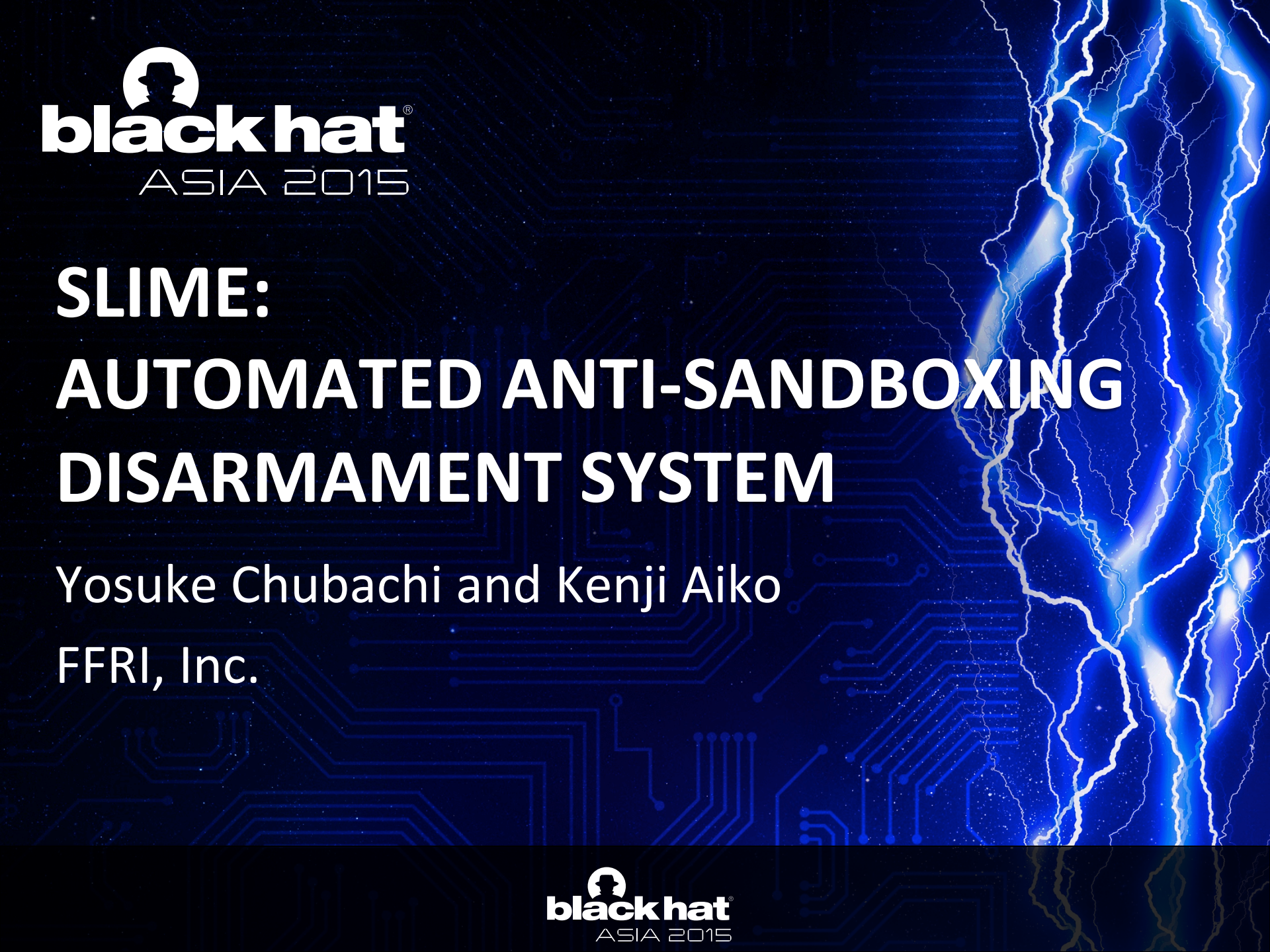


SLIME: AUTOMATED ANTI-SANDBOXING DISARMAMENT SYSTEM

Yosuke Chubachi and Kenji Aiko

FFRI, Inc.

black hat
ASIA 2015

About us

Yosuke Chubachi

He is a security engineer at FFRI, Inc. since last spring. He studied at the graduate school of information system engineering, University of Tsukuba. He is a Security Camp lecturer and a member of executive committee of SECCON since 2012.

Kenji Aiko

He is a programmer at FFRI, Inc., and is a one of the developers of "FFR yarai" which is a targeted attack protection software. He is a Security Camp lecturer and a member of executive committee of SECCON since 2012.

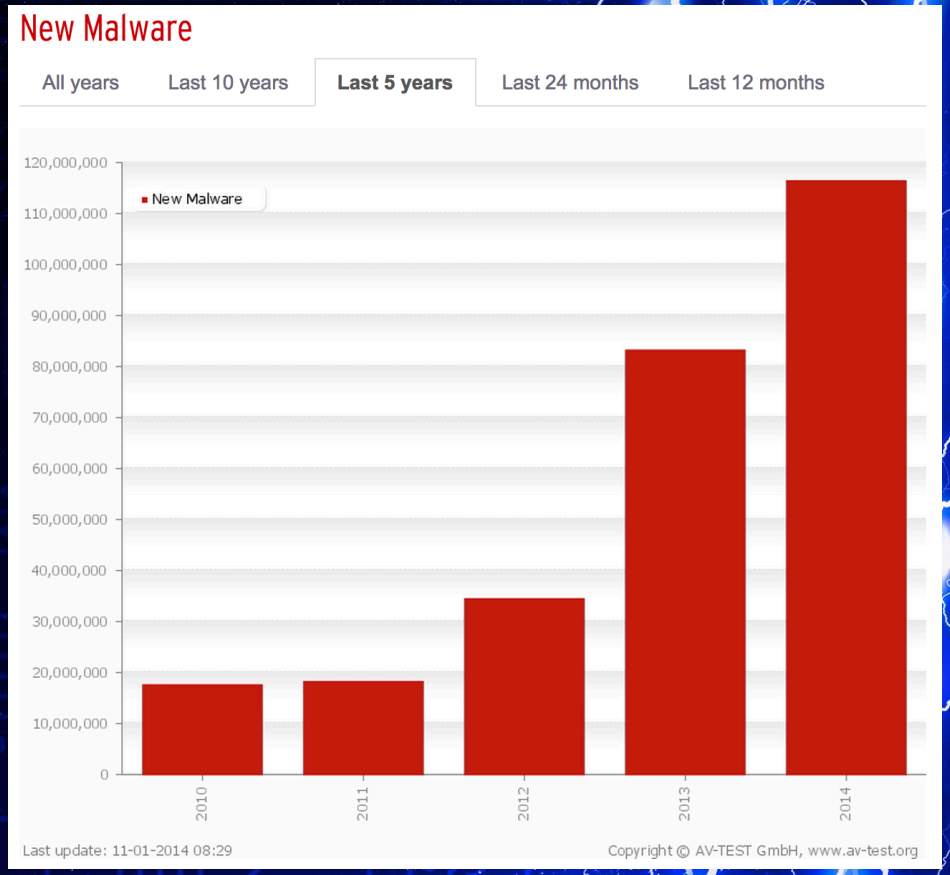
Contents

- Background and Motivation
- State of the Art of Anti-sandboxing
- SLIME Design and Implementation
- Disarming Real Malware
- Experiments
- Conclusion

Background and Motivation

Background

- Malware explosion
  - 120,000,000 over in 2014
- Antivirus is dead…?



AV Test: Statistics –New Malware- (Nov. 05 2014 viewed)
http://www.av-test.org/en/statistics/malware/

We need dynamic and automated malware analysis

- "Scalability" is most important factor in information explosion era
  - — Cloud
  - — Bigdata
  - — IoT
- Malware analysis also needs "scalable" methodology

“Use the sandbox, Luke”

- Security engineer and researcher use sandbox environment for malware analyzing
- Automated dynamic analysis technology also based on VM/application sandbox

Malware strike back

- Sophisticated malware arms many **anti-analyze** techniques
  - Naturally using targeted attacks, cyber espionage, banking malware

- Researchers called those malware "evasive malware"

Related work

- BareCloud [Dhilung K et al., USENIX SEC'14]
  - "5,835 evasive malware out of 110,005 recent samples"

- Prevalent Characteristics in Modern Malware [Gabriel et al., BH USA '14]
  - "80% malware detect vmware using backdoor port"

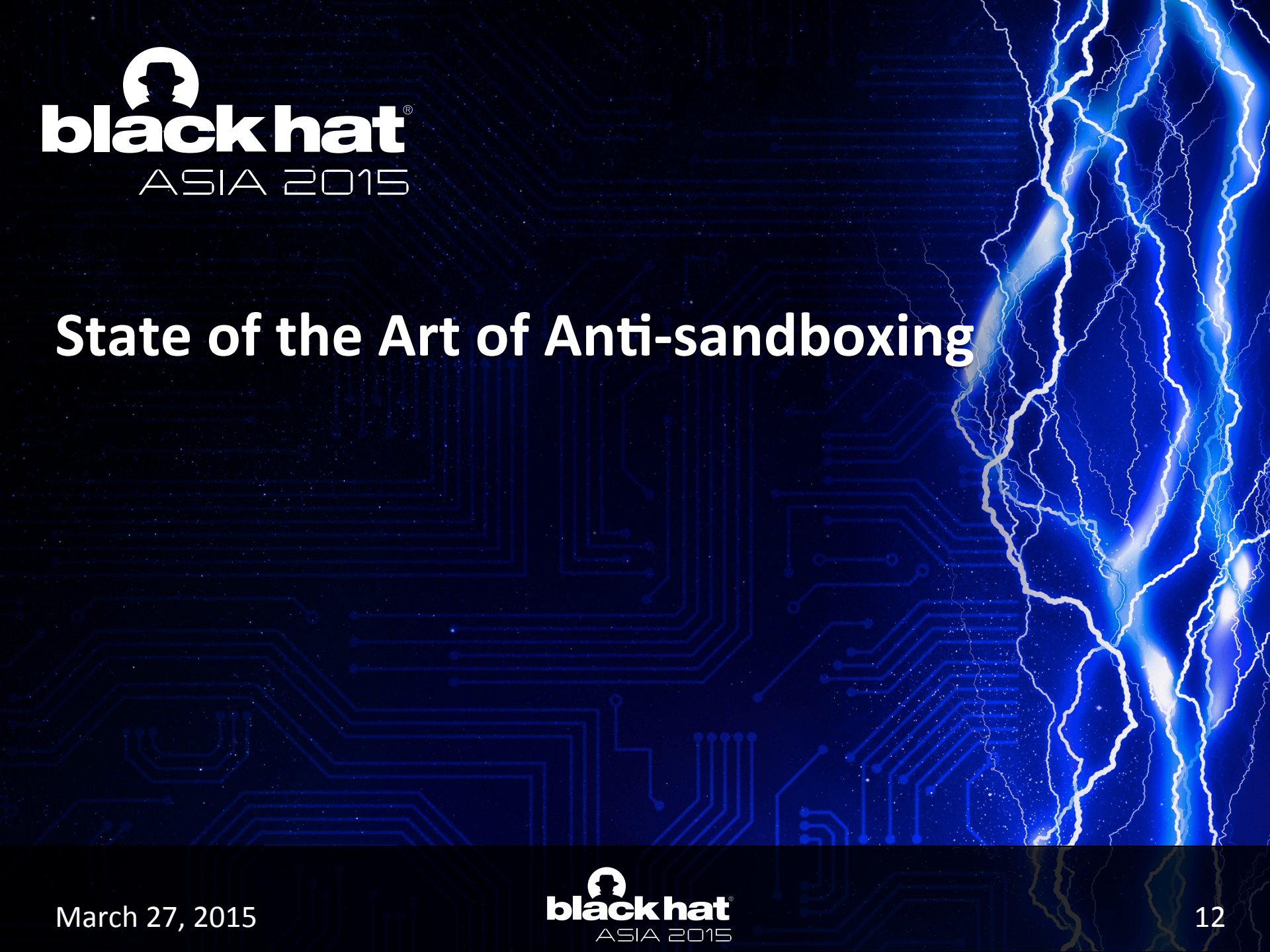
What do you think?

Motivation

- Investigating into a condition used by sandbox evasion automatically for select right sandbox using investigated conditions

Challenges

- Incorporable and standalone
  - Because we are developing anti virus application


black hat
ASIA 2015
State of the Art of Anti-sandboxing

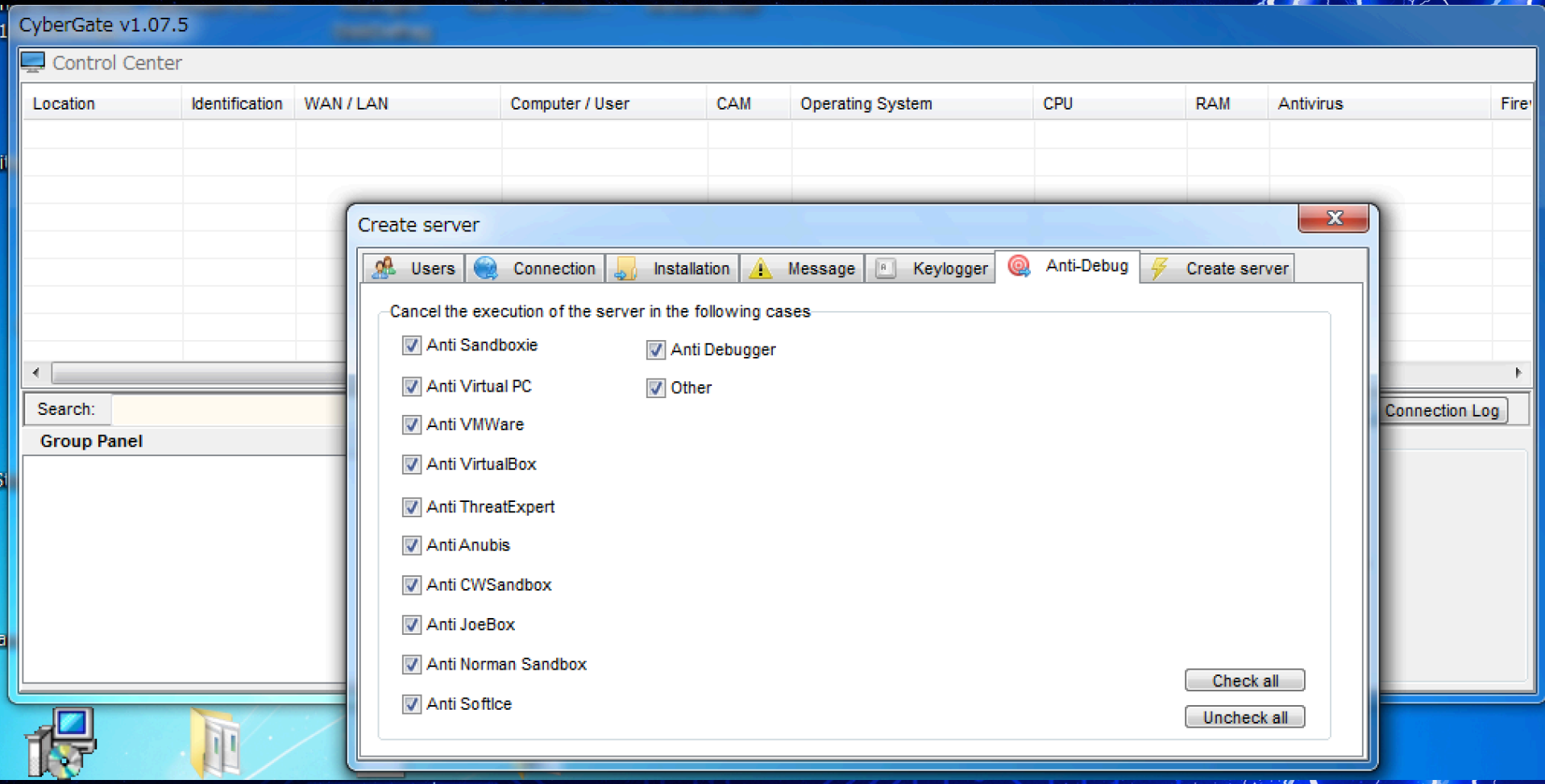
State-of-the-art anti-sandboxing

- CyberGate (RAT)
- Chthonic (Online Banking Malware)

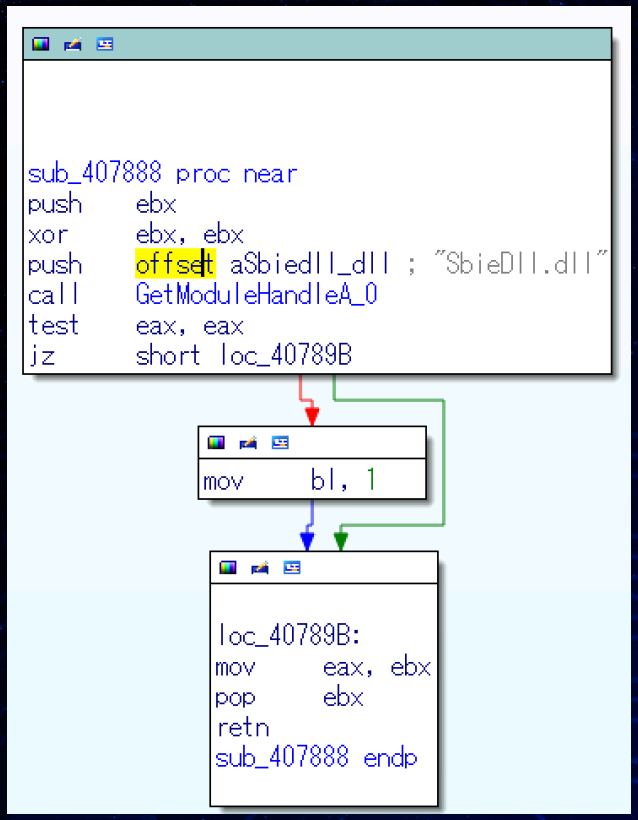
CyberGate

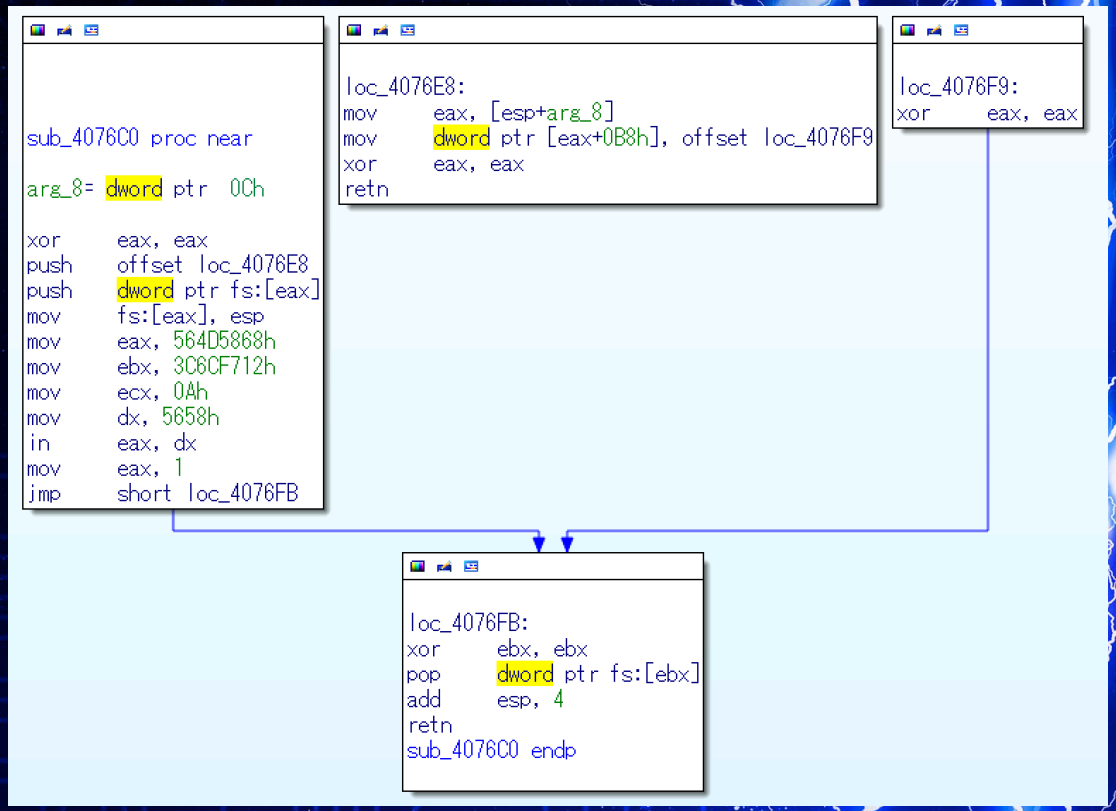
- Popular RAT tools
- CyberGate can generates remote access server for targeting host
- Anti-sandbox option enabled

CyberGate


CyberGate v1.07.5
Control Center
Location
Identification
WAN / LAN
Computer / User
CAM
Operating System
CPU
RAM
Antivirus
Search:
Group Panel
Connection Log
Create server
Users
Connection
Installation
Message
Keylogger
Anti-Debug
Create server
Cancel the execution of the server in the following cases
Anti Sandboxie
Anti Virtual PC
Anti VMWare
Anti VirtualBox
Anti ThreatExpert
Anti Anubis
Anti CWSandbox
Anti JoeBox
Anti Norman Sandbox
Anti SoftIce
Anti Debugger
Other
Check all
Uncheck all

Anti-sandboxing are generated by CyberGate





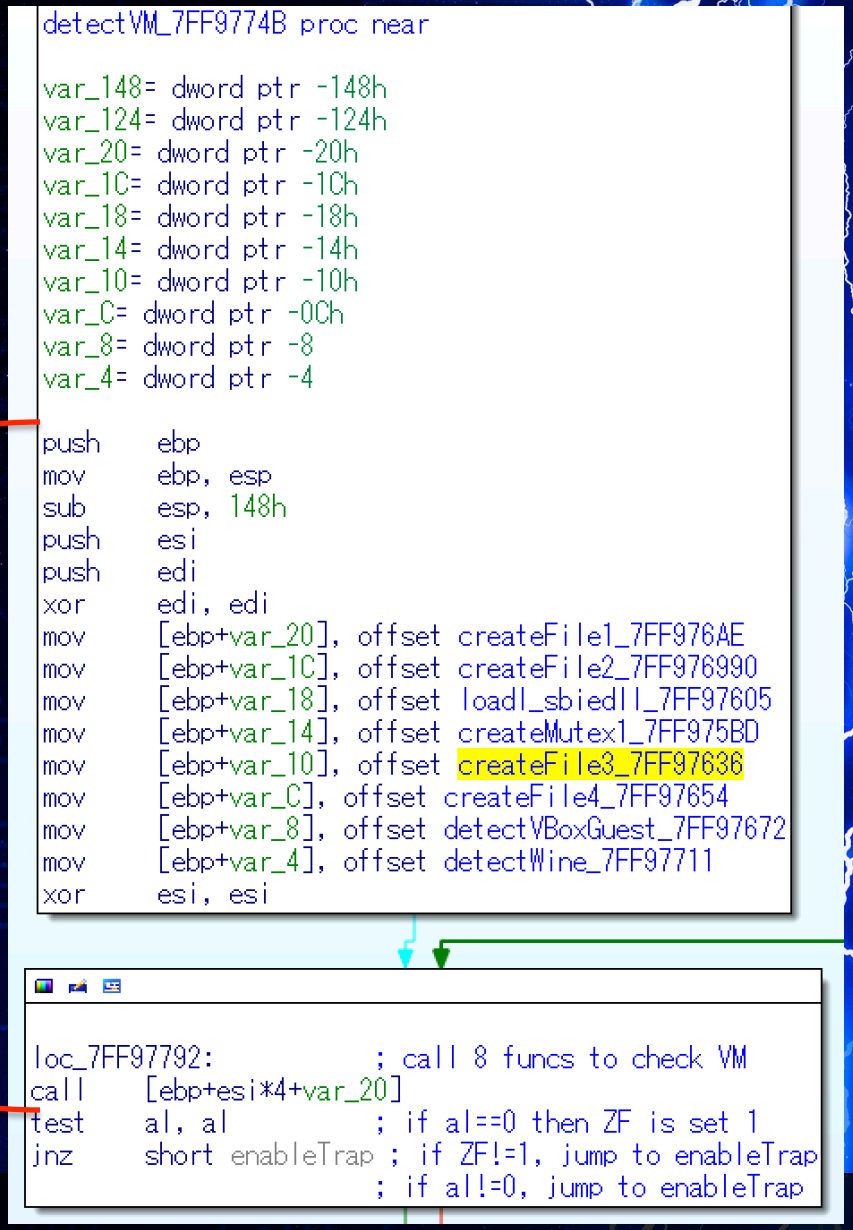
Chthonic

- Banking trojan subspecies of ZeuS Family
- Chthonic downloader injects malicious code into msiexec.exe
- Also downloader changes its behavior if runs on sandbox or virtual machines

See also:
https://securelist.com/blog/virus-watch/68176/chthonic-a-new-modification-of-zeus/

Chthonic

Calling many
vm/sandbox detection



Chthonic anti-sandboxing

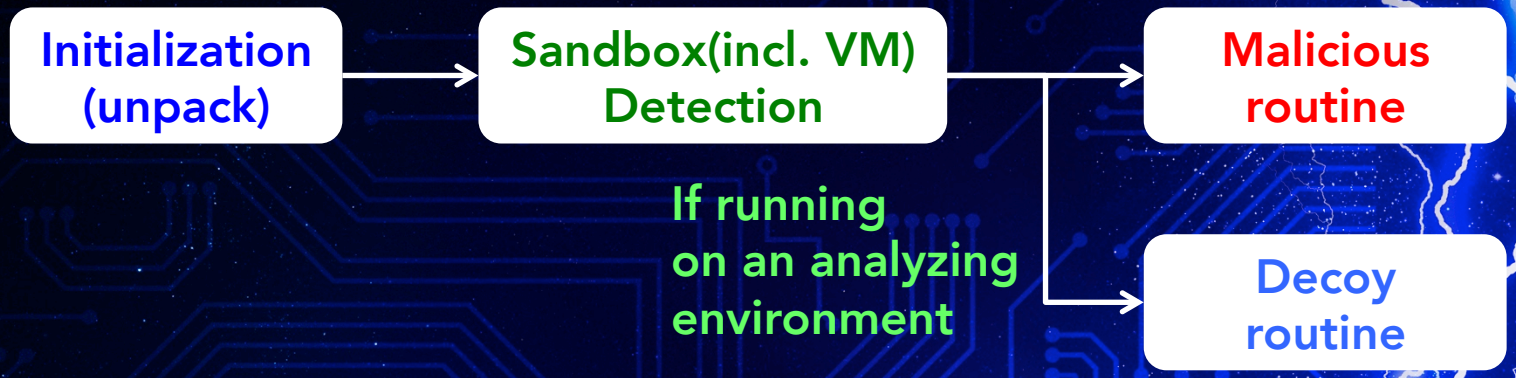
- Finding vm/sandbox artifacts
  - \\.\HGFS , \\.\VBoxGuest, \\.\vmci and \\.\Wine
  - sbie.dll
- Similar "Citadel"
  - Citadel also finding vm/sandbox artifacts

Type of anti-sandbox

- Anti-sandbox maneuver
  - ✓ Environment awareness
    - Using result of vm/sandbox detection
    - Host fingerprinting
  - ☐ (Stalling code)
  - ☐ (User/Network interaction checks)

Environment awareness

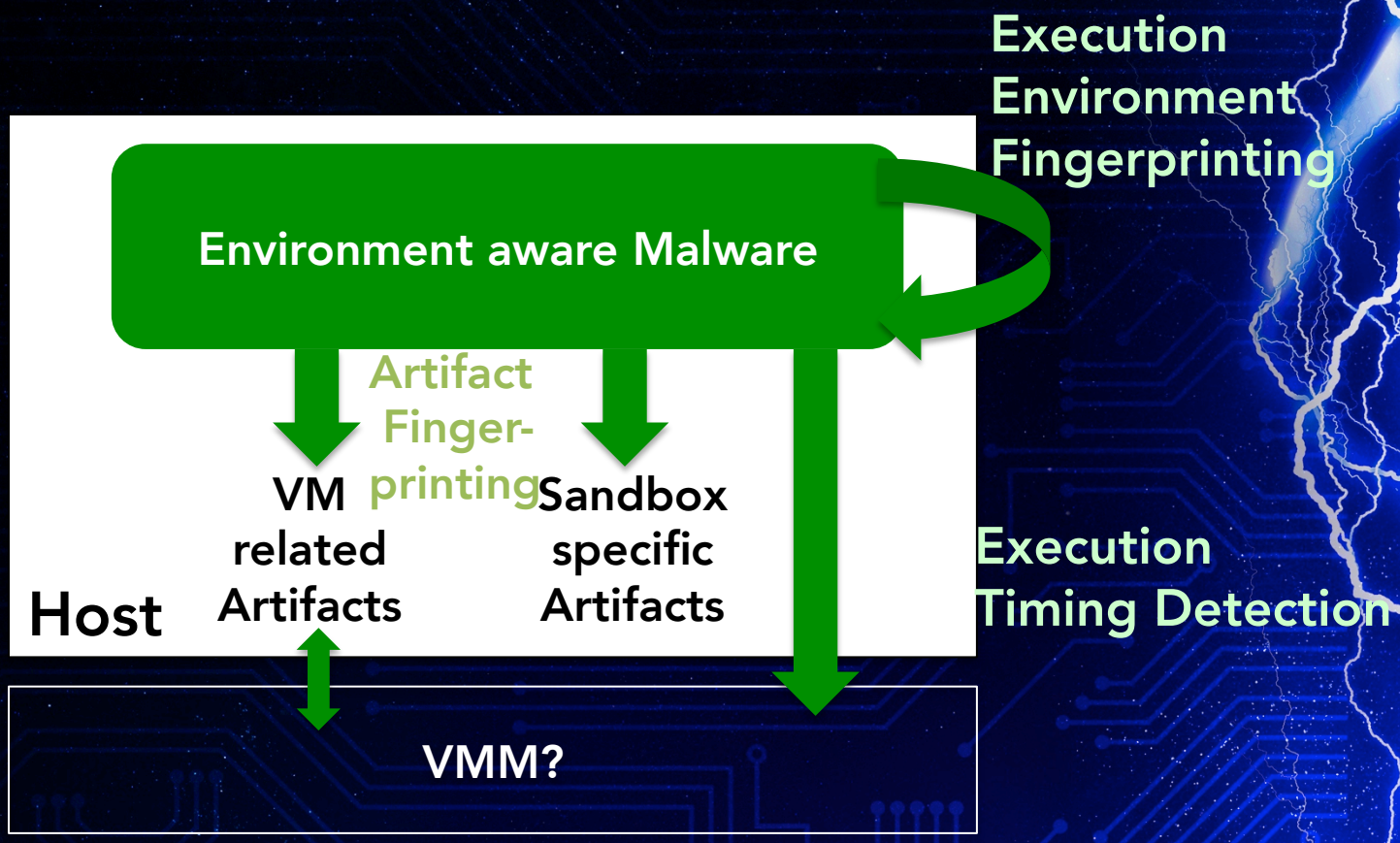
- Checking host environments
- If malware runs decoy routine or exit itself then it detects analyzer's sign
  - Malicious behavior never executed



Sandbox (debug/sandbox/vm) detection

- ✓ Artifact fingerprinting
- ✓ Execution environment fingerprinting
- ☐ (Execution timing detection)

Sandbox (debug/sandbox/vm) detection



Artifact fingerprinting

- Sandbox/VM environment specific files
- Sandbox/VM environment specific registry keys
- Sandbox/VM environment specific devices and its attributes
  - ex). QEMU HDD vendor name
- Sandbox/VM Specific I/O port
  - VMWare backdoor port is most famous artifact in malware
- Sandbox/VM related processes
  - Like vmware, virtualbox etc.

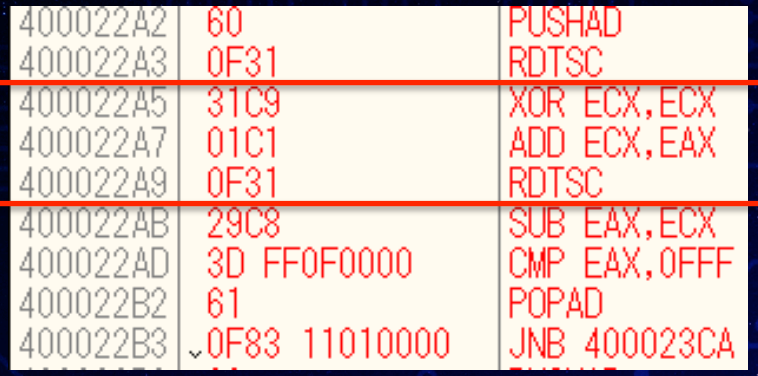
Execution environment fingerprinting

- Using virtual machine implementation specific platform value and reaction
  - CPUID instruction result
  - Redpill
    - Using LDT/GDT and IDT incongruousness
  - Interesting research here: Cardinal Pill Testing

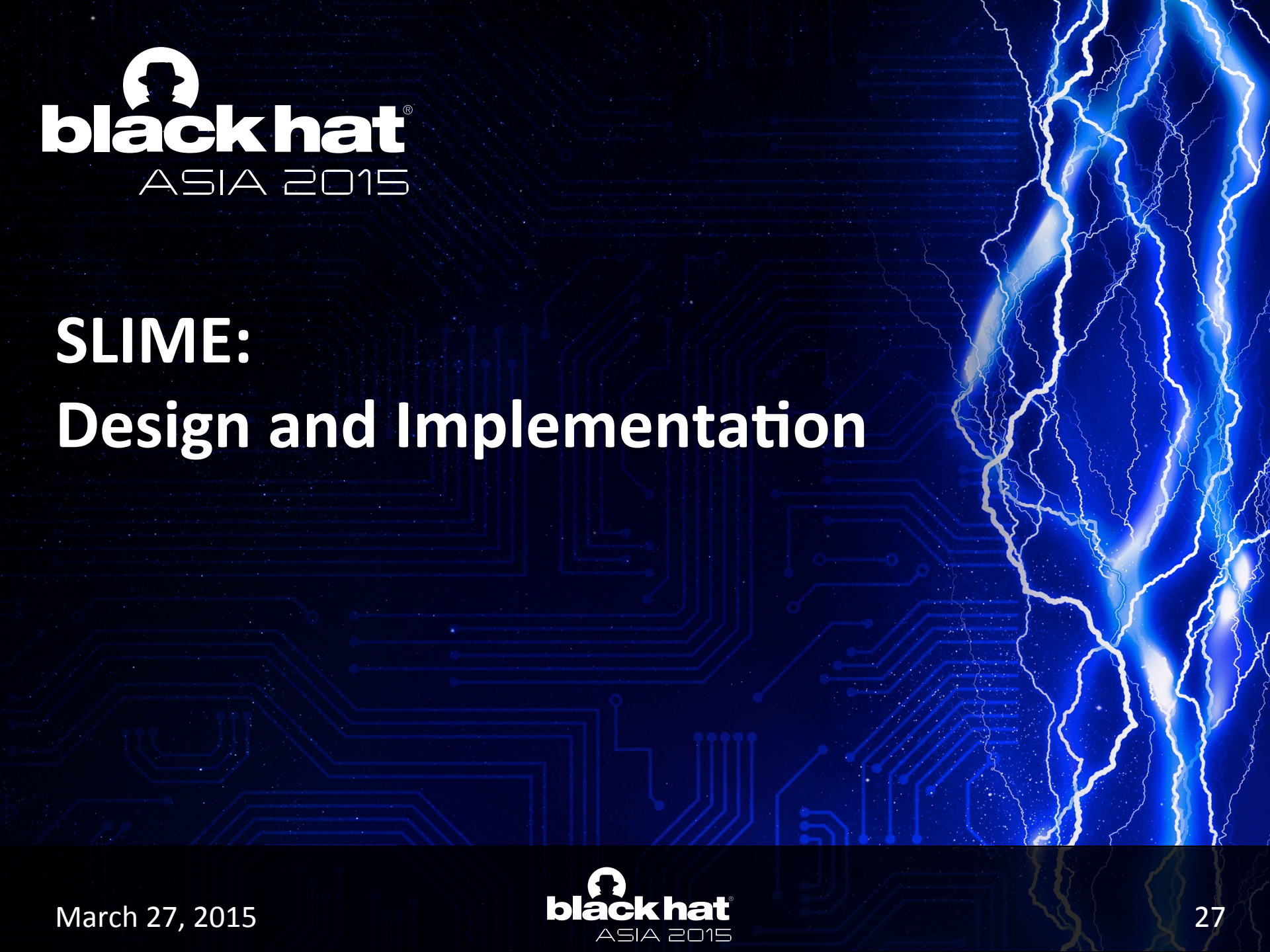
Execution timing detection

- Using clock count differential
  - Traditional anti-debug technique

Comparing TSC differentials



```
400022A2  60              PUSHAD
400022A3  0F31            RDTSC
400022A5  31C9            XOR ECX,ECX
400022A7  01C1            ADD ECX,EAX
400022A9  0F31            RDTSC
400022AB  29C8            SUB EAX,ECX
400022AD  3D FF0F0000     CMP EAX,0FFF
400022B2  61              POPAD
400022B3  0F83 11010000   JNB 400023CA
```



SLIME:
Design and Implementation

SLIME key technologies

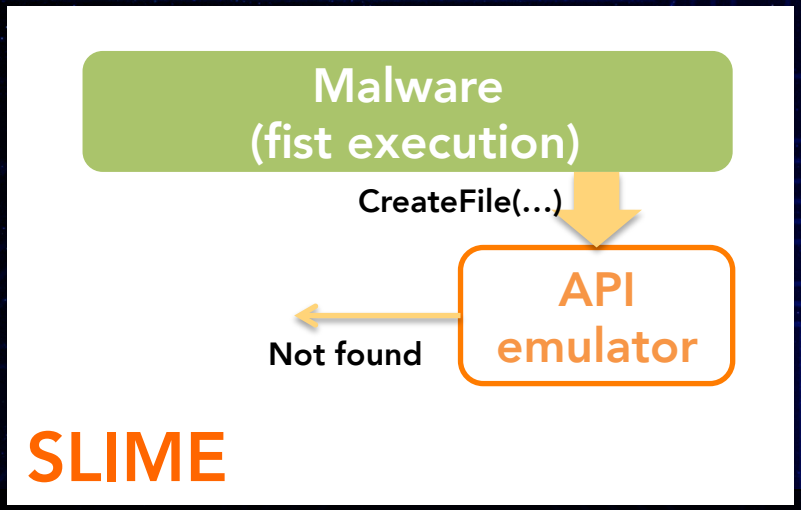
- Malware palpation
- Code execution integrity(CEI)
- Retroactive condition analysis

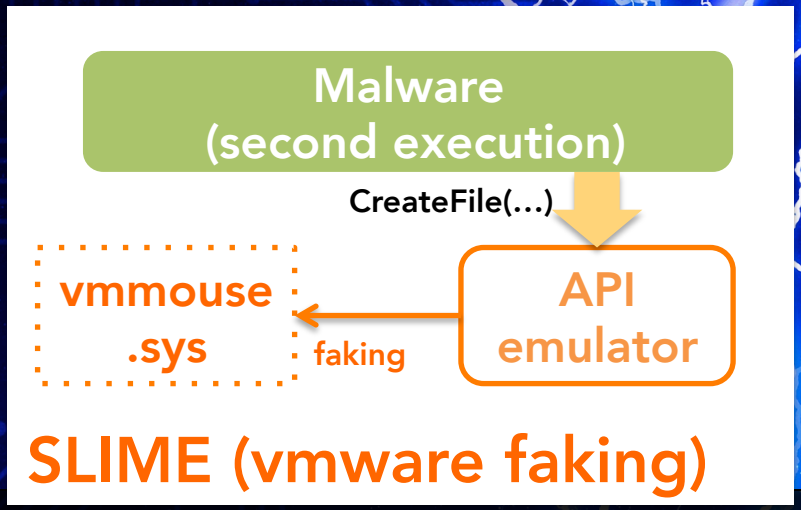
Concept: malware palpation

1. Our sandbox runs malware again and again
   - Changing “virtual” artifacts exposure each execution for execution branch detection
2. Retroactive condition analysis
   - Specifying “branch condition” on unnatural process termination

Malware palpation

- SLIME Sandbox fakes different sandbox-related artifacts each malware execution
  - Detecting execution difference using code execution integrity(CEI)





Code Execution Integrity(CEI)

- CEI shows uniqueness of instruction execution history
  - — Inspired by TPM trust chaining
- "measurement" per instruction

Digest[i] = SHA1(fetched CPU instruction + Digest[i-1])

```
mov   $0x616b6157, %eax   0xb857616b61   d[0] = SHA1(0xb857616b61)
push  %ebx                0x53           d[1] = SHA1(d[0] + 0x53)
push  %eax                0x50           d[2] = SHA1(d[1] + 0x50)
mov   $4, %edx            0xba04000000   d[3] = SHA1(d[2]
mov   $1, %ebx            0xbb01000000                 +0xba04000000)
                                         ...
```

Execution branch detection

- Using execution step count and code execution integrity(CEI) value


```
            in      eax, dx
            cmp     ebx, 0x564D5868h
            jne     NOTVMX
            jmp     ISVMX
NOTVMX:     mov     rc, 0
            jmp     done
ISVMX:      mov     rc, eax
            jmp     done
```

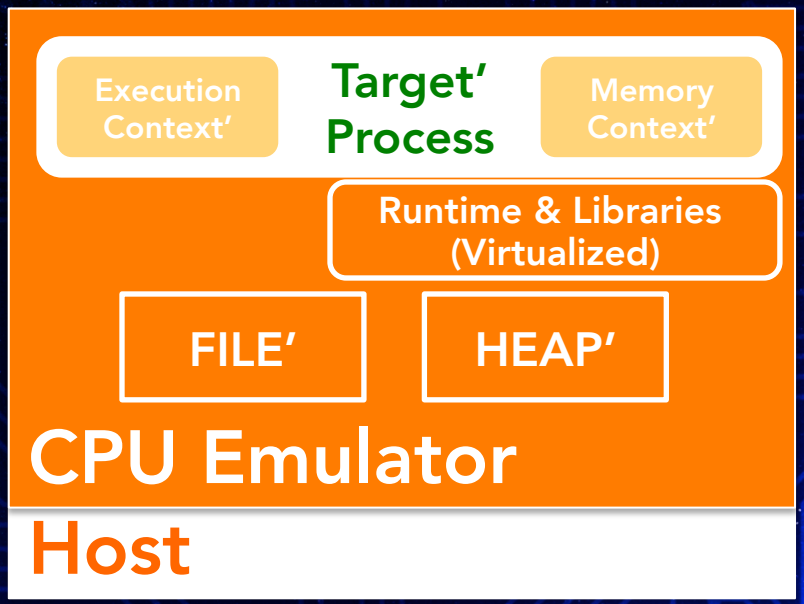
Retroactive condition analysis

- Sandbox retroactive from termination to terminated reason API and arguments when suspicious termination
  - Only a few steps executions
  - To terminate before network activities

```
                sub esp, 1024
                mov ebx, esp
                push 400h
                push ebx
                push 0h
                call GetModuleFileNameA
                lea eax, MyPath
                push eax
                push ebx
                call lstrcmpA
                test eax, eax
                push 0h
                lea eax, MsgCaption
                push eax
                jz _ok
                lea eax, NGMsgText
                push eax
                push 0h
                call MessageBoxA
                invoke ExitProcess, NULL
_ok:            lea eax, OKMsgText
```

Implementation

- We have already CPU Emulator-based sandbox for win32 execution (in-house use)
  - —Like IDA Bochs PE operation mode[11]

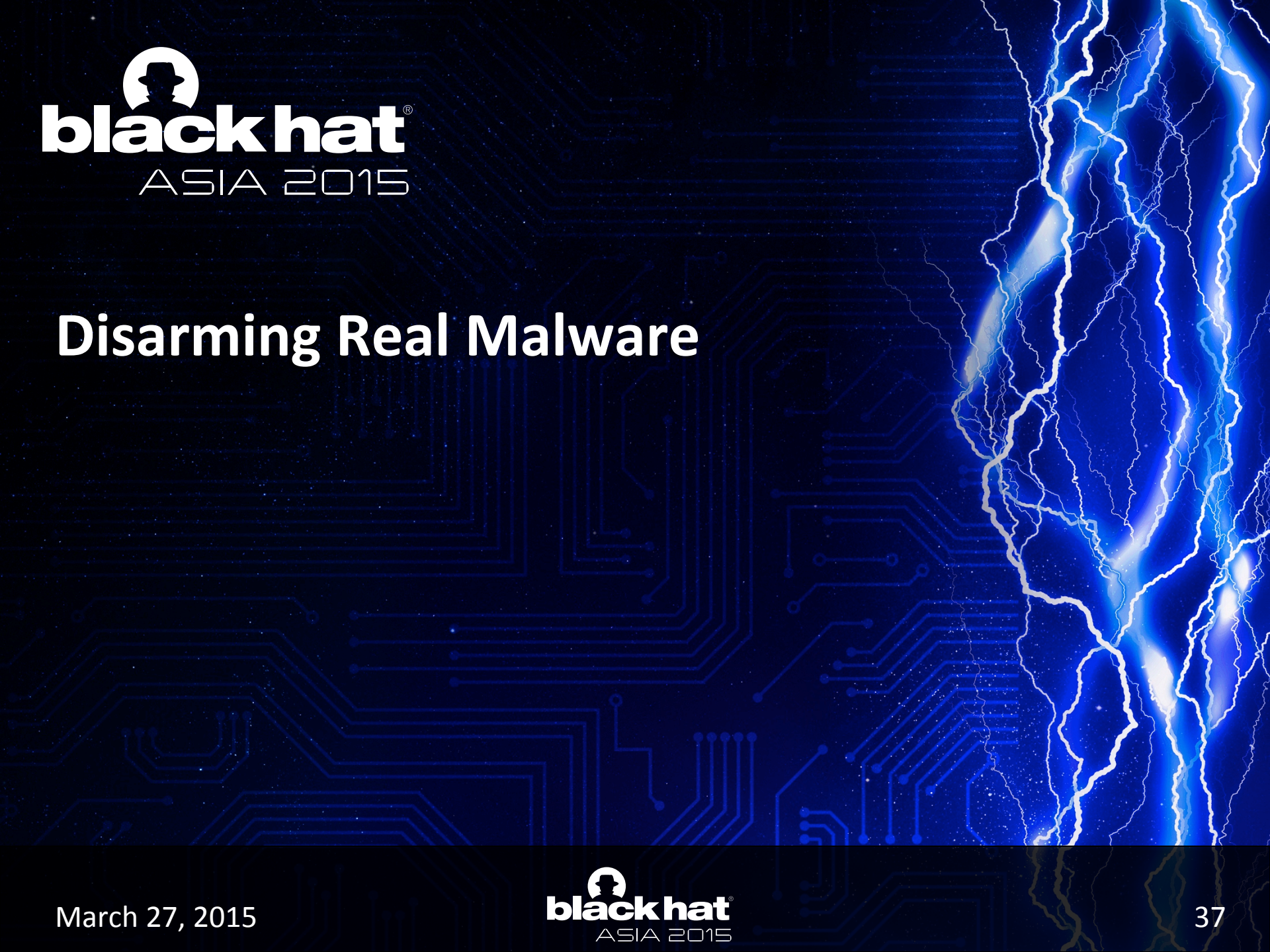


Execution logging framework

- SLIME logs instruction per execution
  - — Tracing specific API call and its arguments for Retroactive condition analysis
    - lstrcmpi, strcmp, GetModuleFileName, …
- Code execution integrity calculation per execution
  - — For execution branch detection

Camouflaging VM/sandbox related artifact existence

- VMWare
  - Camouflaging backdoor port, some registry entry and files
- VirtualBox
  - Some registry entry and files
- QEMU
  - some registry entry and files
- Sandbox
  - Anubis
  - Sandboxie
  - ThreatExpert



Disarming Real Malware

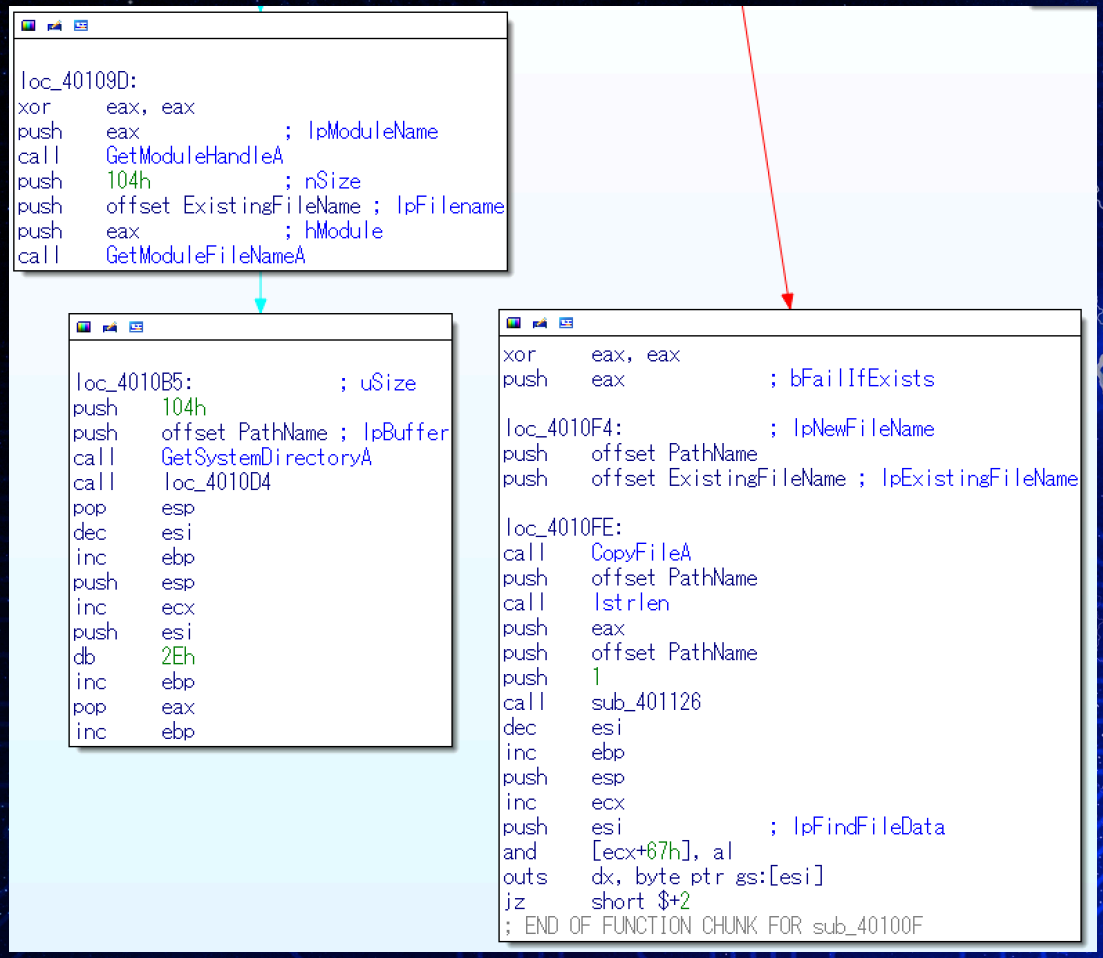
Disarming demo

Anti-VMWare

```
push    edx
push    ecx
push    ebx
mov     eax, 564D5868h
mov     ebx, 0
mov     ecx, 0Ah
mov     edx, 5658h
in      eax, dx
cmp     ebx, 564D5868h
setz    [ebp+var_19]
pop     ebx
pop     ecx
pop     edx
mov     [ebp+ms_exc.registration.TryLevel], 0FFFFFFFFh
mov     al, [ebp+var_19]
mov     ecx, [ebp+ms_exc.registration.Next]
mov     large fs:0, ecx
```

SHA256: C1A7E51E5E2F94193D6E17937B28155D0F121207

Detect sandbox evasion



```
loc_40109D:
xor     eax, eax
push    eax             ; lpModuleName
call    GetModuleHandleA
push    104h            ; nSize
push    offset ExistingFileName ; lpFilename
push    eax             ; hModule
call    GetModuleFileNameA
```

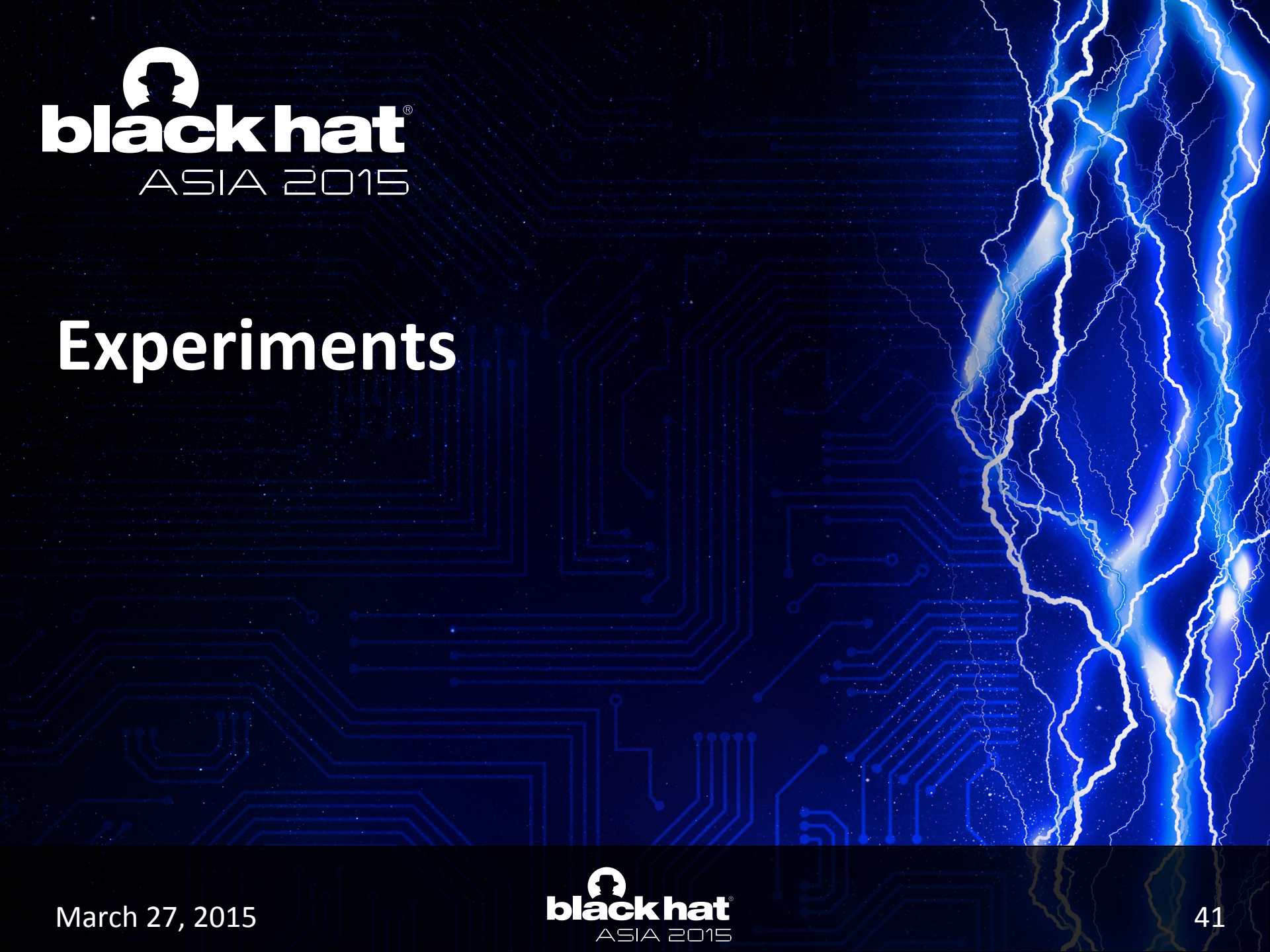
```
loc_4010B5:             ; uSize
push    104h
push    offset PathName ; lpBuffer
call    GetSystemDirectoryA
call    loc_4010D4
pop     esp
dec     esi
inc     ebp
push    esp
inc     ecx
push    esi
db      2Eh
inc     ebp
pop     eax
inc     ebp
```

```
xor     eax, eax
push    eax             ; bFailIfExists

loc_4010F4:             ; lpNewFileName
push    offset PathName
push    offset ExistingFileName ; lpExistingFileName

loc_4010FE:
call    CopyFileA
push    offset PathName
call    lstrlen
push    eax
push    offset PathName
push    1
call    sub_401126
dec     esi
inc     ebp
push    esp
inc     ecx
push    esi             ; lpFindFileData
and     [ecx+67h], al
outs    dx, byte ptr gs:[esi]
jz      short $+2
; END OF FUNCTION CHUNK FOR sub_40100F
```

SHA256: 39517A057CC4A1AE34E786873C8010291A33BAB7



Experiments

Dataset

- Trying to disarm 89,119 malware
  - Collected in one year (2014/01/01-2014/12/31)
  - Original data amounts: 5,244,297
  - Random sampling
  - Filtered in PE(32bit) and loadable our sandbox

Results

Anti-Sandbox Type	Count
Detecting VMWare	63
Detecting VirtualBox	70
Detecting QEMU	84
Detecting Sandbox (sbie.dll and dbghelp.dll)	11,102
Evasive Malware	36

* Throughput: 6 malware per minites

Are Anti-VM Too Few?

- We guess that more Anti-VM malware exists in this dataset CPU
  - — Because our CPU emulator coverage is not enough to run malware
    - Original sandbox was developed for unpacking

Offtopic: Artifact finding by Yara

Anti-Sandbox Type	Count
Found VMWare Signature	11,029
Found VirtualBox Signature	530
Found QEMU Signature	247
Sandbox detection	235

Using customized Anti-VM rules@YaraRules

Anti-Sandbox Type	Count
Found VMWare Signature	10,985
Found VirtualBox Signature	142
Found QEMU Signature	127
Sandbox detection	221

Using SLIME implemented artifact only

Can Virtual Machine Protects You from Malware?

- No
  - The proportion of Anti-VM armed malware is low in the wild
  - Anti-VM activity is one of method of black list avoiding

Can I Ignore Anti-Sandboxing?

- No!
  - Many anti-sandboxing founds before malicious behavior such as suspicious download or code injection
  - If you do not pay attention, you will be miss significant threat

Conclusion

- SLIME can investigate into a condition used by sandbox evasion automatically
- The proportion of anti-VM armed malware is low in the wild
- However, there is no doubt that sophisticated malware often uses anti-sandboxing

Bibliography

- Analyzing Environment-Aware Malware, Lastline, 2014.05.25(viewed) http://labs.lastline.com/analyzing-environment-aware-malware-a-look-at-zeus-trojan-variant-called-citadel-evading-traditional-sandboxes
- Martina Lindorfer, Clemens Kolbitsch, and Paolo Milani Comparetti. 2011. Detecting environment-sensitive malware. In *Proceedings of the 14th international conference on Recent Advances in Intrusion Detection* (RAID'11). Springer-Verlag, Berlin, Heidelberg, 338-357.
- lemens Kolbitsch, Engin Kirda, and Christopher Kruegel. 2011. The power of procrastination: detection and mitigation of execution-stalling malicious code. In Proceedings of the 18th ACM conference on Computer and communications security (CCS '11). ACM, New York, NY, USA, 285-296.
- Min Gyung Kang, Heng Yin, Steve Hanna, Stephen McCamant, and Dawn Song. 2009. Emulating emulation-resistant malware. In Proceedings of the 1st ACM workshop on Virtual machine security (VMSec '09). ACM, New York, NY, USA, 11-22.
- Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. 2014. Barecloud: bare-metal analysis-based evasive malware detection. In Proceedings of the 23rd USENIX conference on Security Symposium (SEC'14). USENIX Association, Berkeley, CA, USA, 287-301.
- Ulrich Bayer, Imam Habibi, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. 2009. A view on current malware behaviors. In Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more (LEET'09). USENIX Association, Berkeley, CA, USA, 8-8.
- Aurélien Wailly. Malware vs Virtualization The endless cat and mouse play, 2014.05.25(viewed) http://aurelien.wail.ly/publications/hip-2013-slides.html
- Lorenzo Martignoni, Roberto Paleari, Giampaolo Fresi Roglia, and Danilo Bruschi. 2009. Testing CPU emulators. In Proceedings of the eighteenth international symposium on Software testing and analysis (ISSTA '09). ACM, New York, NY, USA, 261-272.
- Hao Shi, Abdulla Alwabel and Jelena Mirkovic. 2014. Cardinal Pill Testing of System Virtual Machines. In Proceedings of the 23rd USENIX conference on Security Symposium (SEC'14). USENIX Association, Berkeley, CA, USA,271-285.
- Lorenzo Martignoni, Roberto Paleari, Giampaolo Fresi Roglia, and Danilo Bruschi. 2010. Testing system virtual machines. In Proceedings of the 19th international symposium on Software testing and analysis (ISSTA '10). ACM, New York, NY, USA, 171-182.
- IDA Boch PE operation mode https://www.hex-rays.com/products/ida/support/idadoc/1332.shtml

Fin.