

REVERSE ENGINEERING

SRE

- Software Reverse Engineering
 - Also known as Reverse Code Engineering (RCE)
 - Or simply “reversing”
- Can be used for good...
 - Understand malware
 - Understand legacy code
- ...or not-so-good
 - Remove usage restrictions from software
 - Find and exploit flaws in software
 - Cheat at games, etc.

SRE

- We assume that
 - Reverse engineer is an attacker
 - Attacker only has exe (no source code)
- Attacker might want to
 - Understand the software
 - Modify the software
- SRE usually focused on Windows
- So we'll focus on Windows

SRE TOOLS

- Disassembler
 - Converts exe to assembly — as best it can
 - Cannot always disassemble correctly
 - In general, it is not possible to assemble disassembly into working exe
- Debugger
 - Must step thru code to completely understand it
 - Labor intensive — lack of automated tools
- Hex Editor
 - To patch (make changes to) exe file
- Regmon, Filemon, VMware, etc.

SRE TOOLS

- IDA Pro is the top-rated disassembler
 - Cost is a few hundred dollars
 - Converts binary to assembly (as best it can)
- SoftICE is “alpha and omega” of debuggers
 - Cost is in the \$1000's
 - Kernel mode debugger
 - Can debug anything, even the OS
- OllyDbg is a high quality shareware debugger
 - Includes a good disassembler
- Hex editor — to view/modify bits of exe
 - UltraEdit is good — freeware
 - HIEW — useful for patching exe
- Regmon, Filemon — freeware

WHY IS A DEBUGGER NEEDED?

- Disassembler gives static results
 - Good overview of program logic
 - But need to “mentally execute” program
 - Difficult to jump to specific place in the code
- Debugger is dynamic
 - Can set break points
 - Can treat complex code as “black box”
 - Not all code disassembles correctly
- Disassembler and debugger both required for any serious SRE task

SRE NECESSARY SKILLS

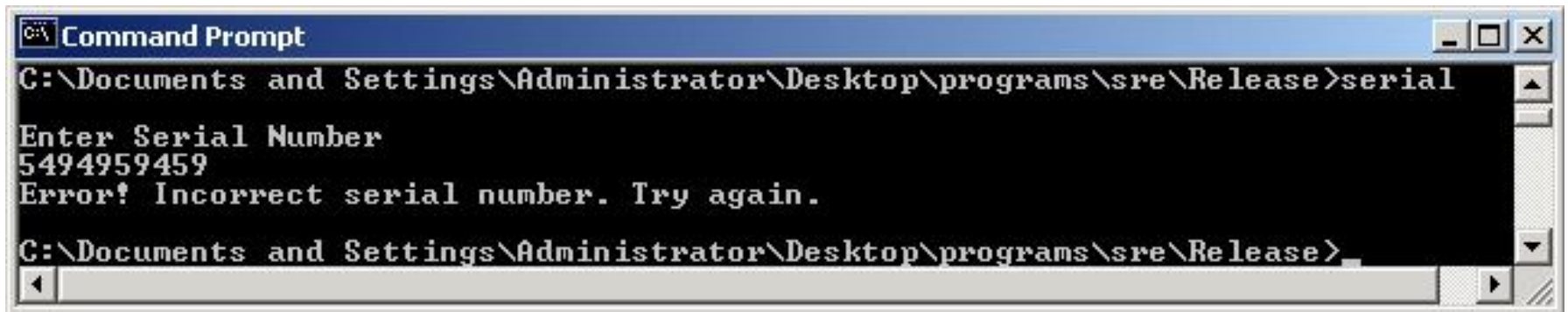
- Working knowledge of target assembly code
- Experience with the tools
 - IDA Pro — sophisticated and complex
 - SoftICE — large two-volume users manual
- Knowledge of Windows Portable Executable (PE) file format
- Boundless patience and optimism
- SRE is tedious and labor-intensive process!

SRE EXAMPLE

- Consider simple example
- This example only requires disassembler (IDA Pro) and hex editor
 - Disassemble to understand code
 - Want to patch the code
- For most real-world code, also need a debugger (SoftICE or OllyDbg)

SRE EXAMPLE

- Program requires serial number
- But Trudy doesn't know the serial number!



```
Command Prompt
C:\Documents and Settings\Administrator\Desktop\programs\sre\Release>serial
Enter Serial Number
5494959459
Error! Incorrect serial number. Try again.
C:\Documents and Settings\Administrator\Desktop\programs\sre\Release>
```

Can you find the serial number?

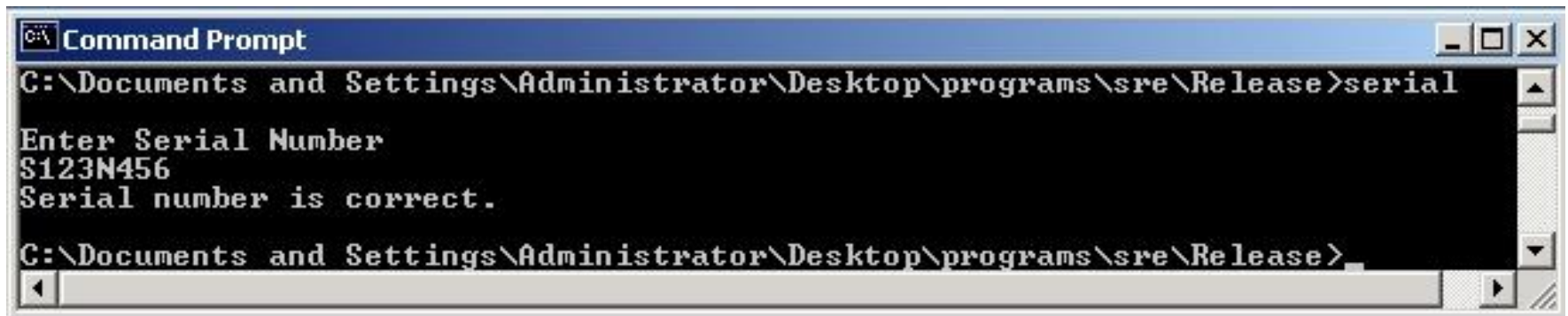
SRE EXAMPLE

```
.text:00401003      push    offset aEnterSerialNum ; "\nEnter Serial Number\n"
.text:00401008      call     sub_4010AF
.text:0040100D      lea      eax, [esp+18h+var_14]
.text:00401011      push    eax
.text:00401012      push    offset aS          ; "%S"
.text:00401017      call     sub_401098
.text:0040101C      push    8
.text:0040101E      lea      ecx, [esp+24h+var_14]
.text:00401022      push    offset aS123n456 ; "S123N456"
.text:00401027      push    ecx
.text:00401028      call     sub_401060
.text:0040102D      add      esp, 18h
.text:00401030      test     eax, eax
.text:00401032      jz       short loc_401045
.text:00401034      push    offset aErrorIncorrect ; "Error! Incorrect serial number."
.text:00401039      call     sub_4010AF
```

Looks like serial number is S123N456

SRE EXAMPLE

- Try the serial number S123N456



```
Command Prompt
C:\Documents and Settings\Administrator\Desktop\programs\sre\Release>serial
Enter Serial Number
S123N456
Serial number is correct.
C:\Documents and Settings\Administrator\Desktop\programs\sre\Release>
```

It works!

Can we do better?

SRE EXAMPLE

```
.text:00401003      push    offset aEnterSerialNum ; "\nEnter Serial Number\n"
.text:00401008      call     sub_4010AF
.text:0040100D      lea      eax, [esp+18h+var_14]
.text:00401011      push    eax
.text:00401012      push    offset aS                ; "%s"
.text:00401017      call     sub_401098
.text:0040101C      push    8
.text:0040101E      lea      ecx, [esp+24h+var_14]
.text:00401022      push    offset aS123n456 ; "S123N456"
.text:00401027      push    ecx
.text:00401028      call     sub_401060
.text:0040102D      add      esp, 18h
.text:00401030      test     eax, eax
.text:00401032      jz       short loc_401045
.text:00401034      push    offset aErrorIncorrect ; "Error! Incorrect serial number."
.text:00401039      call     sub_4010AF
```

And hex view...

```
.text:00401010  04 50 68 84 80 40 00 E8-7C 00 00 00 6A 08 8D 4C
.text:00401020  24 10 68 78 80 40 00 51-E8 33 00 00 00 83 C4 18
.text:00401030  85 C6 74 11 68 4C 80 40-00 E8 71 00 00 00 83 C4
.text:00401040  04 83 C4 14 C3 68 30 80-40 00 E8 60 00 00 00 83
```

SRE EXAMPLE

```
.text:00401003      push    offset aEnterSerialNum ; "\nEnter Serial Number\n"
.text:00401008      call    sub_4010AF
.text:0040100D      lea     eax, [esp+18h+var_14]
.text:00401011      push    eax
.text:00401012      push    offset aS          ; "%s"
.text:00401017      call    sub_401098
.text:0040101C      push    8
.text:0040101E      lea     ecx, [esp+24h+var_14]
.text:00401022      push    offset aS123n456 ; "S123N456"
.text:00401027      push    ecx
.text:00401028      call    sub_401060
.text:0040102D      add     esp, 18h
.text:00401030      test    eax, eax
.text:00401032      jz      short loc_401045
.text:00401034      push    offset aErrorIncorrect ; "Error! Incorrect serial number."
.text:00401039      call    sub_4010AF
```

test eax,eax gives AND of eax with itself

- Result is 0 only if eax is 0
- If test returns 0, then jz is true

Trudy wants jz to always be true!

Can Trudy patch exe so that jz always true?

SRE EXAMPLE

Can you modify exe so that jz always true?

```
.text:00401003      push    offset aEnterSerialNum ; "\nEnter Serial Number\n"
.text:00401008      call    sub_4010AF
.text:0040100D      lea     eax, [esp+18h+var_14]
.text:00401011      push    eax
.text:00401012      push    offset aS              ; "%5"
.text:00401017      call    sub_401098
.text:0040101C      push    8
.text:0040101E      lea     ecx, [esp+24h+var_14]
.text:00401022      push    offset aS123n456 ; "S123N456"
.text:00401027      push    ecx
.text:00401028      call    sub_401060
.text:0040102D      add     esp, 18h
.text:00401030      xor     test    eax, eax
.text:00401032      jz      short loc_401045
.text:00401034      push    offset aErrorIncorrect ; "Error! Incorrect serial number."
.text:00401039      call    sub_4010AF
```


SRE EXAMPLE

- Edit serial.exe with hex editor

serial.exe

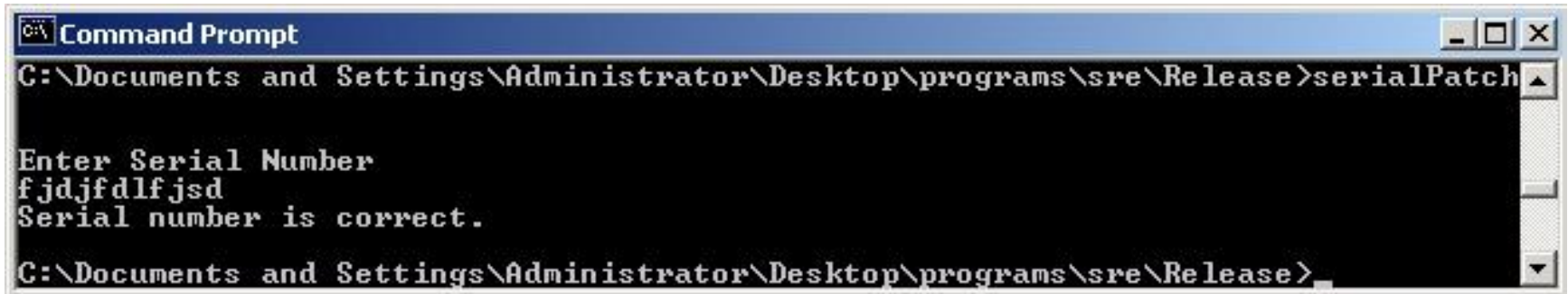
```
00001010h: 04 50 68 84 80 40 00 E8 7C 00 00 00 6A 08 8D 4C
00001020h: 24 10 68 78 80 40 00 51 E8 33 00 00 00 83 C4 18
00001030h: 85 CD 74 11 68 4C 80 40 00 E8 71 00 00 00 83 C4
00001040h: 04 83 C4 14 C3 68 30 80 40 00 E8 60 00 00 00 83
00001050h: C4 04 83 C4 14 C3 90 90 90 90 90 90 90 90 90
```

serialPatch.exe

```
-----
00001010h: 04 50 68 84 80 40 00 E8 7C 00 00 00 6A 08 8D 4C
00001020h: 24 10 68 78 80 40 00 51 E8 33 00 00 00 83 C4 18
00001030h: 33 CD 74 11 68 4C 80 40 00 E8 71 00 00 00 83 C4
00001040h: 04 83 C4 14 C3 68 30 80 40 00 E8 60 00 00 00 83
00001050h: C4 04 83 C4 14 C3 90 90 90 90 90 90 90 90 90
```

Save as serialPatch.exe

SRE EXAMPLE



```
Command Prompt
C:\Documents and Settings\Administrator\Desktop\programs\sre\Release>serialPatch

Enter Serial Number
fjdjfdlfjsd
Serial number is correct.

C:\Documents and Settings\Administrator\Desktop\programs\sre\Release>
```

The image shows a Windows Command Prompt window with a blue title bar. The window title is "Command Prompt". The current directory is "C:\Documents and Settings\Administrator\Desktop\programs\sre\Release". The user has executed the command "serialPatch". The program prompts the user to "Enter Serial Number". The user has entered "fjdjfdlfjsd". The program outputs "Serial number is correct.". The prompt is now "C:\Documents and Settings\Administrator\Desktop\programs\sre\Release>".

SRE EXAMPLE

serial.exe

```
.text:00401003
.text:00401008
.text:0040100D
.text:00401011
.text:00401012
.text:00401017
.text:0040101C
.text:0040101E
.text:00401022
.text:00401027
.text:00401028
.text:0040102D
.text:00401030
.text:00401032
.text:00401034
.text:00401039
```

```
push    offset aEnterSerialNum ; "\nEnter Serial Number\n"
call    sub_4010AF
lea     eax, [esp+18h+var_14]
push    eax
push    offset aS                ; "%5"
call    sub_401098
push    8
lea     ecx, [esp+24h+var_14]
push    offset aS123n456 ; "S123N456"
push    ecx
call    sub_401060
add     esp, 18h
test    eax, eax
jz      short loc_401045
push    offset aErrorIncorrect ; "Error! Incorrect serial number."
call    sub_4010AF
```

serialPatch.exe

```
.text:00401003
.text:00401008
.text:0040100D
.text:00401011
.text:00401012
.text:00401017
.text:0040101C
.text:0040101E
.text:00401022
.text:00401027
.text:00401028
.text:0040102D
.text:00401030
.text:00401032
.text:00401034
.text:00401039
```

```
push    offset aEnterSerialNum ; "\nEnter Serial Number\n"
call    sub_4010AF
lea     eax, [esp+18h+var_14]
push    eax
push    offset aS                ; "%5"
call    sub_401098
push    8
lea     ecx, [esp+24h+var_14]
push    offset aS123n456 ; "S123N456"
push    ecx
call    sub_401060
add     esp, 18h
xor     eax, eax
jz      short loc_401045
push    offset aErrorIncorrect ; "Error! Incorrect serial number."
call    sub_4010AF
```

SRE ATTACK MITIGATION

- Impossible to prevent SRE on open system
- But can make such attacks more difficult
- Anti-disassembly techniques
 - To confuse static view of code
- Anti-debugging techniques
 - To confuse dynamic view of code
- Tamper-resistance
 - Code checks itself to detect tampering
- Code obfuscation
 - Make code more difficult to understand