Security Analysis of Piazza

Yini Qi, Tania Yu, Tiange Zhan
{qyn, taniayu, tzhan}@mit.edu

MIT 6.857 Spring 2016 Final Report

May 11, 2016

Abstract

We performed a security analysis of Piazza's Q&A forum and Careers website and discovered some vulnerabilities that compromise students' privacy. We began by defining a formal security policy for the service and observing where existing implementation fails to meet these requirements. Then, we worked through various web security attacks including cross-site scripting, SQL injection, and cross-site request forgery. We analyzed both the Q&A portion of Piazza and the Careers site, finding many things that Piazza did correctly, as well as a few security issues that allowed us to access students' sensitive information.

1 Introduction

Piazza is a question-and-answer platform that allows users to publicly and anonymously ask questions, answer questions, and post notes in class forums. Over 1000 schools in 68 countries use Piazza for their classes. Piazza allows student and instructor account options in participating in class sites. If allowed by instructors on the class page, students can post anonymously to other students or to everyone. In their privacy statement, Piazza claims that their platform increases student participation, as students would be more comfortable posting anonymously to ask questions they would be afraid to ask in person. The two main goals of Piazza are to maintain the confidentiality and integrity of posts, so that users can receive accurate answers. In this project, we aim to analyze Piazza's success in accomplishing these goals.

Another component of Piazza is the Careers portion of the site. The Careers section allows students to upload their information for companies and recruiters to view. Piazza makes it easy for companies to filter by specific parameters to find particular types of students. Here, confidentiality is also important as student profiles may contain sensitive information that other students should not be able to access.

1.1 Architecture

The web application can be accessed at https://piazza.com, and it uses AJAX calls to dynamically load data on pages. For example, user profiles are loaded as frames within the page https://piazza.com/careers/dashboard#/companies. All requests are sent through HTTPS communication, which helps to provide much of Piazza's security. Users are authenticated by logging in with a valid username and password, and user cookies are stored in the browser after successful authentication.

1.2 Responsible Disclosure

Prior to beginning this project, we obtained permission from Piazza to perform a security analysis on the website. Because some of the findings reveal vulnerabilities that can still be exploited, results should not be published until everything has been disclosed to Piazza and they have had sufficient

opportunity to fix these issues. We are currently in contact with the team and will be disclosing all discoveries before the end of the Spring 2016 semester.

2 Security Policy

To understand the ways in which the security module of Piazza is broken, we must first define a security policy. We defined a list of principals and a list of permissible actions for each principal as follows:

1. Students
    - Can post questions and answers
    - Can edit students' posts
    - Cannot view private posts "only to instructors"
    - Cannot view poll names or results "only to instructors"
    - Cannot view post authors anonymous to students
    - Cannot view other students' private profile information
2. Teaching Assistants
    - Can post notes and announcements
    - Can edit students' and instructor posts
    - Can view private posts
    - Can view post authors anonymous to students
    - Can view poll names (for small classes) and results
    - Can add/remove students/TAs from class
    - Cannot view post authors anonymous to everyone
    - Cannot view students' private profile information
3. Instructors
    - Inherit all permissions of teaching assistants
    - Can create/remove class
    - Can edit class settings

- Can add/remove students/TAs/professors from class
- Cannot view students' private profile information

4. Companies
    - Can view and search all public profile information of students
    - Can send private messages to candidates

3 Existing Security Flaws

Before attempting any attacks, we found some existing configurations on Piazza that break the aforementioned security policy. These include misconfigured user access levels that allow students to gain access to instructor permissions, and possible malicious actions that compromise the integrity of posts.

3.1 Unauthorized Instructor Permissions

Although Piazza's terms of service state that only authorized instructors should be able to create new classes, we found that anyone with a institution email account can create classes for that institution. That means we can create fake class pages with the same names as the real ones and enroll students in them. There is also no name verification, so we can sign up using the name of an actual professor. Another security misconfiguration is that ability for students to sign up as TAs or professors in classes for which they are not. As long as the instructors do not disable the option for instructor self-enrollment, we could access the class as an instructor and gain instructor permissions. Although Piazza notifies all instructors when a new instructor signs up, it is up to the instructors to manually check if the user is an actual instructor. In the time that it takes the instructor to notice that the adversary is a fake instructor, the attacker could have already acquired private posts and the author's identities.

3.2 Unauthorized Post Manipulation

The integrity of post contents are also compromised as any student is able to edit any other student's questions and answers. Student answers that are

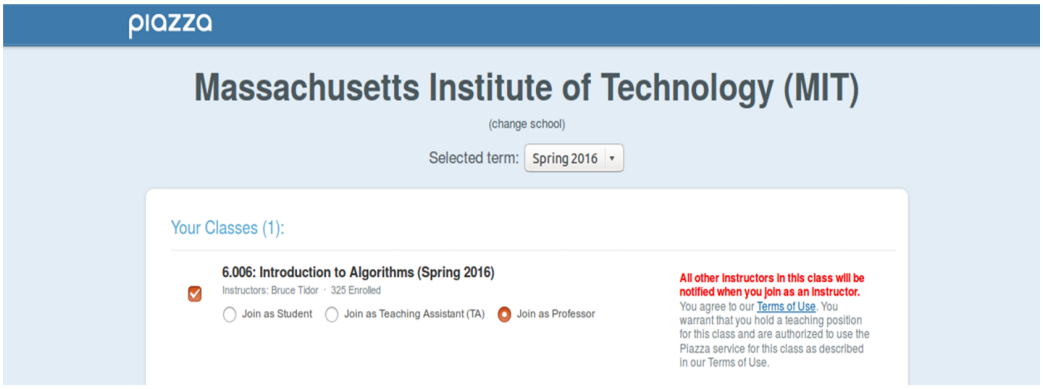

Figure 1: Students can join pages as Professors.

endorsed by instructors may later be edited to a false answer. This makes it easy for an adversary to supply misleading information to students under the guise of being instructor endorsed.

4 Attempted Attacks

Following the immediate security misconfigurations we found, we proceeded to try attacks from OWASP's top 10 list to test Piazza's confidentiality and integrity more rigorously. We found many things that Piazza did correctly to protect against these attacks, and a few vulnerabilities compromising their security policy [1].

4.1 Cross-Site Scripting

Cross-site scripting accounts for an overwhelming fraction of web vulnerabilities [4]. For this reason, we carefully checked user inputs for the possibility of injecting a script. On the Q&A Forum, users have the option to edit notes using either a plain text editor a rich text editor. Using the rich text editor, any HTML entities entered are properly escaped, so that entering `<script>alert('test')</script>` results in this whole string being displayed. With the plain text editor, users have the option to use HTML tags to format their posts. This does allow the possibility for a user to include a script within their post and have it executed when the "preview" option is clicked. However, Piazza is careful to define a whitelist of allowed tags, so

that any script tags and their contained content are stripped from the input on their server when the post is created. This means people can only run a script they wrote themselves, which is a negligible security issue.

The Careers site has a slightly more serious cross-site scripting issue that persists in the server. A student user can edit, for example, the "Personal Projects" box of his or her Careers profile. When the Careers page is loaded, all this profile information is loaded as a javascript object and assigned to a "MY_PROFILE" variable. A malicious user can thus enter something like `</script><script>alert('Attacked!')</script>`. When the change is submitted, the text box displays the input with script tags and their inner HTML removed, making it seem safe from XSS issues. However, any subsequent times the page is reloaded, the script executes. This again, is not a high security threat, because users can only XSS-attack themselves. If another user pulls up the attacker's profile, the attacker's information is not stored as a javascript object in the same way, and is instead directly displayed in the search profile modal, so that the normal user only sees the text with scripts stripped.

4.2 Exposing Private posts with CSRF

In an attempt to access students' private posts, we tried a Cross-Site Request Forgery (CSRF) attack. We created a fake website which attempted to send requests to Piazza on behalf of the currently logged in user, such as requests to create new content or endorse answers. Here, Piazza successfully defended from CSRF attacks. Per the same-origin policy, our fake website was not able to use the authentication of the real Piazza session to execute malicious requests. [7]

4.3 SQL Injection for User Login

We attempted to use SQL injection to break into instructor accounts. We first tried classical SQL injection such as inputting "1' or '1' = '1" into the password field. When various iterations of those queries had no effect, we used a tool called SQLmap [2] to automate the injection of more SQL queries. However, SQLmap was unable to detect the type of back end database used or successfully inject and parameters. Piazza did a good job protecting itself from SQL injection attacks by escaping user inputs and hiding the back end

database. The login form does not return any errors from the server, making it difficult to determine back-end used.

4.4 Missing function level access control

As part of our analysis, we verified that Piazza conducts server-side authentication and authorization checks for all requests. This included changing the front-end code to allow options that are usually hidden, or sending in POST requests for actions which are not allowed directly from the interface. We edited the javascript file before it was loaded; changes included setting the method "hasPermission" to always return "True" and replacing all checks for "if user.isInst" to True. This allowed students to click buttons normally only available to instructors can see, such as one to endorse an answer, or one to link to a page that edits class settings and information. However, none of these changes allowed us to break any security policies. If a student "endorses an answer," in this context, it simply shows up as "thanks" when other users view it in the future. Clicking to edit class settings brings up the page it links to, but it redirects to the Q&A start page.

We attempted a similar attack to bypass access controls for user registration. When a new user signs up for Piazza, he or she must provide a valid email address associated with the university network they are joining. To verify that a new user does indeed own this email, Piazza sends a verification token to the email address, which the user must then input before actual registration can occur. We tried to bypass the check on the email's domain to try to sign up using, for example, a gmail address. Piazza checks this on the server, however, so this could not be done. Then, we tried to entered a fake email address with the required domain. We noticed that the front end does a check to see if the token matches, and if so, it redirects to the user's registration page. We changed this check to always return "True." However, the redirect URL contains the token input by the user, and is only a valid page if the token is correct, so we were unable to hack the registration process.

Furthermore, we recorded the POST requests used in privileged actions, such as viewing a private post, changing class settings, Many of the actions resulted in the same POST request for both students and teachers. For example, both the instructor's "good answer" button and the student's "thanks" button results in a request made with parameters "method": "content.addfeedback",

"tag":"endorse". On the server side, the appropriate action is executed depending on the user who sent the request, explaining why our attack through the front end did not succeed. Other actions we attempted to break included viewing poll results that should have been hidden, revoting in a poll that only allowed voting once, viewing private posts, changing class settings, and posting anonymously in a class with disabled anonymous posting, We verified that all of these actions are securely validated on the server side, with the exception of one issue with the Careers site that will be elaborated in section 4.7.

4.5 Session management

A preliminary analysis of session management revealed that Piazza has no limit on the number of times someone can unsuccessfully attempt to login to an account. This gives a malicious user the potential to brute force a victim's password.

Piazza manages its cookies relatively well. It sends cookies with the "HttpOnly" flag set to specify that only the server can access the cookies, mitigating the risk of a client-side script accessing the cookies. It does not set the "secure" flag, which would ensure that the cookies are never sent over unencrypted HTTP connections. However, this quite unlikely to cause problems, since all traffic to Piazza occurs through HTTPS anyways. [5]

One more subtle issue we discovered is that Piazza allows cookies to be reused, implying that the server does not destroy old cookies. We were able to save the cookies from our sessions, and create new POST requests using those cookies, even after we had logged out. In fact, even after logging in again and getting new cookies for the same user, we were still able to continue using the original cookies. This means that as long as an adversary gets access to cookies used at some point in the past by a victim, he or she can continue to use those cookies in the future.

4.6 Clickjacking

Through a clickjacking attack, a malicious website can a user into clicking a UI element that the user might think does something else. [6] To test Piazza's vulnerability to clickjacking, we created a fake website ((see HTML

code below) with a transparent iframe containing a Piazza page. Beneath that iframe, we placed some of our own content, including a link that was positioned directly under a "good answer" button. If an instructor visits this malicious site while logged into Piazza and clicks on the link, he or she will unwittingly submit the POST request to endorse the answer. This type of attack can be used to trick users into having many types of unexpected interactions with Piazza, and forms an important security issue that should be addressed.

Clickjacking HTML

```
<HTML>
    <body>
        <div style="position:relative">
            <iframe
   src="https://piazza.com/class/in880gq7ni4107?cid=12"
   style="opacity:0" width="90%" height="80%"></iframe>
            <a href="http://www.google.com" target="_blank"
   style="position:absolute;top:71%;left:32%;z-index:-1">CLICK
   ME!</a>
    </div>
    </body>
</HTML>
```

4.7 Sensitive Data Exposure

Because Piazza provides a recruiting platform, many students post contact information, work history, and other potentially sensitive data in their profiles. Attackers may, therefore, try to gain access to this data for anything from spam advertising to identity theft. Users have the option of changing their personal settings and making their profiles and/or resumes private from other students, but the system in place is insecure. If a student sets his profile to private, other users cannot discover his profile by searching his name. However, the server does not defend against POST requests for the profile. As long as an attacker knows the student's user id, he can very easily send a POST request and retrieve all profile information (2). Although user id's are not immediately evident, there are a number of places in which they are exposed in Piazza. Namely, each class page of the Q&A forum displays the number of currently active users. A quick look at the code returns the user

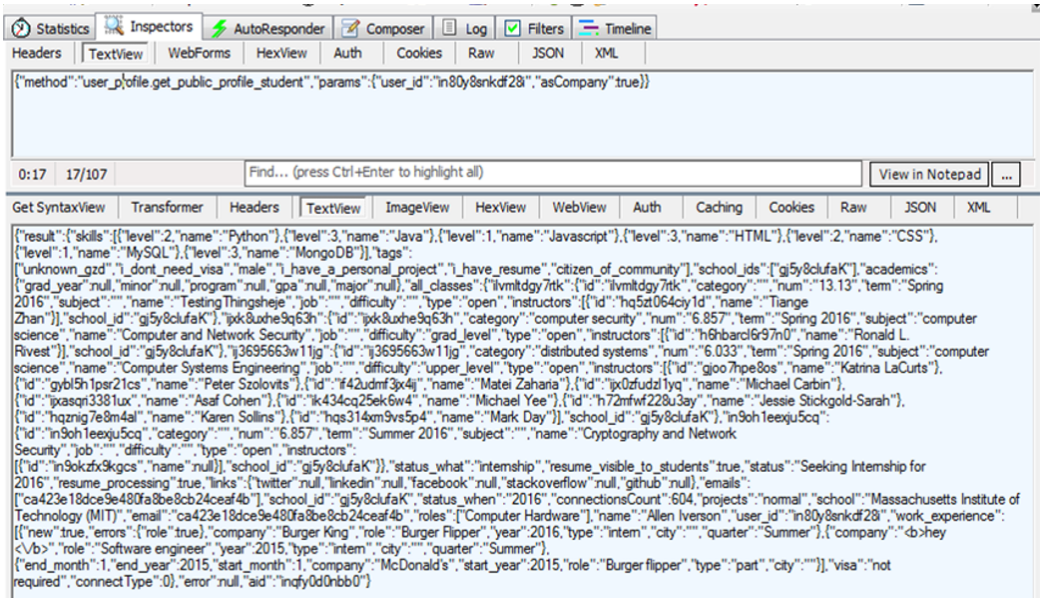

Figure 2: Response to a POST request for a student's profile information. The request was made using the Fiddler Web Debugger tool.

id's of these people, which can then be used to request information from the server. Although Piazza performs server-side permission checks for actions taken in the Q&A portion, here there is no enforcement of confidentiality at the server. This is a severe breach of privacy for users of this system.

Another privacy setting choice that students have concerns the visibility of their resumes. Even with a public profile, users may opt to hide their uploaded resumes from the view of other students. Closer examination of the code reveals a similar issue as with private profiles – information is not truly secured, only hidden from view in the interface. In the HTML code itself, a frame is created for the resume with all the appropriate attributes, except that it is "hidden." That is, it is not displayed when the page is rendered, but it is readily accessible, and anyone can find the URL to the student's resume on Piazza's cloud server (3) This is especially dangerous because many users state detailed contact and even residential information on their resumes. Thus, the lack of privacy is a blatant violation of Piazza's security policy and an easily exploited vulnerability.

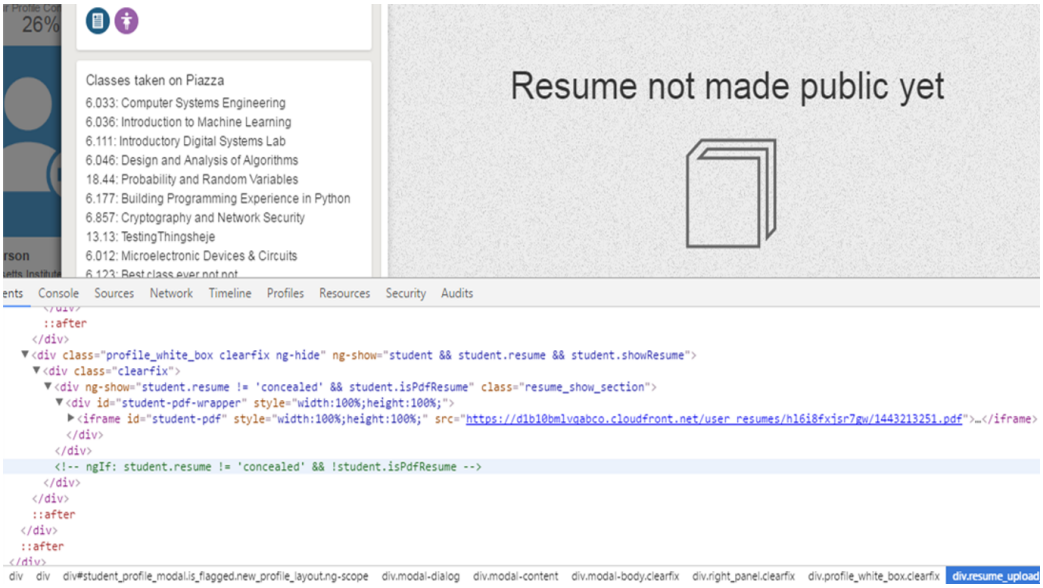

Figure 3: The link to a student's resume can easily be accessed, even if the student specified it to be hidden.

Consequently, confidentiality of sensitive information on Piazza is easily compromised, and an attacker may gain copious amounts of otherwise private data on users of the site.

4.8 Insecure File Upload

We found that the resume uploading service on the Careers site did not check for the extension of files uploaded, so users could upload malicious executables in the resume box. Normally Piazza Careers displays the resume in a pdf viewer, but if the file is not a pdf, users will have the option of downloading the file. Thus, an unsuspecting user could end up downloading a malicious file from what they believe to be a student's resume.

5 Recommendations

In light of the security issues we have found, we recommend Piazza take the following actions to improve their user security.

5.1 Improving Security Definition

We recommend that Piazza enhance its user validation methods to better comply with its security policies. Many instructors are unaware that there is an option for instructor self-enrolling and do not turn off this feature. It would be useful to clarify this feature upon registering a class. The lack of authentication beyond possessing an email address from the school's domain also means that any student or other personnel can pose as the instructor for an actual class on Piazza. We suggest requiring some additional identity check for users claiming to be the instructor of a course. Alternatively, Piazza could disable instructor self-enrollment by default, and require an existing instructor to verify new ones.

We suggest that Piazza alert users when their posts have been edited and require the original poster to approve edits. This is especially crucial for instructor-endorsed posts, as instructors should approve of edits to the answers they have endorsed.

5.2 Checking HTTP Requests on Server

In the Q&A forum portion of Piazza, server-side checking of HTTP requests has already been implemented, but the Careers site still contains some vulnerabilities. Namely, the server responds to all user profile requests, regardless of the user's profile privacy settings. Thus, a similar check of function level access should be made on the server for user profile POST requests as for user registration or answer endorsement in classes.

5.3 Hiding Resume Links in HTML

Instead of generating a frame and loading the resume link in the HTML code every time a user's profile is opened, Piazza should instead retrieve the information from the server if and only if it has already verified that the user has not set viewing of his resume to be private. This would actually enforce the privacy of potentially sensitive user information.

5.4 Checking Upload File Types

As mentioned in Section 6.4, Piazza does not check the file type when users upload resumes. This is a severe vulnerability, as users may upload executables or other forms of malware for others to download. Thus, we recommend that only PDFs, Microsoft Word documents, or other text-based files be accepted by the site.

5.5 Limiting Login Attempts

After a certain login rate limit has been surpassed, Piazza should implement a timeout during which no more login attempts can be made. They could also implement a CAPTCHA so that passwords cannot be brute forced with high computational power. Login rates could be limited per IP address or per IP block to prevent attackers with access to a large cluster of machines, such as a botnet. Alternatively, they could be limited per account username [3].

5.6 Destroying Previous Cookies

When a user logs out, Piazza should make sure the cookies are destroyed both in the browser and on the server. This will prevent adversaries who find cookies from previous sessions from using them to make requests impersonating the cookies' original owners.

5.7 Setting the X-Frame-Options Header

In order to prevent the possibility of clickjacking, Piazza can simply add an appropriate X-Frame-Options header to any HTML. Setting this header to "deny" will ensure the content cannot be framed, while specifying a domain for the "allow-from" will ensure that only trusted websites can frame the content.

5.8 Sanitizing HTML Inputs

In all text boxes that allow HTML tags as part of the input, Piazza should ensure that inputs are properly sanitized. Currently, Piazza does well defending against cross-site scripting attacks in this regard. It only needs to implement the same checking for the Careers profile that it does for Q&A forum posts, taking care that invalid tags such as scripts are either stripped from the input, or properly escaped, before they are stored in the server.

6 Conclusion

After a thorough investigation of the Piazza website as both students and as instructors, we found the Q&A forum aspect to be mostly secure, whereas the Careers portion contained many vulnerabilities. Some of the most significant issues include clickjacking, exposing sensitive information, and allowing insecure file upload. With the above recommendations, Piazza should be able to better enforce its security policy and provide a safer platform for all users.

However, there remain some aspects we have not covered in this analysis. We were unable to explore the website as the third principal–a company. Because it composes the other half of the recruiting platform, this is also an essential role to secure and authenticate. Furthermore, Piazza offers a mobile application as well, which has not been investigated. With the increasingly large user base for mobile devices, it is equally important to enforce usage and information security in the application as on the traditional website. Thus, future analyses of Piazza's security should focus on these two aspects: the company perspective and the mobile platform.

Permission and Acknowledgements

We would like to thank John Knight and Rachel Lam from the Piazza team for granting us permission to do this project. We would also like to thank the 6.857 course staff, especially Professor Ron Rivest and Connor Fromknecht for their feedback and advice on this project.

References

[1] OWASP. "Top 10 2013-Top 10." *OWASP.org*. OWASP, 2013. Web. <https://www.owasp.org/index.php/Top_10_2013-Top_10>.

[2] Damele, Bernardo, and Miroslav Stampar. "SqlmapÂő." *Sqlmap*. N.p., 2006. Web. <http://sqlmap.org/>.

[3] Timoh. "Rate-limiting Web Application Login Attempts." *Rate-limiting Web Application Login Attempts*. N.p., n.d. Web. <http://timoh6.github.io/2015/05/07/Rate-limiting-web-application-login-attempts.html>.

[4] Protalinski, Emil. "Cross-site Scripting Attacks Up 69%." *The Next Web RSS.* 2012. Web.

[5] Hunt, Troy. "C Is for Cookie, H Is for Hacker âĂŞ Understanding HTTP Only and Secure Cookies." *Troyhunt.com.* Troy Hunt, 26 Mar. 2013. Web. <https://Fwww.troyhunt.com//c-is-for-cookie-h-is-for-hacker>.

[6] Kantor, Ilya. "JavaScript Tutorial." The Clickjacking Attack, X-Frame-Options. Ilya Kantor, 2004. Web. <http://javascript.info/tutorial/clickjacking>.

[7] Zeller, William, and Edward W. Felten. "Cross-site request forgeries: Exploitation and prevention." *The New York Times* (2008): 1-13. APA