Internet 蠕虫传播模型分析与模拟

2021.03.15

1 介绍和定义

1.1 介绍

在 Internet 中，由于传输层上 TCP/IP 协议对下层结构的屏蔽，在应用层进程通信的角度看可以认为是一个完全链接网络，而蠕虫传播模型由于其以计算机系统为攻击对象和主动传播的特性，与传染病模型在数学理论上具有很大的相似性和契合度。

本文讨论不同传播模型对不同条件的适应性和合理性，第二部分分析了模型的数学原理与物理意义，第三部分使用 python matplotlib 库进行模拟并分析相应结果，涉及讨论的模型包括 SI 模型（Susceptible-Infective Model）、SIS 模型（Susceptible-Infectious-Susceptible Model）、SIR 模型（Susceptible-Infectious-Recovered Model）、SEIR 模型（Susceptible- Exposed-Infectious-Recovered Model）及其相关变形。

1.2 相关定义

在时间戳 t 上，计算机分类种类定义如下：

- **SUSCEPTIBLE**：易感者，潜在的可感染计算机
- **EXPOSED**：潜伏者，已经被感染但是没有表现出来的计算机
- **INFECTIVE**：感染者，表现出感染症状的计算机
- **RESISTANCE**：抵抗者，感染者痊愈后获得抗性的计算机

总计算机数为 $N(t)$，其中有 $N(t) = S(t) + E(t) + I(t) + R(t)$。

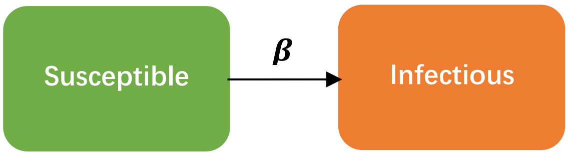
对于其他相关参数，有定义如下：

- α为系统存在的常数输入率
- μ 表示各类的自然淘汰率
- γ 表示病毒激发率
- β为传染率系数

2 现有模型分析

2.1 SI 模型

只考虑感染者和易感者，易感染者有一定概率被感染，而感染者感染后不能恢复，拓扑结构中计算机总数有一定的增长率，增长的计算机都归在易感者中。

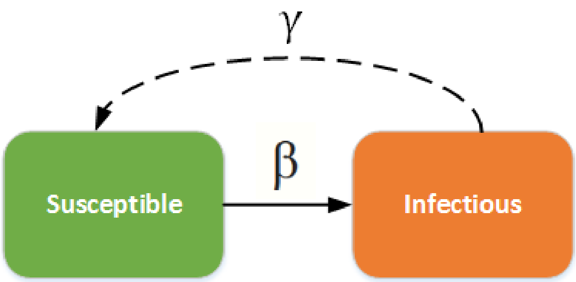


数学模型：

$$\frac{dS}{dt} = \alpha N - \frac{\beta SI}{N}$$
$$\frac{dI}{dt} = \frac{\beta SI}{N}$$

2.2 SIS 模型

只考虑感染者和易感染者，易感染者有一定概率被感染，而感染者感染后可以恢复，以一定概率恢复为易感者，拓扑结构中计算机总数有一定的增长率，增长的计算机都归在易感者中。



数学模型：

$$\frac{dS}{dt} = \alpha N - \frac{\beta SI}{N} + \gamma I$$
$$\frac{dI}{dt} = \frac{\beta SI}{N} - \gamma I$$

2.3 SIR 模型

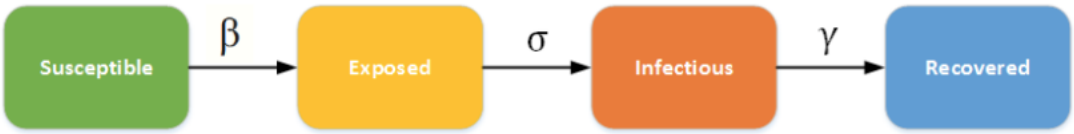
引入 Recovered 或 Resistance 群体，表示被感染后恢复后对该病毒产生针对性的应对措施（例如杀毒工具对该病毒特征码进行录入），或是本身就具有抵抗性（例如操作系统不同，而该病毒不具有跨操作系统性）。于是有易感染者有一定概率被感染，而感染者感染后可以恢复，以一定概率恢复为恢复者，拓扑结构中计算机总数有一定的增长率，增长的计算机都归在易感者中。

数学模型：

$$\frac{dS}{dt} = \alpha N - \frac{\beta SI}{N}$$
$$\frac{dI}{dt} = \frac{\beta SI}{N} - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

2.4 SEIR 模型

引入潜伏期，由于病毒具有一定的触发条件，往往不是在感染后立即发生作用，因此潜伏期更符合实际应用情况。易感者以一定概率变为潜伏者，潜伏者以一定概率变为感染者，感染者有一定概率变为恢复者，拓扑结构中计算机总数有一定的增长率，增长的计算机都归在易感者中。



数学模型：

$$\frac{dS}{dt} = \alpha N - \frac{\beta SI}{N}$$
$$\frac{dE}{dt} = \frac{\beta SI}{N} - \sigma E$$
$$\frac{dI}{dt} = \sigma E - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

3 SEIR 模型的改进设计

在计算机安全的实际应用场景下，SEIR 模型的设定显然是比较理想化的，四种状态之间的转化相对比较简单。现通过对病毒的实际行为的模拟分析，重新引入如下概念，以使模型与现实病毒的行为与应用场景更加契合：

1. 拓扑结构中新增的主机中有一部分为抵抗者，例如针对 windows 下的病毒，新增 linux 系统主机天生对该病毒具有抵抗性，这里假设新增主机对该病毒具有抵抗性的概率为p。

   数学模型：

$$\frac{dS}{dt} = \alpha(1-p)N - \frac{\beta SI}{N}$$
$$\frac{dE}{dt} = \frac{\beta SI}{N} - \sigma E$$
$$\frac{dI}{dt} = \sigma E - \gamma I$$

$$\frac{dR}{dt} = \gamma I + \alpha p N$$

2. 存在一定的主机报废率，这里假设为μ。

数学模型：

$$\frac{dS}{dt} = \alpha(1-p)N - \frac{\beta SI}{N} - \mu S$$
$$\frac{dE}{dt} = \frac{\beta SI}{N} - \sigma E - \mu E$$
$$\frac{dI}{dt} = \sigma E - \gamma I - \mu I$$
$$\frac{dR}{dt} = \gamma I + \alpha p N - \mu R$$

3. 病毒在潜伏期仍然应当具有传染性，实际上 E to I 的过程为触发过程。

数学模型：

$$\frac{dS}{dt} = \alpha(1-p)N - \frac{\beta S(I+E)}{N} - \mu S$$
$$\frac{dE}{dt} = \frac{\beta S(I+E)}{N} - \sigma E - \mu E$$
$$\frac{dI}{dt} = \sigma E - \gamma I - \mu I$$
$$\frac{dR}{dt} = \gamma I + \alpha p N - \mu R$$

4. E 和 S 都有可能通过用户的防御措施而加强，这里令防护措施实施率在 S 和 E 上分别为k和l。

数学模型：

$$\frac{dS}{dt} = \alpha(1-p)N - \frac{\beta S(I+E)}{N} - \mu S - kS$$
$$\frac{dE}{dt} = \frac{\beta S(I+E)}{N} - \sigma E - \mu E - lE$$
$$\frac{dI}{dt} = \sigma E - \gamma I - \mu I$$
$$\frac{dR}{dt} = \gamma I + \alpha p N - \mu R + kS + lE$$

5. 防护不具有彻底性，具有不完全实施率。从 I 和 E 中得到防御或加强而不再受病毒威胁的主机中并不是所有都成为 R，应当有一定比例变为 R 一定比例变为 S，假设这两部分不完全实施的比例（转化到 S）分别为η和ω。

数学模型：

$$\frac{dS}{dt} = \alpha(1-p)N - \frac{\beta S(I+E)}{N} - \mu S - kS + \eta\gamma I + \omega lE$$
$$\frac{dE}{dt} = \frac{\beta S(I+E)}{N} - \sigma E - \mu E - lE$$
$$\frac{dI}{dt} = \sigma E - \gamma I - \mu I$$

$$\frac{dR}{dt} = (1-\eta)\gamma I + \alpha p N - \mu R + kS + (1-\omega) l E$$

4 模拟

4.0 初始化参数

```
N = 1e5         # Total number of computer hosts in the network topology
T = 200         # Simuation time / Day
```

这里假设网络中计算机主机数为 10000 台，时间为 200 天，起始感染计算机数为 1。

4.1 SI 模型

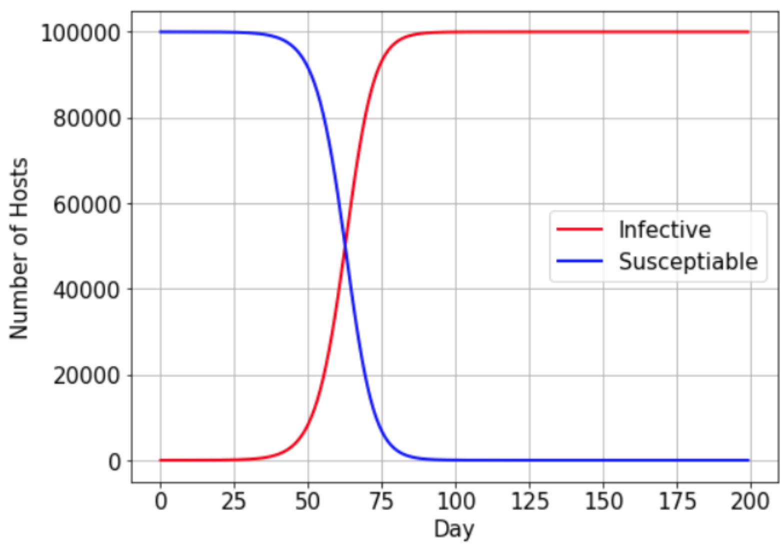
代码：

```
s = np.zeros([T])       # susceptiable
i = np.zeros([T])       # infective
beta = 0.2              # contact rate
alpha = 0               # growth rate
i[0] = 1                # initialize infective
s[0] = N - i[0]         # initialize susceptiable

for t in range(T-1):
    s[t + 1] = s[t] + alpha * N - beta * s[t] * i[t] / N
    i[t + 1] = i[t] + beta * s[t] * i[t] / N
```

```
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(i, c='r', lw=2, label='Infective')
ax.plot(s, c='b', lw=2, label='Susceptiable')
ax.set_xlabel('Day',fontsize=15)
ax.set_ylabel('Number of Hosts', fontsize=15)
ax.grid(1)
plt.legend(fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15);
```

模拟结果：



4.2 SIS 模型

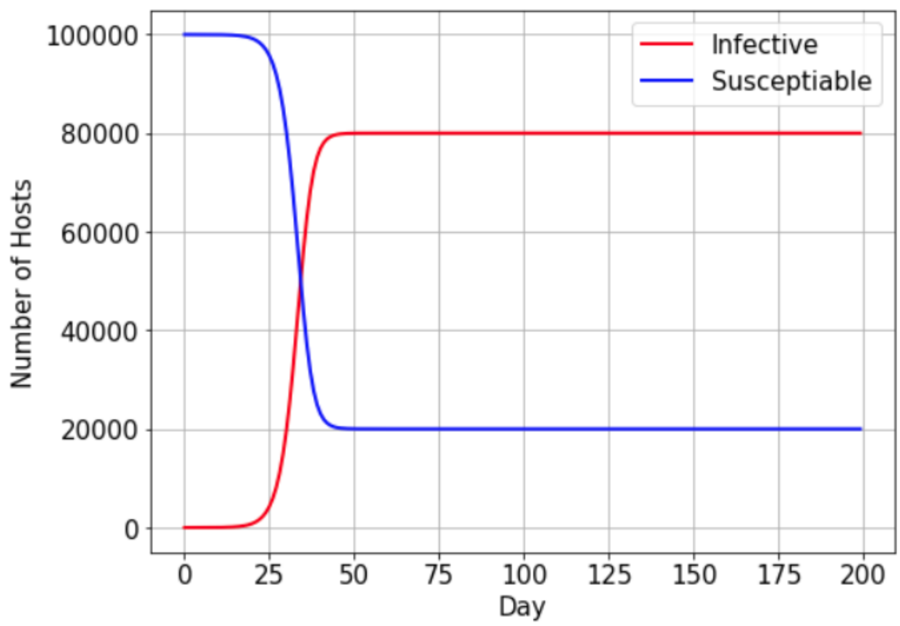
代码：

```
s = np.zeros([T])      # susceptiable
i = np.zeros([T])      # infective
beta = 0.5             # contact rate
alpha = 0              # growth rate
gamma = 0.1            # recovery rate
i[0] = 1               # initialize infective
s[0] = N - i[0]        # initialize susceptiable

for t in range(T-1):
    s[t + 1] = s[t] + alpha * N - beta * s[t] * i[t] / N + gamma * i[t]
    i[t + 1] = i[t] + beta * s[t] * i[t] / N - gamma * i[t]
```

```
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(i, c='r', lw=2, label='Infective')
ax.plot(s, c='b', lw=2, label='Susceptiable')
ax.set_xlabel('Day',fontsize=15)
ax.set_ylabel('Number of Hosts', fontsize=15)
ax.grid(1)
plt.legend(fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15);
```

模拟结果：



4.3 SIR 模型

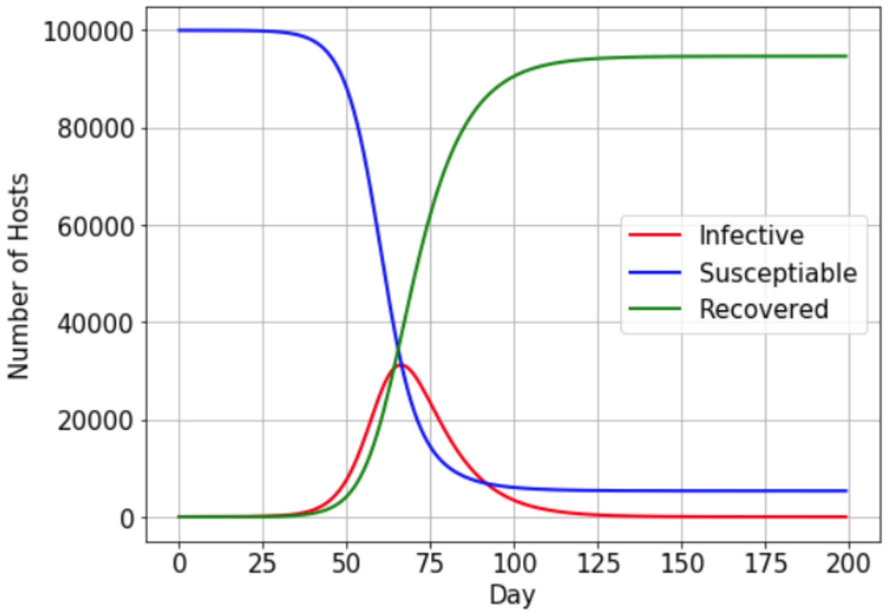
代码：

```
s = np.zeros([T])       # susceptiable
i = np.zeros([T])       # infective
r = np.zeros([T])       # recovered
beta = 0.3              # contact rate
alpha = 0               # growth rate
gamma = 0.1             # recovery rate
i[0] = 1                # initialize infective
s[0] = N - i[0]         # initialize susceptiable
r[0] = 0                # initialize recovered

for t in range(T-1):
    s[t + 1] = s[t] + alpha * N - beta * s[t] * i[t] / N
    i[t + 1] = i[t] + beta * s[t] * i[t] / N - gamma * i[t]
    r[t + 1] = r[t] + gamma * i[t]
```

```
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(i, c='r', lw=2, label='Infective')
ax.plot(s, c='b', lw=2, label='Susceptiable')
ax.plot(r, c='g', lw=2, label='Recovered')
ax.set_xlabel('Day',fontsize=15)
ax.set_ylabel('Number of Hosts', fontsize=15)
ax.grid(1)
plt.legend(fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15);
```

模拟结果：



4.4 SEIR 模型

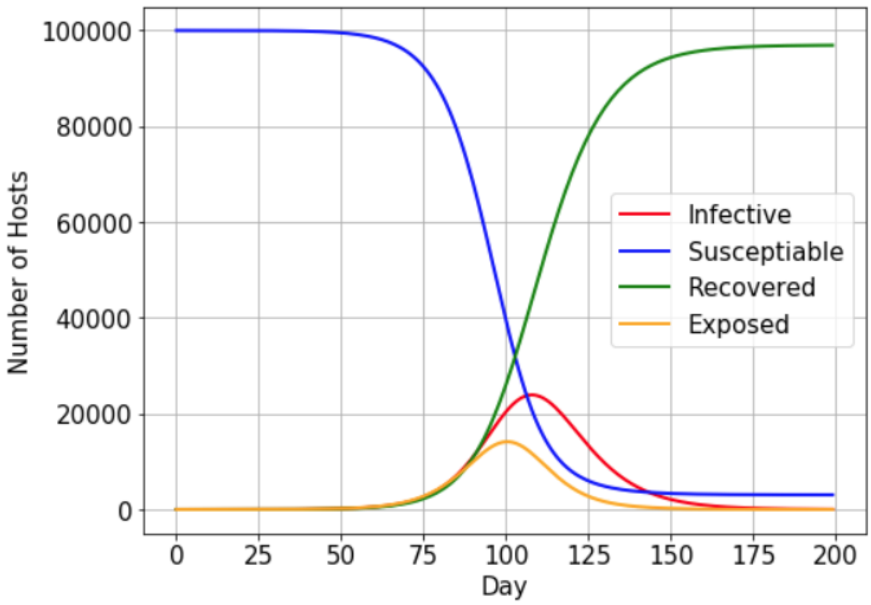
代码：

```
s = np.zeros([T])      # susceptiable
i = np.zeros([T])      # infective
r = np.zeros([T])      # recovered
e = np.zeros([T])      # exposed
beta = 0.35             # contact rate
alpha = 0              # growth rate
gamma = 0.1            # recovery rate
delta = 0.2            # trigger rate
i[0] = 1               # initialize infective
s[0] = N - i[0]        # initialize susceptiable
r[0] = 0               # initialize recovered
e[0] = 0               # initialize exposed

for t in range(T-1):
    s[t + 1] = s[t] + alpha * N - beta * s[t] * i[t] / N
    e[t + 1] = e[t] + beta * s[t] * i[t] / N - delta * e[t]
    i[t + 1] = i[t] + delta * e[t] - gamma * i[t]
    r[t + 1] = r[t] + gamma * i[t]
```

```
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(i, c='r', lw=2, label='Infective')
ax.plot(s, c='b', lw=2, label='Susceptiable')
ax.plot(r, c='g', lw=2, label='Recovered')
ax.plot(e, c='orange', lw=2, label='Exposed')
ax.set_xlabel('Day',fontsize=15)
ax.set_ylabel('Number of Hosts', fontsize=15)
ax.grid(1)
plt.legend(fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15);
```

模拟结果：



4.5 SEIR 改进模型

代码：

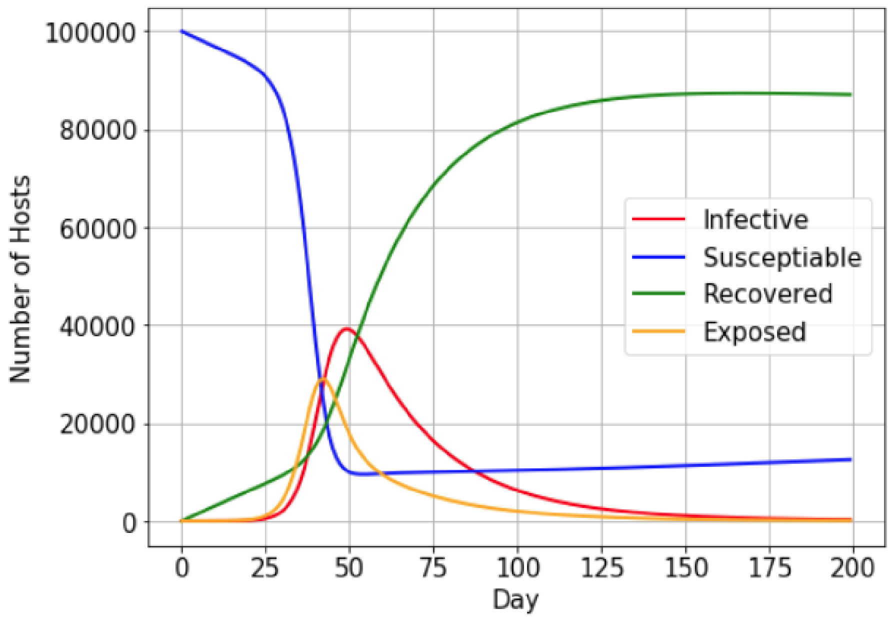
```
s = np.zeros([T])          # susceptiable
i = np.zeros([T])          # infective
r = np.zeros([T])          # recovered
e = np.zeros([T])          # exposed
beta = 0.40                # contact rate
alpha = 0.001              # growth rate
gamma = 0.1                # recovery rate
delta = 0.2                # trigger rate
p = 0.2                    # initially resisted rate
mu = 0.001                 # scrap rate
k = 0.003                  # Implementation rate of protective measures on S
l = 0.005                  # Implementation rate of protective measures on E
eta = 0.5                  # Ratio of I to S
omega = 0.5                # Ratio of E to S

i[0] = 1                   # initialize infective
s[0] = N - i[0]            # initialize susceptiable
r[0] = 0                   # initialize recovered
e[0] = 0                   # initialize exposed

for t in range(T-1):
    s[t + 1] = s[t] + (1 - p) * alpha * N - beta * s[t] * (i[t] + e[t]) / N - mu * s[t] - k * s[t]
    s[t + 1] += eta * gamma * i[t] + omega * l * e[t]
    e[t + 1] = e[t] + beta * s[t] * (i[t] + e[t]) / N - delta * e[t] - mu * e[t] - l * e[t]
    i[t + 1] = i[t] + delta * e[t] - gamma * i[t] - mu * i[t]
    r[t + 1] = r[t] + (1 - eta) * gamma * i[t] + alpha * p * N - mu * r[t] + k * s[t] + (1 - omega) * l * e[t]
```

```
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(i, c='r', lw=2, label='Infective')
ax.plot(s, c='b', lw=2, label='Susceptiable')
ax.plot(r, c='g', lw=2, label='Recovered')
ax.plot(e, c='orange', lw=2, label='Exposed')
ax.set_xlabel('Day',fontsize=15)
ax.set_ylabel('Number of Hosts', fontsize=15)
ax.grid(1)
plt.legend(fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15);
```

模拟结果：



5 参考文献

[1] 刘功申，孟魁，王轶骏，姜开达，李生红．计算机病毒与恶意代码——原理、技术及防范[M]．第四版．北京:清华大学出版社, 2019:42-43.

[2]刘建芳，刘启明，刘立红．计算机蠕虫病毒传播的数学模型分析[J]．数学的实践与认识, 2008, 38(12):49-52.

[3]关于传染病的数学模型有哪些？www.zhihu.com/question/367466399

[4]陈永雪.一类计算机病毒传播模型的数学分析[J].福建农林大学学报(自然科学版),2014,43(05):551-555.

[5]张友声，米安然.计算机病毒与木马程序剖析[M].北京:北京科海电子出版社, 2003.

[6]冯丽萍，王鸿斌，冯素琴.基于生物学原理的计算机网络病毒传播模型[J].计算机工程, 2011, 37 (11) :155-157.

[7]KEPHART J O, ORKIN G B, CHESS D M, et al.Fighting computer viruses[J].Computer and Security, 1997, 16 (8) :676-677.