Sage 50 Payroll 2012 Authentication Bypass

Version Number 18.00.031
SBD Desktop Version 2.0.0.160
Website http://www.sage.co.uk/sage-50-payroll
Author – Richard Davy
Email – rmdavy@rmdavy.karoo.co.uk
Tools – Immunity Debugger http://debugger.immunityinc.com/

Preamble

Sage 50 Payroll offers three levels of security:

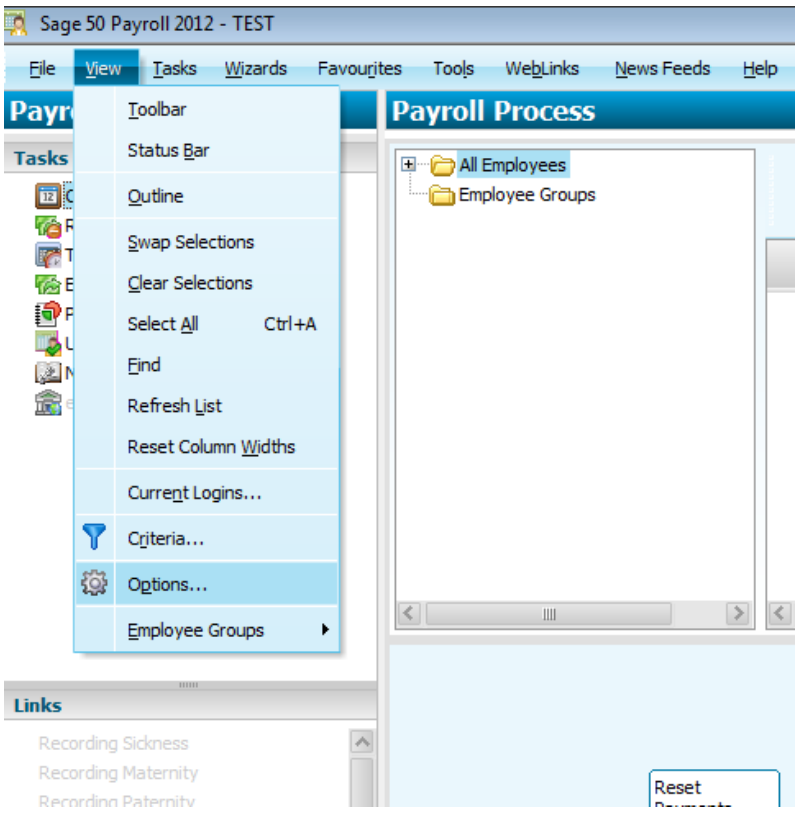
1) a password can be required to open the program
2) an individual username and password is then required to access the information stored in the database
3) database encryption

The aim of this paper is, using Immunity Debugger to demonstrate how to bypass these levels of security.

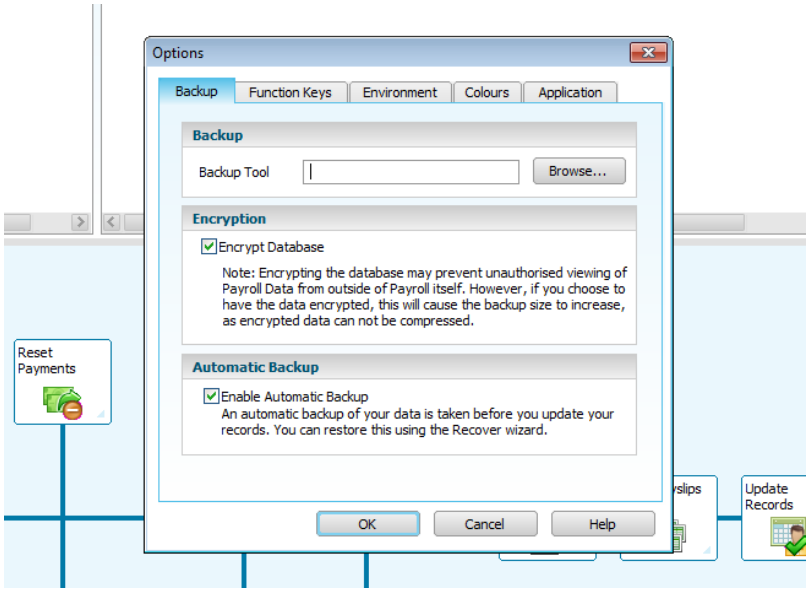
Let's get started- Setup Phase

So, start by firing up Sage Payroll and the first thing that we will do is turn on database encryption.

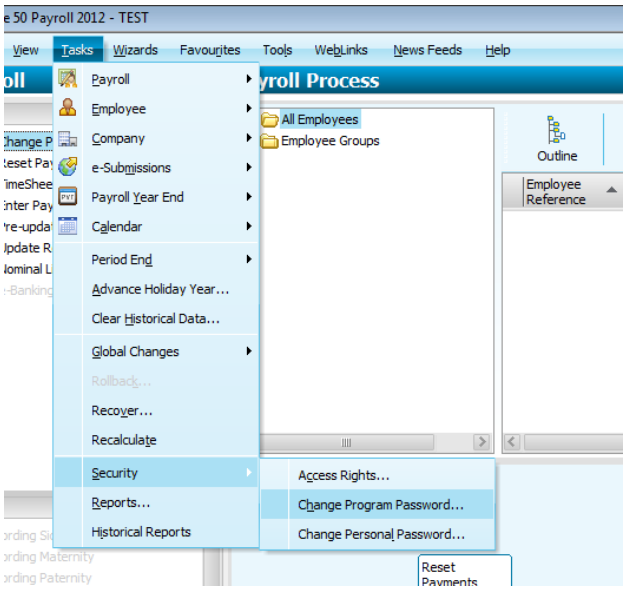
To do this we click on View and then Options

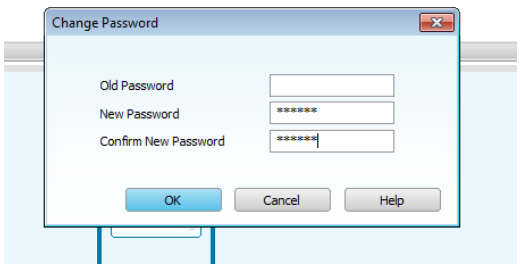


We then click on Encrypt Database and OK



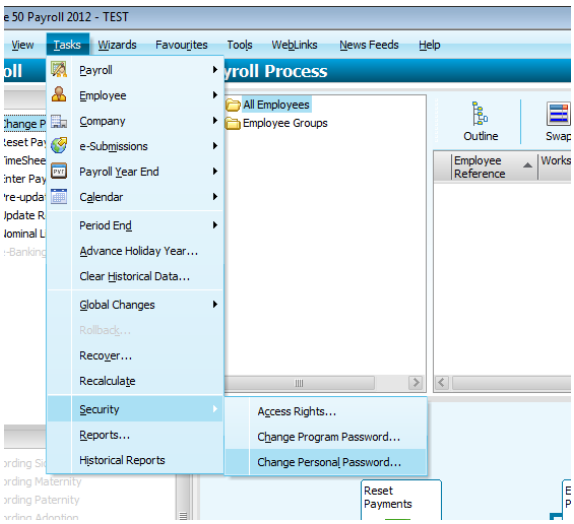
Next thing to do is to turn on the request for a password when we open the application. We achieve this by clicking on Tasks, Security and then Change Program Password. Enter a new password and then click ok



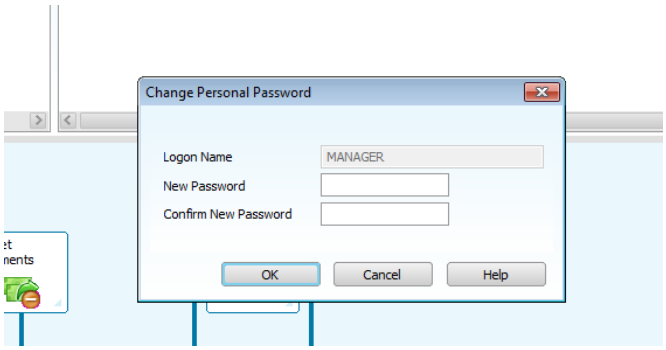


We will then add/change the Manager password. For those of you that don't know the MANAGER account in Sage is the top account and has highest level access. This is the account we are going for.

So we click on Tasks, Security and Change Personal Password



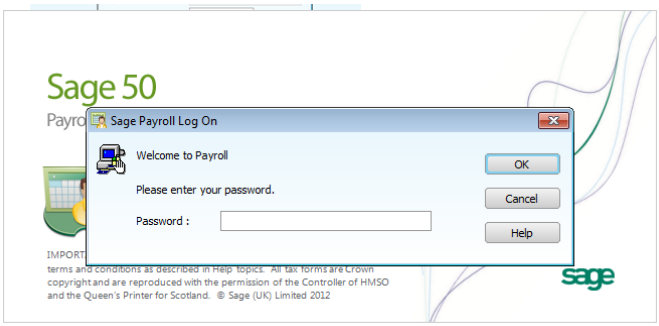
Enter a New Password and then Click OK



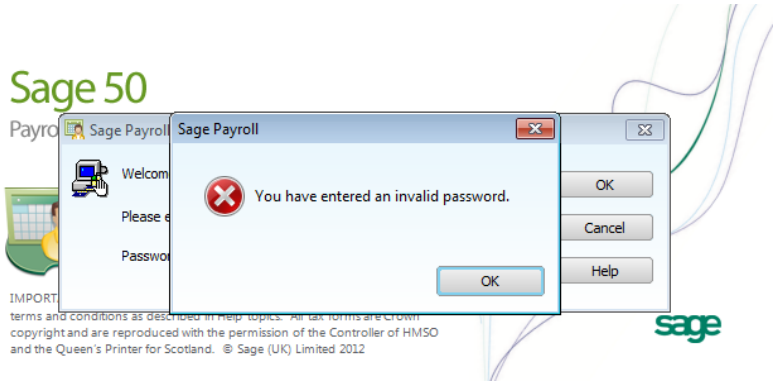
Great, so we have now set up our authentication checks and it's now time to start finding a way around them! So quit Sage Payroll and start it up again.

The Fun Phase!

Fire up Sage Payroll and you should be greeted with the following screen



If you enter an incorrect password and click on OK, you are greeted with an error message

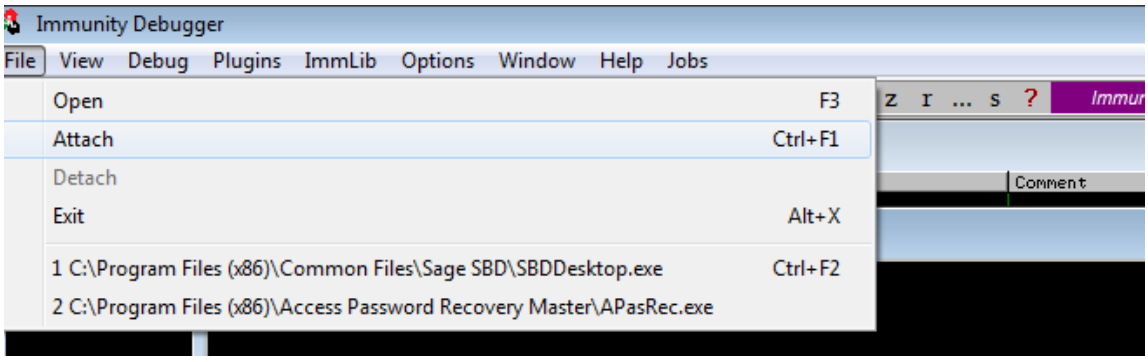


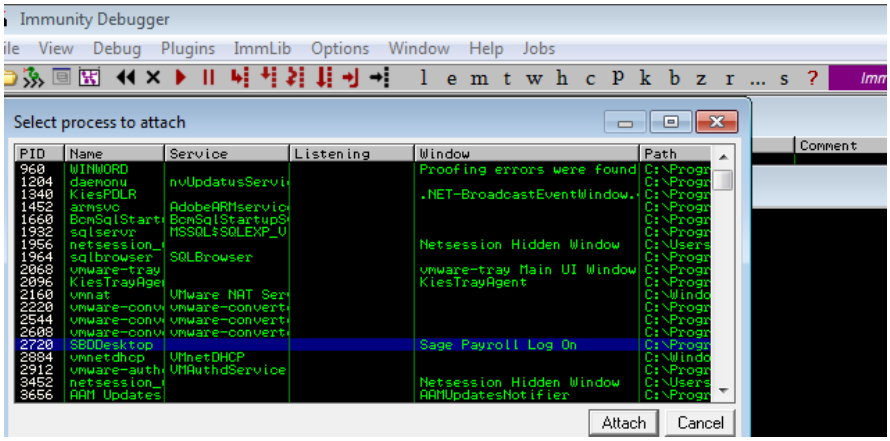
This is actually the bit that we are interested in, because often when poor methods of authentication are put in place we can trace to before the messagebox is called and simply modify a conditional statement to force it to believe that we have entered the correct password.

So how do we do this?

On the logon screen I have entered a password of 446697 (my actual password is 321654). Whilst the Error Message is visible fire up Immunity Debugger and see whether the string of 446697 has been copied into memory or not.

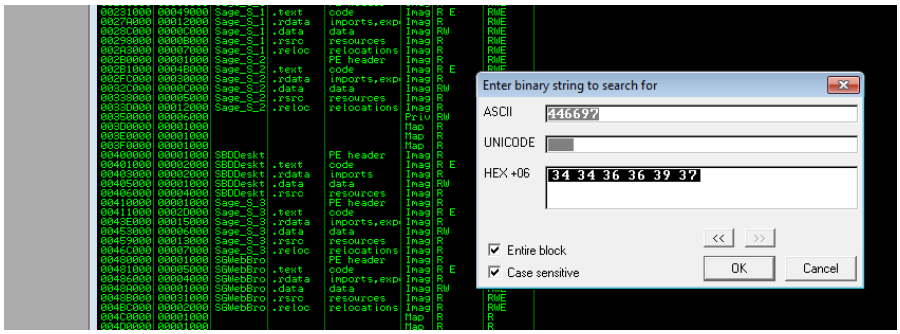
So from IM, let's attach to the Sage Process – File/Attach



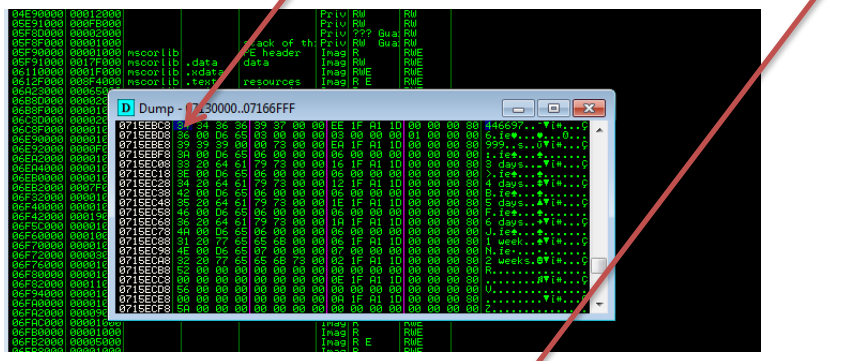


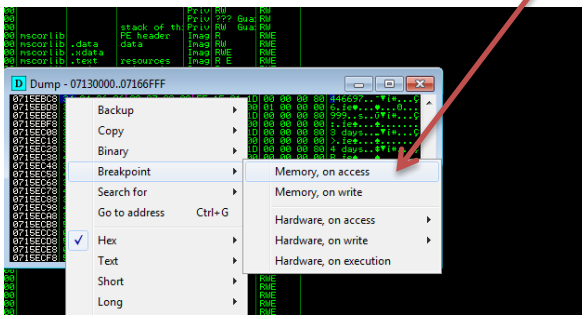
Click on Sage Payroll Log On which is the Title of our Log On window and then click on Attach

Press Alt+M to bring up the Memory Map window and then Press Ctrl+B to bring up the search window. Enter 446697 (in my case) into the ASCII window and then click on OK

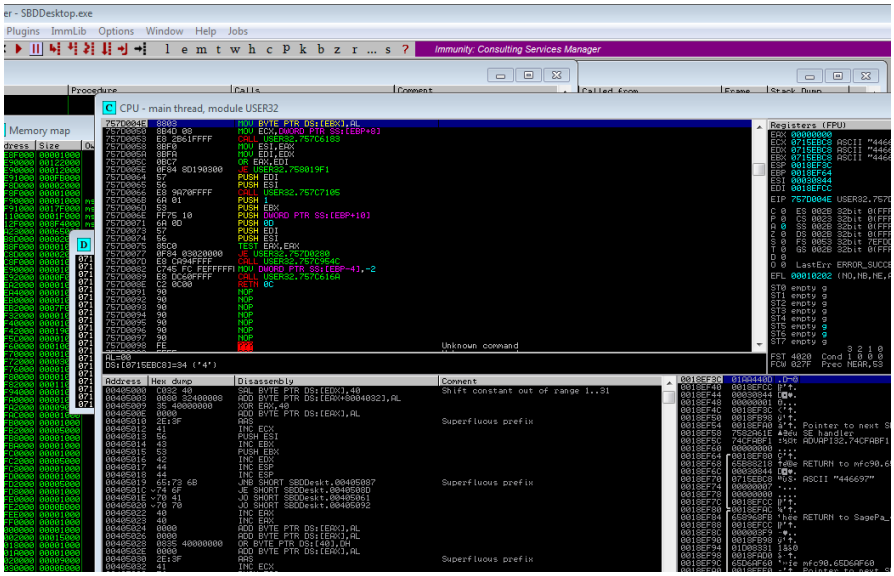


The following box should pop up (note the memory locations will not be the same)
Right Click on the first 34 and select Breakpoint, Memory, on Access

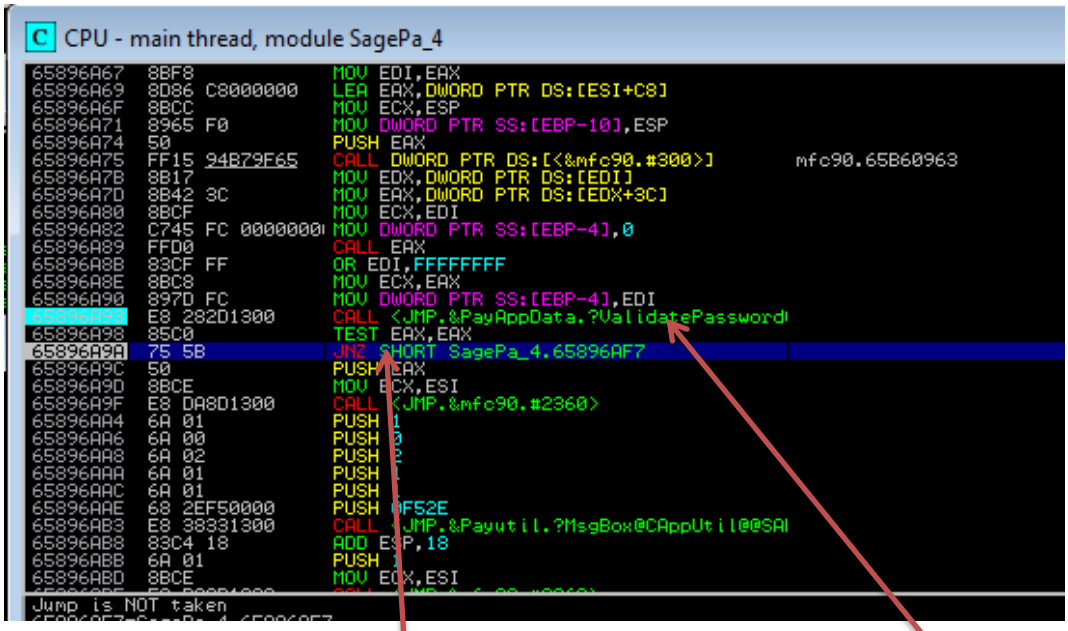




Go back to Sage Payroll and click OK to get rid of the error message and then click on OK again to re-authenticate. This time Immunity Debugger should step in and the memory breakpoint that we set should be triggered.

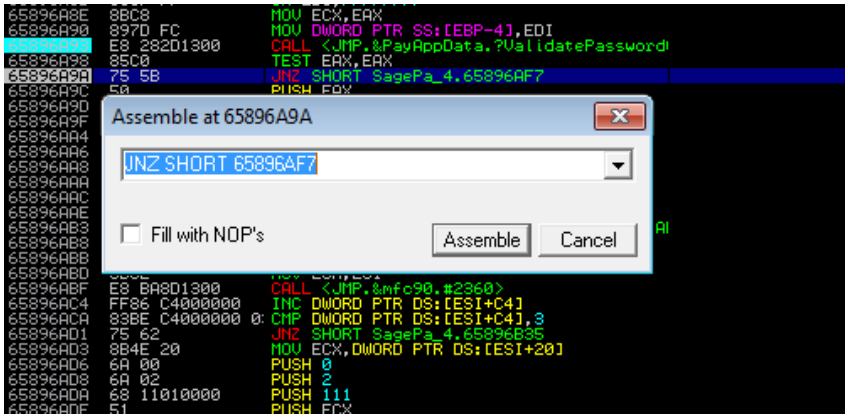


Single Step through the code by pressing F8 until you reach



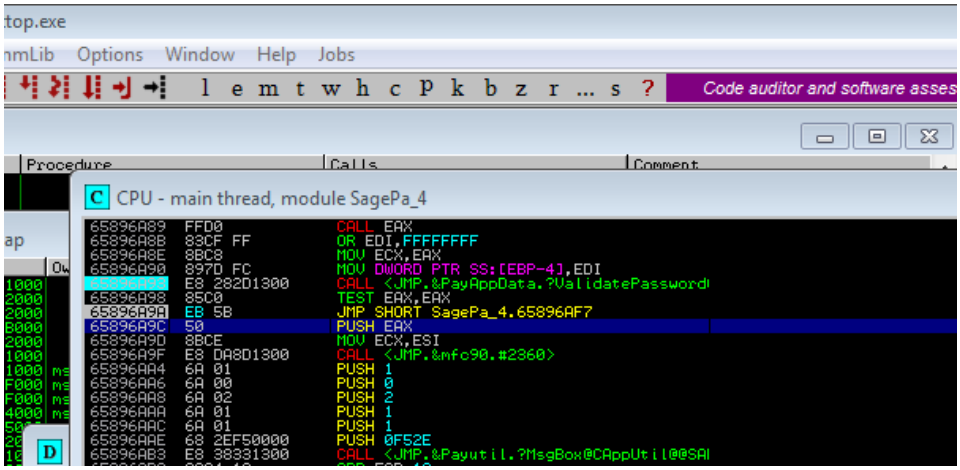
As I am sure you can deduct, the password validation routine is held in this call and on its return there is a test to see whether to pop the error message or jump past it.

We need to modify the code here to always jump and validate as ok. So double click on this line

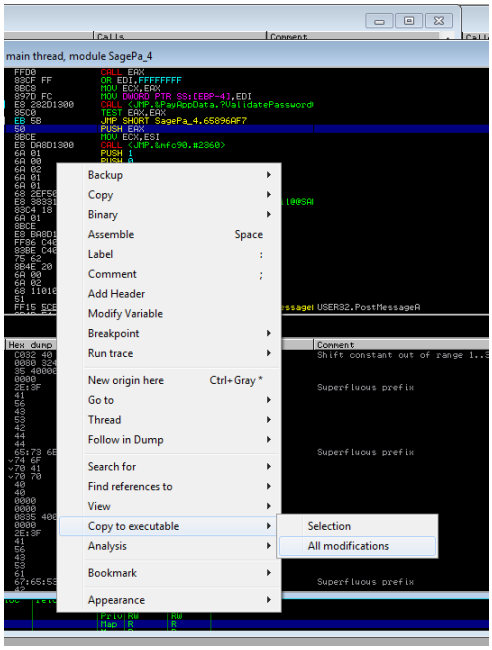


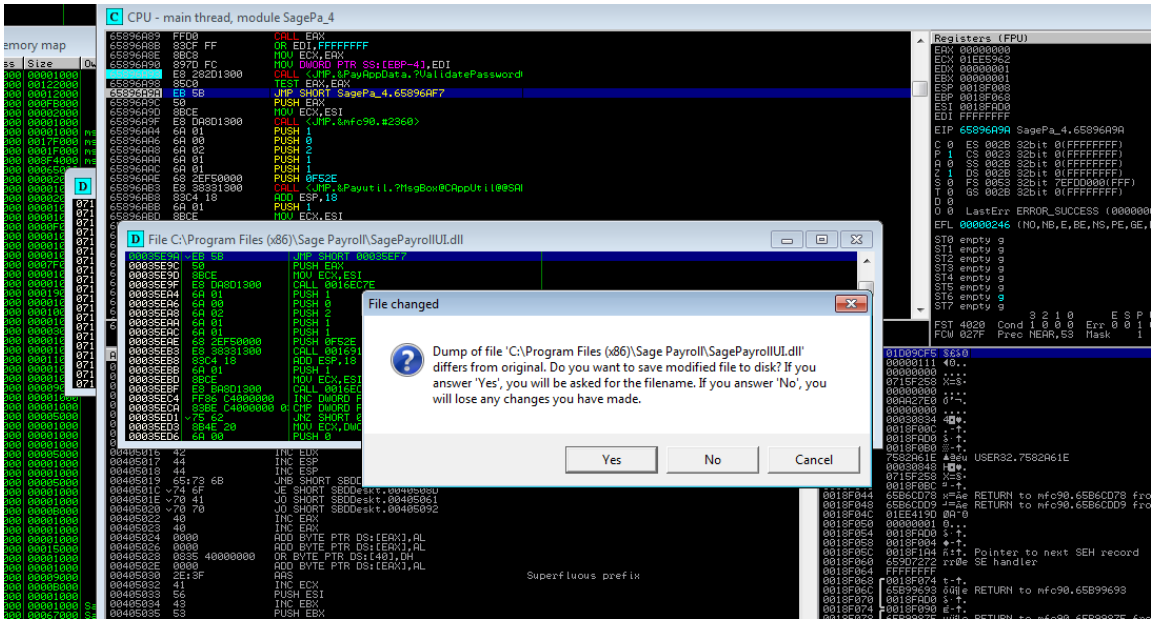
And change the code to read JMP SHORT 65896AF& then click on Assemble and close the window.

Your code should now look like



To save this change permanently, right click on the code and select Copy to executable, selection and All Modifications

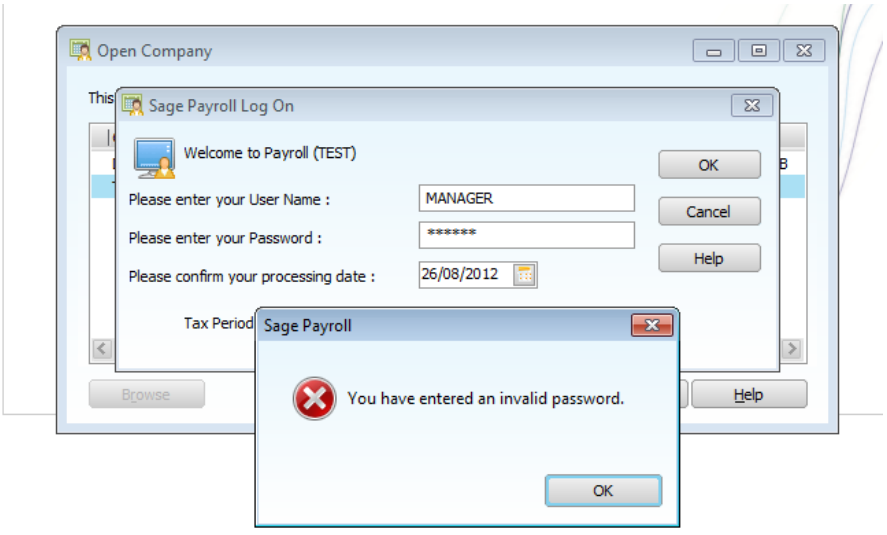


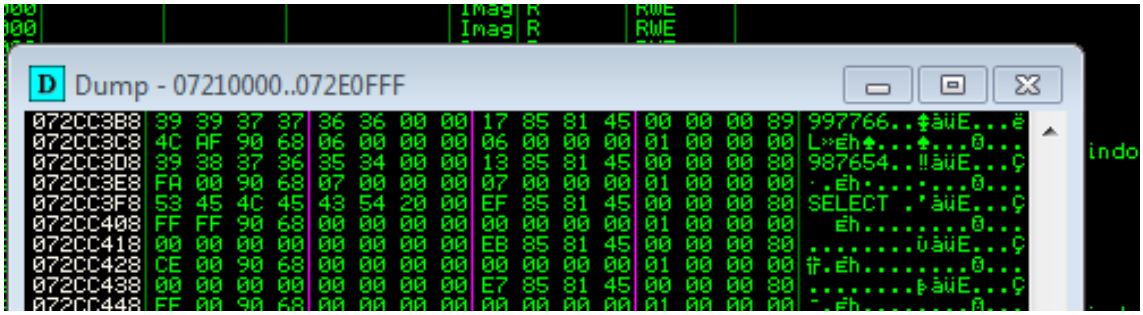


Click on Yes and then Save. If you continue the execution of the program via Immunity or close it and reopen it you will see that you can now enter any password in the box and that it will be accepted.

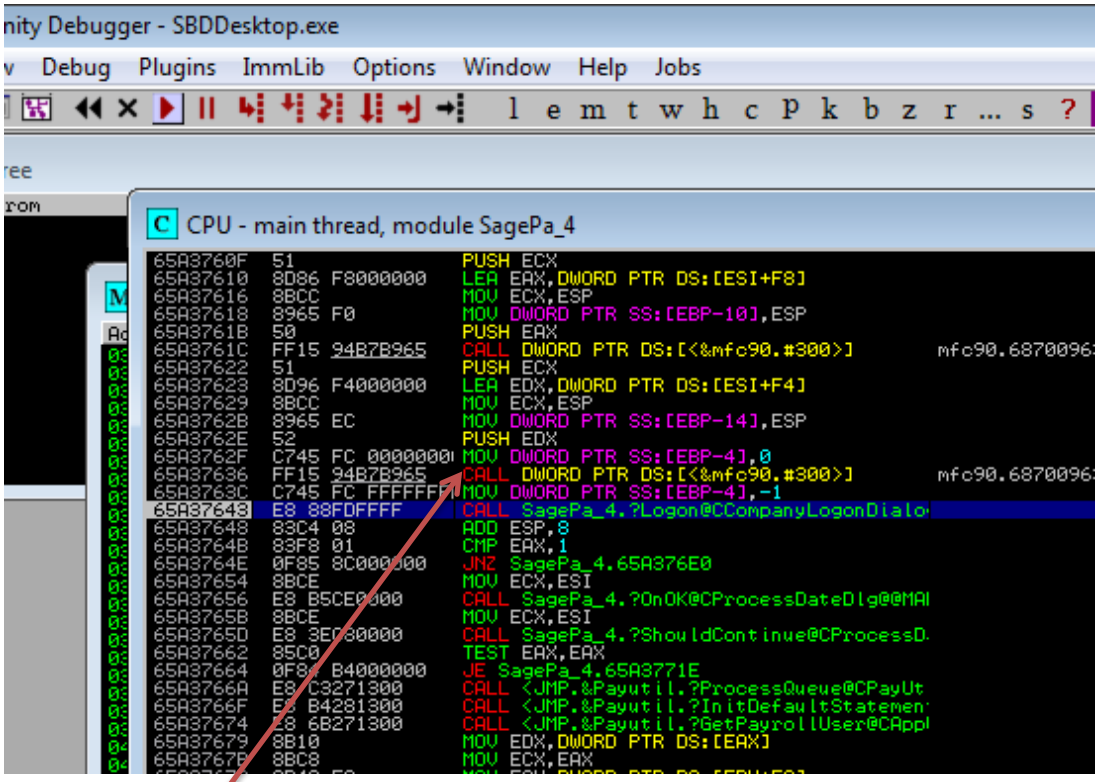
Voila - Part one complete. We will now employ the same technique to bypass our Manager password.

From our Log On screen I entered the password of 997766 and clicked on OK whilst the error box is open, fire up Immunity Debugger and attach as before and do a memory search for 997766.





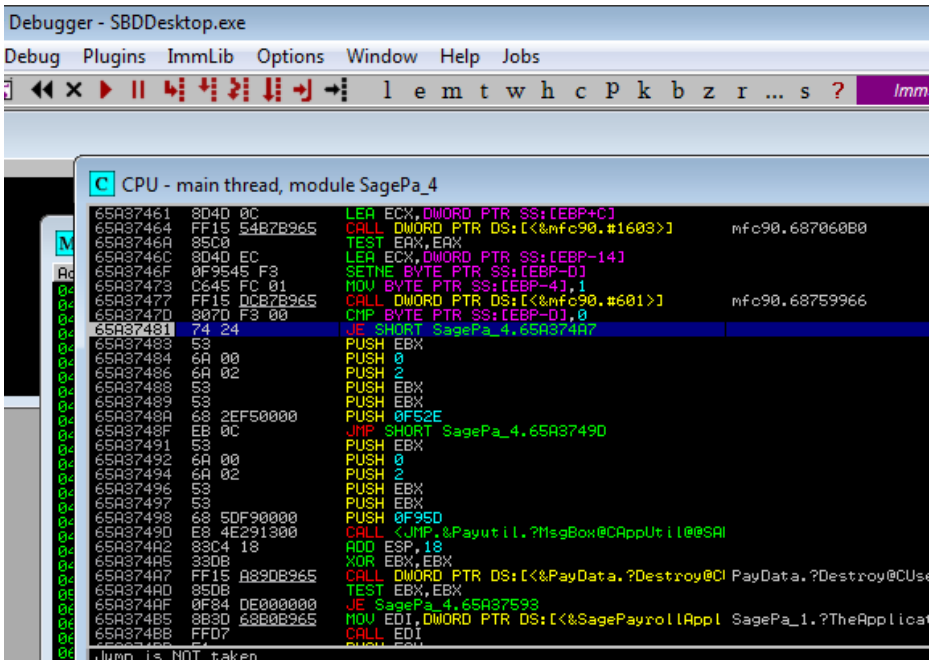
This should reveal 997766 is memory. As before set a breakpoint and continue with program execution. Click on OK again to trigger the breakpoint and cycle to the code below.



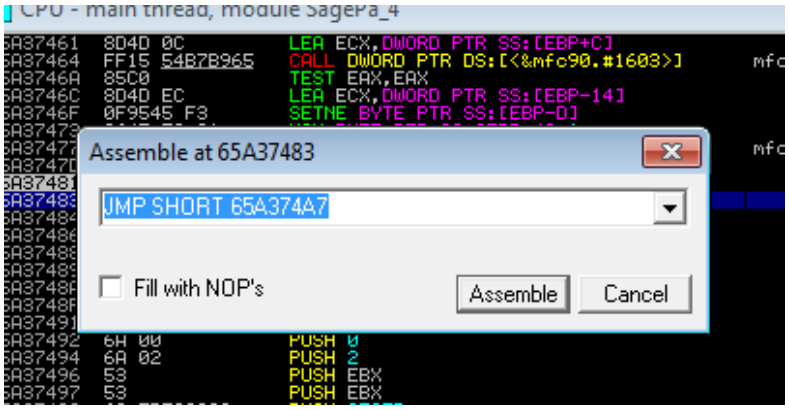
If you cycle to the line highlighted a message box will pop up which indicates that we need to trace the call here

We do this by repeating the process but when we get to this line, we press F7 instead of F8 to Step Into instead of Step Over.

Repeat the above and step into the call using F7 and single step using F8 until you reach the following code



Change the JE to JMP by double clicking to change the code. Click Assemble and then close the window.



Save the data to file as detailed above.

Voila - Part two complete

If you close the Sage Payroll application and reopen it, you should now find that you no longer need the correct password for the program password or for logging on as Manager.

Part 3 – Database Encryption

I mentioned above that the program also allows you to encrypt the database which, as you will see, reveals to be a completely pointless exercise as we can bypass any authentication forms.

Attack Vector

A modified version of the SagePayrollUI.dll which is located in the Sage working directory could be placed on the victims machine which would allow for the attacker to come back at any time and have access or if it is possible to obtain a backup copy of a database a modified version could be copied to an attackers computer and the attacker could to view/amend data etc. at their leisure.

Summary/Recommendations

With regard to this program, users are made to feel secure and there are certainly areas for improvement.

Anyone using this product certainly needs to be careful who can access backups as there is no real encryption. Care also needs to be taken as to who can access the computer which this software is installed on.

The file locations and bytes which have been patched are listed below. For those that are wondering why there are three locations one of them corresponds to the error message when we try and change the program access password and we don't know the original password.

00019803 – 75 to EB
00035E9A – 75 to EB
00036880 – 74 to EB