**INTRO:**

PoisonIV

ZeXshell

Verizon data breach report

**Some good talks:**

Defcon 17 malware freak show

Defcon 17 making fun of malware

Defcon 18 my life as sypware developer

**Types of Virus:**

Virus (file infector)

Trojan

Worms

Bot

Rootkits

RAT(remote access trojans)

More specified : scareware, spyware, adware, backdoor credential stealers, anti-analysis Defenses stealth, loader/downloader

**Tools:**

SysInternals

MAP Pack

010 - sweetscape.com/010editor

PE viewer : CFF Explorer PE Explorer PE View PE Studio

IDA Pro or Hopper

Cygwin(add binutils, xxd , gcc)

7zip

**Some interesting stuff on malware:**

Contagio Malware dump

KernelMode.info

Malshare

Malware, Iu's AVCaesar

MalwareBlacklist

Malware DB

Malwr

Open Malware

SecuBoX

**LAB SETUP:**

The usual industrial standard setup is victim(Windows VM) and a router(Linux VM) and NAT to the internet

to configure internet

For the windows machine we use a virtual network adapter

and for kali we use NAT and the same virtual adapter

vim /etc/network/interfaces

for the VMS to talk to each other

allow-eth0

iface eth0 inet static

address 10.0.0.1

netmask 255.255.255.0

gateway 10.0.0.1

change allow-hostplug etho to eth1

iface eth1 inet dhcp

next do

ifconfig eth0 down

ifconfig eth0 up

ifconfig eth1 down

ifconfig eth1 up

go to control panel in windows

network and internet connection

go to network connections

for local area connections

set the IP address manually

set it as 10.0.0.2

put default gateway 10.0.0.1

**Dynamic analysis**

Capture BAT (captures major system events)

Regshot (capture registry events also watch file system changes)

PEiD (look for packers and if they are commoly encrypted look for signatures)

LordPE (dump something out of memory)

Import Reconstructor (to rebuild the lost structures that is dumped)

Ollybg 2.0 (debugger)

Visual tools

Other tools/automations:

Local VM

Scripting Local VM

Automating ESX(i)

Cuckoo Sandbox

Malware Farm

VirusTotal, Anubis

Fireye

**Dynamic Malware Analysis:**

IOC (Indicators of Compromise)

We are looking at the Dyre malware:

Install CaptureBAT

Add shortcut to Desktop

Open that properties

Target will be something like: "C:\ProgamFiles\Capture\CaptureBAT.exe" -c -l (add -c and -l to it) -l is for storing the output of captureBAT and -c is for capture

Take a snapshot

Extract the Dyre file (pass infected)

The malware will have a pdf icon

Then we open a hex editor and if we see it will have the MZ file editor

(4D 5A) which means it is a dos executable

Down there you can see there is a PE executable

Now we are first going to run captureBAT

then run the Dyre Sample

We know that on executing the file deletes itself

After some time we will stop capture bat(ctrl+c)

see the log file it created

When we are going through the docs

We see that it creates some files and also some executable in

C:\Documents and Settings\Administrator\Application Data\googleupdaterr.exe

We can see that this updater executable creates a registry key

Then we can see it does a file write which shows it created a deleted file

if we go to c:/program files /capture/logs/deleted files/c/documents and settings/Administrator/Desktop/Dyre

using MAPpack we can get a MD5hash(just right click and do MD5 hash)

now we will go to that googleupdaterr get the MD5 hash

(it will be the same that means it copied itself)

Then it added a registy entry

it is HKCU which means current user logins in

if it is HKLU mean means local user and runs whenever any user logins

These are our first indicators of compromise

another good tool is regshot

launch the tool

take a screenshot of the registary

then run the malware

take the second screenshot

it will generate a log file

if we open it will show something similar

Then we can reset to the screenshot

Now we are going to use sysinternals

Open the tool

Run the malware

Refresh the tool

We can see the GoogleUpdate

Then we can see it

We can jump to entry and delete the key and that way the malware will be disabled

**Why some advanced malwares don’t run?**

Checks for username

sees if its running in sandbox

can it access internet

looks for installed software

sees dates

countries

checks for language detected

sometimes it sleeps for sometime so that it dosent run when or show in dynamic analysis.

**Dynamic Analysis: PART 2**

Indicators of Compromise

File hashes

Strings

Registry Keys

File names

File Paths

Process Names

Industry standard IoC:

OpenIoC project made my mandiant is a group of IoC

Yara is another such project()

for this demo we will be using the illuionBot.

now we check the network settings

the windows machine should be on virtual lan

ie. 10.0.0.2

and the kali machine running

should be connected to 10.0.0.1(eth1)

and connected to the internet (eth0)

so we can ping each other they should work

we will start wireshark and keep monitoring the traffic

start capture bat

Now unzip the malware and run it .

check wireshark

we can see that its trying to reach a particular internet address and dest port is 6667

Now we want the malware to act normally to think its a safe enviroment

so we use some tool in kali called as inetsim which simulates different services

open termainal and type inetsim

do control c to stop the services

we open the inetsim config file(/usr/share/inetsim/conf/inetsim.conf)

vim <file>

we seach in the file using a /

/10.

and we uncomment the line

service\_bind\_address 10.0.0.1

do another seach

/dns

and find the

dns\_default\_ip

uncomment and change it to

dns\_default\_ip 10.0.0.1

:wq

now we run the inetsim

reset wireshark

it shows that its not resolving the dns but just trying to reach an IP

we will stop captureBat now

then we add a new thing in terminal

route ADD 0.0.0.0 mask 255.255.255.255 10.0.0.1

this will funnel all the traffic to 10.0.0.1

now in kali we are going to add a rule

iptables -t nat -A PREROUTING -i eth1 -j REDIRECT

after doing this we can see that the malware made a successful connection

we can follow the TCP stream

we can see that it joins to the irc channel

skaters\_1990

we can find other bots there

the password we can see for the channel is 2580

so we can hop on the channel now

now we can open the inetsim config

comment the start\_service irc

now the irc port and process is no longer working

we can see there is a failed connection that the victim is trying to make. since inetsim resets the irc connection

now what we can do is netcat

so we use

nc -l -p 6667

(now we are listening on that port)

we can see that its the same username and it does the same thing

joins the irc

Try to send some strings like

Hi

Pass

nothing happens

so we send

Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

something like that

and go to windows we can see that

the program crashes which means the malware was not well written

now we stop nc

if we see the wireshark we can see that it dosent contact the server anymore

**BASIC STATIC ANALYSIS:**

What we use here is

Debuggers and Disassemblers

Used to understand the behavioral analysis of the malware.

What we can get from static analysis is

Understand behavior

Find IoC by:

Encrypted Strings/Payloads

Domain Generation Algorithms (some don’t have hardcoded IP)

Network Traffic Encryption Algorithms

Determine Defenses

Determine Capabilities for Assess Risk and Impact

Assembly execution

Windows uses PE files to know what section to execute what verison and all such other metadata about the executable package.

PE file format parsers are:

PE Explorer

COFF Explorer

PEiD

PE Studio

(Make your own : from the malware analysis cookbook)

Most common exec formats are:

exe dll src cpl ocx sys drv efi fon

SHow the Screenshot for the headers

show the screenshot for EFLAGs Register

The Stack

it is slowly going down everytime we do a push and the heap keeps going up

PUSH - Decrements ESP(stack pointer and moves data to that location)

POP - moves the ESP value and increment the stack

CALL- Pushes the EIP and JMPS to the address

RET - JMPS to the return address which was pushed to the stack during the call instruaction

Endian

Little Endian : stored in RAM "little end" first. The least significant byte of a word or larger is stored in the lower address.

Big Endian: Here it is saved as it is.

Next is Calling convention

two types :

cdecl

push reverse order parameters

Caller is responsible for cleaning up the stack

the next command after the call will be to clean the stack

stdcall

Push in reverse order

Here is callee is responsible for cleaning up the stack

so at the end of the call we will have some pop statements

open illusionbot in IDApro

and open the BOTbinary.exe

it shows that sp analysis failed

we can that there is no stack initialization is done

but there are some pushes done and some address is loaded into the eax and it is pushed to the stack

now the retn is executed

from this we can see that lot of things are pushed to the stack

but it is not popped so when the last loaded address is used as the next instruction pointer address from which it should be continued.

which is a first sign of something being wrong

now when we are in the IDA view we can see that there is a function called CreateMutex which when googled shows its a sign of malware.

So this can be passed over to the SOCK team and they develop a kind of patch for that

now we open the mutex and see whats happening

to prevent disassemblers from identifying calls we

mov dword\_382423, eax

call dword\_382423

So what we get in this is to look for IoCs

See for code obfuscations that might be done to hide some mailcious content

**Why do we perform static analysis ?**

Understanding the Malware

Discover Indicators of Compromise

Confirm Dynamic Analysis

Discover Anti-Debugging code

**Tricks used by Malware so far:**

Stack Corruption

Import Hiding

Dynamic function Resolving

String Obfuscation

Dealing with Obfuscated String

> Fully reverse engineer the code re-implement it then apply the same process to the string

>Easy way is to use the native code

**PACKERS:**

They are self-decrypting executables

Which makes it hard for malware analysts

Cause the way it works is that to deal with compressing code size developers introduced packers which basically compresses the code and creates the executable and when that it run it decompresses into the memory and then it is further executed.

Now the problem with that it will be difficult to view imports or strings from tools like IDA pro.

But this will act as an advantage for malware authors.

To look for Indicator of packers present:

We can see strings like

UPX or Mystic compressor or something like that which are indicator of packers being present

**Goals :**

Hide strings

Change the hash

Mask binary signatures

**Why doesn’t anti malwares flag this ?**

Cause it is also used for :

Code compression(to prevent large file sizes)

Intellectual Property Protection(to prevent keygen algorithms and other specialized code to see that their algo is hided)

Anti-Reverse Enineering

Anti-Cheat (prevent video game softwares from being altered)

Digital Rights Management

**Malware Defenses:**

Anti-debugging :

API

Process and Thread

Hardware and register based

Timing based

Code Based

Anti-Virtual Machine

API

Memeory Constants

File/Process Names

Anti-Disassembly

Anti-Analysis tool

Sometimes , We run the malware and see the CaptureBAT log it dosent show anything

Then we also run wireshark on that interface but nothing really happens

so we open the malware in IDA pro go to view open>subviews>strings

we see a string called sanboxing (which are malware analysis autmated solutions) we can see from that static analysis from there is function written in the malware which would read out all the running function names in the OS and compares it to the string and if it finds something to be equal then it will terminate

we can see that it checks for virtual\_vbox check

Some Anti-debugging techniques

IsBeingDebugged() // to mitigate this use a hardware breakpoint what basically happens is that it searches for some string so we just set a breakpoint

CheckRemoteDebuggerPresent()

FindWindow()

NtqueryInformationProcess(ProcessDebugFlags)