Finding 0-days by Using VulnHunt

Wei Wang(alert7)
Code Audit Labs of Vulnhunt Co.,Ltd

翰海源

翰海源 www.vulnhunt.com

Agenda

- About author
- Vulnhunt Internal
  - Ring3 VM
  - Tainted Propagate Analysis
  - Modules of checking vulnerability
- How to find pdf 0day(now fixed) with Vulnhunt
  - Normal steps
  - To check
  - Locate and Analysis
- How to find alzip 0day withvulnhunt (Demo)

About author

- Aka Alert7 ,XFOCUS member
- Find lot of vulnerabilities in Microsoft, ORACLE , ADOBE , QUICKTIME , LINUX KERNEL , Trend Micro , CA…
- With other XFOCUS guys wrote a book 《Network Penetration Testing》
- He was McAfee Labs Security Researcher
- Now he is CTO of Vulnhunt co.,Ltd www.vulnhunt.com

Data Stream Analysis

- Why Data stream Analysis
  - To identify exploitable vulnerabilities which is triggered by specific user data stream
- Type of Data stream analyze
  - Static data stream analyze
    - Coverity
    - Fortify
    - Klocwork
  - Dynamic data stream analyze
    - Valgrind (for linux)
    - Immunity Debugger
    - PEIMEI
    - Vtrace

What's vulnhunt?

- Abbreviation of Vulnerability hunt
- Two technology
  - RING3 VM
  - Tainted Propagate Analysis

Virtualization Technology

- Data Control
  - Where does user data stream come from
  - and go to?
- Execution Control
  - Instruction execution path Prediction
  - Flow Control

RING3VM

- Normal VM
  - Cpu,FPU,MEMORY
  - IO
    - Acpi,network,cdrom,DMA,pic,BIOS,Harddisk…
  - Load OS, app run in OS
- RING3 VM
  - CPU,FPU
  - App run in RING3VM or RING3VM step execute app.

RING3VM main functions

- Framework for Instruction-level Tracing
  - Instruction-level Taint Propagation
    - Trace tainted data: from and to.
      - example

    mov ecx, dword ptr ds:[eax]

    we need check if eax is tainted,check if [eax] is tainted, mark ecx as tainted.

  - Instruction-level Other functions(TODO)

RING3VM

- Pros
  - Simplified design
  - Better performance
- Cons
  - Tampered by malicious code from target
  - OS Dependency
  - Bad for muti-thread

Tainted Source Marking

- Function Calls

  call dword ptr ds:0x40606c

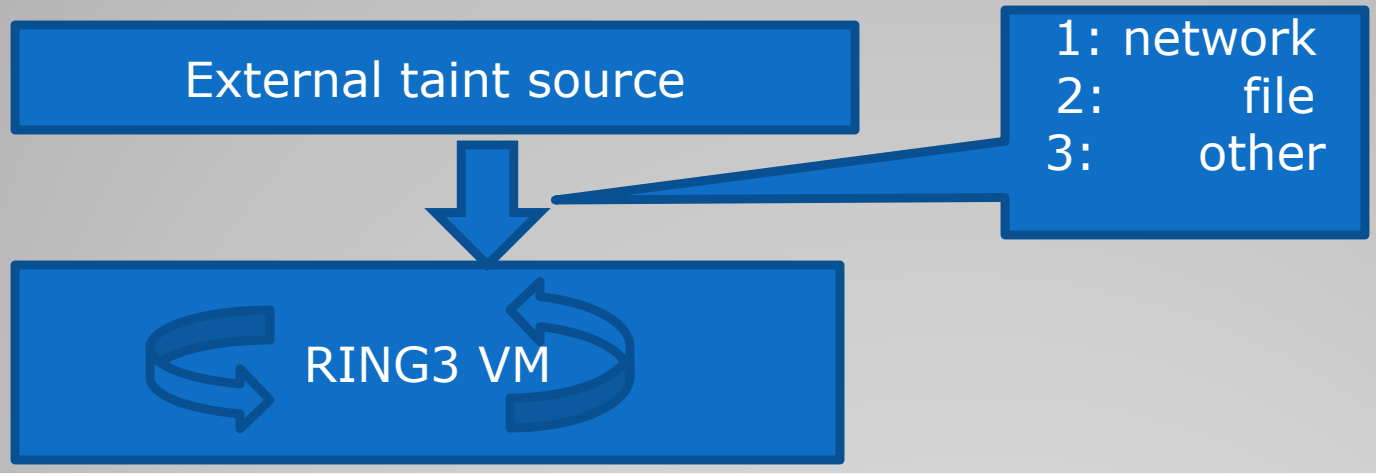
  The data of Address 0x40606c is 0x7c810e17

  Which function call we need Identify address 0x7c810e17 for? (loop every module to check export table)

- Mark taint source

  recv(SOCKET s,char *buf,int len,int flags)

  set_buff_tainted(GET_ARG(a->esp,1),GET_ARG(a->esp,2));



Two level API

- Function-Level API
  - API-- Taint Tracking
  - API–- HOOK Functions
  - API– alert,log
- Instruction-level API
  - Instruction-level check

Taint Tracking API

- Hook a function

1:
```
FUNC_INFO f;
        f.comment = NULL;
                f.enter_func= false;
                f.func_name="malloc";
                f.handler = malloc_handler;
                regist_func_handler(&f);
```
2:
```
static int malloc_handler(void  * p,unsigned int eip,struct allregs * a)
{
FUNC_INFO * info = (FUNC_INFO *) p;
    checker_log(CHECK_MODULE"%s_handler called at %p %s(0x%x)\n",
                info->func_name,
                eip,
                info->func_name,
                GET_ARG(a->esp,0));
    bool t = buffer_from_tainted(GET_ARG_OFFSET(a->esp,0));
    if ( t )
                checker_log(CHECK_MODULE SECURITY_WARNING"%s(0x%x) arg0 is %s",
                                                        info->func_name,
                                                        GET_ARG(a->esp,0),
                                                        "[tainted]");
    return 0;
}
```

Taint Tracking API

- Mark tainted source

```
static int recv_handler(void * p,unsigned int eip,struct allregs * a)
{
    set_buff_tainted(GET_ARG(a->esp,1),GET_ARG(a->esp,2));
    return 0;

}
```

Taint Tracking API

- Goal : let behind false positives, discover as many tainted check point as possible
  - 100 to 10 ? Not bad!
- Thoughts about reducing false+
  - Future work.

Embedded hundreds of checker and fucntion Abstract

- Memcpy
- memcpy_s
- vsprintf_l
- Syslog
- Printf
- Open
- Fread
- Read
- ReadFile
- …

Case Study -- MALLOC MEMORY

- example
  - Len=get_int_from_net();
  - Len+=1;
  - Char *buf = malloc(Len);
- Malloc, RtlAllocateHeap, HeapAlloc
  - buffer_from_tainted(GET_ARG_OFFSET(a->esp,x));
  - Check if length param is tainted, alert for manual review

Case Study -Format string

- example
  - printf(get_buf_from_net());
- Printf,fprintf,sprintf…
  - If format string param is tainted, alert as potential format string vulnerability
  - Printf(“%s%n”),it’s a real format string vulnerability, but due to the param is not tainted, so it’s a unexploitable . So ignore.

Case Study - rep movs

- example
  - Memcpy(dst,src,get_int_from_net());
- Alert if ECX register is tainted

Case Study - signed integer

- Example
  - Int a;
  - char *buf = Get_buf_from_net();
  - a = buf[0]; // signed extend
- movsx ecx, byte ptr ss:[esp+0x4]
  - Check movsx, alert if op_2 is tainted.

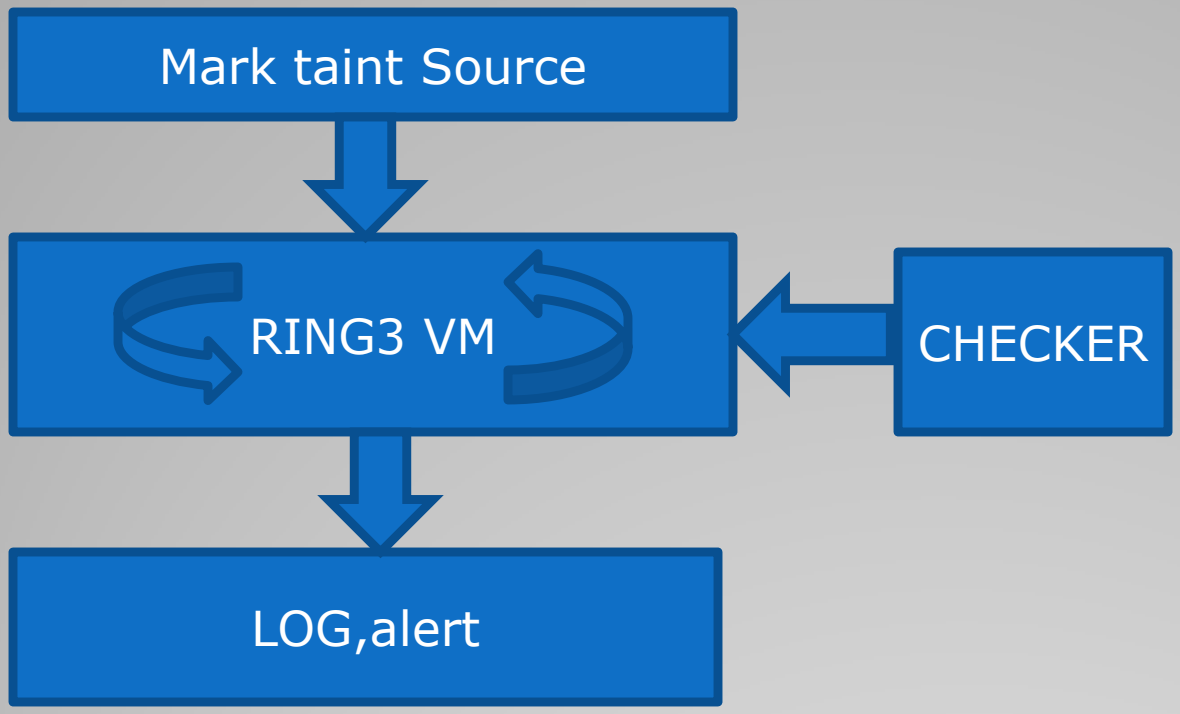
Case Study – DIV ZERO

- Example
  - Int a;
  - a/=get_int_from_net();
- Idiv/div eax, ecx
- Check idiv/div, alert if op_2 is tainted

Case Study – Integer Overflow

- example
  - Int a;
  - a*=get_int_from_net();
- Imul/mul edi, [ebp+arg_8]
  - Alert if one of op1,op2 is tainted
- lea ecx, dword ptr ds:[eax*8]
  - Sometime,using lea express multiplication

Components



Perf Optimization

- Rep movsd dword ptr es:[edi], dword ptr ds:[esi]
  - Exece once , eq memcpy(edi,esi,ecx*4)
- Skip irrelevant instructions
  - I.e: Skip instructions in ntdll.dll user32.dll

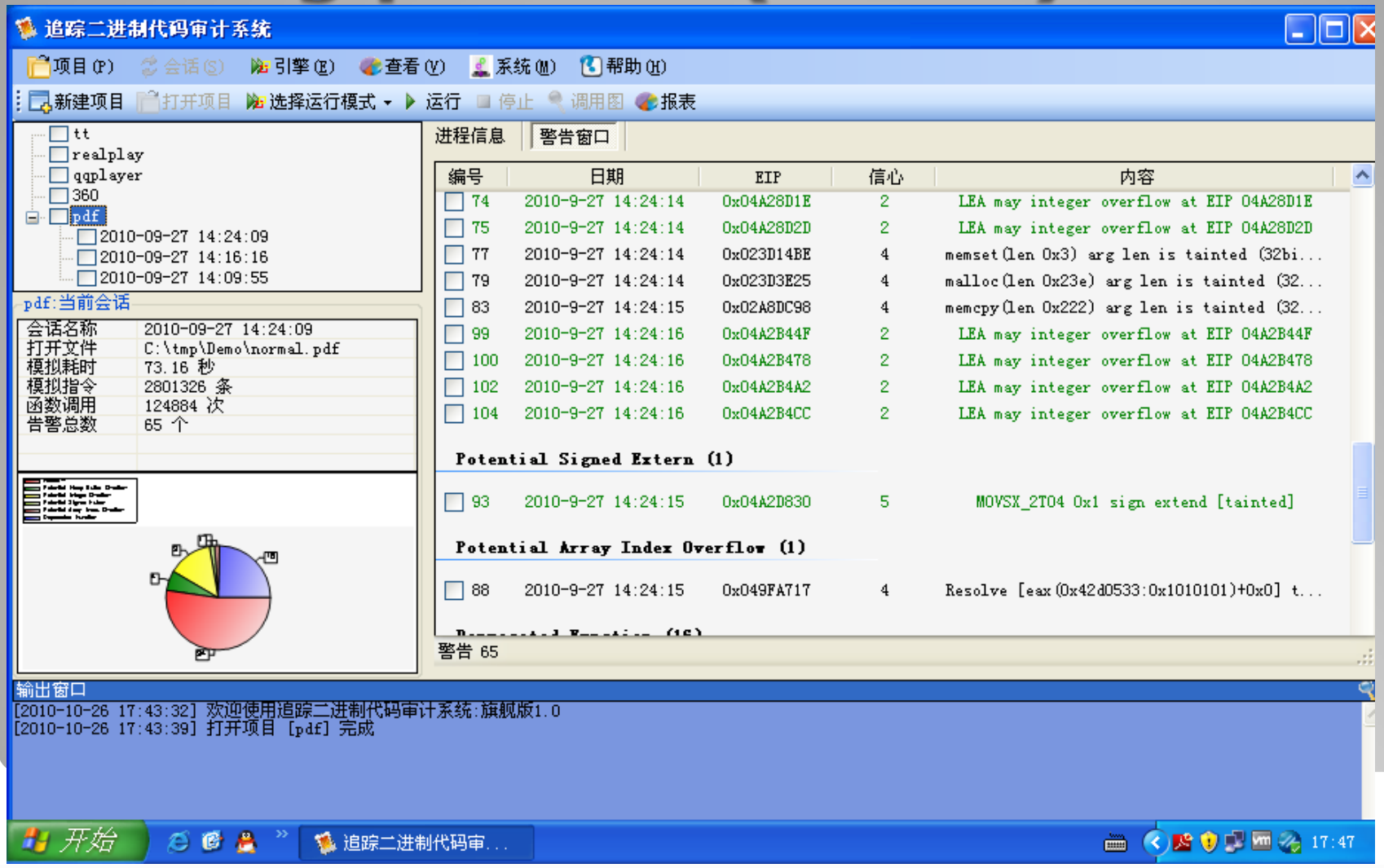
Finding pdf vuln(CVE-2009-3955)

- One example, all (Potential)vulnerabilities of this execution path
- Show how to use vulnhunt find CVE-2009-3955 Adobe Reader and Acrobat JpxDecode Memory Corruption Vulnerability。

Finding pdf vuln(configure)

- Our configure, AcroRd32.exe open a normal example normal.pdf , run mode INS_HOOK.

Finding pdf vuln(result)



Finding pdf vuln(checking)

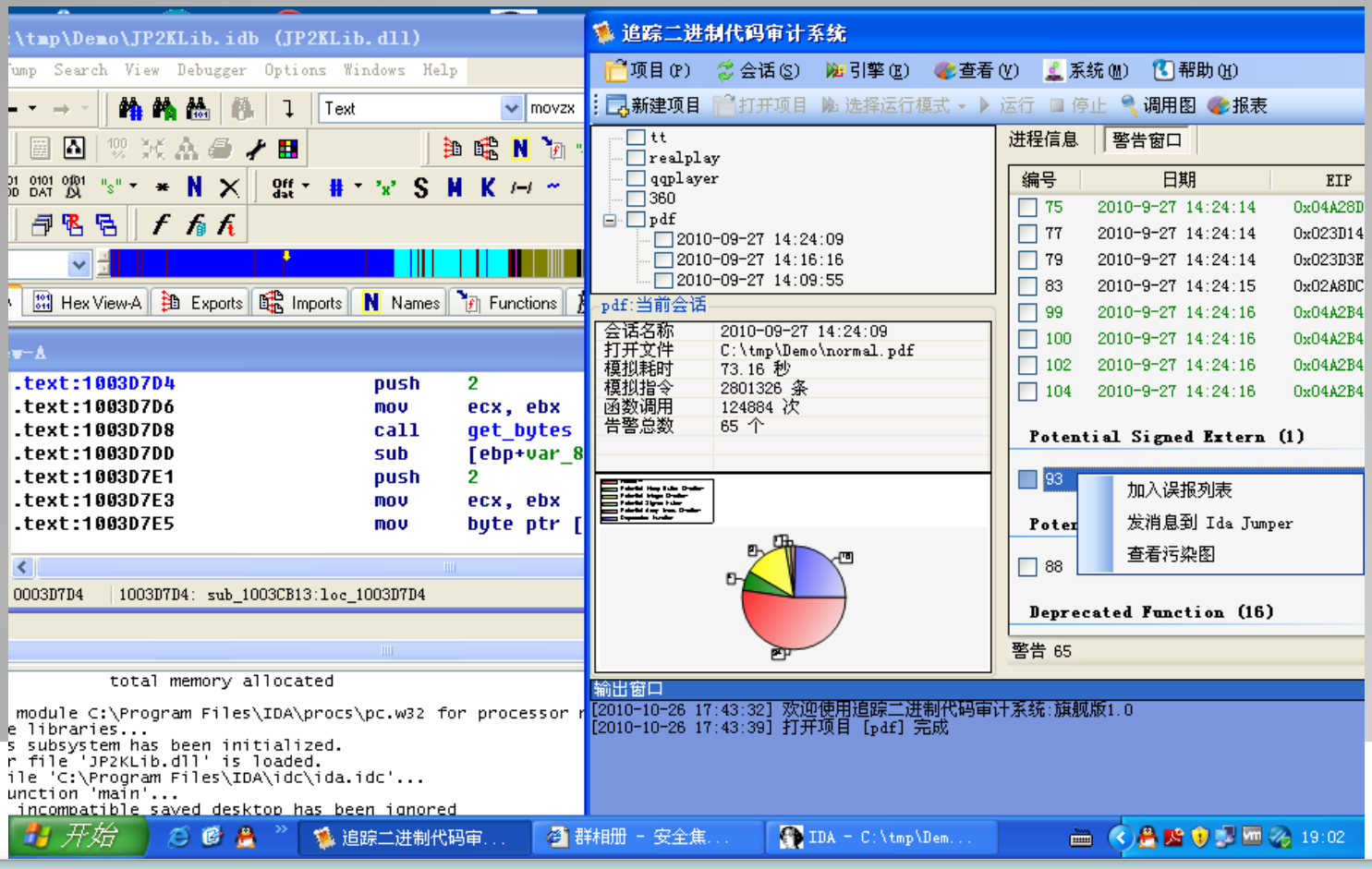
- Currently , output many alerts , need manual checking.
- Goal : let behind false positives, discover as many tainted check point as possible
  - 100 to 10 ? Not bad!

Finding pdf vuln(Analysis 1)

- **confirm Potential signed extern**
  - Checking No93 ,type is Potential signed extern, content is MOVSX_2TO4 0x1 sign extern.

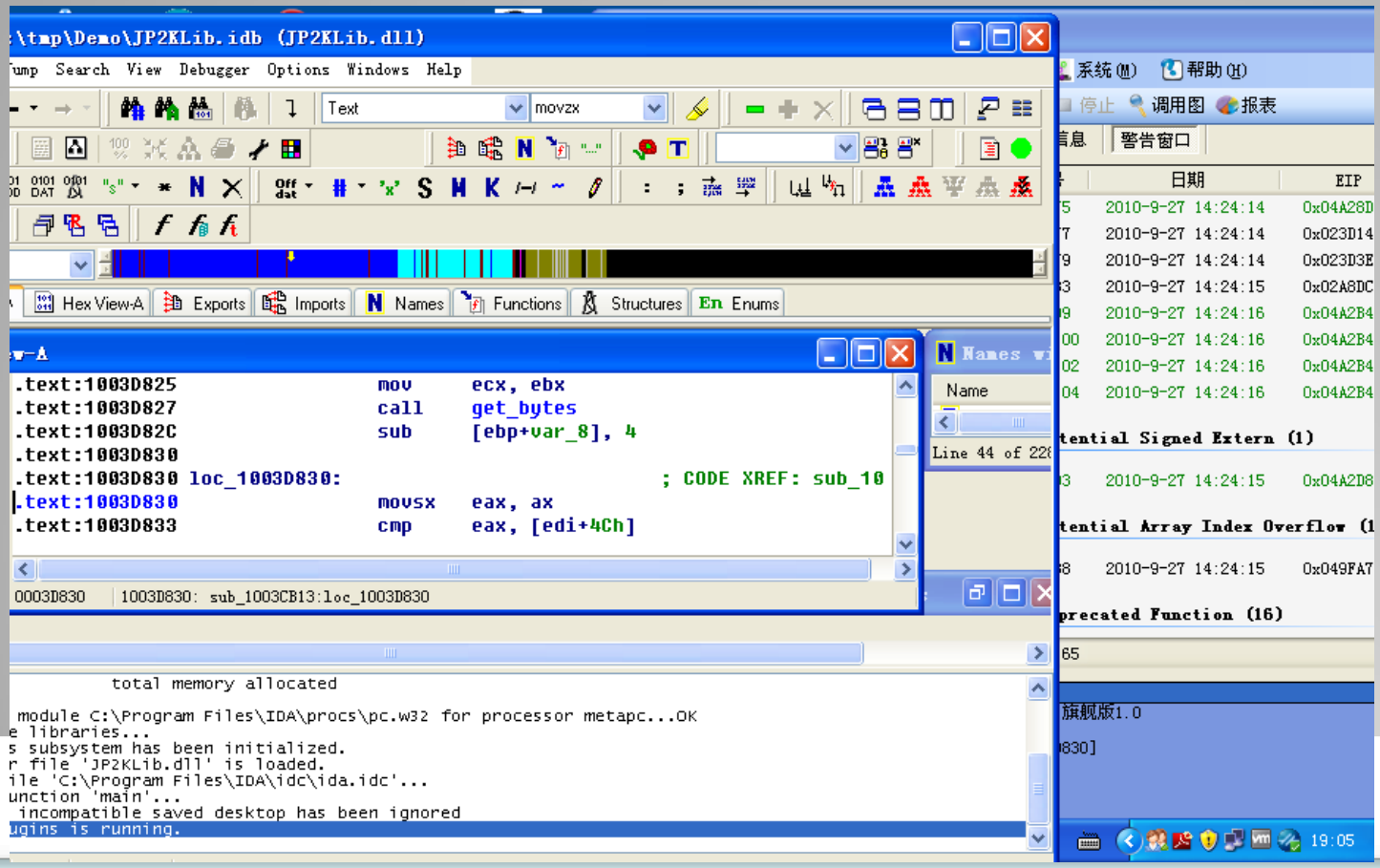
Finding pdf vuln(Analysis 2)

- ALT_5 run Ida jumper, Send to Ida jumper



Finding pdf vuln(Analysis 3)

- Auto locate to Ida ASM from alert



Finding pdf vuln(Analysis 4)

- .text:1003D830 loc_1003D830: ; CODE XREF: sub_1003CB13+D04j
- .text:1003D830 movsx eax, ax
- .text:1003D833 cmp eax, [edi+4Ch]
- .text:1003D836 jge loc_1003DC8A
- .text:1003D83C mov ecx, [edi+2Ch]
- .text:1003D83F mov esi, eax
- .text:1003D841 imul esi, 0Ch
- .text:1003D844 mov [esi+ecx], eax
- .text:1003D847 mov eax, [edi+2Ch]
- Ax is tainted , mosx extern to eax,so eax can be negative. bypass jpe checking.
- mov [esi+ecx], eax , can put eax value to limited arbitrary address. Cause memory Corruption.

Finding alzip 0day(Demo)

THANK YOU

Question?