XFOCUS TEAM
from internet,for internet

Binary Comparison of Security Patch

Hume@nsfocus.com

X'CON 2004

OVERVIEW

- **Comparison of patch – A common method to diclose what's hidden in patch**
  - **è** Who need to compare security patch
  - **è** Open source Vs. close source software
  - **è** The difficulty of binary comparison
- **Some comparison methods and their defects as to security patch**
  - **è** Simple byte-to-byte comparison
  - **è** Disassembly – >comapring as text
  - **è** Other methods

- **Ø The recent methods**
  - **è** graph isomorphisms based on instruction similarity(Todd Sabin@razor)
  - **è** Structual comaparing(halvar flake)
  - **è** False negatives of these method
- **Ø Understanding program and peculiarity of binary comparison**
  - **è** Function – >instructions
  - **è** peculiarity of binary comparison
- **Ø Comparing security patch**
  - **è** Structual comparision、semantic-sensitive analysis

- è Design function signature
- è Filtering(WI)
- è Generate graphs and use graph

Ø Some patch comarision examples

- è Microsoft Windows schannel.dll PCT1 Buffer Overflow
- è MS04-11 DsRolepDebugDumpRoutine Buffer Overflow

Comparison of patch – A common method to diclose what's hidden in patch

- **Who need to compare security patch**
  - Security defence，vulneralbility analysis，virus variants analysis
  - Vendor who utilized the undocumented characteristics
  - The hacker^_^
- **Open source Vs. close source software**
  - Open source><source comparison is simple

- x close source patch><much work to do

v The difficulty of binary comparison

- x Current security patch often patch several vuls in the same binary
- x Hard to recognize the unrelated changing
- x Compiler's optimizations

è Modifying,compiling src< – >reverse engineer – >information asymetry

à The traditional comparison methods can hardly release us from boring "eye-diffing"

Some comparison methods and their defects

- Simple byte-to-byte comparison。FC,etc can only be used when only several bytes changed
- Disassembly – >comapring as text。Beyond compare,vi,emacs…can't understand program logic,only apply to samll executables and few functions were changed
- Other methods。Regular expression?

Recent Methods

- graph isomorphisms based on instruction similarity。

  Todd Sabin：《Comparing binaries with graph isomorphisms》

  - Every function – every instruction is a node of a graph – reduce graph – merge graph – human recognizing

Recent Methods

v Structual comaparison

Halvar Flake：《Structural Comparison of Executable Objects》

- **Structural function signatures(logic blocks,subcalls,links)matching – generate call trees for those can't match and those not one-to-one accurate match – get the result**

Recent Methods

- **Advantages of them**
  - Structural function signatures are hardware independent，easy to port
  - Structural function signatures are less possibly affected by compiler optimizations
  - graph isomorphisms based on instruction similarity won't omit the non-structural changing（though not many of them）
  - Graph is rather straight-forward to human

Recent Methods

- **Some disadvantages of these method**
  - à Structural function signatures may omit some non structural-changeing
  - à There maybe more functions have same sig that can't match by calltree for Structural function signatures
  - à Once parent function inaccurately matched,may produce more false matched functions
  - à Instruction similarity suffer more on compiler optimizations
  - à Merging graphs sometimes not complete

Understanding Programs

- Programs are consisted of instruction sequence
  instrunction ： Opcode[act] Operand[object]
- Function is the basic logic unit
- software engineering： Separation of iterface and implemetations
- Incremental link

Pecularity of Security Patch

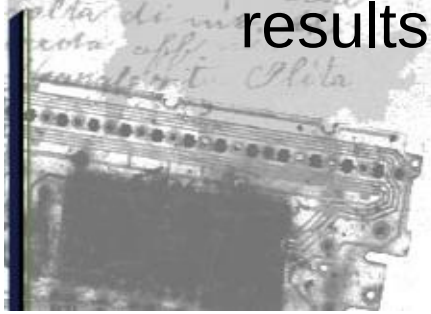
- Two binary are similar,i.e,the changed functions are less than 20%
- Usually they are compiled by same compiler of compiler of the same series
- Most binay codes are the same,but plentity relocations in operand would change
- Compiler optimizations

→ Our aim：to find the samantic changes

Some ideal methods

- Shield lowest level binary differences – >Decompile to uniform HLL or IML
  - à too many compilers,not mutual tech. availble
- Directed graph comparison
  - à Directed graph comparison – NP?
- Others?

Ú **Reality**：speed and complexities compromise – simple methods,less complexities to generate usable results



Comparing security patch

ü Structual comparision steps(not new)

è Take exe obj as a “graph”

è Take function as the basic semantic unit – “subgraphs”

è Finding the starting point(interface etc…)

è Begin comparision

è Structual match the diff functions and mark the relations of them

Comparing security patch

ü Design function signatures

è Platform independent signatures

† blocks – subcalls – links

† blocks – subcalls – links – instructions/stacksize

† blocks – subcalls – links – other intended sigs

È design your own signatures

Þ Platform independent sigs are easy to port,simple rules can eliminate branch optimizations(jz/jnz/jmp) à It's not "accurate" in essence,some function pairs must be deduced from structual analysis or dataflow analysis

Comparing security patch

- **Design function signatures(continued)**
  - **è** Platform dependent signatures
    - † IDA Flirt signatures
    - † Instruction sequence sensitive signatures
    - † Instruction sequence In-sensitive signatures
    - † (Instr – operand)type signatures
    - È design your own expected signatures,eliminate the affect of relocations

Comparing security patch

ü Design function signatures(continued)

Þ Platform dependent signatures can be more “accurate”,proper designed signatures can deal with register exchange optimizations.

Þmore accurate match than Platform dependent signatures(more lossely),sometimes can avoid the situation that subsequent mismatching caused by parent mismatching

à Not so easy to port,more difficult to deal with branch optimizations

Comparing security patch

- **Filtering of results(WI – weak intelligence)**
  - à Every signature has it's advantages and disadvantages
  - à Analysis of the difference produced by different methods often yields more accurate intended result
  - **è** Combine them together would help!
  - **è** Human analysis would benefit from the ability to check emphasis intended filtering results.Need a database?Yes In fact it is.

Comparing security patch

ü **Graphical comparing – how to generate graph and view them**

- Þ It's hard to generate abstract expressionism,but easy to generate a flow graph
- Þ It's alittle difficult to display directed graph,but many open src or free tools available:
  Win32graph AiSee
- Þ It's more easy for Human to recognize the difference of two colored flowchart than just the assembly!

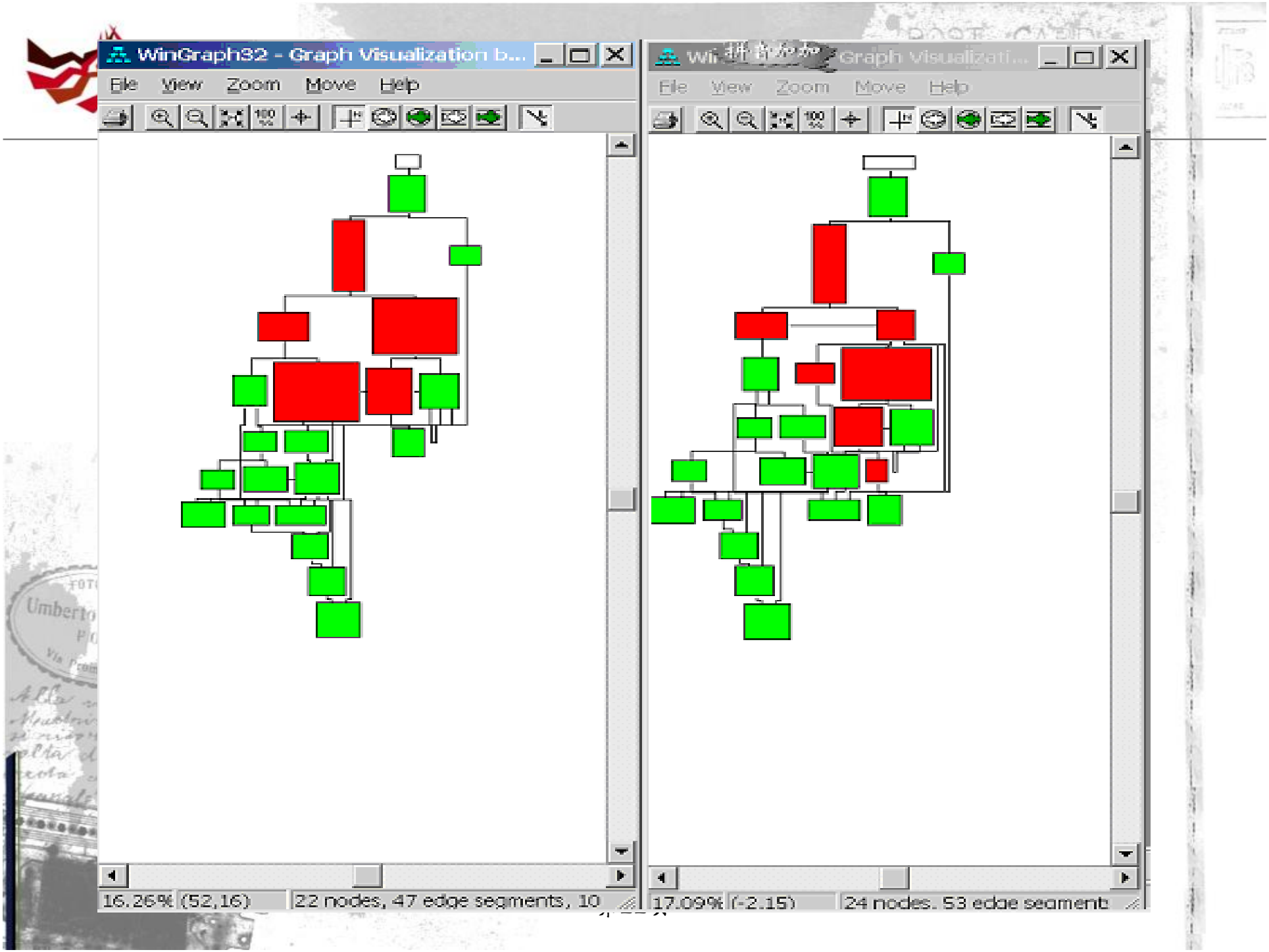
è Generate flowchart and color it

Examples

- Microsoft Windows schannel.dll PCT1 **buffer overflow**
- Analysis of patched and orginal schannel.DLLproduce about 20 functions tchanged,one of them is the function :_Pct1SrvHandleUniHello

```
.text:766AE2BD          mov     [ebp+8], eax
.text:766AE2C0          mov     eax, [edx+0Ch]
.text:766AE2C3          lea     ebx, [eax+eax]
.text:766AE2C6          cmp     ebx, 20h
.text:766AE2C9          jbe     short loc_766AE2D2
```

- Futher analysis disclose that it's a bufov



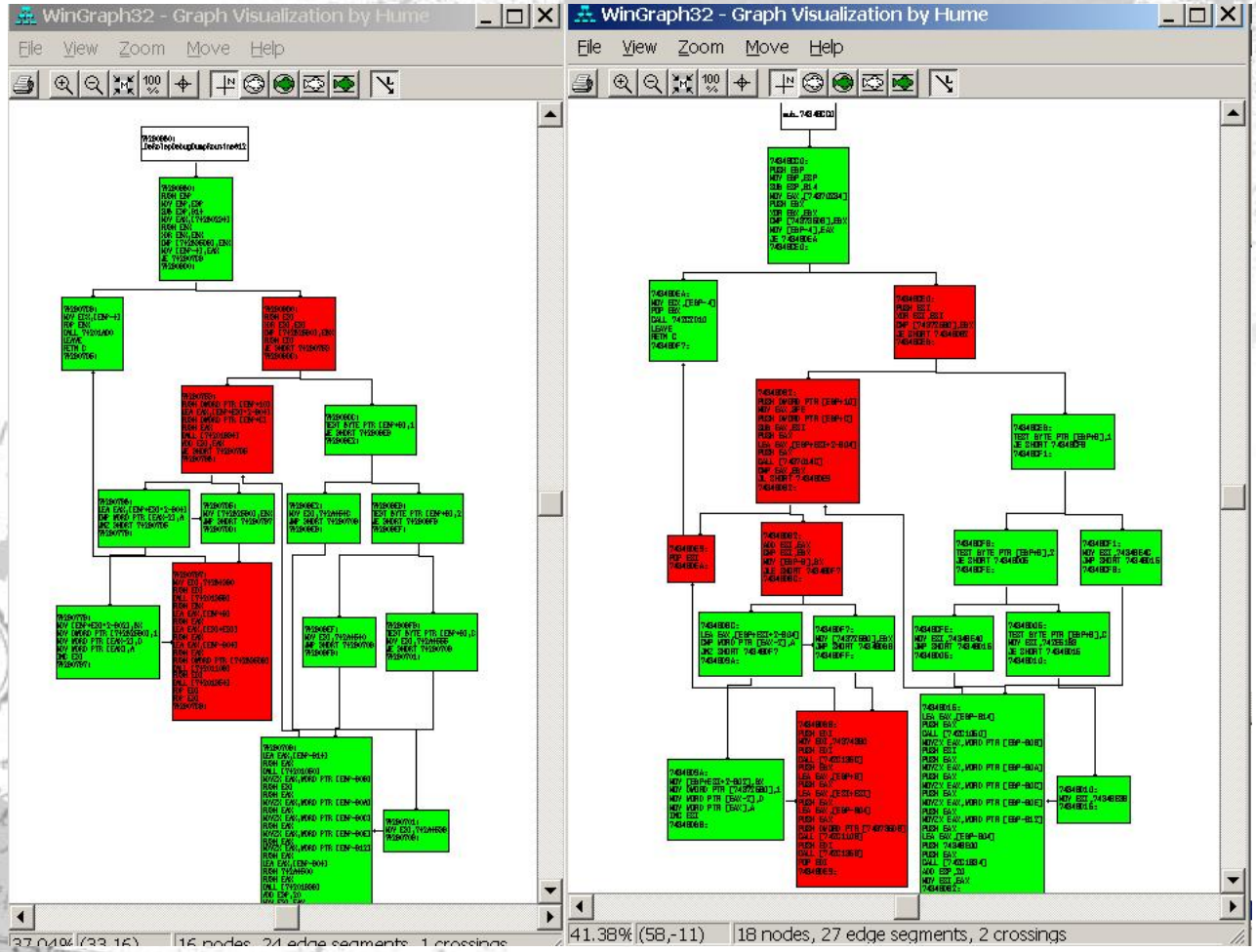
Examples

- MS04-11 LSASRV.DLL comparing p/np version:
  - totally about 20 functions,some are:

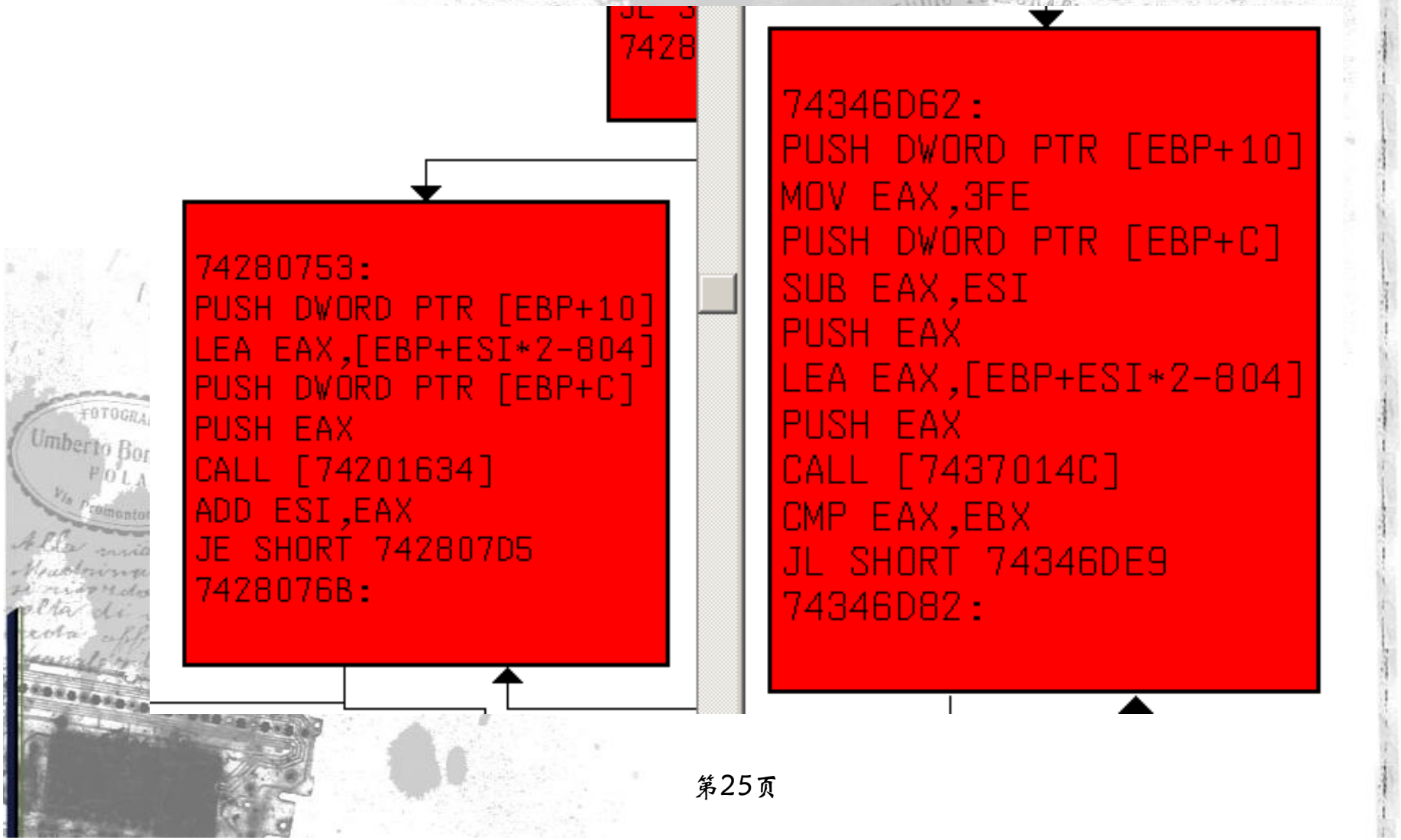
?NegpCrackRequest - sub_742DBEB0
?NegpDetermineTokenPackage - sub_742FB2E0
?SetFlags - sub_74319CF0
_LsapDbOpenTrustedDomainByName - sub_74321A80
_DsRolepDebugDumpRoutine - sub_74346CC0

  - two of the functions fixed two holes,one of them was exploited by sessar worm.

_DsRolepDebugDumpRoutine - sub_74346CC0



_DsRolepDebugDumpRoutine - sub_74346CC0



_DsRolepDebugDumpRoutine - sub_74346CC0

```
add     esp, 20h
mov     esi, eax

                        ; CODE XREF: D
push    [ebp+arg_8]
lea     eax, [ebp+esi*2+var_804]
push    [ebp+arg_4]
push    eax
call    ds:__imp__wvsprintfW@12 ; __de
add     esi, eax
jz      short loc_742807D5
lea     eax, [ebp+esi*2+var_804]
```

```
                                ; CODE X
push    [ebp+psz1]              ; 参数1
mov     eax, 3FEh               ; 3FE==1
push    [ebp+pszfmt]            ; pszfmt
sub     eax, esi
push    eax                     ; cchLim
lea     eax, [ebp+esi*2+Buffer]
push    eax                     ; lpout
call    wvnsprintfW
cmp     eax, ebx                ; EAX==
jl      short loc_74346DE9
add     esi, eax
```

谢谢！
Thanks !

oThx halvar flake and Todd Sabin for their sharing

oThx my company nsfocus and my workmates!

oThx my xfocus friends.

oAlso those who helped me a lot!

XFOCUS TEAM
from internet,for internet

Any Questions?

X'CON 2004