://www.wtcamerastore.com	CameraStore	/var/www/webtracking/CameraStore
http://www.wtmobilestore.com	MobileStore	/var/www/webtracking/MobileStore
http://www.wtelectronicsstore.com	ElectronicStore	/var/www/webtracking/ElectronicStore
http://www.wtshoestore.com	ShoeStore	/var/www/webtracking/ShoeStore
http://www.wtlabadserver.com	ReviveAdserver	/var/www/webtracking/adserver

4.2 Clear History and cookies

Please follow the instructions to clear history and cookies from the Firefox browser.

1. Open Firefox browser, select History from the top menu, and click on Clear Recent History option from the menu. A window Clear All History pops up.
2. Select all the check boxes and click on Clear Now button in the pop up window. Close the Firefox browser, re open and start browsing.

4.3 Open a new private window in Firefox

Please follow the instructions to open a new private window in Firefox and start a private browsing session.

1. On the left desktop menu, Right Click on the Firefox icon, Select Open a New Private Window option.
2. New Private browsing Firefox window opens up, start browsing in that private browser.

4 Lab Tasks

4.1 Task 1: Understand the basic working of the web tracking

Nowadays the online web user tracking helps in displaying ads to the targeted audience. When a user visits a website, there are certain ads, of which some of them are targeted advertisements. Say a user visits a certain product in an E-commerce website, he visits the product multiple times, checks the reviews and reads more about the product. Sometime later when the user visits another website, to his surprise he finds the previously visited product is displayed as an advertisement.

The objective of this task is to understand the basic working of the web tracking. In this task you need to open the E-commerce websites, view details of one or more products. Once you login to the Elgg website you should see the most visited product displayed as an advertisement.

1. Open Elgg website without visiting any website and describe your observation in the lab report.
2. Open Firefox and open the CameraStore, MobileStore, ElectronicStore and ShoeStore websites.
3. Click on view details for any products in the websites.
4. Refresh the Elgg website in Firefox and describe your observation.
5. Close the browser, reopen it and browse the Elgg website. Describe your observation.

Note: If you want to repeat the observations for step 1, clear the Browsing History and Cookies from the Firefox browser. Please follow the instructions to clear history and cookies in section 2.3

4.2 Task 2: Importance of cookie in Web tracking

Cookies are created when a user's browser loads a particular website. The website sends information to the browser which then creates a text file. Every time the user goes back to the same website, the browser retrieves and sends this file to the website's web server. Computer Cookies are created not just by the website that the user is browsing but also by other websites that run ads, widgets, or other elements on the web page which are being loaded. These cookies regulate the ad display and functioning of other elements on the web page.

The objective of this task is to understand the importance of cookie in web tracking. In this task you need to identify the tracking cookie using the LiveHTTPHeaders in Firefox. Please follow the steps below and give your observation.

1. Open any one of the E-commerce websites CameraStore, MobileStore, ElectronicStore and ShoeStore.
2. Click on view details for any product in websites and capture LiveHTTPHeader traffic.
3. In LiveHTTPHeaders, identify the HTTP request, which set the third party cookies, and take the screenshot.
4. Right click on the productDetail page and select View Page Source. Find out how the request for tracking cookie is sent from the webpage, please take a screenshot and describe your observation.

Third party cookies are cookies that are set by web site with a domain name other than the one the user is currently visiting. For example, user visits website abc.com, say the web page abc.com has an image to fetch from xyz.com. That image request can set cookie on domain xyz.com, and the cookie set on xyz.com domain is known as a third-party cookie. Some advertisers use these types of cookies to track your visits to the various websites on which they advertise.

The objective of this task is to understand how third party cookies are used in web tracking. In this task you need to identify the third party cookie using Firebug (Firefox browser extension, which is present in right corner of the browser.) and record your observations. Please strictly follow the steps below and give your observation.

1. Open any one of the E Commerce websites CameraStore, MobileStore, ElectronicStore, ShoeStore and view details for any product.
2. Open the ad server web page http://www.wtlabadserver.com.
3. Open Firefox extension Firebug. Observe the Firebug in ad server web page and product web page. Switch between the products webpage and ad server webpage. Describe your observation. (Please do NOT reload the products webpage).

Identify the third party cookie used for tracking in Firebug extension. Describe your observations in the report and explain why is it called a third party cookie? Give reasons and screenshots to support your observation.

Note: If you wish to redo the task from beginning, please delete history and cookies from your Firefox browser.

4.3 Task 3: Tracked user interests and data

The ad servers update their database from users browsing history. They keep track of the web pages visited, articles read, videos watched and any other footprints which user can provide. The objective of this task is to figure out the user interests and view the logged user impressions. In this task you need to understand that all the products viewed by you will be logged in the ad server database. Please follow the steps below and give your observation.

1. Open the E Commerce websites `CameraStore`, `MobileStore`, `ElectronicStore` and `ShoeStore`.
2. Click on view details for any product in the website.
3. Open `www.wtlabadserver.com/preferences.php` in a new tab and observe the webpage.

Explain how the user impressions are logged in ad server database, and how is it mapped to a user. Give evidences to support your observation.

4.4 Task 4: How ads are displayed in website

The ad servers use the user profile (browsing history, recent product visits) to display the advertisements and now that the `cookie` is set to track the user, the ad servers display the targeted advertisements.

In this task you need to observe how the ad is rendered and displayed in the website. Please follow the steps below and give your observation.

1. Open the `Elgg` website in Firefox browser.
2. Capture and observe the `LiveHTTPHeader` traffic of the `Elgg` website, identify the HTTP requests which are from a different `domain` (third party).

Explain in detail how the `Elgg` website displays the targeted ads of the user. Provide evidences to support your explanation. (Hint: Use the table displayed in Task3 and `LiveHTTPHeader` traffic in Task2).

4.5 Task 5: Tracking in a Private browser window

In `InPrivate` browsing the browser stores some information such as `cookies` and temporary Internet files so the webpages you visit will work correctly. However, at the end of your `InPrivate` browsing session, this information is discarded. Once the `InPrivate` browser is closed the `cookies` are cleared, and temporary internet files are deleted for that session.

The objective of this task is to understand the working of the web tracking in a private browser window. In this task you need to open the E-commerce websites, view details of one or more products. Once you login to the `Elgg` website (in the same private browser) you should see the most visited product displayed as an advertisement.

1. Open `Elgg` website without visiting any website and describe your observation in the lab report.
2. Open Firefox and open the `CameraStore`, `MobileStore`, `ElectronicStore` and `ShoeStore` websites.
3. Click on view details for any products in the websites.
4. Refresh the `Elgg` website in Firefox and describe your observation.

5. Close the InPrivate browser, reopen it and browse the Elgg website. Describe your observation.

Compare your observations with Task1. Explain the reasons and provide evidence to support your observations.

Note: Please follow the instructions in section 2.3 to open a new private window in Firefox.

4.6 Task 6: Real world tracking

The web tracking in real world involves many ad servers, each ad servers have their own technique of tracking the user interests. In this task you need to visit any of the websites given below and identify the web requests which are sent to the ad servers using the LiveHTTPHeaders in Firefox. The websites are:

1. http://dictionary.reference.com
2. http://www.amazon.com
3. http://www.careerbuilder.com

Open the websites, observe the HTTP request and response in LiveHTTPHeaders. Capture screenshot of one HTTP request to the real world ad server for each web site. Also identify the third party cookie used for that HTTP request.

4.7 Task 7: Countermeasures

There are certain countermeasures for the web tracking but most of the websites wont work properly after implementing the counter measures. Most of the websites are highly dependent on JavaScript and third party cookies. You must have observed that the web tracking tasks are mostly dependent on the third party cookies.

The objective of this task is to understand the countermeasures. In this task you should disable the third party cookies in Firefox browser and figure out if your impressions are tracked. Please follow the steps below and give your observation:

1. Disable the third party cookies from the Firefox browser. Please follow the instructions of how to disable third party cookies in Firefox browser in https://support.mozilla.org/en-US/kb/disable-third-party-cookies.
2. After disabling the third party cookies, open the CameraStore, MobileStore, ElectronicStore, ShoeStore websites and LiveHTTPHeaders.
3. Click on view details for any products in the websites.
4. In LiveHTTPHeaders, identify the HTTP request, which set the third party cookies, and take the screenshot.
5. Open Elgg website and describe your observation. Also take the screenshot of HTTP request to ads server in LiveHTTPHeaders. Compare it with the HTTP request to ads server in Task 4 and explain the difference.

Also there are other ways to mitigate the web tracking. To opt out of targeted advertisement, add browser extensions like RequestPolicy, NoScript and Ghostery which control the third party requests from the web browser. Also one can keep cookies for the browsing session, by setting a cookie policy only keep cookies until I close my browser which will delete all the cookies after the browser window is closed.

Major web browsers provide with an option of `Do Not Track`, which is a feature to let third party trackers know your preference to opt out third party tracking, and it is done by sending a HTTP header for every web request. This `Do Not Track` preference may or may not adhered by the third party trackers. Some third party trackers provide with an option of `Opt Out` of targeted advertisement. Some of them may interpret "Opt Out" to mean "do not show me targeted ads", rather than "do not track my behavior online". You can check your tracked online profile created by Google in `www.google.com/settings/ads`. You can also find the Opt out option provided in the above Google URL.

4 Submission

You need to submit a detailed lab report to describe what you have done and what you have observed. Please provide details using `LiveHTTPHeaders`, `Wireshark`, and/or screen shots. You also need to provide explanation to the observations that are interesting or surprising.

References

[1] HTTP Cookie - Wikipedia. Available at the following URL: `http://en.wikipedia.org/wiki/HTTP_cookie`.

[2] New Cookie Technologies : Harder to See and Remove, Widely Used to Track you https://www.eff.org/deeplinks/2009/09/new-cookie-technologies-harder-see-and-remove-wide

[3] How Online Tracking companies know most of what you do online `https://www.eff.org/deeplinks/2009/09/online-trackers-and-social-networks`.

Problem 5 ***SQL Injection Attack*** (20 points—*taken from SEED labs `http://www.cis.syr.edu/~wedu/seed/`*)

5 Lab Overview

SQL injection is a code injection technique that exploits the vulnerabilities in the interface between web applications and database servers. The vulnerability is present when user's inputs are not correctly checked within the web applications before sending to the back-end database servers.

Many web applications take inputs from users, and then use these inputs to construct SQL queries, so the web applications can pull the information out of the database. Web applications also use SQL queries to store information in the database. These are common practices in the development of web applications. When the SQL queries are not carefully constructed, SQL-injection vulnerabilities can occur. SQL-injection attacks is one of the most frequent attacks on web applications.

In this lab, we modified a web application called `Collabtive`, and disabled several countermeasures implemented by `Collabtive`. As the results, we created a version of `Collabtive` that is vulnerable to the SQL-Injection attack. Although our modifications are artificial, they capture the common mistakes made by many web developers. Students' goal in this lab is to find ways to exploit the SQL-Injection vulnerabilities, demonstrate the damage that can be achieved by the attacks, and master the techniques that can help defend against such attacks.

5 Lab Environment

You need to use our provided virtual machine image for this lab. The name of the VM image that supports this lab is called `SEEDUbuntu12.04.zip`, which is built in June 2014. If you happen to have an older version of our pre-built VM image, you need to download the most recent version, as the older version does not support this lab. Go to our SEED web page (`http://www.cis.syr.edu/~wedu/seed/`) to get the VM image.

5.1 Environment Configuration

In this lab, we need three things, are of which are already installed in the provided VM image: (1) the Firefox web browser, (2) the Apache web server, and (3) the `Collabtive` project management web application. For the browser, we need to use the `LiveHTTPHeaders` extension for Firefox to inspect the HTTP requests and responses. The pre-built ubuntu VM image provided to you has already installed the Firefox web browser with the required extensions.

Starting the Apache Server. The Apache web server is also included in the pre-built ubuntu image. However, the web server is not started by default. You need to first start the web server using the following command:

```
% sudo service apache2 start
```

The `Collabtive` Web Application. We use an open-source web application called `Collabtive` in this lab. `Collabtive` is a web-based project management system. This web application is already set up in the pre-built ubuntu VM image. We have also created several user accounts on the `Collabtive` server. To see all the users' account information, first log in as the admin using the following password; other users' account information can be obtained from the post on the front page.

```
username: admin
password: admin
```

Configuring DNS. We have configured the following URL needed for this lab. To access the URL , the Apache server needs to be started first:

URL	Description	Directory
`http://www.sqllabcollabtive.com`	Collabtive	`/var/www/SQL/Collabtive/`

The above URL is only accessible from inside of the virtual machine, because we have modified the `/etc/hosts` file to map the domain name of each URL to the virtual machine's local IP address (`127.0.0.1`). You may map any domain name to a particular IP address using `/etc/hosts`. For example you can map `http://www.example.com` to the local IP address by appending the following entry to `/etc/hosts`:

```
127.0.0.1       www.example.com
```

If your web server and browser are running on two different machines, you need to modify `/etc/hosts` on the browser's machine accordingly to map these domain names to the web server's IP address, not to `127.0.0.1`.

5.2 Turn Off the Countermeasure

PHP provides a mechanism to automatically defend against SQL injection attacks. The method is called magic quote, and more details will be introduced in Task 3. Let us turn off this protection first (this protection method is deprecated after PHP version 5.3.0).

1. Go to `/etc/php5/apache2/php.ini`.
2. Find the line: `magic_quotes_gpc = On`.
3. Change it to this: `magic_quotes_gpc = Off`.
4. Restart the Apache server by running "`sudo service apache2 restart`".

5 Lab Tasks

5.1 Task 1: SQL Injection Attack on `SELECT` Statements

In this task, you need to manage to log into `Collabtive` at `www.sqllabcollabtive.com`, without providing a password. You can achieve this using SQL injections. Normally, before users start using `Collabtive`, they need to login using their user names and passwords. `Collabtive` displays a login window to users and ask them to input `username` and `password`. The login window is displayed in the following:

The authentication is implemented by `include/class.user.php` in the Collabtive root directory (i.e., `/var/www/SQL/Collabtive/`). It uses the user-provided data to find out whether they match with the `username` and `user_password` fields of any record in the database. If there is a match, it means the user has provided a correct username and password combination, and should be allowed to login. Like most web applications, PHP programs interact with their back-end databases using the standard SQL language. In `Collabtive`, the following SQL query is constructed in `class.user.php` to authenticate users:

```
[frame=single]
  $sel1 = mysql_query ("SELECT ID, name, locale, lastlogin, gender,
     FROM  USERS_TABLE
     WHERE (name = '$user' OR email = '$user') AND pass = '$pass'");

  $chk = mysql_fetch_array($sel1);

  if (found one record)
  then {allow the user to login}
```

In the above SQL statement, the `USERS_TABLE` is a macro in PHP, and will be replaced by the users table named `user`. The variable `$user` holds the string typed in the `Username` textbox, and `$pass` holds the string typed in the `Password` textbox. User's inputs in these two textboxs are placed directly in the SQL query string.

SQL Injection Attacks on Login: There is a SQL-injection vulnerability in the above query. Can you take advantage of this vulnerability to achieve the following objectives?

- **Task 1.1**: Can you log into another person's account without knowing the correct password?
- **Task 1.2**: Can you find a way to modify the database (still using the above SQL query)? For example, can you add a new account to the database, or delete an existing user account? Obviously, the above SQL statement is a query-only statement, and cannot update the database.

However, using SQL injection, you can turn the above statement into two statements, with the second one being the update statement. Please try this method, and see whether you can successfully update the database.
To be honest, we are unable to achieve the update goal. This is because of a particular defense mechanism implemented in MySQL. In the report, you should show us what you have tried in order to modify the database. You should find out why the attack fails, what mechanism in MySQL has prevented such an attack. You may look up evidences (second-hand) from the Internet to support your conclusion. However, a first-hand evidence will get more points (use your own creativity to find out first-hand evidences). If in case you find ways to succeed in the attacks, you will be awarded bonus points.

5.2 Task 2: SQL Injection on `UPDATE` Statements

In this task, you need to make an unauthorized modification to the database. Your goal is to modify another user's profile using SQL injections. In `Collabtive`, if users want to update their profiles, they can go to `My account`, click the `Edit` link, and then fill out a form to update the profile information. After the user sends the update request to the server, an `UPDATE` SQL statement will be constructed in `include/class.user.php`. The objective of this statement is to modify the current user's profile information in the `users` table. There is a SQL injection vulnerability in this SQL statement. Please find the vulnerability, and then use it to do the following:

- Change another user's profile without knowing his/her password. For example, if you are logged in as Alice, your goal is to use the vulnerability to modify Ted's profile information, including Ted's password. After the attack, you should be able to log into Ted's account.

5.3 Task 3: Countermeasures

The fundamental problem of SQL injection vulnerability is the failure of separating code from data. When constructing a SQL statement, the program (e.g. PHP program) knows what part is data and what part is code. Unfortunately, when the SQL statement is sent to the database, the boundary has disappeared; the boundaries that the SQL interpreter sees may be different from the original boundaries, if code are injected into the data field. To solve this problem, it is important to ensure that the view of the boundaries are consistent in the server-side code and in the database. There are various ways to achieve this: this objective.

- **Task 3.1: Escaping Special Characters using `magic_quotes_gpc`.** In the PHP code, if a data variable is the string type, it needs to be enclosed within a pair of single quote symbols (’). For example, in the SQL query listed above, we see `name = '$user'`. The single quote symbol surrounding `$user` basically "tries" to separate the data in the `$user` variable from the code. Unfortunately, this separation will fail if the contents of `$user` include any single quote. Therefore, we need a mechanism to tell the database that a single quote in `$user` should be treated as part of the data, not as a special character in SQL. All we need to do is to add a backslash (\) before the single quote.
  PHP provides a mechanism to automatically add a backslash before single-quote (’), double quote (`"`), backslash (\), and NULL characters. If this option is turned on, all these characters in the inputs from the users will be automatically escaped. To turn on this option, go to `/etc/php5/apache2/php.ini`, and add `magic_quotes_gpc = On` (the option is already on in the VM provided to you). Remember, if you update `php.ini`, you need to

restart the Apache server by running "`sudo service apache2 restart`"; otherwise, your change will not take effect.
Please turn on/off the magic quote mechanism, and see how it help the protection. Please be noted that starting from PHP 5.3.0 (the version in our provided VM is 5.3.5), the feature has been DEPRECATE[1], due to several reasons:

- Portability: Assuming it to be on, or off, affects portability. Most code has to use a function called `get_magic_quotes_gpc()` to check for this, and code accordingly.
- Performance and Inconvenience: not all user inputs are used for SQL queries, so mandatory escaping all data not only affects performance, but also become annoying when some data are not supposed to be escaped.

- **Task 3.2: Escaping Special Characters using** `mysql_real_escape_string`. A better way to escape data to defend against SQL injection is to use database specific escaping mechanisms, instead of relying upon features like magical quotes. MySQL provides an escaping mechanism, called `mysql_real_escape_string()`, which prepends backslashes to a few special characters, including `\x00`, `\n`, `\r`, `\`, `'`, `"` and `\x1A`. Please use this function to fix the SQL injection vulnerabilities identified in the previous tasks. You should disable the other protection schemes described in the previous tasks before working on this task.
- **Task 3.3: Prepare Statement.** A more general solution to separating data from SQL logic is to tell the database exactly which part is the data part and which part is the logic part. MySQL provides the prepare statement mechanism for this purpose.

```
$db = new mysqli("localhost", "user", "pass", "db");
$stmt = $db->prepare("SELECT ID, name, locale, lastlogin FROM users
                      WHERE name=? AND age=?");
$stmt->bind_param("si", $user, $age);
$stmt->execute();

//The following two functions are only useful for SELECT statements
$stmt->bind_result($bind_ID, $bind_name, $bind_locale, $bind_lastlogin);
$chk=$stmt->fetch();
```

Parameters for the `new mysqli()` function can be found in `/config/standard/config.php`. Using the prepare statement mechanism, we divide the process of sending a SQL statement to the database into two steps. The first step is to send the code, i.e., the SQL statement without the data that need to be plugged in later. This is the prepare step. After this step, we then send the data to the database using `bind_param()`. The database will treat everything sent in this step only as data, not as code anymore. If it's a `SELECT` statement, we need to bind variables to a prepared statement for result storage, and then fetch the results into the bound variables.
Please use the prepare statement mechanism to fix the SQL injection vulnerability in the `Collabtive` code. In the `bind_param` function, the first argument "`si`" means that the

[1] In the process of authoring computer software, its standards or documentation, `deprecation` is a status applied to software features to indicate that they should be avoided, typically because they have been superseded. Although deprecated features remain in the software, their use may raise warning messages recommending alternative practices, and deprecation may indicate that the feature will be removed in the future. Feature are deprecated- rather than immediately removed-in order to provide backward compatibility, and give programmers who have used the feature time to bring their code into compliance with the new standard.

first parameter (`$user`) has a string type, and the second parameter (`$age`) has an integer type.

5 Guidelines

Print out debugging information. When we debug traditional programs (e.g. C programs) without using any debugging tool, we often use `printf()` to print out some debugging information. In web applications, whatever are printed out by the server-side program is actually displayed in the web page sent to the users; the debugging printout may mess up with the web page. There are several ways to solve this problem. A simple way is to print out all the information to a file. For example, the following code snippet can be used by the server-side PHP program to print out the value of a variable to a file.

```
$myFile = "/tmp/mylog.txt";
$fh = fopen($myFile, 'a') or die("can't open file");
$Data = "a string";
fwrite($fh, $Data . "\n");
fclose($fh);
```

A useful Firefox Add-on. Firefox has an add-on called `"Tamper Data"`, it allows you to modify each field in the HTTP request before the request is sent to the server. For example, after clicking a button on a web page, an HTTP request will be generated. However, before it is sent out, the `"Tamper Data"` add-on intercepts the request, and gives you a chance to make an arbitrary change on the request. This tool is quite handy in this lab.

The add-on only works for Firefox versions 3.5 and above. If your Firefox has an earlier version, you need to upgrade it for this add-on. In our most recently built virtual machine image (SEEDUbuntu11.04-Aug-2011), Firefox is already upgraded to version 5.0, and the `"Tamper Data"` add-on is already installed.

5 Submission

You need to submit a detailed lab report to describe what you have done and what you have observed. Please provide details using `LiveHTTPHeaders`, `Wireshark`, and/or screen shots. You also need to provide explanation to the observations that are interesting or surprising.