Malware Analysis

WIST KE

Whoami?

ALIEN-WITHIN

Humanizing infosec in a cruel world

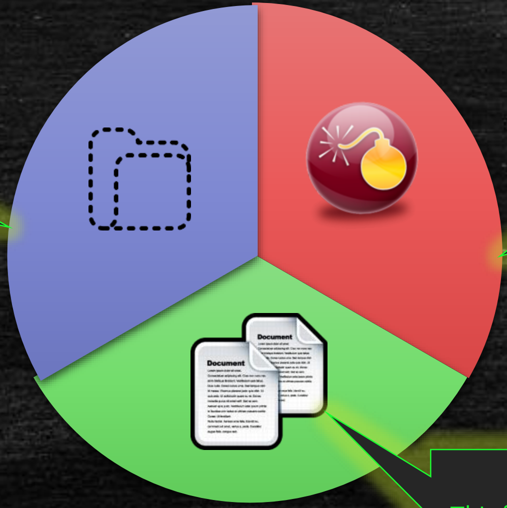
Munir is an Cyber security consultant with over 8 years experience in a wide variety of Information Security related projects. Munir has lead various security assessments in the financial sevices and telecommunication sectors; He mainly focuses on malware analysis, web and mobile based applications testing and methodologies. He is a member of the Africahackon team which is East Africa's premier technical computer security collective that brings together the individual talents of the best and brightest security professionals in the region, through live presentations, engaging discussions and hands on demonstrations.

Munir is also a project leader for the OWASP Project dubbed "OWASP Mth3l3m3nt Framework" which is an exploitation framework that aids in a number of activities and uses minimal resources as all it needs is a webserver which can even be run from an android phone without a problem and optionally a database server. Munir is also the leader of the OWASP Kenyan chapter which focuses on Secure Software development and deployment standards. He has published a course on malware analysis with Packtub Publishing and is also a technical writer for pentest magazine as well as hackin9 magazine.   Some of the methodologies Munir is well versed in include:

- OWASP (Open Web Application Security Project)Top 10
- SAMM (Software Assurance Maturity Model)
- OSSTMM (Open Source Security Testing methodology Manual)

What is Malware

Malware is a broad term that refers to destructive software or that which has malicious intent. Malware is generally composed of 3 main components which also determine how malware are classified:



This feature keeps malware hidden it may be a complex or simple feature.

This feature or Function causes actual damage or defines the true intent of the malware in regard to a target device

This feature enables malware propagate.

What is Malware Analysis

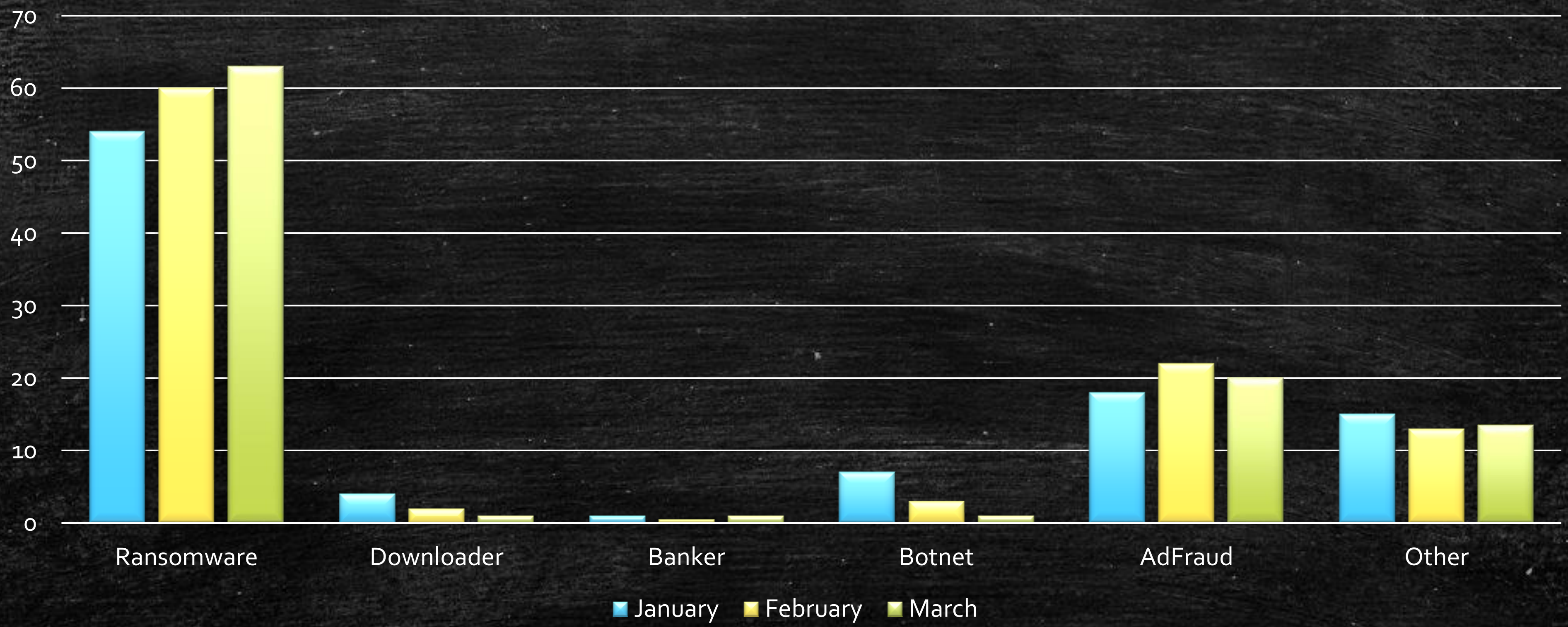
Malware analysis can be said to be an investigative process that is aimed at getting inside knowledge of how malicious software works and its intent on a system.

Get signatures

Understand Functionality

Classify Malware

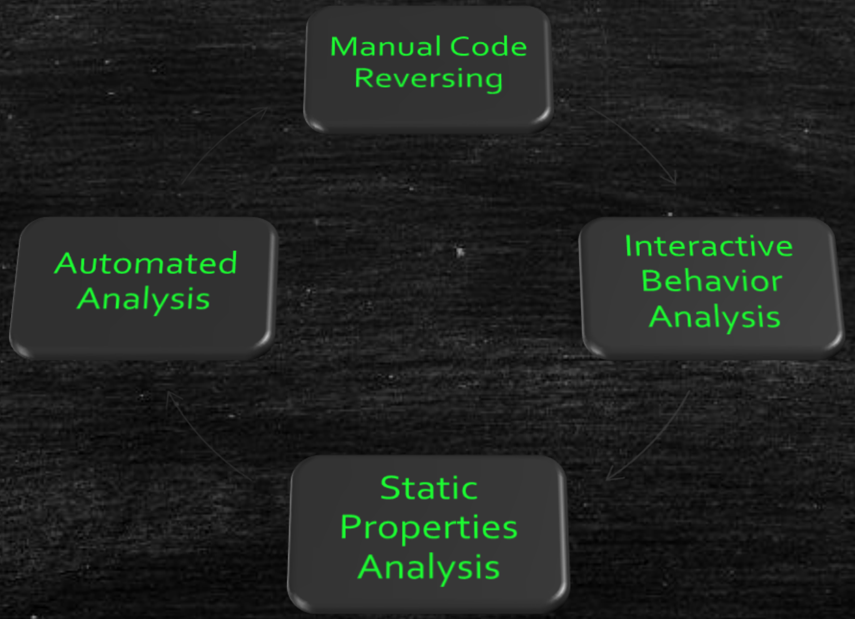
Malware infection statistics Q1 2017



Source: Malware Bytes

Malware Analysis Methodology

There are 4 general stages in malware analysis and they are stated below:



Static Analysis

Static Analysis

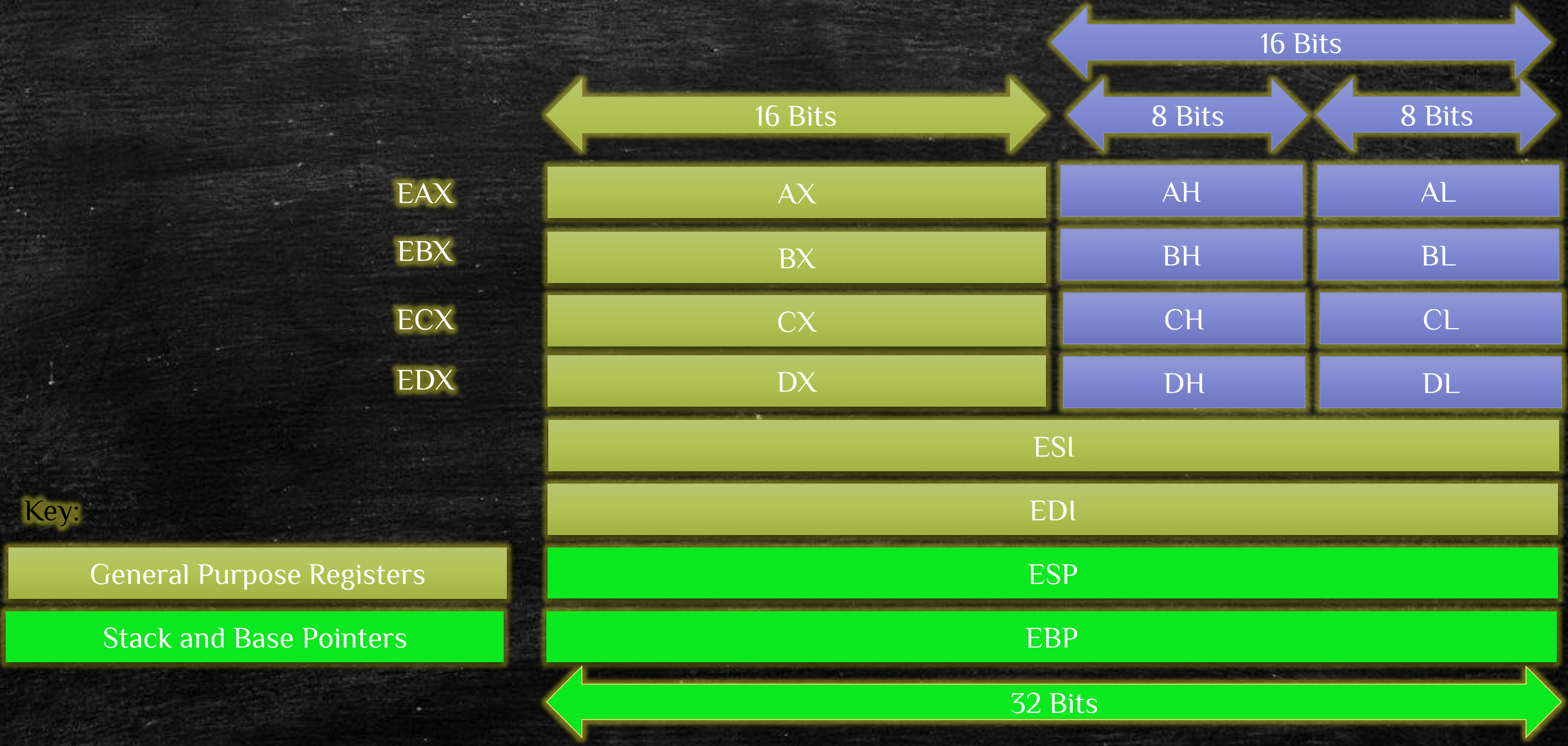
Static Analysis – This is a process that studies general attributes of any application/software; It doesn't involve execution. Some of the key activities under this include:

- Decompiling
- Static PE properties analysis
- Analysis of system calls
- Analysis of strings

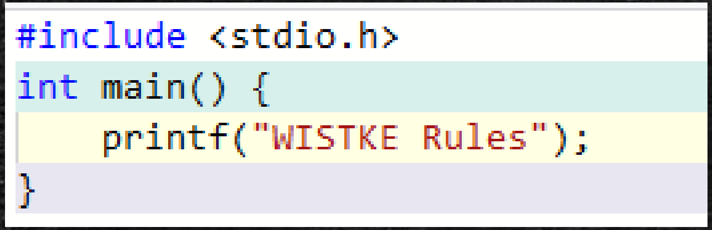
ABCD of x86

- The x86 architecture forms the basis of modern computing ; There are some key elements that comprise the x86 architecture:
- Registers – bits of memory i.e. memory cells within the CPU used to perform tasks
  - **A**ccumulator Registers
  - **B**ase Registers
  - **C**ontrol Registers
  - **D**ata Registers

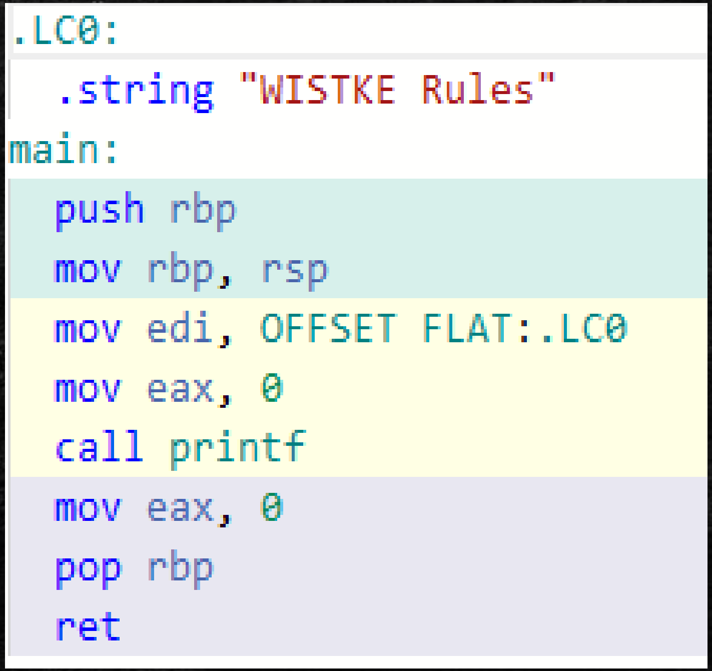
What’s the E in x86 Architecture?



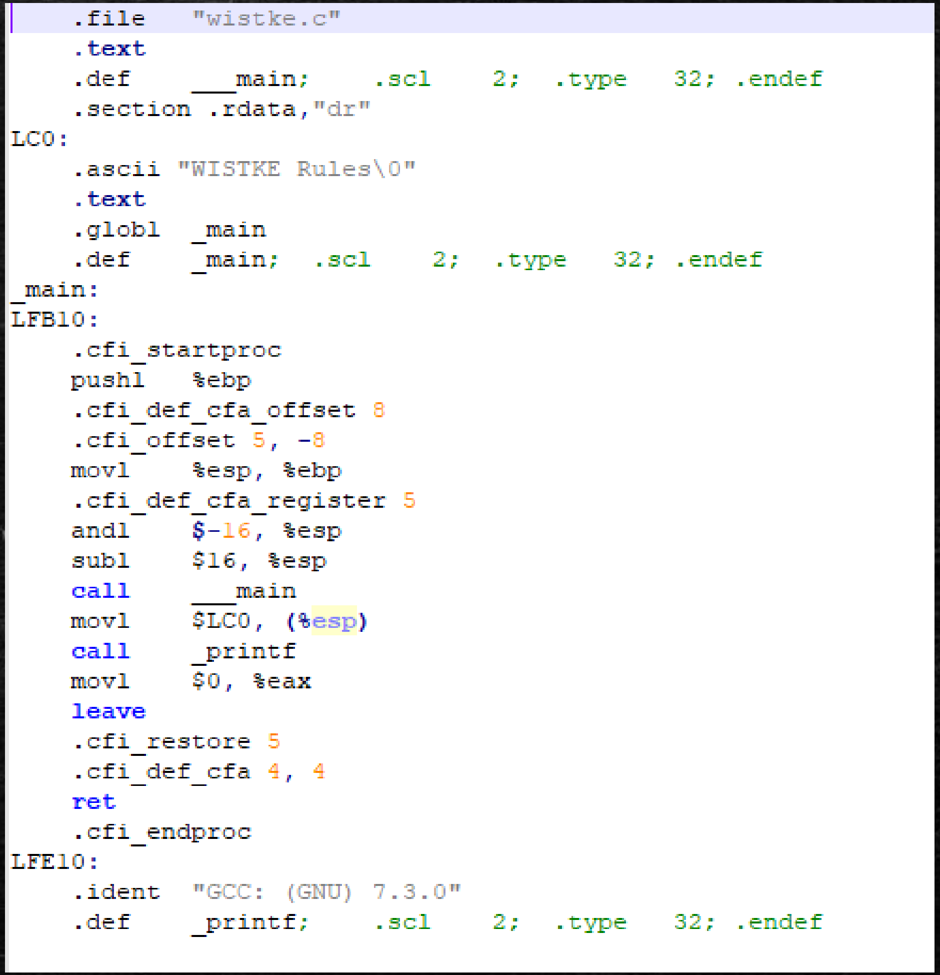
How do they apply?



```c
#include <stdio.h>
int main() {
    printf("WISTKE Rules");
}
```

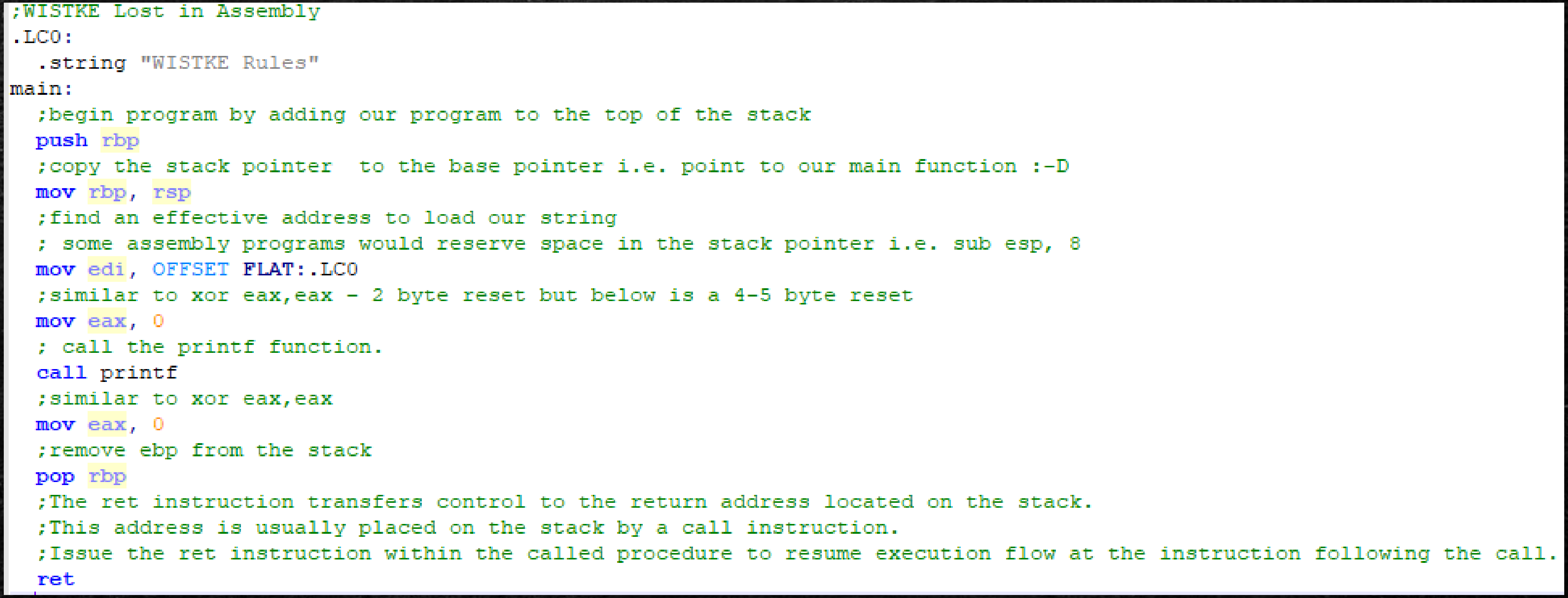


```
.LC0:
  .string "WISTKE Rules"
main:
  push rbp
  mov rbp, rsp
  mov edi, OFFSET FLAT:.LC0
  mov eax, 0
  call printf
  mov eax, 0
  pop rbp
  ret
```



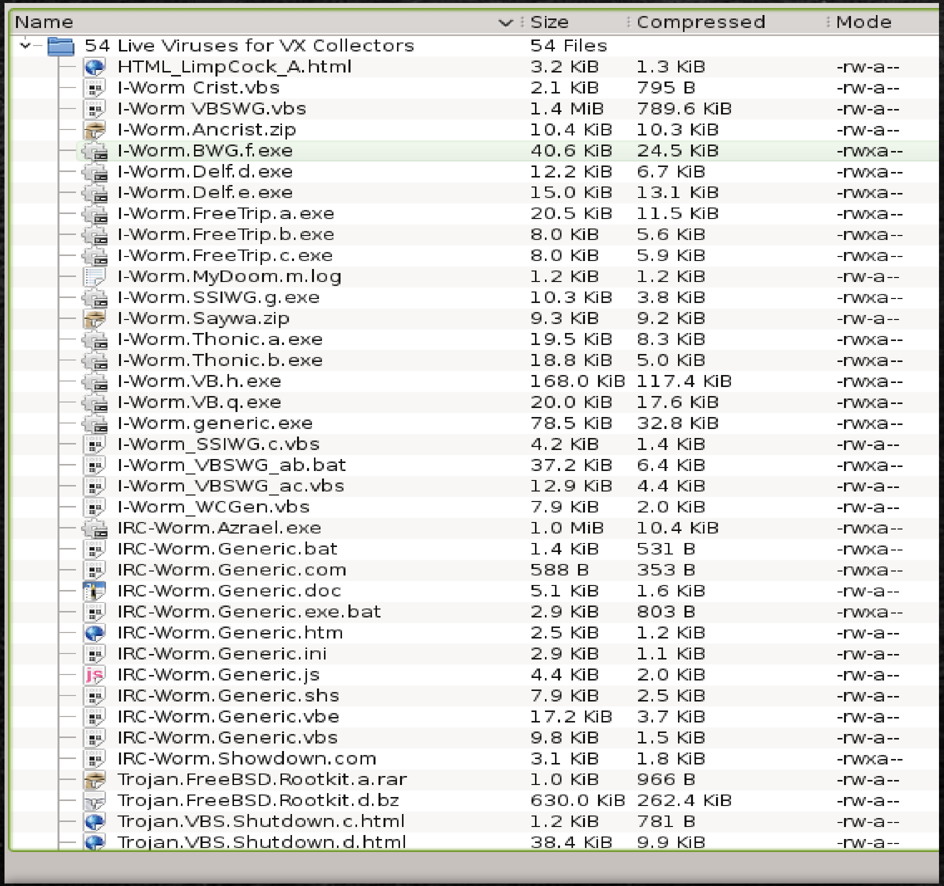
```
	.file	"wistke.c"
	.text
	.def	___main;	.scl	2;	.type	32;	.endef
	.section .rdata,"dr"
LC0:
	.ascii "WISTKE Rules\0"
	.text
	.globl	_main
	.def	_main;	.scl	2;	.type	32;	.endef
_main:
LFB10:
	.cfi_startproc
	pushl	%ebp
	.cfi_def_cfa_offset 8
	.cfi_offset 5, -8
	movl	%esp, %ebp
	.cfi_def_cfa_register 5
	andl	$-16, %esp
	subl	$16, %esp
	call	___main
	movl	$LC0, (%esp)
	call	_printf
	movl	$0, %eax
	leave
	.cfi_restore 5
	.cfi_def_cfa 4, 4
	ret
	.cfi_endproc
LFE10:
	.ident	"GCC: (GNU) 7.3.0"
	.def	_printf;	.scl	2;	.type	32;	.endef
```

Human View of Assembly



```
;WISTKE Lost in Assembly
.LC0:
  .string "WISTKE Rules"
main:
  ;begin program by adding our program to the top of the stack
  push rbp
  ;copy the stack pointer  to the base pointer i.e. point to our main function :-D
  mov rbp, rsp
  ;find an effective address to load our string
  ; some assembly programs would reserve space in the stack pointer i.e. sub esp, 8
  mov edi, OFFSET FLAT:.LC0
  ;similar to xor eax,eax - 2 byte reset but below is a 4-5 byte reset
  mov eax, 0
  ; call the printf function.
  call printf
  ;similar to xor eax,eax
  mov eax, 0
  ;remove ebp from the stack
  pop rbp
  ;The ret instruction transfers control to the return address located on the stack.
  ;This address is usually placed on the stack by a call instruction.
  ;Issue the ret instruction within the called procedure to resume execution flow at the instruction following the call.
  ret
```

File Formats



Name	Size	Compressed	Mode
54 Live Viruses for VX Collectors	54 Files		
HTML_LimpCock_A.html	3.2 KiB	1.3 KiB	-rw-a--
I-Worm Crist.vbs	2.1 KiB	795 B	-rw-a--
I-Worm VBSWG.vbs	1.4 MiB	789.6 KiB	-rw-a--
I-Worm.Ancrist.zip	10.4 KiB	10.3 KiB	-rw-a--
I-Worm.BWG.f.exe	40.6 KiB	24.5 KiB	-rwxa--
I-Worm.Delf.d.exe	12.2 KiB	6.7 KiB	-rwxa--
I-Worm.Delf.e.exe	15.0 KiB	13.1 KiB	-rwxa--
I-Worm.FreeTrip.a.exe	20.5 KiB	11.5 KiB	-rwxa--
I-Worm.FreeTrip.b.exe	8.0 KiB	5.6 KiB	-rwxa--
I-Worm.FreeTrip.c.exe	8.0 KiB	5.9 KiB	-rwxa--
I-Worm.MyDoom.m.log	1.2 KiB	1.2 KiB	-rw-a--
I-Worm.SSIWG.g.exe	10.3 KiB	3.8 KiB	-rwxa--
I-Worm.Saywa.zip	9.3 KiB	9.2 KiB	-rw-a--
I-Worm.Thonic.a.exe	19.5 KiB	8.3 KiB	-rwxa--
I-Worm.Thonic.b.exe	18.8 KiB	5.0 KiB	-rwxa--
I-Worm.VB.h.exe	168.0 KiB	117.4 KiB	-rwxa--
I-Worm.VB.q.exe	20.0 KiB	17.6 KiB	-rwxa--
I-Worm.generic.exe	78.5 KiB	32.8 KiB	-rwxa--
I-Worm_SSIWG.c.vbs	4.2 KiB	1.4 KiB	-rw-a--
I-Worm_VBSWG_ab.bat	37.2 KiB	6.4 KiB	-rwxa--
I-Worm_VBSWG_ac.vbs	12.9 KiB	4.4 KiB	-rw-a--
I-Worm_WCGen.vbs	7.9 KiB	2.0 KiB	-rw-a--
IRC-Worm.Azrael.exe	1.0 MiB	10.4 KiB	-rwxa--
IRC-Worm.Generic.bat	1.4 KiB	531 B	-rwxa--
IRC-Worm.Generic.com	588 B	353 B	-rwxa--
IRC-Worm.Generic.doc	5.1 KiB	1.6 KiB	-rw-a--
IRC-Worm.Generic.exe.bat	2.9 KiB	803 B	-rwxa--
IRC-Worm.Generic.htm	2.5 KiB	1.2 KiB	-rw-a--
IRC-Worm.Generic.ini	2.9 KiB	1.1 KiB	-rw-a--
IRC-Worm.Generic.js	4.4 KiB	2.0 KiB	-rw-a--
IRC-Worm.Generic.shs	7.9 KiB	2.5 KiB	-rw-a--
IRC-Worm.Generic.vbe	17.2 KiB	3.7 KiB	-rw-a--
IRC-Worm.Generic.vbs	9.8 KiB	1.5 KiB	-rw-a--
IRC-Worm.Showdown.com	3.1 KiB	1.8 KiB	-rwxa--
Trojan.FreeBSD.Rootkit.a.rar	1.0 KiB	966 B	-rw-a--
Trojan.FreeBSD.Rootkit.d.bz	630.0 KiB	262.4 KiB	-rw-a--
Trojan.VBS.Shutdown.c.html	1.2 KiB	781 B	-rw-a--
Trojan.VBS.Shutdown.d.html	38.4 KiB	9.9 KiB	-rw-a--

Malware comes in various forms ; this determines how to handle analysis of the malware. Examples of formats malware is distributed in include:

- Obfuscated Scripts
- Documents
- Self extracting archives
- Portable executables e.g. The hexadecimal representation for the start of the PE format ("MZ") would be 4D 5A. Underneath this should be bytes for things relating to DOS (e.g. "This program cannot be run in DOS mode"

Static Analysis

Test Static Super power

Dynamic Analysis

Malware Analysis Types – Dynamic Analysis

- Dynamic Analysis – This is the study of the behaviour of software to determine if it is malicious or not. Some of the activities performed include:

Network analysis

File system modification

Registry analysis

Memory analysis

Techniques

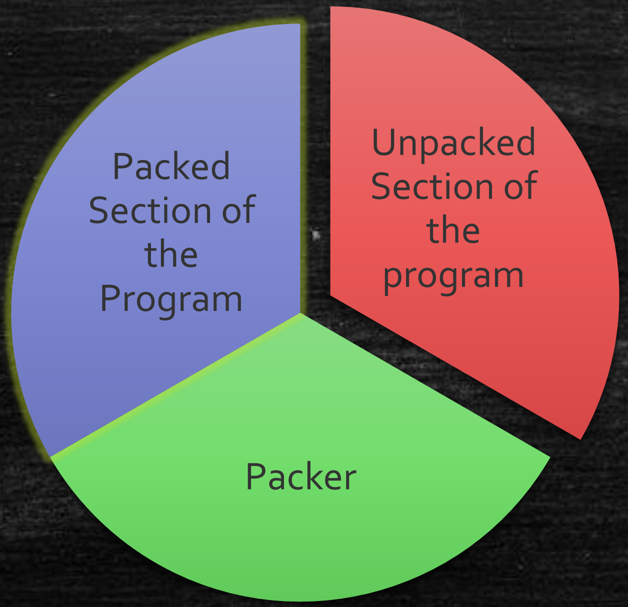
- Sandboxing
- Process monitoring
- Registry Snapshots
- Packet Sniffing and network monitoring tools
- Network Service spoofing tools

Packed Malware

- Packers are software that apply different compression techniques; in modern day packers some extra features include code protection and execution through virtual environments. The main aim in a malware context is to hide intent and make it harder to analyze the sample. Some examples are:
- UPX, themida, PECompact, ASPack, Kkrunchy and WWPack32

How they work

- Packers exist as self extracting archives ; If malware is fully packed it can’t infect the computer until it is unpacked; Malware developers have come up with runtime packers that only pack a partial set of the program.


Packed Section of the Program
Unpacked Section of the program
Packer

Dynamic Analysis

Test Static Super power

Thank You

Questions