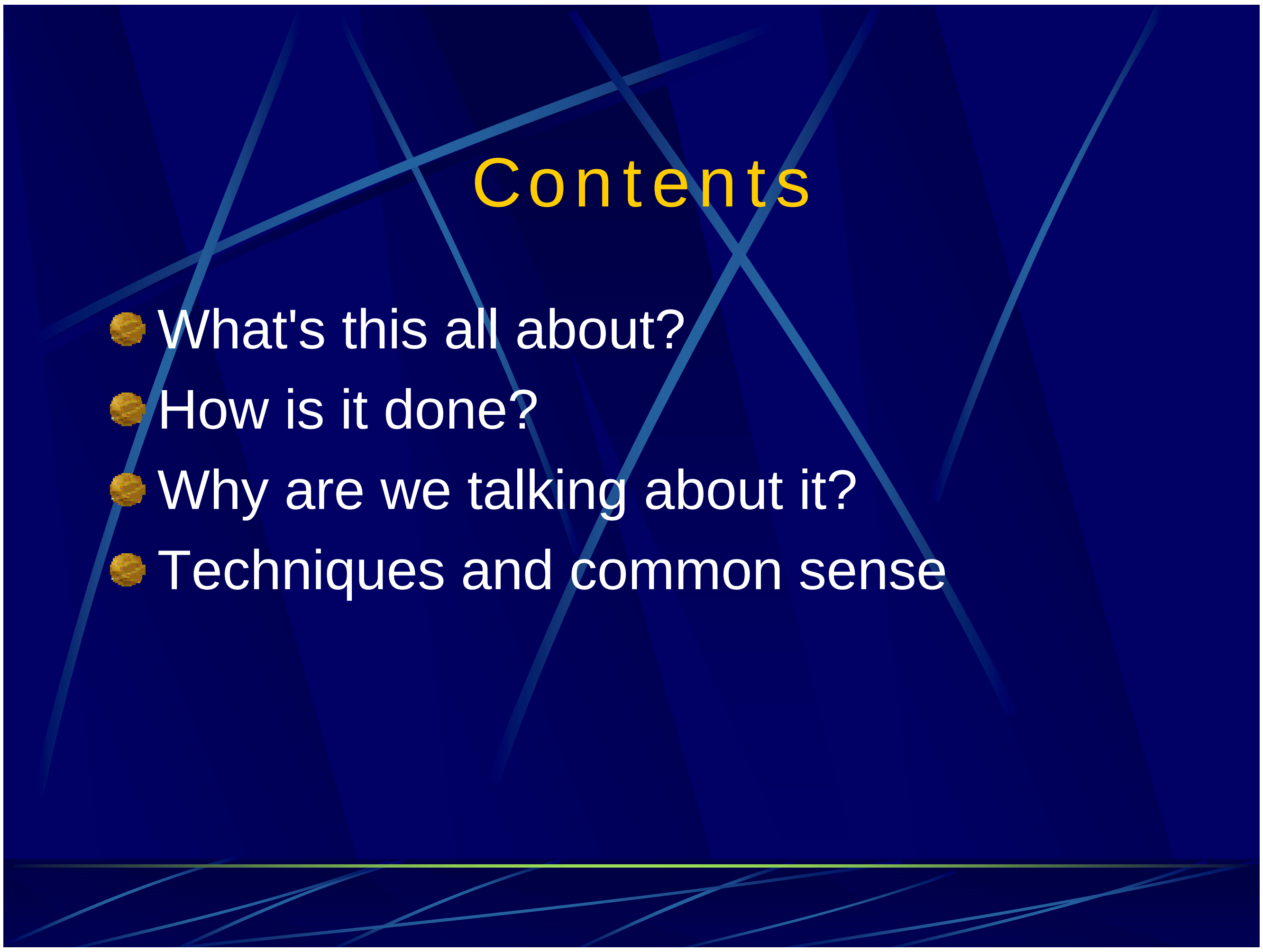
Reverse Engineering

"Protecting digital assets from RE attack"

Steve Ingram
Oren Ben-Menachem

Besser fri'er bevorent aider shpeter bevaint

Better caution at first than tears afterwards



Contents

- What's this all about?
- How is it done?
- Why are we talking about it?
- Techniques and common sense

What is this all about?

- What is reverse engineering?
- Why is it done?
- Who are the players?
- How is it done?

How is it done?

- Patience and understanding
- Tools
  - Steppers, Dry-listers
- Skills required
  - Understanding of:
    - System (including OpSys)
    - Language (assembly and source)
    - Compilers (how is assembly generated from higher level languages)
    - Developers and the process they use

Example : C Source

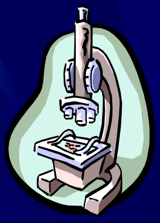
```
#include <stdio.h>
void main(void)
//<><><><><><><><><><><><><><><><><><><><><><><><>
// Input: None                                  //
// Output: predetermined    , printf * 2        //
// Termination: No constraint                   //
// Loop: None                                   //
// Variables: 3 * local int     //
// Memory alloc: None                           //
// Perf: O                      //
// Tests a value by /13, if T then OK           //
//<><><><><><><><><><><><><><><><><><><><><><><><>
{           int key;
            int test;
            int temp;

key = 0;
test = 0;
temp = 0;

key = 13; //a base prime
test = 38; //the input validation request

temp = test / key; //create a temporary value, to be used to identify if key and test are related

if (temp * key == test) // test if key and test are actually related, succeed or fail on result
            printf("success\n");
else
            printf("failure\n");
printf("finished\n"); //completion
}
```



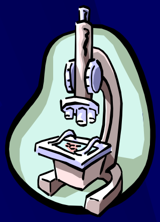
Relevant 8086 Source Segment

```
:00401000 55                   push ebp
:00401001 8BEC                 mov ebp, esp
:00401003 83EC0C               sub esp, 0000000C                # Setup local vars
:00401006 C745F800000000       mov [ebp-08], 00000000           # init the 3 locals to 0
:0040100D C745FC00000000       mov [ebp-04], 00000000
:00401014 C745F400000000       mov [ebp-0C], 00000000
:0040101B C745F80D000000       mov [ebp-08], 0000000D           #load 0x0D into var 2
:00401022 C745FC26000000       mov [ebp-04], 00000026           #load 0X26 into var 1
:00401029 8B45FC               mov eax, dword ptr [ebp-04]
:0040102C 99                   cdq
:0040102D F77DF8               idiv [ebp-08]                    #perform div
:00401030 8945F4               mov dword ptr [ebp-0C], eax      #return result to var 3
:00401033 8B45F4               mov eax, dword ptr [ebp-0C]
:00401036 0FAF45F8             imul eax, dword ptr [ebp-08]     #mul var1 to var 2
:0040103A 3B45FC               cmp eax, dword ptr [ebp-04]      #compare mul result to var 1
:0040103D 750F                 jne 0040104E                     #not equal jump to 0x0040104E

* Possible StringData Ref from Data Obj ->"success"
                               |
:0040103F 6830604000           push 00406030                    #push pointer to string
:00401044 E823000000           call 0040106C                    #call printf
:00401049 83C404               add esp, 00000004                #drop result from stack
:0040104C EB0D                 jmp 0040105B                     #jump to 0x0040105B

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0040103D(C)
* Possible StringData Ref from Data Obj ->"failure"
                               |
:0040104E 683C604000           push 0040603C                    #push pointer to string
:00401053 E814000000           call 0040106C                    #call printf
:00401058 83C404               add esp, 00000004                #drop result from stack

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0040104C(U)
* Possible StringData Ref from Data Obj ->"finished"
                               |
:0040105B 6848604000           push 00406048                    #push pointer to string
:00401060 E807000000           call 0040106C                    #call printf
:00401065 83C404               add esp, 00000004                #drop result from stack
:00401068 8BE5                 mov esp, ebp                     #tidy up
:0040106A 5D                   pop ebp
:0040106B C3                   ret
```



Why are we talking about it here?

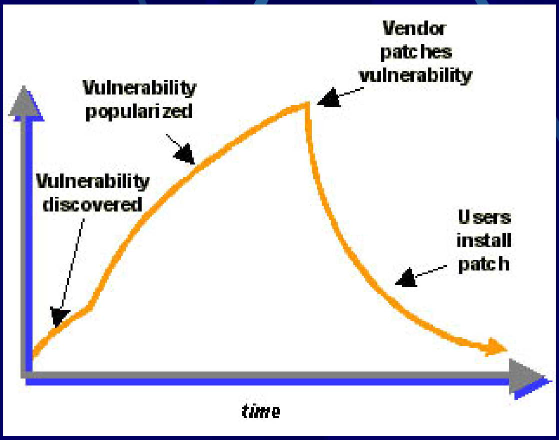
- You want to become a Reverse Engineer
- You want to *protect* digital assets from being compromised by RE techniques

Techniques

- Learn your art, be a craftsman
- Understand how your digital asset works / interacts
- Become an expert of the tools you use to protect or compromise
- Can you protect against a concerted attack? Do you want or need to?
- Is a risk management approach applicable?

Risk Management

What's risk management?
What are the trade-offs?



Design Considerations

- What's the risk?
- Why should a check only occur once?
- Why should it only occur against the whole key when its checked?
- Self heal
- RE states of Digital Assets don't occur in the wild
- Check yourself for intrusion
- Why hold keys in the clear?
- Why hold exports in the clear?

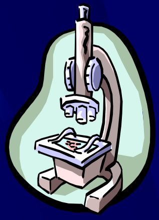
Example

```
                 :00401000 55                      push ebp
:00401001 8BEC                    mov ebp, esp
:00401003 83EC0C                  sub esp, 0000000C                          # Setup local vars
:00401006 C745F800000000          mov [ebp-08], 00000000          # init the 3 locals to 0
:0040100D C745FC00000000          mov [ebp-04], 00000000
:00401014 C745F400000000          mov [ebp-0C], 00000000
:0040101B C745F80D000000          mov [ebp-08], 0000000D          #load 0x0D into var 2
:00401022 C745FC26000000          mov [ebp-04], 00000026          #load 0X26 into var 1
:00401029 8B45FC                  mov eax, dword ptr [ebp-04]
:0040102C 99                      cdq
:0040102D F77DF8                  idiv [ebp-08]                   #perform div
:00401030 8945F4                  mov dword ptr [ebp-0C], eax     #return result to var 3
:00401033 8B45F4                  mov eax, dword ptr [ebp-0C]
:00401036 0FAF45F8                imul eax, dword ptr [ebp-08]    #mul var1 to var 2
:0040103A 3B45FC                  cmp eax, dword ptr [ebp-04]     #compare mul result to var 1
:0040103D 90                      nop
                 :0040103E 90                      nop

* Possible StringData Ref from Data Obj ->"success"
                                  |
:0040103F 6830604000              push 00406030                   #push pointer to string
:00401044 E823000000              call 0040106C                   #call printf
:00401049 83C404                  add esp, 00000004                          #drop result from stack
:0040104C EB0D                    jmp 0040105B                    #jump to 0x0040105B

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0040103D(C)
* Possible StringData Ref from Data Obj ->"failure"
                                  |
:0040104E 683C604000              push 0040603C                   #push pointer to string
:00401053 E814000000              call 0040106C                   #call printf
:00401058 83C404                  add esp, 00000004                          #drop result from stack

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0040104C(U)
* Possible StringData Ref from Data Obj ->"finished"
                                  |
:0040105B 6848604000              push 00406048                   #push pointer to string
:00401060 E807000000              call 0040106C                   #call printf
:00401065 83C404                  add esp, 00000004                          #drop result from stack
:00401068 8BE5                    mov esp, ebp                    #tidy up
:0040106A 5D                      pop ebp
:0040106B C3                      ret
```



Things to Try

- An RE run yourself
- Confusing RE tools
  - odd offset jumping
  - stepper triggers and tripwires
  - who runs the process, who owns it

Word of Warning

- Don’t self destruct - request clarification
- Don’t bomb - why destroy someone else’s work
- Don’t assume
- Don’t ship what you don’t want used
- Check for stack busting

Q&A