DNS Data Exfiltration

Using SQL Injection

Bob Ricks
G2, Inc.
SQL Injection

- We assume knowledge of how it works
- Basic types of data exfiltration
  - Verbose
    - Displayed on page
    - Error based
  - Blind
    - Timing
    - HTTP Request
    - Change in page
    - DNS Exfiltration
Related Work on DNS Exfiltration

• [http://pentestmonkey.net/blog/mssql-dns/](http://pentestmonkey.net/blog/mssql-dns/)

• David Litchfield: *The Oracle Hacker’s Handbook: Hacking and Defending Oracle*

• Squeeza
Attacking Oracle

- Because it’s there, and out there
- Most of the DNS Exfiltration tools attack MS-SQL Server
- Until Oracle 11g, access to UTL_INADDR defaulted to on and unprotected. Access to UTL_HTTP defaults to on, but Oracle recommends turning it off unless needed.
Why we are here

- **DNS is Usually available**
  - HTTP connections should be blocked
  - There is usually a DNS path available
    - Even if the database has no outbound comms
    - DNS server for DMZ will probably forward requests

- **Speed**
  - Timing/change in page extract ~1 bit per injection

- **Completeness**
  - Non-standard table and column names
  - Data types
What we built

• Tool to exfiltrate arbitrary data from Oracle
• Automatically generates injection strings
• Receives and processes DNS queries
• Asks for additional information based on responses from the database
Our Design

• Submit a number of queries in each round
  • We know from the position in the DNS request which subquery each field matches
  • Can configure how many subqueries and maximum length of each return value
• Random characters plus query number
• Use a short domain name, xg2.us
  • e.g. 0414243.DATABASE.sal0.xg2.us
What we learned

- **DNS Restrictions**
  - Total size 248 characters, including overhead
    - Require use of entire domain, own domain name
  - Each field needs to be 1-63 characters
    - Each subquery must return 1 column and 1 row
How we process data types

- **RAW**
  - Uses approximately twice as many characters
  - Nothing has to be changed, all chars valid
- **Character strings**
  - Good if all characters and numbers
  - Need conversion if there are spaces
  - Marker to determine if truncated
- **Numbers**
Tool used on HR Schema

- Standard HR Schema

Displaying Schema

User:
Attributes:
  authentication => DATABASE
  username => HR
  web_server_internal_ip => 127.0.0.1
  language => AMERICAN_AMERICA.WE8MSWIN1252
  database_ip => 192.168.10.93
  lang => US
  web_host => hawker

8 Tables
Table: "USERS" has 2 columns and approximately 1 rows
  Column: "USERNAME" (VARCHAR2)
  Column: "PASSWORD" (VARCHAR2)
Table: "COUNTRIES" has 3 columns and approximately 25 rows
  Column: "COUNTRY_ID" (CHAR)
  Column: "COUNTRY_NAME" (VARCHAR2)
  Column: "REGION_ID" (NUMBER)
Table: "EMPLOYEES" has 11 columns and approximately 107 rows
  Column: "EMPLOYEE_ID" (NUMBER)
  Column: "FIRST_NAME" (VARCHAR2)
  Column: "LAST_NAME" (VARCHAR2)
  Column: "EMAIL" (VARCHAR2)
  Column: "PHONE_NUMBER" (VARCHAR2)
  Column: "HIRE_DATE" (DATE)
  Column: "JOB_ID" (VARCHAR2)
  Column: "SALARY" (NUMBER)
  Column: "COMMISSION_PCT" (NUMBER)
  Column: "MANAGER_ID" (NUMBER)
  Column: "DEPARTMENT_ID" (NUMBER)
  ... (4 More Tables)

- Time 5 min, ALL data
Absinthe on same DB

- Graph shows
  - Initialization
  - Schema Name
  - Table names
- 5 Took minutes
- Our tool got basically all this in 6 seconds
COUNTRIES

• Absinthe is getting column names, data types, etc.
• Took about 5 minutes
• Much higher CPU utilization on
### Table: “USERS”

<table>
<thead>
<tr>
<th>USERNAME</th>
<th>PASSWORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>password</td>
</tr>
<tr>
<td>bob</td>
<td>;alfkjsdj023jr;oajsdco890asfdja023j</td>
</tr>
</tbody>
</table>

**Another example**

<table>
<thead>
<tr>
<th>USERNAME</th>
<th>PASSWORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iamgettingtiredofcomingupwithfakeusernamesandpasswords</td>
<td>&quot;Thisisjustpainfultohavetokeepdoingthisajf0923ja09a0fja}{F03927&quot;</td>
</tr>
</tbody>
</table>
Wireshark
What the tool does not do

- Find SQL injection sites for you
- Does not process “long” data type because you cannot use functions on it
  - Extensive use of the following functions
    - LENGTH()
    - SUBSTR()
    - UTL_RAW cast to raw()
Future Work

- Retry queries/fields that failed
- Create GUI front end
  - Would work well on a web server since we could have the web server control a domain
  - Specify target, parameters, cookies
- Harden tool
Prevention

- Revoke privileges on UTL_INADDR to Oracle user used by web pages
- No outgoing DNS requests from DMZ
- Fix SQL injection sites
- There are no good fixes for bad programming
- Always check ALL input from “users”
  - Strings, passwords, cookies
- Double-check login information
Summary

- SQL Injection is bad news (in a good way)
- DNS exfiltration can be very effective
- DBAs should block DNS for web users
- Web programmers should guard against SQL injection
- Parameterized SQL
Extra Slides

- String strategy
- Additional data tables
  - Stress nonstandard table names
  - File names are URL Encoded
  - Varying data types
String Strategy

• If possible and starts with non-'0' pull as is
• If necessary convert to ‘0’ plus raw (hex)
• Ask for substring of allowable length
• If return is maximum length
  • Ask for length
Table: EMPTY

"BLANK","EMPTY","NONE","NIL"
<table>
<thead>
<tr>
<th>NAME</th>
<th>COMMENTS</th>
<th>KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>comments</td>
<td>41</td>
</tr>
<tr>
<td>null</td>
<td></td>
<td>25</td>
</tr>
<tr>
<td>null</td>
<td>70.17.254.77</td>
<td>27</td>
</tr>
<tr>
<td>null</td>
<td>127.0.0.1</td>
<td>28</td>
</tr>
<tr>
<td>null</td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>null</td>
<td>172.16.1.102</td>
<td>29</td>
</tr>
<tr>
<td>null</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>null</td>
<td>127.0.0.1,12345678901234568790</td>
<td>36</td>
</tr>
<tr>
<td>null</td>
<td>70.17.254.75</td>
<td>36</td>
</tr>
<tr>
<td>null</td>
<td>209.35.68.205</td>
<td>37</td>
</tr>
<tr>
<td>null</td>
<td>64.236.91.24</td>
<td>39</td>
</tr>
<tr>
<td>null</td>
<td>why isn't the terminal working?</td>
<td>42</td>
</tr>
<tr>
<td>&quot;Valid $ Column&quot;,&quot;Weird@@@&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;39 digits&quot;,1234567890123456789012345678901234567890</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;really 39 digits&quot;,1234567890123456789012345678901234567890</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;40 digits&quot;,12345678901234567890123456789012345678901234567890</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;42 digits&quot;,123456789012345678901234567890123456789012345678900</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;42 digits round up&quot;,123456789012345678901234567890123456789100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;null&quot;,</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table: “VALID `@$!()%$` TABLE NAME”

"THAT`S ALL FOLKS","TRY `THIS` ON FOR SIZE ***","Let`s Try Precision/No Scale--","bang"

"first line",99,1234567,0