Heartland Payment System Breach Case Study

Bastian Tenbergen

April 2021

Heartland Payment System Breach

Background

The Heartland Payment System is a transaction management system for money transfers and credit card payments, handling over 100 million transactions per month. In 2008, Heartland belonged to one of the largest payment providers in the world and was very confident about their transaction management system, going so far as to provide its users with a "breach warranty" against data theft. However, between March 2008 and mid-2009, several million transactions, users accounts, and credit card information were compromised resulting in approximately $100 million in debit and credit card payments to be stolen by attackers.

Case Study Overview

The Heartland Payment System theft constitutes one of the largest financial transaction system breaches of its time. In two, possibly different attacks, hackers managed to compromise not only the transaction system through a carefully crafted SQL injection, but also by physically removing computer hardware with stored employee passwords from corporate offices. Specifically, the SQL injection attack may have been the culprit that allowed attackers to redirect large amounts of transactions to accounts controlled by them, thereby perpetrating the theft.

Student Instructions

This case study primarily deals with technology, vector of attack, and prevention of SQL attacks. Complete the following tasks:

1. Describe what constitutes an SQL injection attack. Be sure to explain the required components and basic structure of an architecture that allows SQL injection attacks to take place.
2. Implement a simple architecture allowing SQL injection attacks in a simple system. Demonstrate the attack vector by attacking your system. Record your procedure either textually (by writing down the sequence of steps) or by capturing a short video.
3. Describe suitable counter measures. You may describe them either by example of your system from Part 2) of this case study, or by describing them more generally. For example, you could discuss frameworks and how they work.
4. Implement one (or more) of the countermeasures in your system from Part 2). Repeat the steps you took in Part 2) to attack your system and demonstrate that your system is now sufficiently hardened.

Instructor notes

This case study is best assigned as an individual exercise, either in synchronous or asynchronous educational settings. The implementation parts may be left out, in which case the case study might be useful for introductory-type exercises. Depending on the requested depth of the implementation component, it may also be used as a project with a partner, or as a microproject to hone cybersecurity tricks.

A sensible variation might be to split up the case study parts among students, where one student implements the system, another student executes the attack, and a third student improves on the implementation, with a fourth student retrying the attack. The team of four students could be asked to round-robin their implementations and draft a joint security assessment report.

Example solution

1. Describe what constitutes an SQL injection attack.
    a. Relatively simple attack typically launched against web-facing systems that are backed by a SQL-type database (e.g., MySQL, PostgreSQL)
    b. Like all attacks, involves three phases:
        i. Research: insert SQL statements that either evaluate to an error or that evaluate to true to gather data on type of database
        ii. Attack: insert appropriate SQL queries to either log in as certain user on the system or to impair DB functionality
        iii. Exploit: take control of captured user account or impair database, e.g., by dropping tables.

2. Implement a simple architecture allowing SQL injection attacks in a simple system.
    a. The simple architecture involves some HTML form for user login and a database that is being POSTed to.
    b. Inserting “ `'OR 1 = 1; DROP TABLE users;'`” may result in the table “usres” be removed from the database.
3. Describe suitable counter measures.
    a. Properly validate user inputs at the frontend (i.e., client-side), but also in a middleware component that shields the DB backend from direct user inputs. Filtering out SQL commands through a redlist and filtering out special characters such as quotation marks prevent injection attacks.
    b. Using a noSQL-type database (e.g., MongoDB, CouchDB) does not require SQL queries and communicates between front- and backend through a RESTful API. API endpoints allow for easy integration of authentication tokens, thereby making other, noSQL injections more difficult (but not necessarily impossible).
4. Implement one (or more) of the countermeasures in your system from Part 2).
    a. The simplest implementation would be to replace quotation mark characters with white spaces in user input fields.

References

Lewis, D.: “Heartland Payment Systems Suffers Data Breach,” online resource available at: https://www.forbes.com/sites/davelewis/2015/05/31/heartland-payment-systems-suffers-data-breach/?sh=5e92aac744ad, 2015. Accessed 27 Mar 2021.

McGlasson, L.: “Heartland Payment Systems, Forcht Bank Discover Data Breaches,” online resource available at: https://www.bankinfosecurity.com/heartland-payment-systems-forcht-bank-discover-data-breaches-a-1168, 2009. Accessed 27 Mar 2021.

Jackson Higgins, K.: “Russian Hackers Sentences in Heartland Payment Systems Breach Case,” online resource available at: https://www.darkreading.com/attacks-breaches/russian-hackers-sentenced-in-heartland-payment-systems-breach-case/d/d-id/1331080, 2018. Accessed 27 Mar 2021.

Lukic, D.: “Hearthland Paymen Systems Breach, What Lessons to be Learned,” online resource available at: https://www.idstrong.com/sentinel/heartland-data-breach/, 2020. Accessed 27 Mar 2021.

Kerbs, B.: “Hacker Ring Stole 160 Million Credit Cards,” online resource available at: https://krebsonsecurity.com/2013/07/hacker-ring-stole-160-million-credit-cards/, 2013. Accessed 27 Mar 2021.

Korolov, M.: “8 PCI DSS questions every CISO should be able to answer,” online resource available at: https://www.csoonline.com/article/3529475/8-pci-dss-questions-every-ciso-should-be-able-to-answer.html, 2020. Accessed 27 Mar 2021.

Porup, J. M.: “What is SQL injection? How these attacks work and how to prevent them,” online resource available at: https://www.csoonline.com/article/3257429/what-is-sql-injection-how-these-attacks-work-and-how-to-prevent-them.html, 2018. Accessed 27 Mar 2021.

Oftedal, E.: “noSQL-ibjection,” online resource available at: https://erlend.oftedal.no/blog/static-110.html, 2010. Accessed 27 Mar 2021.