Hello CETS,

I was referred to this e-mail address in order to request a Security Review on a web application I developed for the SEAS Technical Communication Program (TCP). The application, tentatively called “TCP Online”, serves as an appointment booking system and basic CRM tool for the TCP department to use in our interactions with SEAS students.

The current code is located in ~tcp/cgi-bin/booking\_system/ on the ENIAC cluster, and is also ZIPped and attached to this e-mail for your convenience. Because the code is extensive and a little tough to jump into, I am providing a summary of my approach to securing this app below.

Please let me know if you need a detailed “tour” of the code in order to assess the app’s security. The current running version of the site has “dummy” data that was used during testing. If you would like to play around with the app, logging in at <https://fling.seas.upenn.edu/~tcp/cgi-bin/booking_system> will give you the Student view of the system. To see the Director view or the Fellow view (See below for more details), you can uncomment lines 227 or 228 (respectively) in booking\_system/res/tcpdb.php to override the internal authentication.

**TCP Online Security Approach**

* **Authentication**
  + We rely on Penn WebLogin for authentication. The entire application resides in a directory with a .htaccess directive that requires valid Pennkey credentials.
    - TCP Online has 3 internal authentication levels: Penn Students, TCP Fellows (the TCP staff members that meet with students), and TCP Director. The system bases internal auth level on the user’s Pennkey, which is obtained by reading from the REMOTE\_USER environment variable. Fellow and Director Pennkeys are stored in our database, and all other users with a valid Pennkey are authenticated as Penn Students.
    - Every page checks the user’s internal authentication level before doing anything else, and gracefully handles “Access Denied” scenarios
    - Scripts, classes, and libraries that are meant to be include()s and dependencies of other pages are protected so that users accessing them independently will not result in any undesired effects or create any security holes.
* **Input Validation**
  + All user input (passed via GET URI params and POST data) is validated before being handled by each page. The application gracefully handles unexpected or malformed input.
  + For all pages that access appointment information, we ensure that the user has permission to view/modify the specific appointment before granting access to the associated appointment data.
    - **Example**: The Cancel Appointment page accepts an appointment ID (the primary key in our Appointments table) as a GET URI param. If a student changes the appointment ID to that of an appointment he or she did not book, the system will not allow this. This precaution is taken across our system.
* **SQL Injection Protection**
  + Most queries use vectors to protect against malformed inputs and SQL injection attacks
  + In addition to the aforementioned input parameter validation, there is also a layer of data sanitization for any input that will be used in a SQL query.
    - The sanitization process strips/escapes any characters that could be used in a SQL Injection attack.
    - If a MySQL query fails (this would only happen if the server is down or perhaps data is corrupted), for security purposes, MySQL error information ( mysql\_error() ) will not be displayed to the user.
* **Cross-Site Scripting (XSS) Prevention**
  + There exists only one instance in the application where an XSS exploit could even occur (Aside from features that are only Director-facing) – The user types in a comment when booking an appointment, and this comment is displayed to the TCP Fellow on the “View Appointment” page. This is circumvented by stripping this data of all characters that could be used to construct HTML tags/Javascript code. (This is actually accomplished via the normal data sanitation used to protect from SQL injections).
* **CRON Job Security**
  + TCP Online currently uses two CRON jobs: One automatically sends notifications of upcoming TCP appointments, and the other sends feedback forms afterwards. The CRON jobs were written as standalone scripts that do not rely on other classes/libraries in the web application. The scripts called by the CRON jobs are executed by the PHP interpreter, and permissions on these scripts are chmod’ed to 700 such that only the owner (the “tcp” SEAS account) and the cron daemon can execute the script.
* **FERPA Compliance**
  + Students are not required to use this system in order to book their appointments. If they do not want to submit any data or user information to our application, they do not have to – They can still book appointments by e-mailing us directly. (Previously, when we used a 3rd-party appointment booking solution, we had to offer this option in order to comply with FERPA regulations… Not sure if that still holds now that we have our own solution, but we offer it anyway.)

Thank you very much for your time, and please let me know what other information I can provide for you.

Best,

Zach Zarrow  
zzarrow@seas.upenn.edu