COSEINC®

Solid Security.Verified.

Automatic Malware Analysis

Udi Shamir (tal0n)

udi@coseinc.com

COSEINC®

Solid Security.Verified.

Malware

Program which infiltrate machine with out the user awareness.

Few Vectors:

Exploits - CVE-2008-4250 (MS-RPC Vulnerability) Conficker.

Social Engineering – email , phishing.

Drive by Downloads – infected web site

0days – new vulnerabilities , RBN.

can install keyloggers to steal sensitive information

can hijack machines, install applications, servers, filter drivers

can make the machine participate in DDoS Attacks

COSEINC®

Solid Security.Verified.

The Challenge ..

Analysis Facts

Malwares authors tends to use packers.
 (Originally used to reduce the binary size, Encode strings and data protection)

Moderns Packers:
Hard to disassemble
polymorphic engines (uber-kewl)
VM and debugger's detection

time consuming
Millions are getting infected, thousands of new variants daily.

Polymorphic Code

Malware uses the same algorithm while its engine mutate the code.

The idea is to change the code each time the malware is being executed so it will create difficulties for AV vendors to produce efficient signature's as well as for malware researcher, OS Vendors (Most Are Lame) to understand the malware behavior.

Polymorphic Code

```
Start:
GOTO Decryption_Code Encrypted:
        ... lots of encrypted code ...
Decryption_Code:
        A = Encrypted
Loop: B = *A
        B = B XOR CryptoKey
        *A = B
        A = A + 1
GOTO Loop IF NOT A = Decryption_Code
GOTO Encrypted CryptoKey:
        some_random_number
```

COSEINC®

Solid Security.Verified.

Polymorphic Code (Mutation)

Start:

GOTO Decryption_Code Encrypted: ->> same algorithm

... lots of encrypted code ... ->> same algorithm

Decryption_Code:

c=2^(c*4) << -- Mutation

A = Encrypted : ->> same algorithm

Loop: B = *A : ->> same algorithm

C=C*(A+A) << -- Mutation

B = B XOR CryptoKey

*A = B ->> same algorithm

A = A + 1 << -- Mutation

GOTO Loop IF NOT A = Decryption_Code

C=C+8 << -- Mutation

GOTO Encrypted CryptoKey:

some_random_number ->> same algorithm

COSEINC®

Solid Security.Verified.

Anti Debugging

some common ways malwares are using to check if a debugger is present.

.. Groovy Way ..
push offset @not_debugged
push dword fs:[0] <← Install SEH handler
mov fs:[0], esp <<- Set the Hook
push 1234h ;invalid handle <<- **Generate Fault**
call CloseHandle
; if fall here (cant MOV next), process is debugged

push @not_debugged
call SetUnhandledExceptionFilter <<- **No Handler Should Exist**
xor eax, eax
mov eax, dword [eax] ; trigger exception
;program terminated if debugged

COSEINC®

Solid Security.Verified.

Popular Packers

Armadillo – considered one of the best packers, supports encryption, debugger detection, vm detection.

Themida – considered one of the best packers, supports encryption, debugger detection

VMProtect – an advanced protection and compression product

UPX – Free and popular opensource packer, uses LZMA

ASPack – consider weak and outdated.

FSG – No loner maintained but still popular, supports only static PE

The Need Simplicity

* Malware analysis is not an easy task.
* Deep Knowledge with operating system internals.
* Experience with low level programming.
* Be a Bold Guy ☺

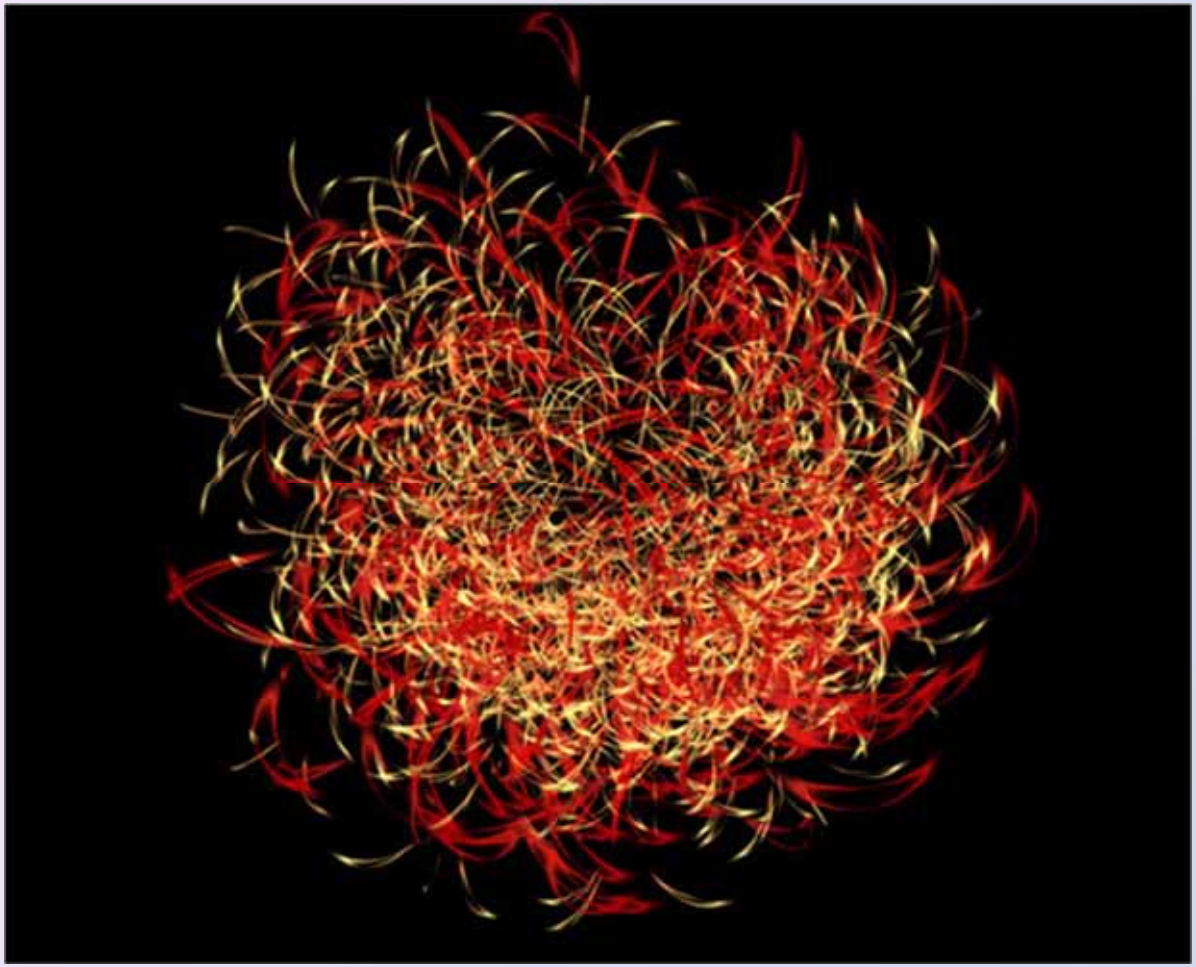
Why Automation ?

* Not every one wants to disassemble code
* Speed up the analysis process
* Help malware researchers to test more variants and learn new attack vectors (Observing from the side)
* its fun ?

COSEINC®

Solid Security.Verified.

Coseinc Advanced Malware Analysis Lab



Goals

*Research and learn new malwares attack vectors
*Speed up the analysis process time
*Provide the analyst with real time information regarding the binary activity inside the operating system internals

COSEINC®

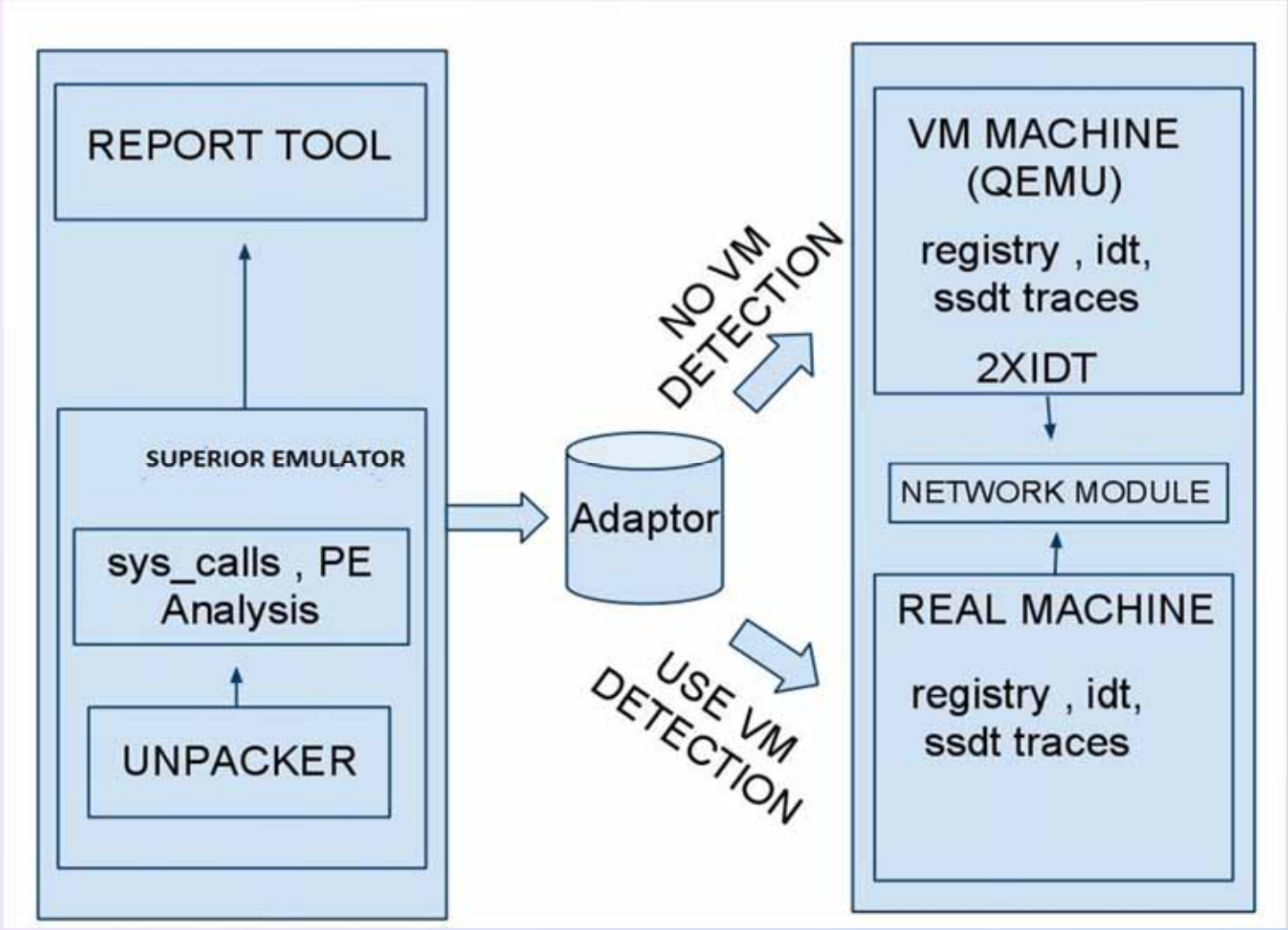
Solid Security.Verified.

CAMAL Internals

COSEINC®

Solid Security.Verified.

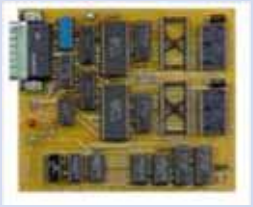
Framework Modules



COSEINC®

Solid Security.Verified.

Camal Superior Emulator (CSE)



*Emulating the following CPU architectures:
ARM | MIPS | MOS 6502 | Motorola 68K | Motorola 88K | X86 | PowerPC | FAPRA

* build over low level virtual machine (LLVM) for the backend. user mode and system emulation, dynamic as well as static recompilation.
* Detect SIDT VMM query techniques by looking for ring 3 0xff*XXXXXX address.*
* 248K foot print.

COSEINC®

Solid Security.Verified.

UNPACKER

* Uses the Superior emulator for the execution process.
* Support unpacking for the following packers:
  UPX(x)
  Aspack
  PE-Compact
  FSG
* Support Real Time Brute Force (avoid packers lock).

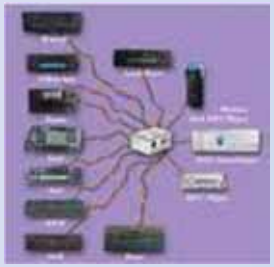
PE Analysis

* Section Header: The section table is an array of IMAGE_SECTION_HEADERs structures. An IMAGE_SECTION_HEADER provides information about its associated section, including location, length, and characteristics.
* Import Tables : Load the imports the windows loader needs to use when executing, for instance kernel32.dll, user32.dll.
* RVA: simply an offset in memory, relative to where the PE file was loaded. (PE GPS)

COSEINC®

Solid Security.Verified.

Adapter



* Parse the emulator results
* If vm detect found (SIDT 0xfff), send sample to real machine
* Sample compiled for un-supported architecture
* report hangs, signals , abnormal behavior

COSEINC®

Solid Security.Verified.

QEMU (Virtual machine)

* pre-installed with CAMAL DTSR driver
* compiled in emulation mode (non VMX support)
* modified revert version

COSEINC®

Solid Security.Verified.

Real Machine

* Non emulation environment
* Support reverting by reloading via BOOTP
* Pre-installed with CAMAL DTSR driver

COSEINC®

Solid Security.Verified.

Software being used

* QEMU *www.qemu.org*
* tcpdump www.tcpdump.org
* sysinternals *technet.microsoft.com/en-us/sysinternals/default.aspx*
* *debian debian.org*

COSEINC®

Solid Security.Verified.

LETS DEMO THE FRAMEWORK …