Security Analysis of Rallyteam

December 11, 2016

Emmett Tan (37087129), Harris Lin (32007122), Irene Chen (51068096), Kaibo Ma (32400129)

Group #2

Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada

UBCEmmettTan@gmail.com
karu.irene@gmail.com
kaiboma@gmail.com
harrislin94@gmail.com

***Abstract*—We are currently analyzing the security of a web application owned by Rallyteam. As Rallyteam's web application contains confidential information, its security is important in protecting Rallyteam's clients and stakeholders. To carry out a systematic evaluation of Rallyteam's vulnerabilities, we will follow OWASP penetration testing methodologies. We will then present our findings to Rallyteam and make recommendations that can mitigate discovered vulnerabilities.**

I. Introduction

Rallyteam is a startup that has created a web platform for users to simplify professional networking, employee development, as well as workforce and community management. Rallyteam emphasizes connecting people with varied skillsets and providing a user-friendly and professional interface for them to collaborate on projects. At first glance, the website appears polished. Rallyteam's developers have also conducted security analyses themselves to uncover security issues. However, as a startup, oftentimes shortcuts are taken in the code to develop essential features at a fast pace. Thus, some security measures may have been overlooked or ignored. To protect Rallyteam's clients' sensitive information and to protect Rallyteam from legal risks, security flaws should be rectified. This report will reveal some of Rallyteam's security holes and provide recommendations with respect to our findings.

Our team follows the OWASP testing methodology to carry out a systematic analysis of Rallyteam's web application. We began by investigating Rallyteam's logic and flow using a proxy that lies between the web browser and the actual server. With this setup, we were able to learn more about the HTTP requests and responses, the logic behind the web application, and access points / gates such as headers, parameters, and cookies. Also in this initial phase, we crafted a variety of queries to Rallyteam's server. Doing so allowed us to observe whether the client can find any information that only the server should be able to see or allow the client to make changes to the web application that he or she is should not have access to.
In our second phase, we will actively search and gather information to expose the flaws that we have discovered during our initial inspection of Rallyteam. We will perform testing in specific categories. These include authentication, authorization, session management, input validation, error handling, cryptography, and client-side testing. Through this security analysis, Rallyteam will be able gain perspective on the security holes in its application and consider mitigations to these vulnerabilities.

II. Analyzed System

Rallyteam's target customers are companies and their employees. They offer a cloud web platform (SaaS) for the employees to connect and create groups and projects. Each company is given its own Rallyteam domain. For example, our penetration testing Rallyteam domain is http://pentest-cpen442.rallyteam.com and our CPEN 442 Rallyteam domain is http://cpen442-ubc.rallyteam.com. Within each domain, employees can sign up or be invited for personal accounts. In other words, domains do not share Rallyteam accounts, and the same person would need separate accounts on separate domains. Knowing this, we can infer that the two domains are separate in the databases. A data leak within one domain would not affect the other, although the vulnerability that exposed the data leak would also be present in other domains.

On Rallyteam's platform, users can share their interests, skills, and other information on their public profiles. Other users are able to see what skills and interests another employee has. They can create groups, projects and events which allow others with similar interests and skills to connect and share ideas on Rallyteam's platform.

On the technical side, Rallyteam is built on top of Microsoft's Windows Azure system. It uses Bootstrap and

AngularJS on the frontend to provide the users with an aesthetic and smooth interface. On the backend, Rallyteam uses Web API. Since Web API is written in C#, which is a safe language, buffer overflow attacks will be ineffective. Service endpoints include opportunities, tags, users, groups, events, count, files, and track. Information sent from the client side to the server side is through the HTTP requests and responses. From our first phase of learning about Rallyteam's logic, we found that some requests are encrypted, but responses are generally not encrypted.

On the backend, data are stored in Microsoft SQL and Azure blobs. It is worth noting that SQL injection attacks do not work on Microsoft SQL. Strings are sanitized before being saved into the database. For example, one cannot store raw hyperlink strings. Rather, the web application will change it to another format. For example, if the input string is "https://www.youtube.com/watch?v=SkXxKnboE0M", then this string will be sanitized and parsed to become something like "SkXxKnboE0M" when it is stored into the database.

Azure's Web role is responsible for servicing HTTP requests. Google analytics is used for front-end telemetry, while Visual Studio App Insights is used for backend telemetry.

On the application layer, HTTPS is used. SendGrid is used to manage and send emails on Rallyteam's platform. Users are able to create an account using an email or sign up through a Google or Microsoft Outlook account. In-house accounts are authenticated by Rallyteam, whereas Google and Outlook accounts are managed through auth0.

III. Related Work

Rallyteam has previously undergone penetration tests according to our advisor, Ildar Muslukhov. Muslukhov has stated that in the previous test, many vulnerabilities were discovered throughout the system. One of the major flaws was the site's vulnerability to a cross-site request. Not all of the known security flaws in the system have been fixed.

Similar companies have been victim to rogue penetration testing. For example, Slack, another web application that creates domains for groups to collaborate ideas and knowledge had its security compromised in February 2014. Usernames, email addresses, registration details, and passwords worth up to $2 billion USD were stolen from a database due to an unauthorized access to a Slack database storing user profile information. These passwords and user data were salted and hashed using bcrypt. However due to the extent of this leak, the stolen data can be decrypted. After investigation into the breach, Slack implemented two-factor authentication and password kill switches to prevent attacks like this in the future. We may want to consider Slack's security policies in evaluating Rallyteam's security. It is important to be diligent with penetration testing. All it takes is for one company to leak out user passwords. These passwords can then be used to match other accounts from different companies. For example, stolen user information and passwords can be used to hack into user accounts across other sites such as Netflix, Google and Amazon. Users that reuse user identification and password combos are most at risk. This is why performing a penetration test and analysis of Rallyteam is important. All companies should always take the butterfly effect into consideration and look at the possible harm that could propagate from one small data leak.

IV. Analysis Methodology

System Analysis

Our main tools for evaluating Rallyteam's security are browser debuggers and proxies. Through browser debuggers like Mozilla Firefox and Google Chrome Developer Tools, we are able to view Rallyteam's client side code. Furthermore, we can send HTTP requests directly from the browser debuggers. By sending some legitimate requests, we familiarized ourselves with how the requests are constructed. Then, with proxies, we proceeded to make requests in the same format and inject modifications in order to execute actions that are outside of our permissions.

By familiarizing ourselves with the client side code and constructing properly formed HTTP requests, we have discovered that although there seems to be plenty of client-side validation, the same is not true for the server side. For example, when creating groups, the text field for group name is limited to 200 characters, but by sending a POST request to the groups API directly (not through the web interface), we can create a group whose name exceeds 200 characters.

By intercepting and changing the parameters of intercepted requests, we have also discovered that we can change the fields of client requests successfully. Again going back to the groups API, we found that we can change the privacy setting, moderator, name of the group, among many other fields..

Ethical Considerations

As aforementioned, Rallyteam has provided us with a domain for our security analysis. Thus, we are able to conduct attacks on the web application without harmful effects on Rallyteam's existing clients. Furthermore, we will keep Rallyteam's security flaws confidential so that attackers will not be able to exploit flaws due to our analysis.

Risk Management

Our security evaluation of Rallyteam has been explicitly authorized by Rallyteam's Chief Technology Officer. Furthermore, we have Rallyteam's full support in pursuing this security analysis, as demonstrated by their willingness in providing us with a domain intended for penetration testing. As Rallyteam is still in its initial stages of development, there is not a large risk yet posed to stakeholders, as the client base is still small.

V. Failed Attacks

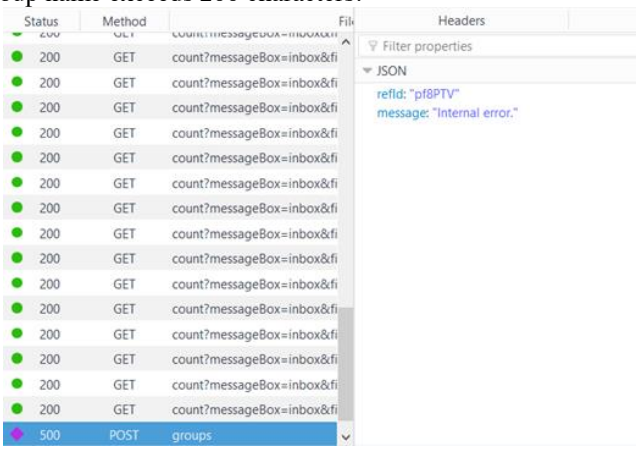
Intercepting and Modifying HTTP Requests and Responses

We attempted to sniff the HTTP requests and responses to Rallyteam's server. However, Rallyteam's HTTP requests and responses appear to be secure, as it employs Secure Sockets Layer (SSL) and HTTPS for networking. As a result, even

though confidential information such as email addresses and passwords are sent in plaintext, the information should be safe from a snooping third-party. Such an attack should only be plausible if an attacker creates a fake certificate and public key, and the human user to chooses to ignore the browser warning regarding the untrusted certificate.

Bypassing the Client-side Data Validation

We attempted to bypass certain data validation rules enforced by Rallyteam's web application client by sending HTTP requests directly to the server. However, we found that Rallyteam employs server-side validation in addition to client-side validation. For example, group names are limited to 200 characters, and the web client does not allow the user to submit the form should the user input for group name exceed 200 characters. When bypassing the web client and sending the request directly to the server, an exception is thrown if the group name exceeds 200 characters.

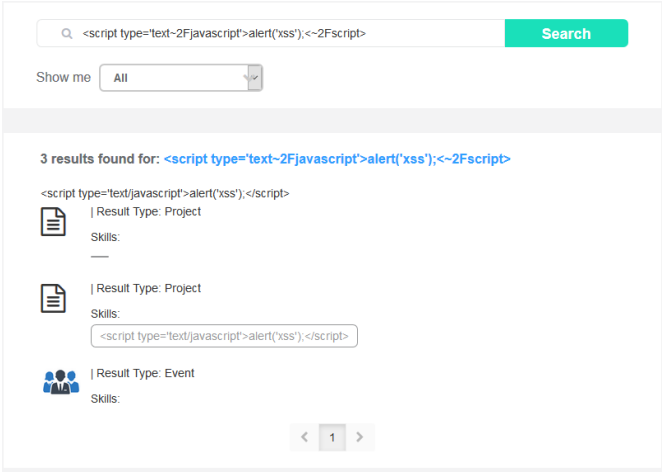


Server response from receiving a POST request to the groups endpoint with a group name longer than 200 characters

Cross-site Scripting

Rallyteam appears to be relatively safe to cross-site scripting attacks. We have attempted to inject JavaScript and HTML through functionalities such as comments, chat, discussion, to no avail.

We tried several types of strings to see if we could bypass Rallyteam's string sanitization. The first type of string that we tried was "<script type='text/javascript'>alert('xss');</script>". Rallyteam defends against this attack by replacing forward slashes with "~2F".

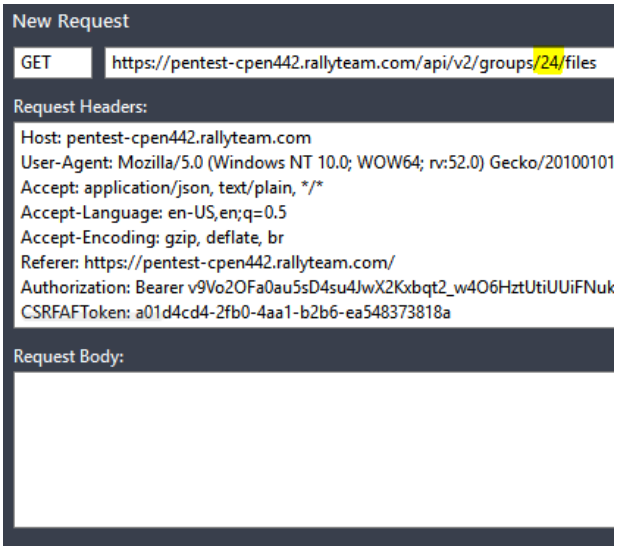


Example of sanitized user input.

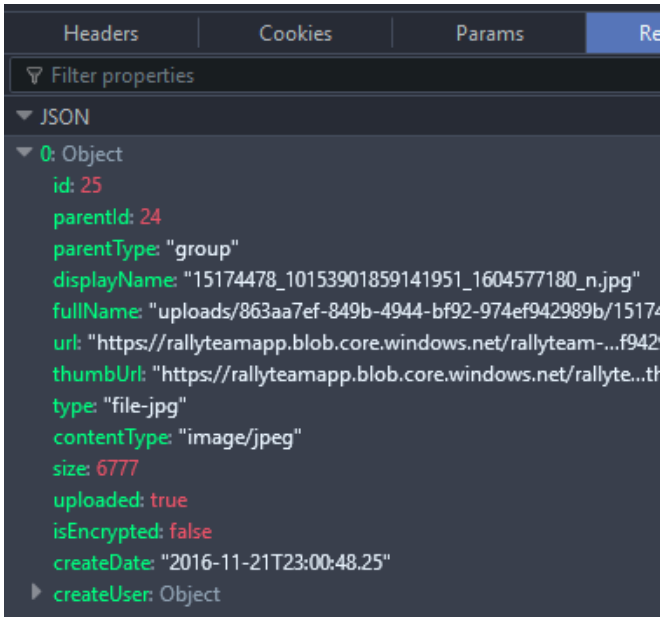
VI. Vulnerabilities Discovered

Disclosure of Confidential Information

Unauthorized users are able to view the projects, events, and files of private groups by sending a properly formed request with the ID of the private group. Unauthorized users may even download the files of private groups with the corresponding blob URL.



Here, we are editing the group ID to a group that we do not have access to



Retrieving the content URL from the response

Users can also view files associated to a group even when the group is deleted. For example, consider a legitimate user who creates a private group for himself for temporary storage of personal information. He stores sensitive information that should only be viewable to himself and should be destroyed once the purpose is over. He then deletes the group after it has fulfilled its purpose. Files associated with the group should be deleted along with the group. However, another user is able to guess the ID of the deleted group, then using the previous method, that other user can access files associated to that deleted private group.

VII. Discussion of the Results

Adversary Models

The objective of the adversary who exploits Rallyteam's disclosure of confidential information is to view information that the user does not have access to. The adversary's initial capabilities include the ability to send HTTP requests as an external user. They have access to API Endpoints and can edit and resend requests simply on a browser develper tools. ID's for group and projects are numbered in order of creation i.e. first group made is ID 1 and second group made is id 2 and so on. The attacker may find the ID's of private projects or groups by simply by guessing numbers in order or looking at the public groups that she is authorized for then deducing the ID's of the private groups. For example, if the attacker can see groups 1 through 10 as well as 12, she would know that group 11 was either deleted or private. The adversary's capabilities during the attack include the ability to obtain private files and information.

Violations of the Principles of Designing Secure Systems

Complete Mediation:

According to the principle of complete mediation, "Every access to every object must be checked for authority." For RallyTeam, it means that it should constantly check authorization of a user at all steps. No validation is done when the api GET requests for project and group files are sent to the server. External users are able to access files from projects that they do not have access to. This violates the principle of complete mediation.

Defense in Depth:

Rallyteam's lack of authorization check for the file endpoint demonstrates that they assume users will not directly send requests to the file endpoint for groups and projects in which they are not authorized because no graphical user interface exists for them to do so. Thus, no authorization check was implemented for access to the files endpoint, since it is implicitly assumed that authorization was checked by the projects and groups endpoints.

Psychological Acceptability:

The user interface for inviting users has client-side validation to check for invalid or malicious input. However, it is not clear whether users should separate the emails by punctuation, spaces, or something else, which can lead to much frustration when users try to send invites. Rallyteam should not make the validation process harder for the user to use the web application.

VIII. Recommendations

We recommend that Rallyteam always check the authorization of the user on the server-side to avoid disclosing confidential information. Their API endpoint to retrieve file data should be consistent with checking for authorization. Rallyteam should assume that attackers can send queries directly to the server, and should not assume that they need only to check for authorization when there is a graphical user interface that allows users to make certain types of requests. Furthermore, we recommend that Rallyteam check for authorization as one of the first steps upon receiving a request. Thus, this authorization check should be done in its Web API controllers. More specifically, the the specific classes that inherit from the ApiController class should call the functions that execute authorization checks.

We also recommend Rallyteam make the user-interface more user-friendly, especially in cases of input validation. For example, for the user interface where administrators are inviting users by email, rather than refusing to submit the form if the administrator does not separate emails with commas, allow administrators to separate emails with spaces too. Another option is to add instructions in the invitation user interface stating that commas are required to separate the

emails or even provide a screenshot example on the correct way.

IX. Conclusion

Rallyteam has a number of security measures currently in place. These measures include include SSL, HTTPS, input sanitization, and server-side validation, among others. However, we have discovered that Rallyteam is not thorough in checking the access rights of client, leading to data leaks and violating the principles of complete mediation and defense in depth. Furthermore, having a secure system is more than implementing security protocols and algorithms. Rallyteam should also consider the psychological acceptability of its security features. In light of these principles of designing secure systems, we recommend that (1) Rallyteam complete a thorough check of all its API endpoints to ensure that all requests are checked for access permissions, and (2) Rallyteam improve the user interface for accepting user input.

Appendix A. Project Code of Conduct

Throughout the project, our team has ensured that we keep the information regarding Rallyteam's security flaws and technology infrastructure confidential. In anticipation of the potential negative consequences from our actions, such as our classmates using our findings to exploit Rallyteam, we will use discretion in presenting these flaws to the class. In addition, we have avoided negatively impacting the experience of our team's classmates by conducting penetration testing on a domain separate from our classmates'.

Appendix B. Responsible Disclosure

Our contact from Rallyteam is Ildar Muslukhov (ildar.muslukhov@rallyteam.com; 778-707-1073). We will meet with him on November 28th after our lab in MacLeod 228 to discuss our final findings.

Any vulnerabilities discovered during our security Analysis on Rallyteam will be documented along with the steps necessary to reproduce the vulnerabilities. It will be a continuous integration with our analysis method. We will constantly provide feedback to Rallyteam with all security issues we may come upon. We will then provide an estimate of the potential severity for each security issue. The timeline for disclosure depends on the severity of the vulnerability but may proceed as follows:

Minor Security Flaws

These are security flaws that do not have large repercussions. For example, it does not leak out personal vital user information, but may allow you to view private groups with unauthorized requests. These vulnerabilities will be tracked and eventually revealed to Ildar Muslukhov during our full report.

Major Security Flaws

Any vulnerability that places user information at risk needs to be addressed immediately. It could potentially spread to other services like in the example of Slack where stolen user data was used on other services like Amazon and Netflix to steal more information. Zero-day vulnerabilities can be catastrophic if not fixed. We will immediately disclose this information to Ildar Muslukhov through phone and email.

X. References

[1] Anderson, Ross. Security Engineering -- A Guide to Building Dependable Distributed Systems. John Wiley & Sons, 2008, Second Edition.

[2] Fay, John. Contemporary Security Management. Burlington, MA: Butterworth-Heinemann, 2011. Print.

[3] Grossman, Jeremiah. XSS Attacks : Cross-site Scripting Exploits and Defense. Burlington, Mass: Syngress, 2007. Print.

[4] Mayhew, Bruce, Nanne Baars, and Jason White. "Category:OWASP WebGoat Project." Category:OWASP WebGoat Project - OWASP. OWASP, n.d. Web. 16 Nov. 2016.

[5] Mark Stamp, Information Security : Principles and Practice, Second Edition, Wiley-Interscience, 2011. Print.

[6] Pernul, Günther. Computer Security – ESORICS 2015 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21–25, 2015, Proceedings, Part I. Springer International Publishing, 2015. Web. 11 Dec. 2016.