ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

****

****

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**AN TOÀN MẠNG**

**Hierarchical reinforcement learning for efficient and effective automated penetration testing of large networks**

**GIẢNG VIÊN HƯỚNG DẪN: ThS. NGHI HOÀNG KHOA**

**LỚP: NT140.P11.ANTT - NHÓM 17:**

|  |  |  |
| --- | --- | --- |
| **Hà Minh Quân** | **22521177** | **Trưởng nhóm** |
| **Nguyễn Thanh Hưng** | **22520519** | **Thành viên** |
| **Từ Chí Kiên** | **22520713** | **Thành viên** |

***TP. Hồ Chí Minh, tháng 1 năm 2025***

**Lời Cảm Ơn**

Lời đầu tiên, Nhóm em gửi lời cảm ơn tới thầy Nghi Hoàng Khoa , cảm ơn thầy vì những bài học hữu ích trên lớp và những lời dặn dò của thầy , đã giúp cho chúng em thêm phần hiểu biết về ngành an toàn thông tin này , và cuối cùng nhóm em xin chúc thầy mạnh khỏe và thành công hơn trong công việc và cuộc sống. Nhóm chúng em xin chân thành cảm ơn.

*Thành phố Hồ Chí Minh, ngày 12 tháng 12 năm 2024*

**Mục Lục**

[**Chương 1: Tổng quan 2**](#_heading=h.gelg8rhex4pv)

[1.1 Ngữ cảnh thực tế 2](#_heading=h.n2462u1mt69r)

[1.2. Cơ sở lý thuyết 2](#_heading=h.4vwp3ark9m5)

[1.2.1. Khái niệm Penetration Testing 2](#_heading=h.j75s7xba6fb1)

[1.2.2. Khái niệm Reinforcement Learning 2](#_heading=h.rnczs9qveasi)

[1.2.3. Hierarchical reinforcement learning vào xâm nhập kiểm thử tự động 3](#_heading=h.2g2l9lv2sn7u)

[**Chương 2: Mục tiêu nghiên cứu 3**](#_heading=h.wbva3xiz3lt0)

[**Chương 3: Giải Quyết Vấn Đề Của POMDP. 3**](#_heading=h.k5amtsf5a3pm)

[3.1. Định Nghĩa Về POMDP. 3](#_heading=h.3bacmle6g41l)

[3.2. Các Thành Phần Cơ Bản Của POMDP. 3](#_heading=h.xwoaeh1gxhd0)

[3.3. Các Hàm Của Mô Hình . 5](#_heading=h.92lc7e26kwzg)

[3.4. Vấn Đề Của RL Trong Mô hình POMDP. 10](#_heading=h.rmtha4z7jgmt)

[**Chương 4: Thực nghiệm 10**](#_heading=h.jghovy4imv5u)

[4.1. Theo bài báo 10](#_heading=h.x8sfa0brdwu0)

[4.1.1. Thiết lập phần cứng 10](#_heading=h.x021qkf8bo0d)

[4.1.2. Công cụ sử dụng 11](#_heading=h.t0loltxqzyem)

[4.1.3. Môi trường thực nghiệm 11](#_heading=h.1lbtwfif1eue)

[4.1.4. Kết quả 11](#_heading=h.di2kmcdwgyir)

[4.2. Tự thực hiện 14](#_heading=h.bhs8ozj8wbx4)

[4.2.1. Thiết lập phần cứng 14](#_heading=h.9y81cwc6ygyk)

[4.2.2. Công cụ sử dụng 14](#_heading=h.p5ct89kel2rg)

[4.2.3. Môi trường thực nghiệm 19](#_heading=h.1ub6niex6qpk)

[**Chương 5: Kết quả demo 20**](#_heading=h.7wwa03gsd0p6)

**Báo Cáo**

# Chương 1: Tổng quan

## 1.1 Ngữ cảnh thực tế

Trong bối cảnh an ninh mạng ngày càng trở nên quan trọng, việc kiểm thử xâm nhập nhằm xác định và khai thác các lỗ hổng trong mạng máy tính là cần thiết. Các phương pháp Pentest truyền thống thường tốn thời gian và dễ phát sinh lỗi vì lí do con người, đặc biệt là trong các hệ thống mạng quy mô lớn. Vì vậy nhu cầu sử dụng phần mềm kiểm thử tự động sử dụng Học tăng cường (Reinforcement Learning) là cần thiết trong các hệ thống mạng quy mô lớn.

## 1.2. Cơ sở lý thuyết

### 1.2.1. Khái niệm Penetration Testing

Penetration Testing (kiểm thử xâm nhập) là hình thức đánh giá mức độ an toàn của một hệ thống mạng bằng các cuộc tấn công mô phỏng thực tế. Hiểu đơn giản, pentest cố gắng xâm nhập vào hệ thống để phát hiện ra những điểm yếu tiềm tàng của hệ thống mà attackers có thể khai thác và gây thiệt hại.

Mục tiêu của pentest là giúp tổ chức phát hiện càng nhiều lỗ hổng càng tốt, từ đó khắc phục chúng để loại trừ khả năng bị tấn công trong tương lai.

### 1.2.2. Khái niệm Reinforcement Learning

Học tăng cường (RL) là một loại kỹ thuật machine learning mà hệ thống tự động tìm hiểu và cải thiện hành vi của mình thông qua tương tác với môi trường. Quy trình này dựa trên nguyên tắc học từ feedback và reward để tối đa hóa một hàm phần thưởng được xác định trước.

Học tăng cường hoạt động như một tín hiệu cho các hành vi tích cực và tiêu cực. Mục tiêu duy nhất của nó là tìm ra một mô hình hành động phù hợp để tăng tổng phần thưởng thu được. Phương pháp học tập này cho phép agent đưa ra một loạt quyết định nhằm tối đa hóa phần thưởng cho nhiệm vụ mà không cần sự can thiệp của con người hay lập trình rõ ràng để đạt được nhiệm vụ.

### 1.2.3. Hierarchical reinforcement learning vào xâm nhập kiểm thử tự động

Khung HRL chia quá trình kiểm thử xâm nhập thành các nhiệm vụ cấp cao và cấp thấp:

* Nhiệm vụ cấp cao: Xác định chiến lược kiểm tra tổng thể, chẳng hạn như ưu tiên các subnet hoặc dịch vụ quan trọng.
* Nhiệm vụ cấp thấp: Thực hiện các hành động cụ thể, chẳng hạn như khai thác lỗ hổng hoặc chuyển tiếp đến các nút khác.

HRL cho phép hệ thống học hỏi từ các tác vụ đã thực hiện trước đó, cải thiện khả năng phát hiện và khai thác lỗ hổng và tránh các hành động dư thừa hoặc không liên quan trong các lần tiếp theo.

# Chương 2: Mục tiêu nghiên cứu

Đồ án hướng tới việc thiết kế, triển khai và đánh giá HRL framework để kiểm thử xâm nhập tự động trong các hệ thống mạng máy tính lớn. Các mục tiêu chính bao gồm:

1. Tối ưu hóa quy trình kiểm thử xâm nhập.
2. Nâng cao khả năng mở rộng của các phương pháp Học tăng cường.
3. Cải thiện khả năng phát hiện những lỗ hổng bảo mật tiềm ẩn.
4. Chứng minh tính hữu ích của HRL trong kịch bản thực tế.

# Chương 3: Giải Quyết Vấn Đề Của POMDP.

## 3.1. Định Nghĩa Về POMDP.

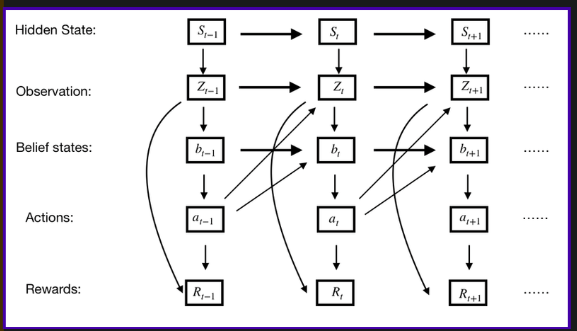
POMDP là viết tắt của Partially Observable Markov Decision Process (Quá trình quyết định Markov quan sát không đầy đủ). Đây là một mô hình toán học thường được sử dụng để giải quyết các bài toán quyết định trong môi trường không chắc chắn và thông tin không đầy đủ.

POMDP mô hình hóa các nhiệm vụ ra quyết định trong đó một tác nhân phải đưa ra quyết định dựa trên thông tin trạng thái không đầy đủ hoặc không chắc chắn.

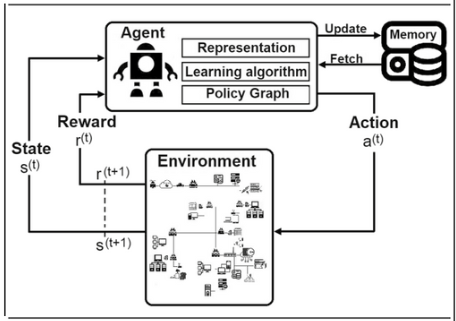
## 3.2. Các Thành Phần Cơ Bản Của POMDP.

* States (S): Một tập hợp hữu hạn các trạng thái biểu diễn tất cả các điều kiện có thể xảy ra trong hệ thống.
* Actions (A): Một tập hợp hữu hạn các hành động mà tác nhân có thể thực hiện.
* Transition Function (T): xác suất chuyển đổi từ trạng thái này sang trang thái khác
* Observations (O): Một tập hợp hữu hạn các quan sát mà tác nhân có thể nhận thức được.
* Rewards (R): gán một giá trị điểm cho từng hành động thực hiện được.
* Belief State (B): phân bố xác suất trên tất cả các trạng thái có thể có.

Cấu trúc ví dụ ở hình 1.



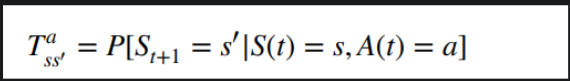
Hình 1.



Hình 2 :Mô hình POMDP trong tình cảnh kiểm thử xâm nhập

## 3.3. Các Hàm Của Mô Hình .

* *Hàm Xác Suất Chuyển Trạng Thái*:



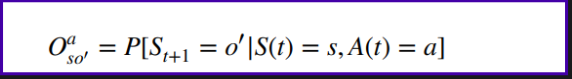
T(a,ss’) : Xác Suất trạng thái tiếp theo s’ sẽ xảy ra sau khi thực hiện hành động a trong trạng thái s.

S(t) : Trạng thái hiện tại tại thời điểm

St+1​: Trạng thái tiếp theo sau khi thực hiện hành động

A(t)​: Hành động được chọn tại thời điểm t.

* *Hàm Xác Suất Quan Sát:*

**

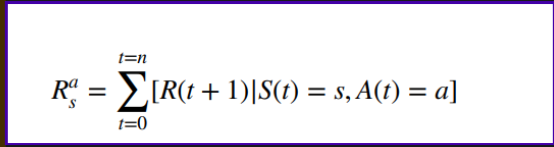
O(a,s'o): Xác suất nhận được quan sát o nếu trạng thái tiếp theo là s' sau hành động a.

S(t) : Trạng thái hiện tại tại thời điểm

St+1​: Trạng thái tiếp theo sau khi thực hiện hành động

A(t)​: Hành động được chọn tại thời điểm t.

* *Hàm Thưởng:*

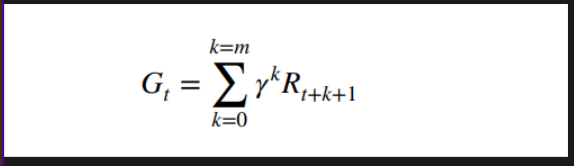
**

R(a,s): Tổng phần thưởng khi hành động a được thực hiện trong trạng thái s.

∑(n,t=0): Tổng các phần thưởng từ thời điểm t đến n.

R(t+1): Phần thưởng tại bước tiếp theo

* *Hàm Phần Thưởng Tổng (Giảm Giá)*

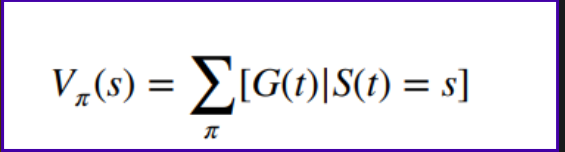


G(t​): Tổng phần thưởng từ thời điểm t trở đi.

γ^k: Hệ số giảm giá (giá trị từ 0 đến 1).

R(t+k+1​): Phần thưởng nhận được tại thời điểm tiếp theo.

* *Hàm Giá Trị Trạng Thái (Value Function)*

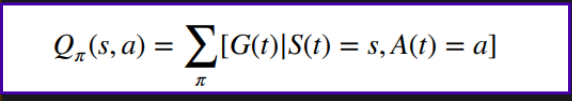
******

Vπ​(s): Giá trị mong đợi khi ở trạng thái s và tuân theo chính sách π.

P(τ): Xác suất của một chuỗi hành động τ.

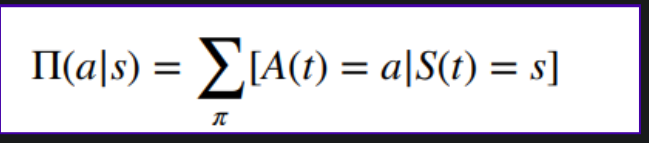
G(t): Phần thưởng tổng từ thời điểm t.

* *Hàm Giá Trị Hành Động (Action Value Function)*

**

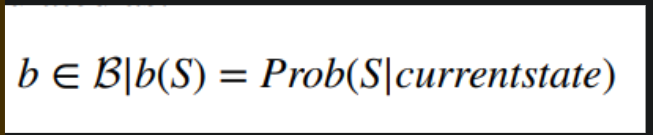
*Qπ​(s,a): Giá trị mong đợi khi thực hiện hành động a trong trạng thái s, theo chính sách π.*

* *Hàm chính sách (Policy Function)*

**

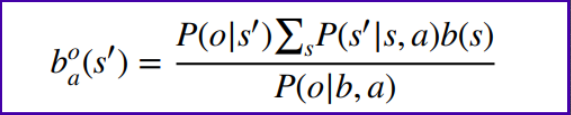
*Π(a|s): Xác suất thực hiện hành động 𝑎 một trạng thái s theo chính sách π.*

* *Hàm tính giá trị Trạng Thái niềm tin.*

**

*Π(a|s): Xác suất thực hiện hành động 𝑎 một trạng thái s theo chính sách π.*

* *Hàm cập nhập trạng thái niềm tin.*

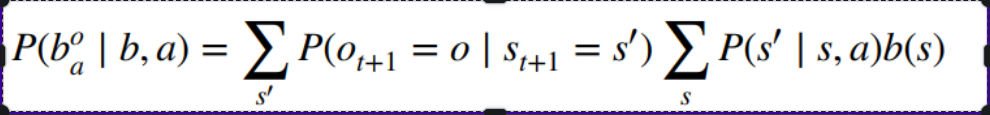
**

*P(o∣s′): Xác suất nhận được giá trị quan sát o cho trạng thái s′*

*∑s​P(s′∣s,a)b(s): Dự đoán xác suất đạt đến trạng thái s′ bằng cách tổng hợp tất cả các trạng thái hiện tại, được cân nhắc bởi trạng thái niềm tin b*

*P(o∣b,a): Xác suất nhận được giá trị quan sát o với niềm tin b và hành động a.*

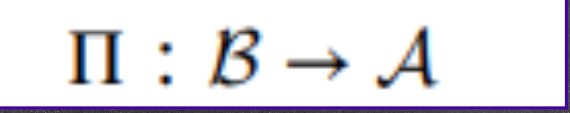
* *Hàm cập nhật giá trị Trạng Thái niềm tin*

**

*P(s′∣s,a)b(s): Điều này sẽ tính toán xác suất chuyển sang trạng thái s′ sau hành động a ở trạng thái s với sự cân nhắc của trạng thái niềm tin b*

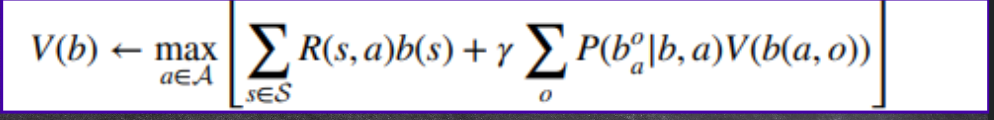
*P(ot+1​=o∣st+1​=s′): Cho trạng thái tiếp theo được dự đoán s′ thì xác suất quan sát được trạng thái o.*

* *Hàm Cập Nhật Chính Sách*

**

*Π: Chính sách được cập nhật dựa trên trạng thái niềm tin B thực hiện hành động A*

* *Phương trình Bellman để sao lưu POMDP*

**

*∑s∈S​R(s,a)b(s): Phần này cho biết phần thưởng mong đợi khi thực hiện hành động a với trạng thái niềm tin b*

*γ∑o​P(bo,a​∣b,a)V(b(a,o)):*

*P(bo,a​∣b,a): là khả năng chuyển sang niềm tin mới b phụ thuộc vào sự quan sát o và hành động a*

*V(b(a,o)): Giá trị niềm tin mới*

*Hệ số chiết khấu(γ): Điều này đảm bảo rằng phần thưởng trong tương lai sẽ ít quan trọng hơn phần thưởng trước mắt.*

## 3.4. Vấn Đề Của RL Trong Mô hình POMDP.

a, RL cần lượng dữ liệu tương tác rất lớn để học chính xác. Tốn nhiều tài nguyên và thời gian, đòi hỏi môi trường mô phỏng phức tạp.

b, RL thường chỉ hiệu quả trong các nhiệm vụ cụ thể. Khó áp dụng kiến thức học được từ một nhiệm vụ sang nhiệm vụ khác, yêu cầu đào tạo lại từ đầu khi môi trường thay đổi.

c,Hiệu suất RL dễ biến động trong quá trình học, đặc biệt ở giai đoạn đầu do tính ngẫu nhiên khi khám phá hành động. Gây mất ổn định trong các ứng dụng quan trọng.

d,Khó khăn trong việc xác định chức năng phần thưởng. Hướng đến tối ưu hóa phần thưởng nhưng không đạt mục tiêu thực sự.

# Chương 4: Thực nghiệm

## 4.1. Theo bài báo

### 4.1.1. Thiết lập phần cứng

- Thông số máy:

High-performance HP Z2 tower với CPU Intel Xeon E7-2176 (8 lõi, 3.70GHz, bộ nhớ đệm 20MB)

Bộ nhớ 64GB DDR4 2666 DIMM ECC

GPU Nvidia Quadro P4000 8GB

- Hệ điều hành:

Linux Calculate 20 với phiên bản kernel 5.4.6

### 4.1.2. Công cụ sử dụng

- PERSEUS: là một thuật toán xấp xỉ giải quyết POMDP bằng cách tối ưu hóa chính sách thông qua việc sử dụng một tập mẫu trạng thái

- RL-GIP-LPSolve: một phương pháp học tăng cường (Reinforcement Learning) kết hợp với giải pháp LPSolve để giải quyết bài toán POMDP.

- RL-GIP-LPSolve+Initial Belief : là một biến thể của RL-GIP-LPSolve, trong đó trạng thái ban đầu (Initial Belief) được sử dụng như một thành phần quan trọng để hướng dẫn quá trình học chính sách.

- HRL-GIP-LPSolve : HRL (Hierarchical Reinforcement Learning) kết hợp với GIP và LPSolve, sử dụng cách tiếp cận phân cấp để giải quyết POMDP.

- HRL-GIP-LPSolve+Initial Belief: là phiên bản nâng cao của HRL-GIP-LPSolve, trong đó Initial Belief được sử dụng để cung cấp thông tin ban đầu cho từng bài toán con trong cấu trúc phân cấp.

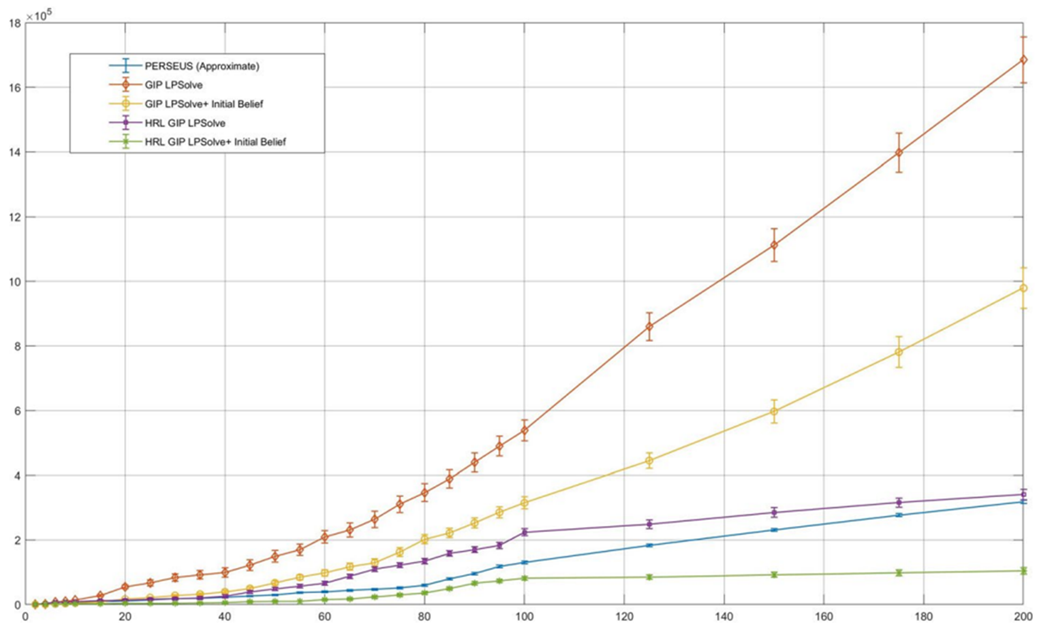
### 4.1.3. Môi trường thực nghiệm

Trên VirtualBox

Quy mô từ 2 đến 200 máy, tập trung vào 30-100 và 100-200 nút

### 4.1.4. Kết quả

Khi so sánh các công cụ trên thì sẽ cho ra kết quả sau

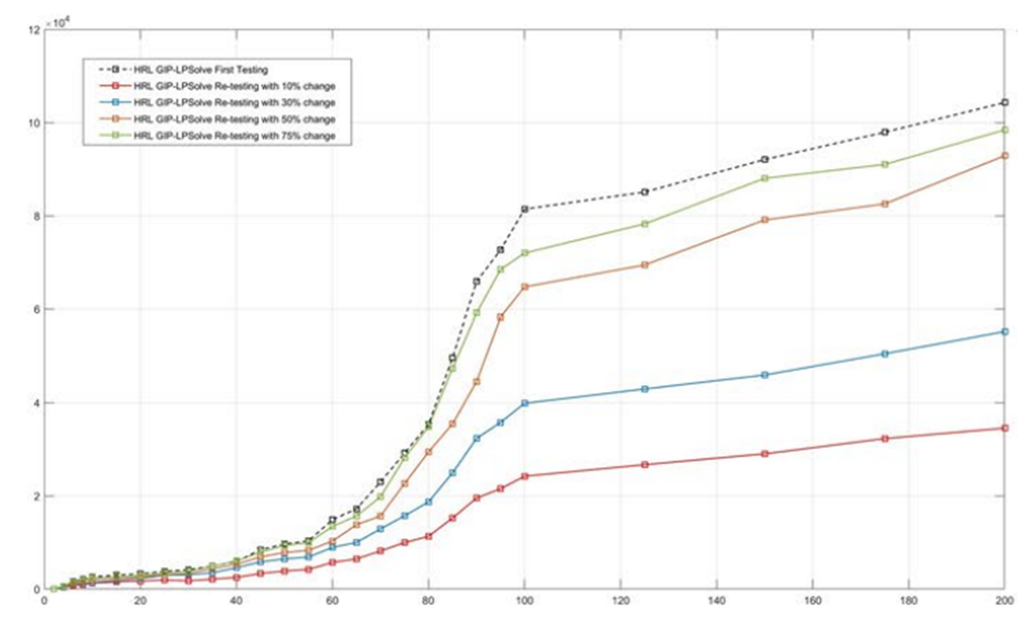


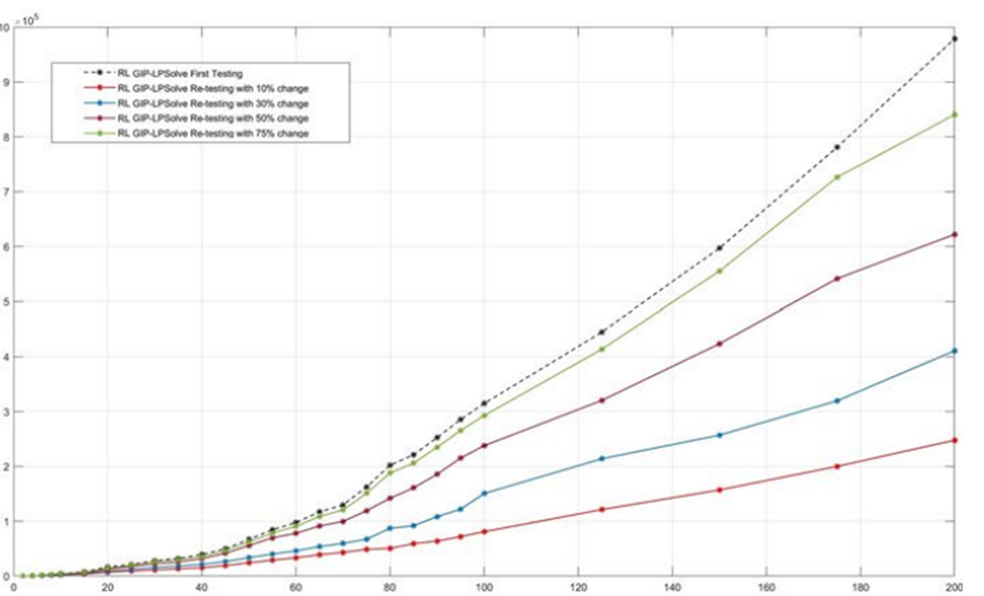
Với số lượng máy lên tới 10 máy (4 cụm, 33 lỗ hổng, 24 lần khai thác) cho thấy sự giảm hiệu suất đối với HRL, điều này là do phân cụm thường tạo ra một số lượng lớn các cụm bảo mật và do đó có nhiều cụm rất nhỏ.

Với mạng 100 máy (25 cụm, 102 lỗ hổng, 80 lần khai thác), HRL phương pháp tiếp cận yêu cầu 224087,118 ± 12564,7 (2,6 ngày) so với 538318,624 ± 31964.2 (6,2 ngày) trong RL-GIP thông thường.

Vượt xa quy mô 100 máy, HRL đang ở mức hiệu quả hơn ít nhất 4 lần và đạt tới 200 kích thước máy (52 cụm, 153 lỗ hổng, 115 lần khai thác), HRL-GIP hoạt động gần như xấp xỉ PERSEUS. yêu cầu 340582,592 ± 16297,8 (3,9 ngày) so với 1685011,539 ± 71160,5 (19,5 ngày) đối với RL-GIP.

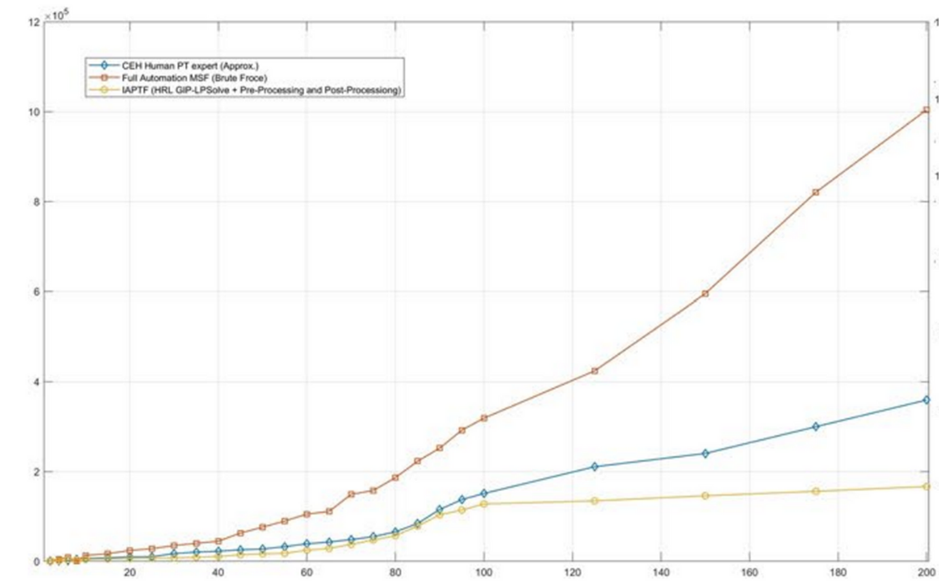
Tiếp theo với môi trường mạng với số lượng thay đổi 10%, 30%, 50% và 75%. Các thử nghiệm đã được thực hiện cho biến thể thuật toán với niềm tin ban đầu được tùy chỉnh, cụ thể là RL-GIP-LPSolve+Niềm tin ban đầu và HRL-GIP-LPSolve+Niềm tin ban đầu.





Hai biểu đồ trên cho thấy với tùy chỉnh niềm tin ban đầu, thời gian giải quyết đã giảm một nửa đối với 100-200 mạng LAN máy.

Cuối cùng sẽ kiểm tra sự hiệu quả so với do người kiểm thử CEH và kiểm thử tự động bằng bruteforce (MSF)



Kết quả ở trên cho thấy là IAPTF vượt trội hơn so với thử nghiệm CEH thủ công và tự động hóa brute-force (MSF), với phạm vi phủ sóng cao hơn đáng kể trong các mạng lớn.

## 4.2. Tự thực hiện

### 4.2.1. Thiết lập phần cứng

CPU: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz

System: Linux kali

Kernel: 6.8.11-amd64

Architecture: x86\_64

Operating System: GNU/Linux

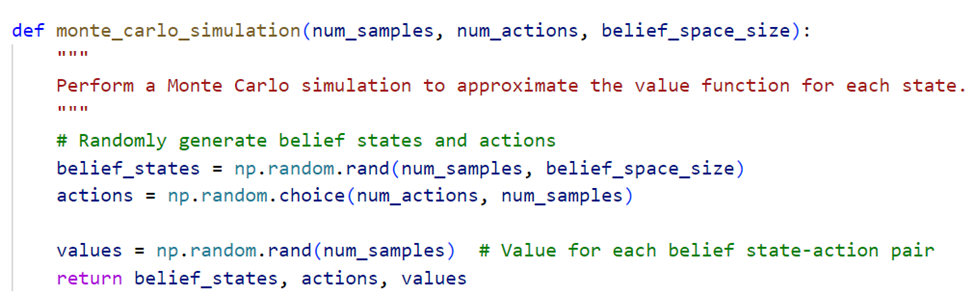
Graphic Card: VMware SVGA II Adapter

### 4.2.2. Công cụ sử dụng

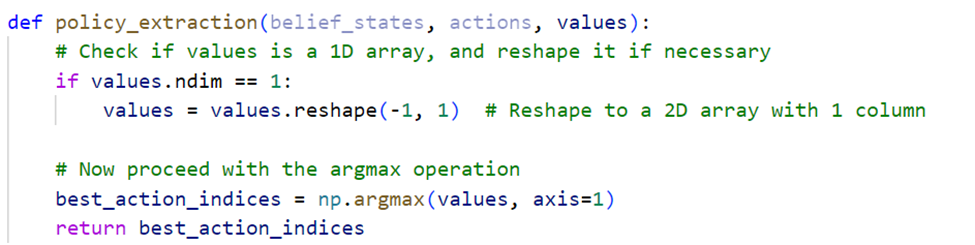
Các công cụ được tự tạo lại như bộ giải POMDP dựa trên Python và các cấu hình được điều chỉnh. Cụ thể là:

-PERSEUS

Hàm monte\_carlo\_simulation sẽ tạo action, trạng thái belief đầu cho mô hình POMDP.



Hàm policy\_extraction có mục đích để xác định hành động tốt nhất cho từng trạng thái niềm tin bằng cách tìm chỉ số giá trị tối đa.



Trong hàm solve\_pomdp sẽ tiến hành giải mô hình POMDP



-RL-GIP-LPSolve

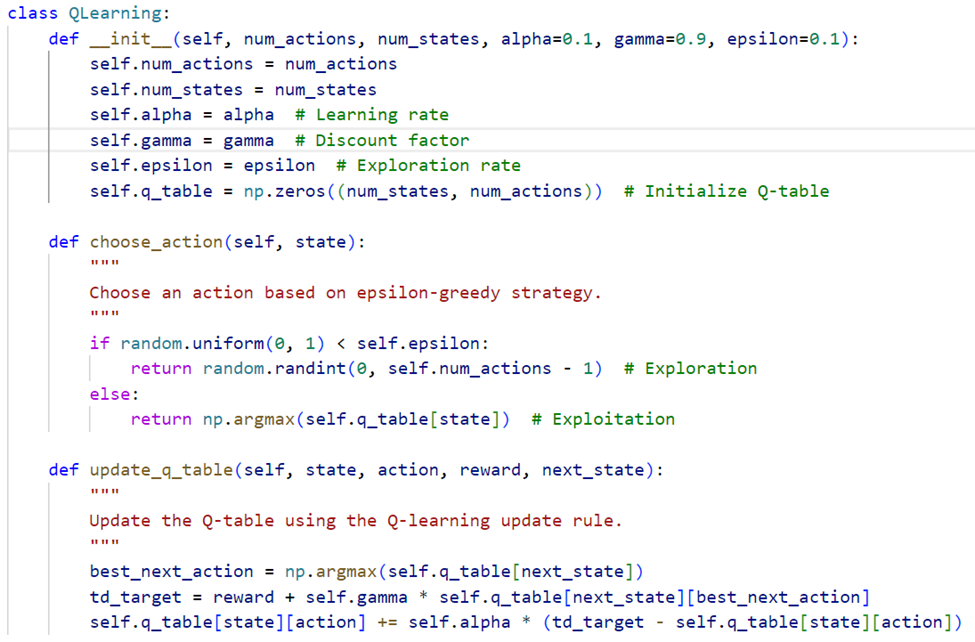
Trong RL có một lớp QLearning bao gồm 2 hàm:

choose\_action: Triển khai chiến lược epsilon-greedy để cân bằng việc thăm dò (hành động ngẫu nhiên) và khai thác (hành động tham lam dựa trên giá trị Q hiện tại).

update\_q\_table: Cập nhật Q-value.

Q(s,a)←Q(s,a)+α(r+γ maxa′ Q(s′,a′)−Q(s,a)).

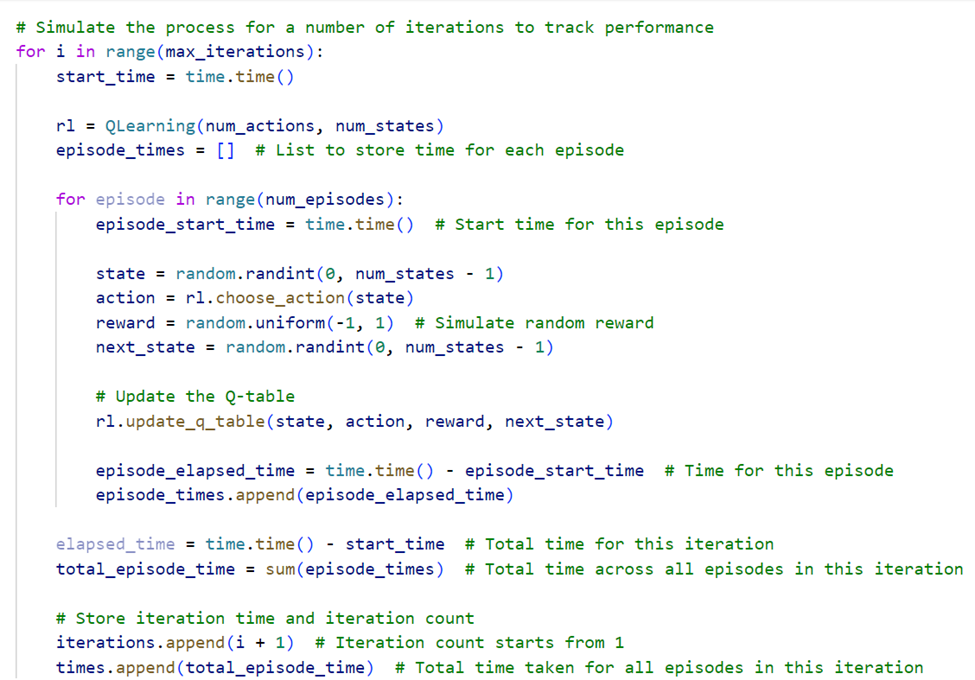
Mục đích là triển khai thuật toán Q-learning để giải quyết các vấn đề học tăng cường.



Hàm solve\_pomdp

Để giải sẽ bắt đầu bằng một Q-learning agent.

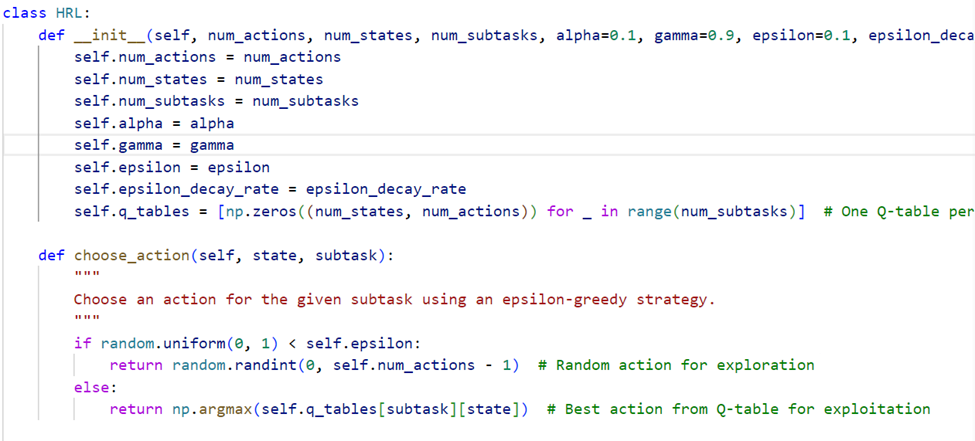
Chạy num\_episodes để cập nhật Q-table.



-HRL-GIP-LPSolve

Trong lớp HRL sẽ chứa một bảng Q riêng cho mỗi nhiệm vụ phụ

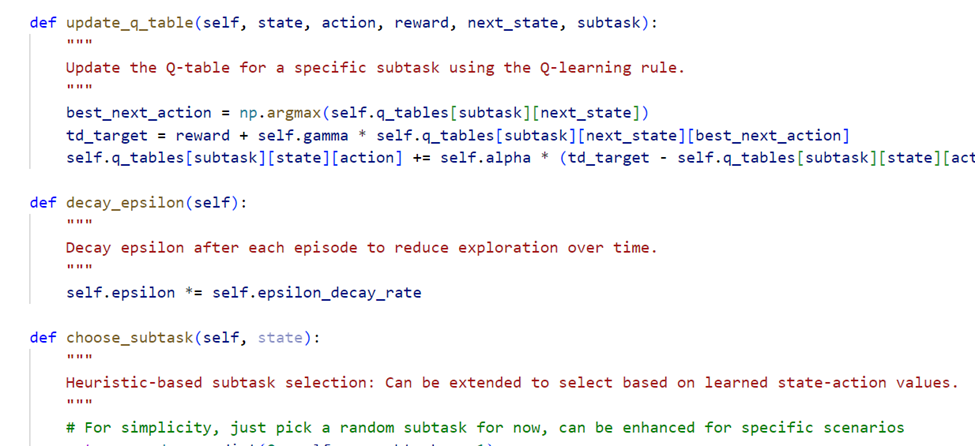
Hàm choose\_action: Lựa chọn hành động thông qua chiến lược epsilon-greedy



Hàm update\_q\_table: Cập nhật bảng Q cho từng nhiệm vụ con bằng quy tắc Q-learning.

Hàm decay\_epsilon: Sự phân rã của epsilon để cân bằng việc thăm dò và khai thác theo thời gian.

Hàm choose\_subtask: Lựa chọn nhiệm vụ con dựa trên heuristic.

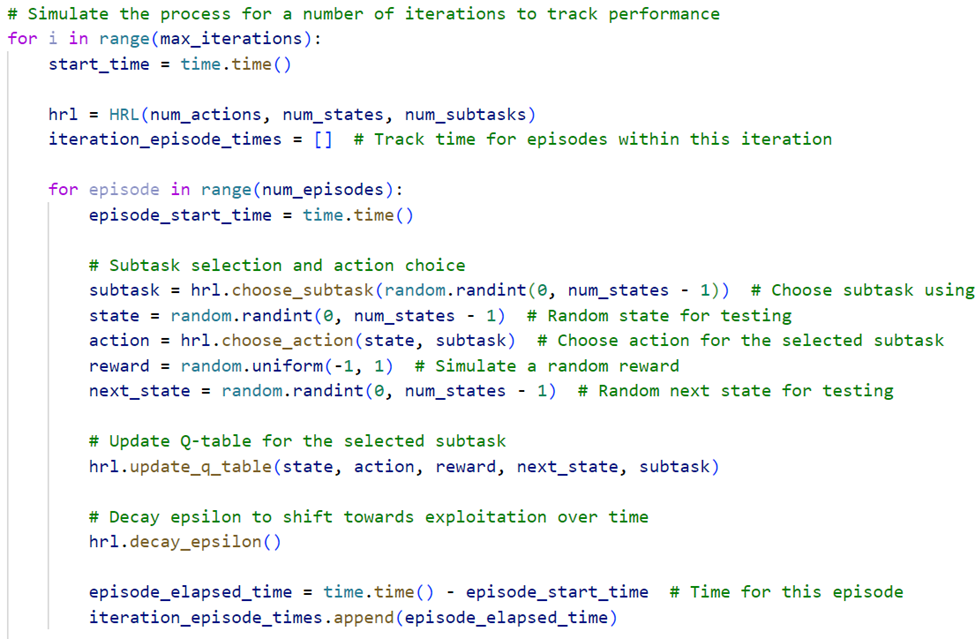


Hàm solve\_pomdp:

Agent chia vấn đề thành các nhiệm vụ con, mỗi nhiệm vụ có bảng Q riêng.

Một heuristic (lựa chọn nhiệm vụ con ngẫu nhiên) xác định nhiệm vụ phụ nào đang hoạt động.

Phân rã Epsilon ban đầu khuyến khích sự khám phá và chuyển sang khai thác khi quá trình đào tạo tiến triển.



### 4.2.3. Môi trường thực nghiệm

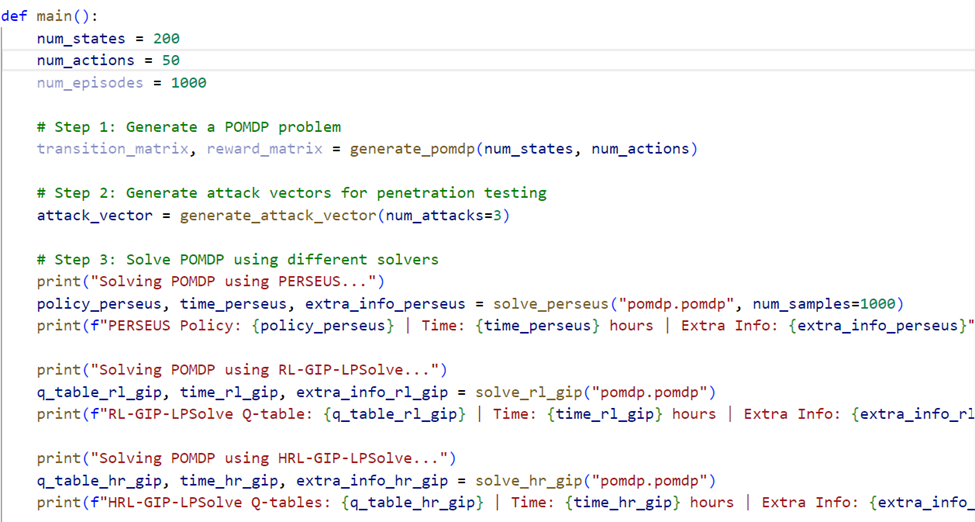
- Thực hiện trên môi trường máy ảo: Vmware

Giới hạn ở một tập hợp con của thử nghiệm ban đầu do hạn chế về mặt tính toán.

Môi trường được tạo ra dưới một hàm tạo tệp POMDP



Trong hàm main sẽ điều khiển mô hình POMDP được tạo ra qua 3 biến num\_states (số trạng thái), num\_actions (số hành động của mô hình), num\_episodes (số lần mỗi công cụ sẽ chạy)

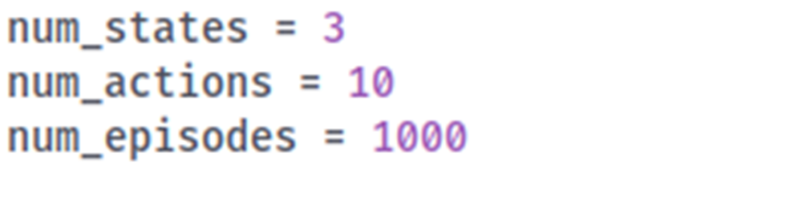


# Chương 5: Kết quả demo

Dựa trên những thông số của phần 4.2, sau khi thực nghiệm nhóm đã thu được kết quả sau:

Với số trạng thái trong POMDP là 3, số hành động có thể thực hiện là 10, số lần thử nghiệm hoặc vòng lặp trong quá trình học chính sách là 1000.

Giả lập một môi trường mạng nhỏ.



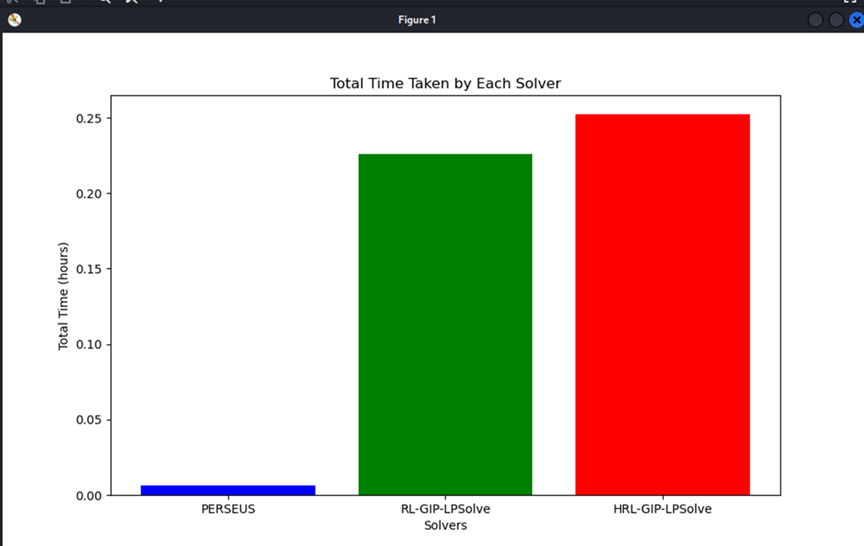
Sau khi thực hiện thì thấy được là

Perseus sẽ có thời gian nhanh nhất

RL có thời gian đứng thứ hai

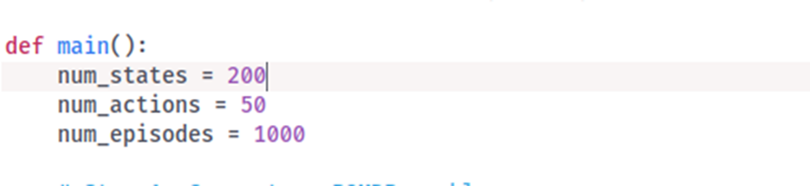
HRL sẽ có thời gian chậm nhất

Nguyên nhân là do HRL có phần chia các thành phần trong mô hình thành các mục nhỏ rồi sau đó các mục đó mới được RL xử lí.



Với số trạng thái trong POMDP là 200, số hành động có thể thực hiện là 50, số lần thử nghiệm hoặc vòng lặp trong quá trình học chính sách là 1000.

Giả lập một môi trường mạng lớn.



Trong trường hợp này:

Perseus: vẫn là công cụ nhanh nhất

RL: Chậm hơn HRL

HRL: có sự cải thiện, chạy nhanh hơn RL



Đánh giá:

So với môi trường thử nghiệm lớn hơn và công cụ, phần thử nghiệm cũng cho ra kết quả tương tự với bài báo về sự hiệu quả của HRL trong môi trường mạng lớn.

**--- HẾT ---**