Phát triển Framework ứng dụng AI hỗ trợ tự động khai thác lỗ hổng bảo mật

Nguyễn Mạnh Thiên, Phạm Đăng Khoa, Nguyễn Đức Vượng, Nguyễn Việt Hùng

***Tóm tắt*—Hiện nay, nhiệm vụ đánh giá an toàn thông tin cho các hệ thống thông tin có ý nghĩa quan trọng trong đảm bảo an toàn thông tin. Đánh giá/khai thác lỗ hổng bảo mật cần được thực hiện thường xuyên và ở nhiều cấp độ khác nhau đối với các hệ thống thông tin. Tuy nhiên, nhiệm vụ này đang gặp nhiều khó khăn trong triển khai diện rộng do thiếu hụt đội ngũ chuyên gia kiểm thử chất lượng ở các cấp độ khác nhau. Trong khuôn khổ bài báo này, chúng tôi trình bày nghiên cứu phát triển Framework có khả năng tự động trinh sát thông tin và tự động lựa chọn các mã để tiến hành khai thác mục tiêu dựa trên công nghệ học tăng cường (Reinforcement Learning). Bên cạnh đó Framework còn có khả năng cập nhật nhanh các phương pháp khai thác lỗ hổng bảo mật mới, hỗ trợ tốt cho các cán bộ phụ trách hệ thống thông tin nhưng không phải là chuyên gia bảo mật có thể tự động đánh giá hệ thống của mình, nhằm giảm thiểu nguy cơ từ các cuộc tấn công mạng.**

***Abstract*—Currently, security assessment is one of the most important proplem in information security. Vulnerability assessment/exploitation should be performed regularly with different levels of complexity for each information system. However, this task is facing many difficulties in large-scale deployment due to the lack of experienced testing experts. In this paper, we proposed a Framework that can automatically gather information and automatically select suitable module to exploit the target based on reinforcement learning technology. Furthermore, our framework has intergrated many scanning tools, exploited tools that help pentesters doing their work. It also can be easily updated new vulnerabilities exploit techniques.**

Từ khóa—Đánh giá kiểm thử xâm nhập; Phân tích lỗ hổng bảo mật; Học tăng cường; Mô hình Asynchronous Advantage Actor-Critic; A3C.

Keywords—Penetration testing, Vulnerability analysis, Reinforcement learning, automated exploitation.

I. GIỚI THIỆU CHUNG

Lỗ hổng bảo mật đề cập đến sự thiếu sót hoặc yếu kém của hệ thống có thể dẫn đến việc đánh cắp dữ liệu bí mật, phá vỡ tính toàn vẹn của dữ liệu hoặc ảnh hưởng đến tính khả dụng của ứng dụng. Kẻ tấn công có thể truy cập thông qua những điểm yếu này và sau đó khai thác, phá hoại hệ thống.

Để kiểm thử khai thác lỗ hổng bảo mật, thông thường phải trải qua giai đoạn trinh sát thông tin. Sau khi thu thập đủ các thông tin cần thiết, kiểm thử viên sẽ tiến hành đánh giá các lỗ hổng bảo mật, lựa chọn ra các lỗ hổng có tiềm năng khai thác. Kiểm thử viên sẽ xây dựng hoặc lựa chọn các công cụ tấn công phù hợp, tiến hành thử nghiệm khai thác. Quá trình này thường sẽ được thực hiện trực tiếp bởi các chuyên gia có kinh nghiệm, với sự giúp đỡ của các công cụ hỗ trợ. Hiện nay, nhiều công cụ đã được phát triển để giúp thực hiện thu thập thông tin và kiểm tra các lỗ hổng bảo mật tồn tại trong hệ thống, có thể kể đến như:

- Nmap ("Network Mapper") [1] là một công cụ bảo mật được phát triển bởi Gordon Lyon. Nó được dùng để phát hiện các máy chủ và các dịch vụ trên mạng máy tính bằng cách gửi các gói tin và phân tích các phản hồi.

- Acunetix Web Vulnerability Scanner [2]: là một công cụ kiểm tra bảo mật ứng dụng web. Nó tự động kiểm tra các ứng dụng web để tìm kiếm các lỗ hổng bảo mật như SQL Injection, Cross-Site Scripting (XSS), … Đây là một phần mềm trả phí.

- Metasploit [3]: Là một môi trường dùng để kiểm tra, tấn công và khai thác lỗi của các service. Nó được phát triển từ ngôn ngữ Perl và có thể chạy trên hầu hết các hệ điều hành: Linux, Windows, MacOS.

- Wappalyzer [4]: Là một tiện ích đa nền tảng giúp khám phá các công nghệ được sử dụng trên các trang web.

Bài báo được nhận ngày 25/9/2021. Bài báo được nhận xét bởi phản biện thứ nhất ngày 5/10/2021 và được chấp nhận đăng ngày 02/11/2021. Bài báo được nhận xét bởi phản biện thứ hai ngày 10/10/2021 và được chấp nhận đăng ngày 25/10/2021.

Phần lớn các phần mềm kể trên đều thực hiện một chức năng nhất định, không có phần mềm nào hỗ trợ hiệu quả đầy đủ các bước như đã mô tả nên người dùng không có được đánh giá chính xác về mức độ an toàn thông tin của hệ thống. Bên cạnh đó, việc xuất hiện nhiều trường hợp cảnh báo giả khi dò quét lỗ hổng bảo mật mang đến nhiều khó khăn cho cán bộ phụ trách hệ thống thông tin nếu những người này có ít kiến thức về bảo mật hệ thống.

Hiện nay đã tồn tại một số Framework khác nhau để kiểm thử bảo mật. Trong đó có thể kể đến như:

- Rengine [5]: Là một Framework thu thập thông tin. Nó cho phép tích hợp các công cụ thu thập thông tin khác để thực thi và đưa ra một kết quả tổng hợp.

- VulScan [6]: Là một Framework cho phép tích hợp các mã khai thác cho các lỗ hổng phổ biến(CVE) và kiểm thử trên mục tiêu với một giao diện web.

- Nexpose [7]: Là một công cụ tính phí cho phép thu thập thông tin, dò quét, khai thác các lỗ hổng bảo mật (nếu có) của các hệ thống.

- TestREx [8]: Một Framework cho phép kiểm thử mã khai thác với các cấu hình hệ thống khác nhau của website. Sử dụng kết hợp docker kết hợp selenium lặp lại thao tác kiểm thử trên giao diện thử nghiệm mã khai thác với các lỗ hổng khác nhau(XSS, SQL injection, …).

Các Framework hoặc công cụ trên được thiết kế và xây dựng chỉ dành cho một số mục đích nhất định như: thu thập thông tin, khai thác lỗ hổng bảo mật nhưng người sử dụng phải điền thông tin chính xác điểm cuối(endpoint) để tiến hành việc kiểm thử,… Một số công cụ có mã nguồn đóng chỉ cho phép kiểm thử dựa trên các công cụ đã được cấu hình sẵn trong hệ thống gây khó khăn cho người dùng và ảnh hưởng đến kết quả kiểm thử. Từ đó đặt ra việc phải phát triển một Framework cho phép người dùng có thể sử dụng các công cụ mong muốn, thậm chí là do người dùng tự định nghĩa để có thể kiểm thử bảo mật hệ thống thông tin. Việc tích hợp là cần thiết giúp tận dụng điểm mạnh của từng công cụ, tối đa hóa kết quả phục vụ cho việc kiểm thử bảo mật. Việc kiểm thử không chỉ giới hạn trong các website mà mở rộng ra đến cả các hệ thống thông tin chỉ cần có kết nối mạng đến.

Framework được thiết kế và xây dựng nhằm cho phép người dùng có thể thực hiện đầy đủ các bước trong quy trình khai thác bảo mật, trong đó có module lựa chọn mã khai thác dựa vào AI để thử nghiệm tấn công vào các lỗ hổng bảo mật. Hệ thống cho phép chạy song song các tiến trình thu thập thông tin và đánh giá các lỗ hổng bảo mật với công nghệ multithreading [9] và docker [10]. Dựa vào đó, thời gian thực hiện quy trình thu thập thông tin của hệ thống được rút ngắn. Bên cạnh đó, hệ thống còn thực hiện tiến trình kiểm tra lại (re-testing), cho phép hạn chế các trường hợp cảnh báo giả và đưa ra cảnh báo về lỗ hổng bảo mật chính xác đến người quản trị.

II. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

Với mục tiêu xây dựng nền tảng cho phép tích hợp các công cụ thu thập thông tin mã nguồn mở, tự động tìm kiếm, kiểm thử các lỗ hổng, chúng tôi đã tiến hành khảo sát, phân tích quy trình kiểm thử, khai thác lỗ hổng bảo mật, từ đó thiết kế ra Framework có các chức năng phù hợp.

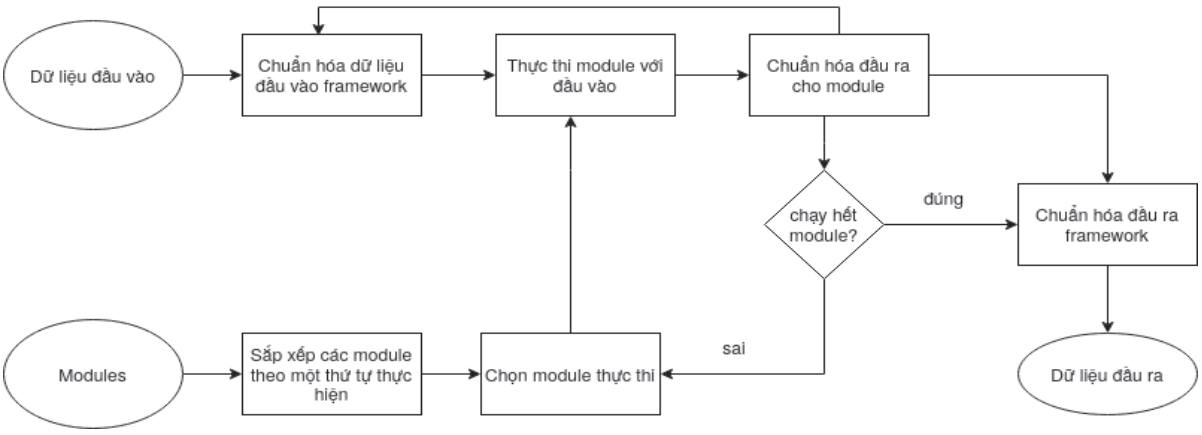
Các chức năng chính của Framework gồm:

- Thêm mục tiêu, thông tin về mục tiêu (bằng các thông tin có được từ các nguồn khác, nếu có). Thực hiện nhiệm vụ này là nhóm module đầu vào có chức năng nhận thông tin từ phía người dùng và tiền xử lý thông tin.

- Thu thập thông tin về đối tượng thông qua các module thu thập thông tin. Nhóm module dò quét sẽ phục vụ mục đích thu thập thông tin, dò quét các lỗ hổng, thông tin đối tượng hướng tới.

- Kiểm thử các lỗ hổng CVE, các lỗ hổng nghiêm trọng. Nhóm module kiểm tra lỗ hổng có mục đích là kiểm tra các lỗ hổng trong quá trình dò quét, đồng thời tìm ra các lỗ hổng CVE tiềm năng mà module dò quét không tìm được.

- Đưa ra báo cáo về quá trình kiểm thử gồm các thông tin về quá trình dò quét và thông tin về các lỗ hổng tìm được đồng thời có thể thực thi các lỗ hổng RCE khi cần thiết. Nhóm module đầu ra có mục đích đưa thông tin ra sau mỗi lần kiểm thử tới người dùng bằng báo cáo, hoặc xuất thông tin cho các công cụ khai thác lỗ hổng trực tiếp khác.



Hình 1. Luồng dữ liệu được xử lý trong Framework.

- Cho phép bổ sung các module được viết bằng python vào Framework dễ dàng. Các công cụ, mã nguồn mở được cài đặt trên môi trường docker, CVE-plugins được viết bằng python.

Để có thể thực hiện được các module rời rạc nhau, Framework cần phải tổ chức dữ liệu, xây dựng quy trình chung cho quá trình dò quét, khai thác lỗ hổng . Đưa ra luồng dữ liệu trao đổi giữa các module, thứ tự thực hiện, khả năng đồng bộ từ đó đưa ra luồng dữ liệu (Hình 1).

Dữ liệu đầu vào: Là dữ liệu được nhập từ phía người dùng, các thông tin liên quan tới mục tiêu cần dò quét, khai thác lỗ hổng. Các thông tin này bao gồm các thông tin mà người dùng có.

Modules: Là các module được thiết lập sẵn trên Framework, người dùng sẽ truyền các module vào Framework thông qua truyền tên của các module. Mỗi module được thiết kế theo một khuôn mẫu nhất định mà chỉ Framework có thể thực thi được.

Chuẩn hóa dữ liệu đầu vào Framework: Với mỗi loại dữ liệu đầu vào khác nhau cần chuẩn hóa sao cho Framework có thể đọc được theo các trường đã được khởi tạo, ngoài các trường đó ra thì các đầu vào khác sẽ được loại bỏ. Các trường của đầu vào được lấy ra từ đầu vào của các module. Từ đầu vào đó sẽ được đưa vào module tương ứng để module có thể hoạt động được.

Sắp xếp các module theo thứ tự thực hiện: Các module sẽ được chia thành các vào các nhóm module. Hiện tại Framework hỗ trợ 4 nhóm module chính như sau: Module_Input, Module_Reconnaissance, Module_Expoit, Module_Output và một nhóm hỗ trợ: Module_Other. Trong quá trình thực thi, các module được truyền vào sẽ được phân vào 4 nhóm, thứ tự thực hiện các module được sắp xếp theo nhóm module.

Thực thi module với đầu vào: Các module sẽ được chuyển thành các đối tượng tương ứng với mỗi module được thực thi. Đầu vào từ Framework sẽ được chuyển đổi thành đầu vào của module, điều này giúp module có thể sử dụng chính xác các trường cần thiết.

Chuẩn hóa đầu ra cho module: Các module sẽ trả về các giá trị, các trường khác nhau. Vậy nên cần thiết phải chuẩn hóa lại theo một mẫu thống nhất chung để các module sau có thể sử dụng. Sau đó phần dữ liệu này sẽ truyền ngược lại thông qua chuẩn hóa đầu vào để cho module tiếp theo có thể sử dụng.

Chuẩn hóa đầu ra cho Framework: Quá trình xuất dữ liệu đầu ra cho Framework sẽ tiếp tục được chuẩn hóa sao cho dữ liệu được chính xác nhất. Các module khác nhau có thể trả về cùng kiểu dữ liệu vậy cần thiết phải loại bỏ các module trùng nhau. Kết quả thu được sẽ gửi ra ngoài Framework.

Điểm mạnh của Framework là tính mở, có thể cho phép dễ dàng tích hợp module khai thác mới vào Framework. Mỗi module sẽ chuẩn hóa đầu ra sao cho khớp với các trường thông tin mà Framework đã có. Công việc chuẩn hóa bao gồm như sau:

- Phân loại dữ liệu đầu ra về các trường thông tin khớp với Framework đã có hoặc bổ sung thêm các trường mới nếu Framework chưa có.

- Loại bỏ các thông tin trùng lặp trong một trường. Định nghĩa giống nhau sẽ do các chính các module quy định.

- Gộp các thông tin khác nhau, xung đột với nhau thành một danh sách.

Chi tiết các thành phần để Framework có thể hoạt động hoàn chỉnh được trình bày trong hình dưới đây (Hình 2).

Tầng GUI: gồm API và Web-GUI cung cấp khả năng truy cập dữ liệu vào Framework.

Tầng Process: đảm nhiệm phân luồng dữ liệu tới các module, truy xuất thông tin tới database, bao gồm:

- *Process Main:* bộ xử lý chính của Framework có nhiệm vụ: Đảm nhận truy xuất dữ liệu từ database với GUI. Truy xuất dữ liệu từ database, chuyển định dạng đầu vào cho từng module do mỗi module khác nhau thì sẽ có định dạng khác nhau.

- *Database:* lưu trữ dữ liệu sử dụng mongoDB.

Tầng nhóm module: chứa các module thực thi chính của Framework, gồm 4 nhóm chính:

Module-Input: Các module hỗ trợ nhập dữ liệu đầu vào và sau đó chuyển thành một chuẩn cố định.

Module-Reconnaissance: Nhóm các module thu thập thông tin, đầu vào sẽ lấy từ module-input hoặc từ các module thu thập thông tin khác.

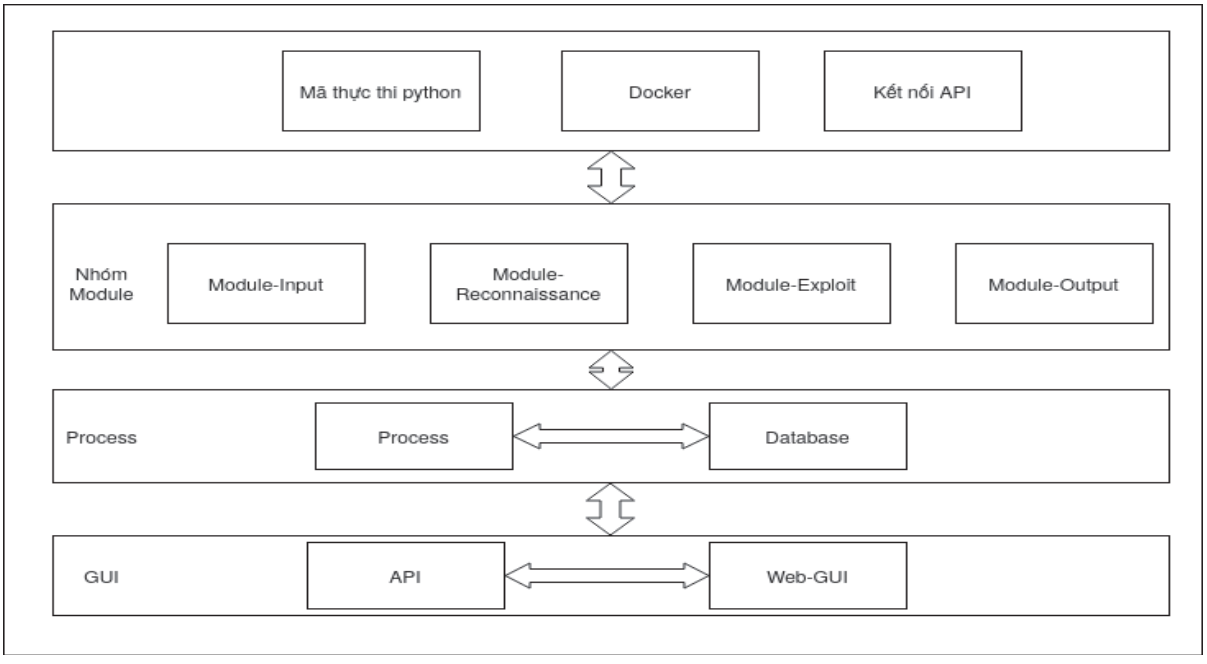
Module-Exploit: Nhóm các module sẽ sử dụng thông tin từ nhóm module-reconnaissance để tiến hành kiểm thử và tìm kiếm lỗ hổng trên mục tiêu.

Module-Output: Nhóm module xuất đầu ra cho Framework.

Các Nhóm module sẽ chưa các module, các module sẽ thực hiện các chức năng khác nhau nhìn chung chức năng của của module sẽ dựa trên 3 thành phần chính:

Mã thực thi python: Module trực tiếp được viết bằng python, thực thi các yêu cầu và trả về kết quả. Các công cụ sử dụng hình thức này thông thường là các công cụ đơn giản, không bị xung đột nhiều với các công cụ khác.

Docker: Các công cụ được triển khai trên môi trường docker sẽ có khả năng tách biệt với các công cụ khác, ổn định hơn trong quá trình khai thác. Chúng tôi khuyến khích nên sử dụng docker để phát triển các module.



Hình 2. Các thành phần chính của Framework.

Kết nối API: Kết nối, sử dụng tới các công cụ trong mạng, sử dụng thông qua các kết nối giao tiếp để điều khiển, gửi và nhận dữ liệu.

Các thành phần góp phần hỗ trợ quá trình tìm kiếm, khai thác lỗ hổng thuận tiện nhất. Hiện tại, nhóm chúng tôi đã xây dựng một số module phục vụ mục đích tìm kiếm, khai thác lỗ hổng (Bảng 1).

BẢNG 1. MỘT SỐ MODULE TRONG FRAMEWORK

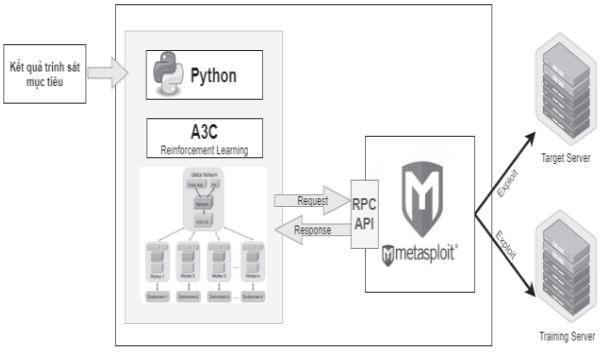
TT	Nhóm Module	Tên module	Chức năng
1	Input	Module-Target	Chuyển đầu vào của mục tiêu thành dạng chuẩn
2	Reconnaissance	Module-Nmap	Sử dụng công cụ Nmap dò quét cổng, dịch vụ, hệ điều hành,.. sử dụng docker nmap [11]
3		Module-WebApp Detect	Xác định các cổng mở dịch vụ web
4		Module-DirSearch	Dò quét đường dẫn đối với trang web sử dụng docker Dirsearch [12]
5		Module-Wappalyzer	Dò quét dịch vụ chạy trên website sử dụng docker Wappalyzer [14]
6		Module-Nuclei	Dò quét lỗ hổng, dịch vụ của mục tiêu sử dụng docker Nuclei [13]
7	Exploit	Module-PoCCheck	Kiểm thử đối với các lỗ hổng CVE sử dụng các script khai thác
8		Module-MetasploitAI	Kiểm thử sử dụng module AI chọn lọc payload Metasploit
9	Output	Module-Report	Xuất báo cáo dạng pdf
10		Module-OutputMetasploit	Thực thi lỗ hổng qua server Metasploit dựa trên kết nối msgrpc [23]
11		Module-OutputShell	Thực thi lỗ hổng thủ công bằng shell

III. MODULE ỨNG DỤNG AI TRONG TỰ ĐỘNG KHAI THÁC LỖ HỔNG BẢO MẬT

Trong Framework xây dựng, chúng tôi có phát triển và tích hợp module khai thác lỗ hổng bảo mật tự động sử dụng trí tuệ nhân tạo. Module này sẽ nhận thông tin trinh sát từ module Reconnaissance, sau đó tự động lựa chọn mã khai thác trong công cụ Metasploit và thực hiện khai thác.

A. Ý tưởng chung

Để giải quyết vấn đề đặt ra, chúng tôi nghiên cứu việc áp dụng Học tăng cường (Reinforcement Learning) cho kiểm tra xâm nhập mạng tự động. Trong nghiên cứu này, chúng tôi cải tiến mô hình học tăng cường Asynchronous Advantage Actor-Critic (A3C) để giải quyết vấn đề. Module này kết nối công cụ Metasploit với một mô hình thuật toán A3C để tự động hóa việc thực hiện một loạt các thao tác từ thu thập thông tin đến phát hiện và khai thác các lỗ hổng bảo mật trên hệ thống. Hệ thống kết nối với Metasploit thông qua RPC API (Remote Procedure Call) của Metasploit để thực thi các câu lệnh khai thác của Metasploit từ xa. Dựa vào thông tin trinh sát được từ phần Trinh sát của Framework tới máy chủ mục tiêu, hệ thống kết nối với Metasploit và mô hình học tăng cường A3C để đưa ra các thông tin được sử dụng để có thể khai thác được trên Metasploit (rhost, rport, module exploit, payload, target,…), từ đó gửi thông tin khai thác đến Metasploit, thực hiện khai thác và đưa ra kết quả khi thành công (tạo được session đến máy mục tiêu).



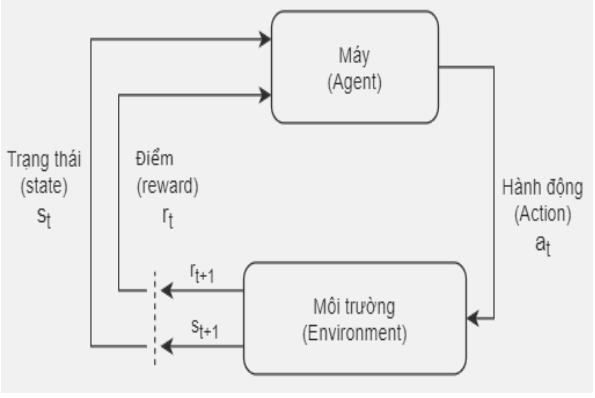
Hình 3. Các thành phần trong module tự động khai thác ứng dụng trí tuệ nhân tạo.

Trong bài toán này, từ các thông tin trinh sát được (dịch vụ, phiên bản dịch vụ, hệ điều hành) sẽ lấy ra được các module khai thác và target (option target trong Metasploit) tương ứng có thể sử dụng. Nhiệm vụ của máy lúc này là đưa ra được payload thích hợp tương ứng để thực hiện tấn công. Để giải quyết vấn đề này, chúng tôi sử dụng thuật toán học tăng cường A3C để đánh giá và lựa chọn được cặp trạng thái - hành động tối ưu, từ đó đưa ra các thông tin khai thác.

B. Mô tả mô hình AI

1. Học tăng cường

Học tăng cường là kỹ thuật cho phép đưa ra chính sách và từ đó học chính sách tốt nhất thông qua tương tác với môi trường. Học tăng cường liên quan đến việc dạy cho máy (Agent) thực hiện tốt một nhiệm vụ bằng cách tương tác với môi trường (Environment) thông qua hành động (Action) và nhận được điểm (Reward) và trạng thái tiếp theo của máy. Điểm này có thể tốt hoặc xấu, từ đó, máy sẽ bắt đầu học và cố gắng thực hiện các hành động mà có thể nhận về được điểm tốt. Cách học này được gọi là trial-and-error [17].



Hình 4. Cơ chế hoạt động của học tăng cường.

Học tăng cường bao gồm [17]:

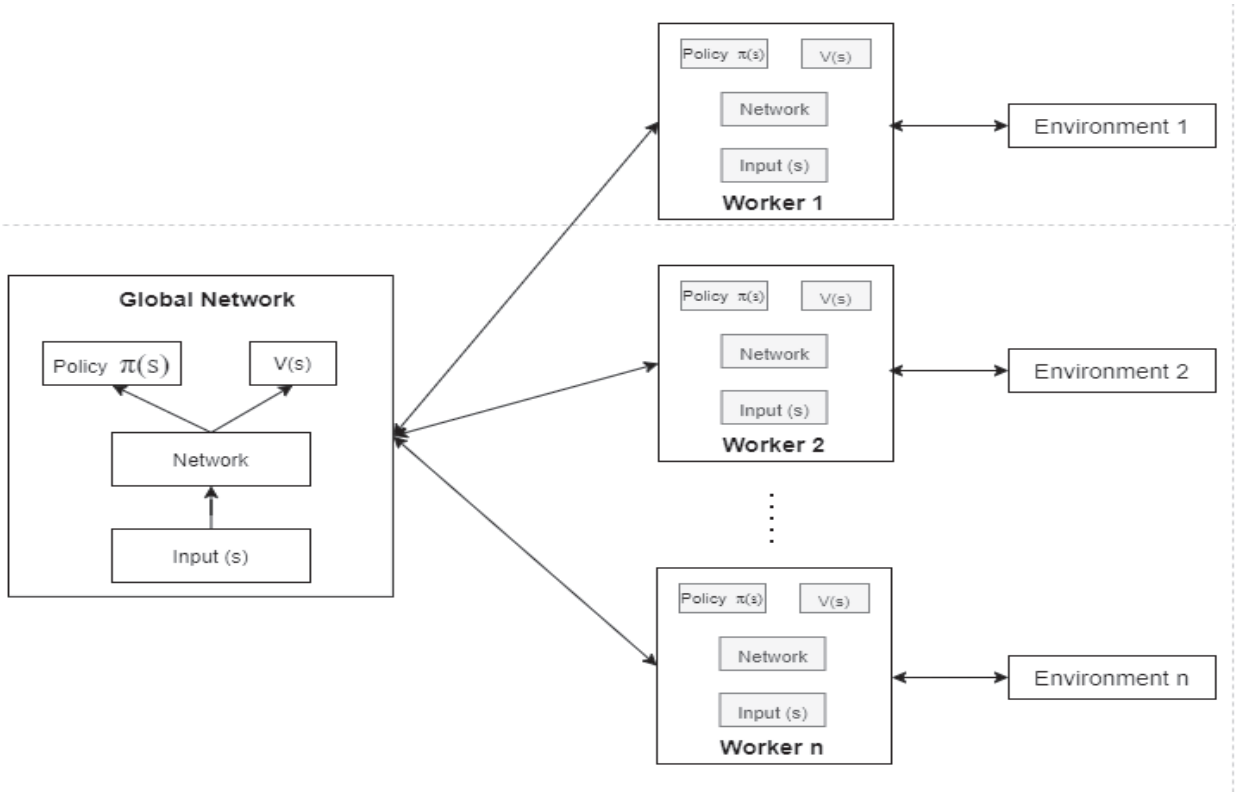
- $S = \{s_1, s_2, s_3, \ldots, s_n\}$: là tập biểu diễn các trạng thái của hệ thống.
- $A = \{a_1, a_2, a_3, \ldots, a_m\}$: là tập biểu diễn các hành động của máy có thể được thực hiện tại trạng thái hiện tại.
- $R(s_i, a_j)$: là điểm tức thời mà máy thu được khi thực hiện hành động a_j tại trạng thái s_i.
- π: được gọi là một chính sách, được máy sử dụng để tìm ra hành động tiếp theo dựa trên trạng thái hiện tại.

2. Mô hình

Thuật toán A3C được phát hành bởi nhóm DeepMind của Google vào năm 2016 và tạo nên cuộc cách mạng trong học củng cố, nhanh chóng vượt qua thuật toán DQN quen thuộc. Nó nhanh hơn, đơn giản hơn, mạnh mẽ hơn và có thể đạt được kết quả tốt hơn nhiều [18]. A3C có thể áp dụng trong cả không gian hành động liên tục và hành động rời rạc.

Thuật toán cho phép máy hoạt động một cách không đồng bộ - Asynchronous. Thay vì chỉ sử dụng một máy để cố gắng học chính sách tối ưu như Deep Q learning, trong A3C sử dụng nhiều máy tương tác với môi trường. Ta có nhiều máy tương tác với môi trường trong cùng một thời điểm, tương tác với môi trường giống nhau [18]. Ta gọi các máy này là 'worker', đồng thời cũng có một máy riêng biệt được gọi là 'global network'. Các 'worker' được huấn luyện song song với nhau và cập nhật vào 'global network' sau T bước. Mỗi worker sau T bước sẽ cập nhật trọng số vào 'global network', vì thế các bản cập nhật này không diễn ra đồng thời, và đó là nguyên nhân của sự không đồng bộ. Sau mỗi lần cập nhật, các 'worker' cập nhật lại trọng số của nó bằng cách sao chép lại trọng số của 'global network'. Thuật toán sử dụng hàm lợi thế - Advantage cho mạng nơ-ron học. Trong đó có 2 mạng là mạng Actor và Critic. Actor là một thuật toán Policy Gradient để xác định chọn hành động nào, Critic là thuật toán Q-learning sử dụng hàm lợi thế để đánh giá hành động mà mạng Actor đã chọn [18].

Advantage – lợi thế, được tính bằng hàm Advantage-function là hiệu của Q-function (state-action value function) và V-function (state value function) [18]. Q-function đánh giá hành



Hình 5. Thuật toán A3C.

động a trong một trạng thái *s* và V-function dùng để đánh giá trạng thái *s* trong không gian trạng thái *S*. Từ đó, Advantage-function sẽ cho ta biết mức độ cải thiện khi thực hiện hành động a so với giá trị mong đợi của trạng thái đó. Nói cách khác, hàm này là điểm phụ nhận được nếu thực hiện hành động a vượt quá với giá trị mong đợi của trạng thái đó. Nếu máy chọn đúng hành động, tức là $Q(s,a) > V(s)$, lợi thế sẽ là một giá trị dương, tuy nhiên, nếu là một hành động xấu: $Q(s,a) < V(s)$ thì lợi thế sẽ là một giá trị âm. Do đó, Advantage-function được sử dụng để cải thiện hành vi của máy [18].

$$A(s, a) = Q(s, a) – V(s) \quad (1)$$

Bởi vì không thể tính được giá trị Q trực tiếp trong A3C, nên ta có thể ước tính Q bằng cách sử dụng lợi nhuận chiết khấu (discounted return) R:

$$R = r_n + \gamma r_{n-1} + \gamma^2 r_{n-2} \quad (2)$$

Hàm lợi thế sẽ được ước tính bằng:

$$A(s, a) = R – V(s) \quad (3)$$

Khi đó, hàm mất mát giá trị tính bằng:

$$ValueLoss(L_v) = \Sigma\ (R – V(s))^2 \quad (4)$$

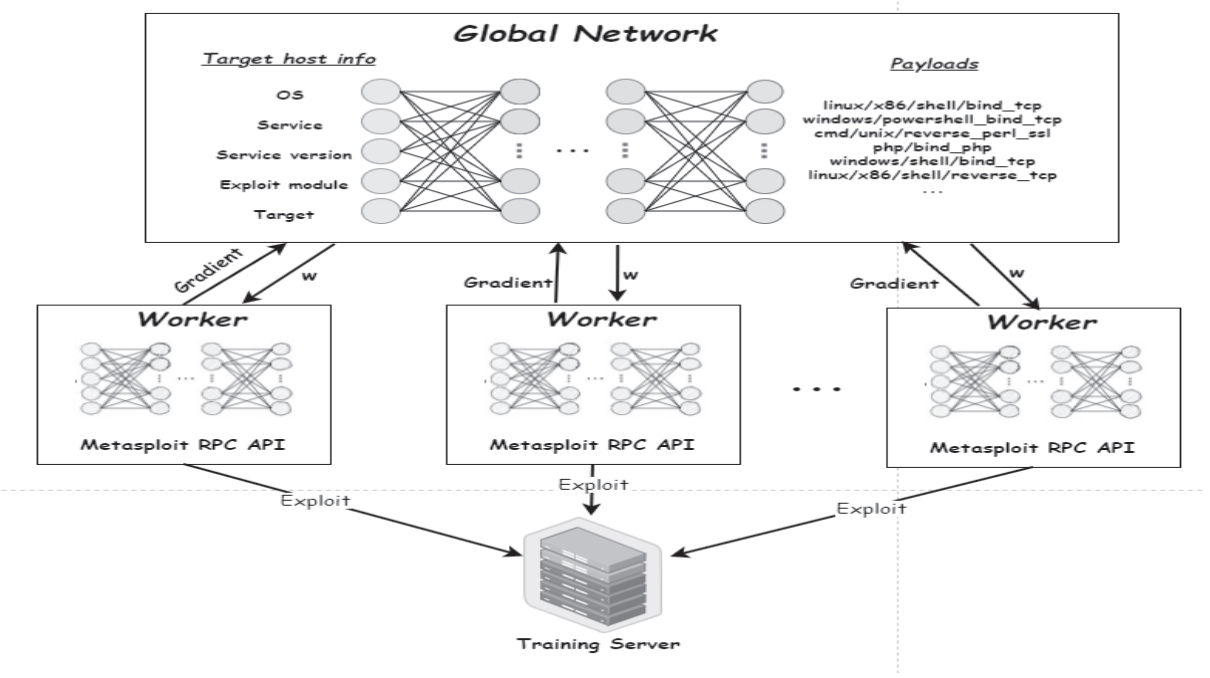
Hàm mất mát chính sách được tính theo công thức:

$$PolicyLoss(L_p) = -log(\pi(a \mid s)) * A(s) - \beta * H(\pi) \quad (5)$$

Trong đó, H(π) là Entropy, là thước đo mức độ lan tỏa của các xác suất, được sử dụng để đảm bảo thăm dò đầy đủ về chính sách. Khi Entropy cao, xác suất của mọi hành động sẽ như nhau, do đó, máy sẽ không chắc chắn về hành động nào sẽ được thực hiện, khi giá trị Entropy thấp, một hành động sẽ có xác suất cao hơn các hành động khác và máy có thể chọn hành động có xác suất cao này. Do đó, việc thêm Entropy vào hàm mất mát sẽ khuyến khích máy khám phá thêm và tránh bị mắc kẹt tại điểm tối ưu cục bộ.

Gộp 2 hàm mất mát ta được hàm mất mát tổng thể:

$$L = 0.5 * \Sigma(R - V(s))^2 - log(\pi(a \mid s)) * A(s) - \beta * H(\pi) \quad (6)$$



Hình 6. Thuật toán A3C trong tự động khai thác bằng Metasploit [19].

Trong bài toán này, để xây dựng mô hình A3C ta cần xác định một số yếu tố. Đầu tiên, môi trường mà máy tương tác là môi trường kết nối giữa Metasploit và máy chủ mục tiêu. Nhiệm vụ của máy là gửi module khai thác đến máy chủ mục tiêu thông qua Metasploit, làm cho việc khai thác thành công. Trạng thái trong bài toán này gồm 5 trường:

- Loại hệ điều hành (OS type).
- Dịch vụ (Service)
- Phiên bản dịch vụ (Service version)
- Module khai thác (Exploit module)
- Mục tiêu khai thác (target trong Metasploit)

Đây là các trường thông tin cần thiết để thực hiện khai thác một lỗ hổng bảo mật trên Metasploit.

Nhiệm vụ của mô hình A3C lúc này là đánh giá được mức độ tối ưu của các hành động (action) cho các trạng thái, tức là đánh giá mức độ tối ưu của các cặp trạng thái – hành động, từ đó xác định được thông tin để khai thác thành công.

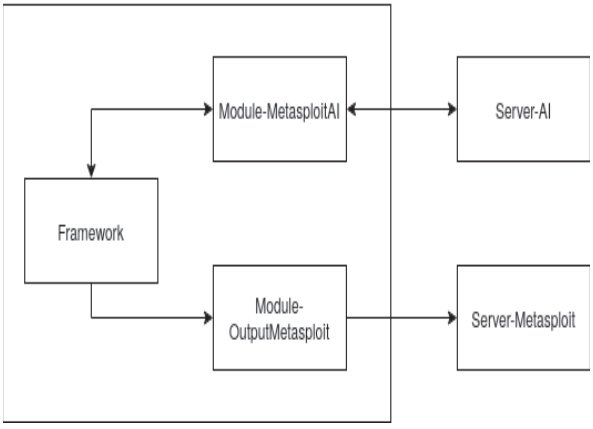
Các payload của Metasploit là các hành động nên số payload sẽ bằng số hành động của máy. Tuỳ các phiên bản Metasploit mà sẽ có số payload khác nhau. Tuy nhiên, có các module không sử dụng payload nên số hành động phải cộng thêm với 1 (ứng với "no payload").

Mạng A3C lấy đầu vào là trạng thái của máy, đầu ra là các hành động, vì thế ta có lớp đầu vào của mạng gồm 5 node tương ứng với 5 trường của trạng thái, lớp đầu ra sẽ có số node tương ứng bằng số hành động ($N_{payload}$ + 1).

Trong bài toán này, với mỗi một dịch vụ trinh sát được sẽ có thể có nhiều module trong Metasploit, với mỗi module lại có nhiều payload và target tương ứng. Vì vậy sẽ có rất nhiều cặp trạng thái – hành động xảy ra. Mô hình A3C sử dụng Advantage-function để đánh giá mức độ tốt của hành động (Payload) trong một trạng thái so với giá trị mong đợi, tức là đánh giá cặp trạng thái – hành động. Cặp trạng thái – hành động trong bài toán này chứa thông tin sử dụng để khai thác trong Metasploit (Module, Target, Payload). Do đó, mô hình A3C được sử dụng để đánh giá và module MetasploitAI chỉ lựa chọn những cặp trạng thái – hành động có điểm đánh giá cao nhất để khai thác mà không cần phải thử qua tất cả các trường hợp.

C. Tích hợp module AI trong Framework

Để thuận tiện cho quá trình kiểm thử, tìm kiếm lỗ hổng, module AI trong Framework được xây dựng dưới dạng client-server để quá trình thực thi được phân tán, dễ dàng triển khai trên hệ thống mạng.



Hình 7. Module kết hợp MetasploitAI với Framework.

Hai module Module-MetasploitAI và Module-OutputMetasploit được xây dựng để phục vụ quá trình khai thác đối với Framework. Server AI sẽ nhận các thông tin từ đối tượng thông qua các API, sau đó tiến hành tìm ra các payload có khả năng tồn tại lỗ hổng bảo mật nhất. Sau đó sẽ tiến hành tự động khai thác thông qua Metasploit RPC [22] từ đó xác định được lỗ hổng có thực sự tồn tại. Trong quá trình tự động khai thác có thể sử dụng các dữ liệu thu thập thông tin để hỗ trợ quá trình. Sau đó sẽ gửi kết quả về lại Framework. Server-Metasploit là một server nhận các yêu cầu thực thi từ Module-OutputMetasploit để tạo mới session và thực thi các câu lệnh theo yêu cầu.

IV. ĐÁNH GIÁ THỬ NGHIỆM HỆ THỐNG

1. Các mã khai thác đã xây dựng

Tuỳ vào từng đối tượng khác nhau, chúng tôi đã dựng các môi trường tồn tại các lỗ hổng để kiểm thử quá trình khai thác, triển khai trên môi trường mạng LAN và môi trường mạng Internet. Qua kiểm thử trên hệ thống: máy sử dụng hệ điều hành Kali Linux với cấu hình CPU 2.5 Ghz, RAM 4Gb, tiến hành khai thác với các lỗ hổng CVE thuộc các nền tảng khác nhau, chúng tôi thu được kết quả (Bảng 2) như sau:

BẢNG 2. BẢNG KẾT QUẢ KIỂM THỬ FRAMEWORK VỚI MỘT SỐ CVE

TT	Loại POC	POC thử nghiệm	Kết quả
1	window	CVE-2020-1472	Thành công
2	CMS	CVE-2019-9978	Thành công
3	apache	CVE-2019-0232	Thành công
4	git	CVE-2021-22192	Thành công
5	big-ip	CVE-2021-22986	Thành công
6	vmware	CVE-2021-21985	Thành công
7	exchange	CVE-2021-34473	Thành công
8	linux	CVE-2021-3156	Thành công

Kết quả cho thấy khả năng khai thác lỗ hổng của Framework có thể đáp ứng nhiều loại lỗ hổng khác nhau, trên nhiều nền tảng khác nhau. Việc kết hợp các script khai thác lỗ hổng vào Framework, giúp quá trình kiểm thử trở nên nhanh chóng mà vẫn đảm bảo tính chính xác. Ngoài ra, khi sử dụng các mã script trên nền tảng ngôn ngữ Python, quá trình thêm mới, chỉnh sửa sẽ đơn giản hơn, được hỗ trợ nhiều thông qua các hệ thống mã nguồn mở (một lượng lớn các mã khai thác được công bố được viết bằng Python - Github [25]). Ngôn ngữ Python cũng đáp ứng các kết nối giao tiếp SSH, TCP, HTTP, NRPC [24],... Vậy nên, ngôn ngữ Python đủ để đáp ứng cho quá trình khai thác lỗ hổng của Framework.

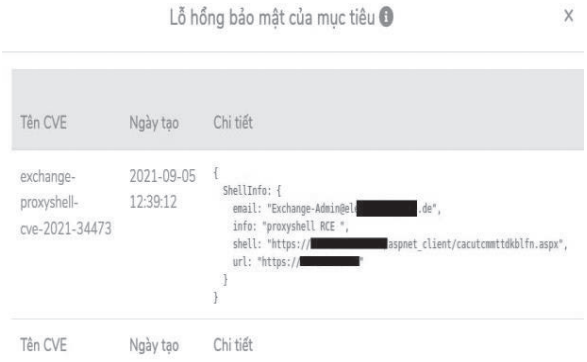
Để làm phong phú thêm khả năng tìm kiếm và khai thác lỗ hổng, chúng tôi đã xây dựng một số mã script của những lỗ hổng mới nhằm tăng khả năng tìm kiếm lỗ hổng của Framework (Bảng 3):

BẢNG 3. MỘT SỐ SCRIPT KIỂM THỬ CVE TRONG FRAMEWORK

TT	Loại POC	Số lượng	Đánh giá		
			RCE	Auth	Leo Thang
1	window	6	4	1	2
2	CMS	8	8	0	0

3	apache	5	5	0	0
4	git	2	2	0	0
5	big-ip	2	2	0	0
6	vmware	3	3	0	0
7	exchange	4	4	1	0
8	linux	2	1	0	1
	Tổng	32	29	2	3

Để kiểm thử khả năng khai thác đối với những mục tiêu ngoài phòng thí nghiệm, chúng tôi đã thử nghiệm Framework dò quét hệ thống thông qua mạng Internet và thu được kết quả:



Hình 8. Lỗ hổng tìm kiếm được đối với một mục tiêu mail exchange.

Quá trình kiểm thử với hệ thống mail exchange đã tìm ra lỗ hổng bảo mật, tiến hành khai thác thử nghiệm đảm bảo không còn dương tính giả. Kết quả thử nghiệm có thể thực thi lỗ hổng ngay trên server exchange.

2. Kết quả thu thập thông tin

Framework cho phép lựa chọn sử dụng nhiều công cụ dò quét thông tin của hệ thống như Nmap, Nuclei, Wappalyzer,…, đưa ra được nhiều thông tin đa dạng và chính xác hơn. Chúng tôi tích hợp công cụ Nmap để dò quét các cổng và dịch vụ, đồng thời sử dụng Scrapy tìm ra các đường dẫn dịch vụ web, sau đó gửi request và phân tích response bằng kỹ thuật so khớp chuỗi để tìm ra các thông tin về ứng dụng web (kỹ thuật được sử dụng trong công cụ DeepExploit [19]).

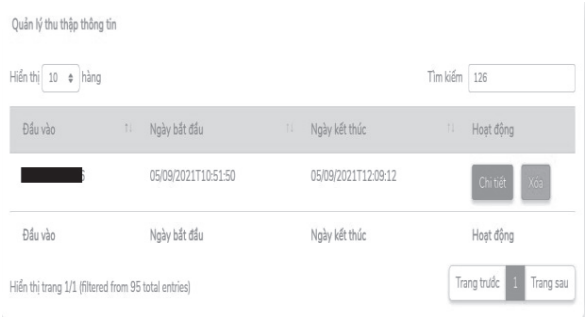
Để thử nghiệm độ chính xác của quá trình thu thập thông tin nhóm đã tiến hành so sánh độ chính xác kết quả thu thập thông tin mà DeepExploit với Framework cho kết quả như Bảng 4.

BẢNG 4. KẾT QUẢ THU THẬP THÔNG TIN CỦA FRAMEWORK

	DeepExploit[19]	**Framework**
Công cụ	Nmap, scapy Cài đặt trực tiếp	Nmap, WebDetect, Wappalyzer, Nuclei Sử dụng thông qua docker
Môi trường	Metasploitable2	Metasploitable2
Kết quả		
Cổng mở	Tổng số: 13/33 Tỉ lệ chính xác: 39.39%	Tổng số: 13/13 Tỉ lệ chính xác: 100%
Dịch vụ trên cổng	Số dịch vụ: 13/33 Tỉ lệ đúng: 39.39%	Số dịch vụ: 13/13 Tỉ lệ đúng: 100%
Công nghệ sử dụng	Tổng số: 10/16 Có version: 3 Tỉ lệ chính xác: 62.50%	Tổng số 11/12 Có version: 5 Tỉ lệ chính xác: 91.67%
Lỗ hổng bảo mật	Tổng số: 0	Tổng số: 22/27 lỗ hổng chính xác Tỉ lệ chính xác: 81.84%
Hệ điều hành	Tổng số: 1 Tỉ lệ chính xác: 100%	Tổng số: 1 Tỉ lệ chính xác: 100%
Thời gian thực hiện	262s	550s
Nhận xét	Thời gian thu thập nhanh, tốn ít bộ nhớ, kết quả thu được tỉ lệ chính xác không được cao, xuất hiện nhiều thông tin sai	Thời gian thu thập khá lâu, chạy tốn bộ nhớ. Kết quả thu được bảo đảm, tỉ lệ chính xác cao hơn.

Kết quả thu thập thông tin có kết quả chính xác cao(100%) thể hiện kết quả trả về có tỉ lệ chính xác cao hơn, sàng lọc được các cảnh báo dương tính giả (false positive).

Tuy cho kết quả với độ chính xác cao nhưng đồng nghĩa sẽ mất nhiều thời gian hơn, số request được gửi đi nhiều hơn. Trong quá trình thu thập thông tin, dò quét hệ thống đối với mục tiêu là server mail exchange, có sử dụng firewall thông qua môi trường Internet. Kết quả thu được quá trình mất tới hơn 120 phút mới ra được kết quả.



Hình 9. Thời gian dò quét lỗ hổng.

3. Kết quả thử nghiệm mô hình AI

Mô hình A3C được huấn luyện trong môi trường máy ảo Metasploitable 1 [20]. Thông tin thiết lập cấu hình cho máy ảo Metasploitable 1 được thể hiện trong Hình 10.

```
Discovery:
----------
ftp             21/tcp          220 ProFTPD 1.3.1 Server (Debian) [::ffff:10.0.0.48]
ssh             22/tcp          SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1
telnet          23/tcp          Ubuntu 8.04\x0metasploitable login:
smtp            25/tcp          220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
dns             53/tcp
dns             53/udp          BIND 9.4.2
http            80/tcp          Apache/2.2.8 (Ubuntu) PHP/5.2.4-2ubuntu5.10 with Suhosin-Patch
smb             139/tcp
smb             445/tcp         Unix Samba 3.0.20-Debian (language: Unknown)
mysql           3306/tcp        5.0.51a-3ubuntu5
distccd         3632/tcp
postgres        5432/tcp        8.3.8
http            8180/tcp        Apache-Coyote/1.1

Bruteforce:
-----------
smb         Anonymous
ssh         6 sessions
telnet      6 sessions
bind        n/a
apache      2 web apps (twiki and tikiwik)
postgres    db compromise (postgres:postgres)
mysql       db compromise (root:root)
tomcat 5.5  shelled (tomcat:tomcat)
```

Hình 10. Máy ảo Metasploitable 1.

Xây dựng máy chủ mục tiêu hệ điều hành Linux, có cài đặt các phiên bản dịch vụ chứa lỗ hổng bảo mật để thử nghiệm.

Sau khi thử nghiệm với module trí tuệ nhân tạo tự động khai thác, đạt được kết quả như sau:

BẢNG 5. KẾT QUẢ THỬ NGHIỆM MODULE AI

STT	Lỗ hổng	Kết quả	Ghi chú
1	CVE 2010-4221	Thành công	
2	CVE 2017-7722	Thành công	
3	CVE 2011-4862	Thất bại	
4	CVE 2005-1099	Thành công	
5	CVE 2013-2028	Thành công	
6	Apache Tomcat Manager Application Deployer Authenticated Code Execution	Thất bại	Dự đoán đúng module và payload nhưng khai thác không thành công.
7	Ra1NX PHP Bot PubCall Authentication Bypass Remote Code Execution	Thất bại	
8	CVE 2007-5423	Thất bại	
9	Samba “username map script” Command Execution	Thất bại	
10	Oracle MySQL UDF Payload Execution	Thành công	
11	PostgreSQL for Linux Payload Execution	Thất bại	
12	CVE 2019-9193	Thành công	
13	CVE 2017-12617	Thất bại	Dự đoán đúng module và payload nhưng khai thác không thành công.

Module đã có thể bước đầu phát hiện và khai thác được một số lỗ hổng bảo mật hoàn toàn tự động, tuy nhiên hiệu quả và độ chính xác còn chưa cao, lý do là môi trường dùng để huấn luyện cho mô hình AI còn ít và hạn chế. Trong thực nghiệm, chúng tôi sử dụng máy chủ Metasploitable 1 để huấn luyện nên mô hình học còn chưa ổn định, chưa thực sự có hiệu quả cao với các lỗ hổng bảo mật mới. Bên cạnh đó, cũng có một số trường hợp module gửi đúng các options của lỗ hổng (exploit module, payload, rhost, rport,…) đến Metasploit RPC API, tuy nhiên lại thực hiện khai thác không thành công do những biến môi trường thay đổi. Chúng tôi đang cố gắng khắc phục và cải thiện mô hình để có được kết quả tốt hơn trong tương lai.

KẾT LUẬN

Bài báo đã giới thiệu về Framework tự động hỗ trợ đánh giá/khai thác lỗ hổng bảo mật của các hệ thống thông tin. Framework của chúng tôi có tính mở cao, cho phép bổ sung nhiều công cụ trinh sát, dò quét hệ thống để thu thập được thông tin. Framework cũng đồng thời tích hợp khả năng mở rộng, có thêm các Script, module khai thác mới. Dựa trên môi trường Metasploit, chúng tôi đã xây dựng thành công module có khả năng dự đoán các module khai thác và payload tương ứng, tự động sử dụng để khai thác lỗ hổng bảo mật của hệ thống.

Framework sẽ thuận tiện triển khai hướng tới việc sử dụng rộng rãi công cụ đánh giá, kiểm thử hệ thống thông tin. Những cán bộ ở cơ sở không cần có kiến thức chuyên gia cũng có thể dễ dàng vận hành hệ thống để đánh giá/kiểm thử khai thác lỗ hổng bảo mật sau khi hệ thống được cập nhật các module khai thác mới do các chuyên gia ở trung tâm bổ sung.

Tuy nhiên, Framework vẫn còn nhiều hạn chế như: dò quét thông tin mất tương đối nhiều thời gian, độ chính xác còn hạn chế khi thông qua tường lửa, chưa phát hiện được các bẫy honeypot. Trong thời gian tới, chúng tôi sẽ tiếp tục mở rộng các chức năng của hệ thống, xây dựng kho dữ liệu chứa các mã khai thác cập nhật trên nhiều nền tảng, nhiều phiên bản khác nhau để hỗ trợ việc đánh giá, khai thác lỗ hổng bảo mật được hiệu quả hơn.

TÀI LIỆU THAM KHẢO

[1] Mujahid Shah, Sheeraz Ahmed, Khalid Saeed, Muhammad Junaid, Hamayun Khan, Ata-ur-rehman (2019), "Penetration Testing Active Reconnaissance Phase – Optimized Port Scanning With Nmap Tool".

[2] Angel Rajan, Emre Erturk (2017), "Web Vulnerability Scanners: A case Study".

[3] Sudhanshu Raj, Navpreet Kaur Walia (2020), "A Study on Metasploit Framework: A Pen-Testing Tool".

[4] Wappalyzer - Identify technologies on websites https://www.wappalyzer.com/ [Accessed: September, 22, 2021].

[5] Rengine wiki for developer https://rengine.wiki/ [Accessed: September, 14, 2021].

[6] Vulscan github https://github.com/vulscanteam/vulscan [Accessed: September, 05, 2021].

[7] Vulnerability Scanning with Nexpose https://docs.rapid7.com/metasploit/vulnerability-scanning-with-nexpose [Accessed: September, 14, 2021].

[8] Stanislav Dashevskyi, Daniel Ricardo dos Santos, Fabio Massacci, and Antonino Sabetta (2017), "TestREx: a Framework for Repeatable Exploits", Journal reference: Int. J. Software Tools for Technology Transfer, 2017.

[9] Python Multithreaded Programming – Tutorials Point, https://www.tutorialspoint.com/python/python_multithreading.htm [Accessed: September, 1, 2021].

[10] Sachchidanand Singh; Nirmala Singh (July, 2016), "Containers & Docker: Emerging roles & future of Cloud technology", IEEE 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT).

[11] Nmap Docker image 7.92 của Instrumentisto https://hub.docker.com/r/instrumentisto/nmap [Accessed: September, 14, 2021].

[12] Dirsearch - Web path discovery image, Maurosoria https://github.com/maurosoria/dirsearch [Accessed: September, 14, 2021].

[13] Nuclei - Fast and customisable vulnerability scanner based on simple YAML based DSL github https://github.com/projectdiscovery/nuclei [Accessed: September, 14, 2021].

[14] Wappalyzer https://hub.docker.com/r/wappalyzer/cli [Accessed: September, 14, 2021].

[15] J¨urgen Cito, Vincenzo Ferme, Harald C. Gall (2016), "Using docker Containers to Improve Reproducibility in Software and web Engineering Research".

[16] Jonathon Schwart (2018), "Autonomous Penetration Testing using Reinforcement Learning", The University of Queensland, Australia.

[17] Julien Vitay, "Deep Reinforcement Learning", https://julien-vitay.net/deeprl/, [Accessed: September, 14, 2021].

[18] Sudharsan Ravichandiran (2018), "Hands-On Reinforcement Learning with Python", Published by Packt Publishing.

[19] Isao Takaesu (2018), "DeepExploit", https://github.com/13o-bbr-bbq/machine_learning_security/tree/master/DeepExploit [Accessed: June, 24, 2021].

[20] Metasploit (2010), "Introducing Metasploitable", http://blog.metasploit.com/2010/05/introducing-metasploitable.html, [Accessed: September, 14, 2021].

[21] Eashan Kaushik (2020), "Asynchronous Advantage Actor Critic with Random Exploration Exploitation (A3C-REE)", International Research Journal of Engineering and Technology (IRJET).

[22] Rapid7 (2016), "Metasploit Pro RPC API Guide".

[23] SpiderLabs, "msgrpc", https://github.com/SpiderLabs/msfrpc, [Accessed: September, 14, 2021].

[24] Dirkjanm, PoC for Zerologon - all research credits go to Tom Tervoort of Secura https://github.com/dirkjanm/CVE-2020-1472 [Accessed: September, 2, 2021].

[25] A small place to discover languages in GitHub https://madnight.github.io/githut/#/pull_requests/2021/2 [Accessed: September, 19, 2021].

SƠ LƯỢC VỀ TÁC GIẢ

Nguyễn Mạnh Thiên

Đơn vị công tác: Học viện Kỹ thuật Quân sự

Email: nguyenmanhthien98@gmail.com

Quá trình đào tạo: Học viên năm thứ 5 – Học viện Học viện Kỹ thuật Quân sự

Hướng nghiên cứu hiện nay: An toàn ứng dụng web.

Phạm Đăng Khoa

Đơn vị công tác: Học viện Kỹ thuật Quân sự

Email: Khoa0010098@gmail.com

Quá trình đào tạo: Học viên năm thứ 5 – Học viện Học viện Kỹ thuật Quân sự

Hướng nghiên cứu hiện nay: An toàn ứng dụng web.

Nguyễn Đức Vượng

Đơn vị công tác: Học viện Kỹ thuật Quân sự

Email: ndvuong98@gmail.com

Quá trình đào tạo: Học viên năm thứ 5 – Học viện Học viện Kỹ thuật Quân sự

Hướng nghiên cứu hiện nay: Trí tuệ nhân tạo.

Nguyễn Việt Hùng

Đơn vị công tác: Học viện Kỹ thuật Quân sự

Email: hungnv@lqdtu.edu.vn

Quá trình đào tạo: Tốt nghiệp Cử nhân (2006), Thạc sĩ (2008) và Tiến sĩ (2012) tại Đại học Vật lý Kỹ thuật Mát-x-cơ-va.

Hướng nghiên cứu hiện nay: An toàn không gian mạng, trí tuệ nhân tạo.