Postmortem: Online web based service outage

Postmortem Summary

The web based application failed to load. It return message "*Error: SQLSTATE[HY000] [2002] No such file or directory*". These happened on *22:00UTC 1-OCT-21* and stayed till *04:00UTC 2-OCT-21* finally the issue has been resolved. It impacted probably all of our precious customers and company staffs alike. It had been impossible to access the services for the car renting business since the website was having difficulty connecting to a database and running basic services. After getting hint about application database being compromised vulnerability assessment was conducted. It was found that there was a loop hole at user inputs validation. One input was not being sanitized and validated to make matters worse It is being used for the SQL queries internally by the application. This was suspicious enough to curse SQL injection attacks.

Events Timeline:

The web service offline issue unfolded as follows:-

- *21:55UTC 1-Oct-21*: the website not responding issue was first reported by a customer call around.
- *22:00UTC 1-OCT-21:* an engineer noticed a monitor alert *"website not returning HTTP 200"*
- *22:15UTC 1-OCT-21:* on call engineer immediately commenced debugging to find issues
- *22:30UTC 1-OCT-21:* engineer connected to server checked free space , daemon running and responsiveness.
- *23:00UTC 1-OCT-21: engineer* observed that the database was not found or may be missing
- *22:30UTC 1-OCT-21:* the *database admin* was called upon to recover data and restore databases
- *00:15UTC 2-OCT-21:* issue has been escalated to our *security engineers* and they begun vulnerability assessment
- *02:00UTC 2-OCT-21:* the database was successfully reconstructed
- *03:15UTC 2-OCT-21: site engineer* performed testing and integrity checks
- *03:30UTC 2-OCT-21:* performed patch per recommendation for user authentication input fields
- *04:00UTC 2-OCT-21:* performed final unittest checks and website is up and running

Root Cause and corrective action

The application's database was not in place and has been deleted by unauthorized user. The most likely cause was SQL injection. The security team investigated vulnerabilities and determined a loop hole for such a scenario. A user authenticating to a web service by using a trick on user password field by setting value to:

```
password' OR 1=1
```

will actually cause database server to run:

```
SELECT id FROM users WHERE username='username' AND
password='password' OR 1=1'
```

Hence forth bypassing authentication and getting to the administrator privileges. To resolve the issue the engineering team modified the software application patches to effect login input forms sanitization

removed malicious code elements like single quotes and turned off visibility of database errors on production sites as a preventative measures.

Prevention measures

The engineering team underlined tasks to be implemented to prevent such occurrences outlined as follows:-

- ✓ continuous updating web development environment, technologies and using latest version will enhancing SQLi protection.
- ✓ Always to be precarious with any user input contents within web page or application
- ✓ To deploy more efficient web application firewall(WAF) and sanitize inputs temporarily, somehow this isolates database and applications from SQLi.
- ✓ Advise to scan regularly so that no sql injections were introduced by any external modules or libraries.
- ✓ Always follow best practices: to use parameterized queries or stored procedures. Attack surface reduction limiting users capabilities and roles to bare minimum required.
- ✓ To encrypt confidential data stored in database. Just in case you may be opted to add another layer of protection make salting encrypted hashes.