Why Hackers Love eHealth Applications

Goyal, Rohit; Dragoni, Nicola

Published in:
Proceedings of the 3rd EAI International Conference on IoT Technologies for HealthCare (HealthyIoT), 2016, LNICST

Publication date:
2016

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Goyal, R., & Dragoni, N. (2016). Why Hackers Love eHealth Applications. In M. Uddin Ahmed, S. Begum, & W. Raad (Eds.), Proceedings of the 3rd EAI International Conference on IoT Technologies for HealthCare (HealthyIoT), 2016, LNICST Springer. (Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Why Hackers Love eHealth Applications

Rohit Goyal[1] and Nicola Dragoni[2]

[1] Technical University of Denmark (DTU)
rgoyal.pec@gmail.com,
[2] Technical University of Denmark (DTU), and Örebro University, Sweden
ndra@dtu.dk

Abstract. The tsunami of Internet-of-Things and mobile applications for healthcare is giving hackers an easy way to burrow deeper into our lives as never before. In this paper we argue that this security disaster is mainly due to a lack of consideration by the healthcare IT industry in security and privacy issues. By means of a representative healthcare mobile app, we analyse the main vulnerabilities that eHealth applications should deal with in order to protect user data and related privacy.

1 Introduction

In February 2015, 78.8 million of Anthem[1] customers were hacked. This has been the largest healthcare breach so far, and it opened the floodgates on a landmark year. According to the Office of Civil Rights under Health and Human Services[2], more than 113 million medical records were compromised in 2015. This security disaster was further validated by Gemalto, whose report on data breach worldwide for the first half of 2015 [6] showed that the healthcare industry is taking the lead with 84.4 million total records lost. In this paper we argue that the main reason behind this alarming situation is that security and privacy are not seriously taken into account by the healthcare IT industry yet. Security and privacy are seen only as a patch to be applied in case of discovered information leakage, never as key design features that should be considered from the very first system design phase and throughout all the healthcare IT system life cycle.

Contribution of the Paper. In order to provide a concrete example of our argument, in this paper we perform a security and privacy analysis of one of the most promising healthcare mobile system under development in Sweden, namely RAPP (Development of the Recovery Assessments by Phone Points) [7]. RAPP is a healthcare mobile and Web system for tracking the health of patients post surgery and assist them with the required care. The portal is accessible through a Web and mobile application which records patient's responses to a set of survey questions related to their recovery process. The responses are sent to the server which also provides administrative access to the healthcare organization. The RAPP architecture is very common to all the applications in the healthcare

[1] https://www.anthem.com
[2] https://ocrportal.hhs.gov/ocr/breach/breach_report.jsf

IT domain, which makes RAPP a perfect candidate for our investigation of key security and privacy issues in healthcare mobile applications. However, it's important to stress that we are using RAPP as case study only, while the outcome of the analysis is sufficiently general to be applied to a generic healthcare mobile system, as the number of recent security and privacy breaches worldwide prove.

Methodology and Outline of the Paper. We consider a patient Bob who had surgery and has been relieved from the hospital for recovery at his home. The hospital has provided him access to the RAPP system with credentials containing his username and password. He has been asked to use the Web based interface or the application installed in his smartphone. In Sections 2-6, we look into the security implications of this system and how Bob's privacy could be compromised due to system vulnerabilities. In Section 7 we then sum up, providing basic security/privacy recommendations for developing healthcare IT applications.

2 Username and Password

Attack Scenario. An attacker Charlie gets the information that hospital is using 4 digit usernames and passwords for authentication. He decides to hack into the system by impersonating some users and sending rogue survey responses to the hospital administrator, thus interrupting the usual service and causing unnecessary confusion. His goal could also be a denial of service to the legitimate users who need service from the hospital. To achieve his goal, he uses a simple tool from Internet to try all combinations of 4 digit usernames and passwords.

Technical Description. In RAPP, username and password (both 4-digit numeric strings) are used for authenticating users in the system. Both have a very small search space of depth 10 and length 4 which makes them vulnerable to brute force attacks. An online tool GRC [11] estimated that 11 seconds would be required to break this password (with 1000 guesses per second) if the attacker knows the search space. This design choice is very weak from security standpoint.

Recommendations. It is recommended that systems use more complex passwords with greater length. The security of the passwords is greatly enhanced by use of lower and upper case characters and special symbols. A detailed guide about the password authentication can be found at OWASP Web page [1].

3 User Authentication Workflow

Attack Scenario 1. Bob decides to notify the hospital authorities about his condition after the surgery. When he tries to log in into the system, he realizes that he has forgotten his credentials. There seems to be no way for him to retrieve the forgotten password.

Attack Scenario 2. Bob's computer crashed all of a sudden due to some technical problems. He borrows his neighbor Charlie's laptop to request help from the hospital authorities. He logs in into the system and submits the survey response. He realizes that there is no option to log out from the system. He is not sure if he should return Charlie's laptop with his account logged in.

Technical Description. The systems that use password authentication need to provide a secure mechanism to the user to retrieve passwords via email or phone in case they forget it. This is important to ensure the uninterrupted service to the user. Once the user is logged in, he would want to log out from the system when he is done. The log out functionality ensures that the users can share their systems with other people which is a quite common use case.

Recommendations. The login page should provide a secure procedure for a user to retrieve or change his/her username or password, for instance by email or a security code on the mobile phone. The Web site should have logout functionality which allows users to log out from the system after each session.

4 Network Communication

Attack Scenario. Bob has a wireless Access Point configured in his house with weak or no security. An attacker Charlie who lives in the next house is always sniffing neighbors' wireless network traffic for privacy invasion. Bob opens the RAPP Website, fills the survey response and submits it. Charlie who was listening to this traffic using wireshark or similar tools would be able to see all the communication between Bob and the RAPP server. He could also replay or modify the contents of the request. In this way, the security and privacy of Bob is compromised without his awareness.

Technical Description. The RAPP client-server communication is based on API calls for logging in users and sending survey responses to the server. The following experiment has been conducted to monitor the traffic between the end points: an HTTP proxy software, known as Charles proxy [2], has been installed on the client machine listening to HTTP traffic on port 8070. The Web browser is configured to forward its HTTP traffic to localhost on port 8070. The target domain is requested by entering the URL in the address bar. The network requests to the target domain can be seen in the proxy interface along with header information and responses. It has been found that the client and server communicate over HTTP protocol. This application level protocol is inherently insecure because the communication is not encrypted. The attacker could sniff the traffic and read/modify/replay the contents of the requests and responses. The username and password are transmitted in clear text. This is a major privacy leak. Also, since the request is made on HTTP, the server is not authenticated before starting the communication. This could lead to man in the middle attack [3].

Recommendations. The communication should be protected by means of the HTTPS protocol [10], in order to have a secure channel between client and server.

The major goals of the protocol are to authenticate the server and maintain privacy and integrity of the data exchanged. The server is authenticated by use of trusted Certificate Authority signed certificates providing resilience against man-in-the-middle attacks [5]. The confidentiality and integrity of the communication is achieved by encryption and use of digital signatures. The current standard for certificate keys for Public Key Infrastructure (PKI) is 2048 [12].

5 Local Storage

Attack Scenario 1. Bob is not feeling well after he got back home from surgery. He decides to notify his condition to the hospital authorities. He opens the browser and logs in into the RAPP Web portal. He fills out the survey responses but does not submit it immediately. Meanwhile, his friend Charlie visits him in his house. Charlie who is an evil neighbor opens Bob's computer and accesses his private information from the browser Local Storage. Bob's privacy gets compromised due to the saved survey responses in the browser Web storage.

Attack Scenario 2. RAPP server gets infected by malicious javascript code. When Bob loads the Website, the javascript accesses browser's local Web storage and redirects Bob to the malicious site. Bob's private data and authentication information gets compromised.

Technical Description. The local offline storage in browser (also known as Web storage) is used by Websites to save some data. This data is accessible to the user with local privileges. It has been found that the RAPP server stores the authentication ticket and the survey results in the Web storage. According to OWASP guidelines, it is recommended not to store any sensitive information in the local storage [9], as vulnerable to cross site scripting attacks [4].

Recommendations. The patient survey data can be stored in session storage rather than local storage since this information is not required for persistent use and is relevant only for the current session. As soon as the browser tab is closed, the session would be discarded. The auth token should be saved as a cookie rather than a key value pair in local storage. This because a cookie can be set with an expiry date, invalidating the user after some time. Though a cookie can be set forever until the browser cookies are cleared, it still gives more control to the administrator of the Web site. Moreover, the Web storage can be vulnerable to cross site scripting attack, thus exposing the auth token or session identifier to the attacker via malicious javascript. Also, the cookie can be set with HttpOnly flag which does not allow javascript to access it. Thus, only the target domain can access the cookie information.

6 Information Gathering

Web Server Fingerprinting. The knowledge of Web server, database, frameworks, etc..., used by a Web site gives a big advantage to the attacker to exploit known vulnerabilities in the specific version of the software or take advantage of possible misconfigurations. We did 2 simple experiments to highlight how it is easy to leak information. In the first experiment, we have pinged the RAPP URL `rapp-productiontest.nethouse.solutions`. The ping response exposes the server on which the Web application is hosted. The URL gives information about the company, location of the server and hosting service provider. In the second experiment we have configured a Charles proxy on the browser, then visited the same RAPP URL and monitored the traffic on proxy interface. The response headers for the request expose information about the Web server, Web application framework, its version, and several other similar technical details.

Recommendations. The server response headers could be disabled or obfuscated to prevent the leakage of important information. The exact steps to disable these headers would depend on the server and framework. For example, ASP.NET version header could be disabled by turning the flag "enableVersionHeader" in project's web.config to "false". By making these changes in the configuration, the developer could test the Web site again using the tools and make sure that only the necessary information is sent in the response headers.

Cookie Information. The RAPP admin portal sets the cookie when the hospital admin logs in into the portal. The authentication cookie is stored with cookie name as ".ASPXAUTH". The cookie should not reveal any information related to how it is generated. Otherwise, it makes it easier for the attacker to exploit vulnerability of a specific framework.

Recommendations. The cookie should not reveal any other information except the token. Also, MSDN documentation suggests the usage of "requireSSL" for forms posting cookie information [8]. The cookie is sent by the browser only to a secure page that is requested using an HTTPS URL.

Information Leakage. The Web site should not leak any important information related to database in the authentication step. Indeed, any specific information helps the attacker to deduce more information about the system. In one of the authentication attempts on RAPP patient Website using admin credentials, we receive the error message "Only patients can log in to the app". This message provides more information than required. An attacker can deduce that the credentials might belong to non-patient user roles. The RAPP hospital admin portal also responds with similar error messages.

Recommendations. All kinds of errors regarding log in failure should convey only the generic information e.g. "Username or password is incorrect".

7 Conclusion

The tsunami of Internet of Things and mobile applications for healthcare is not currently supported by adequate privacy and security measures, as shown by the increasing number of health data breaches in the last few years. This paper calls the healthcare IT industry to take security and privacy into serious consideration and to raise the stakes against well known security and privacy vulnerabilities. To show a concrete example of what kind of vulnerabilities an healthcare system should deal with, we have done a security and privacy analysis of the RAPP mobile healthcare system. As key outcome of this analysis, there are various basic principles and recommendations that should be taken into consideration while developing healthcare applications. We summarise these findings below.

- The user ids and passwords should follow the technical standards in terms of length, inclusion of special characters etc.
- The user log in and out should be implemented according to the use cases.
- The network communication should take place over HTTPS only.
- The authentication token should be stored safely in the browser.
- The session related information should be stored in the browser session storage.
- The Web server should not leak unnecessary information in the headers or special requests.

By considering these basic security issues from the very beginning of the application design and implementation, the final system will result significantly more secure and trustworthy, reducing the impressive numbers of successful attacks that have recently involved the healthcare industry.

References

1. Authentication Cheat Sheet, OWASP, `https://www.owasp.org/index.php/Authentication_Cheat_Sheet` [May 2015].
2. Charles proxy, `https://www.charlesproxy.com/` [May 2015].
3. M. Conti, N. Dragoni, V. Lesyk. A Survey of Man In The Middle Attacks. IEEE Communications Surveys & Tutorials, doi: 10.1109/COMST.2016.2548426, 2016.
4. Cross-site Scripting (XSS), `https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)` [May 2015].
5. Description of the Server Authentication Process During the SSL Handshake, `https://support.microsoft.com/en-us/kb/257587` [May 2015].
6. Gemalto, First Half Review 2015, `http://www.gemalto.com/brochures-site/download-site/Documents/Gemalto_H1_2015_BLI_Report.pdf` [May 2015].
7. M. Jaensson et al. The Development of the Recovery Assessments by Phone Points (RAPP): A Mobile Phone App for Postoperative Recovery Monitoring and Assessment. JMIR mHealth uHealth 2015;3(3):e86.
8. How To: Protect Forms Authentication in ASP.NET 2.0, `https://msdn.microsoft.com/en-us/library/ff648341.aspx` [May 2015].
9. HTML5 Security Cheat Sheet, `https://www.owasp.org/index.php/HTML5_Security_Cheat_Sheet` [May 2015].
10. HTTPS protocol, `https://en.wikipedia.org/wiki/HTTPS` [May 2015].
11. Password strength, `https://www.grc.com/haystack.htm` [May 2015].
12. PKI (Public Key Infrastructure), `http://searchsecurity.techtarget.com/definition/PKI` [May 2015].