2018补天白帽大会暨补天五周年盛典
HACK FOR SECURITY

补 天
漏洞响应平台

Dynamic Analysis Skills of Reverse Engineering

Atum

About me

- Atum
  - CTF Player @Eur3kA @blue-lotus @Tea Deliverers
  - Master candidate @ICST SECLAB, Peking University
  - http://atum.li lgcpku@gmail.com
- Software Security
  - Focus on Vulnerability Discovering & Exploiting & Defense
  - Active in CTF, PWN/Reverse

Static Analysis VS Dynamic Analysis

- Static Analysis
  - A fundamental skill
  - Works well on small binary
  - Open IDA pro, audit->eat->sleep->audit
- Dynamic Analysis
  - A tricky skill
  - Useful when reversing large binary
  - A complementary technique to static analysis
  - Various tools available, run->audit->guessing

Dynamic Analysis techniques

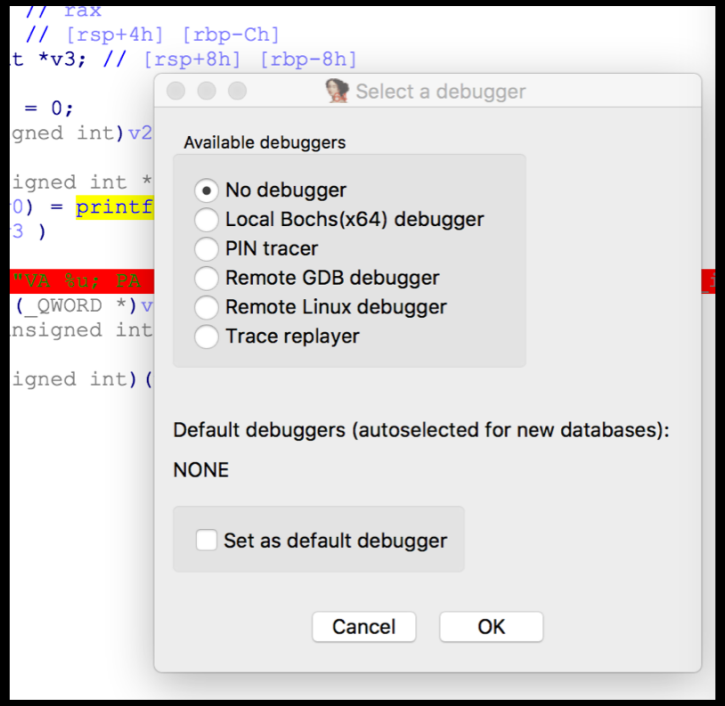
- Debugging
- Tracing
- Symbolic Execution
- Dynamic Taint Analysis

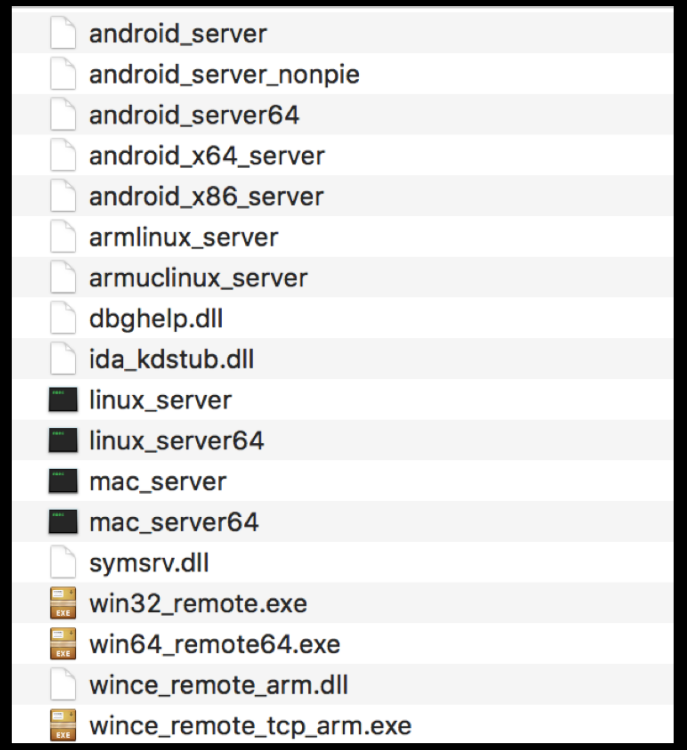
Debugging

- The most popular dynamic analysis skill
- Some tricks:
  - Source Level Debugging without source code
  - Hardware breakpoint
  - Record & Replay (Reverse Debugging)

Source Level Debugging without source code

- IDA hex-rays debugger
  - Stepping F5 code
  - Debugging while Auditing
  - Full-platform support
  - Characteristic UI





Record & Replay

- gdb reverse step/continue
  - Use record command to record execution trace
  - Use reverse-* command to reverse debugging
- Example: QWBCTF hide
- This technique is also supported by ollydbg

```
   0x400a08 <frame_dummy+40> jmp    0x400960 <register_tm_clones>
   0x400a0d <main+0>         push   rbp
   0x400a0e <main+1>         mov    rbp, rsp
-> 0x400a11 <main+4>         lea    rsp, [rsp-0x1020]
   0x400a19 <main+12>        or     QWORD PTR [rsp], 0x0
   0x400a1e <main+17>        lea    rsp, [rsp+0x1020]
   0x400a26 <main+25>        lea    rsi, [rip+0x2b69b3]        # 0x6b7
   0x400a2d <main+32>        lea    rdi, [rip+0x8c750]        # 0x48d1
   0x400a34 <main+39>        mov    eax, 0x0
------------------------------------------------------------------------
[#0] Id 1, Name: "readme_revenge", stopped, reason: BREAKPOINT
------------------------------------------------------------------------
[#0] 0x400a11->Name: main()
------------------------------------------------------------------------
gef> record full
gef> reverse-
reverse-continue  reverse-next      reverse-search    reverse-stepi
reverse-finish    reverse-nexti     reverse-step
gef> reverse-step
```

Record & Replay

- VEX instruction set is not supported by gdb record
- rr project
  - http://rr-project.org
  - a replacement of gdb record
  - not a living debugging

```
->0x555555554a95 <main+4>          sub    rsp, 0x20
   0x555555554a99 <main+8>          mov    DWORD PTR [rbp-0x14], edi
   0x555555554a9c <main+11>         mov    QWORD PTR [rbp-0x20], rsi
   0x555555554aa0 <main+15>         mov    eax, 0x0
   0x555555554aa5 <main+20>         call   0x555555554993 <menu>
   0x555555554aaa <main+25>         mov    eax, 0x0
------------------------------------------------------------------------------
[#0] Id 1, Name: "300", stopped, reason: BREAKPOINT
------------------------------------------------------------------------------
[#0] 0x555555554a95->Name: main()
------------------------------------------------------------------------------
gef> record full
gef> c
Continuing.
Process record does not support instruction 0xc5 at address 0x7ffff7b738c5.
Process record: failed to record execution log.

atum@AtumNUC:~/Research/intelpt/test/300$ rr record ./300
rr: Saving execution to trace directory `/home/atum/.local/share/rr/300-1'.
1) alloc
2) write
3) print
4) free
^C
atum@AtumNUC:~/Research/intelpt/test/300$ rr replay
```

Tracing

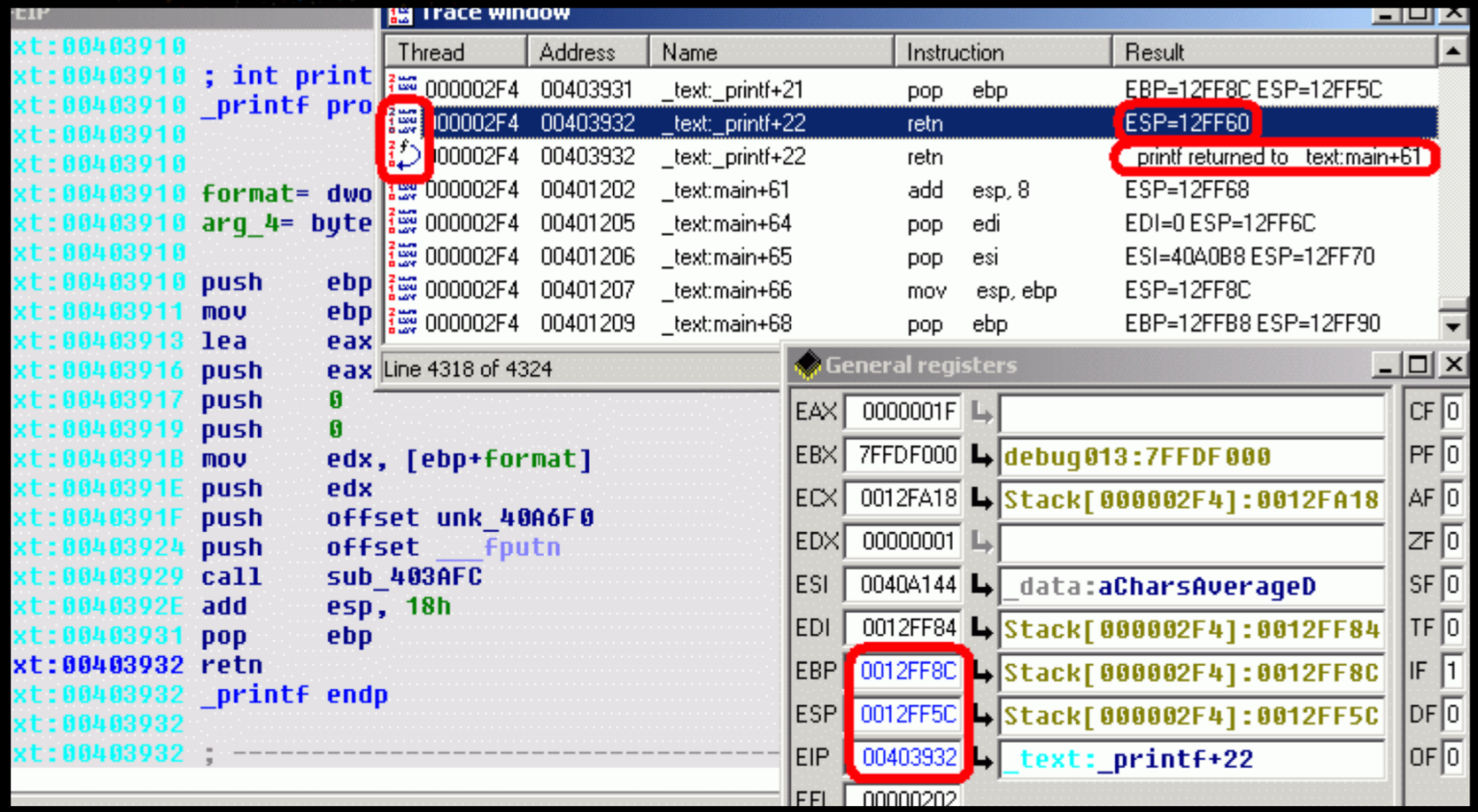
- Records program execution, one of most useful techniques
- Trace Levels
  - Instruction tracing / Basic block tracing / Function tracing / API tracing etc.
- Trace Techniques
  - Debugger based tracing (easy to use)
  - Injected based tracing (use on anti-dbg program)
  - Hardware based tracing (trace kernel code, low overhead)

Application Scenarios Examples

- Find hidden code
  - 家徒四壁 @ HITCON CTF / hide @ QWB CTF
- Find key code when reversing large binary
  - Usually combine with hardware breakpoint, I think
  - API trace to find which function called a socket/recv/read call
- Get control flow information
  - Fuzzing
  - Use hardware trace to trace kernel

Useful tracing tools

- IDA tracer
  - Trace with IDA debugger
- gdb tracer
  - record & replay
- strace
  - Trace system call on Linux
  - Easy to use
  - Strace /path/to/bin



Useful tracing tools

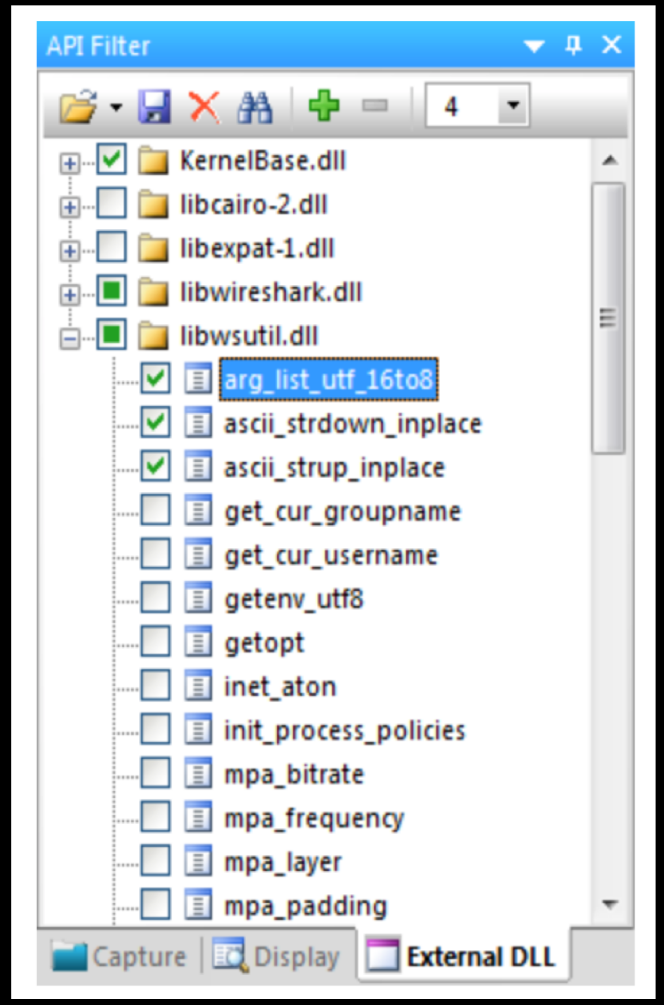
- Pin (injected tracing)
  - Pin is a tool for the instrumentation of programs.
  - https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool

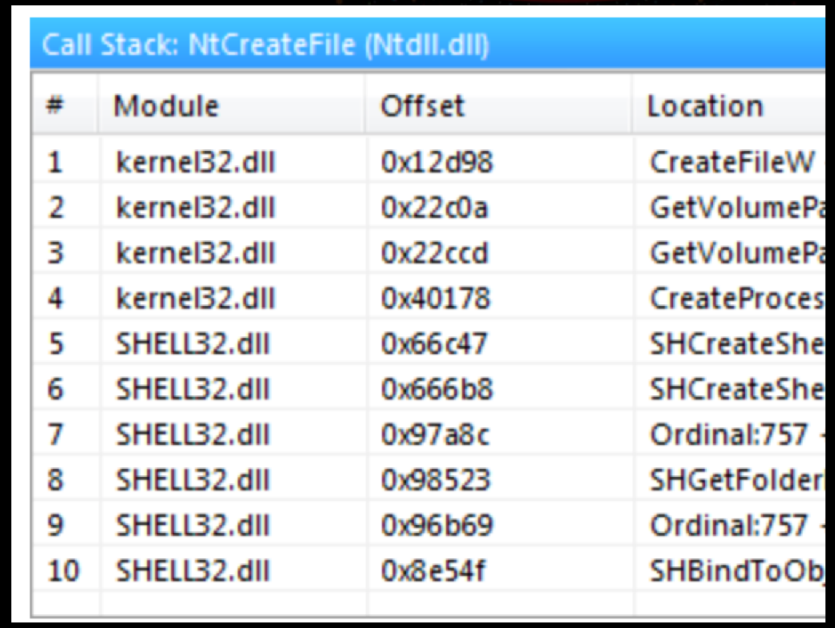
```
int main(int argc, char * argv[])
{
    // Initialize pin
    if (PIN_Init(argc, argv)) return Usage();

    OutFile.open(KnobOutputFile.Value().c_str());

    // Register Instruction to be called to instrument instructions
    INS_AddInstrumentFunction(Instruction, 0);

    // Register Fini to be called when the application exits
    PIN_AddFiniFunction(Fini, 0);

    // Start the program, never returns
    PIN_StartProgram();

    return 0;
}
```

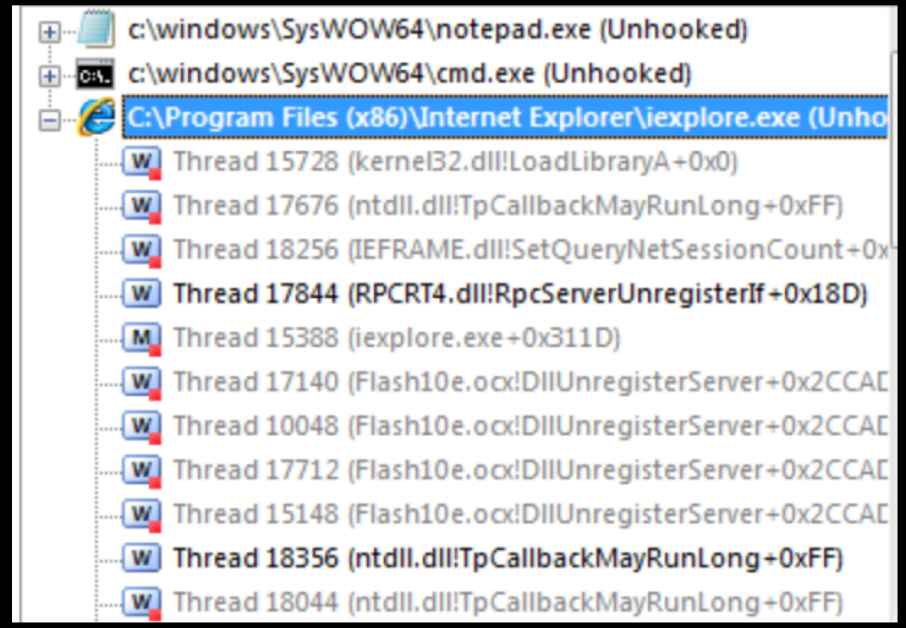
Useful tracing tools

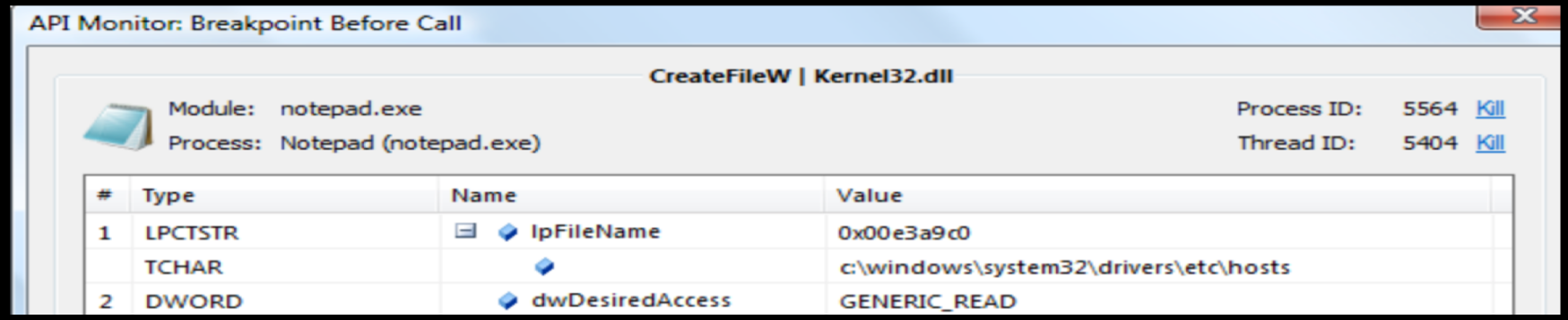
- API monitor (injected tracing)
  - Monitor and control API calls made by applications and services on windows
  - An easy-to-use Filter
  - http://www.rohitab.com/apimonitor#Download
- Traceview
  - API tracing for android application
  - https://developer.android.com/studio/profile/traceview.html

API monitor


API Filter
4
KernelBase.dll
libcairo-2.dll
libexpat-1.dll
libwireshark.dll
libwsutil.dll
arg_list_utf_16to8
ascii_strdown_inplace
ascii_strup_inplace
get_cur_groupname
get_cur_username
getenv_utf8
getopt
inet_aton
init_process_policies
mpa_bitrate
mpa_frequency
mpa_layer
mpa_padding
Capture
Display
External DLL


Call Stack: NtCreateFile (Ntdll.dll)
Module Offset Location
1 kernel32.dll 0x12d98 CreateFileW
2 kernel32.dll 0x22c0a GetVolumePa
3 kernel32.dll 0x22ccd GetVolumePa
4 kernel32.dll 0x40178 CreateProces
5 SHELL32.dll 0x66c47 SHCreateShe
6 SHELL32.dll 0x666b8 SHCreateShe
7 SHELL32.dll 0x97a8c Ordinal:757
8 SHELL32.dll 0x98523 SHGetFolder
9 SHELL32.dll 0x96b69 Ordinal:757
10 SHELL32.dll 0x8e54f SHBindToOb


c:\windows\SysWOW64\notepad.exe (Unhooked)
c:\windows\SysWOW64\cmd.exe (Unhooked)
C:\Program Files (x86)\Internet Explorer\iexplore.exe (Unho
Thread 15728 (kernel32.dll!LoadLibraryA+0x0)
Thread 17676 (ntdll.dll!TpCallbackMayRunLong+0xFF)
Thread 18256 (IEFRAME.dll!SetQueryNetSessionCount+0x
Thread 17844 (RPCRT4.dll!RpcServerUnregisterIf+0x18D)
Thread 15388 (iexplore.exe+0x311D)
Thread 17140 (Flash10e.ocx!DllUnregisterServer+0x2CCAD
Thread 10048 (Flash10e.ocx!DllUnregisterServer+0x2CCAD
Thread 17712 (Flash10e.ocx!DllUnregisterServer+0x2CCAD
Thread 15148 (Flash10e.ocx!DllUnregisterServer+0x2CCAD
Thread 18356 (ntdll.dll!TpCallbackMayRunLong+0xFF)
Thread 18044 (ntdll.dll!TpCallbackMayRunLong+0xFF)


API Monitor: Breakpoint Before Call
CreateFileW | Kernel32.dll
Module: notepad.exe
Process: Notepad (notepad.exe)
Process ID: 5564 Kill
Thread ID: 5404 Kill
Type Name Value
1 LPCTSTR lpFileName 0x00e3a9c0
TCHAR c:\windows\system32\drivers\etc\hosts
2 DWORD dwDesiredAccess GENERIC_READ

API monitor

Summary | 18 of 55 calls | 67% dropped | 31 KB used

#	Relative Time	TID	Module	API	Return Value
31	0:00:00:062	6432	MSVBVM60.DLL	CoGetClassObject (Microsoft WinSock Control, version 6.0 <MSWinsock.Winsock.1>,	S_OK
32	0:00:00:062	6432	ole32.dll	LoadLibraryExA ("CLBCatQ.DLL", NULL, 0)	0x76de0000
33	0:00:00:062	6432	ntdll.dll	DllMain (0x76de0000, DLL_PROCESS_ATTACH, NULL)	TRUE
34	0:00:00:062	6432	ole32.dll	LoadLibraryExW ("C:\Windows\SysWow64\MSWINSCK.OCX", NULL, LOAD_WITH_...	0x22170000
35	0:00:00:062	6432	ntdll.dll	DllMain (0x77c70000, DLL_PROCESS_ATTACH, NULL)	TRUE
36	0:00:00:062	6432	ntdll.dll	DllMain (0x75c70000, DLL_PROCESS_ATTACH, NULL)	TRUE
37	0:00:00:062	6432	ntdll.dll	DllMain (0x735e0000, DLL_PROCESS_ATTACH, NULL)	TRUE
38	0:00:00:062	6432	ntdll.dll	DllMain (0x22170000, DLL_PROCESS_ATTACH, NULL)	TRUE
39	0:00:00:062	6432	MSWINSCK.OCX	WSAStartup (257, 0x0018dd68)	0
40	0:00:00:062	6432	WS2_32.dll	LoadLibraryExW ("version.dll", NULL, 0)	0x73610000
41	0:00:00:062	6432	VERSION.dll	LoadLibraryExW ("d:\Desktop\vb6\Project.exe", NULL, LOAD_LIBRARY_A...	0x00400000
42	0:00:00:062	6432	VERSION.dll	LoadLibraryExW ("d:\Desktop\vb6\Project.exe", NULL, LOAD_LIBRARY_A...	0x00400000
43	0:00:00:078	6432	MSWINSCK.OCX	CreateWindowExA (0, "NMNotifyWindowClass", "Notification Window", W..	0x0034051c
44	0:00:00:078	6432	ole32.dll	DllGetClassObject (...)	0
45	0:00:00:078	6432	USP10.dll	LoadLibraryExA ("ADVAPI32.dll", NULL, 0)	0x77310000
46	0:00:00:093	6432	USER32.dll	LoadLibraryExW ("C:\Windows\syswow64\MSCTF.dll", NULL, LOAD_WITH_ALTERED_...	0x75950000
47	0:00:01:060	6432	ntdll.dll	DllMain (0x22170000, DLL_PROCESS_DETACH, 0x00000001)	TRUE
48	0:00:01:060	6432	ntdll.dll	DllMain (0x735e0000, DLL_PROCESS_DETACH, 0x00000001)	TRUE

Traceview

Name	Incl Cpu Tir	Incl Cpu Time	Excl Cpu Time %	Excl Cpu Time	Incl Real Time %	Incl Real Time	Excl Real
16 android.os.Parcel.writeInt (I)V	9.2%	0.562	5.5%	0.335	4.5%	0.558	
Parents							
20 android.os.Parcel.writeValue (Ljava/lan...	53.4%	0.300			53.4%	0.298	
9 android.os.BaseBundle.writeToParcelInn...	15.5%	0.087			15.4%	0.086	
18 android.content.Intent.writeToParcel (L...	14.4%	0.081			14.5%	0.081	
5 android.app.ActivityManagerProxy.start...	9.1%	0.051			9.0%	0.050	
13 android.os.Parcel.writeArrayMapIntern...	4.8%	0.027			4.7%	0.026	
119 android.net.Uri.writeToParcel (Landro...	2.8%	0.016			3.0%	0.017	
Children							
self	59.6%	0.335			59.7%	0.333	
33 android.os.Parcel.nativeWriteInt (JI)V	40.4%	0.227			40.3%	0.225	
17 android.os.BaseBundle.putInt (Ljava/lang/Str...	8.1%	0.498	1.2%	0.072	4.2%	0.518	
18 android.content.Intent.writeToParcel (Landro...	8.1%	0.496	0.5%	0.033	4.0%	0.497	
19 android.util.ArrayMap.put (Ljava/lang/Object...	7.3%	0.447	3.4%	0.210	3.8%	0.468	
20 android.os.Parcel.writeValue (Ljava/lang/Obj...	7.1%	0.437	1.6%	0.098	3.5%	0.434	
21 android.view.IWindow$Stub.onTransact (ILan...	6.9%	0.420	1.1%	0.065	5.4%	0.668	
22 android.app.Activity.isTopOfTask ()Z	6.7%	0.410	0.2%	0.012	5.0%	0.619	

Find:

Hardware tracing

- Intel Last Branch Record
  - store the source and destination addresses related to recently executed branches using corresponding LastBranchToIP and LastBranchFromIP MSRs
  - No additional overhead
  - perf
- Branch Trace Store
  - Similar as LBR, but using either cache-as-RAM or system DRAM
  - 50%+ overhead
  - perf

Hardware tracing

- Intel Processor Tracing
  - Introduced since Broadwell, very active techniques
  - 5% trace overhead (use for fuzz?), 100x decoding overhead
  - gdb/perf/windowspt/simplept
  - perf record -e intel_pt//u ls
  - perf report / perf script

Hardware tracing

```
7fc3def8e0cc __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so) =>     55e233a288ec myputs (/home/atum/Research/intelpt/test/300/300)
55e233a288fd myputs (/home/atum/Research/intelpt/test/300/300) =>     55e233a28748 _init (/home/atum/Research/intelpt/test/300/300)
55e233a28748 _init (/home/atum/Research/intelpt/test/300/300) =>     7fc3def8e0b0 __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so)
7fc3def8e0c2 __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so) =>                0 [unknown] ([unknown])
           0 [unknown] ([unknown]) =>     7fc3def8e0c4 __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so)
7fc3def8e0cc __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so) =>     55e233a28902 myputs (/home/atum/Research/intelpt/test/300/300)
55e233a28904 myputs (/home/atum/Research/intelpt/test/300/300) =>     55e233a289bb menu (/home/atum/Research/intelpt/test/300/300)
55e233a289c2 menu (/home/atum/Research/intelpt/test/300/300) =>     55e233a288c0 myputs (/home/atum/Research/intelpt/test/300/300)
55e233a288d3 myputs (/home/atum/Research/intelpt/test/300/300) =>     55e233a28750 _init (/home/atum/Research/intelpt/test/300/300)
55e233a28750 _init (/home/atum/Research/intelpt/test/300/300) =>     7fc3df0088c0 __strlen_avx2 (/lib/x86_64-linux-gnu/libc-2.26.so)
7fc3df0088cf __strlen_avx2 (/lib/x86_64-linux-gnu/libc-2.26.so) =>     7fc3df0088f0 __strlen_avx2 (/lib/x86_64-linux-gnu/libc-2.26.so)
7fc3df008915 __strlen_avx2 (/lib/x86_64-linux-gnu/libc-2.26.so) =>     55e233a288d8 myputs (/home/atum/Research/intelpt/test/300/300)
55e233a288e7 myputs (/home/atum/Research/intelpt/test/300/300) =>     55e233a28748 _init (/home/atum/Research/intelpt/test/300/300)
55e233a28748 _init (/home/atum/Research/intelpt/test/300/300) =>     7fc3def8e0b0 __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so)
7fc3def8e0c2 __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so) =>                0 [unknown] ([unknown])
           0 [unknown] ([unknown]) =>     7fc3def8e0c4 __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so)
7fc3def8e0cc __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so) =>     55e233a288ec myputs (/home/atum/Research/intelpt/test/300/300)
55e233a288fd myputs (/home/atum/Research/intelpt/test/300/300) =>     55e233a28748 _init (/home/atum/Research/intelpt/test/300/300)
55e233a28748 _init (/home/atum/Research/intelpt/test/300/300) =>     7fc3def8e0b0 __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so)
7fc3def8e0c2 __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so) =>                0 [unknown] ([unknown])
           0 [unknown] ([unknown]) =>     7fc3def8e0c4 __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so)
7fc3def8e0cc __GI___libc_write (/lib/x86_64-linux-gnu/libc-2.26.so) =>     55e233a28902 myputs (/home/atum/Research/intelpt/test/300/300)
55e233a28904 myputs (/home/atum/Research/intelpt/test/300/300) =>     55e233a289c7 menu (/home/atum/Research/intelpt/test/300/300)
55e233a289c9 menu (/home/atum/Research/intelpt/test/300/300) =>     55e233a28aaa main (/home/atum/Research/intelpt/test/300/300)
55e233a28aaf main (/home/atum/Research/intelpt/test/300/300) =>     55e233a28905 read_int (/home/atum/Research/intelpt/test/300/300)
55e233a2893d read_int (/home/atum/Research/intelpt/test/300/300) =>     55e233a28760 _init (/home/atum/Research/intelpt/test/300/300)
55e233a28760 _init (/home/atum/Research/intelpt/test/300/300) =>     7fc3def8e010 read (/lib/x86_64-linux-gnu/libc-2.26.so)
```

Symbolic Execution

- Not very practical yet
- Used to solve boring constraints
  - use_your_ida @LCTF 2017
  - RedVelvet @Codegate CTF 2018
- Used to deflate flating codes
  - obf @QWB CTF 2018
  - https://security.tencent.com/index.php/blog/msg/112

```
st = p.factory.blank_state(addr=0x401284)
st.regs.rbp = 0xffffffd800000
st.regs.rsp = 0xffffffe800000
flag = claripy.BVS('flag', 26*8)

for i in flag.chop(8):
    st.add_constraints(i >= 0x20)
    st.add_constraints(i <= 0x7f)

st.memory.store(st.regs.rbp-0x100, flag)
p.hook(0x40140E, UserHook(user_func=ptrace, length=5))

sm = p.factory.simgr(st)
sm.explore(find=0x401527)

found = sm.found[0]
print(found.se.eval(flag, cast_to=str))
```

Dynamics Taint Analysis

- Mostly, this term is only refer to a academic technique
- But there are still some simple practical usage
  - Pattern create/pattern offset
  - Find function pointer by filling pattern to memory
    - LFA @ 34C3 CTF , solidcore @ QWBCTF
    - http://atum.li/2017/12/31/LFA/

```
gdb-peda$ pattern create 100
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA
gdb-peda$ pattern offset AEAA
AEAA found at offset: 35
gdb-peda$
```

```
payload="A"*4*$ooa_size

while i<$ooa_size do
    payload_u=u64(payload[j,j+8])
    $arr[i]=toint(payload_u&0xffffffff)
    $arr[i+1]=toint(payload_u>>32)
    i=i+2
    j=j+8
end
```

Call For Players

- Team Eur3kA
  - Champion of HCTF2017 & N1CTF 2018
  - Sponsored by JD.com
- Future Schedule
  - XCTF League & Other National CTF (Eur3kA)
  - 0CTF/TCTF (lotus-r3kapig = blue-lotus + flappypig + Eur3kA)
  - DEFCON China CTF/WCTF/HITCON CTF/Nuit du Hack CTF/TMCTF/HITB Singapore/HITB Amsterdam (JD-r3kapig = flappypig + Eur3kA)
- eu@r3ka.eu

Thanks

Questions are welcome